Measurement Consistent Tweedie's: Solving Inverse Problems with the Conditional Posterior Mean

Henry Li [*1] **Jon Patsenker** [*1] **Myeongseob Ko** [2] **Ruoxi Jia** [2] **Yuval Kluger** [1]

Abstract

Diffusion models have been firmly established as principled zero-shot solvers for linear and nonlinear inverse problems, owing to their powerful image prior and iterative sampling algorithm. These approaches rely on Tweedie's formula, which uses the score function at each step to relate the diffusion variate $\mathbf{x}_t$ to the posterior mean $\mathbb{E}[\mathbf{x}_0|\mathbf{x}_t]$, which is used to estimate the error of the final denoised sample $\mathbf{x}_0$. However, this approach leaves out information from the measurement $\mathbf{y}$ itself, which must then be integrated downstream. In this work, we propose to directly estimate the *conditional* posterior mean $\mathbb{E}[\mathbf{x}_0|\mathbf{x}_t, \mathbf{y}]$, which can be formulated as a lightweight, single-parameter maximum likelihood estimation problem. The resulting prediction can be summarized as a *data-conditional* score and integrated into any standard sampler, resulting in a fast and memory-efficient inverse solver. Moreover, our optimizer is amenable to a noise-aware likelihood-based stopping criteria that is robust to measurement noise in $\mathbf{y}$. We demonstrate comparable or improved performance against a wide selection of contemporary inverse solvers across multiple datasets and tasks.

1. Introduction

In this work, we study a broad class of problems involving the recovery of a signal $\mathbf{x}$ from a measurement

$$\mathbf{y} = \mathcal{A}(\mathbf{x}) + \boldsymbol{\eta}. \tag{1}$$

with noise $\boldsymbol{\eta}$ and measurement operator $\mathcal{A}$. Known as inverse problems, such formulations appear in a multitude of fields, with applications including acoustic reconstruction (Kac, 1966), seismic profiling (Hardage, 1985), X-ray computed tomography and magnetic resonance imaging (Suetens, 2017), and a large number of computer vision reconstruction tasks such as inpainting, deconvolution, colorization, super-resolution, and phase retrieval (Andrews and Hunt, 1977).

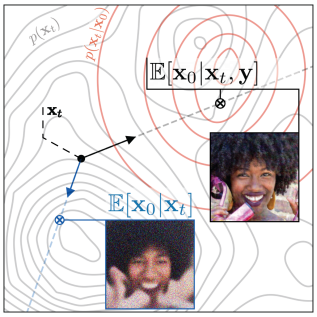


Figure 1: An illustration of the **posterior mean** versus the **conditional posterior mean** at time t. The latter can be obtained by estimating the *data-conditional* score with the extra information contained in $\mathbf{y}$.

Often, inverting $\mathcal{A}$ is numerically intractable (Appendix C), meaning that solutions $\mathbf{x}$ satisfying $\mathcal{A}(\mathbf{x}) = \mathbf{y}$ are not directly obtainable or unique (Vogel, 2002). Moreover, due to measurement noise, it is often possible, but not practically desirable to fit perfectly to $\mathbf{y}$ for risk of *overfitting* to $\boldsymbol{\eta}$ (Aster et al., 2018). Therefore, a fundamental quandary in solving inverse problems is how to select the best option from an equivalence class of solutions, i.e., choosing $\mathbf{x}_* \in \{\mathbf{x} : \mathcal{A}(\mathbf{x}) \approx \mathbf{y}\}$.

In classical solvers, this is carried out by a regularizer on a normed error loss (Engl et al., 1996). One seeks

$$\mathbf{x}_* = \arg\min_{\mathbf{x}} R(\mathbf{x}) \qquad \text{s.t.} \quad ||\mathcal{A}(\mathbf{x}) - \mathbf{y}|| \leq \epsilon, \tag{2}$$

where ϵ is a soft error margin and R is a simple function that satisfies user-specified heuristics, e.g., smoothness or total variation (Beck and Teboulle, 2009). However, such approaches often fail to produce realistic results, as R lacks the ability to reconstruct details lost by $\mathcal{A}$. With the advent of deep generative models, practitioners found that restricting solutions to the range of a generative model G can greatly improve realism: let $\mathbf{x} = G(\mathbf{w})$ and optimize over $\mathbf{w}$, which can be latent inputs (Bora et al., 2017) or

[*]Equal contribution [1]Yale University [2]Virginia Tech. Correspondence to: Henry Li <henry.li@yale.edu>.

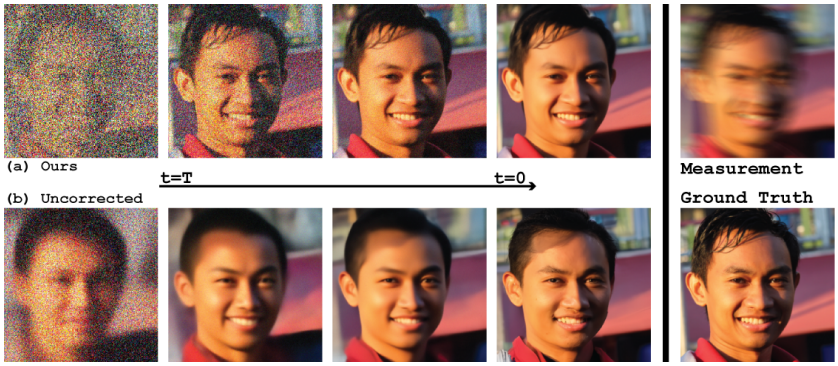


Figure 2: (a) $\mathbb{E}[\mathbf{x}_0|\mathbf{x}_t]$ via the unconditional score versus (b) $\mathbb{E}[\mathbf{x}_0|\mathbf{x}_t, \mathbf{y}]$ via the data-conditional score obtained by our maximum likelihood estimator. With the *unconditional* score, $\hat{\mathbf{x}}_0$ estimates the posterior mean of the dataset, rather than a sample $\mathbf{x}$ that satisfies $\mathcal{A}(\mathbf{x}) \approx \mathbf{y}$, especially at $T \gg 0$ (Section 2).

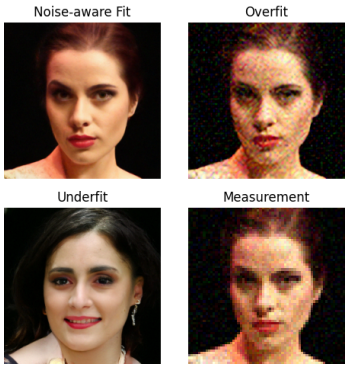


Figure 3: The hazard of over- or under-fitting for a super-resolution task. An ideal noise-aware fit balances between the prior and the noisy measurement $\mathbf{y}$.

Table 1: Overview of pixel-based solvers used for comparisons in this work. We list the type (Section A.2), whether it requires backpropagation through a neural function evaluation, runtime, and memory footprint.

Solver	Type	No NFE Backprop	Runtime	Memory
DCS (Ours)	Hybrid	✓	1x	1x
MCG (Chung et al., 2022b)	Projection	✗	2.6x	3.2x
DPS (Chung et al., 2023)	Posterior	✗	2.5x	3.2x
DPS-JF (Chung et al., 2023)	Posterior	✓	1.2x	1.1x
DDNM (Wang et al., 2022)	Projection	✓	1.5x	1x
DDRM (Kawar et al., 2022)	Projection	✓	1.5x	1x
RED-Diff (Mardani et al., 2024)	Projection	✓	1.5x	1x
LGD-MC (Song et al., 2023b)	Posterior	✗	2x	3.2x

weights (Ulyanov et al., 2018) of a deep neural network. Overall, these methods improve the fidelity of $\mathbf{x}$, but they lack interpretability and require a judiciously selected R and ϵ.

Recently, great strides have been made in solving inverse problems with diffusion models (Ho et al., 2020), which produce diverse, realistic samples (Dhariwal and Nichol, 2021; Esser et al., 2024) with robust generalization guarantees (Kadkhodaie et al., 2023). Moreover, they are interpretable, directly modeling the (Stein) score $\nabla \log p_t(\mathbf{x}_t)$. *Unconditional sampling* proceeds by reversing a noising process on $\mathbf{x}_0 \sim p_{\text{data}}$ roughly described (in black) by

$$\mathbf{x}_{t-1} = \overbrace{\texttt{denoise}[\mathbf{x}_t, \nabla \log p_t(\mathbf{x}_t)]}^{\texttt{unconditional sampling}} \underbrace{+ \texttt{guidance}}_{\texttt{conditional sampling}}.$$

Solvers then employ a *conditional sampling* process via a `guidance` term that pushes samples toward solutions consistent with $\mathbf{y}$. This approach faces a fundamental challenge: the `guidance` term depends on a *consistency error* $||\mathcal{A}(\mathbf{x}) - \mathbf{y}||$ that is only tractable for $\mathbf{x} = \mathbf{x}_0$ (Chung et al., 2022a). Such methods thus rely (explicitly or implicitly via (Song et al., 2020a)) on Tweedie's formula, which estimates $\mathbf{x}_0$ given a noise prediction $\boldsymbol{\epsilon}_t \approx -\sigma_t \nabla \log p_t(\mathbf{x}_t)$:

$$\hat{\mathbf{x}}_0 = \mathbb{E}[\mathbf{x}_0|\mathbf{x}_t] = \frac{1}{\sqrt{\alpha_t}}(\mathbf{x}_t - \sigma_t \boldsymbol{\epsilon}_t). \tag{3}$$

This can then be substituted for $\mathbf{x}$ in the consistency error, producing a differentiable function with respect to $\mathbf{x}_t$.

However, a naive implementation of Eq. 3 introduces significant approximation error, as $\mathbb{E}[\mathbf{x}_0|\mathbf{x}_t] = \mathbf{x}_0$ *if and only if* $\mathbf{x}_t$ is normally distributed (Theorem 2.1). Existing methods use the unconditional score $\nabla \log p_t(\mathbf{x}_t)$ where this assumption does not generally hold. In this work, we use the *data-conditional* score $\nabla \log p_t(\mathbf{x}_t|\mathbf{x}_0)$ and a *measurement consistent* variant of Tweedie's formula that incorporates additional information from $\mathbf{y}$ to obtain the *conditional* posterior mean $\mathbb{E}[\mathbf{x}_0|\mathbf{x}_t, \mathbf{y}]$ (Section 3).

The *data-conditional* score $\nabla \log p_t(\mathbf{x}_t|\mathbf{x}_0)$ plays a crucial role during diffusion model training but is intractable during unconditional sampling where $\mathbf{x}_0$ is unknown. Surprisingly, in inverse problems, the information present in $\mathbf{y}$ allows this term to be recovered to great accuracy by simple maximum likelihood estimation with the measurement model. We outline this framework (Section 3) and show theoretically that our choice of estimator is statistically sufficient for the measurement, meaning the *conditional posterior mean* $\mathbb{E}[\mathbf{x}_t|\mathbf{x}_0, \mathbf{y}]$ contains all of the information contained in $\mathbf{y}$.

Our **contributions** are as follows:

- We identify a fundamental limitation with using Tweedie's formula to predict $\mathbf{x}_0$ in inverse problems: the approximation is only exact when $\mathbf{x}_t$ is normally distributed, which is almost never true.
- We sidestep this by considering a simpler *conditional* diffusion process defined in terms of $p(x_t|x_0)$. We

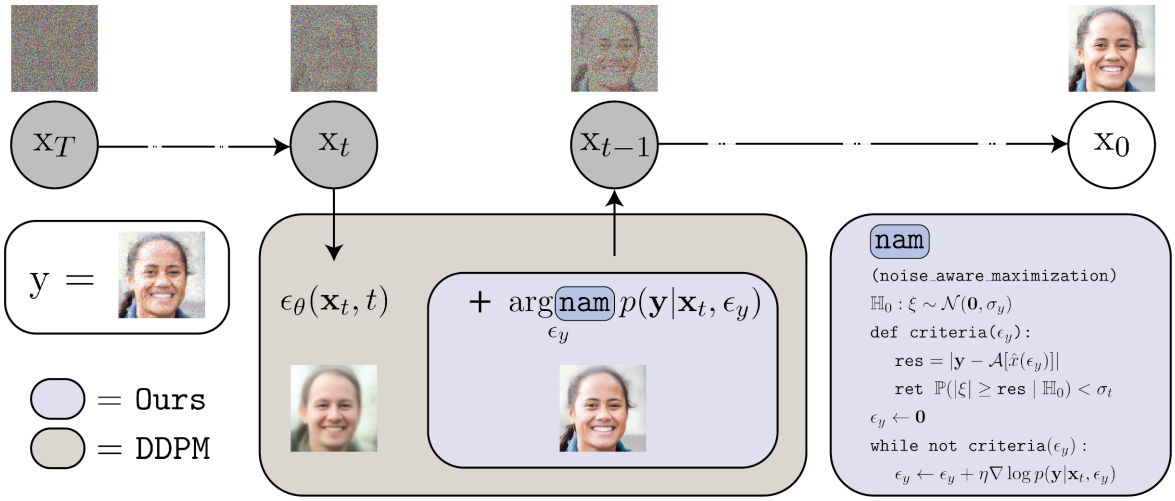


Figure 4: An illustration of our proposed sampling algorithm. An initial noise prediction ϵ_θ is corrected by the solution ϵ_y of a **n**oise-**a**ware **m**aximization scheme of the measurement likelihood $p(y|x_t, \epsilon_y)$. This results in the corrected data-conditional noise prediction $(\epsilon_\theta + \epsilon_y) \approx -\sigma_t^{-1} \nabla \log p_t(x_t|x_0)$. For details see Section 3.

propose a maximum likelihood estimator for its score, and show that it captures all information present in y, even under significant measurement noise (Figure 5).

- We demonstrate how this score can be plugged into any standard sampler (e.g., DDPM), resulting in an algorithm that is simple, noise-robust, neural backpropagation-free, and stable across time steps. Moreover, it achieves improvements in performance on a large selection of inverse problems, datasets and noise levels[1].

2. Revisiting Tweedie's for Inverse Problems

Diffusion models (Ho et al., 2020) reverse a noise-corrupting forward process with marginals $x_t \sim p_t(x_t|x_0)$,

$$x_t = \sqrt{\alpha_t} x_0 + \underbrace{\sqrt{1-\alpha_t}}_{\sigma_t} z, \qquad z \sim \mathcal{N}(0, I), \tag{4}$$

parameterized by a monotonically time-decreasing scalar α_t. New samples are generated via the reverse diffusion process which leverages the learned score function $s_\theta = -\sigma_t^{-1} \epsilon_\theta \approx \nabla \log p_t(x_t)$ (Anderson, 1982; Vincent, 2011; Song et al., 2020b). Diffusion-based solvers for inverse problems can be categorized as **posterior** or **projection** solvers, and aim to *modify* the reverse process such that the final variate x_0 coincides with a member of the solution set $\{x_0 : \mathcal{A}(x_0) \approx y\}$[2]. However, this paradigm is afflicted by a fundamental *computability paradox*: since the consistency error is only explicitly known at $t = 0$ via the likelihood function

$$p(y|x_0) \propto \exp\left(-\frac{1}{2\sigma_y^2} ||y - \mathcal{A}(x_0)||_2^2\right) \tag{5}$$

we cannot exactly guide the diffusion process at time $t > 0$ without first solving for x_0. Simultaneously, we cannot generally obtain x_0 without first computing x_t. Accurately estimating $\hat{x}_0 \approx x_0$ is a fundamental problem all solvers must contend with to function properly.

In **posterior** solvers, this culminates in the computation of $\nabla \log p(y|x_t)$, which is approximated by $\nabla \log p(y|\hat{x}_0)$. In **projection** solvers, this is the projection step Px_t, which is driven by a projection on $\hat{x}_0$, followed by a DDIM step (Song et al., 2020a) that involves $\hat{x}_0$. In both cases, Tweedie's formula is used to create an estimate for x_0, given the current x_t by predicting the **posterior mean**

$$\mathbb{E}[x_0|x_t] = \int_{\mathbb{R}^d} x_0 p(x_0|x_t) dx_0. \tag{6}$$

A surprising limitation. This estimator has an acute limitation: $\mathbb{E}[x_0|x_t]$ only coincides with x_0 when x_t is normally distributed. We formalize this *necessary and sufficient* condition below.

Theorem 2.1. *Let x_t be sampled from a diffusion process (as in Eq. 4). $\mathbb{E}[x_0|x_t] = x_0$ **if and only if** $p(x_t)$ is a simple isotropic Gaussian with mean $\sqrt{\alpha_t} x_0$ and variance $\sigma_t I$.*

Note that x_t is *almost never* Gaussian, since x_t is distributed as $\phi_\sigma * p_{\text{data}}$. While ϕ_σ is a simple isotropic Gaussian, p_{data}

[1] Code for method and experiments provided in https://anonymous.4open.science/r/diffusion_conditional_sampling

[2] We defer a more extensive discussion on diffusion models diffusion-based inverse problem solvers to Appendix A.

Figure 5: Reconstruction quality at various noise levels $\sigma_{\mathbf{y}} \in \{0.01, 0.1, 0.5\}$. Our approach strikes a careful balance between quality at each noise level (Table 2) and computational cost (Table 1). More examples in Appendix F.

is not — the data distribution can be arbitrarily non-convex and multimodal.

We visualize this in Figure 2, where at larger values of t, the fidelity of the estimated $\mathbf{x}_0$ is poor, resulting in a low quality prediction that is inconsistent with the measurement $\mathbf{y}$. Ultimately, Eq. 6 is a weighted average over *all* data $\mathbf{x} \sim p_{\text{data}}$, and often cannot properly estimate $\mathbf{x}_0$ without incorporating measurement information from $\mathbf{y}$. Instead, we propose to use the *conditional* posterior mean

$$\mathbb{E}[\mathbf{x}_0|\mathbf{x}_t, \mathbf{y}] = \int_{\mathbb{R}^d} \mathbf{x}_0 p(\mathbf{x}_0|\mathbf{x}_t, \mathbf{y}) d\mathbf{x}_0, \tag{7}$$

which only considers those $\mathbf{x}_0 \sim p_{\text{data}}$ consistent with $\mathbf{y}$. In the following section, we outline a method for directly incorporating this conditional information into the Tweedie's estimate.

3. Diffusion Conditional Sampling

We propose a novel framework for solving inverse problems with diffusion models via a *measurement consistent* Tweedie's formula predicting the *conditional* posterior mean (Eq. 7).

At each step, we form a single-parameter measurement model whose maximum likelihood estimator approximates the *data-conditional* score $\nabla \log p_t(\mathbf{x}_t|\mathbf{x}_0)$ (Section 3.1). This estimator is optimized with a noise-robust, likelihood-based early stopping criterion (Section 3.2). The learned score is then input to standard DDPM sampler (Ho et al., 2020), resulting in Algorithm 1. This approach is motivated by both powerful theoretical guarantees (Section 3.3), as well as significant computational advantages (Section 3.4).

