Diffusion Models and SDEs

Lecture 2:
SDEs and how to manipulate them

Diffusion Models and SDEs

Lecture 2:
SDEs and how to manipulate them

Time Reversal and Score-based modelling

Probability Flow ODE and Flow Matching

(Lecture 3)

Conditioning SDEs via Doob's h-transform

Score-based generative modelling

Training: learn score $\color{red}{s_t}(x) := \nabla \log p_t(x)$ by

1. Sample from target $x_1 \sim p_1$
2. Run "forward" SDE to "noise" x_1 until it becomes "simple" $x_0 \sim p_0$
3. Minimize some loss $\propto \left\| \hat{s}_t(x) - \color{red}{s_t}(x) \right\|^2$

Inference: sample using "reverse" SDE or prob-flow ODE

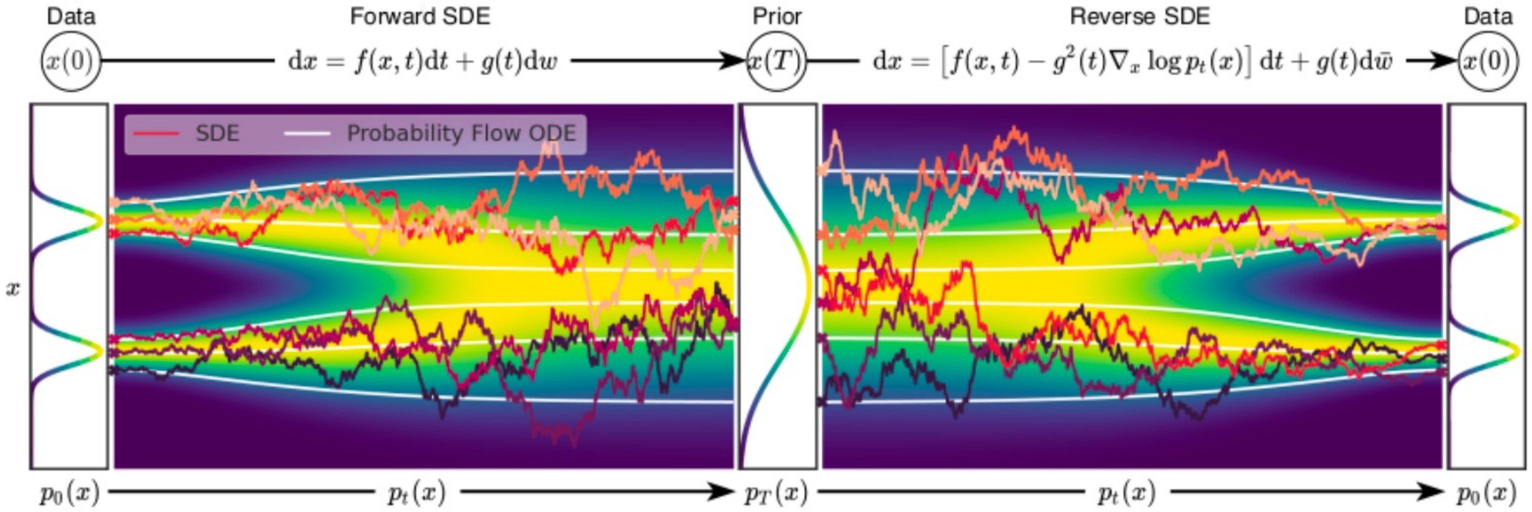


Figure 2 from Song et al. (2020)

Time Reversal - Chain Rule

A discrete time "heuristic" sketch

Via the chain rule we can decompose the joint in either direction,

$$p_{t|t+\delta}(x|y)p_{t+\delta}(y)=p_{t+\delta|t}(y|x)p_t(x)$$

Now consider an EM approx transition density, for the forward kernel:

$$p_{t+\delta|t}(y|x) = \mathcal{N}(y|x + f^+(x)\delta, \delta\sigma^2)$$

$$p_{t|t+\delta}(x|y) = ?$$

Time Reversal - Chain Rule

A discrete time "heuristic" sketch

Via the chain rule we can decompose the joint in either direction,

$$p_{t|t+\delta}(x|y)p_{t+\delta}(y)=p_{t+\delta|t}(y|x)p_t(x)$$

Now consider an EM approx transition density, for the forward kernel:

$$p_{t+\delta|t}(y|x) = \mathcal{N}(y|x + f^+(x)\delta, \delta\sigma^2)$$

$$p_{t|t+\delta}(x|y) = p_{t+\delta|t}(y|x)\frac{p_t(x)}{p_{t+\delta}(y)}$$

Time Reversal - Chain Rule

A discrete time "heuristic" sketch

Via Taylors Theorem we can expand time t marginal around y:

$$p_{t|t+\delta}(x|y) = p_{t+\delta|t}(y|x)\frac{p_t(y)e^{(x-y)^\top \nabla_y \ln p_t(y) + \mathcal{O}(\delta^2)}}{p_{t+\delta}(y)}$$

Assuming $|\ln p_t(x) - \ln p_s(x)| = \mathcal{O}(|t-s|^2)$

$$p_{t|t+\delta}(x|y) = p_{t+\delta|t}(y|x)e^{(x-y)^\top \nabla_y \ln p_t(y) + \mathcal{O}(\delta^2)}$$

Time Reversal - Chain Rule

A discrete time “heuristic” sketch

Regrouping and completing the square:

$$p_{t|t+\delta}(x|y) = \frac{e^{-\frac{||x-(y-f^{+}(y)\delta+\sigma^2\nabla_y \ln p_t(y)\delta||^2}{\sigma^2\delta}+\mathcal{O}(\delta^2)}}{\sqrt{2\pi}\delta^{d/2}\sigma^d}$$

Which corresponds to the Euler Maruyama discretization of the following SDE (seem familiar ?):

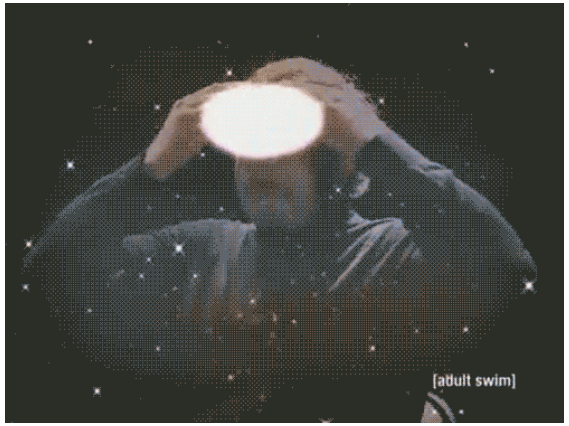
$$\mathrm{d}X_t = \left(-f^{+}(X_t, T-t) + \sigma^2\nabla_{X_t}\ln p_{T-t}(X_t)\right)\mathrm{d}t + \sigma \mathrm{d}W_t$$

Time Reversal - Chain Rule

A discrete time "heuristic" sketch

Inspecting the relationship between the drifts yields Nelsons duality formula:

$$f^{-}(x,t) + f^{+}(x,T-t) = \sigma^2 \nabla_x \ln p_{T-t}(x)$$



Time Reversal - Chain Rule

A discrete time "heuristic" sketch

Inspecting the relationship between the drifts yields Nelsons duality formula:

$$f^{-}(x, t) + f^{+}(x, T - t) = \sigma^2 \nabla_x \ln p_{T-t}(x)$$

Looks slightly different to Song et al. 2021, why ?

Time Reversal - Chain Rule

Nelsons Relation – Semantics Clarification

Looks slightly different to Song et al. 2020, why ?

Due to 2 equivalent ways of representing time reversals:

$$\mathrm{d}Y_t = f^+(Y_t, t)\mathrm{d}t + \sigma \mathrm{d}W_t$$

Forward SDE (e.g. De Bortoli 2021)

- Travels forward in time

$$\mathbf{dX_t = f^-(X_t, t)dt + \sigma dW_t}$$

$$\mathbf{f^-(x,t) + f^+(x, T-t) = \sigma^2 \nabla_x \ln p_{T-t}(x)}$$

- Flips / No longer the same joint

$$\mathrm{Law}(\mathbf{x_t})_{\mathbf{t=0}}^{\mathbf{T}} = \mathrm{Law}(\mathbf{y_{T-t}})_{\mathbf{t=0}}^{\mathbf{T}}$$

Backwards SDE (e.g. Song 2021)

- Travels Backwards in time

$$\mathbf{dX_t^- = f^-(X_t^-, t)dt + \sigma dW_t^-}$$

$$\mathbf{f^-(x,t) - f^+(x,t) = \sigma^2 \nabla_x \ln p_t(x)}$$

- Encodes the same joint

$$\mathrm{Law}(\mathbf{x_t})_{\mathbf{t=0}}^{\mathbf{T}} = \mathrm{Law}(\mathbf{y_t})_{\mathbf{t=0}}^{\mathbf{T}}$$

Time Reversal – Generative Modelling

Time reversing VP-SDE / OU Process [Song 2021, De Bortoli 2021]

Consider the time homogenous VP-SDE (OU Process):

