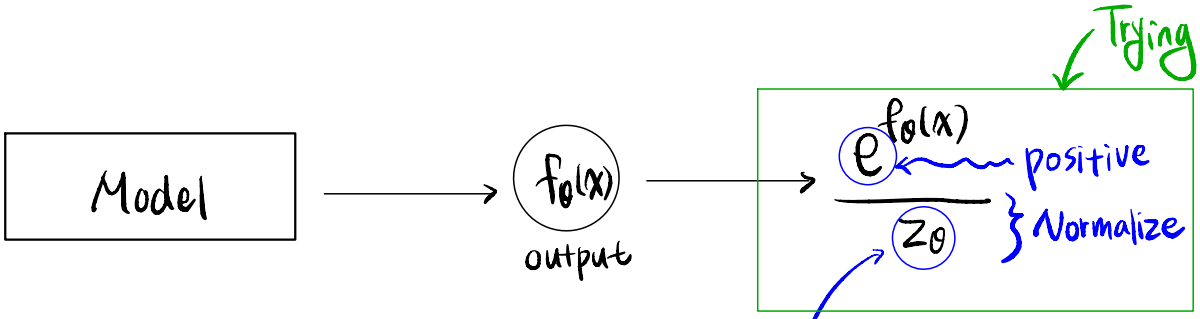
Diffusion and Score-based Generative Models



$$Z_\theta = \int e^{f_\theta(x)} dx$$ ← Complex & intractable

cf. Normalizing constant for Gaussian

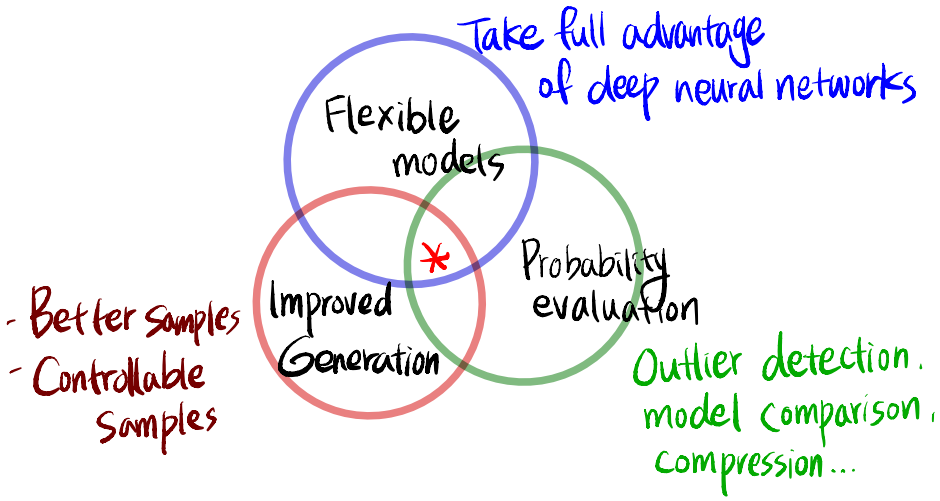
$$Z_\mu = \frac{1}{(2\pi)^{\frac{d}{2}}}$$

⟨Tackling the intractable normalizing constant⟩

- Approximating the normalizing constant
  - Energy-based models ← inaccurate probability evaluation
- Using restricted neural network models
  - Autoregressive models
  - Normalizing flow models
  - Variational auto-encoders

  } ← restricted model family (=limit flexibility)
- Modeling the Generation Process Only
  - Generative Adversarial Networks (GANs) ← cannot evaluate probabilities

<Desiderata of better generative modeling>



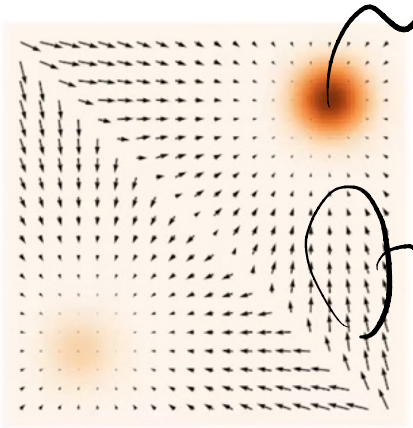
<Our proposal: working with score functions>

$p(x)$

Probability density function

$\nabla_x \log p(x)$

(Stein) score function



Score vs. density function (color coded)

gives direction to where the density function grows quickly

(→ score functions preserve every information about density function)

[Score-based generative modeling: outline]

Flexible models ↓

- Bypass the normalizing constant
- Principled statistical methods

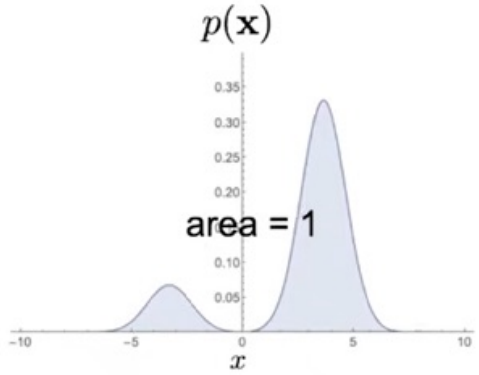
Improved Generation ↓

- Higher sample quality than GANs
- Controllable generation

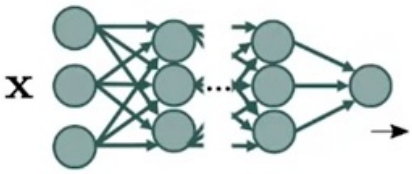
Probability evaluation ↓

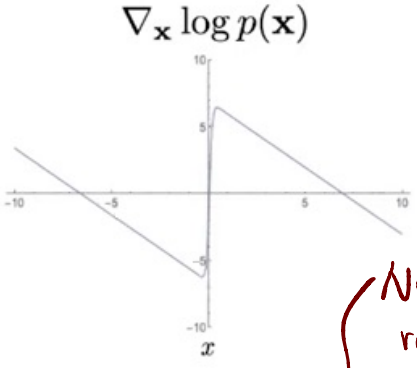
- Accurate probability evaluation
- Better estimation of data probabilities

* Score functions bypass the normalizing constant



Probability density function





Score function

Normalizing restriction ✗

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

Score model

$\nabla_\mathbf{x} \log Z_\theta \rightarrow 0$ (gradient of constant)

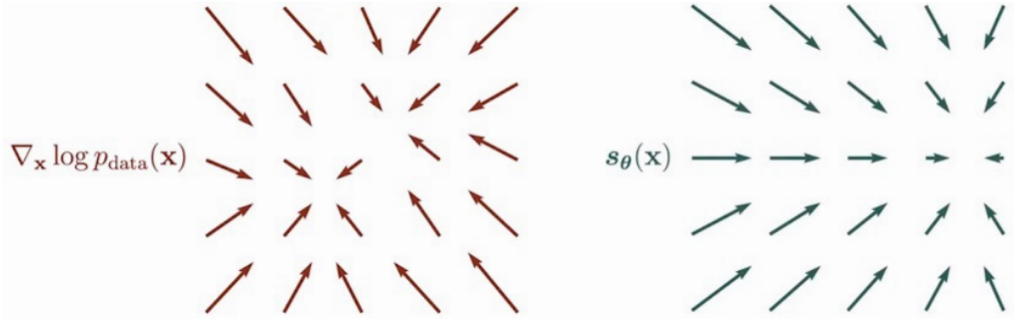
[Score models can be estimated from data]

Given: $\{x_1, x_2, \ldots, x_N\} \overset{i.i.d}{\sim} p_{data}(x)$

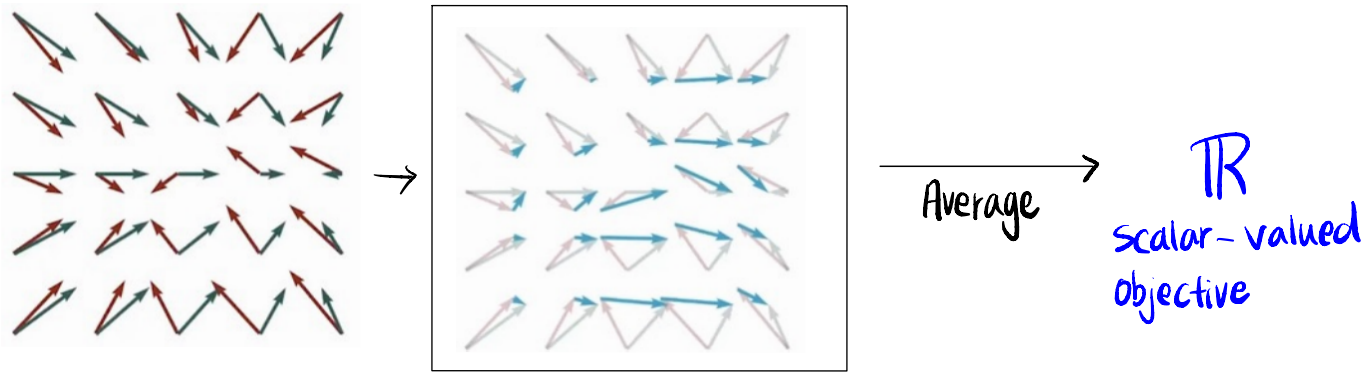
Goal: $\nabla_x \log p_{data}(x)$

Score Model: $s_\theta(x): \mathbb{R}^d \to \mathbb{R}^d \approx \nabla_x \log p_{data}(x)$

Objective: How to compare two vector fields of scores?



