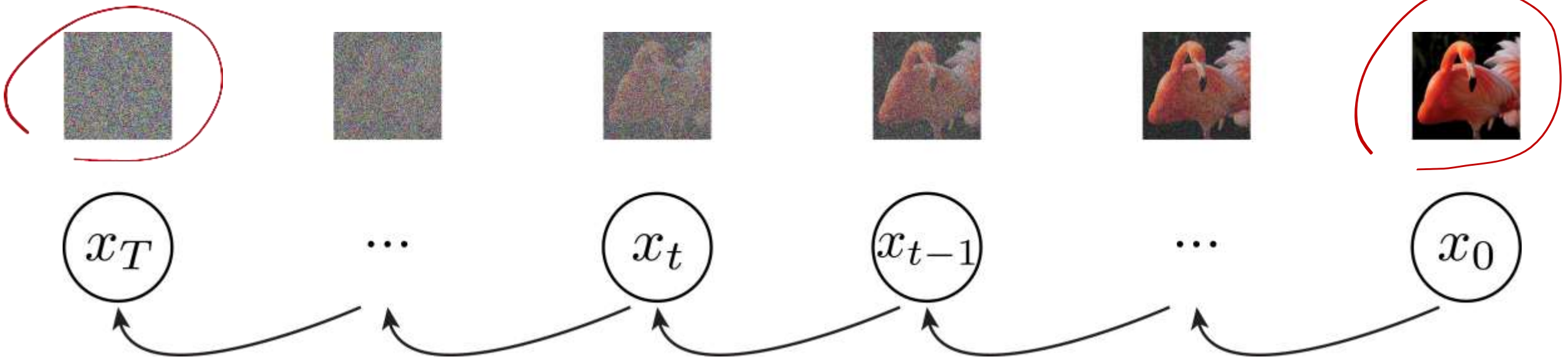
Recap.

... in a nutshell

noise

data

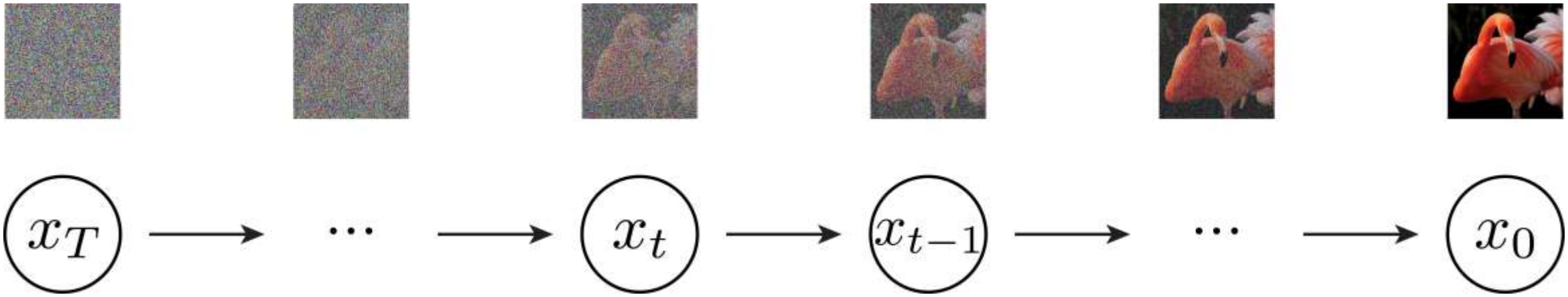


Forward process: add noise

... in a nutshell

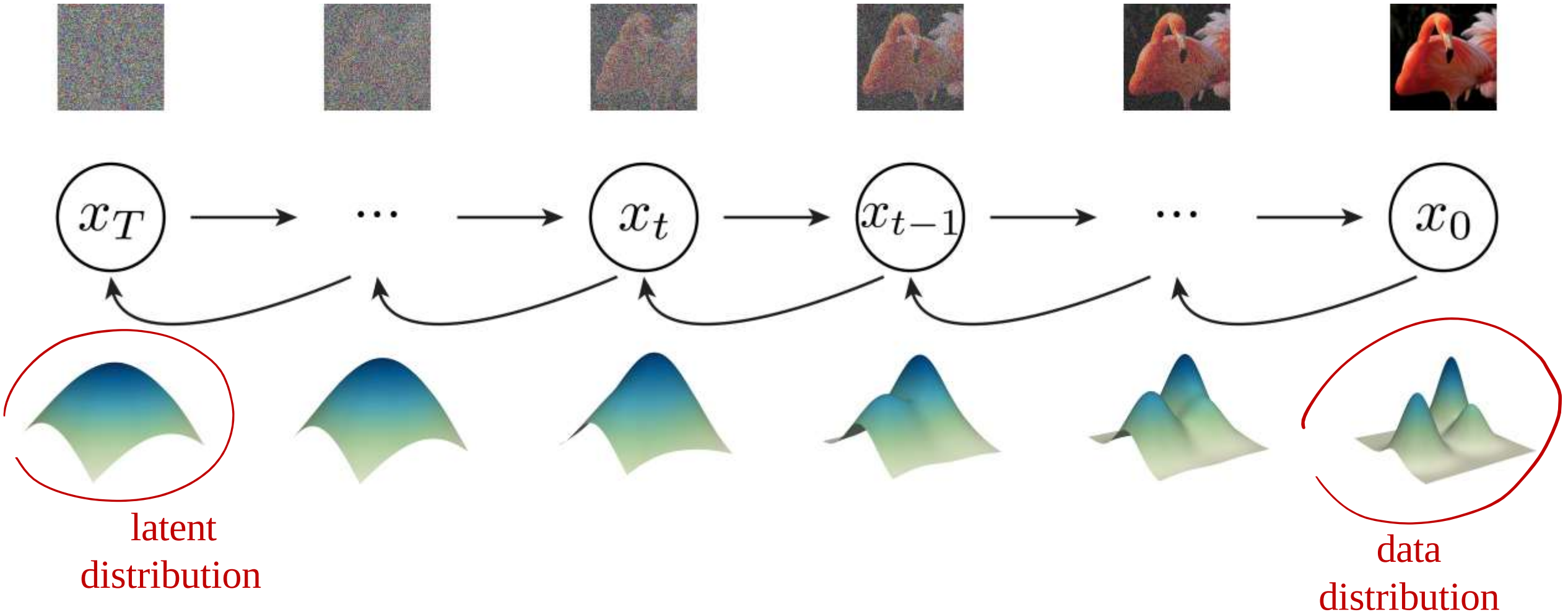
noise

data

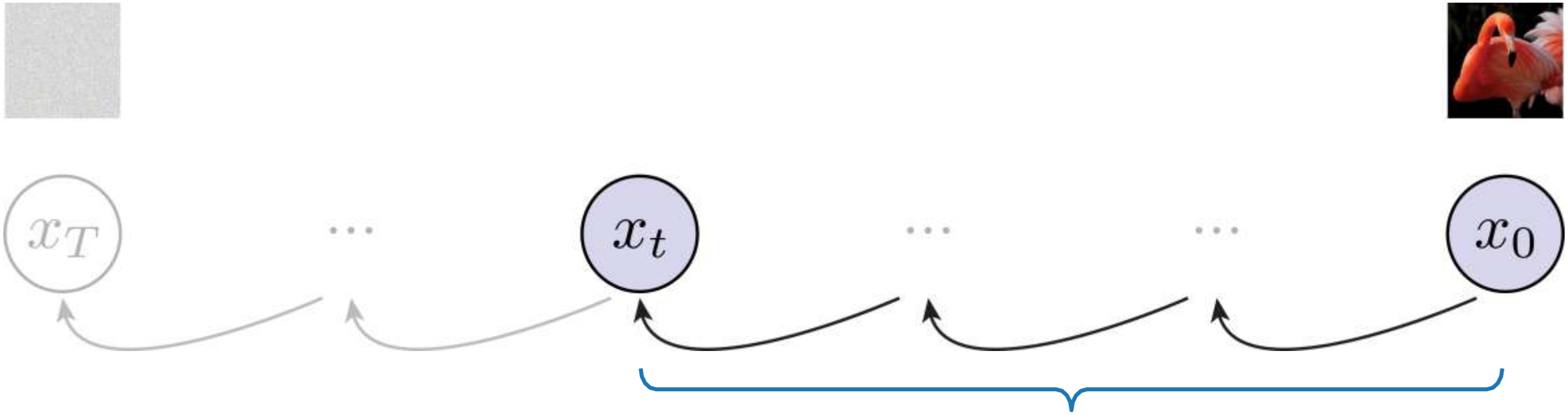


Reverse process: denoise

What is noise?



Forward Process



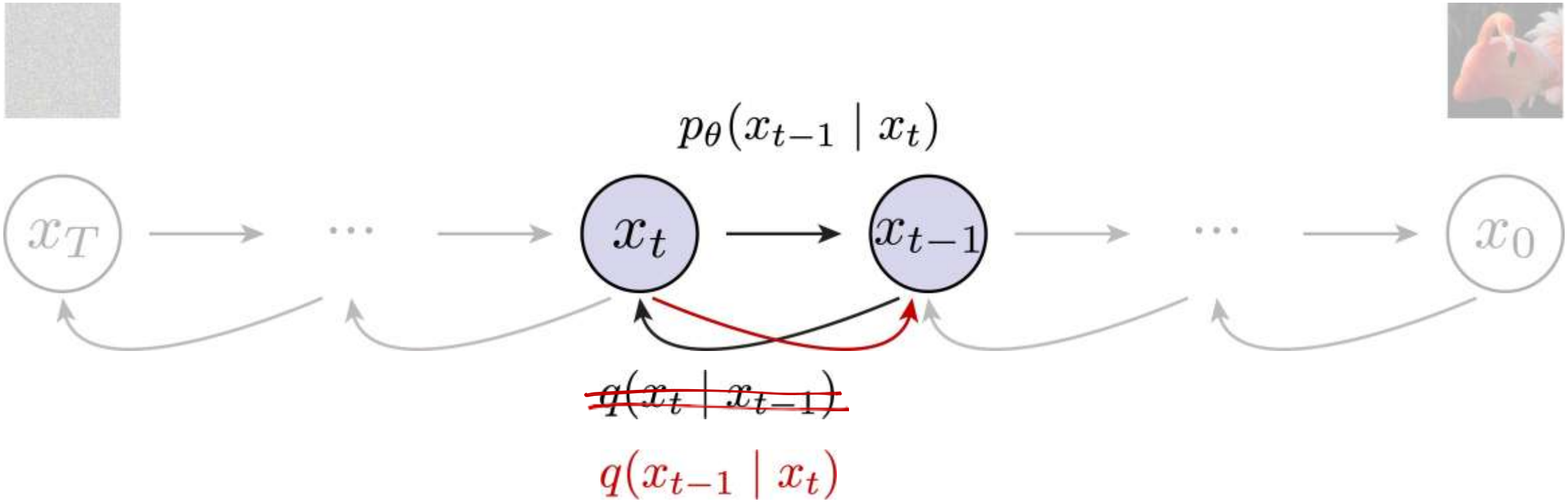
- sampling without simulation
- x_t from x_0 in closed form

$$q(x_t \mid x_0) = \mathcal{N}(x_t \mid \sqrt{\bar{\alpha}_t} x_0, (1 - \bar{\alpha}_t)\mathbf{I})$$

coefficients given by β

$$\alpha_t := 1 - \beta_t$$
$$\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$$

Reverse Process



- our target
- but unknown

Reverse Process

$$q(x_t|x_0) = \sqrt{\bar{\alpha}_t}\, + \sqrt{1-\bar{\alpha}_t}$$

$$q(x_{t-1}|x_0) = \sqrt{\bar{\alpha}_{t-1}}\, + \sqrt{1-\bar{\alpha}_{t-1}}$$

$$q(x_t|x_{t-1}) = \sqrt{1-\beta_t}\, + \sqrt{\beta_t}$$

$$q(x_{t-1}|x_t, x_0) = \frac{q(x_{t-1}, x_t, x_0)}{q(x_t, x_0)} = \frac{q(x_t|x_{t-1})q(x_{t-1}|x_0)q(x_0)}{q(x_t|x_0)q(x_0)} = \frac{q(x_t|x_{t-1})q(x_{t-1}|x_0)}{q(x_t|x_0)}$$

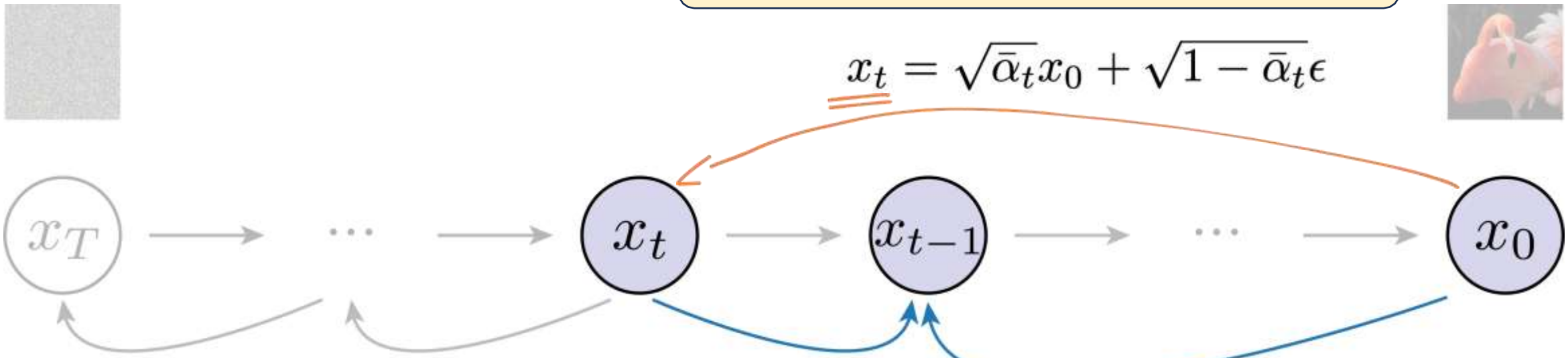
- known
- Gaussian

- known
- Gaussian

- known
- Gaussian

Reverse Process

tl; dr:
- outcome of the dependency graph
- some linear combinations



$$q(x_{t-1} \mid x_t, x_0) = \mathcal{N}(x_{t-1} \mid \tilde{\mu}(x_t, x_0), \tilde{\beta}_t \mathbf{I})$$

$$\tilde{\mu}_t(x_t, x_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}x_t$$

mean

$$= \frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right)$$

"noise"

$$\tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

var

x_0

Sample x_t

x_t

t

Denoise → ? ⇢ $\frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon\right)$

To predict

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon$$

$$x_t - \sqrt{1-\bar{\alpha}_t}\varepsilon = \sqrt{\bar{\alpha}_t}x_0$$

$$\frac{x_t - \sqrt{1-\bar{\alpha}_t}\varepsilon}{\sqrt{\bar{\alpha}_t}} = x_0$$

Algorithm 2 Sampling

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

Diffusion Models

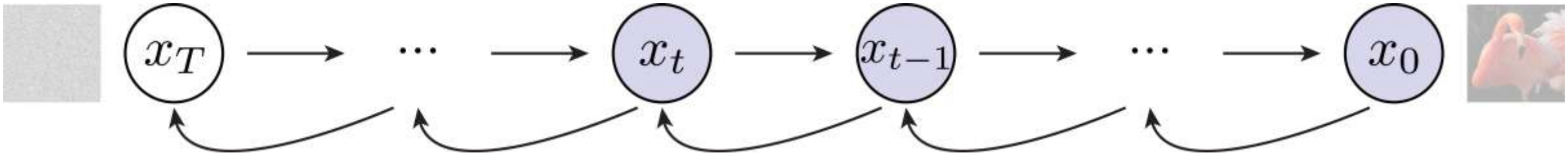
- Forward process
- add noise to data

- Reverse process
- learn to denoise

- Training objective
- from Hierarchical VAE to L2 loss

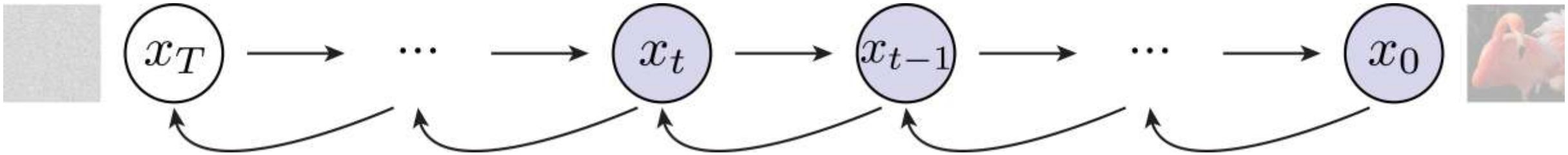
- Noise Conditional Network
- represent a distribution

Training Objective



$$
\begin{aligned}
\mathcal{L}_{\text{VLB}} &:= \mathcal{L}_T + \mathcal{L}_{T-1} + \ldots + \mathcal{L}_0 \\
\mathcal{L}_T &:= D_{\text{KL}}\Big(q(x_T \mid x_0) \,\|\, p_\theta(x_T)\Big) \\
\mathcal{L}_{t-1} &:= D_{\text{KL}}\Big(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t)\Big) \\
\mathcal{L}_0 &:= -\log p_\theta(x_0 \mid x_1)
\end{aligned}
$$

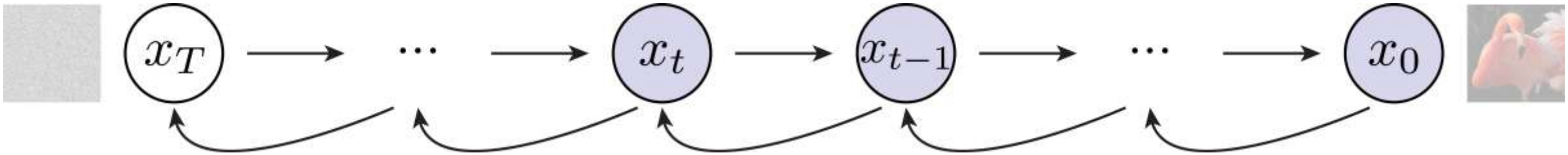
Training Objective



- variational lower bound
- like ELBO

$$
\begin{aligned}
\mathcal{L}_{\text{VLB}} &:= \mathcal{L}_T + \mathcal{L}_{T-1} + \ldots + \mathcal{L}_0 \\
\mathcal{L}_T &:= D_{\text{KL}}\Big(q(x_T \mid x_0) \,\|\, p_\theta(x_T)\Big) \\
\mathcal{L}_{t-1} &:= D_{\text{KL}}\Big(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t)\Big) \\
\mathcal{L}_0 &:= -\log p_\theta(x_0 \mid x_1)
\end{aligned}
$$

Training Objective – toVAE



- variational lower bound
- like ELBO

$$\mathcal{L}_{\text{VLB}} := \mathcal{L}_T + \mathcal{L}_{T-1} + \ldots + \mathcal{L}_0$$

