Deep Generative Models

16. Flow Matching

- 국가수리과학연구소 산업수학혁신센터 김민중

Denoising score matching with Langevin dynamics

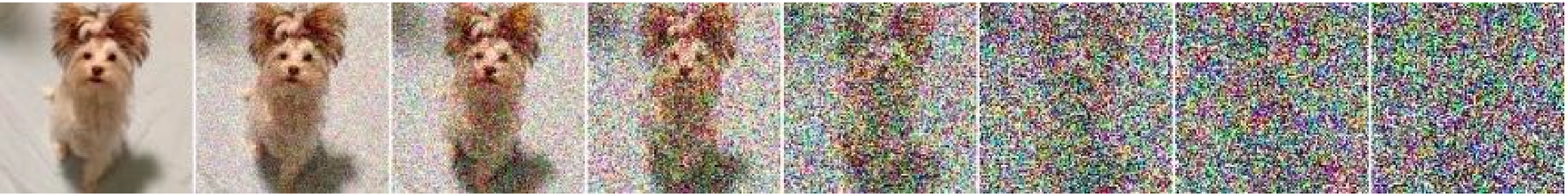
- Let $q_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x}) := N(\tilde{\boldsymbol{x}}|\boldsymbol{x}, \sigma^2 I)$, $q_\sigma(\tilde{\boldsymbol{x}}) := \int p_{data}(\boldsymbol{x}) q_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x}) d\boldsymbol{x}$
- Consider a sequence of positive noise scales

$$\sigma_1 < \sigma_2 < \cdots < \sigma_L$$

- σ_1 is small enough $q_{\sigma_1}(\boldsymbol{x}) \approx p_{data}(\boldsymbol{x})$
- σ_L is large enough $q_{\sigma_L}(\boldsymbol{x}) \approx N(\boldsymbol{x}|\boldsymbol{0}, \sigma_L^2 \boldsymbol{I})$

Data space

Noise space



p_{data} q_{σ_1} q_{σ_2} $\cdots$ $q_{\sigma_L} \approx N(\boldsymbol{0}, \sigma_L^2 \boldsymbol{I})$

Denoising diffusion probabilistic models(DDPM)

- Positive noise scales $0 < \beta_1 < \beta_2 \cdots < \beta_T < 1$
- $\boldsymbol{x}_0 \sim p_{data}(\boldsymbol{x})$, construct latent variables $\{\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_T\}$ s.t.

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) \coloneqq \boldsymbol{N}(\boldsymbol{x}_t|\sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t\boldsymbol{I})$$

- I.e., $q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \boldsymbol{N}(\boldsymbol{x}_0|\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)\boldsymbol{I})$ where $\alpha_t \coloneqq 1-\beta_t$, $\bar{\alpha}_t \coloneqq \prod_{s=1}^{t} \alpha_s$
- Similar to SMLD, we can denote the perturbed data distribution

$$q(\boldsymbol{x}_t) \coloneqq \int q(\boldsymbol{x}_t|\boldsymbol{x}) p_{data}(\boldsymbol{x}) d\boldsymbol{x}$$

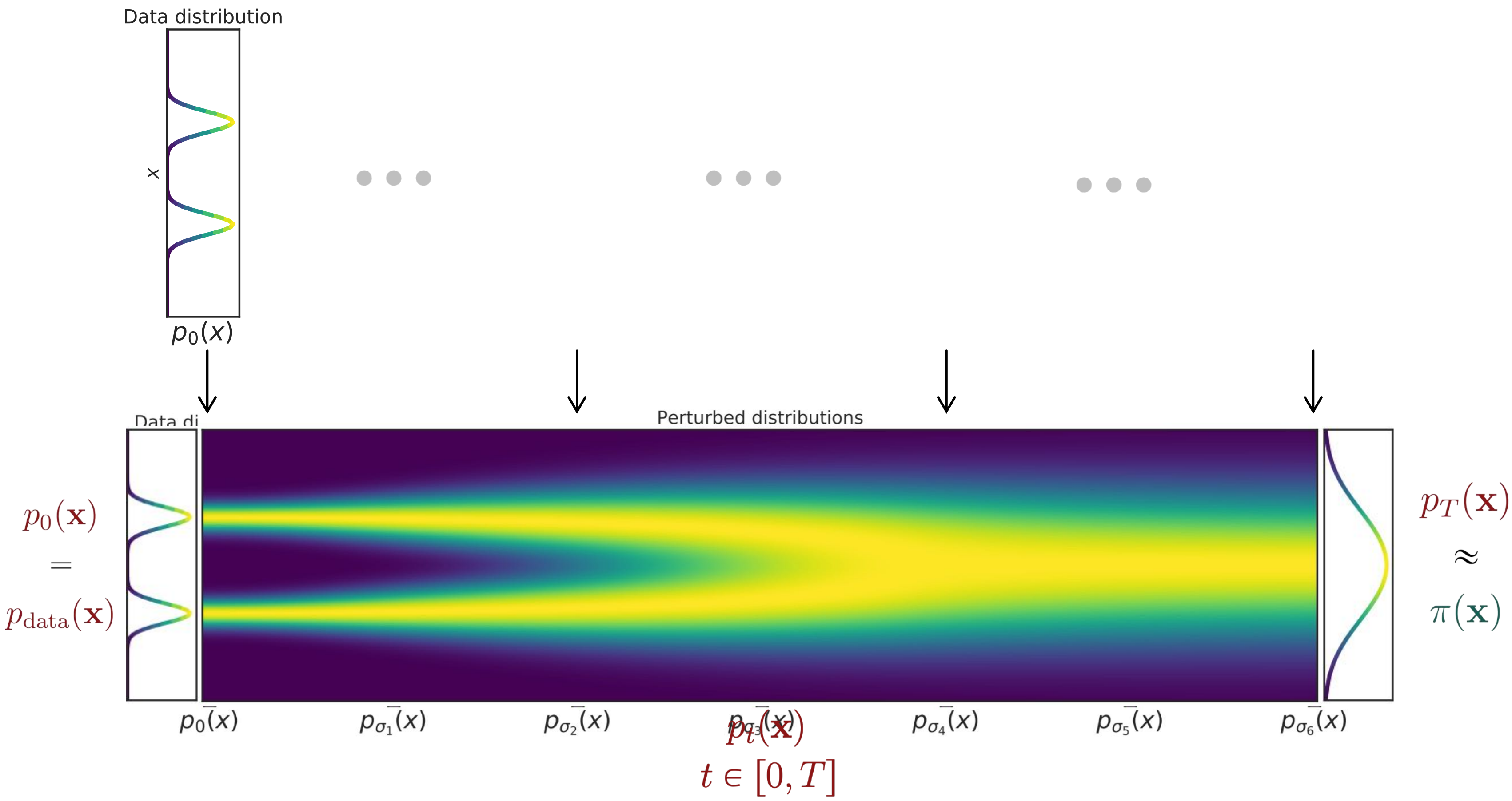
- The noise scales are prescribed s.t. $\boldsymbol{x}_T \sim q(\boldsymbol{x}_T) \approx N(\boldsymbol{0}, \boldsymbol{I})$

p_{data} $q(\boldsymbol{x}_1)$ $q(\boldsymbol{x}_2)$ $\cdots$ $q(\boldsymbol{x}_T)$

Summary of score-based models

- **SMLD** and **DDPM** involve sequentially corrupting training data with slowly increasing noise, and then learning to reverse this corruption to form a generative model of the data
- **SMLD** estimates the score at each noise scale and then use Langevin dynamics to sample from a sequence of decreasing noise scales during generation
- **DDPM** trains a sequence of probabilistic models to reverse each step of the noise corruption, using knowledge of the functional form of the reverse distributions to make training tractable

Infinite noise levels



Stochastic differential equation

- For $t \geq 0$, consider an SDE which possesses the following form
$$d\boldsymbol{x}_t = \boldsymbol{f}(\boldsymbol{x}_t, t)dt + g(t)d\boldsymbol{w}_t$$
  - $\boldsymbol{f}(\cdot, t): \mathbb{R}^d \to \mathbb{R}^d$ (drift coefficient)
  - $g(t) \in \mathbb{R}$ (diffusion coefficient)
  - $\boldsymbol{w}_t$ denotes a standard Brownian motion
  - $d\boldsymbol{w}_t$ can be viewed as infinitesimal white noise
  - $\{\boldsymbol{x}_t\}_{t\in[0,T]}$ is a stochastic process

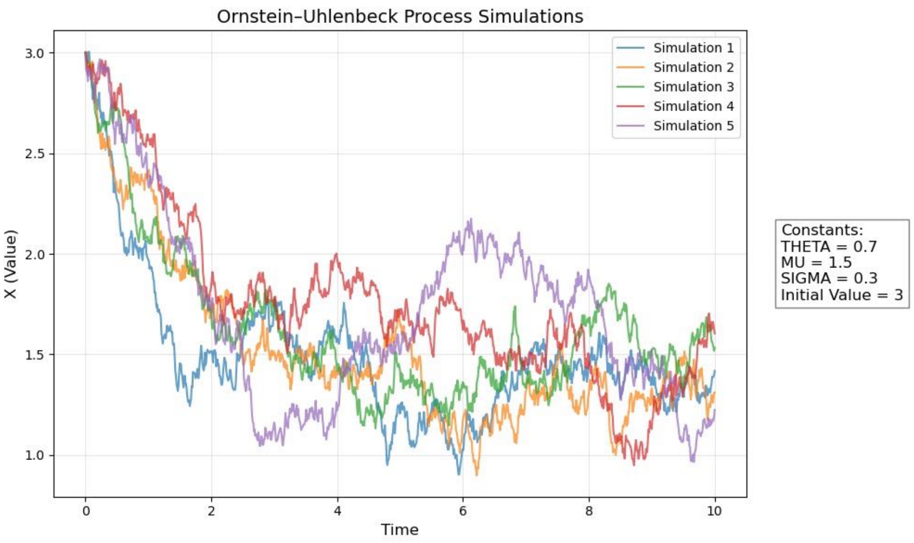
- Numerically, the SDE can be seen as the limit
$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \Delta t f(\boldsymbol{x}_i, i\Delta t) + g(i\Delta t)\sqrt{\Delta t}\boldsymbol{z}_i \quad i = 0,1,\cdots$$
- Under $\Delta t \to 0$, where $t = i\Delta t$ and $\boldsymbol{z}_i \sim N(\boldsymbol{0}, \boldsymbol{I})$