GT — compare — Predicted



Difference → Average → $\mathbb{R}$ scalar-valued objective

Objective: $\frac{1}{2}\mathbb{E}_{p_{data}(x)}\left[\|\nabla_x \log p_{data}(x) - s_\theta(x)\|_2^2\right]$

(Fisher divergence)

Cannot be computed because we don't know the ground truth value

Score Matching [Hyvärinen 2005]:

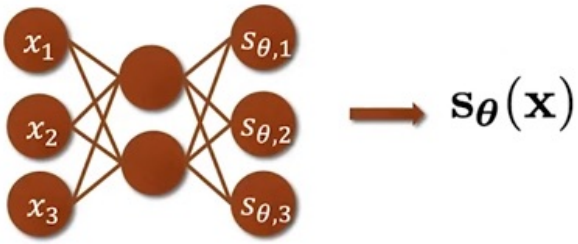
$$\mathbb{E}_{p_{data}(x)}\left[\frac{1}{2}\|s_\theta(x)\|_2^2 + \text{trace}\left(\underbrace{\nabla_x s_\theta(x)}_{\text{Jacobian of } s_\theta(x)}\right)\right]$$

$$\approx \frac{1}{N}\sum_{i=1}^{N}\left[\frac{1}{2}\|s_\theta(x_i)\|_2^2 + \text{trace}(\nabla_x s_\theta(x_i))\right]$$

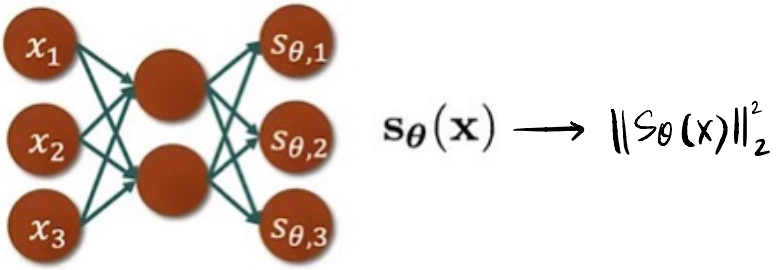
↘ can be approximated using the empirical mean

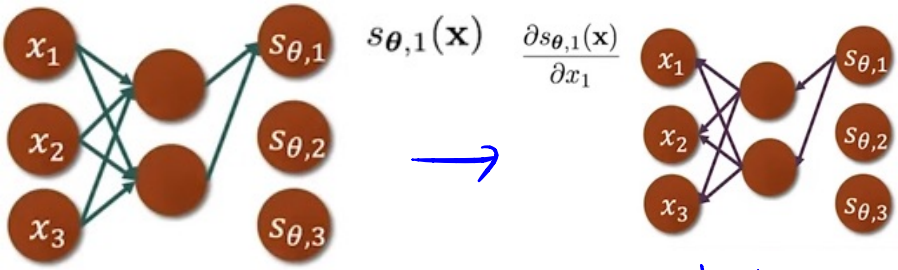
Score Matching is not scalable

- Deep score models



- Need to compute: $\|s_\theta(x)\|_2^2$ and $\text{trace}(\nabla_x s_\theta(x))$





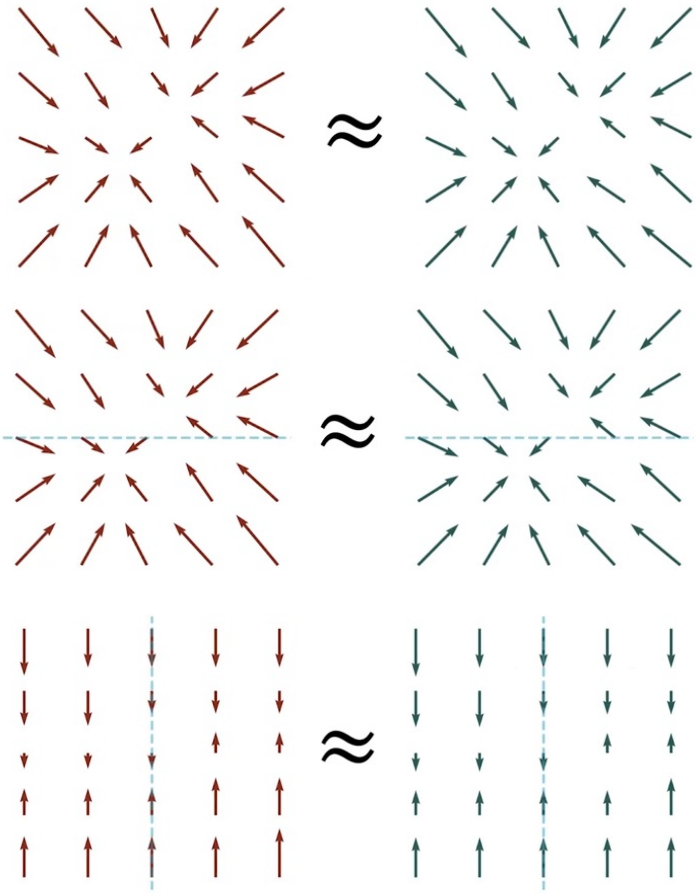
O(# dimensions of x)

$$\nabla_{\mathbf{x}}\mathbf{s}_\theta(\mathbf{x}) = \begin{pmatrix} \frac{\partial s_{\theta,1}(\mathbf{x})}{\partial x_1} & \frac{\partial s_{\theta,1}(\mathbf{x})}{\partial x_2} & \frac{\partial s_{\theta,1}(\mathbf{x})}{\partial x_3} \\ \frac{\partial s_{\theta,2}(\mathbf{x})}{\partial x_1} & \frac{\partial s_{\theta,2}(\mathbf{x})}{\partial x_2} & \frac{\partial s_{\theta,2}(\mathbf{x})}{\partial x_3} \\ \frac{\partial s_{\theta,3}(\mathbf{x})}{\partial x_1} & \frac{\partial s_{\theta,3}(\mathbf{x})}{\partial x_2} & \frac{\partial s_{\theta,3}(\mathbf{x})}{\partial x_3} \end{pmatrix}$$

Σ : trace

Sliced score matching

- Intuition: one dimensional problem should be easier
- Idea: project onto random directions



project them on random one dimensional direction

one dimensional scalar fields

Randomized Objective: Sliced Fisher Divergence

$$\frac{1}{2}\mathbb{E}_{p_v}\mathbb{E}_{p_{data}(x)}\left[\left(v^T\nabla_x \log p_{data}(x) - v^T s_\theta(x)\right)^2\right]$$

v: projection direction $\in \mathbb{R}^d$

p_v: distribution of projection direction

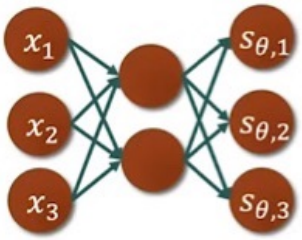
Integration by parts → Sliced Score Matching

$$\mathbb{E}_{p_v}\mathbb{E}_{p_{data}(x)}\left[\boxed{v^T\nabla_x s_\theta(x) v} + \frac{1}{2}\left(v^T s_\theta(x)\right)^2\right]$$

scalable!

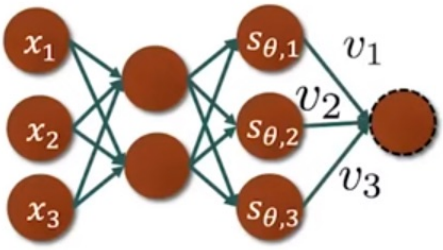
Computing Jacobian-vector products is scalable

$$v^T \nabla_x s_\theta(x) v = v^T \nabla_x (v^T s_\theta(x))$$



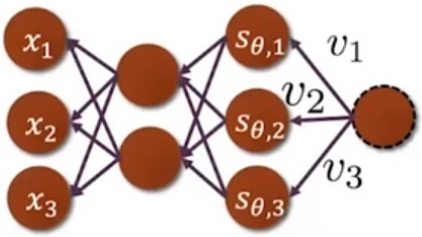
$\mathbf{s_\theta(x)}$

$v^T \nabla_x (v^T s_\theta(x))$



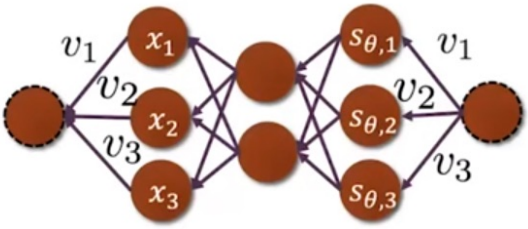
$\mathbf{v^T s_\theta(x)}$

$v^T \nabla_x (v^T s_\theta(x))$



$\nabla_{\mathbf{x}}(\mathbf{v^T s_\theta(x)})$

$v^T \nabla_x (v^T s_\theta(x))$



$\mathbf{v^T \nabla_x (v^T s_\theta(x))}$

$v^T \nabla_x (v^T s_\theta(x))$

One Backprop! (efficient)

How sliced score matching works in practice

- Sample a minibatch of data points $\{x_1, x_2, \cdots x_n\} \sim p_{data}(x)$
- Sample a minibatch of projection directions $\{v_1, v_2, \cdots, v_n\} \sim p_v$
- Estimate the sliced score matching loss with empirical means

$$\frac{1}{n}\sum_{i=1}^{n}\left[v_i^T \nabla_x s_\theta(x_i) v_i + \frac{1}{2}(v_i^T s_\theta(x_i))^2\right]$$

- The projection distribution is typically Gaussian or Rademacher
- Stochastic gradient descent
- Can use more projections per datapoint to boost performance

Denoising score matching

gaussian distribution

- Matching the score of a noise-perturbed distribution

$$\frac{1}{2}\mathbb{E}_{q_\sigma(\tilde{x})}\left[\| \nabla_{\tilde{x}} \log q_\sigma(\tilde{x}) - s_\theta(\tilde{x})\|_2^2\right]$$

- Denoising score matching

$$\frac{1}{2}\mathbb{E}_{p_{data}(x)}\mathbb{E}_{q_\sigma(\tilde{x}|x)}\left[\| \nabla_{\tilde{x}} \log q_\sigma(\tilde{x}|x) - s_\theta(\tilde{x})\|_2^2\right]$$

scalable

✶ Cannot estimate scores of noise-free distributions

$\mathbf{x}$

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

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

$\tilde{\mathbf{x}}$

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

- Denoising score matching:
  matching the score of a noise-perturbed distribution

⟨Procedure⟩

- Sample a minibatch of datapoints $\{x_1, x_2, \cdots, x_n\} \sim p_{\text{data}}(x)$
- Sample a minibatch of perturbed datapoints $\{\tilde{x}_1, \tilde{x}_2, \cdots, \tilde{x}_n\} \sim q_\sigma(\tilde{x})$

$$\tilde{x}_i \sim q_\sigma(\tilde{x}_i \mid x_i)$$

- Estimate the denoising score matching loss with empirical means

$$\frac{1}{2n} \sum_{i=1}^{n} \left[\| s_\theta(\tilde{x}_i) - \nabla_{\tilde{x}} \log q_\sigma(\tilde{x}_i \mid x_i) \|_2^2 \right]$$

- If Gaussian perturbation

$$\frac{1}{2n} \sum_{i=1}^{n} \left[\| s_\theta(\tilde{x}_i) + \frac{\tilde{x}_i - x_i}{\sigma^2} \|_2^2 \right]$$

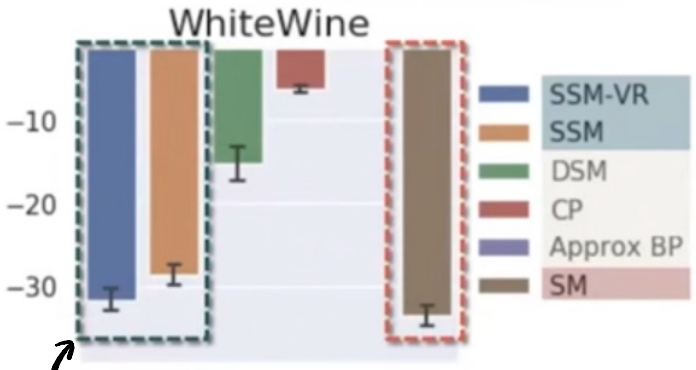
- Stochastic gradient descent

- Need to choose a very small σ!

