Diffusion Model and Score Matching

Instructor: Lei Wu [1]

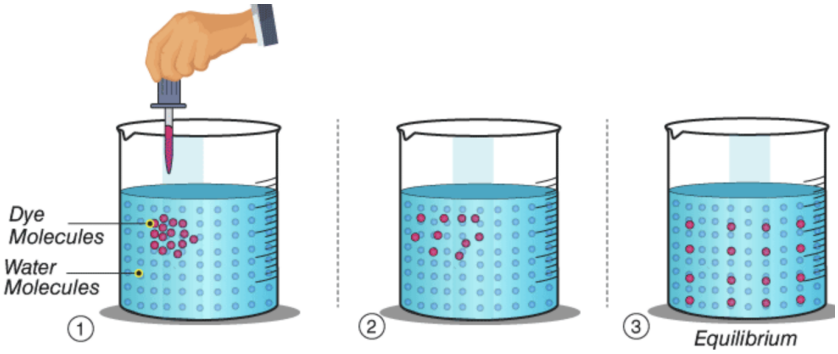
Mathematical Introduction to Machine Learning

Peking University, Fall 2025

[1]School of Mathematical Sciences; Center for Machine Learning Research

Outline

What Is Diffusion?



Dye molecules **diffuse** throughout the entire space by colliding with water molecules. A visualization illustrating this microscale mechanism is available at this link.

Mathematical Model of Diffusion: Brownian motion

- Let $\{x_k\}_{k\geq 0}$ be the trajectory of dye molecules. We can model its dynamics as follows

$$x_{k+1} = x_k + \sqrt{\eta}\xi_k, \qquad 0 \leq k \leq N-1,$$

where $\xi_k \overset{iid}{\sim} \mathcal{N}(0,1)$ and η is a small factor [2].

[2] η depends on the temperature, time unit, etc.

Mathematical Model of Diffusion: Brownian motion

- Let $\{x_k\}_{k\geq 0}$ be the trajectory of dye molecules. We can model its dynamics as follows

$$x_{k+1} = x_k + \sqrt{\eta}\xi_k, \qquad 0 \leq k \leq N-1,$$

  where $\xi_k \overset{iid}{\sim} \mathcal{N}(0,1)$ and η is a small factor [2].
- Thus, we have after N steps

$$x_N = x_0 + \sqrt{\eta}\sum_{k=0}^{N-1} \xi_k \sim \mathcal{N}(x_0, \eta N).$$

[2] η depends on the temperature, time unit, etc.

Mathematical Model of Diffusion: Brownian motion

- Let $\{x_k\}_{k\geq 0}$ be the trajectory of dye molecules. We can model its dynamics as follows

$$x_{k+1} = x_k + \sqrt{\eta}\xi_k, \qquad 0 \leq k \leq N-1,$$

  where $\xi_k \overset{iid}{\sim} \mathcal{N}(0,1)$ and η is a small factor [2].
- Thus, we have after N steps

$$x_N = x_0 + \sqrt{\eta}\sum_{k=0}^{N-1} \xi_k \sim \mathcal{N}(x_0, \eta N).$$

- Consider the continuous-time limit: $\eta \to 0$. Let $t = N\eta$. Then, we have

$$x_N \to X_t \sim \mathcal{N}(X_0, t).$$

[2] η depends on the temperature, time unit, etc.

Mathematical Model of Diffusion: Brownian motion

- Let $\{x_k\}_{k\geq 0}$ be the trajectory of dye molecules. We can model its dynamics as follows

$$x_{k+1} = x_k + \sqrt{\eta}\xi_k, \qquad 0 \leq k \leq N-1,$$

  where $\xi_k \overset{iid}{\sim} \mathcal{N}(0,1)$ and η is a small factor [2].
- Thus, we have after N steps

$$x_N = x_0 + \sqrt{\eta}\sum_{k=0}^{N-1} \xi_k \sim \mathcal{N}(x_0, \eta N).$$

- Consider the continuous-time limit: $\eta \to 0$. Let $t = N\eta$. Then, we have

$$x_N \to X_t \sim \mathcal{N}(X_0, t).$$

- We call $B_t := X_t - X_0$ Brownian motion.

[2] η depends on the temperature, time unit, etc.

Important Properties of Brownian Motion

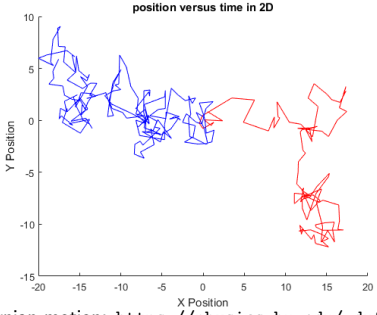


Figure 1: A animation of Brownian motion: `https://physics.bu.edu/~duffy/HTML5/brownian_motion.html`

[3]Albert Einstein, On the motion of small particles suspended in liquids at rest required by the molecular-kinetic theory of heat, 1905.

Important Properties of Brownian Motion

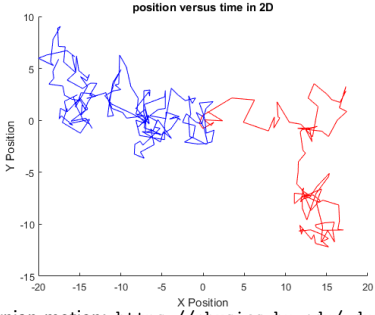


Figure 1: A animation of Brownian motion: `https://physics.bu.edu/~duffy/HTML5/brownian_motion.html`

- After time t, dye molecules only move $O(\sqrt{t})$:

$$\mathbb{E}[B_t] = 0, \quad \mathbb{E}[B_t^2] = t,$$

where the second property was first derived by Albert Einstein [3].

[3] Albert Einstein, On the motion of small particles suspended in liquids at rest required by the molecular-kinetic theory of heat, 1905.

Important Properties of Brownian Motion

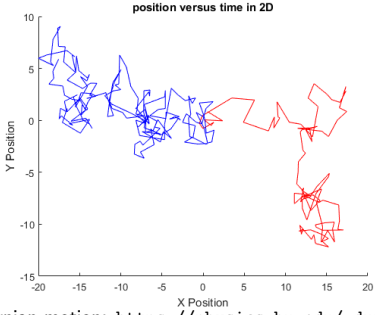


Figure 1: A animation of Brownian motion: `https://physics.bu.edu/~duffy/HTML5/brownian_motion.html`

- After time t, dye molecules only move $O(\sqrt{t})$:

$$\mathbb{E}[B_t] = 0, \quad \mathbb{E}[B_t^2] = t,$$

  where the second property was first derived by Albert Einstein [3].

- $B_t - B_s$ and B_s are independent. The trajectory is continuous but **non-differentiable almost everywhere**.

[3] Albert Einstein, On the motion of small particles suspended in liquids at rest required by the molecular-kinetic theory of heat, 1905.

General Diffusion Process (Modeled by Itô SDE)

Consider dye molecules moving under a force field $f(x,t)$ while undergoing heterogeneous collisions:

- Over a small interval $[t, t+\eta]$, the position update can be modeled as

$$x_{t+\eta} - x_t = \underbrace{f(x_t,t)\,\eta}_{\text{drift}} + \underbrace{\sigma(x_t,t)\,\sqrt{\eta}\,\xi_t}_{\text{diffusion}},$$

  where $\xi_t \sim \mathcal{N}(0, I)$ represents random collisions.

[4] A standard reference: Bernt Øksendal, *Stochastic Differential Equations: An Introduction with Applications*.

General Diffusion Process (Modeled by Itô SDE)

Consider dye molecules moving under a force field $f(x,t)$ while undergoing heterogeneous collisions:

- Over a small interval $[t, t+\eta]$, the position update can be modeled as

$$x_{t+\eta} - x_t = \underbrace{f(x_t,t)\,\eta}_{\text{drift}} + \underbrace{\sigma(x_t,t)\,\sqrt{\eta}\,\xi_t}_{\text{diffusion}},$$

  where $\xi_t \sim \mathcal{N}(0, I)$ represents random collisions.

- Taking $\eta \to 0$ leads to the corresponding **stochastic differential equation (SDE)**[4]:

$$\mathrm{d}x_t = f(x_t,t)\ \mathrm{d}t + \sigma(x_t,t)\ \mathrm{d}B_t.$$

[4] A standard reference: Bernt Øksendal, *Stochastic Differential Equations: An Introduction with Applications*.

