NỀN TẢNG AI TẠO SINH

(IT5410 – Foundation of Generative AI)

SCORE-BASED MODELS

Phạm Minh Hiếu

Thân Quang Khoát

Trường CNTT&TT, ĐHBKHN

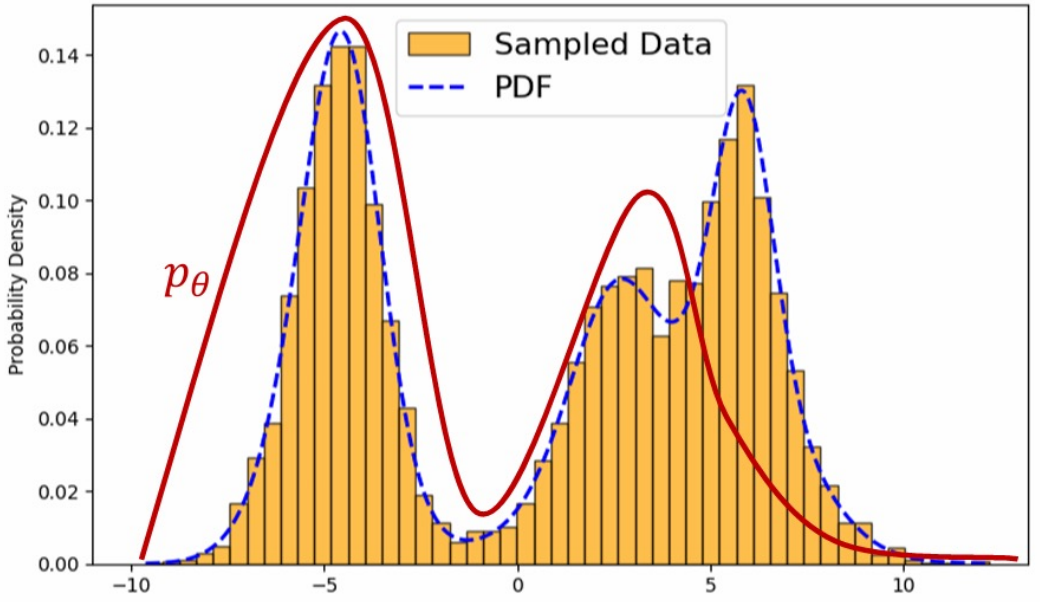
2025

Nội dung

- Mở đầu
- Một số vấn đề của Học sâu
- Một số kiến trúc mạng nơron
- **Mô hình sinh sâu**
- Đánh giá chất lượng
- Học tăng cường

Learning a generative model

- Given a training set of examples, e.g., images of dogs
- We want to learn a probability distribution P(**x**) over image **x** such that
  - **Generation:** If we sample $\boldsymbol{x}_{new} \sim P(\boldsymbol{x})$, $\boldsymbol{x}_{new}$ should look like a dog (sampling)
  - **Density estimation:** $P(\mathbf{x})$
  - **Unsupervised representation learning:** We should be able to learn what these images have in common, e.g., ears, tail, …
- In practice
  - We often find a $P_\theta(\boldsymbol{x})$ to **approximate** $P(\boldsymbol{x})$ as **"close"** as possible
  - **How do we evaluate "closeness"?**
- How to choose a good model family?
  - $P_\theta(\boldsymbol{x})$



Score estimation: Minimizing the Fisher divergence

- **Score function:** $s(\boldsymbol{x}) = \nabla_x \log p(\boldsymbol{x})$
- **Fisher divergence**: Given two distribution P and Q whose pdfs are differentiable, the *Fisher divergence* between P and Q is given by:

$$D_F(P, Q) = \frac{1}{2}\boldsymbol{E}_{\boldsymbol{x}\sim P(\boldsymbol{x})}[\|\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} \log q(\boldsymbol{x})\|_2^2]$$

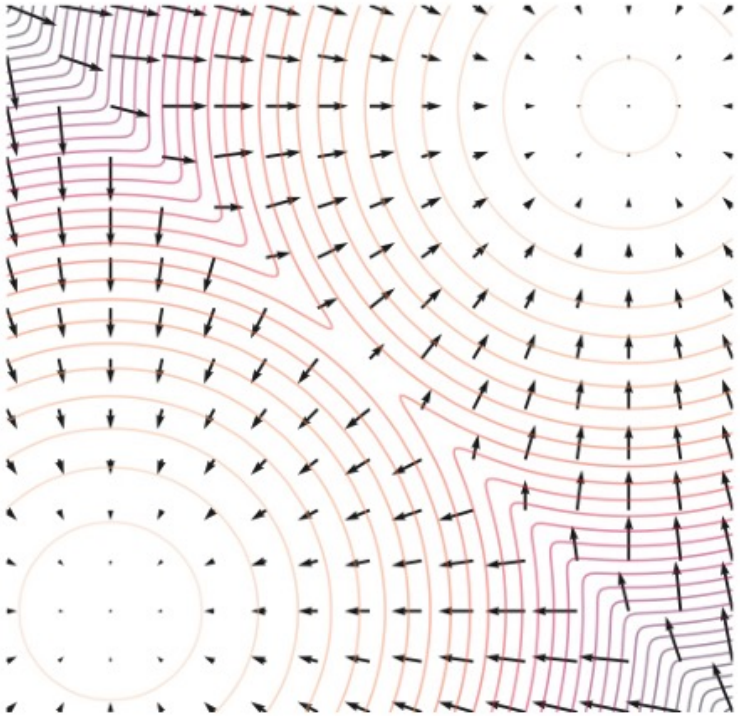
- **Score matching:** minimizing the Fisher divergence between $P(\boldsymbol{x})$ and $P_\theta(\boldsymbol{x})$:

$$\min_\theta \frac{1}{2}\boldsymbol{E}_{\boldsymbol{x}\sim P(\boldsymbol{x})}[\|\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}) - s_\theta(\boldsymbol{x})\|_2^2]$$

where $s_\theta(x)$: $R^d \rightarrow R^d$ is the *score model*.

(d is the number of dimensions)

- Problem: $P(\boldsymbol{x})$ is unknown!



Visualization of pdf and score function of a distribution

Hyvarinen A., Estimation of Non-Normalized Statistical Models by Score Matching, JMLR (2005)

Score matching: Minimizing the Fisher divergence

- Fortunately, we can directly work with the objective after some mathematical transformations (using integration by parts)!

$$\min_{\theta} \frac{1}{2} \boldsymbol{E}_{\boldsymbol{x}\sim P(\boldsymbol{x})}[\|\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}) - s_\theta(\boldsymbol{x})\|_2^2]$$

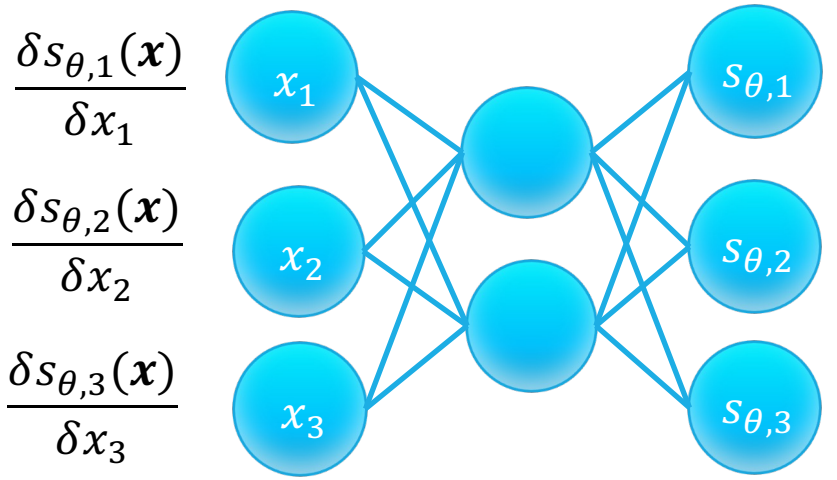
- The objective is equivalent to:

$$\min_{\theta} \boldsymbol{E}_{\boldsymbol{x}\sim P(\boldsymbol{x})} \left[\frac{1}{2}\|s_\theta(\boldsymbol{x})\|_2^2 + tr(\nabla_{\boldsymbol{x}} s_\theta(\boldsymbol{x}))\right]$$

Jacobian of score function

- **Problem:** Computing $tr(\nabla_{\boldsymbol{x}} s_\theta(\boldsymbol{x}))$ requires $O(d)$ backprops.

→ Not scalable as the number of dimensions increases!



Score matching: Training procedure

- Sample a minibatch of data points $\mathrm{B} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\} \sim P(\boldsymbol{x})$.
- Estimate the score matching loss and optimize:

$$L_{SM}(B, \theta) = \frac{1}{m} \sum_{i=1}^{m} \left(\frac{1}{2} \| s_\theta(\boldsymbol{x}_i) \|_2^2 + tr\left(\nabla_x s_\theta(\boldsymbol{x}_i) \right) \right)$$

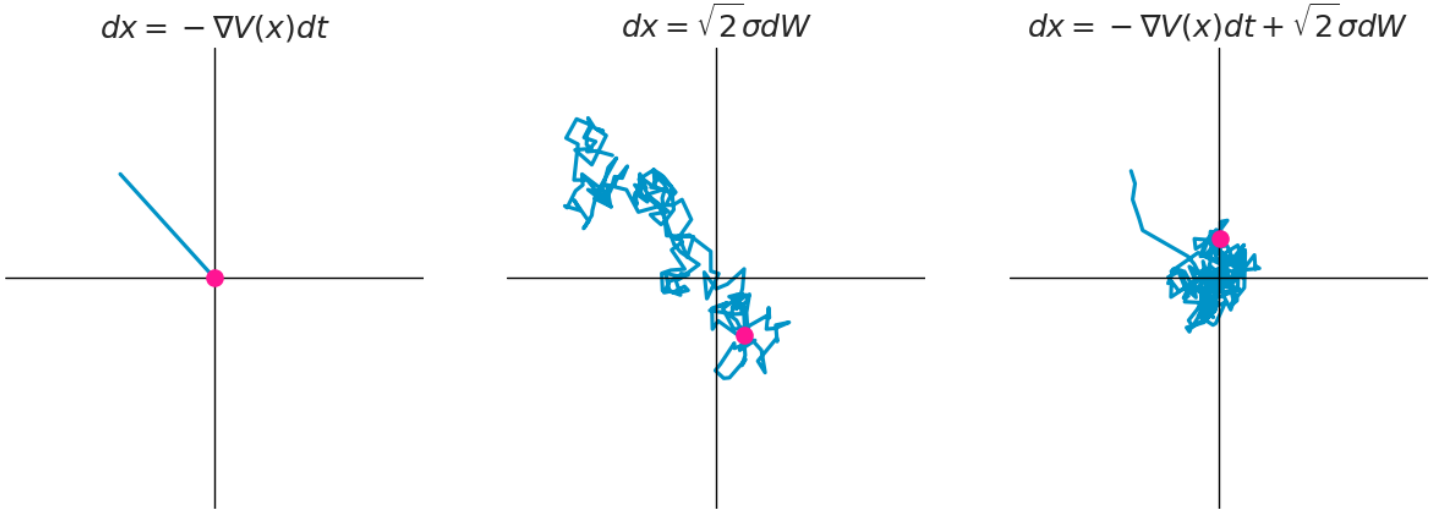
- Repeat until convergence.

