Lecture 3

CME 296: Diffusion & Large Vision Models

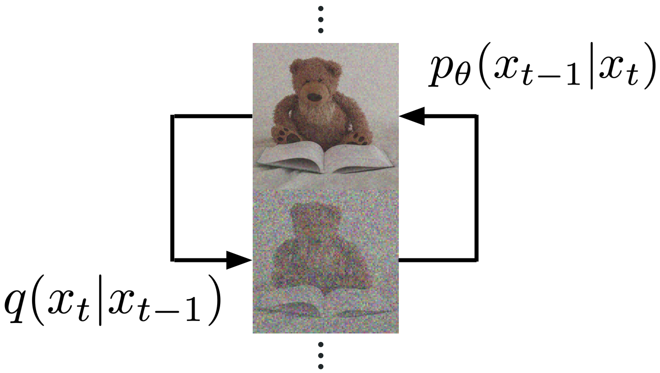
 Afshine Amidi & Shervine Amidi 

Stanford University

ICME

Recap of last episodes...

Lecture 1 (Diffusion with DDPM)



Predict **noise to remove** via $\mathcal{L}_{\text{DDPM}}$

$$\left\|\epsilon_\theta(\sqrt{\overline{\alpha}_t}x_0 + \sqrt{1-\overline{\alpha}_t}\epsilon, t) - \epsilon\right\|^2$$

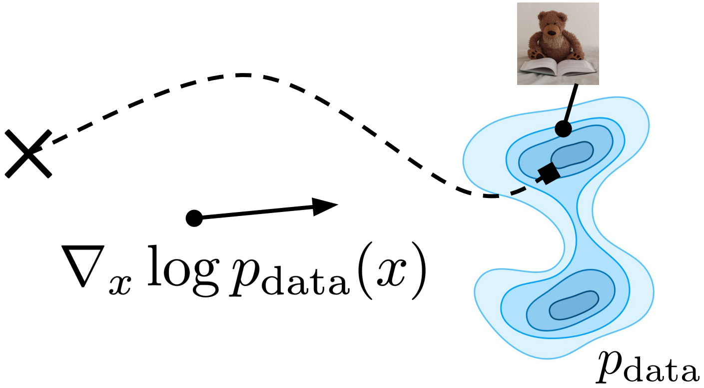
Recap of last episodes...

Lecture 1 (Diffusion with DDPM)

Predict **noise to remove** via $\mathcal{L}_{\text{DDPM}}$

$$\left\|\epsilon_\theta(\sqrt{\overline{\alpha}_t}x_0 + \sqrt{1-\overline{\alpha}_t}\epsilon, t) - \epsilon\right\|^2$$

Lecture 2 (Score matching with DSM)



Predict **score** via $\mathcal{L}_{\text{NCSN}}$

$$\left\|s_\theta(x, \sigma_i) - \nabla_x \log q_{\sigma_i}(x)\right\|^2$$

Recap of last episodes...

Lecture 1 (Diffusion with DDPM) ⟷ **Lecture 2** (Score matching with DSM)

Score from noisy to clean

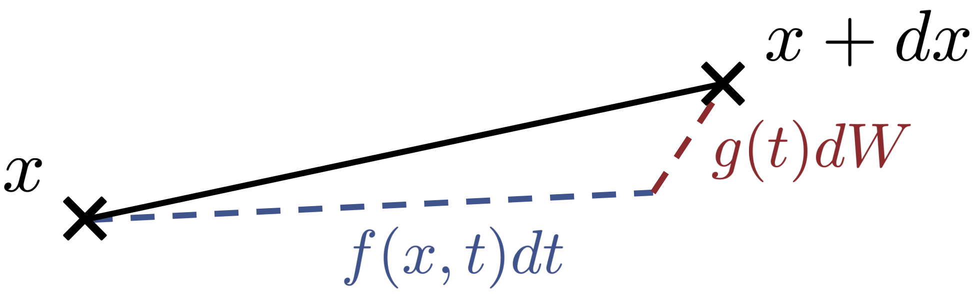
Noise to remove from noisy sample

$$\nabla_{x_t} \log(q(x_t|x_0)) = -\frac{\epsilon}{\sqrt{1-\overline{\alpha}_t}}$$

Recap of last episodes...

Unified view (SDE formulation)

Forward $$dx = \underbrace{f(x,t)dt}_{\textbf{Drift}} + \underbrace{g(t)dW}_{\textbf{Diffusion}}$$



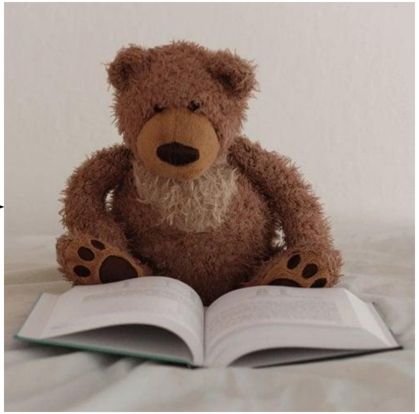
Reverse $$dx = \left[f(x,t) - g(t)^2 \nabla_x \log(p_t(x))\right] dt + g(t)d\overline{W}$$

Today's lecture

Generation paradigm

A teddy bear reading a book

Black box architecture



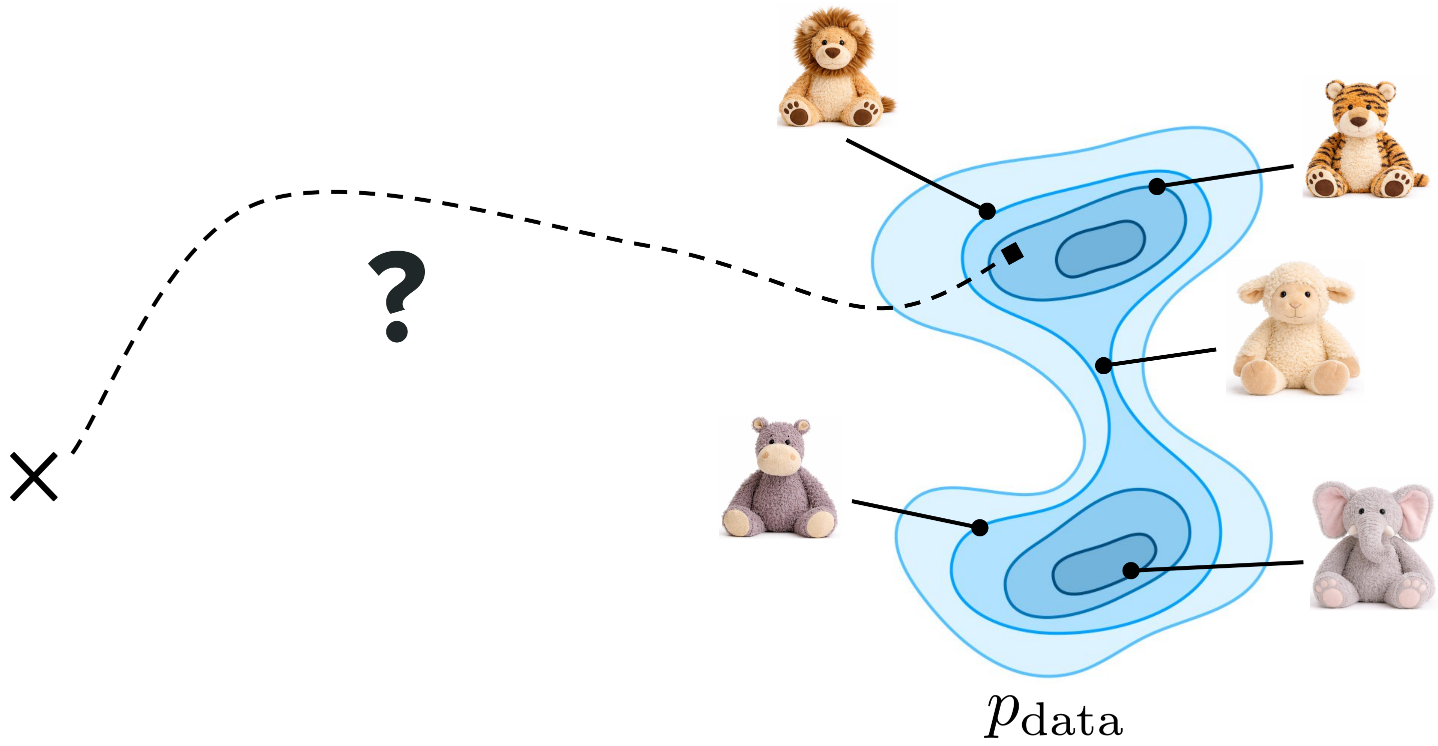
Today's lecture: **Flow matching**

Diffusion & Large Vision Models

Stanford University

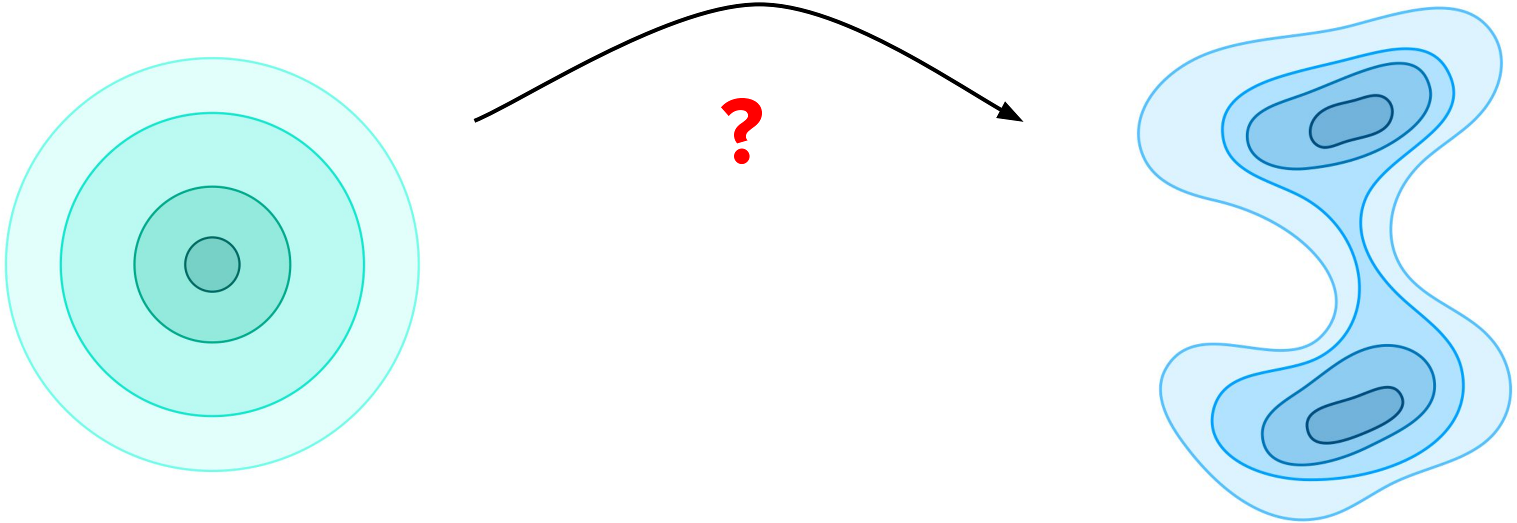
ICME

Problem formulation



Intuition behind flows

Idea. Transport probability mass from initial to target

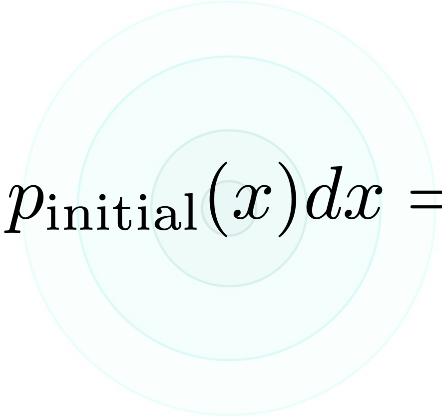


Initial distribution

Target distribution

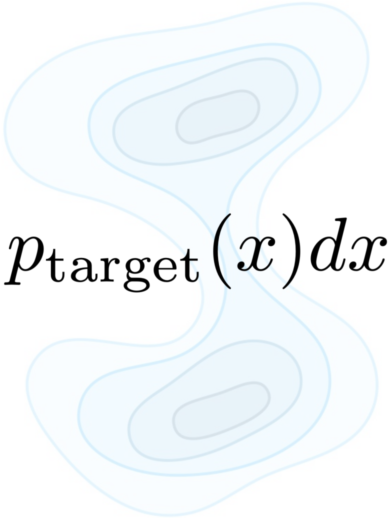
Intuition behind flows

Idea. Transport probability mass from initial to target



$$\int p_{\text{initial}}(x)\,dx = 1$$

?

