From Guided Sampling to Learned Solvers: Diffusion Models for Inverse Problems

Nicolas Zilberstein
Rice University
PhD defense

September 16, 2026

ChatGPT

ChatGPT

A description of Rice

Here's a concise and clear description of **Rice University** you can use in academic or professional contexts:

Rice University is a leading private research institution located in Houston, Texas, renowned for its strong emphasis on interdisciplinary collaboration, innovation, and academic excellence. With a low student-to-faculty ratio and a vibrant residential college system, Rice fosters a close-knit and intellectually stimulating community. The university is particularly recognized for its strengths in engineering, natural sciences, computer science, and social sciences, as well as its commitment to translating research into real-world impact. Situated in the heart of Houston's dynamic innovation ecosystem, Rice maintains strong connections with industry, government, and international partners, supporting cutting-edge research that addresses global challenges.

Prompt: "A description of Rice"

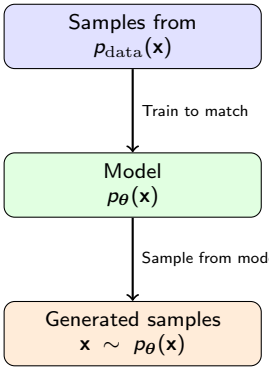
Stable Diffusion (text-to-image)



Prompt: "Rice logo"

High-level idea

1. We are given samples from an unknown data distribution $p_{\text{data}}(\mathbf{x})$
2. We train a model $p_\theta(\mathbf{x})$ to **approximate** this distribution
3. Once trained, we can **sample new data points** $\mathbf{x} \sim p_\theta(\mathbf{x})$ that resemble the real data





Real images



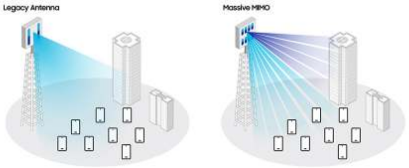
Generated images

- Generating realistic images *per se* is impressive, but often not directly useful
- Can we leverage generative models for **real-world applications**?

- Generating realistic images *per se* is impressive, but often not directly useful
- Can we leverage generative models for **real-world applications**?
- Yes. They have shown remarkable potential across several downstream tasks:
  - $\Rightarrow$ Inverse problems
  - $\Rightarrow$ Graph generation with specific attributes
  - $\Rightarrow$ Molecular and material design
- In this work, we focus on **inverse problems** (and a bit more)

- Inverse problems

  $\Rightarrow$ Recover an unknown signal or quantity from its noisy measurement



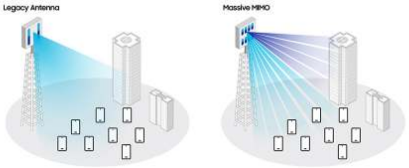
Channel estimation in massive multiple-input multiple-output (MIMO) systems



Image inpainting

- Inverse problems

  $\Rightarrow$ Recover an unknown signal or quantity from its noisy measurement



Channel estimation in massive multiple-input multiple-output (MIMO) systems



Image inpainting

- Challenges in modern systems

  $\Rightarrow$ High-dimensional signals $\rightarrow$ intractable computations

  $\Rightarrow$ Ill-posed problems $\rightarrow$ infinite number of solutions

- We need to impose some structural assumptions

- Forward model for inverse problems:

$$\mathbf{y} = \mathcal{A}(\mathbf{x}_0) + \mathbf{w}, \qquad \mathbf{w} \sim \mathcal{N}(\mathbf{0}, \sigma_0^2 \mathbf{I})$$

  $\Rightarrow$ Unknown signal $\mathbf{x}_0 \in \mathbb{R}^d$, measurement $\mathbf{y} \in \mathbb{R}^m$, known forward operator $\mathcal{A} : \mathbb{R}^d \to \mathbb{R}^m$

  $\Rightarrow$ Linear case: $\mathcal{A}(\mathbf{x}_0) = \mathbf{H}\mathbf{x}_0$ with $\mathbf{H} \in \mathbb{R}^{m \times d}$

- Goal: recover $\mathbf{x}_0$ from $\mathbf{y}$, related by the posterior $p(\mathbf{x}_0|\mathbf{y}) \propto p_{\mathbf{w}}(\mathbf{y}|\mathbf{x}_0)p_{\mathbf{x}}(\mathbf{x}_0)$

- Forward model for inverse problems:

$$\mathbf{y} = \mathcal{A}(\mathbf{x}_0) + \mathbf{w}, \qquad \mathbf{w} \sim \mathcal{N}(\mathbf{0}, \sigma_0^2 \mathbf{I})$$

  $\Rightarrow$ Unknown signal $\mathbf{x}_0 \in \mathbb{R}^d$, measurement $\mathbf{y} \in \mathbb{R}^m$, known forward operator $\mathcal{A} : \mathbb{R}^d \to \mathbb{R}^m$

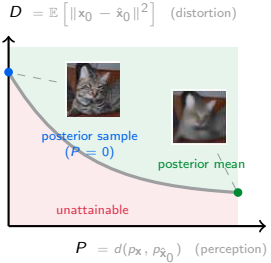
  $\Rightarrow$ Linear case: $\mathcal{A}(\mathbf{x}_0) = \mathbf{H}\mathbf{x}_0$ with $\mathbf{H} \in \mathbb{R}^{m \times d}$

- Goal: recover $\mathbf{x}_0$ from $\mathbf{y}$ ($p(\mathbf{x}_0|\mathbf{y}) \propto p_{\mathbf{w}}(\mathbf{y}|\mathbf{x}_0)p_{\mathbf{x}}(\mathbf{x}_0)$)
- Different estimators based on the downstream task

  $\Rightarrow$ MAP: $\hat{\mathbf{x}}_0^{\mathrm{MAP}} = \mathrm{argmax}_{\mathbf{x}_0} p(\mathbf{x}_0|\mathbf{y})$ (distortion)

  $\Rightarrow$ Posterior sampling: $\hat{\mathbf{x}}_0 \sim p(\mathbf{x}_0|\mathbf{y})$ ($P = 0$)

  $\Rightarrow$ Posterior mean: $\mathbf{x}^\star = \mathbb{E}[\mathbf{x}_0|\mathbf{y}]$ (lowest D)



- Forward model for inverse problems:

$$\mathbf{y} = \mathcal{A}(\mathbf{x}_0) + \mathbf{w}, \qquad \mathbf{w} \sim \mathcal{N}(\mathbf{0}, \sigma_0^2 \mathbf{I})$$

  $\Rightarrow$ Unknown signal $\mathbf{x}_0 \in \mathbb{R}^d$, measurement $\mathbf{y} \in \mathbb{R}^m$, known forward operator $\mathcal{A} : \mathbb{R}^d \to \mathbb{R}^m$

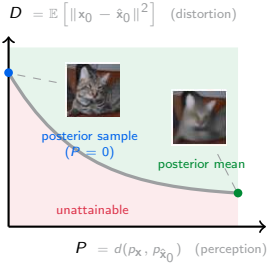
  $\Rightarrow$ Linear case: $\mathcal{A}(\mathbf{x}_0) = \mathbf{H}\mathbf{x}_0$ with $\mathbf{H} \in \mathbb{R}^{m \times d}$

- Goal: recover $\mathbf{x}_0$ from $\mathbf{y}$ ($p(\mathbf{x}_0|\mathbf{y}) \propto p_{\mathbf{w}}(\mathbf{y}|\mathbf{x}_0) p_{\mathbf{x}}(\mathbf{x}_0)$)
- Different estimators based on the downstream task

  $\Rightarrow$ MAP: $\hat{\mathbf{x}}_0^{\mathrm{MAP}} = \mathrm{argmax}_{\mathbf{x}_0} p(\mathbf{x}_0|\mathbf{y})$ (distortion)

  $\Rightarrow$ Posterior sampling: $\hat{\mathbf{x}}_0 \sim p(\mathbf{x}_0|\mathbf{y})$ ($P = 0$)

  $\Rightarrow$ Posterior mean: $\mathbf{x}^\star = \mathbb{E}[\mathbf{x}_0|\mathbf{y}]$ (lowest D)



Parameterize the prior $p_{\mathbf{x}}(\mathbf{x}_0)$ with a diffusion model. Why?

Traditional Priors

- Sparsity
- Total-variation

Learned Priors (Generative Models)

- GANs: Fast and high-quality samples
  $\Rightarrow$ But lack diversity and unstable training
- Normalizing Flows: Exact likelihood
  $\Rightarrow$ But rigid parameterization

- Diffusion Models: Simple and scalable loss, high-quality and diverse samples

Traditional Priors

- Sparsity
- Total-variation

Learned Priors (Generative Models)

- GANs: Fast and high-quality samples
  - $\Rightarrow$ But lack diversity and unstable training
- Normalizing Flows: Exact likelihood
  - $\Rightarrow$ But rigid parameterization

- Diffusion Models: Simple and scalable loss, high-quality and diverse samples
- But using them for solving inverse problems is challenging and has some limitations
  - $\Rightarrow$ Sampling is slow
  - $\Rightarrow$ Imposing external constraints requires approximations
  - $\Rightarrow$ We do not have access to the exact likelihoods/density

Can we build solvers that overcome these limitations?

Contribution of this work

Design diffusion-based inverse solvers by exploiting the different degrees of freedom

Goal	This thesis	Venue
Inference-time algorithms on pre-trained priors		
Fast sampling	✓ Higher-order annealed Langevin diffusion	TSP 2024
Diversity in latent space	✓ Repulsive latent score distillation	ICLR 2025
Hard constraints	✓ Shielded Langevin with navigation potentials	Asilomar 2025
Training the model for improved inference		
Task-informed source	✓ Prior-informed flow matching	SPIGM ICML 2026[†]
Access to the density	✓ Energy models with anisotropic noise	ICML 2026
Control of the output	✓ Flow map denoisers	SPIGM ICML 2026[†]

[†] under review

Contribution of this work

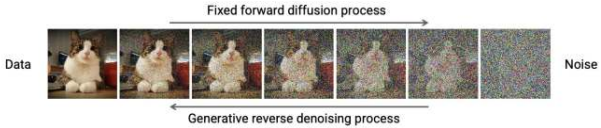
Design diffusion-based inverse solvers by exploiting the different degrees of freedom

Goal	This thesis	Venue
Inference-time algorithms on pre-trained priors		
Fast sampling	✓ Higher-order annealed Langevin diffusion	TSP 2024
Diversity in latent space	✓ Repulsive latent score distillation	ICLR 2025
Hard constraints	✓ Shielded Langevin with navigation potentials	Asilomar 2025
Training the model for improved inference		
Task-informed source	✓ Prior-informed flow matching	SPIGM ICML 2026[†]
Access to the density	✓ Energy models with anisotropic noise	ICML 2026
Control of the output	✓ Flow map denoisers	SPIGM ICML 2026[†]