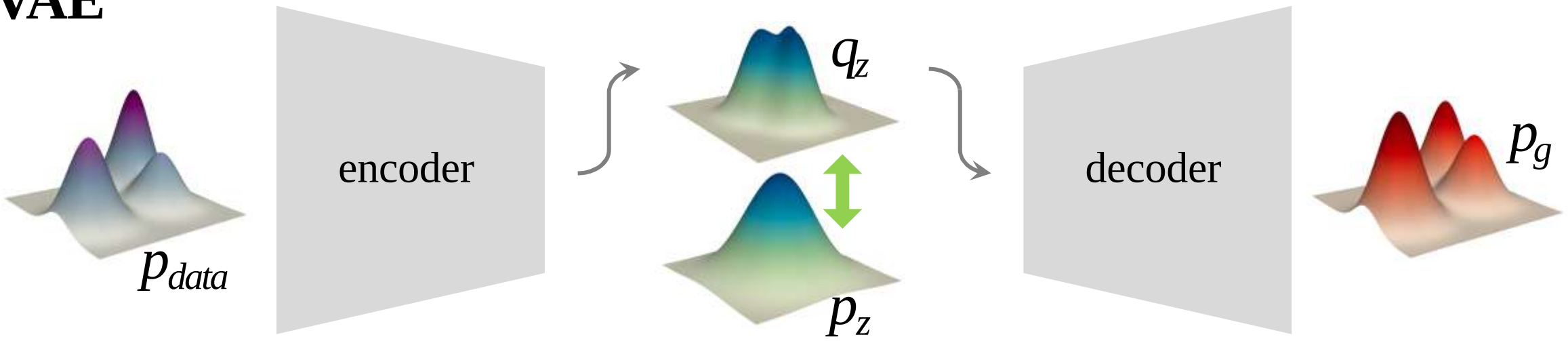
$$\mathcal{L}_T := D_{\text{KL}}\left(q(x_T \mid x_0) \,\|\, p_\theta(x_T)\right)$$

it's ELBO if one step

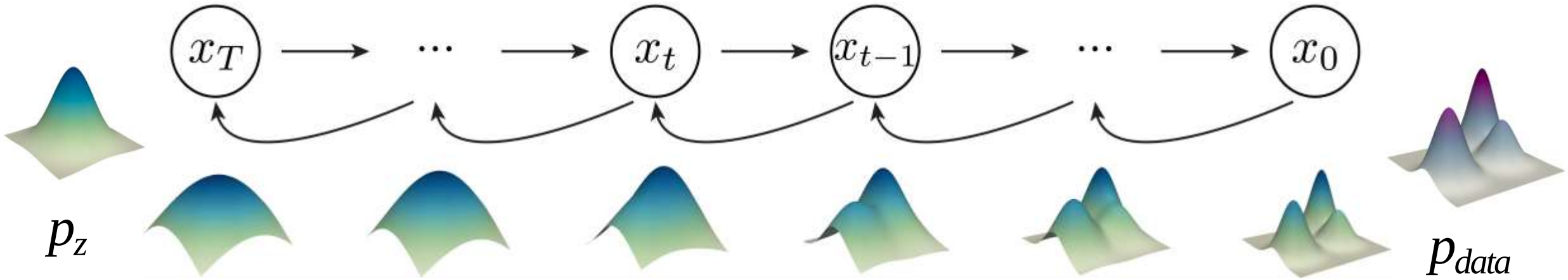
$$\mathcal{L}_{t-1} := D_{\text{KL}}\left(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t)\right)$$

$$\mathcal{L}_0 := -\log p_\theta(x_0 \mid x_1)$$

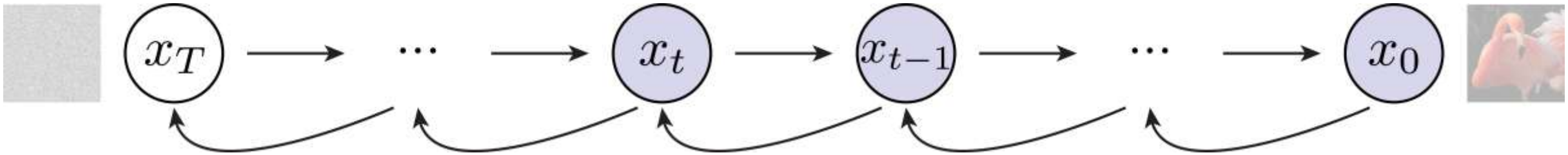
VAE



DM/Flow



Training Objective



$$\mathcal{L}_{\text{VLB}} := \mathcal{L}_T + \mathcal{L}_{T-1} + \ldots + \mathcal{L}_0$$

no parameter, unlike VAE's q_φ

$$\mathcal{L}_T := D_{\text{KL}}\left(q(x_T \mid x_0) \,\|\, p_\theta(x_T)\right)$$

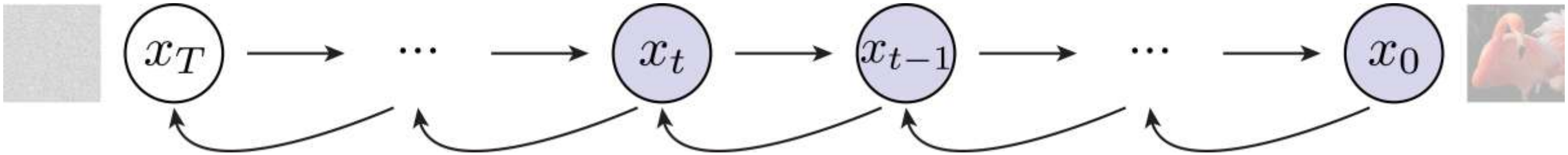
constant

Gaussian

$$\mathcal{L}_{t-1} := D_{\text{KL}}\left(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t)\right)$$

$$\mathcal{L}_0 := -\log p_\theta(x_0 \mid x_1)$$

Training Objective



$$\mathcal{L}_{\text{VLB}} := \mathcal{L}_T + \mathcal{L}_{T-1} + \ldots + \mathcal{L}_0$$

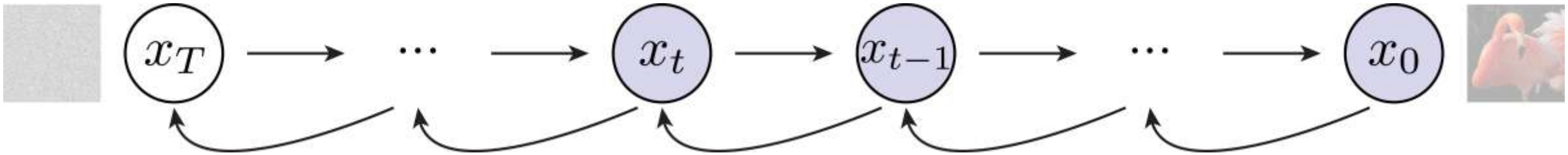
$$\mathcal{L}_T := D_{\text{KL}}\Big(q(x_T \mid x_0) \,\|\, p_\theta(x_T)\Big)$$

$$\mathcal{L}_{t-1} := D_{\text{KL}}\Big(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t)\Big)$$

reconstruction loss, like VAE

$$\mathcal{L}_0 := -\log p_\theta(x_0 \mid x_1)$$

Training Objective



$$\mathcal{L}_{\text{VLB}} := \mathcal{L}_T + \mathcal{L}_{T-1} + \ldots + \mathcal{L}_0$$

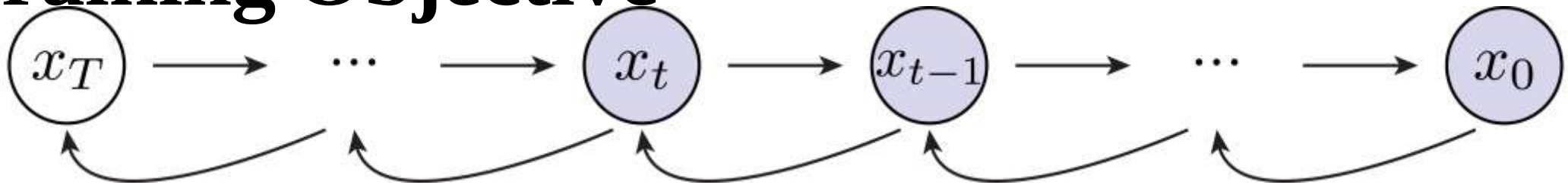
$$\mathcal{L}_T := D_{\text{KL}}\Big(q(x_T \mid x_0) \,\|\, p_\theta(x_T)\Big)$$

L2 loss on noise

$$\mathcal{L}_{t-1} := D_{\text{KL}}\Big(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t)\Big)$$

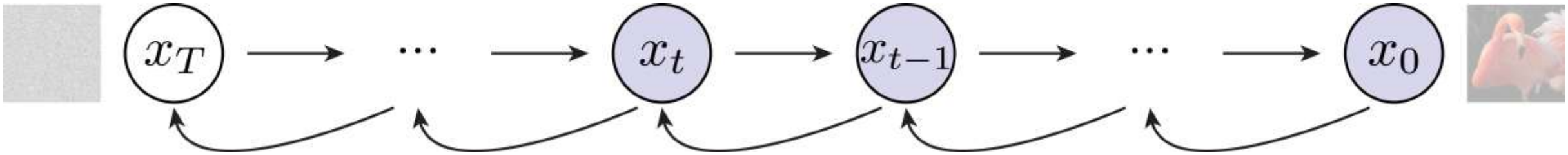
$$\mathcal{L}_0 := -\log p_\theta(x_0 \mid x_1)$$

Training Objective



$$\mathcal{L} = \mathbb{E}_{x_0,t,\epsilon}\left[w_t \|\epsilon - \epsilon_\theta(x_t, t)\|^2\right]$$

Training Objective



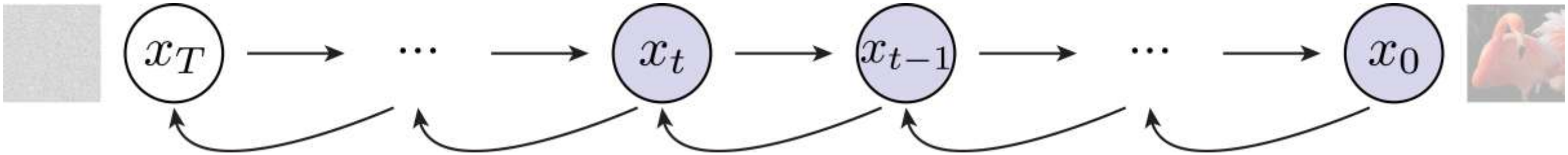
$$\mathcal{L} = \mathbb{E}_{x_0, t, \epsilon}\left[w_t \|\epsilon - \epsilon_\theta(x_t, t)\|^2\right]$$

over p_{data}

over $[1, T]$

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

Training Objective



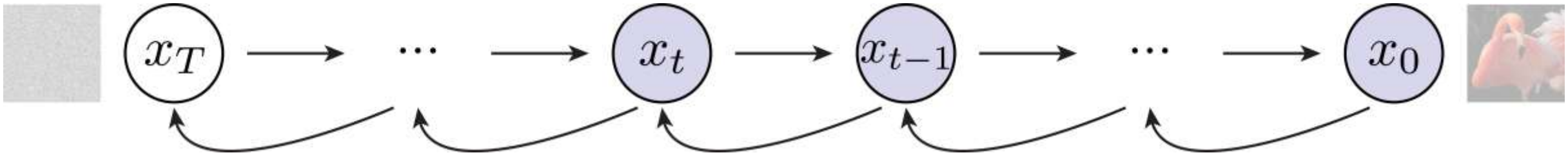
$$\mathcal{L} = \mathbb{E}_{x_0,t,\epsilon}\left[w_t \|\epsilon - \epsilon_\theta(x_t, t)\|^2\right]$$

set as 1 **(critical)**

Objective	IS	FID
L, learned diagonal $\boldsymbol{\Sigma}$	–	–
L, fixed isotropic $\boldsymbol{\Sigma}$	7.67 ± 0.13	13.51
$\|\tilde{\boldsymbol{\epsilon}} - \boldsymbol{\epsilon}_\theta\|^2$ (L_{simple})	$\mathbf{9.46 \pm 0.11}$	**3.17**

[Ho et al. 2020]; see more in [Salimans & Ho, 2022]

Training Objective



$$\mathcal{L} = \mathbb{E}_{x_0,t,\epsilon}\left[w_t \|\epsilon - \epsilon_\theta(x_t, t)\|^2\right]$$

network to predict noise

conditioned on noise level **(critical)**

Diffusion Models

- Forward process
- add noise to data

- Reverse process
- learn to denoise

- Training objective
- from Hierarchical VAE to L2 loss

- Noise Conditional Network
- represent a distribution

Noise Conditional Network

- Diffusion models decompose a distribution into **many** simpler ones.
- We need the same # networks to fit **all** of them.
- We can **combine** all into one “powerful” network.
- This network is conditioned on noise level *t*.

- **Noise Conditional Network** [Song & Ermon 2019]: things made work

*It is called Noise Conditional Score Network (NCSN) in [Song & Ermon 2019] in the context of score matching.

Noise Conditional Network

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

- network input: x_t
- network output: μ and σ of a distribution

- parametrize μ by: $\epsilon_\theta(x_t, t)$

noisy image:
- condition
- network input

noise level:
- condition
- network input

Noise Conditional Network

x_t

network input

t

also network input

$\epsilon_\theta(x_t, t)$

Unet, ViT, ...
(behaves like Denoising AE)

ϵ

output: same size

Diffusion algorithm annotated:

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

x_t

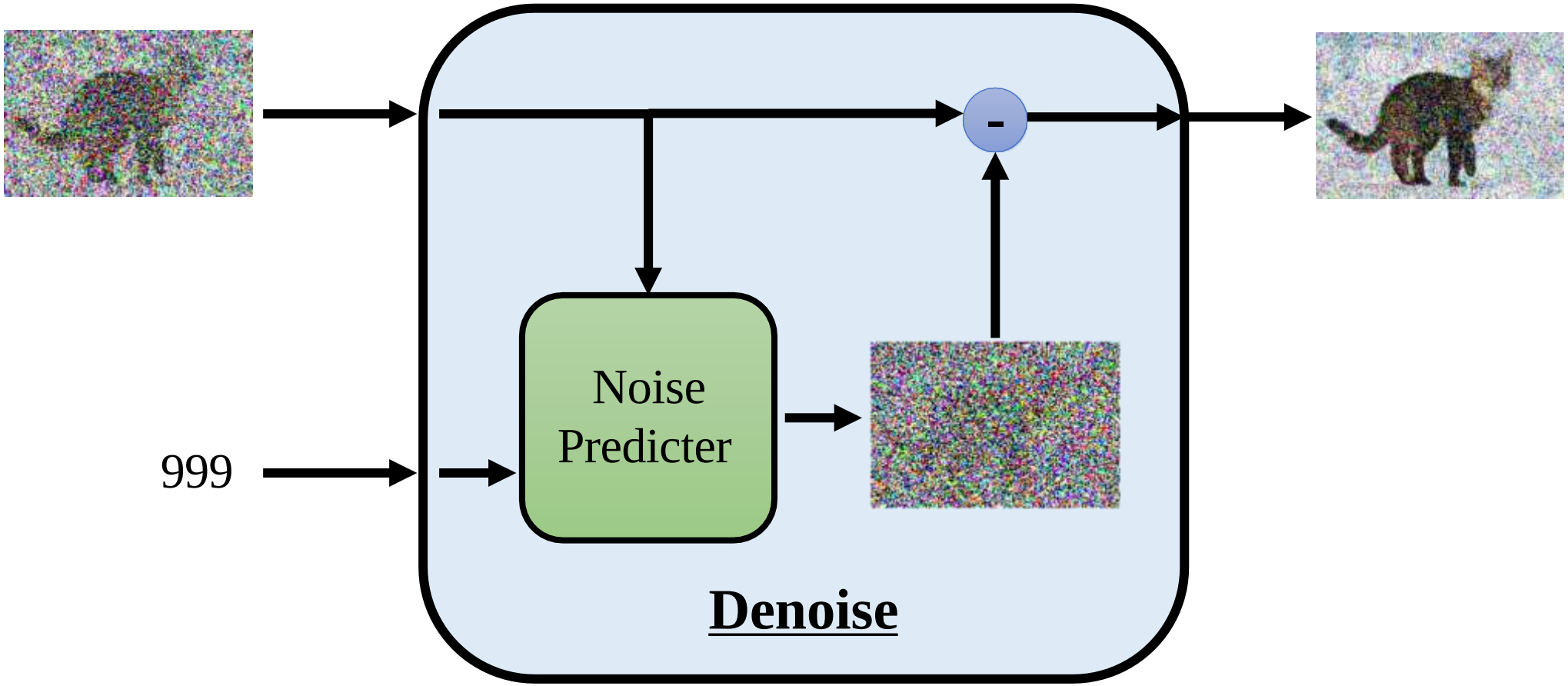
Algorithm 2 Sampling

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** $\mathbf{x}_0$

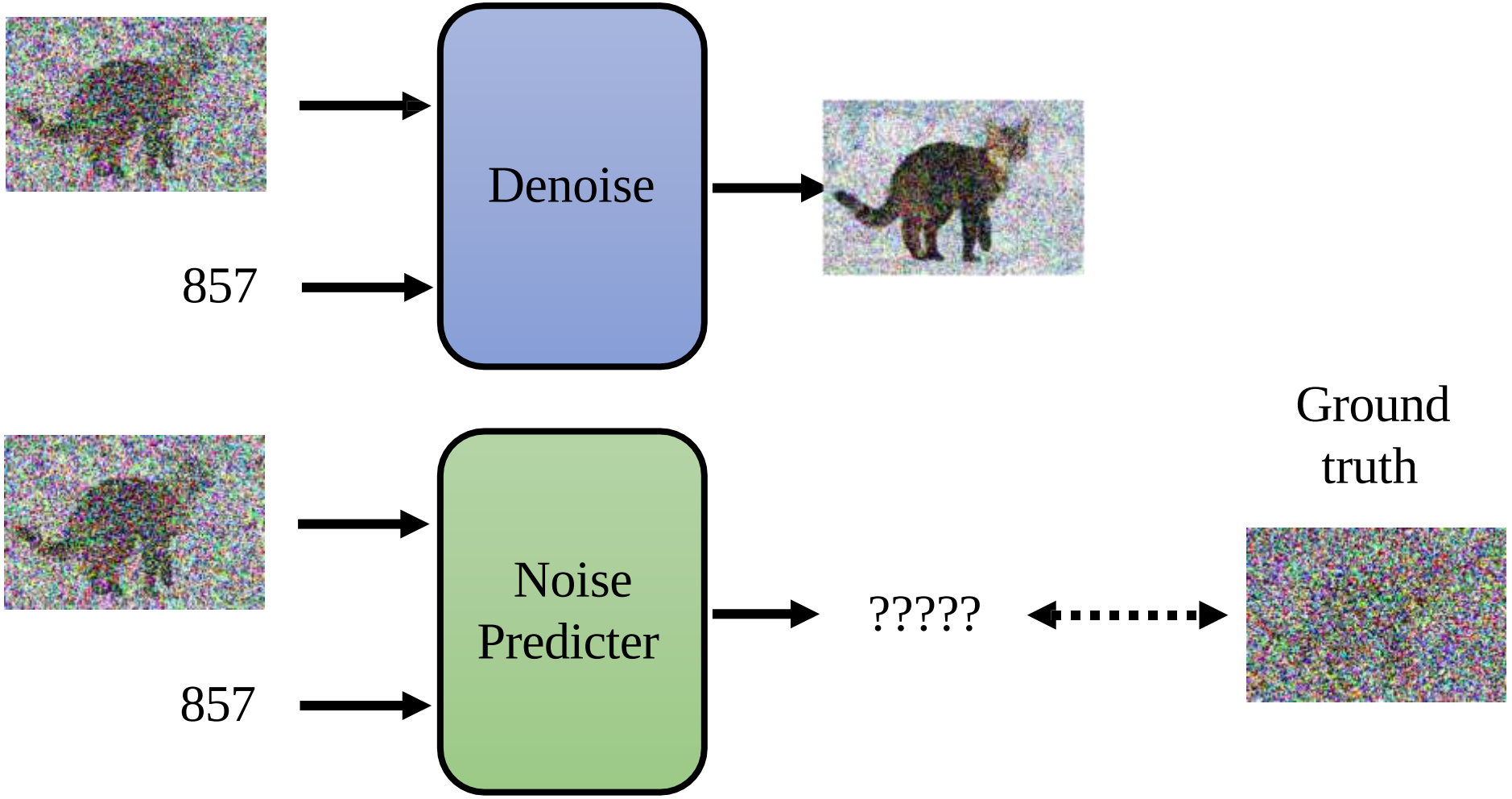
estimated μ

sampling from estimated distribution

Noise Conditional Network



Noise Conditional Network



Diffusion algorithm annotated:

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** $\mathbf{x}_0$

tl; dr: noising and denoising

- Turns out to be extremely simple
- Being “simple and effective” moves the needle

Example: Unconditional Generation on CIFAR-10

noise
x_T

generated
x_0

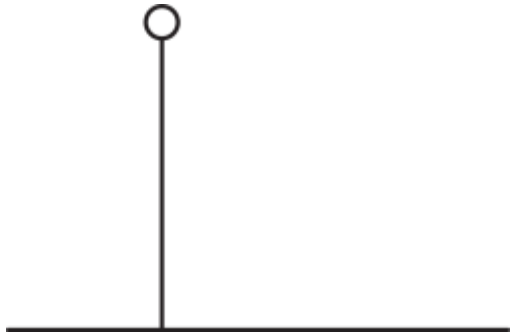
[Ho et al, 2020]

Example: shared intermediate latents

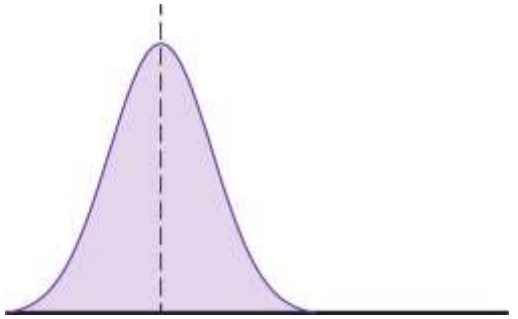
[Ho et al, 2020]

Recap.

