Flow Matching and Diffusion Models

Li Ju
li.ju@it.uu.se

July 8, 2025

A very broad problem of interest is that, given $z \sim P_{\text{data}}$, with samples $\{z_i\}_{i=1}^N$ from P_{data}, how to sample more data from P_{data}.

1 Basics of differential equations

In this section, we familiarise ourselves with a specific type of ordinary differential equations (ODE) and stochastic differential equations (SDE), which provide an alternative perspective for us to derive and understand flow matching and diffusion models.

1.1 ODE

Vector field First, we define a type of function called time-varying vector field $u : \mathbb{R}^d \times [0,1] \to \mathbb{R}^d, (x,t) \mapsto u_t(x)$, where d is the dimension of the vector field, x denotes the location of the field, t denotes the time and $u_t(x)$ is the direction of the field at location x at time t.

ODE Now we define the specific type of the ODE we are interested in:

$$\frac{d}{dt}X_t = u_t(X_t),\ X_0 = x_0 \tag{1}$$

where $X : [0,1] \to \mathbb{R}^d, t \mapsto X_t$ is a *trajectory* (of a particle), which describe its movement along time t. The ODE describe a trajectory such that at any time t, the velocity of X at X_t is following $u_t(X_t)$.

Given a fixed vector field u_t and an initial condition x_0, under certain regularity conditions, there exists its solution trajectory X_t.

Flow For a given vector field u_t, the initial condition x_0 determines the solution trajectory. For different initial conditions there exist different trajectories. The collection of all these trajectories with infinite starting points form a *flow*, described by $\psi : \mathbb{R}^d \times [0,1] \to \mathbb{R}^d, (x_0,t) \mapsto \psi_t(x_0)$. This could be understood from a perspective of higher order function: given an initial condition x_0, partially applied function $\psi(x_0)$ returns a trajectory function, which takes time t and return location $\psi_t(x_0)$.

In summary, given a fixed *vector field*, we can define an *ODE*, which has infinitely many solution *trajectories* depending on their initial conditions. The function taking an initial condition and returning a solution trajectory is called *flow*.

Simulations In general, given a vector field (ODE) u, it is not possible to analytically get its flow function. Instead using different initial conditions we can obtain arbitrarily many trajectories to simulate the flow using the following discretized form:

$$\begin{aligned} X_0 &= x_0, \\ X_{t+h} &= X_t + h * u_t(X_t), \end{aligned} \tag{2}$$

where $t = h, 2h, 3h, \dots 1-h, h = 1/n$ and $n > 1$.

Flow model Essentially, a flow describes how a collection of data is transformed into another collection of data in $\mathbb{R}^d$, which is characterised by a vector field u via an ODE. Likewise, such an ODE can also convert/transform a distribution P_0 into another distribution P_1:

$$\begin{aligned} &X_0 \sim P_0,\ \frac{d}{dt}X_t = u_t(X_t) \\ \Longrightarrow &X_1 \sim P_1. \end{aligned} \tag{3}$$

Data generation is nothing but to sample from an unknown data distribution P_{data}. If we can find a vector field to characterise an ODE which converts a simple predefined distribution P_{init} to the data distribution P_{data}, then with the simulation method provided by Equation 2 we can generate new data by sampling from P_{data}.

To find such a vector field, we generally use a parameterised neural network u^θ to do so. Counter-intuitively, flow model does *not* model the flow using neural network but model the vector field. The flow can be obtained by sampling trajectories using the vector field:

Algorithm 1 Euler's method for sampling with a flow method

```
Require: Initial distribution $P_{\text{init}}$, vector field $u^\theta$, total steps $T$
  $h = 1/T, t = 0$
  Draw a sample $X_0 \sim P_{\text{init}}$ as $x_0$
  for $i = 1, 2, \ldots, T$ do
      $x_{t+h} = x_t + hu_t^\theta(x_t)$
      Update $x_t = x_{t+h}$ and $t = t + h$
  end for
  Return $x_t$
```

1.2 SDE

For ODE method, for a certain vector field, given a fixed initial point, the solution trajectory is deterministic with a fixed sample. It is possible to extend this into stochastic with stochastic trajectories via stochastic differential equations.

Stochastic trajectory A stochastic trajectory X is denoted by $(X_t)_{t\in[0,1]}$, characterised by

- $\forall t \in [0,1], X_t$ is a random variable (instead of a deterministic point),
- any sample from $(X_t)_{t\in[0,1]}$ is a trajectory $[0,1] \to \mathbb{R}^d$.

Brownian motion SDE are generally constructed by a simplest stochastic process, Brownian motion $(W_0)_{t\geq 0}$, with following conditions:

- $W_0 = 0$,
- **Normal increments:** $W_t - W_s \sim \mathcal{N}(0, (t-s)I_d)$ for all $0 \leq s < t$, where d denotes the dimension,
- **Independent increments:** For any $0 \leq t_0 < t_1 < \ldots t_n$, the increments $W_1 - W_0, \ldots W_n - W_{n-1}$ are independent random variables.

To simulate Brownian motions, one can simply use $W_{t+h} = W_t + \sqrt{h}\epsilon$ where $\epsilon \sim \mathcal{N}(0, I_d)$ and $t = 0, h, 2h, \ldots$.

From ODE to SED An ODE is defined by the derivative of the solutions. However for SDE making use of Brownian motions, it is impossible since each single step is random, which could not be taken derivatives. Instead, an SDE is formally defined by its integral form but here we informally define an SDE via its increments:

$$dX_t = u_t(X_t)dt + \sigma_t dW_t,\ X_0 = x_0 \tag{4}$$

With $\sigma_t = 0$, we can see that ODE is a special case of SDE (or SDE is an extension of ODE, reversely).

