GUESS & GUIDE: GRADIENT-FREE ZERO-SHOT DIFFUSION GUIDANCE

Abduragim Shtanchaev
MBZUAI, Abu Dhabi, United Arab Emirates
abduragim.shtanchaev@mbzuai.ac.ae

Albina Ilina
MBZUAI, Abu Dhabi, United Arab Emirates

Yazid Janati
MBZUAI, Abu Dhabi, United Arab Emirates
Institute of Foundation Models

Arip Asadulaev
MBZUAI, Abu Dhabi, United Arab Emirates

Martin Takáč
MBZUAI, Abu Dhabi, United Arab Emirates

Eric Moulines
MBZUAI, Abu Dhabi, United Arab Emirates
EPITA, Paris, France

ABSTRACT

Pretrained diffusion models serve as effective priors for Bayesian inverse problems. These priors enable zero-shot generation by sampling from the conditional distribution, which avoids the need for task-specific retraining. However, a major limitation of existing methods is their reliance on surrogate likelihoods that require vector-Jacobian products at each denoising step, creating a substantial computational burden. To address this, we introduce a lightweight likelihood surrogate that eliminates the need to calculate gradients through the denoiser network. This enables us to handle diverse inverse problems without backpropagation overhead. Experiments confirm that using our method, the inference cost drops dramatically. At the same time, our approach delivers the highest results in multiple tasks. Broadly speaking, we propose the fastest and *Pareto-optimal* method for Bayesian inverse problems.

1 INTRODUCTION

Zero-shot generative modeling refers to leveraging a pre-trained generative model of the data distribution $p(\mathbf{x})$ and adapting it *without task-specific retraining* to solve downstream tasks it has never seen during training Moufad et al. (2025). In this setting, the only new ingredient at test time is a description of the task (e.g., a measurement operator or constraint), and the generative prior is used to produce realistic solutions that satisfy these new conditions.

A convenient probabilistic formalization of many such zero-shot tasks is through *Bayesian inverse problems*. Inverse problems arise when a signal of interest must be recovered from incomplete, corrupted, or indirect observations. Formally, this involves retrieving an original signal $\mathbf{x}$ from a measurement $\mathbf{y} = \mathcal{A}(\mathbf{x}) + n$, where $\mathcal{A}$ is a degradation operator (such as a blur kernel or masking) and n is noise. Because information is usually lost during this process, the problem is ill-posed and requires strong prior knowledge to solve.

The Bayesian framework offers a principled approach to these challenges by combining a data likelihood with a learned or structured prior distribution. A prominent strategy is to employ a strong generative prior at inference time to solve arbitrary inverse problems. These methods function as training-free posterior sampling frameworks, enabling plug-and-play adaptation across diverse observation models. In this setup, the diffusion model Sohl-Dickstein et al. (2015); Ho et al. (2020); Song & Ermon (2019); Song et al. (2020b) serves as a gradient field that iteratively *denoises* the candidate solution, guiding it toward the manifold of realistic data while ensuring it remains consistent with the observed measurements. Recent methods achieved impressive results in such inverse problems as inpainting, deblurring, super-resolution, and source separation Lugmayr et al. (2022); Kawar et al. (2021; 2022); Saharia et al. (2022)

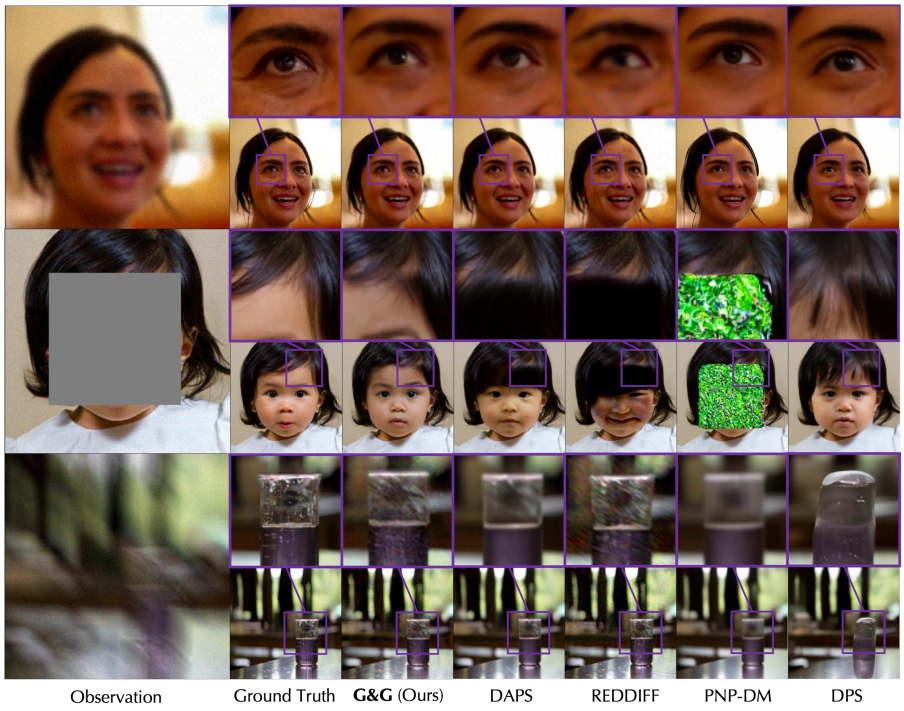

Figure 1: **FFHQ and ImageNet Qualitative results.** Comparison against baseline methods over different tasks: Gaussian Deblurring and Center Inpainting on `FFHQ` (*top and middle rows respectively*), and Motion Deblurring on `ImageNet` (*bottom row*). Guess and Guide (ours) achieves more accurate and detailed restoration compared to existing approaches. See Appendix H/J for more samples.

At the core of diffusion-based posterior sampling is a sequential denoising process that approximates intermediate posterior distributions. These distributions combine the learned prior with progressively refined likelihoods at each noise level. However, the exact posterior denoiser involves an intractable likelihood gradient, leading most methods to rely on approximations that *require computing vector-Jacobian products through the denoiser* network at every step Chung et al. (2022a); Song et al. (2023b). While effective, this introduces substantial computational and memory overhead, limiting scalability and practical deployment.

To avoid these problems, we present the **Guess and Guide (G&G)** framework for *backpropagation-free* zero-shot generative modeling. Our approach introduces a lightweight optimization procedure for Bayesian Inverse Inference that avoids costly vector-Jacobian products at each sampling step. First, an **initial guess phase** rapidly obtains a high-quality estimate through iterative optimization. Next, a **guided denoising phase** refines this estimate, using a flexible custom scheduling strategy. We show that our simple approach significantly reduces memory usage and inference time. Importantly, our method achieves high reconstruction quality across diverse tasks, including super-resolution, inpainting, deblurring, phase retrieval, and HDR reconstruction. Together, we propose a fast, general, and gradient-free framework to solve diffusion-based inverse problems.

2 Background

Denoising diffusion models (DDMs) Sohl-Dickstein et al. (2015); Song & Ermon (2019); Ho et al. (2020) generate samples from a target distribution p_0 by constructing a distribution interpolation path $\{p_t\}_{t\in[0,1]}$ that connects the data distribution p_0 to the standard Gaussian base distribution $p_1 := \mathcal{N}(0, I_d)$. This interpolation is defined through a noising process where each intermediate distribution $p_t = \text{Law}(X_t)$ arises from the mixture

$$X_t = \alpha_t X_0 + \sigma_t X_1, \quad X_0 \sim p_0, \quad X_1 \sim p_1, \tag{1}$$

with independent random variables $X_0 \sim p_0$ and $X_1 \sim p_1$. The coefficients $(\alpha_t)_{t\in[0,1]}$ and $(\sigma_t)_{t\in[0,1]}$ are deterministic schedules that satisfy monotonicity constraints (non-increasing and

non-decreasing, respectively) and boundary conditions $(\alpha_0, \sigma_0) := (1, 0)$ and $(\alpha_1, \sigma_1) := (0, 1)$. Common instantiations include the *variance-preserving schedule* where $\alpha_t^2 + \sigma_t^2 = 1$ (Ho et al., 2020; Dhariwal & Nichol, 2021), and the *linear schedule* with $(\alpha_t, \sigma_t) = (1 - t, t)$ (Lipman et al., 2022; Esser et al., 2024; Gao et al., 2024).

To sample from p_0, DDIM Song et al. (2020a) starts from a sample drawn from p_1 at time $t_K = 1$. It then performs a sequence of deterministic reverse transitions, progressively denoising each sample $\hat{X}_{t_{k+1}}$ into a less noisy sample $\hat{X}_{t_k}$, until reaching the clean distribution at $t_0 = 0$. The DDIM transition from t_{k+1} to t_k is defined as

$$\hat{X}_{t_k} = \alpha_{t_k} \hat{\mathbf{x}}_0^\theta(\hat{X}_{t_{k+1}}, t_{k+1}) + \sigma_{t_k} \hat{\mathbf{x}}_1^\theta(\hat{X}_{t_{k+1}}, t_{k+1}),$$

where $\hat{\mathbf{x}}_0^\theta(\cdot, t_{k+1})$ and $\hat{\mathbf{x}}_1^\theta(\cdot, t_{k+1})$ are respectively neural network approximations of the denoiser $\mathbb{E}[X_0|X_t]$ and noise prediction $\mathbb{E}[X_1|X_t]$ under the model (1). Since it holds, following the same model, that $\mathbb{E}[X_1|X_t] = (X_t - \alpha_t \mathbb{E}[X_0|X_t])/\sigma_t$, only the denoisers $(\hat{\mathbf{x}}_0^\theta(\cdot, t))_{t\in[0,1]}$ are trained by minimizing an L_2 denoising loss and $\hat{\mathbf{x}}_1^\theta(\mathbf{x}, t)$ is set to $(\mathbf{x} - \alpha_t \hat{\mathbf{x}}_0^\theta(\mathbf{x}, t))/\sigma_t$. The DDIM framework also allows for stochastic updates. In this case,

$$\hat{X}_{t_k} \quad = \quad \alpha_{t_k} \hat{\mathbf{x}}_0^\theta(\hat{X}_{t_{k+1}}, t_{k+1}) \quad + \quad \sqrt{\sigma_{t_k}^2 - \eta_{t_k}^2}\, \hat{\mathbf{x}}_1^\theta(\hat{X}_{t_{k+1}}, t_{k+1}) \quad + \quad \eta_{t_k} W_k, \quad (2)$$

where $W_k \sim \mathcal{N}(0, \mathbf{I})$ and $0 \leq \eta_{t_k} \leq \sigma_{t_k}$. In what follows we write $p^\theta_{t_k|t_{k+1}}(\cdot|\mathbf{x}_{t_{k+1}})$ to refer to the associated conditional density (the dependence on η is left implicit).

In practice, most diffusion models are implemented in the latent space Rombach et al. (2022); Esser et al. (2024), assuming an encoder-decoder pair $(\mathcal{E}, \mathcal{D})$. The model is trained on $Z_0 = \mathcal{E}(X_0)$ where $X_0 \sim p_0$. At inference, reverse diffusion occurs in the latent space, and the final sample is obtained via decoding. We adopt this framework, assuming p_0 and the process $(Z_t)_t$ (analogous to (1)) are defined in the latent space. However, our approach applies equally to pixel space by treating $\mathcal{E}$ and $\mathcal{D}$ as identity mappings.

Inverse problems. In this work, we investigate a broad class of image restoration and reconstruction problems formulated as inverse tasks. We consider an unknown clean image $\mathbf{x}_* \in \mathbb{R}^d$ that is observed through a (possibly nonlinear) measurement process described by an operator $\mathcal{A} : \mathbb{R}^d \to \mathbb{R}^m$. The observed data are modeled as

$$\mathbf{y} = \mathcal{A}(\mathbf{x}_*) + \sigma_y W, \qquad W \sim \mathcal{N}(0, \mathbf{I}_m), \quad (3)$$

where $\sigma_y > 0$ denotes the noise level. The task is to recover a reconstruction $\hat{\mathbf{x}}$ that explains the observation $(\mathcal{A}(\hat{\mathbf{x}}) \approx \mathbf{y})$ while remaining consistent with the statistical properties of natural images. Within a Bayesian framework, the prior distribution p_0 captures encoded natural-image statistics, and the measurement process is modeled by the Gaussian likelihood $\mathcal{N}(\mathbf{y}; \mathcal{A}(\mathbf{x}), \sigma_y^2 \mathbf{I}_m)$, where σ_y governs the trade-off between measurement fidelity and regularization. The posterior distribution over encoded images in the latent space is then given by $\pi_0(\mathbf{z}|\mathbf{y}) \propto \ell_0(\mathbf{y}|\mathbf{z})\, p_0(\mathbf{z})$, where $\ell_0(\mathbf{y}|\mathbf{z}) = \mathcal{N}(\mathbf{y}; \mathcal{A}(\mathcal{D}(\mathbf{z})), \sigma_y^2 \mathbf{I}_m)$. This serves as the target distribution for inference. This probabilistic formulation offers a unified perspective on various classical image restoration problems, including inpainting, deblurring, super-resolution, phase retrieval, and High Dynamic Range (HDR) reconstruction, each characterized by a specific choice of the operator $\mathcal{A}$ and noise level σ_y.

Zero-shot posterior sampling. A powerful feature of diffusion models is their ability to solve inverse problems at inference time without additional training, as demonstrated in the seminal works (Song & Ermon, 2019; Kadkhodaie & Simoncelli, 2020; Song et al., 2020b; Kawar et al., 2021). The key insight is to modify the reverse sampling process on-the-fly using guidance terms that steer generated samples to satisfy observational constraints encoded in a likelihood $\ell_0(\mathbf{y}|\cdot)$ while at the same time remaining plausible under the prior p_0. This approach treats inverse problems through the Bayesian lens presented in the previous section; we seek to perform approximate sampling from a posterior distribution that conditions on the observation $\mathbf{y}$. Achieving this with a diffusion model requires having access to the denoiser conditioned on the additional observation $\mathbf{y}$. Following (Daras et al., 2024, Equations 2.15, 2.17) it writes as

$$\mathbb{E}[Z_0|Z_t, \mathbf{y}] = \mathbb{E}[Z_0|Z_t] + \alpha^{-1} \sigma_t^2 \nabla_{\mathbf{z}_t} \log \ell_t(\mathbf{y}|Z_t), \quad (4)$$

with $\ell_t(\mathbf{y}|\mathbf{z}) = \mathbb{E}[\ell_0(\mathbf{y}|Z_0)|Z_t = \mathbf{z}]$. The correction term added to the denoiser is referred to as *guidance*. Since we already have access to the pretrained prior denoiser $\hat{\mathbf{x}}_0^\theta(\cdot, t)$ approximating the first term, estimating the posterior denoiser (4) reduces to approximating the score $\nabla_{\mathbf{z}_t} \log \ell_t(\mathbf{y}|Z_t)$, which is generally intractable. Chung et al. (2022a) considers the approximation $\ell_t(\mathbf{y}|\mathbf{z}) \approx \ell_0(\mathbf{y}|\hat{\mathbf{x}}_0(\mathbf{z}, t))$ which ultimately consists in swapping expectation and likelihood. This approximation is usually referred to as the Diffusion Posterior Sampling (DPS) approximation Chung et al. (2022a); Song et al. (2023b).

3 METHOD

Computational bottleneck of guidance methods. Standard guidance methods that rely on the DPS approximation encounter a significant computational bottleneck. In order to use this approximation to estimate the observation conditioned denoiser (4), we need to compute the gradient $\nabla_{\mathbf{z}_t} \log \ell_0(\mathbf{y}|\hat{\mathbf{x}}_0(\mathbf{z}_t, t))$ at every reverse diffusion step. This operation involves repeated vector–Jacobian products (VJPs), which are both memory-intensive and computationally expensive, substantially increasing inference time and limiting scalability to high-resolution images.

To overcome this limitation, our goal is to design a zero-shot guidance framework that preserves the task-agnostic flexibility of existing approaches while being computationally and memory efficient. In particular, we aim to remove the need for backpropagation through the denoiser and decoder altogether, replacing it with a lightweight optimization procedure applied at selected diffusion steps.

Our idea. We initialize our algorithm by first seeking an approximate sample $\mathbf{z}_{t_*}^{\mathbf{y}}$ at a timestep $t_* \ll 1$, from the marginal distribution of $Z_{t_*}^{\mathbf{y}} = \alpha_{t_*} Z_0^{\mathbf{y}} + \sigma_{t_*} Z_1$, where $Z_0^{\mathbf{y}} \sim \pi_0(\cdot \mid \mathbf{y})$. This corresponds to the interpolation (1) but initialized at the posterior distribution. Such initialization allows the backward diffusion process to start directly from timestep t_*, eliminating the need to begin from the base distribution p_1 and thereby accelerating inference. Naturally, our algorithm proceeds in two phases. The first phase aims to obtain a sufficiently accurate approximate sample at step t_*, while the second uses this estimate to initialize the reverse process toward the posterior distribution. As data-fidelity computation (guidance signal) is performed in pixel space, we avoid backpropagation through the denoiser and decoder, reducing both memory usage and computational cost.

Algorithm 1: Guess-and-Guide

Require: Observation $\mathbf{y}$, forward operator $\mathcal{A}$, encoder $\mathcal{E}$, decoder $\mathcal{D}$, denoiser $\hat{\mathbf{x}}_0^\theta(\cdot,\cdot)$, timesteps $\{t_1, \dots, t_M\}$, warm-start iterations N, DDIM steps n

Phase 1: Initial Guess
$\mathbf{z}_{t_*}^0 \sim \mathcal{N}(\alpha_{t_*}\mathcal{E}(\mathbf{y}), \sigma_{t_*}^2 \mathbf{I})$
for $k = 0, \dots, N-1$ **do**
  $\hat{\mathbf{z}}_0^k \leftarrow \hat{\mathbf{x}}_0^\theta(\mathbf{z}_{t_*}^k, t_*)$ ▷ Predict clean latent
  $\epsilon^k \leftarrow (\mathbf{z}_{t_*}^k - \alpha_{t_*}\hat{\mathbf{z}}_0^k)/\sigma_{t_*}$
  $\mathbf{x}_0^k \leftarrow \mathsf{Optimize}(\mathbf{x} \mapsto \|\mathbf{y} - \mathcal{A}(\mathbf{x})\|^2, \mathbf{x}_0 = \mathcal{D}(\hat{\mathbf{z}}_0^k))$
  $\boldsymbol{\mu}^k \leftarrow \alpha_{t_*}\mathcal{E}(\mathbf{x}_0^k) + \sigma_{t_*}\hat{\mathbf{z}}_0^k + \sigma_{t_*}\epsilon^k$
  $\sigma^k \leftarrow \sigma_{t_*}^2 \mathbf{I}$
  $\mathbf{z}_{t_*}^{k+1} \sim \mathcal{N}(\alpha_{t_*}\boldsymbol{\mu}^k, \sigma_{t_*}^2\sigma^k)$ ▷ Re-noise
$\mathbf{z}_{t_M}^{\mathbf{y}} \leftarrow \mathbf{z}_{t_*}^N$
Phase 2: Guided denoising
for $k = M-1, \dots, 1$ **do**
  $\tilde{\mathbf{z}}_{t_k} \sim p^\theta_{t_k|t_{k+1}}(\cdot|\mathbf{z}_{t_{k+1}}^{\mathbf{y}})$
  $\tilde{\mathbf{z}}_0 \leftarrow \hat{\mathbf{x}}_0^\theta(\tilde{\mathbf{z}}_{t_k}, t_k)$
  $\epsilon_{t_k} \leftarrow (\tilde{\mathbf{z}}_{t_k} - \alpha_{t_k}\tilde{\mathbf{z}}_0)/\sigma_{t_k}$
  $\tilde{\mathbf{x}}_0 \leftarrow \mathcal{D}(\tilde{\mathbf{z}}_0)$
  $\mathbf{x}_0^* \leftarrow \mathsf{Optimize}\big(\mathbf{x} \mapsto \|\mathbf{y} - \mathcal{A}(\mathbf{x})\|^2 + \lambda\|\mathbf{x} - \tilde{\mathbf{x}}_0\|^2, \mathbf{x}_0 = \tilde{\mathbf{x}}_0\big)$
  $\boldsymbol{\mu}_{t_{k+1}} \leftarrow \alpha_{t_{k+1}}\mathcal{E}(\mathbf{x}_0^*) + \sigma_{t_{k+1}}\alpha_{t_{k+1}}\epsilon_{t_k}$
  $\sigma'_{t_{k+1}} \leftarrow \sigma_{t_{k+1}}^2 \mathbf{I}$
  $\tilde{\mathbf{z}}_{t_{k+1}} \sim \mathcal{N}(\boldsymbol{\mu}_{t_{k+1}}, \sigma_{t_{k+1}}^2\sigma'_{t_{k+1}})$
  $\mathbf{z}_{t_k}^{\mathbf{y}} \leftarrow \mathsf{DDIM}(\tilde{\mathbf{z}}_{t_{k+1}}, t_{k+1} \xrightarrow{n} t_k)$ ▷ DDIM in n steps
return $\hat{\mathbf{x}} = \mathcal{D}(\hat{\mathbf{x}}_0^\theta(\mathbf{z}_{t_1}^{\mathbf{y}}, 0))$

Phase 1: warm start. The objective of the first phase is to obtain a high-quality initial estimate at a fixed timestep $t_* \ll 1$ representing a moderately noisy state.

