“Creating noise from data is easy; creating data from noise is generative modeling”

Source: Song, Yang, et al. "Score-based generative modeling through stochastic differential equations." arXiv preprint arXiv:2011.13456 (2020).

Score-based Generative Modeling Through Stochastic Differential Equations

ICLR outstanding paper award winner

Critical Summary

Natnael Daba

March 25, 2024

Authors

Dr. Yang Song*
Researcher
OpenAI

Dr. Jascha Sohl-Dickstein
Principal Scientist
Google DeepMind

Dr. Diederik P. Kingma
Research Scientist
Google Brain

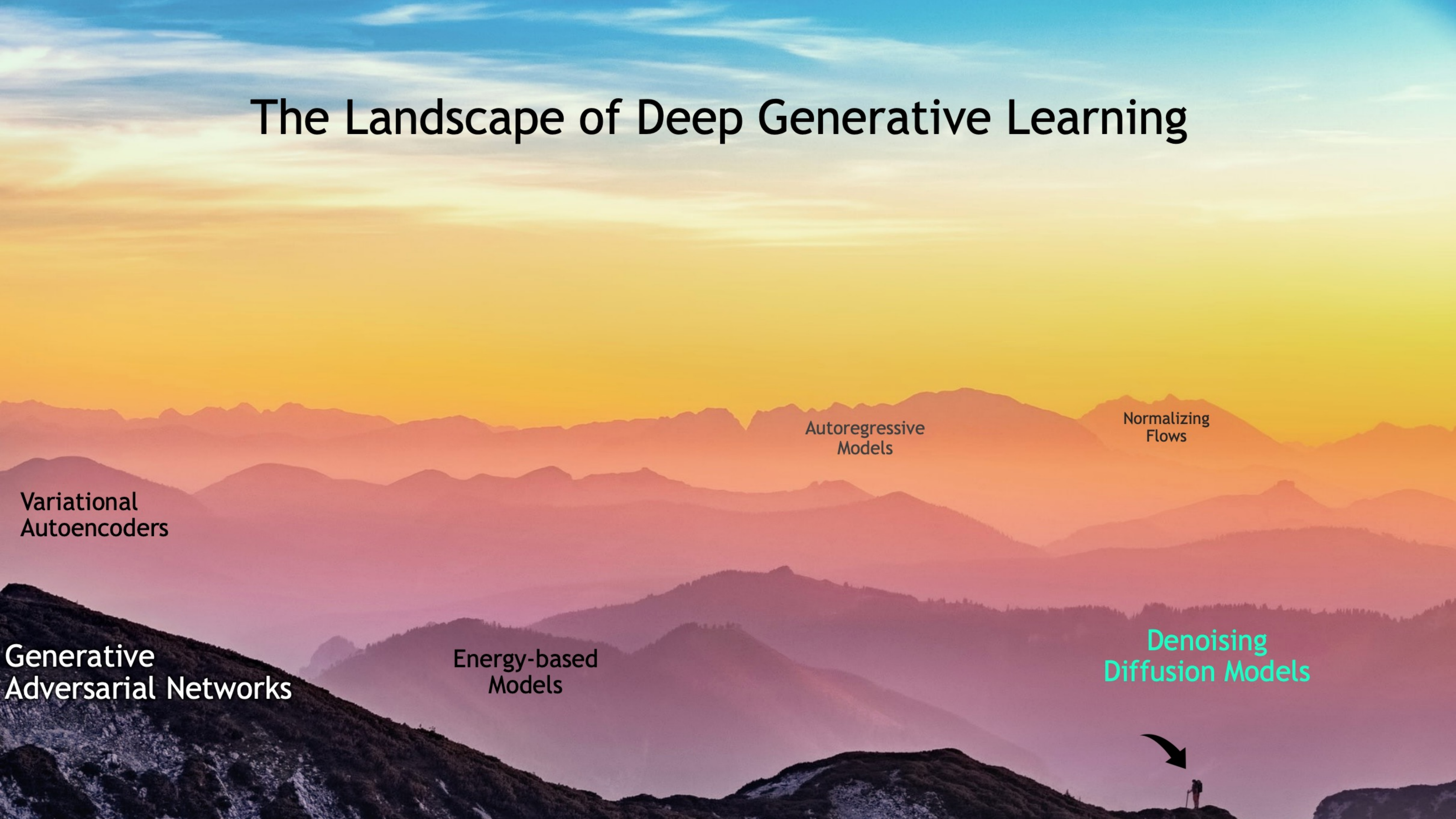
Dr. Abhishek Kumar
Research Scientist
Google Brain

Dr. Stefano Ermon
Associate Professor
Stanford University

Dr. Ben Poole
Research Scientist
Google Brain

*first author

Content

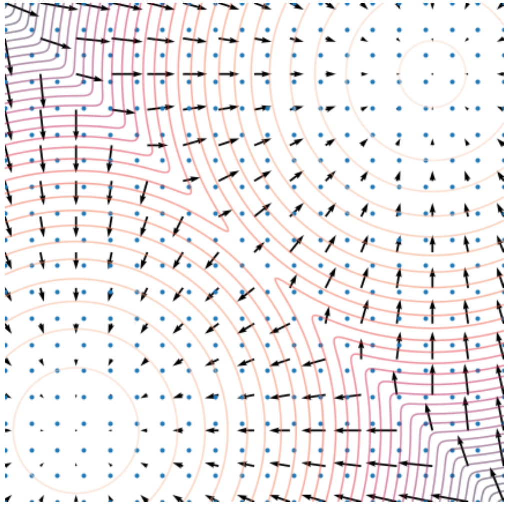

The Landscape of Deep Generative Learning
Autoregressive Models
Normalizing Flows
Variational Autoencoders
Generative Adversarial Networks
Energy-based Models
Denoising Diffusion Models

Introduction

Introduction

Two successful classes of probabilistic generative models employing corruption and then denoising of data

Score matching with Langevin dynamics (SMLD)



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

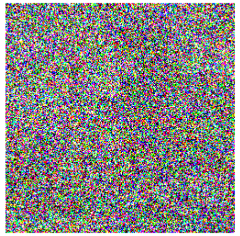
Langevin dynamics:

$$\mathbf{x}_0 \approx \pi(\mathbf{x})$$
$$\mathbf{x}_{i+1} \leftarrow \mathbf{x}_i + \varepsilon \nabla_{\mathbf{x}} \log p(\mathbf{x}) + \sqrt{2\varepsilon}\mathbf{z}_i$$
$$i = 0, 1, \dots, K,$$
$$\mathbf{z}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

Denoising diffusion probabilistic modeling (DDPM)

Training

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



Sampling

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

Introduction

Two successful classes of probabilistic generative models employing corruption and then denoising of data

Score matching with Langevin dynamics (SMLD)

Denoising diffusion probabilistic modeling (DDPM)

For continuous state spaces, DDPM implicitly computes scores at each noise scale (Song & Ermon, 2019).

Score-based generative models

Introduction

Goal: propose a unified framework that generalizes previous approaches through Stochastic Differential Equations (SDEs).

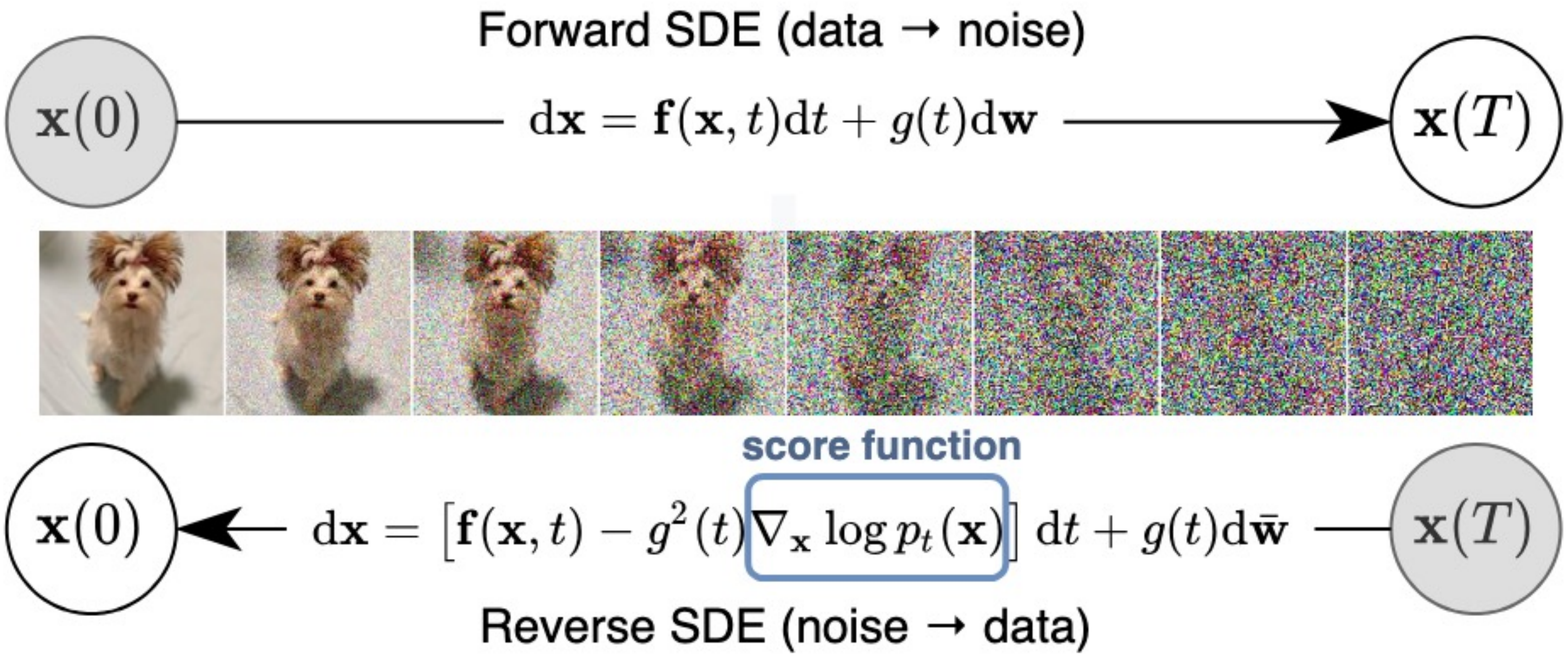
Why?