[†] under review

- Intro to diffusion models
- Inference-time algorithms on pre-trained priors
  - Sampling algorithm
    - Repulsive latent score distillation
- Training the model for improved inference
  - The model itself
    - Energy models with anisotropic noise
    - Flow map denoisers
- Discussion

- Diffusion models consist of two processes
  - $\Rightarrow$ Forward process (go from data to random noise)
  - $\Rightarrow$ Reverse process (go from random noise to data)
  - $\Rightarrow$ $\mathbf{x}_0 \sim p_{\text{data}}(\mathbf{x}_0)$ which is unknown



- Forward process
  - $\Rightarrow$ Continuous-time Markov process in $t \in [0, T]$

$$\mathrm{d}\mathbf{x}_t = \mathbf{f}(t)\mathbf{x}_t \mathrm{d}t + g(t)\mathrm{d}\mathbf{w}_t \quad \mathbf{x}_0 \sim p_{\text{data}},\ \mathbf{x}_T \sim p_T$$

  - $\Rightarrow$ $\mathbf{f}(t)\mathbf{x}_t$: drift
  - $\Rightarrow$ $\mathbf{w}_t$: Brownian motion
  - $\Rightarrow$ p_T typically $\mathcal{N}(\mathbf{0}, \mathbf{I})$
- Backward process in $t \in [T, 0]$ [Anderson 1982]:

$$\mathrm{d}\mathbf{x}_t = \left[\mathbf{f}(t)\mathbf{x}_t - g(t)^2 \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)\right] \mathrm{d}t + g(t)\mathrm{d}\overline{\mathbf{w}}_t$$

- $\mathbf{s}_t(\mathbf{x}_t) := \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$ is the score function $\rightarrow$ unknown

- Forward process
  - $\Rightarrow$ Continuous-time Markov process in $t \in [0, T]$

$$\mathrm{d}\mathbf{x}_t = \mathbf{f}(t)\mathbf{x}_t \mathrm{d}t + g(t)\mathrm{d}\mathbf{w}_t \quad \mathbf{x}_0 \sim p_{\text{data}},\ \mathbf{x}_T \sim p_T$$

  - $\Rightarrow$ $\mathbf{f}(t)\mathbf{x}_t$: drift
  - $\Rightarrow$ $\mathbf{w}_t$: Brownian motion
  - $\Rightarrow$ p_T typically $\mathcal{N}(\mathbf{0}, \mathbf{I})$
- Backward process in $t \in [T, 0]$ [Anderson 1982]:

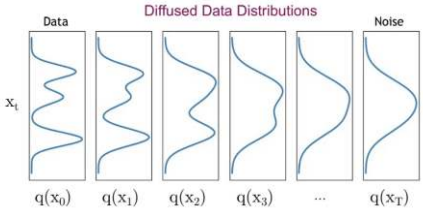
$$\mathrm{d}\mathbf{x}_t = \left[\mathbf{f}(t)\mathbf{x}_t - g(t)^2 \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)\right] \mathrm{d}t + g(t)\mathrm{d}\overline{\mathbf{w}}_t$$

- $\mathbf{s}_t(\mathbf{x}_t) := \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$ is the score function $\rightarrow$ unknown
- What do we need besides the score?
  - $\Rightarrow$ Specify $\mathbf{f}(\cdot)$, $g(\cdot)$ and p_T $\rightarrow$ define the particular instance of the process
  - $\Rightarrow$ Discretization scheme

As an example, we consider the variance exploding instance: $\mathbf{f}(t) = 0, g(t) = \sqrt{\frac{\mathrm{d}\sigma_t^2}{\mathrm{d}t}}$

- Forward process:
  Convolution with Gaussian kernel
  $p_t(\mathbf{x}_t) = \int p_{\text{data}}(\mathbf{x}_0) p_{t|0}(\mathbf{x}_t|\mathbf{x}_0) \mathrm{d}\mathbf{x}_0$

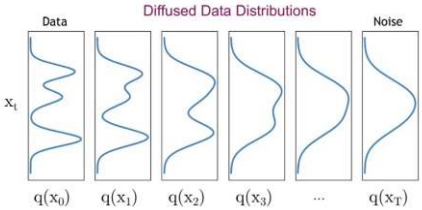
$$\mathbf{x}_t = \mathbf{x}_0 + \sigma_t \boldsymbol{\epsilon} \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$



As an example, we consider the variance exploding instance: $\mathbf{f}(t) = 0, g(t) = \sqrt{\frac{\mathrm{d}\sigma_t^2}{\mathrm{d}t}}$

- Forward process:
  Convolution with Gaussian kernel
  $p_t(\mathbf{x}_t) = \int p_{\text{data}}(\mathbf{x}_0) p_{t|0}(\mathbf{x}_t|\mathbf{x}_0) \mathrm{d}\mathbf{x}_0$

$$\mathbf{x}_t = \mathbf{x}_0 + \sigma_t \boldsymbol{\epsilon} \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$



- Backward process
  Denoising operation

$$\mathbf{x}_t = \mathbf{x}_{t+\tau} + \eta_{t+\tau} \nabla_{\mathbf{x}_{t+\tau}} \log p_{t+\tau}(\mathbf{x}_{t+\tau}) + \sqrt{2\eta_{t+\tau}}\, \mathbf{w}_t, \qquad \eta \text{ the step size}$$

$$\mathbb{E}[\mathbf{x}_0|\mathbf{x}_t] = \mathbf{x}_t + \sigma_t^2 \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) \quad \text{(Tweedie)}$$

- We need an estimation of $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) \rightarrow$ parameterize with a neural network $\mathbf{s}_\theta(\mathbf{x}_t, t)$
  $\Rightarrow$ Train with denoising score matching

1. Design choices: fix the components of the diffusion

$$\mathrm{d}\mathbf{x}_t = \mathbf{f}(t)\mathbf{x}_t\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}_t \quad \mathbf{x}_T \sim p_T,\ \mathbf{x}_0 \sim p_{\text{data}}(\mathbf{x}_0)$$

2. Training: given a dataset $\{\mathbf{x}_0^{(i)}\}_{i=1}^N \sim p_{\text{data}}$, optimize[1]

$$\mathcal{L}_{\text{DSM}}(\theta) = \int_0^T \tilde{\lambda}(t)\,\mathbb{E}_{\mathbf{x}_0,\mathbf{x}_t}\left[||\mathbf{D}_{\boldsymbol{\theta}}(\mathbf{x}_t,\sigma_t) - \mathbf{x}_0||_2^2\right]\mathrm{d}t,$$

where, by Tweedie, $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t,t) = \frac{\mathbf{D}_{\boldsymbol{\theta}}(\mathbf{x}_t,\sigma_t)-\mathbf{x}_t}{\sigma_t^2}$, with $\mathbf{D}_{\boldsymbol{\theta}}(\mathbf{x}_t,\sigma_t) := \mathbb{E}\left[\mathbf{x}_0|\mathbf{x}_t\right]$

$\Rightarrow$ DSM upper-bounds the KL:

$$\mathrm{KL}(p_{\text{data}} \,\|\, p_\theta) \le \mathcal{L}_{\text{DSM}}(\theta) + C$$

3. Inference: run the backward process using some discretization

$$\left.\begin{aligned}\text{SDE: } \mathrm{d}\mathbf{x}_t &= \left[\mathbf{f}(t)\mathbf{x}_t - g(t)^2\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)\right]\mathrm{d}t + g(t)\mathrm{d}\overline{\mathbf{w}}_t \\ \text{ODE: } \mathrm{d}\mathbf{x}_t &= \left[\mathbf{f}(t)\mathbf{x}_t - \tfrac{1}{2}g(t)^2\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)\right]\mathrm{d}t\end{aligned}\right\} \text{ same } p_t(\mathbf{x}_t)$$

[1] Using σ_t and t is equivalent

1. Design choices: the process $\mathbf{f}(t), g(t)$ and the source p_T
2. Training: the objective $\mathcal{L}_{\mathrm{DSM}}(\theta)$
3. Inference: simulate the reverse process with a *fixed, learned* model

Step 3 is the **sampler**

Steps 1–2 are the **model**

Inference-time algorithms using pre-trained priors

- Repulsive latent score distillation

Training the model for improved inference

- Energy models with anisotropic noise
- Flow map denoisers

- Intro to diffusion models
- Inference-time algorithms on pre-trained priors
  - Sampling algorithm
    - Repulsive latent score distillation
- Training the model for improved inference
  - The model itself
    - Energy models with anisotropic noise
    - Flow map denoisers
- Discussion

- Recall we want to recover $\mathbf{x}_0$ from $\mathbf{y}$, through $p(\mathbf{x}_0|\mathbf{y}) \propto p_{\mathbf{w}}(\mathbf{y}|\mathbf{x}_0)p_{\mathbf{x}}(\mathbf{x}_0)$
  $\Rightarrow \mathbf{y} = \mathcal{A}(\mathbf{x}_0) + \mathbf{w}$, $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \sigma_0^2\mathbf{I})$

- Recall we want to recover $\mathbf{x}_0$ from $\mathbf{y}$, through $p(\mathbf{x}_0|\mathbf{y}) \propto p_{\mathbf{w}}(\mathbf{y}|\mathbf{x}_0)p_{\mathbf{x}}(\mathbf{x}_0)$
  $\Rightarrow \mathbf{y} = \mathcal{A}(\mathbf{x}_0) + \mathbf{w}$, $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \sigma_0^2\mathbf{I})$
- We need the *conditional* score $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t|\mathbf{y})$

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

- Recall we want to recover $\mathbf{x}_0$ from $\mathbf{y}$, through $p(\mathbf{x}_0|\mathbf{y}) \propto p_{\mathbf{w}}(\mathbf{y}|\mathbf{x}_0)p_{\mathbf{x}}(\mathbf{x}_0)$
  $\Rightarrow \mathbf{y} = \mathcal{A}(\mathbf{x}_0) + \mathbf{w}$, $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \sigma_0^2\mathbf{I})$
- We need the *conditional* score $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t|\mathbf{y})$

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

- $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) \rightarrow$ parameterized with the score network $\mathbf{s}_\theta(\mathbf{x}_t, t)$ (unconditional prior)
- $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{y}|\mathbf{x}_t) \rightarrow$ intractable

$$p_t(\mathbf{y}|\mathbf{x}_t) = \int p(\mathbf{y}|\mathbf{x}_0) p_{0|t}(\mathbf{x}_0|\mathbf{x}_t) \mathrm{d}\mathbf{x}_0$$

- Several approximations have been proposed (SVD, Heuristics, Implicit prior)

$$p_t(\mathbf{y}|\mathbf{x}_t) \approx p(\mathbf{y} \mid \hat{\mathbf{x}}_0 := \mathbb{E}[\mathbf{x}_0|\mathbf{x}_t]) = \mathcal{N}\left(\mathbf{y}; \mathcal{A}(\hat{\mathbf{x}}_0), \sigma_0^2\mathbf{I}\right) \quad \text{[Chung 2023]}$$

