Stochastic Differential Equations with Applications to Diiffusion and Flow-based models

Kintan Saha, Jithendra Rao Kasibhatla

May 2025

The Generative Process

- Modern generative models generate objects(which are represented as vectors $\in \mathbb{R}^d$) by iteratively converting **noise into data**.
- The target data will have an underlying distribution p_{data}. We have access to a dataset $z_1, z_2 \ldots z_N \sim p_{data}$
- This transformation is facilitated by simulating Ordinary or Stochastic Differential Equations (ODEs/SDEs).
- Flow matching and denoising diffusion models are techniques to:
  - Construct
  - Train
  - Simulate

  such ODEs/SDEs at large scale with deep neural networks.

The Probabilistic View of Generation

Key Idea (Generation as Sampling)

Generating an object z is modeled as sampling from the data distribution $z \sim p_{data}$.

- Often, we want to generate an object conditioned on some data y.
- Example: Generate an image conditioned on $y =$ "a dog running down a hill...".
- This is sampling from a conditional distribution.

Key Idea (Conditional Generation)

Conditional generation involves sampling $z \sim p_{data}(\cdot|y)$, where y is a conditioning variable.

- Unconditional generation can be easily generalised to conditional generation

The Transformation Process

- **How is generation achieved?**
- Assume access to an initial distribution p_{init} that is easy to sample from (e.g., Gaussian $\mathcal{N}(0, I_d)$).
- Goal of a generative model: Transform samples $x \sim p_{init}$ into samples from p_{data}.
- p_{init} doesn't have to be a simple Gaussian; this offers flexibility.

Differential Equations for Transformation

- Previous section: Generative modeling is sampling from p_{data}.
- This sampling can be achieved by transforming samples from a simple p_{init} (e.g., $\mathcal{N}(0, I_d)$) to p_{data}.
- This section: How this transformation is obtained via simulating differential equations.
- Flow Matching: Uses **Ordinary Differential Equations (ODEs)**.
- Diffusion Models: Uses **Stochastic Differential Equations (SDEs)**.

Defining ODEs, Trajectories and Flows

- A solution to an ODE is a trajectory: $X : [0,1] \to \mathbb{R}^d$, $t \mapsto X_t$.
- Every ODE is defined by a **vector field** u: $u : \mathbb{R}^d \times [0,1] \to \mathbb{R}^d$, $(x,t) \mapsto u_t(x)$.
- $u_t(x)$ specifies a velocity in space at time t and location x.
- ODE: Trajectory X_t is governed by u_t, starting at x_0.

$$\frac{d}{dt}X_t = u_t(X_t) \qquad \text{(ODE)}$$

$$X_0 = x_0 \qquad \text{(initial conditions)}$$

- Flow(ψ): Family of trajectories obtained from the ODE under different initial conditions

$$\psi : \mathbb{R}^d \times [0,1] \mapsto \mathbb{R}^d, \quad (x_0, t) \mapsto \psi_t(x_0)$$

$$\frac{d}{dt}\psi_t(x_0) = u_t(\psi_t(x_0)) \qquad \text{(flow ODE)}$$

$$\psi_0(x_0) = x_0 \qquad \text{(flow initial conditions)}$$

Visualising Flows

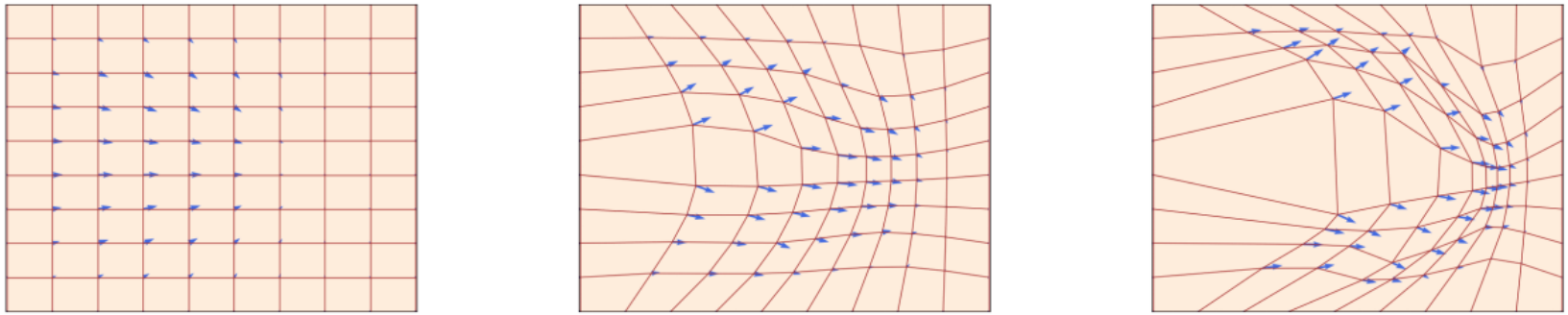


Figure 1: A flow $\psi_t : \mathbb{R}^d \to \mathbb{R}^d$ (red square grid) is defined by a velocity field $u_t : \mathbb{R}^d \to \mathbb{R}^d$ (visualized with blue arrows) that prescribes its instantaneous movements at all locations (here, $d = 2$). We show three different times t. As one can see, a flow is a diffeomorphism that "warps" space. Figure from [15].

Figure: Evolution of trajectories

Flow Existence and Uniqueness

- Does a solution exist? Is it unique? Yes, under weak assumptions on u_t.
- **Flow existence and uniqueness:** If $u : \mathbb{R}^d \times [0,1] \to \mathbb{R}^d$ is continuously differentiable with a bounded derivative, then the corresponding ODE in has a unique solution given by a flow ψ_t. In this case, ψ_t is a diffeomorphism for all t.
- This assumption is almost always fulfilled in machine learning (modelling $u_t(x)$ using neural networks)
- Now, how do solve ODEs numerically?

Numerical Methods]

- Explicit computation of trajectories(and hence flow) is often not possible for complex u_t.
- Numerical methods are used for simulation
- **Euler Method**:
  - Initialize $X_0 = x_0$.
  - Update: $X_{t+h} = X_t + hu_t(X_t)$ for $t = 0, h, 2h, ..., 1 - h$.
  - $h = \frac{1}{n}$ is the step size.
- **Heun's Method**:
  - $X'_{t+h} = X_t + hu_t(X_t)$ (initial guess)
  - $X_{t+h} = X_t + \frac{h}{2}(u_t(X_t) + u_{t+h}(X'_{t+h}))$ (update with average u)
- Other methods such as Runge-Kutta (RK) methods also exist

Constructing a Generative Flow Model

- Goal: Convert simple p_{init} into complex p_{data}.
- Simulation of an ODE is a natural choice for this transformation.
- A **flow model** is described by:

$$X_0 \sim p_{init} \qquad \text{(random initialization)}$$

$$\frac{d}{dt}X_t = u_t^\theta(X_t) \qquad \text{(ODE)}$$

- Vector field u_t^θ is a **neural network** parametrized by θ.
- Goal: Make the endpoint X_1 of the trajectory have distribution p_{data}, i.e., $X_1 = \psi_1^\theta(X_0) \sim p_{data}$.
- Note: Neural network parameterizes the vector field, not the flow directly.