General Diffusion Process (Modeled by Itô SDE)

Consider dye molecules moving under a force field $f(x,t)$ while undergoing heterogeneous collisions:

- Over a small interval $[t, t+\eta]$, the position update can be modeled as

$$x_{t+\eta} - x_t = \underbrace{f(x_t,t)\,\eta}_{\text{drift}} + \underbrace{\sigma(x_t,t)\,\sqrt{\eta}\,\xi_t}_{\text{diffusion}},$$

  where $\xi_t \sim \mathcal{N}(0, I)$ represents random collisions.

- Taking $\eta \to 0$ leads to the corresponding **stochastic differential equation (SDE)**[4]:

$$\mathrm{d}x_t = f(x_t,t)\ \mathrm{d}t + \sigma(x_t,t)\ \mathrm{d}B_t.$$

- In physics, this SDE is often written formally by setting $\omega_t = \dot{B}_t$:

$$\dot{x}_t = f(x_t,t) + \sigma(x_t,t)\,\omega_t,$$

  where ω_t can be interpreted as *white noise*.

[4] A standard reference: Bernt Øksendal, *Stochastic Differential Equations: An Introduction with Applications*.

Fokker–Planck Equation

Consider the Itô SDE:

$$\mathrm{d}x_t = f(x_t, t)\ \mathrm{d}t + \sigma(x_t, t)\ \mathrm{d}B_t,$$

and let $p_t(x)$ denote the density of x_t. Then the evolution of p_t is governed by the **Fokker–Planck equation**:

$$\boxed{\frac{\partial p_t}{\partial t} = -\nabla \cdot \big(f(x,t)\, p_t(x)\big) + \nabla \cdot \Big(D(x,t)\, \nabla p_t(x)\Big)}$$

where

$$D(x,t) = \frac{1}{2}\sigma(x,t)\sigma(x,t)^\top.$$

- The term $-\nabla \cdot (fp)$ describes **drift**: deterministic transport of probability mass.

Fokker–Planck Equation

Consider the Itô SDE:

$$\mathrm{d}x_t = f(x_t, t)\ \mathrm{d}t + \sigma(x_t, t)\ \mathrm{d}B_t,$$

and let $p_t(x)$ denote the density of x_t. Then the evolution of p_t is governed by the **Fokker–Planck equation**:

$$\boxed{\frac{\partial p_t}{\partial t} = -\nabla \cdot \big(f(x,t)\, p_t(x)\big) + \nabla \cdot \Big(D(x,t)\, \nabla p_t(x)\Big)}$$

where

$$D(x,t) = \frac{1}{2}\sigma(x,t)\sigma(x,t)^\top.$$

- The term $-\nabla \cdot (fp)$ describes **drift**: deterministic transport of probability mass.
- The term $\nabla \cdot (D \nabla p)$ describes **diffusion**: random spreading of the density.

Fokker–Planck Equation

Consider the Itô SDE:

$$\mathrm{d}x_t = f(x_t, t)\ \mathrm{d}t + \sigma(x_t, t)\ \mathrm{d}B_t,$$

and let $p_t(x)$ denote the density of x_t. Then the evolution of p_t is governed by the **Fokker–Planck equation**:

$$\boxed{\frac{\partial p_t}{\partial t} = -\nabla \cdot \big(f(x,t)\, p_t(x)\big) + \nabla \cdot \Big(D(x,t)\, \nabla p_t(x)\Big)}$$

where

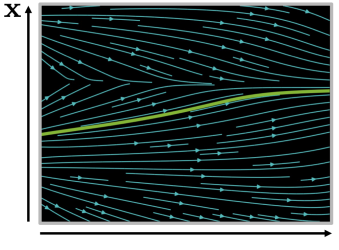
$$D(x,t) = \frac{1}{2}\sigma(x,t)\sigma(x,t)^\top.$$

- The term $-\nabla \cdot (fp)$ describes **drift**: deterministic transport of probability mass.
- The term $\nabla \cdot (D \nabla p)$ describes **diffusion**: random spreading of the density.
- This PDE is fundamental in diffusion models: reversing the Fokker–Planck flow gives the **reverse-time SDE**.

A Comparison Between SDE and ODE [5]

Ordinary Differential Equation (ODE):

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



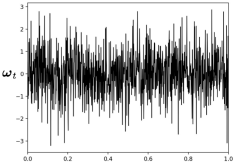
Analytical Solution: $\mathbf{x}(t) = \mathbf{x}(0) + \int_0^t \mathbf{f}(\mathbf{x}, \tau)\mathrm{d}\tau$

Iterative Numerical Solution: $\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \mathbf{f}(\mathbf{x}(t), t)\Delta t$

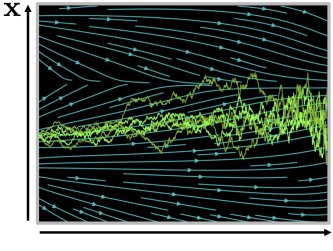
Stochastic Differential Equation (SDE):

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

Wiener Process (Gaussian White Noise)



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



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

[43]

[5] taken from https://cvpr2022-tutorial-diffusion-models.github.io/

Langevin Dynamics: Definition and Stationary Distribution

- (Over-damped) **Langevin dynamics** describes a stochastic motion driven by a potential $U(x)$:

$$\mathrm{d}x_t = -\nabla U(x_t)\,\mathrm{d}t + \sqrt{2\beta^{-1}}\,\mathrm{d}B_t.$$

  It can be viewed as gradient flow perturbed by Gaussian noise.

Langevin Dynamics: Definition and Stationary Distribution

- (Over-damped) **Langevin dynamics** describes a stochastic motion driven by a potential $U(x)$:

$$\mathrm{d}x_t = -\nabla U(x_t)\,\mathrm{d}t + \sqrt{2\beta^{-1}}\,\mathrm{d}B_t.$$

  It can be viewed as gradient flow perturbed by Gaussian noise.

- Let $p_t = \mathrm{Law}(x_t)$ denote the distribution of x_t. Under mild regularity and dissipativity conditions, Langevin dynamics converges to the Gibbs distribution:

$$p_t(x) \;\to\; Z_\beta^{-1} e^{-\beta U(x)} \qquad \text{as } t \to \infty.$$

Langevin Dynamics: Definition and Stationary Distribution

- (Over-damped) **Langevin dynamics** describes a stochastic motion driven by a potential $U(x)$:

$$\mathrm{d}x_t = -\nabla U(x_t)\,\mathrm{d}t + \sqrt{2\beta^{-1}}\,\mathrm{d}B_t.$$

  It can be viewed as gradient flow perturbed by Gaussian noise.

- Let $p_t = \mathrm{Law}(x_t)$ denote the distribution of x_t. Under mild regularity and dissipativity conditions, Langevin dynamics converges to the Gibbs distribution:

$$p_t(x) \;\to\; Z_\beta^{-1} e^{-\beta U(x)} \qquad \text{as } t \to \infty.$$

- **Interpretation.** Langevin dynamics provides a way to sample from the Gibbs measure by injecting noise that balances the potential force. This principle underlies many generative models, including diffusion models.

Euler–Maruyama Approximation and OU Process

- To simulate Langevin dynamics, we can use the **Euler–Maruyama scheme**:

$$X_{k+1} = X_k - \eta \nabla U(X_k) + \sqrt{2\beta^{-1}\eta}\,\xi_k, \qquad \xi_k \sim \mathcal{N}(0, I_d).$$

  This can be interpreted as noisy gradient descent.

Euler–Maruyama Approximation and OU Process

- To simulate Langevin dynamics, we can use the **Euler–Maruyama scheme**:

$$X_{k+1} = X_k - \eta \nabla U(X_k) + \sqrt{2\beta^{-1}\eta}\,\xi_k, \qquad \xi_k \sim \mathcal{N}(0, I_d).$$

  This can be interpreted as noisy gradient descent.

- The **Ornstein–Uhlenbeck (OU) process** is the simplest Langevin system:

$$\mathrm{d}X_t = -\theta X_t \,\mathrm{d}t + \sigma \,\mathrm{d}B_t.$$

  It corresponds to a quadratic potential:

$$U(x) = \tfrac{\theta}{2}\|x\|^2, \qquad \beta^{-1} = \sigma^2/2.$$

Euler–Maruyama Approximation and OU Process

- To simulate Langevin dynamics, we can use the **Euler–Maruyama scheme**:

$$X_{k+1} = X_k - \eta \nabla U(X_k) + \sqrt{2\beta^{-1}\eta}\,\xi_k, \qquad \xi_k \sim \mathcal{N}(0, I_d).$$

  This can be interpreted as noisy gradient descent.

