LEARNED END-TO-END GUIDANCE SCHEDULES FOR DIFFUSION MODELS

Anonymous authors
Paper under double-blind review

Abstract

Diffusion models are a powerful generative paradigm used across multimedia and scientific applications. Guided diffusion methods impose requirements on the generation by adding the gradient of a differentiable loss (the *guidance function*) as a drift term during inference. The weight of this drift (the *guidance scale*) is critical for the trade-off between data quality and requirement satisfaction. To achieve both of these goals, guided diffusion must resort to small guidance scales and lengthy sampling, incurring high computational costs. This work proposes *learned end-to-end guidance schedules* (LEEGS) to achieve these objectives with *fewer sampling steps*. LEEGS trains a time-dependent schedule by minimizing the guidance function over a small set of examples using stochastic gradient descent. Backpropagating through guided sampling is computationally expensive, so LEEGS uses an approximation of the gradient that cuts training time by a factor of 4. We evaluate LEEGS on diverse guidance tasks, including (a) image inpainting, (b) noisy image inverse problems, (c) face-ID-guided generation, and (d) forward and inverse PDE problems, outperforming baselines at equal budget (50 or 100 NFEs), or matching constant guidance with only 10% of the steps.

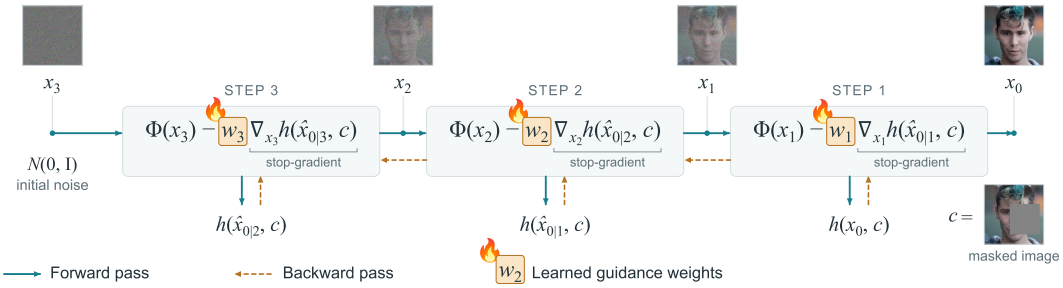
1 Introduction

Diffusion models have been successful in modeling and generating data across many domains, from multimedia tasks [e.g., image synthesis (Ho et al., 2020; Dhariwal & Nichol, 2021), image restoration (Kawar et al., 2022; Chung et al., 2023), text-to-image generation (Rombach et al., 2022; Nichol et al., 2022)] to scientific and engineering applications [e.g., molecular generation (Hoogeboom et al., 2022; Xu et al., 2022), protein generation (Watson et al., 2023), solutions to partial differential equations (PDEs) (Huang et al., 2024a; Bastek et al., 2025), design optimization (Maze & Ahmed, 2023)]. Guided diffusion [e.g., (Dhariwal & Nichol, 2021; Bansal et al., 2023; Wu et al., 2023; Corso et al., 2023)] allows these models to produce data satisfying ancillary requirements, such as the physical viability of samples for scientific and engineering applications (Chen et al., 2024; Christopher et al., 2024a), or solve inverse problems (Chung et al., 2023; Huang et al., 2024a). While several training-free approaches induce a requirement in diffusion models, such as particle-based (Wu et al., 2023; Corso et al., 2023) or projection-based methods (Christopher et al., 2024b;a), if its violation can be expressed as a differentiable loss $h(x)$ (the *guidance function*), then modifying the diffusion with a drift $-\nabla_w h$ of strength $w > 0$ (the *guidance scale*) yields a powerful, general approach [see, e.g. (Dhariwal & Nichol, 2021; Bansal et al., 2023)].

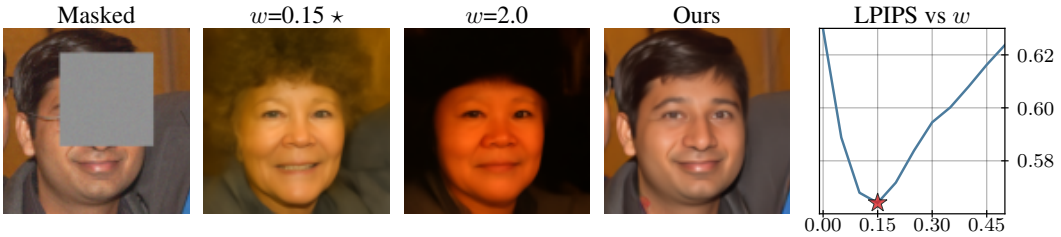
Yet, guidance exacerbates a well-known limitation of diffusion models: slow inference. All guidance techniques face a trade-off between data quality (staying close to the model's learned distribution) and requirement satisfaction (minimizing the violation h). Small scales ($w \ll 1$) do not sufficiently reduce the violation h, yielding samples that violate the requirements; while large scales ($w \gg 1$) cause substantial changes to the original diffusion trajectory, yielding unrealistic samples. To generate high-quality data satisfying requirements, guided generation algorithms such as (Chung et al., 2023; Yu et al., 2023) resort to *small guidance scales* and many inference steps, resulting in high computational cost. Our aim is to reduce the number of steps in the sampling process to balance data quality and the satisfaction of requirements.

Different approaches have been proposed to tackle this problem. For instance, Yang et al. (2024) rescales the guidance gradient relative to the sampling noise at each step, enabling larger step sizes

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) **LEEGS training procedure.** The forward pass (computing $\mathcal{L}(W)$) runs the guided diffusion sampling, while the backward pass (updating W) backpropagates the objective through the sampling trajectory.



(b) LEEGS achieves convincing inpainting results with *only 10 inference steps* while constant guidance only reconstructs a plausible-looking but incorrect face even at $w = 0.15$, which achieves the optimal LPIPS on a held-out set of 100 images. A larger scale ($w = 2.0$) does not improve consistency with the unmasked regions, but produces unrealistic, saturated artifacts instead. See Section 4.1 for details on the noisy box-inpainting task.

and thus shorter trajectories, but still requires choosing a guidance scale. Methods that alternate between denoising and guidance optimization [e.g. (Zampini et al., 2025; Jayaram et al., 2025; Huang et al., 2024b)] control the deviation explicitly via a trust region or a proximal penalty. However, tuning the trust-region radius or penalty weight is as delicate as tuning the guidance scale itself. A different strategy for mitigating this trade-off has been explored in the classifier-free guidance (CFG) setting, where the guidance scale is made time-dependent (Kynkäänniemi et al., 2024; Wang et al., 2024; Sadat et al., 2024; Malarz et al., 2025) following the observation that constant guidance is suboptimal for CFG (Kynkäänniemi et al., 2024). *Learning* a time-dependent scale has likewise been studied for CFG (Yehezkel et al., 2025; Galashov et al., 2025; Sun et al., 2025), but not for loss-based guidance, whose main advantage is that it conditions a pretrained model without retraining.

We bring *learned* time-dependent guidance scales (*guidance schedules*) to loss-guided diffusion, achieving better trade-offs between data quality and requirement satisfaction. LEEGS trains a schedule for a given model, task, and sampler by minimizing the value of the guidance function along the diffusion trajectory, averaged over a small set of examples. This requires backpropagating through the sampling trajectory, a naive implementation of which would incur untenable peak memory and per-batch compute. We use gradient checkpointing to bound the memory cost, similarly to when optimizing other diffusion parameters, such as sampling coefficients (Watson et al., 2022), discretizations (Tong et al., 2025), and noise (Tang et al., 2024). We further propose an approximation of the objective's gradient that reduces per-batch compute. Put together, these improvements allow LEEGS to effectively manage the guidance trade-off in a tractable manner.

The remainder of the paper is organized as follows. We first review guided diffusion sampling in Section 2, then present our problem formulation and method in Section 3. We introduce our main empirical results in Section 4, starting with a description of the applications we consider, namely various guided image generation tasks and PDE problems, followed by further experiments and ablation studies. Finally, we conclude in Section 6.

2 Guided diffusion

Modern diffusion and score-based generative models (Ho et al., 2020; Song et al., 2021) use a learned noise-prediction network $\epsilon_\theta(x, t)$ that defines a vector field whose reverse-time dynamics

are integrated numerically to generate samples $x_0 \in \mathcal{X}$ from a random starting point $x_S \sim p(x_S) = \mathcal{N}(0, \mathbf{I})$. We denote by

$$x_{s-1} = \Phi(x_s, t_s; \epsilon_\theta), \qquad s = S, \dots, 1, \tag{1}$$

a single step of this reverse diffusion, where Φ represents a (possibly stochastic) numerical integrator [e.g., DDIM (Song et al., 2022), DDPM(Ho et al., 2020), or higher-order solvers (Karras et al., 2022)] and $T = t_S > \ldots > t_1 > t_0 = 0$ is a discretization of the diffusion time. The resulting trajectory $x_{0:S}$ is a Markov chain with joint distribution $p(x_{0:S})$ and marginal distribution at time t_s denoted $p(x_s)$, for $s = S, \dots, 0$. Since ϵ_θ is fixed throughout, we suppress θ and write $\hat{x}_{0|s} \coloneqq \hat{x}_0(x_s, t_s; \epsilon_\theta)$ for the one-step denoised estimate of x_s, recovered from $\epsilon_\theta(x_s, t_s)$ via Tweedie's formula (Appendix A (6)). The diffusion model is trained so that $p(x_0)$ approximates the data distribution.

The reverse diffusion process can be modified to produce samples that satisfy additional properties or requirements using the mechanism of guidance. The guided generation process uses a non-negative guidance function $h : \mathcal{X} \times \mathcal{C} \to \mathbb{R}_+$ that measures the violation of a desired property or requirement, often parametrized by a condition $c \in \mathcal{C}$. The goal is to generate samples x_0 that minimize $h(x_0, c)$ [ideally, $h(x_0, c) = 0$]. Consider the example of image inpainting where the goal is to restore an image missing a block of pixels. Here, the condition c is the degraded image itself and the guidance function is the normed difference between the generated sample and c at the observed pixel locations (see Sec. 4.1 for more examples).

The guidance function induces a vector field via its gradient $\nabla_{x_s} h\big(\hat{x}_{0|s}, c\big)$ evaluated at the one-step denoised estimate, which can be used to modify the diffusion reverse-time dynamics from (1). Given a guidance schedule $W \in \mathbb{R}^S_{\geq 0}$ that collects the time-dependent guidance scales w_s, the guided trajectory is obtained via

$$x_{s-1} = \Phi(x_s, t_s; \epsilon_\theta) - w_s \, \nabla_{x_s} h\big(\hat{x}_{0|s}, c\big), \qquad s = S, \dots, 1. \tag{2}$$

We follow Chung et al. (2023) and evaluate h on $\hat{x}_{0|s}$ instead of x_s, since the former is a better proxy for the final sample. Note that computing $\nabla_{x_s} h\big(\hat{x}_{0|s}, c\big)$ requires differentiating through the diffusion model, since $\hat{x}_{0|s}$ depends on $\epsilon_\theta(x_s, t_s)$. We denote by $p_W(x_{0:S} \mid c)$ the joint distribution of the resulting trajectory and by $p_W(x_s \mid c)$, $s = S, \dots, 0$, the marginal distributions of the samples at time t_s. As the guided marginals $p_W(x_s \mid c)$ ($s > 0$) get farther from the unguided $p(x_s)$, sampling error accumulates, pushing $p_W(x_0 \mid c)$ away from the data distribution and producing artifacts. Prior methods control this deviation by constraining each guidance step [a trust region or a proximal step (Jayaram et al., 2025; Huang et al., 2024b)], but the constraint radius or penalty weight is itself as delicate a hyperparameter as the guidance scale. Our approach instead *learns* time-dependent scales that directly optimize h.

3 Learned end-to-end guidance schedules

Objective. We seek a guidance schedule W that yields samples that both satisfy the requirement encoded by $h(x_0, c)$ and stay close to the unguided distribution $p(x_0)$. We impose the requirements on average over conditions c drawn from a fixed distribution $p_{\mathcal{C}}$, with the pre-trained model ϵ_θ held fixed. The natural objective is to minimize the guidance function at the terminal sample alone, i.e. $\mathbb{E}_{c,\, x_0 \sim p_W(\cdot|c)}\, [h(x_0, c)]$; however, this terminal-only objective is hard to optimize directly, since the gradient of $h(x_0, c)$ with respect to w_s decays for steps far from the end of the trajectory (large s), leaving long trajectories with little training signal (Appendix C.6). Including all the terms $h(\hat{x}_{0|s}, c)$ ($0 \leq s < S$) along the trajectory (with the convention that $\hat{x}_{0|0} = x_0$) restores a usable gradient at every step, giving the objective we minimize :

$$\mathcal{L}(W) = \mathbb{E}_{c \sim p_{\mathcal{C}},\, x_{0:S} \sim p_W(\cdot|c)} \left[\sum_{s=0}^{S-1} h(\hat{x}_{0|s}, c) \right]. \tag{3}$$

Computing the objective for a single draw (x_S, c) requires simulating the full guided trajectory (2), a cost that scales linearly with the number of sampling steps S.

Gradient computation. Assume that the sampler Φ is deterministic, so that the entire trajectory $x_{0:S}$ (and hence every $\hat{x}_{0|s}$) is deterministic given c, x_S, and W. Writing $\ell(W; x_S, c) \coloneqq$

$\sum_{s=0}^{S-1} h(\hat{x}_{0|s}, c)$ for this now-deterministic function of x_S, the objective of (3) becomes $\mathcal{L}(W) = \mathbb{E}_{c\sim p_\mathcal{C},\, x_S\sim p(x_S)}\left[\ell(W; x_S, c)\right]$. Unlike $p_W(\cdot \mid c)$, $p_\mathcal{C}$ and $p(x_S)$ are fixed and do not depend on W. Removing W from the sampling distribution lets us differentiate under the expectation, so that $\nabla_W \mathcal{L}(W) = \mathbb{E}_{c,\, x_S}\left[\nabla_W \ell(W; x_S, c)\right]$ can be estimated by sampling $c \sim p_\mathcal{C}$, $x_S \sim p(x_S)$ and computing the average of $\nabla_W \ell(W; x_S, c)$ over these samples. For stochastic Φ, a similar argument holds by reparameterizing the sampler noise (Kingma & Welling, 2013). Define $g_s = \nabla_{x_s} h(\hat{x}_{0|s}, c)$, the guidance gradient at step s. Expanding the derivative of $\ell(W; x_S, c)$ with respect to w_k, $k \in \{1, \ldots, S\}$, we obtain

$$\begin{aligned}\frac{\partial \ell(W; x_S, c)}{\partial w_k} &= \sum_{s=0}^{S-1} \frac{\partial}{\partial x_s} h\big(\hat{x}_{0|s}, c\big) \cdot \frac{\partial x_s}{\partial w_k} \\ &= g_{k-1} \cdot \frac{\partial x_{k-1}}{\partial w_k} + \sum_{s=0}^{k-2} g_s \cdot \frac{\partial x_s}{\partial x_{s+1}} \frac{\partial x_{s+1}}{\partial x_{s+2}} \cdots \frac{\partial x_{k-2}}{\partial x_{k-1}} \frac{\partial x_{k-1}}{\partial w_k}. \end{aligned} \tag{4}$$

To obtain the second equality, observe that x_s does not depend on w_k for $s \geq k$. Differentiating the guided update (2) with respect to x_s gives

$$\frac{\partial x_{s-1}}{\partial x_s} = \frac{\partial \Phi(x_s, t_s; \epsilon_\theta)}{\partial x_s} - w_s \cdot \frac{\partial g_s}{\partial x_s}. \tag{5}$$

Likewise, computing the partial derivative $\frac{\partial x_{k-1}}{\partial w_k}$ by differentiating the guided update (2) with respect to w_k, we obtain $\frac{\partial x_{k-1}}{\partial w_k} = -g_k$. This follows since x_k does not itself depend on w_k, and w_k enters the update only through the linear term $-w_k g_k$. While the first term in (4) captures the effect of w_k on the guidance function at the immediate next step, the second term captures its effect on all later steps. However, the latter term involves repeated computations of $\frac{\partial g_s}{\partial x_s}$ in (5), which involves second derivatives of the network.

Training with the true gradient is therefore prohibitive at large S, so we approximate (5) by ignoring the term $w_s \frac{\partial g_s}{\partial x_s}$. This is equivalent to applying a stop-gradient to the guidance gradient g_s during sampling, so that w_k still affects later steps through the sampler Jacobian $\frac{\partial \Phi}{\partial x_s}$, but not through g_s. This is roughly $4\times$ cheaper to compute than the true gradient (Appendix C.2). We optimize W by projected stochastic gradient descent, clipping negative entries to zero after each step; once trained, W is applied at sampling time with no additional cost. We refer to the resulting method as LEEGS.

Naive backpropagation stores the activations of all S network evaluations along the trajectory, so peak memory grows linearly in S and quickly exceeds GPU capacity. We avoid this with gradient checkpointing (rematerialization) (Kumar et al., 2019; Tong et al., 2025): rather than store a step's activations, we recompute them during the backward pass, making peak memory independent of S in exchange for extra computation.

Regularization. Attempting to drive the guidance function h too low may occasionally push the guided marginals $p_W(x_s \mid c)$ too far from the unguided $p(x_s)$, yielding unrealistic samples, which can be seen as a form of reward hacking (Tang et al., 2024; Zhai et al., 2025). This manifests in Figure 2 as an increase in the LPIPS metric after a certain point in the optimization. This phenomenon can be mitigated either by early stopping, when a held-out reward is available to detect the decline in performance, or by penalizing the magnitude of the guidance schedule W in the training objective as in, e.g. minimizing $\mathcal{L}(W) + \gamma \left\|W\right\|_2^2$ with $\gamma > 0$. Appendix C.3 explores which tasks are prone to reward hacking and how the penalty alleviates it. Typically, even on tasks which *are* prone to reward hacking, it appears only at high S (e.g., $S = 100$), not at lower S (e.g., $S = 25$).

Parametrization. LEEGS optimizes the schedule freely over $W \in \mathbb{R}_+^S$. While W can be constrained to a parametric family, $w_s = \rho_\phi(s)$ with ϕ small, trading flexibility for fewer parameters and a smoother schedule, this requires the functional form to be fixed in advance. Since LEEGS's cost is dominated by the guided sampling, we find this trade-off unnecessary in practice.

4 Experiments

We evaluate LEEGS on guided image generation and PDE problems, comparing it with baseline methods across budgets and analyzing its schedules, sample efficiency, and sensitivity to sampler choice and inference steps.

4.1 Benchmark tasks

Image inpainting. Inpainting seeks to recover a complete image from a partial observation. With condition $c = (y, M)$, where y is the observed image and M a binary mask marking the missing pixels, we consider the guidance function $h(x, c) = \|M \odot x - y\|_2$, where $\odot$ denotes elementwise multiplication. Unlike projection-renoising methods, such as RePaint (Lugmayr et al., 2022), which inject the observed pixels directly and can leave visible seams at low step counts, guided sampling enforces the observed pixels softly. We generate synthetic inpainting problems on ImageNet and CelebA-HQ using the five mask types from Lugmayr et al. (2022): wide and narrow (large or thin contiguous regions are not observed), super-resolution $2\times$ (one pixel kept per 2×2 block), alternating-lines (every other row removed), and half (right half removed). We compare against RePaint at a matched 200-NFE budget and include the original results using 2400 NFEs from Lugmayr et al. (2022) as a reference. We report LPIPS (Zhang et al., 2018), the perceptual distance to the ground truth image (lower is better).

Noisy inverse problems. We use image restoration to illustrate an inverse problem (Kawar et al., 2022; Chung et al., 2023; Wang et al., 2023) with a known degradation operator $\mathcal{A}$. The noisy measurement of the image x_{GT} under condition $c = (y, \mathcal{A})$ is $y = \mathcal{A}(x_{\text{GT}}) + \sigma n$, with an unknown noise level $\sigma > 0$ and $n \sim \mathcal{N}(0, \mathbf{I})$. We choose the guidance function as $h(x, c) = \|\mathcal{A}(x) - y\|_2$. Following Chung et al. (2023), we synthesize problems from held-out FFHQ and ImageNet images and evaluate five tasks at $\sigma = 0.05$: $4\times$ super-resolution, box inpainting (a large centred square removed), random inpainting (each pixel dropped independently with high probability), and Gaussian and motion deblurring (details of the operators are in App. B). Unlike the noiseless inpainting benchmark above, the observations here are corrupted, handicapping methods that inject observed pixels directly, such as RePaint. We compare against recent methods for inverse problems (CDIM (Jayaram et al., 2025), DSG (Yang et al., 2024), and DAPS (Zhang et al., 2025)) at 50 and 100 NFEs, and include the 1000-NFE DPS (Chung et al., 2023) numbers reported in Chung et al. (2023) as a reference. We also compare against constant guidance and heuristic increasing and decreasing linear schedules, each with a single parameter tuned by grid search on a held-out set (App. B). Alongside LPIPS, we report FID (Heusel et al., 2017), the distributional distance between generated samples and a reference set (lower is better).

Face-ID guidance. Following Yu et al. (2023); Yang et al. (2024), we generate images of a target identity given a single reference photo x_{ref} (the condition c), using the guidance function $h(x, x_{\text{ref}}) = \|F(x) - F(x_{\text{ref}})\|_2$, where F is the feature embedding of a pretrained face-recognition network [we use the ArcFace model (Deng et al., 2019)]. Such a network embeds faces so that images of the same person lie close together, so minimizing h steers the generated image toward the reference identity. We compare against constant guidance and DSG (Yang et al., 2024) on CelebA-HQ at 50 and 100 NFEs. We report FID and the guidance value itself, the *FaceID score* (identity distance to the reference, lower is better).

