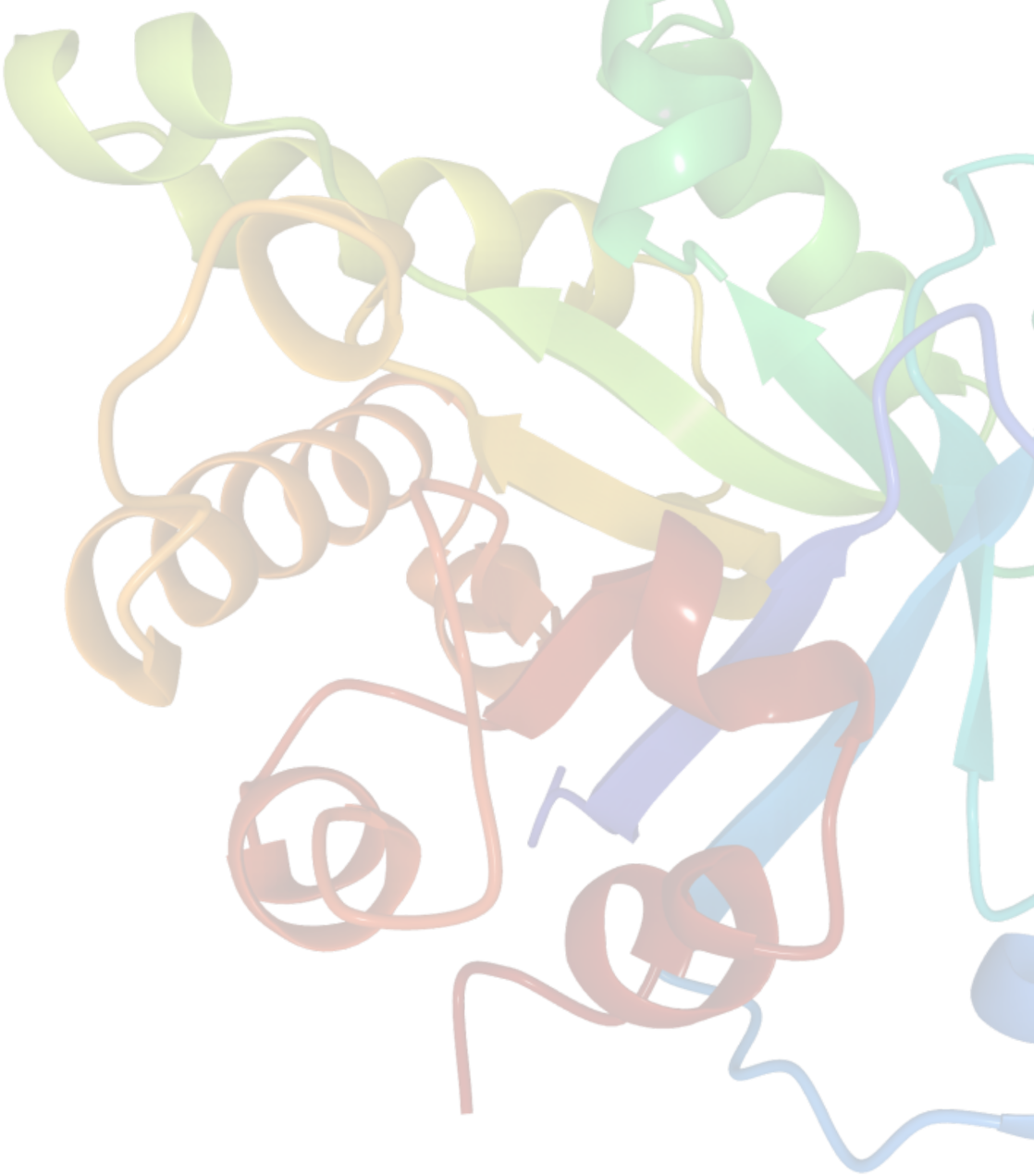
Flow Matching

Alex Tong

Probabilistic AI Summer School 2025



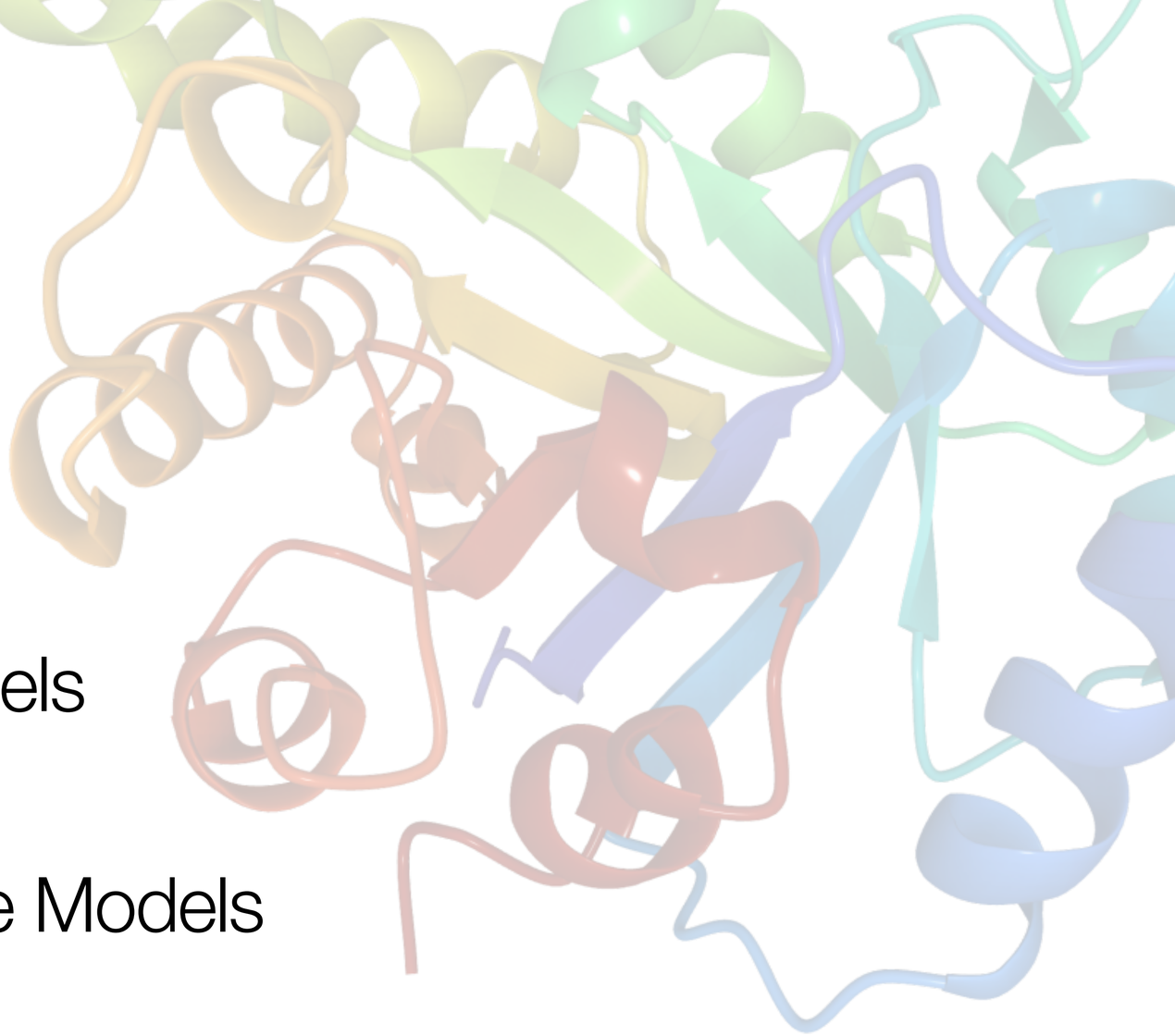
Slides + Code based on work with Joey Bose, Heli Ben-Hamu, Kilian Fatras

Outline:

Part I: Dynamical Systems and Simulation-based Generative Models

Part II: Simulation-Free Generative Models

Part III: Extensions

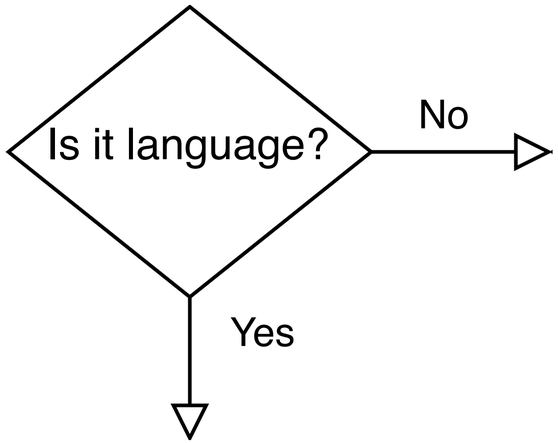


What do deep learning people mean by generative models?

Models trained to generate samples from the data

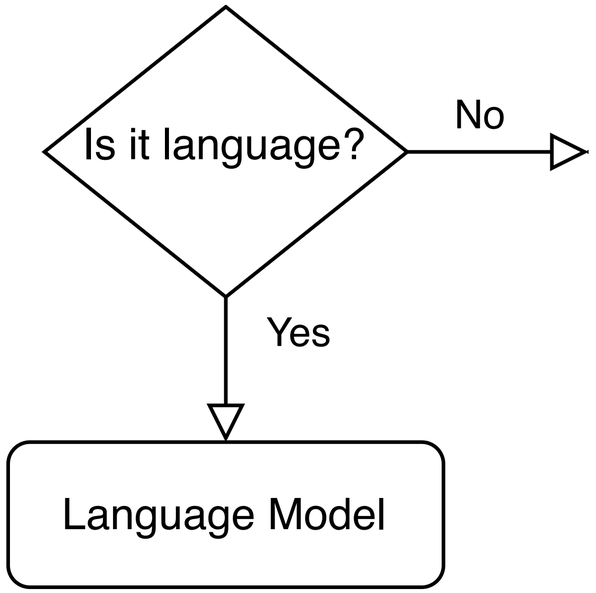
What do deep learning people mean by generative models?

Models trained to generate samples from the data



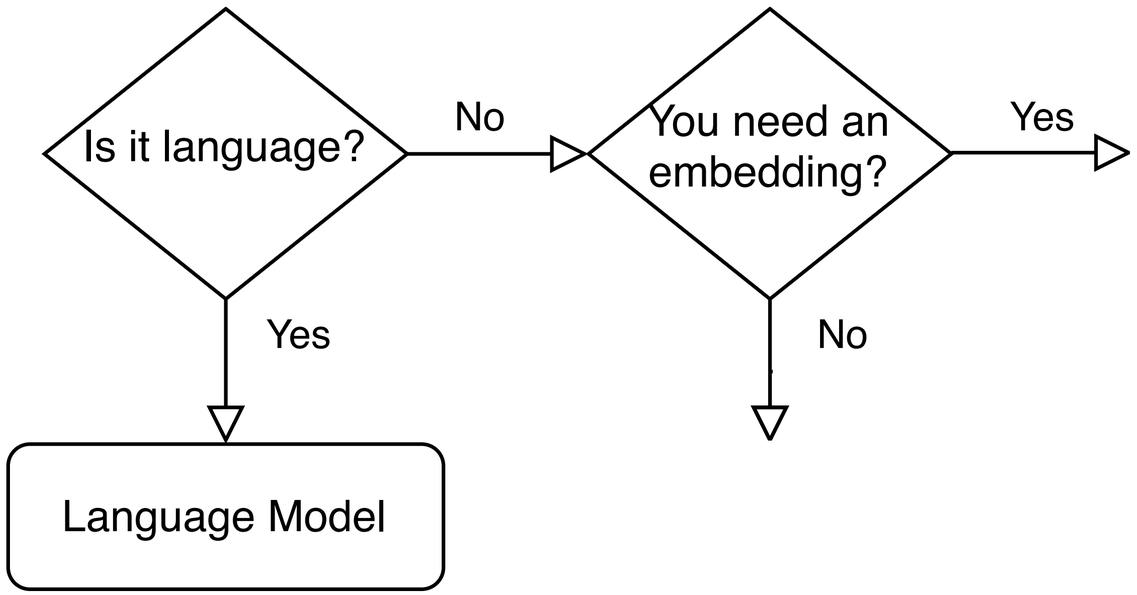
What do deep learning people mean by generative models?

Models trained to generate samples from the data



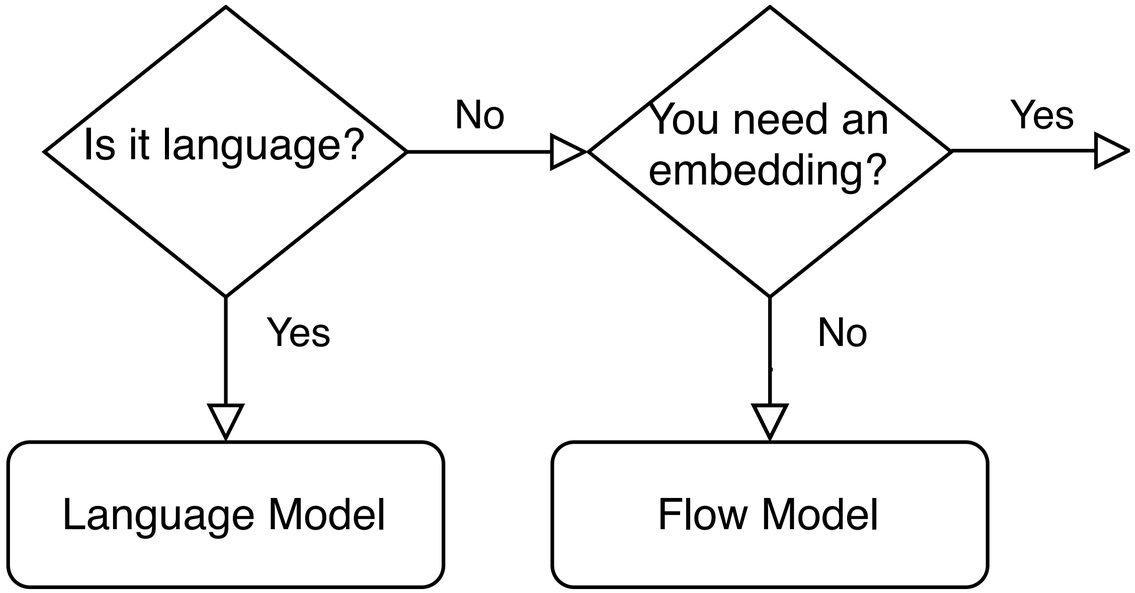
What do deep learning people mean by generative models?

Models trained to generate samples from the data



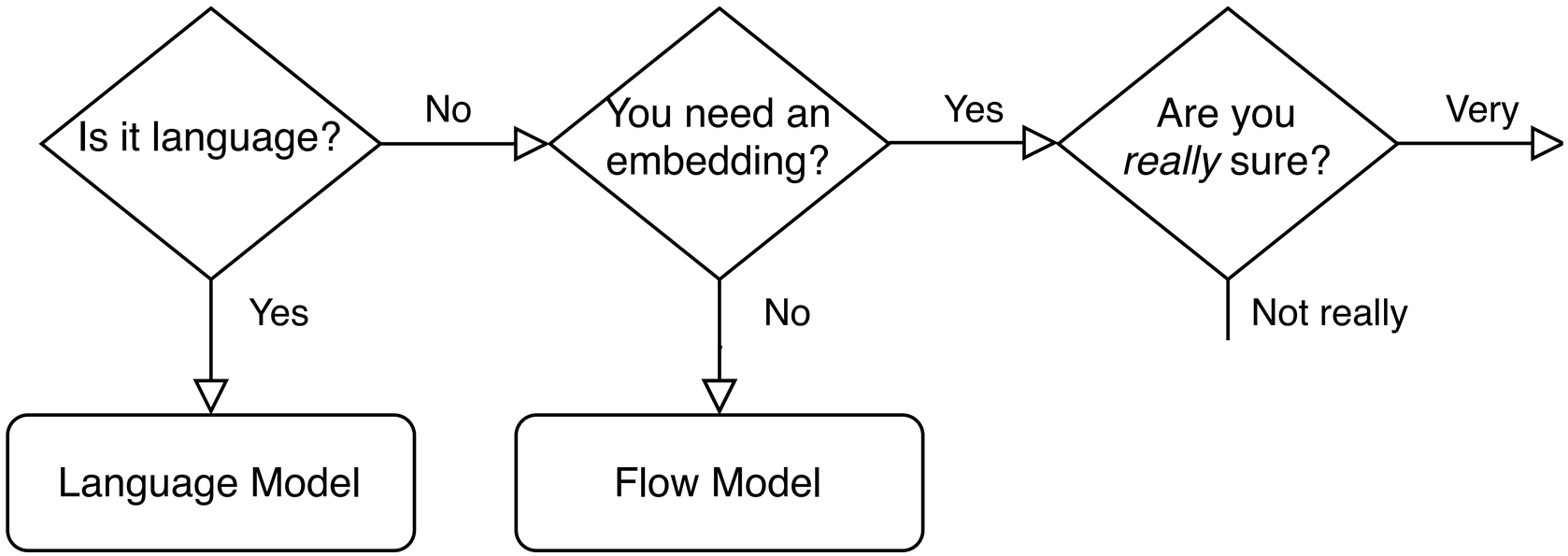
What do deep learning people mean by generative models?

Models trained to generate samples from the data



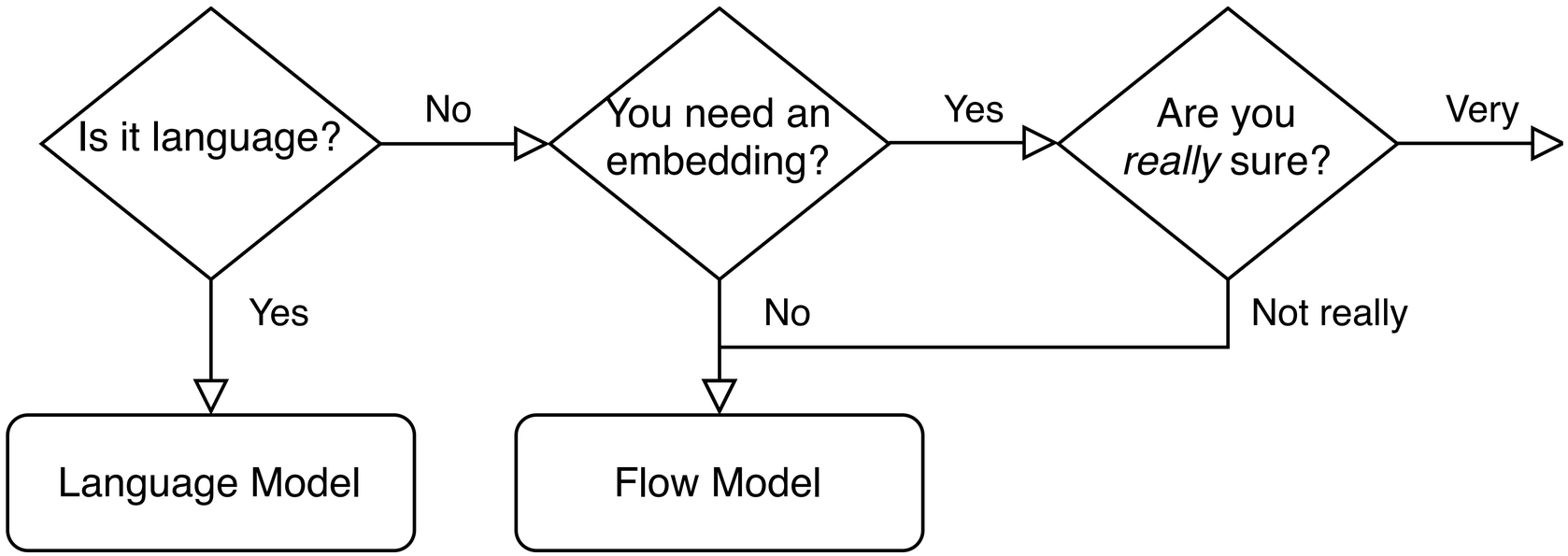
What do deep learning people mean by generative models?