- These approximations are
  - $\Rightarrow$ Computationally expensive as we need to backpropagate through the score network

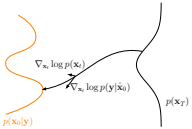
$$\nabla_{\mathbf{x}_t} \log p_t\left(\mathbf{y}|\mathbf{x}_t\right) \approx \nabla_{\mathbf{x}_t} \log \mathcal{N}\left(\mathbf{y}; \mathcal{A}(\mathbf{D}_{\boldsymbol{\theta}}(\mathbf{x}_t, \sigma_t)), \sigma_0^2 \mathbf{I}\right)$$

  - $\Rightarrow$ Slow as we need to run the full backward process

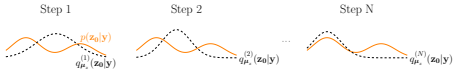
- These approximations are
  - ⇒ Computationally expensive as we need to backpropagate through the score network

$$\nabla_{\mathbf{x}_t} \log p_t(\mathbf{y}|\mathbf{x}_t) \approx \nabla_{\mathbf{x}_t} \log \mathcal{N}\left(\mathbf{y}; \mathcal{A}(\mathbf{D}_{\boldsymbol{\theta}}(\mathbf{x}_t, \sigma_t)), \sigma_0^2 \mathbf{I}\right)$$

  - ⇒ Slow as we need to run the full backward process
- How can we improve the inference, particularly with latent diffusion (latest models)?
  - ⇒ Derive a better sampling (backward) process → Higher-order Langevin
  - ⇒ Derive a different approximation of the likelihood → Variational inference



Improving the sampling process



Approximation via variational inference

- Latest diffusion models are defined in a latent space
  - $\Rightarrow$ Scale better (lower dimensional diffusion)
  - $\Rightarrow$ Forward and backward process in a latent space $\mathcal{Z}$
  - $\Rightarrow$ Generated image is $\mathbf{x}_0 = \mathcal{D}_\phi(\mathbf{z}_0)$, with $\mathcal{D}_\phi$ the decoder

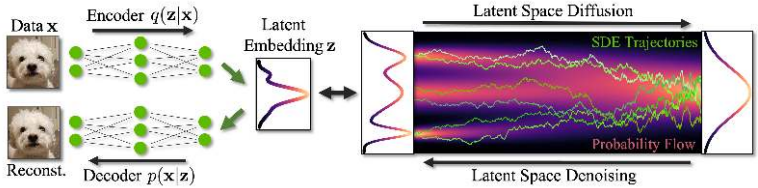


Image from https://neurips2023-ldm-tutorial.github.io/

- Latest diffusion models are defined in a latent space
  - $\Rightarrow$ Scale better (lower dimensional diffusion)
  - $\Rightarrow$ Forward and backward process in a latent space $\mathcal{Z}$
  - $\Rightarrow$ Generated image is $\mathbf{x}_0 = \mathcal{D}_\phi(\mathbf{z}_0)$, with $\mathcal{D}_\phi$ the decoder

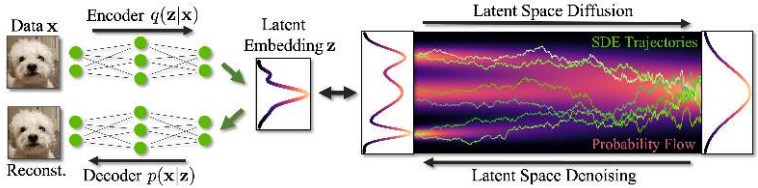


Image from https://neurips2023-ldm-tutorial.github.io/

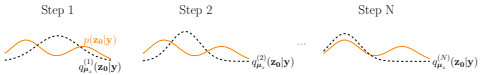
- Backpropagation through the decoder and the score network $\rightarrow$ expensive

- Variational (optimization) perspective of sampling from the posterior
- Given the posterior $p(\mathbf{x}_0|\mathbf{y})$, we seek a variational approximation

$$q^\star(\mathbf{x}_0|\mathbf{y}) = \underset{q(\mathbf{x}_0|\mathbf{y})}{\operatorname{argmin}} \mathrm{KL}\big(q(\mathbf{x}_0|\mathbf{y}) \| p(\mathbf{x}_0|\mathbf{y})\big) \propto \mathbb{E}_{q(\mathbf{x}_0|\mathbf{y})}[-\log p(\mathbf{y}|\mathbf{x}_0)] + \mathrm{KL}(q(\mathbf{x}_0|\mathbf{y}) \| p_{\mathbf{x}}(\mathbf{x}_0))$$

where $q(\mathbf{x}_0|\mathbf{y}) \sim \mathcal{N}(\boldsymbol{\mu}, \sigma^2\mathbf{I})$

$\Rightarrow$ Reverse KL (not forward KL / max. likelihood) $\rightarrow$ mode-seeking



- How can we incorporate the diffusion model as a prior?

- Variational (optimization) perspective of sampling from the posterior
- Given the posterior $p(\mathbf{x}_0|\mathbf{y})$, we seek a variational approximation

$$q^\star(\mathbf{x}_0|\mathbf{y}) = \underset{q(\mathbf{x}_0|\mathbf{y})}{\operatorname{argmin}} \operatorname{KL}\big(q(\mathbf{x}_0|\mathbf{y}) \| p(\mathbf{x}_0|\mathbf{y})\big) \propto \mathbb{E}_{q(\mathbf{x}_0|\mathbf{y})}[-\log p(\mathbf{y}|\mathbf{x}_0)] + \operatorname{KL}(q(\mathbf{x}_0|\mathbf{y}) \| p_{\mathbf{x}}(\mathbf{x}_0))$$

  where $q(\mathbf{x}_0|\mathbf{y}) \sim \mathcal{N}(\boldsymbol{\mu}, \sigma^2\mathbf{I})$

  $\Rightarrow$ Reverse KL (not forward KL / max. likelihood) $\rightarrow$ mode-seeking

- How can we incorporate the diffusion model as a prior? Reuse the DSM loss of the pretrained denoiser

$$\min_{\boldsymbol{\mu}} \underbrace{\tfrac{1}{2\sigma_0^2}\|\mathbf{y} - \mathcal{A}(\boldsymbol{\mu})\|^2}_{\text{data fidelity}} + \underbrace{\int_0^T \tilde{\lambda}(t)\, \mathbb{E}_{\boldsymbol{\epsilon}} \left[\|\mathbf{D}_{\boldsymbol{\theta}}(\mathbf{x}_t, \sigma_t) - \boldsymbol{\mu}\|_2^2\right] \mathrm{d}t}_{\mathcal{L}_{\mathrm{DSM}} \text{ evaluated at } \mathbf{x}_0 \leftarrow \boldsymbol{\mu}}$$

  $\Rightarrow$ $\mathbf{x}_t = \alpha_t \boldsymbol{\mu} + \sigma_t \boldsymbol{\epsilon}$: denoisers at all noise scales regularize the optimization

  $\Rightarrow$ As $\sigma \rightarrow 0$, $q \rightarrow \delta(\mathbf{x}_0 - \boldsymbol{\mu}^\star)$: we recover the MAP estimator

- Gradient of the distillation term is tractable [Mardani et al., 2023]: stop-gradient on $\boldsymbol{\theta}$

$$\nabla_{\boldsymbol{\mu}} \mathbb{E}_{\boldsymbol{\epsilon}}\left[\tilde{\lambda}(t)\|\mathbf{D}_{\boldsymbol{\theta}}(\mathbf{x}_t, \sigma_t) - \boldsymbol{\mu}\|_2^2\right] \propto \boldsymbol{\mu} - \mathbf{D}_{\boldsymbol{\theta}}(\mathbf{x}_t, \sigma_t), \qquad \mathbf{x}_t = \alpha_t \boldsymbol{\mu} + \sigma_t \boldsymbol{\epsilon}$$

Score distillation sampling

Require: measurement $\mathbf{y}$, pretrained denoiser $\mathbf{D}_{\boldsymbol{\theta}}$, step size η
  initialize $\boldsymbol{\mu} \leftarrow \mathcal{A}^\dagger \mathbf{y}$
  for $k = 1, \ldots, K$ **do**
    $t \sim \mathcal{U}[0, T], \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad \mathbf{x}_t = \alpha_t \boldsymbol{\mu} + \sigma_t \boldsymbol{\epsilon}$ ▷ noise current estimate
    $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} - \eta\left[\nabla_{\boldsymbol{\mu}} \frac{1}{2\sigma_0^2}\|\mathbf{y} - \mathcal{A}(\boldsymbol{\mu})\|^2 + \tilde{\lambda}(t)\left(\boldsymbol{\mu} - \mathbf{D}_{\boldsymbol{\theta}}(\mathbf{x}_t, \sigma_t)\right)\right]$
  end for

- One denoiser call per step (vs. backpropagating through the score network)

- Gradient of the distillation term is tractable [Mardani et al., 2023]: stop-gradient on $\boldsymbol{\theta}$

$$\nabla_{\boldsymbol{\mu}} \mathbb{E}_{\boldsymbol{\epsilon}} \left[\tilde{\lambda}(t) \| \mathbf{D}_{\boldsymbol{\theta}}(\mathbf{x}_t, \sigma_t) - \boldsymbol{\mu} \|_2^2 \right] \propto \boldsymbol{\mu} - \mathbf{D}_{\boldsymbol{\theta}}(\mathbf{x}_t, \sigma_t), \qquad \mathbf{x}_t = \alpha_t \boldsymbol{\mu} + \sigma_t \boldsymbol{\epsilon}$$

Score distillation sampling

Require: measurement $\mathbf{y}$, pretrained denoiser $\mathbf{D}_{\boldsymbol{\theta}}$, step size η
initialize $\boldsymbol{\mu} \leftarrow \mathcal{A}^\dagger \mathbf{y}$
for $k = 1, \dots, K$ **do**
$t \sim \mathcal{U}[0, T], \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad \mathbf{x}_t = \alpha_t \boldsymbol{\mu} + \sigma_t \boldsymbol{\epsilon}$ ▷ noise current estimate
$\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} - \eta \left[\nabla_{\boldsymbol{\mu}} \frac{1}{2\sigma_0^2} \| \mathbf{y} - \mathcal{A}(\boldsymbol{\mu}) \|^2 + \tilde{\lambda}(t) \left(\boldsymbol{\mu} - \mathbf{D}_{\boldsymbol{\theta}}(\mathbf{x}_t, \sigma_t) \right) \right]$
end for

