A Mixture-Based Framework for Guiding Diffusion Models

Yazid Janati [*1] **Badr Moufad** [*1] **Mehdi Abou El Qassime** [1]
Alain Durmus [1] **Eric Moulines** [1] **Jimmy Olsson** [2]

Abstract

Denoising diffusion models have driven significant progress in the field of Bayesian inverse problems. Recent approaches use pre-trained diffusion models as priors to solve a wide range of such problems, only leveraging inference-time compute and thereby eliminating the need to retrain task-specific models on the same dataset. To approximate the posterior of a Bayesian inverse problem, a diffusion model samples from a sequence of intermediate posterior distributions, each with an intractable likelihood function. This work proposes a novel mixture approximation of these intermediate distributions. Since direct gradient-based sampling of these mixtures is infeasible due to intractable terms, we propose a practical method based on Gibbs sampling. We validate our approach through extensive experiments on image inverse problems, utilizing both pixel- and latent-space diffusion priors, as well as on source separation with an audio diffusion model.

1 Introduction

Inverse problems occur when a signal X of interest must be inferred from an incomplete and noisy observation Y, a challenge frequently encountered in diverse fields such as weather forecasting, image reconstruction (*e.g.*, tomography or black-hole imaging), and speech processing. Such problems are typically ill-posed, making it essential to incorporate additional constraints, regularization techniques, or prior knowledge to arrive at meaningful and realistic solutions.

The Bayesian framework, in conjunction with generative modeling, offers a systematic approach to the challenges associated with inverse problems. Prior knowledge about the signal of interest, often represented through samples from its underlying distribution p_0, can be leveraged to train a generative model p_0^θ that acts as a prior. By combining it with the conditional density $g_0(\mathbf{y}|\mathbf{x})$ of the observation given the signal, deduced from the form of the inverse problem at hand, we can compute the posterior distribution. Samples drawn from this posterior encapsulate plausible solutions that harmonize prior knowledge with the observed data. One straightforward approach to approximate sampling from the posterior distribution involves constructing a paired dataset of i.i.d. signals and observations, $(X_i, Y_i)_{i=1}^N$, where $X_i \sim p_0$ and $Y_i \sim g_0(\cdot|X_i)$, and learning a direct mapping (Dong et al., 2015) or generative model (Ledig et al., 2017; Isola et al., 2017). The latter, when queried with multiple independent noise samples alongside an observation, generates a diverse set of potential reconstructions. However, this approach is inherently *task-specific*, delivering reliable reconstructions only when the conditional distribution of the observation remains unchanged at test time. As a result, it cannot straightforwardly adapt to unseen tasks with the same prior. Adaptation to a new task can only be achieved by retraining a new generative model.

An increasingly popular approach consists in learning a generative model only for the prior p_0, and then leveraging inference-time compute to solve any inverse problem for which the likelihood function $\mathbf{x} \mapsto g_0(\mathbf{y}|\mathbf{x})$ is provided in a closed form. This strategy eliminates the need for expensive and inefficient task-specific training. Initially explored with generative models such as variational autoencoders and generative adversarial networks (Xia et al., 2022), this framework has recently been extended to denoising diffusion models (DDMs) (Song et al., 2021; Kadkhodaie & Simoncelli, 2020; Kawar et al., 2021; 2022; Chung et al., 2023; Song et al., 2023a; Daras et al.), which are the focus of the present paper.

DDMs (Sohl-Dickstein et al., 2015; Song & Ermon, 2019; Ho et al., 2020) achieve state-of-the-art generative performance across a wide range of domains. At their core is a forward noising process that transforms the data distribution p_0 into a Gaussian distribution. A generative model is then learned by reversing this noising process. With a specific parameterization of the backward process,

[*]Equal contribution [1]Ecole polytechnique [2]KTH Royal Institute of Technology. Correspondence to: Yazid Janati, Badr Moufad <first.last@polytechnique.edu>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

which converts noise into data samples, training the generative model reduces to approximating denoisers for each noise level introduced during the forward process. Recent methods for training-free posterior sampling aim to approximate the denoisers for the posterior distribution, enabling the use of diffusion models for sampling (Ho et al., 2022; Chung et al., 2023; Song et al., 2023a). A posterior distribution denoiser can be decomposed into two terms: the prior denoiser at the same noise level (provided by a pre-trained diffusion model) and the gradient of the log-likelihood of the observation conditioned on the current noisy sample. The latter term, which is intractable, is what guides the samples during the denoising process towards the posterior distribution. Various approximations for this gradient term have been proposed. However, they are often crude and require significant adjustments and heuristics to ensure stability and satisfactory performance. When applied to latent diffusion models, they often demand additional, model-specific adjustments (Rout et al., 2024).

Our contribution. In this paper, we present a principled method that circumvents these issues by introducing a new approximation of the likelihood term, paired with a sampling scheme based on Gibbs sampling (Geman & Geman, 1984). Our key observation is that multiple approximations can be derived for each likelihood term at a fixed noise level using a simple identity that it satisfies. However, the scores of these new likelihood approximations are not available in closed form, preventing us from deriving a direct posterior denoiser approximation by combining, through a mixture, the different likelihood approximations. We overcome this limitation by constructing a mixture approximation of the intermediate posterior distributions defined by the diffusion model for the original posterior. Our algorithm, MIXTURE-GUIDED DIFFUSION MODEL (MGDM), proceeds by sequentially sampling from these mixtures using Gibbs sampling. This is enabled by a carefully designed data augmentation scheme that ensures straightforward Gibbs updates. A key advantage of our approach is its adaptability to available computational resources. Specifically, the number of Gibbs iterations acts as a tunable parameter, allowing substantial improvements with increased inference-time compute. MGDM demonstrates strong empirical performance across 10 image-restoration tasks involving both pixel-space and latent-space diffusion models, as well as in musical source separation, even matching the performance of supervised methods.

2 Background

2.1 Diffusion models

DDMs define a generative process for a data distribution p_0 on $\mathbb{R}^d$ by sequentially sampling from a series of progressively less smoothed distributions $(p_t)_{t=T}^0$, starting from a highly smoothed prior p_T and ending at the data distribution p_0. For all $s, t \in [\![0, T]\!]$ with $s < t$, define the noising Markov transition kernels

$$q_{t|s}(\mathbf{x}_t|\mathbf{x}_s) = \mathrm{N}(\mathbf{x}_t; (\alpha_t/\alpha_s)\mathbf{x}_s, \sigma_{t|s}^2 \mathbf{I}_d)\,, \tag{1}$$

where $(\alpha_t)_{t=0}^T$ is monotonically decreasing with $\alpha_0 = 1$, $\alpha_T \approx 0$, and $\sigma_{t|s}^2 = 1 - (\alpha_t/\alpha_s)^2$. Each smoothed distribution is a noised version of p_0 and has density $p_t(\mathbf{x}_t) := \int q_{t|0}(\mathbf{x}_t|\mathbf{x}_0) p_0(\mathbf{x}_0)\, \mathrm{d}\mathbf{x}_0$. The final distribution p_T is close to $\mathcal{N}(0_d, \mathbf{I}_d)$. Moreover, define the backward Markov transition kernels $p_{s|t}(\mathbf{x}_s|\mathbf{x}_t) \propto p_s(\mathbf{x}_s) q_{t|s}(\mathbf{x}_t|\mathbf{x}_s)$ with $s < t$. Note that for all $\ell < s$, the backward transitions satisfy

$$p_{\ell|t}(\mathbf{x}_\ell|\mathbf{x}_t) = \int p_{\ell|s}(\mathbf{x}_\ell|\mathbf{x}_s) p_{s|t}(\mathbf{x}_s|\mathbf{x}_t)\, \mathrm{d}\mathbf{x}_s\,. \tag{2}$$

Consecutive distributions p_t and p_{t+1} are linked through the identity $p_{t+1}(\mathbf{x}_{t+1}) = \int q_{t+1|t}(\mathbf{x}_{t+1}|\mathbf{x}_t) p_t(\mathbf{x}_t)\, \mathrm{d}\mathbf{x}_t$. Hence, given a sample $X_{t+1} \sim p_{t+1}$, $X_t \sim p_{t|t+1}(\cdot|X_{t+1})$ is an exact sample from p_t. This procedure defines a generative model, in the sense that the last state X_0 of the Markov chain $(X_t)_{t=T}^0$, where the initial state X_T is sampled from p_T, is a sample from p_0.

However, simulating the backward transitions is impracticable in most applications, so the following Gaussian approximation is used in practice. First, for $s \in [\![1, t-1]\!]$, define the conditional density of X_s given X_0 and X_t:

$$\begin{aligned} &q_{s|0,t}(\mathbf{x}_s|\mathbf{x}_0, \mathbf{x}_t) \\ &\quad = \mathrm{N}(\mathbf{x}_s; \gamma_{t|s}\alpha_{s|0}\mathbf{x}_0 + (1-\gamma_{t|s})\alpha_{t|s}^{-1}\mathbf{x}_t, \sigma_{s|0,t}^2 \mathbf{I}_d)\,, \end{aligned} \tag{3}$$

where $\gamma_{t|s} := \sigma_{t|s}^2/\sigma_{t|0}^2$ and $\sigma_{s|0,t}^2 := \sigma_{t|s}^2 \sigma_{s|0}^2/\sigma_{t|0}^2$. Next, define by $D_{t+1}(\mathbf{x}_{t+1}) := \int \mathbf{x}_0\, p_{0|t+1}(\mathbf{x}_0|\mathbf{x}_{t+1})\, \mathrm{d}\mathbf{x}_0$ the conditional expectation of X_0 given $X_{t+1} = \mathbf{x}_{t+1}$ (referred to as the denoiser). Denote by D_{t+1}^θ a parametric approximation of D_{t+1}. Following Ho et al. (2020) and given an approximate sample $\hat{X}_{t+1}$ from p_{t+1}, sampling from the bridge kernel $q_{t|0,t+1}(\cdot|D_{t+1}^\theta(\hat{X}_{t+1}), \hat{X}_{t+1})$, where $\mathbf{x}_0$ is replaced by the estimate $D_{t+1}^\theta(\hat{X}_{t+1})$, yields an approximate sample from p_t. The complete sampling process proceeds as follows: first, $\hat{X}_T \sim \mathcal{N}(0_d, \mathbf{I}_d)$; then, recursively, for every $t \geq 1$, $\hat{X}_t \sim p_{t|t+1}^\theta(\cdot|\hat{X}_{t+1})$, where for all $s < t$,

$$p_{s|t}^\theta(\mathbf{x}_s|\mathbf{x}_t) := q_{s|0,t}(\mathbf{x}_s|D_t^\theta(\mathbf{x}_t), \mathbf{x}_t)\,. \tag{4}$$

The final sample is defined as $\hat{X}_0 := D_1^\theta(\hat{X}_1)$ and serves as an approximate sample from p_0. The parametric approximations of the denoisers are trained by minimizing, with respect to the parameter θ, an L_2 denoising loss across all time steps. Finally, using the Tweedie formula (Robbins, 1956), we obtain the identity $D_t(\mathbf{x}_t) = \alpha_t^{-1}\big(\mathbf{x}_t + \sigma_t^2 \nabla \log p_t(\mathbf{x}_t)\big)$. Consequently, the trained denoisers not only serve as generative models but also provide parametric approximations of the score functions $\nabla \log p_t(\mathbf{x}_t)$.

2.2 Training-free guidance.

After training a diffusion model for the data distribution p_0, it can be leveraged through *guidance* to address various downstream tasks without the need for additional fine-tuning. This line of research was pioneered in the seminal works of Song & Ermon (2019), Kadkhodaie & Simoncelli (2020), Song et al. (2021), and Kawar et al. (2021), where the sampling process described in the previous section is adapted on-the-fly to address Bayesian inverse problems. In this setting, the user observes a realization $\mathbf{y}$ of a random variable $Y \in \mathbb{R}^{d_\mathbf{y}}$, assumed to be drawn from the distribution with density $p_Y(\mathbf{y}) := \int g_0(\mathbf{y}|\mathbf{x})p_0(\mathbf{x})\,\mathrm{d}\mathbf{x}$, where $g_0(\mathbf{y}|\mathbf{x})$ is a likelihood term that encapsulates the knowledge of the *forward model*. A typical example is inverse problems with Gaussian noise, *i.e.* $g_0(\mathbf{y}|\mathbf{x}) = \mathrm{N}(\mathbf{y}; \mathbf{A}(\mathbf{x}), \mathbf{\Sigma}_\mathbf{y})$, where $\mathbf{A} : \mathbb{R}^d \to \mathbb{R}^{d_\mathbf{y}}$ and $\mathbf{\Sigma}_\mathbf{y}$ is a covariance matrix. The objective is to recover plausible underlying signals $\mathbf{x}$, for which prior information is encoded in p_0. This recovery is achieved by sampling from the posterior distribution

$$\pi_0^\mathbf{y}(\mathbf{x}_0) \propto g_0(\mathbf{y}|\mathbf{x}_0)p_0(\mathbf{x}_0)\,.$$

A common approach to constructing a sampler for this posterior distribution is to adopt the diffusion model framework by sequentially sampling from the smoothed distributions $\pi_T^\mathbf{y}, \dots, \pi_1^\mathbf{y}$, which are defined analogously to those introduced in the previous section:

$$\pi_t^\mathbf{y}(\mathbf{x}_t) := \int q_{t|0}(\mathbf{x}_t|\mathbf{x}_0)\pi_0^\mathbf{y}(\mathbf{x}_0)\,\mathrm{d}\mathbf{x}_0\,. \tag{5}$$

Following the derivations above, sampling these distributions backwards in time is feasible provided that the conditional denoisers $(D_t^\mathbf{y})_{t=1}^T$ are accessible. Each conditional denoiser is defined by

$$D_t^\mathbf{y}(\mathbf{x}_t) := \int \mathbf{x}_0\, \pi_{0|t}^\mathbf{y}(\mathbf{x}_0|\mathbf{x}_t)\,\mathrm{d}\mathbf{x}_0,$$

where the conditional posterior $\pi_{0|t}^\mathbf{y}(\mathbf{x}_0|\mathbf{x}_t)$ is given by $\pi_{0|t}^\mathbf{y}(\mathbf{x}_0|\mathbf{x}_t) \propto \pi_0^\mathbf{y}(\mathbf{x}_0)q_{t|0}(\mathbf{x}_t|\mathbf{x}_0)$. By analogy with the smoothed distributions defined for the prior, we obtain that

$$\begin{aligned}\pi_t^\mathbf{y}(\mathbf{x}_t) &\propto \int g_0(\mathbf{y}|\mathbf{x}_0)q_{t|0}(\mathbf{x}_t|\mathbf{x}_0)p_0(\mathbf{x}_0)\,\mathrm{d}\mathbf{x}_0 \\ &\propto g_t(\mathbf{y}|\mathbf{x}_t)p_t(\mathbf{x}_t),\end{aligned} \tag{6}$$

where

$$g_t(\mathbf{y}|\mathbf{x}_t) := \int g_0(\mathbf{y}|\mathbf{x}_0)p_{0|t}(\mathbf{x}_0|\mathbf{x}_t)\,\mathrm{d}\mathbf{x}_0, \tag{7}$$

and we used that $p_0(\mathbf{x}_0)q_{t|0}(\mathbf{x}_t|\mathbf{x}_0) = p_{0|t}(\mathbf{x}_0|\mathbf{x}_t)p_t(\mathbf{x}_t)$. Next, using the Tweedie formula, the posterior and prior denoisers can be related as

$$D_t^\mathbf{y}(\mathbf{x}_t) = D_t(\mathbf{x}_t) + \alpha_t^{-1}\sigma_t^2\nabla \log g_t(\mathbf{y}|\mathbf{x}_t)\,. \tag{8}$$

This shows that in order to estimate $D_t^\mathbf{y}$ we only need to estimate $\nabla \log g_t(\mathbf{y}|\cdot)$, as we already have access to a pre-trained parametric approximation of D_t. A widely used approximation of this likelihood term (Ho et al., 2022; Chung et al., 2023), which we will also use in the next section, is

$$\hat{g}_t^\theta(\mathbf{y}|\mathbf{x}_t) := g(\mathbf{y}|D_t^\theta(\mathbf{x}_t)), \tag{9}$$

and amounts to approximating the posterior distribution $p_{0|t}(\cdot|\mathbf{x}_t)$ with a Dirac mass at $D_t^\theta(\mathbf{x}_t)$, which we express as $p_{0|t}(\cdot|\mathbf{x}_t) \approx \delta_{D_t^\theta(\mathbf{x}_t)}$ with a slight abuse of notation. To improve the quality of the sample, $\nabla \log \hat{g}_t^\theta(\mathbf{y}|\mathbf{x}_t)$ is rescaled with a suitable weight (possibly depending on $\mathbf{x}_t$); see (Ho et al., 2022, Equation 8) and (Chung et al., 2023, Algorithm 1). We emphasize that the rescalings generally used are only heuristic. Compared to previous works, methods that perform guidance using the approximation (9) incur additional computational overhead due to the calculation of a vector-Jacobian product when evaluating $\nabla \log \hat{g}_t^\theta(\mathbf{y}|\mathbf{x}_t)$. Nevertheless, subsequent works using this approximation have shown remarkable improvements in performance across various applications; see for example (Song et al., 2023a; Rozet & Louppe, 2023; Yu et al., 2023; Wu et al., 2023; Jiang et al., 2023; Rozet et al., 2024; Moufad et al., 2024).

3 Guidance with mixtures

We now present our main contribution: a novel density approximation of the smoothed posteriors $\pi_t^\mathbf{y}$. Since their scores are intractable, gradient-based samplers cannot be directly applied. Thus, we develop a Gibbs sampling scheme targeting a data augmentation of our smoothed posterior approximation, marking our second key contribution.

In the next two sections we develop an algorithm for the ideal generative model, *i.e.*, we assume that we have at hand the true marginals p_t and backward transitions $p_{s|t}$; then, in Section 3.2, we provide a practical implementation involving the learned model.

3.1 Guidance approximation

We begin by extending the likelihood approximation in (9) introduced by Ho et al. (2022); Chung et al. (2023). First, note that by combining (2) with (7), we find that $g_t(\mathbf{y}|\cdot)$ satisfies

$$g_t(\mathbf{y}|\mathbf{x}_t) = \int g_s(\mathbf{y}|\mathbf{x}_s)p_{s|t}(\mathbf{x}_s|\mathbf{x}_t)\,\mathrm{d}\mathbf{x}_s,$$

for all $t \in [\![1, T]\!]$ and $s \in [\![0, t-1]\!]$. Thus, we obtain $t-1$ different approximations of $g_t(\mathbf{y}|\cdot)$ by simply setting, for $s \in [\![1, t-1]\!]$,

$$\hat{g}_t^s(\mathbf{y}|\mathbf{x}_t) := \int \hat{g}_s(\mathbf{y}|\mathbf{x}_s)p_{s|t}(\mathbf{x}_s|\mathbf{x}_t)\,\mathrm{d}\mathbf{x}_s\,, \quad t \geq 2\,, \tag{10}$$

where $\hat{g}_s(\mathbf{y}|\cdot)$ denotes the counterpart of (9), with the learned denoiser D_s^θ replaced by the true denoiser D_s. In contrast to $\hat{g}_t^\theta(\mathbf{y}|\cdot)$ in (9), the scores of these approximations remain intractable even when the approximate model is used, as they involve an intractable integral. Instead, we take a different approach and use $\hat{g}_t^s(\mathbf{y}|\cdot)$ to define density approximations

$$\hat{\pi}_t^s(\mathbf{x}_t) := \frac{\hat{g}_t^s(\mathbf{y}|\mathbf{x}_t)p_t(\mathbf{x}_t)}{\int \hat{g}_t^s(\mathbf{y}|\mathbf{x}_t')p_t(\mathbf{x}_t')\,\mathrm{d}\mathbf{x}_t'} \tag{11}$$

of the smoothed posteriors $\pi_t^{\mathbf{y}}$. Since we have $t-1$ such approximations, we consider a weighted mixture approximation of $\pi_t^{\mathbf{y}}$ defined, for $t \geq 1$, as

$$\hat{\pi}_t^{\mathbf{y}}(\mathbf{x}_t) := \sum_{s=1}^{t-1} \omega_t^s \hat{\pi}_t^s(\mathbf{x}_t)\,, \tag{12}$$

where $(\omega_t^s)_{s=1}^{t-1}$ are time-dependent weights and $\sum_{s=1}^{t-1}\omega_t^s = 1$ with $\omega_t^s \geq 0$. Then, to sample approximately from $\pi_0^{\mathbf{y}}$ we can use a sequential sampling procedure that runs through the intermediate distributions $\hat{\pi}_T^{\mathbf{y}}, \dots, \hat{\pi}_1^{\mathbf{y}}$. Similar sequential sampling procedures from posterior sequences different from $(\pi_t^{\mathbf{y}})_t$ have also been utilized in previous works. For instance, Wu et al. (2023); Rozet & Louppe (2023) use $\hat{\pi}_t^{\mathbf{y}}(\mathbf{x}_t) \propto \hat{g}_t(\mathbf{y}|\mathbf{x}_t)p_t(\mathbf{x}_t)$. Our approach differs from these prior works by employing a mixture-based formulation with non-standard approximations of $\pi_t^{\mathbf{y}}$. However, sampling from $\pi_t^{\mathbf{y}}()$ remains a non-trivial challenge. Indeed, a naive procedure would consist in sampling an index $s \sim \text{Categorical}(\{\omega_t^\ell\}_{\ell=1}^{t-1})$ and then use an approximate sampler only for $\hat{\pi}_t^s$. However, we must address the intractability of both $\hat{\pi}_t^s(\cdot)$ and its score. In the next section, we propose a method that fully overcomes these challenges. The discussion on selecting the weight sequence $(\omega_t^s)_{s=1}^{t-1}$ is postponed until after presenting the algorithm, more specifically at the beginning of Section 5.

3.2 Data augmentation and Gibbs sampling

We first detail how to sample from a single component $\hat{\pi}_t^s$ of the mixture (12) for given $t \in [\![2, T]\!]$ and $s \in [\![1, t-1]\!]$. Consider first the extended distribution

$$\begin{aligned}&\overline{\pi}_{0,s,t}^{\mathbf{y}}(\mathbf{x}_0, \mathbf{x}_s, \mathbf{x}_t)\\ &\qquad \propto p_{0|s}(\mathbf{x}_0|\mathbf{x}_s)\hat{g}_s(\mathbf{y}|\mathbf{x}_s)p_{s|t}(\mathbf{x}_s|\mathbf{x}_t)p_t(\mathbf{x}_t)\,.\end{aligned} \tag{13}$$

From the definitions in (12) and (10) it follows that $\hat{\pi}_t^s$ is the $\mathbf{x}_t$-marginal of (13), *i.e.*,

$$\hat{\pi}_t^s(\mathbf{x}_t) = \int \overline{\pi}_{0,s,t}^{\mathbf{y}}(\mathbf{x}_0, \mathbf{x}_s, \mathbf{x}_t)\,\mathrm{d}\mathbf{x}_0\,\mathrm{d}\mathbf{x}_s.$$

To sample approximately from $\hat{\pi}_t^s$, we employ a sampler targeting $\overline{\pi}_{0,s,t}^{\mathbf{y}}$ and retain only the $\mathbf{x}_t$-coordinate of its output. Specifically, we use a Gibbs sampler (GS) (Geman & Geman, 1984; Casella & George, 1992; Gelfand, 2000), which, in this context, constructs a Markov chain $(\bar{X}_0^r, \bar{X}_s^r, \bar{X}_t^r)_{r\in\mathbb{N}}$ having $\overline{\pi}_{0,s,t}^{\mathbf{y}}$ as its stationary distribution. Denote by $\overline{\pi}_{s|0,t}^{\mathbf{y}}$, $\overline{\pi}_{t|0,s}^{\mathbf{y}}$, and $\overline{\pi}_{0|s,t}^{\mathbf{y}}$ its three full conditionals given by

$$\begin{cases}\overline{\pi}_{s|0,t}^{\mathbf{y}}(\mathbf{x}_s|\mathbf{x}_0, \mathbf{x}_t) = \frac{\hat{g}_s(\mathbf{y}|\mathbf{x}_s)q_{s|0,t}(\mathbf{x}_s|\mathbf{x}_0,\mathbf{x}_t)}{\int \hat{g}_s(\mathbf{y}|\mathbf{x}_s')q_{s|0,t}(\mathbf{x}_s'|\mathbf{x}_0,\mathbf{x}_t)\,\mathrm{d}\mathbf{x}_s'}\,,\\ \overline{\pi}_{t|0,s}^{\mathbf{y}}(\mathbf{x}_t|\mathbf{x}_0, \mathbf{x}_s) = q_{t|s}(\mathbf{x}_t|\mathbf{x}_s),\\ \overline{\pi}_{0|s,t}^{\mathbf{y}}(\mathbf{x}_0|\mathbf{x}_s, \mathbf{x}_t) = p_{0|s}(\mathbf{x}_0|\mathbf{x}_s).\end{cases}$$

The proof of this fact is postponed to Appendix A.2. Then, one step of the associated (deterministic scan) GS is described in Algorithm 1.

Algorithm 1 Gibbs sampler targeting (13)

Input: $(\bar{X}_0^r, \bar{X}_s^r, \bar{X}_t^r)$
draw $\bar{X}_s^{r+1} \sim \overline{\pi}_{s|0,t}^{\mathbf{y}}(\cdot|\bar{X}_0^r, \bar{X}_t^r)$
draw $\bar{X}_t^{r+1} \sim q_{t|s}(\cdot|\bar{X}_s^{r+1})$ `//noising`
draw $\bar{X}_0^{r+1} \sim p_{0|s}(\cdot|\bar{X}_s^{r+1})$ `//denoising`

Since (13) admits $\hat{\pi}_t^s$ as marginal, the process $(\bar{X}_t^r)_{r\in\mathbb{N}}$ will, at stationarity of $(\bar{X}_0^r, \bar{X}_s^r, \bar{X}_t^r)_{r\in\mathbb{N}}$, have $\hat{\pi}_t^{\mathbf{y}}$ as a marginal distribution. We provide basic background on Gibbs sampling in Appendix A.1 and refer the reader to (Casella & George, 1992).

It is clear from Algorithm 1 that only the update of $\bar{X}_s^r$ depends on the observation $\mathbf{y}$, while the updates of the remaining components are sampled via (i) a noising step involving the forward transition (1), which can be performed exactly, and (ii) a denoising step involving the prior diffusion model, which can be approximated using the pre-trained model.

Finally, to target the mixture (12), we first sample the mixture index $s \sim \text{Categorical}\big(\{\omega_t^\ell\}_{\ell=1}^{t-1}\big)$, which determines the component of the mixture $\hat{\pi}_t^{\mathbf{y}}(\mathbf{x}_t)$. Next, we apply Algorithm 1 R times to update the remaining coordinates, treating s as fixed, and output the result $\bar{X}_t^R$. Note that an alternative to our method would be to consider a Gibbs sampler for which one of its marginal is directly the mixture (12) incorporating also the mixture index s as a state. However, this would then require sweeping over all states $(\bar{X}_0, \dots, \bar{X}_t)$, rendering it computationally expensive and impractical. We discuss other possible data augmentations and their limitations in Appendix A.4.

3.3 Practical implementation

For simplicity, we present the algorithm in the case where we progressively sample from each $\hat{\pi}_t^{\mathbf{y}}$ for $t \in [\![2, T]\!]$. In practice, however, we subsample a small number K of timesteps $(t_i)_{i=K}^1$, with $t_1 > 1$ and $t_K = T$, and apply the algorithm only to $(\hat{\pi}_{t_i}^{\mathbf{y}})_{i=K}^i$.