Models trained to generate samples from the data



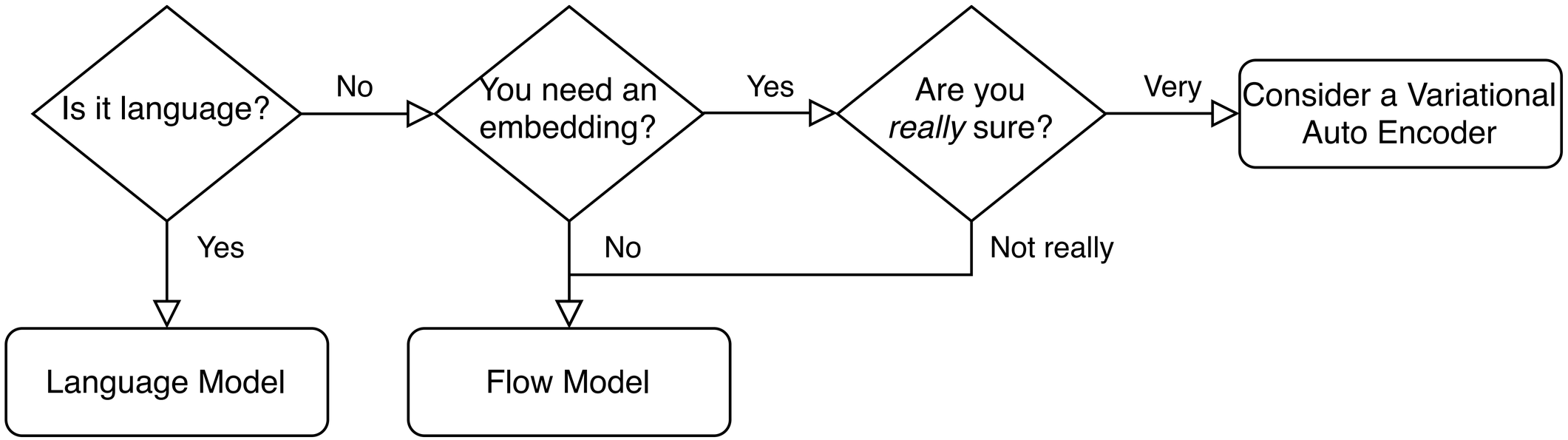
What do deep learning people mean by generative models?

Models trained to generate samples from the data



What do deep learning people mean by generative models?

Models trained to generate samples from the data



Flow Matching

- Image generation
- Video generation
- Text to speech
- Audio generation
- Molecule generation
- Protein design

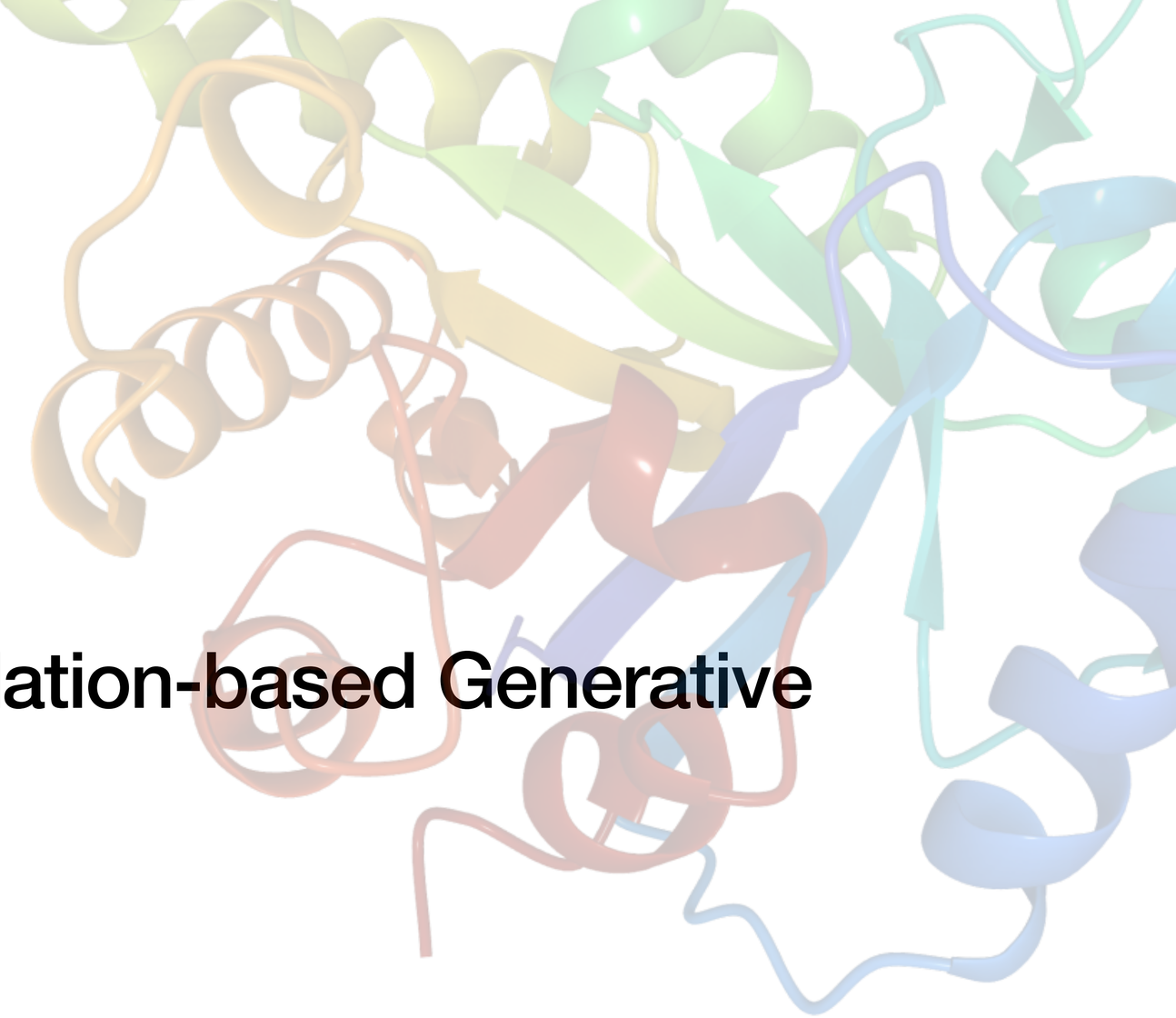
atong01/**conditional-flow-matching** Public

TorchCFM: a Conditional Flow Matching library

Python ☆ 1.8k 141

Generated images with OT-CFM at iteration 40k (FID: 47.4)

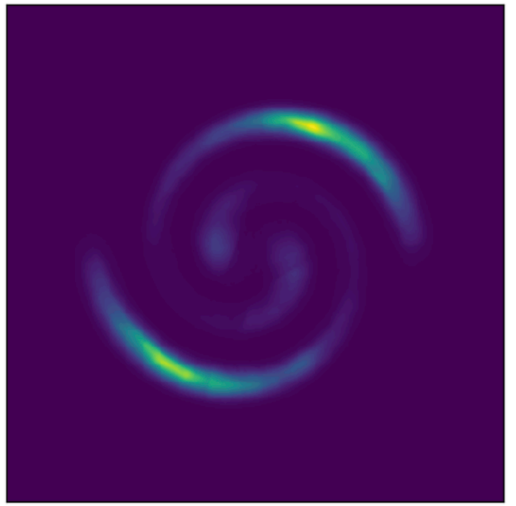
Tong et al. TMLR 2024



Part I:
Dynamical Systems and Simulation-based Generative Models

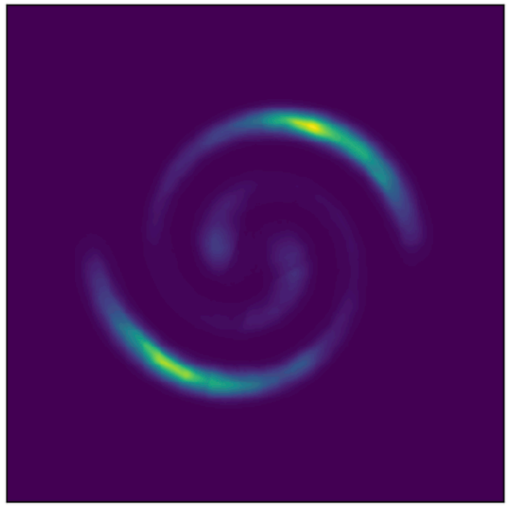
Problem Setting: Generative Modeling

- Unknown: data distribution q



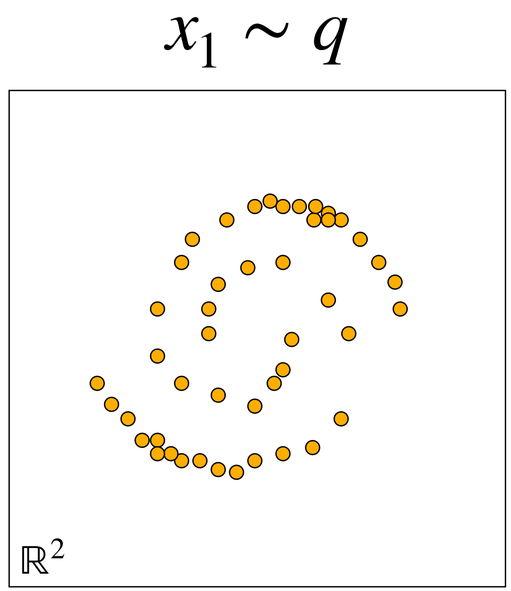
Problem Setting: Generative Modeling

- Unknown: data distribution q
- Given: samples $x_1 \sim q$



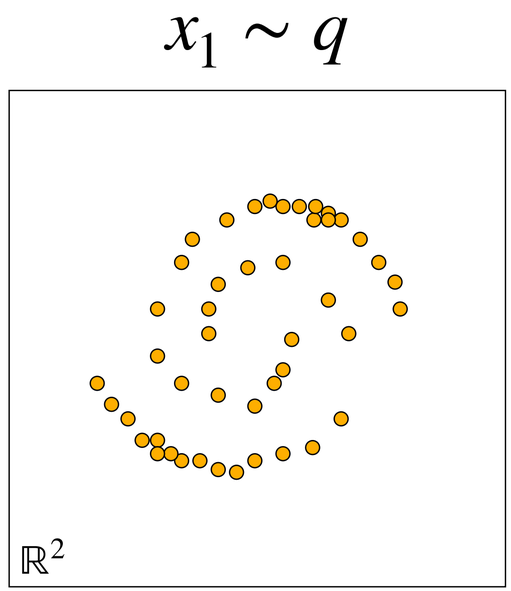
Problem Setting: Generative Modeling

- Unknown: data distribution q
- Given: samples $x_1 \sim q$



Problem Setting: Generative Modeling

- Unknown: data distribution q
- Given: samples $x_1 \sim q$

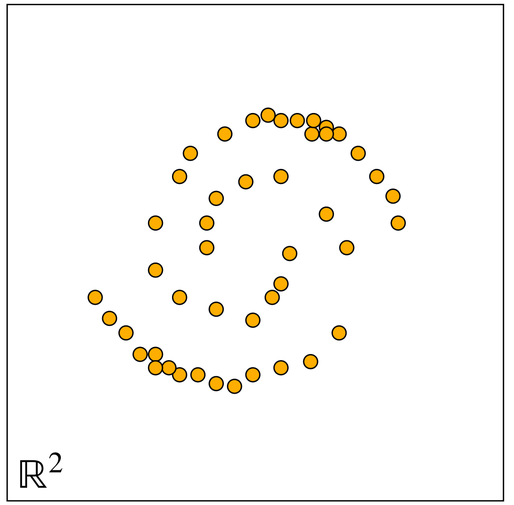


Goal: learn *a sampler* from the unknown q

Deep Generative Modeling

- Unknown: data distribution q
- Given: samples $x_1 \sim q$

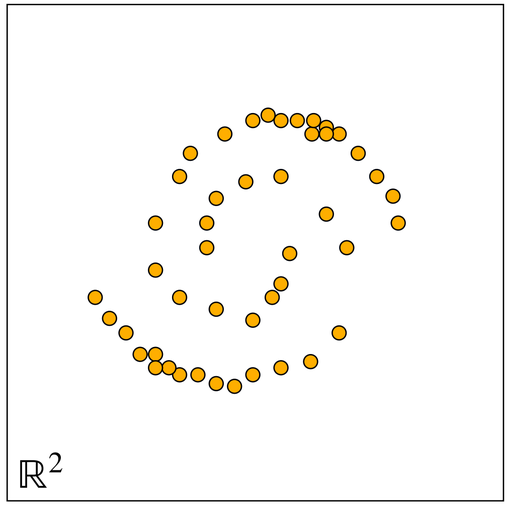
$x_1 \sim q$



Deep Generative Modeling

- Unknown: data distribution q
- Given: samples $x_1 \sim q$
- Learn: neural network with parameters θ

$x_1 \sim q$



Generative Model

(ψ_θ , p_θ)

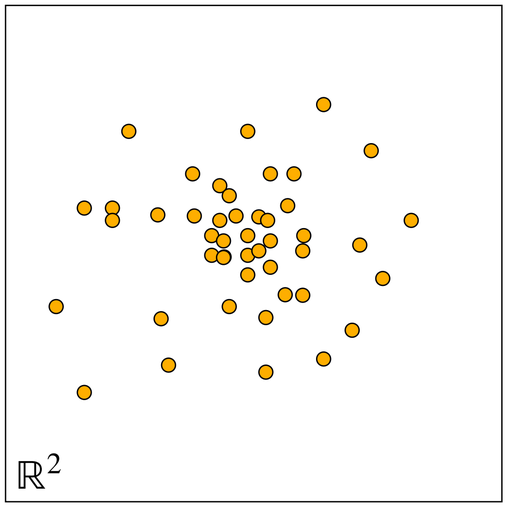
Generator Underlying Density

Goal: find parameters θ s.t. $p_\theta \approx q$

Deep Generative Modeling

Easy to sample from

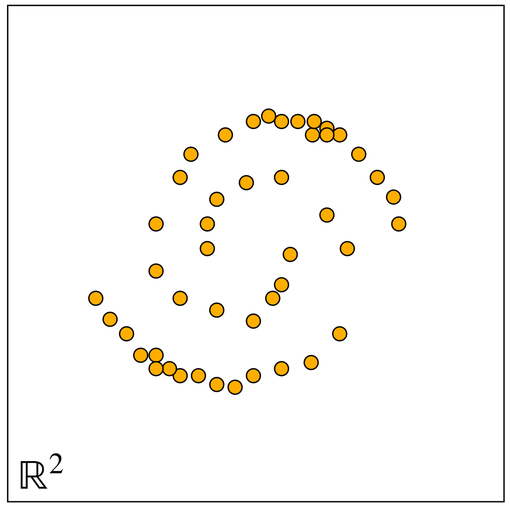
$x_0 \sim p$



Generator

ψ_θ

$x_1 \sim q$

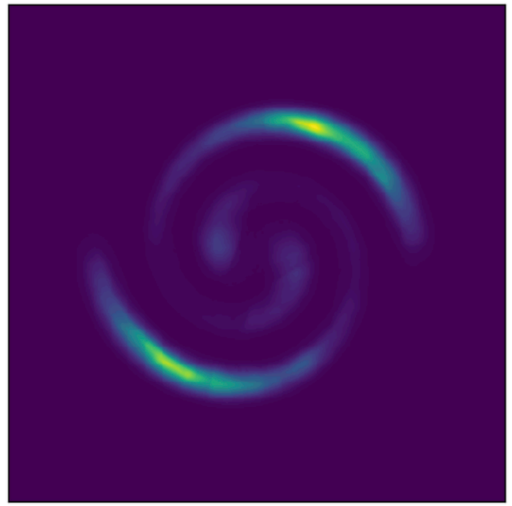


Sampling

$x_0 \sim p$

$\psi_\theta(x_0) \sim q$

How to model ψ_θ?



