INVERSE PROBLEM SOLVING BY WALKING THE SPHERE

Anonymous authors
Paper under double-blind review

ABSTRACT

Latent-space inverse problem solving searches a generator's latent space for a code whose output matches the measurement, making data fit the only objective. With GAN, VAE, and normalizing-flow priors, this recovers signals well, but the set of codes the generator decodes reliably is known only implicitly, so an iterate that drifts away from it can only be pulled back by a penalty that competes with the data fit. Sphere encoders make this set explicit: their latents lie on a hypersphere by construction, and a single normalization returns any iterate to it. We solve inverse problems by projected gradient descent on this sphere, a method we call Walk the Sphere (WTS), and introduce a variant, WTS+Loop, that periodically re-encodes the iterate. Against five diffusion-based solvers tuned under the same budget, WTS or WTS+Loop ranks first, outright or tied, in 13 of 18 settings spanning ImageNet-256 and FFHQ-256 restoration and three scientific problems from InverseBench. The solver needs only three hyperparameters and 300 gradient steps, taking about two seconds per image on FFHQ-256.

1 INTRODUCTION

Inverse problems ask us to recover a signal $\boldsymbol{x}$ from measurements $\boldsymbol{y} = A(\boldsymbol{x}) + \boldsymbol{\eta}$, where A is a known *forward operator*. They arise throughout imaging, the physical sciences, and engineering, and they are typically ill-posed, since many signals are consistent with the same measurement, so a solver must balance fidelity to the measurement against a prior over plausible signals.

Diffusion models are currently the prior of choice. Conditioning the reverse process on $\boldsymbol{y}$ requires the likelihood score $\nabla \log p_t(\boldsymbol{y} \mid \boldsymbol{x}_t)$, which has no closed form at intermediate noise levels because $p_t(\boldsymbol{y} \mid \boldsymbol{x}_t)$ marginalizes over every clean image consistent with $\boldsymbol{x}_t$, and diffusion-based solvers differ in how they approximate it. DPS evaluates the likelihood at the posterior-mean estimate of the clean image (Chung et al., 2023), and LGD averages it over samples of that image (Song et al., 2023b); ΠGDM, DDRM, and DDNM exploit the pseudoinverse or SVD of a linear operator (Song et al., 2023a; Kawar et al., 2022; Wang et al., 2023); DiffPIR and PnP-DM alternate a denoising step with a proximal data-fit step (Zhu et al., 2023; Wu et al., 2024); DAPS decouples the noise schedule and samples the posterior at each level (Zhang et al., 2025); RED-Diff replaces sampling with a variational objective (Mardani et al., 2024); and sequential Monte Carlo methods reweight particles by the likelihood (Cardoso et al., 2024; Dou & Song, 2024). Every approximation introduces a guidance term that must be balanced against the prior score at each timestep (Belardi et al., 2026; Huang, 2026), so every solver carries step-size and noise schedules, and several carry problem-specific hyperparameters besides.

An older approach avoids this balancing act by searching the latent space of a generative model. We write $\boldsymbol{x} = D(\boldsymbol{z})$ for a generator D and minimize $\|A(D(\boldsymbol{z})) - \boldsymbol{y}\|^2$ over $\boldsymbol{z}$, so that the fidelity loss is the only objective. Early instantiations used GANs (Goodfellow et al., 2014; Abdal et al., 2019; 2020; Jalal et al., 2021), VAEs (Kingma & Welling, 2013), and normalizing flows (Dinh et al., 2017; Kingma & Dhariwal, 2018), and two problems held them back. The generators of the time limited reconstruction quality (Asim et al., 2020; Tov et al., 2021), and the data manifold $\{D(\boldsymbol{z})\}$ was defined only implicitly, so an iterate that drifted off it could not be projected back (Zhu et al., 2016). As diffusion-based solvers improved, the latent-space approach was largely abandoned.

The first problem has receded as autoencoders have improved; sphere encoders (Yue et al., 2026) remove the second. They are autoencoders whose latent space is, by construction, the hypersphere

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.

$\mathbb{S}^{d-1}_{\sqrt{d}} = \{\boldsymbol{z} \in \mathbb{R}^d : \|\boldsymbol{z}\|_2 = \sqrt{d}\}$. The encoder output is RMS-normalized onto the sphere and the decoder is trained only on such points, so the data manifold has a closed form and projecting back onto it is a single normalization.

We use this structure to revisit latent-space inverse problem solving. Our method, *Walk the Sphere* (WTS), minimizes the fidelity loss by projected gradient descent on the latent sphere of a sphere autoencoder trained on the problem's data distribution. We also study a variant, WTS+Loop, which additionally passes the iterate through a decode–encode loop every few steps. Our contributions are: (i) we introduce a simple latent-space solver with three hyperparameters (five for WTS+Loop) in which backpropagating through the normalization yields the tangent-space gradient for free (subsection 3.1); (ii) we benchmark it on 18 (problem, difficulty) settings spanning ImageNet-256 super-resolution with off-the-shelf priors, three FFHQ-256 restoration tasks, and inverse scattering, black hole imaging, and multi-coil MRI from InverseBench (Zheng et al., 2025), where WTS or WTS+Loop takes or ties for first in 13 against five diffusion-based solvers at comparable or shorter runtimes (section 4); and (iii) we show by ablation on FFHQ-256 that the sphere is what makes the latent search work: at a fixed latent dimension and tuning protocol, replacing a plain autoencoder or VAE by a sphere-trained decoder and then turning on the hard projection each improve every setting on every metric, and together they are worth 1–4 dB of PSNR (subsection 4.6).

2 Background

2.1 Inverse problems

We consider inverse problems of the form

$$\boldsymbol{y} = A(\boldsymbol{x}^\star) + \boldsymbol{\eta}, \qquad \boldsymbol{\eta} \sim \mathcal{N}(\boldsymbol{0}, \sigma_y^2 \boldsymbol{I}), \tag{1}$$

where $\boldsymbol{x}^\star \in \mathbb{R}^n$ is the unknown signal, $A : \mathbb{R}^n \to \mathbb{R}^m$ is a known forward operator (linear or nonlinear), and $\boldsymbol{y} \in \mathbb{R}^m$ is the observation. Since A is typically rank-deficient or ill-conditioned, many signals fit $\boldsymbol{y}$ equally well and most of them are implausible, so a prior $p(\boldsymbol{x})$ over the data distribution $\mathcal{X}$ is needed to choose among them, giving the maximum a posteriori (MAP) estimate

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \frac{1}{2\sigma_y^2} \|A(\boldsymbol{x}) - \boldsymbol{y}\|^2 - \log p(\boldsymbol{x}). \tag{2}$$

The solvers we compare differ in how the prior enters: diffusion-based solvers take $p(\boldsymbol{x})$ from a pretrained diffusion model and sample from, or approximately optimize, the posterior behind Equation 2; latent-space solvers replace the prior term by a constraint on $\boldsymbol{x}$.

2.2 Inverse problem solving with diffusion models

A diffusion model trained on $\mathcal{X}$ provides the score $\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t)$ at every noise level t and draws a sample by following it through the reverse process. Conditioning on $\boldsymbol{y}$ calls for the posterior score, which Bayes' rule splits into a prior term and a likelihood term,

$$\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t \mid \boldsymbol{y}) = \underbrace{\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t)}_{\text{prior score}} + \underbrace{\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{y} \mid \boldsymbol{x}_t)}_{\text{likelihood score}}. \tag{3}$$

The prior score is what the model provides; the likelihood score is exact only at $t = 0$ and has no closed form for $t > 0$. Guidance methods approximate it from an estimate of the clean image (DPS, LGD, ΠGDM); projection methods enforce consistency with a linear A in its range or spectral space (DDRM, DDNM); splitting methods alternate a prior step with a data-consistency step (DiffPIR, PnP-DM, DAPS); RED-Diff replaces sampling with a variational objective; and sequential Monte Carlo methods reweight particles by the likelihood. In every case each step trades data consistency against the prior through a weight, a step size, or a projection, and these weights and the noise schedule are per-problem hyperparameters (Huang, 2026).

2.3 Inverse problem solving with GANs, VAEs, and normalizing flows

An older line of work puts the prior into a generator $D : \mathcal{Z} \to \mathbb{R}^n$ trained on $\mathcal{X}$ and searches its latent space for a code whose output fits the measurement,

$$\hat{\boldsymbol{z}} = \arg\min_{\boldsymbol{z} \in \mathcal{Z}} \|A(D(\boldsymbol{z})) - \boldsymbol{y}\|^2, \qquad \hat{\boldsymbol{x}} = D(\hat{\boldsymbol{z}}). \tag{4}$$

The constraint $\boldsymbol{x} \in D(\mathcal{Z})$ replaces the prior term of Equation 2, the data fit is the only objective, and gradient descent through D returns a point estimate; some variants add a penalty on $\|\boldsymbol{z}\|$, and invertible generators, whose range is all of $\mathbb{R}^n$, rely on that penalty or on the optimizer's implicit bias alone (Asim et al., 2020). The recipe goes back to traversing the manifold of a pretrained GAN (Zhu et al., 2016); it was adapted to inverse problems with GAN and VAE priors (Bora et al., 2017), scaled to high-resolution GAN inversion (Abdal et al., 2019; 2020), and extended to normalizing flows (Asim et al., 2020). Two limitations held it back. The first was the generators: mode collapse (Goodfellow, 2016; Salimans et al., 2016) and bottleneck blurriness (Larsen et al., 2016) left parts of $\mathcal{X}$ uncovered, and reconstructions inherited this representation error (Asim et al., 2020). The second was that the constraint set was known only implicitly. D decodes well only near the codes it saw in training; for a Gaussian latent these concentrate near a shell of radius $\sqrt{d}$, but the decoder was never constrained to it, and inversion in practice moved to extended spaces such as StyleGAN's $\mathcal{W}+$ (Abdal et al., 2019), whose well-decoded region has no closed form (Tov et al., 2021). A gradient step could carry $\boldsymbol{z}$ away from those codes, and the only remedy was a penalty that trades data fit against staying in range.

2.4 Sphere encoders

Sphere autoencoders (Yue et al., 2026) supply the missing projection. Their latent space is the hypersphere $\mathbb{S}^{d-1}_{\sqrt{d}}$ of radius $\sqrt{d}$. The encoder $E : \mathbb{R}^n \to \mathbb{R}^d$ produces an unconstrained vector $\tilde{\boldsymbol{z}} = E(\boldsymbol{x})$, and an RMS normalization places it on the sphere,

$$\boldsymbol{z} = f(\tilde{\boldsymbol{z}}) := \tilde{\boldsymbol{z}} \Big/ \sqrt{\tfrac{1}{d}\|\tilde{\boldsymbol{z}}\|_2^2 + \varepsilon}, \tag{5}$$

so that $\frac{1}{d}\|\boldsymbol{z}\|_2^2 = 1$ up to ε and the decoder $D : \mathbb{S}^{d-1}_{\sqrt{d}} \to \mathbb{R}^n$ sees only points on the sphere during training. Two consistency losses shape the latent space: $E(D(\boldsymbol{z})) \approx \boldsymbol{z}$, so that E approximately inverts D on the sphere, and $D(\boldsymbol{z}) \approx D(f(\boldsymbol{z} + \sigma_z \boldsymbol{\xi}))$ for $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and a per-sample noise scale $\sigma_z \sim \mathcal{U}[0, \sigma_{\max}]$, so that the decoder is insensitive to small moves along the sphere. This is exactly what Equation 4 was missing: the in-distribution set is now a known surface with a one-line projection onto it, so any drift caused by the fidelity gradient is undone by applying f once, and the iterate stays where the decoder was trained.

3 Walk the Sphere

We solve Equation 1 by projected gradient descent on the latent sphere of a sphere encoder. The problem we solve is

$$\hat{\boldsymbol{z}} = \operatorname*{arg\,min}_{\boldsymbol{z} \in \mathbb{S}^{d-1}_{\sqrt{d}}} \mathcal{L}(D(\boldsymbol{z}), \boldsymbol{y}), \qquad \mathcal{L}(\boldsymbol{x}, \boldsymbol{y}) := \tfrac{1}{2}\|A(\boldsymbol{x}) - \boldsymbol{y}\|^2, \qquad \hat{\boldsymbol{x}} = D(\hat{\boldsymbol{z}}), \tag{6}$$

where D is the decoder of a sphere encoder trained on the data distribution $\mathcal{X}$ and A is the forward operator. The constraint is enforced by the RMS-normalization f of Equation 5. Since f is the map used during training, every iterate lies on the surface the decoder was trained on. WTS is a point estimator: it returns one reconstruction per measurement, deterministic given its initialization (WTS+Loop adds latent noise at each re-encoding). Figure 1 gives an overview.

3.1 Projected Gradient Descent

Each iteration decodes the current latent, computes the fidelity gradient on the decoded image, backpropagates it through the decoder to obtain a gradient on $\boldsymbol{z}$, takes an Adam step in $\mathbb{R}^d$, and projects the result back onto the sphere with f. The only non-standard detail is where we place f. We evaluate the loss with f inside the forward pass,

$$\mathcal{L}(D(f(\boldsymbol{z})), \boldsymbol{y}) \;=\; \tfrac{1}{2}\|A(D(f(\boldsymbol{z}))) - \boldsymbol{y}\|^2,$$

so that backpropagation passes through the projection. This gives us a tangent-space gradient for free. At any point on the sphere, the Jacobian of f is the orthogonal projector onto the tangent space, $\partial f / \partial \boldsymbol{z}\big|_{\|\boldsymbol{z}\|_2 = \sqrt{d}} = \boldsymbol{I} - \boldsymbol{z}\boldsymbol{z}^\top / d$. The chain rule therefore removes the radial component of $\nabla_{\boldsymbol{z}} \mathcal{L}$, and we obtain the Riemannian gradient without an explicit projection step; applying f once more after the Adam step retracts the iterate onto the sphere. Adam's moment estimates are kept in the ambient coordinates of $\mathbb{R}^d$; only the iterate is retracted. Algorithm 1 in Appendix A gives the full procedure.

