# Generative Image Restoration Using Diffusion Models:
## A Paradigm Shift from Sparsity to Generative Modeling

**Xiangming Meng**

Assistant Professor
ZJU-UIUC Institute, Zhejiang University

Email： xiangmingmeng@intl.zju.edu.cn

August 19th, 2024
Shenzhen, China

**Collaborator： Yoshiyuki Kabashima, The University of Tokyo, Japan**

# Contents

# Background

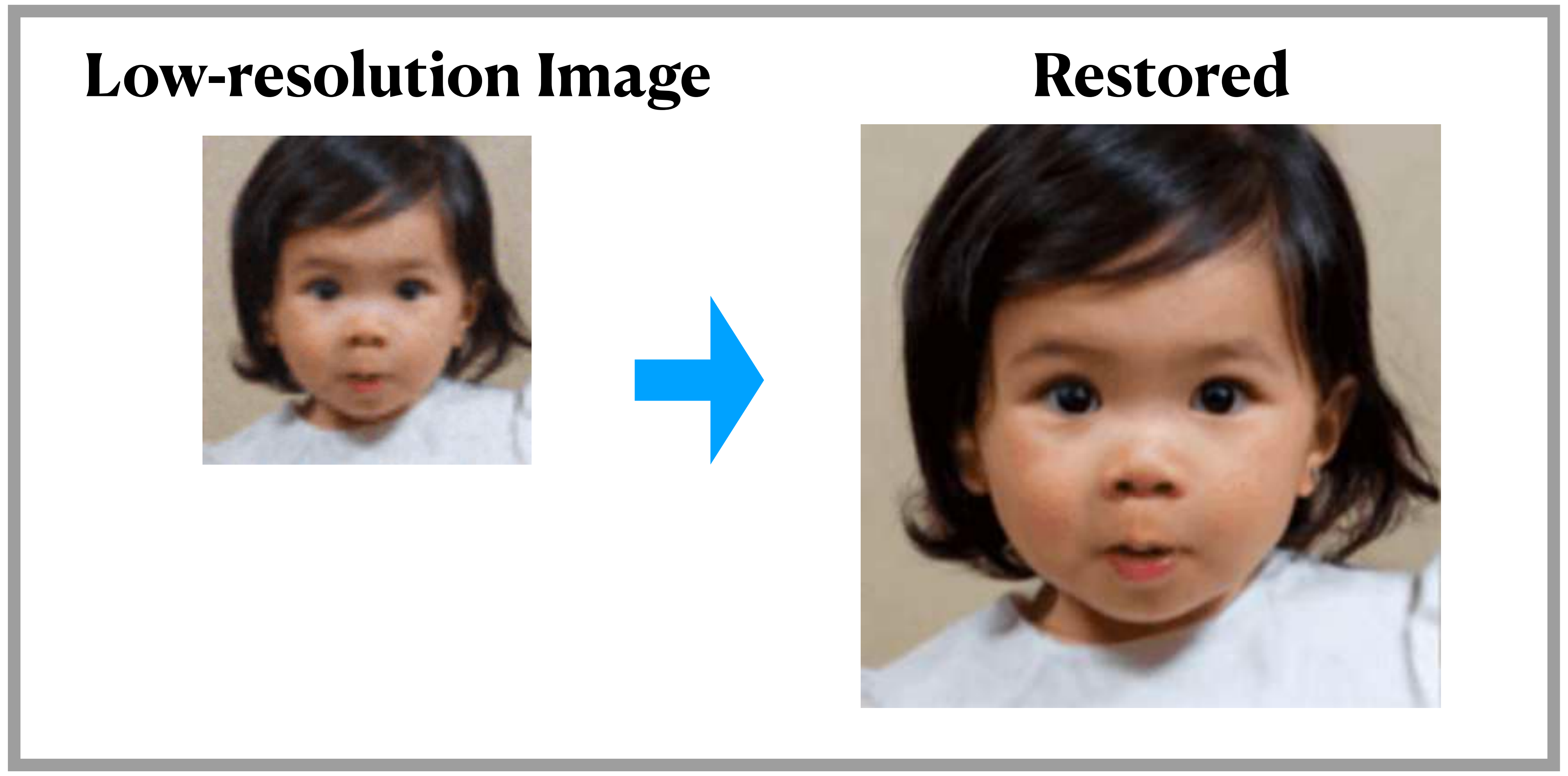
- **Image Restoration**


**Image Restoration**

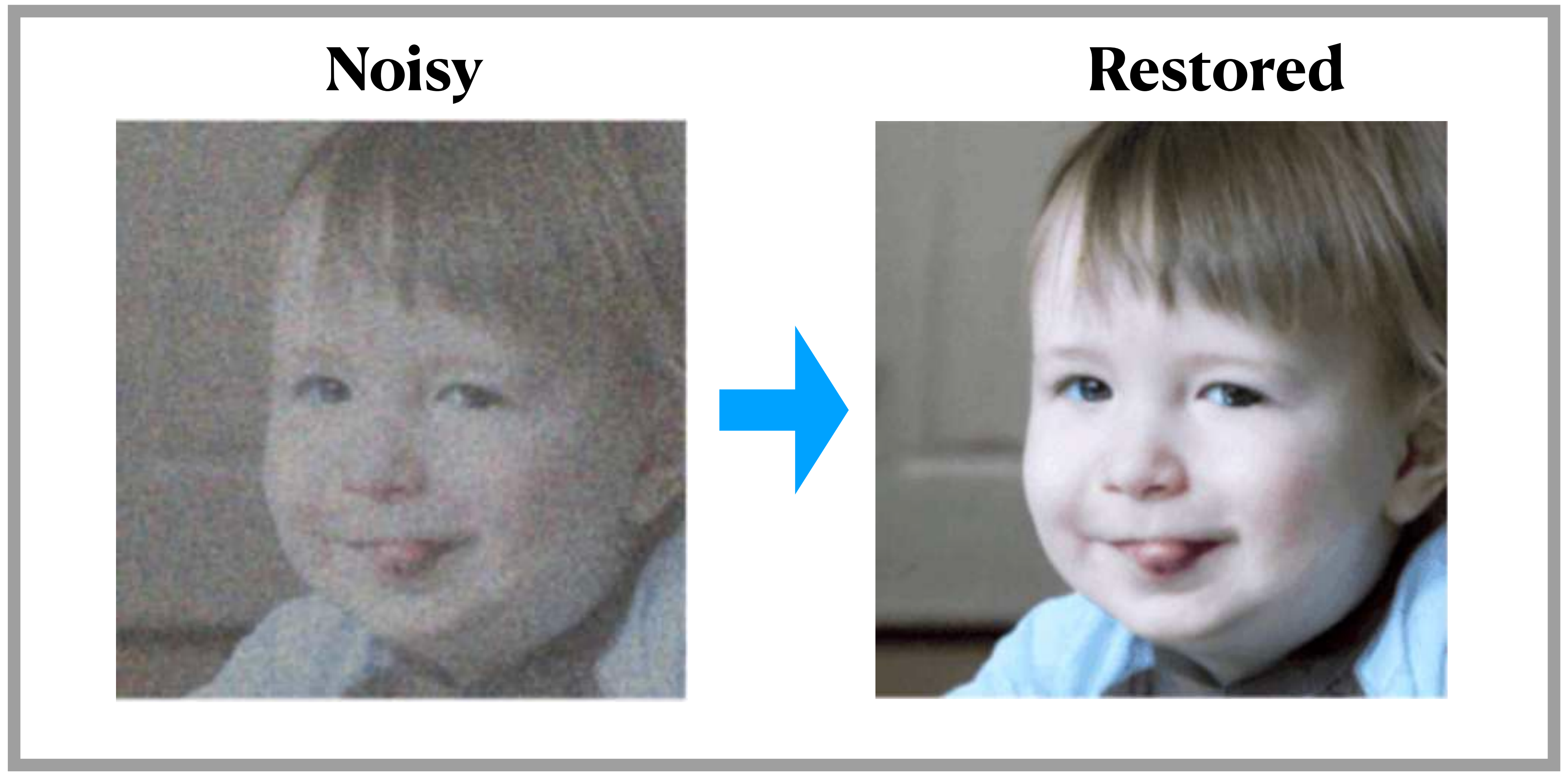
# Background

■ **Image Restoration**

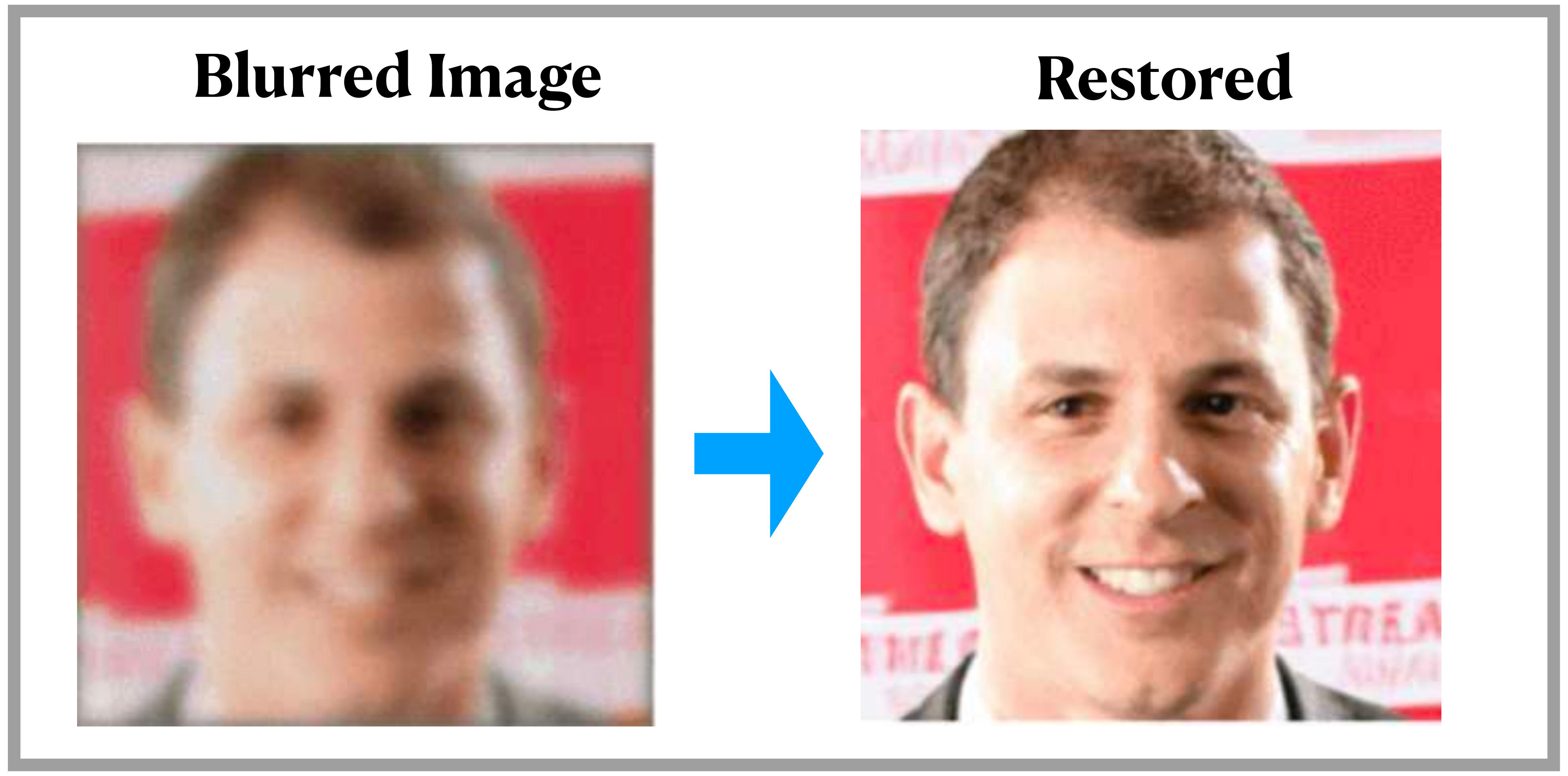
### Super-resolution



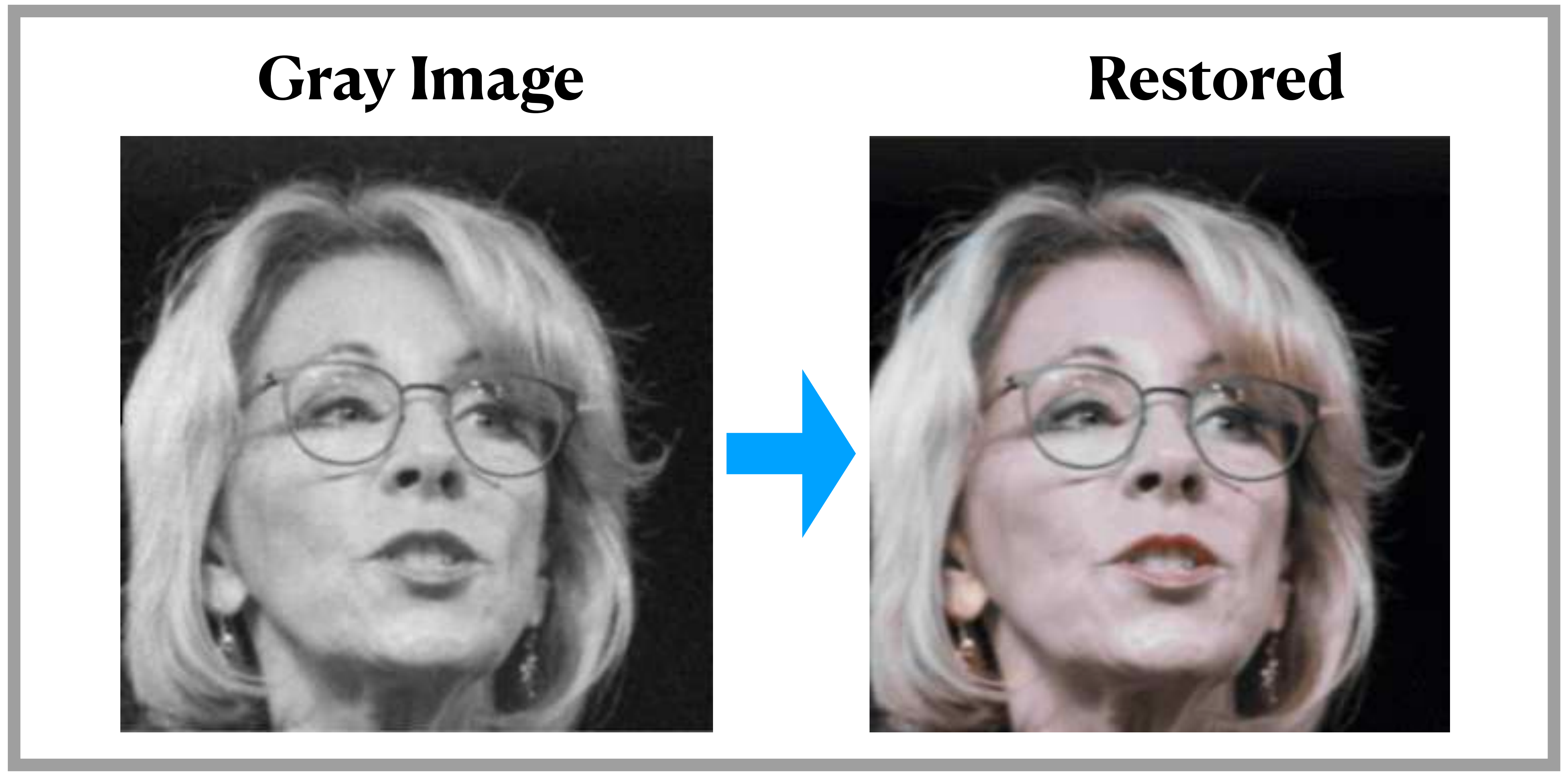

### Denoising




### Deblurring

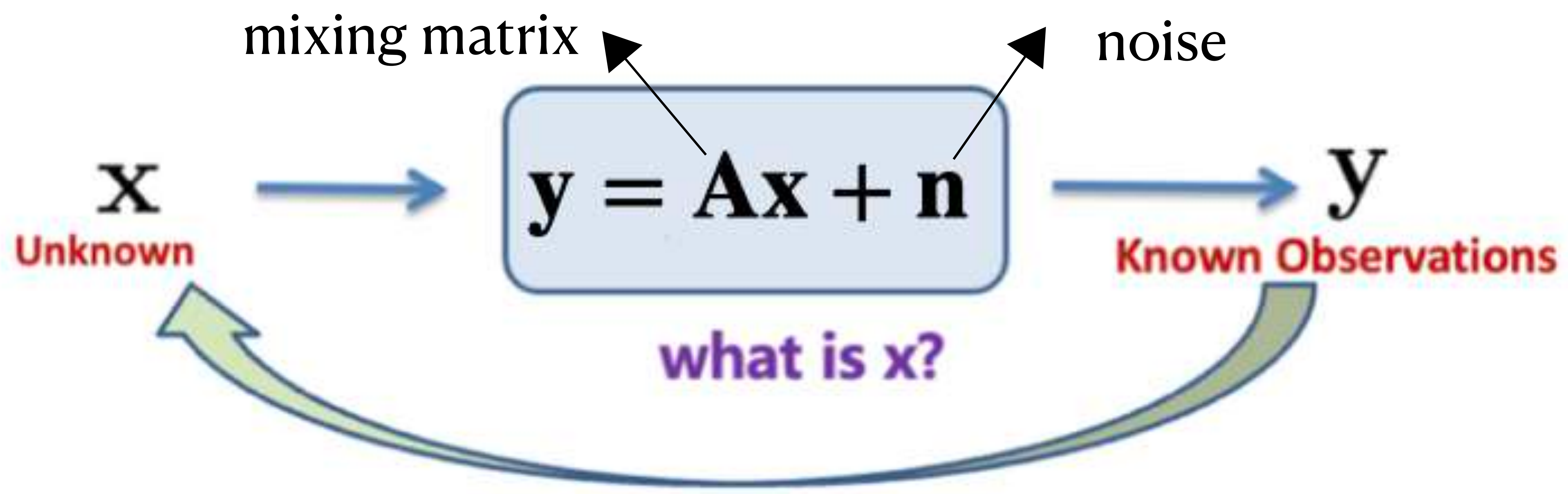



### Colorization



# Background

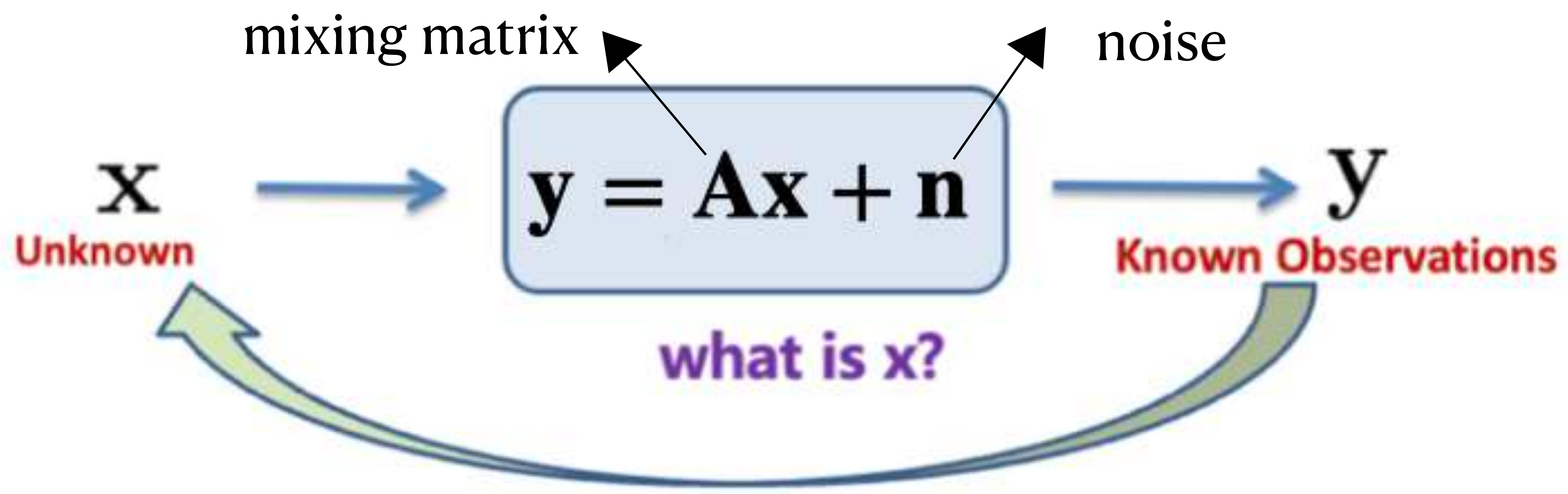
- **Mathematical Formulation**




**Linear Inverse Problems**

# Background

- **Mathematical Formulation**




**Linear Inverse Problems**

- **Examples of A**

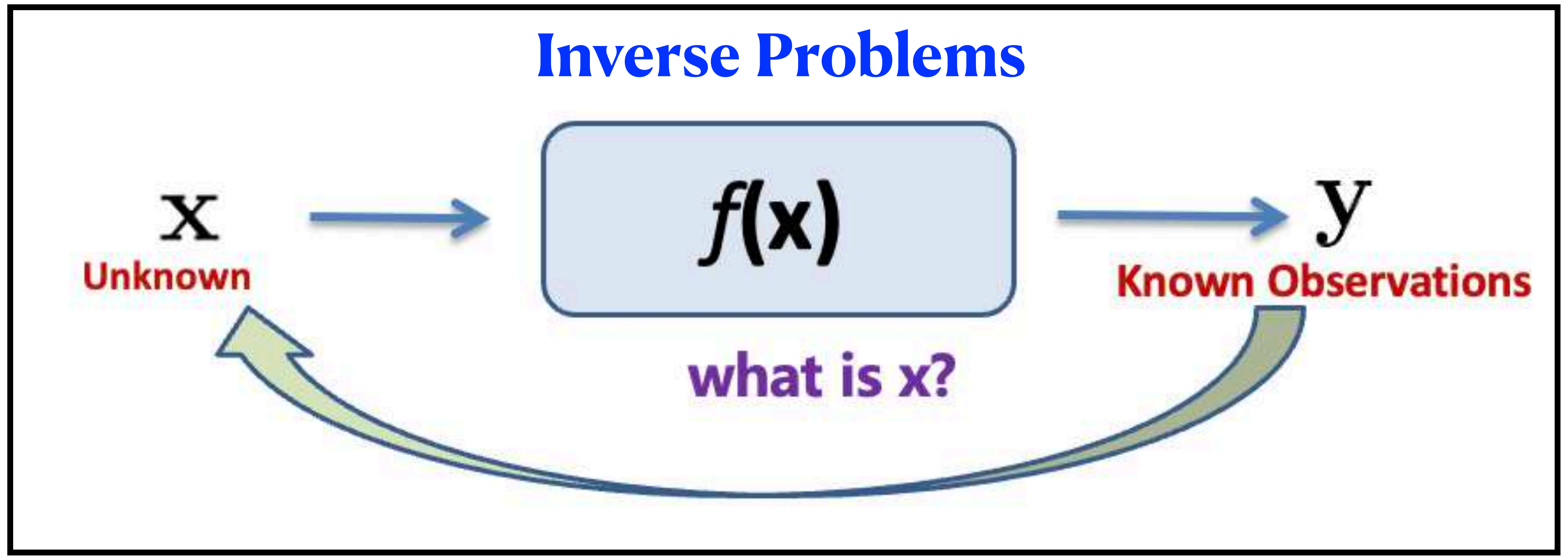
| | | |
|---|---|---|
| Super-resolution | $\mathbf{A} = (\mathbf{I} \otimes \mathbf{k}^T)\,\mathbf{P}$ | $\mathbf{k}$ is a vector of size $r^2$ and $\mathbf{P}$ is a permutation matrix that reorders a vectorized image into patches |
| Denoising | $\mathbf{A} = \mathbf{I}$ | |
| Deblurring | $\mathbf{A} = \mathbf{A}_r \otimes \mathbf{A}_c$ | For a 2D blurring kernel $\mathbf{K} = \mathbf{r}\mathbf{c}^T$, $\mathbf{A}_c$ and $\mathbf{A}_r$ apply a 1D convolution with kernels $\mathbf{c}$ and $\mathbf{r}$, respectively |
| Colorization | $(\mathbf{A}\mathbf{x})_i = \mathbf{k}^T \mathbf{p}_i$ | $\mathbf{k}^T = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ and $\mathbf{p}_i$ is the 3-valued $i$-th pixel of the original color image |