Euler Method for Flow Model Sampling

Algorithm 1: Sampling from a Flow Model (Euler method)

Require: Neural network vector field u_t^θ, number of steps n.

1. Set $t = 0$.
2. Set step size $h = \frac{1}{n}$.
3. Draw a sample $X_0 \sim p_{init}$.
4. **for** $i = 0, \ldots, n-1$ **do** (Note: original doc says $n-1$ iterations, typical is n steps from $t = 0$ to $t = 1$)
   1. $X_{t+h} = X_t + h u_t^\theta(X_t)$.
   2. Update $t \leftarrow t + h$.
5. **end for**
6. **return** X_1.

Introducing Randomness

- SDEs extend deterministic ODE trajectories to **stochastic** trajectories.
- A stochastic trajectory is a **stochastic process** $(X_t)_{0 \le t \le 1}$.
- X_t is a random variable for every $0 \le t \le 1$.
- $X : [0,1] \to \mathbb{R}^d, t \mapsto X_t$ is a random trajectory for every initialization.

$$dX_t = u_t(X_t)dt + \sigma_t dW_t$$
$$X_0 = x_0$$

- Simulating the same SDE twice can yield different outcomes due to stochastic dynamics.

Brownian Motion

- SDEs are constructed via **Brownian motion**(also called a Wiener process) $(W_t)_{0 \le t \le 1}$, a fundamental stochastic process
- Properties of Brownian Motion W:
  1. $W_0 = 0$.
  2. Trajectories $t \mapsto W_t$ are continuous.
  3. **Normal increments**: $W_t - W_s \sim \mathcal{N}(0, (t-s)I_d)$ for $0 \le s < t$.
  4. **Independent increments**: For $0 \le t_0 < \ldots < t_n = 1$, increments $W_{t_1} - W_{t_0}, \ldots, W_{t_n} - W_{t_{n-1}}$ are independent.
- Simulation of Brownian motion: $W_0 = 0$, $W_{t+h} = W_t + \sqrt{h}\epsilon_t$, where $\epsilon_t \sim \mathcal{N}(0, I_d)$.

Visualization of Brownian Motion

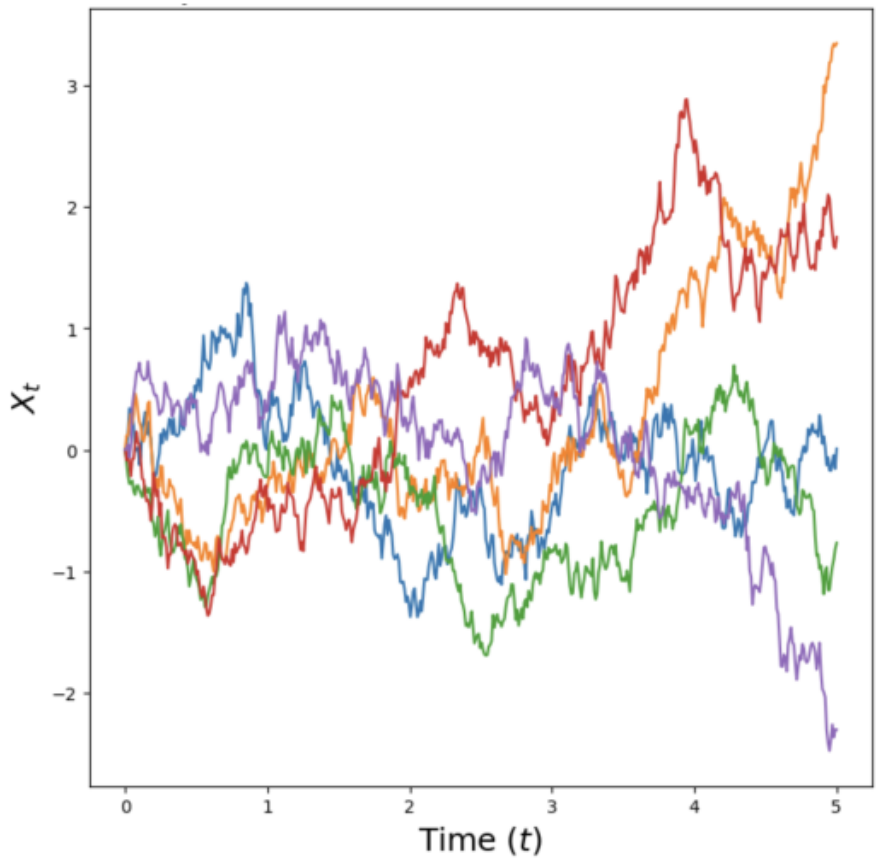


Figure 2: Sample trajectories of a Brownian motion W_t in dimension $d = 1$ simulated using eq. (5).

From ODEs to SDEs

- **Idea:** Extend ODE dynamics by adding stochastic dynamics driven by Brownian motion.
- ODE (infinitesimal update form):

$$X_{t+h} = X_t + hu_t(X_t) + hR_t(h), \quad \text{where } R_t(h) \to 0 \text{ as } h \to 0.$$

- SDE (infinitesimal update form):

$$X_{t+h} = X_t + hu_t(X_t) + \sigma_t(W_{t+h} - W_t) + hR_t(h)$$

- $\sigma_t \geq 0$ is the diffusion coefficient.
- Every ODE is an SDE with $\sigma_t = 0$
- Similar uniqueness result holds for SDEs

Example: Ornstein-Uhlenbeck Process

- Constant diffusion $\sigma_t = \sigma \geq 0$, linear drift $u_t(x) = -\theta x$ for $\theta > 0$.
- SDE: $dX_t = -\theta X_t dt + \sigma dW_t$.

