A personal journey toward

Diffusion Models for Inverse Problems in Medical Imaging

Jong Chul Ye

Professor
Graduate School of Artificial Intelligence
KAIST, Korea

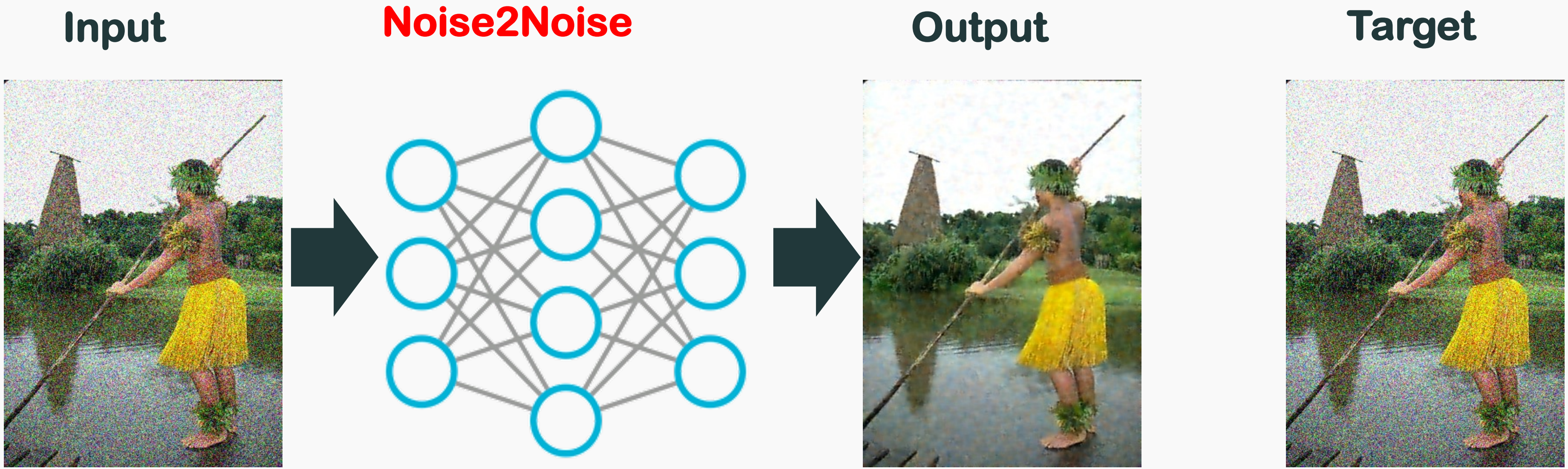
Future of AI? Yann LeCun's Cake Analogy

- "Pure" Reinforcement Learning (cherry)
  - The machine predicts a scalar reward given once in a while.
  - **A few bits for some samples**
- Supervised Learning (icing)
  - The machine predicts a category or a few numbers for each input
  - Predicting human-supplied data
  - **10→10,000 bits per sample**
- Unsupervised/Predictive Learning (cake)
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**

- (Yes, I know, this picture is slightly offensive to RL folks. But I'll make it up)

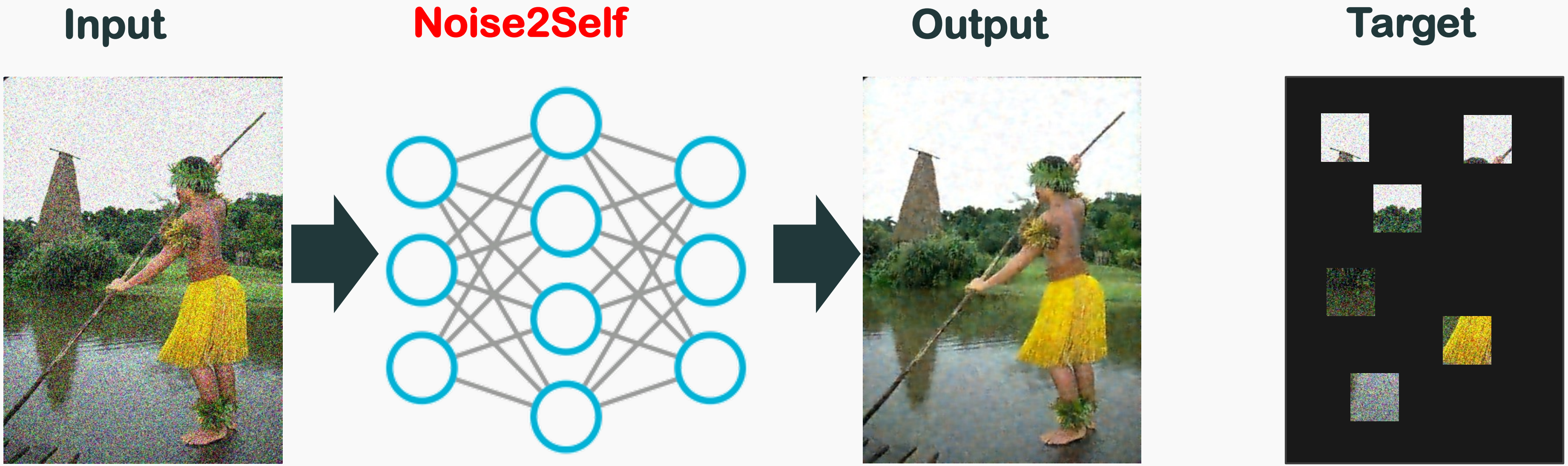
Slide courtesy of Yann LeCun's Keynote talk

Self-Supervised Denoising: Noise2X



Lehtinen et al, ICML 2018

Self-Supervised Denoising: Noise2X



Batson et al, ICML 2019

Self-Supervised Denoising: SURE

Soltanayev et al, NeurIPS, 2018

Self-supervised denoising using Stein Unsupervised Risk Estimator (SURE)

$$\ell_{SURE}(\Theta) = \mathbb{E}_{y \sim P_Y} d(y, F_\Theta(y))$$

$$d(y, F_\Theta(y)) := \underbrace{\|y - F_\Theta(y)\|^2}_{\text{Autoencoder loss}} + \underbrace{2\sigma^2 \mathrm{div}_y F_\Theta}_{\text{Divergence-based penalty}}$$

Autoencoder loss

Divergence-based penalty

Any unified mathematical framework?

Tweedie's Formula for Exponential Family Distribution

B Efron, Journal of the American Statistical Association, 2011

- **Probability distribution of exponential family:**

$$p(y|\eta) = p_0(y)\exp(\underline{\eta^\top T(y)} - \varphi(\eta))$$

- **Using the Bayes' rule, the posterior density:**

$$p(\eta|y) = \exp(\underline{\eta^\top T(y)} - \lambda(y))[p(\eta)e^{-\varphi(\eta)}], \quad \text{where} \quad \lambda(y) = \log\frac{p(y)}{p_0(y)}$$

Tweedie's Formula for Exponential Family Distribution

Kim and Ye, NeurIPS, 2021

- Tweedie's formula → **Posterior mean**

- The closed form solution for the posterior mean:

$$
\begin{aligned}
\hat{\eta}^\top T'(y) &= \lambda'(y) = -\nabla_y \log p_0(y) + \nabla_y \log p(y) \\
&= -l_0'(y) + l'(y)
\end{aligned}
$$

$$\ell'(y) = \nabla_y \log p(y)$$

Score function

Tweedie's Formula for Exponential Family Distribution

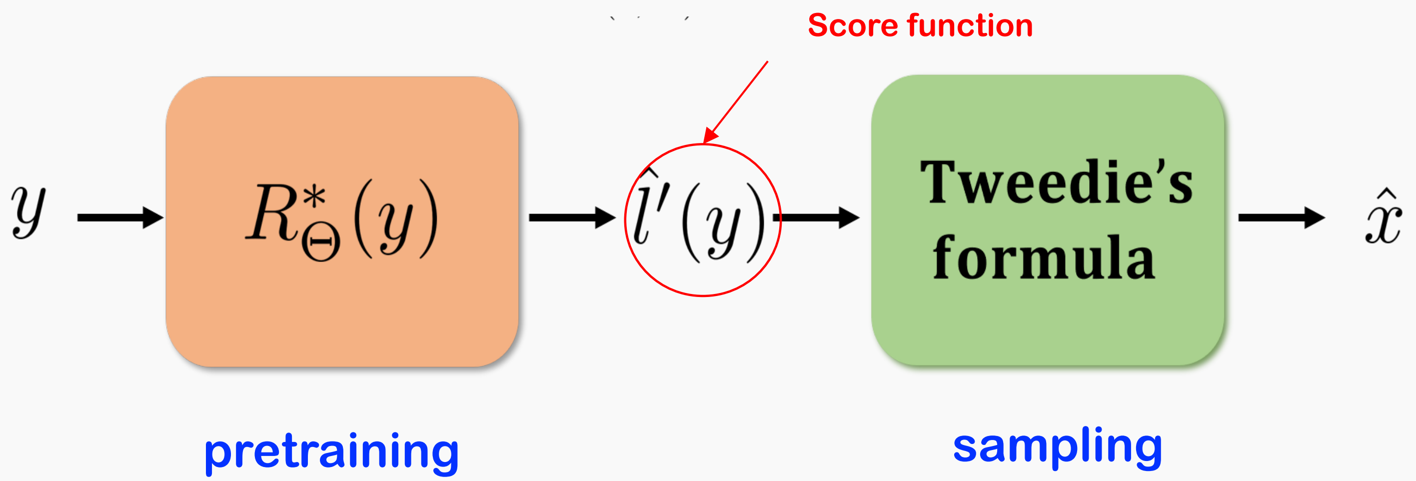
Kim and Ye, NeurIPS, 2021

Distribution	$p(y\|x)$	η	$T(y)$	$p_0(y)$	$l_0'(y)$	$\hat{x}$
Gaussian	$\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(y-x)^2}{2\sigma^2}}$	x/σ^2	y	$\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{y^2}{2\sigma^2}}$	$-\frac{y}{\sigma^2}$	$y+\sigma^2 l'(y)$
Gaussian	$\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(y-x)^2}{2\sigma^2}}$	$\left[\frac{x}{\sigma^2}, -\frac{1}{2\sigma^2}\right]^\top$	$\left[y, y^2\right]^\top$	$\frac{1}{\sqrt{2\pi}}$	0	$y+\sigma^2 l'(y)$
Poisson	$\frac{x^y e^{-x}}{y!}$	$\log(x)$	y	$\frac{1}{y!}$	$\simeq -\log\left(y+\frac{1}{2}\right)$	$\left(y+\frac{1}{2}\right)\exp(l'(y))$
Gamma(α,β)	$\frac{\beta^\alpha}{\Gamma(\alpha)}\left(\frac{y}{x}\right)^{\alpha-1}e^{-\beta\frac{y}{x}}$	$\left[\alpha-1, -\frac{\beta}{x}\right]^\top$	$[\log y, -y]^\top$	1	0	$\frac{\beta y}{(\alpha-1)-yl'(y)}$
Bernoulli	$x^y(1-x)^{(1-y)}$	$\log\left(\frac{x}{1-x}\right)$	y	1	0	$\frac{e^{l'(y)}}{1+e^{l'(y)}}$
Exponential	$xe^{-yx}, y\geq 0$	$-x$	y	1	0	$-l'(y)$

As long as we can compute the score function,
optimal denoising can be achieved by using Tweedie's formula.

Noise2Score

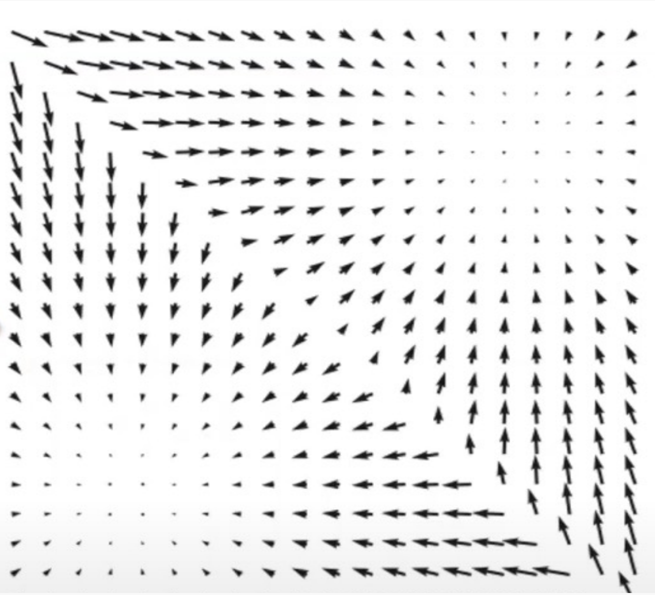
Kim and Ye, NeurIPS, 2021



How to estimate the score function?

Score Function

$p_{data}(x)$



$\nabla_x \underline{\log p_{data}(x)}$

Score function

Figure courtesy from https://yang-song.github.io/blog/2021/score/

Denoising Autoencoder (DAE)

Alain et al, JMLR, 2014

- DAE F_Θ is similar to Noise2X

$$\ell_{DAE}(\Theta) = \mathop{\mathbb{E}}_{\substack{y\sim P_Y \\ u\sim\mathcal{N}(0,I) \\ \sigma_a\sim\mathcal{N}(0,\delta^2)}} \|y - F_\Theta(y+\sigma_a u)\|^2$$

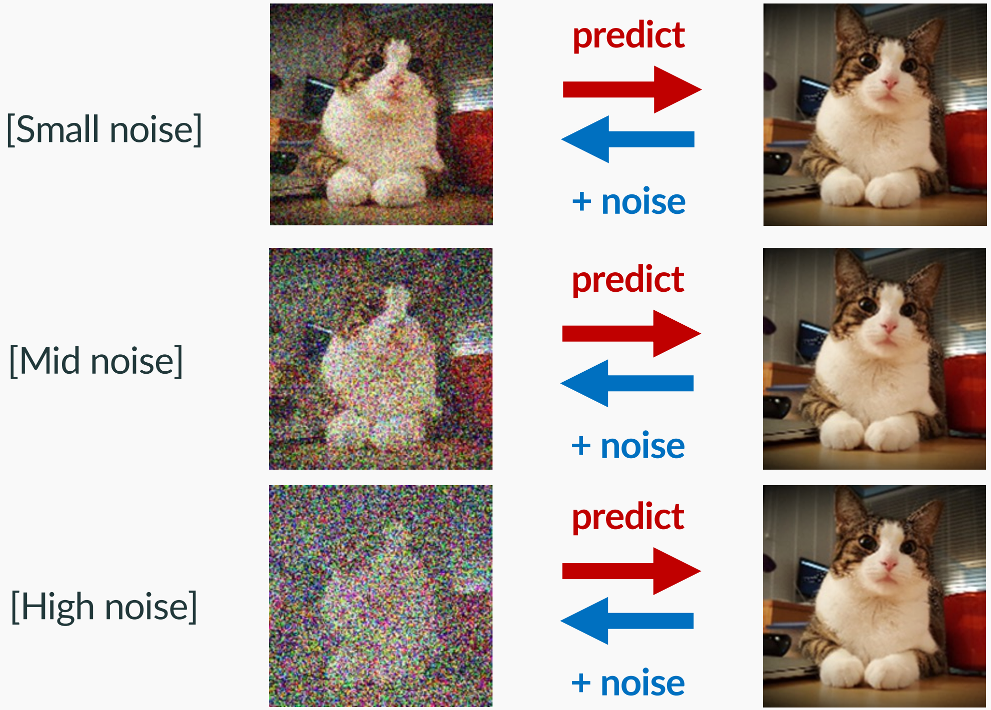
- DAE can be used to estimate the score function of data y

$$F_{\Theta^*}(y) = y + \sigma_a^2 l'(y) + o(\sigma_a^2)$$

$$\sigma_a \to 0 \quad \Longrightarrow \quad F_{\Theta^*}(y) = y + \sigma_a^2 l'(y)$$

Equal to Tweedie's formula for Gaussian noises

Denoising Autoencoder (DAE)



Slide courtesy of Jiaming Song

Relation to SURE

By using residual form: $$F_\Theta(y) = \sigma^2 R_\Theta(y) + y$$

$$\hat{l}'(y) = \frac{F_{\Theta^*}(y) - y}{\sigma_a^2} = R_\Theta(y)$$

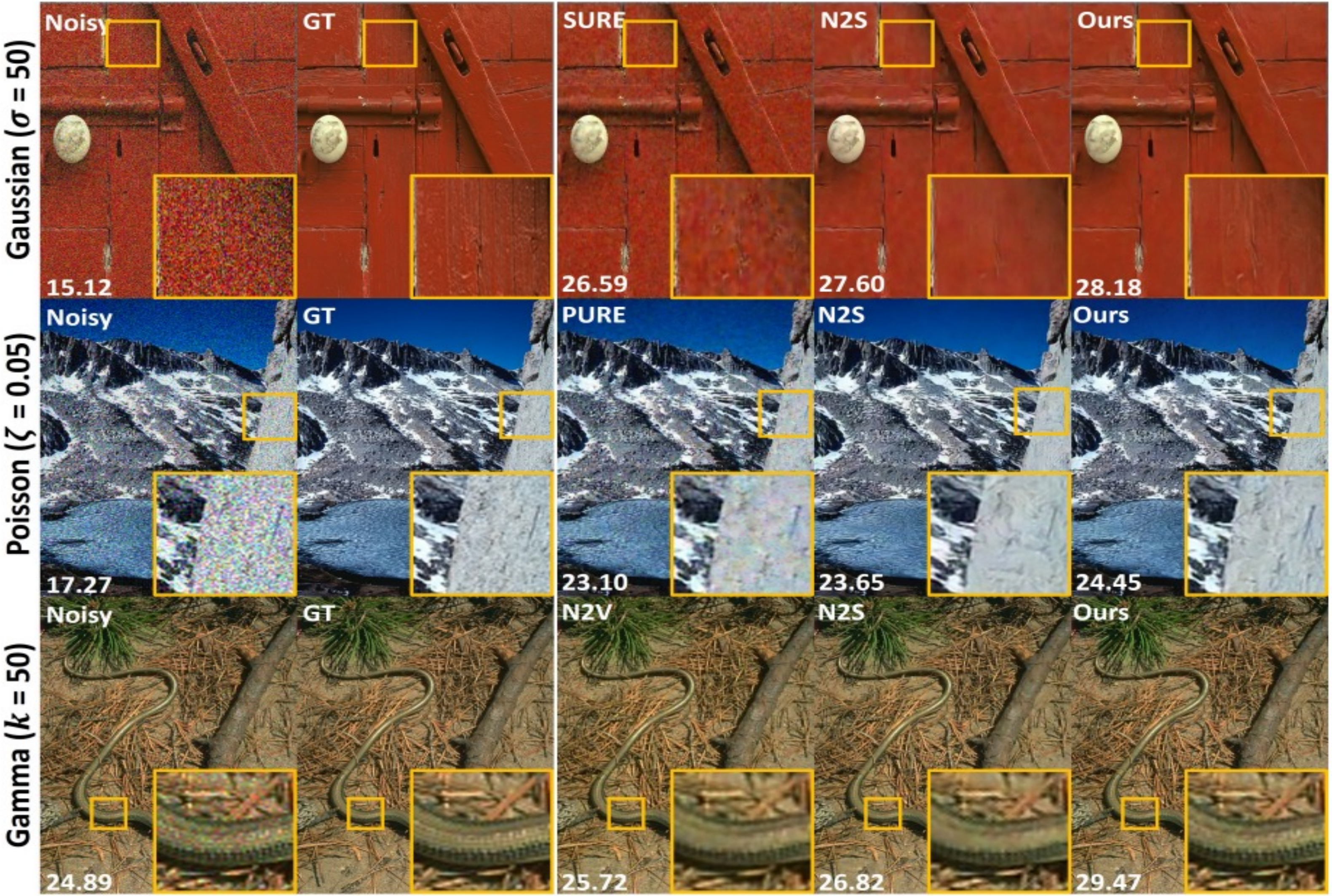
$$\begin{aligned} \ell_{SURE}(\Theta) &= \mathbb{E}_{y\sim P_Y}\left\{\|y - F_\Theta(y)\|^2 + 2\sigma^2 \mathrm{div}_y F_\Theta(y)\right\} \\ &= \mathbb{E}_{y\sim P_Y}\left\{\sigma^4\|R_\Theta(y)\|^2 + 2\sigma^4 \mathrm{div}_y R_\Theta(y)\right\} + 2\sigma^2 \mathrm{dim}(y) \end{aligned}$$

$$\|$$

$$\ell_{ISM}(\Theta) = \mathbb{E}_{y\sim P_Y}\left\{\frac{1}{2}\|\Psi_\Theta(y)\|^2 + \mathrm{div}_y \Psi_\Theta(y)\right\}$$

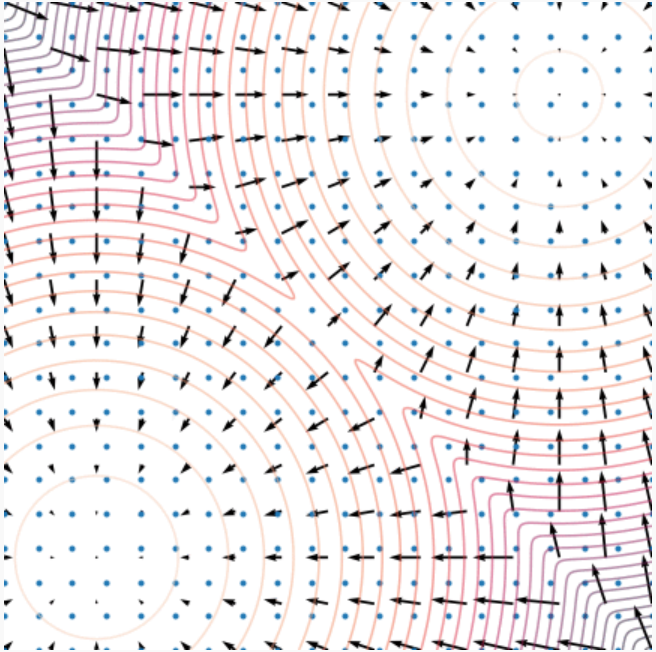
Implicit Score matching cost by **Hyvärinen et al, JMLR, 2005**

Noise2X, SURE are
score-based approaches!

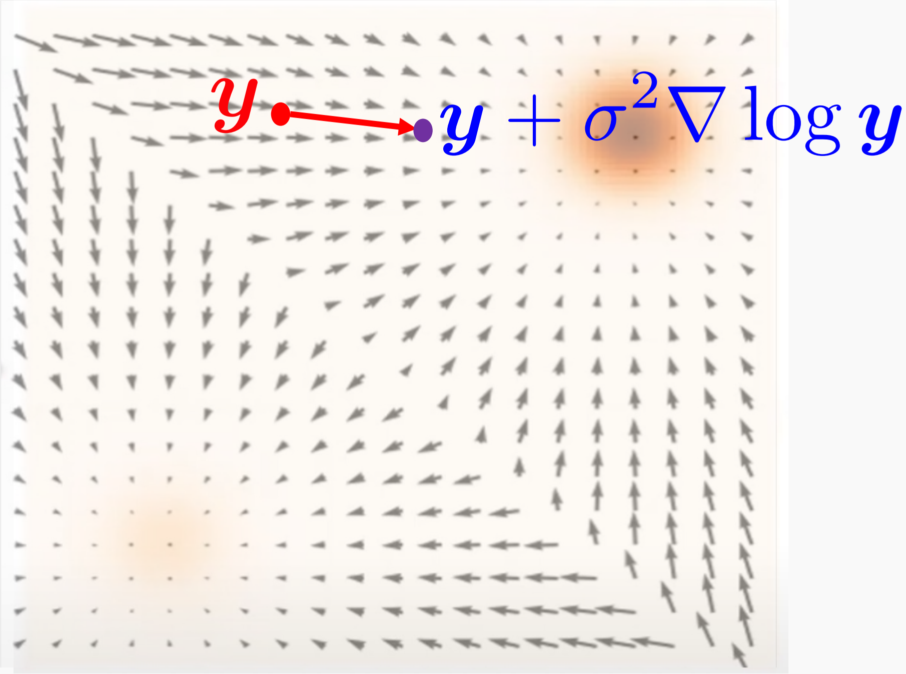

Gaussian (σ = 50)
Poisson (ζ = 0.05)
Gamma (k = 50)
Noisy
GT
SURE
N2S
Ours
15.12
26.59
27.60
28.18
Noisy
GT
PURE
N2S
Ours
17.27
23.10
23.65
24.45
Noisy
GT
N2V
N2S
Ours
24.89
25.72
26.82
29.47

Generalization beyond the one-step denoising?