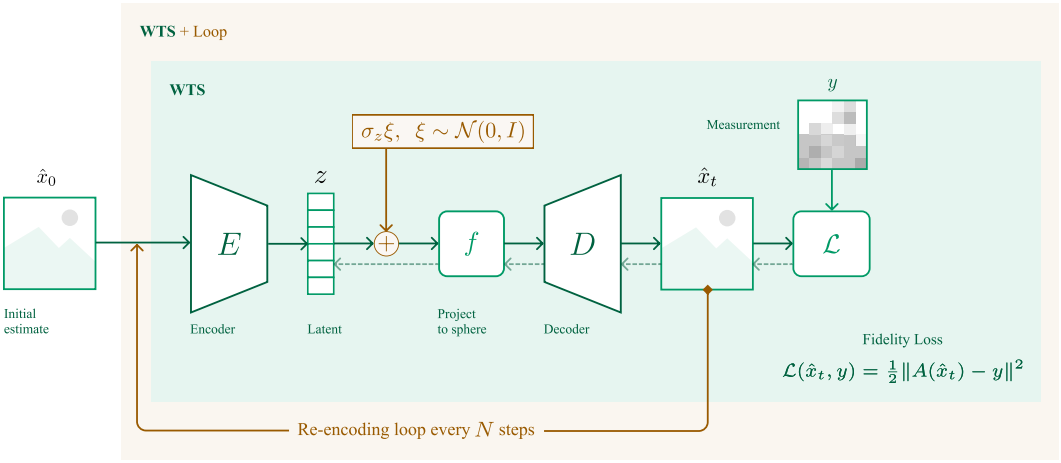

Figure 1: Overview of WTS and WTS+Loop. An initial estimate $\hat{\boldsymbol{x}}_0$ (subsection 3.3) is encoded to a latent $\boldsymbol{z}$. Each WTS step projects $\boldsymbol{z}$ onto the sphere with f, decodes it to $\hat{\boldsymbol{x}}_t$, and evaluates the fidelity loss $\mathcal{L}(\hat{\boldsymbol{x}}_t, \boldsymbol{y})$ against the measurement; the gradient is backpropagated through the decoder and the projection to $\boldsymbol{z}$ (dashed arrows). WTS+Loop additionally re-encodes the current estimate every N steps, perturbing the latent with $\sigma_z\boldsymbol{\xi}$ before the projection (Equation 7).

3.2 Re-encoding Loops (WTS+Loop)

Not every point on the sphere decodes well. The decoder was trained only on points that the encoder produces from real data, and Adam can drift into regions of the sphere where the decoder is less constrained. WTS+Loop addresses this with the sphere encoder's own sampling primitive (Yue et al., 2026). Every N steps we apply one step of what we call a *re-encoding loop*,

$$\boldsymbol{z} \leftarrow E\big(D\big(f(\boldsymbol{z} + \sigma_z\boldsymbol{\xi})\big)\big), \qquad \boldsymbol{\xi} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}), \tag{7}$$

where the perturbation is the latent noise of subsection 2.4 at a fixed scale $\sigma_z = \texttt{noise_scaler} \cdot \sigma_{\max}$. Intuitively, decoding and re-encoding returns the iterate to a point that the encoder actually produces from data, while the perturbation biases it toward higher-density regions of the sphere. Switching the loop off recovers WTS.

3.3 Initialization

For most forward operators a coarse estimate of the signal is available from $\boldsymbol{y}$ alone. We encode this estimate and start Adam from its latent. We use the linear backprojection for inverse scattering, the masked observation for random inpainting, the upsampled observation for super-resolution, the blurred observation for Gaussian deblur, and the MVUE estimate (zero-filled IFFT followed by coil combination) for MRI. Black hole imaging has no obvious measurement-derived estimate, so we start from the mean image of the training set, which uses the prior only and never the test measurement.

4 Experiments

We evaluate WTS and WTS+Loop on ImageNet-256 super-resolution with off-the-shelf priors, on three FFHQ-256 (Karras et al., 2019) restoration tasks (super-resolution, random-mask inpainting, and Gaussian deblurring), and on three scientific inverse problems from InverseBench (Zheng et al., 2025): inverse scattering, black hole imaging, and multi-coil MRI. We benchmark each task at two or three difficulties. For the image tasks we report PSNR, SSIM, LPIPS, and runtime per sample; for the scientific problems, reconstruction quality and a measurement-fidelity metric, using InverseBench's operators, splits, and noise levels verbatim. MRI is the one problem on which neither variant leads, and we report it in Appendix D; all numbers are tabulated in Appendix I.

Baselines. We run five diffusion-based solvers in every setting: DPS (Chung et al., 2023), DAPS (Zhang et al., 2025), DiffPIR (Zhu et al., 2023), PnP-DM (Wu et al., 2024), and RED-

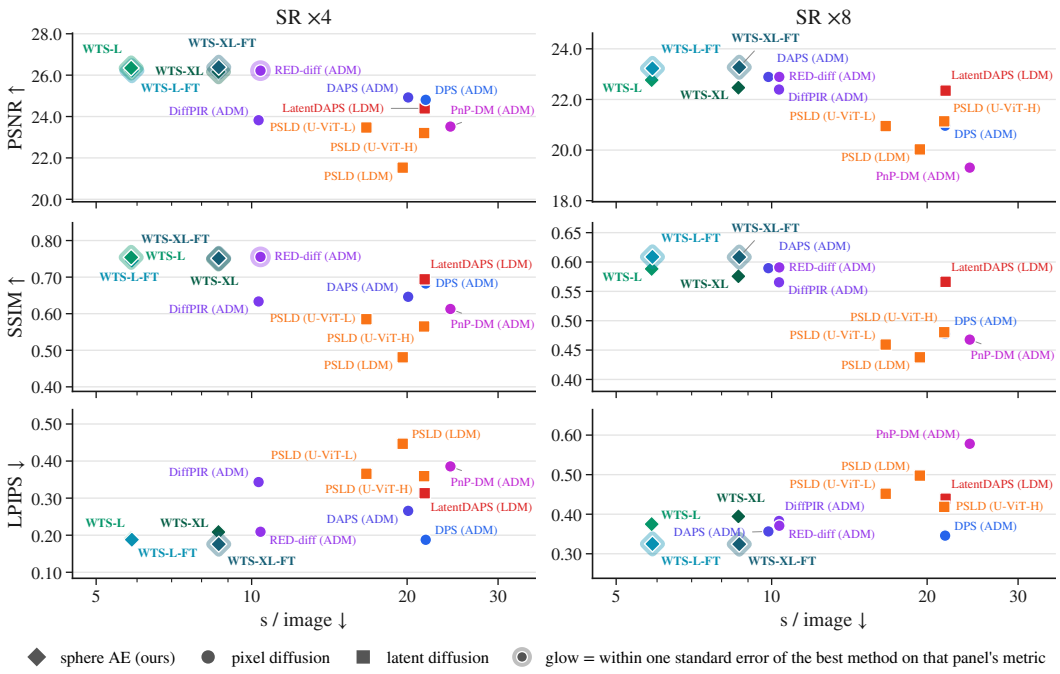

Figure 2: ImageNet-256 super-resolution at $4\times$ (left) and $8\times$ (right): PSNR, SSIM, and LPIPS vs. inference time per image (log scale) on 900 test images. Marker shape gives the prior family: WTS on the released sphere-L and sphere-XL autoencoders and on a $4\times$ more compressed fine-tuned version of each (-FT); the five pixel-space diffusion baselines on an unconditional ADM prior; PSLD and LatentDAPS on three latent-diffusion priors. Where markers overlap, the better one is on top. WTS+Loop arms are in Table 5. Glow as in Figure 3.

Diff (Mardani et al., 2024). Where applicable we include InverseBench's additional diffusion-based and non-diffusion baselines. Baseline numbers on the scientific problems are taken from InverseBench. On the image tasks we tune and evaluate every method ourselves on the same hardware.

Hyperparameter selection. WTS has three solver hyperparameters (the learning rate and the Adam moments β_1 and β_2), and WTS+Loop adds the re-encoding frequency and noise scaler; the number of steps is fixed at 300 in every setting and is not tuned. The diffusion baselines are tuned over the hyperparameters used by Zheng et al. (2025). Every method receives the same budget: Bayesian optimization on a held-out validation split for 5 hours on a single NVIDIA B200, after which the best validation configuration is evaluated once on the test split (see Appendix C). The budget is wall-clock, so slower methods complete fewer trials; the tables in Appendix I list the count for every method.

4.1 ImageNet-256 super-resolution

ImageNet-256 uses off-the-shelf priors only: we run WTS on the released sphere-L and sphere-XL autoencoders (Yue et al., 2026) and on a $4\times$ more compressed version of each (-FT), obtained by narrowing the released latent through a rank-16 adapter and fine-tuning for three epochs (Appendix F). The pixel-space baselines use the unconditional ADM prior (Dhariwal & Nichol, 2021), and the latent-space solvers PSLD (Rout et al., 2023) and LatentDAPS (Zhang et al., 2025) use LDM-4 (Rombach et al., 2022) and the U-ViT-L/2 and U-ViT-H/2 priors (Bao et al., 2023), for fifteen arms in total under one protocol (see Table 5). At $4\times$, the four WTS arms and RED-Diff share the top of the PSNR and SSIM panels and WTS-XL-FT is the single best method on LPIPS, see Figure 2. At $8\times$, the two -FT arms lead alone on all three metrics; they differ from the released models by both a $4\times$ smaller latent and three epochs of fine-tuning, so the gain cannot be attributed to the compression alone, although the FFHQ-256 ladder shows that the solve tolerates it (Appendix E). The WTS arms take 6–9 s per image compared to 10–24 s for the pixel-space baselines, and no latent-diffusion arm leads any panel.

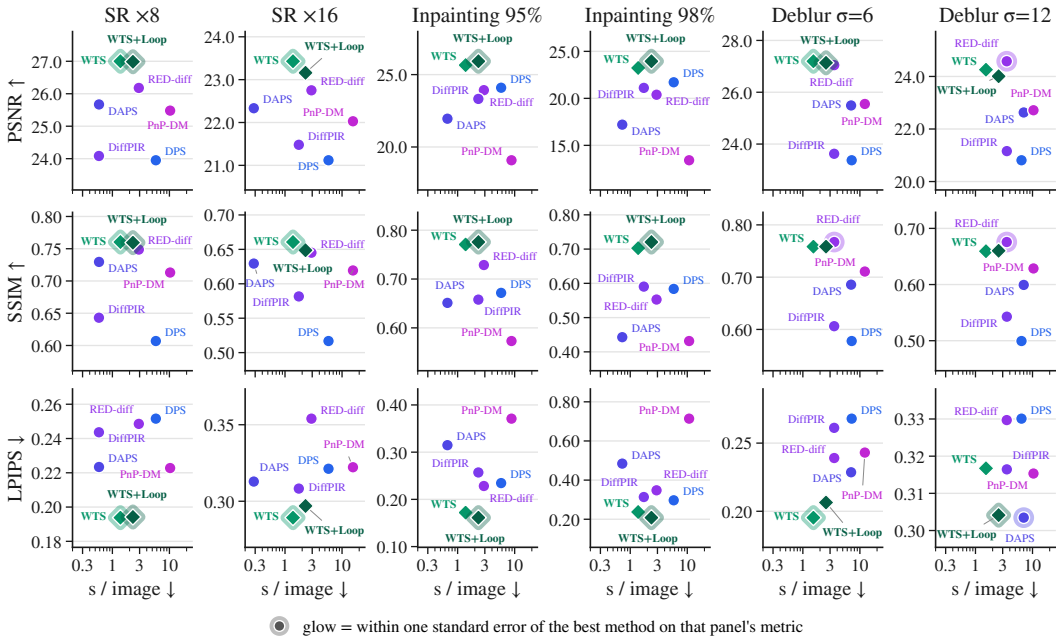

Figure 3: FFHQ-256 image inverse problems: PSNR, SSIM, and LPIPS vs. inference time per sample on six settings, 900 test images each. WTS and WTS+Loop take the same time per image, so their markers are offset slightly. A glow marks every method within one standard error of the best in that panel.



Figure 4: FFHQ-256 reconstructions at the harder difficulty of each task (16× super-resolution, 98% random inpainting, Gaussian deblur at intensity 12). Columns are WTS+Loop, WTS, and the three strongest diffusion baselines on FFHQ-256; DPS and PnP-DM, the weakest on every FFHQ-256 metric, are omitted. Each reconstruction is the single test-split output of the tuned configuration. More in Appendix H.

The re-encoding loop costs the most here, with WTS+Loop below WTS on 22 of 24 cells, so the figure shows WTS.