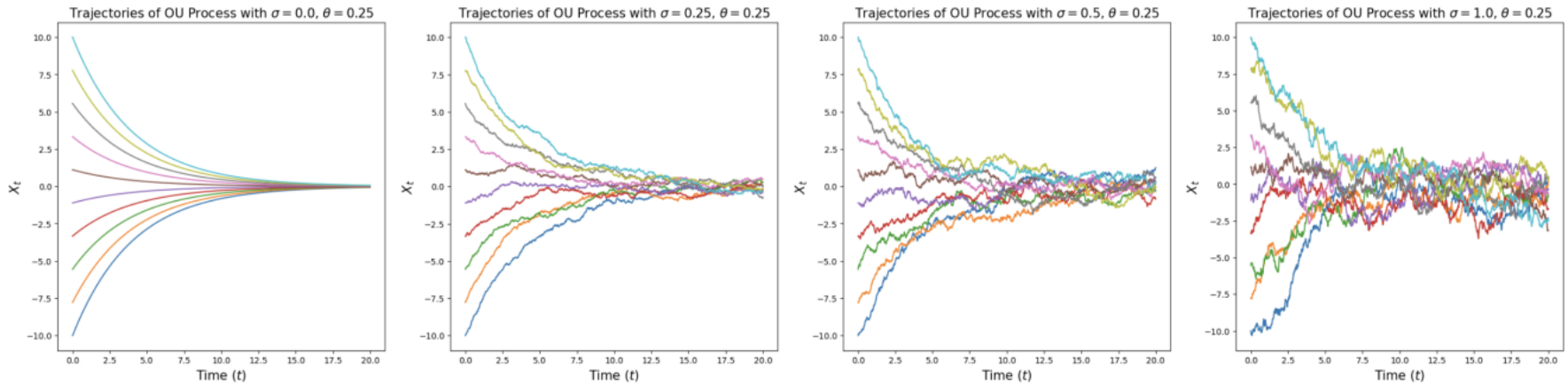


Figure 3: Illustration of Ornstein-Uhlenbeck processes (eq. (8)) in dimension $d = 1$ for $\theta = 0.25$ and various choices of σ (increasing from left to right). For $\sigma = 0$, we recover a flow (smooth, deterministic trajectories) that converges to the origin as $t \to \infty$. For $\sigma > 0$ we have random paths which converge towards the Gaussian $\mathcal{N}(0, \frac{\sigma^2}{2\theta})$ as $t \to \infty$.

Figure: Evolution of trajectories under different σ_t

Simulating an SDE

- How to simulate an SDE?
- **Euler-Maruyama method:**
  - Initialize $X_0 = x_0$.
  - Update iteratively: $X_{t+h} = X_t + hu_t(X_t) + \sqrt{h}\sigma_t\epsilon_t$, where $\epsilon_t \sim \mathcal{N}(0, I_d)$.
  - $h = \frac{1}{n}$ is the step size.
- Intuitively, take a small step in direction $u_t(X_t)$ and add Gaussian noise scaled by $\sqrt{h}\sigma_t$.

Diffusion Models for Generative Modeling

- Goal: Convert simple p_{init} into complex p_{data}.
- Simulate an SDE initialized to $X_0 \sim p_{init}$
- Parameterize **only** the vector field u_t by a neural network u_t^θ
- A **diffusion model** is given by:

$$dX_t = u_t^\theta(X_t)dt + \sigma_t dW_t \qquad \text{(SDE)}$$
$$X_0 \sim p_{init} \qquad \text{(random initialization)}$$

- The diffusion coefficient σ_t is a known quantity and isn't parametrized by a neural network

Euler-Maruyama for Diffusion Model Sampling

Algorithm 2: Sampling from a Diffusion Model (Euler-Maruyama)

Require: Neural network u_t^θ, number of steps n, diffusion coefficient σ_t.

1. Set $t = 0$.
2. Set step size $h = \frac{1}{n}$.
3. Draw a sample $X_0 \sim p_{init}$.
4. **for** $i = 0, \ldots, n-1$ **do**
   1. Draw a sample $\epsilon \sim \mathcal{N}(0, I_d)$.
   2. $X_{t+h} = X_t + h u_t^\theta(X_t) + \sigma_t \sqrt{h} \epsilon$.
   3. Update $t \leftarrow t + h$.
5. **end for**
6. **return** X_1.

Key Components of a Diffusion Model

Summary: SDE Generative Model

A diffusion model consists of:

- **Neural network**: $u^\theta : \mathbb{R}^d \times [0,1] \to \mathbb{R}^d, (x,t) \mapsto u_t^\theta(x)$ with parameters θ.
- **Fixed**: $\sigma_t : [0,1] \to [0,\infty), t \mapsto \sigma_t$ (diffusion coefficient).

Procedure to obtain samples:

- **Initialization**: $X_0 \sim p_{init}$ (e.g., Gaussian).
- **Simulation**: $dX_t = u_t^\theta(X_t)dt + \sigma_t dW_t$ (Simulate SDE from 0 to 1).
- **Goal**: $X_1 \sim p_{data}$.

A diffusion model with $\sigma_t = 0$ is a flow model.

The Need for a Training Target

- In the previous section, we defined the model as:

$$X_0 \sim p_{\text{init}}, \quad dX_t = u_t^\theta(X_t)dt \quad (\text{or } + \sigma_t dW_t \text{ for SDEs})$$

- A randomly initialized neural network u_t^θ will produce meaningless results.
- Therefore, we must train u_t^θ by minimizing a loss function $L(\theta)$.
- A simple example: mean squared error loss

$$L(\theta) = \|u_t^\theta(x) - u_t^{\text{target}}(x)\|^2$$

- We will now aim to derive a suitable expression for $u_t^{\text{target}} : \mathbb{R}^d \times [0,1] \to \mathbb{R}^d$.

Probability Path: Interpolating Distributions

- First step: Specify a **probability path** $p_t(x|z)$: a trajectory in the space of distributions
- Can be interpreted as a gradual interpolation between noise p_{init} and a data point z.
- For data point $z \in \mathbb{R}^d$, δ_z is the Dirac delta distribution (always returns z).
- A **conditional (interpolating) probability path** $p_t(x|z)$ satisfies:

$$p_0(\cdot|z) = p_{init}, \quad p_1(\cdot|z) = \delta_z \quad \text{for all } z \in \mathbb{R}^d$$

Marginal Probability Path

- The conditional paths $p_t(x|z)$ induces a **marginal probability path** $p_t(x)$.
- Sampling from marginal path: $z \sim p_{data}, x \sim p_t(\cdot|z) \implies x \sim p_t$.
- Density of marginal path: $p_t(x) = \int p_t(x|z)p_{data}(z)dz$.
- We know how to sample from p_t, but the $p_t(x)$ is often intractable.
- By construction, The marginal path p_t interpolates between p_{init} and p_{data}:

$$p_0 = p_{init} \quad \text{and} \quad p_1 = p_{data}$$

Example: Gaussian Conditional Probability Path

- Let α_t, β_t be noise schedulers: continuously differentiable, monotonic functions with $\alpha_0 = \beta_1 = 0$ and $\alpha_1 = \beta_0 = 1$.
- Conditional path: $p_t(\cdot|z) = \mathcal{N}(\alpha_t z, \beta_t^2 I_d)$.
- Sampling from marginal Gaussian path:
  $z \sim p_{data}, \epsilon \sim p_{init} = \mathcal{N}(0, I_d) \implies x = \alpha_t z + \beta_t \epsilon \sim p_t$.