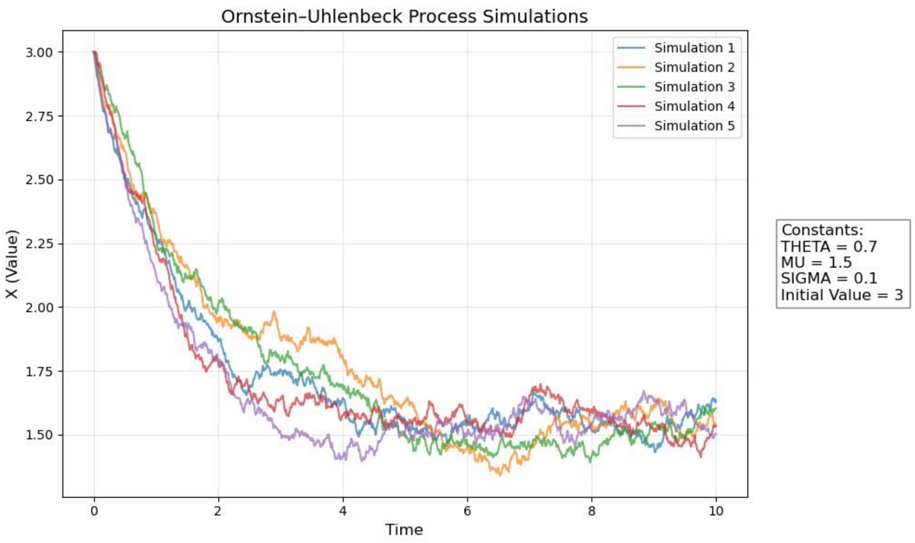
Example: 1-dim Ornstein-Uhlenbeck process

- The Ornstein-Uhlenbeck process x_t is defined by

$$dx_t = \theta(\mu - x_t)dt + \sigma dw_t$$

- where $\theta > 0$, $\sigma > 0$, $\mu \in \mathbb{R}$ and w_t is 1-dim standard Brownian motion

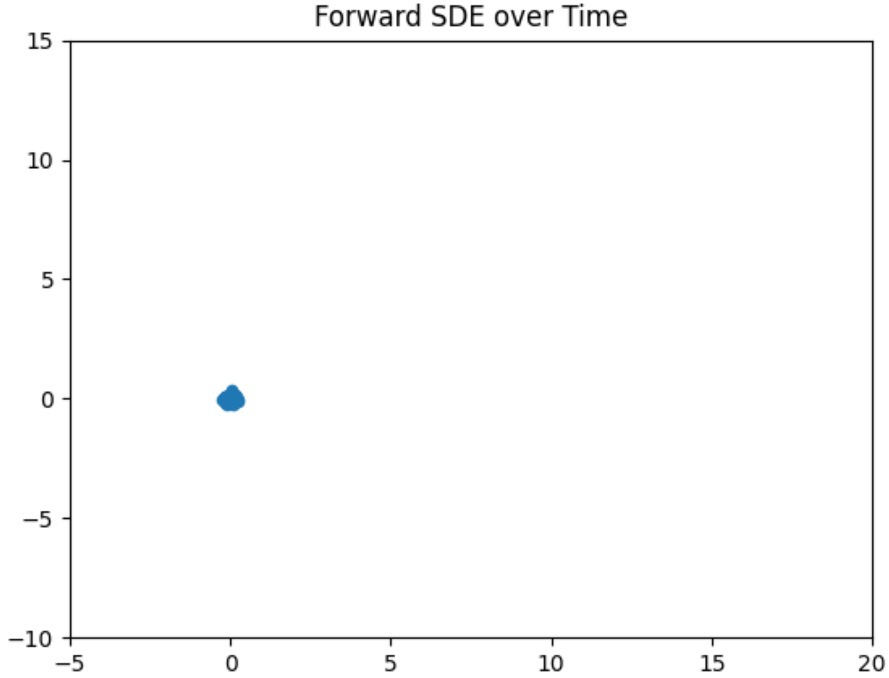




Example: Forward SDE

$$d\boldsymbol{x}_t = \begin{pmatrix} 1 \\ 0 \end{pmatrix} dt + \begin{pmatrix} 0.1 & 0 \\ 0 & 0.1 \end{pmatrix} d\boldsymbol{w}_t, \qquad p_0(\boldsymbol{x}) = N\left(\boldsymbol{x} \middle| \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0.1 & 0 \\ 0 & 0.1 \end{pmatrix}\right)$$

- Then, $p_t(\boldsymbol{x}) = N\left(\boldsymbol{x} \middle| \begin{pmatrix} t \\ 0 \end{pmatrix}, \begin{pmatrix} 0.1+t & 0 \\ 0 & 0.1+t \end{pmatrix}\right)$



Example: 1-dim Ornstein-Uhlenbeck process

- Consider the Ornstein-Uhlenbeck process x_t is defined by

$$dx_t = -\theta x_t dt + \sigma dw_t$$

- Then,

$$p(x_t|x_0) = N\left(x_t \middle| e^{-\theta t}x_0, \frac{\sigma^2}{2\theta}\left(1 - e^{-2\theta t}\right)\right)$$

- If $x_0 \sim N\left(0, \frac{\sigma^2}{\theta}\right)$, then

$$x_t \sim N\left(0, \frac{\sigma^2}{2\theta}\right), \qquad p_t(x) = \frac{1}{\sqrt{\pi\sigma^2/\theta}} \exp\left[-\frac{\theta}{\sigma^2}x^2\right]$$

- $p_t(x)$ satisfies the FP equation

$$0 = \partial_t p_t(x) - \partial_x\left(f p_t(x)\right) + \frac{g^2}{2}\partial_x^2\left(p_t(x)\right)$$

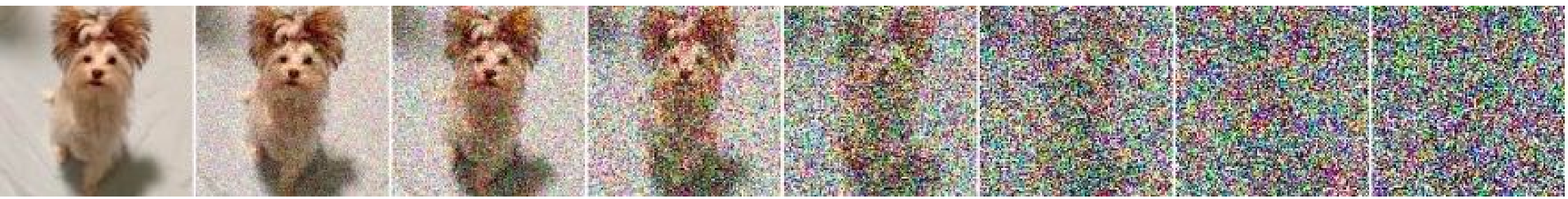
$$= \partial_x\left(\theta x p_t(x)\right) + \frac{g^2}{2}\partial_x^2\left(p_t(x)\right) = 0$$

Example: Ornstein-Uhlenbeck process

- The Ornstein-Uhlenbeck process

$$d\boldsymbol{x}_t = -\theta \boldsymbol{x}_t dt + \sigma d\boldsymbol{w}_t$$

- with $\theta \geq 0$ and $\sigma > 0$ adds noise to the datapoint $\boldsymbol{x}_t$
- As $T \to \infty$, all information is lost



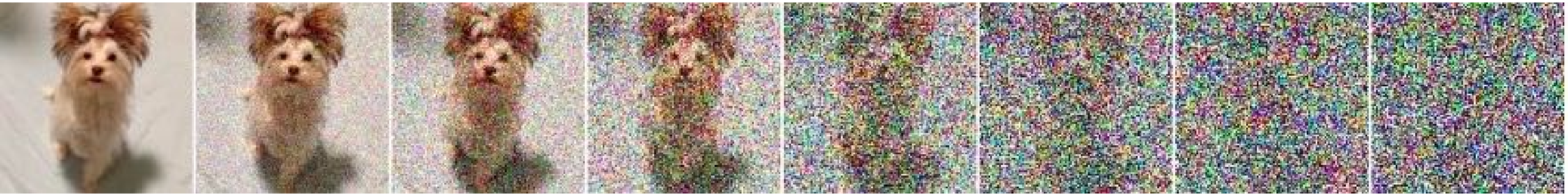
p_{data} … $p_t(\boldsymbol{x})$ … $p_T(\boldsymbol{x})$

Example: Ornstein-Uhlenbeck process

- The Ornstein-Uhlenbeck process

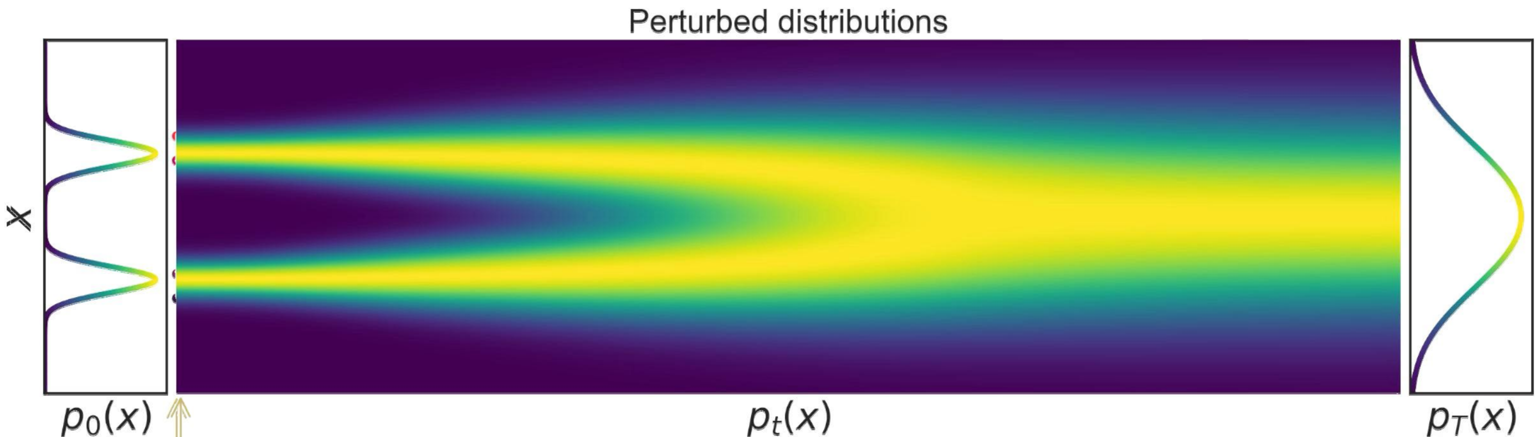
$$d\boldsymbol{x}_t = -\theta \boldsymbol{x}_t dt + \sigma d\boldsymbol{w}_t$$

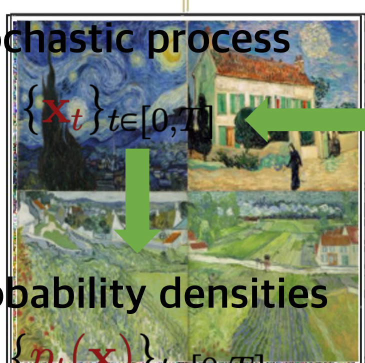
- with $\theta \geq 0$ and $\sigma > 0$ adds noise to the datapoint $\boldsymbol{x}_t$
- As $T \to \infty$, all information is lost



- Since $p(\boldsymbol{x}_t|\boldsymbol{x}_0) = N\left(\boldsymbol{x}_t \middle| e^{-\theta t}\boldsymbol{x}_0, \frac{\sigma^2}{2\theta}\left(1 - e^{-2\theta t}\right)\boldsymbol{I}\right)$, we have $\boldsymbol{x}_T$ is approximately distributed as $N\left(\boldsymbol{0}, \frac{\sigma^2}{2\theta}\boldsymbol{I}\right)$ if $\theta > 0$ and $T \approx \infty$
- Sampling $\boldsymbol{x}_T \sim N\left(\boldsymbol{0}, \frac{\sigma^2}{2\theta}\boldsymbol{I}\right)$ is easy. Can we reverse the SDE to sample $\boldsymbol{x}_0$?

