Divide-and-Conquer Posterior Sampling for Denoising Diffusion Priors

Yazid Janati[*,1] **Badr Moufad**[*,1]
Alain Durmus[1] **Eric Moulines**[1,3] **Jimmy Olsson**[2]
[1] CMAP, Ecole polytechnique [2] KTH Royal Institute of Technology [3] MBZUAI

Abstract

Recent advancements in solving Bayesian inverse problems have spotlighted denoising diffusion models (DDMs) as effective priors. Although these have great potential, DDM priors yield complex posterior distributions that are challenging to sample. Existing approaches to posterior sampling in this context address this problem either by retraining model-specific components, leading to stiff and cumbersome methods, or by introducing approximations with uncontrolled errors that affect the accuracy of the produced samples. We present an innovative framework, divide-and-conquer posterior sampling, which leverages the inherent structure of DDMs to construct a sequence of intermediate posteriors that guide the produced samples to the target posterior. Our method significantly reduces the approximation error associated with current techniques without the need for retraining. We demonstrate the versatility and effectiveness of our approach for a wide range of Bayesian inverse problems. The code is available at https://github.com/Badr-MOUFAD/dcps

1 Introduction

Many problems in machine learning can be formulated as inverse problems, such as superresolution, deblurring, and inpainting, to name but a few. They all have the same goal, namely to recover a signal of interest from an indirect observation. One line of research addresses these problems through the lens of the Bayesian framework by specifying two components: a prior distribution, which embodies the specification of the signal, and a likelihood that describes the law of the observation conditionally on the signal. Once these elements are specified, the inverse problem is solved by sampling from the posterior distribution, which, after including the observation, contains all available information about the signal and thus about its uncertainty as well [12]. The importance of the specification of the prior in solving Bayesian ill-posed inverse problems is paramount. In the last decade, the success of priors based on deep generative models has fundamentally changed the field of linear inverse problems [40, 55, 19, 36, 24]. Recently, denoising diffusion probabilistic models (DDMs) have received special attention. Thanks to their ability to learn complex and multimodal data distributions, DDM represent the state-of-the-art in many generative modeling tasks, *e.g.* image generation [45, 20, 50, 52, 15, 46, 49], super-resolution [43, 1], and inpainting [45, 11, 22].

Popular methods to sample from posterior distribution include Markov chain Monte Carlo (MCMC) and variational inference; see [53, 6] and the references therein. These methods are iterative schemes that require an explicit procedure to evaluate pointwise the prior distribution and often its (Stein) score function [21] in order to compute acceptance ratios and construct efficient proposals. While

* Equal contribution
Corresponding authors: {yazid.janati,badr.moufad}@polytechnique.edu

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

sampling from the DDM priors is straightforward, posterior sampling is usually challenging since the intractability of the posterior density and its score make them computationally prohibitive and thus invalidate all conventional simulation methods. Although approximations exist, their associated iterative sampling schemes can be computationally intensive and exhibit high sensitivity to the choice of hyperparameters; see *e.g.* [24].

This paper proposes the DIVIDE-AND-CONQUER POSTERIOR SAMPLER (DCPS), a novel approach to posterior sampling in Bayesian inverse problems with DDM priors. Thanks to the Markov property of the data-generating backward diffusion, the posterior can be expressed as the marginal distribution of a Feynman–Kac (FK) path measure [13], whose length corresponds to the number of diffusion steps and whose user-defined potentials serve to bias the dynamics of the data-generating backward diffusion to align with the likelihood of the observation. Besides, for a given choice of potentials, the FK path law becomes Markovian, making it possible to express the posterior as the marginal of a time-reversed inhomogeneous Markov chain.

This approach is tempting, yet, the backward Markov decomposition remains difficult to apply in practice as these specific potential functions are difficult to approximate, especially when the number of diffusion steps is large. We tackle this problem with a divide-and-conquer approach. More precisely, instead of targeting the given posterior by a single simulation run through the full backward decomposition, our proposed scheme targets backward a sequence $(\pi_{k_\ell})_{\ell=0}^{L}$ of distributions along the path measure leading to the target posterior distribution (section 3). These distributions are induced by a sequence of increasingly complex potentials and converge to the target distribution. Starting with a sample from $\pi_{k_{\ell+1}}$, a draw from π_{k_ℓ} is formed by a combination of Langevin iterations and the simulation of an inhomogeneous Markov chain. In other words, π_{k_ℓ} is expressed as the final marginal distribution of a time-reversed inhomogeneous Markov chain of moderate length $k_{\ell+1} - k_\ell \in \mathbb{N}^*$ with an initial distribution $\pi^{\ell}_{k_{\ell+1}}$. This chain, whose transition densities are intractable, is approximately sampled using Gaussian variational inference. The rationale behind our approach stems from the observation that the Gaussian approximation error can be reduced by shortening the length of the intermediate FK path measures (*i.e.*, by increasing L); a result that we show in Proposition A.1. We finally illustrate that our algorithm can provide high-quality solutions to Bayesian inverse problems involving a variety of datasets and tasks.

To sum up our contribution, we

- show that the existing approximations of the Markovian backward decomposition can be improved using a bridge-kernel smoothing technique
- design a novel divide-and-conquer sampling approach that enables efficient bias-reduced sampling from the posterior, and illustrate its performance on several Bayesian inverse problems including inpainting, outpainting, Poisson imaging, and JPEG dequantization,
- propose a new technique to efficiently generate approximate samples from the backward decomposition using Gaussian variational inference.

Notation. For $(m, n) \in \mathbb{N}^2$ such that $m < n$, we let $[\![m, n]\!] := \{m, \dots, n\}$. We use $\mathrm{N}(x; \mu, \Sigma)$ to denote the density at x of a Gaussian distribution with mean μ and covariance matrix Σ. I_d is the d-dimensional identity matrix and δ_a denotes the Dirac mass at a. W_2 denotes the Wasserstein distance of order 2. We use uppercase for random variables and lowercase for their realizations.

2 Posterior sampling with DDM prior

DDM priors. We provide a brief overview of DDMs [45, 50, 20]. Suppose we can access an empirical sample from some data distribution p_{data} defined on $\mathbb{R}^{d_x}$. For $n \in \mathbb{N}$ large enough and $k \in [\![0, n]\!]$, define the distribution $q_k(x_k) := \int p_{\text{data}}(x_0)\, q_{k|0}(x_k|x_0)\mathrm{d}x_0$ with $q_{k|0}(x_k|x_0) := \mathrm{N}(x_k; \sqrt{\alpha_k}x_0, (1-\alpha_k)I_{d_x})$, where $(\alpha_k)_{k=0}^{n}$ is a decreasing sequence with $\alpha_0 = 1$ and α_n approximately equals zero. The probability density q_k corresponds to the marginal distribution at time k of an auto-regressive process on $\mathbb{R}^{d_x}$ given by $X_{k+1} = \sqrt{\alpha_{k+1}/\alpha_k}X_k + \sqrt{1-\alpha_{k+1}/\alpha_k}\epsilon_{k+1}$, with $X_0 \sim p_{\text{data}}$ and $(\epsilon_k)_{k=0}^{n}$ being a sequence of i.i.d. d_x-dimensional standard Gaussians.

DDMs leverage parametric approximations $\hat{x}^{\theta}_{0|k}$ of the mappings $x_k \mapsto \int x_0\, q_{0|k}(x_0|x_k)\mathrm{d}x_0$, where $q_{0|k}(x_0|x_k) \propto p_{\text{data}}(x_0)q_{k|0}(x_k|x_0)$ is the conditional distribution of X_0 given $X_k = x_k$. Each $\hat{x}^{\theta}_{0|k}$

is defined as $\hat{x}^{\theta}_{0|k}(x_k) := (x_k - \sqrt{1-\alpha_k}\hat{\epsilon}^{\theta}_k(x_k))/\sqrt{\alpha_k}$, where $\hat{\epsilon}^{\theta}_k$ is a noise predictor network trained by minimizing a denoising objective; see [46, Eq. (5)] and Appendix A for details. Following [15, Section 4.2], $\hat{\epsilon}^{\theta}_k$ also provides an estimate of the score $\nabla \log q_k(x_k)$ given by $\hat{s}^{\theta}_k(x_k) := -\big(x_k - \sqrt{\alpha_k}\hat{x}^{\theta}_{0|k}(x_k)\big)/(1-\alpha_k)$. We denote by $\theta^\star$ the minimizer of the denoising objective. Having access to $\theta^\star$, we can define a generative model for p_{data} by adopting the denoising diffusion probabilistic model (DDPM) framework of [20]. As long as n is large enough, q_n can be confused with a multivariate standard Gaussian. Define the *bridge kernel* $q_{k|0,k+1}(x_k|x_0, x_{k+1}) \propto q_{k|0}(x_k|x_0)q_{k+1|k}(x_{k+1}|x_k)$ which is a Gaussian distribution with mean $\mu_{k|0,k+1}(x_0, x_{k+1})$ and diagonal covariance $\sigma^2_{k|k+1} I_{d_x}$ defined in Appendix A.1. Define the generative model for p_{data} as

$$p^{\theta^\star}_{0:n}(x_{0:n}) = p_n(x_n) \prod_{k=0}^{n-1} p^{\theta^\star}_{k|k+1}(x_k|x_{k+1}) \,, \tag{2.1}$$

where for every $k \in [\![1, n-1]\!]$, the backward transitions are

$$p^{\theta^\star}_{k|k+1}(x_k|x_{k+1}) := q_{k|0,k+1}(x_k|\hat{x}^{\theta^\star}_{0|k+1}(x_{k+1}), x_{k+1}) \,, \tag{2.2}$$

with $p^{\theta^\star}_{0|1}(\cdot|x_1) := \delta_{\hat{x}^{\theta^\star}_{0|1}(x_1)}$ and $p_n(x_n) = \mathrm{N}(x_n; 0, I_{d_x})$. In the following, we assume that we have access to a pre-trained DDM and omit the superscript $\theta^\star$ from the notation, writing simply p and $\hat{x}_{0|k}$ when referring to the generative model and the denoiser, respectively. In addition, we denote by p_k the k-th marginal of $p_{0:n}$ and write, for all $(\ell, m) \in [\![0, n]\!]^2$ such that $\ell < m$, $p_{\ell|m}(x_\ell|x_m) := \prod_{k=\ell}^{m-1} p_{k|k+1}(x_k|x_{k+1})$.

Posterior sampling. Let g_0 be a nonnegative function on $\mathbb{R}^{d_x}$. When solving Bayesian inverse problems, g_0 is taken as the likelihood of the signal given the observation specified using the forward model (see the next section). Our objective is to sample from the posterior distribution

$$\pi_0(x_0) := g_0(x_0)\, p_0(x_0)/\mathcal{Z} \,, \tag{2.3}$$

where $\mathcal{Z} := \int g_0(x_0)\, p_0(x_0) \mathrm{d}x_0$ is the normalizing constant and the prior p_0 is the marginal of (2.1) w.r.t. x_0, in which case the posterior (2.3) can be expressed as

$$\pi_0(x_0) = \frac{1}{\mathcal{Z}} \int g_0(x_0) \prod_{k=0}^{n-1} p_{k|k+1}(x_k|x_{k+1})\, p_n(x_n)\, \mathrm{d}x_{1:n} \,.$$

Thus, Equation (2.3) can be interpreted as the marginal of a time-reversed FK (Feynman–Kac) model with a non-trivial potential only for $k = 0$; see [13] for a comprehensive introduction to FK models. In this work, we twist, without modifying the law of the FK model, the backward transitions $p_{k|k+1}$ by artificial positive potentials $(g_k)_{k=0}^n$, each being a function on $\mathbb{R}^{d_x}$, and write

$$\pi_0(x_0) = \frac{1}{\mathcal{Z}} \int g_n(x_n)\, p_n(x_n) \prod_{k=0}^{n-1} \frac{g_k(x_k)}{g_{k+1}(x_{k+1})}\, p_{k|k+1}(x_k|x_{k+1})\, \mathrm{d}x_{1:n} \,. \tag{2.4}$$

This allows the posterior of interest to be expressed as the time-zero marginal of an FK model with initial distribution p_n, Markov transition kernels $(p_{k|k+1})_{k=0}^{n-1}$, and $(g_k)_{k=0}^n$.

Recent works that aim to sample from the posterior (2.3) generally employ the FK representation (2.4). These studies, however, adopt varying auxiliary potentials [10, 47, 60, 4, 54, 59]. FK models can be effectively sampled using sequential Monte Carlo (SMC) methods; see, *e.g.*, [13, 9]. SMC methods sequentially propagate weighted samples, whose associated weighted empirical distributions target the flow of the FK marginal distributions. The effectiveness of this technique depends heavily on the choice of intermediate potentials $(g_k)_{k=1}^n$, as discussed in [54, 59, 7, 16]. However, SMC methods require a number of samples proportional and often exponential in the dimensionality of the problems hence limiting their application in these setups due to the resulting probabitive memory cost [2]. On the other hand, reducing the number of samples makes them vulnerable to mode collapse.

In the following, we will focus on a particular choice of potential functions $(g_k)_{k=1}^n$ for which the posterior π_0 can be expressed as the time-zero marginal distribution of a time-reversed Markov chain. The transition densities of this chain are obtained by twisting the transition densities of the generative model with the considered potential functions. More precisely, define, for all k,

the potentials $g_k^\star(x_k) := \int g_0(x_0)\, p_{0|k}(x_0|x_k)\, \mathrm{d}x_0$. Note that these potentials satisfy the recursion $g_{k+1}^\star(x_{k+1}) = \int g_k^\star(x_k)\, p_{k|k+1}(x_k|x_{k+1})\, \mathrm{d}x_k$. Builing upon that, define the Markov transitions

$$\pi_{k|k+1}(x_k|x_{k+1}) := \frac{g_k^\star(x_k)}{g_{k+1}^\star(x_{k+1})}\, p_{k|k+1}(x_k|x_{k+1}), \tag{2.5}$$

allowing the posterior (2.4) to be rewritten as

$$\pi_0(x_0) = \int \pi_n(x_n) \prod_{k=0}^{n-1} \pi_{k|k+1}(x_k|x_{k+1})\, \mathrm{d}x_{1:n}\,, \quad \pi_n(x_n) = g_n^\star(x_n) p_n(x_n)/\mathcal{Z}. \tag{2.6}$$

In other words, the distribution π_0 is the time-zero marginal of a Markov model with transition densities $(\pi_{k|k+1})_{k=n-1}^0$ and initial distribution π_n. According to this decomposition, a sample $X_0^\star$ from the posterior (2.3) can be obtained by sampling $X_n^\star \sim \pi_n$ and then, recursively sampling $X_k^\star \sim \pi_{k|k+1}(\cdot|X_{k+1}^\star)$ from $k = n-1$ till $k = 0$. In practice, however, neither the Markov transition densities $\pi_{k|k+1}$ nor the probability density function π_n are tractable. The main challenge in estimating $\pi_{k|k+1}$ stems essentially from the intractability of the potential $g_k^\star(x_k)$ as it involves computing an expectation under the high-cost sampling distribution $p_{0|k}(\cdot|x_k)$.