Density Estimation

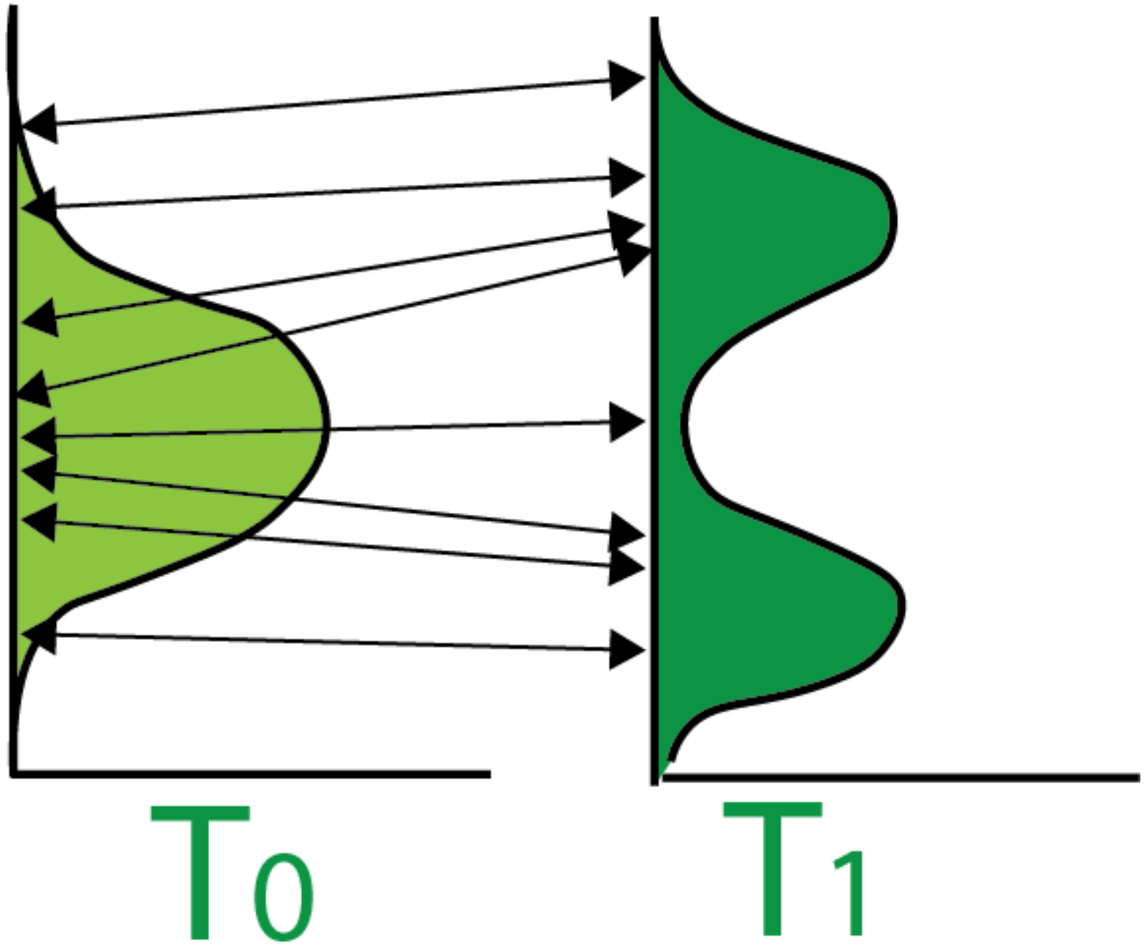
$p_\theta \approx q$

Normalizing Flows (NFs)

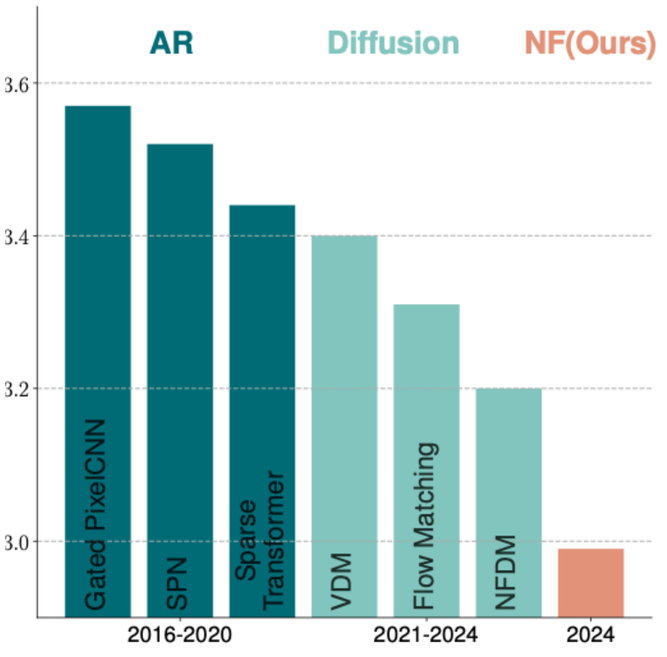
Sample from some complicated distribution by sampling from a simple distribution then applying U

- Begin with a simple distribution

$$p_{t_0}(x) \sim \mathcal{N}(0, 1)$$



Normalizing Flows (NFs)



Normalizing Flows are Capable Generative Models

Shuangfei Zhai [1] **Ruixiang Zhang** [1] **Preetum Nakkiran** [1] **David Berthelot** [1] **Jiatao Gu** [1] **Huangjie Zheng** [1]
Tianrong Chen [1] **Miguel Angel Bautista** [1] **Navdeep Jaitly** [1] **Josh Susskind** [1]

Normalizing Flows (NFs)

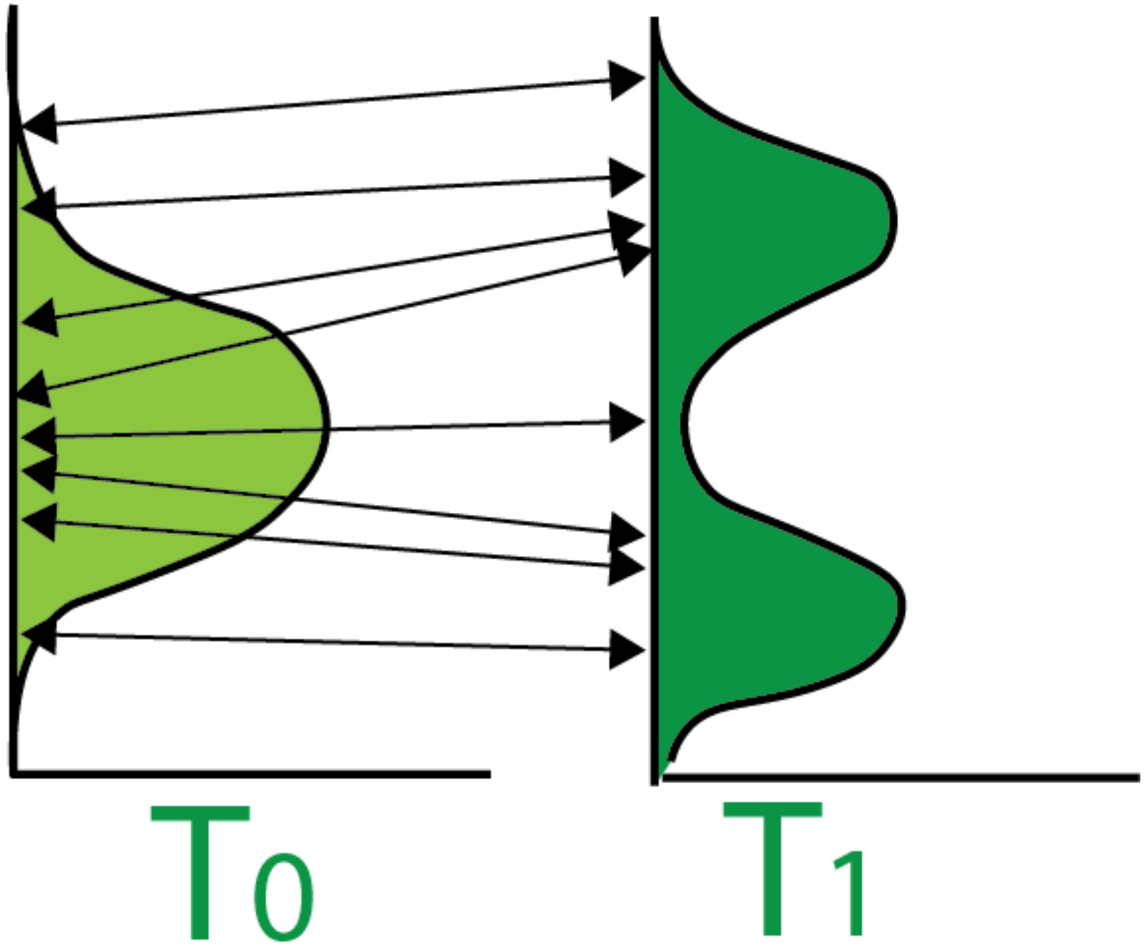
Sample from some complicated distribution by sampling from a simple distribution then applying U

- Begin with a simple distribution

  $p_{t_0}(x) \sim \mathcal{N}(0, 1)$

- Apply an invertible transformation(s)

  $x_{t_1} = U(x_{t_0})$



Normalizing Flows (NFs)

Sample from some complicated distribution by sampling from a simple distribution then applying U

- Begin with a simple distribution

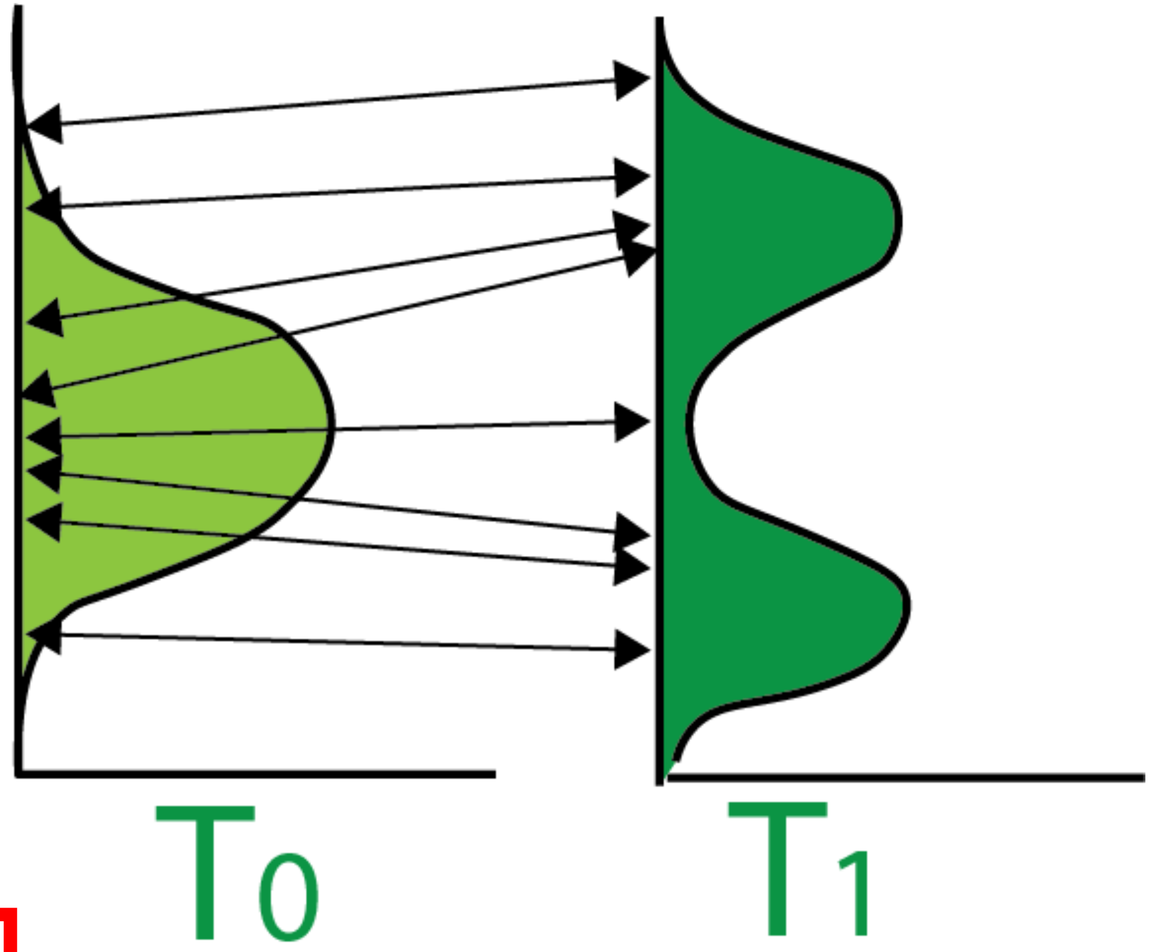
$$p_{t_0}(x) \sim \mathcal{N}(0, 1)$$

- Apply an invertible transformation(s)

$$x_{t_1} = U(x_{t_0})$$

- Use change of variables to calculate probability

$$\log p_{t_1}(x_{t_1}) = \log p_{t_0}(x_{t_0}) - \log \det \left| \frac{\partial U}{\partial x_{t_0}} \right|$$



Deep Normalizing Flows (NFs)

Apply a series of transformations

$$x_{t_1} = U(x_{t_0}) \Rightarrow x_{t_N} = u_N \circ u_{N-1} \circ \cdots \circ u_1(x_{t_0})$$

Use change of variables to calculate probability

$$\log p_{t_1}(x_{t_1}) = \log p_{t_0}(x_{t_0}) - \log \det \left| \frac{\partial U}{\partial x_{t_0}} \right| \Rightarrow \log p_{t_N}(x_{t_N}) = \log p_{t_0}(x_{t_0}) - \sum_{n=1}^{N} \log \det \left| \frac{\partial u_n}{\partial x_{t_{n-1}}} \right|$$

Continuous Normalizing Flows

Apply a series of transformations

$$x_{t_N} = u_N \circ u_{N-1} \circ \cdots \circ u_1(x_{t_0}) \quad \Rightarrow \quad x_{t_1} = U(x_{t_0}) = \int_{t_0}^{t_1} u(x(t), t)\mathrm{d}t$$

Use change of variables to calculate probability

$$\log p_{t_N}(x_{t_N}) = \log p_{t_0}(x_{t_0}) - \sum_{n=1}^{N} \log \det \left| \frac{\partial u_n}{\partial x_{t_{n-1}}} \right| \quad \Rightarrow \quad \log p_{t_1}(x_{t_1}) = \log p_{t_0}(x_{t_0}) - \int_{t_0}^{t_1} \mathrm{Tr}\left(\frac{\partial u}{\partial x(t)} \right) \mathrm{d}t$$

[Chen et al. 2018; Grathwohl et al. 2018]

Continuous Normalizing Flows

Apply a series of transformations

$$x_{t_N} = u_N \circ u_{N-1} \circ \cdots \circ u_1(x_{t_0}) \quad \Rightarrow \quad x_{t_1} = U(x_{t_0}) = \int_{t_0}^{t_1} u(x(t), t)\mathrm{d}t$$

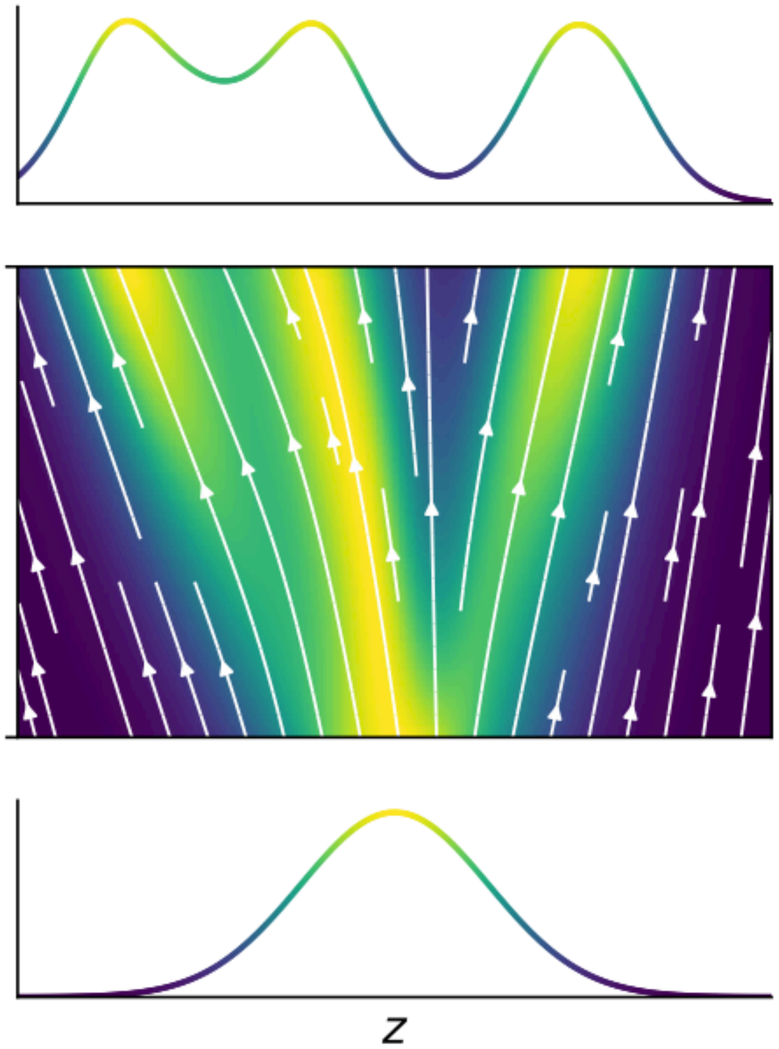
Use change of variables to calculate probability

$$\log p_{t_N}(x_{t_N}) = \log p_{t_0}(x_{t_0}) - \sum_{n=1}^{N} \log \det \left| \frac{\partial u_n}{\partial x_{t_{n-1}}} \right| \quad \Rightarrow \quad \log p_{t_1}(x_{t_1}) = \log p_{t_0}(x_{t_0}) - \int_{t_0}^{t_1} \mathrm{Tr}\left(\frac{\partial u}{\partial x(t)} \right) \mathrm{d}t$$

Constructs an invertible transformation ψ for any f!