Noise2Score vs Diffusion Models



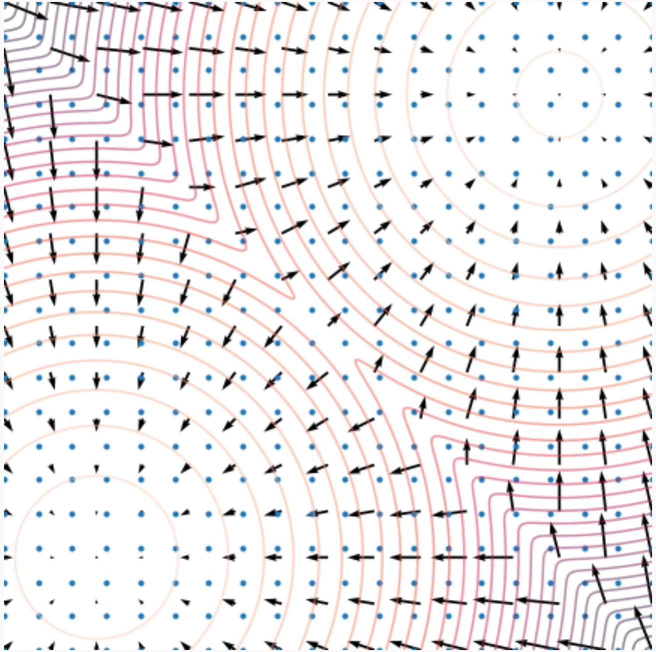
Diffusion model



Noise2Score

http://yang-song.github.io/blog/2021/score/

Method1: Score-method with Langevin Dynamics (SMLD)



- Once the score model is trained to optimality,
  - i.e. $s_\theta(\mathbf{x}) \simeq \nabla_{\mathbf{x}} p(\mathbf{x})$
- Example: Use Langevin dynamics to draw samples

$$\mathbf{x}_{i+1} \leftarrow \mathbf{x}_i + \epsilon \nabla_{\mathbf{x}} \log p(\mathbf{x}) + \sqrt{2\epsilon}\mathbf{z}_i$$

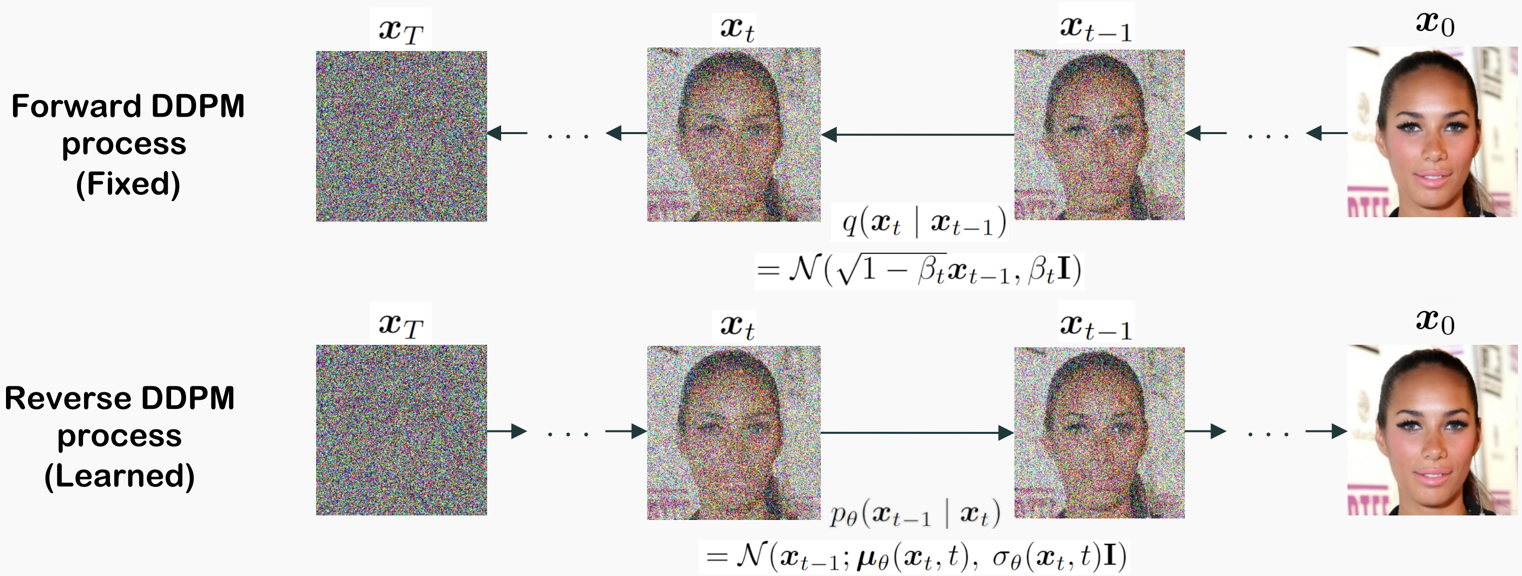
$$i = 0, 1, \dots, K$$

"Variance Exploding (VE)" parameterization

http://yang-song.github.io/blog/2021/score/

Method2: Diffusion Denoising Probabilistic Model (DDPM)

(Ho et al. NeurIPS 2020)



DDPM Training & Sampling

Algorithm 1 Training

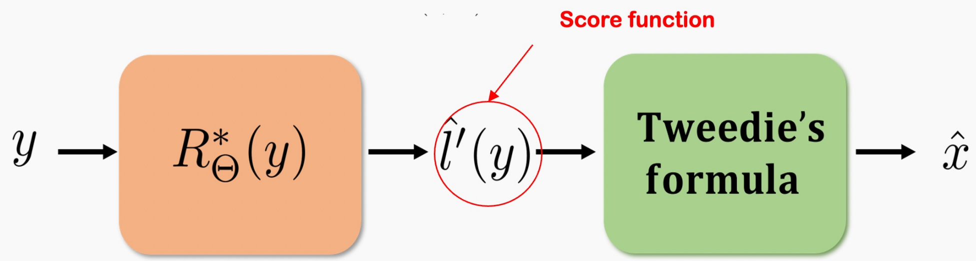
repeat
  $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
  $t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
  $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  Take gradient descent step on
  $$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
until converged

Algorithm 2 Sampling

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \ldots, 1$ **do**
  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t \mathbf{z}$
end for
return $\mathbf{x}_0$

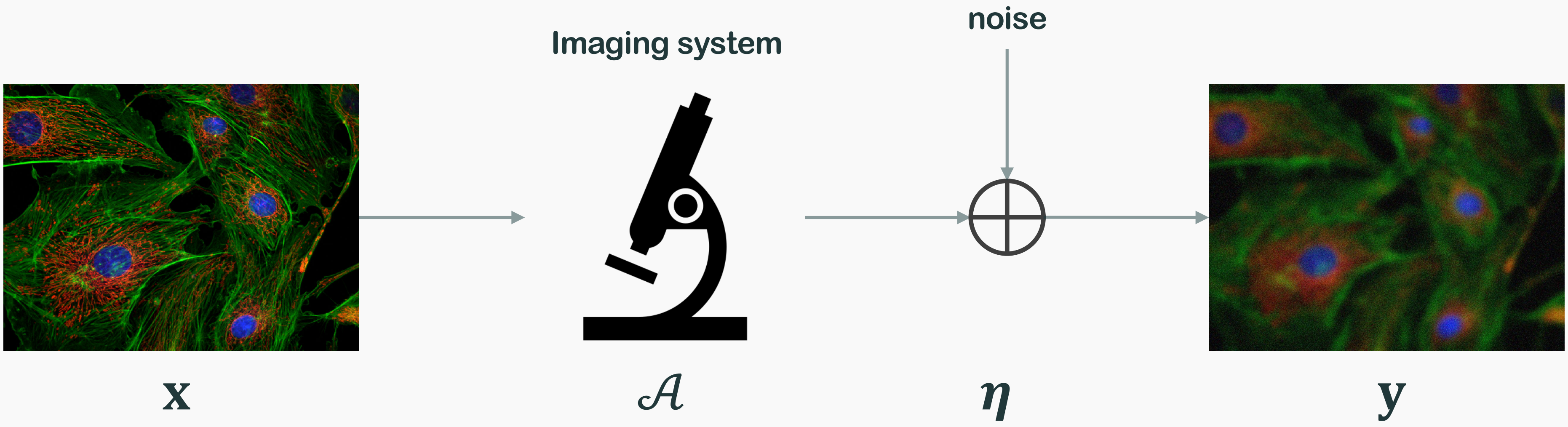
"Variance Preserving (VP)" parameterization

Cf. Noise2Score



How to use for **inverse problems?**

Typical Setup of Inverse Problems



Biomedical imaging systems cast as the above **forward model**

- Acquiring the image $\mathbf{x}$: **inverse problem**
- System naturally **ill-posed**: what is the best solution?
- Ex: microscopy, MRI, CT, optics, etc.

Slide courtesy of Song, Chung, Ye’s 2023 ISBI Tutorial

Score-based Diffusion Models for Accelerated MRI

Jung et al, MEDIA, 2022

$$\min_{\mathbf{x}} \|y - A\mathbf{x}\|^2$$

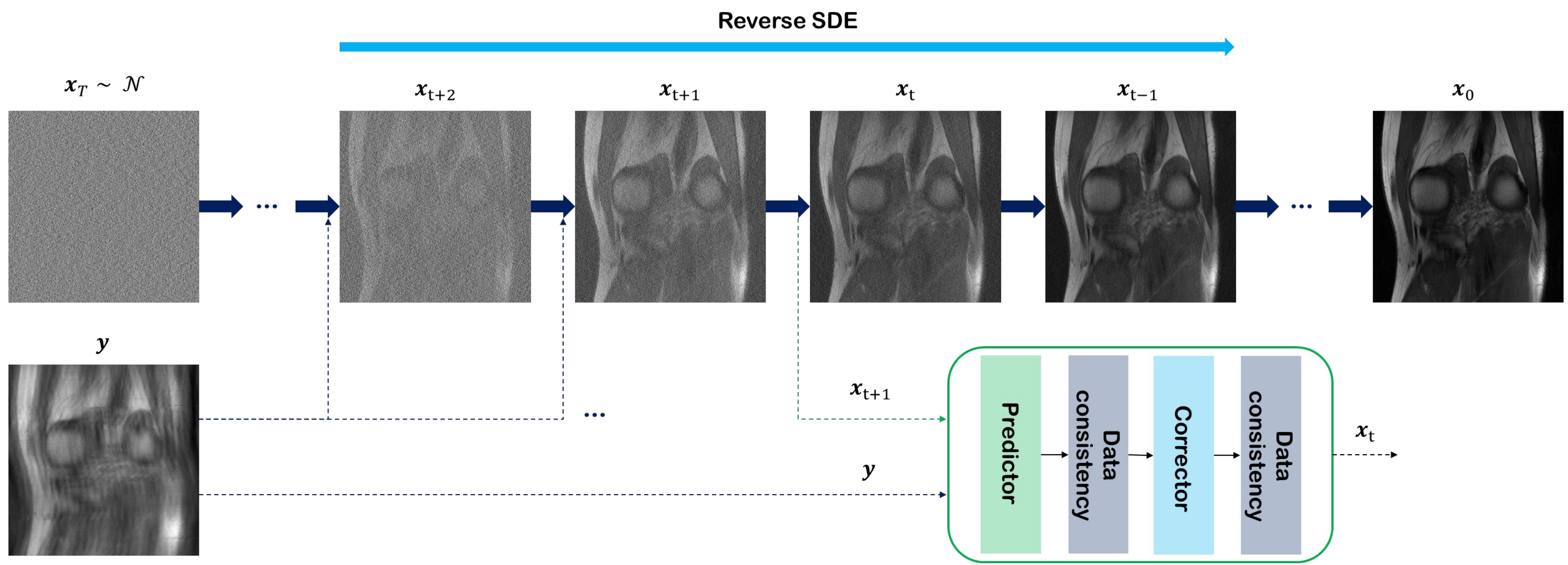
$$\boldsymbol{x}_i \leftarrow \boldsymbol{x}_{i+1} + \epsilon_i \boldsymbol{s}_\theta(\boldsymbol{x}_{i+1}, \sigma_{i+1}) + \sqrt{2\epsilon_i}\boldsymbol{z}$$

Denoising step (reverse SDE)

$$\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i + \lambda A^*(y - A\boldsymbol{x}_i),$$

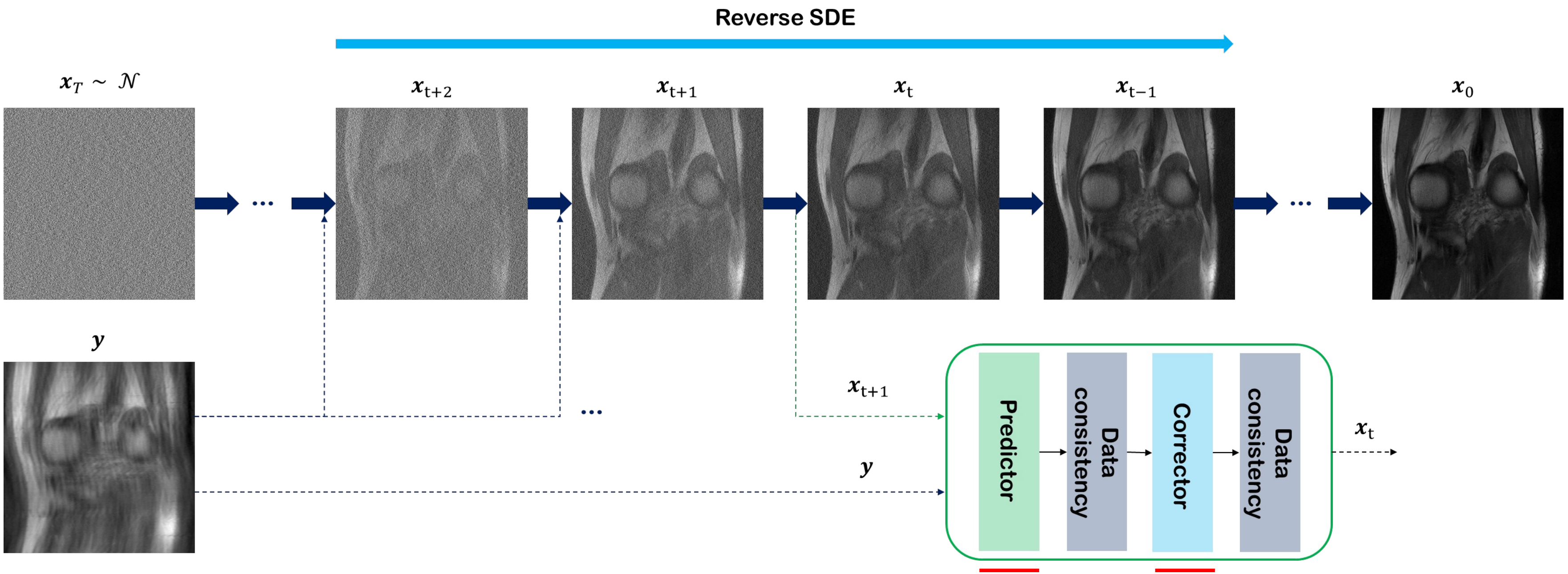
Data consistency step (e.g. GD, POCS)

Score-based Diffusion Models for Accelerated MRI



[MeDIA 2022] Chung & Ye, Score-based diffusion models for accelerated MRI

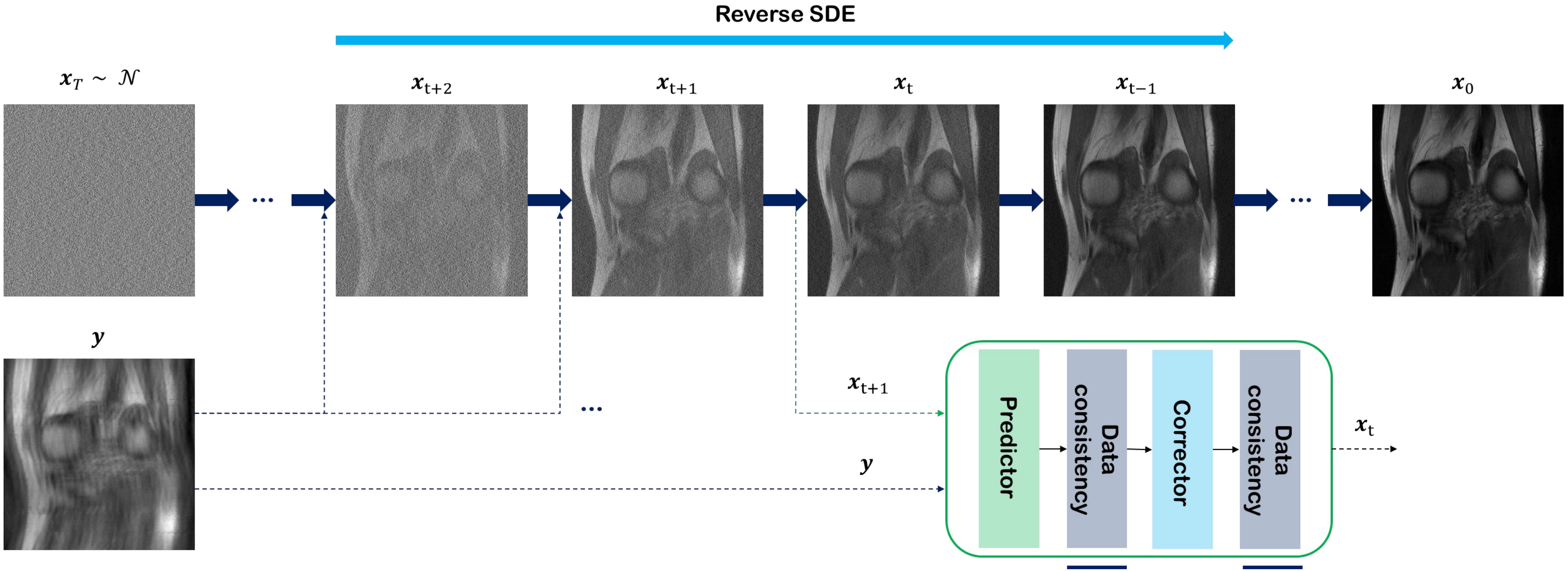
Score-based Diffusion Models for Accelerated MRI



Single step of **unconditional** denoising

[MeDIA 2022] Chung & Ye, Score-based diffusion models for accelerated MRI

Score-based Diffusion Models for Accelerated MRI

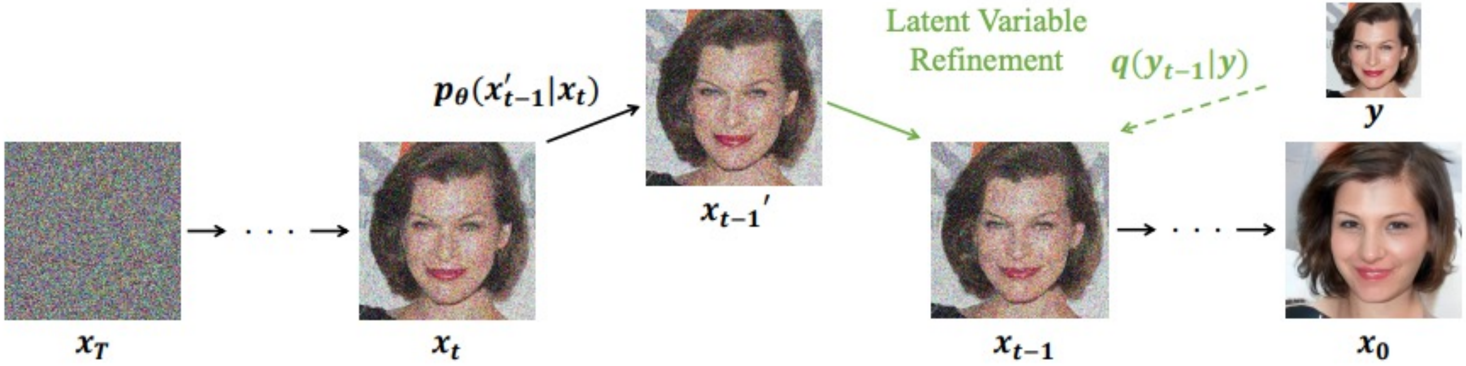


Injection of k-space info from y

[MeDIA 2022] Chung & Ye, Score-based diffusion models for accelerated MRI

ILVR: Conditioning Method for DDPM

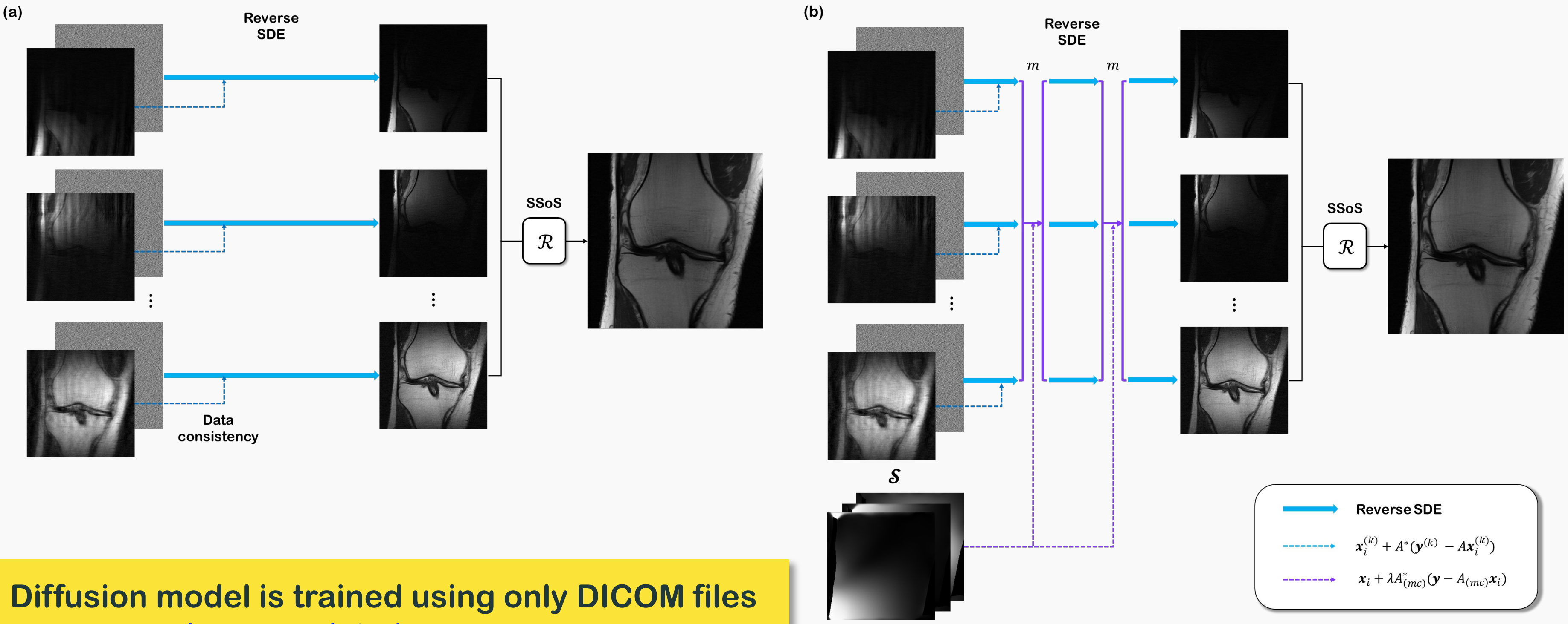
Choi et al, ICLR, 2021



$$x'_{t-1} \sim p_\theta(x'_{t-1}|x_t),$$
$$x_{t-1} = \phi(y_{t-1}) + (I - \phi)(x'_{t-1}).$$

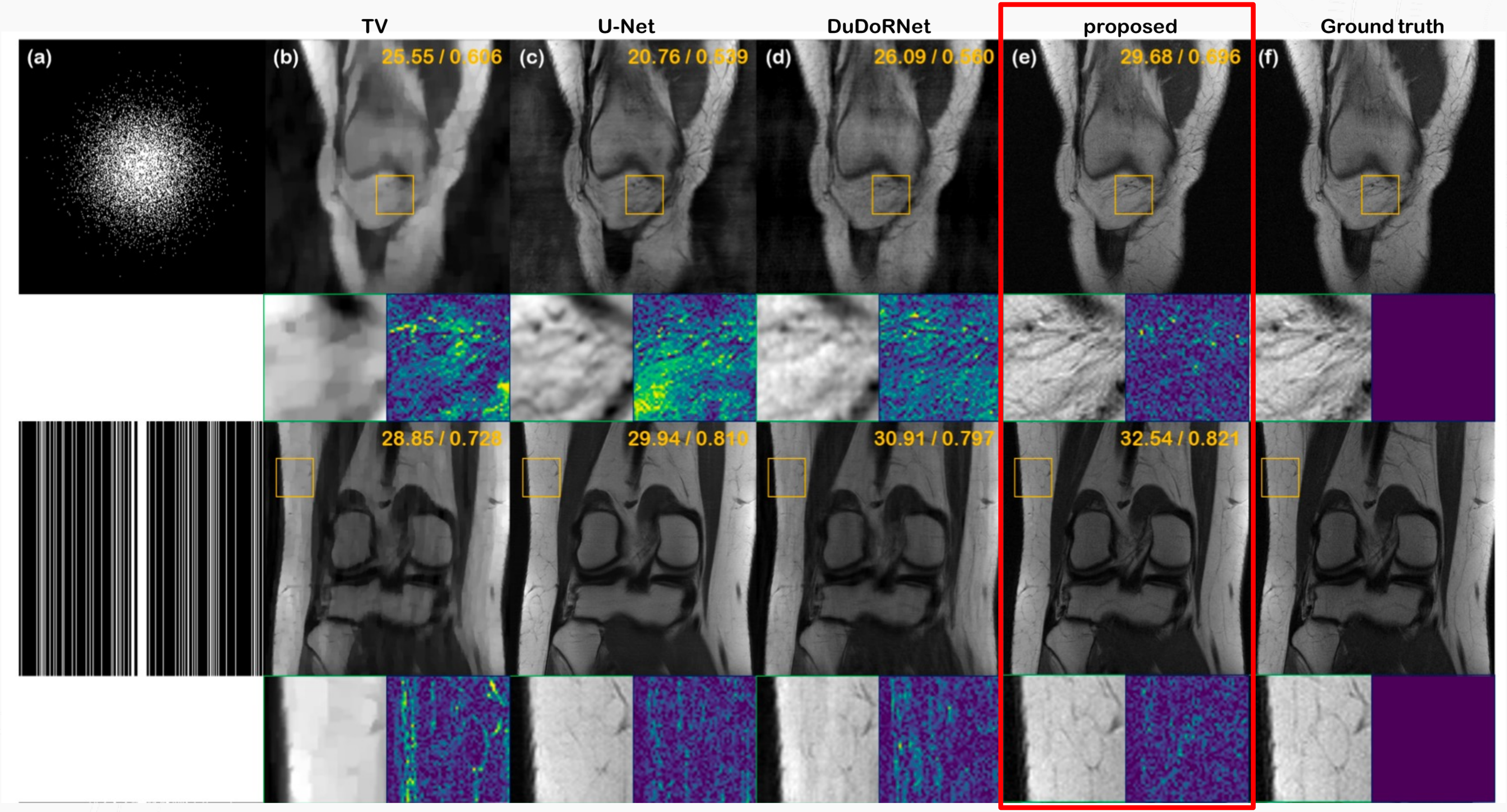
- **Denoising step** (reverse SDE)
- **Data consistency step**