3.1. Measurement Likelihood Model

We wish to reverse a diffusion process originating from a fixed $\mathbf{x}_0$ — the desired signal $\mathbf{x}$ (Eq. 1), where

$$p_*(\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_t; \sqrt{\alpha_t}\mathbf{x}_0, (1-\alpha_t)\mathbf{I}). \tag{8}$$

This is equivalent to the conditional distribution $p_t(\mathbf{x}_t|\mathbf{x}_0)$ of the forward process in a standard diffusion model. While $\mathbf{x}_0$ is unknown, we can apply Tweedie's to Eq. 8 and solve for the related *data-conditional score* via a closed form expression for the likelihood (Eq. 5):

$$\log p(\mathbf{y}|\mathbf{x}_0(\epsilon_{\mathbf{y}}, \mathbf{x}_t)) \propto -\frac{1}{2\sigma_{\mathbf{y}}^2} \left|\left| \mathbf{y} - \mathcal{A}\left(\frac{1}{\sqrt{\alpha_t}} [\mathbf{x}_t + \sigma_t^2 \underbrace{\nabla_{\mathbf{x}_t} \log p_*(\mathbf{x}_t)}_{\nabla \log p_t(\mathbf{x}_t|\mathbf{x}_0)}] \right) \right|\right|_2^2. \tag{9}$$

Since $p_t(\mathbf{x}_t|\mathbf{x}_0)$ is distributed as an isotropic Gaussian *by construction*, the posterior mean would recover $\mathbf{x}_0$ *exactly*

Algorithm 1 Diffusion Conditional Sampler (DCS)

Input: $\mathbf{y}, \mathcal{A}, \epsilon_\theta$ | **Output:** $\mathbf{x}_0$
$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T$ **to** 1 **do**
  $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\epsilon_{\mathbf{y}} \leftarrow \arg\underset{\epsilon_{\mathbf{y}}}{\texttt{nam}}\, p_t\left(\mathbf{y}|\frac{\mathbf{x}_t+\sigma_t^2(\epsilon_\theta+\epsilon_{\mathbf{y}})}{\sqrt{\alpha_t}}\right)$
  $\mathbf{x}_{t-1} \leftarrow \texttt{ddpm\_step}(\mathbf{x}_t, \epsilon_\theta + \epsilon_y)$
end for

Algorithm 2 Noise-aware Maximization (`nam`)

Input: $\mathbf{y}, \mathcal{A}, \mathbf{x}_t, \epsilon_\theta$ | **Output:** ϵ_y
$\epsilon_y \leftarrow \mathbf{0}$
$\hat{\mathbf{x}} \leftarrow \texttt{Tweedie's}(\mathbf{x}_t, \epsilon_\theta + \epsilon_y)$
while $2\Phi[-||y - \mathcal{A}[\hat{\mathbf{x}}]||_1^1/(d\sigma_{\mathbf{y}})] < \sigma_t$ **do**
  $\epsilon_y \leftarrow \epsilon_y + \eta\nabla \log p_t\left(\mathbf{y}|\frac{\mathbf{x}_t+\sigma_t^2(\epsilon_\theta+\epsilon_{\mathbf{y}})}{\sqrt{\alpha_t}}\right)$
  $\hat{\mathbf{x}} \leftarrow \texttt{Tweedie's}(\mathbf{x}_t, \epsilon_\theta + \epsilon_y)$
end while

(Theorem 2.1), which we further discuss below. Now, defining the equivalence,

$$\nabla \log p_t(\mathbf{x}_t|\mathbf{x}_0) = -\sigma_t^{-1}[\epsilon_\theta(\mathbf{x}_t, t) + \epsilon_{\mathbf{y}}], \tag{10}$$

we can solve for our single parameter ϵ_y by maximizing the joint likelihood between the measurement $\mathbf{y}$ and our parameter $\epsilon_{\mathbf{y}}$. This forms our estimator for the true *data-conditional score*, $\nabla \log p_t(\mathbf{x}_t|\mathbf{x}_0)$:

$$s_{\text{corrected}} = -\sigma_t^{-1}\left[\epsilon_\theta(\mathbf{x}_t, t) + \epsilon^*_{\mathbf{y}}\right], \tag{11}$$

where

$$\epsilon^*_{\mathbf{y}} = \arg\max_{\epsilon_{\mathbf{y}}} \frac{-1}{2\sigma_{\mathbf{y}}^2}\left|\left|\mathbf{y} - \mathcal{A}\left(\frac{\mathbf{x}_t - \sigma_t[\epsilon_\theta(\mathbf{x}_t, t) + \epsilon_{\mathbf{y}}]}{\sqrt{\alpha_t}}\right)\right|\right|_2^2. \tag{12}$$

Finally, $s_{\text{corrected}}$ can then be input to any standard diffusion model sampler.

Now we turn to estimating $\epsilon^*_{\mathbf{y}}$. Given the often noisy and ill-posed nature of Eq. 12 (Appendix C), we seek to select from the solution set $\{\epsilon_{\mathbf{y}} : \mathcal{A}[\hat{\mathbf{x}}_0] \approx \mathbf{y}\}$ through a noise-aware maximization algorithm, which we outline below.

3.2. Noise-Aware Maximization

We propose a **n**oise-**a**ware **m**aximization scheme (`nam`) to improve stability across noise levels. Given a single noisy measurement $\mathbf{y} = \mathcal{A}[\mathbf{x}]+\boldsymbol{\eta}$, there is a high risk of overfitting to noise $\boldsymbol{\eta}$ (Figures 3 and 5). To mitigate this problem, we propose a maximization scheme with a specialized early stopping criterion based on the measurement likelihood. We leverage the intuition that the corrected data-conditional score should yield a prediction via Eq. 32 where a vector of residuals,

$$\texttt{res} = \mathbf{y} - \mathcal{A}[\hat{\mathbf{x}}_0] \tag{13}$$

should be i.i.d. normally distributed with variance $\sigma_{\mathbf{y}}^2$. In other words, each index of `res` should come from the same distribution as an index of $\boldsymbol{\eta}$. Let this be the *null hypothesis* $\mathbb{H}_0$ — we thus seek to end the likelihood maximization process as soon as $\mathbb{H}_0$ holds.

Formally, we optimize Eq. 9 until the likelihood of the *alternate hypothesis* $\mathbb{H}_1$, that `res` is *not* distributed as η, is below a desired threshold p_{critical}. Since overfitting is more problematic at the end of sampling ($t \approx 0$) than the beginning of sampling ($t \approx T$), we set p_{critical} dynamically as a function of t, namely $p_{\text{critical}}(t) = \sigma_t$. This scheme is heavily inspired by the classical two-sided **z**-test (Hogg et al., 2013) with d samples, where d is the dimensionality of the image.

We use the early-stopping criterion at each time t

$$\mathbb{P}(|\xi| > |\texttt{res}|\big|\mathbb{H}_0) = 2\Phi(-|\texttt{res}|/\sigma_{\mathbf{y}}) < \sigma_t, \tag{14}$$

where $\xi_i \overset{\text{iid}}{\sim} \mathcal{N}(\mathbf{0}, \sigma_{\mathbf{y}}^2)$ and Φ is the CDF of a standard normal distribution. We note that this differs from a classical **z**-test since we are not seeking to reject the null hypothesis, but optimizing until the null hypothesis can no longer be rejected with sufficiently high probability. The full noise-aware maximization algorithm can be summarized by Algorithm 2. Since our loss function (Eq. 9) is quadratic, our proposed `nam` has worst-case linear convergence guarantees due to classical results in gradient descent (Boyd and Vandenberghe, 2004; Ryu and Boyd, 2016) with linear $\mathcal{A}$.

3.3. Theory

We highlight two key theoretical properties of our sampler, the *correctness* of the Tweedie's approximation and the *sufficiency* of the resulting score with respect to $\mathbf{y}$.

Correctness Our algorithm makes use of the conditional variant of Tweedie's formula, which obeys a very similar set of rules as Tweedie's (Theorem 2.1).

Theorem 3.1. *Let $\mathbf{x}_t$ be sampled from a conditional diffusion process given $\mathbf{y}$ (as in Eq. 4, with $\mathbf{x}_0 \sim p(\mathbf{x}_0|\mathbf{y})$). $\mathbb{E}[\mathbf{x}_0|\mathbf{x}_t, \mathbf{y}] = \mathbf{x}_0$ **if and only if** $p(\mathbf{x}_t)$ is a simple isotropic Gaussian with mean $\sqrt{\alpha_t}\mathbf{x}_0$ and variance $\sigma_t\mathbf{I}$.*

What differs from existing approaches is our setting, and in two key ways. First, we incorporate *conditional* information from $\mathbf{y}$. Second, rather than $p_t(\mathbf{x}_t)$, we consider $p_t(\mathbf{x}_t|\mathbf{x}_0)$, which is an isotropic Gaussian distribution *by construction*.

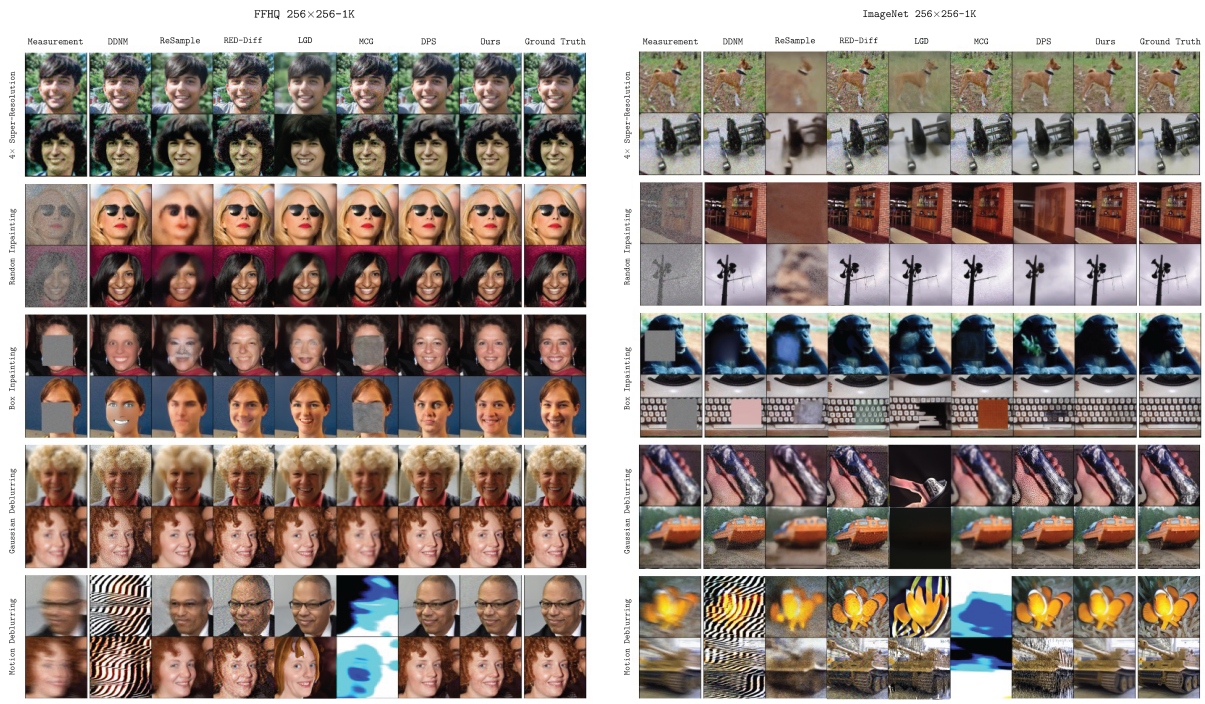


Figure 6: Qualitative comparison of our proposed method against related work on FFHQ 256×256-1K (left) and ImageNet 256×256-1K (right). Further comparisons can be found in Appendix F.

Therefore, given $\nabla \log p_t(\mathbf{x}_t|\mathbf{x}_0)$, Theorem 3.1 tells us that the application of Tweedie's formula in Eq. 9 will **exactly** recover $\mathbf{x}_0$.

Sufficiency Even with the optimization framework in Eq. 11, $\nabla \log p_t(\mathbf{x}_t|\mathbf{x}_0)$ can only be computed up to the information present in $\mathbf{y}$. However, we show that this is provably the best possible estimate. Namely, $\epsilon^*_{\mathbf{y}}$ (and therefore $s_{\text{corrected}}$) is a sufficient statistic for the ground truth $\mathbf{x}_0$ given measurement $\mathbf{y}$ under regularity conditions on $\mathcal{A}$ and $\boldsymbol{\eta}$:

Theorem 3.2 ($\epsilon_{\mathbf{y}_*}$ is a sufficient statistic)**.** *Let $\mathbf{y} = \mathcal{A}(\mathbf{x}_0) + \boldsymbol{\eta}$ be an observation from the forward measurement model, and let $\epsilon_{\mathbf{y}_*}$ be defined as in Eq. 12. Then $p(\mathbf{y}|\epsilon_{\mathbf{y}_*}) = p(\mathbf{y}|\mathbf{x}_0)$, given that either $\boldsymbol{\eta} = 0$, or $\mathcal{A}$ is linear.*

We extend this result to more general conditions in Theorem B.5. In this sense, **DCS** effectively closes an information "leak" by ensuring that the only information about $\mathbf{x}_0$ lost in the sampling process is *solely* that which is irrevocably destroyed by the operator $\mathcal{A}$.

3.4. Efficiency

Empirically, **DCS** enjoys two main computational advantages. First, it does not need to compute expensive gradients of the score function. Second, it boasts stable performance across choices of T due to the linearity of the data-conditional diffusion process.

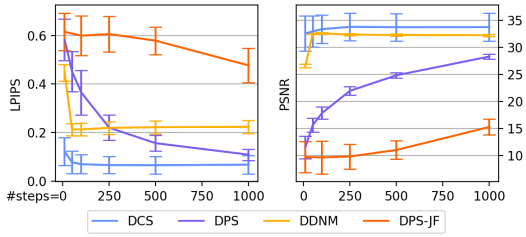


Figure 7: A study on the effect of T on solver performance. While other approaches exhibit poor performance due to the nonlinearity of the original reverse diffusion process, our method remains nearly invariant to T due to the near-linearity of the *data-conditional* diffusion process.

No Expensive $\nabla s_\theta(\mathbf{x}_t, t)$ Evaluations A drawback of many existing algorithms is the need to compute gradients of the score network during sampling (Table 1). This is the most expensive computation in the diffusion step, increasing the runtime of the algorithm by **2-3**×. However, this is unavoidable in posterior solvers without sacrificing performance (Section 4.3). Projection solvers sidestep this issue by framing a diffusion process in a subspace of $\mathcal{A}$ — however, this cannot be done when $\mathcal{A}$ is nonlinear.

A Near-Linear Reverse Process As **DCS** models $\nabla \log p_t(\mathbf{x}_t|\mathbf{x}_0)$, it is able to sample approximately from the *data-conditional* reverse diffusion process, which re-

verses the forward process defined in Eq. 4. When the *data-conditional* score is exactly estimated, Tweedie's recovers $\mathbf{x}_0$, and the diffusion process can be solved in a single step. In reality, our approximation of this process is correct up to the information about $\mathbf{x}_0$ present in $\mathbf{y}$ (Theorem B.5), under the assumptions detailed in the previous section.

In Figure 7, we validate the robustness of our algorithm to the total diffusion steps (T) with the super-resolution task on a subset of the FFHQ 256×256 dataset. We compare against DPS (Chung et al., 2023), DPS-JF (a neural backpropagation-free variant of DPS), and DDNM (Wang et al., 2022) at $\sigma_{\mathbf{y}} = 0.05$.

4. Experiments

We examine the empirical performance of **DCS** across a variety of natural image based inverse problems, against a range of state-of-the-art methods. Quantitatively, we leverage three key metrics to evaluate the quality of signal recovery: Learned Perceptual Image Patch Similarity (LPIPS), peak signal-to-noise ratio (PSNR), and Frechet Inception Distance (FID).

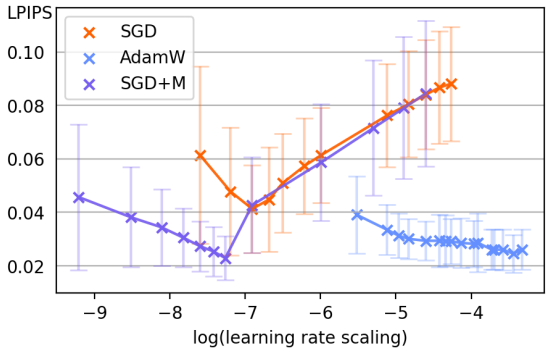
We run **DCS** and the other methods listed in Table 1 on the FFHQ-256 (Karras et al., 2019), (Kazemi and Sullivan, 2014) and ImageNet (Deng et al., 2009) datasets. For the prior network ϵ_θ, we use the corresponding pretrained model weights from (Chung et al., 2022a).

We examine five operator inversion tasks: Super-Resolution, Gaussian Deblurring, Motion Deblurring, Random Inpainting, and Box Inpainting. We first run experiments with additive Gaussian noise of standard deviation $\sigma_{\mathbf{y}} = 0.01$ (we present results at a higher noise level in Section 4.1). We also present quantitative results on FFHQ and ImageNet in Table 2, and a qualitative comparison in Figure 6. We delegate further experiments, such as evaluations on subsets of FFHQ used in other works, additional qualitative comparisons, and details of the implementation to Appendix D, E and F.

We find that **DCS** either outperforms, or is comparable to all existing methods. While some methods may excel at certain metrics in certain tasks and fail to recover the signal at other times, **DCS** is consistently performs well across experiments. For example, **DCS** is one of few methods that has reasonable results on Motion Deblurring at high noise levels. DDNM and RED-Diff, on the other hand, are powerful across inpainting tasks in general, but fail to perform Motion Deblurring and show underwhelming qualitative performance on many tasks, especially in the high noise regime (Figures 5, 6).

We also notice that **DCS** provides a significant speedup and reduction in memory footprint compared to most methods,

Figure 8: **DCS** performance across several choices of optimizers. LPIPS score of the predicted $\mathbf{x}_0$ images is plotted against the natural log of learning rate scaling factor for each optimizer.



as noted in Table 1. We achieve this by not requiring backpropagation of the score network, as well as limiting the required number of neural function evaluations by using the measurement-consistent version of Tweedie's formula.

4.1. Higher Noise Levels

We investigate the noise-robustness of **DCS** by running the same benchmarks at higher noise levels. We display quantitative results for FFHQ and ImageNet datasets in Table 2 at $\sigma_{\mathbf{y}} = 0.1$ (where experimental settings outside of noise level are kept identical to the previous section), and qualitative results for the FFHQ dataset in Figure 5. We again see **DCS** achieve comparable or superior results at every task. Projection methods such as DDNM and DDRM further deteriorate, as they overfit and attempt to reproduce the noise. Other methods such as LGD-MC degrade more gracefully, however we can see from qualitative examples that they are likely underfitting in all regimes, and therefore only gain noise-robustness by sacrificing performance at lower noise levels. Both **DCS** and DPS strike a much clearer balance between overfitting and underfitting, which is apparent from quantitative results as well as qualitative results in Figures 5 and 6.

4.2. Ablation on the Noise-aware Maximization Optimizer

We investigate how the choice of optimizer and parameters affects the noise-aware maximization algorithm in **DCS**. We note that the flexibility of using an optimizer enables us to make use of a frequentist stopping criterion as detailed in Section 3. In Figure 8 we run **DCS** with AdamW (Loshchilov et al., 2017), SGD with momentum, and vanilla SGD to solve the SRx4 task on a subset of FFHQ. Runs of each optimizer at learning rate scaling factors are displayed to show the best performance, ensuring a fair comparison. It is clear in Figure 8 that the addition of a momentum term to the optimization process (both present in AdamW and

Table 2: Quantitative comparison on FFHQ 256x256-1K and ImageNet-1K datasets across various inverse problem tasks and noise levels ($\sigma_{\mathbf{y}} \in \{0.01, 0.1\}$).

