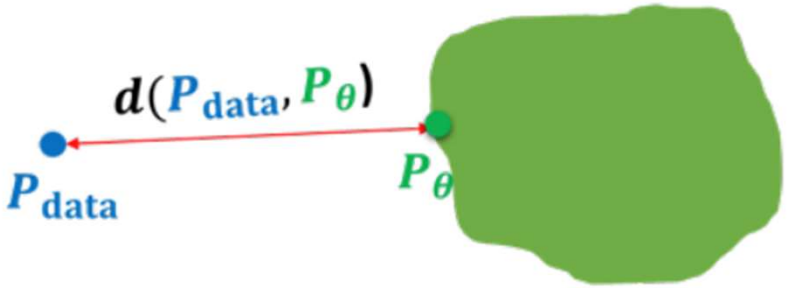
Score-Based Models

LECTURE 14

CS236: Deep Generative Models

Summary

$\mathbf{x}_i \sim P_{\text{data}}$
$i = 1, 2, \ldots, n$



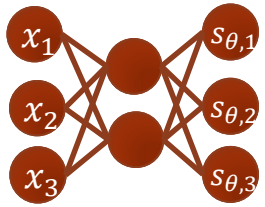
$\theta \in M$

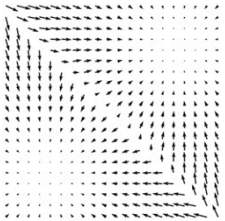
Model family

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

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

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

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

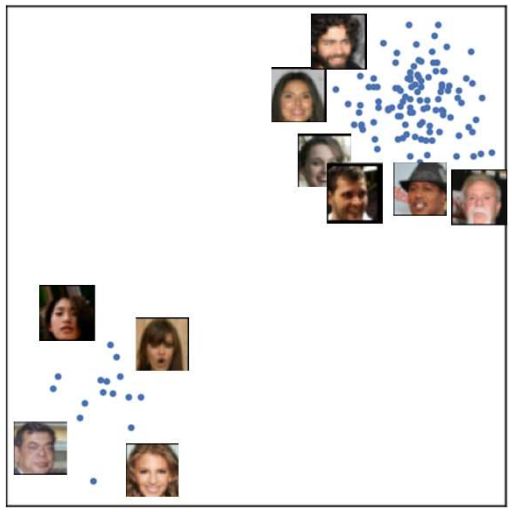
Sliced score matching

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

$$= E_{\mathbf{v}\sim p_{\mathbf{v}}, \mathbf{x}\sim p_{\text{data}}} \left[\mathbf{v}^\intercal \nabla_{\mathbf{x}} \boldsymbol{s}_\theta(\mathbf{x}) \mathbf{v} + \frac{1}{2} (\mathbf{v}^\intercal \boldsymbol{s}_\theta(\mathbf{x}))^2 \right] + \text{const.}$$

- Projection distribution $p_{\mathbf{v}}$ can be Gaussian or Rademacher
- **Pros:**
  - Much more scalable than score matching;
  - Estimates the true data score.
- **Con**: Slower than denoising score matching.

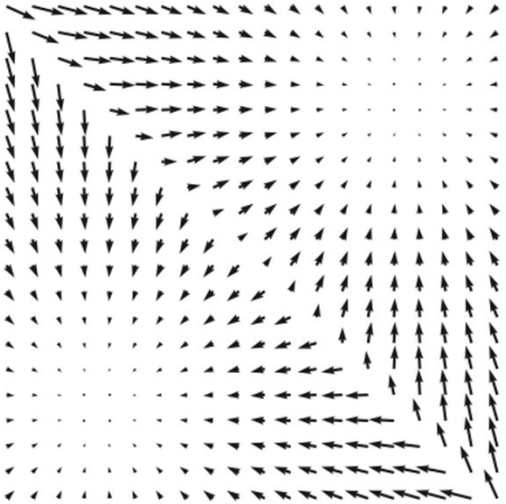
Score-based generative modeling



Data samples

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

score matching



Scores

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

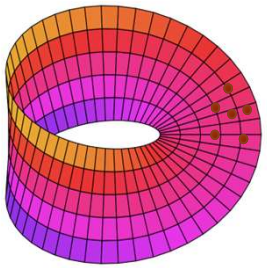
Langevin dynamics

New samples

Pitfalls

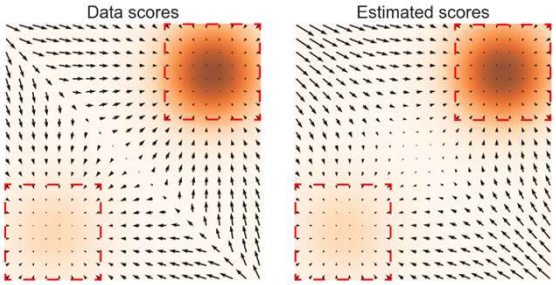
- Manifold hypothesis. Data score is undefined.

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

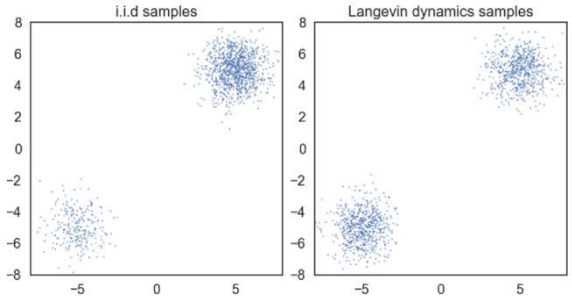


Data points

- Score matching fails in low data density regions

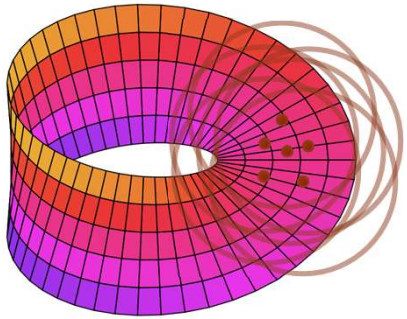


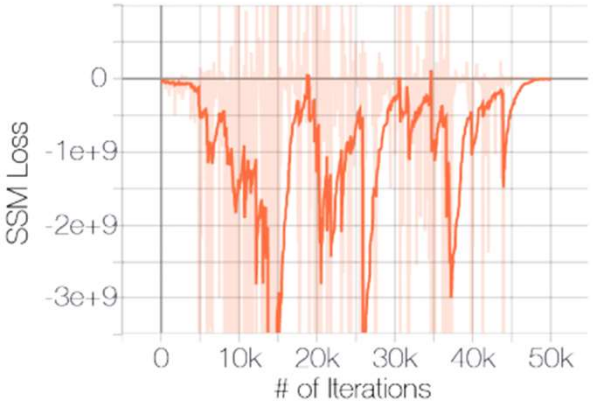
- Langevin dynamics converges very slowly



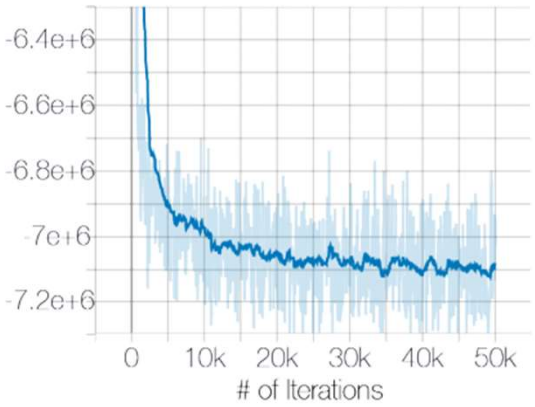
Gaussian perturbation

- The solution to all pitfalls: **Gaussian perturbation!**
- Manifold + noise
- Score matching on noisy data.