Simulation For SDE, it is even harder (if even possible) to obtain the flow function. But there exists an easy way to simulate it using the following discretized form:

$$\begin{aligned} X_0 &= x_0, \\ X_{t+h} &= X_t + h * u_t(X_t) + \sqrt{h}\epsilon, \quad \epsilon \sim \mathcal{N}(0, I_d), \end{aligned} \tag{5}$$

where $t = h, 2h, 3h, \ldots 1-h, h = 1/n$ and $n > 1$.

Diffusion model An SDE with a fixed vector field u and a predefined diffusion coefficient σ can also convert a collection of data into another collection. Similarly, we also want to find an SDE which can convert an initial distribution P_{init} to the distribution we are interested in P_{data}, characterised by a vector field u and a diffusion coefficient σ as follows:

$$\begin{aligned} & X_0 \sim P_{\text{init}},\ \frac{d}{dt}X_t = u_t(X_t) + \sigma_t dW_t \\ \textbf{target:}\quad & X_1 \sim P_{\text{data}}. \end{aligned} \tag{6}$$

This is called a diffusion model. In general, the diffusion coefficient $\sigma : [0,1] \to \mathbb{R}, t \mapsto \sigma_t$ is predefined and we use a parameterised neural network $u^\theta : \mathbb{R}^d \times [0,1] \to \mathbb{R}^d, (x,t) \mapsto u^\theta_t(x)$ to approximate u.

With the neural network vector field u^θ, we can generate new data via sampling using the following algorithm:

Algorithm 2 Euler-Maruyama's method for sampling from a diffusion model

```
Require:
Require: Initial distribution $P_{\text{init}}$, vector field $u^\theta$, diffusion coefficient $\sigma_t$, total steps $N$
  $h = 1/N, t = 0$
  Draw a sample from $P_{\text{init}}$ as $x_0$.
  for $i = 1, 2, \ldots, T$ do
      Draw a sample $\epsilon_t$ from a standard Gaussian $\mathcal{N}(0, I_d)$
      $x_{t+h} = x_t + hu^\theta_t(x_t) + \sqrt{h}\epsilon_t$
      Update $x_t = x_{t+h}$ and $t = t + h$.
  end for
  Return $x_t$
```

2 Constructing the training target

In this section we will introduce the construction of the training target and we will show how to approximate it with a neural network.

2.1 Conditional and marginal probability path

Recall that using a certain vector field u, we can convert data from a predefined distribution P_0 to our target distribution P_1. This can be seen as a continuous path of probability distributions $P_t, t \in [0,1]$. We are aiming to convert $P_0 = P_{\text{init}}$ to $P_1 = P_{\text{data}}$.

2.1.1 Conditional probability path

First we define *Dirac distribution* Δ_z where $z \in \mathbb{R}^d$, characterised by its probability density function (PDF)

$$\delta_z(x) = \begin{cases} 1 & \text{if } x = z \\ 0 & \text{otherwise} \end{cases}.$$

Then a *conditional probability path* is a set of time-dependent probability distributions $P_t(\cdot \mid z)$ over $\mathbb{R}^d$ such that

$$P_0(\cdot \mid z) = P_{\text{init}} \quad \text{and} \quad P_1(\cdot \mid z) = \Delta_z.$$

A probability path can be seen as a trajectory path in the probability space and can be characterised by its time dependent PDF $p_t(\cdot \mid z)$,

2.1.2 Marginal probability path

For each conditional probability path $P_t(\cdot \mid z)$, if there exists an associated *marginal probability path* P_t over $\mathbb{R}^d$ if we integrate z out over P_{data}:

$$\begin{aligned} &z \sim P_{\text{data}},\ x \mid z \sim P_t(\cdot \mid z) \\ &\Longrightarrow x \sim P_t,\ p_t(x) = \int p_t(x \mid z) p_{\text{data}}(z) dz \end{aligned}$$

where $p_t(\cdot)$ is the PDF of the marginal probability path P_t.

We can see that

$$\begin{aligned} p_0(x) &= \int p_{\text{init}}(x) p_{\text{data}}(z) dz = p_{\text{init}} \int (x) p_{\text{data}}(z) dz = p_{\text{init}}(x) \qquad \Longrightarrow P_0 \sim P_{\text{init}} \\ p_1(x) &= \int \delta_z(x) p_{\text{data}}(z) dz = p_{\text{data}}(x) \qquad \Longrightarrow P_1 \sim P_{\text{data}}. \end{aligned}$$

In summary, *conditional probability path* describe the trajectory between P_{init} to a specific data point $z \sim P_{\text{data}}$, while *marginal probability path* is the trajectory between P_{init} to P_{data} in the probability space.

Gaussian example Let's construct a gaussian example for the conditional probability path and derive its associated marginal probability path.

We define that

$$P_t(\cdot \mid z) \sim \mathcal{N}(\alpha_t z, \beta_t^2 I) \quad \text{such that} \quad \alpha_0 = 0, \alpha_1 = 1, \beta_0 = 1, \beta_1 = 0.$$

This gives that

$$P_{\text{init}} = P_0(\cdot \mid z) \sim \mathcal{N}(0, I) \quad \text{and} \quad P_1(\cdot \mid z) = \Delta_z.$$

There exist an associated marginal probability path P_t. Though we do not have access to its PDF directly, we can draw a sample x from P_t with

$$z \sim P_{\text{data}}, \epsilon \sim \mathcal{N}(0, I) \Longrightarrow x = \alpha_t z + \beta_t \epsilon.$$

2.2 Conditional and marginal vector field

Now that we can construct a marginal probability path to convert P_{init} to P_{data}, we are interested in the *marginal vector field* u^{target} which can make the actual conversion.