FFHQ	SR ×4			Random Inpainting			Box Inpainting			Gaussian Deblurring			Motion Deblurring		
$\sigma_{\mathbf{y}} = 0.01$	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓
Ours	**0.137**	**30.14**	**19.45**	**0.024**	**34.84**	**21.19**	**0.088**	25.11	**19.25**	**0.103**	28.69	**22.62**	**0.087**	**29.48**	**26.67**
DDRM	0.502	13.00	222.5	0.393	15.94	163.9	0.472	12.15	209.2	-	-	-	-	-	-
MCG	0.144	24.84	31.47	0.073	30.59	22.22	0.453	15.44	185.54	0.209	23.51	67.88	0.217	22.93	292.1
DDNM	0.208	26.28	51.33	0.040	33.08	23.35	0.209	18.12	88.32	0.235	26.09	71.47	0.424	14.22	250.9
LGD-MC	0.238	23.45	39.55	0.272	23.46	57.70	0.372	15.45	86.00	0.405	18.78	64.31	0.520	13.90	106.0
DPS	0.163	25.91	33.21	0.105	29.54	29.72	0.113	23.52	24.41	0.129	26.48	26.85	0.159	24.41	29.84
RED-Diff	0.178	29.81	45.68	0.035	33.72	25.03	0.090	**25.20**	19.98	0.234	**29.72**	52.09	0.191	29.14	116.9
FFHQ	SR ×4			Random Inpainting			Box Inpainting			Gaussian Deblurring			Motion Deblurring		
$\sigma_{\mathbf{y}} = 0.1$	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓
Ours	**0.175**	**24.88**	**30.11**	**0.149**	27.54	**32.80**	0.163	**23.22**	**26.44**	**0.176**	25.96	**26.08**	0.224	**24.61**	**31.40**
DDRM	0.785	6.327	271.7	0.602	11.00	255.9	0.632	9.636	288.1	-	-	-	-	-	-
MCG	0.546	20.44	102.6	0.227	26.00	50.40	0.579	15.30	207.2	0.429	25.80	69.29	0.973	-7.104	295.3
DDNM	0.623	21.49	145.9	0.179	24.96	39.18	0.334	19.20	72.11	1.220	10.73	176.8	0.739	5.099	524.0
LGD-MC	0.256	22.31	39.58	0.288	22.22	56.05	0.384	15.38	87.72	0.415	18.30	66.04	0.524	13.65	105.4
DPS	0.185	24.79	35.46	0.157	26.72	35.24	**0.158**	22.58	32.47	0.180	24.720	33.53	**0.212**	22.41	35.09
RED-Diff	0.665	22.10	155.1	0.155	**28.62**	34.78	0.298	22.96	61.14	0.447	**26.93**	106.3	0.423	24.16	120.1
ImageNet	SR ×4			Random Inpainting			Box Inpainting			Gaussian Deblurring			Motion Deblurring		
$\sigma_{\mathbf{y}} = 0.01$	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓
Ours	**0.238**	23.45	**39.41**	0.142	26.06	34.46	**0.230**	**20.63**	37.11	**0.253**	24.22	**38.96**	**0.203**	**24.62**	**38.63**
DDRM	0.907	6.592	277.8	0.835	10.15	215.8	0.758	11.70	198.8	-	-	-	-	-	-
MCG	0.638	15.62	89.39	0.198	24.34	35.19	0.273	16.68	80.35	0.645	21.18	124.6	0.980	-5.726	231.1
DDNM	0.333	**25.16**	51.33	**0.084**	**28.35**	20.27	0.258	17.42	85.41	0.456	24.35	67.98	0.694	5.721	304.2
LGD-MC	0.662	14.460	113.5	0.650	14.57	129.8	0.696	11.63	133.9	0.796	10.46	165.7	0.807	9.609	184.7
DPS	0.309	23.99	49.81	0.266	25.05	38.87	0.301	18.76	**34.85**	0.493	19.14	61.59	0.460	18.65	53.21
RED-Diff	0.386	25.07	57.06	0.090	28.17	**16.71**	0.239	19.99	54.38	0.459	**24.70**	68.71	0.376	23.66	55.77
ImageNet	SR ×4			Random Inpainting			Box Inpainting			Gaussian Deblurring			Motion Deblurring		
$\sigma_{\mathbf{y}} = 0.1$	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓
Ours	**0.402**	**22.99**	**48.21**	**0.166**	**26.04**	34.47	**0.243**	**19.70**	**46.03**	**0.407**	22.28	**51.13**	**0.435**	20.43	**61.48**
DDRM	0.985	5.981	425.8	0.937	7.391	358.1	0.841	8.646	241.0	-	-	-	-	-	-
MCG	0.886	14.01	145.1	0.459	19.92	78.86	0.433	15.63	124.0	0.650	22.00	117.4	0.984	-6.868	231.3
DDNM	0.751	20.98	133.3	0.1693	25.63	35.72	0.400	18.06	110.8	1.221	9.602	202.7	0.783	5.009	350.1
LGD-MC	0.671	14.01	116.51	0.661	14.19	131.2	0.701	11.53	134.58	0.804	10.44	167.3	0.806	9.587	185.4
DPS	0.540	18.63	85.06	0.506	20.10	82.74	0.479	18.03	83.06	0.412	20.57	65.07	0.450	18.91	75.65
RED-Diff	0.747	20.66	136.35	0.167	25.38	**32.99**	0.374	19.68	88.20	0.660	**23.19**	110.9	0.591	**21.27**	138.8

SGD with momentum) can attain a higher level of image fidelity and solver stability than vanilla SGD. This provides empirical evidence for optimizer bias having an effect on solver performance in **DCS**. We see from this experiment that AdamW produces the most consistent results across learning rates, which motivates its use in our implementation.

4.3. Improvement on Jacobian-Free Implementations

A major gain in the empirical performance of **DCS** comes from the fact that it no longer requires backpropagations through the neural score function, which allows for reductions in both runtime and memory footprint. In theory, most diffusion-based solvers can be modified to remove this neural backpropagation step by applying a `stop_gradient` to the score function output (e.g., RED-Diff (Mardani et al., 2024)). We compare against RED-Diff in the main text, and additionally against ablated variants of DPS (Chung et al., 2022a) and LGD (Song et al., 2023b) in Appendix D.2, and demonstrate clear improvements on these methods.

5. Conclusion

We proposed an effective adjustment to the diffusion-based inverse problem solver framework in the literature that improves speed and stability. Observing that the marginals of the diffusion process which solves the inverse problem is Gaussian distributed at each time t, we derived a simple, single-parameter likelihood model, whose sole unknown variate is obtained via a tractable maximum likelihood estimation algorithm. We leveraged this new perspective to create a noise-aware maximization scheme, and demonstrated the effectiveness of our method via a suite of numerical experiments, and qualitative comparisons.

Impact Statement Our work proposes a significant improvement to a general framework for noise-robust guided generation via diffusion models. While the inverse problems we study in this work are not pernicious in nature, many ma-

licious tasks can be framed as inverse problems, including deepfake generation. Therefore, we hope that researchers and practitioners keep this in mind when implementing and applying the methods proposed in this work, and when applying techniques in the general research area of inverse problems at large.

References

Brian DO Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3): 313–326, 1982.

Harry C Andrews and Bobby Ray Hunt. *Digital image restoration*. Prentice Hall Professional Technical Reference, 1977.

Richard C Aster, Brian Borchers, and Clifford H Thurber. *Parameter estimation and inverse problems*. Elsevier, 2018.

Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

Ashish Bora, Ajil Jalal, Eric Price, and Alexandros G Dimakis. Compressed sensing using generative models. In *International conference on machine learning*, pages 537–546. PMLR, 2017.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Gabriel Cardoso, Yazid Janati El Idrissi, Sylvain Le Corff, and Eric Moulines. Monte carlo guided diffusion for bayesian linear inverse problems. *arXiv preprint arXiv:2308.07983*, 2023.

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

Hyungjin Chung, Byeongsu Sim, Dohoon Ryu, and Jong Chul Ye. Improving diffusion models for inverse problems using manifold constraints. *Advances in Neural Information Processing Systems*, 35:25683–25696, 2022b.

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. URL https://openreview.net/forum?id=OnD9zGAGT0k.

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.

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

Bradley Efron. Tweedie's formula and selection bias. *Journal of the American Statistical Association*, 106(496): 1602–1614, 2011.

Heinz Werner Engl, Martin Hanke, and Andreas Neubauer. *Regularization of inverse problems*, volume 375. Springer Science & Business Media, 1996.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. *arXiv preprint arXiv:2403.03206*, 2024.

Bob A Hardage. Vertical seismic profiling. *The Leading Edge*, 4(11):59–59, 1985.

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

Robert V Hogg, Joseph W McKean, Allen T Craig, et al. *Introduction to mathematical statistics*. Pearson Education India, 2013.

Mark Kac. Can one hear the shape of a drum? *The american mathematical monthly*, 73(4P2):1–23, 1966.

Zahra Kadkhodaie, Florentin Guth, Eero P Simoncelli, and Stéphane Mallat. Generalization in diffusion models arises from geometry-adaptive harmonic representation. *arXiv preprint arXiv:2310.02557*, 2023.

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.

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

Vahid Kazemi and Josephine Sullivan. One millisecond face alignment with an ensemble of regression trees. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1867–1874, 2014.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Ilya Loshchilov, Frank Hutter, et al. Fixing weight decay regularization in adam. *arXiv preprint arXiv:1711.05101*, 5, 2017.

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. URL https://openreview.net/forum?id=1YO4EE3SPB.

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.

Litu Rout, Yujia Chen, Abhishek Kumar, Constantine Caramanis, Sanjay Shakkottai, and Wen-Sheng Chu. Beyond first-order tweedie: Solving inverse problems using latent diffusion. *arXiv preprint arXiv:2312.00852*, 2023.

Litu Rout, Negin Raoof, Giannis Daras, Constantine Caramanis, Alex Dimakis, and Sanjay Shakkottai. Solving linear inverse problems provably via posterior sampling with latent diffusion models. *Advances in Neural Information Processing Systems*, 36, 2024.

Ernest K Ryu and Stephen Boyd. Primer on monotone operator methods. *Appl. comput. math*, 15(1):3–43, 2016.

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

Bowen Song, Soo Min Kwon, Zecheng Zhang, Xinyu Hu, Qing Qu, and Liyue Shen. Solving inverse problems with latent diffusion models via hard data consistency. *arXiv preprint arXiv:2307.08123*, 2024.

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

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, 2023b.

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

Paul Suetens. *Fundamentals of medical imaging*. Cambridge university press, 2017.

Yu Sun, Zihui Wu, Yifan Chen, Berthy T Feng, and Katherine L Bouman. Provable probabilistic imaging using score-based generative priors. *IEEE Transactions on Computational Imaging*, 2024.

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.

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

Curtis R Vogel. *Computational methods for inverse problems*. SIAM, 2002.

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

A. Background and Related Work

A.1. Diffusion Models

Inspired by non-equilibrium thermodynamics, denoising diffusion probabilistic models (Ho et al., 2020) convert data $\mathbf{x}_0 \sim p_{\text{data}}(\mathbf{x})$ to noise $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ via a diffusion process described by the variance-preserving stochastic differential equation (VP-SDE)

$$d\mathbf{x} = -\frac{\beta(t)}{2}\mathbf{x}dt + \sqrt{\beta(t)}d\mathbf{w}, \tag{15}$$

where $\beta(t) : \mathbb{R} \to [0, 1]$ is a monotonically increasing noise schedule and $\mathbf{w}$ is the standard Wiener process (Song et al., 2020b). This leads to the marginal distribution

$$p_t(\mathbf{x}_t) = \mathbb{E}_{\mathbf{x}_0 \sim p_{\text{data}}}\left[\mathcal{N}(\mathbf{x}_t; \sqrt{\alpha_t}\mathbf{x}_0, \underbrace{(1-\alpha_t)}_{\sigma_t^2}\mathbf{I})\right], \quad \alpha_t = e^{-\frac{1}{2}\int_0^t \beta(s)ds}, \tag{16}$$

where $\mathcal{N}(\,\cdot\,; \mu, \Sigma)$ is the probability density function (pdf) of a normal distribution centered at μ with covariance Σ. Sampling from $p_{\text{data}}(\mathbf{x})$ can then occur by modeling the reverse diffusion, which has a simple form given by (Anderson, 1982)

$$d\overline{\mathbf{x}} = \left[-\frac{\beta(t)}{2}\mathbf{x} - \beta(t)\nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t)\right]dt + \sqrt{\beta(t)}d\overline{\mathbf{w}}, \tag{17}$$

with reverse-time Wiener process $\overline{\mathbf{w}}$ and score function $\nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t)$. Therefore, diffusion model training consists of approximating the score function with a model

$$s_\theta(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t), \tag{18}$$

and sampling consists of obtaining solutions to the reverse-time SDE (17) with numerical solvers. A simple approach is given by the DDIM sampler with $\sigma_t = \sqrt{1-\alpha_t}$ (Song et al., 2020a)

$$\mathbf{x}_{t-1} = \sqrt{\alpha_{t-1}}\frac{\mathbf{x}_t + \sigma_t^2 \nabla \log p_t(\mathbf{x}_t)}{\sqrt{\alpha_t}} + \sigma_{t-1}\boldsymbol{\epsilon}. \tag{19}$$

A.2. Solving Inverse Problems with Diffusion Models

When solving inverse problems with diffusion models, the aim is to leverage information from $\mathbf{y}$ to define a **modified** reverse diffusion process

$$\mathbf{x}_T, \mathbf{x}_{T-1}, \ldots, \mathbf{x}_1, \mathbf{x}_0, \tag{20}$$

such that $\mathbf{x}_t$ coincides with the desired $\mathbf{x}$ (Eq. 1) precisely at $t = 0$. Previous approaches can generally be sorted into two categories, which we designate **posterior solvers** and **projection solvers**.

Posterior Solvers An intuitive approach is leveraging Bayes' rule to sample from the **posterior** distribution given a prior $p_t(\mathbf{x}_t)$ and observation $\mathbf{y}$:

$$\mathbf{x}_t \sim p(\mathbf{x}_t|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x}_t)p(\mathbf{x}_t)}{p(\mathbf{y})}. \tag{21}$$

Taking logs and gradients of both sides of the equation, we obtain a form of the conditional density that can be accurately approximated with the modeled score function

$$\nabla \log p(\mathbf{x}_t|\mathbf{y}) = \nabla \log p(\mathbf{y}|\mathbf{x}_t) + \nabla \log p(\mathbf{x}_t) \approx \nabla \log p(\mathbf{y}|\mathbf{x}_t) + s_\theta(\mathbf{x}_t, t), \tag{22}$$

and describes the core method of the DPS algorithm (Chung et al., 2022a). This strategy can also be extended to latent diffusion models, resulting in Latent-DPS and PSLD (Rout et al., 2023). Generally, the conditional term $\nabla \log p(\mathbf{y}|\mathbf{x}_t)$ cannot be exact due to reasons we will investigate subsequently in Section 2, though these approximations are improved in LGD (Song et al., 2023b) and STSL (Rout et al., 2024). More recent work (Sun et al., 2024) propose an annealed Monte-Carlo-based perspective to posterior sampling, which results in a very similar algorithm to DPS. Much like MCG and ReSample (discussed in the next category), posterior solvers require estimating $\frac{\partial}{\partial \mathbf{x}_t}\mathbf{x}_0$ which involves backpropagation through the diffusion model, and significantly increases runtime and hampers scalability compared to unconditional sampling.

Projection Solvers Another approach involves guiding the reverse diffusion process by directly **projecting** $\mathbf{x}_t$ onto a manifold $\mathcal{M} = \{\mathbf{x} : \mathcal{A}(\mathbf{x}) = y\} \subseteq \mathbf{R}^d$ at each time step, i.e.

$$\mathbf{x}'_t = \mathbf{P}\hat{\mathbf{x}}_0[\mathbf{x}_t] \tag{23}$$

$$\mathbf{x}_{t-1} = \sqrt{\alpha_{t-1}}\frac{\mathbf{x}'_t + \sigma_t^2 \nabla \log p(\mathbf{x}'_t|\hat{\mathbf{x}}_0[\mathbf{x}_t])}{\sqrt{\alpha_t}} + \sigma_{t-1}\boldsymbol{\epsilon}. \tag{24}$$

Where $\hat{\mathbf{x}}_0[\mathbf{x}_t]$ is some prediction of $\mathbf{x}_0$ given only $\mathbf{x}_t$ (we elaborate in Section 2), and $\mathbf{P}$ is either a projection onto the low rank subspace or range of $\mathcal{A}$. The resulting algorithms are DDRM (Kawar et al., 2022) and DDNM (Wang et al., 2022), respectively. Of course, this strategy is often restricted to situations where two conditions simultaneously hold true: (1) the measurement operator $\mathcal{A}$ is linear, and (2) the inverse problem is noiseless, i.e, η is identically $\mathbf{0}$. These assumptions drastically limit the applicability of such models. The linearity restriction can be lifted by taking derivatives the measurement discrepancy, as in MCG (Chung et al., 2022b) and ReSample (Song et al., 2024), though this comes at the cost of significantly increased computation, requiring $\frac{\partial}{\partial \mathbf{x}_t}\mathbf{x}_0$ which involves backpropagating through the score network. Finally, (Cardoso et al., 2023) straddles the line between both categories — while `MCGdiff` is ostensibly a Bayesian solver, it bears greater resemblance to projection solvers since it does not form the decomposition in Eq. 22 and also samples by projecting each iterate to the null-space of $\mathcal{A}$, thus implementing a projected n-particle sequential monte carlo (SMC) sampling algorithm.

A Maximum Likelihood Solver We take a different perspective on solving the inverse problem. As seen in Section 2, both **projection** and **posterior** solvers must quantify the discrepancy between $\mathbf{x}_t$ and $\mathbf{y}$ via the consistency error $||\mathcal{A}(\mathbf{x}_0) - \mathbf{y}||$ at each diffusion step. Due to the complexity of the diffusion process, this involves approximating a fundamentally intractable quantity. In Section 3, we construct a simpler process whose parameters can be obtained via maximum likelihood estimation. Unlike the evidence lower bound proposed in (Mardani et al., 2024), we derive an explicit likelihood model, which is amenable to an optimization scheme with a probabilistic noise-aware stopping criterion. Finally, we show that the resulting algorithm is simple, fast, and adaptable to noise.

B. Additional Theorems and Proofs

B.1. Proof of Tweedie's Formula

For completeness, we include the statement and proof for Tweedie's formula.

Theorem B.1 (Tweedie's Formula)**.** *Let $\mathbf{x}_0$ be a sample drawn from a distribution $p(\mathbf{x}_0)$. Then for any*

$$\mathbf{x}_t = \alpha_t \mathbf{x}_0 + \sigma_t \mathbf{z} \qquad \mathbf{z} \sim \mathcal{N}(0, \mathbf{I}) \tag{25}$$

*drawn from the marginal of the diffusion process on $p(\mathbf{x}_0)$ at time t, the **posterior mean** given $\mathbf{x}_t$ is*

$$\mathbb{E}[\mathbf{x}_0|\mathbf{x}_t] = \frac{1}{\alpha_t}\left[\mathbf{x}_t + \sigma_t^2 \nabla \log p_t(\mathbf{x}_t)\right]. \tag{26}$$

Proof (of Lemma B.1). Let ϕ_σ be the pdf of $\mathcal{N}(0, \sigma\mathbf{I})$. We first note that the marginal distribution at time t can be written as

$$p_t(\mathbf{x}_t) = \left(p_{\alpha_t} * \phi_{\sigma_t}\right)(\mathbf{x}_t) = \int \phi_{\sigma_t}(\mathbf{x}_t - \mathbf{x}) p_{\alpha_t}(\mathbf{x}) d\mathbf{x}, \tag{27}$$

where

$$p_{\alpha_t}(\mathbf{x}|\mathbf{y}) = \frac{1}{\alpha_t} p\left(\alpha_t^{-1}\mathbf{x}\right) \tag{28}$$

due to the probabilistic change-of-variables formula. Letting $\mathbf{x}' = \alpha_t \mathbf{x}_0$, we have the equality

$$\begin{aligned}
\frac{\mathbb{E}[\mathbf{x}'|\mathbf{x}_t, \mathbf{y}] - \mathbf{x}_t}{\sigma_t^2} &= \int \frac{\mathbf{x}' - \mathbf{x}_t}{\sigma_t^2} p(\mathbf{x}'|\mathbf{x}_t) d\mathbf{x}' \\
&= \int \frac{\mathbf{x}' - \mathbf{x}_t}{\sigma_t^2} \frac{p(\mathbf{x}', \mathbf{x}_t)}{p(\mathbf{x}_t)} d\mathbf{x}' \\
&= \int \frac{\alpha_t \mathbf{x}_0 - \mathbf{x}_t}{\sigma_t^2} \frac{\phi_{\sigma_t}(\mathbf{x}_t - \mathbf{x}') p_{\alpha_t}(\mathbf{x}')}{\int \phi_{\sigma_t}(\mathbf{x}_t - \mathbf{x}) p_{\alpha_t}(\mathbf{x}) d\mathbf{x}} d\mathbf{x}' \\
&= \int [\nabla_{\mathbf{x}_t} \phi_{\sigma_t}(\mathbf{x}_t - \mathbf{x}')] \frac{\phi_{\sigma_t}(\mathbf{x}_t - \mathbf{x}') p_{\alpha_t}(\mathbf{x}')}{\int \phi_{\sigma_t}(\mathbf{x}_t - \mathbf{x}) p_{\alpha_t}(\mathbf{x}) d\mathbf{x}} d\mathbf{x}' \\
&= \nabla_{\mathbf{x}_t} \log \left[\phi_{\sigma_t}(\mathbf{x}_t - \mathbf{x}') p_{\alpha_t}(\mathbf{x}') \right] \\
&= \nabla \log p_t(\mathbf{x}_t).
\end{aligned}$$

Re-arranging terms on either side of the equation, we obtain

$$\mathbb{E}[\mathbf{x}'|\mathbf{x}_t] = \mathbf{x}_t + \sigma_t^2 \nabla \log p_t(\mathbf{x}_t). \tag{29}$$

Finally, we expand $\mathbf{x}' = \alpha_t \mathbf{x}_0$ and invoke the linearity of the expectation to arrive at

$$\mathbb{E}[\mathbf{x}_0|\mathbf{x}_t] = \frac{1}{\alpha_t} \left[\mathbf{x}_t + \sigma_t^2 \nabla \log p_t(\mathbf{x}_t|\mathbf{y}) \right]. \tag{30}$$

as desired. □

B.2. Proof for Theorem 2.1 ($\mathbf{x}_t \sim \mathcal{N}(\boldsymbol{\mu}(\mathbf{x}_0), \sigma_t^2 \mathbf{I}) \iff \mathbb{E}[\mathbf{x}_0|\mathbf{x}_t] = \mathbf{x}_0$)

We demonstrate sufficiency of the Gaussian-distributed condition by proving Lemma B.2.

