Score-based generative models and diffusion models

Why denoising is so powerful?

Rabiul Awal
5 Sep 2025

Slides adapted from Stefano Ermon

A really long agenda

- Challenges of generative modelling [5 mins]
- Score matching [15 mins]
- Score-based generative models [20 mins]
- Diffusion models as score-based model [20 mins]
- Diffusion via SDE [10 mins]
- Probability flow ODE [10 mins]
- Quiz + QA [20 mins]
- Interesting research problems (optional)

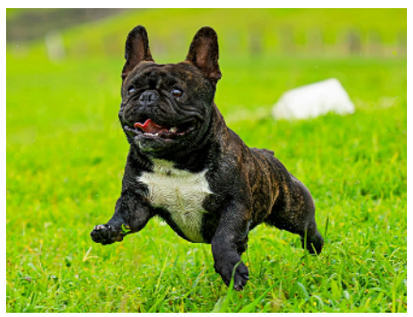
Outline

- Score function
- Score matching
- Score-based models
- Denoising score matching
- Noise Conditioned Score networks
- Sampling as iterative denoising

Estimating the probability distribution of data

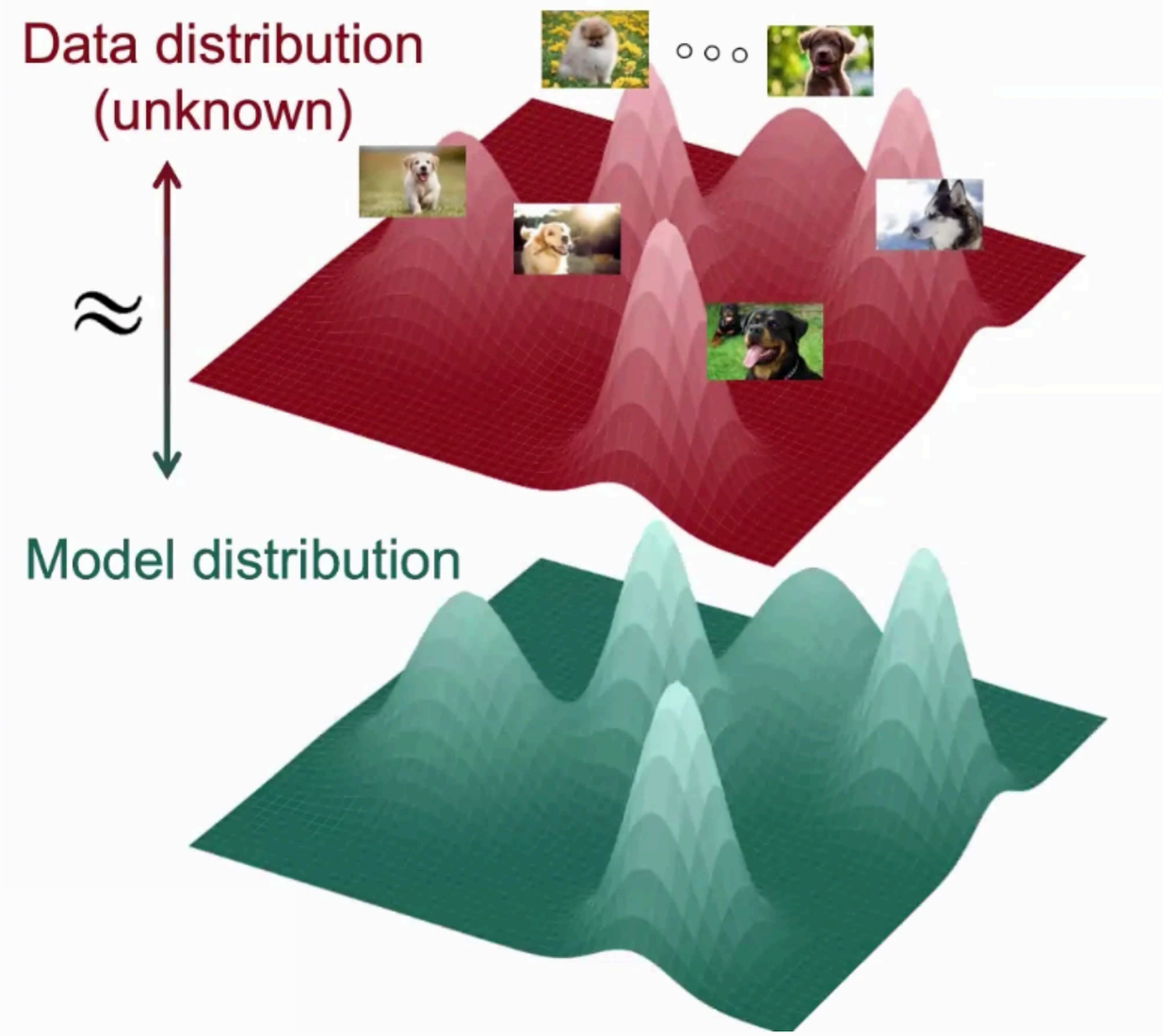




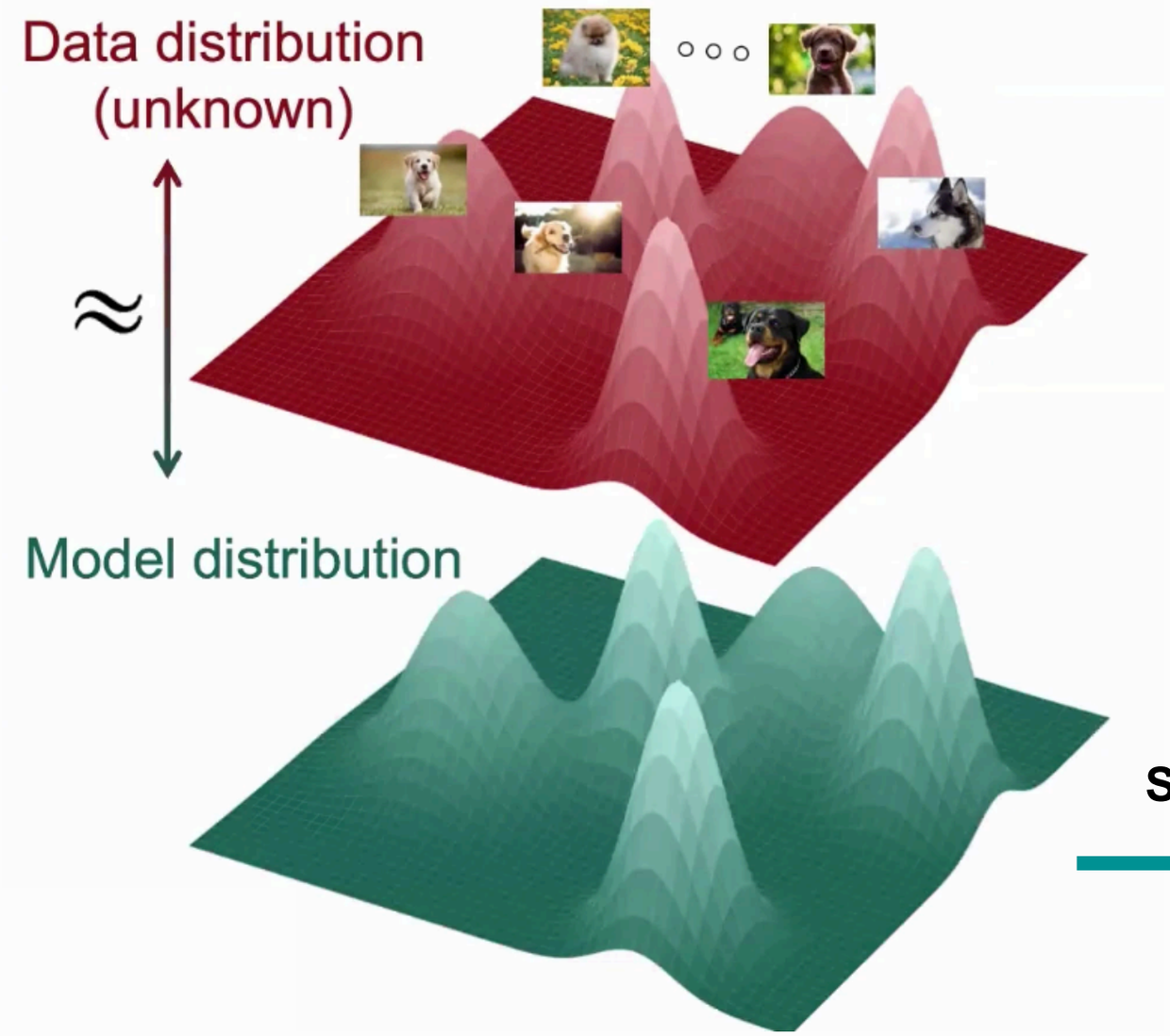




Data samples

The key challenge for building complex generative models

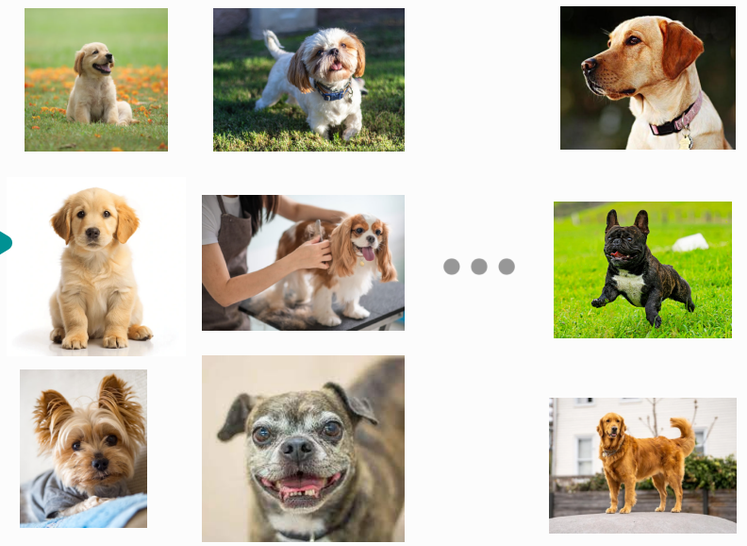


The key challenge for building complex generative models

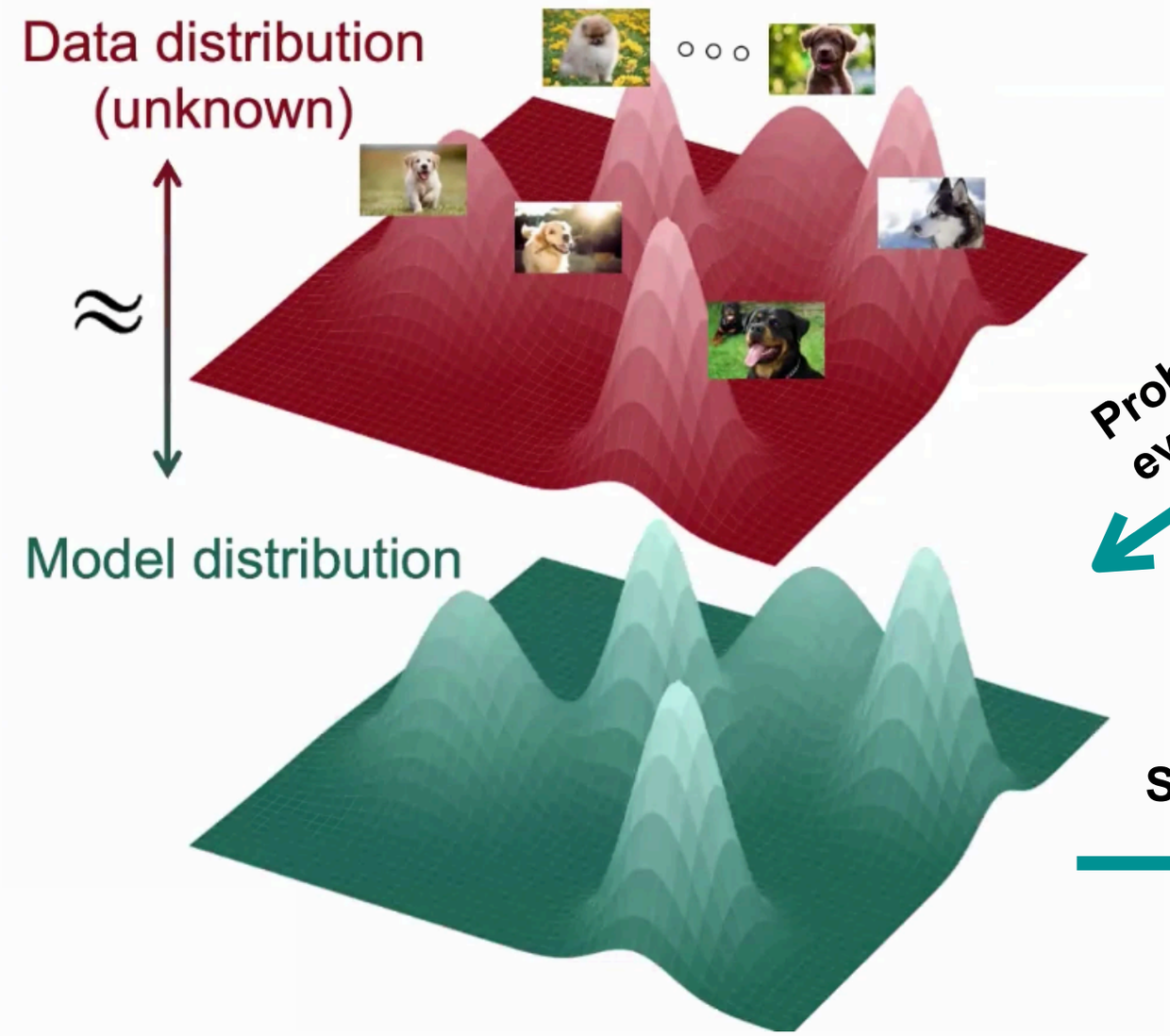


Sampling

Novel data points



The key challenge for building complex generative models



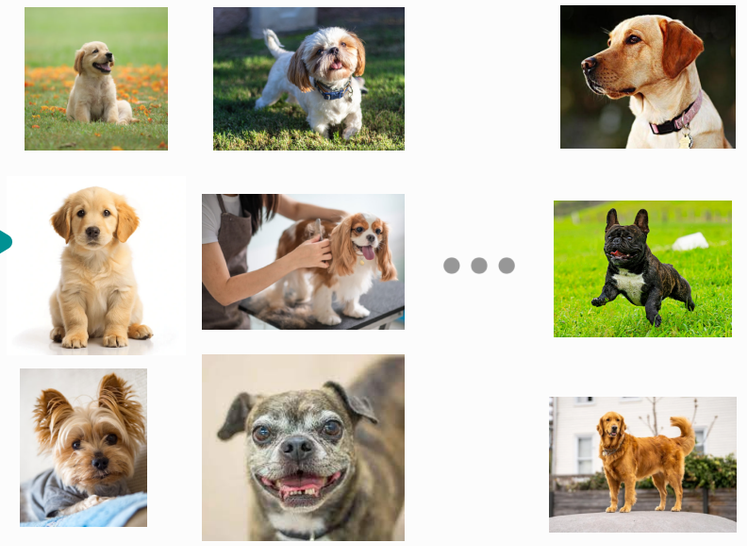
Probability evaluation

High probability

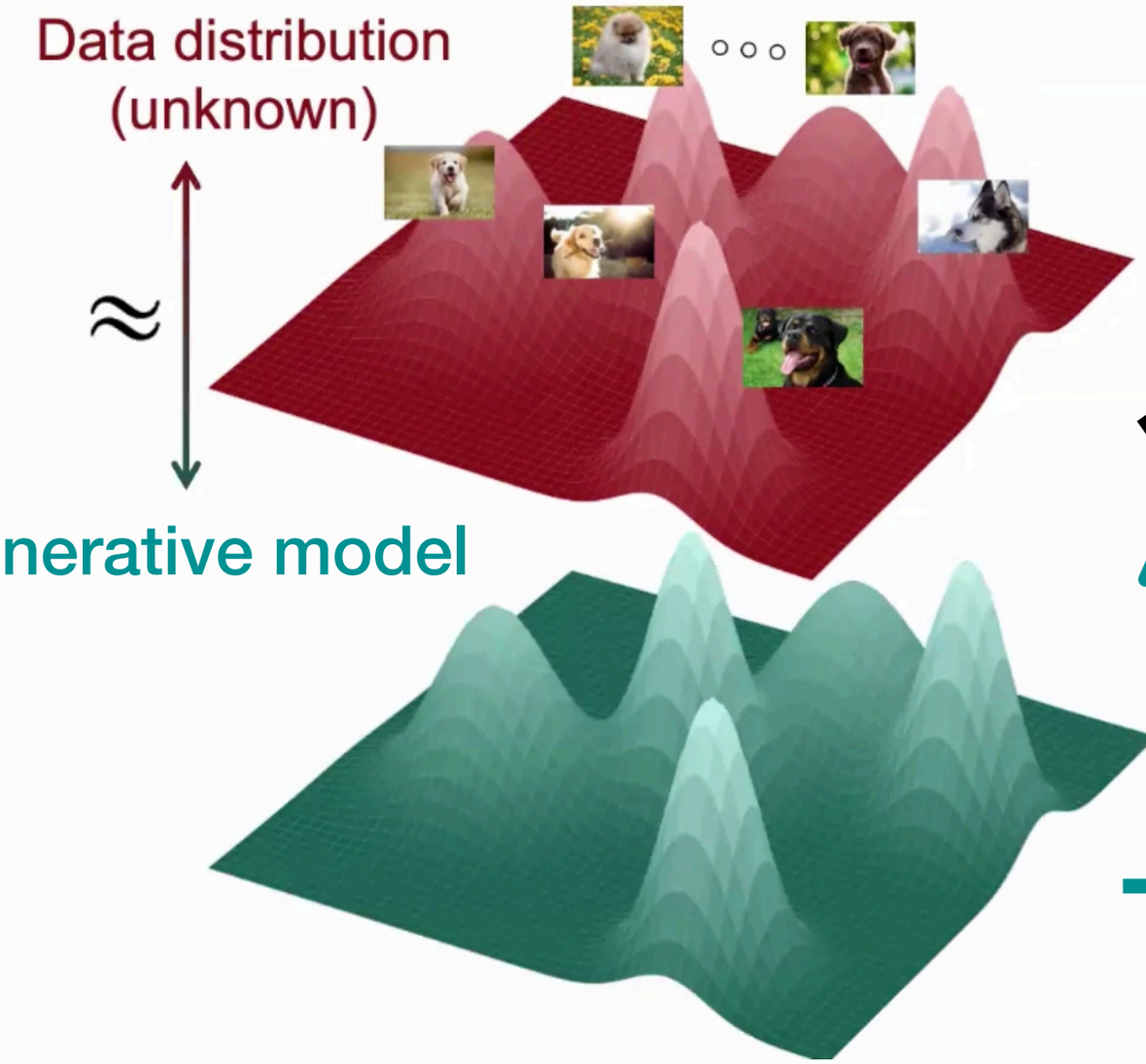
Low probability

Sampling

Novel data points



The key challenge for building complex generative models



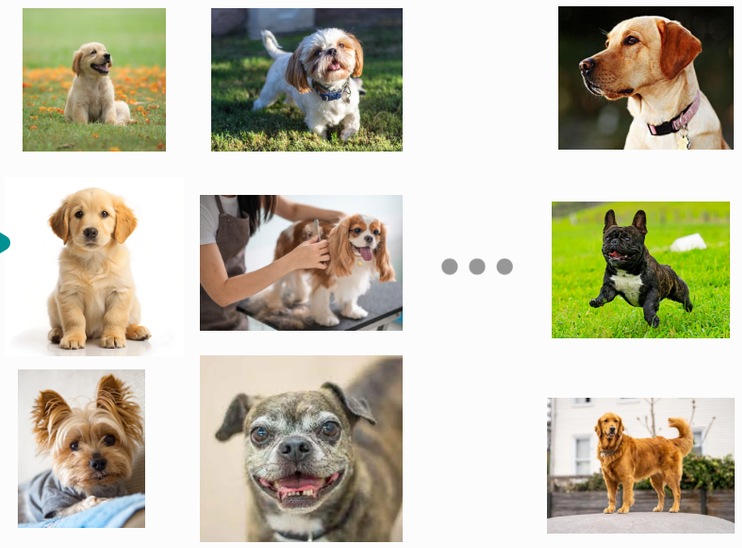
Probability evaluation

High probability

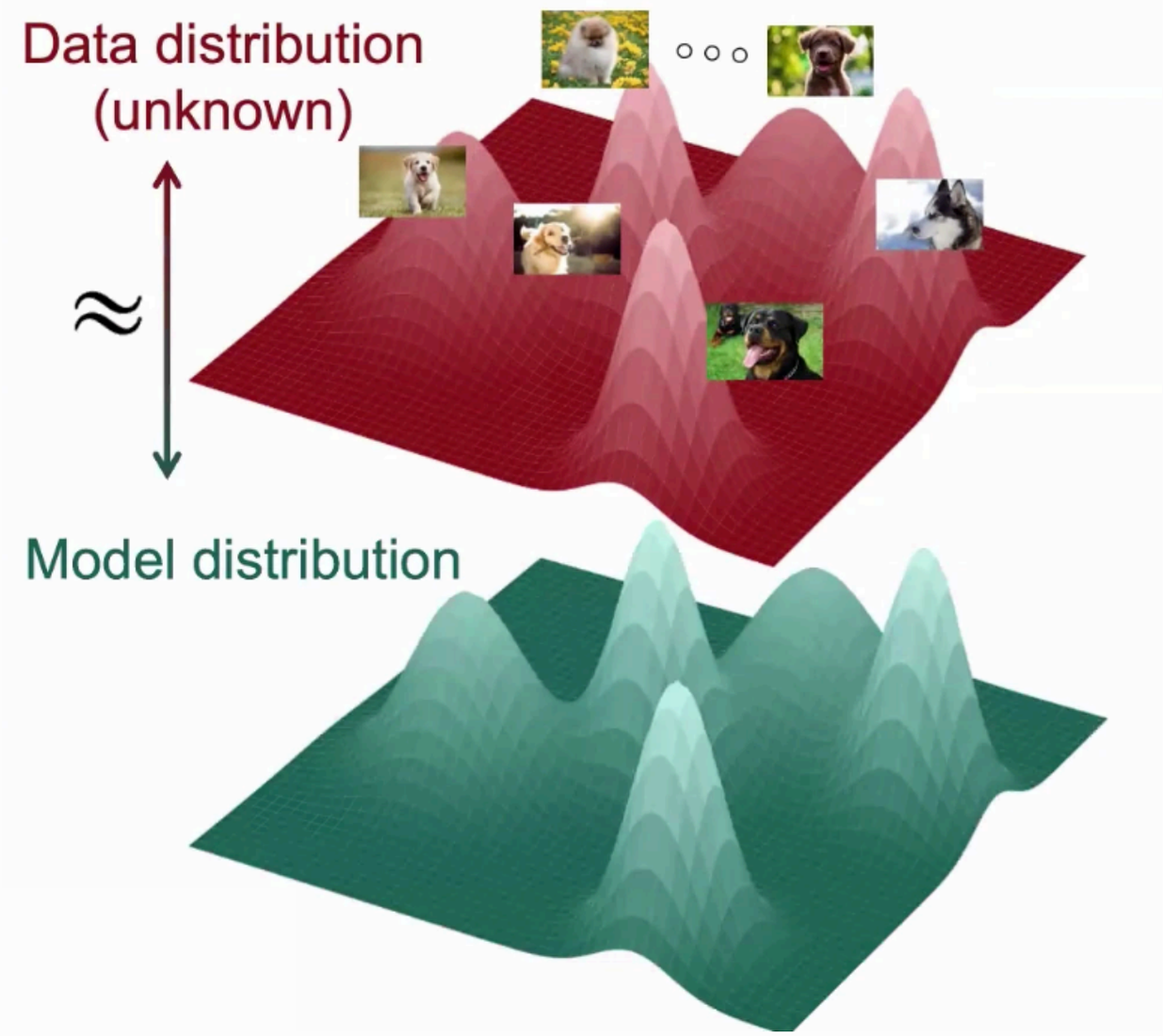
Low probability

Sampling

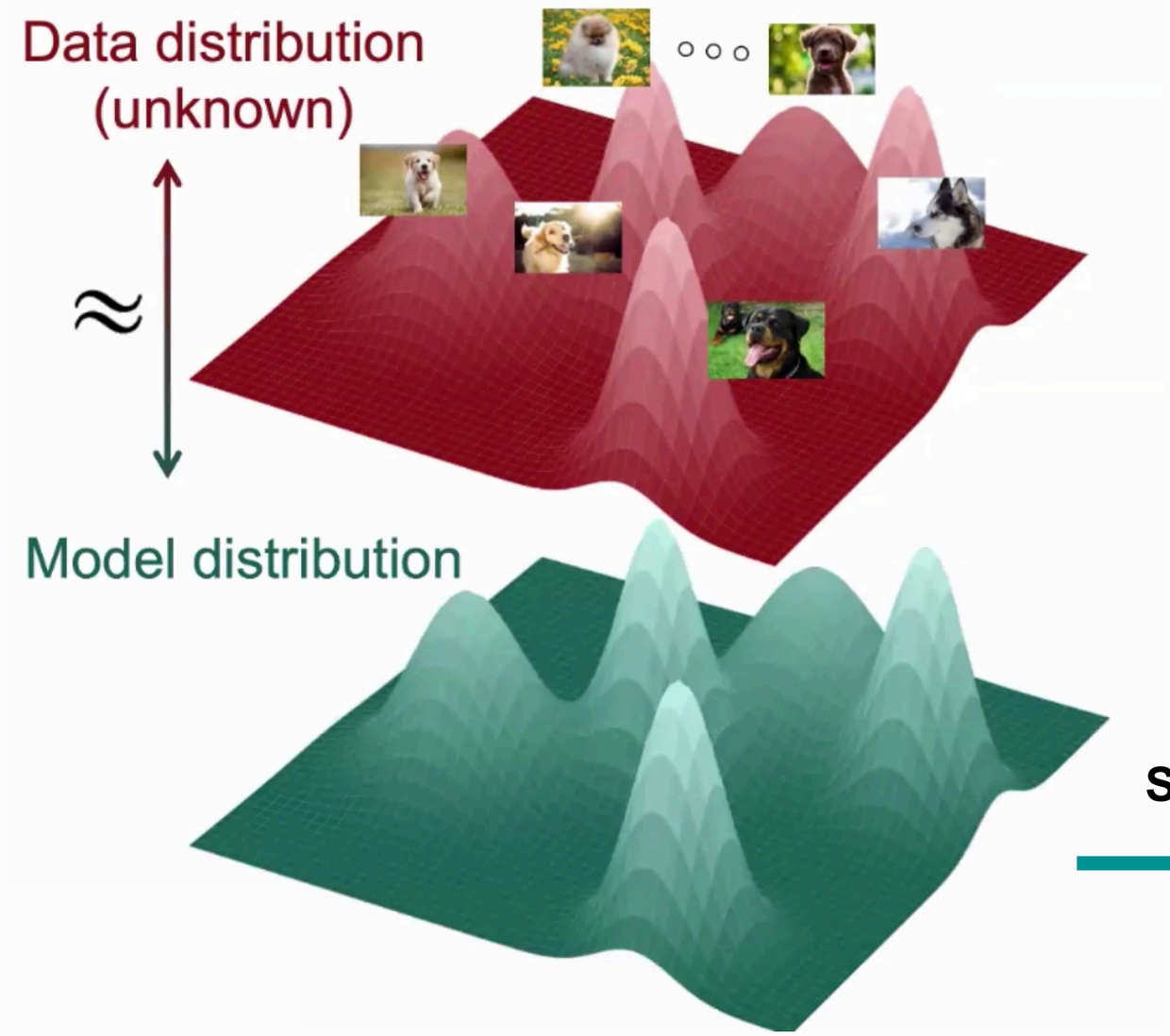
Novel data points



The key challenge for building complex generative models

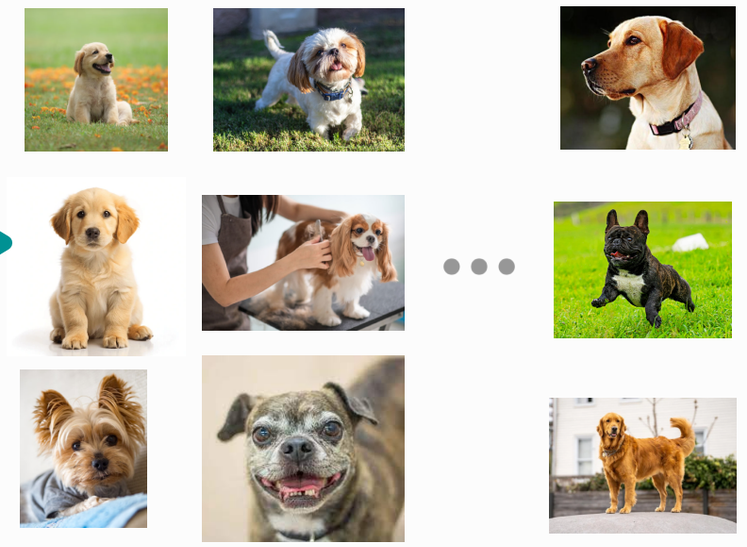


The key challenge for building complex generative models

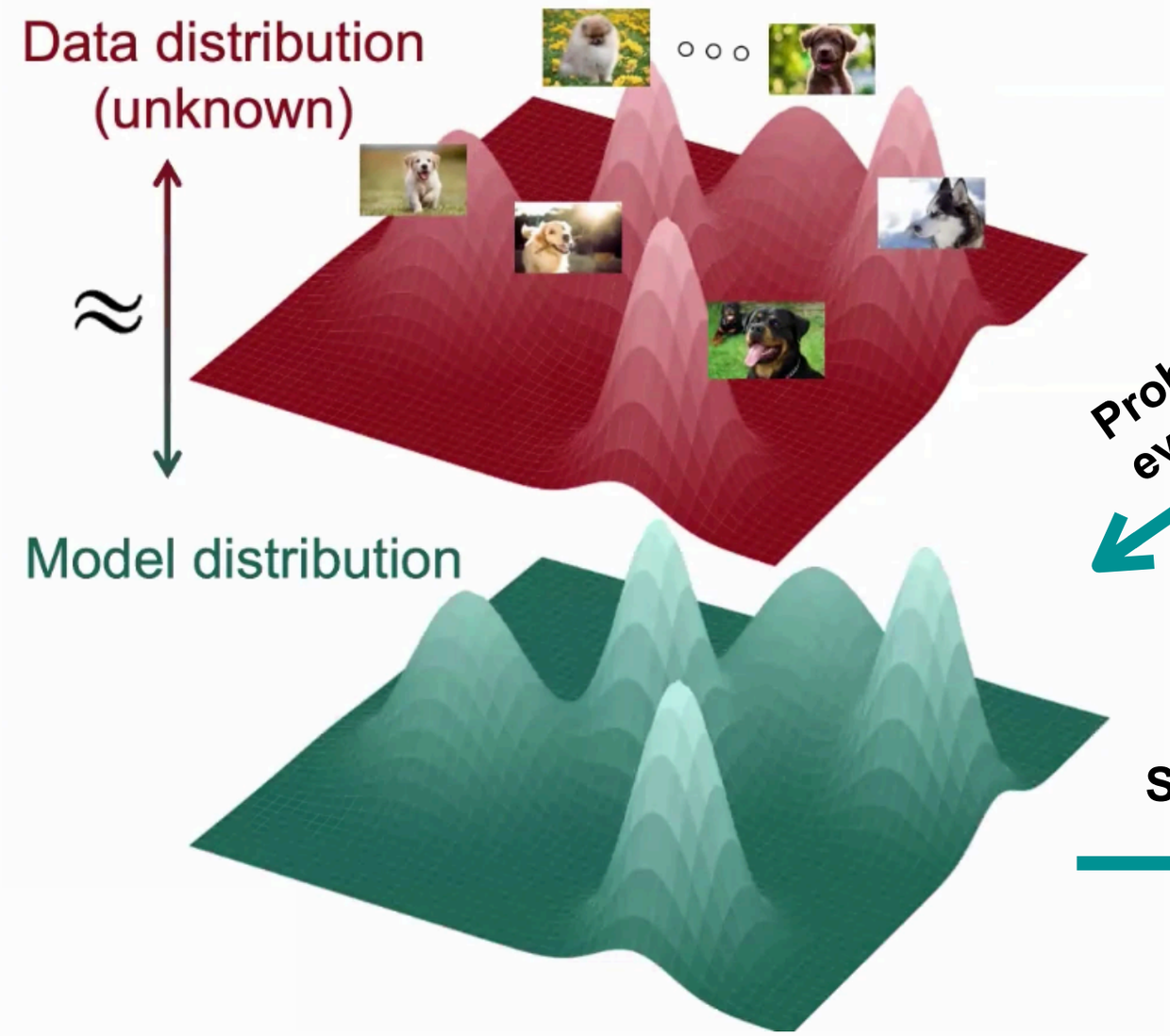


Sampling

Novel data points



The key challenge for building complex generative models



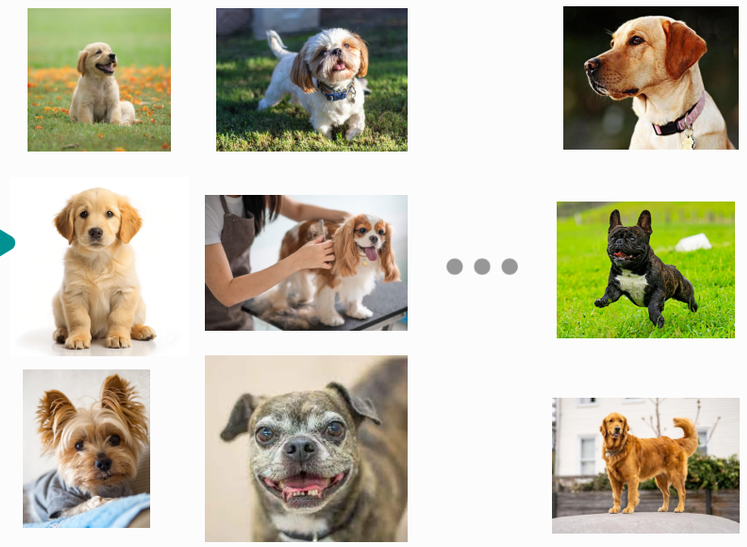
Probability evaluation

High probability

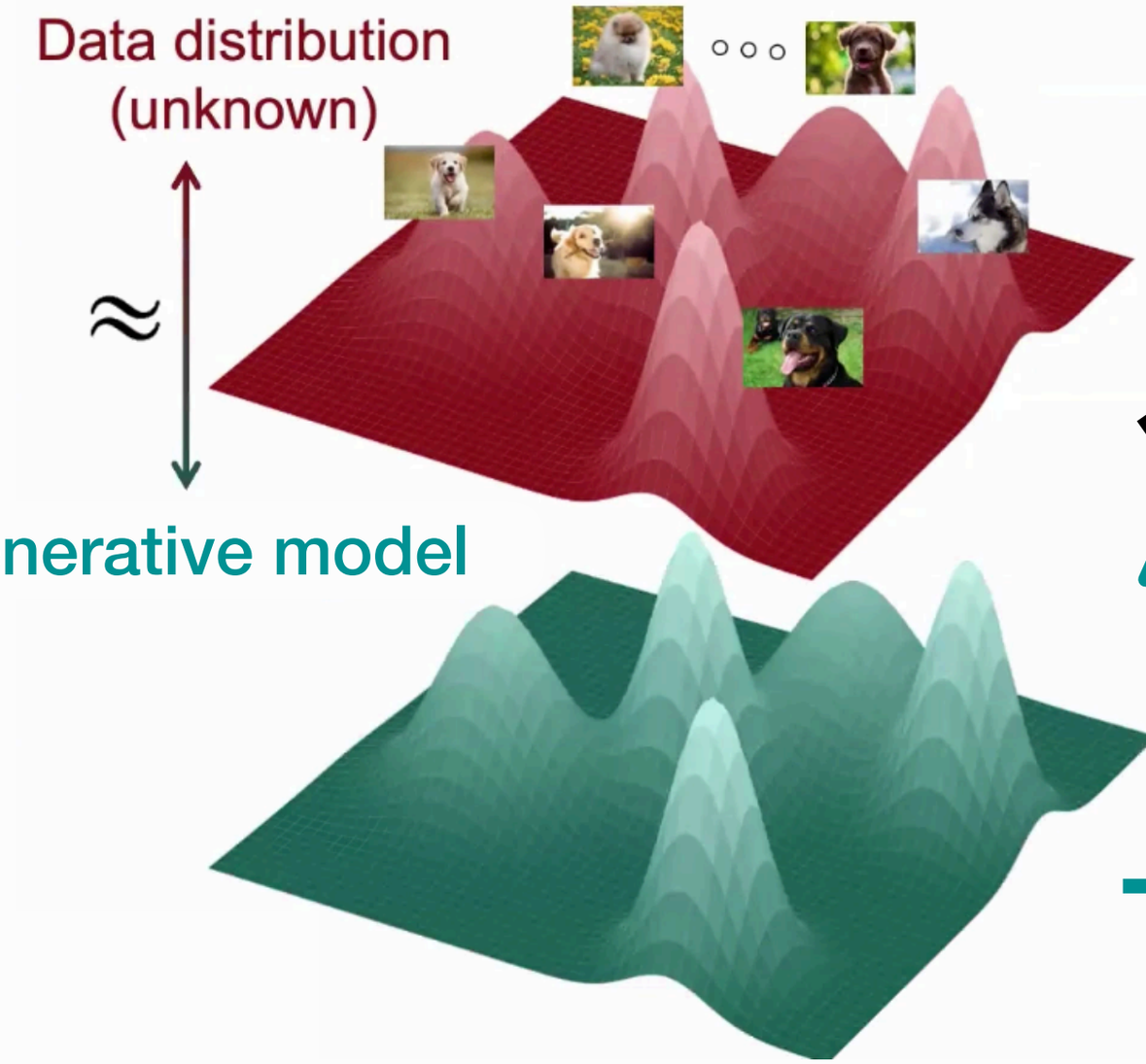
Low probability

Sampling

Novel data points



The key challenge for building complex generative models



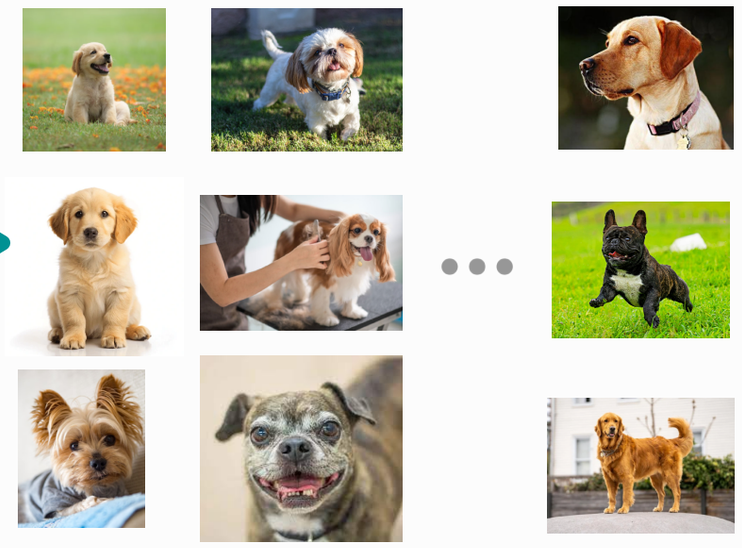
Probability evaluation

High probability

Low probability

Sampling

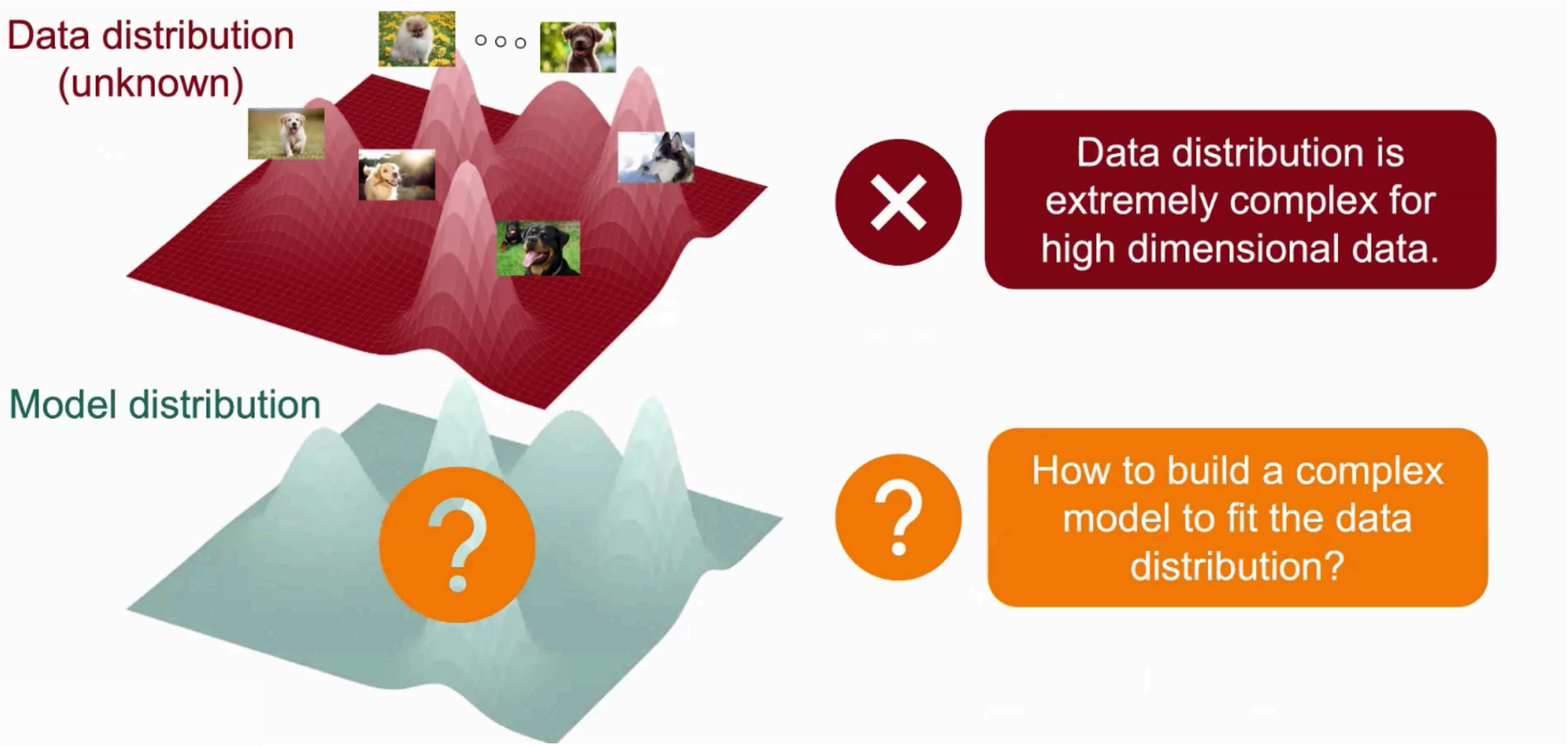
Novel data points



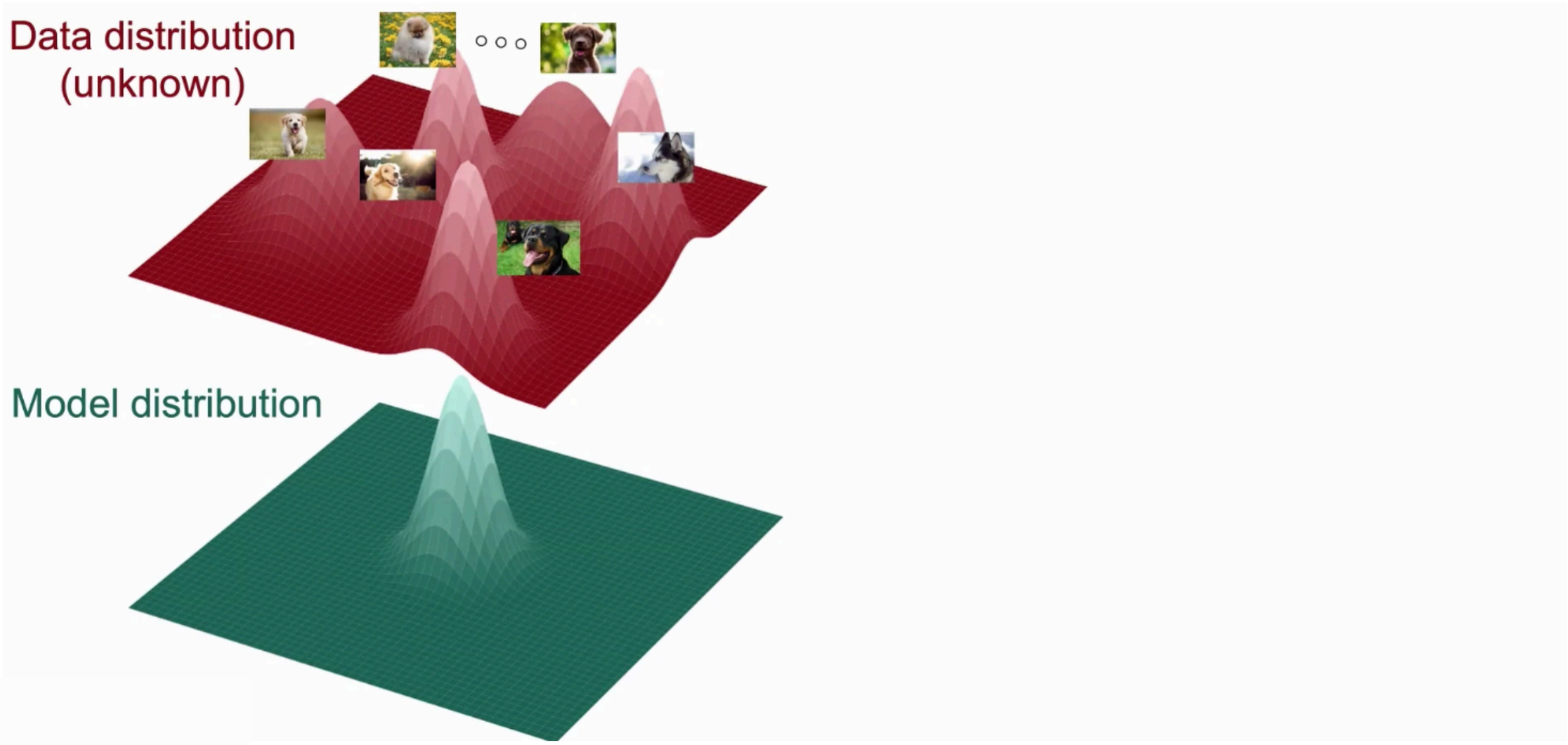
The key challenge for building complex generative models

Data distribution is extremely complex for high dimensional data.