- The **Ornstein–Uhlenbeck (OU) process** is the simplest Langevin system:

$$\mathrm{d}X_t = -\theta X_t \,\mathrm{d}t + \sigma \,\mathrm{d}B_t.$$

  It corresponds to a quadratic potential:

$$U(x) = \tfrac{\theta}{2}\|x\|^2, \qquad \beta^{-1} = \sigma^2/2.$$

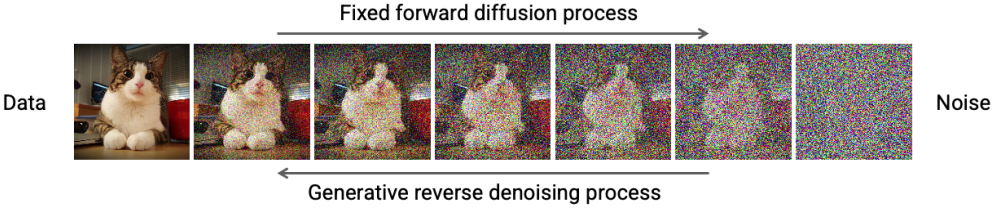
- The stationary distribution of OU process is Gaussian:

$$p_\infty(x) \propto \exp\left(-\frac{\|x\|^2}{2\sigma^2}\right).$$

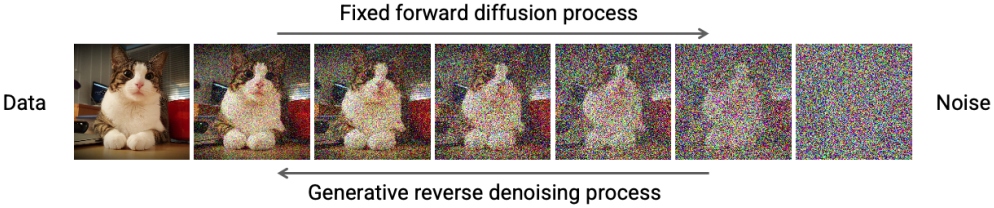
Diffusion Models

In diffusion models,

- We gradually inject noise into a data sample until it becomes nearly pure Gaussian noise. This defines the **forward diffusion process**.
- A diffusion model learns a (probabilistic) **reverse process** that inverts this noising procedure, transforming noise back into a realistic data sample.



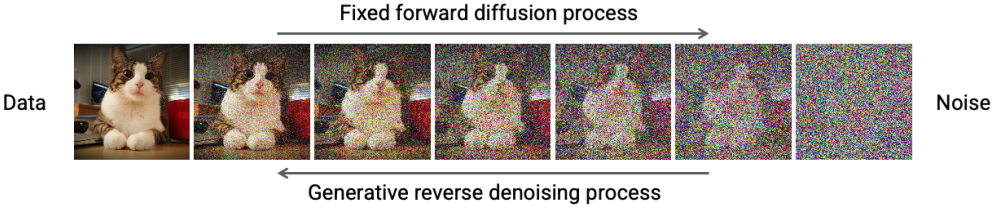
Diffusion Models



Why Diffusion Models Learn Better?

- Diffusion models are guided by a **known forward noising process**, which provides informative **intermediate states**.
- Learning becomes a sequence of small denoising steps, instead of a one-shot generation problem.
- GANs, flows, and VAEs have **no forward process**, so they must learn the whole mapping directly.

Diffusion Models



There are two central questions in diffusion models:

- How to design an appropriate forward diffusion process.
- How to leverage this forward process to efficiently learn the reverse (denoising) dynamics.

Denoising Diffusion Probabilistic Models (DDPM) [6]

- DDPM chooses the following **variance-preserving** forward diffusion process:

$$x_{k+1} = \sqrt{1-\beta_k}\, x_k + \sqrt{\beta_k}\, \xi_k, \quad 0 \le k \le N-1,$$

where $\xi_k \overset{iid}{\sim} \mathcal{N}(0, I_d)$.

[6] Jonathan Ho, Ajay Jain, Pieter Abbeel, *Denoising Diffusion Probabilistic Models*, NeurIPS 2020.

Denoising Diffusion Probabilistic Models (DDPM) [6]

- DDPM chooses the following **variance-preserving** forward diffusion process:

$$x_{k+1} = \sqrt{1-\beta_k}\, x_k + \sqrt{\beta_k}\, \xi_k, \quad 0 \le k \le N-1,$$

  where $\xi_k \overset{iid}{\sim} \mathcal{N}(0, I_d)$.
- Consider $\beta_t = \beta = o(1)$. Then, we have

$$x_{k+1} = x_k - \frac{\beta x_k}{2} + \sqrt{\beta}\xi_k + o(\beta) \tag{1}$$

[6] Jonathan Ho, Ajay Jain, Pieter Abbeel, *Denoising Diffusion Probabilistic Models*, NeurIPS 2020.

Denoising Diffusion Probabilistic Models (DDPM) [6]

- DDPM chooses the following **variance-preserving** forward diffusion process:

$$x_{k+1} = \sqrt{1-\beta_k}\, x_k + \sqrt{\beta_k}\, \xi_k, \quad 0 \le k \le N-1,$$

  where $\xi_k \overset{iid}{\sim} \mathcal{N}(0, I_d)$.
- Consider $\beta_t = \beta = o(1)$. Then, we have

$$x_{k+1} = x_k - \frac{\beta x_k}{2} + \sqrt{\beta}\xi_k + o(\beta) \tag{1}$$

- Letting $\beta \to 0$ and rescaling time by $t = k\beta$ gives the limiting SDE:

$$\mathrm{d}x_t = -\frac{x_t}{2}\,\mathrm{d}t + \mathrm{d}B_t,$$

  which is a **Ornstein–Uhlenbeck** (OU) process.

[6] Jonathan Ho, Ajay Jain, Pieter Abbeel, *Denoising Diffusion Probabilistic Models*, NeurIPS 2020.

Forward OU Process: Conditional Distribution

- Recall the continuous-time forward process (OU SDE):

$$\mathrm{d}x_t = -\frac{1}{2}x_t \, \mathrm{d}t + \mathrm{d}B_t, \qquad x_0 \sim P_0.$$

Forward OU Process: Conditional Distribution

- Recall the continuous-time forward process (OU SDE):

$$\mathrm{d}x_t = -\frac{1}{2}x_t \, \mathrm{d}t + \mathrm{d}B_t, \qquad x_0 \sim P_0.$$

- This linear SDE admits an explicit solution:

$$x_t = e^{-t/2}x_0 + \int_0^t e^{-(t-s)/2} \, \mathrm{d}B_s.$$

Forward OU Process: Conditional Distribution

- Recall the continuous-time forward process (OU SDE):

$$\mathrm{d}x_t = -\frac{1}{2}x_t \, \mathrm{d}t + \mathrm{d}B_t, \qquad x_0 \sim P_0.$$

- This linear SDE admits an explicit solution:

$$x_t = e^{-t/2}x_0 + \int_0^t e^{-(t-s)/2} \, \mathrm{d}B_s.$$

- Conditioned on x_0, the stochastic integral is Gaussian with

$$\mathbb{E}\left[\int_0^t e^{-(t-s)/2} \, \mathrm{d}B_s \,\middle|\, x_0\right] = 0,$$

$$\mathrm{Cov}\left[\int_0^t e^{-(t-s)/2} \, \mathrm{d}B_s \,\middle|\, x_0\right] = \int_0^t e^{-(t-s)} \, \mathrm{d}s \; I_d = (1 - e^{-t})I_d.$$

Forward OU Process: Conditional Distribution

- Recall the continuous-time forward process (OU SDE):

$$\mathrm{d}x_t = -\frac{1}{2}x_t \, \mathrm{d}t + \mathrm{d}B_t, \qquad x_0 \sim P_0.$$

- This linear SDE admits an explicit solution:

$$x_t = e^{-t/2}x_0 + \int_0^t e^{-(t-s)/2} \, \mathrm{d}B_s.$$

- Conditioned on x_0, the stochastic integral is Gaussian with

$$\mathbb{E}\left[\int_0^t e^{-(t-s)/2} \, \mathrm{d}B_s \,\middle|\, x_0\right] = 0,$$

$$\mathrm{Cov}\left[\int_0^t e^{-(t-s)/2} \, \mathrm{d}B_s \,\middle|\, x_0\right] = \int_0^t e^{-(t-s)} \, \mathrm{d}s \; I_d = (1 - e^{-t})I_d.$$

- Define

$$\alpha_t := e^{-t/2}, \qquad \sigma_t^2 := 1 - \alpha_t^2 = 1 - e^{-t}.$$

Then

$$\boxed{x_t \mid x_0 \sim \mathcal{N}\big(\alpha_t x_0, \; \sigma_t^2 I_d\big)\,.}$$

Gaussian Smoothing and Convergence

- Recall the conditional law

$$x_t \mid x_0 \sim \mathcal{N}\big(\alpha_t x_0,\ \sigma_t^2 I_d\big)\,, \qquad \alpha_t = e^{-t/2},\ \sigma_t^2 = 1 - \alpha_t^2.$$

Gaussian Smoothing and Convergence

- Recall the conditional law

$$x_t \mid x_0 \sim \mathcal{N}\big(\alpha_t x_0,\ \sigma_t^2 I_d\big)\,, \qquad \alpha_t = e^{-t/2},\ \sigma_t^2 = 1 - \alpha_t^2.$$

- Let P_0 be the law of x_0 and P_t the law of x_t. By the law of total probability,

$$P_t(x) = \int p(x \mid x_0)\, P_0(\mathrm{d}x_0),$$

where $p(x \mid x_0)$ is the density of $\mathcal{N}(\alpha_t x_0, \sigma_t^2 I_d)$.