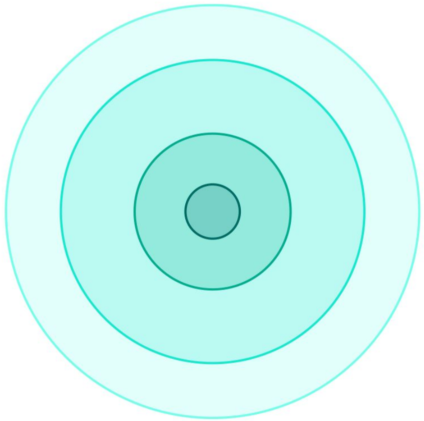


$$\int p_{\text{target}}(x)\,dx = 1$$

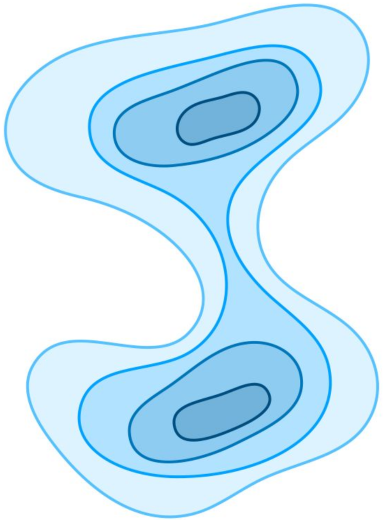
Initial distribution

Target distribution

Notations



$p_0 = \mathcal{N}(0, 1)$



$p_1 = p_{\text{data}}$

⚠️ Important note on conventions ⚠️

Diffusion, score matching, SDE formulation (lectures 1 and 2)

$t = 0$ $\longrightarrow$ $t = T$

clean　　　　　　　　　　　　　　　　　　noise

⚠️ Important note on conventions ⚠️

Diffusion, score matching, SDE formulation (lectures 1 and 2)

$t = 0$ $\longrightarrow$ $t = T$

clean | noise

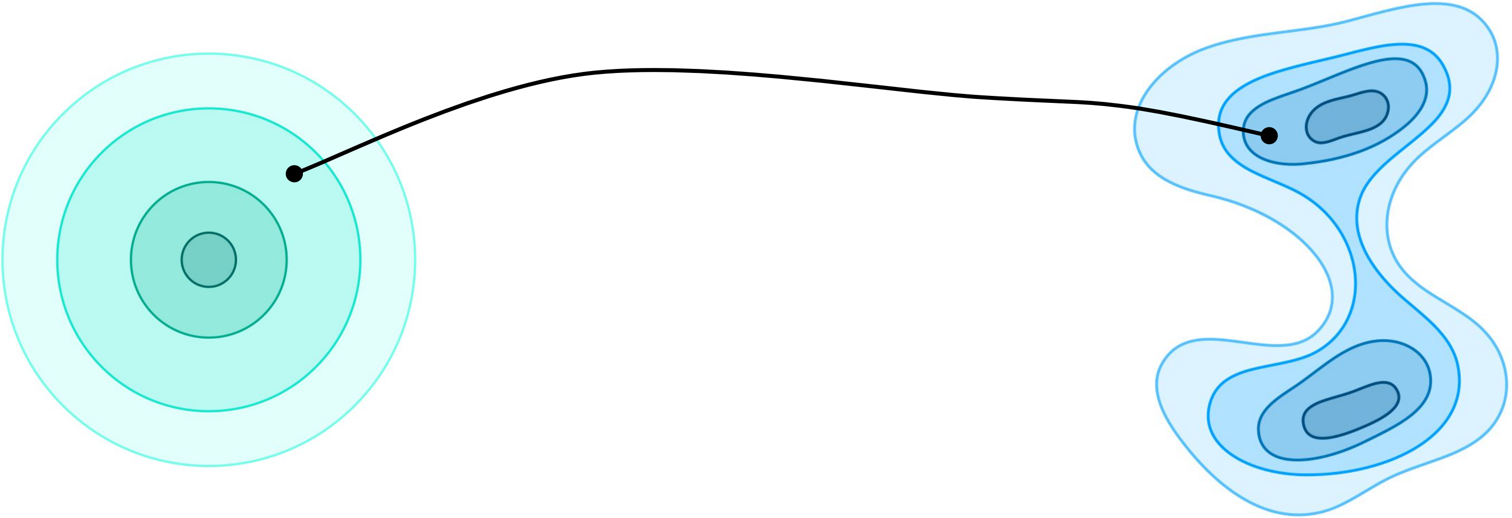
Flow matching (this lecture!)

$t = 0$ $\longrightarrow$ $t = 1$

Initial data distribution | **Target** data distribution

Trajectory

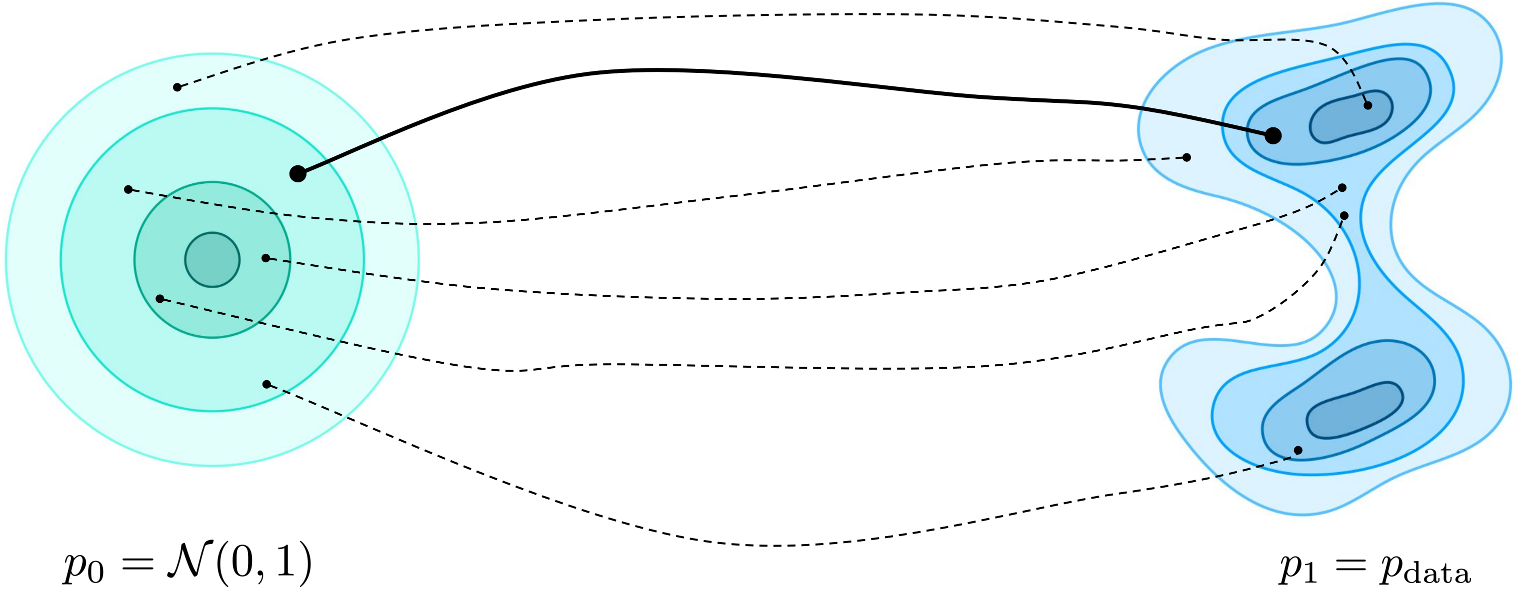
Trajectory x_t : path taken by the observation for time $t \in [0, 1]$



$p_0 = \mathcal{N}(0, 1)$ $p_1 = p_{\text{data}}$

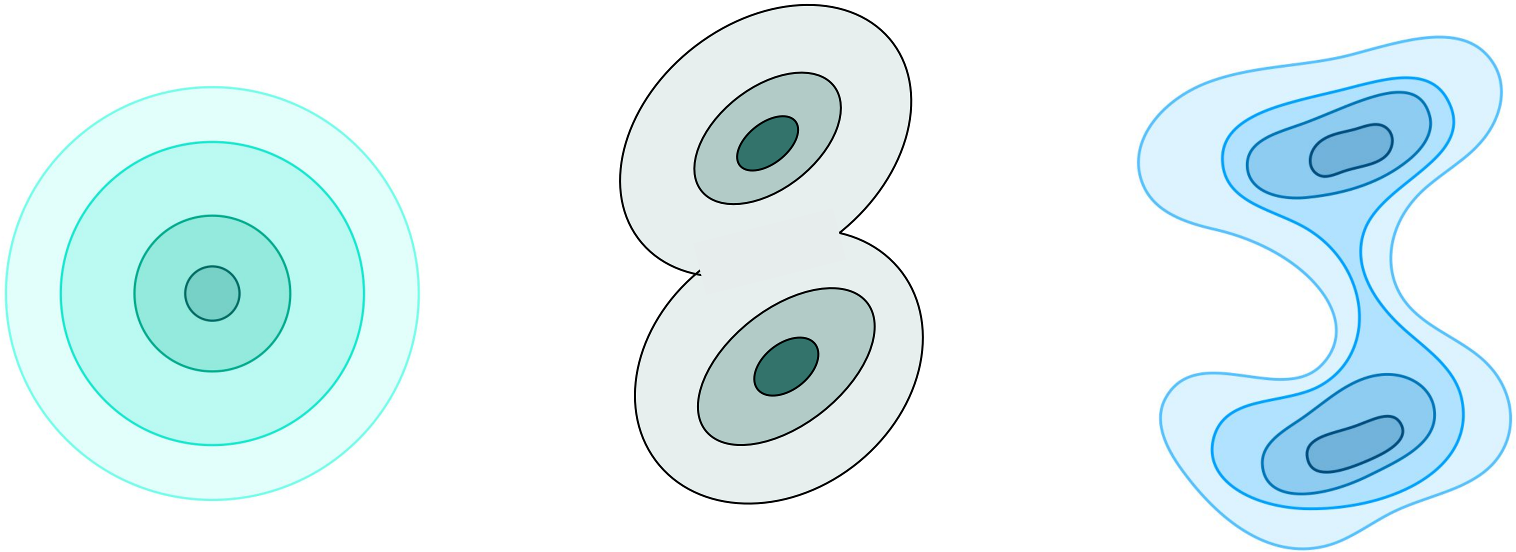
Flow

Flow $\psi_t(x_0)$: collection of trajectories x_t which start from different x_0



Probability path

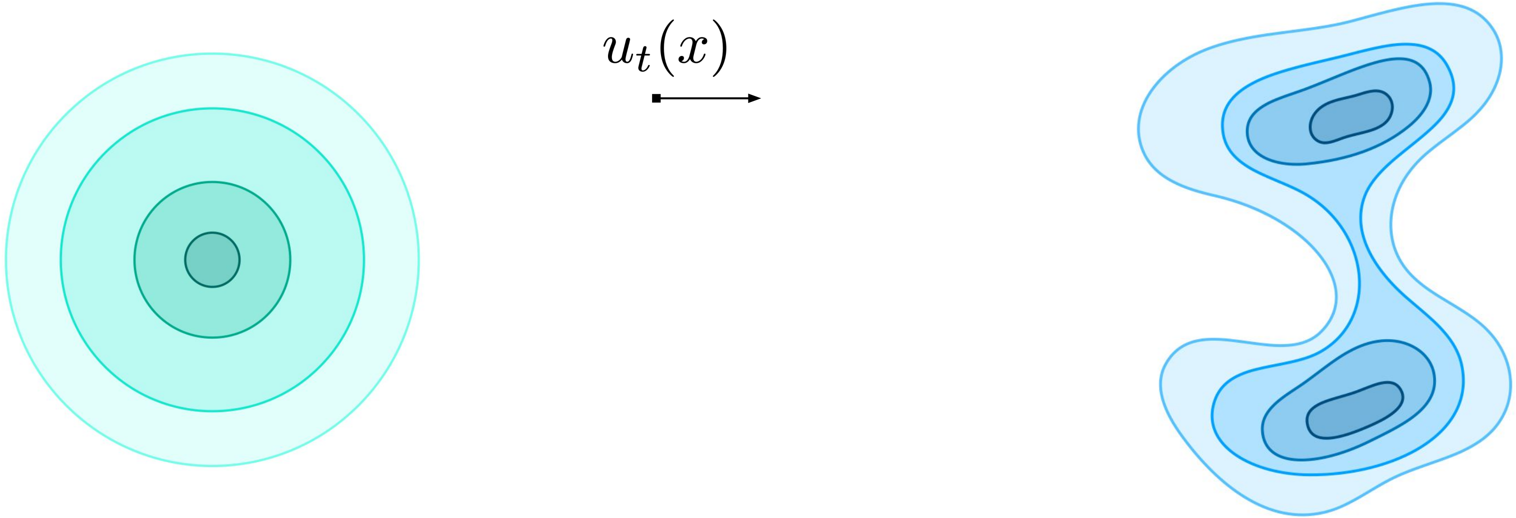
Probability path $p_t(x)$: probability distribution of x_t at time t



$p_0 = \mathcal{N}(0, 1)$ $\qquad$ p_t $\qquad$ $p_1 = p_{\mathrm{data}}$

Vector field, also known as velocity

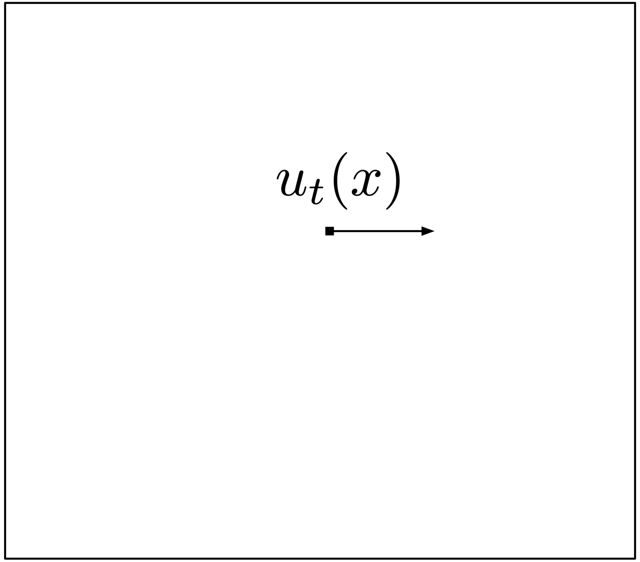
Vector field $u_t(x)$: where to move (direction, speed) at time t and location x