Recent works have focused on developing tractable approximations of $p_{0|k}(\cdot|x_k)$. For the *Diffusion Posterior Sampling* (DPS) algorithm [10], the point mass approximation $\delta_{\hat{x}_{0|k}(x_k)}$ of $p_{0|k}(\cdot|x_k)$ results in the estimate $\nabla_{x_k} \log g_0(\hat{x}_{0|k}(x_k))$ of $\nabla_{x_k} \log g_k^\star(x_k)$. Then, given a sample X_{k+1}, an approximate sample X_k from $\pi_{k|k+1}(\cdot|X_{k+1})$ is obtained by first sampling $\tilde{X}_k \sim p_{k|k+1}(\cdot|X_{k+1})$ and then setting

$$X_k = \tilde{X}_k + \zeta \nabla_{x_{k+1}} \log g_0(\hat{x}_{0|k+1}(x_{k+1}))|_{x_{k+1}=X_{k+1}}\,, \tag{2.7}$$

where $\zeta > 0$ is a tuning parameter. As noted in [48, 7, 4], the DPS updates (2.7) do not lead to an accurate approximation of the posterior π_0 even in the simplest examples; see also Section 4. Alternatively, [47] proposed the *Pseudoinverse-Guided Diffusion Model* (ΠGDM), which uses a Gaussian approximation of $p_{0|k}(\cdot|x_k)$ with mean $\hat{x}_{0|k}(x_k)$ and diagonal covariance matrix set to $(1-\alpha_k)I_{d_x}$, which corresponds to the covariance of $q_{0|k}(\cdot|x_k)$ if p_{data} had been a standard Gaussian; see [47, Appendix 1.3]. More recently, [17, 4] proposed to approximate the exact KL projection of $p_{0|k}(x_0|x_k)$ onto the space of Gaussian distributions by noting that both its mean and covariance matrix can be estimated using $\hat{x}_{0|k}(x_k)$ and its Jacobian matrix. We discuss in more depth the related works in Appendix B.

3 The DCPS algorithm

Smoothing the DPS approximation. The bias of the DPS updates (2.7) stems from the point mass approximation of the conditional distribution $p_{0|k}(\cdot|x_k)$. This approximation becomes more accurate as k tends to zero and is crude otherwise. We aim here to mitigate the resulting approximation errors. A core result that we leverage in this paper is that for any $(k,\ell) \in [\![0,n]\!]^2$ such that $\ell < k$, we can construct an estimate $\hat{p}_{\ell|k}(\cdot|x_k)$ of $p_{\ell|k}(\cdot|x_k)$ that bears a smaller approximation error than the estimate $\delta_{\hat{x}_{0|k}(x_k)}$ relatively to $p_{0|k}(\cdot|x_k)$. Formally, let $\hat{p}_{0|k}(\cdot|x_k)$ denote any approximation of $p_{0|k}(\cdot|x_k)$, such as that of the DPS or ΠGDM, and define the approximation of $p_{\ell|k}(\cdot|x_k)$

$$\hat{p}_{\ell|k}(x_\ell|x_k) := \int q_{\ell|0,k}(x_\ell|x_0,x_k)\hat{p}_{0|k}(x_0|x_k)\, \mathrm{d}x_0\,, \tag{3.1}$$

where $q_{\ell|0,k}(x_\ell|x_0,x_k)$ is defined in (A.4). We then have the following result.

Proposition 3.1 (informal). *Let $k \in [\![1,n]\!]$. For all $\ell \in [\![0,k-1]\!]$ and $x_k \in \mathbb{R}^{d_x}$,*

$$W_2(\hat{p}_{\ell|k}(\cdot|x_k), p_{\ell|k}(\cdot|x_k)) \leq \frac{\sqrt{\alpha_\ell}(1-\alpha_k/\alpha_\ell)}{(1-\alpha_k)} W_2(\hat{p}_{0|k}(\cdot|x_k), p_{0|k}(\cdot|x_k))\,. \tag{3.2}$$

The proof is postponed to Appendix A.3. Note that the ratio in the right-hand-side of (3.2) is less than 1 and decreases as ℓ increases. As an illustration, using the DPS approximation of $p_{0|k}(\cdot|x_k)$, we find that $\hat{p}_{\ell|k}(x_\ell|x_k) = q_{\ell|0,k}(x_\ell|\hat{x}_{0|k}(x_k),x_k)$ improves upon DPS in terms of approximation error.

This observation prompts to consider DPS-like approximations on shorter time intervals; instead of approximating expectations under $p_{0|k}(\cdot|x_k)$, such as the potential $g_k^\star(x_k)$, we should transform our initial sampling problem so that we only have to estimate expectations under $p_{\ell|k}(\cdot|x_k)$ for any ℓ such that the difference $k-\ell$ is small. This motivates the *blocking approach* introduced next.

Intermediate posteriors. We approach the original problem of sampling from π_0 via a series of simpler, *intermediate* posterior sampling problems of increasing difficulty. More precisely, let us consider the intermediate posteriors defined as

$$\pi_{k_\ell}(x_{k_\ell}) := g_{k_\ell}(x_{k_\ell}) p_{k_\ell}(x_{k_\ell}) / \mathcal{Z}_{k_\ell}, \quad \text{with} \quad \mathcal{Z}_{k_\ell} := \int g_{k_\ell}(x_{k_\ell}) p_{k_\ell}(x_{k_\ell}) \, \mathrm{d}x_{k_\ell}, \tag{3.3}$$

where $(g_{k_\ell})_{\ell=1}^L$ are potential functions designed by the user and $(k_\ell)_{\ell=0}^L$ is an increasing sequence in $[\![0, n]\!]$ such that $k_0 = 0$ and $k_L = n$. Here, L is typically much smaller than n. To obtain an approximate sample from $\pi_0 = \pi_{k_0}$, the DCPS algorithm recursively uses an approximate sample $X_{k_{\ell+1}}$ from $\pi_{k_{\ell+1}}$ to obtain an approximate sample X_{k_ℓ} from π_{k_ℓ}. Indeed, mirroring (2.6) it holds

$$\pi_{k_\ell}(x_{k_\ell}) = \int \pi^\ell_{k_{\ell+1}}(x_{k_{\ell+1}}) \prod_{m=k_\ell}^{k_{\ell+1}-1} \pi^\ell_{m|m+1}(x_m|x_{m+1}) \, \mathrm{d}x_{k_\ell+1:k_{\ell+1}} \,, \tag{3.4}$$

where for $m \in [\![k_\ell, k_{\ell+1} - 1]\!]$,

$$\pi^\ell_{k_{\ell+1}}(x_{k_{\ell+1}}) := g^{\ell,\star}_{k_{\ell+1}}(x_{k_{\ell+1}}) p_{k_{\ell+1}}(x_{k_{\ell+1}}) / \mathcal{Z}_{k_\ell} \,,$$
$$\pi^\ell_{m|m+1}(x_m|x_{m+1}) := g^{\ell,\star}_m(x_m) p_{m|m+1}(x_m|x_{m+1}) / g^{\ell,\star}_{m+1}(x_{m+1})$$

and for $m \in [\![k_\ell + 1, k_{\ell+1}]\!]$,

$$g^{\ell,\star}_m(x_m) := \int g_{k_\ell}(x_{k_\ell}) p_{k_\ell|m}(x_{k_\ell}|x_m) \, \mathrm{d}x_{k_\ell} \,. \tag{3.5}$$

We emphasize that the initial distribution $\pi^\ell_{k_{\ell+1}}$ in (3.4) is *different* from the posterior $\pi_{k_{\ell+1}}$ as the former involves the user-defined potential whereas the latter the intractable one. The main advantage of our approach lies in the fact that, unlike the potentials in the transition densities (2.5), which involve expectations under $p_{0|k}(\cdot|x_k)$, the potentials (3.5) are given by expectations under the distributions $p_{k_\ell|m}(\cdot|x_m)$, which are easier to approximate in the light of Proposition 3.1. In the sequel, we use this approximation for the estimation of the potentials (3.5); this yields approximate potentials

$$\hat{g}^{\ell,\star}_m(x_m) := \int g_{k_\ell}(x_{k_\ell}) \hat{p}_{k_\ell|m}(x_{k_\ell}|x_m) \, \mathrm{d}x_{k_\ell} \,, \quad m \in [\![k_\ell + 1, k_{\ell+1}]\!] \,, \tag{3.6}$$

which serve as a substitute for the intractable $g^{\ell,\star}_m$. Let us now summarize how our algorithm works. Starting from a sample $X_{k_{\ell+1}}$, which is approximately distributed according to $\pi_{k_{\ell+1}}$, the next sample X_{k_ℓ} is generated in the next two steps:

1. Perform Langevin Monte Carlo steps initialized at $X_{k_{\ell+1}}$ and targeting $\pi^\ell_{k_{\ell+1}}$, yielding $X^\ell_{k_{\ell+1}}$.
2. Simulate a Markov chain $(X_j)_{j=k_{\ell+1}}^{k_\ell}$ initialized with $X_{k_{\ell+1}} = X^\ell_{k_{\ell+1}}$ and whose transition from X_{j+1} to X_j is the minimizer of
$$\mathsf{KL}(\lambda^\varphi_{j|j+1}(\cdot|X_{j+1}) \parallel \pi^\ell_{j|j+1}(\cdot|X_{j+1})), \tag{3.7}$$
where $\lambda^\varphi_{j|j+1}$ is a mean-field Gaussian approximation with parameters $\varphi := (\hat{\mu}, \hat{\sigma}) \in \mathbb{R}^{d_x} \times \mathbb{R}^{d_x}_{>0}$. X_j is drawn from $\lambda^{\varphi_j(X_{j+1})}_{j|j+1}(\cdot|X_{j+1})$, where $\varphi_j(X_{j+1})$ is a minimizer of the proxy of (3.7).

In the following, we elaborate more on Step 1 and Step 2 and discuss the choice of the intermediate potentials. The pseudo-code of the DCPS algorithm is in Algorithm 1.

Sampling the initial distribution. In order to perform **Step 1**, we use the discretized Langevin dynamics [38] with the estimate $\nabla \log \hat{g}^{\ell,\star}_{k_{\ell+1}} + \hat{s}_{k_{\ell+1}}$ of the score $\nabla \log \pi^\ell_{k_{\ell+1}}$. This estimate results from the use of $\hat{s}_{k_{\ell+1}}$ as an approximation of $\nabla \log p_{k_{\ell+1}}$ in combination with the approximate potential (3.6). We then obtain the approximate sample $X^\ell_{k_{\ell+1}}$ of $\pi_{k_{\ell+1}}$ by running M steps of the tamed unadjusted Langevin (TULA) scheme [5]; see Algorithm 1. Here, the intractability of the involved densities hinder the usage of the Metropolis-Hastings corrections to reduce the inherent bias of the Langevin algorithm.

Sampling the transitions. We now turn to **Step 2**. Given X_{j+1}, we optimize the following estimate of Equation (3.7), where we simply replace $g_j^{\ell,\star}$ by the approximation (3.6):

$$-\int \log \hat{g}_j^{\ell,\star}(x_j)\lambda^{\varphi}_{j|j+1}(x_j|x_{j+1})\,\mathrm{d}x_j + \mathsf{KL}(\lambda^{\varphi}_{j|j+1}(\cdot|x_{j+1}) \parallel p_{j|j+1}(\cdot|x_{j+1}))\,.$$

Letting $\lambda^{\varphi}_{j|j+1}(x_j|x_{j+1}) = \mathrm{N}(x_j; \hat{\mu}_j, \mathrm{diag}(\mathrm{e}^{\hat{v}_j}))$, where the variational parameters $\hat{\mu}_j, \hat{v}_j$ are in $\mathbb{R}_{d_x}$, the previous estimate yields the objective

$$\mathcal{L}_j(\hat{\mu}_j, \hat{v}_j; x_{j+1}) := -\mathbb{E}\big[\log \hat{g}_j^{\ell,\star}(\hat{\mu}_j + \mathrm{e}^{\hat{v}_j/2}Z)\big] + \frac{\|\hat{\mu}_j - \mu_{j|j+1}(x_{j+1})\|^2}{2\sigma^2_{j|j+1}} - \frac{1}{2}\sum_{i=1}^{d_x}\left(\hat{v}_{j,i} - \frac{\mathrm{e}^{\hat{v}_{j,i}}}{\sigma^2_{j|j+1}}\right), \quad (3.8)$$

where Z is d_x-dimensional standard Gaussian and $\mu_{j|j+1}(x_{j+1})$ is the mean of (2.2). Note here that we have used the reparameterization trick [26] and the closed-form expression of the KL divergence between two multivariate Gaussian distributions. We optimize the previous objective using a few steps of SGD by estimating the first term on the r.h.s. with a single sample as in [26]. For each $j \in [\![k_\ell, k_{\ell+1} - 1]\!]$, we use $\mu_{j|j+1}$ and $\log \sigma^2_{j|j+1}$ as initialization for $\hat{\mu}_j$ and $\hat{v}_j$.

Intermediate potentials. Here, we give general guidelines to choose the user-defined potentials $(g_{k_\ell})_{\ell=1}^L$. Our design choice is to rescale the input and then anneal the initial potential g_0. Therefore, we suggest

$$g_{k_\ell}(x) = g_0(\tfrac{x}{\beta_{k_\ell}})^{\gamma_{k_\ell}}\,, \quad (3.9)$$

where $\gamma_{k_\ell}, \beta_{k_\ell} > 0$ are tunable paramerters. This design choice is inspired from the tempering sampling scheme [33] which uses the principle of progressively moving an intial distribution to the targeted one. We provide some examples in the case of Bayesian inverse problems where the unobserved signal and the observation are modelled jointly as a realization of $(X, Y) \sim p(y|x)p_0(x)$, where $p(y|x)$ is the conditional density of Y given $X = x$. In this case, the posterior π_0 of X given $Y = y$ is given by (2.3) with $g_0(x) = p(y|x)$.

Linear inverse problems with Gaussian noise. In this case, $g_0(x) = \mathrm{N}(y; Ax, \sigma_y^2 I_{d_y})$, where $A \in \mathbb{R}^{d_y \times d_x}$. Popular applications in image processing include super-resolution, inpainting, outpainting, and deblurring. We use (3.9) with $(\beta_{k_\ell}, \gamma_{k_\ell}) = (\sqrt{\alpha_{k_\ell}}, \alpha_{k_\ell})$,

$$g_{k_\ell}(x) = \mathrm{N}(\sqrt{\alpha_{k_\ell}}y; Ax, \sigma_y^2 I_{d_y})\,, \quad (3.10)$$

which corresponds to the likelihood of x given the *pseudo observation* $\sqrt{\alpha_{k_\ell}}y$ under the same linear observation model that defines g_0. This choice of g_{k_ℓ} enables exact computation of (3.6) and allows information on the observation y to be taken into account early in the denoising process.

