Diffusion Models for Inverse Problems Done Right

Ju Sun
Computer Science & Engineering, UMN
Mar 06, 2025

SIAM 2025 Conference on Computational Science and Engineering

Generative Machine Learning Models for Uncertainty Quantification

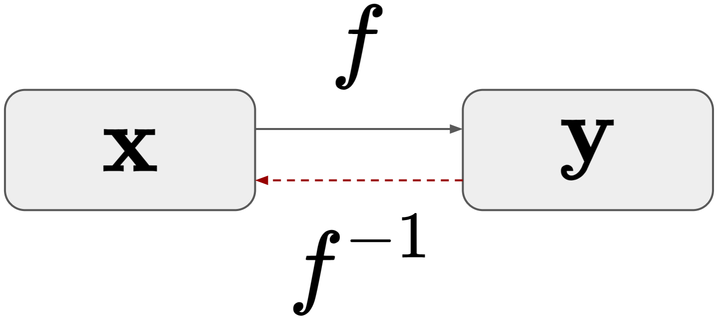
University of Minnesota
Driven to Discover℠

Inverse Problems

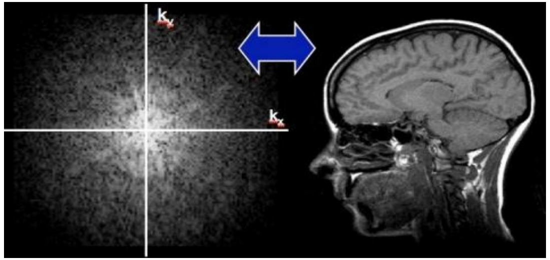
Inverse problems

Inverse problem: given $\mathbf{y} = f(\mathbf{x})$, recover $\mathbf{x}$

Image denoising

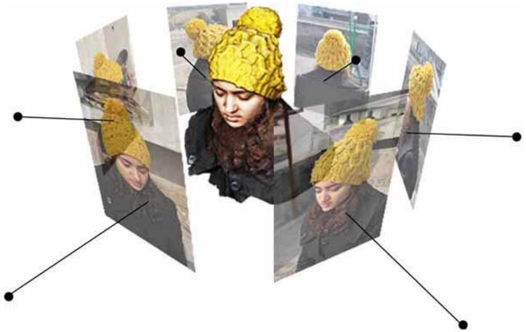


$$\mathbf{x} \xrightarrow{f} \mathbf{y}, \quad \mathbf{y} \xrightarrow{f^{-1}} \mathbf{x}$$

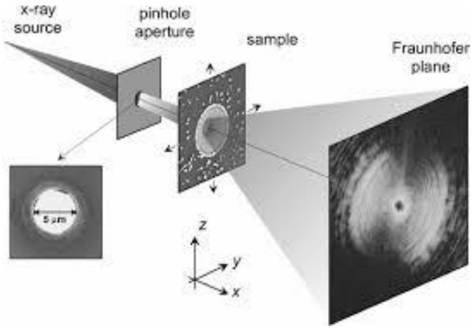


MRI reconstruction

Image super-resolution



3D reconstruction



Coherent diffraction imaging (CDI)

Traditional methods

Inverse problem: given $\mathbf{y} = f(\mathbf{x})$, recover $\mathbf{x}$

$$\min_{\mathbf{x}} \underbrace{\ell(\mathbf{y}, f(\mathbf{x}))}_{\text{data fitting}} + \lambda \underbrace{\mathrm{R}(\mathbf{x})}_{\text{regularizer}}$$

RegFit

Questions

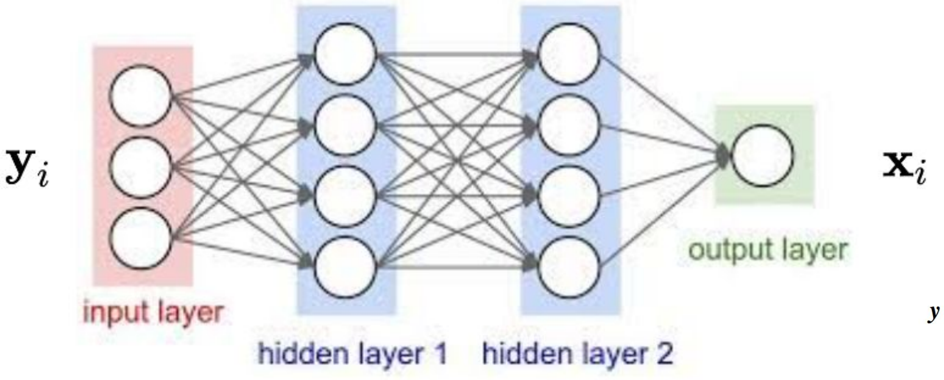
- Which ℓ? (e.g., unknown/compound noise)
- Which R? (e.g., structures not amenable to math description)
- Speed

Deep learning has changed everything

With paired datasets $\{(\boldsymbol{y}_i, \boldsymbol{x}_i)\}_{i=1,\dots,N}$

Direct inversion

Learn f^{-1} from $\{(\boldsymbol{y}_i, \boldsymbol{x}_i)\}_{i=1,\dots,N}$

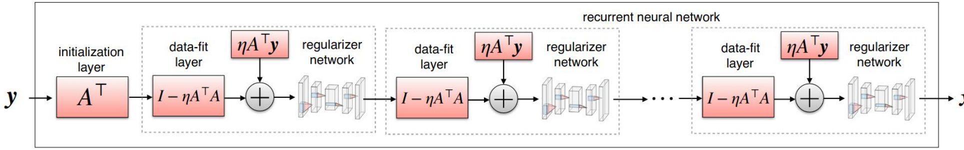


Algorithm unrolling

$$\min_{\mathbf{x}} \ell(\mathbf{y}, f(\mathbf{x})) + \lambda \, \mathrm{R}(\mathbf{x})$$

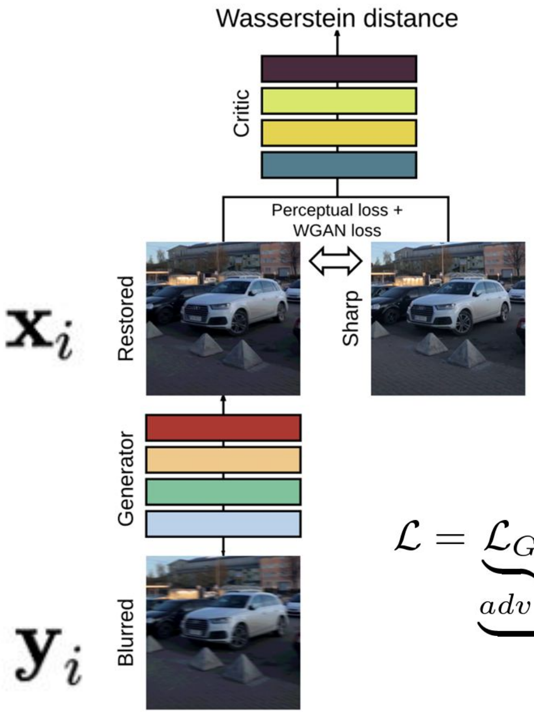
$$\mathbf{x}^{k+1} = \mathcal{P}_R(\mathbf{x}^k - \eta \nabla^\top f(\mathbf{x}^k) \ell'(\mathbf{y}, f(\mathbf{x}^k)))$$