Gaussian Smoothing and Convergence

- Recall the conditional law

$$x_t \mid x_0 \sim \mathcal{N}\big(\alpha_t x_0,\ \sigma_t^2 I_d\big)\,, \qquad \alpha_t = e^{-t/2},\ \sigma_t^2 = 1 - \alpha_t^2.$$

- Let P_0 be the law of x_0 and P_t the law of x_t. By the law of total probability,

$$P_t(x) = \int p(x \mid x_0)\, P_0(\mathrm{d}x_0),$$

  where $p(x \mid x_0)$ is the density of $\mathcal{N}(\alpha_t x_0, \sigma_t^2 I_d)$.
- Equivalently, we can write

$$x_t \overset{d}{=} \alpha_t x_0 + \sigma_t \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I_d),$$

  so P_t is obtained from P_0 by **shrinking the signal** (α_t) and **adding Gaussian noise** $(\sigma_t \epsilon)$. We may view this as a Gaussian smoothing of P_0.

Gaussian Smoothing and Convergence

- Recall the conditional law

$$x_t \mid x_0 \sim \mathcal{N}\big(\alpha_t x_0,\ \sigma_t^2 I_d\big)\,, \qquad \alpha_t = e^{-t/2},\ \sigma_t^2 = 1 - \alpha_t^2.$$

- Let P_0 be the law of x_0 and P_t the law of x_t. By the law of total probability,

$$P_t(x) = \int p(x \mid x_0)\, P_0(\mathrm{d}x_0),$$

  where $p(x \mid x_0)$ is the density of $\mathcal{N}(\alpha_t x_0, \sigma_t^2 I_d)$.
- Equivalently, we can write

$$x_t \overset{d}{=} \alpha_t x_0 + \sigma_t \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I_d),$$

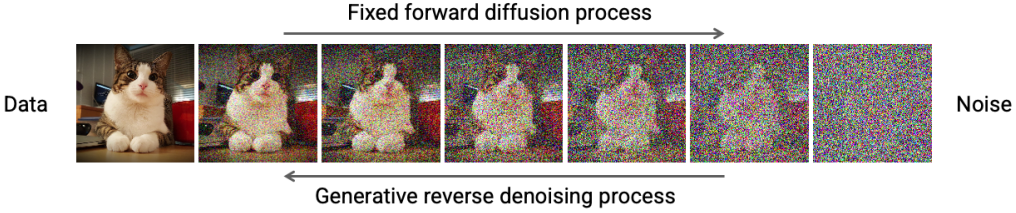
  so P_t is obtained from P_0 by **shrinking the signal** (α_t) and **adding Gaussian noise** $(\sigma_t \epsilon)$. We may view this as a Gaussian smoothing of P_0.
- Moreover, the OU process converges to Gaussian exponentially fast:

$$D_{\mathrm{KL}}(P_t \,\|\, \mathcal{N}(0, I_d)) \le C e^{-t} D_{\mathrm{KL}}(P_0 \,\|\, \mathcal{N}(0, I_d))\,,$$

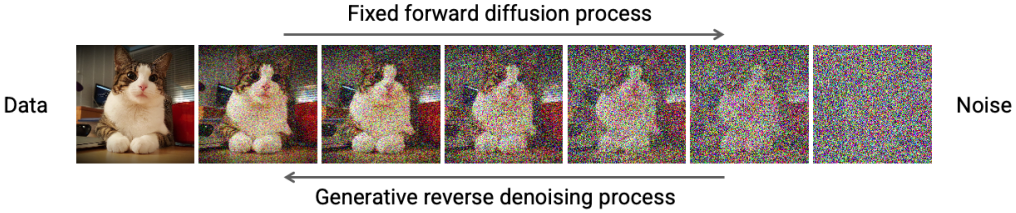
  so for sufficiently large T, $\mathrm{Law}(x_T) \approx \mathcal{N}(0, I_d)$.

Reversing a Diffusion Process

What does it mean to reverse a diffusion process?



Reversing a Diffusion Process



Definition 1

Let $\{X_t\}_{t\in[0,T]}$ be a forward diffusion process. A process $\{\widetilde{X}_t\}_{t\in[0,T]}$ is called its **reverse process** if

$$\widetilde{X}_t \overset{d}{=} X_{T-t}, \qquad \forall\, t \in [0,T].$$

- Although $\widetilde{X}_t$ runs forward in its own time variable $t \in [0,T]$, it represents the evolution of the original process backward from T to 0.
- The reverse process is generally *not unique*; different dynamics may induce the same time-reversed marginal distributions.

An Explicit Construction of Reverse Processes

- Consider a general forward diffusion process:

$$\mathrm{d}x_t = f(x_t, t)\ \mathrm{d}t + g(t)\ \mathrm{d}B_t, \qquad t \in [0, T].$$

[7] Brian Anderson, *Reverse-time diffusion equation models*, Stochastic Processes and their Applications 12 (1982)

An Explicit Construction of Reverse Processes

- Consider a general forward diffusion process:

$$\mathrm{d}x_t = f(x_t, t)\ \mathrm{d}t + g(t)\ \mathrm{d}B_t, \qquad t \in [0, T].$$

- **Anderson (1982)** [7] showed that the time-reversed process $\{\tilde{x}_t\}_{t\in[0,T]}$ satisfies the SDE

$$\boxed{\mathrm{d}\tilde{x}_t = \left[\, f(\tilde{x}_t, T-t) - g^2(T-t)\, \nabla_x \log p_{T-t}(\tilde{x}_t) \,\right] \mathrm{d}t \ + \ g(T-t)\ \mathrm{d}\bar{B}_t,}$$

where p_t is the density of x_t and $\bar{B}_t$ is a backward Brownian motion.

[7] Brian Anderson, *Reverse-time diffusion equation models*, Stochastic Processes and their Applications 12 (1982)

An Explicit Construction of Reverse Processes

- Consider a general forward diffusion process:

$$\mathrm{d}x_t = f(x_t, t)\ \mathrm{d}t + g(t)\ \mathrm{d}B_t, \qquad t \in [0, T].$$

- **Anderson (1982)** [7] showed that the time-reversed process $\{\tilde{x}_t\}_{t\in[0,T]}$ satisfies the SDE

$$\boxed{\mathrm{d}\tilde{x}_t = \left[\, f(\tilde{x}_t, T-t) - g^2(T-t)\, \nabla_x \log p_{T-t}(\tilde{x}_t) \,\right] \mathrm{d}t \ + \ g(T-t)\ \mathrm{d}\bar{B}_t,}$$

  where p_t is the density of x_t and $\bar{B}_t$ is a backward Brownian motion.
- The proof follows from differentiating the Fokker–Planck equation and verifying that the above SDE generates the reversed marginals p_{T-t}. (Details omitted.)

[7] Brian Anderson, *Reverse-time diffusion equation models*, Stochastic Processes and their Applications 12 (1982)

An Explicit Construction of Reverse Processes

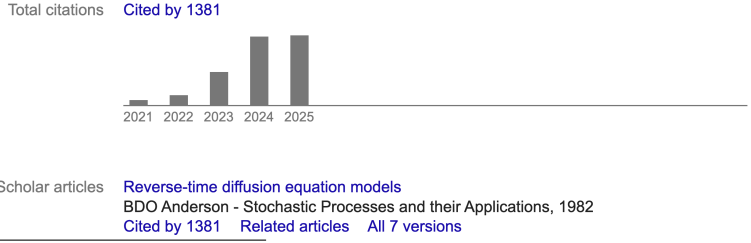
- Consider a general forward diffusion process:

$$\mathrm{d}x_t = f(x_t, t)\ \mathrm{d}t + g(t)\ \mathrm{d}B_t, \qquad t \in [0, T].$$

- **Anderson (1982)** [7] showed that the time-reversed process $\{\tilde{x}_t\}_{t\in[0,T]}$ satisfies the SDE

$$\boxed{\mathrm{d}\tilde{x}_t = \left[\, f(\tilde{x}_t, T-t) - g^2(T-t)\, \nabla_x \log p_{T-t}(\tilde{x}_t) \,\right] \mathrm{d}t \ + \ g(T-t)\ \mathrm{d}\bar{B}_t,}$$

  where p_t is the density of x_t and $\bar{B}_t$ is a backward Brownian motion.
- The proof follows from differentiating the Fokker–Planck equation and verifying that the above SDE generates the reversed marginals p_{T-t}. (Details omitted.)