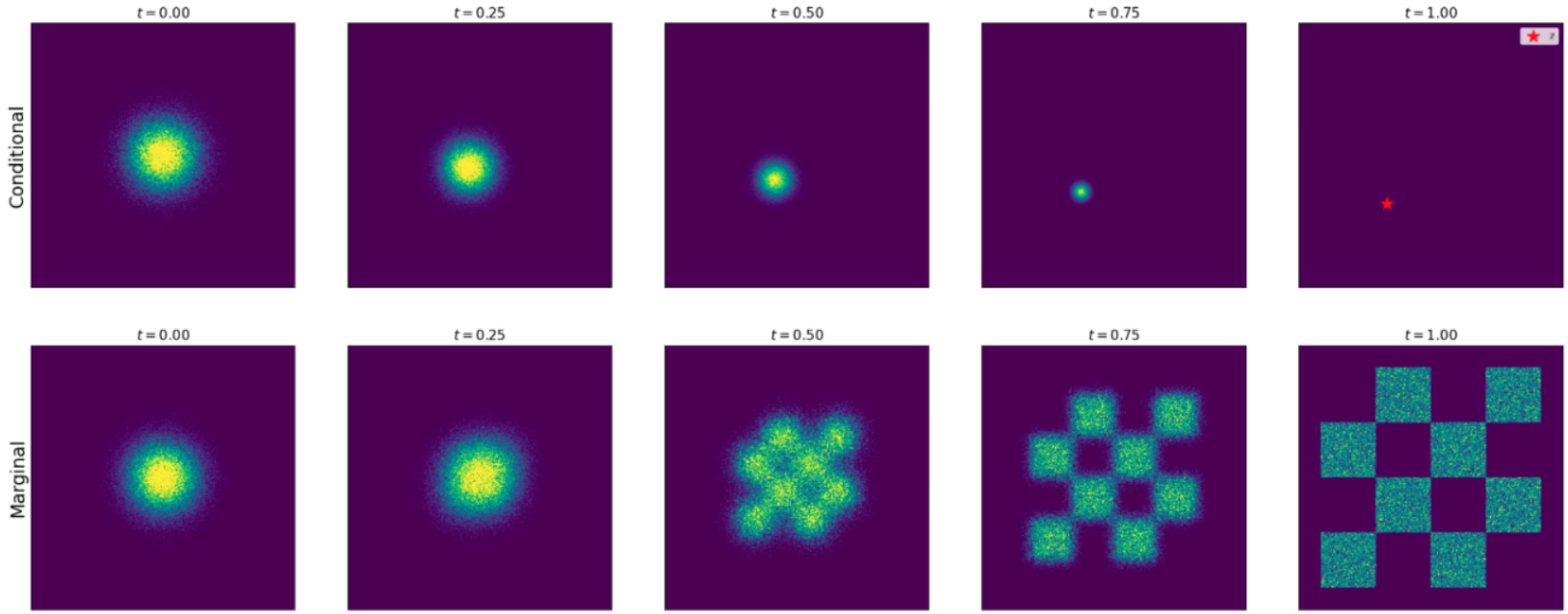


Figure: Gaussian probability paths with $\alpha_t = t, \beta_t = 1 - t$

Constructing u_t^{target} for Flow Models

- Goal: Construct u_t^{target} for a flow model using probability path p_t.
- For every $z \in \mathbb{R}^d$, let $u_t^{\text{target}}(\cdot|z)$ be the **conditional vector field** such that its ODE yields $p_t(\cdot|z)$:

$$X_0 \sim p_{init}, \frac{d}{dt}X_t = u_t^{\text{target}}(X_t|z) \implies X_t \sim p_t(\cdot|z).$$

- **Marginalization trick:** The marginal vector field $u_t^{\text{target}}(x)$, defined by

$$u_t^{\text{target}}(x) = \int u_t^{\text{target}}(x|z)\frac{p_t(x|z)p_{data}(z)}{p_t(x)}dz,$$

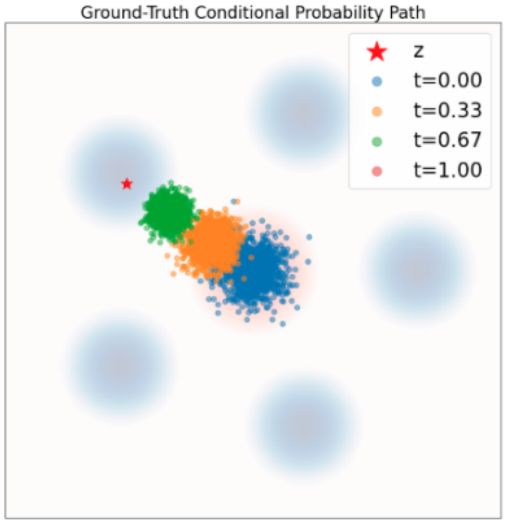
follows the marginal probability path p_t:

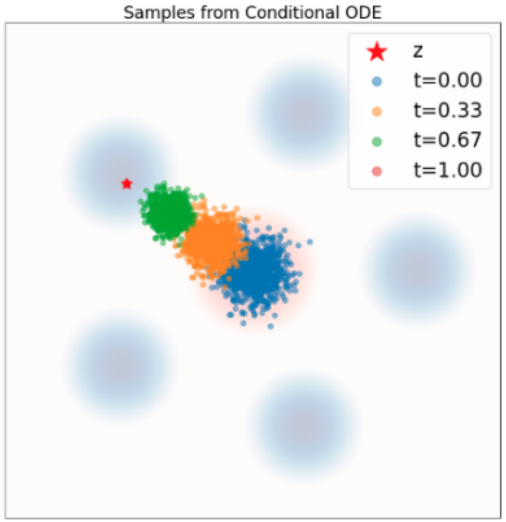
$$X_0 \sim p_{init}, \frac{d}{dt}X_t = u_t^{\text{target}}(X_t) \implies X_t \sim p_t.$$

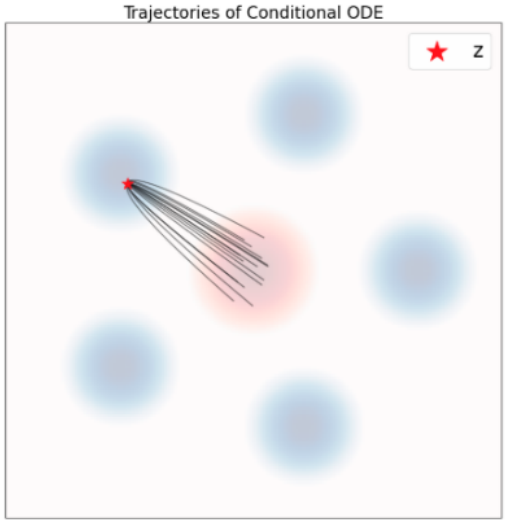
In particular, $X_1 \sim p_{data}$.