4.2 FFHQ-256 IMAGE INVERSE PROBLEMS

FFHQ-256 is our controlled comparison: we trained the sphere autoencoder and the diffusion prior behind every baseline on the same 69,000 images (Appendix B), and we tuned all seven solvers

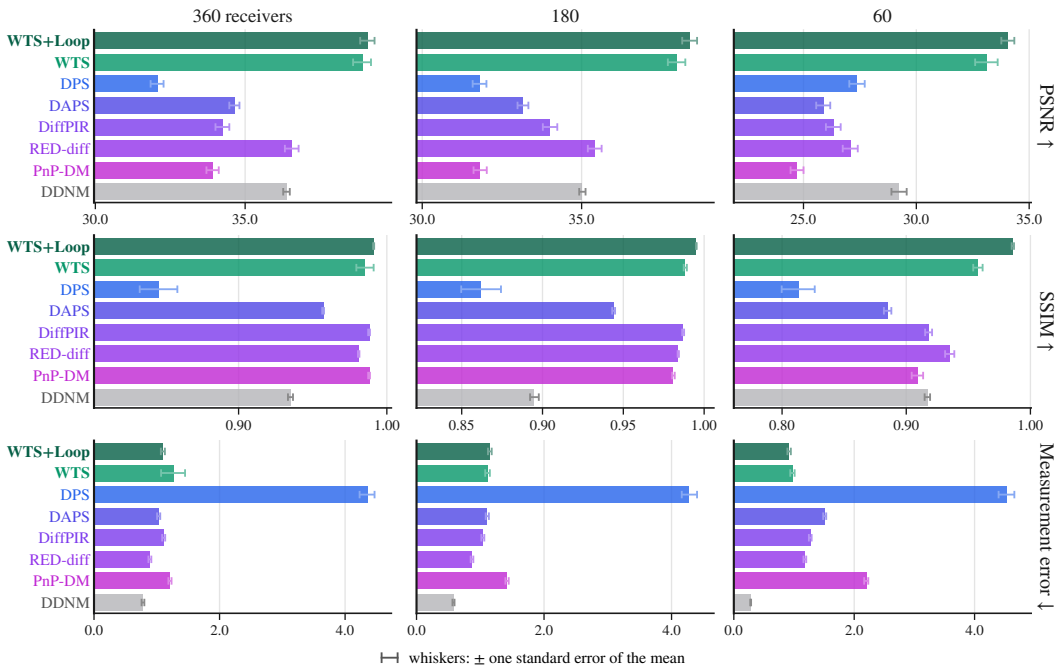

Figure 5: Inverse scattering: PSNR, SSIM, and measurement error at 360, 180, and 60 receivers (easy to hard). Bars are test-set means, whiskers ± one standard error. Baseline rows are the published InverseBench numbers; DDNM is the strongest additional method on this problem, and the rest are in Table 6.

under the same protocol. On super-resolution and inpainting both variants lead on all three metrics at roughly 2 s per image, see Figure 3 and Figure 4. Gaussian deblur is the one task where a baseline leads: both variants lead or tie on LPIPS, but RED-Diff wins SSIM at the lower intensity and both PSNR and SSIM at the higher one. DPS and PnP-DM are 3–9× slower and lower in quality on every metric, while DiffPIR is as fast but lower in quality.

4.3 Inverse Scattering

Both variants lead on PSNR at all three receiver counts, with the largest margin at the hardest setting of 60 receivers, and both reach measurement errors of roughly 1%. On SSIM, WTS+Loop leads at every count and WTS leads at the two harder ones. The methods with lower measurement error (DDNM, FPS, and MCG-diff) reconstruct worse images, DPS trades a 4–5% measurement error for modest PSNR, and FISTA-TV degrades on both as receivers are removed.

4.4 Blackhole Imaging

Both variants start from the training-set mean image (subsection 3.3). WTS+Loop has the highest mean PSNR at 100% and 10% observation time and is within the leading tie group at 3%; WTS ties for first at 100% and 10% and falls below the tie group at 3%. On closure-phase fidelity both variants are on par with the best diffusion baselines, PnP-DM and DAPS, at every ratio. The classical imagers SMILI and EHT-Imaging fit the closure quantities more tightly at the sparser ratios but recover much lower PSNR, as expected for methods without a learned prior.

4.5 Cross-task summary

No single diffusion baseline leads everywhere. RED-Diff is strongest on Gaussian deblur, real-k-space MRI, and ImageNet, DPS on FFHQ inpainting, and PnP-DM on simulated-k-space MRI and the sparsest black hole ratio, while DAPS is near the lead on most panels but collapses on 98% inpainting. Ranking all seven solvers jointly by PSNR (Figure 7), where methods whose mean lies

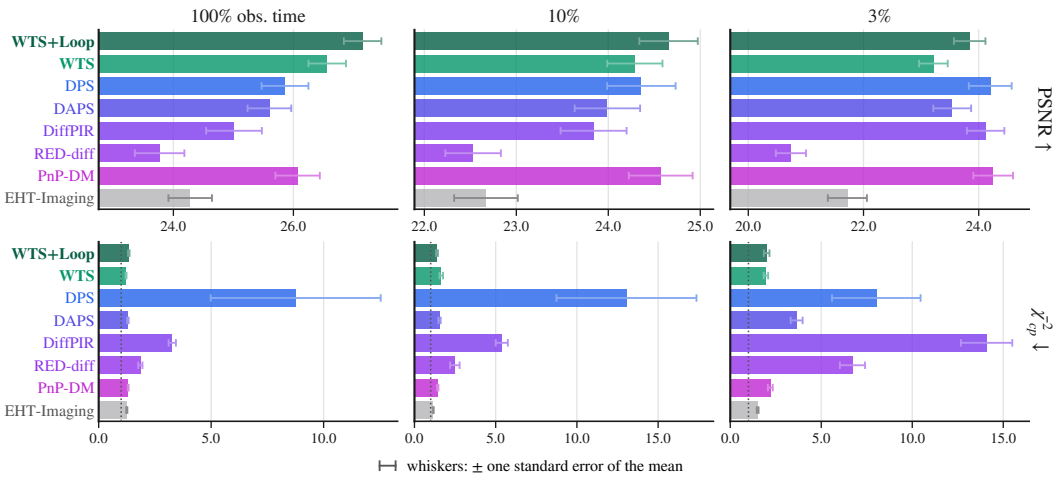

Figure 6: Blackhole imaging: PSNR and reduced closure-phase $\bar{\chi}^2_{cp}$ (ideally 1) at 100%, 10%, and 3% observation time (easy to hard). Bars are test-set means, whiskers ± one standard error. Baseline rows are the published InverseBench numbers; EHT-Imaging is the strongest additional method on this problem, and closure amplitude, blur PSNR, and the remaining baselines are in Table 7.

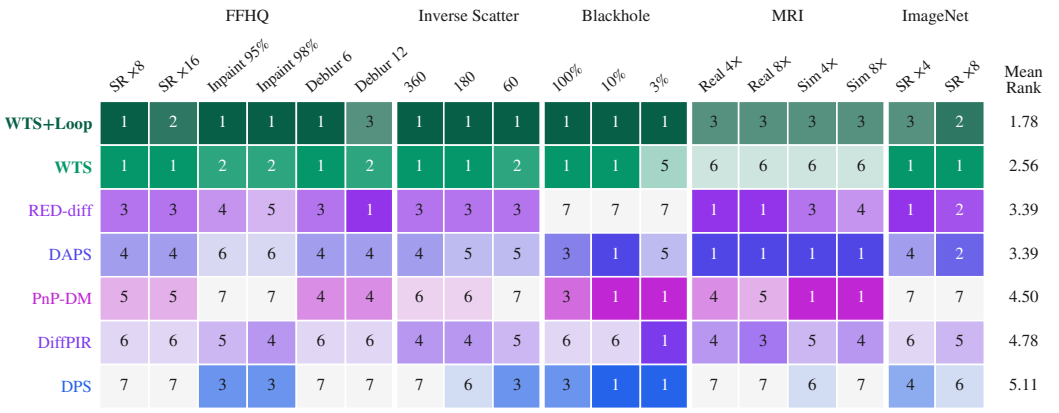

Figure 7: Cross-task PSNR ranks. The seven solvers present in every evaluation, ranked jointly by test PSNR within each of the 18 (problem, difficulty) settings (1 = best); on ImageNet-256 each variant is represented by its sphere model with the highest validation PSNR. Methods whose mean ± one standard error overlaps the best remaining method share its rank, and the next rank skips past them. The right-hand column is the mean rank.

within one standard error of the best share its rank, one of our two variants is first, alone or tied, in 13 of the 18 settings; the exceptions are the higher-intensity deblur and the four MRI splits.

4.6 Ablations

We add one ingredient at a time on the six FFHQ-256 settings (Figure 8). Each of the three steps up to WTS improves every setting on every metric. A VAE improves on a plain autoencoder, by the most on inpainting, where the plain autoencoder fails outright; a sphere-trained decoder with the constraint off improves further; and turning on the hard sphere projection adds another $+0.3$ to $+1.5$ dB PSNR, again most on inpainting, with the LPIPS improvement coming with it. None of this is explained by decoder quality: the VAE reconstructs the test split within 0.5 dB of the sphere autoencoder, yet its solver rows trail WTS by 1–4 dB. The re-encoding loop helps where the operator leaves the most freedom. It adds $+0.3$ and $+0.7$ dB PSNR and lowers LPIPS on the two inpainting settings and ties or loses up to 0.3 dB on the four low-pass tasks; beyond FFHQ, it helps on every scientific setting and

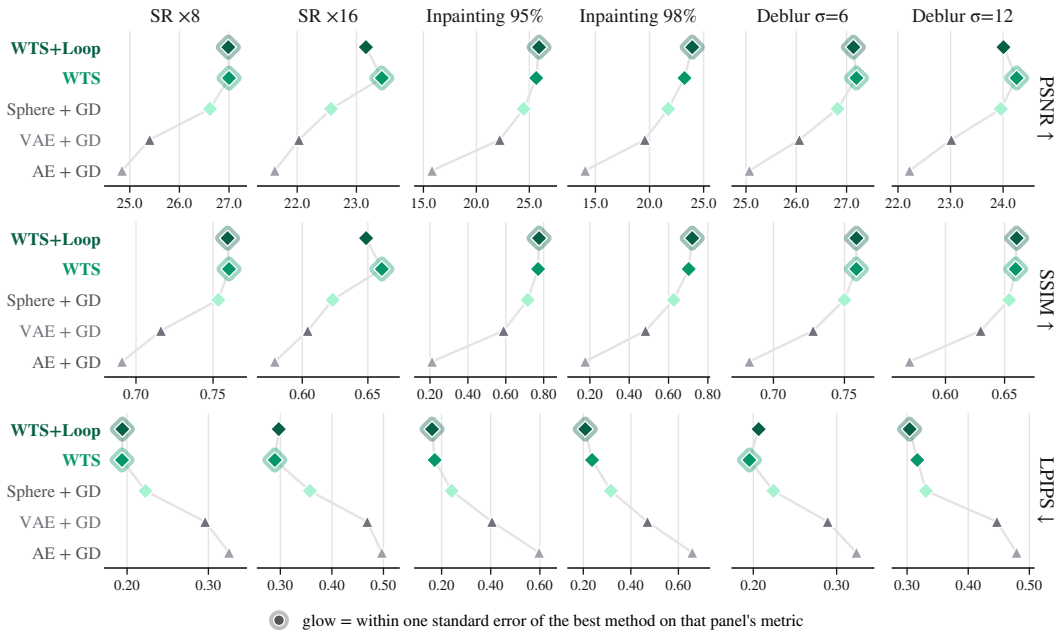

Figure 8: What each ingredient buys. PSNR, SSIM, and LPIPS on the six FFHQ-256 settings for five solvers sharing one latent dimension and one tuning protocol, bottom to top: latent gradient descent in a plain autoencoder (AE + GD), in a VAE (VAE + GD), and in the sphere autoencoder with the constraint off (Sphere + GD); with the hard sphere projection (WTS); and with the re-encoding loop (WTS+Loop). Glow as in Figure 3.

hurts on ImageNet. Compressing the autoencoder's latent by $16\times$ lowers its own reconstruction by 7 dB while the solve barely moves, so the solver is not limited by the prior's fidelity (Appendix E).

5 Limitations and Conclusion

WTS requires a sphere autoencoder trained on the data distribution. It is cheaper to train than a diffusion prior on the same data (Appendix B), but must be trained once per distribution. WTS returns a point estimate with no posterior samples or uncertainty, and the measurement noise σ_y does not enter its objective. Finally, WTS does not lead on multi-coil MRI, where WTS+Loop is third among the zero-shot methods on every split (Appendix D), or on the higher-intensity Gaussian deblur, where RED-Diff leads. Appendix G examines when the measurement grounds the prior: on FFHQ-256 inpainting, the arrangement of the missing pixels rather than their number sets how much of the prior's freedom the observed pixels check, and this grounding predicts how far WTS lands below its own autoencoder's reconstruction (per-image Spearman correlation of -0.9). A large contiguous hole leaves the prior unchecked, and there a solver that samples from a generative prior has the advantage.

We revisited latent-space inverse problem solving with one structural change: the latent space is the hypersphere of a sphere encoder, so an off-manifold iterate can be corrected by a single normalization. The resulting solver runs roughly 300 latent gradient steps in about two seconds per image on FFHQ-256 and leads or ties the strongest diffusion-based solvers on 13 of 18 settings, and an ablation attributes its advantage over unconstrained latent search to the sphere.

AI use statement

We used generative AI tools to assist with writing and editing the text of this paper and with writing code for experiments, analysis, and figures. All AI-assisted text and code was reviewed by the authors, who take full responsibility for the content of this paper.