$p_0 = \mathcal{N}(0, 1)$ $p_1 = p_{\text{data}}$

Differences between velocity and score function

Velocity



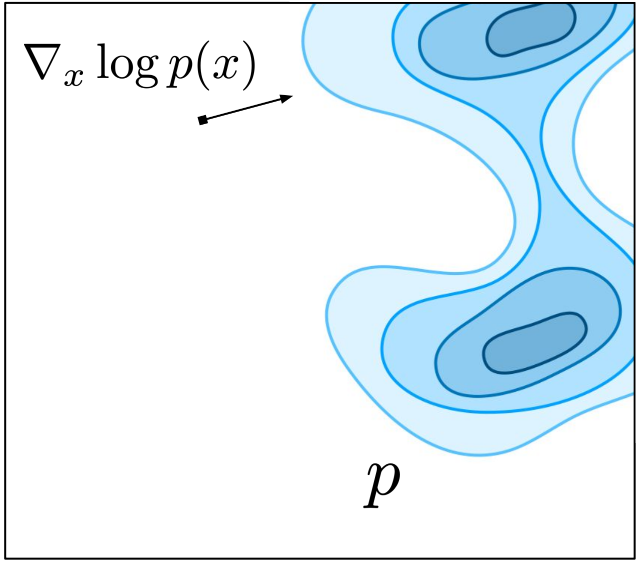
Highway

Differences between velocity and score function

Velocity

Highway

Score function



Compass

Perspective of a single sample with ODE

Ordinary differential equation:

Vector field at

location x and **time** t

Change in

sample position

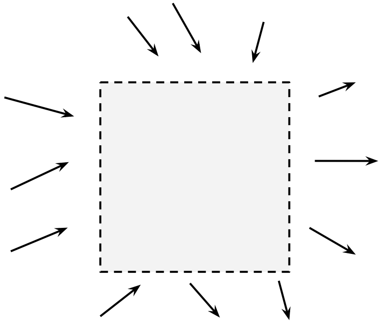
$$dx = u_t(x)dt$$

Change in

time

$\underbrace{\textbf{Trajectory} \text{ starting from } x_0}_{\psi_t(x_0)}$ is **unique** if velocity $u_t(x)$ is **Lipschitz continuous**

Perspective of a distribution via "mass conservation"

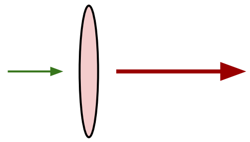


$$\underbrace{\text{\textbf{Temporal evolution} of density at time t at a particular location}}_{\frac{\partial p_t}{\partial t}(x)} = \underbrace{\text{"\textbf{Inflow}" of density} - \text{"\textbf{outflow}" of density at that particular location}}_{?}$$

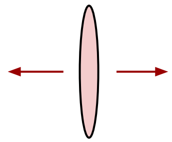
Intuition behind divergence in 1D

Divergence $\geqslant 0$

Leaving more than what is coming



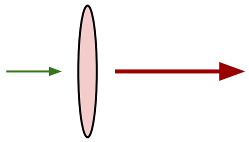
Leaving



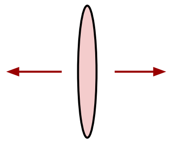
Intuition behind divergence in 1D

Divergence $\geqslant 0$

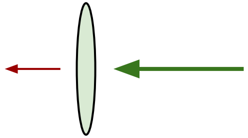
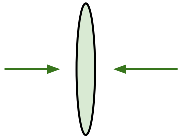
$$\frac{\partial f}{\partial x} \geqslant 0$$



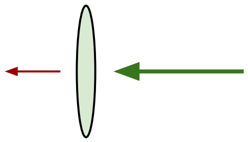
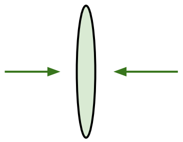
$$\frac{\partial f}{\partial x} \geqslant 0$$



Intuition behind divergence in 1D

Divergence $\geqslant 0$	**Divergence** $\leqslant 0$
$\dfrac{\partial f}{\partial x} \geqslant 0$	Coming more than what is leaving 
$\dfrac{\partial f}{\partial x} \geqslant 0$	Coming 

Intuition behind divergence in 1D

Divergence $\geqslant 0$		**Divergence** $\leqslant 0$	
$\dfrac{\partial f}{\partial x} \geqslant 0$		$\dfrac{\partial f}{\partial x} \leqslant 0$	
$\dfrac{\partial f}{\partial x} \geqslant 0$		$\dfrac{\partial f}{\partial x} \leqslant 0$	

Intuition behind divergence in 1D

Divergence $\geqslant 0$		Divergence $\leqslant 0$	
$\dfrac{\partial f}{\partial x} \geqslant 0$		$\dfrac{\partial f}{\partial x} \leqslant 0$	
$\dfrac{\partial f}{\partial x} \geqslant 0$		$\dfrac{\partial f}{\partial x} \leqslant 0$	

Generalization for n dimensions:

$$\boxed{\operatorname{div}(f) = \nabla \cdot f = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}}$$

Divergence: yes. But of what?

Question. Which f should we consider?

❌ Vector field $u_t(x)$

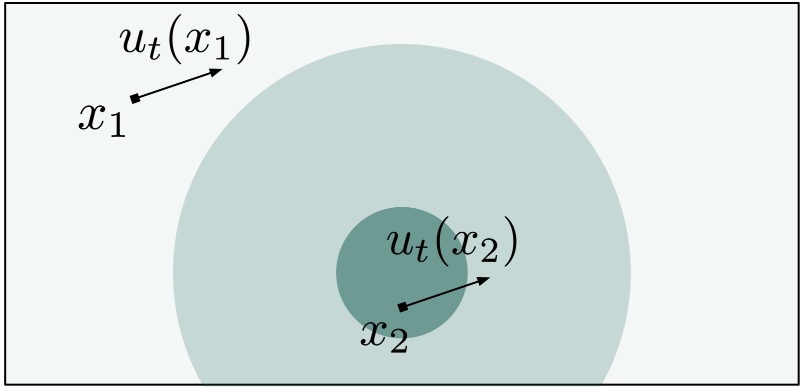
✅ **Probability flux** $(p_t u_t)(x)$

Probability density

Velocity

Suppose:

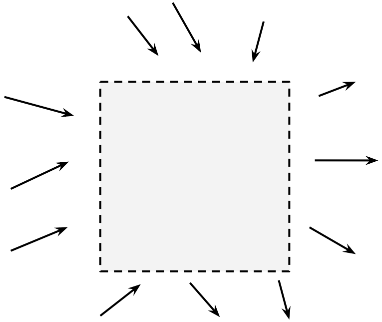
$$u_t(x_1) = u_t(x_2) \qquad p_t(x_1) < p_t(x_2)$$



We want:

$$\|f(x_1)\| < \|f(x_2)\|$$

Perspective of a distribution with continuity equation

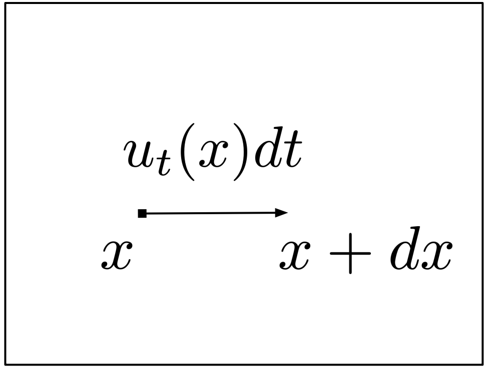


$$\underbrace{\text{\textbf{Temporal evolution} of density at time t at a particular location}}_{\frac{\partial p_t}{\partial t}(x)} = \underbrace{\text{"\textbf{Inflow}" of density} - \text{"\textbf{outflow}" of density at that particular location}}_{-\nabla \cdot (p_t u_t)(x)}$$

Two perspectives involving the velocity

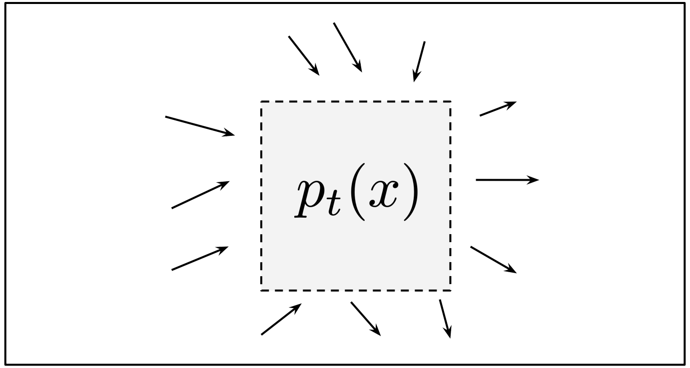
Single sample

$$\frac{dx}{dt} = u_t(x)$$



Distribution

$$\frac{\partial p_t}{\partial t}(x) = -\nabla \cdot (p_t u_t)(x)$$



Vector field $u_t(x)$ "**generates**" the probability path $p_t(x)$

Two perspectives involving the velocity

Single sample $\longleftrightarrow$ **Distribution**

$$\boxed{\begin{gathered} x_0 \sim p_0 \\ \frac{dx_t}{dt} = u_t(x_t) \end{gathered}} \implies \boxed{x_t \sim p_t}$$

Flow models

Goal. Map $x_0 \sim p_0$ to $x_1 \sim p_1$

Strategy.

1. **Training**. Estimate $u_t(x)$ for all time t and all locations x via $u_t^\theta(x)$

2. **Inference**. Sample from initial distribution and solve numerically the ODE using the learned vector field $u_t^\theta(x)$

$$\widehat{x}_1 = x_0 + \int_0^1 u_t^\theta(x) dt$$

Focus of the rest of this lecture

Goal. Map x_0 to x_1

Strategy.

1 **Training**. Estimate $u_t(x)$ for all time t and all locations x via $u_t^\theta(x)$

2 **Inference**. Sample from initial distribution and solve numerically the ODE using the learned vector field $u_t^\theta(x)$

$$\widehat{x}_1 = x_0 + \int_0^1 u_t^\theta(x)dt$$

Earlier attempts to learn the velocity

Goal. Learn vector field $u_t(x)$ via **maximum likelihood**

Idea. Transform continuity equation to:

$$\frac{d}{dt} \log p_t(x) = -\nabla \cdot u_t(x)$$

Then, simulate ODE at training time to maximize likelihood:

$$\log p_1^\theta(x_1) = \log p_0(x_0) + \int_0^1 -\nabla \cdot u_t^\theta(x)\, dt$$

Limitation. **Training** is **slow** and **expensive**!

Neural Ordinary Differential Equations, Chen et al., 2018.

Diffusion & Large Vision Models

Stanford University

ICME

Flow matching

FM = **F**low **M**atching

Goal. Estimate the vector field with $u_t^\theta(x)$

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

Flow matching

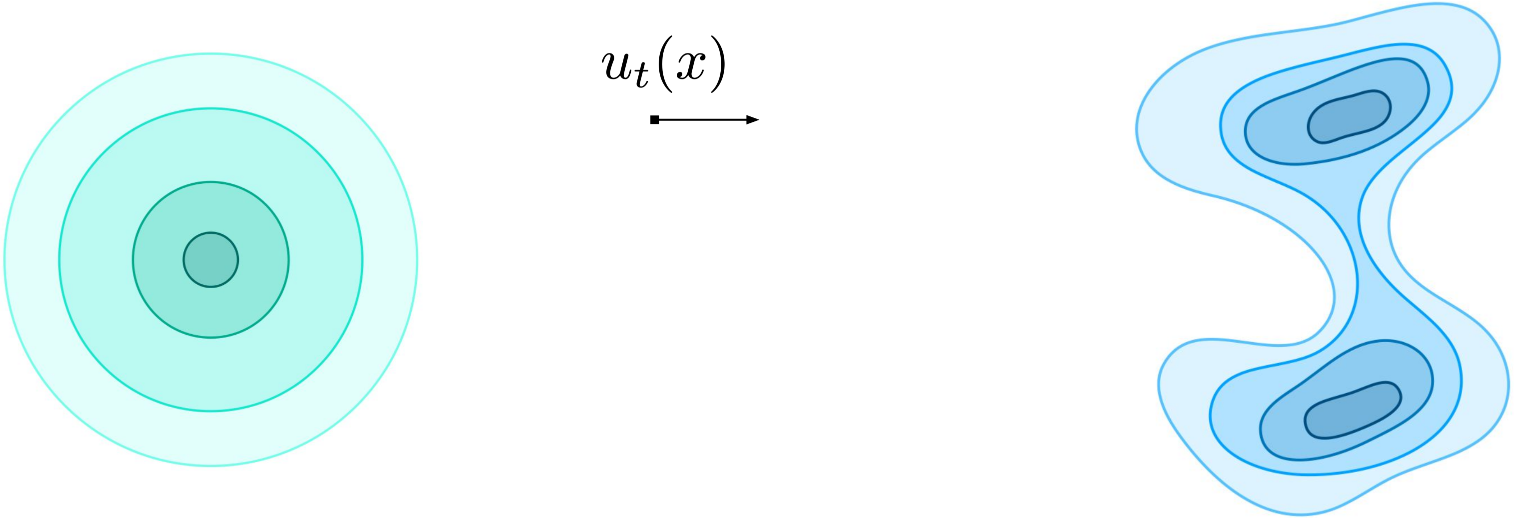
FM = **F**low **M**atching

Goal. Estimate the vector field with $u_t^\theta(x)$

$$\mathcal{L}_{\mathrm{FM}} = \mathbb{E}_{t,x}\left[\left|\left|u_t^\theta(x) - {\color{red}u_t(x)}\right|\right|^2\right]$$