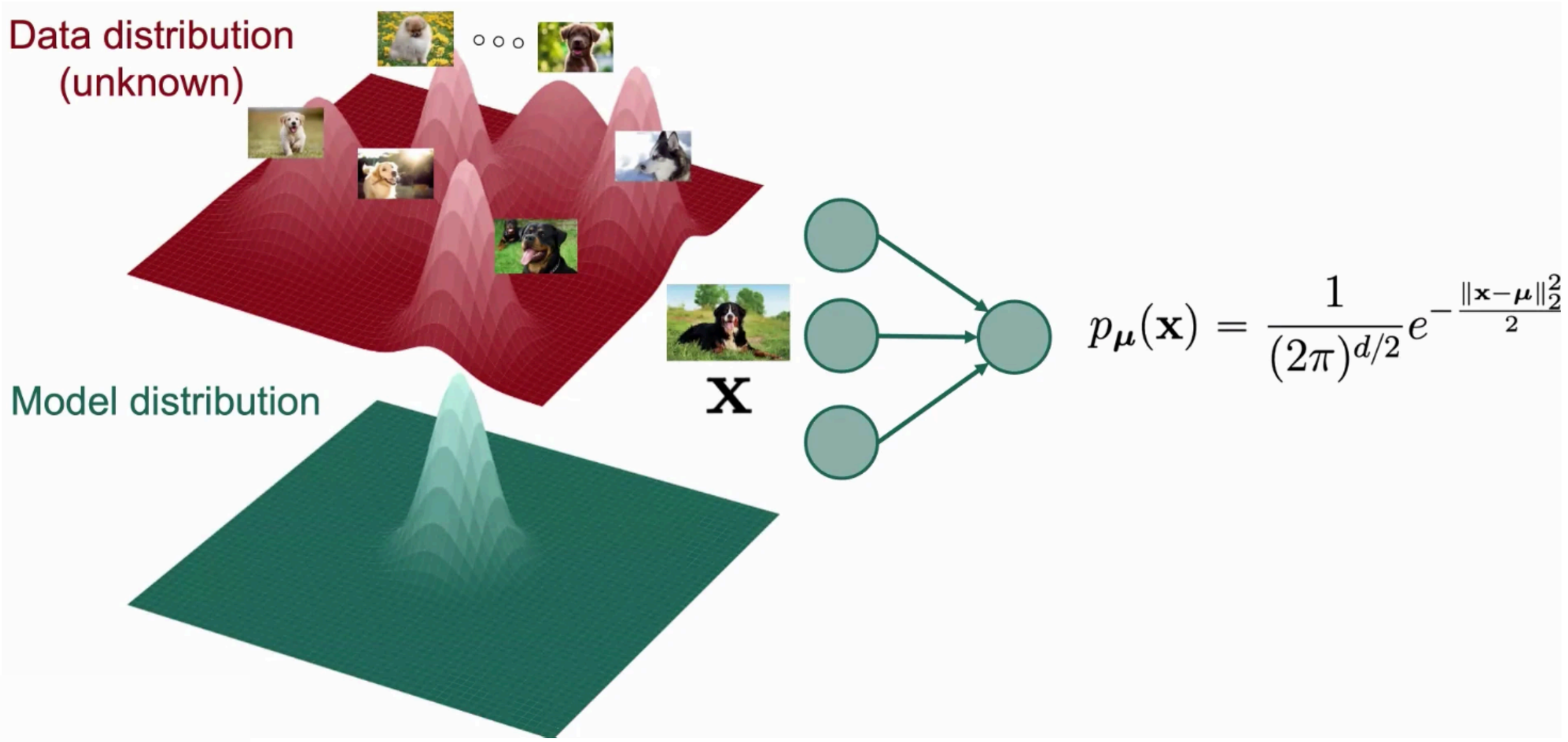
The key challenge for building complex generative models



The key challenge for building complex generative models

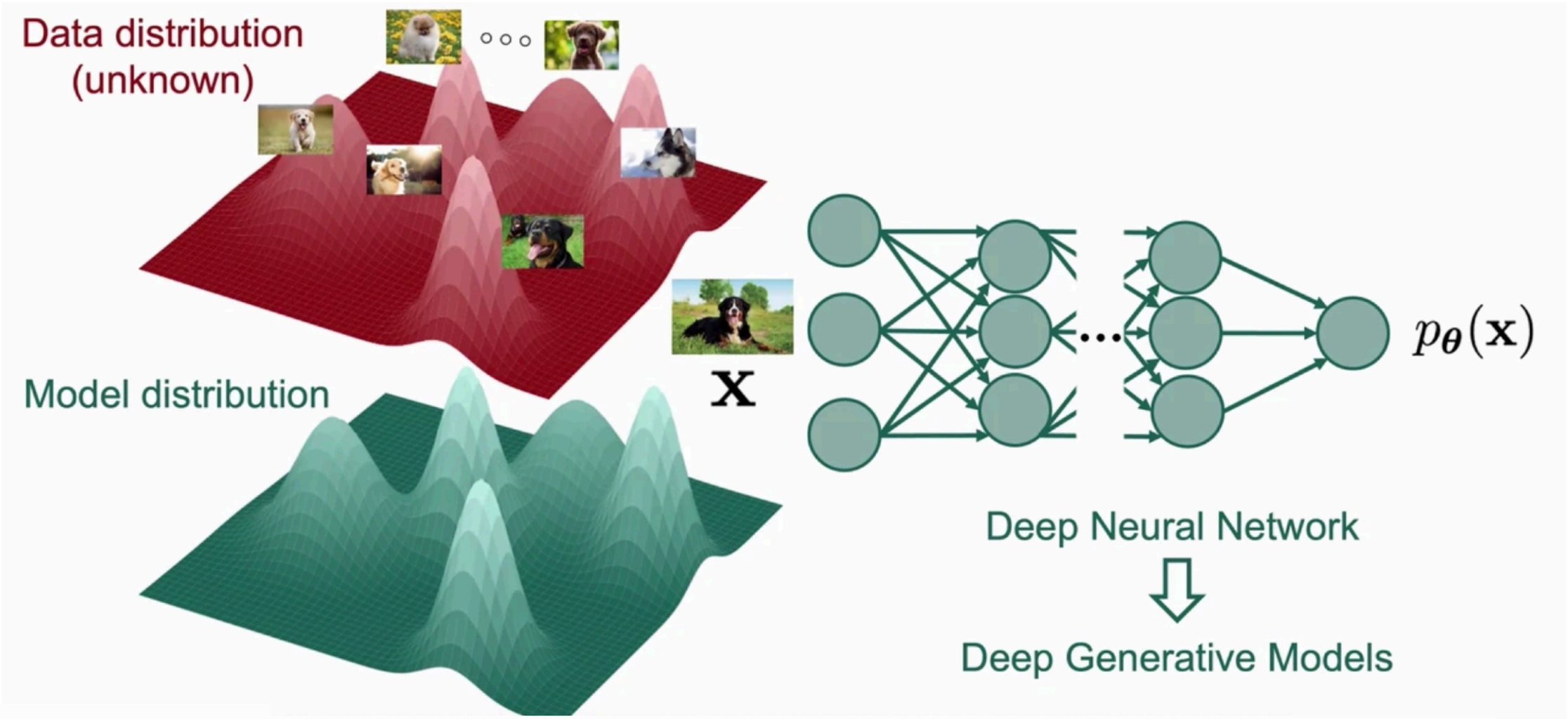


The key challenge for building complex generative models

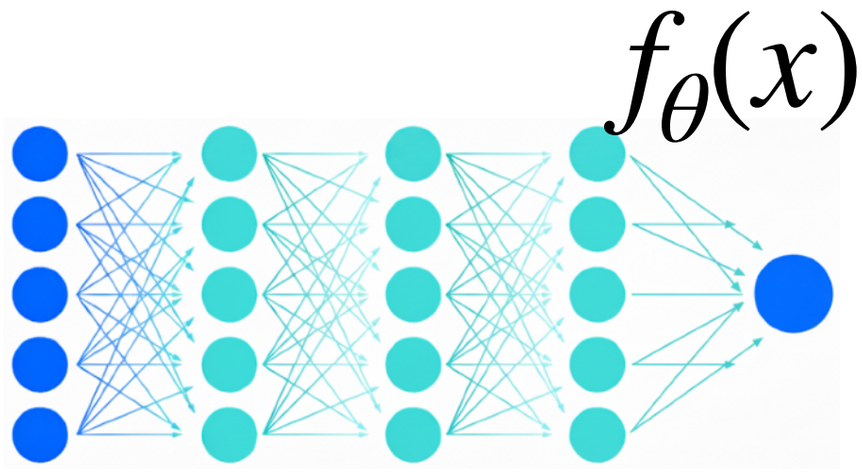


$$p_{\boldsymbol{\mu}}(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}} e^{-\frac{\|\mathbf{x}-\boldsymbol{\mu}\|_2^2}{2}}$$

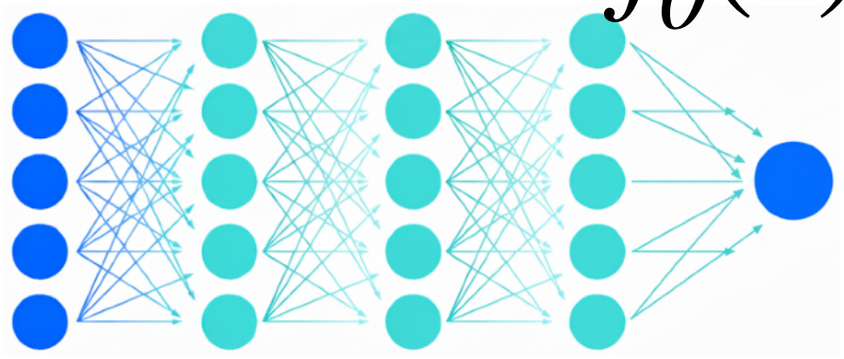
The key challenge for building complex generative models



The key challenge for building complex generative models

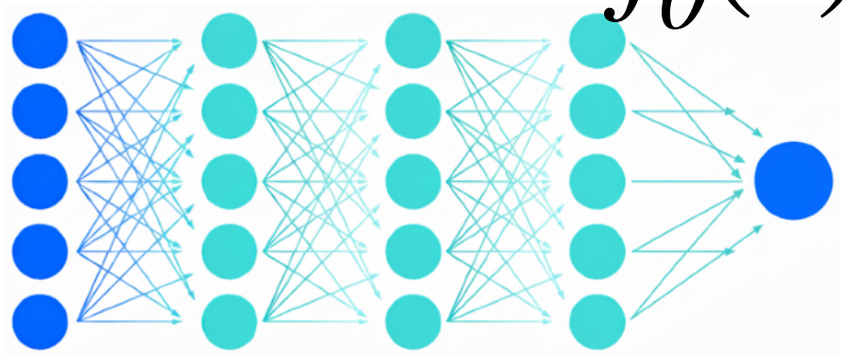


The key challenge for building complex generative models



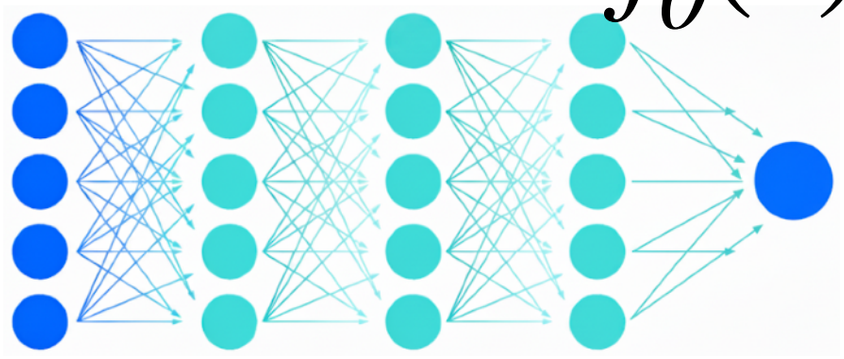
$$f_\theta(x) \longrightarrow e^{f_\theta(x)}$$

The key challenge for building complex generative models

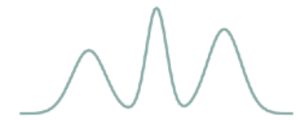


$$f_\theta(x) \rightarrow e^{f_\theta(x)} \rightarrow \frac{e^{f_\theta(x)}}{Z}$$

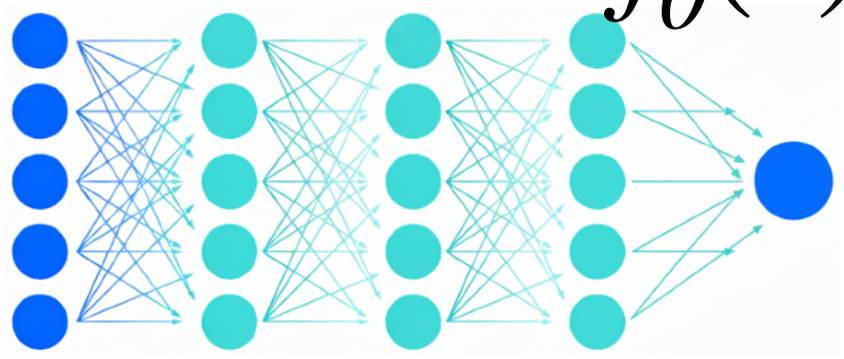
The key challenge for building complex generative models



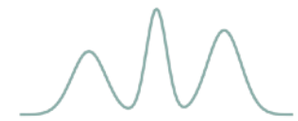
$$f_\theta(x) \rightarrow e^{f_\theta(x)} \rightarrow \frac{e^{f_\theta(x)}}{Z} = p_\theta(x)$$



The key challenge for building complex generative models

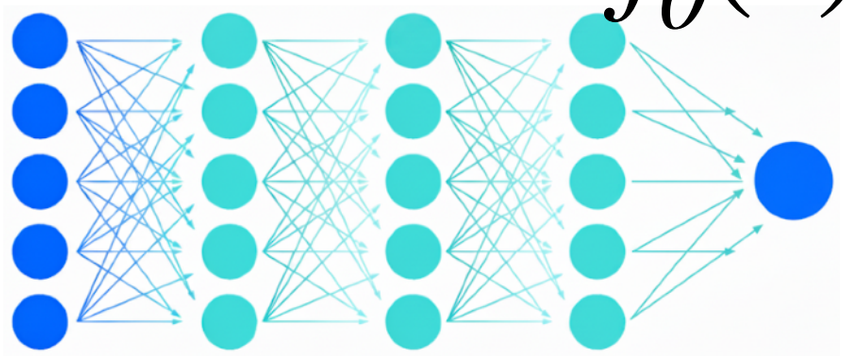


$$f_\theta(x) \rightarrow e^{f_\theta(x)} \rightarrow \frac{e^{f_\theta(x)}}{Z} = p_\theta(x)$$

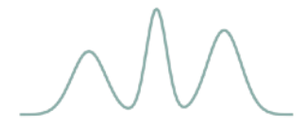


Normalizing constant

The key challenge for building complex generative models



$$f_\theta(x) \rightarrow e^{f_\theta(x)} \rightarrow \frac{e^{f_\theta(x)}}{Z} = p_\theta(x)$$

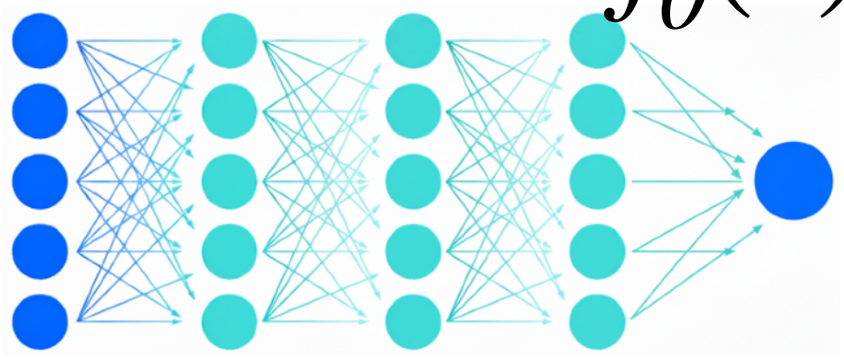


Normalizing constant

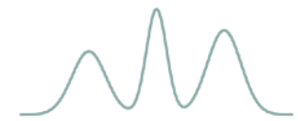
Normalization constant for Gaussian

$$Z_\mu = \frac{1}{(2\pi)^{d/2}}$$

The key challenge for building complex generative models

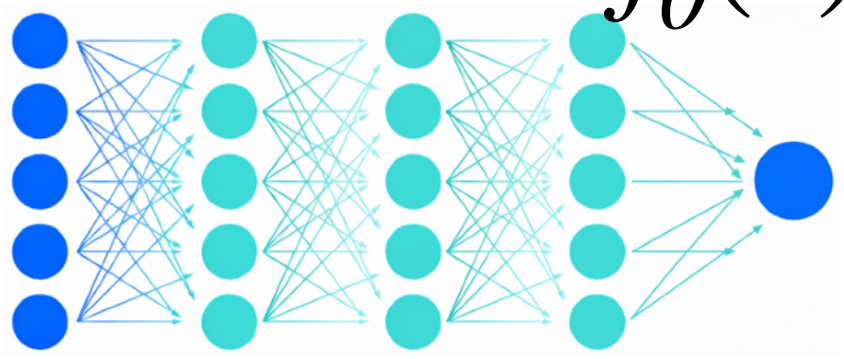


$$f_\theta(x) \rightarrow e^{f_\theta(x)} \rightarrow \frac{e^{f_\theta(x)}}{Z} = p_\theta(x)$$

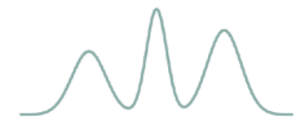


Normalizing constant

The key challenge for building complex generative models



$$f_\theta(x) \rightarrow e^{f_\theta(x)} \rightarrow \frac{e^{f_\theta(x)}}{Z} = p_\theta(x)$$



Normalizing constant

$$Z_\theta = \int e^{f_\theta(\mathbf{x})}\, d\mathbf{x}$$

Approximating Normalizing Constant

Approximating Normalizing Constant

Approximating Normalizing Constant

Inaccurate probability evaluation

Approximating Normalizing Constant

Inaccurate probability evaluation

Approximating Normalizing Constant

- ✗ Inaccurate probability evaluation
- ✗ Restricted model family

Approximating Normalizing Constant

- ✗ Inaccurate probability evaluation
- ✗ Restricted model family

Approximating Normalizing Constant

- Inaccurate probability evaluation
- Restricted model family
- Cannot evaluate probabilities

Desiderata of better generative modelling

- ✗ Inaccurate probability evaluation
- ✗ Restricted model family
- ✗ Cannot evaluate probabilities

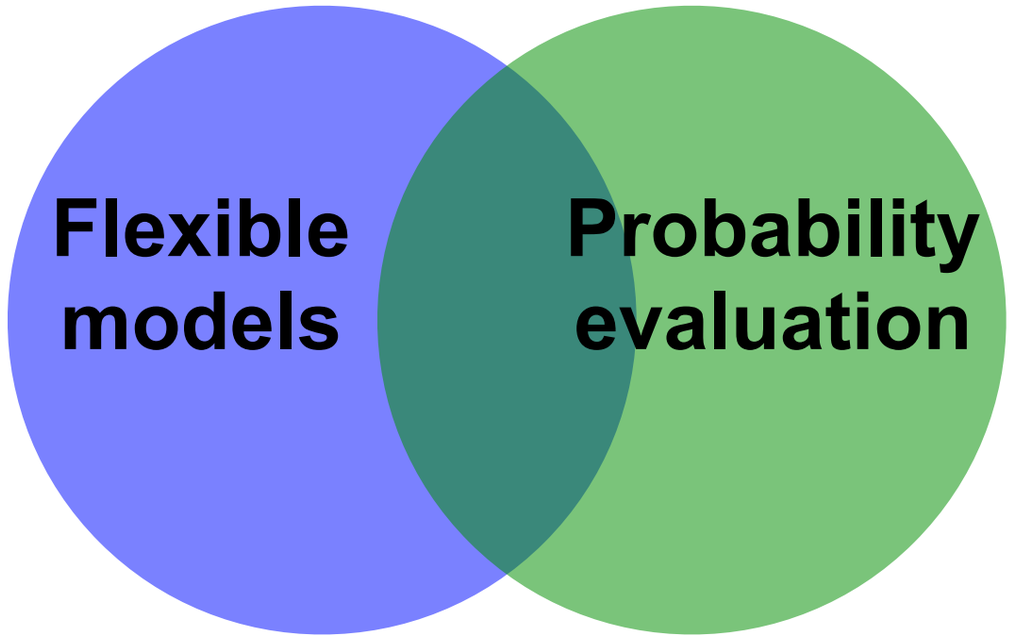
Desiderata of better generative modelling

- ✗ Inaccurate probability evaluation
- ✗ Restricted model family
- ✗ Cannot evaluate probabilities

Flexible models

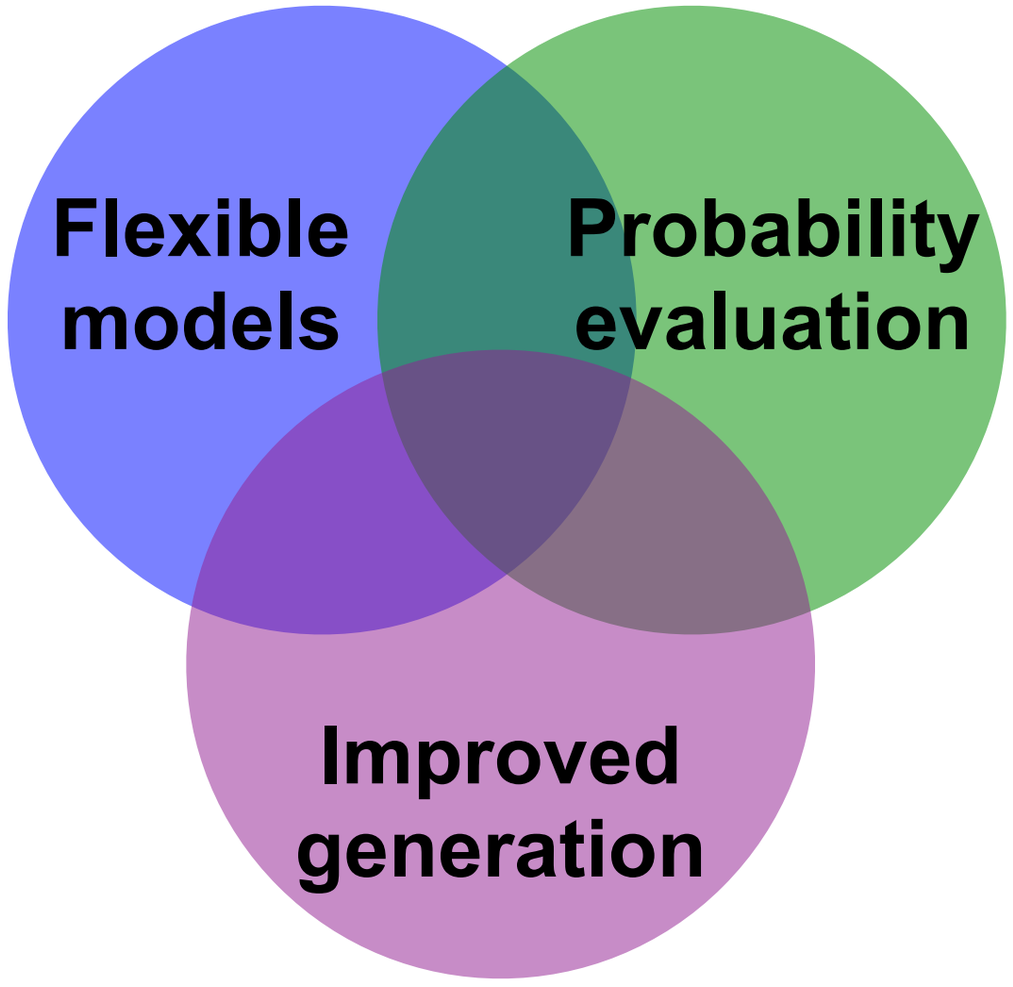
Desiderata of better generative modelling

- ✗ Inaccurate probability evaluation
- ✗ Restricted model family
- ✗ Cannot evaluate probabilities



Desiderata of better generative modelling

- Inaccurate probability evaluation
- Restricted model family
- Cannot evaluate probabilities



A solution: working with score functions

$$p(x)$$

density function

$$\nabla \log p(x)$$

(Stein) Score function

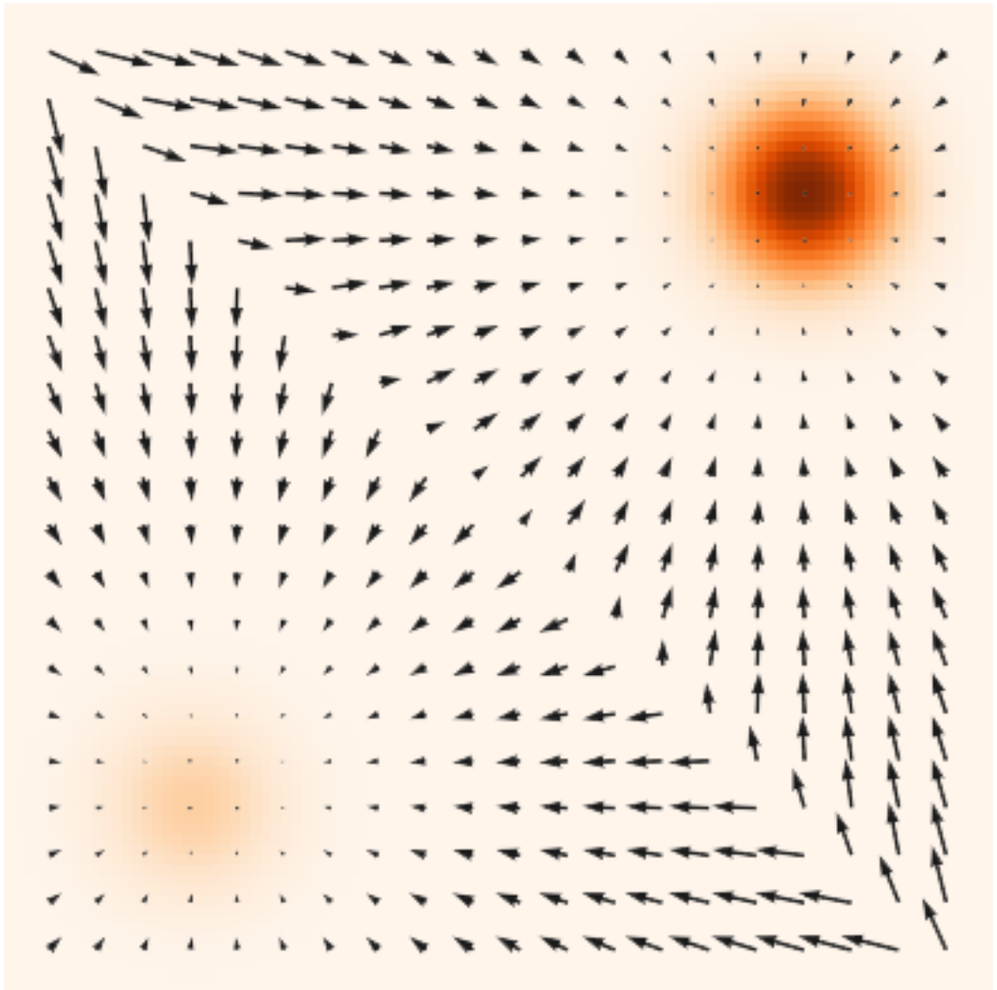
A solution: working with score functions

$$p(x)$$

density function

$$\nabla \log p(x)$$

(Stein) Score function



(pdf and score)

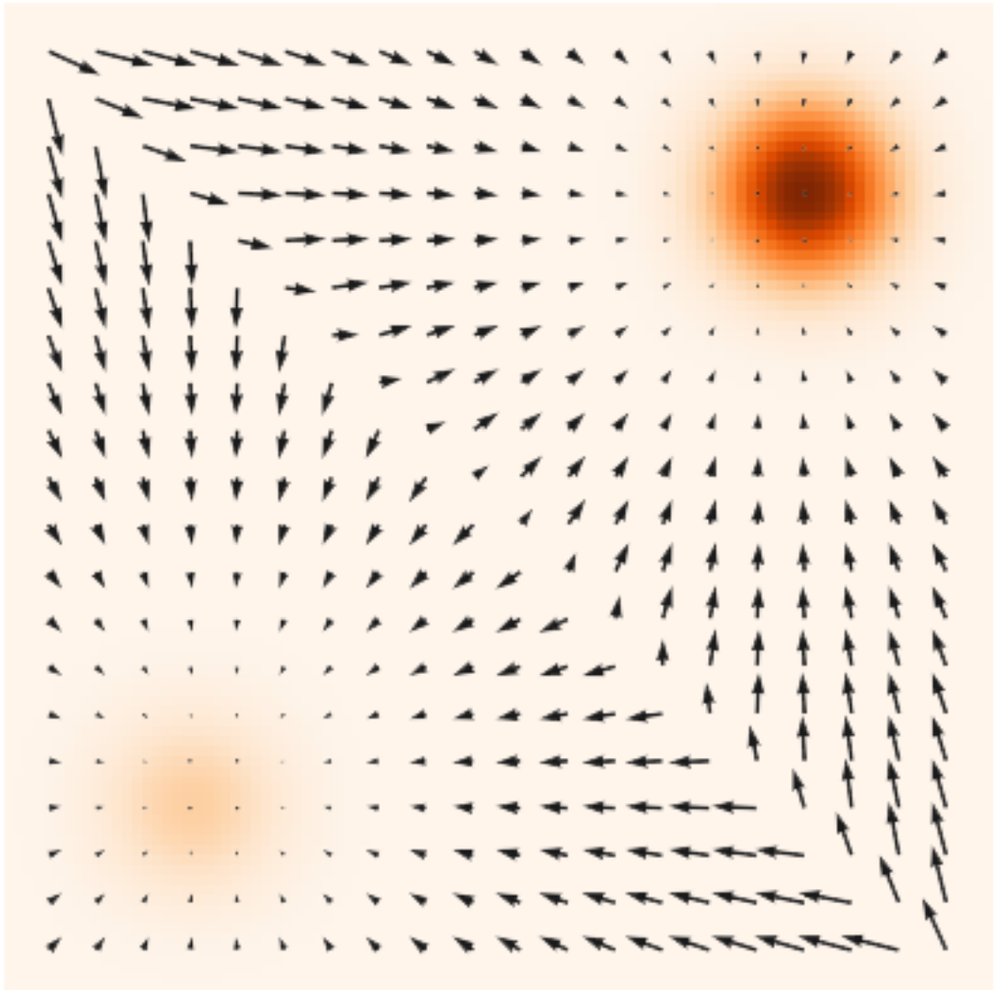
A solution: working with score functions

$$p(x)$$

density function

$$\nabla \log p(x)$$

(Stein) Score function



(pdf and score)

A solution: working with score functions

Flexible models

- Bypass the normalizing constant
- Principled statistical methods

[**Song** et al. UAI 2019 oral]

Improved generation

- Higher sample quality than GANs
- Controllable generation

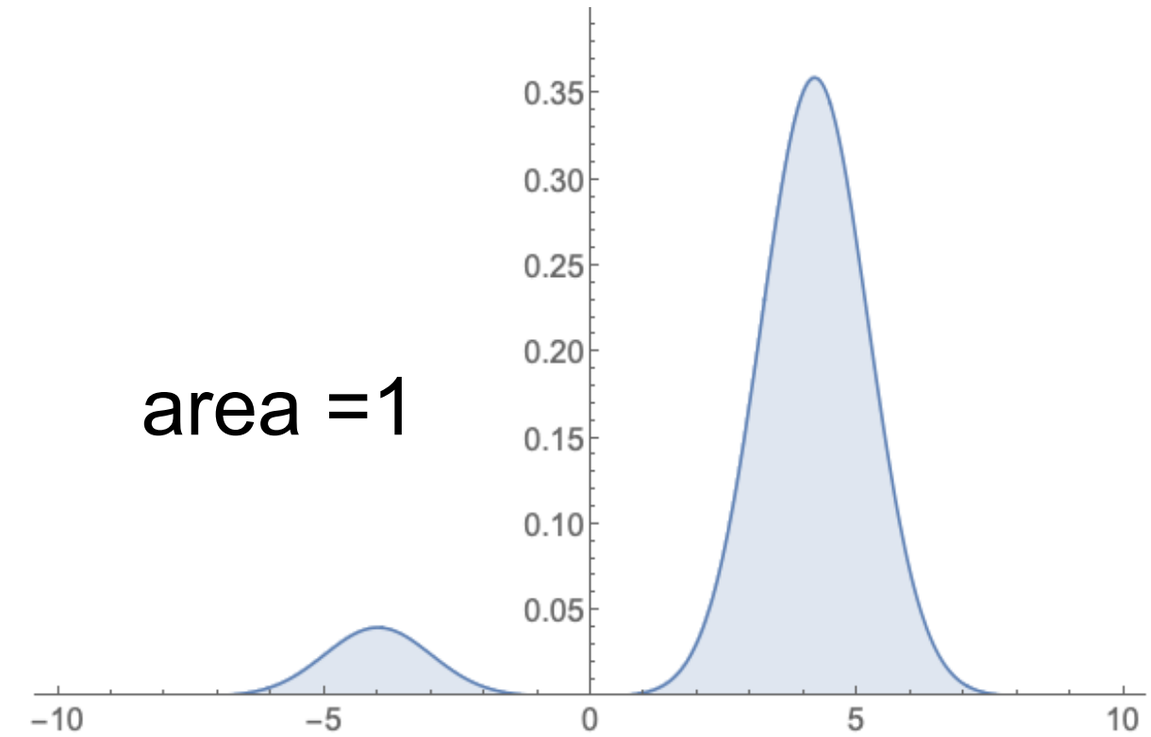
[**Song** & Ermon. NeurIPS 2019 oral]
[**Song** & Ermon. NeurIPS 2020]
[**Song** et al. ICLR 2021 oral]
(Outstanding Paper Award)
[**Song** et al. ICLR 2022]

Probability evaluation

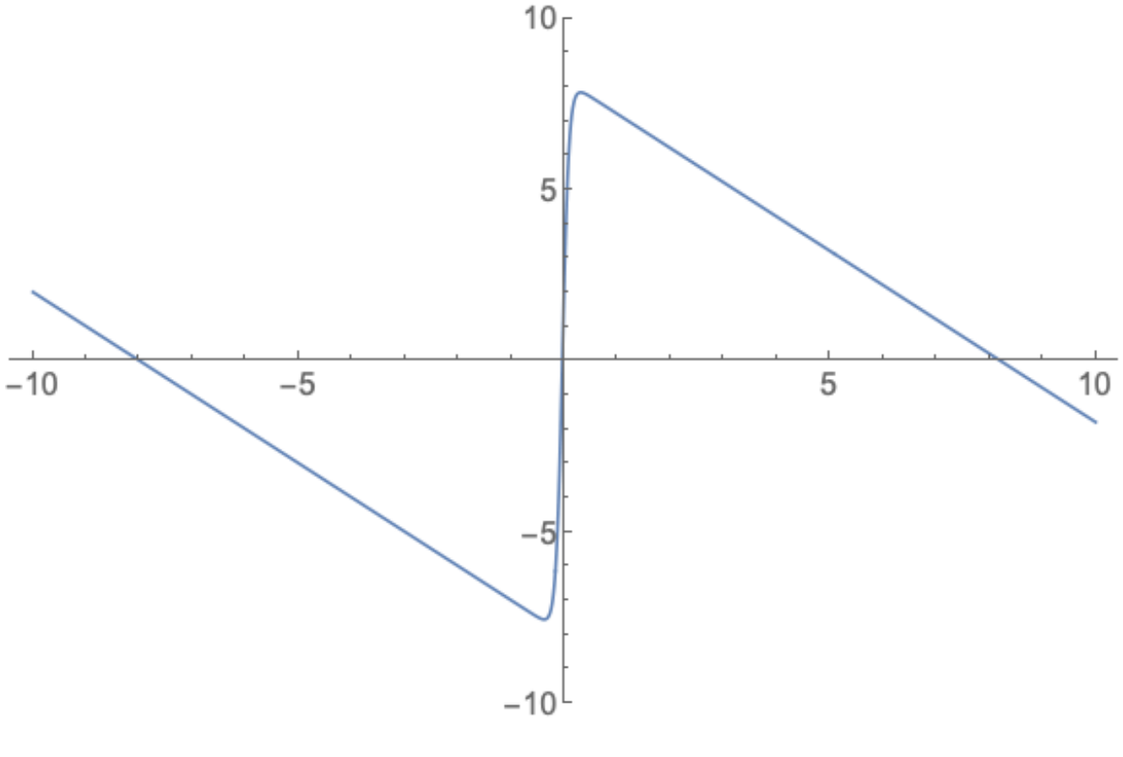
- Accurate probability evaluation
- Better estimation of data probabilities

[**Song** et al. ICLR 2021 oral]
[**Song** et al. NeurIPS 2021 spotlight]

area =1



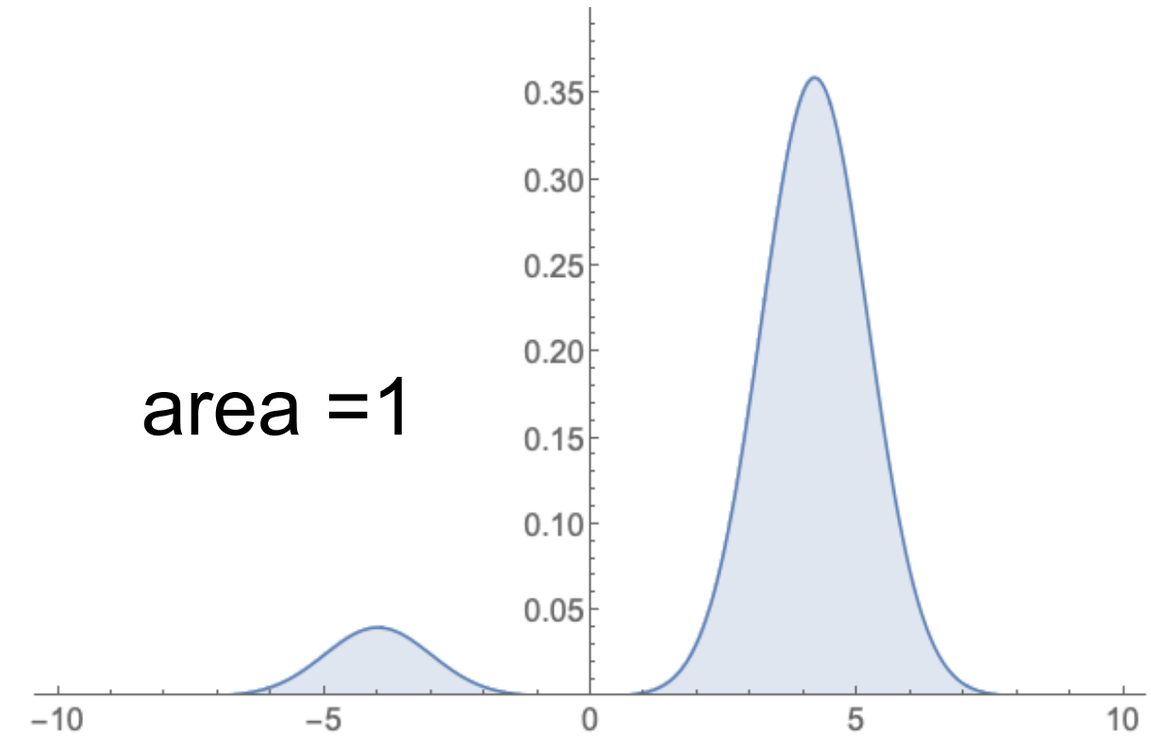
Probability density function



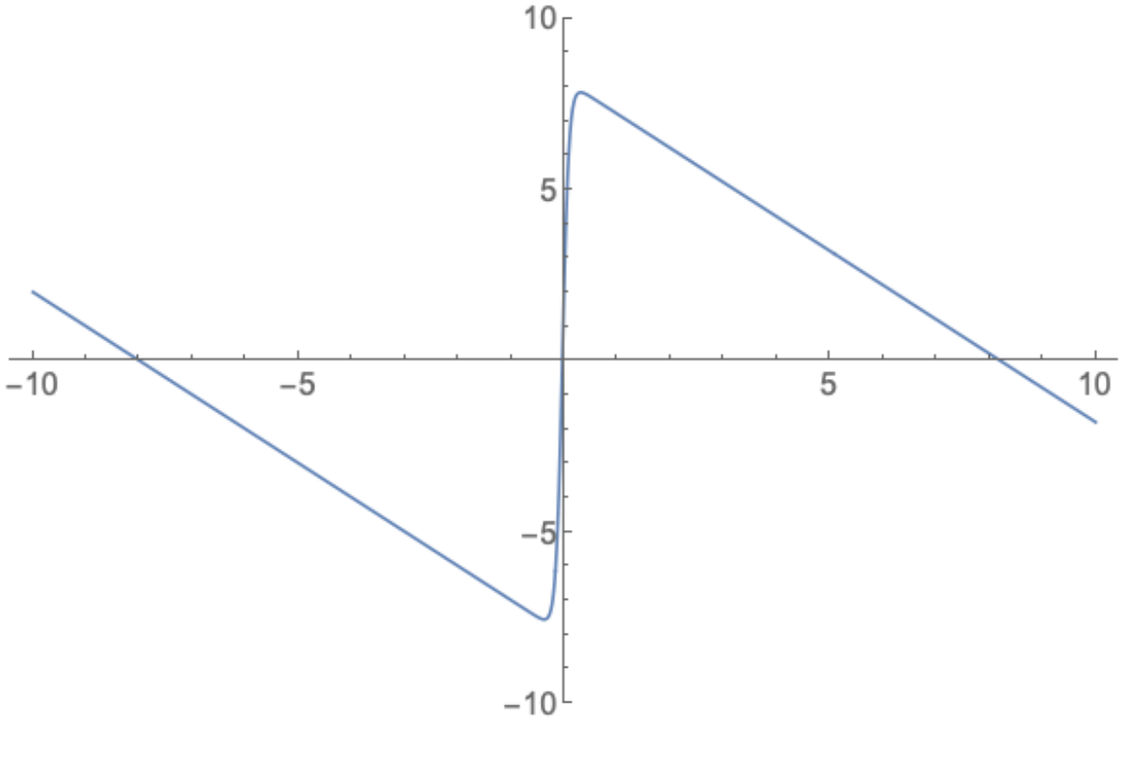
Score function

$$\nabla_{\mathbf{x}} \log p_\theta(\mathbf{x}) = \nabla_{\mathbf{x}} f_\theta(\mathbf{x}) - \nabla_{\mathbf{x}} \log Z_\theta$$

area =1



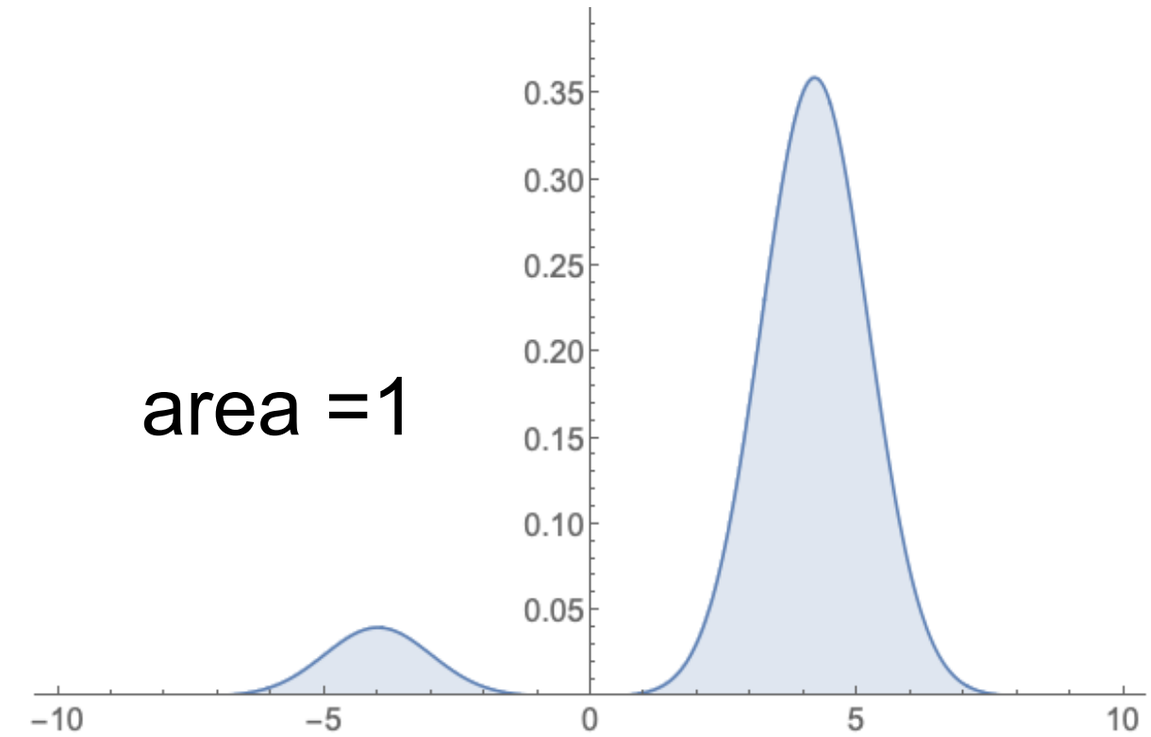
Probability density function



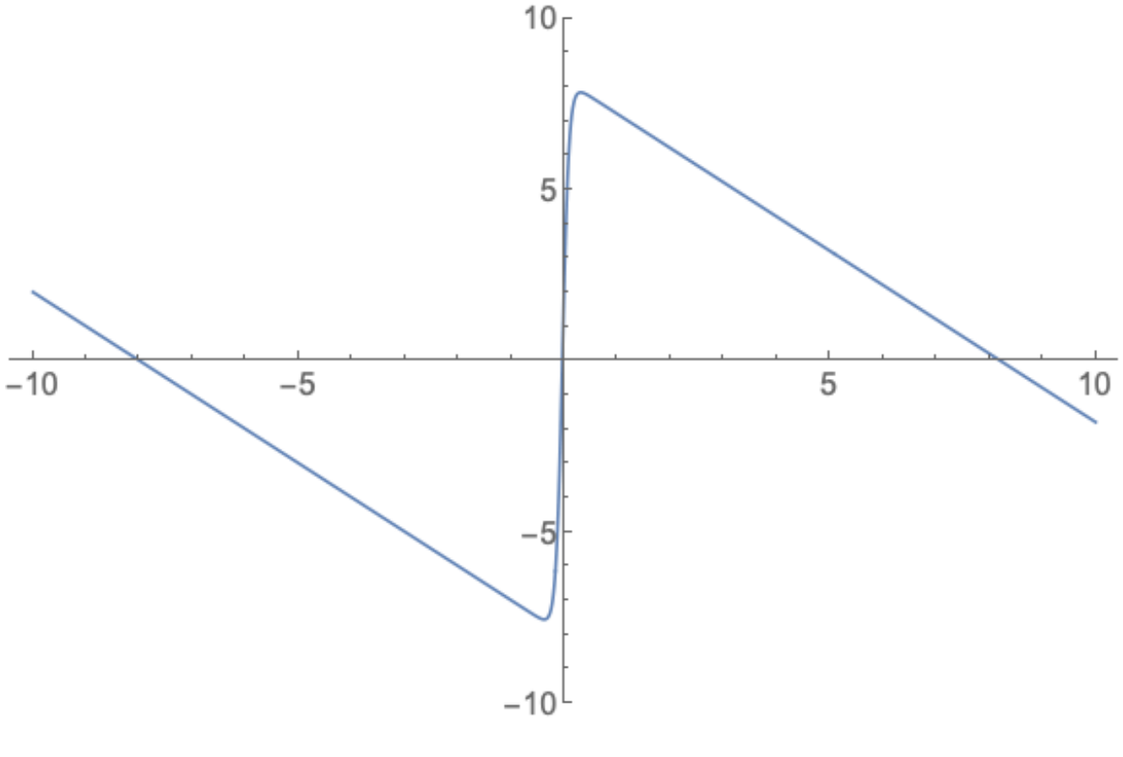
Score function

$$\nabla_{\mathbf{x}} \log p_\theta(\mathbf{x}) = \nabla_{\mathbf{x}} f_\theta(\mathbf{x}) - \nabla_{\mathbf{x}} \log Z_\theta$$

area =1



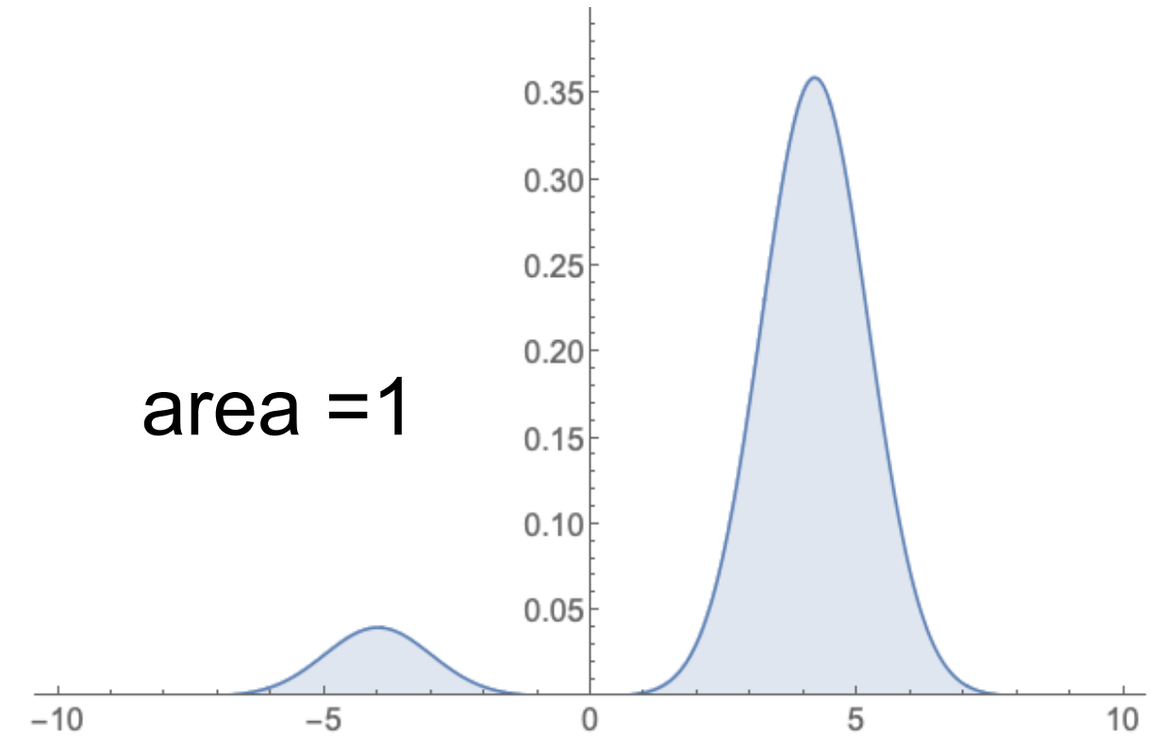
Probability density function



Score function

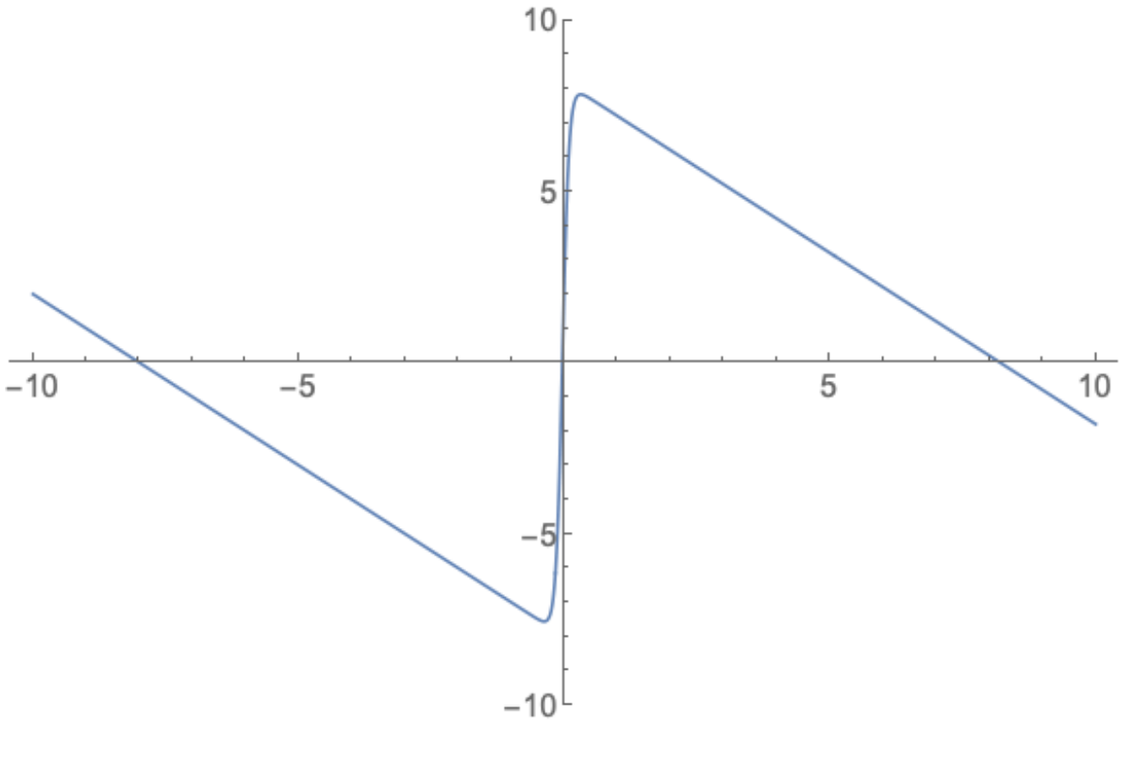
$$\nabla_{\mathbf{x}} \log p_\theta(\mathbf{x}) = \nabla_{\mathbf{x}} f_\theta(\mathbf{x}) - \nabla_{\mathbf{x}} \log Z_\theta$$

area =1



Probability density function

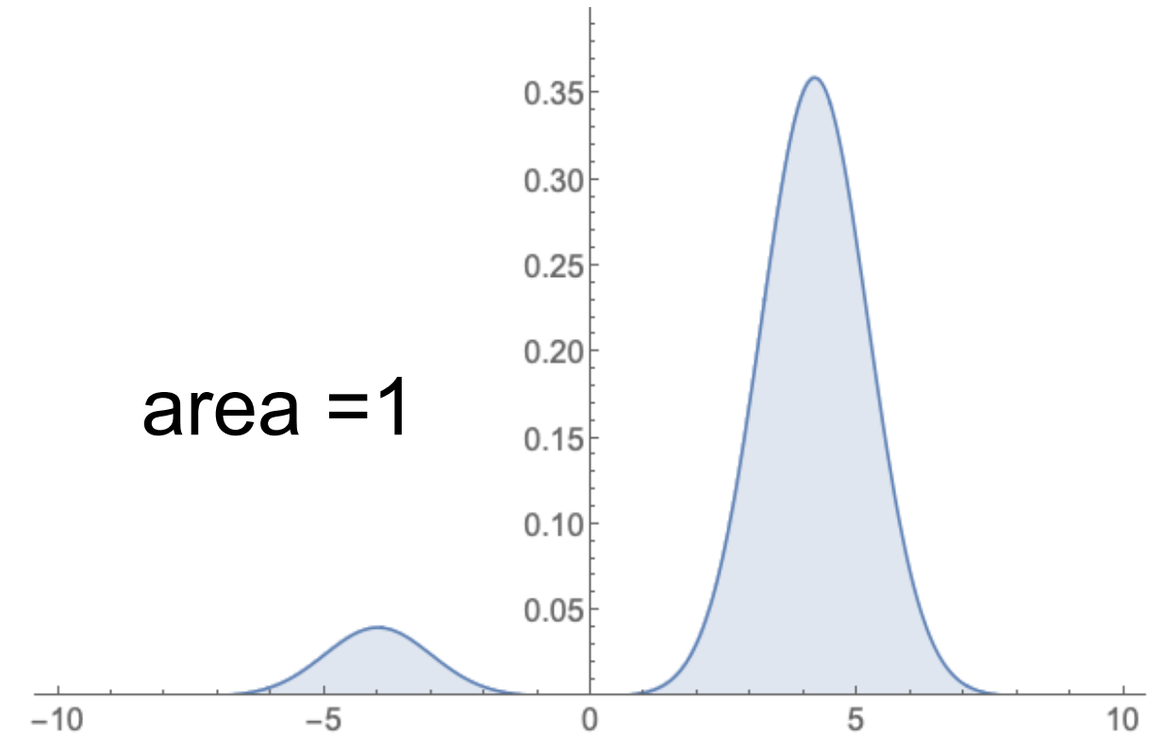
$$\frac{e^{f_\theta(x)}}{Z}$$



Score function

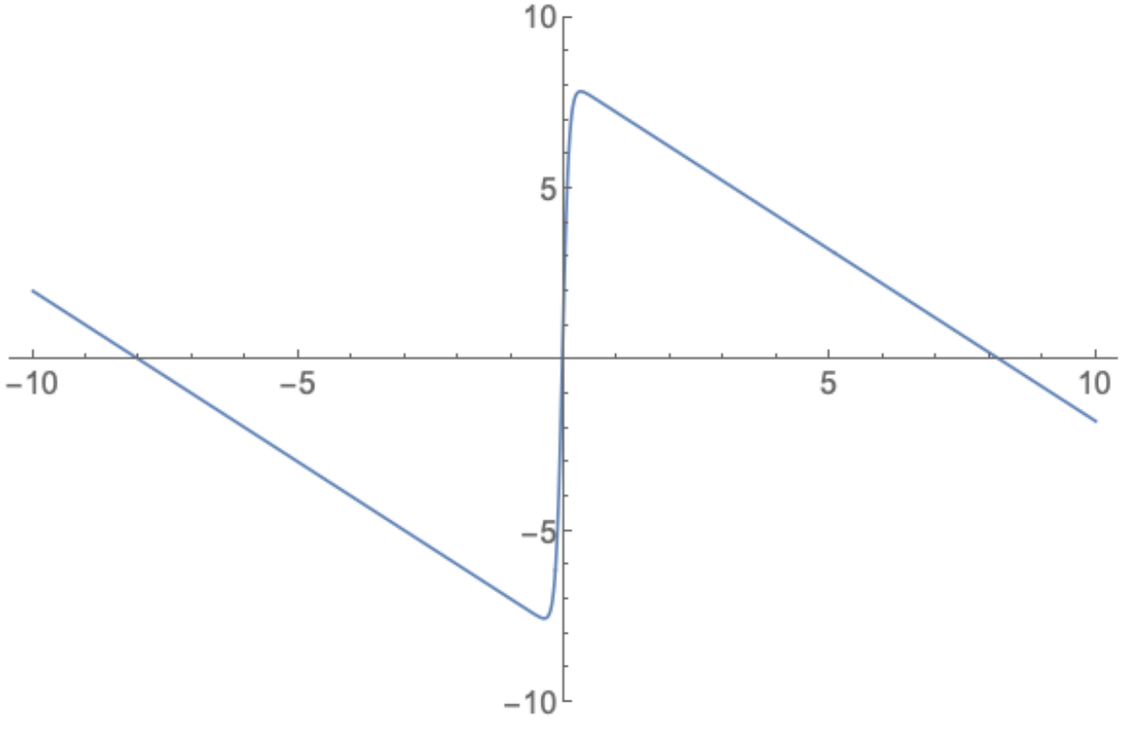
$$\nabla_{\mathbf{x}} \log p_\theta(\mathbf{x}) = \nabla_{\mathbf{x}} f_\theta(\mathbf{x}) - \nabla_{\mathbf{x}} \log Z_\theta$$

area =1



Probability density function

$$\frac{e^{f_\theta(x)}}{\cancel{Z}} = p_\theta(x)$$



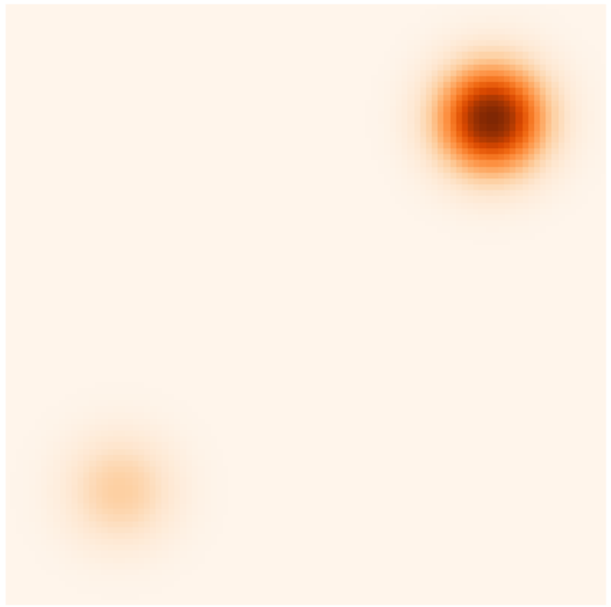
Score function

$$\nabla_{\mathbf{x}} \log p_\theta(\mathbf{x}) = \nabla_{\mathbf{x}} f_\theta(\mathbf{x}) - \nabla_{\mathbf{x}} \log Z_\theta$$

Score estimation by training score-based models

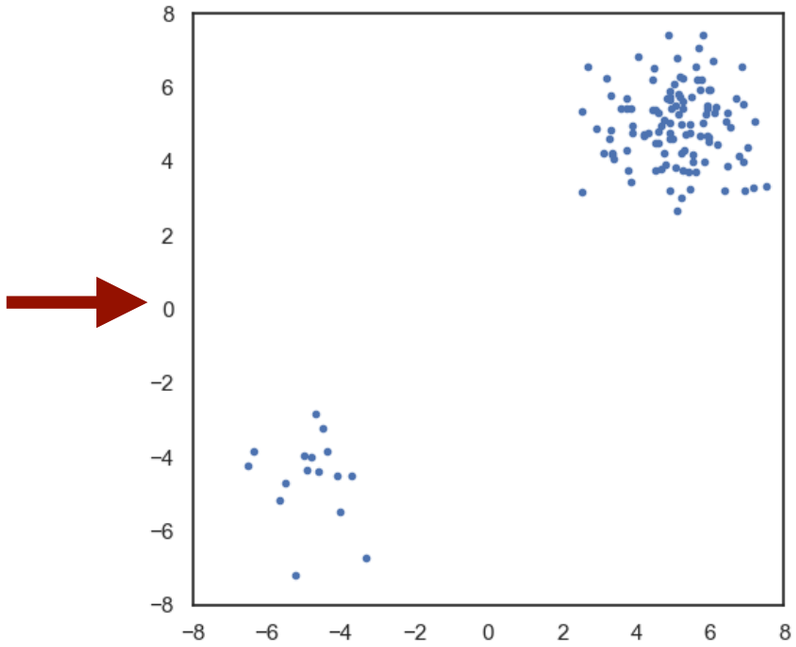
Probability density

$$p_{\text{data}}(\mathbf{x})$$



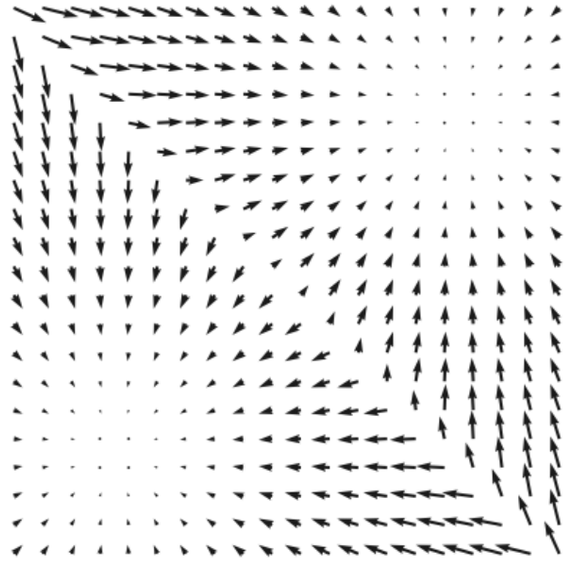
i.i.d. samples

$$\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n$$



Score function

$$\boldsymbol{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$$



Score estimation by training score-based models

Score estimation by training score-based models

- **Given:** i.i.d. samples

Score estimation by training score-based models

- **Given:** i.i.d. samples $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \sim p_{\text{data}}(\mathbf{x})$

Score estimation by training score-based models

- **Given:** i.i.d. samples $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \sim p_{\text{data}}(\mathbf{x})$
- **Task:** Estimating the score

Score estimation by training score-based models

- **Given:** i.i.d. samples $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \sim p_{\text{data}}(\mathbf{x})$
- **Task:** Estimating the score $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$

Score estimation by training score-based models

- **Given:** i.i.d. samples $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \sim p_{\text{data}}(\mathbf{x})$
- **Task:** Estimating the score $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$
- **Score Model:** A learnable vector-valued function

Score estimation by training score-based models

- **Given:** i.i.d. samples $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \sim p_{\text{data}}(\mathbf{x})$
- **Task:** Estimating the score $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$
- **Score Model:** A learnable vector-valued function $\boldsymbol{s}_\theta(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}^d$

Score estimation by training score-based models

- **Given:** i.i.d. samples $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \sim p_{\text{data}}(\mathbf{x})$
- **Task:** Estimating the score $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$
- **Score Model:** A learnable vector-valued function $\boldsymbol{s}_\theta(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}^d$
- **Goal:**

Score estimation by training score-based models

- **Given:** i.i.d. samples $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \sim p_{\text{data}}(\mathbf{x})$
- **Task:** Estimating the score $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$
- **Score Model:** A learnable vector-valued function $\boldsymbol{s}_\theta(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}^d$
- **Goal:** $\boldsymbol{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$

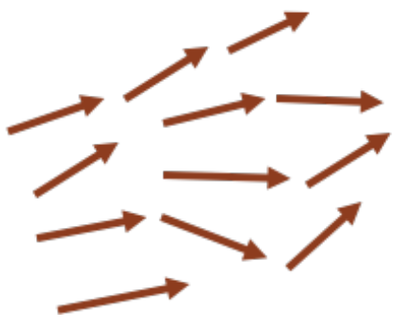
Score estimation by training score-based models

- **Given:** i.i.d. samples $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \sim p_{\text{data}}(\mathbf{x})$
- **Task:** Estimating the score $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$
- **Score Model:** A learnable vector-valued function $\boldsymbol{s}_\theta(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}^d$
- **Goal:** $\boldsymbol{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$
- How to compare two vector fields of scores?