Ethics statement

Inverse problem solving recovers physical quantities from indirect or incomplete measurements, and its applications in science and medicine are potentially beneficial. Generative models carry broader risks, but we do not believe our work is especially susceptible to, or facilitating of, misuse. All data are public research benchmarks used under their terms, and no human subjects were involved.

Reproducibility statement

All datasets, forward operators, data splits, and noise levels are those of InverseBench, and the FFHQ-256 and ImageNet-256 data are publicly available. The algorithm is given in Appendix A; the trained models and their compute in Appendix B; the tuning protocol, budget, validation splits, and selected hyperparameters in Appendix C; and every reported number, with its standard deviation, in Appendix I. Code, configuration files, and trained checkpoints will be released upon publication.

References

Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan: How to embed images into the stylegan latent space? In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan++: How to edit the embedded images? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2623–2631, 2019.

Muhammad Asim, Mara Daniels, Oscar Leong, Ali Ahmed, and Paul Hand. Invertible generative models for inverse problems: mitigating representation error and dataset bias. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 399–409. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/asim20a.html.

Fan Bao, Shen Nie, Kaiwen Xue, Yue Cao, Chongxuan Li, Hang Su, and Jun Zhu. All are worth words: A vit backbone for diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22669–22679, June 2023.

Christian Belardi, Justin Lovelace, Kilian Q Weinberger, and Carla P Gomes. Adaptive moments are surprisingly effective for plug-and-play diffusion sampling. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=qYDObsHldZ.

Ashish Bora, Ajil Jalal, Eric Price, and Alexandros G. Dimakis. Compressed sensing using generative models. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 537–546. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/bora17a.html.

Gabriel Cardoso, Yazid Janati el idrissi, Sylvain Le Corff, and Eric Moulines. Monte carlo guided denoising diffusion models for bayesian linear inverse problems. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=nHESwXvxWK.

Hyungjin Chung, Jeongsol Kim, Michael Thompson Mccann, Marc Louis Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=OnD9zGAGT0k.

Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat GANs on image synthesis. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=AAWuCvzaVt.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=HkpbnH9lx.

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

Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Anbu Huang. Navigating the manifold — a geometric perspective on diffusion-based inverse problems. In *ICLR Blogposts 2026*, 2026. URL https://iclr-blogposts.github.io/2026/blog/2026/diffusion-inverse-problems/. https://iclr-blogposts.github.io/2026/blog/2026/diffusion-inverse-problems/.

Ajil Jalal, Marius Arvinte, Giannis Daras, Eric Price, Alexandros G Dimakis, and Jon Tamir. Robust compressed sensing mri with deep generative priors. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 14938–14954. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/7d6044e95a16761171b130dcb476a43e-Paper.pdf.

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

Bahjat Kawar, Michael Elad, Stefano Ermon, and Jiaming Song. Denoising diffusion restoration models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=kxXvopt9pWK.

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

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/d139db6a236200b21cc7f752979132d0-Paper.pdf.

Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In Maria Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1558–1566, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/larsen16.html.

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

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 10674–10685. ieee, 2022.

Litu Rout, Negin Raoof, Giannis Daras, Constantine Caramanis, Alex Dimakis, and Sanjay Shakkottai. Solving linear inverse problems provably via posterior sampling with latent diffusion models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 49960–49990. Curran Associates, Inc., 2023. doi: 10.52202/075280-2174. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/9c70cfa2e7d9328c649c94d50cbf8faf-Paper-Conference.pdf.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

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

Jiaming Song, Qinsheng Zhang, Hongxu Yin, Morteza Mardani, Ming-Yu Liu, Jan Kautz, Yongxin Chen, and Arash Vahdat. Loss-guided diffusion models for plug-and-play controllable generation. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 32483–32498. PMLR, 23–29 Jul 2023b. URL https://proceedings.mlr.press/v202/song23k.html.

Omer Tov, Yuval Alaluf, Yotam Nitzan, Or Patashnik, and Daniel Cohen-Or. Designing an encoder for stylegan image manipulation. *ACM Transactions on Graphics (TOG)*, 40(4):1–14, 2021.

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

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

Kaiyu Yue, Menglin Jia, Ji Hou, and Tom Goldstein. Image generation with a sphere encoder. *arXiv preprint arXiv:2602.15030*, 2026.

Bingliang Zhang, Wenda Chu, Julius Berner, Chenlin Meng, Anima Anandkumar, and Yang Song. Improving diffusion inverse problem solving with decoupled noise annealing. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 20895–20905, 2025.

Hongkai Zheng, Wenda Chu, Bingliang Zhang, Zihui Wu, Austin Wang, Berthy Feng, Caifeng Zou, Yu Sun, Nikola Borislavov Kovachki, Zachary E Ross, Katherine Bouman, and Yisong Yue. Inversebench: Benchmarking plug-and-play diffusion priors for inverse problems in physical sciences. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=U3PBITXNG6.

Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. In *European conference on computer vision*, pp. 597–613. Springer, 2016.

Yuanzhi Zhu, Kai Zhang, Jingyun Liang, Jiezhang Cao, Bihan Wen, Radu Timofte, and Luc Van Gool. Denoising diffusion models for plug-and-play image restoration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1219–1229, 2023.

A THE WTS ALGORITHM

Algorithm 1 Walk The Sphere

```
# y                 observed measurement
# forward_operator  the operator A; .grad(x, y) is the gradient of the
#                   data-fit loss with respect to x
# encode, decode    sphere-encoder encoder and decoder
# f                 RMS normalization onto the sphere
# sigma_max         largest latent-noise scale used in AE training
# reencode_freq     None: WTS;  N: WTS+Loop, re-encoding every N steps
# noise_scaler      WTS+Loop perturbation scale, relative to sigma_max

def walk_the_sphere(y, n, lr, beta1, beta2,
                    reencode_freq=None, noise_scaler=0.0):
    # operator-specific warm start (subsection 3.3)
    x_hat = initial_guess(y)
    z = f(encode(x_hat))
    optimizer = Adam([z], lr=lr, betas=(beta1, beta2))

    for i in range(1, n + 1):
        # gradient step; backprop passes through decode and f
        optimizer.zero_grad()
        x_hat = decode(f(z))
        x_hat.backward(forward_operator.grad(x_hat, y))
        optimizer.step()
        z = f(z)  # retract onto the sphere

        # WTS+Loop only: perturb, then decode-encode retraction
        if reencode_freq and i % reencode_freq == 0 and i < n:
            eps = randn_like(z)
            z = f(z + noise_scaler * sigma_max * eps)
            z = f(encode(decode(z)))

    return decode(z)
```

B TRAINING AND INFERENCE COMPUTE

Models. We train one sphere autoencoder per problem; one autoencoder backs all FFHQ-256 tasks. The architecture is the upstream sphere-encoder ViT (Yue et al., 2026) adapted to the native channel count of each problem (1 for inverse scattering and blackhole, 2 for MRI, 3 for FFHQ-256 and ImageNet-256): a thin wrapper swaps the three channel-bearing layers of the upstream model (the encoder patch embedding and two decoder pixel-head layers) and re-runs the upstream weight initialization on them, leaving every other architectural and training detail unchanged. Each problem applies a per-channel affine normalization $\boldsymbol{x} \mapsto (\boldsymbol{x} - \mu)/\sigma$ at the dataset boundary, with (μ, σ) chosen per problem from percentiles of the training set rather than by z-scoring, which would amplify the tails of heavy-tailed data such as MRI; the solver undoes the transform on the decoder output before applying A, so the forward operator always sees data in its native units.

Training. Every model we trained ourselves ran on one node of the same cluster (NVIDIA B200 GPUs, eight per node); Table 1 lists each with the GPUs it used and the wall-clock and GPU-hours it consumed, read from the cluster's job accounting. On FFHQ-256 both priors were trained from scratch on the same 69,000-image train split: the sphere autoencoder (`base` preset, $12\times$ compression, 1000 epochs) and the VP-precond ADM diffusion prior (1M optimization steps) shared by every diffusion baseline there (DPS, DAPS, DiffPIR, RED-Diff, PnP-DM). The diffusion prior used $2.4\times$ the autoencoder's GPU-hours. Three further FFHQ-256 sphere autoencoders at $1.5\times$, $6\times$, and $24\times$ compression form the ladder of Appendix E, and a plain autoencoder and a VAE at $12\times$, trained with the same recipe minus the sphere constraint and its consistency losses, are the lower rungs of the ablation in subsection 4.6. Each scientific problem has its own sphere autoencoder (`small` preset) trained on the InverseBench training split. The MRI autoencoder adds an SSIM term (weight 2) to the

reconstruction loss; its own encode–decode reconstruction of the test split reaches 33.98 dB PSNR and 0.849 SSIM, above every zero-shot method in Table 8, and the solver lands about 4 dB below it. On ImageNet-256 we train nothing from scratch: WTS-L and WTS-XL run on the sphere-L and sphere-XL checkpoints released by Yue et al. (2026), and the two -FT arms narrow each released latent from $3\times$ to $12\times$ compression with a rank-16 adapter and then fine-tune the whole autoencoder for three epochs on the ImageNet-1k train split at the release's global batch of 256 (Appendix F). The ADM, LDM-4, and U-ViT priors of the ImageNet baselines are used as released. Hyperparameter tuning (Appendix C) adds 136 studies of 5 GPU-hours each, 680 B200-hours in total: 48 studies for our two variants in the reported settings, 36 for the compression and ablation ladders on FFHQ-256, and 52 for the baselines we tuned ourselves on FFHQ-256 and ImageNet-256.

Table 1: Models trained for this paper, with wall-clock hours and GPU-hours from the cluster's job accounting; every job ran on one node of NVIDIA B200 GPUs. The first two rows back the FFHQ-256 results; the next five back the compression ladder (Appendix E) and the ablation (subsection 4.6). -FT rows fine-tune the released sphere-L / sphere-XL checkpoints through a rank-16 adapter that narrows the latent to $12\times$ compression. The `small` preset has 55–75M parameters and `base` about 180M, depending on input resolution and channel count.

Model	Data	Training	GPUs	Hours	GPU-h
Sphere AE, `base`, $12\times$	FFHQ-256	1000 ep.	8	35.3	282
ADM diffusion prior (VP)	FFHQ-256	1M steps	8	83.3	667
Sphere AE, `base`, $1.5\times$	FFHQ-256	1000 ep.	8	35.5	284
Sphere AE, `base`, $6\times$	FFHQ-256	1000 ep.	8	35.3	282
Sphere AE, `base`, $24\times$	FFHQ-256	1000 ep.	8	35.3	282
Plain AE, `base`, $12\times$	FFHQ-256	1000 ep.	8	18.5	148
VAE, `base`, $12\times$	FFHQ-256	1000 ep.	8	18.5	148
Sphere AE, `small`, $3\times$	Inverse scattering	2000 ep.	4	9.4	38
Sphere AE, `small`, $3\times$, +SSIM loss	MRI	800 ep.	8	8.6	69
Sphere AE, `small`, $24\times$	Blackhole	400 ep.	8	1.7	13
Sphere-L-FT	ImageNet-256	3 ep.	4	14.3	57
Sphere-XL-FT	ImageNet-256	3 ep.	8	8.3	66

Inference. A WTS solve is 300 latent gradient steps, each one decoder forward and one backward through the decoder plus the forward operator and its adjoint; WTS+Loop adds one decode–encode pass at each re-encoding event, at most 15 per solve at the highest tuned frequency of $1/16$. The cost is dominated by the decoder and does not depend on a noise schedule. The tables of Appendix I give the time per sample of every method we ran, measured on one B200 and amortized over the batch each solver processes at once; we did not rerun the science baselines, so no comparable runtime exists for them.

C Hyperparameter selection

Every method we ran, ours and the baselines, is tuned with the same protocol: a Bayesian-optimization study with a Tree-structured Parzen Estimator (TPE) surrogate (Akiba et al., 2019) on a held-out validation split, with a fixed budget of 5 hours on one NVIDIA B200 and no cap on the number of trials. No new trial starts after the 5 hours; a trial in flight at that point runs to completion. Each trial solves the whole validation split once and is scored by mean PSNR; the configuration with the highest validation PSNR is then run once on the disjoint test split, and those test numbers are what we report. Each (method, task) pair has its own study, so tuning effort never leaks across methods or difficulty levels. The validation and test splits are 100 and 900 images on FFHQ-256 and on ImageNet-256 (a seeded 1000-image draw from the ILSVRC-2012 validation set), and the InverseBench splits on the scientific problems: 3 and 100 objects for inverse scattering, 5 and 100 images for blackhole imaging, and 6 and 96 cases for MRI. For WTS the search space is the learning rate and Adam's β_1 and β_2; WTS+Loop adds the re-encoding frequency and the noise scaler; for the diffusion baselines on FFHQ-256 and ImageNet-256 it is the set of hyperparameters Zheng et al. (2025) tune for each method. Baseline rows on the scientific problems are the published InverseBench numbers, tuned by its authors under their own protocol; we do not re-tune them.