Perturbing data with stochastic processes





Stochastic process

$\{\mathbf{x}_t\}_{t\in[0,T]}$

Probability densities

$\{p_t(\mathbf{x})\}_{t\in[0,T]}$

Stochastic differential equation (SDE)

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

Deterministic drift

Infinitesimal noise

$$p_T(\mathbf{x}) \approx \pi(\mathbf{x})$$

Forward-time SDE

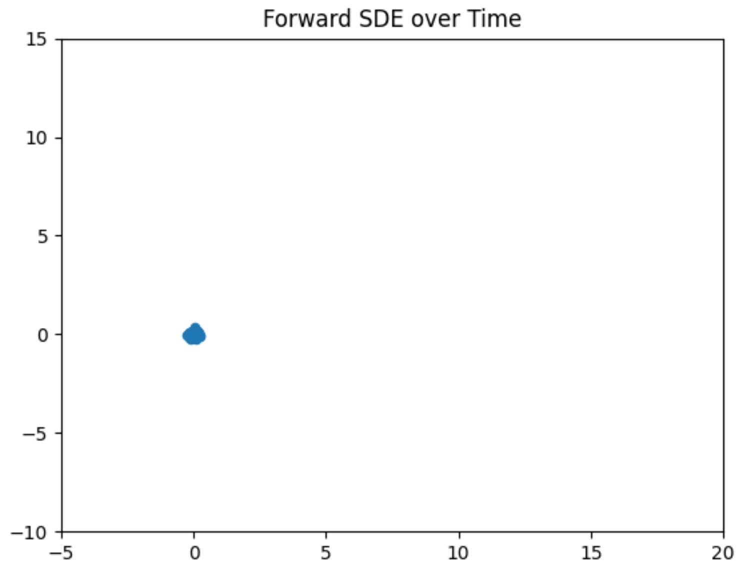
- To simulate

$$d\boldsymbol{x}_t = \boldsymbol{f}(\boldsymbol{x}_t, t)dt + g(t)d\boldsymbol{w}_t, \qquad \boldsymbol{x}_0 \sim p_0$$

- for $0 < t$, sample $\boldsymbol{x}_0 \sim p_0$ and compute

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \Delta t f(x_i, i\Delta t) + g(i\Delta t)\sqrt{\Delta t}\boldsymbol{z}_i \quad i = 0,1,\cdots$$

- for sufficiently small $\Delta t > 0$ and $\boldsymbol{z}_i \sim N(\boldsymbol{0}, \boldsymbol{I})$



Generating samples by reversing the SDE

- For an SDE,

$$d\boldsymbol{x}_t = \boldsymbol{f}(\boldsymbol{x}_t, t)dt + g(t)d\boldsymbol{w}_t, \qquad \boldsymbol{x}_0 \sim p_0$$

- has a corresponding reverse SDE, whose closed form is given by

$$d\boldsymbol{x}_t = \left[\boldsymbol{f}(\boldsymbol{x}_t, t) - g^2(t)\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t)\right]dt + g(t)d\bar{\boldsymbol{w}}_t, \qquad \boldsymbol{x}_T \sim p_T$$

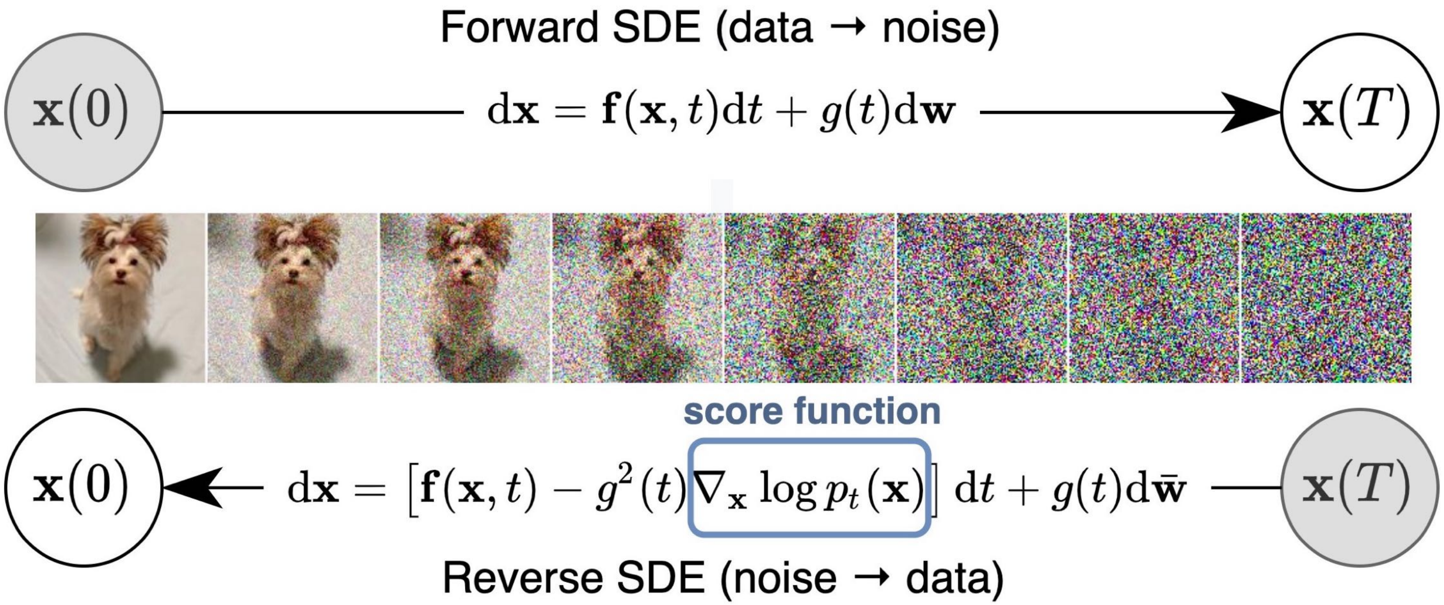
  - dt represents a negative infinitesimal time step
  - $\bar{\boldsymbol{w}}_t$ is a standard BM when time flows backwards from T to 0. I.e. $\bar{\boldsymbol{w}}_t = \boldsymbol{w}_T - \boldsymbol{w}_{T-t}$

- In order to compute the reverse SDE, we need to estimate $\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})$ which is the score function of $p_t(\boldsymbol{x})$

Reverse-time diffusion equation models
B. D. O. Anderson. Stochastic Processes and their Applications. 1982

Generating samples by reversing the SDE

- In order to compute the reverse SDE, we need to estimate $\nabla_x \log p_t(x)$ which is the score function of $p_t(x)$



Estimating the reverse SDE with score-based models

- Solving the reverse SDE requires us to know the terminal distribution $p_T(\boldsymbol{x})$, and the score function $\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})$
- By design, $p_T(\boldsymbol{x})$ is close to the prior distribution $\pi(\boldsymbol{x})$ which is fully tractable

- In order to estimate $\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})$, train a time-dependent score-based model $\boldsymbol{s}_\theta(\boldsymbol{x}, t)$ such that

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

- This is analogous to the NCSM $\boldsymbol{s}_\theta(\boldsymbol{x}, i)$ used for finite noise scales, trained such that $\boldsymbol{s}_\theta(\boldsymbol{x}, i) \approx \nabla_{\boldsymbol{x}} \log p_{\sigma_i}(\boldsymbol{x})$

Estimating the reverse SDE with score-based models

- Training objective for $\boldsymbol{s}_\theta(\boldsymbol{x}, t)$ is a continuous weighted combination of Fisher divergences, given by

$$E_{t\sim U(0,T)}\left[\lambda(t)E_{\boldsymbol{x}\sim p_t(\boldsymbol{x})}[\,\|\boldsymbol{s}_\theta(\boldsymbol{x}, t) - \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})\|_2^2]\right]$$

- where $U(0, T)$ denotes a uniform distribution over the time interval $[0, T]$ and $\lambda: \mathbb{R}_+ \rightarrow \mathbb{R}_+$ is a positive weighting function

Foundation of score-based models

$$\underset{\theta}{\operatorname{argmin}}\, E_{\boldsymbol{x}\sim p_t(\boldsymbol{x})}\left[\, \left\|\boldsymbol{s}_\theta(\boldsymbol{x}, t) - \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})\right\|_2^2\right]$$

$$= \underset{\theta}{\operatorname{argmin}}\, E_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})} E_{\boldsymbol{x}_t\sim p(\boldsymbol{x}_t|\boldsymbol{x})}\left[\, \left\|\boldsymbol{s}_\theta(\boldsymbol{x}_t, t) - \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t|\boldsymbol{x})\right\|_2^2\right]$$

Estimating the reverse SDE with score-based models

- Training objective for $\boldsymbol{s}_\theta(\boldsymbol{x}, t)$ is a continuous weighted combination of Fisher divergences, given by

$$E_{t\sim U(0,T)}\left[\lambda(t) E_{\boldsymbol{x}\sim p_t(\boldsymbol{x})}[\,\|\boldsymbol{s}_\theta(\boldsymbol{x}, t) - \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})\|_2^2]\right]$$

- Where $U(0, T)$ denotes a uniform distribution over the time interval $[0, T]$ and $\lambda: \mathbb{R}_+ \to \mathbb{R}_+$ is a positive weighting function

- The objective can be written as

$$E_{t\sim U(0,T)}\left[\lambda(t) E_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})} E_{\boldsymbol{x}_t\sim p(\boldsymbol{x}_t|\boldsymbol{x})}\left[\,\|\boldsymbol{s}_\theta(\boldsymbol{x}_t, t) - \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t|\boldsymbol{x})\|_2^2\right]\right]$$

- Typically, we use $\lambda(t) \propto 1/E\left[\left\|\nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t|\boldsymbol{x})\right\|_2^2\right]$ to balance the magnitude of different score matching losses across time

Remark of the transition kernel $p(\boldsymbol{x}_t|\boldsymbol{x})$

- We typically need to know the transition kernel $p(\boldsymbol{x}_t|\boldsymbol{x})$
- When $\boldsymbol{f}(\cdot,t)$ is affine, the transition kernel is always a (conditional) Gaussian distribution, where the mean and variance are often known in closed-forms