Score estimation by training score-based models

- **Given:** i.i.d. samples $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \sim p_{\text{data}}(\mathbf{x})$
- **Task:** Estimating the score $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$
- **Score Model:** A learnable vector-valued function $\boldsymbol{s}_\theta(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}^d$
- **Goal:** $\boldsymbol{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$
- How to compare two vector fields of scores?

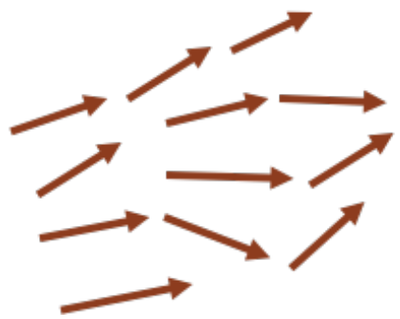
$\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$



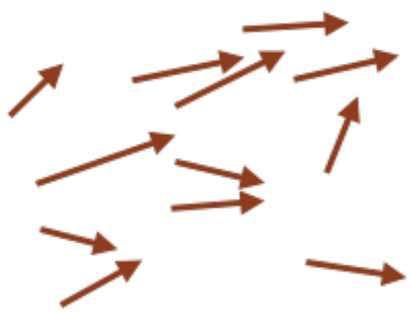
Score estimation by training score-based models

- **Given:** i.i.d. samples $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \sim p_{\text{data}}(\mathbf{x})$
- **Task:** Estimating the score $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$
- **Score Model:** A learnable vector-valued function $\boldsymbol{s}_\theta(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}^d$
- **Goal:** $\boldsymbol{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$
- How to compare two vector fields of scores?

$\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$



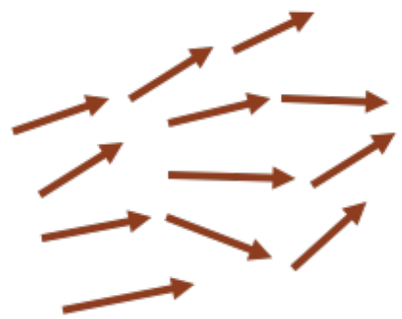
$\boldsymbol{s}_\theta(\mathbf{x})$



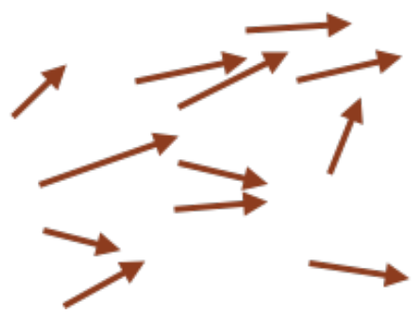
Score estimation by training score-based models

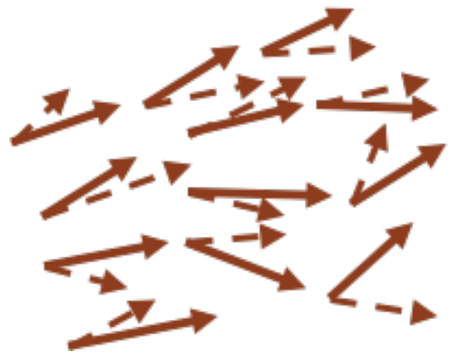
- **Given:** i.i.d. samples $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \sim p_{\text{data}}(\mathbf{x})$
- **Task:** Estimating the score $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$
- **Score Model:** A learnable vector-valued function $\boldsymbol{s}_\theta(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}^d$
- **Goal:** $\boldsymbol{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$
- How to compare two vector fields of scores?

$\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$



$\boldsymbol{s}_\theta(\mathbf{x})$





Average
Euclidean distance
over the space

- **Objective:** Average Euclidean distance over the whole space.

- **Objective:** Average Euclidean distance over the whole space.

$$\frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}} [\| \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) - \boldsymbol{s}_\theta(\mathbf{x}) \|_2^2]$$

(Fisher divergence)

- **Objective:** Average Euclidean distance over the whole space.

$$\frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}} [\| \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) - \boldsymbol{s}_\theta(\mathbf{x}) \|_2^2]$$

(Fisher divergence)

- **Objective:** Average Euclidean distance over the whole space.

$$\frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}} [\| \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) - \boldsymbol{s}_\theta(\mathbf{x}) \|_2^2]$$

(Fisher divergence)

- **Objective:** Average Euclidean distance over the whole space.

$$\frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}} [\| \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) - \boldsymbol{s}_\theta(\mathbf{x}) \|_2^2]$$

(Fisher divergence)

- **Score matching [Hyv¨arinen 2016]:**

- **Objective:** Average Euclidean distance over the whole space.

$$\frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}} [\| \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) - \boldsymbol{s}_\theta(\mathbf{x}) \|_2^2]$$

(Fisher divergence)

- **Score matching [Hyv¨arinen 2016]:**

$$E_{\mathbf{x} \sim p_{\text{data}}} \left[\frac{1}{2} \| \boldsymbol{s}_\theta(\mathbf{x}) \|_2^2 + \text{tr}(\underbrace{\nabla_{\mathbf{x}} \boldsymbol{s}_\theta(\mathbf{x})}_{\text{Jacobian of } \boldsymbol{s}_\theta(\mathbf{x})}) \right]$$

- **Objective:** Average Euclidean distance over the whole space.

$$\frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}} [\| \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) - \boldsymbol{s}_\theta(\mathbf{x}) \|_2^2]$$

(Fisher divergence)

- **Score matching [Hyv¨arinen 2016]:**

$$E_{\mathbf{x} \sim p_{\text{data}}} \left[\frac{1}{2} \| \boldsymbol{s}_\theta(\mathbf{x}) \|_2^2 + \text{tr}(\underbrace{\nabla_{\mathbf{x}} \boldsymbol{s}_\theta(\mathbf{x})}_{\text{Jacobian of } \boldsymbol{s}_\theta(\mathbf{x})}) \right]$$

- **Objective:** Average Euclidean distance over the whole space.

$$\frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}} [\| \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) - \boldsymbol{s}_\theta(\mathbf{x}) \|_2^2]$$

(Fisher divergence)

- **Score matching [Hyv¨arinen 2016]:**

$$E_{\mathbf{x} \sim p_{\text{data}}} \left[\frac{1}{2} \| \boldsymbol{s}_\theta(\mathbf{x}) \|_2^2 + \text{tr}(\underbrace{\nabla_{\mathbf{x}} \boldsymbol{s}_\theta(\mathbf{x})}_{\text{Jacobian of } \boldsymbol{s}_\theta(\mathbf{x})}) \right]$$

- **Requirements:**

- **Objective:** Average Euclidean distance over the whole space.

$$\frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}} [\| \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) - \boldsymbol{s}_\theta(\mathbf{x}) \|_2^2]$$

(Fisher divergence)

- **Score matching [Hyv¨arinen 2016]:**

$$E_{\mathbf{x} \sim p_{\text{data}}} \left[\frac{1}{2} \| \boldsymbol{s}_\theta(\mathbf{x}) \|_2^2 + \text{tr}(\underbrace{\nabla_{\mathbf{x}} \boldsymbol{s}_\theta(\mathbf{x})}_{\text{Jacobian of } \boldsymbol{s}_\theta(\mathbf{x})}) \right]$$

- **Requirements:**
  - The score model must be efficient to evaluate.

- **Objective:** Average Euclidean distance over the whole space.

$$\frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}} [\| \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) - \boldsymbol{s}_\theta(\mathbf{x}) \|_2^2]$$

(Fisher divergence)

- **Score matching [Hyv¨arinen 2016]:**

$$E_{\mathbf{x} \sim p_{\text{data}}} \left[\frac{1}{2} \| \boldsymbol{s}_\theta(\mathbf{x}) \|_2^2 + \text{tr}(\underbrace{\nabla_{\mathbf{x}} \boldsymbol{s}_\theta(\mathbf{x})}_{\text{Jacobian of } \boldsymbol{s}_\theta(\mathbf{x})}) \right]$$

- **Requirements:**
  - The score model must be efficient to evaluate.
  - Do we need the score model to be a proper score function (i.e., gradient of a scalar “energy” function)?

- **Objective:** Average Euclidean distance over the whole space.

$$\frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}} [\| \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) - \boldsymbol{s}_\theta(\mathbf{x}) \|_2^2]$$

(Fisher divergence)

- **Score matching [Hyv¨arinen 2016]:**

$$E_{\mathbf{x} \sim p_{\text{data}}} \left[\frac{1}{2} \| \boldsymbol{s}_\theta(\mathbf{x}) \|_2^2 + \text{tr}\left(\underbrace{\nabla_{\mathbf{x}} \boldsymbol{s}_\theta(\mathbf{x})}_{\text{Jacobian of } \boldsymbol{s}_\theta(\mathbf{x})} \right) \right]$$

- **Requirements:**
  - The score model must be efficient to evaluate.
  - Do we need the score model to be a proper score function (i.e., gradient of a scalar “energy” function)?

- **Objective:** Average Euclidean distance over the whole space.

$$\frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}} [\| \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) - \boldsymbol{s}_\theta(\mathbf{x}) \|_2^2]$$

(Fisher divergence)

- **Score matching [Hyv¨arinen 2016]:**

$$E_{\mathbf{x} \sim p_{\text{data}}} \left[\frac{1}{2} \| \boldsymbol{s}_\theta(\mathbf{x}) \|_2^2 + \text{tr}\left(\underbrace{\nabla_{\mathbf{x}} \boldsymbol{s}_\theta(\mathbf{x})}_{\text{Jacobian of } \boldsymbol{s}_\theta(\mathbf{x})} \right) \right]$$

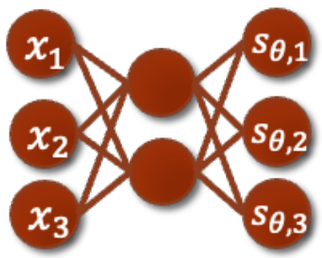
$O(d)$ Backprops!

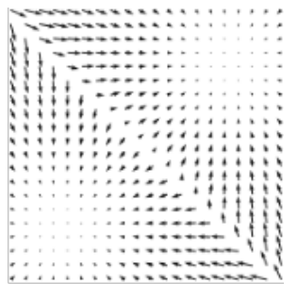
- **Requirements:**
  - The score model must be efficient to evaluate.
  - Do we need the score model to be a proper score function (i.e., gradient of a scalar “energy” function)?

Score-based models

- A model that represents the score function

$$s_\theta(\mathbf{x})$$





- **Score estimation:** training the score-based model from datapoints
- Score matching

$$\frac{1}{2}E_{\mathbf{x}\sim p_{\text{data}}}[\|\nabla_{\mathbf{x}}\log p_{\text{data}}(\mathbf{x}) - \boldsymbol{s}_\theta(\mathbf{x})\|_2^2]$$
$$=\frac{1}{2}E_{\mathbf{x}\sim p_{\text{data}}}\left[\frac{1}{2}\|\boldsymbol{s}_\theta(\mathbf{x})\|_2^2 + \text{tr}(\nabla_{\mathbf{x}}\boldsymbol{s}_\theta(\mathbf{x}))\right] + \text{const.}$$

- Not scalable for deep score-based models and high dimensional data

Denoising score matching (Vincent, 2011)

$\mathbf{x} \sim p_{\text{data}}(\mathbf{x})$

Data distribution

$q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})$

Perturbation distribution/kernel

$\tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbf{x}})$

Noise-perturbed data distribution

Denoising score matching (Vincent, 2011)

$\mathbf{x} \sim p_{\text{data}}(\mathbf{x})$

Data distribution

$q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})$

Perturbation distribution/kernel

$\tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbf{x}})$

Noise-perturbed data distribution

$$\frac{1}{2} E_{\tilde{\mathbf{x}} \sim p_{\text{data}}}[\| \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}) \|_2^2]$$

$$= \frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x}), \tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})}[\| \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) \|_2^2] + \text{const.}$$

Denoising score matching (Vincent, 2011)

$\mathbf{x} \sim p_{\text{data}}(\mathbf{x})$

Data distribution

$q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})$

Perturbation distribution/kernel

$\tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbf{x}})$

Noise-perturbed data distribution

$$\frac{1}{2} E_{\tilde{\mathbf{x}} \sim p_{\text{data}}} [\| \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}) \|_2^2]$$

$$= \frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x}), \tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})} [\| \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) \|_2^2] + \text{const.}$$

$$= \frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}, \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})} \left[\left\| \boldsymbol{s}_\theta(\mathbf{x} + \sigma \mathbf{z}) + \frac{\mathbf{z}}{\sigma} \right\|_2^2 \right] + \text{const.}$$

Denoising score matching (Vincent, 2011)

$\mathbf{x} \sim p_{\text{data}}(\mathbf{x})$

Data distribution

$q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})$

Perturbation distribution/kernel

$\tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbf{x}})$

Noise-perturbed data distribution

$$\frac{1}{2} E_{\tilde{\mathbf{x}} \sim p_{\text{data}}} [\| \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}) \|_2^2]$$

$$= \frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x}), \tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})} [\| \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) \|_2^2] + \text{const.}$$

$$= \frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}, \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})} \left[\left\| \boldsymbol{s}_\theta(\mathbf{x} + \sigma \mathbf{z}) + \frac{\mathbf{z}}{\sigma} \right\|_2^2 \right] + \text{const.}$$

- **Pros:**
  - Much more scalable than score matching
  - Reduces score estimation to a denoising task
- **Con:** estimates score of noise-perturbed data

Denoising score matching (Vincent, 2011)

$\mathbf{x} \sim p_{\text{data}}(\mathbf{x})$

Data distribution

$q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})$

Perturbation distribution/kernel

$\tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbf{x}})$

Noise-perturbed data distribution

$$\frac{1}{2} E_{\tilde{\mathbf{x}} \sim p_{\text{data}}} [\| \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}) \|_2^2]$$

$$= \frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x}), \tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})} [\| \boldsymbol{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) \|_2^2] + \text{const.}$$

$$= \frac{1}{2} E_{\mathbf{x} \sim p_{\text{data}}, \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})} \left[\left\| \boldsymbol{s}_\theta(\mathbf{x} + \sigma \mathbf{z}) + \frac{\mathbf{z}}{\sigma} \right\|_2^2 \right] + \text{const.}$$

- **Pros:**
  - Much more scalable than score matching
  - Reduces score estimation to a denoising task
- **Con:** estimates score of noise-perturbed data

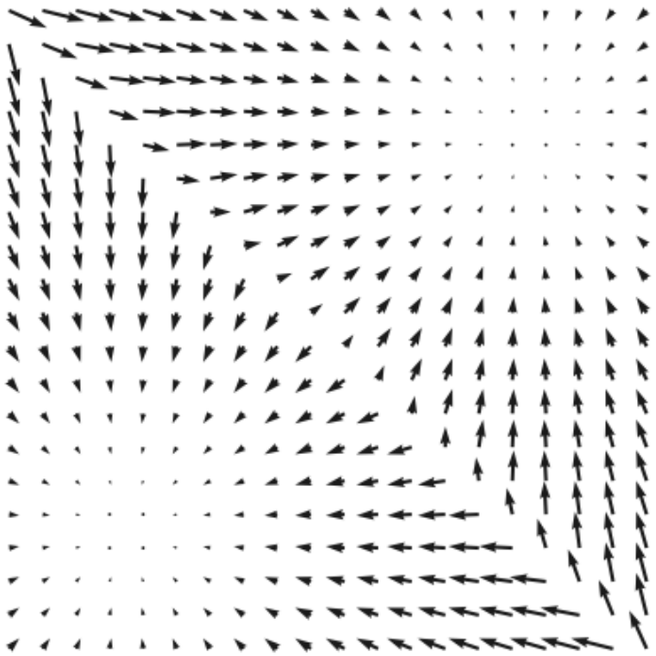
$$\boldsymbol{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log q_\sigma(\mathbf{x}) \neq \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$$

Why denoising is so powerful?

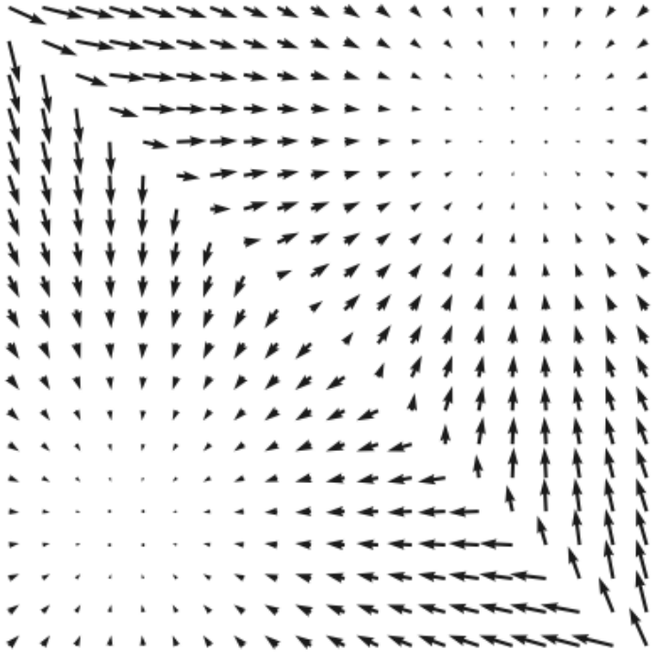
- Because it estimates the gradients of the log likelihood of data and neural nets are great at following gradients!

Quiz!

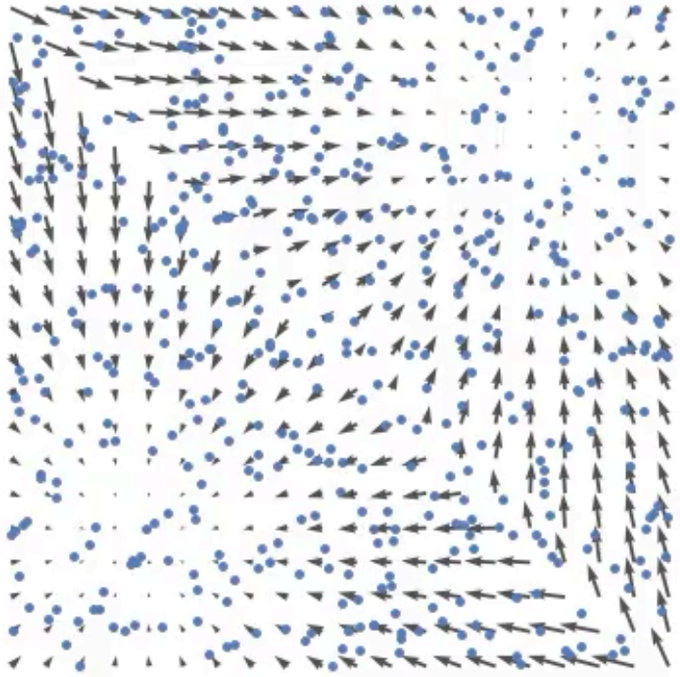
- What makes score function estimation challenging for high-dimensional data?



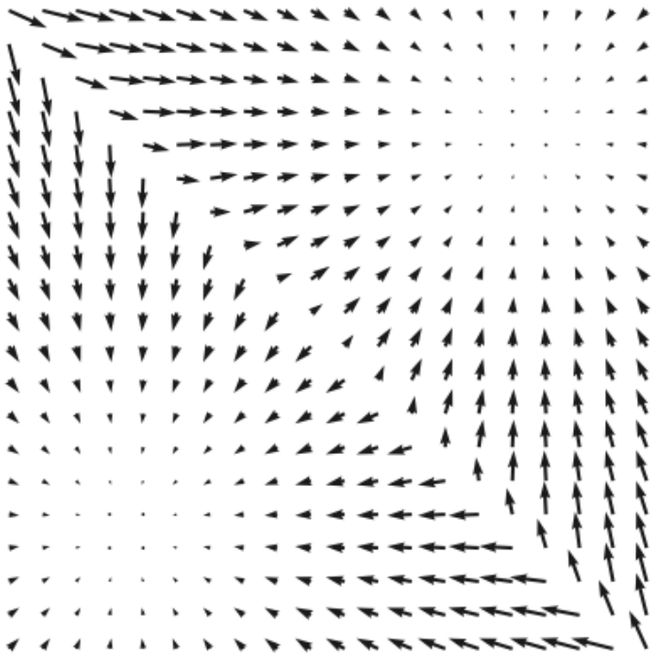
Score function



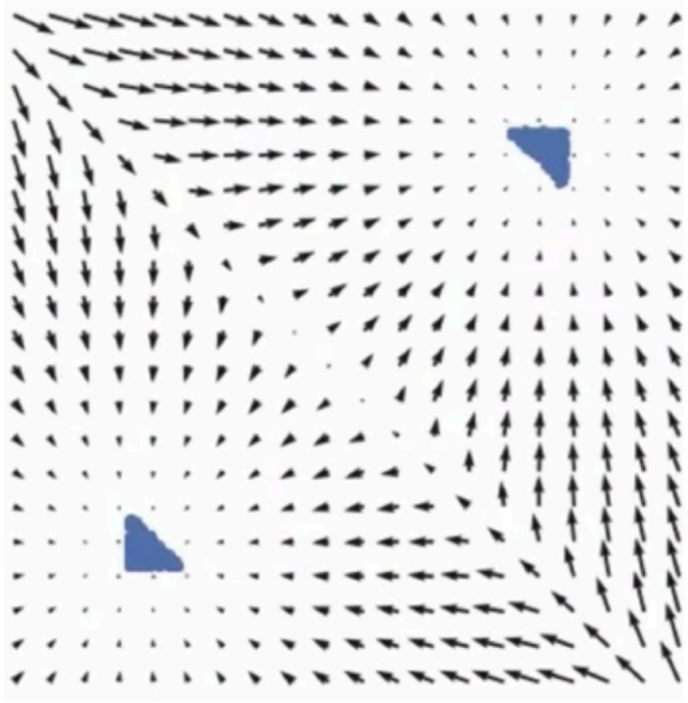
Score function



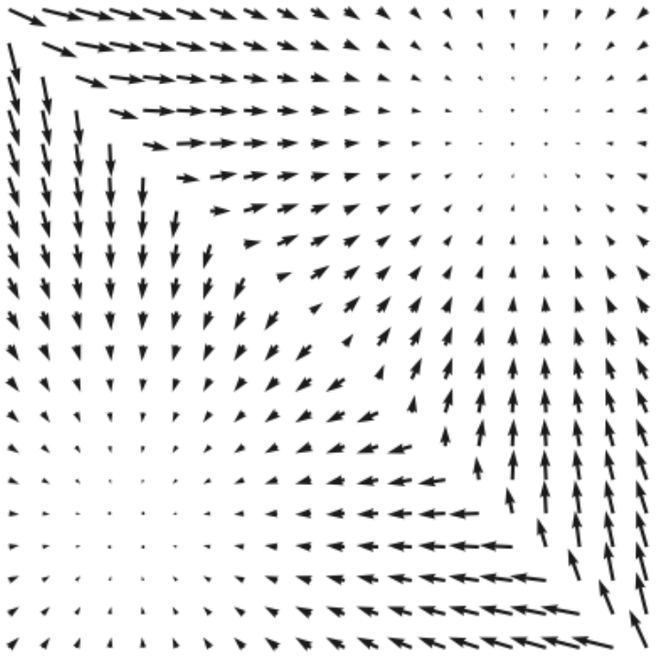
Follow the scores



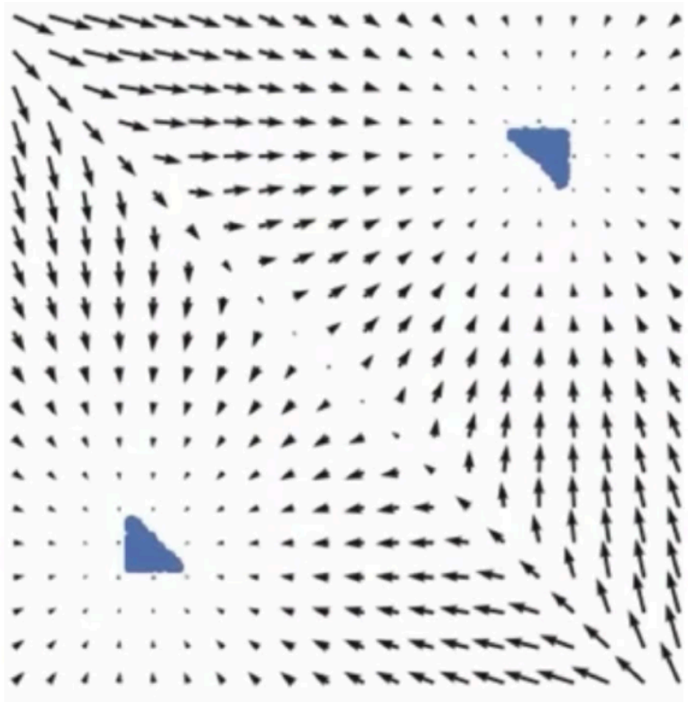
Score function



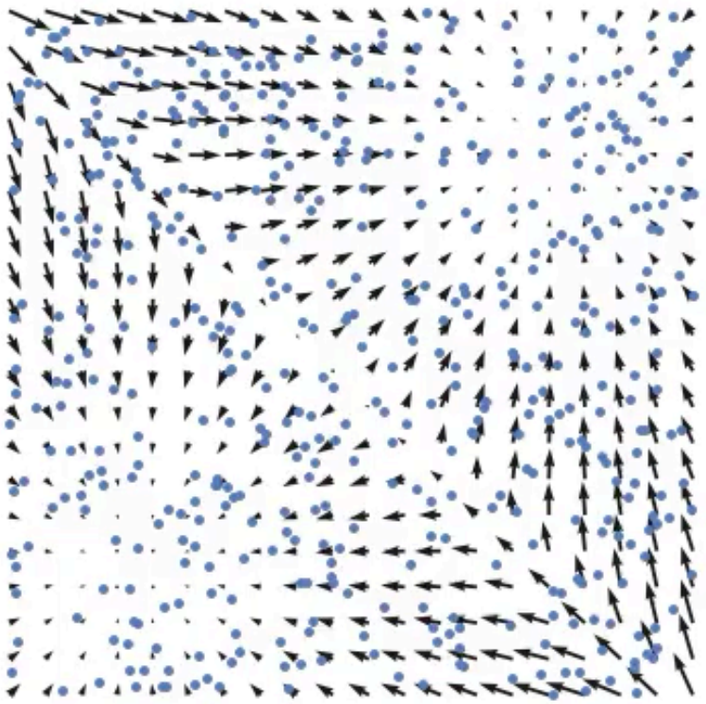
Follow the scores



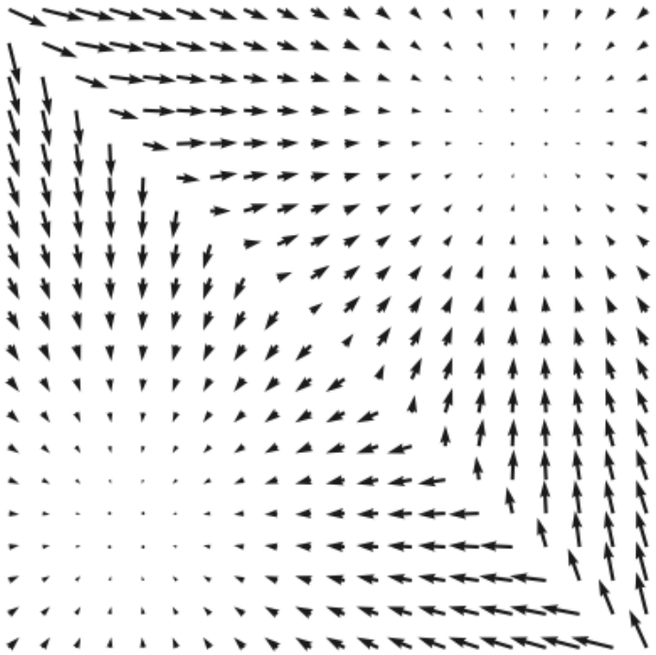
Score function



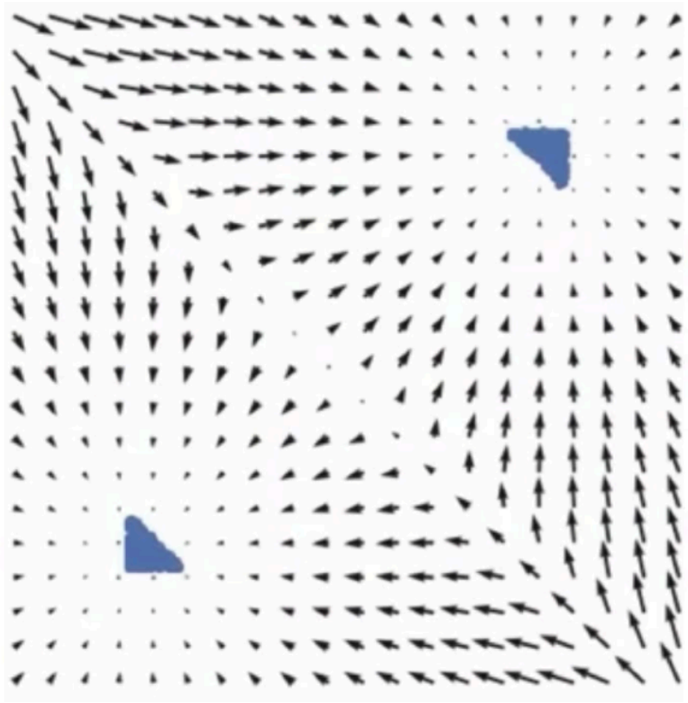
Follow the scores



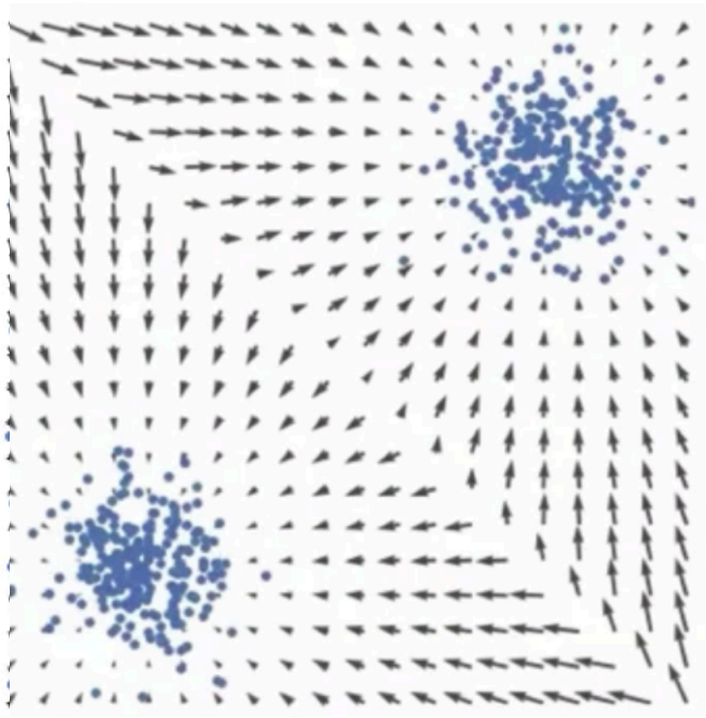
Follow the noisy



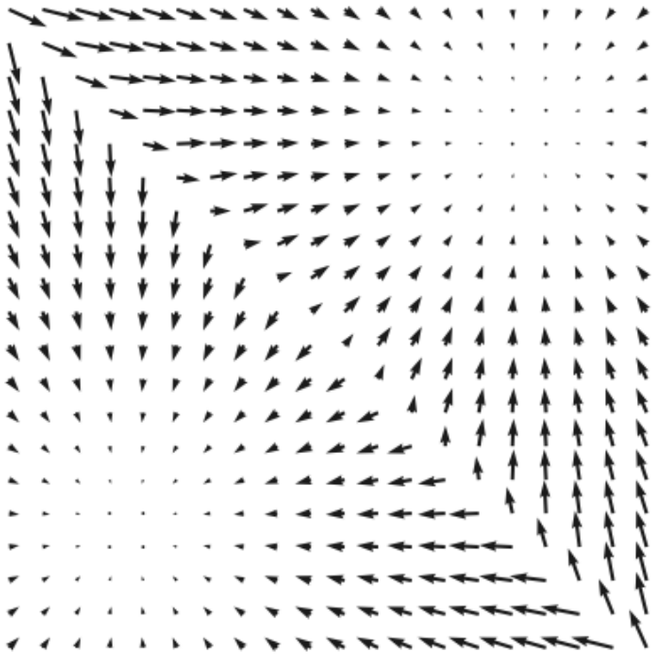
Score function



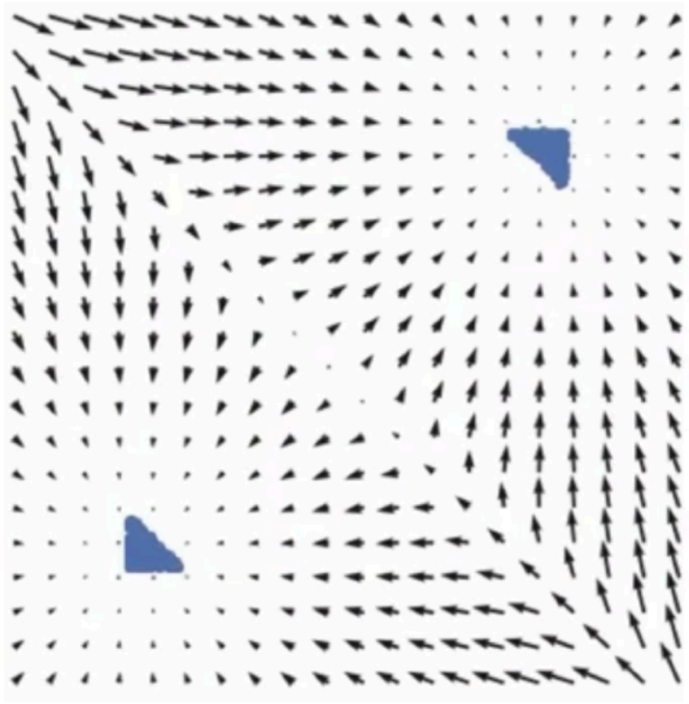
Follow the scores



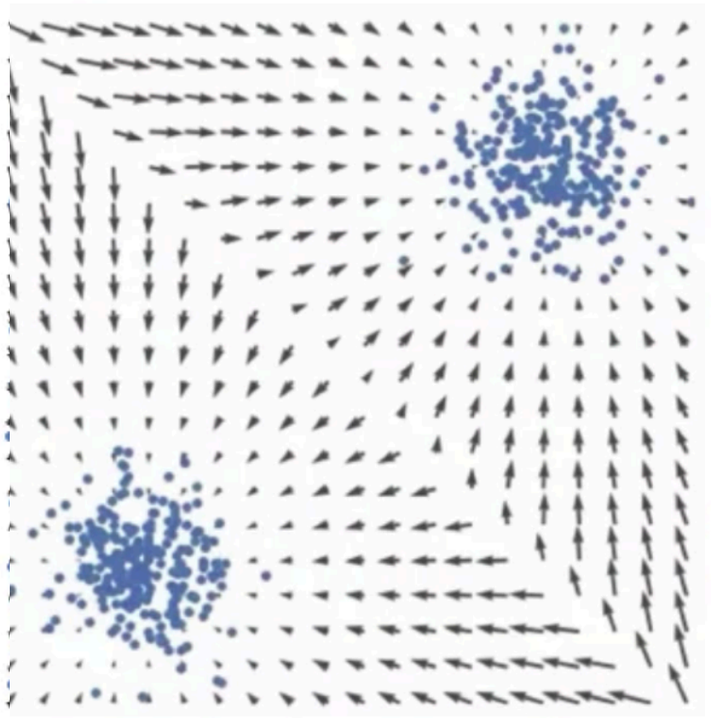
Follow the noisy



Score function

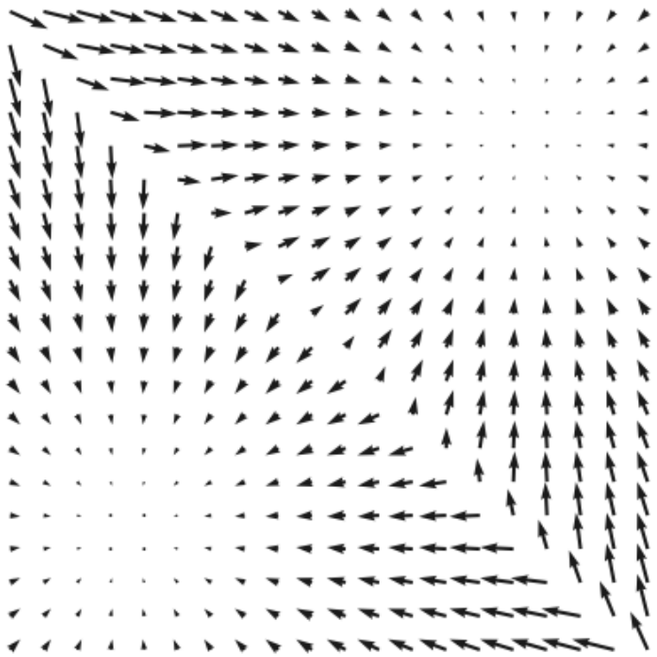


Follow the scores

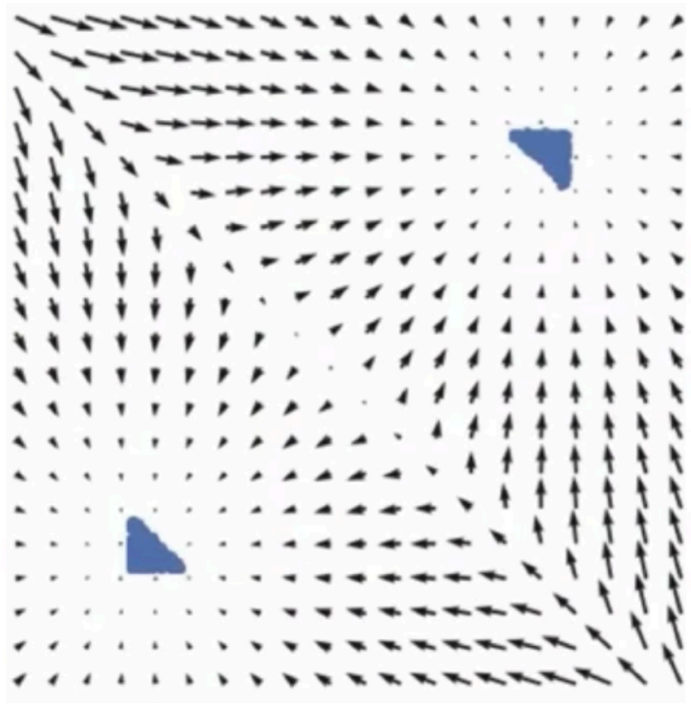


Follow the noisy

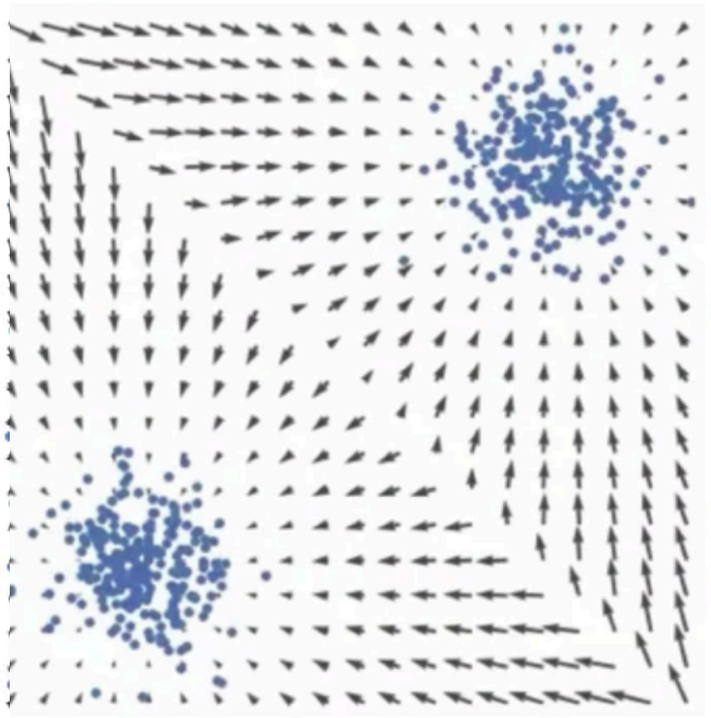
Langevin dynamics [Parisi 1981]



Score function



Follow the scores

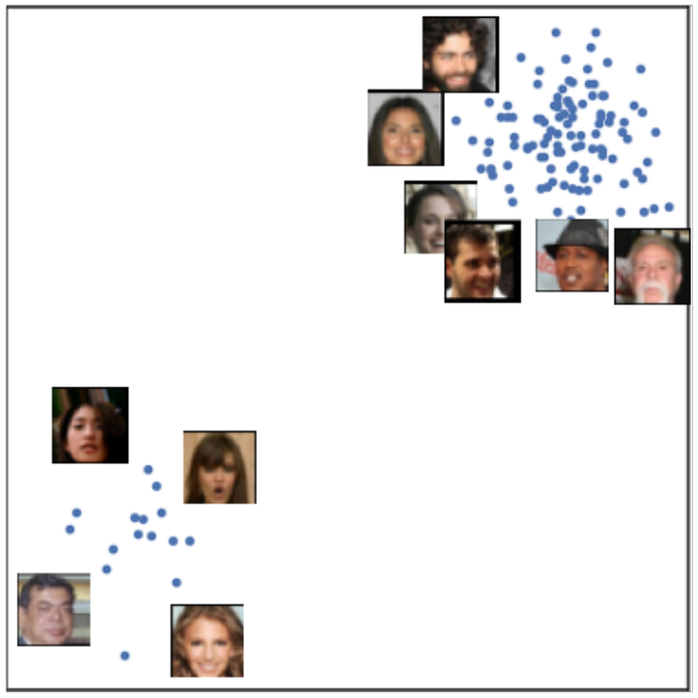


Follow the noisy

Langevin dynamics [Parisi 1981]

Guaranteed to produce correct samples.