**Fundamental Challenge:**

**Due to incomplete/noisy measurements, the image restoration problem is ill-posed!**

# A Bayesian Perspective

- **Bayesian Perspective for Image Restoration**




**Bayesian Inference**

Posterior　　Prior　　Likelihood

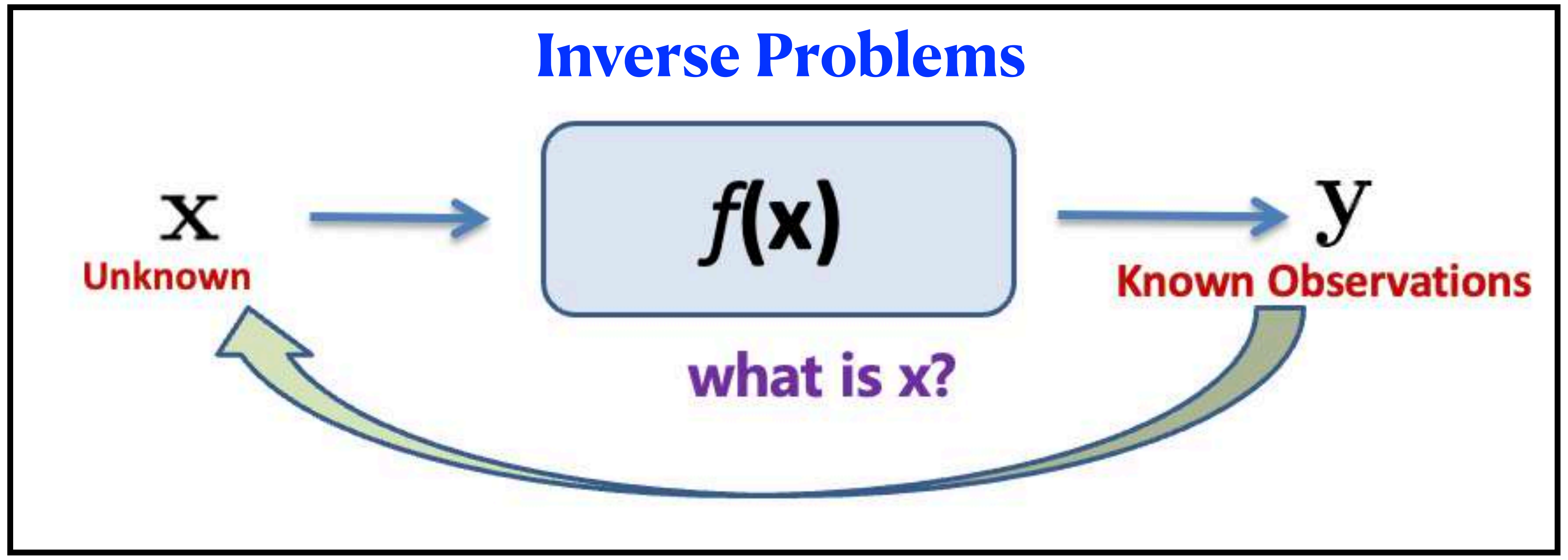
$$p(\mathbf{x} \mid \mathbf{y}) = \frac{p(\mathbf{x})p(\mathbf{y} \mid \mathbf{x})}{p(\mathbf{y})}$$

**Bayes' rule**

Thomas Bayes (1702-1761)

# A Bayesian Perspective

## ■ Bayesian Perspective for Image Restoration




**Bayesian Inference**

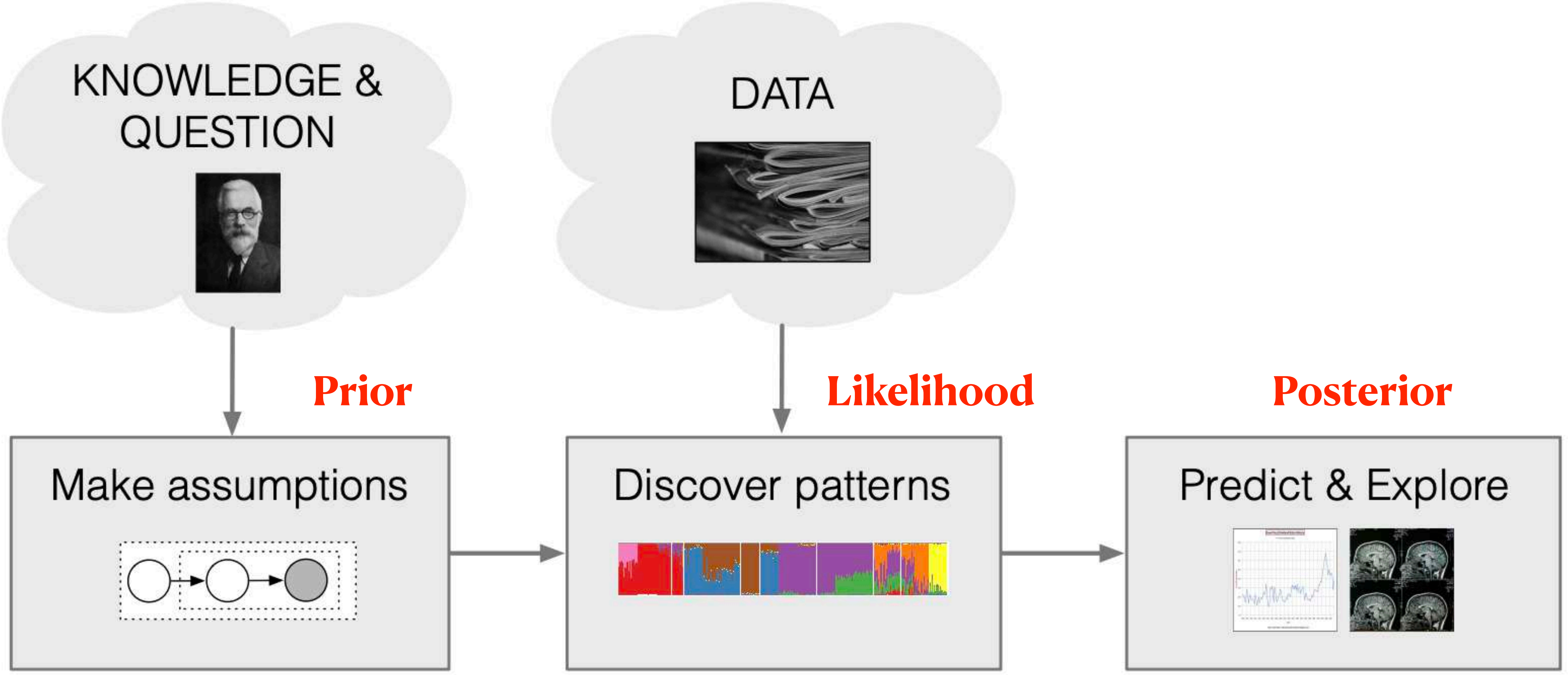
Posterior  Prior  Likelihood

$$p(\mathbf{x}\,|\,\mathbf{y}) = \frac{p(\mathbf{x})p(\mathbf{y}\,|\,\mathbf{x})}{p(\mathbf{y})}$$

**Bayes' rule**

Thomas Bayes (1702-1761)

## ■ Bayesian Learning Framework




**Bayesian Learning Framework**

[David Blei 2016]

# A Bayesian Perspective

- **Key idea**

**The more you know *a priori* the less you need!**

You can easily recognize someone you are familiar with at one single sight

# A Bayesian Perspective

- **Key idea**

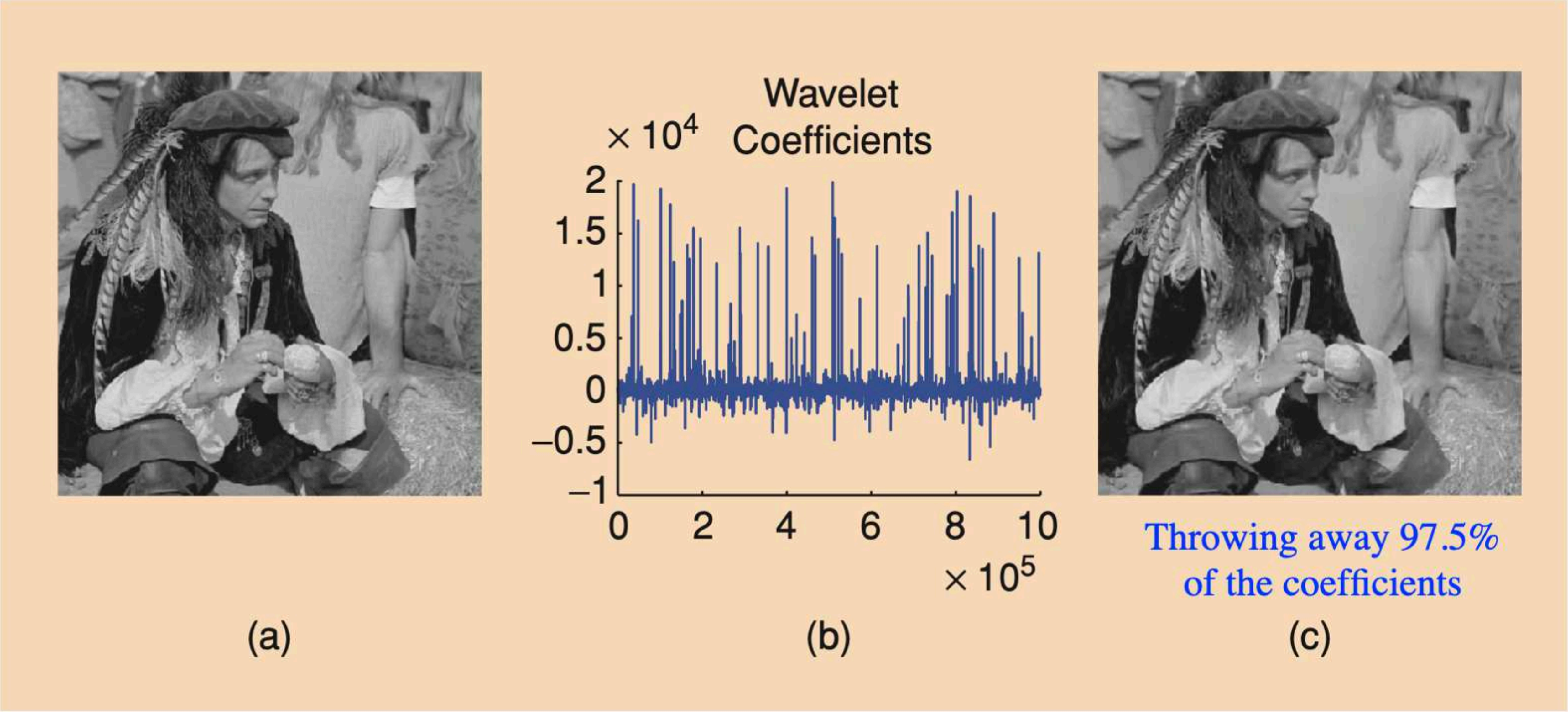
**The more you know *a priori* the less you need!**

You can easily recognize someone you are familiar with at one single sight

**How to obtain good prior knowledge?**

# Classic Approach: Sparsity Modeling

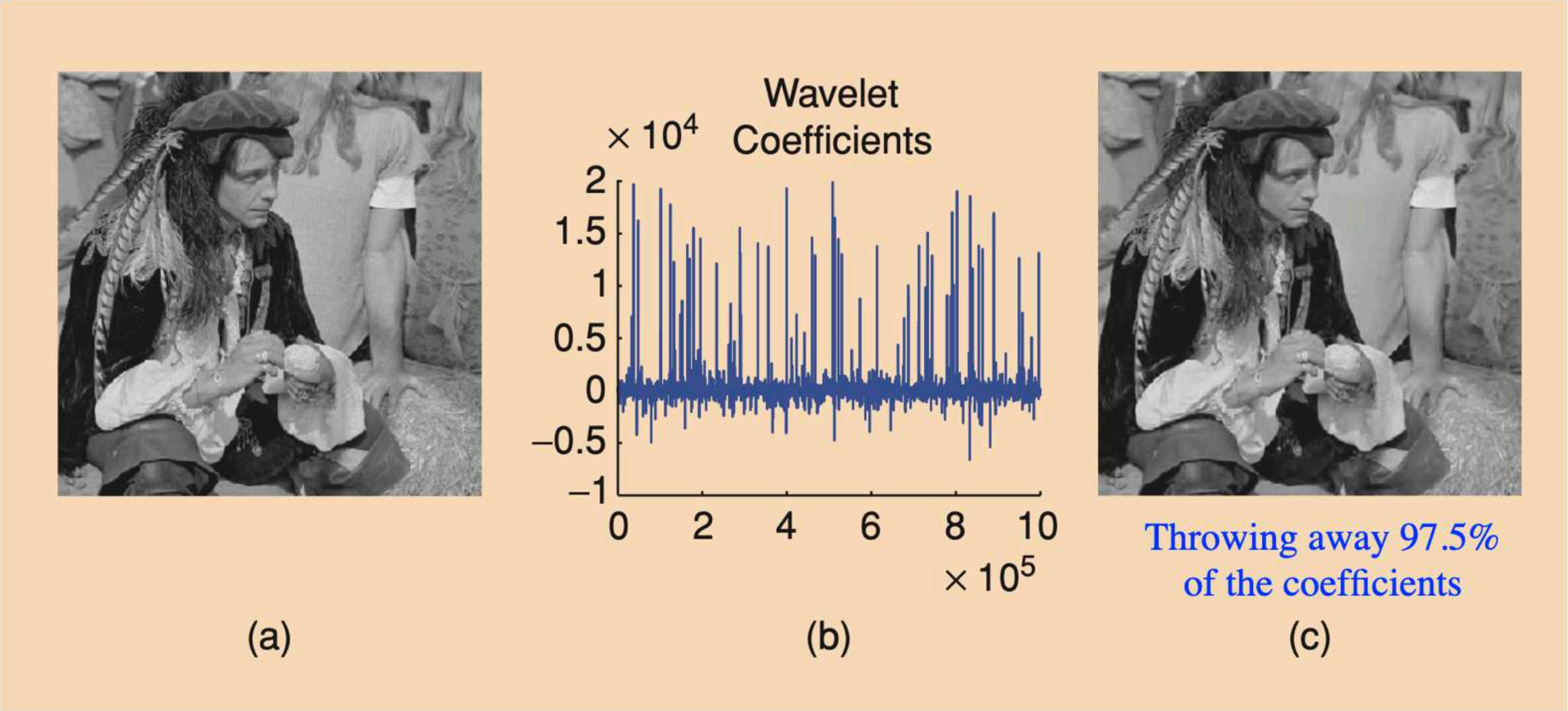
■ **Sparsity Modeling**




• **Sparsity**: The target signal x is ***sparse,*** i.e.***,*** *most elements are zero (*under some transformation)

# Classic Approach: Sparsity Modeling

■ **Sparsity Modeling**




• **Sparsity**: The target signal x is ***sparse,*** i.e.***,*** *most elements are zero* (under some transformation)

• **Compressed Sensing**

Sparse Regularization

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + \lambda r(\mathbf{x})$$

**Commonly used** $r(\mathbf{x})$

| | |
|---|---|
| $L_1$ sparsity (Lasso) | $r(x) = \|x\|_1$ |
| Group Lasso | $r(x) = \sum_g \|x_g\|_2$ |
| Structured Sparsity | Tree-structured/Graph sparsity<br>Total Variation Regularization … |

Sparsity Modeling & Compressed Sensing

1. The standard $L_1$ sparsity is equivalent to Laplace prior distribution.
2. More complicated priors, e.g., group Lasso, structured sparsity, can be used to improve performance.
3. However, such hand-crafted priors might still fail to capture the rich structure in natural signals.

# Classic Approach： Sparsity Modeling

**Is sparse prior good enough?**

**"What I cannot create, I do not understand"**

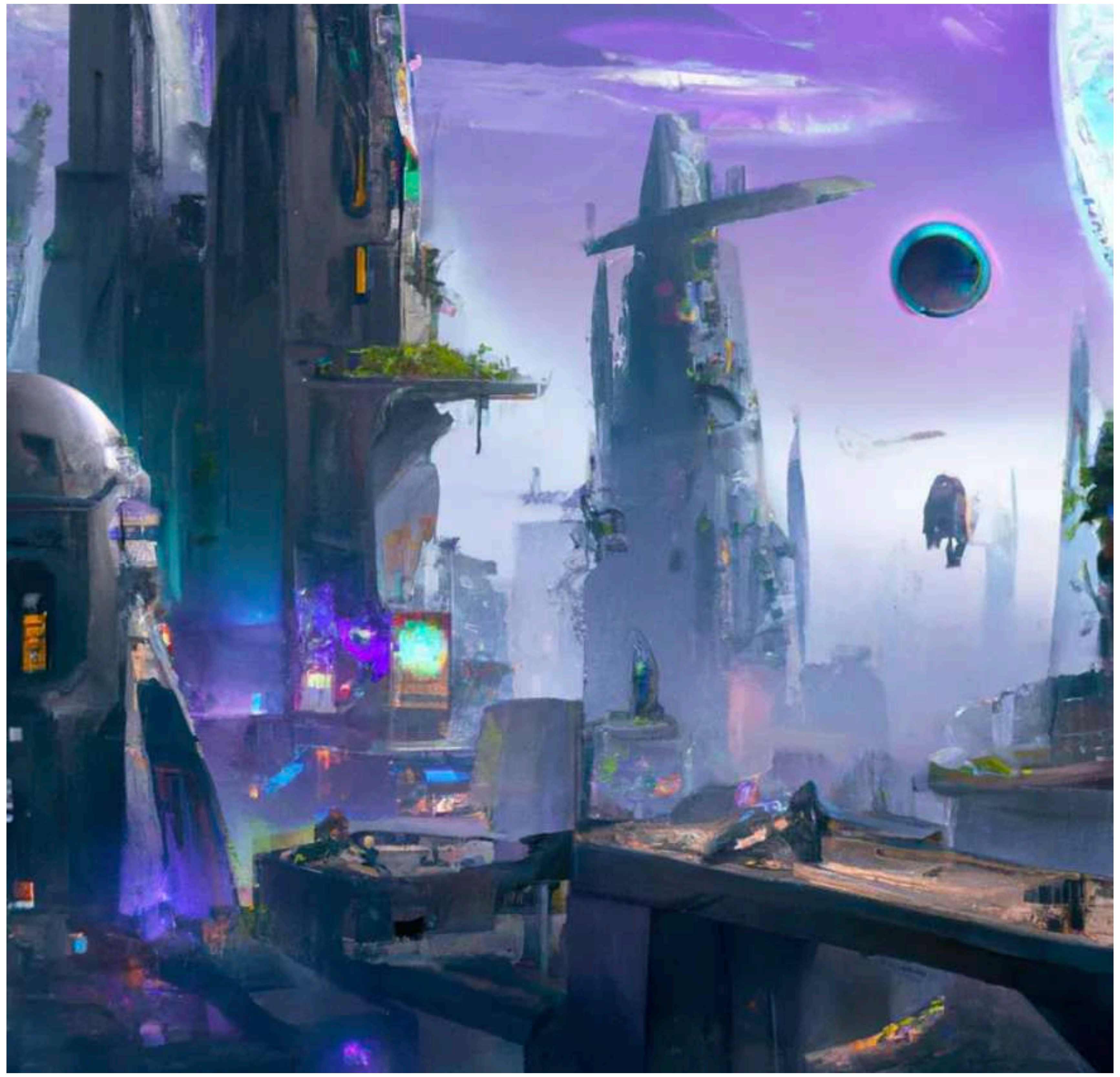
**——Richard Feynman**

**Can we create realistic images with a sparse prior ?**

# A New Era: Generative AI


by ChatGPT-4

by DALL·E 2

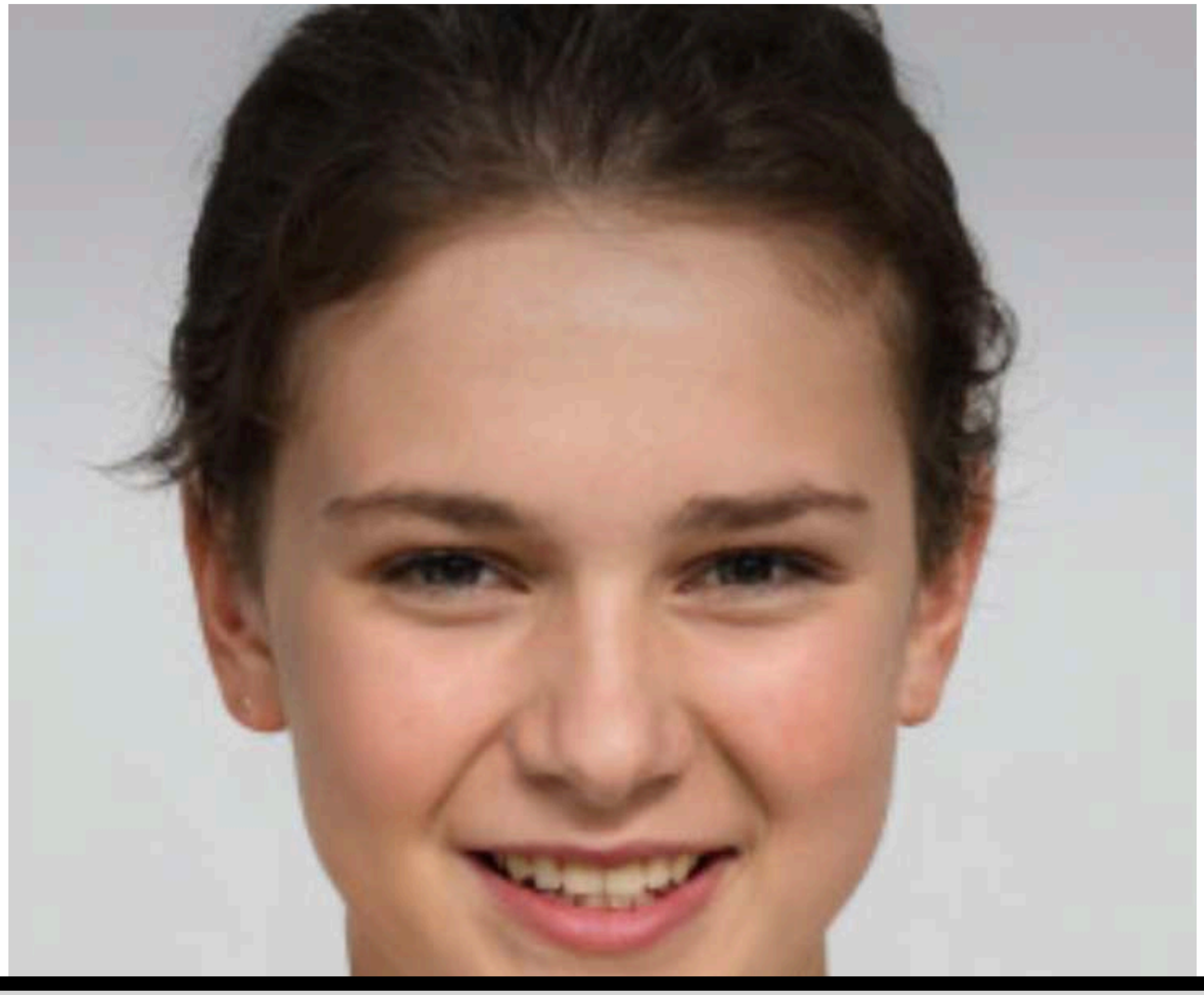


by DALL·E 2

# A New Era: Generative AI

Both are AI generated faces….