Parallel Imaging

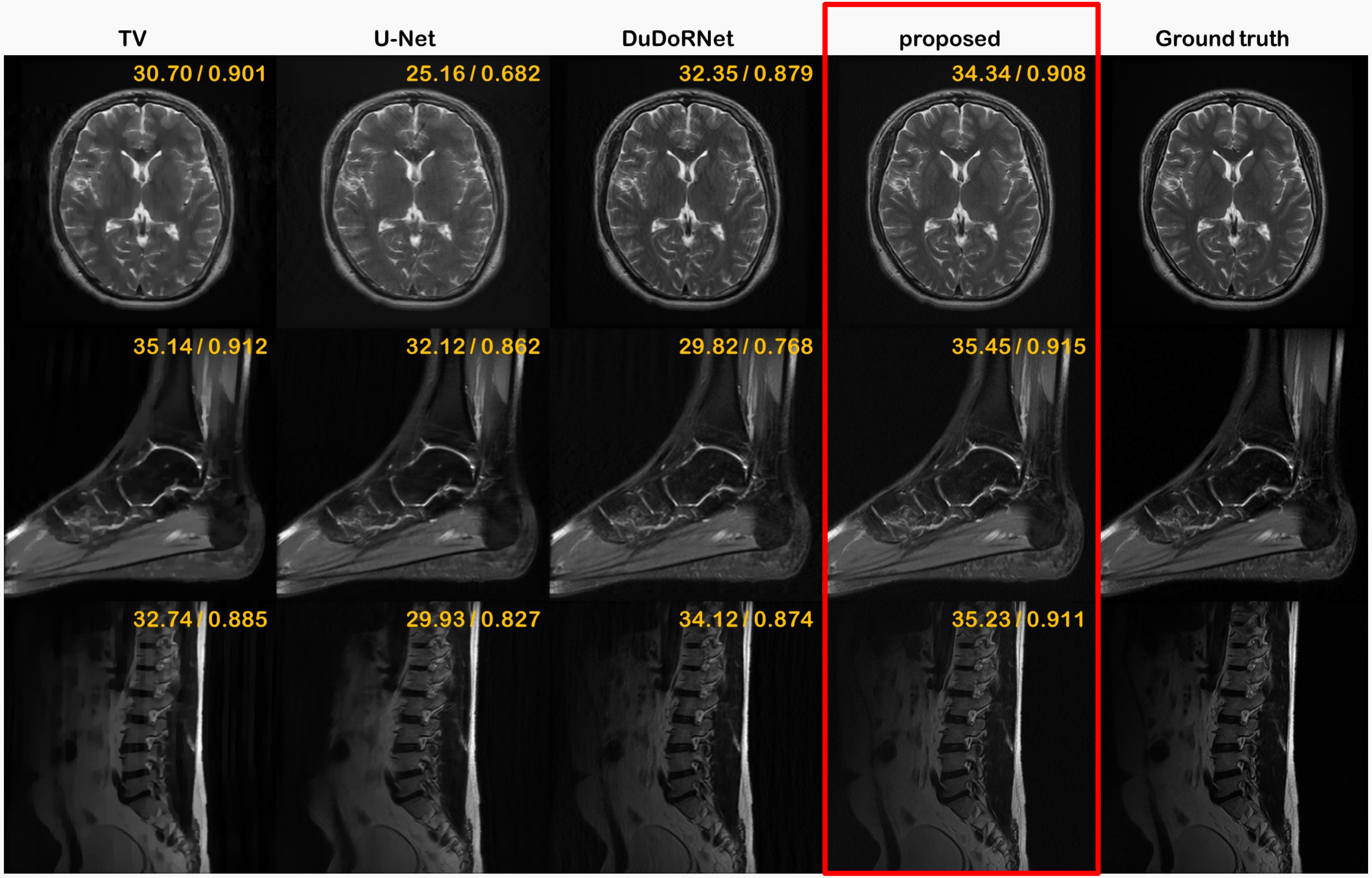


Diffusion model is trained using only DICOM files
--> no k-space data is necessary

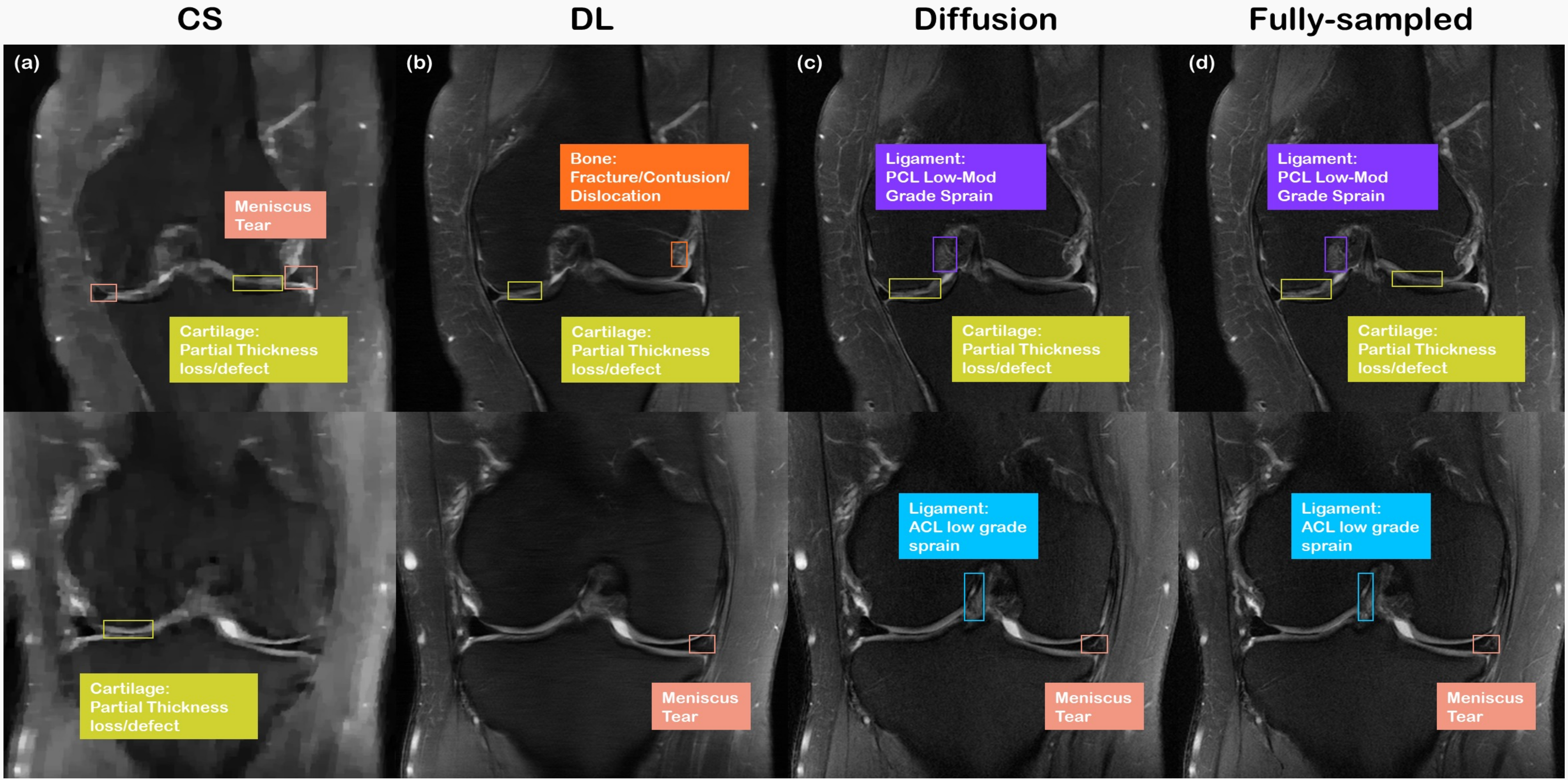
State-of-the-art Performance



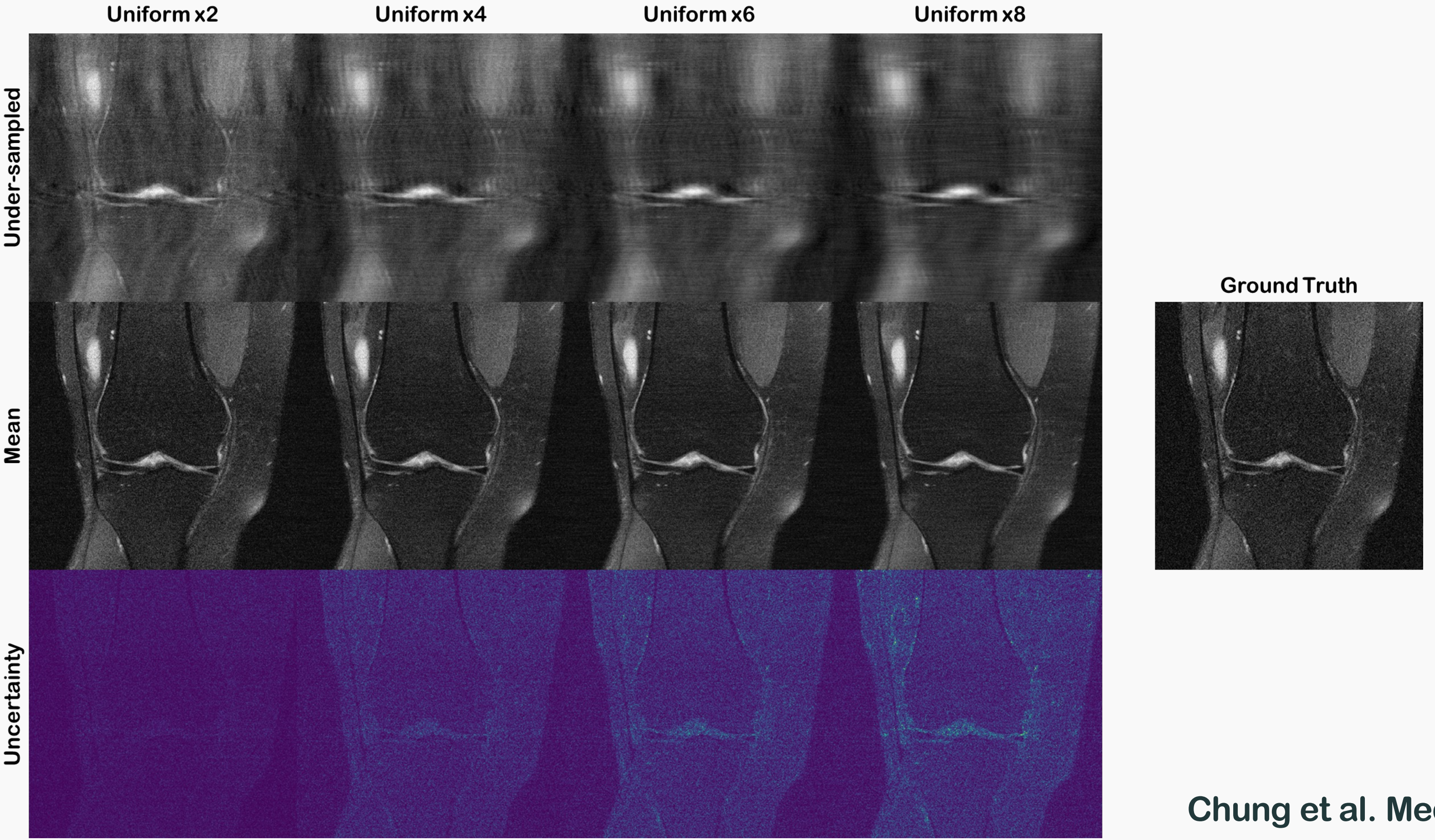
Generalization Capability



Diagnostic Capability



Uncertainty Quantification



Chung et al. MedIA, 2022

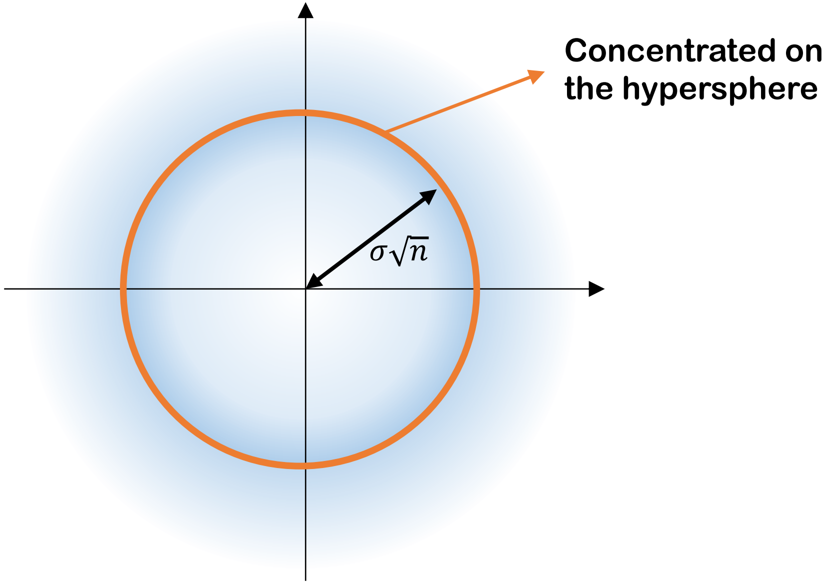
How to further improve?

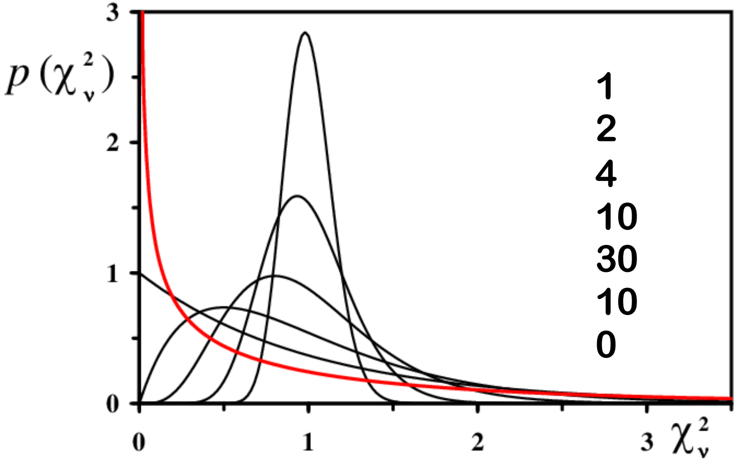
Chung et al, NeurIPS, 2022; ICLR 2023

Concentration of Measures

$RV: X = (X_1, X_2, \dots, X_n), X_i \sim \mathcal{N}(0, \sigma^2)$

$\frac{||X||^2}{n} = \frac{X_1^2 + X_2^2 + \cdots + X_n^n}{n} \rightarrow \sigma^2$ **(Law of Large number)**



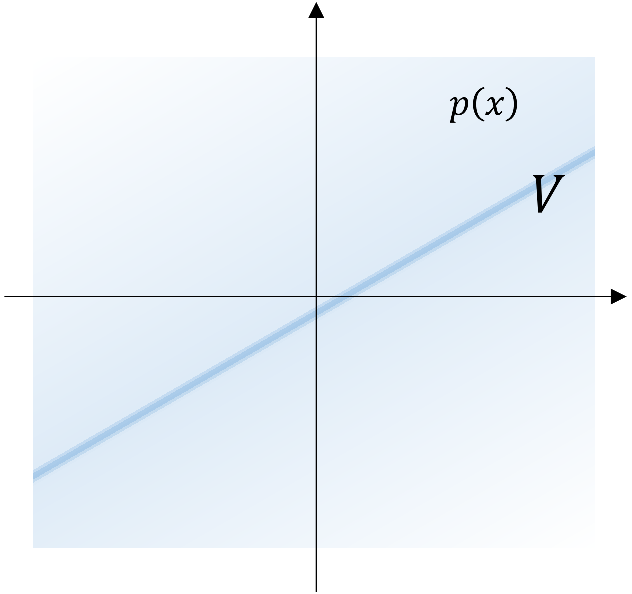


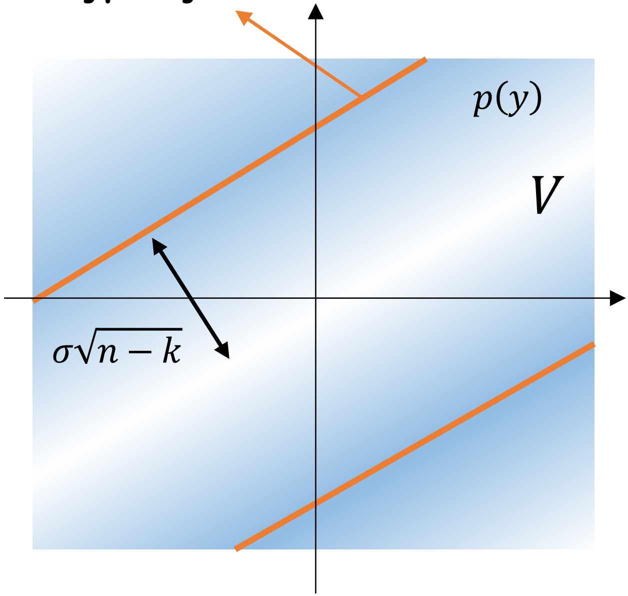
Concentration of Measures

$p(x) > 0 \leftrightarrow x \in V$, V: k **dimensional linear space**

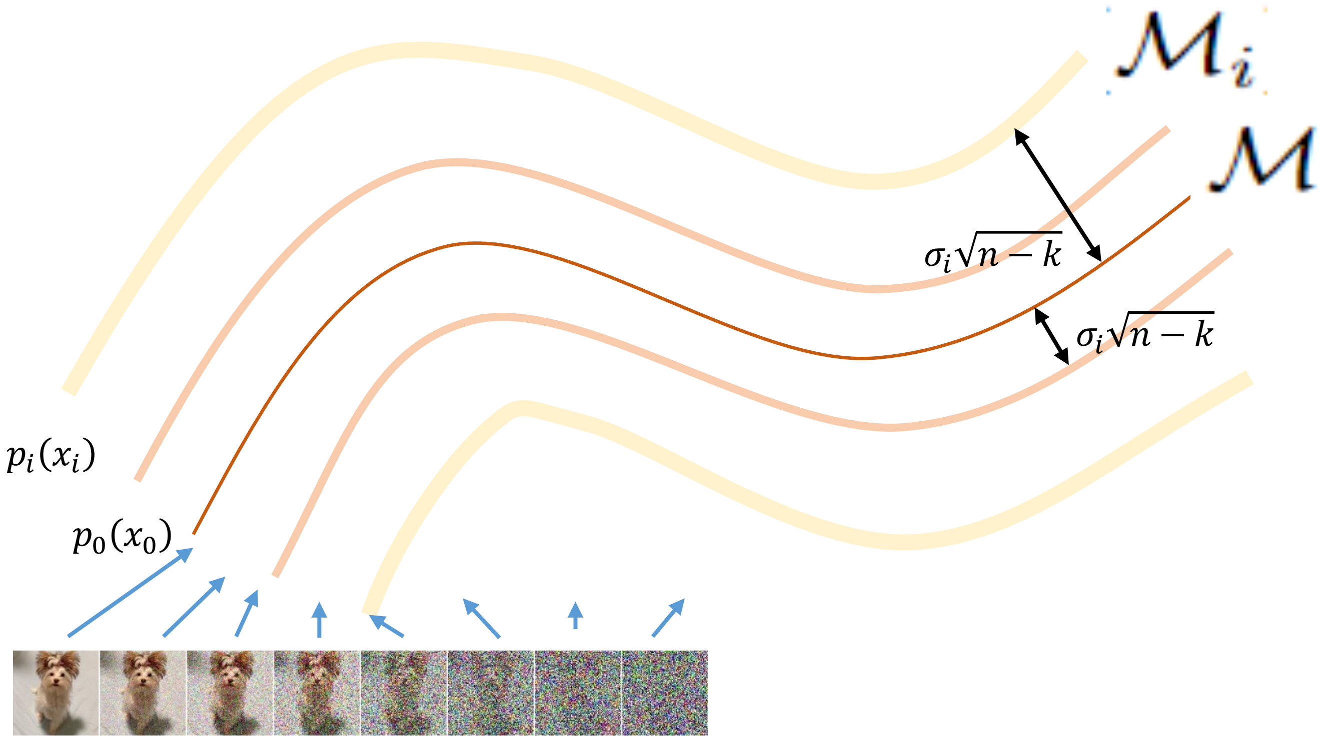
When $p(y|x) = \mathcal{N}(x, \sigma^2)$.

Marginal $p(y)$ **?**

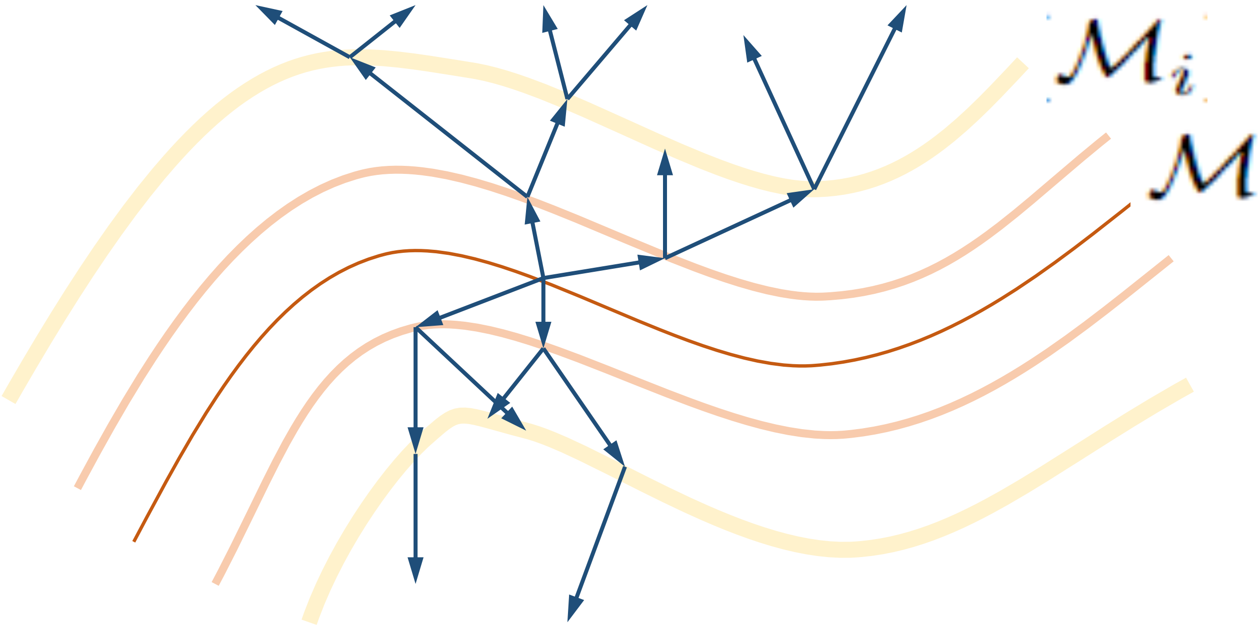




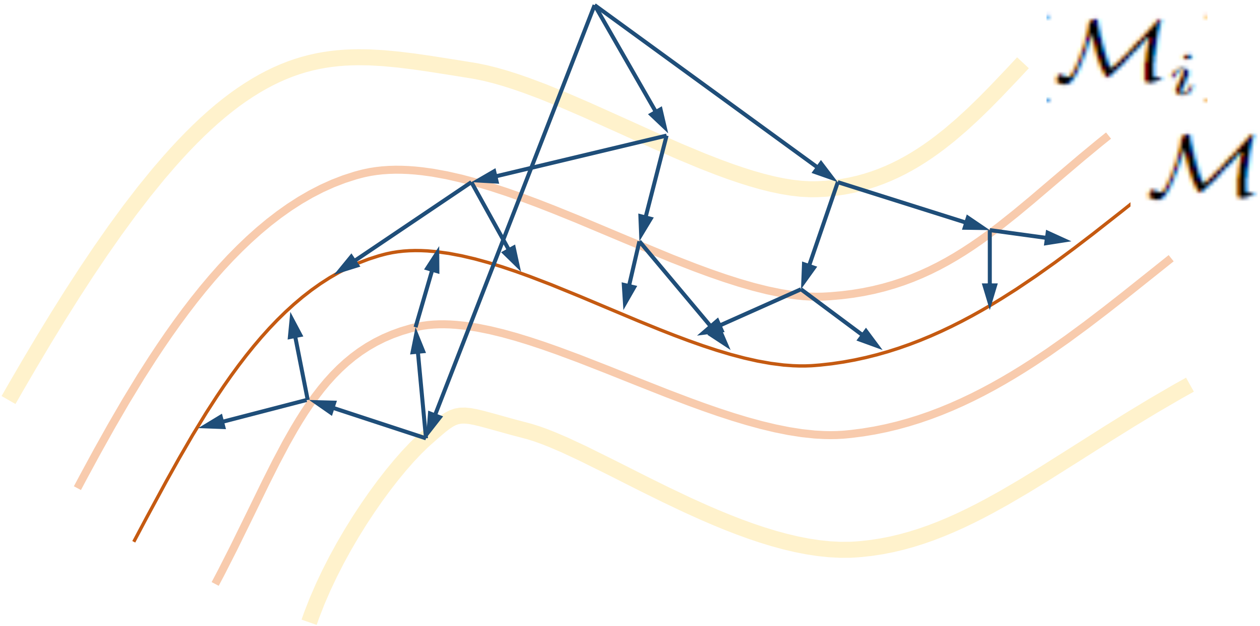
Concentration on Manifolds



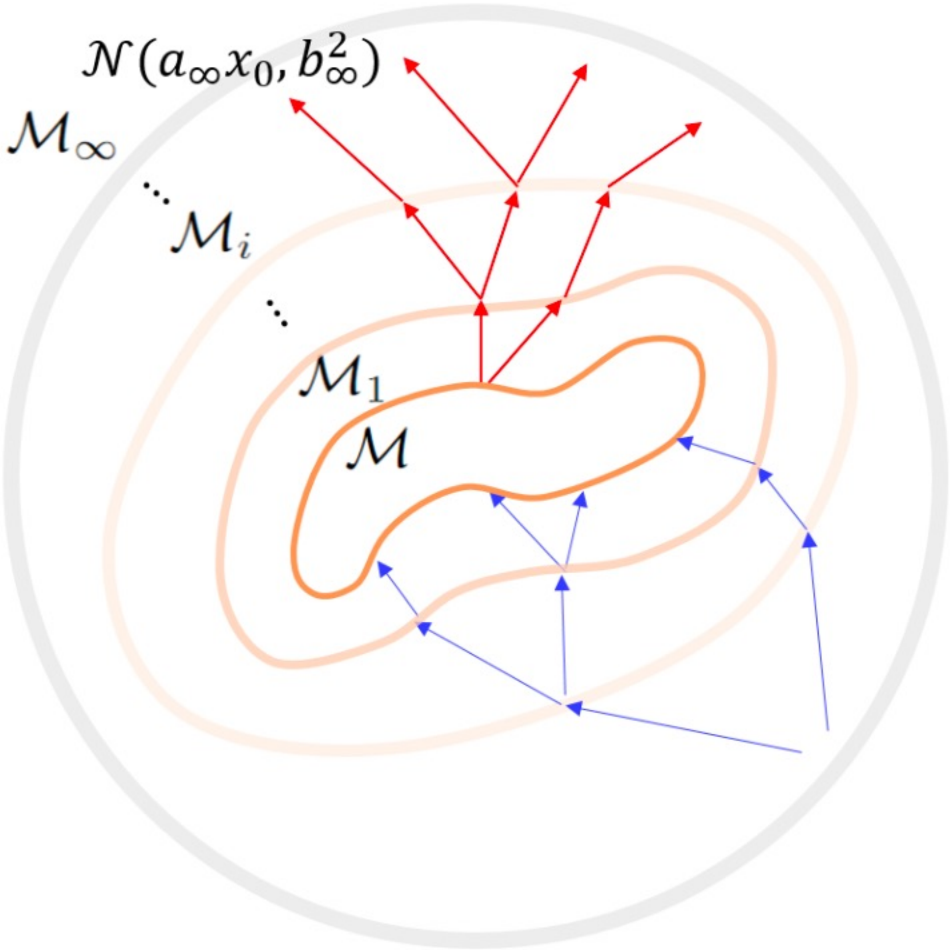
Forward Diffusion



Reverse Diffusion



Diffusion step is the transition from one manifold to another



$\mathcal{N}(a_\infty x_0, b_\infty^2)$
$\mathcal{M}_\infty$
$\mathcal{M}_i$
$\mathcal{M}_1$
$\mathcal{M}$

DPS: DIFFUSION POSTERIOR SAMPLING

Chung et al, ICLR, 2023.