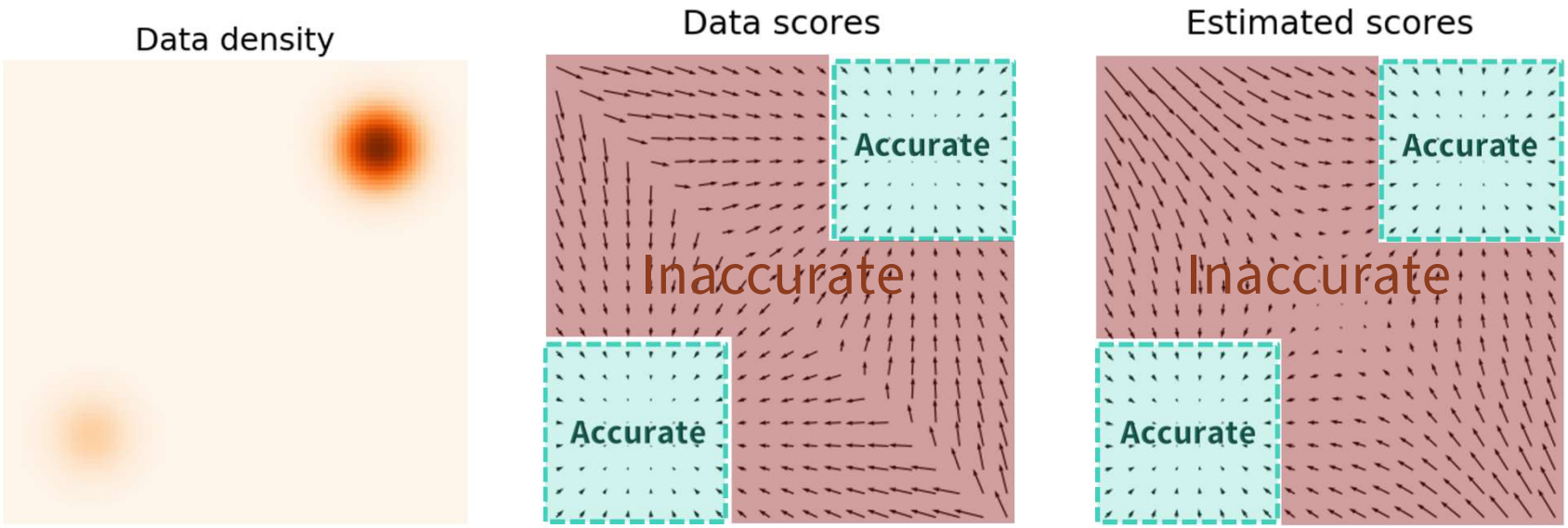
CIFAR-10



Noisy CIFAR-10

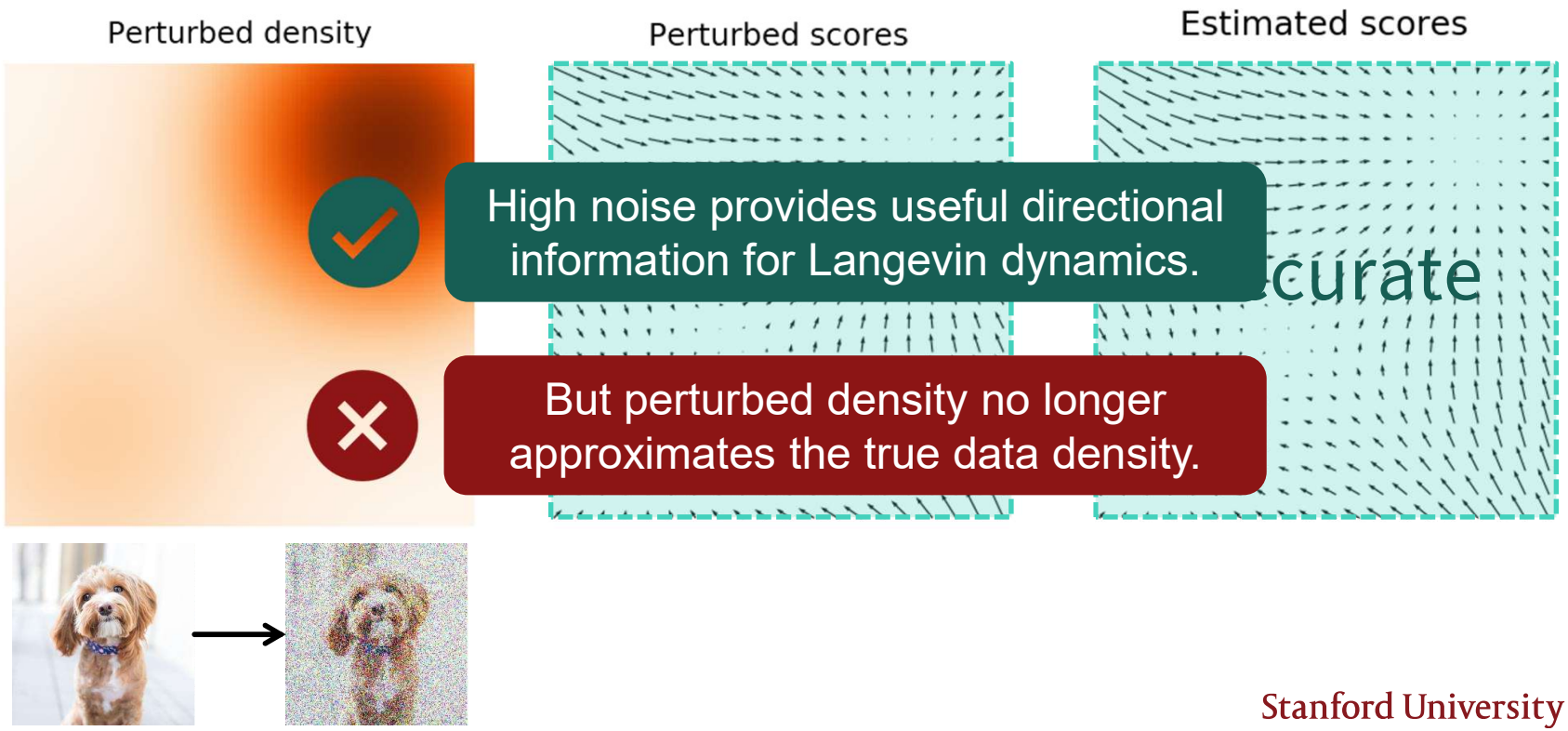
$$\mathcal{N}(0; 0.0001)$$

Challenge in low data density regions



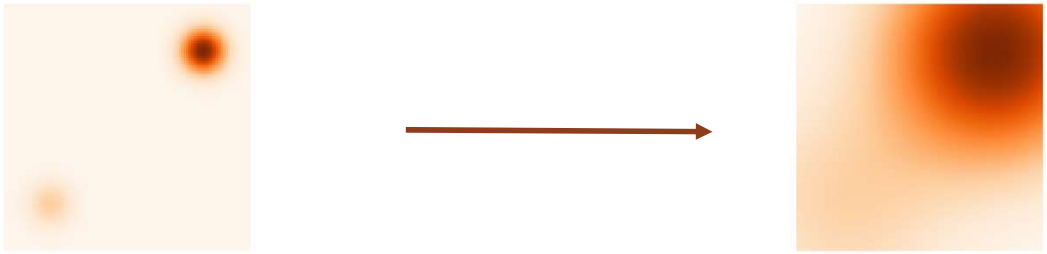
Song and Ermon. “Generative Modeling by Estimating Gradients of the Data Distribution.” NeurIPS 2019.

Improving score estimation by adding noise



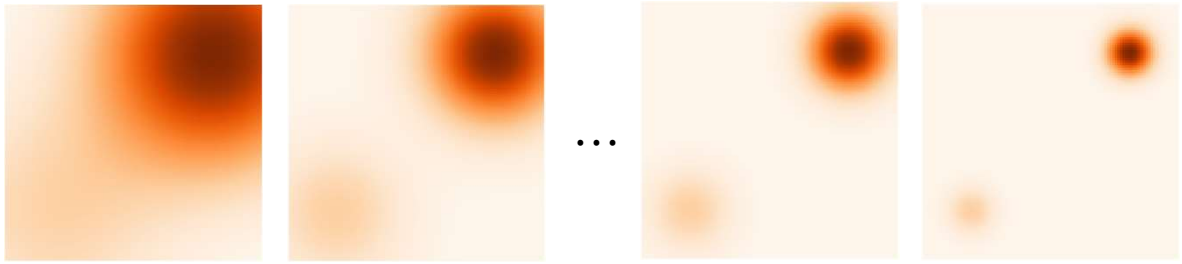
Multi-scale Noise Perturbation

- How much noise to add?

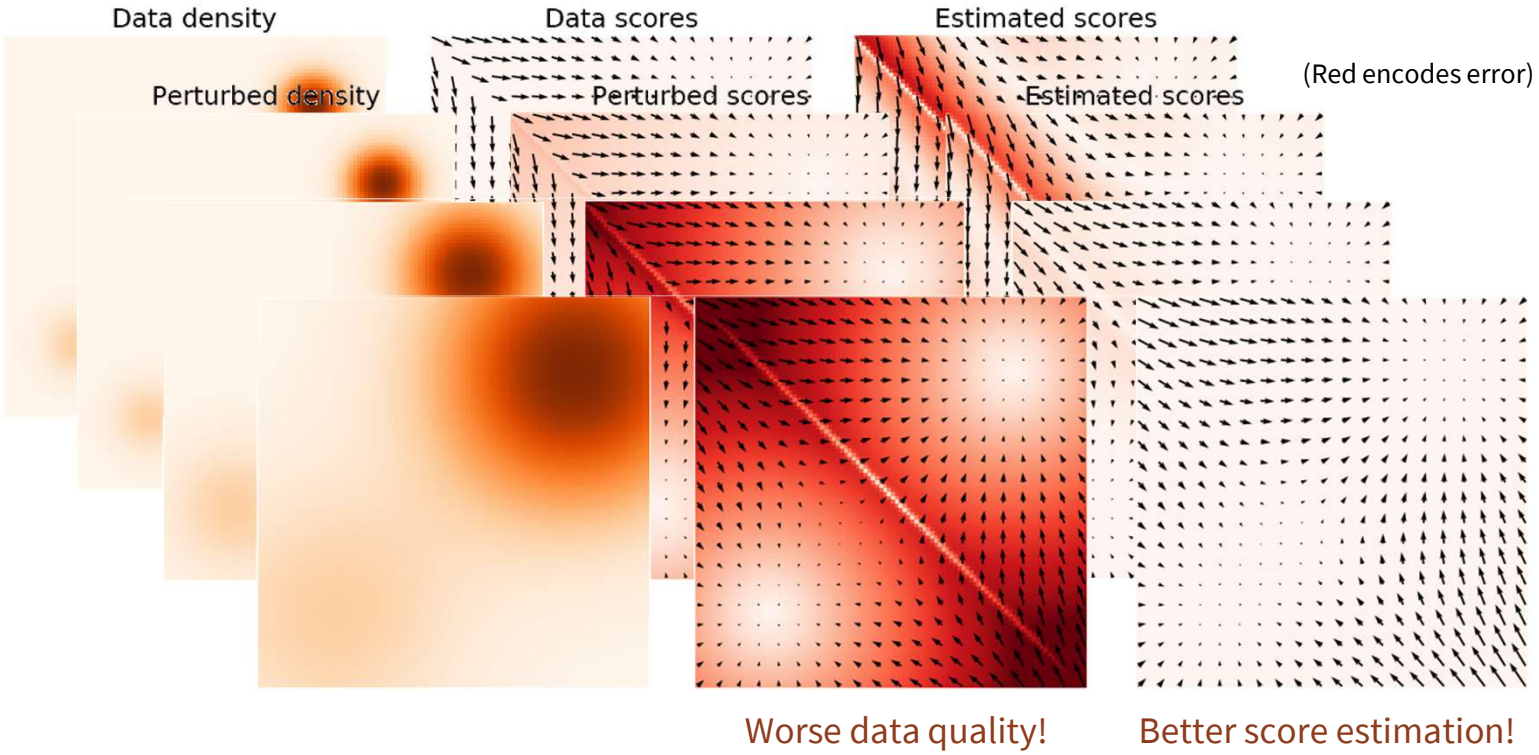


- Multi-scale noise perturbations.