Score matching: Sampling via Langevin dynamics

- Allow sampling given only the score function of a distribution.
- The Langevin dynamics for sampling from a distribution $P(\boldsymbol{x})$ is given by the following stochastic differential equation (SDE):

$$d\boldsymbol{x} = \underbrace{\nabla \log p(\boldsymbol{x}) dt}_{\text{drift term}} + \sqrt{2} \underbrace{\sigma}_{\text{volatility}} d\boldsymbol{w}$$

where $\boldsymbol{w}$ denotes the standard Brownian motion (Wiener process).



Source: Langevin Monte Carlo: Sampling using Langevin Dynamics

Score matching: Sampling via Langevin dynamics

Langevin Markov Chain Monte Carlo:

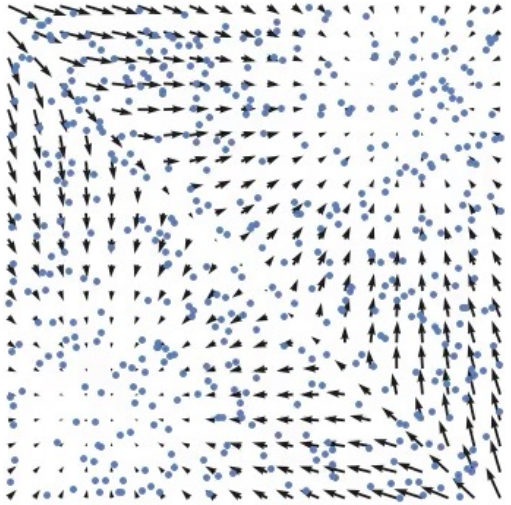
- Sample $\boldsymbol{x}_0 \sim \pi(\boldsymbol{x})$, where $\pi(\boldsymbol{x})$ is a prior distribution that is easy to sample from.
- For $t = 0, 1, \ldots, T-1$:

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \epsilon \nabla_x \log p(\boldsymbol{x}) + \sqrt{2\epsilon}\boldsymbol{z}_t,$$

where $\boldsymbol{z}_t \sim N(0, \boldsymbol{I})$.

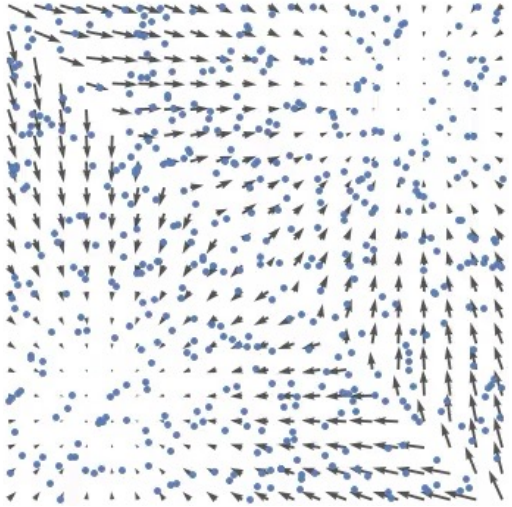
- After training our score-based model: $\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}) \approx s_\theta(\boldsymbol{x})$.

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \epsilon s_\theta(\boldsymbol{x}) + \sqrt{2\epsilon}\boldsymbol{z}_t$$



Follow the score only

$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \epsilon s_\theta(\boldsymbol{x})$



Langevin MCMC

Source: CS236 – Stanford University

Denoising score matching

- Working with the noise-perturbed distribution instead of the original one.

$$q_\sigma(\tilde{\boldsymbol{x}} \mid \boldsymbol{x}) = \boldsymbol{N}(\boldsymbol{x}; \boldsymbol{0}, \sigma^2 \boldsymbol{I})$$

$$q_\sigma(\tilde{\boldsymbol{x}}) = \int p(\boldsymbol{x}) q_\sigma(\tilde{\boldsymbol{x}} \mid \boldsymbol{x}) d\boldsymbol{x}$$

- Score matching on $q_\sigma(\tilde{\boldsymbol{x}})$:

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

- Again, using mathematical transformations, we can convert the objective to a tractable one:

$$\min_\theta \frac{1}{2} \boldsymbol{E}_{\boldsymbol{x} \sim P(\boldsymbol{x}), \tilde{\boldsymbol{x}} \sim q_\sigma(\tilde{\boldsymbol{x}} \mid \boldsymbol{x})} [\| s_\theta(\tilde{\boldsymbol{x}}) - \nabla_{\tilde{\boldsymbol{x}}} \log q_\sigma(\tilde{\boldsymbol{x}} \mid \boldsymbol{x}) \|_2^2]$$

Vincent, P., A Connection Between Score Matching and Denoising Autoencoders, Neural Computation (2010)

Denoising score matching

Since $q_\sigma(\tilde{\boldsymbol{x}} \mid \boldsymbol{x}) = \boldsymbol{N}(\boldsymbol{x}; \boldsymbol{0}, \sigma^2 \boldsymbol{I})$, the density function of $q_\sigma(\tilde{\boldsymbol{x}} \mid \boldsymbol{x})$ is given by:

$$q_\sigma(\tilde{\boldsymbol{x}} \mid \boldsymbol{x}) = \frac{1}{\sqrt{\det(2\pi\sigma^2 \boldsymbol{I})}} \exp\left(\frac{-1}{2\sigma^2} \|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|_2^2\right)$$

Taking the natural logarithm of both sides:

$$\log q_\sigma(\tilde{\boldsymbol{x}} \mid \boldsymbol{x}) = \log \frac{1}{\sqrt{\det(2\pi\sigma^2 \boldsymbol{I})}} - \frac{-1}{2\sigma^2} \|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|_2^2$$

Hence:

$$\nabla_{\tilde{\boldsymbol{x}}} \log q_\sigma(\tilde{\boldsymbol{x}} \mid \boldsymbol{x}) = \frac{-(\tilde{\boldsymbol{x}} - \boldsymbol{x})}{\sigma^2}$$

Denoising score matching

The objective

$$\min_{\theta} \frac{1}{2} \boldsymbol{E}_{\boldsymbol{x}\sim P(\boldsymbol{x}),\, \tilde{\boldsymbol{x}}\sim q_\sigma(\tilde{\boldsymbol{x}}\mid \boldsymbol{x})} [\| s_\theta(\tilde{\boldsymbol{x}}) - \nabla_{\tilde{\boldsymbol{x}}} \log q_\sigma(\tilde{\boldsymbol{x}} \mid \boldsymbol{x}) \|_2^2]$$

can be written as:

$$\min_{\theta} \frac{1}{2} \boldsymbol{E}_{\boldsymbol{x}\sim P(\boldsymbol{x}),\, \tilde{\boldsymbol{x}}\sim q_\sigma(\tilde{\boldsymbol{x}}\mid \boldsymbol{x})} \left[\left\| s_\theta(\tilde{\boldsymbol{x}}) + \frac{\tilde{\boldsymbol{x}} - \boldsymbol{x}}{\sigma^2} \right\|_2^2 \right]$$

Denoising score matching: Training procedure

- Sample a minibatch of data points $\mathrm{B} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\} \sim P(x)$.
- Sample a minibatch of perturbed data points $\tilde{B} = \{\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2, \ldots, \tilde{\boldsymbol{x}}_m\} \sim q_\sigma(\tilde{\boldsymbol{x}} \mid \boldsymbol{x})$.
- Estimate the denoising score matching loss and optimize:

$$L_{DSM}\left(B, \tilde{B}, \theta\right) = \frac{1}{2m} \sum_{i=1}^{m} \left\| s_\theta(\tilde{\boldsymbol{x}}_i) + \frac{\tilde{\boldsymbol{x}}_i - \boldsymbol{x}_i}{\sigma^2} \right\|_2^2$$

- Repeat until convergence.

Denoising score matching: Sampling

- We can obtain samples from $q_\sigma(\tilde{\boldsymbol{x}})$ via Langevin dynamics.
- Given a noisy sample, how to get a 'clean' sample?
- Two natural approaches:
  - Choose the 'most likely' clean sample.

$$\boldsymbol{x}^* = \arg\max_{\boldsymbol{x}} p(\boldsymbol{x} \mid \tilde{\boldsymbol{x}})$$

Intractable!

  - Choose the expected clean sample.

$$\boldsymbol{x}^* = \boldsymbol{E}_{\boldsymbol{x} \sim P(\boldsymbol{x} \mid \tilde{\boldsymbol{x}})}[\boldsymbol{x}]$$

Here, $p(\boldsymbol{x} \mid \tilde{\boldsymbol{x}})$ is given by Bayes' formula:

$$p(\boldsymbol{x} \mid \tilde{\boldsymbol{x}}) = \frac{q_\sigma(\tilde{\boldsymbol{x}} \mid \boldsymbol{x}) p(\boldsymbol{x})}{q_\sigma(\tilde{\boldsymbol{x}})}$$

Denoising score matching: Sampling

- We consider the second approach (choosing the expected clean sample).

$$
\begin{aligned}
\boldsymbol{E}_{\boldsymbol{x}\sim P(\boldsymbol{x}\mid\tilde{\boldsymbol{x}})}[\boldsymbol{x}] &= \tilde{\boldsymbol{x}} + \boldsymbol{E}_{\boldsymbol{x}\sim P(\boldsymbol{x}\mid\tilde{\boldsymbol{x}})}[\boldsymbol{x} - \tilde{\boldsymbol{x}}] \\
&= \tilde{\boldsymbol{x}} + \int p(\boldsymbol{x}\mid\tilde{\boldsymbol{x}})(\boldsymbol{x} - \tilde{\boldsymbol{x}})d\boldsymbol{x} \\
&= \tilde{\boldsymbol{x}} + \int \frac{q_\sigma(\tilde{\boldsymbol{x}}\mid\boldsymbol{x})p(\boldsymbol{x})}{q_\sigma(\tilde{\boldsymbol{x}})}\sigma^2\nabla_{\tilde{\boldsymbol{x}}}\log q_\sigma(\tilde{\boldsymbol{x}}\mid\boldsymbol{x})\,d\boldsymbol{x} \\
&= \tilde{\boldsymbol{x}} + \sigma^2\int \frac{p(\boldsymbol{x})}{q_\sigma(\tilde{\boldsymbol{x}})}\nabla_{\tilde{\boldsymbol{x}}}q_\sigma(\tilde{\boldsymbol{x}}\mid\boldsymbol{x})d\boldsymbol{x} \\
&= \tilde{\boldsymbol{x}} + \frac{\sigma^2}{q_\sigma(\tilde{\boldsymbol{x}})}\nabla_{\tilde{\boldsymbol{x}}}\int p(\boldsymbol{x})\,q_\sigma(\tilde{\boldsymbol{x}}\mid\boldsymbol{x})d\boldsymbol{x} \\
&= \tilde{\boldsymbol{x}} + \frac{\sigma^2}{q_\sigma(\tilde{\boldsymbol{x}})}\nabla_{\tilde{\boldsymbol{x}}}q_\sigma(\tilde{\boldsymbol{x}}) \\
&= \tilde{\boldsymbol{x}} + \sigma^2\nabla_{\tilde{\boldsymbol{x}}}\log q_\sigma(\tilde{\boldsymbol{x}})
\end{aligned}
$$

Denoising score matching: Sampling

- **Tweedie's formula**:

$$\boldsymbol{E}_{\boldsymbol{x}\sim P(\boldsymbol{x}\mid\tilde{\boldsymbol{x}})}[\boldsymbol{x}] = \tilde{\boldsymbol{x}} + \sigma^2 \nabla_{\tilde{\boldsymbol{x}}} \log q_\sigma(\tilde{\boldsymbol{x}})$$

- After training our score-based model: $\nabla_{\tilde{\boldsymbol{x}}} \log p(\tilde{\boldsymbol{x}}) \approx s_\theta(\tilde{\boldsymbol{x}})$.

$$\boldsymbol{E}_{\boldsymbol{x}\sim P(\boldsymbol{x}\mid\tilde{\boldsymbol{x}})}[\boldsymbol{x}] \approx \tilde{\boldsymbol{x}} + \sigma^2 s_\theta(\tilde{\boldsymbol{x}})$$

- A suitable denoising strategy: follow the score (gradient)!