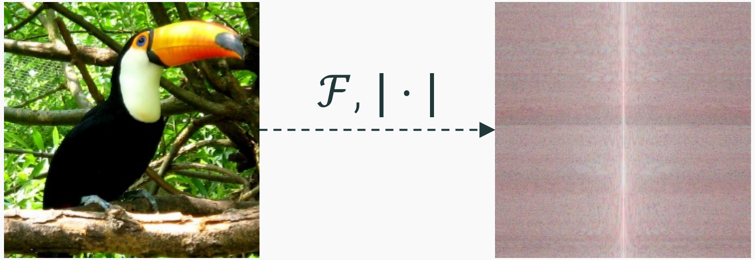
General Non-Linear Inverse Problems

Case 1. Linear

$$\mathcal{A}(\mathbf{x}) \equiv \boldsymbol{A}\mathbf{x}$$

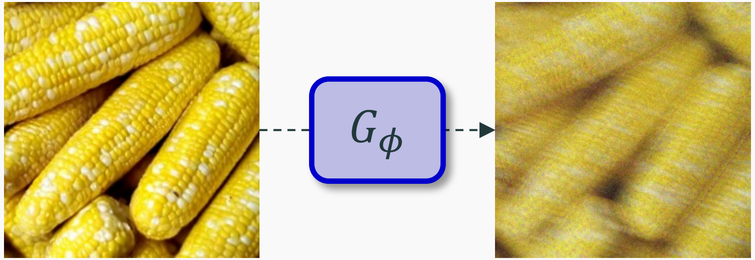
Case 2. Phase retrieval (Non-linear)

$$\mathcal{A}(\mathbf{x}) \equiv |\mathcal{F}\mathbf{x}|$$



Case 3. Neural network

$$\mathcal{A}(\mathbf{x}) \equiv G_\phi(\mathbf{x})$$



Diffusion using Posterior Sampling

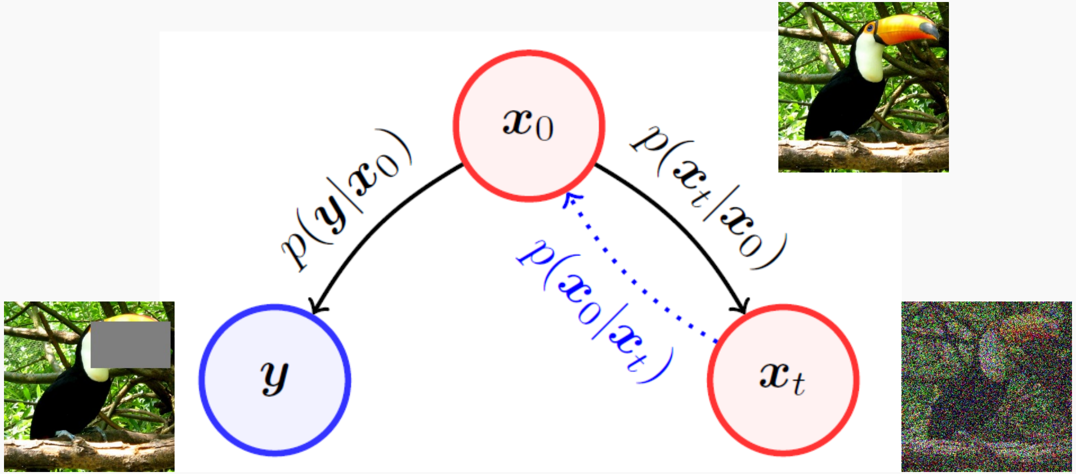
Suppose that our goal is to generate based on some condition

$$\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t|\mathbf{y}) = \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log q(\mathbf{y}|\mathbf{x}_t)$$ (Bayes' rule)

Posterior sampling using diffusion model:

$$d\boldsymbol{x} = \left[-\frac{\beta(t)}{2}\boldsymbol{x} - \beta(t)(\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t) + \nabla_{x_t} \log p_t(\boldsymbol{y}|\boldsymbol{x}_t))\right] dt + \sqrt{\beta(t)}d\bar{\boldsymbol{w}},$$

The Devil is in the Likelihood

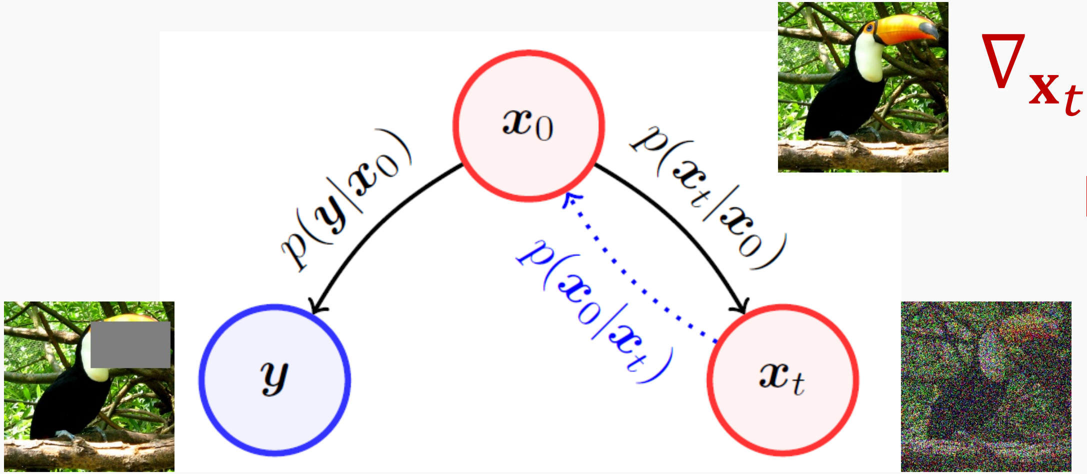


$$p(\mathbf{y}|\mathbf{x}_0) = \mathcal{N}(\mathbf{y}|\mathcal{A}\mathbf{x_0}, \sigma^2\boldsymbol{I})$$

$$\log p(\mathbf{y}|\mathbf{x}_0) = -||\boldsymbol{y} - \mathcal{A}\mathbf{x_0}||_2^2/\sigma^2$$

[ICLR 2023] Chung et al. Diffusion Posterior Sampling for General Noisy Inverse Problems

The Devil is in the Likelihood



$\nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)$

Intractable!

$$p(\mathbf{y}|\mathbf{x}_0) = \mathcal{N}(\mathbf{y}|\mathcal{A}\mathbf{x}_0, \sigma^2 \boldsymbol{I})$$

$$\log p(\mathbf{y}|\mathbf{x}_0) = -\|\boldsymbol{y} - \mathcal{A}\mathbf{x}_0\|_2^2/\sigma^2$$

[ICLR 2023] Chung et al. Diffusion Posterior Sampling for General Noisy Inverse Problems

Tweedie's Formula for Approximation

$$p(\mathbf{y}|\mathbf{x}_\mathrm{t}) = \int p(\mathbf{y}|\mathbf{x}_0, \mathbf{x}_\mathrm{t}) p(\mathbf{x}_0|\mathbf{x}_\mathrm{t}) d\mathbf{x}_0$$

$$= \int p(\mathbf{y}|\mathbf{x}_0) p(\mathbf{x}_0|\mathbf{x}_\mathrm{t}) d\mathbf{x}_0$$

$$= \mathbb{E}_{p(\mathbf{x_0}|\mathbf{x}_t)}[p(\mathbf{y}|\mathbf{x_0})]$$

[ICLR 2023] Chung et al. Diffusion Posterior Sampling for General Noisy Inverse Problems

Tweedie's Formula for Approximation

$$p(\mathbf{y}|\mathbf{x}_\mathrm{t}) = \int p(\mathbf{y}|\mathbf{x}_0, \mathbf{x}_\mathrm{t}) p(\mathbf{x}_0|\mathbf{x}_\mathrm{t}) d\mathbf{x}_0$$

$$= \int p(\mathbf{y}|\mathbf{x}_0) p(\mathbf{x}_0|\mathbf{x}_\mathrm{t}) d\mathbf{x}_0$$

$$= \mathbb{E}_{p(\mathbf{x_0}|\mathbf{x}_t)}[p(\mathbf{y}|\mathbf{x_0})]$$

$$\simeq p(\mathbf{y}|\mathbb{E}[\mathbf{x}_0|\mathbf{x}_t])$$

Push expectation inside

[ICLR 2023] Chung et al. Diffusion Posterior Sampling for General Noisy Inverse Problems

Tweedie's Formula for Approximation

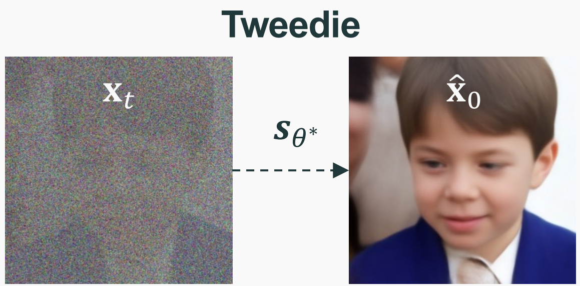
$$p(\mathbf{y}|\mathbf{x}_t) = \int p(\mathbf{y}|\mathbf{x}_0, \mathbf{x}_t) p(\mathbf{x}_0|\mathbf{x}_t) d\mathbf{x}_0$$

$$= \int p(\mathbf{y}|\mathbf{x}_0) p(\mathbf{x}_0|\mathbf{x}_t) d\mathbf{x}_0$$

$$= \mathbb{E}_{p(\mathbf{x}_0|\mathbf{x}_t)}[p(\mathbf{y}|\mathbf{x}_0)]$$

$$\simeq p(\mathbf{y}|\mathbb{E}[\mathbf{x}_0|\mathbf{x}_t])$$

$$= p(\mathbf{y}|\hat{\mathbf{x}}_0)$$



[ICLR 2023] Chung et al. Diffusion Posterior Sampling for General Noisy Inverse Problems

Jensen Bound for the Approximation

Theorem 1. *For the given measurement model (6) with* $\boldsymbol{n} \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I})$*, we have*

$$p(\boldsymbol{y}|\boldsymbol{x}_t) \simeq p(\boldsymbol{y}|\hat{\boldsymbol{x}}_0), \tag{13}$$

where the approximation error can be quantified with the Jensen gap, which is upper bounded by

$$\mathcal{J} \leq \frac{d}{\sqrt{2\pi\sigma^2}} e^{-1/2\sigma^2} \|\nabla_{\boldsymbol{x}} \mathcal{A}(\boldsymbol{x})\| m_1, \tag{14}$$

where $\|\nabla_{\boldsymbol{x}} \mathcal{A}(\boldsymbol{x})\| := \max_{\boldsymbol{x}} \|\nabla_{\boldsymbol{x}} \mathcal{A}(\boldsymbol{x})\|$ *and* $m_1 := \int \|\boldsymbol{x}_0 - \hat{\boldsymbol{x}}_0\| p(\boldsymbol{x}_0|\boldsymbol{x}_t)\, d\boldsymbol{x}_0$.

[ICLR 2023] Chung et al. Diffusion Posterior Sampling for General Noisy Inverse Problems

Diffusion Posterior Sampling

$$\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|\mathbf{y}) = \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)$$

[ICLR 2023] Chung et al. Diffusion Posterior Sampling for General Noisy Inverse Problems

Diffusion Posterior Sampling

$$\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|\mathbf{y}) = \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)$$

$$\simeq \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\hat{\mathbf{x}}_0)$$

Theorem 1.

Diffusion Posterior Sampling

$$\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|\mathbf{y}) = \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)$$

$$\simeq \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\hat{\mathbf{x}}_0)$$

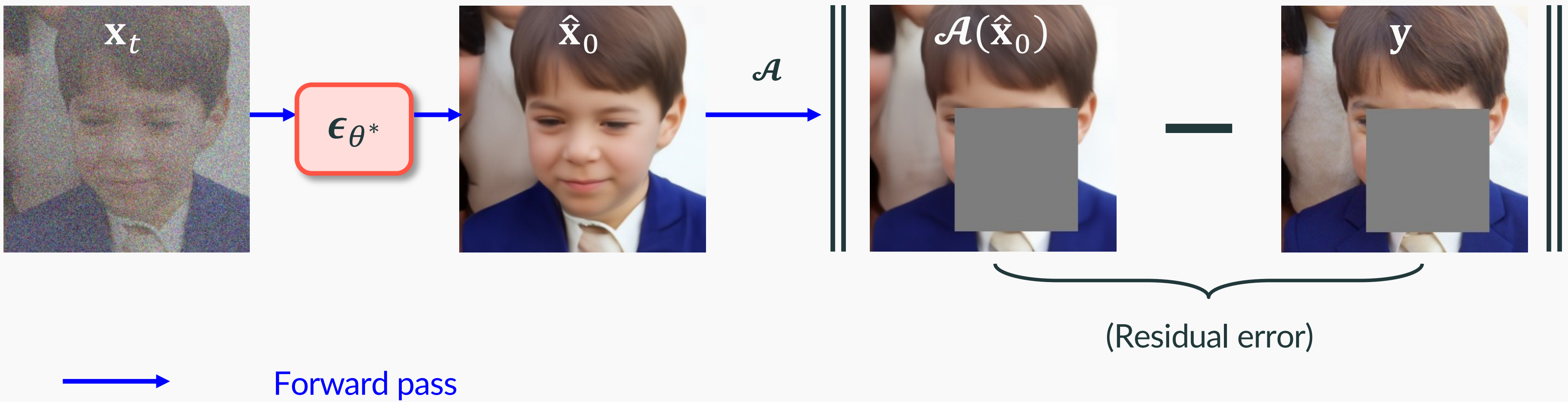
$$\simeq s_{\theta^*}(\mathbf{x}_t, t) + \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\hat{\mathbf{x}}_0)$$

1. Gaussian $\nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\hat{\mathbf{x}}_0) = -\rho \nabla_{\mathbf{x}_t} \|\mathbf{y} - \mathcal{A}(\hat{\mathbf{x}}_0)\|_2^2$

2. Poisson $\nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\hat{\mathbf{x}}_0) \simeq -\rho \nabla_{\mathbf{x}_t} \|\mathbf{y} - \mathcal{A}(\hat{\mathbf{x}}_0)\|_\Lambda^2$

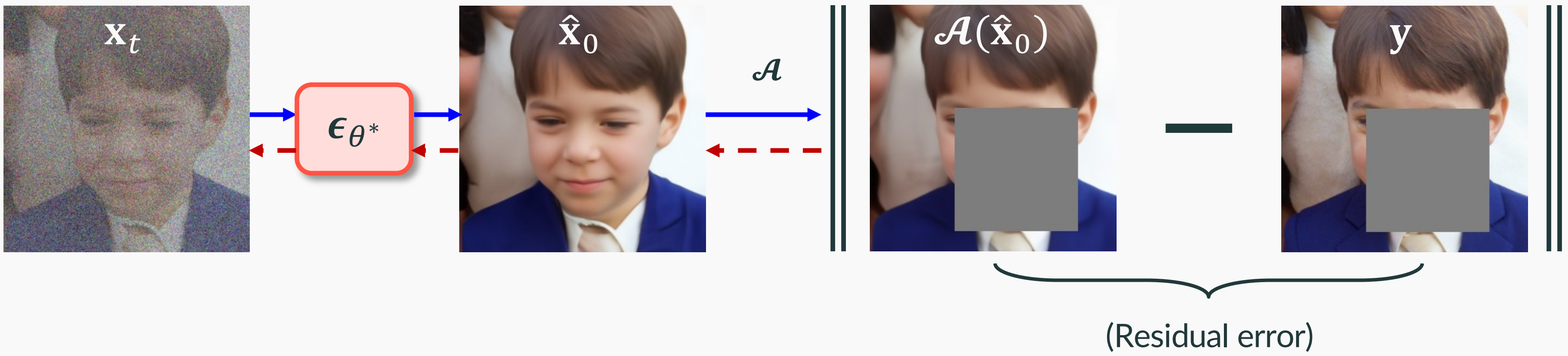
Diffusion Posterior Sampling

$$\nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\hat{\mathbf{x}}_0) = -\rho \nabla_{\mathbf{x}_t} \|\mathbf{y} - \mathcal{A}(\hat{\mathbf{x}}_0)\|_2^2$$



Diffusion Posterior Sampling

$$\nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\hat{\mathbf{x}}_0) = -\rho \nabla_{\mathbf{x}_t} \|\mathbf{y} - \mathcal{A}(\hat{\mathbf{x}}_0)\|_2^2$$

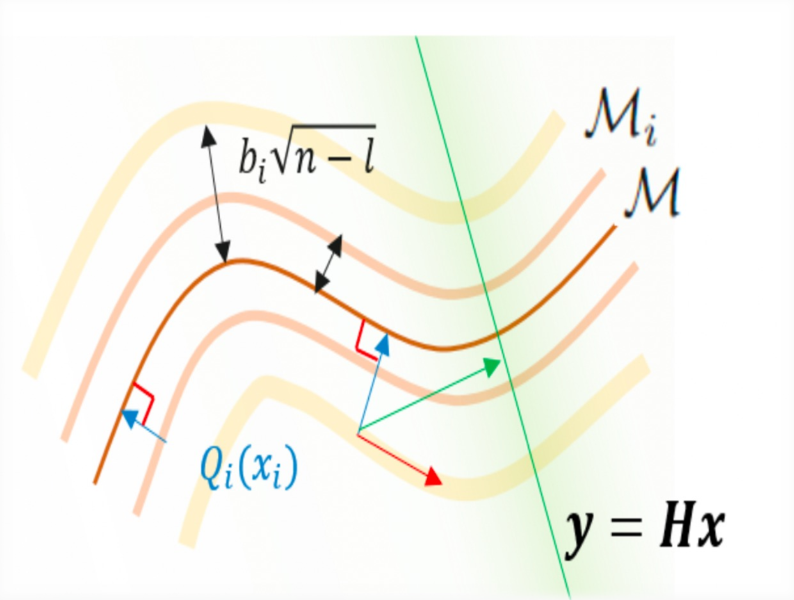


Forward pass

Backward propagation

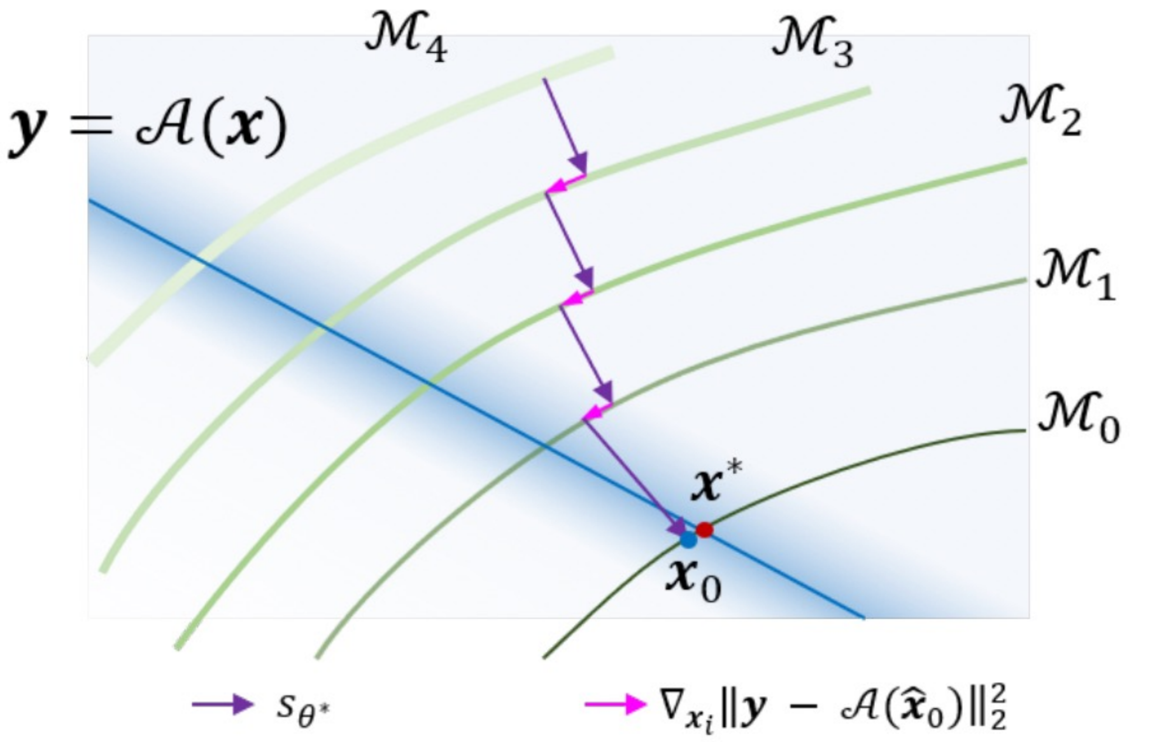
Manifold Constrained Gradient (MCG)

Chung et al, NeurIPS, 2022; Arxiv 2023



$$\frac{\partial}{\partial \boldsymbol{x}_i}\|\boldsymbol{W}(\boldsymbol{y}-\boldsymbol{H}\hat{\boldsymbol{x}}_0)\|_2^2 = -2\boldsymbol{J}_{Q_i}^T\boldsymbol{H}^T\boldsymbol{W}^T\boldsymbol{W}(\boldsymbol{y}-\boldsymbol{H}\hat{\boldsymbol{x}}_0) \in T_{\hat{\boldsymbol{x}}_0}\mathcal{M}$$

Geometry of DPS



Some Examples

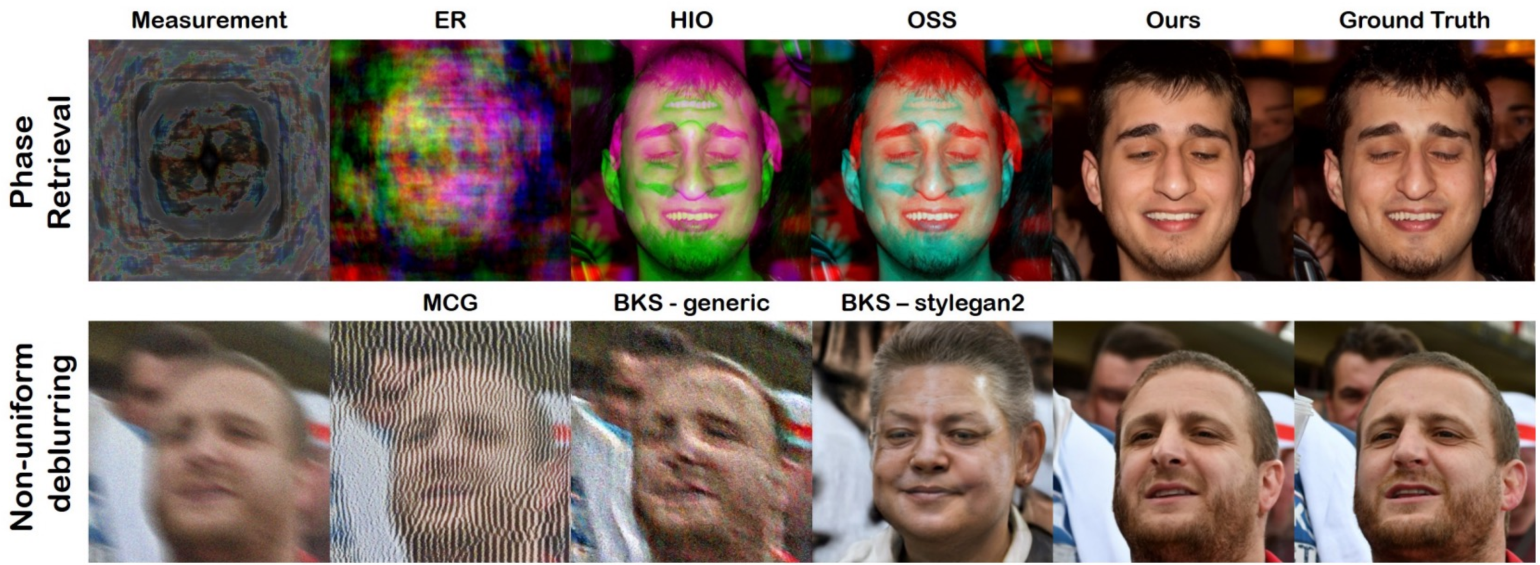
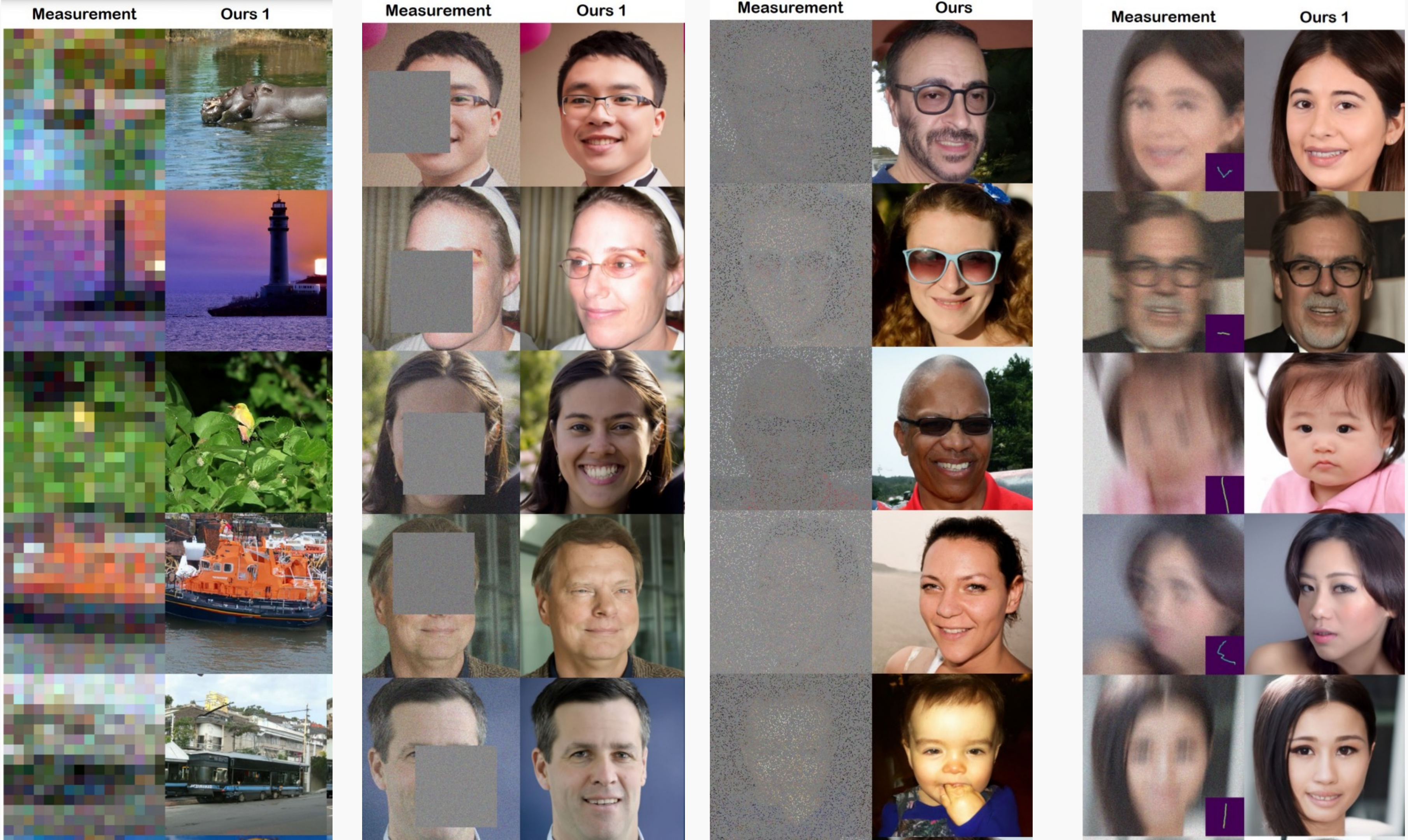
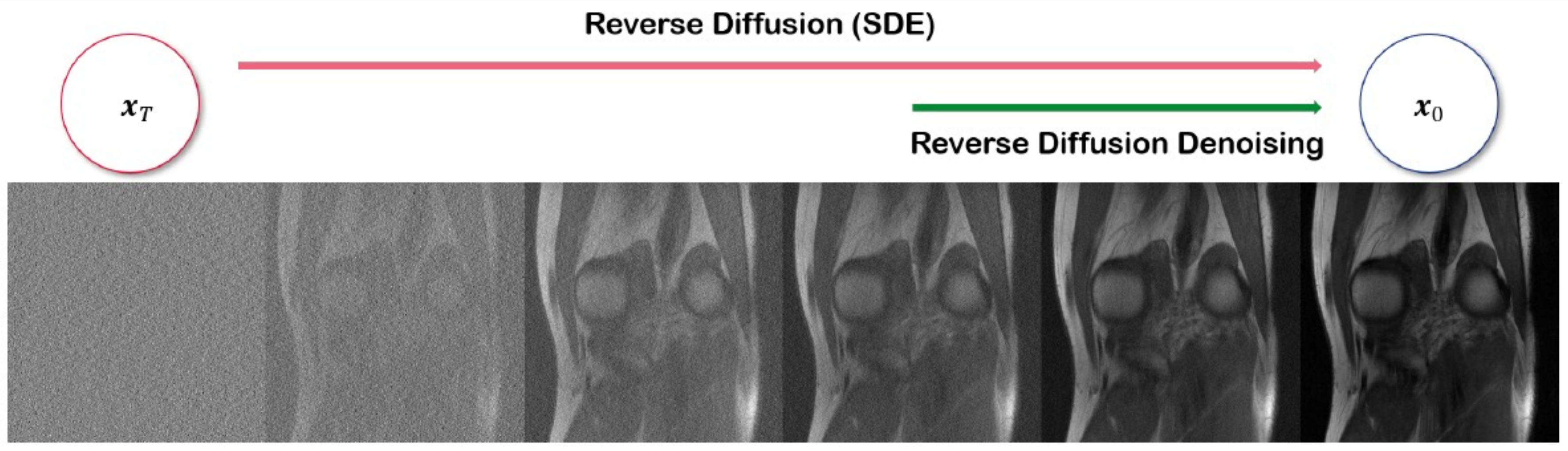


Figure 6: Results on solving nonlinear inverse problems with Gaussian noise ($\sigma = 0.05$).