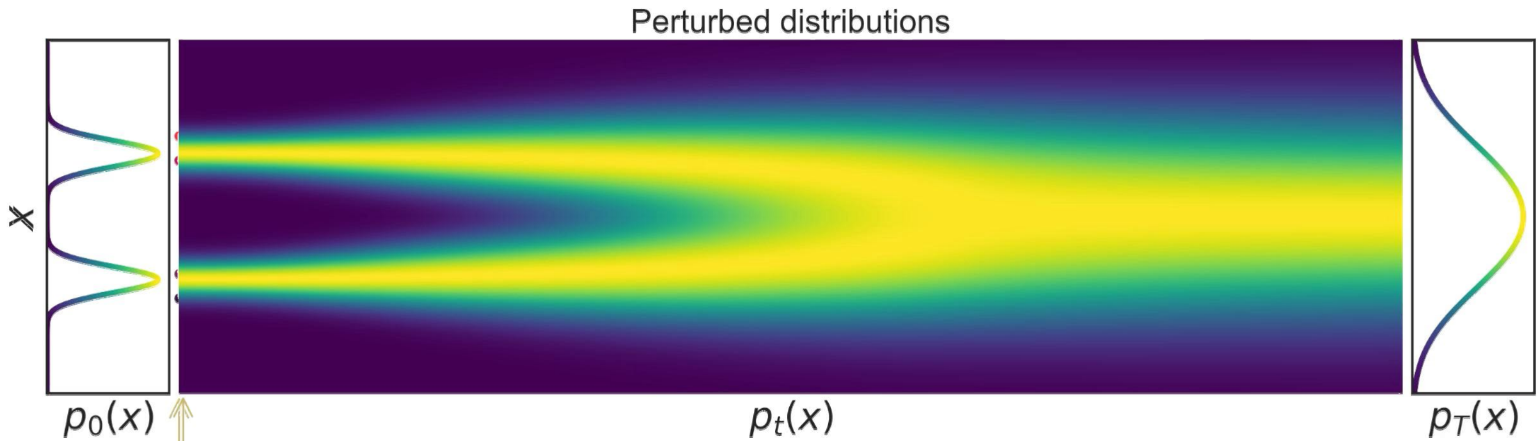
How to solve the reverse SDE

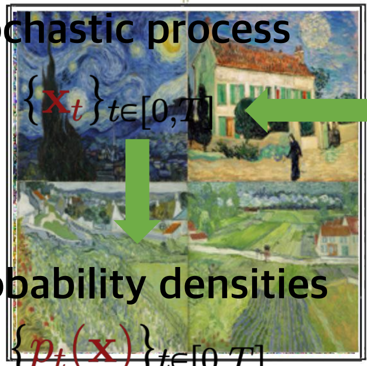
- By solving the estimated reverse SDE with numerical SDE solvers, we can simulate the reverse stochastic process for sample generation
- **Euler-Maruyama method**(analogous to Euler for ODEs)
  - Small positive time step $\Delta t \approx 0$
  - Initializes $t = T$, and iterates the following procedure until $t \approx 0$

$$\Delta \boldsymbol{x} \leftarrow [\boldsymbol{f}(\boldsymbol{x}, t) - g^2(t)\boldsymbol{s}_\theta(\boldsymbol{x}, t)]\Delta t + g(t)\sqrt{\Delta t}\boldsymbol{z}$$
$$\boldsymbol{x} \leftarrow \boldsymbol{x} + \Delta \boldsymbol{x}$$
$$t \leftarrow t - \Delta t$$

  - Here $\boldsymbol{z} \sim N(\boldsymbol{0}, \Delta t \boldsymbol{I})$
  - I.e. $\boldsymbol{x}_{t-\Delta t} = \boldsymbol{x}_t - \Delta t[\boldsymbol{f}(\boldsymbol{x}_t, t) - g^2(t)\boldsymbol{s}_\theta(\boldsymbol{x}_t, t)] + g(t)\sqrt{\Delta t}\boldsymbol{z}$

Perturbing data with stochastic processes





Stochastic process

$\{\mathbf{x}_t\}_{t\in[0,T]}$

Probability densities

$\{p_t(\mathbf{x})\}_{t\in[0,T]}$

Stochastic differential equation (SDE)

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

Deterministic drift

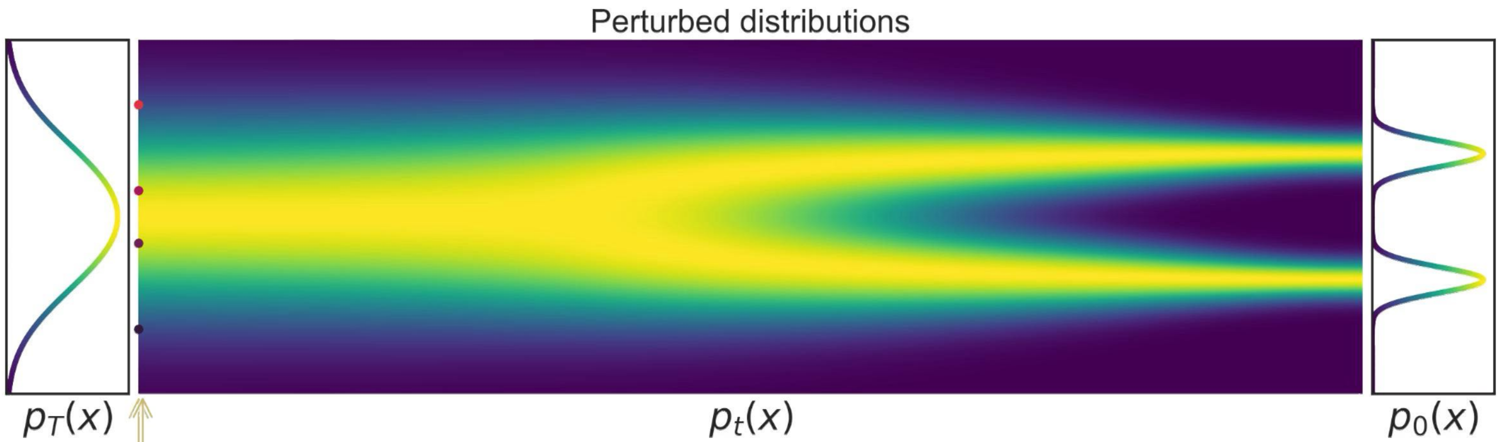
Infinitesimal noise

WLOG: Toy SDE

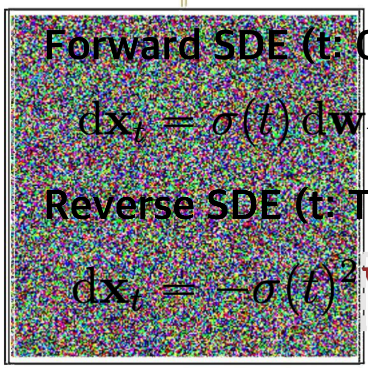
$$\mathrm{d}\mathbf{x}_t = \sigma(t)\,\mathrm{d}\mathbf{w}_t$$

$$p_T(\mathbf{x}) \approx \pi(\mathbf{x})$$

Generation via reverse stochastic processes



$\pi(\mathbf{x}) \approx p_T(\mathbf{x})$



Forward SDE (t: 0→T)

$$\mathrm{d}\mathbf{x}_t = \sigma(t)\,\mathrm{d}\mathbf{w}_t$$

Reverse SDE (t: T→0)

$$\mathrm{d}\mathbf{x}_t = -\sigma(t)^2\,\nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t)\,\mathrm{d}t + \sigma(t)\,\mathrm{d}\bar{\mathbf{w}}_t$$

Infinitesimal noise in the reverse time direction

Score function!

Score-based generative modeling via SDEs

- Time-dependent score-based model

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

- Training objective

$$E_{t\sim U(0,T)}\left[\lambda(t) E_{\boldsymbol{x}\sim p_t(\boldsymbol{x})}[\, \|\boldsymbol{s}_\theta(\boldsymbol{x}, t) - \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})\|_2^2]\right]$$

Score-based generative modeling via SDEs

- Time-dependent score-based model

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

- Training objective

$$E_{t\sim U(0,T)}\left[\lambda(t) E_{\boldsymbol{x}\sim p_t(\boldsymbol{x})}[\,\|\boldsymbol{s}_\theta(\boldsymbol{x}, t) - \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})\|_2^2]\right]$$

- In case of $d\boldsymbol{x}_t = \sigma(t) d\boldsymbol{w}_t$ with $0 \le t \le T$, the reverse-time SDE is

$$d\boldsymbol{x}_t = -\sigma^2(t)\boldsymbol{s}_\theta(\boldsymbol{x}_t, t)dt + \sigma(t)d\overline{\boldsymbol{w}}_t$$

- Euler-Maruyama method

$$\boldsymbol{x}_{t-\Delta t} = \boldsymbol{x}_t - \sigma^2(t)\boldsymbol{s}_\theta(\boldsymbol{x}_t, t)\Delta t + \sigma(t)\boldsymbol{z}$$

- where $\boldsymbol{z} \sim N(\boldsymbol{0}, \Delta t \boldsymbol{I})$

Predictor-Corrector sampling methods

- In addition, there are two special properties of our reverse SDE that allow for even more flexible sampling methods:
  - estimation of $\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})$ via time-dependent score-based model $\boldsymbol{s}_\theta(\boldsymbol{x}, t)$
  - sampling from each marginal distribution $p_t(\boldsymbol{x})$

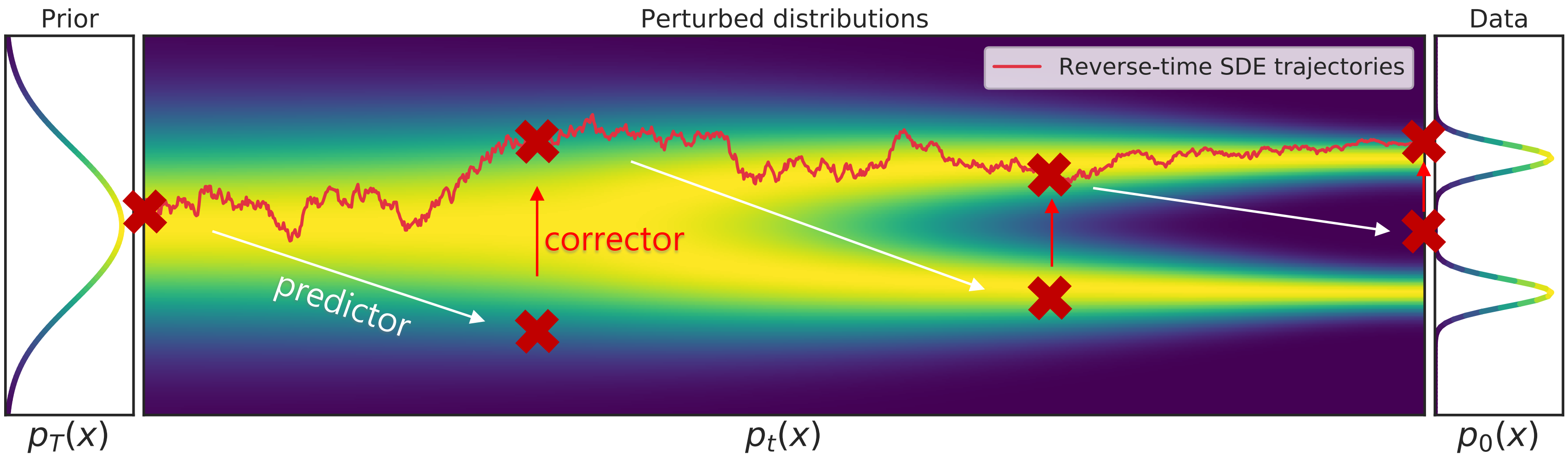
Predictor-Corrector sampling methods

- Thus, we can apply score-based MCMC approaches to fine-tune the trajectories obtained from numerical SDE solvers
- We propose **Predictor-Corrector samplers**
  - **Predictor**: any numerical SDE solver predicting $\boldsymbol{x}_{t-\Delta t} \sim p_{t-\Delta t}(\boldsymbol{x})$ from an existing sample $\boldsymbol{x}_t \sim p_t(\boldsymbol{x})$
  - **Corrector**: score-based MCMC procedure