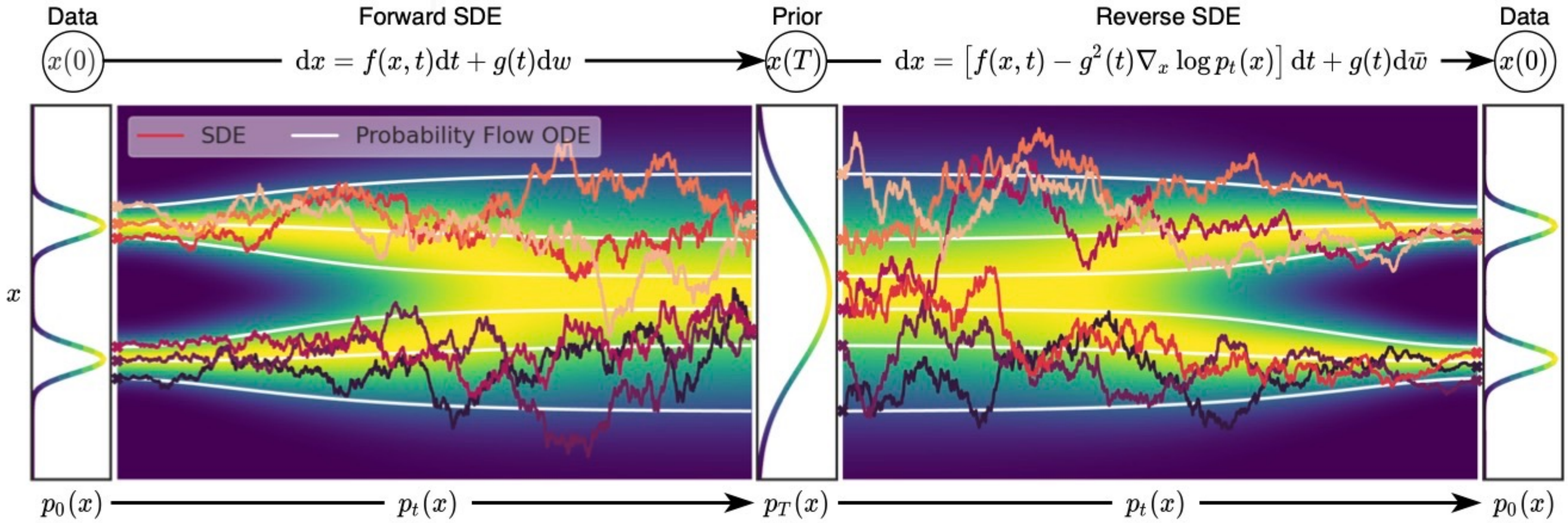
$$X_0 \sim p_{\mathrm{data}}$$
$$\mathrm{d}X_t = -\beta X_t \mathrm{d}t + \sqrt{2\beta}\mathrm{d}W_t$$

Then its time reversal $(Y_t)_{t=0}^T \stackrel{d}{=} (X_{T-t})_{t=0}^T$ satisfies the score SDE [Song 2021]:

$$Y_0 \sim p_T \approx \mathcal{N}(0, I)$$
$$\mathrm{d}Y_t = (\alpha Y_t + 2\alpha \nabla_{Y_t} \ln p_{T-t}(Y_t))\, \mathrm{d}t + \sqrt{2\alpha}\mathrm{d}B_t$$

Where $Y_T \sim p_{\mathrm{data}}$, thus we could instead sample approximately $Y_0 \sim \mathcal{N}(0, I)$ and have $\mathrm{Law} Y_T \approx p_{\mathrm{data}}$ following the mixing rate of the OU [De Bortoli 2021]

Probability flow ODE



Probability Flow ODE

Definition: Every stochastic process described by an SDE has a corresponding deterministic process described by an ODE that has the same marginal probability densities $\{p_t(x)\}_{t=0}^{T}$. This process is called the *probability flow ODE*. For a general SDE of the form $dX_t = \mu(X_t, t)dt + \sigma(X_t, t)dW_t$, the corresponding ODE is given by

$$dX_t = \left[\mu(X_t, t) - \frac{1}{2}\nabla_x[\sigma(X_t, t)\sigma(X_t, t)^T] - \frac{1}{2}\sigma(X_t, t)\sigma(X_t, t)^T \nabla_x \log p_t(x)\right]dt$$

Probability Flow ODE

$$\mathrm{d}X_t = \left[\mu(X_t, t) - \frac{1}{2}\nabla_x[\sigma(X_t, t)\sigma(X_t, t)^T] - \frac{1}{2}\sigma(X_t, t)\sigma(X_t, t)^T \nabla_x \log p_t(x)\right]\mathrm{d}t$$

$$\mathrm{d}X_t = [\mu(X_t, t) - \frac{1}{2}\sigma^2(t)\nabla_x \log p_t(X_t)]\mathrm{d}t$$

Probability Flow ODE

FPE

$$\frac{\partial}{\partial t}p_t(x) = -\frac{\partial}{\partial x}[\mu(x,t)p_t(x)] + \frac{\partial}{\partial x}\frac{\partial}{\partial x}[\frac{1}{2}\sigma(x,t)\sigma(x,t)^T p_t(x)]$$

Product Rule

$$\frac{\partial}{\partial t}p_t(x) = -\frac{\partial}{\partial x}[\mu(x,t)p_t(x)] + \frac{1}{2}\frac{\partial}{\partial x}\left[\nabla_x[\sigma(x,t)\sigma(x,t)^T]p_t(x)\right] + \frac{1}{2}\frac{\partial}{\partial x}\left[\sigma(x,t)\sigma(x,t)^T\frac{\partial p_t(x)}{\partial x}\right]$$

Log Der. Trick

$$\frac{\partial}{\partial t}p_t(x) = -\frac{\partial}{\partial x}[\mu(x,t)p_t(x)] + \frac{1}{2}\frac{\partial}{\partial x}\left[\nabla_x[\sigma(x,t)\sigma(x,t)^T]p_t(x)\right] + \frac{1}{2}\frac{\partial}{\partial x}\left[\sigma(x,t)\sigma(x,t)^T p_t(x)\nabla_x \log p_t(x)\right]$$

Probability Flow ODE

$$\frac{\partial}{\partial t} p_t(x) = -\frac{\partial}{\partial x}[\mu(x,t) p_t(x)] + \frac{1}{2}\frac{\partial}{\partial x}\left[\nabla_x[\sigma(x,t)\sigma(x,t)^T] p_t(x)\right] + \frac{1}{2}\frac{\partial}{\partial x}\left[\sigma(x,t)\sigma(x,t)^T p_t(x) \nabla_x \log p_t(x)\right]$$

Pull der. And p(x) out

$$\frac{\partial}{\partial t} p_t(x) = -\frac{\partial}{\partial x}\left[\mu(x,t) - \frac{1}{2}\nabla_x[\sigma(x,t)\sigma(x,t)^T] - \frac{1}{2}\sigma(x,t)\sigma(x,t)^T \nabla_x[\log p_t(x)]\right]$$

$$\frac{\partial}{\partial t} p_t(x) = -\frac{\partial}{\partial x}\tilde{\mu}(x,t) p_t(x)$$

Generative Modelling

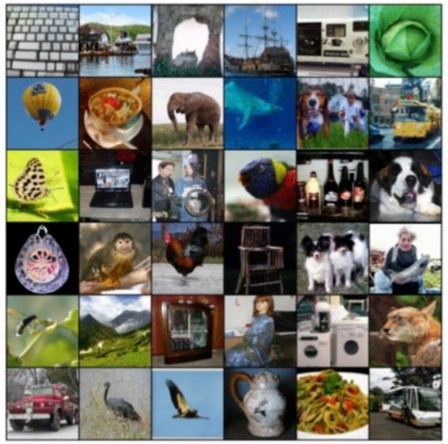
Given samples $x_1, \ldots, x_n$ with $x \sim p$

Infer $\hat{p}$ such that $\hat{p} \approx p$.