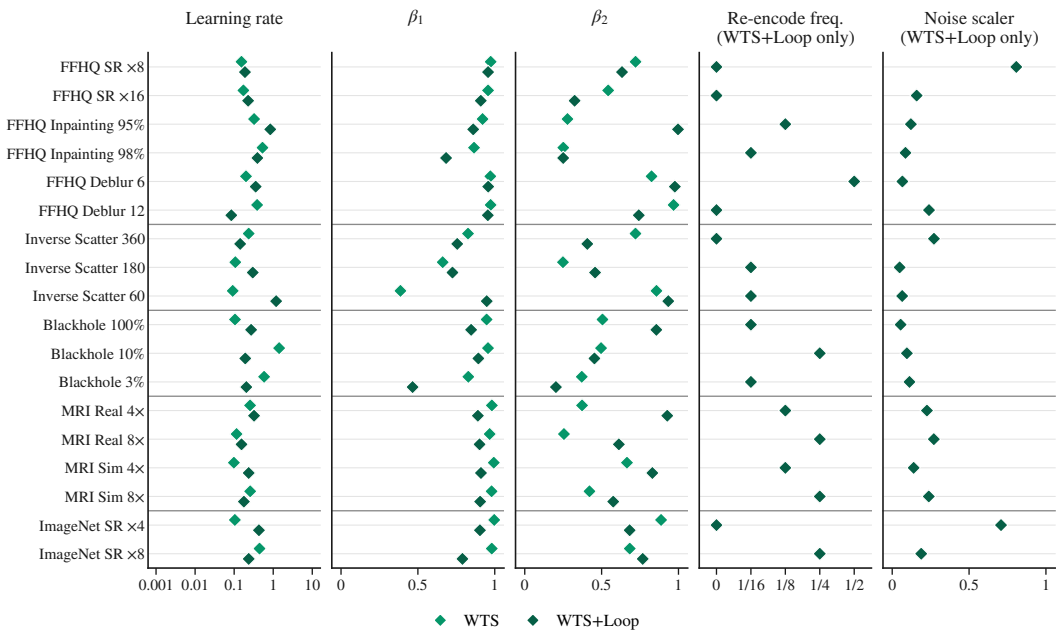

Figure 9: Tuned solver hyperparameters of WTS and WTS+Loop in every setting: the validation-argmax configuration that is applied to the test split. Each row is one (problem, difficulty) setting, with WTS drawn just above the row line and WTS+Loop just below; the re-encoding frequency and noise scaler exist only for WTS+Loop. On ImageNet-256 each variant is shown on its sphere autoencoder with the highest validation PSNR.

Selected hyperparameters. WTS has three solver-side knobs: the learning rate α and the Adam moments β_1 and β_2. WTS+Loop adds two more: the re-encoding frequency (how often we decode-then-reencode the latent during the optimization), and the noise scaler that controls the small spherical perturbation injected at each re-encoding. Figure 9 shows the configuration selected for WTS and WTS+Loop in each setting.

D MULTI-COIL MRI

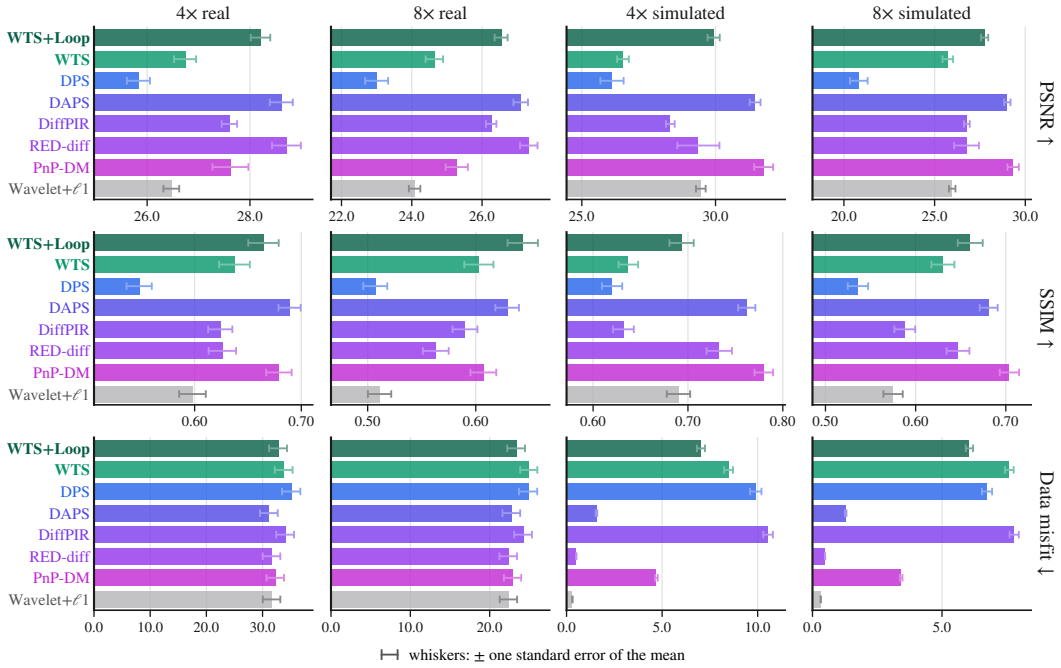

Figure 10: Multi-coil MRI. PSNR (top), SSIM (middle), and data misfit (bottom) on real and simulated k-space at $4\times$ and $8\times$ acceleration. Bars are test-set means and whiskers are $\pm$ one standard error. The WTS and WTS+Loop rows are ours; every baseline row is the published InverseBench number. Alongside the five diffusion baselines we show the strongest additional zero-shot method on this problem (Wavelet $+\ell_1$); the supervised methods and the remaining baselines are in Table 8.

On MRI neither variant leads. WTS+Loop is third among the zero-shot methods on PSNR on every split, reaching 28.21 / 26.54 dB on Real $4\times$ / $8\times$ and 29.93 / 27.76 dB on Sim $4\times$ / $8\times$; on SSIM it leads the diffusion baselines at Real $8\times$. WTS trails it by 1.5–3.4 dB, fifth to eighth of the ten zero-shot methods. RED-Diff leads on the real-k-space splits and PnP-DM on simulated, with DAPS close behind on all four (Table 8).

E SPHERE AUTOENCODER COMPRESSION RATE

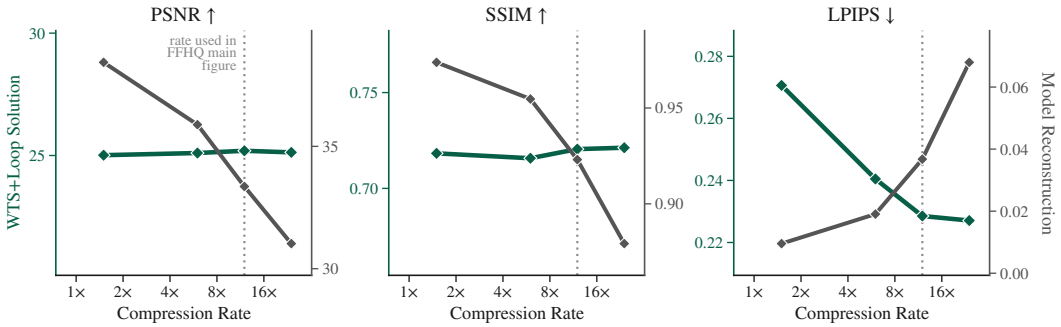

Figure 11: Compression of the sphere autoencoder vs. solve quality. Each panel plots one metric against the compression ratio of the `base` FFHQ-256 sphere autoencoder ($1.5\times$ to $24\times$, log axis): in green (left axis) the WTS+Loop solution, averaged over the six FFHQ-256 settings (the compression ladder was run with WTS+Loop only), and in grey (right axis) the autoencoder's own clean encode–decode reconstruction on the test split. The two axes of a panel share the same span in the metric's units. The dotted line marks the $12\times$ ratio the paper's FFHQ results use.

Figure 11 varies the latent dimension of the `base` FFHQ-256 sphere autoencoder by a factor of 16, with the same solver search space and 5-hour tuning budget as subsection 4.2. The autoencoder's own reconstruction falls steadily with compression, by 7.4 dB PSNR and from 0.97 to 0.88 SSIM across the ladder, while the quality of the WTS+Loop solution barely moves: mean PSNR stays within 0.2 dB and mean SSIM within 0.005 over all four ratios, and mean LPIPS improves from 0.27 at $1.5\times$ to 0.23 at $12\times$ and $24\times$. On FFHQ-256 the solver is therefore not limited by the prior's reconstruction fidelity, which sits 6–13 dB above the solve at every ratio. We use the $12\times$ ratio for the FFHQ-256 results; it is the PSNR optimum of the ladder, and its LPIPS is within 0.002 of the $24\times$ rung.

F FINE-TUNING THE IMAGENET-256 SPHERE ENCODERS WITH A SMALLER LATENT SPACE (-FT)

The released sphere-L and sphere-XL autoencoders (Yue et al., 2026) use patch size 8 with 64 channels per token on 256×256 images, a 65,536-dimensional latent, or $3\times$ compression. On the FFHQ-256 ladder of Appendix E the solve loses nothing in PSNR and improves in LPIPS out to $12\times$ and $24\times$ compression, and training an ImageNet autoencoder from scratch at that ratio is an 800-epoch run we could not afford. We therefore narrow the released latent in place and fine-tune.

Adapter. We replace the sphere projection f of Equation 5 with a rank-k version. Let $\boldsymbol{P} \in \mathbb{R}^{k\times 64}$ have orthonormal rows. The adapter projects each token's channel vector onto the row space of $\boldsymbol{P}$ and then applies the usual RMS normalization,

$$f_k(\tilde{\boldsymbol{z}}) \;=\; f\big((\tilde{\boldsymbol{z}}\boldsymbol{P}^\top)\boldsymbol{P}\big),$$

so the latent lives on the sphere within a k-dimensional subspace of each token. With $k = 16$ the latent has 16,384 dimensions, the same $12\times$ compression as the FFHQ-256 autoencoder. The map is idempotent, which matters because WTS applies it once per step, and it keeps the latent at the norm the pretrained decoder expects. $\boldsymbol{P}$ is kept orthonormal by parametrization throughout training. Since the sphere encoder routes every projection, in training, in the sampling noise path, and in the solver, through this single map, the adapter is attached without any change to the trunk.

Fine-tuning. The only new parameters are the $k \times 64$ entries of $\boldsymbol{P}$, so at fixed encoder weights the adapter can do no better than the best linear rank-k bottleneck of the released latent. The purpose of the fine-tune is to move the encoder so that a rank-k subspace becomes sufficient. We train in two phases from the released checkpoint, with the release's losses and global batch of 256, on the

ImageNet-1k train split. Phase one trains the adapter alone for 500 steps at a constant learning rate of 10^{-3} with the trunk frozen. Phase two trains everything for the remainder of three epochs, with a warmup of 0.2 epochs and then cosine decay from 10^{-4} to 10^{-6}, and the encoder at $0.1\times$ the decoder's learning rate as in the release. Compute is listed in Table 1.

Effect. The fine-tune improves both arms, most at $8\times$ (Table 5). At $8\times$ super-resolution the -FT arms gain $+0.5$ dB (L) and $+0.8$ dB (XL) PSNR and lower LPIPS from 0.375 to 0.325 and from 0.394 to 0.324, which moves them from just behind RED-Diff and DAPS to the top of every panel. At $4\times$, where the released models already share the top tie group, PSNR moves by at most 0.2 dB and LPIPS improves by 0.003 (L) and 0.032 (XL). The -FT arms differ from the released models in both the compression and the three epochs of fine-tuning, so the gain cannot be attributed to the smaller latent alone; the FFHQ-256 ladder shows only that the solve tolerates it.

G When does the measurement ground the prior?

WTS can only learn what belongs in an unobserved region through the pixels it does observe: the observed pixels are local constraints that every candidate reconstruction must satisfy, and they are what keeps the search grounded. An edit the prior makes to the missing region is checked by the measurement only if it also moves some observed pixel. Scattered missing pixels satisfy this almost automatically, since every missing pixel has observed neighbours; a large contiguous hole does not, because edits inside it never reach an observed pixel. We make this precise and measure it.

Grounding. Let $\boldsymbol{z}^* = f(E(\boldsymbol{x}^*))$ be the encoded ground truth, $\boldsymbol{J}$ the Jacobian of $\boldsymbol{z} \mapsto D(f(\boldsymbol{z}))$ at $\boldsymbol{z}^*$ restricted to the tangent space of the sphere, and $\boldsymbol{M}$ the diagonal mask of observed pixels, which observes a fraction ρ of the image. The prior's *edits* to the missing region are the top right singular vectors $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_{256}$ of $(\boldsymbol{I} - \boldsymbol{M})\boldsymbol{J}$: the latent directions that change the decoder's output fastest, counting only the pixels we are inpainting. The **grounding** of an edit is the share of its image change that falls on observed pixels, relative to chance,

$$g(\boldsymbol{q}) \;=\; \frac{\|\boldsymbol{M}\boldsymbol{J}\boldsymbol{q}\|^2}{\rho\,\|\boldsymbol{J}\boldsymbol{q}\|^2}, \tag{8}$$

so $g = 1$ when the change is spread like a random set of pixels and $g = 0$ when it lies entirely inside the missing region, where the measurement cannot see it. We report the median over the 256 edits. Grounding is a local, measurable counterpart of the condition behind the recovery guarantee of Bora et al. (2017), which requires the operator to preserve distances between images the generator can produce; an edit with $g \approx 0$ is exactly a pair of such images that the measurement cannot tell apart. It depends only on the prior and the mask, not on any solver.

Setup. On FFHQ-256 inpainting we fix the number of missing pixels at 25% or 95% and change only their arrangement, grouping them into randomly placed blocks of 1, 4×4, 16×16 or 64×64 pixels, or a single central box. Masks are drawn once and shared by every image. Grounding is computed at the encoded ground truth of 64 test images. WTS+Loop is run on 100 test images with the configuration tuned for 95% random inpainting, unchanged across masks, and scored on the missing pixels against the sphere autoencoder's own reconstruction $D(f(E(\boldsymbol{x}^*)))$, the best the prior can represent.