- At each step of the Predictor-Corrector sampler, we first use the **predictor** to choose a proper step size $\Delta t > 0$, and then predict $\boldsymbol{x}_{t-\Delta t}$ based on the current sample $\boldsymbol{x}_t$
- Next, we run several **corrector** steps to improve the sample $\boldsymbol{x}_{t-\Delta t}$ according to our score-based model $\boldsymbol{s}_\theta(\boldsymbol{x}_{t-\Delta t}, t-\Delta t)$ so that $\boldsymbol{x}_{t-\Delta t}$ becomes a high-quality sample from $p_{t-\Delta t}(\boldsymbol{x})$

Predictor-Corrector sampling methods

- Predictor-Corrector sampling
  - **Predictor**: Numerical SDE solver
  - **Corrector**: Score-based MCMC



VE and VP forward SDEs

- The O-U process x_t is defined by

$$d\boldsymbol{x}_t = -\theta \boldsymbol{x}_t dt + \sigma d\boldsymbol{w}_t$$

- where $\theta > 0$, $\sigma > 0$ and $\boldsymbol{w}_t$ is d-dim standard Brownian motion

- Two types O-U processes are primarily considered for the forward SDE
  - **Variance-exploding(VE)**

$$d\boldsymbol{x}_t = \sigma d\boldsymbol{w}_t$$

$$p(\boldsymbol{x}_t|\boldsymbol{x}_0) = (\boldsymbol{x}_t|\gamma_t \boldsymbol{x}_0, \sigma_t^2 \boldsymbol{I}), \qquad \gamma_t = 1, \sigma_t^2 = t\sigma^2$$

  - **Variance-preserving(VP)**

$$d\boldsymbol{x}_t = -\theta \boldsymbol{x}_t dt + \sigma d\boldsymbol{w}_t$$

$$p(\boldsymbol{x}_t|\boldsymbol{x}_0) = (\boldsymbol{x}_t|\gamma_t \boldsymbol{x}_0, \sigma_t^2 \boldsymbol{I}), \qquad \gamma_t = e^{-\theta t}, \sigma_t^2 = \frac{\sigma^2}{2\theta}\left(1 - e^{-2\theta t}\right)$$

VE and VP forward SDEs

- Two types O-U processes are primarily considered for the forward SDE
  - **Variance-exploding(VE)**

$$d\boldsymbol{x}_t = \sigma d\boldsymbol{w}_t$$

$$p(\boldsymbol{x}_t|\boldsymbol{x}_0) = (\boldsymbol{x}_t|\gamma_t\boldsymbol{x}_0, \sigma_t^2\boldsymbol{I}), \qquad \gamma_t = 1, \sigma_t^2 = t\sigma^2$$

  - **Variance-preserving(VP)**

$$d\boldsymbol{x}_t = -\theta\boldsymbol{x}_t dt + \sigma d\boldsymbol{w}_t$$

$$p(\boldsymbol{x}_t|\boldsymbol{x}_0) = (\boldsymbol{x}_t|\gamma_t\boldsymbol{x}_0, \sigma_t^2\boldsymbol{I}), \qquad \gamma_t = e^{-\theta t}, \sigma_t^2 = \frac{\sigma^2}{2\theta}\left(1 - e^{-2\theta t}\right)$$

  - In both cases,

$$p(\boldsymbol{x}_t|\boldsymbol{x}_0) = (\boldsymbol{x}_t|\gamma_t\boldsymbol{x}_0, \sigma_t^2\boldsymbol{I})$$

  - i.e. $\boldsymbol{x}_t|\boldsymbol{x}_0 = \gamma_t\boldsymbol{x}_0 + \sigma_t\boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon} \sim N(\boldsymbol{0}, \boldsymbol{I})$

General VE SDE

- Let $\sigma(t)$ be a non-decreasing function of t
- General VE SDE:

$$d\boldsymbol{x}_t = \sqrt{\frac{d[\sigma^2(t)]}{dt}}\, d\boldsymbol{w}_t$$

$$p(\boldsymbol{x}_t|\boldsymbol{x}_0) = N(\boldsymbol{x}_t|\gamma_t \boldsymbol{x}_0, \sigma_t^2 \boldsymbol{I}), \qquad \gamma_t = 1, \sigma_t^2 = \sigma^2(t)$$

- Although the mean is preserved, the variance explodes

General VP SDE

- Let $\theta : [0, \infty) \to \mathbb{R}_+$ be a function

- General VP SDE:

$$d\boldsymbol{x}_t = -\frac{\theta(t)}{2}\boldsymbol{x}_t dt + \sqrt{\theta(t)} d\boldsymbol{w}_t$$

$$p(\boldsymbol{x}_t|\boldsymbol{x}_0) = N(\boldsymbol{x}_t|\gamma_t \boldsymbol{x}_0, \sigma_t^2 \boldsymbol{I}),$$

$$\gamma_t = e^{-\frac{1}{2}\int_0^t \theta(s)ds}, \sigma_t^2 = 1 - e^{-\int_0^t \theta(s)ds}$$

- In particular,

$$\mathrm{Var}(\boldsymbol{x}_t) = \boldsymbol{I} + e^{-\int_0^t \theta(s)ds}(\mathrm{Var}(\boldsymbol{x}_0) - \boldsymbol{I})$$

  - If $\mathrm{Var}(\boldsymbol{x}_0) = \boldsymbol{I}$, then

$$\mathrm{Var}(\boldsymbol{x}_t) = \boldsymbol{I}$$

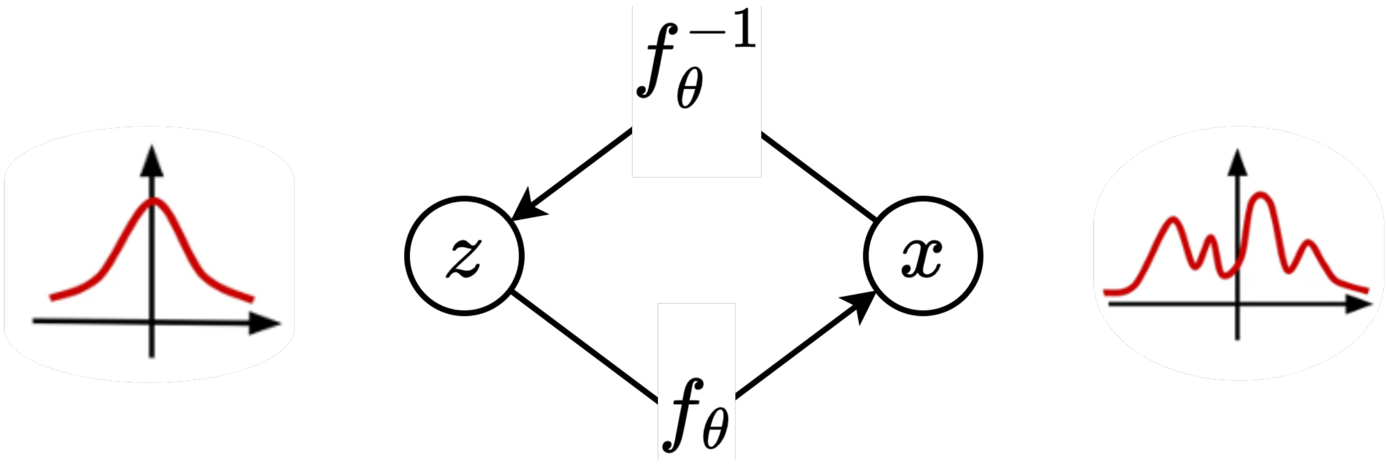
Training with O-U and DSM

- Using $\boldsymbol{x}_t|\boldsymbol{x}_0 = \gamma_t \boldsymbol{x}_0 + \sigma_t \boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \boldsymbol{I})$, the score function simplifies to

$$\nabla_{x_t} \log p(\boldsymbol{x}_t|\boldsymbol{x}) = \frac{\gamma_t \boldsymbol{x} - \boldsymbol{x}_t}{\sigma_t^2} = -\frac{\boldsymbol{\epsilon}}{\sigma_t}$$

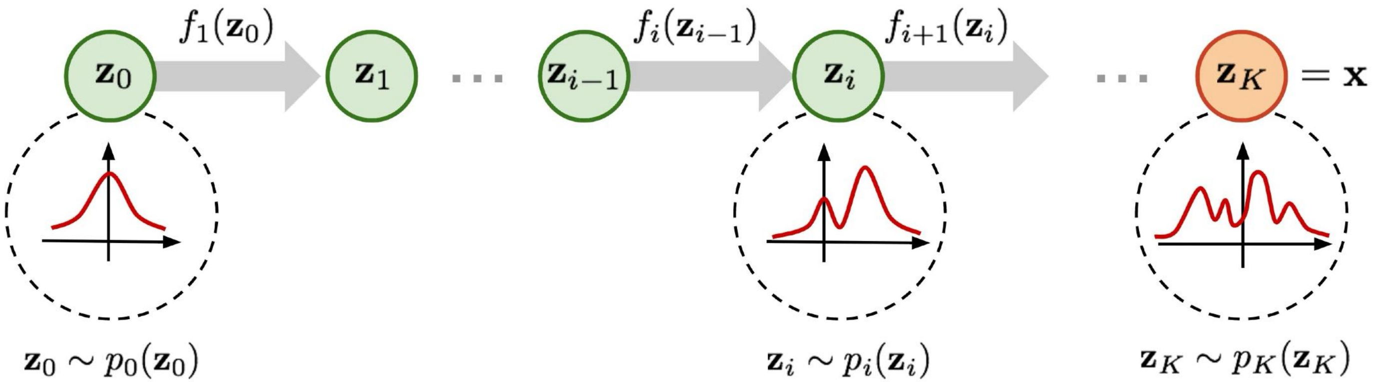
Normalizing flow models

- Consider a directed, latent variable model over observed variables X and latent variables Z
- In a normalizing flow model, the mapping between Z and X, given by $\boldsymbol{f}_\theta: \mathbb{R}^d \rightarrow \mathbb{R}^d$, is deterministic and invertible such that $X = \boldsymbol{f}_\theta(Z)$ and $Z = \boldsymbol{f}_\theta^{-1}(X)$