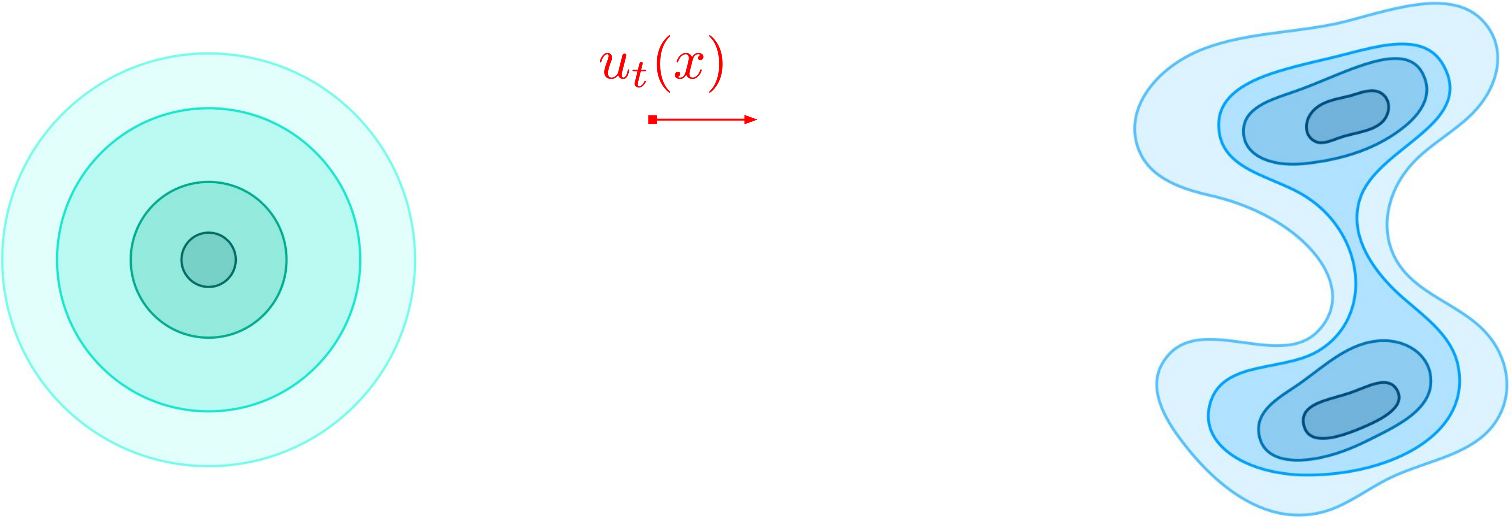
We don't have access to it

Back to the definition of a vector field



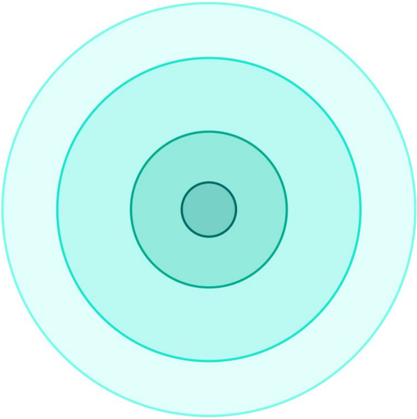
$p_0 = \mathcal{N}(0, 1)$ $\qquad$ $p_1 = p_{\text{data}}$

Vector field is unknown to us

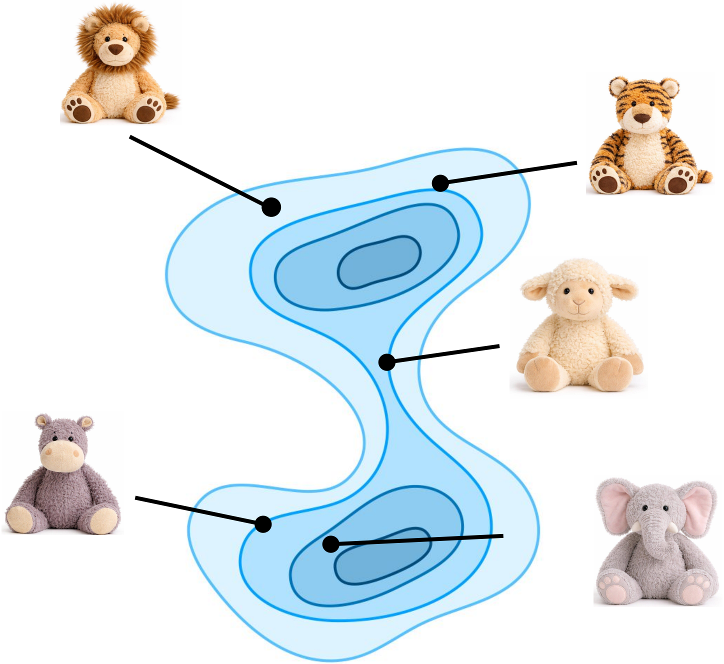


$p_0 = \mathcal{N}(0,1)$ $\qquad$ $p_1 = p_{\text{data}}$

Let's fall back to an easier setup

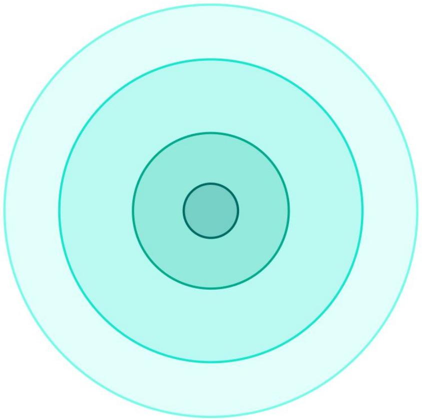


$$p_0 = \mathcal{N}(0, 1)$$

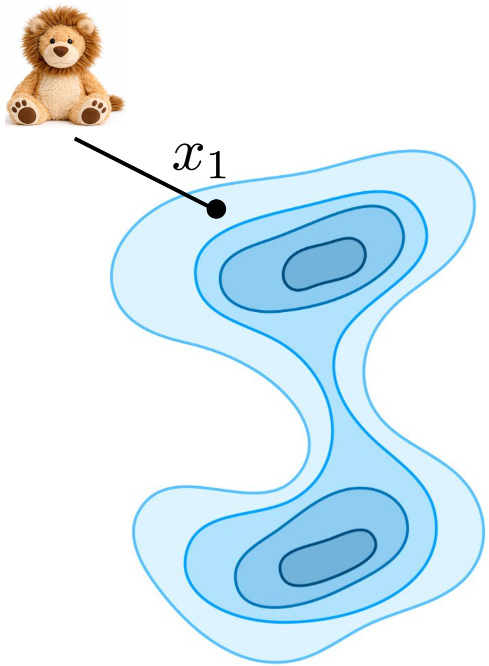


$$p_1 = p_{\text{data}}$$

Let's fall back to an easier setup

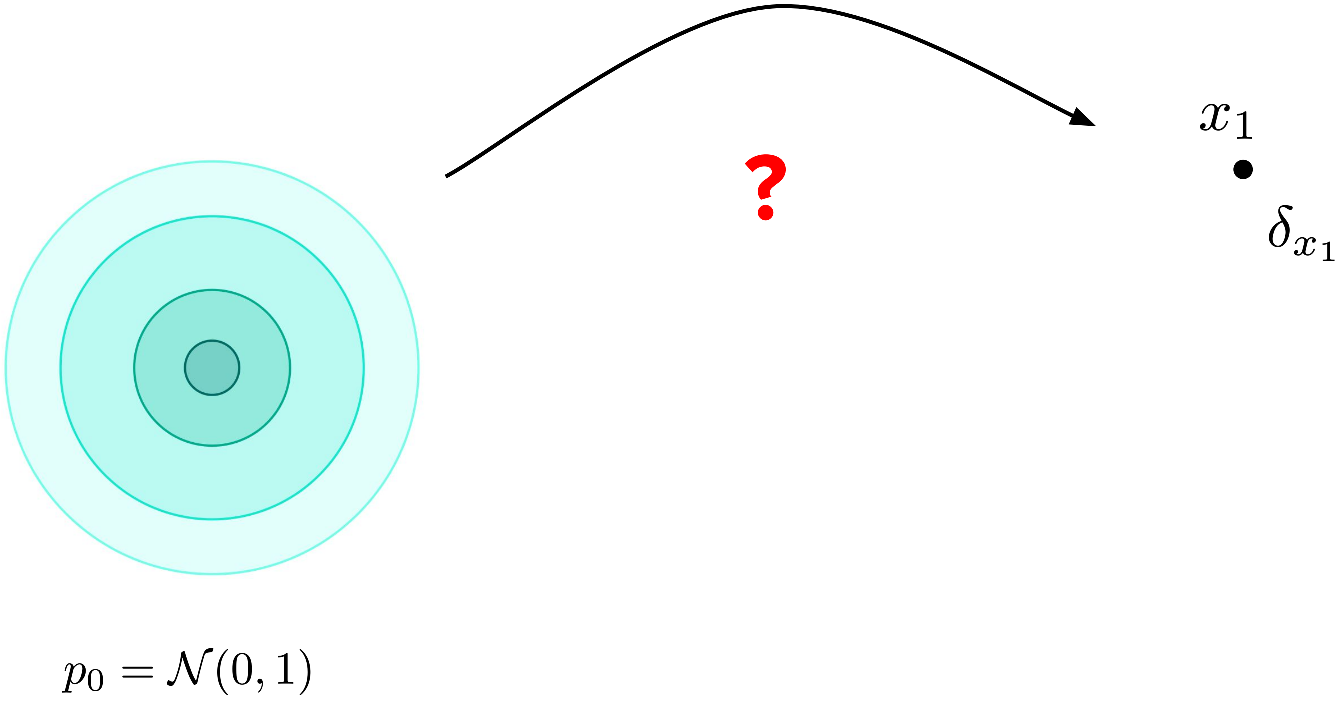


$p_0 = \mathcal{N}(0, 1)$



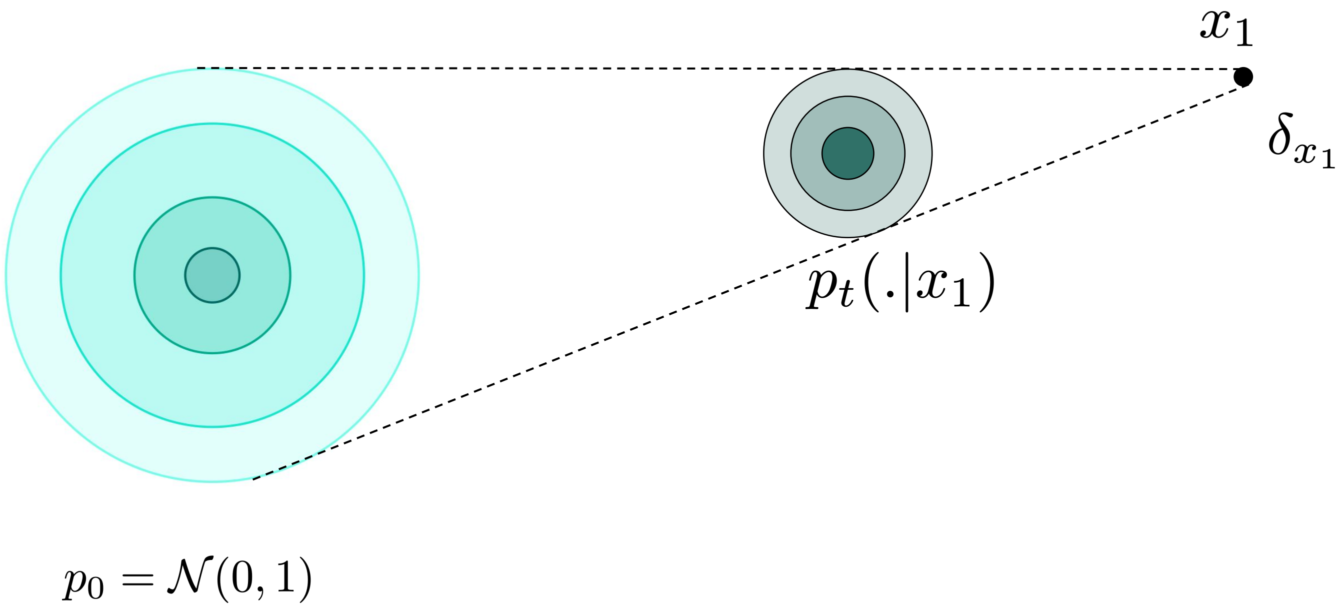
$p_1 = p_{\text{data}}$

How to go from initial to Dirac distribution?