Low-count (or shot-noise) Poisson denoising. In a Poisson model for an image, the grey levels of the image pixels are modelled as Poisson-distributed random variables. More specifically, let $A \in \mathbb{R}^{d_y \times d_x}$ be a matrix with nonnegative entries and $x \in [0, 255]^{C \times H \times W}$, where C is the number of channels and H the height and W the width. For every $i \in [\![1, d_y]\!]$, Y_i is Poisson-distributed with mean $(Ax)_i$, and the likelihood of x given the observation is therefore given by $x \mapsto \prod_{j=1}^{d_y} (\lambda Ax)_j^{y_j} \mathrm{e}^{-(\lambda Ax)_j}/y_j!$ where $\lambda > 0$ is the rate. Following [10] we consider as likelihood its normal approximation, *i.e.* $g_0 = \prod_{j=1}^{d_y} \mathrm{N}(y_j; \lambda(Ax)_j, y_j)$. This model is relevant for many tasks such as low-count photon imaging and computed tomography (CT) reconstruction [35, 39, 31]. We use (3.9) with $\beta_{k_\ell} = \gamma_{k_\ell} = \sqrt{\alpha_{k_\ell}}$:

$$g_{k_\ell}(x) = \prod_{j=1}^{d_y} \mathrm{N}(\sqrt{\alpha_{k_\ell}}y_j; \lambda(Ax)_j, \sqrt{\alpha_{k_\ell}}y_j)\,. \quad (3.11)$$

JPEG dequantization. JPEG [57] is a ubiquitous method for lossy compression of images. Use h_q to denote the JPEG encoding function with *quality factor* $q \in [\![0, 100]\!]$, where a small q is associated with high compression. Denote by $h_q^\dagger$ the JPEG decoding function that returns an image in RGB space with a certain loss of detail, depending on the degree of compression q, compared to the original image. Since we require the potential to be differentiable almost everywhere, we use the differentiable approximation of JPEG developed in [44], which replaces the rounding function used in the quantization matrix with a differentiable approximation that has non-zero derivatives almost everywhere. In this case, $g_0(x) = \mathrm{N}(h_q^\dagger(y); h_q^\dagger(h_q(x)), \sigma_y^2 I_{d_y})$, where y is in YCbCr space. Combining this with Equation (3.9) with $(\beta_{k_\ell}, \gamma_{k_\ell}) = (\alpha_{k_\ell}, \alpha_{k_\ell})$ and assuming that the composition $h_q^\dagger \circ h_q$ is a homogenious map, the intermediate potentials are $g_{k_\ell}(x) = \mathrm{N}(\sqrt{\alpha_{k_\ell}}\, h_q^\dagger(y); h_q^\dagger(h_q(x)), \sigma_y^2 I_{d_x})$.

4 Experiments

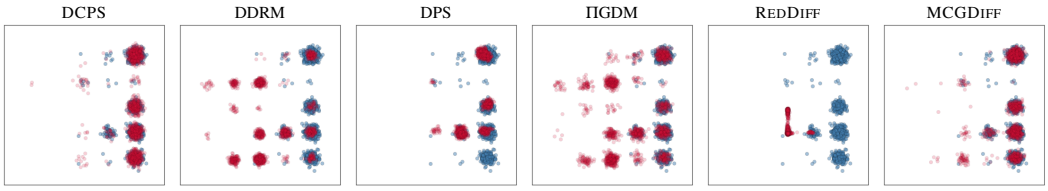

Figure 1: First two dimensions of samples (in red) from each algorithm on the 25 component Gaussian mixture posterior sampling problem with $(d_x, d_y) = (100, 1)$. The true posterior samples are given in blue.

In this section, we demonstrate the performance of DCPS and compare it with DPS [10], ΠGDM [47], DDRM [24], REDDIFF [32], and MCGDIFF [7] on several Bayesian inverse problems. We also benchmark our algorithm against DIFFPIR [62], DDNM [58], FPS [16], and SDA [42] but we defer the results to the Appendix C.5.

First, we consider a simple toy experiment in which the posterior distribution is available in closed form. Next, we apply our algorithm to superresolution (SR $4\times$ and $16\times$), inpainting and outpainting tasks with Gaussian and Poisson noise, and JPEG dequantization. For these imaging experiments, we use the `FFHQ256` [23] and `ImageNet256` [14] datasets and the publicly available pre-trained models of [8] and [15]. Finally, we benchmark our method on a trajectory inpainting task using the pedestrian dataset `UCY` for which we have trained a Diffusion model. All details can be found in Appendix C.1.

Gaussian mixture. We first evaluate the accuracy of DCPS on a linear inverse problem with a Gaussian mixture (GM) prior, for which the posterior can be explicitly computed: it is also a Gaussian mixture whose means, covariance matrices, and weights are in a closed form; see Appendix C.2. In this case, the predictor $\hat{x}_{0|k}^{\theta^*}$ is available in a closed form; see Appendix C.2 for more details. We consider a Gaussian mixture prior with 25 components in dimensions $d_x = 10$ and $d_x = 100$. The potential is $g_0(x) = \mathrm{N}(y; Ax, \sigma_y^2 I_{d_y})$ with $d_y = 1$ and A is a $1 \times d_x$ vector. The results are averaged over 30 randomly generated replicates of the measurement model (y, A, σ_y^2) and the mixture weights. Then, for each pair of prior distribution and measurement model, we generate $N_s = 2000$ samples with each algorithm and compare them with N_s samples from the true posterior distribution using the sliced Wasserstein (SW) distance. For DCPS, we used $L = 3$ blocks and $K = 2$ gradient steps, respectively, and compared two configurations, denoted by DCPS_{50} and DCPS_{500}, of the algorithm with $M = 50$ and $M = 500$ Langevin steps, respectively. See Algorithm 1. The results are reported in Table 1. It is worthwhile to note that DCPS outperforms all baselines except for MCGDIFF. However, by increasing the number of Langevin steps, its performance closely matches that of MCGDIFF.

Table 1: 95% confidence interval for the SW on the GM experiment.

	$d_x = 10, d_y = 1$	$d_x = 100, d_y = 1$
DCPS_{50}	2.91 ± 0.74	4.04 ± 1.00
DCPS_{500}	$\mathbf{2.19} \pm 0.68$	$\underline{3.29} \pm 0.95$
DPS	5.80 ± 0.75	5.68 ± 0.73
DDRM	3.77 ± 0.96	5.70 ± 0.78
ΠGDM	4.23 ± 0.90	4.61 ± 0.68
REDDIFF	6.36 ± 1.27	7.47 ± 0.87
MCGDIFF	$\underline{2.28} \pm 0.75$	$\mathbf{2.83} \pm 0.71$

Imaging experiment. Table 2 reports the results for the linear inverse problems with Gaussian noise with two noise variance levels $\sigma_y = 0.05$ and $\sigma_y = 0.3$, Table 3 for the JPEG dequantization problem with $\sigma_y = 10^{-3}$, QF $\in \{2, 8\}$, and Table 6 for the Poisson denoising task with rate $\lambda = 0.1$. For all tasks and datasets, we use the same parameters for DCPS and therefore do not perform any task or dataset-specific tuning. We use $L = 3$, $K = 2$ gradient steps, and $M = 5$ Langevin steps. To ensure a fair comparison with DPS and ΠGDM we use 300 DDPM steps for DCPS and 1000 steps for both DPS and ΠGDM, which ensures that all the algorithms have the same runtime and memory footprint; see Table 4. For MCGDIFF, which has a large memory requirement, we use $N = 32$ particles in the SMC sampling step and then randomly draw one sample from the resulting particle approximation of the posterior. Finally, for DDRM we use 200 diffusion steps and for REDDIFF we use 1000 gradient steps and the parameters recommended in the original paper. We provide the implementation details for all algorithms in Appendix C.1.

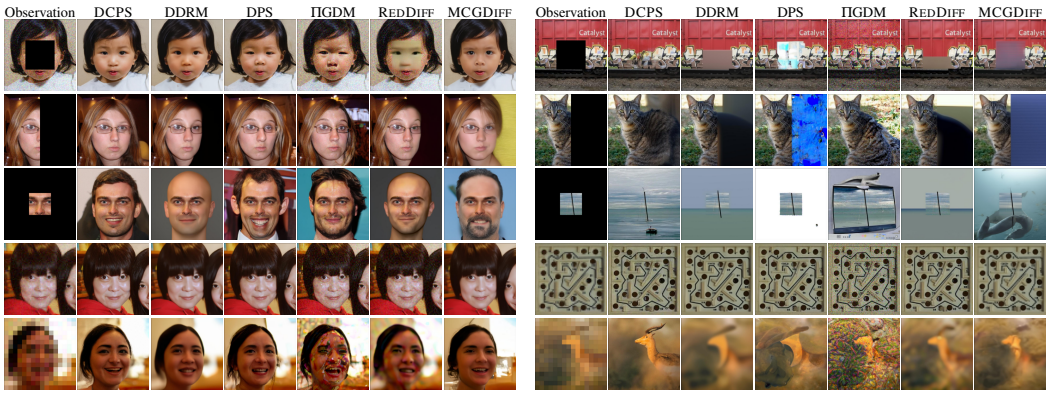

Figure 2: Sample images for inpainting with center, half, expand masks and for Super Resolution with $4\times$ and $16\times$ factors. On the left: `FFHQ` dataset and on the right `ImageNet` dataset.

For the JPEG dequantization task, we use $\sigma_y = 10^{-3}$ and $\lambda = 0.1$. We only benchmark our method against ΠGDM and REDDIFF, since MCGDIFF and DDRM do not handle non-linear inverse problems. We did not include DPS in our benchmark because we have not managed to find a suitable choice of hyperparameters to achieve reasonable results. Finally, for the Poisson-shot noise case, we compare against DPS. We use the step size for super-resolution recommended in the original paper [see 10, Appendix D.1], and found, via a grid search, that the same value is also effective for the other tasks.

Table 2: Mean LPIPS value on different tasks. Lower is better.

Dataset / σ_y	Task	DCPS	DDRM	DPS	ΠGDM	REDDIFF	MCGDIFF
`FFHQ` / 0.05	Half	**0.20**	0.25	0.24	0.26	0.28	0.36
	Center	**0.05**	0.06	0.07	0.19	0.12	0.24
	SR $4\times$	**0.09**	0.18	0.09	0.33	0.36	0.15
	SR $16\times$	**0.23**	0.36	0.24	0.44	0.51	0.32
`FFHQ` / 0.3	Half	**0.25**	0.30	0.31	0.64	0.76	0.80
	Center	**0.10**	0.13	0.11	0.62	0.75	0.55
	SR $4\times$	0.21	0.26	**0.19**	0.77	0.77	0.65
	SR $16\times$	**0.35**	0.41	0.43	0.64	0.74	0.52
`ImageNet` / 0.05	Half	**0.35**	0.40	0.44	0.38	0.44	0.83
	Center	0.18	**0.14**	0.31	0.29	0.22	0.45
	SR $4\times$	**0.24**	0.38	0.41	0.78	0.56	1.32
	SR $16\times$	**0.44**	0.72	0.50	0.60	0.83	1.33
`ImageNet` / 0.3	Half	**0.40**	0.46	0.48	0.82	0.76	0.86
	Center	**0.24**	0.25	0.40	0.68	0.71	0.47
	SR $4\times$	**0.43**	0.50	0.47	0.87	0.83	1.31
	SR $16\times$	0.72	0.77	**0.57**	0.72	0.92	0.67
Average		**0.28**	0.35	0.32	0.57	0.60	0.67

Evaluation. As shown in Table 2, DCPS outperforms the other baselines on 13 out of 16 tasks and has the best average performance. In particular, it compares favorably with ΠGDM and DPS, its closest competitors, while exhibiting the same runtime and memory requirements; see Table 4, where we give the average runtime and memory usage for each algorithm. The memory consumption is measured by how many samples each algorithm can generate in parallel on a single 48GB L40S NVIDIA GPU for the Diffusion model trained on `FFHQ` [15]. We emphasize that DCPS is more robust to larger noise levels than ΠGDM and REDDIFF, as evidenced by the large increase in the LPIPS value for these algorithms in the case $\sigma_y = 0.3$. On the JPEG dequantization task (Table 3), DCPS also shows better performance than these algorithms and even more so for the high compression level (QF $= 2$). On the Poisson-shot noise tasks, DCPS outperforms DPS by a significant margin; see Table 6. Finally, we display various reconstructions obtained with each algorithm. More specifically, we have generated 4 samples each, with the same seed. Figure 2 displays the first sample and the remaining ones are deferred to Appendix D. For MCGDIFF we show 4 random samples of the same particle filter. Due to the collapse of the particle filter in very large dimensions [2], they are all similar. Surprisingly, the samples produced by DDRM and REDDIFF for the outpainting tasks also show striking similarities, although the samples have been drawn independently.

Table 3: Mean LPIPS value on JPEG dequantization.

Dataset	Task	DCPS	ΠGDM	REDDIFF
`FFHQ`	QF $= 2$	**0.20**	0.37	0.32
	QF $= 8$	**0.08**	0.15	0.18
`ImageNet`	QF $= 2$	**0.44**	0.93	0.50
	QF $= 8$	**0.24**	0.95	0.31

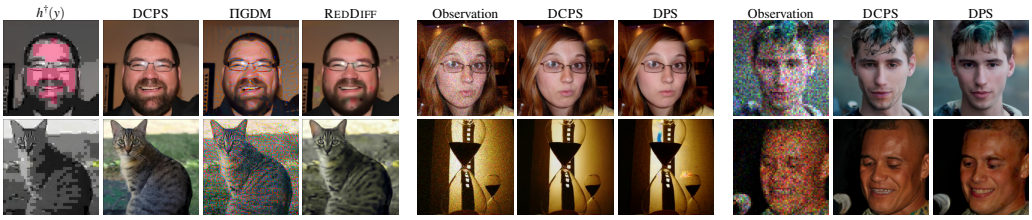

Figure 3: Left: JPEG dequantization with QF = 2. Middle: Poisson denoising. Right: SR 4× Poisson denoising.

Trajectory prediction. We evaluate our algorithm on the UCY dataset consisting of pedestrian trajectories, encoded as 2D time series with 20 time steps [27, 29, 18, 30]. We pre-train a trajectory model on this dataset and then use it for trajectory reconstruction tasks. The model architecture and implementation are detailed in Appendix C.4. We focus on the completion of trajectories where only a few timesteps are observed. The missing steps are filled in based on the observations and the pre-trained prior model, similar to the inpainting task in the previous section. We use MCGDIFF with 5000 particles to obtain approximate samples from the posterior. Indeed, as the dimension of the observation space is low ($d_x = 40$) and MCGDIFF is asymptotically exact as the number of particles tends to infinity, it yields an accurate approximation of the posterior; see [7, Proposition 2.1]. Then, we compute the ℓ_2 distance between the median, quantile 25, and quantile 75 of the MCGDIFF samples and the reconstructions of each algorithm. We report these results in Table 5. Finally, in Figure 4 we illustrate the reconstructed trajectories on a specific trajectory completion problem.

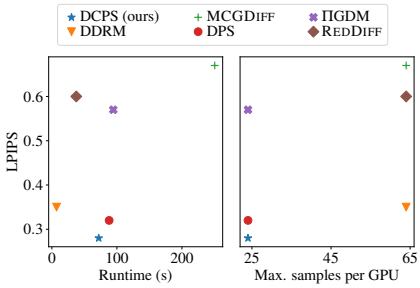

Table 4: LPIPS metric against the runtime and memory cost of the algorithms.