Results: Sliced Score Matching for EBMs

Sliced score matching methods | Other Baselines | Score Matching

Efficiency



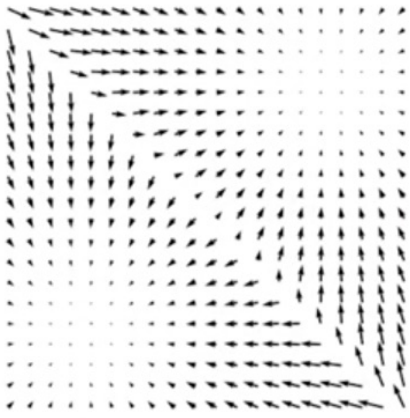
Performance on density estimation

minimal performance loss

Song*, Garg*, Shi, Ermon. "Sliced Score Matching: A Scalable Approach to Density and Score Estimation." UAI 2019.

Sampling from score functions

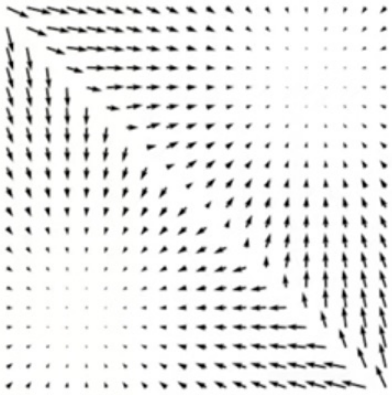
Data examples



Score functions

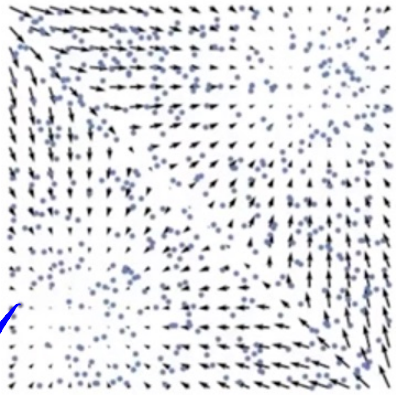
New data

Sampling from score functions: Langevin dynamics



Score function

$s_\theta(x)$

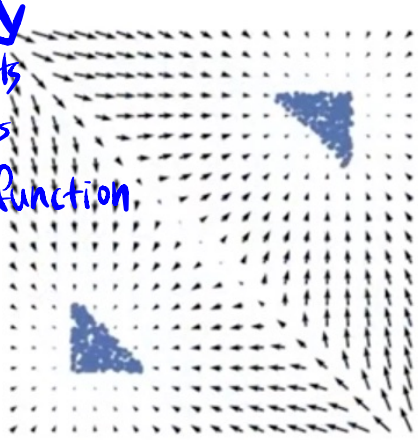


Follow the scores

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

moving the points following the directions predicted by the score function.

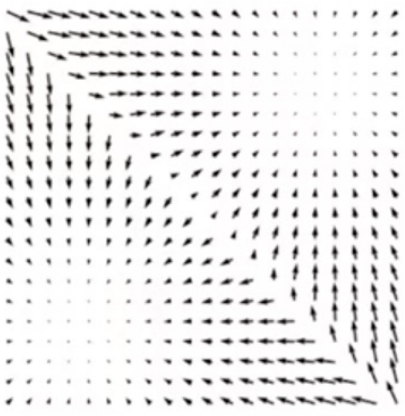
→ points will collapse into each other :(



follow noisy-version of score functions

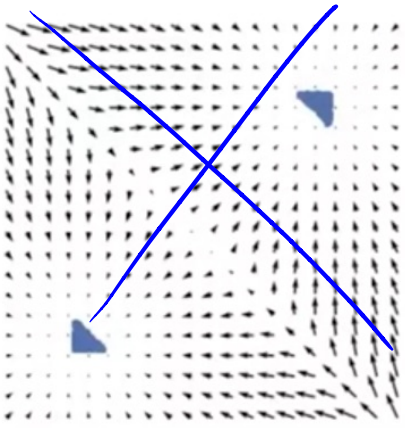
→ Correct samples guaranteed

Sampling from score functions: Langevin dynamics



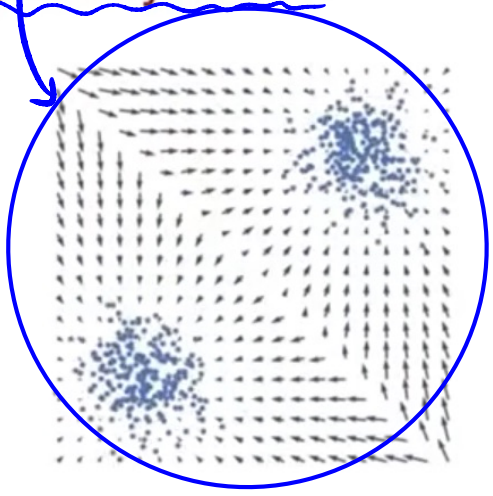
Score function

$s_\theta(x)$



Follow the scores

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



Follow the noisy scores

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

$$\tilde{x}_{t+1} \leftarrow \tilde{x}_t + \frac{\epsilon}{2} s_\theta(\tilde{x}_t) + \sqrt{\epsilon}\, z_t$$

inject Gaussian Noise

Langevin dynamics sampling

- Sample from $p(x)$ using only the score $\nabla_x \log p(x)$
- Initialize $x^0 \sim \pi(x)$ ← prior distribution
- Repeat for $t \leftarrow 1, 2, \cdots, T$

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

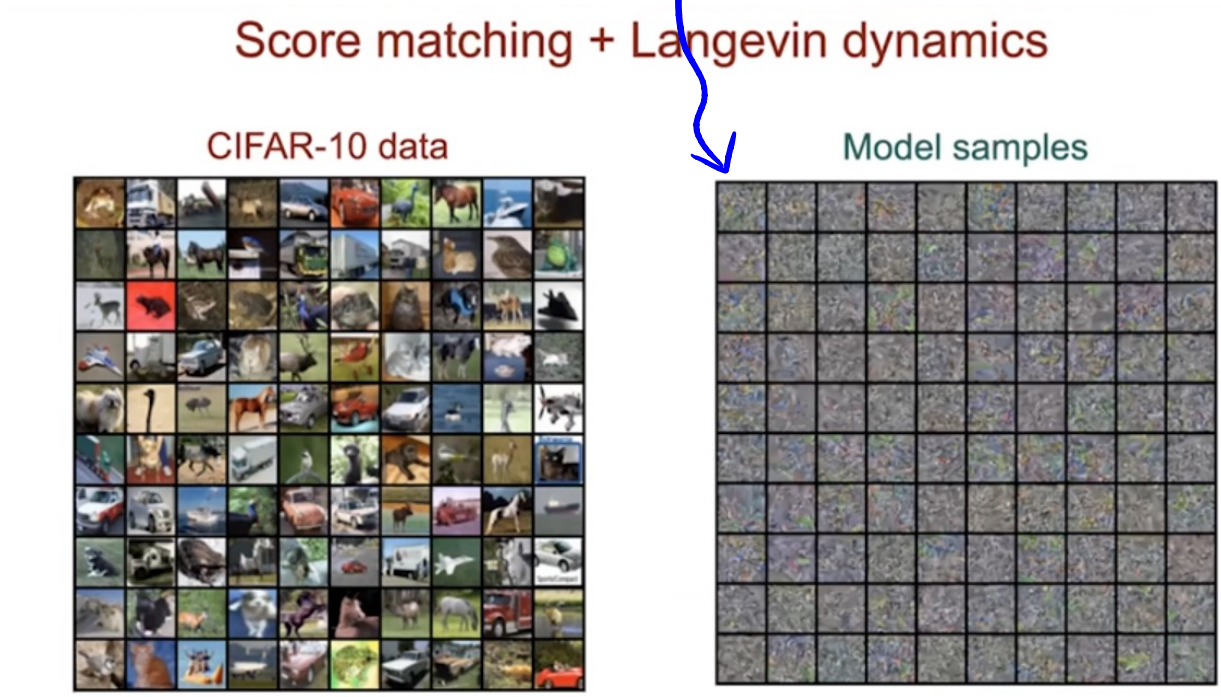
$$x^t \leftarrow x^{t-1} + \frac{\epsilon}{2} \nabla_x \log p(x^{t-1}) + \sqrt{\epsilon}\, z^t$$

- If $\epsilon \to 0$ and $T \to \infty$, we are guaranteed to have $x^T \sim p(x)$

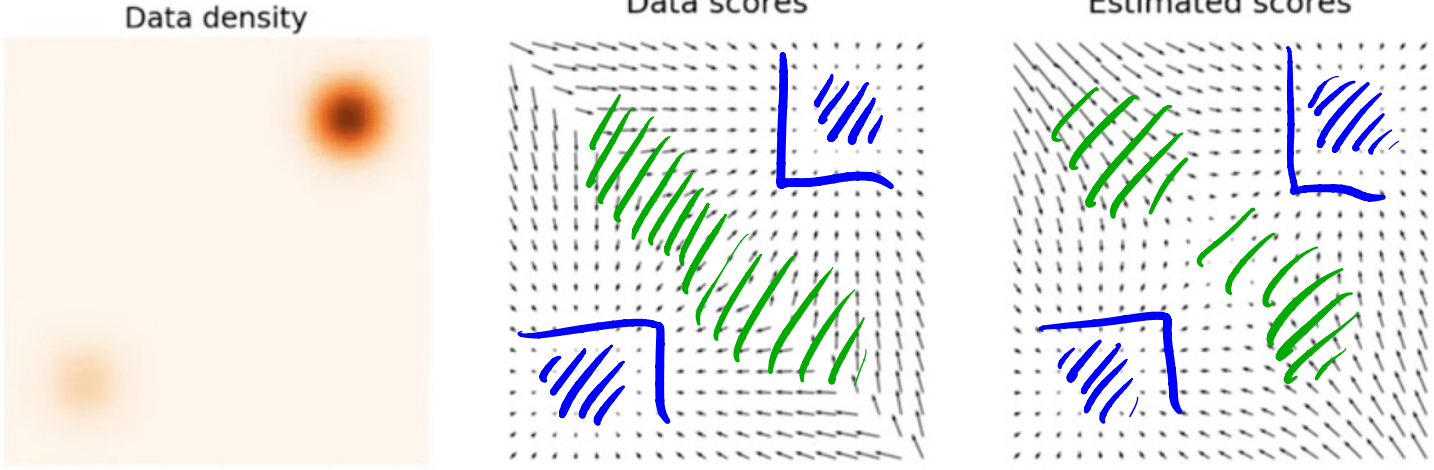
- Langevin dynamics + score estimation

$$s_\theta(x) \approx \nabla_x \log p(x)$$

doesn't work well in practice with naive approach



Challenge in low data density regions

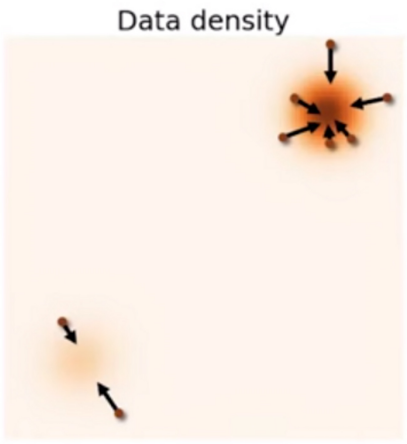


actual

predicted

/// : High density → accurate

/// : Low density → inaccurate



why

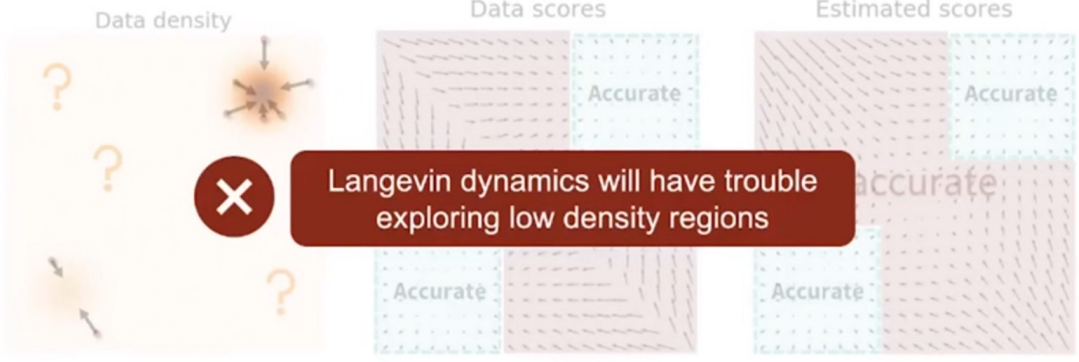
: Score matching compares the difference between ground truth only at datasamples