<u>Idea</u>: make $\mathcal{P}_R$ trainable



With paired datasets $\{(\boldsymbol{y}_i, \boldsymbol{x}_i)\}_{i=1,\ldots,N}$

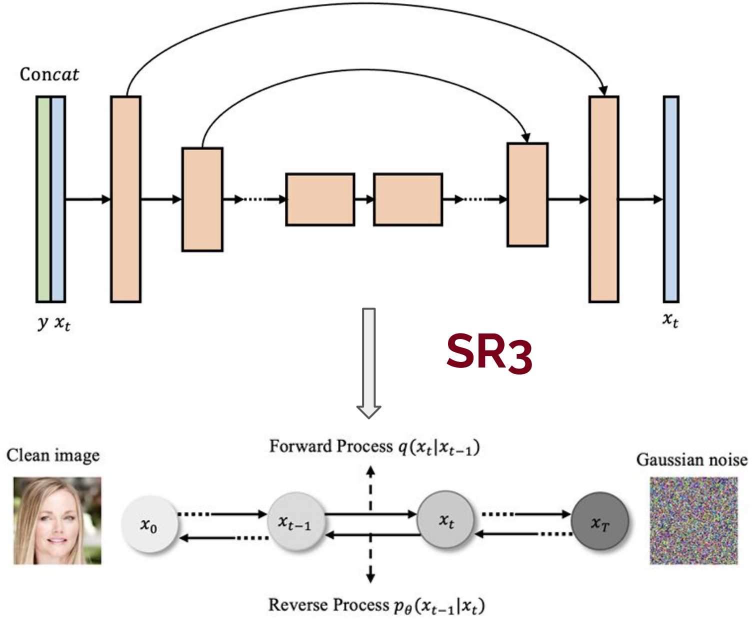
Conditional generation & regularization



DeblurGAN

$$\mathcal{L} = \underbrace{\mathcal{L}_{GAN}}_{adv\ loss} + \underbrace{\lambda \cdot \mathcal{L}_X}_{content\ loss}$$

total loss



SR3

Image credit: https://arxiv.org/abs/2308.09388

With object datasets only $\{\boldsymbol{x}_i\}_{i=1,\ldots,N}$

Model the distribution of the objects first, and then plug the prior in

GAN Inversion

Pretraining: $\mathbf{x}_i \approx G_\theta(\mathbf{z}_i) \ \forall i$

Deployment: $\min_{\mathbf{z}} \ \ell(\mathbf{y}, \ f \circ G_\theta(\mathbf{z})) + \lambda R \circ G_\theta(\mathbf{z})$

Interleaving pretrained diffusion models

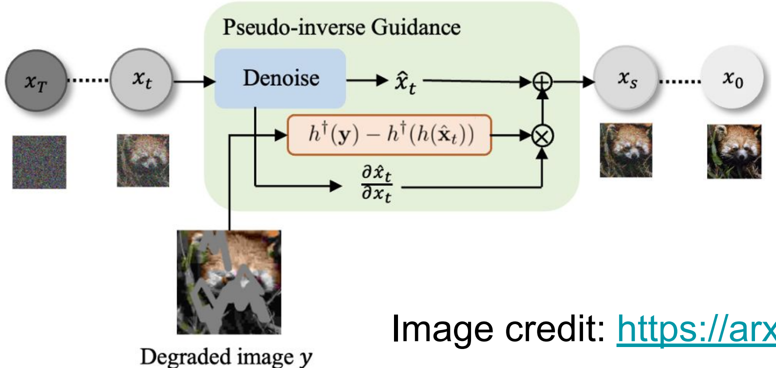


Image credit: https://arxiv.org/abs/2308.09388

Without datasets? Single-instance methods

Deep image prior (DIP) $\mathbf{x} \approx G_\theta(\mathbf{z})$ G_θ (and $\mathbf{z}$) trainable

$$\min_{\mathbf{x}} \underbrace{\ell(\mathbf{y}, f(\mathbf{x}))}_{\text{data fitting}} + \lambda \underbrace{\mathrm{R}(\mathbf{x})}_{\text{regularizer}}$$

$$\Downarrow$$

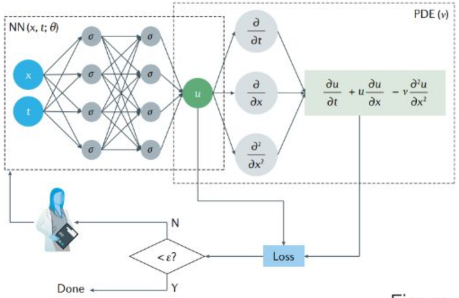
$$\min_{\theta} \ell(\mathbf{y}, f \circ G_\theta(\mathbf{z})) + \lambda R \circ G_\theta(\mathbf{z})$$

No extra training data!

Neural implicit representation (NIR)

$$\mathbf{x} \approx \mathcal{D} \circ \overline{\mathbf{x}} \qquad \mathcal{D}: \text{discretization} \qquad \overline{\mathbf{x}}: \text{continuous function}$$

Physics-informed neural networks (PINN)



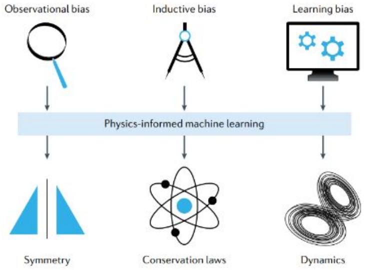


Figure credit: https://www.nature.com/articles/s42254-021-00314-5

Table 2: *Major categories of methods learning to solve inverse problems based on what is known about the forward model $\mathcal{A}$ and the nature of the training data, with examples for each. Details are described throughout Section 4.*

	Supervised with matched (x, y) pairs	**Train from unpaired x's and y's (Unpaired ground truths and Measurements)**	**Train from x's only (Ground truth only)**	**Train from y's only (Measurements only)**
$\mathcal{A}$ fully known during training and testing (§4.1)	§4.1.1: Denoising auto-encoders [16], U-Net [78], Deep convolutional framelets [79] Unrolled optimization [80–83], Neumann networks [84]	*amounts to training from $(\boldsymbol{x}, \boldsymbol{y})$ pairs*	*amounts to training from $(\boldsymbol{x}, \boldsymbol{y})$ pairs*	§4.1.2: SURE LDAMP [85, 86], Deep Basis Pursuit [87]
$\mathcal{A}$ known only at test time (§4.2)	§4.2.2	§4.2.2	§4.2.1: CSGM [25], LDAMP [88], OneNet [22], Plug-and-play [89], RED [90]	§4.2.2
$\mathcal{A}$ partially known (§4.3)	§4.3.1	§4.3.2: CycleGAN [91]	§4.3.3: Blind deconvolution with GAN's [92–94]	§4.3.4: AmbientGAN [76], Noise2Noise [95], UAIR [96]
$\mathcal{A}$ unknown (§4.4)	§4.4.1: AUTOMAP [97]	§4.4.2	§4.4.2	§4.4.2

Deep Learning Techniques for Inverse Problems in Imaging

Gregory Ongie,[*] Ajil Jalal,[†] Christopher A. Metzler[‡]
Richard G. Baraniuk,[§] Alexandros G. Dimakis,[¶] Rebecca Willett[‖]

https://arxiv.org/abs/2005.06001

But focused on linear IPs

Other specialized surveys

Algorithm Unrolling: Interpretable, Efficient Deep Learning for Signal and Image Processing

Vishal Monga, *Senior Member, IEEE,* Yuelong Li, *Member, IEEE,* and Yonina C. Eldar, *Fellow, IEEE*

Focused on alg. unrolling

Untrained Neural Network Priors for Inverse Imaging Problems: A Survey

Deep Internal Learning: Deep Learning from a Single Input

Tom Tirer *Member,*

Understanding Untrained Deep Models for Inverse Problems: Algorithms and Theory

Ismail Alkhouri, Evan Bell, Avrajit Ghosh, Shijun Liang, Rongrong Wang,

Focused on single-instance methods

Theoretical Perspectives on Deep Learning Methods in Inverse Problems

Jonathan Scarlett, Reinhard Heckel, Miguel R. D. Rodrigues, Paul Hand, and Yonina C. Eldar

Focused on theories for linear IPs

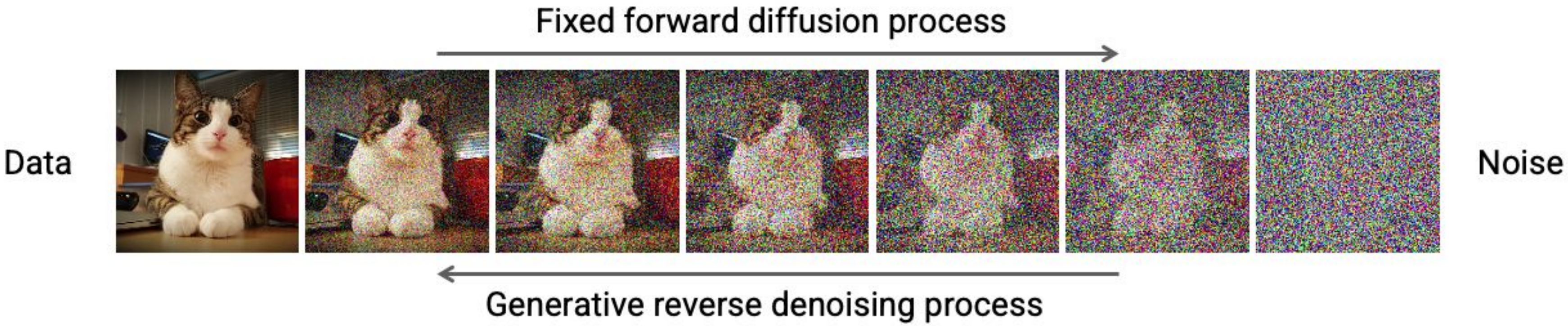
This talk:

Solving Inverse Problems (IPs)

Using Pretrained Diffusion Models

Diffusion models

$$d\boldsymbol{x} = -\beta_t/2 \cdot \boldsymbol{x} dt + \sqrt{\beta_t} d\boldsymbol{w},$$

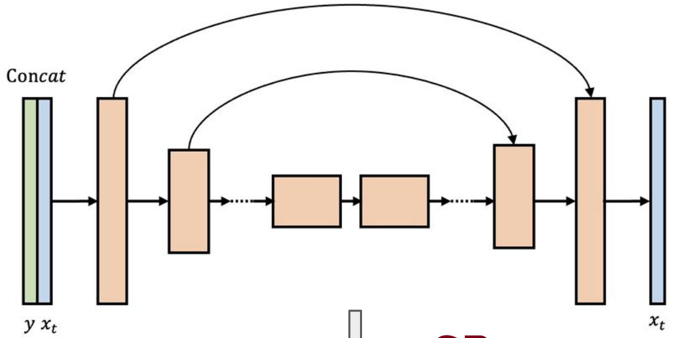


$$d\boldsymbol{x} = -\beta_t \left[\boldsymbol{x}/2 + \boxed{\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})}\right] dt + \sqrt{\beta_t} d\bar{\boldsymbol{w}}.$$

$$\cong \quad \boldsymbol{\varepsilon}_{\boldsymbol{\theta}}^{(t)}(\boldsymbol{x})$$

Diffusion models for inverse problems (IPs)

Supervised



SR3

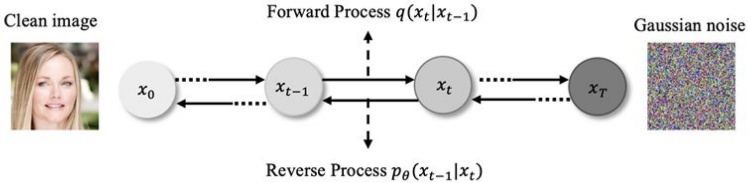
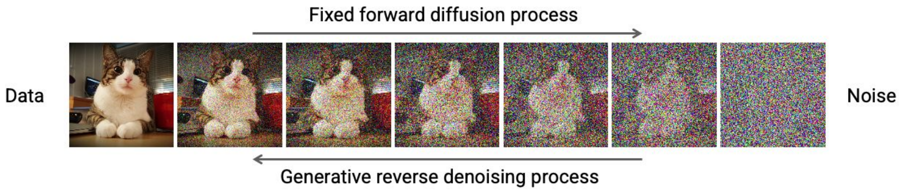
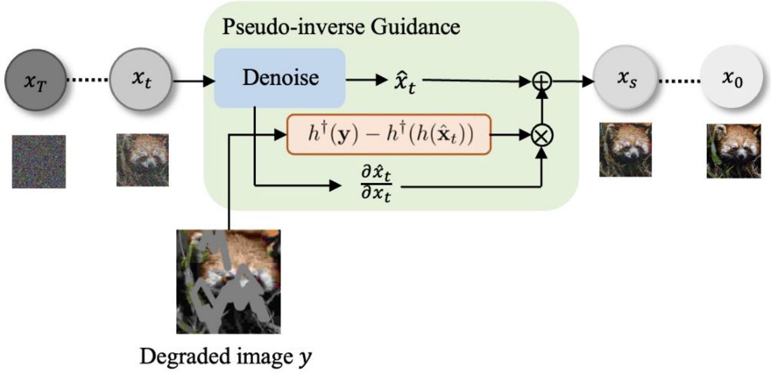


Image credit: https://arxiv.org/abs/2308.09388

Zero-shot





Focus: IPs with pretrained diffusion models

$$\text{(Reverse SDE for DDPM)}\; d\boldsymbol{x} = -\beta_t \left[\boldsymbol{x}/2 + \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})\right] dt + \sqrt{\beta_t} d\bar{\boldsymbol{w}}$$

Think of **conditional score function**

$$\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}|\boldsymbol{y}) = \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}) + \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{y}|\boldsymbol{x})$$

Conditional reverse SDE

$$d\boldsymbol{x} = \left[-\beta_t/2 \cdot \boldsymbol{x} - \beta_t(\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}) + \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{y}|\boldsymbol{x}))\right] dt + \sqrt{\beta_t} d\bar{\boldsymbol{w}}$$

Coping with conditional score function

$$\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}|\boldsymbol{y}) = \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}) + \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{y}|\boldsymbol{x})$$

$$\cong \boldsymbol{\varepsilon}_{\boldsymbol{\theta}}^{(t)}(\boldsymbol{x})$$

$$p_t(\boldsymbol{y}|\boldsymbol{x}(t))$$

$$\cong p_t(\boldsymbol{y}|\widehat{\boldsymbol{x}}(0)[\boldsymbol{x}(t)])$$

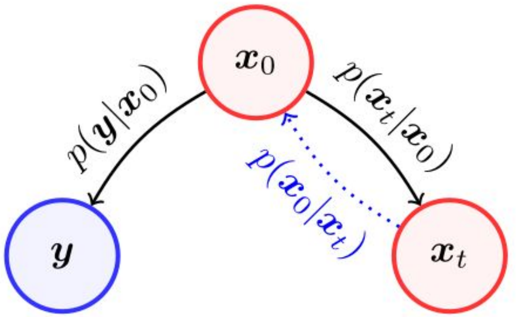


Figure 2: Probabilistic graph. Black solid line: tractable, blue dotted line: intractable in general.

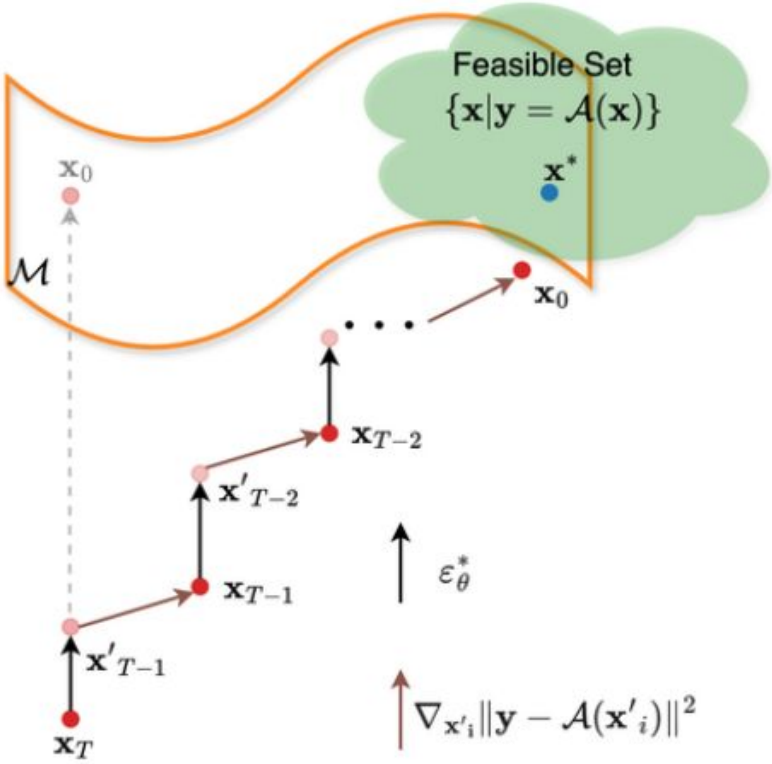
Interleaving methods

Algorithm 1 Template for interleaving methods

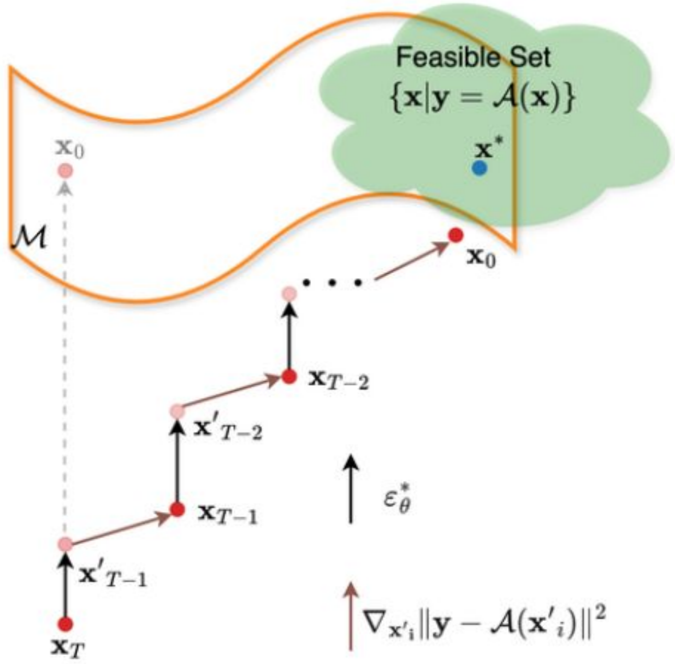
Input: # Diffusion steps T, measurement $\boldsymbol{y}$