Algorithm 2 MIXTURE-GUIDED DIFFUSION MODEL

Input: Timesteps $(t_i)_{i=1}^K$ with $t_1 > 1$ and $t_K = T$, Gibbs repetitions R, DDPM steps M, gradient steps G, probabilities $\{\omega_{t_i}^\ell\}_{i=K,\ell=1}^{2,t_{i-1}}$
$\hat{X}_{t_K} \sim \mathcal{N}(0_d, \mathbf{I}_d)$
$\hat{X}_0 \leftarrow D_{t_K}^\theta(\hat{X}_{t_K})$, $\hat{X}_0^* \leftarrow \hat{X}_0$
for $i = K$ to 2 **do**
  $s \sim \text{Categorical}(\{\omega_{t_i}^\ell\}_{\ell=1}^{t_{i-1}})$
  $\hat{X}_0 \leftarrow \hat{X}_0^*$
  $\hat{X}_{t_i} \sim q_{t_i|0,t_{i+1}}(\cdot|\hat{X}_0^*, \hat{X}_{t_{i+1}})$
  for $r = 1$ **to** R **do**
    $\hat{X}_s \leftarrow \text{Gauss\_VI}(\hat{X}_0, \hat{X}_{t_i}, s, G)$ `//see A.3`
    $\hat{X}_0 \leftarrow \text{DDPM}(\hat{X}_s, s, M)$
    $\hat{X}_{t_i} \sim q_{t_i|s}(\cdot|\hat{X}_s)$
  end for
  $\hat{X}_0^* \leftarrow \hat{X}_0$
end for
Output: X_0^*

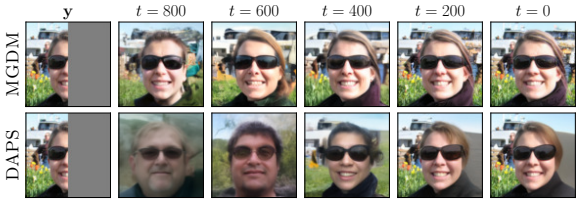

Figure 1: Evolution of $\hat{X}_0^*$ throughout the iterations for MGDM and DAPS (Zhang et al., 2024).

The denoising step in Algorithm 1 can be approximated by sampling from the learned diffusion model. To reduce runtime, we again subsample a small number of timesteps $\{s_i\}_{i=0}^M \subset [\![0, s-1]\!]$, ensuring that $s_0 = 0$ and $s_M = s$. We then generate $(X_{s_i})_{i=0}^M$ by sampling iteratively $X_{s_i} \sim p_{s_i|s_{i+1}}^\theta(\cdot|X_{s_{i+1}})$ and retaining only X_{s_0}. This operation is referred to as $\text{DDPM}(\cdot, s, M)$ on Line 10 in Algorithm 2. As for the step involving $\hat{\pi}_{s|0,t}^{\mathbf{y}}$, we follow Moufad et al. (2024) and sample approximately by fitting a Gaussian variational approximation. More specifically, given $(\mathbf{x}_0, \mathbf{x}_t)$, we draw from the Gaussian variational approximation $\lambda_{s|0,t}^{\boldsymbol{\varphi}} := \mathcal{N}\big(\boldsymbol{\mu}_{s|0,t}, \text{diag}(\mathrm{e}^{\boldsymbol{\rho}_{s|0,t}})\big)$ where the parameters $\boldsymbol{\varphi}_{s|0,t} := (\boldsymbol{\mu}_{s|0,t}, \boldsymbol{\rho}_{s|0,t}) \in \mathbb{R}^d \times \mathbb{R}^d$ are obtained by optimizing the right-hand side of

$$\begin{aligned}&\text{KL}(\lambda_{s|0,t}^{\boldsymbol{\varphi}} \parallel \pi_{s|0,t}^{\mathbf{y}}(\cdot|\mathbf{x}_0, \mathbf{x}_t))\\ &\approx -\mathbb{E}\big[\log \hat{g}_s^\theta(\mathbf{y}|\hat{X}_s^{\boldsymbol{\varphi}})\big] + \text{KL}(\lambda_{s|0,t}^{\boldsymbol{\varphi}} \parallel q_{s|0,t}(\cdot|\mathbf{x}_0, \mathbf{x}_t)),\end{aligned}$$

where $\hat{X}_s^{\boldsymbol{\varphi}} \sim \lambda_{s|0,t}^{\boldsymbol{\varphi}}$. The gradient of this quantity can be estimated straightforwardly using the reparameterization trick (Kingma & Welling, 2013). The initial parameters $\boldsymbol{\varphi}_{s|0,t}$ are set to the mean and covariance of $q_{s|0,t}(\cdot|\mathbf{x}_0, \mathbf{x}_t)$ defined in (3). This step corresponds to the Gauss_VI routine in Algorithm 2 and is detailed in Appendix A.3. Regarding the initialization of the GS for $\hat{\pi}_t^{\mathbf{y}}$, we use the output of the previous GS targeting $\hat{\pi}_{t+1}^{\mathbf{y}}$; see Lines 6 and 7 in Algorithm 2. We maintain a running variable $\hat{X}_0^*$, which is iteratively updated and serves as the initialization for the other variables at the beginning of each loop iteration. It is also the output of the algorithm. Indeed, note that the last distribution to which we apply the GS is $\overline{\pi}_{0,1,2}^{\mathbf{y}}(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2)$ of which the $\mathbf{x}_0$-marginal is proportional to $p_0(\mathbf{x}_0) \int \hat{g}_1(\mathbf{y}|\mathbf{x}_1) q_{1|0}(\mathbf{x}_1|\mathbf{x}_0)\, \mathrm{d}\mathbf{x}_1$. Since the Gaussian density $q_{1|0}(\cdot|\mathbf{x}_0)$ has a very small variance and $\hat{g}_1(\mathbf{y}|\cdot) \approx g_0(\mathbf{y}|\cdot)$, we may assume that $\int \hat{g}_1(\mathbf{y}|\mathbf{x}_1)\, q_{1|0}(\mathbf{x}_1|\mathbf{x}_0) \mathrm{d}\mathbf{x}_1 \approx g_0(\mathbf{y}|\mathbf{x}_0)$ and hence that the posterior $\pi_0^{\mathbf{y}}$ of interest is approximately the $\mathbf{x}_0$-marginal of the last extended distribution. As a result, we can take the $\mathbf{x}_0$-coordinate of the output of the last GS, which is $\hat{X}_0$ and hence $\hat{X}_0^*$, as an approximate sample from $\pi_0^{\mathbf{y}}$. In the first row of Figure 1 we display the evolution of $\hat{X}_0^*$ throughout the iterations with a DDM pre-trained on the `FFHQ` dataset. It is seen that the algorithm reaches a plausible reconstruction of $\mathbf{y}$ rather fast, at $t = 800$ with $T = 1000$, and then spends the remaining iterations refining the details. As a comparison, the DAPS algorithm proposed by Zhang et al. (2024), which displayed in the second row, also maintains a running variable at time 0 that serves as output to the algorithm.

4 Related works

Alternative likelihood approximations. In addition to this work, several other papers introduce alternative approximations of $g_t(\mathbf{y}|\cdot)$. (Song et al., 2023a) proposes a Gaussian approximation of $p_{0|t}$ with mean given by the denoiser D_t^θ and covariance being left as a hyperparameter. For linear inverse problems with Gaussian noise, the likelihood $g_0(\mathbf{y}|\cdot)$ can be integrated exactly against this Gaussian approximation, providing an alternative approximation of $g_t(\mathbf{y}|\cdot)$. Finzi et al. (2023); Stevens et al. (2023); Boys et al. (2023) use that the covariance of $p_{0|t}(\cdot|\mathbf{x}_t)$ is proportional to the Jacobian of the denoiser (Meng et al., 2021). Computing the score of the resulting likelihood approximation, for linear inverse problems, is prohibitively expensive. To mitigate this, these works and subsequent ones assume that the Jacobian of the denoiser is constant with respect to $\mathbf{x}_t$. Despite this simplification, the score approximation still involves an expensive matrix inversion. Boys et al. (2023) use diagonal approximation of the covariance based on its row sums. Rozet et al. (2024) use conjugate gradient to perform the matrix inversion efficiently. For general likelihoods $g_0(\mathbf{y}|\cdot)$, Song et al. (2023b) use Gaussian approximations of Song et al. (2023a) to estimate $g_t(\mathbf{y}|\cdot)$ using a standard Monte Carlo approach. For latent diffusion models, Rout et al. (2024) apply the approximation in (9) together with a regularization term that penalizes latent variables deviating from

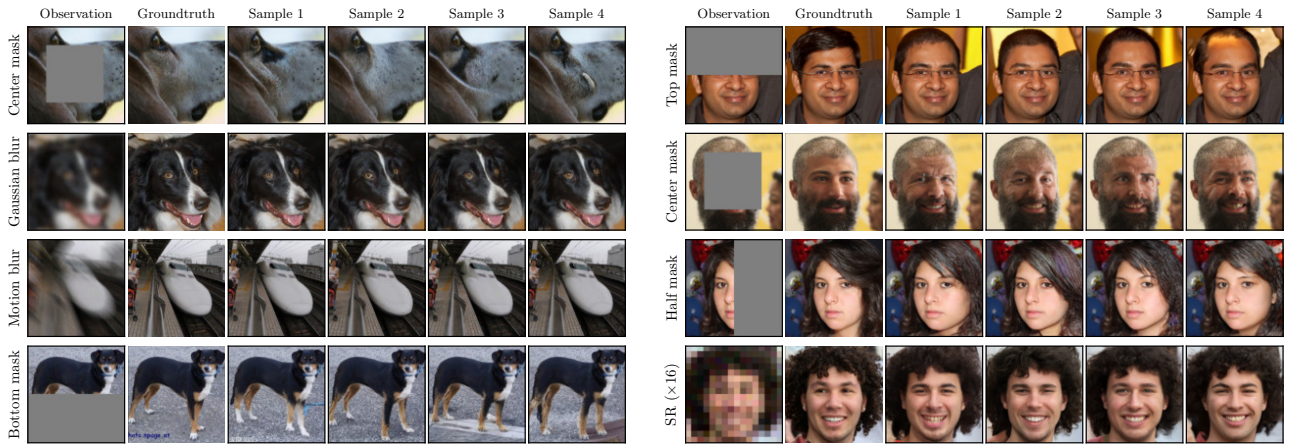

Figure 2: MGDM sample images for various tasks on `ImageNet` (left) and `FFHQ` (right) datasets.

fixed points of the decoder-encoder composition. Moufad et al. (2024) propose a general method for both vanilla and latent space diffusion models. At step t of the diffusion process they first sample, at an intermediate timestep $s < t$, a state conditionally on $\mathbf{y}$ with the approximation (9), before returning back to the timestep t. In Appendix A.5 we explain in more details how the present work differs from this method.

Asymptotically exact methods. Trippe et al. (2023); Wu et al. (2023); Cardoso et al. (2024); Dou & Song (2024); Corenflos et al. (2024); Li et al. (2024) use the sequential Monte Carlo (SMC) framework to construct an empirical approximation of the posterior distribution represented by N samples. The samples undergo transitions guided by user-defined updates, are reweighted using an appropriate importance weight, and are subsequently resampled to focus computational effort on the most promising candidates. The performance of these methods improves by scaling the number of samples N, which impacts both the memory requirement and compute time. As evidenced by the experiments in the next section, our method improves by increasing the number of Gibbs steps, which impacts only the runtime.

Gibbs sampling approaches. The recent works (Wu et al., 2024; Xu & Chi, 2024) on PNP-DM also propose a Gibbs sampling-inspired algorithm. They consider (within the variance exploding framework) the distribution sequence $(\tilde{\pi}_t^{\mathbf{y}})_{t=0}^T$, where each distribution $\tilde{\pi}_t^{\mathbf{y}}(\mathbf{x}_t) \propto g_0(\mathbf{y}|\mathbf{x}_t)p_t(\mathbf{x}_t)$ is the $\mathbf{x}_t$-marginal of the extended distribution

$$\tilde{\pi}_{0,t}^{\mathbf{y}}(\mathbf{x}_0, \mathbf{x}_t) \propto g_0(\mathbf{y}|\mathbf{x}_t)p_0(\mathbf{x}_0)q_{t|0}(\mathbf{x}_t|\mathbf{x}_0)\,.$$

As its full conditionals are $\tilde{\pi}_{0|t}^{\mathbf{y}}(\mathbf{x}_0|\mathbf{x}_t) = p_{0|t}(\mathbf{x}_0|\mathbf{x}_t)$ and $\tilde{\pi}_{t|0}^{\mathbf{y}}(\mathbf{x}_t|\mathbf{x}_0) \propto g_0(\mathbf{y}|\mathbf{x}_t)q_{t|0}(\mathbf{x}_t|\mathbf{x}_0)$, the GS targeting this joint distribution also proceeds with a prior denoising step. On the other hand, sampling from $\tilde{\pi}_{t|0}^{\mathbf{y}}(\cdot|\mathbf{x}_0)$ can be performed exactly when $g_0(\mathbf{y}|\cdot)$ is the likelihood of a linear inverse problem with Gaussian noise, since $q_{t|0}(\cdot|\mathbf{x}_0)$ is a Gaussian distribution. For more general problems, this step can be implemented using MCMC methods; see *e.g.* (Xu & Chi, 2024, Algorithms 3 & 4). Compared to our algorithm, PNP-DM has a lower memory footprint because it does not require a vector-Jacobian product as it uses the likelihood $g_0(\mathbf{y}|\cdot)$ instead of $\hat{g}_t(\mathbf{y}|\cdot)$. However, as we show in the next section, this comes at the cost of performance, especially when using latent diffusion models. The REPAINT algorithm (Lugmayr et al., 2022), which applies to noiseless linear inverse problems, uses noising and denoising steps repeatedly and can also be viewed as a variant of a specific Gibbs sampler. Finally, the recently proposed DAPS (Zhang et al., 2024) can also be related to a Gibbs sampler targeting a specific sequence of distributions. Further details and comparisons to MGDM are provided in Appendix A.5.

5 Experiments

We evaluate MGDM on image inverse problems using both pixel-space and latent-space diffusion, as well as on musical source separation tasks. For the pixel-space diffusion and the audio diffusion model, we compare MGDM against *seven* competitors: DPS (Chung et al., 2023), PGDM (Song et al., 2023a), DDNM (Wang et al., 2023), DIFFPIR (Zhu et al., 2023), REDDIFF (Mardani et al., 2024), DAPS (Zhang et al., 2024), and PNP-DM (Wu et al., 2024). In the latent space setting, we benchmark against *four* competitors: PSLD (Rout et al., 2024), RESAMPLE (Song et al., 2024), DAPS (Zhang et al., 2024), and PNP-DM (Wu et al., 2024). In Appendixes B.2-B.4, we provide a complete formal description of the parameters of our algorithm as well as the implementation details of each competitor and its hyperparameters. We emphasize that we have tuned the parameters

Table 1: Mean LPIPS for linear/nonlinear imaging tasks on the FFHQ and ImageNet datasets with $\sigma_{\mathbf{y}} = 0.05$. Lower metrics are better.

	FFHQ								ImageNet							
Task	MGDM	DPS	PGDM	DDNM	DiffPIR	ReDDiff	DAPS	PnP-DM	MGDM	DPS	PGDM	DDNM	DiffPIR	ReDDiff	DAPS	PnP-DM
SR (×4)	0.09	0.09	0.30	0.15	0.10	0.39	0.16	0.10	0.26	0.25	0.56	0.34	0.31	0.57	0.37	0.66
SR (×16)	0.24	0.23	0.42	0.33	0.23	0.55	0.40	0.29	0.55	0.44	0.62	0.71	0.50	0.85	0.75	1.03
Box inpainting	0.10	0.17	0.17	0.12	0.14	0.19	0.13	0.18	0.23	0.35	0.29	0.28	0.30	0.36	0.30	0.42
Half mask	0.20	0.24	0.24	0.23	0.25	0.28	0.23	0.32	0.31	0.40	0.34	0.38	0.40	0.46	0.40	0.54
Gaussian Deblur	0.12	0.17	0.87	0.20	0.12	0.24	0.24	0.14	0.30	0.37	1.00	0.45	0.30	0.53	0.59	0.76
Motion Deblur	0.09	0.17	–	–	–	0.22	0.19	0.21	0.22	0.40	–	–	–	0.39	0.42	0.52
JPEG (QF = 2)	0.14	0.34	1.12	–	–	0.32	0.22	0.29	0.38	0.60	1.32	–	–	0.49	0.45	0.56
Phase retrieval	0.11	0.40	–	–	–	0.26	0.14	0.34	0.55	0.62	–	–	–	0.61	0.50	0.66
Nonlinear deblur	0.27	0.51	–	–	–	0.68	0.28	0.31	0.41	0.82	–	–	–	0.66	0.41	0.49
HDR	0.12	0.40	–	–	–	0.20	0.10	0.19	0.21	0.84	–	–	–	0.19	0.14	0.31

of our algorithm per dataset and not per task.

Index sampling and Gibbs steps. During the first 75% of the diffusion process, at timestep t_i, we sample the index s from Uniform$[\tau, t_{i-1}]$ with $\tau = 10$ to mitigate instabilities. In the final 25% of the steps we set $s = t_{i-1}$ as this yields slightly improved results. On the image inverse problems we use 100 diffusion steps with $R = 1$ Gibbs step. On the source separation task we use 20 diffusion steps with $R = 6$ Gibbs steps. The choice of weight sequence $\{\omega_t^\ell\}_{t=T,\ell=1}^{2,t-1}$ plays an important role for the algorithm's performance. Intuitively, it holds that that $\hat{g}_t^s(\mathbf{y}|\cdot) \approx g_t(\mathbf{y}|\cdot)$ when $s \approx 0$, suggesting that for all $t \in [\![2, T]\!]$, the weights should be set to 0 beyond a certain threshold to ensure that s is sampled near 0. We found, however, that this strategy does not yield good performance for our algorithm. Instead, sampling the index uniformly leads to a faster mixing. On high-dimensional image datasets, we observe that when s is consistently sampled near 0 at all iterations, the algorithm struggles to overcome the errors that accumulate at initialization, leading to suboptimal reconstructions. We provide both quantitative and qualitative evidence in Appendix B.1.

Images. We evaluate our method on a diverse set of six linear inverse problems and four nonlinear inverse problems with three different image priors with 256×256 resolution: the pixel-space FFHQ model of Choi et al. (2021), the latent-space FFHQ of Rombach et al. (2022), and the ImageNet model of Dhariwal & Nichol (2021). We use the noise level $\sigma_{\mathbf{y}} = 0.05$ for all tasks. The linear problems include image inpainting with two masking configurations: a 150×150 central box mask and a half-mask covering the right side of the image; Super Resolution (SR) tasks with upscaling factors of $\times 4$ and $\times 16$; Gaussian and motion deblurring, both using a kernel size of 61×61 following the experimental setup described by Chung et al. (2023, Section 4). For the nonlinear setting, we consider JPEG dequantization with a quality factor of 2%, implemented using the differentiable operator proposed by Shin & Song (2017); phase retrieval with an oversampling factor of $\times 2$; non-uniform deblurring using the operator introduced by Tran et al. (2021); High Dynamic Range (HDR) reconstruction following the setup detailed in Mardani et al. (2024, Section 5.2). The evaluation is done on a subset of 300 validation images per dataset. For FFHQ, we use the first 300 images, while for ImageNet, we randomly sample 300 images to avoid class bias. We report the LPIPS metric (Zhang et al., 2018) in Tables 1 and 2 and defer the complete tables with FID, PSNR and SSIM along side 95% confidence interval to Table 6, Table 7, and Table 8. For the phase retrieval task specifically, we draw 4 samples for each algorithm and keep only the best scoring one in terms of LPIPS. A similar strategy is used in (Chung et al., 2023; Zhang et al., 2024; Wu et al., 2024). Across table rows, we highlight the best value in , the 2nd best in and 3rd best in . We provide a large gallery of exemplar reconstructions in Appendix B.9. Aside, we also extend our evaluation to higher-noise setup and Poisson-noise likelihood in Appendix B.8 and Appendix B.9.

Results. Our method with a single Gibbs step consistently achieves competitive performance, ranking first on most tasks and standing out as the only approach to maintain robust performance across all tasks. On latent FFHQ, we outperform ReSample and PSLD, both of which are specifically designed for latent problems, while our method is applied seamlessly off-the-shelf without any adaptation to latent diffusion. Qualitative comparisons in Figure 2 and in Appendix B.9 reveal that our method provides diverse, visually coherent and sharp reconstructions. In contrast, DAPS, DDNM and DiffPIR, despite scoring higher in PSNR and SSIM on some tasks, provide less coherent reconstructions; see Appendix B.6 for a discussion and examples. Finally, a key strength of our algorithm is its ability to improve performance by increasing the number R of Gibbs steps. This is demonstrated for the most challenging task, phase retrieval, in Figure 3. In this experiment, we compute the LPIPS using a single sample per image (instead of four) and achieve a threefold reduction in average LPIPS simply by increasing the compute time in the right direction. Indeed, increasing the number of gradient steps brings only marginal gains in this case whereas increasing the number of Gibbs steps leads to significant performance gains.

Source separation. We now consider a linear inverse problem with an audio diffusion prior that generates four dependent instrument soundtracks: bass, drums, guitar, and

Table 2: Mean LPIPS for linear/nonlinear imaging tasks on FFHQ dataset with LDM prior and $\sigma_{\mathbf{y}} = 0.05$. Lower metrics are better.

Task	MGDM	ReSample	PSLD	DAPS	PNP-DM
SR (×4)	0.14	0.22	0.21	0.28	0.40
SR (×16)	0.30	0.38	0.36	0.52	0.71
Box inpainting	0.18	0.22	0.27	0.37	0.31
Half mask	0.26	0.30	0.32	0.49	0.44
Gaussian Deblur	0.18	0.16	0.59	0.32	0.32
Motion Deblur	0.22	0.20	0.70	0.36	0.36
JPEG (QF = 2)	0.23	0.26	–	0.32	0.36
Phase retrieval	0.29	0.39	–	0.25	0.50
Nonlinear deblur	0.29	0.33	–	0.37	0.37
High dynamic range	0.16	0.12	–	0.24	0.24

	$R=1$	$R=2$	$R=4$	$R=6$	$R=1, G\gg 1$
Bass	15.46	18.07	18.53	18.49	19.89
Drums	16.28	17.93	18.19	18.07	18.95
Guitar	12.58	14.73	16.26	16.68	16.07
Piano	11.82	14.34	15.38	16.17	16.50
All	14.03	16.27	17.09	17.35	17.85

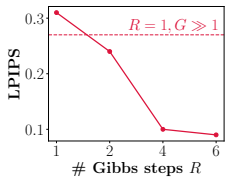

Figure 3: Performance of MGDM as a function of the number of Gibbs steps R. The setup $R = 1, G \gg 1$ represents MGDM with $R = 1$ and a number of gradient steps resulting in a runtime equivalent to using $R = 6$. Left: Mean SI-SDR$_\text{I}$ for multisource–audio separation task on slakh2100 test dataset. Right: Mean LPIPS for the phase retrieval task on FFHQ.

piano. The task involves separating the individual sources from a mixture $\mathbf{y}$ of these four instruments; *i.e.* denoting by d' the dimension of one instrument soundtrack, the linear operator is $\mathbf{A} : \mathbf{x} \in \mathbb{R}^{4\times d'} \mapsto \sum_{i=1}^{4} \mathbf{x}_i \in \mathbb{R}^{d'}$. We assume *no noise* in the measurement and use the audio diffusion model of Mariani et al. (2023). The evaluation is conducted on the publicly available slakh2100 test dataset (Manilow et al., 2019) with the scale-invariant SDR improvement (SI-SDR$_\text{I}$) metric (Roux et al., 2019). The SI-SDR$_\text{I}$ metric measures the improvement between the original audio source $\mathbf{x}_i$ and the generated source $\hat{\mathbf{x}}_i$, relative to the mixture baseline $\mathbf{y}$, *i.e.* it computes the difference SI-SDR$(\mathbf{x}_i, \hat{\mathbf{x}}_i)$ − SI-SDR$(\mathbf{x}_i, \mathbf{y})$ where

$$\text{SI-SDR}(\mathbf{x}_i, \hat{\mathbf{x}}_i) = 10 \log_{10} \frac{\|\alpha \mathbf{x}_i\|^2 + \epsilon}{\|\alpha \mathbf{x}_i - \hat{\mathbf{x}}_i\|^2 + \epsilon},$$

where $\alpha = \frac{\mathbf{x}_i^\top \hat{\mathbf{x}}_i + \epsilon}{\|\mathbf{x}_i\|^2 + \epsilon}$, and $\epsilon = 10^{-8}$. Following Mariani et al. (2023, Section 5.2), tracks from the test dataset are evaluated using a sliding window approach with 4-second chunks and a 2-second overlap. We report the SI-SDR$_\text{I}$ metric in Table 3. For this task we compare against three other competing algorithms. First, the best version of the MSDM algorithm in (Mariani et al., 2023) which uses the same pre-trained model and is directly comparable to our method. Then, the ISDM algorithm from the same paper and which relies on separate pre-trained models for each instrument, as well as the Demucs model (Défossez et al., 2019), trained with supervision to specifically solve source separation, augmented with 512 Gibbs sampling steps (Manilow et al., 2022) and

Table 3: Mean SI-SDR$_\text{I}$ on slakh2100 test dataset. The last row displays the mean over the four stems. Higher metrics are better.

Stems	MGDM	DPS	PGDM	DDNM	MSDM	ISDM	DEMUCS$_{512}$
Bass	18.49	16.50	16.41	14.94	17.12	19.36	17.16
Drums	18.07	18.29	18.14	19.05	18.68	20.90	19.61
Guitar	16.68	9.90	12.84	14.38	15.38	14.70	17.82
Piano	16.17	10.41	12.31	11.46	14.73	14.13	16.32
All	17.35	13.77	14.92	14.96	16.48	17.27	17.73

is, to the best of our knowledge, considered to be state-of-the-art. We refer to it as DEMUCS$_{512}$. Finally, since the inverse problem is noiseless, we smooth it by using the likelihood $g_0(\mathbf{y}|\mathbf{x}) = \mathrm{N}(\mathbf{y}; \mathbf{A}(\mathbf{x}), \sigma_{\mathbf{y}}^2 \mathbf{I}_{d_{\mathbf{y}}})$ with $\sigma_{\mathbf{y}} = 10^{-4}$. This smoothing is applied consistently across all competitors except the best-performing versions of MSDM and ISDM, which are tailored for noiseless problems, and DEMUCS$_{512}$. The results are reported in Table 3. Due to space constraints, we only show the best performing competitors and defer the complete table to Appendix B.6.

Results. We outperform, on average, the other training-free competitors that use the same pre-trained model by a substantial margin. In particular, we outperform the MSDM algorithm of Mariani et al. (2023) as well as ISDM which uses a different model. With $R = 6$ Gibbs steps MGDM falls short of matching the performance DEMUCS$_{512}$. We found instead that setting $R = 1$ and using a number of gradient steps ensuring equivalent runtime, as we did for the phase retrieval example, allows to achieve superior performance; see Figure 3. It is also seen that the average SI-SDR$_\text{I}$ increases monotonically with the number of Gibbs steps.

6 Conclusion

We have developed a novel posterior sampling scheme for denoising diffusion priors. The proposed algorithm proceeds by sequentially sampling, using a Gibbs sampler, from a sequence of mixture approximations of the smoothed posteriors. Our experiments show that MGDM not only matches but often surpasses state-of-the-art performance and reconstruction quality across various tasks. Furthermore, we have demonstrated that the Gibbs sampling perspective allows favorable performance improvement with inference-time compute scaling.

This work has certain limitations that open avenues for further exploration. While we outperform the state-of-the-art on most tasks and remains competitive overall on latent diffusion, we still fall short of what we achieve with pixel-space diffusion. We believe that bridging this gap requires a more careful selection of the weight sequence. More broadly, an observation-driven approach to sampling the index could further enhance MGDM. A second limitation is that our methodology does not extend to ODE-based samplers or DDIM, and adapting related ideas to these methods

is an interesting research direction. Finally, like all existing methods relying on (9), our approach incurs a higher memory cost compared to unconditional diffusion. It remains an open question whether the vector-Jacobian product can be eliminated without compromising performance.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

Acknowledgements

The work of Y.J. and B.M. has been supported by Technology Innovation Institute (TII), project Fed2Learn. The work is supported by the Swedish Research Council, project 2024-05680. The work of Eric Moulines has been partly funded by the European Union (ERC-2022-SYG-OCEAN-101071601). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them. This work was granted access to the HPC resources of IDRIS under the allocation 2025-AD011015980 made by GENCI.