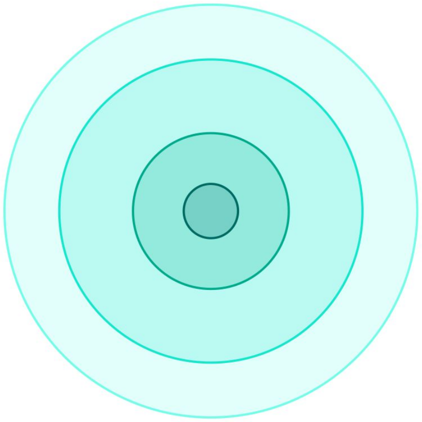
Proposed conditional probability path

Conditional Gaussian **probability path** $p_t(x|x_1) = \mathcal{N}(tx_1, (1-t)^2 I)$

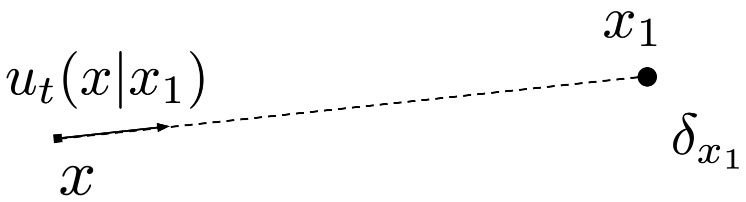


Conditional vector field

Conditional **vector field** $u_t(x|x_1) = \dfrac{x_1 - x}{1 - t}$



$p_0 = \mathcal{N}(0, 1)$



Generation of conditional probability path

Conditional **vector field** $u_t(.\|x_1)$	**GENERATES**	Conditional **probability path** $p_t(.\|x_1)$

Derivation: Use continuity equation

Generation of conditional probability path

Conditional
vector field
$u_t(.|x_1)$

GENERATES

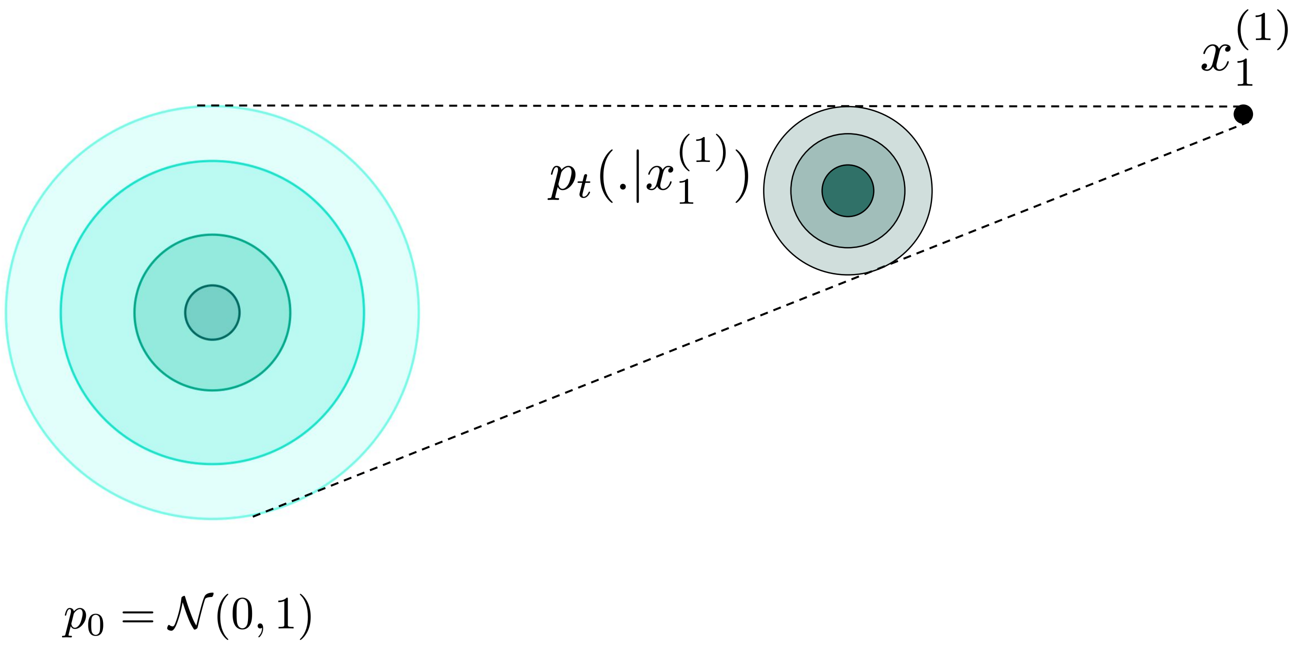
Conditional
probability path
$p_t(.|x_1)$

Consequence. If $x_0 \sim p_0(.|x_1)$ and $\frac{dx_t}{dt} = u_t(x_t|x_1)$ then:

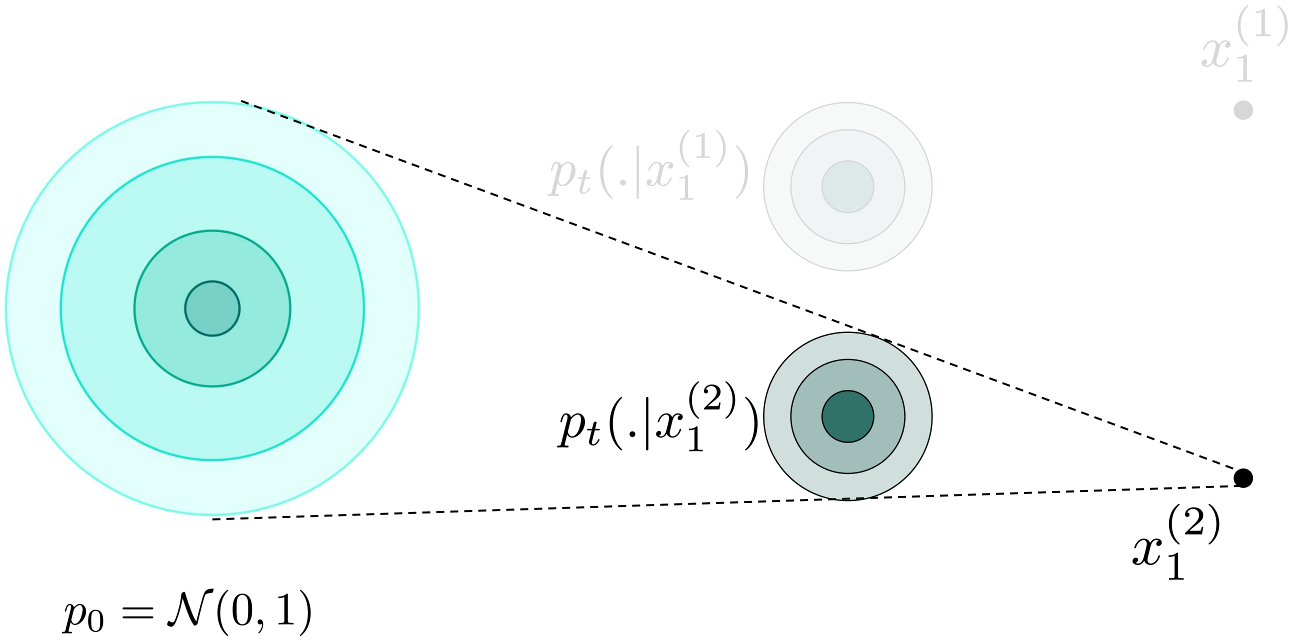
$$x_t \sim p_t(.|x_1)$$

Derivation: Use continuity equation

Marginal probability path

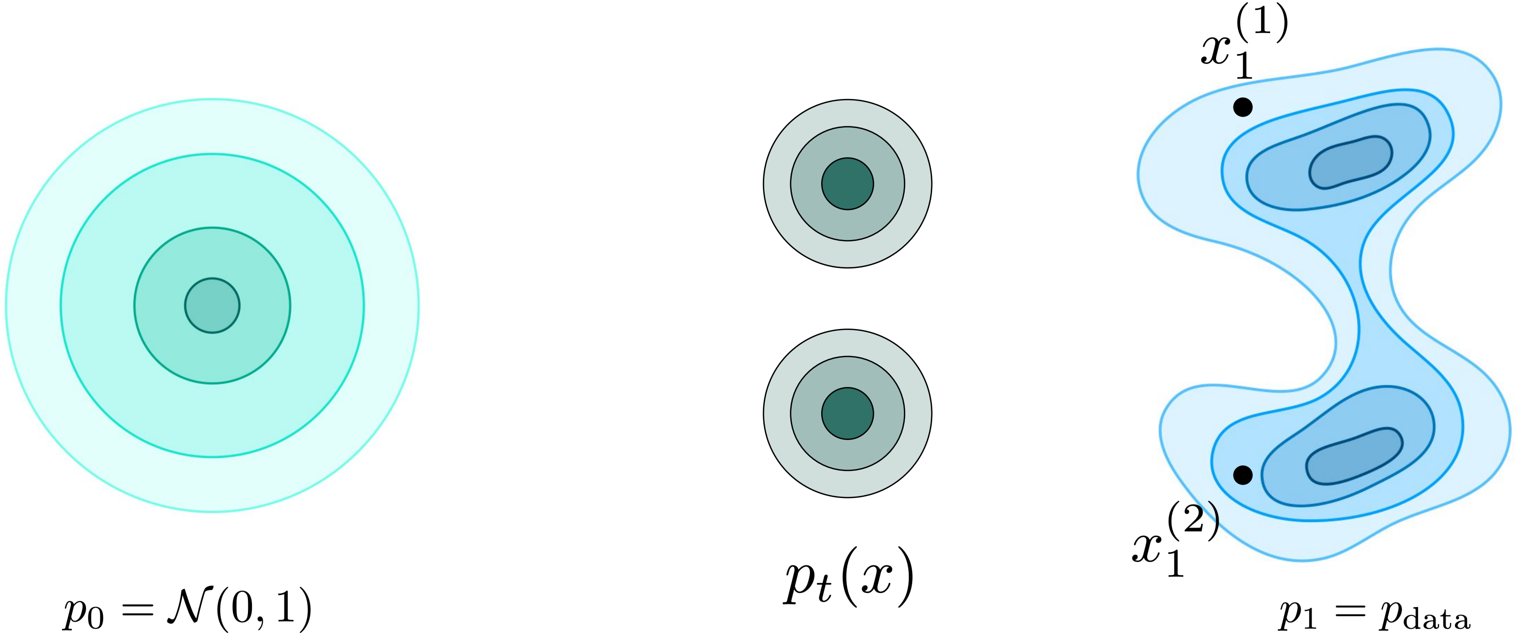


Marginal probability path



Marginal probability path

Marginal probability path p_t is an aggregation of the $p_t(.|x_1^{(i)})$



 Note: Illustration shows marginal if training set has 2 data points for readability purposes.

Marginal probability path

Marginal probability path $p_t(x)$:

$$p_t(x) = \int p_t(x|x_1)p_{\text{data}}(x_1)dx_1$$

This definition leads to:

$$p_{t=0} = p_0$$

Marginal probability path for $t = 0$

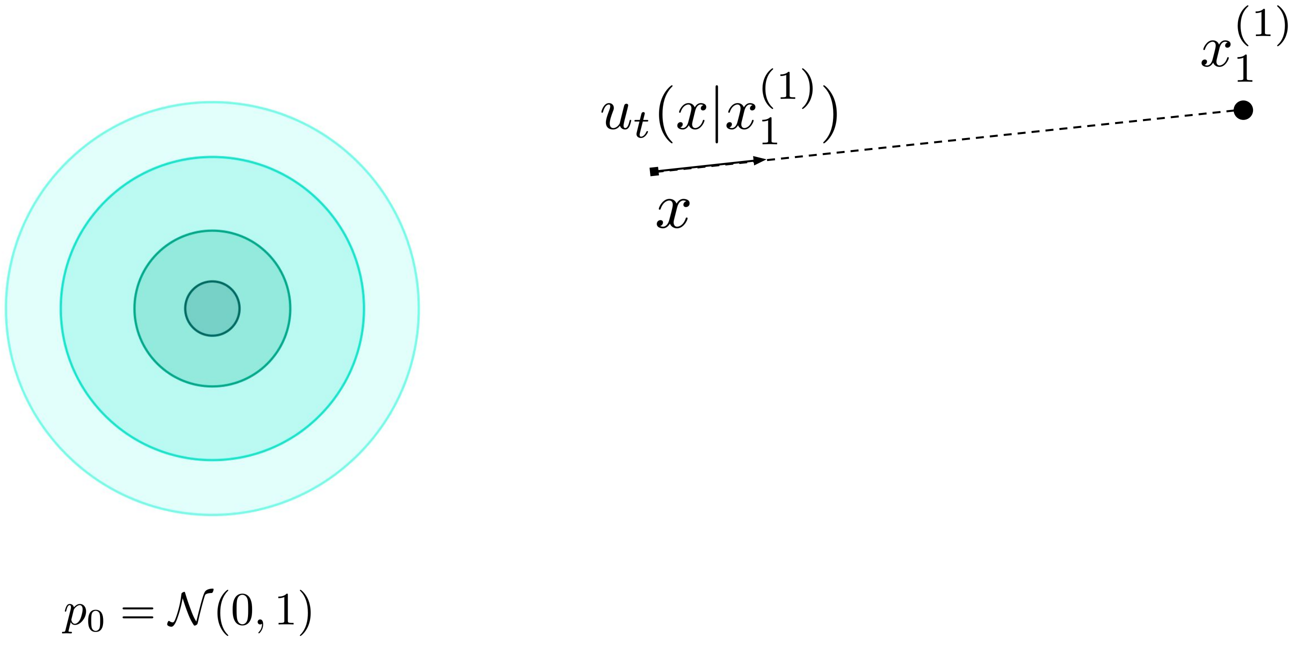
Initial probability distribution

$$p_{t=1} = p_{\text{data}}$$

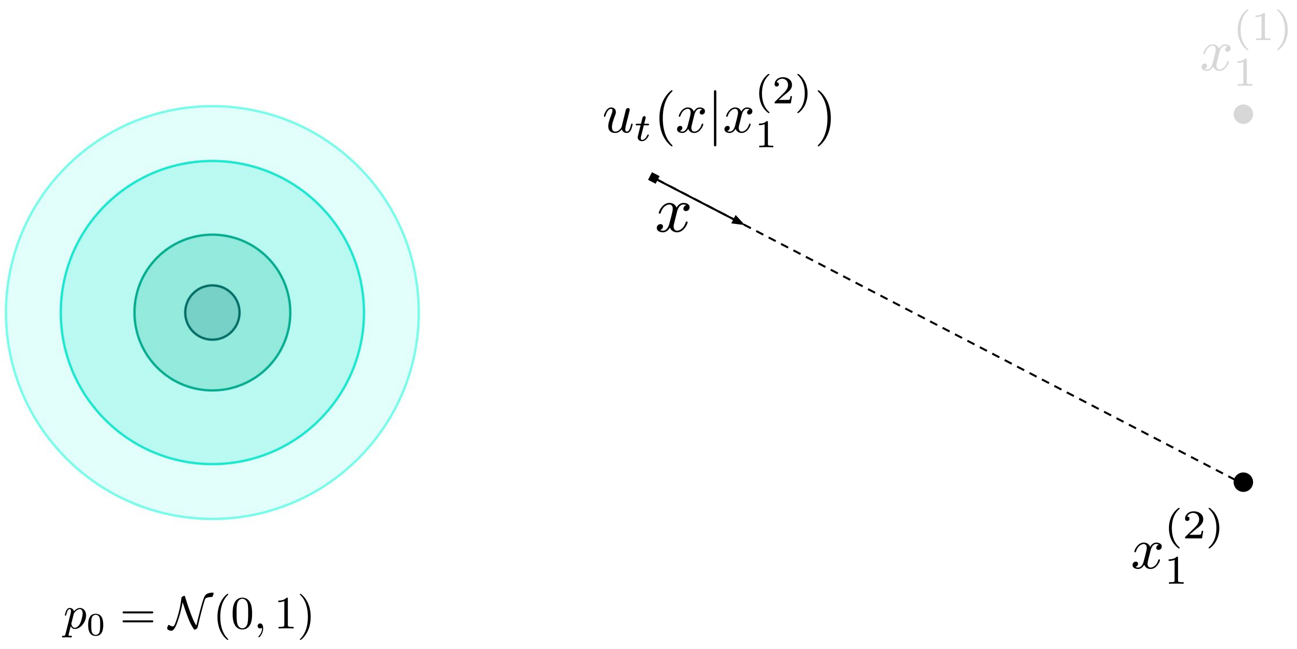
Marginal probability path for $t = 1$

Target probability distribution

Marginal vector field

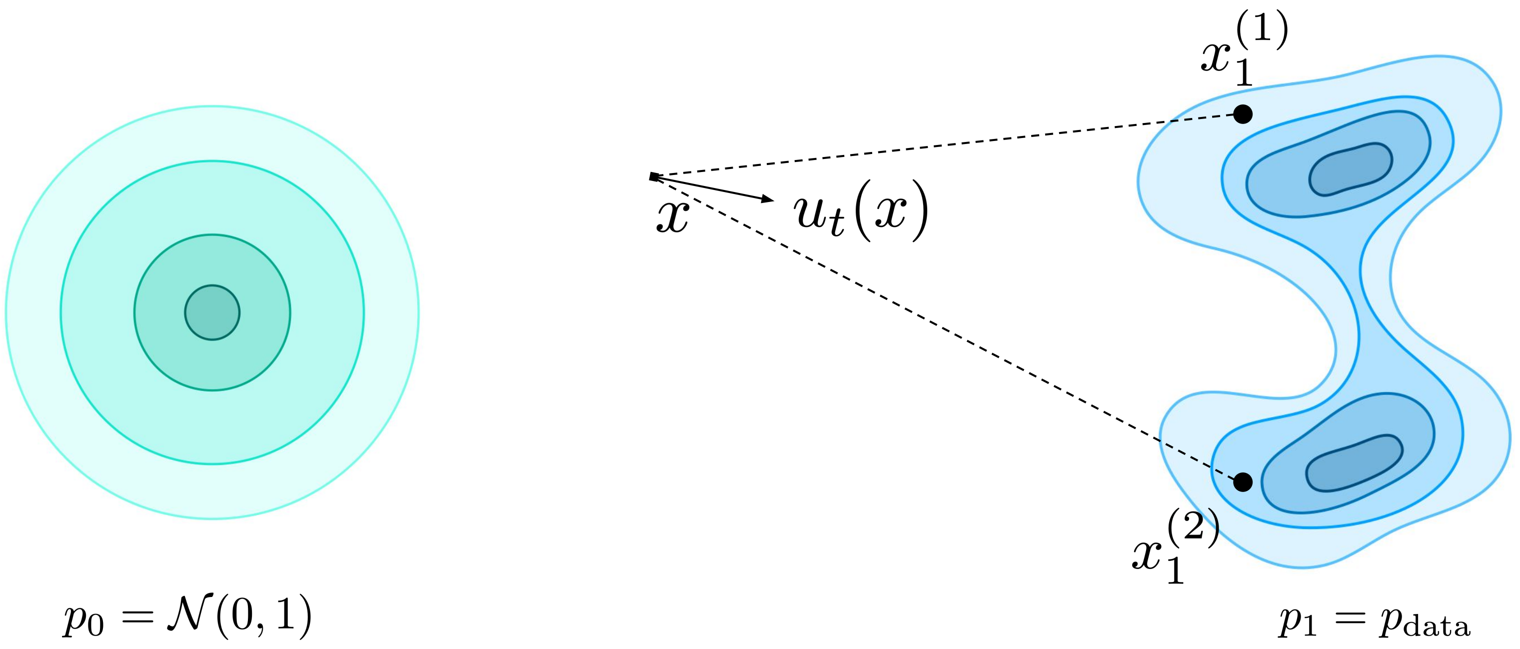


Marginal vector field



Marginal vector field

Marginal vector field u_t is an aggregation of the $u_t(.|x_1^{(i)})$



Note: Illustration shows marginal if training set has 2 data points for readability purposes.

Marginal vector field

Marginal vector field $u_t(x)$:

$$u_t(x) = \int u_t(x|x_1) \boxed{\frac{p_t(x|x_1)p_{\text{data}}(x_1)}{p_t(x)}} dx_1$$

Posterior mean

$p(x_1|x)$

Interpretation of the weight. Given **where you are**, **where should you go**?

Generation of probability path

Marginal **vector field**

u_t

GENERATES

Marginal **probability path**

p_t

Derivation: Use marginal definition with continuity equation

Generation of probability path

Marginal **vector field**

u_t

GENERATES

Marginal **probability path**

p_t

Consequence. If $x_0 \sim p_0$ and $\frac{dx_t}{dt} = u_t(x_t)$ then:

$$x_t \sim p_t$$

Derivation: Use marginal definition with continuity equation

Deriving conditional flow matching

Definition of flow matching for $u_t(x)$

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

This is equivalent to optimizing the following loss:

$$\mathcal{L}_{\mathrm{CFM}} = \mathbb{E}_{t,x_1,x}\left[\left\| u_t^\theta(x) - \underbrace{u_t(x \mid x_1)}_{x_1 - x_0} \right\|^2\right]$$

tractable!

Derivation: Expand squared norm and use definition of marginal

Full derivation in Flow Matching for Generative Modeling, Lipman et al., 2022.

Conditional flow matching

CFM = **C**onditional **F**low **M**atching

We now have a **tractable** loss!

$$\mathcal{L}_{\text{CFM}} = \mathbb{E}_{t,x_1,x}\left[\left\|u_t^\theta(x) - \underbrace{u_t(x \mid x_1)}_{x_1 - x_0}\right\|^2\right]$$

Flow Matching for Generative Modeling, *Lipman et al., 2022.*

Recap

Strategy.

1. Derive target **vector field** for a **simple case** ← **Dirac**

Recap

Strategy.

1. Derive target **vector field** for a **simple case** ⟵ **Dirac**
2. Construct target **vector field** for **our case** ⟵ **Continuity equation**

Recap

Strategy.

1. Derive target **vector field** for a **simple case** ⟵ **Dirac**
2. Construct target **vector field** for **our case** ⟵ **Continuity equation**
3. Show loss function is **tractable** ⟵ **Gradient of FM and CFM are equal**

Recap

Strategy.

1. Derive target **vector field** for a **simple case** ⟵ **Dirac**
2. Construct target **vector field** for **our case** ⟵ **Continuity equation**
3. Show loss function is **tractable** ⟵ **Gradient of FM and CFM are equal**
4. Deduce **final loss function** ⟵ **Incredibly simple loss!**

Recommended read: Flow Matching paper

Preprint

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

Abstract

We introduce a new paradigm for generative modeling built on Continuous Normalizing Flows (CNFs), allowing us to train CNFs at unprecedented scale. Specifically, we present the notion of Flow Matching (FM), a simulation-free approach for training CNFs based on regressing vector fields of fixed conditional probability paths. Flow Matching is compatible with a general family of Gaussian probability paths for transforming between noise and data samples—which subsumes existing diffusion paths as specific instances. Interestingly, we find that employing FM with diffusion paths results in a more robust and stable alternative for training diffusion models. Furthermore, Flow Matching opens the door to training CNFs with other, non-diffusion probability paths. An

Wait, why "Flow Matching"

- Historical reasons (normalizing flows, continuous normalizing flows)

- **1:1 mapping** thanks to Lipschitz continuity of vector field!

$$dx = u_t(x)dt$$

Vector field $\longleftrightarrow$ **Flow**

Diffusion & Large Vision Models

Stanford University

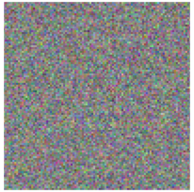
ICME

Training procedure

1. Sample:

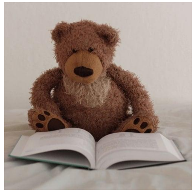
noise

$x_0 \sim \mathcal{N}(0, I)$



clean image

$x_1 \sim p_{\text{data}}$



time step

$t \sim \mathcal{U}(0, 1)$

noised image

$x_t = (1 - t)x_0 + tx_1$

Training procedure

1. Sample:

noise
$x_0 \sim \mathcal{N}(0, I)$

clean image
$x_1 \sim p_{\text{data}}$

time step
$t \sim \mathcal{U}(0, 1)$

noised image

$$x_t = (1 - t)x_0 + t x_1$$

2. Use x_t and t to **predict** $x_1 - x_0$ via $u_t^\theta(x_t)$

Training procedure

1. Sample:

noise $x_0 \sim \mathcal{N}(0, I)$

clean image $x_1 \sim p_{\text{data}}$

time step $t \sim \mathcal{U}(0, 1)$

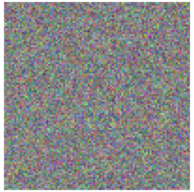
noised image $x_t = (1 - t)x_0 + tx_1$

2. Use x_t and t to **predict** $x_1 - x_0$ via $u_t^\theta(x_t)$

3. Compute **loss** $\mathcal{L} = \left\| u_t^\theta(x_t) - (x_1 - x_0) \right\|^2$ and **backpropagate** through u^θ

Diffusion & Large Vision Models

Inference

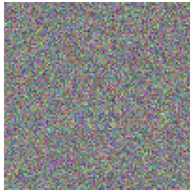
1. Sample: **noise** $x_0 \sim \mathcal{N}(0, I)$ 

Inference

1. Sample:

noise

$$x_0 \sim \mathcal{N}(0, I)$$



2. Use Euler to numerically solve the ODE:

$$x_{t_i} = x_{t_{i-1}} + u^\theta_{t_{i-1}}(x_{t_{i-1}})(t_i - t_{i-1})$$

Inference

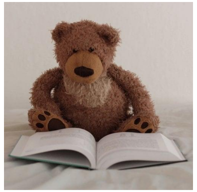
1. Sample: **noise**

$x_0 \sim \mathcal{N}(0, I)$

2. Use Euler to numerically solve the ODE:

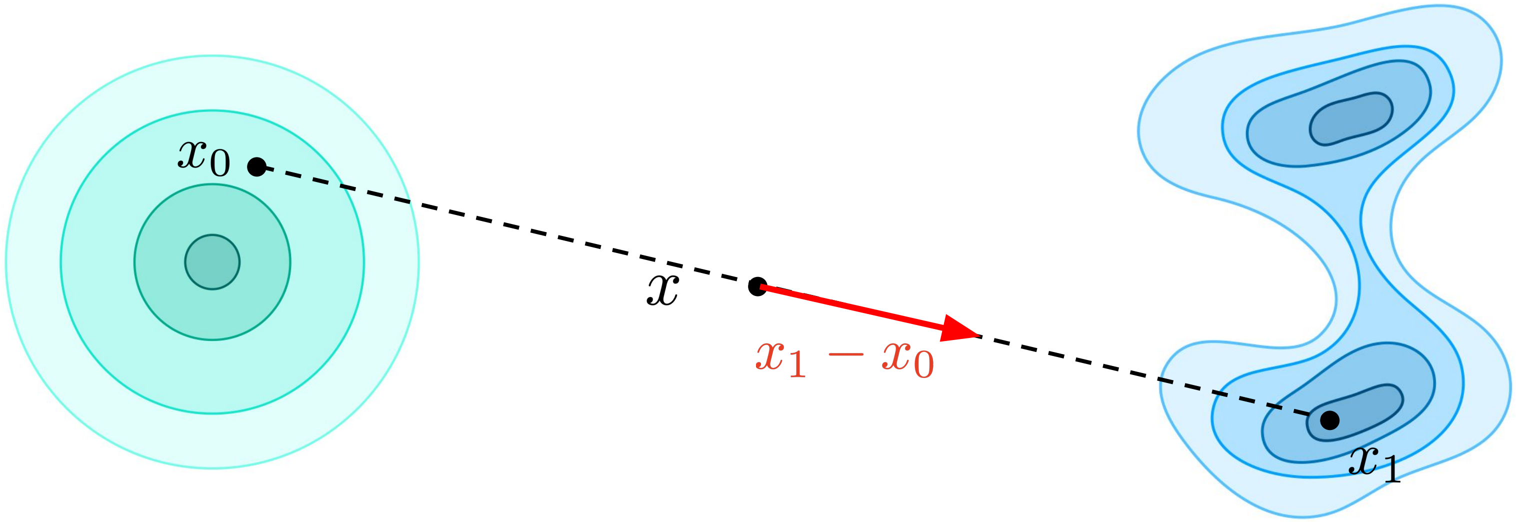
$$x_{t_i} = x_{t_{i-1}} + u^{\theta}_{t_{i-1}}(x_{t_{i-1}})(t_i - t_{i-1})$$