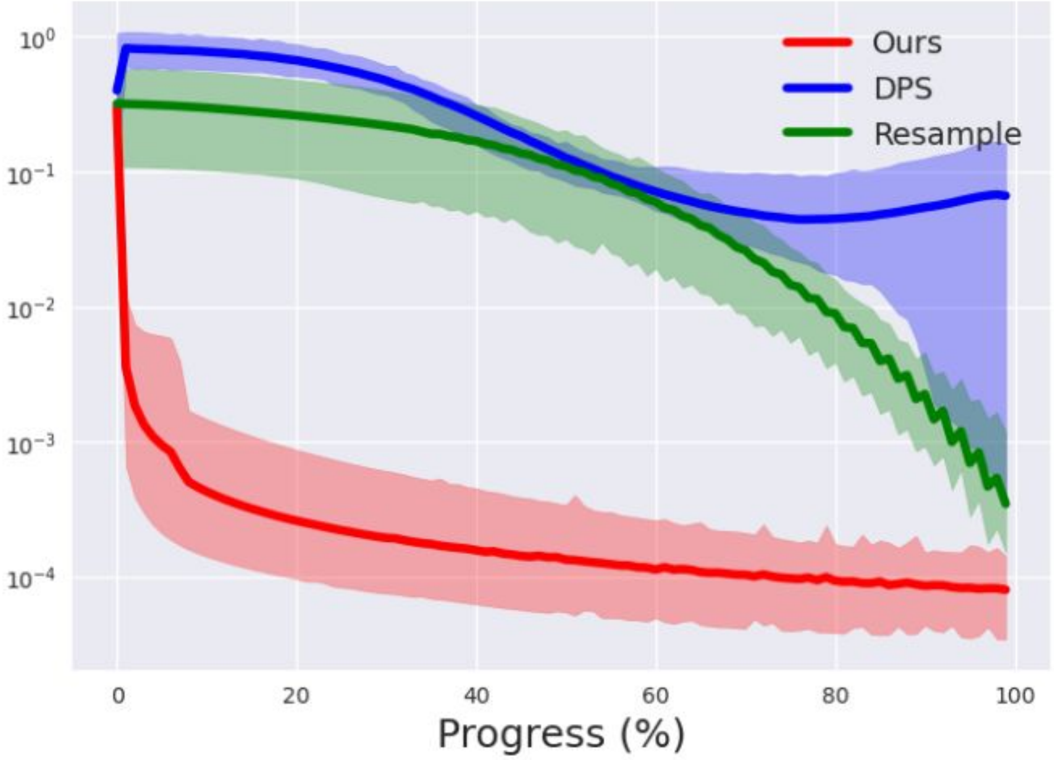
1: $\boldsymbol{x}_T \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
2: **for** $i = T - 1$ to 0 **do**
3: $\hat{\boldsymbol{s}} \leftarrow \boldsymbol{\varepsilon}_{\boldsymbol{\theta}}^{(i)}(\boldsymbol{x}_i)$
4: $\hat{\boldsymbol{x}}_0 \leftarrow \frac{1}{\sqrt{\bar{\alpha}_i}}(\boldsymbol{x}_i - \sqrt{1-\bar{\alpha}_i}\hat{\boldsymbol{s}})$
5: $\boldsymbol{x}'_{i-1} \leftarrow$ DDIM reverse with $\hat{\boldsymbol{x}}_0$ and $\hat{\boldsymbol{s}}$
6: $\boldsymbol{x}_{i-1} \leftarrow$ (Approximately) Projection [39, 30, 33, 32, 40, 41, 34] or gradient update [20, 28, 19, 21, 29, 27, 26] with $\hat{\boldsymbol{x}}_0$ and $\boldsymbol{x}'_{i-1}$ to get closer to $\{\boldsymbol{x}|\boldsymbol{y} = \mathcal{A}(\boldsymbol{x})\}$
7: **end for**