Table 5: ℓ_2 distance quantiles with MCGDIFF as reference.

	$\sigma_y = 0.005$			$\sigma_y = 0.01$		
	$q50$	$q25$	$q75$	$q50$	$q25$	$q75$
DCPS	**1.31**	**1.33**	**1.47**	**1.33**	**1.42**	**1.42**
DPS	<u>1.34</u>	1.40	1.61	<u>1.36</u>	1.48	1.52
DDRM	1.48	1.46	1.61	1.59	1.62	1.61
ΠGDM	1.36	<u>1.35</u>	**1.47**	1.37	<u>1.43</u>	**1.42**
REDDIFF	1.67	1.57	1.82	1.56	1.54	1.65

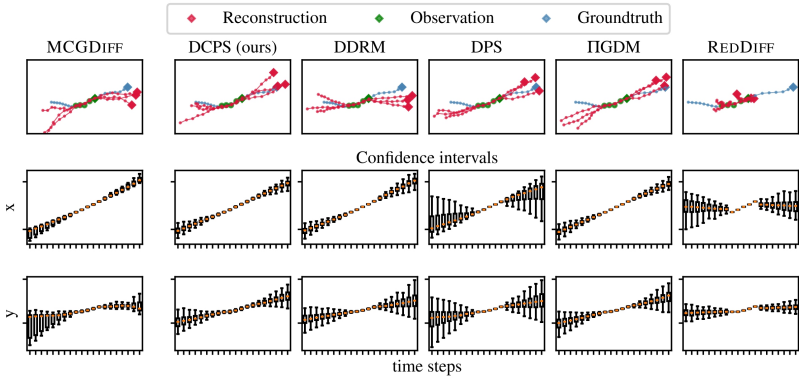

Figure 4: Trajectory completion where only the middle part of the trajectory is observed. The figures in the 1st row display 3 reconstructions per algorithm. The 2nd and 3rd rows show confidence intervals across different time steps. The *Groundtruth* is a trajectory taken from the UCY dataset.

5 Conclusion.

In this paper, we introduce DCPS to handle Bayesian linear inverse problems with DDM priors without the need for problem-specific additional training. Our divide-and-conquer strategy helps to reduce the approximation error of existing approaches, and our variational framework provides a

principled method for estimating the backward kernels. DCPS applies to various relevant inverse problems and is competitive with existing methods.

Limitations and future directions. Our method has some limitations that shed light on opportunities for further development and refinement. First, the intermediate potentials that we considered were specifically designed for each problem, meaning our method is not universally applicable to all inverse problems. For instance, our approach can not be applied to for linear inverse problems using latent diffusion models [41] since there is no clear choice of intermediate potentials. Therefore, in our opinion, deriving a learning procedure that is capable to automatically design effective intermediate potentials applicable to any g_0 is an important research direction. Moreover, there is an aspect of the choice of the intermediate potentials and the number of blocks L that remains to be understood properly. Indeed, while our backward approximations reduce the local approximation errors w.r.t. DPS and ΠGDM; nonetheless DCPS requires appropriate intermediate potentials in order to perform well. DCPS can still provide decent performance with *irrelevant* intermediate potentials as long as the number of Langevin steps, in-between the blocks, is large enough. Finally, although our method provides decent results with the same computational cost as DPS and ΠGDM, it remains slower than REDDIFF and DDRM which which do not compute vector-jacobian product over the denoiser. Therefore, overcoming this bottleneck when optimizing the KL objective would be a significant improvement for our method.

Acknowledgments. The work of Y.J. and B.M. has been supported by Technology Innovation Institute (TII), project Fed2Learn. 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.

References

[1] Georgios Batzolis, Jan Stanczuk, Carola-Bibiane Schönlieb, and Christian Etmann. Conditional image generation with score-based diffusion models. *arXiv preprint arXiv:2111.13606*, 2021.

[2] P. Bickel, B. Li, and T. Bengtsson. Sharp failure rates for the bootstrap particle filter in high dimensions. In B. Clarke and S. Ghosal, editors, *Pushing the Limits of Contemporary Statistics: Contributions in Honor of Jayanta K. Ghosh*, pages 318–329. Institute of Mathematical Statistics, 2008.

[3] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[4] Benjamin Boys, Mark Girolami, Jakiw Pidstrigach, Sebastian Reich, Alan Mosca, and O Deniz Akyildiz. Tweedie moment projected diffusions for inverse problems. *arXiv preprint arXiv:2310.06721*, 2023.

[5] Nicolas Brosse, Alain Durmus, Éric Moulines, and Sotirios Sabanis. The tamed unadjusted langevin algorithm. *Stochastic Processes and their Applications*, 129(10):3638–3663, 2019.

[6] Daniela Calvetti and Erkki Somersalo. Inverse problems: From regularization to Bayesian inference. *Wiley Interdisciplinary Reviews: Computational Statistics*, 10(3):e1427, 2018.

[7] Gabriel Cardoso, Yazid Janati, Eric Moulines, and Sylvain Le Corff. Monte carlo guided denoising diffusion models for bayesian linear inverse problems. In *The Twelfth International Conference on Learning Representations*, 2024.

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

[9] Nicolas Chopin, Omiros Papaspiliopoulos, et al. *An introduction to sequential Monte Carlo*, volume 4. Springer, 2020.

[10] Hyungjin Chung, Jeongsol Kim, Michael Thompson Mccann, Marc Louis Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. In *The Eleventh International Conference on Learning Representations*, 2023.

[11] Hyungjin Chung, Byeongsu Sim, and Jong Chul Ye. Come-closer-diffuse-faster: Accelerating conditional diffusion models for inverse problems through stochastic contraction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12413–12422, 2022.

[12] Masoumeh Dashti and Andrew M. Stuart. *The Bayesian Approach to Inverse Problems*, pages 311–428. Springer International Publishing, Cham, 2017.

[13] Pierre Del Moral. Feynman-kac formulae. In *Feynman-Kac Formulae*, pages 47–93. Springer, 2004.

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

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

[16] Zehao Dou and Yang Song. Diffusion posterior sampling for linear inverse problem solving: A filtering perspective. In *The Twelfth International Conference on Learning Representations*, 2024.

[17] Marc Anton Finzi, Anudhyan Boral, Andrew Gordon Wilson, Fei Sha, and Leonardo Zepeda-Núñez. User-defined event sampling and uncertainty quantification in diffusion models for physical dynamical systems. In *International Conference on Machine Learning*, pages 10136–10152. PMLR, 2023.

[18] Tianpei Gu, Guangyi Chen, Junlong Li, Chunze Lin, Yongming Rao, Jie Zhou, and Jiwen Lu. Stochastic trajectory prediction via motion indeterminacy diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17113–17122, 2022.

[19] Bichuan Guo, Yuxing Han, and Jiangtao Wen. Agem: Solving linear inverse problems via deep priors and sampling. *Advances in Neural Information Processing Systems*, 32, 2019.

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

[21] Aapo Hyvärinen. Some extensions of score matching. *Computational statistics & data analysis*, 51(5):2499–2512, 2007.

[22] Bowen Jing, Gabriele Corso, Renato Berlinghieri, and Tommi Jaakkola. Subspace diffusion generative models. In *European Conference on Computer Vision*, pages 274–289. Springer, 2022.

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

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

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

[26] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[27] Alon Lerner, Yiorgos Chrysanthou, and Dani Lischinski. Crowds by example. In *Computer graphics forum*, 2007.

[28] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[29] Karttikeya Mangalam, Yang An, Harshayu Girase, and Jitendra Malik. From goals, waypoints & paths to long term human trajectory forecasting. In *IEEE/CVF*, 2021.

[30] Weibo Mao, Chenxin Xu, Qi Zhu, Siheng Chen, and Yanfeng Wang. Leapfrog diffusion model for stochastic trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5517–5526, 2023.

[31] Willem Marais and Rebecca Willett. Proximal-gradient methods for poisson image reconstruction with bm3d-based regularization. In *2017 IEEE 7th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, pages 1–5. IEEE, 2017.

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

[33] Radford M Neal. Annealed importance sampling. *Statistics and computing*, 11:125–139, 2001.

[34] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pages 8162–8171. PMLR, 2021.

[35] Robert D Nowak and Eric D Kolaczyk. A statistical multiscale framework for poisson inverse problems. *IEEE Transactions on Information Theory*, 46(5):1811–1825, 2000.

[36] Xingang Pan, Xiaohang Zhan, Bo Dai, Dahua Lin, Chen Change Loy, and Ping Luo. Exploiting deep generative prior for versatile image restoration and manipulation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):7474–7489, 2021.

[37] Michael K Pitt and Neil Shephard. Filtering via simulation: Auxiliary particle filters. *J. Amer. Statist. Assoc.*, 94(446):590–599, 1999.

[38] Gareth O. Roberts and Richard L. Tweedie. Geometric convergence and central limit theorems for multidimensional Hastings and Metropolis algorithms. *Biometrika*, 83:95–110, 1996.

[39] Isabel Rodrigues, Joao Sanches, and Jose Bioucas-Dias. Denoising of medical images corrupted by poisson noise. In *2008 15th IEEE international conference on image processing*, pages 1756–1759. IEEE, 2008.

[40] Yaniv Romano, Michael Elad, and Peyman Milanfar. The little engine that could: Regularization by denoising (red). *SIAM Journal on Imaging Sciences*, 10(4):1804–1844, 2017.

[41] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

[42] Franccois Rozet and Gilles Louppe. Score-based data assimilation. *Advances in Neural Information Processing Systems*, 36:40521–40541, 2023.

[43] Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):4713–4726, 2022.

[44] Richard Shin and Dawn Song. Jpeg-resistant adversarial images. In *NIPS 2017 workshop on machine learning and computer security*, volume 1, page 8, 2017.

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

[46] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.

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

[48] Jiaming Song, Qinsheng Zhang, Hongxu Yin, Morteza Mardani, Ming-Yu Liu, Jan Kautz, Yongxin Chen, and Arash Vahdat. Loss-guided diffusion models for plug-and-play controllable generation. In *International Conference on Machine Learning*, pages 32483–32498. PMLR, 2023.

[49] Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. Maximum likelihood training of score-based diffusion models. *Advances in Neural Information Processing Systems*, 34:1415–1428, 2021.

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

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

[52] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

[53] Andrew M Stuart. Inverse problems: a Bayesian perspective. *Acta numerica*, 19:451–559, 2010.

[54] Brian L. Trippe, Jason Yim, Doug Tischer, David Baker, Tamara Broderick, Regina Barzilay, and Tommi S. Jaakkola. Diffusion probabilistic modeling of protein backbones in 3d for the motif-scaffolding problem. In *The Eleventh International Conference on Learning Representations*, 2023.

[55] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9446–9454, 2018.

[56] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[57] Gregory K Wallace. The jpeg still picture compression standard. *IEEE transactions on consumer electronics*, 38(1):xviii–xxxiv, 1992.

[58] Yinhuai Wang, Jiwen Yu, and Jian Zhang. Zero-shot image restoration using denoising diffusion null-space model. In *The Eleventh International Conference on Learning Representations*, 2023.

[59] Luhuan Wu, Brian L. Trippe, Christian A Naesseth, John Patrick Cunningham, and David Blei. Practical and asymptotically exact conditional sampling in diffusion models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[60] Guanhua Zhang, Jiabao Ji, Yang Zhang, Mo Yu, Tommi Jaakkola, and Shiyu Chang. Towards coherent image inpainting using denoising diffusion implicit models. In *International Conference on Machine Learning*, pages 41164–41193. PMLR, 2023.

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

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

A Methodology details

A.1 Denoising Diffusion models

DDMs learn a sequence $(\hat{x}^\theta_{0|t})_{t=1}^T$ of denoisers by minimizing, using SGD, the objective

$$\sum_{t=1}^{T} w_t \mathbb{E}\left[\|\epsilon_t - \hat{\epsilon}^\theta_t(\sqrt{\alpha_t}X_0 + \sqrt{1-\alpha_t}\epsilon_t)\|^2\right] \tag{A.1}$$

w.r.t. the neural network parameter θ, where $(\epsilon_t)_{t=1}^T$ are i.i.d. standard normal vectors and $(w_t)_{t=1}^T$ are some nonnegative weights. We denote by $\theta^\star$ an estimator of the minimizer of the previous loss. Having access to $\theta^\star$, we can define a generative model for p_{data}. Let $(t_k)_{k=0}^n$ be an increasing sequence of time instants in $[\![0, T]\!]$ with $t_0 = 0$. We assume that t_n is large enough so that q_{t_n} is approximately multivariate standard normal. For convenience, we assign the index k to any quantity depending on t_k; *e.g.*, we denote p_{t_k} by p_k. For $(j, k) \in [\![1, n-1]\!]^2$ such that $j < k$, define

$$\mu_{j|0,k}(x_0, x_k) := \frac{\sqrt{\alpha_j}(1-\alpha_k/\alpha_j)}{1-\alpha_k}x_0 + \frac{\sqrt{\alpha_k/\alpha_j}(1-\alpha_j)}{1-\alpha_k}x_k\,, \tag{A.2}$$

$$\sigma^2_{j|k} := \frac{(1-\alpha_j)(1-\alpha_k/\alpha_j)}{1-\alpha_k}\,. \tag{A.3}$$

Then the bridge kernel

$$q_{j|0,k}(x_j|x_0, x_k) = q_{j|0}(x_j|x_0)q_{k|j}(x_k|x_j)/q_{k|0}(x_k|x_0) \tag{A.4}$$

is a Gaussian distribution with mean $\mu_{j|0,k}(x_0, x_k)$ and covariance $\sigma^2_{j|k}I_{d_x}$. DDPM [20] posits the following variational approximation

$$p^\theta_{0:n}(x_{0:n}) = p_n(x_n)\prod_{k=0}^{n-1} p^\theta_{k|k+1}(x_k|x_{k+1})\,,$$

where $p^\theta_{k|k+1}(x_k|x_{k+1}) = q_{k|0,k+1}(x_k|\hat{x}^\theta_{0|k+1}(x_{k+1}), x_{k+1})$ and $p^\theta_{0|1}(\cdot|x_1) = \delta_{\hat{x}^\theta_{0|1}(x_1)}$. An efficient generative model is then obtained by plugging in the parameter $\theta^\star$.

A.2 Further details on DCPS

In this section we provide further details on **Steps 1** and **2** detailed in the main paper. The complete algorithm is given in Algorithm 1.

Tamed unadjusted Langevin. For the tamed unadjusted Langevin steps we simulate the Markov chain $(\tilde{X}_j)_{j=0}^M$ where

$$\tilde{X}_{j+1} = \tilde{X}_j + \gamma G^\ell_\gamma(\tilde{X}_j) + \sqrt{2\gamma}Z_j\,, \quad \tilde{X}_0 = X_\ell + 1\,, \tag{A.5}$$

and $(Z_j)_{j=0}^{M-1}$ are i.i.d. d_x-dimensional standard normal, $X_\ell + 1$ is an approximate sample from $\pi_{\ell+1}$ obtained from the previous iteration of the algorithm, and for all $x \in \mathbb{R}^{d_x}$ and $\gamma > 0$,

$$G^\ell_\gamma(x) := \frac{\nabla \log \hat{g}^{\ell,\star}_{\ell+1}(x) + \hat{s}_{\ell+1}(x)}{1 + \gamma\|\nabla \log \hat{g}^{\ell,\star}_{\ell+1}(x) + \hat{s}_{\ell+1}(x)\|}\,. \tag{A.6}$$

We then set $X^\ell_{\ell+1} := \tilde{X}_M$, which serves as an initialization of the Markov chain in **Step 2**.