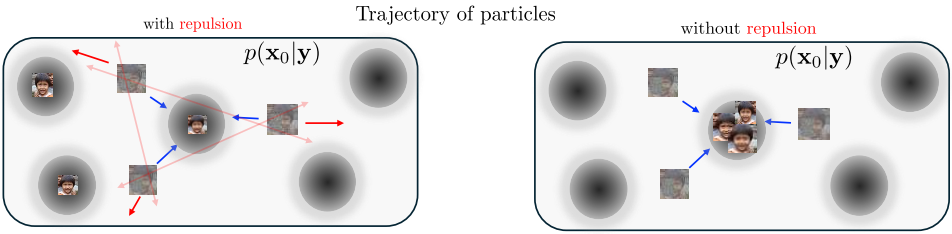
- One denoiser call per step (vs. backpropagating through the score network)
- Challenges:
  - $\Rightarrow$ Unimodal q + reverse KL $\rightarrow$ mode collapse
  - $\Rightarrow$ MAP + unimodal q + autoencoder compression $\rightarrow$ over-smoothed reconstruction

As we will show later, high-probability solutions tend to be over-smooth

- Mode collapse: lift μ to N particles $\{\mathbf{z}_0^{(i)}\}_{i=1}^N$ and couple them via a joint q

$$q_\tau(\mathbf{z}_{t,\tau}^{(1)}, \cdots, \mathbf{z}_{t,\tau}^{(N)}) = \frac{1}{Z}\mathcal{R}(\mathbf{z}_{t,\tau}^{(1)}, \cdots, \mathbf{z}_{t,\tau}^{(N)}) \prod_{i=1}^{N} q_\tau(\mathbf{z}_{t,\tau}^{(i)})$$

  $\Rightarrow \nabla\mathcal{R}$ acts as a repulsive force, exploring different modes



- A lower probability mode might be a better solution for some application

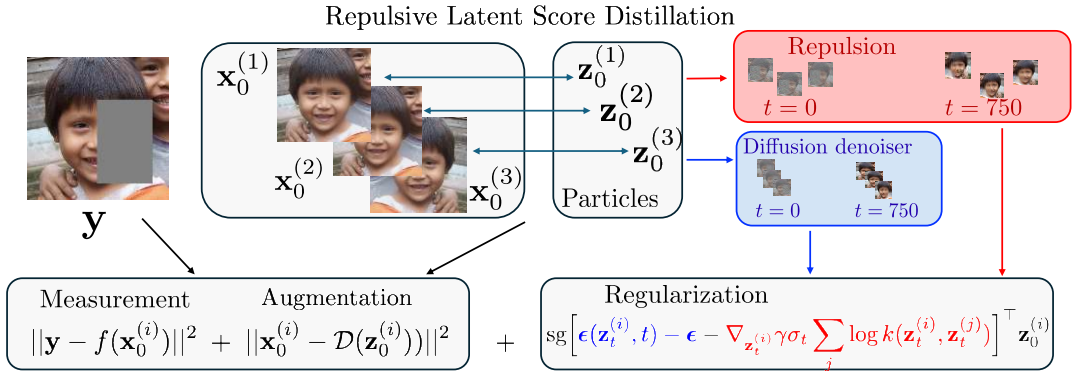
- Over-smoothing: decouple data $\mathbf{x}_0$ from the generative mechanism $\mathbf{z}_0$
- Augmented distribution $q(\mathbf{x}_0, \mathbf{z}_0|\mathbf{y})$: same objective, plus a soft coupling term

$$\min_{q(\mathbf{x}_0,\mathbf{z}_0|\mathbf{y})} \mathbb{E}_q\left[\frac{1}{2\sigma_0^2}\|\mathbf{y}-\mathcal{A}(\mathbf{x}_0)\|^2\right] + \int_0^T \tilde{\lambda}(t)\,\mathbb{E}_\epsilon\left[\|\mathbf{D}_\theta(\mathbf{z}_t,\sigma_t)-\mathbf{z}_0\|_2^2\right]\mathrm{d}t + \mathbb{E}_q\left[\frac{1}{2\rho^2}\|\mathbf{x}_0-\mathcal{D}_\phi(\mathbf{z}_0)\|^2\right]$$

$\Rightarrow$ $\mathbf{x}_0$ gains degrees of freedom outside the decoder range $\rightarrow$ recovers fine detail

$\Rightarrow$ $\rho \to 0$ recovers the hard constraint $\mathbf{x}_0 = \mathcal{D}_\phi(\mathbf{z}_0)$: the latent-only problem

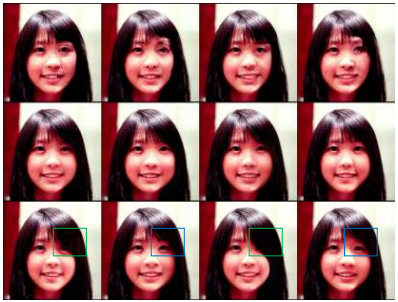
Two knobs: repulsion for diversity, augmentation for detail



Repulsion + augmentation: a **double regularization** of the distillation loss (in ϵ-notation)

- Comparison with baselines using Stable Diffusion 2.1





Box inpainting (half face) with $\sigma_0 = 0.001$ – FFHQ 512.

Sampler	PSNR [dB] ↑	LPIPS ↓	FID ↓	Diversity
PSLD	22.72	0.082	57.7	0.03
Latent RED-diff	23.5	0.15	92.59	0.009
ReSample	19.44	0.2	146.68	-
RED-Diff (Pixel)	23.1	**0.067**	29.79	0.004
DPS (Pixel)	23.4	0.14	78.88	**0.04**
NonAug-RLSD ($\gamma = 50$)	23.34	0.164	98.65	0.035
NonRepuls-RLSD (mean)	24.98	0.079	29.18	0.004
NonRepuls-RLSD (max)	25.82	0.071	29.18	0.004
RLSD ($\gamma = 50$) (mean)	24.69	0.111	31.41	0.015
RLSD ($\gamma = 50$) (max)	**25.84**	0.069	**27.84**	0.015
Hybrid-RLSD	24.72	0.096	30.48	0.018

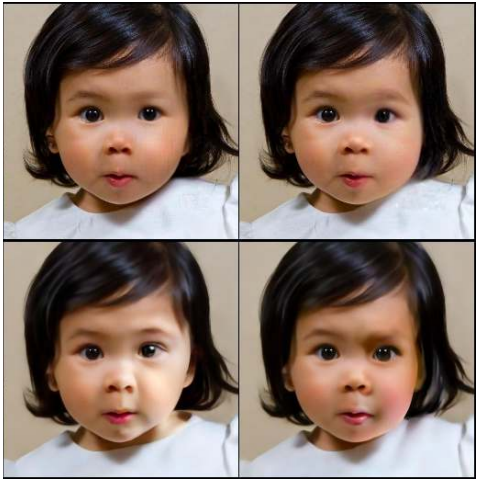
Augmentation removes the over-smoothing



ground truth / measurement

with augmentation

without augmentation



Augmentation restores with sharper details

- Intro to diffusion models
- Inference-time algorithms on pre-trained priors
  - Sampling algorithm
    - Repulsive latent score distillation
- Training the model for improved inference
  - The model itself
    - Energy models with anisotropic noise
    - Flow map denoisers
- Discussion

- We learn the score $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$, never the density itself
  - $\Rightarrow$ Recovering $\log p(\mathbf{x}_0)$ requires integrating the score $\rightarrow$ expensive
  - $\Rightarrow$ The normalization is intractable $\rightarrow$ no likelihoods, no uncertainty, no blind inverse problems



- We learn the score $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$, never the density itself
  - $\Rightarrow$ Recovering $\log p(\mathbf{x}_0)$ requires integrating the score $\rightarrow$ expensive
  - $\Rightarrow$ The normalization is intractable $\rightarrow$ no likelihoods, no uncertainty, no blind inverse problems
- To solve inverse problems we need $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t|\mathbf{y})$, equivalently the denoiser $\mathbb{E}[\mathbf{x}_0|\mathbf{x}_t, \mathbf{y}]$
  - $\Rightarrow$ Existing approaches sit at two extremes

1. Learn a prior, then Bayes (previous slides)

$$p(\mathbf{x}_0|\mathbf{y}) \propto p_{\mathbf{x}}(\mathbf{x}_0)\, p(\mathbf{y}|\mathbf{x}_0)$$

$\Rightarrow$ Map noise to sample $\mathbf{x}_0 \sim p(\mathbf{x}_0|\mathbf{y})$

$\Rightarrow$ Only clean images $\mathbf{x}_0$ (unsupervised)

$\Rightarrow$ Likelihood needs approximations

(Kadkhodaie et al., 2021; Kawar et al., 2022; Chung et al., 2023; etc)

2. Learn a data-dependent diffusion (informed source)

$$\mathcal{N}(f(\mathbf{y}), \mathbf{\Sigma}_h) \longrightarrow \mathbf{x}_0$$

$f(\mathbf{y}) = \mathbf{y}$ or $\mathbb{E}[\mathbf{x}_0|\mathbf{y}]$, $\mathbf{\Sigma}_h$ a hyperparameter

$\Rightarrow$ Map $f(\mathbf{y})$ to sample $\mathbf{x}_0 \sim p(\mathbf{x}_0|\mathbf{y})$

$\Rightarrow$ No approximation of the likelihood

$\Rightarrow$ Supervised training on paired data $(\mathbf{x}_0, \mathbf{y})$

$\Rightarrow$ One model per $\mathbf{H}$

(Saharia et al., 2022; Delbracio and Milanfar, 2023; Negrel et al., 2025)

Can one model cover general $\mathbf{H}$ *without approximations, and give a normalized density?*

- Previous works use isotropic Gaussian denoisers as priors (denoising operation)

$$\mathbf{x}_t = \mathbf{x}_0 + \sigma_t \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

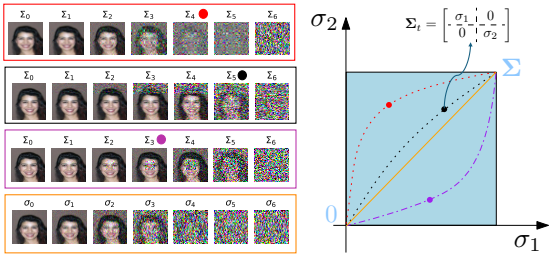
- Previous works use isotropic Gaussian denoisers as priors (denoising operation)

$$\mathbf{x}_t = \mathbf{x}_0 + \sigma_t \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

- Incorporate the degradation in the forward-backward process [Daras 2023, Negrel 2025]

$$\mathbf{y} = \mathbf{H}\mathbf{x}_0 + \sigma\mathbf{v} \iff \tilde{\mathbf{y}} = \mathbf{x}_0 + \tilde{\mathbf{v}}, \quad \tilde{\mathbf{v}} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}), \quad \boldsymbol{\Sigma} = \sigma^2 \mathbf{H}^{-1}\mathbf{H}^{-\top}$$

$\Rightarrow$ Solving inverse problems = removing anisotropic noise indexed by a covariance $\boldsymbol{\Sigma}$



Different denoising paths

- Proposal: Learn a covariance-dependent (H-dep.) energy $U_\theta(\mathbf{y}, \mathbf{\Sigma}) \approx U(\mathbf{y}, \mathbf{\Sigma}) = -\log p(\mathbf{y}|\mathbf{\Sigma})$
  $\Rightarrow \nabla_\mathbf{y} U_\theta(\mathbf{y}, \mathbf{\Sigma}) = -\nabla_\mathbf{y} \log p(\mathbf{y}|\mathbf{\Sigma}) \rightarrow$ score network
- Compute an anisotropic version of denoising score matching:

Anisotropic Denoising Score Matching (A-DSM) Loss

$$\ell_{\mathrm{A-DSM}}(\theta) = \mathbb{E}_{\mathbf{x}_0, \mathbf{y}, \mathbf{\Sigma}} \left[\left\| \mathbf{\Sigma}^{\frac{1}{2}} \nabla_\mathbf{y} U_\theta(\mathbf{y}, \mathbf{\Sigma}) - \mathbf{\Sigma}^{-\frac{1}{2}} (\mathbf{y} - \mathbf{x}_0) \right\|^2 \right] \tag{1}$$

$\Rightarrow$ The reweighting by $\mathbf{\Sigma}^{\frac{1}{2}}$ makes the loss scale-invariant

- Are we learning a *normalized* density?