[Chen et al. 2018; Grathwohl et al. 2018]

Continuous Normalizing Flows



A ons

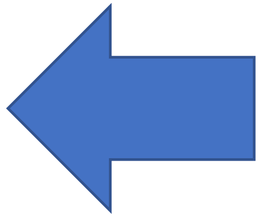
x_t $;t_0)$ $\Rightarrow$ $x_{t_1} = U(x_{t_0}) = \int_{t_0}^{t_1} u(x(t), t)\mathrm{d}t$

U culate probability

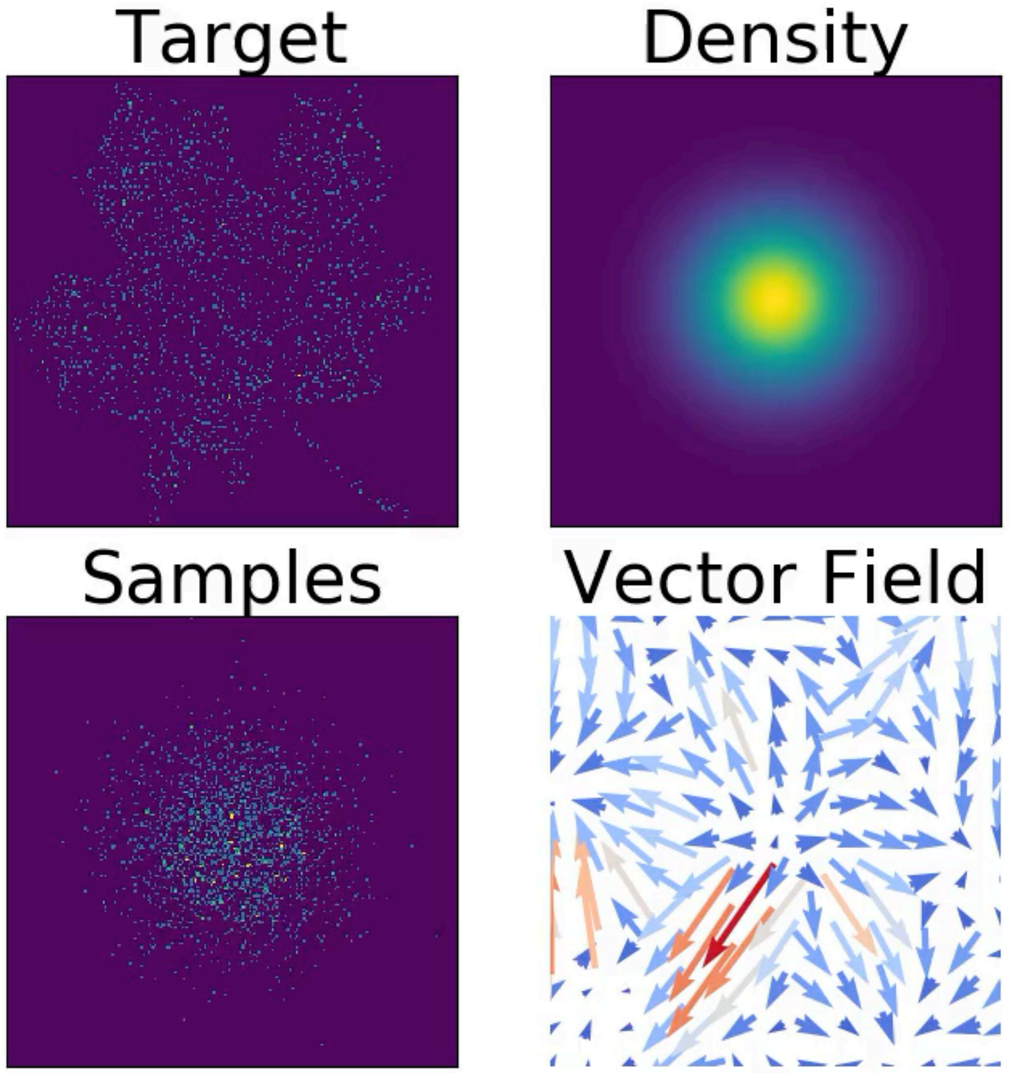
$\log p_{t_N}($ $\frac{\partial u_n}{x_{t_{n-1}}}\Big| \Rightarrow \log p_{t_1}(x_{t_1}) = \log p_{t_0}(x_{t_0}) - \int_{t_0}^{t_1} \mathrm{Tr}\left(\frac{\partial u}{\partial x(t)}\right)\mathrm{d}t$

[Chen et al. 2018; Grathwohl et al. 2018]

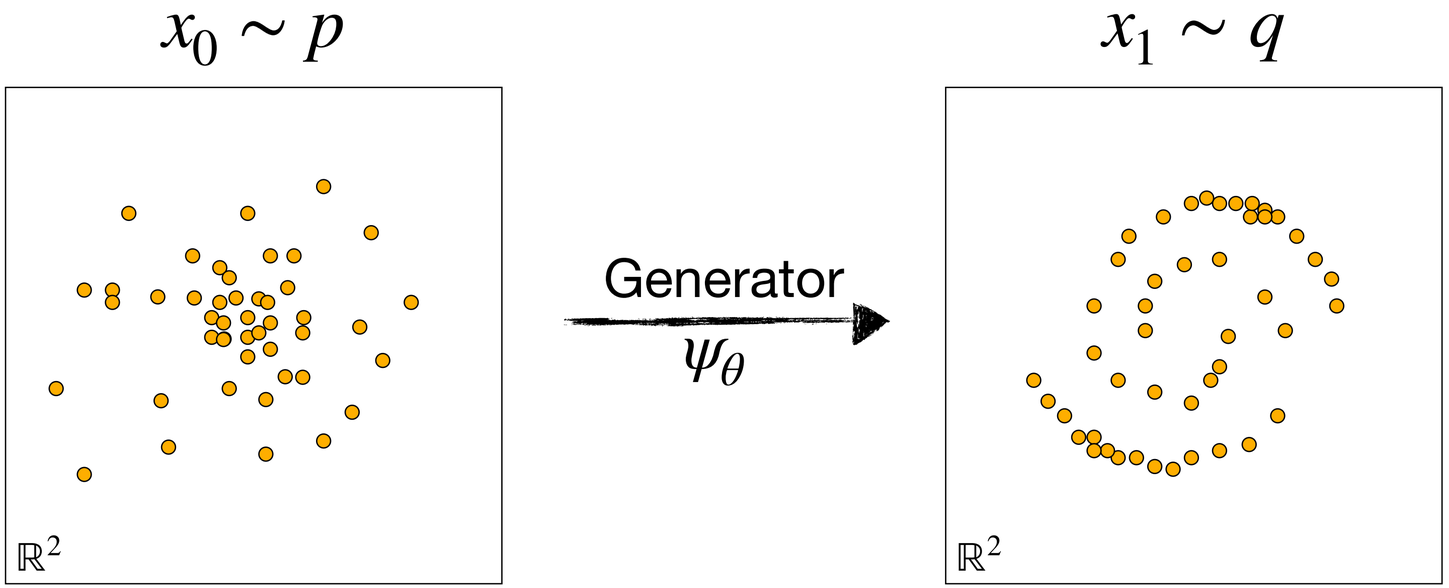
What does this look like?



What does this look like?

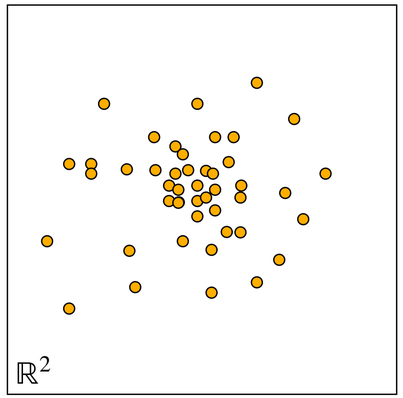


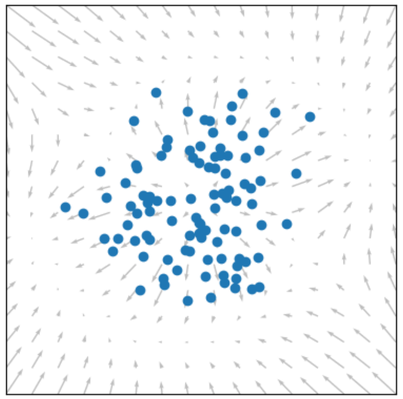
Focus: Generative Models as Dynamical Systems



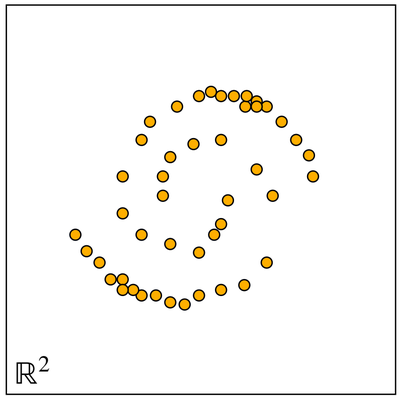
Focus: Dynamical Systems as Generative Models

$x_0 \sim p$





$x_1 \sim q$



$$\psi_t : [0,1] \times \mathbb{R}^2 \to \mathbb{R}^2$$

Time-Dependent Generator

Sampling = Simulating

$$x_0 \sim p \quad \rightarrow \quad \text{simulate}(x_0, t) = \psi_t(x_0)$$

Deterministic and Stochastic Dynamics

Flows

ODE $$dx_t = u_t(x_t)dt$$

Velocity field

Diffusion

SDE $$dx_t = f_t(x_t)dt + g_t dw_t$$

Drift

Diffusion Coefficient

Brownian Motion

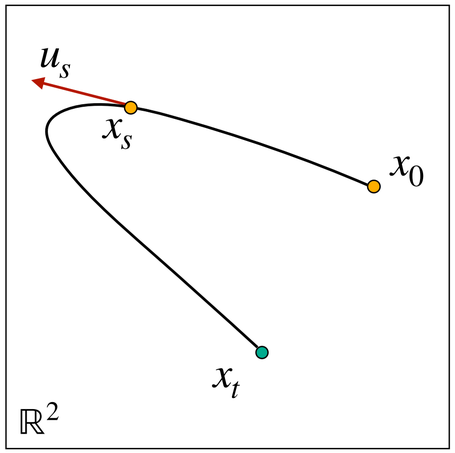
Deterministic and Stochastic Dynamics

Flows

ODE $$dx_t = u_t(x_t)dt$$

Velocity field

Deterministic



$$x_t = x_0 + \int_0^t u_s(x_s)ds$$

Diffusion

SDE $$dx_t = f_t(x_t)dt + g_t dw_t$$

Drift

Diffusion Coefficient

Brownian Motion

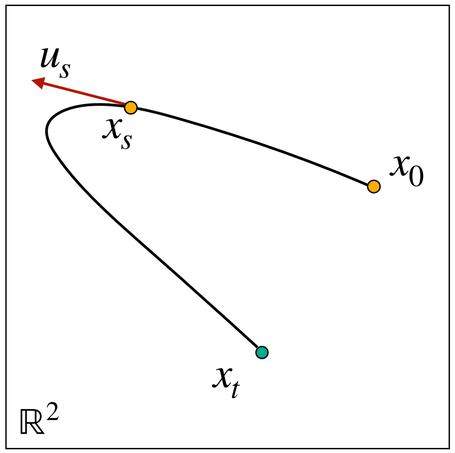
Deterministic and Stochastic Dynamics

Flows

ODE $$dx_t = u_t(x_t)dt$$

Velocity field

Deterministic



$$x_t = x_0 + \int_0^t u_s(x_s)ds$$

Diffusion

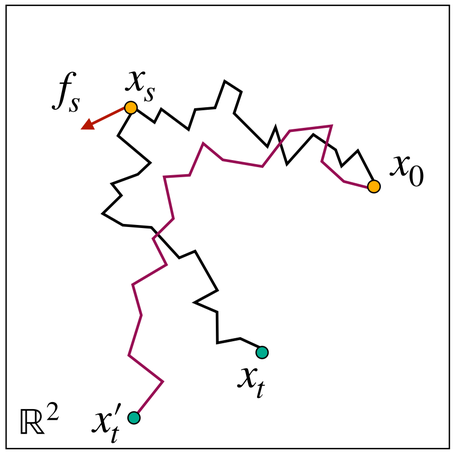
SDE $$dx_t = f_t(x_t)dt + g_t dw_t$$

Drift

Diffusion Coefficient

Brownian Motion

Stochastic



$$x_t = x_0 + \int_0^t f_s(x_s)ds + \int_0^t g_s dw_s$$

Deterministic and Stochastic Dynamics

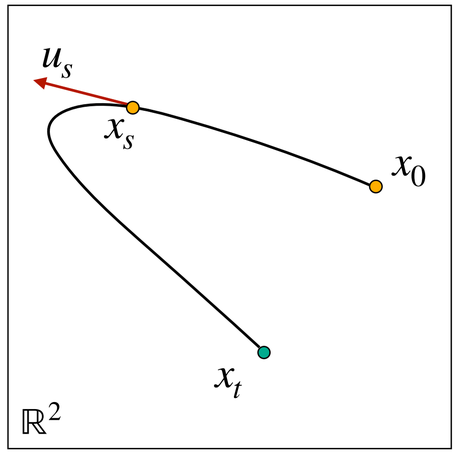
Flows

ODE $$dx_t = u_t(x_t)dt$$

Velocity field

$$x_{t+\Delta} = x_t + u_t(x_t)\Delta$$

Deterministic



$$x_t = x_0 + \int_0^t u_s(x_s)ds$$

Diffusion

SDE $$dx_t = f_t(x_t)dt + g_t dw_t$$

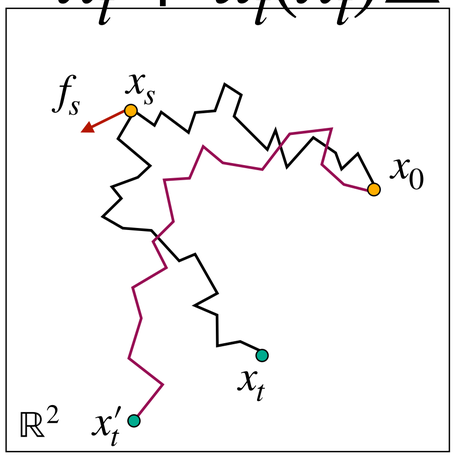
Drift

Diffusion Coefficient

Brownian Motion

$$x_{t+\Delta} = x_t + u_t(x_t)\Delta + g_t\epsilon\sqrt{\Delta}$$

Stochastic



$$x_t = x_0 + \int_0^t f_s(x_s)ds + \int_0^t g_s dw_s$$

Where are the probabilities?

Flows

ODE $$dx_t = u_t(x_t)dt$$

Velocity field

The Continuity Equation

$$\partial_t p_t = -\operatorname{div}(p_t u_t)$$

Diffusion

SDE $$dx_t = f_t(x_t)dt + g_t dw_t$$

Drift

Diffusion Coefficient

Brownian Motion

The Fokker-Planck Equation

$$\partial_t p_t = -\operatorname{div}(p_t f_t) + \frac{1}{2} g_t^2 \nabla^2 p_t$$

Where are the probabilities?

Flows

ODE $$dx_t = u_t(x_t)dt$$

Velocity field

The Continuity Equation

$$\partial_t p_t = -\operatorname{div}(p_t u_t)$$

Yes!

Diffusion

SDE $$dx_t = f_t(x_t)dt + g_t dw_t$$

Drift

Diffusion Coefficient

Brownian Motion

The Fokker-Planck Equation

$$\partial_t p_t = -\operatorname{div}(p_t f_t) + \frac{1}{2} g_t^2 \nabla^2 p_t$$

Need one more thing…

Can we build a generative model with these?

Diffusion and Score Models

SDE $$dx_t = f_t(x_t)dt + g_t dw_t$$

Drift

Diffusion Coefficient

Brownian Motion

The Fokker-Planck Equation

$$\partial_t p_t = -\operatorname{div}(p_t f_t) + \frac{1}{2} g_t^2 \nabla^2 p_t$$

Need one more thing…

Diffusion and Score Models

Forward SDE $dx_t = f_t(x_t)dt + g_t dw_t$

Data → Noise

The Fokker-Planck Equation

$$\partial_t p_t = -\operatorname{div}(p_t f_t) + \frac{1}{2} g_t^2 \nabla^2 p_t$$

Need one more thing…

Diffusion and Score Models

Forward SDE $dx_t = f_t(x_t)dt + g_t dw_t$ Data → Noise

Reverse SDE $d\bar{x}_t = (f_t(x_t) - g_t^2 \nabla \log p_t)dt + g_t d\bar{w}_t$ Noise → Data

The Fokker-Planck Equation

$$\partial_t p_t = -\operatorname{div}(p_t f_t) + \frac{1}{2} g_t^2 \nabla^2 p_t$$

Need one more thing… The Score!

Diffusion and Score Models

Forward SDE $dx_t = f_t(x_t)dt + g_t dw_t$ Data → Noise

Reverse SDE $d\bar{x}_t = (f_t(x_t) - g_t^2 \nabla \log p_t)dt + g_t d\bar{w}_t$ Noise → Data

Learn the score by regressing to conditional scores:

$$\min_\theta \mathbb{E}_{p_{data}, p_t(x|x_{data})} \left[\| s_t^\theta(x) - \nabla \log p_t(x \mid x_{data}) \|^2 \right]$$

Simulation-free

Known SDEs: Variance Exploding
Variance Preserving

Where are the probabilities?

Flows

ODE $dx_t = u_t(x_t)dt$

Velocity field

The Continuity Equation

$$\partial_t p_t = -\operatorname{div}(p_t u_t)$$

Yes!

Diffusion

SDE $dx_t = f_t(x_t)dt + g_t dw_t$

Drift

Diffusion Coefficient

Brownian Motion

The Fokker-Planck Equation

$$\partial_t p_t = -\operatorname{div}(p_t f_t) + \frac{1}{2} g_t^2 \nabla^2 p_t$$

Learn: score $\nabla \log p_t$

- Only Gaussian source
- Solution asymptotically reaches source

Where are the probabilities?

Flows

ODE $dx_t = u_t(x_t)dt$

Velocity field

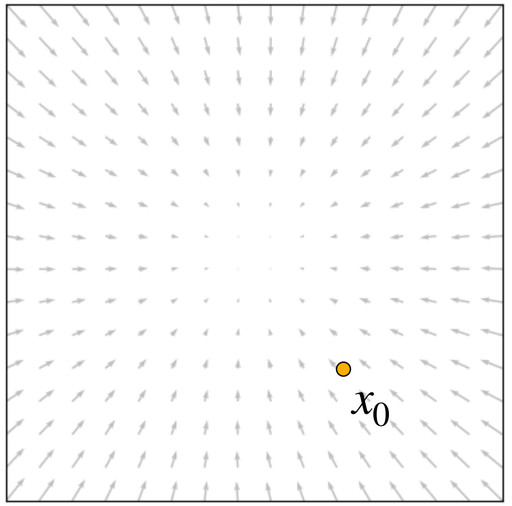
The Continuity Equation

$$\partial_t p_t = -\operatorname{div}(p_t u_t)$$

Yes!

Flow ODE

$$\dot{\psi}_t(x_0) = u_t(\psi_t(x_0))$$



$\psi_t(x)$ is smooth with smooth **inverse** defined by $-u_t(x)$

Where are the probabilities?

Flow ODE

$$\dot{\psi}_t(x_0) = u_t(\psi_t(x_0))$$

Diffusion

SDE $$dx_t = f_t(x_t)dt + g_t dw_t$$

Drift

Diffusion Coefficient

Brownian Motion

The Continuity Equation

$$\partial_t p_t = -\operatorname{div}(p_t u_t)$$

The Liouville Equation

$$\partial_t p_t = -\operatorname{div}(p_t(f_t - \frac{1}{2} g_t^2 \nabla \log p_t))$$

Learn: velocity field u_t

- Universal transformation between densities
- Defined on finite time interval

Learn: score $\nabla \log p_t$

- Only Gaussian source
- Solution asymptotically reaches source

Where are the probabilities?