- Adding Gaussian noise ⇔ sampling $x \sim \mathcal{N}(x \mid x_0, \sigma)$

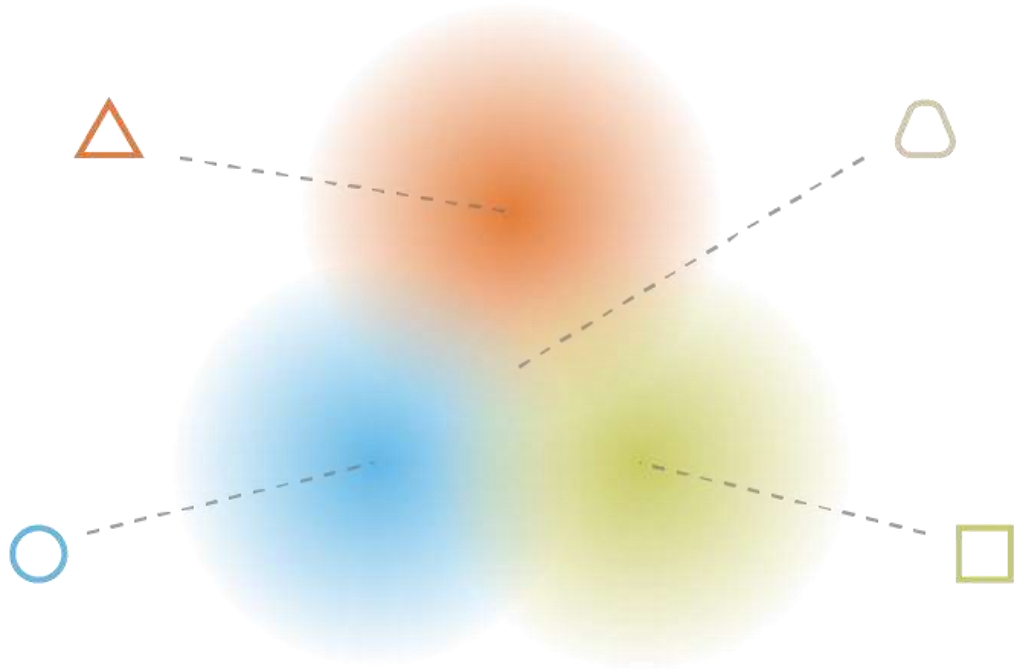


$p(x) = \delta(x - x_0)$



$p(x) = \mathcal{N}(x \mid x_0, \sigma)$

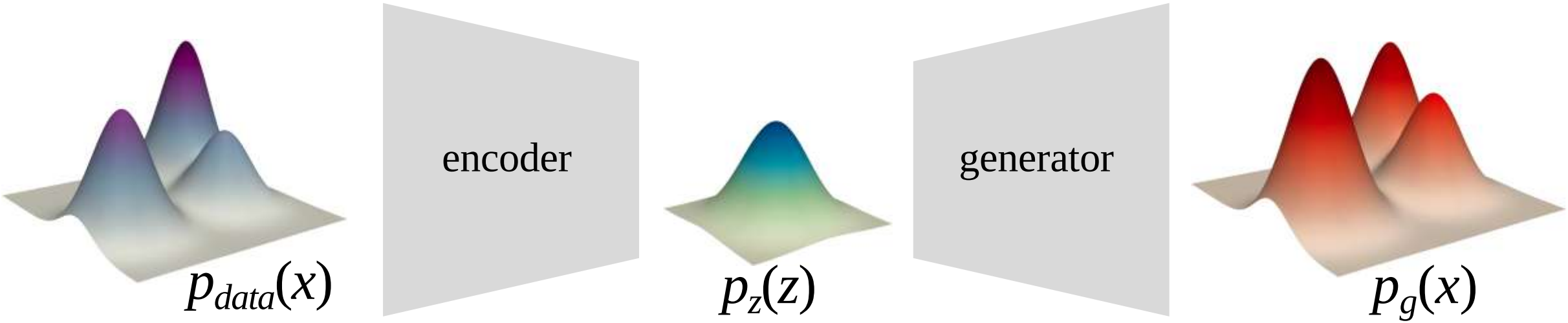
Problems



Recap: Variational Autoencoder (VAE)

Autoencoding distributions:

"Encoding" data distribution p_{data} into latent distribution p_z



"Reconstruction" loss

Summary

Forward process

- add noise to data

Reverse process

- learn to denoise

Training objective

- from Hierarchical VAE to L2 loss

Noise Conditional Network

- represent distributions

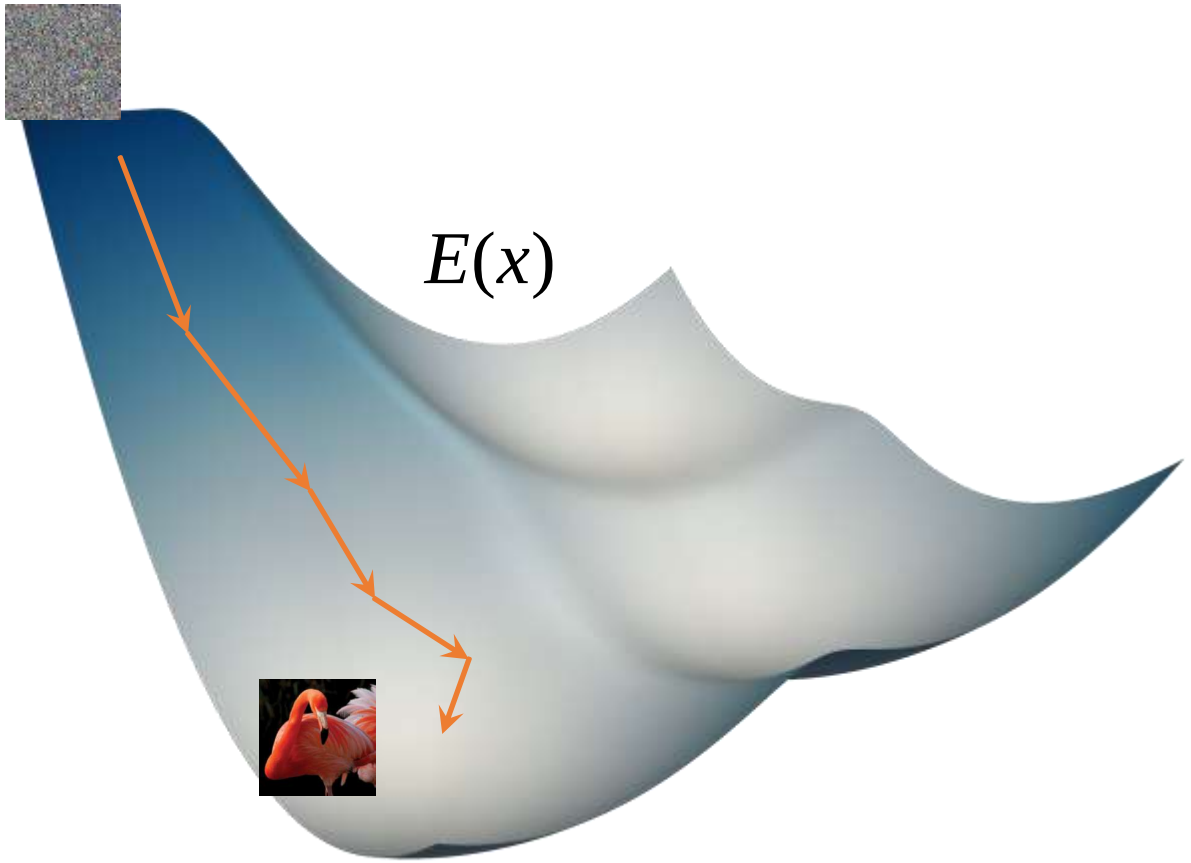
Energy-based Models and Score Matching

Diffusion and Score Matching

- Diffusion Models are closely related to **Score Matching**.
- Score Matching is one solution to **Energy-based Models**.
- Energy-based Models:
  - can be probabilistic or non-probabilistic
  - can be generative or discriminative
- Many useful concepts in diffusion co-evolved w/ score matching
  - Annealed importance sampling [Neal 1998]
  - Denoising score matching [Vincent 2011]
  - Noise Conditional Score Network [Song & Ermon 2019]

Energy-based Models

- Define a scalar function, called “energy”.
- At inference time, find x that minimizes energy

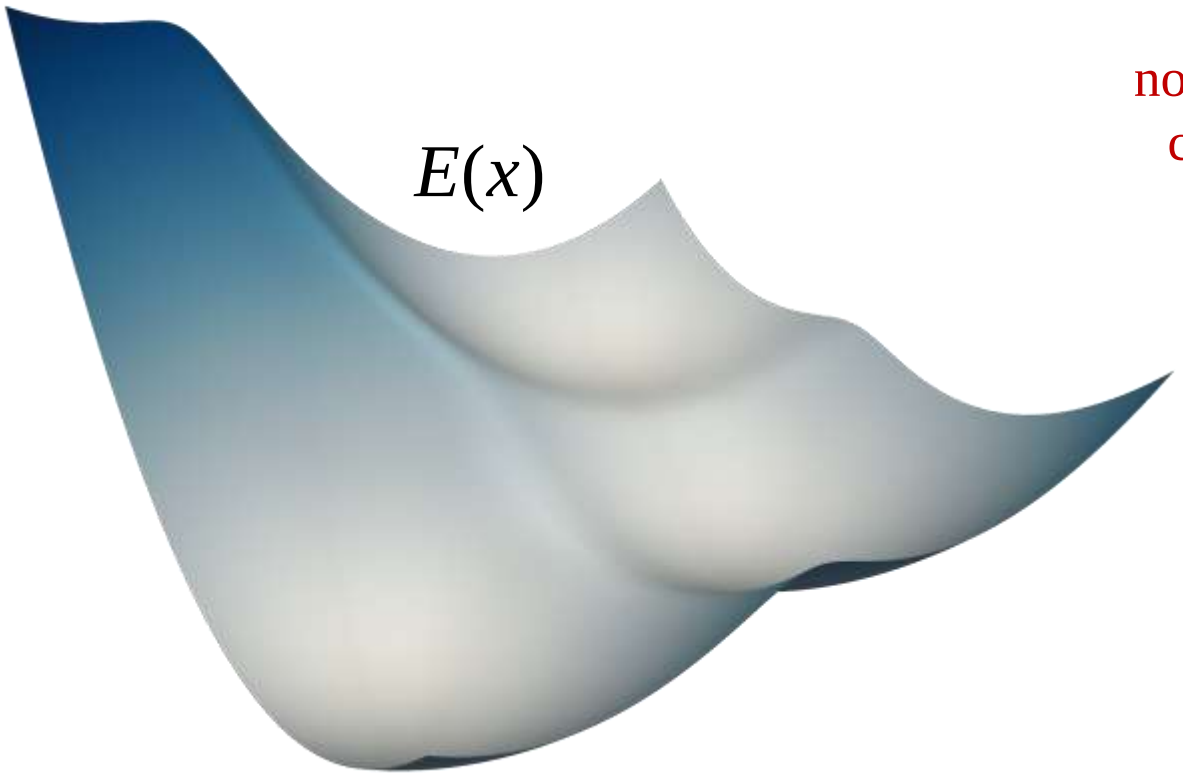


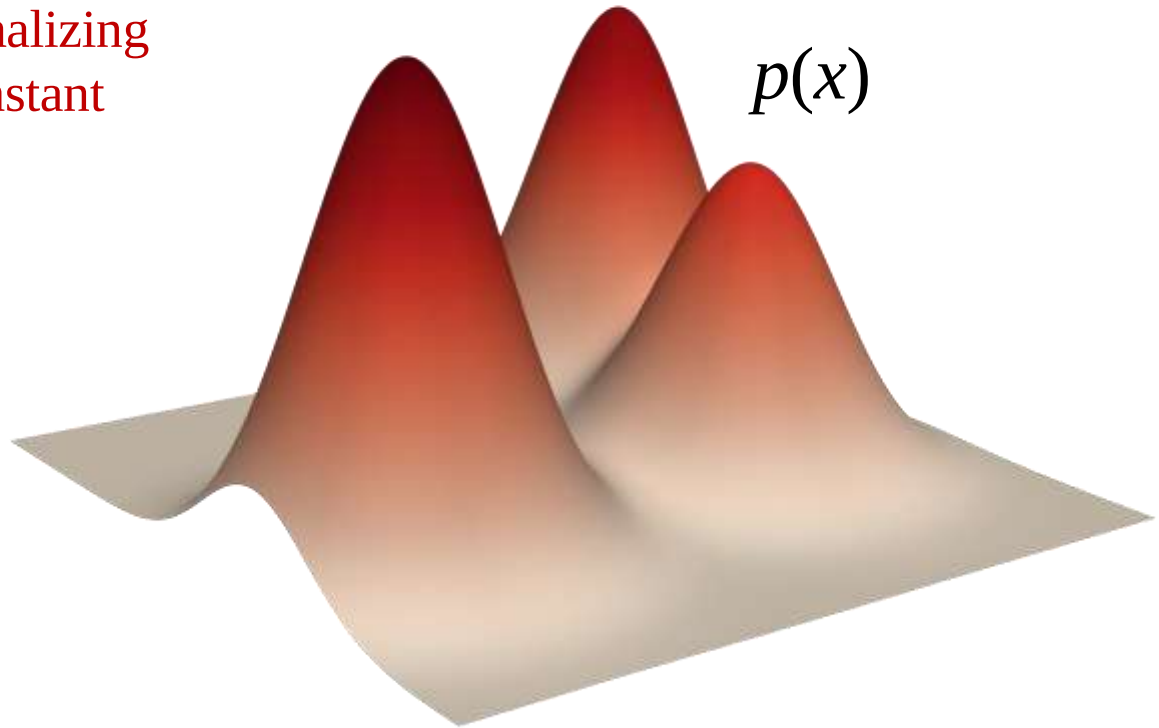
Energy-based Models

- We can use an energy to model a probability distribution

$$p(x) = \frac{\exp(-E(x))}{Z}$$

normalizing constant



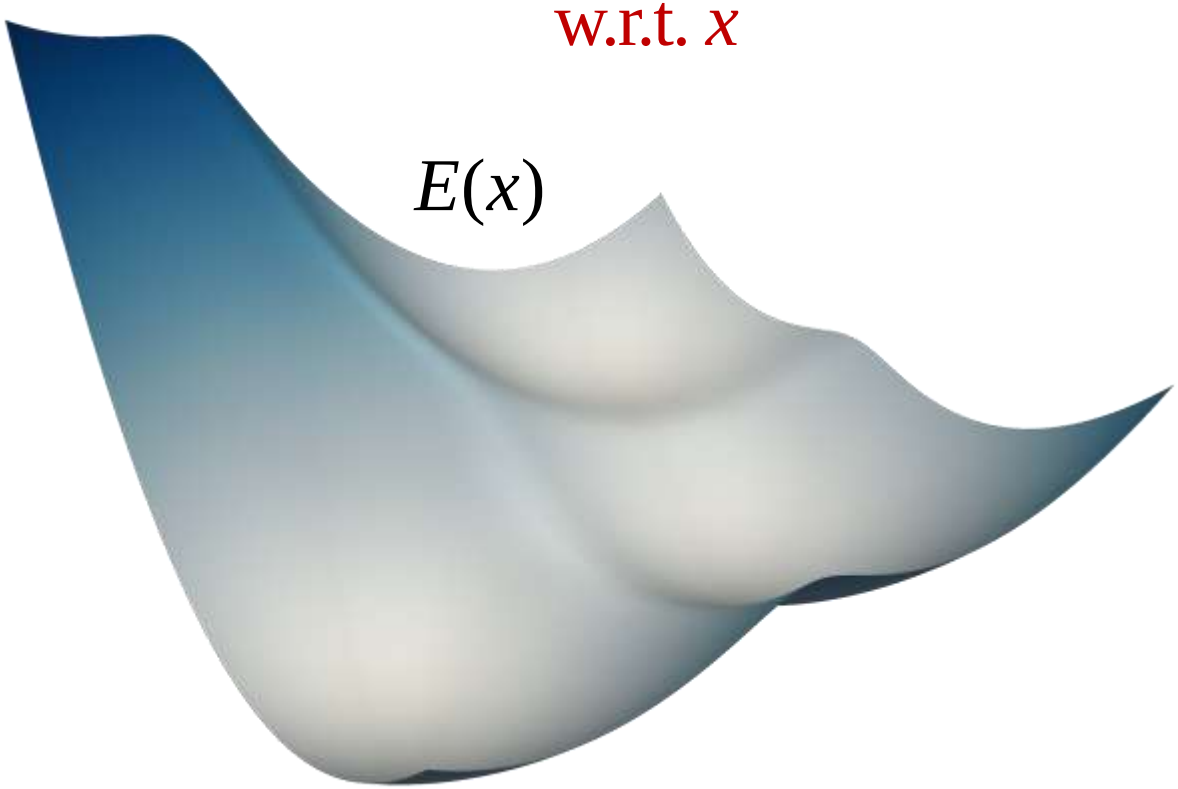


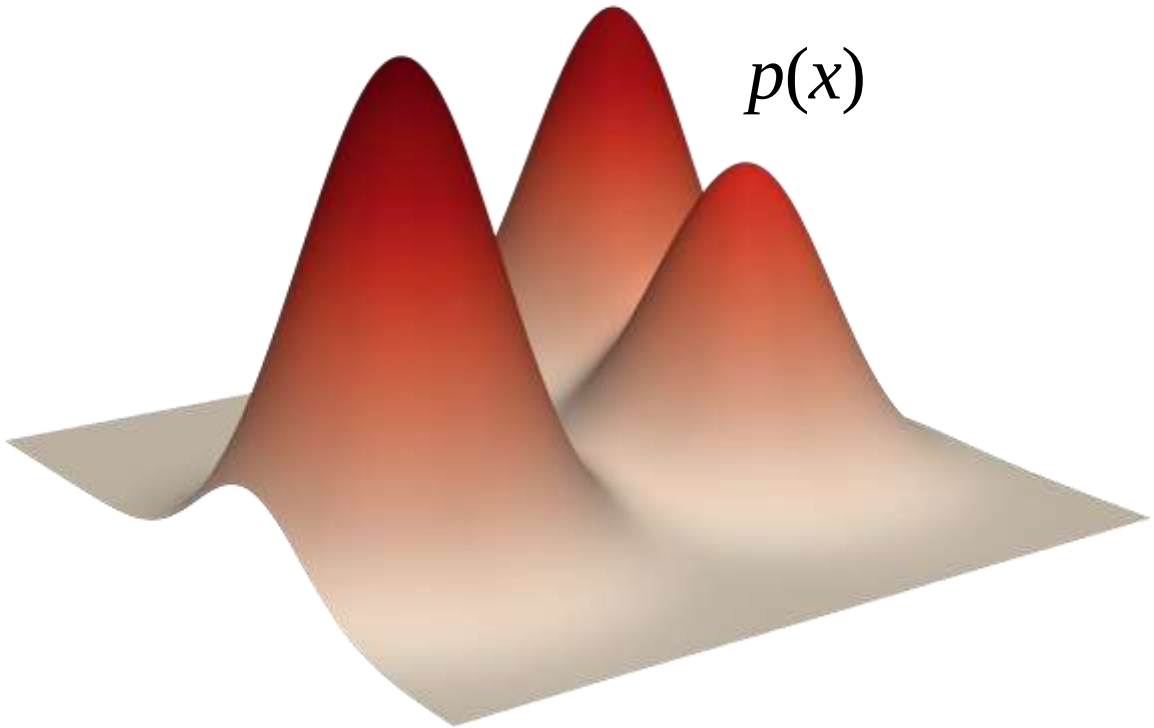
Energy-based Models

- **“Score function”**: gradient of log-probability.