Output: Recovered object $\boldsymbol{x}_0$

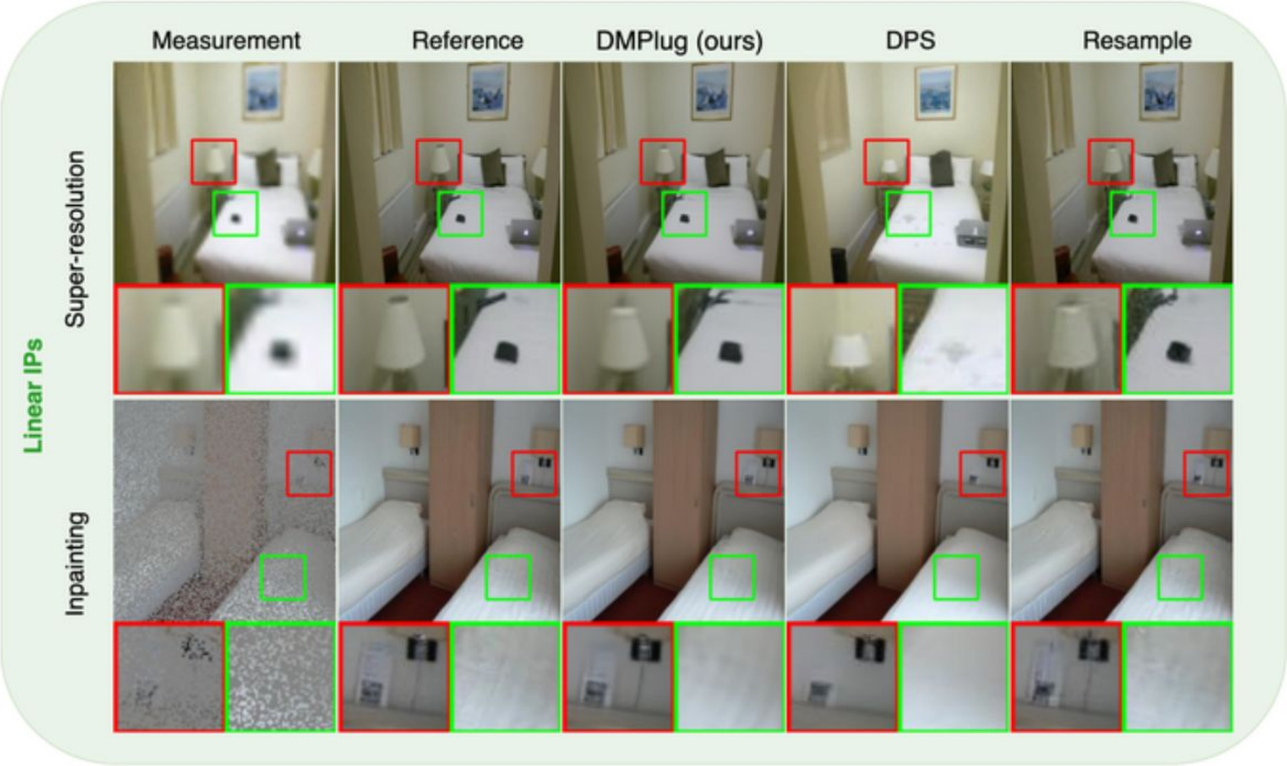


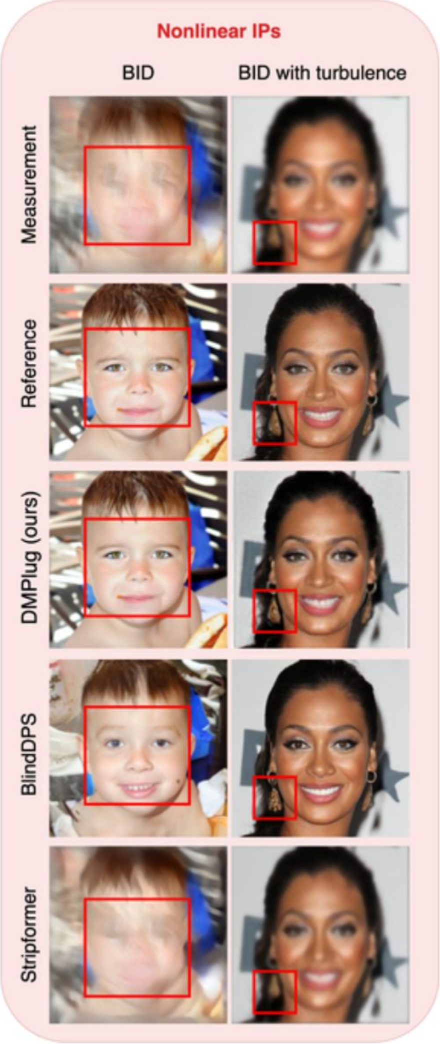
Issue I: Measurement feasibility





Issue 2: Manifold feasibility





Issue 3: Robustness to unknown noise

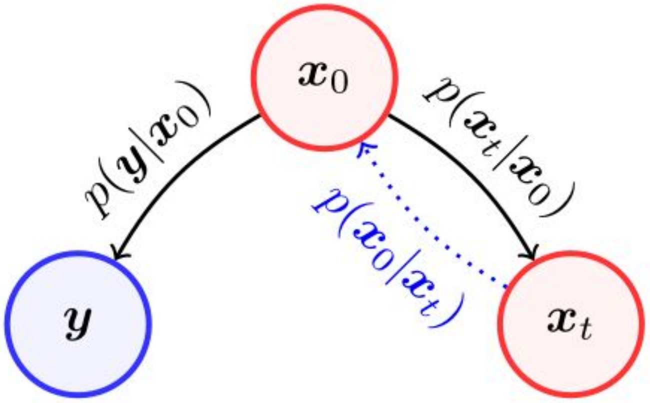


Figure 2: Probabilistic graph. Black solid line: tractable, blue dotted line: intractable in general.

Algorithm 1 DPS - Gaussian

Require: $N, \boldsymbol{y}, \{\zeta_i\}_{i=1}^N, \{\tilde{\sigma}_i\}_{i=1}^N$

$\boldsymbol{x}_N \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$

for $i = N-1$ **to** 0 **do**

$\quad \hat{\boldsymbol{s}} \leftarrow \boldsymbol{s}_\theta(\boldsymbol{x}_i, i)$

$\quad \hat{\boldsymbol{x}}_0 \leftarrow \frac{1}{\sqrt{\bar{\alpha}_i}}(\boldsymbol{x}_i + (1-\bar{\alpha}_i)\hat{\boldsymbol{s}})$

$\quad \boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$

$\quad \boldsymbol{x}'_{i-1} \leftarrow \frac{\sqrt{\alpha_i}(1-\bar{\alpha}_{i-1})}{1-\bar{\alpha}_i}\boldsymbol{x}_i + \frac{\sqrt{\bar{\alpha}_{i-1}}\beta_i}{1-\bar{\alpha}_i}\hat{\boldsymbol{x}}_0 + \tilde{\sigma}_i \boldsymbol{z}$

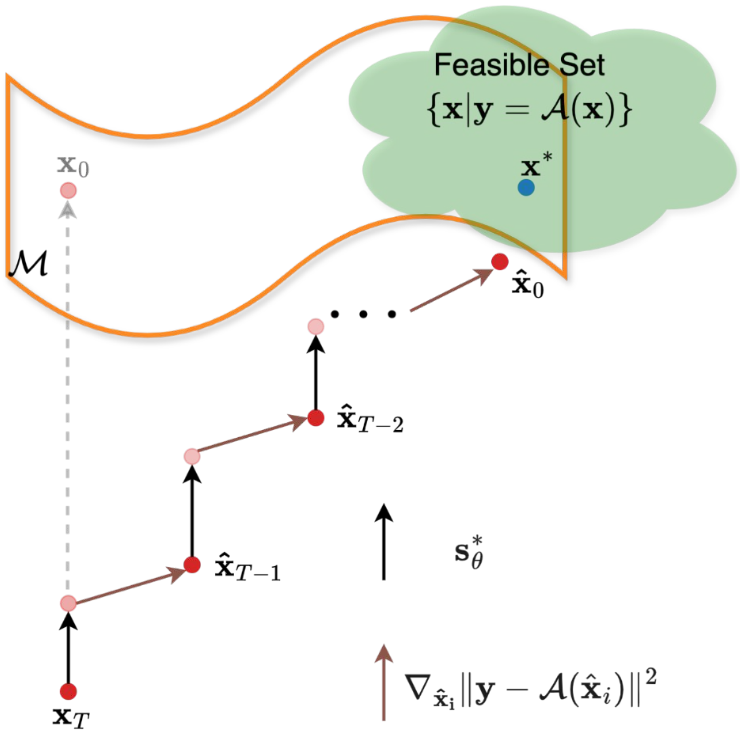
$\quad \boldsymbol{x}_{i-1} \leftarrow \boldsymbol{x}'_{i-1} - \zeta_i \nabla_{\boldsymbol{x}_i} \|\boldsymbol{y} - \mathcal{A}(\hat{\boldsymbol{x}}_0)\|_2^2$

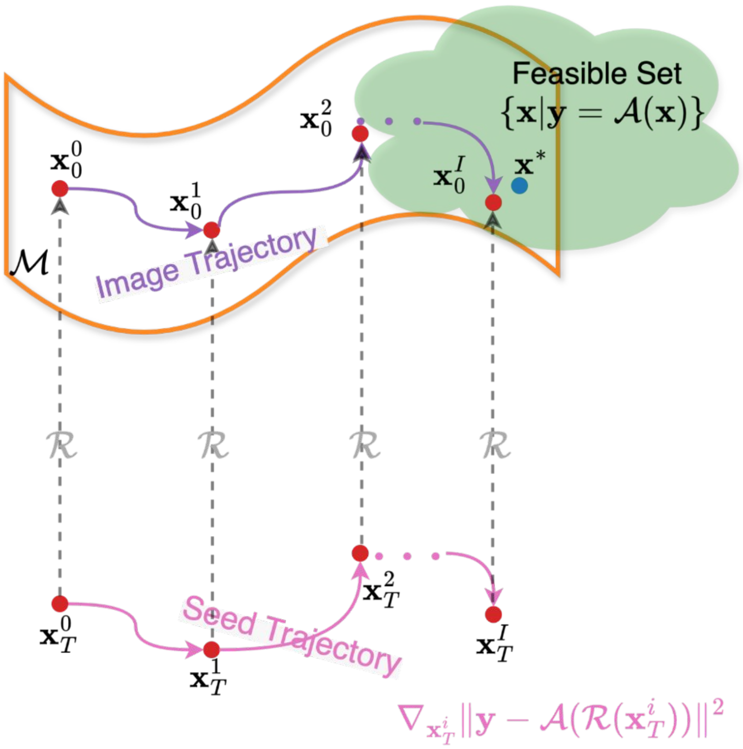
end for

return $\hat{\mathrm{x}}_0$

depending on noise level

Our solution: DMPlug





Our solution: DMPlug

Viewing the reverse process as a function $\mathcal{R}$

$$\mathcal{R} = g_{\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}^{(0)}} \circ g_{\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}^{(1)}} \circ \cdots \circ g_{\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}^{(T-2)}} \circ g_{\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}^{(T-1)}}. \qquad (\circ \text{ means function composition})$$

$$(\textbf{DMPlug}) \; \boldsymbol{z}^* \in \arg\min_{\boldsymbol{z}} \; \ell(\boldsymbol{y}, \mathcal{A}(\mathcal{R}(\boldsymbol{z}))) + \Omega(\mathcal{R}(\boldsymbol{z})), \qquad \boldsymbol{x}^* = \mathcal{R}(\boldsymbol{z}^*).$$