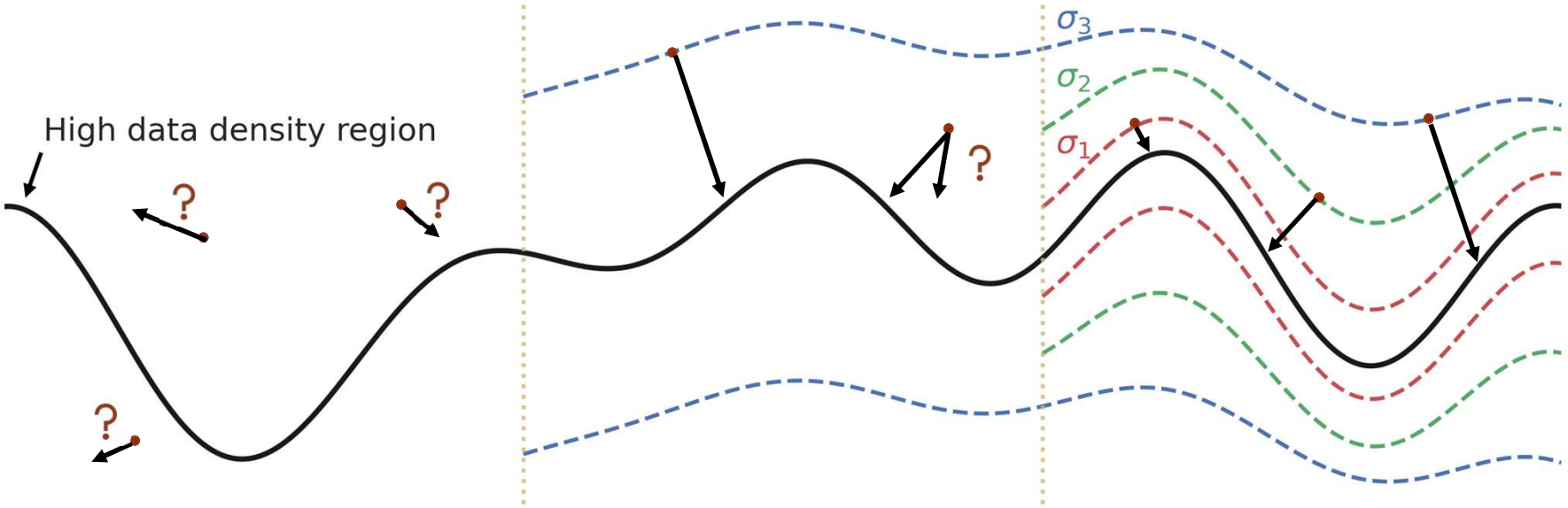
$$\sigma_1 > \sigma_2 > \cdots > \sigma_{L-1} > \sigma_L$$



Trading off Data Quality and Estimation Accuracy

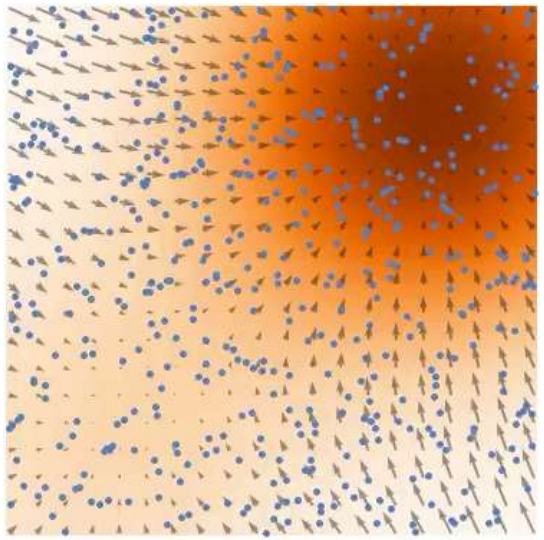


Using multiple noise scales

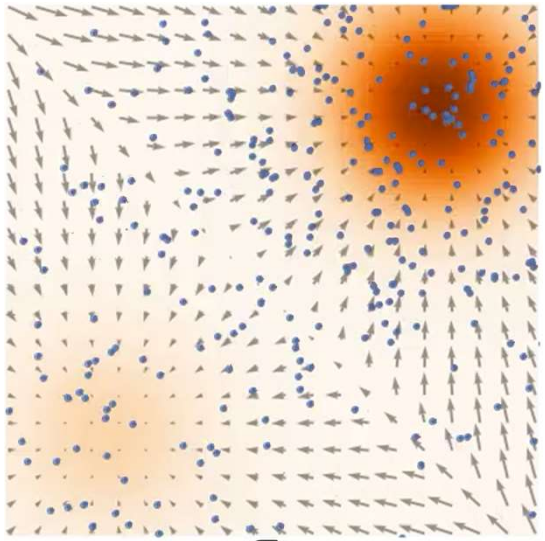


Annealed Langevin Dynamics: Joint Scores to Samples

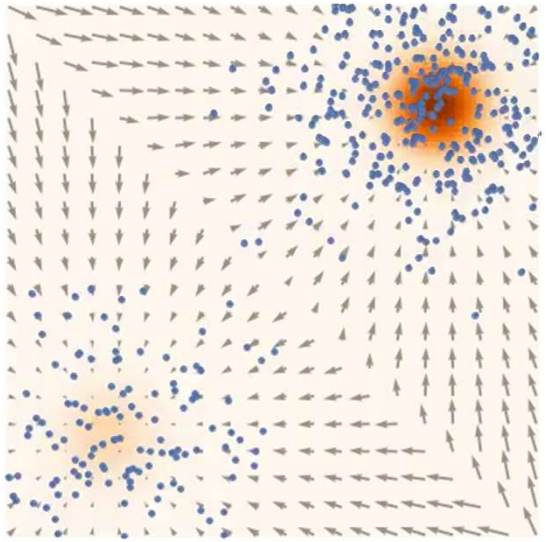
- Sample using $\sigma_1, \sigma_2, \cdots, \sigma_L$ sequentially with Langevin dynamics.
- Anneal down the noise level.
- Samples used as initialization for the next level.