$$\nabla_x \log p(x) = -\nabla_x E(x) - \nabla_x \log Z$$

w.r.t. x =0



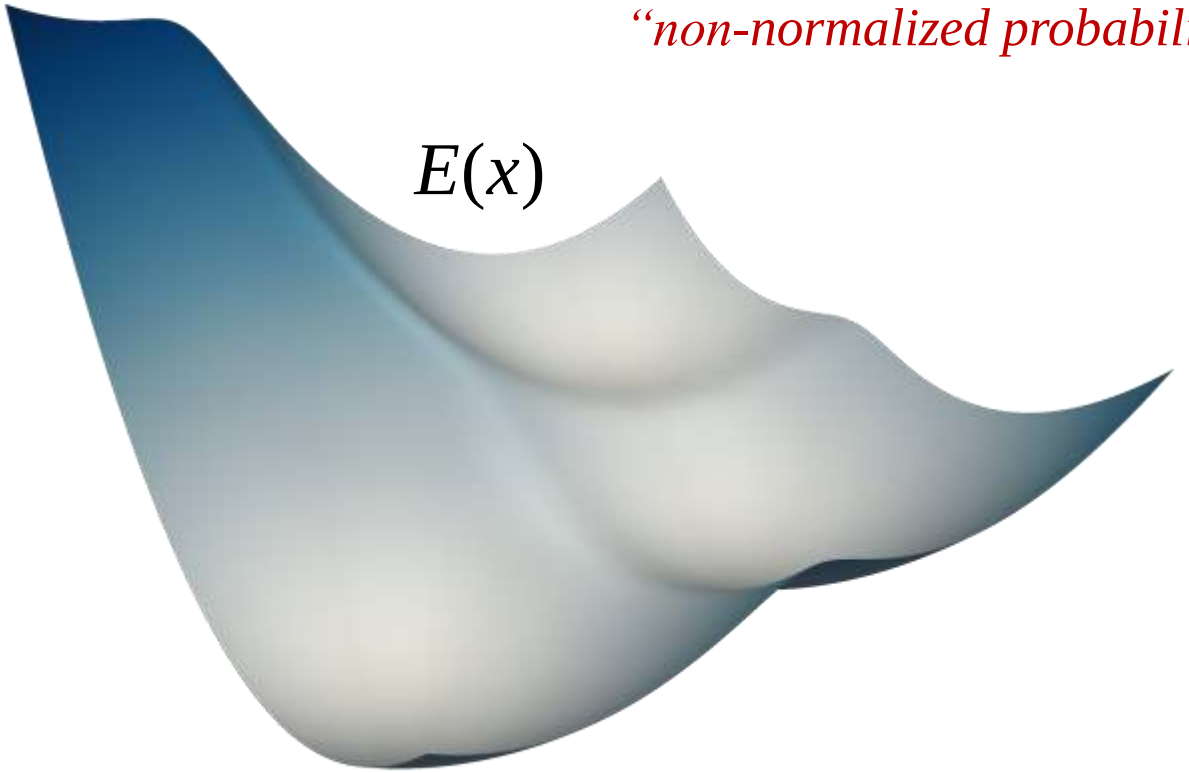


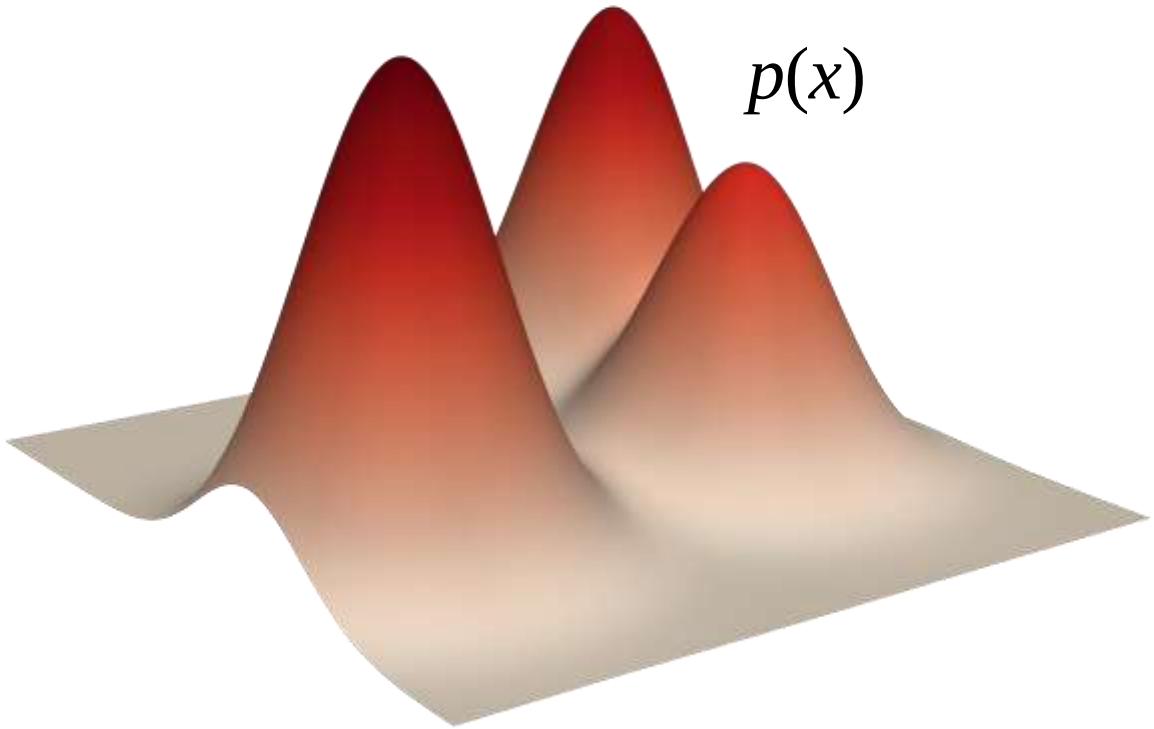
Energy-based Models

- **“Score function”**: gradient of log-probability

$$\nabla_x \log p(x) = -\nabla_x E(x)$$

“non-normalized probabilistic models”



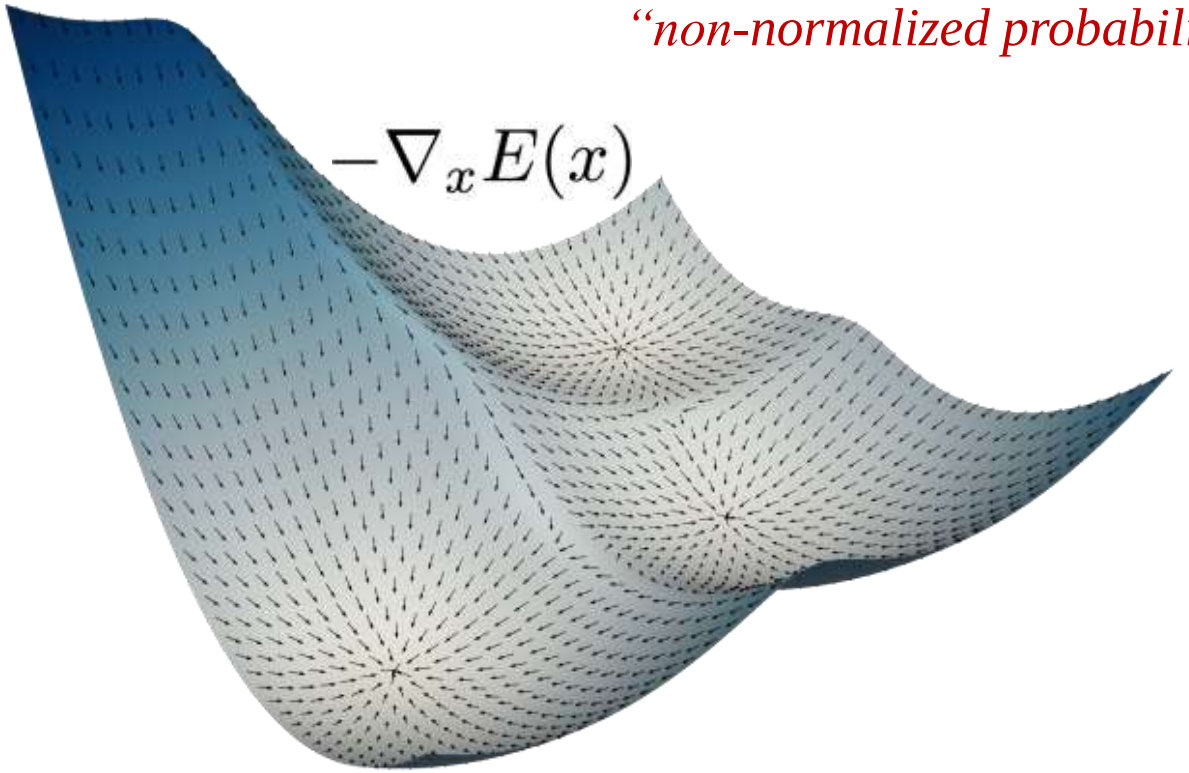


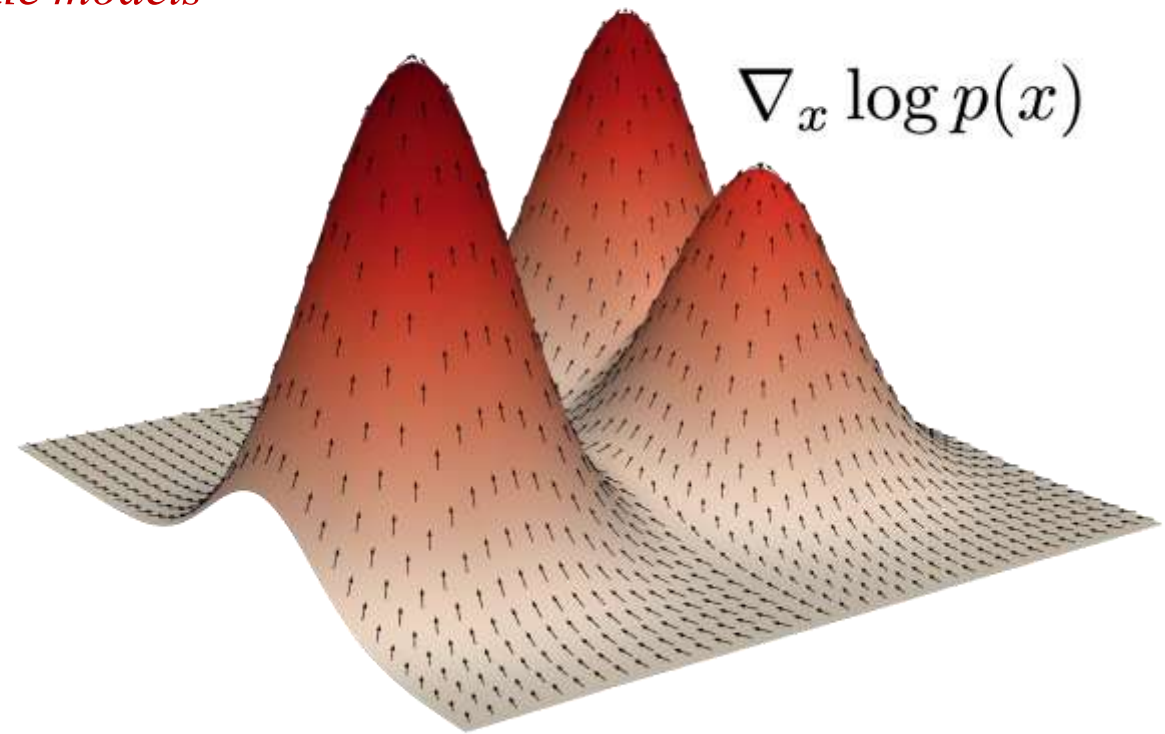
Energy-based Models

- **“Score function”**: gradient of log-probability

$$\nabla_x \log p(x) = -\nabla_x E(x)$$

“non-normalized probabilistic models”



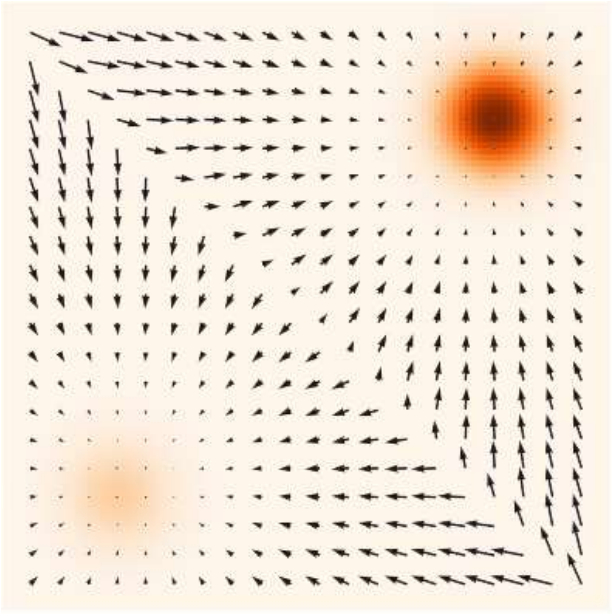


*only visualize

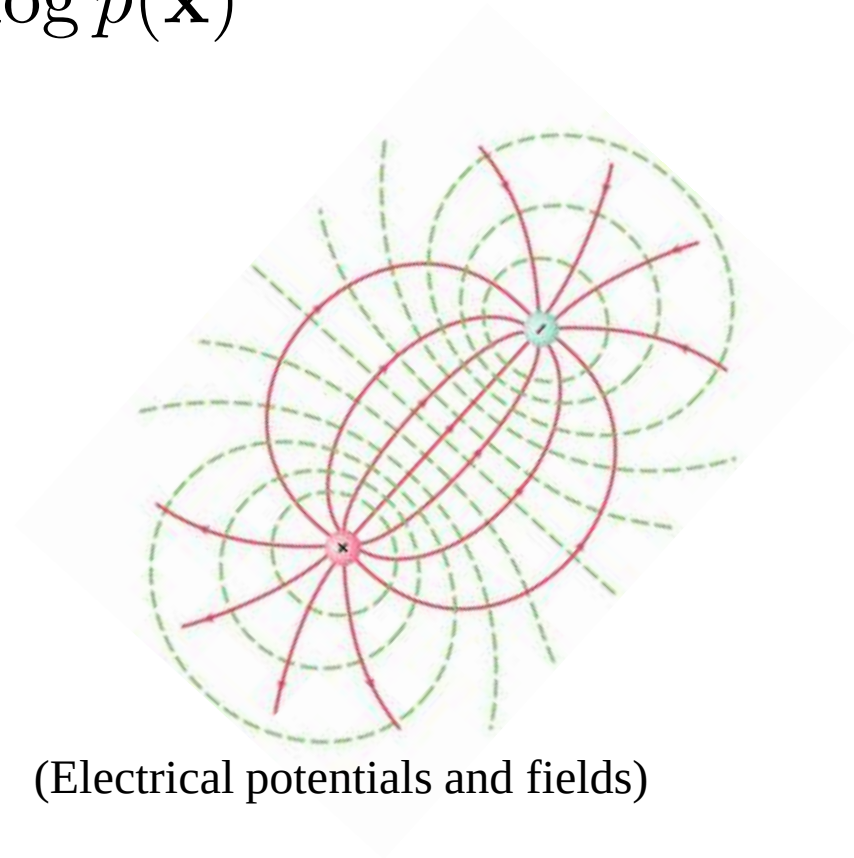
How to represent probability distributions?

- When the pdf is differentiable, we can compute the gradient of a probability density.

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



(pdf and score)

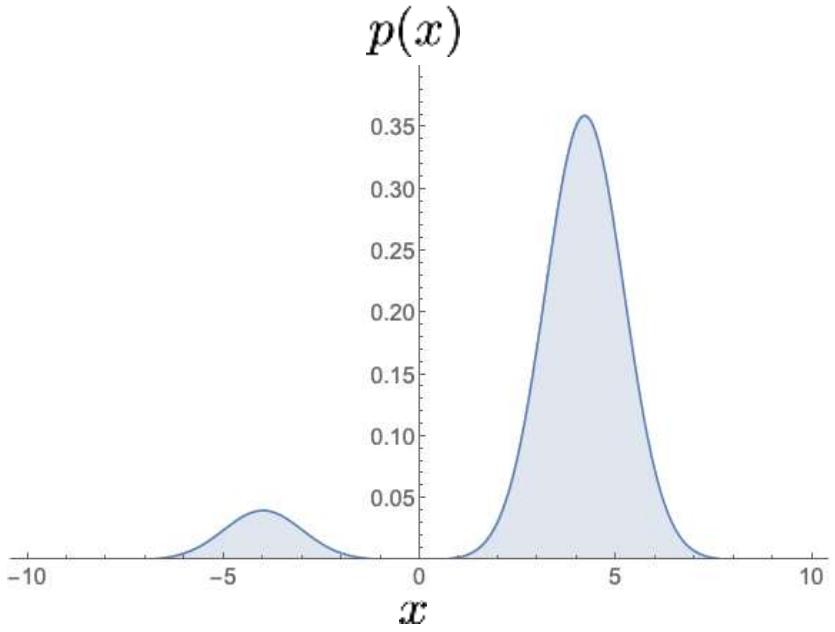


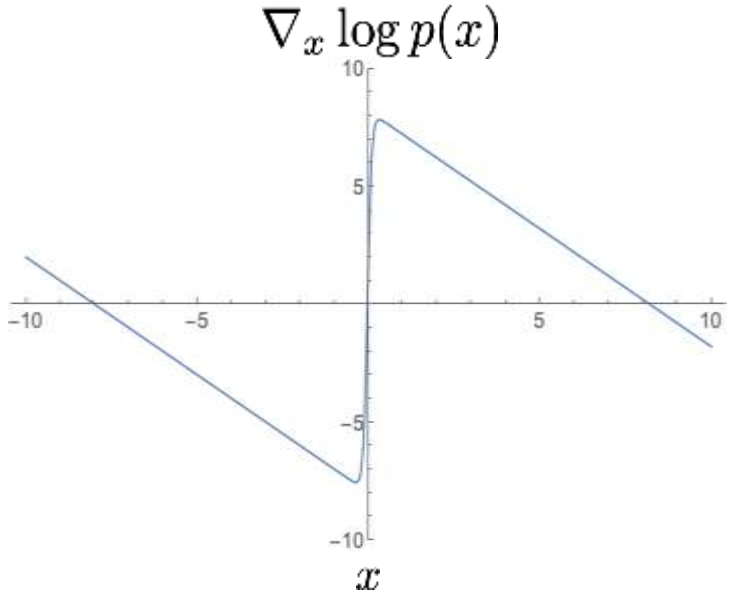
(Electrical potentials and fields)

How to represent probability distributions?

- When the pdf is differentiable, we can compute the gradient of a probability density.