A Flow of Transformations

$$f_\theta(z_0) \coloneqq f_K \circ f_{K-1} \circ \cdots \circ f_1(z_0) = z_K$$



- Start with a simple distribution for z_0 (e.g., Gaussian)
- Apply a sequence of K invertible transformations to finally obtain $x = z_K$

$$f_\theta^{-1}(x) = f_1^{-1} \circ f_2^{-1} \circ \cdots \circ f_K^{-1}(x)$$

A Flow of Transformations

$$\boldsymbol{f}_\theta(\boldsymbol{z}_0) \coloneqq \boldsymbol{f}_K \circ \boldsymbol{f}_{K-1} \circ \cdots \circ \boldsymbol{f}_1(\boldsymbol{z}_0) = \boldsymbol{z}_K = \boldsymbol{x}$$
$$\boldsymbol{f}_\theta^{-1}(\boldsymbol{x}) = \boldsymbol{f}_1^{-1} \circ \boldsymbol{f}_2^{-1} \circ \cdots \circ \boldsymbol{f}_K^{-1}(\boldsymbol{x})$$

- The marginal likelihood $p_X(\boldsymbol{x})$ is given by

$$\begin{aligned} p_X(\boldsymbol{x};\theta) &= p_Z\left(\boldsymbol{f}_\theta^{-1}(\boldsymbol{x})\right)\left|\det\left(\frac{\partial \boldsymbol{f}_\theta^{-1}(\boldsymbol{x})}{\partial \boldsymbol{x}}\right)\right| \\ &= p_Z(\boldsymbol{z})\left|\det\left(\frac{\partial \boldsymbol{f}_\theta(\boldsymbol{z})}{\partial \boldsymbol{z}}\right)\right|^{-1} \\ &= p_Z\left(\boldsymbol{f}_\theta^{-1}(\boldsymbol{x})\right)\prod_{k=1}^{K}\left|\det\left(\frac{\partial \boldsymbol{f}_k^{-1}(\boldsymbol{x}_k)}{\partial \boldsymbol{x}_k}\right)\right| \end{aligned}$$

Learning and Inference

- Learning via maximum likelihood over the dataset D

$$\max_{\theta} \log p_X(D;\theta) = \sum_{x\in D} \log p_Z\left(\boldsymbol{f}_{\theta}^{-1}(\boldsymbol{x})\right) + \log \left|\det\left(\frac{\partial \boldsymbol{f}_{\theta}^{-1}(\boldsymbol{x})}{\partial \boldsymbol{x}}\right)\right|$$

- Exact likelihood evaluation via inverse transformation $\boldsymbol{x} \mapsto \boldsymbol{z}$ and change of variables formula
- Sampling via forward transformation $\boldsymbol{z} \mapsto \boldsymbol{x}$

$$\boldsymbol{z} \sim \boldsymbol{p}_Z(\boldsymbol{z}), \boldsymbol{x} = \boldsymbol{f}_{\theta}(\boldsymbol{z})$$

- Latent representations inferred via inverse transformation (no inference network required): $\boldsymbol{z} = \boldsymbol{f}_{\theta}^{-1}(\boldsymbol{x})$

Remark

- How to enforce invertibility?
- How to compute its inverse?
- How to compute the Jacobian efficiently?

Residual flow (2019, 2010)

- Flow has the form

$$\boldsymbol{f}_{k+1}(\boldsymbol{z}_k) = \boldsymbol{z}_k + \delta \boldsymbol{u}_k(\boldsymbol{z}_k)$$

- for some $\delta > 0$ and Lipschitz residual connection $\boldsymbol{u}_k$

Continuous time limit

- Residual flow are transformations of the form

$$\boldsymbol{f}_{k+1}(\boldsymbol{z}_k) = \boldsymbol{z}_k + \delta \boldsymbol{u}_k(\boldsymbol{z}_k)$$

- for some $\delta > 0$ and Lipschitz residual connection $\boldsymbol{u}_k$
- We can re-arrange this to get

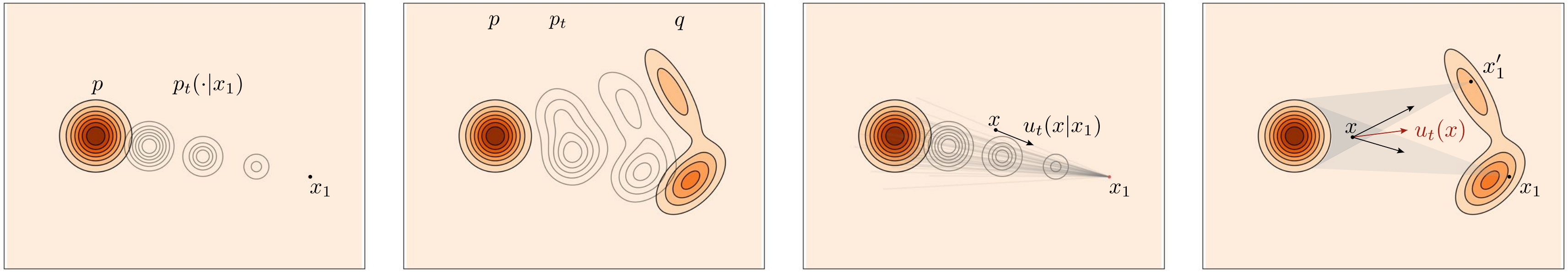
$$\frac{\boldsymbol{f}_{k+1}(\boldsymbol{z}_k) - \boldsymbol{z}_k}{\delta} = \boldsymbol{u}_k(\boldsymbol{z}_k)$$

Continuous time limit

- Let $\delta = 1/K$ and take $K \to \infty$. Then a composition of residual flows
$$\boldsymbol{f}_K \circ \boldsymbol{f}_{K-1} \circ \cdots \circ \boldsymbol{f}_1$$
- is given by an ODE
$$\frac{d\boldsymbol{z}_t}{dt} = \lim_{\delta\to 0} \frac{\boldsymbol{z}_{t+\delta} - \boldsymbol{z}_t}{\delta} = \lim_{\delta\to 0} \frac{\boldsymbol{f}_{t+\delta}(\boldsymbol{z}_t) - \boldsymbol{z}_t}{\delta} = \boldsymbol{u}_t(\boldsymbol{z}_t)$$
- where the flow of ODE $\boldsymbol{f}: [0,1]\times\mathbb{R}^d \to \mathbb{R}^d$ is defined s.t.,
$$\frac{d\boldsymbol{f}_t}{dt}(\boldsymbol{z}) = \boldsymbol{u}_t\big(\boldsymbol{f}_t(\boldsymbol{z})\big)$$
- I.e., $\boldsymbol{f}_t$ maps initial condition $\boldsymbol{z}_0$ to the ODE at time $t > 0$:
$$\boldsymbol{z}_t := \boldsymbol{f}_t(\boldsymbol{z}_0) = \boldsymbol{z}_0 + \int_0^t \boldsymbol{u}_s(\boldsymbol{z}_s)ds$$

Flow Matching (2022)

- New paradigms for generative modeling build on Continuous Normalizing Flow
- Present the notion of FM, a **simulation-free** approach for training CNFs based on regressing **vector fields** of fixed **conditional probability paths**



(a) Conditional probability path $p_t(x|x_1)$. **(b)** (Marginal) Probability path $p_t(x)$. **(c)** Conditional velocity field $u_t(x|x_1)$. **(d)** (Marginal) Velocity field $u_t(x)$.

Preliminaries: CNFs

- **Time-dependent vector field**

$$\boldsymbol{u}: [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$$

- Vector field $\boldsymbol{u}_t$ can be used to construct a time-dependent diffeomorphic map called flow $\psi: [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$ defined via ODE

$$\frac{d\psi_t}{dt}(\boldsymbol{x}) = \boldsymbol{u}_t\big(\psi_t(\boldsymbol{x})\big), \qquad \psi_0(\boldsymbol{x}) = \boldsymbol{x}$$

Preliminaries: CNFs

- Data space: $\mathbb{R}^d$
- **Probability density path**

$$p\colon [0,1]\times\mathbb{R}^d \to \mathbb{R}_+$$

- which is a time-dependent probability density function. I.e. $\int p_t(\boldsymbol{x})d\boldsymbol{x} = 1$ for any $t \in [0,1]$

- **Time-dependent vector field**

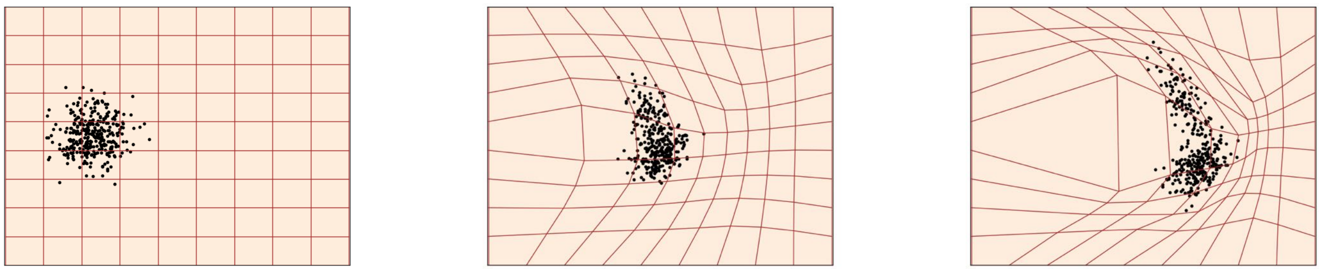
$$\boldsymbol{u}\colon [0,1]\times\mathbb{R}^d \to \mathbb{R}^d$$

- Vector field $\boldsymbol{u}_t$ can be used to construct a **time-dependent diffeomorphic** map called flow $\psi\colon [0,1]\times\mathbb{R}^d \to \mathbb{R}^d$ defined via ODE

$$\frac{d\psi_t}{dt}(\boldsymbol{x}) = \boldsymbol{u}_t\big(\psi_t(\boldsymbol{x})\big), \qquad \psi_0(\boldsymbol{x}) = \boldsymbol{x}$$

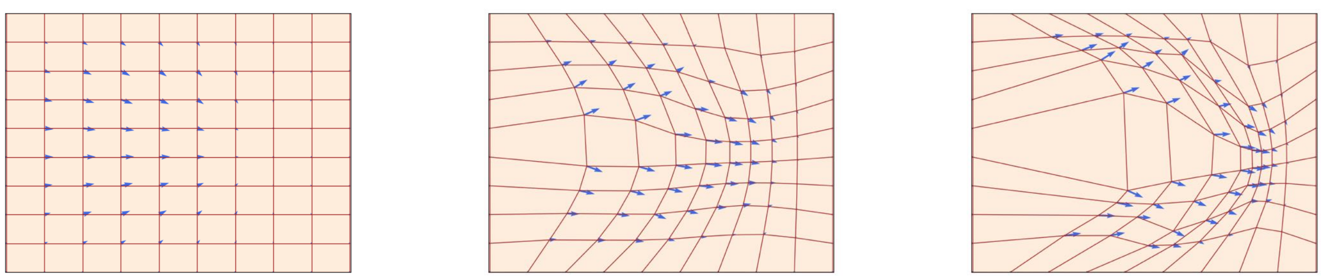
Preliminaries: CNFs

- A flow model $\boldsymbol{x}_t = \psi_t(\boldsymbol{x}_0)$ is defined by a diffeomorphism $\psi_t: \mathbb{R}^d \to \mathbb{R}^d$