- Are we learning a *normalized* density?
- **No**. The reverse diffusion is a stochastic process with marginals $p(\mathbf{y}|\boldsymbol{\Sigma})$

  $\Rightarrow$ Time evolution described by the Fokker–Planck equation $\rightarrow$ mass-preserving equation

  $\Rightarrow$ However, score matching learns (implicitly) marginals that are not mass preserving [Choi 2022, Guth 2025]

- Are we learning a *normalized* density?
- **No**. The reverse diffusion is a stochastic process with marginals $p(\mathbf{y}|\boldsymbol{\Sigma})$
  - $\Rightarrow$ Time evolution described by the Fokker–Planck equation $\rightarrow$ mass-preserving equation
  - $\Rightarrow$ However, score matching learns (implicitly) marginals that are not mass preserving [Choi 2022, Guth 2025]
- We incorporate a regularization term: match the covariance score $\nabla_{\boldsymbol{\Sigma}} U_\theta(\mathbf{y}, \boldsymbol{\Sigma})$

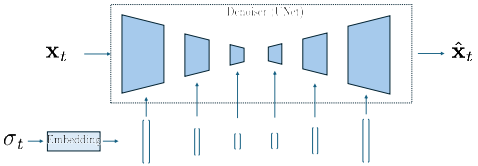
Anisotropic Covariance Score Matching (A-CSM) Loss

$$\ell_{\mathrm{A-CSM}}(\theta) = \mathbb{E}_{\mathbf{x}_0,\mathbf{y},\boldsymbol{\Sigma}}\left[\left\|\boldsymbol{\Sigma}^{\frac{1}{2}}\nabla_{\boldsymbol{\Sigma}} U_\theta(\mathbf{y},\boldsymbol{\Sigma})\boldsymbol{\Sigma}^{\frac{1}{2}} - \frac{1}{2}\mathbf{I} + \frac{1}{2}\boldsymbol{\Sigma}^{-\frac{1}{2}}(\mathbf{y}-\mathbf{x}_0)(\mathbf{y}-\mathbf{x}_0)^\top\boldsymbol{\Sigma}^{-\frac{1}{2}}\right\|_F^2\right] \tag{2}$$

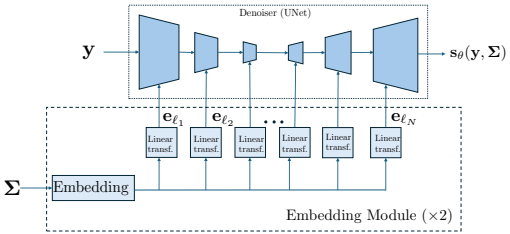
- A-CSM forces the model to learn normalized marginals [Guth 2025] $\rightarrow$ possible by learning the energy
  - $\Rightarrow$ Proxy for the continuity equation
  - $\Rightarrow$ Total objective $\frac{1}{d}\ell_{\mathrm{A-DSM}} + \frac{1}{d^2}\ell_{\mathrm{A-CSM}}$

- $\boldsymbol{\Sigma}$ is a matrix, not a scalar $\rightarrow$ a scalar noise embedding is no longer enough

Standard: scalar level σ_t



Ours: position-dependent $\boldsymbol{\Sigma}$



- Energy parameterized from the score network: $U_\theta(\mathbf{y}, \boldsymbol{\Sigma}) = \frac{1}{2}\langle \mathbf{y}, \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{y}, \boldsymbol{\Sigma})\rangle$
- In practice, $\boldsymbol{\Sigma}$ is diagonal in a fixed basis (pixel, frequency)

- The learned energy matches the true one up to a constant c

$$U_\theta(\mathbf{y}, \mathbf{\Sigma}) = -\log p(\mathbf{y} \mid \mathbf{\Sigma}) + c$$

- A-CSM enforces mass conservation across covariances $\rightarrow$ c does not depend on $\mathbf{\Sigma}$

- The learned energy matches the true one up to a constant c

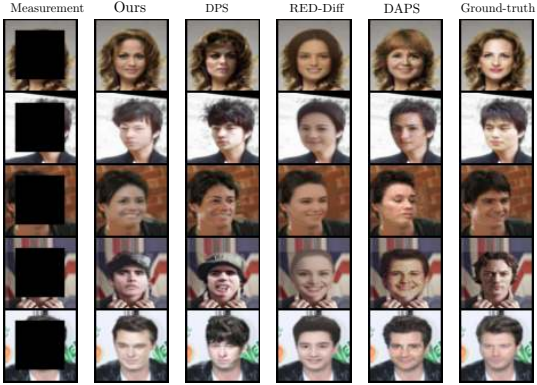
$$U_\theta(\mathbf{y}, \mathbf{\Sigma}) = -\log p(\mathbf{y} \mid \mathbf{\Sigma}) + c$$

- A-CSM enforces mass conservation across covariances $\rightarrow$ c does not depend on $\mathbf{\Sigma}$
  - $\Rightarrow$ For large $\mathbf{\Sigma}$, the marginal is Gaussian: $p(\mathbf{y} \mid \mathbf{\Sigma}) \approx \mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$
  - $\Rightarrow$ Allows to normalize the model for *every* $\mathbf{\Sigma}$

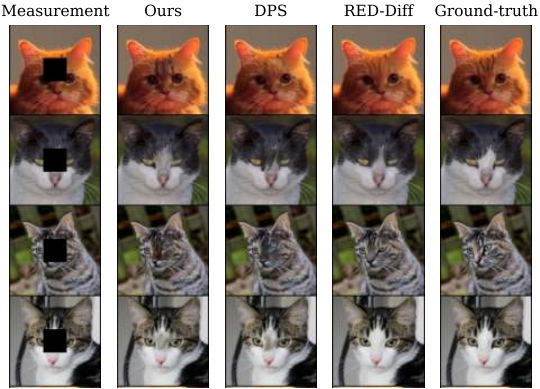
$$U_\theta(\mathbf{y}, \mathbf{\Sigma}) \longleftarrow U_\theta(\mathbf{y}, \mathbf{\Sigma}) - \mathbb{E}_{\mathbf{y}}[U_\theta(\mathbf{y}, \mathbf{\Sigma}) \mid \mathbf{\Sigma}] + \tfrac{1}{2} \log\det(2\pi \mathrm{e}\, \mathbf{\Sigma})$$

- We can now evaluate likelihoods, compare solvers, and solve blind problems w/a single NFE

- We draw samples and compare with well-established solvers

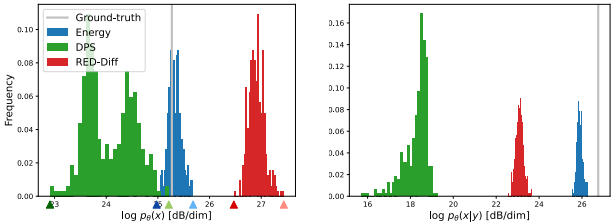


CelebA 64 × 64



AFHQ 192 × 192

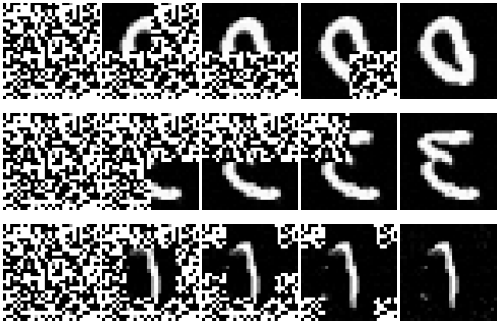
- The normalized density lets us score the samples of any solver
  - $\Rightarrow$ Prior and posterior probabilities for DPS, RED-Diff and ours





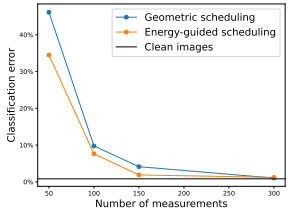
- Ours balances ground-truth prior probabilities with near ground-truth posterior probabilities

- The path in $\boldsymbol{\Sigma}$ is a free choice $\rightarrow$ let the energy pick it
  - $\Rightarrow$ Only possible because the density is normalized (c does not depend on $\boldsymbol{\Sigma}$)



Same model and noise, three unmasking orders

$$\mathbf{x}_{t-1} = \mathbf{x}_t - \boldsymbol{\delta\Sigma}_t \nabla_{\mathbf{y}} U_\theta(\mathbf{x}_t, \boldsymbol{\Sigma}_t) + (\boldsymbol{\delta\Sigma}_t)^{\frac{1}{2}} \mathbf{v}_t$$
$$\boldsymbol{\Sigma}_{t-1} = \boldsymbol{\Sigma}_t - \boldsymbol{\delta\Sigma}_t$$
$$\boldsymbol{\delta\Sigma}_t \propto \boldsymbol{\Sigma}_t \nabla_{\boldsymbol{\Sigma}} U_\theta(\mathbf{x}_t, \boldsymbol{\Sigma}_t) \boldsymbol{\Sigma}_t$$

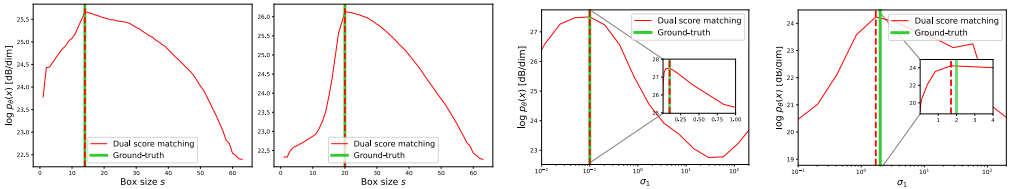


MNIST from k random pixels

No schedule to hand-pick, and it beats a geometric one when measurements are scarce