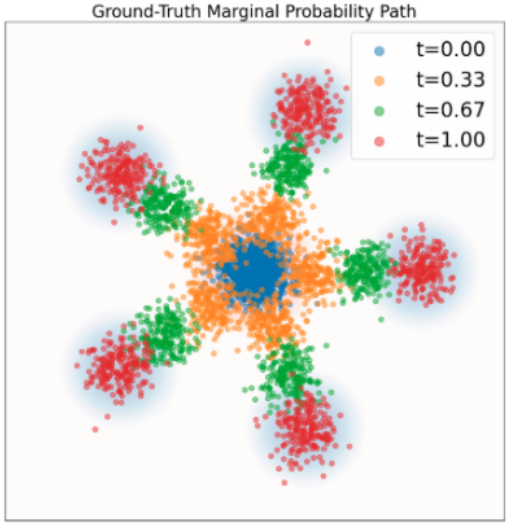
- Thus, u_t^{target} transforms p_{init} to p_{data}

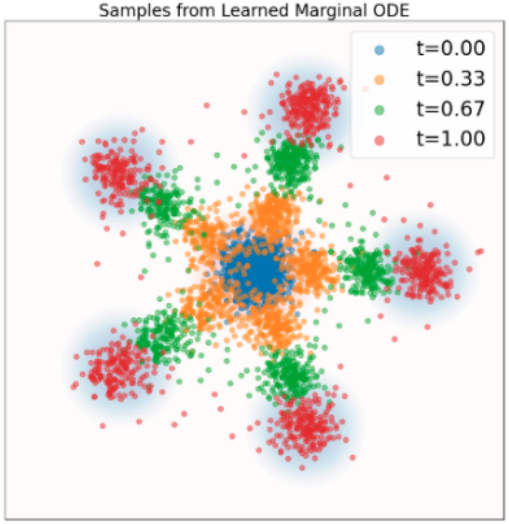
Simulation of probability paths with ODEs

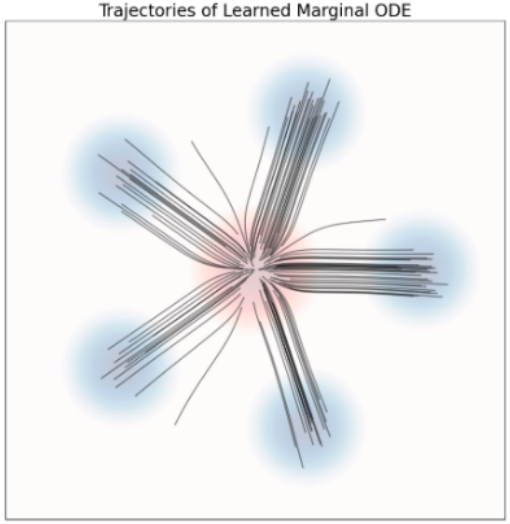












Why the Marginalization Trick is Useful: Gaussian Path Example

- The marginalization trick allows constructing the marginal vector field $u_t^{\text{target}}(x)$ from a simpler conditional one $u_t^{\text{target}}(x|z)$.
- This is often easier because $u_t^{\text{target}}(x|z)$ can be computed analytically.
- **Example:** Consider conditional paths $p_t(\cdot|z) = \mathcal{N}(\alpha_t z, \beta_t^2 I_d)$ with $\dot{\alpha}_t = \partial_t \alpha_t$, $\dot{\beta}_t = \partial_t \beta_t$.
- The conditional Gaussian vector field is:

$$u_t^{\text{target}}(x|z) = \left(\dot{\alpha}_t - \frac{\dot{\beta}_t}{\beta_t}\alpha_t\right) z + \frac{\dot{\beta}_t}{\beta_t} x$$

- This vector field generates ODE trajectories X_t such that $X_t \sim p_t(\cdot|z)$ if $X_0 \sim \mathcal{N}(0, I_d)$.
- We still need to figure out how to make the computation of u_t^{target} tractable

The Continuity Equation

- The continuity equation is a fundamental result that characterizes the evolution of probability densities under a flow.
- The divergence operator is defined as:

$$\operatorname{div}(v_t)(x) = \sum_{i=1}^{d} \frac{\partial}{\partial x_i} v_t(x)_i$$

Continuity Equation:

- Consider a flow model with vector field u_t^{target} and initial distribution $X_0 \sim p_{\text{init}}$.
- Then $X_t \sim p_t$ for all $0 \leq t \leq 1$ **if and only if**:

$$\partial_t p_t(x) = -\operatorname{div}(p_t u_t^{\text{target}})(x) \quad \text{for all } x \in \mathbb{R}^d, 0 \leq t \leq 1$$

where $\partial_t p_t(x) = \frac{d}{dt} p_t(x)$.

- **Intuition:** The change in probability density at point x over time is equal to the net inflow of probability mass at x, governed by the divergence of the density-weighted vector field.

Extending to SDEs: Score Functions

- Define the **conditional score function** as $\nabla \log p_t(x|z)$
- Define the **marginal score function** of p_t as $\nabla \log p_t(x)$. Then,

$$\nabla \log p_t(x) = \int \nabla \log p_t(x|z) \frac{p_t(x|z)p_{data}(z)}{p_t(x)} dz,$$

- **Extending from ODEs to SDEs** Define conditional $u_t^{\text{target}}(x|z)$ and marginal $u_t^{\text{target}}(x)$ vector fields **as before**. For **any** diffusion coefficient $\sigma_t \geq 0$, the SDE:

$$X_0 \sim p_{init}, \quad dX_t = \left(u_t^{\text{target}}(X_t) + \frac{\sigma_t^2}{2} \nabla \log p_t(X_t) \right) dt + \sigma_t dW_t$$

satisfies $X_t \sim p_t$ for $0 \leq t \leq 1$. In particular, $X_1 \sim p_{data}$. The same holds for conditional versions.