Lemma B.2 (Sufficent condition)**.** *Let $\mathbf{x}_0$ be given. Suppose $\mathbf{x}_t$ is distributed as*

$$p_t(\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_t; \sqrt{\alpha_t}\mathbf{x}_0, \underbrace{1 - \alpha_t}_{\sigma_t^2} \mathbf{I}). \tag{31}$$

Then $\mathbf{x}_0$ can be recovered via

$$\mathbf{x}_0 = \frac{1}{\sqrt{\alpha_t}} \left[\mathbf{x}_t + \sigma_t^2 \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) \right]. \tag{32}$$

Proof (of Lemma B.2).

$$\begin{aligned}
\frac{1}{\sqrt{\alpha_t}} \left[\mathbf{x}_t + \sigma_t^2 \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) \right] &= \frac{1}{\sqrt{\alpha_t}} \left[\mathbf{x}_t - \nabla_{\mathbf{x}_t} \sigma_t^2 \frac{1}{2\sigma_t^2} ||\mathbf{x}_t - \sqrt{\alpha_t}\mathbf{x}_0||_2^2 \right] && (33) \\
&= \frac{1}{\sqrt{\alpha_t}} \left[\mathbf{x}_t - (\mathbf{x}_t - \sqrt{\alpha_t}\mathbf{x}_0) \right] && (34) \\
&= \mathbf{x}_0. && (35)
\end{aligned}$$

□

To demonstrate the necessary condition, we show that the inverse of Lemma B.2 also holds.

Lemma B.3 (Necessary condition)**.** *If $\mathbf{x}_0$ can be recovered via Eq. 32, then $p_t(\mathbf{x}_t|\mathbf{x}_0)$ takes the form Eq. 31.*

Proof (of Lemma B.3). Suppose that

$$\mathbf{x}_0 = \frac{1}{\sqrt{\alpha_t}} \left[\mathbf{x}_t + \sigma_t^2 \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) \right] \tag{36}$$

Then we may re-arrange terms, obtaining

$$\frac{\sqrt{\alpha_t}\mathbf{x}_0 - \mathbf{x}_t}{\sigma_t^2} = \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t). \tag{37}$$

Taking the anti-derivative of both sides, we conclude that

$$\log p_t(\mathbf{x}_t) = \frac{1}{2\sigma_t^2}||\mathbf{x}_t - \sqrt{\alpha_t}\mathbf{x}_0||_2^2 + C. \tag{38}$$

Since $\log p_t(\mathbf{x}_t|\mathbf{x}_0)$ can only take this form when $p_t(\mathbf{x}_t|\mathbf{x}_0)$ is a simple isotropic Gaussian distribution, we conclude our proof. □

Proof (of Theorem 2.1). First, we note that Tweedie's formula (Efron, 2011) tells us that the posterior mean of a data distribution $\mathbf{x}_t \sim p_t(\mathbf{x}_t|\mathbf{x}_0)$ can be obtained via the relation

$$\mathbf{E}[\mathbf{x}_0|\mathbf{x}_t] = \frac{1}{\sqrt{\alpha_t}}\left[\mathbf{x}_t + \sigma_t^2 \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)\right]. \tag{39}$$

Then, since Lemmas B.2 and B.3 are converses of each other, we demonstrate that the conditions stated in Lemma B.2 are necessary and sufficient. □

B.3. Proof for Theorem 3.1 ($\mathbf{x}_t \sim \mathcal{N}(\boldsymbol{\mu}(\mathbf{x}_0, \mathbf{y}), \sigma_t^2\mathbf{I}) \iff \mathbb{E}[\mathbf{x}_0|\mathbf{x}_t, \mathbf{y}] = \mathbf{x}_0$)

First, we show that the *measurement conditional* Tweedie's formula holds for a given diffusion variate $\mathbf{x}_t$ and measurement $\mathbf{y}$.

Theorem B.4 (Conditional Tweedie's Formula)**.** *Let $\mathbf{x}_0$ be a sample drawn from a conditional distribution $p(\mathbf{x}_0|\mathbf{y})$. Then for any*

$$\mathbf{x}_t = \alpha_t\mathbf{x}_0 + \sigma_t\mathbf{z} \qquad \mathbf{z} \sim \mathcal{N}(0, \mathbf{I}) \tag{40}$$

*drawn from the marginal of the diffusion process on $p(\mathbf{x}_0|\mathbf{y})$ at time t, the **conditional posterior mean** given $\mathbf{x}_t$ is*

$$\mathbb{E}[\mathbf{x}_0|\mathbf{x}_t, \mathbf{y}] = \frac{1}{\alpha_t}\left[\mathbf{x}_t + \sigma_t^2 \nabla \log p_t(\mathbf{x}_t|\mathbf{y})\right]. \tag{41}$$

Proof (of Lemma B.4). Let ϕ_σ be the pdf of $\mathcal{N}(0, \sigma\mathbf{I})$. We first note that the marginal distribution at time t can be written as

$$p_t(\mathbf{x}_t|\mathbf{y}) = \left(p_{\alpha_t}(\cdot|\mathbf{y}) * \phi_{\sigma_t}\right)(\mathbf{x}_t) = \int \phi_{\sigma_t}(\mathbf{x}_t - \mathbf{x}) p_{\alpha_t}(\mathbf{x}|\mathbf{y}) d\mathbf{x}, \tag{42}$$

where

$$p_{\alpha_t}(\mathbf{x}|\mathbf{y}) = \frac{1}{\alpha_t} p\left(\alpha_t^{-1}\mathbf{x}|\mathbf{y}\right) \tag{43}$$

due to the probabilistic change-of-variables formula. Letting $\mathbf{x}' = \alpha_t\mathbf{x}_0$, we have the equality

$$\begin{aligned}
\frac{\mathbb{E}[\mathbf{x}'|\mathbf{x}_t, \mathbf{y}] - \mathbf{x}_t}{\sigma_t^2} &= \int \frac{\mathbf{x}' - \mathbf{x}_t}{\sigma_t^2} p(\mathbf{x}'|\mathbf{x}_t, \mathbf{y}) d\mathbf{x}' \\
&= \int \frac{\mathbf{x}' - \mathbf{x}_t}{\sigma_t^2} \frac{p(\mathbf{x}', \mathbf{x}_t|\mathbf{y})}{p(\mathbf{x}_t|\mathbf{y})} d\mathbf{x}' \\
&= \int \frac{\alpha_t\mathbf{x}_0 - \mathbf{x}_t}{\sigma_t^2} \frac{\phi_{\sigma_t}(\mathbf{x}_t - \mathbf{x}')p_{\alpha_t}(\mathbf{x}'|\mathbf{y})}{\int \phi_{\sigma_t}(\mathbf{x}_t - \mathbf{x})p_{\alpha_t}(\mathbf{x}|\mathbf{y})d\mathbf{x}} d\mathbf{x}' \\
&= \int \left[\nabla_{\mathbf{x}_t}\phi_{\sigma_t}(\mathbf{x}_t - \mathbf{x}')\right] \frac{\phi_{\sigma_t}(\mathbf{x}_t - \mathbf{x}')p_{\alpha_t}(\mathbf{x}'|\mathbf{y})}{\int \phi_{\sigma_t}(\mathbf{x}_t - \mathbf{x})p_{\alpha_t}(\mathbf{x}|\mathbf{y})d\mathbf{x}} d\mathbf{x}' \\
&= \nabla_{\mathbf{x}_t} \log \left[\phi_{\sigma_t}(\mathbf{x}_t - \mathbf{x}')p_{\alpha_t}(\mathbf{x}'|\mathbf{y})\right] \\
&= \nabla \log p_t(\mathbf{x}_t|\mathbf{y}).
\end{aligned}$$

Re-arranging terms on either side of the equation, we obtain

$$\mathbb{E}[\mathbf{x}'|\mathbf{x}_t, \mathbf{y}] = \mathbf{x}_t + \sigma_t^2 \nabla \log p_t(\mathbf{x}_t|\mathbf{y}). \tag{44}$$

Finally, we expand $\mathbf{x}' = \alpha_t \mathbf{x}_0$ and invoke the linearity of the expectation to arrive at

$$\mathbb{E}[\mathbf{x}_0|\mathbf{x}_t, \mathbf{y}] = \frac{1}{\alpha_t}\left[\mathbf{x}_t + \sigma_t^2 \nabla \log p_t(\mathbf{x}_t|\mathbf{y})\right]. \tag{45}$$

as desired. □

Now, the main theorem follows.

Proof (of Theorem 2.1). We note that the proofs for Lemmas B.2 and B.3 remain the same if we let $p_t(\mathbf{x}_t|\mathbf{x}_0) = p_t(\mathbf{x}_t|\mathbf{x}_0, \mathbf{y})$. Thus we again observe that the proofs for Lemmas B.2 and B.3 are converses of each other, and demonstrate that the conditions stated in Lemma B.2 are necessary and sufficient. □

B.4. Theorems for Sufficiency

We set up Theorems to show that the estimator in Eq. 11 is a sufficient statistic under different properties of $\mathcal{A}$. Letting $\mathbf{f}(\mathbf{y})$ be the function that obtains $\nabla \log p_t(\mathbf{x}_t|\mathbf{x}_0)$ via Eq. 11, we show that $\mathbf{y}$ is measurable under the sigma algebra induced by the measurement $\mathbf{f}$.

Intuitively, we demonstrate that $\mathbf{f}(\mathbf{y})$ contains as much information as possible about the underlying signal $\mathbf{x}_0$ as can be gathered via $\mathbf{y}$. The theoretical and intuitive statements can be summarized by the simple conditional equivalence

$$p(\mathbf{y}|\epsilon_{\mathbf{y}*}) = p(\mathbf{y}|\mathbf{x}_0). \tag{46}$$

In Theorem 3.2, we consider two simple and theoretically similar cases: when $\mathbf{y} = \mathcal{A}(\mathbf{x})$ is noise-free, and when $\mathcal{A}$ is linear. We restate it here in a less condensed form for clarity:

Theorem 3.2. Let $\mathbf{y} = \mathcal{A}(\mathbf{x}_0) + \boldsymbol{\eta}$ be an observation from the forward measurement model, and let

$$\epsilon_{\mathbf{y}*} = \arg\max_{\epsilon_{\mathbf{y}}} \log p\left(\mathbf{y}\middle|\frac{1}{\sqrt{\alpha_t}}(\mathbf{x}_t + \sigma_t^2 \epsilon_{\mathbf{y}})\right). \tag{47}$$

Then

$$p(\mathbf{y}|\epsilon_{\mathbf{y}*}) = p(\mathbf{y}|\mathbf{x}_0). \tag{48}$$

given that either $\boldsymbol{\eta} = 0$, or $\mathcal{A}$ is linear.

We now investigate the general noisy case where $\mathcal{A}$ is allowed to be nonlinear. We find that our results can still be quite general: we only need to assume $\mathcal{A}$ surjective, meaning that there exists some $\mathbf{x} \in \text{domain}(\mathcal{A})$ such that $\mathcal{A}(\mathbf{x}) = \mathbf{y}$. In fact, this result is slightly stronger — we are able to show that sufficiency holds for $\mathcal{A}$ that are compositions of linear and surjective functions.

Theorem B.5. *Let $\epsilon_{\mathbf{y}*}$ be as defined in Theorem 3.2. Suppose the twice-differentiable operator $\mathcal{A} := \mathbf{P}^T \circ \phi$ is composed of $\mathbf{P} : \mathbb{R}^d \to \mathbb{R}^r$, a linear projection, and $\phi : \mathbb{R}^n \to \mathbb{R}^r$, an arbitrary surjective function. We have that*

$$p(\mathbf{y}|\epsilon_{\mathbf{y}*}) = p(\mathbf{y}|\mathbf{x}_0). \tag{49}$$

To prove Theorems 3.2 and B.5, we establish the following Lemma which characterizes useful information about $\mathbf{x}_0^*$.

Lemma B.6. *Suppose $\mathbf{y} \in \mathbb{R}^k$ is fixed, $\mathbf{x}_t \in \mathbb{R}^n$, with twice differentiable linear operator $\mathcal{A} : \mathbb{R}^n \to \mathbb{R}^k$. Then, for $\epsilon_{\mathbf{y}} = \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t|\mathbf{x}_0)$ which maximizes $p(\mathbf{y}|\mathbf{x}_0)$, the following holds true:*

1. *if $\eta = 0$ (i.e. the noiseless regime), $\mathcal{A}(\mathbf{x}_0) = \mathcal{A}(\mathbf{x}_t + \sigma_t^2 {\epsilon_{\mathbf{y}}}^*)$*
2. *if $\mathcal{A}$ is surjective, $\mathcal{A}(\mathbf{x}_0) = \mathcal{A}(\mathbf{x}_t + \sigma_t^2 {\epsilon_{\mathbf{y}}}^*)$*

3. *if $\mathcal{A}$ is linear,* $\langle \mathbf{y} - \mathcal{A}(\mathbf{x}_t + \sigma_t^2 \epsilon_{\mathbf{y}}{}^*), \mathcal{A}(\mathbf{x}_t + \sigma_t^2 \epsilon_{\mathbf{y}}{}^*) - \mathcal{A}(\mathbf{x}_0) \rangle = 0$.

An interpretation of statement 3 reads that the optimal solution $\epsilon_{\mathbf{y}}{}^*$ for estimating $\mathbf{x}_0$ is orthogonal to the error to $\mathbf{y}$ in the linear case. The requirements of statement 3 may be relaxed to the statement $\mathcal{A}(\mathbf{x}) - \mathcal{A}(\mathbf{z})$ is in the range of the Jacobian of $\mathcal{A}$ at $\mathbf{z}$, however this is less intuitive than linearity. We avoid invoking linearity of $\mathcal{A}$ as long as possible to illustrate the fact that other transformations may share this property as well.

Proof (of Lemma B.6). We will make use of the bijective mapping $\mathbf{z} \mapsto \mathbf{x}_t + \sigma_t^2 \epsilon_{\mathbf{y}}$, and charactarize the minima which maximize $\log p(\mathbf{y}|\mathbf{x}_0)$. We can solve the optimization problem,

$$\arg\min_{\mathbf{z}} \ ||\mathbf{y} - \mathcal{A}(\mathbf{z})||_2^2$$

A minima to this objective can be characterized by the first order necessary condition,

$$\begin{aligned} \nabla_{\mathbf{z}} ||\mathbf{y} - \mathcal{A}(\mathbf{z})||_2^2 &= -2\mathbf{J}_{\mathbf{z}}[\mathcal{A}](\mathbf{z})^T (\mathbf{y} - \mathcal{A}(\mathbf{z})) \\ &= -2\mathbf{J}_{\mathbf{z}}[\mathcal{A}](\mathbf{z})^T (\mathcal{A}(\mathbf{x}_0) - \eta - \mathcal{A}(\mathbf{z})) := 0. \end{aligned}$$

We can confirm it is a minima by checking the solution of the above with,

$$\begin{aligned} \mathbf{H}_{\mathbf{z}} \left[||\mathbf{y} - \mathcal{A}(\mathbf{z})||_2^2 \right] (\mathbf{z}^*) &= 2\nabla_{\mathbf{z}} \left[\mathbf{J}_{\mathbf{z}}[\mathcal{A}](\mathbf{z})^T (\mathcal{A}(\mathbf{x}_0) + \eta) \right] (\mathbf{z}^*) \\ &= 2\mathbf{J}_{\mathbf{z}}[\mathcal{A}](\mathbf{z}^*)^T \mathbf{J}_{\mathbf{z}}[\mathcal{A}](\mathbf{z}^*) + \sum_{j=1}^{k} \mathbf{H}_{\mathbf{z}}[\mathcal{A}_{(j)}](\mathbf{z}^*) \left(\mathbf{y} - \mathcal{A}(\mathbf{z}^*)\right) \\ &\succcurlyeq 0. \end{aligned}$$

If $\eta = 0$, we have that $\mathcal{A}(\mathbf{x}_0) = \mathbf{y}$, and therefore choosing any $\mathcal{A}(\mathbf{z}^*) = \mathcal{A}(\mathbf{x}_0)$ satisfies the first order condition. The second order condition is furthermore satisfied, as $\mathbf{y} - \mathcal{A}(\mathbf{z}^*) = 0$, meaning,

$$\mathbf{H}_{\mathbf{z}} \left[||\mathbf{y} - \mathcal{A}(\mathbf{z})||_2^2 \right] (\mathbf{z}^*) = 2\mathbf{J}_{\mathbf{z}}[\mathcal{A}](\mathbf{z}^*)^T \mathbf{J}_{\mathbf{z}}[\mathcal{A}](\mathbf{z}^*) \succcurlyeq 0.$$

This satisfies statement 1. Statement 2 is satisfied similarly, by choosing the same $\mathbf{z}$. Note that this case differs, in that $\mathbf{z} = \mathbf{x}_0$ is no longer necessarily a valid solution.

Statement 3, is already satisfied in the cases where $\mathcal{A}$ has rank equal to the dimension of its co-domain (if $d = n$, this is equivalent to being full rank), since $\mathbf{y} - \mathcal{A}(\mathbf{z}^*) = 0$. Therefore we assume $\mathcal{A}$ is low-rank to prove the remaining cases.

To show orthogonality between $\mathbf{y} - \mathcal{A}(\mathbf{z}^*)$ and $\mathcal{A}(\mathbf{z}^*) - \mathcal{A}(\mathbf{x}_0)$ in other cases, we let $\eta = \delta + \delta_\perp$. We can choose an optimal value for $\delta_\perp$ that satisfies $\delta_\perp^* = \inf_{\delta_\perp} \left\{ ||\mathbf{y} - \mathcal{A}(\mathbf{z}^*) - \delta_\perp||_2^2 \right\}$, for the optimal value, $\mathbf{z}^*$. Due to the non-negativity and 0 preserving properties of norms, we have,

$$\begin{aligned} \delta_\perp^* &= \mathbf{y} - \mathcal{A}(\mathbf{z}^*) \\ &= \mathcal{A}(\mathbf{x}_0) + \eta - \mathcal{A}(\mathbf{z}^*) \\ &= \mathcal{A}(\mathbf{x}_0) + \delta_\perp^* + \delta^* - \mathcal{A}(\mathbf{z}^*) \\ \implies \delta^* &= \mathcal{A}(\mathbf{z}^*) - \mathcal{A}(\mathbf{x}_0). \end{aligned}$$

At the optima of the original objective, $\mathbf{z}^*$, the first order necessary condition dictates that,

$$\mathbf{J}_{\mathbf{z}}[\mathcal{A}](\mathbf{z}^*)^T(y - \mathcal{A}(\mathbf{z}^*)) = \mathbf{J}_{\mathbf{z}}[\mathcal{A}](\mathbf{z}^*)^T\delta_\perp^* := 0.$$

For a linear $\mathcal{A}$, the Jacobian is constant, so let $\mathbf{J}_{\mathbf{z}}[\mathcal{A}] = \mathbf{J}$. Therefore, $\mathbf{J}^T\delta_\perp^* = 0$, meaning $\delta_\perp^* \in \mathcal{N}(\mathbf{J}^T)$.