- To enable new and efficient sampling methods.
- Extend capabilities of score-based generative models.

How?

- Generalize number of noise scales to infinity.
- Formulate forward process with continuous noise scales using SDE.
- Derive reverse SDE (that describes reverse process) from forward SDE.
- Approximate reverse-time SDE by training a NN to estimate the score.

Introduction



Solving a reverse time SDE yields a score-based generative model. This SDE can be reversed if we know the score of the distribution at each intermediate timestep $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$

Background

Background: SMLD training

$p_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) := \mathcal{N}(\tilde{\mathbf{x}}; \mathbf{x}, \sigma^2 \mathbf{I})$ perturbation kernel

$\sigma_{\min} = \sigma_1 < \sigma_2 < \cdots < \sigma_N = \sigma_{\max}$ sequence of positive noise scales

$p_{\sigma_{\min}}(\mathbf{x}) \approx p_{\text{data}}(\mathbf{x})$ and $p_{\sigma_{\max}}(\mathbf{x}) \approx \mathcal{N}(\mathbf{x}; \mathbf{0}, \sigma_{\max}^2 \mathbf{I})$

Train a Noise Conditional Score Network (NCSN) $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, \sigma)$ with a sum of denoising score matching objectives:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{N} \sigma_i^2 \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{p_{\sigma_i}(\tilde{\mathbf{x}}|\mathbf{x})} \left[\|\mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}, \sigma_i) - \nabla_{\tilde{\mathbf{x}}} \log p_{\sigma_i}(\tilde{\mathbf{x}} \mid \mathbf{x})\|_2^2 \right]$$

Background: SMLD sampling

- For sampling, run M steps of Langevin MCMC to get a sample for each $p_{\sigma_i}(\mathbf{x})$ sequentially:

$$\mathbf{x}_i^m = \mathbf{x}_i^{m-1} + \epsilon_i \mathbf{s}_{\boldsymbol{\theta}*}(\mathbf{x}_i^{m-1}, \sigma_i) + \sqrt{2\epsilon_i}\mathbf{z}_i^m, \quad m = 1, 2, \cdots, M,$$

  where $\epsilon_i > 0$ is the step size, and $\mathbf{z}_i^m$ is standard normal.

- Repeat for $i = N, N-1, \cdots, 1$ with $\mathbf{x}_N^0 \sim \mathcal{N}(\mathbf{x} \mid \mathbf{0}, \sigma_{\max}^2 \mathbf{I})$ and $\mathbf{x}_i^0 = \mathbf{x}_{i+1}^M$ when $i < N$
- $\mathbf{x}_1^M$ becomes an exact sample from $p_{\sigma_{\min}}(\mathbf{x}) \approx p_{\text{data}}(\mathbf{x})$ as $M \to \infty$ and $\epsilon_i \to 0$ under some regularity conditions.

Background: DDPM training

- $p_{\alpha_i}(\mathbf{x}_i \mid \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_i; \sqrt{\alpha_i}\mathbf{x}_0, (1-\alpha_i)\mathbf{I})$; perturbation kernel
  where $\alpha_i := \prod_{j=1}^{i}(1-\beta_j)$
- $0 < \beta_1, \beta_2, \cdots, \beta_N < 1$; sequence of positive noise scales
- For each $\mathbf{x}_0 \sim p_{\text{data}}(\mathbf{x})$, a discrete Markov chain $\{\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_N\}$ is constructed s.t. $p_{\alpha_i}(\mathbf{x}_i \mid \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_i; \sqrt{\alpha_i}\mathbf{x}_0, (1-\alpha_i)\mathbf{I})$
- $p_{\boldsymbol{\theta}}(\mathbf{x}_{i-1}|\mathbf{x}_i) = \underbrace{\mathcal{N}(\mathbf{x}_{i-1}; \frac{1}{\sqrt{1-\beta_i}}(\mathbf{x}_i + \beta_i \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_i, i)), \beta_i\mathbf{I})}_{\text{Parametrization of the variational Markov chain in the reverse direction}}$

Parametrization of the variational
Markov chain in the reverse direction

Background: DDPM training

Trained with a re-weighted variant of the evidence lower bound (ELBO):

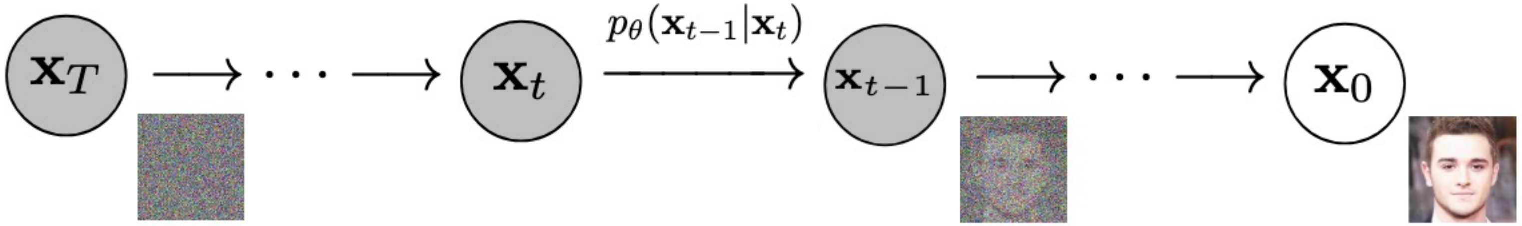
$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{N} (1-\alpha_i)\mathbb{E}_{p_{\mathrm{data}}(\mathbf{x})}\mathbb{E}_{p_{\alpha_i}(\tilde{\mathbf{x}}|\mathbf{x})}[\|\mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}, i) - \nabla_{\tilde{\mathbf{x}}} \log p_{\alpha_i}(\tilde{\mathbf{x}} \mid \mathbf{x})\|_2^2]$$

Background: DDPM sampling

Once we get optimal score model $\mathbf{s}_{\boldsymbol{\theta}^*}(\mathbf{x}, i)$ from training, samples can be generated by starting from $\mathbf{x}_N \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and following:

$$\mathbf{x}_{i-1} = \underbrace{\frac{1}{\sqrt{1-\beta_i}}(\mathbf{x}_i + \beta_i \mathbf{s}_{\boldsymbol{\theta}^*}(\mathbf{x}_i, i)) + \sqrt{\beta_i}\mathbf{z}_i, \quad i = N, N-1, \cdots, 1.}_{\textit{Ancestral sampling}}$$

since it amounts to performing ancestral sampling from the graphical model $\prod_{i=1}^{N} p_{\boldsymbol{\theta}}(\mathbf{x}_{i-1} \mid \mathbf{x}_i)$



Background: SMLD and DDPM comparison

- SMLD training:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{N} \sigma_i^2 \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{p_{\sigma_i}(\tilde{\mathbf{x}}|\mathbf{x})} \big[\| \mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}, \sigma_i) - \nabla_{\tilde{\mathbf{x}}} \log p_{\sigma_i}(\tilde{\mathbf{x}} \mid \mathbf{x}) \|_2^2 \big]$$

- DDPM training:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{N} (1 - \alpha_i) \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{p_{\alpha_i}(\tilde{\mathbf{x}}|\mathbf{x})} [\| \mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}, i) - \nabla_{\tilde{\mathbf{x}}} \log p_{\alpha_i}(\tilde{\mathbf{x}} \mid \mathbf{x}) \|_2^2]$$

Background: Stochastic Process

- **Stochastic process**: a sequence of random variables $\{X(t) : t \in T\}$ defined on a common probability space $(\Omega, \mathcal{F}, P)$

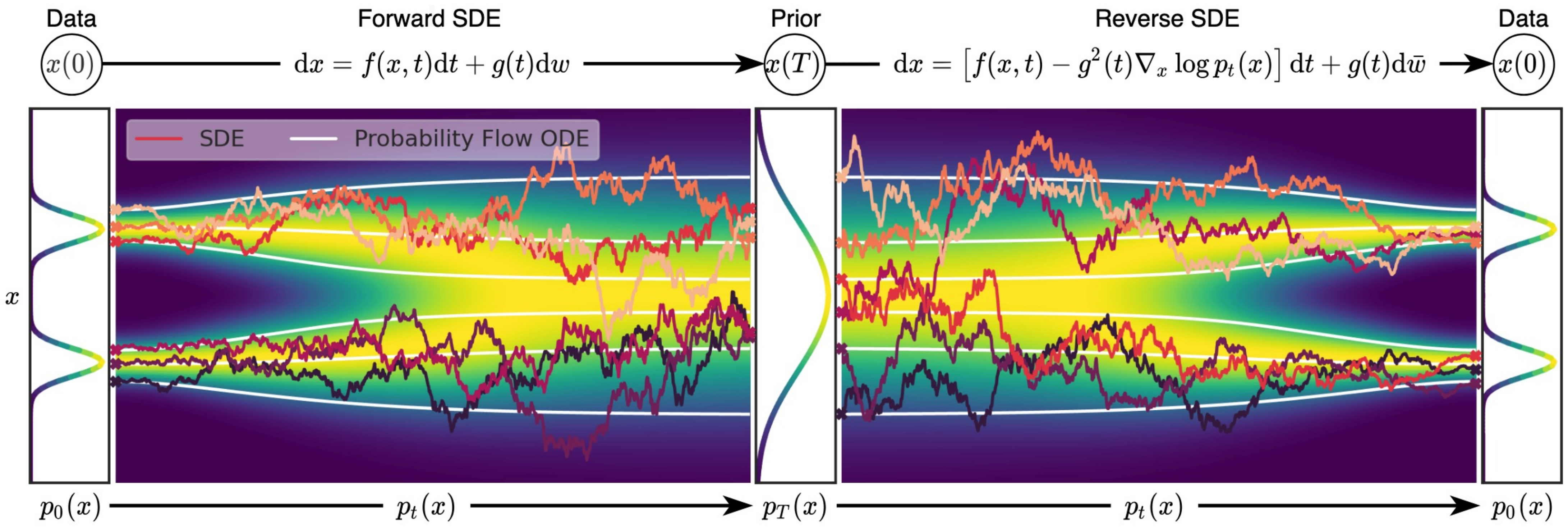
where:

T : parameter space or time space

$\mathrm{range}(X(t))$: state space

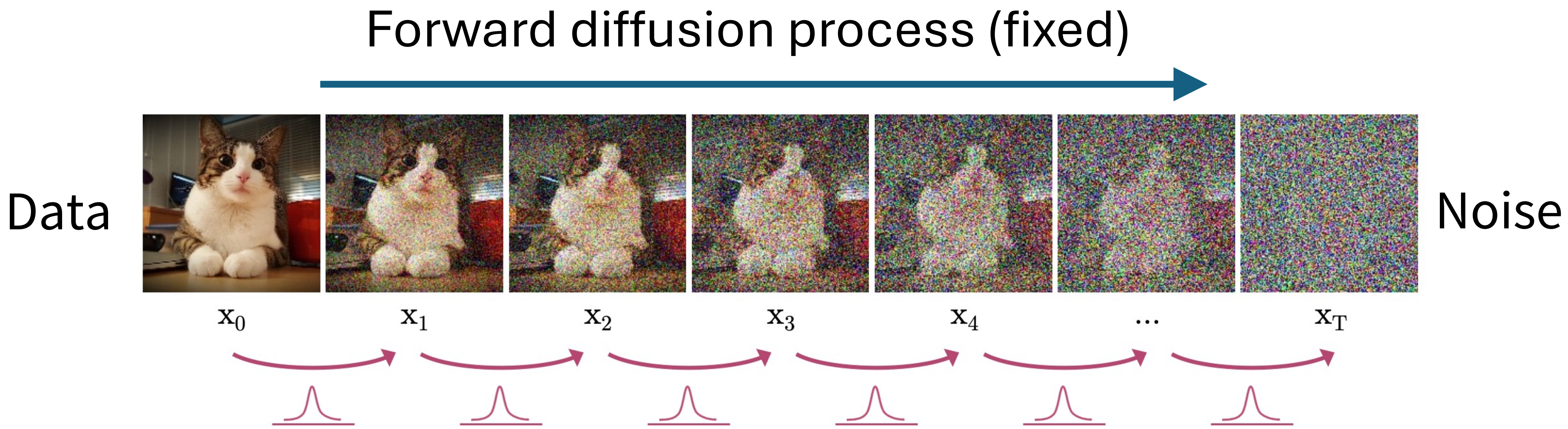
Score-based Generative Modeling with SDEs

Overview



Overview of score-based generative modeling through SDEs

Forward diffusion process



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

Forward diffusion process

Forward diffusion process (fixed)

Data  Noise

$\mathbf{x}_0$ $\mathbf{x}_1$ ... consider the limit of **many small steps** ... $\mathbf{x}_T$

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

$$\mathbf{x}_t = \sqrt{1-\beta_t}\,\mathbf{x}_{t-1} + \sqrt{\beta_t}\,\mathcal{N}(\mathbf{0},\mathbf{I})$$

$$= \sqrt{1-\beta(t)\Delta t}\,\mathbf{x}_{t-1} + \sqrt{\beta(t)\Delta t}\,\mathcal{N}(\mathbf{0},\mathbf{I}) \qquad (\beta_t := \beta(t)\Delta t)$$

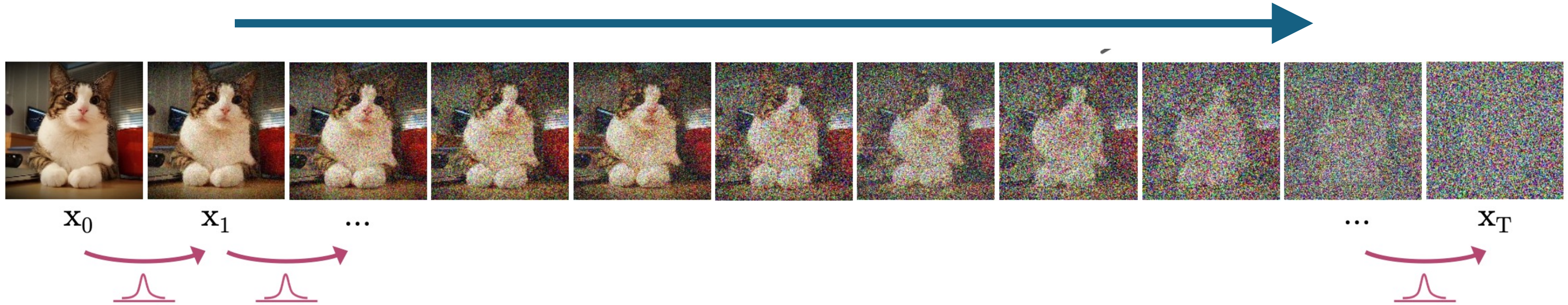
$$\approx \mathbf{x}_{t-1} - \frac{\beta(t)\Delta t}{2}\mathbf{x}_{t-1} + \sqrt{\beta(t)\Delta t}\,\mathcal{N}(\mathbf{0},\mathbf{I})$$

Taylor series expansion of $f(\Delta t) = \sqrt{1-\beta(t)\Delta t}$ around $\Delta t = 0$

Forward diffusion process

Forward diffusion process (fixed)

Data



Noise

$$\mathbf{x}_t \approx \mathbf{x}_{t-1} - \frac{\beta(t)\Delta t}{2}\mathbf{x}_{t-1} + \sqrt{\beta(t)\Delta t}\,\mathcal{N}(\mathbf{0}, \mathbf{I})$$

as $\Delta t \to 0$

$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\mathbf{x}_t\,\mathrm{d}t + \sqrt{\beta(t)}\mathrm{d}\boldsymbol{\omega}_t$$

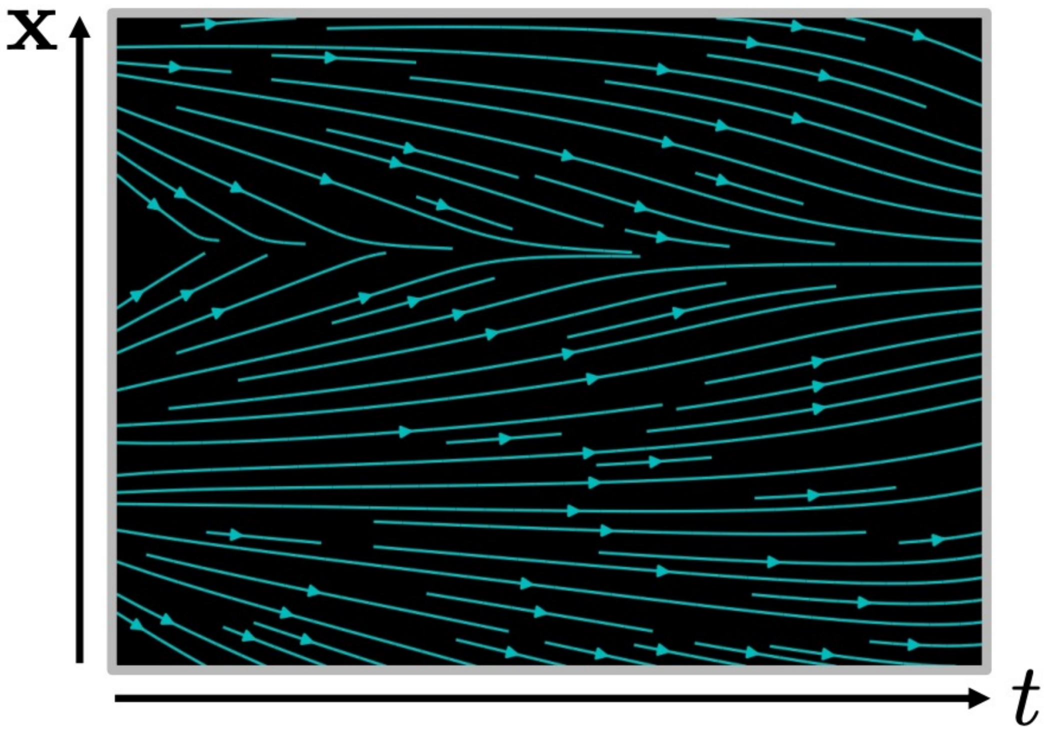
Stochastic Differential Equation (SDE)

describing the diffusion in infinitesimal limit

Differential Equations: A review

Ordinary Differential Equation (ODE):

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

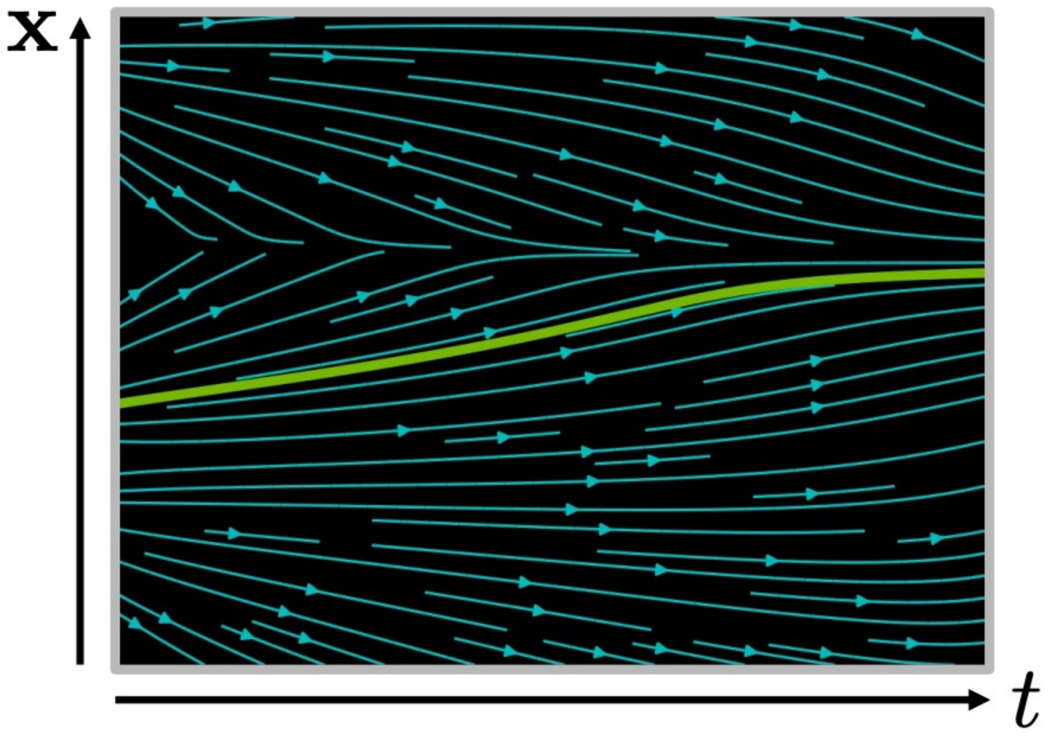


Differential Equations: A review

Ordinary Differential Equation (ODE):