- This is useful since marginal score can be expressed via conditional score, which is often known analytically

The Fokker-Planck Equation

- The Fokker-Planck equation generalizes the continuity equation from deterministic (ODE) to stochastic (SDE) settings.

Fokker-Planck Equation:

- Let p_t be a probability density and consider the stochastic differential equation:

$$dX_t = u_t(X_t)dt + \sigma_t dW_t \quad \text{with } X_0 \sim p_{\text{init}}.$$

- Then $X_t \sim p_t$ for all $0 \leq t \leq 1$ if and only if:

$$\partial_t p_t(x) = -\text{div}(p_t u_t)(x) + \frac{\sigma_t^2}{2}\Delta p_t(x) \quad \text{for all } x \in \mathbb{R}^d.$$

- When $\sigma_t = 0$, the Fokker-Planck equation reduces to the continuity equation.
- The Laplacian term Δp_t models the effect of diffusion, similar to how heat spreads over time.

Summary of Deriving the Training Target

Flow Training Target u_t^{target}

1. Choose conditional path $p_t(x|z)$ with $p_0(\cdot|z) = p_{init}, p_1(\cdot|z) = \delta_z$.
2. Find conditional vector field $u_t^{\text{flow}}(x|z)$ whose flow $\psi_t^{\text{target}}(x_0|z)$ yields $X_t \sim p_t(\cdot|z)$ for $X_0 \sim p_{init}$.
3. Marginal vector field: $u_t^{\text{target}}(x) = \int u_t^{\text{flow}}(x|z) \frac{p_t(x|z)p_{data}(z)}{p_t(x)} dz$.
4. This u_t^{target} makes $X_0 \sim p_{init}, dX_t = u_t^{\text{target}}(X_t)dt \implies X_1 \sim p_{data}$.

Extending to SDEs

SDE: $dX_t = \left(u_t^{\text{target}}(X_t) + \frac{\sigma_t^2}{2}\nabla \log p_t(X_t)\right) dt + \sigma_t dW_t \implies X_1 \sim p_{data}$.

Marginal score: $\nabla \log p_t(x) = \int \nabla \log p_t(x|z) \frac{p_t(x|z)p_{data}(z)}{p_t(x)} dz$.

Learning the Target Vector Field

- We aim to train a neural network vector field u_t^θ to approximate the unknown target field u_t^{target}.
- This can be approached via:
  - **Flow Matching** for ODE-based models.
  - **Score Matching** for SDE-based models.
  - A special case of these includes Gaussian paths used in denoising diffusion models.

Flow Matching Loss

Flow Matching Setup:

- Define a flow model governed by:

$$dX_t = u_t^\theta(X_t)\,dt \quad \text{with } X_0 \sim p_{\text{init}}.$$

- The goal is to learn u_t^θ such that $u_t^\theta(x) \approx u_t^{\text{target}}(x)$ for all x and t.

Flow Matching Loss:

$$L_{FM}(\theta) = \mathbb{E}_{t\sim\text{Unif},x\sim p_t}\left[\|u_t^\theta(x) - u_t^{\text{target}}(x)\|^2\right]$$

- Alternatively, if p_t is defined conditionally:

$$L_{FM}(\theta) = \mathbb{E}_{t\sim\text{Unif},z\sim p_{\text{data}},x\sim p_t(\cdot|z)}\left[\|u_t^\theta(x) - u_t^{\text{target}}(x)\|^2\right]$$

- However, computing the marginal $u_t^{\text{target}}(x)$ often involves an intractable integral:

$$u_t^{\text{target}}(x) = \int u_t^{\text{target}}(x|z)\frac{p_t(x|z)p_{\text{data}}(z)}{p_t(x)}\,dz$$

Conditional Flow Matching and Loss Equivalence

- Conditional Flow Matching (CFM) leverages the tractable conditional vector field $u_t^{\text{target}}(x|z)$ instead of the marginal $u_t^{\text{target}}(x)$.
- This is useful because $u_t^{\text{target}}(x|z)$ is often available in closed form, making it feasible to directly supervise the model.
- **Conditional Flow Matching Loss**:

$$L_{CFM}(\theta) = \mathbb{E}_{t\sim\text{Unif},z\sim p_{\text{data}},x\sim p_t(\cdot|z)}\left[\|u_t^\theta(x) - u_t^{\text{target}}(x|z)\|^2\right]$$

- **Loss Equivalence:** The marginal flow matching loss is equivalent to the conditional one up to a constant:

$$L_{FM}(\theta) = L_{CFM}(\theta) + C$$

where C is independent of θ.

- As a result, their gradients are the same:

$$\nabla_\theta L_{FM}(\theta) = \nabla_\theta L_{CFM}(\theta)$$

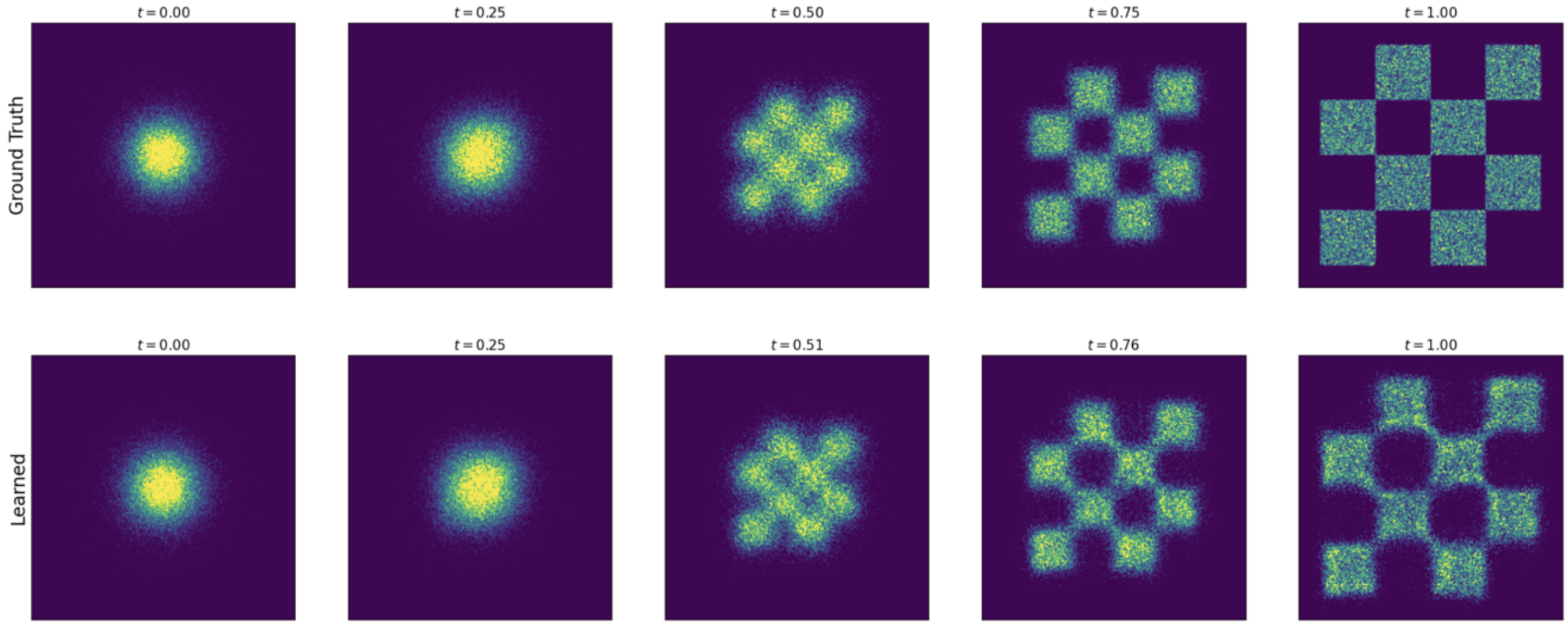
- Therefore, minimizing $L_{CFM}(\theta)$ is equivalent to minimizing $L_{FM}(\theta)$.