2.2.1 Tools

Before our derivation, we start with a definition of *divergence* operator div and *continuity equation.*

Divergence div of a vector field $v_t : (\mathbb{R} \times t) \to \mathbb{R}^d$ is defined as follows:

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

Continuity equation Consider a flow model with vector field v_t and $X_0 \sim P_0$. Then $X_t \sim P_t$ for all $t \in [0,1]$ if and only if

$$\frac{dp_t(x)}{dt} = -\operatorname{div}(p_t v_t)(x) \quad \forall x \in \mathbb{R}^d, t \in [0,1],$$

where p_t denote the PDF of P_t.

2.2.2 Conditional vector field

Before we look at the marginal vector field, we check the *conditional vector field* $u^{\text{target}}(\cdot \mid z)$, which converts P_{init} to Δ_z. Conditional vector field is generally tractable for a given probability path. Specifically, it is derived with continuity equation. We here use the gaussian probability path as an example.

Gaussian example Recall that the gaussian probability path is defined as

$$P_t(\cdot \mid z) \sim \mathcal{N}(\alpha_t z, \beta_t^2 I).$$

For any position x at t, we have

$$x = \alpha_t z + \beta_t \epsilon, \ \epsilon \sim \mathcal{N}(0, I_d).$$

The vector field $u_t^{\text{target}}(\cdot \mid z)$ is then given by

$$\begin{aligned}
u_t^{\text{target}}(x \mid z) &= \frac{\partial x}{\partial t}(x,t,z) = \dot{\alpha}_t z + \dot{\beta}_t \epsilon \\
&= \frac{\partial x}{\partial t}(x,t,z) = \dot{\alpha}_t z + \dot{\beta}_t \frac{x - \alpha_t z}{\beta_t} \qquad (\text{Given by } \epsilon = \frac{x - \alpha_t z}{\beta_t}) \\
&= \left(\dot{\alpha}_t - \alpha_t \frac{\dot{\beta}_t}{\beta_t}\right) z + \frac{\dot{\beta}_t}{\beta_t} x
\end{aligned}$$

2.2.3 Marginal vector field

Although marginal vector field is the one of our interest, it is very hard to derive it directly. Instead, we derive the conditional ones first and then use them to build the marginal vector field.

Assume that we have data $z \sim P_{\text{data}}$ and conditional vector field $u_t^{\text{target}}(\cdot \mid z)$, the marginal vector field u_t^{target} is given by

$$u_t^{\text{target}}(x) = \int u_t^{\text{target}}(x \mid z) \frac{p_t(x \mid z) p_{\text{data}}(z)}{p_t(x)} dz, \tag{7}$$

which follows the marginal probability path P_t.

Proof. To prove it, we need to make use of the continuity equation:

$$\begin{aligned}
\frac{d}{dt} p_t(x) &= \frac{d}{dt} \int p_t(x \mid z) p_{\text{data}}(z) dz \\
&= \int \frac{d}{dt} p_t(x \mid z) p_{\text{data}}(z) dz \quad (\text{under some regularity conditions}) \\
&= \int \operatorname{div}(p_t(\cdot \mid z) u_t^{\text{target}}(\cdot \mid z))(x) p_{\text{data}}(z) dz \quad (\text{continuity equation}) \\
&= \int \operatorname{div}\left(p_t(\cdot) p_t(\cdot \mid z) \frac{p_{\text{data}}(z)}{p_t(\cdot)} u_t^{\text{target}}(\cdot \mid z)\right)(x) dz \\
&= \operatorname{div}\left(p_t(\cdot) \underbrace{\int \frac{p_t(\cdot \mid z) p_{\text{data}}(z)}{p_t(\cdot)} u_t^{\text{target}}(\cdot \mid z) dz}_{u_t^{\text{target}}}\right)(x). \quad (\text{swap operators})
\end{aligned}$$

Again, using continuity equation, we have

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

□

2.3 Extending to SDE

2.3.1 Tools

Fokker-Planck equation Let P_t be a probability path and consider an SDE

$$X_0 \sim P_{\text{init}}, dX_t = u_t(X_t)dt + \sigma_t dW_t.$$

Then X_t has a distribution path of P_t characterised by its PDF p_t if and only if

$$\frac{\partial}{\partial t} p_t(x) = -\operatorname{div}(p_t u_t)(x) + \frac{\sigma_t^2}{2} \Delta p_t(x), \forall x \in \mathbb{R}^d, t \in [0,1],$$

where Δ denotes the Laplacian operator $\Delta w(x) = \sum_{i=1}^d \frac{\partial^2 w(x)}{\partial^2 x_i} = \operatorname{div}(\nabla w)(x)$.

Score function Term $\nabla_x \log p(x)$ is generally called *score function.* Thus, we have $\nabla_x \log p_t(x \mid z)$ as the *conditional score function* and $\nabla_x \log p_t(x)$ *marginal score function* in our context.

2.3.2 Main results

Now we have known how to construct a vector field to convert P_{init} to P_{data} via ODE. Alternatively, we can also do this conversion via an SDE:

$$\begin{aligned} X_0 \sim P_{\text{init}},\ dX &= \left(u_t^{\text{target}}(x) + \frac{\sigma_t^2}{2} \nabla_x \log p_t(x)\right) dt + \sigma_t dW, \\ \Longrightarrow X &\sim P_t, t \in [0,1]. \end{aligned} \tag{8}$$

If we replace all marginal entities to conditional ones the conversion still holds.