- The degradation is unknown: estimate $\hat{\boldsymbol{\Sigma}} = \operatorname{argmax}_{\boldsymbol{\Sigma} \in \mathcal{S}} \log p_\theta(\mathbf{y} \mid \boldsymbol{\Sigma})$ over a candidate set $\mathcal{S}$
  - $\Rightarrow$ Only possible because the density is normalized across $\boldsymbol{\Sigma}$





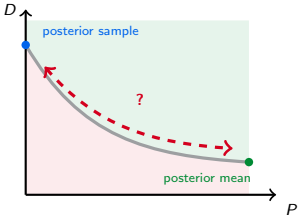
Two observations with $(\sigma_1, s) \in \{(0.1, 14), (2, 20)\}$; $\log p_\theta(\mathbf{y}|\Sigma_s)$ against box size s (left) and noise level σ_1 (right). Estimated (red, dashed) tracks the ground truth (green, solid)

- Intro to diffusion models
- Inference-time algorithms on pre-trained priors
  - Sampling algorithm { • Repulsive latent score distillation
- Training the model for improved inference
  - The model itself { • Energy models with anisotropic noise; • Flow map denoisers
- Discussion

- Every method so far sits at a single operating point of the DP plane

  $\Rightarrow$ Guided samplers are, by construction, posterior samplers ($P = 0$)

  $\Rightarrow$ The energy model gives the posterior mean $\mathbb{E}[\mathbf{x}_0|\mathbf{y}]$ and a posterior sample, but limited to linear $\mathbf{H}$
- Traversing the plane means *two* models, retraining per operating point, or at least two network evaluations



The optimal estimator at perception level P is an interpolation of the two ends [Freirich et al., 2021]

$$\hat{\mathbf{x}}_P = \left(1 - \frac{P}{P^*}\right)\hat{\mathbf{x}}_0 + \frac{P}{P^*}\,\mathbf{x}^\star$$

- $\hat{\mathbf{x}}_0$ perfect perceptual estimator
- $\mathbf{x}^\star$ posterior mean, P^* its perceptual gap

Can a single model traverse the entire DP plane?

Convention change: from here on $t = 0$ is noise and $t = 1$ is data
(the diffusion part of the talk used $\mathbf{x}_0$ for the clean signal)

- A family of paths interpolating noise $\mathbf{x}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and data $\mathbf{x}_1$

$$\mathbf{x}_t = (1 - t)\mathbf{x}_0 + t\,\mathbf{x}_1$$

  $\Rightarrow$ Equivalent to diffusion with a linear schedule, with $\mathbf{x}_1 \sim p_{\text{data}}(\mathbf{x}_1)$
- Induces the velocity field $\mathbf{v}(\mathbf{x}, t) = \mathbb{E}[\dot{\mathbf{x}}_t \mid \mathbf{x}_t = \mathbf{x}]$ and the ODE $\dot{\mathbf{x}}_t = \mathbf{v}(\mathbf{x}_t, t)$
  $\Rightarrow$ Related to the denoiser by Tweedie: $\mathbf{v}(\mathbf{x}, t) = \frac{1}{1-t}\left(\mathbb{E}[\mathbf{x}_1 \mid \mathbf{x}_t = \mathbf{x}] - \mathbf{x}\right)$
  $\Rightarrow$ Learned by regression (conditional flow matching loss)
- Sampling still requires numerical integration: many function evaluations

- Learn the ODE solution operator

$$X_{s,t}(\mathbf{x}_s) = \mathbf{x}_t = \mathbf{x}_s + (t-s)\,\mathbf{v}(\mathbf{x}_s, s, t)$$

$\Rightarrow \mathbf{v}(\mathbf{x}_s, s, t) = \frac{1}{t-s}\int_s^t \mathbf{v}(\mathbf{x}_\tau, \tau)\,\mathrm{d}\tau$: the average velocity over $[s, t]$

$\Rightarrow$ One evaluation jumps from s to t

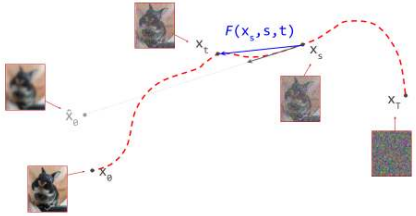


Image from sander.ai/2026/05/06/flow-maps.html

(drawn in the diffusion convention: data at x_0)

- Learn the ODE solution operator

$$X_{s,t}(\mathbf{x}_s) = \mathbf{x}_t = \mathbf{x}_s + (t-s)\,\mathbf{v}(\mathbf{x}_s, s, t)$$

  $\Rightarrow \mathbf{v}(\mathbf{x}_s, s, t) = \frac{1}{t-s}\int_s^t \mathbf{v}(\mathbf{x}_\tau, \tau)\,\mathrm{d}\tau$: the average velocity over $[s, t]$

  $\Rightarrow$ One evaluation jumps from s to t

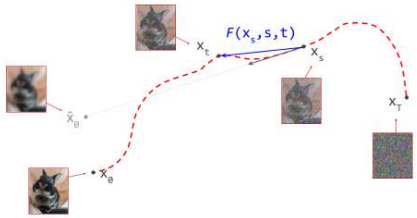


Image from sander.ai/2026/05/06/flow-maps.html

(drawn in the diffusion convention: data at x_0)

- Several losses exist (MeanFlow, Eulerian, semigroup); we use Lagrangian self-distillation

$$\mathcal{L}_{\mathrm{LSD}}(\hat{\mathbf{v}}) = \int_0^1 \int_0^t \mathbb{E}\left[\left\| \partial_t \hat{X}_{s,t}(\mathbf{x}_s) - \mathbf{v}_{\boldsymbol{\theta}}\left(\hat{X}_{s,t}(\mathbf{x}_s), t\right)\right\|^2\right] \mathrm{d}s\,\mathrm{d}t$$

  $\Rightarrow$ The instantaneous change of the flow map is the velocity

- Stop-gradient on the second term; total loss $\mathcal{L} = \mathcal{L}_{\mathrm{CFM}} + \mathcal{L}_{\mathrm{LSD}}$

  $\Rightarrow$ $\partial_t \hat{X}_{s,t}$ costs one extra forward-mode JVP per step

- Learn the ODE solution operator

$$X_{s,t}(\mathbf{x}_s) = \mathbf{x}_t = \mathbf{x}_s + (t-s)\,\mathbf{v}(\mathbf{x}_s, s, t)$$

  $\Rightarrow \mathbf{v}(\mathbf{x}_s, s, t) = \frac{1}{t-s}\int_s^t \mathbf{v}(\mathbf{x}_\tau, \tau)\,\mathrm{d}\tau$: the average velocity over $[s, t]$

  $\Rightarrow$ One evaluation jumps from s to t

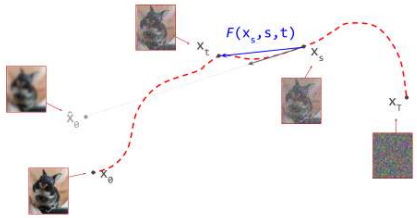


Image from sander.ai/2026/05/06/flow-maps.html

(drawn in the diffusion convention: data at x_0)

- Several losses exist (MeanFlow, Eulerian, semigroup); we use Lagrangian self-distillation

$$\mathcal{L}_{\mathrm{LSD}}(\hat{\mathbf{v}}) = \int_0^1 \int_0^t \mathbb{E}\left[\left\| \partial_t \hat{X}_{s,t}(\mathbf{x}_s) - \mathbf{v}_{\boldsymbol{\theta}}\left(\hat{X}_{s,t}(\mathbf{x}_s), t\right)\right\|^2\right] \mathrm{d}s\,\mathrm{d}t$$

  $\Rightarrow$ The instantaneous change of the flow map is the velocity

- Stop-gradient on the second term; total loss $\mathcal{L} = \mathcal{L}_{\mathrm{CFM}} + \mathcal{L}_{\mathrm{LSD}}$

  $\Rightarrow$ $\partial_t \hat{X}_{s,t}$ costs one extra forward-mode JVP per step

- These models are used for few-step sampling, but can we use them to **traverse the DP plane**?

- **Two-time (average) denoiser**: the denoiser associated with the flow map

$$D_{s,t}(\mathbf{x}) := \mathbf{x} + (1-s)\,\mathbf{v}(\mathbf{x}, s, t)$$

- Properties

  $\Rightarrow$ Diagonal consistency: at $t = s$ it recovers the classic instantaneous denoiser

$$D_{s,s}(\mathbf{x}) = \mathbf{x} + (1-s)\,\mathbf{v}(\mathbf{x}, s, s) = \mathbb{E}\left[\mathbf{x}_1 \mid \mathbf{x}_s = \mathbf{x}\right]$$

  $\Rightarrow$ Perceptual endpoint: as $t \to 1$, it behaves like a sampler

- **Two-time (average) denoiser**: the denoiser associated with the flow map

$$D_{s,t}(\mathbf{x}) := \mathbf{x} + (1-s)\,\mathbf{v}(\mathbf{x}, s, t)$$

- Properties
  - $\Rightarrow$ Diagonal consistency: at $t = s$ it recovers the classic instantaneous denoiser

$$D_{s,s}(\mathbf{x}) = \mathbf{x} + (1-s)\,\mathbf{v}(\mathbf{x}, s, s) = \mathbb{E}\left[\mathbf{x}_1 \mid \mathbf{x}_s = \mathbf{x}\right]$$

  - $\Rightarrow$ Perceptual endpoint: as $t \to 1$, it behaves like a sampler
- **Continuous control**: the lookahead $t \in [s, 1]$ is a knob between the MMSE and the perceptual regimes
  - $\Rightarrow$ A single trained model: both s and t are inputs of $\mathbf{v}_\theta(\mathbf{x}, s, t)$

- For a Gaussian target $x_1 \sim \mathcal{N}(0, \sigma_p^2)$, both estimators are linear in x_s
  - $\Rightarrow$ Optimal DP: $\hat{x}_P = \Gamma(\alpha)\, x_s$, with $\alpha = P/P^*$ the perception level
  - $\Rightarrow$ Two-time denoiser: $D_{s,t}(x_s) = \Lambda(s,t)\, x_s, \quad \Lambda(s,t) = \dfrac{(1-s)\sigma_t - (1-t)\sigma_s}{\sigma_s\,(t-s)}$
- Both are monotone with the same endpoints $\rightarrow$ a bijection $\alpha(s,t)$ between lookahead and perception

$D_{s,t}$ traces the optimal DP curve exactly as t varies in $[s,1]$

- Beyond the Gaussian case there is no closed form, so we check the DP behavior empirically
  - $\Rightarrow$ Flow map trained on CelebA 128×128
  - $\Rightarrow$ Measurements at noise level α: $\mathbf{y} = (1-\alpha)\mathbf{x}_1 + \alpha\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  - $\Rightarrow$ For $\alpha = 0.3$, evaluate $D_{s,t}$ at multiple lookaheads $t \in [1-\alpha, 1]$ ($\mathbf{y} = 0.7\mathbf{x}_1 + 0.3\epsilon$)



The lookahead sharpens the reconstruction, one model throughout

- How to use the flow map denoiser as a plug-and-play prior for general degradations?