Measurement feasibility

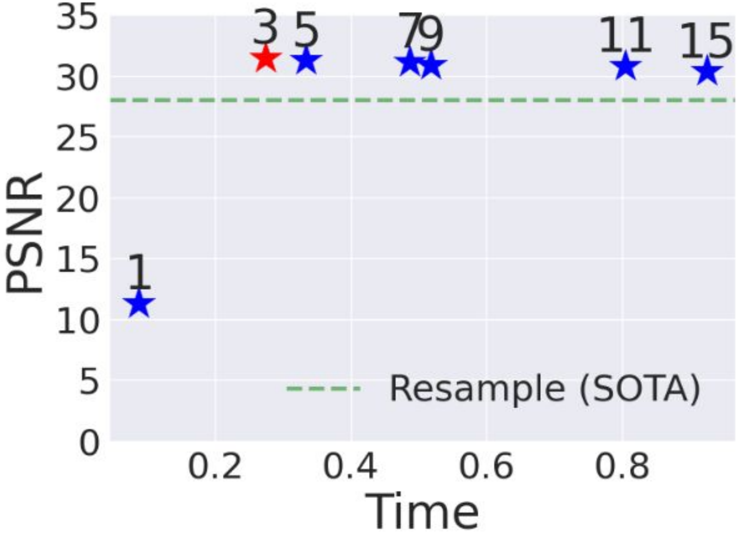
Manifold feasibility

Overcoming the computational bottleneck

$$\mathcal{R} = g_{\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}^{(0)}} \circ g_{\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}^{(1)}} \circ \cdots \circ g_{\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}^{(T-2)}} \circ g_{\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}^{(T-1)}}. \qquad (\circ \text{ means function composition})$$

$$(\mathbf{DMPlug}) \ \boldsymbol{z}^* \in \arg\min_{\boldsymbol{z}} \ \ell(\boldsymbol{y}, \mathcal{A}(\mathcal{R}(\boldsymbol{z}))) + \Omega(\mathcal{R}(\boldsymbol{z})), \qquad \boldsymbol{x}^* = \mathcal{R}(\boldsymbol{z}^*).$$

Issue: T blocks of DNNs involved, and we have to back-propagate through it



On linear IPs

Table 1: (Linear IPs) **Super-resolution** and **inpainting** with additive Gaussian noise ($\sigma = 0.01$). (**Bold**: best, under: second best, green: performance increase, red: performance decrease)

	Super-resolution ($4\times$)						Inpainting (Random 70%)					
	CelebA [65] (256×256)			FFHQ [66] (256×256)			CelebA [65] (256×256)			FFHQ [66] (256×256)		
	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑
ADMM-PnP [68]	0.217	26.99	0.808	0.229	26.25	0.794	0.091	31.94	0.923	0.104	30.64	0.901
DMPS [29]	0.070	28.89	0.848	**0.076**	28.03	0.843	0.297	24.52	0.693	0.326	23.31	0.664
DDRM [32]	0.226	26.34	0.754	0.282	25.11	0.731	0.185	26.10	0.712	0.201	25.44	0.722
MCG [30]	0.725	19.88	0.323	0.786	18.20	0.271	1.283	10.16	0.049	1.276	10.37	0.050
ILVR [41]	0.322	21.63	0.603	0.360	20.73	0.570	0.447	15.82	0.484	0.483	15.10	0.450
DPS [19]	0.087	28.32	0.823	0.098	27.44	0.814	0.043	32.24	0.924	0.046	30.95	0.913
ReSample [20]	0.080	28.29	0.819	0.108	25.22	0.773	**0.039**	30.12	0.904	0.044	27.91	0.884
DMPlug (ours)	**0.067**	**31.25**	**0.878**	0.079	**30.25**	**0.871**	**0.039**	**34.03**	**0.936**	**0.038**	**33.01**	**0.931**
Ours vs. Best compe.	-0.003	$+2.36$	$+0.030$	$+0.003$	$+2.22$	$+0.028$	-0.000	$+1.79$	$+0.012$	-0.006	$+2.06$	$+0.018$

On nonlinear IPs

Table 2: (Nonlinear IP) **Nonlinear deblurring** with additive Gaussian noise ($\sigma = 0.01$). (**Bold**: best, under: second best, green: performance increase, red: performance decrease)

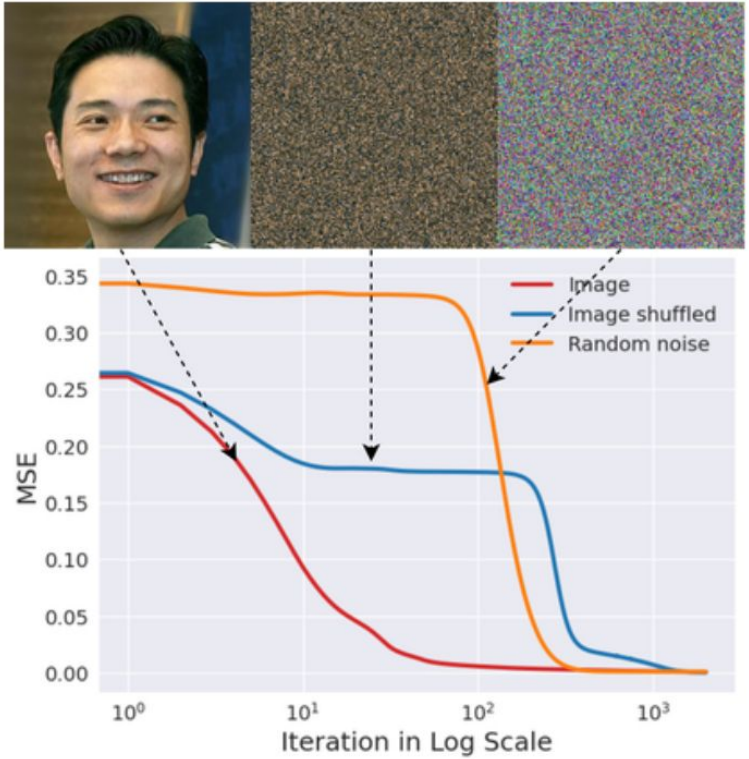
	CelebA [65] (256 × 256)			FFHQ [66] (256 × 256)			LSUN [67] (256 × 256)		
	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑
BKS-styleGAN [69]	1.047	22.82	0.653	1.051	22.07	0.620	0.987	20.90	0.538
BKS-generic [69]	1.051	21.04	0.591	1.056	20.76	0.583	0.994	18.55	0.481
MCG [30]	0.705	13.18	0.135	0.675	13.71	0.167	0.698	14.28	0.188
ILVR [41]	0.335	21.08	0.586	0.374	20.40	0.556	0.482	18.76	0.444
DPS [19]	0.149	24.57	0.723	0.130	25.00	0.759	0.244	23.46	0.684
ReSample [20]	0.104	28.52	0.839	0.104	27.02	0.834	0.143	26.03	0.803
DMPlug (ours)	**0.073**	**31.61**	**0.882**	**0.057**	**32.83**	**0.907**	**0.083**	**30.74**	**0.882**
Ours vs. Best compe.	−0.031	+3.09	+0.043	−0.047	+5.79	+0.073	−0.060	+4.71	+0.079