Sliced score matching

- Instead of working with the original score, we consider projections onto random directions.
- One dimensional problems should be much easier.
- **Objective:** minimizing the *sliced Fisher divergence.*

$$\min_{\theta} \frac{1}{2} \boldsymbol{E}_{\boldsymbol{v}\sim p_{\boldsymbol{v}}} \boldsymbol{E}_{\boldsymbol{x}\sim P(\boldsymbol{x})} \left[\left(\boldsymbol{v}^T \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}) - \boldsymbol{v}^T s_\theta(\boldsymbol{x}) \right)^2 \right]$$

- Again (for the third time!), we can transform the objective into a tractable one:

$$\min_{\theta} \boldsymbol{E}_{\boldsymbol{v}\sim p_{\boldsymbol{v}}} \boldsymbol{E}_{\boldsymbol{x}\sim P(\boldsymbol{x})} \left[\boldsymbol{v}^T \nabla_{\boldsymbol{x}} s_\theta(\boldsymbol{x}) \boldsymbol{v} + \frac{1}{2} \left(\boldsymbol{v}^T s_\theta(\boldsymbol{x}) \right)^2 \right]$$

  - The objective involves $\nabla_{\boldsymbol{x}} s_\theta(\boldsymbol{x})$, which seems unscalable.

Song et al., Sliced Score Matching: A Scalable Approach to Density and Score Estimation (2019)

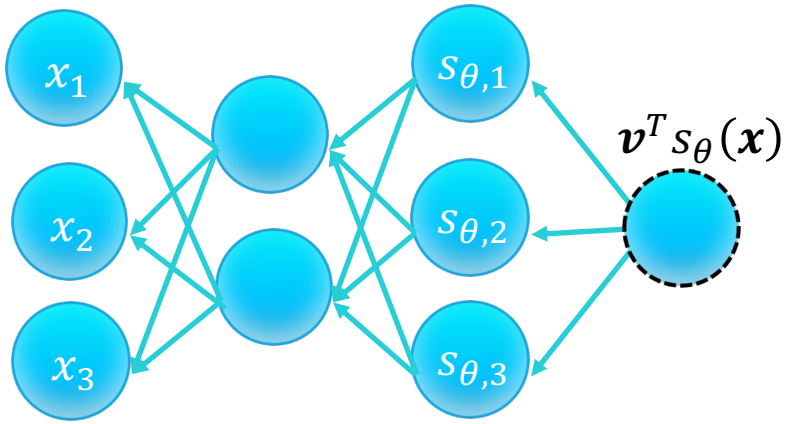
Sliced score matching

- A trick to reduce complexity:

$$\boldsymbol{v}^T \quad \nabla_{\boldsymbol{x}} s_\theta(\boldsymbol{x}) \quad \boldsymbol{v} = \boldsymbol{v}^T \nabla_{\boldsymbol{x}} (\boldsymbol{v}^T s_\theta(\boldsymbol{x}))$$

$O(d)$ ~~backprops~~ ~~$O(1)$ backprop~~!

→ Sliced score matching is scalable!



Sliced score matching: Training procedure

- Sample a minibatch of data points $\mathrm{B} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\} \sim P(\boldsymbol{x})$.
- Sample a minibatch of projection directions $\mathrm{V} = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_m\} \sim p_{\boldsymbol{v}}$. The projection distribution is usually standard Gaussian or multivariate Rademacher.
- Estimate the sliced score matching loss and optimize:

$$L_{SSM}(B, V, \theta) = \frac{1}{m} \sum_{i=1}^{m} \left[\boldsymbol{v_i}^T \nabla_{\boldsymbol{x}} s_\theta(\boldsymbol{x_i}) \boldsymbol{v_i} + \frac{1}{2} \left(\boldsymbol{v_i}^T s_\theta(\boldsymbol{x_i}) \right)^2 \right]$$

- Repeat until convergence.

Sliced score matching: A better estimator for training loss

- Recall the training objective:

$$\min_{\theta} \boldsymbol{E}_{\boldsymbol{v}\sim p_{\boldsymbol{v}}} \boldsymbol{E}_{\boldsymbol{x}\sim P(\boldsymbol{x})} \left[\boldsymbol{v}^T \boldsymbol{\nabla}_{\boldsymbol{x}} s_\theta(\boldsymbol{x}) \boldsymbol{v} + \frac{1}{2} \left(\boldsymbol{v}^T s_\theta(\boldsymbol{x}) \right)^2 \right]$$

- Denote $\boldsymbol{v} = [v_1, v_2, \dots, v_d]^T$ and $s_\theta(\boldsymbol{x}) = [s_\theta(\boldsymbol{x})_1, s_\theta(\boldsymbol{x})_2, \dots, s_\theta(\boldsymbol{x})_d]^T$.
- If the projection distribution is Standard Gaussian or multivariate Rademacher, then

$$\begin{aligned}
\boldsymbol{E}_{\boldsymbol{v}\sim p_{\boldsymbol{v}}} \left[\left(\boldsymbol{v}^T s_\theta(\boldsymbol{x}) \right)^2 \right] &= \boldsymbol{E}_{\boldsymbol{v}\sim p_{\boldsymbol{v}}} \left[\left(\sum_{i=1}^{d} v_i s_\theta(\boldsymbol{x})_i \right)^2 \right] \\
&= \boldsymbol{E}_{\boldsymbol{v}\sim p_{\boldsymbol{v}}} \left[\sum_{i=1}^{d} v_i^2 s_\theta(\boldsymbol{x})_i^2 + \sum_{1\le i\ne j\le d} v_i v_j s_\theta(\boldsymbol{x})_i s_\theta(\boldsymbol{x})_j \right] \\
&= \sum_{i=1}^{d} s_\theta(\boldsymbol{x})_i^2 \underbrace{\boldsymbol{E}_{v_i\sim p(v_i)}[v_i^2]}_{1} + \sum_{1\le i\ne j\le d} s_\theta(\boldsymbol{x})_i s_\theta(\boldsymbol{x})_j \underbrace{\boldsymbol{E}_{v_i,v_j\sim p(v_i,v_j)}[v_i v_j]}_{0} \\
&= \|s_\theta(\boldsymbol{x})\|_2^2
\end{aligned}$$

- The above transformations apply for all projection distributions of mean $\boldsymbol{0}$ and covariance $\boldsymbol{I}$.

Sliced score matching: A better estimator for training loss

- The objective can be rewritten as:

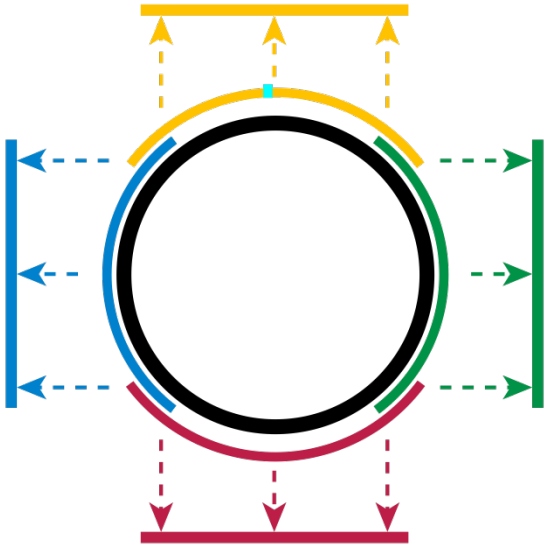
$$\min_{\theta} \boldsymbol{E}_{\boldsymbol{x}\sim P(\boldsymbol{x})}\left[\boldsymbol{E}_{\boldsymbol{v}\sim p_{\boldsymbol{v}}}[\boldsymbol{v}^T\nabla_{\boldsymbol{x}}s_\theta(\boldsymbol{x})\boldsymbol{v}] + \frac{1}{2}\|s_\theta(\boldsymbol{x})\|_2^2\right]$$

- We can train using an estimator with **reduced variance**:

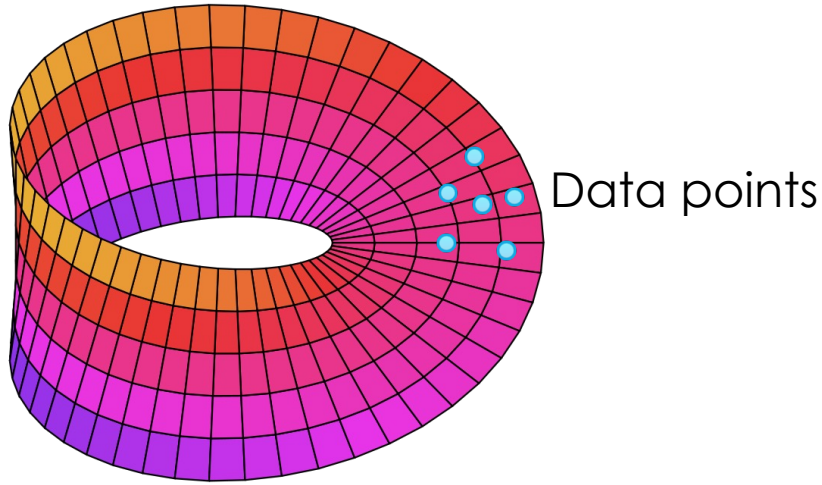
$$L_{SSM\_RV}(B,\theta) = \frac{1}{m}\sum_{i=1}^{m}\left[\boldsymbol{v_i}^T\nabla_{\boldsymbol{x}}s_\theta(\boldsymbol{x_i})\boldsymbol{v_i} + \frac{1}{2}\|s_\theta(\boldsymbol{x_i})\|_2^2\right]$$

Score matching: pitfalls

- For non-differentiable $\log p(\mathbf{x})$, the score is not defined!
- **Manifold hypothesis:** high-dimensional datasets actually lie along low-dimensional latent manifolds.