We know everything in this SDE but the score function $\nabla_x \log p_t(x)$, which can be derived as follows:

$$\begin{aligned} \nabla_x \log p_t(x) &= \frac{\nabla_x p_t(x)}{p_t(x)} \\ &= \frac{\nabla_x \int p_t(x \mid z) p_{\text{data}}(z) dz}{p_t(x)} \\ &= \frac{\int \nabla_x p_t(x \mid z) p_{\text{data}}(z) dz}{p_t(x)} \\ &= \int \frac{p_t(x \mid z) p_{\text{data}}(z)}{p_t(x)} \nabla_x \log p_t(x \mid z) dz. \end{aligned} \tag{9}$$

we can see that the marginal score function is also the weighted average of the conditional score function, with the same weights as those of the vector fields.

Now we prove the main results.

Proof.

$$\begin{aligned}
\frac{\partial p_t(x)}{\partial t} &= -\operatorname{div}(p_t u_t^{\text{target}})(x) \\
&= -\operatorname{div}(p_t u_t^{\text{target}})(x) - \frac{\sigma_t^2}{2}\Delta p_t(x) + \frac{\sigma_t^2}{2}\Delta p_t(x) \\
&= -\operatorname{div}(p_t u_t^{\text{target}})(x) - \frac{\sigma_t^2}{2}\operatorname{div}(\nabla p_t)(x) + \frac{\sigma_t^2}{2}\Delta p_t(x) \\
&= -\operatorname{div}(p_t u_t^{\text{target}})(x) - \frac{\sigma_t^2}{2}\operatorname{div}(p_t \nabla \log p_t)(x) + \frac{\sigma_t^2}{2}\Delta p_t(x) \\
&= -\operatorname{div}\left(p_t\left(u_t^{\text{target}} + \frac{\sigma_t^2}{2}\nabla \log p_t\right)\right)(x) + \frac{\sigma_t^2}{2}\Delta p_t(x).
\end{aligned}$$

Thus, the vector field that can make the conversion is given by $u_t^{\text{target}} + \frac{\sigma_t^2}{2}\nabla \log p_t$. □

Remark If the probability path is static, i.e. $P_t = P$, we have $u_t^{\text{target}} = 0$ and this is called *Langevin dynamics*. Using Euler-Maruyama's method, one can sample from a static distribution P.

Gaussian example The conditional gaussian score function $\nabla_x \log p_t(x \mid z)$ is given by

$$\begin{aligned}
\nabla_x \log p_t(x \mid z) &= \nabla_x \left(-\frac{1}{2}\beta_t^{-2}(x - \alpha_t z)^\top (x - \alpha_t z)\right) \\
&= -\beta_t^{-2}(x - \alpha_t z)
\end{aligned}$$

3 Deriving loss functions

In this section, we derive the loss functions for the training. We start from the flow model, and then we extend it to diffusion model, and then we recover the classic denoising diffusion model with other notes.

3.1 Loss for flow model

u_t^{target} is the vector field we are interested in which can convert P_{init} to P_{data}. We want to use a parameterised neural network u_t^θ to approximate it. A straightforward approach is to minimise the ℓ_2 norm of the two functions:

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{t\sim\mathcal{U}(0,1), x\sim P_t}\left[\|u_t^\theta(x) - u_t^{\text{target}}(x)\|^2\right], \tag{10}$$

where $\mathcal{U}(0,1)$ denotes a uniform distribution over 0 and 1.

From Equation 7, we know that

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

However, the target is intractable: 1). the integration is hard to solve, 2). we do not know p_{data} and $p_t(x)$.

Another way around is that we use the following loss

$$\mathcal{L}_{\text{CFM}} = \mathbb{E}_{t\sim\mathcal{U}(0,1), x\sim P_t(\cdot|z), z\sim P_{\text{data}}}\left[\|u_t^\theta(x) - u_t^{\text{target}}(x \mid z)\|^2\right]. \tag{11}$$

We claim that $\mathcal{L}_{\text{FM}} = \mathcal{L}_{\text{CFM}} + C$, where C is a constant.

Proof.