→ In low density regions, we don't have enough samples

→ x enough information to infer the true score functions

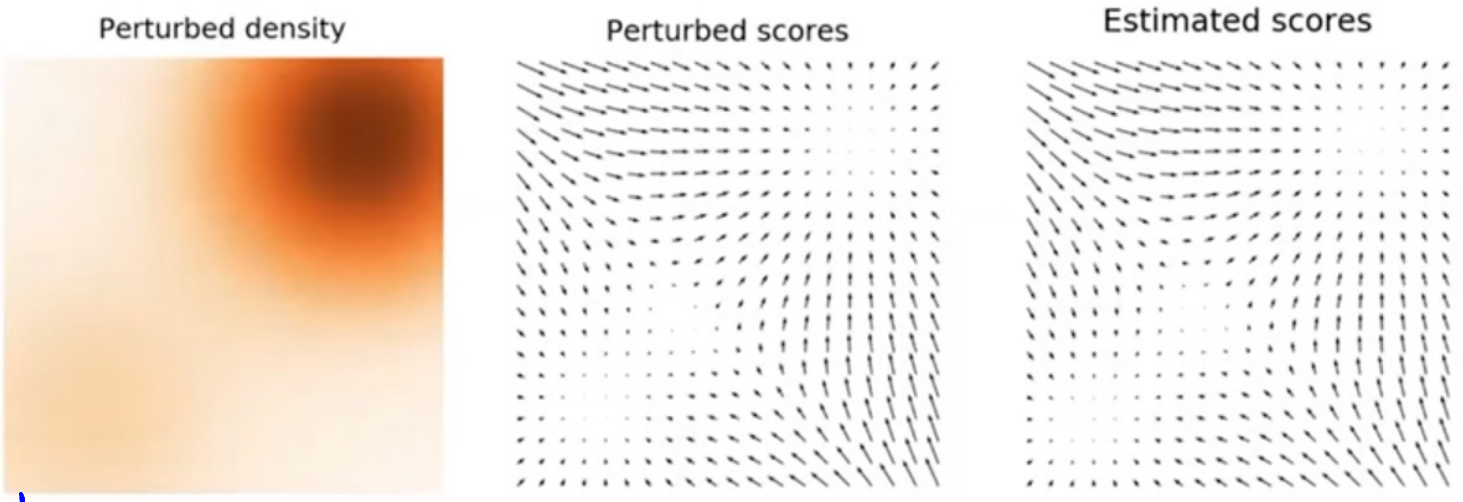
$$\frac{1}{2}\mathbb{E}_{p_{\text{data}}(\mathbf{x})}[\|\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) - \boldsymbol{s}_{\boldsymbol{\theta}}(\mathbf{x})\|_2^2]$$

Challenge in low data density regions



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

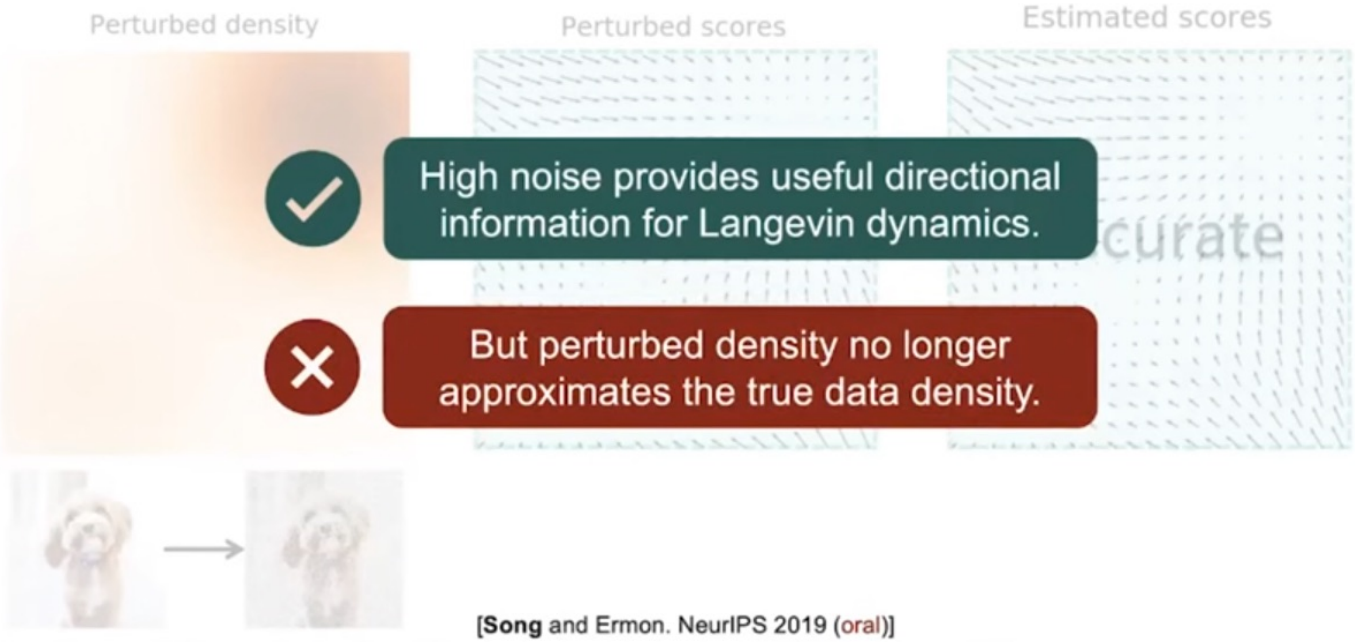
Improving score estimation by adding noise



↳ After adding enough Gaussian noise, perturb the datapoints to everywhere in space

→ the size of lower density regions become smaller

Improving score estimation by adding noise



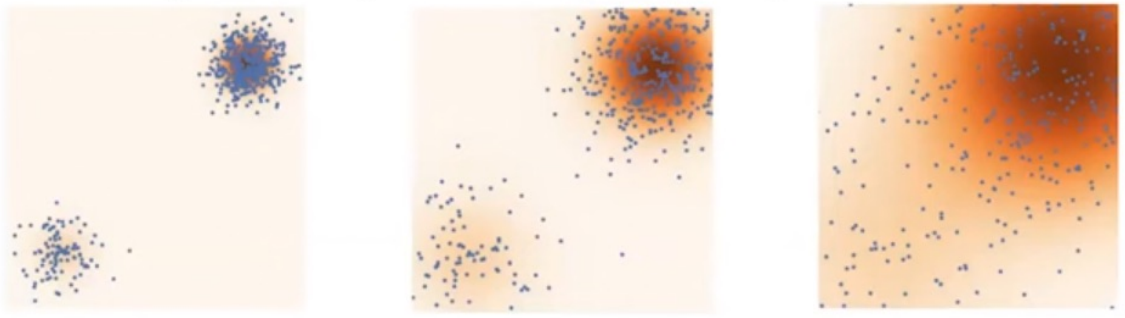
[**Song** and Ermon. NeurIPS 2019 (oral)]

Using multiple noise levels

std: σ_1 < std: σ_2 < std: σ_3

Data

$\sim \mathcal{N}(0, \sigma)$



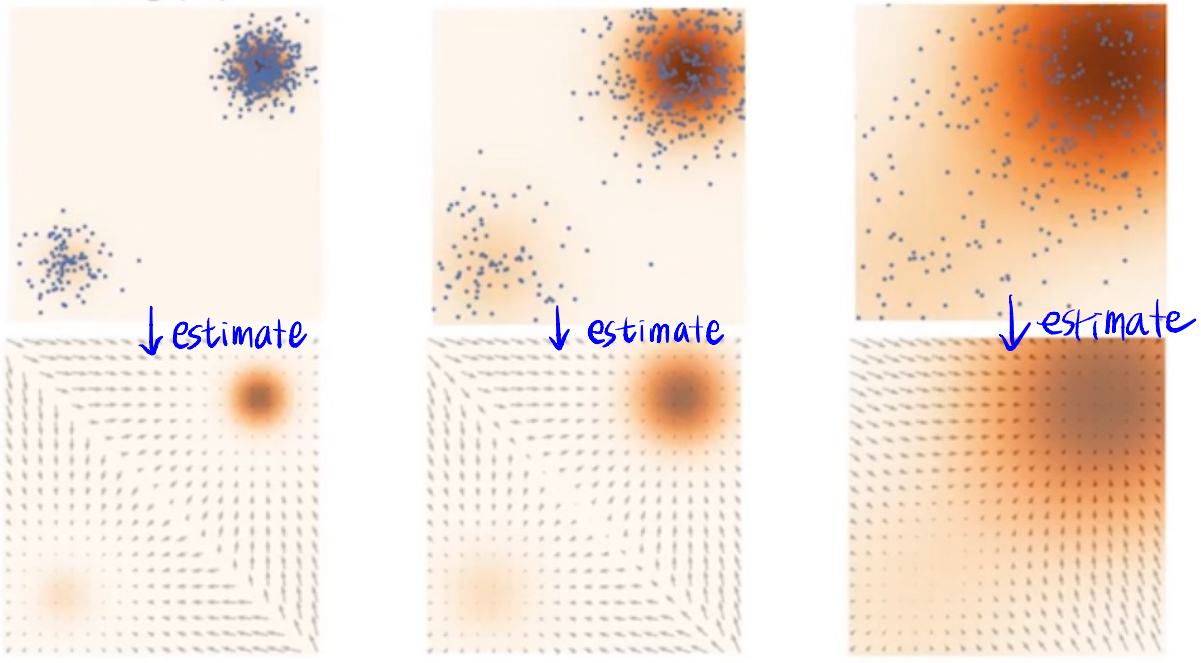
$p_{\sigma_1}(x)$ $p_{\sigma_2}(x)$ $p_{\sigma_3}(x)$

Corresponding noisy-data density $p_0(x)$

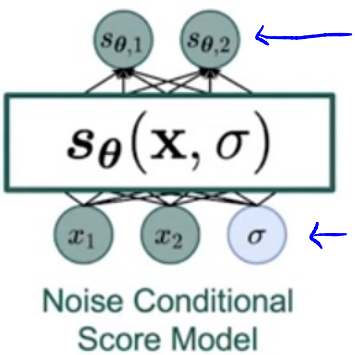
Using multiple noise levels

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

noisy Data

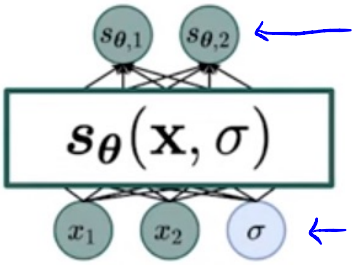


estimate the underlying score function



← output: score function of data density perturbed with noise label σ

← input: input ⊕ noise label σ



← output: score function of data density perturbed with noise label σ

← input: input ⊕ noise label σ

how can we train this model ?