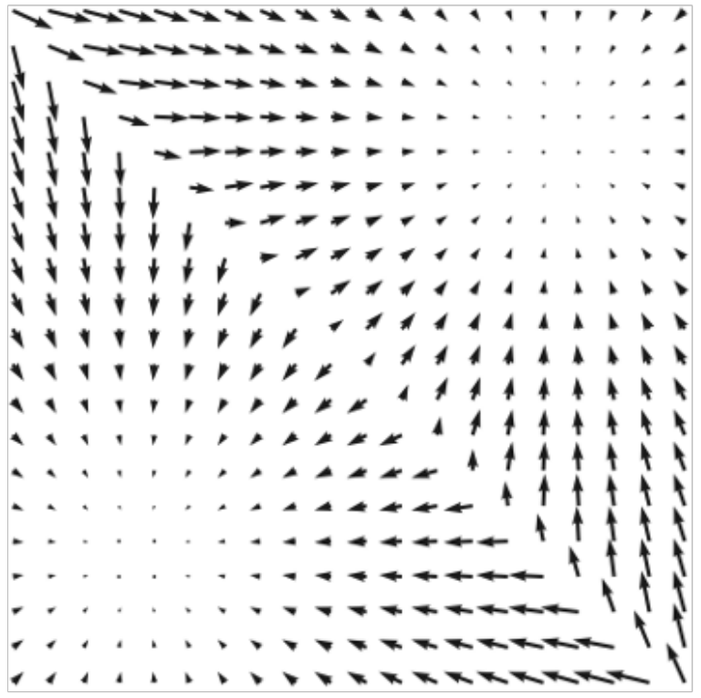
Score-based generative modelling



Data samples

$$\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \sim p_{\text{data}}(\mathbf{x})$$

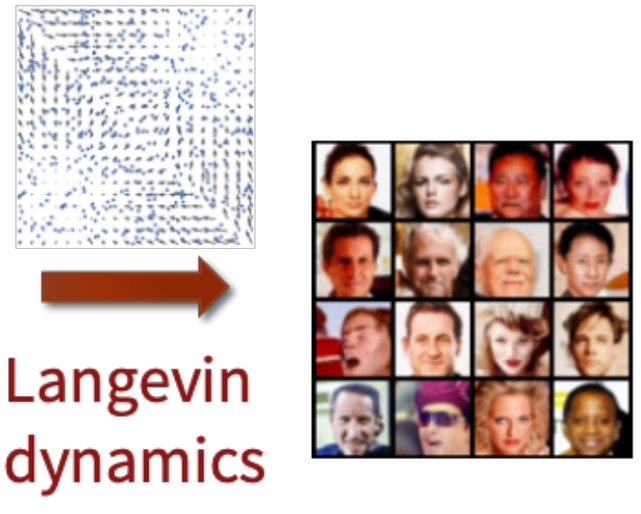
score matching



Scores

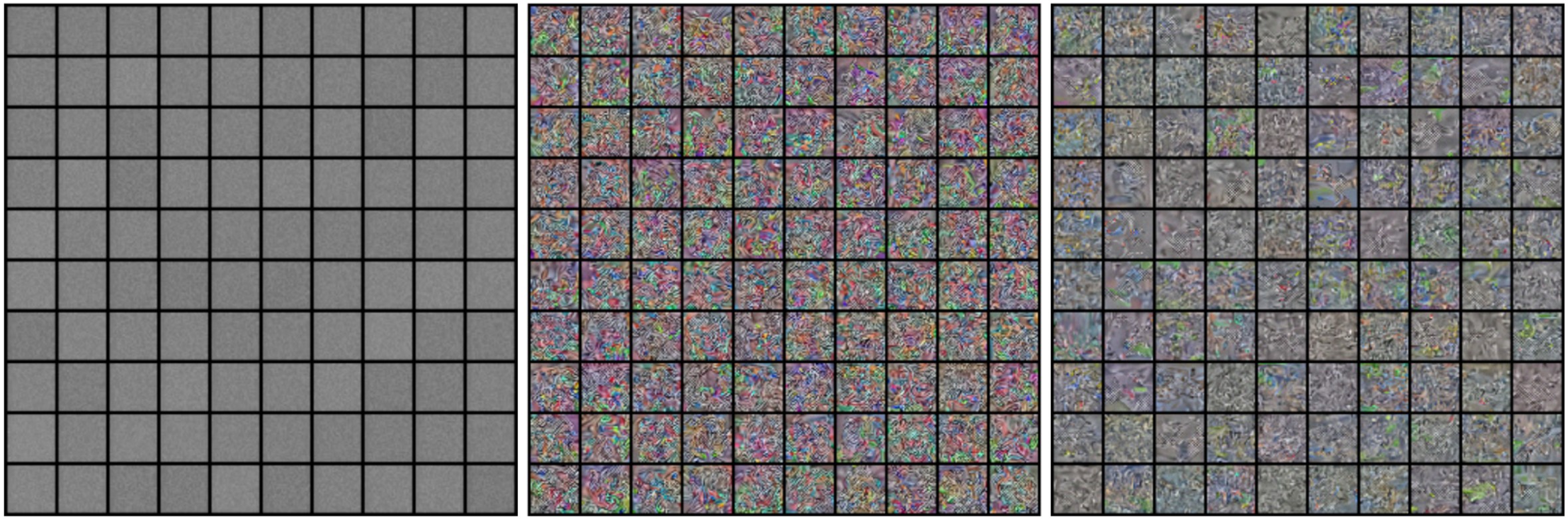
$$\boldsymbol{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$$

Langevin dynamics



New samples

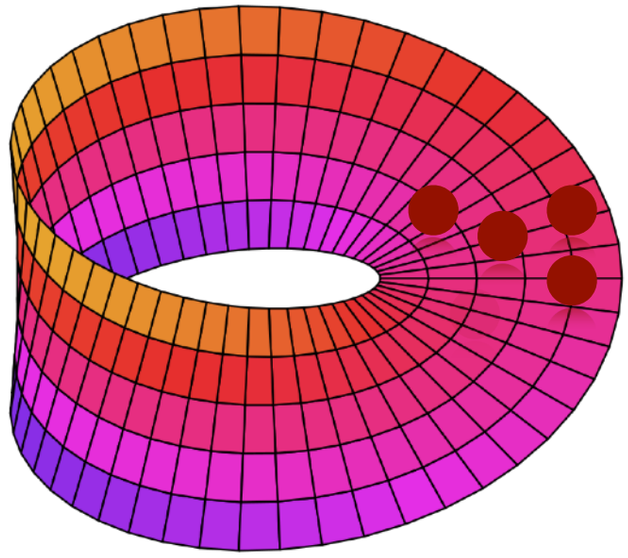
Score-based generative modelling results

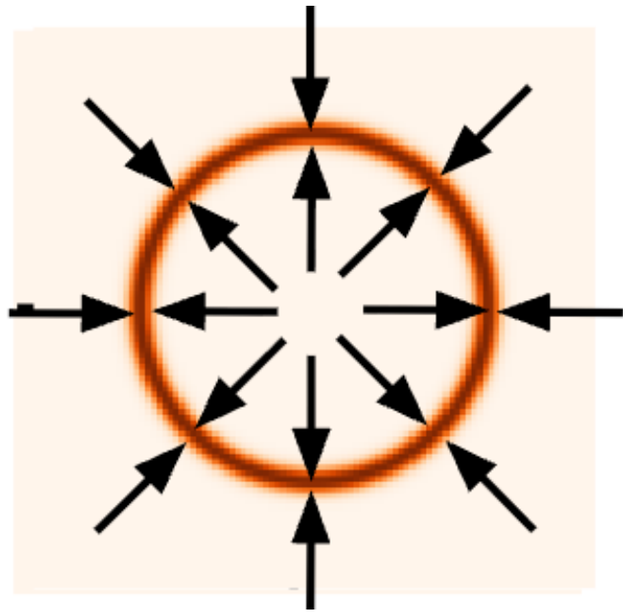


(a) MNIST (b) CelebA (c) CIFAR-10

Pitfall 1: manifold hypothesis

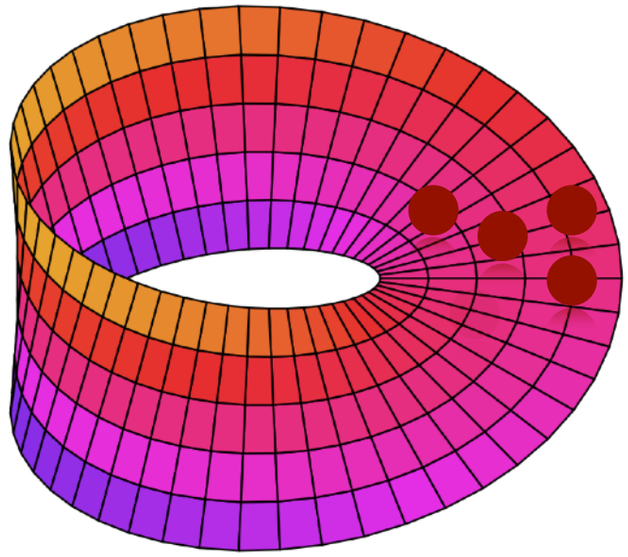
In real world, data points lie in low-dimensional spaces with a much smaller dimension than the data dimension.





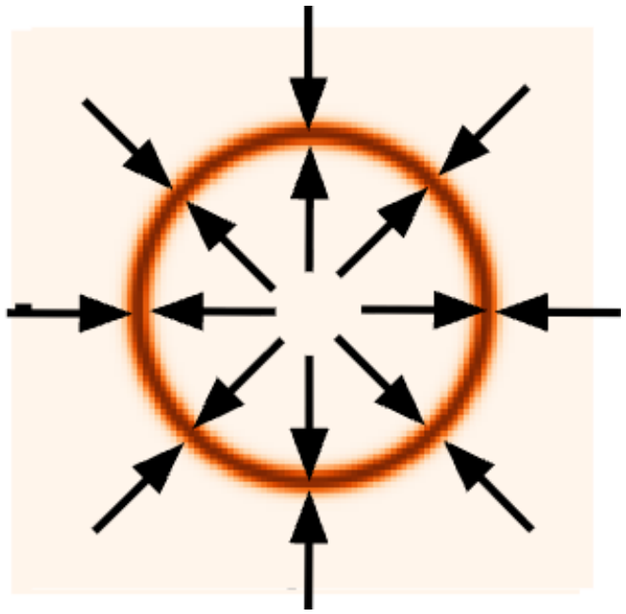
Pitfall 1: manifold hypothesis

In real world, data points lie in low-dimensional spaces with a much smaller dimension than the data dimension.

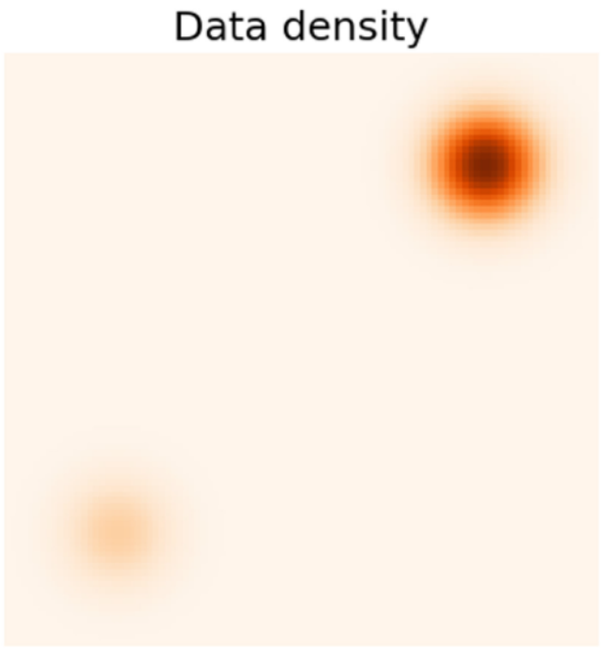


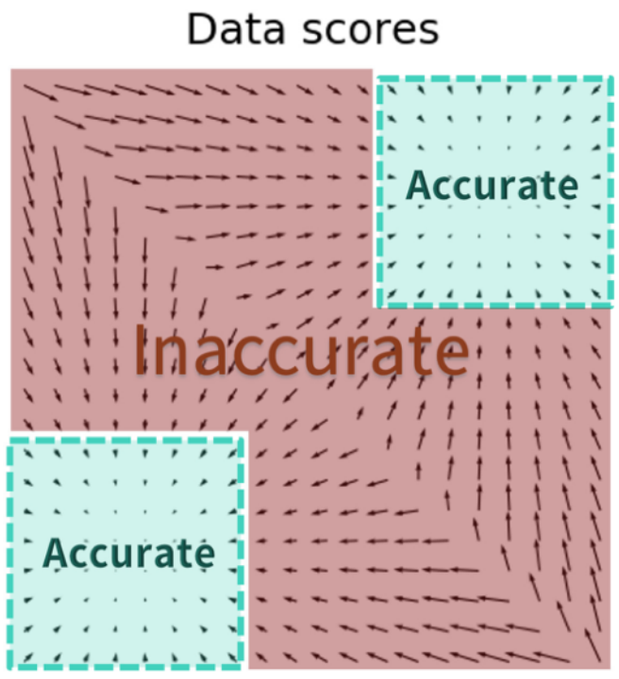
Data score is undefined.

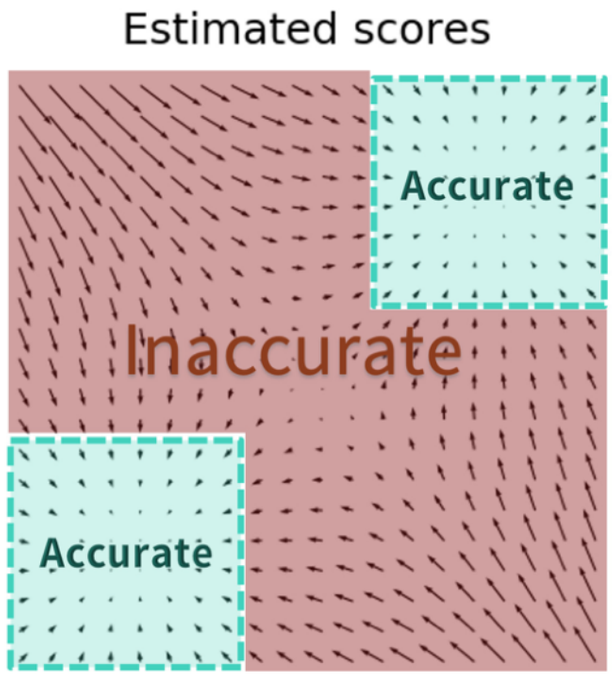
$$\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$$



Pitfall 2

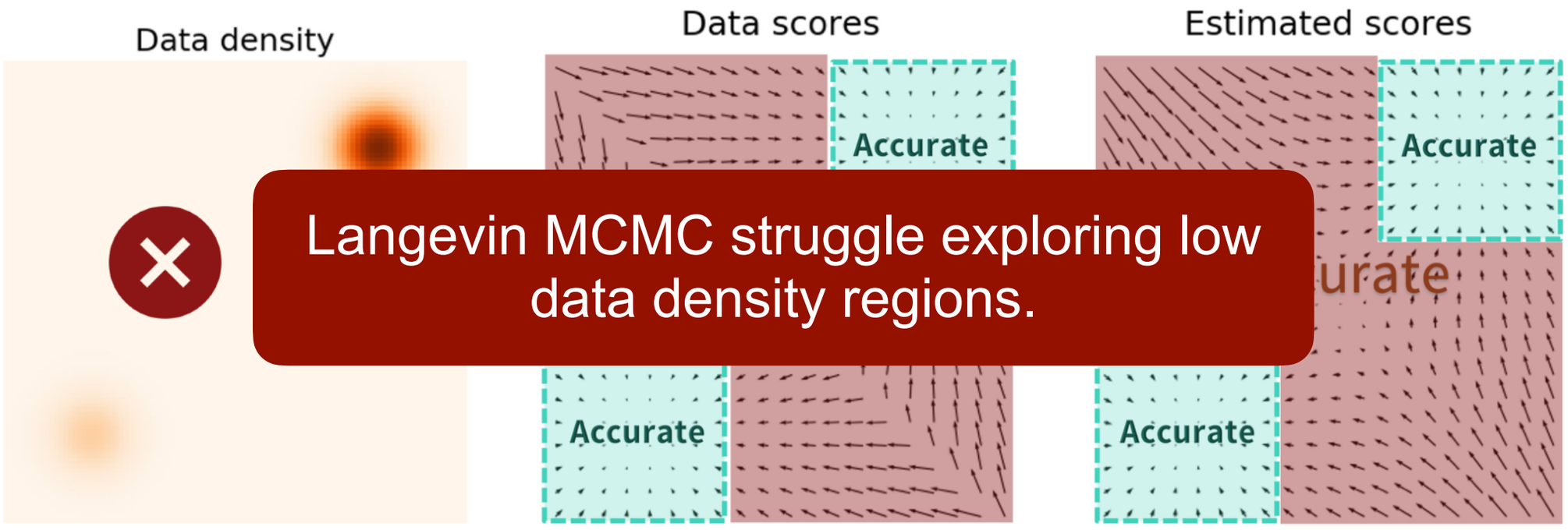






[Song and Ermon, 2019]

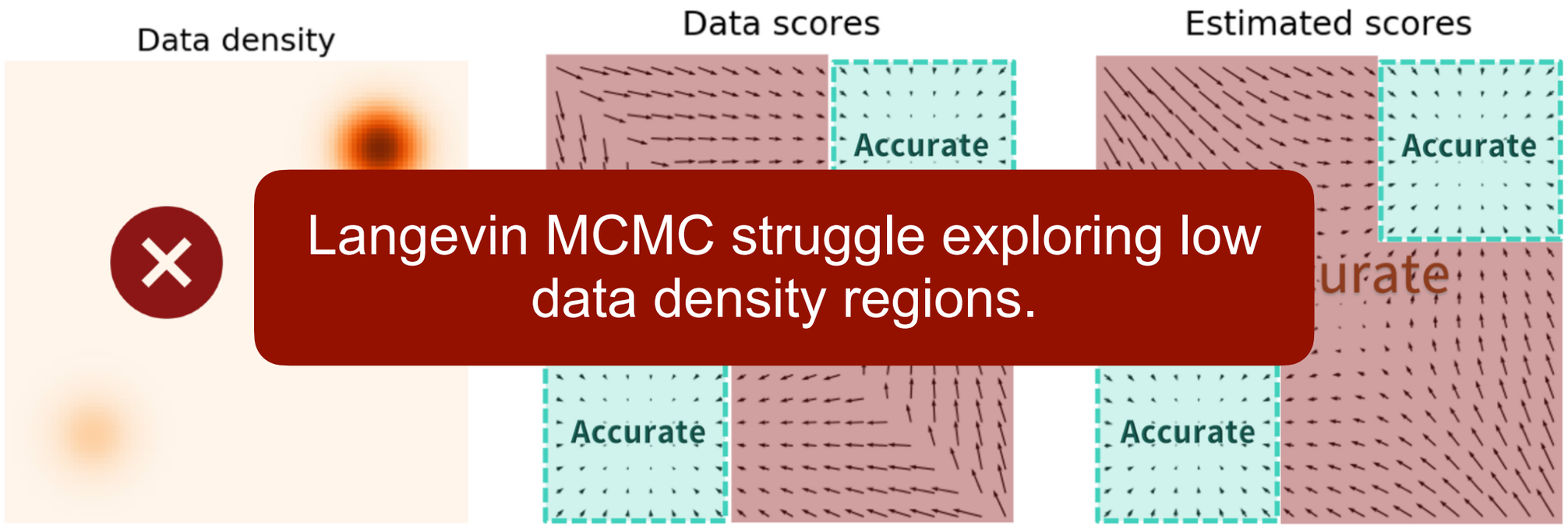
Pitfall 2

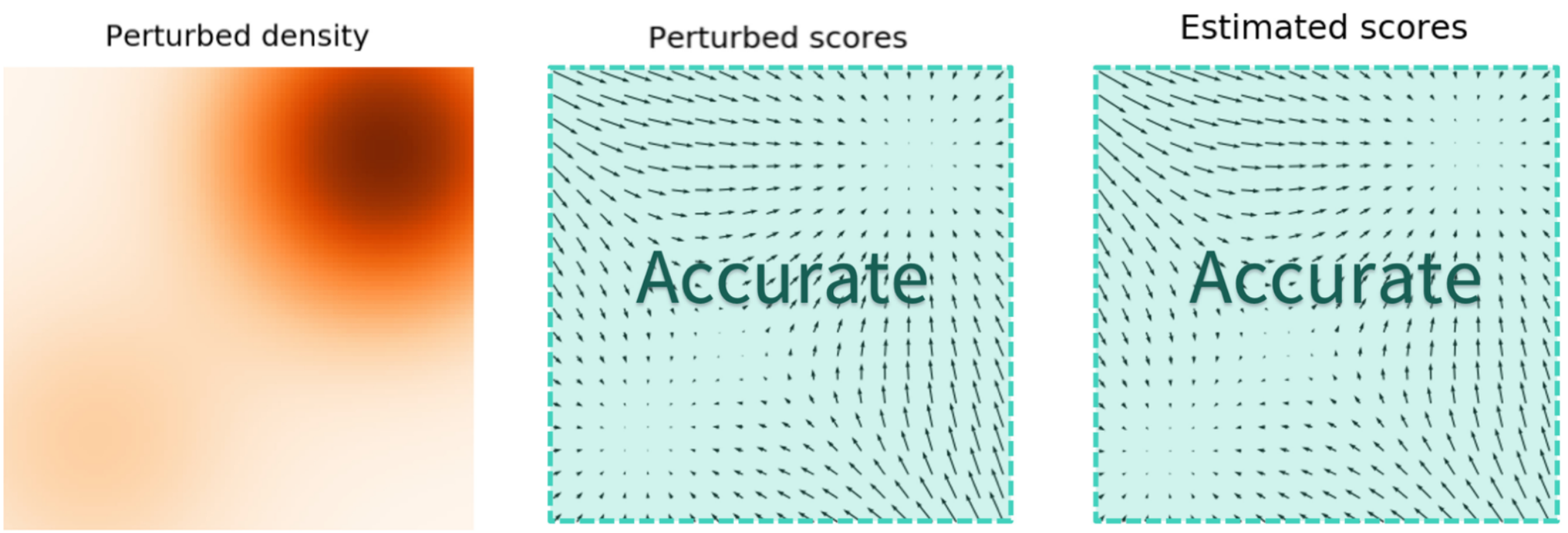


Langevin MCMC struggle exploring low data density regions.

[Song and Ermon, 2019]

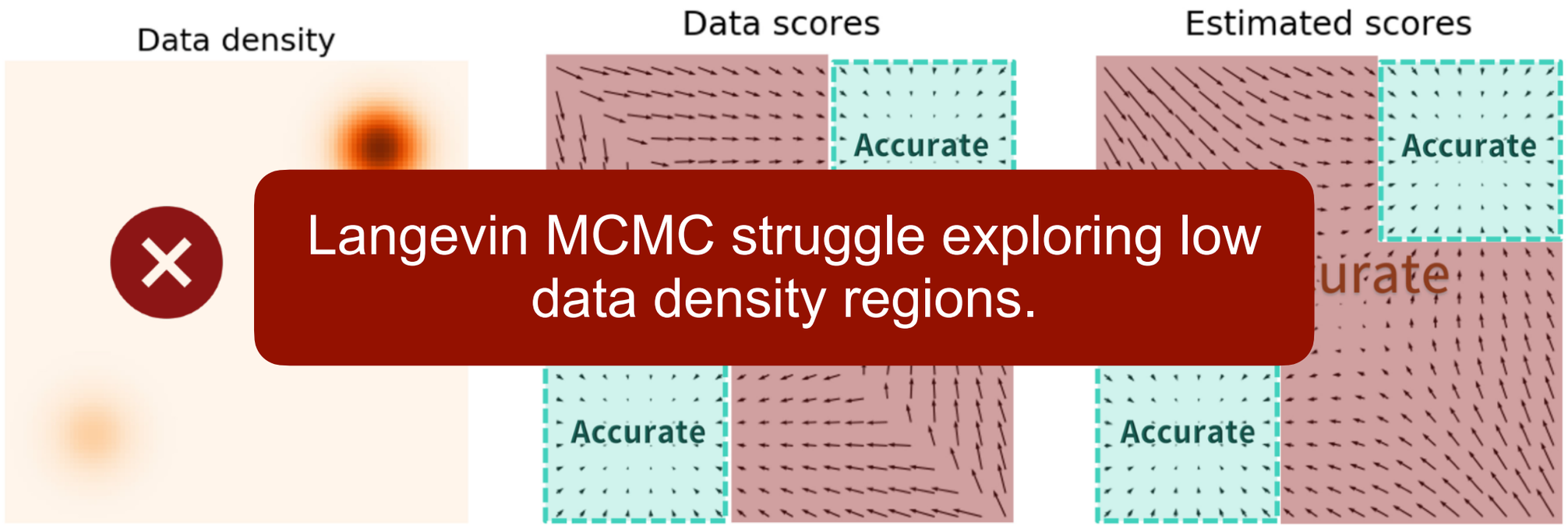
Pitfall 2

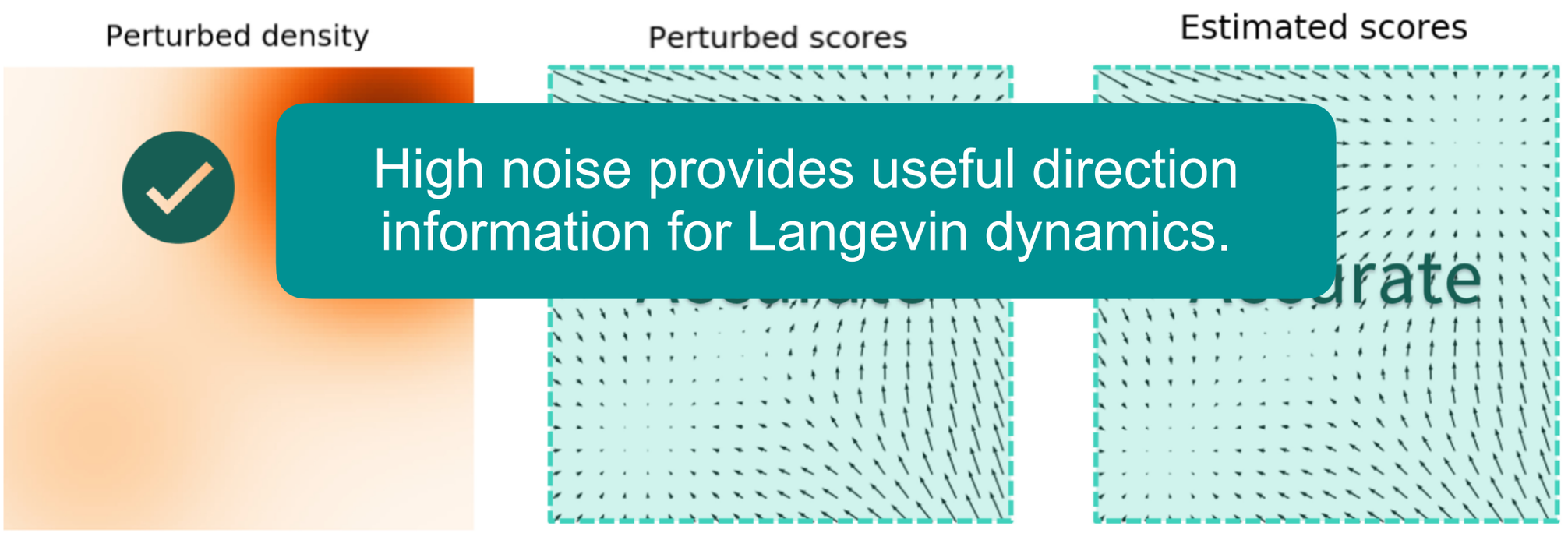




[Song and Ermon, 2019]

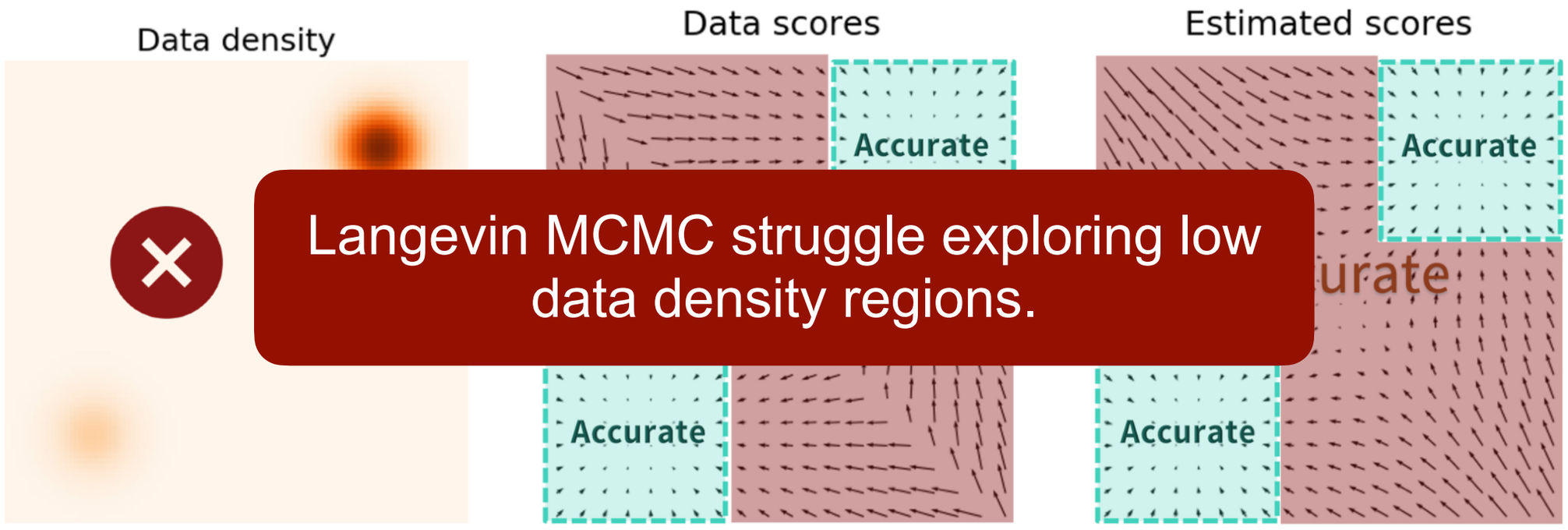
Pitfall 2





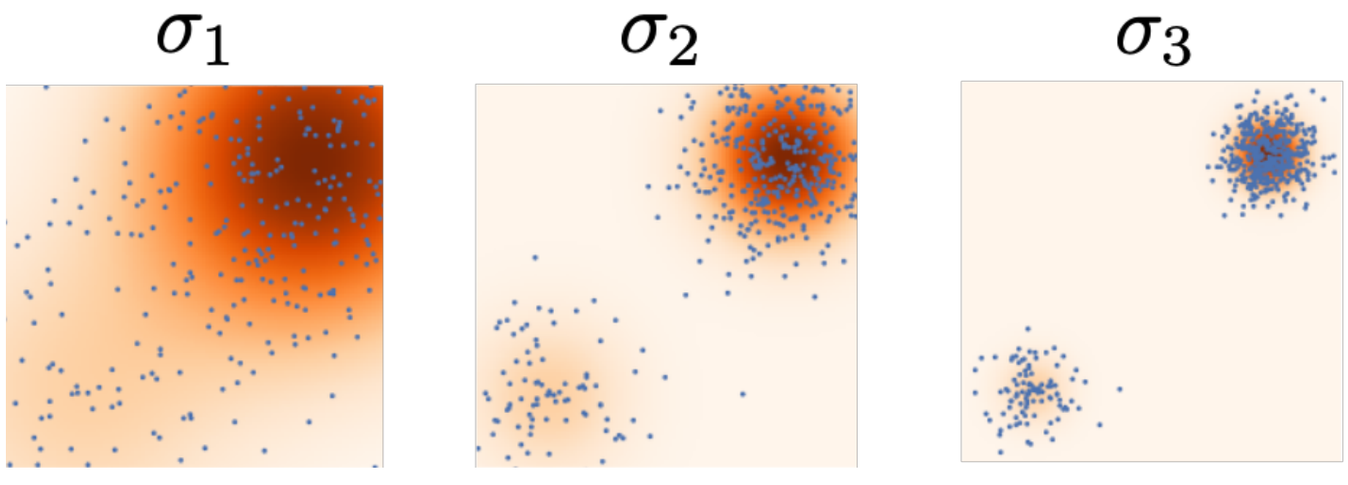
[Song and Ermon, 2019]

Pitfall 2

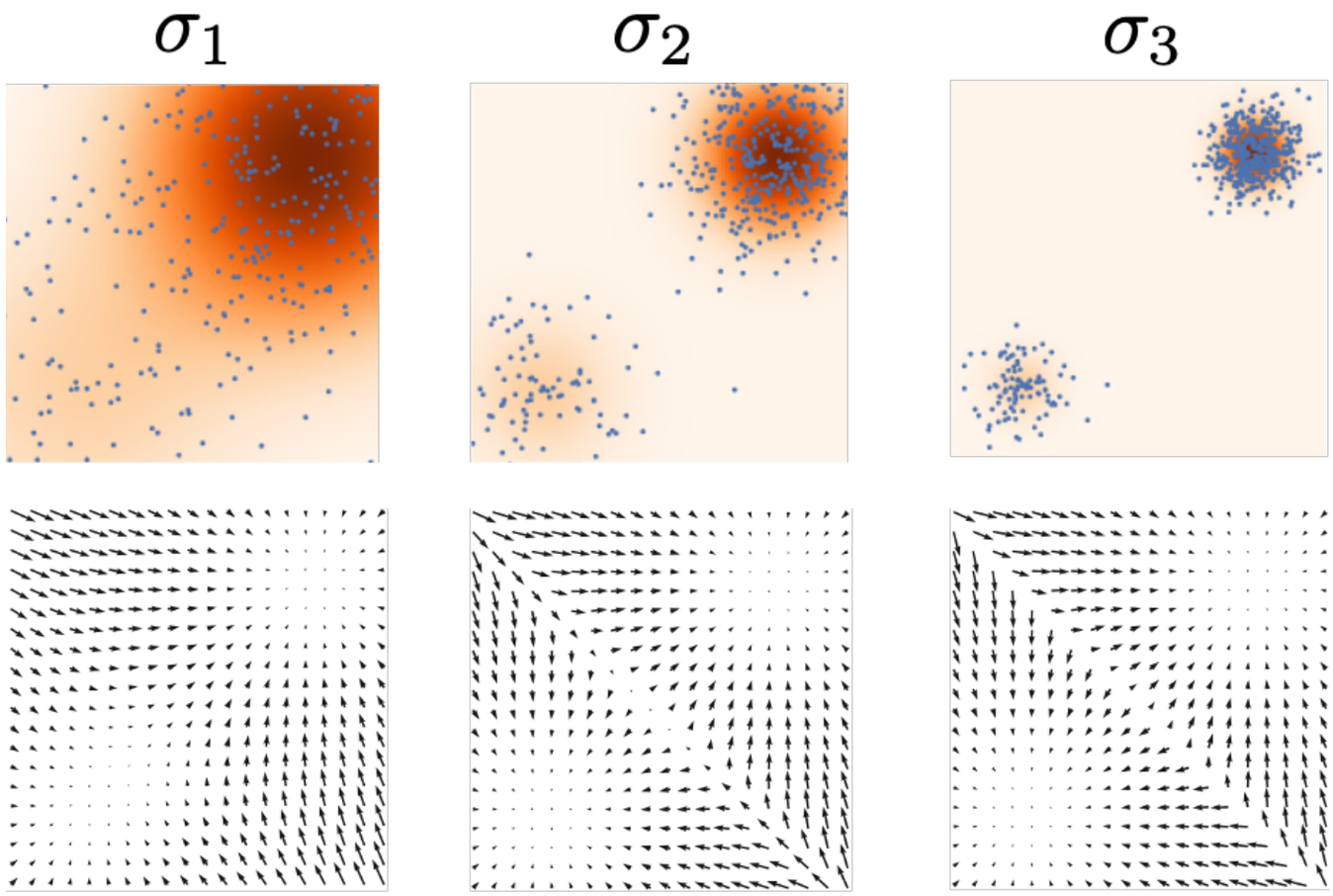


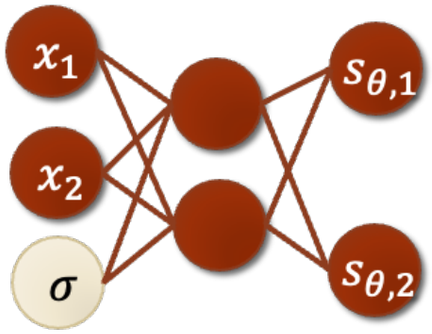
[Song and Ermon, 2019]

Joint Score Estimation via Noise Conditional Score Networks



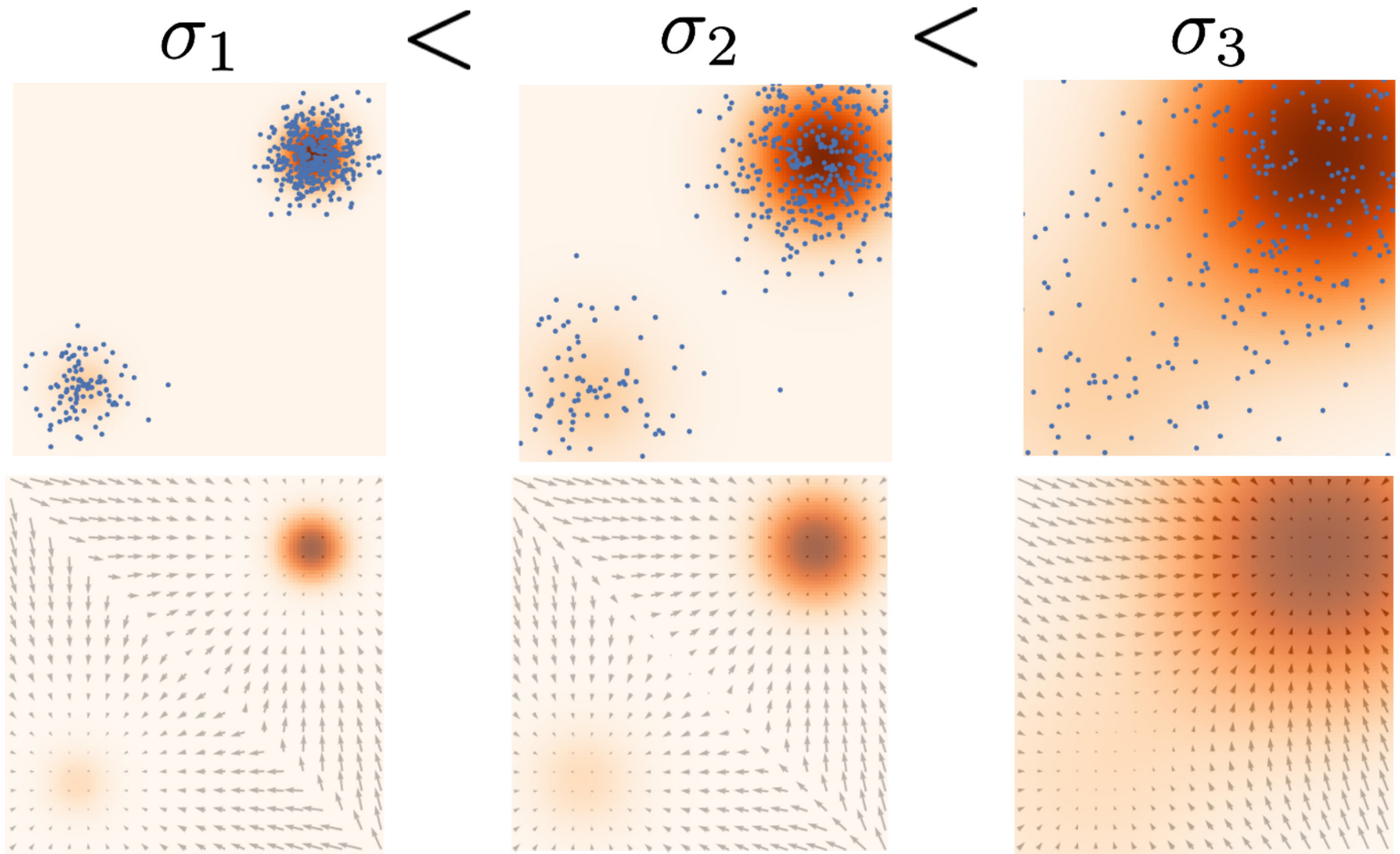
Joint Score Estimation via Noise Conditional Score Networks





Noise Conditional Score Network (NCSN)

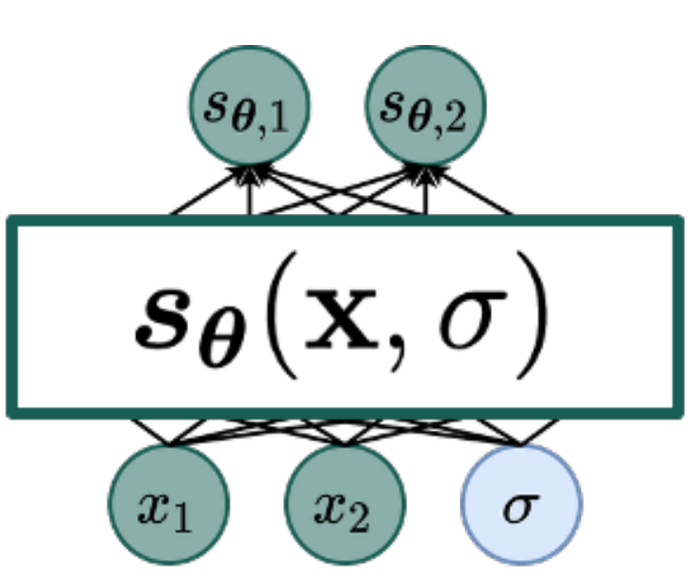
Using multiple noise levels



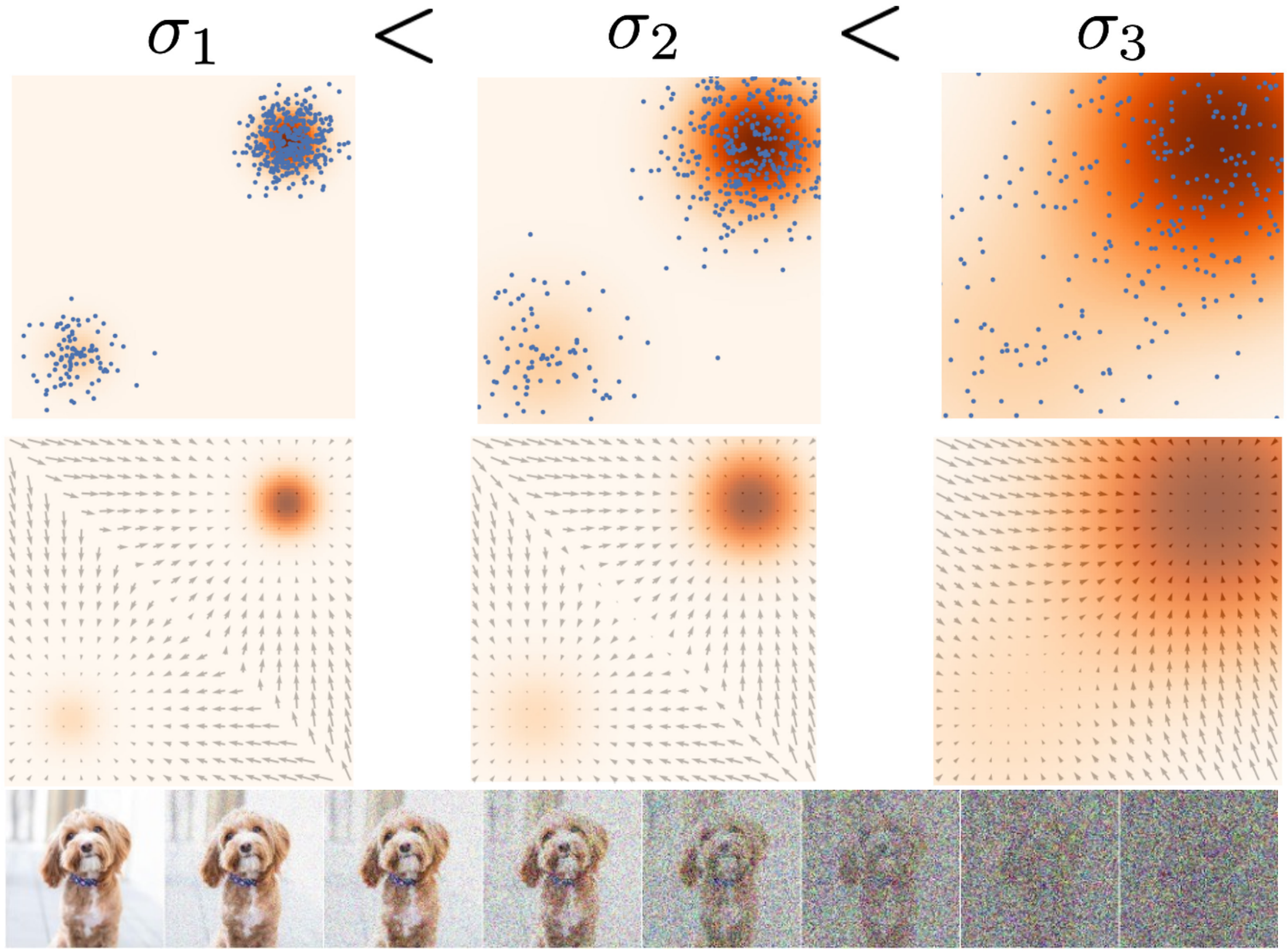
Score matching loss

Using multiple noise levels

Data



Noise Conditional Score Model

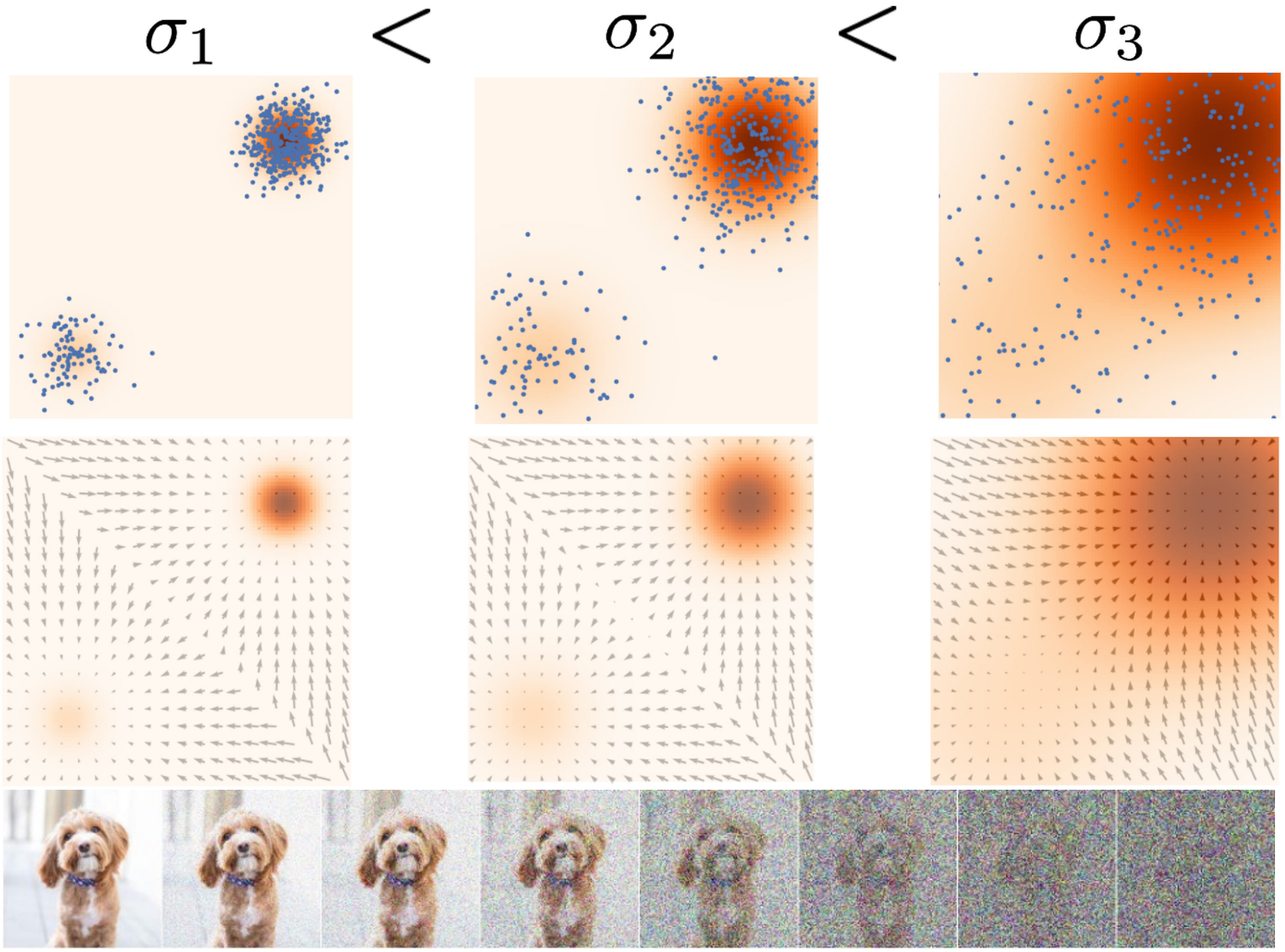


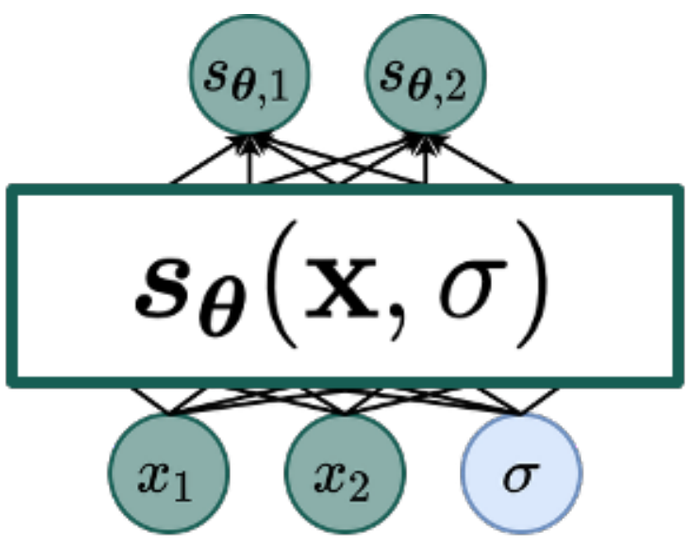
$$\frac{1}{N}\sum_{i=1}^{N}\lambda(\sigma_i)\mathbb{E}_{p_{\sigma_i}(\mathbf{x})}[\|\underbrace{\nabla_{\mathbf{x}}\log p_{\sigma_i}(\mathbf{x}) - s_\theta(\mathbf{x},\sigma_i)}\|_2^2]$$

Score matching loss

Using multiple noise levels

Data





Noise Conditional Score Model

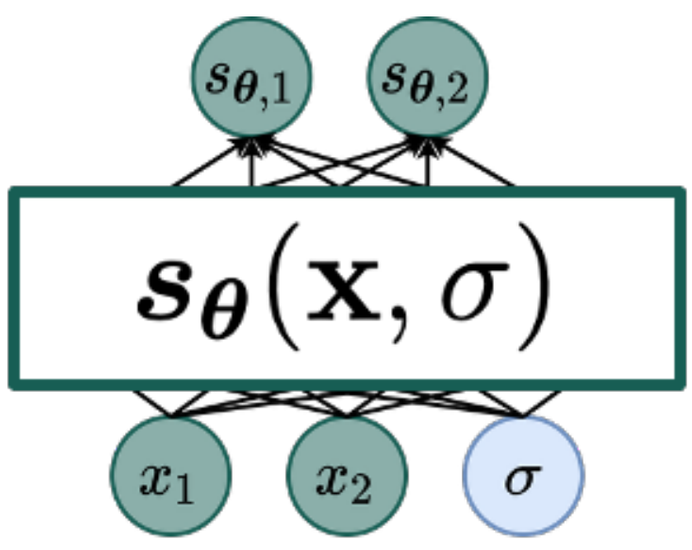
$$\frac{1}{N}\sum_{i=1}^{N}\lambda(\sigma_i)\mathbb{E}_{p_{\sigma_i}(\mathbf{x})}[\|\underbrace{\nabla_{\mathbf{x}}\log p_{\sigma_i}(\mathbf{x}) - s_\theta(\mathbf{x},\sigma_i)}\|_2^2]$$

Score matching loss

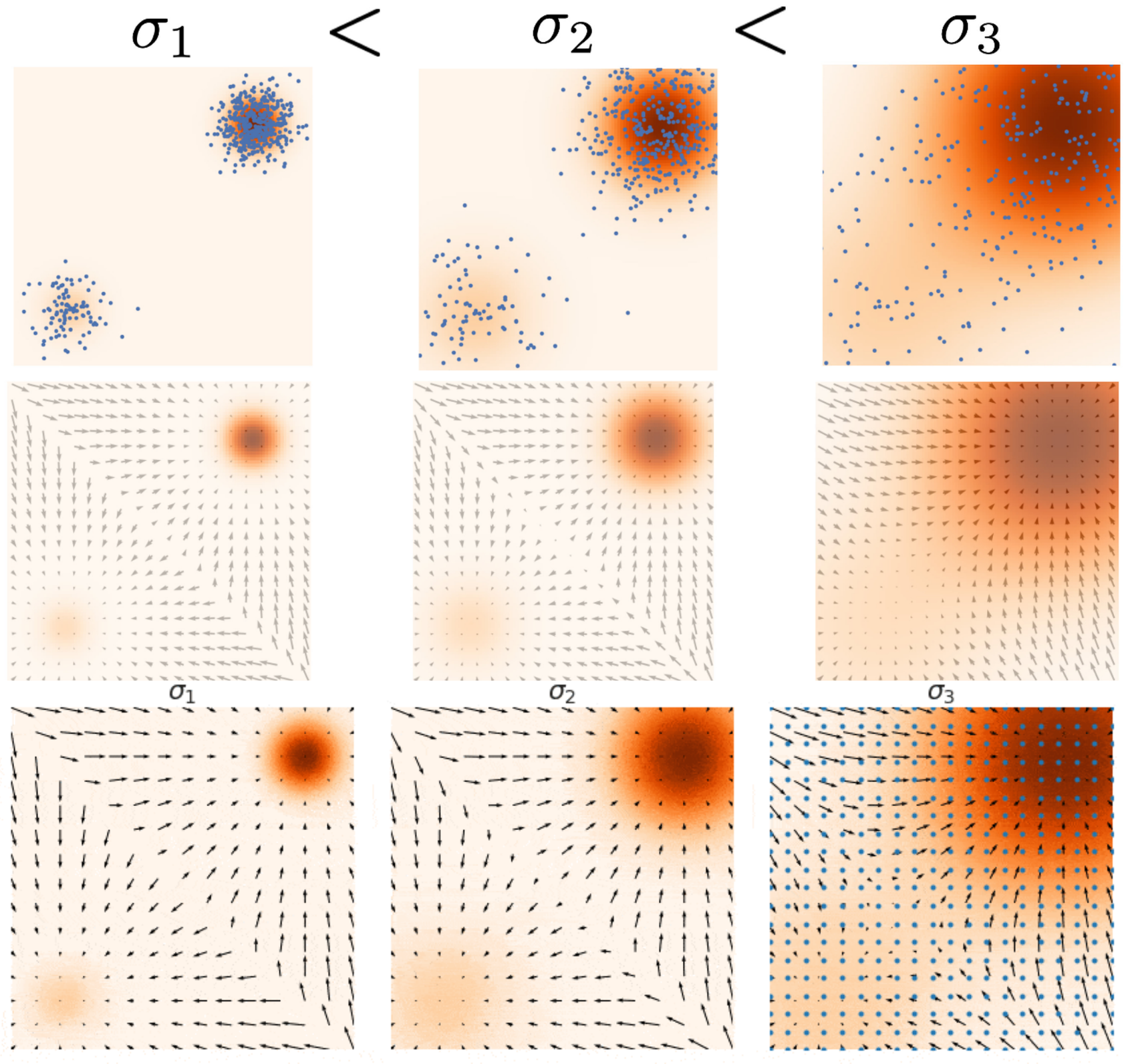
Diffusion, Sohl-Dickstein et al. 2015
DDPM, Ho et al. 2021