Simultaneously, since $\delta^* = \mathcal{A}(\mathbf{x}) - \mathcal{A}(\mathbf{z}^*) = \mathcal{A}(\mathbf{x} - \mathbf{z}^*) = \mathbf{J}(\mathbf{x} - \mathbf{z}^*)$, we have $\delta^* \in \mathcal{R}(\mathbf{J}^T)$. Therefore due to the orthogonality of range and null spaces of a matrix, $\langle \delta_\perp^*, \delta^* \rangle = 0$, completing the proof.

□

We are now able to prove the theorems in the main text.

Proof of Theorem 3.2. We leverage the theory of sufficient statistics to demonstrate our result. Namely, if $\epsilon_{\mathbf{y}*}$ is a sufficient statistic for $\mathbf{y}$, then,

$$p(\mathbf{y}|\epsilon_{\mathbf{y}*}) = p(\mathbf{y}|\epsilon_{\mathbf{y}*}, \mathbf{x}_0) = p(\mathbf{y}|\mathbf{x}_0). \tag{50}$$

Therefore it suffices to demonstrate that $\epsilon_{\mathbf{y}*}$ is a sufficient statistic for $\mathbf{y}$.

By the Neyman-Fisher Factorization theorem, we have that a necessary and sufficient condition is if there exists non-negative functions g_θ and h such that

$$p(\mathbf{y}|\mathbf{x}_0) = g(\epsilon_{\mathbf{y}*}, \mathbf{x}_0)h(\mathbf{y}). \tag{51}$$

We observe that since $\eta \sim \mathcal{N}(\mathbf{0}, \sigma_{\mathbf{y}}^2\mathbf{I})$, our random variable $\mathbf{y}$ can be characterized by the density function

$$p(\mathbf{y}|\mathbf{x}_0) = \mathcal{N}(\mathbf{y}; \mu = \mathcal{A}(\mathbf{x}_0), \Sigma = \sigma_{\mathbf{y}}^2\mathbf{I}). \tag{52}$$

Therefore, letting $\mathbf{y}_{\epsilon_{\mathbf{y}*}} = \mathcal{A}(\frac{1}{\sqrt{\alpha_t}}(\mathbf{x}_t + \sigma_t^2\epsilon_{\mathbf{y}*}))$, we can write

$$p(\mathbf{y}|\mathbf{x}_0) = (2\pi\sigma_{\mathbf{y}}^2)^{-n/2}\exp\left(-\frac{1}{2\sigma_{\mathbf{y}}^2}||\mathbf{y} - \mathcal{A}(\mathbf{x}_0)||_2^2\right) \tag{53}$$

$$= (2\pi\sigma_{\mathbf{y}}^2)^{-n/2}\exp\left(-\frac{1}{2\sigma_{\mathbf{y}}^2}\left(||\mathbf{y} - \mathbf{y}_{\epsilon_{\mathbf{y}*}}||_2^2 + ||\mathbf{y}_{\epsilon_{\mathbf{y}*}} - \mathcal{A}(\mathbf{x}_0)||_2^2 + 2\langle \mathbf{y} - \mathbf{y}_{\epsilon_{\mathbf{y}*}}, \mathbf{y}_{\epsilon_{\mathbf{y}*}} - \mathcal{A}(\mathbf{x}_0)\rangle\right)\right) \tag{54}$$

$$= (2\pi\sigma_{\mathbf{y}}^2)^{-n/2}\exp\left(-\frac{1}{2\sigma_{\mathbf{y}}^2}||\mathbf{y}_{\epsilon_{\mathbf{y}*}} - \mathcal{A}(\mathbf{x}_0)||_2^2\right)\exp\left(-\frac{1}{2\sigma_{\mathbf{y}}^2}||\mathbf{y} - \mathbf{y}_{\epsilon_{\mathbf{y}*}}||_2^2\right), \tag{55}$$

where the third equality is due to Lemma B.6. In the case that $\mathcal{A}$ is surjective, or the noiseless regime, statements 2 and 1 respectively satisfy the equality above trivially, as $\mathbf{y} = \mathbf{y}_{\epsilon_{\mathbf{y}*}}$. If the operator is otherwise linear, statement 3 shows the cross term vanishes.

Therefore, we can assign

$$g(\epsilon_{\mathbf{y}*}, \mathbf{x}_0) = (2\pi\sigma_{\mathbf{y}}^2)^{-n/2}\exp\left(\frac{1}{2\sigma_{\mathbf{y}}^2}||\mathbf{y}_{\epsilon_{\mathbf{y}*}} - \mathcal{A}(\mathbf{x}_0)||_2^2\right) \tag{56}$$

$$h(\mathbf{y}) = \exp\left(\frac{1}{2\sigma_{\mathbf{y}}^2}||\mathbf{y} - \mathbf{y}_{\epsilon_{\mathbf{y}*}}||_2^2\right). \tag{57}$$

In the case where the measurement process $\mathcal{A}(\mathbf{x}) = \mathbf{y}$ is noiseless, this implies $h(\mathbf{y}) = 1$. □

We now modify the argument in order to relax the linearity assumption.

Proof of Theorem B.5. Let $\mathbf{z} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t + \sigma_t^2\epsilon_{\mathbf{y}}\right)$, and $\mathbf{z}^* = \arg\min_{\mathbf{z}}\{||\mathbf{y} - \mathcal{A}(\mathbf{z})||\}$.

Since $\mathbf{z}^*$ minimizes the objective $||\mathbf{y} - \mathcal{A}(\mathbf{z})||$, we also have that,

$$\phi(\mathbf{z}^*) := \arg\min_{\alpha} \left\{ ||\mathbf{y} - \mathbf{P}^T(\alpha)|| \right\} = \arg\max_{\alpha} \; p(\mathbf{y}|\alpha).$$

We can invoke Lemma B.6 to say

$$||\mathbf{y} - \mathbf{P}^T\phi(\mathbf{x}_0)||_2^2 = ||\mathbf{y} - \mathbf{P}^T\phi(\mathbf{z}^*)||_2^2 + ||\mathbf{P}^T\phi(\mathbf{z}^*) - \mathbf{P}^T\phi(\mathbf{x}_0)||_2^2,$$

since $\mathbf{P}^T$ is a linear operator, and $\phi(\mathbf{z}^*)$ satisfies the conditions in the lemma. Therefore, we have,

$$\begin{aligned} p(\mathbf{y}|\mathbf{x}_0) &= (2\pi\sigma_{\mathbf{y}}^2)^{-n/2} \exp\left(-\frac{1}{2\sigma_{\mathbf{y}}^2}||\mathbf{y} - \mathbf{P}^T\phi(\mathbf{x}_0)||_2^2\right) \\ &= (2\pi\sigma_{\mathbf{y}}^2)^{-n/2} \exp\left(-\frac{1}{2\sigma_{\mathbf{y}}^2}||\mathbf{y} - \mathbf{P}^T\phi(\mathbf{z}^*)||_2^2\right) \exp\left(-\frac{1}{2\sigma_{\mathbf{y}}^2}||\mathbf{P}^T\phi(\mathbf{z}^*) - \mathbf{P}^T\phi(\mathbf{x}_0)||_2^2\right). \end{aligned}$$

We assign terms,

$$g(\mathbf{z}_*, \mathbf{x}_0) = (2\pi\sigma_{\mathbf{y}}^2)^{-n/2} \exp\left(-\frac{1}{2\sigma_{\mathbf{y}}^2}||\mathbf{P}^T\phi(\mathbf{z}^*) - \mathbf{P}^T\phi(\mathbf{x}_0)||_2^2\right) \tag{58}$$

$$h(\mathbf{y}) = \exp\left(-\frac{1}{2\sigma_{\mathbf{y}}^2}||\mathbf{y} - \mathbf{P}^T\phi(\mathbf{z}^*)||_2^2\right), \tag{59}$$

$$\tag{60}$$

and once again invoke the Neyman-Fisher Factorization theorem to show $\mathbf{z}^*$ is sufficient for $\mathbf{y}$. Since $\epsilon_{\mathbf{y}_*}$ is a bijective mapping from $\mathbf{z}^*$, we have that $\epsilon_{\mathbf{y}_*}$ is sufficient, and similarly to Theorem 3.2 we state, $p(\mathbf{y}|\epsilon_{\mathbf{y}_*}) = p(\mathbf{y}|\epsilon_{\mathbf{y}_*}, \mathbf{x}_0) = p(\mathbf{y}|\mathbf{x}_0)$. □

Finally, we note that this proof provides necessary conditions, but not sufficient conditions for the sufficiency of **DCS**'s estimator. In this work, we do not investigate operators outside of the scope of Theorem B.5, there are potentially even weaker conditions on $\mathcal{A}$ that exist.

C. Invertibility of $\mathcal{A}$

Often, $\mathcal{A}$ is simply non-invertible (e.g. for super-resolution, inpainting, phase retrieval, and sparse MRI reconstruction tasks). With other tasks such as signal deblurring, the invertibility of $\mathcal{A}$ is often mathematically possible, but not numerically stable. In theory, blurring operator can be represented as convolution operators on the signal $\mathbf{x}$. Theorefore, the convolution theorem tells us that inverting a blurring operator $\mathcal{G}(*)$ on $\mathbf{x}$ is as simple as taking the quotient of the convolved signal $\mathbf{y} = \mathcal{A}(\mathbf{x})$ against the convolution kernel in the frequency domain, i.e.,

$$\mathbf{x} = \mathcal{F}^{-1}[\mathcal{F}(\mathbf{y})/\mathcal{F}(\mathcal{G})] = \mathbf{y} * \mathcal{F}^{-1}[\mathcal{F}(\mathcal{G})^{-1}] \tag{61}$$

where $\mathcal{F}$ denotes the Fourier operator. However, in practice, there are implicit assumptions in 61, such as the computability of $\mathcal{F}(\mathcal{G})$ and the existence of $\mathcal{F}(\mathcal{G})^{-1}$, that may not always hold. In particular, blur kernels are often truncated in practice, resulting in highly ill-conditioned (or compactly supported) $\mathcal{F}(\mathcal{G})$ in the frequency domain, and numerical unstable (or non-existent) inverses. Ultimately, directly inverting $\mathcal{A}$ often fails to produce the highest quality results, even though it is possible.

Table 3: Description of latent and Jacobian-free solvers used for comparisons in text. For each solver we list the type (as described in Section A.2), optimization space (pixel or latent), whether it requires backpropagation through a neural function evaluation (NFE, i.e., the score network call), as well as runtime and memory footprint.

Solver	Type	Space	No NFE Backprop	Runtime	Memory
DCS (Ours)	Hybrid	Pixel	✓	**1x**	**1x**
Latent-DPS (Chung et al., 2023) [3]	Posterior	Latent	✗	6.1x	8.9x
PSLD (Rout et al., 2023)	Posterior	Latent	✗	7.5x	15x
STSL (Rout et al., 2024)	Posterior	Latent	✗	1.85x	9x
ReSample (Song et al., 2024)	Projection	Latent	✓[4]	29.5x	8.95x
DPS-JF (Chung et al., 2023)	Posterior	Pixel	✓	1.5x	1.1x
LGD-MC (n=10) (Song et al., 2023b)	Posterior	Pixel	✗	6x	3.2x
LGD-MC-JF (n=10) (Song et al., 2023b)	Posterior	Pixel	✗	2x	1.1x

D. Additional Experiments

In this section, we provide further comparisons against latent and Jacobian-free methods (Table 3).

D.1. Comparison against Latent Models (Table 4)

We show that our pixel-based model also performs favorably against latent models in Table 4. We retain the same experimental setting on pixel-based models as in Table 2. For FFHQ, we use the pretrained FFHQ model weights from (Chung et al., 2022a) for our method, and the pretrained FFHQ model with a VQ-F4 first stage model (Rombach et al., 2022) in latent space models. For ImageNet, we again use pretrained model weights from (Chung et al., 2022a) in pixel-based diffusion solvers, and the Stable Diffusion v1.5 latent model for latent solvers. As with pixel-based methods many existing works suffer in the presence of additional noise. Further implementation details are discussed in Appendix E.

D.2. Comparison against other Jacobian-Free Methods (Table 5)

A major advantage of **DCS** is the fact that it is Jacobian-free (Section 3.4) — this results in at least $6\times$ reduction in memory cost during inference compared with Jacobian-based methods, which can be a major enabling factor for the adoption of such algorithms on consumer GPUs and edge devices. However, naively removing the backpropagation through the score network can reduce the quality of the measurement consistency correction step in inverse solvers. In this experiment, we demonstrate that our treatment via the maximum likelihood framework and the **n**oise-**a**ware **m**aximization results in significantly higher quality samples, compared to a naive implementation in DPS-JF and LGD-JF, which are both Jacobian-free variants of the original algorithms (Chung et al., 2022a) and (Song et al., 2023a). Namely, we approximate the Jacobian with respect to the input to the denoising network (left hand side) by the Jacobian with respect to the predicted $\mathbf{x}_0$ (right hand side)

$$\frac{\partial}{\partial \mathbf{x}_t}||\mathbf{y} - \mathcal{A}(\hat{\mathbf{x}})||_2^2 \approx \frac{\partial}{\partial \hat{\mathbf{x}}}||\mathbf{y} - \mathcal{A}(\hat{\mathbf{x}})||_2^2, \tag{62}$$

where $\hat{\mathbf{x}} = \mathbf{f}(\mathbf{x}_t, \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t))$ and $\mathbf{f}$ is an algorithm-dependent function of $\mathbf{x}_t$ and its score. (Note that the right hand side no longer involves backpropagation through $\mathbf{f}$ and therefore $\boldsymbol{\epsilon}_\theta$).

[3] Latent-DPS is a direct application of DPS (Chung et al., 2023) to latent diffusion models. It is also mentioned in (Rout et al., 2023).

[4] As described in (Song et al., 2024), ReSample does not run backpropagation on the score network, however the implementation does (Appendix E.4).

Table 4: Quantitative comparison against latent models on FFHQ 256x256-1K and ImageNet-1K datasets across various inverse problem tasks and noise levels ($\sigma_{\mathbf{y}} \in \{0.01, 0.1\}$).

FFHQ	SR ×4			Random Inpainting			Box Inpainting			Gaussian Deblurring			Motion Deblurring		
$\sigma_{\mathbf{y}} = 0.01$	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓
Ours	**0.137**	**30.138**	**19.45**	**0.024**	**34.839**	**21.19**	**0.088**	**25.112**	**19.25**	**0.103**	**28.688**	**22.62**	**0.087**	**29.480**	**26.67**
Latent-DPS	0.324	20.086	100.27	0.249	22.64	297.43	0.227	22.184	211.23	0.390	25.608	321.5	0.950	-6.753	354.95
PSLD	0.311	20.547	42.26	0.250	22.84	214.08	0.221	22.23	204.87	0.200	23.77	318.20	0.213	23.277	359.40
STSL	0.614	16.063	327.38	0.476	17.859	190.64	0.436	11.843	190.64	0.583	15.196	364.07	0.604	10.095	388.68
ReSample	0.221	24.699	48.87	0.467	22.488	96.89	0.247	20.852	50.3	0.191	<u>27.151</u>	46.5	0.281	<u>25.138</u>	65.06
FFHQ	SR ×4			Random Inpainting			Box Inpainting			Gaussian Deblurring			Motion Deblurring		
$\sigma_{\mathbf{y}} = 0.1$	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓
Ours	**0.1748**	**24.879**	**30.107**	**0.1490**	**27.536**	**32.800**	**0.1631**	**23.217**	**26.444**	**0.1763**	**25.955**	**26.083**	**0.2238**	**24.612**	**31.400**
Latent-DPS	0.3444	19.971	45.052	0.4455	18.117	109.83	0.6410	11.365	326.75	0.6398	13.762	330.93	0.6360	12.524	334.43
PSLD	0.3481	19.251	47.864	0.3105	20.588	41.737	0.3121	19.874	40.428	0.2897	21.068	36.600	0.3307	19.224	40.374
STSL	0.3161	20.279	40.163	0.3722	19.247	54.648	0.5481	13.864	183.00	0.5137	16.411	169.32	0.5188	15.463	163.65
ReSample	0.2613	24.184	50.224	0.5267	21.575	103.62	0.2789	20.581	53.263	0.2984	23.980	56.489	0.6456	19.912	110.42
ImageNet	SR ×4			Random Inpainting			Box Inpainting			Gaussian Deblurring			Motion Deblurring		
$\sigma_{\mathbf{y}} = 0.01$	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓
Ours	**0.238**	**23.452**	**39.41**	**0.142**	**26.063**	**34.46**	**0.230**	**20.625**	**37.11**	**0.253**	**24.218**	**38.96**	**0.203**	**24.619**	**38.63**
Latent-DPS	0.642	17.973	144.82	0.603	19.881	144.81	0.751	11.964	138.33	0.805	10.532	139.62	0.821	10.697	150.49
PSLD	0.380	22.690	168.08	0.306	24.167	125.25	0.330	18.290	156.30	<u>0.397</u>	23.076	134.18	<u>0.453</u>	<u>21.576</u>	187.21
STSL	0.617	19.682	143.62	0.599	20.500	137.09	0.832	9.560	170.93	0.869	8.708	183.38	0.882	8.527	195.74
ReSample	0.552	20.260	133.42	0.820	17.775	229.82	0.504	16.795	138.97	0.513	21.578	116.04	0.573	20.430	145.67
ImageNet	SR ×4			Random Inpainting			Box Inpainting			Gaussian Deblurring			Motion Deblurring		
$\sigma_{\mathbf{y}} = 0.1$	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓
Ours	**0.4015**	**22.988**	**48.211**	**0.1655**	**26.043**	**34.469**	**0.2428**	**19.697**	**46.026**	**0.4068**	**22.283**	**51.131**	**0.4348**	**20.428**	**61.48**
Latent-DPS	0.7257	15.676	147.65	0.7973	9.4153	146.69	0.7980	9.3345	146.51	0.7988	9.3032	193.84	0.8525	9.1369	170.08
PSLD	0.4731	20.875	130.99	0.6068	19.668	145.51	0.7028	13.909	146.74	0.7372	14.181	139.90	0.7504	13.767	149.75
ReSample	0.6514	18.997	155.26	0.9654	13.612	281.82	0.5980	15.843	168.06	0.6814	19.233	173.72	1.0461	15.249	223.52

Table 5: Quantitative comparison against other Jacobian-free methods on FFHQ 256x256-1K and ImageNet-1K datasets across various inverse problem tasks and noise levels ($\sigma_{\mathbf{y}} \in \{0.01, 0.1\}$).