References

Boys, B., Girolami, M., Pidstrigach, J., Reich, S., Mosca, A., and Akyildiz, O. D. Tweedie moment projected diffusions for inverse problems. *arXiv preprint arXiv:2310.06721*, 2023.

Cardoso, G., el idrissi, Y. J., Corff, S. L., and Moulines, E. Monte carlo guided denoising diffusion models for bayesian linear inverse problems. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=nHESwXvxWK.

Casella, G. and George, E. I. Explaining the gibbs sampler. *The American Statistician*, 46(3):167–174, 1992.

Choi, J., Kim, S., Jeong, Y., Gwon, Y., and Yoon, S. Ilvr: Conditioning method for denoising diffusion probabilistic models. in 2021 ieee. In *CVF international conference on computer vision (ICCV)*, volume 1, pp. 2, 2021.

Chung, H., Kim, J., Mccann, M. T., Klasky, M. L., and Ye, J. C. Diffusion posterior sampling for general noisy inverse problems. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=OnD9zGAGT0k.

Corenflos, A., Zhao, Z., Särkkä, S., Sjölund, J., and Schön, T. B. Conditioning diffusion models by explicit forward-backward bridging. *arXiv preprint arXiv:2405.13794*, 2024.

Daras, G., Chung, H., Lai, C.-H., Mitsufuji, Y., Milanfar, P., Dimakis, A. G., Ye, C., and Delbracio, M. A survey on diffusion models for inverse problems. 2024. *URL https://giannisdaras. github. io/publications/diffusion_survey. pdf*.

Défossez, A., Usunier, N., Bottou, L., and Bach, F. Music source separation in the waveform domain. *arXiv preprint arXiv:1911.13254*, 2019.

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

Dong, C., Loy, C. C., He, K., and Tang, X. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2015.

Dou, Z. and Song, Y. Diffusion posterior sampling for linear inverse problem solving: A filtering perspective. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=tplXNcHZs1.

Finzi, M. A., Boral, A., Wilson, A. G., Sha, F., and Zepeda-Núñez, L. User-defined event sampling and uncertainty quantification in diffusion models for physical dynamical systems. In *International Conference on Machine Learning*, pp. 10136–10152. PMLR, 2023.

Gelfand, A. E. Gibbs sampling. *Journal of the American statistical Association*, 95(452):1300–1304, 2000.

Geman, S. and Geman, D. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, (6):721–741, 1984.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Ho, J., Salimans, T., Gritsenko, A., Chan, W., Norouzi, M., and Fleet, D. J. Video diffusion models. *Advances in Neural Information Processing Systems*, 35:8633–8646, 2022.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1125–1134, 2017.

Jiang, C., Cornman, A., Park, C., Sapp, B., Zhou, Y., Anguelov, D., et al. Motiondiffuser: Controllable multi-agent motion prediction using diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9644–9653, 2023.

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

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

Kawar, B., Elad, M., Ermon, S., and Song, J. Denoising diffusion restoration models. *Advances in Neural Information Processing Systems*, 35:23593–23606, 2022.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4681–4690, 2017.

Li, X., Zhao, Y., Wang, C., Scalia, G., Eraslan, G., Nair, S., Biancalani, T., Ji, S., Regev, A., Levine, S., et al. Derivative-free guidance in continuous and discrete diffusion models with soft value-based decoding. *arXiv preprint arXiv:2408.08252*, 2024.

Lugmayr, A., Danelljan, M., Romero, A., Yu, F., Timofte, R., and Van Gool, L. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11461–11471, 2022.

Manilow, E., Wichern, G., Seetharaman, P., and Le Roux, J. Cutting music source separation some slakh: A dataset to study the impact of training data quality and quantity. In *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pp. 45–49, 2019. doi: 10.1109/WASPAA.2019.8937170.

Manilow, E., Hawthorne, C., Huang, C.-Z. A., Pardo, B., and Engel, J. Improving source separation by explicitly modeling dependencies between sources. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 291–295. IEEE, 2022.

Mardani, M., Song, J., Kautz, J., and Vahdat, A. A variational perspective on solving inverse problems with diffusion models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=1YO4EE3SPB.

Mariani, G., Tallini, I., Postolache, E., Mancusi, M., Cosmo, L., and Rodolà, E. Multi-source diffusion models for simultaneous music generation and separation. *arXiv preprint arXiv:2302.02257*, 2023.

Meng, C., Song, Y., Li, W., and Ermon, S. Estimating high order gradients of the data distribution by denoising. *Advances in Neural Information Processing Systems*, 34:25359–25369, 2021.

Moufad, B., Janati, Y., Bedin, L., Durmus, A., Douc, R., Moulines, E., and Olsson, J. Variational diffusion posterior sampling with midpoint guidance. *arXiv preprint arXiv:2410.09945*, 2024.

Robbins, H. E. An empirical bayes approach to statistics. In *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*, 1956. URL https://api.semanticscholar.org/CorpusID:26161481.

Roberts, G. O. and Smith, A. F. Simple conditions for the convergence of the gibbs sampler and metropolis-hastings algorithms. *Stochastic processes and their applications*, 49(2):207–216, 1994.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. 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.

Rout, L., Raoof, N., Daras, G., Caramanis, C., Dimakis, A., and Shakkottai, S. Solving linear inverse problems provably via posterior sampling with latent diffusion models. *Advances in Neural Information Processing Systems*, 36, 2024.

Roux, J. L., Wisdom, S., Erdogan, H., and Hershey, J. R. Sdr – half-baked or well done? In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 626–630, 2019. doi: 10.1109/ICASSP.2019.8683855.

Rozet, F. and Louppe, G. Score-based data assimilation. *Advances in Neural Information Processing Systems*, 36:40521–40541, 2023.

Rozet, F., Andry, G., Lanusse, F., and Louppe, G. Learning diffusion priors from observations by expectation maximization. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=7v88Fh6iSM.

Schneider, F., Kamal, O., Jin, Z., and Schölkopf, B. Musai: text-to-music generation with long-context latent diffusion. arxiv preprint. *arXiv preprint arXiv:2301.11757*, 2023.

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

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pp. 2256–2265. PMLR, 2015.

Song, B., Kwon, S. M., Zhang, Z., Hu, X., Qu, Q., and Shen, L. Solving inverse problems with latent diffusion models via hard data consistency. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=j8hdRqOUhN.

Song, J., Vahdat, A., Mardani, M., and Kautz, J. Pseudoinverse-guided diffusion models for inverse problems. In *International Conference on Learning Representations*, 2023a. URL https://openreview.net/forum?id=9_gsMA8MRKQ.

Song, J., Zhang, Q., Yin, H., Mardani, M., Liu, M.-Y., Kautz, J., Chen, Y., and Vahdat, A. Loss-guided diffusion models for plug-and-play controllable generation. In *International Conference on Machine Learning*, pp. 32483–32498. PMLR, 2023b.

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

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

Stevens, T. S., van Gorp, H., Meral, F. C., Shin, J., Yu, J., Robert, J.-L., and van Sloun, R. J. Removing structured noise with diffusion models. *arXiv preprint arXiv:2302.05290*, 2023.

Tran, P., Tran, A. T., Phung, Q., and Hoai, M. 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.

Trippe, B. L., Yim, J., Tischer, D., Baker, D., Broderick, T., Barzilay, R., and Jaakkola, T. S. Diffusion probabilistic modeling of protein backbones in 3d for the motif-scaffolding problem. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=6TxBxqNME1Y.

Wang, Y., Yu, J., and Zhang, J. Zero-shot image restoration using denoising diffusion null-space model. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=mRieQgMtNTQ.

Wu, L., Trippe, B. L., Naesseth, C. A., Cunningham, J. P., and Blei, D. Practical and asymptotically exact conditional sampling in diffusion models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=eWKqr1zcRv.

Wu, Z., Sun, Y., Chen, Y., Zhang, B., Yue, Y., and Bouman, K. Principled probabilistic imaging using diffusion models as plug-and-play priors. *Advances in Neural Information Processing Systems*, 37:118389–118427, 2024.

Xia, W., Zhang, Y., Yang, Y., Xue, J.-H., Zhou, B., and Yang, M.-H. Gan inversion: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 45(3):3121–3138, 2022.

Xu, X. and Chi, Y. Provably robust score-based diffusion posterior sampling for plug-and-play image reconstruction. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=SLnsoaY4u1.

Yu, J., Wang, Y., Zhao, C., Ghanem, B., and Zhang, J. Freedom: Training-free energy-guided conditional diffusion model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 23174–23184, 2023.

Zhang, B., Chu, W., Berner, J., Meng, C., Anandkumar, A., and Song, Y. Improving diffusion inverse problem solving with decoupled noise annealing. *arXiv preprint arXiv:2407.01521*, 2024.

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 586–595, 2018.

Zhu, Y., Zhang, K., Liang, J., Cao, J., Wen, B., Timofte, R., and Van Gool, L. Denoising diffusion models for plug-and-play image restoration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1219–1229, 2023.

A Methodology details

A.1 Primer on Gibbs sampling

In this section we lay out the basic properties of Gibbs sampling. We use measure-theoretic notation for conciseness.

Let $\mu_{0,1}(\mathrm{d}(\mathbf{x}_0, \mathbf{x}_1))$ be a probability measure on $\mathbb{R}^d \times \mathbb{R}^d$. We denote by $\mu_{0|1}(\mathrm{d}\mathbf{x}_0|\mathbf{x}_1)$ and $\mu_{1|0}(\mathrm{d}\mathbf{x}_1|\mathbf{x}_0)$ the associated full conditionals and we write μ_0, μ_1 for its marginals. Define the transition kernels

$$P_0(\mathrm{d}(\mathbf{x}_0', \mathbf{x}_1')|\mathbf{x}_0, \mathbf{x}_1) := \mu_{0|1}(\mathrm{d}\mathbf{x}_0'|\mathbf{x}_1)\delta_{\mathbf{x}_1}(\mathrm{d}\mathbf{x}_1'),$$
$$P_1(\mathrm{d}(\mathbf{x}_0', \mathbf{x}_1')|\mathbf{x}_0, \mathbf{x}_1) := \mu_{1|0}(\mathrm{d}\mathbf{x}_1'|\mathbf{x}_0)\delta_{\mathbf{x}_0}(\mathrm{d}\mathbf{x}_0').$$

Each transition kernel updates only one coordinate at a time. A full update of the coordinates is obtained by composition of the kernels, *i.e.*

$$P_0P_1(\mathrm{d}(\mathbf{x}_0', \mathbf{x}_1')|\mathbf{x}_0, \mathbf{x}_1) := \int P_1(\mathrm{d}(\mathbf{x}_0', \mathbf{x}_1')|\tilde{\mathbf{x}}_0, \tilde{\mathbf{x}}_1)P_0(\mathrm{d}(\tilde{\mathbf{x}}_0, \tilde{\mathbf{x}}_1)|\mathbf{x}_0, \mathbf{x}_1)\,.$$

Each transition admits the joint distribution $\mu_{0,1}$ as stationary distribution, meaning that $\mu_{0,1}(\mathrm{d}(\mathbf{x}_0, \mathbf{x}_1)) = \int P_0(\mathrm{d}(\mathbf{x}_0, \mathbf{x}_1)|\mathbf{x}_0', \mathbf{x}_1')\,\mu_{0,1}(\mathrm{d}(\mathbf{x}_0', \mathbf{x}_1'))$. Indeed, this is seen by noting that

$$\begin{aligned}
P_0(\mathrm{d}(\mathbf{x}_0, \mathbf{x}_1)|\mathbf{x}_0', \mathbf{x}_1')\mu_{0,1}(\mathrm{d}(\mathbf{x}_0', \mathbf{x}_1')) &= \mu_{0|1}(\mathrm{d}\mathbf{x}_0|\mathbf{x}_1')\delta_{\mathbf{x}_1'}(\mathrm{d}\mathbf{x}_1)\mu_{0,1}(\mathrm{d}(\mathbf{x}_0', \mathbf{x}_1')) \\
&= \mu_{0|1}(\mathrm{d}\mathbf{x}_0|\mathbf{x}_1')\delta_{\mathbf{x}_1'}(\mathrm{d}\mathbf{x}_1)\mu_{0|1}(\mathrm{d}\mathbf{x}_0'|\mathbf{x}_1')\mu_1(\mathrm{d}\mathbf{x}_1') \\
&= \mu_{0|1}(\mathrm{d}\mathbf{x}_0|\mathbf{x}_1)\mu_1(\mathrm{d}\mathbf{x}_1)\mu_{0|1}(\mathrm{d}\mathbf{x}_0'|\mathbf{x}_1')\delta_{\mathbf{x}_1}(\mathrm{d}\mathbf{x}_1'),
\end{aligned}$$

and then integrating both sides w.r.t. $(\mathbf{x}_0', \mathbf{x}_1')$. It then follows immediately that also P_0P_1 admits $\mu_{0,1}$ as stationary distribution. Letting $\big((X_0^k, X_1^k)\big)_{k\in\mathbb{N}}$ be a Markov chain with transition kernel P_0P_1, the law of (X_0^k, X_1^k) converges to $\mu_{0,1}$ as $k \to \infty$ under mild conditions; see (Roberts & Smith, 1994).

A.2 Full Gibbs conditionals

In the main paper we consider the following data augmentation of the mixture $\hat{\pi}_t^{\mathbf{y}}$ (12)

$$\overline{\pi}_{0,s,t}^{\mathbf{y}}(\mathbf{x}_0, \mathbf{x}_s, \mathbf{x}_t) = p_{0|s}(\mathbf{x}_0|\mathbf{x}_s)\frac{\hat{g}_s(\mathbf{y}|\mathbf{x}_s)p_{s|t}(\mathbf{x}_s|\mathbf{x}_t)p_t(\mathbf{x}_t)}{\int \hat{g}_s(\mathbf{y}|\mathbf{x}_s')p_{s|t}(\mathbf{x}_s'|\mathbf{x}_t')p_t(\mathbf{x}_t')\,\mathrm{d}\mathbf{x}_{s,t}'}\,. \tag{14}$$

From this definition it is straightforward to see that $\overline{\pi}_{0|s,t}^{\mathbf{y}}(\mathbf{x}_0|\mathbf{x}_s, \mathbf{x}_t) = p_{0|s}(\mathbf{x}_0|\mathbf{x}_s)$. In order to compute the full conditional $\overline{\pi}_{s|0,t}^{\mathbf{y}}(\mathbf{x}_s|\mathbf{x}_0, \mathbf{x}_t)$ we use the identity

$$p_{0|s}(\mathbf{x}_0|\mathbf{x}_s)p_{s|t}(\mathbf{x}_s|\mathbf{x}_t)p_t(\mathbf{x}_t) = p_0(\mathbf{x}_0)q_{s|0}(\mathbf{x}_s|\mathbf{x}_0)q_{t|s}(\mathbf{x}_t|\mathbf{x}_s), \tag{15}$$

from which it follows that

$$\begin{aligned}
\overline{\pi}_{s|0,t}^{\mathbf{y}}(\mathbf{x}_s|\mathbf{x}_0, \mathbf{x}_t) &= \frac{p_{0|s}(\mathbf{x}_0|\mathbf{x}_s)\hat{g}_s(\mathbf{y}|\mathbf{x}_s)p_{s|t}(\mathbf{x}_s|\mathbf{x}_t)}{\int p_{0|s}(\mathbf{x}_0|\mathbf{x}_s')\hat{g}_s(\mathbf{y}|\mathbf{x}_s')p_{s|t}(\mathbf{x}_s'|\mathbf{x}_t)\,\mathrm{d}\mathbf{x}_s'} \\
&= \frac{q_{s|0}(\mathbf{x}_s|\mathbf{x}_0)\hat{g}_s(\mathbf{y}|\mathbf{x}_s)q_{t|s}(\mathbf{x}_t|\mathbf{x}_s)}{\int q_{s|0}(\mathbf{x}_s'|\mathbf{x}_0)\hat{g}_s(\mathbf{y}|\mathbf{x}_s')q_{t|s}(\mathbf{x}_t|\mathbf{x}_s')\,\mathrm{d}\mathbf{x}_s'} \\
&= \frac{q_{s|0}(\mathbf{x}_s|\mathbf{x}_0)\hat{g}_s(\mathbf{y}|\mathbf{x}_s)q_{t|s}(\mathbf{x}_t|\mathbf{x}_s)/q_{t|0}(\mathbf{x}_t|\mathbf{x}_0)}{\int q_{s|0}(\mathbf{x}_s'|\mathbf{x}_0)\hat{g}_s(\mathbf{y}|\mathbf{x}_s')q_{t|s}(\mathbf{x}_t|\mathbf{x}_s')/q_{t|0}(\mathbf{x}_t|\mathbf{x}_0)\,\mathrm{d}\mathbf{x}_s'}\,.
\end{aligned}$$

Then, by noting that the bridge transition (3) satisfies $q_{s|0,t}(\mathbf{x}_s|\mathbf{x}_0, \mathbf{x}_t) = q_{s|0}(\mathbf{x}_s|\mathbf{x}_0)q_{t|s}(\mathbf{x}_t|\mathbf{x}_s)/q_{t|0}(\mathbf{x}_t|\mathbf{x}_0)$, we find that

$$\overline{\pi}_{s|0,t}^{\mathbf{y}}(\mathbf{x}_s|\mathbf{x}_0, \mathbf{x}_t) = \frac{\hat{g}_s(\mathbf{y}|\mathbf{x}_s)q_{s|0,t}(\mathbf{x}_s|\mathbf{x}_0, \mathbf{x}_t)}{\int \hat{g}_s(\mathbf{y}|\mathbf{x}_s')q_{s|0,t}(\mathbf{x}_s'|\mathbf{x}_0, \mathbf{x}_t)\,\mathrm{d}\mathbf{x}_s'}$$

Finally, for the third conditional, using again the identity (15), we find that

$$\begin{aligned}
\overline{\pi}_{t|0,s}^{\mathbf{y}}(\mathbf{x}_t|\mathbf{x}_0, \mathbf{x}_s) &= \frac{p_{0|s}(\mathbf{x}_0|\mathbf{x}_s)\hat{g}_s(\mathbf{y}|\mathbf{x}_s)p_{s|t}(\mathbf{x}_s|\mathbf{x}_t)p_t(\mathbf{x}_t)}{\int p_{0|s}(\mathbf{x}_0|\mathbf{x}_s)\hat{g}_s(\mathbf{y}|\mathbf{x}_s)p_{s|t}(\mathbf{x}_s|\mathbf{x}_t')p_t(\mathbf{x}_t')\,\mathrm{d}\mathbf{x}_t'} \\
&= q_{t|s}(\mathbf{x}_t|\mathbf{x}_s)\,.
\end{aligned}$$

A.3 Variational approximation

In this section we describe the variational approach of Moufad et al. (2024), which we use to fit a Gaussian variational approximation to $\pi^{\mathbf{y}}_{s|0,t}(\cdot|\mathbf{x}_0, \mathbf{x}_t)$ for fixed $(\mathbf{x}_0, \mathbf{x}_t)$. Similarly to the main paper we consider the variational approximation

$$\lambda^{\boldsymbol{\varphi}}_{s|0,t} := \mathcal{N}\big(\boldsymbol{\mu}_{s|0,t}, \mathrm{diag}(\mathrm{e}^{\boldsymbol{\rho}_{s|0,t}})\big), \tag{16}$$

and let $\boldsymbol{\varphi}_{s|0,t} := (\boldsymbol{\mu}_{s|0,t}, \boldsymbol{\rho}_{s|0,t}) \in \mathbb{R}^d \times \mathbb{R}^d$ denote the variational parameters. The reverse KL divergence writes, following definition (3),

$$\begin{aligned}
&\mathsf{KL}(\lambda^{\boldsymbol{\varphi}}_{s|0,t} \parallel \pi^{\mathbf{y}}_{s|0,t}(\cdot|\mathbf{x}_0, \mathbf{x}_t)) \\
&\quad = \int \log \frac{\lambda^{\boldsymbol{\varphi}}_{s|0,t}(\mathbf{x}_s)}{\hat{g}_s(\mathbf{y}|\mathbf{x}_s) q_{s|0,t}(\mathbf{x}_s|\mathbf{x}_0, \mathbf{x}_t)} \lambda^{\boldsymbol{\varphi}}_{s|0,t}(\mathbf{x}_s)\, \mathrm{d}\mathbf{x}_s + \mathrm{C} \\
&\quad = \mathbb{E}_{\lambda^{\boldsymbol{\varphi}}_{s|0,t}}\left[-\log \hat{g}_s(\mathbf{y}|\hat{X}^{\boldsymbol{\varphi}}_s) + \frac{\|\hat{X}^{\boldsymbol{\varphi}}_s - \big(\gamma_{t|s}\alpha_{s|0}\mathbf{x}_0 + (1-\gamma_{t|s})\alpha^{-1}_{t|s}\mathbf{x}_t\big)\|^2}{2\sigma^2_{s|0,t}} \right] - \frac{1}{2}\boldsymbol{\rho}^T_{s|0,t}\mathbf{1}_d + \mathrm{C}'.
\end{aligned} \tag{17}$$

Using the reparameterization trick (Kingma & Welling, 2013) and plugging-in the neural network approximation $\hat{g}^\theta_s(\mathbf{y}|\cdot)$ of $\hat{g}_s(\mathbf{y}|\cdot)$, we obtain the gradient estimator

$$\begin{aligned}
\nabla_{\boldsymbol{\varphi}} \mathcal{L}^s_t(\boldsymbol{\varphi}; \mathbf{x}_0, \mathbf{x}_t, Z) := &-\nabla_{\boldsymbol{\varphi}} \log \hat{g}^\theta_s(\mathbf{y}|\boldsymbol{\mu}_{s|0,t} + \mathrm{diag}(\mathrm{e}^{\boldsymbol{\rho}_{s|0,t}})^{1/2} Z) \\
&+ \nabla_{\boldsymbol{\varphi}} \left[\frac{\|\boldsymbol{\mu}_{s|0,t} + \mathrm{diag}(\mathrm{e}^{\boldsymbol{\rho}_{s|0,t}})^{1/2} Z - \big(\gamma_{t|s}\alpha_{s|0}\mathbf{x}_0 + (1-\gamma_{t|s})\alpha^{-1}_{t|s}\mathbf{x}_t\big)\|^2}{2\sigma^2_{s|0,t}} - \frac{1}{2}\boldsymbol{\rho}^T_{s|0,t}\mathbf{1}_d \right],
\end{aligned}$$

where $Z \sim \mathcal{N}(0_d, \mathbf{I}_d)$. We initialize the variational parameters with the mean and covariance of the bridge kernel (3), *i.e.*, at initialization, $\boldsymbol{\mu}^0_{s|0,t} := \gamma_{t|s}\alpha_{s|0}\mathbf{x}_0 + (1-\gamma_{t|s})\alpha^{-1}_{t|s}\mathbf{x}_t$ and $\boldsymbol{\rho}^0_{s|0,t} = \log \sigma^2_{s|0,t}\mathbf{I}_d$. The Gauss_VI routine is summarized in Algorithm 3.

Algorithm 3 Gauss_VI routine

Input: vectors $(\mathbf{x}_0, \mathbf{x}_t)$, timesteps (s, t), gradient steps G
$\boldsymbol{\mu} \leftarrow \gamma_{t|s}\alpha_{s|0}\mathbf{x}_0 + (1-\gamma_{t|s})\alpha^{-1}_{t|s}\mathbf{x}_t$
$\boldsymbol{\rho} \leftarrow \log \sigma^2_{s|0,t}$
for $g = 1$ to G **do**
  $Z \sim \mathcal{N}(0_d, \mathbf{I}_d)$
  $(\boldsymbol{\mu}, \boldsymbol{\rho}) \leftarrow \mathsf{OptimizerStep}(\nabla_{\boldsymbol{\varphi}} \mathcal{L}^s_t(\cdot, \mathbf{x}_0, \mathbf{x}_t, Z))$
end for
$Z \sim \mathcal{N}(0_d, \mathbf{I}_d)$
Output: $\boldsymbol{\mu} + \mathrm{diag}(\mathrm{e}^{\boldsymbol{\rho}/2})Z$

Remark A.1. While the expectation of the squared norm in (17) can be computed exactly, we found that, in practice, doing so degraded the algorithm's performance, producing blurrier images compared to simply using a Monte Carlo estimator for the full expectation.

Remark A.2. The fact that the density of our target distribution can be computed approximately by plugging the denoiser approximation allows us to add a Metropolis–Hastings (MH) correction with approximate acceptance ratio. Indeed, once we fit the Gaussian approximation, we can improve the accuracy of our sampler by simulating a Markov chain $(\hat{X}^k_s)_k$ where, given $\hat{X}^k_s$,

$$\hat{X}^{k+1}_s \sim M_s(\mathrm{d}\mathbf{x}_s|\hat{X}^k_s) := \int \lambda^{\boldsymbol{\varphi}}_{s|0,t}(z) \left[r_s(\hat{X}^k_s, z)\delta_z(\mathrm{d}\mathbf{x}_s) + (1 - r_s(\hat{X}^k_s, z))\delta_{\hat{X}^k_s}(\mathrm{d}\mathbf{x}_s) \right] \mathrm{d}z\,,$$

with

$$r_s(\mathbf{x}_s, \mathbf{x}^*_s) = \min\left(1, \frac{\hat{g}_s(\mathbf{y}|\mathbf{x}^*_s) q_{s|0,t}(\mathbf{x}^*_s|\mathbf{x}_0, \mathbf{x}_t) \lambda^{\boldsymbol{\varphi}}_{s|0,t}(\mathbf{x}_s)}{\hat{g}_s(\mathbf{y}|\mathbf{x}_s) q_{s|0,t}(\mathbf{x}_s|\mathbf{x}_0, \mathbf{x}_t) \lambda^{\boldsymbol{\varphi}}_{s|0,t}(\mathbf{x}^*_s)}\right).$$

A.4 Alternative data augmentation and sequence

Data augmentation. Our algorithm is based on one data-augmentation approach, but alternative augmentations could also be considered. Let $s \in [\![1, t-1]\!]$. Then the most obvious and natural data augmentation involves simply marginalizing out the $\mathbf{x}_0$ variable in (14), yielding

$$\overline{\pi}^{\mathbf{y}}_{s,t}(\mathbf{x}_s, \mathbf{x}_t) \propto \hat{g}_s(\mathbf{y}|\mathbf{x}_s) p_{s|t}(\mathbf{x}_s|\mathbf{x}_t) p_t(\mathbf{x}_t)\,.$$

Its full conditionals are $\overline{\pi}^{\mathbf{y}}_{s|t}(\mathbf{x}_s|\mathbf{x}_t) \propto \hat{g}_s(\mathbf{y}|\mathbf{x}_s) p_{s|t}(\mathbf{x}_s|\mathbf{x}_t)$ and $\overline{\pi}^{\mathbf{y}}_{t|s}(\mathbf{x}_t|\mathbf{x}_s) = q_{t|s}(\mathbf{x}_t|\mathbf{x}_s)$. The first conditional is intractable for sampling, and we could approximate it with a Gaussian variational distribution, similar to our approach for $\overline{\pi}^{\mathbf{y}}_{s|0,t}(\cdot|\mathbf{x}_0, \mathbf{x}_t)$. Indeed, this is possible since $\nabla_{\mathbf{x}_s} \log \overline{\pi}^{\mathbf{y}}_{s|t}(\mathbf{x}_s|\mathbf{x}_t) = \nabla_{\mathbf{x}_s} \log \hat{g}_s(\mathbf{y}|\mathbf{x}_s) + \nabla_{\mathbf{x}_s} \log p_s(\mathbf{x}_s) + \nabla_{\mathbf{x}_s} \log q_{t|s}(\mathbf{x}_t|\mathbf{x}_s)$, which can then be approximated using the parametric approximations $\nabla \log \hat{g}^{\theta}_s(\mathbf{y}|\mathbf{x}_s)$ and $\nabla \log p_s(\mathbf{x}_s) \approx (-\mathbf{x}_s + \alpha_s D^{\theta}_s(\mathbf{x}_s))/(1-\alpha_s^2)$.