Flow Matching Pipeline

- Once the neural vector field u_t^θ is trained (e.g., via minimizing L_{CFM}), sample from the learned distribution by simulating:

$$dX_t = u_t^\theta(X_t)dt, \quad X_0 \sim p_{\text{init}}$$

- This overall training and sampling framework is known as **Flow Matching**.



Example: Flow Matching for Gaussian Conditional Paths

Example: Gaussian Conditional Path

- Conditional path: $p_t(\cdot|z) = \mathcal{N}(\alpha_t z, \beta_t^2 I_d)$.
- Sample: $x_t = \alpha_t z + \beta_t \epsilon$ with $\epsilon \sim \mathcal{N}(0, I_d)$.
- Conditional target vector field:

$$u_t^{\text{target}}(x|z) = \left(\dot{\alpha}_t - \frac{\dot{\beta}_t}{\beta_t}\alpha_t\right) z + \frac{\dot{\beta}_t}{\beta_t} x$$

- Loss becomes:

$$L_{CFM}(\theta) = \mathbb{E}_{t,z,\epsilon}\left[\|u_t^\theta(\alpha_t z + \beta_t \epsilon) - (\dot{\alpha}_t z + \dot{\beta}_t \epsilon)\|^2\right]$$

- Special case: $\alpha_t = t, \beta_t = 1 - t$ leads to:

$$L_{CFM}(\theta) = \mathbb{E}_{t,z,\epsilon}\left[\|u_t^\theta(tz + (1-t)\epsilon) - (z - \epsilon)\|^2\right]$$

Flow Matching Training

Algorithm 3: Flow Matching Training (for Gaussian CondOT path)

Require: Dataset $z \sim p_{data}$, neural network u_t^θ.

1. **for** each mini-batch of data **do**
   1. Sample data z from dataset.
   2. Sample random time $t \sim \text{Unif}[0, 1]$.
   3. Sample noise $\epsilon \sim \mathcal{N}(0, I_d)$.
   4. Set $x = tz + (1 - t)\epsilon$. (General: $x \sim p_t(\cdot|z)$)
   5. Compute loss $L(\theta) = \|u_t^\theta(x) - (z - \epsilon)\|^2$. (General: $\|u_t^\theta(x) - u_t^{\text{target}}(x|z)\|^2$)
   6. Update model parameters θ via gradient descent on $L(\theta)$.
2. **end for**

Score Matching for SDEs

- Target SDE: $dX_t = (u_t^{\text{target}}(X_t) + \frac{\sigma_t^2}{2}\nabla \log p_t(X_t))dt + \sigma_t dW_t$.
- Marginal score: $\nabla \log p_t(x) = \int \nabla \log p_t(x|z) \frac{p_t(x|z)p_{data}(z)}{p_t(x)} dz$.
- Approximate marginal score $\nabla \log p_t$ with a **score network** $s_t^\theta(x)$.
- **Score Matching Loss** (L_{SM}):

$$L_{SM}(\theta) = \mathbb{E}_{t,z,x\sim p_t(\cdot|z)}[\|s_t^\theta(x) - \nabla \log p_t(x)\|^2]$$

- **Conditional Score Matching Loss** (L_{CSM}):

$$L_{CSM}(\theta) = \mathbb{E}_{t,z,x\sim p_t(\cdot|z)}[\|s_t^\theta(x) - \nabla \log p_t(x|z)\|^2]$$

- L_{CSM} is tractable because $\nabla \log p_t(x|z)$ is often known.

Equivalence of Score Matching Losses and Gaussian Example

- The score matching loss equals the conditional score matching loss up to a constant:

$$L_{SM}(\theta) = L_{CSM}(\theta) + C$$

- Therefore, their gradients coincide, and minimizing either gives:

$$\theta^* \Rightarrow s_t^{\theta^*}(x) = \nabla \log p_t(x)$$

- **Gaussian Example:** Assume conditional paths $p_t(x|z) = \mathcal{N}(\alpha_t z, \beta_t^2 I_d)$
- The conditional score is:

$$\nabla \log p_t(x|z) = -\frac{x - \alpha_t z}{\beta_t^2}$$

- The conditional score matching loss becomes:

$$\mathcal{L}_{CSM}(\theta) = \mathbb{E}_{t\sim \text{Unif}, z\sim p_{data}, \epsilon\sim\mathcal{N}(0,I_d)} \left[\frac{1}{\beta_t^2} \left\| \beta_t s_t^{\theta}(\alpha_t z + \beta_t \epsilon) + \epsilon \right\|^2 \right]$$

Diffusion Model Training and Sampling

- After training u_t^θ (via flow matching) and s_t^θ (via score matching), choose a noise scale $\sigma_t \geq 0$ and simulate:

$$X_0 \sim p_{init}, \quad dX_t = \left(u_t^\theta(X_t) + \frac{\sigma_t^2}{2} s_t^\theta(X_t) \right) dt + \sigma_t dW_t$$

  to generate $X_1 \sim p_{data}$.
- In theory, any choice of σ_t works if the models are trained perfectly.
- In practice, errors (e.g., numerical or from imperfect training) mean σ_t must be tuned empirically.
- For Gaussian paths, s_t^θ and u_t^θ can often be outputs of the same network or derived from one another.