⇒ leverage the idea of score-matching

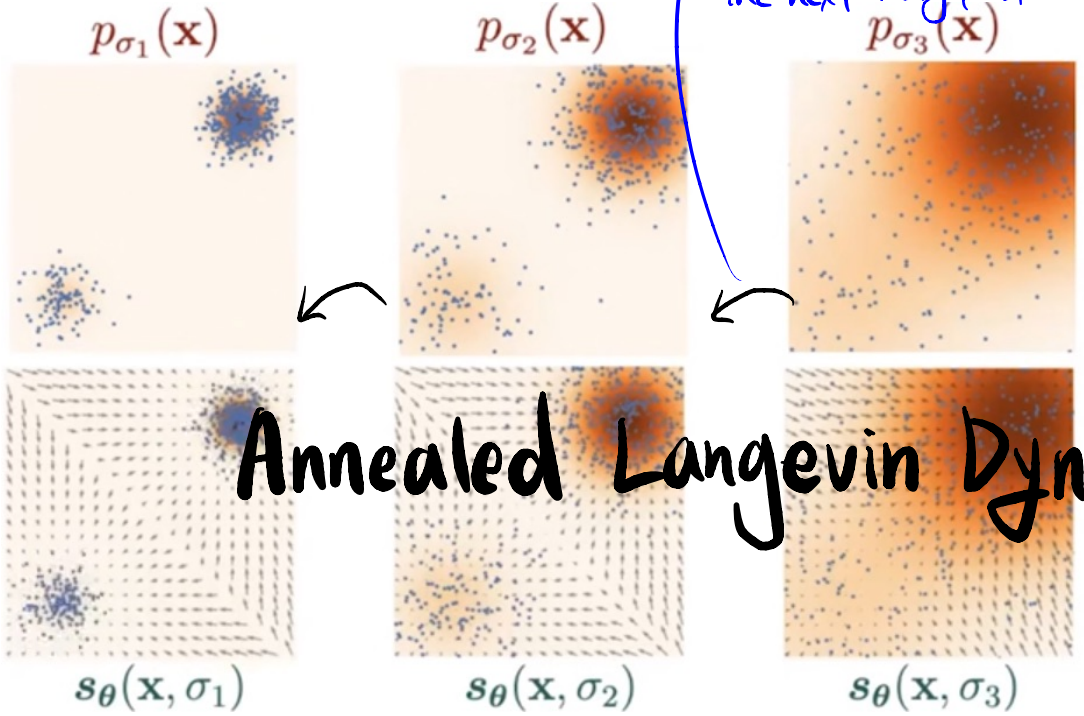
$$\frac{1}{N}\sum_{i=1}^{N} \boxed{\lambda(\sigma_i)}\, \mathbb{E}_{p_{\sigma_i}(x)}\left[\|\nabla_x \log p_{\sigma_i}(x) - s_\theta(x, \sigma_i)\|_2^2\right]$$

positive weighting function

score matching loss for all noise labels σ_i

- A generalization to the training objective of diffusion probabilistic models
- First unveiled by DDPM

Samples will be used as initialization to sample from the score model of the next noisy label

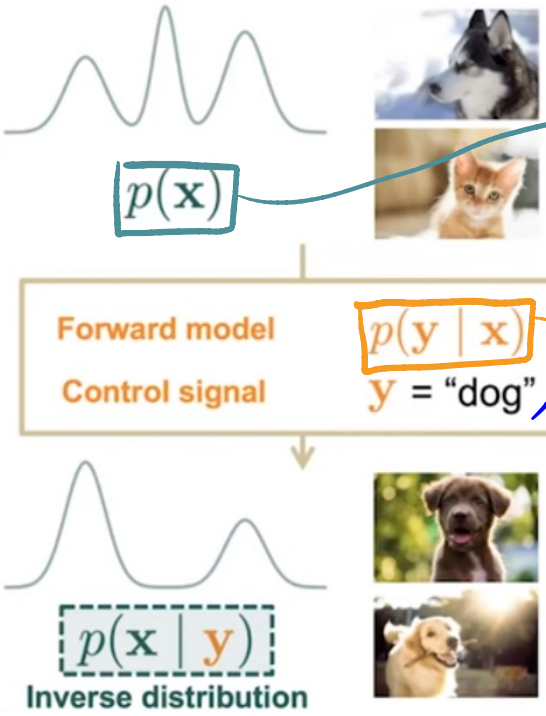


Annealed Langevin Dynamics

Quantitative results on CIFAR0-10

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 ± .07	36.4
MoLM [45]	7.90 ± .10	**18.9**
SNGAN [36]	8.22 ± .05	21.7
ProgressiveGAN [25]	8.80 ± .05	-
NCSN (Ours)	**8.87** ± .12	25.32

Control the generation process



classification

"Generate images of dogs"

- Bayes' rule:

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

↑ Normalizing constant

- Bayes' rule for score functions:

$$\nabla_x \log p(x|y)$$
$$= \nabla_x \log \left\{ \frac{p(x)\,p(y|x)}{p(y)} \right\}$$
$$= \nabla_x (\log p(x) + \log p(y|x) - \log p(y))$$
$$= \nabla_x \log p(x) + \nabla_x \log p(y|x) - \nabla_x \log p(y)$$

$\nabla_x \log p(y) \to 0$

$$= \nabla_x \log p(x) + \nabla_x \log p(y|x)$$

Plug in different forward models for the same score model

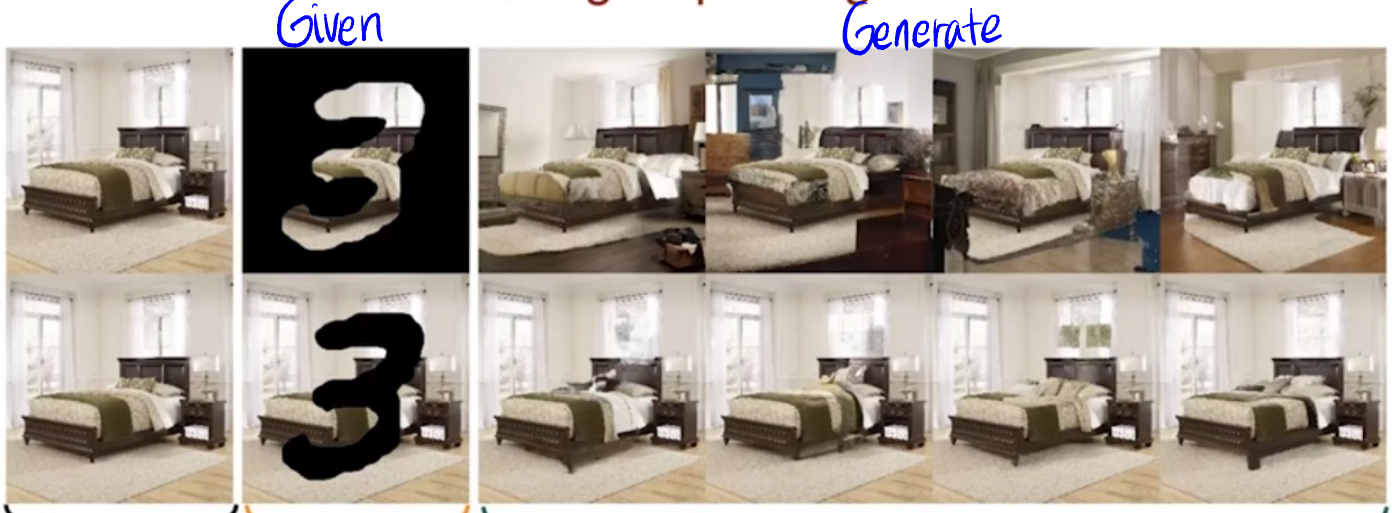
$\nabla_x \log p(x)$: Unconditional score $\approx s_\theta(x)$

$\nabla_x \log p(y|x)$: Forward model

Controllable Generation: class-conditional generation

- $\mathbf{y}$ is the **class label**
- $p_t(\mathbf{y} \mid \mathbf{x})$ is a time-dependent classifier (classifier-guidance)

Image inpainting



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

Forward model can be directly specified

[**Song** et al. ICLR 2021 (Outstanding Paper Award)]

Image colorization

Given (= Control signals) Generate

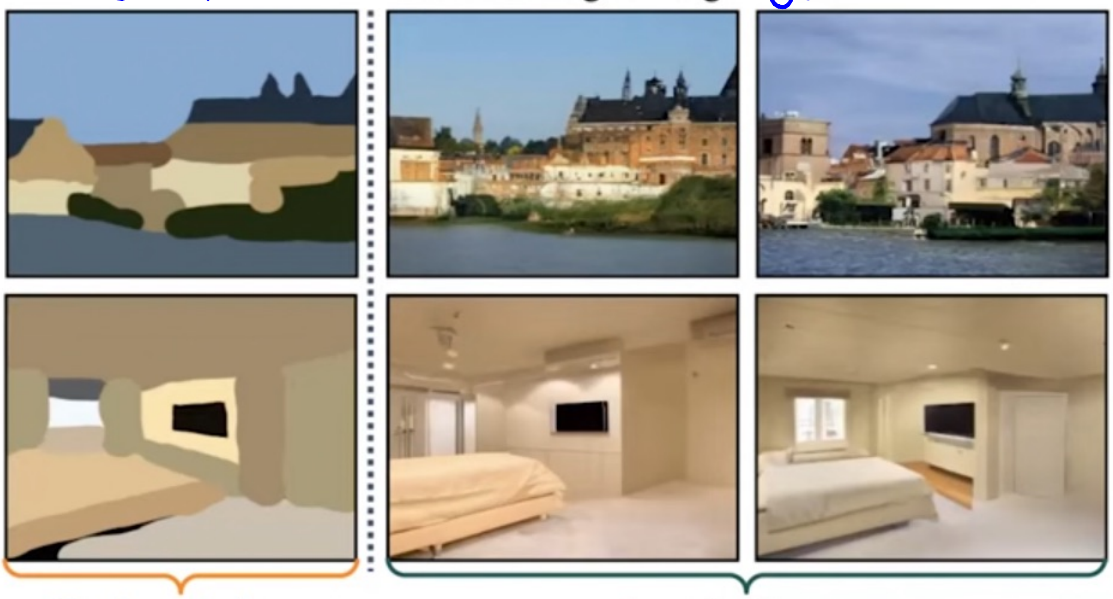
Ground truth | Gray images $\mathbf{y}$ | Colorized images $\mathbf{x} \mid \mathbf{y}$

Forward model can be directly specified

→ One score-based generative model for both image inpainting and colorization

Stroke to image synthesis

Given Stroke Painting to Image Generate



Forward model $p(\mathbf{y} \mid \mathbf{x})$ can be specified.