The first drawback of this approach is that, in practice, it tends to degrade reconstruction quality—*e.g.*, introducing blurriness—as t tends to 0, due to the poor approximation of the score near the data distribution. Additionally, beyond the loss of quality, we observe that it produces more incoherent reconstructions with noticeable artifacts. We hypothesize that this issue arises because the distribution we aim to approximately sample involves the prior transition $p_{s|t}$, which can be highly multi-modal when $s \ll t$. This multi-modality may make the posterior $\overline{\pi}^{\mathbf{y}}_{s|t}(\cdot|\mathbf{x}_t)$ more challenging to approximately sample from. On the other hand, when further conditioning on $\mathbf{x}_0$, the sampling problem becomes more well-behaved, as we then target the posterior of a Gaussian distribution. Finally, while the score of $\overline{\pi}^{\mathbf{y}}_{s|t}(\mathbf{x}_s|\mathbf{x}_t)$ can be easily approximated, its density cannot, preventing the use of a Metropolis–Hastings correction, unless we use the independent proposal $p_{s|t}(\cdot|\mathbf{x}_t)$. However, this approach is suboptimal, as it does not incorporate any information from the observation. This is not the case of the data-augmentation approach we use in MGDM as we highlight in Remark A.2.

Alternative sequence. An alternative to the mixture of posterior approximations (12), on which MGDM is based, is the posterior formed as a mixture of likelihoods:

$$\hat{\pi}^{\mathbf{y}}_t(\mathbf{x}_t) = \frac{\sum_{s=1}^{t-1} \omega_t^s \hat{g}_t^s(\mathbf{y}|\mathbf{x}_t) p_t(\mathbf{x}_t)}{\int \sum_{s=1}^{t-1} \omega_t^s \hat{g}_t^s(\mathbf{y}|\mathbf{x}'_t) p_t(\mathbf{x}'_t)\,\mathrm{d}\mathbf{x}'_t}\,,$$

being the $\mathbf{x}_t$-marginal of the extended distribution

$$\overline{\pi}^{\mathbf{y}}_{0,\setminus,t}(s, \mathbf{x}_0, \mathbf{z}, \mathbf{x}_t) \propto \omega_t^s p_{0|s}(\mathbf{x}_0|\mathbf{z}) \hat{g}_s(\mathbf{y}|\mathbf{z}) p_{s|t}(\mathbf{z}|\mathbf{x}_t) p_t(\mathbf{x}_t)\,. \tag{18}$$

Now, let $(s, \bar{X}_0, \bar{Z}, \bar{X}_t) \sim \overline{\pi}^{\mathbf{y}}_{0,\setminus,t}$; then, conditionally on s, the distribution of $(\bar{X}_0, \bar{Z}, \bar{X}_t)$ is $\overline{\pi}^{\mathbf{y}}_{0,s,t}$, whereas

$$s|\bar{X}_0, \bar{Z}, \bar{X}_t \sim \text{Categorical}\left(\left\{\frac{\omega_t^{\ell} \hat{g}_{\ell}(\mathbf{y}|\bar{Z}) q_{\ell|0,t}(\bar{Z}|\bar{X}_0, \bar{X}_t)}{\sum_{k=1}^{t-1} \omega_t^k \hat{g}_k(\mathbf{y}|\bar{Z}) q_{k|0,t}(\bar{Z}|\bar{X}_0, \bar{X}_t)}\right\}_{\ell=1}^{t-1}\right)\,.$$

A Gibbs sampler targeting (18) is described in Algorithm 4. It allows updating the index s in an observation-driven fashion, but is unfortunately computationally expensive as we need to evaluate the denoiser at $\bar{Z}$ in parallel for $t-1$ timesteps. A cheaper alternative could be to block the variables $(s, \bar{Z})$ and use an independent MH step to target their joint conditional distribution. Denoting by λ the joint proposal distribution on $[\![1, t-1]\!] \times \mathbb{R}^d$ used in this independent MH step, the probability of accepting a candidate $(s^*, \mathbf{z}^*)$ is

$$r_t\big((s, \mathbf{z}), (s^*, \mathbf{z}^*)\big) = \min\left(1, \frac{\omega_t^{s^*} \hat{g}_{s^*}(\mathbf{y}|\mathbf{z}^*) q_{s^*|0,t}(\mathbf{z}^*|\mathbf{x}_0, \mathbf{x}_t) \lambda(s, \mathbf{z})}{\omega_t^s \hat{g}_s(\mathbf{y}|\mathbf{z}) q_{s|0,t}(\mathbf{z}|\mathbf{x}_0, \mathbf{x}_t) \lambda(s^*, \mathbf{z}^*)}\right)\,.$$

Remark A.3. Note that we could have used a similar data augmentation (18) for the mixture used in MGDM. This would yield the full conditional

$$s|\bar{X}_0, \bar{Z}, \bar{X}_t \sim \text{Categorical}\left(\left\{\frac{\omega_t^{\ell} \overline{\pi}^{\mathbf{y}}_{\ell|0,t}(\bar{Z}|\bar{X}_0, \bar{X}_t)}{\sum_{k=1}^{t-1} \omega_t^k \overline{\pi}^{\mathbf{y}}_{\ell|0,t}(\bar{Z}|\bar{X}_0, \bar{X}_t)}\right\}_{k=1}^{t-1}\right)\,,$$

which is, however, intractable due to the normalizing constant involved in each $\overline{\pi}^{\mathbf{y}}_{\ell|0,t}$.

Algorithm 4 Gibbs sampler targeting (13)

Input: $(s^r, \bar{X}_0^r, \bar{Z}^r, \bar{X}_t^r)$
draw $s^{r+1} \sim \text{Categorical}\left(\left\{\frac{\omega_t^\ell \hat{g}_\ell(\mathbf{y}|\bar{Z}^r) q_{\ell|0,t}(\bar{Z}^r|\bar{X}_0^r, \bar{X}_t^r)}{\sum_{k=1}^{t-1} \omega_t^k \hat{g}_k(\mathbf{y}|\bar{Z}^r) q_{k|0,t}(\bar{Z}^r|\bar{X}_0^r, \bar{X}_t^r)}\right\}_{k=1}^{t-1}\right)$
draw $\bar{Z}^{r+1} \sim \overline{\pi}^{\mathbf{y}}_{s^{r+1}|0,t}(\cdot|\bar{X}_0^r, \bar{X}_t^r)$
draw $\bar{X}_t^{r+1} \sim q_{t|s^{r+1}}(\cdot|\bar{Z}^{r+1})$
draw $\bar{X}_0^{r+1} \sim p_{0|s^{r+1}}(\cdot|\bar{Z}^{r+1})$

A.5 Related algorithms

Comparison with Zhang et al. (2024) In this section we clarify the difference between MGDM and the DAPS algorithm (Zhang et al., 2024), which shares some similarities with our approach. The sampling procedure in DAPS relies on sequential approximate sampling from the joint distribution

$$\tilde{\pi}^{\mathbf{y}}_{0:T}(\mathbf{x}_{0:T}) := \pi^{\mathbf{y}}_T(\mathbf{x}_T) \prod_{t=0}^{T-1} \tilde{\pi}_{t|t+1}(\mathbf{x}_t|\mathbf{x}_{t+1}),$$

where

$$\tilde{\pi}^{\mathbf{y}}_{t|t+1}(\mathbf{x}_t|\mathbf{x}_{t+1}) := \int q_{t|0}(\mathbf{x}_t|\mathbf{x}_0) \pi^{\mathbf{y}}_{0|t+1}(\mathbf{x}_0|\mathbf{x}_{t+1}) \, \mathrm{d}\mathbf{x}_0 \tag{19}$$

and $\pi^{\mathbf{y}}_{0|t+1}(\mathbf{x}_0|\mathbf{x}_{t+1}) = \pi^{\mathbf{y}}_0(\mathbf{x}_0) q_{t+1|0}(\mathbf{x}_{t+1}|\mathbf{x}_0) / \pi^{\mathbf{y}}_{t+1}(\mathbf{x}_{t+1})$. From this definition it follows that

$$\pi^{\mathbf{y}}_t(\mathbf{x}_t) = \int \tilde{\pi}^{\mathbf{y}}_{t|t+1}(\mathbf{x}_t|\mathbf{x}_{t+1}) \pi^{\mathbf{y}}_{t+1}(\mathbf{x}_{t+1}) \, \mathrm{d}\mathbf{x}_{t+1} \,,$$

and hence that the marginals of the joint distribution $\tilde{\pi}^{\mathbf{y}}_{0:T}$ are $(\pi^{\mathbf{y}}_t)_{t=0}^T$. The canonical backward transition $\pi^{\mathbf{y}}_{t|t+1}(\mathbf{x}_t|\mathbf{x}_{t+1}) \propto \pi^{\mathbf{y}}_t(\mathbf{x}_t) q_{t+1|t}(\mathbf{x}_{t+1}|\mathbf{x}_t)$ has the alternative form

$$\pi^{\mathbf{y}}_{t|t+1}(\mathbf{x}_t|\mathbf{x}_{t+1}) = \int q_{t|0,t+1}(\mathbf{x}_t|\mathbf{x}_0, \mathbf{x}_{t+1}) \pi^{\mathbf{y}}_{0|t+1}(\mathbf{x}_0|\mathbf{x}_{t+1}) \, \mathrm{d}\mathbf{x}_0 \,,$$

which differs from (19) in the use of the bridge transition $q_{t|0,t+1}$ instead of the forward transition $q_{t|0}$.

In order to sample from $\tilde{\pi}_{t|t+1}(\cdot|\mathbf{x}_{t+1})$, one needs to first sample $X_0 \sim \pi^{\mathbf{y}}_{0|t+1}(\cdot|\mathbf{x}_{t+1})$ and then $X_t \sim q_{t|0}(\cdot|X_0)$. DAPS performs the former step using Langevin dynamics on an approximation of $\pi^{\mathbf{y}}_{0|t+1}(\cdot|\mathbf{x}_{t+1})$. More specifically, the authors use the approximation

$$\pi^{\mathbf{y}}_{0|t+1}(\mathbf{x}_0|\mathbf{x}_{t+1}) \approx \frac{g_0(\mathbf{y}|\mathbf{x}_0) \mathrm{N}(\mathbf{x}_0; D_{t+1}(\mathbf{x}_{t+1}), r_{t+1}^2 \mathbf{I}_d)}{\int g_0(\mathbf{y}|\mathbf{x}_0') \mathrm{N}(\mathbf{x}_0'; D_{t+1}(\mathbf{x}_{t+1}), r_{t+1}^2 \mathbf{I}_d) \, \mathrm{d}\mathbf{x}_0'} \,,$$

where r_{t+1}^2 is a hyperparameter. This approximation follows by noting that $\pi^{\mathbf{y}}_{0|t+1}(\mathbf{x}_0|\mathbf{x}_{t+1}) \propto g_0(\mathbf{y}|\mathbf{x}_0) p_{0|t+1}(\mathbf{x}_0|\mathbf{x}_{t+1})$ and using the Gaussian approximation of $p_{0|t+1}(\cdot|\mathbf{x}_{t+1})$ proposed by Song et al. (2023a). The Langevin step is initialized with a sample obtained by discretizing the probability flow ODE (Song et al., 2021) between $t+1$ and 0.

Both MGDM and DAPS perform full noising and denoising steps and operate in a similar manner in this respect (with the distinction that we use DDPM instead of the probability flow ODE). The first fundamental difference is that we sample, conditionally on $\mathbf{y}$ and at a random timestep s, by drawing from $\overline{\pi}^{\mathbf{y}}_{s|0,t}(\cdot|\mathbf{x}_0, \mathbf{x}_t) \propto \hat{g}_s(\mathbf{y}|\mathbf{x}_s) q_{s|0,t}(\mathbf{x}_s|\mathbf{x}_0, \mathbf{x}_t)$. Unlike DAPS, our method does not rely on a density approximation prior to applying an approximate sampler. The second main difference is the fact that within each denoising step, we can increase the number of Gibbs iterations to improve the overall performance, as demonstrated in Figure 3. This is on top of the number of gradient steps that we use to fit the variational approximation and which enhance the performance when we increase them.

On the other hand, DAPS does not require the computation of vector-Jacobian products of the denoiser and is thus more efficient in terms of memory. However it requires many calls to the likelihood function, which can substantially increase the runtime if it is expensive to evaluate. For example, with a latent diffusion model, the runtime of DAPS is at least three times larger than that of MGDM, RESAMPLE, and PSLD.

Comparison with Moufad et al. (2024) The more recent MGPS algorithm of Moufad et al. (2024) is also related to MGDM. Similarly to DAPS (Zhang et al., 2024), their methodology relies on sampling approximately from the posterior transition $\pi^{\mathbf{y}}_{t|t+1}(\cdot|\mathbf{x}_{t+1})$ at each step of the backward denoising process. It builds on the following decomposition, which holds for all $s \in [\![0, t-1]\!]$:

$$\pi^{\mathbf{y}}_{t|t+1}(\mathbf{x}_t|\mathbf{x}_{t+1}) = \int q_{t|s,t+1}(\mathbf{x}_t|\mathbf{x}_s, \mathbf{x}_{t+1}) \pi^{\mathbf{y}}_{s|t+1}(\mathbf{x}_s|\mathbf{x}_{t+1}) \, \mathrm{d}\mathbf{x}_s \, .$$

One step of MGPS proceeds by first sampling from an approximation of the posterior transition $\pi^{\mathbf{y}}_{s|t+1}(\cdot|\mathbf{x}_{t+1})$ and then sampling from the bridge transition to return back to time t. The approximation of the posterior transition used in the MGPS is

$$\pi^{\mathbf{y}}_{s|t+1}(\mathbf{x}_s|\mathbf{x}_{t+1}) \approx \frac{\hat{g}^{\theta}_s(\mathbf{y}|\mathbf{x}_s) p^{\theta}_{s|t+1}(\mathbf{x}_s|\mathbf{x}_{t+1})}{\int \hat{g}_s(\mathbf{y}|\mathbf{x}'_s) p^{\theta}_{s|t+1}(\mathbf{x}'_s|\mathbf{x}_{t+1}) \, \mathrm{d}\mathbf{x}'_s} \, . \tag{20}$$

Here one can then choose s to be sufficiently small to enhance the likelihood approximation, while still having an accurate Gaussian approximation of the transition $p_{s|t+1}(\cdot|\mathbf{x}_{t+1})$. The authors demonstrate, using a solvable toy example, that this trade-off indeed exists; see (Moufad et al., 2024, Example 3.2). The approximate sampling step is then performed by fitting a Gaussian variational approximation to the approximation on the r.h.s. of (20), similarly to what we do in Algorithm 2.

Both MGDM and MGPS leverage the same idea of using, at step time t, likelihood approximations at earlier steps $s < t$. While in MGPS the time s is set deterministically as a function of t, we sample it randomly. However, the main difference lies in the step where we sample conditionally on the observation $\mathbf{y}$. Once the index s is sampled we proceed with R rounds of reverse KL minimization w.r.t. to a *different* target distribution. Indeed, following Algorithm 2, in the first round we seek to fit a distribution with density proportional to $\mathbf{x}_s \mapsto \hat{g}^{\theta}_s(\mathbf{y}|\mathbf{x}_s) q_{s|0,t}(\mathbf{x}_s|\hat{X}^*_0, \hat{X}_t)$, where $\hat{X}^*_0$ is an output from the previous step of the algorithm. At step r, we fit $\mathbf{x}_s \mapsto \hat{g}^{\theta}_s(\mathbf{y}|\mathbf{x}_s) q_{s|0,t}(\mathbf{x}_s|\hat{X}^{r-1}_0, \hat{X}^{r-1}_t)$, where $\hat{X}^{r-1}_0$ is sampled using a few DDPM steps starting from $\hat{X}^{r-1}_s$ at time s and $\hat{X}^{r-1}_t \sim q_{t|s}(\cdot|\hat{X}^{r-1}_s)$. On the other hand, MGPS fits in a single round the distribution with density proportional to $\mathbf{x}_s \mapsto \hat{g}^{\theta}_s(\mathbf{y}|\mathbf{x}_s) q_{s|0,t+1}(\mathbf{x}_s|D^{\theta}_{t+1}(\hat{X}_{t+1}), \hat{X}_{t+1})$, where $\hat{X}_{t+1}$ is the output of the previous step. Finally, the authors report that the performance of MGPS improves when the number of gradient steps is increased. In our case, we have two axes, Gibbs iterations R and gradient steps, that allow us to improve the performance when more compute is available.

B Experiments details

B.1 Choice of weight sequence

In all our experiments we draw the index s, at time t_i, from Uniform$[\![\tau, t_{i-1}]\!]$ with $\tau = 10$. The main motivation behind setting $\tau = 10$ and not $\tau = 1$, which is more natural, is that we have found that otherwise it may lead to instabilities. This arises typically when an index s is sampled very close to 0 when $t \approx T$. To avoid such behavior we use a smaller learning rate in Algorithm 3 for the first few iterations and set $\tau > 1$. For the last 25% diffusion steps we set s deterministically to t_{i-1} as we have found that this slightly improves the reconstructions quality. We also ramp up the number of gradient steps as this significantly sharpens the details in the images.

While it is more intuitive to sample s close to 0 as it provides the best approximation error for the likelihood, we have found that this can significantly slow the mixing of the Gibbs sampler in very large dimensions and provides rather poor results when used with a small number of Gibbs steps. Practically speaking, significant artifacts arise during the initial iterations of the algorithm due to the optimization procedure, and they tend to persist in subsequent iterations when s is sampled close to 0. To see why this is the case consider the following empirical discussion on a simplified scenario. We write $\mathbf{x} = [\bar{\mathbf{x}}, \underline{\mathbf{x}}]$ where $\bar{\mathbf{x}} \in \mathbb{R}^{d_{\mathbf{y}}}$ and $\underline{\mathbf{x}} \in \mathbb{R}^{d-d_{\mathbf{y}}}$. We assume that $g_0(\mathbf{y}|\mathbf{x}) = \mathrm{N}(\mathbf{y}; \bar{\mathbf{x}}, \sigma^2_{\mathbf{y}} \mathbf{I}_{d_{\mathbf{y}}})$, *i.e.*, we observe only the first $d_{\mathbf{y}}$ coordinates of the hidden state. Since s is sampled near 0 we may assume that $\hat{g}_s(\mathbf{y}|\cdot) = g_0(\mathbf{y}|\cdot)$. Then, sampling $Z \sim \bar{\pi}^{\mathbf{y}}_{s|0,t}(\cdot|\mathbf{x}_0, \mathbf{x}_t)$ is equivalent to sampling

$$\begin{aligned}
\bar{Z} &\sim \mathcal{N}\left(\frac{\sigma^2_{s|0,t}}{\sigma^2_{\mathbf{y}} + \sigma^2_{s|0,t}} \mathbf{y} + \frac{\sigma^2_{\mathbf{y}}}{\sigma^2_{\mathbf{y}} + \sigma^2_{s|0,t}} \left[\gamma_{t|s} \alpha_{s|0} \bar{\mathbf{x}}_0 + (1 - \gamma_{t|s}) \alpha^{-1}_{t|s} \bar{\mathbf{x}}_t \right], \frac{\sigma^2_{\mathbf{y}} \sigma^2_{s|0,t}}{\sigma^2_{\mathbf{y}} + \sigma^2_{s|0,t}} \mathbf{I}_{d_{\mathbf{y}}} \right) , \\
\underline{Z} &\sim \mathcal{N}(\gamma_{t|s} \alpha_{s|0} \underline{\mathbf{x}}_0 + (1 - \gamma_{t|s}) \alpha^{-1}_{t|s} \underline{\mathbf{x}}_t, \sigma^2_{s|0,t} \mathbf{I}_{d-d_{\mathbf{y}}}) \, ,
\end{aligned}$$

Table 4: LPIPS on the FFHQ dataset for the two time-sampling distributions given in (21) and (22). We use $R = 4$ Gibbs steps for the phase retrieval task.

Distribution	Phase retrieval ($R = 4$)	JPEG2	Gaussian deblurring	Motion deblurring
μ_t^*	**0.10**	**0.14**	**0.12**	**0.09**
μ_t^0	0.53	0.19	0.16	0.19

Figure 4: Evolution of the running state $\hat{X}_0^*$ in Algorithm 2 for the two time-sampling distributions given in (21) and (22).

setting $Z = [\bar{Z}, \underline{Z}]$ and then concatenating both vectors. It is thus seen that the observed part of the state is updated with the observation whereas the bottom part is simply drawn from the prior. Moreover, if $\sigma^2_{s|0,t} \approx 0$ then $\gamma_{t|s}\alpha_{s|0} \approx 1$ and $\underline{Z}$ is almost the same as $\mathbf{x}_0$. In Algorithm 2, once we have sampled $\hat{X}_s \sim \overline{\pi}^{\mathbf{y}}_{s|0,t}(\cdot|\hat{X}_0, \hat{X}_t)$, we first denoise it to obtain the new $\hat{X}_0$ and then noise it to obtain the new $\hat{X}_t$. As s is sampled near 0, the denoising step will merely modify $\hat{X}_s$ whereas the noising step will add significant noise to $\hat{X}_s$ and may help with removing the artifacts. This noised sampled has however only a small impact on the next samples $\hat{X}_0, \hat{X}_s$ since $(1 - \gamma_{t|s})\alpha_{t|s}^{-1} \approx 0$. In short, the first $d_{\mathbf{y}}$ coordinates of the running state $\hat{X}_0^*$ will be quickly replaced by the observation whereas the last $d - d_{\mathbf{y}}$ coordinates will be stuck at their initialization and will evolve only by a small amount throughout the iterations of the algorithm. We illustrate this situation on a concrete example in Figure 4 where we consider a half mask inpainting task. The first and second rows show the evolution of the running state $\hat{X}_0^*$ with the time-sampling distributions

$$\mu_i^* = \begin{cases} \text{Uniform}[\![\tau, t_{i-1}]\!] & \text{if} \quad i > \lfloor K/4 \rfloor \\ t_{i-1} & \text{else} \end{cases}, \tag{21}$$

$$\mu_i^0 = \text{Uniform}[\![1, \lfloor t_i/5 \rfloor]\!], \tag{22}$$

i.e., the time-sampling distribution we use in all our experiments, where K is the number of diffusion steps, and the one that we use to sample only close to 0, respectively. In Table 4 we compute the LPIPS for both distributions on a subset of the tasks we consider in the main paper. It is clear that μ_i^* outperforms μ_i^0, even when we increase the number of Gibbs steps (see phase retrieval task).

B.2 Hyperparameters setup of MGDM

The details about the hyperparameters of MGDM are reported in Table 5. We adjust the optimization of the Gaussian Variational approximation in Algorithm 3 during the first and last diffusion steps. We ramp up the number of gradient steps during the final diffusion steps. This allows us to substantially improve the fine grained details of the reconstructions. Similarly, we reduce the learning rate in the early step to alleviate potential instabilities.

B.3 Audio source separation

In our experiment, the diffusion model employed provided by (Mariani et al., 2023) is trained on the slakh2100 training dataset[1], using only the four abundant instruments (bass, drums, guitar and piano) downsampled to 22 kHz. The denoiser network is based on a non-latent, time-domain unconditional variant of (Schneider et al., 2023).

Its architecture follows a U-Net design, comprising an encoder, bottleneck, and decoder. The encoder consists of six layers

[1] http://www.slakh.com/

Table 5: The hyperparameters used in MGDM for the considered datasets. The index i of the timesteps $\{t_i\}_{i=K}^{0}$ is taken in reverse order. The symbol # stands for *"number of"*.

	# Gibbs repetitions R	# Diffusion steps K	# Denoising steps M	Time-sampling distribution	Learning rate η	# Gradient steps G
`FFHQ`	$R = 1$	$K = 100$	$M = 20$	μ_i^* as in (21)	$\eta = \begin{cases} 0.01 & \text{if } i \geq \lfloor 3K/4 \rfloor \\ 0.03 & \text{otherwise} \end{cases}$	$G = \begin{cases} 20 & \text{if } i \leq \lfloor K/4 \rfloor \\ 5 & \text{otherwise} \end{cases}$
`FFHQ` LDM	$R = 1$	$K = 100$	$M = 20$	μ_i^* as in (21)	$\eta = \begin{cases} 0.01 & \text{if } i \geq \lfloor 3K/4 \rfloor \\ 0.03 & \text{otherwise} \end{cases}$	$G = \begin{cases} 20 & \text{if } i \leq \lfloor K/4 \rfloor \\ 20 & \text{if } i \mod 10 = 0 \\ 3 & \text{otherwise} \end{cases}$
`ImageNet`	$R = 1$	$K = 100$	$M = 20$	μ_i^* as in (21)	$\eta = \begin{cases} 0.01 & \text{if } i \geq \lfloor 3K/4 \rfloor \\ 0.03 & \text{otherwise} \end{cases}$	$G = \begin{cases} 20 & \text{if } i \leq \lfloor K/4 \rfloor \\ 5 & \text{otherwise} \end{cases}$
Audio-source separation	$R = 6$	$K = 20$	$M = 15$	μ_i^* as in (21)	$\eta = 0.005$	$G = \begin{cases} 20 & \text{if } i \leq \lfloor K/4 \rfloor \\ 3 & \text{otherwise} \end{cases}$
Audio-source separation (Best result in Table 3)	$R = 1$	$K = 20$	$M = 15$	μ_i^* as in (21)	$\eta = 0.005$	$G = 90$

with channel numbers $[256, 512, 1024, 1024, 1024, 1024]$, where each layer includes two convolutional ResNet blocks, and multihead attention is applied in the last three layers. The decoder mirrors the encoder structure in reverse. The bottleneck contains a ResNet block, followed by a self-attention mechanism, and then another ResNet block. Training is performed on the four stacked instruments using the publicly available trainer from repository[2].

B.4 Implementation of the competitors

In this section, we provide implementation details of the competitors. We adopt the hyperparameters recommended by the authors tune them on each dataset if they are not provided. We emphasize that we use the total number of diffusion steps available (1000 steps) for DPS, PGDM, DDNM, DiffPIR, and PSLD. For the other algorithms, we tuned the compute time by increasing Langevin/denoising/optimization steps until performance plateaued. The complete set of hyperparameters and their values for both image experiments and audio-sound separation can be found in the supplementary material under the folders `configs/experiments/sampler` and `configs/exp_sound/sampler`.

DPS. We implemented Chung et al. (2023, Algorithm 1) and selected the hyperparameters of each considered task based on Chung et al. (2023, App. D). We tuned the algorithm for the other tasks, namely, we use $\gamma = 0.2$ for JPEG 2%, $\gamma = 0.07$ for High Dynamic Range tasks, and $\gamma = 1$ for audio-source separation.

DiffPIR. We implemented Zhu et al. (2023, Algorithm 1) to make it compatible with our existing code base. We adopt the hyperparameters recommended in the official, released version[3]. We followed the guidelines in (Zhu et al., 2023, Eqn. (13)) to extend the algorithm to nonlinear problems. However, we noticed that the algorithm diverges in these cases and we could not follow up as the paper and the released code lack examples of nonlinear problems. Zhu et al. (2023) provides an FFT-based solution for the motion blur tasks which is only valid in the case of circular convolution. Hence, and since we adapted the experimental setup of Chung et al. (2023), we do not run the algorithm on motion blur task as it uses convolution with reflect padding. For audio-source separation, we found that $\lambda = \mu = 1$ works best.