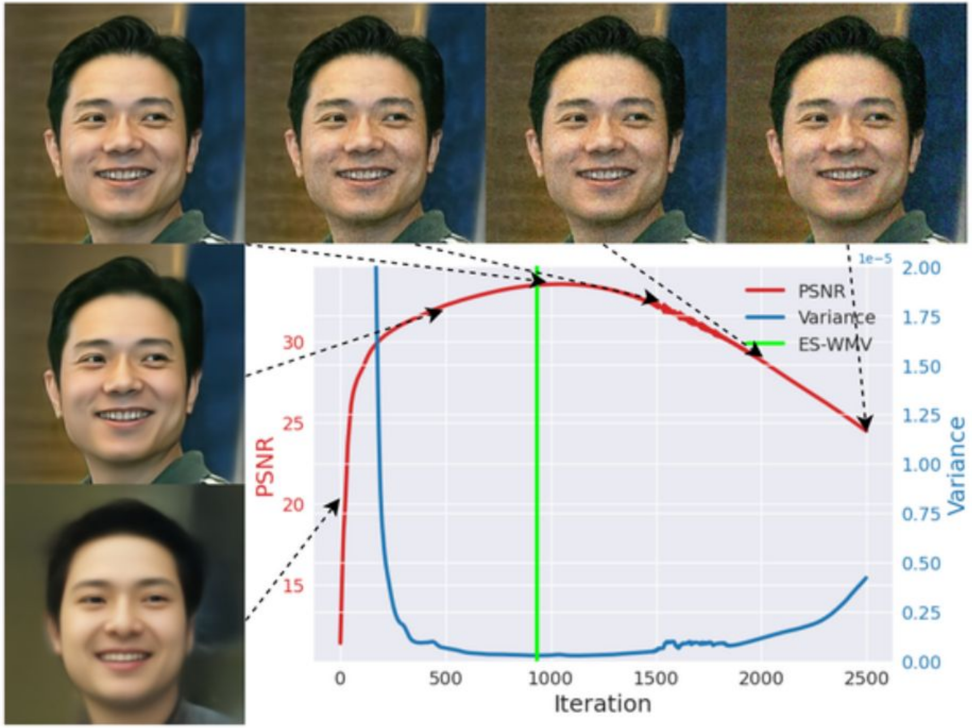
More on nonlinear IPs

Table 4: (Nonlinear IP) **BID** with additive Gaussian noise ($\sigma = 0.01$). (**Bold**: best, under: second best, green: performance increase, red: performance decrease)

	CelebA [65] (256 × 256)						FFHQ [66] (256 × 256)					
	Motion blur			**Gaussian blur**			**Motion blur**			**Gaussian blur**		
	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑
SelfDeblur [75]	0.568	16.59	0.417	0.579	16.55	0.423	0.628	16.33	0.408	0.604	16.22	0.410
DeBlurGANv2 [5]	0.313	20.56	0.613	0.350	24.29	0.743	0.353	19.67	0.581	0.374	23.58	0.726
Stripformer [6]	0.287	22.06	0.644	0.316	25.03	0.747	0.324	21.31	0.613	0.339	24.34	0.728
MPRNet [7]	0.332	20.53	0.620	0.375	22.72	0.698	0.373	19.70	0.590	0.394	22.33	0.685
Pan-DCP [73]	0.606	15.83	0.483	0.653	20.57	0.701	0.616	15.59	0.464	0.667	20.69	0.698
Pan-ℓ_0 [74]	0.631	15.16	0.470	0.654	20.49	0.675	0.642	14.43	0.443	0.669	20.34	0.671
ILVR [41]	0.398	19.23	0.520	0.338	21.20	0.588	0.445	18.33	0.484	0.375	20.45	0.555
BlindDPS [21]	0.164	23.60	0.682	0.173	25.15	0.721	0.185	21.77	0.630	0.193	23.83	0.693
DMPlug (ours)	**0.104**	**29.61**	**0.825**	**0.140**	**28.84**	**0.795**	**0.135**	**27.99**	**0.794**	**0.169**	**28.26**	**0.811**
Ours vs. Best compe.	−0.060	+6.01	+0.143	−0.033	+3.69	+0.048	−0.050	+6.22	+0.164	−0.024	+3.92	+0.083

How to achieve robustness to unknown noise?





Early-learning-then-overfitting (OLTO)

Algorithm 3 DMPlug+ES–WMV for solving general IPs

Input: # diffusion steps T, $\boldsymbol{y}$, window size W, patience P, empty queue $\mathcal{Q}$, iteration counter $e = 0$, $\mathrm{VAR}_{\min} = \infty$

while not stopped **do**
    for $i = T - 1$ to 0 **do**
        $\hat{\boldsymbol{s}} \leftarrow \boldsymbol{\varepsilon}_{\boldsymbol{\theta}}^{(i)}(\boldsymbol{z}_i^e)$
        $\hat{\boldsymbol{z}}_0^e \leftarrow \frac{1}{\sqrt{\bar{\alpha}_i}}\left(\boldsymbol{z}_i^e - \sqrt{1-\bar{\alpha}_i}\hat{\boldsymbol{s}}\right)$
        $\boldsymbol{z}_{i-1}^e \leftarrow$ DDIM reverse with $\hat{\boldsymbol{z}}_0^e$, $\hat{\boldsymbol{s}}$
    end for
    Update $\boldsymbol{z}_T^{e+1}$ from $\boldsymbol{z}_T^e$ via a gradient update for Eq. (7)
    push $\mathcal{R}(\boldsymbol{z}_T^{e+1})$ to $\mathcal{Q}$, pop queue if $|\mathcal{Q}| > W$
    if $|\mathcal{Q}| = W$ **then**
        compute VAR of elements in $\mathcal{Q}$ via Eq. (15)
        if $\mathrm{VAR} < \mathrm{VAR}_{\min}$ **then**
            $\mathrm{VAR}_{\min} \leftarrow \mathrm{VAR}$, $\boldsymbol{z}^* \leftarrow \boldsymbol{z}_T^{e+1}$
        end if
        if $\mathrm{VAR}_{\min}$ stagnates for P iterations **then**
            stop and return $\boldsymbol{z}^*$
        end if
    end if
    $e = e + 1$
end while

Output: Recovered object $\mathcal{R}(\boldsymbol{z}^*)$

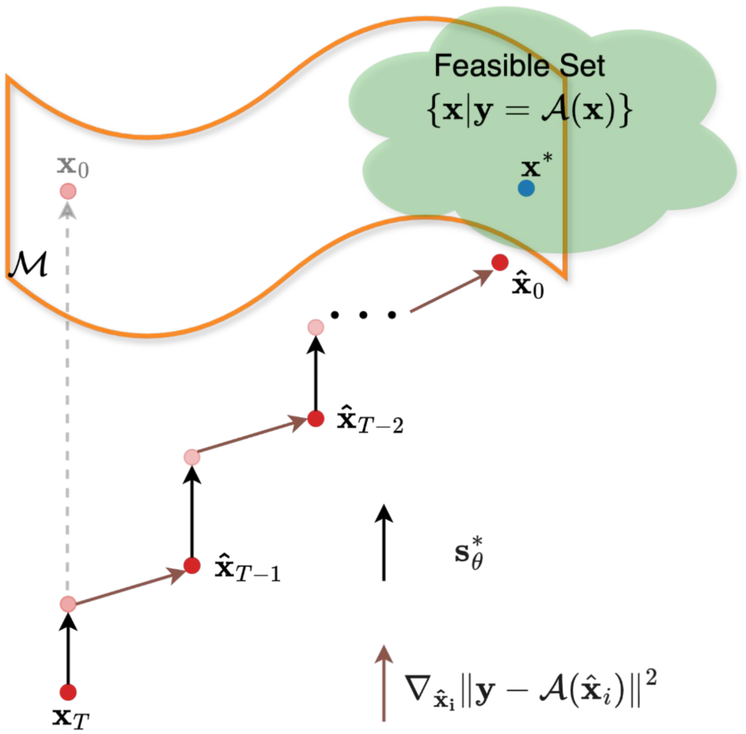
Early stopping based on running variance

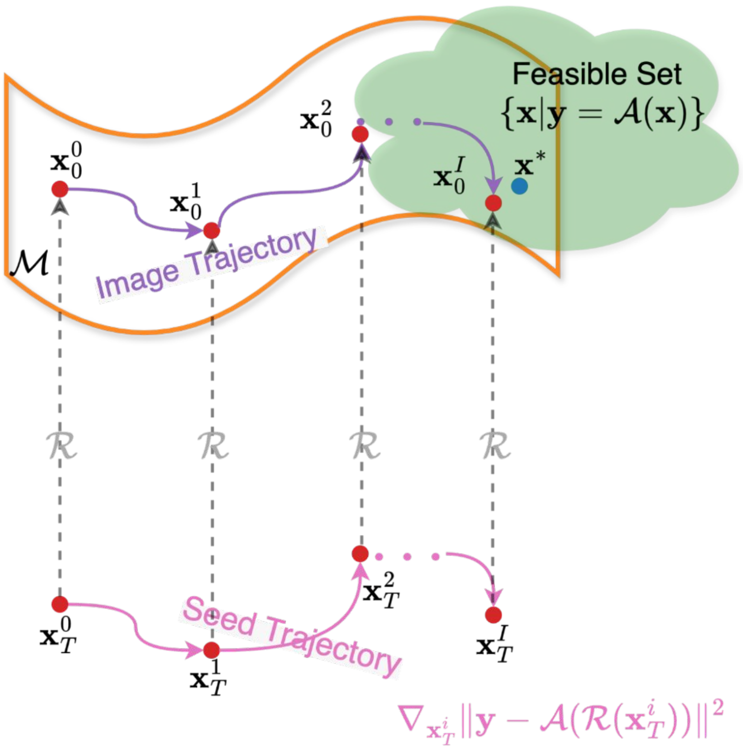
Robustness to unknown noise

Table 5: (**Robustness and ES**) **Super-resolution** and **nonlinear deblurring** on CelebA [65] with different types and levels of noise. We only show PSNR↑ and PSNR Gap↓ to save space. (**Bold**: best, under: second best, green: performance increase, red: performance decrease)