A circle is a 1-D manifold.
(Source: Manifold [Wikipedia])



Data points lie in low-dimensional manifolds
(Source: CS236, Stanford University)

Score matching: pitfalls

- Score matching fails in low data density regions.
- Recall the training objective of score matching:

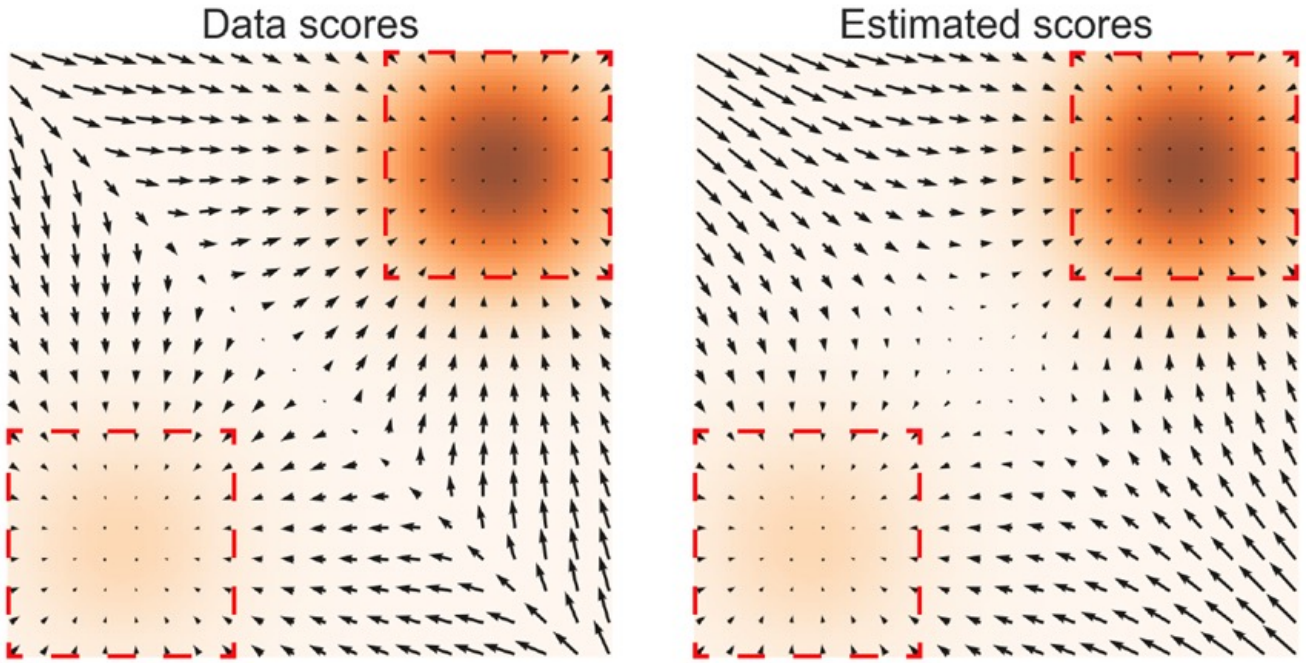
$$\frac{1}{2}\boldsymbol{E}_{\boldsymbol{x}\sim P(\boldsymbol{x})}[\|\nabla_{\boldsymbol{x}}\log p(\boldsymbol{x}) - s_\theta(\boldsymbol{x})\|_2^2]$$

$$= \frac{1}{2}\int p(\boldsymbol{x})\|\nabla_{\boldsymbol{x}}\log p(\boldsymbol{x}) - s_\theta(\boldsymbol{x})\|_2^2 d\boldsymbol{x}$$

- In regions where no data points are sampled ($p(\boldsymbol{x}) \approx 0$), score matching is inaccurate.

Score matching: pitfalls

- Score matching fails in low data density regions.
- Derailment when sampling by Langevin MCMC.



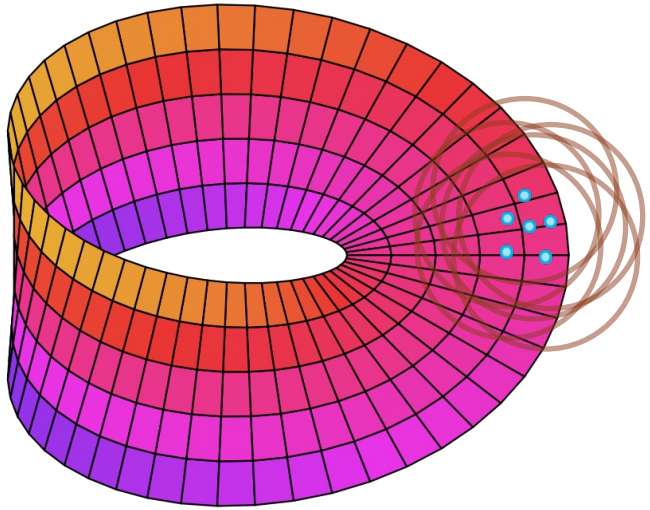
Using sliced score matching to estimate scores of a mixture of Gaussians

Song Y. & Ermon S., Generative Modeling by Estimating Gradients of the Data Distribution, NeurIPS 2019

Denoising score matching rises!

- Working on a Gaussian-perturbed distribution can solve all of the problems mentioned above.
- **Theorem**: *If X is a random variable and ϵ is an absolutely continuous random variable, then $X + \epsilon$ is an absolutely continuous random variable. As a result, the density of $X + \epsilon$ is differentiable almost everywhere.*

→ Solve the problem of undefined score function!



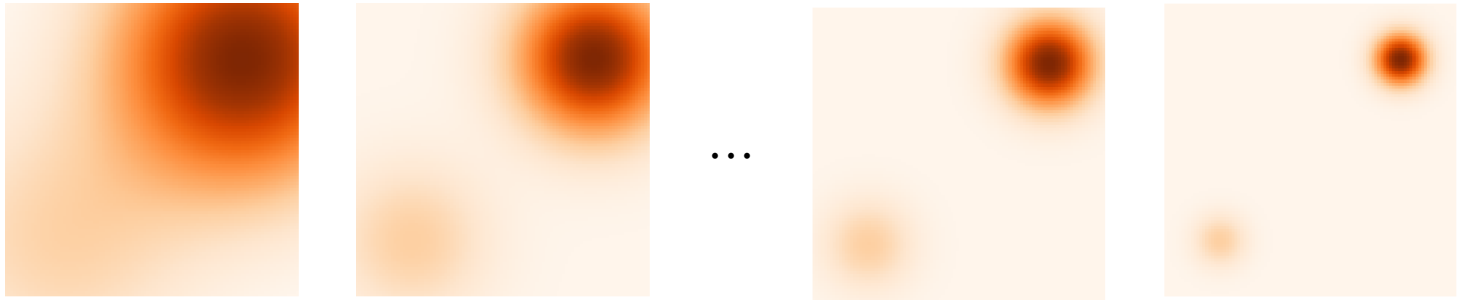
Source: CS236 – Stanford University

Arjovsky M. & Bottou L., Towards Principled Methods For Training Generative Adversarial Networks, ICLR 2017

Trade-off between data quality and estimation accuracy

- Working on a Gaussian-perturbed distribution can solve all of the problems mentioned above.
- But how much noise is enough?
  - High noise: covers more low density region but might over-corrupt the data.
  - Low noise: less corruption of the original data distribution, but reduces estimation accuracy.
- Solution: use multiple scales of noise perturbations.

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



Score-based models with multiple noise perturbations

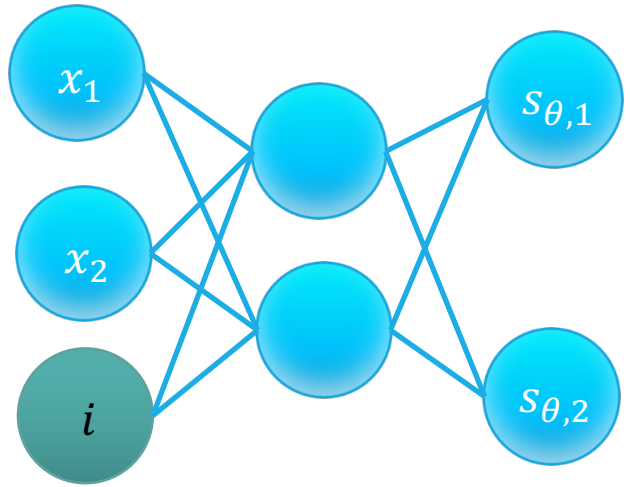
- Suppose there are L decreasing standard deviations $\sigma_1 > \sigma_2 > \cdots > \sigma_L$. For each Gaussian noise $N(\mathbf{0}, \sigma_i^2 \boldsymbol{I})$, its corresponding noise-perturbed distribution is given by:

$$p_{\sigma_i}(\tilde{\boldsymbol{x}}) = \int p(\boldsymbol{x}) N(\tilde{\boldsymbol{x}}; \boldsymbol{x}, \sigma_i^2 \boldsymbol{I}) d\boldsymbol{x}$$

- We want to train a **Noise Conditional Score Network (NCSN)** $s_\theta(\tilde{\boldsymbol{x}}, i)$, such that:

$$s_\theta(\tilde{\boldsymbol{x}}, i) \approx \nabla_{\tilde{\boldsymbol{x}}} \log p_{\sigma_i}(\boldsymbol{x})$$

- How to train and design a NCSN?



A noise conditional score network.

NCSN: objective

- For each σ_i, the denoising score matching objective is given by:

$$l(\theta, \sigma_i) = \frac{1}{2} \boldsymbol{E}_{\boldsymbol{x} \sim P(\boldsymbol{x})} \boldsymbol{E}_{\tilde{\boldsymbol{x}} \sim N(\boldsymbol{x}, \sigma_i^2 \boldsymbol{I})} \left[\left\| s_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}, i) + \frac{\tilde{\boldsymbol{x}} - \boldsymbol{x}}{\sigma_i^2} \right\|_2^2 \right]$$

- The objective of NCSN is a weighted sum of denoising score matching losses:

$$L\left(\theta, \{\sigma_i\}_{i=1}^L\right) = \frac{1}{L} \sum_{i=1}^L \lambda(\sigma_i) l(\theta, \sigma_i),$$

where $\lambda(\sigma_i) > 0$ is the weighting function.

- In practice, the weighting function is often chosen as $\lambda(\sigma_i) = \sigma_i^2$.

NCSN: training procedure

- Sample a minibatch of data points $\mathrm{B} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\} \sim P(\boldsymbol{x})$.
- Sample a minibatch of indices $T = \{i_1, i_2, \ldots, i_m\} \sim U\{1, 2, \ldots, L\}$, where U denotes the uniform distribution.
- Sample a minibatch of noise-perturbed samples $\tilde{B} = \{\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2, \ldots, \tilde{\boldsymbol{x}}_m\}$, where $\tilde{\boldsymbol{x}}_k \sim N(\boldsymbol{x}_k, \sigma_{i_k}^2 \boldsymbol{I})$.
- Estimate the loss and optimize with respect to θ:

$$L\left(B, T, \tilde{B}, \theta\right) = \frac{1}{2m} \sum_{k=1}^{m} \lambda(\sigma_{i_k}) \left\| s_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}_k, i_k) + \frac{\tilde{\boldsymbol{x}}_k - \boldsymbol{x}_{\boldsymbol{k}}}{\sigma_{i_k}^2} \right\|_2^2$$

- Repeat until convergence.

NCSN: sampling

Annealed Langevin dynamics:

- Sampling by running Langevin dynamics for $i = 1, 2, \ldots, L-1, L$ in sequence.
- Sample from p_{σ_i} is used as initialization for sampling from $p_{\sigma_{i+1}}$.