[7] Brian Anderson, *Reverse-time diffusion equation models*, Stochastic Processes and their Applications 12 (1982)

Score Matching

- The key quantity for the reverse SDE is the (time-dependent) **score function**

$$\nabla_x \log p(\cdot, \cdot) : \mathbb{R}^d \times [0, T] \to \mathbb{R}^d,$$

Therefore, **we can easily reverse the process if the score function is known**.

Score Matching

- The key quantity for the reverse SDE is the (time-dependent) **score function**

$$\nabla_x \log p(\cdot, \cdot) : \mathbb{R}^d \times [0, T] \to \mathbb{R}^d,$$

  Therefore, **we can easily reverse the process if the score function is known**.
- **Model:** Let $s_\theta : \mathbb{R}^d \times [0, T] \to \mathbb{R}^d$ be a neural network to model the score function.

Score Matching

- The key quantity for the reverse SDE is the (time-dependent) **score function**

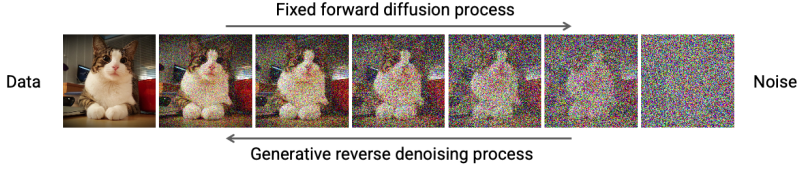
$$\nabla_x \log p(\cdot, \cdot) : \mathbb{R}^d \times [0, T] \to \mathbb{R}^d,$$

  Therefore, **we can easily reverse the process if the score function is known**.
- **Model:** Let $s_\theta : \mathbb{R}^d \times [0, T] \to \mathbb{R}^d$ be a neural network to model the score function.
- **Training objective:** Let $p_t = p(\cdot, t)$ and

$$L_t(\theta) = \mathbb{E}_{x \sim p_t} \left[\| s_\theta(x, t) - \nabla_x \log p(x, t) \|^2 \right].$$

  Let π be a (weighted) distribution supported on $[0, T]$. Consider the learning via

$$\min_\theta L(\theta) := \mathbb{E}_{t \sim \pi}[L_t(\theta)] \qquad \textbf{(score matching)}. \tag{2}$$



Score Matching (Cont'd)

- Why is this objective informative for training? The problem nearly becomes a sequential of supervised learning: Score matching at different times.

Score Matching (Cont'd)

- Why is this objective informative for training? The problem nearly becomes a sequential of supervised learning: Score matching at different times.
- **Bad News:** $\nabla_x \log p(\cdot, t)$ is unknown. Instead, we have only access to the noisy data $\{x_i(t)\}_{i\in[n]}$ generated by the forward process, starting from the clean data $\{x_i(0) = x_i\}_{i\in[n]}$.

Score Matching (Cont'd)

- Why is this objective informative for training? The problem nearly becomes a sequential of supervised learning: Score matching at different times.
- **Bad News:** $\nabla_x \log p(\cdot, t)$ is unknown. Instead, we have only access to the noisy data $\{x_i(t)\}_{i\in[n]}$ generated by the forward process, starting from the clean data $\{x_i(0) = x_i\}_{i\in[n]}$.
- **Approach:** Reformulate the objective so that it becomes an expectation over $p(\cdot, t)$ that can be estimated using the available noisy samples. (This is a general and powerful principle!)

Implicit Score Matching

We start by expanding the score-matching objective using the **log-derivative trick**:

$$\begin{aligned} L_t(\theta) &= \mathbb{E}_{x\sim p_t}\left[\|s_\theta(x,t) - \nabla_x \log p(x,t)\|^2\right] \\ &= \mathbb{E}_{p_t}\,\|s_\theta(x,t)\|^2 + \mathbb{E}_{p_t}\,\|\nabla_x \log p(x,t)\|^2 - 2\,\mathbb{E}_{p_t}\langle s_\theta(x,t), \nabla_x \log p(x,t)\rangle \\ &= \mathbb{E}_{p_t}\,\|s_\theta(x,t)\|^2 + \mathbb{E}_{p_t}\,\|\nabla_x \log p(x,t)\|^2 + 2\,\mathbb{E}_{p_t}[\nabla_x \cdot s_\theta(x,t)], \end{aligned}$$

where the second step use the log-derivative trick.

Implicit Score Matching

We start by expanding the score-matching objective using the **log-derivative trick**:

$$
\begin{aligned}
L_t(\theta) &= \mathbb{E}_{x\sim p_t}\left[\|s_\theta(x,t) - \nabla_x \log p(x,t)\|^2\right] \\
&= \mathbb{E}_{p_t}\,\|s_\theta(x,t)\|^2 + \mathbb{E}_{p_t}\,\|\nabla_x \log p(x,t)\|^2 - 2\,\mathbb{E}_{p_t}\langle s_\theta(x,t), \nabla_x \log p(x,t)\rangle \\
&= \mathbb{E}_{p_t}\,\|s_\theta(x,t)\|^2 + \mathbb{E}_{p_t}\,\|\nabla_x \log p(x,t)\|^2 + 2\,\mathbb{E}_{p_t}[\nabla_x \cdot s_\theta(x,t)],
\end{aligned}
$$

where the second step use the log-derivative trick.

Both blue and red terms are now expectations under p_t, but two issues remain:

- **Expensive:** The gradient of the red term requires $\nabla_\theta(\nabla_x \cdot s_\theta)$ — a second-order derivative.
- **High variance:** The divergence term $\nabla_x \cdot s_\theta$ is noisy and unstable during training.

Therefore we need an alternative formulation. The key observation: the conditional distribution $p_t(x|z)$ is tractable; see Eq. (5).

Denoising Score Matching

Noting that $p(x,t) = \int p_t(x|z)p_0(z)\,\mathrm{d}z$, we have

$$\begin{aligned}
\int_{\mathbb{R}^d} [\nabla_x \cdot s_\theta(x,t)]p(x,t)\,\mathrm{d}x &= \int_{\mathbb{R}} p_0(z)\,\mathrm{d}z \int_{\mathbb{R}^d} [\nabla_x \cdot s_\theta(x,t)]p_t(x|z)\,\mathrm{d}x \\
&= -\int_{\mathbb{R}} p_0(z)\,\mathrm{d}z \int_{\mathbb{R}^d} \langle s_\theta(x,t), \nabla_x p_t(x|z)\rangle\,\mathrm{d}x \\
&= -\int_{\mathbb{R}} p_0(z)\,\mathrm{d}z \int_{\mathbb{R}^d} \langle s_\theta(x,t), \nabla_x \log p_t(x|z)\rangle p_t(x|z)\,\mathrm{d}x \\
&= -\,\mathbb{E}_{z\sim p_0}\,\mathbb{E}_{x\sim p_t(\cdot|z)}\left[\langle s_\theta(x,t), \nabla_x \log p_t(x|z)\rangle\right]
\end{aligned}$$

Based on the preceding derivation, we have

$$\begin{aligned}
L_t(\theta) &= \mathbb{E}_{x\sim p_t}\left[\|s_\theta(x,t) - \nabla_x \log p(x,t)\|^2\right] \\
&= \mathbb{E}_{z\sim p_0}\,\mathbb{E}_{x\sim p_t(\cdot|z)}\,\|s_\theta(x,t)\|^2 - 2\,\mathbb{E}_{z\sim p_0}\,\mathbb{E}_{x\sim p_t(\cdot|z)}\left[\langle s_\theta(x,t), \nabla_x \log p_t(x|z)\rangle\right] + C \\
&= \mathbb{E}_{z\sim p_0}\,\mathbb{E}_{x\sim p_t(\cdot|z)}\left[\|s_\theta(x,t) - \nabla_x \log p_t(x|z)\|^2\right] + C
\end{aligned}$$

The key observation:

- In this formula, the input gradient term $\nabla_x \log p_t(x|z)$ is explicit, eliminating the need for backpropagation and making it well-controlled.