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

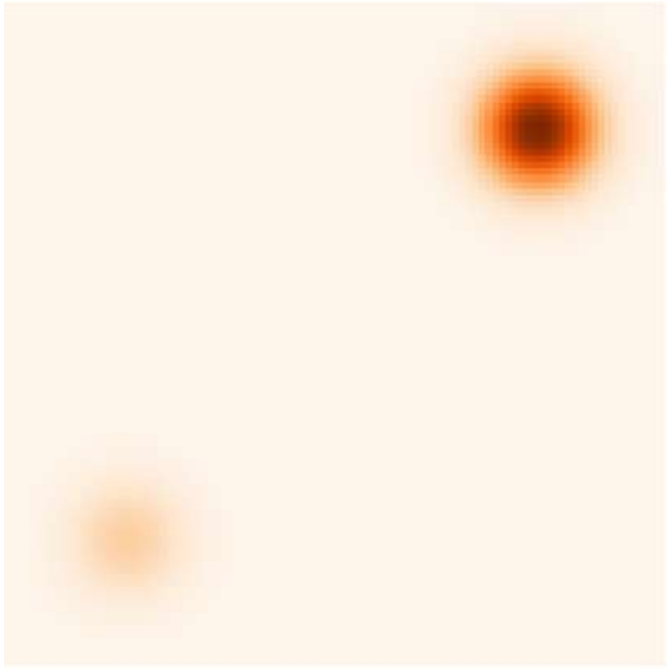




Score estimation by training score-based models

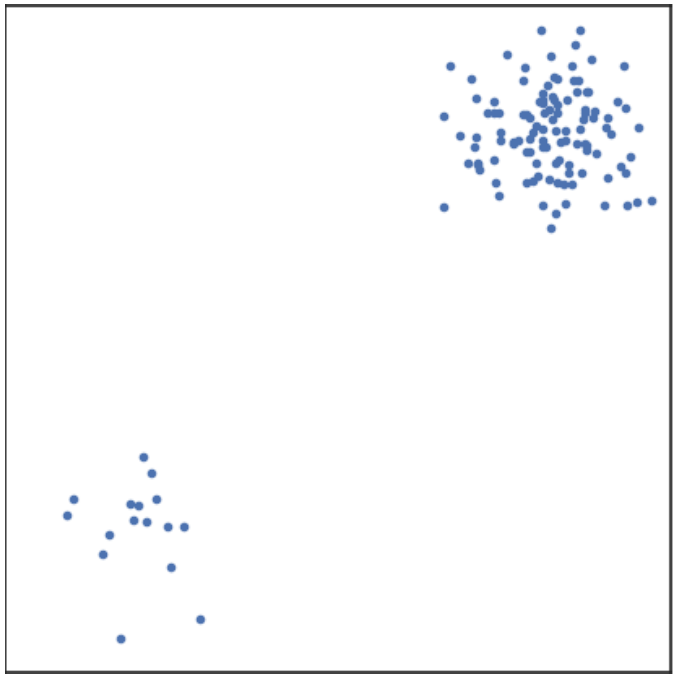
Probability density

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



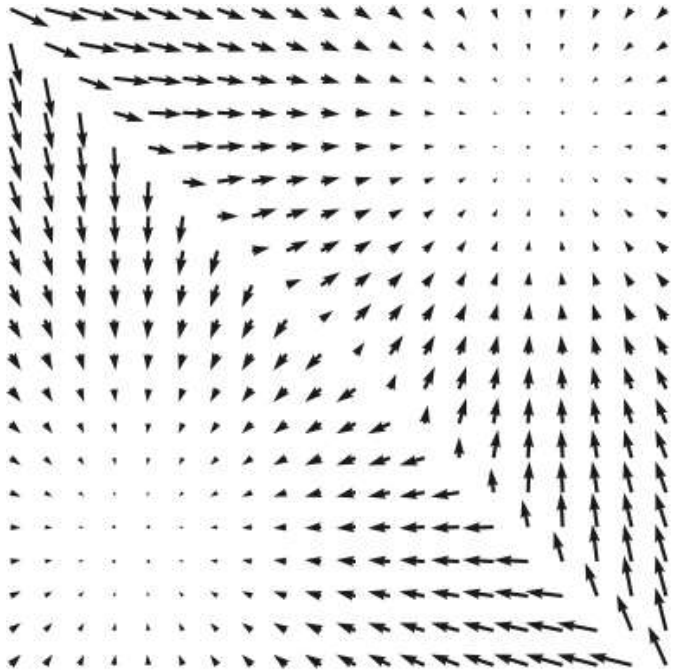
i.i.d. samples

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



Score function

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

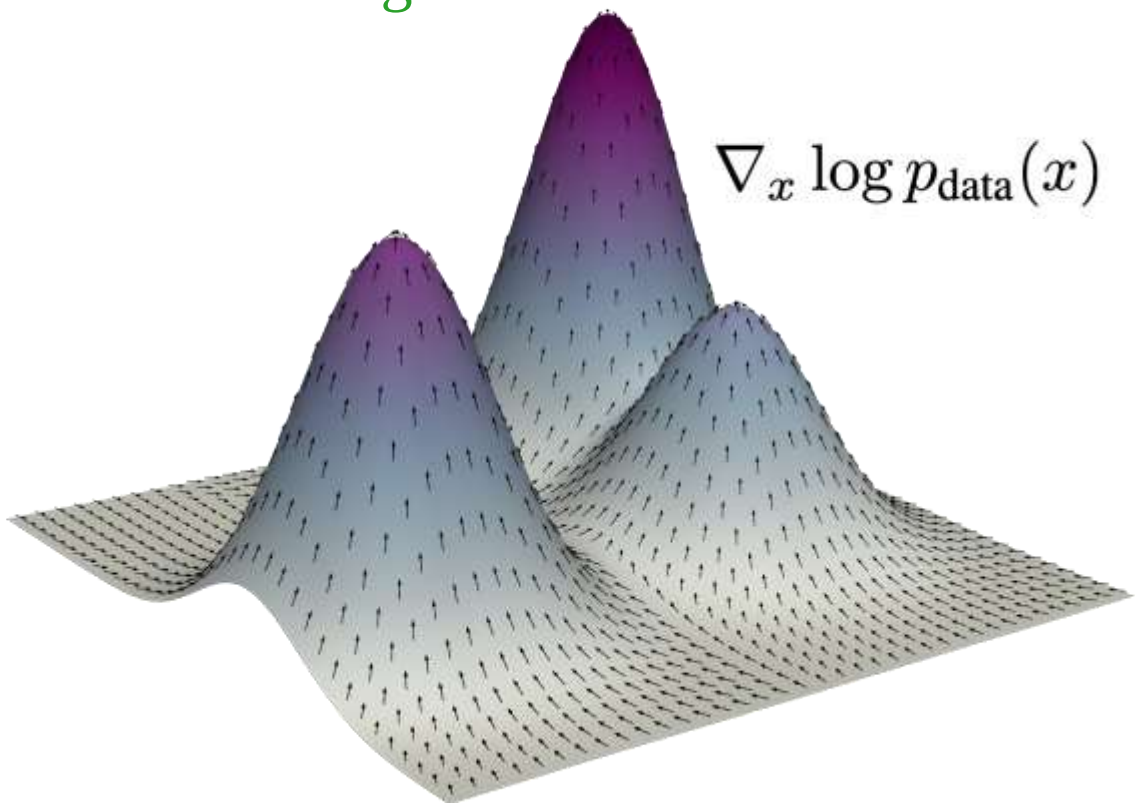


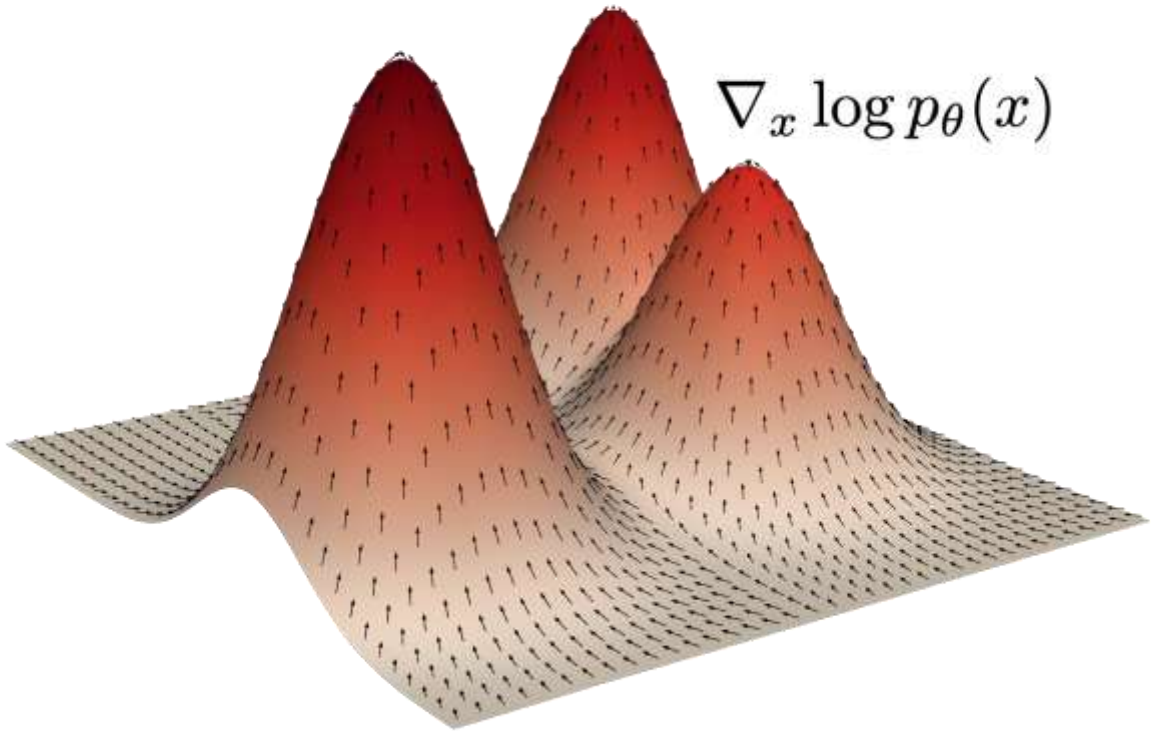
Score Matching

- Instead of parametrizing *p*, we can parametrize the score

$$D_F(p_{\text{data}}(x) \parallel p_\theta(x)) = \mathbb{E}_{p_{\text{data}}(x)}\left[\frac{1}{2}\|\nabla_x \log p_{\text{data}}(x) - \nabla_x \log p_\theta(x)\|^2\right]$$

Fisher divergence — score of data — parameterized score





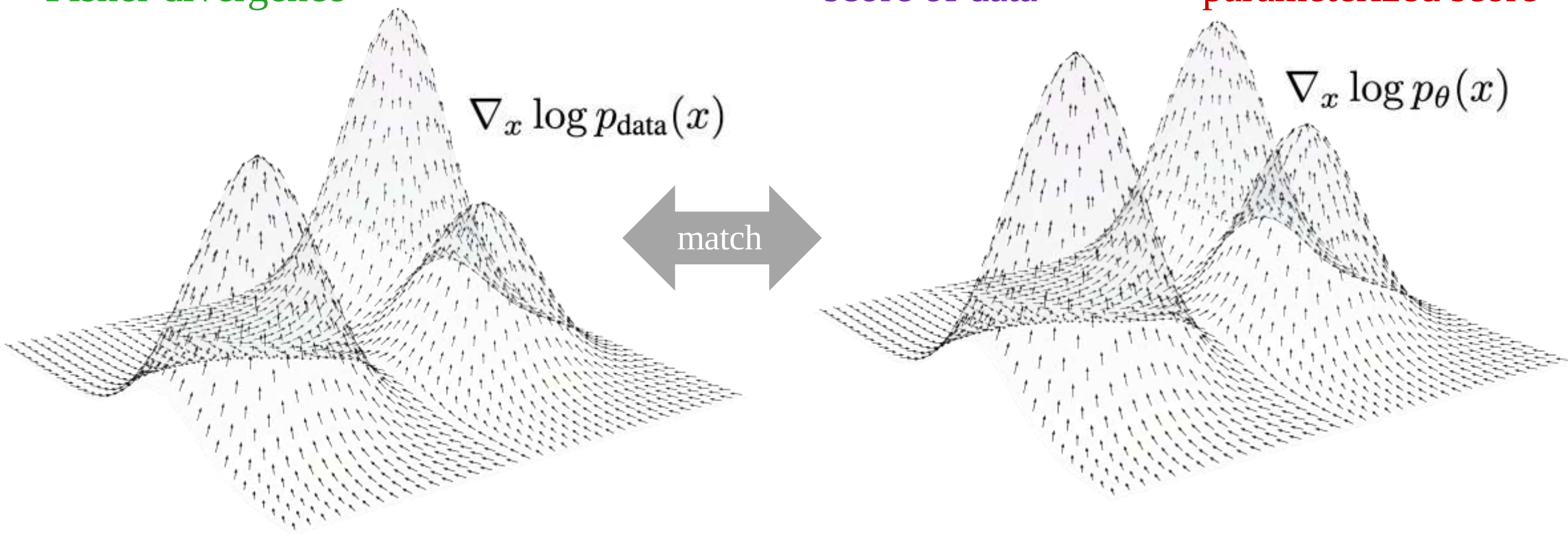
*only visualize

Score Matching

- Instead of parametrizing *p*, we can parametrize the score

$$D_F(p_{\text{data}}(x) \parallel p_\theta(x)) = \mathbb{E}_{p_{\text{data}}(x)}\left[\frac{1}{2}\|\nabla_x \log p_{\text{data}}(x) - \nabla_x \log p_\theta(x)\|^2\right]$$

Fisher divergence — score of data — parameterized score

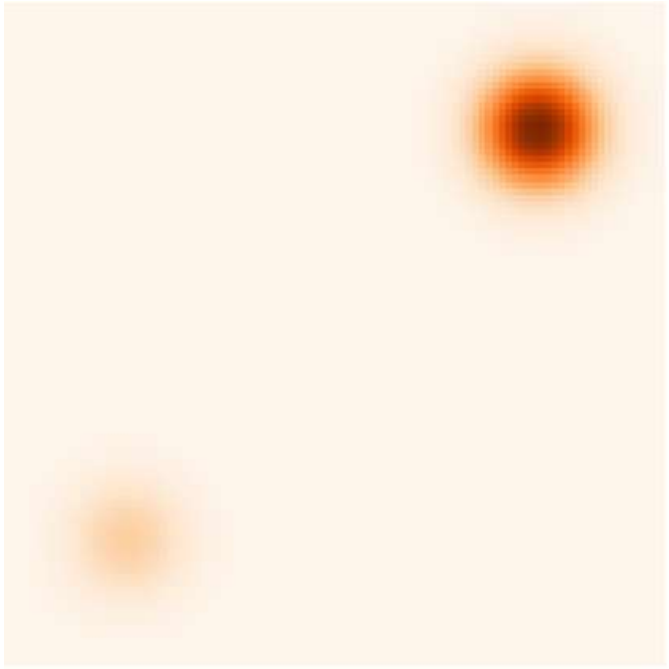


*only visualize

Score estimation by training score-based models

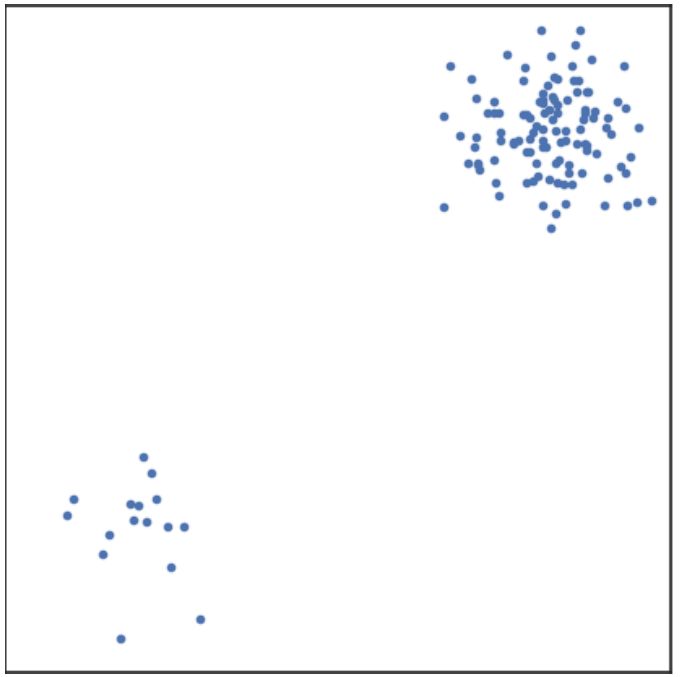
Probability density

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



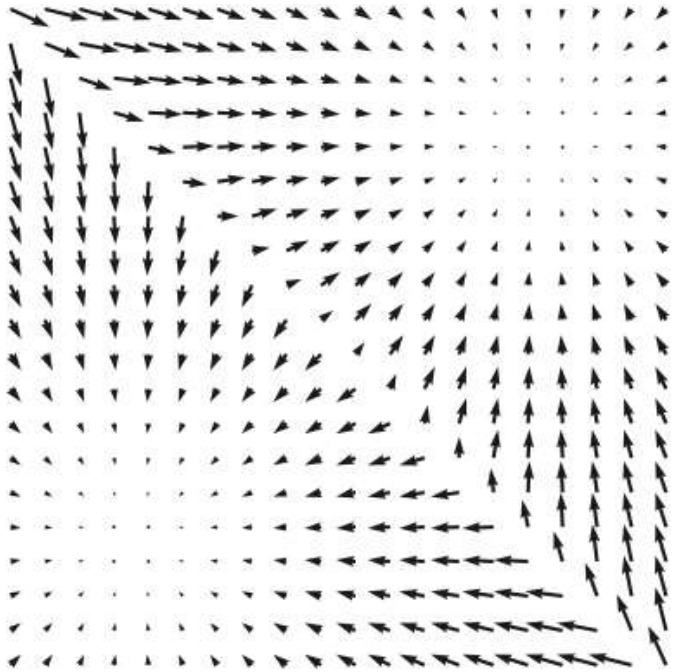
i.i.d. samples

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



Score function

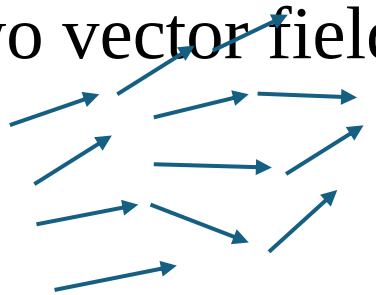
$$\boldsymbol{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$$



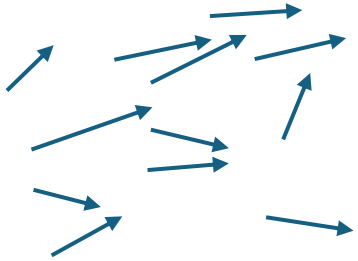
Score estimation by training score-based models

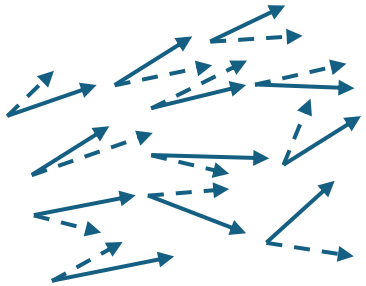
- Given: i.i.d. samples $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \sim p_{\text{data}}(\mathbf{x})$
- Task: Estimating the score $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$
- Score Model: A learnable vector-valued function $\boldsymbol{s}_\theta(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}^d$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

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



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





Average Euclidean distance over the space

Denoising Score Matching

- with noised data $\tilde{x} := x + \epsilon$, it can be proven: [Vincent, 2011]

$$D_F(q(\tilde{x}) \parallel p_\theta(\tilde{x})) = \mathbb{E}_{q(x,\tilde{x})}\left[\frac{1}{2}\left\|\nabla_{\tilde{x}} \log q(\tilde{x} \mid x) - \nabla_{\tilde{x}} \log p_\theta(\tilde{x})\right\|^2\right] + \text{constant}$$

Fisher divergence of noised data

joint distribution

score of conditional

parameterized score

See: Vincent, “A Connection Between Score Matching and Denoising Autoencoders”, Neural Computation, 2011

Denoising Score Matching

- with noised data $\tilde{x} := x + \epsilon$, it can be proven: [Vincent, 2011]

$$D_F(q(\tilde{x}) \parallel p_\theta(\tilde{x})) = \mathbb{E}_{q(x,\tilde{x})}\left[\frac{1}{2}\left\|\nabla_{\tilde{x}} \log q(\tilde{x} \mid x) - \nabla_{\tilde{x}} \log p_\theta(\tilde{x})\right\|^2\right] + \text{constant}$$

a network to predict (negative) noise

Gaussian noise:

$$= \frac{1}{\sigma^2}(x - \tilde{x})$$

$$-\epsilon$$

See: Vincent, “A Connection Between Score Matching and Denoising Autoencoders”, Neural Computation, 2011

Langevin Dynamics

- Given a score function, we can sample *x* from *p* by iterating:

$$x_t \leftarrow x_{t-1} + \frac{\sigma^2}{2}\nabla_x \log p_\theta(x_{t-1}) + \sigma z_t$$

step size

score function
(don't need to know *p*)

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

(neg) gradient of energy

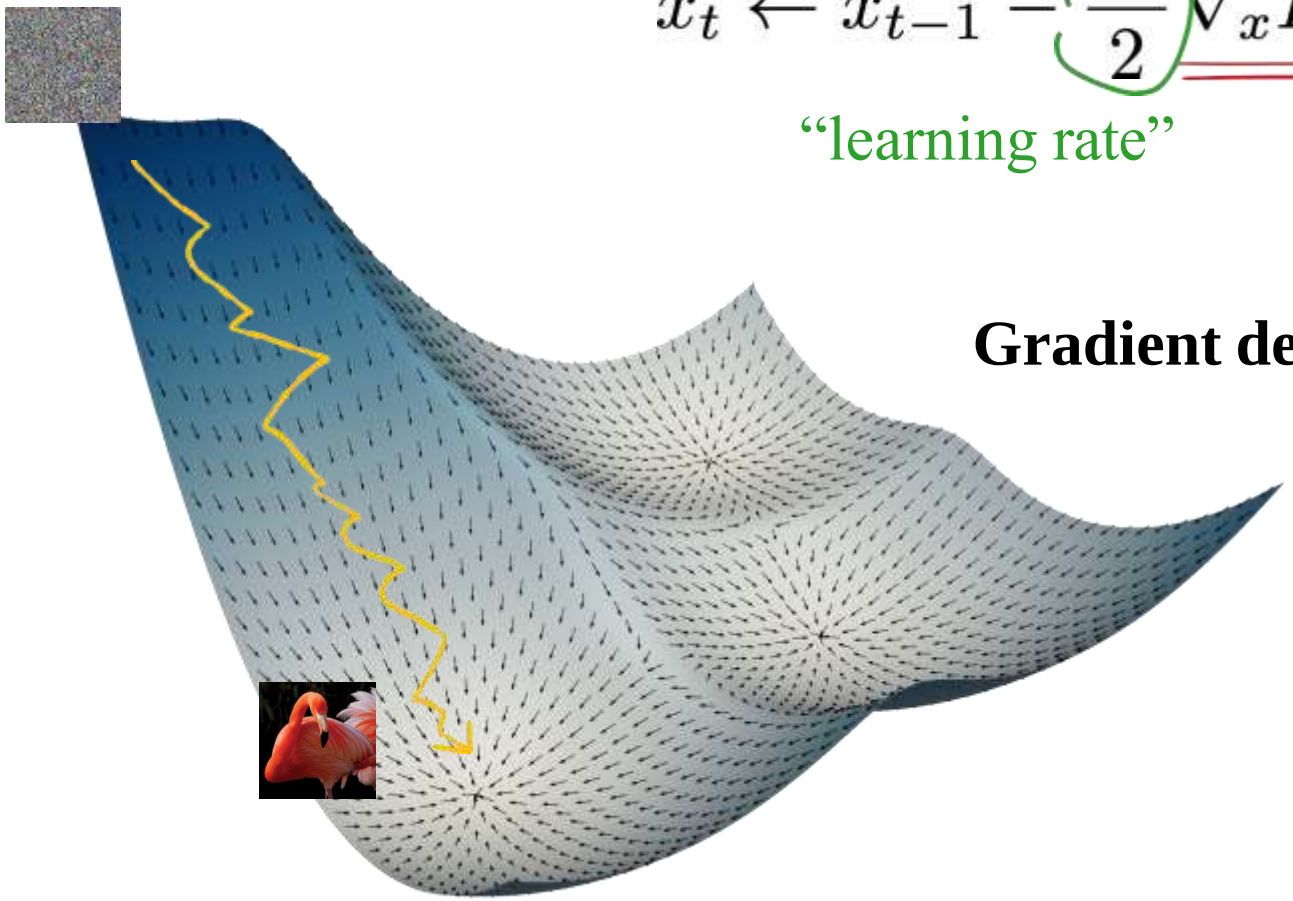
$$-\nabla_x E_\theta(x_{t-1})$$

Langevin Dynamics

- Given a score function, we can sample x from p by iterating:

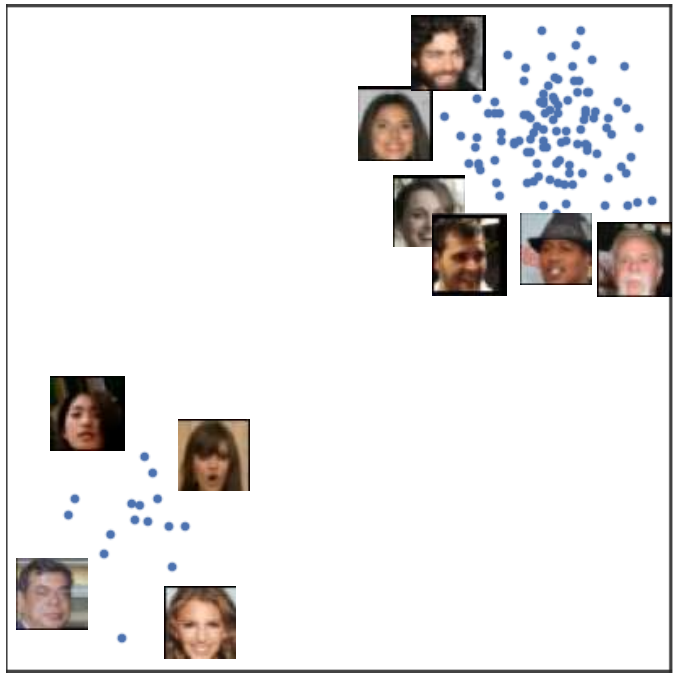
$$x_t \leftarrow x_{t-1} - \frac{\sigma^2}{2}\nabla_x E_\theta(x_{t-1}) + \sigma z_t$$

"learning rate" gradient perturbation



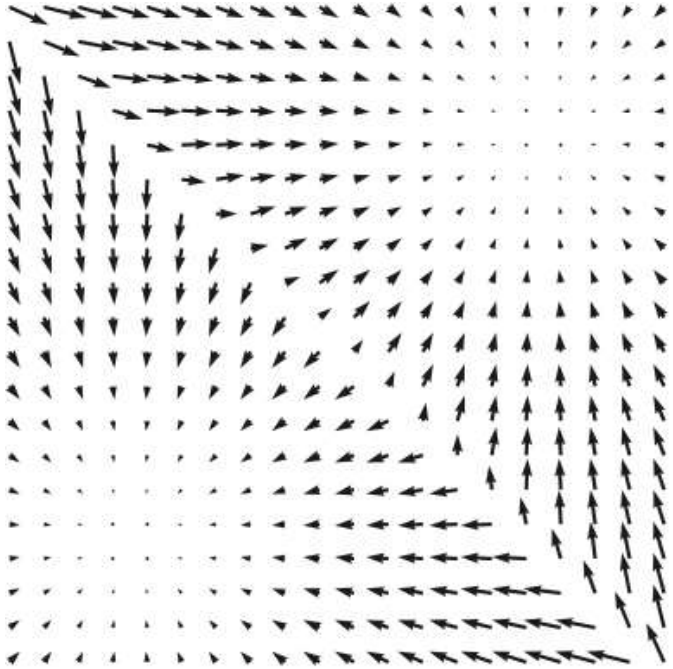
Gradient decent in the energy landscape

Score-based generative modeling



Data samples

Score Matching



Scores

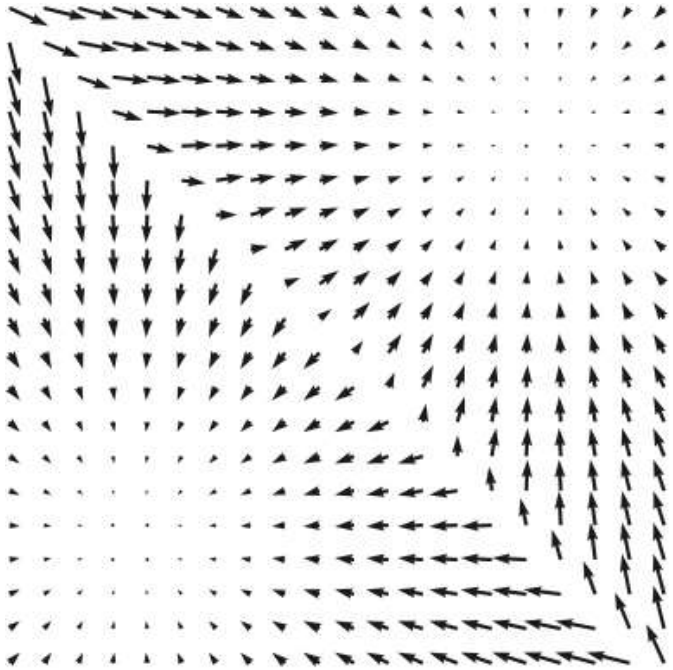
?

New samples

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

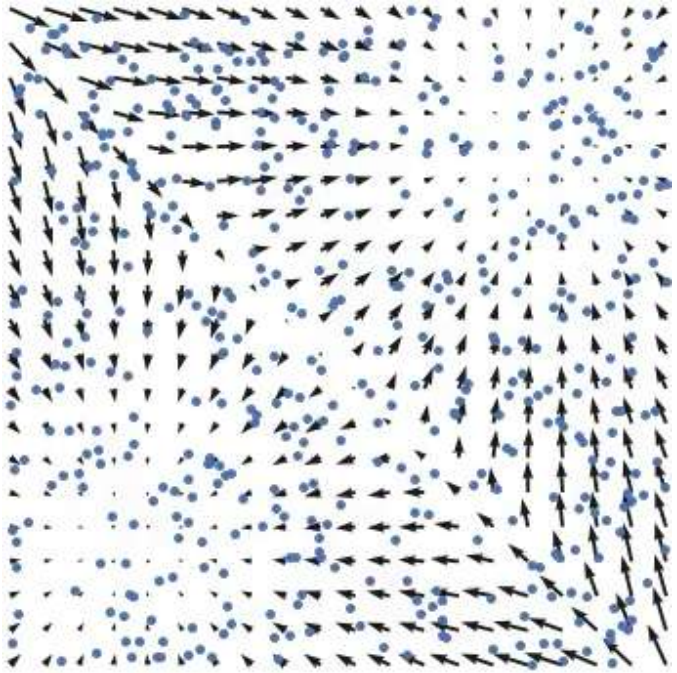
$$\boldsymbol{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$$

From scores to samples: Langevin MCMC



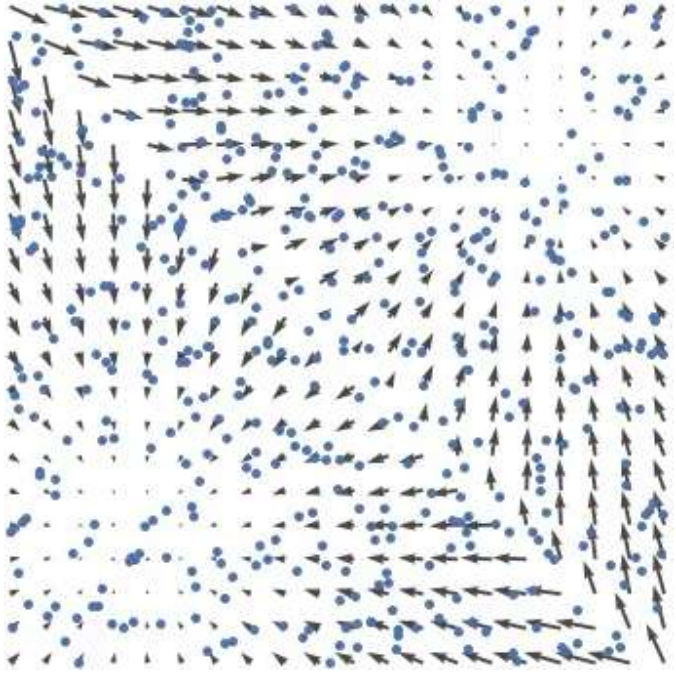
Scores

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



Follow the scores

$$\tilde{\mathbf{x}}_{t+1} \leftarrow \tilde{\mathbf{x}}_t + \frac{\epsilon}{2}\mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}_t)$$

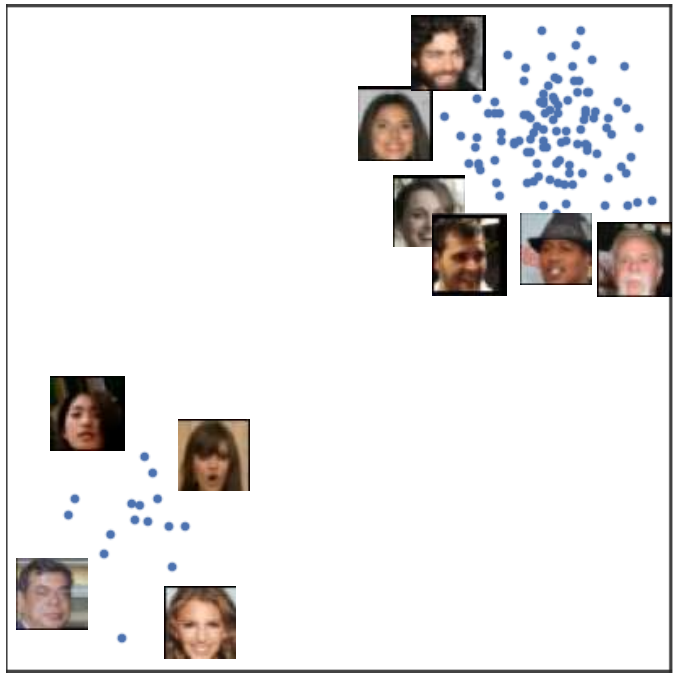


Follow noisy scores:
Langevin MCMC

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

$$\tilde{\mathbf{x}}_{t+1} \leftarrow \tilde{\mathbf{x}}_t + \frac{\epsilon}{2}\mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}_t) + \sqrt{\epsilon}\,\mathbf{z}_t$$

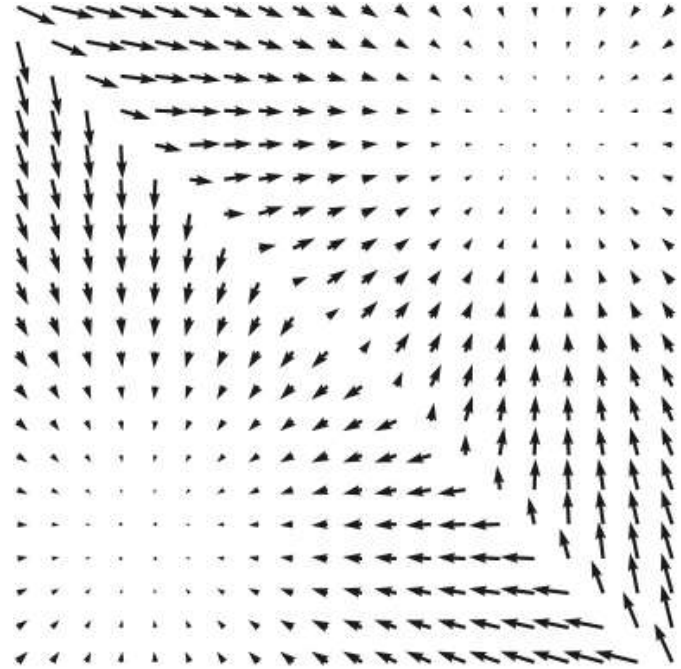
Score-based generative modeling



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

(Recap) Diffusion algorithm

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

score function

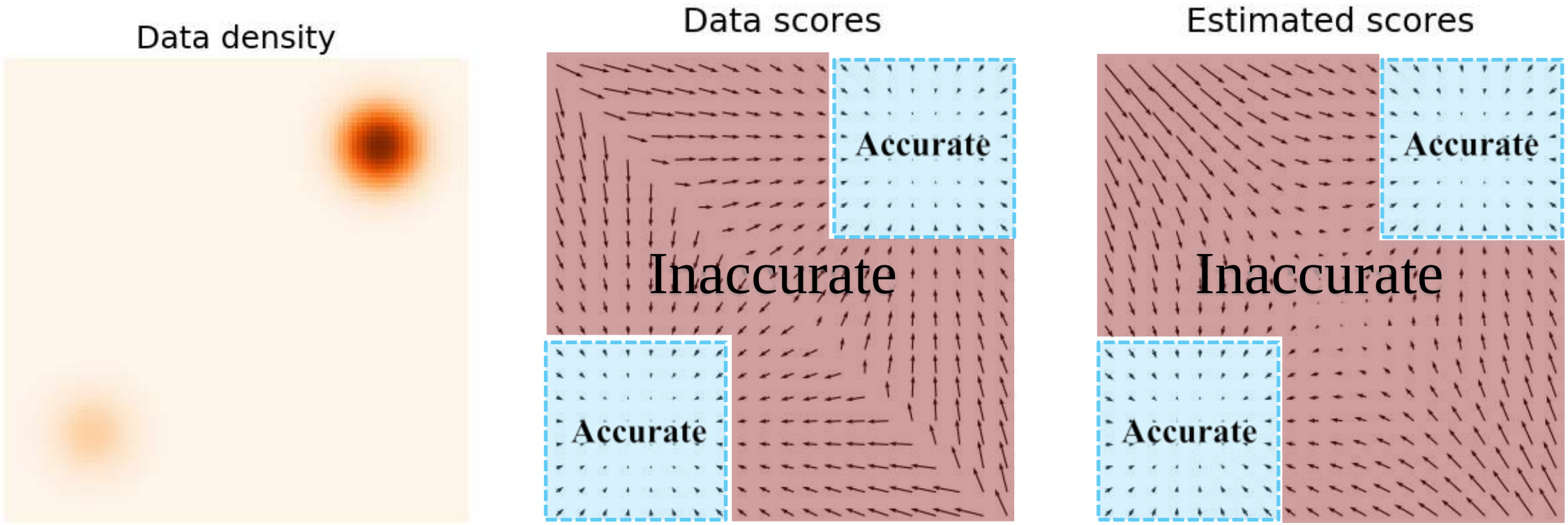
Algorithm 2 Sampling

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** $\mathbf{x}_0$

score function

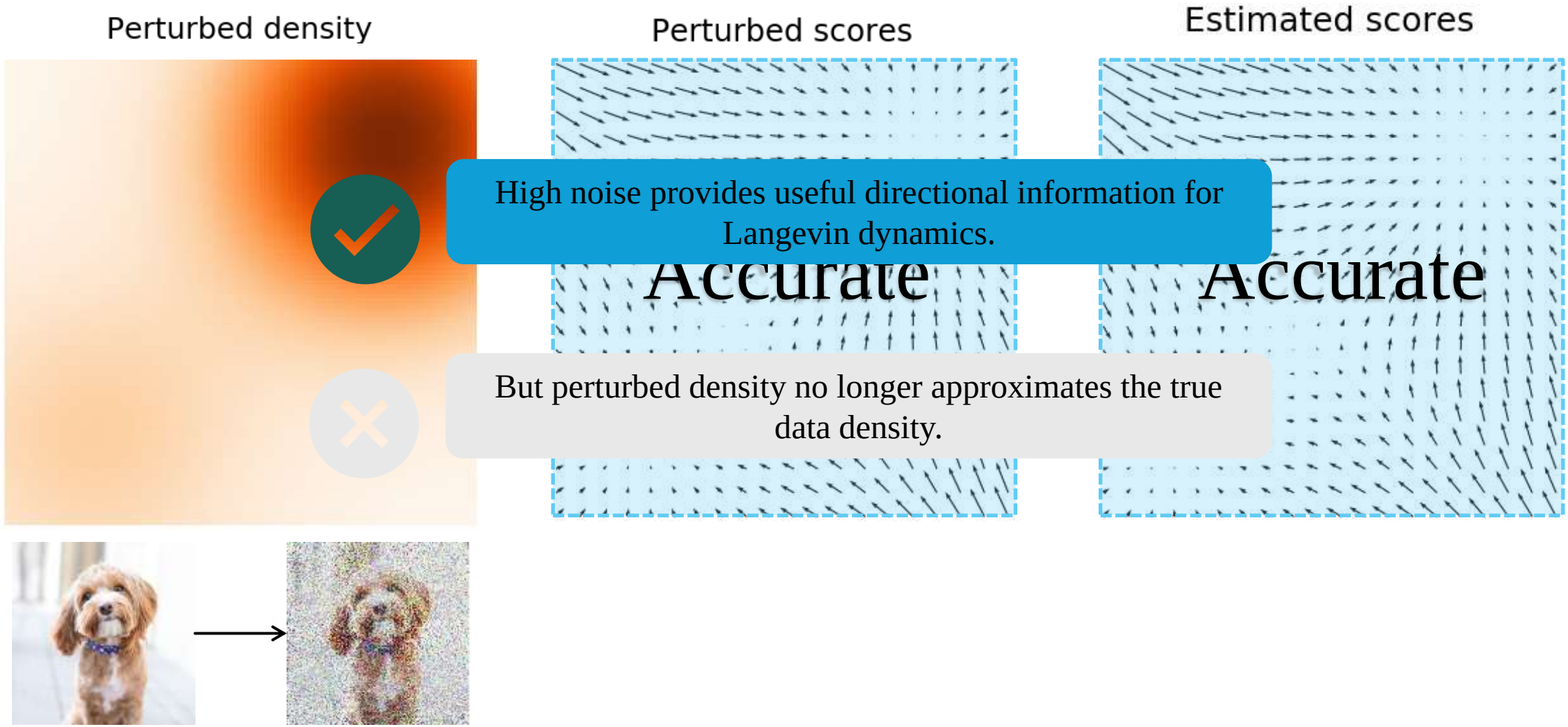
Langevin Dynamics

Challenge in low data density regions



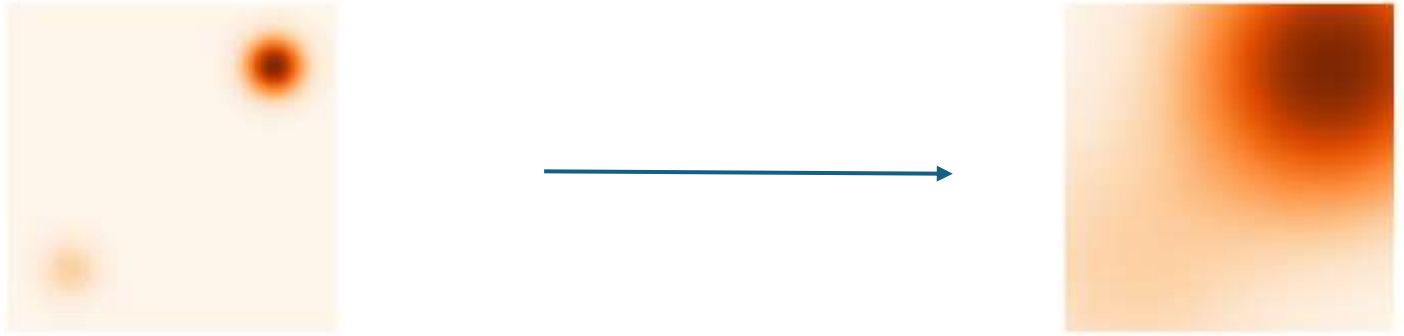
Song and Ermon. “Generative Modeling by Estimating Gradients of the Data Distribution.” NeurIPS 2019.