σ_1

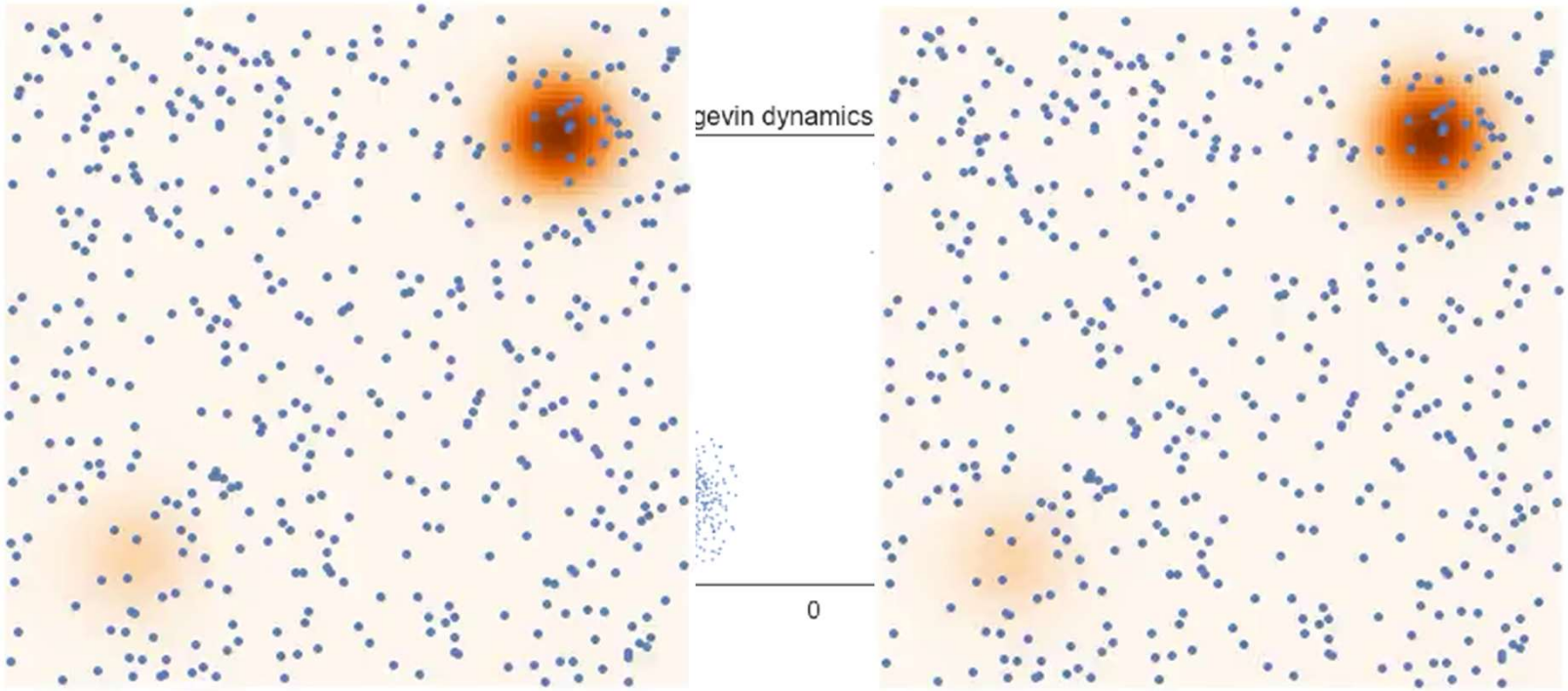


σ_2



σ_3

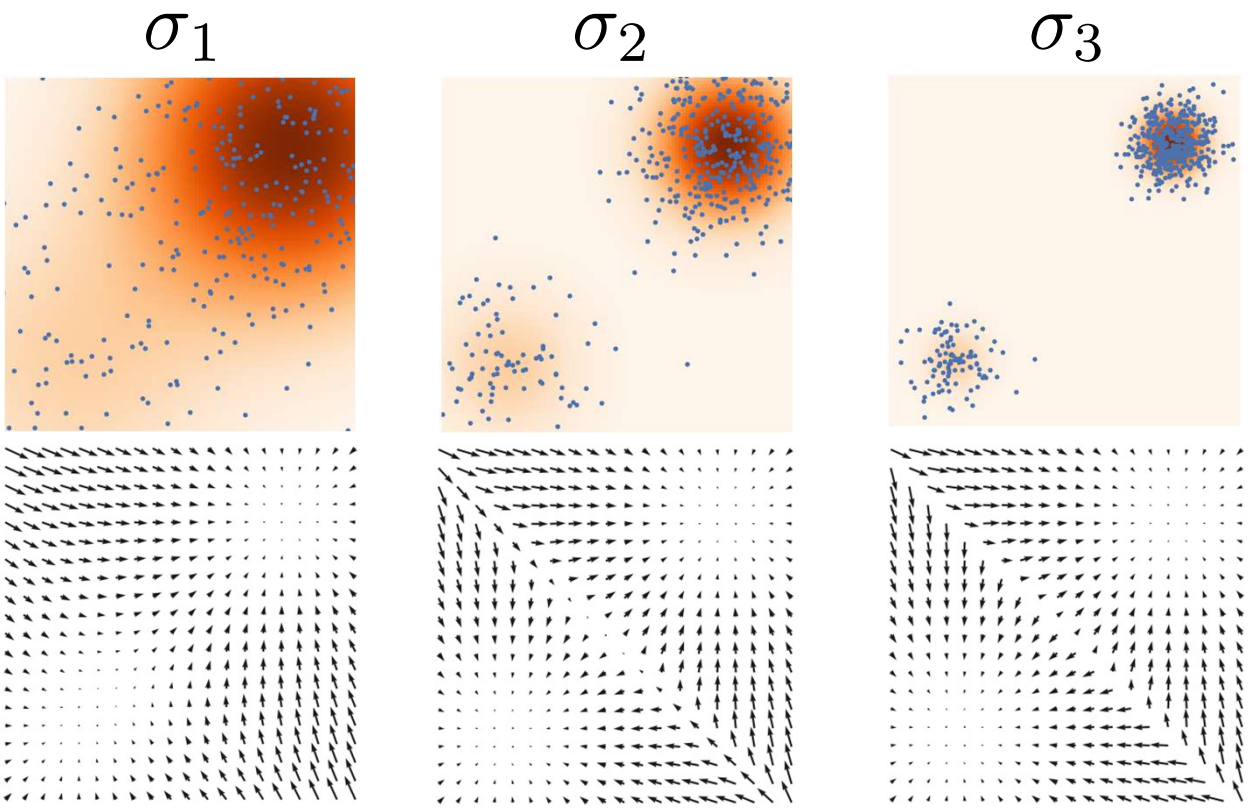
Comparison to the vanilla Langevin dynamics

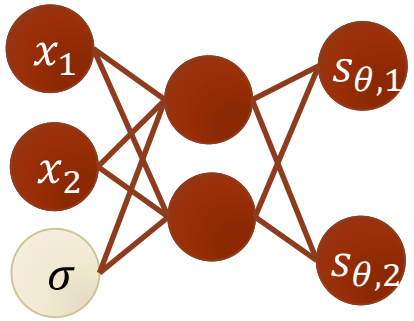


Langevin dynamics

Annealed Langevin dynamics

Joint Score Estimation via Noise Conditional Score Networks





Noise Conditional Score Network (NCSN)

Stanford University

Training noise conditional score networks

- Which score matching loss?
  - Sliced score matching?
  - Denoising score matching?
- Denoising score matching is naturally suitable, since the goal is to estimate the score of perturbed data distributions.
- Weighted combination of denoising score matching losses

$$\frac{1}{L}\sum_{i=1}^{L}\lambda(\sigma_i)E_{q_{\sigma_i}(\mathbf{x})}[\|\nabla_{\mathbf{x}}\log q_{\sigma_i}(\mathbf{x})-\boldsymbol{s}_\theta(\mathbf{x},\sigma_i)\|_2^2]$$

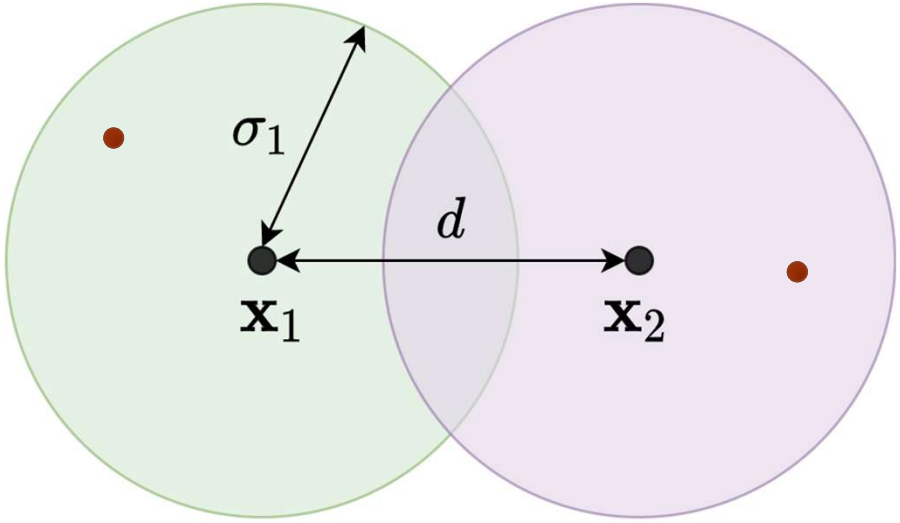
$$=\frac{1}{L}\sum_{i=1}^{L}\lambda(\sigma_i)E_{\mathbf{x}\sim p_{\text{data}},\mathbf{z}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})}[\|\nabla_{\tilde{\mathbf{x}}}\log q_{\sigma_i}(\tilde{\mathbf{x}}\mid\mathbf{x})-\boldsymbol{s}_\theta(\tilde{\mathbf{x}},\sigma_i)\|_2^2]+\text{const.}$$

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

Choosing noise scales

- Maximum noise scale

$$\sigma_1 \approx \text{maximum pairwise distance between datapoints}$$



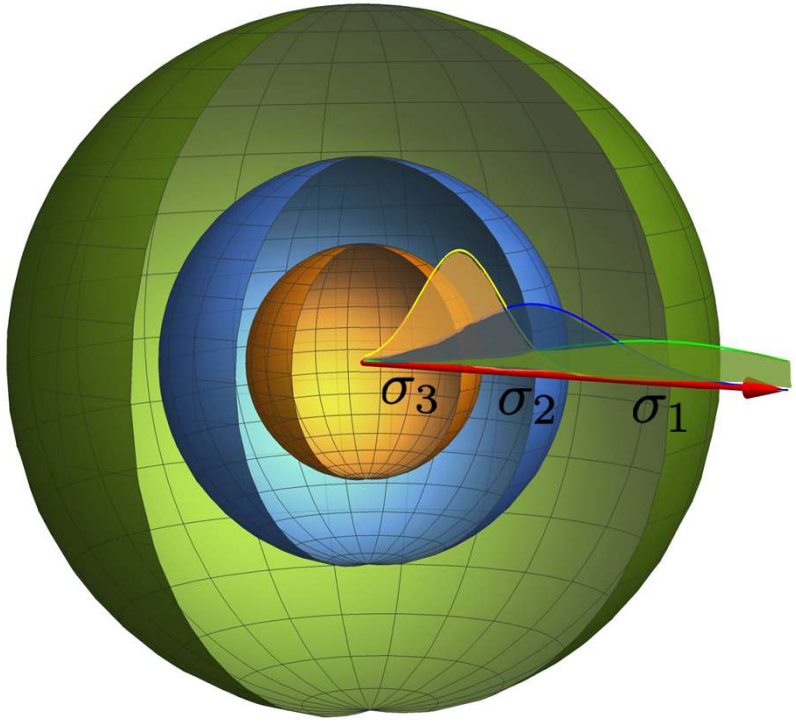
- Minimum noise scale: σ_L should be sufficiently small so that noise in final samples is negligible.