Potential computation. In order to perform the tamed Langevin steps and to optimize the variational approximation using the criterion (3.8), it is crucial to be able to compute exactly the potential (3.6). The optimal potentials we have proposed for both linear inverse problems with Gaussian noise (3.10) and low-count Poisson denoising (3.11) (for $\ell > 0$) are available in a closed form:

$$\hat{g}^{\ell,\star}_j(x_j) = \mathrm{N}(\sqrt{\alpha_\ell}\,y, A\mu_{\ell|j}(x_j), \Sigma^\ell_j)\,, \tag{A.7}$$

where

$$\Sigma_j^\ell = \sigma_{\ell|j}^2 AA^\intercal + \sigma_y^2 I_{d_y}\,, \qquad \text{(Linear inverse problem)}$$

$$\Sigma_j^\ell = \sigma_{\ell|j}^2 AA^\intercal + \sqrt{\alpha_\ell}\mathrm{diag}(y)\,, \quad \ell > 0\,, \qquad \text{(Poisson-shot noise)}$$

$\mu_{\ell|j}(x_j) := \mu_{\ell|0,j}(\hat{x}_{0|j}(x_j), x_j)$, and $\sigma_{\ell|j}^2$ is defined in (A.2). As a result, the first term of the variational criterion $\mathcal{L}(\hat{\mu}_j, \hat{v}_j; x_{j+1})$ in (3.8), given by

$$\mathbb{E}\big[\log \hat{g}_j^{\ell,\star}(\hat{\mu}_j + \mathrm{e}^{\hat{v}_j/2} Z)\big] = \int \log \hat{g}_j^{\ell,\star}(x_j) \lambda_{j|j+1}^{\varphi}(x_j|x_{j+1})\,\mathrm{d}x_j,$$

can be computed exactly. Indeed, as $\mu_{\ell|j}$ is a linear function of x_j, this expectation is simply that of a quadratic function under a Gaussian density, given by

$$\mathbb{E}\big[\log \hat{g}_j^{\ell,\star}(\hat{\mu}_j + \mathrm{e}^{\hat{v}_j/2} Z)\big] = -\frac{1}{2}\left[\big\|\sqrt{\alpha_\ell}\, y - A\mu_{\ell|j}(\hat{\mu}_j)\big\|^2_{(\Sigma_j^\ell)^{-1}} + \mathrm{tr}\big((\Sigma_j^\ell)^{-1}\mathrm{diag}(\mathrm{e}^{\hat{v}_j})\big)\right] + C\,.$$

Hence, for these cases, (3.8) has a closed-form expression. However, it involves the computation of an inverse matrix which, for many problems, can be prohibitively expensive. To avoid this inversion, we instead optimize a *biased* estimate of $\mathcal{L}_j(\hat{\mu}_j, \hat{v}_j; x_{j+1})$ obtained by drawing two noise vectors $(Z, Z') \sim \mathrm{N}(0_{d_x}, I_{d_x})$ and setting

$$\begin{aligned}\widetilde{\mathcal{L}}_j(\hat{\mu}_j, \hat{v}_j; x_{j+1}) := &-\log g_\ell(\mu_{\ell|j}(\hat{\mu}_j + \mathrm{e}^{\hat{v}_j/2} Z) + \sigma_{\ell|j}^2 Z') \\ &+ \frac{\|\hat{\mu}_j - \mu_{j|j+1}(x_{j+1})\|^2}{2\sigma_{j|j+1}^2} - \frac{1}{2}\sum_{i=1}^{d_x}\left(\hat{v}_{j,i} - \frac{\mathrm{e}^{\hat{v}_{j,i}}}{\sigma_{j|j+1}^2}\right). \end{aligned} \tag{A.8}$$

This estimator is computable for any choice choice of potential and we have found in practice that it is sufficient to ensure good enough performance for our algorithm. Regarding the tamed unadjusted Langevin steps, we use the same biased estimate when the matrix inversions are expensive to compute; *i.e.* at each Langevin step, we approximate $G_\gamma^\ell(\tilde{X}_j)$ by

$$\widetilde{G}_\gamma^\ell(\tilde{X}_j) := \frac{\nabla_{x_{\ell+1}} \log g_\ell(\mu_{\ell|\ell+1}(x_{\ell+1}) + \sigma_{\ell|\ell+1}\tilde{Z}_\ell) + \hat{s}_{\ell+1}(x_\ell + 1)}{\|\nabla_{x_{\ell+1}} \log g_\ell(\mu_{\ell|\ell+1}(x_{\ell+1}) + \sigma_{\ell|\ell+1}\tilde{Z}_\ell) + \hat{s}_{\ell+1}(x_\ell + 1)\|}\,. \tag{A.9}$$

Algorithm 1 DIVIDE-AND-CONQUER POSTERIOR SAMPLER (DCPS)

Input: timesteps $(k_\ell)_{\ell=0}^L$, learning-rate ζ, numbers K and M of gradient and Langevin steps, respectively.
Initial sample $X_{k_L} \sim \mathcal{N}(0_{d_x}, I_{d_x})$;
for $\ell = L-1$ **to** 0 **do**
  Draw $Z \sim \mathrm{N}(0_{d_x}, I_{d_x})$ and compute $\widetilde{G}_\gamma^\ell(X_{k_{\ell+1}}^\ell)$ (A.9);
  $X_{k_{\ell+1}}^\ell \leftarrow X_{k_{\ell+1}}$
  for $i = 1$ **to** M **do**
    $Z \sim \mathrm{N}(0_{d_x}, I_{d_x})$;
    $X_{k_{\ell+1}}^\ell \leftarrow X_{k_{\ell+1}}^\ell + \gamma \widetilde{G}_\gamma^\ell(X_{k_{\ell+1}}^\ell) + \sqrt{2\gamma} Z$;
  end for
  for $j = k_{\ell+1} - 1$ **to** k_ℓ **do**
    $\hat{\mu}_j \leftarrow \mu_{j|j+1}(X_{j+1}^\ell)$; $\hat{v}_j \leftarrow \log \sigma_{j|j+1}^2 \cdot \mathbf{1}_{d_x}$;
    for $r = 1$ **to** K **do**
      Draw $(Z, Z') \sim \mathrm{N}(0_{d_x}, I_{d_x})$ and compute $\widetilde{\mathcal{L}}_j(\hat{\mu}_j, \hat{v}_j; X_{j+1}^\ell)$ (A.8);
      $\begin{bmatrix}\hat{\mu}_j \\ \hat{v}_j\end{bmatrix} \leftarrow \begin{bmatrix}\hat{\mu}_j \\ \hat{v}_j\end{bmatrix} - \zeta \|\nabla_{\hat{\mu}_j, \hat{v}_j} \widetilde{\mathcal{L}}_j(\hat{\mu}_j, \hat{v}_j; X_{j+1}^\ell)\|^{-1} \nabla_{\hat{\mu}_j, \hat{v}_j} \widetilde{\mathcal{L}}_j(\hat{\mu}_j, \hat{v}_j; X_{j+1}^\ell)$
    end for
    $\varepsilon \sim \mathcal{N}(0_{d_x}, I_{d_x})$
    $X_j^\ell \leftarrow \hat{\mu}_j + \mathrm{diag}(\mathrm{e}^{\hat{v}_j/2})\varepsilon$;
  end for
  $X_{k_\ell} \leftarrow X_{k_\ell}^\ell$;
end for

A.3 Proof of Proposition 3.1

For all $k \in [\![0, n-1]\!]$ we denote by $q_{k|k+1}(x_k|x_{k+1})$ the *exact* backward kernel which satisfies

$$q_{k+1}(x_{k+1})q_{k|k+1}(x_k|x_{k+1}) = q_k(x_k)q_{k+1|k}(x_{k+1}|x_k)\,. \tag{A.10}$$

Note that the backward kernels $p_{k|k+1}$ are to be understood as Gaussian approximations of the true backward kernels $q_{k|k+1}$. Below we give a complete statement of the proposition and provide a proof.

Proposition A.1. *Let $k \in [\![1, n]\!]$. Assume that $q_{k|k+1}(x_k|x_{k+1}) = p_{k|k+1}(x_k|x_{k+1})$ for all $(x_k, x_{k+1}) \in (\mathbb{R}^{d_x})^2$. For all $\ell \in [\![0, k-1]\!]$ and $x_k \in \mathbb{R}^{d_x}$,*

$$W_2(\hat{p}_{\ell|k}(\cdot|x_k), p_{\ell|k}(\cdot|x_k)) \leq \frac{\sqrt{\alpha_\ell}(1-\alpha_k/\alpha_\ell)}{(1-\alpha_k)} W_2(\hat{p}_{0|k}(\cdot|x_k), p_{0|k}(\cdot|x_k))\,.$$

Proof of Proposition A.1. Under the assumptions of the proposition, we have, for all $m > \ell$,

$$p_{\ell|k}(x_\ell|x_k) = q_{\ell|k}(x_\ell|x_k) = \int q_{\ell|0,k}(x_\ell|x_0, x_k)\, q_{0|k}(\mathrm{d}x_0|x_k)\,.$$

Indeed, by definition of the backward kernel $q_{0|k}(x_0|x_k)$ and (A.10), it holds that

$$\begin{aligned}
\int q_{\ell|0,k}(x_\ell|x_0, x_k) q_{0|k}(x_0|x_k)\,\mathrm{d}x_0 &= \int \frac{q_{\ell|0}(x_\ell|x_0) q_{k|\ell}(x_k|x_\ell)}{q_{k|0}(x_k|x_0)} \frac{q_0(x_0) q_{k|0}(x_k|x_0)}{q_k(x_k)}\,\mathrm{d}x_0 \\
&= \frac{q_{k|\ell}(x_k|x_\ell)}{q_k(x_k)} \int q_0(x_0) q_{\ell|0}(\mathrm{d}x_\ell|x_0)\,\mathrm{d}x_0 \\
&= q_{\ell|k}(x_\ell|x_k)\,.
\end{aligned}$$

As a result, we have that

$$\begin{aligned}
p_{\ell|k}(x_\ell|x_k) &= \int q_{\ell|0,k}(\mathrm{d}x_\ell|x_0, x_k) q_{0|k}(x_0|x_k)\,\mathrm{d}x_0\,, \\
\hat{p}_{\ell|k}(x_\ell|x_k) &= \int q_{\ell|0,k}(\mathrm{d}x_\ell|x_0, x_k) \hat{p}_{0|k}(x_0|x_k)\,\mathrm{d}x_0\,,
\end{aligned}$$

where, by definition, $\hat{p}_{0|k}(\cdot|x_k)$ is a Gaussian approximation of $q_{0|k}(\cdot|x_k)$ as defined in the main paper.

Next, let $\Pi_{0|k}(\cdot|x_k)$ denote a coupling of $q_{0|k}(\cdot|x_k)$ and $\hat{p}_{0|k}(\cdot|x_k)$, *i.e.*, for all $A \in \mathcal{B}(\mathbb{R}^{d_x})$,

$$\begin{aligned}
\int \mathbb{1}_A(x_0) \mathbb{1}_{\mathbb{R}^{d_x}}(\hat{x}_0)\, \Pi_{0|k}(x_0, \hat{x}_0|x_k)\,\mathrm{d}x_0\mathrm{d}\hat{x}_0 &= \int \mathbb{1}_A(x_0)\, q_{0|k}(x_0|x_k)\,\mathrm{d}x_0\,, \\
\int \mathbb{1}_{\mathbb{R}^{d_x}}(x_0) \mathbb{1}_A(\hat{x}_0)\, \Pi_{0|k}(x_0, \hat{x}_0|x_k)\,\mathrm{d}x_0\mathrm{d}\hat{x}_0 &= \int \mathbb{1}_A(\hat{x}_0)\, \hat{p}_{0|k}(\hat{x}_0|x_k)\,\mathrm{d}\hat{x}_0\,.
\end{aligned}$$

Consider then the random variables

$$\begin{aligned}
X_{\ell|k} &= \frac{\sqrt{\alpha_\ell}(1-\alpha_k/\alpha_\ell)}{1-\alpha_k} X_{0|k} + \frac{\sqrt{\alpha_k/\alpha_\ell}(1-\alpha_\ell)}{1-\alpha_k} x_k + \frac{\sqrt{(1-\alpha_\ell)(1-\alpha_k/\alpha_\ell)}}{\sqrt{1-\alpha_k}} Z\,, \\
\hat{X}_{s|k} &= \frac{\sqrt{\alpha_\ell}(1-\alpha_k/\alpha_\ell)}{1-\alpha_k} \hat{X}_{0|k} + \frac{\sqrt{\alpha_k/\alpha_\ell}(1-\alpha_\ell)}{1-\alpha_k} x_k + \frac{\sqrt{(1-\alpha_\ell)(1-\alpha_k/\alpha_\ell)}}{\sqrt{1-\alpha_k}} Z\,,
\end{aligned}$$

where $(X_{0|k}, \hat{X}_{0|k}) \sim \Pi_{0|k}(\cdot|x_k)$ and $Z \sim \mathcal{N}(0_{d_x}, I_{d_x})$. Then $(X_{\ell|k}, \hat{X}_{\ell|k})$ is distributed according to a coupling of $\hat{p}_{\ell|k}(\cdot|x_k)$ and $p_{\ell|k}(\cdot|x_k)$, and consequently

$$\begin{aligned}
W_2(\hat{p}_{\ell|k}(\cdot|x_k), p_{\ell|k}(\cdot|x_k)) &\leq \mathbb{E}\left[\|X_{\ell|k} - \hat{X}_{\ell|k}\|^2\right]^{1/2} \\
&\leq \frac{\sqrt{\alpha_\ell}(1-\alpha_k/\alpha_\ell)}{(1-\alpha_k)} \mathbb{E}\left[\|X_{0|k} - \hat{X}_{0|k}\|^2\right]^{1/2}\,.
\end{aligned}$$

The result is obtained by taking the infinimum of the rhs with respect to all couplings of $q_{0|k}(\cdot|x_k)$ and $\hat{p}_{0|k}(\cdot|x_k)$. □

B Related works.

In this section we discuss in more details existing works that bear some similarities with DCPS.