The Denoising/Noise-Prediction Interpretation

Consider the DDPM-type [8] forward process and let $\alpha_t = e^{-t/2}$ and $\sigma_t^2 = 1 - e^{-t}$. Then,

$$p_t(x|x_0) \propto \exp\left(-\frac{\|x - \alpha_t x_0\|^2}{2\sigma_t^2}\right).$$

Thus, the total objective becomes

$$L(\theta) = \mathbb{E}_t\, \mathbb{E}_{x_0}\, \mathbb{E}_{x_t|x_0} \left[\left\| s_\theta(x_t, t) - \frac{x_t - \alpha_t x_0}{\sigma_t^2} \right\|^2 \right] \tag{3}$$

[8] DDPM-type forward process gives an explicit conditional density.

The Denoising/Noise-Prediction Interpretation

Consider the DDPM-type [8] forward process and let $\alpha_t = e^{-t/2}$ and $\sigma_t^2 = 1 - e^{-t}$. Then,

$$p_t(x|x_0) \propto \exp\left(-\frac{\|x - \alpha_t x_0\|^2}{2\sigma_t^2}\right).$$

Thus, the total objective becomes

$$L(\theta) = \mathbb{E}_t\, \mathbb{E}_{x_0}\, \mathbb{E}_{x_t|x_0}\left[\left\| s_\theta(x_t, t) - \frac{x_t - \alpha_t x_0}{\sigma_t^2}\right\|^2\right] \tag{3}$$

Noting that $x_t|x_0 \sim \mathcal{N}(\alpha_t x_0, (1-\alpha_t) I_d)$, we can rewrite

$$\underbrace{x_t}_{\text{noisy sample}} = \alpha_t x_0 + \sqrt{1-\alpha_t^2}\xi_t = \underbrace{\alpha_t x_0}_{\text{clean sample}} + \underbrace{\sigma_t \xi_t}_{\text{noise}} \quad \text{with } \xi_t \sim \mathcal{N}(0, I_d).$$

- One can interpret $x_t - \alpha_t x_0$ as the "direction of denoising".
- Plugging it back into (3) gives the **noise-prediction** objective:

$$L(\theta) = \mathbb{E}_t\, \mathbb{E}_{x_0}\, \mathbb{E}_{\xi_t \sim \mathcal{N}(0, I_d)}\left[\left\| s_\theta(x_t, t) - \frac{\xi_t}{\sigma_t}\right\|^2\right]$$

[8] DDPM-type forward process gives an explicit conditional density.

Training Procedure

$$L(\theta) = \mathbb{E}_t \, \mathbb{E}_{x_0} \, \mathbb{E}_{\xi_t \sim \mathcal{N}(0, I_d)} \left[\left\| s_\theta(x_t, t) - \frac{\xi_t}{\sigma_t} \right\|^2 \right]$$

Parameterize s_θ with neural networks. Then, SGD of batch size 1 updates as follows:

Algorithm

- Step 1: $t \sim \pi, x_0 \sim p_0, \xi_t \sim \mathcal{N}(0, I_d)$
- Step 2: $\alpha_t = e^{-t/2}, x_t = \alpha_t x_0 + \sqrt{1 - \alpha_t} \xi_t$
- Step 3: $\hat{L}(\theta) = \left\| s_\theta(x_t, t) - \frac{\xi_t}{\sigma_t} \right\|^2$
- Step 4: $\theta_{k+1} = \theta_k - \eta \nabla_\theta \hat{L}(\theta_k)$

The Choice of Time Weighting

How to choose π?

$$L(\theta) = \mathbb{E}_{t\sim\pi}\, \mathbb{E}_{x_0}\, \mathbb{E}_{\xi_t\sim\mathcal{N}(0,I_d)} \left[\left\| s_\theta(x_t, t) - \frac{\xi_t}{\sigma_t} \right\|^2 \right]$$

Key observation: When $t \to 0$, $\sigma_t = \sqrt{1 - e^{-t}} \to 0$. Loss heavily amplified when sampling t close to 0. High variance!

[9] Take a look this note

The Choice of Time Weighting

How to choose π?

$$L(\theta) = \mathbb{E}_{t\sim\pi}\, \mathbb{E}_{x_0}\, \mathbb{E}_{\xi_t\sim\mathcal{N}(0,I_d)} \left[\left\| s_\theta(x_t, t) - \frac{\xi_t}{\sigma_t} \right\|^2 \right]$$

Key observation: When $t \to 0$, $\sigma_t = \sqrt{1 - e^{-t}} \to 0$. Loss heavily amplified when sampling t close to 0. High variance!

- Training with **time cut-off** η:

$$\pi = \mathrm{Unif}([\eta, T]).$$

- Variance reduction via **importance sampling** [9]:

$$\pi(t) \propto \frac{1}{\sigma_t^2}.$$

  Intuitively, we need more samples for when t is close to 0, otherwise the Monte-Carlo estimate may give a rather wrong estimate.

[9] Take a look this note

Probability-Flow ODE

- The reverse SDE [10] is given by

$$\mathrm{d}\tilde{x}_t = \left[f(\tilde{x}_t, t) - g^2(t)\nabla_x \log p(\tilde{x}, t)\right]\mathrm{d}t + g(t)\,\mathrm{d}\bar{B}_t.$$

(Song et al. 2021) showed that the following probability-flow ODE is also a reverse process

$$\mathrm{d}\tilde{x}_t = f(\tilde{x}_t, t) - \frac{1}{2}g^2(t)\nabla_x \log p(\tilde{x}, t)\,\mathrm{d}t.$$

[10] Note: For notational simplicity, we reuse t to index the reverse process (technically running in time $T - t$).

Probability-Flow ODE

- The reverse SDE [10] is given by

$$\mathrm{d}\tilde{x}_t = \left[f(\tilde{x}_t, t) - g^2(t)\nabla_x \log p(\tilde{x}, t)\right] \mathrm{d}t + g(t)\, \mathrm{d}\bar{B}_t.$$

  (Song et al. 2021) showed that the following probability-flow ODE is also a reverse process

$$\mathrm{d}\tilde{x}_t = f(\tilde{x}_t, t) - \frac{1}{2} g^2(t)\nabla_x \log p(\tilde{x}, t)\, \mathrm{d}t.$$

- For the DDPM-type forward process, the probability-flow ODE becomes

$$\mathrm{d}\tilde{x}_t = -\frac{1}{2}(x_t + \nabla_x \log p(\tilde{x}, t))\, \mathrm{d}t$$

- The probability-flow ODE can be interpreted as a continuous-time normalizing flow (CNF).

[10] Note: For notational simplicity, we reuse t to index the reverse process (technically running in time $T - t$).

A Comparison between SDE and ODE for Generating Samples

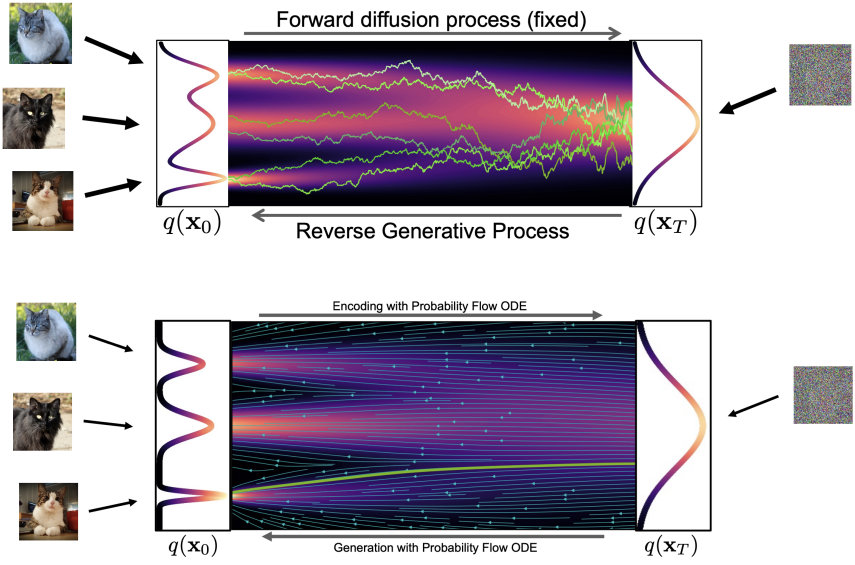


Figure 2: (Up) SDE; (Down) ODE.

ODE Enables Faster Sampling

- Sampling in diffusion models requires discretizing the reverse dynamics (SDE or ODE). Small step sizes are needed to ensure accuracy and stability.

ODE Enables Faster Sampling

- Sampling in diffusion models requires discretizing the reverse dynamics (SDE or ODE). Small step sizes are needed to ensure accuracy and stability.
- **SDE limitation:** Diffusion trajectories are nowhere differentiable $\Rightarrow$
  - no higher-order numerical solvers exist in general,
  - Euler–Maruyama is essentially the only choice.