Flow ODE

$$\dot{\psi}_t(x_0) = u_t(\psi_t(x_0))$$

Diffusion

SDE $$dx_t = f_t(x_t)dt + g_t dw_t$$

Drift

Diffusion Coefficient

Brownian Motion

The Continuity Equation

$$\partial_t p_t = -\operatorname{div}(p_t u_t)$$

The Liouville Equation

$$\partial_t p_t = -\operatorname{div}(p_t(f_t - \frac{1}{2} g_t^2 \nabla \log p_t))$$

Learn: velocity field u_t

- Universal transformation between densities
- Defined on finite time interval

Learn: score $\nabla \log p_t$

- Only Gaussian source
- Solution asymptotically reaches source

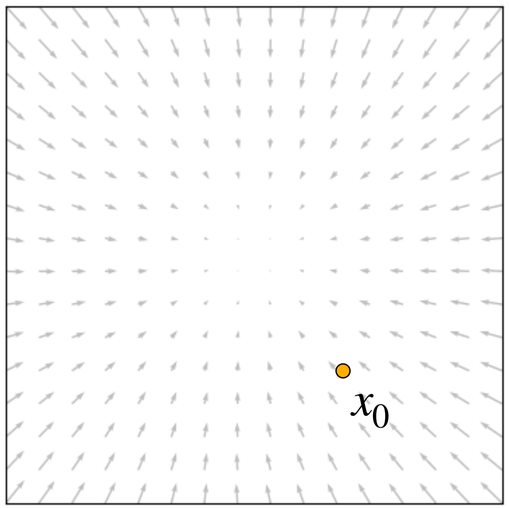
Flows as Generative Models

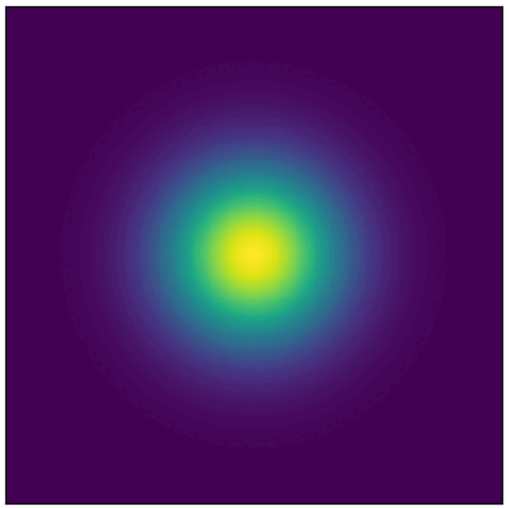
Flow ODE

$$\dot{\psi}_t(x_0) = u_t(\psi_t(x_0))$$

Continuity Equation

$$\partial_t p_t = -\operatorname{div}(p_t u_t)$$





Learn: velocity field u_t

Goal: find velocity field u_t s.t. $p_1 \approx q$

Chen et al., *Neural Ordinary Differential Equations.* (NeurIPS 2018)

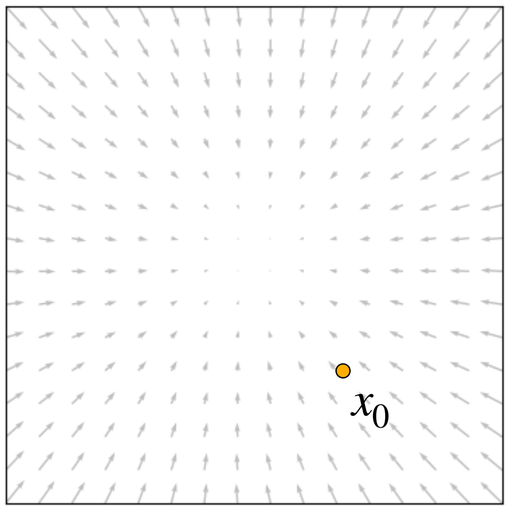
Training with Simulation

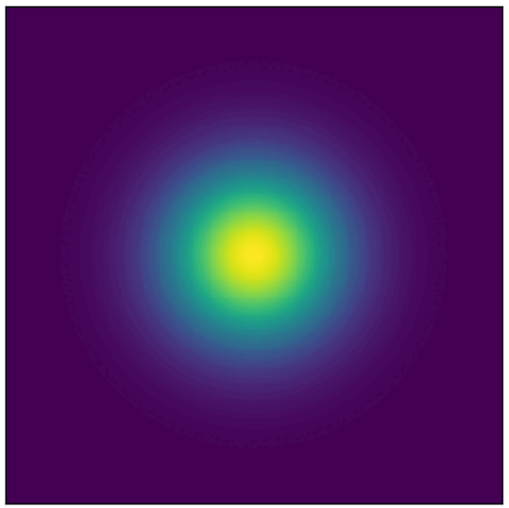
Flow ODE

$$\dot{\psi}_t(x_0) = u_t(\psi_t(x_0))$$

Continuity Equation

$$\partial_t p_t = -\operatorname{div}(p_t u_t)$$





Learn: velocity field u_t

Log-likelihood computation

$$\log p_1(x_1) = \log p(x_0) + \int_1^0 \operatorname{div}(u_t(x_t))dt$$

$$x_t = x_1 + \int_1^t u_s(x_s)ds$$

Maximum Likelihood Objective

$$D_{\mathrm{KL}}(q \,\|\, p_1) = -\,\mathbb{E}_{x\sim q} \log p_1(x) + c$$

Chen et al., *Neural Ordinary Differential Equations.* (NeurIPS 2018)

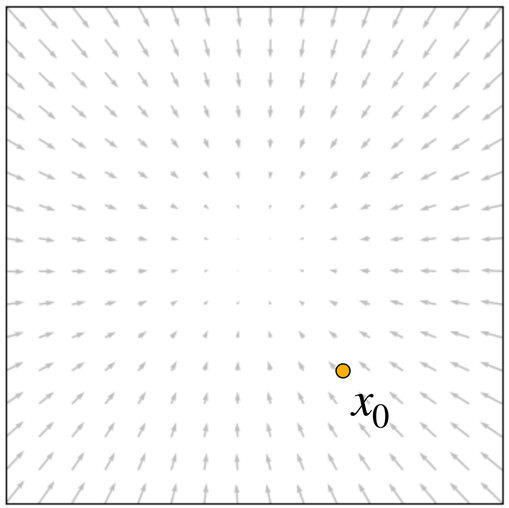
Training with Simulation

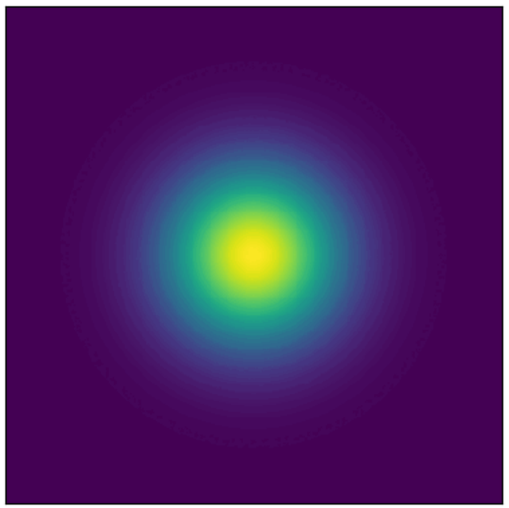
Flow ODE

$$\dot{\psi}_t(x_0) = u_t(\psi_t(x_0))$$

Continuity Equation

$$\partial_t p_t = -\operatorname{div}(p_t u_t)$$





Log-likelihood computation

$$\log p_1(x_1) = \log p(x_0) + \int_1^0 \operatorname{div}(u_t(x_t))dt$$

$$x_t = x_1 + \int_1^t u_s(x_s)ds$$

Maximum Likelihood Objective

$$D_{\mathrm{KL}}(q \,\|\, p_1) = -\,\mathbb{E}_{x\sim q} \log p_1(x) + c$$

Vs. Normalizing Flows:

- Avoids invertible architectures!
- Requires computing divergence

Chen et al., *Neural Ordinary Differential Equations.* (NeurIPS 2018)

Training with Simulation

Flows

ODE $$dx_t = u_t(x_t)dt$$

Velocity field

Log-likelihood computation

$$\log p_1(x_1) = \log p(x_0) + \int_1^0 \operatorname{div}(u_t(x_t))dt$$

$$x_t = x_1 + \int_1^t u_s(x_s)ds$$

The Continuity Equation

$$\partial_t p_t = -\operatorname{div}(p_t u_t)$$

Requires:

- Simulating x_t
- Backprop through simulation
- (Unbiased) estimator of $\operatorname{div}(u_t)$
- Can compute $\log p(x)$

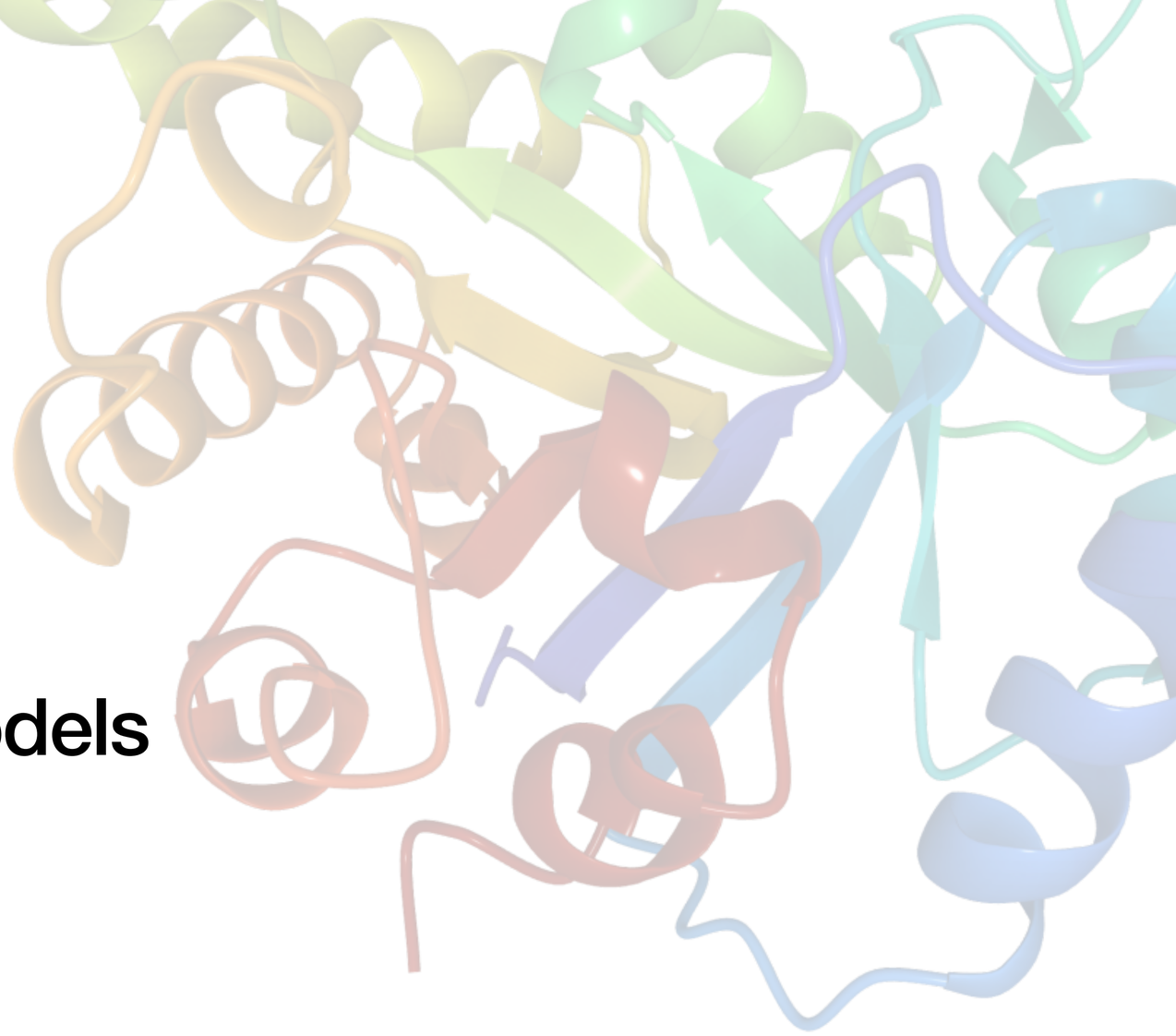
Maximum Likelihood Objective

$$D_{\mathrm{KL}}(q \,\|\, p_1) = -\mathbb{E}_{x\sim q} \log p_1(x) + c$$

Chen et al., *Neural Ordinary Differential Equations.* (NeurIPS 2018)

Part II:

Simulation Free Generative Models



Flow Matching

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

ICLR 2023

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

ICLR 2023

FLOW STRAIGHT AND FAST: LEARNING TO GENERATE AND TRANSFER DATA WITH RECTIFIED FLOW

Xingchao Liu*, **Chengyue Gong***, **Qiang Liu**
Department of Computer Science
University of Texas at Austin
{xcliu, cygong, lqiang}@cs.utexas.edu

ICLR 2023

Flow Matching

NON-DENOISING FORWARD-TIME DIFFUSIONS

Stefano Peluchetti
Date: 18 Nov 2021

Rejected ICLR 2022

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

ICLR 2023

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

ICLR 2023

FLOW STRAIGHT AND FAST: LEARNING TO GENERATE AND TRANSFER DATA WITH RECTIFIED FLOW

Xingchao Liu*, **Chengyue Gong***, **Qiang Liu**
Department of Computer Science
University of Texas at Austin
{xcliu, cygong, lqiang}@cs.utexas.edu

ICLR 2023

Flow Matching

$$L_{\text{FM}}(\theta) = \min \mathbb{E}_{t,\, p_t(x)} \| u_t^{\theta}(x) - u_t(x) \|^2$$

Core Principle:

u_t generates p_t
iff they satisfy the continuity equation

Construct:

- Target probability path p_t s.t. $p_0 = p$, $p_1 \approx q$
- Generating velocity field u_t

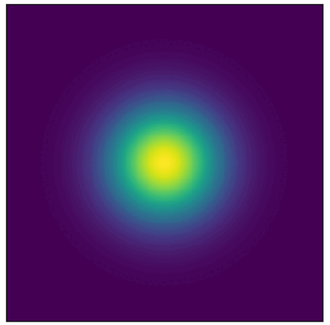
Find a tractable optimization objective

Conditional Probability Paths

Note! This is the simplest case, we will consider other z in the future

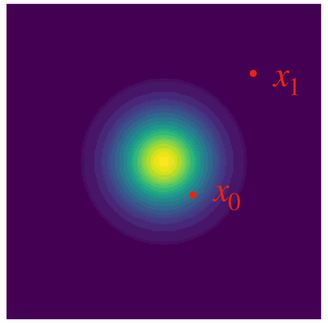
Law of total probability $z := (x_0, x_1)$

$$p_t(x) = \int p_t(x \mid z) q(z) dz$$



Marginal path

$p_t(x \mid z)$

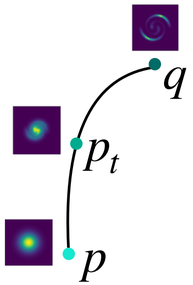


Conditional path

Boundary conditions:

$$p_0 = p$$

$$p_1 = q$$



$$p_0(\cdot \mid z) = \delta_{x_0}$$

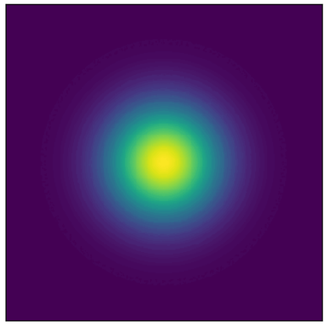
$$p_1(\cdot \mid z) = \delta_{x_1}$$

Lipman et al., *Flow Matching for Generative Modeling.* (ICLR 2023)

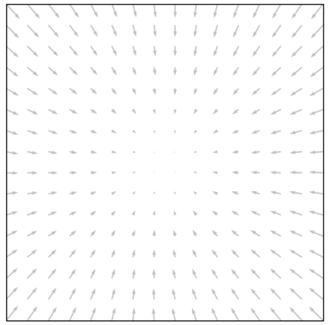
The marginalization “trick”

Marginal path

$$p_t(x) = \int p_t(x \mid z) q(z) dz$$

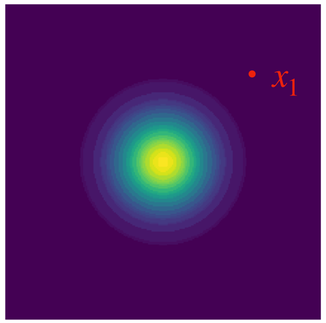


$$u_t(x) = \int u_t(x \mid z) \frac{p_t(x \mid z) q(z)}{p_t(x)} dz$$

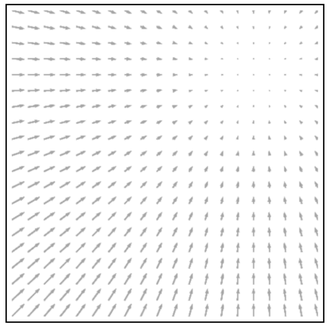


Conditional path

$p_t(x \mid z)$



$u_t(x \mid z)$



Lipman et al., *Flow Matching for Generative Modeling.* (ICLR 2023)

The marginalization "trick"

Continuity Equation PDE

$$\partial_t p_t = -\operatorname{div}(p_t u_t)$$

Claim: $u_t(x) = \int u_t(x \mid z) \frac{p_t(x \mid z) q(z)}{p_t(x)} dz$ generates $p_t(x) = \int p_t(x \mid z) q(z) dz$

Proof:

$$\begin{aligned}
\operatorname{div}(p_t(x) u_t(x)) &= \\
&= \operatorname{div}(\int u_t(x \mid z) p_t(x \mid z) q(z) dz) \\
&= \int \operatorname{div}(u_t(x \mid z) p_t(x \mid z)) q(z) dz \\
&= \int -\frac{d}{dt} p_t(x \mid z) q(z) dz \\
&= -\frac{d}{dt} \int p_t(x \mid z) q(z) dz \\
&= -\frac{d}{dt} p_t(x)
\end{aligned}$$

Lipman et al., *Flow Matching for Generative Modeling.* (ICLR 2023)

Flow Matching

$$L_{\text{FM}}(\theta) = \min \mathbb{E}_{t,\, p_t(x)} \| u_t^\theta(x) - u_t(x) \|^2$$

$$u_t(x) = \int u_t(x \mid z) \frac{p_t(x \mid z) q(z)}{p_t(x)} dz$$

Construct:

- Target probability path p_t s.t. $p_0 = p$, $p_1 \approx q$
- Generating velocity field u_t

Find a tractable optimization objective

Lipman et al., *Flow Matching for Generative Modeling.* (ICLR 2023)

Flow Matching

$$L_{\text{FM}}(\theta) = \min \mathbb{E}_{t,\, p_t(x)} \| u_t^\theta(x) - u_t(x) \|^2$$

$$u_t(x) = \int u_t(x \mid z) \frac{p_t(x \mid z) q(z)}{p_t(x)} dz$$

Useful example:

$$z = (x_0, x_1)$$

Construct:

- Target probability path p_t s.t. $p_0 = p$, $p_1 \approx q$
- Generating velocity field u_t

Find a tractable optimization objective

Lipman et al., *Flow Matching for Generative Modeling.* (ICLR 2023)