We begin by constructing a noisy version of the observation at timestep t_*. For latent diffusion models Rombach et al. (2022), we first encode the observation through the encoder, yielding $\mathcal{E}(\mathbf{y})$. The initial noisy latent is then $\mathbf{z}_{t_*}^0 \sim \mathcal{N}(\alpha_{t_*}\mathcal{E}(\mathbf{y}), \sigma_{t_*}^2 \mathbf{I}_d)$. Starting from this sample, we perform N iterations of prediction, optimization, and re-noising. Given the current noisy state at iteration k denoted by $\mathbf{z}_{t_*}^k$, we repeatedly:

(1) Optimize in pixel-space. We compute a denoised latent prediction $\hat{\mathbf{z}}_0^k = \hat{\mathbf{x}}_0^\theta(\mathbf{z}_{t_*}^k, t_*)$ and decode it to $\hat{\mathbf{x}}_0^k = \mathcal{D}(\hat{\mathbf{z}}_0^k)$. We then optimize the pixel-space objective $\mathbf{x} \mapsto \|\mathbf{y} - \mathcal{A}(\mathbf{x})\|^2$ initialized at $\hat{\mathbf{x}}_0^k$, yielding $\mathbf{x}_0^k$, and map it back to the latent space via $\mathbf{z}_0^k = \mathcal{E}(\mathbf{x}_0^k)$. Crucially, this optimization requires backpropagation only through the forward operator $\mathcal{A}(\cdot)$ and not the decoder $\mathcal{D}$ or denoiser $\hat{\mathbf{x}}_0(\cdot,\cdot)$, making it computationally lightweight compared to standard guidance methods operating in latent space Song et al. (2023a); Zhang et al. (2025); Kim et al. (2025).

(2) Re-noise in latent space. We push the solution to the previous optimization problem into the latent space and then noise it to get the next noisy state, *i.i.d.* sampling

$$\mathbf{z}_{t_*}^{k+1} \sim \mathcal{N}(\alpha_{t_*}\boldsymbol{\mu}^k, \sigma_{t_*}^2\sigma^k),$$

where $\boldsymbol{\mu}^k := \alpha_{t_*}\mathbf{z}_0^k + \sigma_{t_*}\hat{\mathbf{z}}_0^k + \sigma_{t_*}\epsilon^k$ is the mixture of the optimized solution and the denoiser prediction, $\sigma^k := \sigma_{t_*}^2\mathbf{I}$ is the noise covariance, and $\epsilon^k := (\mathbf{z}_{t_*}^k - \alpha_{t_*}\hat{\mathbf{z}}_0^k)/\sigma_{t_*}$ is the predicted noise at t_*. This mixing ensures that the blend of optimized solution and prediction is injected back to the sample at $\mathbf{z}_{t_*}^{k+1}$. We initialize the next phase at the time step t_* with the sample $\mathbf{z}_{t_*}^N$.

Phase 2: guided denoising. We define a grid of time steps $t_1, \ldots, t_M$, with $t_M = t_*$ and $t_1 \approx 0$, over which we perform optimization steps. Spacing timesteps is defined by a specifically designed scheduler. M is, in practice, chosen rather small. The state at timestep t_M is set to $\mathbf{z}_{t_M}^{\mathbf{y}} = \mathbf{z}_{t_*}^N$. Given the time step t_{k+1} and the state $\mathbf{z}_{t_{k+1}}^{\mathbf{y}}$, we first jump to the time step t_k by sampling $\tilde{\mathbf{z}}_{t_k} \sim p_{t_k|t_{k+1}}^\theta(\cdot|\mathbf{z}_{t_{k+1}}^{\mathbf{y}})$. This state is then denoised to $\tilde{\mathbf{z}}_0 = \hat{\mathbf{x}}_0^\theta(\tilde{\mathbf{z}}_{t_k}, t_k)$ and its decoding $\tilde{\mathbf{x}}_0 = \mathcal{D}(\tilde{\mathbf{z}}_0)$ is used to initialize optimization steps on the *pixel-space* objective

$$\mathbf{x} \mapsto \|\mathbf{y} - \mathcal{A}(\mathbf{x})\|^2 + \lambda\|\mathbf{x} - \tilde{\mathbf{x}}_0\|^2 .$$

Letting $\mathbf{z}_0^* = \mathcal{E}(\mathbf{x}_0^*)$ denote the encoding of the solution $\mathbf{x}_0^*$ to this optimization problem, we return to the timestep t_{k+1} by noising, *i.i.d.* sampling

$$\tilde{\mathbf{z}}_{t_{k+1}} \sim \mathcal{N}(\boldsymbol{\mu}_{t_{k+1}}, \sigma_{t_{k+1}}^2\sigma'_{t_{k+1}}),$$

where $\boldsymbol{\mu}_{t_{k+1}} := \alpha_{t_{k+1}}\mathbf{z}_0^* + \sigma_{t_{k+1}}\alpha_{t_{k+1}}\epsilon_{t_k}$; $\sigma'_{t_{k+1}} := \sigma_{t_{k+1}}^2\mathbf{I}$ and $\epsilon_{t_k} := (\tilde{\mathbf{z}}_{t_k} - \alpha_{t_k}\tilde{\mathbf{z}}_0)/\sigma_{t_k}$. This ensures that re-noising process has information from previous step t_k. Then we initialize and perform the DDIM steps from t_{k+1} to t_k in n steps to obtain the final state $\mathbf{z}_{t_k}^{\mathbf{y}}$ at t_k. The complete procedure is summarized in Algorithm 1.

Overall, the algorithm alternates between learned prior refinement (via the denoiser and diffusion dynamics) and data consistency enforcement (via explicit optimization), yielding reconstructions that are both realistic and faithful to the observations. See Appendix A for derivations and Appendix B for a theoretical interpretation.

Takeaway: Our method accelerates inference by replacing the full reverse SDE ($t = 1 \to 0$) with a truncated trajectory ($t_* \to 0$). Crucially, we avoid expensive gradient calculations through the network by decoupling the guidance: data consistency is enforced via lightweight pixel-space optimization, while the prior is handled separately by standard latent-space denoising.

4 Related Work

Likelihood-score guidance with VJPs. Diffusion Posterior Sampling (DPS) Chung et al. (2022a) and Pseudoinverse-Guided Diffusion Models (PGDM) Song et al. (2023b) approximate the intractable posterior score in (4) to guide reverse diffusion. While effective, these approaches require vector–Jacobian products (VJPs) through the denoiser (and through the decoder for latent diffusion models), which leads to large memory overhead and slow inference. Several works reduce or restructure this cost, e.g., via decoupled schedules (DAPS) Zhang et al. (2025), variational reformulations (RED-Diff) Mardani et al. (2023), or alternative transition designs Kim et al. (2025), but they still fundamentally rely on differentiating through the generative model.

Gradient-free posterior sampling and plug-and-play. For linear operators with known SVD, DDNM Wang et al. (2022) and DDRM Kawar et al. (2022) enforce data consistency through

Table 1: Comparison of training-free diffusion/consistency-model inverse-problem solvers. VJP: backprop through denoiser $\hat{\mathbf{x}}_0(\cdot,\cdot)$ or encoder/decoder $\mathcal{E}/\mathcal{D}$ (†latent models only). DC Guidance Density: frequency of data-consistency or likelihood guidance application.

Method	Inference Objective	DC Guidance Density	VJP $\hat{\mathbf{x}}_0^\theta(\cdot,\cdot)$?	VJP $\mathcal{E}/\mathcal{D}$?	Assumptions on $\mathcal{A}$
DPS, PGDM	Score-based posterior approximation	Dense	Yes	Yes†	General
DAPS	Decoupled guidance	Moderate	Yes	Yes†	General
ReSample, RED-Diff	Hard consistency via projection, Variational optimization	Dense + inner solve	No	Often†	General
PnP-DM, PSLD, FlowDPS	Alternating proximal sampling	Dense + inner loops	No	Often†	General
DDNM, DDRM	Subspace projection	Dense	No	No	Linear + SVD
DiffPIR	Proximal-MAP optimization	Dense	No	No	Typically linear
CCDF	Stochastic contractive mapping	Dense	No	No	Linear
G&G (ours)	Sparse time-marginal guidance	Sparse	No	No	General

closed-form updates (pseudo-inverse/projection), avoiding VJPs but restricting applicability. More general plug-and-play formulations avoid VJPs by alternating denoiser predictions with explicit data-consistency steps, e.g., DiffPIR Zhu et al. (2023) and PnP-DM Wu et al. (2024). ReSample Song et al. (2023a) extends hard data consistency to latent diffusion models via optimization and resampling back to the noisy manifold. Our method is closest in spirit to these alternating schemes, but differs in that we (i) require gradients only through the forward operator $\mathcal{A}$ (no VJPs through denoiser or encoder/decoder), (ii) apply data consistency at a sparse set of timesteps with a learned schedule, and (iii) introduce a warm-start phase to skip expensive early denoising steps.

Warm starts and accelerated conditional diffusion. Several works explore alternatives to pure-noise initialization. SDEdit Meng et al. (2021) starts from an intermediate noise level using a noisy guide image, while ILVR Choi et al. (2021) iteratively injects reference information. CCDF Chung et al. (2022b) accelerates conditional diffusion for inverse problems by initializing from a data-consistent estimate and leveraging contraction of alternating diffusion and consistency steps. Our Phase 1 can be viewed as iterating this principle at a fixed noise level t_* to approximate a posterior time-marginal before running the guided reverse process.

Consistency models and few-step guidance. Consistency models Song et al. (2023c) enable few-step generation and have recently been adapted to inverse problems, e.g., CM4IR Garber & Tirer (2025). While these methods rely on few-step updates with noise correction, our approach uses a diffusion-model prior and derives re-noising as a noise-preserving coupling (Appendix A) that minimally perturbs the current sample when enforcing data consistency.

In comparison to all existing methods, a *warm start* algorithm is novel. Our method bypasses the computationally expensive early diffusion steps entirely. This is complemented by a decoupled guidance framework that performs optimization strictly in pixel space. For a systematic comparison of the probabilistic targets, required gradients, and transition designs across methods, see Table 1.

5 Experiments

Datasets, Models and Baselines. We conduct experiments using both latent-space and pixel-space diffusion models. For latent-space experiments, we employ the `FFHQ` latent diffusion model from Rombach et al. Rombach et al. (2022). For pixel-space experiments, we use the pixel-space `FFHQ` model from Choi et al. Choi et al. (2021) and the `ImageNet` model from Dhariwal and Nichol Dhariwal & Nichol (2021). Evaluation is performed on 100 validation samples from `FFHQ` Karras et al. (2019) (the first 100) and `ImageNet` Russakovsky et al. (2015) (randomly sampled to mitigate class bias). In all reported metrics, mean and standard deviation are shown. All pixel-space experiments are conducted at 256×256 resolution.

For pixel-space models, we compare against DPS Chung et al. (2022a), PGDM Song et al. (2023b), DDNM Wang et al. (2022), DiffPir Zhu et al. (2023), RED-Diff Mardani et al. (2023),

Table 2: Mean LPIPS for linear/nonlinear imaging tasks on the **FFHQ** datasets with $\sigma_y = 0.05$. Lower metrics are better.

	↓ LPIPS FFHQ							
Task	G&G	DAPS	RED-DIFF	PNP-DM	DPS	DDNM	PGDM	DIFFPIR
Gauss. Deblur	0.17 ± 0.06	0.19 ± 0.06	0.25 ± 0.07	0.20 ± 0.07	0.16 ± 0.05	0.84 ± 0.04	0.87 ± 0.14	–
Mot. Deblur	0.13 ± 0.04	0.19 ± 0.06	0.20 ± 0.06	0.21 ± 0.06	0.21 ± 0.06	–	–	–
SR (×4)	0.20 ± 0.07	0.19 ± 0.06	0.44 ± 0.08	0.21 ± 0.06	0.22 ± 0.07	0.36 ± 0.08	0.30 ± 0.07	–
SR (×16)	0.35 ± 0.08	0.45 ± 0.10	0.59 ± 0.08	0.60 ± 0.15	0.36 ± 0.08	0.52 ± 0.09	0.42 ± 0.06	–
Box Inp.	0.14 ± 0.04	0.12 ± 0.04	0.18 ± 0.05	0.55 ± 0.07	0.20 ± 0.08	0.18 ± 0.05	0.17 ± 0.04	0.14 ± 0.04
Half Inp.	0.24 ± 0.06	0.24 ± 0.06	0.29 ± 0.07	0.60 ± 0.04	0.26 ± 0.06	0.28 ± 0.06	0.25 ± 0.05	0.25 ± 0.05
JPEG (QF=2)	0.16 ± 0.06	0.22 ± 0.07	0.32 ± 0.08	0.27 ± 0.07	0.28 ± 0.07	–	–	–
Phase Retr.	0.35 ± 0.24	0.30 ± 0.25	0.35 ± 0.24	0.44 ± 0.23	0.48 ± 0.18	–	–	–
HDR	0.13 ± 0.05	0.08 ± 0.05	0.19 ± 0.06	0.18 ± 0.07	0.64 ± 0.36	–	–	–
Memory (MB)	1983	2095	1985	1985	3309	2019	3409	1985
Runtime (sec)	25	75	50	194	105	47	101	50

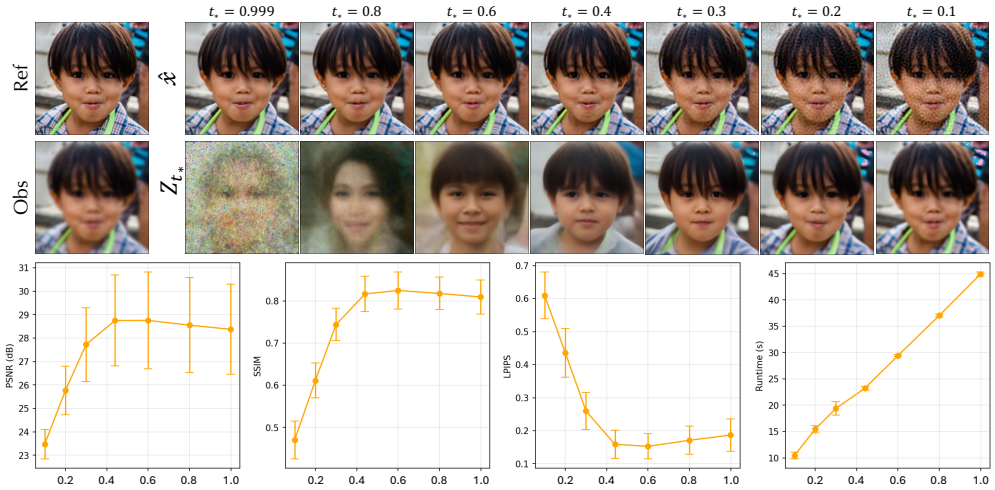

Figure 2: Impact of initial timestep t_* on the performance of G&G method. The first row of images shows visual reconstruction quality for different t_* values, and the second row shows what the initial guess looks like. Plots display the variance of metrics and runtime at different t_* values and the standard deviation of metrics.

DAPS Zhang et al. (2025), and PNP-DM Wu et al. (2024). For latent-space models, we compare against FLOWDPS Kim et al. (2025), DAPS Zhang et al. (2025), RED-DIFF Mardani et al. (2023), RESAMPLE Song et al. (2023a), PSLD Rout et al. (2023), and PNP-DM Wu et al. (2024). See Appendix D for more details on the implementation of each competitor and hyperparameters.

Inverse Tasks and Settings. We evaluate on a diverse set of linear and nonlinear inverse problems following the experimental protocol of Janati et al. Janati et al. (2025). *Linear problems* include: (1) Super-resolution (SR) with upscaling factors of $\times 4$ and $\times 16$; (2) image inpainting with two mask configurations – a 150×150 central box mask and a half-mask covering the right portion of the image; (3) Gaussian deblurring and motion deblurring, both using a 61×61 kernel following Chung et al. Chung et al. (2022a). *Nonlinear problems* include: (1) JPEG dequantization at quality factor 2%, implemented using the differentiable operator from Shin et al. Shin et al. (2017); (2) Phase Retrieval with $\times 2$ oversampling; (3) non-uniform deblurring using the operator from Tran et al. Tran et al. (2021); and (4) High Dynamic Range (HDR) reconstruction following the setup in Mardani et al. Mardani et al. (2023). All experiments use a noise level $\sigma_y = 0.05$ and are evaluated on these samples for each task.

Table 3: Mean LPIPS for linear/nonlinear imaging tasks on the **ImageNet** datasets with $\sigma_y = 0.05$. Lower metrics are better.

	↓ LPIPS ImageNet							
Task	G&G	DAPS	RED-DIFF	PNP-DM	DPS	DDNM	PGDM	DIFFPIR
Gauss. Deblur	0.34 ± 0.10	0.41 ± 0.12	0.44 ± 0.11	0.47 ± 0.13	0.35 ± 0.14	0.42 ± 0.13	1.06 ± 0.10	–
Mot. Deblur	0.28 ± 0.09	0.40 ± 0.11	0.35 ± 0.09	0.50 ± 0.13	0.41 ± 0.13	–	–	–
SR (×4)	0.39 ± 0.11	0.41 ± 0.12	0.53 ± 0.13	0.49 ± 0.14	0.47 ± 0.14	0.27 ± 0.09	0.63 ± 0.12	–
SR (×16)	0.68 ± 0.13	0.80 ± 0.13	0.82 ± 0.11	0.88 ± 0.10	0.58 ± 0.10	0.68 ± 0.17	0.59 ± 0.08	–
Box Inp.	0.32 ± 0.05	0.31 ± 0.05	0.36 ± 0.06	0.71 ± 0.14	0.46 ± 0.13	0.28 ± 0.06	0.28 ± 0.05	0.28 ± 0.05
Half Inp.	0.38 ± 0.08	0.40 ± 0.05	0.46 ± 0.05	0.70 ± 0.12	0.47 ± 0.10	0.38 ± 0.07	0.33 ± 0.05	0.34 ± 0.07
JPEG (QF=2)	0.30 ± 0.11	0.44 ± 0.12	0.47 ± 0.12	0.54 ± 0.13	0.57 ± 0.12	–	–	–
Phase Retr.	0.65 ± 0.14	0.60 ± 0.18	0.64 ± 0.12	0.73 ± 0.11	0.67 ± 0.08	–	–	–
HDR	0.17 ± 0.10	0.14 ± 0.11	0.21 ± 0.13	0.34 ± 0.18	0.88 ± 0.16	–	–	–
Memory (MB)	4991	4993	4995	4993	8701	5031	8741	5007
Run time (sec)	83	219	165	636	360	296	371	169

5.1 MAIN RESULTS

Our method achieves better or comparable metrics on both datasets and models while being at least **2x** faster than all baseline methods. See Tables 2, 3, 4, and Figure 1 for a quantitative and qualitative comparison of our method with baselines. We report LPIPS Zhang et al. (2018) as our primary metric, with PSNR and SSIM provided in the supplementary material (see Appendix F) since PSNR and SSIM perform pixel-wise comparisons and favor overly smooth images (see Janati et al. (2025), Appendix B.6). We highlight the best performing method in each row with a colored background as and the 2nd best performing method as . The '–' sign in the tables denotes that a baseline is not applicable or not working on the given task.

Reconstruction quality. Across all three experimental settings (FFHQ pixel-space, ImageNet pixel-space, and FFHQ latent-space), G&G demonstrates competitive or superior performance compared to state-of-the-art baselines with significantly faster runtime. On pixel-space models, our method achieves best or second-best results on most linear inverse problems, including deblurring and super-resolution tasks. For nonlinear problems such as JPEG dequantization, phase retrieval, and HDR reconstruction, we maintain strong performance across both datasets. In the latent diffusion setting, our method provides a good balance between reconstruction quality and computational efficiency, achieving results comparable to specialized methods like RESAMPLE and PSLD.

Computational efficiency. A key advantage of our method is computational efficiency. By eliminating backpropagation through the denoiser network (and Encoder-Decoder in case of latent-space models), G&G achieves substantially lower memory consumption and faster runtime compared to gradient-based baselines at least **2x**. For instance, on `FFHQ` pixel-space experiments, our method requires only 1983 MB of memory and 25 seconds runtime, compared to DPS (3309 MB, 105 sec) and PNP-DM (194 sec). Similar efficiency gains are observed on `ImageNet`. In latent diffusion experiments, runtime differences are even more drastic achieving **20x** speedup compared to RESAMPLE and **50x** to DAPS. This makes our method particularly suitable for practical deployment and high-resolution generation tasks.

Hyperparameters. For the warm-start phase (Phase 1), we set the initial timestep t_* depending on task complexity, typically in the range $t_* \in [0.44, 0.8]$. We use $t_* \in [0.44, 0.5]$ for relatively simple linear problems like deblurring, while more challenging tasks such as super-resolution at $\times 16$ or inpainting use $t_* \in [0.6, 0.8]$ to allow for sufficient prior regularization. The number of warm-start iterations N ranges from 5 to 30 depending on task difficulty, with each iteration performing $N_{\text{OPT}}^{\text{int}} = 50$ gradient steps on the observation likelihood. We set the learning rate for this phase to $\eta_{\text{init}} \in [1 \times 10^{-4}, 2 \times 10^{-4}]$.

For the guided denoising phase (Phase 2), we define a grid of M timesteps, where $M \in [\![20, 40]\!]$ depending on the task. At each guidance step, we perform G_{OPT} optimization iterations, where $G_{\text{OPT}} \in [\![50, 200]\!]$ (50 for simpler problems and 200 for highly ill-posed tasks like $\times 16$ super-resolution). The main optimization learning rate is set to $\eta_{\text{main}} = 1 \times 10^{-3}$ across all experiments. The regularization weight λ in the pixel-space objective is set to 0 for most tasks, apart from phase

Table 4: Mean LPIPS for linear/nonlinear imaging tasks on the **FFHQ** datasets with LDM prior and $\sigma_y = 0.05$.