ODE Enables Faster Sampling

- Sampling in diffusion models requires discretizing the reverse dynamics (SDE or ODE). Small step sizes are needed to ensure accuracy and stability.
- **SDE limitation:** Diffusion trajectories are nowhere differentiable $\Rightarrow$
  - no higher-order numerical solvers exist in general,
  - Euler–Maruyama is essentially the only choice.
- **Advantage of the probability-flow ODE:**
  - trajectories are smooth and deterministic,
  - high-order ODE solvers (Runge–Kutta, adaptive solvers) can be used,
  - significantly fewer steps are required to reach good sample quality.

ODE Enables Exact Likelihood Computation

- The probability-flow ODE defines a **continuous normalizing flow**:

$$\dot{x}_t = f(x_t, t), \qquad x_0 \sim p_0.$$

ODE Enables Exact Likelihood Computation

- The probability-flow ODE defines a **continuous normalizing flow**:

$$\dot{x}_t = f(x_t, t), \qquad x_0 \sim p_0.$$

- Let Φ_T be the flow map such that $x_T = \Phi_T(x_0)$. Then the exact log-likelihood satisfies:

$$\boxed{\log p_T(x_T) = \log p_0(x_0) - \int_0^T \nabla \cdot f(x_t, t)\ \mathrm{d}t.}$$

ODE Enables Exact Likelihood Computation

- The probability-flow ODE defines a **continuous normalizing flow**:

$$\dot{x}_t = f(x_t, t), \qquad x_0 \sim p_0.$$

- Let Φ_T be the flow map such that $x_T = \Phi_T(x_0)$. Then the exact log-likelihood satisfies:

$$\boxed{\log p_T(x_T) = \log p_0(x_0) - \int_0^T \nabla \cdot f(x_t, t)\ \mathrm{d}t.}$$

- Computing divergence directly costs $O(d)$ backpropagations. Using the Hutchinson–Skilling estimator:

$$\nabla \cdot h(x) = \mathbb{E}_\epsilon\left[\epsilon^\top \nabla h(x) \epsilon\right] \approx \frac{1}{m} \sum_{j=1}^{m} \epsilon_j^\top \nabla h(x) \epsilon_j,$$

where $\epsilon_j \sim \mathrm{Unif}(\{\pm 1\}^d)$.

ODE Enables Exact Likelihood Computation

- The probability-flow ODE defines a **continuous normalizing flow**:

$$\dot{x}_t = f(x_t, t), \qquad x_0 \sim p_0.$$

- Let Φ_T be the flow map such that $x_T = \Phi_T(x_0)$. Then the exact log-likelihood satisfies:

$$\boxed{\log p_T(x_T) = \log p_0(x_0) - \int_0^T \nabla \cdot f(x_t, t)\ \mathrm{d}t.}$$

- Computing divergence directly costs $O(d)$ backpropagations. Using the Hutchinson–Skilling estimator:

$$\nabla \cdot h(x) = \mathbb{E}_\epsilon\left[\epsilon^\top \nabla h(x) \epsilon\right] \approx \frac{1}{m} \sum_{j=1}^{m} \epsilon_j^\top \nabla h(x) \epsilon_j,$$

  where $\epsilon_j \sim \mathrm{Unif}(\{\pm 1\}^d)$.

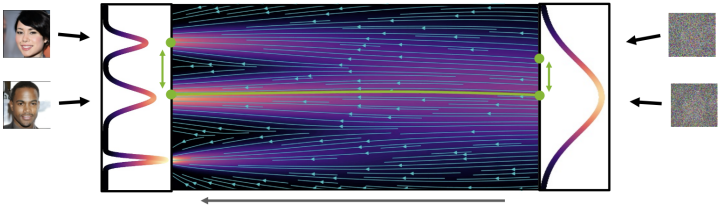
- **Implication:** Diffusion models become *exact likelihood models* under the ODE formulation.

ODE Enables Controllable Latent Space

The ODE defines a deterministic map, which allows:

- smooth interpolations in latent space,
- vector arithmetic and semantic manipulation.

Interpolation



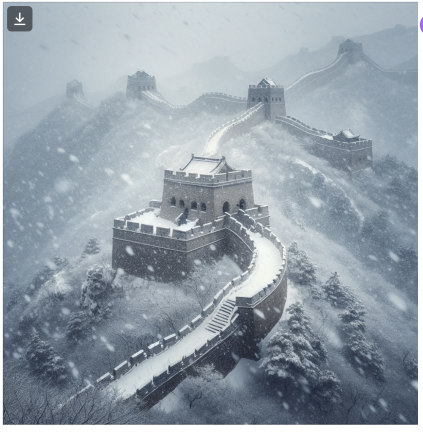
Controllable Generalization

Generate One-Class of Samples

Text to Images

You
请画一幅暴雪中的长城景色

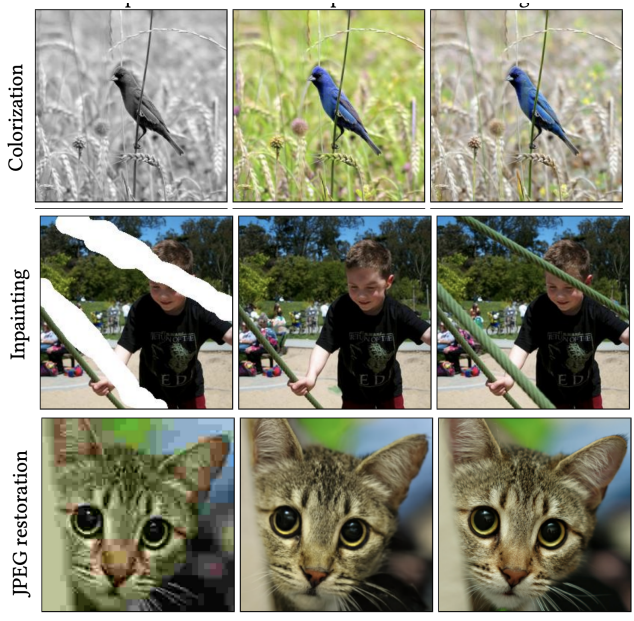
ChatGPT



You
请用八大山人的风格重绘这幅画，不要改变内容，只改变风格。请注意留白和落款，不用输出任何文字描述。

ChatGPT

Some Classical Tasks



Controllable Generalization via Diffusion Models

- Controlled generalization can be modeled as sampling from $P(x|y)$ where y denotes the control factor.

Controllable Generalization via Diffusion Models

- Controlled generalization can be modeled as sampling from $P(x|y)$ where y denotes the control factor.
- Baye's rule:$P(x|y) = \frac{P(y|x)P(x)}{P(y)}$. Accordingly, the score function:

$$\nabla_x \log P(x|y) = \nabla_x \log P(y|x) + \nabla_x \log P(x) \tag{4}$$

where the normalizing constant disappears.

Controllable Generalization via Diffusion Models

- Controlled generalization can be modeled as sampling from $P(x|y)$ where y denotes the control factor.
- Baye's rule:$P(x|y) = \frac{P(y|x)P(x)}{P(y)}$. Accordingly, the score function:

$$\nabla_x \log P(x|y) = \nabla_x \log P(y|x) + \nabla_x \log P(x) \tag{4}$$

  where the normalizing constant disappears.
- Then we can directly couple (4) with the reverse-time SDE or ODE:

$$\begin{aligned}\dot{\tilde{x}}_t &= f(\tilde{x}_t, t) - \frac{1}{2} g(t)^2 \log p_t(\tilde{x}|y) \\ &= f(\tilde{x}_t, t) - \frac{1}{2} \nabla_x \left[\log p(\tilde{x}_t, t) + \log p(y|\tilde{x}_t)\right].\end{aligned}$$

Controllable Generalization via Diffusion Models

- Controlled generalization can be modeled as sampling from $P(x|y)$ where y denotes the control factor.
- Baye's rule:$P(x|y) = \frac{P(y|x)P(x)}{P(y)}$. Accordingly, the score function:

$$\nabla_x \log P(x|y) = \nabla_x \log P(y|x) + \nabla_x \log P(x) \tag{4}$$

  where the normalizing constant disappears.
- Then we can directly couple (4) with the reverse-time SDE or ODE:

$$\begin{aligned}\dot{\tilde{x}}_t &= f(\tilde{x}_t, t) - \frac{1}{2} g(t)^2 \log p_t(\tilde{x}|y) \\ &= f(\tilde{x}_t, t) - \frac{1}{2} \nabla_x \left[\log p(\tilde{x}_t, t) + \log p(y|\tilde{x}_t)\right].\end{aligned}$$

- Therefore, as long as we have a good "classifier" $p(y|x)$, then we can couple it with the unconditional model $s_\theta(x,t) \approx \nabla \log p(x,t)$ in a very **simple and principled** approach.