Using multiple noise levels

Data

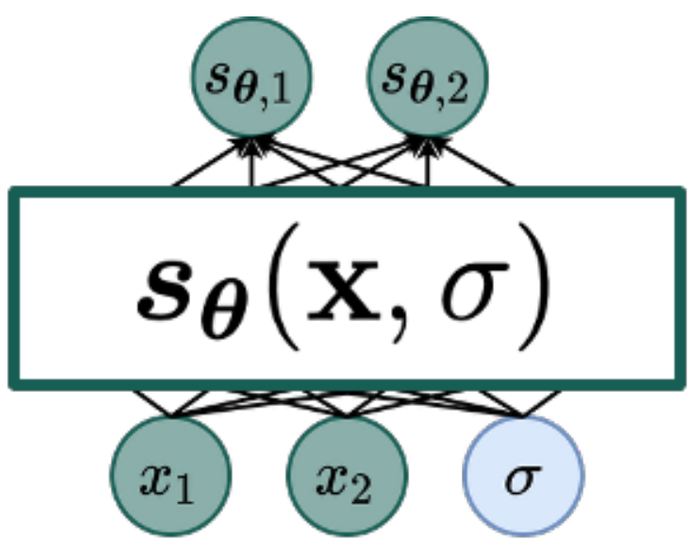


Noise Conditional Score Model

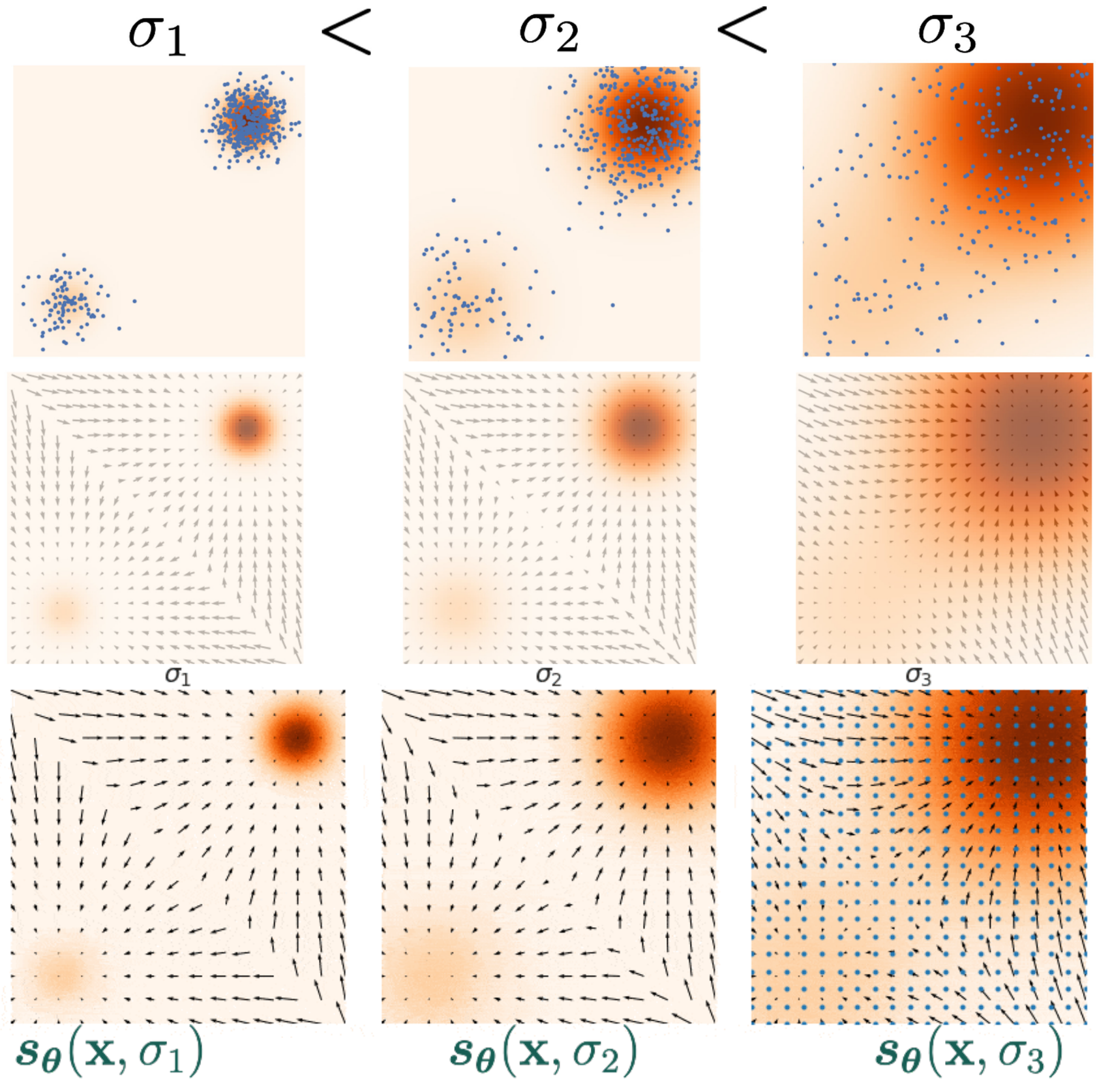


Using multiple noise levels

Data

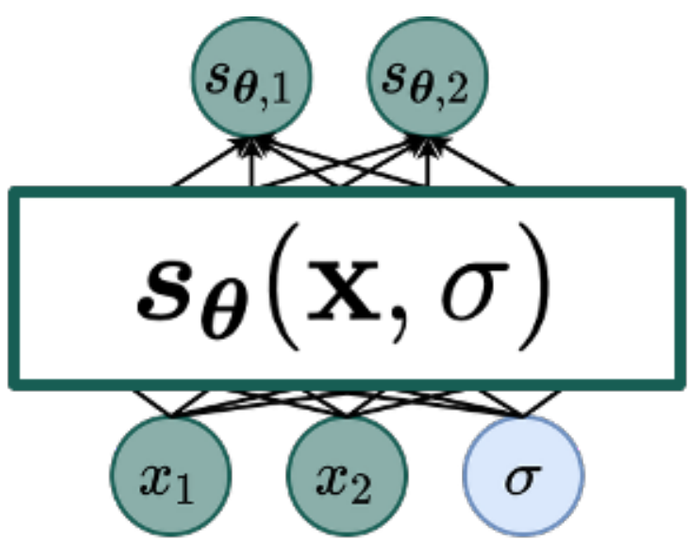


Noise Conditional Score Model

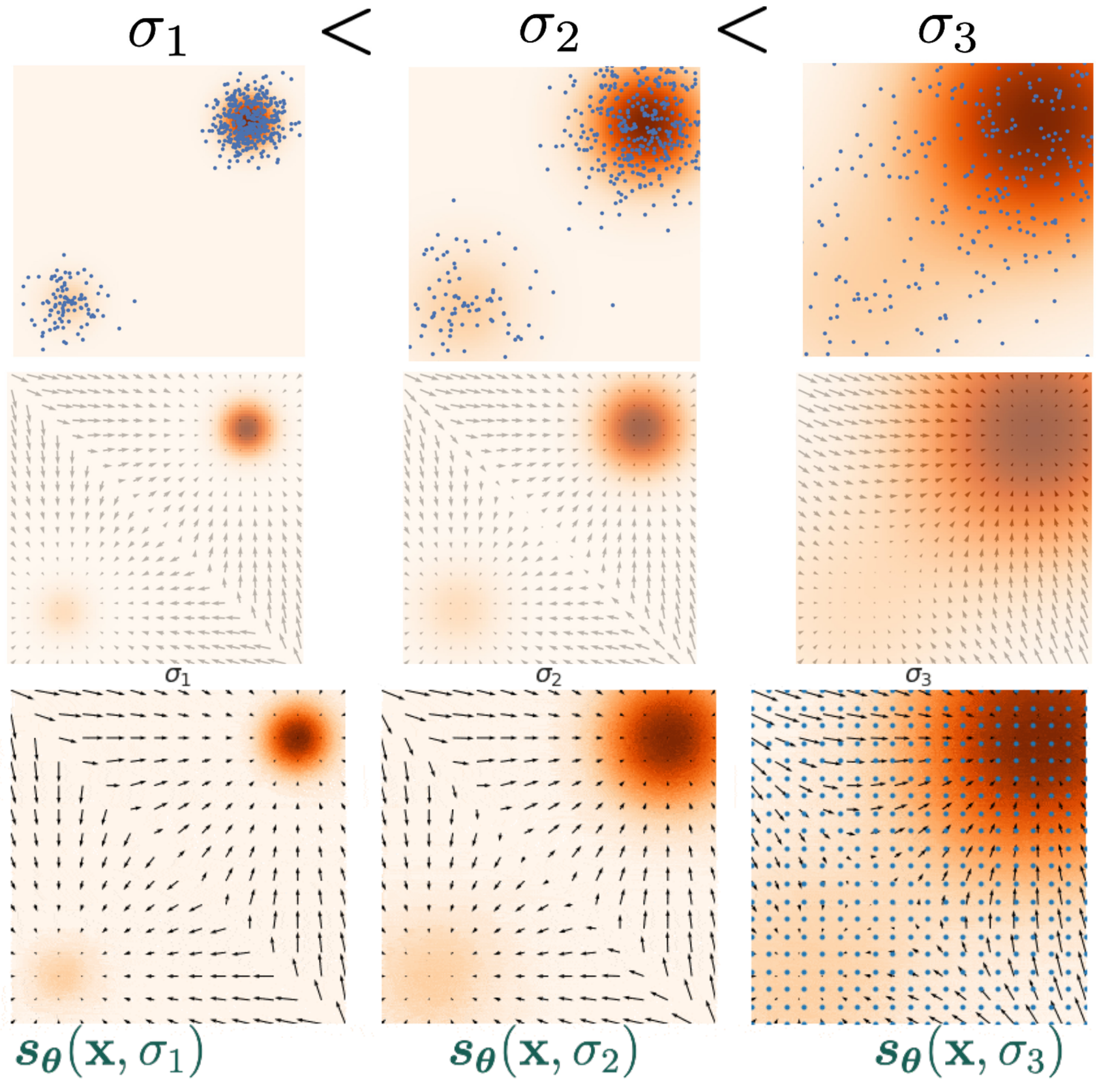


Using multiple noise levels

Data

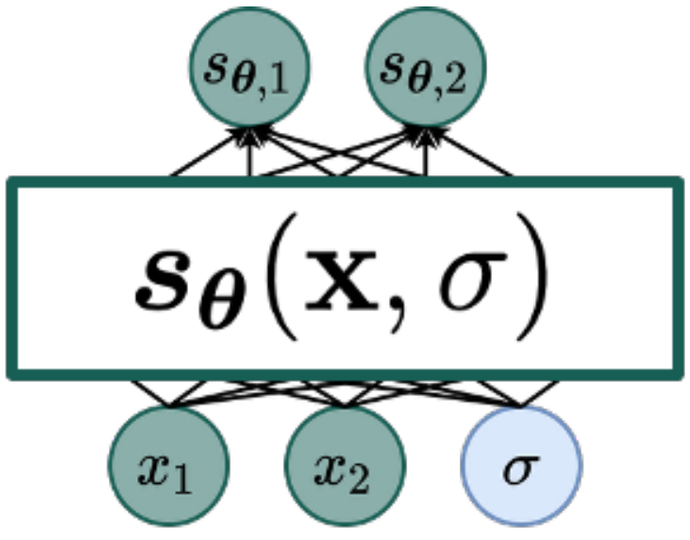


Noise Conditional Score Model

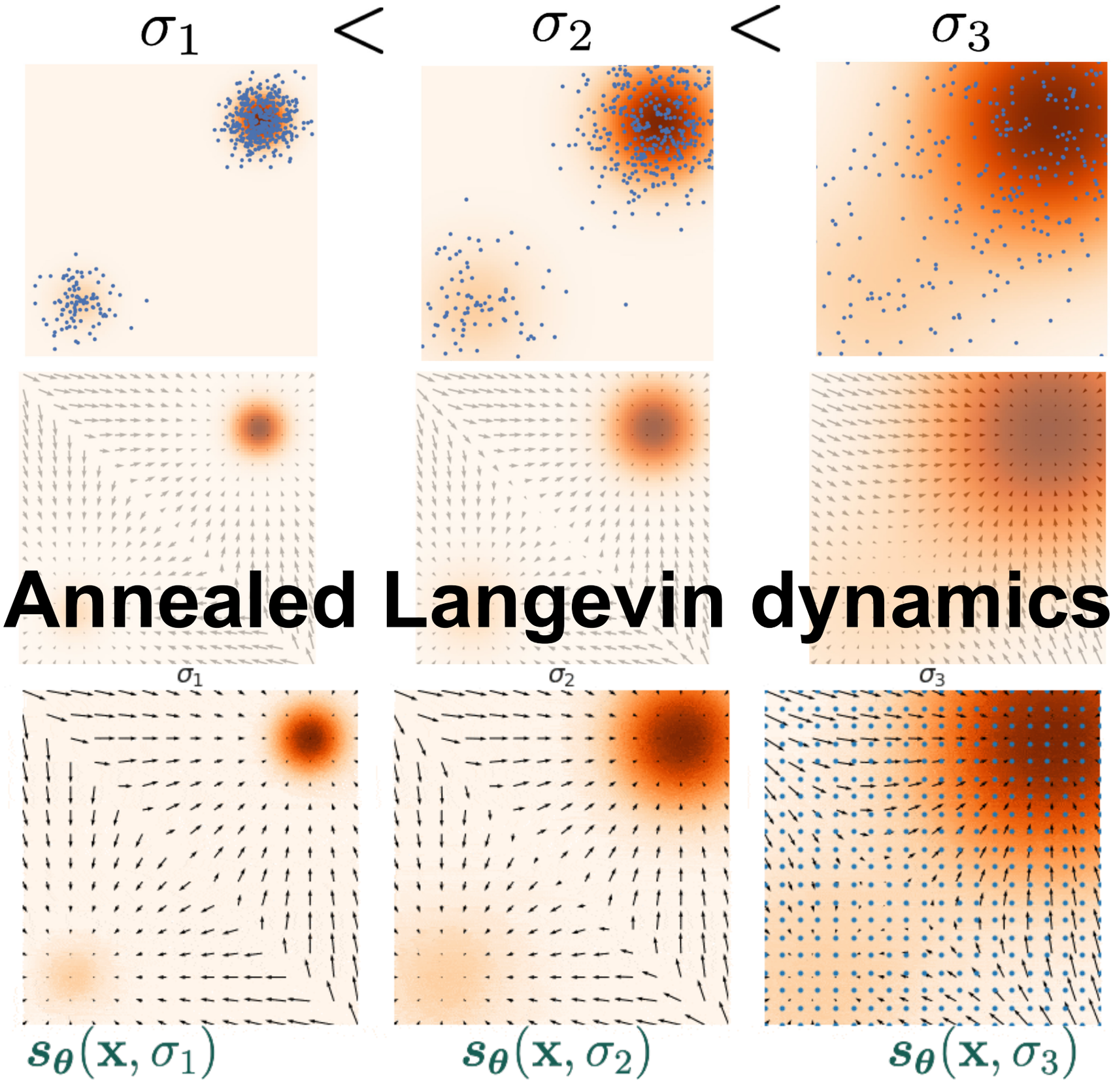


Using multiple noise levels

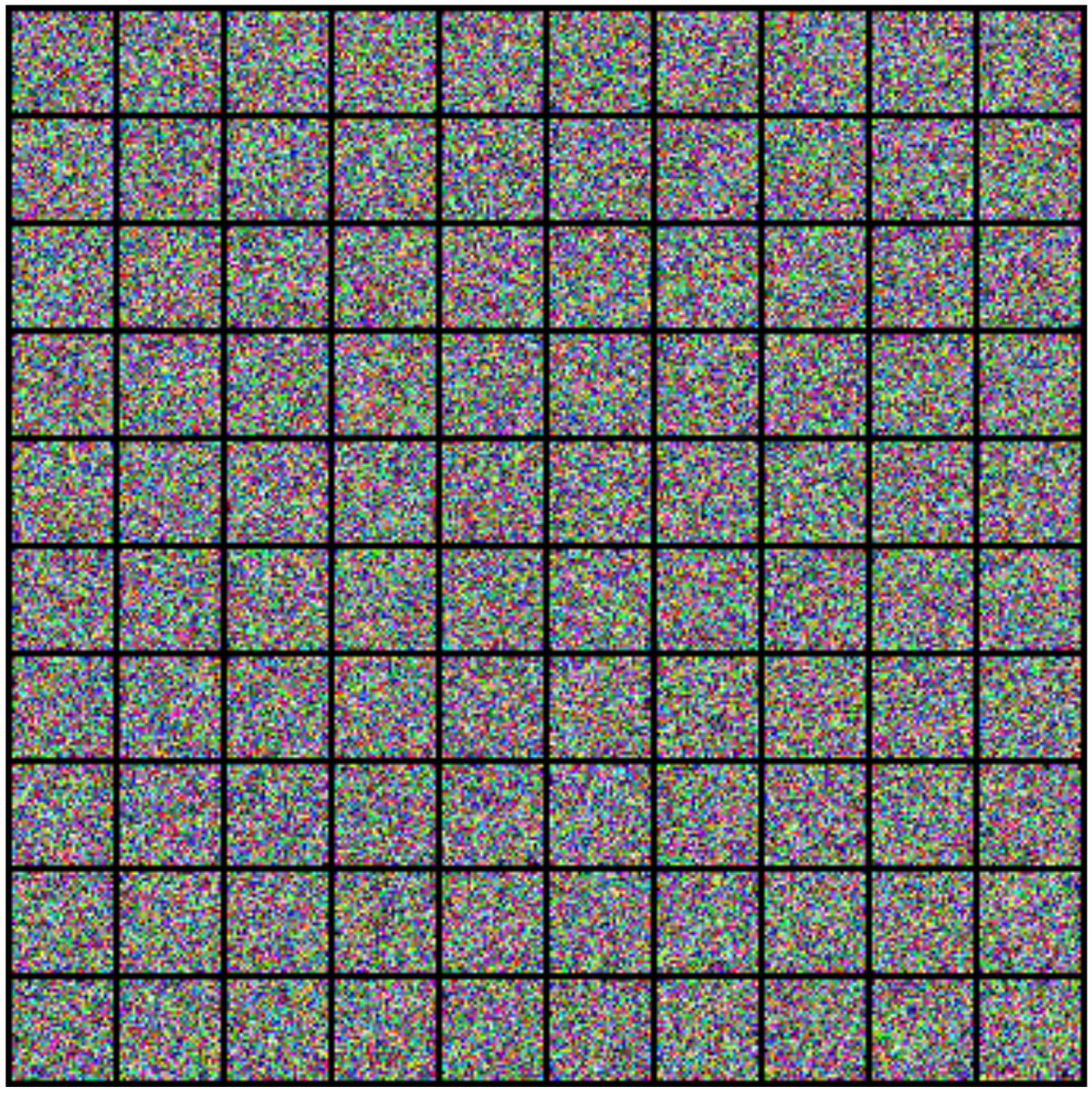
Data

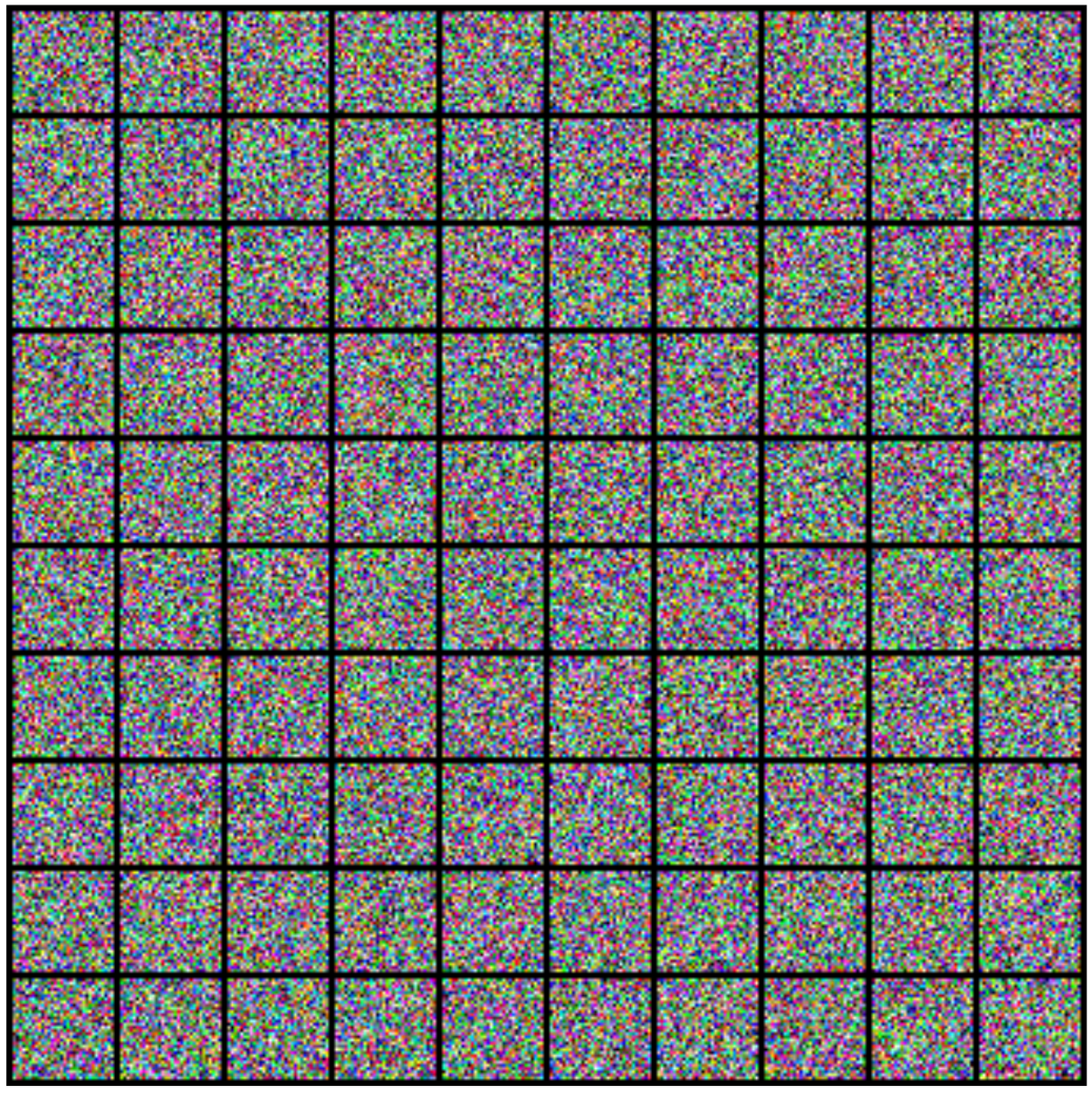


Noise Conditional Score Model

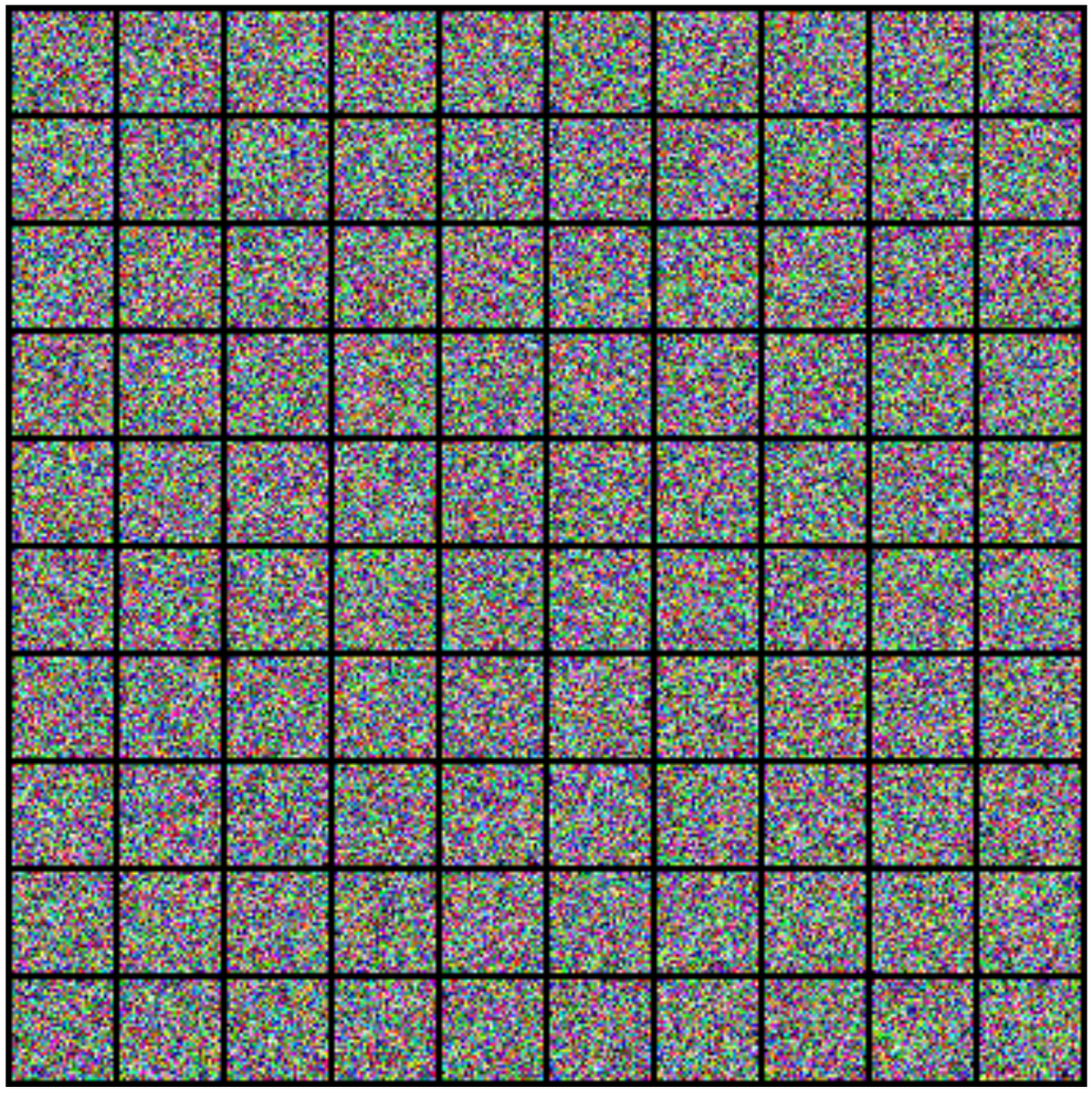


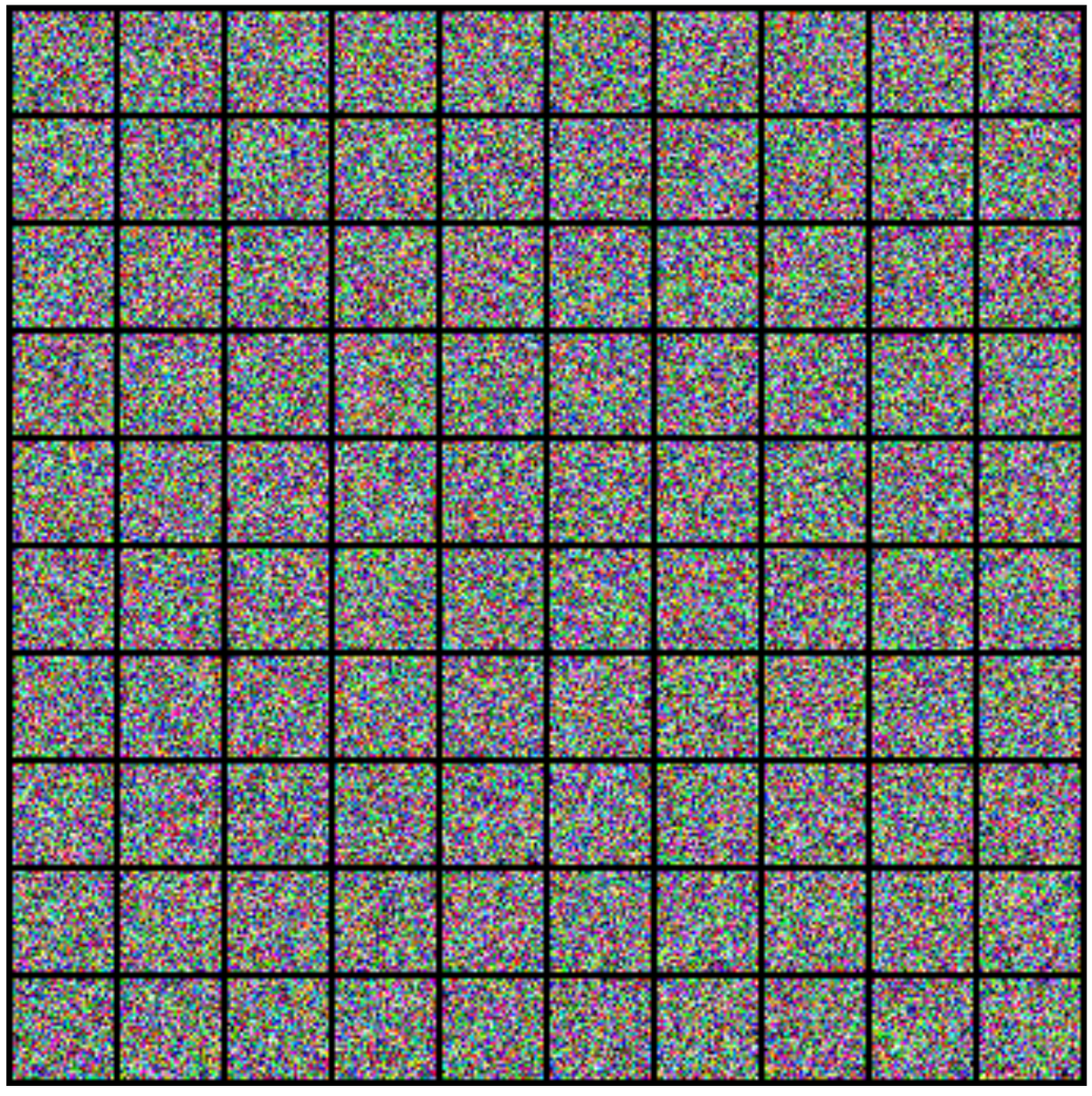
Experiments: sampling



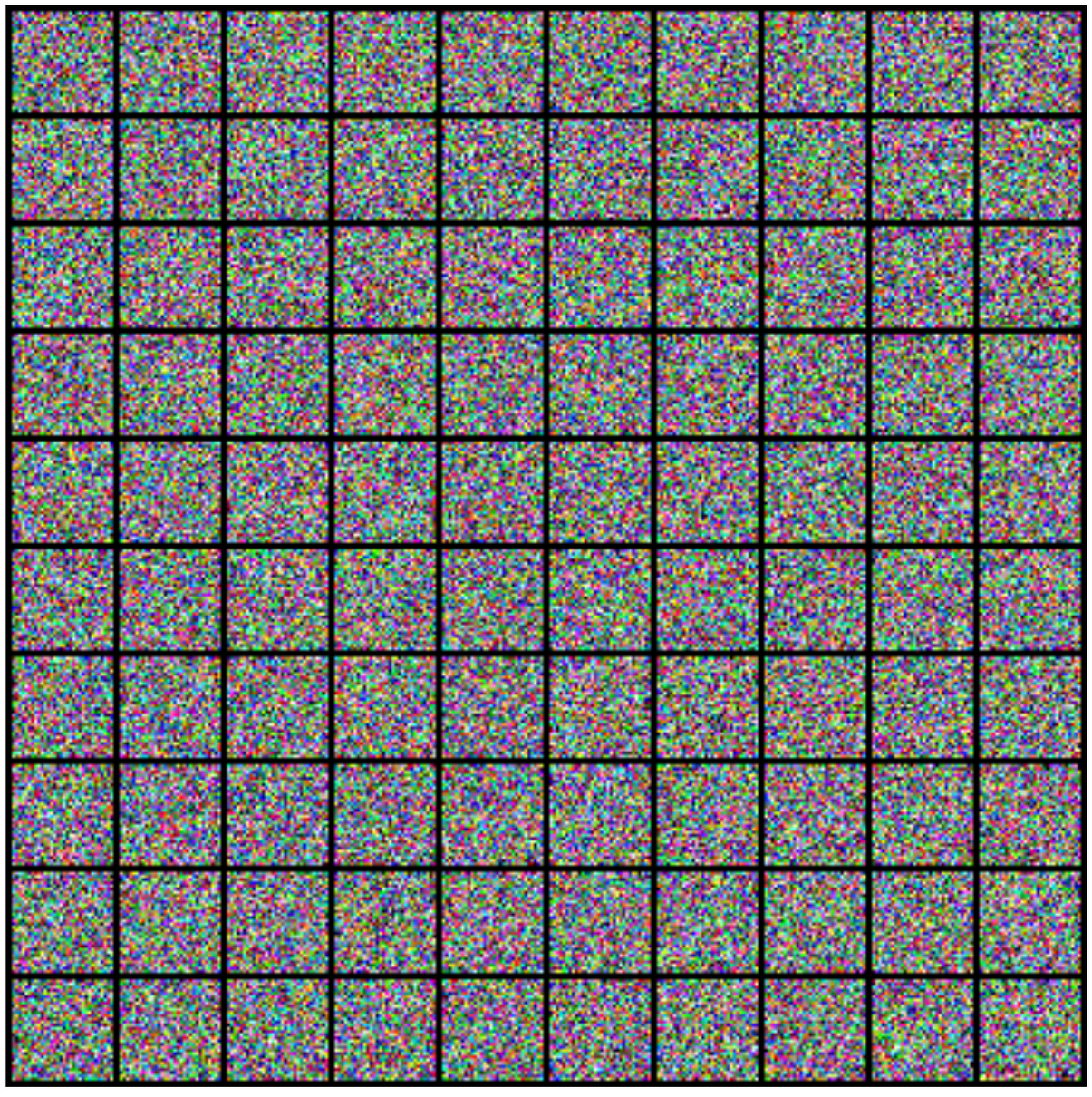


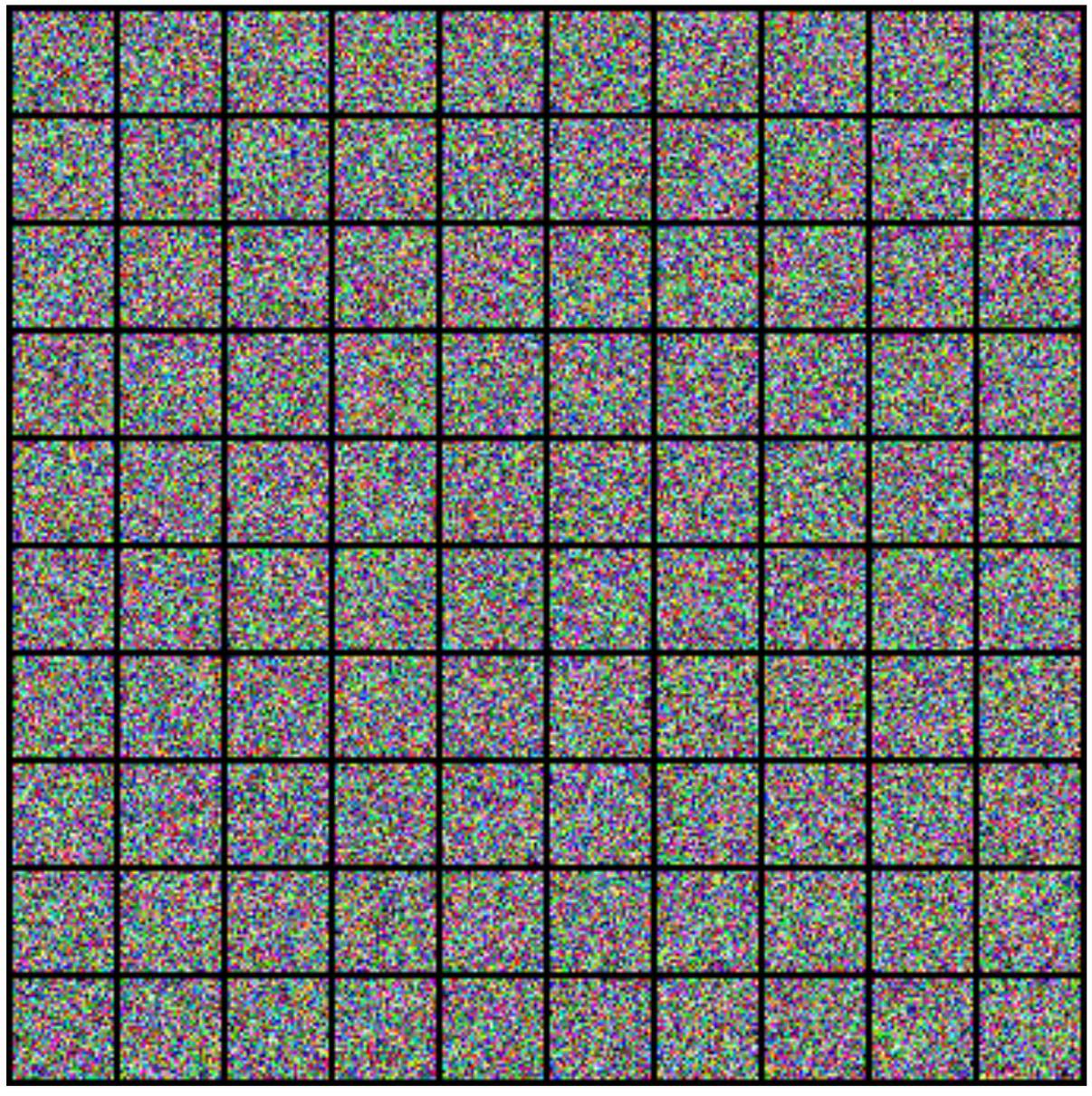
Experiments: sampling





Experiments: sampling





Why denoising is so powerful?

- Because it estimates the gradients of the log likelihood of data and neural nets are great at following gradients.
- Because it can populate low data density regions.

Quiz!

Quiz!

- Does score matching require knowledge of the normalizing constant of the data distribution?

Quiz!

- What is the primary role of the score function in score-based generative models?

Quiz!

- Why is the output of NCSN conditioned on the noise level parameter σ?

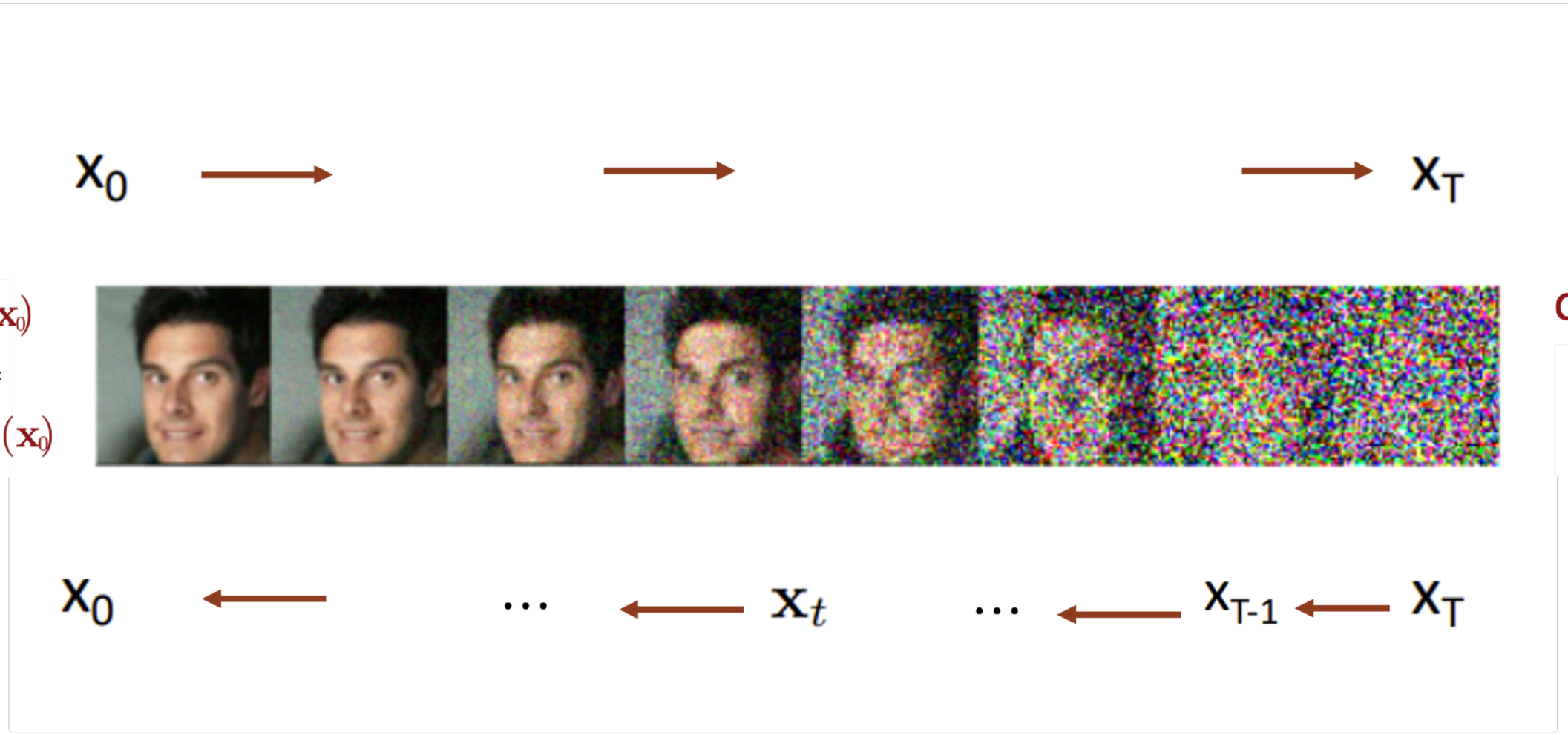
Quiz!

- Why we need Langevin dynamics for score-based generative models?

Outline

- Iterative noising process
- Diffusion perspective
- Diffusion model as hierarchical VAE
- Denoising diffusion probabilistic model
- Training and sampling

Sampling as iterative denoising



Iteratively remove noise

Sampling as iterative denoising

x_0 → → → x_T

$q_0(\mathbf{x}_0) = p_{\text{data}}(\mathbf{x}_0)$

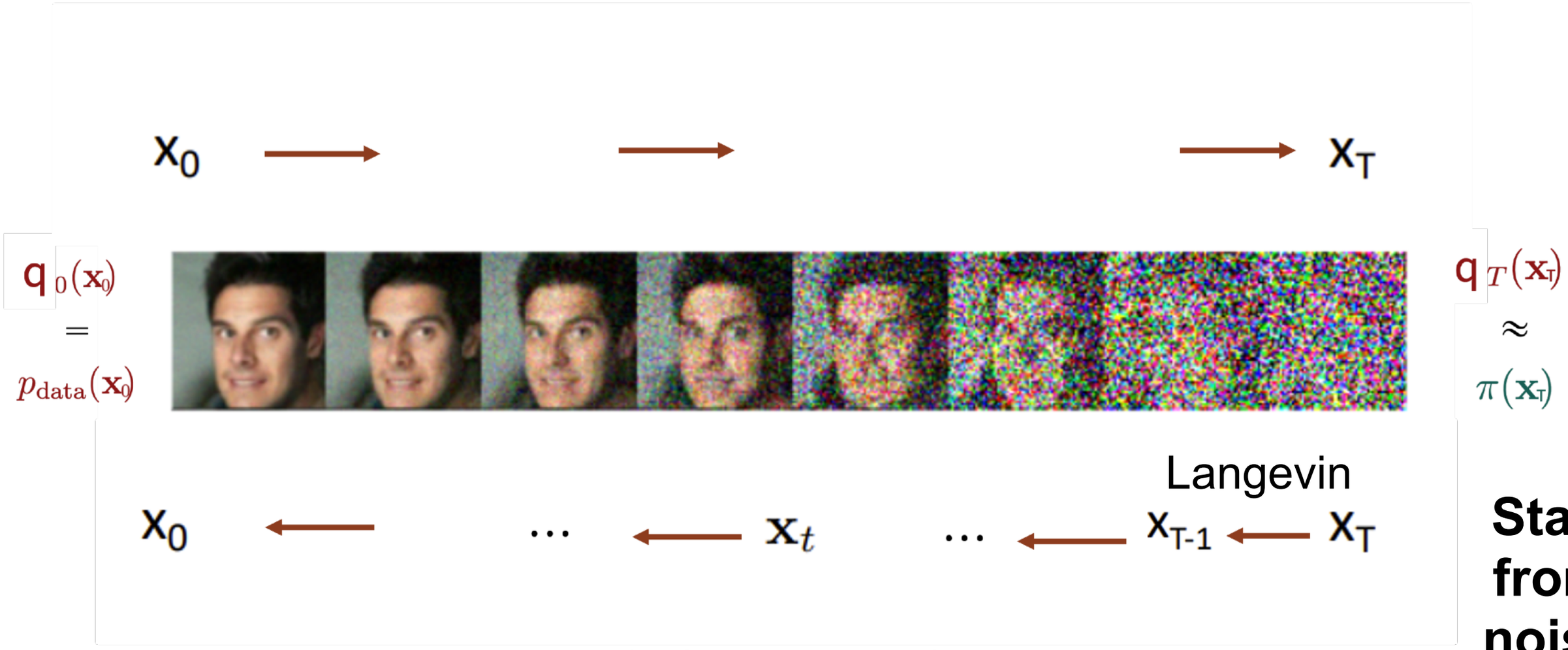
$q_T(\mathbf{x}_T) \approx \pi(\mathbf{x}_T)$

x_0 ← … ← $\mathbf{x}_t$ … ← x_{T-1} ← x_T

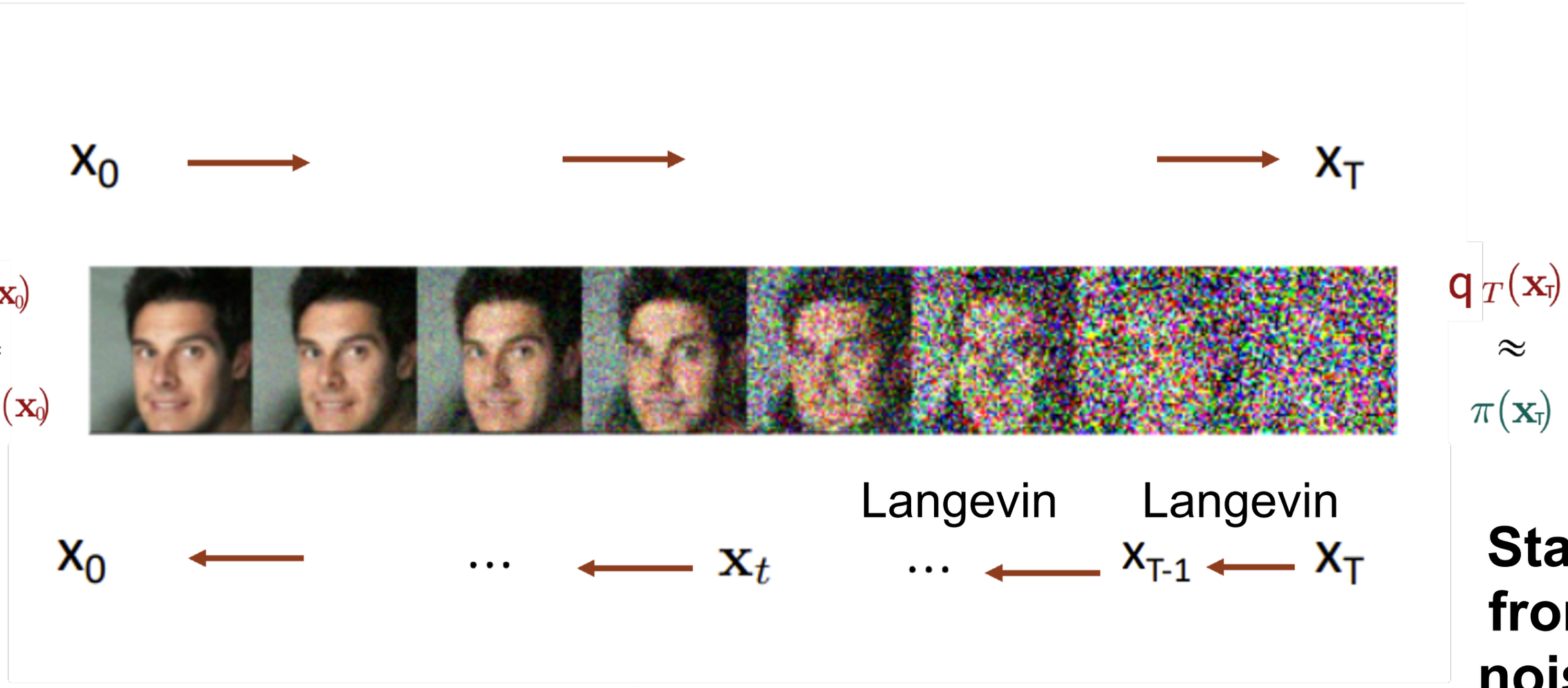
Start from noise

Iteratively remove noise

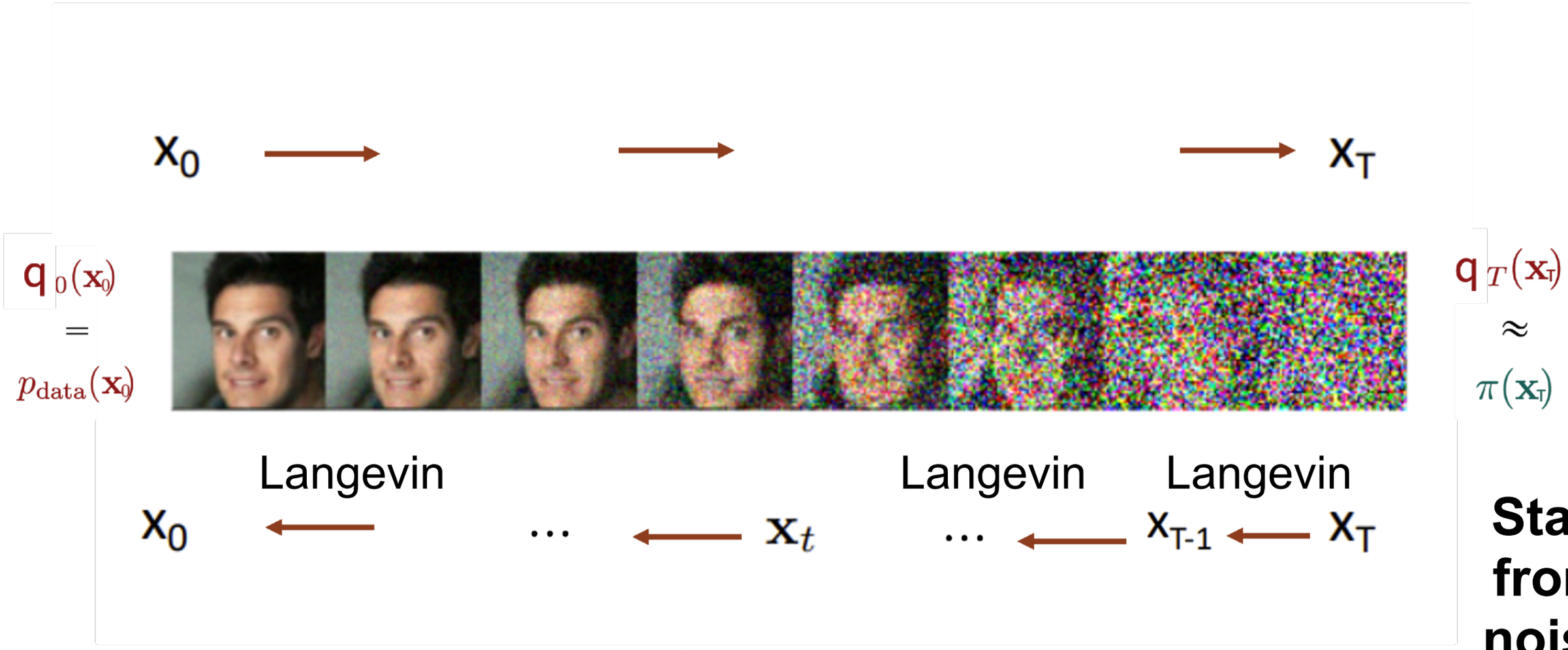
Sampling as iterative denoising



Sampling as iterative denoising



Sampling as iterative denoising



Sampling as iterative denoising

Inverse process: iteratively add Gaussian noise

x_0 → → → x_T

$q_0(\mathbf{x}_0) = p_{\text{data}}(\mathbf{x}_0)$

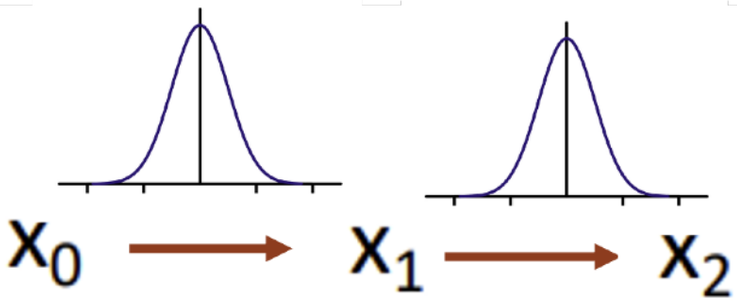
$q_T(\mathbf{x}_T) \approx \pi(\mathbf{x}_T)$