Conditional Flow Matching Loss

$$L_{\mathrm{FM}}(\theta) = \min \mathbb{E}_{t,\, p_t(x)} \| u_t^\theta(x) - u_t(x) \|^2$$

$$L_{\mathrm{CFM}}(\theta) = \min \mathbb{E}_{t,\, q(z),\, p_t(x|z)} \| u_t^\theta(x) - u_t(x \mid z) \|^2$$

The gradients of losses coincide:

$$\nabla_\theta L_{\mathrm{FM}} = \nabla_\theta L_{\mathrm{CFM}}$$

Lipman et al., *Flow Matching for Generative Modeling.* (ICLR 2023)

Conditional Flow Matching Loss

$$L_{\text{FM}}(q\|p_1) = \min \mathbb{E}_{t,\, p_t(x)} \|v_t(x) - u_t(x)\|^2 \qquad L_{\text{CFM}}(q\|p_1) = \min \mathbb{E}_{t,\, q(x_1),\, p_t(x|x_1)} \|v_t(x) - u_t(x \,|\, z)\|^2$$

$$\|v_t(x) - u_t(x)\|^2 = \|v_t(x)\|^2 - 2\langle v_t(x), u_t(x)\rangle + \|u_t(x)\|^2$$

Do not depend on θ

$$\|v_t(x) - u_t(x \,|\, z)\|^2 = \|v_t(x)\|^2 - 2\langle v_t(x), u_t(x \,|\, z)\rangle + \|u_t(x \,|\, z)\|^2$$

Law of total expectation $\mathbb{E}_{t,\, p_t(x)} \|v_t(x)\|^2 = \mathbb{E}_{t,\, q(x_1), p_t(x|z)} \|v_t(x)\|^2$

Lipman et al., *Flow Matching for Generative Modeling.* (ICLR 2023)

Flow Matching

$$L_{\text{FM}}(\theta) = \min \mathbb{E}_{t,\, p_t(x)} \| u_t^\theta(x) - u_t(x) \|^2$$

$$L_{\text{CFM}}(\theta) = \min \mathbb{E}_{t,\, q(z),\, p_t(x|z)} \| u_t^\theta(x) - u_t(x \mid z) \|^2$$

Construct:

- Target probability path p_t s.t. $p_0 = p$, $p_1 \approx q$
- Generating velocity field u_t

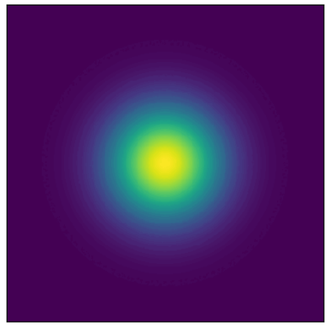
Find a tractable optimization objective

Lipman et al., *Flow Matching for Generative Modeling.* (ICLR 2023)

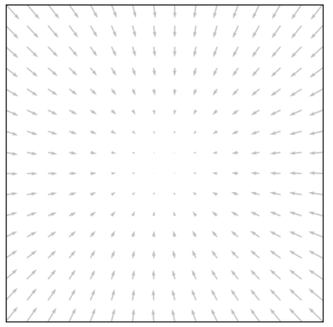
Conditional Flows

Marginal path

$$p_t(x) = \int p_t(x \mid z) q(z) dz$$



$$u_t(x) = \int u_t(x \mid z) \frac{p_t(x \mid z) q(z)}{p_t(x)} dz$$

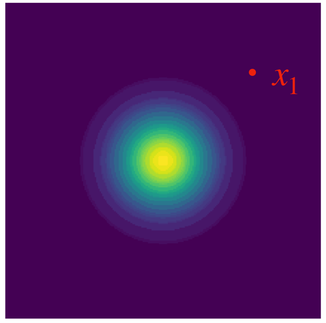


Construct a conditional flow s.t.

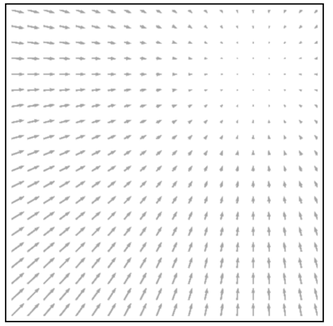
$$\psi_0(x \mid z) = x \quad , \quad \psi_1(x \mid z) = x_1$$

Conditional path

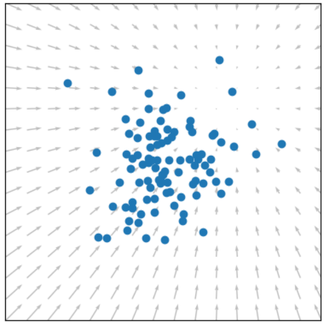
$p_t(x \mid x_1)$



$u_t(x \mid x_1)$



$\psi_t(x_0 \mid x_1)$



Lipman et al., *Flow Matching for Generative Modeling.* (ICLR 2023)

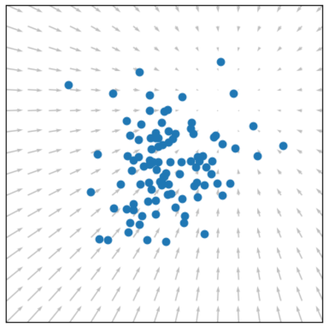
Affine Conditional Flows

Construct a conditional flow
s.t.
$\psi_0(x \mid x_1) = x \;\;,\;\; \psi_1(x \mid x_1) = x_1$

Affine conditional flow: $\psi_t(x_0 \mid x_1) = x_t = \alpha_t x_1 + \sigma_t x_0$

Conditional velocity field: $\frac{d}{dt}\psi_t(x_0 \mid x_1) = u_t(\psi_t(x_0 \mid x_1) \mid x_1)$

$\psi_t(x_0 \mid x_1)$



$u_t(x_t \mid x_1) = \dot{\alpha}_t x_1 + \dot{\sigma}_t x_0$

$u_t(x \mid x_1) = \frac{\dot{\sigma}_t}{\sigma_t} x + (\dot{\alpha}_t - \alpha_t \frac{\dot{\sigma}_t}{\sigma_t}) x_1$

Lipman et al., *Flow Matching for Generative Modeling.* (ICLR 2023)

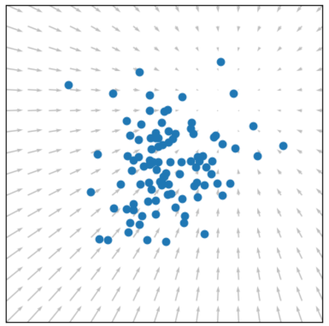
Affine Conditional Flows

Construct a conditional flow
s.t.
$\psi_0(x \mid x_1) = x \;\;,\;\; \psi_1(x \mid x_1) = x_1$

Affine conditional flow: $\psi_t(x_0 \mid x_1) = x_t = \alpha_t x_1 + \sigma_t x_0$

Conditional velocity field: $\frac{d}{dt} x_t = u_t(x_t \mid x_1)$

$\psi_t(x_0 \mid x_1)$



$u_t(x_t \mid x_1) = \dot{\alpha}_t x_1 + \dot{\sigma}_t x_0$

$u_t(x_t \mid x_1) = \frac{\dot{\sigma}_t}{\sigma_t} x + (\dot{\alpha}_t - \alpha_t \frac{\dot{\sigma}_t}{\sigma_t}) x_1$

Lipman et al., *Flow Matching for Generative Modeling.* (ICLR 2023)

The default setting

Construct a conditional flow s.t.

$$\psi_0(x \mid x_1) = x \quad , \quad \psi_1(x \mid x_1) = x_1$$

flow coefficients:

$$\alpha_t = t \quad , \quad \sigma_t = 1 - t$$

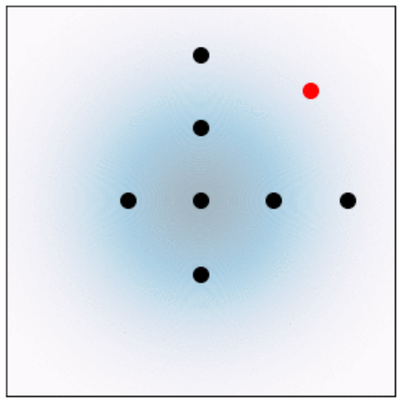
flow:

$$\psi_t(x_0 \mid x_1) = x_t = tx_1 + (1 - t)x_0$$

$$u_t(x_t \mid x_1) = x_1 - x_0$$

$$u_t(x \mid x_1) = \frac{x_1 - x}{1 - t}$$

$\psi_t(x_0 \mid x_1)$



Equivalent to a diffusion loss when

$$\sigma_t = t!$$

Lipman et al., *Flow Matching for Generative Modeling.* (ICLR 2023)

Gaussian Affine Conditional Flows

Construct a conditional flow
s.t.
$\psi_0(x \mid x_1) = x \;\;,\;\; \psi_1(x \mid x_1) = x_1$

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

Affine conditional flow: $\psi_t(x_0 \mid x_1) = x_t = \alpha_t x_1 + \sigma_t x_0$

$x_t \sim \mathcal{N}(\alpha_t x_1, \sigma_t^2 I)$

Popular Diffusion Paths:

Variance Exploding: $x_t \sim \mathcal{N}(x_1, \sigma_t^2 I)$

Variance Preserving: $x_t \sim \mathcal{N}(\alpha_t x_1, (1 - \alpha_t)_t I)$

Lipman et al., *Flow Matching for Generative Modeling.* (ICLR 2023)

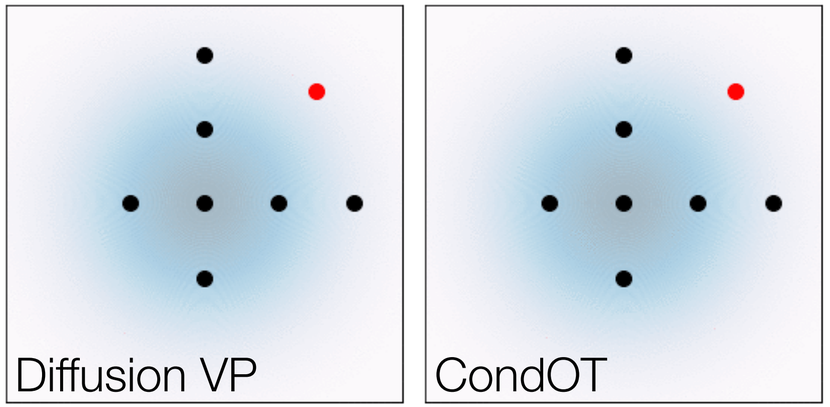
Gaussian Affine Conditional Flows

Construct a conditional flow
s.t.
$\psi_0(x \mid x_1) = x \;\;,\;\; \psi_1(x \mid x_1) = x_1$

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

Affine conditional flow: $\psi_t(x_0 \mid x_1) = x_t = \alpha_t x_1 + \sigma_t x_0$

$x_t \sim \mathcal{N}(\alpha_t x_1, \sigma_t^2 I)$



Popular Diffusion Paths:

Variance Exploding: $x_t \sim \mathcal{N}(x_1, \sigma_t^2 I)$

Variance Preserving: $x_t \sim \mathcal{N}(\alpha_t x_1, (1 - \alpha_t)_t I)$

Lipman et al., *Flow Matching for Generative Modeling.* (ICLR 2023)

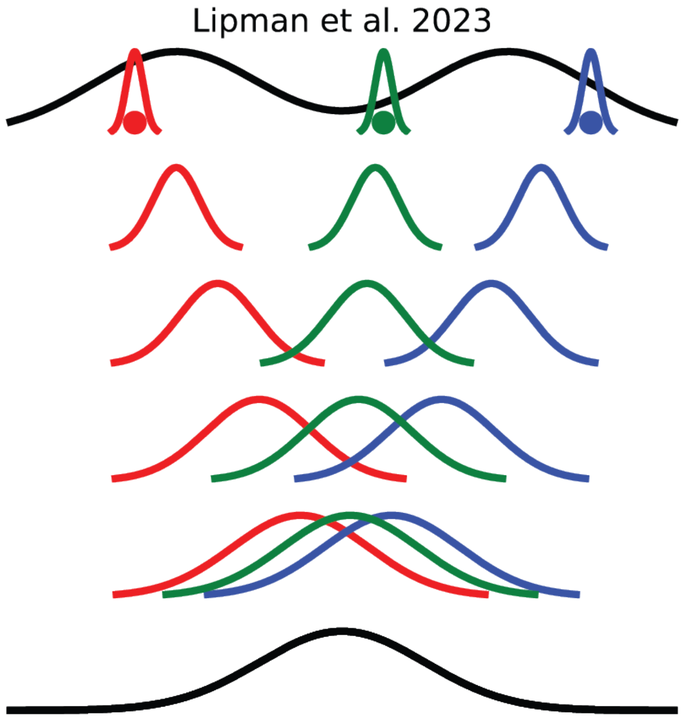
A Geometric / Algorithmic point of view

Flow Matching

Diffusion

OT

$q(x_1)$



$q(x_0)$

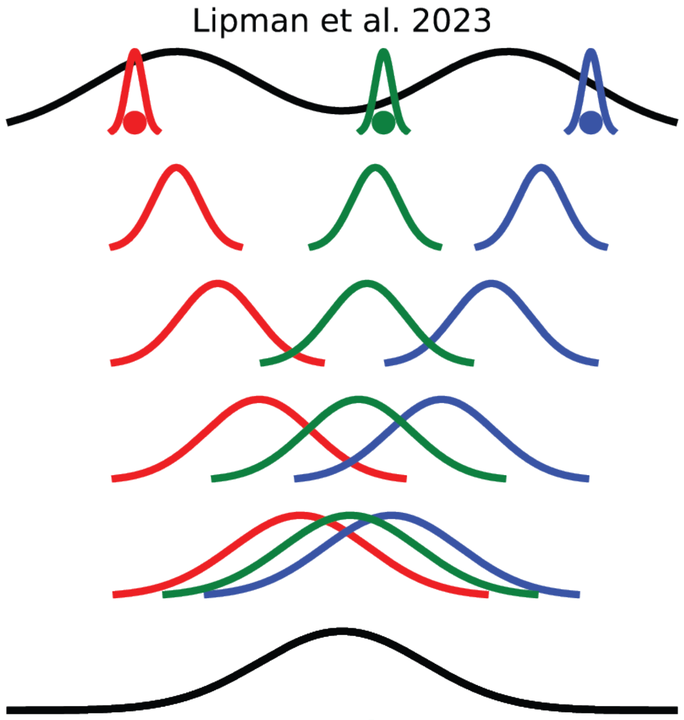
Flow Matching

Diffusion

OT

$p_t(x \mid z^1)$ $p_t(x \mid z^3)$

Lipman et al. 2023



$q(z) := q(x_1)\, p_t(x \mid z^2)$

Flow Matching

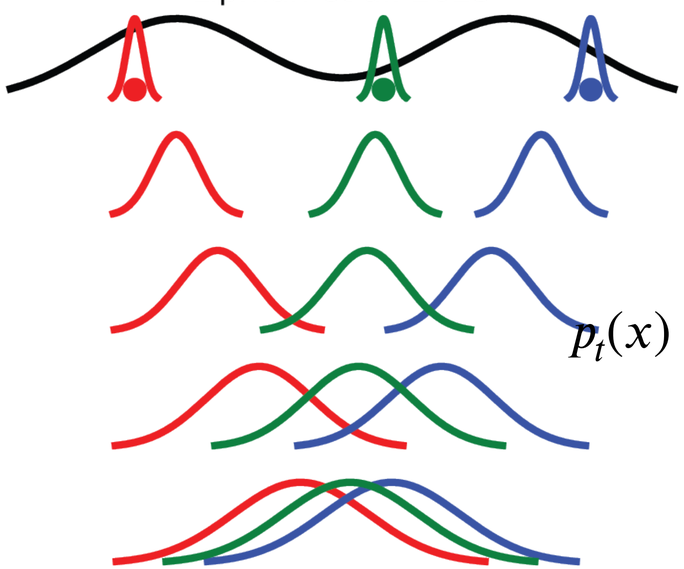
Diffusion

OT

$p_t(x \mid z^1)$ $p_t(x \mid z^3)$

Lipman et al. 2023

$q(x_1)$



$q(x_0)$

$$p_t(x) = \sum_z p_t(x \mid z)$$

$q(z) := q(x_1)\, p_t(x \mid z^2)$

Flow Matching

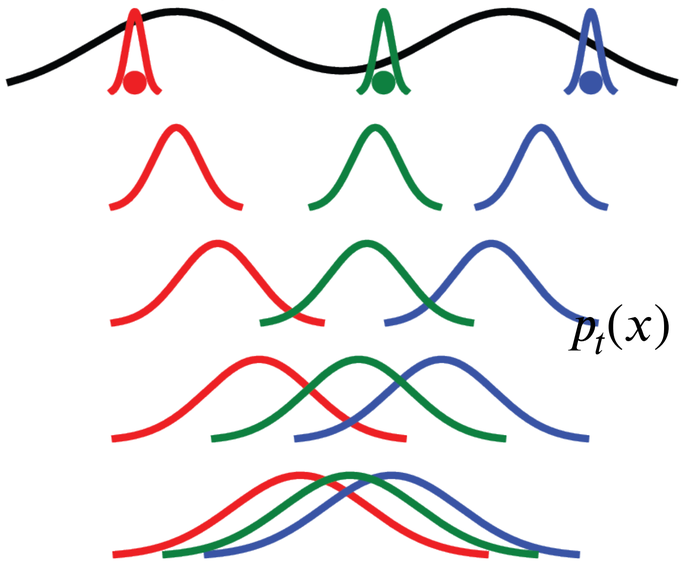
Diffusion

OT

$p_t(x \mid z^1)$ $p_t(x \mid z^3)$

Lipman et al. 2023

$q(x_1)$



$q(x_0)$

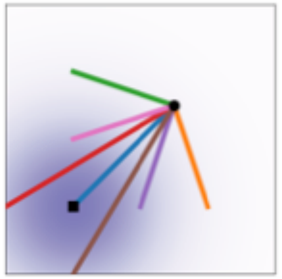
$q(z) := q(x_1)\, p_t(x \mid z^2)$

$$p_t(x) = \sum_z p_t(x \mid z)$$

Closed form Gaussian Conditional Flow! $p_t(x \mid z) = \mathrm{N}(x \mid \mu_t,\ \sigma_t)$ then $u_t(x \mid z) = \frac{\sigma_t'(z)}{\sigma_t(z)}(x - \mu_t(z)) + \mu_t'(z)$