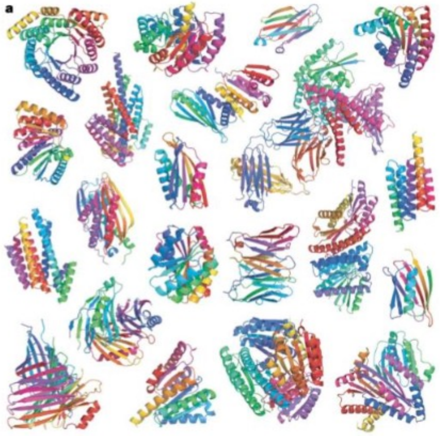
Tasks

Evaluate the likelihood of new data $\hat{p}(x)$

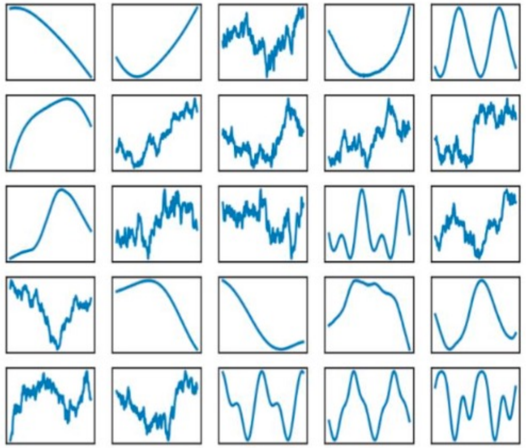
Sample $x \sim \hat{p}$



Images



Proteins



Functions

Generative Modelling

$$\max_{\theta} \mathbb{E}_{x \sim p_{data}} \log p_{\theta}(x)$$

Generative Modelling

$$\max_{\theta} \mathbb{E}_{x \sim p_{data}} \log p_\theta(x)$$

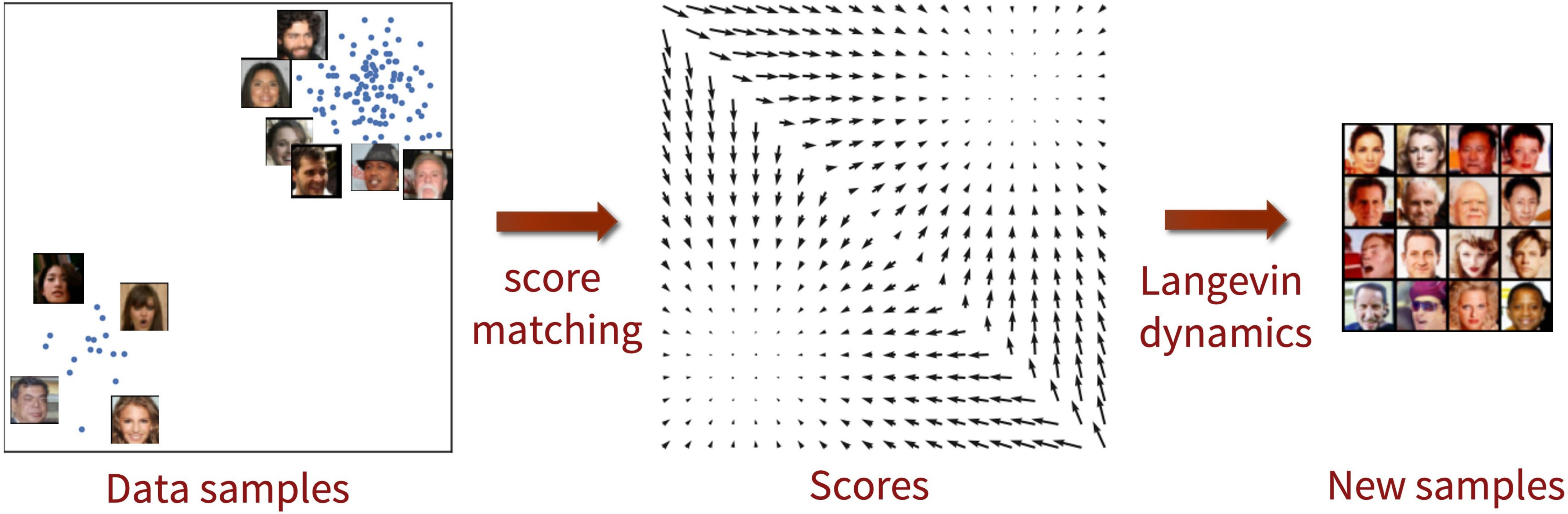
$$\begin{aligned} s_\theta(x) &= \nabla_x \log p_\theta(x) \\ &= \nabla_x \log \left(\frac{f_\theta(x)}{Z_\theta}\right) \\ &= \nabla_x \log f_\theta(x) - \underbrace{\nabla_x \log Z_\theta}_{=0} \\ &= \nabla_x \log f_\theta(x) \end{aligned}$$

How to train and simulate?

$$\mathbb{E}_{p(\mathbf{x})}[\|\nabla_{\mathbf{x}} \log p(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x})\|_2^2]$$

$$\mathbb{E}_{p(\mathbf{x})}[\|\nabla_{\mathbf{x}} \log p(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x})\|_2^2] = \int p(\mathbf{x})\|\nabla_{\mathbf{x}} \log p(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x})\|_2^2 \mathrm{d}\mathbf{x}.$$

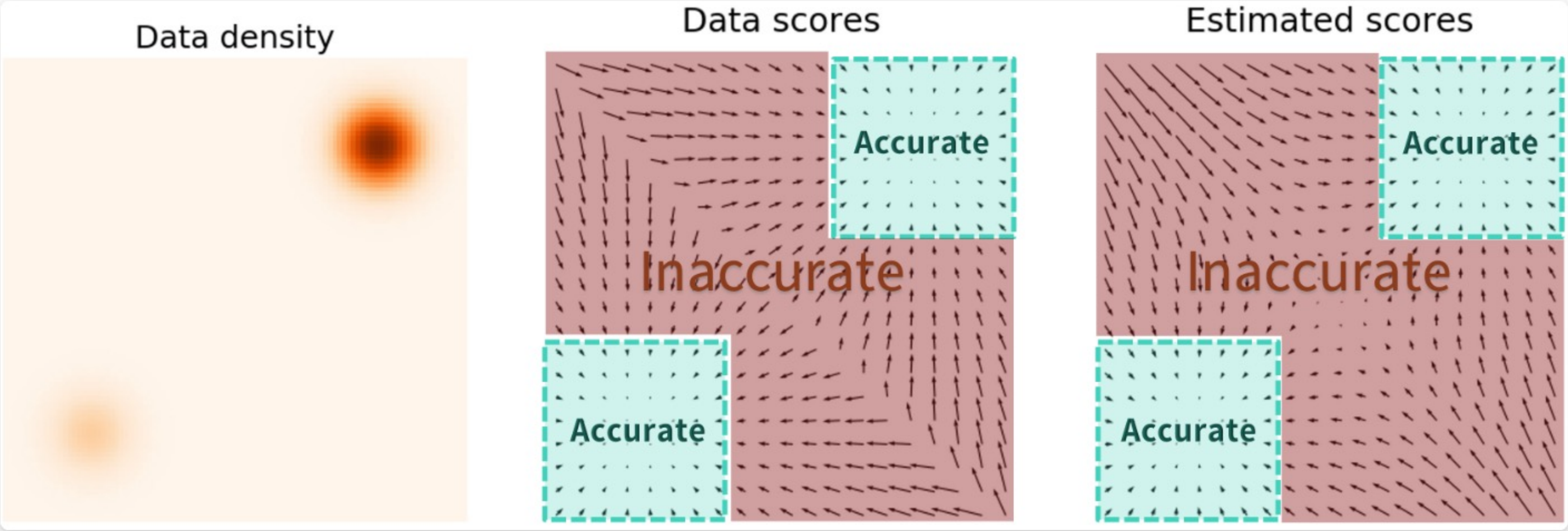
Naïve Score Matching



$$\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\} \overset{\text{i.i.d.}}{\sim} p(\mathbf{x})$$

$$\mathbf{s}_\theta(\mathbf{x}) \approx \nabla_\mathbf{x} \log p(\mathbf{x})$$

Naïve Score Matching

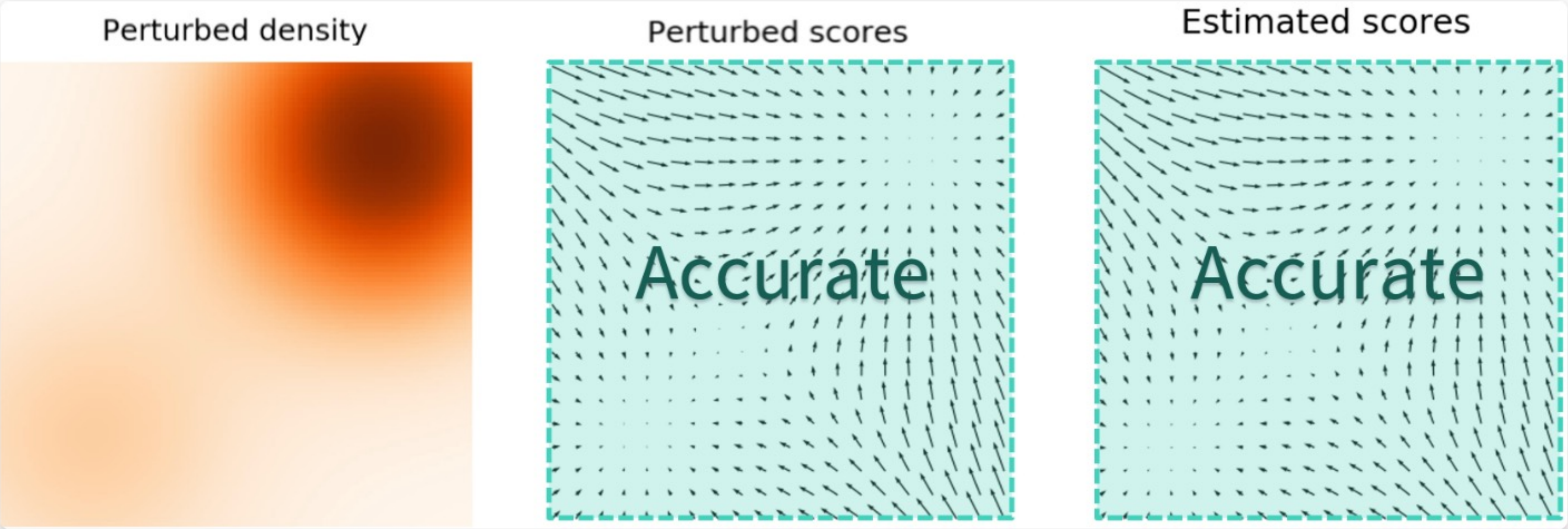


How to train and simulate?