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

- Highly complex non-linear function
- Integration might not be possible



Analytical Solution:

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

Iterative Numerical Solution:

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

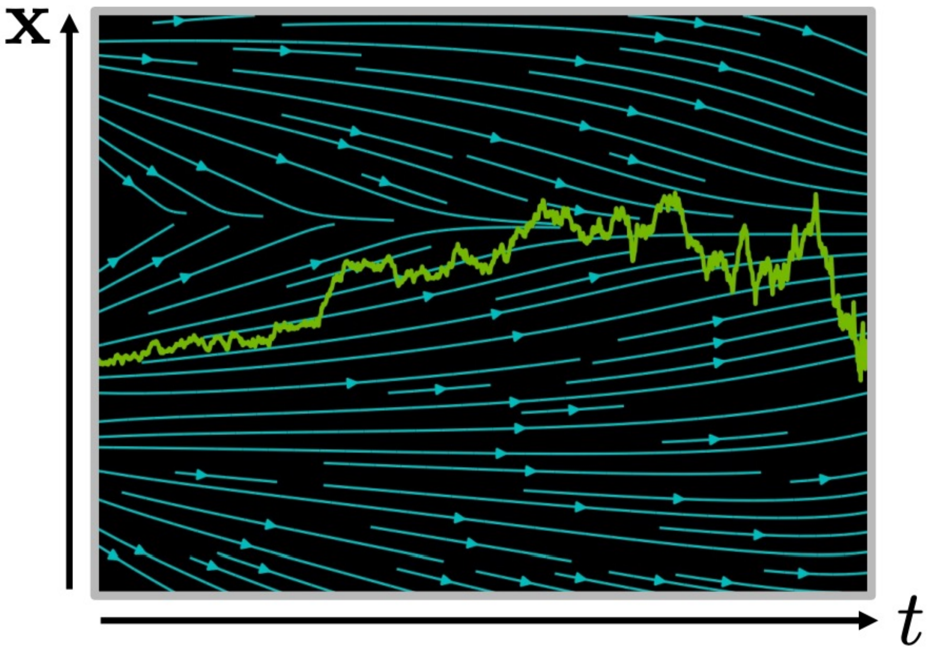
Differential Equations: A review

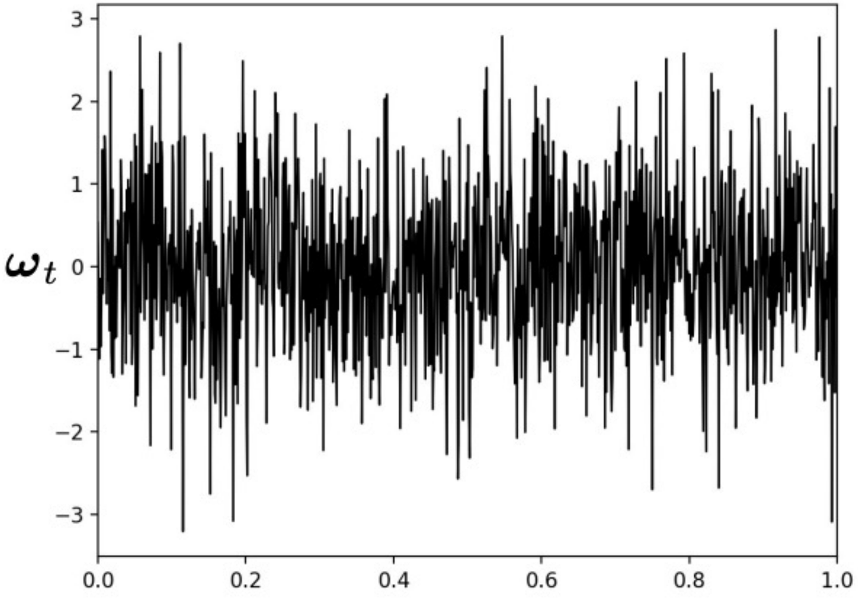
Stochastic Differential Equation (SDE):

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

Wiener Process (Gaussian White Noise)

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





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

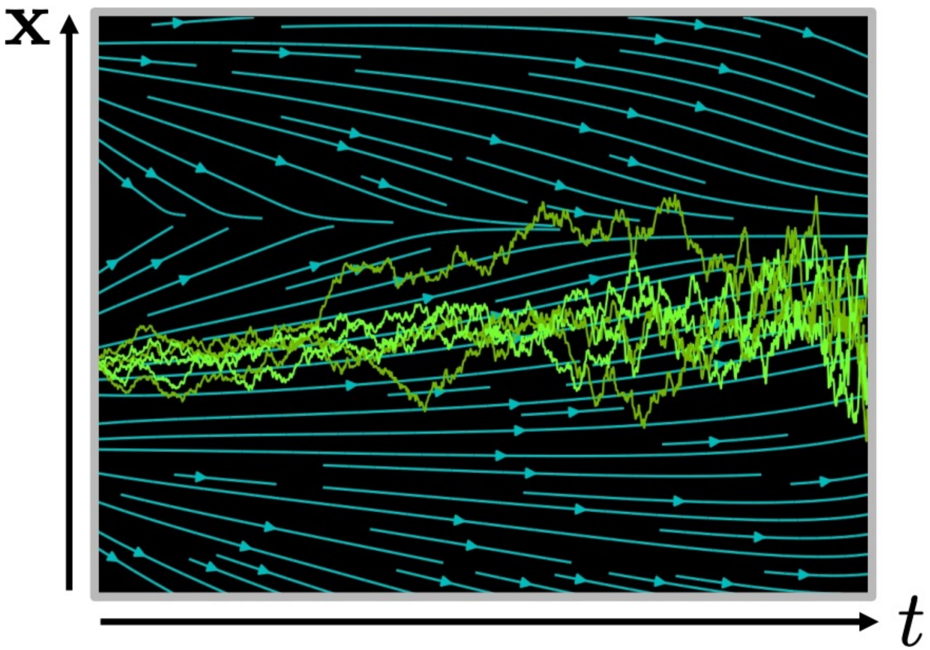
Differential Equations: A review

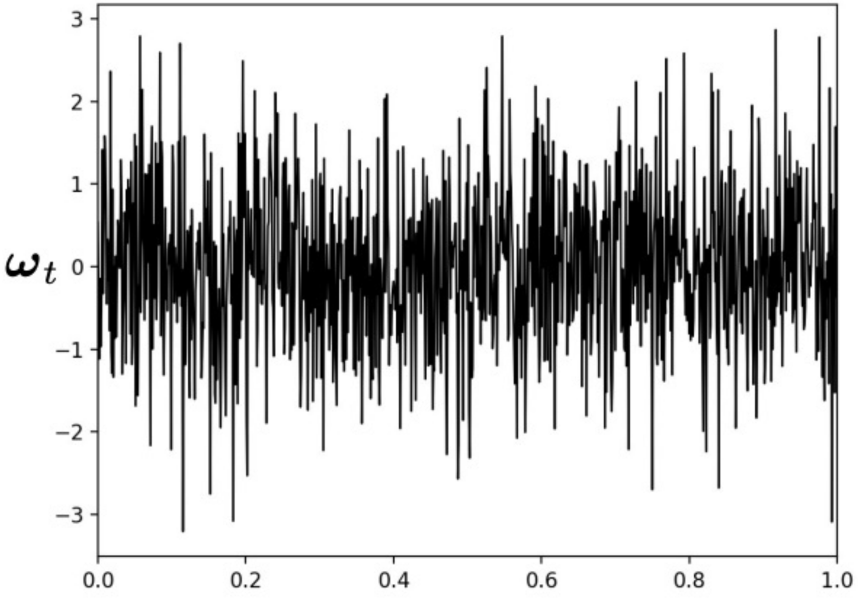
Stochastic Differential Equation (SDE):

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

Wiener Process (Gaussian White Noise)

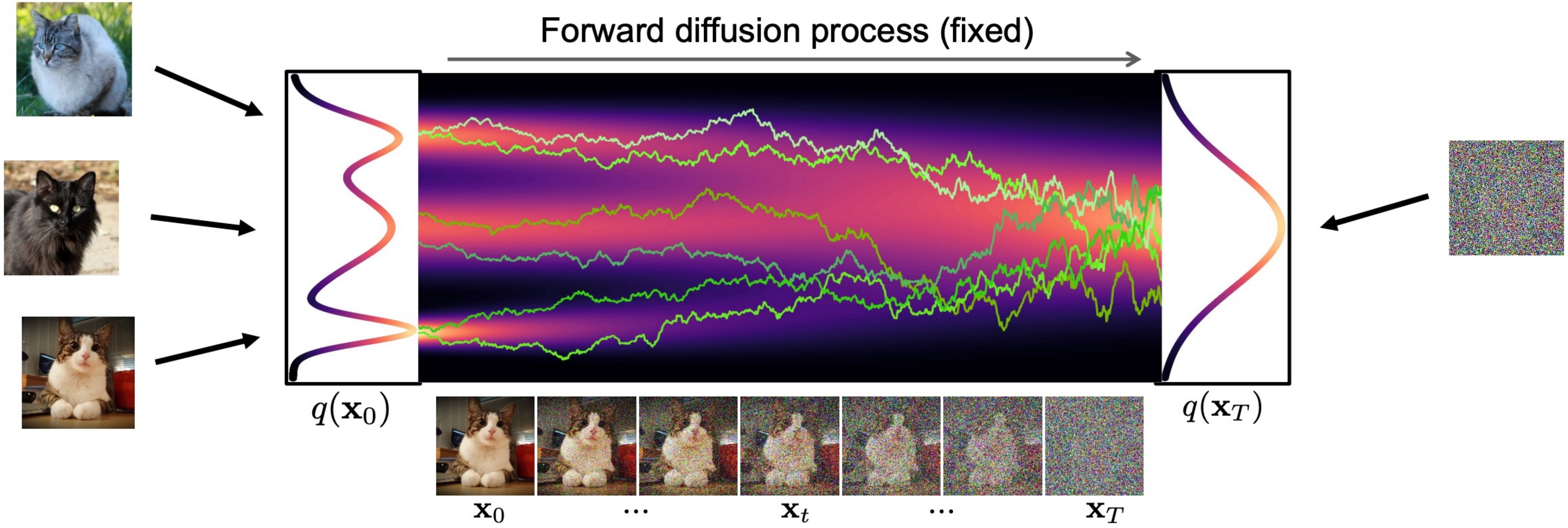
$$\left(\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\,\mathrm{d}t + \sigma(\mathbf{x}, t)\,\mathrm{d}\boldsymbol{\omega}_t\right)$$