$\boldsymbol{x}_0 \sim p_0(\boldsymbol{x}_0)$ $\psi_t(\boldsymbol{x}_0)$ $\psi_1(\boldsymbol{x}_0)$

- A flow $\psi_t: \mathbb{R}^d \to \mathbb{R}^d$ (square grid) is defined by a velocity field $\boldsymbol{u}_t: \mathbb{R}^d \to \mathbb{R}^d$ (blue arrows)



Equivalence between flows and velocity fields

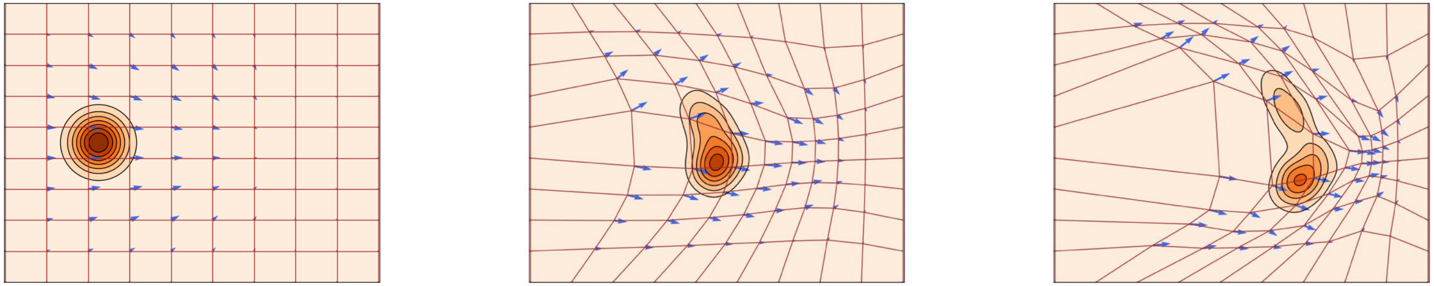
- A C^r flow $\psi: [0,1]\times\mathbb{R}^d \to \mathbb{R}^d$ can be defined in terms of a $C^r\left([0,1]\times\mathbb{R}^d, \mathbb{R}^d\right)$ velocity field $\boldsymbol{u}: [0,1]\times\mathbb{R}^d \to \mathbb{R}^d$ implementing $\boldsymbol{u}: (t, \boldsymbol{x}) \longmapsto \boldsymbol{u}_t(\boldsymbol{x})$ via the following ODE:
$$\frac{d\psi_t}{dt}(\boldsymbol{x}) = \boldsymbol{u}_t\big(\psi_t(\boldsymbol{x})\big), \qquad \psi_0(\boldsymbol{x}) = \boldsymbol{x}$$

- If $\boldsymbol{u}$ is $C^r\left([0,1]\times\mathbb{R}^d, \mathbb{R}^d\right)$, $r \geq 1$, then the ODE has a unique solution which is a $C^r\left(\Omega, \mathbb{R}^d\right)$ diffeomorphism ψ_t defined over an open set Ω which is a super-set of $\{0\}\times\mathbb{R}^d$

Preliminaries: CNFs

- Chen et al.(2018) suggested the modeling the vector field $\boldsymbol{v}_t$ with a neural network $\boldsymbol{v}_t(\boldsymbol{x}, \theta)$ where θ is learnable parameters
- $\boldsymbol{v}_t(\boldsymbol{x}, \theta)$ leads to a deep parametric model of the flow ψ_t (called CNF)
- CNF is used to reshape a simple prior p_0 to a more complicated one p_1 via push-forward equation

$$p_t(\boldsymbol{x}) = [\psi_t]_* p_0(\boldsymbol{x}) := p_0\left(\psi_t^{-1}(\boldsymbol{x})\right) \det\left[\frac{\partial \psi_t^{-1}}{\partial \boldsymbol{x}}(\boldsymbol{x})\right]$$



A velocity field $\boldsymbol{v}_t$ (in blue) generates a probability path p_t

Preliminaries: Loss of CNFs

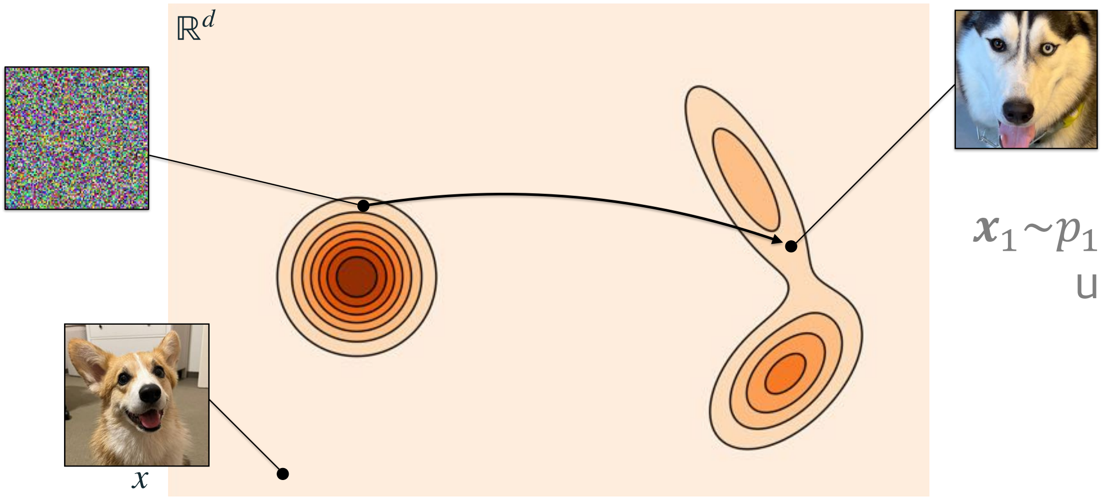
- For $\boldsymbol{x}_{data} = \psi_1(\boldsymbol{x}_0)$,

$$\mathcal{L}_{CNF} = -\log p\big(\psi_1(\boldsymbol{x}_0)\big) = -\log p_0(x_0) + \int_0^1 \nabla \cdot \boldsymbol{u}_t\big(\psi_t(\boldsymbol{x}_0)\big)dt$$

- CNFs define continuous probability density transformations using ordinary differential equations (ODEs)
- However, estimating the log-likelihood requires simulating these ODEs
- This simulation process is computationally expensive, slow, and results in slow inference

Flow Matching

$x_0 \sim p_0(x_0)$
Simple prior



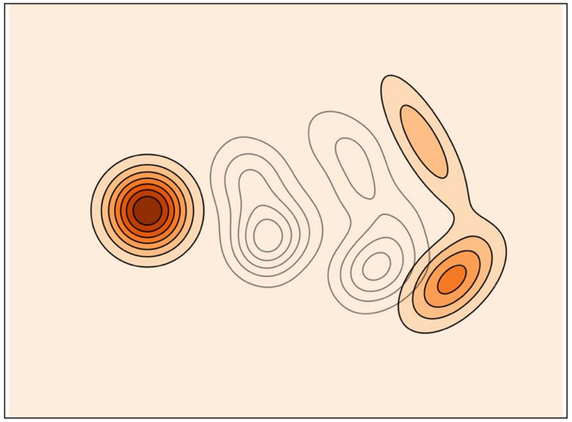
$x_1 \sim p_1(x_1) \approx p_{data}$
unknown

FLOW MATCHING FOR GENERATIVE MODELING
Yaron Lipman et al. ICLR 2023

Flow Matching

- Let p_t be a probability path s.t., p_0 is a simple prior (e.g., standard normal distribution) and let $p_1 \approx p_{data}$

$\boldsymbol{x}_0 \sim p_0(\boldsymbol{x}_0)$
Simple prior



$\boldsymbol{x}_1 \sim p_1(\boldsymbol{x}_1) \approx p_{data}$
unknown

- Probability path p_t is transformed through a time-dependent flow ψ_t or velocity field $\boldsymbol{u}_t$

The objective of FM

- The objective of FM is designed to match this target prob. path
- Given target prob. path $p_t(\boldsymbol{x})$ and corresponding vector field $\boldsymbol{u}_t(\boldsymbol{x})$ which generates $p_t(\boldsymbol{x})$

$$\mathcal{L}_{FM}(\theta) := E_{t\sim U[0,1]} E_{\boldsymbol{x}\sim p_t(\boldsymbol{x})} [\|\boldsymbol{v}_t(\boldsymbol{x};\theta) - \boldsymbol{u}_t(\boldsymbol{x})\|^2]$$

- where θ denotes the learnable parameters of the CNF vector field $\boldsymbol{v}_t$
- FM loss regresses the vector field $\boldsymbol{u}_t$ with a neural network $\boldsymbol{v}_t$
- This objective is intractable since we have no prior knowledge for what an appropriate $\boldsymbol{u}_t$ and p_t

Construction of $p_t, \boldsymbol{u}_t$

- Construct both p_t and $\boldsymbol{u}_t$ using probability paths and vector fields that are only defined **per sample**
- Given a particular data sample $\boldsymbol{x}_1$, we denote by $p_t(\boldsymbol{x}|\boldsymbol{x}_1)$ a conditional probability path s.t.,

$$p_0(\boldsymbol{x}|\boldsymbol{x}_1) = p_0(\boldsymbol{x})$$
$$p_1(\boldsymbol{x}|\boldsymbol{x}_1) = \text{a distribution concentrated around } \boldsymbol{x}_1$$

- E.g., $p_1(\boldsymbol{x}|\boldsymbol{x}_1) = N(\boldsymbol{x}|\boldsymbol{x}_1, \sigma^2\boldsymbol{I})$
- We will define $p_t(\boldsymbol{x}|\boldsymbol{x}_1)$, $0 < t < 1$ conditional probability path per sample $\boldsymbol{x}_1$

Construction of $p_t, \boldsymbol{u}_t$

- Marginalizing the conditional probability paths over p_{data} give rise the marginal probability path

$$p_t(\boldsymbol{x}) = \int p_t(\boldsymbol{x}|\boldsymbol{x}_1)p_{data}(\boldsymbol{x}_1)d\boldsymbol{x}_1$$

- At time $t = 1$, the marginal probability p_1 is a mixture distribution so that $p_1 \approx p_{data}$

$$p_1(\boldsymbol{x}) = \int p_1(\boldsymbol{x}|\boldsymbol{x}_1)p_{data}(\boldsymbol{x}_1)d\boldsymbol{x}_1 \approx p_{data}(\boldsymbol{x})$$

- Let $\boldsymbol{u}_t(\cdot\,|\boldsymbol{x}_1): \mathbb{R}^d \to \mathbb{R}^d$ be a conditional vector field that generates $p_t(\cdot\,|\boldsymbol{x}_1)$ (or conditional flow $\psi_t(\cdot\,|\boldsymbol{x}_1): \mathbb{R}^d \to \mathbb{R}^d$)