Langevin: x_0 ← … ← $\mathbf{x}_t$ … Langevin: x_{T-1} ← Langevin: x_T

Start from noise

Iteratively remove noise

Iterative noising process



Progressively add noise

x_T

$q_0(\mathbf{x}_0)$

$=$

$p_{\text{data}}(\mathbf{x}_0)$

$q_T(\mathbf{x}_T)$

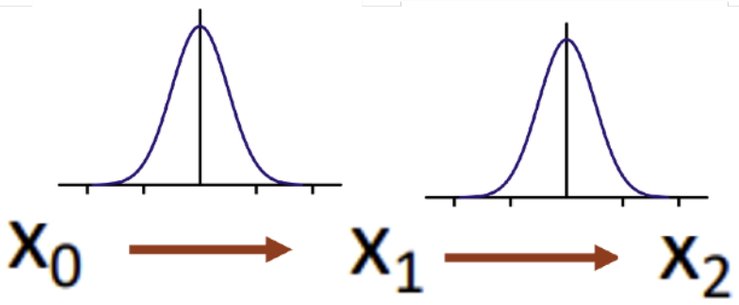
$\approx$

$\pi(\mathbf{x}_T)$

Noise perturbed densities are obtained by adding Gaussian noise

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}\left(\mathbf{x}_t; \sqrt{1-\beta_t}\,\mathbf{x}_{t-1}, \beta_t \mathbf{I}\right)$$

Iterative noising process



Progressively add noise

$q_0(\mathbf{x}_0)$

$=$

$p_{\text{data}}(\mathbf{x}_0)$

$q_T(\mathbf{x}_T)$

$\approx$

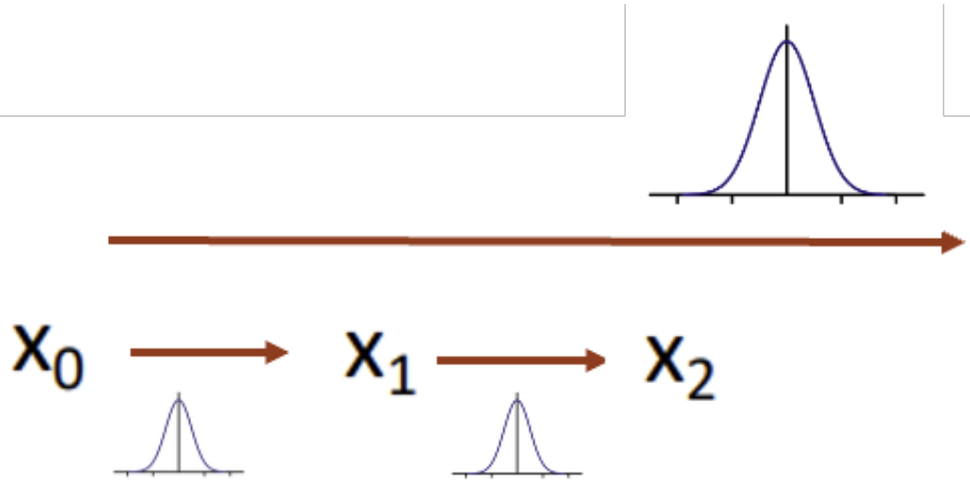
$\pi(\mathbf{x}_T)$

Noise perturbed densities are obtained by adding Gaussian noise

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}\left(\mathbf{x}_t; \sqrt{1-\beta_t}\,\mathbf{x}_{t-1}, \beta_t \mathbf{I}\right)$$

Defines a joint distribution $q(\mathbf{x}_{1:T} \mid \mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t \mid \mathbf{x}_{t-1})$

Multistep transition is easy



$q_0(\mathbf{x}_0)$

$=$

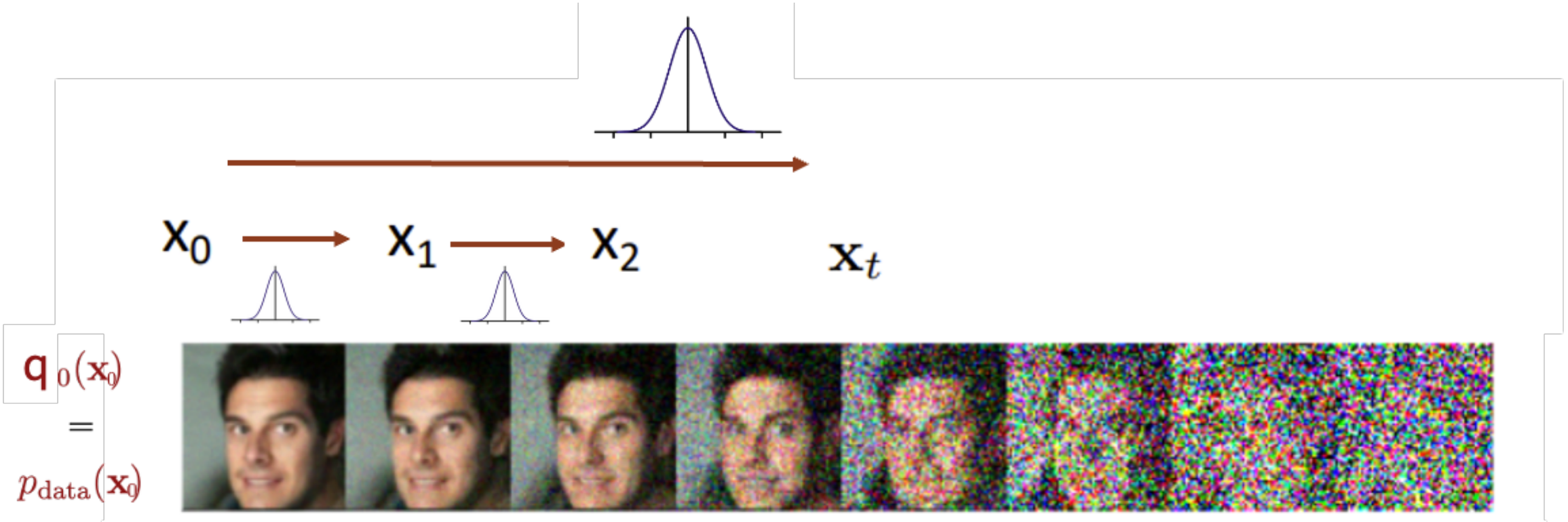
$p_{\text{data}}(\mathbf{x}_0)$

Multi-step transitions are also Gaussian and can be computed in closed form

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}))$$

$$\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$$

Multistep transition is easy



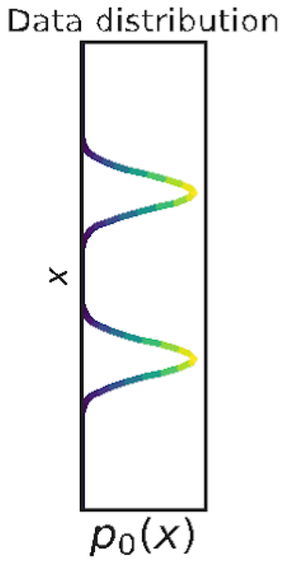
Multi-step transitions are also Gaussian and can be computed in closed form

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}))$$

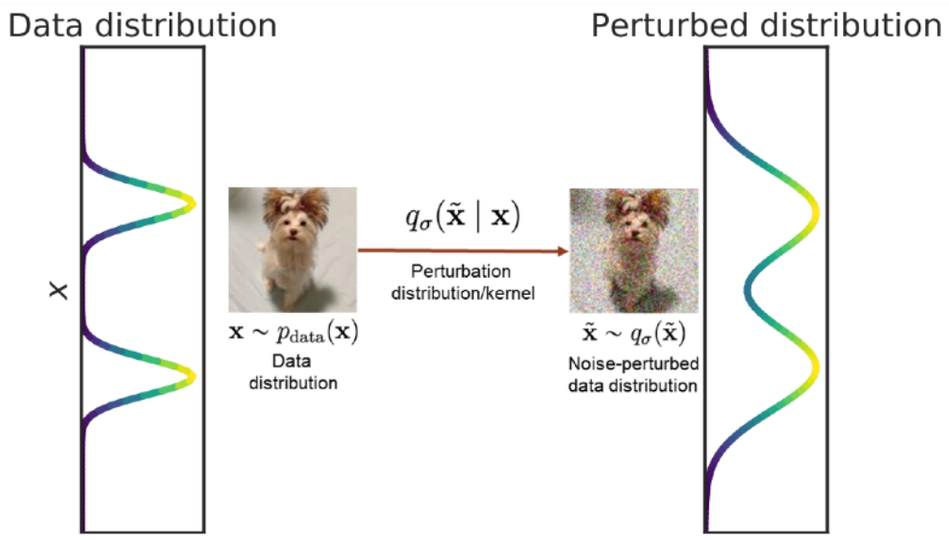
$$\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$$

1. Same as noise-perturbed data distributions in score-based models
2. Efficient sampling at any t

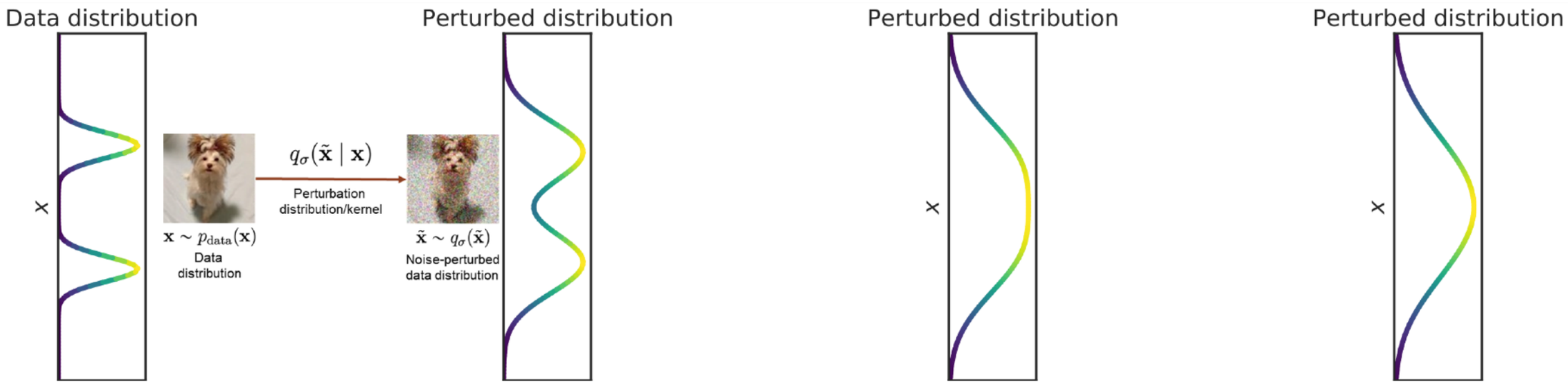
Diffusion perspective



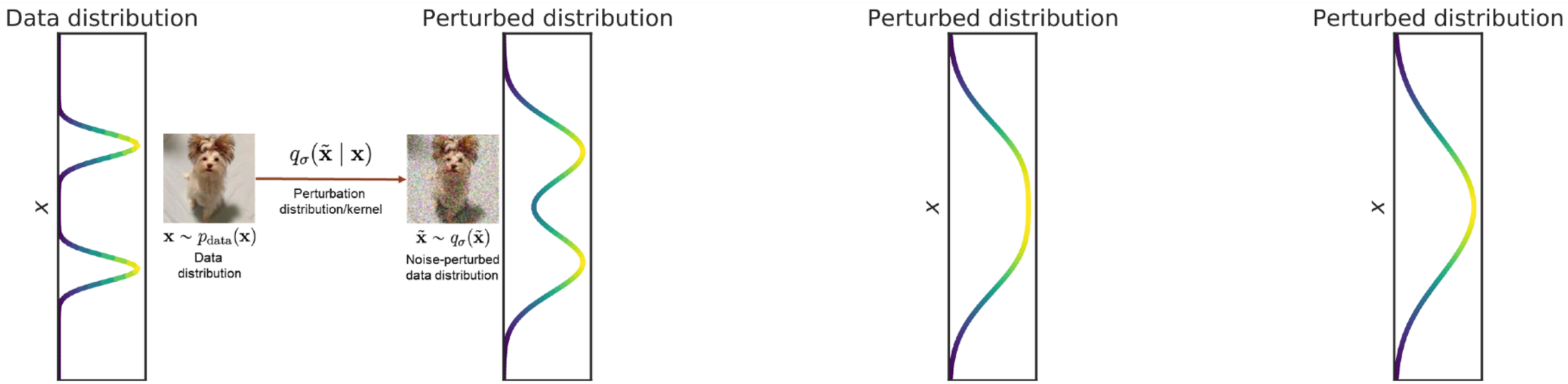
Diffusion perspective

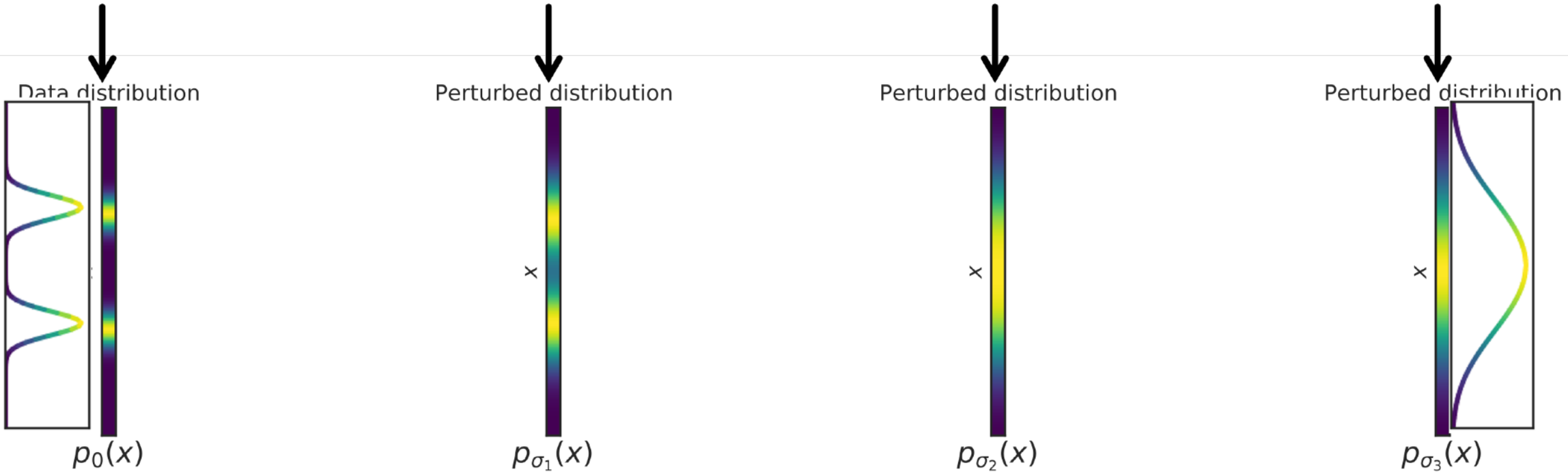


Diffusion perspective

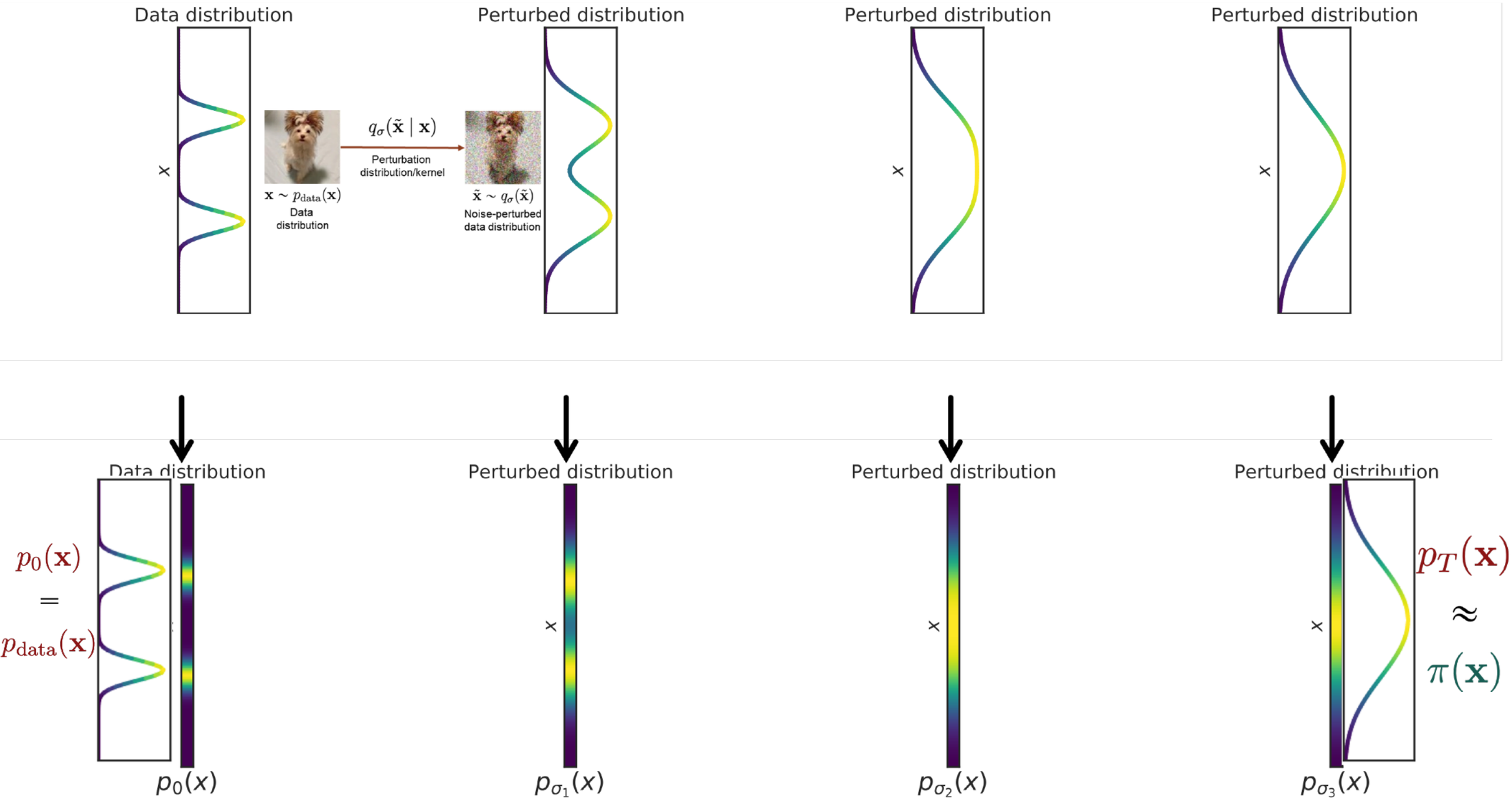


Diffusion perspective

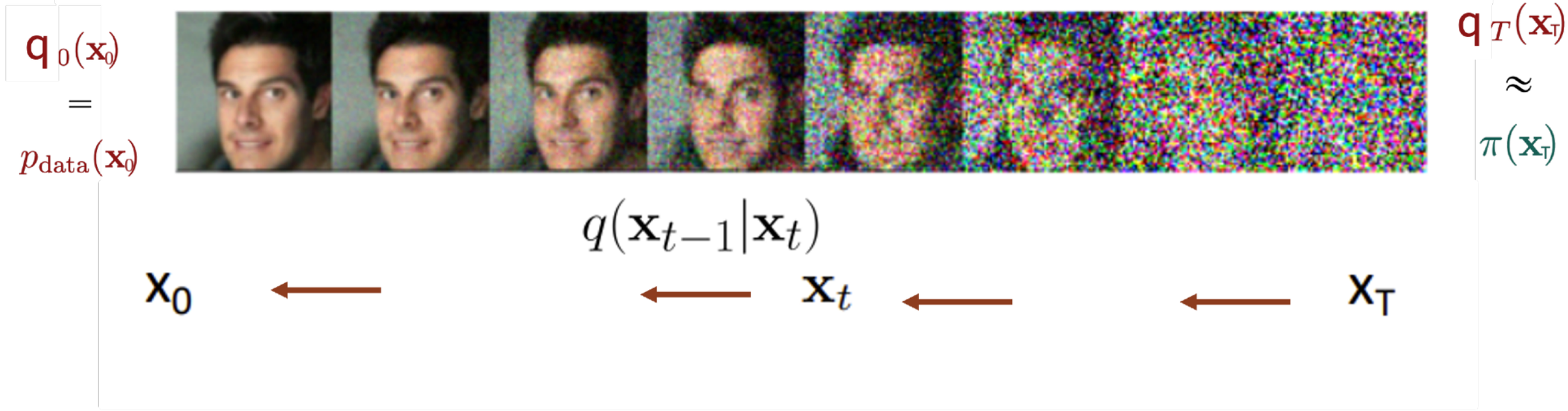




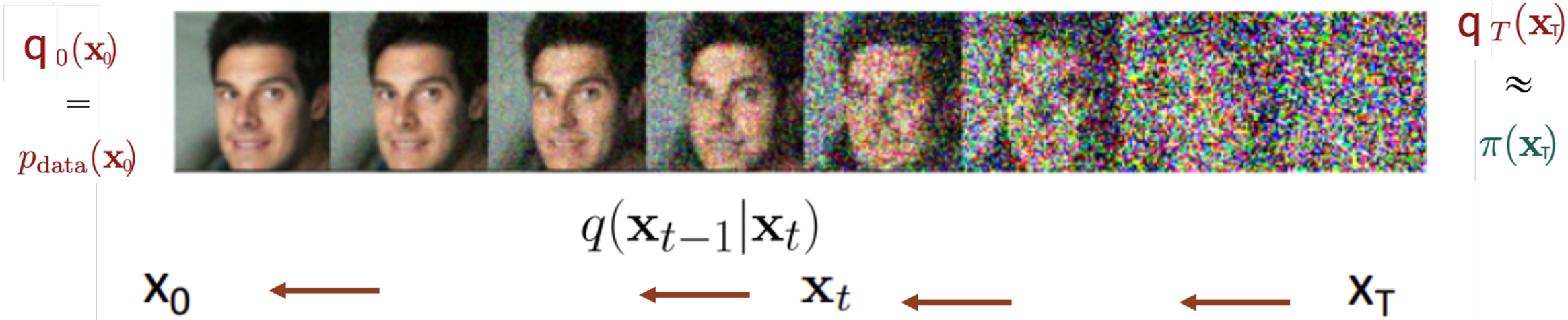
Diffusion perspective



Iterative denoising



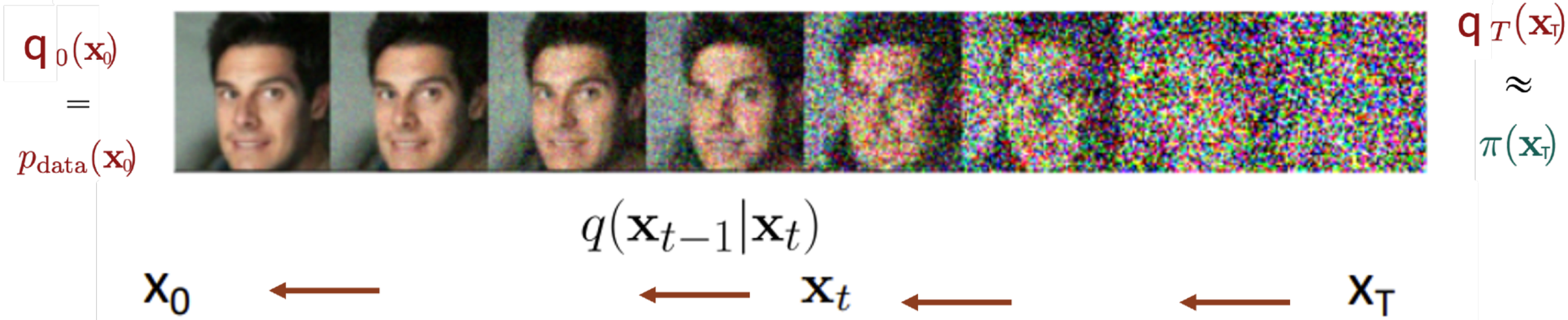
Iterative denoising



Ideal sampling process:

1. Sample $\mathbf{x}_T$ from $\pi(\mathbf{x}_T)$, i.e. from pure noise
2. Iteratively sample from the true denoising distribution $q(x_{t-1} \mid x_t)$

Iterative denoising



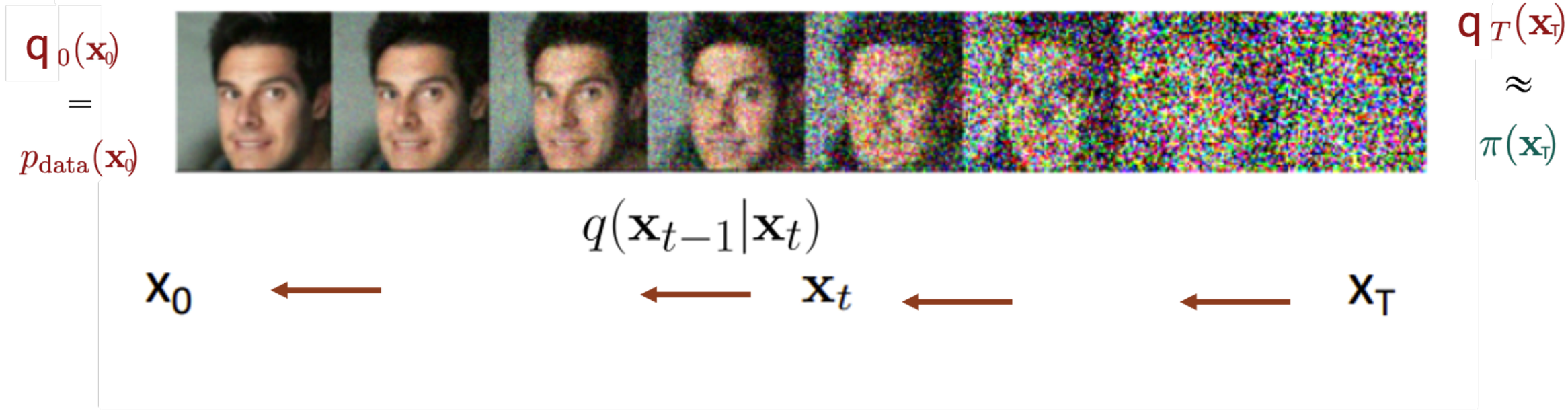
Ideal sampling process:

1. Sample $\mathbf{x}_T$ from $\pi(\mathbf{x}_T)$, i.e. from pure noise
2. Iteratively sample from the true denoising distribution $q(x_{t-1} \mid x_t)$

Issue: Exact denoising distribution is unknown!

Solution: Learn a variational approximation

Iterative denoising



Iterative denoising

$q_0(\mathbf{x}_0) = p_{\text{data}}(\mathbf{x}_0)$

$q_T(\mathbf{x}_T) \approx \pi(\mathbf{x}_T)$

$q(\mathbf{x}_{t-1}|\mathbf{x}_t)$

$\mathbf{x}_0 \leftarrow \cdots \leftarrow \mathbf{x}_t \leftarrow \cdots \leftarrow \mathbf{x}_T$

Sample $\mathbf{x}_T$ from $p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I}) = \pi$

Iterative denoising

$q_0(\mathbf{x}_0) = p_{\text{data}}(\mathbf{x}_0)$

$q_T(\mathbf{x}_T) \approx \pi(\mathbf{x}_T)$

$\mathbf{x}_0 \leftarrow \cdots \leftarrow \mathbf{x}_t \leftarrow \cdots \leftarrow \mathbf{x}_T$

$q(\mathbf{x}_{t-1}|\mathbf{x}_t)$

Sample $\mathbf{x}_T$ from $p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I}) = \pi$

Iteratively sample from $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_t^2 \mathbf{I})$

Iterative denoising

$q_0(\mathbf{x}_0) = p_{\text{data}}(\mathbf{x}_0)$

$q_T(\mathbf{x}_T) \approx \pi(\mathbf{x}_T)$

$q(\mathbf{x}_{t-1}|\mathbf{x}_t)$

$\mathbf{x}_0 \leftarrow \cdots \leftarrow \mathbf{x}_t \leftarrow \cdots \leftarrow \mathbf{x}_T$

$p_\theta(x_{t-1} \mid x_t)$

Sample $\mathbf{x}_T$ from $p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I}) = \pi$

Iteratively sample from $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_t^2 \mathbf{I})$

Iterative denoising

$q_0(\mathbf{x}_0) = p_{\text{data}}(\mathbf{x}_0)$

$q_T(\mathbf{x}_T) \approx \pi(\mathbf{x}_T)$

$q(\mathbf{x}_{t-1}|\mathbf{x}_t)$

$\mathbf{x}_0 \leftarrow \cdots \leftarrow \mathbf{x}_t \leftarrow \cdots \leftarrow \mathbf{x}_T$

$p_\theta(x_{t-1} \mid x_t)$

Sample $\mathbf{x}_T$ from $p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I}) = \pi$

Iteratively sample from $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_t^2 \mathbf{I})$

Joint distribution $p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$

Diffusion models as hierarchical VAE

Encoder (**fixed**):

$\mathbf{x}_0$ → → → $\mathbf{x}_T$

$q_T(\mathbf{x}_T) \approx \pi(\mathbf{x}_T)$

$q_0(\mathbf{x}_0) = p_{\text{data}}(\mathbf{x}_0)$

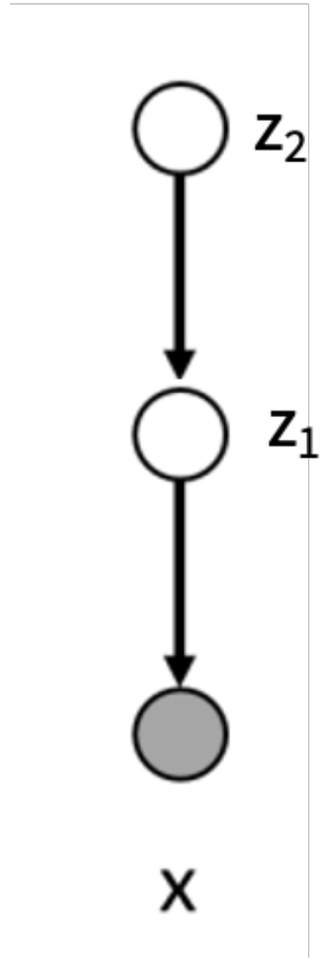
$p(\mathbf{x}_T)$

$\mathbf{x}_0$ ← ← ← $\mathbf{x}_T$

Prior over latents

Decoder (**learnable**): $p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$

From VAE to Hierarchical VAE



Training a denoising diffusion probabilistic model

Encoder (**fixed**): $q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$

$\mathbf{x}_0$ → $\mathbf{x}_T$

$q_0(\mathbf{x}_0) = p_{\text{data}}(\mathbf{x}_0)$

$p(\mathbf{x}_T)$

Decoder: $p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$

$\mathbf{x}_0$ ← $\mathbf{x}_T$

ELBO loss (negative ELBO averaged over data distribution):

$$\mathbb{E}_{q(\mathbf{x}_0)}[-\log p_\theta(\mathbf{x}_0)] \leq \mathbb{E}_{q(\mathbf{x}_0)q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] =: L$$

DDPM, Ho et al. 2021
Song et al. 2020

Training a denoising diffusion probabilistic model

Encoder (**fixed**): $q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$

$\mathbf{x}_0$ → $\mathbf{x}_T$

$q_0(\mathbf{x}_0) = p_{\text{data}}(\mathbf{x}_0)$

$p(\mathbf{x}_T)$

Decoder: $p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$

$\mathbf{x}_0$ ← $\mathbf{x}_T$

ELBO loss (negative ELBO averaged over data distribution):

$$\mathbb{E}_{q(\mathbf{x}_0)}\left[-\log p_\theta(\mathbf{x}_0)\right] \leq \mathbb{E}_{q(\mathbf{x}_0)q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] =: L$$

$\log p(x) = \log \mathbb{E}_{q(z|x)}\left[\frac{p(x,z)}{q(z|x)}\right]$ **<= Regular VAE**

DDPM, Ho et al. 2021
Song et al. 2020

Training a denoising diffusion probabilistic model

Encoder (q) has no learnable parameters!

Encoder (**fixed**): $q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$

$\mathbf{x}_0$ → $\mathbf{x}_T$

$q_0(\mathbf{x}_0) = p_{\text{data}}(\mathbf{x}_0)$

$p(\mathbf{x}_T)$

Decoder: $p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$

$\mathbf{x}_0$ ← $\mathbf{x}_T$

ELBO loss (negative ELBO averaged over data distribution):

$$\mathbb{E}_{q(\mathbf{x}_0)}\left[-\log p_\theta(\mathbf{x}_0)\right] \leq \mathbb{E}_{q(\mathbf{x}_0)q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] =: L$$

$\log p(x) = \log \mathbb{E}_{q(z|x)}\left[\frac{p(x,z)}{q(z|x)}\right]$ **<= Regular VAE**

DDPM, Ho et al. 2021
Song et al. 2020

Diffusion models as score-based models

The ELBO objective is

$$\mathbb{E}_{q(\mathbf{x}_0)}\left[-\log p_\theta(\mathbf{x}_0)\right] \leq \mathbb{E}_{q(\mathbf{x}_0)q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] =: L$$

Decoder parameterization: $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_t^2\mathbf{I})$

DDPM, Ho et al. 2021
Song et al. 2020

Diffusion models as score-based models

The ELBO objective is

$$\mathbb{E}_{q(\mathbf{x}_0)}\left[-\log p_\theta(\mathbf{x}_0)\right] \leq \mathbb{E}_{q(\mathbf{x}_0)q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] =: L$$

Decoder parameterization: $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_t^2\mathbf{I})$

$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{1-\beta_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\right)$$

DDPM, Ho et al. 2021
Song et al. 2020

Diffusion models as score-based models

The ELBO objective is

$$\mathbb{E}_{q(\mathbf{x}_0)}\left[-\log p_\theta(\mathbf{x}_0)\right] \leq \mathbb{E}_{q(\mathbf{x}_0)q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] =: L$$

Decoder parameterization: $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_t^2\mathbf{I})$

$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{1-\beta_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\right)$$

Up to scaling, predict noise that was added and subtract it

DDPM, Ho et al. 2021
Song et al. 2020

Diffusion models as score-based models

The ELBO objective is

$$\mathbb{E}_{q(\mathbf{x}_0)}\left[-\log p_\theta(\mathbf{x}_0)\right] \leq \mathbb{E}_{q(\mathbf{x}_0)q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] =: L$$

Decoder parameterization: $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_t^2\mathbf{I})$

$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{1-\beta_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\right)$$

Up to scaling, predict noise that was added and subtract it

$$L = \mathbb{E}_{\mathbf{x}_0\sim q(\mathbf{x}_0), t\sim\mathcal{U}\{1,T\}, \epsilon\sim\mathcal{N}(0,\mathbf{I})}\left[\lambda_t \|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)\|^2\right]$$

DDPM, Ho et al. 2021
Song et al. 2020

Diffusion models as score-based models

The ELBO objective is

$$\mathbb{E}_{q(\mathbf{x}_0)}\left[-\log p_\theta(\mathbf{x}_0)\right] \leq \mathbb{E}_{q(\mathbf{x}_0)q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] =: L$$

Decoder parameterization: $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_t^2\mathbf{I})$

$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{1-\beta_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\right)$$

Up to scaling, predict noise that was added and subtract it

$$L = \mathbb{E}_{\mathbf{x}_0\sim q(\mathbf{x}_0), t\sim\mathcal{U}\{1,T\}, \epsilon\sim\mathcal{N}(0,\mathbf{I})}\left[\lambda_t \|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)\|^2\right]$$

ELBO loss reduces to denoising score matching!

DDPM, Ho et al. 2021
Song et al. 2020

Training and inference

Algorithm 1 Training

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

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

DDPM, Ho et al. 2021
Song et al. 2020

Training and inference

Algorithm 1 Training

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

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

Denoising score matching over multiple noise scales indexed by t

$$= \frac{1}{L} \sum_{i=1}^{L} E_{\mathbf{x} \sim p_{\text{data}}, \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})} \left[\left\| \boldsymbol{\epsilon}_\theta(\mathbf{x} + \sigma_i \mathbf{z}, \sigma_i) + \mathbf{z} \right\|_2^2 \right] \qquad [\ \boldsymbol{\epsilon}_\theta(\cdot, \sigma_i) := \sigma_i \boldsymbol{s}_\theta(\cdot, \sigma_i)\ ]$$

DDPM, Ho et al. 2021
Song et al. 2020

Training and inference

Algorithm 1 Training

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

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

Denoising score matching over multiple noise scales indexed by t

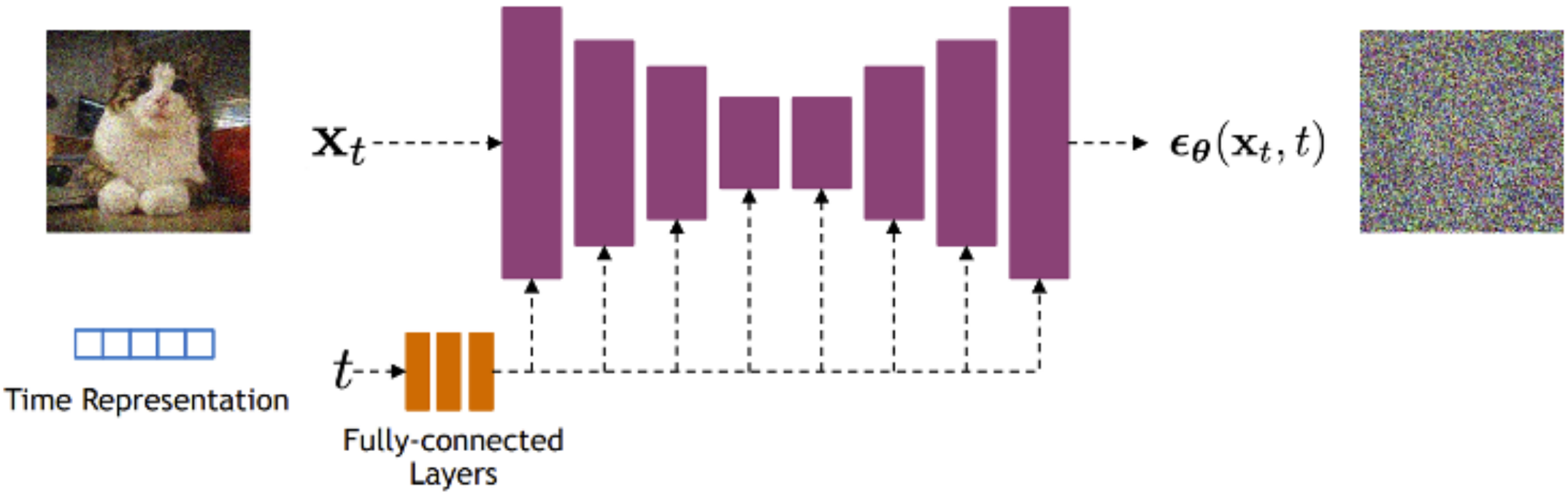
$$= \frac{1}{L} \sum_{i=1}^{L} E_{\mathbf{x} \sim p_{\text{data}}, \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})} \left[\left\| \boldsymbol{\epsilon}_\theta(\mathbf{x} + \sigma_i \mathbf{z}, \sigma_i) + \mathbf{z} \right\|_2^2 \right] \qquad \left[\boldsymbol{\epsilon}_\theta(\cdot, \sigma_i) := \sigma_i \boldsymbol{s}_\theta(\cdot, \sigma_i) \right]$$

Iteratively sample from decoders $p_\theta(x_{t-1} \mid x_t)$
(Annealed Langevin Dynamics)

DDPM, Ho et al. 2021
Song et al. 2020

Architecture of the denoiser

Unet architecture used in practice



Same as the noise-conditioned score network (t represents time index or equivalently the noise level)

DDPM, Ho et al. 2021
Song et al. 2020

High-Fidelity Generation for 1024x1024 Images

DDPM, Ho et al. 2021
Song et al. 2020

Why denoising is so powerful?

- Because it estimates the gradients of the log likelihood of data and neural nets are great at following gradients.
- Because it can populate low data density regions.
- Because hierarchical VAE is more powerful than VAE. It increases the dimensionality t times

Quiz!

Quiz!

- Why does DDPM use a Markov chain to model the diffusion process?

Quiz!

- True or False?

Quiz!

- Initial data distribution must be Gaussian.

Quiz!

- Transition kernel can be any distribution.

Quiz!

- Noisy distribution at the final time step can be non-Gaussian.

Quiz!

- If we assume infinite number of noise levels in diffusion, does VAE perspective hold?

Quiz!

- How would you define the 'shortest path' between two probability distributions in a generative model?

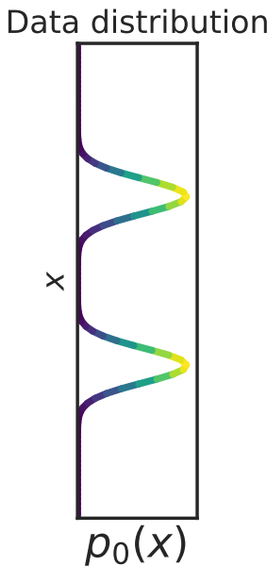
Quiz!

- Assuming you're working with images, what are the possible spaces in which you could formulate the diffusion process?
- What’s autoregressive about diffusion?
- SNR at the end of diffusion process high or low?
- One diffusion perspective suggests it is a kind of hierarchical-vae. Given how successful diffusion is, why don’t we train more and more hierarchical-vaes?
- Did you notice in diffusion-vae, encoder is fixed? If yes, does it seem odd?

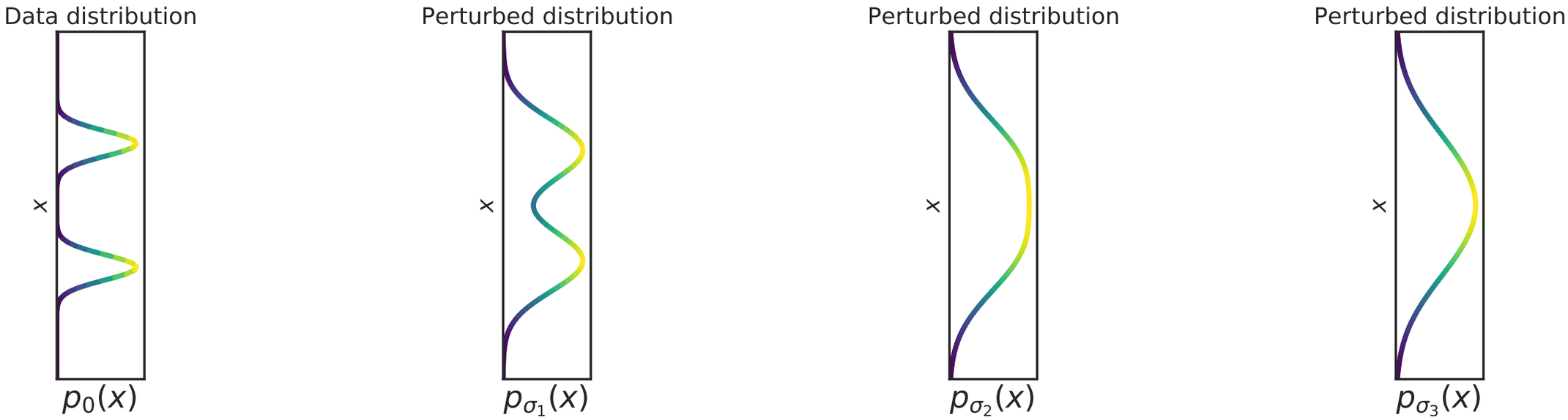
Outline

- Infinite noise levels
- Perturbing data with stochastic processes
- Generation via reverse stochastic process
- Score-based generative modeling via SDEs
- Converting the SDE to an ODE
- Evaluating likelihoods with ODEs (flow model)
- Distillation
- Latent diffusion model
- Control the generation process