Improving score estimation by adding noise



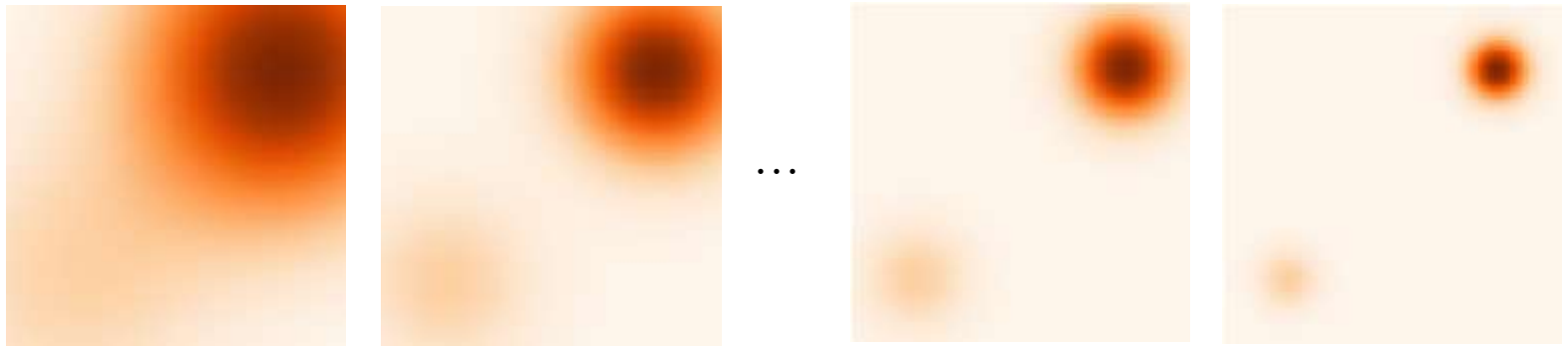
Multi-scale Noise Perturbation

- How much noise to add?

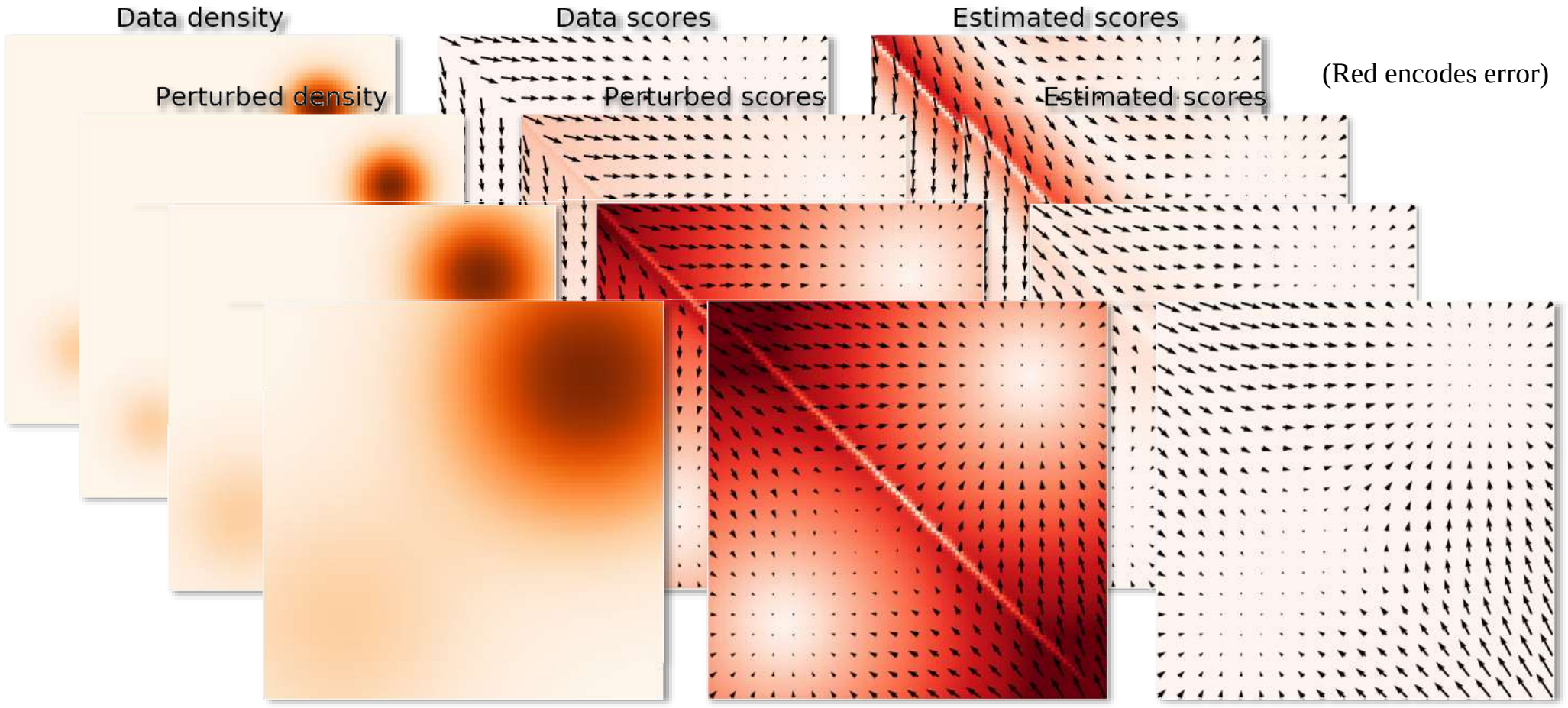


- Multi-scale noise perturbations.

$$\sigma_1 > \sigma_2 > \cdots > \sigma_{L-1} > \sigma_L$$



Trading off Data Quality and Estimation Accuracy

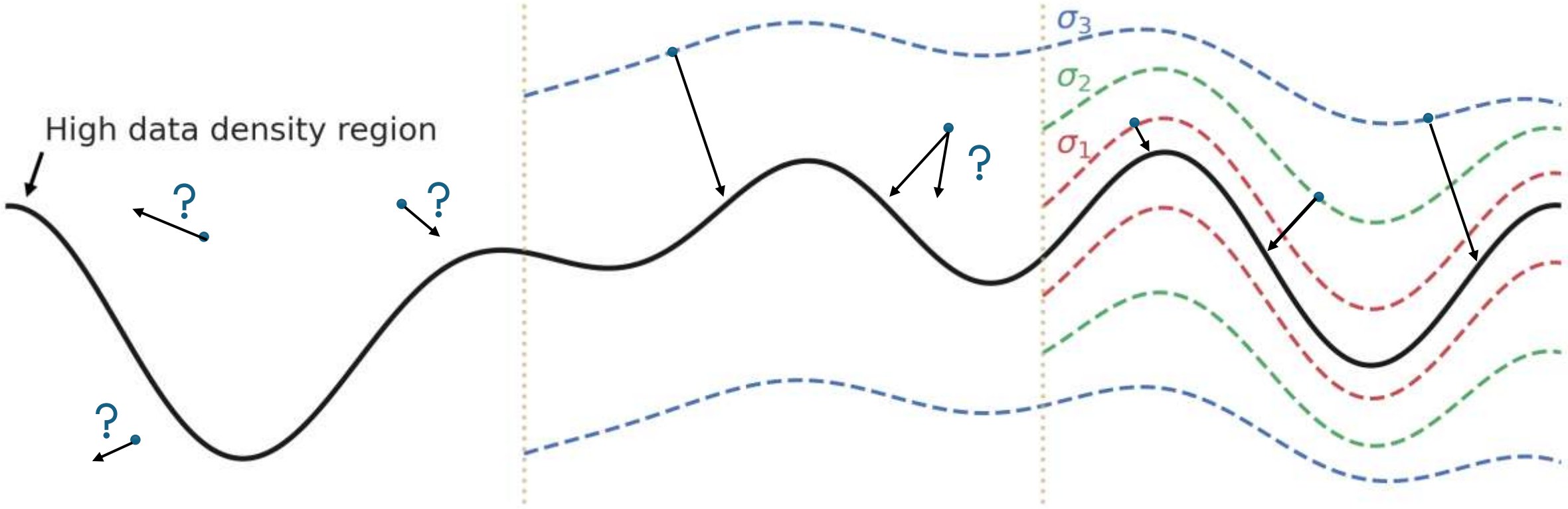


(Red encodes error)

Worse data quality!

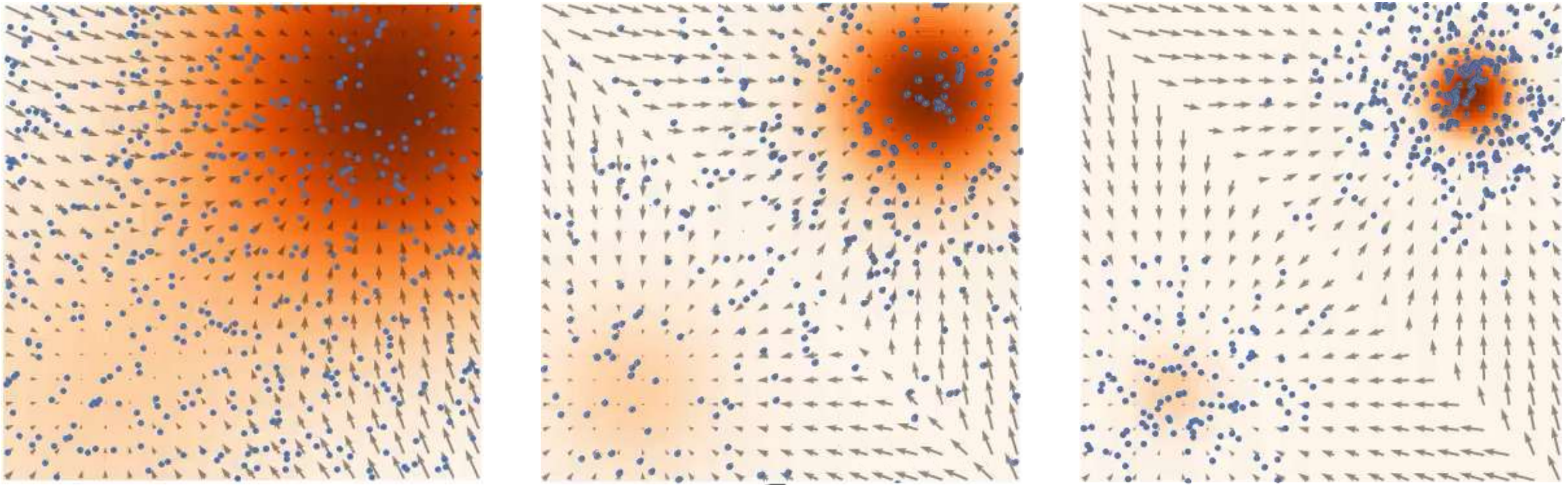
Better score estimation!

Using multiple noise scales



Annealed Langevin Dynamics: Joint Scores to Samples

- Sample using $\sigma_1, \sigma_2, \cdots, \sigma_L$ sequentially with Langevin dynamics.
- Anneal down the noise level.
- Samples used as initialization for the next level.



Annealed Langevin dynamics

Algorithm 1 Annealed Langevin dynamics.

Require: $\{\sigma_i\}_{i=1}^{L}, \epsilon, T$.

Initialize $\tilde{\mathbf{x}}_0$

for $i \leftarrow 1$ to L **do**

  $\alpha_i \leftarrow \epsilon \cdot \sigma_i^2 / \sigma_L^2$ ▷ α_i is the step size.

  for $t \leftarrow 1$ to T **do**

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

    $\tilde{\mathbf{x}}_t \leftarrow \tilde{\mathbf{x}}_{t-1} + \frac{\alpha_i}{2} \mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}_{t-1}, \sigma_i) + \sqrt{\alpha_i}\, \mathbf{z}_t$

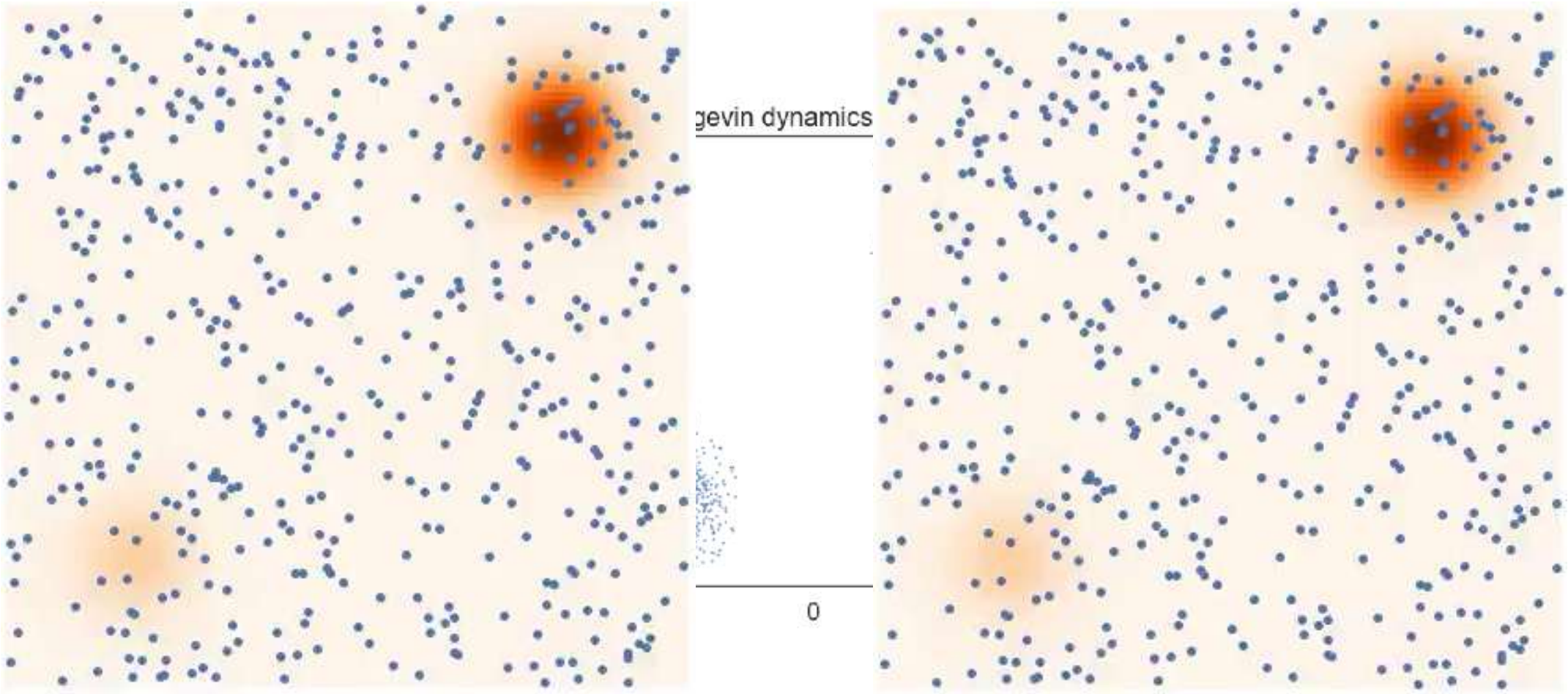
  end for

  $\tilde{\mathbf{x}}_0 \leftarrow \tilde{\mathbf{x}}_T$

end for

return $\tilde{\mathbf{x}}_T$

Comparison to the vanilla Langevin dynamics

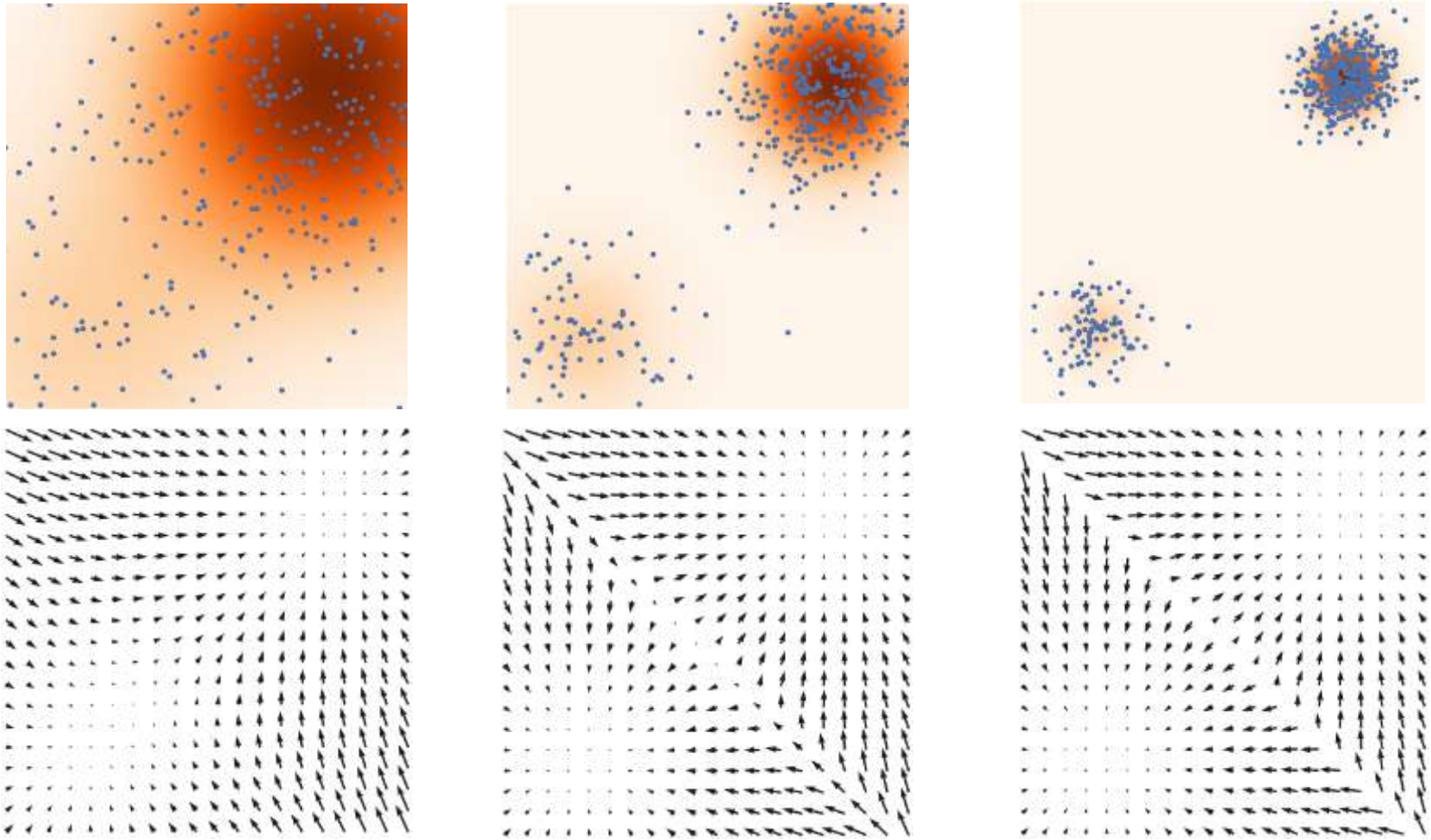


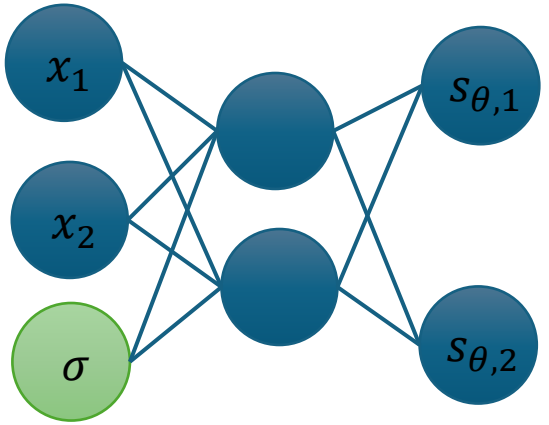
Langevin dynamics

Annealed Langevin dynamics

Joint Score Estimation via Noise Conditional Score Networks

σ_1 σ_2 σ_3



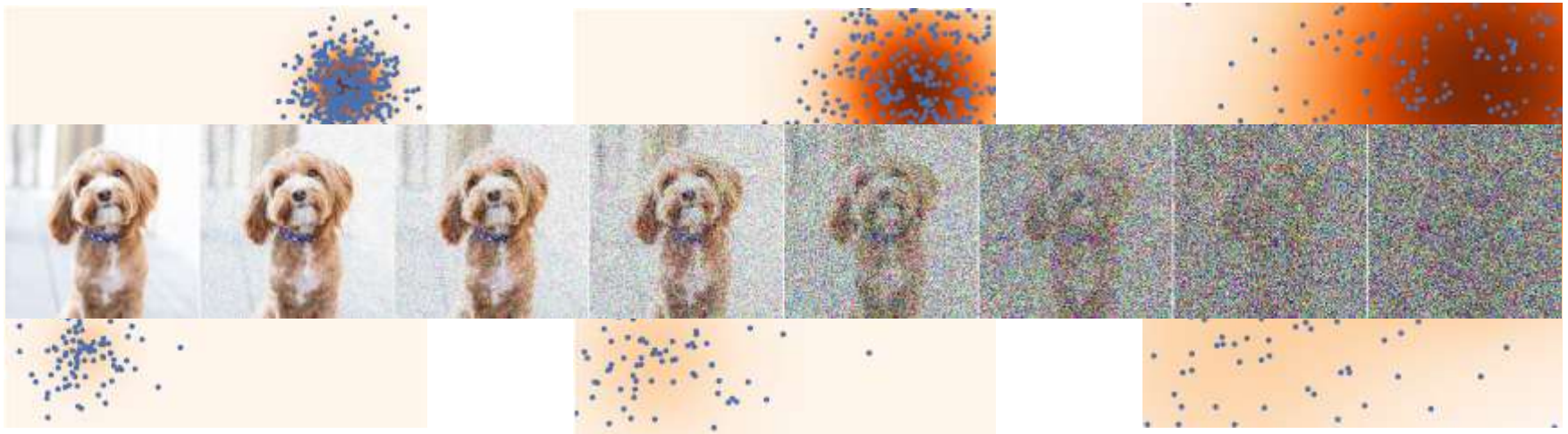


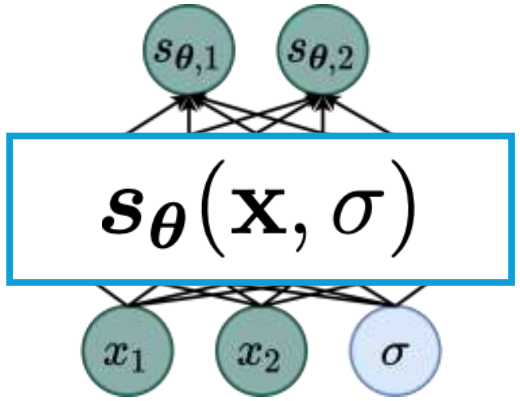
Noise Conditional Score Network (NCSN)