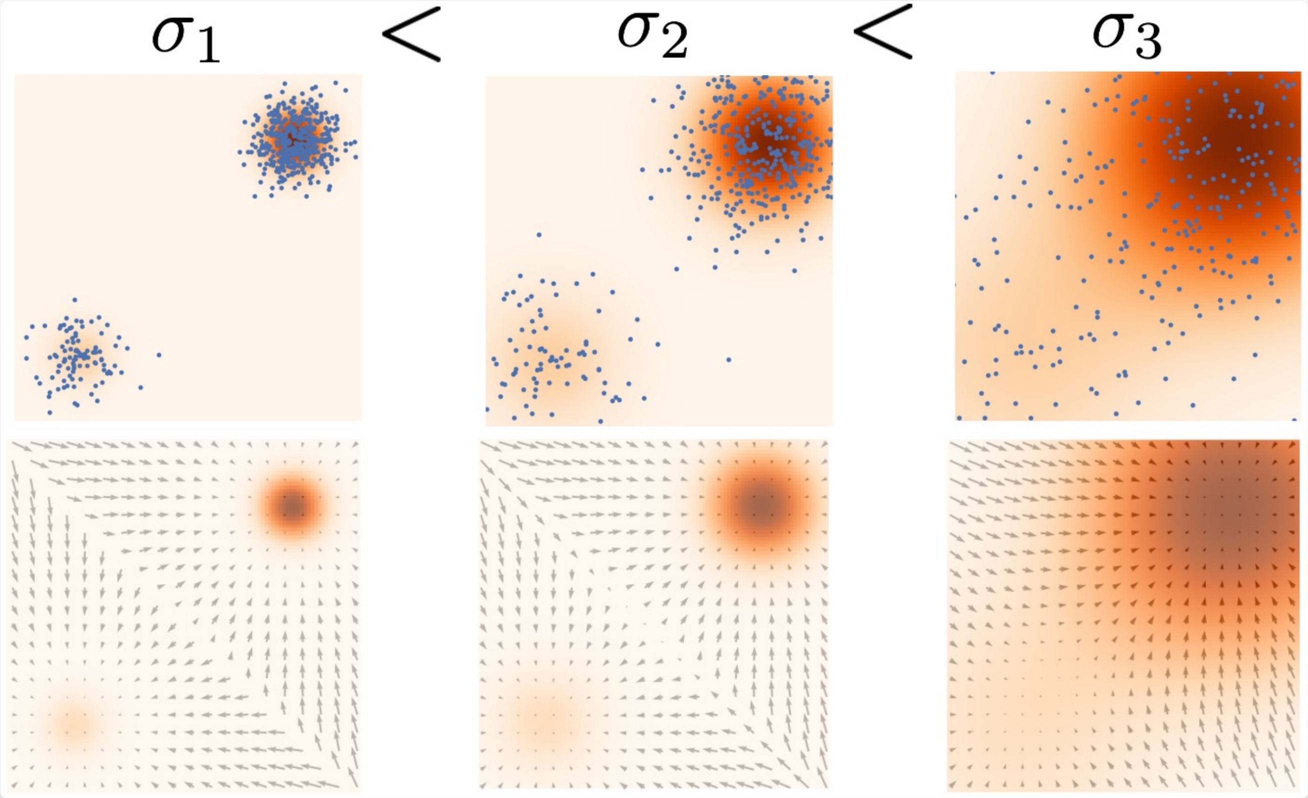
$$\mathbb{E}_{p(\mathbf{x})}[\|\nabla_{\mathbf{x}} \log p(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x})\|_2^2]$$

$$\mathbb{E}_{t \in \mathcal{U}(0,T)} \mathbb{E}_{p_t(\mathbf{x})}[\lambda(t) \|\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x}, t)\|_2^2],$$

Denoising Score Matching



Denoising Score Matching



We apply multiple scales of Gaussian noise to perturb the data distribution (**first row**), and jointly estimate the score functions for all of them (**second row**).

How to train and simulate?

$$\mathbb{E}_{p(\mathbf{x})}[\|\nabla_{\mathbf{x}} \log p(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x})\|_2^2]$$

$$\mathbb{E}_{t \in \mathcal{U}(0,T)} \mathbb{E}_{p_t(\mathbf{x})}[\lambda(t) \|\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x}, t)\|_2^2],$$

$$\begin{aligned} \Delta\mathbf{x} &\leftarrow [\mathbf{f}(\mathbf{x}, t) - g^2(t)\mathbf{s}_\theta(\mathbf{x}, t)]\Delta t + g(t)\sqrt{|\Delta t|}\mathbf{z}_t \\ \mathbf{x} &\leftarrow \mathbf{x} + \Delta\mathbf{x} \\ t &\leftarrow t + \Delta t, \end{aligned}$$

Tractable Score matching loss

$$s^* = \underset{s-\text{is measurable}}{\arg\min} \; \mathbb{E}\left[\int_0^T \left|\left|\nabla \ln p_{t|0}(X_t|X_0) - s(t, X_t)\right|\right|^2 \mathrm{d}t\right]$$

Tractable Score matching loss

$$s^* = \underset{s-\text{is measurable}}{\arg\min}\ \mathbb{E}\left[\int_0^T \left|\left|\nabla \ln p_{t|0}(X_t|X_0) - s(t, X_t)\right|\right|^2 \mathrm{d}t\right]$$

$$s^*(t, x) = \mathbb{E}_{X_0|X_t}[\nabla \ln p_{t|0}(X_t|X_0)|X_t = x]$$

Tractable Score matching loss

$$s^* = \underset{s-\text{is measurable}}{\arg\min} \, \mathbb{E}\left[\int_0^T \left|\left|\nabla \ln p_{t|0}(X_t|X_0) - s(t, X_t)\right|\right|^2 \mathrm{d}t\right]$$

$$s^*(t, x) = \mathbb{E}_{X_0|X_t}[\nabla \ln p_{t|0}(X_t|X_0)|X_t = x]$$

$$s^*(t, x) = \int p_{0|t}(x_0|x) \nabla \ln p_{t|0}(x|x_0) \mathrm{d}x_0$$

Tractable Score matching loss

$$s^* = \underset{s-\text{is measurable}}{\arg\min} \; \mathbb{E}\left[\int_0^T \left|\left|\nabla \ln p_{t|0}(X_t|X_0) - s(t, X_t)\right|\right|^2 \mathrm{d}t\right]$$

$$s^*(t, x) = \mathbb{E}_{X_0|X_t}[\nabla \ln p_{t|0}(X_t|X_0)|X_t = x]$$

$$s^*(t, x) = \int p_{0|t}(x_0|x)\nabla \ln p_{t|0}(x|x_0)\mathrm{d}x_0$$

$$s^*(t, x) = \int \frac{p_{t|0}(x|x_0)p_0(x_0)}{p_t(x)}\nabla \ln p_{t|0}(x|x_0)\mathrm{d}x_0$$

Tractable Score matching loss

$$s^*(t,x) = \int \frac{p_{t|0}(x|x_0)p_0(x_0)}{p_t(x)} \nabla \ln p_{t|0}(x|x_0) \mathrm{d}x_0$$

Tractable Score matching loss

$$s^*(t, x) = \int \frac{p_{t|0}(x|x_0)p_0(x_0)}{p_t(x)} \nabla \ln p_{t|0}(x|x_0) \mathrm{d}x_0$$

$$s^*(t, x) = \frac{1}{p_t(x)} \int p_0(x_0) \nabla p_{t|0}(x|x_0) \mathrm{d}x_0$$

Tractable Score matching loss

$$s^*(t,x) = \int \frac{p_{t|0}(x|x_0)p_0(x_0)}{p_t(x)} \nabla \ln p_{t|0}(x|x_0) \mathrm{d}x_0$$

$$s^*(t,x) = \frac{1}{p_t(x)} \int p_0(x_0) \nabla p_{t|0}(x|x_0) \mathrm{d}x_0$$

$$s^*(t,x) = \frac{1}{p_t(x)} \nabla \int p_0(x_0) p_{t|0}(x|x_0) \mathrm{d}x_0$$

Tractable Score matching loss

$$s^*(t,x) = \int \frac{p_{t|0}(x|x_0)p_0(x_0)}{p_t(x)} \nabla \ln p_{t|0}(x|x_0)\mathrm{d}x_0$$

$$s^*(t,x) = \frac{1}{p_t(x)} \int p_0(x_0)\nabla p_{t|0}(x|x_0)\mathrm{d}x_0$$

$$s^*(t,x) = \frac{1}{p_t(x)} \nabla \int p_0(x_0)p_{t|0}(x|x_0)\mathrm{d}x_0$$

$$s^*(t,x) = \frac{1}{p_t(x)} \nabla p_t(x) = \nabla_x \ln p_t(x)$$

Conditional Flow Matching

Diffusion Process

$$d\mathbf{x}_t = f_t(\mathbf{x}_t)dt + g_t(\mathbf{x}_t)d\mathbf{w}$$

→

Closed-form conditional probability

$$p(\mathbf{x}_t \mid \mathbf{x}_0)$$

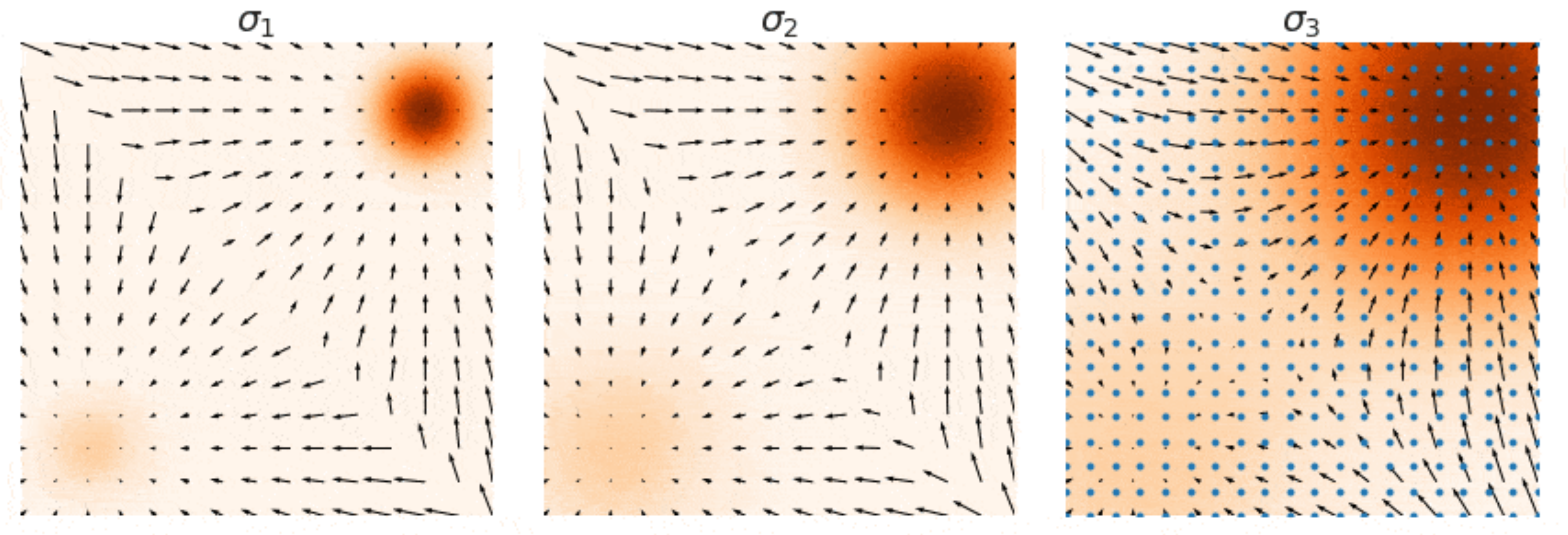
Training

$$\left\| s_\theta(\mathbf{x}_t) - \nabla \log p_t(\mathbf{x}_t \mid \mathbf{x}_0) \right\|^2$$