	↓LPIPS FFHQ Latent				
Task	G&G	ReSample	PSLD	DAPS	PNP-DM
Gauss. Deblur	0.18 ± 0.02	0.16 ± 0.04	0.59 ± 0.03	0.32 ± 0.07	0.32 ± 0.07
Mot. Deblur	0.30 ± 0.05	0.20 ± 0.06	0.70 ± 0.02	0.36 ± 0.07	0.36 ± 0.08
SR (×4)	0.20 ± 0.07	0.22 ± 0.05	0.21 ± 0.03	0.28 ± 0.06	0.40 ± 0.07
SR (×16)	0.45 ± 0.10	0.38 ± 0.12	0.36 ± 0.05	0.52 ± 0.09	0.71 ± 0.11
Box Inp.	0.19 ± 0.06	0.22 ± 0.04	0.27 ± 0.13	0.37 ± 0.05	0.31 ± 0.12
Half Inp.	0.30 ± 0.04	0.30 ± 0.03	0.32 ± 0.02	0.49 ± 0.01	0.44 ± 0.08
JPEG (QF=2)	0.34 ± 0.10	0.26 ± 0.05	–	0.32 ± 0.06	0.36 ± 0.03
Phase Retr.	0.55 ± 0.07	0.39 ± 0.17	–	0.25 ± 0.21	0.50 ± 0.14
HDR	0.27 ± 0.04	0.12 ± 0.04	–	0.24 ± 0.09	0.24 ± 0.05
Memory (MB)	4525	6238	6919	5885	5885
Runtime (sec)	24	509	244	1254	1323

Table 5: LPIPS scores for different noise schedules across tasks

Schedule	Deblur	Motion Deblur	SR4
Uniform	0.16 ± 0.05	0.14 ± 0.04	0.15 ± 0.04
Linear	0.18 ± 0.05	0.19 ± 0.06	0.18 ± 0.05
Polynomial	0.16 ± 0.04	0.14 ± 0.04	0.21 ± 0.05
Exponential	0.15 ± 0.04	0.11 ± 0.03	0.19 ± 0.05
Beta	0.22 ± 0.05	0.32 ± 0.06	0.28 ± 0.06
Gaussian	0.15 ± 0.03	0.12 ± 0.03	0.15 ± 0.03

Takeaway: Gaussian schedule gives the best or tied-best LPIPS across these tasks, so we use it by default.

retrieval and HDR reconstruction, relying entirely on the implicit regularization from the diffusion prior.

The spacing of the guidance/optimization steps grid is controlled by our iteration scheduling strategy. We primarily employ a Gaussian schedule with center $\mu_G \in [0.3, 0.5]$ and width $\sigma_G \in [10.0, 15.0]$, which concentrates optimization steps in the intermediate noise regime (approximately $t_k \in [0.4, 0.6]$). This choice reflects the intuition that guidance applied too close to the data manifold ($t_k \approx 0$) limits the denoiser's ability to correct artifacts introduced by optimization. For interleaving DDIM steps between guidance operations, we use a stochasticity parameter $\eta = 1.0$, corresponding to ancestral sampling. More details about the hyperparameters are provided in Appendix G.

5.2 Ablation Studies

In this section, we will study the impact of Phase 1 and Phase 2 on the performance of our method. All experiments in this section are conducted on FFHQ dataset. As mentioned in Section 3, the choice of initial timestep t_* is critical for both the quality of reconstruction and computational efficiency. Figure 2 displays the impact of the initial timestep t_* on the performance of our method. We see that the best performance is achieved when t_* is in the range around $[0.4, 0.6]$. Starting too early ($t_* \gg 0.6$) does not significantly improve the final reconstruction result (LPIPS flattens), but it significantly increases runtime, and the final reconstruction becomes smoothed out. However, starting too late ($t_* \ll 0.4$) leads to poor reconstruction quality by inducing artifacts. For other metrics, see Appendix E.

Another important factor is the spacing of the steps t_k where the optimization is performed. Table 5 shows the impact of different scheduling strategies on the performance of our method. Results show that employing a standard uniform spacing schedule does not produce the best results with our algorithm. Among all schedulers, the best performance across the tasks is achieved by the Gaussian scheduling strategy. For most tasks, we set $\mu_G = 0.5$ and $\sigma_G = 10.0$, which concentrates most of the optimization steps around $t_k \in [0.4, 0.6]$. See Appendix C for more details about the scheduling strategies and hyperparameters.

Limitations. Choosing the correct initial timestep t_* is critical for both the speed and quality of the reconstruction. If chosen close to the base distribution p_1, in Phase 2 more iterations are needed to converge. In contrast, if chosen close to the target distribution p_0, in Phase 2 it may not infuse enough information to reconstruct the image. The choice of t_* is a trade-off between speed and quality.

The spacing of the optimization steps $\{t_1, \ldots, t_M\}$ is an important factor. The choice of step spacing has a critical impact on reconstruction quality. We consider various scheduling strategies — *linear, polynomial, exponential, Gaussian, and Beta*—that concentrate optimization procedures at different timesteps. In practice, concentrating optimization in the early stages (larger timesteps) generally yields better reconstructions, as the denoiser has more opportunity to inject information and correct artifacts introduced by the optimization procedure before reaching the final clean state.

6 CONCLUSION

We introduced G&G, a computationally efficient framework for zero-shot diffusion-based inverse problem solving. Our method eliminates backpropagation through denoiser networks by decomposing inference into two phases: a warm-start phase that obtains high-quality initial estimates through iterative optimization and re-noising at a fixed timestep, followed by guided denoising that refines estimates through pixel-space optimization. This achieves at least $2\times$ speedup over gradient-based baselines while maintaining competitive reconstruction quality across diverse linear and nonlinear inverse problems on FFHQ and ImageNet datasets.

By restricting the gradient computation to the forward operator $\mathcal{A}(\cdot)$ only, our approach substantially reduces memory consumption and inference time, making it suitable for high-resolution generation where gradient-based methods become prohibitively expensive. The results demonstrate robust performance across multiple diffusion architectures and inverse problem types. Our work demonstrates that practical algorithm design, which trades exact posterior score estimation for computational efficiency, can expand the applicability of pre-trained diffusion priors in deployment scenarios where memory and inference time are critical constraints.

REFERENCES

Heinz H. Bauschke and Patrick L. Combettes. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer, 2 edition, 2017.

Jooyoung Choi, Sungwon Kim, Yonghyun Jeong, Youngjune Gwon, and Sungroh Yoon. Ilvr: Conditioning method for denoising diffusion probabilistic models. *arXiv preprint arXiv:2108.02938*, 2021.

Hyungjin Chung, Jeongsol Kim, Michael T Mccann, Marc L Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. *arXiv preprint arXiv:2209.14687*, 2022a.

Hyungjin Chung, Byeongsu Sim, and Jong Chul Ye. Come-closer-diffuse-faster: Accelerating conditional diffusion models for inverse problems through stochastic contraction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12403–12412, 2022b. doi: 10.1109/CVPR52688.2022.01209.

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. *arXiv preprint arXiv:2410.00083*, 2024.

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

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas M"uller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

Ruiqi Gao, Emiel Hoogeboom, Jonathan Heek, Valentin De Bortoli, Kevin P Murphy, and Tim Salimans. Diffusion meets flow matching: Two sides of the same coin. 2024. *URL https://diffusionflow. github. io*, 2024.

Tomer Garber and Tom Tirer. Zero-shot image restoration using few-step guidance of consistency models (and beyond). In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2398–2407, 2025.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Yazid Janati, Badr Moufad, Mehdi Abou El Qassime, Alain Oliviero Durmus, Eric Moulines, and Jimmy Olsson. A mixture-based framework for guiding diffusion models. In *Forty-second International Conference on Machine Learning*, 2025.

Zahra Kadkhodaie and Eero P Simoncelli. Solving linear inverse problems using the prior implicit in a denoiser. *arXiv preprint arXiv:2007.13640*, 2020.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4401–4410, 2019.

Bahjat Kawar, Gregory Vaksman, and Michael Elad. Snips: Solving noisy inverse problems stochastically. *Advances in Neural Information Processing Systems*, 34:21757–21769, 2021.

Bahjat Kawar, Michael Elad, Stefano Ermon, and Jiaming Song. Denoising diffusion restoration models. *Advances in neural information processing systems*, 35:23593–23606, 2022.

Jeongsol Kim, Bryan Sangwoo Kim, and Jong Chul Ye. Flowdps: Flow-driven posterior sampling for inverse problems. *arXiv preprint arXiv:2503.08136*, 2025.

M. A. Krasnosel'skii. Two remarks on the method of successive approximations. *Uspekhi Matematicheskikh Nauk*, 10(1):123–127, 1955.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11461–11471, 2022.

W. Robert Mann. Mean value methods in iteration. *Proceedings of the American Mathematical Society*, 4(3):506–510, 1953.

Morteza Mardani, Jiaming Song, Jan Kautz, and Arash Vahdat. A variational perspective on solving inverse problems with diffusion models. *arXiv preprint arXiv:2305.04391*, 2023.

Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations. *arXiv preprint arXiv:2108.01073*, 2021.

Badr Moufad, Navid Bagheri Shouraki, Alain Oliviero Durmus, Thomas Hirtz, Eric Moulines, Jimmy Olsson, and Yazid Janati. Efficient zero-shot inpainting with decoupled diffusion guidance. *arXiv preprint arXiv:2512.18365*, 2025.

Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3): 127–239, 2014.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Bj"orn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Litu Rout, Negin Raoof, Giannis Daras, Constantine Caramanis, Alex Dimakis, and Sanjay Shakkottai. Solving linear inverse problems provably via posterior sampling with latent diffusion models. *Advances in Neural Information Processing Systems*, 36:49960–49990, 2023.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115 (3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Chitwan Saharia, William Chan, Huiwen Chang, Chris Lee, Jonathan Ho, Tim Salimans, David Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. In *ACM SIGGRAPH 2022 conference proceedings*, pp. 1–10, 2022.

Richard Shin, Dawn Song, et al. Jpeg-resistant adversarial images. In *NIPS 2017 workshop on machine learning and computer security*, volume 1, pp. 8, 2017.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. pmlr, 2015.

Bowen Song, Soo Min Kwon, Zecheng Zhang, Xinyu Hu, Qing Qu, and Liyue Shen. Solving inverse problems with latent diffusion models via hard data consistency. *arXiv preprint arXiv:2307.08123*, 2023a.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020a.

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

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020b.

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

Phong Tran, Anh Tuan Tran, Quynh Phung, and Minh Hoai. Explore image deblurring via encoded blur kernel space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11956–11965, 2021.

Singanallur V. Venkatakrishnan, Charles A. Bouman, and Brendt Wohlberg. Plug-and-play priors for model based reconstruction. In *2013 IEEE Global Conference on Signal and Information Processing*, pp. 945–948. IEEE, 2013.

Cédric Villani. *Optimal Transport: Old and New*, volume 338 of *Grundlehren der mathematischen Wissenschaften*. Springer, 2009.

Yinhuai Wang, Jiwen Yu, and Jian Zhang. Zero-shot image restoration using denoising diffusion null-space model. *arXiv preprint arXiv:2212.00490*, 2022.

Zihui Wu, Yu Sun, Yifan Chen, Bingliang Zhang, Yisong Yue, and Katherine Bouman. Principled probabilistic imaging using diffusion models as plug-and-play priors. *Advances in Neural Information Processing Systems*, 37:118389–118427, 2024.

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*, pp. 20895–20905, 2025.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

Yuanzhi Zhu, Kai Zhang, Jingyun Liang, Jiezhang Cao, Bihan Wen, Radu Timofte, and Luc Van Gool. Denoising diffusion models for plug-and-play image restoration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 1219–1229, June 2023.

Supplementary Material

A Algorithm

A.1 Phase 1: Warm Start - Derivation of Re-noising Step

In Phase 1, after optimizing to obtain $\mathbf{z}_0^k$, our goal is to re-noise it back to timestep t_* to prepare for the next iteration. We effectively want to blend the information from our optimized solution $\mathbf{z}_0^k$ with the predicted clean state $\hat{\mathbf{z}}_0^k$, while preserving the correct noise statistics.

To achieve this, we consider a weighted combination of the clean states, $\mathbf{z}_0^{\text{blend}} := \alpha_{t_*}\mathbf{z}_0^k + \sigma_{t_*}\hat{\mathbf{z}}_0^k$, where the coefficients ensure that for small t_* we recover the optimized solution (since $\alpha_{t_*} \to 1$ and $\sigma_{t_*} \to 0$), while for larger t_*, the prediction has more influence.

Similarly, we construct a blended noise term by mixing the predicted noise $\epsilon^k = (\mathbf{z}_{t_*}^k - \alpha_{t_*}\hat{\mathbf{z}}_0^k)/\sigma_{t_*}$ with fresh Gaussian noise $\epsilon \sim \mathcal{N}(0, \mathbf{I})$. Specifically, we define:

$$\epsilon^{\text{blend}} := \alpha_{t_*}\epsilon^k + \sigma_{t_*}\epsilon.$$

This mixture ensures that information from the current noisy state is preserved through ϵ^k while introducing fresh stochasticity. Applying the forward noising kernel to this blended state yields:

$$\begin{aligned}
\mathbf{z}_{t_*}^{k+1} &= \alpha_{t_*}\mathbf{z}_0^{\text{blend}} + \sigma_{t_*}\epsilon^{\text{blend}} \\
&= \alpha_{t_*}\left(\alpha_{t_*}\mathbf{z}_0^k + \sigma_{t_*}\hat{\mathbf{z}}_0^k\right) + \sigma_{t_*}\left(\alpha_{t_*}\epsilon^k + \sigma_{t_*}\epsilon\right) \\
&= \alpha_{t_*}\left(\underbrace{\alpha_{t_*}\mathbf{z}_0^k + \sigma_{t_*}\hat{\mathbf{z}}_0^k + \sigma_{t_*}\epsilon^k}_{\boldsymbol{\mu}^k}\right) + \sigma_{t_*}^2\epsilon.
\end{aligned}$$

This formulation aligns directly with Algorithm 1. Defining $\sigma^k := \sigma_{t_*}^2\mathbf{I}$, the above is equivalent to sampling

$$\mathbf{z}_{t_*}^{k+1} \sim \mathcal{N}(\alpha_{t_*}\boldsymbol{\mu}^k, \sigma_{t_*}^2\sigma^k),$$

which completes the derivation of the re-noising step in Phase 1.

A.2 Phase 2: Guided Denoising - Derivation of Re-noising Step

In Phase 2, the re-noising process starts from the pixel-space optimized clean state $\mathbf{x}_0^*$ (and its encoding $\mathbf{z}_0^* = \mathcal{E}(\mathbf{x}_0^*)$) and the noise estimate ϵ_{t_k} obtained from the jump step. We wish to sample the state at the previous timestep t_{k+1} for the subsequent refinement.

Here, we rely directly on $\mathbf{z}_0^*$ as the clean latent state. We blend the existing noise ϵ_{t_k} (derived as $\epsilon_{t_k} = (\tilde{\mathbf{z}}_{t_k} - \alpha_{t_k}\tilde{\mathbf{z}}_0)/\sigma_{t_k}$) with fresh Gaussian noise $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ using the coefficients at t_{k+1}:

$$\epsilon^{\text{blend}} := \alpha_{t_{k+1}}\epsilon_{t_k} + \sigma_{t_{k+1}}\epsilon.$$

Using the reparametrization trick, we sample the state $\tilde{\mathbf{z}}_{t_{k+1}}$:

$$\begin{aligned}
\tilde{\mathbf{z}}_{t_{k+1}} &= \alpha_{t_{k+1}}\mathbf{z}_0^* + \sigma_{t_{k+1}}\epsilon^{\text{blend}} \\
&= \alpha_{t_{k+1}}\mathbf{z}_0^* + \sigma_{t_{k+1}}(\alpha_{t_{k+1}}\epsilon_{t_k} + \sigma_{t_{k+1}}\epsilon) \\
&= \underbrace{\alpha_{t_{k+1}}\mathbf{z}_0^* + \sigma_{t_{k+1}}\alpha_{t_{k+1}}\epsilon_{t_k}}_{\boldsymbol{\mu}_{t_{k+1}}} + \sigma_{t_{k+1}}^2\epsilon.
\end{aligned}$$

Matching this to Algorithm 1, we identify the mean $\boldsymbol{\mu}_{t_{k+1}} := \alpha_{t_{k+1}}\mathbf{z}_0^* + \sigma_{t_{k+1}}\alpha_{t_{k+1}}\epsilon_{t_k}$. The variance term $\sigma_{t_{k+1}}^2\epsilon$ implies a covariance of $\sigma_{t_{k+1}}^4\mathbf{I}$, which we denote by $\sigma'_{t_{k+1}} := \sigma_{t_{k+1}}^2\mathbf{I}$. This gives the final sampling distribution:

$$\tilde{\mathbf{z}}_{t_{k+1}} \sim \mathcal{N}(\boldsymbol{\mu}_{t_{k+1}}, \sigma_{t_{k+1}}^2\sigma'_{t_{k+1}}),$$

completing the derivation for Phase 2.

B THEORETICAL INTERPRETATION

This section provides a concise interpretation of G&G as an *approximate split inference* procedure that alternates between (i) a prior-driven diffusion update and (ii) a data-consistency update implemented via pixel-space optimization.

Notation. We follow the paper's convention that Z_t (resp. X) denotes a latent-space (resp. pixel-space) variable, while bold symbols such as $\mathbf{z}_t$ and $\mathbf{x}$ denote generic realizations/iterates. Algorithm 1 follows this convention and writes iterates in bold (e.g., $\mathbf{z}$, $\mathbf{x}$), while the background uses uppercase notation (Z_t, X_t) when referring to the underlying diffusion random variables and their marginals. In Phase 2, the algorithm uses a constant weight λ; in this section we allow a time-dependent λ_t, and the algorithm can be recovered by taking $\lambda_t \equiv \lambda$ (or λ_{t_k} on the chosen schedule) and, when convenient, absorbing the σ_y^2 scaling from the Gaussian likelihood into λ.

Target posterior and time-marginals. Recall the latent-space posterior at $t = 0$:

$$\pi_0(\mathbf{z}_0 \mid \mathbf{y}) \;\propto\; \ell_0(\mathbf{y} \mid \mathbf{z}_0)\, p_0(\mathbf{z}_0), \qquad \ell_0(\mathbf{y} \mid \mathbf{z}_0) = \mathcal{N}\big(\mathbf{y};\, \mathcal{A}(\mathcal{D}(\mathbf{z}_0)), \sigma_y^2 \mathbf{I}\big)\,. \tag{5}$$

For any noise level $t \in [0, 1]$, define the posterior time-marginal $\pi_t(\mathbf{z}_t \mid \mathbf{y})$ by pushing $\pi_0(\cdot \mid \mathbf{y})$ through the forward noising kernel $p_t(\mathbf{z}_t \mid \mathbf{z}_0)$, (i.e., $\mathbf{z}_t = \alpha_t \mathbf{z}_0 + \sigma_t \mathbf{z}_1$). This yields the classical form

$$\pi_t(\mathbf{z}_t \mid \mathbf{y}) \;\propto\; \ell_t(\mathbf{y} \mid \mathbf{z}_t)\, p_t(\mathbf{z}_t), \qquad \ell_t(\mathbf{y} \mid \mathbf{z}) = \mathbb{E}[\ell_0(\mathbf{y} \mid Z_0) \mid Z_t = \mathbf{z}], \tag{6}$$

and exact posterior sampling can be implemented by reverse-time transitions that depend on the score of ℓ_t, cf. (4). Our method avoids evaluating $\nabla_{\mathbf{z}_t} \log \ell_t(\mathbf{y} \mid \mathbf{z}_t)$ via VJPs, and instead enforces data consistency through optimization steps at selected timesteps.

B.1 PIXEL-SPACE UPDATE AS AN APPROXIMATE MAP / PROXIMAL STEP

The key ingredient is a local Gaussian approximation of the conditional distribution induced by the diffusion prior.

Assumption B.1 (Local Gaussian conditional)**.** At timestep t, conditioned on the noisy state $Z_t = \mathbf{z}_t$, the diffusion model induces an approximate conditional for the clean image in pixel space of the form

$$q_t(\mathbf{x} \mid \mathbf{z}_t) \;\propto\; \exp\!\left(-\frac{\lambda_t}{2}\|\mathbf{x} - \tilde{\mathbf{x}}_0(\mathbf{z}_t)\|^2\right), \tag{7}$$

where $\tilde{\mathbf{x}}_0(\mathbf{z}_t) = \mathcal{D}(\hat{\mathbf{x}}_0^\theta(\mathbf{z}_t, t))$ is the decoded denoiser prediction and $\lambda_t \geq 0$ is a (time-dependent) precision. In particular, for $\lambda_t > 0$, this corresponds to an isotropic Gaussian centered at $\tilde{\mathbf{x}}_0(\mathbf{z}_t)$, so $\tilde{\mathbf{x}}_0(\mathbf{z}_t)$ is treated as the mean parameter of the approximate prior $q_t(\cdot \mid \mathbf{z}_t)$.

Proposition B.2 (Pixel-space objective as approximate conditional MAP)**.** *Assume the Gaussian likelihood in* (5) *and Assumption B.1. Then the maximizer of the approximate conditional posterior* $q_t(\mathbf{x} \mid \mathbf{z}_t, \mathbf{y}) \propto \exp\!\Big(-\frac{1}{2\sigma_y^2}\|\mathbf{y} - \mathcal{A}(\mathbf{x})\|^2\Big)\; q_t(\mathbf{x} \mid \mathbf{z}_t)$ *is given by the solution of*

$$\mathbf{x}_t^* \in \operatorname*{argmin}_{\mathbf{x}} \; \frac{1}{2\sigma_y^2}\|\mathbf{y} - \mathcal{A}(\mathbf{x})\|^2 \;+\; \frac{\lambda_t}{2}\|\mathbf{x} - \tilde{\mathbf{x}}_0(\mathbf{z}_t)\|^2. \tag{8}$$

Proof. By Bayes' rule, the conditional posterior is proportional to likelihood times prior:

$$q_t(\mathbf{x} \mid \mathbf{z}_t, \mathbf{y}) \propto p(\mathbf{y} \mid \mathbf{x})\, q_t(\mathbf{x} \mid \mathbf{z}_t).$$

Substituting the likelihood from Eq. (5) (noting $p(\mathbf{y} \mid \mathbf{x}) \propto \exp(-\frac{1}{2\sigma_y^2}\|\mathbf{y} - \mathcal{A}(\mathbf{x})\|^2)$) and the approximate prior from Assumption B.1, we have

$$q_t(\mathbf{x} \mid \mathbf{z}_t, \mathbf{y}) \propto \exp\!\left(-\frac{1}{2\sigma_y^2}\|\mathbf{y} - \mathcal{A}(\mathbf{x})\|^2\right) \cdot \exp\!\left(-\frac{\lambda_t}{2}\|\mathbf{x} - \tilde{\mathbf{x}}_0(\mathbf{z}_t)\|^2\right).$$

Combining the exponents, the posterior density is proportional to

$$\exp\!\left(-\left[\frac{1}{2\sigma_y^2}\|\mathbf{y} - \mathcal{A}(\mathbf{x})\|^2 + \frac{\lambda_t}{2}\|\mathbf{x} - \tilde{\mathbf{x}}_0(\mathbf{z}_t)\|^2\right]\right).$$

The MAP estimate maximizes this probability, which is equivalent to minimizing the negative log-posterior (the expression in square brackets). Thus, the solution is characterized by

$$\mathbf{x}_t^* \in \operatorname*{argmin}_{\mathbf{x}} \left(\frac{1}{2\sigma_y^2} \|\mathbf{y} - \mathcal{A}(\mathbf{x})\|^2 + \frac{\lambda_t}{2} \|\mathbf{x} - \tilde{\mathbf{x}}_0(\mathbf{z}_t)\|^2 \right),$$

which corresponds exactly to (8). □

Eq. (8) matches the pixel-space subproblem used in Phase 2 (Algorithm 1). In this interpretation, λ_t controls the strength of the local prior term: larger t typically corresponds to higher uncertainty in the denoiser prediction, suggesting smaller λ_t, while as $t \to 0$ one expects λ_t to increase. In our experiments, we often set $\lambda_t = 0$ and rely on the implicit regularization from the diffusion dynamics; nevertheless, Proposition B.2 clarifies the role of λ_t as a principled prior precision.