Stroke paintings $\mathbf{y}$ | Sampled images $\mathbf{x} \mid \mathbf{y}$

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

Language-guided image generation

y

(Prompt)
Treehouse in the style of Studio Ghibli animation

Given

x | y

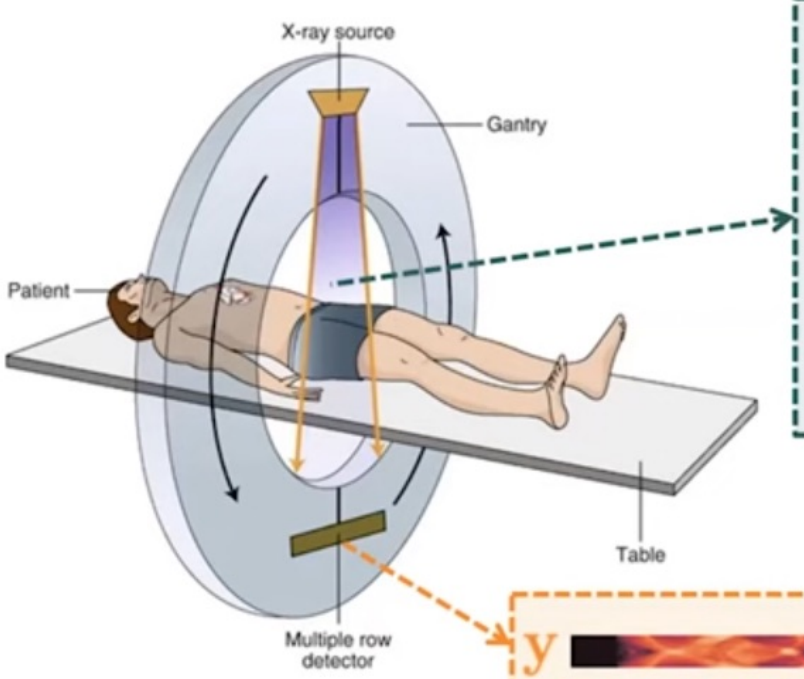
[Work by @danielrussruss]

Generate x|y

Forward model

p(y|x) is an image captioning neural network

Medical image reconstruction



Cross-sectional image

x | y

Generate

y

Sparse-view sinogram

Given

Sparse-view computed tomography (CT)

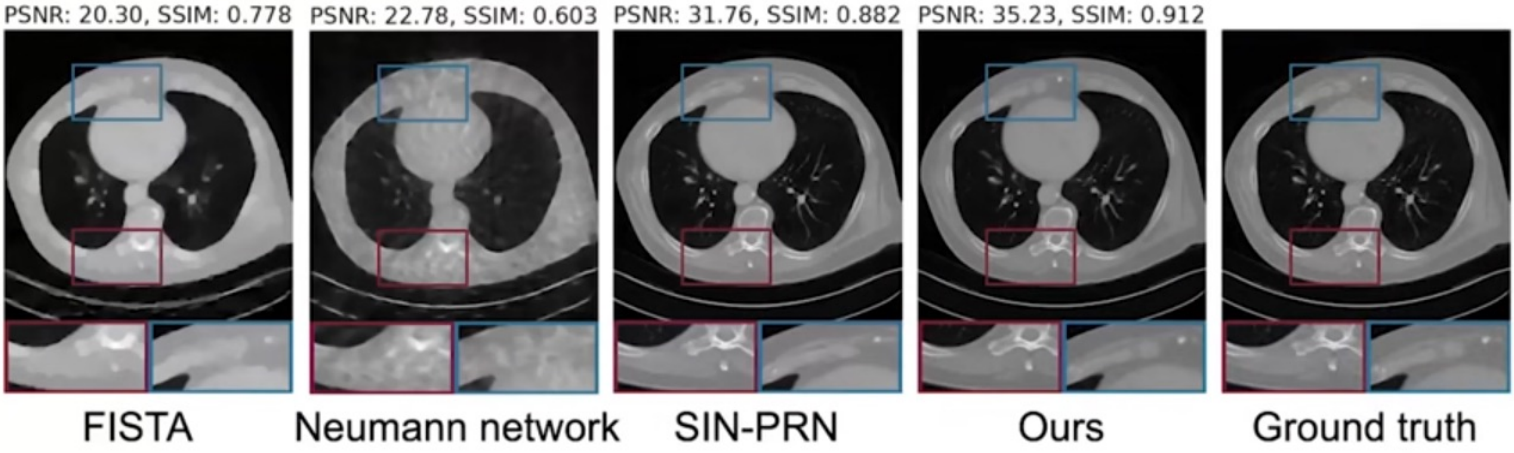
Forward model p(y|x) is given by physical simulation

PSNR : ↑ : better

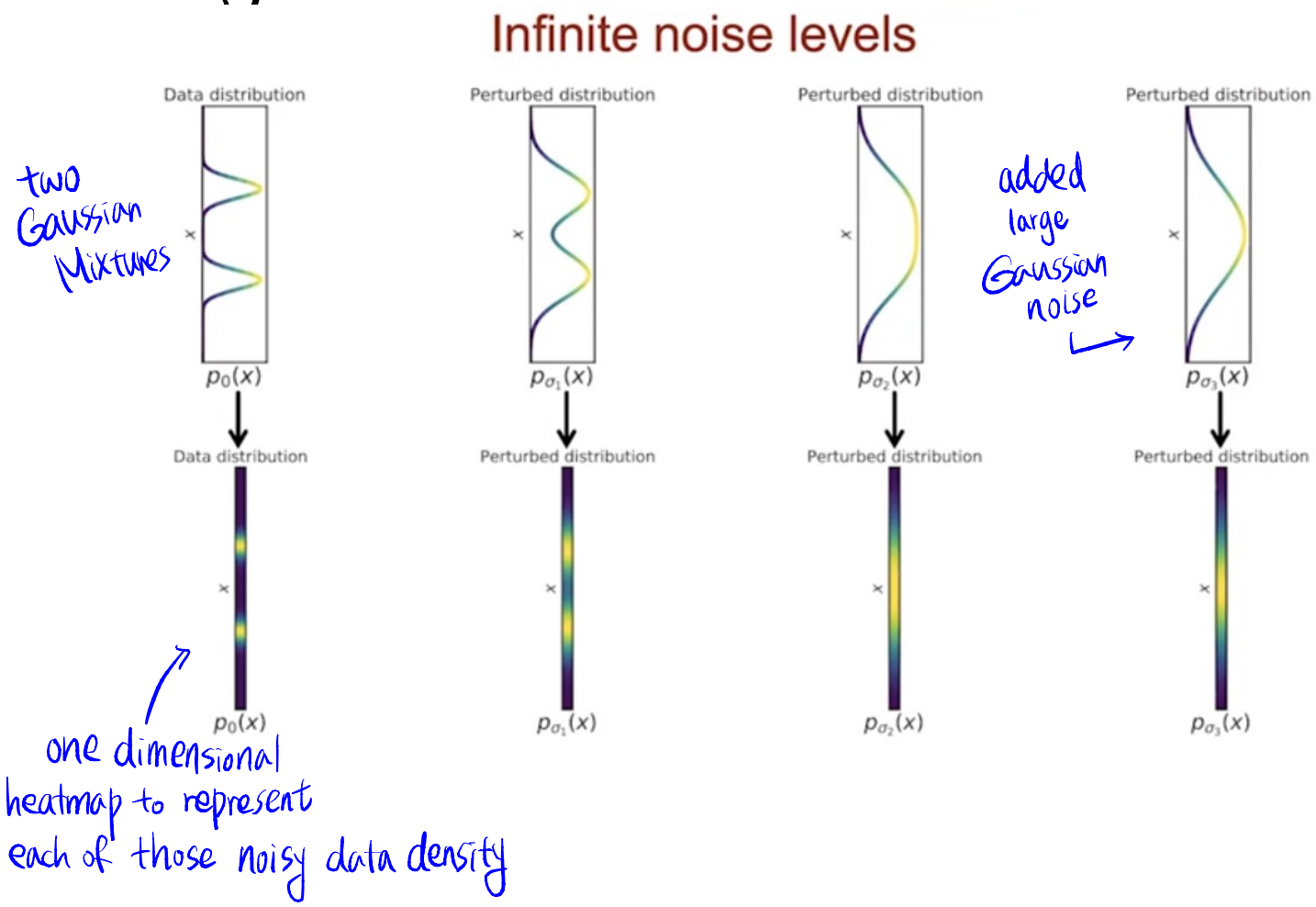
SSIM : ↓ : better

Medical image reconstruction

Sparse-view CT (just 23 projections)