Choosing noise scales

- **Key intuition:** adjacent noise scales should have sufficient overlap to facilitate transitioning across noise scales in annealed Langevin dynamics.
- A geometric progression with sufficient length.

$$\sigma_1 > \sigma_2 > \sigma_3 > \cdots > \sigma_{L-1} > \sigma_L$$

$$\frac{\sigma_1}{\sigma_2} = \frac{\sigma_2}{\sigma_3} = \cdots = \frac{\sigma_{L-1}}{\sigma_L}$$



Choosing the weighting function

- Weighted combination of denoising score matching losses

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

- How to choose the weighting function $\lambda : \mathbb{R}_{>0} \rightarrow \mathbb{R}_{>0}$?
- **Goal:** balancing different score matching losses in the sum → $\lambda(\sigma_i) = \sigma_i^2$

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

Training noise conditional score networks

- Sample a mini-batch of datapoints $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \sim p_{\text{data}}$
- Sample a mini-batch of noise scale indices

$$\{i_1, i_2, \cdots, i_n\} \sim \mathcal{U}\{1, 2, \cdots, L\}$$

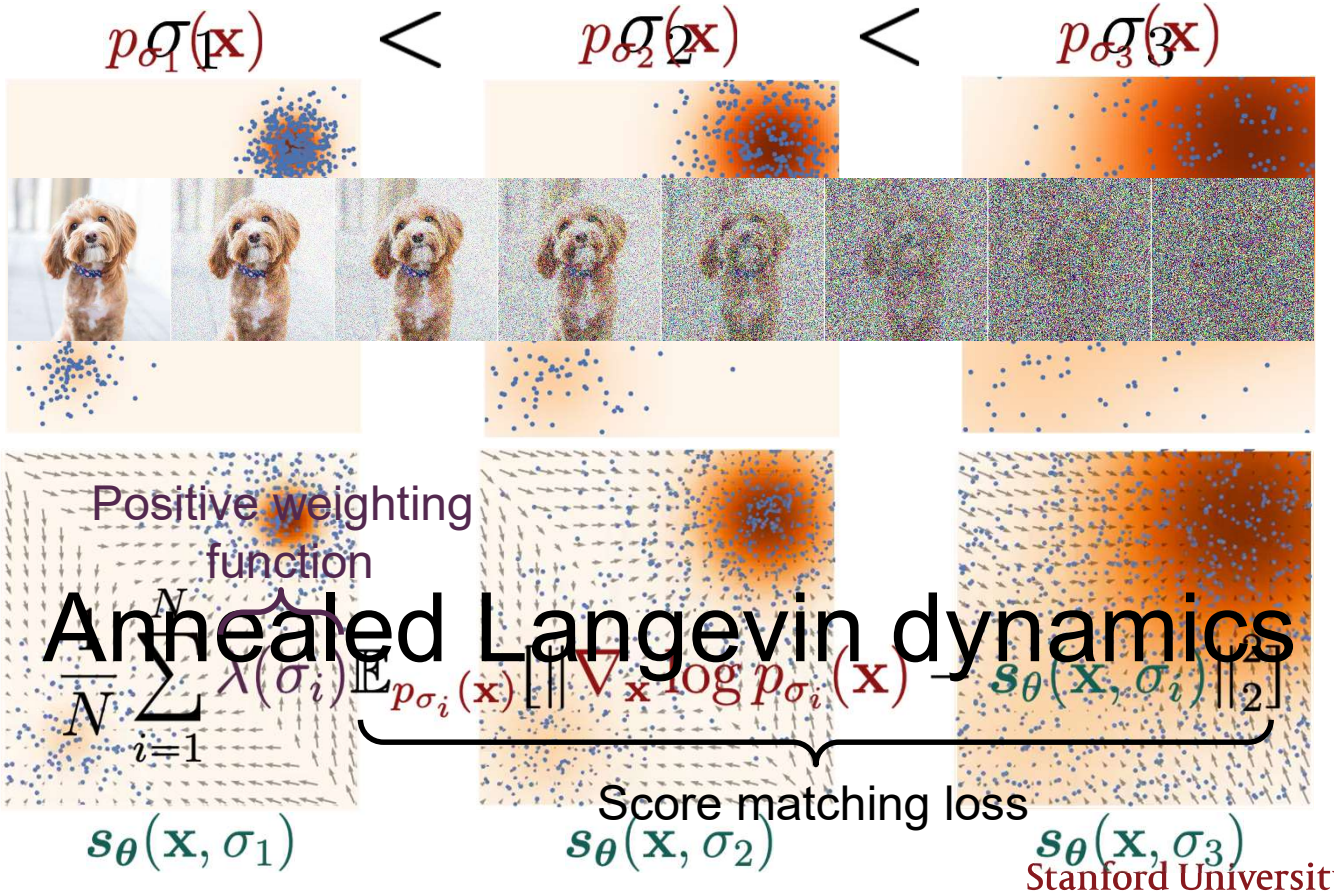
- Sample a mini-batch of Gaussian noise $\{\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n\} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
- Estimate the weighted mixture of score matching losses

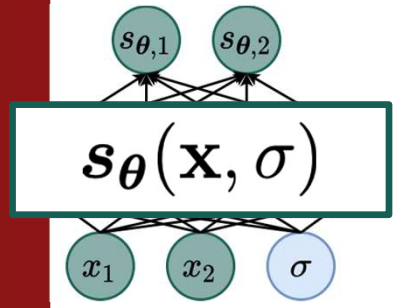
$$\frac{1}{n} \sum_{k=1}^{n} \left[\left\| \sigma_{i_k} \boldsymbol{s}_\theta(\mathbf{x}_k + \sigma_{i_k} \mathbf{z}_k, \sigma_{i_k}) + \mathbf{z}_k \right\|_2^2 \right]$$

- Stochastic gradient descent
- As efficient as training one single non-conditional score-based model