SMC based approaches. The MCGDIFF, the Twisted Diffusion sampler (TDS) of [59] using the FK representation (2.4). MCGDIFF is specific to linear inverse problems and the potentials used are $g_k(x_k) = \mathrm{N}(\sqrt{\alpha_k}\, y; Ax_k, (1-\alpha_k) I_{d_y})$ when $\sigma_y = 0$. TDS applies to any potential g_0 and relies on the DPS approximation for its potentials; i.e. $g_k(x_k) = g_0(\hat{x}_{0|k}(x_k))$. In either cases, a particle approximation of the posterior of interest π_0 is obtained using the Auxiliary Particle filter framework [37]. [16] also use particle filters for the posterior distribution; the potentials used are $g_k(x_k) = \mathrm{N}(\sqrt{\alpha_k} y_k; Ax_k, \alpha_k \sigma_y^2 I_{d_x})$ where $(y_k)_{k=0}^n$, with $y_0 = y$ is a sequence of observations sampled according to an auto-regressive process; see [16, Equation 7]. The posterior is thus viewed as approximately the time 0 marginal of a Hidden Markov model with transition $p_{k|k+1}$ and observation likelihood g_k, which is different from the FK representation (2.4). Our choice of intermediate potentials for linear inverse problems with Gaussian noise differs from that of MCGDIFF by the standard deviation of the observation model, which we set to be σ_y. A major difference of DCPS with these works lies in the fact that we do not rely on particle filters, thus avoiding the collapse in very large dimensions. As we have shown in the experimental section DCPS can achieve comparable performance to MCGDIFF in low dimensions, see Table 1 while also being efficient in very large dimensions, see Table 2. A second and major difference is that we have derived potentials for both the JPEG dequantization and Poisson-shot denoising tasks, which may be used to extend MCGDIFF and FPS-SMC [16] to these problems.

RedDiff. In this work we have also proposed to use Gaussian variational inference to approximate the intractable backward transition $\pi^{\ell}_{k|k+1}$. One particularity of our approach is that we do not use amortized variational inference [26] and instead optimize the variational distribution at each step of the diffusion. A similar approach is used in REDDIFF [32] but in a different way. Indeed, the authors use a *non-amortized* Gaussian variational approximation for the posterior π_0, meaning that in order to draw one sample from REDDIFF, several steps of optimization are performed on a score-matching-like loss. Interestingly, this approach does not require differentiating through the denoising network and is thus faster and more memory efficient. However, we found that this comes at the cost of performance as can be seen in Table 1, 2 and 3.

SDA. In [42], the authors introduce a posterior sampling algorithm for inverse problem where the chosen potential approximation is

$$g_\ell(x_\ell) = N(y; A\hat{x}^{\theta}_{0|\ell}(x_\ell), \sigma_y^2 + \frac{\gamma(1-\alpha_\ell)}{\alpha_\ell} AA^{\mathrm{T}}) \,,$$

with $\gamma > 0$ being a tunable parameter. Noteworthy, this potential is similar to one used in [47] with a slightly different choice of variance. Then, the *Score-based Data Assimilation* (SDA) algorithm proceed following the Predictor-Corrector framework [51]. In the Prediction stage, a sample X_k given X_{k+1} is drawn using the conditional score

$$\hat{s}_{k+1}(x_{k+1}) = \nabla \log \hat{p}_{k+1}(x_{k+1}) + \nabla \log g_{k+1}(x_{k+1}) \,.$$

In the Correction stage, a Langevin MC targeting the marginal distribution $g_k(x_k) p_k(x_k)$ is simulated starting from the predicted sample following

$$X_k^{i+1} = X_k^i + \delta_k(X_k^i)\hat{s}_k(X_k^i) + \sqrt{2\delta_k(X_k^i)} Z_i \,, \quad Z_i \sim \mathrm{N}(0, I) \,,$$

where δ_k is a state-dependent step-size. We emphasise that due to the dependence of the step sizes on the states, these are only Langevin-like updates that do not inherit the theoretical guarantees of the unadjusted Langevin algorithm. While SDA and our algorithm DCPS both use Langevin, the pivotal difference is that its purpose, in our case, is not to correct to ensure that the sample X_{k_ℓ} is distributed according to the marginal $\pi_{k_\ell}(x_{k_\ell}) \propto g_{k_\ell}(x_{k_\ell}) p_{k_\ell}(x_{k_\ell})$, but rather to ensure that the sample is distributed according to the next distribution $\pi_{k_\ell}(x_{k_\ell})$, which is the initial distribution of the next block, as per Equation (3.4). Hence, in our case, Langevin MC is used between blocks and not within blocks.

C Experiments

C.1 Implementation details

In this section we provide the global implementation details for each algorithm. We provide the specific parameters (when needed) used for each experiment (Gaussian mixture, image restoration and trajectory inpainting) in the dedicated sections below.

DCPS. For all the experiments we implement Algorithm 1. We use the same parameters $K = 2$, $L = 3$ and $\zeta = 1$ for all the experiments. For the number of Langevin steps, we set it to $M = 50$ and $M = 500$ (respectively) for the Gaussian mixture experiment and $M = 5$ for the imaging and trajectory inpainting experiments.

DDRM. We have used the official implementation[1] and used the recommended parameters in the original paper. We use 200 steps for DDRM and found that it works better than when we used 1000 steps.

DPS. We have implemented both Algorithm 1 (for linear inverse problems) and Algorithm 2 (for Poisson-shot restoration) given in [10]. In all the experiments we run DPS with 1000 Diffusion steps.

RedDiff. For RedDiff, we have used the publicly available implementation[2]. We have empirically found that RedDiff works best in the low observation standard deviation regime and produces spatially coherent reconstructions in the larger noise regime but struggles with getting rid of the noise as evidenced by the large increase in LPIPS values in Table 2. Note also that it is not clear how the parameters of the algorithm depend on the inverse problem standard deviation; indeed, looking at Algorithm 1 and then Appendix C.2 where the authors consider a noisy inverse problem[3] there seems to be no clear dependence of λ on σ_v (σ_y with our notations). In fact the authors use $\lambda = 0.25$ similarly to the noiseless experiments in the main paper and we believe that the tuning is performed only on the initial step-size of Adam. As a result, for the experiments with $\sigma_y = 0.3$, we have tuned it using a grid-search in $[0.1, 0.25]$ and retained 0.1.

ΠGDM. Regarding ΠGDM [47], note that there is no publicly available implementation and we have thus implemented the noisy version of [47, Algorithm 1] in the original paper. However, we did not manage to obtain appropriate results and found it to be quite unstable. We have further investigated the issue and found that ΠGDM is implemented in the github repository of RedDiff[4], which is by the same authors. We have noted that it has a slight difference with Algorithm 1 of the ΠGDM paper; the gradient term, coined g in [47, Algorithm 1], is multiplied by $\sqrt{\alpha_{t-1}\alpha_t}$ instead of simply $\sqrt{\alpha_t}$. We have found that this stabilizes the algorithm significantly for the linear inverse problem experiment. We use the same rescaling for the Gaussian mixture and trajectory inpainting experiment. However, even with this modification to the algorith we found that ΠGDM does not perform well when the noise standard deviation is large; see Table 2. For the JPEG experiment we do not use this rescaling as we found that the algorithm remains stable.

MCGDiff. For MCGDiff we have used the official implementation[5] with $N = 32$ particles for the imaging experiments. There are no further tuning parameters as far as we can tell.

DIFFPIR We implemented [62, Algorithm 1] and use the hyperparameters recommended in the official, released version[6].

DDNM. We adapted the implementation in the released code[7] to our code base.

[1] https://github.com/bahjat-kawar/ddrm
[2] https://github.com/NVlabs/RED-diff
[3] https://openreview.net/pdf?id=1YO4EE3SPB
[4] https://github.com/NVlabs/RED-diff
[5] https://github.com/gabrielvc/mcg_diff
[6] https://github.com/yuanzhi-zhu/DiffPIR
[7] https://github.com/wyhuai/DDNM

SDA. We implement the posterior sampling algorithm by combining [42, Algo 3 and 4 in Appendix C]. In the experiments, we use two Langevin corrections steps and found that $\gamma = 0.1$ works well across problems for the diagonal approximation the same as $\tau = 0.1$ for the Langevin correction steps size.

FPS We implement [16, Algorithm 2] provided in the appendix.

C.2 Gaussian mixtures

For a given dimension d_x, we consider p_{data} a mixture of 25 Gaussian random variables. The means of the Gaussian components of the mixture are $(\mathbf{m}_i)_{i=1}^{25} := \{(8i, 8j, \cdots, 8i, 8j) \in \mathbb{R}^{d_x} : (i,j) \in \{-2,-1,0,1,2\}^2\}$. The covariance of each component is identity. The mixture (unnormalized) weights $w_{i,j}$ are independently drawn from a Dirichlet distribution.

Metrics. To assess the performance of each algorithm we draw 2000 samples and compare against 2000 samples from the true posterior distribution using the Sliced Wasserstein distance by averaging over 10^4 slices. In Table 1 we report the average SW and the 95% confidence interval over 30 seeds. We found DPS and ΠGDM to be sometimes unstable, resulting in `NaN` values. To account for these unstabilities when computing the average SW distance, we replace `NaN` with 7 which is the typical value obtained when a stable algorithm fails to sample from the posterior.

Parameters. For DPS we use $\zeta_m = 0.1/\|y - A\hat{x}_{0|m}^{\theta^\star}(x_m)\|$ at step m of the Diffusion. As to DCPS we use $\gamma = 10^{-2}$ for the Langevin step-size.

Denoisers. Note that the loss (A.1) can be written as

$$\begin{aligned}
&\sum_{t=1}^{T} w_t \mathbb{E}\left[\|\epsilon_t - \hat{\epsilon}_t^\theta(\sqrt{\alpha_t}X_0 + \sqrt{1-\alpha_t}\epsilon_t)\|^2\right] \\
&\qquad = \sum_{t=1}^{T} \frac{w_t}{1-\alpha_t} \mathbb{E}\left[\|\sqrt{1-\alpha_t}\epsilon_t - \sqrt{1-\alpha_t}\hat{\epsilon}_t^\theta(\sqrt{\alpha_t}X_0 + \sqrt{1-\alpha_t}\epsilon_t)\|^2\right] \\
&\qquad = \sum_{t=1}^{T} \frac{w_t}{1-\alpha_t} \mathbb{E}\left[\|X_t - \sqrt{\alpha_t}X_0 - \sqrt{1-\alpha_t}\hat{\epsilon}_t^\theta(X_t)\|^2\right] \\
&\qquad = \sum_{t=1}^{T} \frac{w_t\alpha_t}{1-\alpha_t} \mathbb{E}\left[\left\|X_0 - \frac{X_t - \sqrt{1-\alpha_t}\hat{\epsilon}_t^\theta(X_t)}{\sqrt{\alpha_t}}\right\|^2\right] .
\end{aligned}$$

Hence the minimizer is

$$\epsilon_t^{\theta^\star}(x_t) = \frac{x_t - \sqrt{\alpha_t}\,\mathbb{E}[X_0|X_t = x_t]}{\sqrt{1-\alpha_t}} ,$$

which yields $\hat{x}_{0|t}^{\theta^\star} = \mathbb{E}[X_0|X_t = \cdot]$. Next, by Tweedie's formula we have that

$$\hat{x}_{0|t}^{\theta^\star}(x_t) = \frac{x_t + (1-\alpha_t)\nabla_x \log q_t(x_t)}{\sqrt{\alpha_t}} .$$

Hence, since q_{data} is a mixture of Gaussians, q_t is also a mixture of Gaussians with means $(\sqrt{\alpha_t}\mathbf{m}_i)_{i=1}^{25}$ and unit covariances. Therefore, $\nabla_x \log q_t(x_t)$ and hence $\hat{x}_{0|t}^{\theta^\star}(x_t)$ can be computed using automatic differentiation libraries.

Measurement model. For a pair of dimensions (d_x, d_y) the measurement model (y, A, σ_y) is drawn as follows: the elements $d_x \times d_y$ elements of the matrix are drawn i.i.d. from a standard Gaussian distribution, then σ_y is drawn uniformly in $[0,1]$ and finally we draw $x^\star \sim p_{\text{data}}$ and $\varepsilon \sim \mathcal{N}(0_{d_y}, I_{d_y})$ and set $y = Ax^\star + \sigma_y\varepsilon$.

Table 6: Mean LPIPS value on low count Poisson restoration.

Dataset	Task	DCPS	DPS
FFHQ	Denoising	**0.07**	0.12
	SR 4×	**0.17**	0.31
ImageNet	Denoising	**0.17**	0.24
	SR 4×	**0.36**	0.80

Posterior. Having drawn both p_{data} and (y, A, σ_y), the posterior can be computed exactly using standard Gaussian conjugation formulas [3, Eq. 2.116] and hence the posterior is a Gaussian mixture where all the components have the same covariance matrix $\Sigma := \left(I_{d_x} + \sigma_y^{-2} \, \mathrm{A}^T \, \mathrm{A}\right)^{-1}$ and means and weights given by

$$\tilde{\mathbf{m}}_i := \Sigma \left(A^\intercal y/\sigma_y^2 + \mathbf{m}_i\right) ,$$
$$\tilde{w}_i \propto w_i \mathrm{N}(y; A\mathbf{m}_i, \sigma_y^2 I_{d_x} + AA^\intercal) .$$

C.3 Imaging experiments

Parameters. For DCPS we set $\gamma = 10^{-3}$ for the Langevin step-size. For DPS we use the parameters recommended in the original paper, which we found to work well even on the half and expand masks; see [10, Appendix D.1].

Evaluation. In order to evaluate each algorithm we compute the LPIPS metric [61] on each dataset using 100 samples from the validation sets and report the average in Table 2, 3 and 6.

JPEG dequantization. We use the differentiable JPEG framework [44] which replaces the rounding function $x \mapsto \lfloor x \rceil$ used in the quantization part with $x \mapsto \lfloor x \rceil + (x - \lfloor x \rceil)^3$ which has non-zero derivatives almost everywhere.

C.4 Trajectory inpainting experiment

Trajectory DDM prior. The denoiser of the diffusion model has a Transformer-like architecture. In the entry of the network, the trajectory is augmented to a higher dimensional space (512) via dense layer. At this stage a positional encoding [56] is added to account for the diffusion step. Afterward, the output is flowed through a transformer encoder [56] whose feedforward layer dimension is 2048 to learn temporal dependence within the trajectory before being feed to an MLP with 4 layers ($512 \to 1024 \to 1024 \to 512$) and in between ReLU activation functions, to output the added noise. A Cosine noise scheduler with 1000 diffusion steps was used [34]. The UCY-student dataset was split int a train and a validation sets with 1450 and 140 trajectories respectively. The batch size was set to 10 times the training set, namely 145 samples The denoiser was trained to minimize the loss of DDPM [20] for 1000 epochs using Adam solver [25] with a Cosine learning rate scheduler [28]. The training was performed on 48GB L40S NVIDIA GPU and took roughly one minute to complete.

Metrics. The trajectory completion experiment was performed on the validation set. Every trajectory was masked randomly. Leveraging MCGDIFF 's asymptotical approximation of the posterior, it was run with 5000 particles to sample 100 samples from the posterior and afterward these were checked against a 100 reconstructions of each other algorithm by computing the *timestep wise* ℓ_2 distance between the quantile 50 (median), 25, 75 and also by computing the Sliced Wasserstein distance. This procedure was repeated for all trajectories in the validation set and later the results of each algorithm were aggregated by the mean ℓ_2 distances. Finally, this experiment was performed for two levels of noise $\sigma_y = 0.005$ and $\sigma_y = 0.01$.