  $\Rightarrow$ Mimic a proximal solver

Plug-and-play with flow map denoisers

Require: Flow map $\mathbf{v}_{\boldsymbol{\theta}}(\cdot, s, t)$, observation $\mathbf{y}$, operator $\mathcal{A}$, noise levels $\{s_k\}$, lookahead $t_k \in [s_k, 1]$, steps $\{\lambda_k\}$

1: Initialize $\mathbf{x}^0$
2: **for** $k = 0, \dots, K-1$ **do**
3: $\quad \mathbf{z}^k = \mathbf{x}^k - \lambda_k \nabla g(\mathbf{x}^k)$ ▷ gradient step (data fidelity)
4: $\quad$ Sample $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: $\quad \bar{\mathbf{z}}^k = (1 - s_k)\,\boldsymbol{\epsilon} + s_k\,\mathbf{z}^k$ ▷ renoising: reproject to noise level s_k
6: $\quad \mathbf{x}^{k+1} = \bar{\mathbf{z}}^k + (1 - s_k)\,\mathbf{v}_{\boldsymbol{\theta}}(\bar{\mathbf{z}}^k, s_k, t_k)$ ▷ flow map denoiser D_{s_k, t_k}
7: **end for**
8: **return** $\mathbf{x}^K$

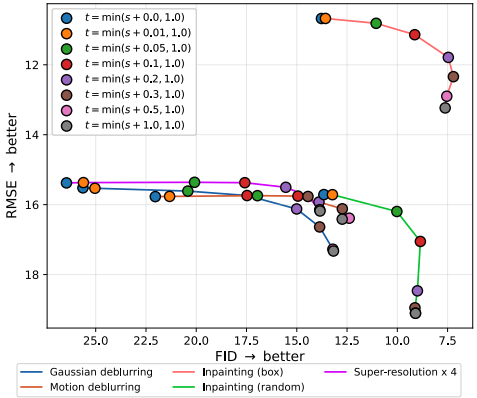
- The renoising step is what makes the lookahead a DP knob

  $\Rightarrow$ It puts the iterate at the noise level s_k the denoiser expects

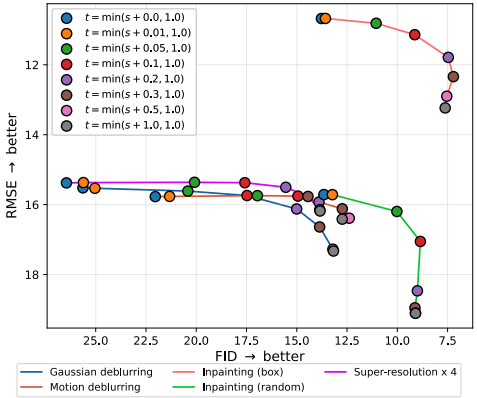
  $\Rightarrow$ It injects the diversity needed at the perceptual end ($t \to 1$)

  $\Rightarrow$ When $s = t$, it reduces to a standard plug-and-play proximal solver

- Inverse problems on AFHQ: inpainting, deblurring, super-resolution
  $\Rightarrow$ Train a flow map from scratch, with a network of $\approx 150M$ parameters

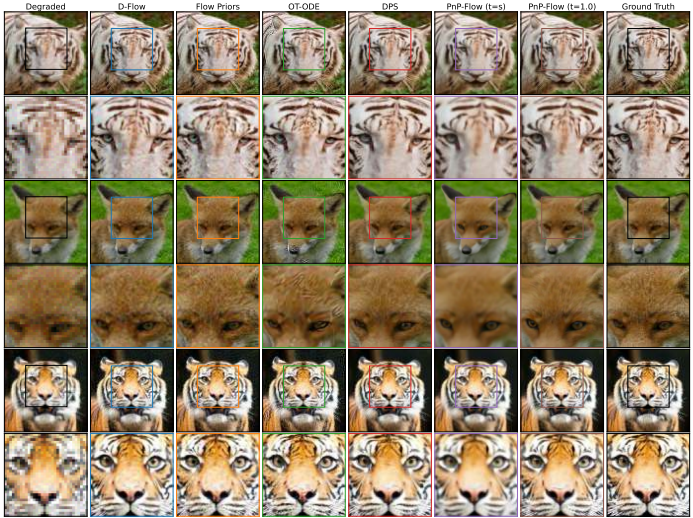


- Inverse problems on AFHQ: inpainting, deblurring, super-resolution
  - $\Rightarrow$ Train a flow map from scratch, with a network of $\approx 150M$ parameters



- Interestingly, the curve is not the same as the convex combination of the two ends
  - $\Rightarrow$ The lookahead is a nonlinear interpolation between the MMSE and the perceptual end

- Baselines are specialized at one end; the lookahead sweeps the whole curve



Diffusion models learn powerful structural priors, with several degrees of freedom to exploit.
This thesis turns that flexibility into solvers that control the properties of the estimate.

Steering a pre-trained prior

- Slow sampling
  → higher-order Langevin
- Mode collapse with latent priors
  → repulsive latent score distillation
- Hard constraints ignored
  → shielded Langevin

Training the model itself

- Task-agnostic source
  → prior-informed flow matching
- Score without density
  → energy models, anisotropic noise
- Instantaneous velocity only
  → flow map denoisers

Looking into the future:

- **Close what is open**: safe sampling with navigation potentials still lacks theoretical guarantees
- **Flow maps** look like the right object for controllable inverse-problem solvers
  - $\Rightarrow$ Characterization of the underlying DP curve
  - $\Rightarrow$ Characterization of the underlying "proximal operator" in the plug-and-play setting
- **Scientific inverse problems**: drug discovery, astronomical imaging, climate modeling

predicting the future is an impossible task, so I will stop here

This work has been published in the following articles



1. Nicolas Zilberstein, Ashutosh Sabharwal, and Santiago Segarra. Solving linear inverse problems using higher-order annealed langevin diffusion.
   IEEE Transactions on Signal Processing, 2024

2. Nicolas Zilberstein, Morteza Mardani, and Santiago Segarra. Repulsive latent score distillation for solving inverse problems.
   Intl. Conf. Learn. Repr. (ICLR), 2025

3. Nicolas Zilberstein, Santiago Segarra, Eero P. Simoncelli, and Florentin Guth. Learning normalized energy models for linear inverse problems.
   Intl. Conf. on Mach. Learn. (ICML), 2026

4. Nicolas Zilberstein, Santiago Segarra, and Luiz Chamon. Sampling with shielded Langevin Monte Carlo using navigation potentials.
   In *Asilomar Conf. on Signals, Systems, and Computers*, 2025

5. Harvey Chen, Nicolas Zilberstein, and Santiago Segarra. Prior-informed flow matching for graph reconstruction.
   ICML workshop SPIGM, 2026

6. Nicolas Zilberstein, Morteza Mardani, Eero P. Simoncelli, and Santiago Segarra. Flow map denoisers: Traversing the distortion–perception plane for inverse problems.
   ICML workshop SPIGM, 2026



- This work was done in collaboration with

Ashutosh Sabharwal Morteza Mardani Florentin Guth

Eero Simoncelli Luiz Chamon Harvey Chen Santiago Segarra

- This work was done in collaboration with

Ashutosh Sabharwal Morteza Mardani Florentin Guth
Eero Simoncelli Luiz Chamon Harvey Chen Santiago Segarra

and many more people than these slides can hold.

Thank you, for everything.

Current Practice

- Negative prompts / filters
- Classifier guidance
- Post-processing operations

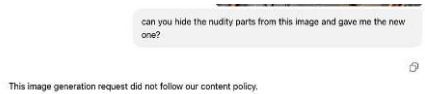
No formal safety guarantees

What we seek

- Enforce **hard constraints** during sampling
- Adaptive steps near forbidden regions
- **Provable avoidance** of unsafe regions

Formal safety guarantees

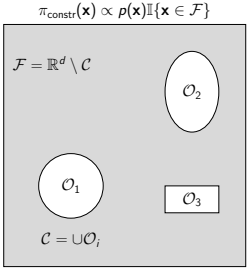


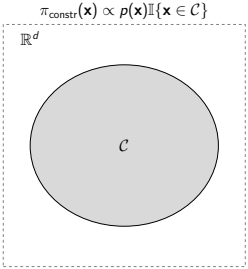


Prompt: *"The artist's sketch captured the model's nudity with bold strokes and dynamic lines, revealing the raw energy of the human form."*

Real systems need (hard) guarantees.

	Hard constraint	**"Soft constraint"**
Convex spaces	mirror/proximal/projected/barriers/primal-dual LMC	penalized LMC
Non-convex spaces	**This proposed algorithm**	—





- Represent obstacle i as a constraint function

$$\beta_i(\mathbf{x}) \leq 0 \iff \mathbf{x} \in \mathcal{O}_i \quad \text{(inside obstacle)}$$

$$\beta_i(\mathbf{x}) > 0 \iff \mathbf{x} \in \text{free space}$$

$\Rightarrow$ Example: ellipsoidal obstacle centered at $\boldsymbol{\mu}_i$

$$\beta_i(\mathbf{x}) = (\mathbf{x} - \boldsymbol{\mu}_i)^\top \boldsymbol{\Sigma}_i (\mathbf{x} - \boldsymbol{\mu}_i) - r^2$$

- Represent obstacle i as a constraint function

$$\beta_i(\mathbf{x}) \leq 0 \iff \mathbf{x} \in \mathcal{O}_i \quad \text{(inside obstacle)}$$

$$\beta_i(\mathbf{x}) > 0 \iff \mathbf{x} \in \text{free space}$$

$\Rightarrow$ Example: ellipsoidal obstacle centered at $\boldsymbol{\mu}_i$

$$\beta_i(\mathbf{x}) = (\mathbf{x} - \boldsymbol{\mu}_i)^\top \boldsymbol{\Sigma}_i (\mathbf{x} - \boldsymbol{\mu}_i) - r^2$$

- Multiple obstacles: combine such that the constraint remains positive when all of them are avoided

$$\beta(\mathbf{x}) = \prod_{i=1}^{n} \beta_i(\mathbf{x})$$

- Original Langevin Monte Carlo[1] is

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \epsilon_k \nabla_{\mathbf{x}_k} \log p(\mathbf{x}_k) + \sqrt{2\tau\epsilon_k}\, \mathbf{w}_k, \qquad \mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

[1] particular instance of reverse process

- Original Langevin Monte Carlo[1] is

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \epsilon_k \nabla_{\mathbf{x}_k} \log p(\mathbf{x}_k) + \sqrt{2\tau\epsilon_k}\, \mathbf{w}_k, \qquad \mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