FFHQ	SR ×4			Random Inpainting			Box Inpainting			Gaussian Deblurring			Motion Deblurring		
$\sigma_{\mathbf{y}} = 0.01$	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓
Ours	**0.137**	**30.138**	**19.45**	**0.024**	**34.839**	**21.19**	**0.088**	**25.112**	**19.25**	**0.103**	**28.688**	**22.62**	**0.087**	**29.480**	**26.67**
DPS-JF	0.488	14.193	44.98	0.335	19.566	58.45	0.178	20.118	28.10	0.211	23.063	34.42	0.289	19.927	40.94
DPS-JF ($T = 100$)	0.589	9.473	41.24	0.578	10.072	42.06	0.571	10.618	43.08	0.563	10.859	43.77	0.566	10.922	41.26
LGD-MC-JF	0.566	10.502	41.25	0.537	12.154	43.85	0.497	13.811	46.40	0.452	15.569	46.22	0.457	15.466	46.08
LGD-MC-JF ($T = 100$)	0.593	9.346	40.60	0.587	9.688	40.99	0.581	10.126	42.30	0.574	10.273	40.59	0.574	10.364	40.51
DDNM	0.208	<u>26.277</u>	51.33	<u>0.040</u>	<u>33.076</u>	23.35	0.209	18.118	88.32	0.235	26.086	71.47	0.424	14.221	250.92
DDRM	0.502	13.002	222.45	0.393	15.935	163.91	0.472	12.148	209.18	-	-	-	-	-	-
FFHQ	SR ×4			Random Inpainting			Box Inpainting			Gaussian Deblurring			Motion Deblurring		
$\sigma_{\mathbf{y}} = 0.1$	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓
Ours	**0.1748**	**24.879**	**30.107**	**0.1490**	**27.536**	**32.800**	<u>0.1631</u>	**23.217**	**26.444**	**0.1763**	**25.955**	**26.083**	<u>0.2238</u>	**24.612**	**31.400**
DPS-JF	0.494	14.111	46.59	0.371	18.310	56.49	0.226	19.451	34.02	0.246	21.808	35.53	0.342	18.339	40.70
DPS-JF ($T = 100$)	0.589	9.432	40.82	0.582	9.900	39.58	0.572	10.552	42.90	0.564	10.894	42.36	0.568	10.943	42.44
LGD-MC-JF	0.557	11.208	44.86	0.511	13.265	49.07	0.452	15.243	48.68	0.396	17.434	46.76	0.400	17.301	45.53
LGD-MC-JF ($T = 100$)	0.594	9.324	41.06	0.589	9.655	41.65	0.580	10.107	42.97	0.578	10.334	41.84	0.574	10.312	41.53
DDNM	0.6230	21.493	145.889	0.179	24.964	39.183	0.334	19.195	72.105	1.220	10.727	176.756	0.739	5.099	524.021
DDRM	0.7853	6.3273	271.70	0.6018	10.995	255.95	0.6323	9.6360	288.11	-	-	-	-	-	-

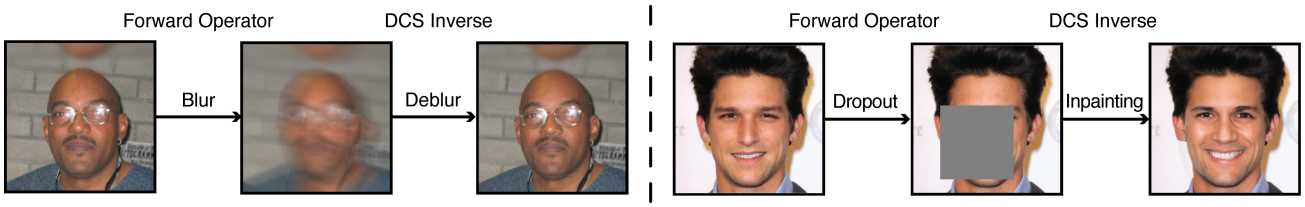


Figure 9: A demonstration of our solver, **DCS**, solving two inverse problems on natural images from the CelebA-HQ dataset. Motion blur (left), and box dropout (right) are examples of forward operators that are non-invertible. We show further results in Section 4

D.3. Further Noise Experiments

Table 6: Quantitative experiments on FFHQ 256x256-1K at $\sigma_{\mathbf{y}} = 0.5$. We compare against pixel-based solvers (upper half) and latent-based solvers (lower half).

FFHQ	SR ×4			Random Inpainting			Box Inpainting			Gaussian Deblurring			Motion Deblurring		
	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓
Ours	0.2287	20.362	100.94	0.2067	22.999	89.312	0.2005	21.298	40.099	0.2109	24.009	82.132	0.2301	22.306	90.403
DPS	0.2000	22.588	92.791	0.2290	22.808	90.739	0.2118	20.278	81.491	0.2268	25.020	83.686	0.2479	20.767	91.972
DDNM	0.7812	9.8324	387.43	0.8721	15.573	233.15	0.9966	12.607	287.79	1.4475	3.5686	408.85	1.3328	3.1782	393.24
ReSample	0.5704	19.948	179.35	0.6892	20.014	200.06	0.4958	17.530	160.47	0.5409	21.166	162.40	0.6380	19.875	194.69

D.4. Subset of FFHQ used in other works

Table 7: Quantitative evaluation of our method on FFHQ 256x256, following the experimental setup of (Song et al., 2024). We compare against pixel-based solvers (upper half) and latent-based solvers (lower half).

	SR ×4			Random Inpainting			Box Inpainting			Gaussian Deblurring			Motion Deblurring			Cost	
	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	Time ↓	Mem. ↓
Ours	**0.074**	**29.51**	**0.811**	**0.052**	**31.13**	**0.850**	**0.102**	**22.07**	0.761	**0.078**	**29.92**	**0.817**	**0.051**	**32.32**	**0.833**	**1x**	**1x**
DPS	0.132	27.10	0.729	0.084	30.91	0.833	0.107	21.62	0.755	0.090	28.26	0.767	0.108	26.816	0.726	6x	3.2x
MCG	0.112	27.07	0.784	0.877	11.02	0.02	0.905	10.883	0.001	0.176	24.89	0.768	-	-	-	6.1x	3.2x
DDNM	0.242	27.63	0.587	0.230	27.92	0.604	0.194	23.08	0.639	0.287	27.24	0.561	0.642	8.682	0.165	1.75x	1x
Latent-DPS	0.324	20.086	0.473	0.249	22.64	0.570	0.227	22.184	0.595	0.209	23.512	0.600	0.217	22.930	0.582	6.1x	8.9x
PSLD	0.311	20.547	0.491	0.250	22.84	0.579	0.221	22.23	0.607	0.200	23.77	0.614	0.213	23.277	0.596	7.5x	15x
STSL	0.242	27.63	0.587	0.230	27.92	0.604	0.194	23.08	0.639	0.287	27.24	0.561	0.641	10.17	0.245	1.85x	9x
ReSample	0.090	29.024	0.791	0.053	30.99	0.844	0.156	20.71	**0.778**	0.113	29.19	0.784	0.197	27.65	0.706	29.5x	8.95x

E. Implementation Details

We provide implementation details of our experiments, as well as those for other experiments we compare against.

E.1. Our Method

Our proposed DCS has just two primary hyperparameters, as described in the table below. First is the number of time steps T. This has relatively little effect on our model performance on most tasks. However, it is occasionally helpful to increase T, especially in box inpainting, where there is zero signal from $\mathbf{y}$ in the masked region. Here, higher T allows the diffusion model to obtain a better solution in this unconditional diffusion process. Second, we have the choice of `minimizer`, which is by default the Adam optimizer (Kingma and Ba, 2014). However, in the case of linear $\mathcal{A}$, this optimizer can be replaced by the closed form analytical solution to $\mathcal{A}(\mathbf{x}) = \mathbf{y}$.

For nearly all experiments, we use the Adam optimizer with 50 optimization steps and a learning rate of 1. The exceptions are the random inpainting and box inpainting tasks, where there is no conditioning information on the masked pixels. This requires more denoising steps, as the diffusion process is totally unconditional inside the mask, up to local correlations learned inside the score network s_θ. Here, we use the analytical solver with $\mathcal{A}^\dagger = \mathcal{A}$. Similarly, for nearly all experiments

we use $T = 50$ as found in Table 7, with the exception being random inpainting and box inpainting tasks, where we found that taking $T = 1000$ steps improved performance. However, there is little increase in runtime, since the minimization step is much faster here.

Notation	Definition
T	The number of diffusion steps used in the sampler.
`minimizer`	The minimizer used to solve for $\epsilon_{\mathbf{y}}$.

E.2. Latent Models on ImageNet

We note that previous latent models use the pretrained weights in (Rombach et al., 2022) for 256×256 resolution datasets. However, there are no published weights in the GitHub repository for unconditional ImageNet, making a fair comparison of our method against latent models more involved. To this end, we leverage a significantly more powerful Stable Diffusion v1.5 model, with publicly available weights on HuggingFace for our experiments. The measurements and the output images are appropriately scaled for a fair comparison.

E.3. STSL

At the time of writing this work, we did not find publicly available code for STSL (Rout et al., 2024). Therefore, we implement the algorithm ourselves in our codebase, and use the hyperparameters provided in the paper.

E.4. ReSample

We directly use the published code of ReSample (Song et al., 2024) with no changes in our paper. We discuss two notable aspects of the experiments with ReSample. First, the implementation on GitHub differs from that pseudocode discussed in the paper. Namely, the pseudocode in the paper describes enforcing latent- and pixel-based consistency occasionally during an otherwise unconditional sampling process.

In the code we observed that the sampling step taken is actually a DPS (Chung et al., 2022a) sampling step, which includes a posterior-based guidance step that takes an expensive gradient of the noise function. To see this, note that L255 in the `resample_sampling` function in `ddim.py` calls a function `measurement_cond_fn`, which is defined at L62 in `main.py` and passed into the resampling function. This function is a member of the class `PosteriorSampling` defined in L53 in `condition_methods.py`. Inspecting this class, we note that it calls `torch.autograd.grad` on the diffusion step as a function of `x_prev` (L33 or L39). In other words, a gradient is computed for the measurement norm with respect to the input to the diffusion model, i.e., a DPS step.

We closely investigated this DPS step in our experiments, ultimately concluding that it has a significant effect on the performance of the algorithm, and that it was a *more* fair comparison to include this step, rather than removing it. However, the inclusion of this sampling step has two primary effects. First, it results in further increases the computation time of ReSample. Second it reveals that ReSample relies significantly on a posterior-based formulation, applying additional resampling steps at each stage.

In experiments, we note that ReSample is significantly slower than other algorithms during sampling (see Table 1). For example, sampling ~ 1000 images with ImageNet takes more than two weeks on an A6000 GPU. Since we run five different experimental conditions for each dataset, this was an unacceptably long runtime for our academic resources. Therefore, we reduce the number of diffusion steps T of ReSample in our experiments, from 500 reported in (Song et al., 2024) to 50. However, we do provide a single experiment from the (Song et al., 2024) paper, where we reproduce the hyperparameters and dataset (a 100 image subset of FFHQ). We note that (Song et al., 2024) took a subset of the FFHQ dataset, where performance differed from the full 256×256-1K dataset performance (c.f. Table 2). Since the subset was not published, we selected a dataset based where ReSample obtained the same performance with its default parameters in (Song et al., 2024) (Table 7).

E.5. DDRM

We used the version of DDRM which is implemented in the DDNM codebase. While DDRM may theoretically be able to handle deblurring tasks, due to the high rank of the forward operators, the SVD cannot be explicitly defined in memory, and no existing code-base for DDRM supplies fast and memory-saving versions of these operators. Because of the relatively

poor performance of DDRM compared to DDNM, and the fact that DDRM can be considered a subtype of DDNM (see Appendix of (Wang et al., 2022)), we do not run on deblurring tasks.

F. Further Qualitative Comparisons

We provide further qualitative examples from the FFHQ 256×256-1K and ImageNet 256×256-1K datasets accompanying our quantitative evaluation in Table 2.

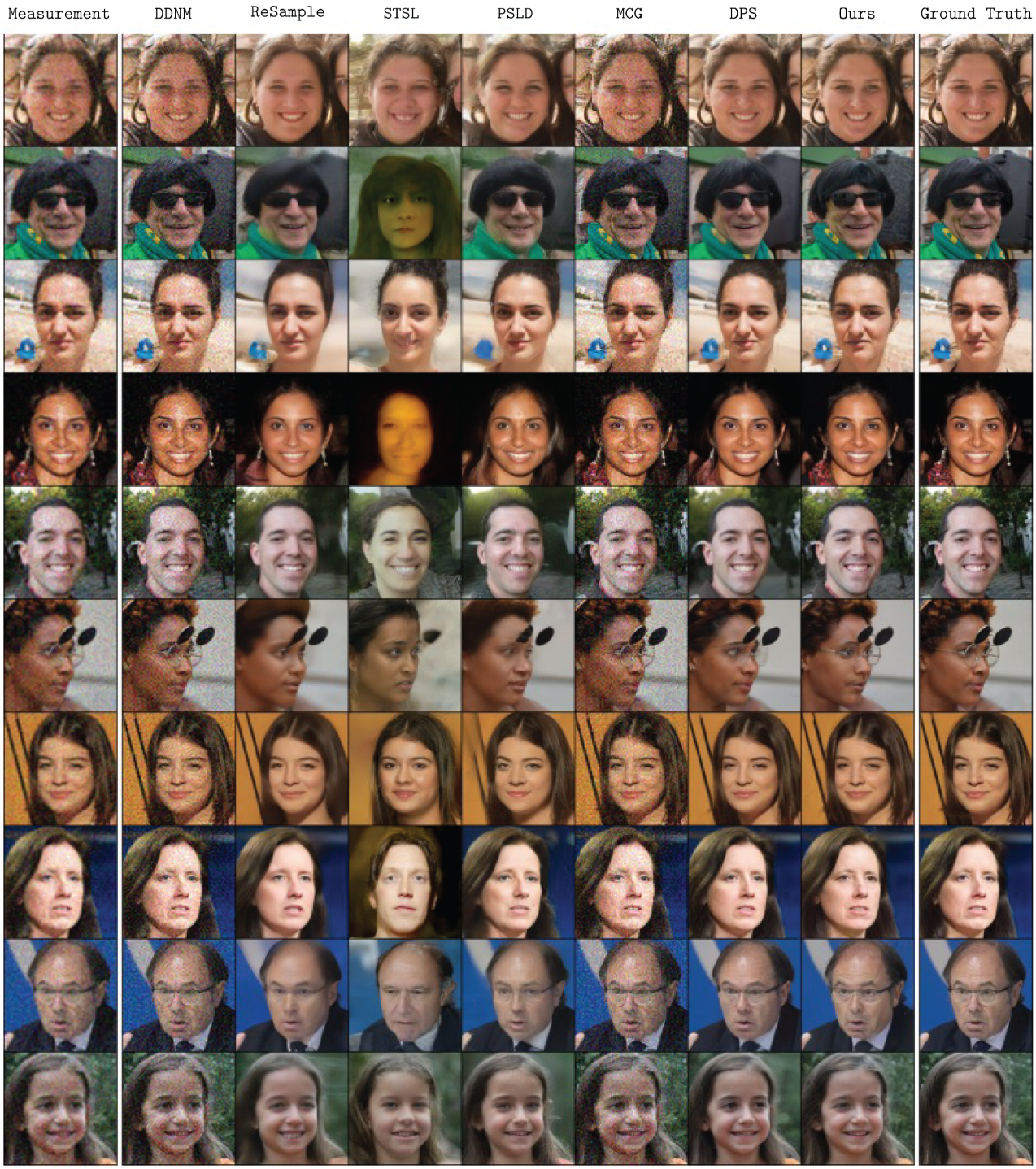


Figure 10: Comparison against competing works on FFHQ 256×256-1K dataset with the 4× super-resolution task.

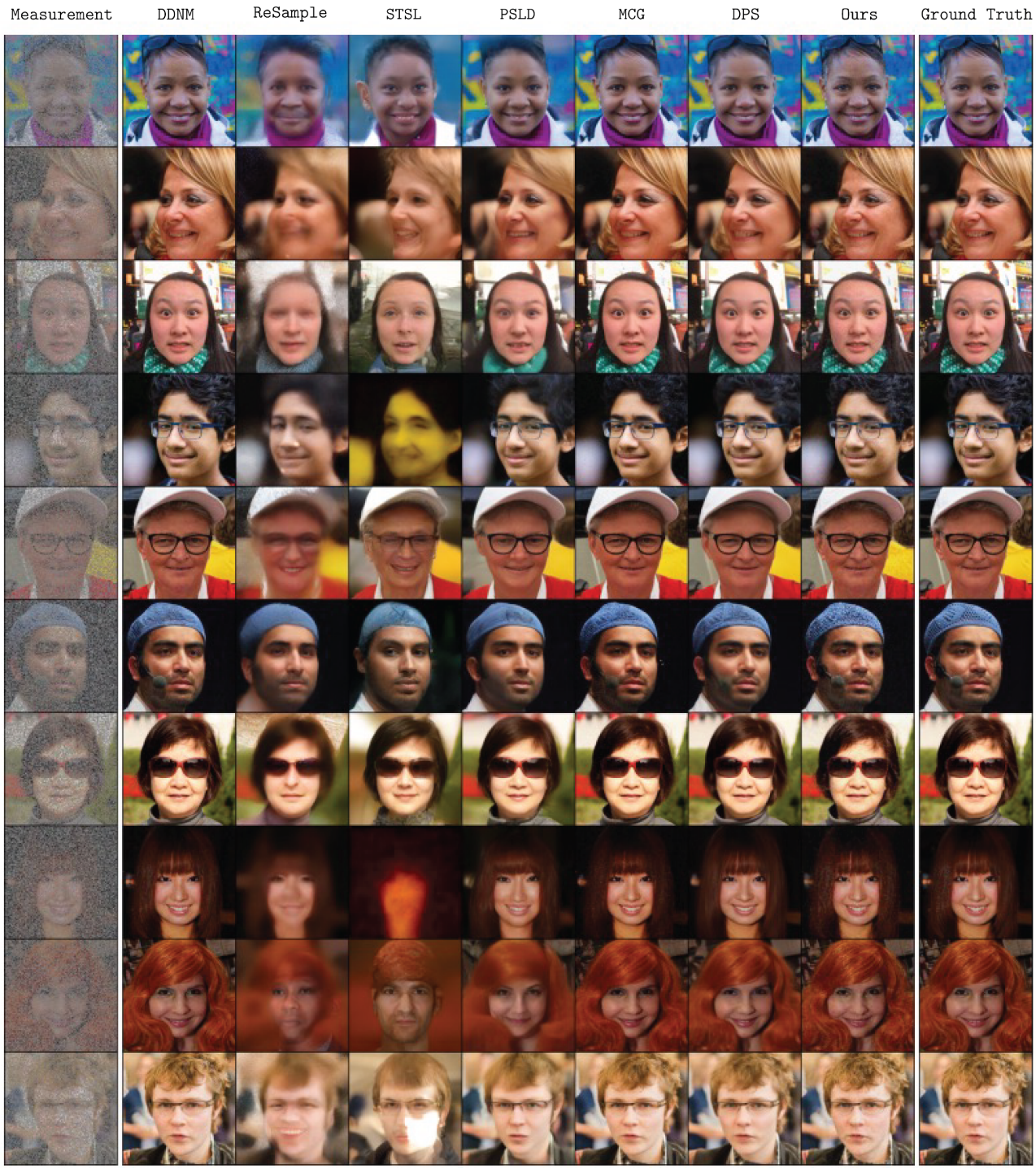


Figure 11: Comparison against competing works on FFHQ 256×256-1K dataset with the random inpainting task.