DDNM. We adapted the implementation provided in the released code[4]. Namely, the authors provide classes, in the module `functions/svd_operators.py` that implement the logic of the algorithm on each degradation operator separately. The adaptation includes factorizing these classes to a single class to support all SVD linear degradation operators. On the other hand, we notice DDNM is unstable for operators whose SVD decomposition is prone to numerical errors, such as Gaussian Blur with wide convolution kernel. This results from the algorithm using the pseudo-inverse of the operator.

RedDiff. We used the implementation of RedDiff available in the released code[5]. For linear problems, we use the pseudo-inverse of the observation as an initialization of the variational optimization problem. On nonlinear problems, for which the pseudo-inverse of the observation is not available, we initialized the optimization with a sample from the standard Gaussian distribution.

[2] https://github.com/archinetai/audio-diffusion-pytorch-trainer
[3] https://github.com/yuanzhi-zhu/DiffPIR
[4] https://github.com/wyhuai/DDNM
[5] https://github.com/NVlabs/RED-diff

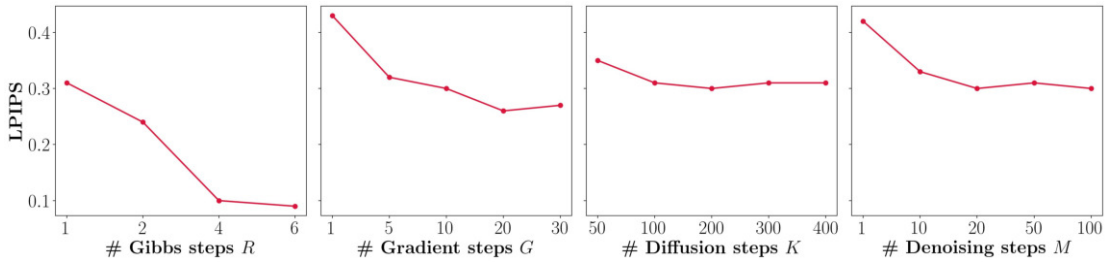

Figure 5: Effect of individually increasing each parameter on the performance of our algorithm for the phase retrieval task. In each case, we vary one parameter while keeping the others fixed to the values specified in the original manuscript.

PGDM. We opted for the implementation available in the REDDIFF́S repository as some the authors are co-authors of PGDMas well. Notably, the implementation introduces a subtle deviation from Song et al. (2023a, Algorithm 1): in the algorithm's final step, the guidance term g is scaled by α_t ($\sqrt{\alpha_t}$ in their notation) whereas the implementation scales it by $\alpha_{t-1}\alpha_t$. This adjustment improves the algorithm for most tasks except for JPEG dequantization. We found that the original scaling by α_t is better in this case.

PSLD. We implemented the PSLD algorithm provided in Rout et al. (2024, Algorithm 2) and referred to the publicly available implementation[6] to set the hyperparameters of the algorithm for the different tasks.

ReSample. We modified the original code[7] provided by the authors to make its hyperparameters directly adjustable, namely, the tolerance ε and the maximum number of iterations N for solving the optimization problems related to hard data consistency, and the scaling factor for the variance of the stochastic resampling distribution γ. We found the algorithm to be sensitive to ε and that setting it to the noise level of the inverse problem yields the best reconstructions across tasks and noise levels. On the other hand, we noticed that γ has less impact on the quality of the reconstructions. Finally, we set a threshold $N = 200$ on the maximum number of gradient iterations to make the algorithm less computationally intensive.

DAPS. We have the official codebase [8]. We referred to Zhang et al. (2024, Table. 7) to set the hyperparameters. For audio-source separation, we set $\sigma_{\max}$ and $\sigma_{\min}$ to match those of the sound model and adapted the Langevin stepsize `lr` and the standard deviation `tau` to the audio-separation task.

PNP-DM. We adapted the implementation provided in the released code[9]. Specifically, we exposed the coupling parameter ρ including its initial value, minimum value, and decay rate, as well as the number of Langevin steps and its step size. The hyperparameters were set based on Wu et al. (2024, Table 3 and Table 4). For inpainting tasks, while it is theoretically possible to perform the likelihood steps using Gaussian conjugacy (Wu et al., 2024, Sec. 3.1), we found that using Langevin produced better results in practice. For example, the reconstructions in the left figure of Figure 8 are obtained by sampling exactly from the posterior whereas on the r.h.s. we use Langevin dynamics. Although the audio separation task is linear and hence the likelihood steps can be implemented exactly, we encountered similar challenges as in inpainting and therefore we used Langevin here as well.

B.5 Experiments reproducibility

Our code will be made available upon acceptange of the paper. In the anonymous codebase provided as companion of the paper we use $\sqrt{\alpha_t}$ instead of α_t to match the conventions of existing codebases. All experiments were conducted on Nvidia Tesla V100 SXM2 GPUs. For the image experiments, we used 300 images from the validation sets of `FFHQ` and `ImageNet` 256×256 that we numbered from 0 to 299. The image number was used to seed the randomness of the experiments on that image. For the audio source separation experiments, the `slakh2100` test dataset has tracks named following the pattern `Track0XXXX`, where `X` represents a digit in $0 - 9$. The number `XXXX` was used as the seed for the

[6] https://github.com/LituRout/PSLD
[7] https://github.com/soominkwon/resample
[8] https://github.com/zhangbingliang2019/DAPS
[9] https://github.com/zihuiwu/PnP-DM-public/

experiments conducted on each track.

B.6 Extended results

We present the complete results with FID, LPIPS, PSNR, and SSIM metrics for the image inverse problems experiment in: Table 6 for `FFHQ` pixel-space, Table 7 for `ImageNet`, and in Table 8 for `FFHQ` LDM. In Figure 5 we provided an extension of the ablation in Figure 3. Similarly, the complete results for the audio source separation experiments that include all competitors are provided in Table 9.

From Table 6 and Table 7, one can note that DDNM, DIFFPIR and DAPS score better in PSNR and SSIM compared to MGDM but score lower in LPIPS. For most of the tasks we considered, one does not expect to recover an image very close to the reference and thus, metrics that perform pixel-wise comparisons are less relevant and favor images that are overly smooth. We provide evidence for this in the gallery of images below where we compare qualitatively the outputs of our algorithm with those of the competitors. It can be seen that our method provides reconstructions with fine-grained details that are more coherent with the reference image. Note for example that DDNM, DIFFPIR and DAPS outperform MGDM in terms of PSNR and SSIM on the half mask task on `ImageNet` while failing to reconstruct the missing r.h.s. of the images.

Table 6: FID and mean LPIPS/PSNR/SSIM metrics along side 95%-confidence interval on FFHQ 256 × 256 dataset with $\sigma_{\mathbf{y}} = 0.05$.

Task	MGDM	DPS	PGDM	DDNM	DIFFPIR	REDDIFF	DAPS	PNP-DM
				LPIPS ↓				
SR (×4)	0.09 ±0.00	0.09 ±0.00	0.30 ±0.01	0.15 ±0.00	0.10 ±0.00	0.39 ±0.01	0.16 ±0.01	0.10 ±0.00
SR (×16)	0.24 ±0.01	0.23 ±0.01	0.42 ±0.01	0.33 ±0.01	0.23 ±0.01	0.55 ±0.01	0.40 ±0.01	0.27 ±0.01
Box inpainting	0.10 ±0.00	0.17 ±0.01	0.17 ±0.00	0.12 ±0.00	0.14 ±0.00	0.19 ±0.01	0.13 ±0.00	0.18 ±0.01
Half mask	0.20 ±0.01	0.23 ±0.01	0.24 ±0.01	0.23 ±0.01	0.25 ±0.01	0.28 ±0.01	0.23 ±0.01	0.32 ±0.01
Gaussian Deblur	0.12 ±0.00	0.17 ±0.01	0.87 ±0.02	0.20 ±0.01	0.12 ±0.00	0.24 ±0.01	0.24 ±0.01	0.14 ±0.01
Motion Deblur	0.09 ±0.00	0.17 ±0.01	–	–	–	0.22 ±0.01	0.19 ±0.01	0.21 ±0.01
JPEG (QF = 2)	0.14 ±0.01	0.34 ±0.03	1.12 ±0.01	–	–	0.32 ±0.01	0.22 ±0.01	0.29 ±0.01
Phase retrieval	0.11 ±0.02	0.40 ±0.02	–	–	–	0.26 ±0.02	0.14 ±0.01	0.34 ±0.02
Nonlinear deblur	0.27 ±0.01	0.51 ±0.04	–	–	–	0.68 ±0.02	0.28 ±0.01	0.31 ±0.01
High dynamic range	0.12 ±0.01	0.40 ±0.06	–	–	–	0.20 ±0.03	0.10 ±0.01	0.19 ±0.01
				FID ↓				
SR (×4)	52.28	55.02	95.22	69.75	54.26	115.01	65.68	55.32
SR (×16)	62.12	61.24	161.64	104.10	58.12	141.76	117.83	63.49
Box inpainting	52.08	89.13	60.99	64.06	68.34	64.24	60.29	75.67
Half mask	58.97	79.31	61.48	64.96	71.86	63.92	63.88	84.16
Gaussian Deblur	55.37	60.55	295.19	73.92	54.29	71.82	76.98	56.69
Motion Deblur	52.65	61.80	–	–	–	102.88	70.76	86.58
JPEG (QF = 2)	56.40	107.54	305.00	–	–	132.81	79.09	108.89
Phase retrieval	83.58	146.79	–	–	–	123.91	58.53	172.20
Nonlinear deblur	87.89	198.33	–	–	–	113.38	92.12	91.30
High dynamic range	54.56	165.70	–	–	–	75.63	56.19	77.29
				PSNR ↑				
SR (×4)	27.66 ±0.22	28.05 ±0.24	24.57 ±0.14	29.45 ±0.23	27.72 ±0.22	26.75 ±0.13	28.44 ±0.21	27.44 ±0.20
SR (×16)	21.01 ±0.20	20.71 ±0.21	18.51 ±0.14	22.32 ±0.21	20.96 ±0.20	21.46 ±0.17	19.75 ±0.17	20.88 ±0.19
Box inpainting	22.38 ±0.27	18.81 ±0.28	21.05 ±0.25	22.34 ±0.31	22.39 ±0.32	21.46 ±0.28	22.06 ±0.30	20.42 ±0.28
Half mask	15.39 ±0.27	15.03 ±0.27	15.29 ±0.28	16.38 ±0.35	16.04 ±0.36	15.68 ±0.34	16.25 ±0.30	14.35 ±0.30
Gaussian Deblur	25.64 ±0.24	24.03 ±0.22	13.34 ±0.10	26.62 ±0.23	25.78 ±0.23	26.68 ±0.22	26.12 ±0.23	25.89 ±0.21
Motion Deblur	27.82 ±0.20	24.12 ±0.21	–	–	–	27.48 ±0.13	27.07 ±0.21	24.91 ±0.24
JPEG (QF = 2)	25.57 ±0.19	19.56 ±0.60	12.57 ±0.10	–	–	24.53 ±0.13	25.72 ±0.18	22.42 ±0.18
Phase retrieval	27.55 ±0.65	16.56 ±0.63	–	–	–	24.58 ±0.67	27.84 ±0.46	21.63 ±0.70
Nonlinear deblur	23.55 ±0.27	16.08 ±0.87	–	–	–	21.94 ±0.25	24.56 ±0.36	24.08 ±0.32
High dynamic range	24.79 ±0.42	18.71 ±0.32	–	–	–	21.69 ±0.20	26.60 ±0.38	21.59 ±0.22
				SSIM ↑				
SR (×4)	0.80 ±0.01	0.81 ±0.01	0.56 ±0.00	0.85 ±0.00	0.78 ±0.01	0.68 ±0.00	0.81 ±0.00	0.77 ±0.01
SR (×16)	0.61 ±0.01	0.58 ±0.01	0.42 ±0.01	0.67 ±0.01	0.59 ±0.01	0.60 ±0.01	0.58 ±0.01	0.57 ±0.01
Box inpainting	0.80 ±0.00	0.77 ±0.00	0.70 ±0.00	0.83 ±0.00	0.82 ±0.00	0.70 ±0.00	0.80 ±0.00	0.75 ±0.00
Half mask	0.67 ±0.01	0.67 ±0.01	0.59 ±0.01	0.74 ±0.01	0.72 ±0.01	0.63 ±0.01	0.71 ±0.01	0.65 ±0.01
Gaussian Deblur	0.73 ±0.01	0.68 ±0.01	0.14 ±0.01	0.77 ±0.01	0.72 ±0.01	0.76 ±0.01	0.75 ±0.01	0.72 ±0.01
Motion Deblur	0.80 ±0.01	0.70 ±0.01	–	–	–	0.71 ±0.01	0.78 ±0.01	0.75 ±0.01
JPEG (QF = 2)	0.74 ±0.01	0.56 ±0.03	0.10 ±0.01	–	–	0.71 ±0.01	0.76 ±0.01	0.70 ±0.01
Phase retrieval	0.78 ±0.02	0.49 ±0.02	–	–	–	0.61 ±0.02	0.81 ±0.01	0.57 ±0.02
Nonlinear deblur	0.67 ±0.01	0.44 ±0.03	–	–	–	0.42 ±0.01	0.71 ±0.01	0.70 ±0.01
High dynamic range	0.76 ±0.02	0.55 ±0.06	–	–	–	0.72 ±0.04	0.85 ±0.01	0.69 ±0.01

Table 7: FID and mean LPIPS/PSNR/SSIM metrics along side 95%-confidence interval on `ImageNet` 256×256 dataset with $\sigma_{\mathbf{y}} = 0.05$.

Task	MGDM	DPS	PGDM	DDNM	DIFFPIR	REDDIFF	DAPS	PNP-DM
				LPIPS ↓				
SR (×4)	0.26 ±0.01	0.25 ±0.01	0.56 ±0.02	0.34 ±0.02	0.31 ±0.01	0.57 ±0.02	0.37 ±0.02	0.66 ±0.02
SR (×16)	0.55 ±0.02	0.44 ±0.01	0.62 ±0.01	0.71 ±0.02	0.50 ±0.02	0.85 ±0.02	0.75 ±0.02	1.03 ±0.02
Box inpainting	0.23 ±0.01	0.35 ±0.01	0.29 ±0.01	0.28 ±0.01	0.30 ±0.01	0.36 ±0.01	0.30 ±0.01	0.42 ±0.01
Half mask	0.31 ±0.01	0.40 ±0.03	0.34 ±0.03	0.38 ±0.03	0.40 ±0.03	0.46 ±0.03	0.40 ±0.01	0.54 ±0.01
Gaussian Deblur	0.30 ±0.01	0.37 ±0.02	1.00 ±0.01	0.45 ±0.02	0.30 ±0.01	0.53 ±0.02	0.59 ±0.02	0.76 ±0.02
Motion Deblur	0.22 ±0.01	0.40 ±0.04	–	–	–	0.39 ±0.04	0.42 ±0.02	0.52 ±0.02
JPEG (QF = 2)	0.38 ±0.02	0.60 ±0.06	1.32 ±0.01	–	–	0.49 ±0.04	0.45 ±0.02	0.56 ±0.02
Phase retrieval	0.55 ±0.02	0.62 ±0.02	–	–	–	0.61 ±0.02	0.50 ±0.02	0.66 ±0.01
Nonlinear deblur	0.41 ±0.01	0.82 ±0.05	–	–	–	0.66 ±0.05	0.41 ±0.02	0.49 ±0.02
High dynamic range	0.21 ±0.02	0.84 ±0.05	–	–	–	0.19 ±0.04	0.14 ±0.01	0.31 ±0.02
				FID ↓				
SR (×4)	97.07	94.77	131.97	109.83	104.44	139.67	113.54	160.78
SR (×16)	142.09	124.72	181.29	223.92	135.37	228.78	224.69	269.45
Box inpainting	113.49	173.77	123.99	133.41	145.42	157.76	138.02	178.59
Half mask	106.39	144.08	112.73	111.63	118.21	133.44	114.60	147.14
Gaussian Deblur	105.92	110.73	287.63	140.06	106.64	148.75	158.50	198.25
Motion Deblur	94.74	114.03	–	–	–	144.47	133.07	166.30
JPEG (QF = 2)	117.15	186.13	340.95	–	–	151.58	145.65	178.50
Phase retrieval	170.71	148.61	–	–	–	215.18	148.06	201.66
Nonlinear deblur	175.50	298.36	–	–	–	171.00	175.44	172.58
High dynamic range	104.01	353.01	–	–	–	99.81	94.38	126.53
				PSNR ↑				
SR (×4)	23.88 ±0.44	24.37 ±0.49	18.45 ±0.26	24.99 ±0.50	23.43 ±0.42	23.33 ±0.35	24.38 ±0.46	16.40 ±0.24
SR (×16)	18.12 ±0.31	17.66 ±0.39	15.27 ±0.23	19.93 ±0.40	18.40 ±0.37	19.06 ±0.33	18.18 ±0.32	14.00 ±0.17
Box inpainting	16.82 ±0.33	13.92 ±0.30	16.73 ±0.31	19.18 ±0.44	19.05 ±0.47	18.21 ±0.40	19.11 ±0.40	18.03 ±0.36
Half mask	13.77 ±0.29	12.15 ±0.19	14.05 ±0.19	15.97 ±0.21	15.64 ±0.22	14.84 ±0.20	16.00 ±0.37	14.88 ±0.26
Gaussian Deblur	21.57 ±0.43	20.65 ±0.43	9.92 ±0.08	22.89 ±0.47	21.80 ±0.44	22.72 ±0.45	22.41 ±0.45	15.85 ±0.22
Motion Deblur	24.46 ±0.42	21.38 ±0.21	–	–	–	24.06 ±0.19	23.64 ±0.44	22.47 ±0.40
JPEG (QF = 2)	21.42 ±0.32	16.33 ±0.27	5.27 ±0.04	–	–	22.07 ±0.18	22.68 ±0.36	20.74 ±0.30
Phase retrieval	16.01 ±0.71	14.12 ±0.49	–	–	–	15.41 ±0.59	18.44 ±0.72	15.02 ±0.50
Nonlinear deblur	21.96 ±0.39	10.13 ±0.28	–	–	–	20.57 ±0.18	22.68 ±0.44	22.20 ±0.43
High dynamic range	22.90 ±0.57	9.56 ±0.26	–	–	–	22.12 ±0.23	24.69 ±0.49	22.23 ±0.46
				SSIM ↑				
SR (×4)	0.65 ±0.02	0.68 ±0.02	0.30 ±0.01	0.71 ±0.02	0.60 ±0.01	0.57 ±0.01	0.66 ±0.02	0.25 ±0.01
SR (×16)	0.31 ±0.01	0.39 ±0.02	0.21 ±0.01	0.49 ±0.02	0.41 ±0.02	0.44 ±0.02	0.44 ±0.02	0.10 ±0.00
Box inpainting	0.71 ±0.01	0.70 ±0.01	0.62 ±0.00	0.77 ±0.01	0.76 ±0.01	0.67 ±0.00	0.74 ±0.01	0.64 ±0.01
Half mask	0.59 ±0.01	0.58 ±0.03	0.52 ±0.02	0.68 ±0.03	0.67 ±0.03	0.59 ±0.03	0.66 ±0.01	0.57 ±0.01
Gaussian Deblur	0.50 ±0.02	0.50 ±0.02	0.08 ±0.00	0.59 ±0.02	0.51 ±0.02	0.57 ±0.02	0.56 ±0.02	0.20 ±0.01
Motion Deblur	0.67 ±0.01	0.55 ±0.05	–	–	–	0.61 ±0.03	0.63 ±0.02	0.57 ±0.02
JPEG (QF = 2)	0.51 ±0.01	0.40 ±0.06	0.02 ±0.00	–	–	0.59 ±0.04	0.62 ±0.02	0.57 ±0.02
Phase retrieval	0.31 ±0.03	0.27 ±0.02	–	–	–	0.25 ±0.02	0.46 ±0.03	0.23 ±0.01
Nonlinear deblur	0.58 ±0.01	0.25 ±0.06	–	–	–	0.41 ±0.04	0.61 ±0.02	0.58 ±0.02
High dynamic range	0.72 ±0.02	0.23 ±0.06	–	–	–	0.72 ±0.04	0.82 ±0.01	0.66 ±0.02

Table 8: FID and mean LPIPS/PSNR/SSIM metrics along side 95%-confidence interval on FFHQ dataset with LDM prior with $\sigma_{\mathbf{y}} = 0.05$.

Task	MGDM	ReSample	PSLD	DAPS	PNP-DM
			LPIPS ↓		
SR (×4)	0.14 ±0.01	0.22 ±0.01	0.21 ±0.01	0.28 ±0.01	0.40 ±0.01
SR (×16)	0.30 ±0.01	0.38 ±0.01	0.36 ±0.01	0.52 ±0.01	0.71 ±0.01
Box inpainting	0.18 ±0.01	0.22 ±0.00	0.27 ±0.01	0.37 ±0.01	0.31 ±0.01
Half mask	0.26 ±0.01	0.30 ±0.03	0.32 ±0.03	0.49 ±0.01	0.44 ±0.01
Gaussian Deblur	0.18 ±0.01	0.16 ±0.01	0.59 ±0.01	0.32 ±0.01	0.32 ±0.01
Motion Deblur	0.22 ±0.01	0.20 ±0.03	0.70 ±0.03	0.36 ±0.01	0.36 ±0.01
JPEG (QF = 2)	0.23 ±0.01	0.26 ±0.03	–	0.32 ±0.01	0.36 ±0.01
Phase retrieval	0.29 ±0.02	0.39 ±0.02	–	0.25 ±0.01	0.50 ±0.02
Nonlinear deblur	0.29 ±0.01	0.33 ±0.04	–	0.37 ±0.01	0.37 ±0.01
High dynamic range	0.16 ±0.01	0.12 ±0.03	–	0.24 ±0.01	0.24 ±0.01
			FID ↓		
SR (×4)	63.91	78.74	78.85	110.21	131.06
SR (×16)	69.26	100.44	92.16	176.67	281.53
Box inpainting	76.22	126.49	83.53	162.11	129.27
Half mask	78.32	91.76	80.00	131.68	116.16
Gaussian Deblur	69.44	69.77	150.43	101.88	103.16
Motion Deblur	74.43	77.97	158.40	120.74	122.12
JPEG (QF = 2)	72.78	90.39	–	123.93	131.08
Phase retrieval	128.57	230.42	–	105.64	169.94
Nonlinear deblur	100.73	99.37	–	150.54	135.81
High dynamic range	75.74	65.79	–	102.54	89.32
			PSNR ↑		
SR (×4)	27.39 ±0.21	25.85 ±0.25	25.80 ±0.33	27.45 ±0.20	23.81 ±0.21
SR (×16)	20.60 ±0.18	20.97 ±0.18	21.42 ±0.19	19.91 ±0.16	17.07 ±0.14
Box inpainting	21.81 ±0.28	18.56 ±0.21	20.01 ±0.28	11.77 ±0.26	19.57 ±0.31
Half mask	15.71 ±0.30	14.89 ±0.17	14.62 ±0.19	9.13 ±0.25	14.15 ±0.28
Gaussian Deblur	26.79 ±0.22	27.28 ±0.22	17.99 ±0.13	26.86 ±0.26	26.11 ±0.19
Motion Deblur	25.27 ±0.20	26.73 ±0.15	17.71 ±0.12	25.37 ±0.21	24.65 ±0.18
JPEG (QF = 2)	24.27 ±0.18	24.77 ±0.15	–	25.22 ±0.18	23.86 ±0.17
Phase retrieval	22.54 ±0.66	20.18 ±0.61	–	27.05 ±0.35	20.03 ±0.61
Nonlinear deblur	23.71 ±0.27	24.10 ±0.19	–	22.03 ±0.23	23.28 ±0.26
High dynamic range	25.59 ±0.32	25.91 ±0.21	–	20.95 ±0.38	20.21 ±0.23
			SSIM ↑		
SR (×4)	0.79 ±0.01	0.68 ±0.01	0.71 ±0.01	0.79 ±0.01	0.70 ±0.01
SR (×16)	0.58 ±0.01	0.56 ±0.01	0.63 ±0.01	0.59 ±0.01	0.52 ±0.01
Box inpainting	0.78 ±0.00	0.75 ±0.00	0.66 ±0.01	0.70 ±0.01	0.73 ±0.01
Half mask	0.69 ±0.01	0.67 ±0.02	0.60 ±0.03	0.55 ±0.01	0.65 ±0.01
Gaussian Deblur	0.77 ±0.01	0.75 ±0.01	0.27 ±0.01	0.78 ±0.01	0.77 ±0.01
Motion Deblur	0.73 ±0.01	0.72 ±0.03	0.24 ±0.02	0.74 ±0.01	0.72 ±0.01
JPEG (QF = 2)	0.71 ±0.01	0.66 ±0.03	–	0.75 ±0.01	0.72 ±0.01
Phase retrieval	0.62 ±0.02	0.49 ±0.02	–	0.79 ±0.01	0.60 ±0.02
Nonlinear deblur	0.69 ±0.01	0.67 ±0.03	–	0.68 ±0.01	0.70 ±0.01
High dynamic range	0.80 ±0.01	0.83 ±0.03	–	0.74 ±0.02	0.73 ±0.01

Table 9: Mean SI-SDR$_{\text{I}}$ along side 95% confidence interval on slakh2100 test dataset. The last row "All" displays the mean over the four stems. Higher metrics are better.

Stems	MGDM	DPS	PGDM	DDNM	DiffPir	RedDiff	DAPS	PNP-DM	MSDM	ISDM	Demucs$_{512}$
Bass	18.49 ±0.92	16.50 ±0.98	16.41 ±1.10	14.94 ±1.57	-2.34 ±0.67	-0.40 ±0.49	11.76 ±1.61	2.90 ±0.79	17.12	19.36	17.16
Drums	18.07 ±0.48	18.29 ±0.62	18.14 ±0.76	19.05 ±0.51	9.47 ±0.60	-0.98 ±0.54	15.62 ±0.42	7.89 ±0.65	18.68	20.90	19.61
Guitar	16.68 ±1.41	9.90 ±1.39	12.84 ±1.99	14.38 ±1.57	-1.01 ±0.53	5.68 ±0.88	11.75 ±1.55	4.51 ±1.19	15.38	14.70	17.82
Piano	16.17 ±1.22	10.41 ±1.14	12.31 ±1.49	11.46 ±1.59	0.97 ±0.95	5.04 ±0.46	9.52 ±1.34	4.09 ±0.64	14.73	14.13	16.32
All	17.35 ±1.01	13.77 ±1.03	14.92 ±1.33	14.96 ±1.31	1.77 ±0.69	2.33 ±0.59	12.16 ±1.23	4.85 ±0.82	16.48	17.27	17.73

B.7 Runtime and memory requirement comparison

We evaluate the runtime and GPU memory consumption for image experiments on the three considered diffusion model priors. Since not all algorithms support every task, we restrict the evaluation to commune tasks. Figure 6 presents the average runtime and GPU memory requirement over both samples and tasks.

MGDM has memory requirements similar to DPSand PGDMin pixel space and aligns closely with other methods in latent space. Importantly, in latent diffusion which is a highly relevant scenario given the prevalence of latent-space models, MGDM is notably faster than all competitors while consistently achieving strong performance across benchmarks. Conversely, when operating directly in pixel space, it exhibits relatively slower runtimes compared to some alternatives. However, this increase in computational overhead is consistently balanced by improved and stable reconstruction quality across all considered tasks as evidenced by the gallery of examples in Appendix B.9. Thus, we position our method as offering a beneficial trade-off, especially in scenarios where quality and consistency of results are paramount.

On the otherhand, we highlight several important points regarding the competitors' runtime: 1) for latent diffusion, DAPS and PNP-DM perform a significant amount of Langevin steps using the gradient of the likelihood. Since the latter involves a vector jacobian product of the decoder, the runtime increases significantly. More generally, when the likelihood function is expensive to evaluate, DAPS and PNP-DM are expected to be much slower than DPS, PGDMand MGDM. For PNP-DM on `FFHQ` we have implemented the likelihood step exactly on the linear tasks. On `ImageNet` however, using Langevin steps provided better results and this explains the significant increase in runtime.