$$
\begin{aligned}
\mathcal{L}_{\mathrm{FM}} &= \mathbb{E}_{t\sim\mathcal{U}(0,1),x\sim P_t(\cdot|z),z\sim P_{\mathrm{data}}}\left[\|u_t^\theta(x) - u_t^{\mathrm{target}}(x\mid z)\|^2\right] \\
&= \mathbb{E}_{t\sim\mathcal{U}(0,1),x\sim P_t(\cdot|z),z\sim P_{\mathrm{data}}}\left[\|u_t^\theta(x)\|^2 - 2u_t^\theta(x)^\top u_t^{\mathrm{target}}(x\mid z) + \underbrace{\|u_t^{\mathrm{target}}(x\mid z)\|^2}_{\text{const.}}\right] \\
&= \mathbb{E}_{t\sim\mathcal{U}(0,1),x\sim P_t}\left[\|u_t^\theta(x)\|^2\right] - 2\mathbb{E}_{t\sim\mathcal{U}(0,1),x\sim P_t(\cdot|z),z\sim P_{\mathrm{data}}}\left[u_t^\theta(x)^\top u_t^{\mathrm{target}}(x\mid z)\right] + C_1 \\
&= \mathbb{E}_{t\sim\mathcal{U}(0,1),x\sim P_t}\left[\|u_t^\theta(x)\|^2\right] - 2\int_0^1\int\int u_t^\theta(x)^\top u_t^{\mathrm{target}}(x\mid z)p_t(x\mid z)p_{\mathrm{data}}(z)dxdzdt + C_1 \\
&= \mathbb{E}_{t\sim\mathcal{U}(0,1),x\sim P_t}\left[\|u_t^\theta(x)\|^2\right] - 2\int_0^1\int\int u_t^\theta(x)^\top p_t(x) u_t^{\mathrm{target}}(x\mid z)\frac{p_t(x\mid z)p_{\mathrm{data}}(z)}{p_t(x)}dxdzdt + C_1 \\
&= \mathbb{E}_{t\sim\mathcal{U}(0,1),x\sim P_t}\left[\|u_t^\theta(x)\|^2\right] - 2\int_0^1\int u_t^\theta(x)^\top p_t(x) u_t^{\mathrm{target}}(x)dxdt + C_1 \quad \text{(use Equation 7)} \\
&= \mathbb{E}_{t\sim\mathcal{U}(0,1),x\sim P_t}\left[\|u_t^\theta(x)\|^2\right] - 2\mathbb{E}_{t\sim\mathcal{U}(0,1),x\sim P_t(\cdot|z),z\sim P_{\mathrm{data}}}\left[u_t^\theta(x)^\top p_t(x) u_t^{\mathrm{target}}(x)\right] + C_1 \\
&= \mathbb{E}_{t\sim\mathcal{U}(0,1),x\sim P_t}\left[\|u_t^\theta(x) - u_t^{\mathrm{target}}(x)\|^2\right] + C_2
\end{aligned}
$$

□

Now we have all the ingredients we have for the training of the target vector field.

Loss for the gaussian example Recall the gaussian example we have

$$P_t(\cdot\mid z) \sim \mathcal{N}(\alpha_t z, \beta_t^2 I),\ \text{ where } \alpha_0 = 0, \alpha_1 = 1, \beta_0 = 1, \beta_1 = 0,$$

which has its conditional vector field (as shown in the last section)

$$u^{\mathrm{target}}(x\mid z) = \left(\dot{\alpha}_t - \alpha_t\frac{\dot{\beta}_t}{\beta_t}\right)z + \frac{\dot{\beta}_t}{\beta_t}x.$$

We know that $x \sim P_t(\cdot\mid z)$ can be rewritten as $\beta_t\epsilon + \alpha_t z$, where $\epsilon \sim \mathcal{N}(0, I)$, the loss can be written as

$$
\begin{aligned}
\mathcal{L}_{\mathrm{CFM}}^{\mathcal{G}}(\theta) &= \mathbb{E}_{t\sim\mathcal{U}(0,1),\epsilon\sim\mathcal{N}(0,I),z\sim P_{\mathrm{data}}}\left[\|u^\theta(\alpha_t z + \beta_t\epsilon) - \left(\dot{\alpha}_t - \alpha_t\frac{\dot{\beta}_t}{\beta_t}\right)z - \frac{\dot{\beta}_t}{\beta_t}(\beta_t\epsilon + \alpha_t)\|^2\right] \\
&= \mathbb{E}_{t\sim\mathcal{U}(0,1),\epsilon\sim\mathcal{N}(0,I),z\sim P_{\mathrm{data}}}\left[\|u^\theta(\alpha_t z + \beta_t\epsilon) - (\dot{\alpha}_t z + \dot{\beta}_t\epsilon)\|^2\right]
\end{aligned}
\tag{12}
$$

If we have $\alpha_t = t$ and $\beta_t = 1 - t$, which gives $\dot{\alpha}_t = 1$ and $\dot{\alpha}_t = -1$, the method is called *condOT* probability path and the loss is given by

$$\mathcal{L}_{\mathrm{CFM}}^{\mathrm{condOT}}(\theta) = \mathbb{E}_{t\sim\mathcal{U}(0,1),\epsilon\sim\mathcal{N}(0,I),z\sim P_{\mathrm{data}}}\left[\|u^\theta(tz + (1-t)\epsilon) - (z - \epsilon)\|^2\right].$$

The algorithm obtained is given by Algorithm 0.

Then with u^θ we can apply Euler's method to sample from P_{data} by simulating the ODE.

3.2 Loss for diffusion model

With the trained u^θ, alternatively we can also sample from P_{data} by simulating the SDE in Equation

$$dX_t = (u^\theta + \frac{\sigma_t^2}{2}\nabla_x \log p_t(x)) + \sigma_t dW_t,\ X_0 \sim P_{\mathrm{init}},$$

where σ_t^2 is a predefined hyper-parameter.

Algorithm 3 Training algorithm for flow methods

```
Require: Dataset $\{z_n\}_{n=1}^N$
  Initialize model $u^\theta$
  for $i = 1, 2, \dots, T$ do
      Draw a sample $z$ from $\{z_n\}_{i=1}^N$.
      Draw a sample $\epsilon \sim \mathcal{N}(0, I)$.
      Draw a sample $t \sim \mathcal{U}(0, 1)$.
      Compute condOT loss $\mathcal{L}(\theta) = \|u^\theta(tz + (1-t)\epsilon) - (z - \epsilon)\|^2$ (or general $\|u^\theta(\alpha_t z + \beta_t \epsilon) - (\dot{\alpha}_t z + \dot{\beta}_t \epsilon)\|^2$).
      Back propagate to update model parameters $\theta$.
  end for
  Return $\theta$
```