# A New Era: Generative AI

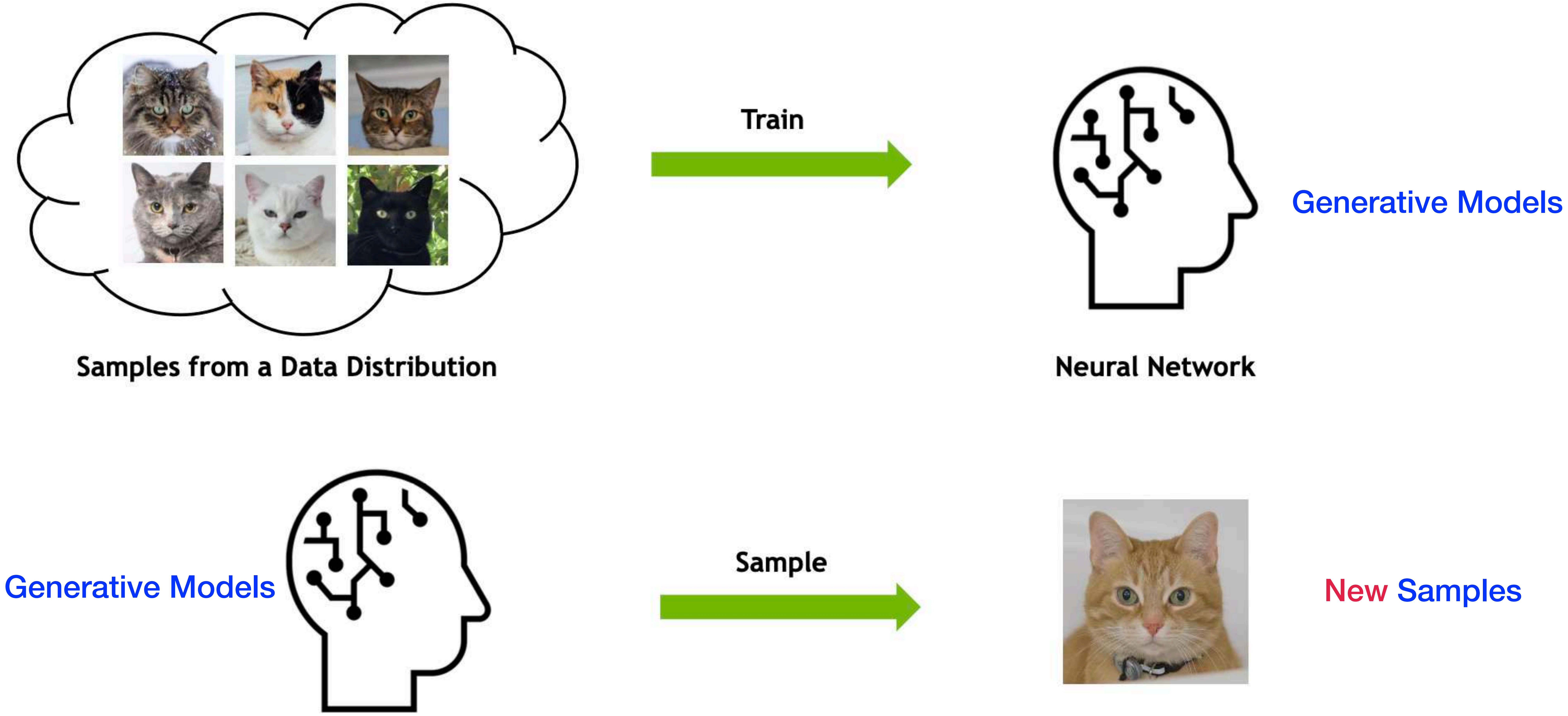


**Motivation: Can we use generative models as prior for image restoration?**

# A Tutorial Introduction to Generative Models

■ **Generative Models**

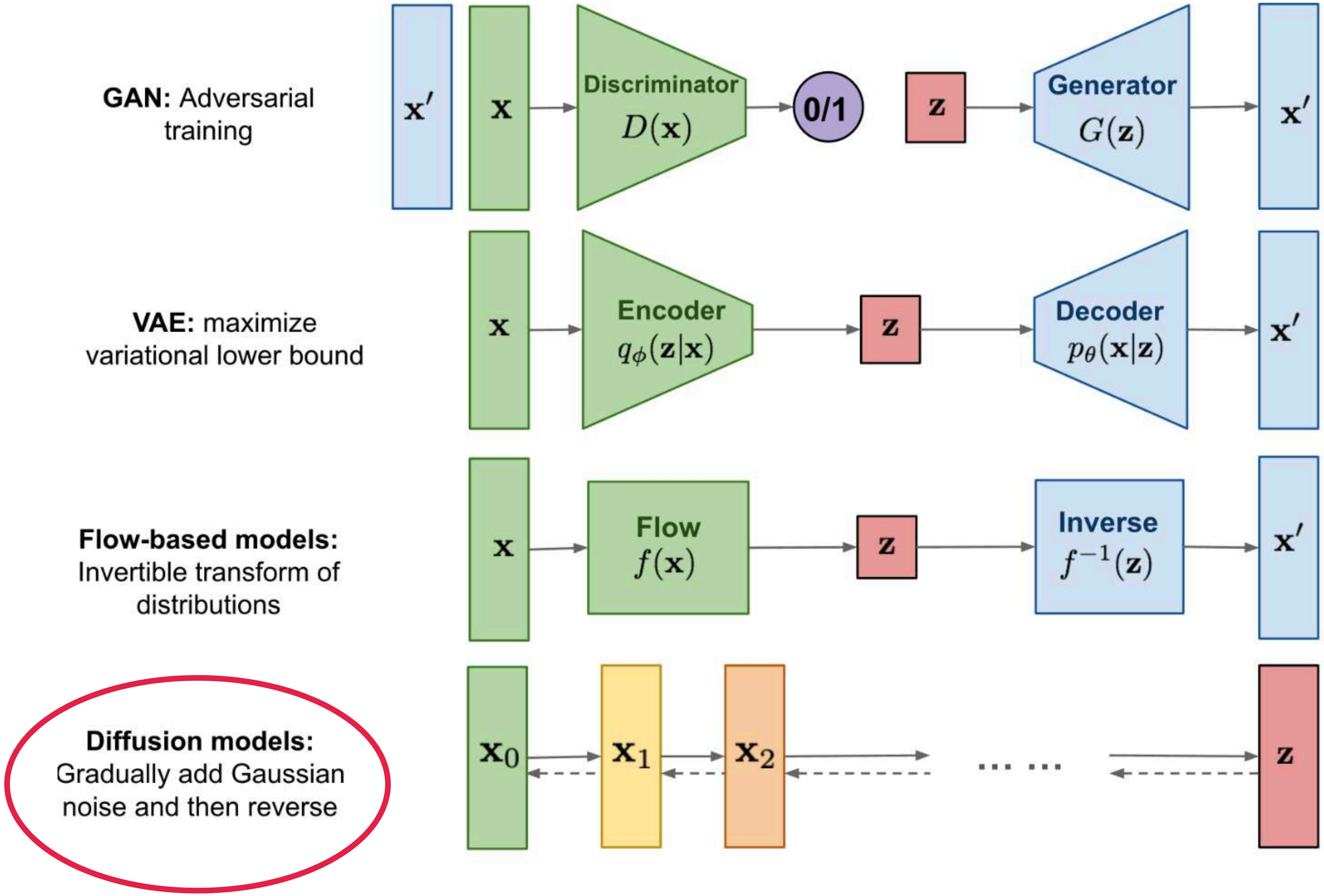
## Generative Learning




Credit to: https://cvpr2022-tutorial-diffusion-models.github.io

# A Tutorial Introduction to Generative Models

- **Different types of generative models**




Diffusion Models: Emerging as most powerful generative models

# An Old Result

- **Sampling with Langevin Dynamics**

Given score function of p(x), one can obtain samples iteratively as follows G. Parisi 1981 Welling, Max; Teh, Yee Whye 2011, Neal 2010

$$\mathbf{x}_{i+1} \leftarrow \mathbf{x}_i + \epsilon \nabla_{\mathbf{x}} \log p(\mathbf{x}) + \sqrt{2\epsilon}\, \mathbf{z}_i, \quad i = 0, 1, \cdots, K$$

step size — score function — Gaussian noise

$\mathbf{x}_K$ **converges to samples from** $p(\mathbf{x})$ **when** $\epsilon \to 0, K \to \infty$

# An Old Result

- **Sampling with Langevin Dynamics**

Given score function of p(x), one can obtain samples iteratively as follows G. Parisi 1981 Welling, Max; Teh, Yee Whye 2011, Neal 2010

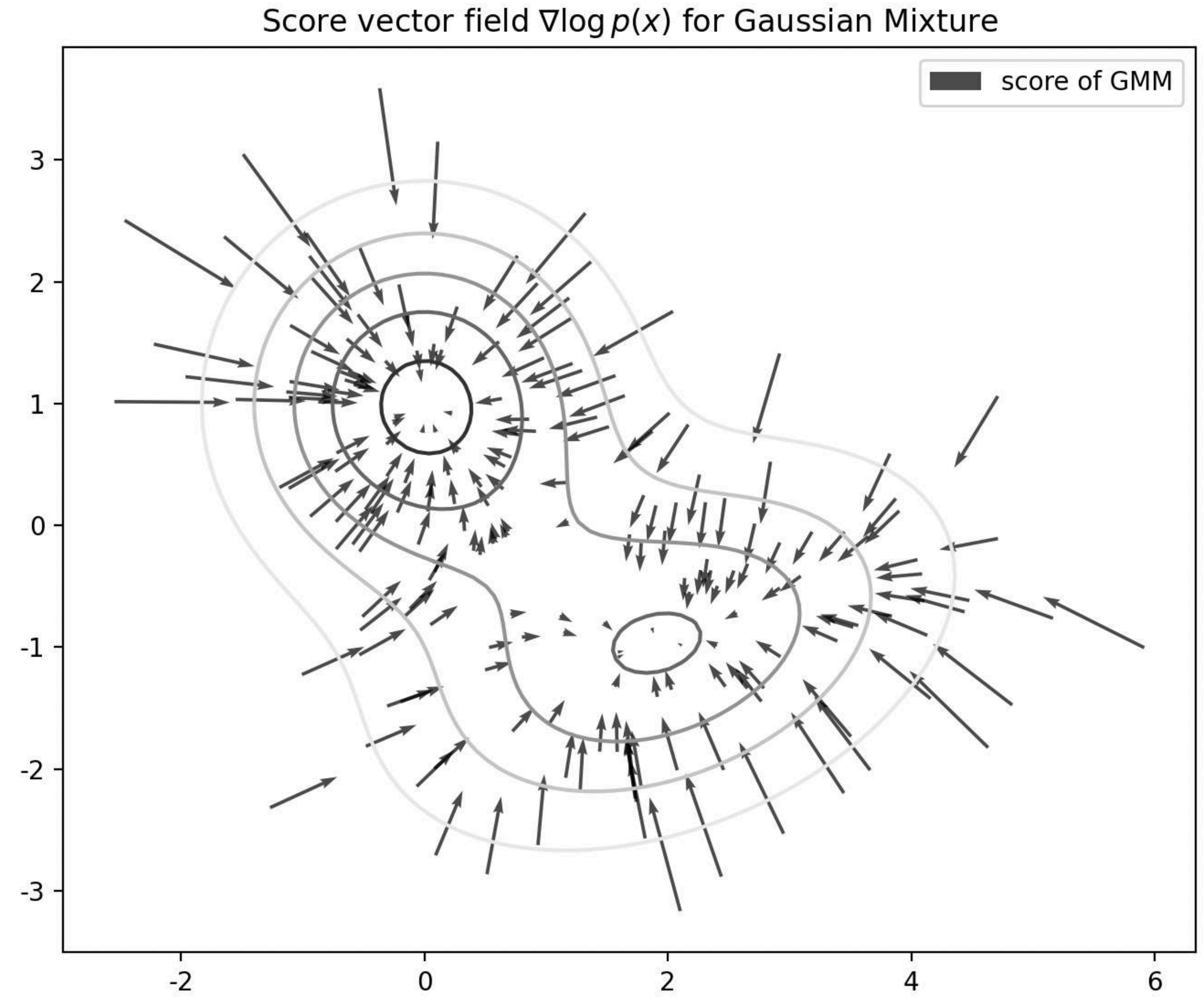
$$\mathbf{x}_{i+1} \leftarrow \mathbf{x}_i + \epsilon \nabla_{\mathbf{x}} \log p(\mathbf{x}) + \sqrt{2\epsilon}\, \mathbf{z}_i, \quad i = 0, 1, \cdots, K$$

step size — score function — Gaussian noise

$\mathbf{x}_K$ **converges to samples from** $p(\mathbf{x})$ **when** $\epsilon \to 0, K \to \infty$

- **A Toy Example**

**Two-Gaussian Mixture**




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

**Score Function: Vector Field**

# How to Estimate Score From Data Samples

- **Key Idea**

  Approximating the score function by a neural network

$$\mathbf{s}_\theta(\mathbf{x}) \approx \nabla_\mathbf{x} \log p(\mathbf{x})$$

Neural network — score function

# How to Estimate Score From Data Samples

- **Key Idea**

  Approximating the score function by a neural network

$$\mathbf{s}_\theta(\mathbf{x}) \approx \nabla_\mathbf{x} \log p(\mathbf{x})$$

Neural network — score function

Network Training

unknown target!

$$\mathbb{E}_{p(\mathbf{x})}[||\nabla_\mathbf{x} \log p(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x})||_2^2]$$

# How to Estimate Score From Data Samples

■ **Key Idea**

Approximating the score function by a neural network

$$\mathbf{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p(\mathbf{x})$$

Neural network — score function

Network Training

$$\mathbb{E}_{p(\mathbf{x})}[||\nabla_{\mathbf{x}} \log p(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x})||_2^2]$$

unknown target!

Score-Matching — A. Hyvarinen 2005

No explicit dependance on unknown $p(x)$

$$\mathbb{E}_{p(\mathbf{x})}\left[\operatorname{tr}(\nabla_{\mathbf{x}} \mathbf{s}_\theta(\mathbf{x})) + \frac{1}{2} \|\mathbf{s}_\theta(\mathbf{x})\|_2^2\right]$$

Valid loss

# How to Estimate Score From Data Samples

- **Key Idea**

Approximating the score function by a neural network

$$\mathbf{s}_\theta(\mathbf{x}) \approx \nabla_\mathbf{x} \log p(\mathbf{x})$$

Neural network — score function

Network Training

$$\mathbb{E}_{p(\mathbf{x})}[||\nabla_\mathbf{x} \log p(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x})||_2^2]$$

unknown target!

Score-Matching

$$\mathbb{E}_{p(\mathbf{x})}\left[\operatorname{tr}(\nabla_\mathbf{x}\mathbf{s}_\theta(\mathbf{x})) + \frac{1}{2}\|\mathbf{s}_\theta(\mathbf{x})\|_2^2\right]$$

Challenging for the high-dimensional case!

# Challenges of High Dimensional Score Estimation

- **Illustration via Two-Gaussian Mixture**

Estimated scores are only accurate in high density regions.

Original distribution
$p(\mathbf{x})$

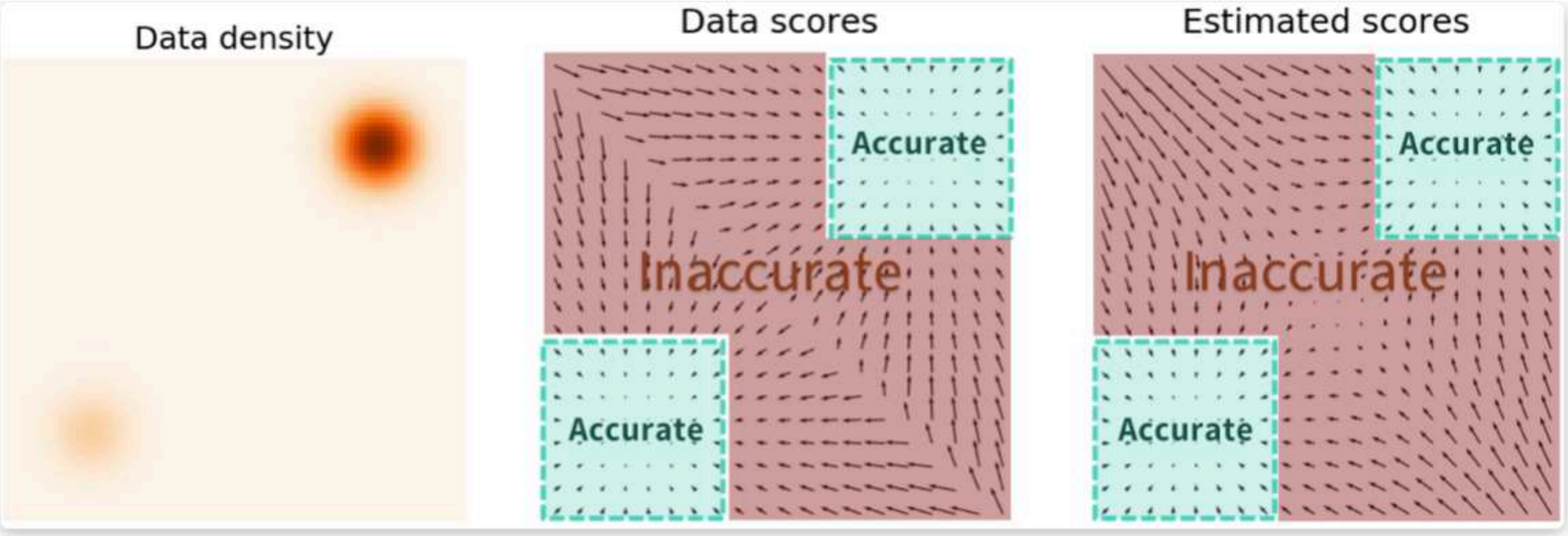


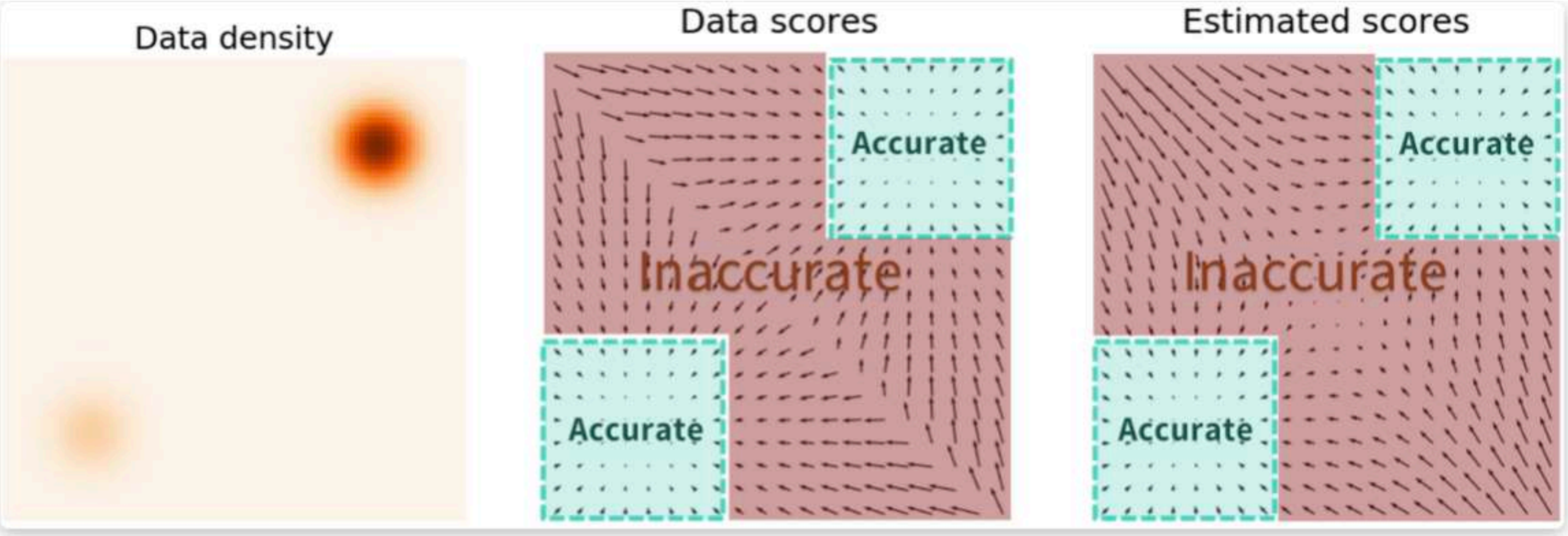

Figure credit to Yang Song

# Challenges of High Dimensional Score Estimation

- **Illustration via Two-Gaussian Mixture**

Estimated scores are only accurate in high density regions.