Results. Figure 12a shows the mechanism on one face. With random missing pixels the edits spread over the whole face and grounding is 0.74; with a box they stay inside it, grounding is 0.01. Across masks (Figure 12b), grounding falls by a factor of about 50 as the same missing pixels are gathered into larger blocks, while the 25% and 95% curves nearly coincide: how the missing pixels are arranged sets grounding, not how many there are. Grounding in turn predicts where WTS falls short (Figure 12c): the weaker the grounding, the further WTS+Loop lands below the prior's own reconstruction, with a per-image Spearman correlation of -0.90 at 25% missing and -0.89 at 95%. The number of missing pixels still matters for the solver, since at equal grounding the 95% masks sit higher, but the ordering within each level is set by grounding.

This separates the settings where a latent-search method like WTS is at home from those where it is not. Measurements spread over the whole signal, like random inpainting or the Gaussian

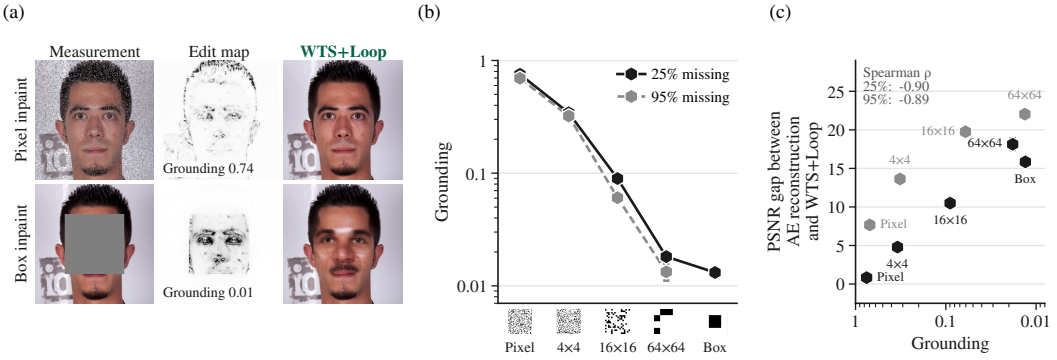

Figure 12: The arrangement of missing pixels, not their number, determines how strongly the measurement grounds the prior. *(a)* A test face with 25% of pixels missing at random (top) or as one central box (bottom). The edit map shows where the prior's 256 edits to the missing pixels change the image (darker = larger change), with the image's grounding (Equation 8). With random missing pixels every edit reaches observed pixels; with a box the prior can change the face unchecked, and WTS+Loop returns a plausible face that is not this person. *(b)* Grounding for masks that group the same missing pixels into blocks of 1, 4, 16 or 64 pixels, or one box, at 25% and 95% missing; geometric mean over 64 test images with 95% bootstrap intervals. The 95% single box is omitted: a box covering 95% of the image must be 250×250, leaving the observed pixels as a 3-pixel frame along the image border. That frame catches edits to lighting and background but none to the face, so grounding rises to 0.11 while the face stays unconstrained. *(c)* WTS+Loop's PSNR on the missing pixels, below the sphere autoencoder's reconstruction of the ground truth, against grounding; 100 test images per point. Spearman ρ over individual (image, mask) pairs is -0.90 (25%) and -0.89 (95%), and -0.81 at 95% if the single box is included; WTS gives -0.89 and -0.89.

measurements assumed by Bora et al. (2017), ground the prior's edits however few of them there are. The same reasoning suggests why WTS does well on inverse scattering (subsection 4.3), where every receiver responds to the whole field, although we have not measured grounding there. A contiguous unobserved region instead leaves the prior to generate its content unchecked, a regime better suited to solvers that sample from a generative prior than to one that searches for a measurement-consistent point.

H QUALITATIVE RESULTS

Figure 13–Figure 15 show further FFHQ-256 reconstructions at the harder difficulty of each task, with the same columns as Figure 4. No image appears more than once across these figures and Figure 4. Each reconstruction is still the single test-split output of the tuned configuration, with no best-of-N selection over samples.

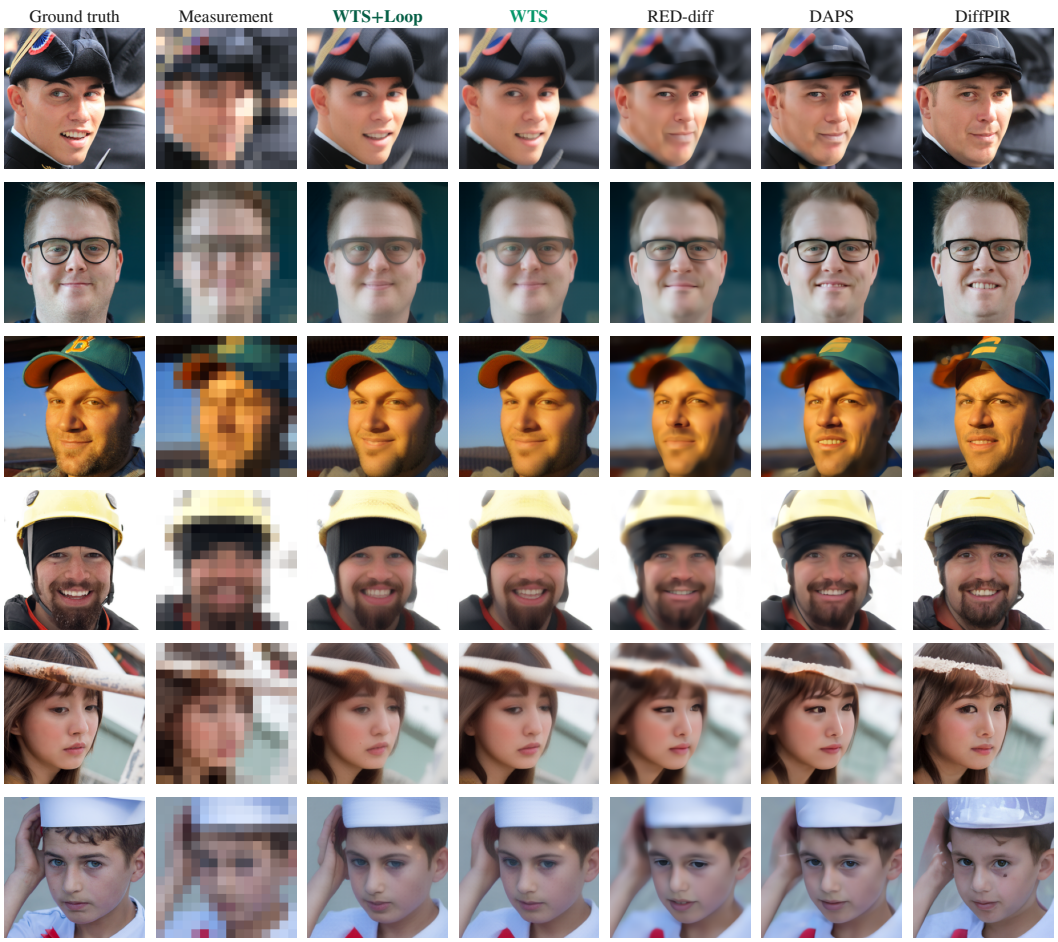

Figure 13: FFHQ-256, 16× super-resolution: ground truth, measurement, WTS+Loop, WTS, RED-Diff, DAPS, and DiffPIR.

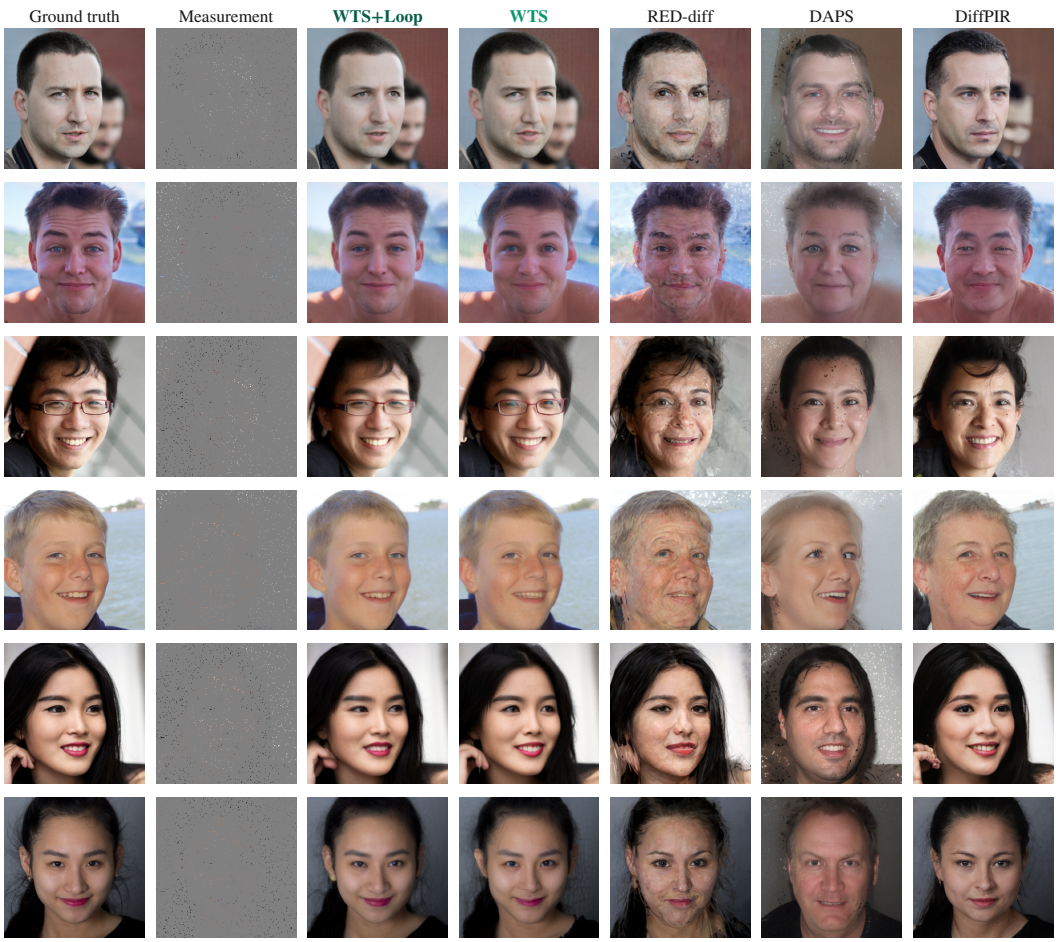

Figure 14: FFHQ-256, random inpainting with 98% of pixels missing: ground truth, measurement, WTS+Loop, WTS, RED-Diff, DAPS, and DiffPIR.



Figure 15: FFHQ-256, Gaussian deblur at intensity 12: ground truth, measurement, WTS+Loop, WTS, RED-Diff, DAPS, and DiffPIR.

Figure 16 and Figure 17 show ImageNet-256 reconstructions for WTS on the released sphere-L and the fine-tuned sphere-XL autoencoders, the two pixel-space baselines with the highest PSNR (RED-Diff and DAPS), and LatentDAPS with the prior that scored best on validation; the remaining arms are reported in Table 5 only. Rows are fixed test ids, identical for every method, so no image is selected per method; the ids were chosen once for recognisable subjects at print size. Each reconstruction is the single test-split output of the tuned configuration reported in the tables, i.e. one reconstruction per test image with no best-of-N selection.

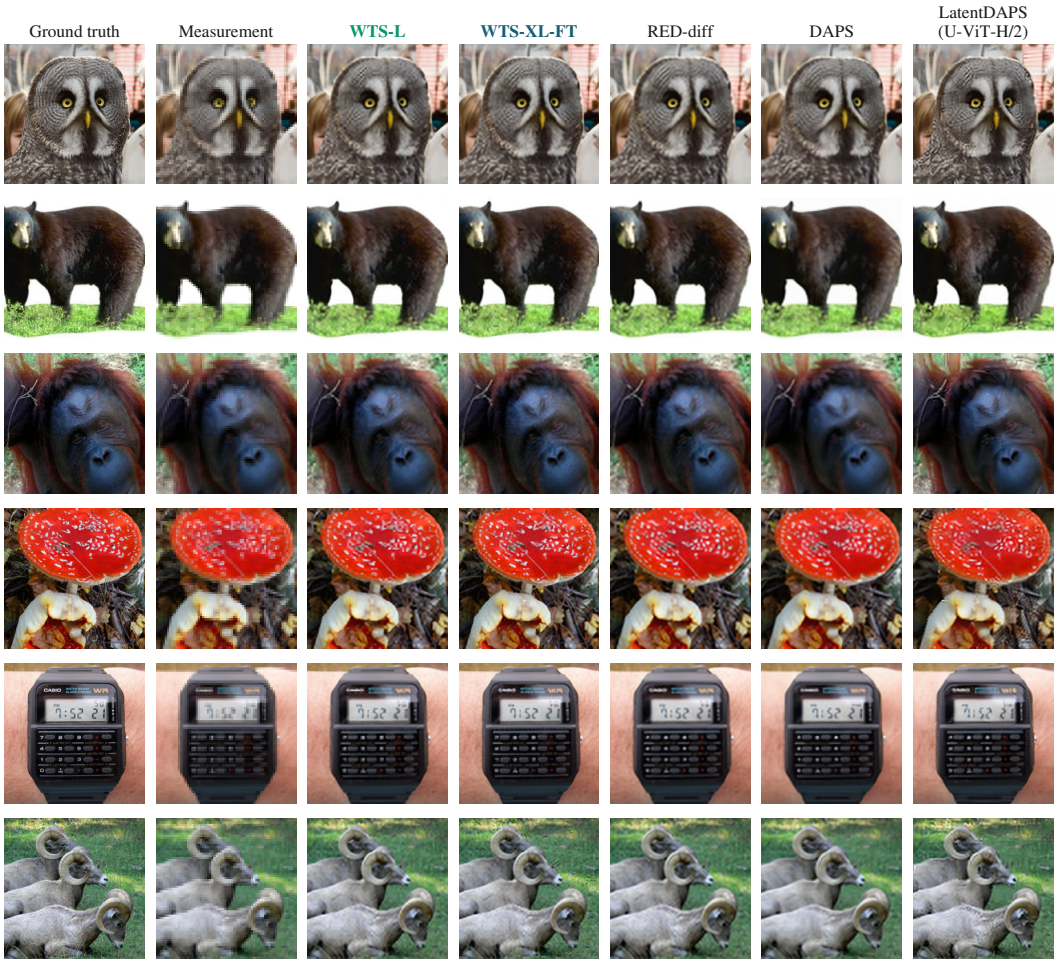

Figure 16: ImageNet-256, $4\times$ super-resolution: ground truth, measurement, WTS-L, WTS-XL-FT, RED-Diff, DAPS, and LatentDAPS on U-ViT-H/2, its validation-best prior at this scale.