B.2 Proximal form, closed-form solutions, and stability

This subsection unpacks the pixel-space optimization subproblem used by G&G. Intuitively, we take the decoded denoiser prediction $\tilde{\mathbf{x}}_0(\mathbf{z}_t)$ as a *proposal* image and then "pull it back" toward data consistency by solving a simple quadratic-regularized objective. This is exactly a proximal step, which immediately gives useful stability intuition.

Takeaway.

- The update in (8) is a proximal map applied to $\tilde{\mathbf{x}}_0(\mathbf{z}_t)$.
- If $\mathcal{A}$ is linear, the update has a closed form (ridge/Tikhonov regression).
- For convex data terms, the map is stable: small changes in $\tilde{\mathbf{x}}_0$ lead to small changes in $\mathbf{x}_t^*$.

Define the data-fidelity term $f(\mathbf{x}) \coloneqq \frac{1}{2\sigma_y^2} \|\mathbf{y} - \mathcal{A}(\mathbf{x})\|^2$. For $\gamma > 0$, the proximal map of f is

$$\operatorname{prox}_{\gamma f}(\mathbf{v}) \coloneqq \operatorname*{argmin}_{\mathbf{x}} \; f(\mathbf{x}) + \frac{1}{2\gamma} \|\mathbf{x} - \mathbf{v}\|^2. \tag{9}$$

Corollary B.3 (Proximal interpretation)**.** *Assume f is convex and $\lambda_t > 0$. Then the unique minimizer of* (8) *satisfies*

$$\mathbf{x}_t^* = \operatorname{prox}_{\frac{1}{\lambda_t} f}(\tilde{\mathbf{x}}_0(\mathbf{z}_t)) \,. \tag{10}$$

Proof. By definition (9), $\operatorname{prox}_{\gamma f}(\mathbf{v})$ minimizes $f(\mathbf{x}) + \frac{1}{2\gamma} \|\mathbf{x} - \mathbf{v}\|^2$. Setting $\mathbf{v} = \tilde{\mathbf{x}}_0(\mathbf{z}_t)$ and $\gamma = 1/\lambda_t$ makes this objective identical to (8). □

Remark B.4 (Linear closed form). If $\mathcal{A}(\mathbf{x}) = \mathbf{A}\mathbf{x}$, (8) admits the closed-form solution:

$$\mathbf{x}_t^* = \left(\frac{1}{\sigma_y^2} \mathbf{A}^\top \mathbf{A} + \lambda_t \mathbf{I} \right)^{-1} \left(\frac{1}{\sigma_y^2} \mathbf{A}^\top \mathbf{y} + \lambda_t \tilde{\mathbf{x}}_0(\mathbf{z}_t) \right), \tag{11}$$

making the trade-off between data consistency and local prior explicit.

Remark B.5 (Stability). For convex f, the proximal map $\operatorname{prox}_{\gamma f}$ is 1-Lipschitz (nonexpansive) (Parikh & Boyd, 2014; Bauschke & Combettes, 2017). Concretely, changing the proposal $\tilde{\mathbf{x}}_0(\mathbf{z}_t)$ by δ can change the optimizer $\mathbf{x}_t^*$ by at most $\|\delta\|$. This formalizes the intuition that the optimization step is a "stable correction" of the denoiser prediction.

Remark B.6 (Scaling λ_t). Since the effective noise on $\mathbf{z}_0$ given $\mathbf{z}_t$ scales as σ_t/α_t, a natural heuristic is

$$\lambda_t \propto \left(\frac{\alpha_t}{\sigma_t} \right)^2, \tag{12}$$

which increases with SNR ($t \to 0$), justifying time-dependent regularization. Note that σ_y^2 and λ_t appear only through their relative weighting in (8), so one can view λ_t as an effective (time-dependent) trade-off parameter.

Remark B.7 (Guidance schedule). Interpreting λ_t as a proxy for SNR, guidance is typically most effective at intermediate noise levels. When $t \to 0$ (high SNR), the diffusion prior/denoiser is already confident and can be "too rigid": heavy optimization may introduce artifacts that the remaining denoising steps cannot easily correct. When $t \to 1$ (low SNR), the denoiser prediction is highly uncertain, so the quadratic anchoring term in (8) becomes less informative. This motivates concentrating optimization at intermediate timesteps (Appendix C).

B.3 Inexact Gibbs / alternating update viewpoint

This viewpoint explains G&G as a simple "alternate between prior and likelihood" routine at a fixed noise level.

At a fixed timestep t, consider the augmented latent posterior over $(\mathbf{z}_0, \mathbf{z}_t)$:

$$\pi(\mathbf{z}_0, \mathbf{z}_t \mid \mathbf{y}) \;\propto\; \ell_0(\mathbf{y} \mid \mathbf{z}_0)\, p_0(\mathbf{z}_0)\, p(\mathbf{z}_t \mid \mathbf{z}_0), \tag{13}$$

where $p(\mathbf{z}_t \mid \mathbf{z}_0) = \mathcal{N}(\alpha_t \mathbf{z}_0, \sigma_t^2 \mathbf{I})$ is the forward noising kernel.

A (hypothetical) exact Gibbs sampler for (13) would alternate between sampling $\mathbf{z}_0 \sim \pi(\mathbf{z}_0 \mid \mathbf{z}_t, \mathbf{y})$ and $\mathbf{z}_t \sim p(\mathbf{z}_t \mid \mathbf{z}_0)$. The first conditional is intractable in general. G&G can be seen as an *inexact* replacement of this step:

1. **Prior prediction:** use the pretrained denoiser to produce $\hat{\mathbf{z}}_0 = \hat{\mathbf{x}}_0^\theta(\mathbf{z}_t, t)$ (a point estimate for $\mathbf{z}_0 \mid \mathbf{z}_t$), and decode to $\tilde{\mathbf{x}}_0 = \mathcal{D}(\hat{\mathbf{z}}_0)$.
2. **Data consistency:** incorporate $\mathbf{y}$ by solving the MAP/prox problem (8), which outputs $\mathbf{x}_t^*$ (Proposition B.2 and Corollary B.3).
3. **Re-noise:** map $\mathbf{x}_t^*$ back to the noisy manifold by sampling from the forward kernel, while preserving the current noise realization as much as possible (Lemma B.15).

This yields an intuitive "prior $\to$ data-consistency $\to$ re-noise" loop.

Remark B.8 (Phase 1 as repeated approximate conditioning at t_*). Phase 1 fixes a single noise level t_* and repeats the inexact update above for N iterations. This can be viewed as iteratively refining an approximate sample from the posterior time-marginal $\pi_{t_*}(Z_{t_*} \mid \mathbf{y})$ using repeated "(prior prediction $\to$ prox data-consistency $\to$ re-noise)" cycles, so that Phase 2 can start closer to the conditional manifold while skipping expensive steps from $t = 1$.

B.4 Fixed-point convergence of an idealized deterministic loop

The full G&G algorithm is stochastic and time-varying (it includes re-noising and a schedule over timesteps). To provide complementary intuition, we now discuss a standard idealization from plug-and-play / proximal analyses: freeze a single timestep t and study a deterministic "denoise + data-consistency" iteration (Parikh & Boyd, 2014; Venkatakrishnan et al., 2013).

Remark B.9 (How to read this subsection). The assumptions below (nonexpansive D_t, convex/strongly convex f) are not meant to precisely model neural denoisers. They are included only to connect G&G's core step to classical fixed-point arguments and to explain why the loop is typically stable in practice.

Let $D_t : \mathbb{R}^d \to \mathbb{R}^d$ denote a generic *prior prediction* operator at timestep t that returns a clean image prediction (e.g., $D_t(\cdot)$ abstracts the decoded denoiser prediction used by the algorithm). Define the data-consistency map

$$P_{\lambda_t}(\mathbf{v}) \coloneqq \operatorname*{argmin}_{\mathbf{x}} \; f(\mathbf{x}) + \frac{\lambda_t}{2}\|\mathbf{x} - \mathbf{v}\|^2 \;=\; \operatorname{prox}_{\frac{1}{\lambda_t} f}(\mathbf{v}), \tag{14}$$

where $f(\mathbf{x}) = \frac{1}{2\sigma_y^2}\|\mathbf{y} - \mathcal{A}(\mathbf{x})\|^2$ and prox is defined in (9).

Lemma B.10 (Nonexpansiveness of the data-consistency map)**.** *If f is convex, then P_{λ_t} is 1-Lipschitz.*

Proof. Let $\mathbf{x} = P_{\lambda_t}(\mathbf{v})$ and $\mathbf{x}' = P_{\lambda_t}(\mathbf{v}')$. By first-order optimality (subgradient optimality for a convex objective), there exist $u \in \partial f(\mathbf{x})$ and $u' \in \partial f(\mathbf{x}')$ such that

$$u + \lambda_t(\mathbf{x} - \mathbf{v}) = 0, \qquad u' + \lambda_t(\mathbf{x}' - \mathbf{v}') = 0. \tag{15}$$

Subtracting the two relations gives

$$(u - u') + \lambda_t(\mathbf{x} - \mathbf{x}') = \lambda_t(\mathbf{v} - \mathbf{v}'). \tag{16}$$

Taking the inner product with $(\mathbf{x} - \mathbf{x}')$ yields

$$\langle u - u', \mathbf{x} - \mathbf{x}' \rangle + \lambda_t \|\mathbf{x} - \mathbf{x}'\|^2 = \lambda_t \langle \mathbf{v} - \mathbf{v}', \mathbf{x} - \mathbf{x}' \rangle. \tag{17}$$

Since f is convex, its subdifferential ∂f is monotone, so $\langle u - u', \mathbf{x} - \mathbf{x}' \rangle \geq 0$ (Bauschke & Combettes, 2017; Parikh & Boyd, 2014). Dropping this nonnegative term in (17) and applying Cauchy–Schwarz gives

$$\lambda_t \|\mathbf{x} - \mathbf{x}'\|^2 \leq \lambda_t \|\mathbf{v} - \mathbf{v}'\| \, \|\mathbf{x} - \mathbf{x}'\|.$$

If $\mathbf{x} = \mathbf{x}'$ the claim holds trivially. Otherwise, divide by $\lambda_t \|\mathbf{x} - \mathbf{x}'\|$ to obtain $\|\mathbf{x} - \mathbf{x}'\| \leq \|\mathbf{v} - \mathbf{v}'\|$. Therefore P_{λ_t} is 1-Lipschitz. □

Consider the composite operator $T_t \coloneqq P_{\lambda_t} \circ D_t$ and the relaxed iteration

$$\mathbf{x}^{k+1} = (1 - \rho)\mathbf{x}^k + \rho \, T_t(\mathbf{x}^k), \qquad \rho \in (0, 1). \tag{18}$$

Assumption B.11 (Nonexpansive prior operator)**.** The prior prediction operator D_t is 1-Lipschitz.

Theorem B.12 (Convergence of the relaxed fixed-point iteration)**.** *Assume f is convex, Assumption B.11 holds, and $\mathrm{Fix}(T_t) \neq \varnothing$. Then the iteration* (18) *converges to a fixed point of T_t.*

Proof. First, P_{λ_t} is nonexpansive by Lemma B.10, and D_t is nonexpansive by Assumption B.11. Therefore, for any $\mathbf{x}, \mathbf{x}'$,

$$\|T_t(\mathbf{x}) - T_t(\mathbf{x}')\| = \|P_{\lambda_t}(D_t(\mathbf{x})) - P_{\lambda_t}(D_t(\mathbf{x}'))\| \leq \|D_t(\mathbf{x}) - D_t(\mathbf{x}')\| \leq \|\mathbf{x} - \mathbf{x}'\|,$$

so T_t is nonexpansive.

Define the relaxed map $S_t(\mathbf{x}) \coloneqq (1 - \rho)\mathbf{x} + \rho \, T_t(\mathbf{x})$, so that (18) is $\mathbf{x}^{k+1} = S_t(\mathbf{x}^k)$. Note that $\mathrm{Fix}(S_t) = \mathrm{Fix}(T_t)$: if $\mathbf{x} = S_t(\mathbf{x})$, then $(1 - \rho)\mathbf{x} + \rho T_t(\mathbf{x}) = \mathbf{x}$ implies $T_t(\mathbf{x}) = \mathbf{x}$, and the converse is immediate.

Since S_t is the Krasnosel'skiĭ–Mann relaxation of the nonexpansive map T_t, the Krasnosel'skiĭ–Mann fixed-point theorem ensures that $(\mathbf{x}^k)_k$ converges to a point in $\mathrm{Fix}(T_t)$ whenever $\mathrm{Fix}(T_t) \neq \varnothing$ (in finite-dimensional Hilbert spaces) (Krasnosel'skii, 1955; Mann, 1953; Bauschke & Combettes, 2017). □

Theorem B.12 connects G&G's core step to standard plug-and-play analyses: guidance can be viewed as seeking a fixed point of a data-consistency proximal map composed with a prior operator (Venkatakrishnan et al., 2013).

Theorem B.13 (Linear convergence under strong convexity)**.** *Assume f is μ-strongly convex and D_t is L_t-Lipschitz. Then P_{λ_t} is $\frac{1}{1+\mu/\lambda_t}$-Lipschitz, and T_t is $\frac{L_t}{1+\mu/\lambda_t}$-Lipschitz. In particular, if $\frac{L_t}{1+\mu/\lambda_t} < 1$, then T_t is a contraction with a unique fixed point $\mathbf{x}^\star$, and the iteration $\mathbf{x}^{k+1} = T_t(\mathbf{x}^k)$ converges to $\mathbf{x}^\star$ at a linear rate.*

Proof. Let $\mathbf{x} = P_{\lambda_t}(\mathbf{v})$ and $\mathbf{x}' = P_{\lambda_t}(\mathbf{v}')$. As in Lemma B.10, there exist $u \in \partial f(\mathbf{x})$ and $u' \in \partial f(\mathbf{x}')$ satisfying the optimality conditions (15), and hence (17) holds:

$$\langle u - u', \mathbf{x} - \mathbf{x}' \rangle + \lambda_t \|\mathbf{x} - \mathbf{x}'\|^2 = \lambda_t \langle \mathbf{v} - \mathbf{v}', \mathbf{x} - \mathbf{x}' \rangle.$$

If f is μ-strongly convex, then ∂f is μ-strongly monotone, meaning $\langle u - u', \mathbf{x} - \mathbf{x}' \rangle \geq \mu \|\mathbf{x} - \mathbf{x}'\|^2$ (Bauschke & Combettes, 2017; Parikh & Boyd, 2014). Plugging this into the previous identity gives

$$(\mu + \lambda_t)\|\mathbf{x} - \mathbf{x}'\|^2 \leq \lambda_t \langle \mathbf{v} - \mathbf{v}', \mathbf{x} - \mathbf{x}' \rangle \leq \lambda_t \|\mathbf{v} - \mathbf{v}'\| \, \|\mathbf{x} - \mathbf{x}'\|.$$

If $\mathbf{x} = \mathbf{x}'$ the claim is trivial; otherwise dividing by $(\mu + \lambda_t)\|\mathbf{x} - \mathbf{x}'\|$ yields

$$\|\mathbf{x} - \mathbf{x}'\| \leq \frac{\lambda_t}{\mu + \lambda_t} \|\mathbf{v} - \mathbf{v}'\| = \frac{1}{1 + \mu/\lambda_t} \|\mathbf{v} - \mathbf{v}'\|.$$

Thus P_{λ_t} is $\frac{1}{1+\mu/\lambda_t}$-Lipschitz.

For $T_t = P_{\lambda_t} \circ D_t$, we have for any $\mathbf{x}, \mathbf{x}'$:

$$\|T_t(\mathbf{x}) - T_t(\mathbf{x}')\| \le \frac{1}{1+\mu/\lambda_t}\|D_t(\mathbf{x}) - D_t(\mathbf{x}')\| \le \frac{L_t}{1+\mu/\lambda_t}\|\mathbf{x} - \mathbf{x}'\|.$$

Therefore, if $q \coloneqq \frac{L_t}{1+\mu/\lambda_t} < 1$, then T_t is a contraction. By Banach's fixed-point theorem, T_t has a unique fixed point $\mathbf{x}^\star$ and the iterates $\mathbf{x}^{k+1} = T_t(\mathbf{x}^k)$ satisfy $\|\mathbf{x}^k - \mathbf{x}^\star\| \le q^k\|\mathbf{x}^0 - \mathbf{x}^\star\|$, which is linear convergence (Bauschke & Combettes, 2017). □

Corollary B.14 (Robustness to inexact updates)**.** *Under the assumptions of Theorem B.13 with contraction factor $q < 1$, consider inexact iterates $\mathbf{x}^{k+1} = T_t(\mathbf{x}^k) + e_k$ with $\|e_k\| \le \varepsilon$. Then* $\limsup_{k\to\infty} \|\mathbf{x}^k - \mathbf{x}^\star\| \le \varepsilon/(1-q)$.

Proof. Since T_t is a contraction and $T_t(\mathbf{x}^\star) = \mathbf{x}^\star$, we have

$$\|\mathbf{x}^{k+1} - \mathbf{x}^\star\| \le \|T_t(\mathbf{x}^k) - T_t(\mathbf{x}^\star)\| + \|e_k\| \le q\|\mathbf{x}^k - \mathbf{x}^\star\| + \varepsilon.$$

Define $a_k \coloneqq \|\mathbf{x}^k - \mathbf{x}^\star\|$. Then $a_{k+1} \le qa_k + \varepsilon$. Unrolling the recursion gives

$$a_k \le q^k a_0 + \varepsilon \sum_{i=0}^{k-1} q^i = q^k a_0 + \varepsilon\,\frac{1-q^k}{1-q}.$$

Taking $\limsup_{k\to\infty}$ and using $q^k \to 0$ yields $\limsup_{k\to\infty} a_k \le \varepsilon/(1-q)$. □

B.5 Re-noising as an optimal coupling (noise preservation)

After updating the clean estimate via optimization, the algorithm re-noises to return to a noisy manifold (Phase 1 and Phase 2). A useful perspective is that re-noising should change the mean induced by the updated clean state *while minimally perturbing the current sample*, which is naturally formalized via optimal couplings of Gaussians.

Lemma B.15 (Optimal coupling of equal-covariance Gaussians)**.** *Let $P = \mathcal{N}(\mathbf{m}, \Sigma)$ and $P' = \mathcal{N}(\mathbf{m}', \Sigma)$ with the same covariance $\Sigma \succ 0$. The coupling $X = \mathbf{m} + \xi,\ X' = \mathbf{m}' + \xi$ with $\xi \sim \mathcal{N}(0, \Sigma)$ minimizes $\mathbb{E}\|X - X'\|^2$ among all couplings of (P, P'), and achieves* $\inf \mathbb{E}\|X - X'\|^2 = \|\mathbf{m} - \mathbf{m}'\|^2$.

Proof. Let (X, X') be any coupling of (P, P'). Write $X = \mathbf{m} + \xi$ and $X' = \mathbf{m}' + \xi'$ where ξ, ξ' have mean 0 and covariance Σ. Then

$$\mathbb{E}\|X - X'\|^2 = \mathbb{E}\|(\mathbf{m} - \mathbf{m}') + (\xi - \xi')\|^2 = \|\mathbf{m} - \mathbf{m}'\|^2 + \mathbb{E}\|\xi - \xi'\|^2,$$

where the cross term vanishes because $\mathbb{E}[\xi - \xi'] = 0$. Since $\mathbb{E}\|\xi - \xi'\|^2 \ge 0$, we obtain the lower bound $\mathbb{E}\|X - X'\|^2 \ge \|\mathbf{m} - \mathbf{m}'\|^2$.

This bound is achievable by the shared-noise coupling: draw $\xi \sim \mathcal{N}(0, \Sigma)$ and set $X = \mathbf{m} + \xi$ and $X' = \mathbf{m}' + \xi$. Then $X \sim P$, $X' \sim P'$, and $\mathbb{E}\|X - X'\|^2 = \|\mathbf{m} - \mathbf{m}'\|^2$. Equivalently, this is the optimal transport plan for quadratic cost between equal-covariance Gaussians (Villani, 2009). □

Lemma B.15 motivates *noise preservation*: when the clean estimate changes (e.g., from $\tilde{\mathbf{x}}_0$ to $\mathbf{x}_t^*$), the least-disruptive way to return to the noisy manifold is to keep the same underlying noise realization and only update the mean term. In practice the true noise is not observed; we therefore approximate it using the predicted noise extracted from the denoiser output (see Appendix A). We additionally mix this predicted noise with fresh Gaussian noise, which preserves the marginal covariance while providing controlled stochastic refresh.