Original distribution
$p(\mathbf{x})$




Estimated scores are accurate everywhere for noise perturbed data

Corrupted noise

$\mathbf{x}' = \mathbf{x} + \beta\mathbf{z}$

Noise-perturbed
$p_\beta(\mathbf{x}')$

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

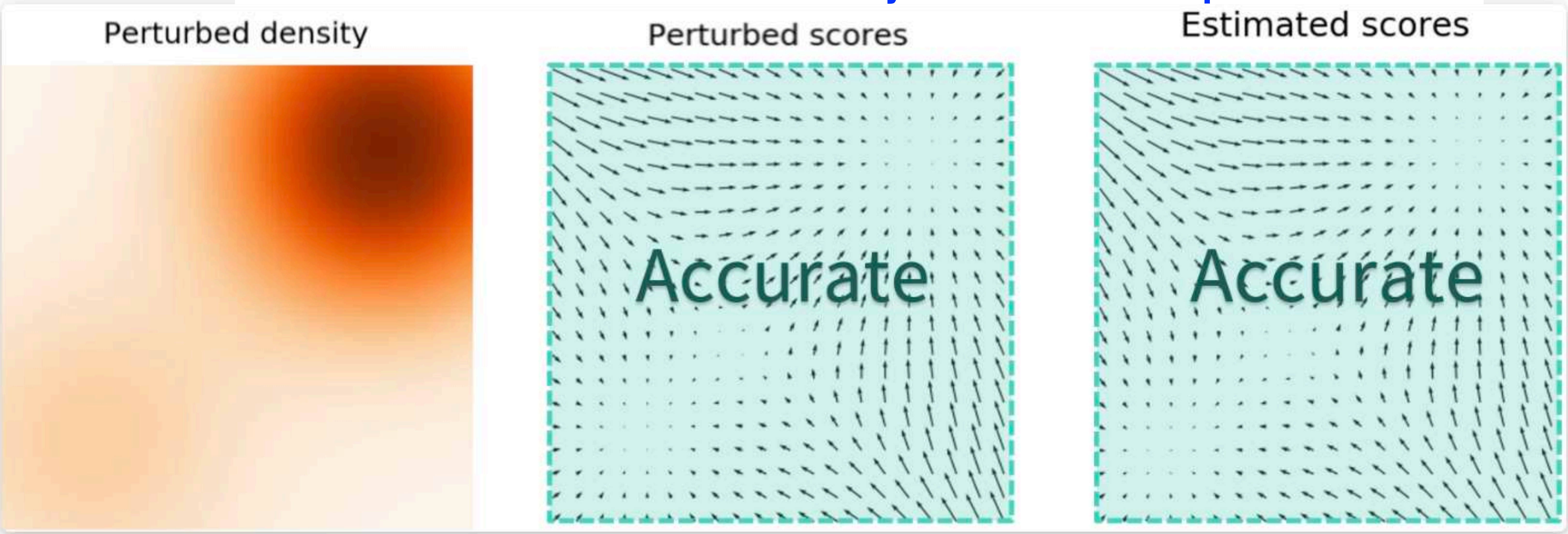


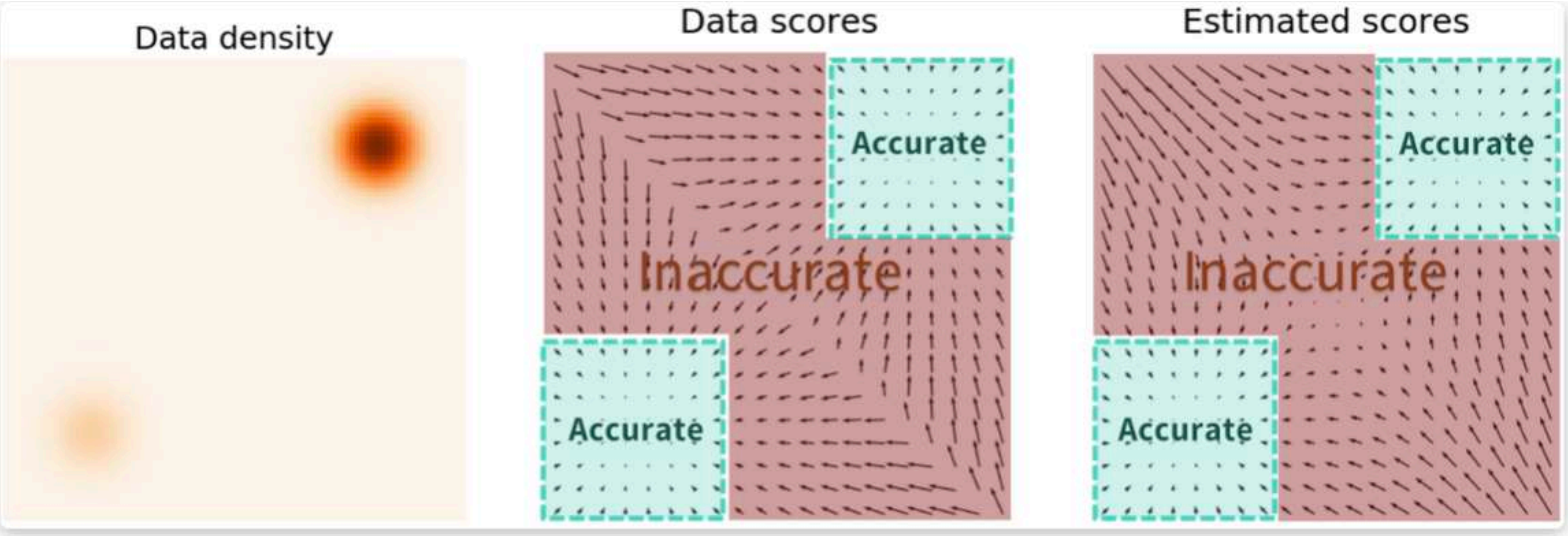

Figure credit to Yang Song

# Challenges of High Dimensional Score Estimation

- **Illustration via Two-Gaussian Mixture**

Estimated scores are only accurate in high density regions.

Original distribution
$p(\mathbf{x})$




Estimated scores are accurate everywhere for noise perturbed data

**how to choose an appropriate noise scale $\beta$ for the perturbation?**

**Large noise:** cover the low-density regions well, but different from the original distribution

**Small noise:** similar to the original distribution, but does not cover low-density regions well

# One Smart Solution: Annealing

- **Key Idea**

**Annealing: using multiple noise scales $\{\beta_t\}_{t=1}^T$ for the perturbation!**

$$\mathbf{x}_t = \mathbf{x} + \beta_t \mathbf{z} \qquad 0 < \beta_1 < \beta_2 < \cdots < \beta_T$$

$$p_{\beta_t}(\mathbf{x}_t) = \int p(\mathbf{x}) N(\mathbf{x}_t \,|\, \mathbf{x}, \beta_t^2) d\mathbf{x}$$

# One Smart Solution: Annealing

- **Key Idea**

**Annealing: using multiple noise scales $\{\beta_t\}_{t=1}^{T}$ for the perturbation!**

$$\mathbf{x}_t = \mathbf{x} + \beta_t \mathbf{z} \qquad 0 < \beta_1 < \beta_2 < \cdots < \beta_T$$

$$p_{\beta_t}(\mathbf{x}_t) = \int p(\mathbf{x}) N(\mathbf{x}_t \mid \mathbf{x}, \beta_t^2) d\mathbf{x}$$

**Network Training**

Using neural network to estimate the score $\nabla_{\mathbf{x}_t} \log p_{\beta_t}(\mathbf{x}_t)$ of each noise-perturbed distribution $p_{\beta_t}(\mathbf{x}_t)$

$$\mathbf{s}_\theta(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} \log p_{\beta_t}(\mathbf{x}_t) \ \forall t$$

Estimated Score — True Score

Loss function: $$\sum_{t=1}^{T} \lambda_t \mathbf{E}_{p_{\beta_t}(\mathbf{x}_t)} \| \nabla_{\mathbf{x}_t} \log p_{\beta_t}(\mathbf{x}_t) - \mathbf{s}_\theta(\mathbf{x}_t, t) \|^2$$

# One Smart Solution: Annealing

- **Key Idea**

**Annealing: using multiple noise scales $\{\beta_t\}_{t=1}^{T}$ for the perturbation!**

$$\mathbf{x}_t = \mathbf{x} + \beta_t \mathbf{z} \qquad 0 < \beta_1 < \beta_2 < \cdots < \beta_T$$

$\beta_1$ $\beta_2$ $\beta_3$

**samples of $\mathbf{x}_t$**

**estimated scores**

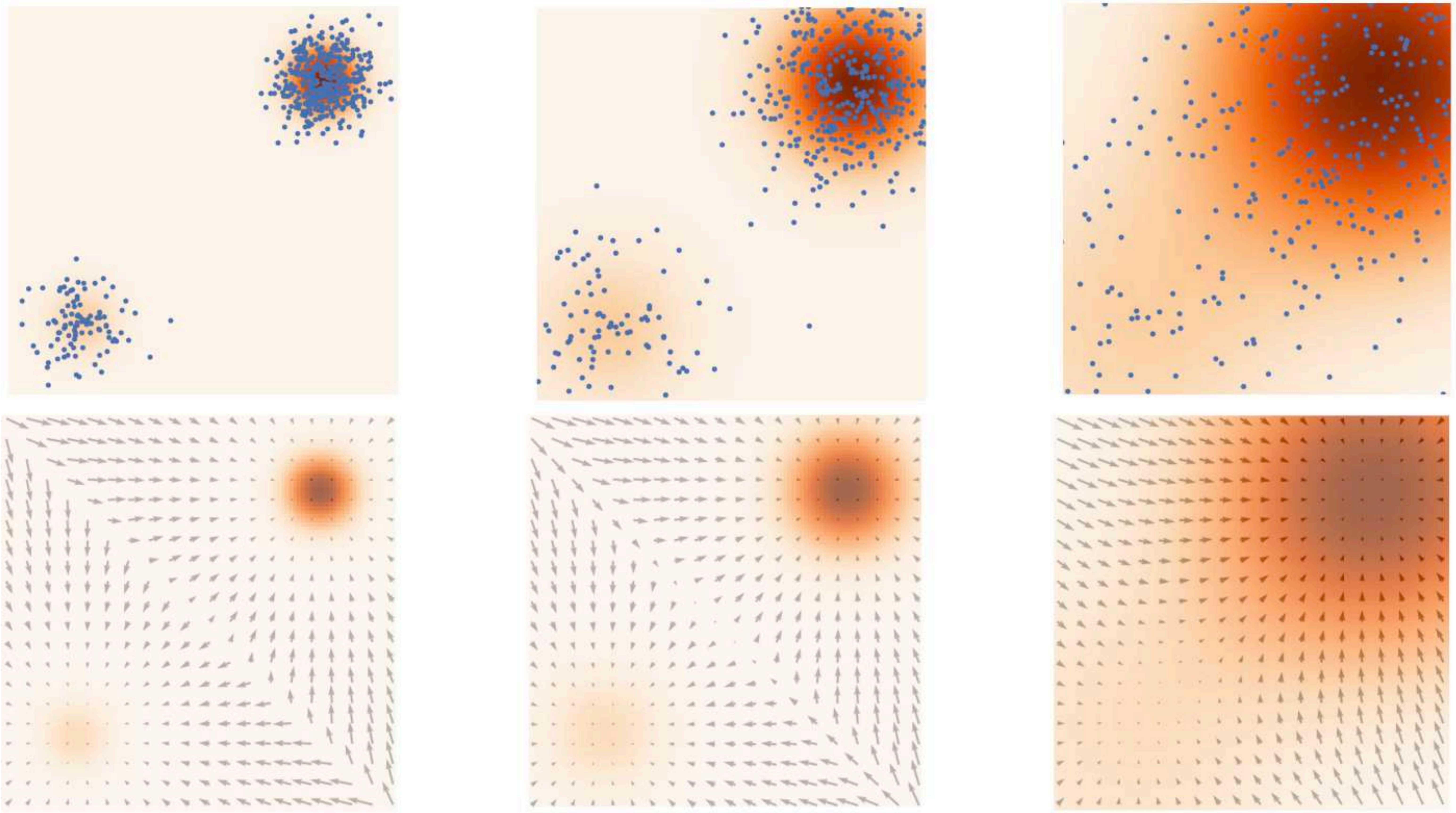


Figure credit to Yang Song

# Putting Ideas Together

- **A Big Picture**

$$\mathbf{x}_t = \mathbf{x}_0 + \beta_t \mathbf{z}_t \qquad 0 < \beta_1 < \beta_2 < \cdots < \beta_T$$

A sequence of noise levels


Forward Process

# Putting Ideas Together

- **A Big Picture**

$$\mathbf{x}_t = \mathbf{x}_0 + \beta_t \mathbf{z}_t$$

$$0 < \beta_1 < \beta_2 < \cdots < \beta_T$$

**A sequence of noise levels**

Forward diffusion process (fixed)

Data

Noise

Reverse denoising process (generative)

$$\mathbf{x}_{t-1}^k = \mathbf{x}_t^k + \alpha_t \nabla_{\mathbf{x}_t} \log p_{\beta_t}(\mathbf{x}_t) + \sqrt{2\alpha_t}\mathbf{z}_t^k$$

**Score function**

**Approximated by neural network**

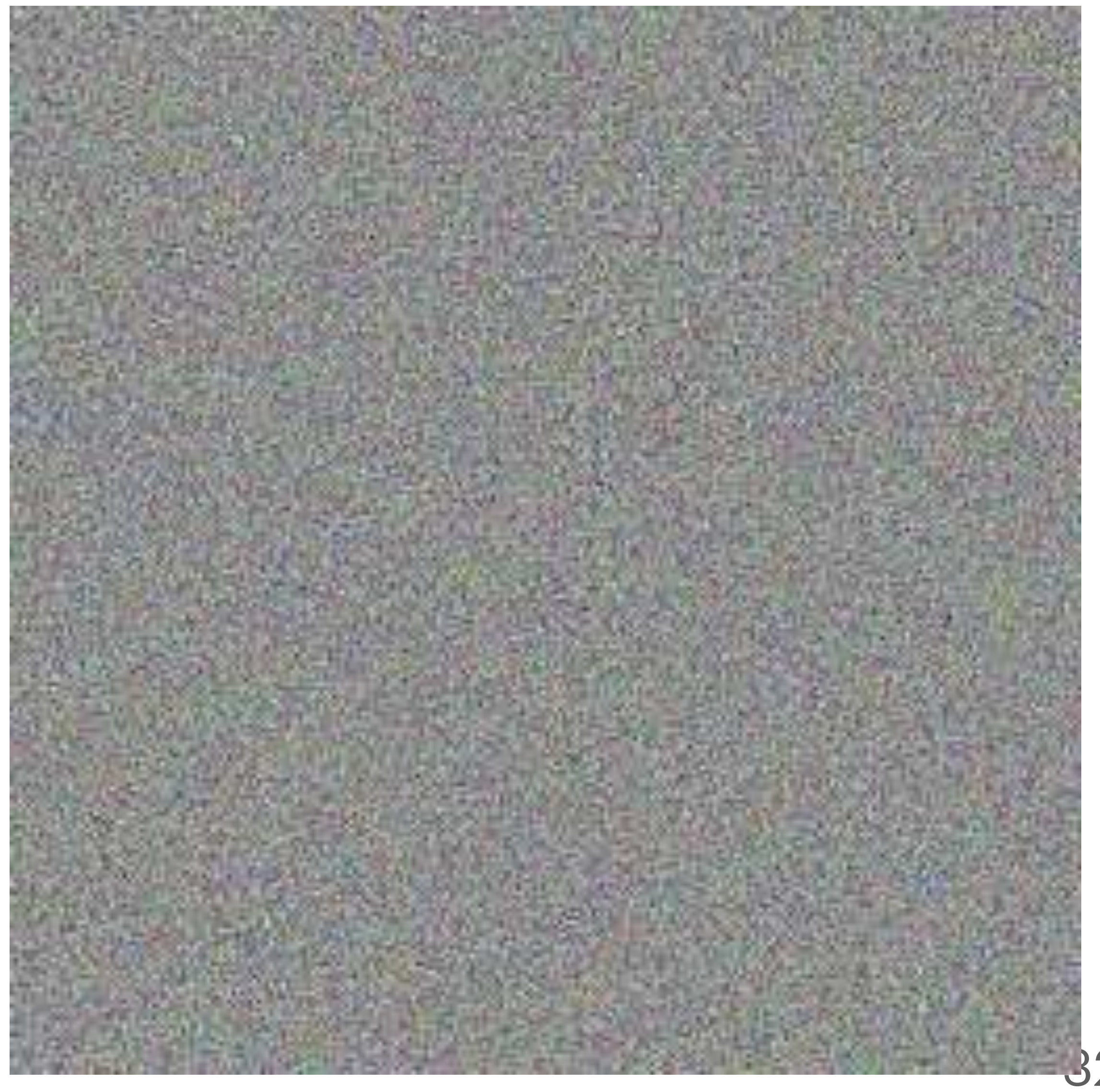
$\mathbf{s}_\theta(\mathbf{x}_t, t)$

**Forward Process**

Annealed Langevin dynamics

**Reverse it!**

**Reverse Process**



# Different Types of Diffusion Models

- **Noise Conditional Score Network (NCSN)** Yang Song, Stefano Ermon 2019

Forward: $$\mathbf{x}_t = \mathbf{x}_0 + \beta_t \mathbf{z}_t$$

Reverse: $$\mathbf{x}_{t-1}^k = \mathbf{x}_t^k + \alpha_t \nabla_{\mathbf{x}_t} \log p_{\beta_t}(\mathbf{x}_t) + \sqrt{2\alpha_t}\mathbf{z}_t^k$$

- **Denoising Diffusion Probabilistic Models (DDPM)** Jonathan Ho et al 2020

Forward: $$\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{t-1}$$

Reverse: $$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}(\mathbf{x}_t + (1-\alpha_t)\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t)) + \beta_t \mathbf{z}_t$$

- **Flow-Matching Models** Yaron Lipman 2022 Xingchao Liu et al 2022, Nanye Ma et al 2024

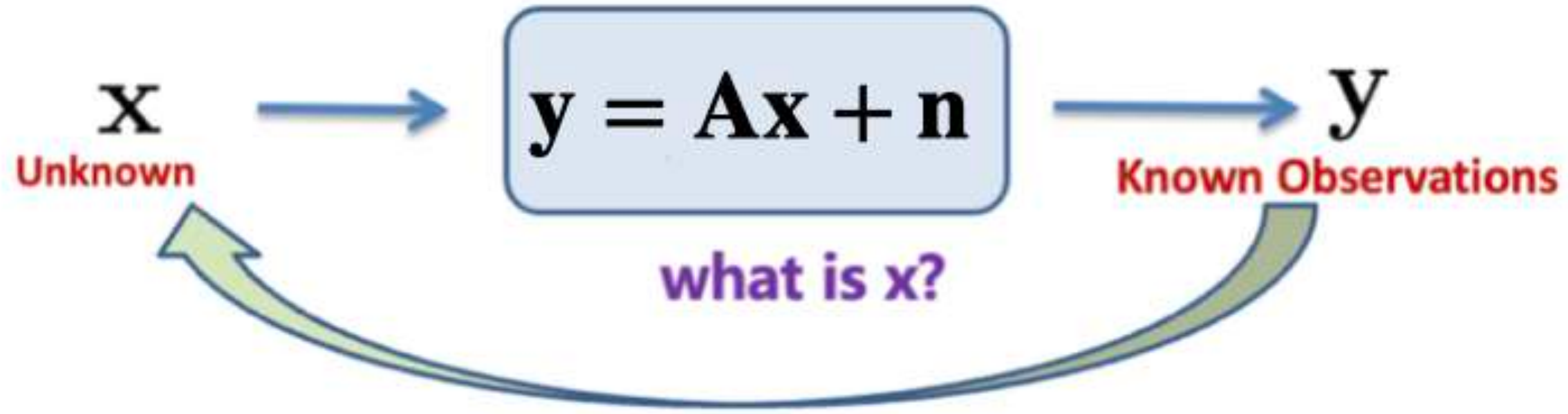
Forward: $$\mathbf{x}_t = a_t \mathbf{x}_0 + b_t \epsilon$$

Reverse: $$\mathbf{x}_{t-1} = \mathbf{x}_t - \left(\frac{\dot{a}_t}{a_t}\mathbf{x}_t + \frac{b_t(\dot{a}_t b_t - a_t \dot{b}_t)}{a_t}\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t)\right)\Delta_t$$

# Contents

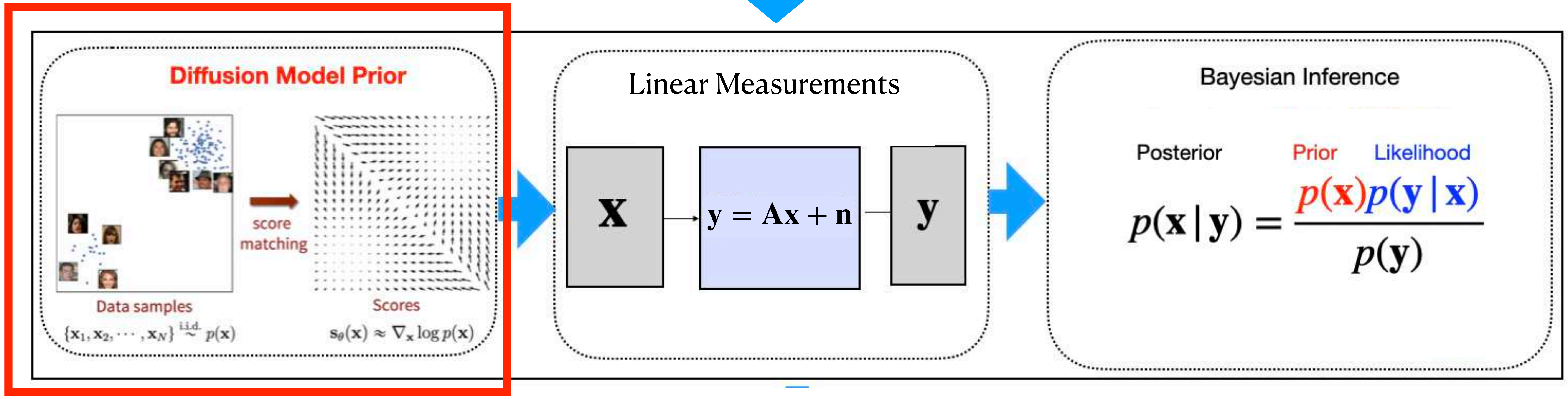
# Generative Image Restoration

- **A New Paradigm For Image Restoration**




**Using Generative Model as Prior**

Generative Modeling $p(\mathbf{x} \mid \mathbf{y}) = \dfrac{p(\mathbf{x})p(\mathbf{y} \mid \mathbf{x})}{p(\mathbf{y})}$




**Challenge: How can we sample from the posterior $p(\mathbf{x} \mid \mathbf{y})$ ?**

# Generative Image Restoration

- **Posterior Sampling**

**Prior Sampling**

NCSN: $\mathbf{x}_{t-1}^{k} = \mathbf{x}_t^k + \alpha_t \nabla_{\mathbf{x}_t} \log p_{\beta_t}(\mathbf{x}_t) + \sqrt{2\alpha_t}\mathbf{z}_t^k$

DDPM: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}(\mathbf{x}_t + (1-\alpha_t)\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t)) + \beta_t \mathbf{z}_t,$

Flow-based: $\mathbf{x}_{t-1} = \mathbf{x}_t - (\frac{\dot{a}_t}{a_t}\mathbf{x}_t + \frac{b_t(\dot{a}_t b_t - a_t \dot{b}_t)}{a_t}\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t))\Delta_t$

Available From Pre-trained Diffusion Models

# Generative Image Restoration

- **Posterior Sampling**

**Prior Sampling**

NCSN: $\mathbf{x}_{t-1}^k = \mathbf{x}_t^k + \alpha_t \nabla_{\mathbf{x}_t} \log p_{\beta_t}(\mathbf{x}_t) + \sqrt{2\alpha_t}\mathbf{z}_t^k$

DDPM: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}(\mathbf{x}_t + (1-\alpha_t)\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t)) + \beta_t \mathbf{z}_t,$

Flow-based: $\mathbf{x}_{t-1} = \mathbf{x}_t - (\frac{\dot{a}_t}{a_t}\mathbf{x}_t + \frac{b_t(\dot{a}_t b_t - a_t \dot{b}_t)}{a_t}\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t))\Delta_t$

Available From Pre-trained Diffusion Models

**Bayes' Rule**

$$p(\mathbf{x} \mid \mathbf{y}) = \frac{p(\mathbf{x})p(\mathbf{y} \mid \mathbf{x})}{p(\mathbf{y})}$$

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

**Posterior Score** — Prior Score — Likelihood Score

# Generative Image Restoration

■ **Posterior Sampling**

**Prior Sampling**

NCSN: $\mathbf{x}_{t-1}^k = \mathbf{x}_t^k + \alpha_t \nabla_{\mathbf{x}_t} \log p_{\beta_t}(\mathbf{x}_t) + \sqrt{2\alpha_t}\mathbf{z}_t^k$

DDPM: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}(\mathbf{x}_t + (1-\alpha_t)\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t)) + \beta_t \mathbf{z}_t,$

Flow-based: $\mathbf{x}_{t-1} = \mathbf{x}_t - (\frac{\dot{a}_t}{a_t}\mathbf{x}_t + \frac{b_t(\dot{a}_t b_t - a_t \dot{b}_t)}{a_t}\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t))\Delta_t$