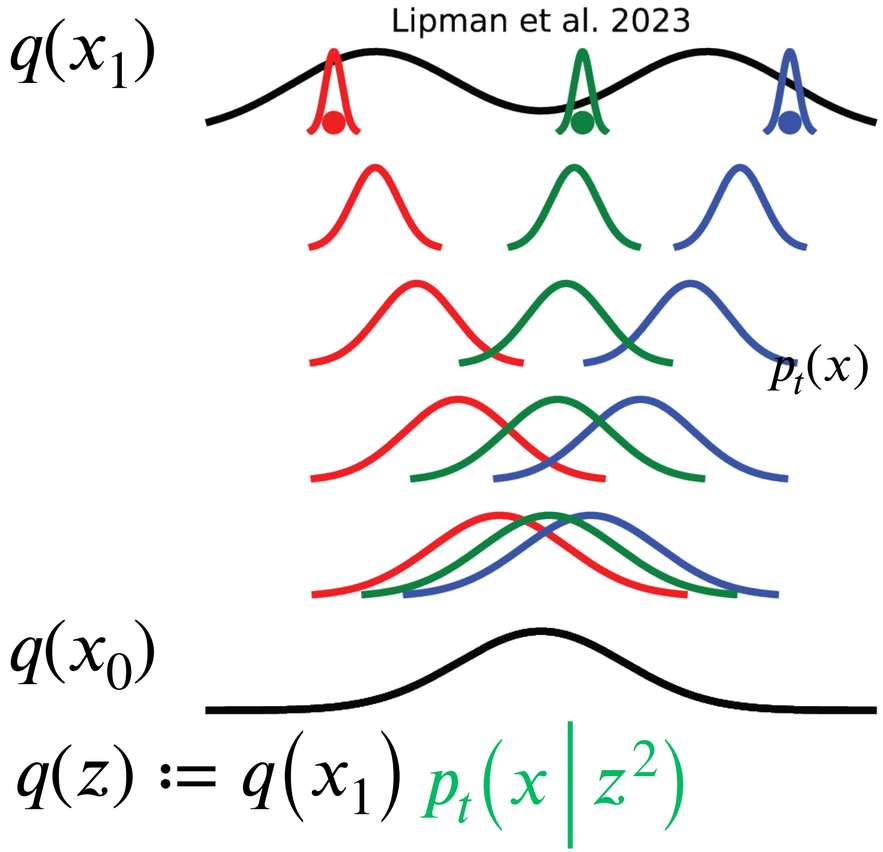
Flow Matching

Diffusion



OT

$p_t(x \mid z^1)$ $p_t(x \mid z^3)$



$$p_t(x) = \sum_z p_t(x \mid z)$$

$q(z) := q(x_1)\, p_t(x \mid z^2)$

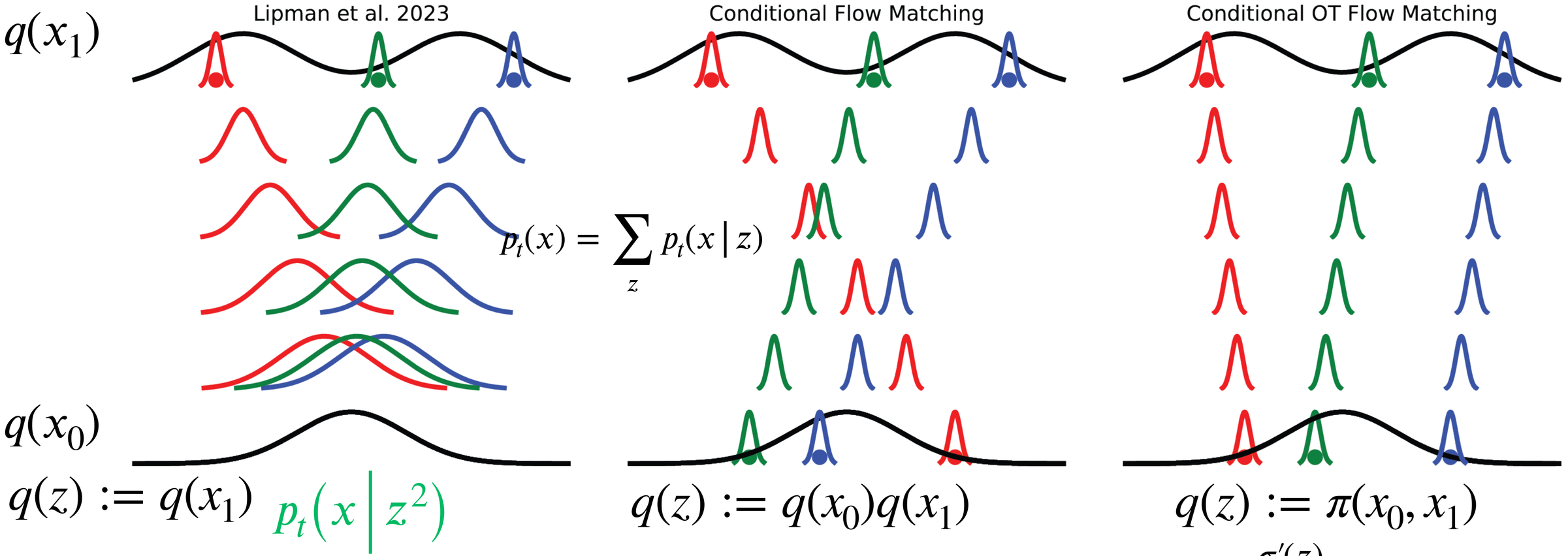
Marginalization for $u_t(x)$ *that generates* $p_t(x)$ *given* $q(z)$

$$u_t(x) = \int \frac{u_t(x \mid z) p_t(x \mid z) q(z)}{p_t(x)} dz$$

Closed form Gaussian Conditional Flow! $p_t(x \mid z) = \mathrm{N}(x \mid \mu_t,\ \sigma_t)$ then $u_t(x \mid z) = \frac{\sigma_t'(z)}{\sigma_t(z)}(x - \mu_t(z)) + \mu_t'(z)$

Conditional Flow Matching

$p_t(x \mid z^1)$ $p_t(x \mid z^3)$



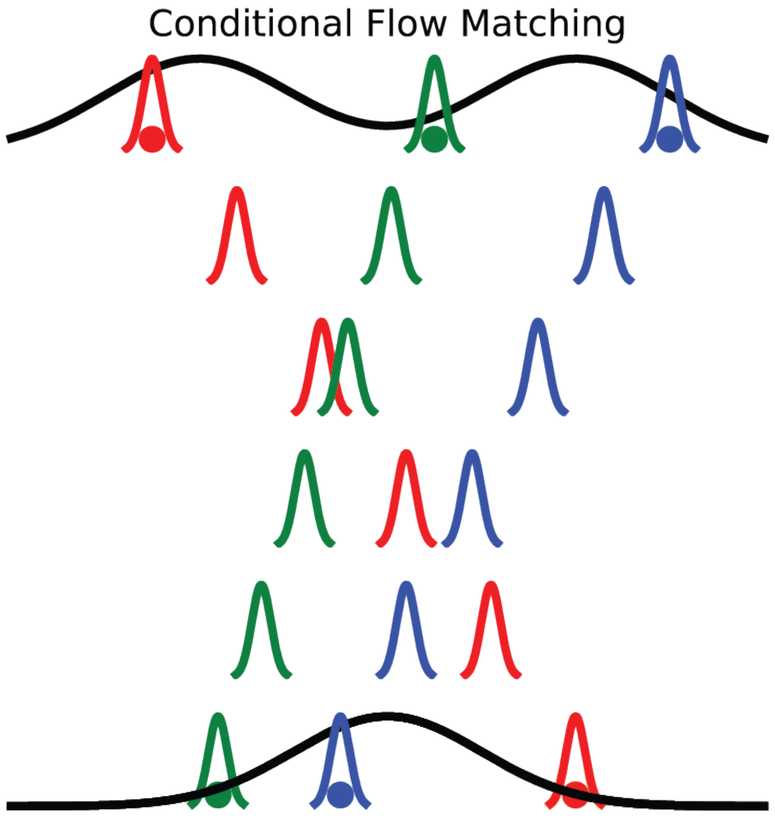
$q(z) := q(x_1)$ $p_t(x \mid z^2)$

$q(z) := q(x_0)q(x_1)$

$q(z) := \pi(x_0, x_1)$

Closed form Gaussian Conditional Flow! $p_t(x \mid z) = \mathrm{N}(x \mid \mu_t, \sigma_t)$ then $u_t(x \mid z) = \frac{\sigma_t'(z)}{\sigma_t(z)}(x - \mu_t(z)) + \mu_t'(z)$

Objective: $L_{CFM}(\theta) = \mathbb{E}_{t,q(z),p_t(x|z)} \|v_\theta(t,x) - u_t(x|z)\|_2^2$



Algorithm 1 Simplified Conditional Flow Matching

Input: Sample-able distributions $\boldsymbol{X}_0, \boldsymbol{X}_1$, bandwidth σ, batchsize b, initial network v_θ.

while Training **do**

/ Sample batches of size b i.i.d. from the datasets */*

$\boldsymbol{x}_0 \sim \boldsymbol{X}_0; \quad \boldsymbol{x}_1 \sim \boldsymbol{X}_1$

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

$\mu_t \leftarrow t\boldsymbol{x}_1 + (1-t)\boldsymbol{x}_0$

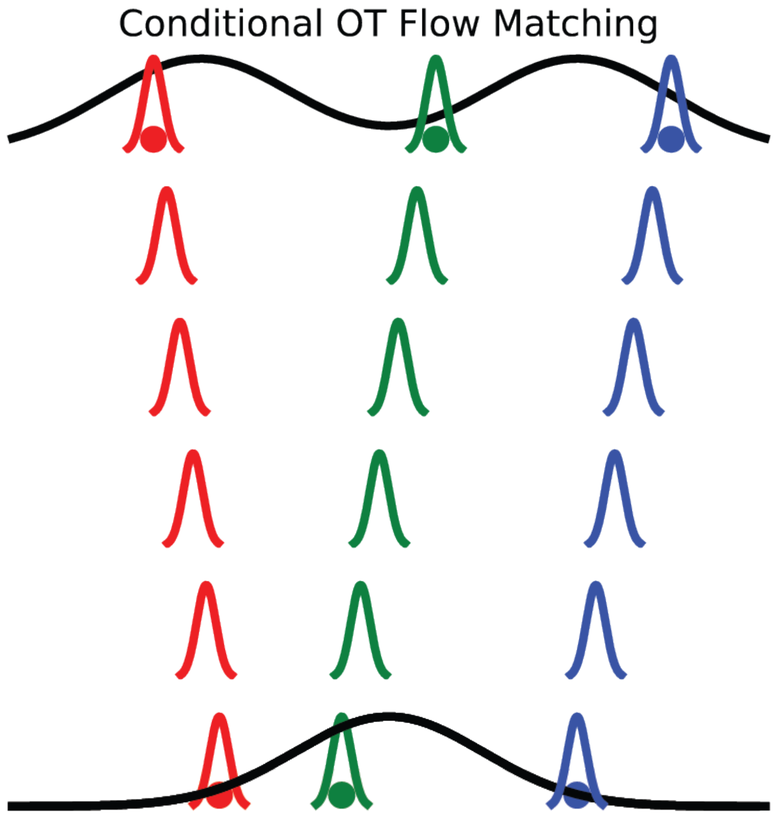
$x \sim \mathcal{N}(\mu_t, \sigma^2 I)$

$L_{CFM}(\theta) \leftarrow \|v_\theta(t,x) - (\boldsymbol{x}_1 - \boldsymbol{x}_0)\|^2$

$\theta \leftarrow \text{Update}(\theta, \nabla_\theta L_{CFM}(\theta))$

return v_θ

+ Static (mini-batch) OT Resampling Step



Algorithm 2 Minibatch OT Conditional Flow Matching

Input: Sample-able distributions $\boldsymbol{X}_0, \boldsymbol{X}_1$, bandwidth σ, batch size b, initial network v_θ.

while Training **do**

/ Sample batches of size b i.i.d. from the datasets */*

$\boldsymbol{x}_0 \sim \boldsymbol{X}_0; \quad \boldsymbol{x}_1 \sim \boldsymbol{X}_1$

$\pi \leftarrow \mathrm{OT}(\boldsymbol{x}_1, \boldsymbol{x}_0)$

$(\boldsymbol{x}_0, \boldsymbol{x}_1) \sim \pi$

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

$\mu_t \leftarrow \boldsymbol{t}\boldsymbol{x}_1 + (1 - \boldsymbol{t})\boldsymbol{x}_0$

$\boldsymbol{x} \sim \mathcal{N}(\mu_t, \sigma^2 I)$

$L_{CFM}(\theta) \leftarrow \|v_\theta(\boldsymbol{t}, \boldsymbol{x}) - (\boldsymbol{x}_1 - \boldsymbol{x}_0)\|^2$

$\theta \leftarrow \mathrm{Update}(\theta, \nabla_\theta L_{CFM}(\theta))$

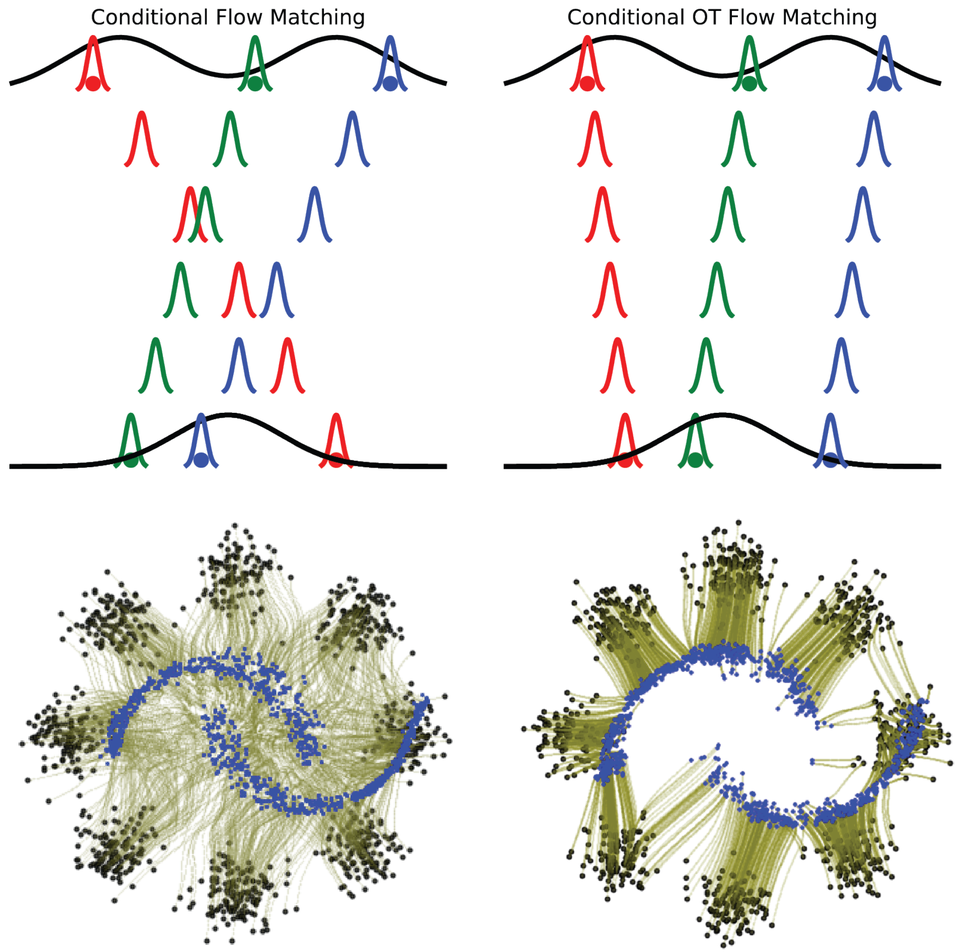
return v_θ

Tong et al. TMLR 2024

Why use optimal transport in flow matching?

More general framework:

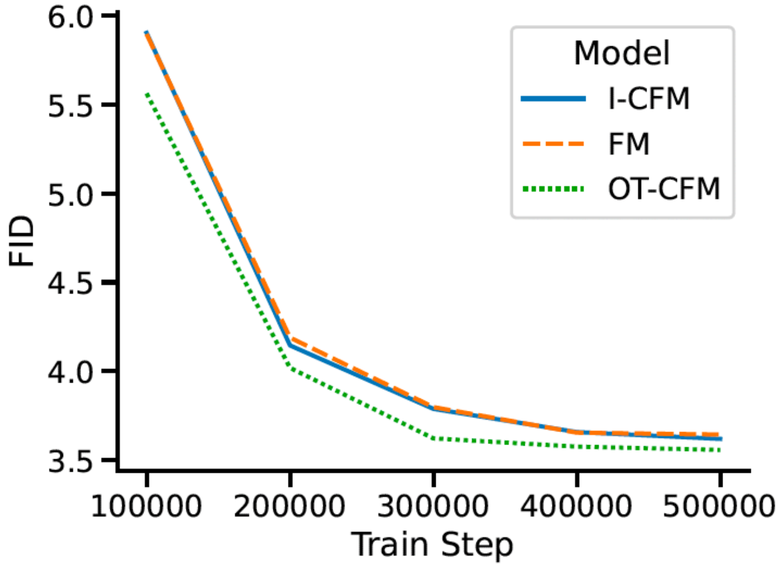
- Reduced variance in the objective via optimal transport leads to **faster training**
- Straighter inference paths via optimal transport leads to **faster inference**
- Can be applied to new problems where we care about the paths

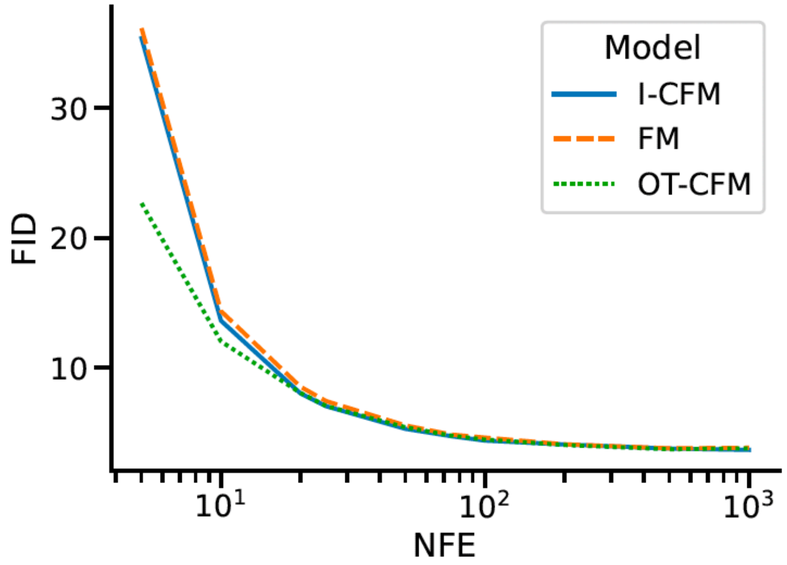


Why use optimal transport in flow matching?