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

Forward diffusion process as SDE

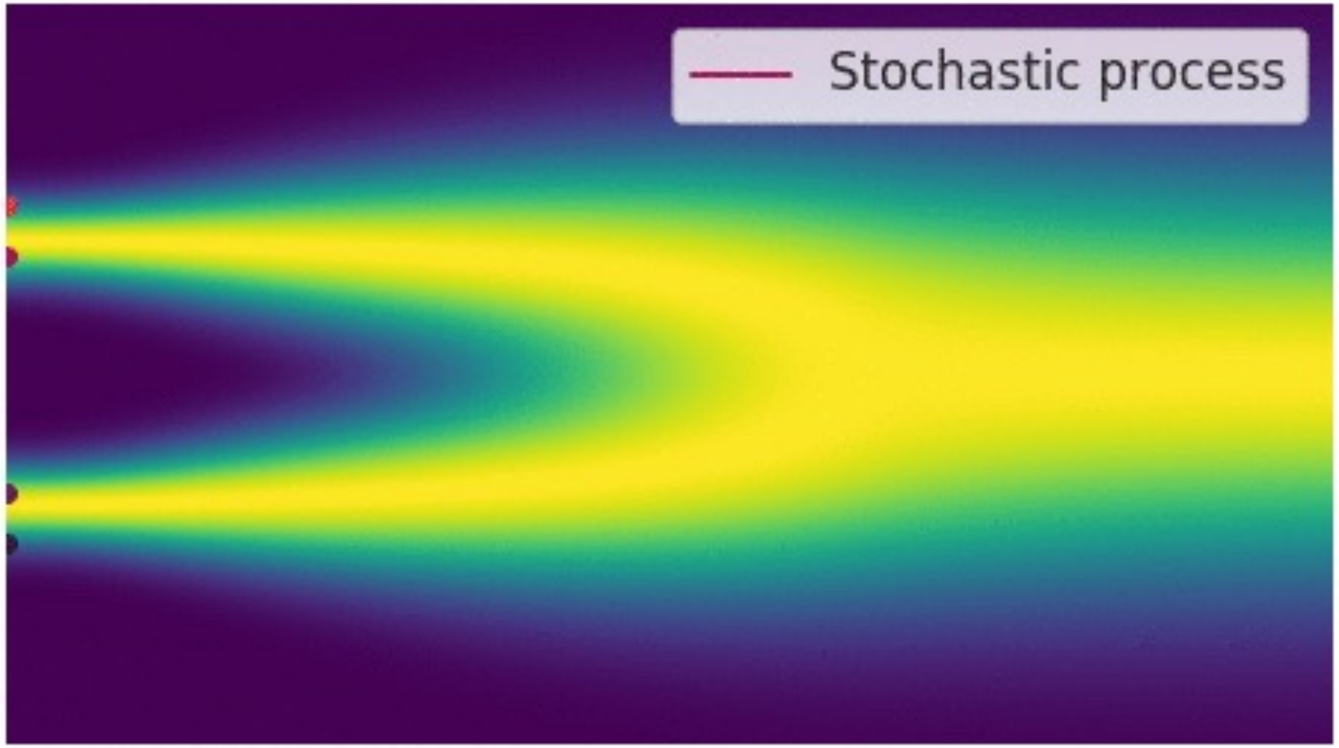


Forward Diffusion SDE:

$$\mathrm{d}\mathbf{x}_t = \underbrace{-\frac{1}{2}\beta(t)\mathbf{x}_t\,\mathrm{d}t}_{\substack{\text{drift term} \\ \text{(pulls towards mode)}}} + \underbrace{\sqrt{\beta(t)}\mathrm{d}\boldsymbol{\omega}_t}_{\substack{\text{diffusion term} \\ \text{(injects noise)}}}$$

Perturbing data with an SDE

Perturbing the data distribution with continuously growing levels of noise.



The noise perturbation procedure is a continuous-time stochastic process.

Source: https://yang-song.net/blog/2021/score/#introduction

Perturbing data with an SDE

- **Goal**: construct a diffusion process $\{\mathbf{x}(t)\}_{t=0}^{T}$ indexed by a continuous time variable $t \in [0, T]$ s.t.:

$$\mathbf{x}(0) \sim \boxed{p_0} \text{ and } \mathbf{x}(T) \sim \boxed{p_T}$$

Data distribution Prior distribution

- Many stochastic processes are solutions of stochastic differential equations (SDEs).

Perturbing data with an SDE

- Therefore, the diffusion process can be modeled as the solution to an Itô SDE:

$g(t) \in \mathbb{R}$

Diffusion **coefficient**

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

$\mathrm{d}\mathbf{w}$

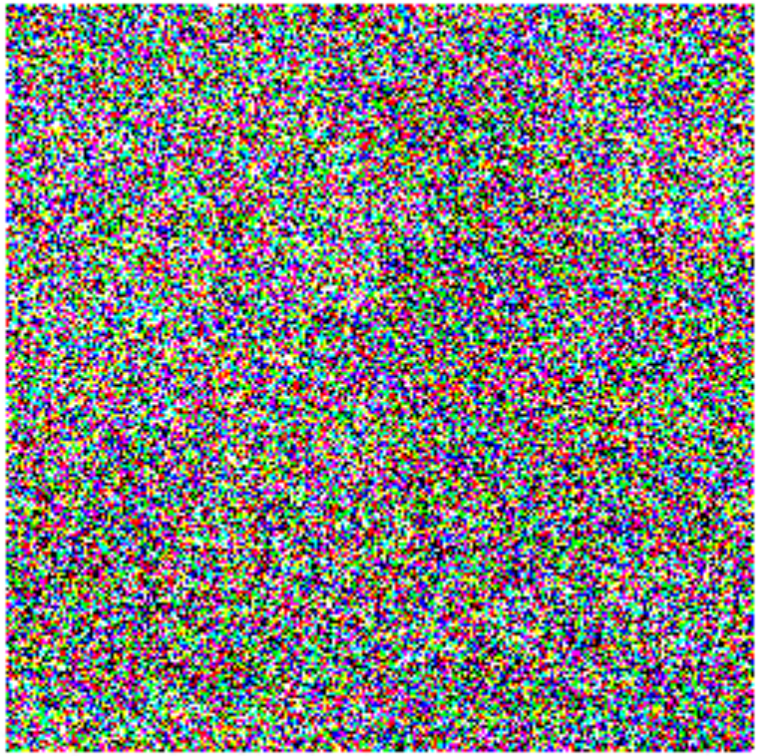
Infinitesimal white noise

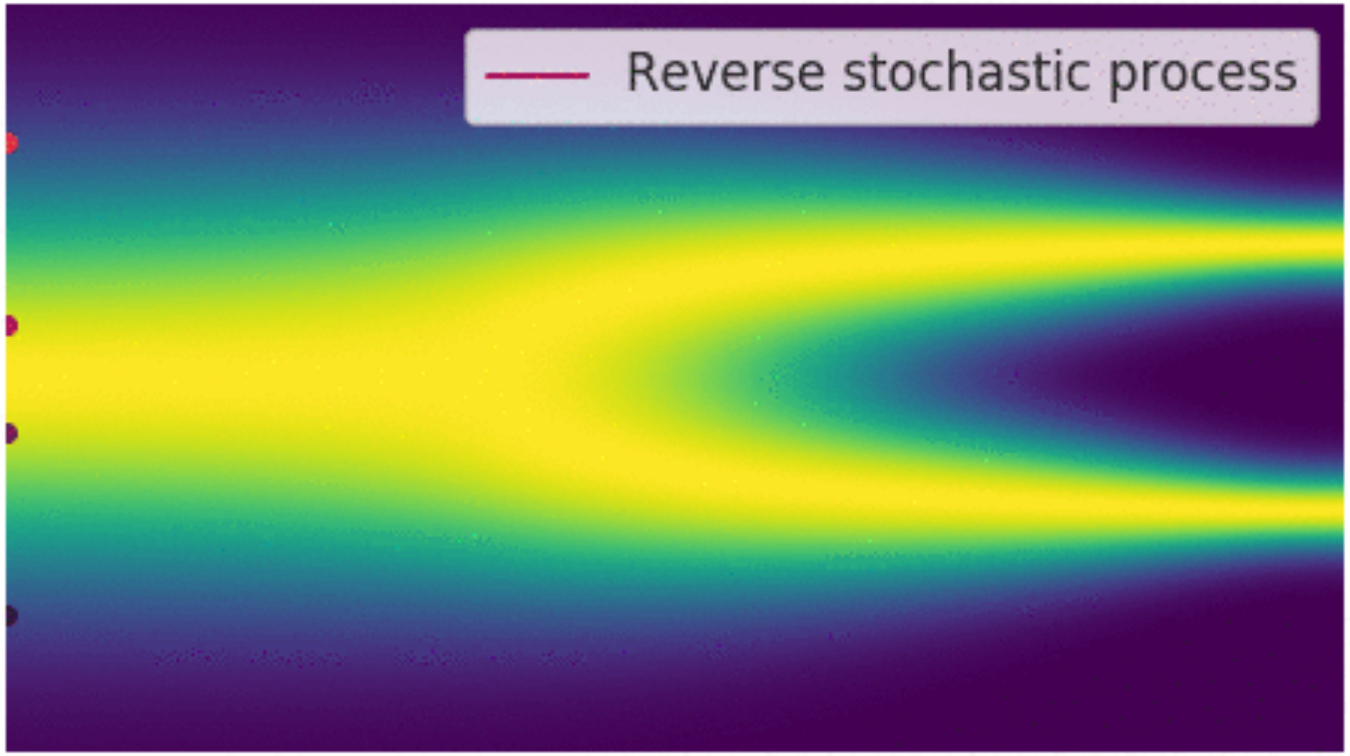
$\mathbf{f}(\cdot, t) : \mathbb{R}^d \to \mathbb{R}^d$

Drift **coefficient**

Standard Wiener process

Generating Samples By Reversing The SDE





Generate data from noise by reversing the perturbation procedure.