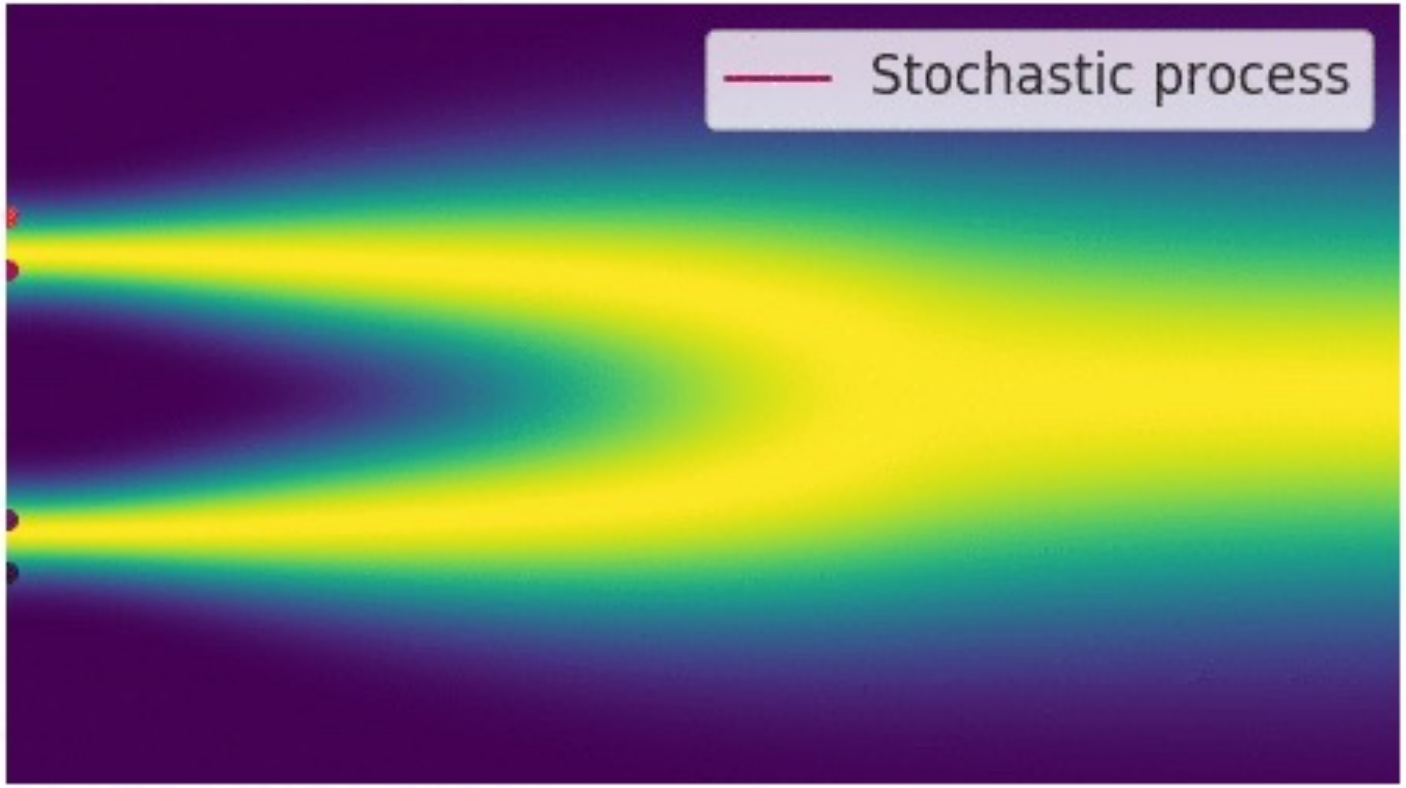
Sampling

$$\dot{\mathbf{x}}_t = f_t(\mathbf{x}_t) - \frac{1}{2} g_t^2 s_\theta(\mathbf{x}_t)$$

Denoising Score Matching



Denoising Score Matching



Key Equations for Score-based Modelling

"Forward" SDE

$$\mathrm{d}\overrightarrow{x}_t = f(\overrightarrow{x}_t, t)\,\mathrm{d}t + g(t)\,\mathrm{d}W \quad \textbf{with} \quad \overrightarrow{x}_0 \sim p_1$$

"Reverse" SDE

$$\mathrm{d}\overleftarrow{x}_t = \left(f(\overleftarrow{x}_t, t) - g(t)^2 \nabla \log p_t(\overleftarrow{x}_t)\right)\mathrm{d}t + g(t)\,\mathrm{d}W \quad \textbf{with} \quad \overleftarrow{x}_0 \sim p_0$$

Probability flow ODE

$$\mathrm{d}x_t = \left[f(x_t, t) - \frac{1}{2} g(t)^2 \nabla \log p_t(x_t)\right]\mathrm{d}t \quad \textbf{with} \quad x_0 \sim p_0$$

Score-based generative modelling

Training: learn score $s_t(x) := \nabla \log p_t(x)$ by

1. Sample from target $x_1 \sim p_1$
2. Run "forward" SDE to "noise" x_1 until it becomes "simple" $x_0 \sim p_0$
3. Minimize some loss $\propto \left\| \hat{s}_t(x) - s_t(x) \right\|^2$

Inference: sample using "reverse" SDE or prob-flow ODE

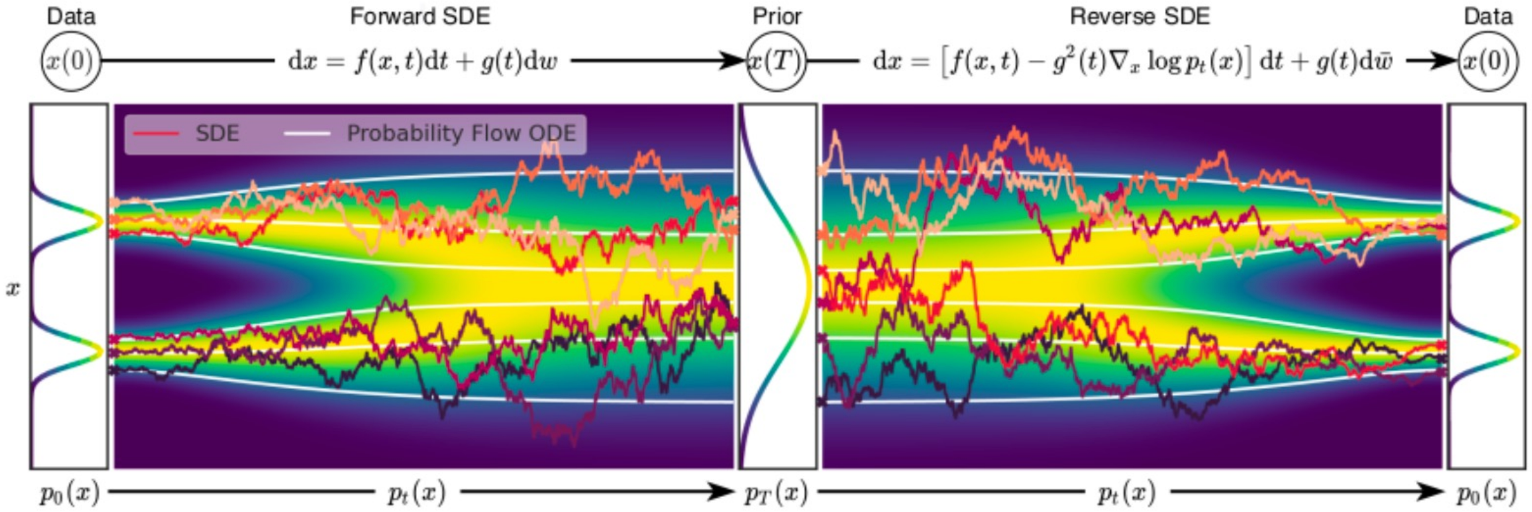
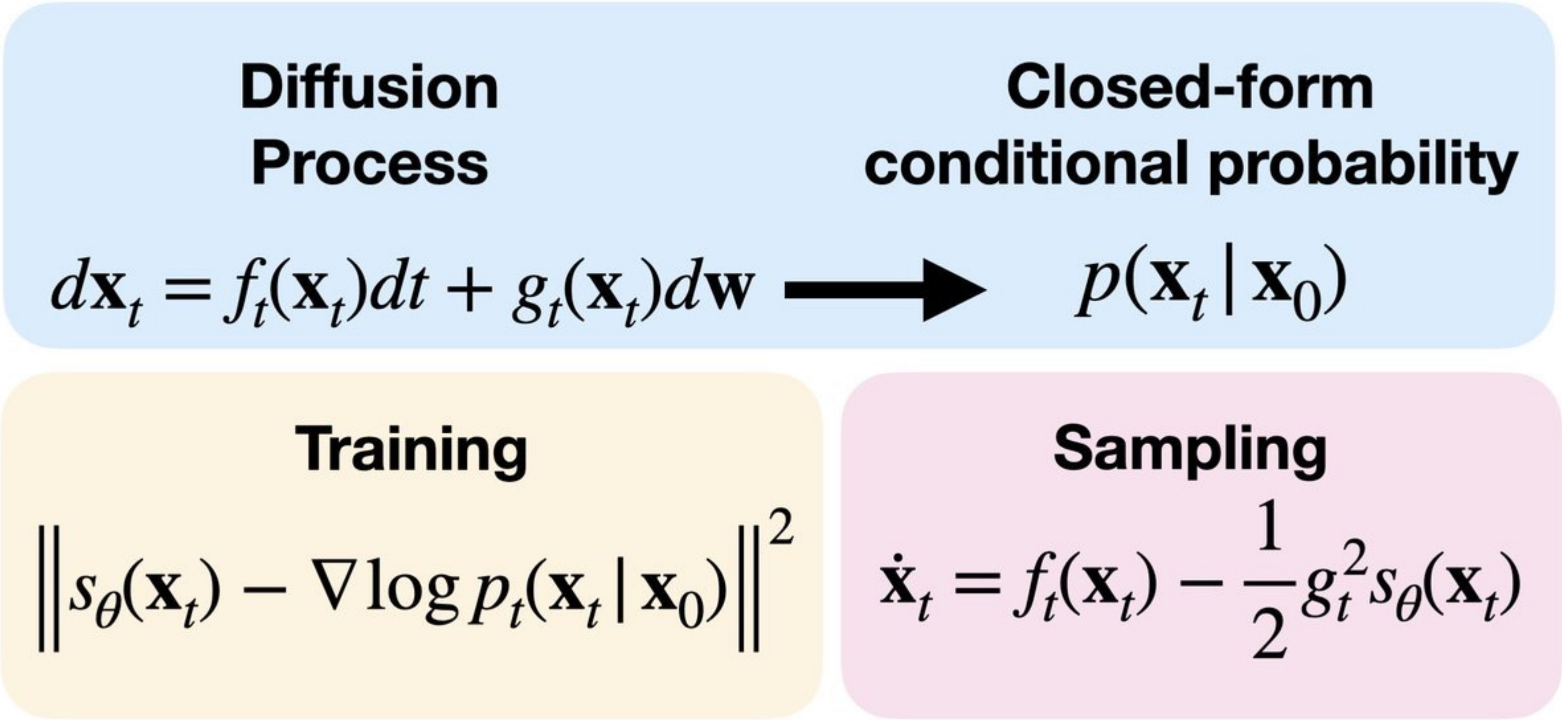