C SAMPLING SCHEDULE

Motivation: In Phase 2 of our method, we perform optimization at a selected subset of M timesteps from the diffusion trajectory. A critical design choice is **where** along this trajectory to concentrate our computational effort. The spacing of these timesteps—controlled by our scheduling strategy—has a significant impact on reconstruction quality. **Key insight**: Optimization applied too late in the denoising process (near $t = 0$) has limited opportunity for the denoiser to refine artifacts, while optimization applied too early (near t_*) may be wasted on high-noise states. Our scheduling strategies allow flexible allocation of computational budget across the denoising trajectory.

C.1 TIMESTEP ALLOCATION FRAMEWORK

Given M guidance steps and starting timestep $t_* = T_{\text{start}}$, we construct a grid of timesteps $\{t_1, t_2, \ldots, t_M\}$ where $t_M = t_*$ and $t_1 > 0$. Note that the timesteps are in **decreasing order**: $t_M > t_{M-1} > \cdots > t_1$.

Weight-based allocation. Rather than spacing timesteps uniformly, we assign a weight w_i to each position $i \in \{1, \ldots, M\}$, then allocate timesteps proportionally to these weights. Higher weights w_i result in larger spacing allocated to that region of the trajectory. We first compute the cumulative timestep decrements:

$$C_i = \left\lfloor (T_{\text{start}} - 1) \cdot \frac{\sum_{j=1}^{i} w_j}{\sum_{j=1}^{M} w_j} \right\rfloor$$

The spacing between consecutive timesteps is then $\Delta_i = C_i - C_{i-1}$, where $C_0 = 0$. Finally, the timesteps themselves are computed in decreasing order $t_M = T_{\text{start}}$

$$t_k = T_{\text{start}} - \sum_{j=k+1}^{M} \Delta_j, \quad k = 1, \ldots, M$$

C.2 SCHEDULING STRATEGIES

We explore five scheduling strategies that concentrate optimization steps at different noise levels. The weight functions for each strategy are summarized in Table 6.

Table 6: Weight functions for different scheduling strategies.

Strategy	**Weight Function** w_i	**Parameters**
Uniform	1	—
Linear	$i + 1$	—
Polynomial	$(i + 1)^p$	$p > 1$
Exponential	k^i	$k > 1$
Gaussian	$\exp\left(-\frac{(i/M-\mu)^2}{2\sigma^2}\right)$	$\mu \in [0, 1], \sigma > 0$
Beta	$\text{Beta}(i/M; \alpha, \beta)^{-1}$	$\alpha, \beta > 0$

Uniform Schedule: $w_i = 1$ for all i. Equal spacing between all guidance steps. This serves as our baseline, distributing computational effort uniformly across the denoising trajectory.

Linear Schedule: $w_i = i + 1$. Progressively allocates more timesteps toward later iterations (lower noise levels). Weights increase linearly, so the spacing between guidance steps grows as we approach cleaner states.

Polynomial Schedule: $w_i = (i + 1)^p$ with power $p > 1$. Similar to linear but with adjustable aggressiveness controlled by power p. Higher values of p create more dramatic concentration toward later iterations. Common choices include $p = 2$ (quadratic growth, moderate concentration) or $p = 3$ (cubic growth, aggressive concentration). We find out this schedule works best for `FFHQ LDM`.

Exponential Schedule: $w_i = k^i$ with decay rate $k > 1$. Rapidly concentrates timesteps toward later iterations using exponential growth. The decay rate k controls how aggressively we shift toward low-noise regions. Typical values are $k \in [1.5, 2.0]$.

Gaussian Schedule (Default):

$$w_i = \exp\left(-\frac{(i/M - \mu)^2}{2\sigma^2}\right)$$

with center $\mu \in [0, 1]$ and width $\sigma > 0$. Concentrates timesteps around a specific relative position μ along the trajectory, with spread controlled by σ. This allows targeting the "sweet spot" where optimization is most effective. For example, $\mu = 0.5$ centers around mid-trajectory, while smaller σ creates tighter concentration. This schedule work best for most tasks. Our experiments (Table 5) show this provides the best balance, typically with $\mu \in [0.3, 0.5]$ and $\sigma \in [10.0, 15.0]$. By concentrating optimization in the intermediate noise regime (approximately $t \in [0.4, 0.6]$), the denoiser has sufficient structure to work with while retaining enough trajectory to refine artifacts before reaching the final output.

Beta Schedule: $w_i = \text{Beta}(i/M; \alpha, \beta)^{-1}$, where

$$\text{Beta}(x; \alpha, \beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha, \beta)}$$

is the Beta distribution probability density with shape parameters $\alpha, \beta > 0$. Most flexible distribution that can model U-shaped, bell-shaped, or skewed patterns depending on α and β. Symmetric concentration occurs when $\alpha = \beta$, while $\alpha < \beta$ skews toward early iterations and $\alpha > \beta$ skews toward later iterations.

D COMPETITORS

In this section, we detail the implementation of baseline methods. We use the hyperparameters suggested by the original authors and perform additional tuning for each dataset when specific values are not provided.

DPS. We implemented the method from Chung et al. (2022a), adopting the hyperparameters for each task as specified in Chung et al. (2022a) (App. D). For tasks not covered in the original work, we performed our own tuning: specifically, we set $\psi = 0.2$ for JPEG 2%, $\psi = 0.07$ for High Dynamic Range.

DIFFPIR. We implemented Zhu et al. (2023) to ensure compatibility with our codebase, using the hyperparameters from the official released version. We attempted to extend the method to nonlinear problems following the guidelines in Zhu et al. (2023) (Eqn. (13)); however, the algorithm diverged in these cases. We were unable to resolve this issue as neither the paper nor the released code provide examples for nonlinear problems. For motion blur, Zhu et al. (2023) offers an FFT-based solution that is only applicable to circular convolution. Since we adopt the experimental setup of Chung et al. (2022a), which employs convolution with reflection padding, we exclude DIFFPIR from the motion blur evaluation.

DDNM (Wang et al., 2022). We adapted the implementation from the released code. The original code provides separate classes for each degradation operator in the module `functions/svd_operators.py`. We refactored these into a single unified class to support all SVD-decomposable linear degradation operators. We observe that DDNM exhibits instability for operators whose `SVD` decomposition is susceptible to numerical errors, such as Gaussian blur with wide convolution kernels. This instability arises from the algorithm's reliance on the pseudo-inverse of the operator.

RED-DIFF (Mardani et al., 2023). We employed the implementation of RED-DIFF from the released code. For linear problems, we initialize the variational optimization using the pseudo-inverse of the observation. For nonlinear problems, where the pseudo-inverse is unavailable, we initialize the optimization with a sample drawn from a standard Gaussian distribution.

PGDM. We use the implementation provided in the RED-DIFF repository, as several authors of RED-DIFF are also co-authors of PGDM. We note a minor deviation from the algorithm presented in Song et al. (2023b) (Algorithm 1): in the final step, the guidance term g is scaled by $\alpha_{t-1}\alpha_t$ in the implementation, whereas the original formulation scales it by $\alpha_t\sqrt{\alpha_t}$. We find that this modification improves performance across most tasks, with the exception of JPEG dequantization, for which the original α_t scaling yields better results.

PSLD. We implemented the PSLD algorithm from Kadkhodaie & Simoncelli (2020) and configured the hyperparameters for each task based on the publicly available implementation.

RESAMPLE. We modified the original code from the authors to enable direct adjustment of key hyperparameters: the tolerance ε and maximum iteration count N for the optimization problems enforcing hard data consistency, as well as the variance scaling factor γ for the stochastic resampling distribution. We observe that the algorithm is sensitive to ε, with optimal reconstructions achieved by setting it equal to the noise level of the inverse problem across all tasks and noise levels. In contrast, we find that γ has minimal impact on reconstruction quality. To reduce computational cost, we set a maximum threshold of $N = 200$ gradient iterations.

DAPS. We use the official codebase and configure the hyperparameters according to Zhang et al. (2025) (Table 7). For audio-source separation, we set $\sigma_{\max}$ and $\sigma_{\min}$ to match the values used in the sound model, and adapt the Langevin step size `lr` and standard deviation `tau` accordingly.

PNP-DM. We adapted the implementation from the released code, making the coupling parameter ρ (including its initial value, minimum value, and decay rate) and the number of Langevin steps

with their step size directly adjustable. We configure the hyperparameters following Wu et al. (2024) (Tables 3 and 4). For inpainting tasks, although exact likelihood steps are theoretically possible via Gaussian conjugacy Wu et al. (2024)(Sec. 3.1), we find that Langevin dynamics yield superior results in practice. For instance, the reconstructions in Figure 6 (left) are obtained by exact posterior sampling, whereas the right-hand side uses Langevin dynamics.

E Ablation Studies

Table 7: SSIM scores for different noise schedules across tasks

Schedule	Deblur	Motion Deblur	SR4
Uniform	0.772 ± 0.057	0.768 ± 0.057	0.772 ± 0.056
Linear	0.692 ± 0.068	0.673 ± 0.076	0.691 ± 0.073
Polynomial	0.818 ± 0.042	0.754 ± 0.041	0.785 ± 0.036
Exponential	0.823 ± 0.045	0.808 ± 0.040	0.805 ± 0.036
Beta	0.759 ± 0.034	0.523 ± 0.046	0.717 ± 0.032
Gaussian	0.825 ± 0.045	0.824 ± 0.041	0.827 ± 0.038

Table 8: PSNR scores for different noise schedules across tasks

Schedule	Deblur	Motion Deblur	SR4
Uniform	26.42 ± 1.71	26.24 ± 1.72	26.38 ± 1.69
Linear	23.63 ± 1.58	23.04 ± 1.82	23.48 ± 1.68
Polynomial	28.78 ± 1.91	27.93 ± 1.64	28.18 ± 1.60
Exponential	28.71 ± 2.04	28.52 ± 1.85	28.40 ± 1.71
Beta	27.84 ± 1.58	24.70 ± 0.92	27.19 ± 1.31
Gaussian	28.72 ± 2.00	28.57 ± 2.00	28.66 ± 1.92

F Additional Experiments

F.1 Computational Resources

All experiments are conducted on a single node with 8 NVIDIA A6000 GPUs. Performance metrics, including runtime and memory usage, are measured on GPUs without competing processes or memory allocation. CPU load is monitored throughout to ensure no performance degradation from CPU-GPU synchronization bottlenecks.

Table 9: Quantitative comparison of different methods across various inverse problems. Results are reported as SSIM ± standard deviation. Higher is better.

	SSIM FFHQ							
Task	G&G	DAPS	Red-Diff	PnP-DM	DPS	DDNM	PGDM	DiffPIR
Gauss. Deblur	0.83 ± 0.04	0.81 ± 0.05	0.81 ± 0.04	0.77 ± 0.06	0.72 ± 0.07	0.03 ± 0.01	0.14 ± 0.09	–
Mot. Deblur	0.82 ± 0.04	0.79 ± 0.05	0.74 ± 0.03	0.75 ± 0.06	0.65 ± 0.07	–	–	–
SR (×4)	0.79 ± 0.04	0.80 ± 0.05	0.63 ± 0.03	0.75 ± 0.05	0.67 ± 0.08	0.69 ± 0.03	0.56 ± 0.03	–
SR (×16)	0.63 ± 0.07	0.55 ± 0.08	0.59 ± 0.06	0.50 ± 0.07	0.50 ± 0.08	0.62 ± 0.07	0.42 ± 0.06	–
Box Inp.	0.76 ± 0.04	0.80 ± 0.03	0.70 ± 0.03	0.55 ± 0.03	0.72 ± 0.06	0.73 ± 0.03	0.70 ± 0.03	0.82 ± 0.03
Half Inp.	0.68 ± 0.05	0.71 ± 0.04	0.63 ± 0.04	0.43 ± 0.02	0.66 ± 0.06	0.65 ± 0.05	0.59 ± 0.04	0.73 ± 0.05
JPEG (QF=2)	0.77 ± 0.05	0.76 ± 0.05	0.72 ± 0.05	0.71 ± 0.06	0.60 ± 0.10	–	–	–
Phase Retr.	0.58 ± 0.21	0.65 ± 0.23	0.53 ± 0.22	0.54 ± 0.21	0.39 ± 0.16	–	–	–
HDR	0.78 ± 0.09	0.85 ± 0.07	0.72 ± 0.09	0.67 ± 0.13	0.34 ± 0.34	–	–	–
Memory (MB)	1983	2095	1985	1985	3309	2019	3409	1985
Run time (sec)	25	75	50	194	105	47	101	50

Table 10: Quantitative comparison of different methods across various inverse problems. Results are reported as PSNR ± standard deviation. Higher is better.

	PSNR FFHQ							
Task	G&G	DAPS	Red-Diff	PnP-DM	DPS	DDNM	PGDM	DiffPIR
Gauss. Deblur	29.0 ± 1.80	28.2 ± 2.08	28.7 ± 1.82	25.2 ± 2.76	24.7 ± 2.07	7.8 ± 0.10	13.2 ± 0.70	–
Mot. Deblur	28.1 ± 1.76	27.0 ± 1.86	27.8 ± 1.22	24.7 ± 2.33	22.2 ± 1.83	–	–	–
SR (×4)	28.2 ± 1.62	27.5 ± 1.76	26.1 ± 0.92	24.7 ± 2.21	22.8 ± 2.05	26.9 ± 1.17	24.6 ± 1.22	–
SR (×16)	21.6 ± 1.69	17.8 ± 1.48	21.2 ± 1.44	16.3 ± 1.10	17.9 ± 1.65	21.6 ± 1.67	18.4 ± 1.23	–
Box Inp.	21.7 ± 2.65	22.4 ± 2.90	21.6 ± 2.60	12.5 ± 0.74	20.9 ± 2.41	22.5 ± 2.78	21.1 ± 2.37	22.6 ± 3.42
Half Inp.	16.0 ± 2.57	15.4 ± 2.51	15.5 ± 2.51	10.8 ± 0.94	15.9 ± 2.44	16.1 ± 2.89	15.0 ± 2.32	16.2 ± 2.82
JPEG (QF=2)	26.2 ± 1.56	25.4 ± 1.73	24.5 ± 1.19	22.4 ± 1.49	20.4 ± 1.71	–	–	–
Phase Retr.	19.7 ± 6.82	21.6 ± 8.92	21.5 ± 7.81	18.1 ± 6.59	14.1 ± 4.32	–	–	–
HDR	22.9 ± 2.87	27.1 ± 2.96	21.7 ± 2.86	21.4 ± 2.12	12.9 ± 7.61	–	–	–
Memory (MB)	1983	2095	1985	1985	3309	2019	3409	1985
Run time (sec)	25	75	50	194	105	47	101	50

Table 11: Quantitative comparison of different methods across various inverse problems. Results are reported as SSIM ± standard deviation. Higher is better.

	SSIM ImageNet							
Task	G&G	DAPS	Red-Diff	PnP-DM	DPS	DDNM	PGDM	DiffPIR
Gauss. Deblur	0.77 ± 0.07	0.68 ± 0.13	0.69 ± 0.10	0.63 ± 0.14	0.57 ± 0.17	0.60 ± 0.15	0.07 ± 0.02	–
Mot. Deblur	0.72 ± 0.05	0.66 ± 0.13	0.65 ± 0.05	0.60 ± 0.14	0.49 ± 0.17	–	–	–
SR (×4)	0.71 ± 0.05	0.67 ± 0.13	0.59 ± 0.05	0.60 ± 0.15	0.51 ± 0.17	0.74 ± 0.10	0.27 ± 0.05	–
SR (×16)	0.58 ± 0.14	0.43 ± 0.15	0.45 ± 0.13	0.40 ± 0.14	0.34 ± 0.16	0.50 ± 0.16	0.21 ± 0.09	–
Box Inp.	0.72 ± 0.07	0.73 ± 0.05	0.36 ± 0.06	0.51 ± 0.04	0.58 ± 0.15	0.76 ± 0.05	0.61 ± 0.02	0.78 ± 0.04
Half Inp.	0.67 ± 0.10	0.65 ± 0.07	0.58 ± 0.05	0.38 ± 0.03	0.52 ± 0.14	0.66 ± 0.08	0.52 ± 0.04	0.72 ± 0.08
JPEG (QF=2)	0.75 ± 0.09	0.64 ± 0.14	0.61 ± 0.11	0.59 ± 0.14	0.46 ± 0.16	–	–	–
Phase Retr.	0.25 ± 0.14	0.36 ± 0.24	0.23 ± 0.12	0.33 ± 0.15	0.20 ± 0.11	–	–	–
HDR	0.77 ± 0.14	0.82 ± 0.11	0.72 ± 0.12	0.64 ± 0.21	0.16 ± 0.21	–	–	–
Memory (MB)	4991	4993	4995	4993	8701	5031	8741	5007
Run time (sec)	83	219	165	636	360	296	371	169

Table 12: Quantitative comparison of different methods across various inverse problems. Results are reported as PSNR ± standard deviation. Higher is better.

	PSNR ImageNet							
Task	G&G	DAPS	RED-DIFF	PNP-DM	DPS	DDNM	PGDM	DIFFPIR
Gaussian Deblur	27.3 ± 2.57	25.4 ± 3.15	25.8 ± 2.99	24.5 ± 2.83	22.6 ± 3.06	23.7 ± 3.26	9.7 ± 0.64	–
Motion Deblur	26.9 ± 2.20	24.7 ± 2.97	25.5 ± 1.97	23.4 ± 2.68	20.4 ± 2.65	–	–	–
SR (×4)	26.6 ± 1.95	25.1 ± 2.91	24.5 ± 1.88	23.8 ± 2.68	20.9 ± 3.00	26.4 ± 3.33	18.2 ± 1.80	–
SR (×16)	20.8 ± 2.78	16.9 ± 1.90	19.7 ± 1.97	15.1 ± 1.37	16.4 ± 2.21	20.4 ± 2.38	15.4 ± 1.89	–
Box Inpainting	18.3 ± 1.99	18.2 ± 2.39	18.0 ± 2.82	12.6 ± 1.00	17.1 ± 2.18	19.2 ± 2.51	16.4 ± 1.85	18.70 ± 3.23
Half Inpainting	15.9 ± 2.75	15.2 ± 3.03	14.3 ± 3.00	10.8 ± 1.51	14.0 ± 2.83	15.5 ± 3.14	13.8 ± 2.09	16.56 ± 4.09
JPEG (QF=2)	25.6 ± 2.26	23.7 ± 2.63	22.9 ± 1.95	21.6 ± 2.00	18.6 ± 2.60	–	–	–
Phase Retrieval	12.6 ± 4.43	14.6 ± 6.76	14.2 ± 4.65	13.0 ± 3.24	11.9 ± 2.44	–	–	–
High Dynamic Range	23.2 ± 4.04	25.6 ± 3.65	22.5 ± 3.31	22.1 ± 3.96	7.98± 3.95	–	–	–
Memory (MB)	4991	4993	4995	4993	8701	5031	8741	5007
Run time (sec)	83	219	165	636	360	296	371	169

G HYPERPARAMETERS

Table 13: Hyperparameters for all tasks on the `ImageNet` and `FFHQ` datasets. Unless stated otherwise, all experiments use $\eta_{\text{main}} = 10^{-3}$, $\lambda = 0.0$, $\eta_{\text{interleave}} = 1.0$, $N_{\texttt{OPT}}^{\texttt{int}} = 50$, $\eta_{\text{init}} = 10^{-4}$, $\mu_G = 0.4$, $\sigma_G = 10.0$, $M = 30$, and a Gaussian schedule. Exceptions: η_{init} is 2×10^{-4} for Box Inpainting on ImageNet; μ_G is $0.4/0.5$ for Gaussian Blur, $0.3/0.4$ for Box Inpainting, and 0.5 for SR ($\times$16) and HDR; σ_G is 15.0 for SR ($\times$16) on FFHQ; M is $40/20$ for SR ($\times$4) and 20 for SR ($\times$16) and Phase Retrieval. For each entry, we report `ImageNet / FFHQ` when the datasets differ; a single value indicates the same hyperparameter is used for both datasets.

Task	t_*	N	$G_{\texttt{OPT}}$
Gaussian Blur	0.44 / 0.50	10	25 / 50
Motion Blur	0.44	10	25 / 50
SR ($\times$4)	0.50	5 / 10	75 / 100
SR ($\times$16)	0.50 / 0.60	1 / 30	200
Box Inpainting	0.70	5	50
Half Inpainting	0.80	30	50
JPEG (QF=2)	0.44 / 0.60	5 / 20	100 / 200
Phase Retrieval	0.70	20	100
HDR	0.44	10	25 / 100

Table 14: Hyperparameters for all tasks on the `FFHQ` dataset using Latent Diffusion Models. Unless stated otherwise, all experiments use $\lambda = 0.0$, $\eta_{\text{init}} = 10^{-4}$, $\eta_{\text{main}} = 10^{-4}$, $\eta_{\text{interleave}} = 1.0$, $N = 30$, $M = 10$, and a polynomial schedule $\textsc{Poly}(p)$. Exceptions: $\eta_{\text{interleave}} = 0.0$ for Gaussian Blur; $N = 10$ for Motion Blur and SR ($\times$4) and $N = 20$ for Box Inpainting; $M = 7$ for SR ($\times$4), $M = 5$ for SR ($\times$16), and $M = 20$ for Box Inpainting, Phase Retrieval, and HDR; $\eta_{\text{main}} = 4 \times 10^{-4}$ for Gaussian Blur and $\eta_{\text{main}} = 5 \times 10^{-4}$ for JPEG (QF=2) and HDR.