Measurement
Ours 1
Measurement
Ours 1
Measurement
Ours
Measurement
Ours 1

Reverse Diffusion Denoising + Super-resolution

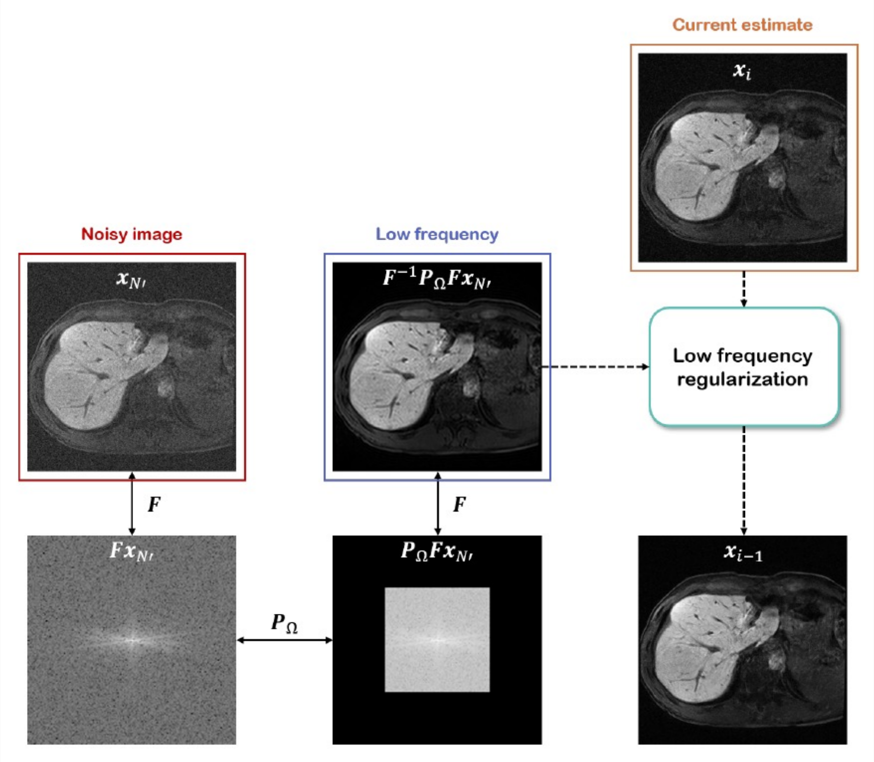


Hijack the reverse diffusion!

[TMI 2022] Chung et al. MR Image Denoising and Super-Resolution Using Regularized Reverse Diffusion

Reverse Diffusion Denoising + Super-resolution

1. Low-frequency regularization

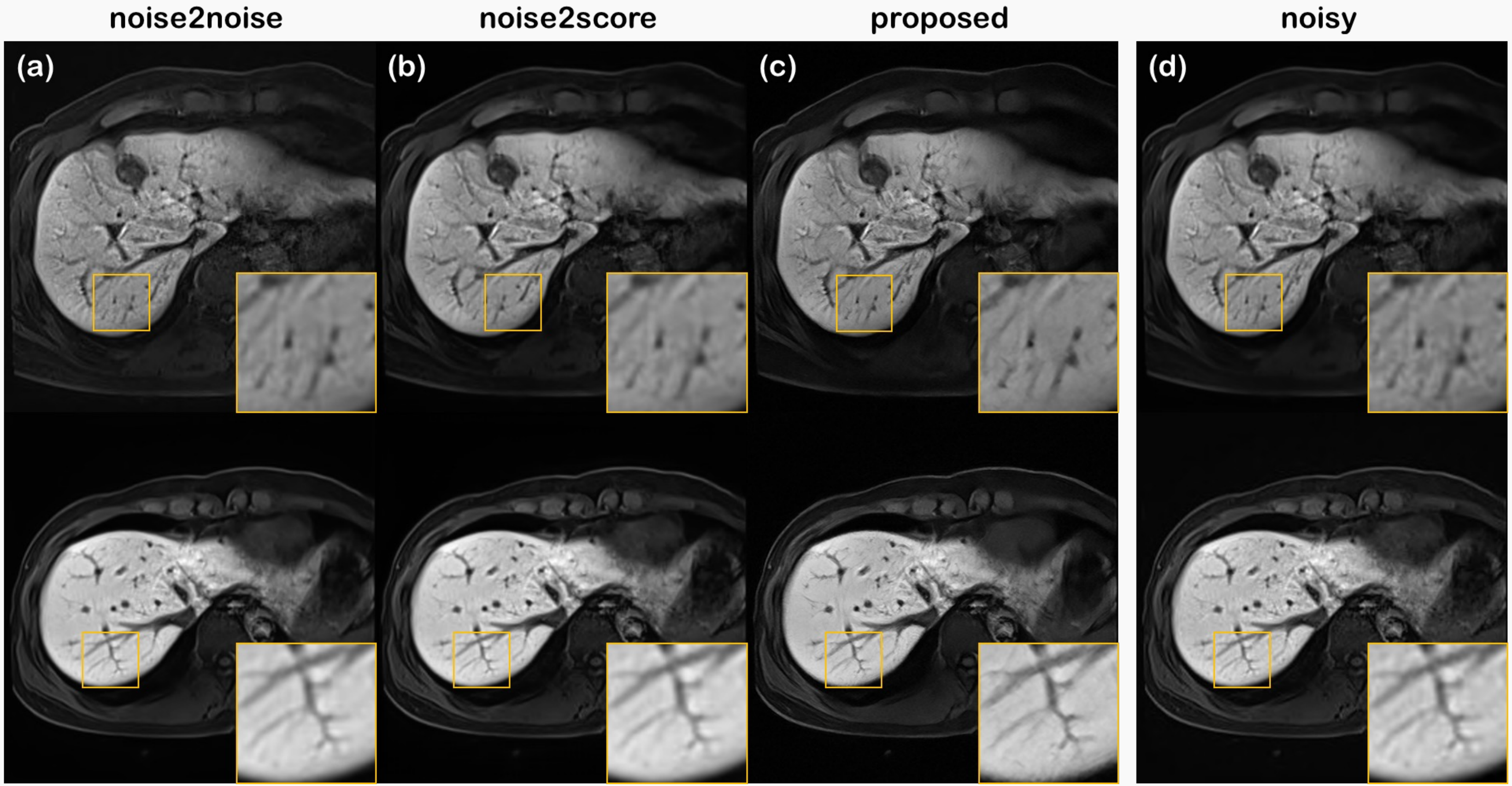


2. Reconstruction Guidance (MCG)

$$\hat{x}_0 = x_{i+1} + \sigma_{i+1}^2 s_\theta(x_{i+1}, \sigma_{i+1})$$
$$x_i'' = x_i' - \gamma \underbrace{\nabla_{x_{i+1}} \|x_i' - \hat{x}_0\|_2^2}_{\text{MCG}},$$

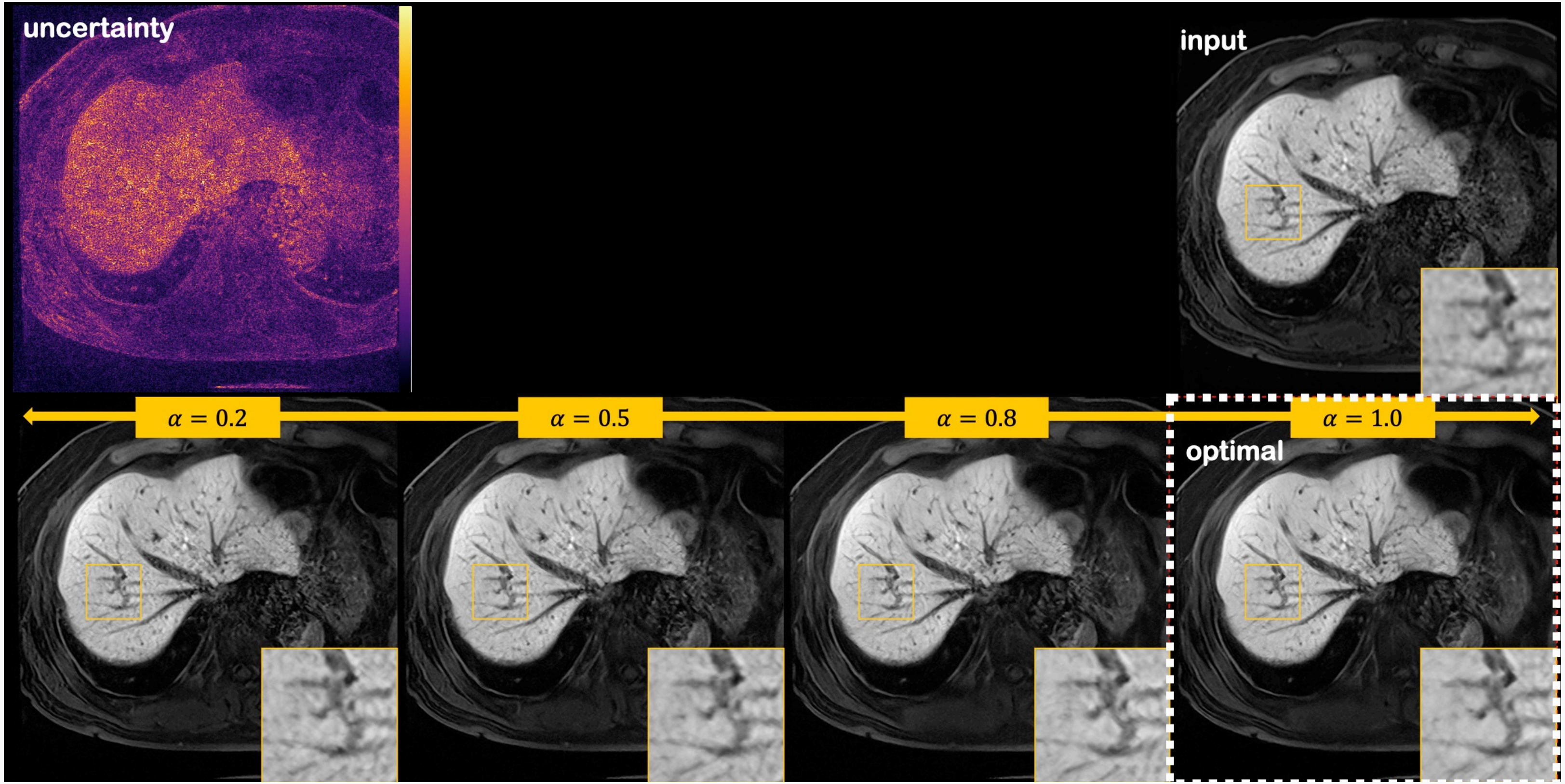
[TMI 2022] Chung et al. MR Image Denoising and Super-Resolution Using Regularized Reverse Diffusion

Reverse Diffusion Denoising + Super-resolution



Chung et al. IEEE TMI, 2022

Uncertainty Quantification, User Controllability

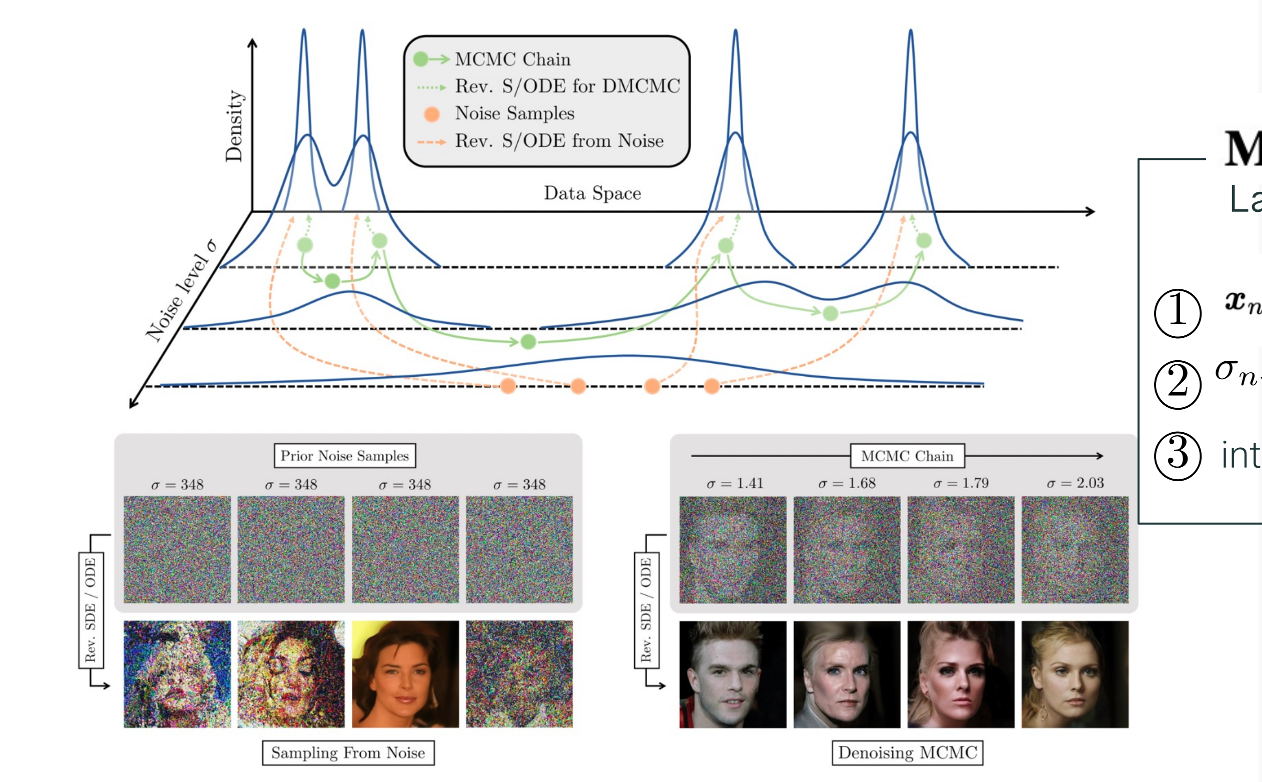


[TMI 2022] Chung et al. MR Image Denoising and Super-Resolution Using Regularized Reverse Diffusion

How to **accelerate?**

Denoising Markov-Chain Monte Carlo (DMCMC)

Kim et al, ICML 2023



MCMC on $\mathcal{X} \times \mathcal{S}$

Langevin Gibbs Sampling

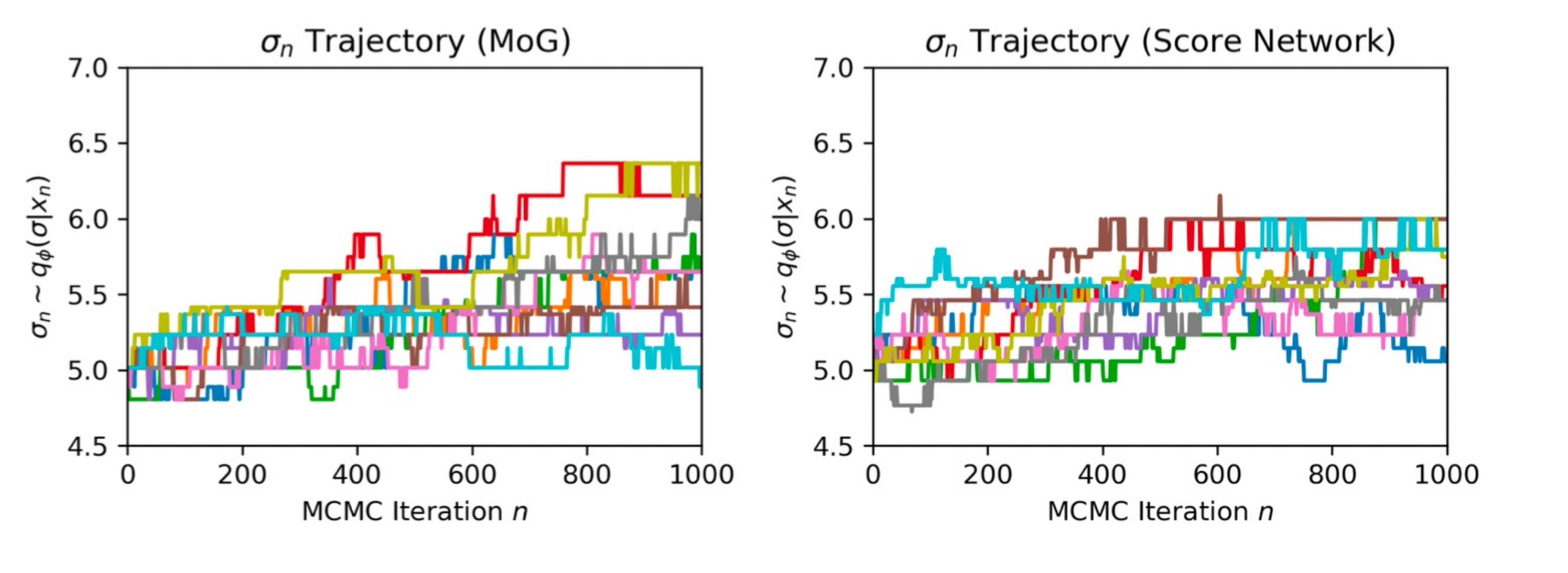
① $\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + (\eta/2) \cdot s_\theta(\boldsymbol{x}_n, \sigma_n) + \sqrt{\eta} \cdot \boldsymbol{\epsilon}$

② $\sigma_{n+1} \sim \hat{p}(\sigma | \boldsymbol{x}_{n+1})$

③ integrating the reverse-S/ODE

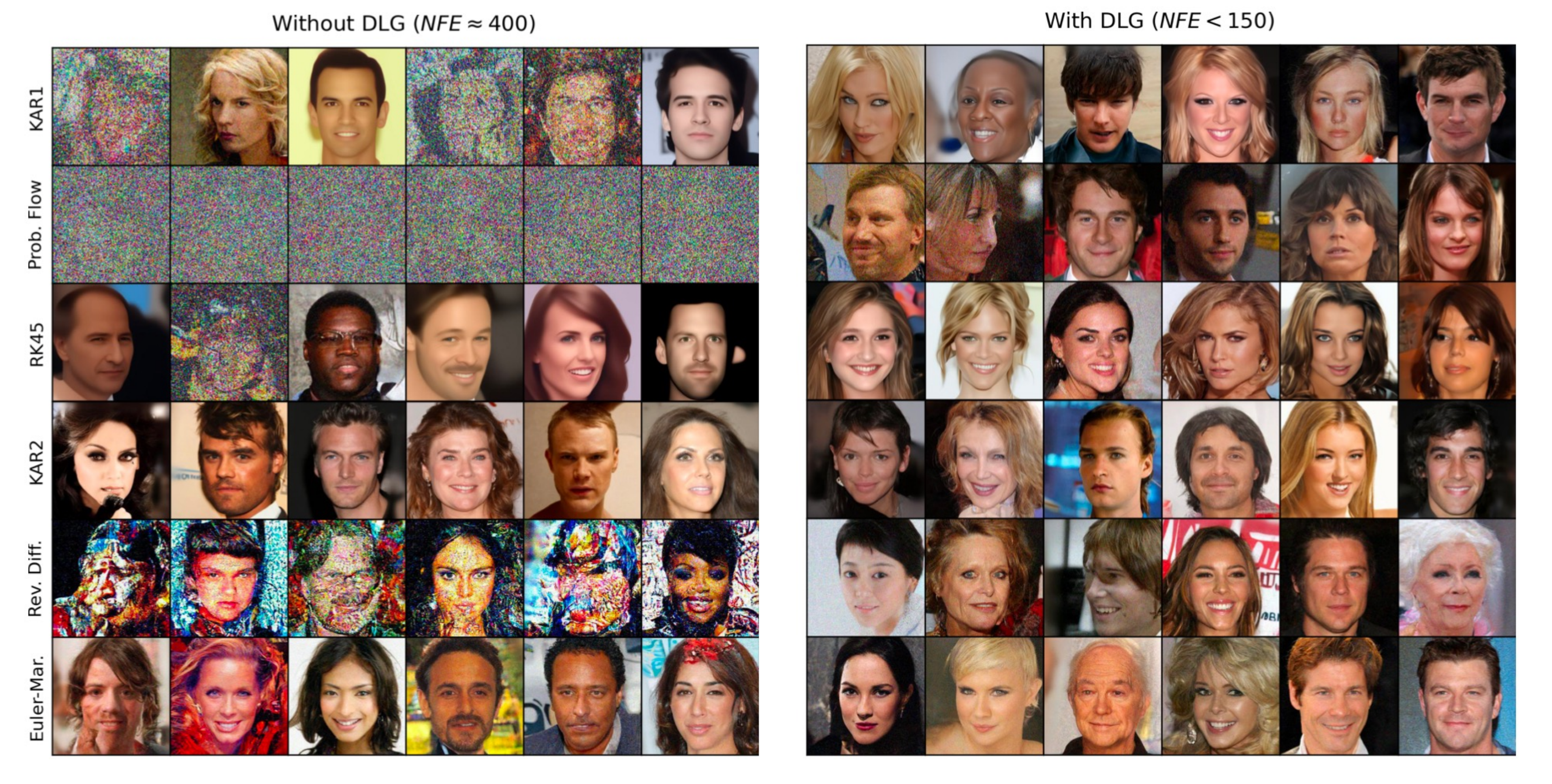
Denoising Markov-Chain Monte Carlo (DMCMC)

Kim et al, ICML 2023



Denoising Markov-Chain Monte Carlo (DMCMC)

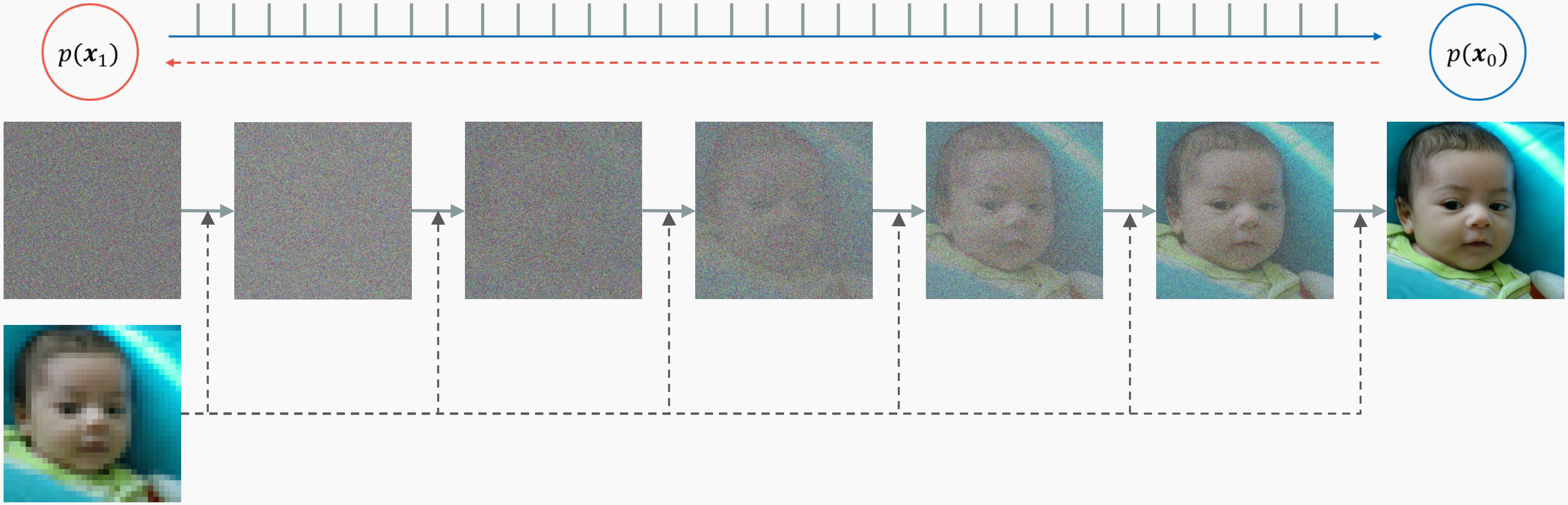
Kim et al, ICML 2023



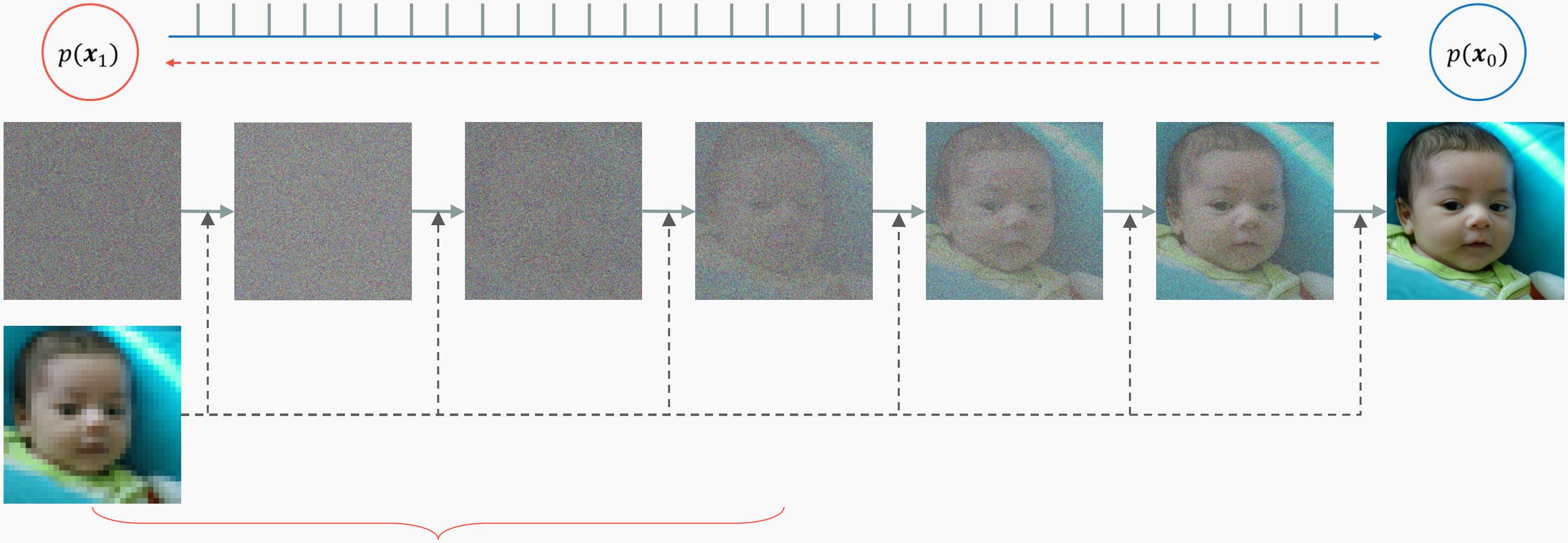
ACCELERATION OF CONDITIONAL DIFFUSION

Chung et al, CVPR, 2022

Intuition: Why use the whole process?