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

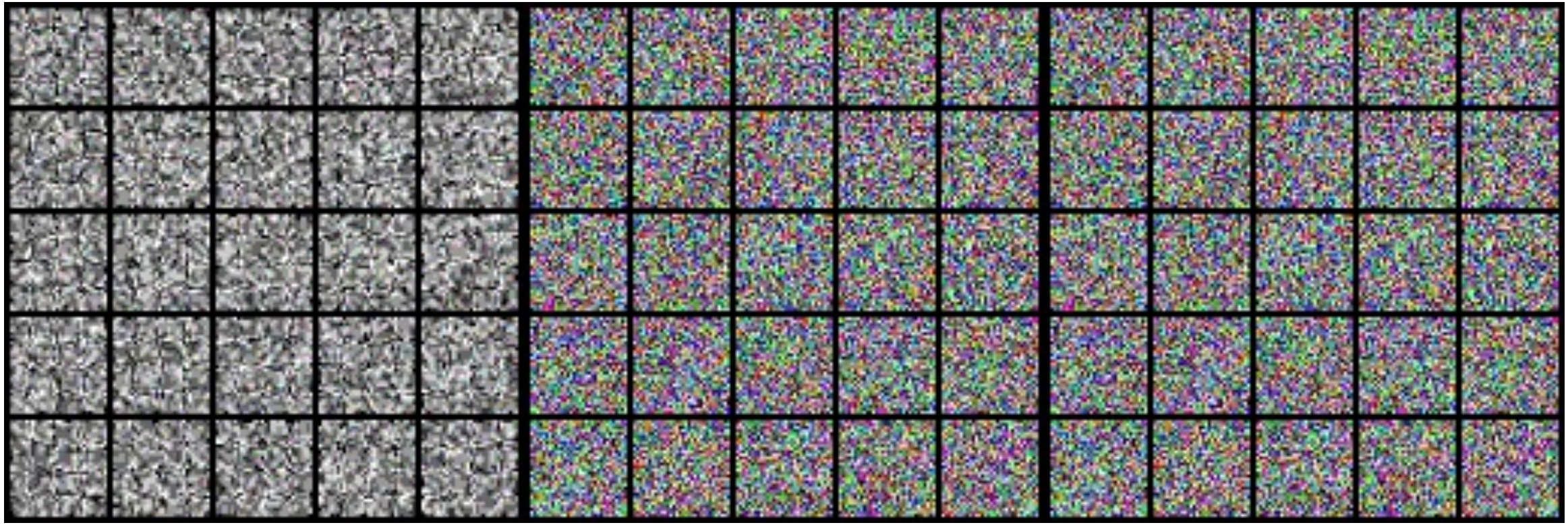
NCSN: choosing noise scales

Some recommendation from the authors:

- Choose $\{\sigma_i\}_{i=1}^{L}$ as a decreasing geometric progression. This ensures that the samples from high-density regions of $p_{\sigma_i}(\boldsymbol{x})$ are also likely to be in the high-density regions of $p_{\sigma_{i+1}}(\boldsymbol{x})$.
- Choose σ_1 as large as the maximum Euclidean distance between all pairs of training data points.

Song Y. & Ermon S., Improved Techniques for Training Score-Based Generative Models, NeurIPS 2020

NCSN: experiments



NCSN: experiments

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
CIFAR-10 Conditional		
EBM [12]	8.30	37.9
SNGAN [36]	8.60 ± .08	25.5
BigGAN [6]	**9.22**	**14.73**

Table 1: Inception and FID scores for CIFAR-10

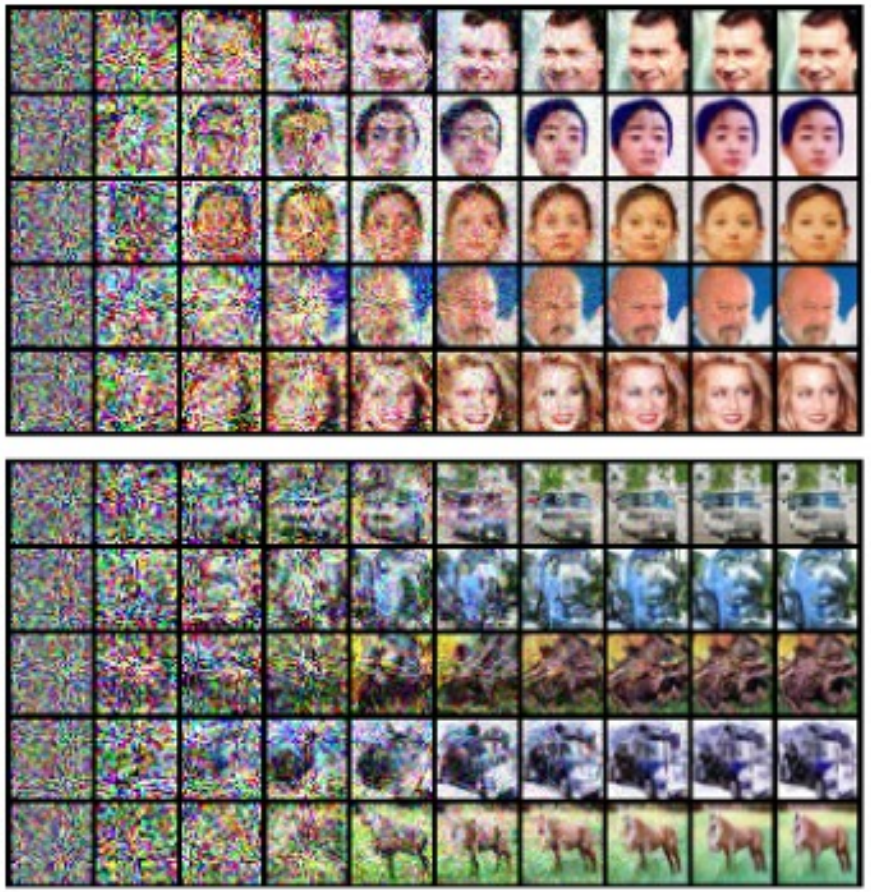


Figure 4: Intermediate samples of annealed Langevin dynamics.

Perturbing data with a diffusion process

- Let $\{\boldsymbol{x}(t) \in R^d\}_{t=0}^T$ be a diffusion process, indexed by the continuous time variable $t \in [0, T]$ and corresponds with a SDE of the form:

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

where:

- $\boldsymbol{f}(\cdot, t): R^d \to R^d$: drift term
- $g(t) \in R$: diffusion term
- $\boldsymbol{w}$: standard Brownian motion

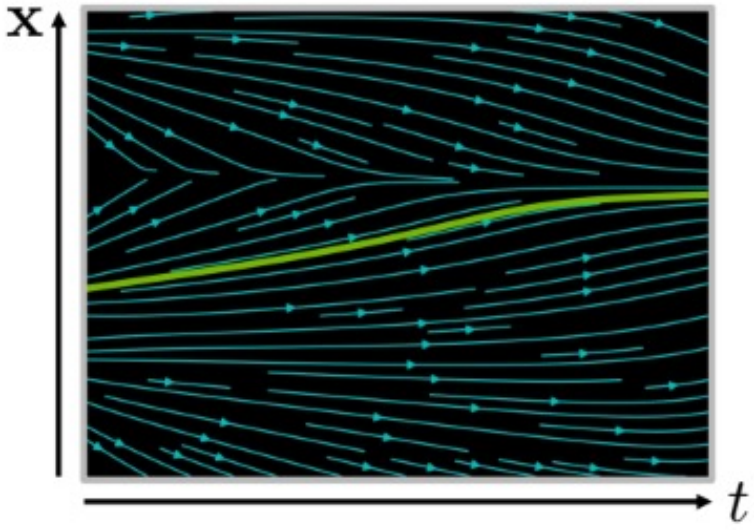
- Denote $p_t(\boldsymbol{x})$ be the probability distribution of $\boldsymbol{x}(t)$. We consider diffusion process such that:
  - $p_0 = P$
  - $p_T \approx \pi$, where π is a prior distribution that is easy to sample from.

Song et al., Score-based Generative Modeling Through Stochastic Differential Equations, ICLR 2021

Perturbing data with a diffusion process

Ordinary Differential Equation (ODE):

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}, t) \quad \text{or} \quad \mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t$$



Analytical Solution:

$$\mathbf{x}(t) = \mathbf{x}(0) + \int_0^t \mathbf{f}(\mathbf{x}, \tau)\mathrm{d}\tau$$

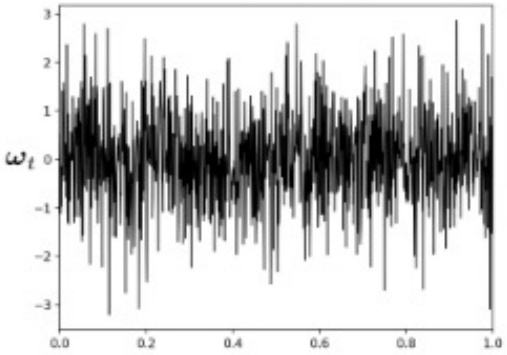
Iterative Numerical Solution:

$$\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \mathbf{f}(\mathbf{x}(t), t)\Delta t$$

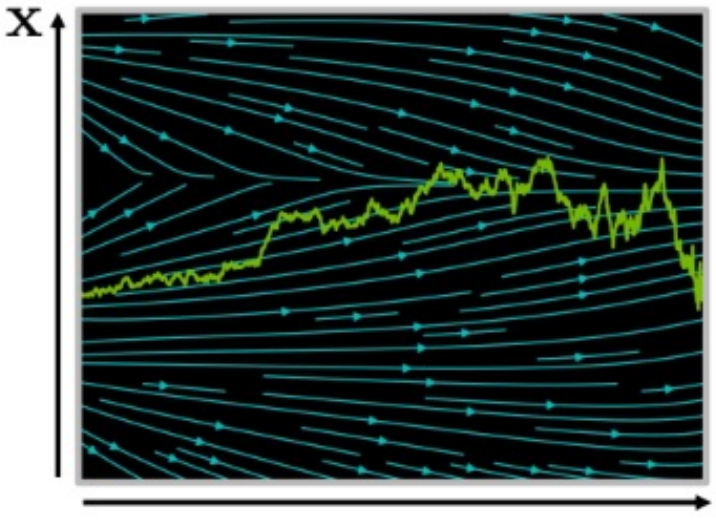
Stochastic Differential Equation (SDE):

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \underbrace{\mathbf{f}(\mathbf{x}, t)}_{\text{drift coefficient}} + \underbrace{\sigma(\mathbf{x}, t)\boldsymbol{\omega}_t}_{\text{diffusion coefficient}}$$

Wiener Process (Gaussian White Noise)



$$\left(\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + \sigma(\mathbf{x}, t)\mathrm{d}\boldsymbol{\omega}_t \right)$$