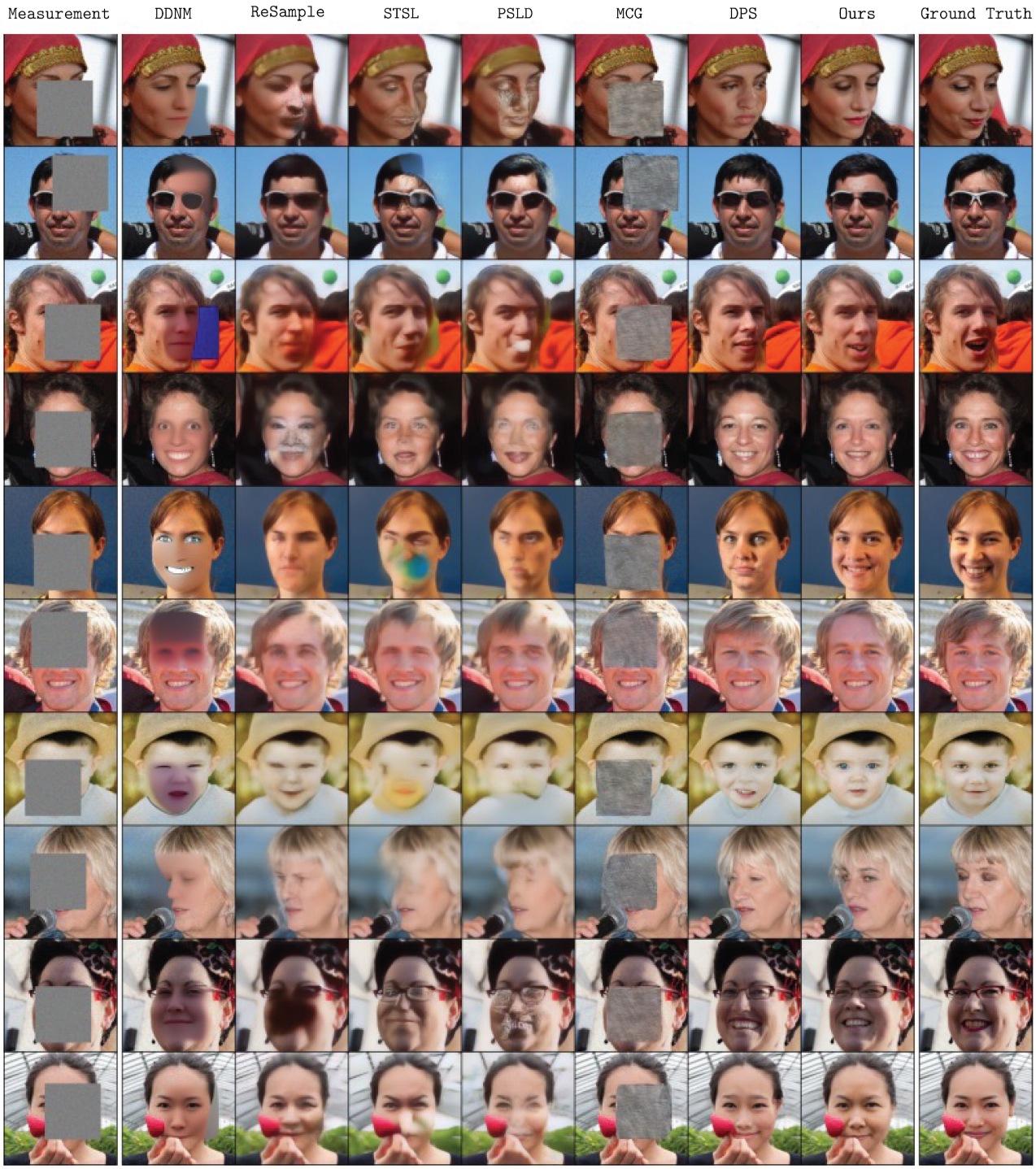


Figure 12: Comparison against competing works on FFHQ 256×256-1K dataset with the box inpainting task.

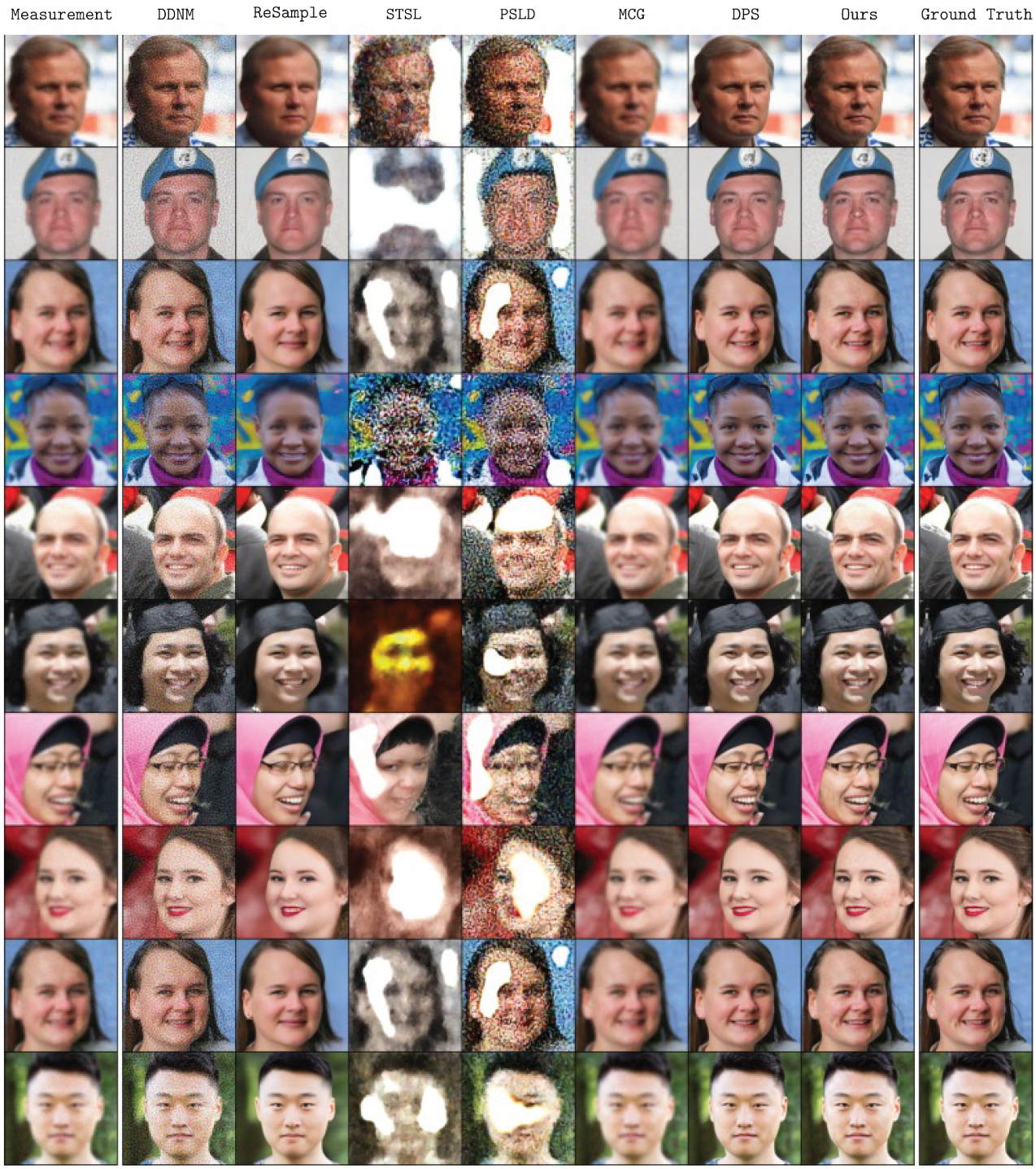


Figure 13: Comparison against competing works on FFHQ 256×256-1K dataset with the Gaussian deblurring task.

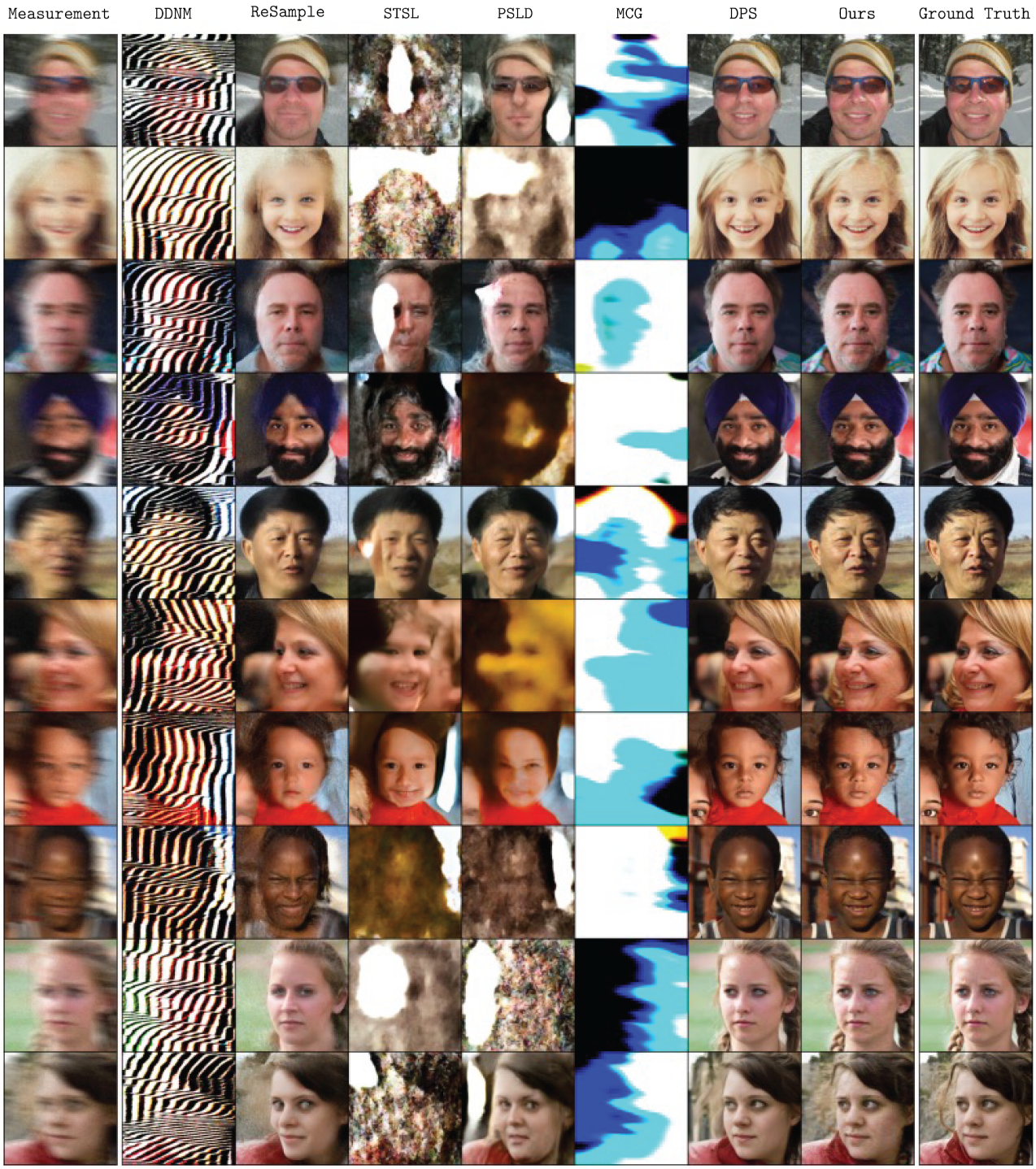


Figure 14: Comparison against competing works on FFHQ 256×256-1K dataset with the motion deblurring task.

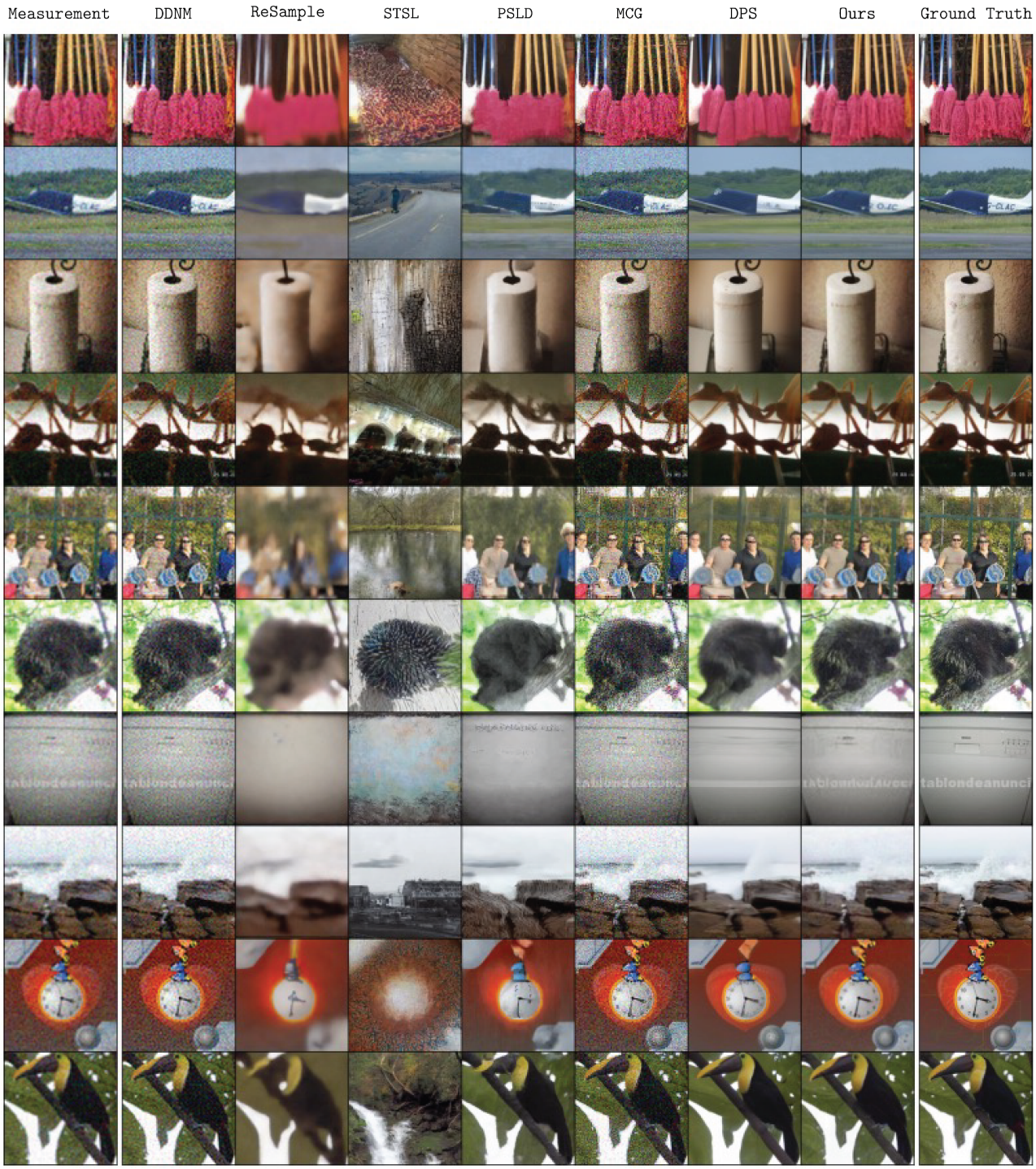


Figure 15: Comparison against competing works on FFHQ 256×256-1K dataset with the 4× super-resolution task.

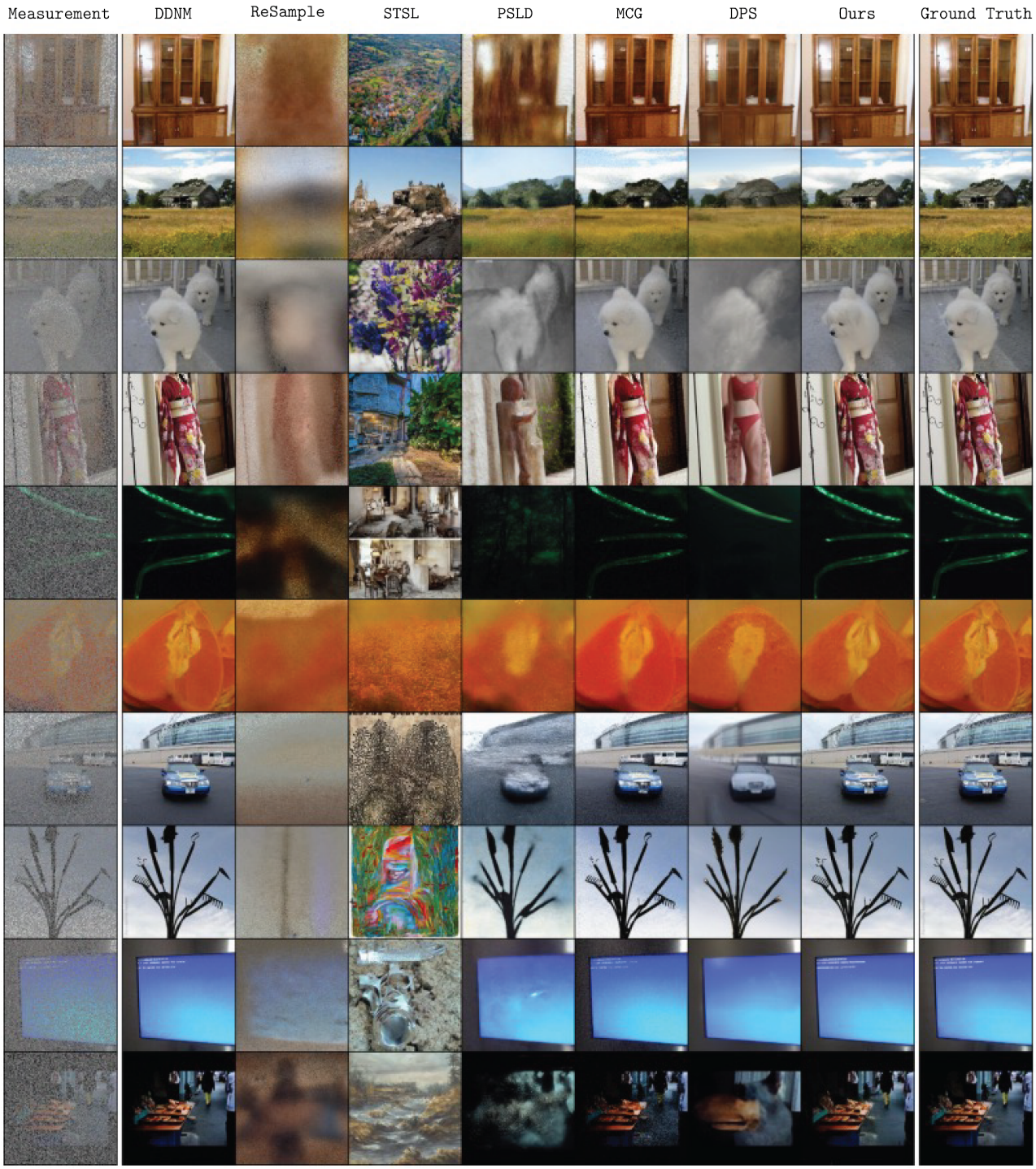


Figure 16: Comparison against competing works on ImageNet 256×256-1K dataset with the random inpainting task.