PDE problems. Diffusion models can solve partial differential equations (PDEs) from sparse observations, complementing classical (Trefethen, 1996; Langtangen & Mardal, 2019) and learned (Raissi et al., 2019; Li et al., 2020) solvers. Following DiffusionPDE (Huang et al., 2024a), we consider boundary-value problems specified by a coefficient field a and its solution field u. The diffusion model is pretrained on a dataset of solved instances and represents the joint distribution over coefficient–solution pairs (a, u) rendered as images. Given observations $\{a_{\text{obs},i}\}_{i=1}^{N_{\text{obs}}}$ of the coefficient at locations $\{z_{\text{obs},i}\}_{i=1}^{N_{\text{obs}}}$, the inverse-problem guidance function is $h_{\text{inv}}(x, c) = \sqrt{\frac{1}{N_{\text{obs}}} \sum_{i=1}^{N_{\text{obs}}} \left(a(z_{\text{obs},i}) - a_{\text{obs},i}\right)^2}$ with condition $c = (a_{\text{obs}}, z_{\text{obs}})$; the forward guidance function is defined analogously on the solution field, $h_{\text{fwd}}(x, c) = \sqrt{\frac{1}{N_{\text{obs}}} \sum_{i=1}^{N_{\text{obs}}} \left(u(z_{\text{obs},i}) - u_{\text{obs},i}\right)^2}$ with condition $c = (u_{\text{obs}}, z_{\text{obs}})$. We evaluate forward (recover u from a) and inverse (recover a from u) problems for the Helmholtz, Poisson, and

Darcy equations, with $\approx 3\%$ of pixels observed. We compare against constant guidance and heuristic increasing and decreasing linear schedules at 50 and 100 steps, and include the DiffusionPDE (Huang et al., 2024a) results at 2000 steps as a reference. While DiffusionPDE additionally applies a PDE-residual guidance term over a small part of the trajectory, we found little benefit from it and omit it for both constant guidance and LEEGS. Following DiffusionPDE, constant guidance uses a second-order sampler; training LEEGS with a second-order sampler did not improve on results obtained with a first-order one. We report the relative error to the ground-truth solution, or the error rate for the binary-valued Darcy coefficient.

	NFEs	SR (x4)		Inpaint (box)		Inpaint (random)		Deblur (gauss)		Deblur (motion)	
		LPIPS	FID	LPIPS	FID	LPIPS	FID	LPIPS	FID	LPIPS	FID
Const.	50	0.409	58.53	0.465	59.06	0.400	47.36	0.430	54.30	0.466	52.74
DAPS	50	0.316	44.60	0.237	38.44	0.497	136.55	0.314	43.76	0.272	45.29
DSG	50	0.274	45.67	0.361	75.21	0.310	47.10	0.313	50.36	0.640	143.59
CDIM	≈ 50	0.250	39.39	0.178	**31.51**	0.245	35.81	0.273	**32.83**	0.283	**30.12**
Linear increase	50	0.224	**35.08**	0.189	36.12	0.199	34.03	0.256	42.52	0.219	40.35
Linear decrease	50	0.521	63.17	0.540	60.65	0.509	62.47	0.525	59.88	0.522	61.50
LEEGS (ours)	50	**0.222**	35.52	**0.175**	33.11	**0.197**	**33.63**	**0.243**	42.55	**0.211**	39.24
Const.	100	0.235	32.75	0.341	47.69	0.259	31.83	0.298	37.57	0.396	36.62
DAPS	100	0.287	39.19	0.230	38.15	0.406	84.38	0.291	39.55	0.241	39.27
DSG	100	0.260	38.16	0.308	73.86	0.250	36.92	0.283	40.14	0.476	62.95
CDIM	≈ 100	0.242	37.43	0.179	31.61	0.243	33.41	0.258	**31.17**	0.265	**28.19**
Linear increase	100	0.218	33.13	0.146	26.72	**0.174**	**26.06**	0.241	36.27	0.195	31.43
Linear decrease	100	0.400	47.74	0.463	50.40	0.405	48.18	0.412	44.91	0.408	46.45
LEEGS (ours)	100	**0.217**	**32.40**	**0.141**	**25.39**	0.176	26.44	**0.228**	34.05	**0.181**	30.29
DPS	1000	0.214	39.35	0.168	33.12	0.212	21.19	0.257	44.05	0.242	39.92

Table 1: Comparison for noisy image inverse tasks (FFHQ). **Green** (bold): best; **blue**: second best (non-reference rows).

	NFEs	SR (x4)		Inpaint (box)		Inpaint (random)		Deblur (gauss)		Deblur (motion)	
		LPIPS	FID	LPIPS	FID	LPIPS	FID	LPIPS	FID	LPIPS	FID
Const.	50	0.666	93.92	0.439	113.71	0.651	104.47	0.649	94.45	0.654	94.34
DAPS	50	0.402	53.43	0.296	70.73	0.597	206.29	0.405	**54.89**	**0.297**	**35.40**
DSG	50	0.428	71.28	0.483	107.38	0.461	88.15	0.471	78.82	0.660	149.75
CDIM	≈ 50	0.379	56.01	**0.245**	**51.44**	0.336	46.78	0.405	61.50	0.404	67.77
Linear increase	50	0.373	55.66	0.354	87.84	0.344	47.74	0.468	77.56	0.425	72.27
Linear decrease	50	0.689	101.31	0.707	107.11	0.702	104.71	0.689	94.88	0.695	99.46
LEEGS (ours)	50	**0.333**	**46.86**	0.286	55.07	**0.272**	**31.54**	**0.395**	58.87	0.338	50.76
Const.	100	0.488	76.41	0.299	82.75	0.416	73.85	0.536	77.64	0.595	85.46
DAPS	100	0.378	48.92	0.287	64.77	0.531	155.06	0.384	52.00	**0.276**	**32.81**
DSG	100	0.343	45.34	0.453	126.49	0.347	47.72	0.419	63.25	0.672	193.45
CDIM	≈ 100	0.363	49.37	0.241	47.95	0.330	43.45	0.391	57.24	0.381	58.27
Linear increase	100	0.355	50.97	0.282	73.71	0.272	31.37	0.437	65.23	0.380	58.26
Linear decrease	100	0.609	88.48	0.647	110.93	0.619	94.36	0.609	83.92	0.609	86.52
LEEGS (ours)	100	**0.319**	**39.62**	**0.234**	**47.73**	**0.235**	**23.69**	**0.368**	**50.38**	0.310	42.33
DPS	1000	0.337	50.66	0.262	38.82	0.303	35.87	0.444	62.72	0.389	56.08

Table 2: Comparison for noisy image inverse tasks (Imagenet). **Green** (bold): best; **blue**: second best (non-reference rows).

Dataset	Method	NFEs	Wide	Narrow	S. res. (2x)	Alternate Lines	Half
Imagenet	RePaint	200	0.186	0.18	0.622	0.394	0.368
	LEEGS (ours)	200	**0.182**	**0.151**	**0.147**	**0.111**	**0.346**
	RePaint	$\approx$2400	0.134	0.064	0.183	0.089	0.304
CelebA-HQ	RePaint	200	0.128	0.128	0.205	0.049	0.283
	LEEGS (ours)	200	**0.102**	**0.080**	**0.065**	**0.042**	**0.229**
	RePaint	$\approx$2400	0.059	0.028	0.029	0.009	0.165

Table 3: Comparison for image inpainting tasks for CelebA-HQ and Imagenet. **Green** (bold): best (non-reference rows).

Method	NFEs	FaceID Score	FID
Const	50	0.456	77.16
DSG	50	0.466	93.38
LEEGS (ours)	50	**0.401**	**76.90**
Const	100	0.376	73.60
DSG	100	0.395	80.24
LEEGS (ours)	100	**0.324**	**71.33**

Table 4: Comparison for Face ID tasks for CelebA-HQ. **Green** (bold): best; **blue**: second best.

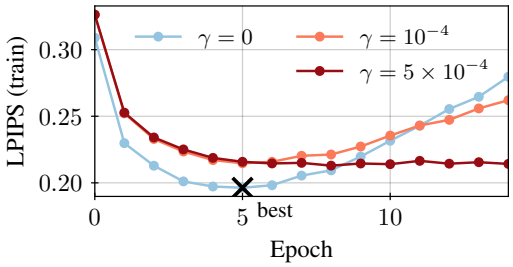


Figure 2: Reward hacking at $S = 100$ for FFHQ superresolution, subsequently solved by applying weight decay $\gamma \|W\|_2^2$.

Method	NFEs	Poisson		Helmholtz		Darcy	
		Forward	Inverse	Forward	Inverse	Forward	Inverse
Const.	50x2	19.34%	33.11%	27.56%	32.15%	21.54%	15.58%
Linear increase	50x1	27.59%	46.43%	39.72%	45.42%	24.70%	20.98%
Linear decrease	50x1	10.25%	29.81%	17.07%	29.84%	**11.03%**	**14.05%**
LEEGS (ours)	50x1	**5.80%**	**29.36%**	**10.03%**	**29.71%**	11.33%	17.84%
Const.	100x2	15.62%	30.10%	22.96%	29.54%	18.17%	16.71%
Linear increase	100x1	22.23%	28.46%	37.68%	28.93%	23.26%	18.36%
Linear decrease	100x1	7.61%	25.39%	12.25%	26.01%	8.53%	**14.79%**
LEEGS (ours)	100x1	**5.84%**	**25.11%**	**9.57%**	**25.84%**	**8.20%**	16.58%
DPDE	2000x2	4.5%	20.0%	8.8%	22.6%	2.5%	3.2%

Table 5: Comparison for forward and inverse PDE problems with sparse observations. **Green** (bold): best; **blue**: second best (non-reference rows).

4.2 Main results

For each task and step budget, LEEGS trains a guidance schedule on 30 examples drawn from a training pool disjoint from the corresponding test set, using Adam with batch size 1 and training separately for every subtask and NFE budget. This pool has 90 images for every task except FFHQ, where we instead use a cross-validation-style split. All evaluations (except reference rows taken from original works) used test sets of size: 950 for FFHQ (Table 1); 1000 for ImageNet noisy inverse problems (Table 2); 100 for the inpainting and Face-ID tasks (Tables 3 and 4); and 1000 for Helmholtz and Darcy, 924 for Poisson, for the PDE problems (Table 5). Exact training and dataset details (e.g., learning rates, epochs, pretrained models) are given in App. B. All results for LEEGS are the mean of three independent runs (with randomly drawn training sets).

Noisy inverse problems. Tables 1 and 2 report LPIPS and FID on FFHQ and ImageNet. On FFHQ, LEEGS is the best or second-best method in nearly every column at both 50 and 100 NFE, with the best LPIPS on four of the five subtasks. On ImageNet, LEEGS is best or second-best in every column at both 50 and 100 NFE, and the best in 4 out of 5 subtasks for ImageNet at 100 NFE with respect to both metrics. The tuned linear-*increasing* schedule is itself a strong heuristic here, while linear-*decreasing* is consistently poor, illustrating that effective schedule shapes are task-dependent (Fig. 3) and LEEGS recovers a good one without that prior. LEEGS at 100 NFE also beats the 1000-NFE DPS reference on both LPIPS and FID for most subtasks, a $10\times$ reduction in steps.

Image inpainting. At a matched 200-NFE budget, LEEGS improves over RePaint on every mask type and both datasets (Table 3); RePaint with $12\times$ the budget stays ahead on most masks.

Face-ID guidance. Table 4 reports FaceID similarity and FID for guided face generation. At 50 NFE, all three rows are similar, except DSG's FID which is significantly worse. However, at 100 NFE, LEEGS performs best on both metrics, improving over DSG and constant guidance.

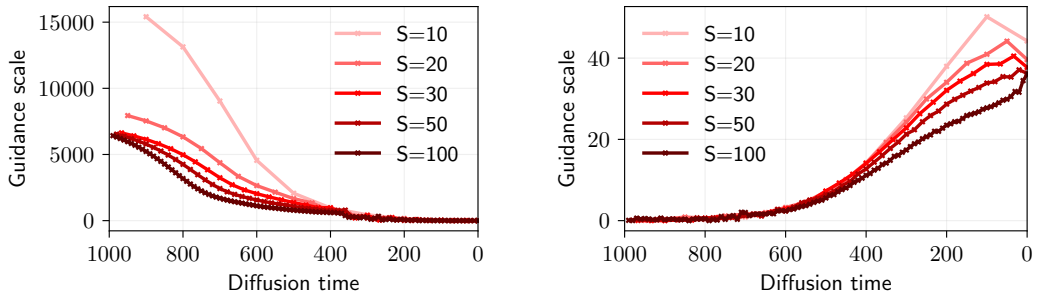


Figure 3: Learned guidance schedules for the forward Helmholtz problem (left) and the noisy box inpainting problem (right). The schedules are similar across different step budgets for a particular problem, but differ between problems, highlighting the need for learning methods.

PDE problems. LEEGS attains the lowest relative error on 4 of the 6 forward/inverse problems at 50 sampling steps, and on 5 of 6 at 100 steps (Table 5), in both cases at half the NFE of the constant-guidance baseline (which runs a second-order sampler). However, DiffusionPDE's results, which use $20\times$ more steps, retain an advantage. The linear-*decreasing* schedule remains a strong fixed baseline here: it is in fact the best method for the inverse Darcy problem at both step budgets, and for the Darcy forward problem at 50 steps, though the same heuristic is among the worst on the noisy-inverse tasks, where linear-*increasing* dominates instead. Which fixed shape works is task- and problem-specific and not knowable in advance; LEEGS matches or outperforms both heuristics on most problems.

4.3 Analysis

Learned schedules. Figure 3 shows example schedules learned by LEEGS. For a given problem they have a similar shape across step budgets, but they differ substantially between problems: increasing schedules help the noisy inverse problems, whereas decreasing schedules help the PDE tasks, mirroring the performance of the linear-schedule baselines in Tables 1, 2, and 5. Schedules differ even between closely related tasks: among the five FFHQ inverse problems, App. C.4 shows that the schedule learned for a task is best on that task in nearly every case, motivating the need for a learned, task-specific schedule.

Beyond DPS-style sampling. LEEGS requires only a differentiable sampling trajectory and makes no assumption on the form of the sampler. Beyond stochastic (DDPM; Tables 1, 2, 3) and deterministic (Euler; Table 5) samplers, we apply it without modification to a sampler that alternates guided reverse diffusion with *renoising*, a common ingredient of inverse-problem solvers such as RePaint (Lugmayr et al., 2022), DDNM (Wang et al., 2023), and DAPS (Zhang et al., 2025). App. D shows that this sampling scheme when combined with LEEGS can perform competitively with the vanilla version of LEEGS on the FFHQ inverse problems (with better results for box and random inpainting), highlighting the flexibility of LEEGS. Moreover, the learned schedules for this scheme consist of a sequence of hill-shaped segments (Fig. 17). Such a schedule would be hard to recover even with a priori knowledge, while the cost of learning them with LEEGS is unchanged.

Effect of inference steps. LEEGS generally improves with more inference steps, with diminishing returns (Tables 1–5; App. C.5). As the budget grows, the mean guidance weight decreases monotonically (Fig. 3): more steps need less aggressive per-step guidance. Since the schedule's shape is largely shared across budgets and its magnitude follows a predictable trend, a schedule trained at small S can be interpolated and rescaled to larger S at inference time, offering a way to reduce the training cost at large S (App. C.5).

Sample efficiency. LEEGS is trained on only 30 examples per subtask. Both the learned schedule and the resulting test performance are largely insensitive to the size of the training set, changing little beyond 30 samples (App. C.1).

5 Related work

Guidance scheduling (for CFG). CFG (Ho & Salimans, 2022) steers a conditional diffusion model toward a condition y (e.g., a class label or text embedding) by combining the conditional and unconditional noise predictions $\epsilon_\theta(x_s, y)$ and $\epsilon_\theta(x_s, \emptyset)$ ($\emptyset$ the null condition) with a guidance scale w. As in our setting, w must be chosen, and can be varied over the course of generation. Several works vary the CFG scale over the trajectory. It was shown by Kynkäänniemi et al. (2024) that a nontrivial scale ($w \neq 1$) is only needed within a limited interval, with the conditional noise prediction $\epsilon_\theta(x_s, y)$ used alone elsewhere. Fixed schedules were compared and proposed by Wang et al. (2024); Sadat et al. (2024); Malarz et al. (2025), and adaptive per-step scales have been derived from the two noise predictions (Zhu et al., 2026; Pavasovic et al., 2025), from estimated log-likelihoods (Koulischer et al., 2025), or by a stepwise greedy search (Papalampidi et al., 2025).

Methods that *learn* the guidance scale [e.g., (Yehezkel et al., 2025; Galashov et al., 2025; Sun et al., 2025; Azangulov et al., 2026)] are closest to our work. Azangulov et al. (2026) is a primarily theoretical work in which the guidance scale is learned as a function of the sample x_s by optimizing an optimal-control objective. Yehezkel et al. (2025); Galashov et al. (2025) learn guidance networks that predict the CFG guidance scale as a function of the intermediate sample x_s and time t_s. The training objectives of these methods do not use rewards and instead rely on labeled data. Sun et al. (2025) uses an evolutionary search to learn a sparse guidance schedule supervised by a constant-guidance reference process; this is expensive, requiring many 1000-step reference rollouts. To our knowledge, LEEGS is the first to *learn* guidance *schedules* for loss-guided diffusion.

Constrained diffusion. Constrained diffusion methods interpret the guidance function h as a constraint function and seek to minimize it, as in our setting. These include projection (Christopher et al., 2024b; Chen et al., 2026; Christopher et al., 2024a), trust-sampling (Huang et al., 2024b), proximal (Zampini et al., 2025; Jayaram et al., 2025), and consensus (Christopher et al., 2026; Blanke et al., 2026) methods. Projection-based methods ensure constraint satisfaction, but closed-form projections are typically only available for convex constraints, so several methods approximate them with an inner optimization loop (Christopher et al., 2025). Moreover, projections take arbitrarily large guidance steps, which can deviate substantially from the distribution induced by the diffusion model; this is typically mitigated by renoising the projected sample, which lengthens the sampling process. Proximal and trust-region sampling methods explicitly address the guidance trade-off by optimizing the constraint only up to a threshold, or within a fixed distance of the denoised iterate. However, the threshold or radius is itself a hyperparameter as delicate to set as the guidance scale.

Consensus-based methods take another approach, splitting inference into two trajectories (one applying projections, one following the unguided reverse process) and annealing them to coincide by the end of sampling. A guidance term implicitly pulls the trajectories together, and Blanke et al. (2026) sets its scale by dual ascent. Consensus methods are closely related to guidance and could be combined with LEEGS. Unlike the projection- and threshold-based methods above, though, LEEGS neither projects nor renoises: it learns a single schedule offline and applies it during ordinary guided sampling at no extra cost.

6 Conclusion

LEEGS is a simple, general method for learning time-dependent guidance schedules for loss-guided diffusion. It trains a schedule on as few as 30 examples by backpropagating the guidance loss through the sampler. Across image inpainting, noisy inverse problems, face-ID guidance, and PDE problems (Section 4), it is competitive with strong task-specific baselines and matches the 1000-NFE DPS reference using only 100 steps. Two findings stand out: a *learned* schedule beats constant guidance at a matched budget, and the useful schedule shape is task-dependent, so it should be learned, not fixed by heuristic. LEEGS is also robust to sampler choice, working with stochastic (DDPM) and deterministic (Euler) samplers alike.

Limitations. Training cost for LEEGS grows with the number of steps S (the objective spans the full trajectory), making very large S costly. While our gradient approximation (Section 3) is efficient, it may be less effective when the guidance field is highly sensitive to x_s. Finally, while we learn a *task-specific* schedule; adapting it to the condition or initial noise remains open.

AI use statement

In this work, we used LLM tools to implement and modify our core codebase, specifically parts of the LEEGS training script, samplers, guidance functions, heuristic baselines such as linear increasing, experiment orchestration scripts, implementation of LEEGS with renoising etc. LLM tools were also used to write notebooks for plotting and analysis that were used to produce the paper's figures and tables, and they were also used to assist in the interpretation of results. LLM tools were used to assist in writing and editing the main text of the paper, and were used to generate initial drafts of several appendix sections. AI tools were also used to search for relevant papers in the early stages of the project, in conjunction with conventional tools for literature search. We did not use generative AI tools to generate synthetic datasets; to develop theoretical models, conceptual frameworks, or for cleaning and reformating datasets. The remaining required disclosure tasks— formulating mathematical claims; providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, assisting with translation are not applicable to this work. We have reviewed all AI-assisted work, including code, code outputs, and text. Critical parts of the code implementing the method or experiments were manually reviewed. All LLM generated text was manually reviewed and edited by the authors. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

References

Iskander Azangulov, Peter Potaptchik, Qinyu Li, Eddie Aamari, George Deligiannidis, and Judith Rousseau. Adaptive diffusion guidance via stochastic optimal control, 2026.

Arpit Bansal, Hong-Min Chu, Avi Schwarzschild, Soumyadip Sengupta, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Universal guidance for diffusion models. In *CVPR Workshops*, pp. 843–852, 2023.

Jan-Hendrik Bastek, WaiChing Sun, and Dennis Kochmann. Physics-informed diffusion models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=tpYeermigp`.

Matthieu Blanke, Yongquan Qu, Sara Shamekh, and Pierre Gentine. Strictly constrained generative modeling via split augmented langevin sampling, 2026.

Jian Chen, Ruiyi Zhang, Yufan Zhou, and Changyou Chen. Towards aligned layout generation via diffusion model with aesthetic constraints. *arXiv.org*, 2024. doi: 10.48550/ARXIV.2402.04754.

Siyuan Chen, Minghao Guo, Caoliwen Wang, Anka He Chen, Yikun Zhang, Jingjing Chai, Yin Yang, Wojciech Matusik, and Peter Yichen Chen. Physically valid biomolecular interaction modeling with gauss-seidel projection, 2026.

Jacob Christopher, Stephen Baek, and Ferdinando Fioretto. Projected generative diffusion models for constraint satisfaction. *arXiv.org*, 2024a. doi: 10.48550/ARXIV.2402.03559.

Jacob Christopher, Michael Cardei, Jinhao Liang, and Ferdinando Fioretto. Neuro-symbolic generative diffusion models for physically grounded, robust, and safe generation. 2025.

Jacob K Christopher, Stephen Baek, and Ferdinando Fioretto. Constrained synthesis with projected diffusion models, 2024b.

Jacob K. Christopher, Austin Seamann, Jingyi Cui, Sagar Khare, and Ferdinando Fioretto. Constrained diffusion for protein design with hard structural constraints, 2026.

Hyungjin Chung, Jeongsol Kim, Michael Thompson McCann, Marc Louis Klasky, and Jong Chul Ye. Diffusion Posterior Sampling for General Noisy Inverse Problems. In *ICLR*. OpenReview.net, 2023.

Gabriele Corso, Yilun Xu, Valentin de Bortoli, Regina Barzilay, and Tommi Jaakkola. Particle guidance: non-i.i.d. diverse sampling with diffusion models. *arXiv*, 2023.

Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *CVPR*, pp. 4690–4699. Computer Vision Foundation / IEEE, 2019.

Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis, 2021.

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

Alexandre Galashov, Ashwini Pokle, Arnaud Doucet, Arthur Gretton, Mauricio Delbracio, and Valentin De Bortoli. Learn to guide your diffusion model, 2025.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020.

Emiel Hoogeboom, Victor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant diffusion for molecule generation in 3d, 2022.

Jiahe Huang, Guandao Yang, Zichen Wang, and Jeong Joon Park. Diffusionpde: Generative pde-solving under partial observation, 2024a.

William Huang, Yifeng Jiang, Tom Van Wouwe, and C. Karen Liu. Constrained diffusion with trust sampling, 2024b.

