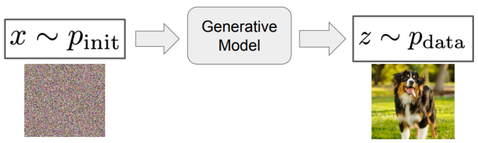
Flow Matching and Diffusion Models

Yunlong Song (宋运龙)

If you want to master something, teach it. — Richard Feynman

Introduction

Generative models like diffusion and flow matching can be seeen as **transporting samples from a simple distribution** (e.g. Gaussian noise) to a complex one (e.g., Images).



To describe this transportation, we can use an **ODE** or **SDE**.

- **ODE** (Ordinary Differential Equation): Deterministic transportation.
- **SDE** (Stochastic Differential Equation): Stochastic transportation.

Ordinary Differential Equation (ODE)

An ODE describes the evolution of a variable over time. It can be written as:

$$\frac{d\mathbf{X}_t}{dt} = u(\mathbf{X}_t) \tag{1}$$

where $\mathbf{X}_t$ is the state variable at time t, and u is a function that defines the rate of change of $\mathbf{X}$.

Stochastic Differential Equation (SDE)

An SDE describes the evolution of a variable over time with an added stochastic component. It can be written as:

$$d\mathbf{X}_t = u(\mathbf{X}_t)dt + g(\mathbf{X}_t)d\mathbf{W}(t) \tag{2}$$

where $\mathbf{W}(t)$ is a Wiener process (or Brownian motion), and g is a function that defines the intensity of the stochastic component. Here, $u(\mathbf{X}_t)$ is the drift term, and $g(\mathbf{X}_t)$ is the diffusion term.

Flow Model

A floew model defines a continuous-time flow that transports samples from a simple distribution to a complex one. It can be described by an ODE:

$$\frac{d\mathbf{X}_t}{dt} = u_t^\theta(\mathbf{X}_t) \tag{3}$$
$$\mathbf{X}_0 \sim p_{init} \tag{4}$$
$$\tag{5}$$

where u_t^θ is a neural network parameterized by θ that defines the velocity field of the flow, and p_{init} is the initial distribution (e.g., Gaussian noise). Our goal is to make the endpoint $\mathbf{x}(1)$ follow the target distribution p_{data}, e.g.

$$\mathbf{X}_1 \sim p_{data} \tag{6}$$

Diffusion Model

A diffusion model defines a continuous-time stochastic process that gradually adds noise to data samples and then learns to reverse this process. It can be described by an SDE:

$$d\mathbf{X}_t = u_t^\theta(\mathbf{X}_t)dt + g_t d\mathbf{W}_t \tag{7}$$
$$\mathbf{X}_0 \sim p_{init} \tag{8}$$
$$\tag{9}$$

where u_t^θ is a neural network parameterized by θ that defines the drift term, g_t is a time-dependent function that defines the diffusion term, and p_{init} is the initial distribution (e.g., Gaussian noise).

Constructing a Training Objectives

The goal of training is to learn the parameters θ such that the model can generate samples from the target distribution p_{data}. This is achieved by minimizing **mean-squared error (MSE)** between the model's predictions and the **training targets**:

$$\mathcal{L}(\theta) = \|u_t^\theta(\mathbf{X}_t) - u_t^{\text{target}}(x)\|^2 \tag{10}$$

where $u_t^{\text{target}}(x)$ is the training target, which depends on the specific model (flow or diffusion).

Probability Paths: Conditional vs. Marginal

Probability paths

how x_t interpolates between noise and data.

Conditional Probability Path

Fix one data point $z \sim p_{\text{data}}$. We want to describe how a noisy sample x_t evolves from pure noise to this specific data point. Formally:

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

- At $t = 0$: $p_0(x \mid z) = \mathcal{N}(0, I)$ (pure noise).
- At $t = 1$: $p_1(x \mid z) = \delta(x - z)$ (all mass at the data point).
- For $0 < t < 1$: a Gaussian "cloud" centered at $\alpha_t z$ with variance β_t^2.

This is the **conditional path**, conditioned on a single data point z. Sampling $x_t \sim p_t(x \mid z)$ gives a point "in between" noise and z.

—

Marginal Probability Path

Now instead of fixing one z, we average over all possible z from the dataset:

$$p_t(x) = \int p_t(x \mid z)\, p_{\text{data}}(z)\, dz.$$

- At $t = 0$: $p_0(x) = p_{\text{init}}(x)$ (e.g. Gaussian noise).
- At $t = 1$: $p_1(x) = p_{\text{data}}(x)$.
- For $0 < t < 1$: a mixture of Gaussians, one for each data point.