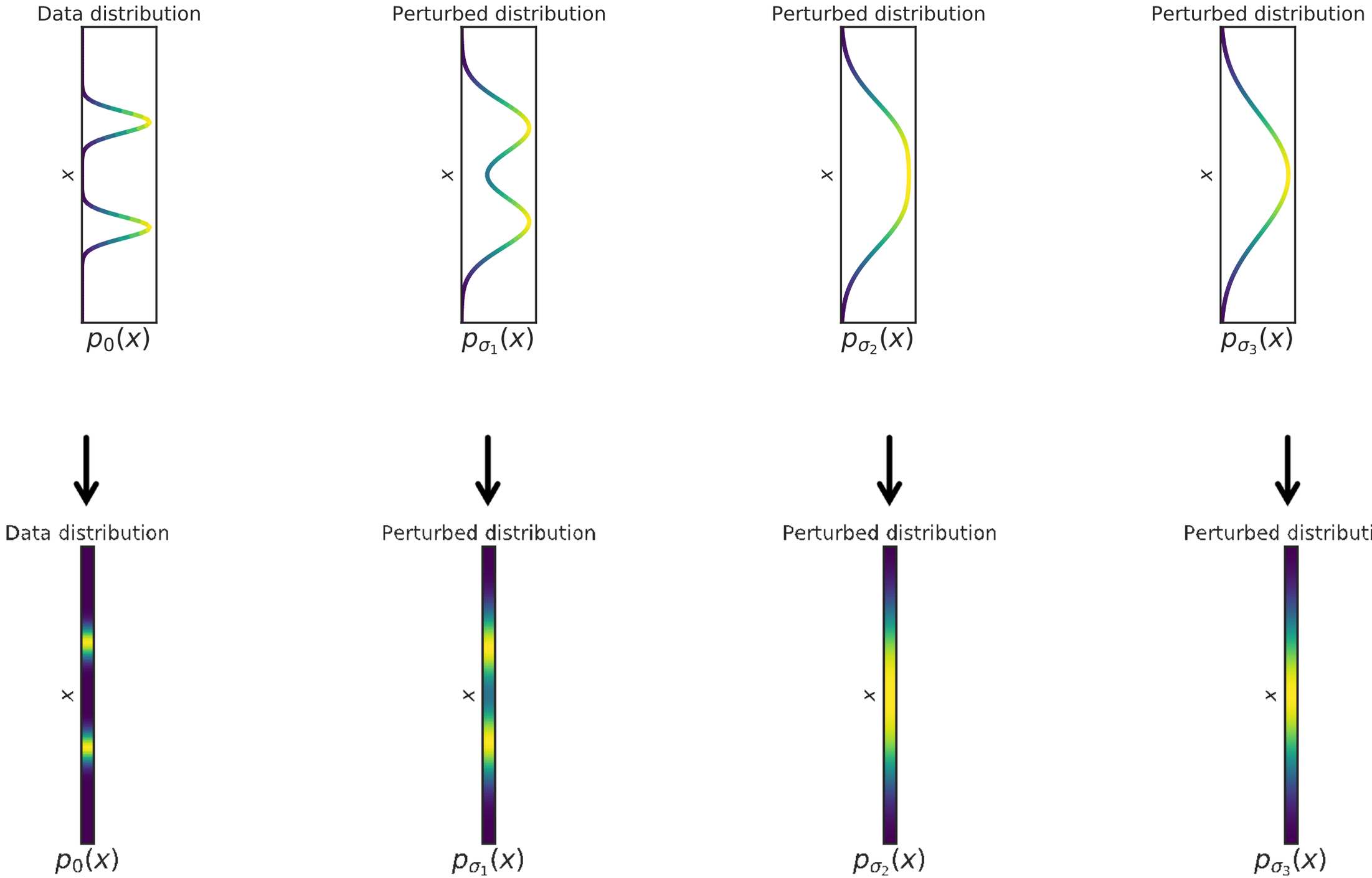
Infinite noise levels



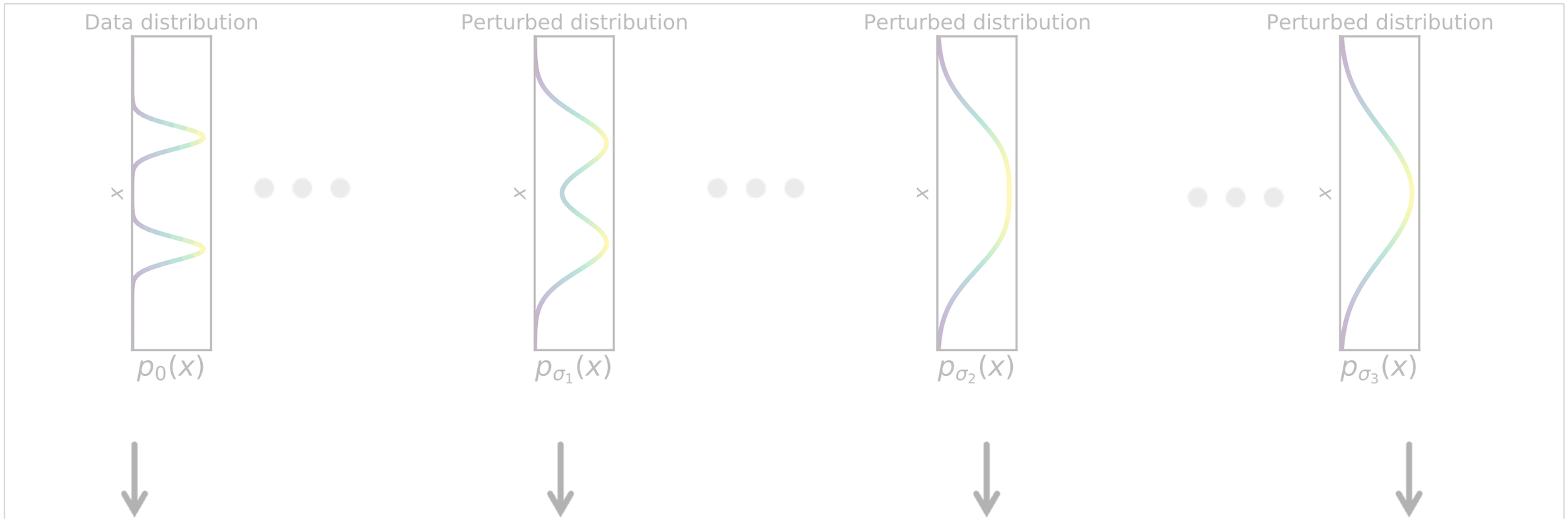
Infinite noise levels



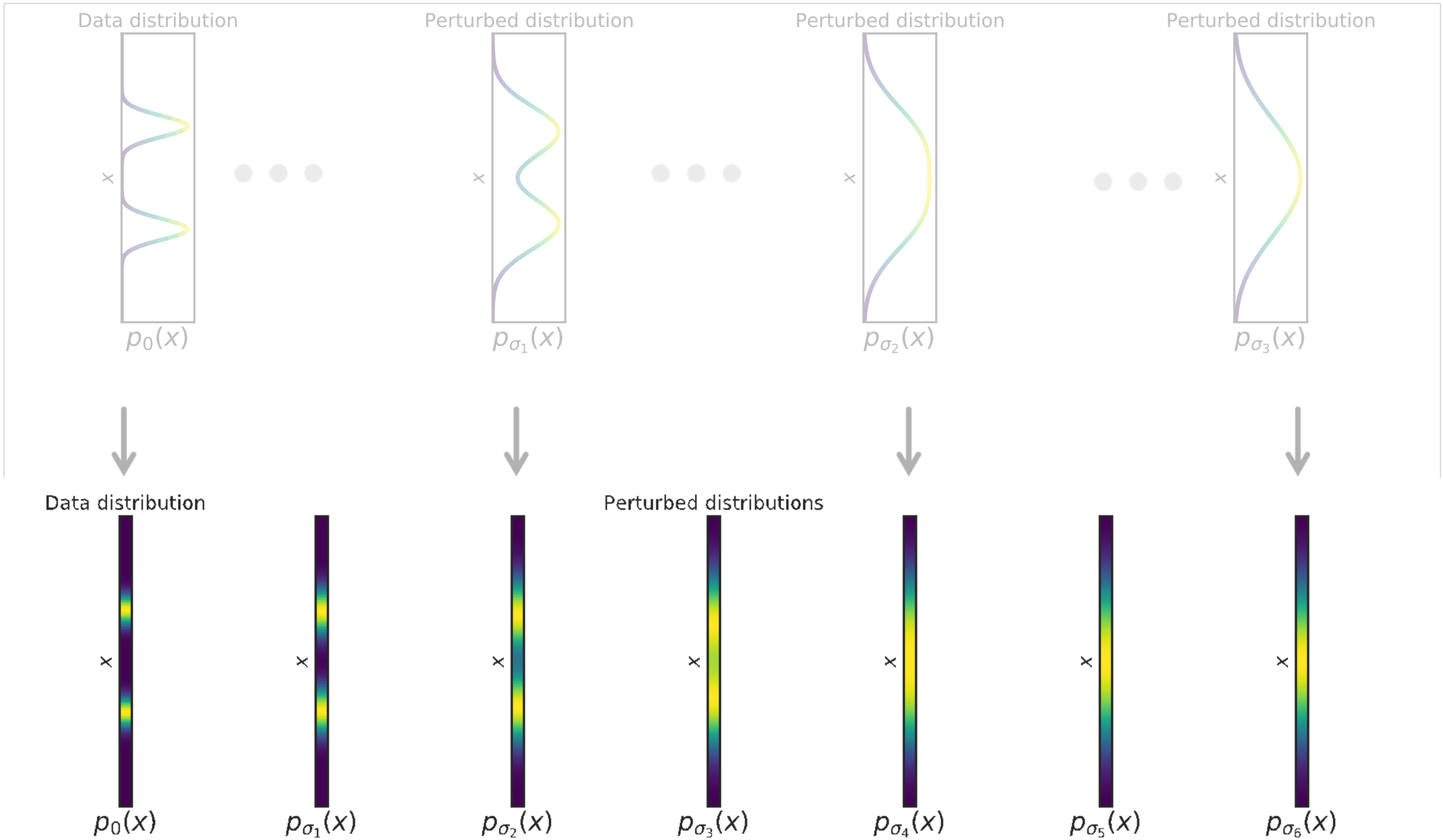
Infinite noise levels



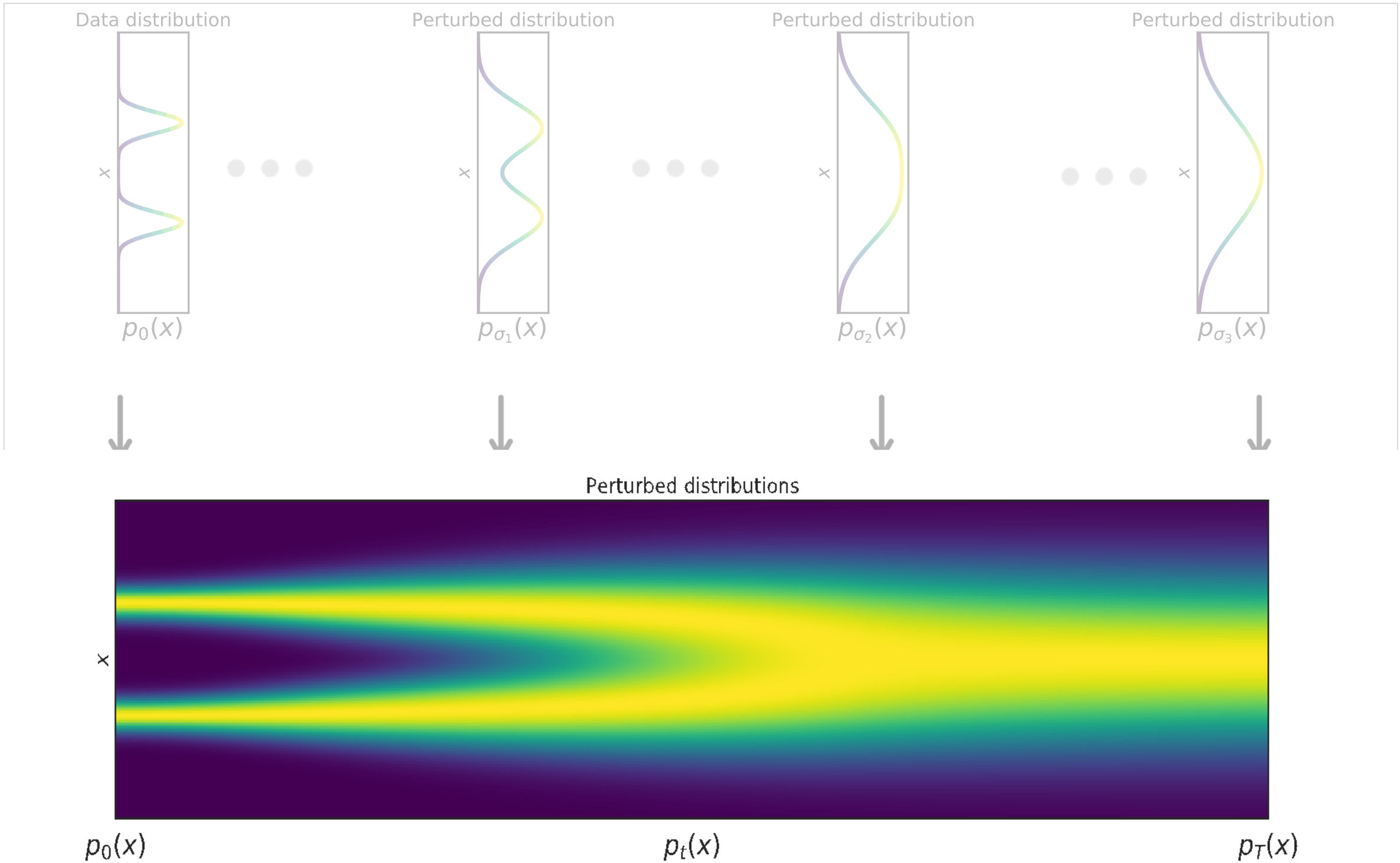
Infinite noise levels



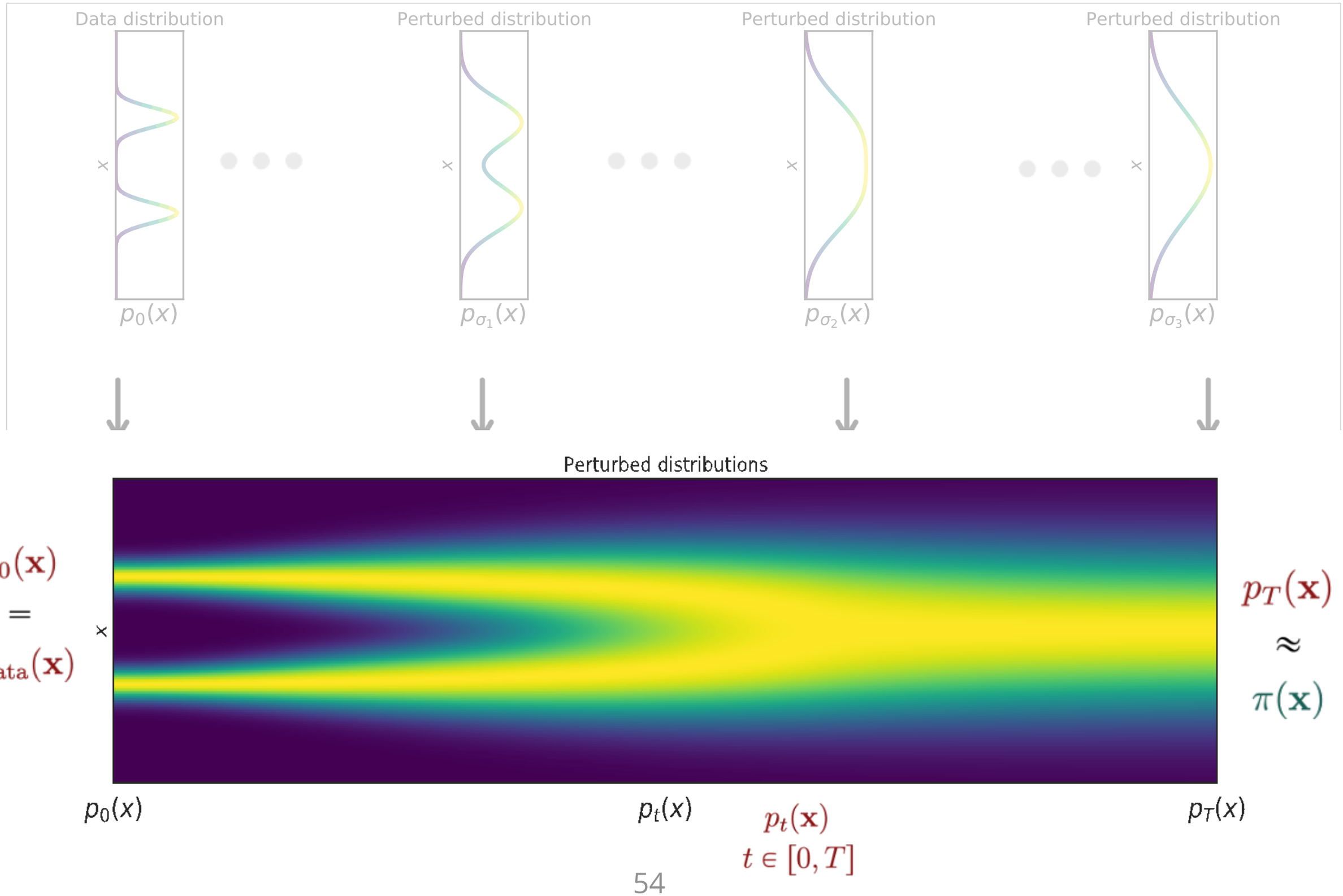
Infinite noise levels



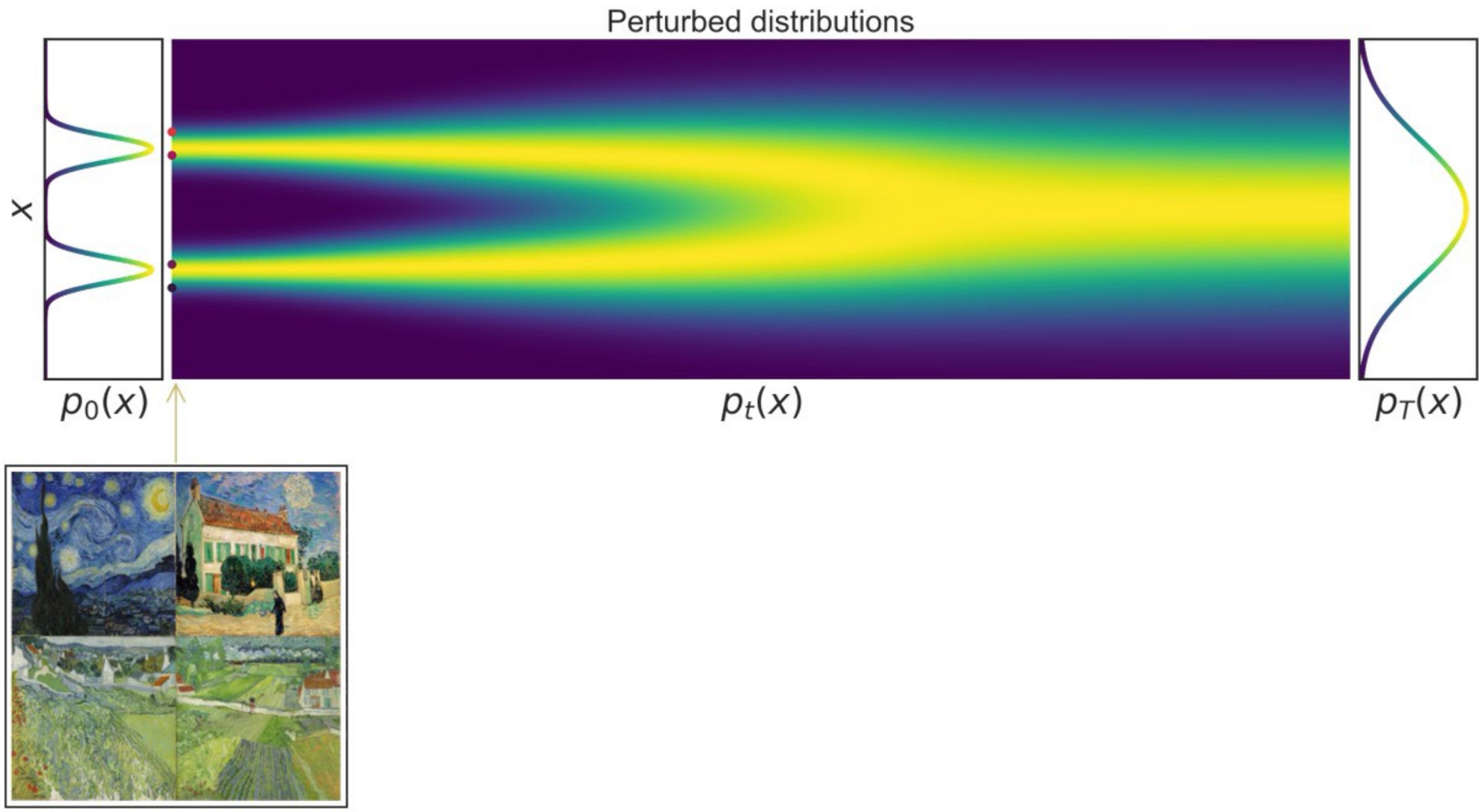
Infinite noise levels



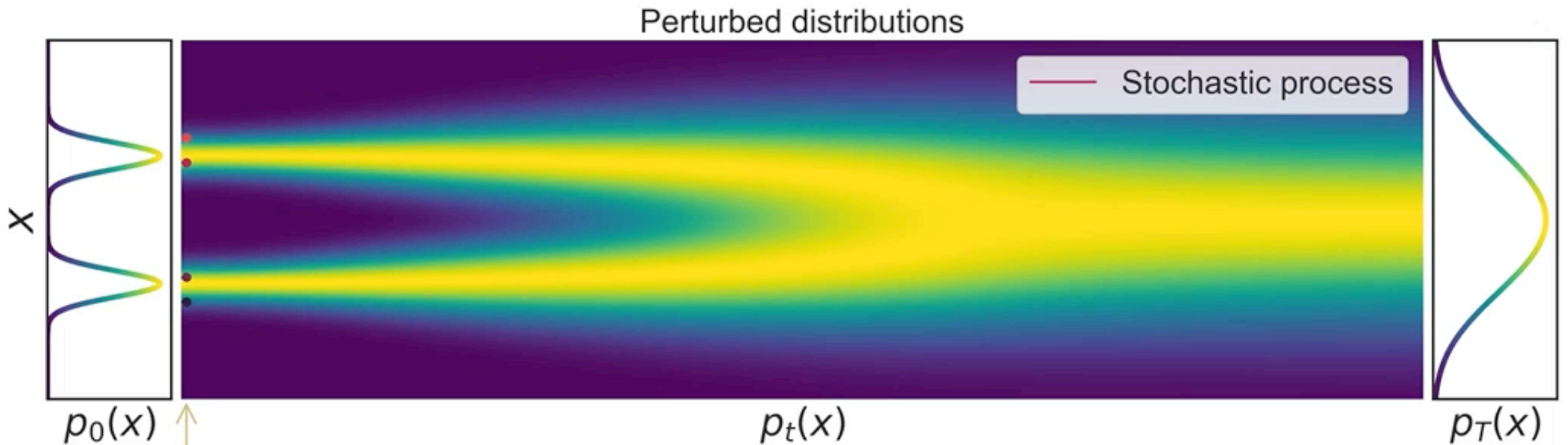
Infinite noise levels



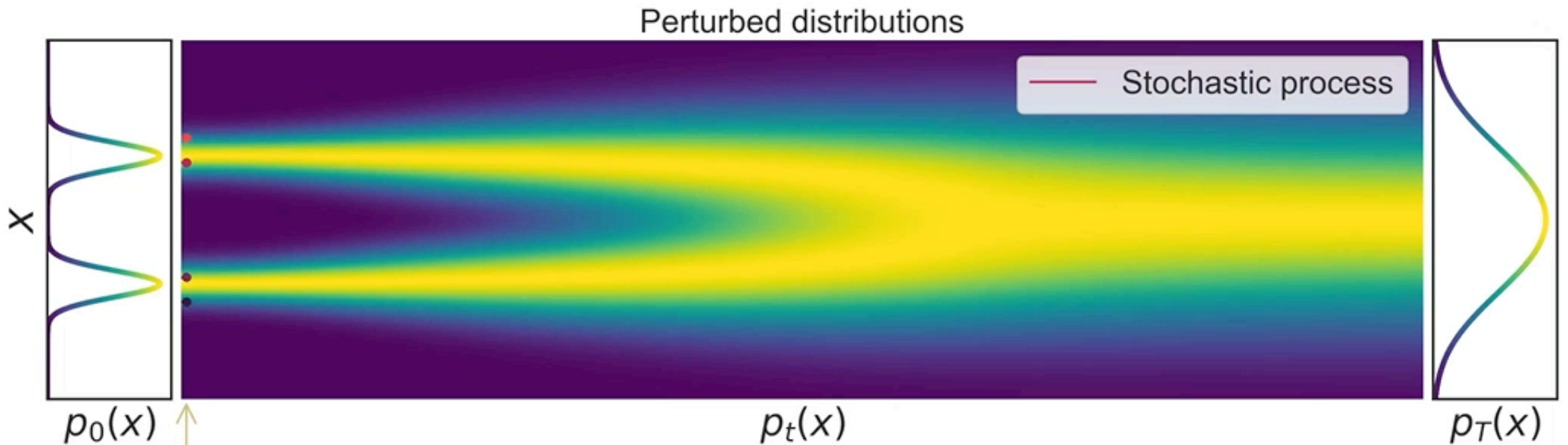
Perturbing data with stochastic processes



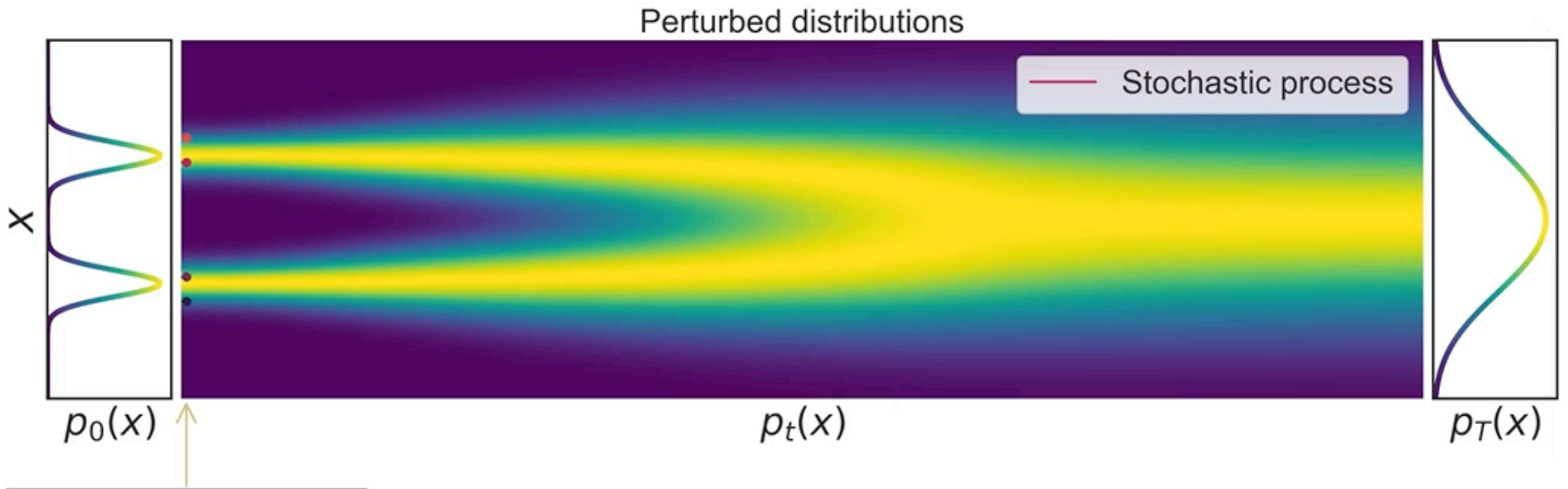
Perturbing data with stochastic processes



Perturbing data with stochastic processes



Perturbing data with stochastic processes

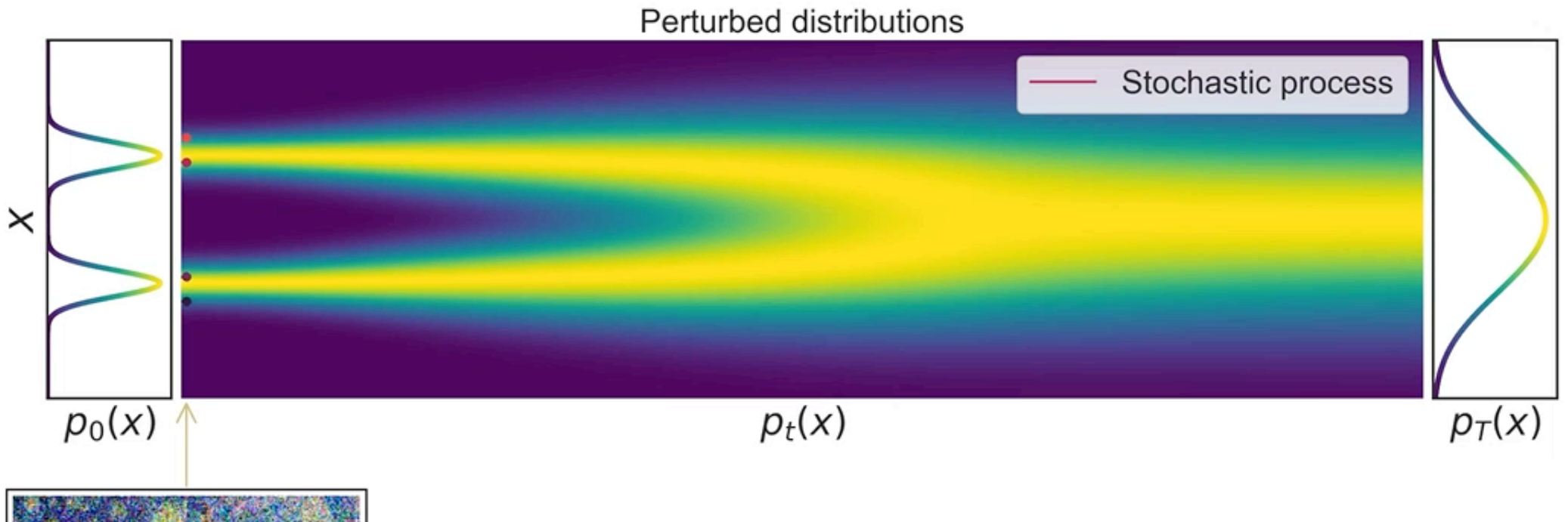


Stochastic process

$\{\mathbf{x}_t\}_{t\in[0,T]}$

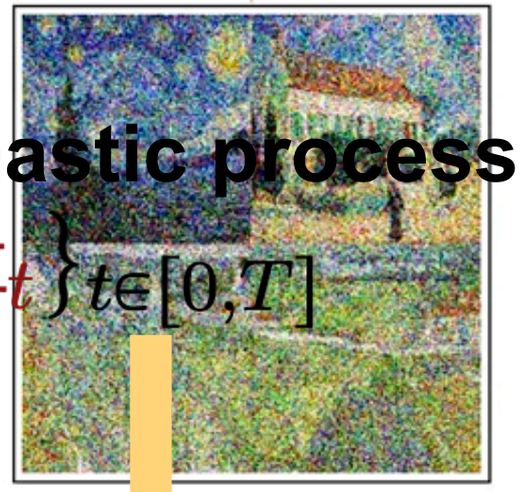
$$\pi(\mathbf{x}) \approx p_T(\mathbf{x})$$

Perturbing data with stochastic processes



Stochastic process

$\{\mathbf{x}_t\}_{t\in[0,T]}$



Probability densities

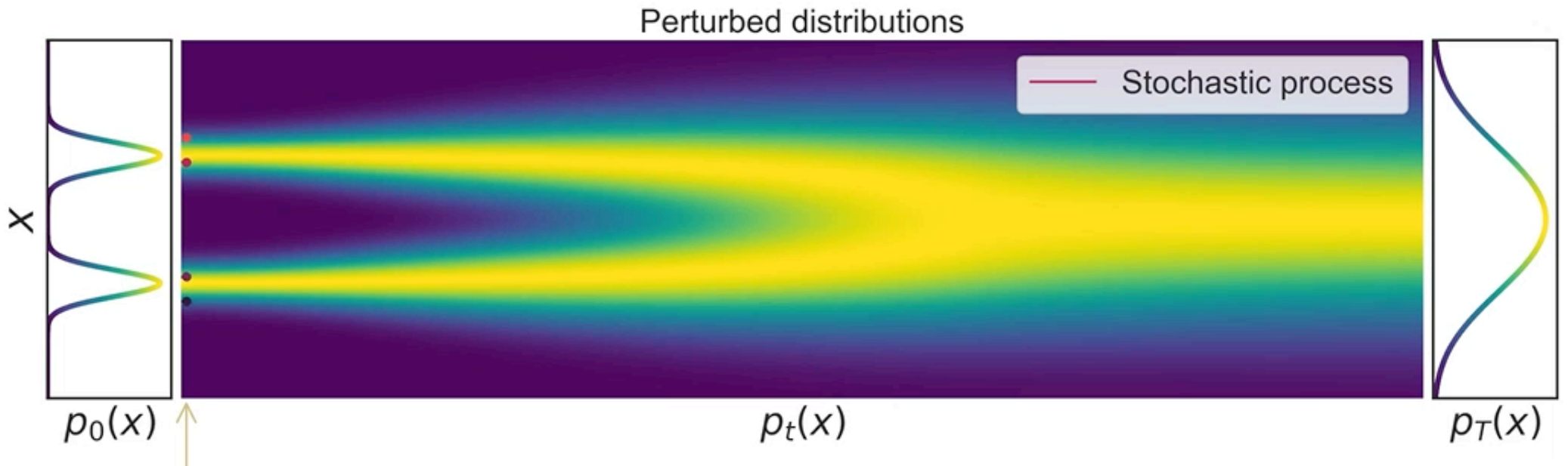
$\{p_t(\mathbf{x})\}_{t\in[0,T]}$

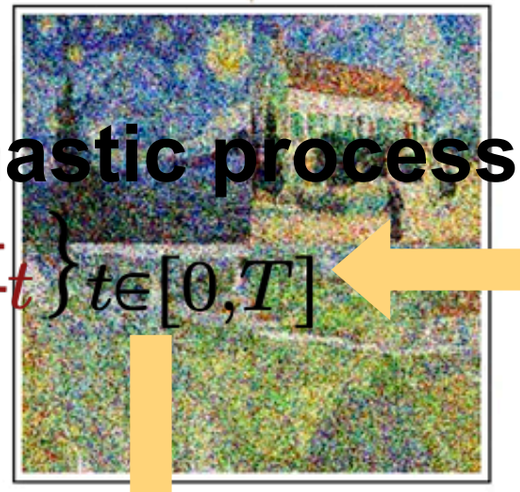
$\pi(\mathbf{x})$

$\approx$

$p_T(\mathbf{x})$

Perturbing data with stochastic processes





Stochastic process

$$\{\mathbf{x}_t\}_{t\in[0,T]}$$

Stochastic differential equation (SDE)

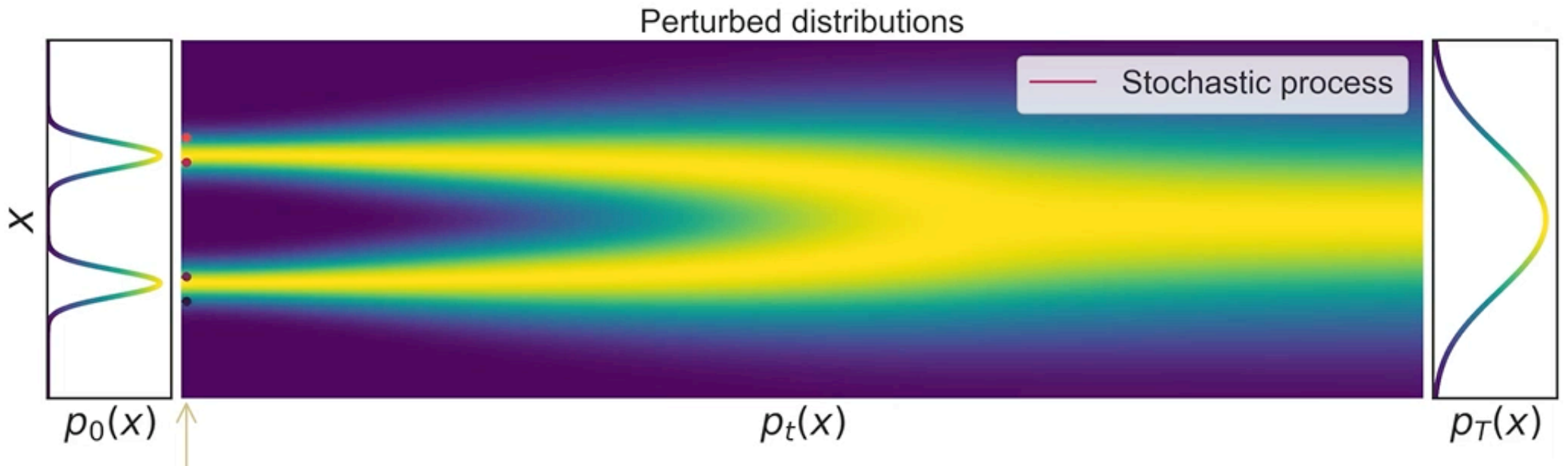
$$\mathrm{d}\mathbf{x}_t = \boldsymbol{f}(\mathbf{x}_t, t)\,\mathrm{d}t + g(t)\,\mathrm{d}\mathbf{w}_t$$

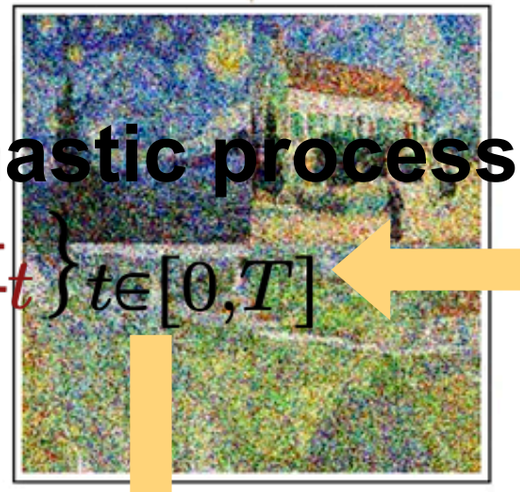
Probability densities

$$\{p_t(\mathbf{x})\}_{t\in[0,T]}$$

$$\pi(\mathbf{x}) \approx p_T(\mathbf{x})$$

Perturbing data with stochastic processes





Stochastic process

$$\{\mathbf{x}_t\}_{t\in[0,T]}$$

Stochastic differential equation (SDE)

$$\mathrm{d}\mathbf{x}_t = \boldsymbol{f}(\mathbf{x}_t, t)\,\mathrm{d}t + g(t)\,\mathrm{d}\mathbf{w}_t$$

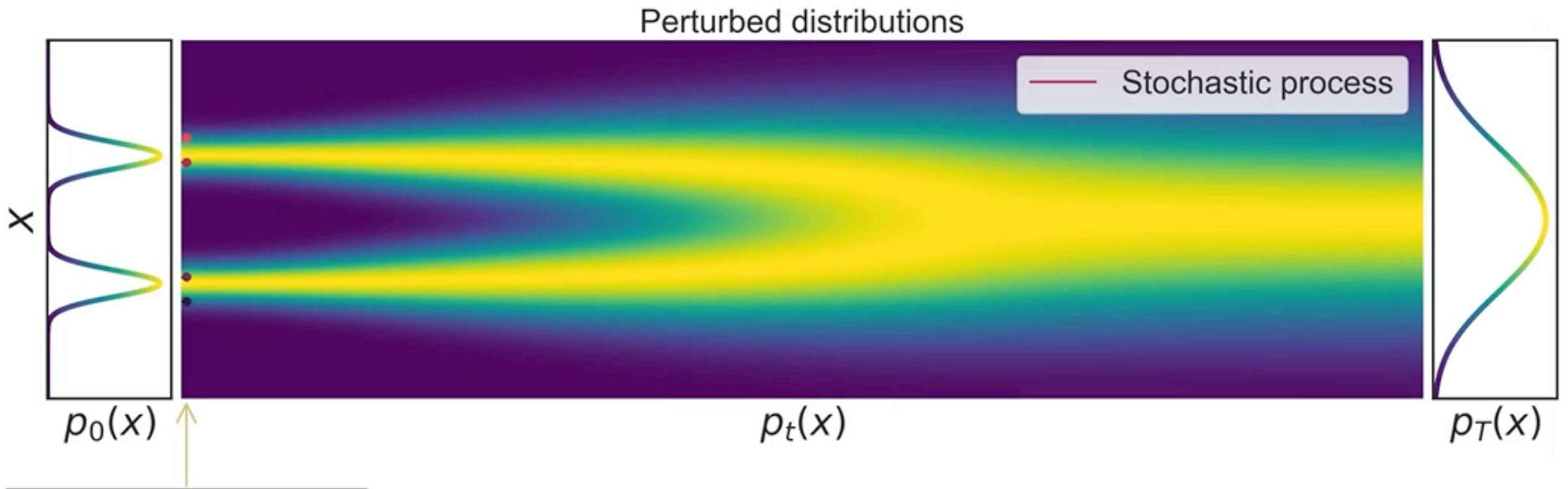
Deterministic drift

Probability densities

$$\{p_t(\mathbf{x})\}_{t\in[0,T]}$$

$$\pi(\mathbf{x}) \approx p_T(\mathbf{x})$$

Perturbing data with stochastic processes



Stochastic process

$$\{\mathbf{x}_t\}_{t\in[0,T]}$$

Stochastic differential equation (SDE)

$$\mathrm{d}\mathbf{x}_t = \boldsymbol{f}(\mathbf{x}_t, t)\,\mathrm{d}t + g(t)\,\mathrm{d}\mathbf{w}_t$$

Deterministic drift

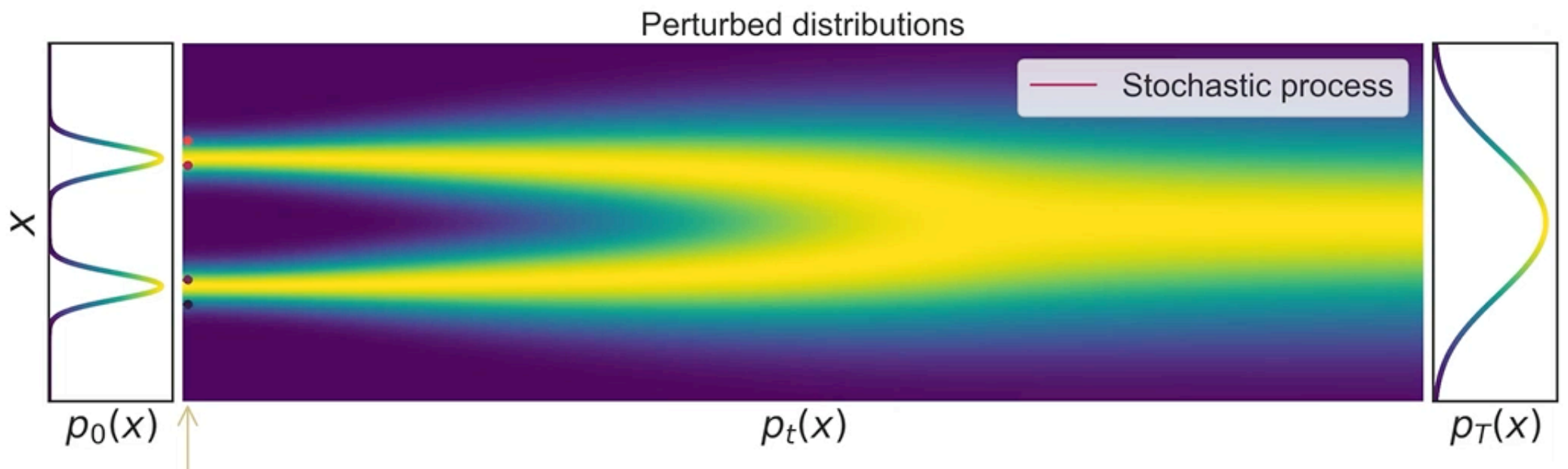
Infinitesimal noise

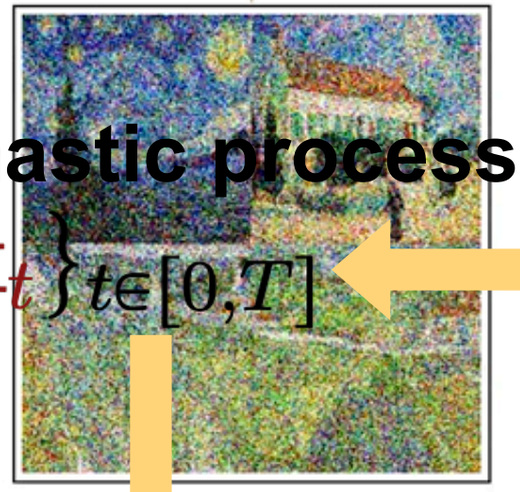
Probability densities

$$\{p_t(\mathbf{x})\}_{t\in[0,T]}$$

$$\pi(\mathbf{x}) \approx p_T(\mathbf{x})$$

Perturbing data with stochastic processes





Stochastic process

$$\{\mathbf{x}_t\}_{t\in[0,T]}$$

Stochastic differential equation (SDE)

$$\mathrm{d}\mathbf{x}_t = \boldsymbol{f}(\mathbf{x}_t, t)\,\mathrm{d}t + g(t)\,\mathrm{d}\mathbf{w}_t$$

Deterministic drift

Infinitesimal noise

Probability densities

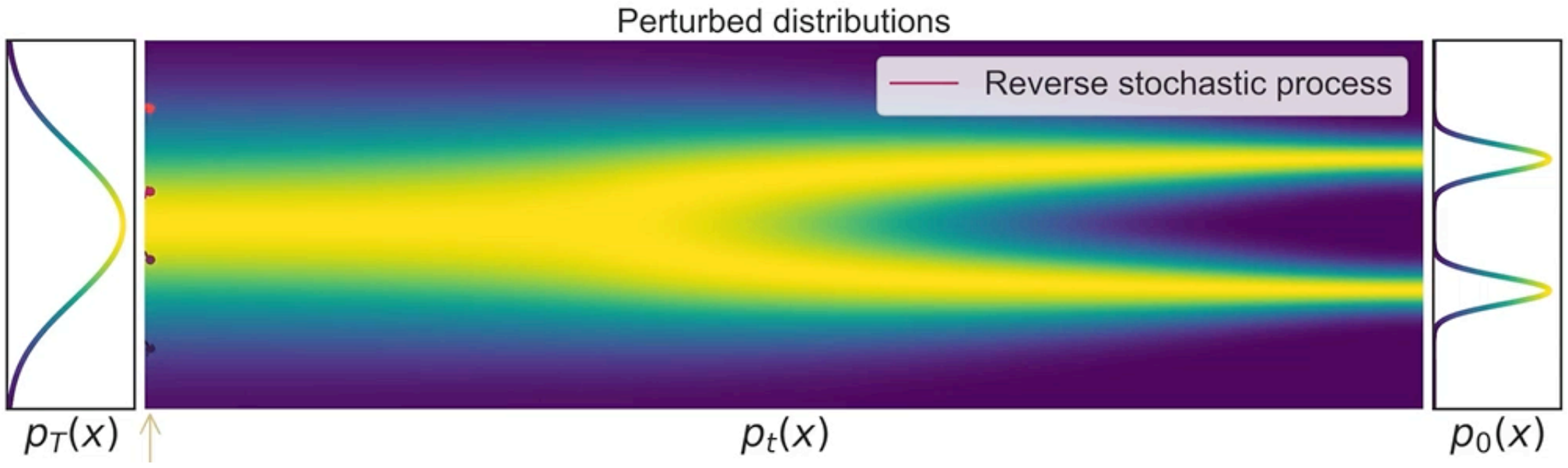
$$\{p_t(\mathbf{x})\}_{t\in[0,T]}$$

WLOG: Toy SDE

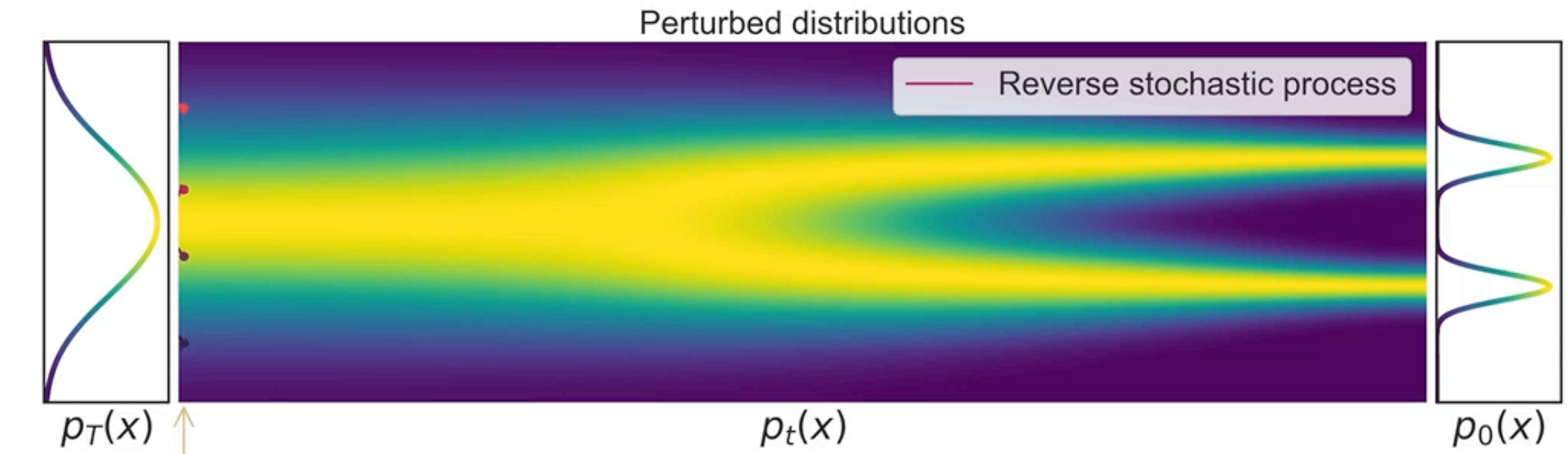
$$\mathrm{d}\mathbf{x}_t = \sigma(t)\,\mathrm{d}\mathbf{w}_t$$

$$\pi(\mathbf{x}) \approx p_T(\mathbf{x})$$

Generation via reverse stochastic process

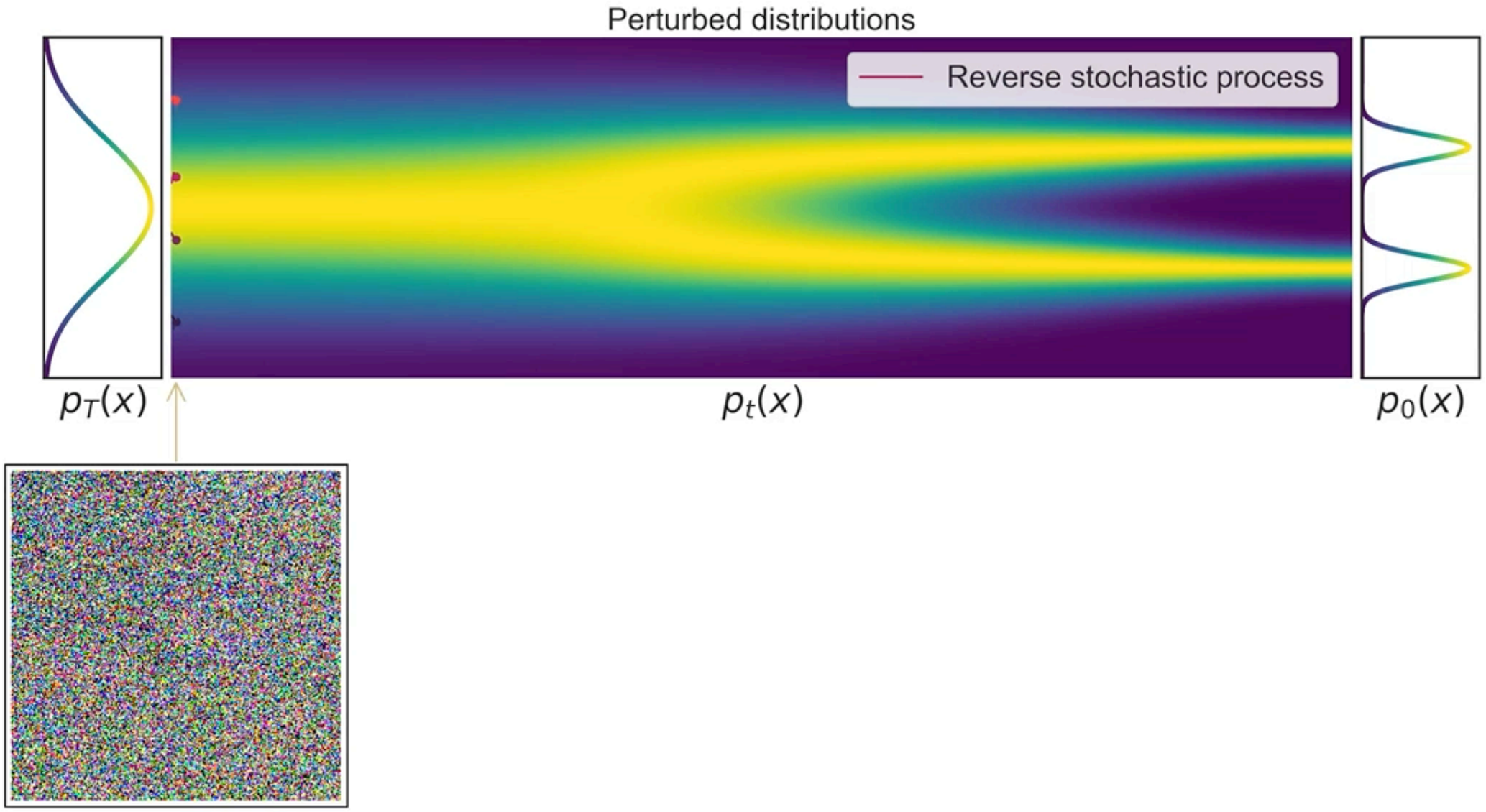


Generation via reverse stochastic process

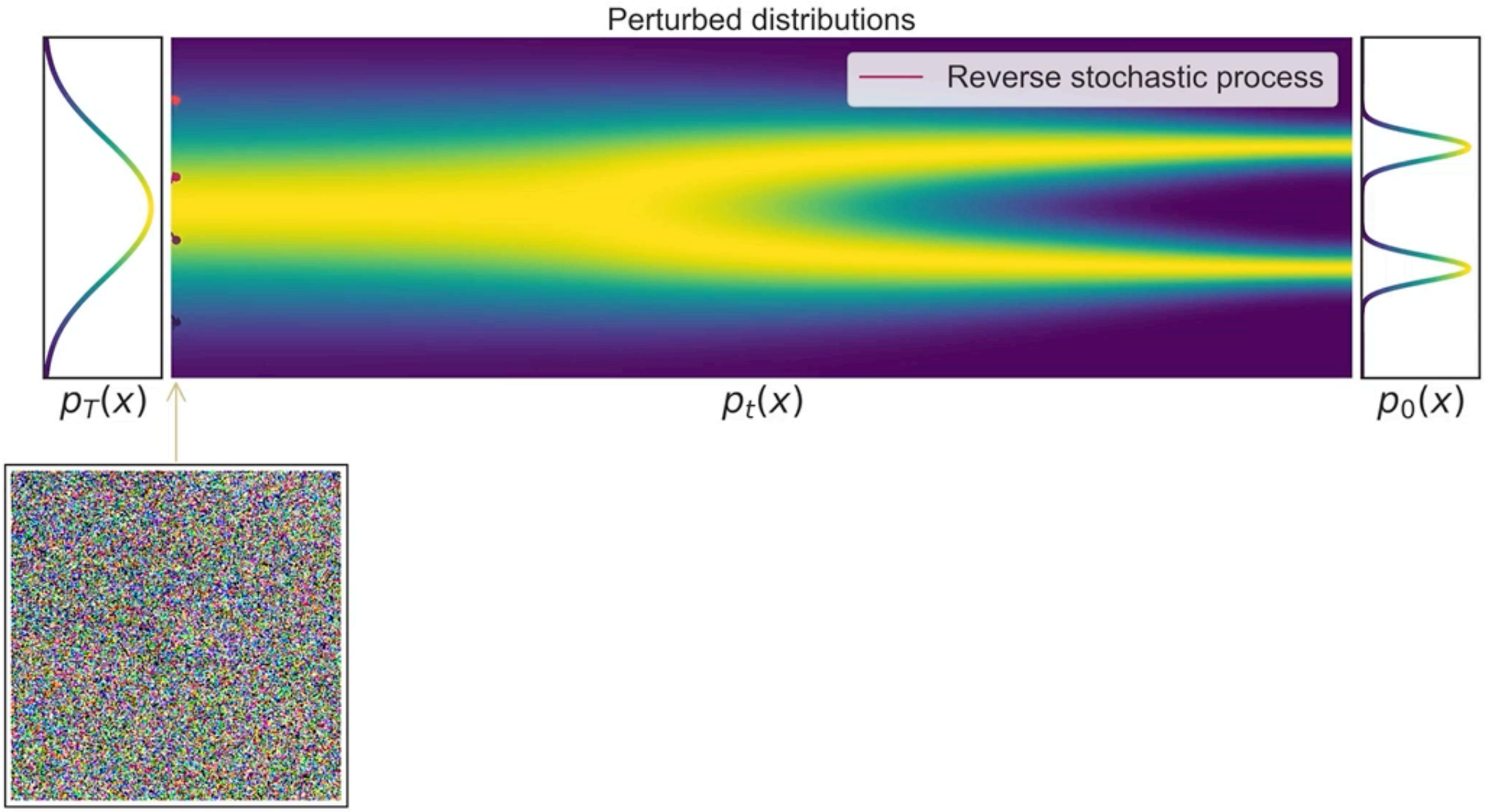


$$\pi(\mathbf{x}) \approx p_T(\mathbf{x})$$

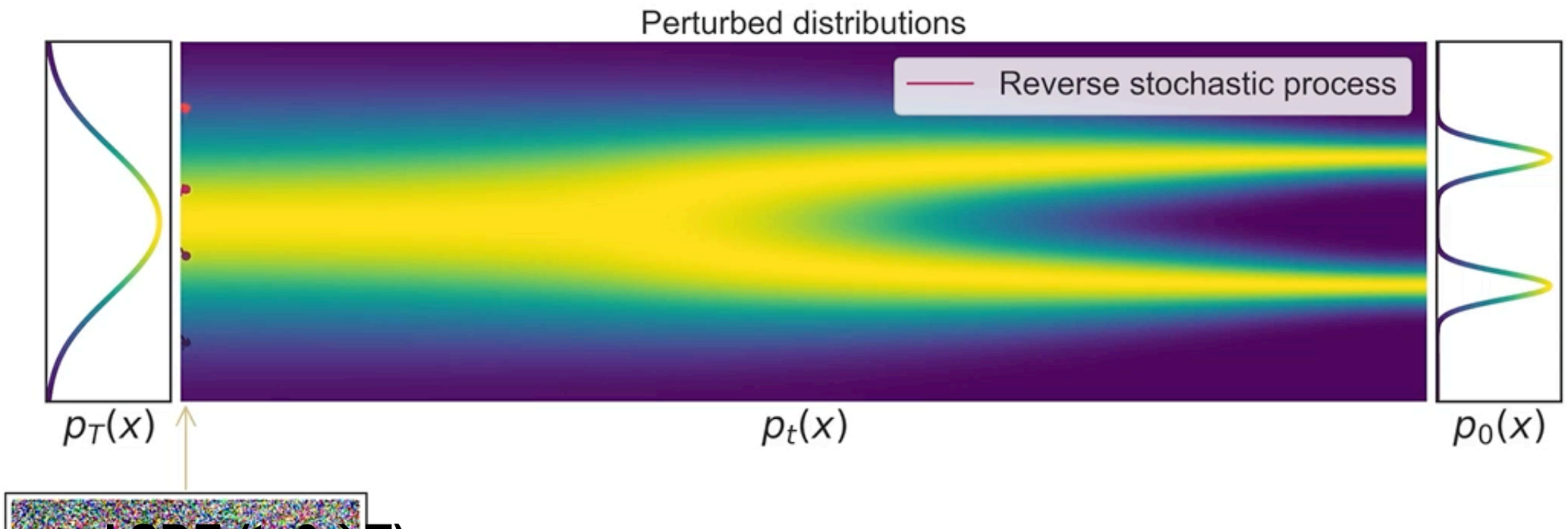
Generation via reverse stochastic process