Using multiple noise levels

Data

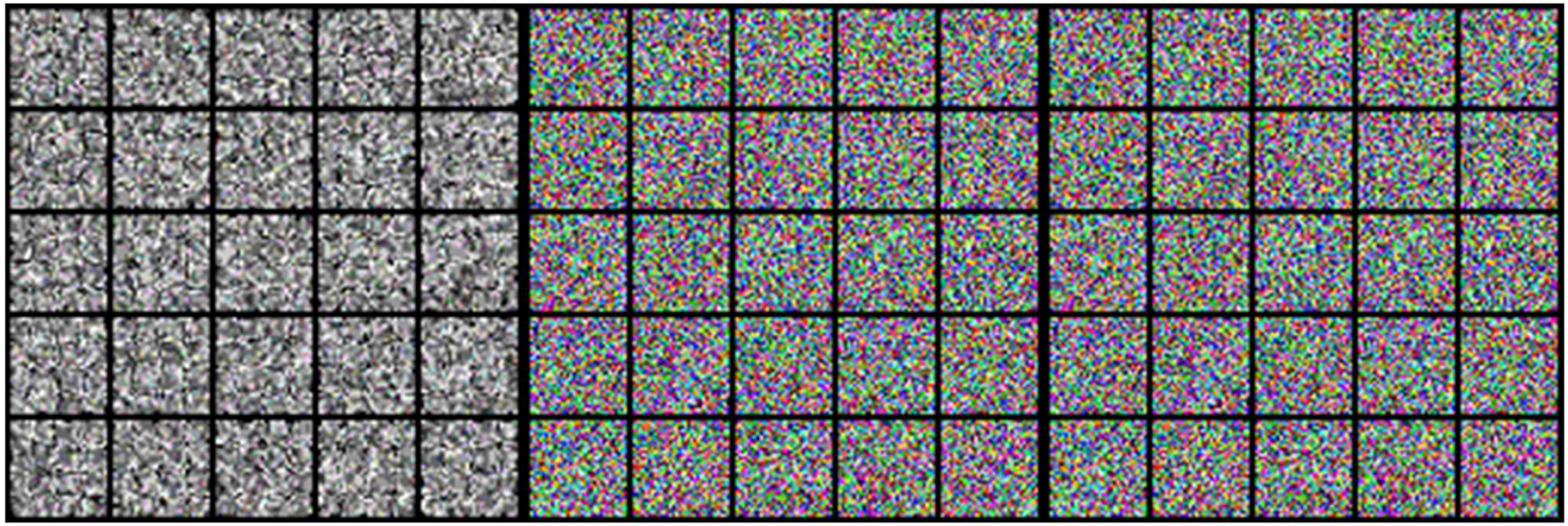




Noise Conditional Score Model

Annealed Langevin dynamics

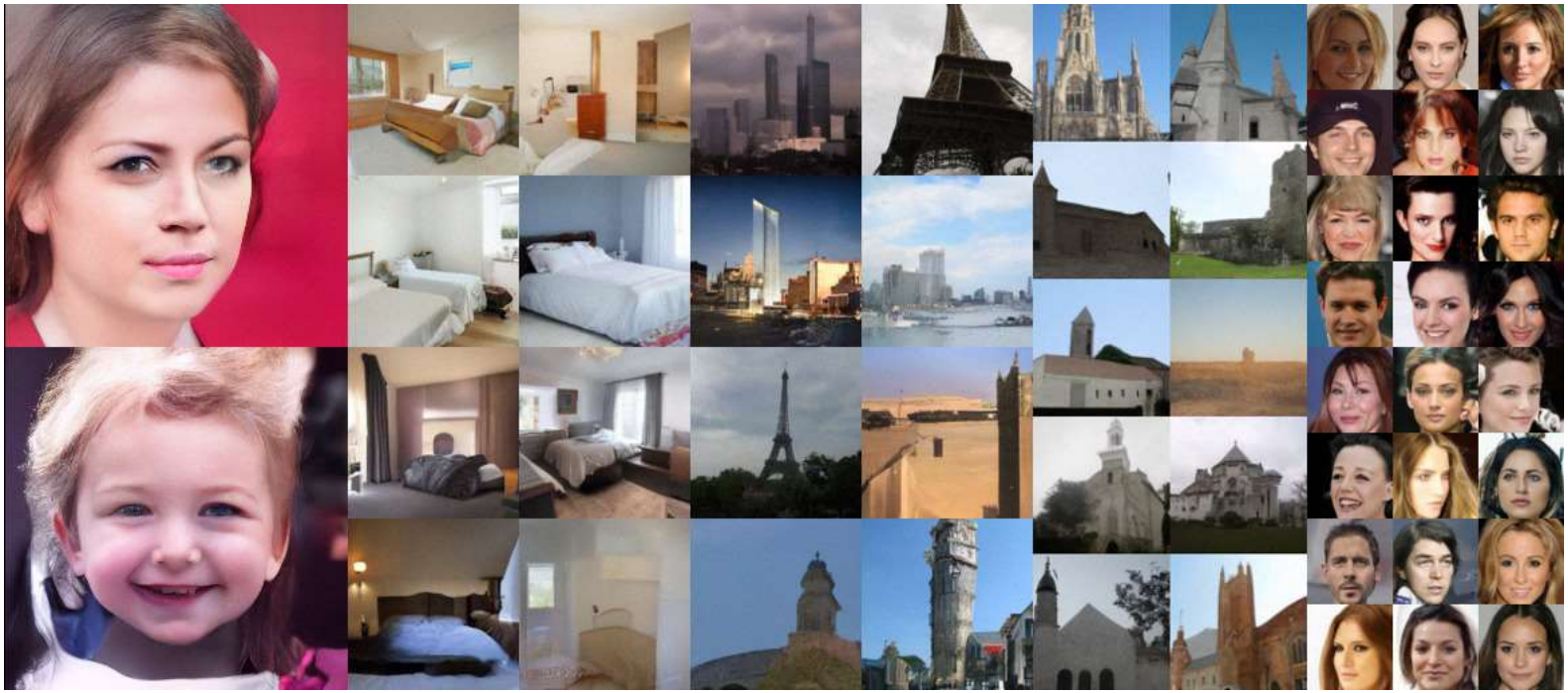
Experiments: Sampling



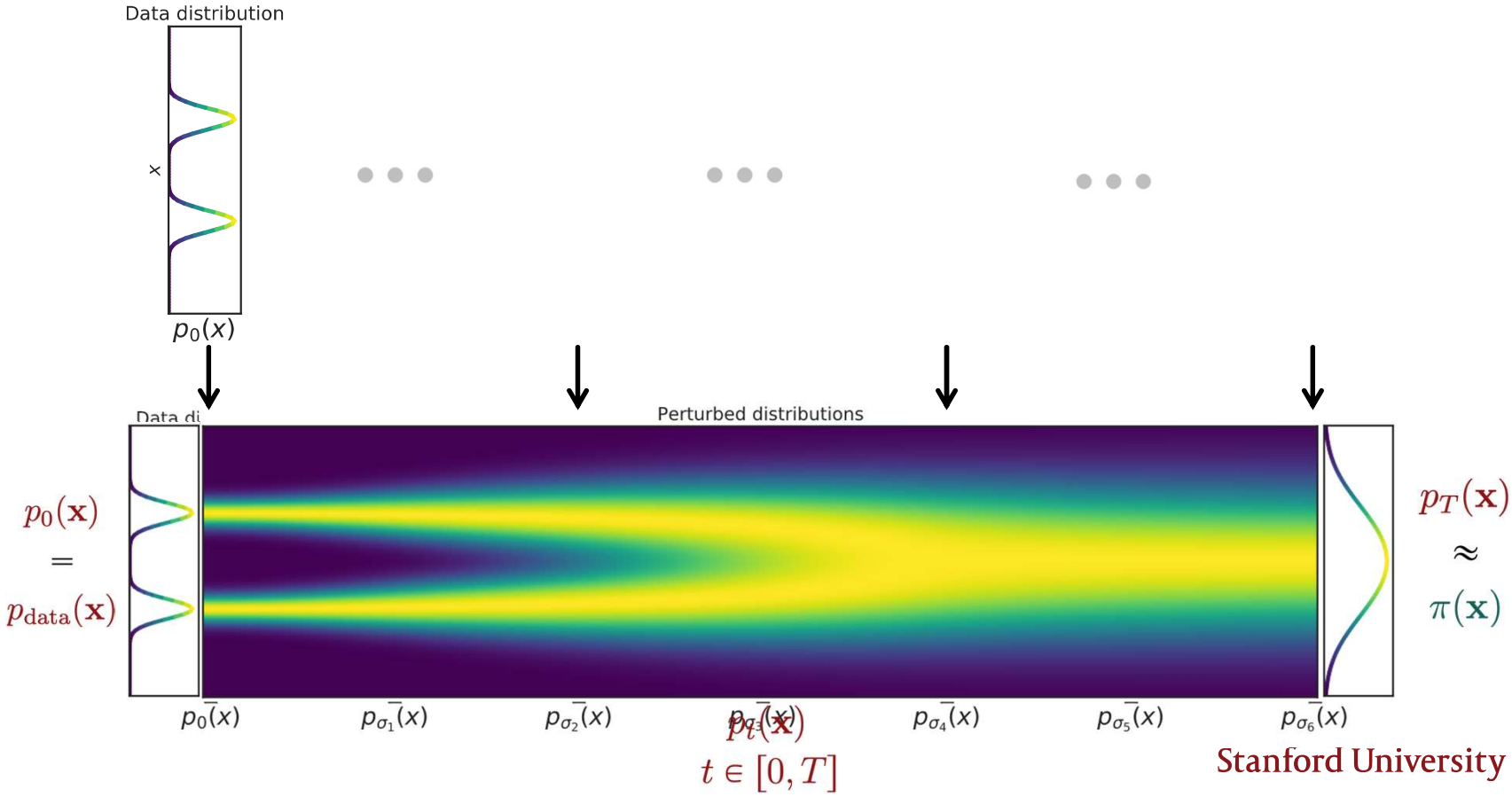
Experiments: sampling

Model	Inception	FID
CIFAR-10 Unconditional		
PixelCNN [59]	4.60	65.93
PixelIQN [42]	5.29	49.46
EBM [12]	6.02	40.58
WGAN-GP [18]	$7.86 \pm .07$	36.4
MoLM [45]	$7.90 \pm .10$	**18.9**
SNGAN [36]	$8.22 \pm .05$	21.7
ProgressiveGAN [25]	$8.80 \pm .05$	-
NCSN (Ours)	$\mathbf{8.87} \pm .12$	25.32

High Resolution Image Generation

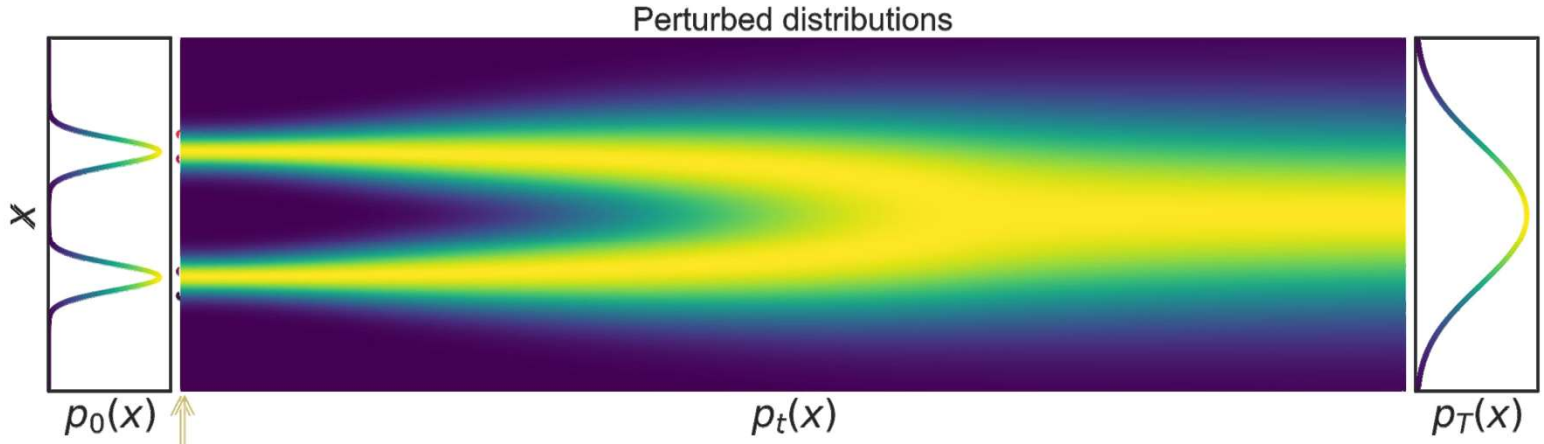


Infinite noise levels

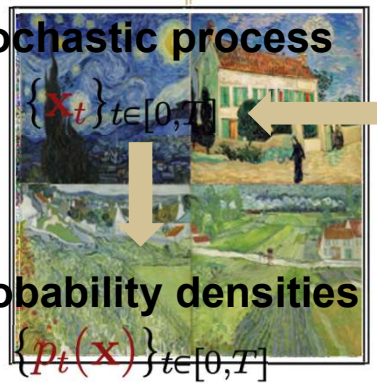


Stanford University

Perturbing data with stochastic processes



Stochastic process



Probability densities

Stochastic differential equation (SDE)