Vivek Jayaram, Ira Kemelmacher-Shlizerman, Steven M. Seitz, and John Thickstun. Linearly constrained diffusion implicit models, 2025.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models, 2022.

Bahjat Kawar, Michael Elad, Stefano Ermon, and Jiaming Song. Denoising diffusion restoration models. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *NeurIPS*, 2022.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013.

Felix Koulischer, Florian Handke, Johannes Deleu, Thomas Demeester, and Luca Ambrogioni. Feedback guidance of diffusion models, 2025.

Ravi Kumar, Manish Purohit, Zoya Svitkina, Erik Vee, and Joshua R. Wang. Efficient rematerialization for deep networks. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché Buc, Emily B. Fox, and Roman Garnett (eds.), *NeurIPS*, pp. 15146–15155, 2019.

Tuomas Kynkäänniemi, Miika Aittala, Tero Karras, Samuli Laine, Timo Aila, and Jaakko Lehtinen. Applying guidance in a limited interval improves sample and distribution quality in diffusion models, 2024.

Hans Petter Langtangen and Kent-Andre Mardal. *Introduction to Numerical Methods for Variational Problems*. Springer International Publishing, 2019. ISBN 9783030237882. doi: 10.1007/978-3-030-23788-2. URL http://dx.doi.org/10.1007/978-3-030-23788-2.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations, 2020.

Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models, 2022.

Dawid Malarz, Artur Kasymov, Maciej Zieba, Jacek Tabor, and Przemyslaw Spurek. Classifier-free guidance with adaptive scaling, 2025.

Francois Maze and Faez Ahmed. Diffusion models beat gans on topology optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

Alexander Quinn Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (eds.), *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pp. 16784–16804. PMLR, 2022.

Pinelopi Papalampidi, Olivia Wiles, Ira Ktena, Aleksandar Shtedritski, Emanuele Bugliarello, Ivana Kajic, Isabela Albuquerque, and Aida Nematzadeh. Dynamic classifier-free diffusion guidance via online feedback, 2025.

Krunoslav Lehman Pavasovic, Jakob Verbeek, Giulio Biroli, and Marc Mezard. Classifier-free guidance: From high-dimensional analysis to generalized guidance forms, 2025.

M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, pp. 10674–10685. IEEE, 2022.

Seyedmorteza Sadat, Jakob Buhmann, Derek Bradley, Otmar Hilliges, and Romann M. Weber. Cads: Unleashing the diversity of diffusion models through condition-annealed sampling, 2024.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations, 2021.

Ruitong Sun, Tianze Yang, Wei Niu, and Jin Sun. Ousac: Optimized guidance scheduling with adaptive caching for dit acceleration, 2025.

Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park, and Victor Lempitsky. Resolution-robust large mask inpainting with fourier convolutions. In *WACV*, pp. 3172–3182. IEEE, 2022.

Zhiwei Tang, Jiangweizhi Peng, Jiasheng Tang, Mingyi Hong, Fan Wang, and Tsung-Hui Chang. Inference-time alignment of diffusion models with direct noise optimization, 2024.

Vinh Tong, Hoang Trung-Dung, Anji Liu, Guy Van den Broeck, and Mathias Niepert. Learning to discretize denoising diffusion odes, 2025.

Lloyd N. Trefethen. *Finite Difference and Spectral Methods for Ordinary and Partial Differential Equations*. 1996. URL `http://people.maths.ox.ac.uk/trefethen/pdetext.html`. Unpublished text.

Xi Wang, Nicolas Dufour, Nefeli Andreou, Marie-Paule Cani, Victoria Fernandez Abrevaya, David Picard, and Vicky Kalogeiton. Analysis of classifier-free guidance weight schedulers, 2024.

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

Daniel Watson, William Chan, Jonathan Ho, and Mohammad Norouzi. Learning fast samplers for diffusion models by differentiating through sample quality, 2022.

Joseph L. Watson, David Juergens, Nathaniel R. Bennett, Brian L. Trippe, Jason Yim, Helen E. Eisenach, Woody Ahern, Andrew J. Borst, Robert J. Ragotte, Lukas F. Milles, Basile I. M. Wicky, Nikita Hanikel, Samuel J. Pellock, Alexis Courbet, William Sheffler, Jue Wang, Preetham Venkatesh, Isaac Sappington, Susana Vázquez Torres, Anna Lauko, Valentin De Bortoli, Emile Mathieu, Sergey Ovchinnikov, Regina Barzilay, Tommi S. Jaakkola, Frank DiMaio, Minkyung Baek, and David Baker. De novo design of protein structure and function with rfdiffusion. *Nature*, 620(7976):1089–1100, 2023.

Luhuan Wu, Brian L Trippe, Christian Naesseth, David Blei, and John P Cunningham. Practical and asymptotically exact conditional sampling in diffusion models. *Advances in Neural Information Processing Systems*, 2023.

Minkai Xu, Lantao Yu, Yang Song, Chence Shi, Stefano Ermon, and Jian Tang. Geodiff: A geometric diffusion model for molecular conformation generation. In *ICLR*, 2022.

Lingxiao Yang, Shutong Ding, Yifan Cai, Jingyi Yu, Jingya Wang, and Ye Shi. Guidance with Spherical Gaussian Constraint for Conditional Diffusion. In *ICML*, volume 235 of *Proceedings of Machine Learning Research*, pp. 56071–56095, 2024.

Shai Yehezkel, Omer Dahary, Andrey Voynov, and Daniel Cohen-Or. Navigating with annealing guidance scale in diffusion space. In *Proceedings of the SIGGRAPH Asia 2025 Conference Papers*, SA Conference Papers '25, pp. 1–11. ACM, December 2025. doi: 10.1145/3757377.3763830.

Jiwen Yu, Yinhuai Wang, Chen Zhao, Bernard Ghanem, and Jian Zhang. Freedom: Training-free energy-guided conditional diffusion model. In *ICCV*. IEEE, 2023.

Stefano Zampini, Jacob K. Christopher, Luca Oneto, Davide Anguita, and Ferdinando Fioretto. Training-free constrained generation with stable diffusion models, 2025.

Kevin Zhai, Utsav Singh, Anirudh Thatipelli, Souradip Chakraborty, Anit Kumar Sahu, Furong Huang, Amrit Singh Bedi, and Mubarak Shah. Mira: Towards mitigating reward hacking in inference-time alignment of t2i diffusion models. *CoRR*, October 2025.

Bingliang Zhang, Wenda Chu, Julius Berner, Chenlin Meng, Anima Anandkumar, and Yang Song. Improving Diffusion Inverse Problem Solving with Decoupled Noise Annealing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20895–20905, 2025.

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

Shangwen Zhu, Qianyu Peng, Zhilei Shu, Yuting Hu, Zhantao Yang, Han Zhang, Zhao Pu, Andy Zheng, Xinyu Cui, Jian Zhao, Ruili Feng, and Fan Cheng. Mambo-g: Magnitude-aware mitigation for boosted guidance, 2026.

LEARNED END-TO-END GUIDANCE SCHEDULES FOR DIFFUSION MODELS

ADDITIONAL MATERIAL

A Pseudocode for LEEGS training **16**

B Additional experimental details **18**

B.1 Noisy inverse problems 18

B.2 Image inpainting 19

B.3 Face-ID guidance 20

B.4 PDE problems 21

B.5 Computational resources 22

C Further experiments and ablations **23**

C.1 Ablation on training dataset size 23

C.2 Comparison of LEEGS vs true gradient 24

C.3 Effect of regularization 25

C.4 Generalization across tasks 26

C.5 Generalization across number of steps 27

C.6 Ablation on trajectory objective vs terminal objective 29

D Extension to sampling with renoising **31**

E Comparison at $S = 10$ NFEs (FFHQ) **33**

F Additional metrics for main experiments **37**

F.1 Variance of LEEGS across draws 37

F.2 Noisy inverse problems 37

F.3 Image inpainting problems 40

G Additional learned guidance schedules **41**

G.1 Noisy inverse problems 41

G.2 Image inpainting 42

G.3 Face-ID guidance 43

G.4 PDE problems 44

H Hyperparameter searches for the baselines **46**

H.1 Noisy inverse problems 46

H.2 Image inpainting 50

H.3 Face-ID guidance 51
H.4 PDE problems 52

I Qualitative results for main experiments (Supplementary material) 54

A Pseudocode for LEEGS training

Algorithm 1 LEEGS training algorithm

Input: training conditions $\mathcal{D} = \{c^{(i)}\}_{i=1}^{N}$, steps S, epochs E, flag `compute_leegs_approx`
Output: guidance schedule W
$W \leftarrow (w_1, \ldots, w_S)$ with $w_s \sim \mathcal{U}[0,1]$ i.i.d. ▷ schedule initialization
for epoch $= 1, \ldots, E$ **do**
  for minibatch $\mathcal{B} \subseteq \mathcal{D}$ **do**
    $\overline{g} \leftarrow \frac{1}{|\mathcal{B}|} \sum_{c \in \mathcal{B}}$ ComputeGradient$(W,\ c,$ `compute_leegs_approx`$)$
    $W \leftarrow$ AdamStep$(W,\ \overline{g})$
    $W \leftarrow \max(W,\ 0)$ ▷ project onto $\mathbb{R}^S_{\geq 0}$
  end for
end for
return W

function ComputeGradient$(W, c,$ `compute_leegs_approx`$)$
  $x \sim \mathcal{N}(0, \mathbf{I})$
  $\ell \leftarrow 0$
  for $s = S, \ldots, 1$ **do**
    $t \leftarrow t_s$ ▷ diffusion time at step s
    $(x, h_s) \leftarrow$ Checkpoint$\big($GuidedStep$,\ x,\ t,\ w_s,\ c,$ `compute_leegs_approx`$\big)$
    $\ell \leftarrow \ell + h_s$ ▷ add intermediate term $h(\hat{x}_{0|s}, c)$
  end for
  $h_0 \leftarrow$ `compute_guidance_function`(x, c) ▷ terminal term $h(x_0, c)$
  $\ell \leftarrow \ell + h_0$
  return Backward$(\ell,\ W)$ ▷ populate leaf nodes of autodiff graph with gradients
end function

function GuidedStep$(x, t, w, c,$ `compute_leegs_approx`$)$
  $($`denoised_x`, `next_x`$) \leftarrow$ `sampler_step`(x, t)
  $h \leftarrow$ `compute_guidance_function`$($`denoised_x`$, c)$
  $g \leftarrow$ Grad$\big(h,\ x;$ `create_graph` $=$ **not** `compute_leegs_approx`$\big)$
  if `compute_leegs_approx` **then**
    $g \leftarrow g$`.detach()` ▷ drop $\partial g/\partial x$; keep $\partial \Phi/\partial x$
  end if
  `next_x` $\leftarrow$ `next_x` $- w\, g$ ▷ guided update, (2)
  return $($`next_x`$,\ h)$
end function

Algorithm 1 gives pseudocode for training a guidance schedule with LEEGS. The outer loop performs Adam updates over epochs and minibatches of training conditions, projecting W onto $\mathbb{R}^S_{\geq 0}$ after every step. ComputeGradient evaluates the inner objective over a freshly sampled trajectory and returns its gradient with respect to W, using gradient checkpointing to keep the peak memory independent of S. GuidedStep applies one guided reverse-diffusion step (2) and returns the updated iterate together with that step's guidance value h_s. We implement this in PyTorch using the built-in checkpointing functionality (`torch.utils.checkpoint`).

Primitives. We assume the following building blocks, each acting on a single sampling step:

- `sampler_step`(x, t) – returns the pair $($`denoised_x`, `next_x`$)$, where `denoised_x` $= \hat{x}_0(x, t; \epsilon_\theta)$ is the denoiser prediction ($\hat{x}_{0|s}$ when $x = x_s$ and $t = t_s$) and `next_x` $= \Phi(x, t; \epsilon_\theta)$ is the unguided reverse-diffusion step of (1). Recall that the (DDIM) forward noising process relates a clean sample to its noised iterate via $x_s = \sqrt{\overline{\alpha}_s}\, x_0 + \sqrt{1 - \overline{\alpha}_s}\, \epsilon$ for $\epsilon \sim \mathcal{N}(0, \mathbf{I})$, where $\overline{\alpha}_s = \prod_{i=1}^{s} \alpha_i$ is the cumulative product of the (per-step) variance-schedule coefficients α_i; accordingly $\sqrt{\overline{\alpha}_s}$ and $\sqrt{1 - \overline{\alpha}_s}$ are the signal and noise coefficients at time t_s. When the network is parameterized as a noise predictor ϵ_θ, the denoiser prediction is recovered from it via Tweedie's formula (Efron,

2011),

$$\hat{x}_0(x_s, t_s; \epsilon_\theta) = \frac{x_s - \sqrt{1-\overline{\alpha}_s}\,\epsilon_\theta(x_s, t_s)}{\sqrt{\overline{\alpha}_s}}. \tag{6}$$

Many diffusion models instead predict the denoised sample $\hat{x}_0$ directly, in which case `denoised_x` is simply read off the network output and (6) is unnecessary.

- `compute_guidance_function`(x, c) – evaluates the guidance function $h(x, c)$ on a (denoised) sample.
- Autodiff primitives, which map directly onto their PyTorch counterparts:
  - GRAD$(y, x;$ `create_graph`$)$ – the gradient $\nabla_x y$ of a scalar y w.r.t. a tensor x (`torch.autograd.grad`). The flag `create_graph` retains the graph of the gradient itself, so that its Jacobian can be differentiated in a later backward pass.
  - BACKWARD(y, x) – the reverse-mode gradient $\nabla_x y$ accumulated through the retained graph (`y.backward()` followed by reading `x.grad`).
  - x`.detach()` – returns x detached from the autodiff graph (`torch.Tensor.detach`).
  - CHECKPOINT$(f, \ldots)$ – evaluates f without storing its internal activations, recomputing them during the backward pass (`torch.utils.checkpoint.checkpoint`).

In GUIDEDSTEP, the flag `compute_leegs_approx` controls whether the guidance gradient g_s retains its graph: with `compute_leegs_approx` $=$ `false` (`full`) nothing is detached and the guidance Hessian $\partial g_s / \partial x_s$ in (5) contributes to the gradient; with `compute_leegs_approx` $=$ `true` (LEEGS) g_s is treated as a constant of x_s, so only the sampler Jacobian $\partial \Phi / \partial x_s$ is propagated across steps. The two settings differ only in the `.detach()` on line 28.

B Additional experimental details

Unless noted otherwise, LEEGS is trained with Adam and batch size 1, separately for each subtask and NFE budget, on 30 examples randomly drawn from a training pool that is disjoint from the test set. The number of epochs is chosen from the convergence of the training objective. Our implementation builds on the codebases of Chung et al. (2023); Huang et al. (2024a); Yang et al. (2024).

B.1 Noisy inverse problems

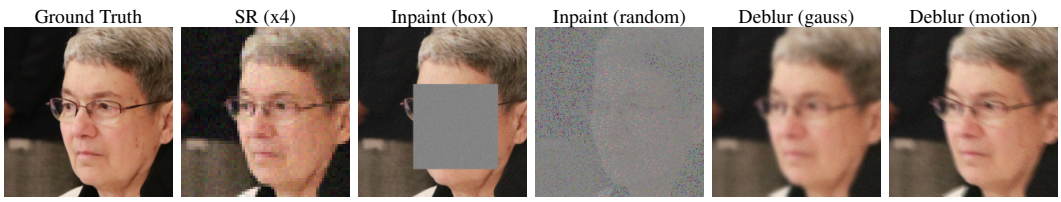


Figure 4: Ground truth and the 5 noisy measurement operators used for Tables 1 and 2 (FFHQ example).

Models. Following Chung et al. (2023), we use the pretrained FFHQ model of Chung et al. (2023) and the pretrained ImageNet model of Dhariwal & Nichol (2021).

Dataset generation. We follow the implementation and settings of Chung et al. (2023) for both FFHQ and ImageNet, which we recap for completeness; Figure 4 illustrates the five measurement operators. For superresolution, the low-resolution images are generated by bicubic downsampling at $4\times$ scale. For Gaussian deblurring, the images are convolved with a 61×61 Gaussian kernel with standard deviation 3.0. For motion deblurring, the images are convolved with a 61×61 motion-blur kernel of intensity 0.5, generated with the implementation from `https://github.com/LeviBorodenko/motionblur`. For box and random inpainting, the masks are sampled randomly for each image, where for box inpainting the mask is an approximately centered 128×128 square, and for random inpainting the mask is a random binary mask with each pixel masked with a probability of 0.92.

For FFHQ, we use the test set of 1000 images of Chung et al. (2023) for training and validation. For the other datasets, we set aside a disjoint pool of 100 images (90 for training and 10 for validation metrics during training), but for FFHQ we use a cross-validation-style split of the 1000 images instead, to maintain a reasonable test set size : the first 50 images are set aside, of which the first 30 are used to train LEEGS and the remaining 20 for validation, and the other 950 images form the test set of Table 1. For ImageNet, we build our own training and test sets. The test set of Table 2 consists of 1000 images sampled from the ILSVRC 2012 validation set, one per class, and a disjoint pool of 100 images is split into 90 images for training, from which each LEEGS run draws 30, and 10 for validation.

Baselines. Unless stated otherwise, the hyperparameters of each baseline are selected by LPIPS on 20 validation images, and the search results are given in Appendix H. For the guidance-scale searches, the grid is expanded whenever the optimum lies at its boundary.

- **Constant guidance** applies a fixed guidance scale w at every step. We select w by a grid search (Figures 29 and 30).
- **Linear increase** ramps the guidance scale linearly from 0 at the first step to w at the final step, and **linear decrease** ramps it from w down to 0. We select w by a grid search, as for constant guidance (Figures 31–34).
- **DSG** (Yang et al., 2024): we use the authors' repository (`https://github.com/LingxiaoYang2023/DSG2024`), apply DSG guidance at every step (interval 1), and select the guidance scale by a grid search, leaving all other hyperparameters at their defaults (Figures 35 and 36).

- **CDIM** (Jayaram et al., 2025): we use the authors' repository (`https://github.com/vivjay30/cdim`) with its default configurations at $T = 25$ and $T = 50$ for the budgets of 50 and 100 NFEs, respectively; these values of T were chosen because they are the authors' own defaults, for which good hyperparameters (e.g. the step size λ) were already established. CDIM's number of NFEs is not fixed but varies per sample, and at these settings its average is consistently higher than 50 and 100, respectively, across every task and dataset (Table 6), except random inpainting at $T = 50$.
- **DAPS** (Zhang et al., 2025) alternates denoising with renoising, and its number of NFEs is the product of the number of outer annealing steps A and inner diffusion (ODE) steps D. Using the authors' repository (`https://github.com/zhangbingliang2019/DAPS`), we search over different splits of the budget: $(A, D) \in \{(10, 5), (25, 2), (50, 1)\}$ for 50 NFEs, and $(A, D) \in \{(10, 10), (20, 5), (50, 2), (100, 1)\}$ for 100 NFEs (Figures 37–40).
- **DPS** (Chung et al., 2023): the 1000-NFE results are taken from Chung et al. (2023).

Subtask	Avg. NFEs ($T = 25$)	Avg. NFEs ($T = 50$)
FFHQ		
Box inpainting	55.7	102.8
Random inpainting	50.7	93.7
Super-resolution ($4\times$)	54.5	104.7
Gaussian deblur	56.0	104.4
Motion deblur	65.8	114.2
ImageNet		
Box inpainting	73.4	123.0
Random inpainting	53.7	95.0
Super-resolution ($4\times$)	62.1	112.6
Gaussian deblur	66.1	113.7
Motion deblur	80.6	128.6

Table 6: Average number of NFEs used by CDIM per subtask at its two settings, $T = 25$ and $T = 50$ (nominally 50 and 100 NFEs), computed over the full test set of Tables 1 and 2.

Training configuration. Table 7 gives the learning rate, number of epochs and weight decay used to train LEEGS for each subtask.

Subtask	Learning rate η	Epochs	Weight decay
FFHQ			
Super-resolution ($4\times$)	0.3	25 ($S = 50$) / 15 ($S = 100$)	1e-4 ($S = 50$) / 5e-4 ($S = 100$)
Box inpainting	0.2	25 ($S = 50$) / 25 ($S = 100$)	0.0
Random inpainting	0.8	25 ($S = 50$) / 20 ($S = 100$)	0.0
Gaussian deblur	0.5	25 ($S = 50$) / 25 ($S = 100$)	0.0
Motion deblur	0.3	30 ($S = 50$) / 25 ($S = 100$)	0.0
ImageNet			
Super-resolution ($4\times$)	0.3	20 ($S = 50$) / 15 ($S = 100$)	0.0 ($S = 50$) / 5e-4 ($S = 100$)
Box inpainting	0.3	20 ($S = 50$) / 15 ($S = 100$)	0.0
Random inpainting	0.8	20 ($S = 50$) / 15 ($S = 100$)	0.0
Gaussian deblur	0.2	20 ($S = 50$) / 15 ($S = 100$)	0.0
Motion deblur	0.3	20 ($S = 50$) / 15 ($S = 100$)	0.0

Table 7: LEEGS training configuration for the noisy inverse problems (Tables 1, 2).

B.2 Image inpainting

Models. Following Lugmayr et al. (2022), we use the pretrained CelebA-HQ model of Lugmayr et al. (2022) and the pretrained ImageNet model of Dhariwal & Nichol (2021).



Figure 5: Ground truth and the 5 noiseless inpainting mask operators used for Table 3 (CelebA-HQ example).

Dataset generation. For CelebA-HQ, we use the validation split (`val_source_256`) produced by the preprocessing pipeline of Suvorov et al. (2022), which is also used by Lugmayr et al. (2022). The test set of Table 3 consists of 100 images from this split, and a disjoint pool of 100 images from the same split is divided into 90 images for training, from which each LEEGS run draws 30, and 10 for validation. For ImageNet, the test set consists of 100 images subsampled from the test set of the noisy inverse problems (Section B.1), and another disjoint pool of 100 images is split in the same way into 90 images for training and 10 for validation. We use the same set of masks provided by Lugmayr et al. (2022) for both training and testing (Figure 5). All mask types except for the wide and narrow masks are fixed across images.

Baselines. We run RePaint using the authors' repository (`https://github.com/andreas128/RePaint`). RePaint has three hyperparameters: the number of sampling steps S, the jump length for renoising J, and the number of times N the jump is taken, resulting in approximately $S \cdot N$ NFEs. We try five configurations that yield approximately 200 NFEs (201 in all cases): ($S = 200, N = 1, J = 1$), ($S = 101, N = 2, J = 5$), ($S = 101, N = 2, J = 10$), ($S = 51, N = 4, J = 10$) and ($S = 51, N = 4, J = 5$), and select the best per task by LPIPS on 20 validation images. The results of the hyperparameter search are shown in Figures 45 and 46 in Appendix H.