This is the **marginal path**: the actual distribution of samples at time t during training, since training averages over the dataset.

—

Toy Example (1D)

Suppose the data distribution consists of two points:

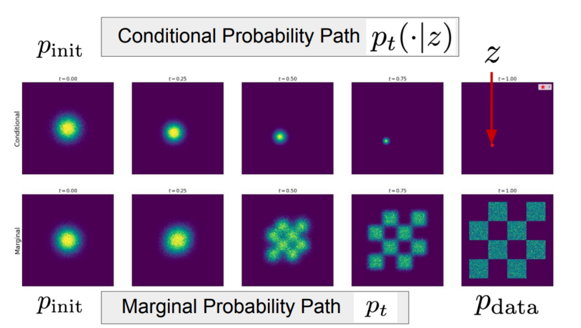
$$z = -2 \quad \text{with probability } 0.5, \qquad z = +2 \quad \text{with probability } 0.5.$$

- **Conditional path:** If $z = +2$, then
$$p_t(x \mid z = +2) = \mathcal{N}(\alpha_t \cdot 2,\, \beta_t^2),$$
a Gaussian drifting toward $+2$.
- **Marginal path:**
$$p_t(x) = 0.5\,\mathcal{N}(\alpha_t \cdot (-2),\, \beta_t^2) \;+\; 0.5\,\mathcal{N}(\alpha_t \cdot 2,\, \beta_t^2),$$
which is a mixture of two Gaussians drifting toward the bimodal data distribution.

—

Summary:

- The **conditional probability path** describes the trajectory toward one chosen data point z.
- The **marginal probability path** describes the average trajectory toward the entire data distribution p_{data}.



Vector Fields: Conditional vs. Marginal

Vetor fields

velocities of these paths (used in *flow matching*).

We now describe the *velocity* of the probability paths. Throughout, assume the conditional path is Gaussian

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

with smooth schedules $\alpha_t, \beta_t > 0$ on $t \in [0,1]$ and time derivatives $\dot{\alpha}_t, \dot{\beta}_t$.

Conditional Vector Field

The conditional vector field specifies the instantaneous rate of change of x_t along the conditional path toward a fixed data point z:

$$u_t^{\text{target}}(x \mid z) = \Big(\dot{\alpha}_t - \frac{\dot{\beta}_t}{\beta_t}\,\alpha_t\Big) z \;+\; \frac{\dot{\beta}_t}{\beta_t}\, x.$$

Interpretation.

- The term proportional to z pulls the trajectory toward the target point.
- The term proportional to x rescales (contracts/expands) the cloud as β_t changes.
- Integrating $\frac{dx}{dt} = u_t^{\text{target}}(x \mid z)$ recovers the conditional path $p_t(x \mid z)$.

Marginal Vector Field

Averaging over $z \sim p_{\text{data}}$ yields the *marginal* velocity:

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

$$\text{where} \quad p_t(x) = \int p_t(x \mid z)\, p_{\text{data}}(z)\, dz.$$

Interpretation.

- $u_t^{\text{target}}(x)$ is the *expected* velocity at location x and time t, weighted by how likely each z is to have produced x under $p_t(x \mid z)$.
- This is the quantity learned in **flow matching**: a neural network $u_\theta(x,t)$ is trained to approximate $u_t^{\text{target}}(x)$.
- Sampling after training solves the ODE $\frac{dx}{dt} = u_\theta(x,t)$ from $t=0$ to $t=1$.

Summary

- **Conditional vector field** $u_t^{\text{target}}(x \mid z)$: velocity toward a *fixed* data point z that reproduces the conditional Gaussian path.
- **Marginal vector field** $u_t^{\text{target}}(x)$: expectation of the conditional velocity under the posterior over z given x at time t; this is the training target in flow matching.

Score Functions: Conditional vs. Marginal

Score functions

gradients of log-density (used in *diffusion models*).

Conditional Score Function

For the Gaussian conditional path

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

the score is the gradient of the log-density:

$$s(x,t \mid z) = \nabla_x \log p_t(x \mid z) = -\frac{1}{\beta_t^2}\,(x - \alpha_t z).$$

Interpretation.

- The score points back toward the mean $\alpha_t z$.
- The magnitude grows as $\|x - \alpha_t z\|$ increases.
- This tells us how to "denoise" a sample x at time t given the true data point z.