- Proposal: sample with the following modified dynamics

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \eta_k \left[\beta(\mathbf{x}_k) \nabla_{\mathbf{x}_k} \log p(\mathbf{x}_k) - \frac{\log p(\mathbf{x}_k)\, \nabla_{\mathbf{x}_k} \beta(\mathbf{x}_k)}{\alpha} \right] + \sqrt{2\eta_k \tau \beta^2(\mathbf{x}_k)}\, \mathbf{w}_k$$

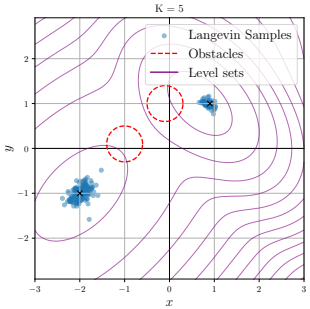
- Inspired by Rimon–Koditschek artificial potential from navigation
  - $\Rightarrow$ Adaptive step-size $\eta_k \beta(\mathbf{x}_k)$ + repulsion effect
  - $\Rightarrow$ Far from obstacles $\rightarrow$ standard Langevin
  - $\Rightarrow$ Near obstacles $\rightarrow$ smaller steps + repulsive drift
- Derivation and analysis via Wasserstein gradient flow in probability space with a *convex* objective

[1] particular instance of reverse process

- Sample from a mixture of 2 Gaussians with 2 obstacles

$$\Rightarrow \boldsymbol{\mu}_1 = \begin{bmatrix} -2 \\ -1 \end{bmatrix}, \boldsymbol{\mu}_2 = \begin{bmatrix} 0.7 \\ 1 \end{bmatrix} \text{ and } \boldsymbol{\Sigma}_1 = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}, \boldsymbol{\Sigma}_2 = \begin{bmatrix} 0.5 & -0.25 \\ -0.25 & 0.5 \end{bmatrix}$$

$$\Rightarrow \beta_1 = \left(\mathbf{x} - \begin{bmatrix} -0.1 \\ 1 \end{bmatrix} \right)^2 - 0.4^2, \beta_2 = \left(\mathbf{x} - \begin{bmatrix} 1 \\ -0.1 \end{bmatrix} \right)^2 - 0.4^2$$

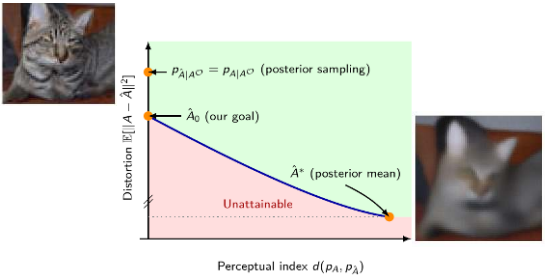


Goal: Recover complete graph $\mathbf{A}$ from partial observations $\mathbf{A}^{\mathcal{O}}$ (equiv. to image inpainting)
Framework: A permutation-equivariant version of the distortion–perception trade-off

$$D(P) = \min_{p(\hat{\mathbf{A}}|\mathbf{A}^{\mathcal{O}})} \left\{ \mathbb{E}\left[\|\mathbf{A} - \hat{\mathbf{A}}\|_F^2 \right] \ : \ d(p_{\mathbf{A}}, p_{\hat{\mathbf{A}}}) \leq P \right\}, \quad d \text{ is some divergence}$$



Blurred observation

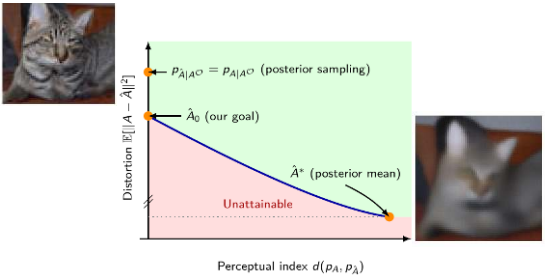


Goal: Recover complete graph $\mathbf{A}$ from partial observations $\mathbf{A}^{\mathcal{O}}$ (equiv. to image inpainting)
Framework: A permutation-equivariant version of the distortion–perception trade-off

$$D(P) = \min_{p(\hat{\mathbf{A}}|\mathbf{A}^{\mathcal{O}})} \left\{ \mathbb{E}\left[\|\mathbf{A} - \hat{\mathbf{A}}\|_F^2 \right] \ : \ d(p_{\mathbf{A}}, p_{\hat{\mathbf{A}}}) \leq P \right\}, \quad d \text{ is some divergence}$$



Blurred observation



At $P = 0$, the optimal solution decomposes into two steps:

- **Step 1 (MMSE)**: Estimate conditional mean via latent variable model
- **Step 2 (OT)**: Learn optimal transport with data-dependent source

- **Assumption 1**: Model edges through latent node representations $\mathbf{z}_1, \ldots, \mathbf{z}_n \in \mathcal{Z}$:

$$A_{ij} \sim \text{Bernoulli}(f(\mathbf{z}_i, \mathbf{z}_j))$$

- **Assumption 2**: Conditional independence assumption: edges depend only on latent structure:

$$P(\mathbf{A}|Z) = \prod_{i<j} P(A_{ij}|\mathbf{z}_i, \mathbf{z}_j) = \prod_{i<j} f(\mathbf{z}_i, \mathbf{z}_j)$$

- **Assumption 1**: Model edges through latent node representations $\mathbf{z}_1, \ldots, \mathbf{z}_n \in \mathcal{Z}$:

$$A_{ij} \sim \text{Bernoulli}(f(\mathbf{z}_i, \mathbf{z}_j))$$

- **Assumption 2**: Conditional independence assumption: edges depend only on latent structure:

$$P(\mathbf{A}|Z) = \prod_{i<j} P(A_{ij}|\mathbf{z}_i, \mathbf{z}_j) = \prod_{i<j} f(\mathbf{z}_i, \mathbf{z}_j)$$

- This model yields the following MMSE estimate:

$$\hat{\mathbf{A}}^\star = \mathbb{E}\left[\mathbf{A}|\mathbf{A}^{\mathcal{O}}\right] = P(\mathbf{A}|Z) \approx f_{\text{prior}}(\mathbf{A}^{\mathcal{O}})$$

where f_{prior} is a neural network that predicts edge probabilities

$\Rightarrow$ Requires an inverse mapping $\mathbf{z}^{-1} : \mathbf{A} \to \mathcal{Z}$

$\Rightarrow$ Any pre-trained link prediction method (node2vec, GraphSAGE, Graphons [Azizpour 2025])

- We consider the following interpolant (rectified flow) between source $\mathbf{A}_0$ and data $\mathbf{A}_1 = \mathbf{A}$

$$\mathbf{A}_t = (1-t)\mathbf{A}_0 + t\mathbf{A}_1, \qquad \mathbf{A}_0 \sim p_0 = \mathcal{N}(\mathbf{0}, \mathbf{I}) \text{ (task-agnostic)}$$

- We consider the following interpolant (rectified flow) between source $\mathbf{A}_0$ and data $\mathbf{A}_1 = \mathbf{A}$

$$\mathbf{A}_t = (1-t)\mathbf{A}_0 + t\mathbf{A}_1, \qquad \mathbf{A}_0 \sim p_0 = \mathcal{N}(\mathbf{0}, \mathbf{I}) \ \text{(task-agnostic)}$$

- **Proposal**: Replace the source with a prior-informed one:

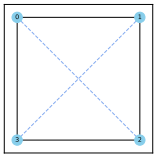
$$\mathbf{A}_0 = \mathbf{A}^{\mathcal{O}} + (1-\xi) \odot \left(f_{\text{prior}}(\mathbf{A}^{\mathcal{O}}) + \boldsymbol{\epsilon}_s\right), \qquad \boldsymbol{\epsilon}_s \sim \mathcal{N}(\mathbf{0}, \sigma_s^2 \mathbf{I})$$

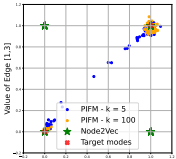
  where ξ is the observation mask and $f_{\text{prior}}(\mathbf{A}^{\mathcal{O}}) \approx \mathbb{E}\left[\mathbf{A} | \mathbf{A}^{\mathcal{O}}\right]$ on the masked pairs
- **Benefits**:
  - $\Rightarrow$ Source distribution conditioned on observed graph $\mathbf{A}^{\mathcal{O}}$
  - $\Rightarrow$ Leverage an initial estimator as prior-informed initialization
  - $\Rightarrow$ Learn global information

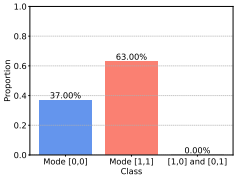
Setup: Four-node graph with constraint: diagonal edges e_{02}, e_{13} must have the same value

$$\mathcal{E} = \{[e_{02} = 1, e_{13} = 1], [e_{02} = 0, e_{13} = 0]\}, \quad P([1,1]) = 0.6,\ P([0,0]) = 0.4$$

- Discretization with Euler method $\rightarrow$ K is the number of steps







Edge-wise Prior (node2vec)

- Predicts each edge independently
- Model is misspecified
- Can generate structurally invalid graphs

Solution: Prior-Informed Flow Model

- Predictions are global
- Learns correct mode proportions
- Generates only valid graphs

Graph reconstruction performance with **50% of edges masked (0.5 Drop)**.

Method	Mask Rate: 50% (0.5 Drop)											
	ENZYMES				PROTEINS				IMDB-B			
	AP ↑	AUC ↑	FNR ↓	FPR ↓	AP ↑	AUC ↑	FNR ↓	FPR ↓	AP ↑	AUC ↑	FNR ↓	FPR ↓
Baselines												
Node2Vec	19.14	55.22	46.37	44.29	23.51	53.83	51.37	44.36	54.20	52.22	48.41	47.51
SIGL	16.88	49.30	72.01	27.85	22.90	52.55	100.00	0.00	50.05	45.41	87.68	18.80
GraphSAGE	22.79	57.77	40.02	52.16	27.71	53.99	32.16	66.86	75.74	75.54	18.18	44.86
DiGress + RePaint	17.34	55.22	77.95	11.62	23.65	55.45	71.46	17.65	56.47	58.89	73.00	10.27
GDSS + RePaint	16.43	49.65	69.45	30.46	22.33	51.42	66.44	32.23	53.39	51.20	69.35	29.22
Flow w/ Gaussian prior	17.43	51.84	98.49	1.07	26.40	55.55	93.21	5.25	78.72	79.76	41.56	14.62
Ours												
PIFM (Node2Vec)	22.95	59.14	90.71	3.53	27.57	59.68	87.05	8.98	84.46	85.71	32.95	15.03
PIFM (GraphSAGE)	25.44	61.36	95.62	1.86	35.50	60.61	85.05	10.23	93.13	93.84	17.52	7.61
PIFM (SIGL)	17.08	49.15	86.06	12.28	28.38	59.58	61.20	20.38	59.83	58.11	38.90	36.76