Training configuration. Table 8 gives the learning rate, number of epochs and weight decay used to train LEEGS for each mask.

Mask	Learning rate η	Epochs	Weight decay
ImageNet			
Wide	0.2	5	0.0
Narrow	0.2	5	0.0
Super-resolve (2×)	0.2	5	0.0
Alternating lines	0.2	5	0.0
Half	0.2	5	0.0
CelebA-HQ			
Wide	0.2	5	0.0
Narrow	0.2	5	0.0
Super-resolve (2×)	0.2	5	0.0
Alternating lines	0.2	5	0.0
Half	0.2	5	0.0

Table 8: LEEGS training configuration for image inpainting (Table 3).

B.3 Face-ID guidance

Models. We use the unconditional CelebA-HQ 256×256 model of Lugmayr et al. (2022), as for image inpainting. Identity is measured with an ArcFace (IR-SE-50) face-recognition network (Deng et al., 2019).

Dataset generation. As for image inpainting (Section B.2), the images are drawn from the CelebA-HQ `val_source_256` split, with a test set of 100 images and a disjoint training pool from which

each LEEGS run draws 30 images. The identity target for each image is the image itself. FID is computed against the 26,000 images of the CelebA-HQ `train_256` split.

Baselines. For constant guidance, we select the guidance scale for each NFE budget by a grid search on 20 validation images, minimizing the mean FaceID score. For DSG (Yang et al., 2024), we tune the guidance scale in the same way, and use our own implementation for sampling. We set its interval to 1, so that DSG guidance is applied at every step.

Training configuration. Table 9 gives the learning rate, number of epochs and weight decay used to train LEEGS for each NFE budget.

NFEs	Learning rate η	Epochs	Weight decay
50	0.1	15	0.0
100	0.1	10	0.0

Table 9: LEEGS training configuration for Face-ID guidance (Table 4).

B.4 PDE problems

Background. We consider static boundary-value problems (BVPs)

$$\begin{aligned} \mathcal{F}(z; a, u) = 0, \quad z \in \Omega \subseteq \mathbb{R}^d, \\ u(z) = g(z), \quad z \in \partial\Omega, \end{aligned} \tag{7}$$

where $\mathcal{F}$ is a (possibly non-linear) differential operator, Ω is a bounded spatial domain with boundary $\partial\Omega$, g specifies the boundary condition, $a(\cdot)$ is a spatially varying coefficient, and $u(\cdot)$ is the solution. The *forward problem* recovers u from a, and the *inverse problem* recovers a from u. Following DiffusionPDE (Huang et al., 2024a), a pretrained diffusion model represents the joint distribution over pairs (a, u) rendered as 128×128 images, and guidance enforces consistency with sparse observations: for the forward problem, a is observed at 500 randomly chosen locations z_{obs} ($\approx 3\%$ of pixels), and analogously u for the inverse problem. Following Huang et al. (2024a), we consider the three problems

$$\begin{aligned} \text{Poisson:} \quad & -\nabla^2 u(z) = a(z), \\ \text{Helmholtz:} \quad & \nabla^2 u(z) + k^2 u(z) = a(z), \\ \text{Darcy flow:} \quad & -\nabla \cdot (a(z)\nabla u(z)) = 1, \end{aligned}$$

each on a bounded domain Ω with $u(z) = 0$ on $\partial\Omega$.

Models. We use the pretrained models of Huang et al. (2024a) for all three problems.

Dataset generation. We use the test set provided by Huang et al. (2024a) (1024 samples per problem). We set aside 100 samples, split into 90 for training, from which each LEEGS run draws 30, and 10 for validation. We form our test set from the remaining samples (1000 for Helmholtz and Darcy, and 924 for Poisson). The 500 observed locations are sampled randomly for each example.

Baselines. As for the noisy inverse problems (Section B.1), the constant-guidance and linear-schedule baselines are each defined by a single guidance scale, which we select by a grid search on 20 validation samples, using the relative error or the error rate as appropriate. The resulting large guidance scales are consistent with those used by Huang et al. (2024a). The search curves are given in Figures 49–51 and 52–57 in Appendix H. Following Huang et al. (2024a), the constant-guidance baseline uses a second-order sampler, and the DiffusionPDE results are those reported by Huang et al. (2024a).

Training configuration. Table 10 gives the learning rate, number of epochs and weight decay used to train LEEGS for each problem.

Problem	Learning rate η	Epochs	Weight decay
Poisson, forward	15.0	15 ($S = 50$) / 15 ($S = 100$)	0.0
Poisson, inverse	500.0	15 ($S = 50$) / 15 ($S = 100$)	0.0
Helmholtz, forward	15.0 ($S = 50$) / 10.0 ($S = 100$)	15 ($S = 50$) / 15 ($S = 100$)	0.0
Helmholtz, inverse	1500.0	15 ($S = 50$) / 10 ($S = 100$)	0.0
Darcy, forward	30.0	15 ($S = 50$) / 15 ($S = 100$)	0.0
Darcy, inverse	150000.0 ($S = 50$) / 100000.0 ($S = 100$)	15 ($S = 50$) / 15 ($S = 100$)	0.0

Table 10: LEEGS training configuration for the PDE problems (Table 5).

B.5 Computational resources

Our experiments were run on an internal cluster with NVIDIA A40 and A100 GPUs and an internal workstation with an NVIDIA RTX 3090 GPU. Table 11 gives per-task training wall-clock times, measured on a single A100 with no validation logging (validation is used only for monitoring, not for early stopping or model selection, so it is not part of the reported training cost). Since different tasks within the same results table can use different epoch counts at a fixed NFE budget S (Tables 7, 8, 10), we report the minimum and maximum times spent at each S whenever they differ.

Results table	S	Epochs	Wall-clock	Time/epoch
FFHQ (Table 1)	50	25 – 30	1h26m – 1h57m	3.4 – 3.9 min
	100	15 – 25	1h52m – 3h06m	7.5 min
ImageNet (Table 2)	50	20	3h11m	9.6 min
	100	15	4h44m	19.0 min
Inpainting (Table 3)	200 (CelebA-HQ)	5	3h13m	38.7 min
	200 (ImageNet)	5	3h14m	38.7 min
PDE problems (Table 5)	50x1	15	1h08m	4.5 min
	100x1	10 – 15	1h37m – 2h21m	9.4 – 9.7 min
Face-ID (Table 4)	50	15	2h34m	10.3 min
	100	10	3h27m	20.8 min

Table 11: Training wall-clock time by results table and NFE budget S, on a single A100 GPU with no validation logging. Where tasks at a fixed S use different epoch counts, both the minimum and maximum are reported (as a range); a single value indicates every task at that S trains for the same number of epochs.

C Further experiments and ablations

C.1 Ablation on training dataset size

This section analyzes the variance in the learned guidance schedule due to the choice of training set. We consider two tasks : the noisy motion deblur on FFHQ, and the Helmholtz forward PDE problem, both at $S = 10$. We train the guidance schedule on datasets of varying size n, where each dataset is drawn randomly from a larger pool of 500 samples. For each n we draw 10 datasets and train a separate guidance schedule on each, holding the total number of training iterations fixed as n varies. For both tasks, we sweep $n \in \{1, 5, 30, 50, 100\}$.

We evaluate each learned schedule on a held-out test set of 300 samples, reporting the reconstruction quality [LPIPS for FFHQ, relative L_2 error with respect to u (the solution) for Helmholtz forward; see Figure 6] and the per-draw distance of the learned schedule from the draw-averaged schedule (see Figure 7). The latter metric directly measures the variability of the schedule, while the former measures its effect on overall performance. We also plot the mean of the learned schedules across the 10 draws for different n in Figure 8.

We see that the variance with respect to both metrics decreases with increasing n. While the evaluation metrics plateau at larger n for both tasks, the variability of the learned schedules continues to decrease for the Helmholtz forward problem. This could be due to the use of a deterministic sampler for the PDE problem, which reduces the variance of the learned schedule. In contrast, the FFHQ motion deblur task uses a stochastic sampler, which introduces additional variability in the learned schedule.

Figure 8 shows that the *mean schedule* is relatively stable across n, especially for the FFHQ motion deblur task. Note that this is not a measure of variance, but the trend in the average schedule learned across the 10 draws for each n. This observation seems to imply that training with fewer samples has more variance and less bias, with respect to the mean schedule learned with larger datasets.

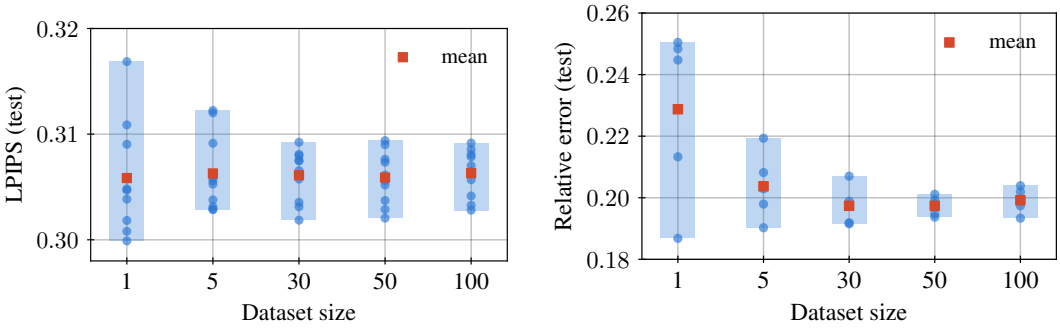


Figure 6: Held-out test-set reconstruction quality as a function of training dataset size n. Scatter points are the 10 individual dataset draws, and the vertical bar spans min to max, with the mean indicated by a red square. *Left:* FFHQ motion deblur. *Right:* Helmholtz forward.

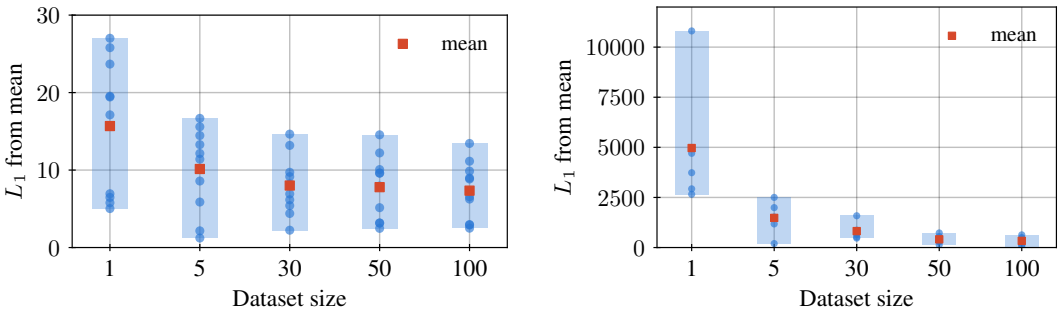


Figure 7: Per-draw L_1 distance of the learned guidance schedule from the draw-averaged schedule (all $S = 10$ steps), as a function of training dataset size n. *Left:* FFHQ motion deblur. *Right:* Helmholtz forward.

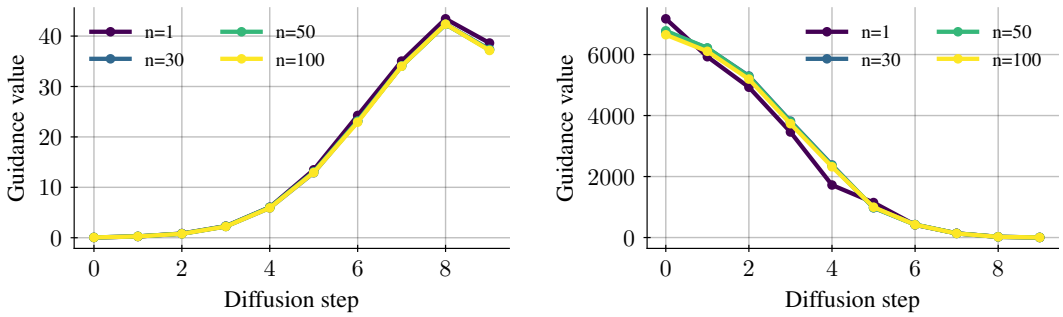


Figure 8: Draw-averaged guidance schedule (mean over 10 dataset draws) as a function of diffusion step, compared across training dataset sizes n *Left:* FFHQ motion deblur. *Right:* Helmholtz forward.

C.2 Comparison of LEEGS vs true gradient

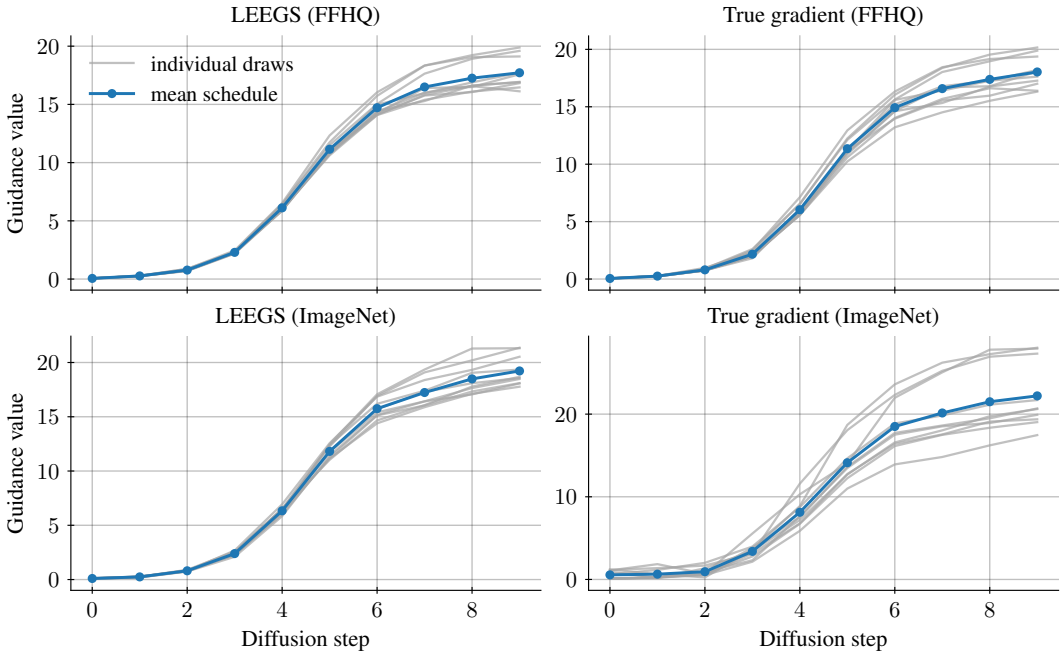


Figure 9: Learned guidance schedules for LEEGS and the true gradient (FFHQ and ImageNet motion deblur, $S = 10$). Each panel shows the 10 per-draw schedules (thin grey) and their mean (bold); the true gradient's higher spread on ImageNet is visible in the wider band around the mean, especially at later diffusion steps.

Schedule comparison. We compare the LEEGS gradient (guidance term detached before backpropagation) against the true gradient (nothing detached) on the noisy motion deblur task (observation noise $\sigma = 0.05$), on FFHQ and ImageNet datasets at $S = 10$ with the DDPM sampler. For each dataset (FFHQ/ImageNet) and method (true gradient/LEEGS) we train 10 runs while varying the dataset of training samples, where each run uses 50 random images resampled from a 500-image pool. Each run is trained for 8 epochs with a learning rate of 0.08.

We evaluate each draw on 300 held-out test images. We report LPIPS against the clean source image, and L_1/L_2 distance of the draw's schedule from the 10-draw mean schedule in Table 12 and plot the learned schedules in Figure 9. On FFHQ, the two methods are comparable, while on ImageNet the true gradient is both worse and far less stable than LEEGS (std 4–6$\times$ higher on LPIPS, distances $\sim$3$\times$ higher). The true gradient's instability is visible in the wider band around the mean schedule in Figure 9, especially at later diffusion steps.

Dataset	Mode	LPIPS (↓)	L_2 dist. from mean	L_1 dist. from mean
FFHQ	LEEGS	0.341 ± 0.004	2.05	4.50
	True gradient	0.342 ± 0.006	2.34	5.52
ImageNet	LEEGS	0.560 ± 0.007	2.28	5.13
	True gradient	0.594 ± 0.041	7.03	17.89

Table 12: LEEGS vs. the true gradient on motion deblur ($S = 10$; mean $\pm$ std over 10 draws). L_1/L_2 dist. from mean is each draw's schedule distance to the 10-draw mean. On ImageNet the true gradient is both worse and far less stable than LEEGS (std 4–6$\times$ higher on LPIPS, distances $\sim 3\times$ higher); on FFHQ the two are comparable.

Timing comparison. We measure the wall-clock cost of one training iteration (batch size 1) as a function of the inference budget $S \in \{5, 10, 20, 40, 80, 100\}$. We compare LEEGS, the true gradient, and include plain sampling (no guidance-schedule optimization) as a reference. We report the mean over 10 iterations, on FFHQ box inpainting (image domain) and Helmholtz forward (PDE domain), measured on a single NVIDIA RTX 5090 GPU. We see that the LEEGS approximation is $\sim 4 - 5\times$ faster than the true gradient, while the previous section shows that the results are comparable on FFHQ, and the true gradient is less stable on Imagenet.

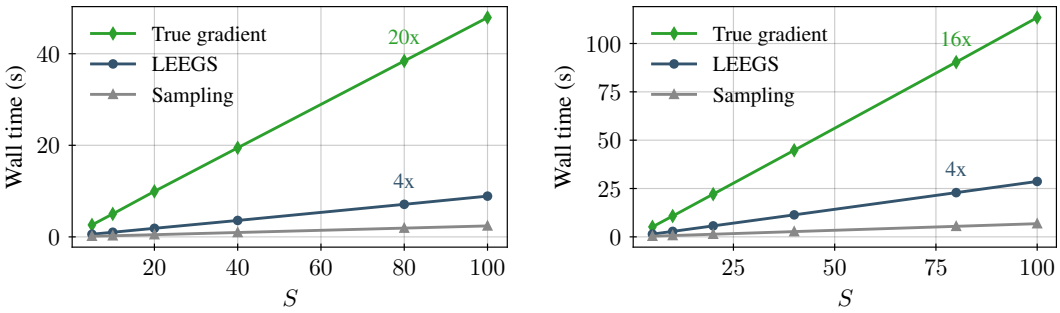


Figure 10: Wall-clock cost per training iteration vs. inference budget S, for LEEGS and the true gradient, relative to plain sampling. *Left:* FFHQ box inpainting – true gradient $\approx 20\times$ sampling, LEEGS $\approx 4\times$. *Right:* Helmholtz forward – true gradient $\approx 16\times$, LEEGS $\approx 4\times$.

C.3 Effect of regularization

This section studies the effect of weight decay on the end-to-end LEEGS guidance schedule, and its interaction with the number of diffusion steps S. We consider two FFHQ inverse tasks that behave differently under prolonged training: *superresolution*, where we observe an overfitting elbow in which the terminal constraint keeps improving while held-out reconstruction quality worsens, and *box inpainting*, where no such elbow appears. For each task we train the guidance schedule with the Adam optimizer's weight decay (wdecay) applied to the guidance-schedule parameters, sweeping wdecay $\in \{0, 10^{-5}, 10^{-4}, 5 \times 10^{-4}\, 10^{-3}\}$ across $S \in \{10, 25, 50, 100\}$ diffusion steps, for a total of 20 configurations per task. We train using a fixed set of 30 samples for all runs. The learning rate and number of training epochs are fixed per task and S. For superresolution, we use a learning rate of 0.3 and train for 30 epochs at $S \in \{10, 25\}$, 25 epochs at $S = 50$, and 15 epochs at $S = 100$. For the box inpainting task we train for 20 epochs with a learning rate of 0.3 for $S = 10$, 25 epochs with a learning rate of 0.3 for $S = 25$, and 25 epochs with a learning rate of 0.2 for $S \in \{50, 100\}$.

We evaluate each learned schedule on the held-out test set of 300 samples, reporting LPIPS between the reconstruction and the ground-truth image. Figure 2 in the main text shows the corresponding *training* LPIPS at $S = 100$: without weight decay, training exhibits a sharp reward-hacking elbow, past which the terminal constraint keeps improving while the reconstruction degrades. Increasing weight decay progressively flattens this elbow, and at $\gamma = 5 \times 10^{-4}$ it disappears entirely. The resulting training LPIPS, however, never reaches the value attained just before the unregularized elbow, so early stopping at that point is another fix for this failure mode.

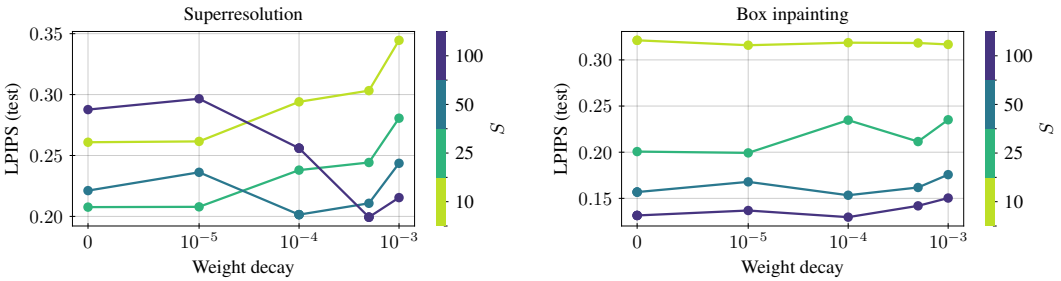


Figure 11: Held-out LPIPS as a function of weight decay γ, one curve per S (discrete colorbar). *Left:* superresolution. *Right:* box inpainting.

Figure 11 (left) shows the analogous picture for superresolution with respect to the LPIPS on the held-out 300 sample test set. At $S = 100$, held-out LPIPS improves with weight decay up to $\gamma = 5 \times 10^{-4}$, then turns back up at $\gamma = 10^{-3}$, placing the optimum at 5×10^{-4} rather than at the largest weight decay tried. The same effect appears at $S = 50$, but weaker, with the optimum instead at $\gamma = 10^{-4}$. At $S = 10, 25$, held-out LPIPS increases monotonically with weight decay: with fewer diffusion steps, the schedule cannot overoptimize the constraint function to begin with, so regularization only costs reconstruction quality.

Box inpainting (Figure 11, right) is comparatively flat: held-out LPIPS is largely insensitive to weight decay across all S, except at $\gamma = 10^{-3}$, where it clearly hurts, except at $S = 10$ where it remains flat.