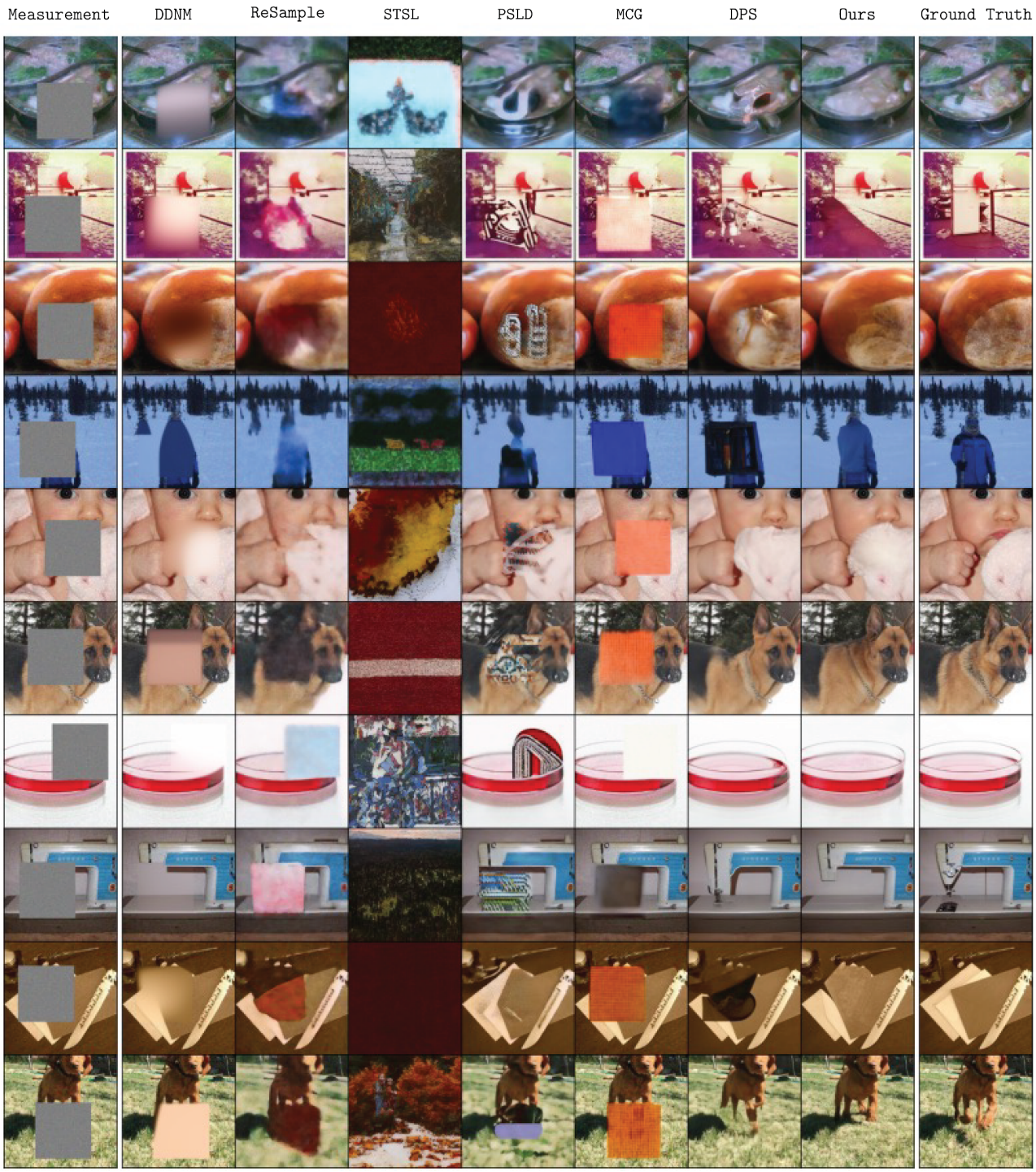


Figure 17: Comparison against competing works on FFHQ 256×256-1K dataset with the box inpainting task.

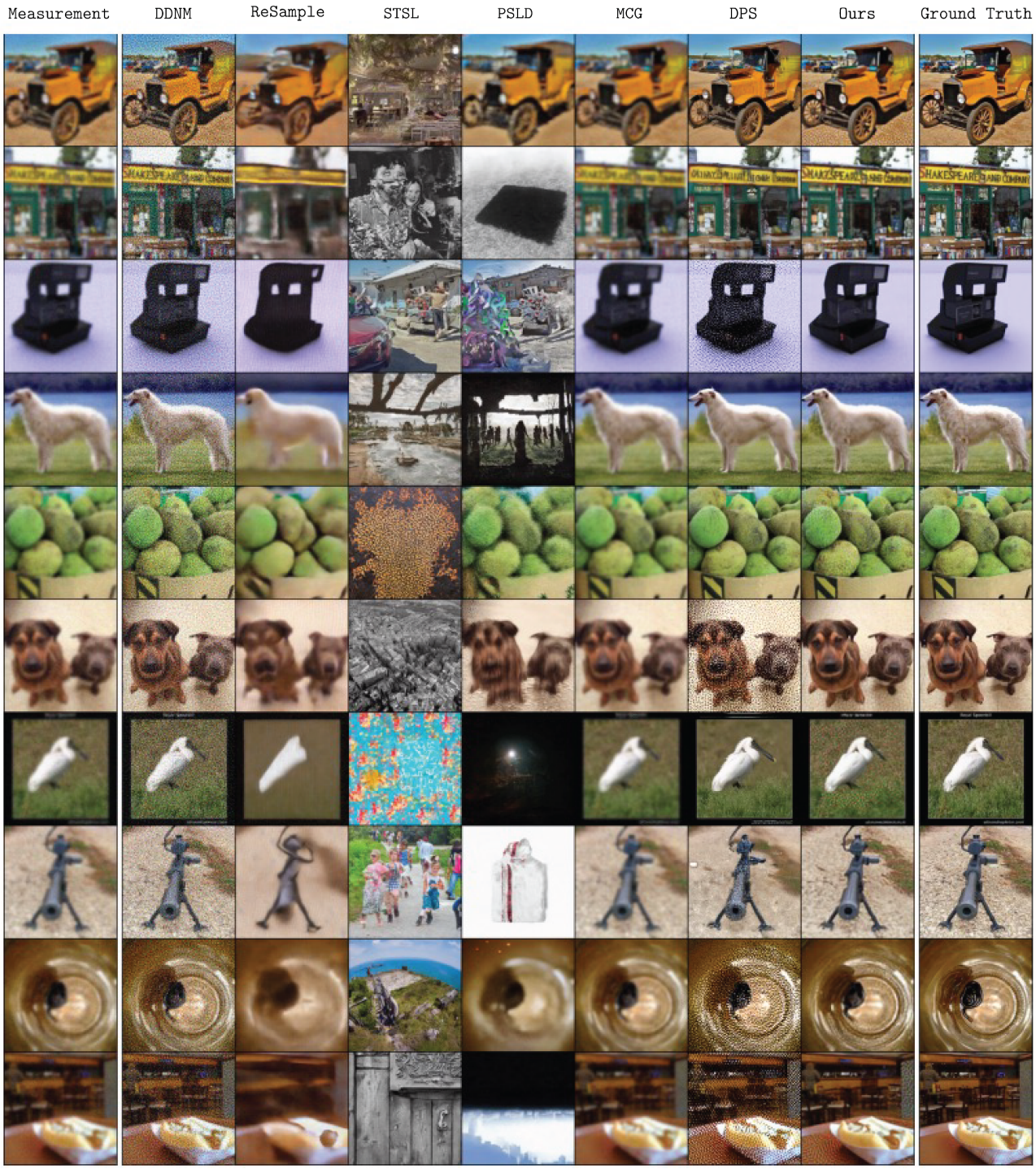


Figure 18: Comparison against competing works on ImageNet 256×256-1K dataset with the Gaussian deblurring task.

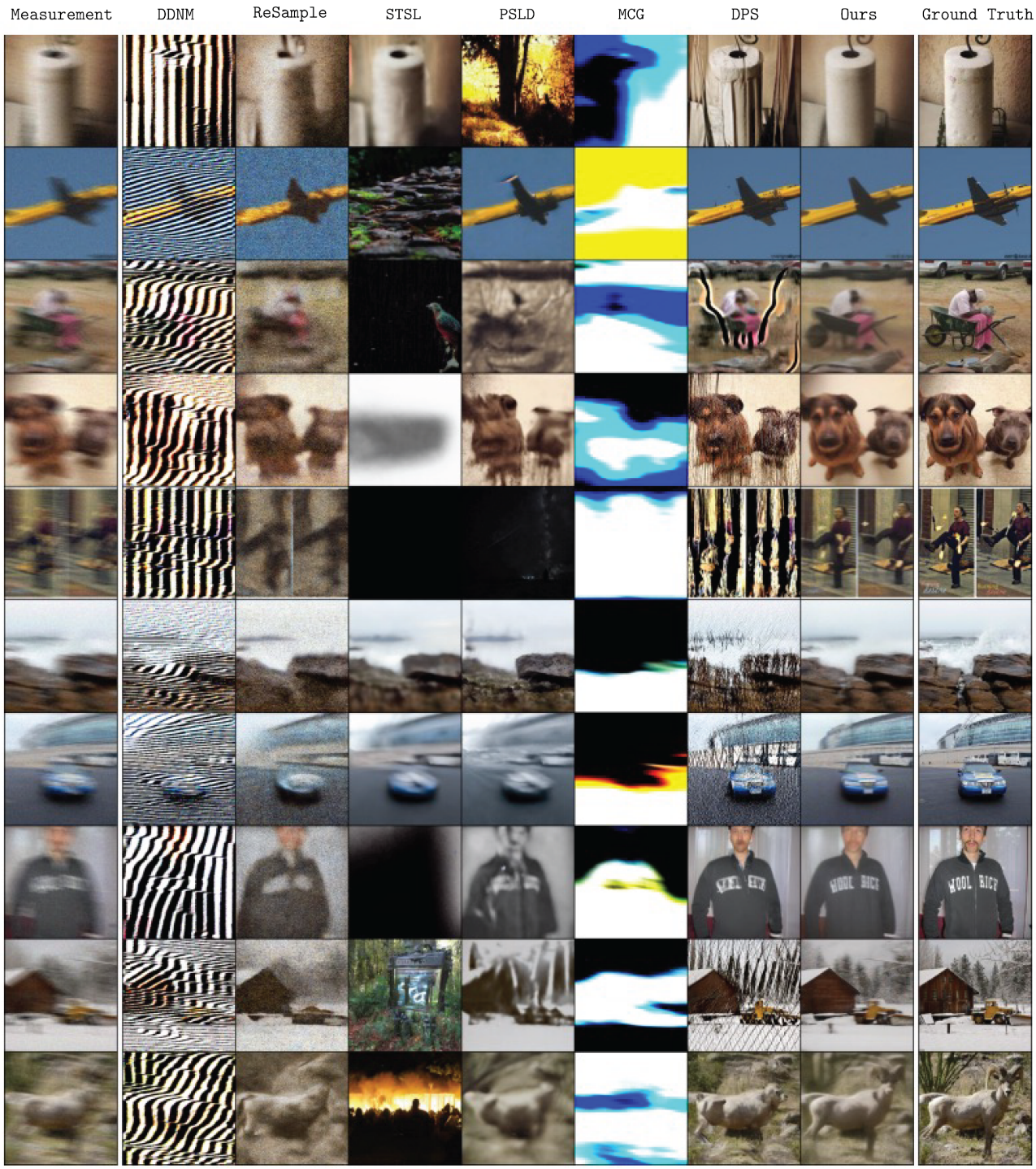


Figure 19: Comparison against competing works on ImageNet 256×256-1K dataset with the motion deblurring task.

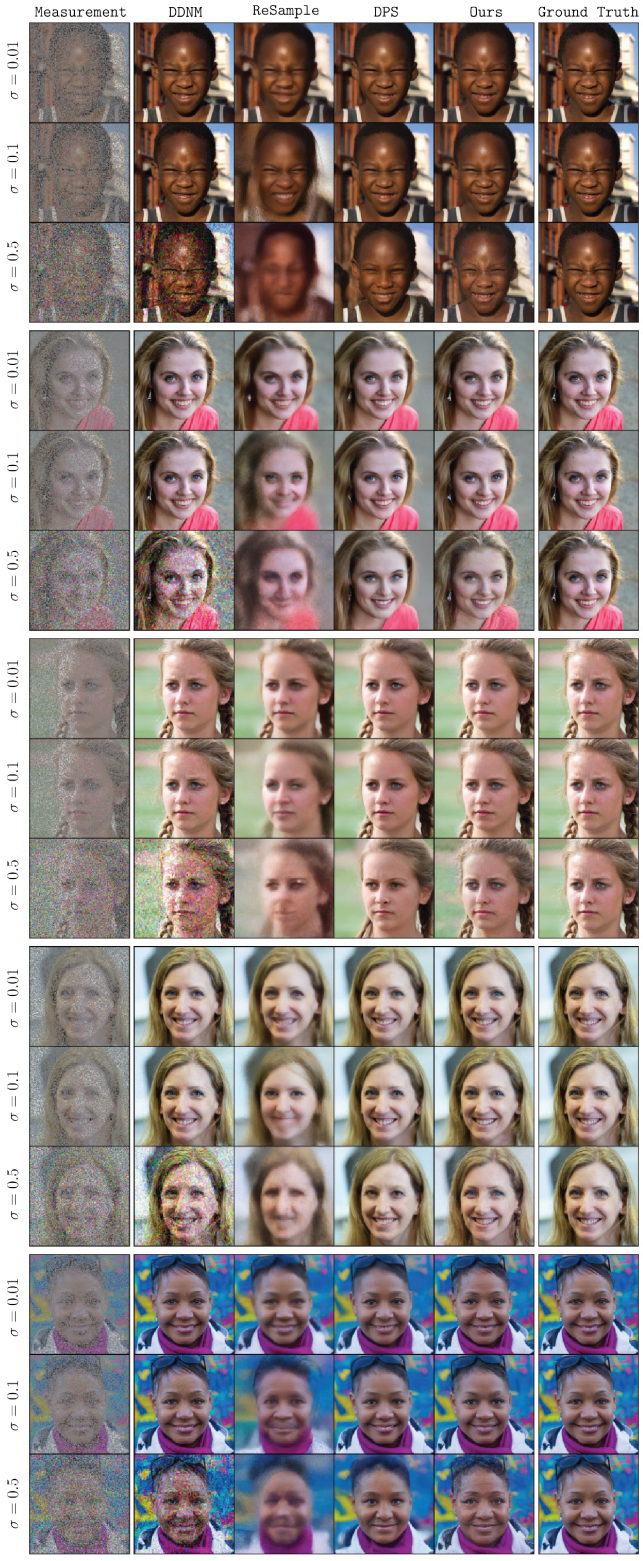


Figure 20: Comparison against competing works on FFHQ 256×256-1K dataset with the random inpainting task at various noise levels.

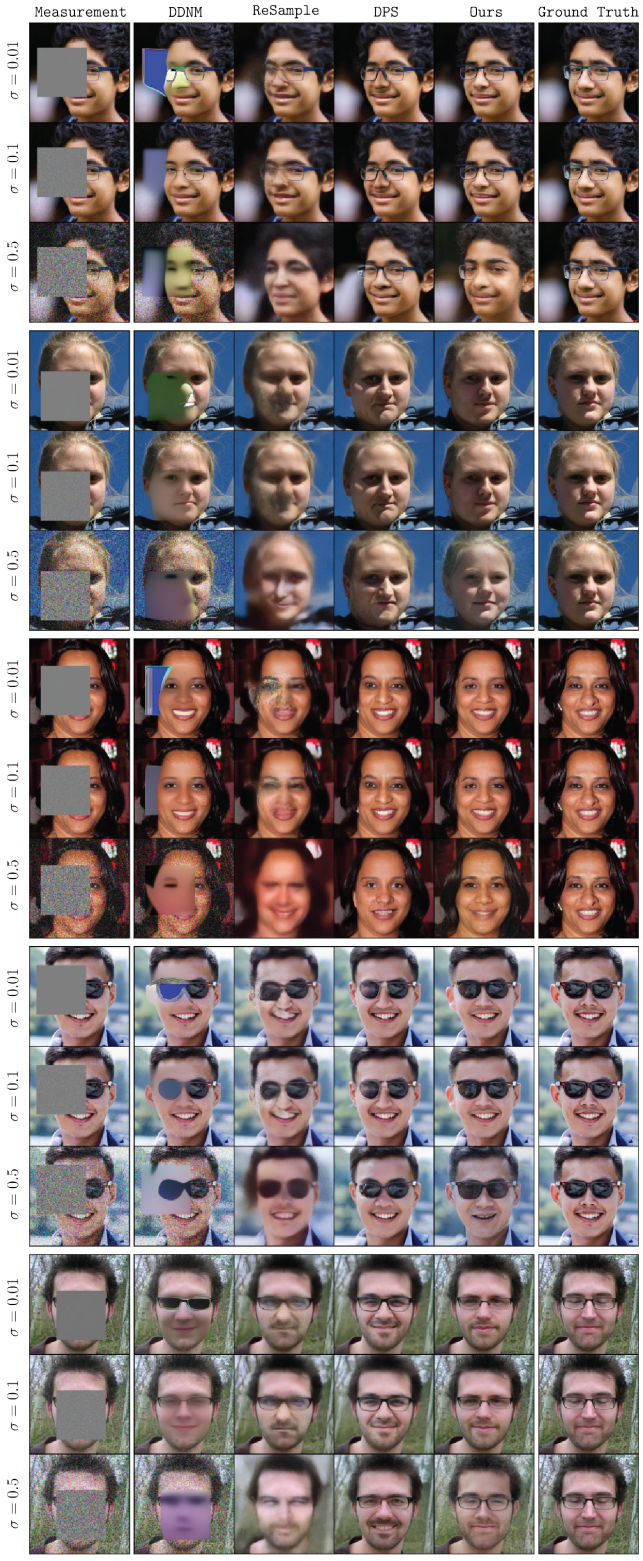


Figure 21: Comparison against competing works on FFHQ 256×256-1K dataset with the box inpainting task at various noise levels.

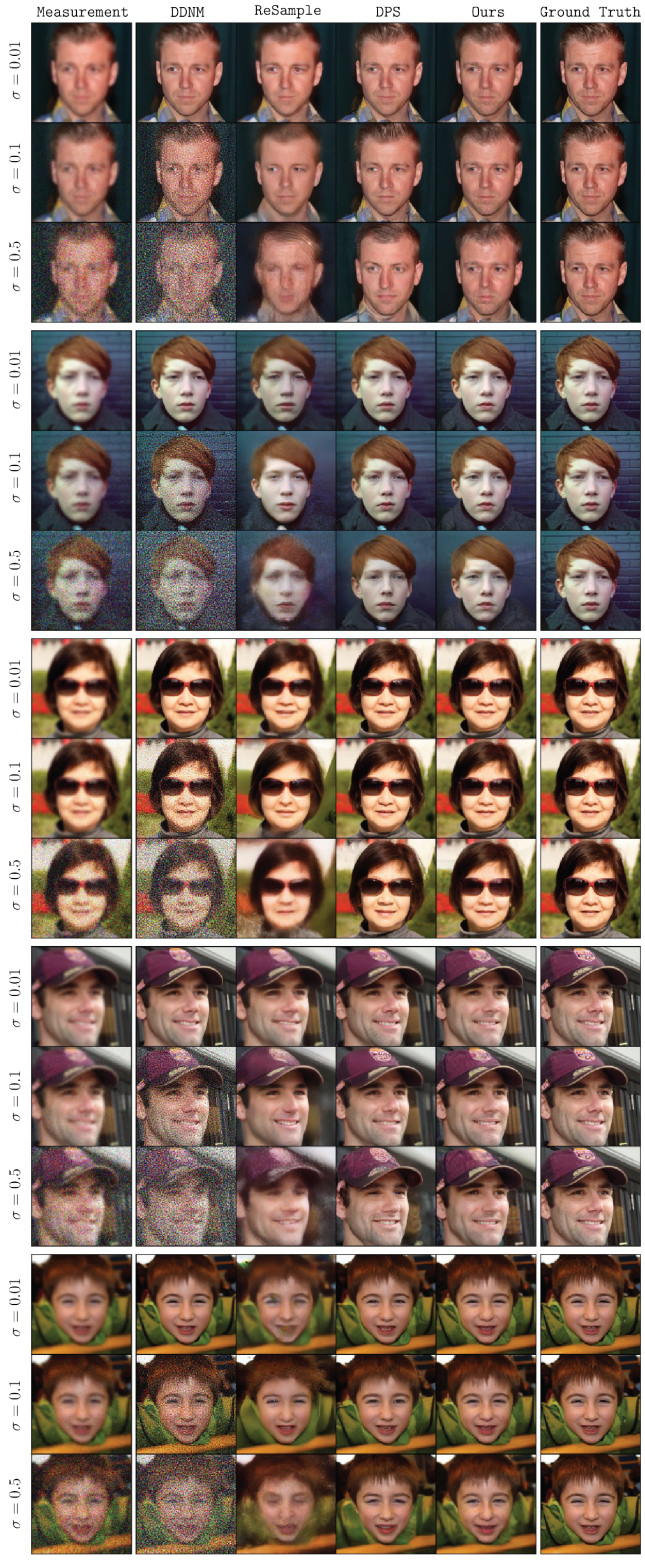


Figure 22: Comparison against competing works on FFHQ 256×256-1K dataset with the Gaussian deblurring task at various noise levels.

Figure 23: Comparison against competing works on FFHQ 256×256-1K dataset with the motion deblurring task at various noise levels.