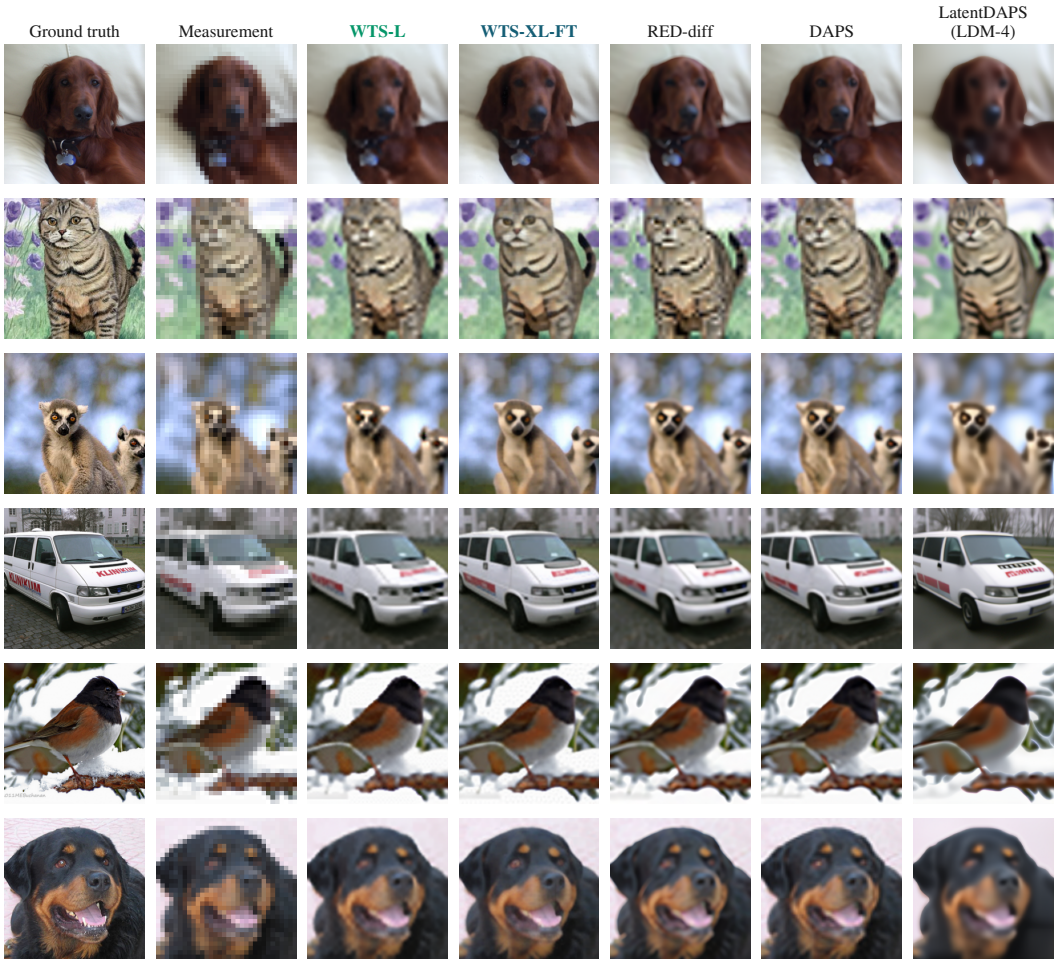

Figure 17: ImageNet-256, 8× super-resolution: ground truth, measurement, WTS-L, WTS-XL-FT, RED-Diff, DAPS, and LatentDAPS on LDM-4, its validation-best prior at this scale.

I Detailed numerical results

This appendix tabulates the per-task, per-method numbers underlying the figures in section 4. Each cell is mean (std) over the test split. Tables follow the order of the experiment subsections: FFHQ-256 image inverse problems (Table 2–Table 4, one per task family), ImageNet-256 super-resolution (Table 5, including the WTS+Loop arms and the two LatentDAPS arms the main figure omits), inverse scattering (Table 6), blackhole imaging (Table 7), and MRI (Table 8). Our rows are the validation-argmax trial of the per-task study applied once to the test split, i.e. the same runs the figures use; baseline rows on FFHQ-256 and ImageNet-256 are our own runs under the same protocol, and baseline rows on the three scientific problems are taken from Zheng et al. (2025).

Table 2: FFHQ-256 super-resolution. *Trials* is the number of completed hyperparameter-tuning trials each method fit into the fixed 5-hour, single-GPU tuning budget (Appendix C).

	8×					16×				
Method	PSNR ↑	SSIM ↑	LPIPS ↓	Time (s) ↓	Trials	PSNR ↑	SSIM ↑	LPIPS ↓	Time (s) ↓	Trials
Diffusion-based										
DPS	23.95 (2.00)	0.607 (0.071)	0.252 (0.036)	5.78	32	21.12 (1.99)	0.517 (0.082)	0.321 (0.043)	5.79	32
DAPS	25.67 (2.09)	0.730 (0.068)	0.223 (0.047)	0.58	184	22.33 (1.90)	0.629 (0.081)	0.313 (0.056)	0.28	267
DiffPIR	24.08 (1.95)	0.643 (0.076)	0.244 (0.044)	0.58	107	21.48 (1.84)	0.582 (0.083)	0.308 (0.053)	1.75	106
RED-diff	26.18 (2.10)	0.749 (0.066)	0.249 (0.048)	2.91	62	22.75 (1.93)	0.645 (0.079)	0.354 (0.060)	2.91	62
PnP-DM	25.48 (2.08)	0.713 (0.066)	0.223 (0.043)	10.29	18	22.03 (2.00)	0.619 (0.081)	0.322 (0.057)	15.54	15
Ours										
WTS	27.00 (2.24)	0.760 (0.065)	0.194 (0.047)	1.78	113	23.43 (2.09)	0.661 (0.080)	0.289 (0.060)	1.78	113
WTS+Loop	26.99 (2.23)	0.759 (0.065)	0.194 (0.047)	1.78	113	23.16 (2.09)	0.649 (0.082)	0.297 (0.060)	1.78	112

Table 3: FFHQ-256 random-mask inpainting. *Trials* is the number of completed hyperparameter-tuning trials each method fit into the fixed 5-hour, single-GPU tuning budget (Appendix C).

	95%					98%				
Method	PSNR ↑	SSIM ↑	LPIPS ↓	Time (s) ↓	Trials	PSNR ↑	SSIM ↑	LPIPS ↓	Time (s) ↓	Trials
Diffusion-based										
DPS	24.09 (2.14)	0.672 (0.061)	0.234 (0.033)	5.79	31	21.71 (2.08)	0.584 (0.071)	0.297 (0.039)	5.79	31
DAPS	21.95 (1.68)	0.651 (0.068)	0.315 (0.052)	0.66	82	17.20 (1.60)	0.443 (0.070)	0.484 (0.056)	0.72	101
DiffPIR	23.32 (1.91)	0.658 (0.069)	0.257 (0.043)	2.31	98	21.11 (1.88)	0.590 (0.079)	0.314 (0.052)	1.73	101
RED-diff	23.93 (1.86)	0.729 (0.060)	0.228 (0.043)	2.89	62	20.39 (1.67)	0.553 (0.077)	0.348 (0.053)	2.89	62
PnP-DM	19.08 (2.48)	0.573 (0.087)	0.371 (0.061)	8.80	20	13.41 (1.08)	0.432 (0.088)	0.714 (0.072)	10.77	19
Ours										
WTS	25.65 (2.38)	0.771 (0.063)	0.172 (0.042)	1.78	113	23.23 (2.40)	0.703 (0.075)	0.237 (0.055)	1.77	113
WTS+Loop	25.92 (2.35)	0.776 (0.062)	0.161 (0.039)	1.80	112	23.95 (2.20)	0.721 (0.070)	0.208 (0.043)	1.82	111

Table 4: FFHQ-256 Gaussian deblur. *Trials* is the number of completed hyperparameter-tuning trials each method fit into the fixed 5-hour, single-GPU tuning budget (Appendix C).

	intensity 6					intensity 12				
Method	PSNR ↑	SSIM ↑	LPIPS ↓	Time (s) ↓	Trials	PSNR ↑	SSIM ↑	LPIPS ↓	Time (s) ↓	Trials
Diffusion-based										
DPS	23.37 (1.94)	0.578 (0.074)	0.268 (0.037)	7.18	26	20.81 (1.93)	0.499 (0.082)	0.330 (0.044)	6.41	28
DAPS	25.49 (1.93)	0.686 (0.061)	0.229 (0.037)	7.01	32	22.63 (1.80)	0.599 (0.072)	0.303 (0.045)	7.02	32
DiffPIR	23.62 (1.90)	0.606 (0.078)	0.261 (0.042)	3.54	68	21.16 (1.81)	0.543 (0.087)	0.316 (0.050)	3.55	85
RED-diff	27.05 (2.11)	0.767 (0.062)	0.239 (0.045)	3.55	51	24.58 (1.92)	0.676 (0.067)	0.330 (0.049)	3.53	51
PnP-DM	25.55 (1.97)	0.711 (0.066)	0.243 (0.046)	12.21	12	22.71 (1.83)	0.629 (0.077)	0.315 (0.056)	10.40	13
Ours										
WTS	27.20 (2.12)	0.758 (0.062)	0.195 (0.044)	1.97	101	24.26 (1.87)	0.659 (0.073)	0.317 (0.057)	1.97	100
WTS+Loop	27.14 (2.16)	0.758 (0.064)	0.206 (0.048)	1.98	100	24.01 (1.93)	0.660 (0.077)	0.304 (0.060)	1.98	101

Table 5: ImageNet-256 super-resolution, all arms. **-FT** marks a released sphere autoencoder whose latent was narrowed to a $4\times$ smaller code by a rank-16 adapter and then fine-tuned for three epochs. **+Loop** rows are WTS+Loop on the same autoencoder; they appear only in this table. *Trials* is the number of completed hyperparameter-tuning trials each method fit into the fixed 5-hour, single-GPU tuning budget (Appendix C).

	$4\times$					$8\times$				
Method	PSNR ↑	SSIM ↑	LPIPS ↓	Time (s) ↓	Trials	PSNR ↑	SSIM ↑	LPIPS ↓	Time (s) ↓	Trials
Pixel-space diffusion										
DPS	24.81 (4.08)	0.683 (0.134)	0.188 (0.059)	21.73	9	20.96 (3.48)	0.478 (0.164)	0.346 (0.068)	21.67	9
DAPS	24.92 (3.43)	0.646 (0.103)	0.266 (0.060)	20.09	20	22.89 (3.67)	0.590 (0.167)	0.356 (0.095)	9.86	23
DiffPIR	23.82 (3.20)	0.633 (0.144)	0.343 (0.092)	10.31	31	22.39 (3.36)	0.565 (0.163)	0.383 (0.098)	10.33	27
RED-diff	26.22 (4.16)	0.755 (0.126)	0.209 (0.078)	10.40	18	22.89 (3.70)	0.591 (0.170)	0.371 (0.097)	10.35	18
PnP-DM	23.51 (3.20)	0.613 (0.150)	0.386 (0.097)	24.26	6	19.31 (2.39)	0.468 (0.165)	0.578 (0.101)	24.19	6
Latent diffusion										
PSLD (LDM-4)	21.53 (3.17)	0.481 (0.163)	0.447 (0.087)	19.61	10	20.03 (2.95)	0.438 (0.165)	0.498 (0.088)	19.36	10
PSLD (U-ViT-L/2)	23.47 (3.59)	0.585 (0.155)	0.366 (0.083)	16.68	11	20.95 (3.08)	0.459 (0.158)	0.452 (0.072)	16.63	11
PSLD (U-ViT-H/2)	23.20 (3.23)	0.565 (0.157)	0.359 (0.095)	21.57	9	21.14 (3.06)	0.480 (0.165)	0.419 (0.095)	21.58	9
LatentDAPS (LDM-4)	24.39 (3.51)	0.694 (0.136)	0.313 (0.089)	21.63	3	22.35 (3.33)	0.566 (0.166)	0.439 (0.106)	21.72	3
LatentDAPS (U-ViT-L/2)	24.73 (3.90)	0.692 (0.137)	0.249 (0.068)	104.55	4	19.95 (3.58)	0.401 (0.165)	0.430 (0.061)	54.39	4
LatentDAPS (U-ViT-H/2)	24.87 (3.89)	0.693 (0.135)	0.243 (0.070)	107.38	4	20.76 (3.60)	0.450 (0.184)	0.392 (0.104)	56.62	4
Ours										
WTS-L	26.35 (4.12)	0.754 (0.125)	0.191 (0.074)	5.85	37	22.77 (3.57)	0.588 (0.167)	0.375 (0.095)	5.86	37
WTS-XL	26.16 (4.08)	0.750 (0.126)	0.208 (0.075)	8.63	23	22.47 (3.44)	0.576 (0.164)	0.394 (0.090)	8.62	23
WTS-L-FT	26.21 (4.11)	0.745 (0.128)	0.188 (0.072)	5.86	37	23.23 (3.69)	0.609 (0.165)	0.325 (0.096)	5.88	36
WTS-XL-FT	26.38 (4.11)	0.751 (0.125)	0.176 (0.071)	8.64	23	23.27 (3.69)	0.608 (0.166)	0.324 (0.095)	8.66	23
WTS-L+Loop	25.28 (3.85)	0.724 (0.126)	0.234 (0.074)	5.90	36	22.63 (3.51)	0.581 (0.166)	0.386 (0.093)	5.88	36
WTS-XL+Loop	25.34 (3.86)	0.725 (0.127)	0.232 (0.074)	8.64	23	22.48 (3.50)	0.563 (0.167)	0.379 (0.097)	8.73	23
WTS-L-FT+Loop	25.81 (3.99)	0.730 (0.129)	0.199 (0.071)	5.87	36	22.79 (3.61)	0.582 (0.166)	0.354 (0.093)	5.89	36
WTS-XL-FT+Loop	25.74 (3.97)	0.730 (0.130)	0.194 (0.071)	8.71	23	22.86 (3.58)	0.590 (0.167)	0.362 (0.098)	8.69	23

Table 6: Inverse Scattering.