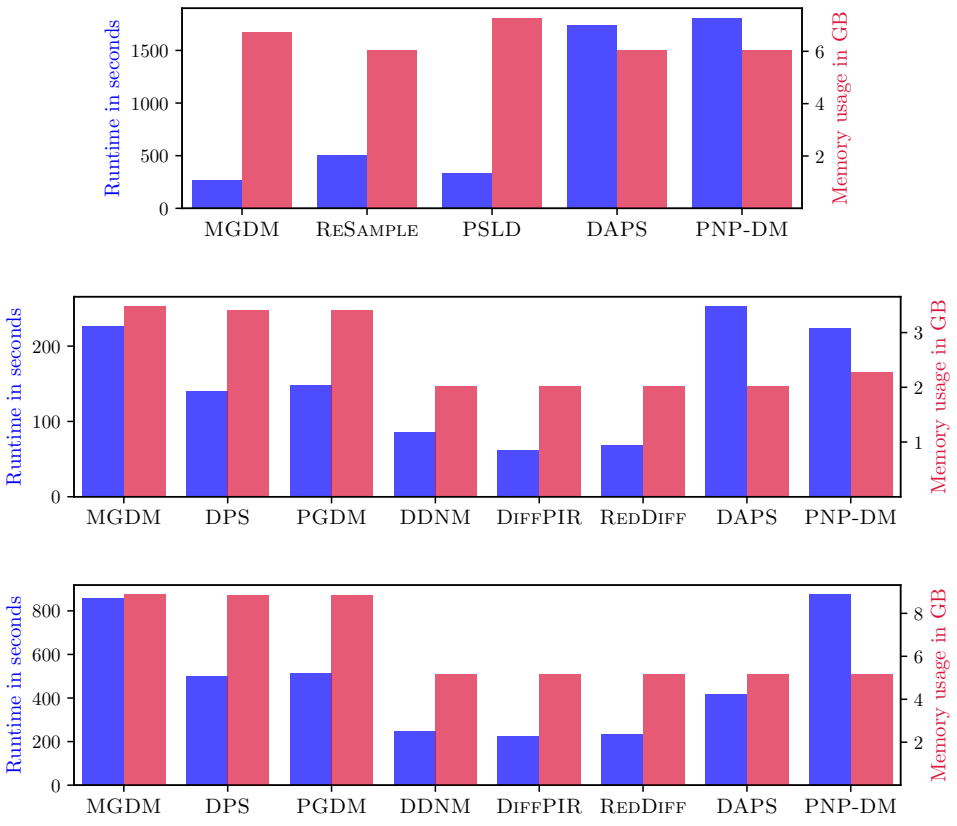

Figure 6: Comparison of the runtime (red bars – left axis) and memory requirement (blue bars – right axis) between the considered algorithms on `FFHQ` latent space (1st row), `FFHQ` pixel space (2nd row), and `ImageNet` (3rd row).

B.8 Experiments with high noise setup

To ensure the performance of the MGDM extend to higher noise setups, we evaluat it on both Half-mask (linear task) and JPEG QF=2% (nonlinear task) while increasing the noise level to $\sigma_{\mathbf{y}} = 0.3$, increased of 0.05. The experiments were

conducted on the FFHQ dataset in pixel-space, with MGDM being compared against DPS and DAPS. As evidenced by Table 10, MGDM consistently outperforms the other algorithms, namely in terms of LPIPS.

Table 10: FID and mean LPIPS/PSNR/SSIM metrics along side 95%-confidence interval on FFHQ 256×256 dataset with $\sigma_{\mathbf{y}} = 0.3$.

	MGDM	DPS	DAPS	MGDM	DPS	DAPS	MGDM	DPS	DAPS	MGDM	DPS	DAPS
Task	LPIPS ↓			FID ↓			PSNR ↑			SSIM ↑		
Half mask	0.23 ±0.01	0.29 ±0.01	0.67 ±0.01	56.77	79.83	136.26	15.59 ±0.28	14.86 ±0.27	15.08 ±0.23	0.64 ±0.01	0.58 ±0.01	0.36 ±0.01
JPEG (QF = 2)	0.21 ±0.01	0.37 ±0.03	0.24 ±0.01	59.78	110.60	65.11	22.70 ±0.16	18.69 ±0.54	23.62 ±0.15	0.67 ±0.01	0.54 ±0.02	0.69 ±0.01

B.9 Experiments with Poisson-noise likelihood

Pervious experiments were performed with Gaussian-noise likelihood, $g_0(\mathbf{y}|\mathbf{x}) = \mathrm{N}(\mathbf{y}; \mathbf{A}(\mathbf{x}), \sigma_{\mathbf{y}}^2\mathbf{I})$. Here, we extend our evaluation to a Poisson-noise likelihood

$$g_0(\mathbf{y}|\mathbf{x}) = \exp\big(-\lambda\mathbf{A}(\mathbf{x})\big)\frac{(-\lambda\mathbf{A}(\mathbf{x}))^{\mathbf{y}}}{\mathbf{y}!},$$

where $\lambda > 0$ is the Poisson rate. We compare MGDM against DPS on several tasks: denoising, super-resolution ($\times 4$), Gaussian deblurring, and motion deblurring. These experiments were conducted on the FFHQ dataset in pixel space with a Poisson rate of $\lambda = 0.05$. Notably, while DPS relies on a Gaussian approximation (Chung et al., 2023, Eqn. (19)) due to the inherent challenges of Poisson likelihoods, we directly implement the Poisson likelihood without approximation. As detailed in Table 11, MGDM consistently outperforms DPS across all evaluated metrics.

Table 11: FID and mean LPIPS/PSNR/SSIM metrics along side 95%-confidence interval on FFHQ 256×256 dataset with Poisson noise with Poisson rate $\lambda = 0.05$.

	MGDM	DPS	MGDM	DPS	MGDM	DPS	MGDM	DPS
Task	LPIPS ↓		FID ↓		PSNR ↑		SSIM ↑	
Denoising	0.08 ±0.00	0.15 ±0.01	54.73	60.48	28.81 ±0.17	18.96 ±0.16	0.83 ±0.00	0.68 ±0.01
SR ($\times 4$)	0.25 ±0.01	0.25 ±0.01	74.75	69.49	21.65 ±0.21	17.62 ±0.12	0.65 ±0.01	0.56 ±0.01
Gaussian Deblur	0.20 ±0.01	0.30 ±0.01	64.83	100.63	23.09 ±0.22	16.49 ±0.16	0.66 ±0.01	0.49 ±0.01
Motion Deblur	0.21 ±0.01	0.27 ±0.01	60.53	69.73	22.52 ±0.20	16.91 ±0.15	0.66 ±0.01	0.52 ±0.01

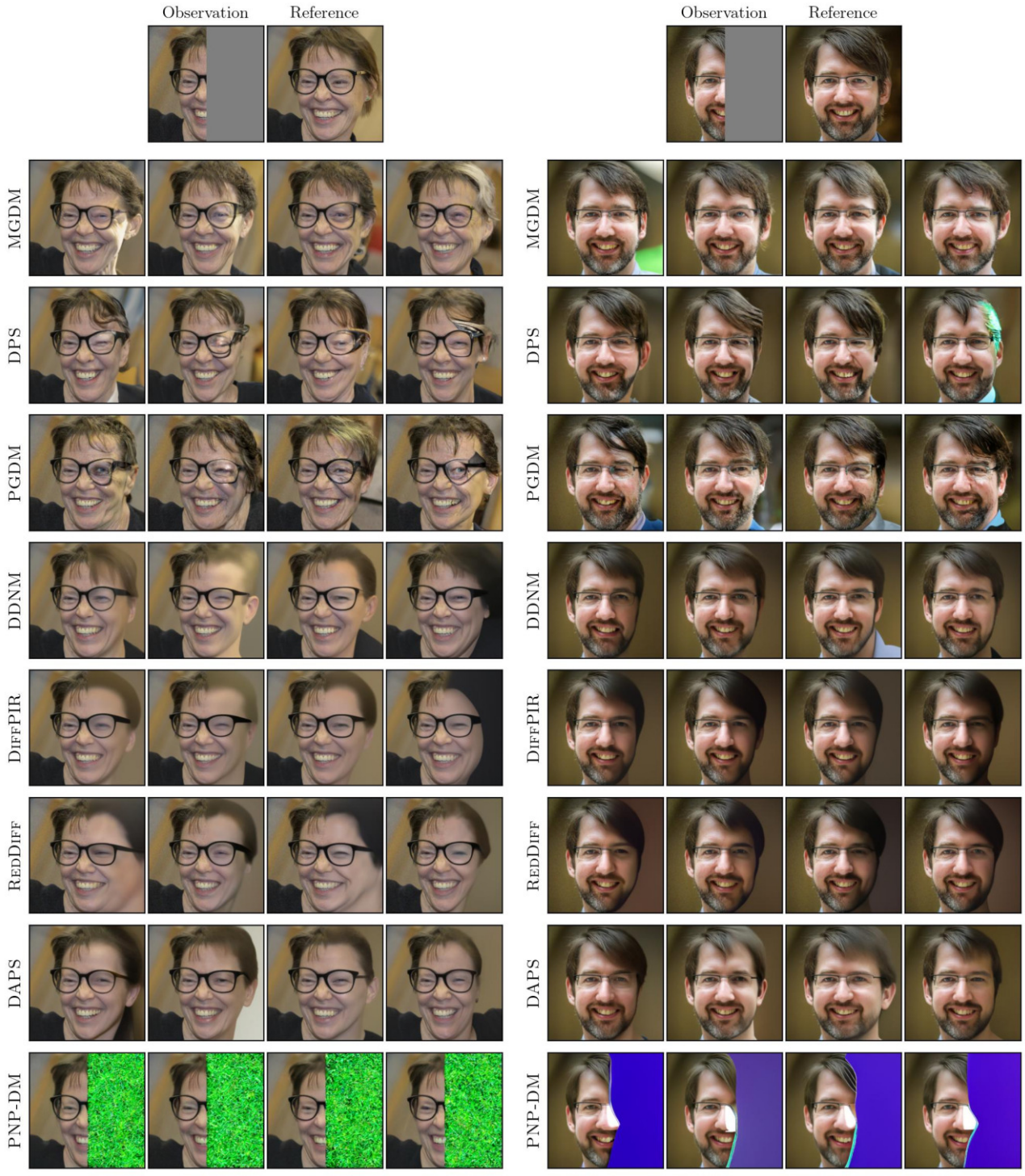

Figure 7: Reconstructions for half mask inpainting on FFHQ dataset.

Figure 8: Reconstructions for box inpainting on FFHQ dataset.

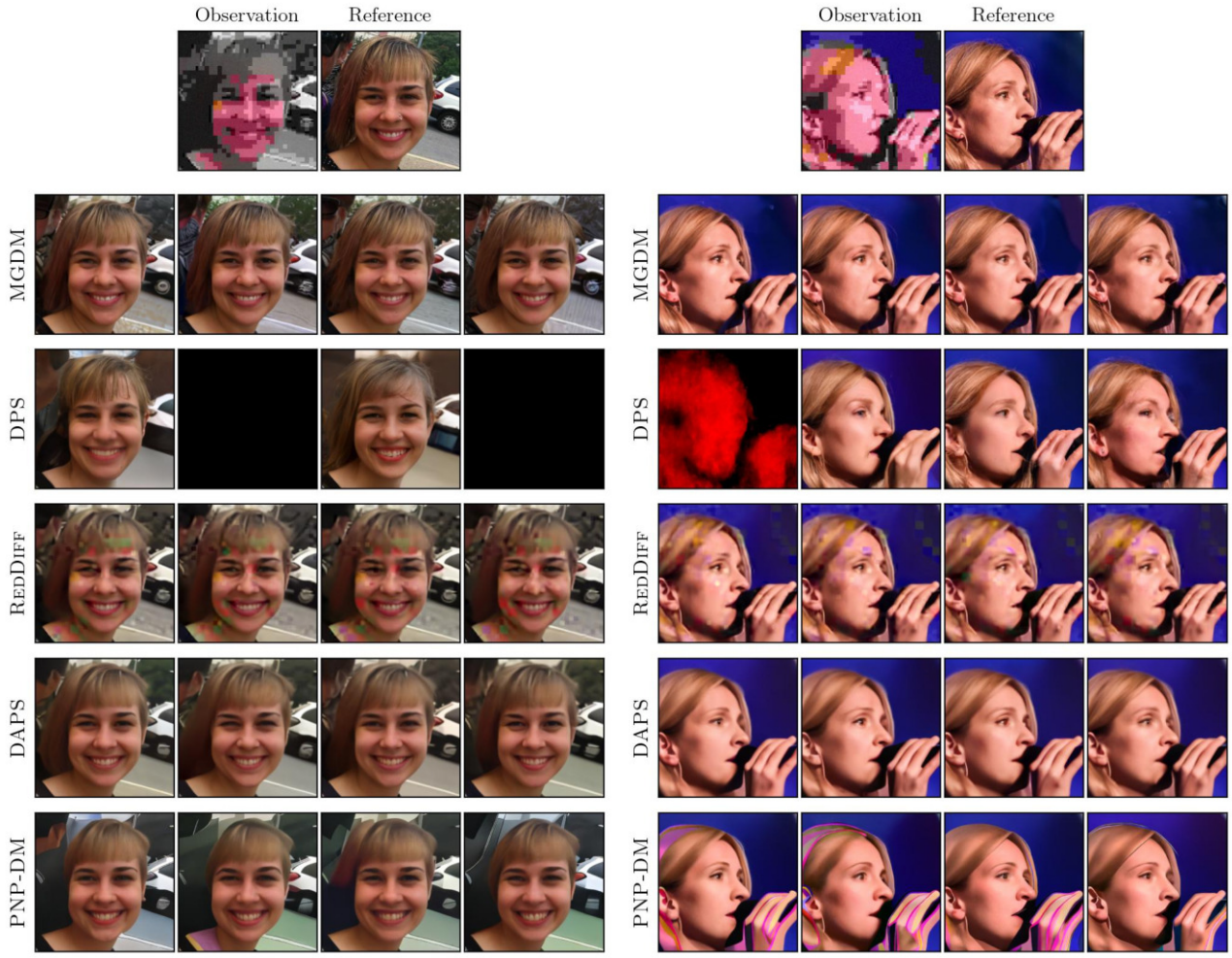

Figure 9: Reconstructions for JPEG dequantization QF=2% on FFHQ dataset.

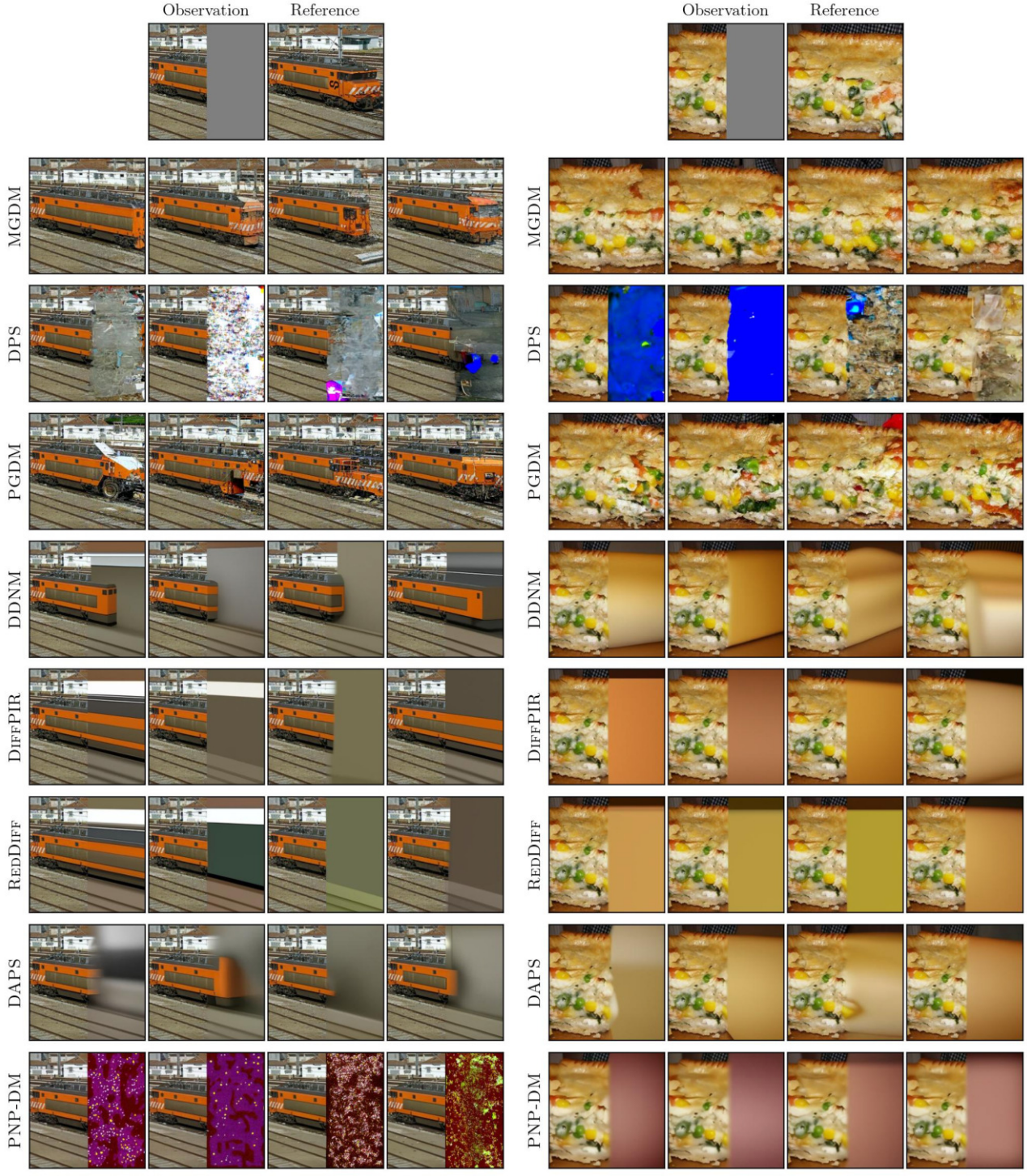

Figure 10: Reconstructions Half mask inpainting on `ImageNet` dataset.

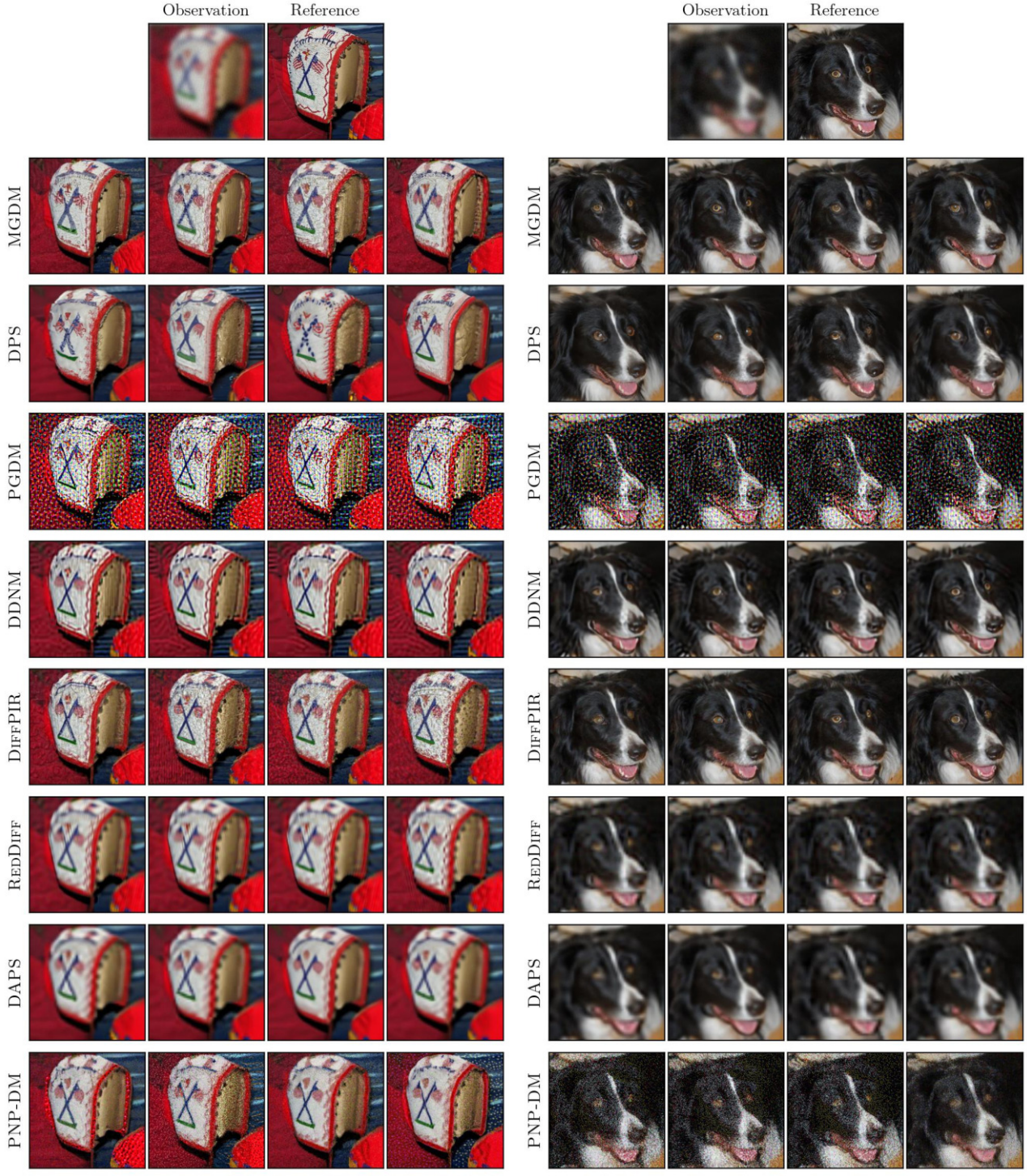

Figure 11: Reconstructions for Gaussian deblurring on `ImageNet` dataset.

Figure 12: Reconstructions for motion deblurring on FFHQ dataset.

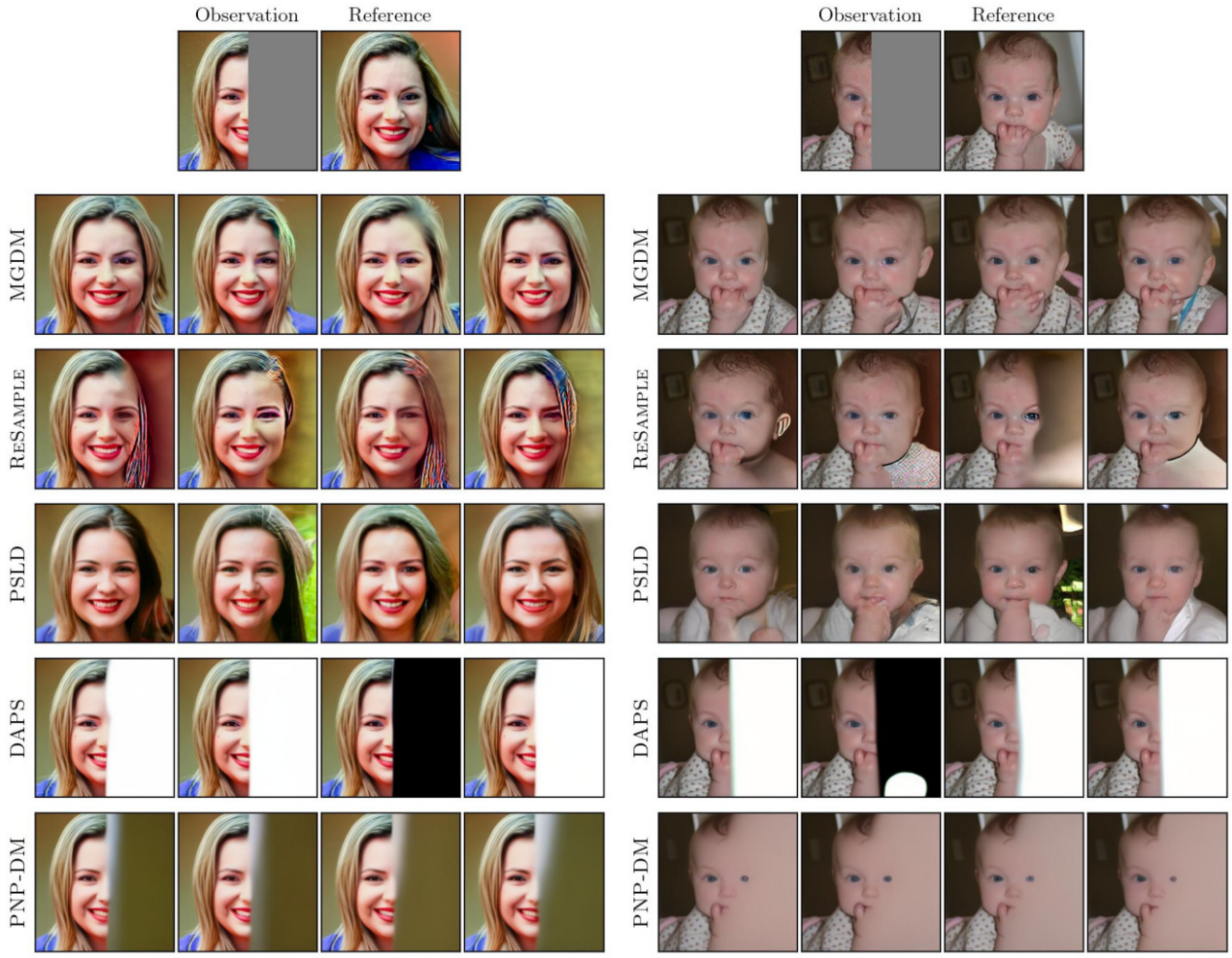

Figure 13: Reconstructions for half mask inpainting on FFHQ dataset with LDM prior.

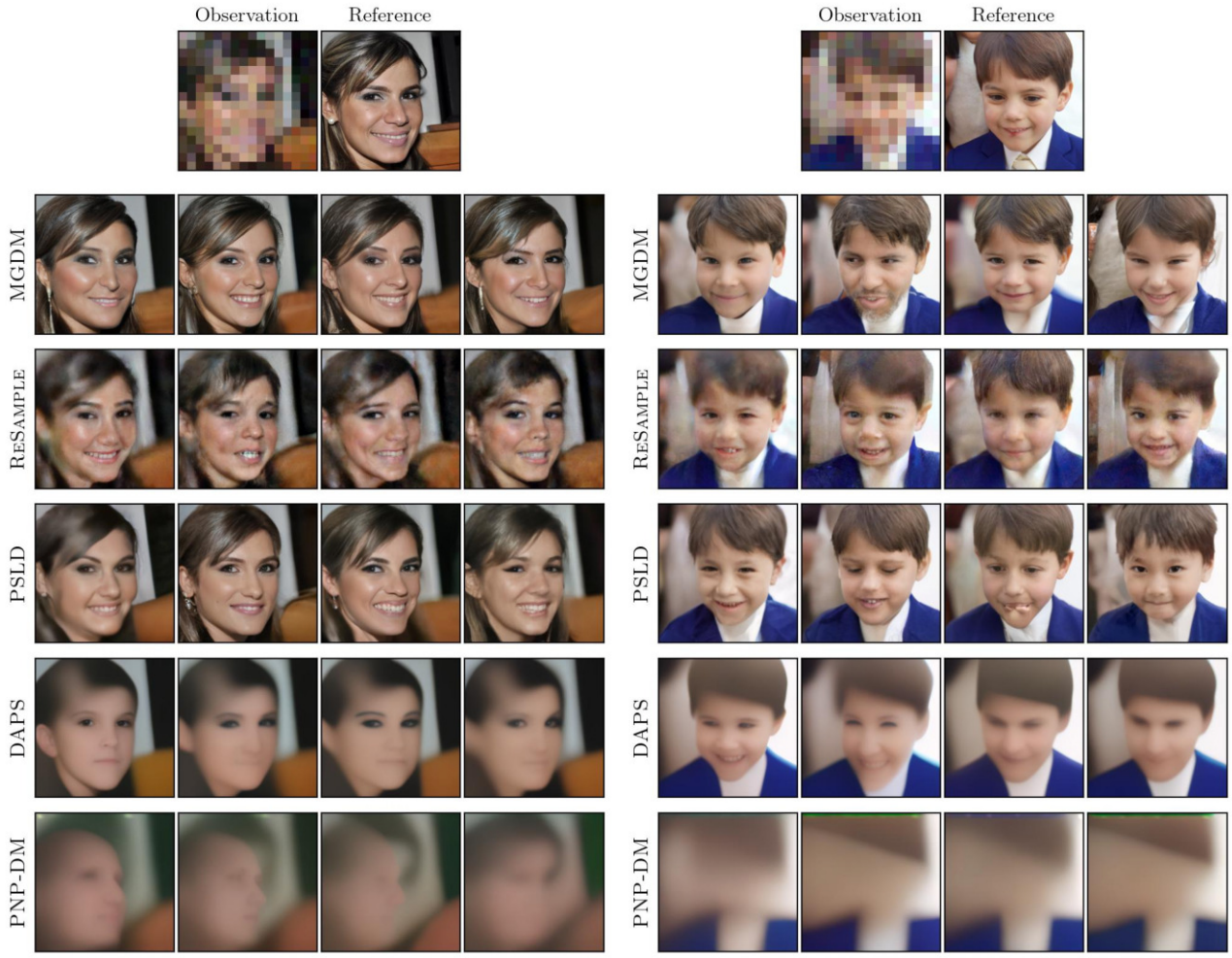

Figure 14: Reconstructions for SR $\times 16$ on FFHQ dataset with LDM prior.