Marginal Score Function

The marginal score is defined as

$$s(x,t) = \nabla_x \log p_t(x),$$

where

$$p_t(x) = \int p_t(x \mid z)\, p_{\text{data}}(z)\, dz.$$

Expanding:

$$s(x,t) = \int s(x,t \mid z)\, \frac{p_t(x \mid z)\, p_{\text{data}}(z)}{p_t(x)}\, dz.$$

Interpretation.

- $s(x,t)$ is the *expected conditional score*, averaged over all possible z consistent with x.
- This is the quantity estimated in **diffusion models** via score matching: a neural network $s_\theta(x,t)$ learns to approximate $s(x,t)$.

Summary

- **Conditional score** $s(x,t \mid z)$: denoising force toward a known target z.
- **Marginal score** $s(x,t)$: average denoising force over all possible targets, used as the training signal in diffusion models.

Training the Generative Model

With the probability paths, vector fields, and score functions defined, we now describe how to train a neural network to approximate them.

Flow Matching: Velocity Regression

The network $u_\theta(x,t)$ aims to approximate the marginal velocity $u(x,t)$. From the Gaussian path, the conditional velocity is

$$u_t^{\text{target}}(x \mid z) = \Big(\dot{\alpha}_t - \frac{\dot{\beta}_t}{\beta_t}\alpha_t\Big) z + \frac{\dot{\beta}_t}{\beta_t} x.$$

The training objective is:

$$\mathcal{L}_{\text{flow}}(\theta) = \mathbb{E}_{z \sim p_{\text{data}},\, t \sim \mathcal{U}[0,1],\, x \sim p_t(x|z)} \left[\|u_\theta(x,t) - u_t^{\text{target}}(x \mid z)\|^2 \right].$$

Algorithm 3 Flow Matching Training Procedure (here for Gaussian CondOT path $p_t(x|z) = \mathcal{N}(tz, (1-t)^2)$)

Require: A dataset of samples $z \sim p_{\text{data}}$, neural network u_t^θ

for each mini-batch of data **do**
- Sample a data example z from the dataset.
- Sample a random time $t \sim \text{Unif}_{[0,1]}$.
- Sample noise $\epsilon \sim \mathcal{N}(0, I_d)$
- Set $x = tz + (1-t)\epsilon$ (General case: $x \sim p_t(\cdot|z)$)
- Compute loss

$$\mathcal{L}(\theta) = \|u_t^\theta(x) - (z - \epsilon)\|^2 \qquad (\text{General case: } = \|u_t^\theta(x) - u_t^{\text{target}}(x|z)\|^2)$$

- Update the model parameters θ via gradient descent on $\mathcal{L}(\theta)$.

end for

Diffusion Models: Score Matching

The network $s_\theta(x,t)$ aims to approximate the marginal score $s(x,t) = \nabla_x \log p_t(x)$. Since $p_t(x)$ is intractable, we use the conditional Gaussian score:

$$s(x,t \mid z) = -\frac{1}{\beta_t^2}(x - \alpha_t z).$$

The training objective is:

$$\mathcal{L}_{\text{score}}(\theta) = \mathbb{E}_{z \sim p_{\text{data}},\, t \sim \mathcal{U}[0,1],\, x \sim p_t(x|z)} \left[\|s_\theta(x,t) - s(x,t \mid z)\|^2 \right].$$

Algorithm 4 Score Matching Training Procedure for Gaussian probability path

Require: A dataset of samples $z \sim p_{\text{data}}$, score network s_t^θ or noise predictor ϵ_t^θ

for each mini-batch of data **do**
- Sample a data example z from the dataset.
- Sample a random time $t \sim \text{Unif}_{[0,1]}$.
- Sample noise $\epsilon \sim \mathcal{N}(0, I_d)$
- Set $x_t = \alpha_t z + \beta_t \epsilon$ (General case: $x_t \sim p_t(\cdot|z)$)
- Compute loss

$$\mathcal{L}(\theta) = \|s_t^\theta(x_t) + \frac{\epsilon}{\beta_t}\|^2 \qquad (\text{General case: } = \|s_t^\theta(x_t) - \nabla \log p_t(x_t|z)\|^2)$$

$$\text{Alternatively: } \mathcal{L}(\theta) = \|\epsilon_t^\theta(x_t) - \epsilon\|^2$$

- Update the model parameters θ via gradient descent on $\mathcal{L}(\theta)$.

end for