Source: https://yang-song.net/blog/2021/score/#introduction

Generating Samples By Reversing The SDE

- Reverse of diffusion process is also a diffusion process and is given by the reverse-time SDE:

We have to estimate the score function

Standard Wiener process when time flows backward

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

Infinitesimal negative timestep

Notation

$p_t(\mathbf{x})$: the probability density of $\mathbf{x}(t)$

$p_{st}(\mathbf{x}(t) \mid \mathbf{x}(s))$: transition kernel from $\mathbf{x}(s)$ to $\mathbf{x}(t)$

where $0 \leqslant s < t \leqslant T$

Estimating Scores For The SDE

- To estimate $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$, we can train a time train a time-dependent score-based model $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)$ using:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_t \Big\{ \lambda(t) \mathbb{E}_{\mathbf{x}(0)} \mathbb{E}_{\mathbf{x}(t)|\mathbf{x}(0)} \Big[\left\| \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}(t), t) - \nabla_{\mathbf{x}(t)} \log p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0)) \right\|_2^2 \Big] \Big\}$$

$\lambda : [0, T] \to \mathbb{R}_{>0}$; is a positive weighting function

$t \sim \mathcal{U}(0, 1)$; time is uniformly sampled

$\mathbf{x}(0) \sim p_0(\mathbf{x})$

$\mathbf{x}(t) \sim p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0))$

Given enough data and model capacity:

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

for almost all $\mathbf{x}$ and t

Examples: Variance Exploding (SE) SDE

$p_{\sigma_i}(\tilde{\mathbf{x}} \mid \mathbf{x}) := \mathcal{N}(\tilde{\mathbf{x}}; \mathbf{x}, \sigma_i^2 \mathbf{I})$; perturbation kernel of **SMLD**

$$\underbrace{\mathbf{x}_i = \mathbf{x}_{i-1} + \sqrt{\sigma_i^2 - \sigma_{i-1}^2}\mathbf{z}_{i-1}, \quad i = 1, \cdots, N,}$$

where $\mathbf{z}_{i-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

as $N \to \infty$

$\{\sigma_i\}_{i=1}^N \to \sigma(t)$

$\mathbf{z}_i \to \mathbf{z}(t)$

$\{\mathbf{x}_i\}_{i=1}^N \to \{\mathbf{x}(t)\}_{t=0}^1$ where $t \in [0, 1]$

$$\mathrm{d}\mathbf{x} = \sqrt{\frac{\mathrm{d}\left[\sigma^2(t)\right]}{\mathrm{d}t}}\mathrm{d}\mathbf{w}$$

Variance Exploding (VE) SDE corresponding to $\{\mathbf{x}(t)\}_{t=0}^1$

Examples: Variance Preserving (VP) SDE

$p(\mathbf{x}_i \mid \mathbf{x}_{i-1}) = \mathcal{N}(\mathbf{x}_i; \sqrt{1-\beta_i}\mathbf{x}_{i-1}, \beta_i \mathbf{I})$; perturbation kernel of **DDPM**

$$\underbrace{\mathbf{x}_i = \sqrt{1-\beta_i}\mathbf{x}_{i-1} + \sqrt{\beta_i}\mathbf{z}_{i-1}, \quad i = 1, \cdots, N}$$

as $N \to \infty$

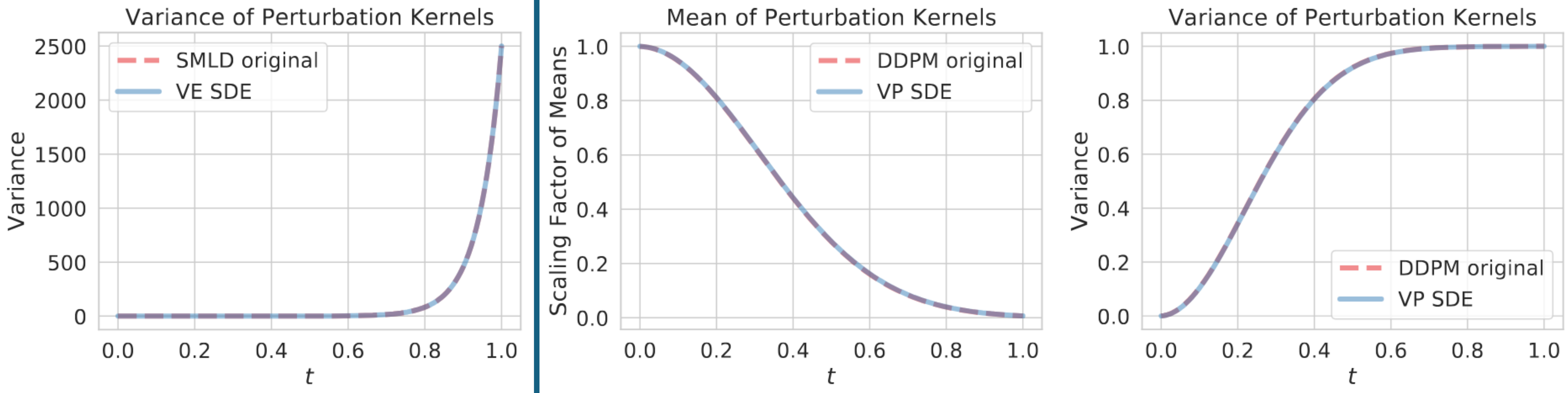
$$\mathrm{d}\mathbf{x} = -\frac{1}{2}\beta(t)\mathbf{x}\,\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\mathbf{w}$$

Variance Preserving (VP) SDE corresponding to $\{\mathbf{x}(t)\}_{t=0}^{1}$

Examples: VE and VP

$$p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0)) = \mathcal{N}\left(\mathbf{x}(t); \mathbf{x}(0), \sigma_{\min}^2 \left(\frac{\sigma_{\max}}{\sigma_{\min}}\right)^{2t} \mathbf{I}\right)$$

$$p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0))$$
$$= \mathcal{N}\left(\mathbf{x}(t); e^{-\frac{1}{4}t^2(\bar{\beta}_{\max}-\bar{\beta}_{\min})-\frac{1}{2}t\bar{\beta}_{\min}}\mathbf{x}(0), \mathbf{I} - \mathbf{I}e^{-\frac{1}{2}t^2(\bar{\beta}_{\max}-\bar{\beta}_{\min})-t\bar{\beta}_{\min}}\right)$$



(a) SMLD (b) DDPM (mean) (c) DDPM (variance)

Figure 5: Discrete-time perturbation kernels and our continuous generalizations match each other almost exactly. (a) compares the variance of perturbation kernels for SMLD and VE SDE; (b) compares the scaling factors of means of perturbation kernels for DDPM and VP SDE; and (c) compares the variance of perturbation kernels for DDPM and VP SDE.

Examples: sub-VP SDE

- New type of SDEs which perform particularly well on likelihoods given by:

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

- The variance of the stochastic process induced by the above SDE is always bounded by the VP SDE at every intermediate time step.

$$p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0)) = \begin{cases} \mathcal{N}\left(\mathbf{x}(t); \mathbf{x}(0), [\sigma^2(t) - \sigma^2(0)]\mathbf{I}\right), & \text{(VE SDE)} \\ \mathcal{N}\left(\mathbf{x}(t); \mathbf{x}(0)e^{-\frac{1}{2}\int_0^t \beta(s)\mathrm{d}s}, \mathbf{I} - \mathbf{I}e^{-\int_0^t \beta(s)\mathrm{d}s}\right) & \text{(VP SDE)} \\ \mathcal{N}\left(\mathbf{x}(t); \mathbf{x}(0)e^{-\frac{1}{2}\int_0^t \beta(s)\mathrm{d}s}, [1 - e^{-\int_0^t \beta(s)\mathrm{d}s}]^2\mathbf{I}\right) & \text{(sub-VP SDE)} \end{cases}$$

Solving the Reverse SDE

Solving the Reverse SDE

- We can use the trained $\mathbf{s}_{\boldsymbol{\theta}}$ to construct the reverse-time SDE.
- We can then simulate it with numerical approaches to generate samples from p_0.

General-purpose Numerical SDE Solvers

- Some general-purpose numerical SDE solvers include:
  - Euler-Maruyama
  - Stochastic Runge-Kutta
- These solvers correspond to different discretizations of the stochastic dynamics.
- ***Ancestral Sampling*** is also a generative SDE sampler!
- Authors propose ***reverse diffusion samplers***:
  - Discretize the reverse-time SDE in the same way as the forward one.

Predictor-Corrector Samplers

Numerical SDE Solver (e.g. Euler-Maruyama)

Predictor - Corrector

Score-based MCMC approach using
$\mathbf{s}_{\boldsymbol{\theta}*}(\mathbf{x}, t) \approx \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$

(e.g. Langevin MCMC)

Predictor-Corrector Samplers

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

Predictor

Corrector

Example: when using the **reverse diffusion SDE solver** (Appendix E) as the **predictor**, and **annealed Langevin dynamics** as the **corrector**

Predictor-Corrector Samplers (results)

Table 1: Comparing different reverse-time SDE solvers on CIFAR-10. Shaded regions are obtained with the same computation (number of score function evaluations). Mean and standard deviation are reported over five sampling runs. "P1000" or "P2000": predictor-only samplers using 1000 or 2000 steps. "C2000": corrector-only samplers using 2000 steps. "PC1000": Predictor-Corrector (PC) samplers using 1000 predictor and 1000 corrector steps.