Figure 2 from Song et al. (2020)

Conditional Flow Matching



Conditional Flow Matching

General conditional probability path

$$p(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}(\mu_t(\mathbf{x}_0), \sigma_t^2(\mathbf{x}_0)I)$$

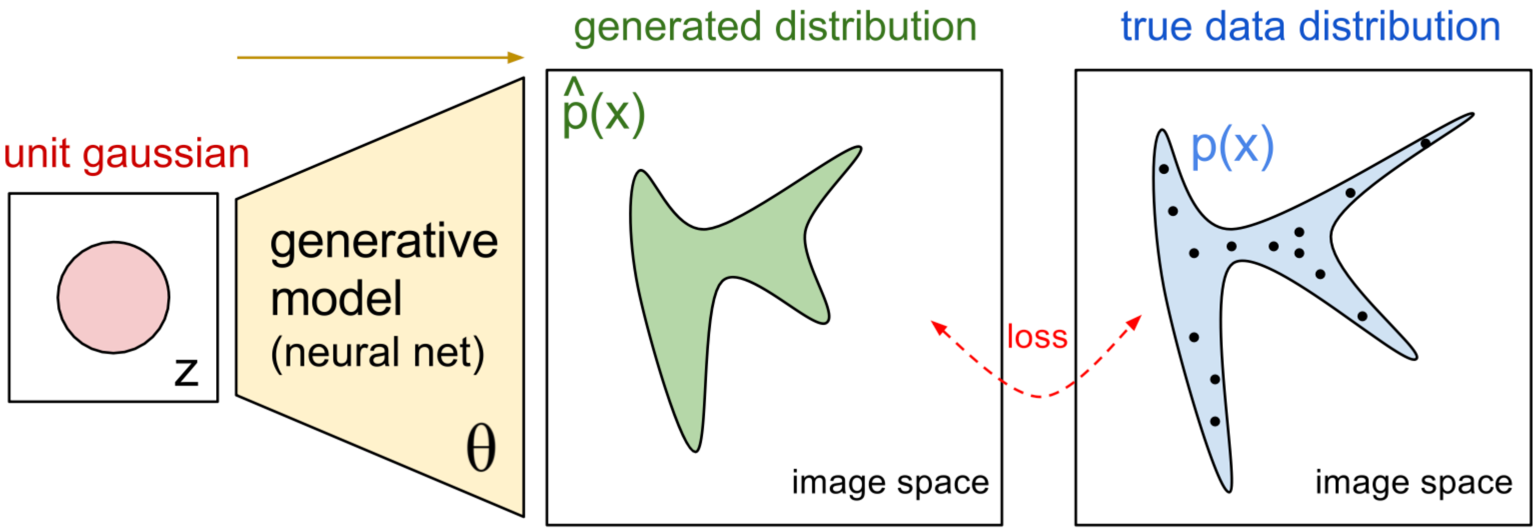
Training

$$\left\| v_\theta(\mathbf{x}_t) - u_t(\mathbf{x}_t \mid \mathbf{x}_0) \right\|^2$$

Sampling

$$\dot{\mathbf{x}}_t = v_\theta(\mathbf{x}_t)$$

Normalising Flows



Let p_0 be a simple distribution on $\mathbb{R}^d$ and $\phi : \mathbb{R}^d \to \mathbb{R}^d$ a transformation (diffeomorphism).

Let p_1 be the distribution of moving the samples of p_0 along $x_1 = \phi(x_0)$.

$$p_1(x_1) = p_0(x_0)\left|\frac{\partial \phi}{\partial x_0}(x_0)\right|^{-1}, \quad \text{where} \quad x_0 = \phi^{-1}(x_1)$$

Normalising Flows

Parameterise the transformation by a deep neural network $\hat{\phi}$.

Maximum log likelihood objective:

$$\mathbb{E}_{x\sim\mathcal{D}}\left[\log p_1(x)\right] = \mathbb{E}\left[\log p_0(x_0) - \log\left|\frac{\partial\hat{\phi}}{\partial x_0}(x_0)\right|\right], \quad \text{where} \quad x_0 = \hat{\phi}^{-1}(x)$$

Challenges Requires computation of inverse ϕ^{-1} and the determinant of Jacobian $\left|\frac{\partial\phi}{\partial z}\right|$

Continuous Normalising Flows (CNFs)

Use multiple 'residual' transformations

$$x = (u_N \circ u_{N-1} \ldots \circ u_1)(x_0).$$

For $N \to \infty$, the flow ϕ_t describes the position of a starting point x_0 along the vector field u_t, defined via an ODE

$$\frac{\mathrm{d}x_t}{\mathrm{d}t} = u_t(x_t),$$

where $x_t = \phi_t(x_0)$ and $u_t : \mathbb{R}^d \to \mathbb{R}^d$ is the tangent field of the flow.

The transformation is then the solution to

$$x \triangleq \phi_1(x_0) = x_0 + \int_0^1 u_t(x_t)\mathrm{d}t.$$

Continuous change-in-variables

Fokker–Planck equation without the diffusion term:

$$\log p_t(x) = \log p_0(x_0) - \int_0^t (\nabla \cdot u_t)(x_t)\mathrm{d}t$$

Maximum likelihood training of Continuous Normalising Flows require:

- expensive numerical ODE simulations
- estimators for the divergence.

Flow Matching

Integral-free approach to training CNF models

Supervised regression objective

Let u_t be a vector field that generates p_t, then we parameterise a neural net $\hat{u} : \mathbb{R}_+ \times \mathbb{R}^{\mathrm{d}} \to \mathbb{R}^{\mathrm{d}}$ and want to learn it using

$$\mathcal{L} = \mathbb{E}_{t,p_t(x)}\left[\|\hat{u}(t,x) - u_t(x)\|^2\right]$$

Challenges:

How do we ensure $p_1 \approx p$

What should p_0 be?

What is u_t?