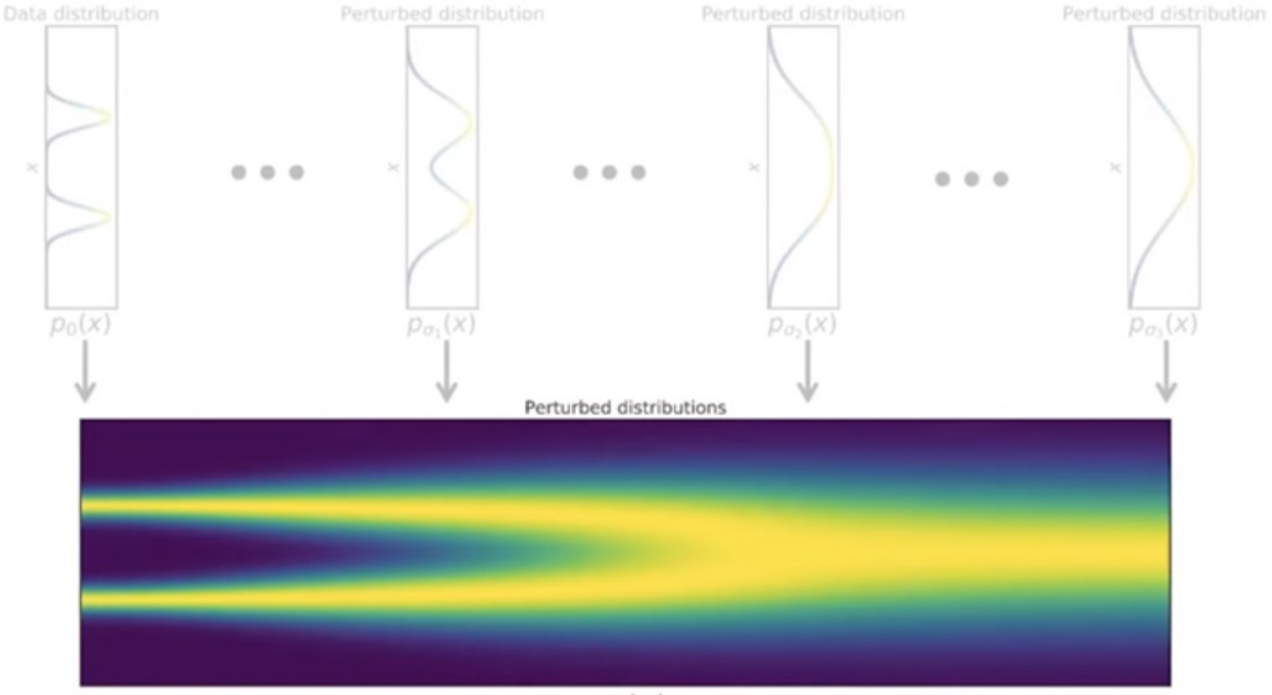
[Probability Evaluation]



two Gaussian Mixtures

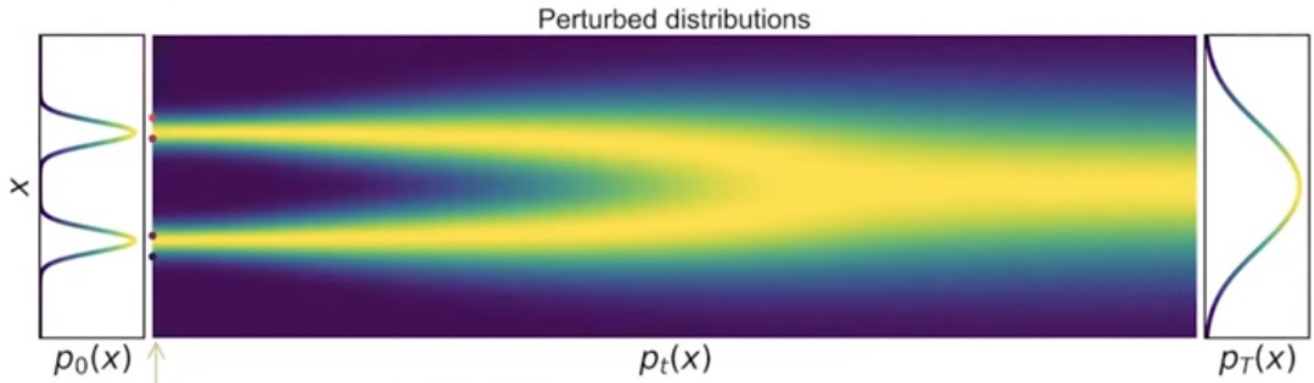
added large Gaussian noise

one dimensional heatmap to represent each of those noisy data density

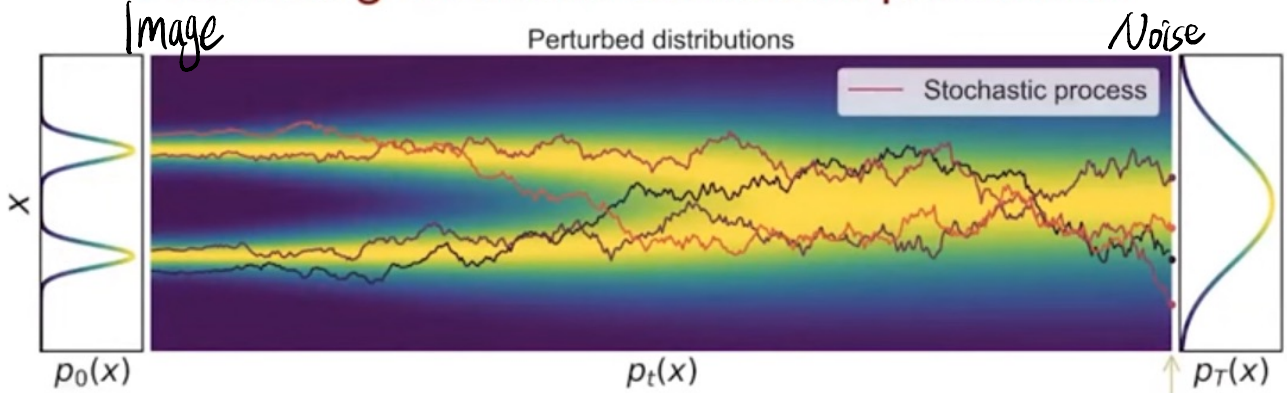


$P_0(x)$: noise-free

$P_T(x)$: Gaussian distribution



Perturbing data with stochastic processes



Stochastic process

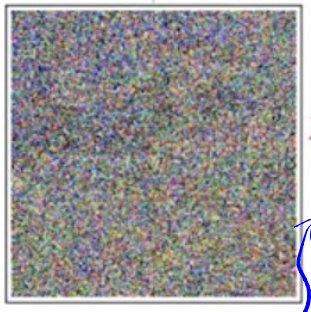
$\{x_t\}_{t \in [0,T]}$ ← random variables

⋮

Corresponding probability densities for random variables

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

one stochastic process ⇕ infinite number of probability densities



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

samples from simple gaussian distribution

How to choose the right stochastic process such that it represents an infinite number of noisy data densities

Stochastic process

$\{x_t\}_{t \in [0,T]}$ ← random variables

⋮

Corresponding probability densities for random variables

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

← Stochastic Differential Equation (SDE)

$$dx_t = f(x_t, t)dt + g(t)dw_t$$

Deterministic drift

Infinitesimal noise

≀ without loss of generality

WLOG: Toy SDE

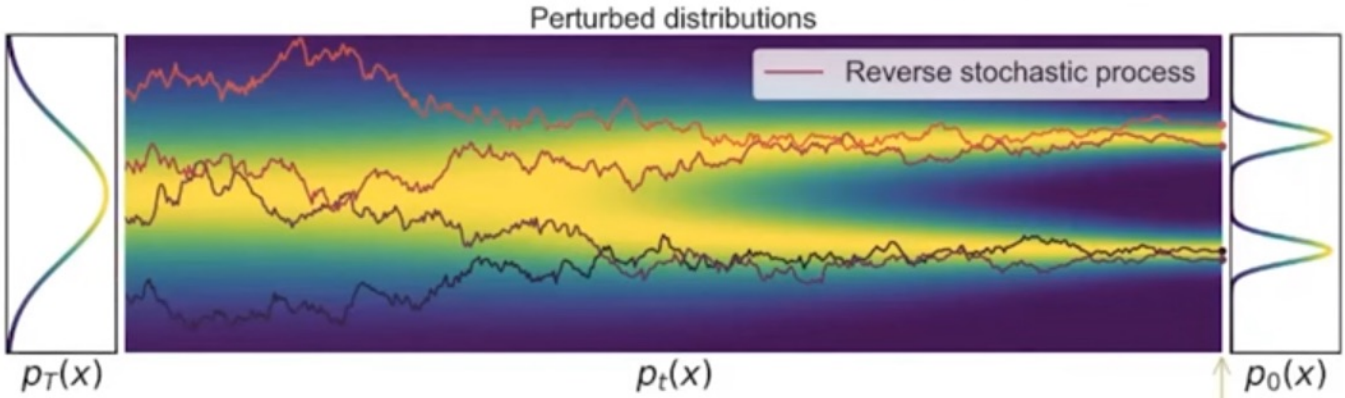
$$dx_t = \sigma(t)dw_t$$

stochastic term

= a continuous time generalization of noise label σ_i

Reverse Stochastic process

Generation via reverse stochastic processes



Forward SDE $(t: 0 \to T)$

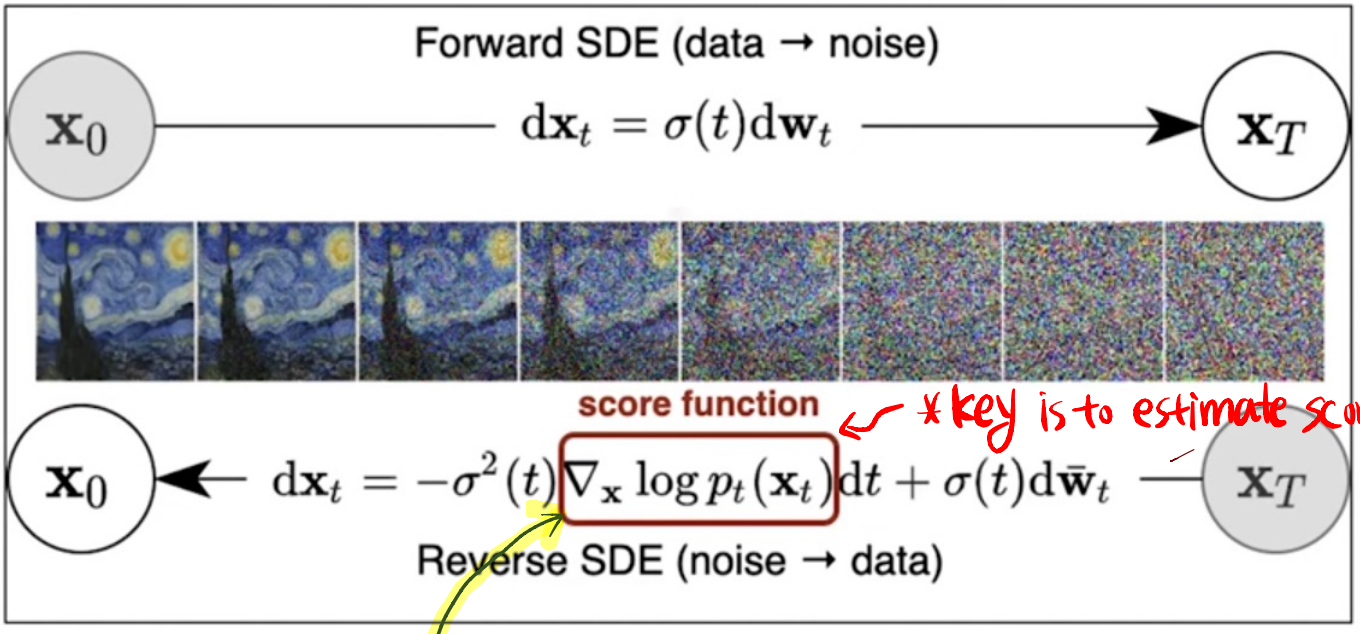
$$dx_t = \sigma(t)\, dw_t$$

Reverse SDE $(t: T \to 0)$

infinitesimal noise in the reverse time direction

$$dx_t = -\sigma(t)^2 \underbrace{\nabla_x \log p_t(x_t)}_{\text{score function}} dt + \sigma(t)\, d\bar{w}_t$$