Construction of $p_t, \boldsymbol{u}_t$

- Define a marginal vector field by

$$\boldsymbol{u}_t(\boldsymbol{x}) = \int \boldsymbol{u}_t(\boldsymbol{x}|\boldsymbol{x}_1)\frac{p_t(\boldsymbol{x}|\boldsymbol{x}_1)p_{data}(\boldsymbol{x}_1)}{p_t(\boldsymbol{x})}d\boldsymbol{x}_1$$

- The marginal vector field $\boldsymbol{u}_t(\boldsymbol{x})$ generates the marginal probability path $p_t(\boldsymbol{x})$

Objective for Conditional Flow Matching

$$\mathcal{L}_{CFM}(\theta) := E_{t\sim U[0,1]} E_{\boldsymbol{x}_1 \sim p_{data}(\boldsymbol{x}_1)} E_{\boldsymbol{x}\sim p_t(\boldsymbol{x}|\boldsymbol{x}_1)} [\|\boldsymbol{v}_t(\boldsymbol{x};\theta) - \boldsymbol{u}_t(\boldsymbol{x}|\boldsymbol{x}_1)\|^2]$$

- Assume that $p_t(\boldsymbol{x}) > 0$ for all $\boldsymbol{x} \in \mathbb{R}^d$ and $t \in [0,1]$. Then
$$\operatorname*{argmin}_\theta \mathcal{L}_{FM}(\theta) = \operatorname*{argmin}_\theta \mathcal{L}_{CFM}(\theta)$$

Gaussian conditional probability paths

- The CMF works with any choice of conditional probability path and conditional vector field
- We will discuss the construction of $p_t(\boldsymbol{x}|\boldsymbol{x}_1)$ and $\boldsymbol{u}_t(\boldsymbol{x}|\boldsymbol{x}_1)$ for a general family of **Gaussian conditional probability paths**

Gaussian conditional probability paths

- The CMF works with any choice of conditional probability path and conditional vector field
- We will discuss the construction of $p_t(\boldsymbol{x}|\boldsymbol{x}_1)$ and $\boldsymbol{u}_t(\boldsymbol{x}|\boldsymbol{x}_1)$ for a general family of **Gaussian conditional probability paths**
- Consider conditional probability paths of the form

$$p_t(\boldsymbol{x}|\boldsymbol{x}_1) = N(\boldsymbol{x}|\boldsymbol{\mu}_t(\boldsymbol{x}_1), \sigma_t(\boldsymbol{x}_1)^2 \boldsymbol{I})$$

  - $\boldsymbol{\mu}$: $[0,1] \times \mathbb{R}^d \to \mathbb{R}^d$ is the time-dependent mean
  - σ: $[0,1] \times \mathbb{R}^d \to \mathbb{R}_+$ is the time-dependent scalar standard deviation
- Set $\boldsymbol{\mu}_0(\boldsymbol{x}_1) = \boldsymbol{0}$, $\sigma_0(\boldsymbol{x}_1) = 1$, so that $p_0(\boldsymbol{x}|\boldsymbol{x}_1) = N(\boldsymbol{x}|\boldsymbol{0}, \boldsymbol{I})$ and $\boldsymbol{\mu}_1(\boldsymbol{x}_1) = \boldsymbol{x}_1$, $\sigma_0(\boldsymbol{x}_1) = \sigma_{min}$

Gaussian conditional probability paths

- **Remark**: there is an infinite number of vector fields $\boldsymbol{u}_t(\boldsymbol{x}|\boldsymbol{x}_1)$ that generate any probability path

- Use the simplest vector field corresponding to a canonical transformation
- Consider the flow (conditioned on $\boldsymbol{x}_1$)
$$\psi_t(\boldsymbol{x}) \coloneqq \sigma_t(\boldsymbol{x}_1)\boldsymbol{x} + \boldsymbol{\mu}_t(\boldsymbol{x}_1)$$
- Since ψ_t is the affine transformation,
$$\psi_t(\boldsymbol{x}) = N(\boldsymbol{x}|\boldsymbol{\mu}_t(\boldsymbol{x}_1), \sigma_t(\boldsymbol{x}_1)^2\boldsymbol{I}), \qquad \text{when } \boldsymbol{x} \sim N(\boldsymbol{0}, \boldsymbol{I})$$
- It means that conditional flow ψ_t pushes the noise distribution $p_0(\boldsymbol{x}|\boldsymbol{x}_1) = p_0(\boldsymbol{x})$ to $p_t(\boldsymbol{x}|\boldsymbol{x}_1)$. I.e.,
$$[\psi_t]_* p_0(\boldsymbol{x}) = p_t(\boldsymbol{x}|\boldsymbol{x}_1)$$

Gaussian conditional probability paths

- This flow provides a vector field that generates the conditional probability path:

$$\frac{d\psi_t}{dt}(\boldsymbol{x}) = \boldsymbol{u}_t(\psi_t(\boldsymbol{x})|\boldsymbol{x}_1)$$

- $\boldsymbol{u}_t(\cdot\,|\boldsymbol{x}_1)$ will see later

- Reparametrize $p_t(\boldsymbol{x}|\boldsymbol{x}_1)$ in terms of $\boldsymbol{x}_0$. Then CFM loss $\mathcal{L}_{CFM}(\theta)$

$$E_{t\sim U[0,1]}E_{\boldsymbol{x}_1\sim p_{data}(\boldsymbol{x}_1)}E_{\boldsymbol{x}\sim p_t(\boldsymbol{x}|\boldsymbol{x}_1)}[\|\boldsymbol{v}_t(\boldsymbol{x};\theta) - \boldsymbol{u}_t(\boldsymbol{x}|\boldsymbol{x}_1)\|^2]$$

- can be written as

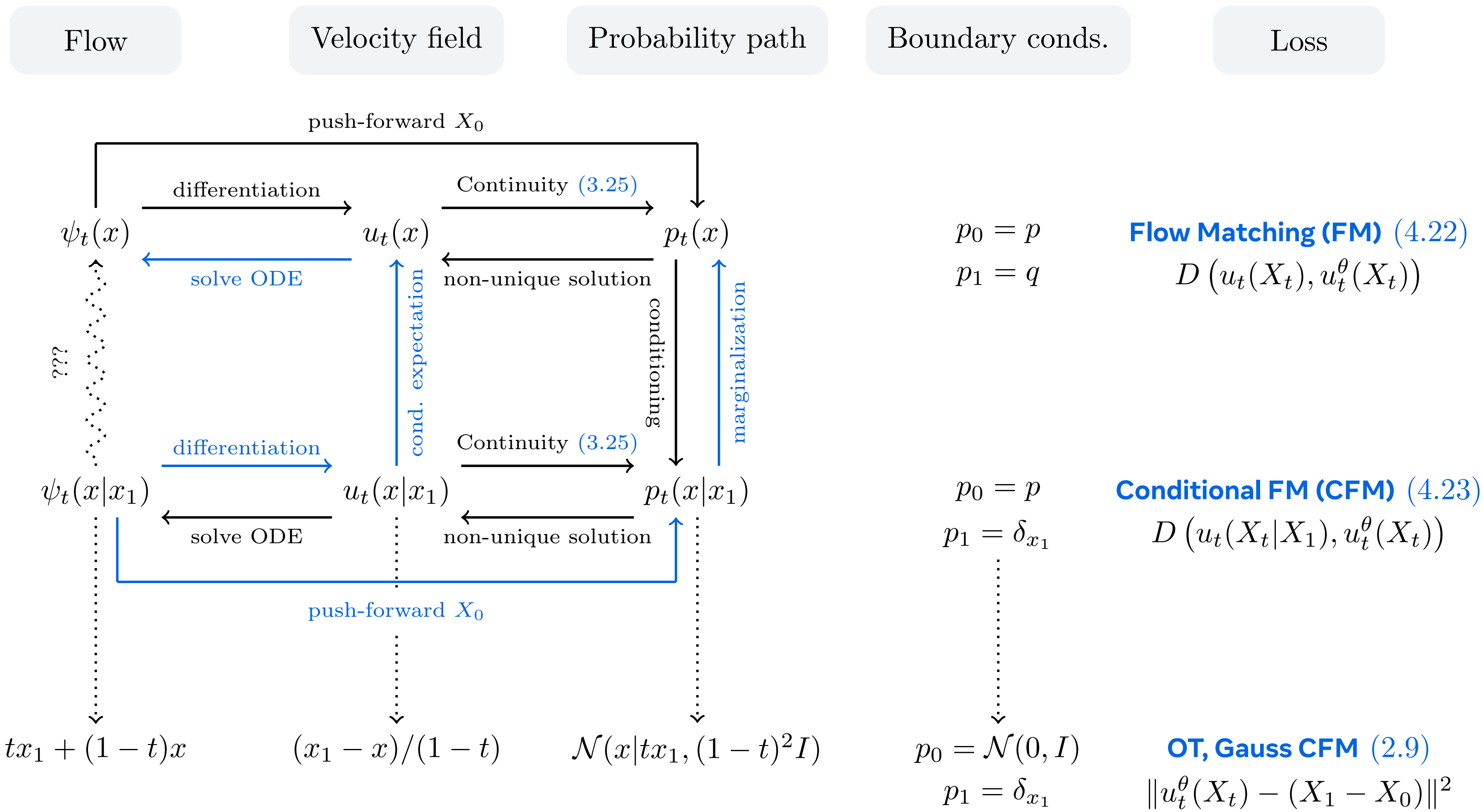
$$E_{t\sim U[0,1]}E_{\boldsymbol{x}_1\sim p_{data}(\boldsymbol{x}_1)}E_{\boldsymbol{x}_0\sim p_0(\boldsymbol{x}_0)}\left[\left\|\boldsymbol{v}_t(\psi_t(\boldsymbol{x}_0)) - \frac{d}{dt}\psi_t(\boldsymbol{x}_0)\right\|^2\right]$$

Gaussian conditional probability paths

- Let $p_t(\boldsymbol{x}|\boldsymbol{x}_1)$ be a Gaussian probability path. I.e., $p_t(\boldsymbol{x}|\boldsymbol{x}_1) = N(\boldsymbol{x}|\boldsymbol{\mu}_t(\boldsymbol{x}_1), \sigma_t(\boldsymbol{x}_1)^2\boldsymbol{I})$
- Let ψ_t be its corresponding flow map. I.e., $\psi_t(\boldsymbol{x}) \coloneqq \sigma_t(\boldsymbol{x}_1)\boldsymbol{x} + \boldsymbol{\mu}_t(\boldsymbol{x}_1)$
- Then the unique vector field $\boldsymbol{u}_t(\boldsymbol{x}|\boldsymbol{x}_1)$ that defines ψ_t has the form

$$\boldsymbol{u}_t(\boldsymbol{x}|\boldsymbol{x}_1) = \frac{\sigma_t'(\boldsymbol{x}_1)}{\sigma_t(\boldsymbol{x}_1)}[\boldsymbol{x} - \boldsymbol{\mu}_t(\boldsymbol{x}_1)] + \boldsymbol{\mu}_t'(\boldsymbol{x}_1)$$

Relation for Flow Matching



Thanks