	(Linear) Super-resolution (4×)				(Nonlinear) Non-uniform image deblurring			
	Gaussian	**Impulse**	**Shot**	**Speckle**	**Gaussian**	**Impulse**	**Shot**	**Speckle**
	Low/High	Low/High	Low/High	Low/High	Low/High	Low/High	Low/High	Low/High
ADMM-PnP [68]	20.17/17.97	14.28/14.52	19.97/17.82	19.42/18.41	N/A	N/A	N/A	N/A
DMPS [29]	20.62/17.54	18.78/16.05	19.96/16.74	20.77/18.73	N/A	N/A	N/A	N/A
DDRM [32]	15.45/14.79	14.82/14.14	15.31/14.59	15.46/15.03	N/A	N/A	N/A	N/A
MCG [30]	17.43/15.83	16.39/15.07	17.19/15.49	17.44/16.43	12.88/12.85	13.16/13.04	13.21/13.13	13.24/13.07
ILVR [41]	21.08/21.03	20.93/20.00	21.19/21.12	20.96/20.89	21.70/21.43	21.43/21.00	21.56/21.24	21.53/21.36
DPS [19]	25.51/24.58	24.89/23.92	25.47/24.27	25.69/24.97	23.97/23.35	23.74/23.18	24.32/23.58	23.45/23.61
ReSample [20]	14.30/13.04	15.56/13.48	14.38/12.87	15.64/14.23	23.17/20.45	20.69/18.91	22.94/20.11	23.59/21.66
BKS-styleGAN [69]	N/A	N/A	N/A	N/A	22.61/22.53	22.64/22.34	22.96/22.79	22.70/22.56
BKS-generic [69]	N/A	N/A	N/A	N/A	16.85/15.09	14.86/13.44	16.69/14.74	17.04/15.99
DMPlug (ours)	**26.49/25.29**	**26.01/24.76**	**26.34/26.34**	**26.81/25.81**	**27.58/26.60**	**27.22/26.13**	**27.71/26.55**	**27.68/26.96**
Ours vs. Best compe.	0.98/0.71	1.12/0.84	0.87/2.07	1.12/0.84	3.61/3.25	3.48/2.95	3.39/2.97	4.23/3.35
PSNR Gap↓	0.36/0.46	0.38/0.60	0.25/0.49	0.20/0.21	0.15/0.12	0.14/0.13	0.10/0.19	0.12/0.09

DMPlug to get everything right





The paper (NeurIPS'24)

[Submitted on 27 May 2024]

DMPlug: A Plug-in Method for Solving Inverse Problems with Diffusion Models

Hengkang Wang, Xu Zhang, Taihui Li, Yuxiang Wan, Tiancong Chen, Ju Sun

Pretrained diffusion models (DMs) have recently been popularly used in solving inverse problems (IPs). The existing methods mostly interleave iterative steps in the reverse diffusion process and iterative steps to bring the iterates closer to satisfying the measurement constraint. However, such interleaving methods struggle to produce final results that look like natural objects of interest (i.e., manifold feasibility) and fit the measurement (i.e., measurement feasibility), especially for nonlinear IPs. Moreover, their capabilities to deal with noisy IPs with unknown types and levels of measurement noise are unknown. In this paper, we advocate viewing the reverse process in DMs as a function and propose a novel plug-in method for solving IPs using pretrained DMs, dubbed DMPlug. DMPlug addresses the issues of manifold feasibility and measurement feasibility in a principled manner, and also shows great potential for being robust to unknown types and levels of noise. Through extensive experiments across various IP tasks, including two linear and three nonlinear IPs, we demonstrate that DMPlug consistently outperforms state-of-the-art methods, often by large margins especially for nonlinear IPs. The code is available at this https URL.

https://arxiv.org/abs/2405.16749

DMPlug for video restoration

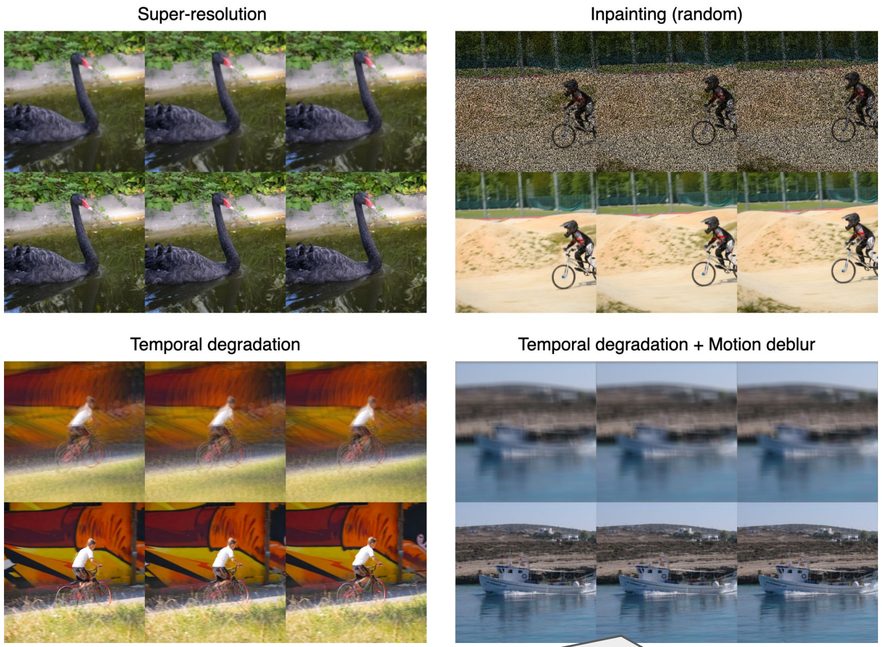
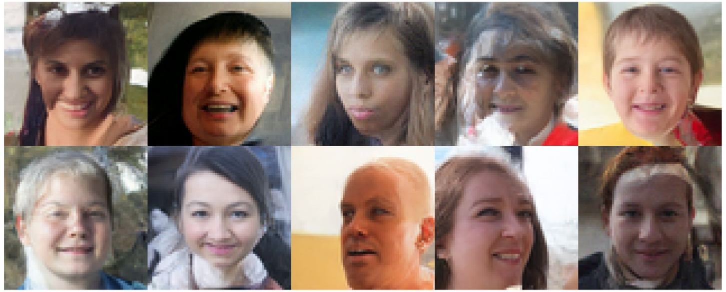


Table 7. Ablation study on two essential components for multi-level temporal consistency, performed on DAVIS dataset for video super-resolution ×4. (**Bold**: best, under: second best)

Method	PSNR↑	SSIM↑	LPIPS↓	WE(10^{-2})↓
SOTA [9]	26.037	0.717	0.339	1.411
Base	24.701	0.612	0.366	1.398
Base + Semantic	26.098	0.703	0.410	1.057
Base + Pixel	27.141	0.736	**0.301**	0.943
Base + Semantic + Pixel	**27.959**	**0.790**	0.321	**0.725**

Wang et al. **Temporal-Consistent Video Restoration with Pre-trained Diffusion Models**. Forthcoming, 2025

Train diffusion models in small-data regime?


(a) Full Gaussian (ambient dimension $d = 12288$)


(b) Restricted Gaussian (subspace dimension $k = 2048$)


(a) Full Gaussian (ambient dimension $d = 16384$, FID-50K=5.09)


(b) Restricted Gaussian (subspace dimension $k = 8192$, FID-50K=3.21)

Luo et al. **Small-Data Flow Matching**. Forthcoming, 2025