Denoising Diffusion Models (DDPMs)

- DDPMs use Gaussian conditionals:

$$p_t(x|z) = \mathcal{N}(\alpha_t z, \beta_t^2 I_d)$$

- The conditional score is:

$$\nabla \log p_t(x|z) = -\frac{x - \alpha_t z}{\beta_t^2}$$

- This gives the conditional score matching loss:

$$\mathcal{L}_{CSM}(\theta) = \mathbb{E}_{t,z,\epsilon}\left[\left\| s_t^\theta(\alpha_t z + \beta_t \epsilon) + \frac{\epsilon}{\beta_t} \right\|^2\right]$$

- Here, s_t^θ learns to predict the noise ϵ — called **denoising score matching**.
- To avoid instability as $\beta_t \to 0$, reparameterize with noise predictor:

$$\epsilon_t^\theta(x) = -\beta_t s_t^\theta(x)$$

- Final DDPM loss:

$$\mathcal{L}_{DDPM}(\theta) = \mathbb{E}_{t,z,\epsilon}\left[\left\| \epsilon_t^\theta(\alpha_t z + \beta_t \epsilon) - \epsilon \right\|^2\right]$$

Training with Noise Prediction (Gaussian Path)

Score Matching Algorithm (Gaussian Path)

Input: Dataset $z \sim p_{data}$, model s_t^θ or noise predictor ϵ_t^θ.

1. For each mini-batch:
   1. Sample data z
   2. Sample $t \sim \text{Unif}[0, 1]$
   3. Sample $\epsilon \sim \mathcal{N}(0, I_d)$
   4. Set $x_t = \alpha_t z + \beta_t \epsilon$
   5. Compute loss:
      - Score matching: $L(\theta) = \left\| s_t^\theta(x_t) + \frac{\epsilon}{\beta_t} \right\|^2$
      - Or: $L(\theta) = \left\| \epsilon_t^\theta(x_t) - \epsilon \right\|^2$
   6. Update θ via gradient descent

Conversion and Sampling for Gaussian Paths

- For Gaussian conditional $p_t(x|z) = \mathcal{N}(\alpha_t z, \beta_t^2 I_d)$, we can express u_t^{target} using $\nabla \log p_t$:

$$u_t^{\text{target}}(x|z) = \left(\beta_t^2 \frac{\dot{\alpha}_t}{\alpha_t} - \beta_t \dot{\beta}_t\right) \nabla \log p_t(x|z) + \frac{\dot{\alpha}_t}{\alpha_t} x$$

$$u_t^{\text{target}}(x) = \left(\beta_t^2 \frac{\dot{\alpha}_t}{\alpha_t} - \beta_t \dot{\beta}_t\right) \nabla \log p_t(x) + \frac{\dot{\alpha}_t}{\alpha_t} x$$

- The marginal version defines the **probability flow ODE**.
- u_t^θ and s_t^θ can be converted into one another.
- No need to train both networks separately.
- Once trained, sampling is done using the SDE:

$$dX_t = \left(\left(\beta_t^2 \frac{\dot{\alpha}_t}{\alpha_t} - \beta_t \dot{\beta}_t + \frac{\sigma_t^2}{2}\right) s_t^\theta(X_t) + \frac{\dot{\alpha}_t}{\alpha_t} X_t\right) dt + \sigma_t dW_t$$

Navigating Diffusion Model Literature

- Many equivalent formulations exist — can be confusing.
- This section overviews key frameworks and differences.

Key Differences:

- Discrete-time vs. Continuous-time
- Forward process vs. Probability paths
- Time-reversal vs. Fokker–Planck approach

Discrete vs. Continuous Time

- Early DDPMs used discrete-time Markov chains ($t = 0, 1, 2, \dots$).
- Discrete formulation is simple but fixes time discretization before training.
- Loss is approximated using ELBO (Evidence Lower Bound).
- Later work showed that this approximates a continuous-time SDE.
- In continuous-time, ELBO becomes tight and sampling errors can be better controlled.
- Despite formulation differences, both models use similar loss functions.

Forward Process vs. Probability Paths

- Early DDPMs used a forward SDE: data z is progressively noised:

$$\bar{X}_0 = z, \quad d\bar{X}_t = u_t^{\text{forw}}(\bar{X}_t)\, dt + \sigma_t^{\text{forw}}\, d\bar{W}_t$$

- This leads to $\bar{X}_T \sim \mathcal{N}(0, I)$ as $t \to \infty$.
- The process defines a family of conditionals $p_t(\cdot|z)$ — known as probability paths.
- For analytical tractability, the forward drift u_t^{forw} is often affine.
- These paths are Gaussian; forward process isn't simulated during training.
- Probability paths (as in Flow Matching) offer a more general and intuitive view.

Time-Reversals vs. Fokker–Planck

- DDPMs originally derived the generative process by time-reversing the forward SDE.
- Time reversal gives another SDE with same path distribution, run backward in time.
- The reversed SDE is:

$$dX_t = \left(-u_t(X_t) + \sigma_t^2 \nabla \log p_t(X_t)\right) dt + \sigma_t \, dW_t$$

- This is a special case of the general probability flow ODE.
- In practice, the final sample X_1 is what matters — full reversal may be unnecessary.
- Alternatively, one can derive the generative process using the Fokker–Planck equation directly.

Flow Matching and Stochastic Interpolants

- Flow Matching (FM) uses ODEs and avoids the need for simulating a forward process.
- Sampling is deterministic — train a vector field u_t^θ via regression to u_t^{target}.
- Stochastic Interpolants (SIs) extend this with an interpolant $I(t, x, z)$ and Langevin noise.
- FM/SIs are more general: work for arbitrary p_{init}, p_{data}, and paths.
- DDPMs are a special case — assume Gaussian paths and noise.

Summary: Training Generative Models

Flow Matching (FM) and Diffusion Models (Score Matching)

Flow Matching: Train u_t^θ with:

$$L_{CFM} = \mathbb{E}_{z,t,x\sim p_t(\cdot|z)} \left[\| u_t^\theta(x) - u_t^{\text{target}}(x|z) \|^2 \right]$$

Sampling: Simulate ODE with u_t^θ.
Score Matching: Train s_t^θ with:

$$L_{CSM} = \mathbb{E}_{z,t,x\sim p_t(\cdot|z)} \left[\| s_t^\theta(x) - \nabla \log p_t(x|z) \|^2 \right]$$

Sampling: Simulate SDE with u_t^θ and s_t^θ.

DDPMs (Gaussian Paths)

Special case where u_t^θ and s_t^θ are interconvertible. Sampling can be done via SDE or ODE. Applies only to Gaussian p_{init} and paths.