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

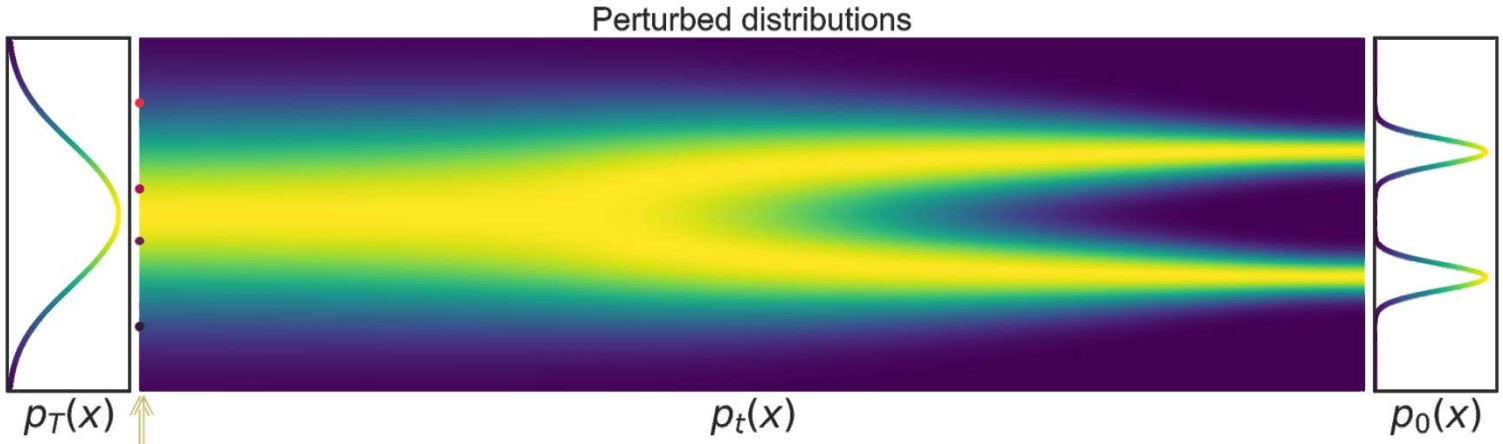
Deterministic drift — Infinitesimal noise

WLOG: Toy SDE

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

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

Generation via reverse stochastic processes



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

Forward SDE (t: 0→T)

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

Reverse SDE (t: T→0)

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

Infinitesimal noise in the reverse time direction

Score function!

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)

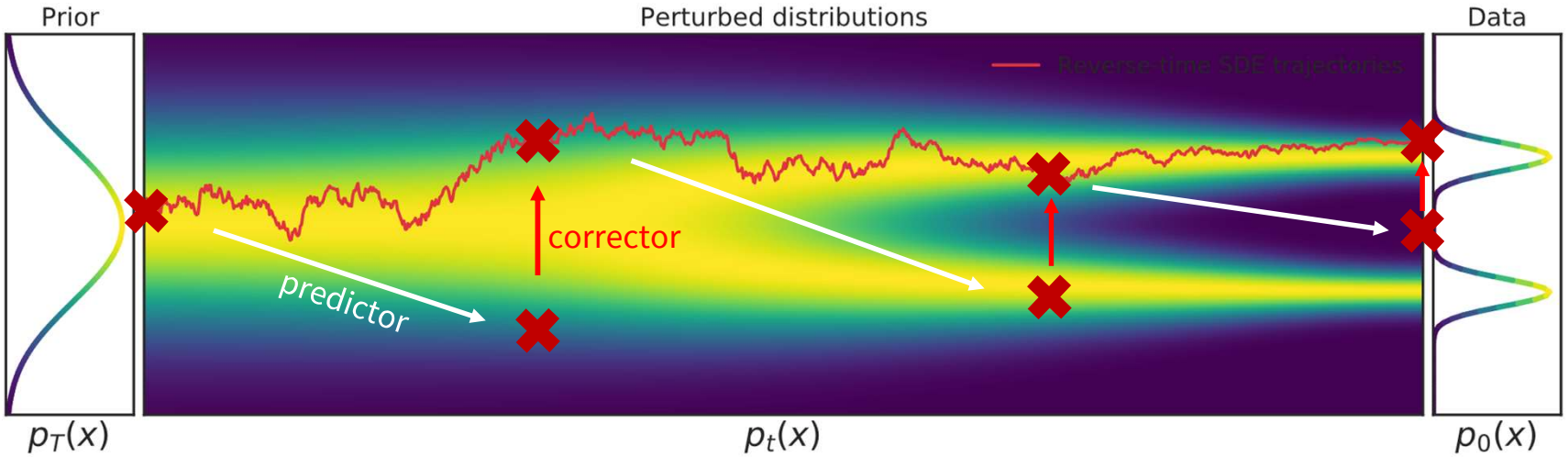
$$\mathbf{x} \leftarrow \mathbf{x} - \sigma(t)^2 \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t) \Delta t + \sigma(t)\ \mathbf{z} \quad (\mathbf{z} \sim \mathcal{N}(\mathbf{0}, |\Delta t|\ \mathbf{I}))$$

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

Song, Sohl-Dickstein, Kingma, Kumar, Ermon, Poole. “Score-Based Generative Modeling through Stochastic Differential Equations.” ICLR 2021.

Predictor-Corrector sampling methods

- Predictor-Corrector sampling.
  - **Predictor:** Numerical SDE solver
  - **Corrector:** Score-based MCMC



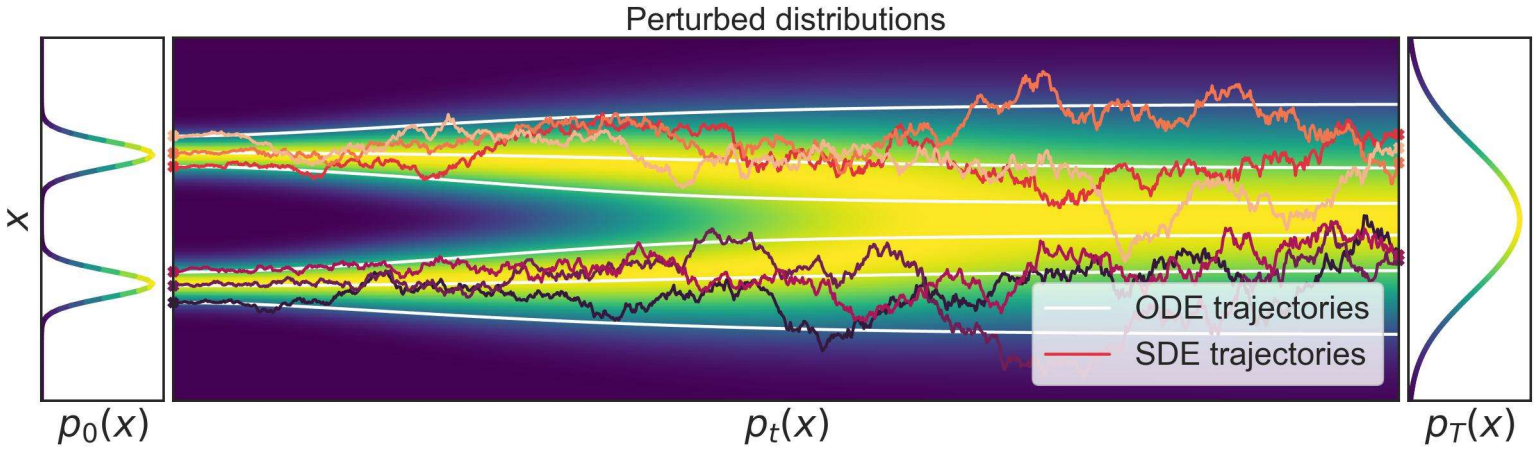
Results on predictor-corrector sampling

Model	FID↓	IS↑
Conditional		
BigGAN (Brock et al., 2018)	14.73	9.22
StyleGAN2-ADA (Karras et al., 2020a)	**2.42**	**10.14**
Unconditional		
StyleGAN2-ADA (Karras et al., 2020a)	2.92	9.83
NCSN (Song & Ermon, 2019)	25.32	8.87 ± .12
NCSNv2 (Song & Ermon, 2020)	10.87	8.40 ± .07
DDPM (Ho et al., 2020)	3.17	9.46 ± .11
DDPM++	2.78	9.64
DDPM++ cont. (VP)	2.55	9.58
DDPM++ cont. (sub-VP)	2.61	9.56
DDPM++ cont. (deep, VP)	2.41	9.68
DDPM++ cont. (deep, sub-VP)	2.41	9.57
NCSN++	2.45	9.73
NCSN++ cont. (VE)	2.38	9.83
NCSN++ cont. (deep, VE)	**2.20**	**9.89**

Song, Sohl-Dickstein, Kingma, Kumar, Ermon, Poole. “Score-Based Generative Modeling through Stochastic Differential Equations.” ICLR 2021.