3. Obtain **final** image x_1



Diffusion & Large Vision Models

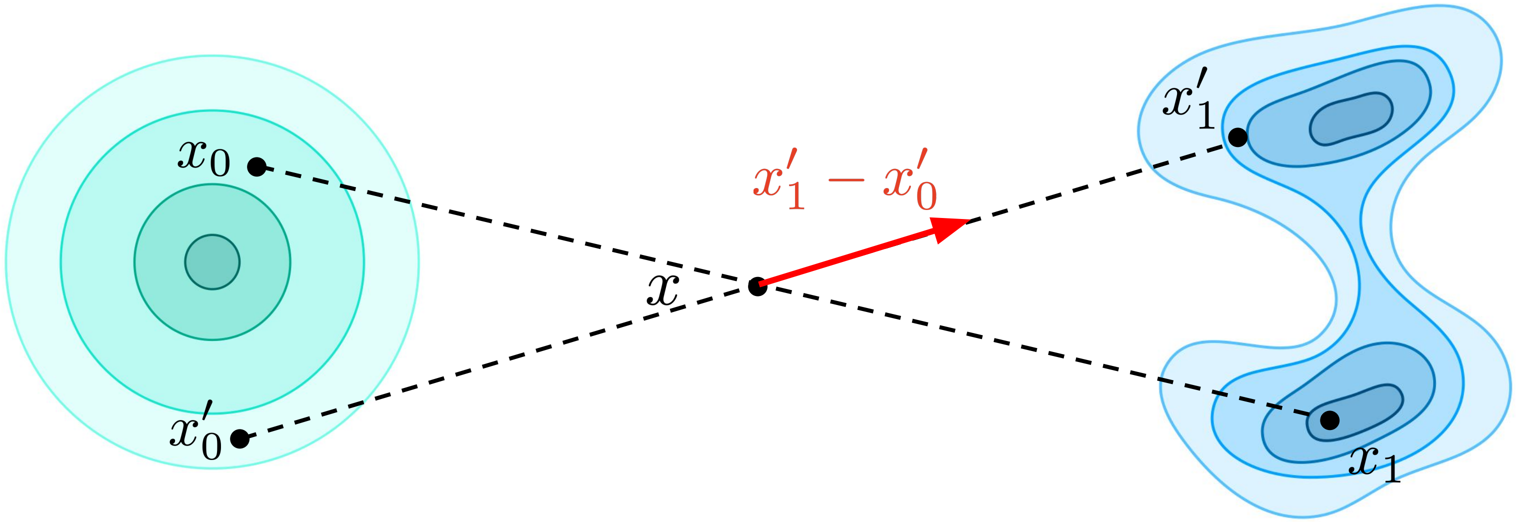
Recap of flow matching training



$p_0 = \mathcal{N}(0,1)$
$x_0 \sim \mathcal{N}(0,1)$

$p_1 = p_{\mathrm{data}}$
$x_1 \sim p_{\mathrm{data}}$

Recap of flow matching training

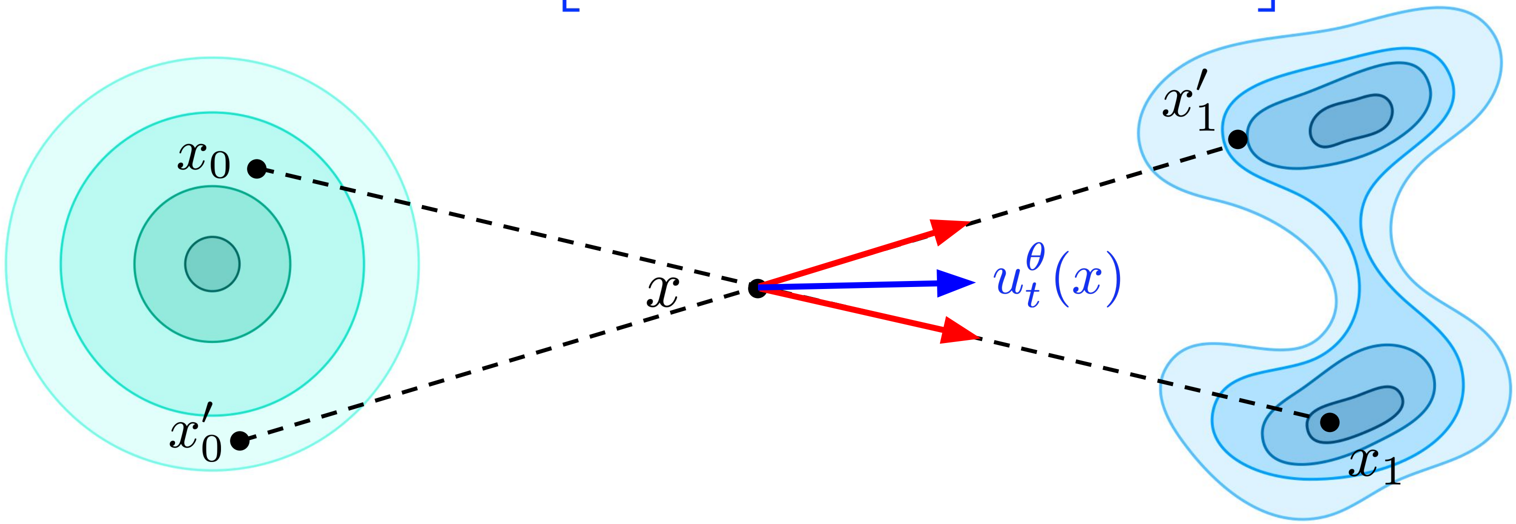


$p_0 = \mathcal{N}(0,1)$
$x_0 \sim \mathcal{N}(0,1)$
$x'_0 \sim \mathcal{N}(0,1)$

$p_1 = p_{\text{data}}$
$x_1 \sim p_{\text{data}}$
$x'_1 \sim p_{\text{data}}$

Recap of flow matching training

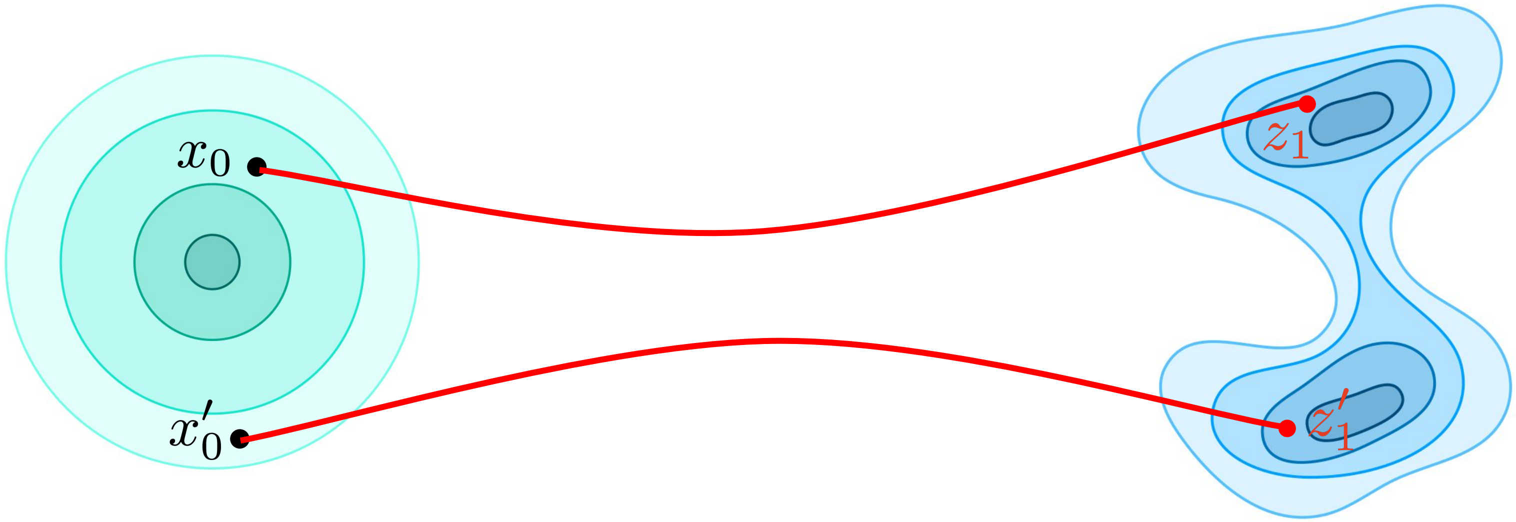
$$\mathcal{L}_{\text{CFM}} = \mathbb{E}_{t,x_1,x}\left[\left\|u_t^\theta(x) - u_t(x \mid x_1)\right\|^2\right]$$



$p_0 = \mathcal{N}(0,1)$
$x_0 \sim \mathcal{N}(0,1)$
$x_0' \sim \mathcal{N}(0,1)$

$p_1 = p_{\text{data}}$
$x_1 \sim p_{\text{data}}$
$x_1' \sim p_{\text{data}}$

Recap of flow matching training



$p_0 = \mathcal{N}(0,1)$
$x_0 \sim \mathcal{N}(0,1)$
$x_0' \sim \mathcal{N}(0,1)$

$p_1 = p_{\text{data}}$
$z_1 = \psi_1^\theta(x_0)$
$z_1' = \psi_1^\theta(x_0')$

Are we happy?

Learning complexity.

- Crossing paths lead to different learned rewiring
- Problem even when paths are not strictly crossing

Are we happy?

Learning complexity.

- Crossing paths lead to different learned rewiring
- Problem even when paths are not strictly crossing

Inefficiency at inference.

- Paths are not straight: needs more steps for approximation
- No magic solver to save the day

$$dx = \boxed{u_t^\theta(x)}dt$$

non-linear

Are we happy?

Learning complexity.

- Crossing paths lead to different learned rewiring
- Problem even when paths are not strictly crossing

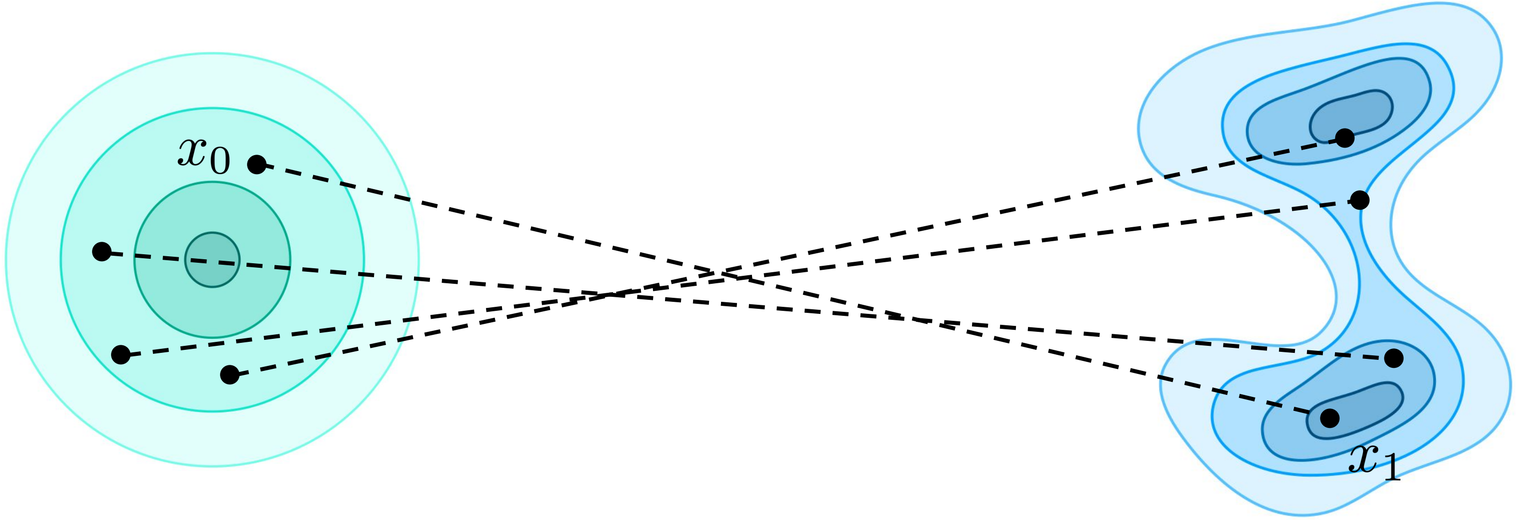
Not quite!