But we do not know the score function $\nabla_x \log p_t(x))$ yet and we need to learn it. The loss we are interested in is

$$\mathcal{L}_{\mathrm{DM}} = \mathbb{E}_{t\sim\mathcal{U}(0,1), x\sim P_t}\left[\|s_t^\theta(x) - \nabla_x \log p_t(x))\|^2\right]. \tag{13}$$

Though it is intractable, following the similar approach as we did for the vector field, we can approve that it is equivalent to the conditional score loss

$$\mathcal{L}_{\mathrm{CDM}} = \mathbb{E}_{t\sim\mathcal{U}(0,1), x\sim P_t(\cdot|z), z\sim P_{\mathrm{data}}}\left[\|s_t^\theta(x) - \nabla_x \log p_t(x \mid z))\|^2\right], \tag{14}$$

where we have $\mathcal{L}_{\mathrm{DM}} = \mathcal{L}_{\mathrm{CDM}} + C$.

Loss for the gaussian example For the gaussian example, we have the conditional score function

$$\nabla_x \log p_t(x \mid z) = -\beta_t^{-2}(x - \alpha_t z).$$

The score matching loss is then given by

$$\mathcal{L}_{\mathrm{CDM}}^{\mathcal{G}} = \mathbb{E}_{t\sim\mathcal{U}(0,1), x\sim P_t(\cdot|z), z\sim P_{\mathrm{data}}}\left[\|s_t^\theta(x) + \beta_t^{-2}(x - \alpha_t z)\|^2\right]. \tag{15}$$

With $x = \alpha_t z + \beta_t \epsilon$ where $\epsilon \sim \mathcal{N}(0, I)$, we further have

$$\mathcal{L}_{\mathrm{CDM}}^{\mathcal{G}} = \mathbb{E}_{t\sim\mathcal{U}(0,1), \epsilon\sim\mathcal{N}(0,I), z\sim P_{\mathrm{data}}}\left[\|s_t^\theta(\alpha_t z + \beta_t \epsilon) + \frac{\epsilon}{\beta_t}\|^2\right]. \tag{16}$$

Since $\beta_t \to 0$ as $t \to 1$, the loss is numerically unstable, it can be reformed as

$$\mathcal{L}_{\mathrm{CDM}}^{\mathcal{G}} = \mathbb{E}_{t\sim\mathcal{U}(0,1), \epsilon\sim\mathcal{N}(0,I), z\sim P_{\mathrm{data}}}\left[\|\epsilon_t^\theta(\alpha_t z + \beta_t \epsilon) - \epsilon\|^2\right], \tag{17}$$

where $\epsilon_t^\theta = -\beta_t s_t^\theta$ predicts the noise added during the corruption, as people did in Denoising Diffusion Probabilistic Model (DDPM).

The algorithm for learning the score function is given by Algorithm 0.

Then with both u^θ and $s^\theta(\epsilon^\theta)$ we can apply Euler-Maruyama's method to sample from P_{data} by simulating the SDE.

3.3 Efficient training for diffusion models

Compared to flow models, diffusion models require both the vector field u^θ and the score function s^θ as opposed to flow models, which seems to double the computational cost. However, there are two main approaches to tackle this problem:

- Training one single neural network with two distinct outputs, which minimize the extra computation.
- For gaussian probability path, the vector field and the score function can be converted to each other. Thus, training a neural network for either should be enough.

Algorithm 4 Training algorithm for score functions

```
Require: Dataset $\{z_n\}_{n=1}^N$, noise scheduler $\alpha_t$ and $\beta_t$.
  Initialize model $s^\theta$ or $\epsilon^\theta$
  for $i = 1, 2, \dots, T$ do
      Draw a sample $z$ from $\{z_n\}_{i=1}^N$.
      Draw a sample $\epsilon \sim \mathcal{N}(0, I)$.
      Draw a sample $t \sim \mathcal{U}(0, 1)$.
      Compute score loss $\mathcal{L}(\theta) = \|s^\theta(\alpha_t z + \beta_t \epsilon) + \frac{\epsilon}{\beta_t}\|^2$ (or DDPM $\mathcal{L}(\theta) = \|\epsilon^\theta(\alpha_t z + \beta_t \epsilon) - \epsilon\|^2$).
      Back propagate to update model parameters $\theta$.
  end for
  Return $\theta$
```

Conversion formula for Gaussian probability path For the gaussian probability path $P_t(\cdot \mid z) \sim \mathcal{N}(\alpha_t z, \beta_t I)$, using simple algebra we have following conversion formula for both conditional and marginal between vector fields and score functions.

$$u_t^{\text{target}}(x \mid z) = \left(\frac{\dot{\alpha}_t}{\alpha_t}\beta_t^2 - \dot{\beta}_t \beta_t\right) \nabla_x \log p_t(x \mid z) + \frac{\dot{\alpha}_t}{\alpha_t} x, \tag{18}$$

$$u_t^{\text{target}}(x) = \left(\frac{\dot{\alpha}_t}{\alpha_t}\beta_t^2 - \dot{\beta}_t \beta_t\right) \nabla_x \log p_t(x) + \frac{\dot{\alpha}_t}{\alpha_t} x \tag{19}$$

Then we can just train to learn either s^θ or u^θ only, and then obtain another from it and use both for sampling/simulating the SDE. If we have s^θ, we can apply arbitrary σ_t to sample from P_{data} using Euler-Maruyama's method via the following SDE:

$$dW = \left[\left(\frac{\dot{\alpha}_t}{\alpha_t}\beta_t^2 - \dot{\beta}_t \beta_t + \frac{\sigma_t^2}{2}\right) s^\theta(x)\right] dt + \frac{\dot{\alpha}_t}{\alpha_t} x + \sigma_t dW_t.$$

4 Comparison to other literatures

Discrete time vs continuous time Diffusion models are firstly developed from a perspective of markov chain in discrete time, but later people realise that understanding it from an SDE/ODE's perspective in continuous time is much simpler and mathematically cleaner. The loss in discrete time is constructed via an evidence lower bound, which is the lower bound of the loss we are interested in in discrete time but becomes tight (not a bound any more) in continuous time. But the loss functions in both approach are identical.