C.4 Generalization across tasks

This section asks whether a guidance schedule learned for one task generalizes to other tasks, or equivalently, whether task-specific training is beneficial. We study this on the five FFHQ noisy inverse tasks in three parts: we compare the learned schedules, evaluate each schedule on every other task, and repeat this evaluation after matching the schedule magnitudes to isolate the effect of shape. All schedules are the LEEGS runs (first draw) used for Table 1 at $S \in \{50, 100\}$; training details are given in Appendix B (Table 7). Learned schedules for ImageNet and the PDE tasks are given in Appendix G.

Learned schedules. Figure 12 (left) shows that the learned schedules differ in magnitude: box inpainting and the two deblur tasks cluster together, superresolution is larger, and random inpainting has by far the largest guidance. To compare shapes independently of magnitude, Figure 12 (right) divides each schedule by its mean. The shapes also differ, consistently across both S: gaussian deblur rises roughly linearly over the second half of the trajectory, while superresolution and random inpainting rise earliest and level off or drop towards the end, with a sharp drop in the final steps for random inpainting. Superresolution appears to differ the most with respect to shape from the other tasks, with a much more gradual slope as well as a hill shape.

Cross-task transfer. To test how much these differences matter in practice, we sample with the schedule learned for each task (row) under the measurement operator of every task (column), giving a 5×5 matrix for each S. Each cell is evaluated on the same fixed subset of 300 images of the test set of Table 1; the diagonal is re-evaluated on this subset and therefore differs slightly from the Table 1. Figure 13 (left) shows that the matched schedule is best in 4 of the 5 columns at both S. The exception is motion deblur, where the gaussian deblur schedule wins, but the two are nearly identical (0.209 vs. 0.212 at $S = 50$, 0.177 vs. 0.179 at $S = 100$). This is expected, since both tasks blur the image with a convolution kernel under the same noise model, and their learned schedules are close in magnitude.

Magnitude-matched transfer. Part of the transfer gap is a magnitude mismatch, most visibly for the random inpainting schedule applied to the deblur tasks. To isolate the effect of shape, we multiply each off-diagonal schedule by a scalar such that its mean guidance over the trajectory equals that of the target task's learned schedule, which leaves the shape unchanged. The diagonal has a factor

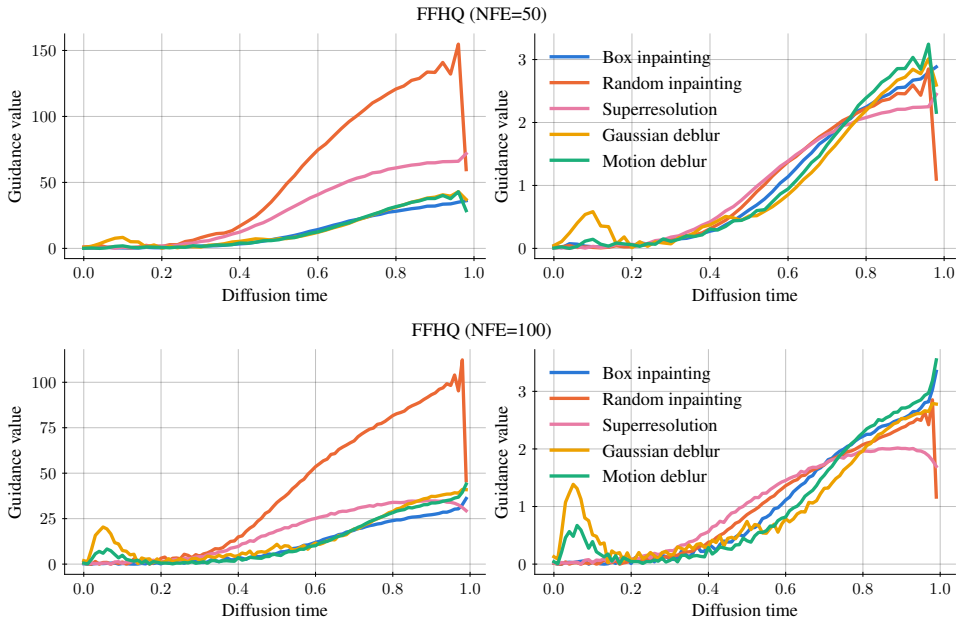


Figure 12: Learned guidance schedules for the five FFHQ noisy inverse tasks against normalized diffusion time (0: noisiest, 1: clean). *Left:* raw. *Right:* each schedule divided by its mean. *Top:* $S = 50$. *Bottom:* $S = 100$. The schedules differ strongly in magnitude and, to a lesser extent, in shape.

of 1 and is reused. Figure 13 (right) shows a mixed result, and the matched schedule no longer has a clear advantage. For most target tasks the columns flatten out (e.g. 0.242–0.247 for gaussian deblur at $S = 50$). Random inpainting is the exception: the random inpainting and superresolution schedules perform best (0.196, 0.217 at $S = 50$; 0.173, 0.171 at $S = 100$), while the other three remain substantially worse (≥ 0.242). These are the two schedules that rise earliest and turn down towards the end (Figure 12, right). Box inpainting is also selective: its own schedule remains best on its task at both S after rescaling.

In summary, task-specific training pays off: the learned schedule is best on its own task in nearly every case, and both of its components (magnitude and shape) matter. The magnitude, which differs by up to a factor of four across tasks, is learned jointly with the shape, and for some tasks (here, random and box inpainting) the task-specific shape gives a further advantage that remains even after matching magnitudes.

C.5 Generalization across number of steps

In this section we investigate how similar the guidance schedules learned at different numbers of steps S are, and whether a schedule learned at one S generalizes to a larger S without retraining. The latter goal may especially be helpful in practical applications where training is especially expensive, e.g. due to a large diffusion backbone. We consider two tasks: FFHQ box inpainting and the Helmholtz forward PDE problem.

Setup. For each task we train a separate schedule at every $S \in \{10, 20, 30, 50, 100\}$, with all other hyperparameters fixed to the task's main-results configuration (Appendix B). Each run is trained for 30 epochs for box inpainting, and 25 epochs for Helmholtz, which is sufficient for convergence of train LPIPS and relative error respectively. We evaluate on held-out samples disjoint from the training set: 300 FFHQ images with LPIPS for box inpainting, and 300 Helmholtz samples with the relative L_2 error on the solution u for Helmholtz.

Given a schedule trained at a source budget S_{src}, we adapt it to a larger target budget S_{tgt} at inference time using two strategies. *Interp.* linearly interpolates the learned schedule of size S_{src}, as a function of normalized diffusion time onto the larger grid of size S_{tgt}. While this preserves the shape of

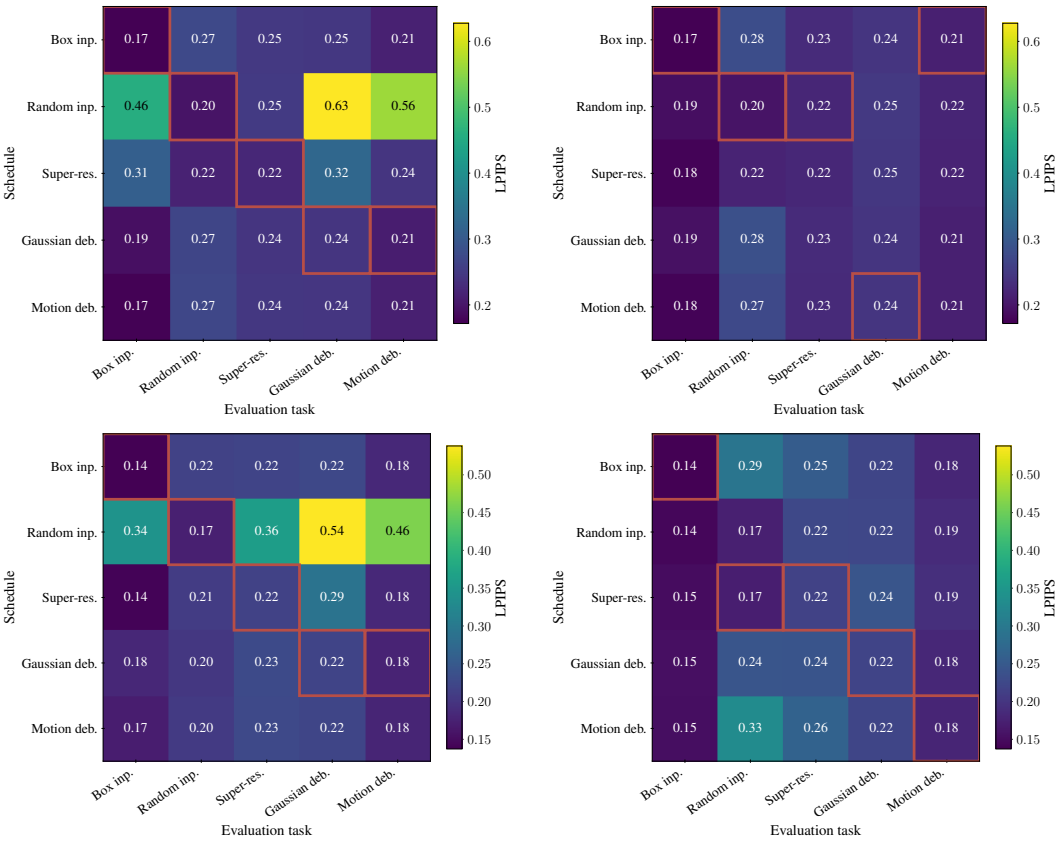


Figure 13: Held-out LPIPS (300 FFHQ images) when sampling with the schedule learned for the row task under the measurement operator of the column task. The best entry per column is in bold. *Left:* original schedules. *Right:* off-diagonal schedules rescaled to the mean guidance of the column task's schedule. *Top:* $S = 50$. *Bottom:* $S = 100$. The matched schedule is best in almost every column before rescaling; after rescaling, some tasks continue to remain selective with respect to shape.

the schedule, the magnitude of the interpolated schedule may not be calibrated for the new budget. Therefore, we try the *Interp.+rescale* strategy, that additionally multiplies the interpolated schedule by a scalar that corrects its magnitude. We fit a line $f(S)$ to the mean guidance value of the five trained schedules as a function of S, and use the multiplier $f(S_{\text{tgt}})/f(S_{\text{src}})$ so that the mean guidance value of the *Interp.+rescale* schedule matches the value $f(S_{\text{tgt}})$ predicted by the fit at the target budget. Both interpolation strategies are compared against the schedule *trained* directly at S_{tgt}.

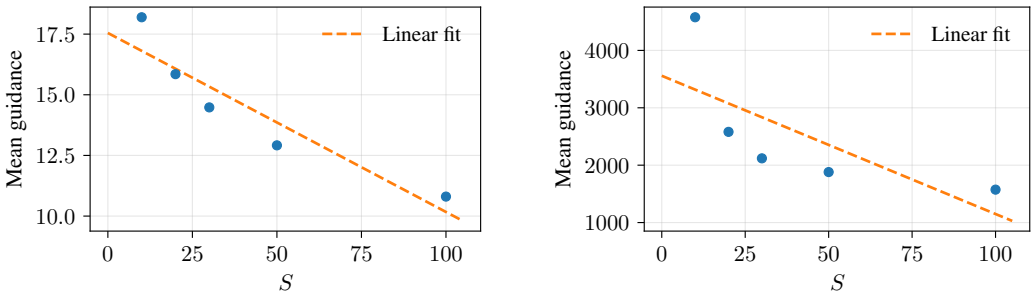


Figure 14: Mean guidance value of the trained schedules as a function of S, with the linear fit used for rescaling. *Left:* FFHQ box inpainting. *Right:* Helmholtz forward. The mean guidance decreases monotonically with S.

Learned schedules. Figure 3 in the main text shows the schedules trained at each S. For both tasks the shapes are similar across S, while the magnitude decreases with S. This is also reflected in Figure 14, which shows that the mean guidance decreases monotonically with S. We fit a line to the data to give a coarse approximation of this trend to compute rescale factors for the *Interp.+rescale* strategy. The decreasing trend is expected: with more diffusion steps, guidance can be more distributed across the trajectory, so the mean guidance can be smaller while still achieving the same overall effect.

$S_{\text{src}} \to S_{\text{tgt}}$	Trained	Interp.	Interp.+rescale
$10 \to 20$	0.217	**0.203**	0.204
$10 \to 30$	0.173	**0.169**	0.170
$10 \to 50$	0.140	0.142	**0.139**
$10 \to 100$	0.119	0.156	**0.113**
$20 \to 30$	0.173	**0.172**	0.174
$20 \to 50$	0.140	**0.139**	0.140
$20 \to 100$	0.119	0.133	**0.115**
$30 \to 50$	0.140	**0.138**	0.141
$30 \to 100$	0.119	0.126	**0.115**
$50 \to 100$	0.119	0.122	**0.114**

Table 13: FFHQ box inpainting: LPIPS (↓) on 300 held-out samples. **Bold**: best of the three columns.

$S_{\text{src}} \to S_{\text{tgt}}$	Trained	Interp.	Interp.+rescale
$10 \to 20$	11.70%	**9.80%**	10.01%
$10 \to 30$	10.34%	9.13%	**9.01%**
$10 \to 50$	9.33%	9.88%	**8.73%**
$10 \to 100$	9.07%	15.05%	**8.77%**
$20 \to 30$	10.34%	**9.54%**	9.77%
$20 \to 50$	9.33%	9.09%	**9.04%**
$20 \to 100$	**9.07%**	11.73%	9.14%
$30 \to 50$	9.33%	**9.06%**	9.24%
$30 \to 100$	**9.07%**	11.11%	9.36%
$50 \to 100$	**9.07%**	10.13%	9.42%

Table 14: Helmholtz forward: relative L_2 error on u (%, ↓) on 300 held-out samples. **Bold**: best of the three columns.

Interpolation and rescaling. Tables 13 and 14 show that plain interpolation performs well for small gaps between S_{src} and S_{tgt}, often matching or even beating the trained schedule (the latter could indicate that the guidance magnitude for the target had not yet converged to the optimal value), but degrades as the gap grows. However interpolating from $S_{\text{src}} \in \{10, 20, 30, 50\}$ to $S_{\text{tgt}} = 100$ performs consistently worse than the trained schedule across both tasks. Rescaling largely removes this degradation: Interp.+rescale performs very close to the trained schedule in every row of both tables. For the smallest gaps, where there is little magnitude mismatch to correct, plain interpolation is sometimes marginally better.

In conclusion, we see that the schedules learned for the same task at different S have a strongly shared shape, and the trend in their mean guidance value is predictable. The latter fact can be leveraged to improve the performance of LEEGS by training at a small number of steps and extrapolating the schedule to a larger number of steps when the computational budget for training is limited.

C.6 Ablation on trajectory objective vs terminal objective

This section studies the effect of dropping the non-terminal (trajectory-level) constraint term from the training objective, keeping only the terminal constraint $h(x_0, c)$, where x_0 is the final denoised

sample. In Section 3 we motivated the trajectory objective by noting that the terminal constraint's gradient with respect to the guidance scale w_s decays for steps s far from the end of the trajectory, so a terminal-only objective gives it progressively less usable training signal as the trajectory gets longer. We now verify this intuition empirically for the box inpainting task on FFHQ, using the same training setup as in the cross-step-budget experiment of Appendix C.5. We reuse the trained schedules from the cross-step-budget experiment of Appendix C.5 ($S \in \{10, 20, 30, 50, 100\}$ for box inpainting, which wer trained for 30 epochs each with the full LEEGS objective $\mathcal{L}(W)$ (which we call "Trajectory"). Using the same hyperparameters, we train a *terminal-only* variant for each S, where the loss function is $\mathcal{L}'(W) = \mathbb{E}_{c,x_s}[h(x_0, c)]$. Both variants are evaluated on the same held-out set of 300 samples used for the cross-step-budget study.

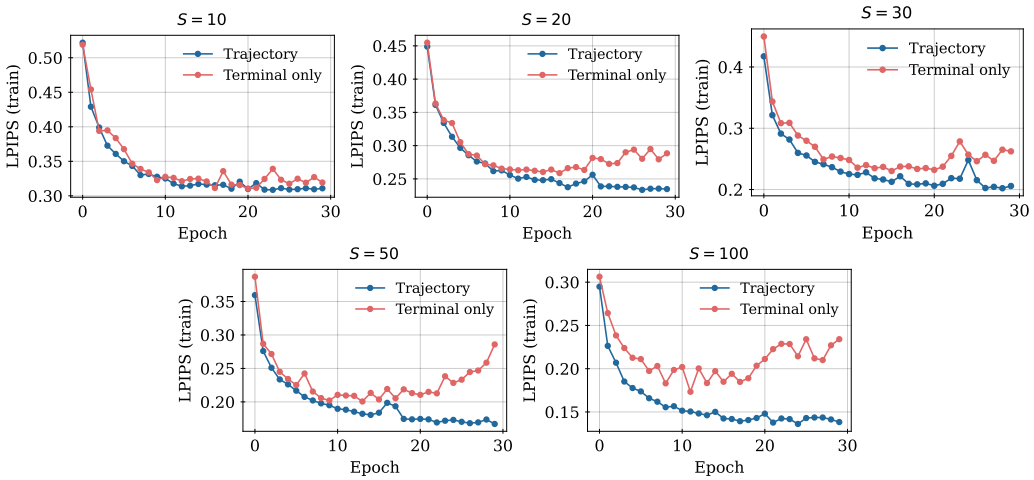


Figure 15: Training LPIPS vs. epoch, full vs. terminal-only constraint set, one panel per diffusion step budget S.

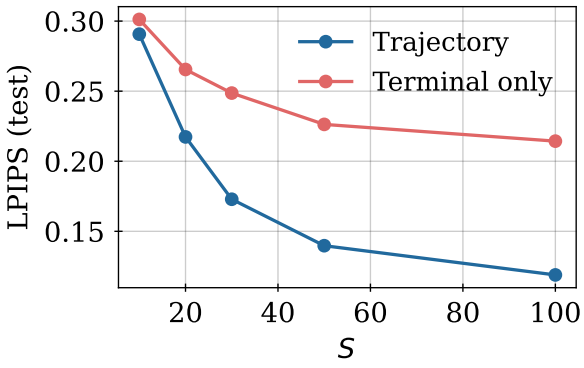


Figure 16: Held-out LPIPS vs. S, full vs. terminal-only constraint set.

Observations. Figure 15 shows the training LPIPS vs. epoch for the full (trajectory) and terminal-only objectives, for each S. We see that the training curve for the terminal-only objective is consistently worse than the full objective, and the gap widens with S. Moreover, the terminal-only objective also exhibits a reward-hacking elbow at which the terminal constraint keeps improving while the reconstruction quality degrades. However, note that the gap between the two objectives with respect to training LPIPS cannot be completely attributed to reward-hacking, since there is a gap even if we consider the best training LPIPS achieved by each objective. Figure 16 shows the held-out LPIPS vs. S for both objectives. Here it is even more visually clear that the gap between the two objectives widens with S. Moreover, we see a plateauing effect for the LPIPS with increasing S for both runs, but the terminal-only curve is much flatter than the full objective curve, indicating that the terminal-only objective is less able to leverage additional diffusion steps to improve reconstruction quality.

D Extension to sampling with renoising

This section applies LEEGS to a sampler that alternates guided reverse diffusion with renoising, i.e. forward diffusion back to a noisier state. Renoising is used by, e.g., RePaint (Lugmayr et al., 2022), DDNM (Wang et al., 2023), which calls it time travel, and DAPS (Zhang et al., 2025). We denote this variant LEEGS-renoise, and the standard DPS-style sampling used in the rest of the paper simply LEEGS. Using normalized diffusion time $\tau \in [0, 1]$ (0: noisiest, 1: clean), the sampler splits the S guided steps into K outer steps of S/K inner steps each. Outer step $j = 0, \ldots, K-1$ starts at $\tau = j/K$ and reverse-diffuses with guidance to the clean end $\tau = 1$ over S/K uniformly spaced steps. The resulting sample is then renoised in without guidance, to the start $\tau = (j+1)/K$ of the next outer step, using the forward process. The sampler therefore moves back and forth between the clean and noisy ends, with progressively later starting points after each outer step. Thus, each outer step also covers a shorter interval with finer steps than the previous one. The final sample is the output of the last outer step. The learned schedule has one weight per guided step, i.e. S weights in total, and the NFE budget is the same as for LEEGS with S steps, since renoising requires no network evaluation. For example, in Figure 17 (box inpainting), the five outer steps start at $\tau = 0, 0.2, 0.4, 0.6, 0.8$, and each has its own 10 weights. Training only changes how the trajectory is unrolled: only COMPUTEGRADIENT of Algorithm 1 is replaced, by the version in Algorithm 2; GUIDEDSTEP itself is unchanged.

Algorithm 2 LEEGS-renoise training (replaces COMPUTEGRADIENT of Algorithm 1)

function COMPUTEGRADIENT(W, c, `compute_leegs_approx`)
  $x \sim \mathcal{N}(0, \mathbf{I})$
  $\ell \leftarrow 0$; $i \leftarrow 1$ ▷ i is the flat index into $W = (w_1, \ldots, w_S)$
  for $j = 0, \ldots, K-1$ **do** ▷ outer step
    for $s = 1, \ldots, S/K$ **do** ▷ inner step, from $\tau = j/K$ to $\tau = 1$
      $t \leftarrow$ `time_at`(j, s) ▷ s-th point of linspace$(j/K,\ 1,\ S/K + 1)$
      $(x, h_i) \leftarrow$ CHECKPOINT$\big($GUIDEDSTEP, x, t, w_i, c, `compute_leegs_approx`$\big)$
      $\ell \leftarrow \ell + h_i$; $i \leftarrow i + 1$
    end for
    if $j < K - 1$ **then**
      $\epsilon \sim \mathcal{N}(0, \mathbf{I})$
      $x \leftarrow$ RENOISE$\big(x, (j+1)/K, \epsilon\big)$ ▷ renoise to the start of outer step $j + 1$
    end if
  end for
  $h_0 \leftarrow$ `compute_guidance_function`(x, c) ▷ terminal term $h(x_0, c)$
  $\ell \leftarrow \ell + h_0$
  return BACKWARD$(\ell,\ W)$
end function

Training and evaluation setup. We use $K = 5$ outer steps of $S/K = 10$ inner steps each ($S = 50$) on all five FFHQ noisy inverse tasks. The learning rates, epochs and weight decay are given in Table 15; all other hyperparameters match the LEEGS runs of Table 1 (Appendix B). Table 15 shows the results of evaluating LEEGS-renoise against the first random run of LEEGS in Table 1, i.e. they share the same training and test set of 950 images. The relative change is computed as (LEEGS-renoise − LEEGS)/LEEGS, so negative values indicate that LEEGS-renoise outperforms LEEGS.

Subtask	Learning rate η	Epochs	Weight decay
Super-resolution (4×)	0.3	15	0.0
Box inpainting	0.2	30	0.0
Random inpainting	0.8	30	0.0
Gaussian deblur	0.5	25	0.0
Motion deblur	0.3	15	0.0

Table 15: LEEGS-renoise training configuration, $S = 50$, $K = 5$.