Available From Pre-trained Diffusion Models

**Bayes' Rule**

$$p(\mathbf{x}\,|\,\mathbf{y}) = \frac{p(\mathbf{x})p(\mathbf{y}\,|\,\mathbf{x})}{p(\mathbf{y})}$$

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

**Posterior Score** | Prior Score | Likelihood Score

**Posterior Sampling**

NCSN: $\mathbf{x}_{t-1}^k = \mathbf{x}_t^k + \alpha_t( \nabla_{\mathbf{x}_t} \log p_{\beta_t}(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p_{\beta_t}(\mathbf{y}\,|\,\mathbf{x}_t)) + \sqrt{2\alpha_t}\mathbf{z}_t^k$

DDPM: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}(\mathbf{x}_t + (1-\alpha_t)\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t))) + \beta_t \mathbf{z}_t,$

Flow-based: $\mathbf{x}_{t-1} = \mathbf{x}_t - (\frac{\dot{a}_t}{a_t}\mathbf{x}_t + \frac{b_t(\dot{a}_t b_t - a_t \dot{b}_t)}{a_t}\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)))\Delta_t$

# Generative Image Restoration

- Posterior Sampling

**Prior Sampling**

NCSN: $\mathbf{x}_{t-1}^k = \mathbf{x}_t^k + \alpha_t \nabla_{\mathbf{x}_t} \log p_{\beta_t}(\mathbf{x}_t) + \sqrt{2\alpha_t}\mathbf{z}_t^k$

DDPM: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}(\mathbf{x}_t + (1-\alpha_t)\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t)) + \beta_t \mathbf{z}_t,$

Flow-based: $\mathbf{x}_{t-1} = \mathbf{x}_t - (\frac{\dot{a}_t}{a_t}\mathbf{x}_t + \frac{b_t(\dot{a}_t b_t - a_t \dot{b}_t)}{a_t}\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t))\Delta_t$

Available From Pre-trained Diffusion Models

**Bayes' Rule**

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

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

**Posterior Score** Prior Score Likelihood Score

**Posterior Sampling**

NCSN: $\mathbf{x}_{t-1}^k = \mathbf{x}_t^k + \alpha_t( \nabla_{\mathbf{x}_t} \log p_{\beta_t}(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p_{\beta_t}(\mathbf{y}|\mathbf{x}_t)) + \ldots$

DDPM: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}(\mathbf{x}_t + (1-\alpha_t)\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)))\ldots$

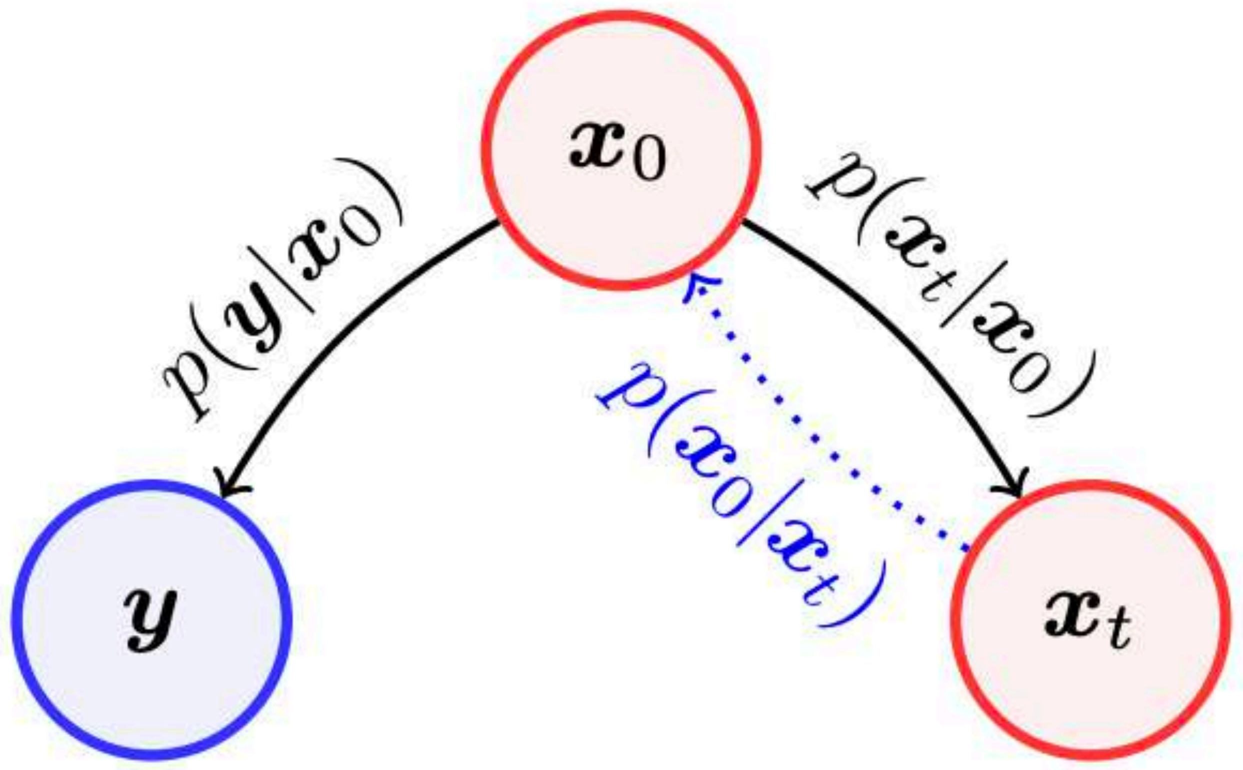
Flow-based: $\mathbf{x}_{t-1} = \mathbf{x}_t - (\frac{\dot{a}_t}{a_t}\mathbf{x}_t + \frac{b_t(\dot{a}_t b_t - a_t \dot{b}_t)}{a_t}\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)))\Delta_t$

**The remaining goal is to Compute $\nabla_{\mathbf{x}} \log p(\mathbf{y}|\mathbf{x})$**

# Generative Image Restoration

■ **Key Challenge**

The likelihood score $\nabla_{\mathbf{x}_t} \log p(\mathbf{y} \mid \mathbf{x}_t)$ is intractable except *t*=0, *even for the linear case* $\mathbf{y} = \mathbf{A}\mathbf{x}_0 + \mathbf{n}$




Graphical Model

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

$$= \int \underbrace{p(\mathbf{y} \mid \mathbf{x}_0)}_{\text{Gauss}} \underbrace{p(\mathbf{x}_0 \mid \mathbf{x}_t)}_{\text{intractable!}} d\mathbf{x}_0,$$

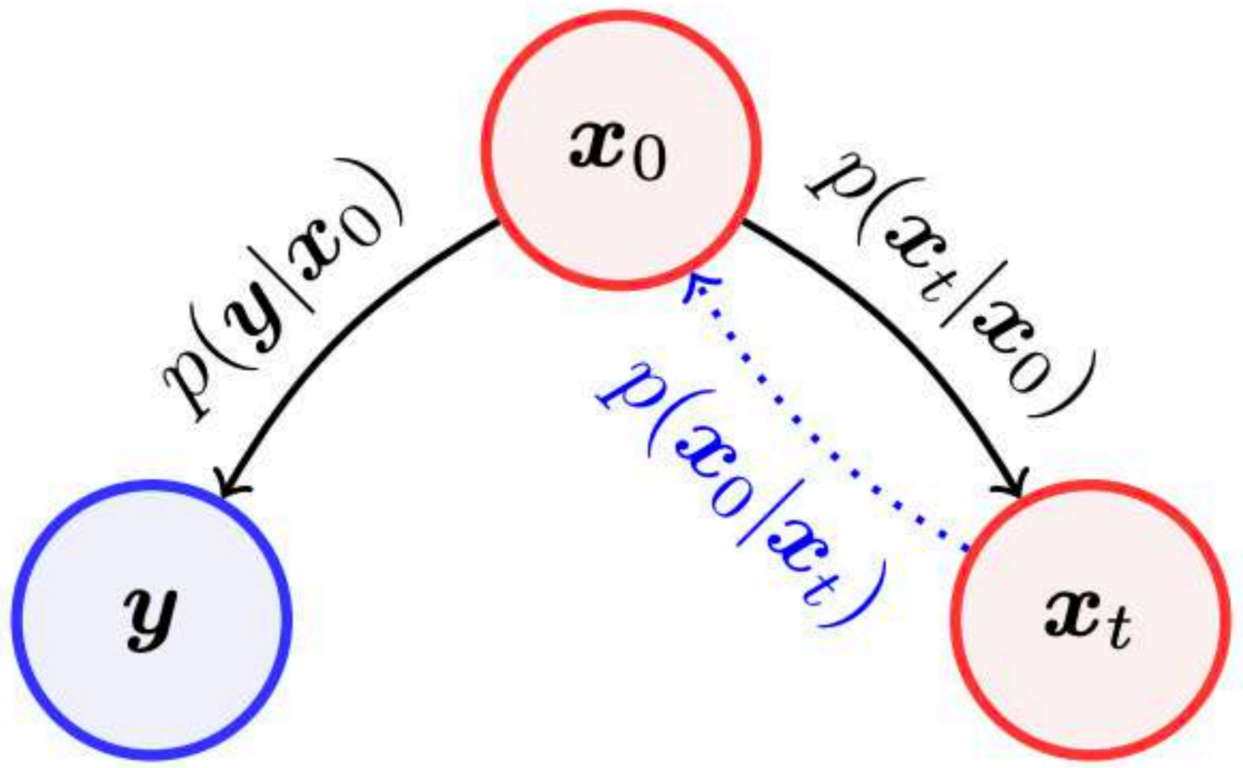
**Tweedie's formula:** (Robbins, 1992; Stein, 1981)

$$\hat{\mathbf{x}}_0(\mathbf{x}_t) := \mathbb{E}[\mathbf{x}_0 \mid \mathbf{x}_t] = \frac{1}{\sqrt{\bar{\alpha}(t)}} \left( \mathbf{x}_t + (1 - \bar{\alpha}(t)) \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) \right)$$

# Generative Image Restoration

- **Key Challenge**

The likelihood score $\nabla_{\mathbf{x}_t} \log p(\mathbf{y} | \mathbf{x}_t)$ is intractable except *t*=0, *even for the linear case* $\mathbf{y} = \mathbf{A}\mathbf{x}_0 + \mathbf{n}$




Graphical Model

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

$$= \int \underbrace{p(\mathbf{y} \mid \mathbf{x}_0)}_{\text{Gauss}} \underbrace{p(\mathbf{x}_0 \mid \mathbf{x}_t)}_{\text{intractable!}} d\mathbf{x}_0,$$

**Tweedie's formula:** (Robbins, 1992; Stein, 1981)

$$\hat{\mathbf{x}}_0(\mathbf{x}_t) := \mathbb{E}[\mathbf{x}_0 \mid \mathbf{x}_t] = \frac{1}{\sqrt{\bar{\alpha}(t)}} \left( \mathbf{x}_t + (1 - \bar{\alpha}(t)) \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) \right)$$

- **Most Popular Solutions**

DPS Chung et al. (2022a)

$$p(\mathbf{y} \mid \mathbf{x}_t) \approx \mathcal{N}(\mathbf{A}\hat{\mathbf{x}}_0(\mathbf{x}_t); \sigma_y^2 \mathbf{I})$$

PGDM Song et al. (2022)

$$p(\mathbf{y} | \mathbf{x}_t) \approx \mathcal{N}(\mathbf{A}\hat{\mathbf{x}}_0(\mathbf{x}_t); \gamma_t^2 \mathbf{A}\mathbf{A}^T + \sigma_y^2 \mathbf{I})$$

$$\nabla_{\mathbf{x}_t} \log p(\mathbf{y} | \mathbf{x}_t) \approx \frac{\partial^T \hat{\mathbf{x}}_0(\mathbf{x}_t)}{\partial \mathbf{x}_t} \nabla_{\hat{\mathbf{x}}_0(\mathbf{x}_t)} \log \tilde{p}(\mathbf{y} | \hat{\mathbf{x}}_0(\mathbf{x}_t))$$

The Jacobian needs back-propagation through diffusion models, which is time-consuming

# One Simple Solution: DMPS

- **A Simple Alternative Approximation**

intractable

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

**Motivation:** Is it possible to obtain a closed-form approximation for $p(\mathbf{x}_0 \mid \mathbf{x}_t)$?

Gaussian Intractable

$$p(\mathbf{x}_0 \mid \mathbf{x}_t) = \frac{\boxed{p(\mathbf{x}_t \mid \mathbf{x}_0)}\ \boxed{p(\mathbf{x}_0)}}{\int p(\mathbf{x}_t \mid \mathbf{x}_0) p(\mathbf{x}_0) d\mathbf{x}_0}$$

closed-form?

# One Simple Solution: DMPS

■ **A Simple Alternative Approximation**

intractable

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

**Motivation:** Is it possible to obtain a closed-form approximation for $p(\mathbf{x}_0 \mid \mathbf{x}_t)$?

Gaussian Intractable

$$p(\mathbf{x}_0 \mid \mathbf{x}_t) = \frac{\boxed{p(\mathbf{x}_t \mid \mathbf{x}_0)} \boxed{p(\mathbf{x}_0)}}{\int p(\mathbf{x}_t \mid \mathbf{x}_0) p(\mathbf{x}_0) d\mathbf{x}_0}$$

closed-form?

- **Assumption 1**

The prior $p(\mathbf{x}_0)$ is non-informative w.r.t. $p(\mathbf{x}_t \mid \mathbf{x}_0)$

Gaussian

$$\boxed{p(\mathbf{x}_0 \mid \mathbf{x}_t) \propto p(\mathbf{x}_t \mid \mathbf{x}_0)}$$

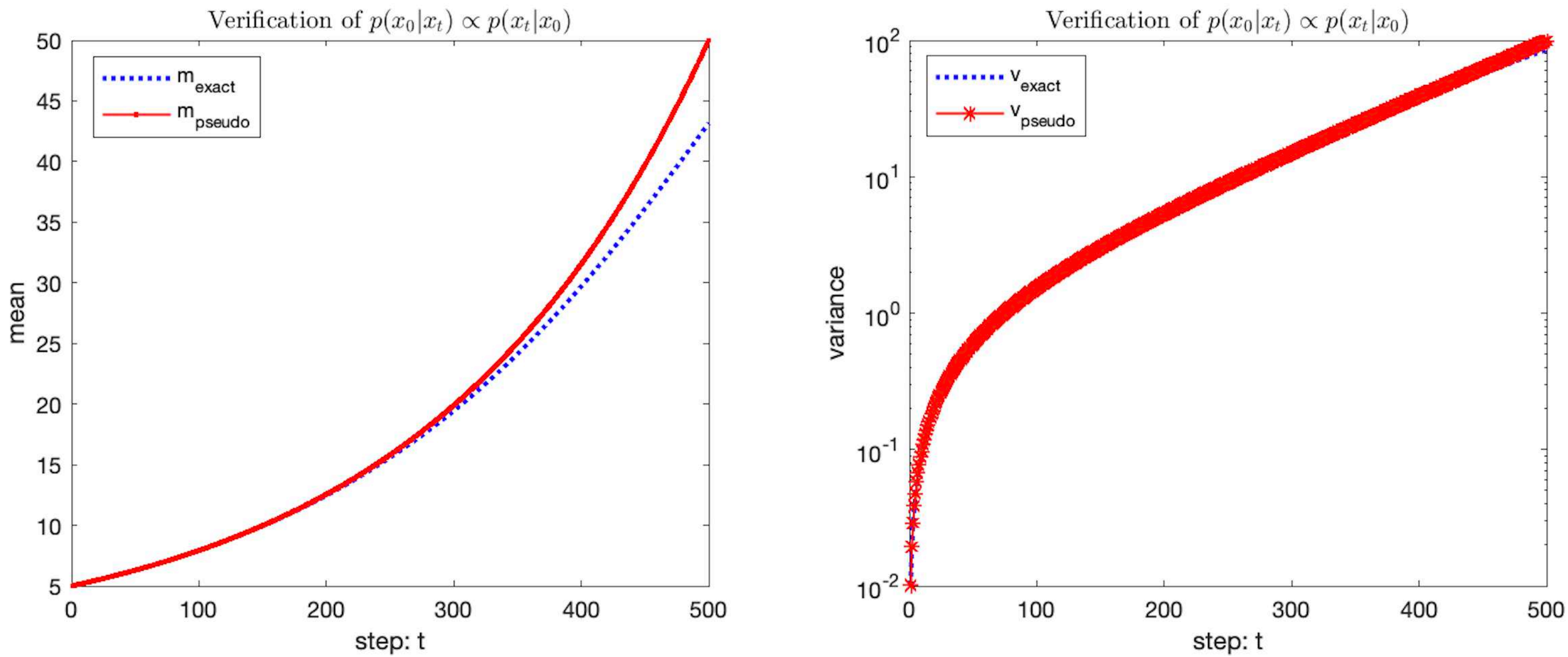
Closed-Form
Gaussian Approximation

**Asymptotically accurate when the perturbed noise is negligible**

# One Simple Solution: DMPS

- **A Simple Alternative Approximation**

**Assumption 1 is asymptotically accurate when the perturbed noise is negligible, i.e., *t* is small**




**A Toy Example with a Gaussian $p(x_0)$**

# One Simple Solution: DMPS

- **Closed-form noise-perturbed likelihood score** $\nabla_{\mathbf{x}_t} \log p(\mathbf{y} \mid \mathbf{x}_t)$

**Theorem 1.** *(noise-perturbed pseudo-likelihood score, DDPM) For DDPM, under Assumption 1, the noise-perturbed likelihood score* $\nabla_{\mathbf{x}_t} \log p(\mathbf{y} \mid \mathbf{x}_t)$ *for* $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{n}$ *in (1) admits a closed-form*

$$\nabla_{\mathbf{x}_t} \log p(\mathbf{y} \mid \mathbf{x}_t) \simeq \nabla_{\mathbf{x}_t} \log \tilde{p}(\mathbf{y} \mid \mathbf{x}_t) = \frac{1}{\sqrt{\bar{\alpha}_t}} \mathbf{A}^T \left(\sigma^2 \mathbf{I} + \frac{1 - \bar{\alpha}_t}{\bar{\alpha}_t} \mathbf{A}\mathbf{A}^T\right)^{-1} \left(\mathbf{y} - \frac{1}{\sqrt{\bar{\alpha}_t}} \mathbf{A}\mathbf{x}_t\right). \tag{10}$$

- **Efficient Computation via SVD**

**Theorem 2.** *(efficient computation via SVD) For DDPM, the noise-perturbed pseudo-likelihood score* $\nabla_{\mathbf{x}_t} \log p(\mathbf{y} \mid \mathbf{x}_t)$ *in (10) of Theorem 1 can be equivalently computed as*

$$\nabla_{\mathbf{x}_t} \log p(\mathbf{y} \mid \mathbf{x}_t) \simeq \nabla_{\mathbf{x}_t} \log \tilde{p}(\mathbf{y} \mid \mathbf{x}_t) = \frac{1}{\sqrt{\bar{\alpha}_t}} \mathbf{V}\mathbf{\Sigma} \left(\sigma^2 \mathbf{I} + \frac{1 - \bar{\alpha}_t}{\bar{\alpha}_t} \mathbf{\Sigma}^2\right)^{-1} \left(\mathbf{U}^T \mathbf{y} - \frac{1}{\sqrt{\bar{\alpha}_t}} \mathbf{\Sigma}\mathbf{V}^T \mathbf{x}_t\right), \tag{12}$$

*where* $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ *is the SVD of* $\mathbf{A}$ *and* $\mathbf{\Sigma}^2$ *denotes element-wise square of* $\mathbf{\Sigma}$.

# One Simple Solution: DMPS

- **Resultant DMPS Algorithm**

**Algorithm 1** DMPS (DDPM version)

**Input:** $\mathbf{y}$, $\mathbf{A}$, $\sigma_y^2$, $\{\tilde{\sigma}_t\}_{t=1}^{T}$, $\lambda$

**Initialization:** $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$

**for** $t = T$ ***to*** $1$ **do**

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

  $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \mathrm{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\right) + \tilde{\sigma}_t \mathbf{z}_t$

  $\nabla_{\mathbf{x}_t} \log \tilde{p}(\mathbf{y}|\mathbf{x}_t)$
  $= \frac{1}{\sqrt{\bar{\alpha}_t}} \mathbf{V}\boldsymbol{\Sigma}\left(\sigma_y^2 \mathbf{I} + \frac{1-\bar{\alpha}_t}{\bar{\alpha}_t}\boldsymbol{\Sigma}^2\right)^{-1} \mathbf{U}^T\left(\mathbf{y} - \frac{1}{\sqrt{\bar{\alpha}_t}}\mathbf{A}\mathbf{x}_t\right)$

  $\mathbf{x}_{t-1} = \mathbf{x}_{t-1} + \lambda \frac{1-\alpha_t}{\sqrt{\alpha_t}} \nabla_{\mathbf{x}_t} \log \tilde{p}(\mathbf{y}|\mathbf{x}_t)$

**Output:** $\mathbf{x}_0$

**Algorithm 2** DMPS (flow-based version)

**Input:** $\mathbf{y}$, $\mathbf{A}$, $\sigma_y^2$, $\Delta_t = 1/T$, $\lambda$

**Initialization:** $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$

**for** $t = T$ ***to*** $1$ **do**

  $\mathbf{x}_{t-1} = \mathbf{x}_t - \mathrm{v}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\Delta_t$

  $\nabla_{\mathbf{x}_t} \log \tilde{p}(\mathbf{y}|\mathbf{x}_t)$
  $= \frac{1}{a_t} \mathbf{V}\boldsymbol{\Sigma}\left(\sigma_y^2 \mathbf{I} + \frac{b_t^2}{a_t^2}\boldsymbol{\Sigma}^2\right)^{-1} \mathbf{U}^T\left(\mathbf{y} - \frac{1}{\sqrt{\bar{\alpha}_t}}\mathbf{A}\mathbf{x}_t\right)$

  $\mathbf{x}_{t-1} = \mathbf{x}_{t-1} - \lambda \frac{b_t(\dot{a}_t b_t - a_t \dot{b}_t)}{a_t} \log \tilde{p}(\mathbf{y}|\mathbf{x}_t)\Delta_t$