	Variance Exploding SDE (SMLD)				Variance Preserving SDE (DDPM)			
FID↓ Sampler / Predictor	P1000	P2000	C2000	PC1000	P1000	P2000	C2000	PC1000
ancestral sampling	4.98 ± .06	4.88 ± .06		**3.62 ± .03**	3.24 ± .02	3.24 ± .02		**3.21 ± .02**
reverse diffusion	4.79 ± .07	4.74 ± .08	20.43 ± .07	**3.60 ± .02**	3.21 ± .02	3.19 ± .02	19.06 ± .06	**3.18 ± .01**
probability flow	15.41 ± .15	10.54 ± .08		**3.51 ± .04**	3.59 ± .04	3.23 ± .03		**3.06 ± .03**

Predictor-Corrector Samplers (results)

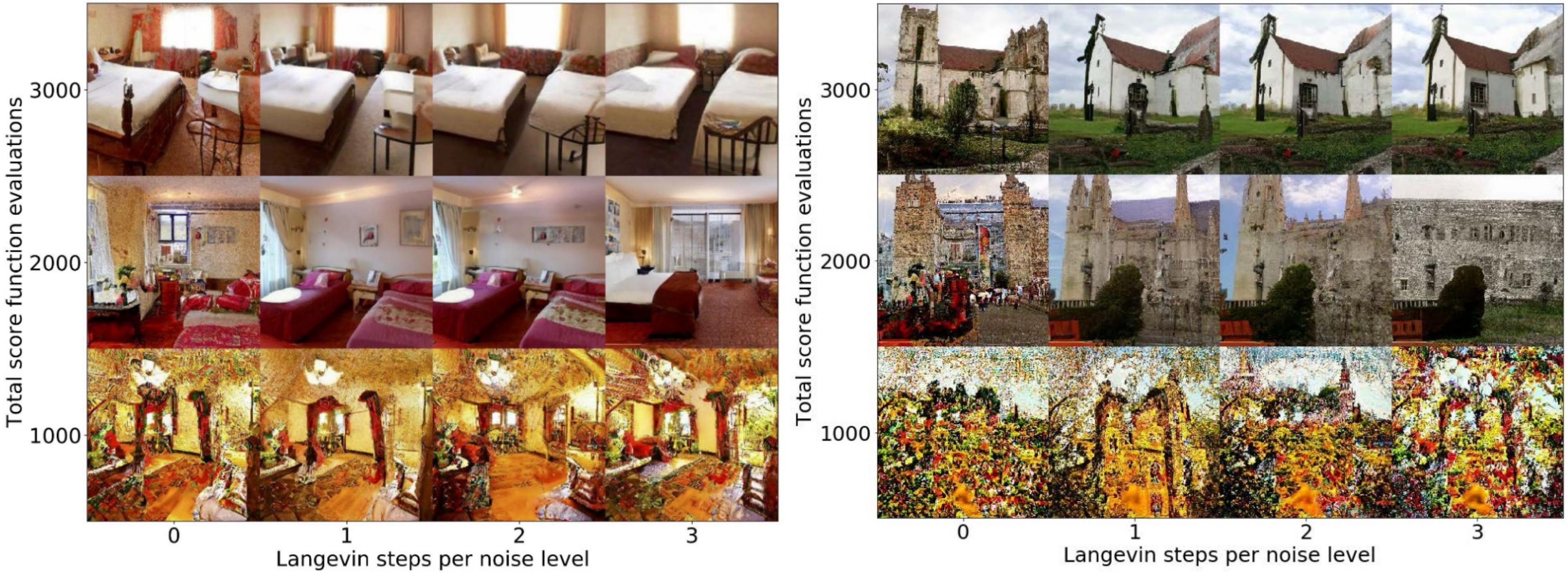


Figure 9: PC sampling for LSUN bedroom and church. The vertical axis corresponds to the total computation, and the horizontal axis represents the amount of computation allocated to the corrector. Samples are the best when computation is split between the predictor and corrector.

Probability Flow and Connection To Neural ODEs

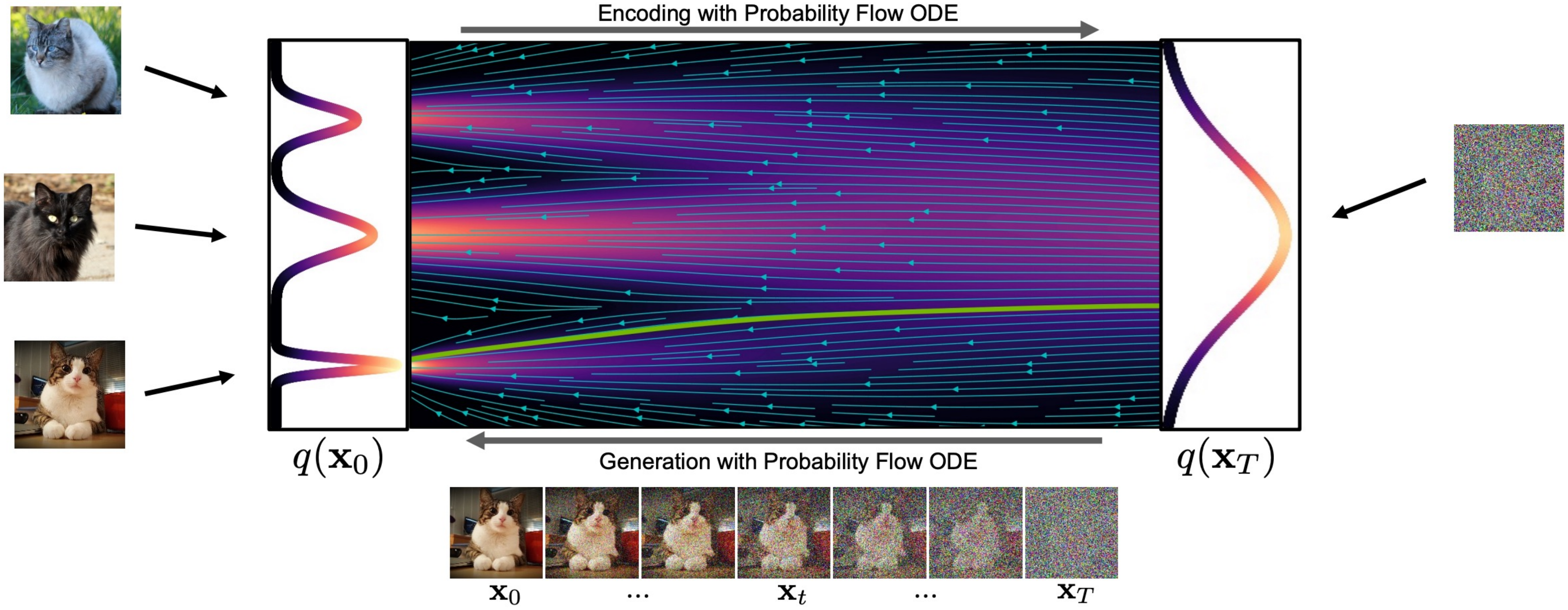
- For all diffusion processes, there exists a corresponding *deterministic process* whose trajectories share the same marginal probability densities $\{p_t(\mathbf{x})\}_{t=0}^{T}$.
- This deterministic process satisfies an ODE:

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

Probability flow ODE

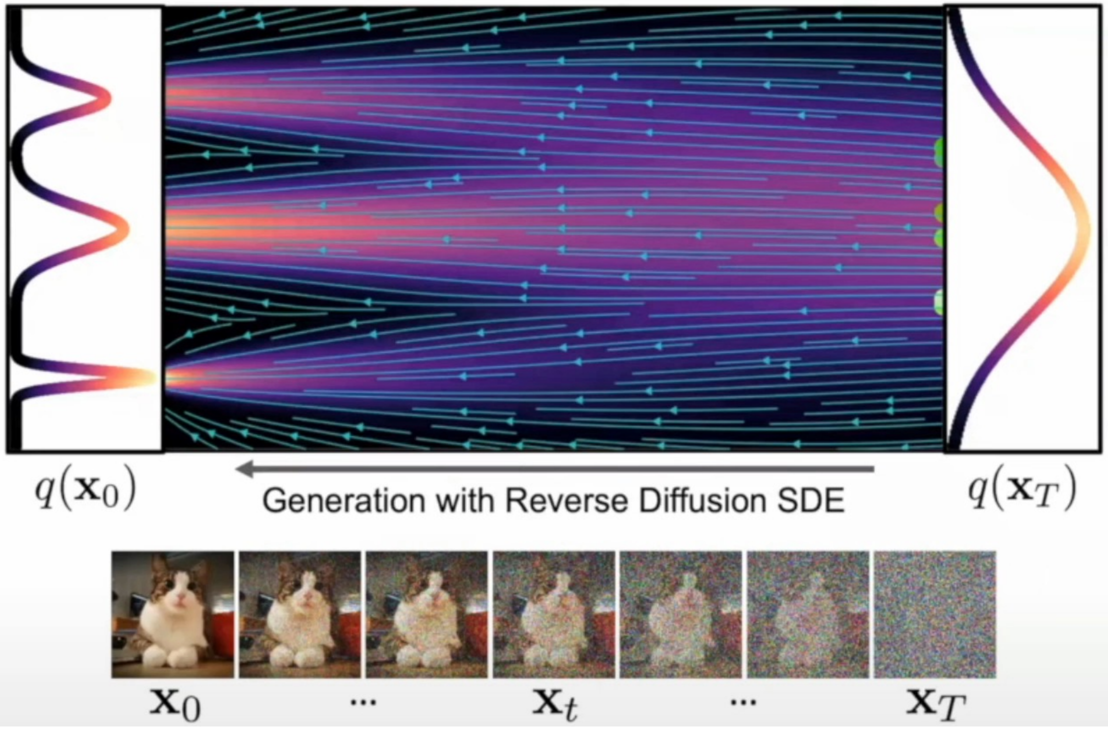
- When the score function is approximated by the time-dependent score-based model, which is typically a neural network, this is an example of a **neural ODE**.