Task	LEEGS	LEEGS-renoise	Relative change
Box inpainting	0.174	**0.165**	-5.2%
Random inpainting	0.196	**0.181**	-7.5%
Gaussian deblur	**0.243**	0.246	$+1.3\%$
Motion deblur	**0.212**	0.227	$+7.0\%$
Superresolution	**0.222**	0.224	$+0.6\%$

Table 16: LPIPS on the FFHQ test set (950 images), $S = 50$.

Results. Table 16 shows that LEEGS-renoise is comparable to LEEGS overall, and improves on both inpainting tasks (-5.2% for box, -7.5% for random inpainting).

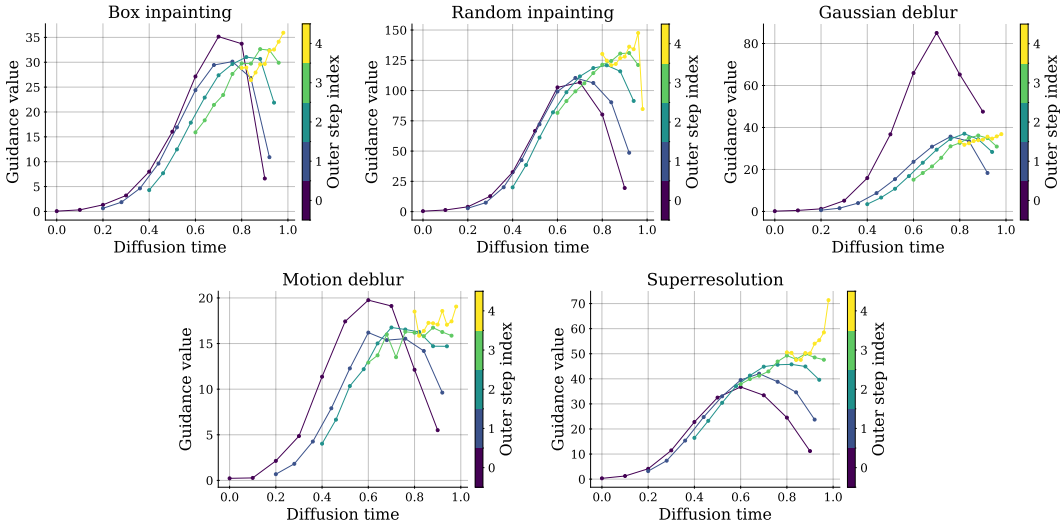


Figure 17: Learned LEEGS-renoise guidance schedules for the five FFHQ tasks ($S = 50$, $K = 5$) against normalized diffusion time (0: noisiest, 1: clean); color indicates the outer step. Every outer step except the last has a hill shape.

Learned schedules. Figure 17 shows the learned schedules. The magnitude varies somewhat between outer steps. Every outer step except the last has a hill shape, rising and then turning down towards its end. The last outer step instead increases roughly linearly, except for random inpainting, where it drops sharply at the final step. Similar to the observation for LEEGS (Appendix C.4), the learned schedules for superresolution and random inpainting are also similar for LEEGS-renoise. For both, the peak guidance grows with each outer step, whereas for the other tasks it stays roughly constant after the first outer step. Gaussian deblur is an outlier, with an anomalously large magnitude in the first outer step (peak ≈ 85, vs. ≈ 37 in later outer steps).

Conclusion. This experiment shows that the main idea of this paper (learning time dependent guidance scales to minimize guidance functions by backpropagating through the sampling trajectory) is applicable to diverse scenarios beyond DPS-style sampling. Moreover, while DPS-style sampling may admit competitive simple schedules, such as the increasing linear schedule in Table 1, it is much harder to tune or approximate a schedule by hand for a sampler with renoising, whose schedule is a sequence of per-outer-step segments. For LEEGS, the cost is the same in both cases, since it depends only on the number of NFEs in the sampling trajectory.

E Comparison at $S = 10$ NFEs (FFHQ)

As a further stress test at a much smaller NFE budget, we repeat the noisy-inverse-problem comparison of Table 1 at $S = 10$ NFEs, on FFHQ. We compare against Const., DAPS, DSG, CDIM and Linear increase, using the same hyperparameter-selection protocol as Appendix B (grid search on 20 validation images for Const./DSG/Linear increase, and a discrete-configuration search for DAPS/CDIM), evaluated on the same 950-image FFHQ test set as Table 1. Linear decrease is again omitted (it is uniformly the worst baseline at every other NFE budget in this paper); we did not extend the comparison to ImageNet or run the 1000-NFE DPS reference at $S = 10$. Unlike the main experiments (three independent training runs per subtask), we train a single LEEGS run per subtask for this $S = 10$ extension. Table 17 gives the learning rate, number of epochs and weight decay used.

Subtask	Learning rate η	Epochs	Weight decay
Super-resolution ($4\times$)	0.3	30	0.0
Box inpainting	0.2	20	0.0
Random inpainting	0.8	30	0.0
Gaussian deblur	0.3	20	0.0
Motion deblur	0.3	20	0.0

Table 17: LEEGS training configuration for the $S = 10$ FFHQ extension (Table 18).

	NFEs	SR (x4)		Inpaint (box)		Inpaint (random)		Deblur (gauss)		Deblur (motion)	
		LPIPS	FID	LPIPS	FID	LPIPS	FID	LPIPS	FID	LPIPS	FID
Const.	10	0.562	104.92	0.569	102.78	0.583	102.25	0.563	108.08	0.570	103.75
DAPS	10	0.331	60.55	**0.207**	41.14	0.619	234.89	0.323	57.76	**0.294**	**58.96**
DSG	10	0.588	133.25	0.616	127.00	0.615	140.01	0.530	131.50	0.535	127.15
CDIM	≈ 10	0.297	**40.11**	0.217	**33.10**	0.363	75.65	0.326	**40.19**	0.411	69.11
Linear increase	10	0.469	100.31	0.499	99.20	0.483	99.07	0.479	96.26	0.474	97.06
LEEGS (ours)	10	**0.289**	54.88	0.323	53.45	**0.305**	**51.34**	**0.322**	63.21	0.305	60.05

Table 18: Comparison for noisy image inverse tasks (FFHQ), at $S = 10$ NFEs. **Green** (bold): best; **blue**: second best.

As at $S = 50$ and $S = 100$ (see the CDIM item in Appendix B and Table 6), CDIM's NFE count varies per sample; the value $T = 4$ used here was chosen as the closest match to 10 NFEs among $T \in \{4, 5, 6\}$, and its actual average NFEs per task range from 9.7 to 16.6 (mean 12.4 across the five tasks).

Overall, LEEGS has the best performance in this comparison: it achieves the best result on four of the ten columns (SR and random-inpainting LPIPS, random-inpainting FID, and Gaussian-deblur LPIPS) and is best or second-best on eight of the ten. DAPS and CDIM are the strongest competitors, each best on three columns, with the latter having an advantage with respect to NFEs. Meanwhile, DSG and Const. perform very poorly, and the Linear increase baseline – a strong heuristic at higher NFE budgets – also fails to be the best at any task at this very low budget.

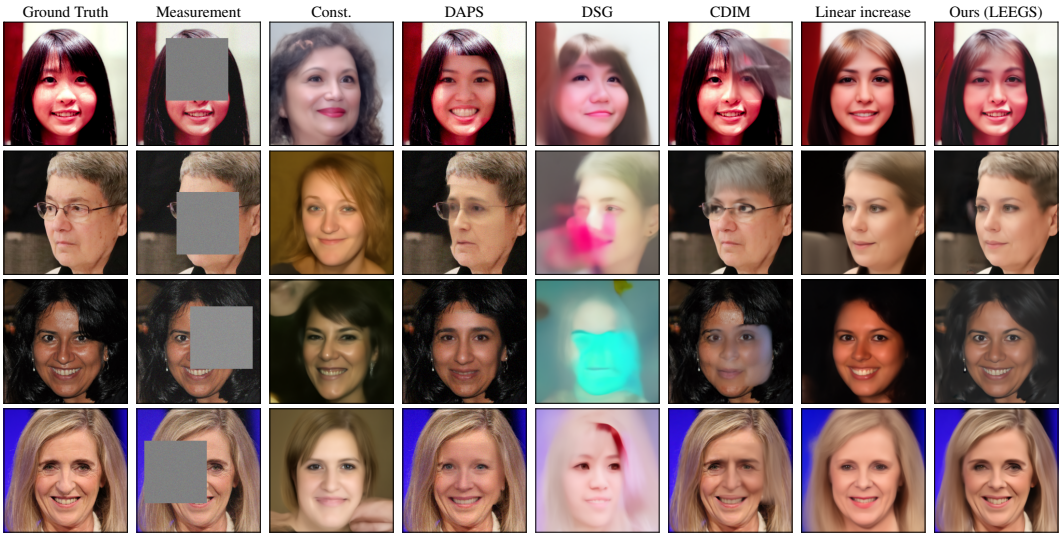


Figure 18: Box inpainting, FFHQ, 10 NFEs.

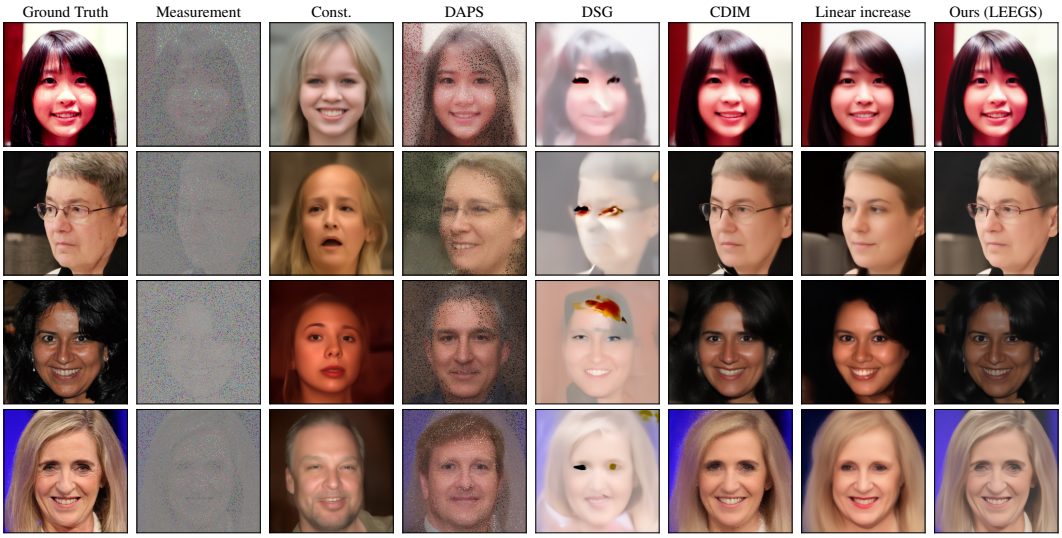


Figure 19: Random inpainting, FFHQ, 10 NFEs.

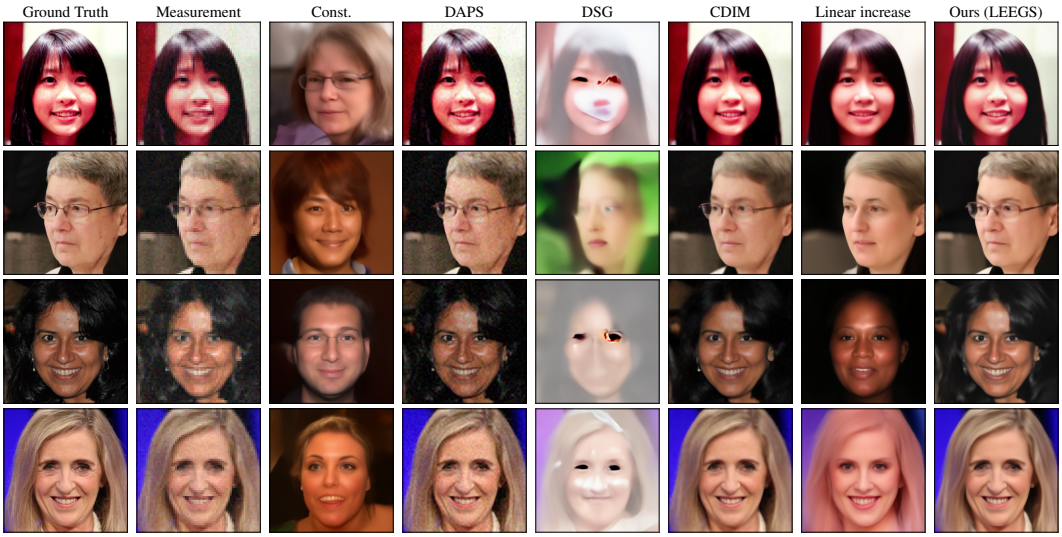


Figure 20: Superresolution ($\times 4$), FFHQ, 10 NFEs.

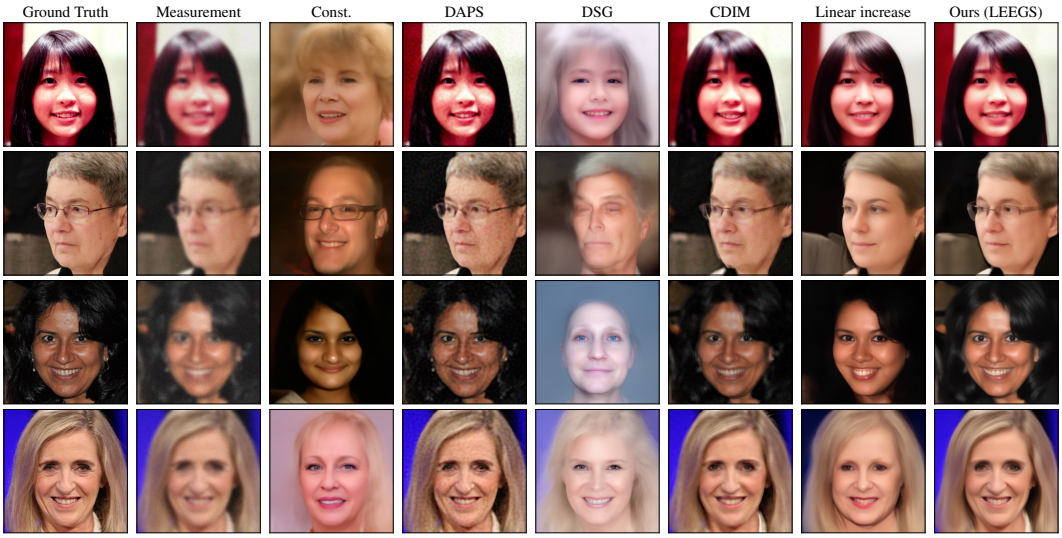


Figure 21: Gaussian deblur, FFHQ, 10 NFEs.

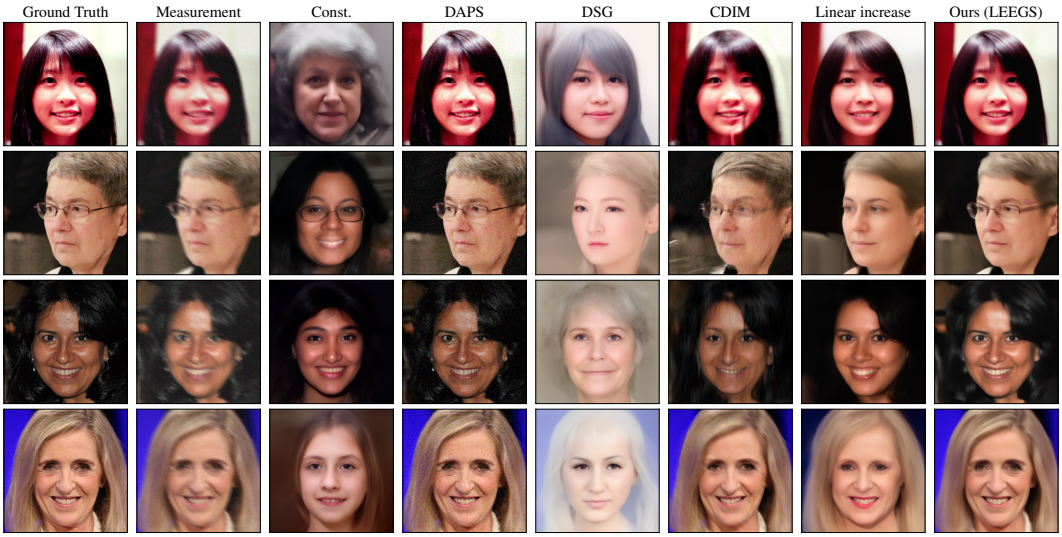


Figure 22: Motion deblur, FFHQ, 10 NFEs.

F Additional metrics for main experiments

This section gives the variance of LEEGS across training draws for the main results, and additional metrics for the noisy inverse and image inpainting problems.

F.1 Variance of LEEGS across draws

The LEEGS rows of Tables 1, 2, 3, 4 and 5 are means over 3 independent draws of the training set. Tables 19–23 give the corresponding standard deviations. The baselines have no repeated draws and are omitted.

NFEs	SR (x4)		Inpaint (box)		Inpaint (random)		Deblur (gauss)		Deblur (motion)	
	LPIPS	FID	LPIPS	FID	LPIPS	FID	LPIPS	FID	LPIPS	FID
50	0.0002	0.01	0.0034	0.22	0.0007	0.14	0.0001	0.16	0.0012	0.44
100	0.0004	0.07	0.0047	0.33	0.0016	0.11	0.0039	0.15	0.0013	0.41

Table 19: Standard deviation across 3 draws for the LEEGS rows of Table 1 (FFHQ).

NFEs	SR (x4)		Inpaint (box)		Inpaint (random)		Deblur (gauss)		Deblur (motion)	
	LPIPS	FID	LPIPS	FID	LPIPS	FID	LPIPS	FID	LPIPS	FID
50	0.0001	0.23	0.0026	1.45	0.0056	1.07	0.0035	1.23	0.0048	2.41
100	0.0013	0.29	0.0092	4.29	0.0006	0.43	0.0039	0.88	0.0151	3.67

Table 20: Standard deviation across 3 draws for the LEEGS rows of Table 2 (ImageNet).

Dataset	NFEs	Wide	Narrow	S. res. (2x)	Alternate Lines	Half
Imagenet	200	0.0019	0.0040	0.0016	0.0018	0.0041
CelebA-HQ	200	0.0010	0.0001	0.0001	0.0005	0.0016

Table 21: Standard deviation across 3 draws for the LEEGS rows of Table 3 (image inpainting).

NFEs	FaceID Score	FID
50	0.0036	0.90
100	0.0119	0.85

Table 22: Standard deviation across 3 draws for the LEEGS rows of Table 4 (Face-ID guidance).

NFEs	Poisson		Helmholtz		Darcy	
	Forward	Inverse	Forward	Inverse	Forward	Inverse
50x1	0.01%	0.06%	0.08%	0.25%	4.27%	1.54%
100x1	0.01%	0.19%	0.06%	0.18%	1.45%	5.28%

Table 23: Standard deviation across 3 draws for the LEEGS rows of Table 5 (PDE problems).

F.2 Noisy inverse problems

For every row of Tables 1 and 2, we report the variance of LPIPS across the test samples (Tables 24 and 25), PSNR (Tables 26 and 27) and SSIM (Tables 28 and 29). PSNR (peak signal-to-noise ratio) measures the pixel-wise error between the reconstruction and the ground truth on a logarithmic scale, as $10\log_{10}(\mathrm{MAX}^2/\mathrm{MSE})$, where MAX is the maximum pixel value. SSIM (structural similarity index) compares local statistics (mean intensity, contrast and correlation) of the two images over sliding windows, and ranges up to 1 for identical images. Higher is better for both. Unlike LPIPS, both

are pixel-level metrics that tend to favor smooth reconstructions over sharp but slightly misaligned details. We compute both with `piq` on images normalized to $[0, 1]$, so that MAX $= 1$. The DPS reference row is omitted, since its results are taken from Chung et al. (2023).

	NFEs	SR (x4)	Inpaint (box)	Inpaint (random)	Deblur (gauss)	Deblur (motion)
Const.	50	0.0161	0.0166	0.0149	0.0185	0.0108
DAPS	50	**0.0011**	0.0014	0.0019	**0.0011**	**0.0011**
DSG	50	0.0021	0.0053	0.0037	0.0025	0.0018
CDIM	50	0.0018	**0.0011**	0.0024	0.0026	0.0037
Linear increase	50	0.0013	0.0015	**0.0016**	0.0021	0.0016
Linear decrease	50	0.0153	0.0146	0.0132	0.0152	0.0150
LEEGS (ours)	50	0.0014	0.0012	**0.0016**	0.0018	0.0014
Const.	100	0.0039	0.0078	0.0056	0.0064	0.0055
DAPS	100	**0.0010**	0.0014	0.0017	**0.0011**	**0.0010**
DSG	100	0.0022	0.0031	0.0020	0.0024	0.0033
CDIM	100	0.0016	0.0011	0.0028	0.0023	0.0034
Linear increase	100	0.0014	0.0009	**0.0012**	0.0020	0.0016
Linear decrease	100	0.0113	0.0165	0.0095	0.0082	0.0079
LEEGS (ours)	100	0.0013	**0.0006**	**0.0012**	0.0016	**0.0010**

Table 24: Variance of LPIPS across test samples (↓) for the noisy inverse problems (FFHQ). **Green** (bold): best; **blue**: second best.

	NFEs	SR (x4)	Inpaint (box)	Inpaint (random)	Deblur (gauss)	Deblur (motion)
Const.	50	0.0084	0.0220	0.0129	0.0121	0.0103
DAPS	50	**0.0038**	0.0032	**0.0021**	**0.0044**	**0.0030**
DSG	50	0.0147	0.0106	0.0144	0.0102	0.0056
CDIM	50	0.0099	**0.0021**	0.0062	0.0090	0.0098
Linear increase	50	0.0114	0.0080	0.0110	0.0110	0.0126
Linear decrease	50	0.0119	0.0108	0.0129	0.0098	0.0131
LEEGS (ours)	50	0.0060	0.0060	0.0077	0.0089	0.0090
Const.	100	0.0182	0.0081	0.0186	0.0143	0.0097
DAPS	100	**0.0041**	0.0031	**0.0041**	**0.0049**	0.0033
DSG	100	0.0069	0.0091	0.0078	0.0098	**0.0024**
CDIM	100	0.0097	**0.0018**	0.0059	0.0093	0.0098
Linear increase	100	0.0086	0.0059	0.0055	0.0100	0.0127
Linear decrease	100	0.0172	0.0190	0.0158	0.0143	0.0146
LEEGS (ours)	100	0.0059	0.0027	0.0046	0.0082	0.0092

Table 25: Variance of LPIPS across test samples (↓) for the noisy inverse problems (ImageNet). **Green** (bold): best; **blue**: second best.