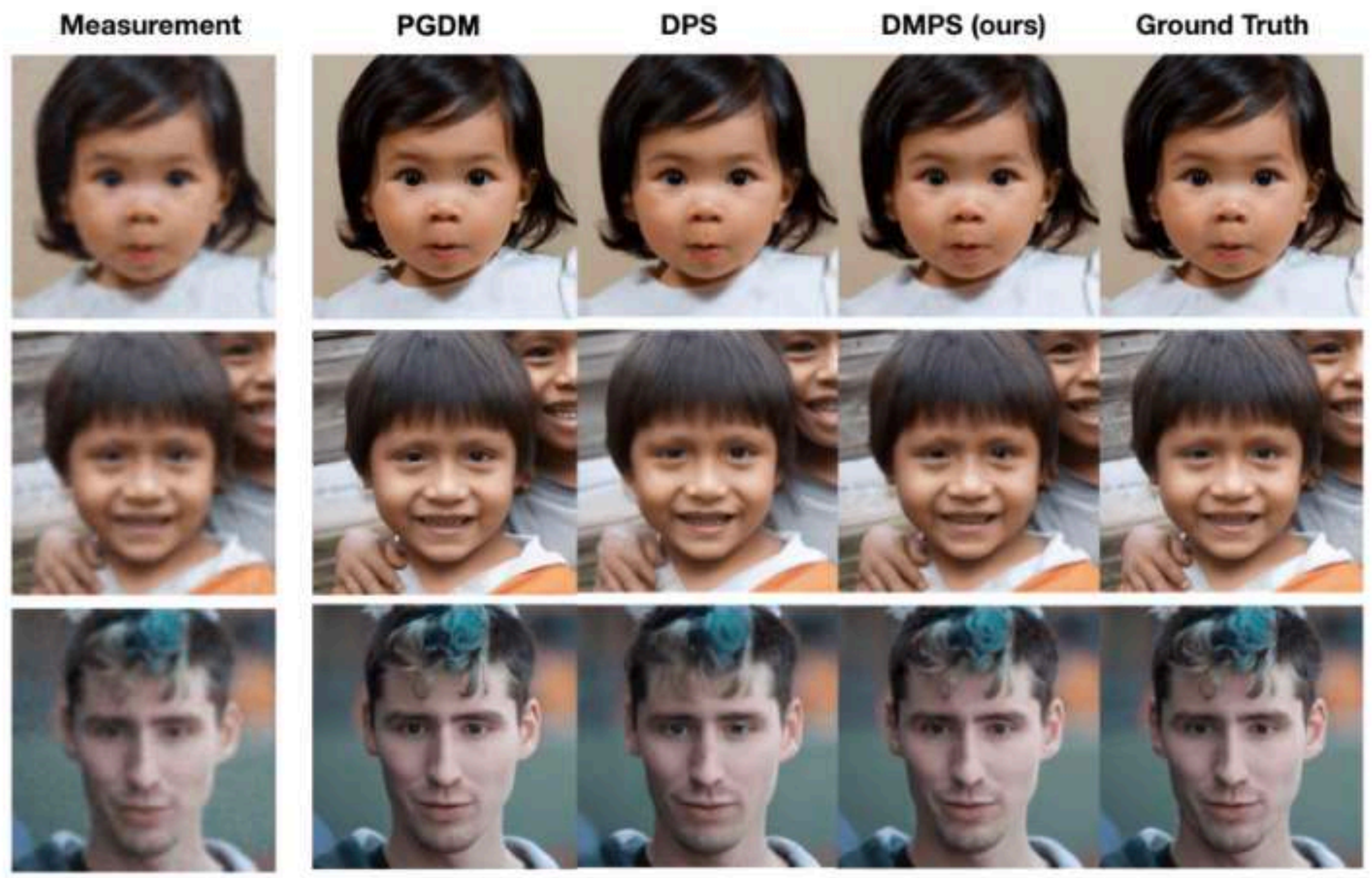
**Output:** $\mathbf{x}_0$

# One Simple Solution: DMPS

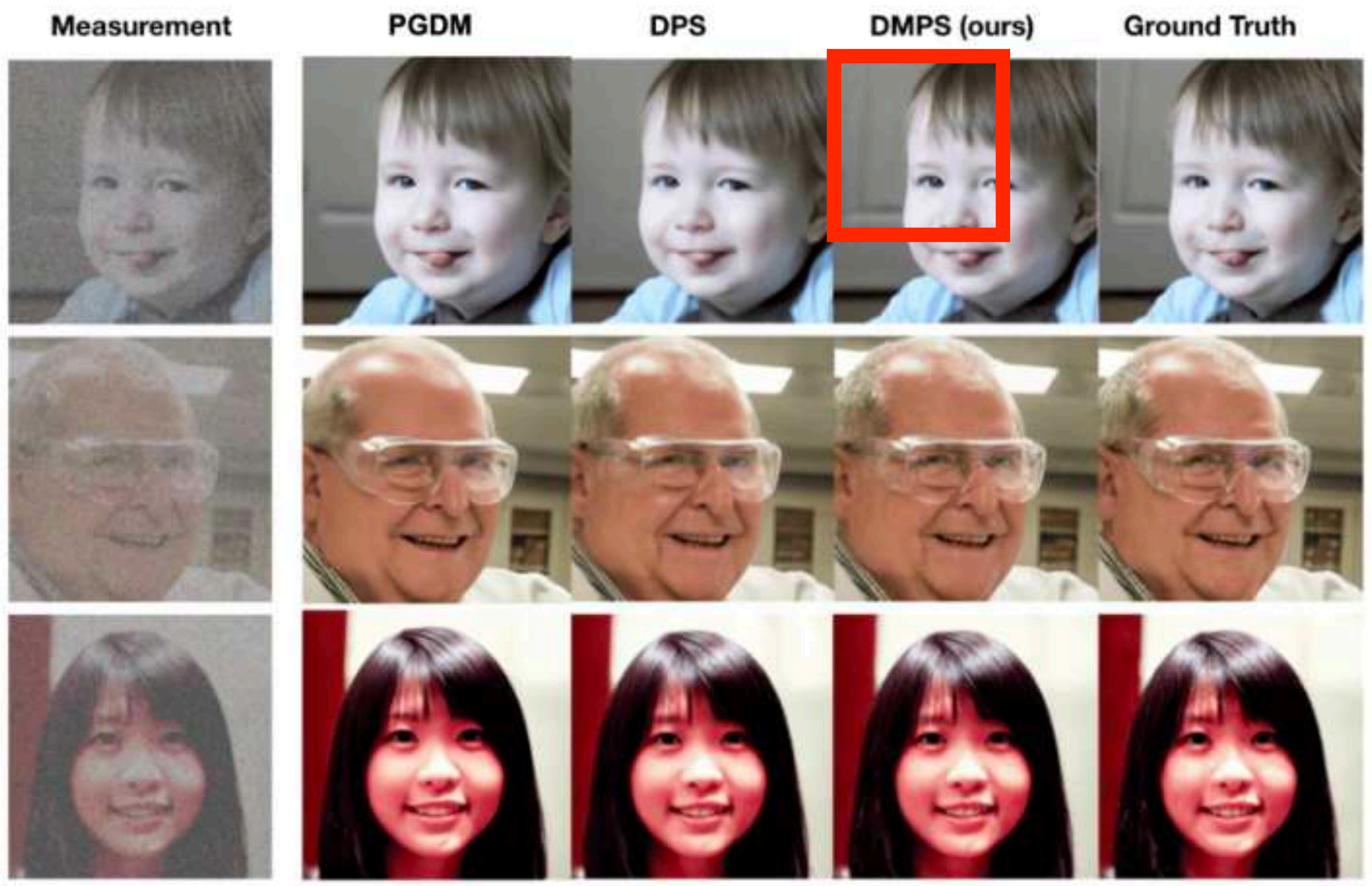
- **Experiments Results**

**Dataset: FFHQ**

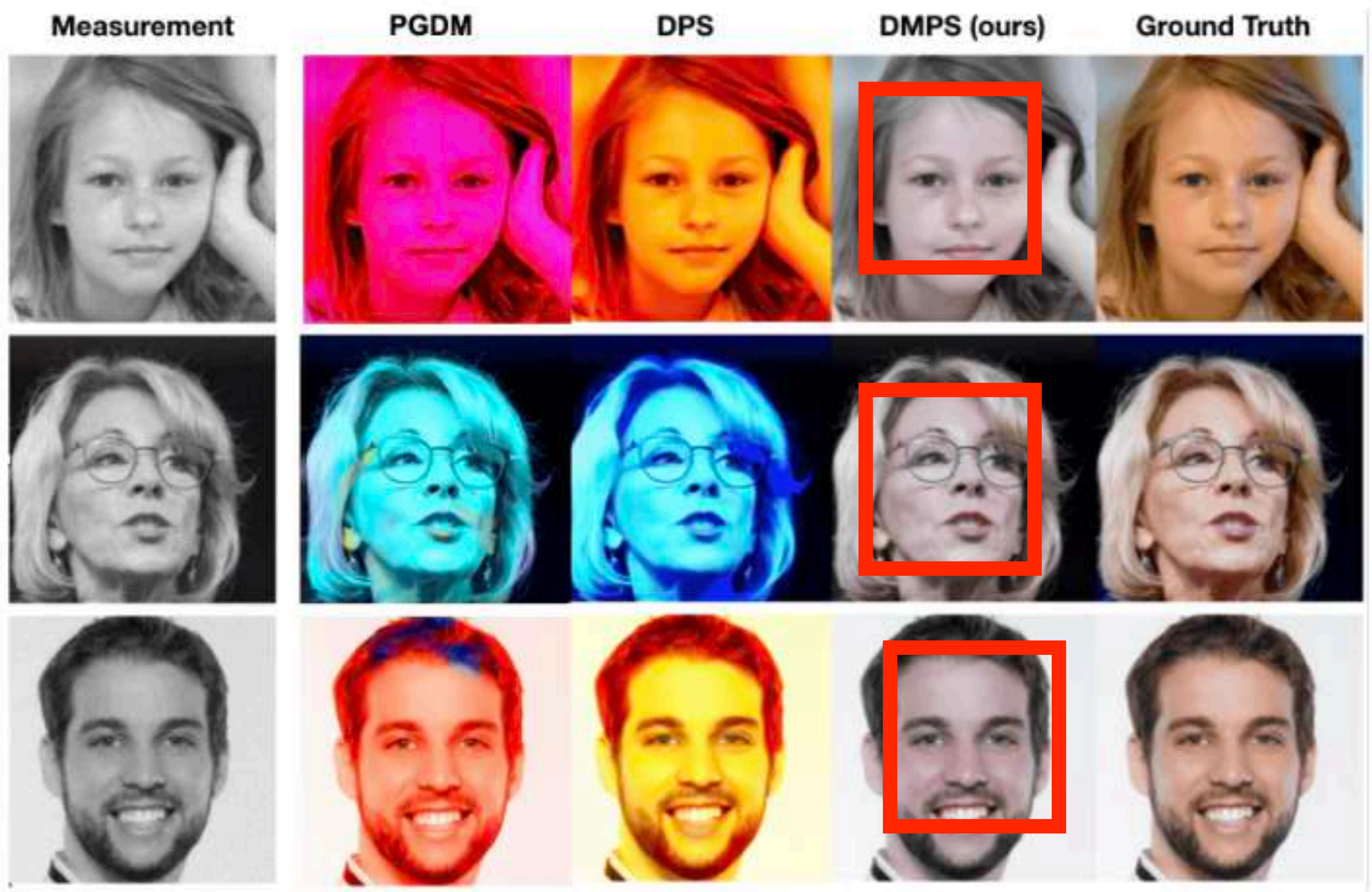
**DDPM Version**



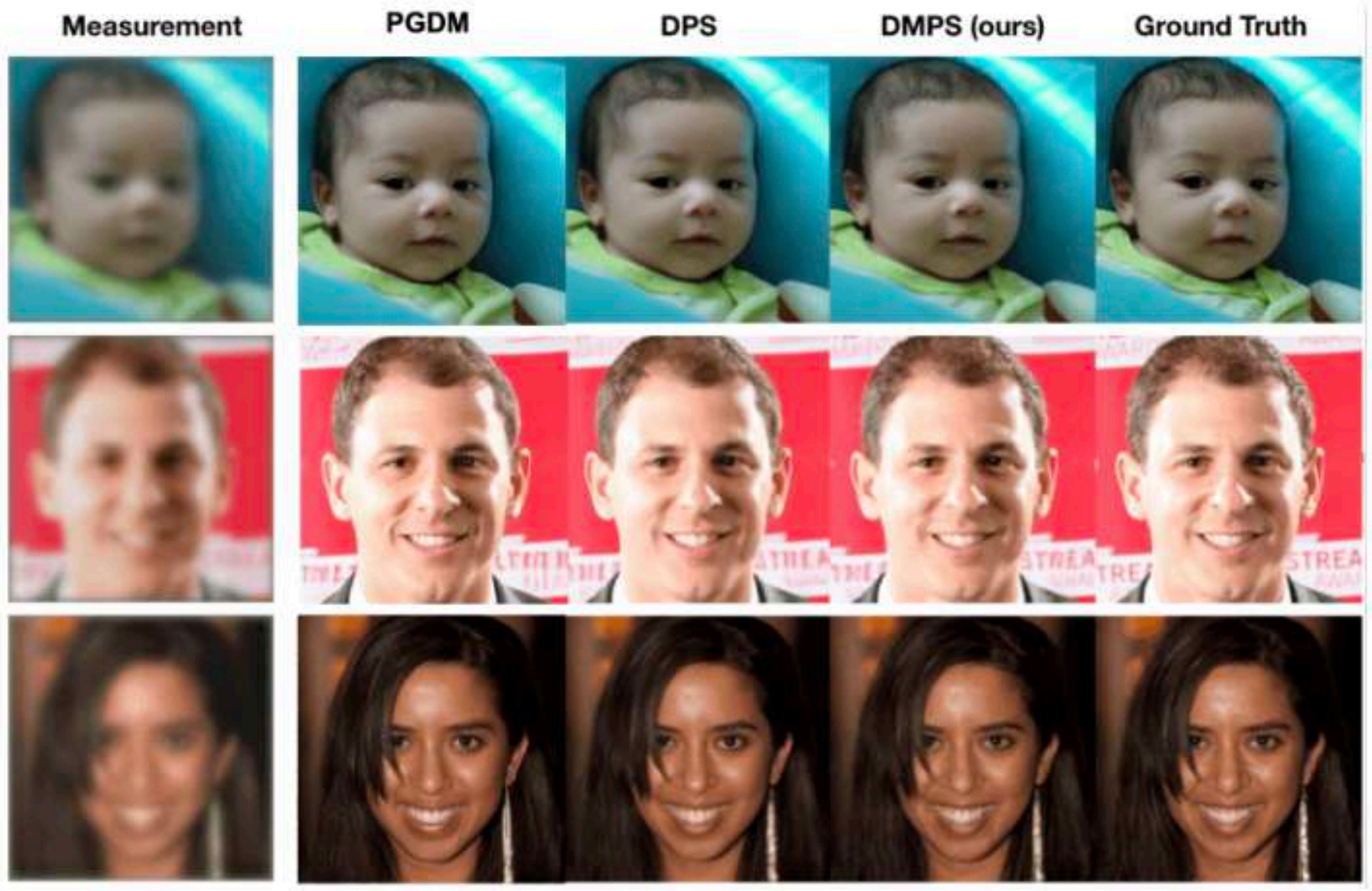

($a$) Super-resolution (SR) ($\times 4$)




($b$) Denoising ($\sigma = 0.5$)




($c$) colorization



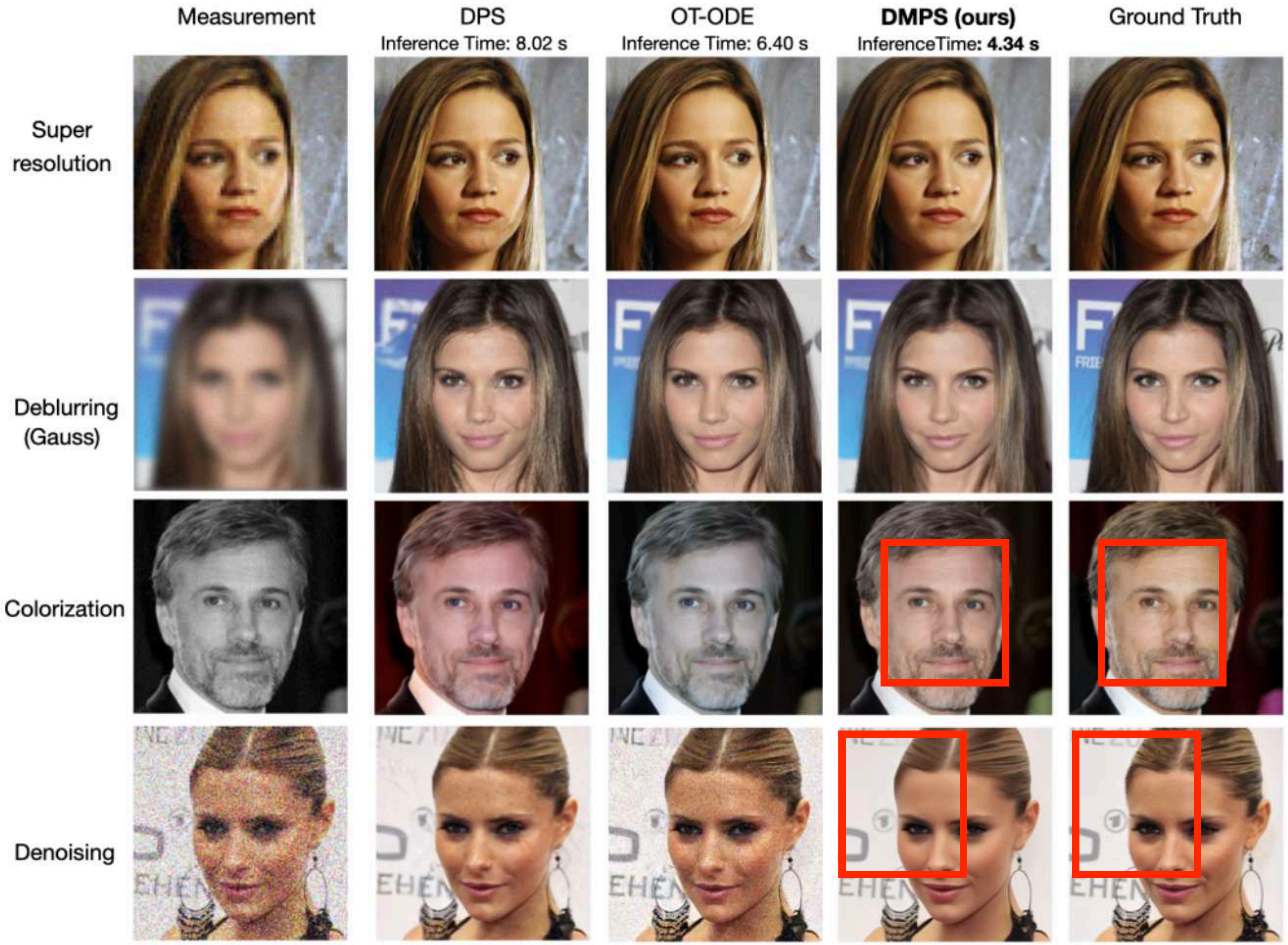

($d$) Deblurring (uniform)

# One Simple Solution: DMPS

- **Experiments Results**

**Dataset: CelebA-HQ**

**Flow-based Version**



# One Simple Solution: DMPS

- **Experiments Results**

**Dataset:** 256x 256 **FFHQ**

**Results of DDPM Version**

| Method | super-resolution | | | deblur | | | colorization | | | denoising | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PSNR ↑ | SSIM ↑ | LPIPS ↓ | PSNR ↑ | SSIM ↑ | LPIPS ↓ | PSNR ↑ | SSIM ↑ | LPIPS ↓ | PSNR ↑ | SSIM ↑ | LPIPS ↓ |
| DMPS (DDPM, ours) | **27.63** | **0.8450** | **0.2071** | **27.26** | 0.7644 | 0.2222 | **21.09** | **0.9592** | **0.2738** | **27.81** | 0.8777 | 0.2435 |
| DPS (DDPM) | 26.78 | 0.8391 | 0.2329 | 26.50 | **0.8151** | 0.2248 | 11.53 | 0.7923 | 0.5755 | 27.22 | **0.8969** | 0.2428 |
| PGDM | 27.60 | 0.8345 | 0.2077 | 26.65 | 0.7458 | **0.2196** | 12.15 | 0.8920 | 0.3969 | 27.60 | 0.8682 | **0.2425** |

**Dataset:** 256x 256 **CelebA-HQ**

**Results of Flow-based Version**

| Method | super-resolution | | | deblur | | | colorization | | | denoising | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PSNR ↑ | SSIM ↑ | LPIPS ↓ | PSNR ↑ | SSIM ↑ | LPIPS ↓ | PSNR ↑ | SSIM ↑ | LPIPS ↓ | PSNR ↑ | SSIM ↑ | LPIPS ↓ |
| DMPS (Flow-based, ours) | **28.29** | **0.8011** | 0.2329 | **26.21** | **0.7235** | 0.2637 | **23.31** | **0.8861** | **0.2901** | **29.04** | **0.8166** | **0.2821** |
| DPS (Flow-based) | 28.05 | 0.7754 | **0.2266** | 22.64 | 0.5787 | 0.3403 | 20.92 | 0.8061 | 0.3335 | 27.93 | 0.7465 | 0.2882 |
| OT-ODE | 27.71 | 0.7657 | 0.2302 | 25.84 | 0.7084 | **0.2573** | 21.67 | 0.8696 | 0.3094 | 22.76 | 0.3820 | 0.4778 |

# One Simple Solution: DMPS

■ **Experiments Results**

**Running Time of DDPM Version**

| Method | Inference Time [s] |
|---|---|
| DMPS (DDPM, ours) | **67.02** |
| DPS (DDPM) | 194.42 |
| PGDM | 182.35 |

**Running Time of Flow-based Version**

| Method | Inference Time [s] |
|---|---|
| DMPS (flow-based, ours) | **4.45** |
| DPS (flow-based) | 8.04 |
| OT-DOE | 6.44 |

**The proposed DMPS is 2-3 times faster than DPS and PGDM (OT-ODE, flow version) while achieving comparable or even better reconstruction performances**

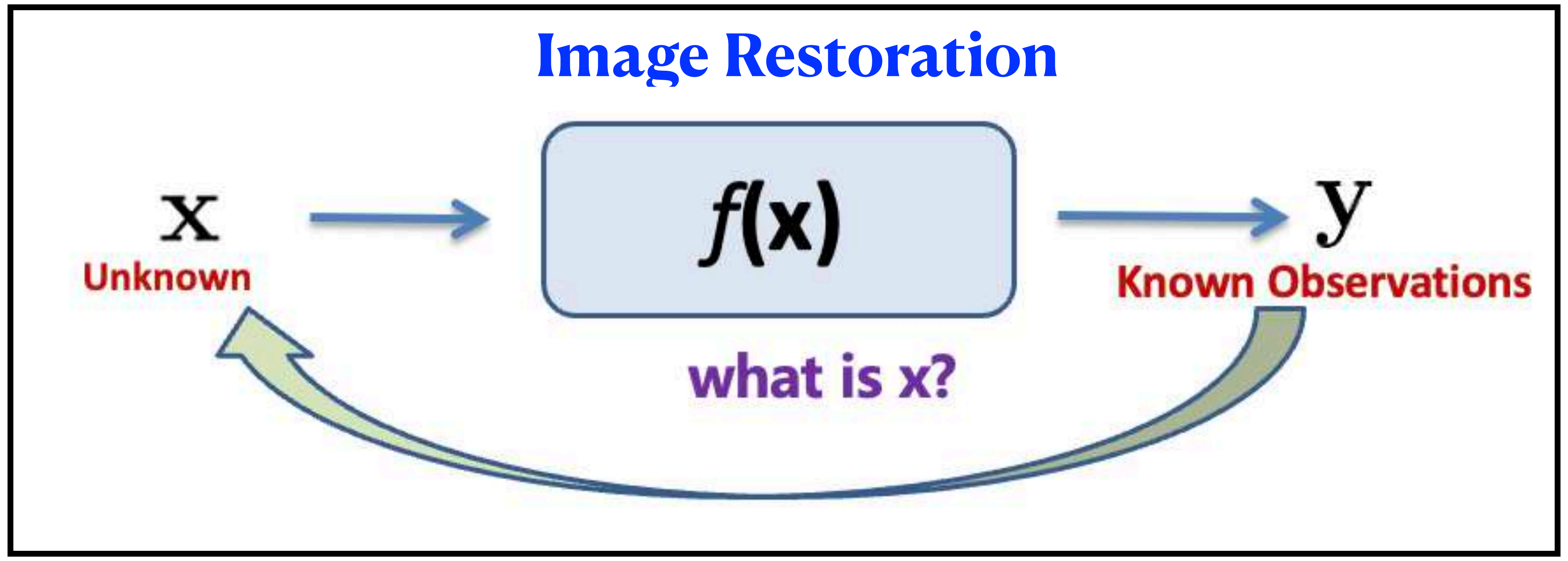
**Paper**: Meng, Xiangming, and Yoshiyuki Kabashima. "Diffusion Model Based Posterior Sampling for Noisy Linear Inverse Problems." *arXiv preprint arXiv:2211.12343v3, 2024*