Task	t_*	$N_{\texttt{OPT}}^{\texttt{int}}$	$G_{\texttt{OPT}}$	p
Gaussian Blur	0.70	50	3000	2.5
Motion Blur	0.50	300	3000	2.0
SR ($\times$4)	0.30	300	2000	2.5
SR ($\times$16)	0.40	300	1000	4.0
Box Inpainting	0.44	800	800	2.5
Half Inpainting	0.44	1000	800	3.0
JPEG (QF=2)	0.40	200	1000	3.0
Phase Retrieval	0.60	50	500	3.0
HDR	0.70	500	3000	3.0

H RECONSTRUCTION SAMPLES ON FFHQ

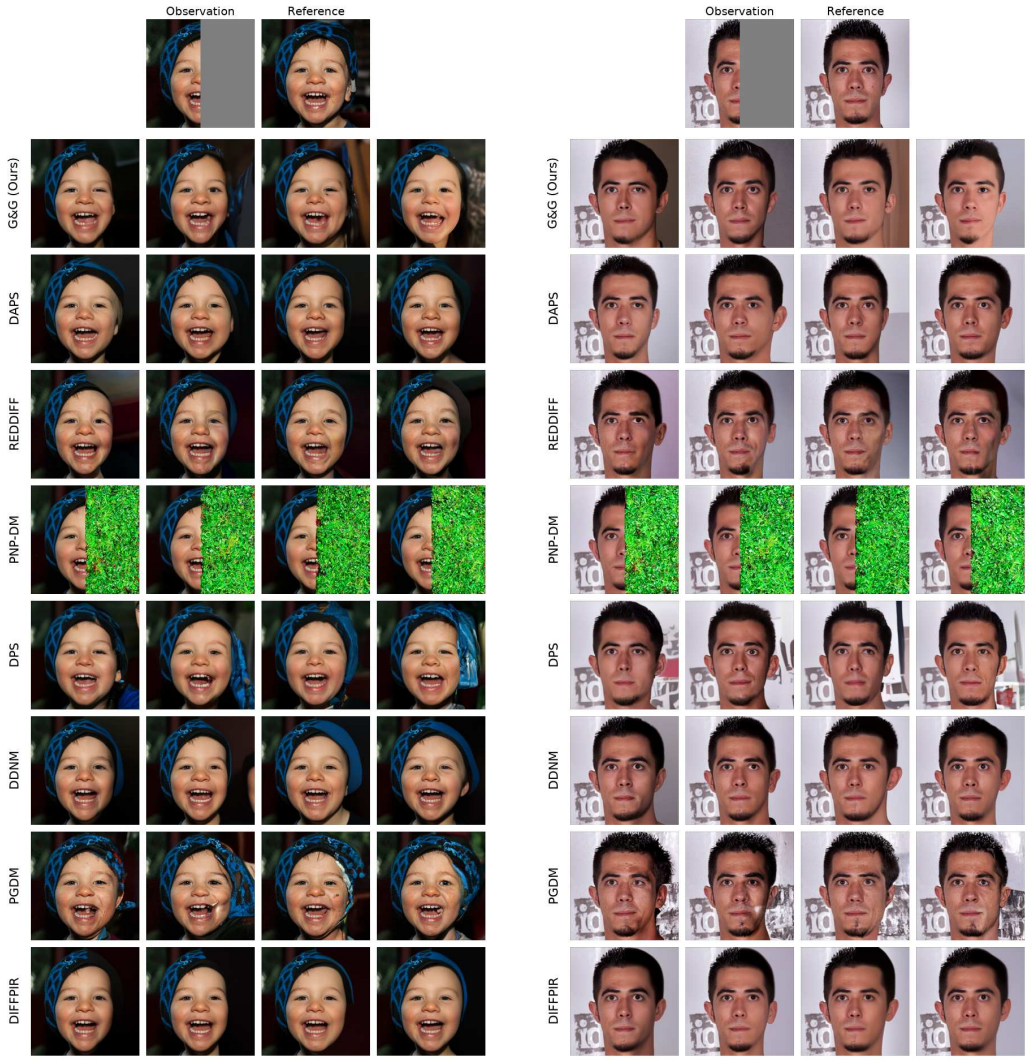

Figure 3: Reconstructions for half mask inpainting on FFHQ dataset.

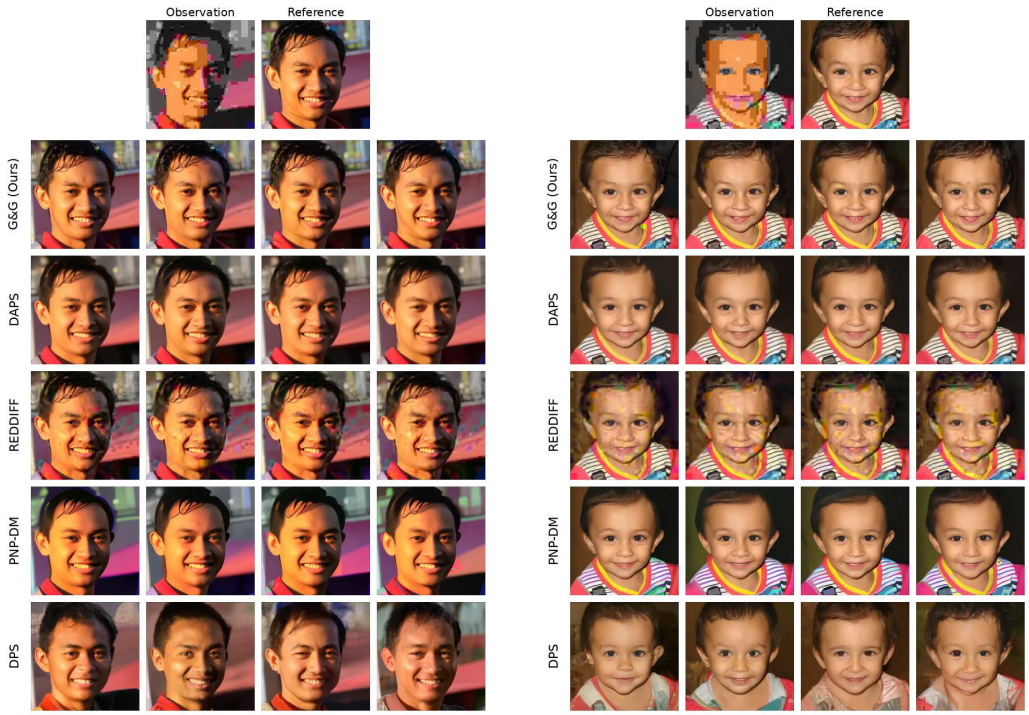

Figure 4: JPEG dequantization with QF = 2 on FFHQ dataset.

Figure 5: Reconstructions for Gaussian deblurring on `ImageNet` dataset.

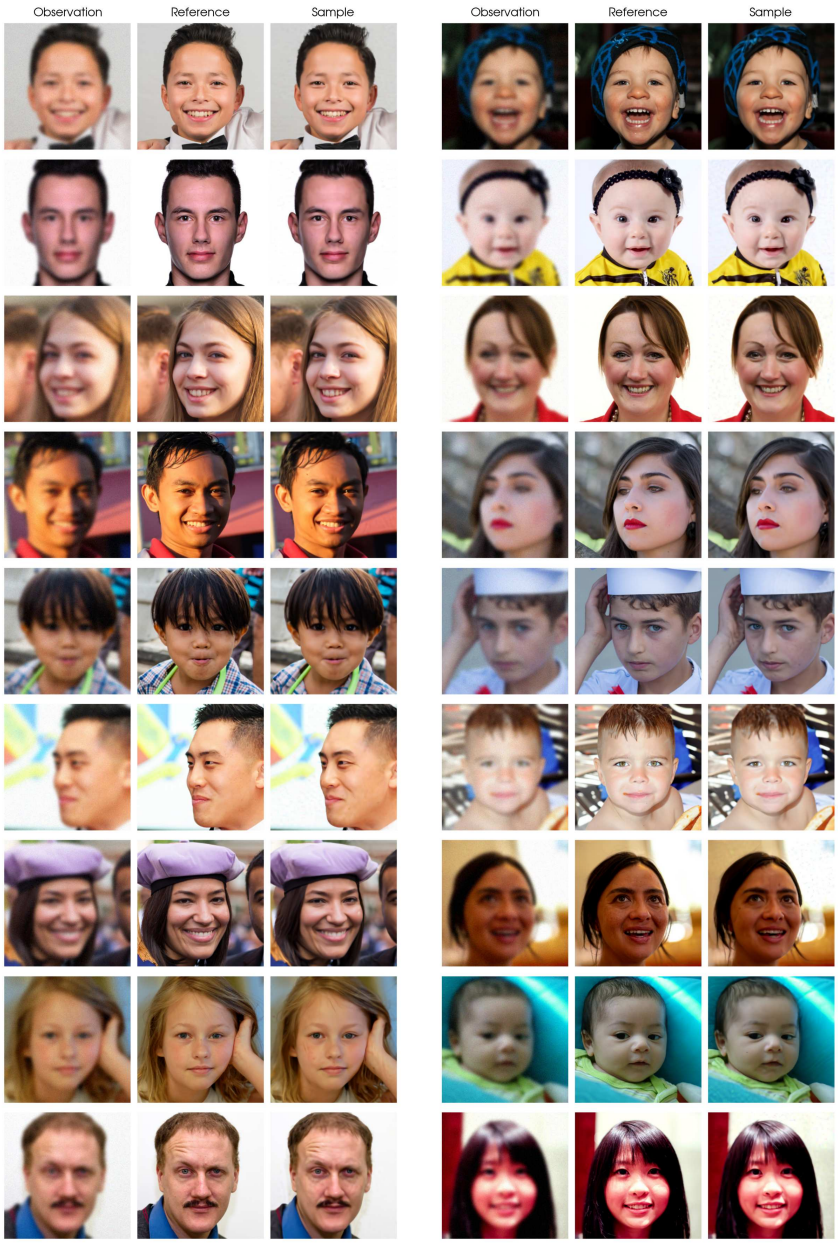

Figure 6: Gaussian Deblurring on FFHQ dataset.

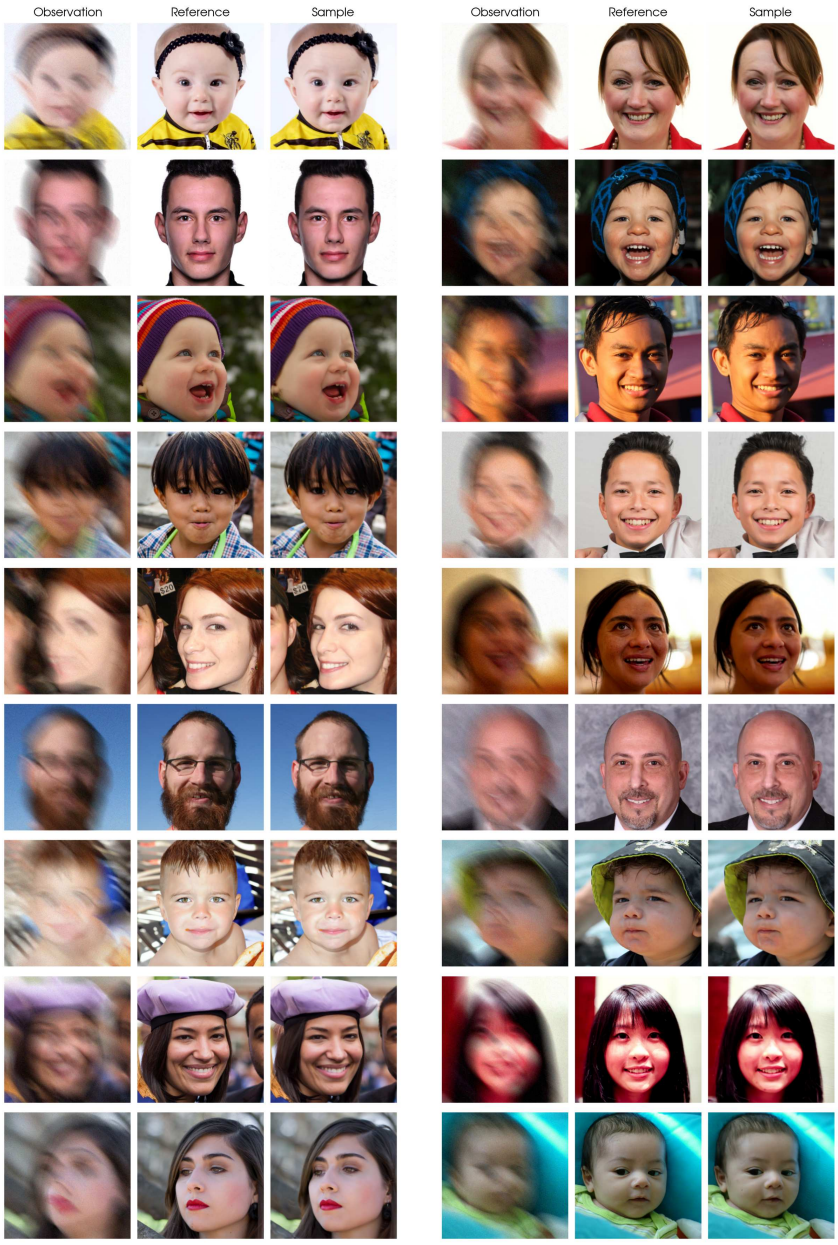

Figure 7: Motion Deblurring on FFHQ dataset.

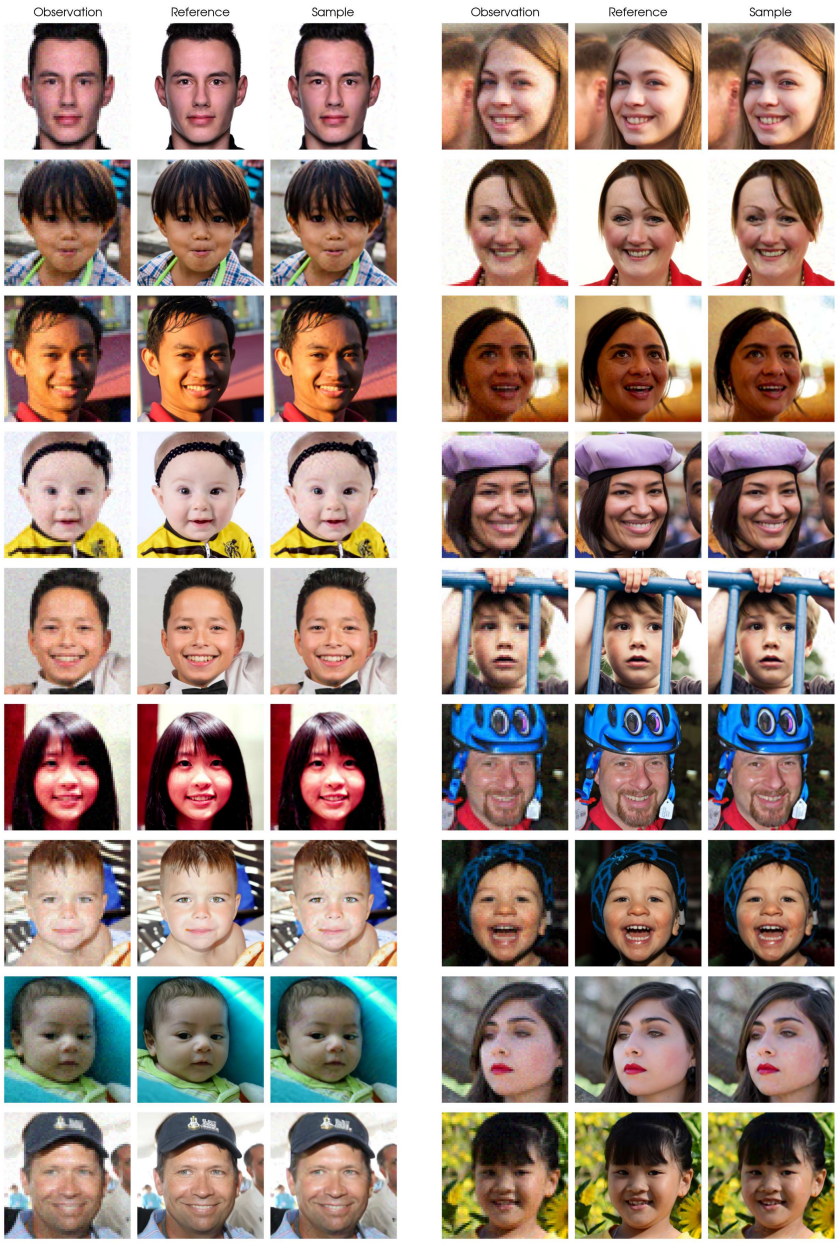

Figure 8: Super-resolution ($\times 4$) on FFHQ dataset.

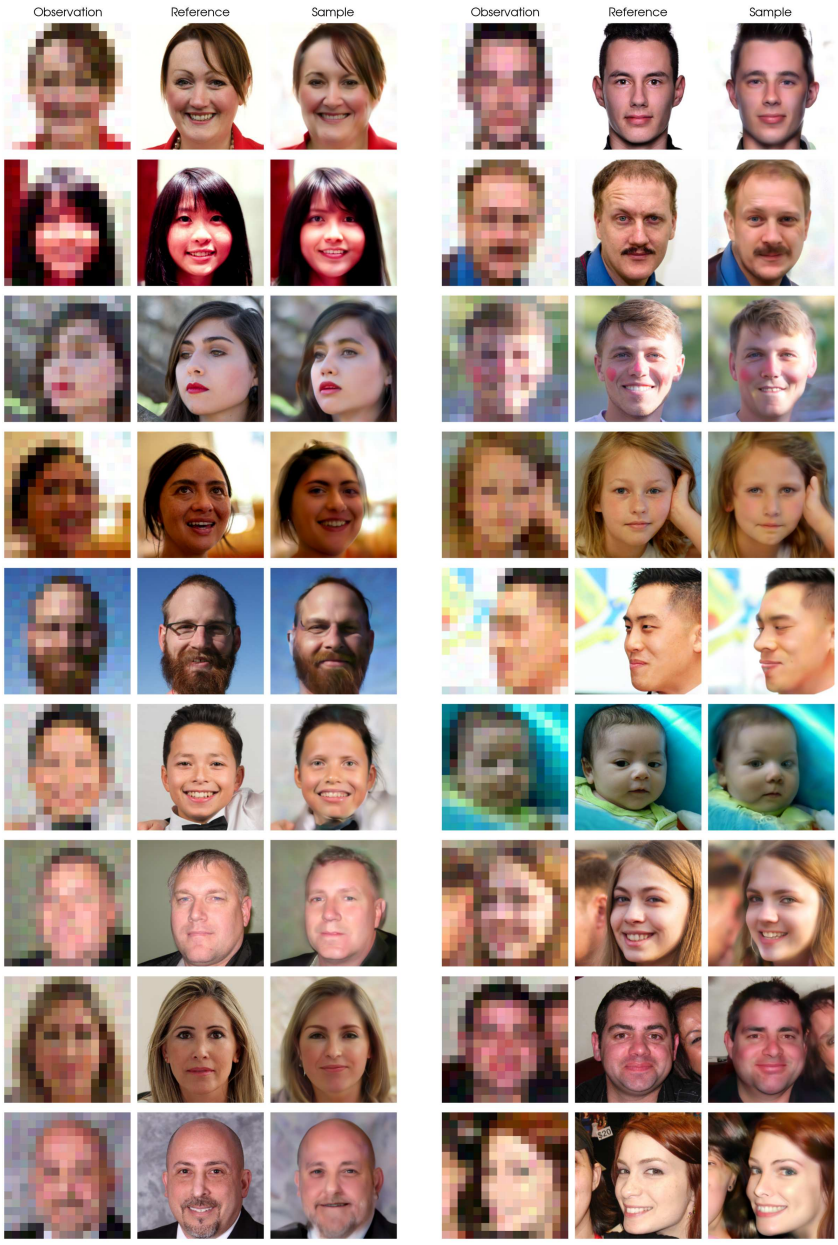

Figure 9: Super-resolution ($\times 16$) on FFHQ dataset.

Figure 10: Box Inpainting on FFHQ dataset.

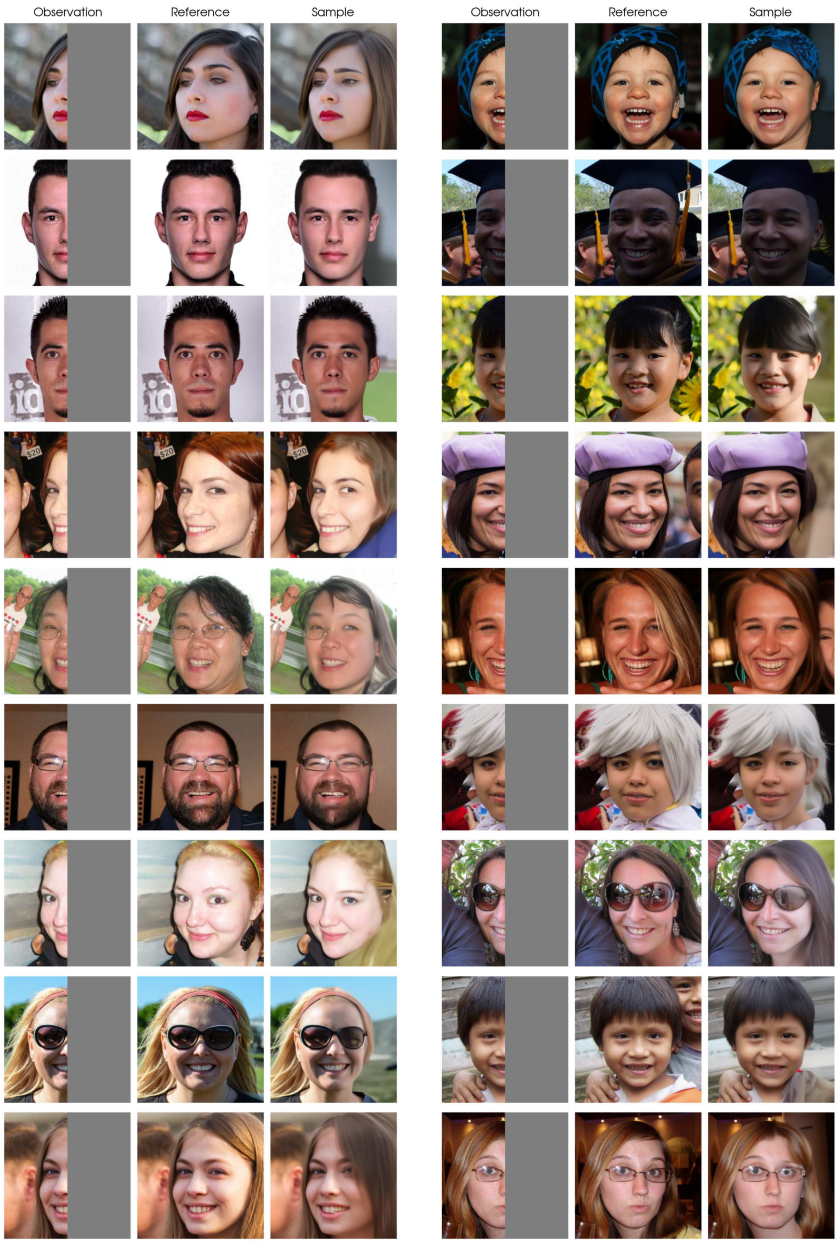

Figure 11: Half Inpainting on FFHQ dataset.