C.5 Additional experiments

Here, we provide the complete tables of results on imaging and trajectories inpainting experiments that includes in addition DIFFPIR, DDNM, FPS , and SDA. These additional experiments were conducted during the rebuttal phase of our work.

Table 7: Mean LPIPS value on different tasks. Lower is better.

Dataset / σ_y	Task	DCPS	DDRM	DPS	ΠGDM	REDDIFF	MCGDIFF	DIFFPIR	DDNM	SDA	FPS
FFHQ / 0.05	Half	**0.20**	0.25	0.24	0.26	0.28	0.36	0.23	0.22	0.23	0.28
	Center	**0.05**	0.06	0.07	0.19	0.12	0.24	0.06	**0.05**	**0.05**	0.09
	SR 4×	**0.09**	0.18	**0.09**	0.33	0.36	0.15	0.13	0.14	0.10	0.10
	SR 16×	**0.23**	0.36	0.24	0.44	0.51	0.32	0.28	0.30	0.44	0.71
FFHQ / 0.3	Half	**0.25**	0.30	0.31	0.64	0.76	0.80	0.30	0.26	0.26	0.67
	Center	**0.10**	0.13	0.11	0.62	0.75	0.55	0.16	0.11	**0.10**	0.69
	SR 4×	0.21	0.26	**0.19**	0.77	0.77	0.65	0.28	0.23	**0.19**	0.75
	SR 16×	**0.35**	0.41	0.43	0.64	0.74	0.52	0.42	0.39	0.49	0.71
ImageNet / 0.05	Half	**0.35**	0.40	0.44	0.38	0.44	0.83	**0.35**	0.38	0.54	0.39
	Center	0.18	0.14	0.31	0.29	0.22	0.45	0.14	**0.13**	0.14	0.19
	SR 4×	**0.24**	0.38	0.41	0.78	0.56	1.32	0.36	0.34	0.85	0.27
	SR 16×	**0.44**	0.72	0.50	0.60	0.83	1.33	0.63	0.70	1.13	0.69
ImageNet / 0.3	Half	**0.40**	0.46	0.48	0.82	0.76	0.86	0.50	0.44	0.61	0.71
	Center	0.24	0.25	0.40	0.68	0.71	0.47	0.36	**0.22**	0.25	0.70
	SR 4×	**0.43**	0.50	0.47	0.87	0.83	1.31	0.61	0.46	1.14	0.84
	SR 16×	0.72	0.77	**0.57**	0.72	0.92	0.67	0.76	0.75	1.19	0.74
Average		**0.28**	0.35	0.32	0.57	0.60	0.67	0.35	0.32	0.48	0.53
Median		**0.24**	0.33	0.35	0.63	0.72	0.60	0.32	0.28	0.35	0.69

Table 8: ℓ_2 distance quantiles with MCGDIFF as reference.

	$\sigma_y = 0.005$			$\sigma_y = 0.01$		
	$q50$	$q25$	$q75$	$q50$	$q25$	$q75$
DCPS	**1.31**	**1.33**	**1.47**	**1.33**	**1.42**	**1.42**
DPS	1.34	1.40	1.61	1.36	1.48	1.52
DDRM	1.48	1.46	1.61	1.59	1.62	1.61
ΠGDM	1.36	1.35	**1.47**	1.37	1.43	**1.42**
REDDIFF	1.67	1.57	1.82	1.56	1.54	1.65
DIFFPIR	1.57	1.84	1.98	1.52	1.94	1.89
DDNM	1.45	1.45	1.65	1.52	1.59	1.59
FPS	2.60	2.61	2.62	2.91	2.90	2.89
SDA	1.52	1.55	1.69	1.54	1.59	1.61

D Sample reconstructions

In this section we display the remaining samples from the experiments in the main paper. We remind the reader that all algorithms are run with the same seed and we draw in parallel 4 samples from each algorithm and display them in their order of appearance.

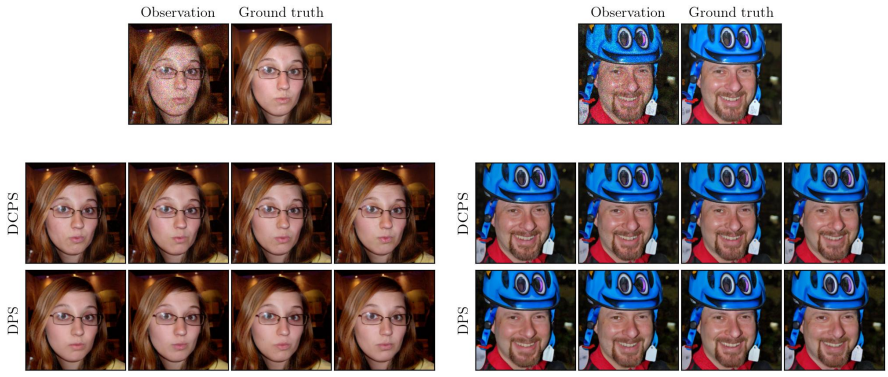

Figure 5: Denoising task with Poisson noise on FFHQ.

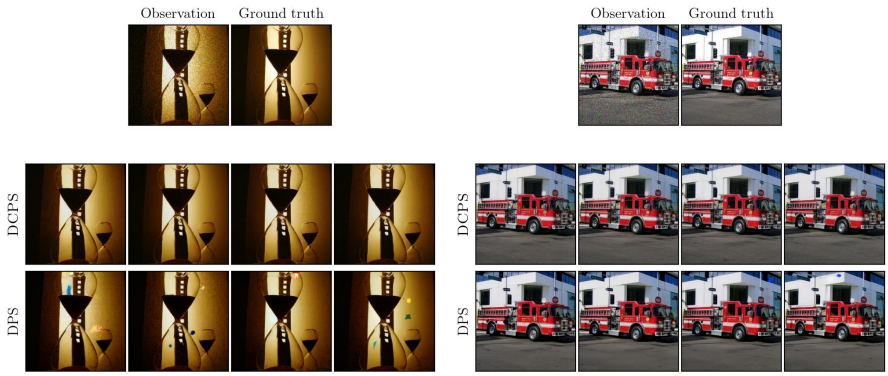

Figure 6: Denoising task with Poisson noise on `ImageNet`.

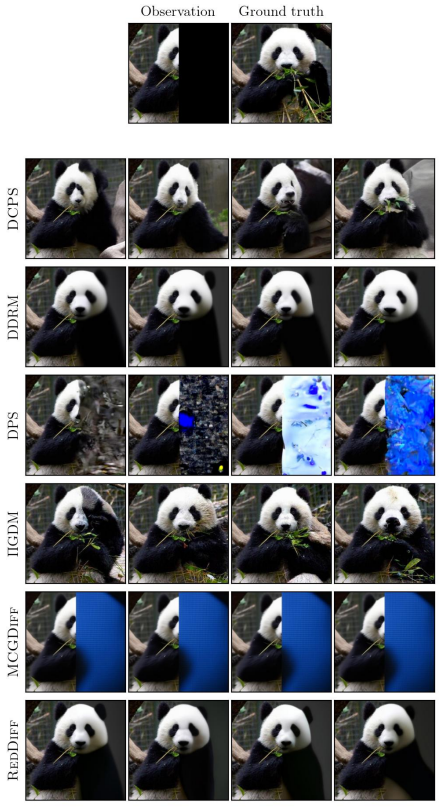

Figure 7: Outpainting task with half mask on `ImageNet`.

Figure 8: Inpainting with box mask on FFHQ.

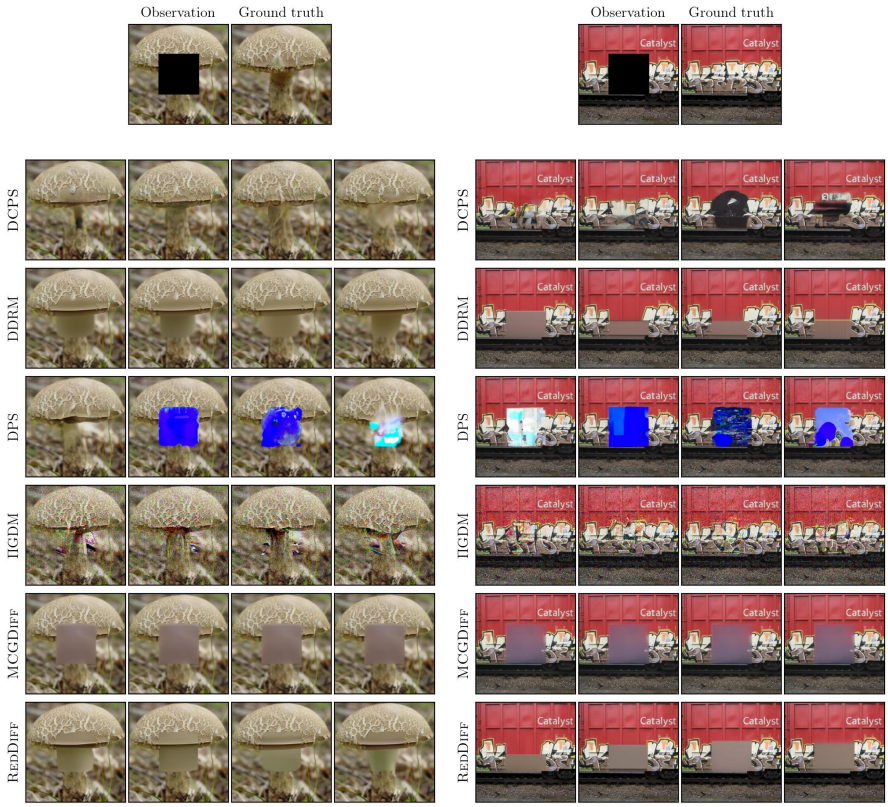

Figure 9: Inpainting task with box mask on ImageNet.

Figure 10: Outpainting task with half mask on `FFHQ`.

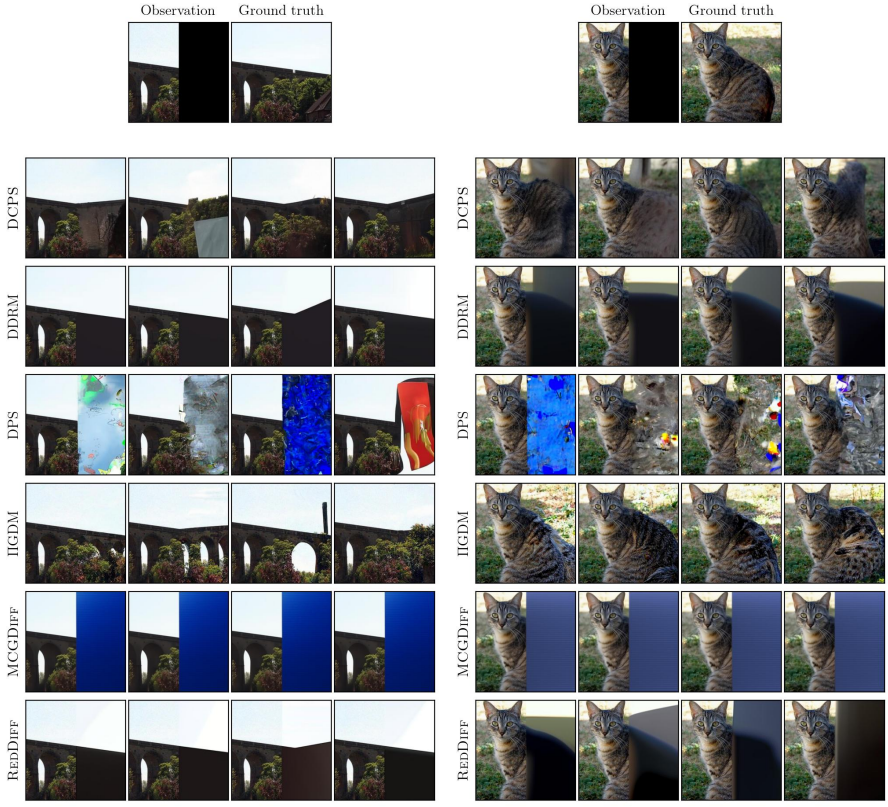

Figure 11: Outpainting task with half mask on `ImageNet`.

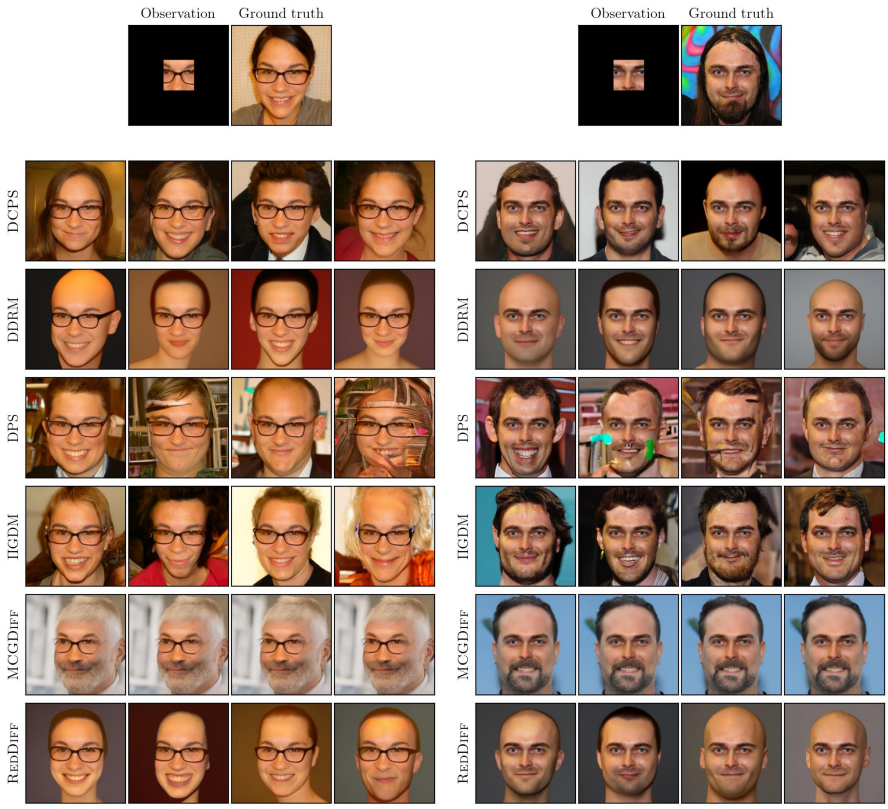

Figure 12: Outpainting expend task on FFHQ.