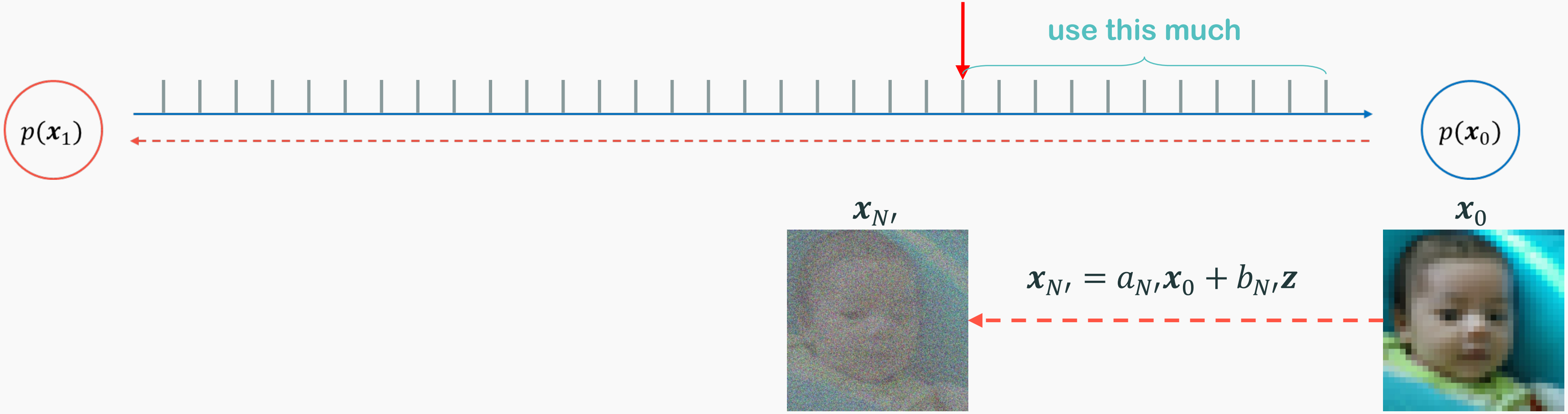
Intuition: Why use the whole process?



Is this part necessary?

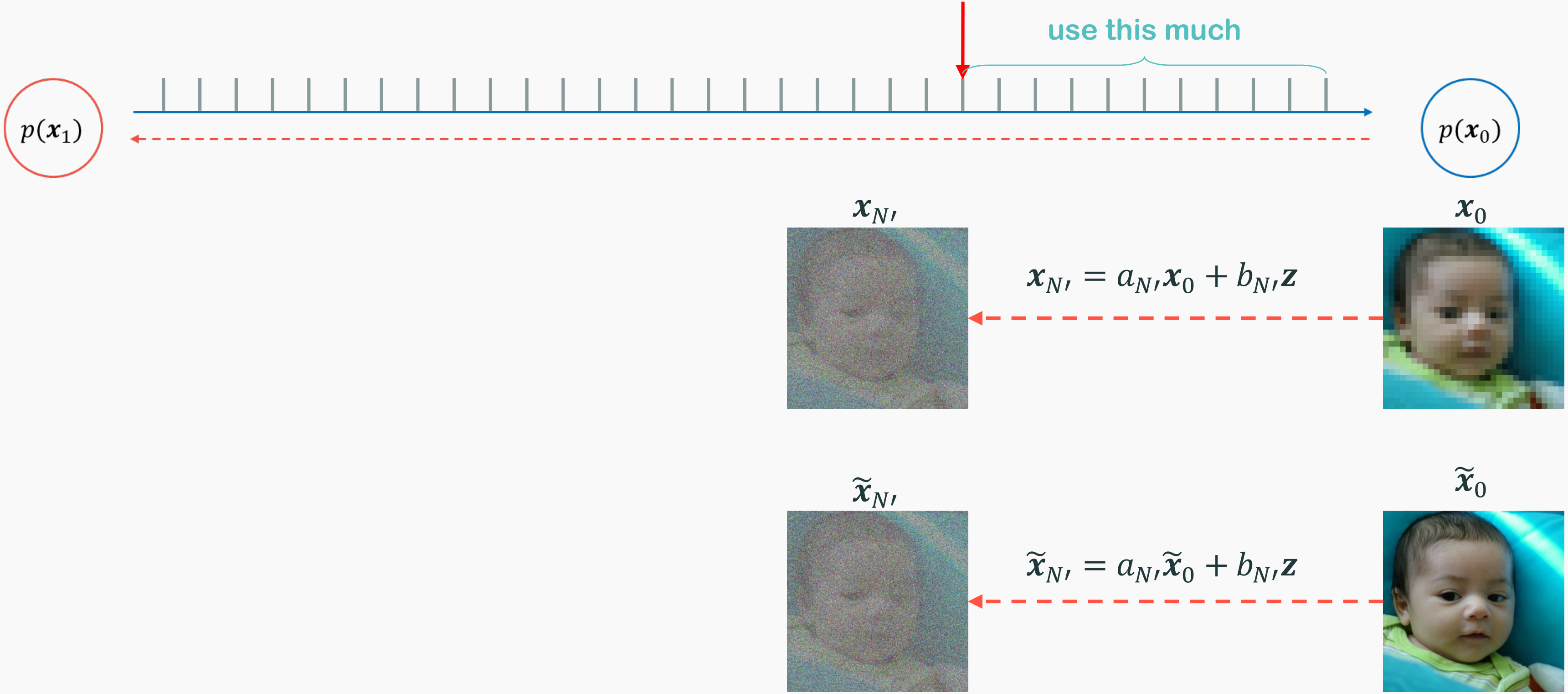
Intuition: Why use the whole process?

- $t_0 = 0.3$
- $N' = t_0 N = 300$

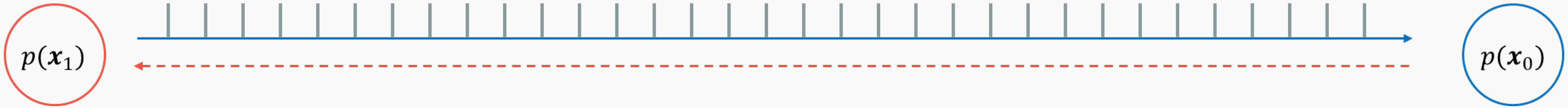


Intuition: Why use the whole process?

- $t_0 = 0.3$
- $N' = t_0 N = 300$



CCDF: The Algorithm



Algorithm 1 Accelerated Super-resolution / inpainting (VP, Markov)

Require: $\boldsymbol{x}_0, \hat{\boldsymbol{x}}_0, N', \{\alpha_i\}_{i=1}^{N'}, \{\sigma_i\}_{i=1}^{N'}, \boldsymbol{s}_\theta$

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

$\boldsymbol{x}_{N'} \leftarrow \sqrt{\bar{\alpha}_{N'}}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_{N'}}\boldsymbol{z}$ ▷ Forward diffusion

for $i = N'$ to 1 **do** ▷ Reverse diffusion

$\quad \boldsymbol{x}'_{i-1} \leftarrow \frac{1}{\sqrt{\alpha_i}}(\boldsymbol{x}_i + (1-\alpha_i)\boldsymbol{s}_\theta(\boldsymbol{x}_i, i))$

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

$\quad \boldsymbol{x}_{i-1} \leftarrow \boldsymbol{x}'_{i-1} + \sigma_i \boldsymbol{z}$ ▷ Unconditional update

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

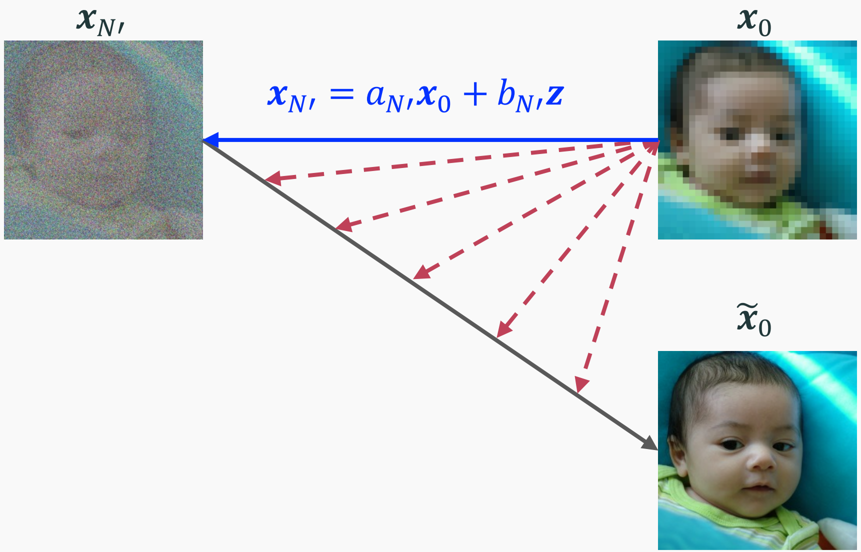
$\quad \hat{\boldsymbol{x}}_i \leftarrow \sqrt{\bar{\alpha}_i}\hat{\boldsymbol{x}}_0 + \sqrt{1-\bar{\alpha}_i}\boldsymbol{z}$

$\quad \boldsymbol{x}_{i-1} = (\boldsymbol{I} - \boldsymbol{P})\boldsymbol{x}_{i-1} + \hat{\boldsymbol{x}}_i$

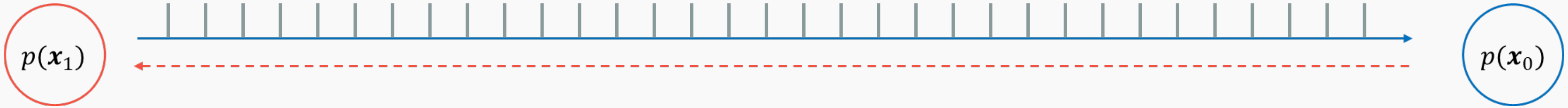
▷ Measurement consistency

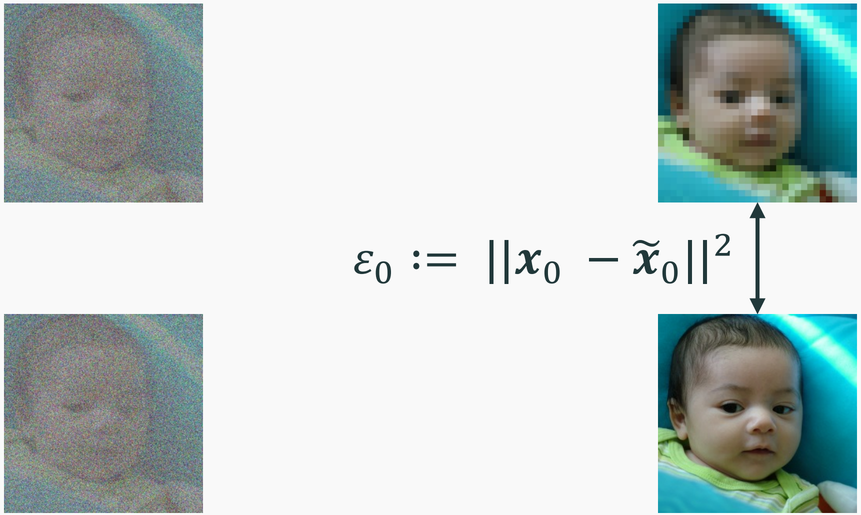
end for

return $\boldsymbol{x}_0$

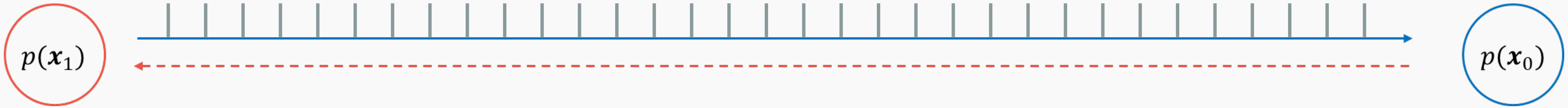


Theoretical Challenges



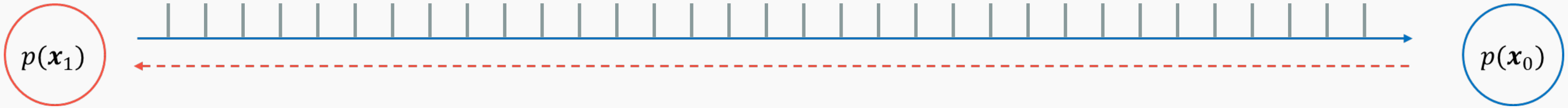


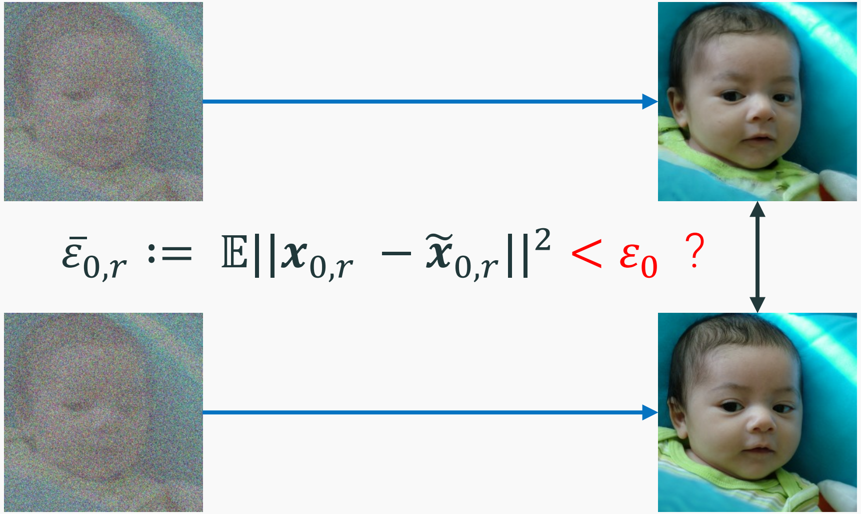
Theoretical Challenges



$$\bar{\varepsilon}_{N'} := \mathbb{E}||\boldsymbol{x}_{N'} - \widetilde{\boldsymbol{x}}_{N'}||^2 \quad > \quad \varepsilon_0 := ||\boldsymbol{x}_0 - \widetilde{\boldsymbol{x}}_0||^2$$

Theoretical Challenges





Theoretical Findings

Chung et al, CVPR, 2022

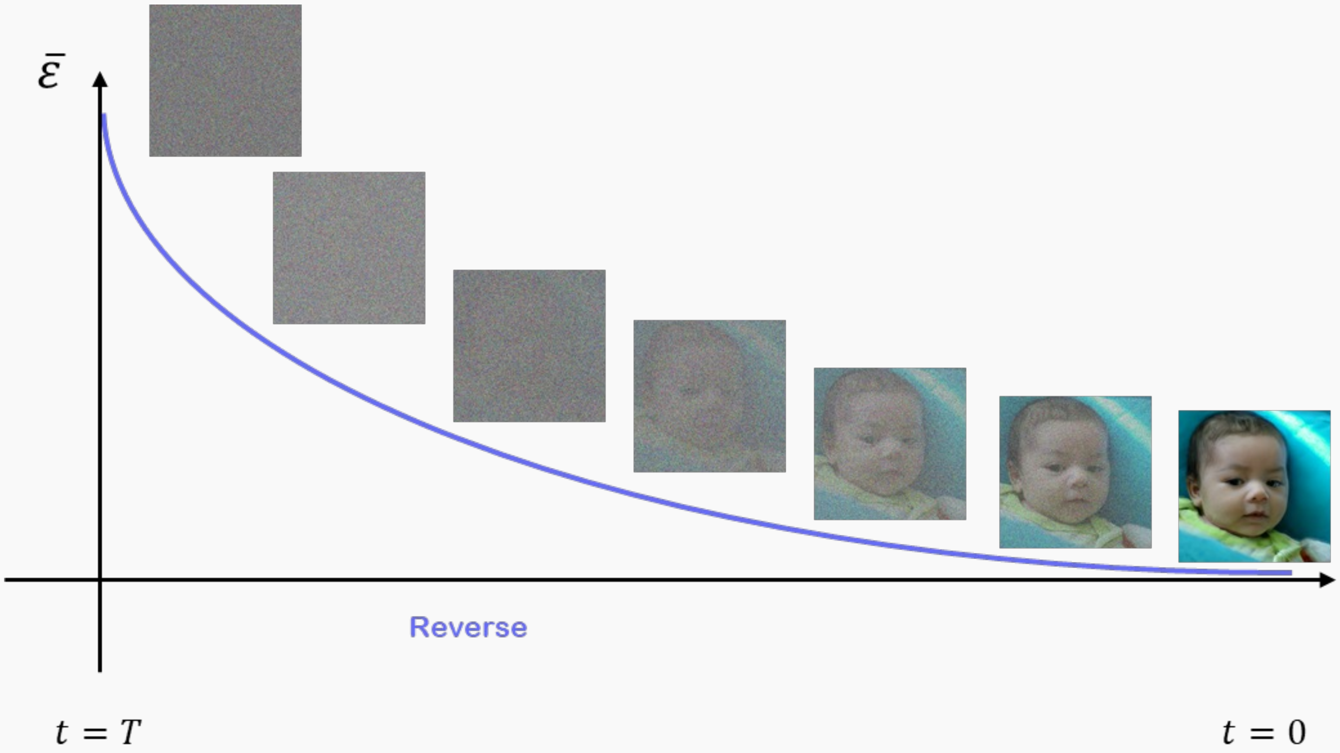
Theorem 2. (shortcut path)

- For any $0 < \mu \leq 1$, there exists a **minimum** N' s.t.

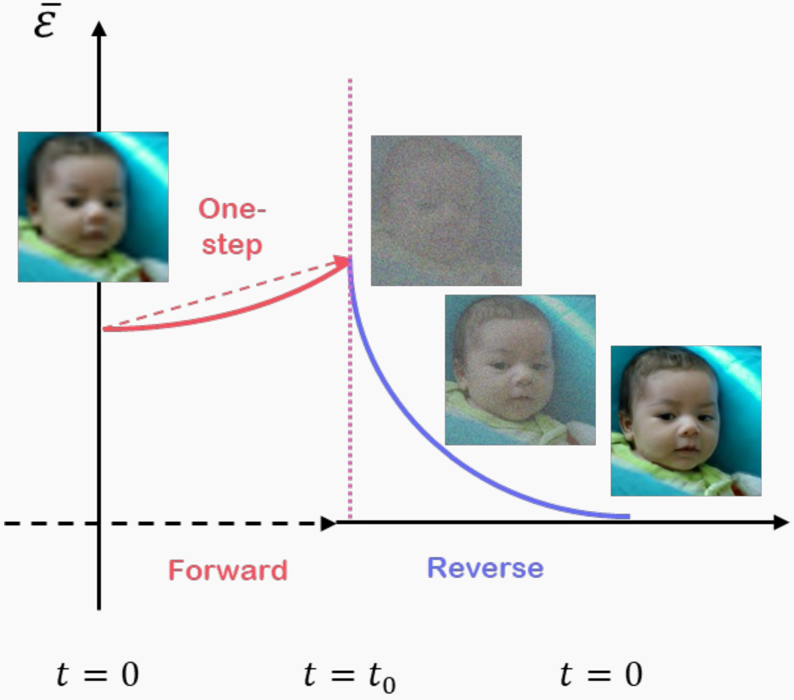
$$\bar{\varepsilon}_{0,r} \leq \mu \varepsilon_0$$

- Optimal N' **decreases** as ε_0 **gets smaller**

Come Closer Diffuse Faster (CCDF)

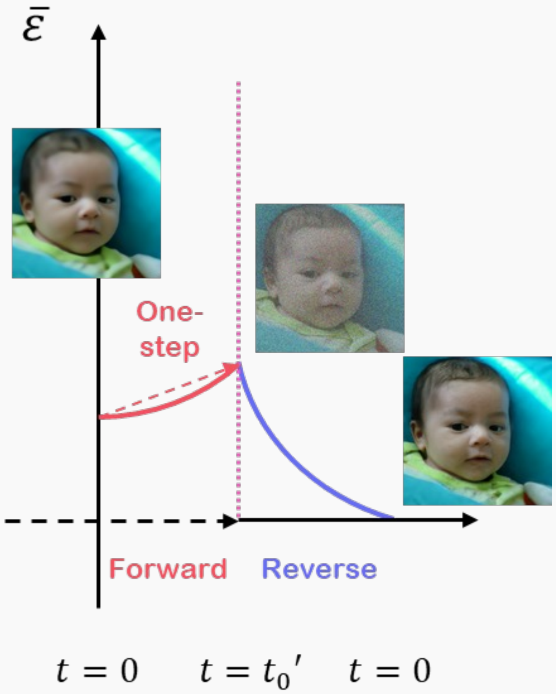


Come Closer Diffuse Faster (CCDF)



Come Closer Diffuse Faster (CCDF)

Feed-forward network correction



Reverse Diffusion is Contracting!

Stochastic Contraction

Theorem 1.

$$\bar{\varepsilon}_{0,r} \leq \frac{2C\tau}{1-\lambda^2} + \lambda^{2N'}\,\bar{\varepsilon}_{N'}$$

$$\tau = \frac{\mathrm{Tr}(\boldsymbol{A}^T\boldsymbol{A})}{n}$$

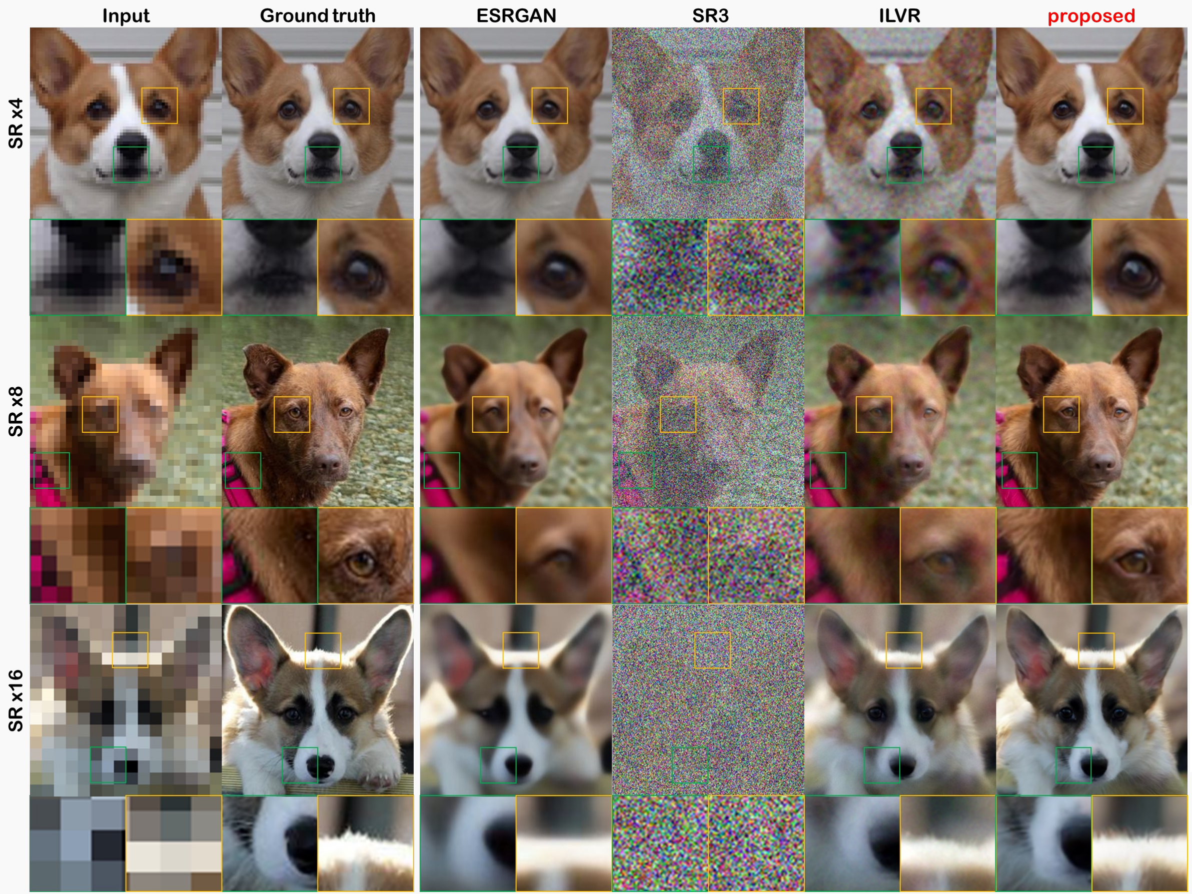
Error decreases exponentially with reverse diffusion!