Score-based generative modeling via SDEs



* key is to estimate score function

[Song et al. ICLR 2021 (Outstanding Paper Award)]

Time conditional score model

$S_\theta(x, t)$

$\approx$

$\nabla_x \log p_t(x_t)$

train 'time conditional score function' model to approximate score function of the data density and time instance t

Training objective:

$$\mathbb{E}_{t \sim \text{Uniform}[0,T]}\left[\lambda(t)\,\mathbb{E}_{p_t(x)}\left[\left\| \nabla_x \log p_t(x) - S_\theta(x, t)\right\|_2^2\right]\right]$$

$\lambda(t)$: positive weighting function

$\mathbb{E}_{p_t(x)}\left[\left\| \nabla_x \log p_t(x) - S_\theta(x, t)\right\|_2^2\right]$: score matching loss for any time instant t

Score-based generative modeling via SDEs

(After training, we achieve:)

- Time-dependent score-based model

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

- Training:

Minimize objective:

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

Train efficiently by { ① denoising score matching ② sliced score matching

- Reverse-time SDE

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

  - Euler-Maruyama (analogous to Euler for ODEs)

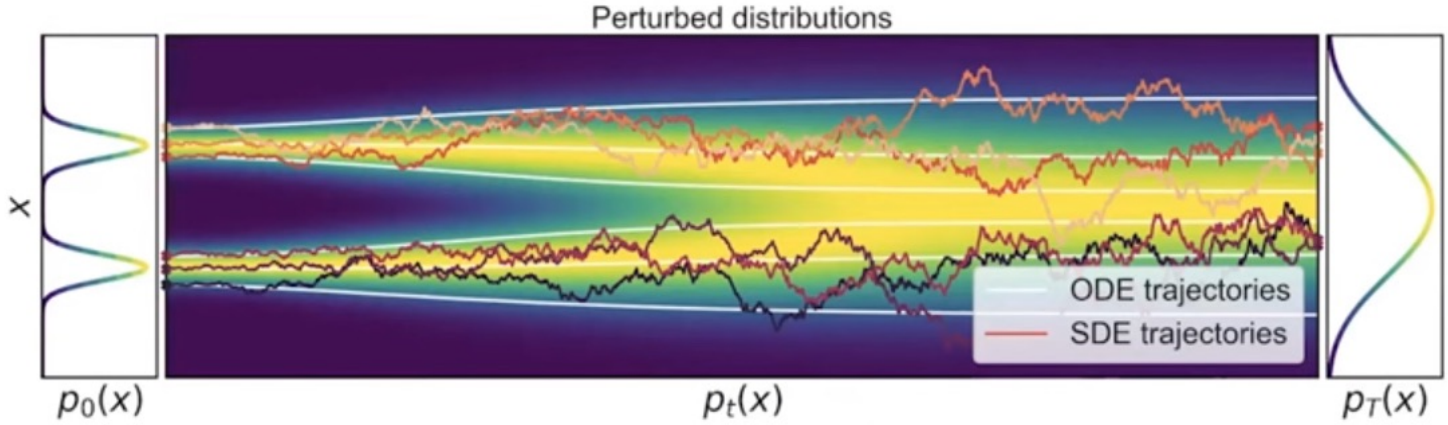
$$\mathbf{x} \leftarrow \mathbf{x} - \sigma(t)^2\mathbf{s}_\theta(\mathbf{x}, t)\Delta t + \sigma(t)\ \mathbf{z} \quad (\mathbf{z} \sim \mathcal{N}(\mathbf{0}, |\Delta t|\ \mathbf{I}))$$
$$t \leftarrow t + \Delta t$$

cf.

SDE vs ODE

Markov non-Markov

Converting the SDE to an ODE



SDE

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

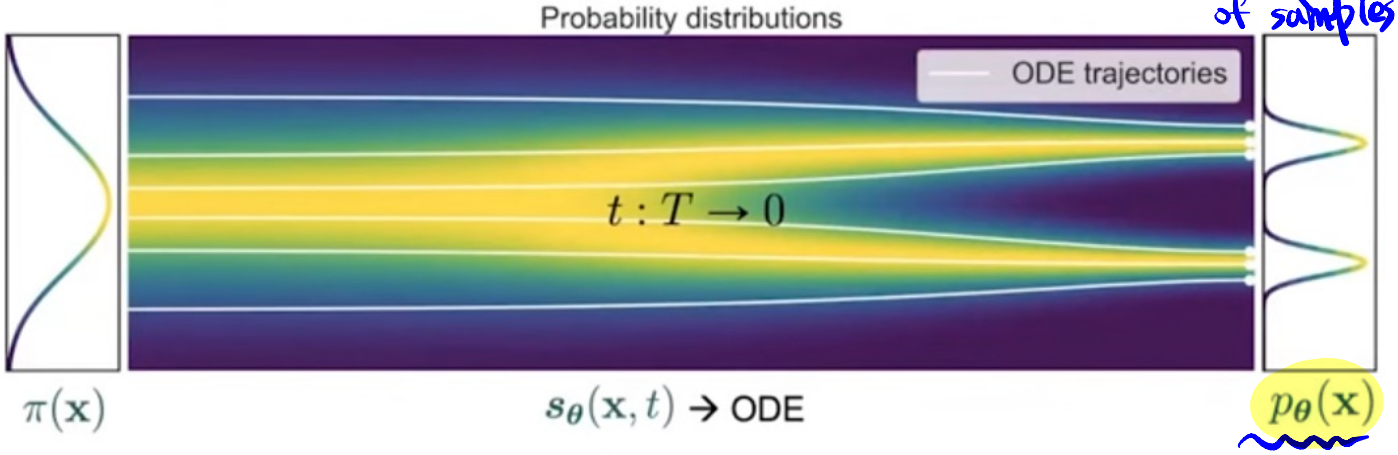
Ordinary differential equation (probability flow ODE)

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

ODE only rely on score function

$\approx s_\theta(x, t)$

Evaluating the probabilities with ODEs



Resulting distribution of samples

Computing the exact likelihood with ODEs

Likelihood:

$$\pi(\mathbf{x}) \xrightarrow[t: T \to 0]{\boldsymbol{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)} p_{\boldsymbol{\theta}}(\mathbf{x})$$

Applications of likelihood:

- Lossless compression (Witten et al. 1987, Townsend et al. 2019)
- Unsupervised anomaly detection (Pimental et al. 2014, **Song** et al. 2018)
- Generative classification (Ng & Jordan 2002, Zimmermann, Schott, **Song**, et al. 2021)
- Density estimation (Silverman, 1986)

Probability flow ODE allows exact likelihood computation:

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

ODE solver

Unbiased estimator

(Change-of-variable formula for ODEs, Chen et al. 2018)

[**Song** et al. ICLR 2021 (Outstanding Paper Award)]

Efficient maximum likelihood training

Theorem (informal): Connection between the Kullback-Leibler (KL) divergence and score matching.

"Likelihood weighting"

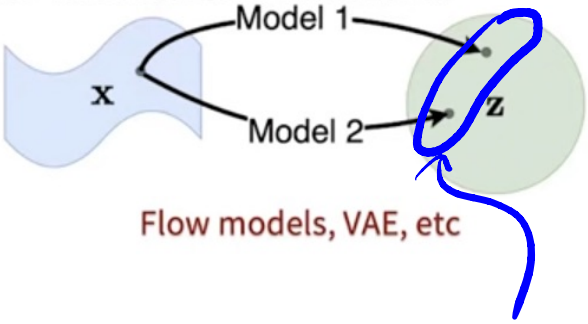
$$\mathrm{KL}(p_{\text{data}} \parallel p_{\boldsymbol{\theta}}) \leqslant \frac{1}{2}\mathbb{E}_{t \sim \text{Uniform}[0,T]}\left[\sigma(t)^2 \mathbb{E}_{p_t(\mathbf{x})}\left[\left\|\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - \boldsymbol{s}_\theta(\mathbf{x}, t)\right\|_2^2\right]\right] + \mathrm{KL}(p_T \parallel \pi) \approx 0$$

Efficient surrogate loss for maximum likelihood training

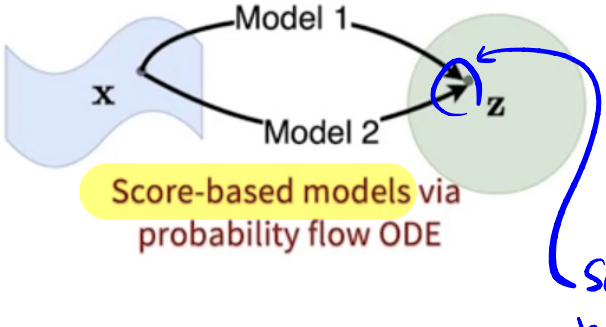
[**Song** et al. NeurIPS 2021 (spotlight)]

Probability flow ODE: uniquely identifiable encoding

- Uniquely **identifiable** encoding



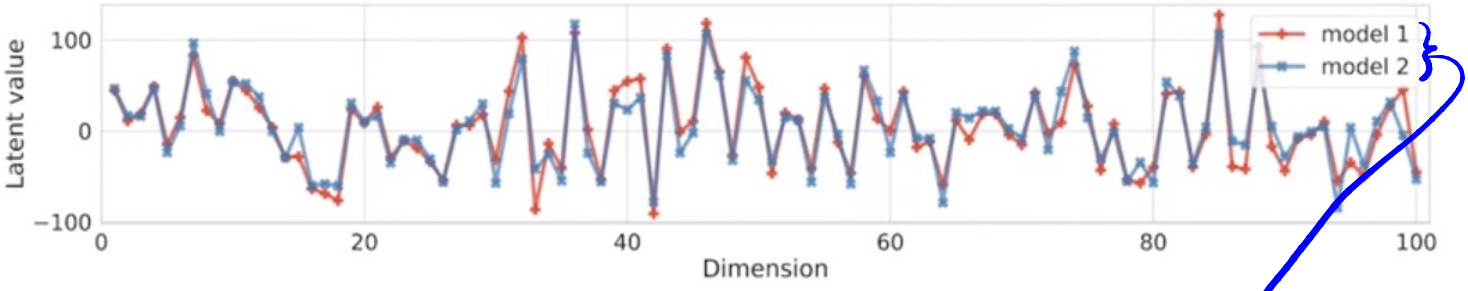
different points



same point

- No trainable parameters in the probability flow ODE

$$dx = \frac{1}{2}\sigma(t)^2 \nabla_x \log p_t(x)\, dt$$



[**Song** et al. ICLR 2021 (Outstanding Paper Award)]

different models, but very similar plots!

Hypothesis on why diffusion models beat GANs

- ☑ More flexible network architectures
- ☑ More diversity