High-Fidelity Generation for 1024x1024 Images

Converting the SDE to an ODE



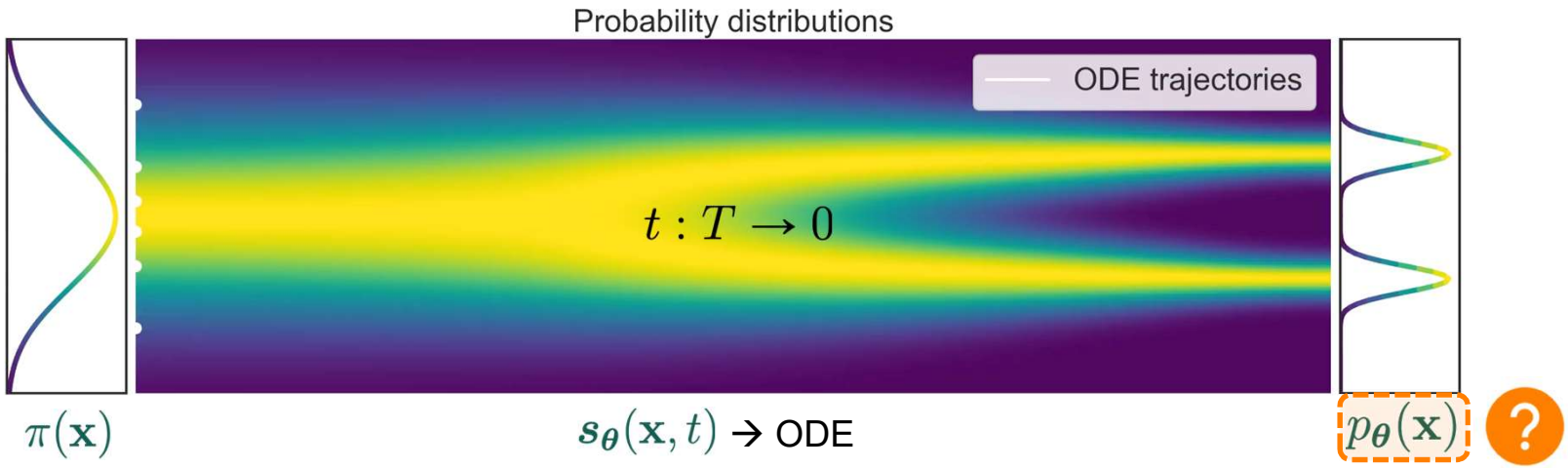
SDE **Ordinary differential equation (ODE)**

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

Score function

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

Evaluating the probabilities with ODEs



Computing the probability density function (change of variables formula)

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

ODE solver

Computable in polynomial time

Unbiased estimator

It is a (continuous-time) normalizing flow model!

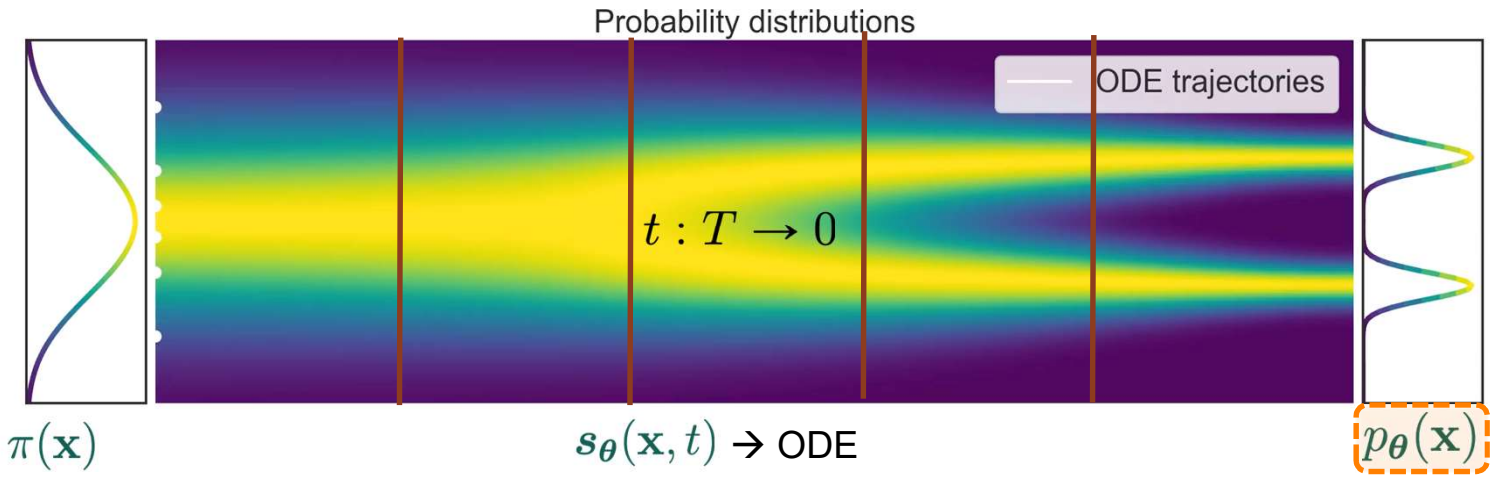
Achieving highest probabilities on test data

Negative log-probability ↓ (**bits/dim**)

Method	CIFAR-10	ImageNet 32x32
PixelSNAIL [Chen et al. 2018]	2.85	3.80
Delta-VAE [Razavi et al. 2019]	2.83	3.77
Sparse Transformer [Child et al. 2019]	2.80	–

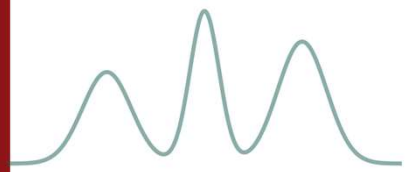
Challenges years of dominance of autoregressive models and VAEs

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

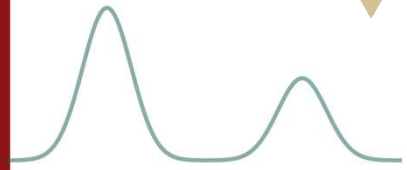
Control the generation process



$p(\mathbf{x})$

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:

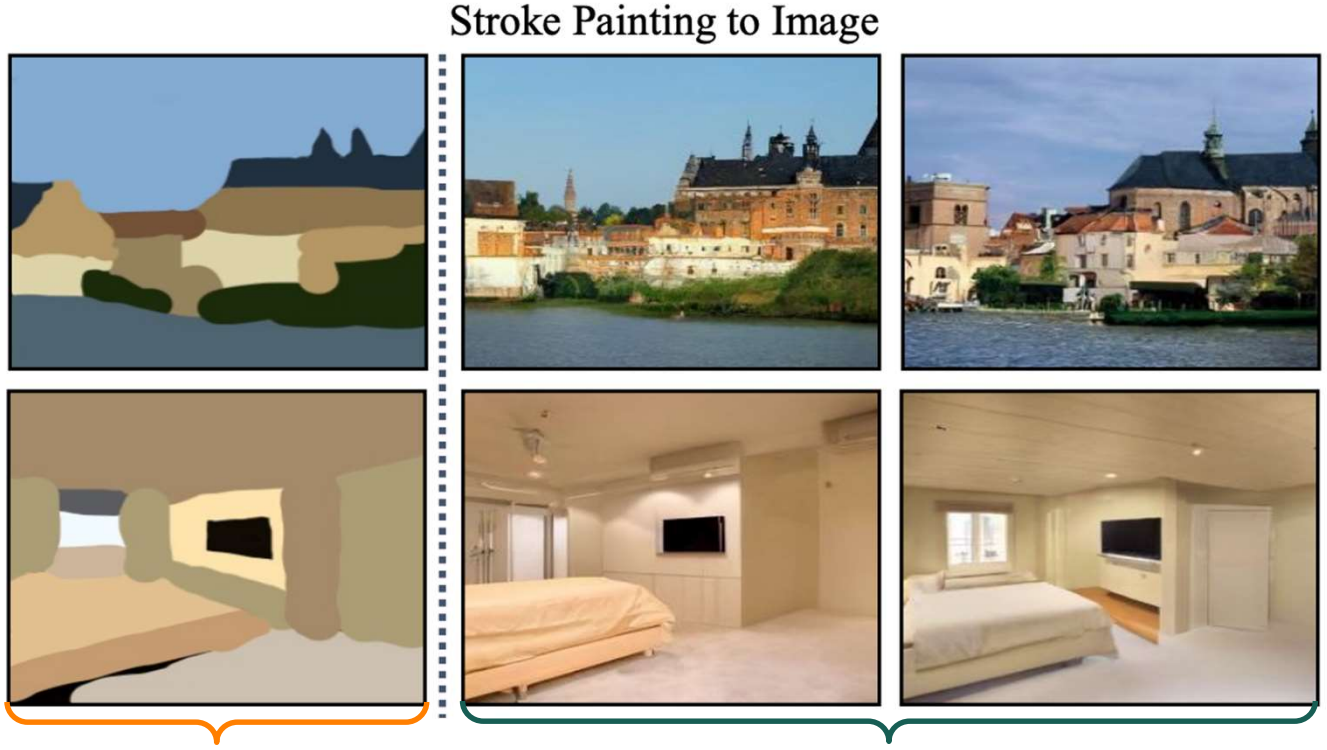
$$\nabla_{\mathbf{x}} \log p(\mathbf{x} \mid \mathbf{y}) = \nabla_{\mathbf{x}} \log p(\mathbf{x}) + \nabla_{\mathbf{x}} \log p(\mathbf{y} \mid \mathbf{x}) - \nabla_{\mathbf{x}} \log p(\mathbf{y})$$

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

$\nabla_{\mathbf{x}} \log p(\mathbf{y})$ → 0

Plug in different forward models for the same score model

Stroke to image synthesis



Stroke paintings

$\mathbf{y}$

Sampled images

$\mathbf{x} \mid \mathbf{y}$

Forward model

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

can be specified.

[Meng, He, Song, Song, Wu, Zhu, Ermon. ICLR 2022]

Language-guided image generation

$\mathbf{y}$

$\mathbf{x} \mid \mathbf{y}$

(Prompt)
Treehouse in the style of Studio Ghibli animation

Forward model

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

is an image captioning neural network.

[Work by @danielrussruss]

Controllable generation: Text-guided generation

An abstract painting of computer science:

A painting of the starry night by van Gogh

https://colab.research.google.com/drive/1FuOobQOmDJuG7rGsMWfQa883A9r4HxEO?usp=sharing

Conclusion

- **Gradients of distributions (scores) can be estimated easily**
  - **Flexible architecture choices** — no need to be normalized/invertible
  - **Stable training** — no minimax optimization
- **Better or comparable sample quality to GANs**
  - State-of-the-art performance on CIFAR-10 and others
  - Scalable to resolution of 1024x1024 for image generation
- **Exact likelihood computation**
  - State-of-the-art likelihood on CIFAR-10
  - Sample editing via manuplating latent codes