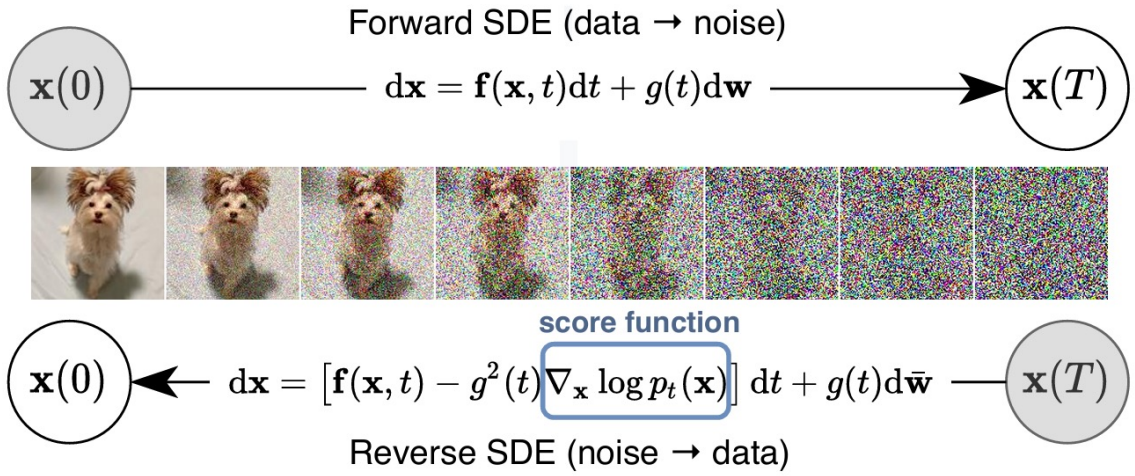
$$\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \mathbf{f}(\mathbf{x}(t), t)\Delta t + \sigma(\mathbf{x}(t), t)\sqrt{\Delta t}\,\mathcal{N}(\mathbf{0}, \mathbf{I})$$

Reversing the diffusion process

- By starting from a sample from the prior distribution $\pi \approx p_T$ and reversing the diffusion process, we will be able to obtain a sample from p_0.
- The reverse process is also a diffusion process running backwards in time, given by the following SDE:

$$d\boldsymbol{x} = [\boldsymbol{f}(\boldsymbol{x}, t) - g^2(t)\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})]dt + g(t)d\bar{\boldsymbol{w}},$$

where $\bar{\boldsymbol{w}}$ is a Brownian motion in the reverse time direction, and dt represents an infinitesimal negative time step.



Score-based model with SDE: objective

- Train a time-dependent score-based model $s_\theta(\boldsymbol{x}, t)$ such that:

$$s_\theta(\boldsymbol{x}, t) \approx \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})$$

- **Objective:** weighted sum of denoising score matching objectives:

$$\min_\theta \boldsymbol{E}_{t\sim U(0,T)}\left[\lambda(t)\boldsymbol{E}_{\boldsymbol{x}(0)\sim p_0(\boldsymbol{x})}\boldsymbol{E}_{\boldsymbol{x}(t)\sim p_{0t}(\boldsymbol{x}(t)\mid \boldsymbol{x}(0))}\left[\left\|s_\theta(\boldsymbol{x}(t),t) - \nabla_{\boldsymbol{x}(t)}\log p_{0t}(\boldsymbol{x}(t)\mid \boldsymbol{x}(0))\right\|_2^2\right]\right],$$

where:

- $p_{0t}(\boldsymbol{x}(t) \mid \boldsymbol{x}(0))$: transition distribution from $p_0(\boldsymbol{x})$ to $p_t(\boldsymbol{x})$.
- $U(0,T)$: uniform distribution over $[0,T]$
- $\lambda(t) > 0$: weighting function.
  - A good choice is $\lambda(t) = g^2(t)$ (we'll discuss about this choice in later slides)

How to compute $\nabla_{\boldsymbol{x}(t)} \log p_{0t}(\boldsymbol{x}(t) \mid \boldsymbol{x}(0))$?

Designing the diffusion process

- Consider a diffusion process $\{\boldsymbol{x}(t) \in R^d\}_{t=0}^T$ given by the following SDE:

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

We need to design $\boldsymbol{f}$ and g such that $p_0 = P$ and $p_T \approx \pi$, where π is a prior distribution that is easy to sample from.

- Some ways of designing the SDE:
  - Variance Exploding (VE) SDE

$$d\boldsymbol{x} = \sqrt{\frac{d[\sigma^2(t)]}{dt}}d\boldsymbol{w}$$

  - Variance Preserving (VP) SDE

$$d\boldsymbol{x} = -\frac{1}{2}\beta(t)\boldsymbol{x}dt + \sqrt{\beta(t)}d\boldsymbol{w}$$

  - Sub-VP SDE

$$d\boldsymbol{x} = -\frac{1}{2}\beta(t)\boldsymbol{x}dt + \sqrt{\beta(t)(1 - e^{-2\int_0^t \beta(s)ds})}\, d\boldsymbol{w}$$

VE SDE

$$d\boldsymbol{x} = \sqrt{\frac{d[\sigma^2(t)]}{dt}}\, d\boldsymbol{w}$$

- Euler-Maruyama (discretizing the above SDE): for $\Delta t \approx 0$ and $\boldsymbol{z}(t) \sim N(\boldsymbol{0}, \boldsymbol{I})$.

$$\begin{aligned}
\boldsymbol{x}(t+\Delta t) &\approx \boldsymbol{x}(t) + \sqrt{\frac{\sigma^2(t+\Delta t) - \sigma^2(t)}{\Delta t}} \cdot \big(\boldsymbol{w}(t+\Delta t) - \boldsymbol{w}(t)\big) \\
&\approx \boldsymbol{x}(t) + \sqrt{\frac{\sigma^2(t+\Delta t) - \sigma^2(t)}{\Delta t}} \cdot \sqrt{\Delta t}\boldsymbol{z}(t) \\
&\approx \boldsymbol{x}(t) + \sqrt{\sigma^2(t+\Delta t) - \sigma^2(t)}\boldsymbol{z}(t)
\end{aligned}$$

- Discretized version: $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \sqrt{\sigma_{i+1}^2 - \sigma_i^2}\boldsymbol{z}_i$, where $\boldsymbol{z}_i \sim N(\boldsymbol{0}, \boldsymbol{I})$ and $\sigma_0 = 0$.
- By induction, one can show that: $p(\boldsymbol{x}_i \mid \boldsymbol{x}_0) = N(\boldsymbol{x}_0, \sigma_i^2 \boldsymbol{I})$ for all i..

→ Similar to score-based models with multiple noise perturbations!

VP SDE

$$d\boldsymbol{x} = -\frac{1}{2}\beta(t)\boldsymbol{x}dt + \sqrt{\beta(t)}d\boldsymbol{w}$$

- Euler-Maruyama (discretizing the above SDE): for $\Delta t \approx 0$ and $\boldsymbol{z}(t) \sim N(\boldsymbol{0}, \boldsymbol{I})$:

$$\begin{aligned}\boldsymbol{x}(t+\Delta t) &\approx \boldsymbol{x}(t) - \frac{1}{2}\beta(t)\boldsymbol{x}(t)\Delta\mathrm{t} + \sqrt{\beta(t)}\big(\boldsymbol{w}(t+\Delta t) - \boldsymbol{w}(t)\big) \\ &\approx \left(1 - \frac{1}{2}\beta(t)\Delta\mathrm{t}\right)\boldsymbol{x}(t) + \sqrt{\beta(t)\Delta t}\,\boldsymbol{z}(t)\end{aligned}$$

- By Taylor expansion: $1 - \frac{1}{2}\beta(t)\Delta\mathrm{t} \approx \sqrt{1-\beta(t)\Delta t}$. Hence:

$$\begin{aligned}\boldsymbol{x}(t+\Delta t) &\approx \sqrt{1-\beta(t)\Delta t}\,\boldsymbol{x}(t) + \sqrt{\beta(t)\Delta t}\,\boldsymbol{z}(t) \\ &\approx \sqrt{1-\bar{\beta}(t)}\,\boldsymbol{x}(t) + \sqrt{\bar{\beta}(t)}\boldsymbol{z}(t),\end{aligned}$$

where we denote $\bar{\beta}(t) = \beta(t)\Delta t$.

- Discretized version: $\boldsymbol{x}_{i+1} = \sqrt{1-\bar{\beta}_i}\boldsymbol{x}_i + \sqrt{\bar{\beta}_i}\boldsymbol{z}_i$, where $\boldsymbol{z}_i \sim N(\boldsymbol{0}, \boldsymbol{I})$.

→ Similar to Denoising Diffusion Probabilistic Models (DDPM)!

Designing the diffusion process

For VE, VP and sub-VP SDEs, their corresponding transition distributions $p_{0t}\big(\boldsymbol{x}(t) \mid \boldsymbol{x}(0)\big)$ are all Gaussian and given as follows:

- For VE SDE:

$$p_{0t}\big(\boldsymbol{x}(t) \mid \boldsymbol{x}(0)\big) = N(\boldsymbol{x}(t); \boldsymbol{x}(0), [\sigma^2(t) - \sigma^2(0)]\boldsymbol{I})$$

- For VP SDE:

$$p_{0t}\big(\boldsymbol{x}(t) \mid \boldsymbol{x}(0)\big) = N\left(\boldsymbol{x}(t); \boldsymbol{x}(0)e^{-\frac{1}{2}\int_0^t \beta(s)ds}, \left[1 - e^{-\int_0^t \beta(s)ds}\right]\boldsymbol{I}\right)$$

- For sub-VP SDE:

$$p_{0t}\big(\boldsymbol{x}(t) \mid \boldsymbol{x}(0)\big) = N\left(\boldsymbol{x}(t); \boldsymbol{x}(0)e^{-\frac{1}{2}\int_0^t \beta(s)ds}, \left[1 - e^{-\int_0^t \beta(s)ds}\right]^2\boldsymbol{I}\right)$$

Sarkka S. & Solin A., Applied stochastic differential equations, volume 10, Cambridge University Press, 2019

Score-based model with SDE: Training procedure

- Sample a minibatch of data points $\mathrm{B} = \{\boldsymbol{x}_1(0), \boldsymbol{x}_2(0), \ldots, \boldsymbol{x}_m(0)\} \sim p_0(\boldsymbol{x})$.
- Sample a minibatch of time index $T = \{t_1, t_2, \ldots, t_m\} \sim U(0, T)$.
- Sample $\tilde{B} = \{\boldsymbol{x}_1(t_1), \boldsymbol{x}_2(t_2), \ldots, \boldsymbol{x}_m(t_m)\}$, where $\boldsymbol{x}_i(t_i) \sim p_{0t_i}\big(\boldsymbol{x}(t_i) \mid \boldsymbol{x}_i(0)\big)$
- Estimate the loss and optimize with respect to θ:

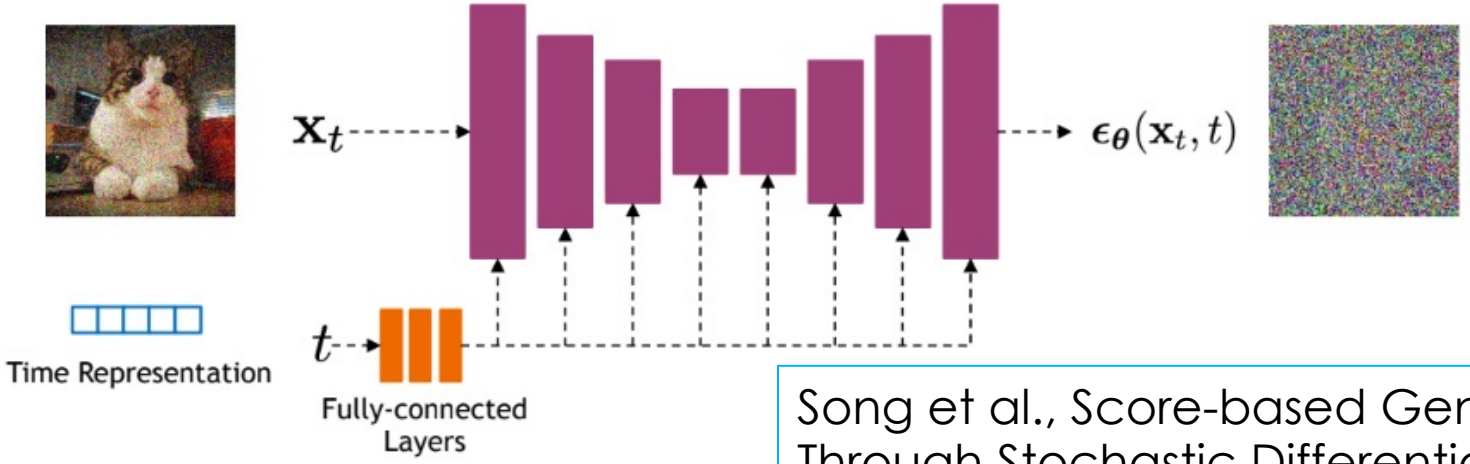
$$L_{SM\_SDE}\big(B, T, \tilde{B}, \theta\big) = \frac{1}{m}\sum_{i=1}^{m} \lambda(t_i)\left\| s_\theta(\boldsymbol{x}_i(t_i), t_i) - \nabla_{\boldsymbol{x}(t_i)} \log p_{0t_i}\big(\boldsymbol{x}_i(t_i) \mid \boldsymbol{x}_i(0)\big)\right\|$$

- Repeat until convergence.

Score-based model with NCSN & SDE: Network architecture

- Quite similar to the network architecture of DDPM: **U-Net architectures with ResNet blocks and self-attention layers**, with some additional improvements.
  Check out the original paper (cited in the bottom right corner) for more details!
- Time representation: using Gaussian random features.
  - Sample $\omega \sim N(\mathbf{0}, s^2\boldsymbol{I})$ and freeze ω (ω is not learnable).
  - For a time step t, the corresponding Gaussian random feature is defined as the vector:

$$[\sin(2\pi\omega t); \cos(2\pi\omega t)]$$



Song et al., Score-based Generative Modeling Through Stochastic Differential Equations, ICLR 2021

Score-based model with SDE: Sampling

- Recall the SDE of the reverse diffusion process:

$$d\boldsymbol{x} = [\boldsymbol{f}(\boldsymbol{x}, t) - g^2(t)\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})]dt + g(t)d\bar{\boldsymbol{w}}$$

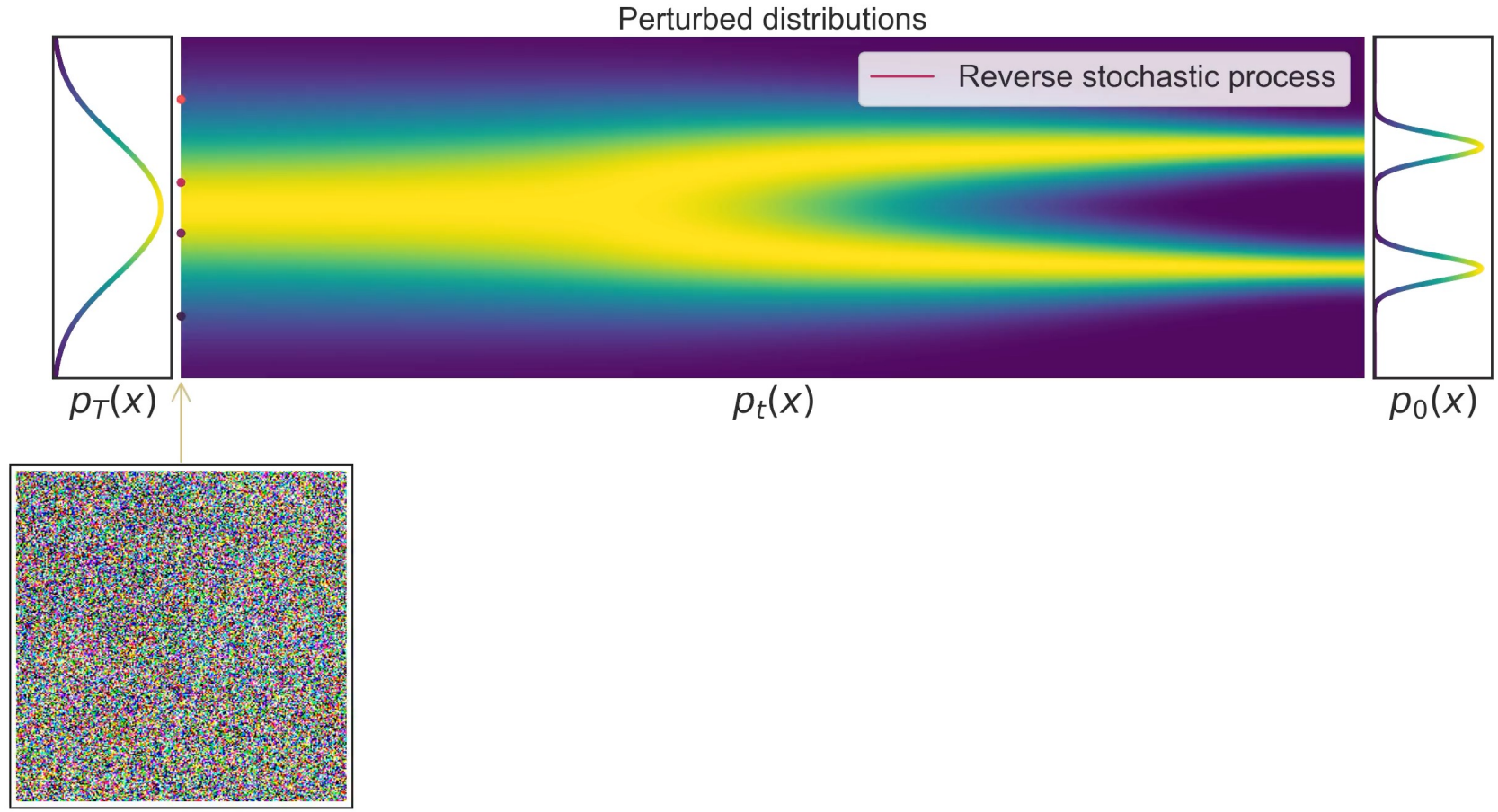
- **Sampling with Numerical SDE Solvers:**
  - Sample from the prior: $\mathbf{x}(T) \sim \pi$
  - Iteratively solve for $\mathbf{x}(0)$ by Euler-Maruyama approach: for $\Delta t \approx 0$:

$$\boldsymbol{x}_{t-\Delta \mathrm{t}} = \boldsymbol{x}_t - [\boldsymbol{f}(\boldsymbol{x}_t, t) - g^2(t)s_\theta(\boldsymbol{x}_t, \boldsymbol{t})]\Delta t + g(t)\sqrt{\Delta t}\boldsymbol{z}_t;$$
$$t \leftarrow t - \Delta t;$$

  where $\boldsymbol{z}_t \sim N(\mathbf{0}, \boldsymbol{I})$.

- Δt controls the trade-off between number of sampling steps and image quality.

Score-based model with SDE: Sampling



Score-based model with SDE: Sampling

Sampling with Predictor-Corrector method

- Apply one step of Numerical SDE solver to obtain $\boldsymbol{x}_{t-\Delta t}$ from $\boldsymbol{x}_t$ (predictor step)
- Apply several steps of Langevin MCMC to refine $\boldsymbol{x}_{t-\Delta t}$, such that $\boldsymbol{x}_{t-\Delta t}$ becomes a more accurate (higher density) sample from $p_{t-\Delta t}(\boldsymbol{x})$ (corrector step)

Algorithm 2 PC sampling (VE SDE)

$\mathbf{x}_N \sim \mathcal{N}(\mathbf{0}, \sigma_{\max}^2 \mathbf{I})$
for $i = N-1$ **to** 0 **do**
  $\mathbf{x}'_i \leftarrow \mathbf{x}_{i+1} + (\sigma_{i+1}^2 - \sigma_i^2)\mathbf{s}_{\boldsymbol{\theta}*}(\mathbf{x}_{i+1}, \sigma_{i+1})$
  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\mathbf{x}_i \leftarrow \mathbf{x}'_i + \sqrt{\sigma_{i+1}^2 - \sigma_i^2}\mathbf{z}$
  for $j = 1$ **to** M **do**
    $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    $\mathbf{x}_i \leftarrow \mathbf{x}_i + \epsilon_i \mathbf{s}_{\boldsymbol{\theta}*}(\mathbf{x}_i, \sigma_i) + \sqrt{2\epsilon_i}\mathbf{z}$
return $\mathbf{x}_0$

Algorithm 3 PC sampling (VP SDE)

$\mathbf{x}_N \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $i = N-1$ **to** 0 **do**
  $\mathbf{x}'_i \leftarrow (2 - \sqrt{1-\beta_{i+1}})\mathbf{x}_{i+1} + \beta_{i+1}\mathbf{s}_{\boldsymbol{\theta}*}(\mathbf{x}_{i+1}, i+1)$
  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\mathbf{x}_i \leftarrow \mathbf{x}'_i + \sqrt{\beta_{i+1}}\mathbf{z}$ — Predictor
  for $j = 1$ **to** M **do** — Corrector
    $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    $\mathbf{x}_i \leftarrow \mathbf{x}_i + \epsilon_i \mathbf{s}_{\boldsymbol{\theta}*}(\mathbf{x}_i, i) + \sqrt{2\epsilon_i}\mathbf{z}$
return $\mathbf{x}_0$

Score-based model with SDE: Sampling

Sampling with Probability Flow ODE:

- For all diffusion process, there exists a corresponding **deterministic process** whose trajectories share the same marginal probability densities. That is:

$$p_t^{SDE}(\boldsymbol{x}) = p_t^{ODE}(\boldsymbol{x}) \; \forall t \in [0, T]$$

This deterministic process satisfies an ODE (called 'probability flow ODE'), which is given as follows:

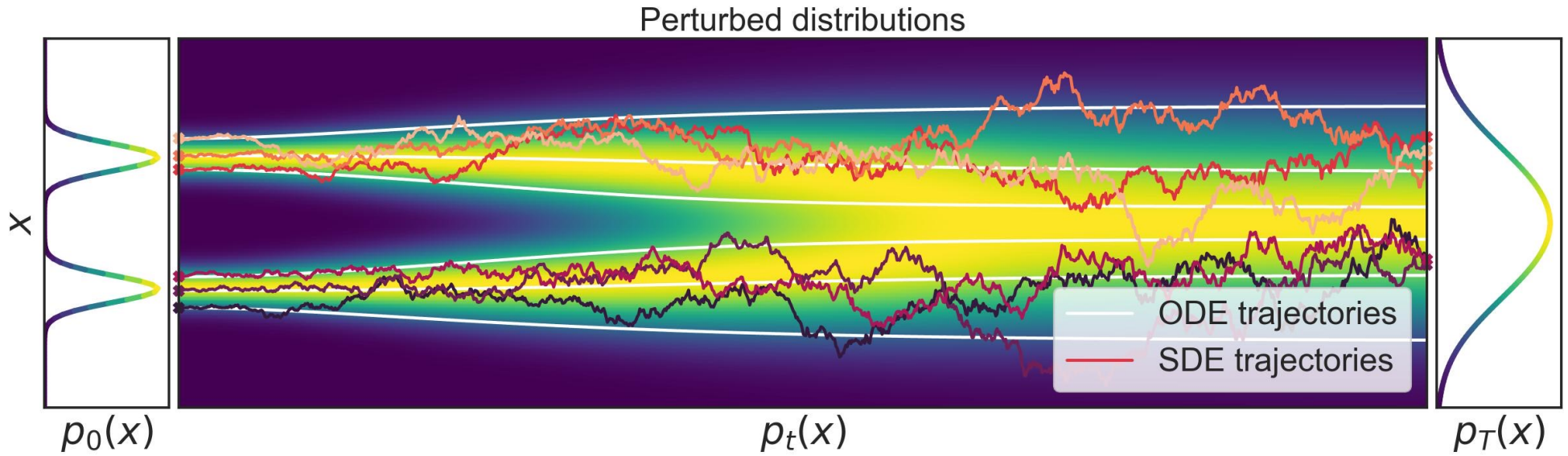
$$d\boldsymbol{x} = \left[f(\boldsymbol{x}, t) - \frac{1}{2} g(t)^2 \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}) \right] dt$$

- Since $s_\theta(\boldsymbol{x}, t) \approx \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})$, we can sample using the following ODE:

$$d\boldsymbol{x} = \left[f(\boldsymbol{x}, t) - \frac{1}{2} g(t)^2 s_\theta(\boldsymbol{x}, t) \right] dt$$

- We can sample $\boldsymbol{x}(T) \sim \pi$ and solve for $\boldsymbol{x}(0)$ using an ODE solver.

Score-based model with SDE: Sampling



Score-based model with SDE: Experiments

Table 2: NLLs and FIDs (ODE) on CIFAR-10.

Model	NLL Test ↓	FID ↓
RealNVP (Dinh et al., 2016)	3.49	-
iResNet (Behrmann et al., 2019)	3.45	-
Glow (Kingma & Dhariwal, 2018)	3.35	-
MintNet (Song et al., 2019b)	3.32	-
Residual Flow (Chen et al., 2019)	3.28	46.37
FFJORD (Grathwohl et al., 2018)	3.40	-
Flow++ (Ho et al., 2019)	3.29	-
DDPM (L) (Ho et al., 2020)	$\leqslant 3.70^{*}$	13.51
DDPM (L_{simple}) (Ho et al., 2020)	$\leqslant 3.75^{*}$	3.17
DDPM	3.28	3.37
DDPM cont. (VP)	3.21	3.69
DDPM cont. (sub-VP)	3.05	3.56
DDPM++ cont. (VP)	3.16	3.93
DDPM++ cont. (sub-VP)	3.02	3.16
DDPM++ cont. (deep, VP)	3.13	3.08
DDPM++ cont. (deep, sub-VP)	**2.99**	**2.92**

Table 3: CIFAR-10 sample quality.

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	$8.87 \pm .12$
NCSNv2 (Song & Ermon, 2020)	10.87	$8.40 \pm .07$
DDPM (Ho et al., 2020)	3.17	$9.46 \pm .11$
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**

Score-based model with SDE: Experiments

Figure 12: Samples on 1024×1024 CelebA-HQ from a modified NCSN++ model trained with the VE SDE.

Score-based model with SDE: Relation to likelihood learning

- Recall the probability flow ODE:

$$d\boldsymbol{x} = \tilde{f}_\theta(\boldsymbol{x}, t)dt$$

where $\tilde{f}_\theta(\boldsymbol{x}, t) = f(\boldsymbol{x}, t) - \frac{1}{2} g(t)^2 s_\theta(\boldsymbol{x}, t)$.

- By the instantaneous change of variable theorem, the log-likelihood of $p_0(\boldsymbol{x})$ can be computed as:

$$\log p_0\big(\boldsymbol{x}(0)\big) = \log p_T\big(\boldsymbol{x}(T)\big) + \int_0^T tr\big(\nabla_{\boldsymbol{x}(t)} \tilde{f}_\theta(\boldsymbol{x}(t), t)\big)dt$$

Theorem 1 (Instantaneous Change of Variables). *Let $\mathbf{z}(t)$ be a finite continuous random variable with probability $p(\mathbf{z}(t))$ dependent on time. Let $\frac{d\mathbf{z}}{dt} = f(\mathbf{z}(t), t)$ be a differential equation describing a continuous-in-time transformation of $\mathbf{z}(t)$. Assuming that f is uniformly Lipschitz continuous in $\mathbf{z}$ and continuous in t, then the change in log probability also follows a differential equation,*

$$\frac{\partial \log p(\mathbf{z}(t))}{\partial t} = -\mathrm{tr}\left(\frac{df}{d\mathbf{z}(t)}\right) \tag{8}$$

Chen et al., Neural Ordinary Differential Equations, NeurIPS 2018

Score-based model with SDE: Relation to likelihood learning

$$\log p_0\big(\boldsymbol{x}(0)\big) = \log p_T\big(\boldsymbol{x}(T)\big) + \int_0^T tr\big(\nabla_{\boldsymbol{x}(t)} \tilde{f}_\theta(\boldsymbol{x}(t), t)\big) dt$$

- The log-likelihood of $p_0(\boldsymbol{x})$ is computable, so is likelihood learning a great idea?
- Likelihood learning is computationally expensive:
  - Computing $tr\big(\nabla_{\boldsymbol{x}(t)} \tilde{f}_\theta(\boldsymbol{x}(t), t)\big)$ is costly.
  - Require calling an ODE solver for every optimization step, which is not suitable for large-scale score-based models.
- Fortunately, if we choose the weighting function to be $\lambda(t) = g^2(t)$, then under some regularity conditions:

$$D_{KL}(P||\, p_\theta) \leq \frac{T}{2} \boldsymbol{E}_{t\sim U(0,T)} \boldsymbol{E}_{p_t(\boldsymbol{x})} [\lambda(t) \|\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}) - s_\theta(\boldsymbol{x}, t)\|_2^2] + D_{KL}(p_T\, ||\pi)$$

- Hence, $\lambda(t) = g^2(t)$ is called the **likelihood weighting function.**

Song et al., Maximum Likelihood Training of Score-Based Diffusion Models, NeurIPS 2021

Score-based model with SDE: Pros and cons

- **Pros:**
  - Great image quality.
  - Flexible architecture choices for score network, as long as the number of dimensions of input and output are the same.
  - Provide exact likelihood computation (for score-based models with SDE).
- **Cons:** require a large number of sampling steps (hundreds or thousands).

DPM-Solver

- Previous work: using general-purpose ODE solvers for solving the probability flow ODE.
- **DPM-Solver:** dedicated solver for diffusion ODEs.
- Can generate high-quality samples in 10 to 20 steps; 4 – 16 times faster than previous state-of-the-art samplers.

Table 3: Sample quality measured by FID ↓ on CIFAR-10 dataset with continuous-time methods, varying the number of function evaluations (NFE).

Sampling method \ NFE			10	12	15	20	50	200	1000
CIFAR-10 (continuous-time model (VP deep) [3], linear noise schedule)									
SDE	Euler (denoise) [3]	$\epsilon = 10^{-3}$	304.73	278.87	248.13	193.94	66.32	12.27	**2.44**
		$\epsilon = 10^{-4}$	444.63	427.54	395.95	300.41	101.66	22.98	5.01
	Improved Euler [20]	$\epsilon = 10^{-3}$	82.42(NFE=48), 2.73(NFE=151), 2.44(NFE=180)						
ODE	RK45 Solver [28, 3]	$\epsilon = 10^{-3}$	19.55(NFE=26), 17.81(NFE=38), 3.55(NFE=62)						
		$\epsilon = 10^{-4}$	51.66(NFE=26), 21.54(NFE=38), 12.72(NFE=50), 2.61(NFE=62)						
	DPM-Solver (**ours**)	$\epsilon = 10^{-3}$	**4.70**	**3.75**	**3.24**	3.99	3.84 (NFE = 42)		
		$\epsilon = 10^{-4}$	6.96	4.93	3.35	**2.87**	**2.59 (NFE = 51)**		

Cheng et al., DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps, NeurIPS 2022

Consistency models

- Instead of solving the probability flow ODE, we learn a function that maps a point on the ODE trajectory to its origin.
- Support fast one-step generation, also allow multistep sampling to trade compute for sample quality.

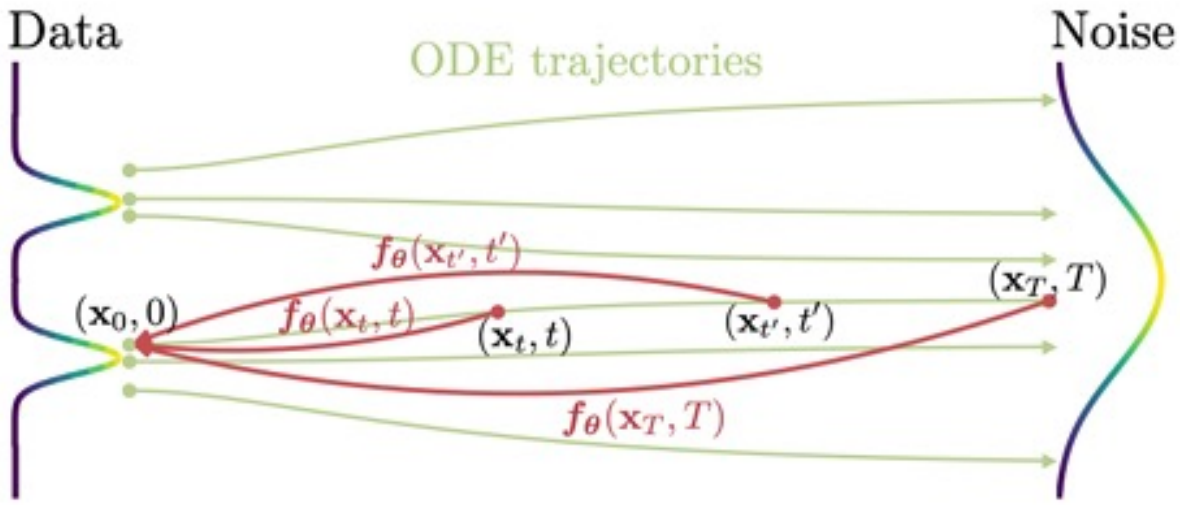


Figure 2: Consistency models are trained to map points on any trajectory of the PF ODE to the trajectory's origin.

Song et al., Consistency Models, ICML 2023

Some useful resources

- CS236 – Deep Generative Models, Stanford University
- Yang Song's blog - Generative Modeling by Estimating Gradients of the Data Distribution
- Training score-based models with SDEs on MNIST dataset using PyTorch