Inefficiency at inference.

- Paths are not straight: needs more steps for approximation
- No magic solver to save the day

Reflow procedure

Step 0. Train initial model: “1-rectified flow”



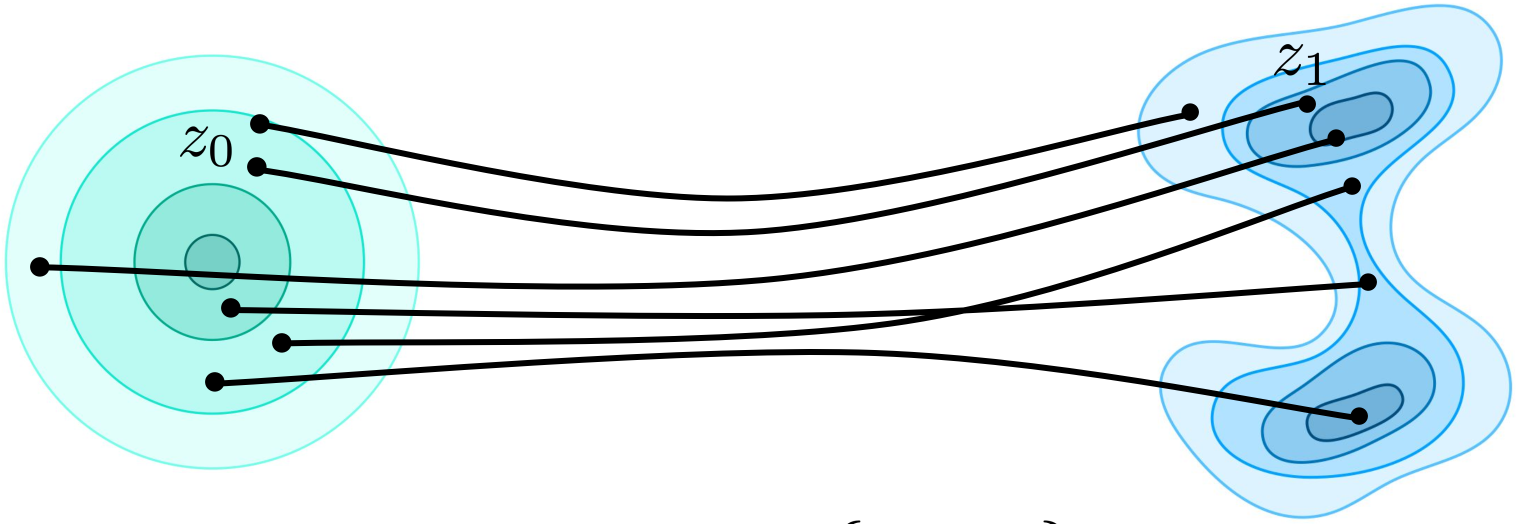
$x_0 \sim \mathcal{N}(0,1)$

$x_1 \sim p_{\text{data}}$

$$(X_0, X_1)$$

Reflow procedure

Step 1. Create paired data out of "1-rectified flow" model

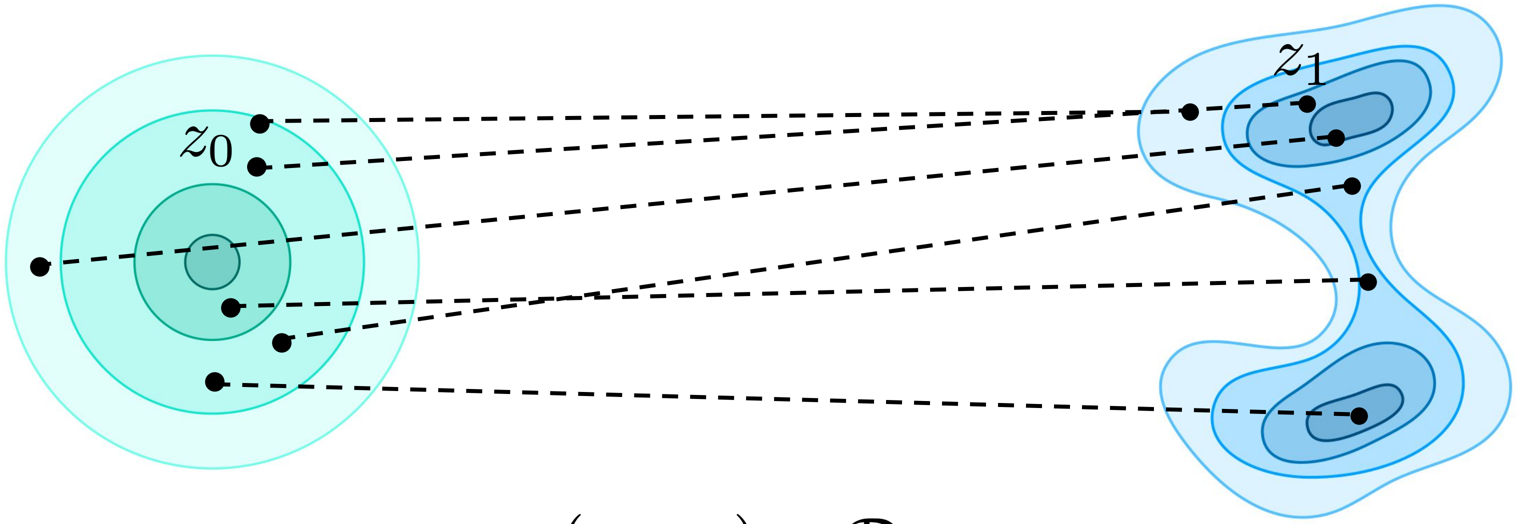


$$\mathcal{D}_{\text{train}} = \cup\{z_0, z_1\}$$
$$z_0 \sim \mathcal{N}(0,1)$$
$$z_1 = \psi_1^{\theta}(z_0)$$

Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow, Liu et al., 2022.

Reflow procedure

Step 2. Train on paired data: “2-rectified flow”



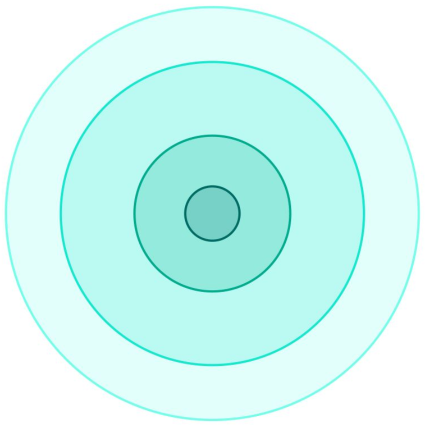
$$(z_0, z_1) \sim \mathcal{D}_{\text{train}}$$

$$(Z_0, Z_1)$$

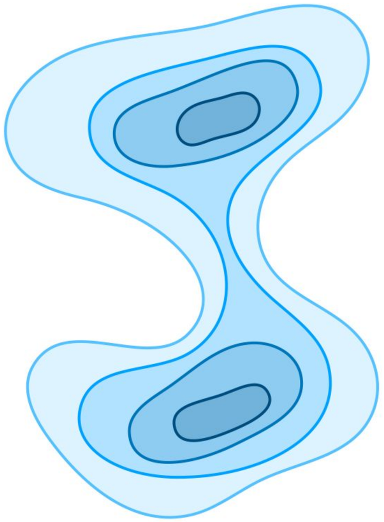
Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow, Liu et al., 2022.

Reflow procedure

Further steps. Repeat steps 2-3 as desired



•••



$$(Z_0^k, Z_1^k)$$

Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow, Liu et al., 2022.

Why does it work?

Property. Resulting mapping still follows target distribution p_{data}

Proof. Define $v^X(x,t) = E[\dot{X}_t | X_t = x]$ $\quad Z_t = X_0 + \int_0^t v^X(Z_s, s)ds$

Derivation: chain rule + law of total expectation

$$\frac{\partial p_t^X}{\partial t} + \nabla \cdot (v_t^X p_t^X) = 0 \qquad \frac{\partial p_t^Z}{\partial t} + \nabla \cdot (v_t^X p_t^Z) = 0$$

Derivation: uniqueness theorem with initial conditions

$$\forall t \in [0,1], \quad p_t^X = p_t^Z$$

Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow, Liu et al., 2022.

Why does it work?

Property. Paths are provably straighter at each reflow.

Proof. Define straightness $S(Z) = \int_0^1 E\left[||(Z_1 - Z_0) - \dot{Z}_t||^2\right] dt$

Derivation: variance decomposition mechanism

$$S(Z^{k+1}) \leq E[||Z_1^k - Z_0^k||^2] - E[||Z_1^{k+1} - Z_0^{k+1}||^2]$$

Derivation: telescoping argument

$$\min_{k \leq K} S(Z^k) \leq O(1/K)$$

Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow, Liu et al., 2022.

Discussion

- Reflow procedure done 1 or 2 times in practice, otherwise, too many errors
- We can afford using simpler techniques at inference time, e.g. Euler
- Quality/speed trade-off

Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow, Liu et al., 2022.

Diffusion & Large Vision Models

Motivation

Flow matching

Training

Inference

Rectified flow

Discussion

Stanford University

ICME

Paradigms seen so far

Discrete-time diffusion.

Score-based diffusion.

?

Flow matching.

Forward process

Discrete-time diffusion. $x_t = \alpha_t x_0 + \sigma_t \epsilon$

Score-based diffusion. $dx = \boxed{f(x,t)} dt + g(t) dW$

$f(t)x$

$$x_t = x_0 \exp\left(\int_0^t f(u)\,du\right) + \sqrt{\int_0^t g(s)^2 \exp\left(2\int_s^t f(u)\,du\right) ds} \cdot \epsilon$$

Flow matching. $x_t = (1-t)x_0 + t x_1$

Forward process

Discrete-time diffusion.

$$x_t = \alpha_t \underbrace{\boxed{x_0}}_{\text{clean sample}}^{t=0} + \sigma_t \epsilon$$

Score-based diffusion.

$$x_t = \underbrace{\boxed{x_0}}_{\text{clean sample}}^{t=0} \exp\left(\int_0^t f(u)\,du\right) + \sqrt{\int_0^t g(s)^2 \exp\left(2\int_s^t f(u)\,du\right) ds} \cdot \epsilon$$

Flow matching.

$$x_t = (1-t)x_0 + t\underbrace{\boxed{x_1}}_{\text{clean sample}}^{t=1}$$

Forward process

Discrete-time diffusion. $$x_t = \alpha_t x_0 + \overbrace{\boxed{\sigma_t \epsilon}}^{t=T}_{\text{noise}}$$

Score-based diffusion.

$$x_t = x_0 \exp\left(\int_0^t f(u)\,du\right) + \sqrt{\int_0^t g(s)^2 \exp\left(2\int_s^t f(u)\,du\right)\,ds} \cdot \overbrace{\boxed{\epsilon}}^{t=T}_{\text{noise}}$$

Flow matching. $$x_t = (1-t)\overbrace{\boxed{x_0}}^{t=0}_{\text{noise}} + t x_1$$

What's learned

Discrete-time diffusion. $$\mathbb{E}_{t,x_0,\epsilon}\left[||\boxed{\epsilon_\theta}(\alpha_t x_0 + \sigma_t \epsilon, t) - \epsilon||^2\right]$$

noise

Score-based diffusion.

$$\mathbb{E}_{t,x_0,\epsilon}\left[||\boxed{s_\theta}(\alpha_t x_0 + \sigma_t \epsilon, t) - \nabla \log p(\alpha_t x_0 + \sigma_t \epsilon | x_0)||^2\right]$$

score

Flow matching. $$\mathbb{E}_{t,x_1,x}\left[||\boxed{u_t^\theta}(x) - u_t(x|x_1)||^2\right]$$

velocity

Generation process

Discrete-time diffusion.

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\epsilon_\theta(x_t, t)\right) + \sigma_t \epsilon$$

Score-based diffusion.

$$dx = \left[f(t)x - g(t)^2 \nabla_x \log p_t(x)\right] dt + g(t) d\overline{W}$$

Flow matching.

$$dx = u_t(x) dt$$

Deterministic counterpart

Discrete-time diffusion. DDIM

$$x_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\,\hat{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}}\,\epsilon_\theta(x_t, t) + \varnothing$$

Score-based diffusion. PF-ODE

$$dx = \left[f(t)x - \frac{1}{2}g(t)^2 \nabla_x \log p_t(x)\right] dt + \varnothing$$

Flow matching. Already deterministic!

$$dx = u_t(x)dt + \varnothing$$

Stochastic interpolants

Journal of Machine Learning Research 26 (2025) 1-80 Submitted 12/23; Revised 9/25; Published 9/25

Stochastic Interpolants: A Unifying Framework for Flows and Diffusions

Michael S. Albergo* ALBERGO@NYU.EDU
Center for Cosmology and Particle Physics
New York University
New York, NY 10012, USA

Nicholas M. Boffi* BOFFI@CIMS.NYU.EDU
Courant Institute of Mathematical Sciences
New York University
New York, NY 10012, USA

Eric Vanden-Eijnden EVE2@CIMS.NYU.EDU
Courant Institute of Mathematical Sciences
New York University
New York, NY 10012, USA

Thank you for your attention!