Forward process vs probability path Using an inverted time convention, the SDE form of the forward process is given by

$$\hat{X}_0 = z, d\hat{X}_t = u^{\text{forw}}(\hat{X}_t)dt + \sigma_t^{\text{forw}} dW_t,$$

which is designed such that for $t \to \infty$, $\hat{X}_t \to \mathcal{N}(0, I)$. This is equivalent to our conditional probability path $P_t(\cdot \mid z)$ in an inverted time convention. However, this construction is annoying since we need to know $\hat{X}_t \mid \hat{X}_0$ in closed form to avoid simulating this SDE during the training stage, which must be an affine transformation of z. This restrict the choice of our noise schedulers.

So, forward pass is a specific way to construct a probability path.

Time-reversal vs continuity (Fokker-Planck) equation In the original paper, u^{target} and $\nabla \log p_t(x)$ are not constructed by continuity and Fokker-Planck equation but via the time-reversal of the constructed probability path where $P_0 = P_{\text{data}}$ and $P_1 = P_{\text{init}}$, which can be described with the following SDE

$$dX_t^{\text{rev}} = (-u_t^{\text{forw}}(X_t^{\text{rev}}) + \frac{(\sigma_t^{\text{rev}})^2}{2} \nabla p_t(X_t^{\text{rev}})) + \sigma_t^{\text{rev}} dW_t,$$

where $\sigma_t^{\text{rev}} = \sigma_{T-t}^{\text{forw}}$.

However, this is an overkill since for generative modelling, we are only interested in the final point X_1 disgarding intermediate points.

Flow matching and stochastic interpolants The construction of this work is from a perspective of flow matching and then extend to its SDE form. Alternatively all the work could be done in the language of *stochastic interpolant*, which intends to interpolate between two arbitrary distributions. One should be clear that using flow matching, the ODE itself is deterministic and all the randomness in the generation is in sampling $x_0 \sim N(0, I)$ and the generated $x_1 \mid x_0$ is always a point. Using SDE, both sampling x_0 and generating process are random, yielding more diverse output.

5 Building an image generator

Now we can model the probability for P_{data}. However, for image generation, instead of generating an arbitrary image, we are more interested in generating an image guided by a caption y. This can be done by modelling the conditional probability on caption $P_{\text{data}}(\cdot \mid y)$.

Terminology Here in the notes, we denote conditioned on image caption y as *guided*. For example we call $P_{\text{data}}(\cdot \mid y)$ guided data distribution, $u^{\text{target}}(\cdot \mid z)$ where $z \sim P_{\text{data}}(\cdot \mid y)$ the guided conditional vector field to avoid the use of term *condition* on y and z at the same time.

Guided generative model We are interested in modelling the guided data distribution $P_{\text{data}}(\cdot \mid y)$ with variable $y \in \mathcal{Y}$.

5.1 Guided flow model

By replacing all P_{data} with $P_{\text{data}}(\cdot \mid y)$, we can obtain the training target for a flow model for a fixed image caption y:

$$\mathcal{L}'_{\text{GCFM}} = \mathbb{E}_{t\sim\mathcal{U}(0,1), x\sim P_t(\cdot|z), z\sim P_{\text{data}}(\cdot|y)} \left[\|u_t^\theta(x \mid y) - u_t^{\text{target}}(x \mid z)\|^2\right].$$

Note here $u^\theta(x \mid y)$ does not depend on y at all since y is always fixed. Further by expanding the expectation to $y \sim \mathcal{Y}$, we then have

$$\mathcal{L}_{\text{GCFM}} = \mathbb{E}_{t\sim\mathcal{U}(0,1), x\sim P_t(\cdot|z), z\sim P_{\text{data}}(\cdot|y), y\sim\mathcal{Y}} \left[\|u_t^\theta(x \mid y) - u_t^{\text{target}}(x \mid z)\|^2\right], \tag{20}$$

where $u^\theta : (\mathbb{R}^d, \mathcal{Y}, [0,1]) \to \mathbb{R}^d, (x, y, t) \mapsto u_t^\theta(x \mid y)$ is a neural network.

Further improvements Although Equation 22 is mathematically sound, empirically models trained with it are not well-fit enough for the guided image generation and stronger guidance is required.

Recall the conversion between the vector field $u_t^{\text{target}}(x)$ and the score function $\nabla_x \log p_t(x)$ for gaussian probability path, we have the guided version of the conversion

$$u_t^{\text{target}}(x \mid y) = ax + b\nabla_x \log p_t(x \mid y)$$

where $a = \frac{\dot\alpha_t}{\alpha_t}$ and $b = \frac{\dot\alpha_t}{\alpha_t}\beta_t^2 - \dot\beta_t\beta_t$.