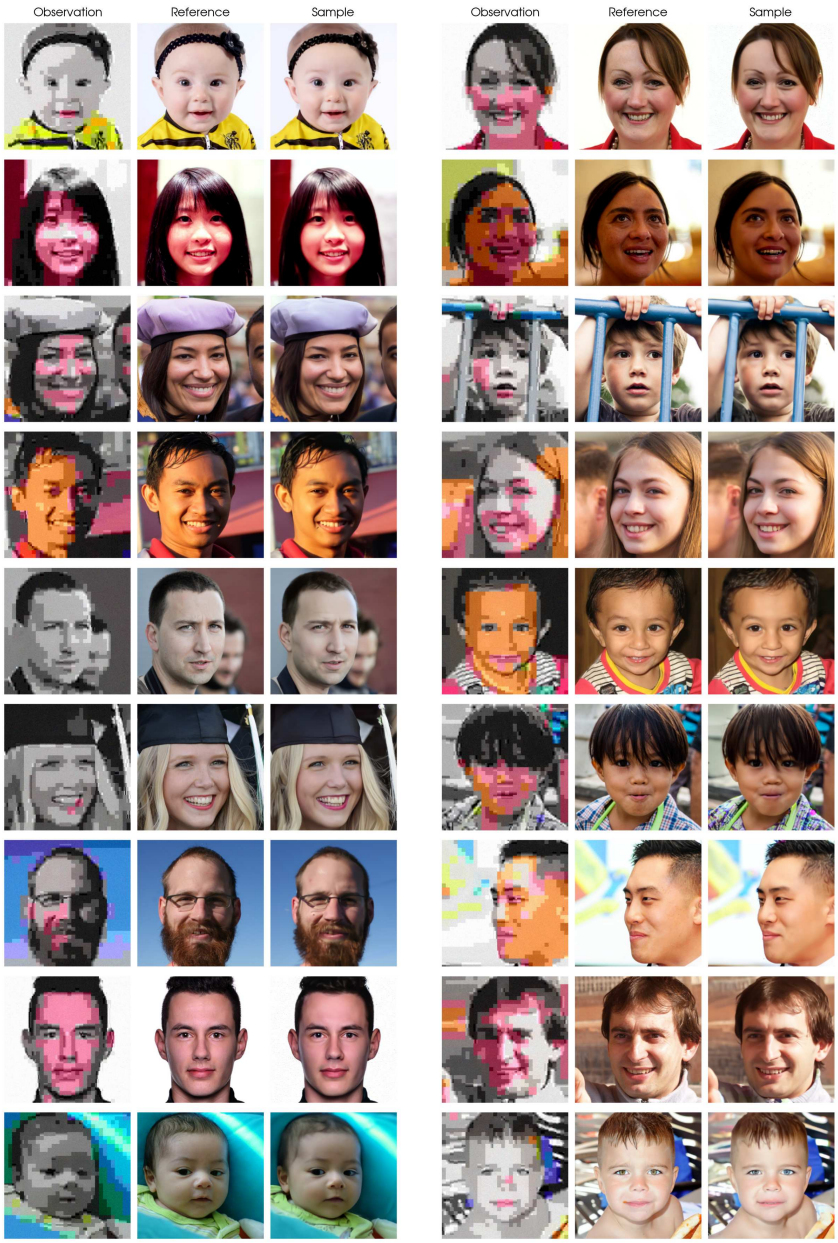

Figure 12: JPEG Compression (QF=2) on FFHQ dataset.

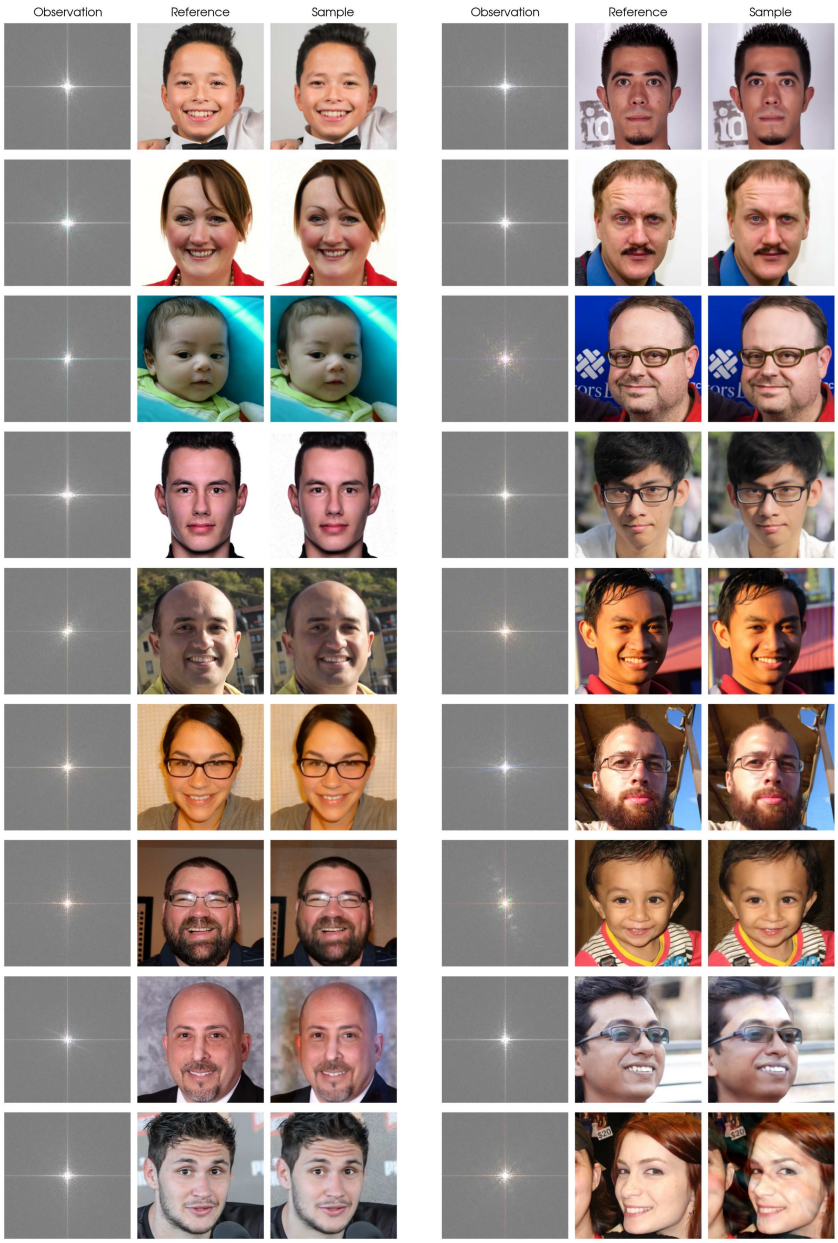

Figure 13: Phase Retrieval on FFHQ dataset.

I RECONSTRUCTION SAMPLES ON FFHQ WITH LDM

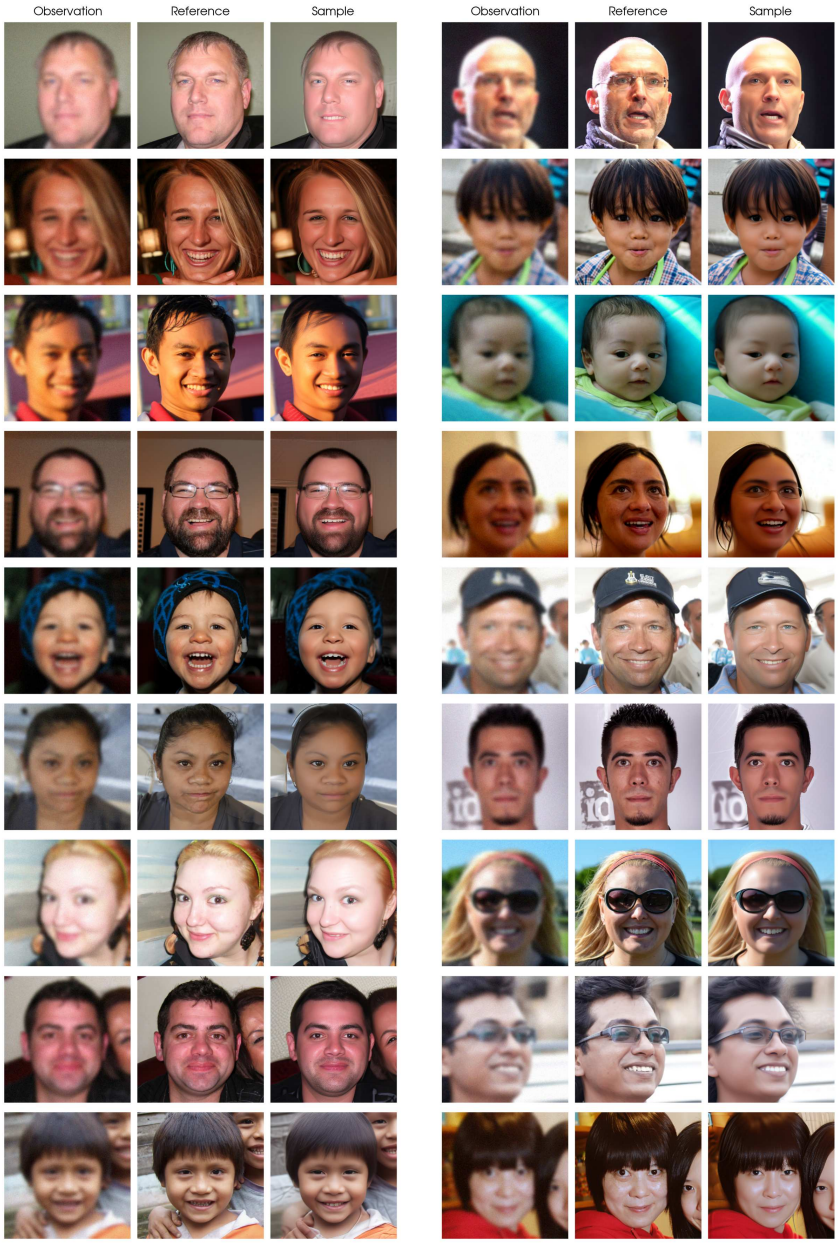

Figure 14: Gaussian Deblurring on FFHQ dataset with LDM prior.

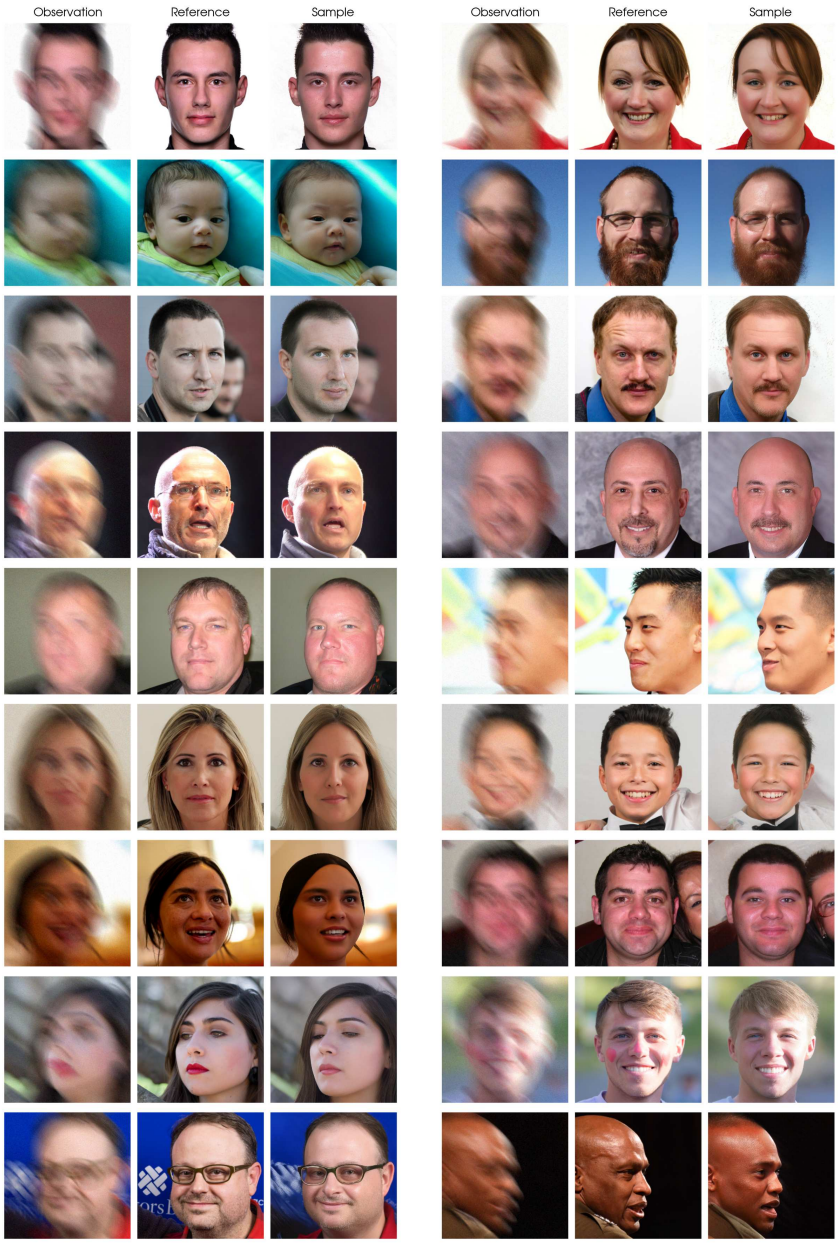

Figure 15: Motion Deblurring on FFHQ dataset with LDM prior.

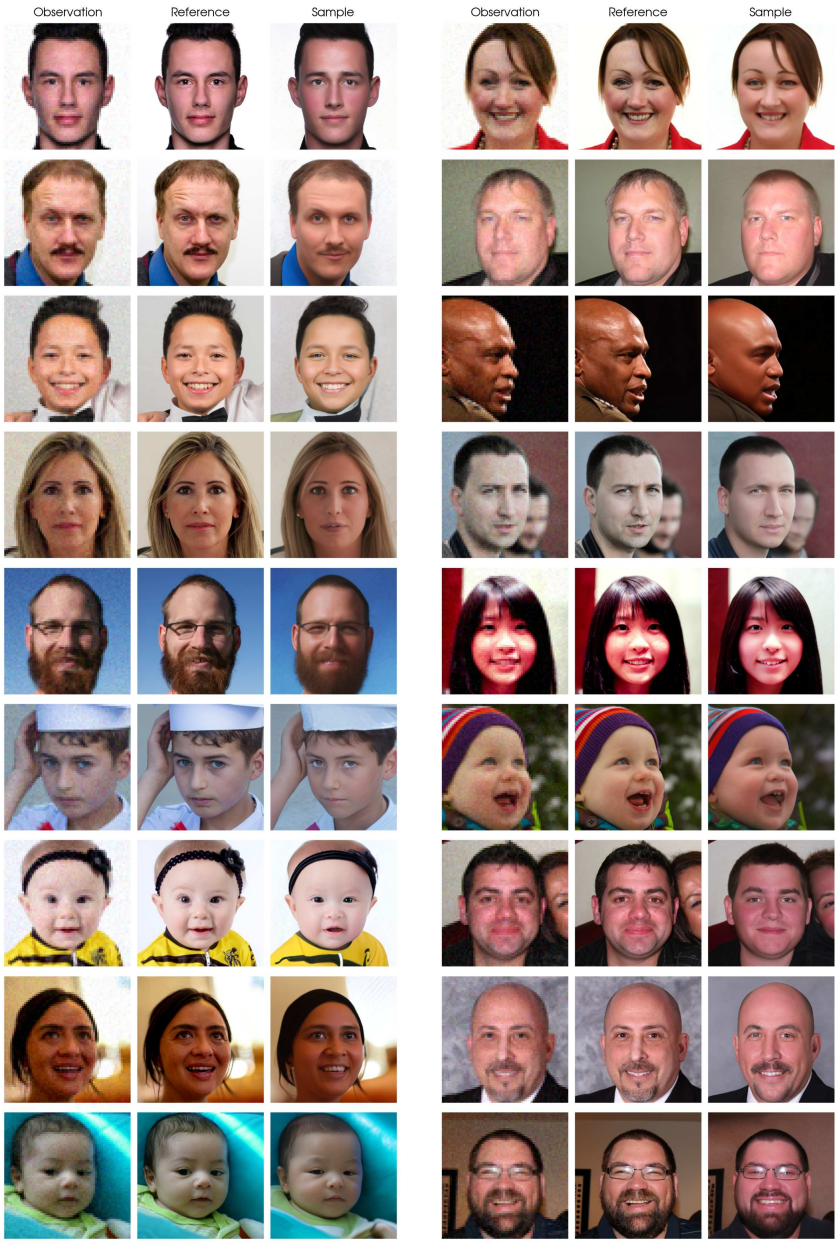

Figure 16: Super-Resolution (×4) on FFHQ dataset with LDM prior.

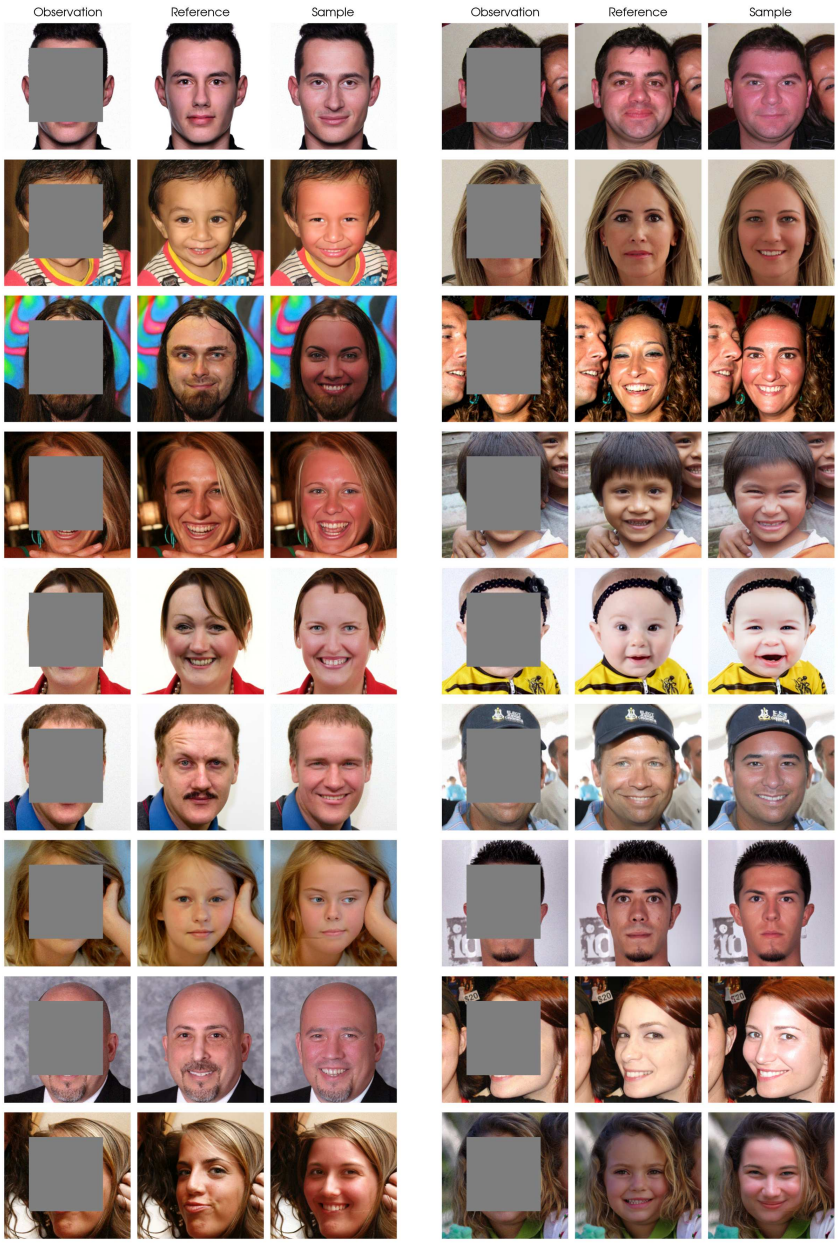

Figure 17: Box Inpainting on FFHQ dataset with LDM prior.