Controllable Generalization via Diffusion Models

- Controlled generalization can be modeled as sampling from $P(x|y)$ where y denotes the control factor.
- Baye's rule:$P(x|y) = \frac{P(y|x)P(x)}{P(y)}$. Accordingly, the score function:

$$\nabla_x \log P(x|y) = \nabla_x \log P(y|x) + \nabla_x \log P(x) \tag{4}$$

  where the normalizing constant disappears.
- Then we can directly couple (4) with the reverse-time SDE or ODE:

$$\begin{aligned}\dot{\tilde{x}}_t &= f(\tilde{x}_t, t) - \frac{1}{2} g(t)^2 \log p_t(\tilde{x}|y) \\ &= f(\tilde{x}_t, t) - \frac{1}{2} \nabla_x \left[\log p(\tilde{x}_t, t) + \log p(y|\tilde{x}_t)\right].\end{aligned}$$

- Therefore, as long as we have a good "classifier" $p(y|x)$, then we can couple it with the unconditional model $s_\theta(x,t) \approx \nabla \log p(x,t)$ in a very **simple and principled** approach.
- **One unconditional models for all tasks**.

Connection with Energy-based Models

- In EBM, $p(x) = e^{-U(x)}/Z$. We learn a potential **energy** $V_\theta(x) \approx U(x) = -\nabla \log p(x)$. That is, score matching is also commonly used in learning energy-based model.
- In score-based models, we learn $s_\theta(x) \approx \nabla_x \log p(x) = -\nabla_x U(x)$, i.e., the **force**.
- With the score functions (aka. the force field), we can also recover samples by running Langevin dynamics

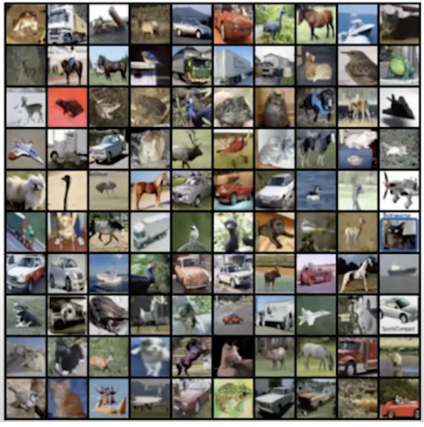
$$\mathrm{d}x_t = -\nabla_x \log p(x_t)\,\mathrm{d}t + \sqrt{2}\,\mathrm{d}B_t.$$

  But its performance is notorious and consequently, using the reverse-time SDE/ODE is always much better.

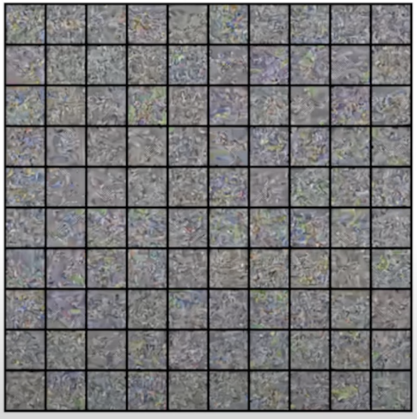
Naive Score Matching + Langevin Dynamics

Often, the learned score function is useless when simulating Langevin dynamics.

CIFAR-10 data



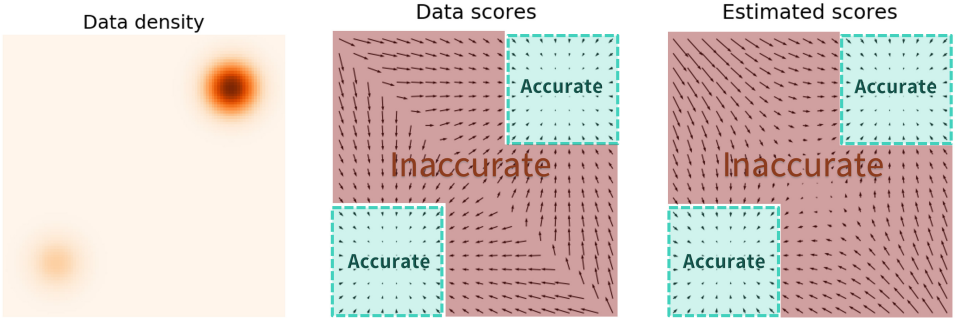
Model samples



Reason 1: Inaccurate Score Functions

The learned score function are inaccurate in the low-density region.

$$L(\theta) = \mathbb{E}_x \left\| s_\theta(x) - \nabla \log p(x) \right\|^2.$$



Reason 2: Sampling with Langevin Dynamics is Slow

- When the target distribution is multimodal, Langevin dynamics (or more generally, MCMC methods) struggle to efficiently sample across different modes.

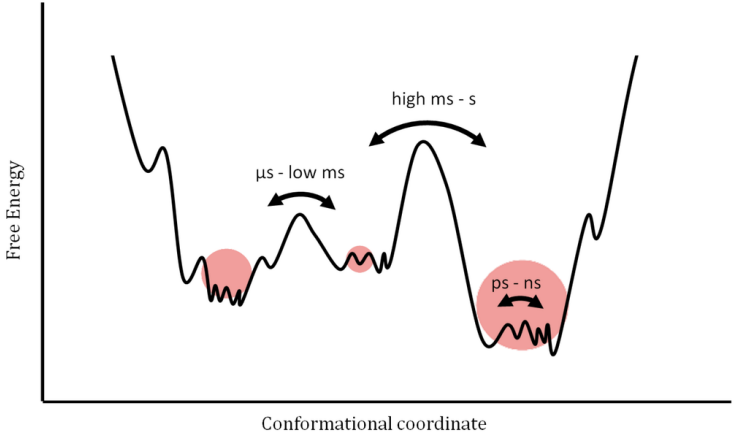


Figure 3: The sampling rate suffers from the curse of dimensionality and loss barrier.

Summary

Advantages

- A principled design grounded in diffusion and score-matching theory, leading to stable training.

Summary

Advantages

- A principled design grounded in diffusion and score-matching theory, leading to stable training.
- High modeling flexibility:

Summary

Advantages

- A principled design grounded in diffusion and score-matching theory, leading to stable training.
- High modeling flexibility:
  - Unlike normalizing flows (e.g., RealNVP), diffusion models do not require invertible architectures; *any* neural network can parameterize the score.

Summary

Advantages

- A principled design grounded in diffusion and score-matching theory, leading to stable training.
- High modeling flexibility:
  - Unlike normalizing flows (e.g., RealNVP), diffusion models do not require invertible architectures; *any* neural network can parameterize the score.
  - Naturally supports controllable generation through conditional score modeling.

Summary

Advantages

- A principled design grounded in diffusion and score-matching theory, leading to stable training.
- High modeling flexibility:
  - Unlike normalizing flows (e.g., RealNVP), diffusion models do not require invertible architectures; *any* neural network can parameterize the score.
  - Naturally supports controllable generation through conditional score modeling.
- As a result, diffusion models can generate remarkably high-quality and diverse samples.

Summary

Advantages

- A principled design grounded in diffusion and score-matching theory, leading to stable training.
- High modeling flexibility:
  - Unlike normalizing flows (e.g., RealNVP), diffusion models do not require invertible architectures; *any* neural network can parameterize the score.
  - Naturally supports controllable generation through conditional score modeling.
- As a result, diffusion models can generate remarkably high-quality and diverse samples.

Summary

Advantages

- A principled design grounded in diffusion and score-matching theory, leading to stable training.
- High modeling flexibility:
  - Unlike normalizing flows (e.g., RealNVP), diffusion models do not require invertible architectures; *any* neural network can parameterize the score.
  - Naturally supports controllable generation through conditional score modeling.
- As a result, diffusion models can generate remarkably high-quality and diverse samples.

Disadvantage:

- Sampling is slow: the reverse process requires many iterative denoising steps due to time discretization.

Summary

Advantages

- A principled design grounded in diffusion and score-matching theory, leading to stable training.
- High modeling flexibility:
  - Unlike normalizing flows (e.g., RealNVP), diffusion models do not require invertible architectures; *any* neural network can parameterize the score.
  - Naturally supports controllable generation through conditional score modeling.
- As a result, diffusion models can generate remarkably high-quality and diverse samples.

Disadvantage:

- Sampling is slow: the reverse process requires many iterative denoising steps due to time discretization.

Further reading:

- Stanley H. Chan, *Tutorial on Diffusion Models for Imaging and Vision*, arXiv:2403.18103.