Using multiple noise levels

$p_{\sigma_1}(\mathbf{x}) < p_{\sigma_2}(\mathbf{x}) < p_{\sigma_3}(\mathbf{x})$

Data





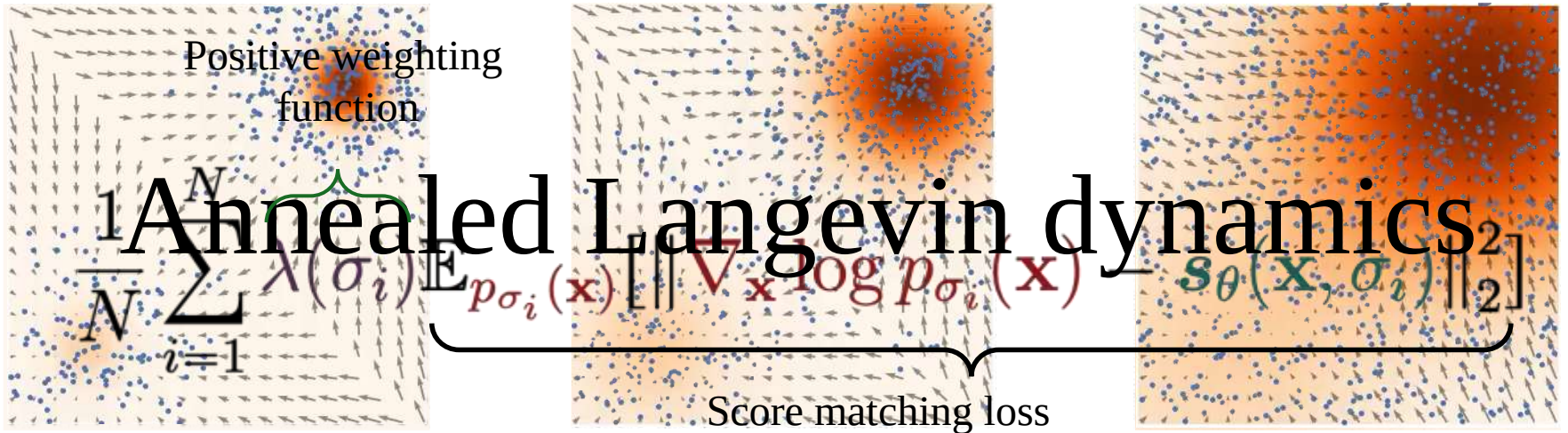
Noise Conditional Score Model

Positive weighting function

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

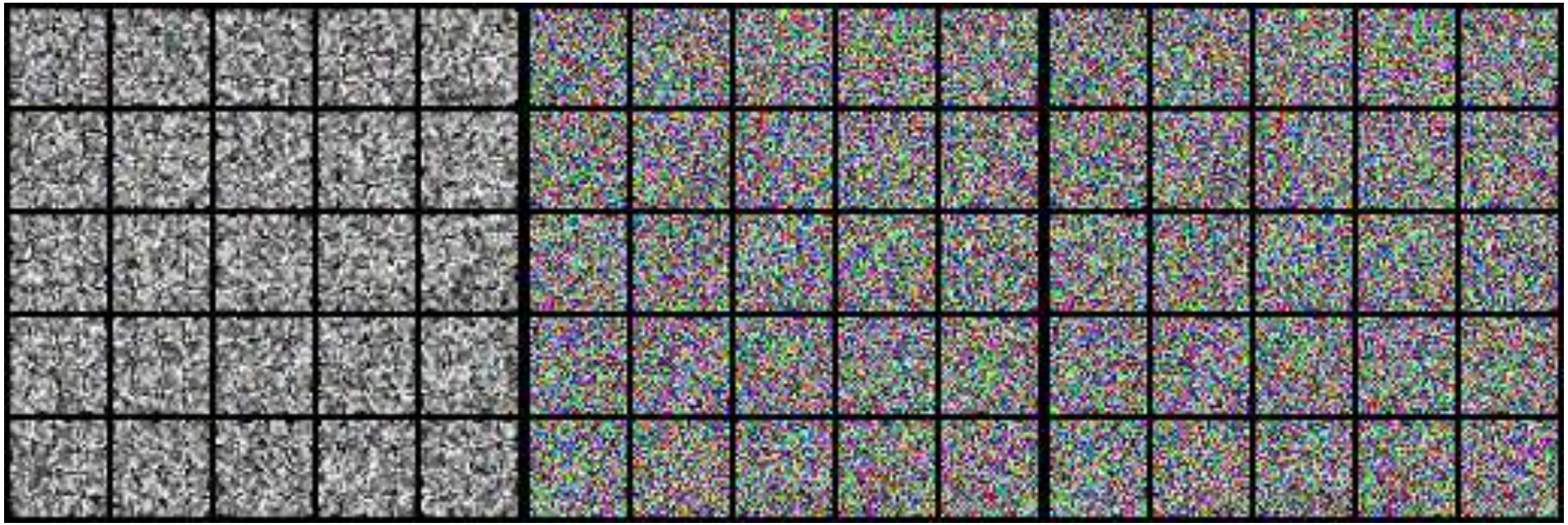
Score matching loss

Annealed Langevin dynamics



$s_\theta(\mathbf{x}, \sigma_1)$ $s_\theta(\mathbf{x}, \sigma_2)$ $s_\theta(\mathbf{x}, \sigma_3)$

Experiments: Sampling



Example: Unconditional Generation on CIFAR-10

noise
x_T

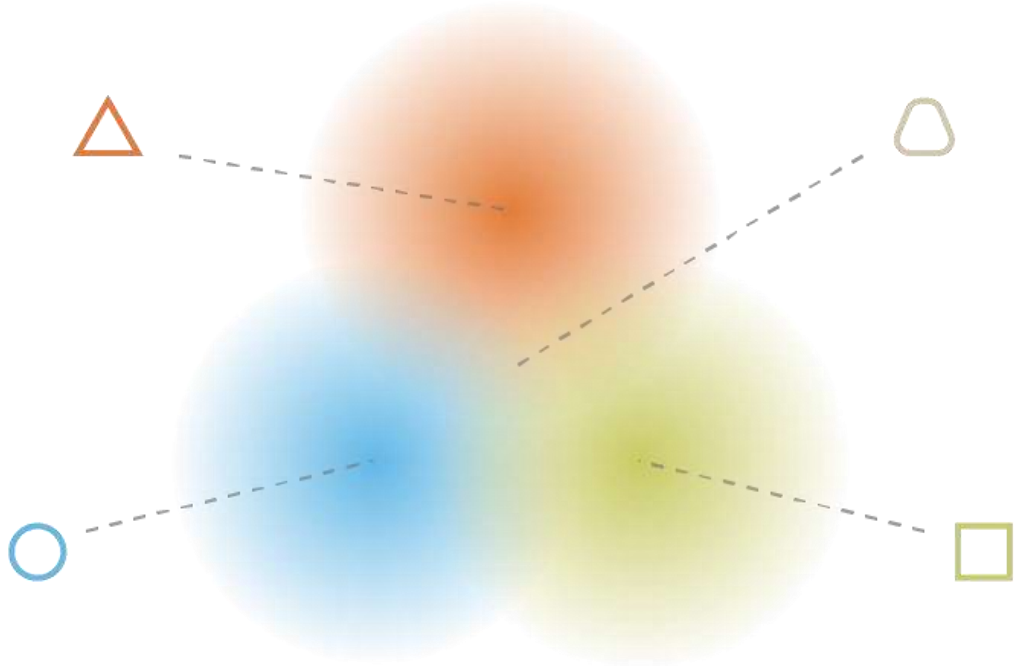
generated
x_0

[Ho et al, 2020]

Example: shared intermediate latents

[Ho et al, 2020]

Problems

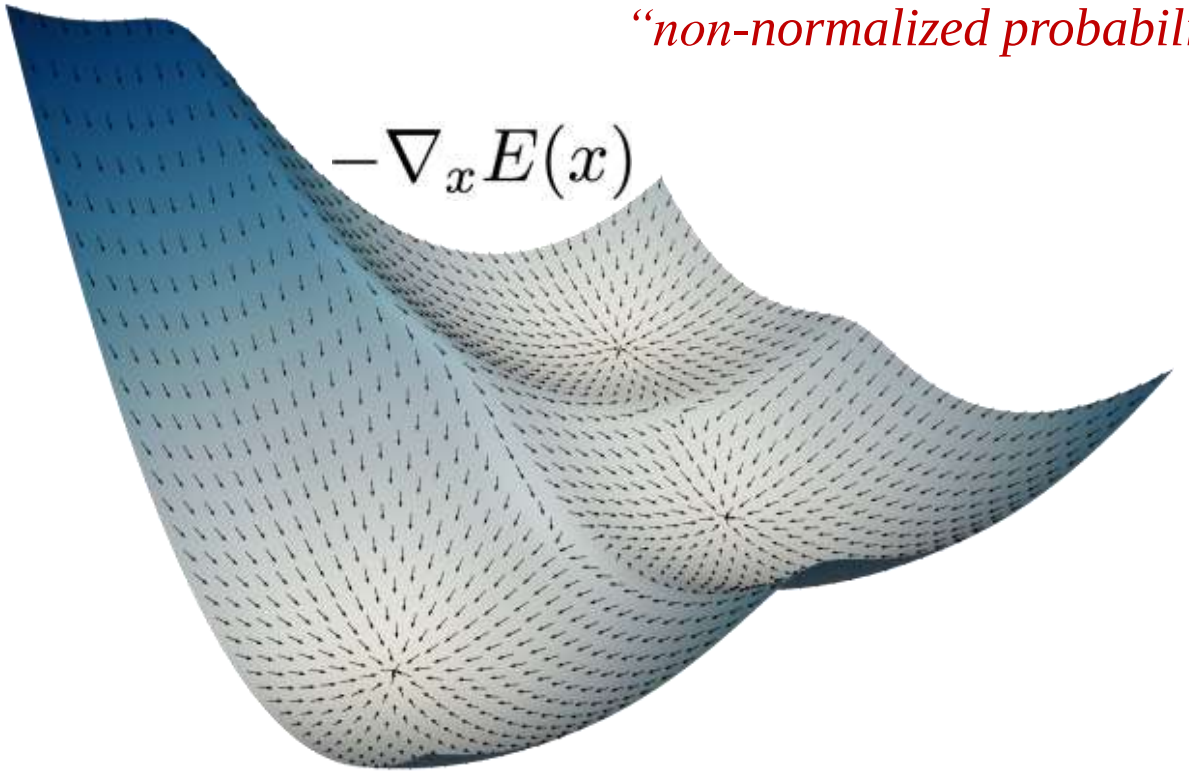


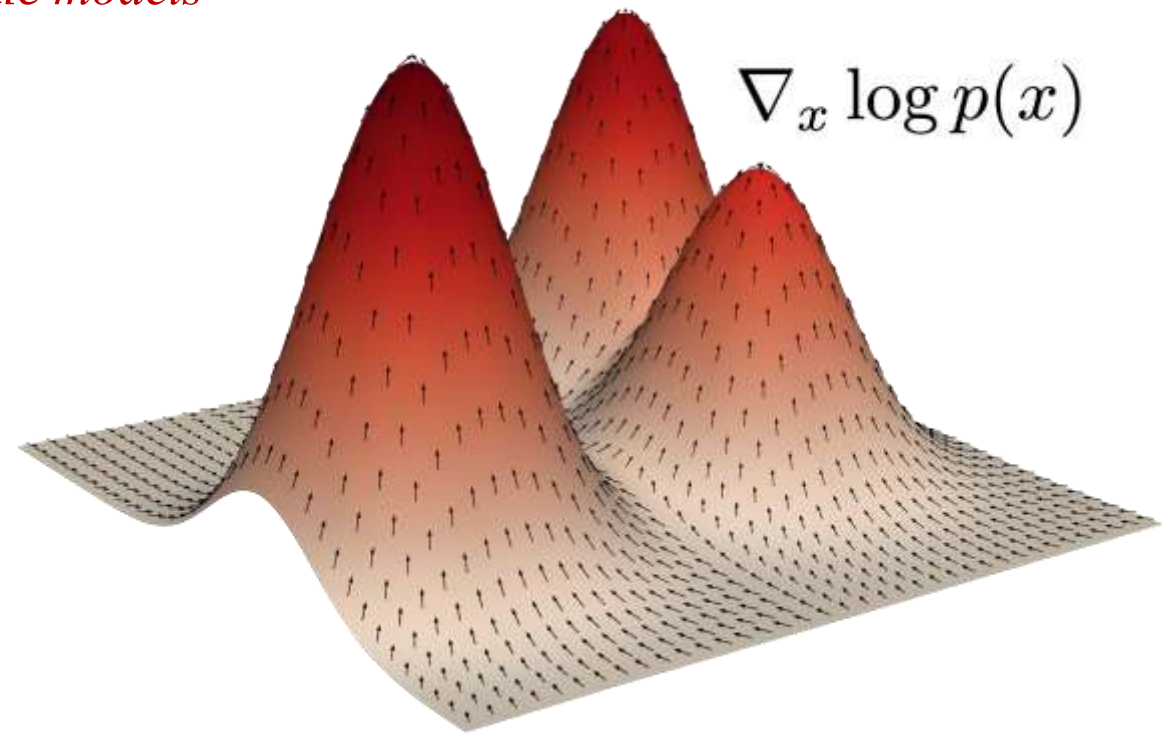
Energy-based Models

- **“Score function”**: gradient of log-probability

$$\nabla_x \log p(x) = -\nabla_x E(x)$$

“non-normalized probabilistic models”

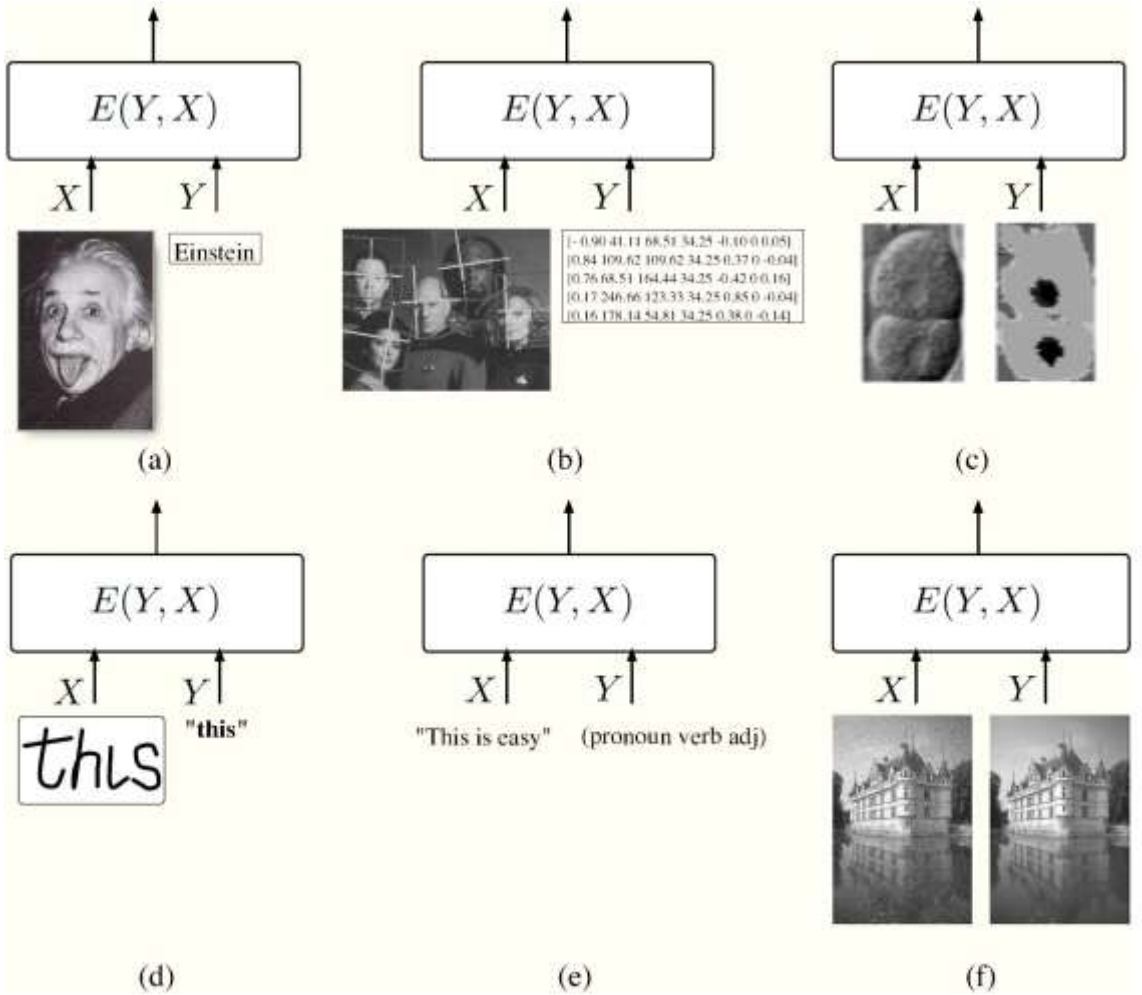




*only visualize directions

More about Energy-based Models ...

- At <u>inference</u> time, find a solution that minimizes energy



LeCun et al., “A Tutorial on Energy-Based Learning”, 2006

Various Perspectives on Diffusion Models ...

- Hierarchical VAE
- Energy-based Models and Score Matching
- Autoregressive models

> be a fully expressive conditional distribution. With these choices, $D_{\mathrm{KL}}(q(\mathbf{x}_T) \parallel p(\mathbf{x}_T)) = 0$, and minimizing $D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))$ trains p_θ to copy coordinates $t+1, \ldots, T$ unchanged and to predict the t^{th} coordinate given $t+1, \ldots, T$. Thus, training p_θ with this particular diffusion is training an autoregressive model.
>
> [Ho et al, 2020]

- SDE and ODE
- Normalizing Flows
- Recurrent Neural Networks

See also: Dieleman, “Perspectives on diffusion”. https://sander.ai/2023/07/20/perspectives.html