**Code**: https://github.com/mengxiangming/dmps

# Contents

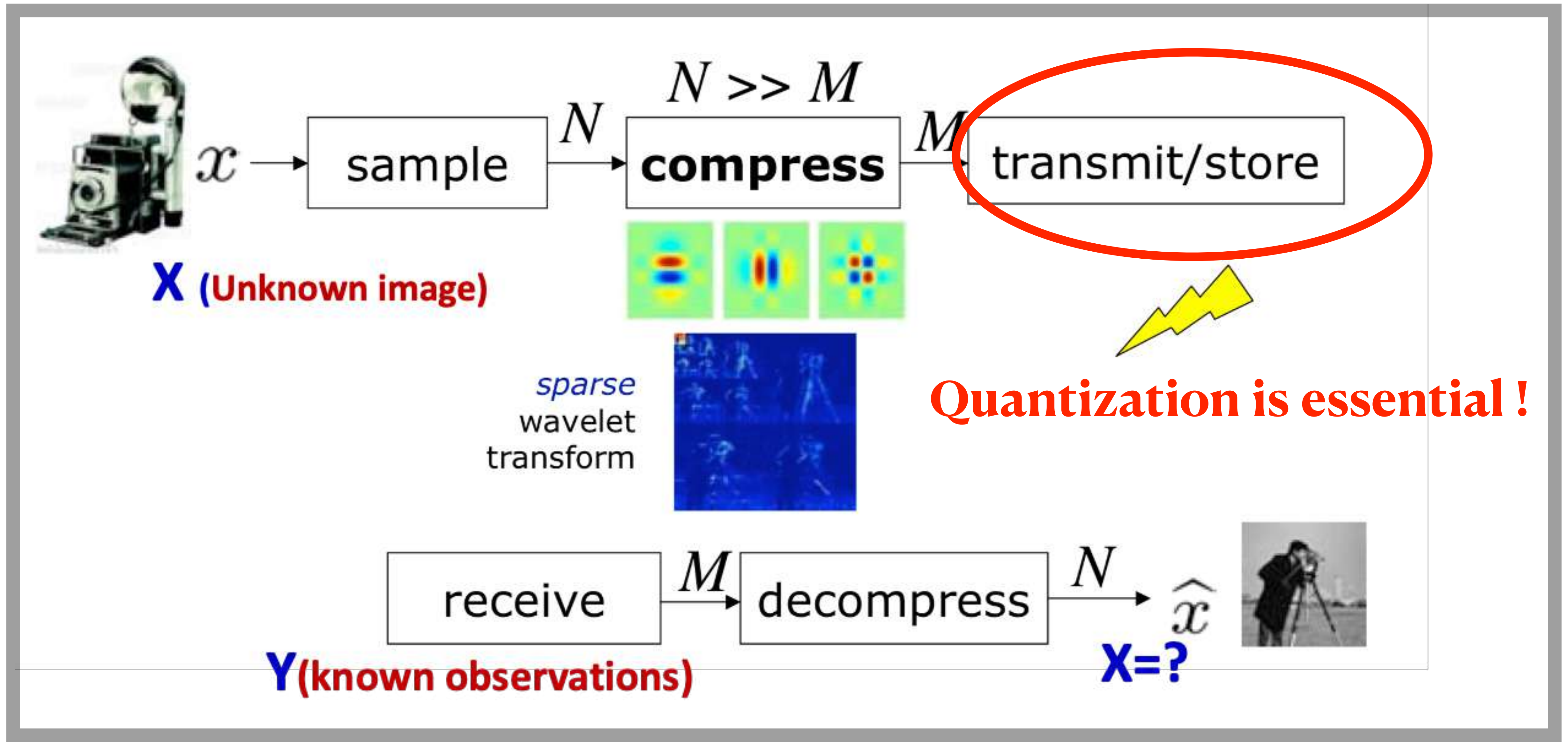
# Nonlinear Image Restoration

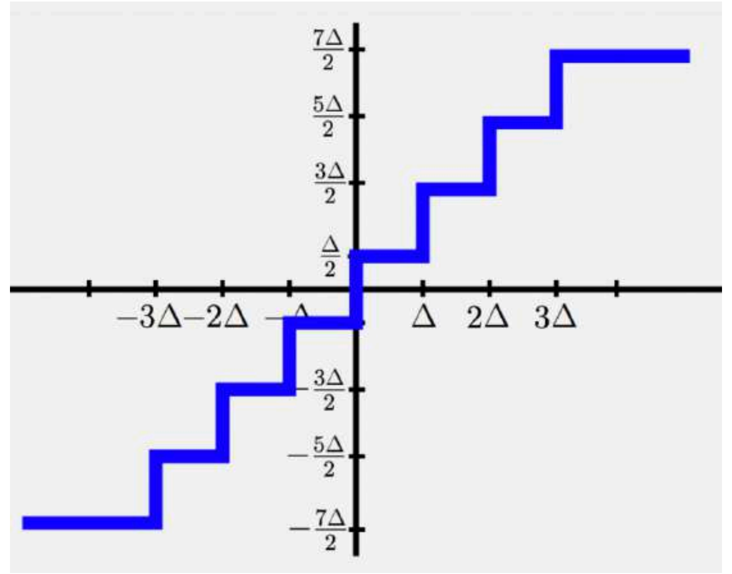
## ■ Nonlinear Image Restoration



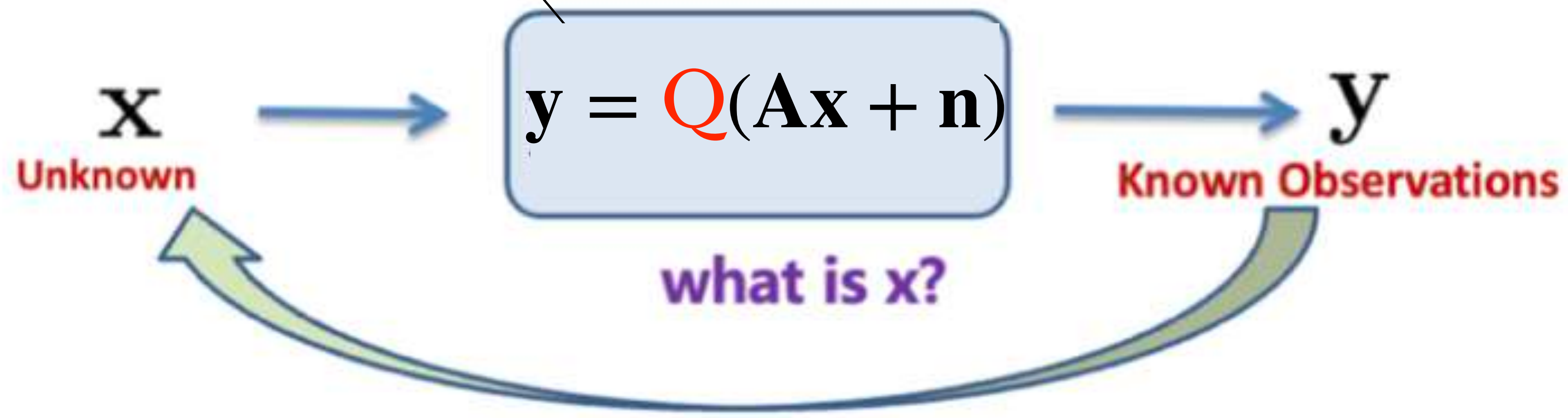

- **Linear Case**: $f(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{n}$
- **Nonlinear Case**: $f(\mathbf{x})$ is nonlinear transformation

## ■ Quantized Compressed Sensing （QCS）



Quantizer

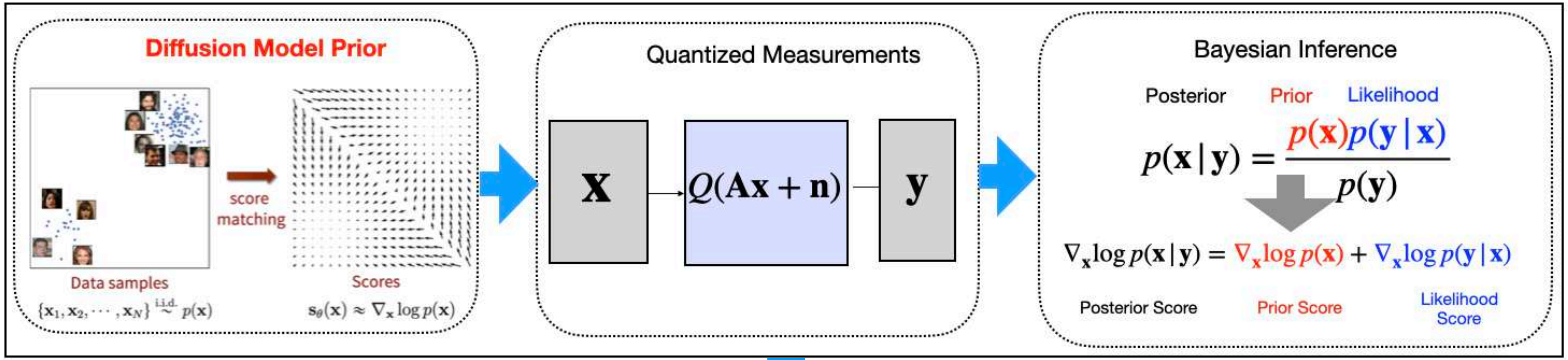


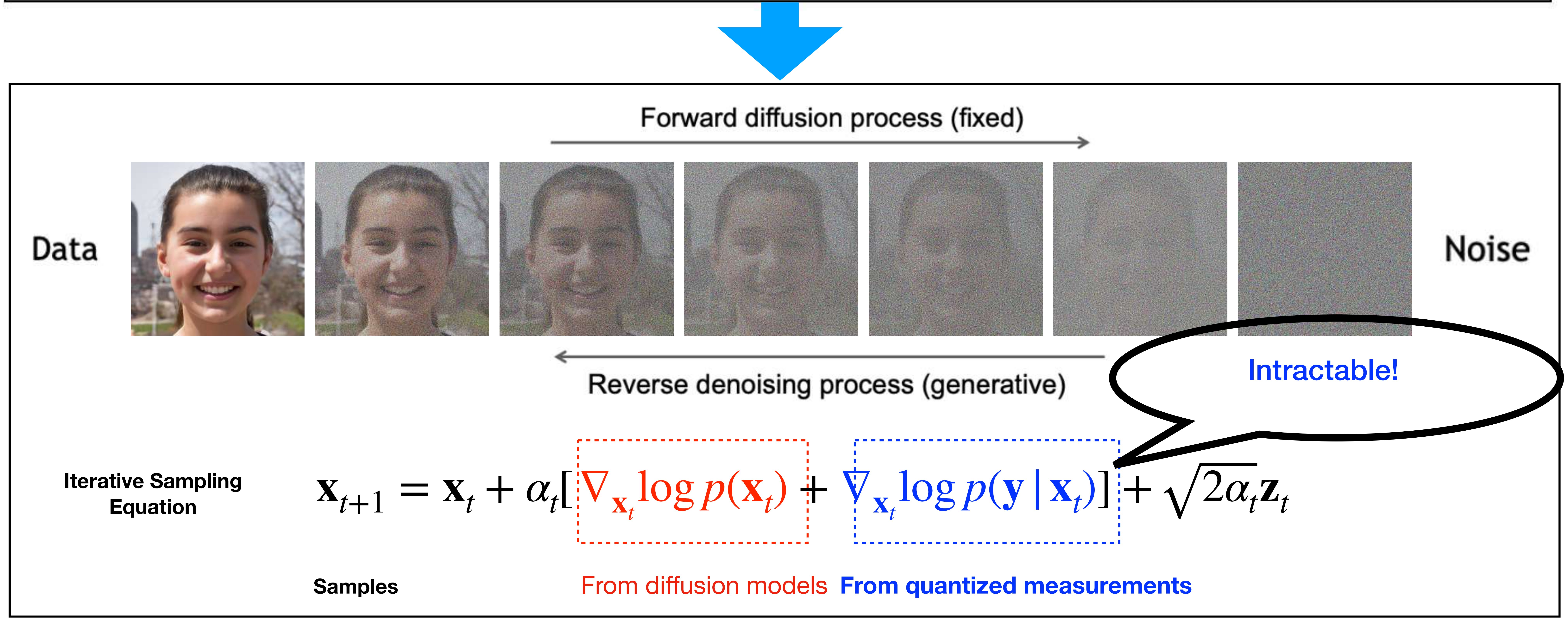

**Extreme case:** 1-bit quantization

$$\mathbf{y} = \text{sign}(\mathbf{A}\mathbf{x} + \mathbf{n})$$

# Quantized CS with Diffusion Models

- **Basic Idea**



# QCS-SGM: Quantized CS with SGM

■ **Two Assumptions of QCS-SGM**

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

non-Gauss Intractable

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

**More difficult to obtain closed-form approximation**

- **Assumption 1**

The prior $p(\mathbf{x}_0)$ is non-informative w.r.t. $p(\mathbf{x}_t \mid \mathbf{x}_0)$

$$p(\mathbf{x}_t \mid \mathbf{x}_0) \propto p(\mathbf{x}_0 \mid \mathbf{x}_t)$$

Unlike linear case, Assumption 1 alone does **not** yield closed-form $p(\mathbf{y} \mid \mathbf{x}_t)$

- **Assumption 2**

The sensing matrix **A** is row-orthogonal, i.e.,

$$\mathbf{A}\mathbf{A}^T = \text{Diagonal matrix}$$

**(Approximately) satisfied by many popular CS matrices**
**e.g., DFT, DCT, Hadamard, and random Gaussian matrices, etc.**

# QCS-SGM: Quantized CS with SGM

## ■ Results of Pseudo-likelihood Score

- **Theorem 1**: Under assumptions 1 and 2, we obtain a closed-form solution to the likelihood score

$$\nabla_{\mathbf{x}_t} \log p(\mathbf{y} \mid \mathbf{x}_t) = \mathbf{A}^T \mathbf{G}(\beta_t, \mathbf{y}, \mathbf{A}, \mathbf{x}_t)$$

where

$$\mathbf{G}(\beta_t, \mathbf{y}, \mathbf{A}, \mathbf{x}_t) = [g_1, g_2, \ldots, g_M]^T \in \mathbb{R}^{M \times 1}$$

$$g_m = \frac{\exp\left(-\frac{\tilde{u}_{y_m}^2}{2}\right) - \exp\left(-\frac{\tilde{l}_{y_m}^2}{2}\right)}{\sqrt{\sigma^2 + \beta_t^2 \left\| \mathbf{a}_m^T \right\|_2^2} \int_{\tilde{l}_{y_m}}^{\tilde{u}_{y_m}} \exp\left(-\frac{t^2}{2}\right) dt} \quad \tilde{u}_{y_m} = \frac{\mathbf{a}_m^T \mathbf{x}_t - u_{y_m}}{\sqrt{\sigma^2 + \beta_t^2 \left\| \mathbf{a}_m^T \right\|_2^2}} \quad \tilde{l}_{y_m} = \frac{\mathbf{a}_m^T \mathbf{x}_t - l_{y_m}}{\sqrt{\sigma^2 + \beta_t^2 \left\| \mathbf{a}_m^T \right\|_2^2}}$$

- **Corollary**: In the special case of standard CS

$$\nabla_{\mathbf{x}_t} \log p(\mathbf{y} \mid \mathbf{x}_t) = \mathbf{A}^T \left(\sigma^2 \mathbf{I} + \beta_t^2 \mathbf{A}\mathbf{A}^T\right)^{-1} \left(\mathbf{y} - \mathbf{A}\mathbf{x}_t\right)$$

✓Explain the necessity of annealing term in Jalal et al. (2021a)

$$\nabla_{\mathbf{x}_t} \log p(\mathbf{y} \mid \mathbf{x}_t) = \frac{\mathbf{A}^T \left(\mathbf{y} - \mathbf{A}\mathbf{x}_t\right)}{\sigma^2 + \gamma_t^2}$$

✓Extend and improve Jalal et al. (2021a) in the general case

# QCS-SGM: Quantized CS with SGM

■ **Resultant Algorithm**

**Algorithm 1:** Quantized Compressed Sensing with SGM (QCS-SGM)

**Input:** $\{\beta_t\}_{t=1}^{T}, \epsilon, K, \mathbf{y}, \mathbf{A}, \sigma^2$, quantization codewords $\mathcal{Q}$ and thresholds $\{[l_q, u_q)|q \in \mathcal{Q}\}$

**Initialization:** $\mathbf{x}_1^0 \sim \mathcal{U}(0,1)$

**for** $t = 1$ *to* $T$ **do**

  $\alpha_t \leftarrow \epsilon \beta_t^2 / \beta_T^2$

  **for** $k = 1$ *to* $K$ **do**

    Draw $\mathbf{z}_t^k \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

    Compute $\mathbf{G}(\beta_t, \mathbf{y}, \mathbf{A}, \mathbf{x}_t^{k-1})$ as (12) (or (15) for 1-bit)

    $\mathbf{x}_t^k = \mathbf{x}_t^{k-1} + \alpha_t \left[ s_{\boldsymbol{\theta}}(\mathbf{x}_t^{k-1}, \beta_t) + \mathbf{A}^T \mathbf{G}(\beta_t, \mathbf{y}, \mathbf{A}, \mathbf{x}_t^{k-1}) \right] + \sqrt{2\alpha_t}\mathbf{z}_t^k$

  $\mathbf{x}_{t+1}^0 \leftarrow \mathbf{x}_t^K$

**Output:** $\hat{\mathbf{x}} = \mathbf{x}_T^K$

Only this term is different from SGM!

**Paper**: Meng, Xiangming, and Yoshiyuki Kabashima. "Quantized Compressed Sensing with Score-Based Generative Models." **ICLR 2023**

**Code**: https://github.com/mengxiangming/QCS-SGM

# QCS-SGM: Quantized CS with SGM

- **Experimental Results**


(a) MNIST, $M = 200, \sigma = 0.05$ (b) CelebA, $M = 4000, \sigma = 0.001$

The proposed QCS-SGM achieves remarkably better performances

# QCS-SGM: Quantized CS with SGM

## ■ Experimental Results

Results of QCS-SGM on CelebA
in the **fixed budget** case
(Q×M = 12288)

(a) Ground Truth

(b) 1-bit, $M = 12288$

(c) 2-bit, $M = 6144$

(d) 3-bit, $M = 4096$

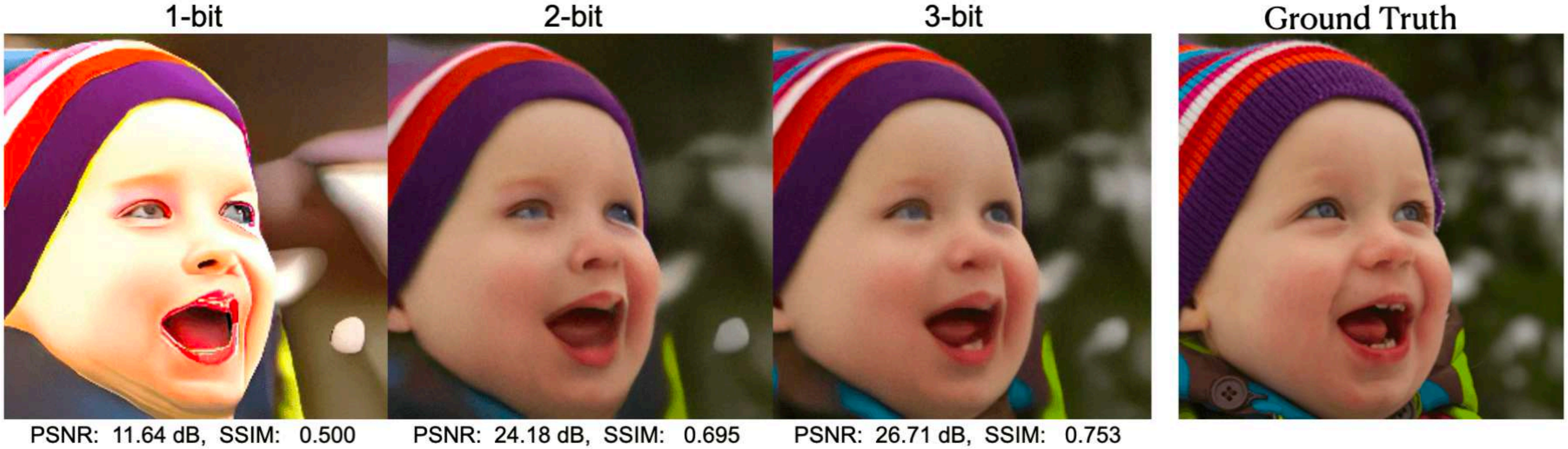
# QCS-SGM: Quantized CS with SGM

- **Experimental Results**

FFHQ $256 \times 256$ high-resolution images

**Compression Ratio** $\frac{M}{N} = \frac{1}{8} \ll 1$

$M = \frac{1}{8}N$




The proposed QCS-SGM can well recover high-resolution image from only a few low-resolution (1,2,3-bit) quantized measurements

# QCS-SGM+: Improved Quantized CS with SGM

■ **Limitation of QCS-SGM**

QCS-SGM is limited to
(approximately) row-orthogonal matrices A

**Why? The pseudo-likelihood is otherwise intractable**

$$\mathbf{C}_t^{-1} = \sigma^2 \mathbf{I} + \beta_t^2 \mathbf{A}\mathbf{A}^T$$

$$p(\mathbf{y}|\mathbf{x}_t) \simeq \tilde{p}(\mathbf{y}|\mathbf{z}_t = \mathbf{A}\mathbf{x}_t) = \int \prod_{m=1}^{M} \mathbb{1}\left((z_{t,m} + \tilde{n}_{t,m}) \in \mathbf{Q}^{-1}(y_m)\right) \mathcal{N}(\tilde{\mathbf{n}}_t; \mathbf{0}, \mathbf{C}_t^{-1}) d\tilde{\mathbf{n}}_t$$

Intractable integration

# QCS-SGM+: Improved Quantized CS with SGM

- **A New Perspective**

**pseudo-likelihood**

$$\underbrace{p(\mathbf{y}|\mathbf{x}_t) \simeq \tilde{p}(\mathbf{y}|\mathbf{z}_t = \mathbf{A}\mathbf{x}_t)}_{\text{Partition Function (normalization term)}} = \int \prod_{m=1}^{M} \mathbb{1}\underbrace{\left((z_{t,m} + \tilde{n}_{t,m}) \in \mathbf{Q}^{-1}(y_m)\right)}_{\text{Likelihood}} \underbrace{\mathcal{N}(\tilde{\mathbf{n}}_t; \mathbf{0}, \mathbf{C}_t^{-1})}_{\text{Prior}} d\tilde{\mathbf{n}}_t$$