	360 receivers			180 receivers			60 receivers		
Method	PSNR ↑	SSIM ↑	Meas err (%) ↓	PSNR ↑	SSIM ↑	Meas err (%) ↓	PSNR ↑	SSIM ↑	Meas err (%) ↓
Traditional									
FISTA-TV	32.13 (2.14)	0.979 (0.009)	1.23 (0.25)	26.52 (2.68)	0.914 (0.040)	2.65 (0.30)	20.94 (2.51)	0.709 (0.103)	6.05 (0.65)
Diffusion-based									
DDRM	32.60 (1.82)	0.929 (0.012)	1.04 (0.26)	28.08 (1.52)	0.890 (0.019)	1.57 (0.39)	20.44 (1.21)	0.545 (0.037)	3.04 (0.92)
DDNM	36.38 (1.10)	0.935 (0.017)	0.78 (0.22)	35.02 (0.99)	0.895 (0.027)	0.58 (0.16)	29.23 (3.38)	0.917 (0.022)	0.28 (0.07)
ΠGDM	27.93 (3.21)	0.889 (0.072)	2.74 (1.23)	26.41 (3.43)	0.816 (0.114)	3.66 (1.79)	20.07 (2.61)	0.540 (0.198)	6.90 (3.38)
DPS	32.06 (2.16)	0.846 (0.127)	4.35 (1.19)	31.80 (2.16)	0.862 (0.123)	4.28 (1.20)	27.37 (3.42)	0.813 (0.133)	4.53 (1.31)
LGD	27.90 (2.35)	0.812 (0.037)	1.17 (0.20)	27.84 (2.34)	0.803 (0.034)	1.06 (0.16)	20.49 (3.03)	0.552 (0.077)	1.45 (0.68)
DiffPIR	34.24 (2.31)	0.988 (0.006)	1.11 (0.24)	34.01 (2.27)	0.987 (0.006)	1.04 (0.23)	26.32 (3.27)	0.918 (0.028)	1.27 (0.23)
PnP-DM	33.91 (2.05)	0.988 (0.006)	1.21 (0.25)	31.82 (2.07)	0.981 (0.008)	1.42 (0.26)	24.71 (2.87)	0.909 (0.046)	2.20 (0.34)
DAPS	34.64 (1.69)	0.957 (0.006)	1.03 (0.25)	33.16 (1.70)	0.944 (0.009)	1.11 (0.25)	25.88 (3.11)	0.885 (0.030)	1.51 (0.25)
RED-diff	36.56 (2.29)	0.981 (0.005)	0.89 (0.23)	35.41 (2.17)	0.984 (0.004)	0.87 (0.21)	27.07 (3.33)	0.935 (0.037)	1.18 (0.23)
FPS	33.24 (1.60)	0.870 (0.026)	0.70 (0.01)	29.62 (1.65)	0.710 (0.040)	0.37 (0.01)	21.32 (1.45)	0.460 (0.030)	0.15 (0.02)
MCG-diff	30.94 (1.96)	0.751 (0.029)	0.70 (0.01)	28.06 (1.67)	0.631 (0.042)	0.38 (0.01)	21.00 (1.57)	0.445 (0.028)	0.21 (0.06)
Ours									
WTS	38.90 (2.98)	0.985 (0.058)	1.26 (1.91)	37.98 (2.74)	0.988 (0.010)	1.11 (0.36)	33.11 (4.99)	0.958 (0.038)	0.97 (0.36)
WTS+Loop	39.08 (2.40)	0.991 (0.003)	1.10 (0.32)	38.39 (2.33)	0.995 (0.002)	1.15 (0.29)	34.05 (2.88)	0.986 (0.009)	0.92 (0.28)

Table 7: Blackhole imaging.

	3% obs				10% obs				100% obs			
Method	PSNR ↑	Blur ↑	$\bar{\chi}^2_{cp}$ ↓	$\bar{\chi}^2_{ca}$ ↓	PSNR ↑	Blur ↑	$\bar{\chi}^2_{cp}$ ↓	$\bar{\chi}^2_{ca}$ ↓	PSNR ↑	Blur ↑	$\bar{\chi}^2_{cp}$ ↓	$\bar{\chi}^2_{ca}$ ↓
Traditional												
SMILI	18.51 (1.39)	23.08 (2.12)	1.48 (0.43)	4.35 (3.83)	20.85 (2.90)	25.24 (3.86)	1.21 (0.17)	21.79 (12.49)	22.67 (3.13)	27.79 (4.02)	1.88 (0.95)	17.61 (10.30)
EHT-Imaging	21.72 (3.39)	25.66 (5.04)	1.51 (0.48)	1.70 (0.54)	22.67 (3.46)	26.66 (3.93)	1.17 (0.16)	1.24 (0.20)	24.28 (3.63)	28.57 (4.52)	1.25 (0.25)	1.26 (0.32)
Diffusion-based												
DPS	24.20 (3.72)	30.83 (5.58)	8.02 (24.34)	5.01 (5.75)	24.36 (3.72)	30.79 (5.75)	13.05 (43.09)	6.61 (26.79)	25.86 (3.90)	32.94 (6.19)	8.76 (37.78)	5.46 (24.18)
LGD	22.51 (3.76)	28.50 (5.49)	15.82 (16.84)	12.86 (12.66)	22.08 (3.75)	27.48 (5.09)	10.78 (21.68)	13.38 (56.40)	21.22 (3.64)	26.06 (4.98)	13.24 (17.23)	13.23 (39.11)
RED-diff	20.74 (2.62)	26.10 (3.35)	6.71 (6.92)	9.13 (19.05)	22.53 (3.02)	27.67 (4.53)	2.49 (2.92)	4.92 (13.22)	23.77 (4.13)	29.13 (6.22)	1.85 (0.94)	2.05 (2.36)
PnP-DM	24.25 (3.45)	30.49 (4.93)	2.20 (1.35)	1.67 (0.55)	24.57 (3.47)	30.80 (5.22)	1.43 (0.42)	1.34 (0.48)	26.07 (3.70)	32.88 (6.02)	1.31 (0.20)	1.20 (0.22)
DAPS	23.54 (3.28)	29.48 (4.88)	3.65 (3.29)	2.33 (1.35)	23.99 (3.56)	30.10 (5.13)	1.54 (0.70)	2.25 (9.90)	25.60 (3.64)	32.78 (5.68)	1.30 (0.32)	1.23 (0.53)
DiffPIR	24.12 (3.25)	30.45 (4.88)	14.09 (14.11)	10.54 (8.86)	23.84 (3.59)	30.04 (5.03)	5.37 (3.73)	5.21 (5.56)	25.01 (4.64)	31.86 (6.56)	3.27 (1.62)	2.97 (1.20)
Ours												
WTS	23.21 (2.49)	27.02 (3.32)	1.96 (1.23)	1.87 (0.94)	24.29 (3.00)	28.34 (3.96)	1.65 (1.11)	1.94 (2.92)	26.56 (3.12)	31.94 (4.69)	1.21 (0.23)	1.19 (0.19)
WTS+Loop	23.84 (2.74)	28.25 (3.87)	2.00 (1.56)	1.87 (1.04)	24.66 (3.18)	28.96 (4.43)	1.39 (0.61)	1.60 (1.98)	27.15 (3.13)	33.24 (4.89)	1.34 (0.35)	1.24 (0.31)

Table 8: MRI.

Method	4× Sim			4× Real			8× Sim			8× Real		
	PSNR ↑	SSIM ↑	Misfit ↓	PSNR ↑	SSIM ↑	Misfit ↓	PSNR ↑	SSIM ↑	Misfit ↓	PSNR ↑	SSIM ↑	Misfit ↓
Traditional												
Wavelet+ℓ_1	29.45 (1.78)	0.690 (0.121)	0.31 (0.05)	26.47 (1.51)	0.598 (0.122)	31.60 (15.29)	25.97 (1.76)	0.575 (0.105)	0.32 (0.04)	24.08 (1.60)	0.511 (0.106)	22.36 (10.73)
TV	27.03 (1.64)	0.518 (0.123)	5.75 (1.28)	26.22 (1.58)	0.509 (0.123)	32.27 (15.41)	24.12 (1.90)	0.432 (0.112)	5.09 (1.05)	23.70 (1.86)	0.427 (0.112)	23.05 (10.85)
End-to-end												
Residual UNet	32.27 (1.81)	0.808 (0.080)	–	31.70 (1.97)	0.785 (0.095)	–	29.75 (1.68)	0.750 (0.088)	–	29.36 (1.75)	0.733 (0.100)	–
E2E-VarNet	33.40 (2.10)	0.836 (0.079)	–	31.71 (2.54)	0.756 (0.102)	–	30.67 (1.76)	0.769 (0.085)	–	30.45 (1.94)	0.736 (0.103)	–
Diffusion-based												
CSGM	28.78 (6.17)	0.710 (0.147)	1.52 (0.43)	25.17 (6.25)	0.582 (0.167)	31.64 (15.38)	26.15 (6.38)	0.625 (0.158)	1.14 (1.08)	21.17 (8.31)	0.425 (0.192)	22.09 (10.74)
ScoreMRI	25.97 (1.68)	0.468 (0.087)	10.83 (1.73)	25.60 (1.62)	0.463 (0.086)	33.70 (15.21)	25.01 (1.53)	0.405 (0.079)	8.36 (1.38)	24.74 (1.48)	0.403 (0.080)	24.03 (10.66)
RED-diff	29.36 (7.71)	0.733 (0.131)	0.51 (0.08)	28.71 (2.75)	0.626 (0.126)	31.59 (15.37)	26.76 (6.70)	0.647 (0.124)	0.48 (0.07)	27.33 (2.44)	0.563 (0.117)	22.34 (10.84)
DiffPIR	28.31 (1.60)	0.632 (0.107)	10.54 (2.47)	27.60 (1.47)	0.624 (0.111)	34.02 (15.52)	26.78 (1.56)	0.588 (0.113)	7.79 (1.74)	26.26 (1.46)	0.590 (0.113)	24.21 (10.92)
DPS	26.13 (4.25)	0.620 (0.105)	9.90 (2.92)	25.83 (2.20)	0.548 (0.116)	35.09 (15.97)	20.82 (4.78)	0.536 (0.111)	6.74 (1.93)	23.00 (3.21)	0.507 (0.109)	24.84 (11.26)
DAPS	31.48 (1.99)	0.762 (0.089)	1.57 (0.39)	28.61 (2.20)	0.689 (0.102)	31.11 (15.50)	29.01 (1.71)	0.681 (0.098)	1.28 (0.30)	27.10 (2.03)	0.629 (0.107)	22.73 (10.93)
PnP-DM	31.80 (3.47)	0.780 (0.096)	4.70 (0.68)	27.62 (3.42)	0.679 (0.117)	32.26 (15.17)	29.33 (3.08)	0.704 (0.105)	3.42 (0.50)	25.28 (3.10)	0.607 (0.117)	22.88 (10.71)
Ours												
WTS	26.54 (2.14)	0.637 (0.101)	8.48 (2.28)	26.74 (2.10)	0.637 (0.141)	33.75 (15.50)	25.73 (2.85)	0.630 (0.125)	7.60 (1.63)	24.64 (2.43)	0.603 (0.132)	24.92 (10.58)
WTS+Loop	29.93 (2.22)	0.693 (0.125)	7.03 (2.05)	28.21 (1.86)	0.665 (0.140)	32.75 (15.59)	27.76 (1.94)	0.661 (0.136)	6.06 (1.37)	26.54 (1.82)	0.643 (0.138)	23.32 (10.98)