Further we have

$$\begin{aligned} u_t^{\text{target}}(x \mid y) &= ax + b\nabla_x \log p_t(x \mid y) \\ &= \underbrace{ax + b\nabla_x \log p_t(x)}_{u_t^{\text{target}}(x)} + b\nabla_x \log p_t(y \mid x) - \underbrace{b\nabla_x \log p_t(y)}_{\text{const.}} \\ &= u_t^{\text{target}}(x) + b\nabla_x \log p_t(y \mid x) - \text{const.} \end{aligned}$$

It is observed that all the information from the guidance is in the second term. To enhance the guidance, it is reasonable to scale it up with $w > 1$:

$$\begin{aligned}\tilde{u}_t^{\text{target}}(x \mid y) &= u_t^{\text{target}}(x) + wb\nabla_x \log p_t(y \mid x)\\ &= ax + b\nabla_x \log p_t(x) + wb(\nabla_x \log p_t(x \mid y) + \underbrace{\nabla_x \log p_t(y)}_{\text{const.}} - \nabla_x \log p_t(x))\\ &= (1-w)(\underbrace{ax + b\nabla_x \log p_t(x)}_{u_t^{\text{target}}(x)}) + w(\underbrace{ax + b\nabla_x \log p_t(x \mid y)}_{u_t^{\text{target}}(x\mid y)})\\ &= (1-w)u_t^{\text{target}}(x) + wu_t^{\text{target}}(x \mid y).\end{aligned}$$

To model both $u_t^{\text{target}}(x)$ and $u_t^{\text{target}}(x \mid y)$ with one single neural network, we can use the following objective function

$$\mathcal{L}_{\text{GCFM}}^{\text{CF}} = \mathbb{E}_{t\sim\mathcal{U}(0,1),x\sim P_t(\cdot|z),z\sim P_{\text{data}}(\cdot|y),y\sim\mathcal{Y}}\left[\|u_t^\theta(x \mid y) - u_t^{\text{target}}(x \mid z)\|^2\right], \text{ replace } y = \emptyset \text{ with probability } \eta. \tag{21}$$

This is can be done with Algorithm below

Algorithm 5 Training algorithm for guided flow method

```
Require: Dataset $\{z_n\}_{n=1}^N$
  Initialize model $u^\theta$
  for $i = 1, 2, \dots, T$ do
      Draw a sample $z$ from $\{z_n\}_{i=1}^N$ with its label $y$.
      Let $y = \emptyset$ at probability of $\eta$.
      Draw a sample $\epsilon \sim \mathcal{N}(0, I)$.
      Draw a sample $t \sim \mathcal{U}(0, 1)$.
      Compute loss $\|u^\theta(\alpha_t z + \beta_t \epsilon \mid y) - (\dot{\alpha}_t z + \dot{\beta}_t \epsilon)\|^2)$.
      Back propagate to update model parameters $\theta$.
  end for
  Return $\theta$
```

Then we have $u_t^{\text{target}}(x) \approx u_t^\theta(x \mid \emptyset)$ and $u_t^{\text{target}}(x \mid y) \approx u_t^\theta(x \mid y)$. With different w by simulating ODE with $\tilde{u}^{\text{target}}(x \mid y)$ we can obtain better empirical results.

5.2 Guided diffusion model

Additional to the vector field, we are also interested in the guided score function $\nabla_x \log p_t(x \mid y)$ to sample from the SDE. This can be done with

$$\mathcal{L}_{\text{GCDM}} = \mathbb{E}_{t\sim\mathcal{U}(0,1),x\sim P_t(\cdot|z),z\sim P_{\text{data}}(\cdot|y),y\sim\mathcal{Y}}\left[\|s_t^\theta(x \mid y) - s_t^{\text{target}}(x \mid z)\|^2\right], \tag{22}$$

where $s^\theta : (\mathbb{R}^d, \mathcal{Y}, [0,1]) \to \mathbb{R}^d, (x, y, t) \mapsto s_t^\theta(x \mid y)$ is a neural network.

Further improvement To make label enhancement align with diffusion models, we have

$$\nabla_x \log p_t(x \mid y) = \nabla_x \log p_t(y \mid x) + \nabla_x \log p_t(x) - \nabla_x \log p_t(y)$$

By enhancing the label y we have

$$\begin{aligned}\tilde{s}_t^{\text{target}}(x \mid y) &= w\nabla_x \log p_t(y \mid x) + \nabla_x \log p_t(x)\\ &= (1-w)s_t^{\text{target}}(x) + ws_t^{\text{target}}(x \mid y).\end{aligned}$$

Then the loss for the score function is given by

$$\mathcal{L}_{\text{GCSM}}^{\text{CF}} = \mathbb{E}_{t\sim\mathcal{U}(0,1),x\sim P_t(\cdot|z),z\sim P_{\text{data}}(\cdot|y),y\sim\mathcal{Y}}\left[\|s_t^\theta(x \mid y) - s_t^{\text{target}}(x \mid z)\|^2\right], \text{ replace } y = \emptyset \text{ with probability } \eta. \tag{23}$$

Inference With $u_t^\theta(x \mid y), s_t^\theta(x \mid y)$ and a fix w, we have

$$\begin{aligned}\tilde{u}_t(x \mid y) &= (1-w)u_t^\theta(x \mid \emptyset) + wu_t^\theta(x \mid y)\\ \tilde{s}_t(x \mid y) &= (1-w)s_t^\theta(x \mid \emptyset) + ws_t^\theta(x \mid y).\end{aligned}$$

Then we simulate the SDE

$$dX_t = \left[\tilde{u}_t(x \mid y) + \frac{\sigma_t^2}{2}\tilde{s}_t(x \mid y)\right] + \sigma_t dW_t,\ \ X_0 \sim P_{\text{inint}}$$