	NFEs	SR (x4)	Inpaint (box)	Inpaint (random)	Deblur (gauss)	Deblur (motion)
Const.	50	19.62	16.25	19.53	18.59	17.15
DAPS	50	28.04	**24.24**	20.65	**28.20**	**29.85**
DSG	50	21.83	15.89	18.78	20.03	10.03
CDIM	50	**28.32**	22.47	26.50	26.79	25.91
Linear increase	50	27.40	22.27	**26.86**	25.66	27.43
Linear decrease	50	15.35	14.50	15.88	15.04	15.19
LEEGS (ours)	50	27.47	22.59	26.37	26.35	27.94
Const.	100	26.26	19.13	24.43	23.60	19.41
DAPS	100	**28.58**	**24.66**	23.63	**28.65**	**30.58**
DSG	100	21.19	16.38	21.78	19.99	12.67
CDIM	100	28.24	22.43	26.21	26.88	26.22
Linear increase	100	27.45	22.80	**27.46**	25.78	27.76
Linear decrease	100	20.11	16.55	19.94	19.51	19.75
LEEGS (ours)	100	27.43	23.62	27.17	26.54	28.76

Table 26: PSNR in dB (↑) for the noisy inverse problems (FFHQ). **Green** (bold): best; **blue**: second best.

	NFEs	SR (x4)	Inpaint (box)	Inpaint (random)	Deblur (gauss)	Deblur (motion)
Const.	50	15.08	14.29	14.79	14.71	13.74
DAPS	50	24.40	18.35	17.99	**24.44**	**27.06**
DSG	50	18.85	13.27	15.77	20.18	11.64
CDIM	50	23.45	**18.64**	22.66	22.37	22.57
Linear increase	50	23.65	16.09	22.25	21.16	22.36
Linear decrease	50	12.91	11.64	12.56	13.12	12.46
LEEGS (ours)	50	**24.45**	17.14	**23.62**	22.44	24.47
Const.	100	20.41	15.29	19.56	18.45	16.13
DAPS	100	**24.71**	**18.86**	20.27	**24.66**	**27.54**
DSG	100	21.42	10.62	20.47	19.60	8.68
CDIM	100	23.66	18.59	22.43	22.66	23.01
Linear increase	100	24.13	16.28	24.26	21.72	23.00
Linear decrease	100	15.81	12.51	15.80	15.86	16.04
LEEGS (ours)	100	24.37	18.20	**24.84**	22.68	25.27

Table 27: PSNR in dB (↑) for the noisy inverse problems (ImageNet). **Green** (bold): best; **blue**: second best.

	NFEs	SR (x4)	Inpaint (box)	Inpaint (random)	Deblur (gauss)	Deblur (motion)
Const.	50	0.614	0.561	0.621	0.568	0.512
DAPS	50	0.710	0.724	0.509	0.715	0.758
DSG	50	0.739	0.658	0.696	0.675	0.333
CDIM	50	0.798	**0.839**	0.799	0.733	0.731
Linear increase	50	0.808	0.829	**0.844**	0.758	0.820
Linear decrease	50	0.470	0.444	0.488	0.457	0.465
LEEGS (ours)	50	**0.812**	0.832	**0.844**	**0.781**	**0.833**
Const.	100	0.787	0.687	0.757	0.692	0.559
DAPS	100	0.753	0.738	0.633	0.752	0.804
DSG	100	0.727	0.695	0.745	0.685	0.449
CDIM	100	0.800	0.839	0.790	0.744	0.745
Linear increase	100	**0.809**	0.835	**0.844**	0.760	0.827
Linear decrease	100	0.600	0.537	0.599	0.567	0.579
LEEGS (ours)	100	0.806	**0.840**	0.838	**0.784**	**0.849**

Table 28: SSIM (↑) for the noisy inverse problems (FFHQ). **Green** (bold): best; **blue**: second best.

	NFEs	SR (x4)	Inpaint (box)	Inpaint (random)	Deblur (gauss)	Deblur (motion)
Const.	50	0.341	0.567	0.385	0.354	0.324
DAPS	50	0.601	0.657	0.351	**0.601**	**0.715**
DSG	50	0.554	0.495	0.516	0.534	0.303
CDIM	50	0.625	**0.771**	0.661	0.575	0.608
Linear increase	50	0.650	0.653	0.678	0.538	0.598
Linear decrease	50	0.316	0.295	0.311	0.304	0.300
LEEGS (ours)	50	**0.683**	0.698	**0.728**	0.598	0.695
Const.	100	0.535	0.686	0.544	0.445	0.357
DAPS	100	0.635	0.669	0.465	**0.630**	**0.752**
DSG	100	0.628	0.470	0.613	0.547	0.207
CDIM	100	0.632	**0.772**	0.650	0.586	0.627
Linear increase	100	0.674	0.691	0.737	0.561	0.628
Linear decrease	100	0.391	0.368	0.399	0.373	0.385
LEEGS (ours)	100	**0.681**	0.744	**0.745**	0.610	0.729

Table 29: SSIM (↑) for the noisy inverse problems (ImageNet). **Green** (bold): best; **blue**: second best.

F.3 Image inpainting problems

For the 200-NFE rows of Table 3, we report the variance of LPIPS across the test samples (Table 30), PSNR (Table 31) and SSIM (Table 32), computed as for the noisy inverse problems. The RePaint reference row at ≈ 2400 NFEs is omitted, since its results are taken from Lugmayr et al. (2022).

Dataset	Method	NFEs	Wide	Narrow	S. res. (2x)	Alternate Lines	Half
Imagenet	**LEEGS (ours)**	200	**0.0058**	**0.0045**	0.0029	**0.0020**	0.0030
	RePaint	200	0.0093	0.0075	**0.0021**	0.0079	**0.0014**
CelebA-HQ	**LEEGS (ours)**	200	**0.0022**	**0.0011**	**0.0003**	**0.0002**	**0.0013**
	RePaint	200	0.0053	0.0045	0.0023	**0.0002**	0.0017

Table 30: Variance of LPIPS across test samples (↓) for image inpainting. **Green** (bold): best.

Dataset	Method	NFEs	Wide	Narrow	S. res. (2x)	Alternate Lines	Half
Imagenet	**LEEGS (ours)**	200	**20.28**	**24.20**	**27.76**	**29.74**	**13.71**
	RePaint	200	18.88	21.69	16.81	21.65	13.47
CelebA-HQ	**LEEGS (ours)**	200	**24.99**	**29.00**	**32.76**	35.00	**16.41**
	RePaint	200	21.40	24.73	29.24	**36.46**	13.88

Table 31: PSNR in dB (↑) for image inpainting. **Green** (bold): best.

Dataset	Method	NFEs	Wide	Narrow	S. res. (2x)	Alternate Lines	Half
Imagenet	**LEEGS (ours)**	200	0.777	**0.808**	**0.834**	**0.880**	**0.594**
	RePaint	200	**0.784**	0.794	0.295	0.635	0.584
CelebA-HQ	**LEEGS (ours)**	200	**0.879**	**0.917**	**0.941**	0.964	**0.720**
	RePaint	200	0.847	0.868	0.845	**0.966**	0.670

Table 32: SSIM (↑) for image inpainting. **Green** (bold): best.

G Additional learned guidance schedules

This section gives the guidance schedules learned by LEEGS for the main experiments, grouped by task. All schedules are plotted against normalized diffusion time (0: noisiest, 1: clean), so that schedules trained at different NFE budgets are directly comparable. The training configurations are given in Appendix B.

G.1 Noisy inverse problems

The schedules for FFHQ are given in Figure 12 (Appendix C.4).

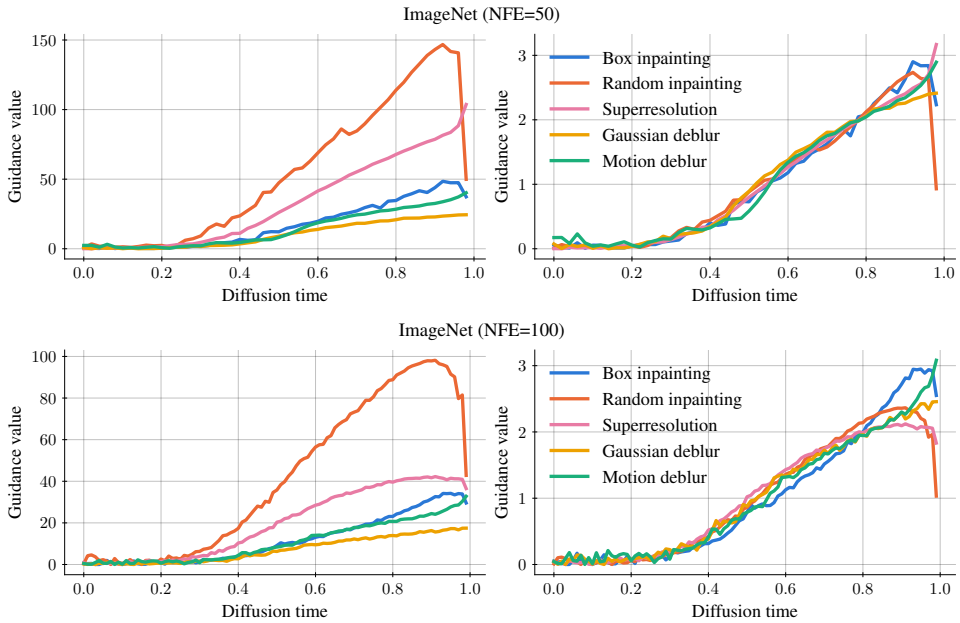


Figure 23: ImageNet. *Left:* raw. *Right:* mean-normalized. *Top:* 50 NFEs. *Bottom:* 100 NFEs.

G.2 Image inpainting

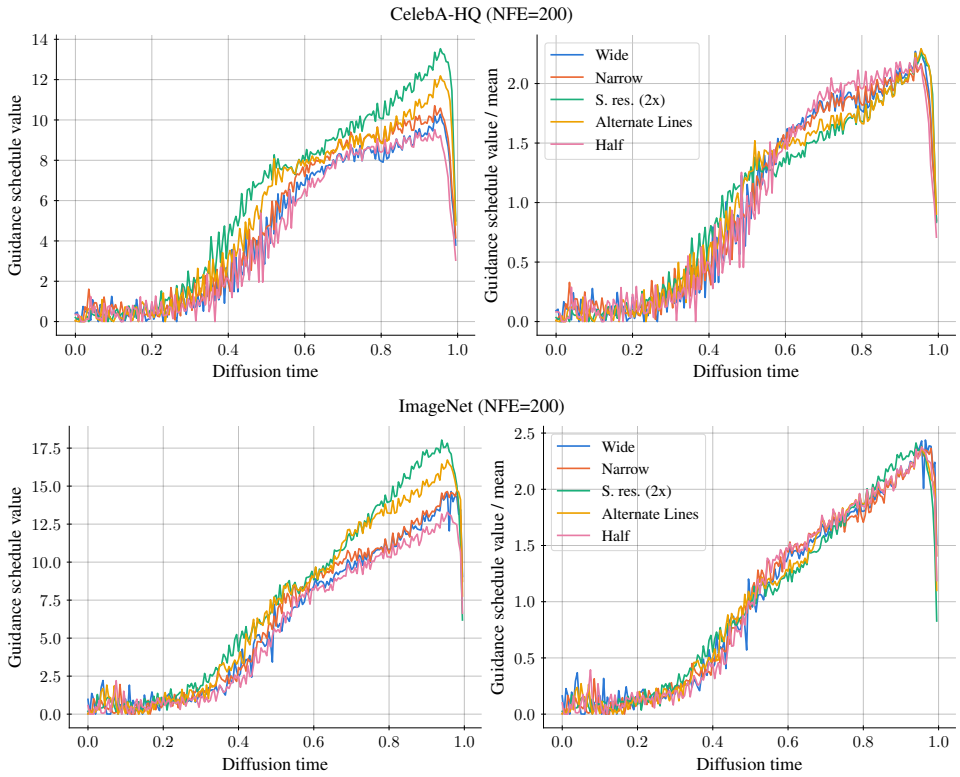


Figure 24: Image inpainting, 200 NFEs. *Left:* raw. *Right:* mean-normalized. *Top:* CelebA-HQ. *Bottom:* ImageNet.

G.3 Face-ID guidance

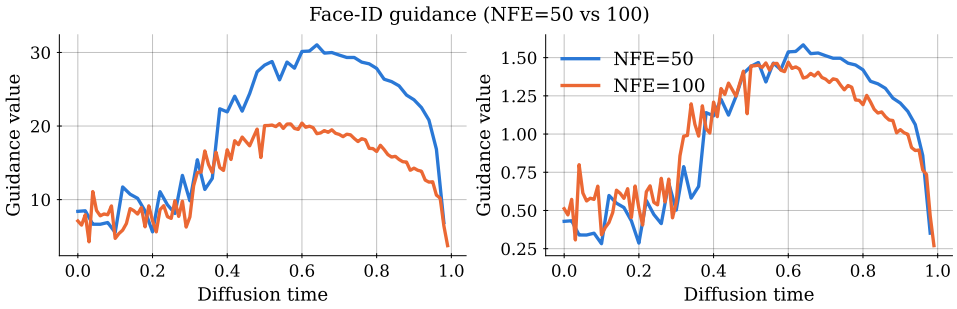


Figure 25: Face-ID guidance. *Left:* raw. *Right:* mean-normalized. Both NFE budgets overlaid.

G.4 PDE problems

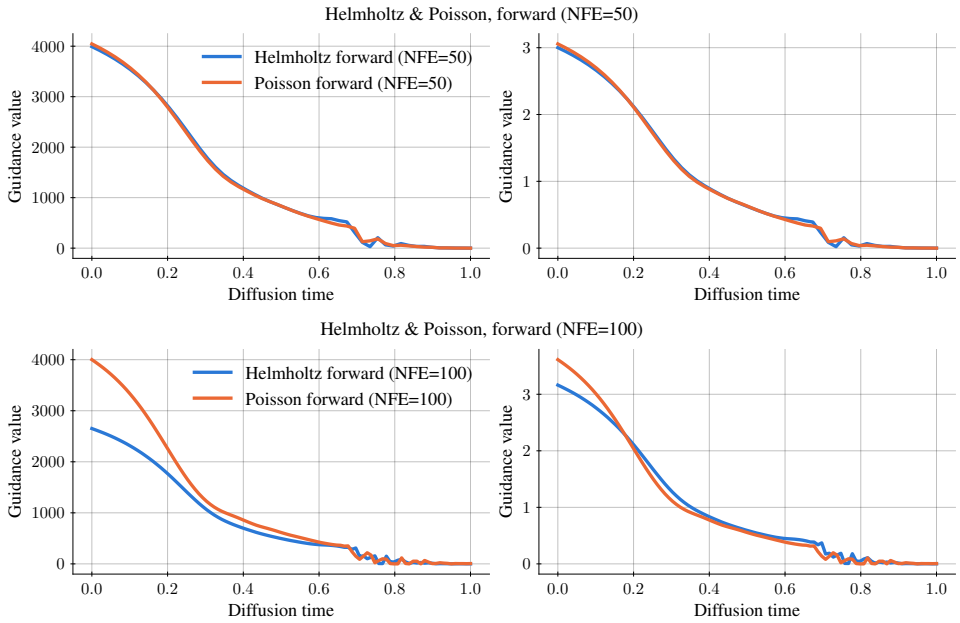


Figure 26: Helmholtz and Poisson, forward. *Left:* raw. *Right:* mean-normalized. *Top:* 50 NFEs. *Bottom:* 100 NFEs.

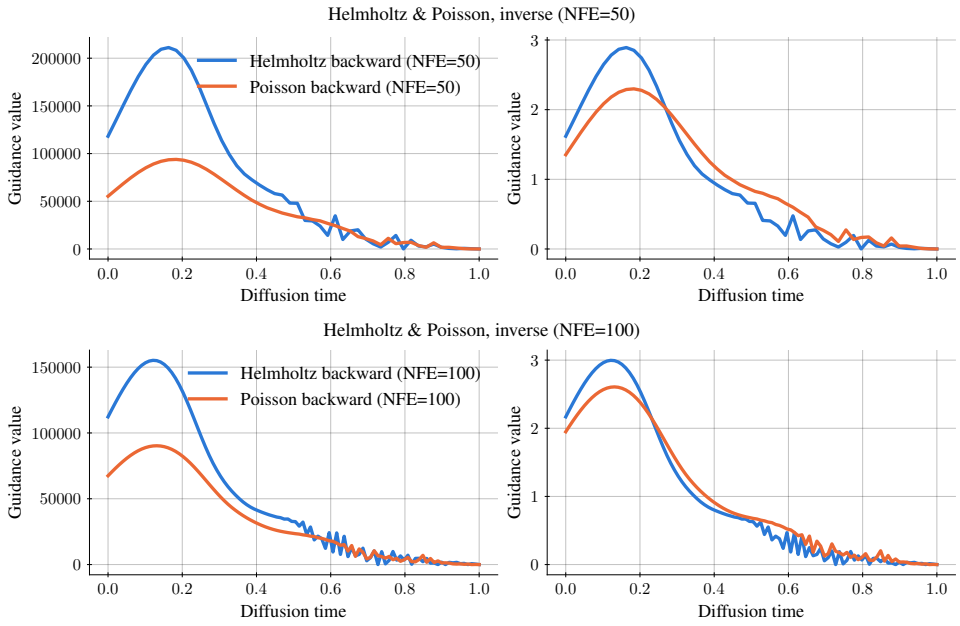


Figure 27: Helmholtz and Poisson, inverse. *Left:* raw. *Right:* mean-normalized. *Top:* 50 NFEs. *Bottom:* 100 NFEs.

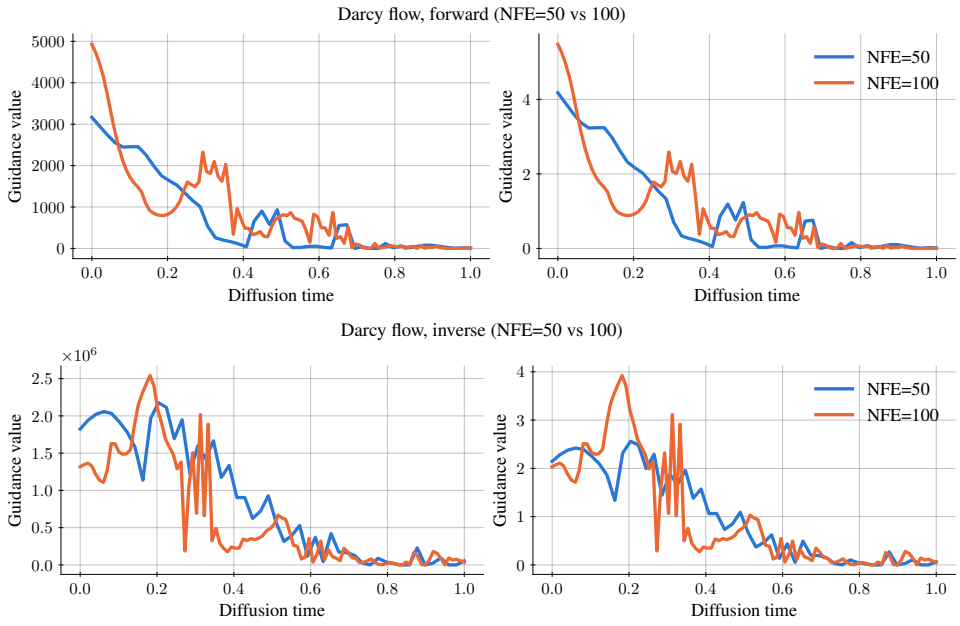


Figure 28: Darcy flow, 50 and 100 NFEs; raw and normalized within each panel. *Top:* forward. *Bottom:* inverse.

H Hyperparameter searches for the baselines

This section gives the hyperparameter-search results for the baselines, grouped by task. The search procedures are described in Appendix B.

H.1 Noisy inverse problems

The selection metric is LPIPS.

FFHQ, 10 NFEs. An additional search at NFEs $= 10$, FFHQ only, for the constant guidance, linear increase, DSG, and DAPS baselines, following the same procedures as above.

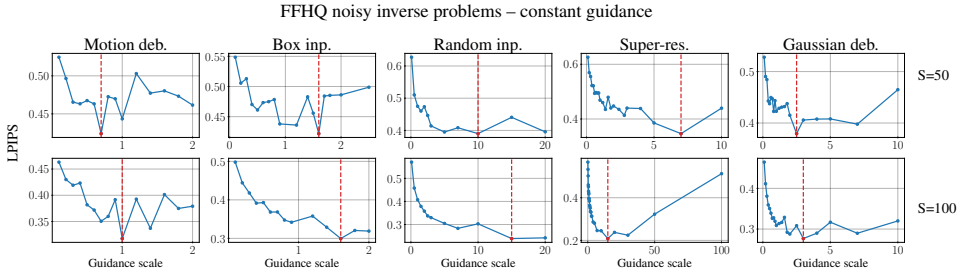


Figure 29: Constant guidance, FFHQ: LPIPS vs. guidance scale; best value in red.

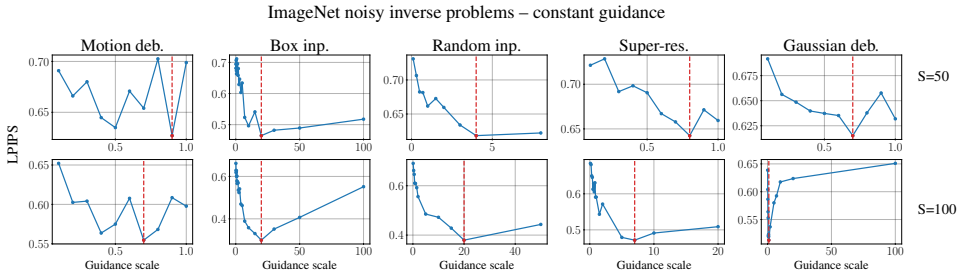


Figure 30: Constant guidance, ImageNet: LPIPS vs. guidance scale; best value in red.

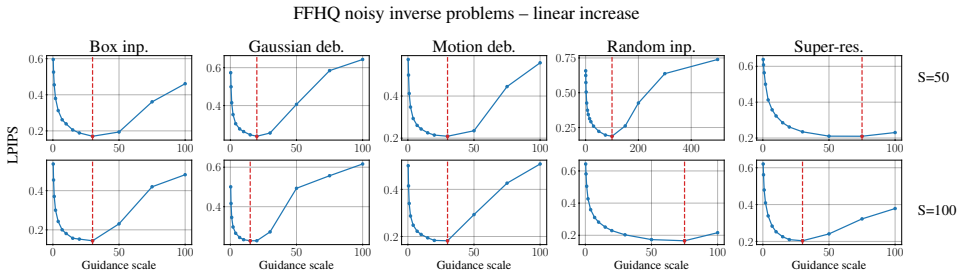


Figure 31: Linear increase, FFHQ: LPIPS vs. guidance scale; best value in red.