Many names – same idea

Flow Matching for Generative Modeling

Yaron Lipman[1,2] **Ricky T. Q. Chen**[1] **Heli Ben-Hamu**[2] **Maximilian Nickel**[1] **Matt Le**[1]
[1]Meta AI (FAIR) [2]Weizmann Institute of Science

Many names – same idea

FLOW MATCHING FOR GENERATIVE MODELING

Yaron Lipman[1,2] **Ricky T. Q. Chen**[1] **Heli Ben-Hamu**[2] **Maximilian Nickel**[1] **Matt Le**[1]
[1]Meta AI (FAIR) [2]Weizmann Institute of Science

BUILDING NORMALIZING FLOWS WITH STOCHASTIC INTERPOLANTS

Michael S. Albergo
Center for Cosmology and Particle Physics
New York University
New York, NY 10003, USA
albergo@nyu.edu

Eric Vanden-Eijnden
Courant Institute of Mathematical Sciences
New York University
New York, NY 10012, USA
eve2@cims.nyu.edu

Many names – same idea

FLOW MATCHING

Yaron Lipman[1,2] **Ricky T. Q.**
[1]Meta AI (FAIR) [2]Weizmann I

BUILDING NORMA INTERPOLANTS

Michael S. Albergo
Center for Cosmology and Particle Physics
New York University
New York, NY 10003, USA
albergo@nyu.edu

Eric Vanden-Eijnden
Courant Institute of Mathematical Sciences
New York University
New York, NY 10012, USA
eve2@cims.nyu.edu

Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow

Xingchao Liu*
University of Texas at Austin
xcliu@utexas.edu

Chengyue Gong*
University of Texas at Austin
cygong@cs.utexas.edu

Qiang Liu
University of Texas at Austin
lqiang@cs.utexas.edu

Many names – same idea

FLOW MATCHING

Yaron Lipman[1,2] **Ricky T. Q.**
[1]Meta AI (FAIR) [2]Weizmann I

BUILDING NORMA INTERPOLANTS

Michael S. Albergo
Center for Cosmology and Particle Ph
New York University
New York, NY 10003, USA
albergo@nyu.edu

Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow

Xingchao Liu*
University of Texas at Austin
xcliu@utexas.edu

Chengyue Gong*
University of Texas at Austin
cygong@cs.utexas.edu

Qiang Liu
University of Texas at Austin
lqiang@cs.utexas.edu

Action Matching: Learning Stochastic Dynamics from Samples

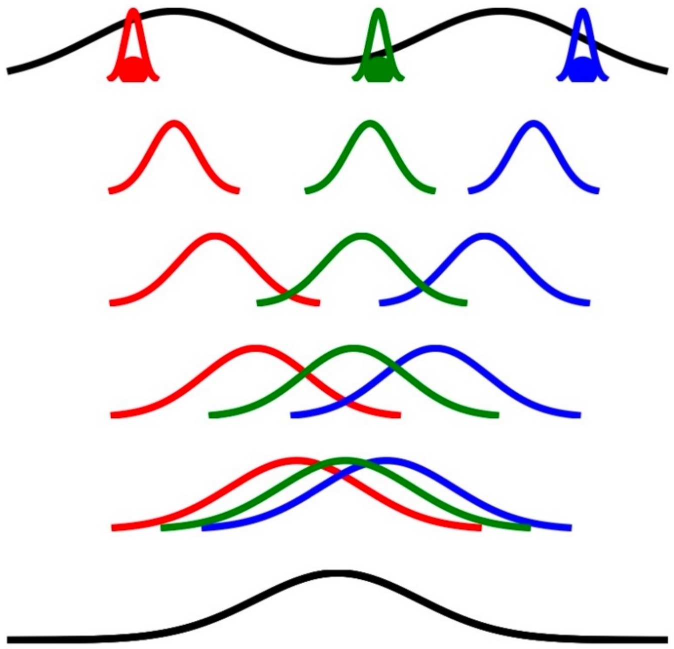
Kirill Neklyudov [1] **Rob Brekelmans** [1] **Daniel Severo** [1 2] **Alireza Makhzani** [1 2]

Conditional Flow Matching

Target probability path: as mixture of simpler probability paths
$p_t = \int p_t(\cdot|x_1)p(x_1)\mathrm{d}x_1$

Conditional probability path $p_t(\cdot|x_1)$ s.t. $p_1(\cdot|x_1) = \delta_{x_1}$ and $p_0(\cdot|x_1) = p_0$

Recover data distribution
$p_1(x) = \int p_1(x|x_1)q(x_1)\mathrm{d}x_1 = \int \delta_{x_1}(x)q(x_1)\mathrm{d}x_1 = q_1(x)$

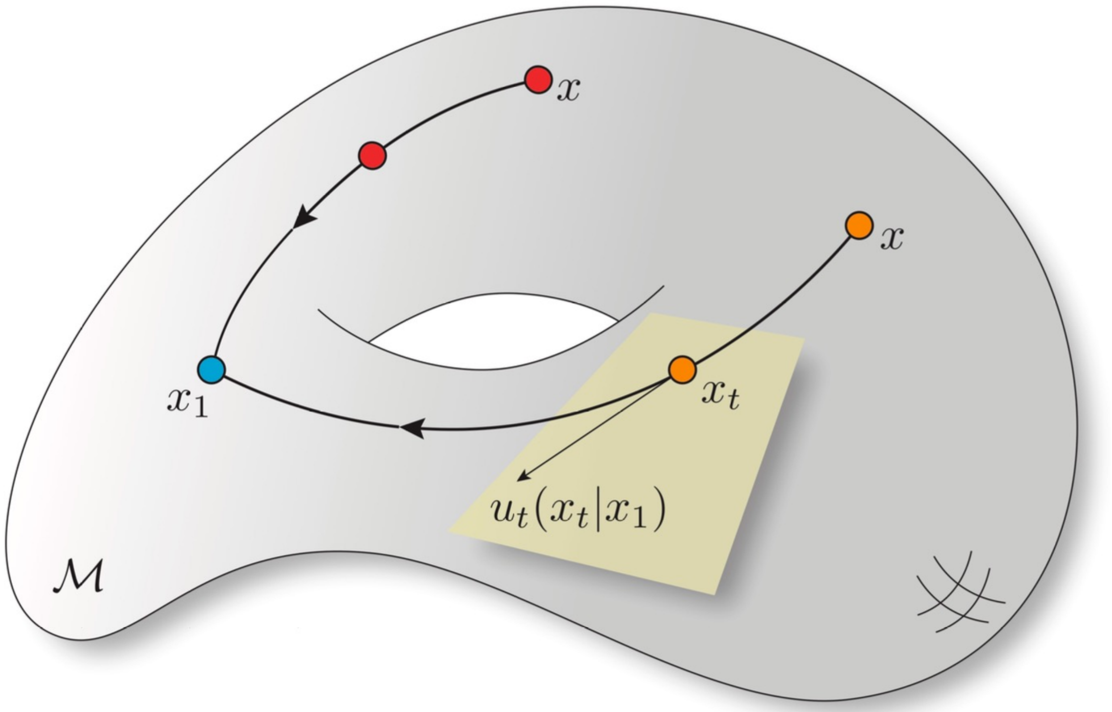


Conditional Flows

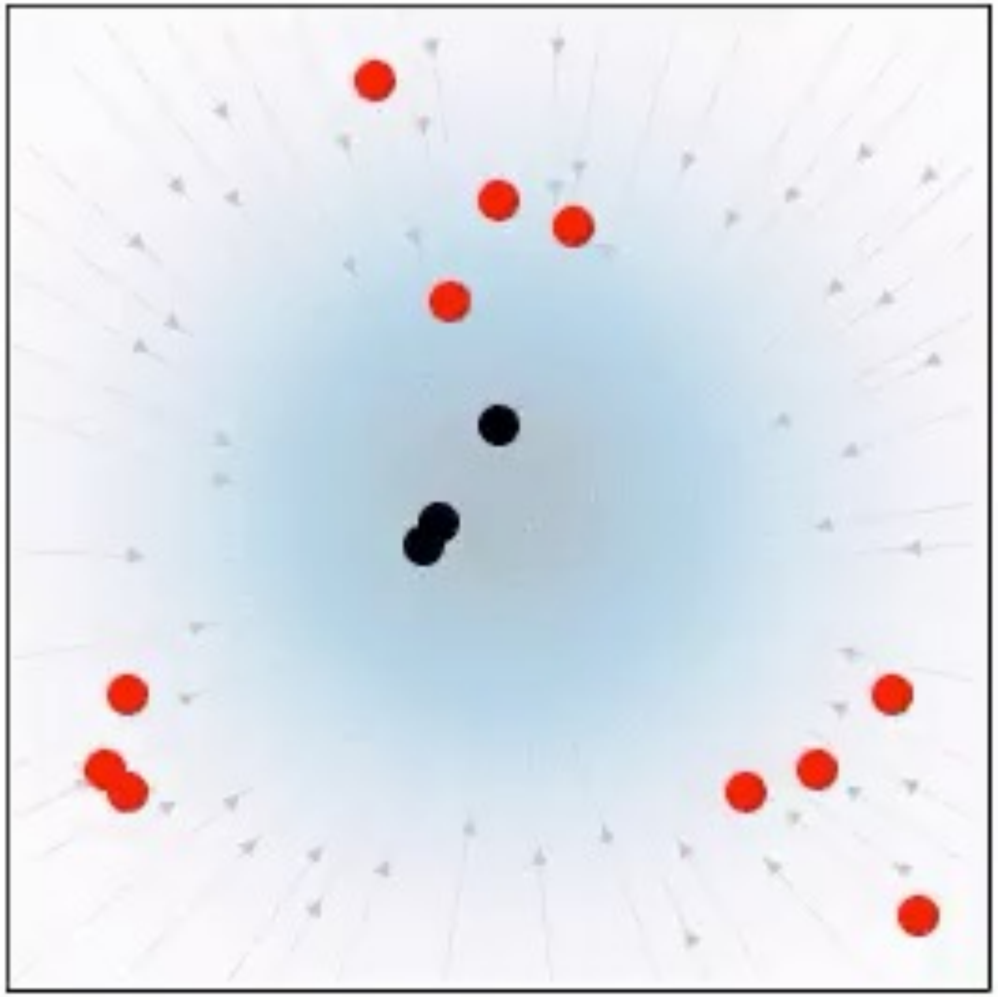
Conditional vector field $u_t(\cdot|x_1)$ inducing conditional probability path $p_t(\cdot|x_1)$