Reverse diffusion is a contracting mapping with λ

$$\lambda = \begin{cases} \max_{i\in[N']} \sqrt{\alpha_i}\left(\frac{1-\bar{\alpha}_{i-1}}{1-\bar{\alpha}_i}\right) & \textit{(DDPM)} \\ \max_{i\in[N']} \frac{\sigma_{i-1}^2-\sigma_0^2}{\sigma_i^2-\sigma_0^2} & \textit{(SMLD)} \\ \max_{i\in[N']} \frac{\sigma_{i-1}}{\sigma_i} & \textit{(DDIM)} \end{cases}$$

$$C = \begin{cases} n(1-\alpha_N) & \textit{(DDPM)} \\ n\max_{i\in[N']} \sigma_i^2 - \sigma_{i-1}^2 & \textit{(SMLD)} \\ 0 & \textit{(DDIM)} \end{cases}$$

Experimental Results: SR



20 step diffusion

- **ILVR, SR3**

  $N = 20, \quad t_0 = 1.0$

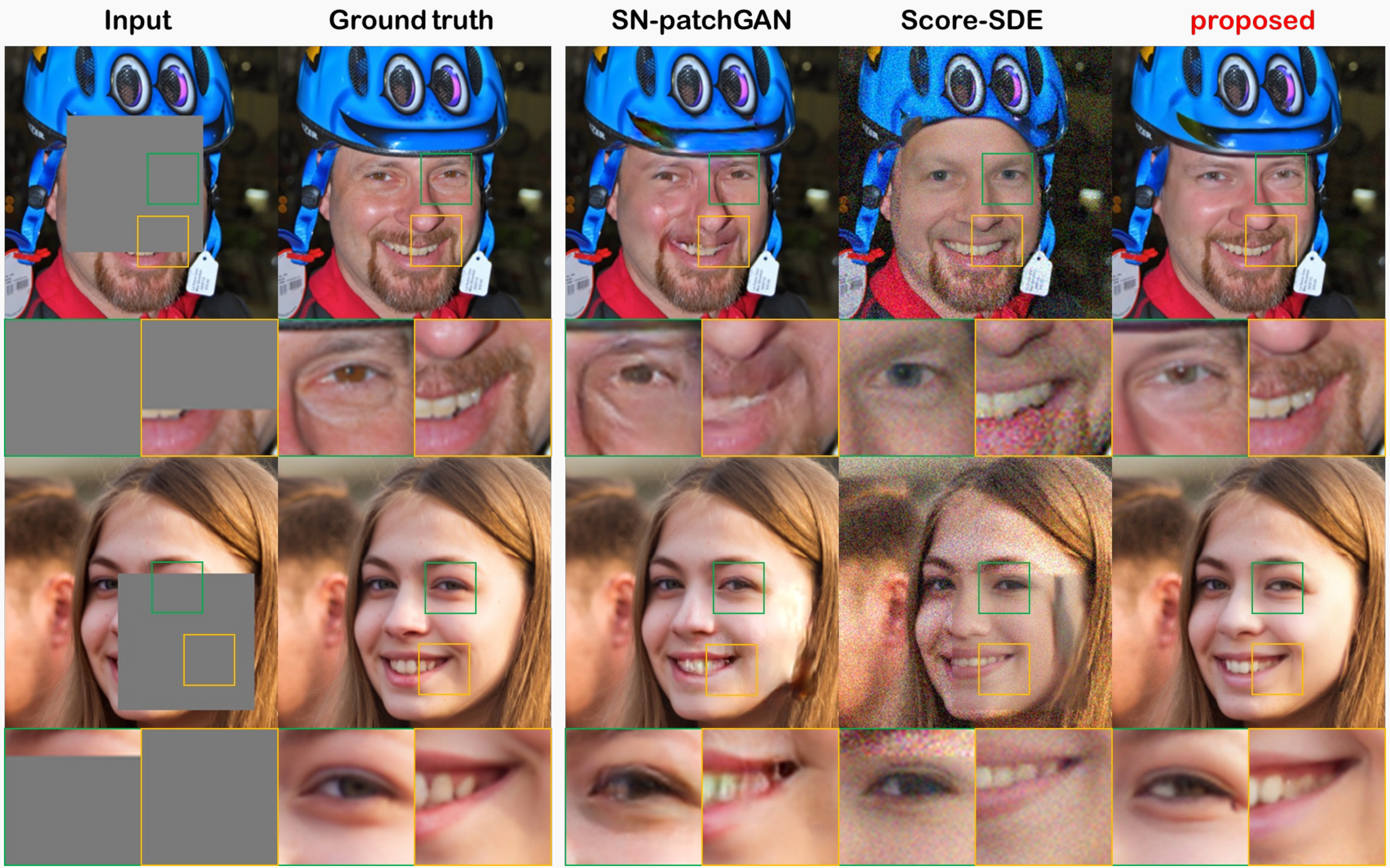
- **proposed**

  $N = 100, \quad t_0 = 0.2$

t_0	0.05	0.1	0.2	0.5	0.75	1.0 [5]
SR ×4	63.90	**60.90**	<u>60.91</u>	64.04	64.14	63.31
SR ×8	85.21	78.13	**75.76**	79.34	79.67	<u>77.34</u>
SR ×16	116.37	101.79	92.59	**88.09**	92.12	<u>88.49</u>

Table 1. FID(↓) scores on FFHQ test set for SR task with $N = 1000$, and varying t_0 values. $t_0 = 1.0$ is the baseline method without any acceleration used in [5]. Numbers in boldface, and underline indicate the best, and the second best scores.

Experimental Results: Inpainting



20 step diffusion

- Score-SDE

  $N = 20, \qquad t_0 = 1.0$

- proposed

  $N = 100, \qquad t_0 = 0.2$

What if We Do Not Know the Forward Model?

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

unknown

[CVPR 2023] Chung et al. Parallel Diffusion Models of Operator and Image for Blind Inverse

What if We Do Not Know the Forward Model?

$$\mathbf{y} = \mathcal{A}_{\phi}(\mathbf{x}) + \boldsymbol{\eta}$$

unknown

[CVPR 2023] Chung et al. Parallel Diffusion Models of Operator and Image for Blind Inverse

What if We Do Not Know the Forward Model?

$$\mathbf{y} = \mathcal{A}_{\phi}(\mathbf{x}) + \boldsymbol{\eta}$$

unknown

$\mathcal{A}_{\phi}$

Blind deconvolution (deblurring)

[CVPR 2023] Chung et al. Parallel Diffusion Models of Operator and Image for Blind Inverse Problems

Diffusion Model for Blind Deconvolution

$$p(\boldsymbol{y}|\boldsymbol{x}_0, \boldsymbol{k}_\mathbf{0}) = \mathcal{N}(\boldsymbol{y}|\boldsymbol{k}_\mathbf{0} * \boldsymbol{x}_\mathbf{0}, \sigma^2 \boldsymbol{I})$$

$$p(\boldsymbol{y}|\boldsymbol{x}_0, \boldsymbol{k}_\mathbf{0}) \propto p(\boldsymbol{y}|\boldsymbol{x}_\mathbf{0}, \boldsymbol{k}_\mathbf{0}) p(\boldsymbol{x}_\mathbf{0}) p(\boldsymbol{k}_\mathbf{0})$$

[CVPR 2023] Chung et al. Parallel Diffusion Models of Operator and Image for Blind Inverse

Diffusion Model for Blind Deconvolution

$$p(\boldsymbol{y}|\boldsymbol{x}_0, \boldsymbol{k_0}) = \mathcal{N}(\boldsymbol{y}|\boldsymbol{k_0} * \boldsymbol{x_0}, \sigma^2 \boldsymbol{I})$$

$$p(\boldsymbol{y}|\boldsymbol{x}_0, \boldsymbol{k_0}) \propto p(\boldsymbol{y}|\boldsymbol{x_0}, \boldsymbol{k_0}) p(\boldsymbol{x_0}) p(\boldsymbol{k_0})$$

Image prior

[CVPR 2023] Chung et al. Parallel Diffusion Models of Operator and Image for Blind Inverse

Diffusion Model for Blind Deconvolution

$$p(\boldsymbol{y}|\boldsymbol{x}_0, \boldsymbol{k}_\mathbf{0}) = \mathcal{N}(\boldsymbol{y}|\boldsymbol{k}_\mathbf{0} * \boldsymbol{x}_\mathbf{0}, \sigma^2 \boldsymbol{I})$$

$$p(\boldsymbol{y}|\boldsymbol{x}_0, \boldsymbol{k}_\mathbf{0}) \propto p(\boldsymbol{y}|\boldsymbol{x}_\mathbf{0}, \boldsymbol{k}_\mathbf{0}) p(\boldsymbol{x}_\mathbf{0}) p(\boldsymbol{k}_\mathbf{0})$$

kernel prior

[CVPR 2023] Chung et al. Parallel Diffusion Models of Operator and Image for Blind Inverse

Diffusion Model for Blind Deconvolution

$$p(\boldsymbol{y}|\boldsymbol{x}_0, \boldsymbol{k}_\mathbf{0}) = \mathcal{N}(\boldsymbol{y}|\boldsymbol{k}_\mathbf{0} * \boldsymbol{x}_\mathbf{0}, \sigma^2 \boldsymbol{I})$$

$$p(\boldsymbol{y}|\boldsymbol{x}_0, \boldsymbol{k}_\mathbf{0}) \propto p(\boldsymbol{y}|\boldsymbol{x}_\mathbf{0}, \boldsymbol{k}_\mathbf{0}) p(\boldsymbol{x}_\mathbf{0}) p(\boldsymbol{k}_\mathbf{0})$$

Choices? Diffusion prior!

[CVPR 2023] Chung et al. Parallel Diffusion Models of Operator and Image for Blind Inverse

Diffusion Model for Blind Deconvolution

$$p(\boldsymbol{y}|\boldsymbol{x}_0, \boldsymbol{k_0}) = \mathcal{N}(\boldsymbol{y}|\boldsymbol{k_0} * \boldsymbol{x_0}, \sigma^2 \boldsymbol{I})$$

$$p(\boldsymbol{y}|\boldsymbol{x}_0, \boldsymbol{k_0}) \propto p(\boldsymbol{y}|\boldsymbol{x_0}, \boldsymbol{k_0}) p(\boldsymbol{x_0}) p(\boldsymbol{k_0})$$

Trained independently with DSM

$$\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}) \simeq s_\theta^I(\boldsymbol{x})$$

$$\nabla_{\boldsymbol{k}} \log p(\boldsymbol{k}) \simeq s_\theta^k(\boldsymbol{k})$$

[CVPR 2023] Chung et al. Parallel Diffusion Models of Operator and Image for Blind Inverse Problems

Diffusion Model for Blind Deconvolution

$$p(\boldsymbol{y}|\boldsymbol{x}_0, \boldsymbol{k_0}) \propto p(\boldsymbol{y}|\boldsymbol{x_0}, \boldsymbol{k_0})p(\boldsymbol{x_0})p(\boldsymbol{k_0})$$

$$d\boldsymbol{x} = \left[-\frac{\beta(t)}{2}\boldsymbol{x} - \beta(t)[\nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t) + \nabla_{\boldsymbol{x}_t} \log p\left(\boldsymbol{y}\middle|\hat{\boldsymbol{x}}_0, \hat{\boldsymbol{k}}_0\right)]\right] dt + \sqrt{\beta(t)}d\,\bar{\boldsymbol{w}}$$

$$d\boldsymbol{k} = \left[-\frac{\beta(t)}{2}\boldsymbol{k} - \beta(t)[\nabla_{\boldsymbol{k}_t} \log p(\boldsymbol{k}_t) + \nabla_{\boldsymbol{k}_t} \log p\left(\boldsymbol{y}\middle|\hat{\boldsymbol{x}}_0, \hat{\boldsymbol{k}}_0\right)]\right] dt + \sqrt{\beta(t)}d\,\bar{\boldsymbol{w}}$$

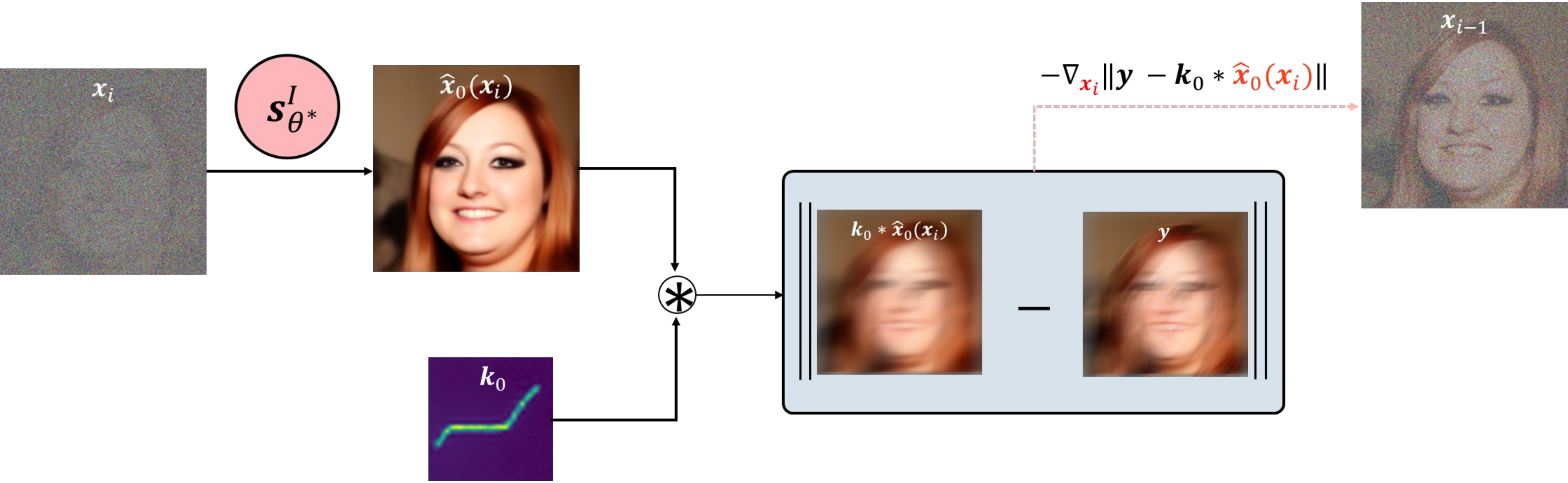
Theorem. Under similar conditions as in DPS,

$$\nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{y}|\boldsymbol{x}_t, \boldsymbol{k}_t) \simeq \nabla_{\mathbf{x}_t} \log p\left(\boldsymbol{y}\middle|\hat{\boldsymbol{x}}_{\mathbf{0}}(\boldsymbol{x}_t), \hat{\boldsymbol{k}}_{\mathbf{0}}(\boldsymbol{k}_t)\right)$$

$$\nabla_{\boldsymbol{k}_t} \log p(\boldsymbol{y}|\boldsymbol{x}_t, \boldsymbol{k}_t) \simeq \nabla_{\boldsymbol{k}_t} \log p\left(\boldsymbol{y}\middle|\hat{\boldsymbol{x}}_{\mathbf{0}}(\boldsymbol{x}_t), \hat{\boldsymbol{k}}_{\mathbf{0}}(\boldsymbol{k}_t)\right)$$

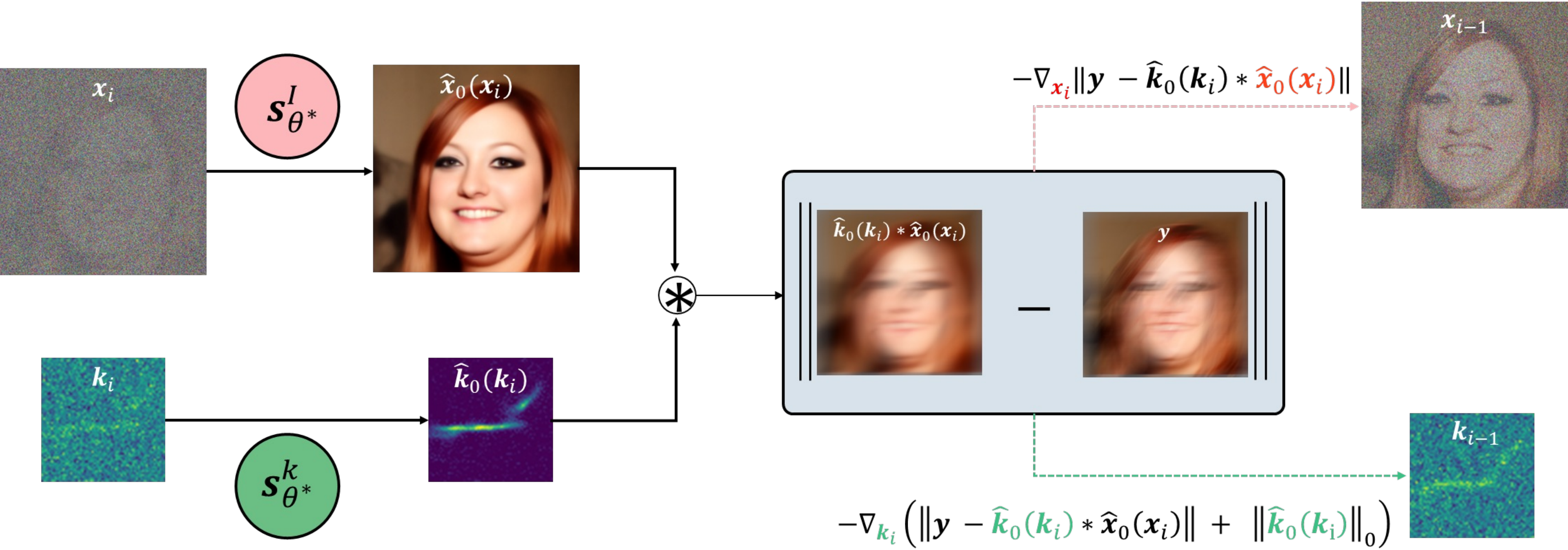
[CVPR 2023] Chung et al. Parallel Diffusion Models of Operator and Image for Blind Inverse Problems

Single Diffusion Model for Non-blind Deconvolution



[CVPR 2023] Chung et al. Parallel Diffusion Models of Operator and Image for Blind Inverse Problems

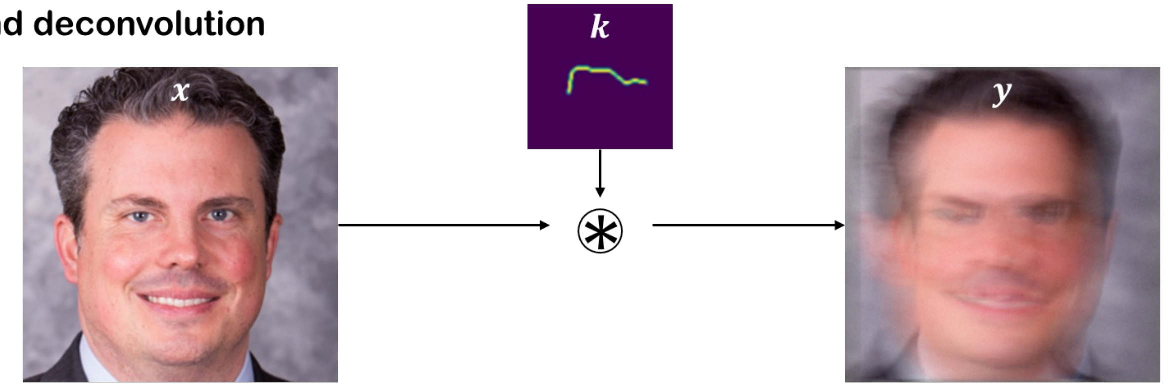
Parallel Diffusion Model for Blind Deconvolution



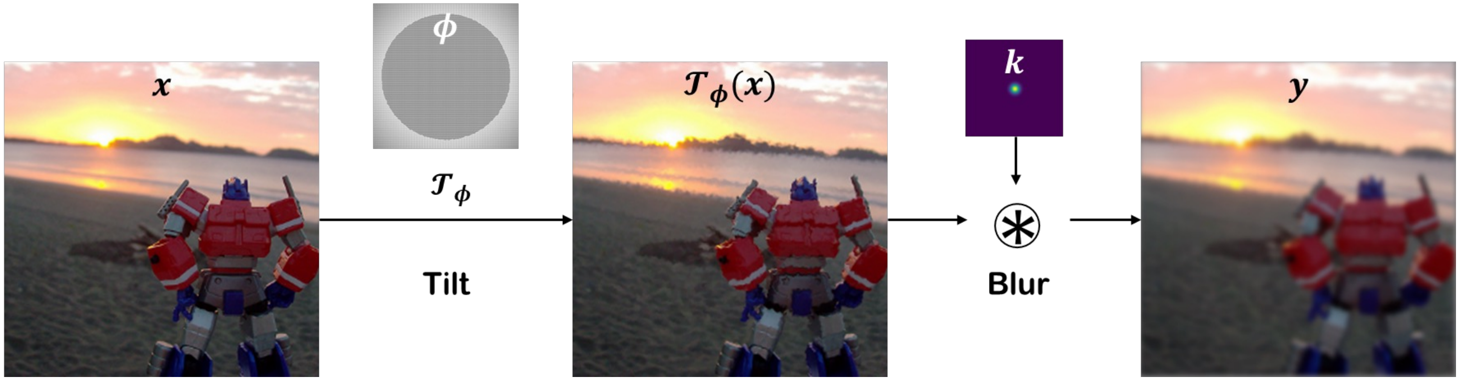
[CVPR 2023] Chung et al. Parallel Diffusion Models of Operator and Image for Blind Inverse Problems

Applicability to Diverse Blind Inverse Problems

1. Blind deconvolution



2. Imaging through turbulence

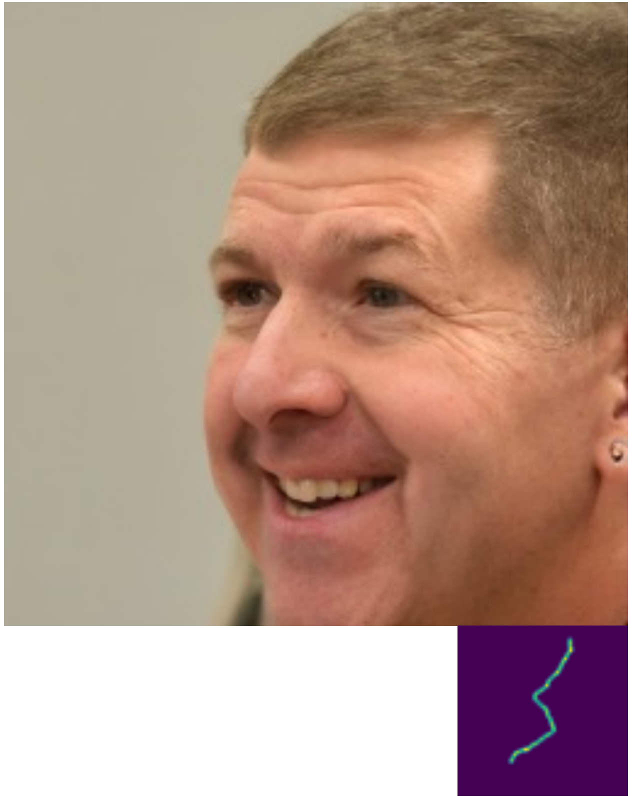


[CVPR 2023] Chung et al. Parallel Diffusion Models of Operator and Image for Blind Inverse

Results: Blind Deblurring

Measurement

Ours

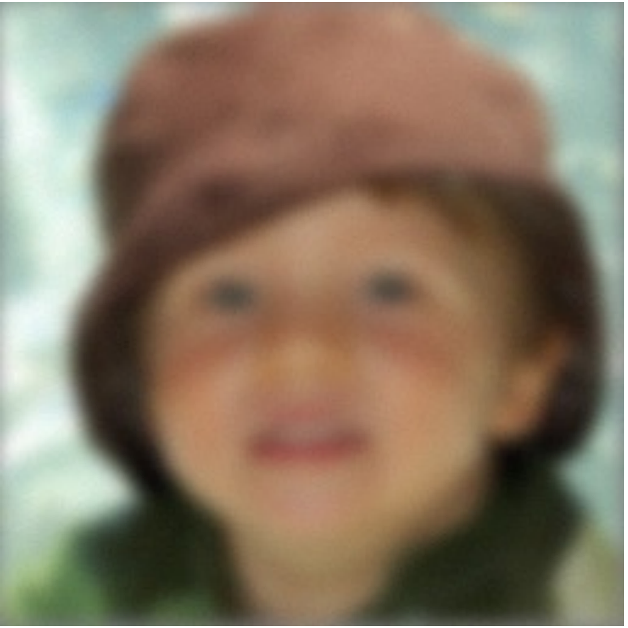


Ground truth

[CVPR 2023] Chung et al. Parallel Diffusion Models of Operator and Image for Blind Inverse