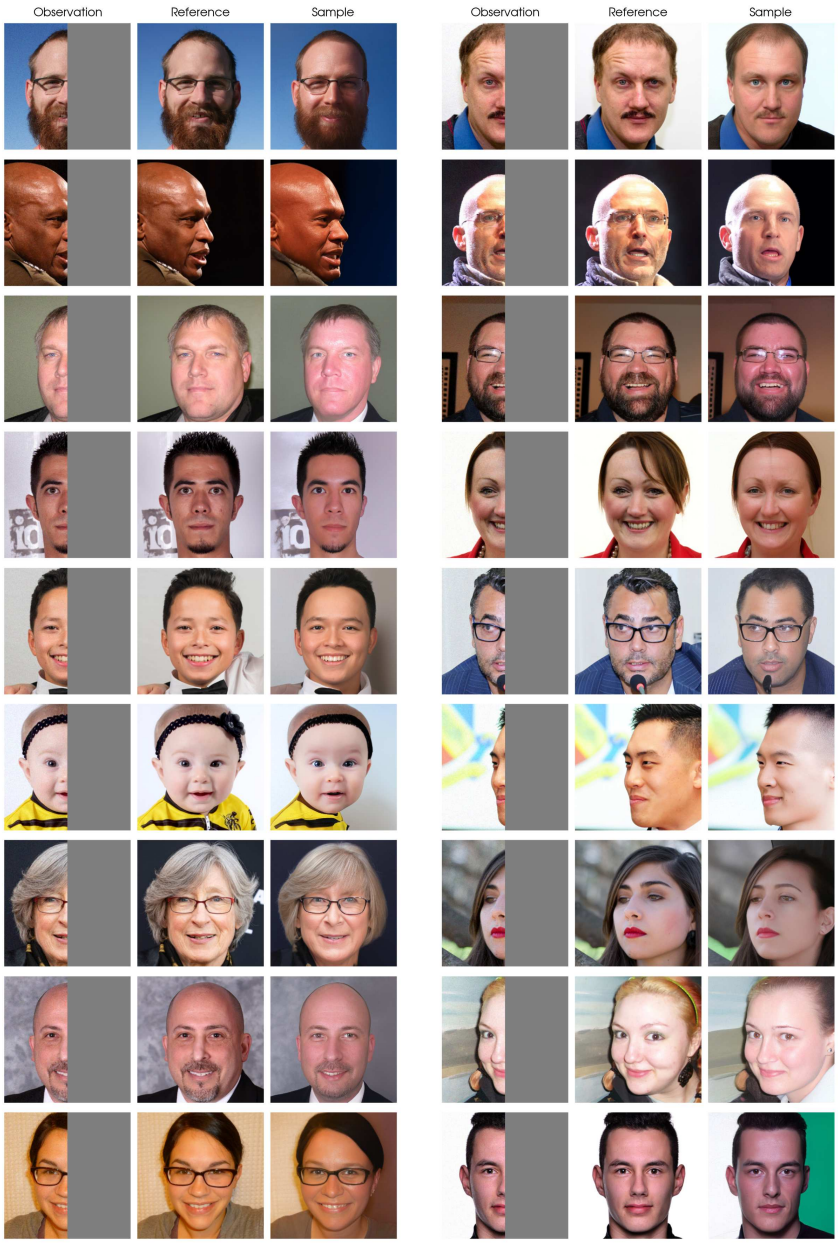

Figure 18: Half Inpainting on FFHQ dataset with LDM prior.

J RECONSTRUCTION SAMPLES ON IMAGENET

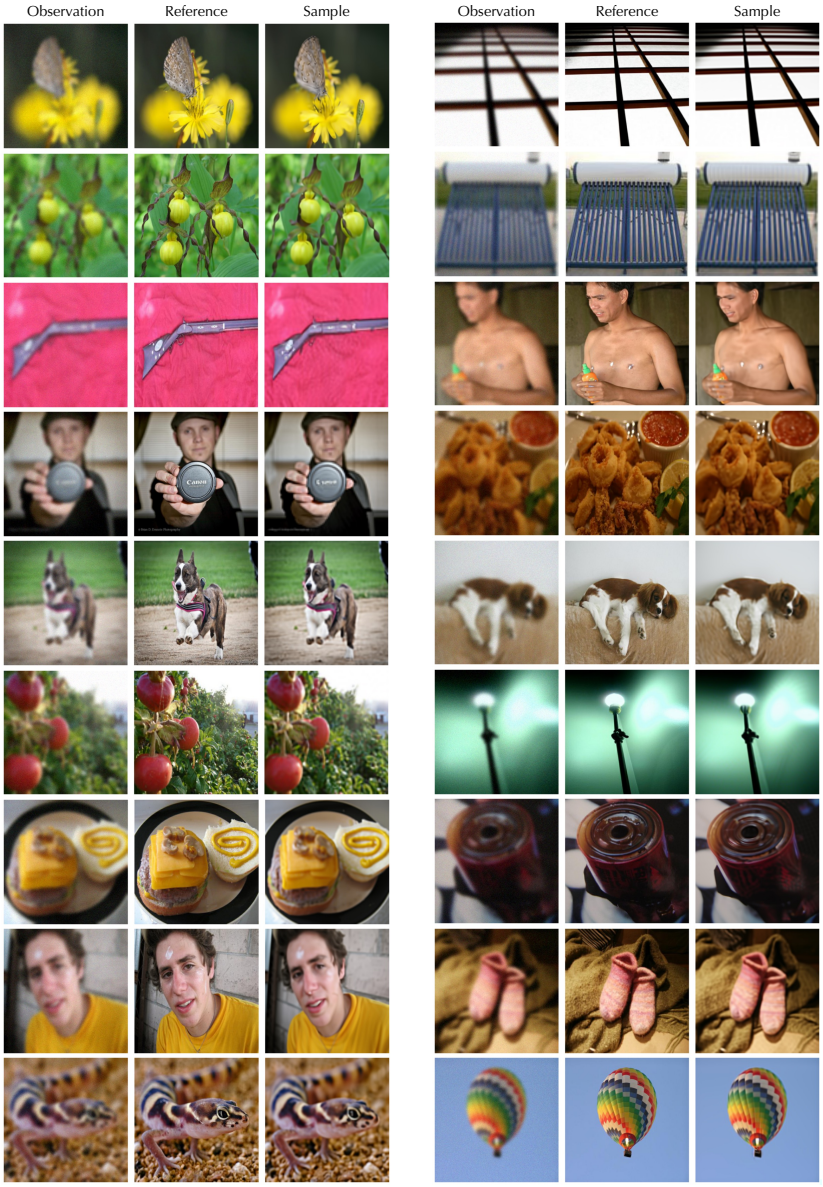

Figure 19: Deblurring on ImageNet dataset.

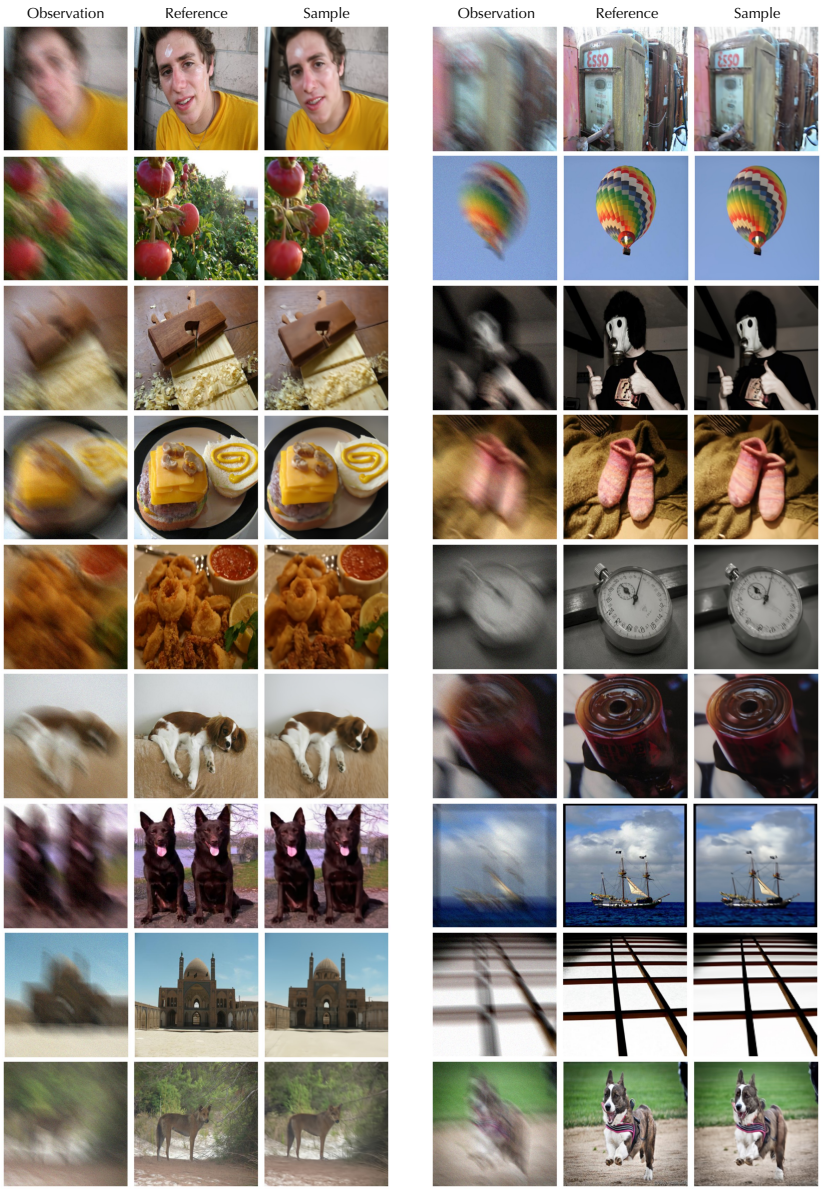

Figure 20: Deblurring on `ImageNet` dataset.

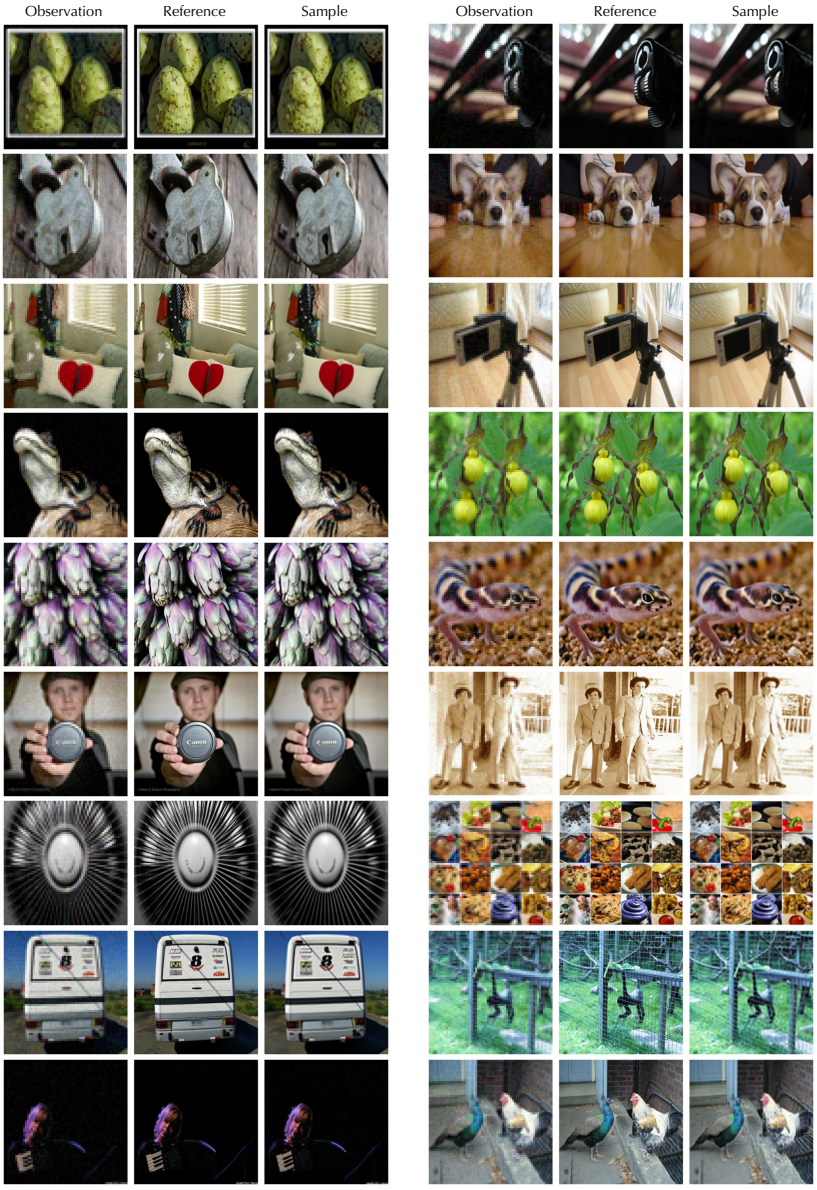

Figure 21: Super-resolution ($\times 4$) on `ImageNet` dataset.

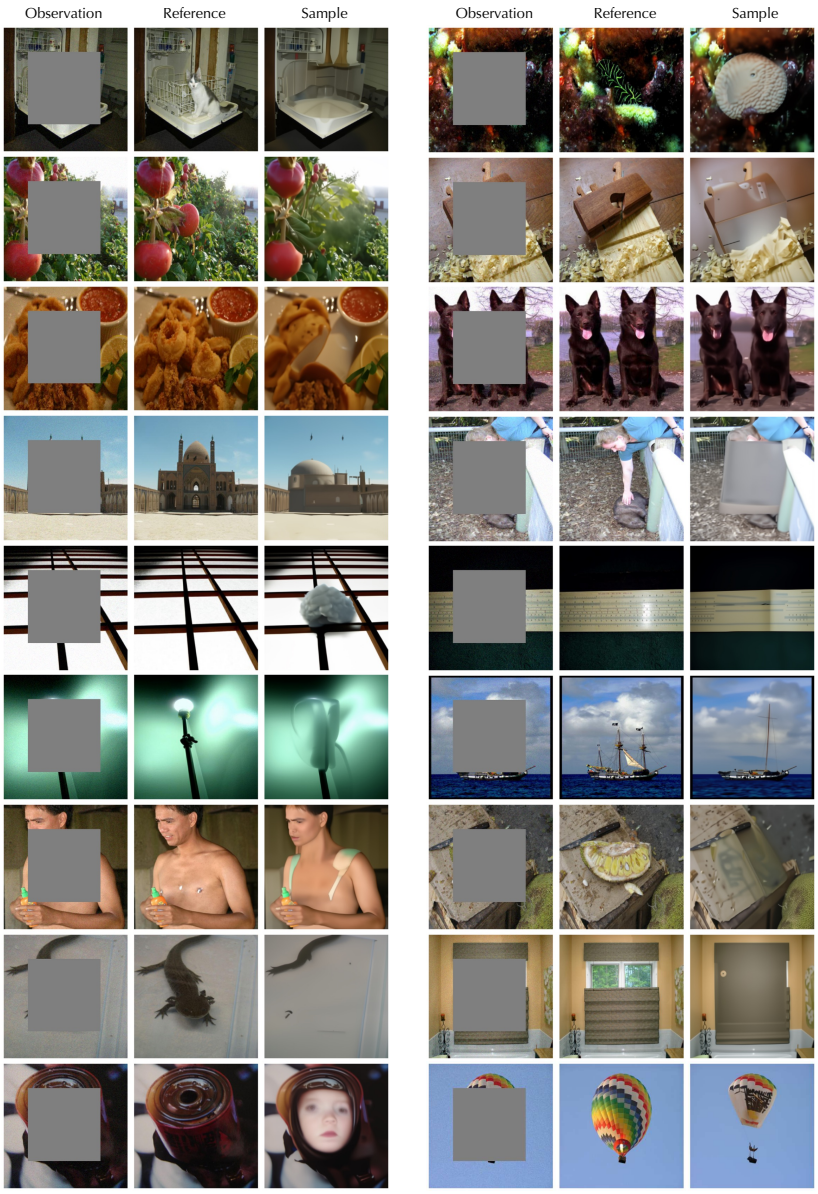

Figure 22: Center inpainting on `ImageNet` dataset.

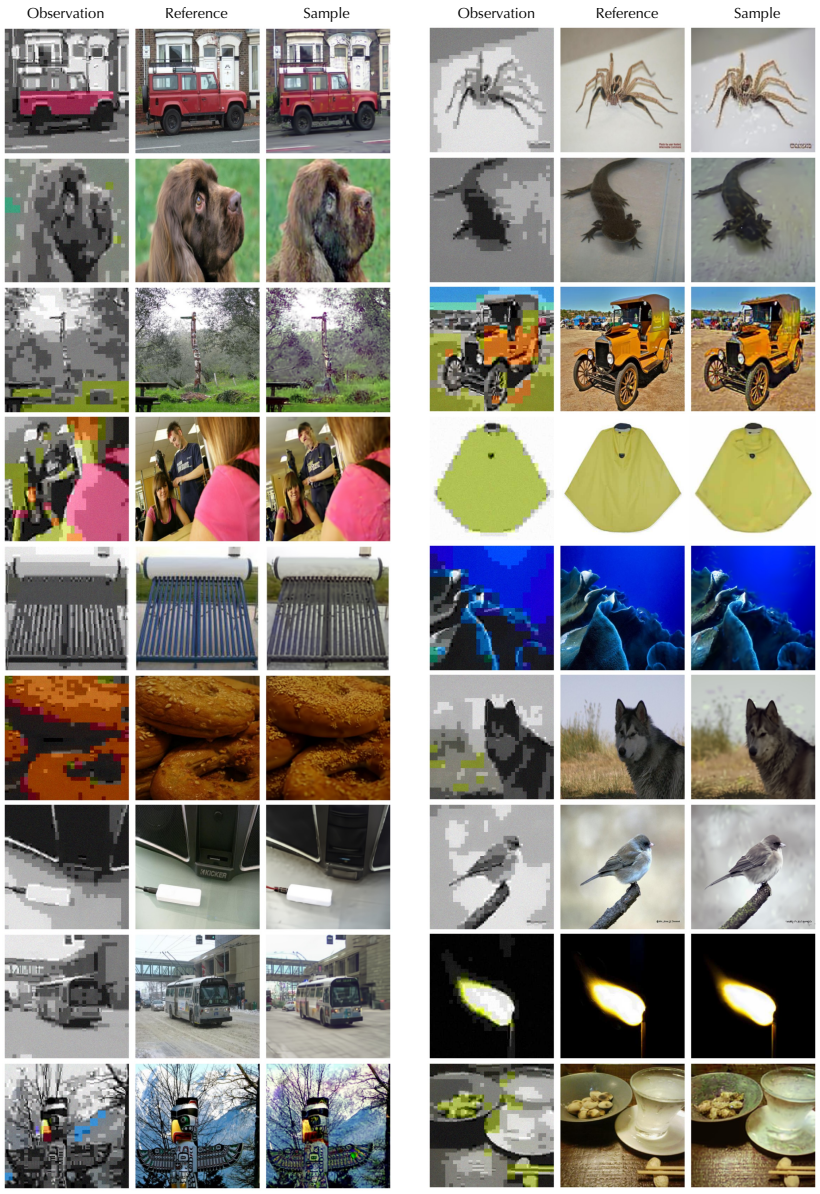

Figure 23: JPEG compression (QF=2) on `ImageNet` dataset.

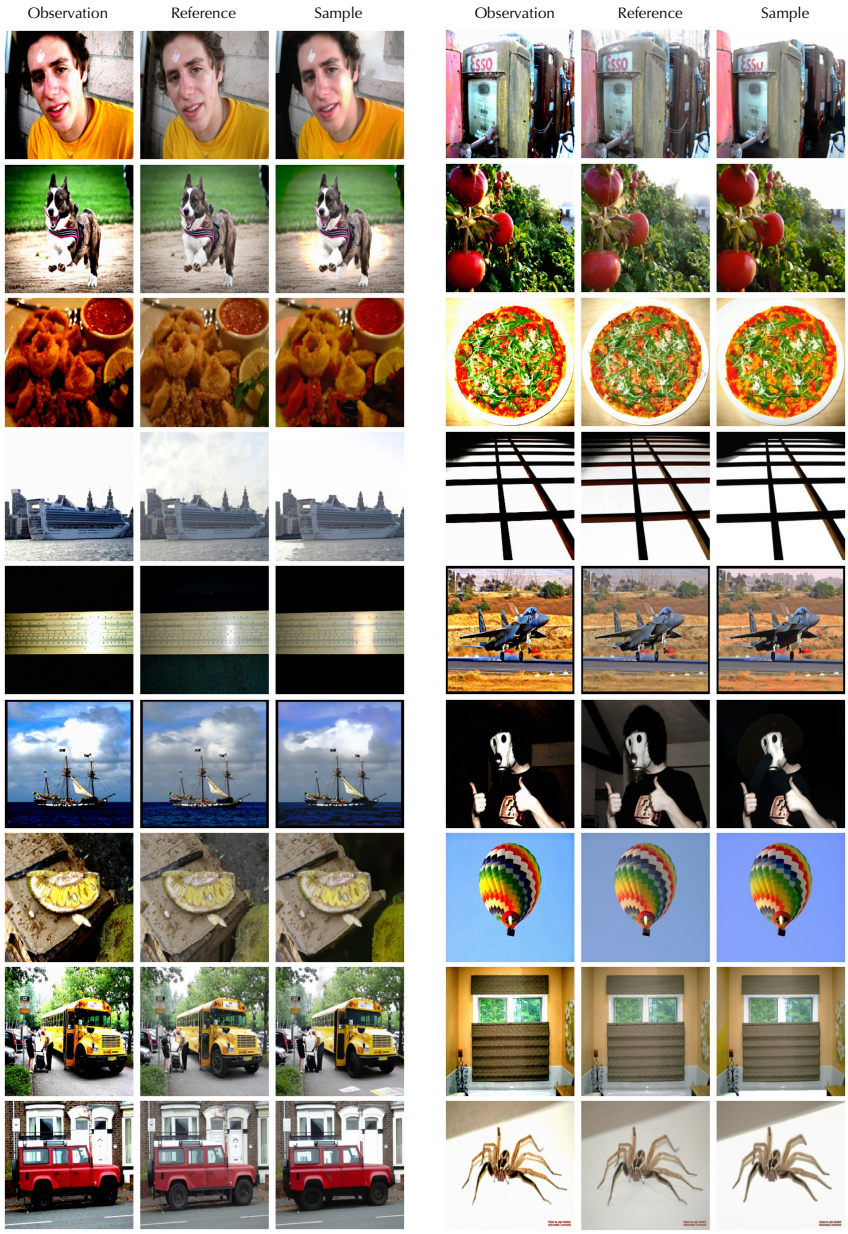

Figure 24: High Dynamic Range on `ImageNet` dataset.