Generation via reverse stochastic process



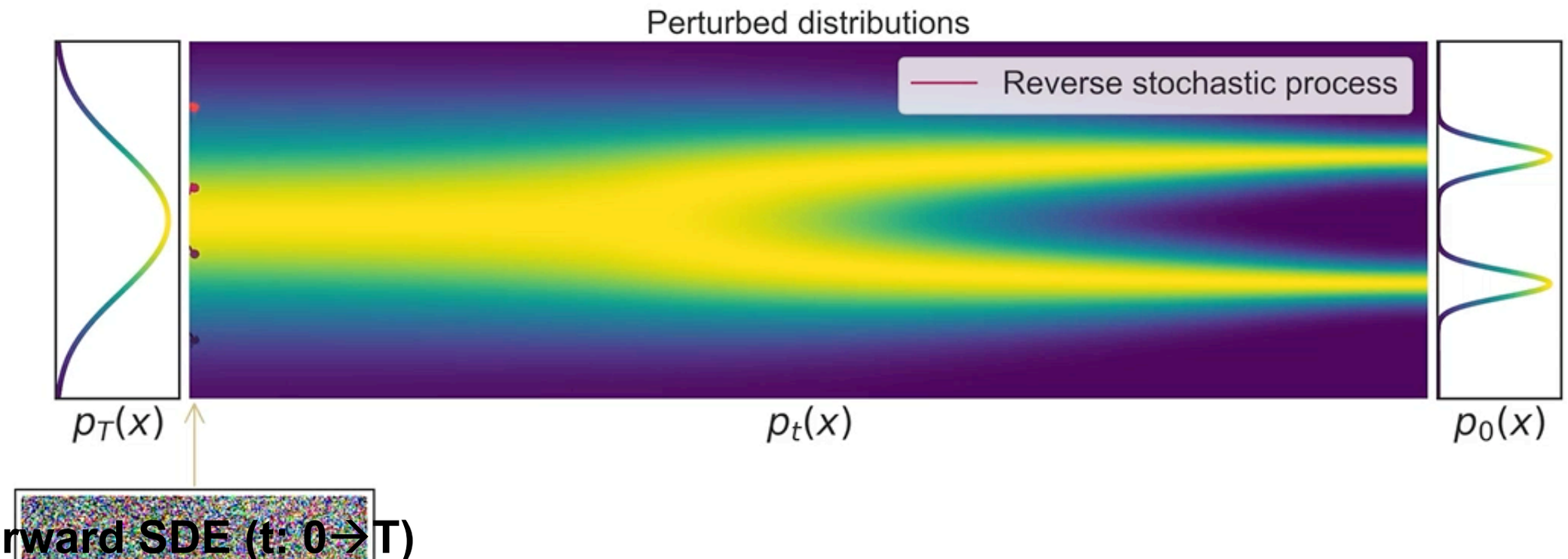
Generation via reverse stochastic process



Forward SDE (t: 0→T)

$$\mathrm{d}\mathbf{x}_t = \sigma(t)\,\mathrm{d}\mathbf{w}_t$$

Generation via reverse stochastic process



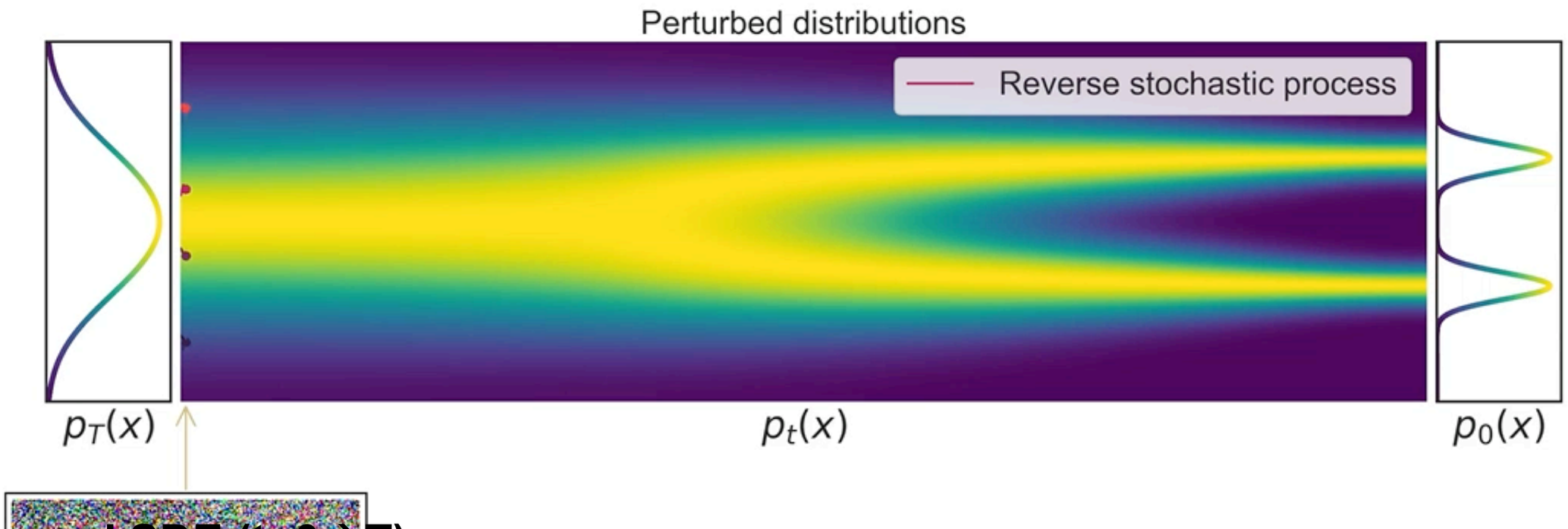
Forward SDE (t: 0→T)

$$\mathrm{d}\mathbf{x}_t = \sigma(t)\,\mathrm{d}\mathbf{w}_t$$

Reverse SDE (t: T→0)

$$\mathrm{d}\mathbf{x}_t = -\sigma(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t)\,\mathrm{d}t + \sigma(t)\,\mathrm{d}\bar{\mathbf{w}}_t$$

Generation via reverse stochastic process



Forward SDE (t: 0→T)

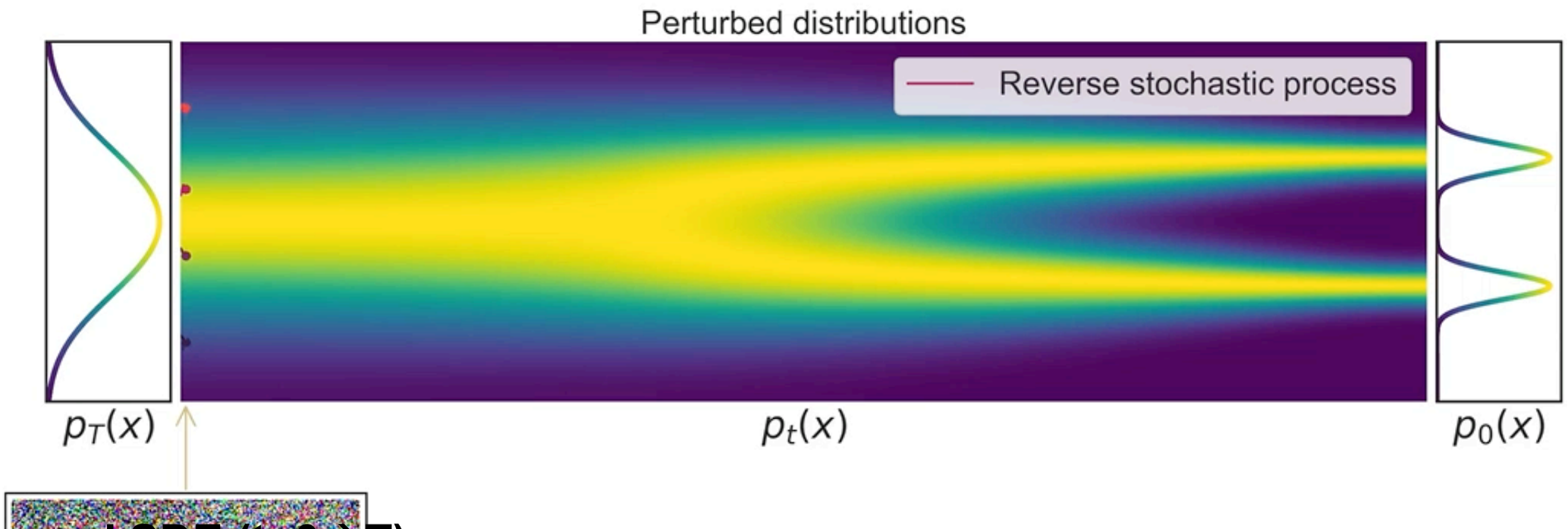
$$\mathrm{d}\mathbf{x}_t = \sigma(t)\,\mathrm{d}\mathbf{w}_t$$

Infinitesimal noise in the reverse time direction

Reverse SDE (t: T→0)

$$\mathrm{d}\mathbf{x}_t = -\sigma(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t)\,\mathrm{d}t + \sigma(t)\,\mathrm{d}\bar{\mathbf{w}}_t$$

Generation via reverse stochastic process



Forward SDE (t: 0→T)

$$\mathrm{d}\mathbf{x}_t = \sigma(t)\,\mathrm{d}\mathbf{w}_t$$

Infinitesimal noise in the reverse time direction

Reverse SDE (t: T→0)

$$\mathrm{d}\mathbf{x}_t = -\sigma(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t)\,\mathrm{d}t + \sigma(t)\,\mathrm{d}\bar{\mathbf{w}}_t$$

Score function!

Score-based generative modeling via SDEs

Score-based generative modeling via SDEs

- Time-dependent score-based model

Score-based generative modeling via SDEs

- Time-dependent score-based model

$$\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t) \approx \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$$

Score-based generative modeling via SDEs

- Time-dependent score-based model

$$\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t) \approx \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$$

- Training:

Score-based generative modeling via SDEs

- Time-dependent score-based model

$$\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t) \approx \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$$

- Training:

$$\mathbb{E}_{t \in \mathcal{U}(0,T)}[\lambda(t)\mathbb{E}_{p_t(\mathbf{x})}[\|\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)\|_2^2]]$$

Score-based generative modeling via SDEs

- Time-dependent score-based model

$$\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t) \approx \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$$

- Training:

$$\mathbb{E}_{t \in \mathcal{U}(0,T)}[\lambda(t)\mathbb{E}_{p_t(\mathbf{x})}[\|\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)\|_2^2]]$$

- Reverse-time SDE

Score-based generative modeling via SDEs

- Time-dependent score-based model

$$\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t) \approx \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$$

- Training:

$$\mathbb{E}_{t \in \mathcal{U}(0,T)}[\lambda(t)\mathbb{E}_{p_t(\mathbf{x})}[\|\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)\|_2^2]]$$

- Reverse-time SDE

$$\mathrm{d}\mathbf{x} = -\sigma^2(t)\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)\mathrm{d}t + \sigma(t)\mathrm{d}\bar{\mathbf{w}}$$

Score-based generative modeling via SDEs

- Time-dependent score-based model

$$\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t) \approx \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$$

- Training:

$$\mathbb{E}_{t \in \mathcal{U}(0,T)}[\lambda(t)\mathbb{E}_{p_t(\mathbf{x})}[\|\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)\|_2^2]]$$

- Reverse-time SDE

$$\mathrm{d}\mathbf{x} = -\sigma^2(t)\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)\mathrm{d}t + \sigma(t)\mathrm{d}\bar{\mathbf{w}}$$

- Euler-Maruyama (analgous to Euler for ODEs)

Score-based generative modeling via SDEs

- Time-dependent score-based model

$$\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t) \approx \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$$

- Training:

$$\mathbb{E}_{t \in \mathcal{U}(0,T)}[\lambda(t)\mathbb{E}_{p_t(\mathbf{x})}[\|\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)\|_2^2]]$$

- Reverse-time SDE

$$\mathrm{d}\mathbf{x} = -\sigma^2(t)\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)\mathrm{d}t + \sigma(t)\mathrm{d}\bar{\mathbf{w}}$$

- Euler-Maruyama (analgous to Euler for ODEs)

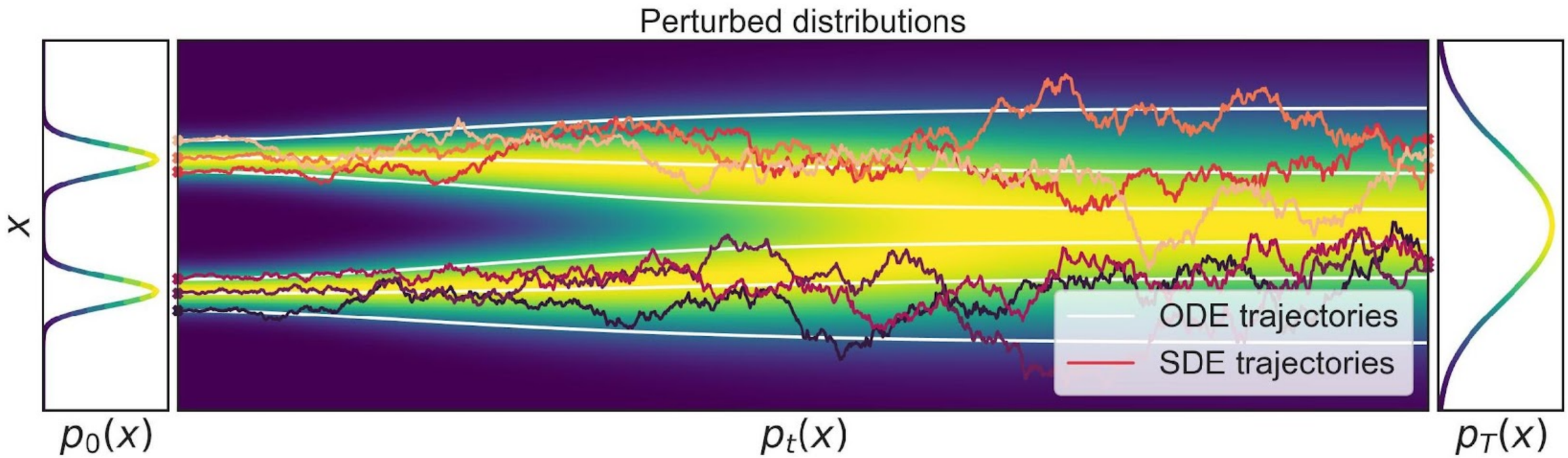
$$\mathbf{x} \leftarrow \mathbf{x} - \sigma(t)^2\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)\Delta t + \sigma(t)\ \mathbf{z} \quad (\mathbf{z} \sim \mathcal{N}(\mathbf{0}, |\Delta t|\ \mathbf{I}))$$

$$t \leftarrow t + \Delta t$$

Why denoising is so powerful?

- Because it estimates the gradients of the log likelihood of data and neural nets are great at following gradients.
- Because it can populate low data density regions.
- Because hierarchical VAE is more powerful than VAE. It increases the dimensionality t times.
- Because iterative denoising over infinite time steps is equivalent to solving reverse-time SDE and statistics physics has rich tools (some are decades-old) to offer for solving SDEs.

Converting the SDE to an ODE



SDE

$$\mathrm{d}\mathbf{x}_t = \sigma(t)\,\mathrm{d}\mathbf{w}_t$$

Ordinary differential equation (ODE)

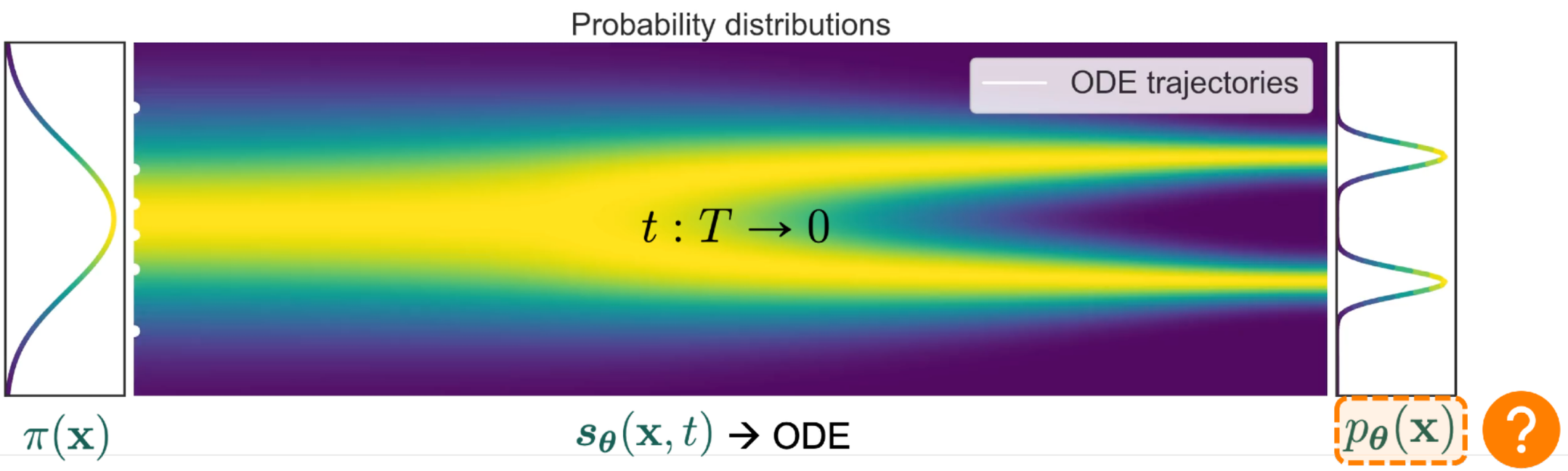
$$\frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t} = -\frac{1}{2}\sigma(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t)$$

Score function

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

Song et al. 2020

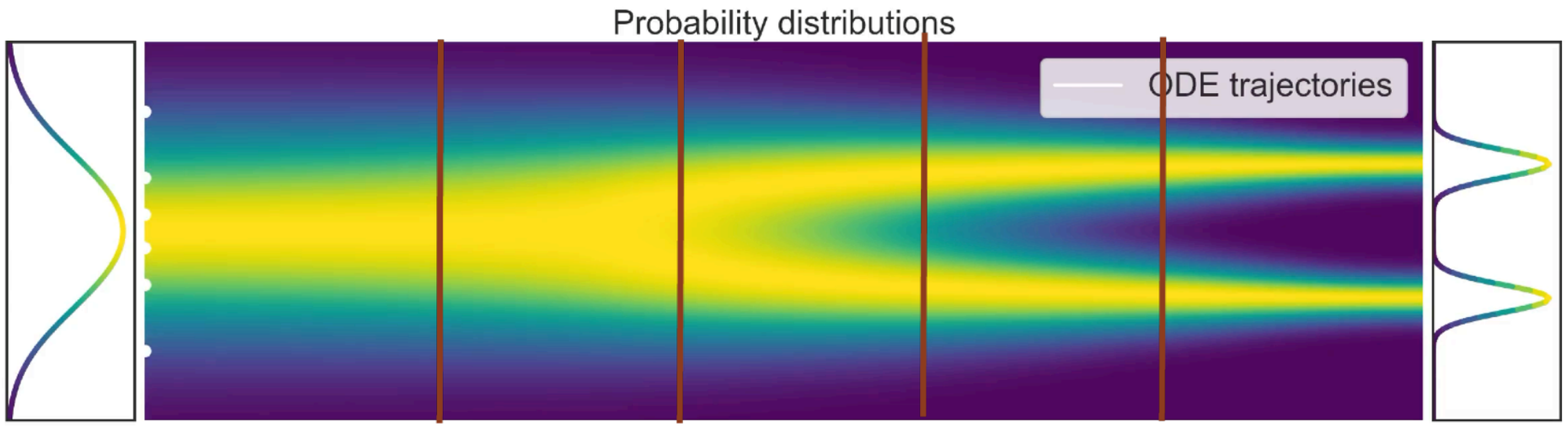
Evaluating likelihoods with ODEs (flow model)



Computing the probability density function (change of variables formula)

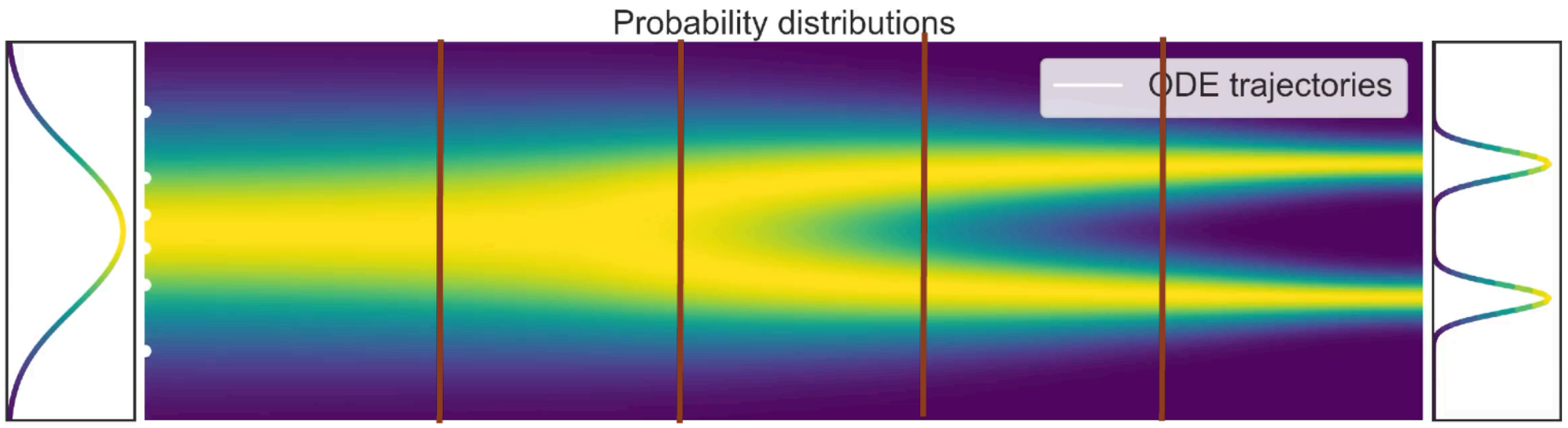
$$\log p_{\boldsymbol{\theta}}(\mathbf{x}_0) = \log \pi(\mathbf{x}_T) - \frac{1}{2} \int_0^T \sigma(t)^2 \operatorname{trace}(\nabla_{\mathbf{x}} \boldsymbol{s_\theta}(\mathbf{x}, t)) \mathrm{d}t$$

Accelerated sampling



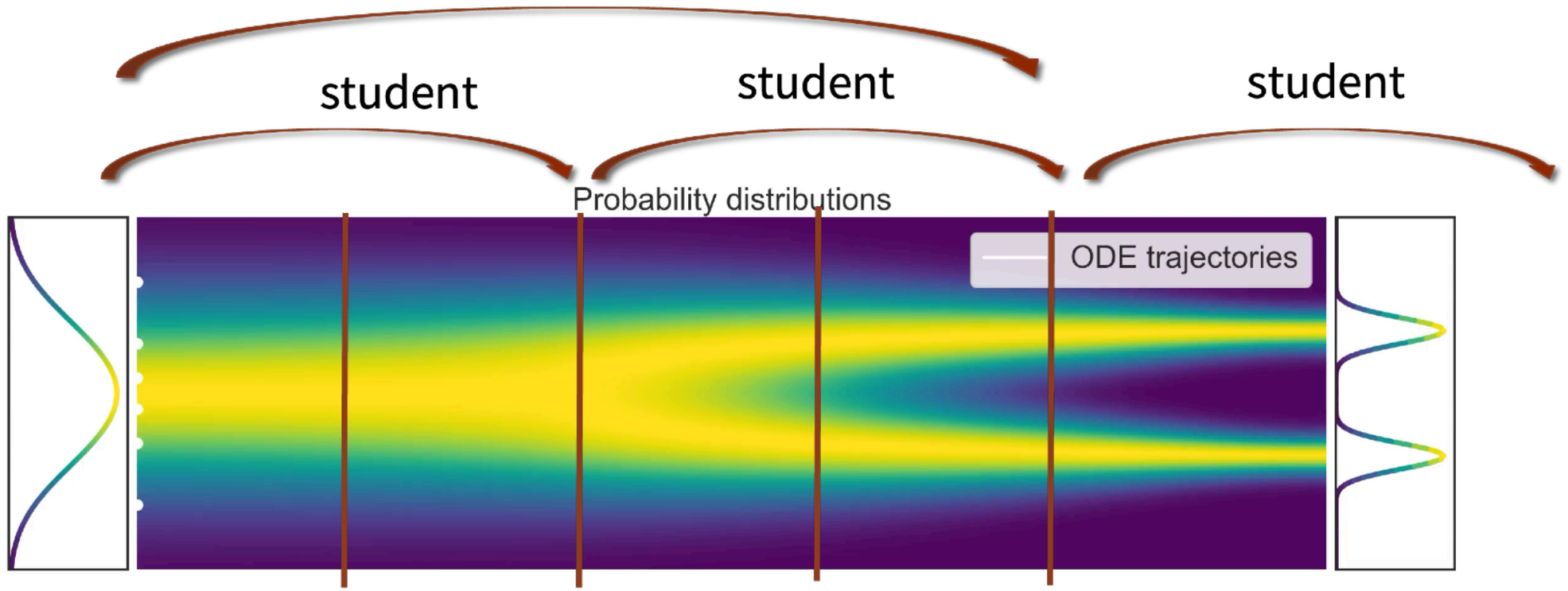
- Numerical methods + ODE formulation to accelerate sampling
- DDIM [Song and Ermon, 2021]:
  - Coarsely discretize the time axis, take big steps
  - Corresponds to exponential integrator (semi-linear ODE) [Lu et al, 2022; Zhang and Chen, 2022]
  - 10x-50x speedups, comparable sample quality

Accelerated sampling



- Numerical methods + ODE formulation to accelerate sampling
- DDIM [Song and Ermon, 2021]:
  - Coarsely discretize the time axis, take big steps
  - Corresponds to exponential integrator (semi-linear ODE) [Lu et al, 2022; Zhang and Chen, 2022]
  - 10x-50x speedups, comparable sample quality

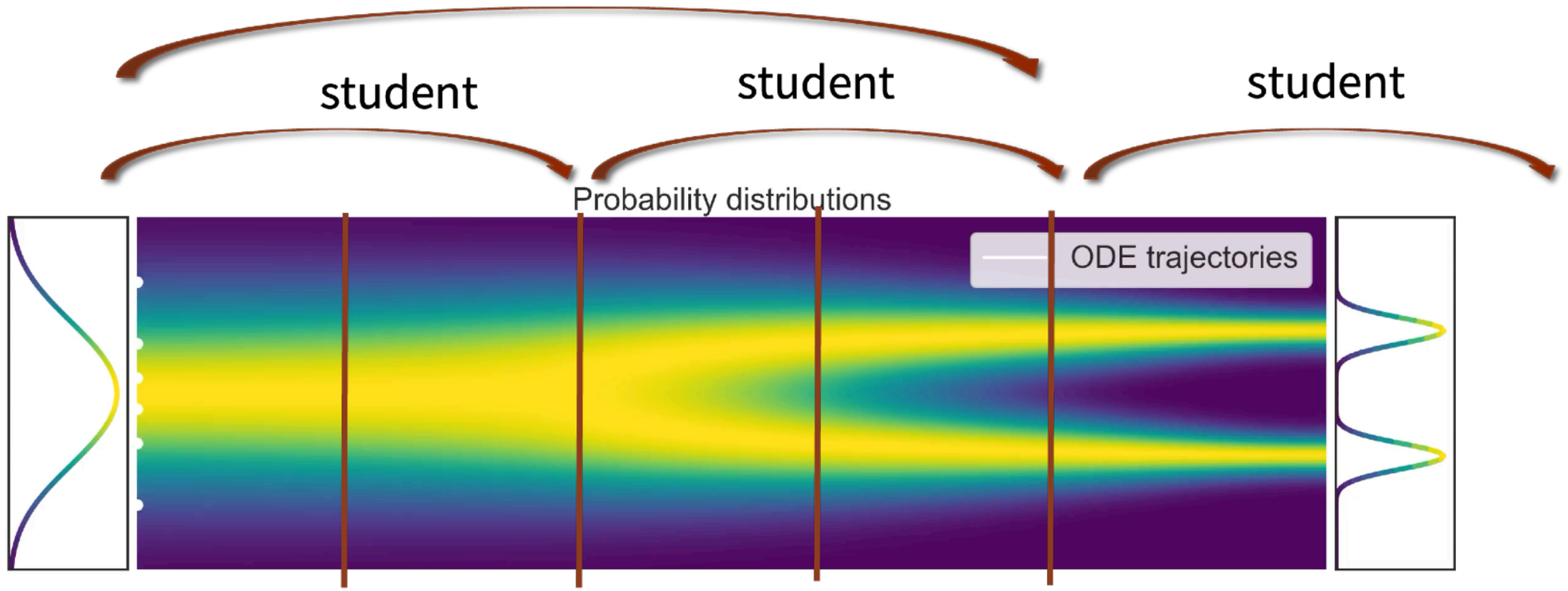
Distillation



Progressive distillation [Salimans, Ho 2022]

- DDIM sampler as a teacher model
- Student model trained to do in 1 step what DDIM achieves in 2 steps
- Applied recursively to drastically reduce the number of steps required

Distillation

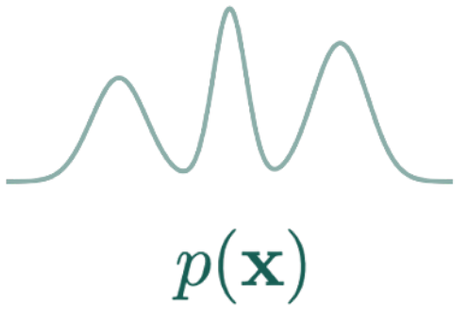


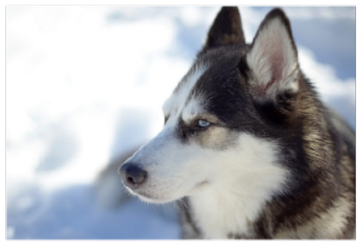
Progressive distillation [Salimans, Ho 2022]

- DDIM sampler as a teacher model
- Student model trained to do in 1 step what DDIM achieves in 2 steps
- Applied recursively to drastically reduce the number of steps required

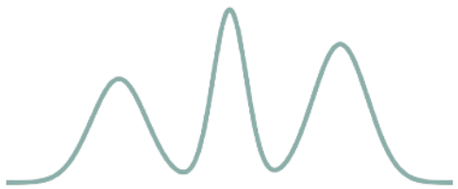
Control the generation process

Control the generation process

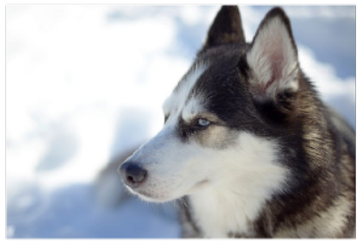




Control the generation process



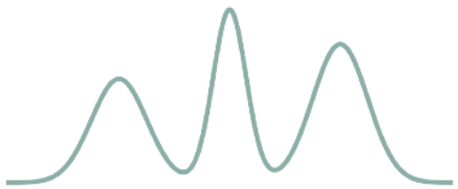
$p(\mathbf{x})$



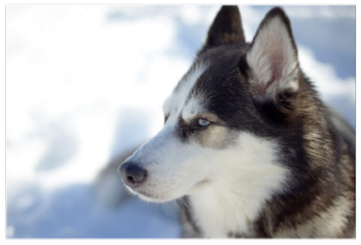
Forward model $p(\mathbf{y} \mid \mathbf{x})$

Control signal $\mathbf{y}$ = “dog”

Control the generation process

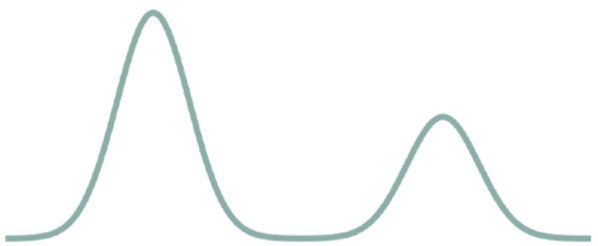


$p(\mathbf{x})$



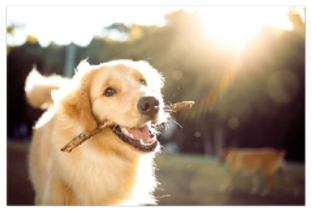
Forward model $p(\mathbf{y} \mid \mathbf{x})$

Control signal $\mathbf{y}$ = “dog”

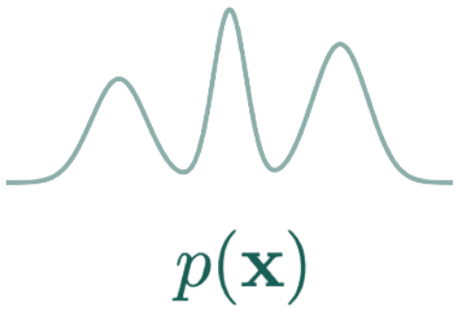


$p(\mathbf{x} \mid \mathbf{y})$

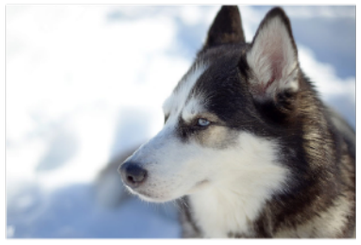
Inverse distribution



Control the generation process

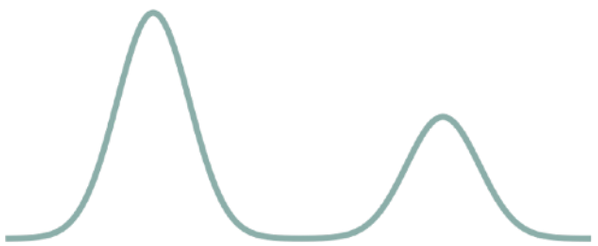


$p(\mathbf{x})$



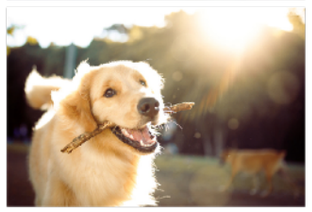
Forward model $p(\mathbf{y} \mid \mathbf{x})$

Control signal $\mathbf{y}$ = “dog”



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

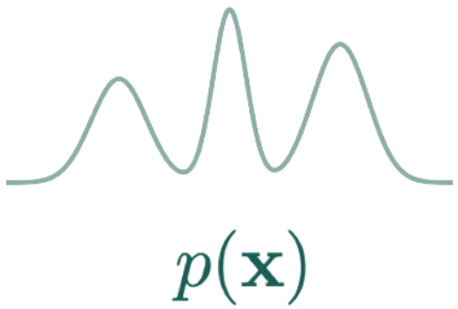
Inverse distribution



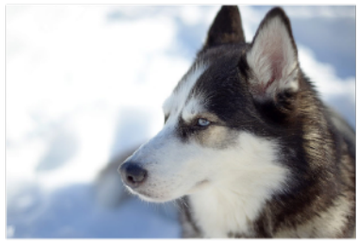
Bayes’ rule:

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

Control the generation process

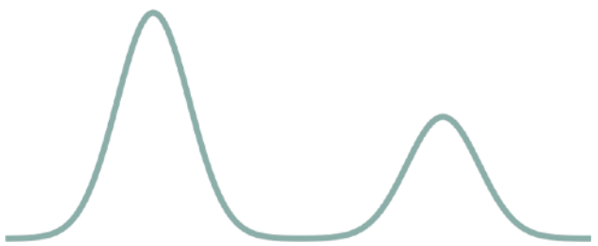


$p(\mathbf{x})$



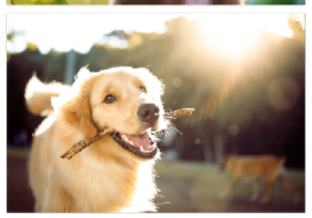
Forward model $p(\mathbf{y} \mid \mathbf{x})$

Control signal $\mathbf{y}$ = “dog”



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

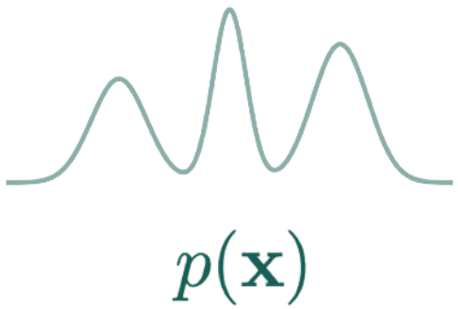
Inverse distribution

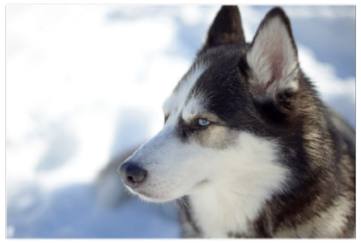


Bayes’ rule:

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

Control the generation process



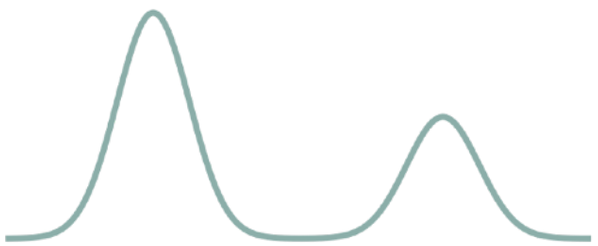


Bayes' rule:

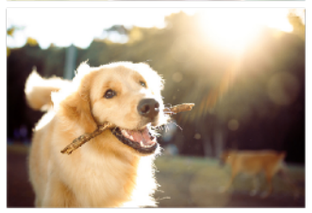
$$p(\mathbf{x} \mid \mathbf{y}) = \frac{p(\mathbf{x})p(\mathbf{y} \mid \mathbf{x})}{p(\mathbf{y})}$$

Forward model $p(\mathbf{y} \mid \mathbf{x})$

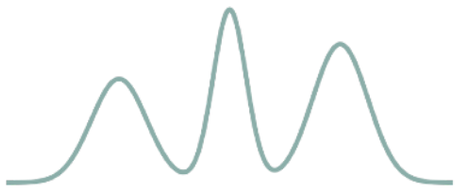
Control signal $\mathbf{y}$ = "dog"



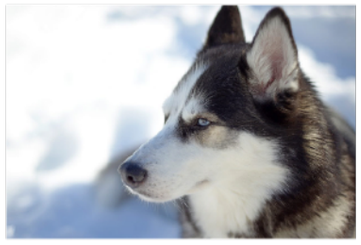
Inverse distribution



Control the generation process

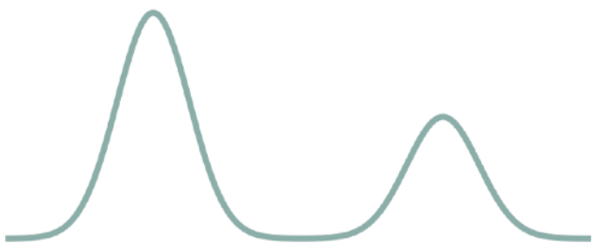


$p(\mathbf{x})$



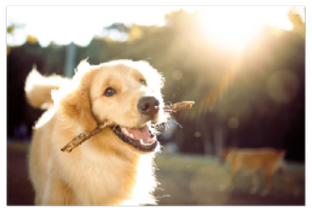
Forward model $p(\mathbf{y} \mid \mathbf{x})$

Control signal $\mathbf{y}$ = “dog”



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

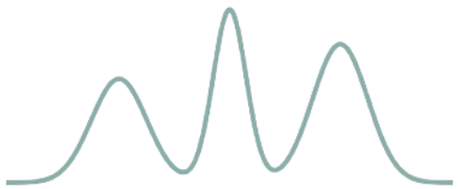
Inverse distribution



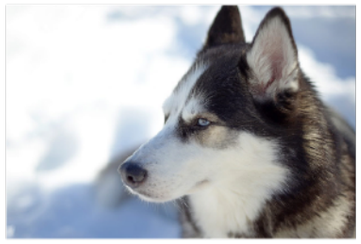
Bayes’ rule:

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

Control the generation process

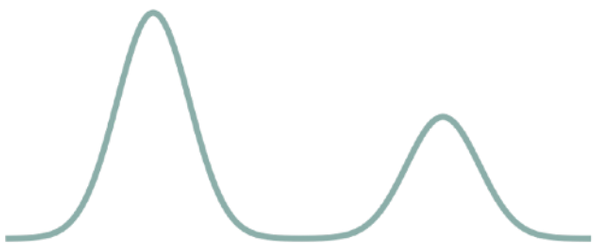


$p(\mathbf{x})$



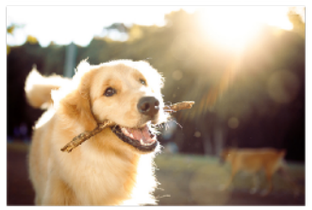
Forward model $p(\mathbf{y} \mid \mathbf{x})$

Control signal $\mathbf{y}$ = “dog”



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

Inverse distribution

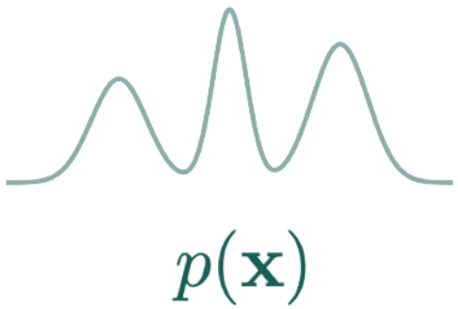


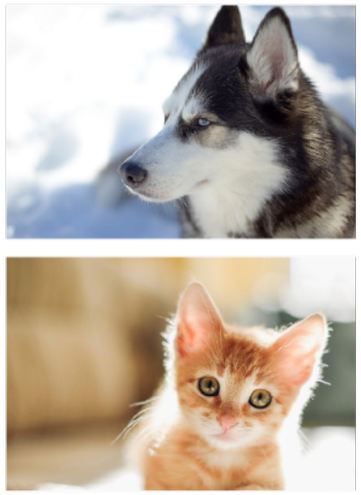
Bayes’ rule:

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

Bayes’ rule for score functions:

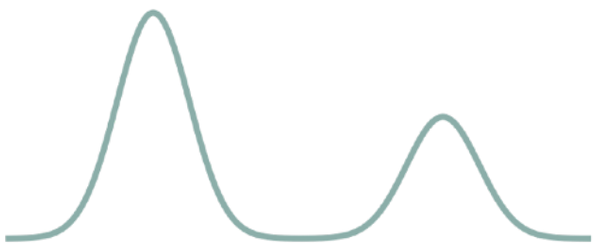
Control the generation process





Forward model $p(\mathbf{y} \mid \mathbf{x})$

Control signal $\mathbf{y}$ = “dog”



Inverse distribution

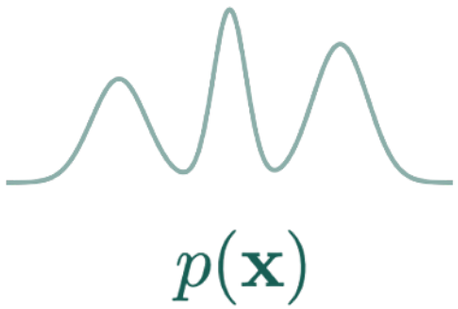
Bayes’ rule:

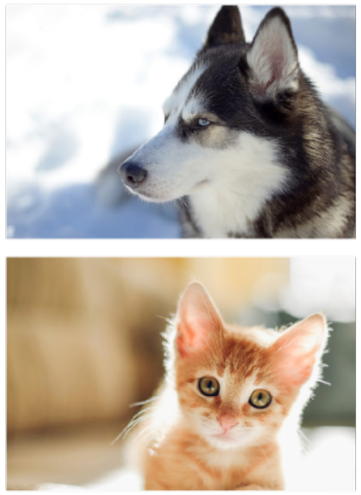
$$p(\mathbf{x} \mid \mathbf{y}) = \frac{p(\mathbf{x})p(\mathbf{y} \mid \mathbf{x})}{p(\mathbf{y})}$$

Bayes’ rule for score functions:

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

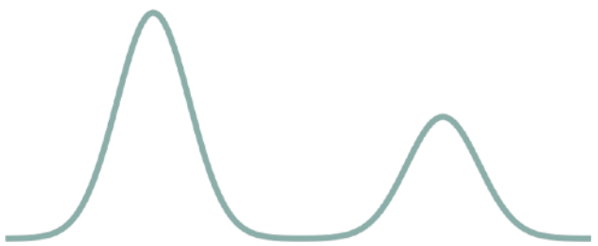
Control the generation process





Forward model $p(\mathbf{y} \mid \mathbf{x})$

Control signal $\mathbf{y}$ = “dog”



Inverse distribution

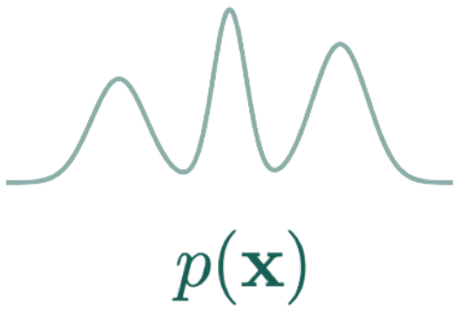
Bayes’ rule:

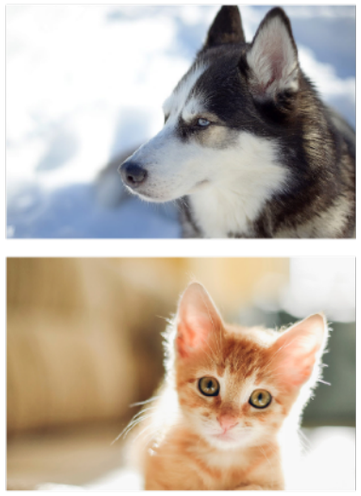
$$p(\mathbf{x} \mid \mathbf{y}) = \frac{p(\mathbf{x})p(\mathbf{y} \mid \mathbf{x})}{p(\mathbf{y})}$$

Bayes’ rule for score functions:

$$\nabla_{\mathbf{x}} \log p(\mathbf{x} \mid \mathbf{y}) = \nabla_{\mathbf{x}} \log p(\mathbf{x}) + \nabla_{\mathbf{x}} \log p(\mathbf{y} \mid \mathbf{x}) - \nabla_{\mathbf{x}} \log p(\mathbf{y}) \quad \mathbf{0}$$

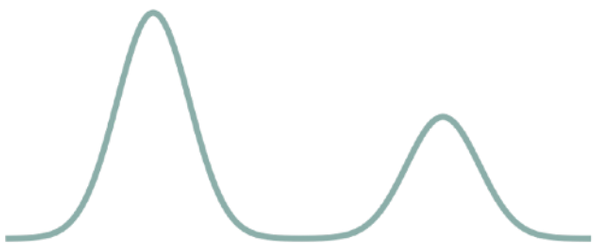
Control the generation process





Forward model $p(\mathbf{y} \mid \mathbf{x})$

Control signal $\mathbf{y}$ = “dog”



Inverse distribution

Bayes’ rule:

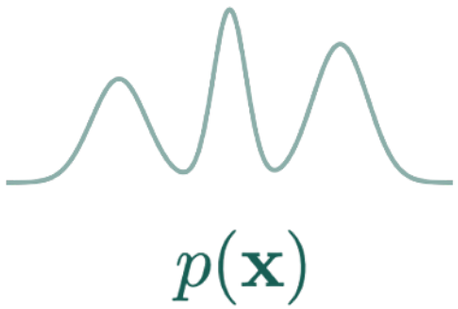
$$p(\mathbf{x} \mid \mathbf{y}) = \frac{p(\mathbf{x})p(\mathbf{y} \mid \mathbf{x})}{p(\mathbf{y})}$$

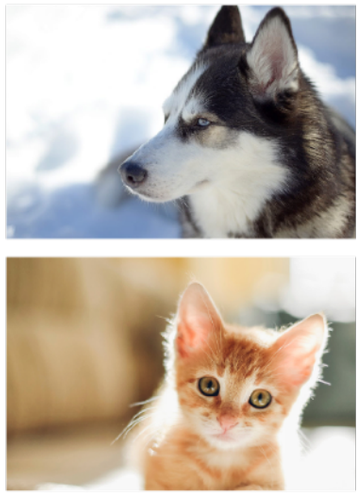
Bayes’ rule for score functions:

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

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

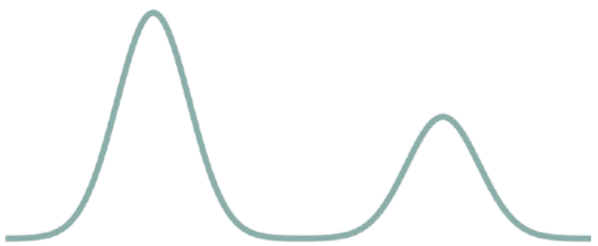
Control the generation process





Forward model $p(\mathbf{y} \mid \mathbf{x})$

Control signal $\mathbf{y}$ = “dog”



Inverse distribution

Bayes’ rule:

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

Bayes’ rule for score functions:

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

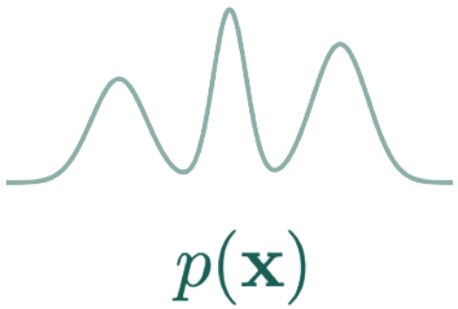
$\mathbf{0}$

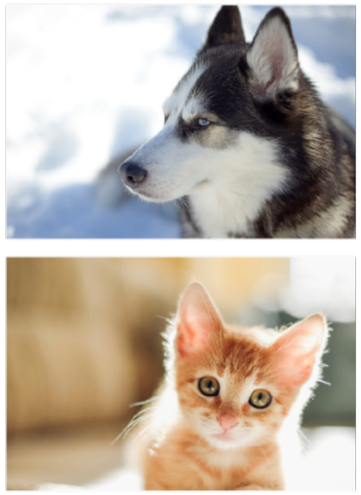
$$= \nabla_{\mathbf{x}} \log p(\mathbf{x}) + \nabla_{\mathbf{x}} \log p(\mathbf{y} \mid \mathbf{x})$$

Unconditional score

$$\approx \boldsymbol{s_\theta}(\mathbf{x})$$

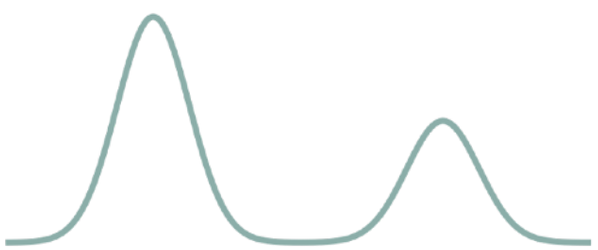
Control the generation process





Forward model $p(\mathbf{y} \mid \mathbf{x})$

Control signal $\mathbf{y}$ = “dog”



Inverse distribution

Bayes’ rule:

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

Bayes’ rule for score functions:

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

$\mathbf{0}$

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

Unconditional score $\approx s_\theta(\mathbf{x})$

Forward model

Why you should work on it?

- Mathematically beautiful, empirically works
- Fits in your GPUS; still can train on ImageNet
- We’re stuck with VAE for like 10 years by now
- Diffusion policies is revolutionizing offline RL; multimodal, expressive policies
- Hybrid models (diffusion + AR) is the future probably
- So many questions left to answer as the field is less matured and more diverse (e.g. ai4science); good for phd timeline
- Some problems are decade old e.g. sampling, literally no worries of getting scoped

Some Problems to think about … diffusion

Diffusion Representation Learning (e,g, classification, segmentation) SecretlyClassifier'23 EmerDiff'24

What's a good latent space e.g. equivariance, object-centricness? EQ-VAE'25 REPA-E'25 Diffusability'25

What's a good forward process e.g. how to introduce noising best (heavy tail issue in Gaussian diffusion, noise warping)? Warpyournoise'25 Critically-Damped Langevin Diffusion'21 Rectified Flows'22

How to reconcile the representation and generation (e.g. reducing high variance in denoiser, ssl, repa)? REPA'24 Diffuse Disperse'24

How to get rid of two stage (the vae component): make one big beautiful diffusion end-to-end?

Sequence length generalization (autoregression) and refinement (diffusion) — combining both into one for test-time scaling BlockDiffusion'25 DiffusionForcing'25 KV Caching

The trilemma of continuous generative models

Few-step or one-step diffusion (Flow matching) MenFlow'25 FlowMaps'25

To classifier free guidance or not to

Reward modeling for diffusion e.g. adjoining matching

Diffusion policy (long-horizon actions smoothing) DiffusionPolicy'25

Diffusion model for almost everything text, action, video, genome, etc

Diffusion duality (gaussian diffusion annealed to discrete diffusion, CTMC'25)

Discrete flow matching