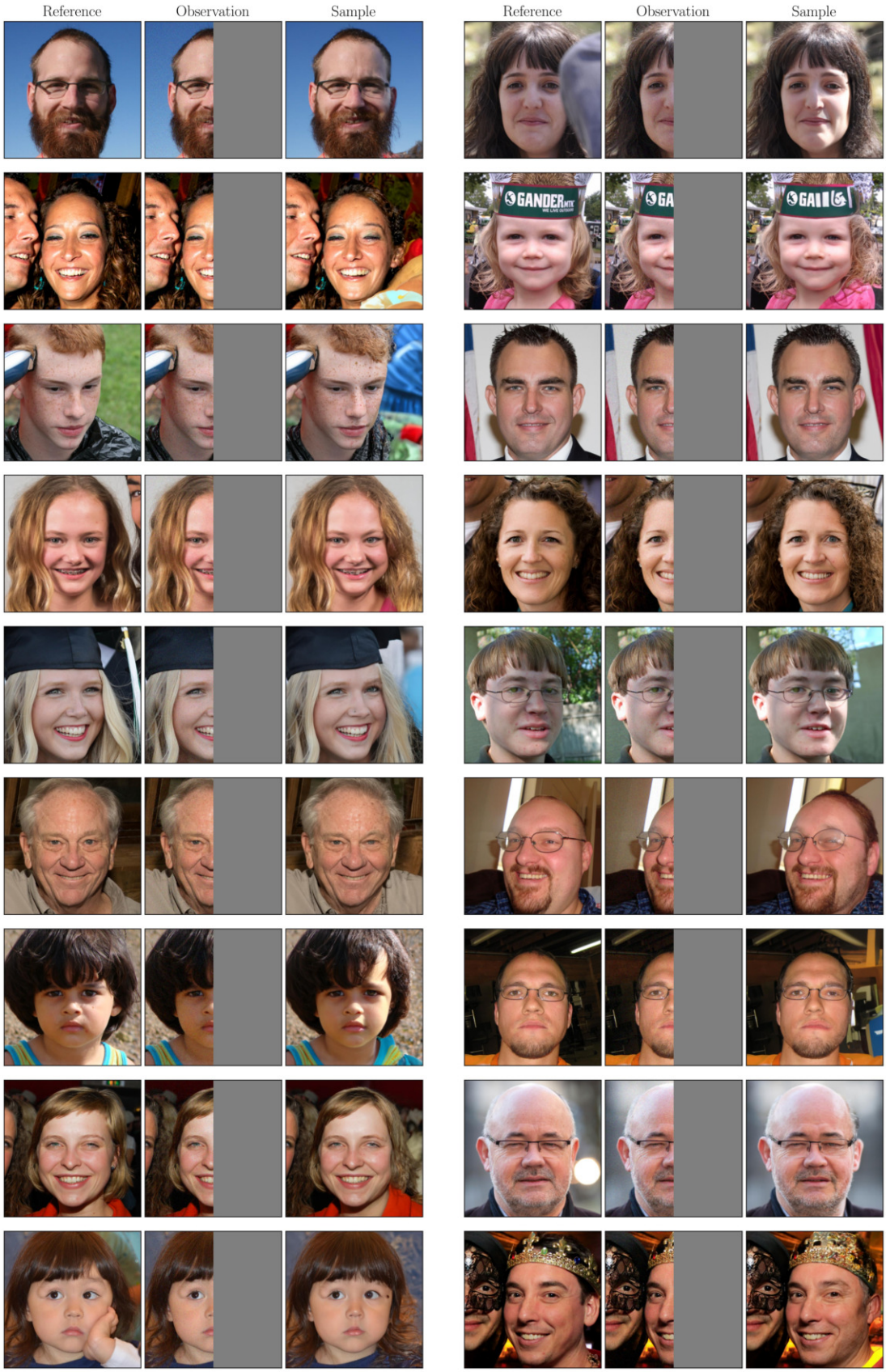

Figure 15: Half mask inpainting on FFHQ dataset.

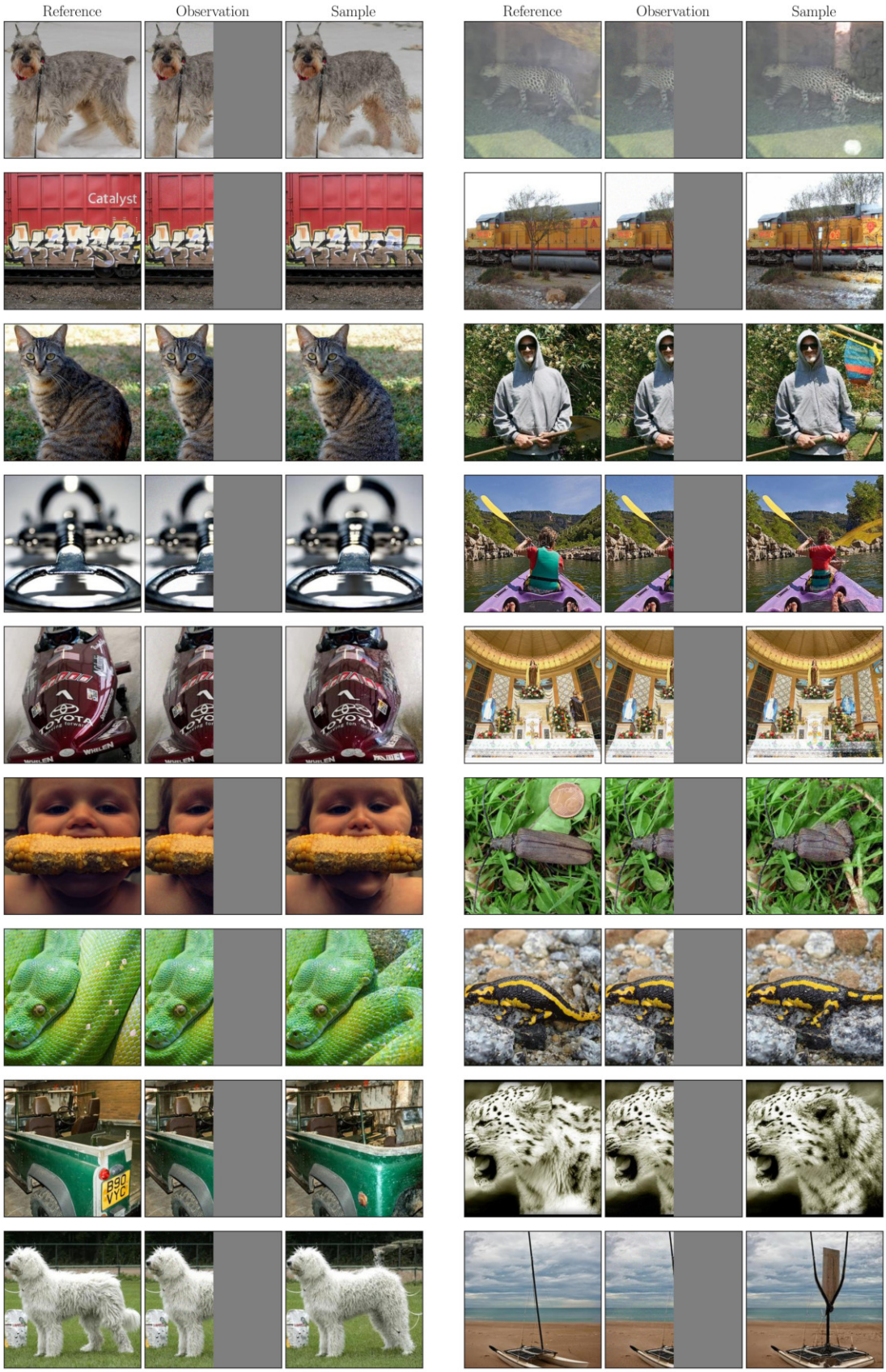

Figure 16: Half mask inpainting on `ImageNet` dataset.

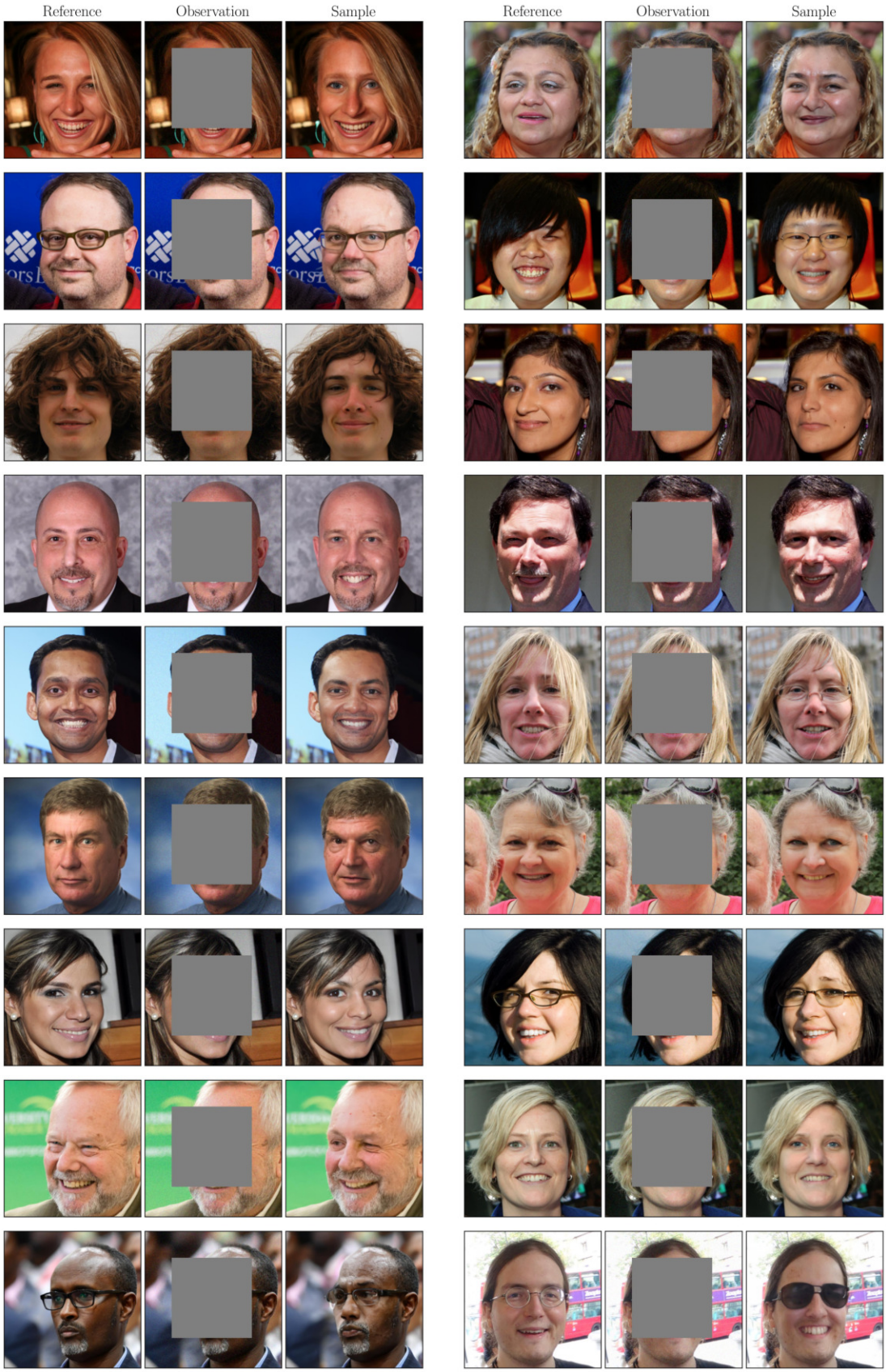

Figure 17: Box inpainting on FFHQ dataset.

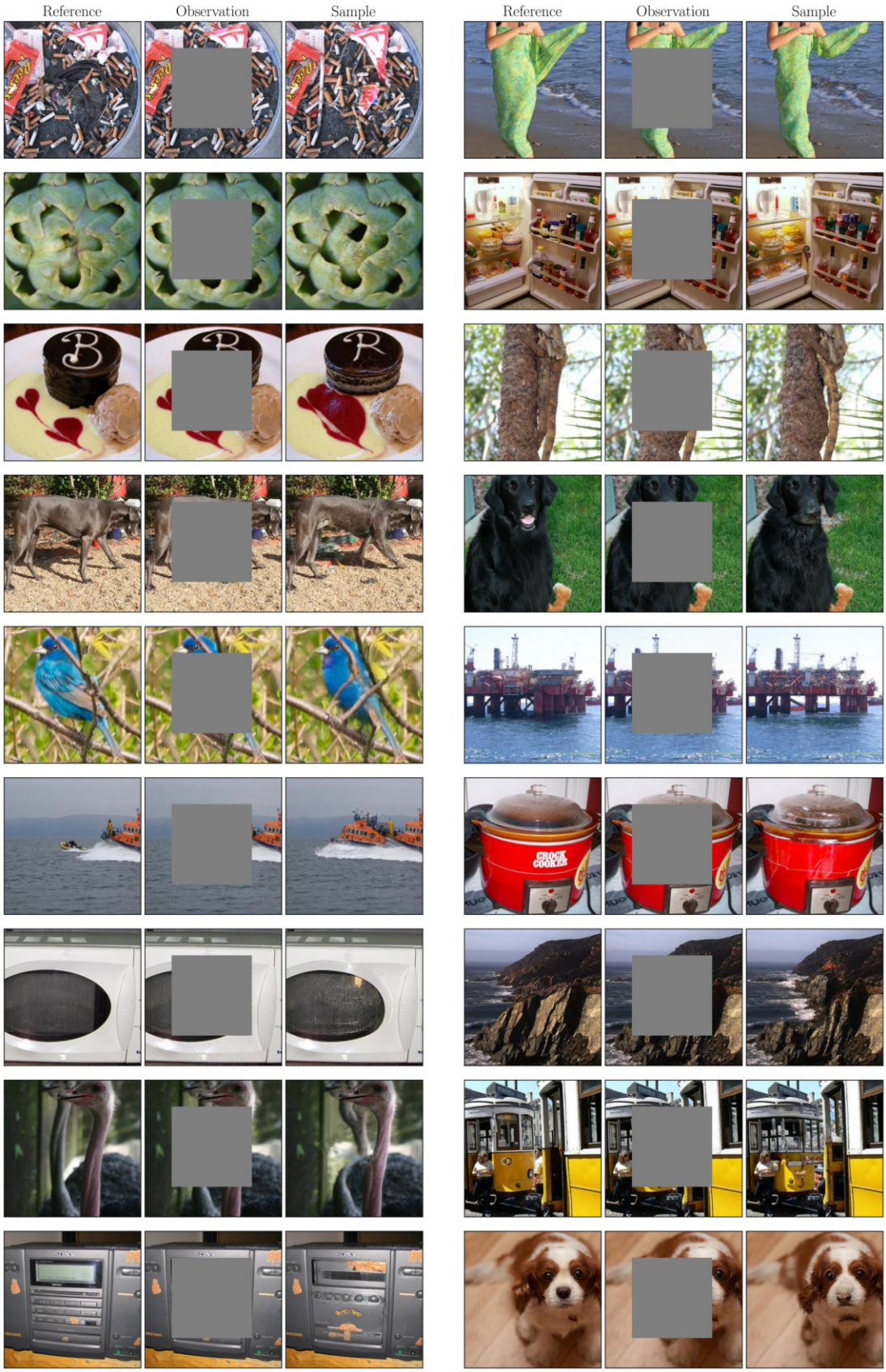

Figure 18: Box inpainting on `ImageNet` dataset.

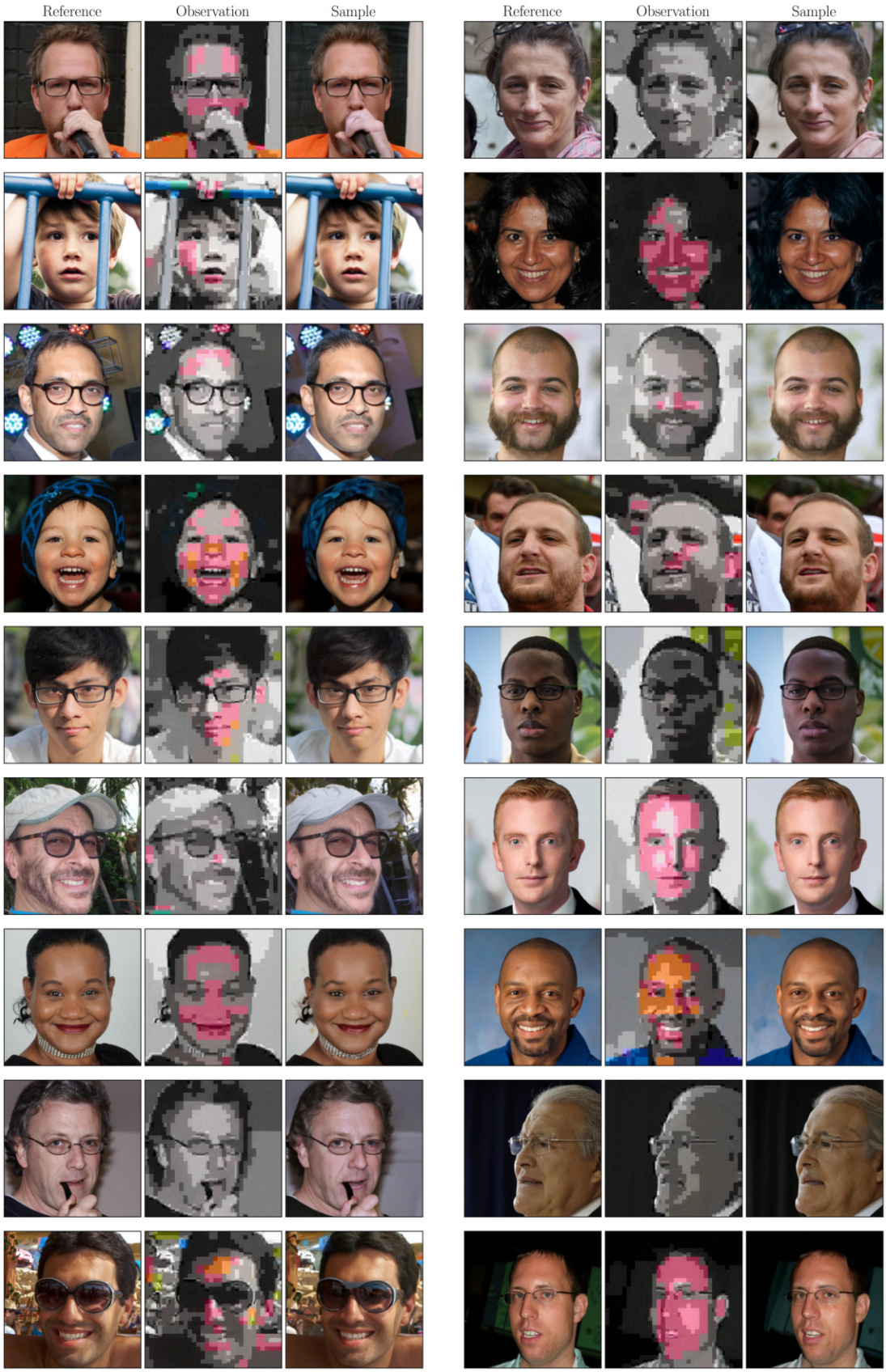

Figure 19: JPEG dequantization with QF $= 2$ on FFHQ dataset.

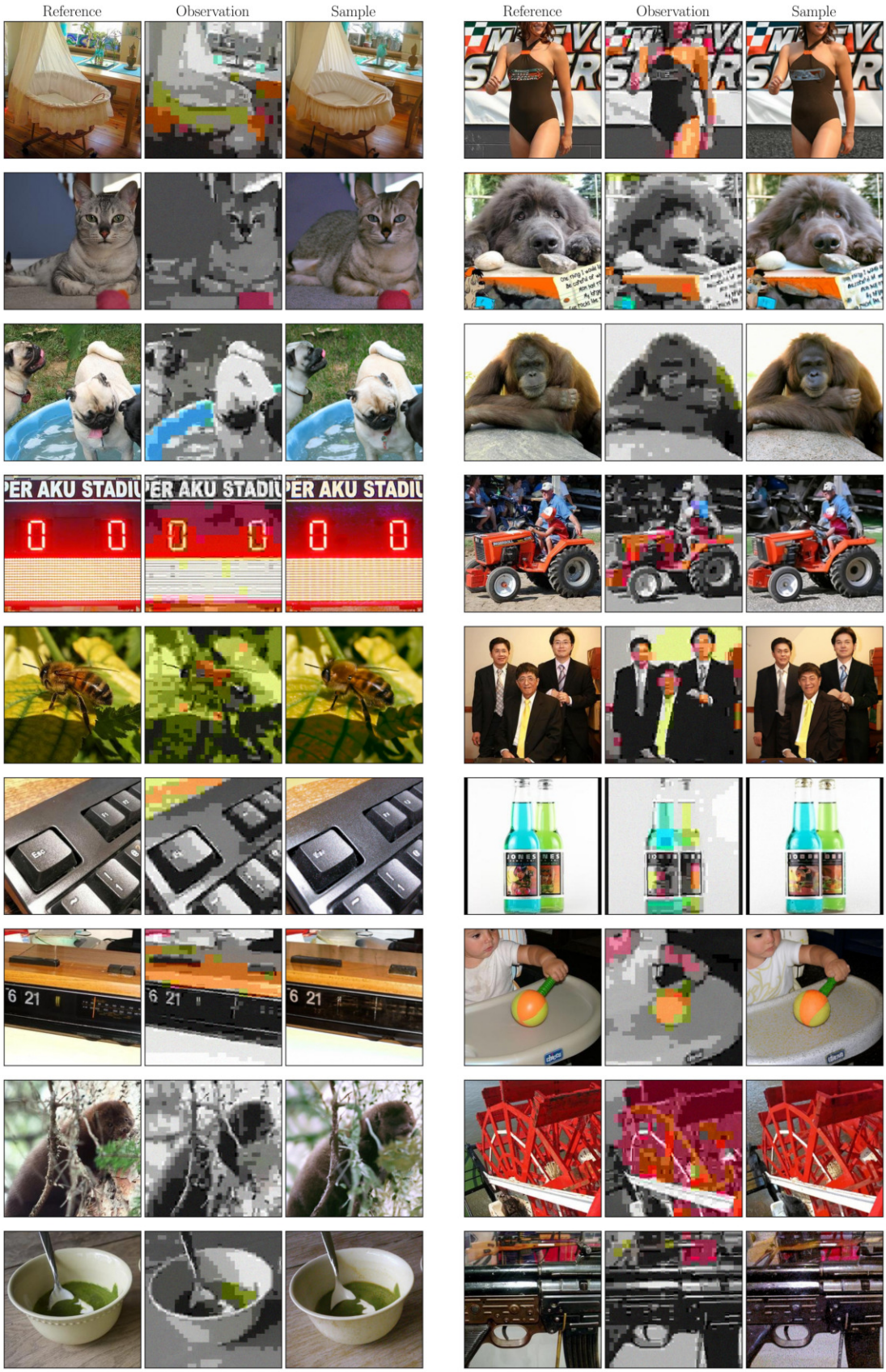

Figure 20: JPEG dequantization with QF $= 2$ on `ImageNet` dataset.

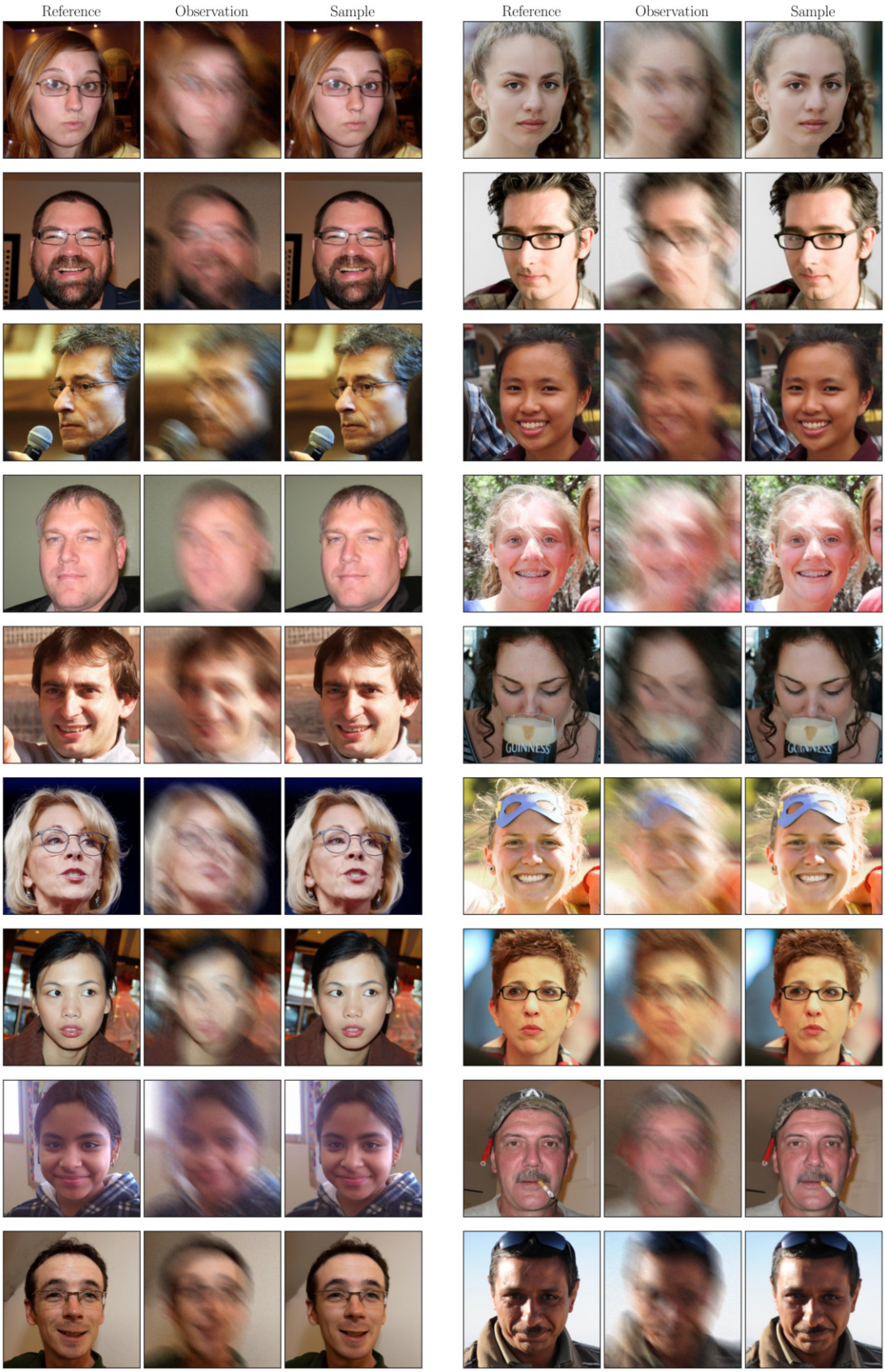

Figure 21: Motion deblurring on FFHQ dataset.