Results: Imaging through Turbulence

Measurement

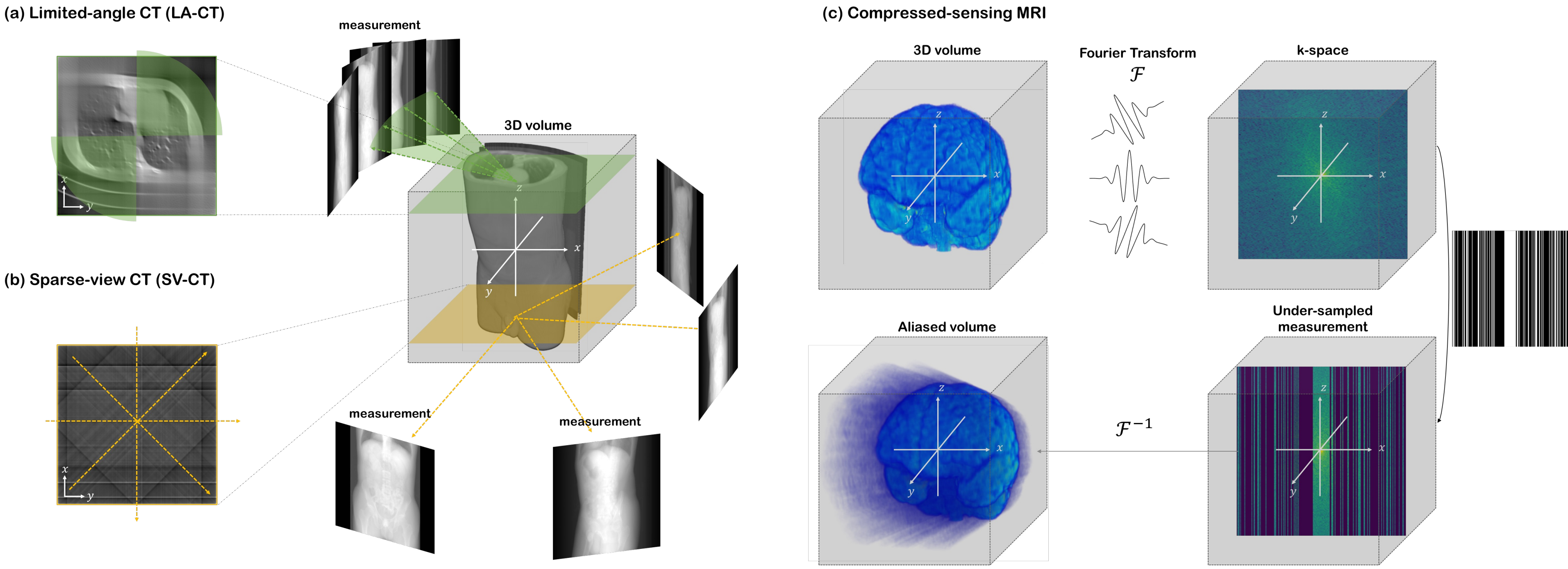


Ours

Ground truth

[CVPR 2023] Chung et al. Parallel Diffusion Models of Operator and Image for Blind Inverse

Scaling Up to 3D Inverse Problems



Naïve application induce **incoherent** reconstructions

[CVPR 2023] Chung et al. Solving 3D Inverse Problems using Pre-trained 2D Diffusion Models

Scaling Up to 3D Inverse Problems

- 3D representations are **memory heavy**
  - **Voxels**: Hard to deal with > 128^3 data
  - **Point clouds**: Sparse representation, but not suitable for medical imaging inverse problems
- **3D voxel diffusion**?
  - The whole diffusion process **stays in the data dimension**
  - **Computationally too heavy**

[CVPR 2023] Chung et al. Solving 3D Inverse Problems using Pre-trained 2D Diffusion Models

Augmenting 2D Diffusion Prior with Model-based Prior

Model-based prior (TV)

$$TV(\boldsymbol{x}) \coloneqq \left\| \left[D_x \boldsymbol{x}, D_y \boldsymbol{x}, D_z \boldsymbol{x} \right] \right\|_1$$

Diffusion prior

$$\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x})$$

[CVPR 2023] Chung et al. Solving 3D Inverse Problems using Pre-trained 2D Diffusion Models

Augmenting 2D Diffusion Prior with Model-based Prior

Model-based prior (TV: z)

$$TV_{\mathrm{z}}(\boldsymbol{x}) \coloneqq \|D_{\mathrm{z}}\boldsymbol{x}\|_1$$

Diffusion prior (xy)

$$\nabla_{\boldsymbol{x}_{\mathrm{xy}}} \log p(\boldsymbol{x}_{\mathrm{xy}})$$

[CVPR 2023] Chung et al. Solving 3D Inverse Problems using Pre-trained 2D Diffusion Models

Augmenting 2D Diffusion Prior with Model-based Prior

1. Denoising with score function (**parallel**)

$$\boldsymbol{x}'_{i-1} \leftarrow \left(\sigma_i^2 - \sigma_{i-1}^2\right)\boldsymbol{s}_{\theta^*}(\boldsymbol{x}_t, t) + \sqrt{\sigma_i^2 - \sigma_{i-1}^2}\,\boldsymbol{\epsilon}$$

2. Data consistency + TV prior augmenting (**joint**)

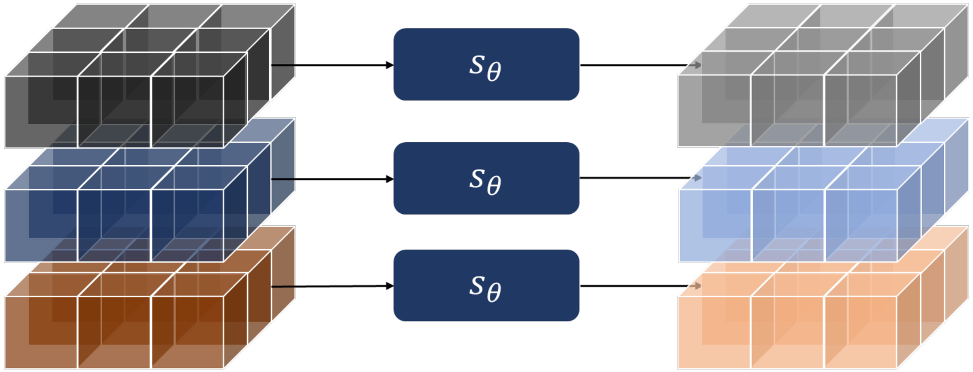
$$\boldsymbol{x}_{\mathrm{i}-1} \leftarrow \underset{\boldsymbol{x}'_{i-1}}{\operatorname{argmin}} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}'_{i-1}\|_2^2 + \|\boldsymbol{D}_{\boldsymbol{z}}\boldsymbol{x}'_{i-1}\|_1$$

Effectively solved with, e.g. ADMM

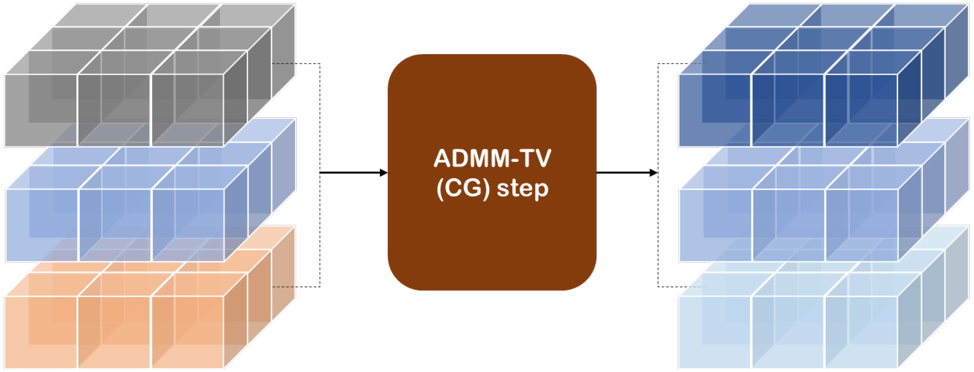
[CVPR 2023] Chung et al. Solving 3D Inverse Problems using Pre-trained 2D Diffusion Models

Augmenting 2D Diffusion Prior with Model-based Prior

1. Score function denoising (parallel)

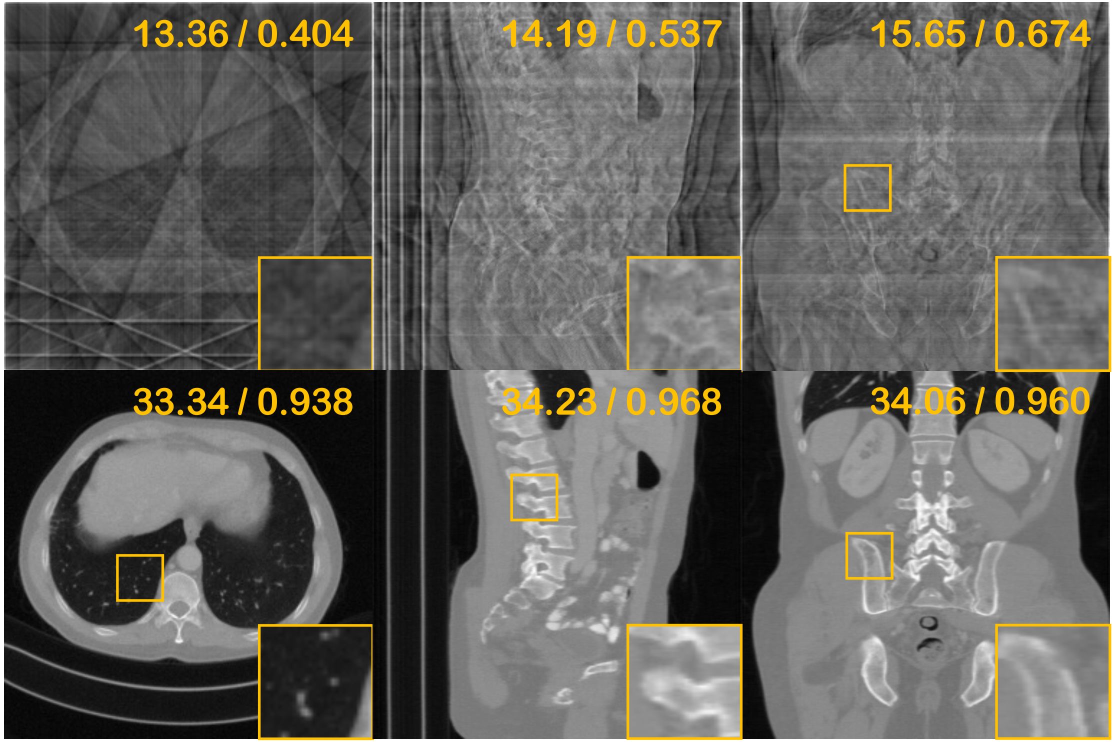


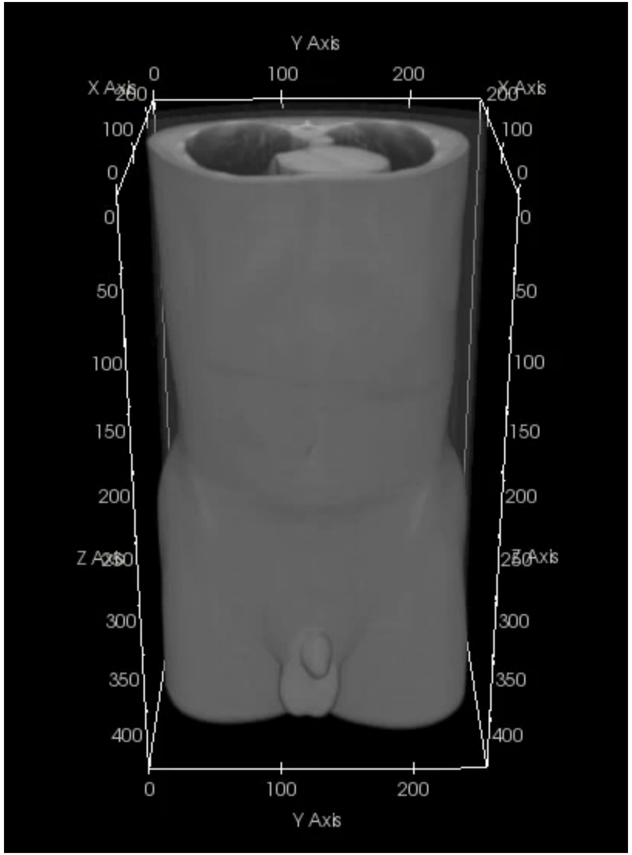
2. ADMM-TV (joint)



[CVPR 2023] Chung et al. Solving 3D Inverse Problems using Pre-trained 2D Diffusion Models

Results: 8-view 3D SV-CT

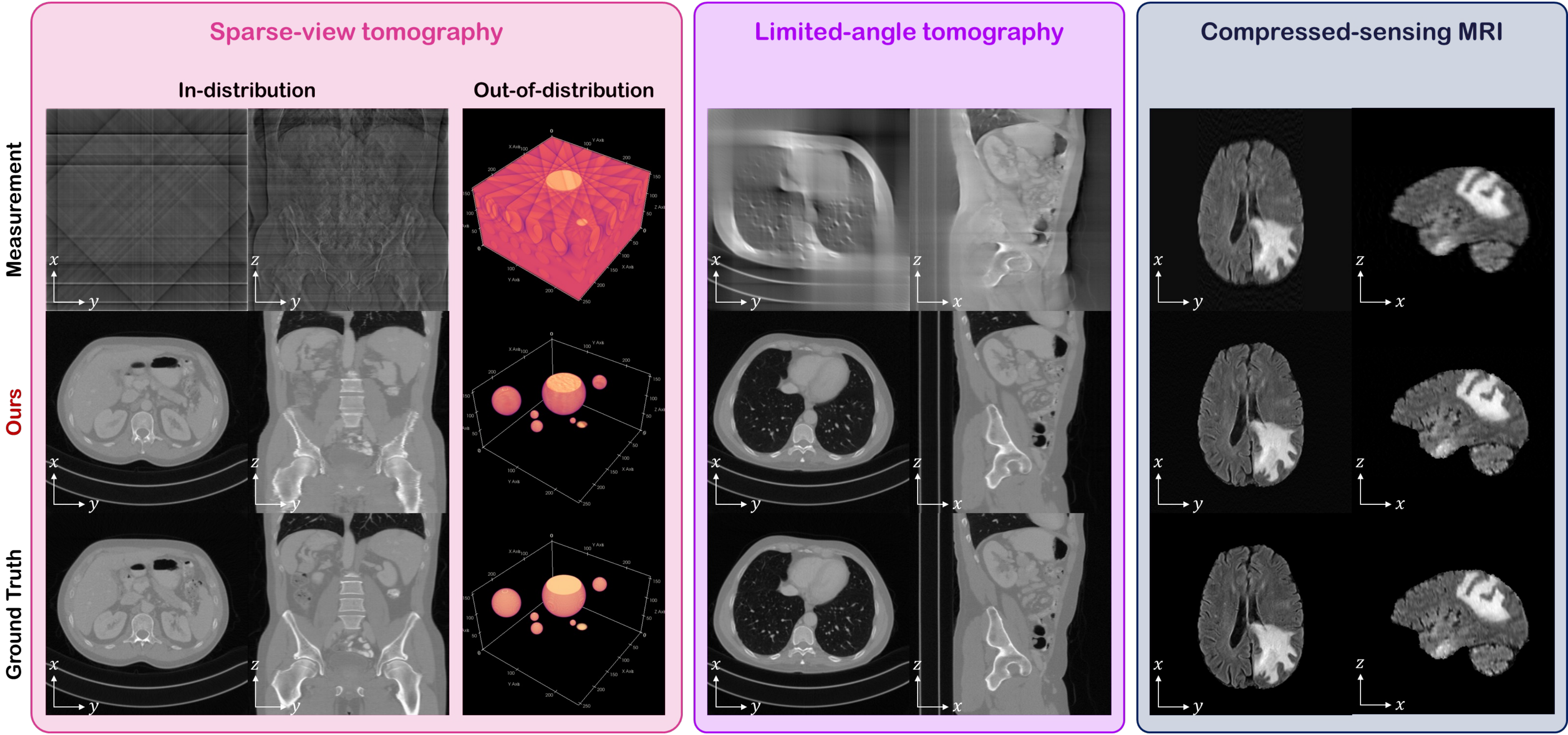




Coherent results across the whole volume

[CVPR 2023] Chung et al. Solving 3D Inverse Problems using Pre-trained 2D Diffusion Models

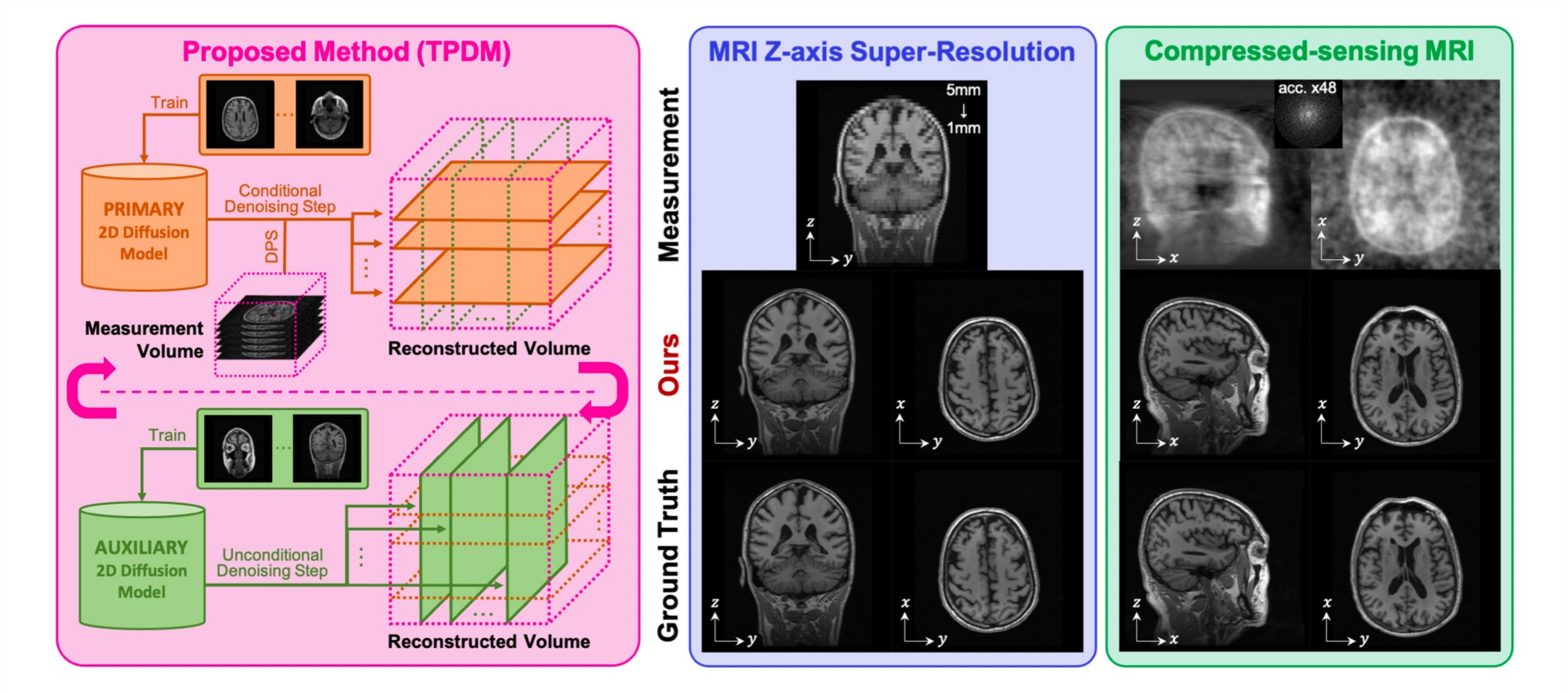
Results: General 3D Problems in Medical Imaging

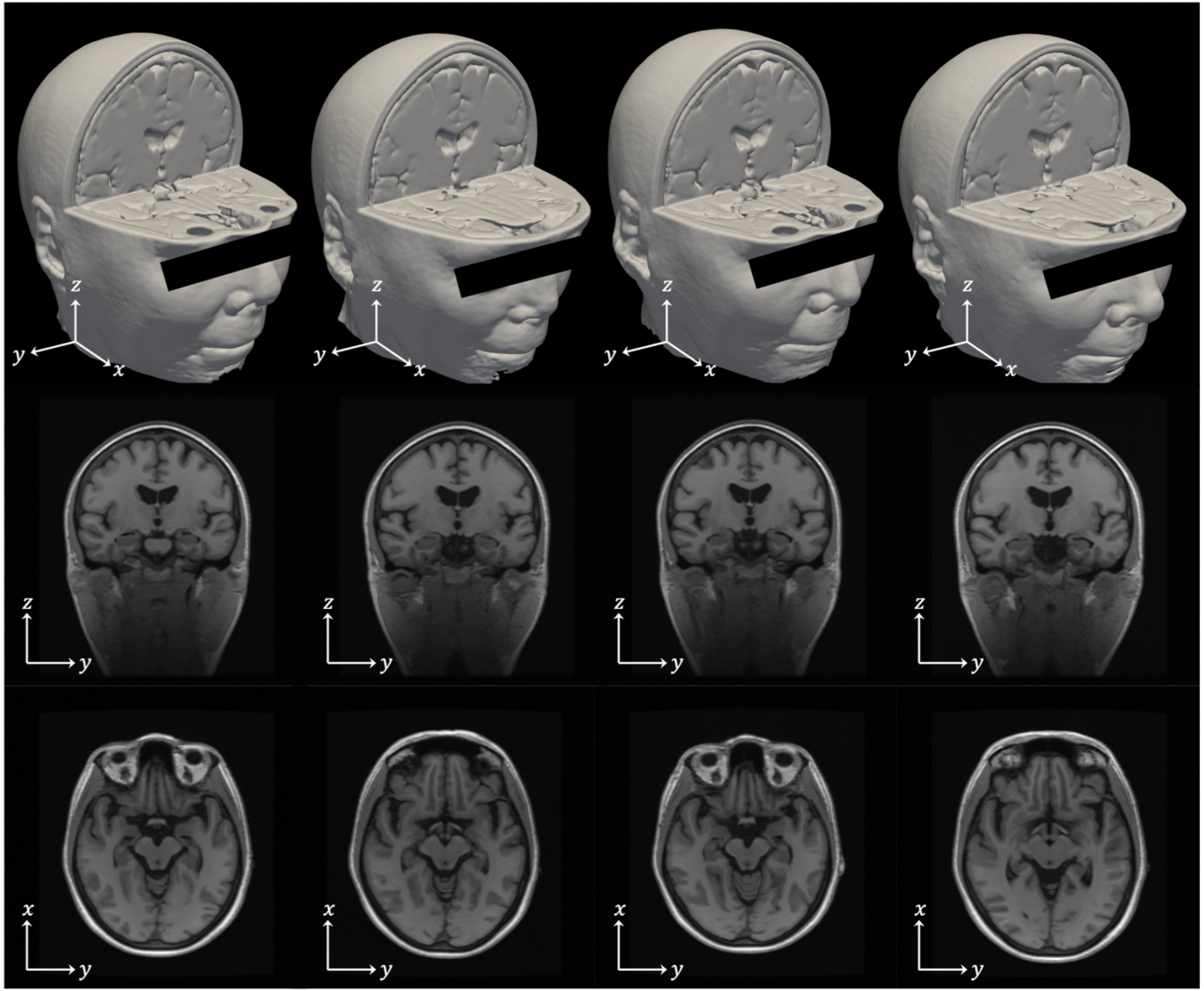


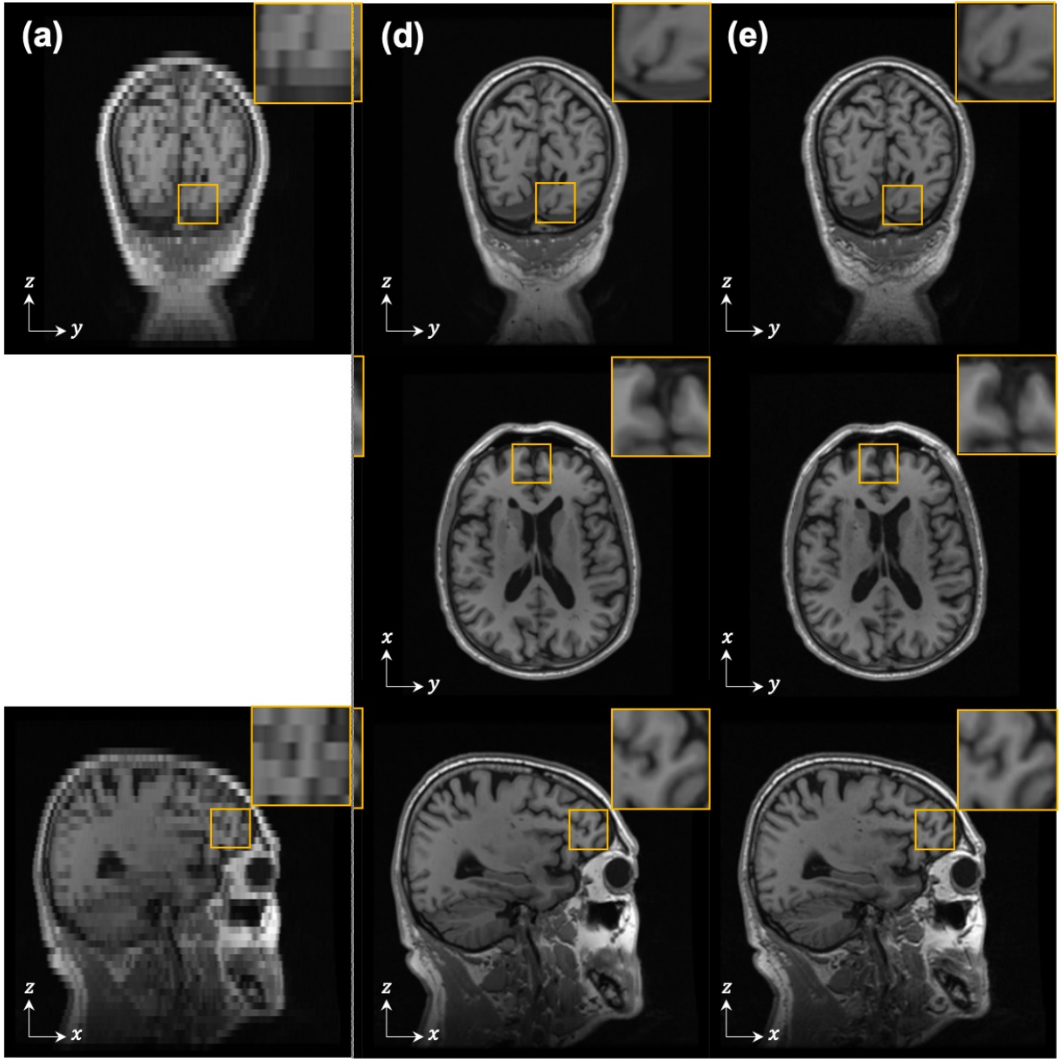
[CVPR 2023] Chung et al. Solving 3D Inverse Problems using Pre-trained 2D Diffusion Models

3D Diffusion using Perpendicular 2D Diffusion Model

Lee et al, arXiv:2303.08440 (2023)




z
y
x
z
y
x
z
y
x
z
y
x
z
y
z
y
z
y
z
y
x
y
x
y
x
y
x
y


(a)
(d)
(e)
z
y
x

Summary

- Score-based approaches: Exciting new path for solving inverse problems without labels
- Universal solver without knowledge about the problem a priori
- Diffusion models: Great generalization capacity
- Acceleration through stochastic contraction theory
- Understanding diffusion geometry helps optimizing algorithms

Questions?