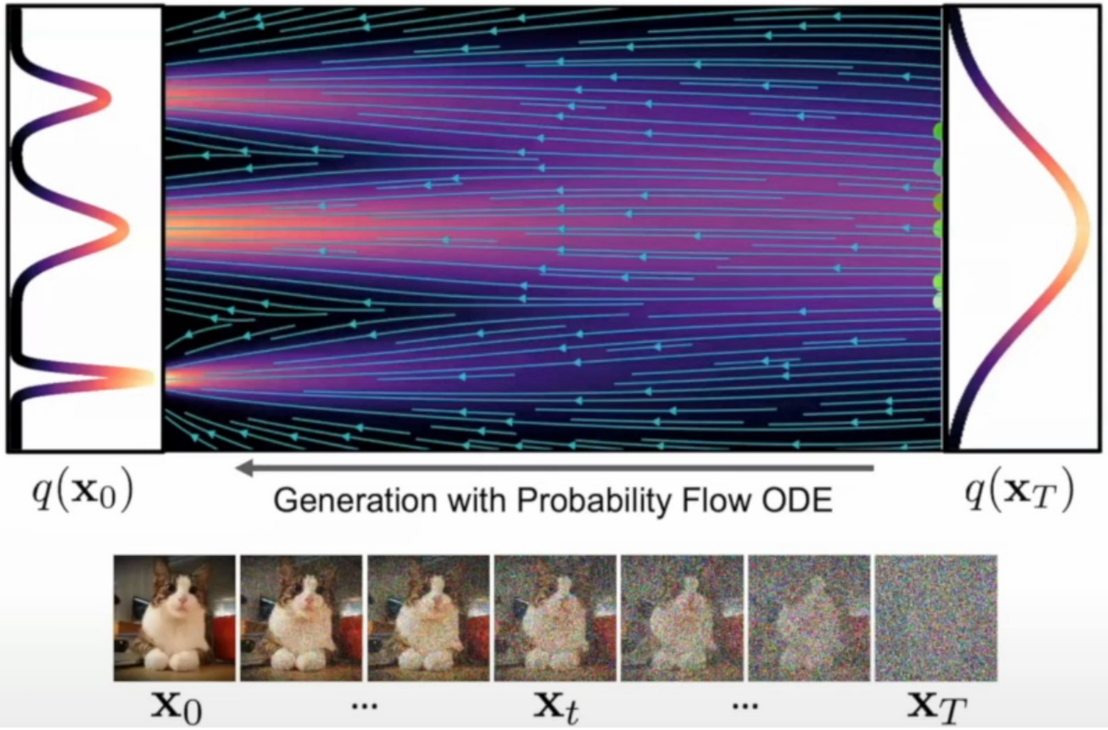
Probability Flow ODE



$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\left[\mathbf{x}_t + \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\right]\mathrm{d}t$$

Synthesis with SDE vs. ODE





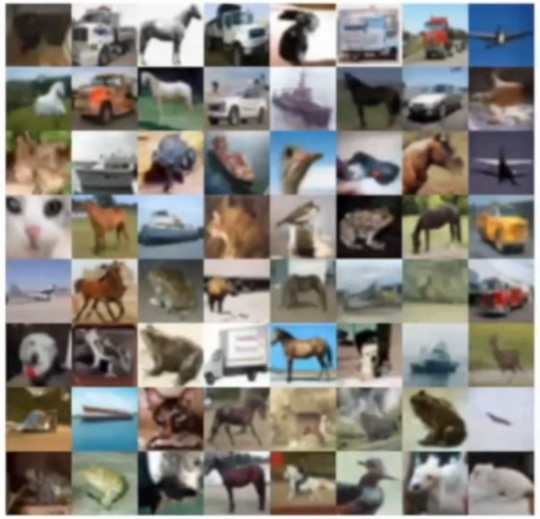
Why Should We Care About Neural ODEs?

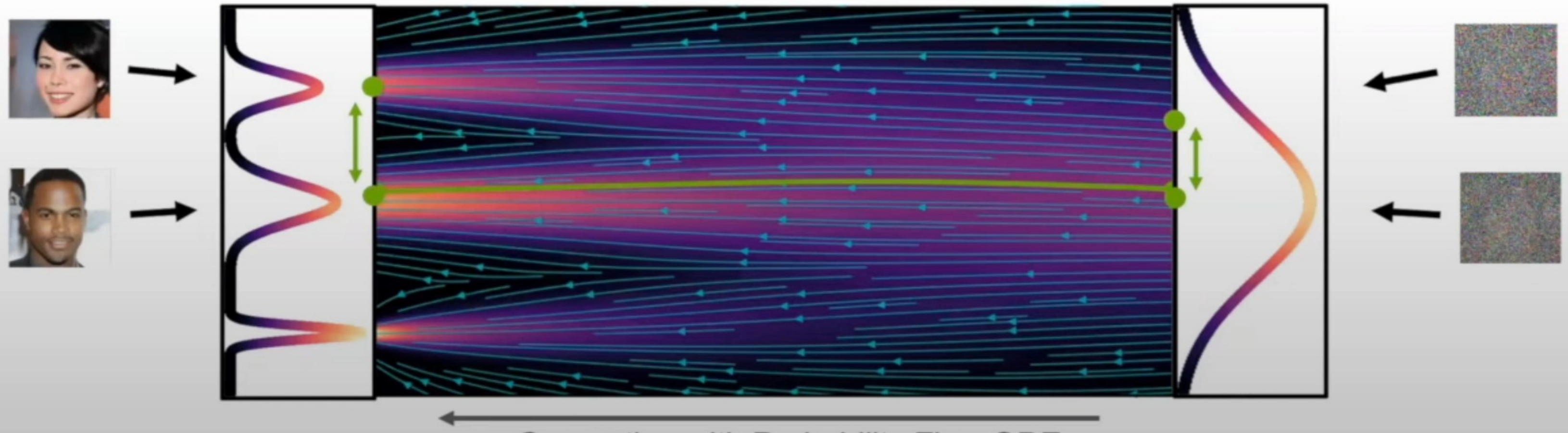
- Enables use of **advanced ODE solvers**.
- **Deterministic encoding and generation** (semantic image interpolation, etc.)
- **Log-likelihood computation** (instantaneous change of variables)

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}_0) = \log p_T(\mathbf{x}_T) - \int_0^T \mathrm{Tr}\left(\frac{1}{2}\beta(t)\frac{\partial}{\partial \mathbf{x}_t}\left[\mathbf{x}_t + \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\right]\right) \mathrm{d}t$$

- **Efficient sampling** by solving neural ODE from different final conditions

Semantic Image Interpolation with Probability Flow ODE





Continuous changes in latent space ($\mathbf{x}_T$) result in continuous, semantically meaningful changes in data space ($\mathbf{x}_0$)!

Semantic Image Interpolation with Probability Flow ODE

Samples from the probability flow ODE for VP SDE on 256 x 256 CelebA-HQ: spherical interpolations between random samples

Semantic Image Interpolation with Probability Flow ODE

Samples from the probability flow ODE for VP SDE on 256 x 256 CelebA-HQ: temperature rescaling (reducing norm of embedding)

Efficient sampling

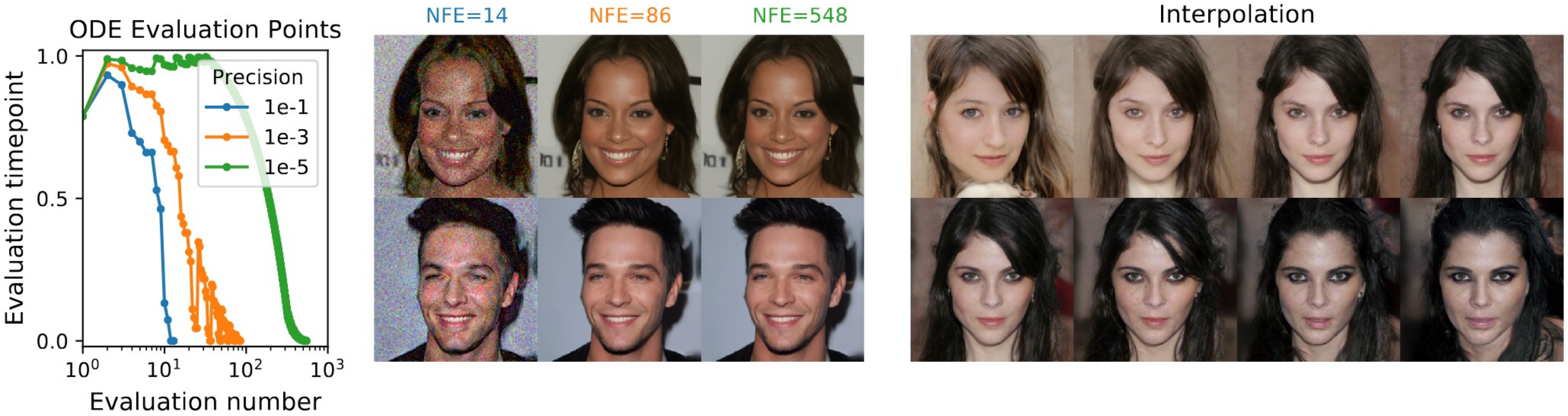


Figure 3: **Probability flow ODE enables fast sampling** with adaptive stepsizes as the numerical precision is varied (*left*), and reduces the number of score function evaluations (NFE) without harming quality (*middle*). The invertible mapping from latents to images allows for interpolations (*right*).

Architecture improvements

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 ± .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**

Architecture improvements

Samples on 1024x1024 CelebA-HQ from a modified NCSN++ model trained with the VE SDE.

Architecture improvements

Samples on 1024x1024 CelebA-HQ from a modified NCSN++ model trained with the VE SDE.

Architecture improvements

Samples on 1024x1024 CelebA-HQ from a modified NCSN++ model trained with the VE SDE.

Controllable Generation

Controllable Generation

- We can also produce data samples from $p_0(\mathbf{x}(0) \mid \mathbf{y})$ if $p_t(\mathbf{y} \mid \mathbf{x}(t))$
- Given a forward SDE $\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}$, we can sample from $p_t(\mathbf{x}(t) \mid \mathbf{y})$ by starting from $p_T(\mathbf{x}(T) \mid \mathbf{y})$ and solving a conditional reverse time SDE:

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

Controllable Generation

Figure 4: *Left*: Class-conditional samples on 32 × 32 CIFAR-10. Top four rows are automobiles and bottom four rows are horses. *Right:* Inpainting (top two rows) and colorization (bottom two rows) results on 256 × 256 LSUN. First column is the original image, second column is the masked/gray-scale image, remaining columns are sampled image completions or colorizations.

Thank you

References

- **CVPR 2022 Tutorial:** Denoising Diffusion-based Generative Modeling: Foundations and Applications
  - Link: https://cvpr2022-tutorial-diffusion-models.github.io/
- **Blog**: Generative Modeling by Estimating Gradients of the Data Distribution
  - Link to blog: https://yang-song.net/blog/2021/score/#introduction