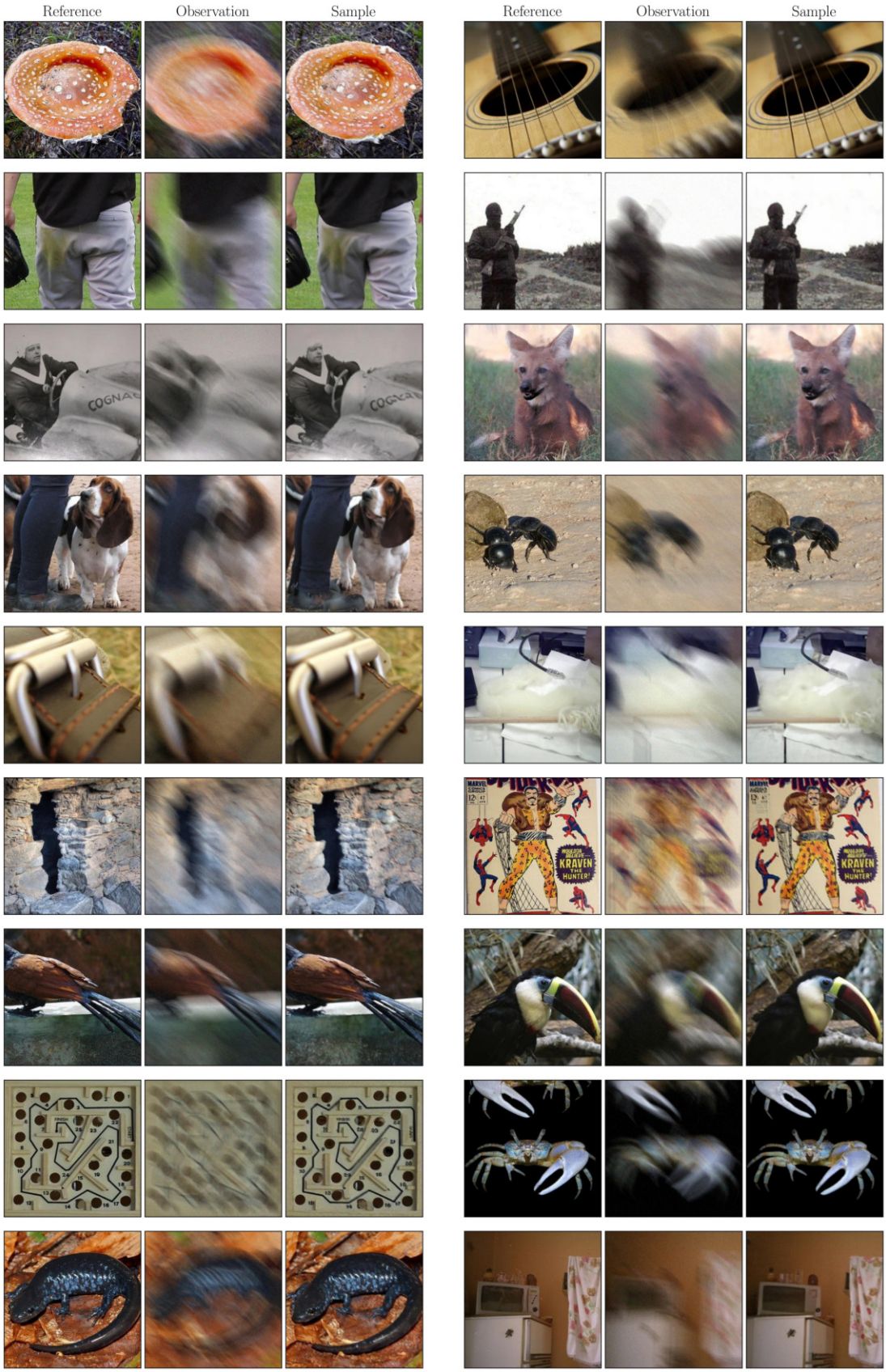

Figure 22: Motion deblurring on `ImageNet` dataset.

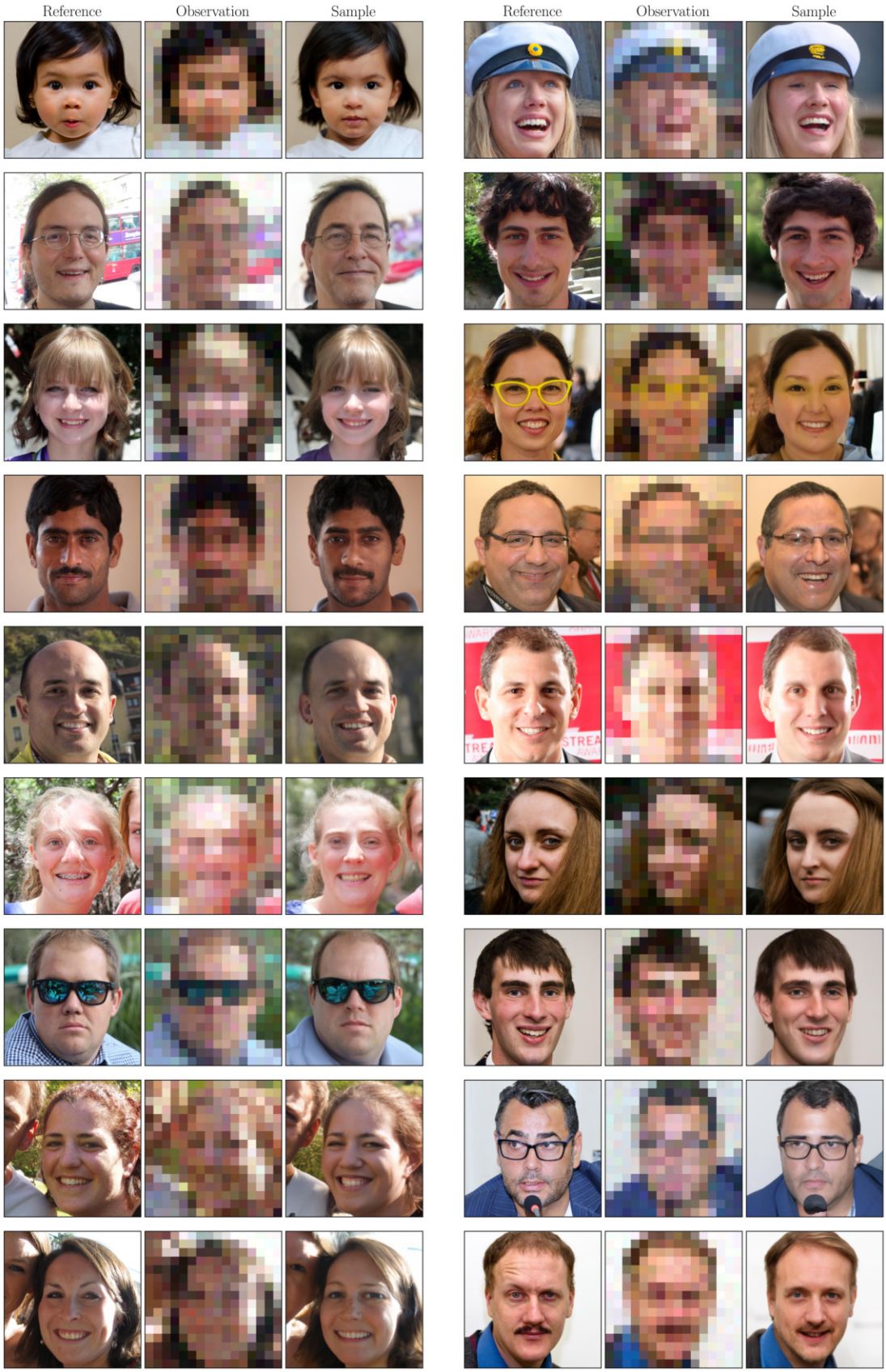

Figure 23: SR(16×) on FFHQ dataset.

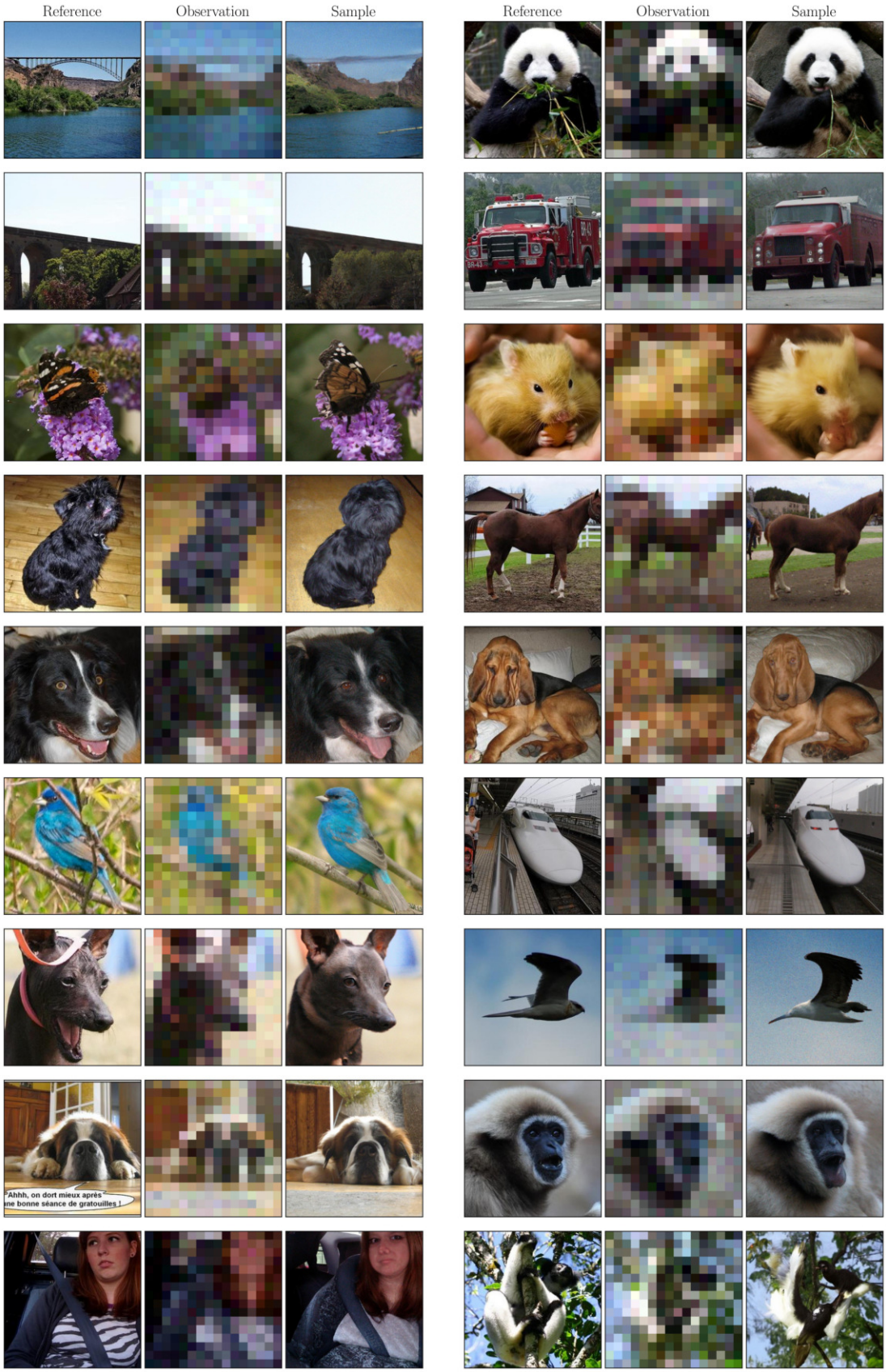

Figure 24: SR(16×) on `ImageNet` dataset.

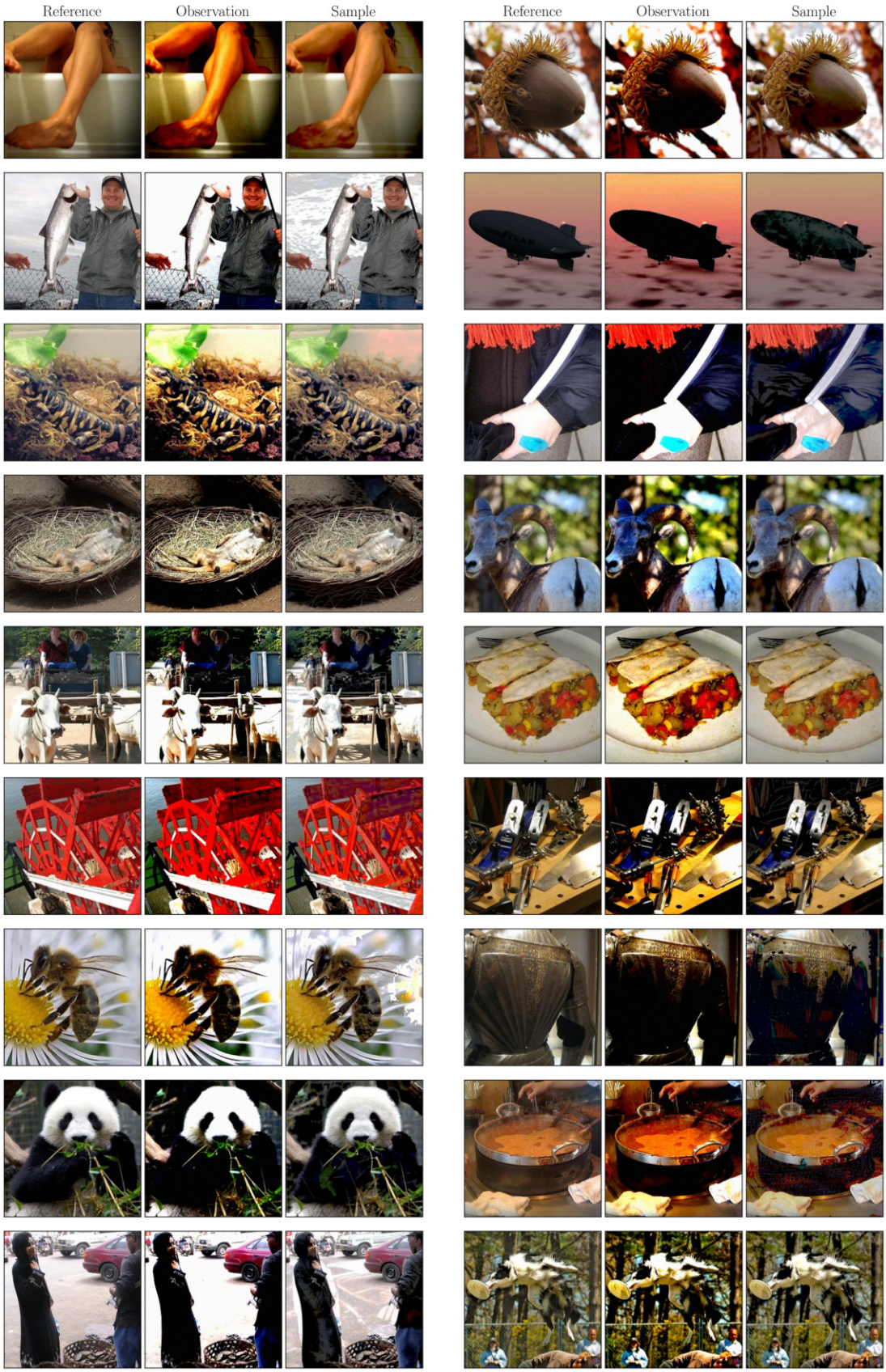

Figure 25: High dynamic range on `ImageNet` dataset.

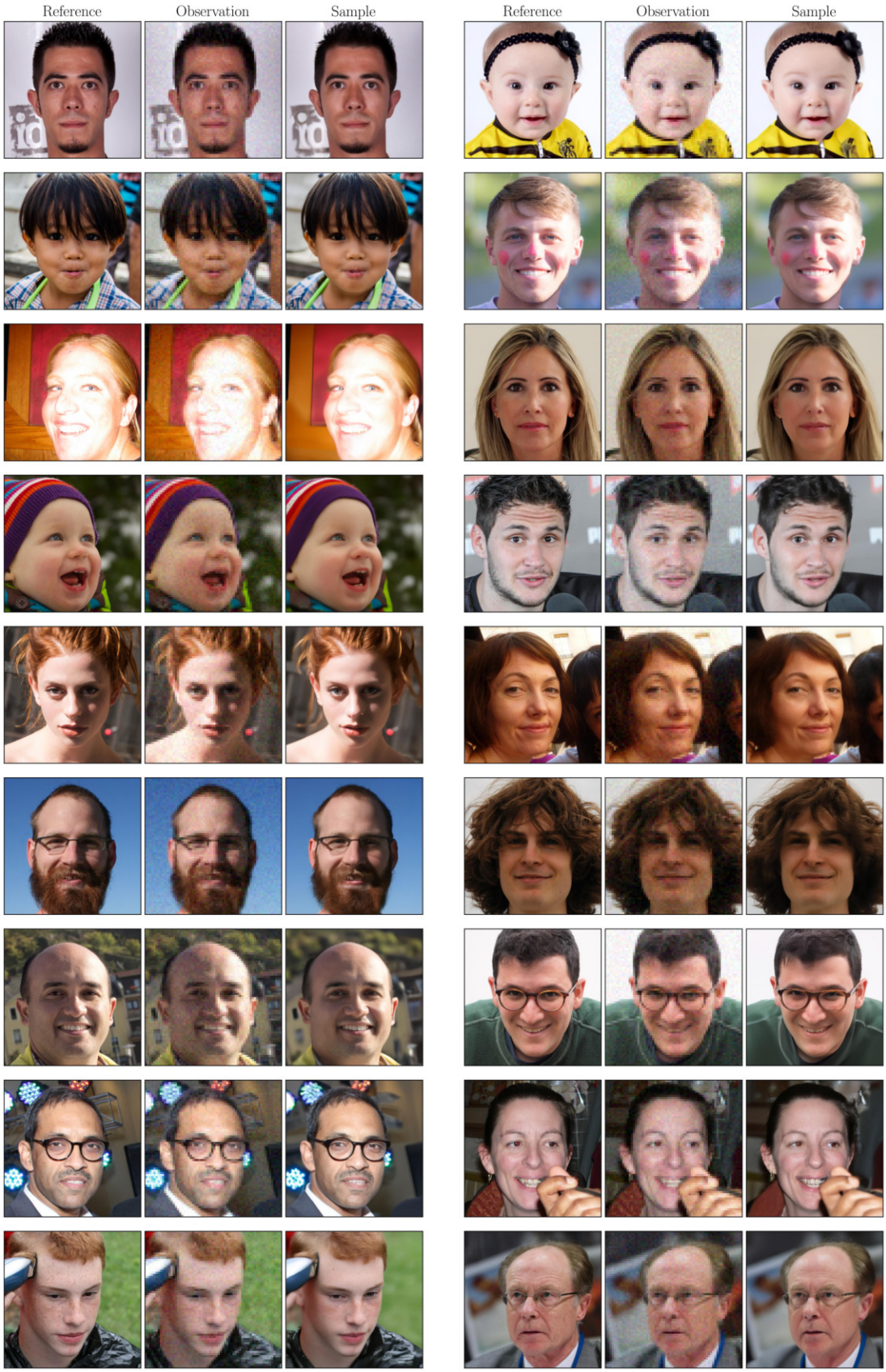

Figure 26: SR(4×) on FFHQ dataset with latent diffusion.

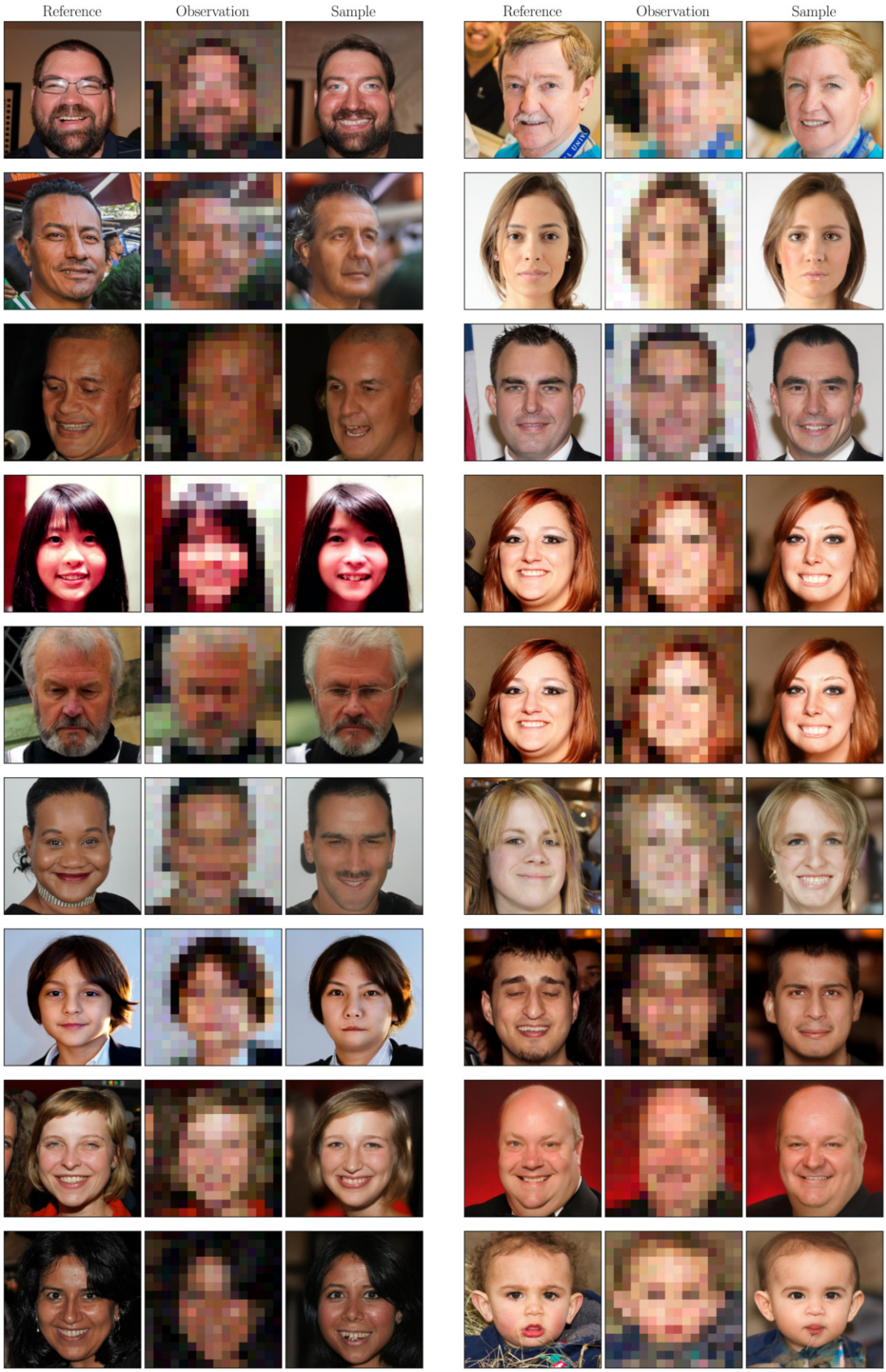

Figure 27: SR(16×) on FFHQ dataset with latent diffusion.

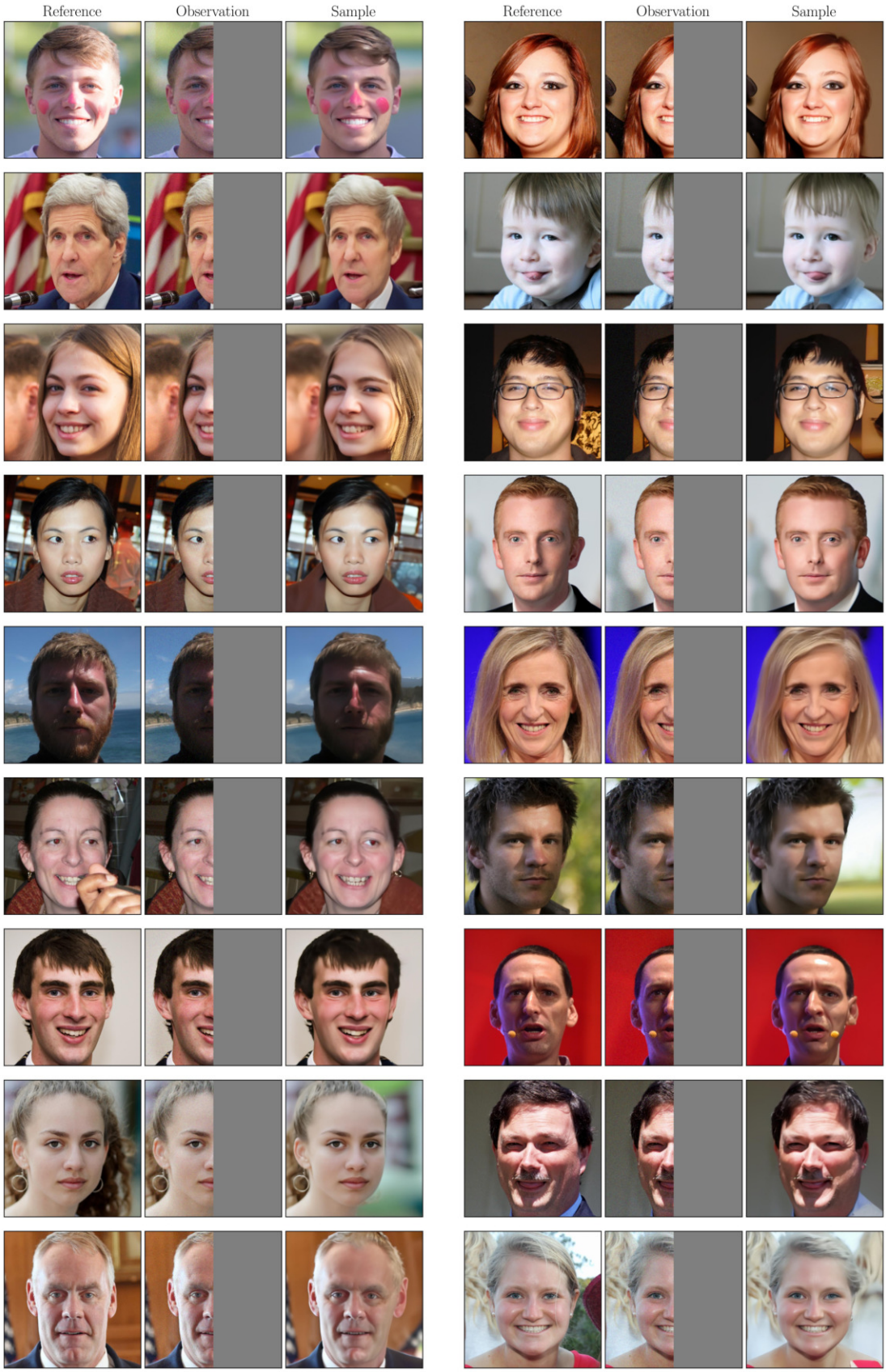

Figure 28: Half mask on FFHQ dataset with latent diffusion.