Marginal vector field u_t via *marginalising* over the *conditional* vector field $u_t(\cdot|x_1)$

$$u_t(x) = \int u_t(x|x_1)p_t(x_1|x)dx_1 = \int u_t(x|x_1)\frac{p_t(x|x_1)q(x_1)}{p_t(x)}dx_1 = \mathbb{E}_{x_0,x_t \sim q_1 p_{t|1}}[u_t(x_t|x_1)]$$



Conditional Flows



(Conditional) Flow Matching: Training

(Exact) flow matching: $\mathcal{L}_{\mathrm{FM}}(\theta) = \mathbb{E}_{t x_t \sim \mathcal{U}[0,1] p_t}[\|u_\theta(t, x_t) - u_t(x_t)\|^2]$ with $u_t(x) = \mathbb{E}_{x_0, x_t \sim q_1 p_{t|1}}[u_t(x_t|x_1)]$

Akin to score matching, one can actually move the expectation outside the ℓ^2 norm

Conditional flow matching:
$\mathcal{L}_{\mathrm{CFM}}(\theta) = \mathbb{E}_{t, x_0, x_t \sim \mathcal{U}[0,1] q_1 p_{t|1}}[\|u_\theta(t, x_t) - u_t(x_t|x_1)\|^2]$

$$\nabla_\theta \mathcal{L}_{\mathrm{FM}}(\theta) = \nabla_\theta \mathcal{L}_{\mathrm{CFM}}(\theta)$$

Sampling x_t and evaluating $u(x_t|x_1)$ is available in closed form.

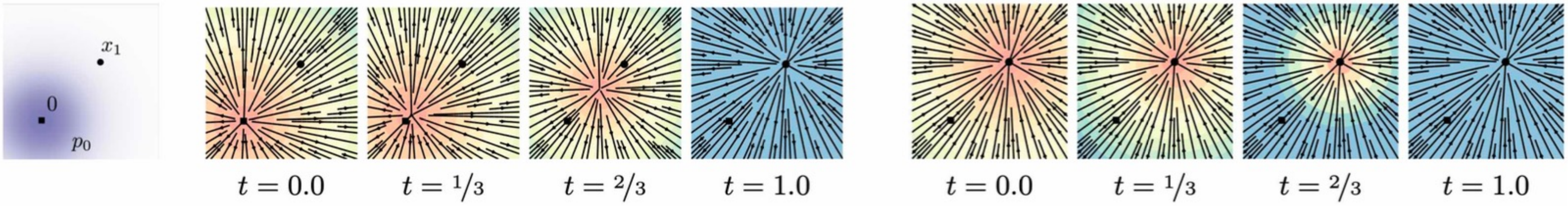
Gaussian Probability Paths

Conditional Probability path: $p_t(x|x_1) = \mathrm{N}(\mu_t(x_1), \sigma_t(x_1)^2 \mathrm{I})$

Conditional vector field: $u_t(x|x_1) = \frac{\sigma_t'(x_1)}{\sigma_t(x_1)}(x - \mu_t(x_1)) + \mu_t'(x_1)$

Example: Linear interpolation

- $\mu_t \triangleq tx_1$ and $\sigma_t \triangleq 1 - t \Rightarrow p_t(x|x_1) = \mathrm{N}(x|tx_1, (1-t)^2)$
- $u_t(x|x_1) = \frac{1}{1-t}(x_1 - x)$



Diffusion path – conditional score function

OT path – conditional vector field

Flow Matching vs. Diffusion

Algorithm 1: Flow Matching training.

Input : dataset q, noise p
Initialize v^θ
while *not converged* **do**
- $t \sim \mathcal{U}([0,1])$ ▷ sample time
- $x_1 \sim q(x_1)$ ▷ sample data
- $x_0 \sim p(x_0)$ ▷ sample noise
- $x_t = \Psi_t(x_0|x_1)$ ▷ conditional flow
- Gradient step with $\nabla_\theta \|v_t^\theta(x_t) - \dot{x}_t\|^2$

Output: v^θ

Algorithm 2: Diffusion training.

Input : dataset q, noise p
Initialize s^θ
while *not converged* **do**
- $t \sim \mathcal{U}([0,1])$ ▷ sample time
- $x_1 \sim q(x_1)$ ▷ sample data
- $x_t = p_t(x_t|x_1)$ ▷ sample conditional prob
- Gradient step with $\nabla_\theta \|s_t^\theta(x_t) - \nabla_{x_t} \log p_t(x_t|x_1)\|^2$

Output: v^θ

$p_t(x_t|x_1)$ general
$p(x_0)$ is general

$p_t(x_t|x_1)$ closed-form from of SDE $dx_t = f_t dt + g_t dw$

- *Variance Exploding*: $p_t(x|x_1) = \mathcal{N}(x|x_1, \sigma_{1-t}^2 I)$
- *Variance Preserving*: $p_t(x|x_1) = \mathcal{N}(x|\alpha_{1-t}x_1, (1-\alpha_{1-t}^2)I)$

$$\alpha_t = e^{-\frac{1}{2}T(t)}$$

$p(x_0)$ is Gaussian
$p_0(\cdot|x_1) \approx p$

Conditional Flow Matching

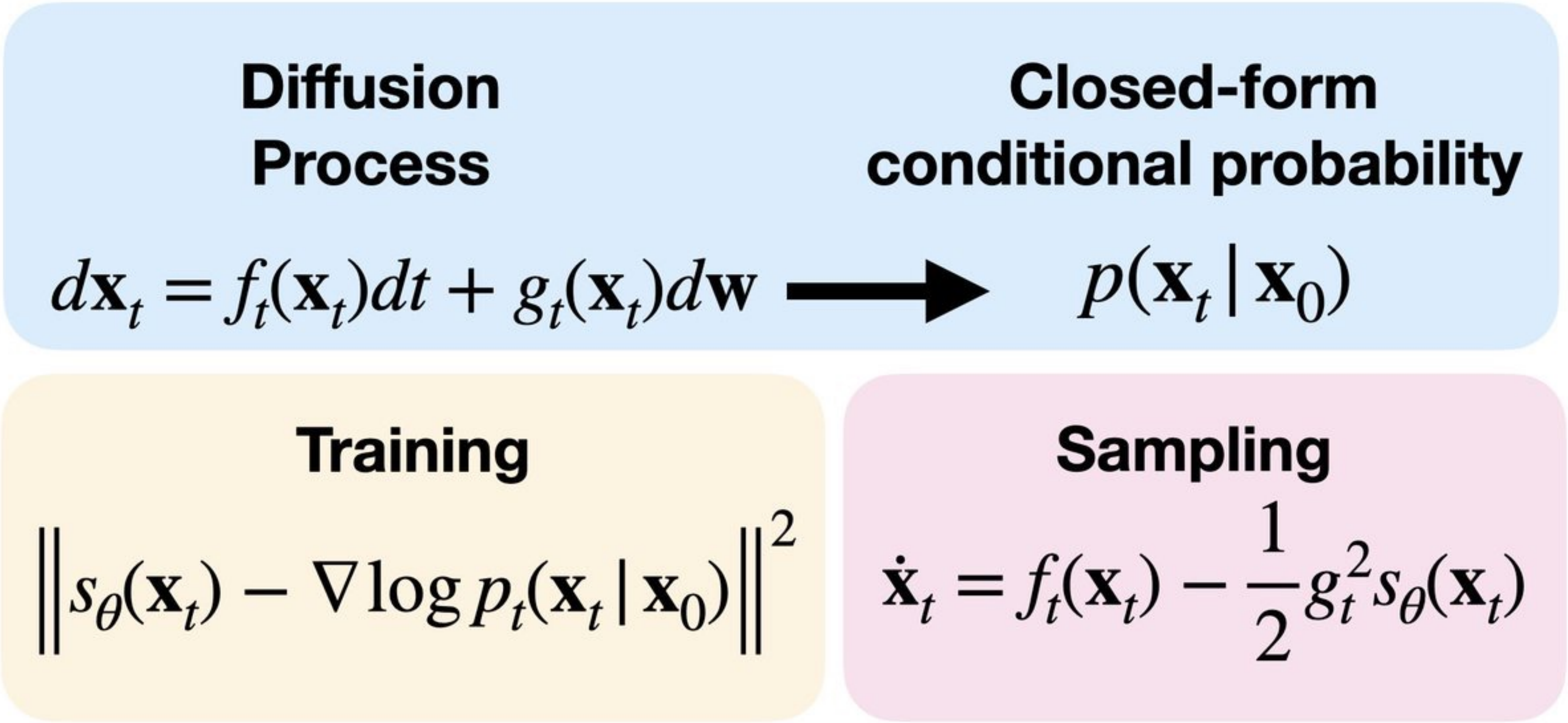
"So, what's the difference between FMs and SBDMs?"

	Learn	ODE inference	SDE inference	Exact endpoints
FM	$u(t, x)$	✓	× [1]	✓
SBDM	$s(t, x)$	✓	✓	× [2]

Which is "better"? 🤷

Note: unclear whether "exact endpoint" and "normalisable guarantee" matters in practice

Score-based modelling



Conditional Flow Matching

General conditional probability path

$$p(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}(\mu_t(\mathbf{x}_0), \sigma_t^2(\mathbf{x}_0)I)$$

Training

$$\left\| v_\theta(\mathbf{x}_t) - u_t(\mathbf{x}_t \mid \mathbf{x}_0) \right\|^2$$

Sampling

$$\dot{\mathbf{x}}_t = v_\theta(\mathbf{x}_t)$$