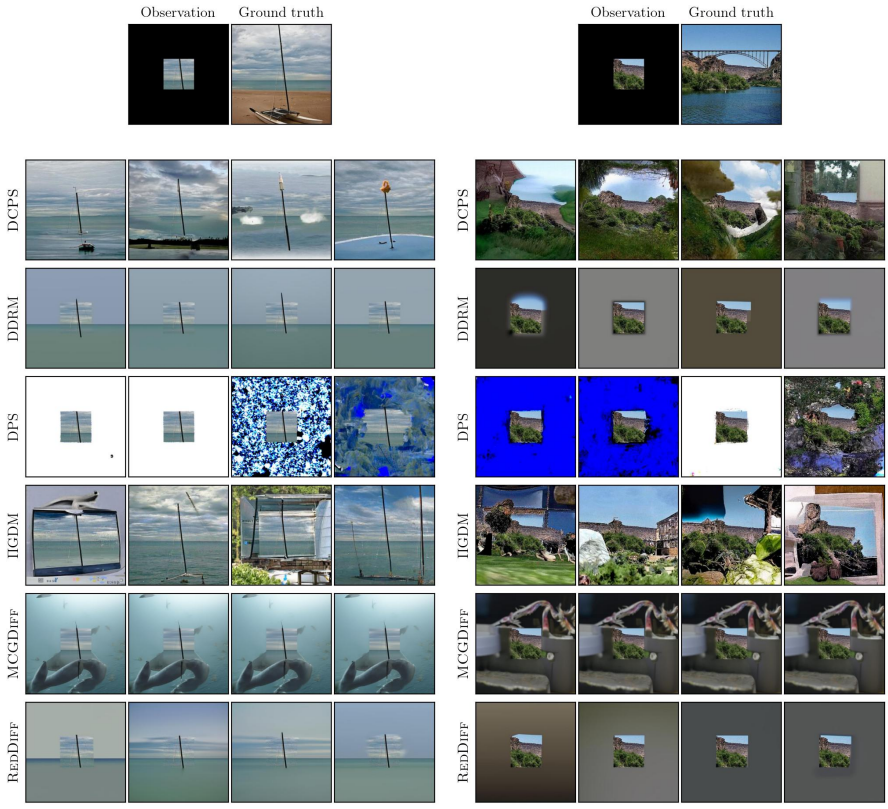

Figure 13: Outpainting expend task on ImageNet.

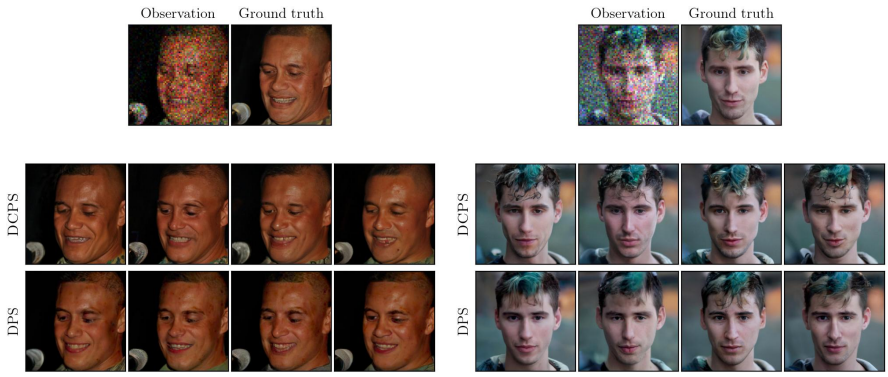

Figure 14: SR $4\times$ task with Poisson noise on `FFHQ`.

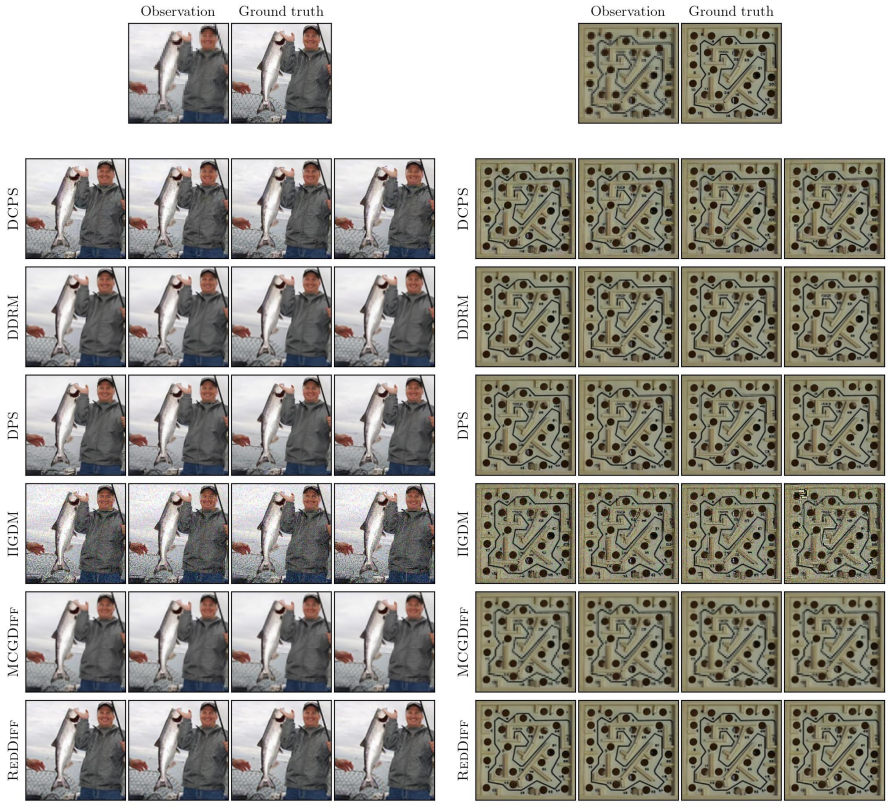

Figure 15: SR $4\times$ task on `ImageNet`.

Figure 16: SR 16× task on FFHQ.

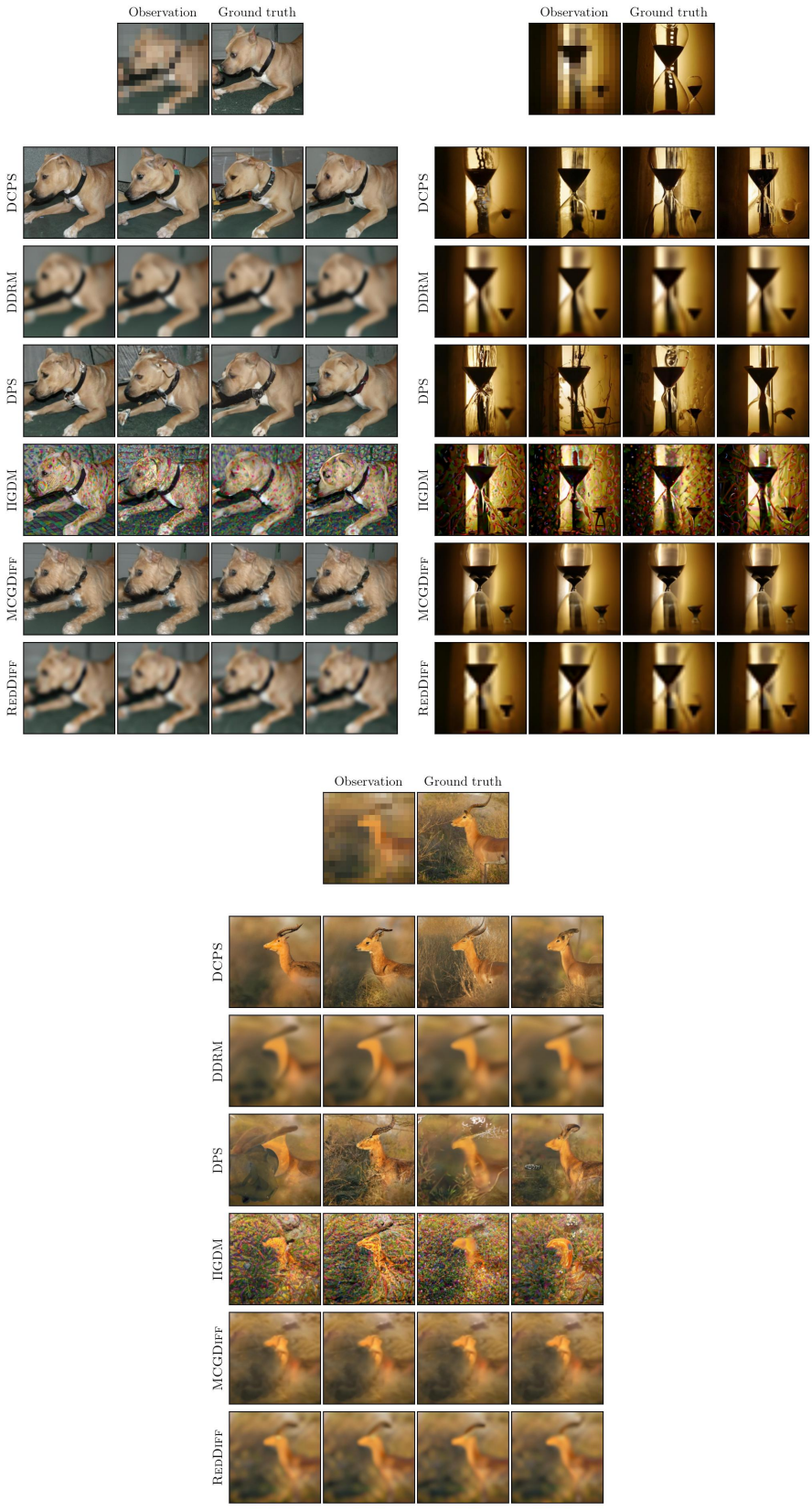

Figure 17: SR 16× task on `ImageNet`.

Figure 18: JPEG task with QF=8 on FFHQ.

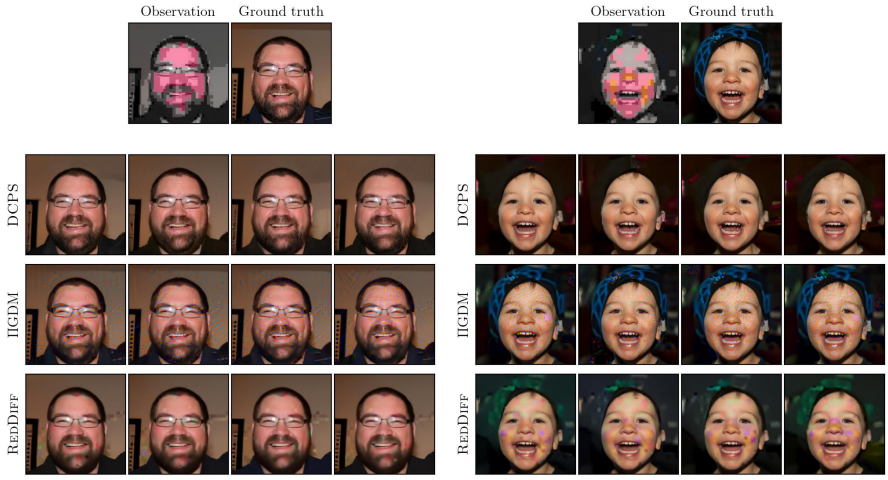

Figure 19: JPEG task with QF=2 on FFHQ.

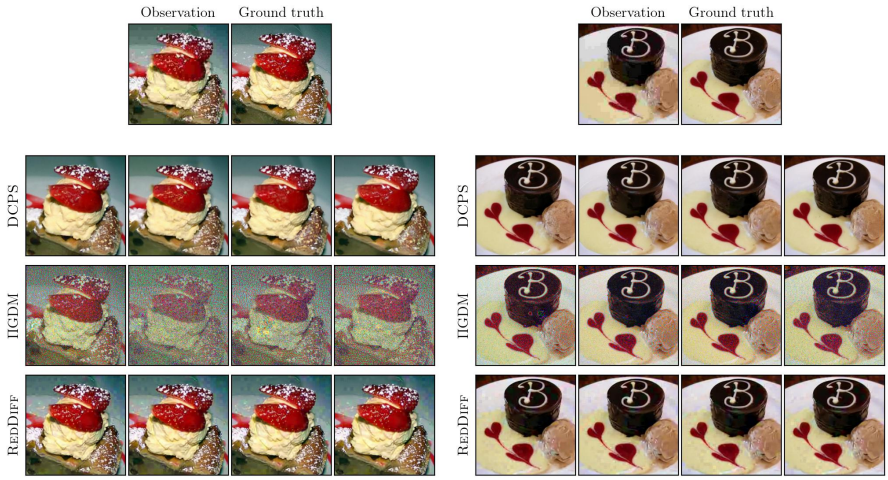

Figure 20: JPEG task with QF=8 on `ImageNet`.

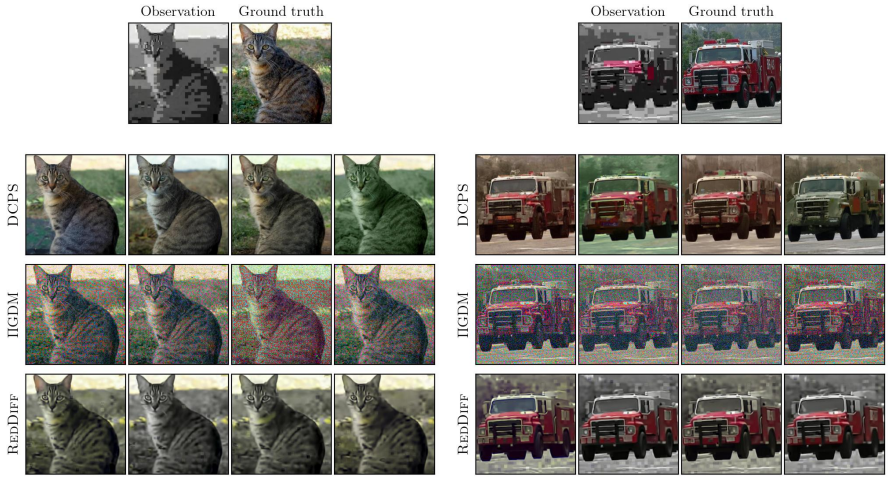

Figure 21: JPEG task with QF=2 on `ImageNet`.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The claims are clearly stated in the introduction.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We have added a limitations section.

   Guidelines:

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

3. **Theory Assumptions and Proofs**

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

   Answer: [Yes]

Justification: All the assumptions needed are stated clearly.

Guidelines:

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

4. **Experimental Result Reproducibility**

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

   Answer: [Yes]

   Justification: We have provided the exact implementation details in the appendix.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: We have provided a link with the relevant code as well as the link to download the datasets we have used

Guidelines:

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

6. **Experimental Setting/Details**

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

   Answer: [Yes]

   Justification: These are all given in Appendix C.

   Guidelines:

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

7. **Experiment Statistical Significance**

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

   Answer: [Yes]

   Justification: We report the 95% confidence intervals for our experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

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

8. **Experiments Compute Resources**

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

   Answer: [Yes]

   Justification: We provide the memory usage and runtime for each algorithm.

   Guidelines:

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

9. **Code Of Ethics**

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

   Answer: [Yes]

   Guidelines:

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

10. **Broader Impacts**

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

    Answer: [NA]

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate

to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [NA]

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We properly cite the authors that have released the datasets and models we use.

    Guidelines:

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

13. **New Assets**

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

Answer: [Yes]

Justification: The released code is accompanied by a README file detailing its contents, installation instructions, and usage guidelines.

Guidelines:

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

14. **Crowdsourcing and Research with Human Subjects**

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

Answer: [NA]

Guidelines:

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

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

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

Answer: [NA]

Guidelines:

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