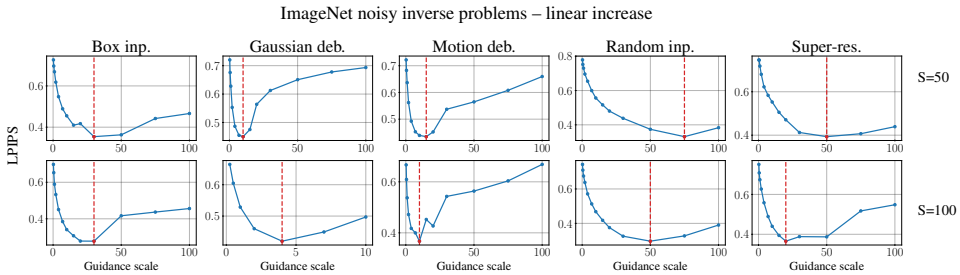


Figure 32: Linear increase, ImageNet: LPIPS vs. guidance scale; best value in red.

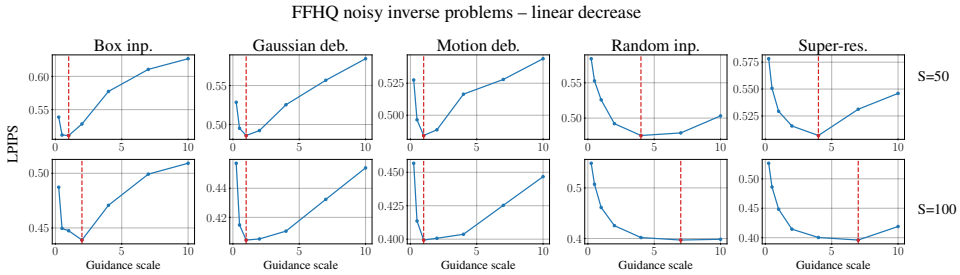


Figure 33: Linear decrease, FFHQ: LPIPS vs. guidance scale; best value in red.

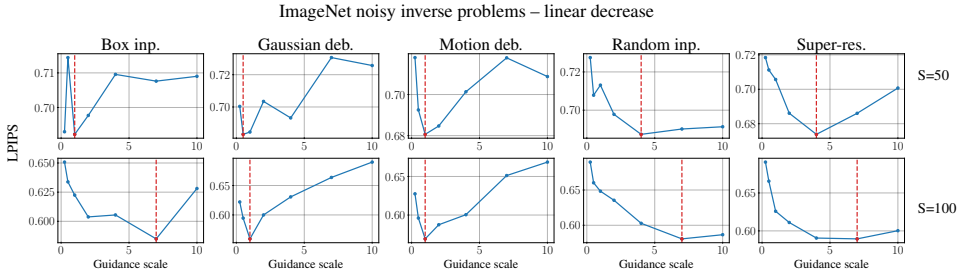


Figure 34: Linear decrease, ImageNet: LPIPS vs. guidance scale; best value in red.

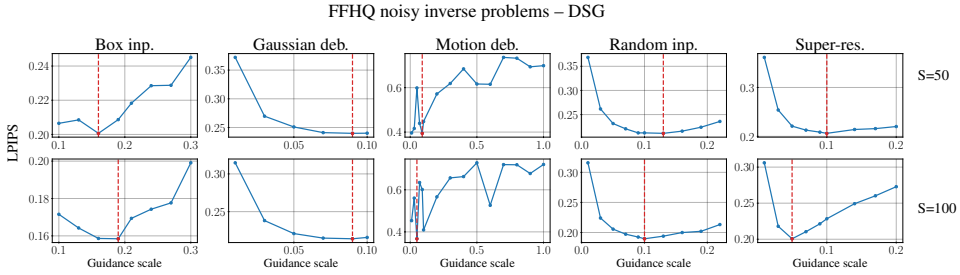


Figure 35: DSG, FFHQ: LPIPS vs. guidance scale; best value in red.

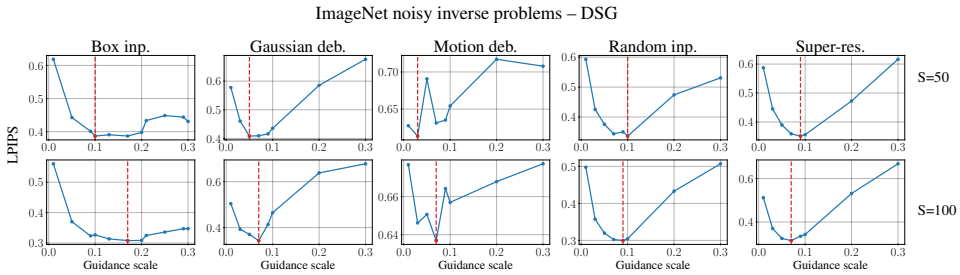


Figure 36: DSG, ImageNet: LPIPS vs. guidance scale; best value in red.

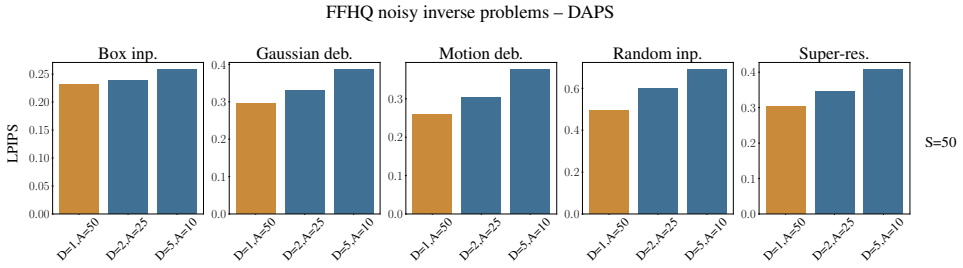


Figure 37: DAPS, FFHQ, 50 NFEs: LPIPS per configuration (D, A); best value in orange.

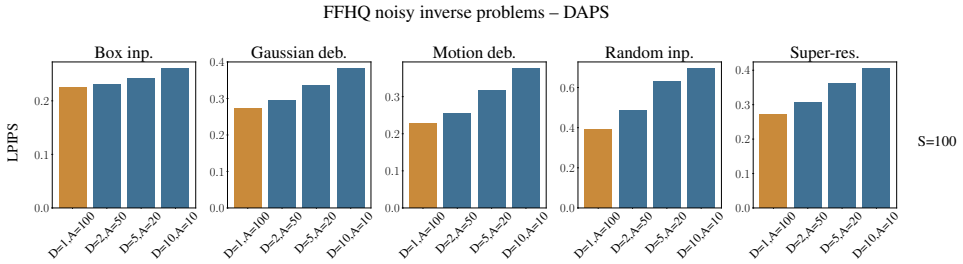


Figure 38: DAPS, FFHQ, 100 NFEs: LPIPS per configuration (D, A); best value in orange.

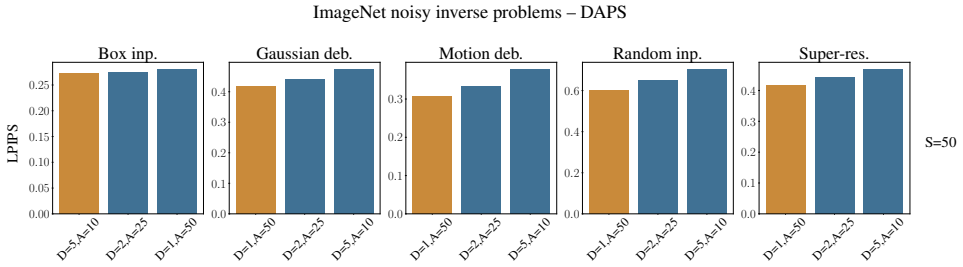


Figure 39: DAPS, ImageNet, 50 NFEs: LPIPS per configuration (D, A); best value in orange.

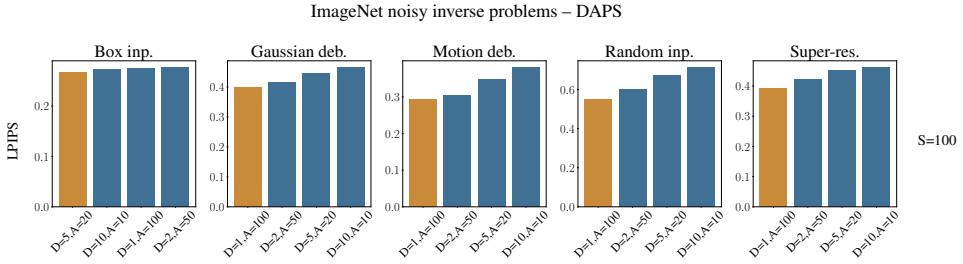


Figure 40: DAPS, ImageNet, 100 NFEs: LPIPS per configuration (D, A); best value in orange.

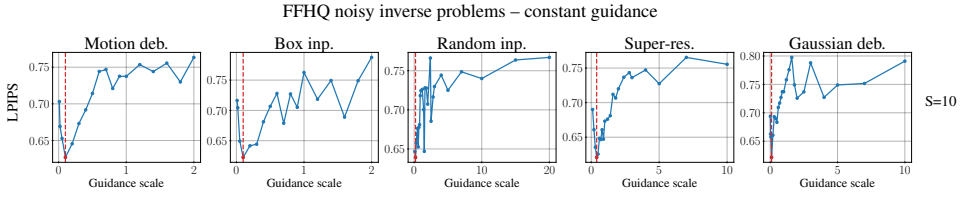


Figure 41: Constant guidance, FFHQ, 10 NFEs: LPIPS vs. guidance scale; best value in red.

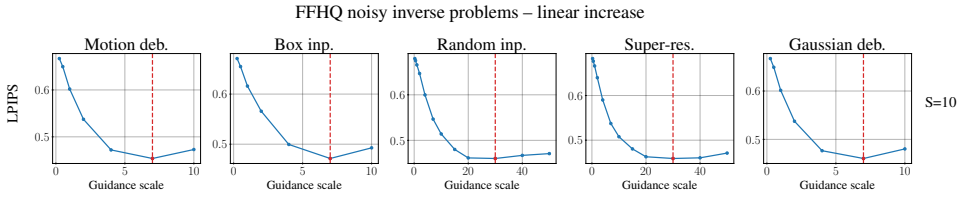


Figure 42: Linear increase, FFHQ, 10 NFEs: LPIPS vs. guidance scale; best value in red.

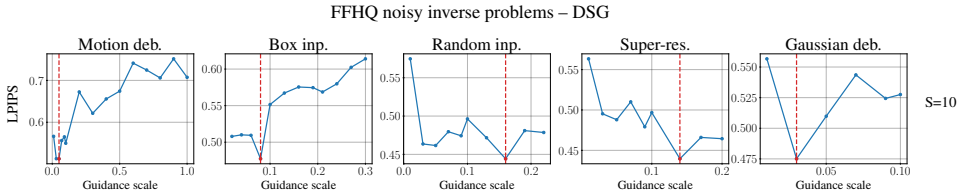


Figure 43: DSG, FFHQ, 10 NFEs: LPIPS vs. guidance scale; best value in red.

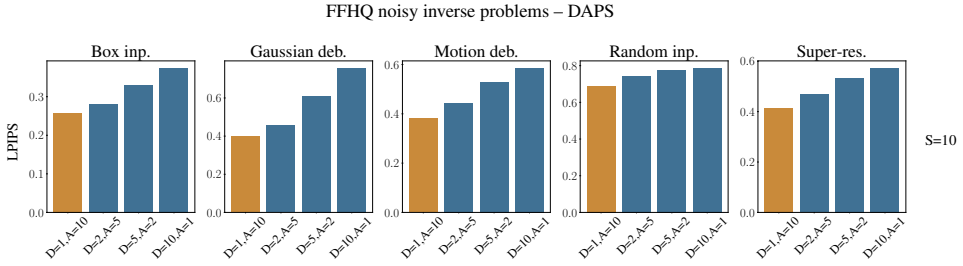


Figure 44: DAPS, FFHQ, 10 NFEs: LPIPS per configuration (D, A); best value in orange.

H.2 Image inpainting

The selection metric is LPIPS.

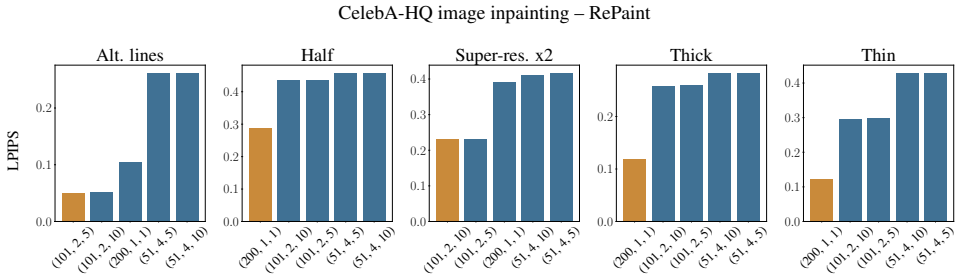


Figure 45: RePaint, CelebA-HQ: LPIPS per configuration (S, N, J); best value in orange.

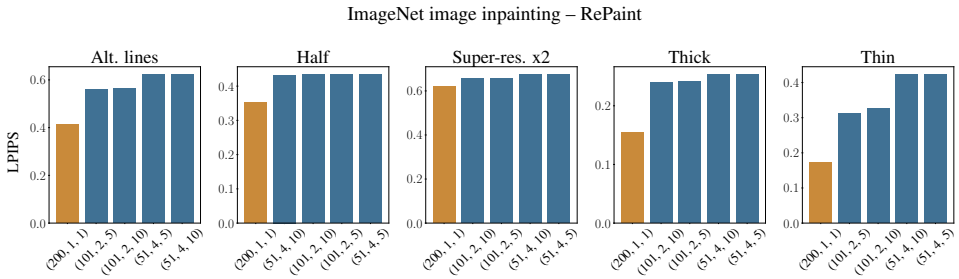


Figure 46: RePaint, ImageNet: LPIPS per configuration (S, N, J); best value in orange.

H.3 Face-ID guidance

The selection metric is the mean FaceID score.

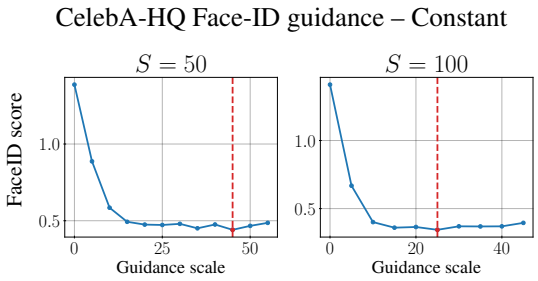


Figure 47: Constant guidance, Face-ID (CelebA-HQ): FaceID score vs. guidance scale; columns: $S = 50, 100$; best value in red.

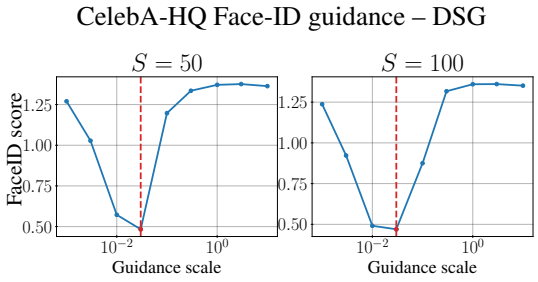


Figure 48: DSG, Face-ID (CelebA-HQ): FaceID score vs. guidance scale; columns: $S = 50, 100$; best value in red.

H.4 PDE problems

The selection metric is the relative error on u for the forward problems, and the relative error on a (Poisson, Helmholtz) or the error rate on a (Darcy) for the inverse problems.

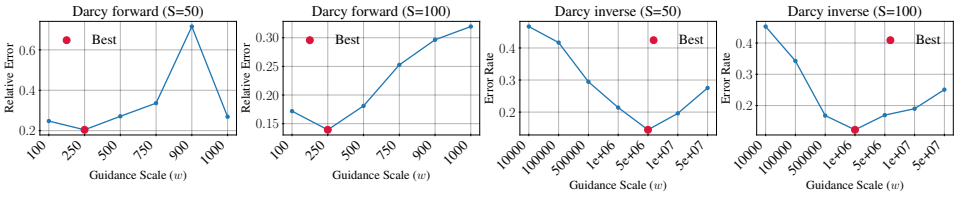


Figure 49: Constant guidance, Darcy: error vs. guidance scale; columns: forward/inverse $\times$ {50, 100 NFEs}; best value in red.

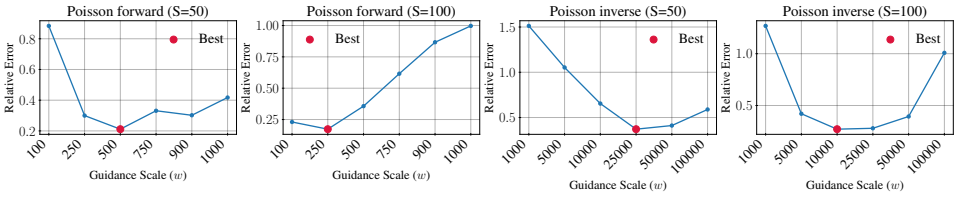


Figure 50: Constant guidance, Poisson: as in Figure 49.

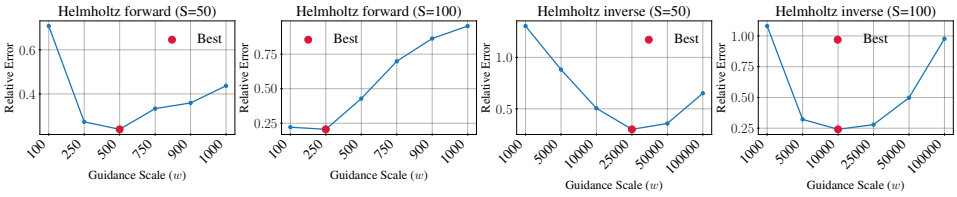


Figure 51: Constant guidance, Helmholtz: as in Figure 49.

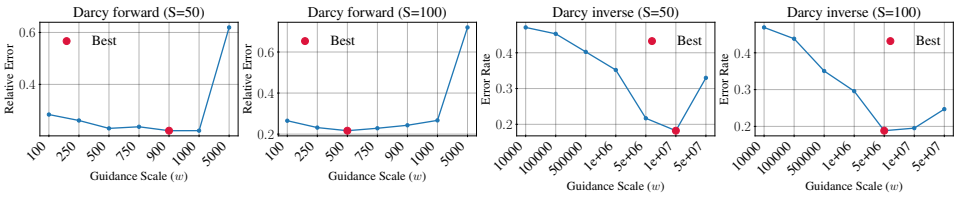


Figure 52: Linear increase, Darcy: error vs. guidance scale; columns: forward/inverse $\times$ {50, 100 NFEs}; best value in red.

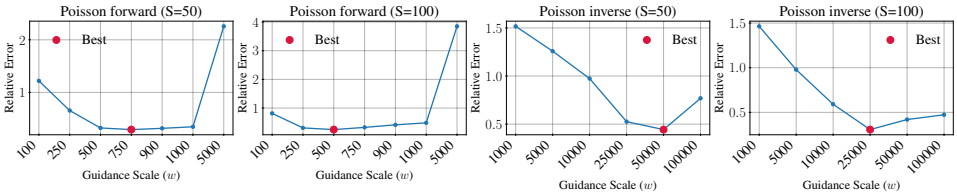


Figure 53: Linear increase, Poisson: as in Figure 52.

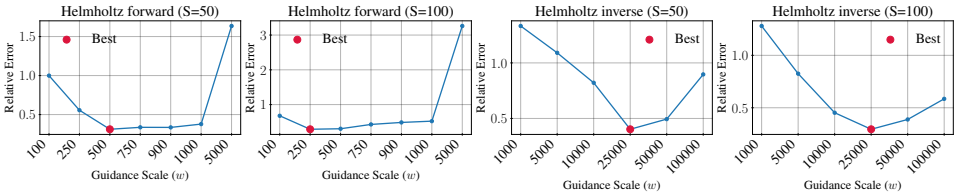


Figure 54: Linear increase, Helmholtz: as in Figure 52.

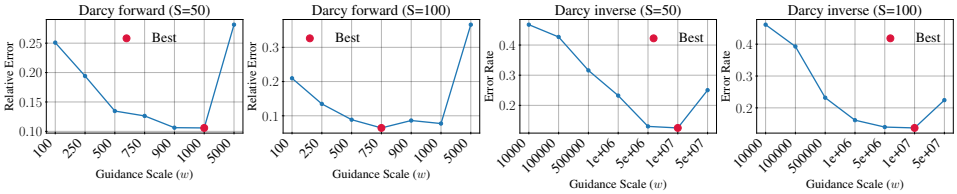


Figure 55: Linear decrease, Darcy: as in Figure 52.

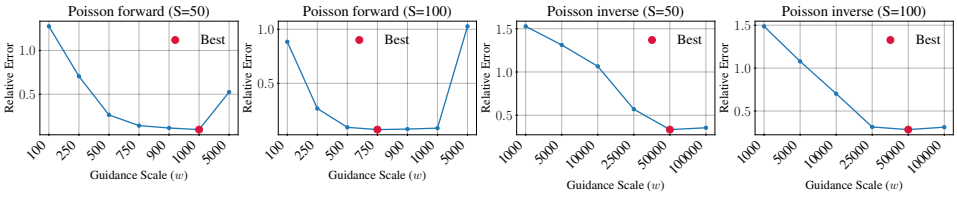


Figure 56: Linear decrease, Poisson: as in Figure 52.

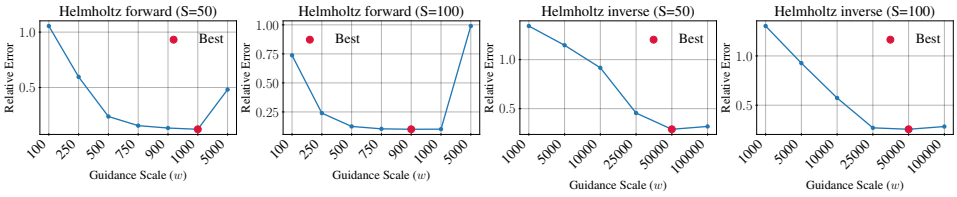


Figure 57: Linear decrease, Helmholtz: as in Figure 52.

I Qualitative results for main experiments (Supplementary material)

This section has been moved to the supplementary material (`iclr2027_supplementary.pdf`) to keep this file under the submission size limit.