One fundamental Problem in Bayesian Inference

**The pseudo-likelihood can be viewed as the partition function of random variables $\tilde{\mathbf{n}}_t$**

# QCS-SGM+: Improved Quantized CS with SGM

## ■ A New Perspective

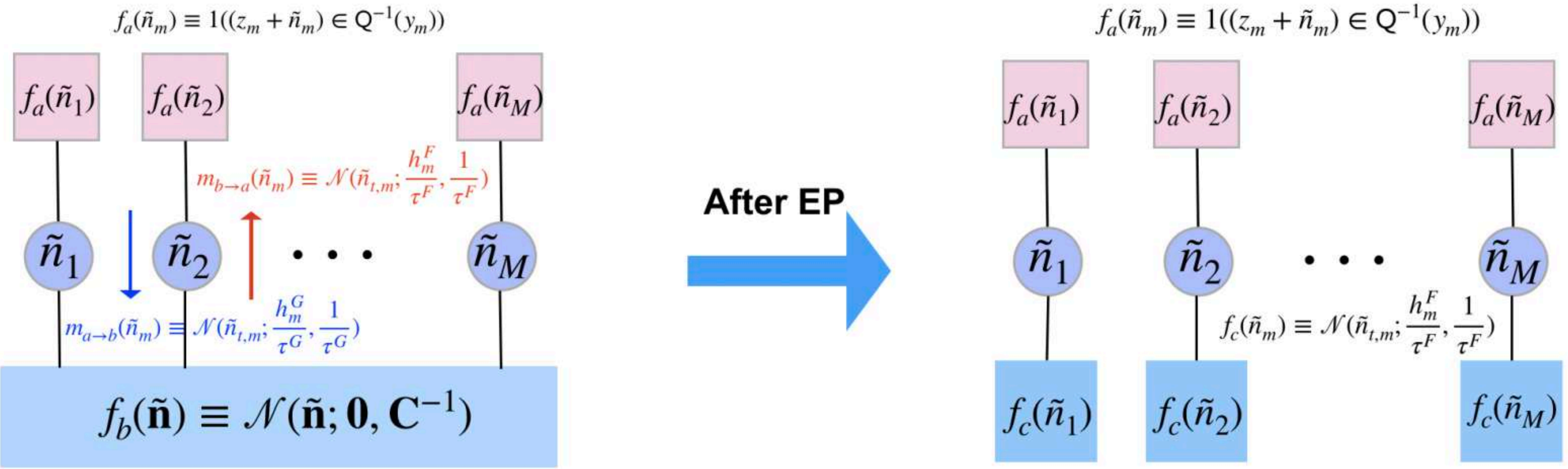
**pseudo-likelihood**

$$\underbrace{p(\mathbf{y}|\mathbf{x}_t) \simeq \tilde{p}(\mathbf{y}|\mathbf{z}_t = \mathbf{A}\mathbf{x}_t)}_{\text{Partition Function (normalization term)}} = \int \prod_{m=1}^{M} \underbrace{\mathbb{1}\left((z_{t,m} + \tilde{n}_{t,m}) \in \mathrm{Q}^{-1}(y_m)\right)}_{\text{Likelihood}} \underbrace{\mathcal{N}(\tilde{\mathbf{n}}_t; \mathbf{0}, \mathbf{C}_t^{-1})}_{\text{Prior}} d\tilde{\mathbf{n}}_t$$

One fundamental Problem in Bayesian Inference

**The pseudo-likelihood can be viewed as the partition function of random variables $\tilde{\mathbf{n}}_t$**

Resort to the famous expectation propagation (EP) Tom Minka 2001




(a) Original factor graph

(b) Factor graph after EP

# QCS-SGM+: Improved Quantized CS with SGM

- **QCS-SGM+**

---

**Algorithm 1:** QCS-SGM+

---

**Input:** $\{\beta_t\}_{t=1}^{T}$, $\epsilon$, $\gamma$, $IterEP$, $K$, $\mathbf{y}$, $\mathbf{A}$, $\sigma^2$, quantization thresholds $\{[l_q, u_q) | q \in \mathcal{Q}\}$

**Initialization:** $\mathbf{x}_1^0 \sim \mathcal{U}(0, 1)$

**for** $t = 1$ ***to*** $T$ **do**

  $\alpha_t \leftarrow \epsilon \beta_t^2 / \beta_T^2$

  **for** $k = 1$ ***to*** $K$ **do**

    Draw $\mathbf{z}_t^k \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

    **Initialization:** $\boldsymbol{h}^F, \tau^F, \boldsymbol{h}^G, \tau^G$

    **for** $it = 1$ ***to*** $IterEP$ **do**

      $\boldsymbol{h}^G = \frac{\boldsymbol{m}^a}{\chi^a} - \boldsymbol{h}^F$

      $\tau^G = \frac{1}{\chi^a} - \tau^F$

      $\boldsymbol{h}^F = \frac{\boldsymbol{m}^b}{\chi^b} - \boldsymbol{h}^G$

      $\tau^F = \frac{1}{\chi^b} - \tau^G$

    Compute $\nabla_{\mathbf{x}_t} \log p_{\beta_t}(\mathbf{y} \mid \mathbf{x}_t)$ as (11)

    $\mathbf{x}_t^k = \mathbf{x}_t^{k-1} + \alpha_t \left[ \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t^{k-1}, \beta_t) + \gamma \nabla_{\mathbf{x}_t} \log p_{\beta_t}(\mathbf{y} \mid \mathbf{x}_t) \right] + \sqrt{2\alpha_t} \mathbf{z}_t^k$

  $\mathbf{x}_{t+1}^0 \leftarrow \mathbf{x}_t^K$

**Output:** $\hat{\mathbf{x}} = \mathbf{x}_T^K$

---

Running EP to approximate the pseudo-likelihood

**Paper**: Meng, Xiangming, and Yoshiyuki Kabashima. "QCM-SGM+: Improved Quantized Compressed Sensing With Score-Based Generative Models." (AAAI 2024)

**Code**: https://github.com/mengxiangming/QCS-SGM-plus

# QCS-SGM+: Improved Quantized CS with SGM

■ **Experimental Results**

- General Matrices

(a) ill-conditioned matrices

$$\mathbf{A} = \mathbf{V}\boldsymbol{\Sigma}\mathbf{U}^{\mathrm{T}}$$

$\mathbf{V}$ and $\mathbf{U}$ are independent Harr-distributed matrices

nonzero singular values of $\mathbf{A}$ satisfy $\frac{\lambda_i}{\lambda_{i+1}} = \kappa^{1/M}$, where $\kappa$ is the condition number.

(b) correlated matrices

$$\mathbf{A} = \mathbf{R}_L \mathbf{H} \mathbf{R}_R$$

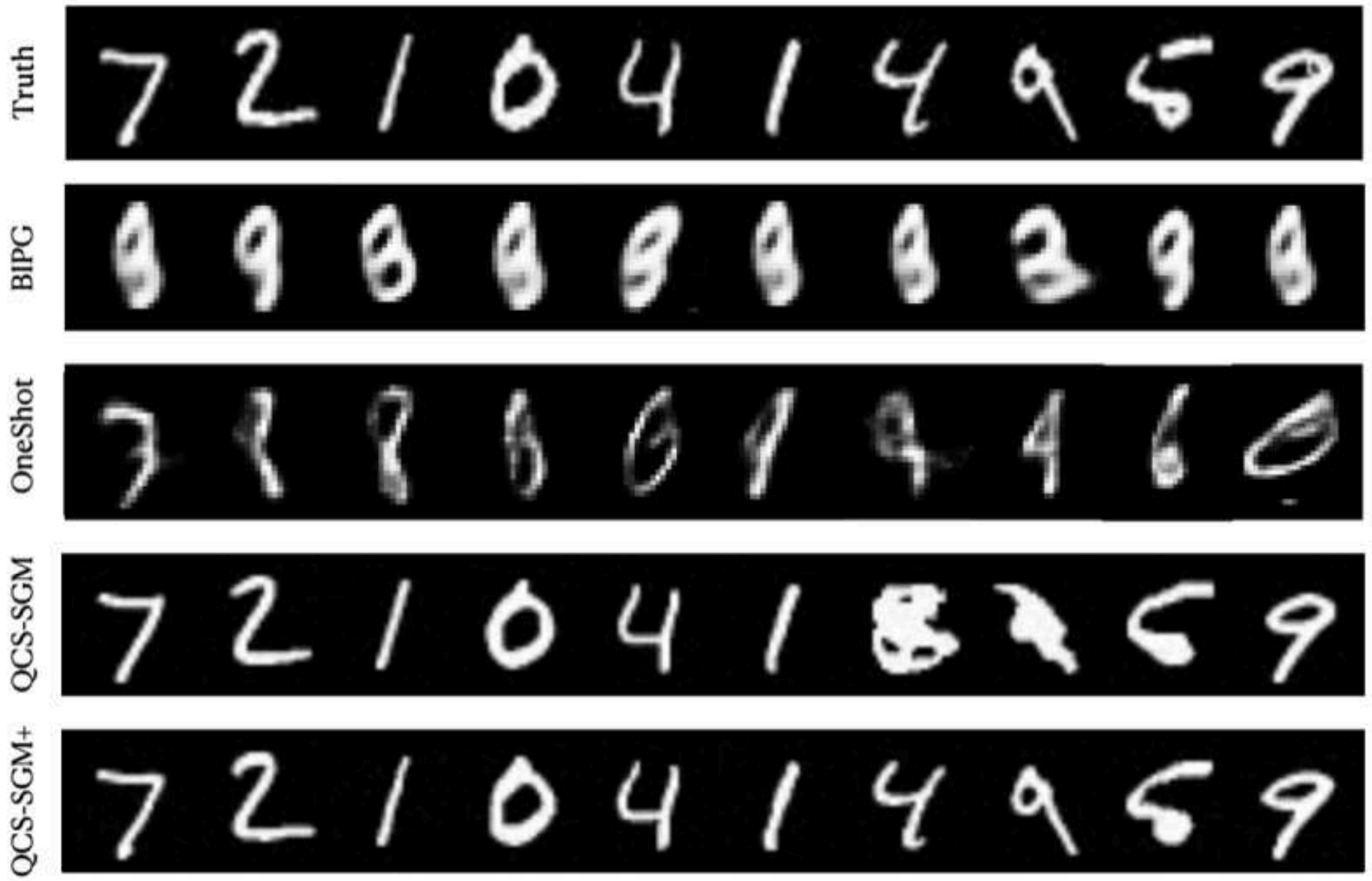
where $\mathbf{R}_L = \mathbf{R}_1^{\frac{1}{2}} \in \mathbb{R}^{M\times M}$ and $\mathbf{R}_R = \mathbf{R}_2^{\frac{1}{2}} \in \mathbb{R}^{N\times N}$, $\mathbf{H} \in \mathbb{R}^{M\times N}$ is a random matrix

The (i, j) th element of both R1 and R2 is $\rho^{|i-j|}$ and $\rho$ is termed the correlation coefficient

# QCS-SGM+: Improved Quantized CS with SGM

- **Experimental Results**

1-bit CS on MNIST and CelebA for ill-conditioned A ($\kappa = 10^3$ for MNIST and $\kappa = 10^6$ for CelebA)



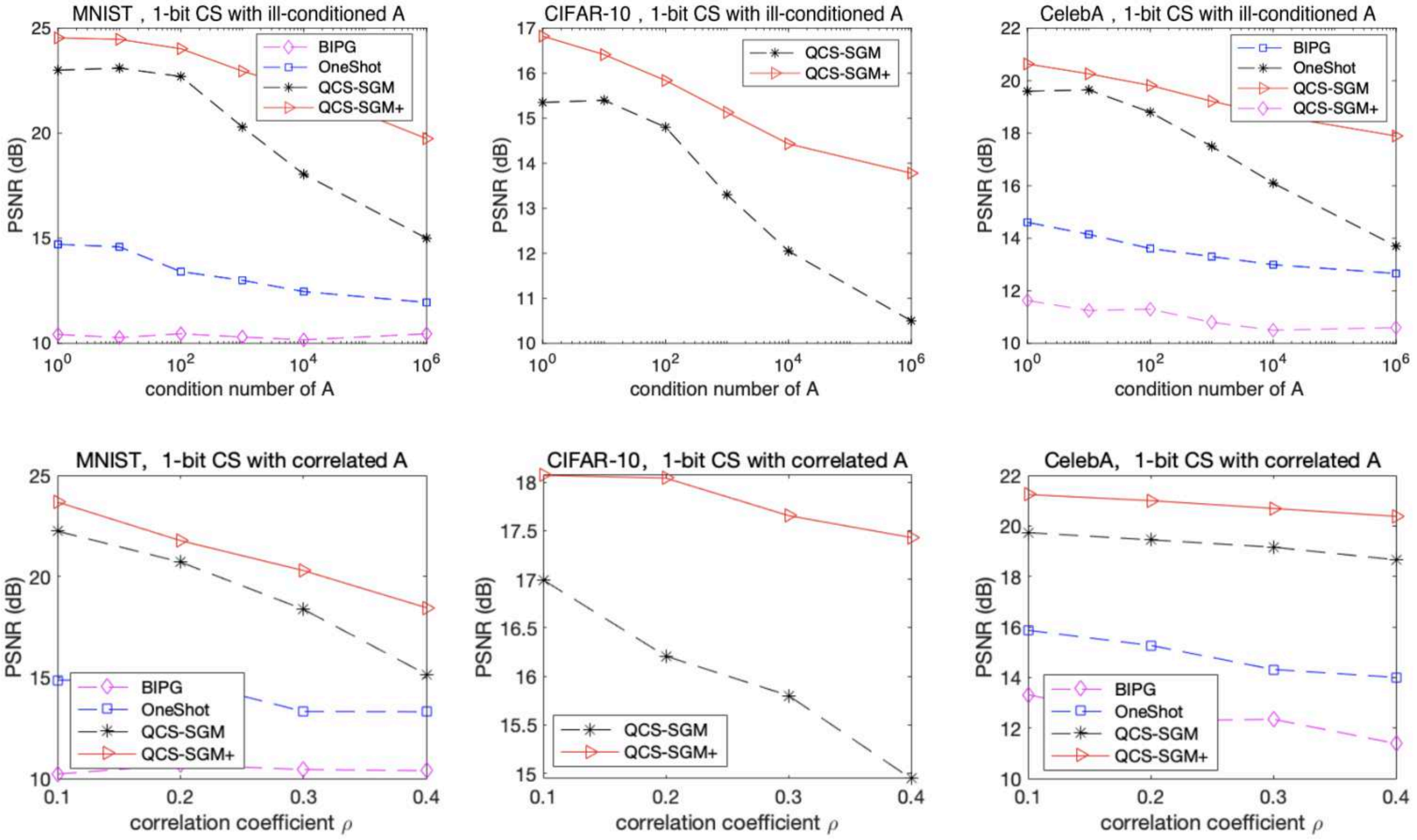

(a) MNIST, $M = 400, \sigma = 0.05, \kappa = 10^3$


(b) CelebA, $M = 4000, \sigma = 0.001, \kappa = 10^6$

**It can be seen that QCS-SGM+ apparently outperforms the original QCS-SGM and other methods.**

# QCS-SGM+: Improved Quantized CS with SGM

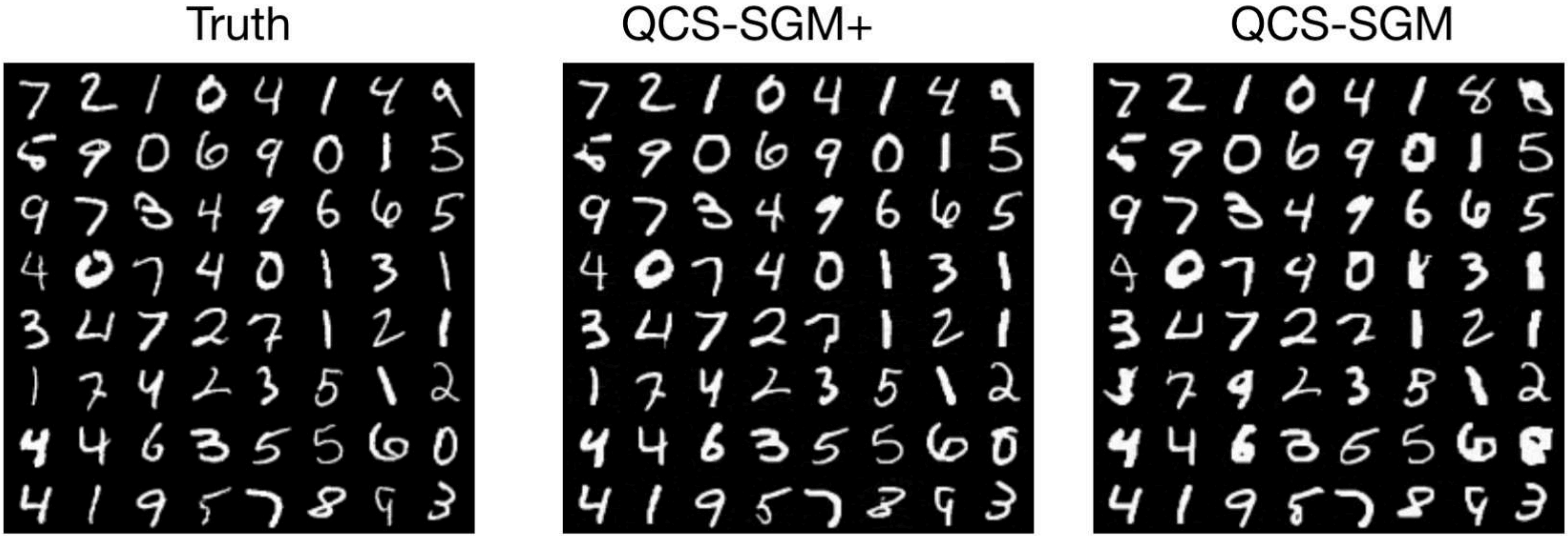
■ **Experimental Results**




**It can be seen that QCS-SGM+ apparently outperforms the original QCS-SGM and other methods.**

# QCS-SGM+: Improved Quantized CS with SGM

■ **Experimental Results**



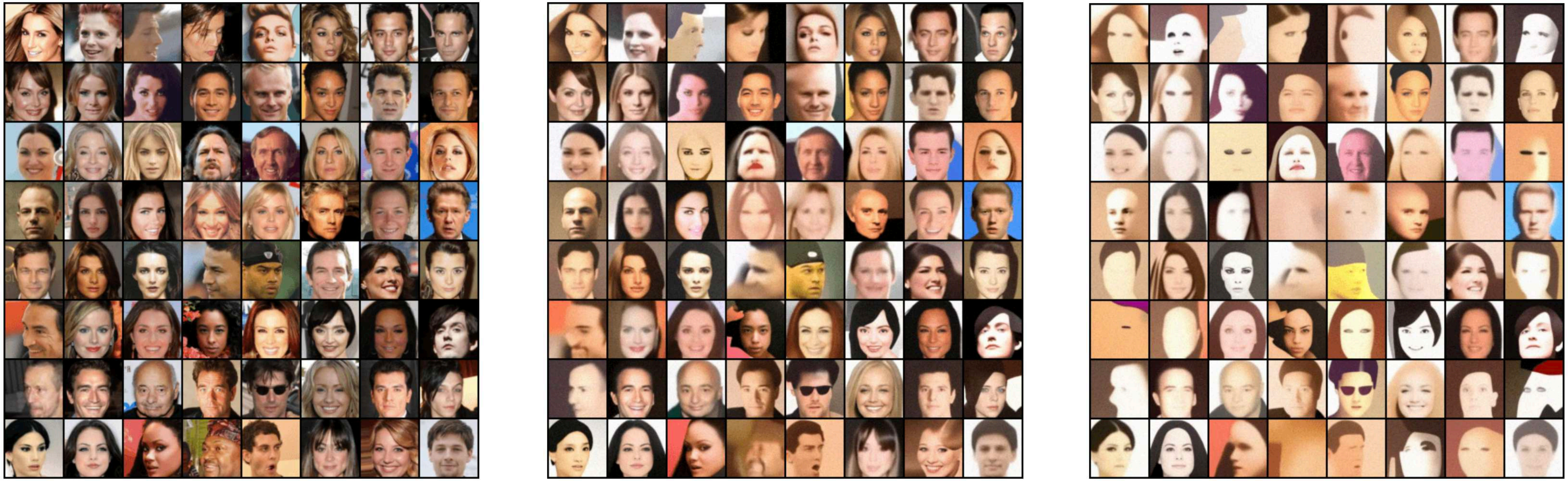

(b) 1-bit CS with correlated $\mathbf{A}, \rho = 0.4, M = 400, \sigma = 0.1$

**It can be seen that QCS-SGM+ apparently outperforms the original QCS-SGM and other methods.**

# QCS-SGM+: Improved Quantized CS with SGM

■ **Experimental Results**

Truth QCS-SGM+ QCS-SGM



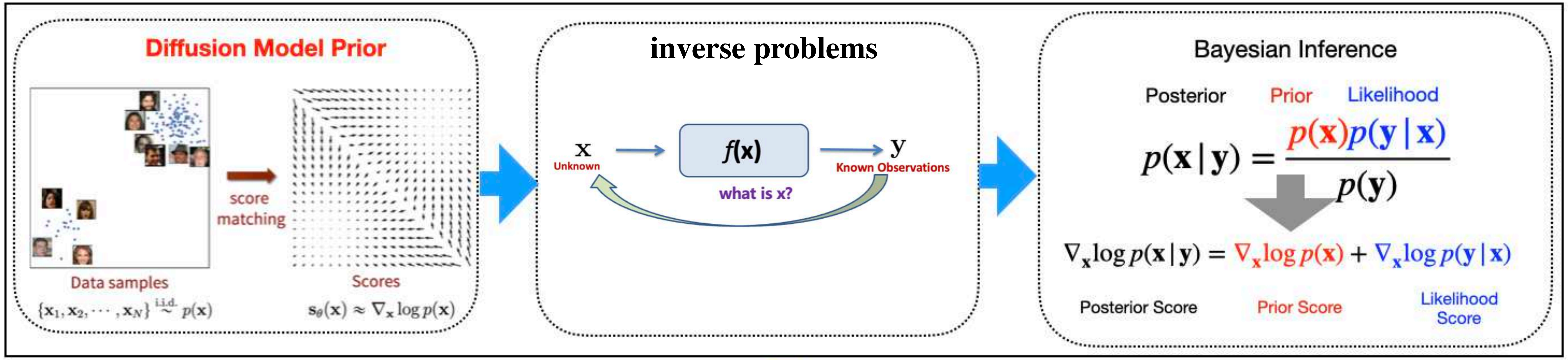
1-bit CS on CelebA for ill-conditioned A ( $\kappa = 10^6$ for CelebA), $M = 4000 \ll N, \sigma = 0.1$

**It can be seen that QCS-SGM+ apparently outperforms the original QCS-SGM.**

# Summary

■ **Generative Image Restoration**

**Image Restoration (linear and nonlinear) with Diffusion Models**




- **Linear case: DMPS for general noisy linear inverse problems**
- **Nonlinear case: QCS-SGM/QCS-SGM+ for quantized compressed sensing**

**For more details, please refer to my personal page (个人主页)：https://mengxiangming.github.io/**

**Paper**: Meng, Xiangming, and Yoshiyuki Kabashima. "Diffusion Model Based Posterior Sampling for Noisy Linear Inverse Problems." *arXiv preprint arXiv:2211.12343v2*(2023)
**Paper**: Meng, Xiangming, and Yoshiyuki Kabashima. "Quantized Compressed Sensing with Score-Based Generative Models." **ICLR 2023**
**Paper**: Meng, Xiangming, and Yoshiyuki Kabashima. "QCM-SGM+: Improved Quantized Compressed Sensing With Score-Based Generative Models." *AAAI 2024*
**Code**: https://github.com/mengxiangming/dmps
**Code**: https://github.com/mengxiangming/QCS-SGM
**Code**: https://github.com/mengxiangming/QCS-SGM-plus

Thank you!

Q&A