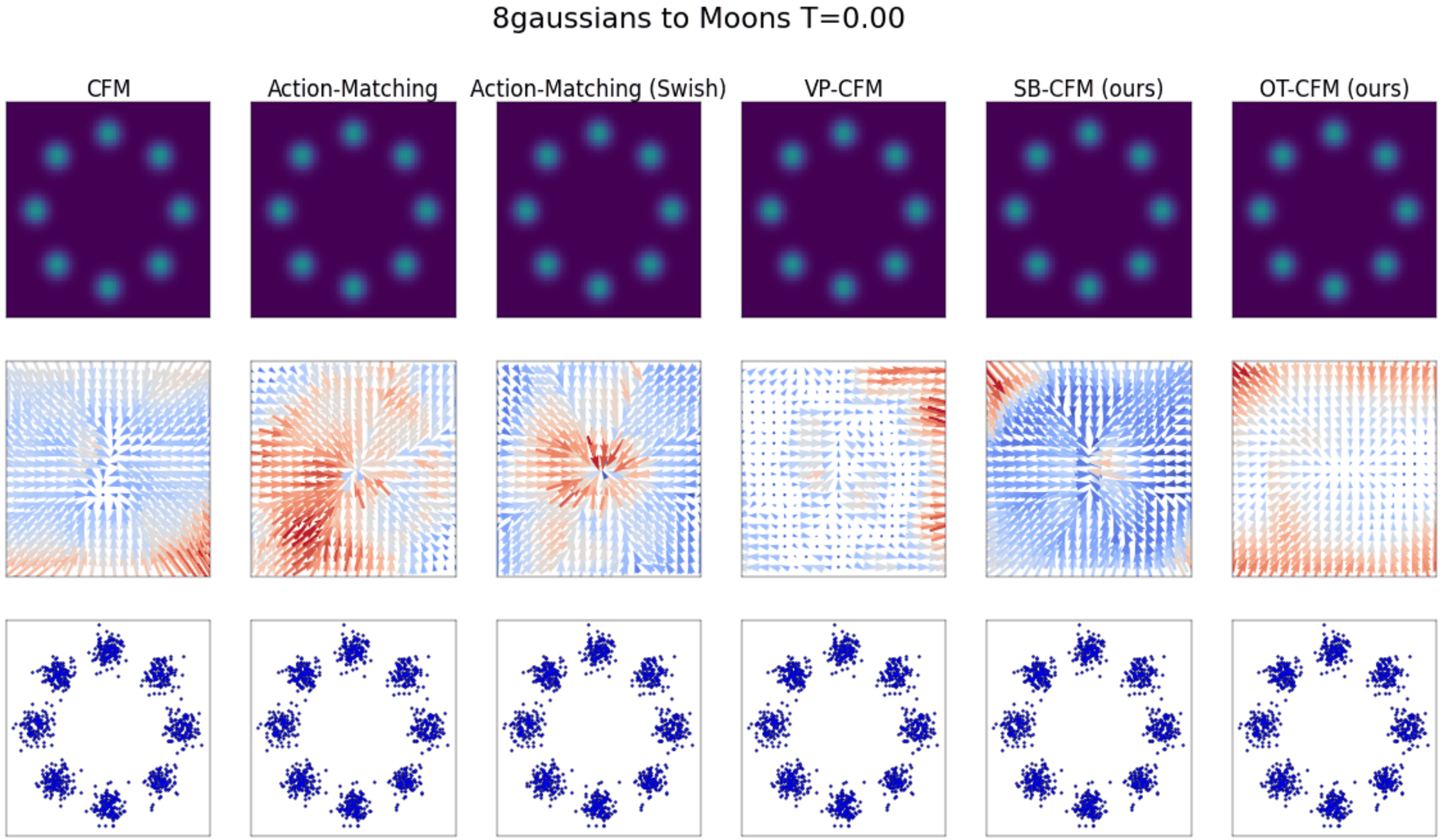
More general framework:

- Reduced variance in the objective via optimal transport leads to **faster training**
- Straighter inference paths via optimal transport leads to **faster inference**
- Can be applied to new problems where we care about the paths





Comparing choices of $u_t(x \mid z)$, $p_t(x \mid z)$, and $q_t(z)$



Flow Matching

$$L_{\text{FM}}(\theta) = \min \mathbb{E}_{t,\, p_t(x)} \| u_t^\theta(x) - u_t(x) \|^2$$

Add slide on types of prediction

$$L_{\text{CFM}}(\theta) = \min \mathbb{E}_{t,\, q(z),\, p_t(x|z)} \| u_t^\theta(x) - u_t(x \mid z) \|^2$$

Construct:

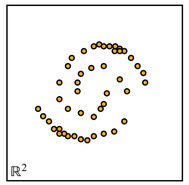
- Target probability path p_t s.t. $p_0 = p$, $p_1 \approx q$
- Generating velocity field u_t

Find a tractable optimization objective

Lipman et al., *Flow Matching for Generative Modeling.* (ICLR 2023)

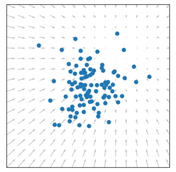
Recipe: Flow Matching

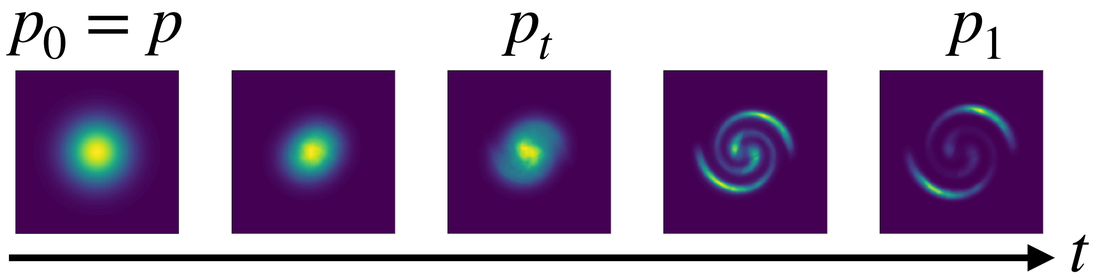
- Given: samples $x_1 \sim q$



- Construct: p_t s.t. $p_0 = p$, $p_1 \approx q$

  via conditional flows $\psi_t(x \mid z)$

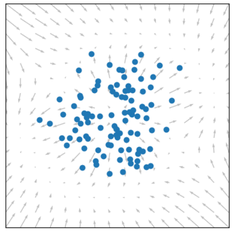




- Learn: velocity field u_t with CFM loss

  s.t. $\psi_t(x_0) \sim p_t$ where $x_0 \sim p$

$$L_{\text{CFM}}(\theta) = \min \mathbb{E}_{t,\, q(z),\, p_t(x|z)} \| u_t^\theta(x) - u_t(x \mid z) \|^2$$



$$\psi_t : [0,1] \times \mathbb{R}^2 \to \mathbb{R}^2$$

Recipe: Flow Matching

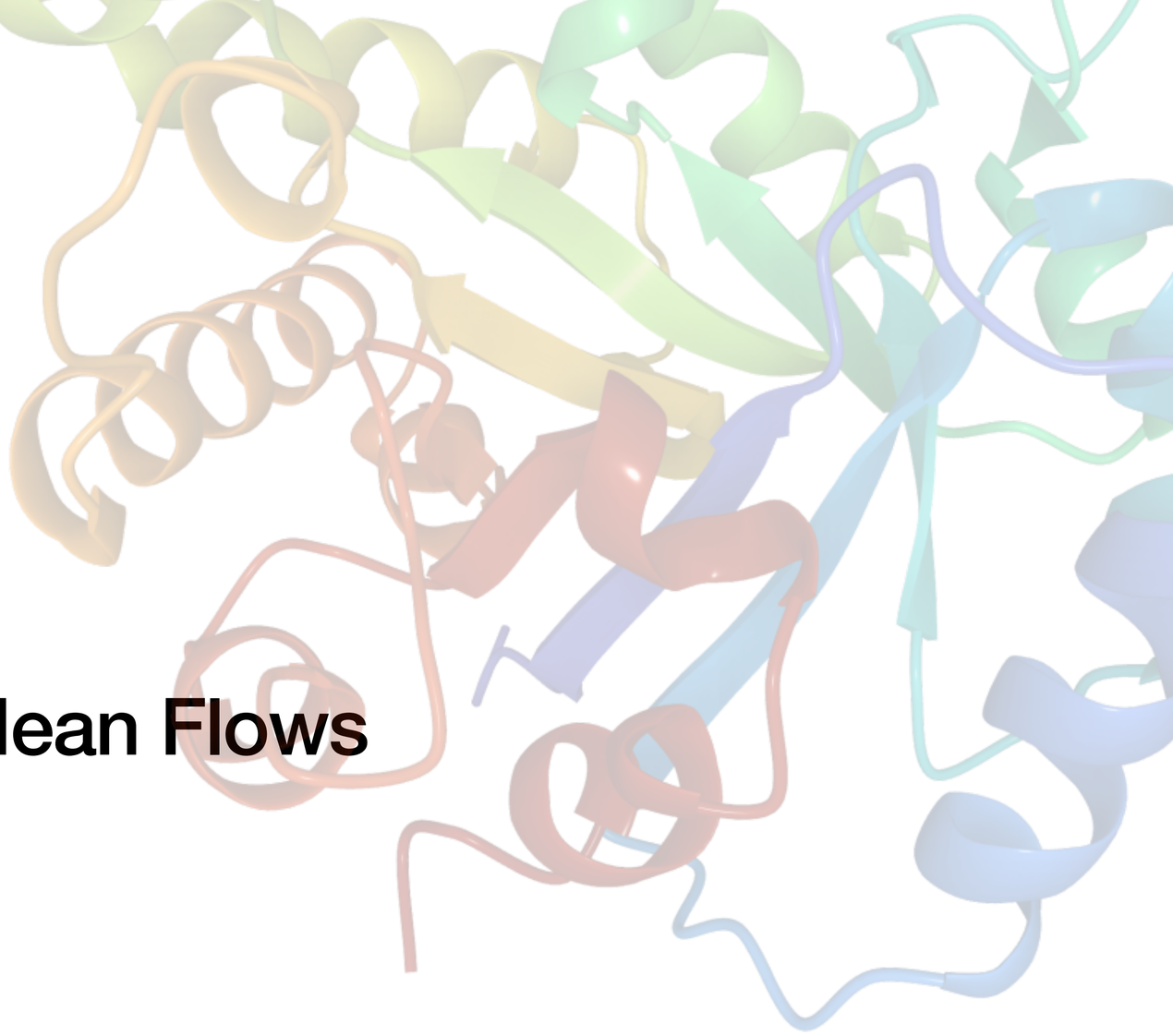
- Given: samples $x_1 \sim q$
- Design Choices:
  - Choose a source distribution p
  - Construct p_t s.t. $p_0 = p$, $p_1 \approx q$
    - Choose conditional flow $\psi_t(x \mid z)$
- Learn: velocity field u_t s.t. $\psi_t(x_0) \sim p_t$ where $x_0 \sim p$
  - We learn velocities by square regressing using L_2-norm

Exercise

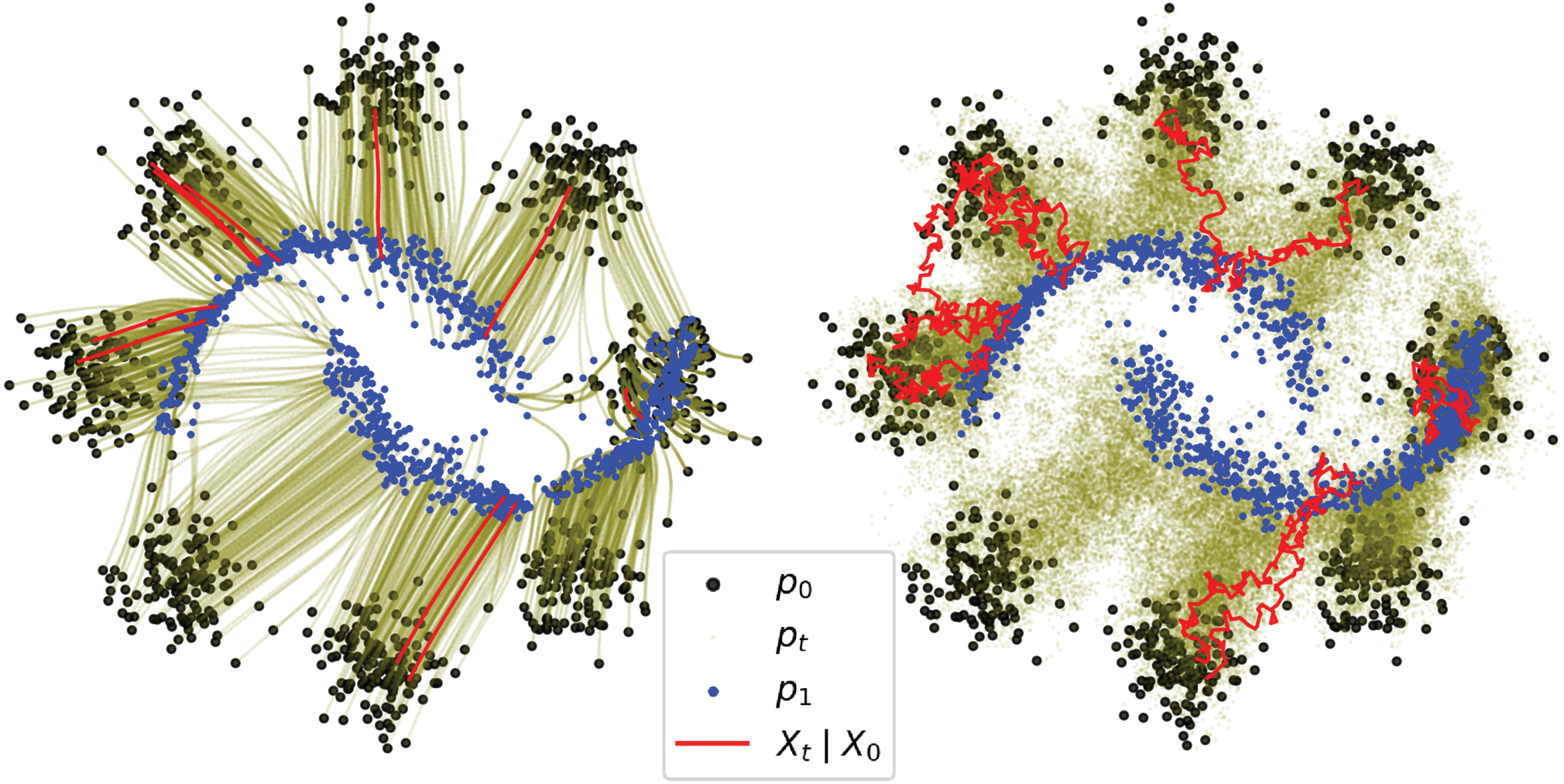
- Implement basic flow matching training and inference
- Implement optimal transport conditional flow matching
- Implement variance preserving variant
- How does it compare to MLE training?

https://github.com/atong01/ProbAI-2025-flow-matching/

Part III: Bridge Matching and Mean Flows



Bridge Matching



Let's first recall diffusion!

Diffusion and Score Models

Forward SDE $dx_t = f_t(x_t)dt + g_t dw_t$ Data → Noise

Reverse SDE $d\bar{x}_t = (f_t(x_t) - g_t^2 \nabla \log p_t)dt + g_t d\bar{w}_t$ Noise → Data

Learn the score by regressing to conditional scores:

$$\min_\theta \mathbb{E}_{p_{data}, p_t(x|x_{data})} \left[\| s_t^\theta(x) - \nabla \log p_t(x \mid x_{data}) \|^2 \right]$$

Simulation-free

Known SDEs: Variance Exploding
Variance Preserving

Different Integration

Forward SDE $dx_t = f_t(x_t)dt + g_t dw_t$ Data → Noise

Reverse SDE $d\bar{x}_t = (f_t(x_t) - g_t^2 \nabla \log p_t)dt + g_t d\bar{w}_t$ Noise → Data

Forward Probability Flow ODE $dx_t = (f_t(x_t) - \frac{1}{2} g_t^2 \nabla \log p_t)dt$ Data → Noise

Reverse Probability Flow ODE $d\bar{x}_t = -(f_t(x_t) - \frac{1}{2} g_t^2 \nabla \log p_t)dt$ Noise → Data

Different Integration with deterministic drift

$$u_t = f_t - \frac{1}{2} g_t^2 \nabla \log p_t$$

Forward SDE

$$dx_t = (u_t + \frac{1}{2} g_t^2 \nabla \log p_t)dt + g_t dw_t$$

Data → Noise

Reverse SDE

$$d\bar{x}_t = (u_t - \frac{1}{2} g_t^2 \nabla \log p_t)dt + g_t d\bar{w}_t$$

Noise → Data

Forward Probability Flow ODE

$$dx_t = u_t(x_t)dt$$

Data → Noise

Reverse Probability Flow ODE

$$d\bar{x}_t = -u_t(x_t)dt$$

Noise → Data

Different Integration with deterministic drift

$$u_t = f_t - \frac{1}{2} g_t^2 \nabla \log p_t$$

Bridge Matching!

Forward SDE

$$dx_t = (u_t + \frac{1}{2} g_t^2 \nabla \log p_t) dt + g_t dw_t$$

Data → Noise

Reverse SDE

$$d\bar{x}_t = (u_t - \frac{1}{2} g_t^2 \nabla \log p_t) dt + g_t d\bar{w}_t$$

Noise → Data

Forward Probability Flow ODE

$$dx_t = u_t(x_t) dt$$

Data → Noise

Reverse Probability Flow ODE

$$d\bar{x}_t = - u_t(x_t) dt$$

Noise → Data

Different Integration with deterministic drift

$$u_t = f_t - \frac{1}{2} g_t^2 \nabla \log p_t$$

Forward SDE

$$dx_t = (u_t + \frac{1}{2} g_t^2 \nabla \log p_t) dt + g_t dw_t$$

Data → Noise

Reverse SDE

$$d\bar{x}_t = (u_t - \frac{1}{2} g_t^2 \nabla \log p_t) dt + g_t d\bar{w}_t$$

Noise → Data

Forward Probability Flow ODE

$$dx_t = u_t(x_t) dt$$

Data → Noise

Flow Matching!

Reverse Probability Flow ODE

$$d\bar{x}_t = - u_t(x_t) dt$$

Noise → Data

Different Integration with deterministic drift

$$u_t = f_t - \frac{1}{2} g_t^2 \nabla \log p_t$$

Forward SDE

$$dx_t = (u_t + \frac{1}{2} g_t^2 \nabla \log p_t) dt + g_t dw_t$$

Data → Noise

Reverse SDE

$$d\bar{x}_t = (u_t - \frac{1}{2} g_t^2 \nabla \log p_t) dt + g_t d\bar{w}_t$$

Noise → Data

Forward Probability Flow ODE

$$dx_t = u_t(x_t) dt$$

Data → Noise

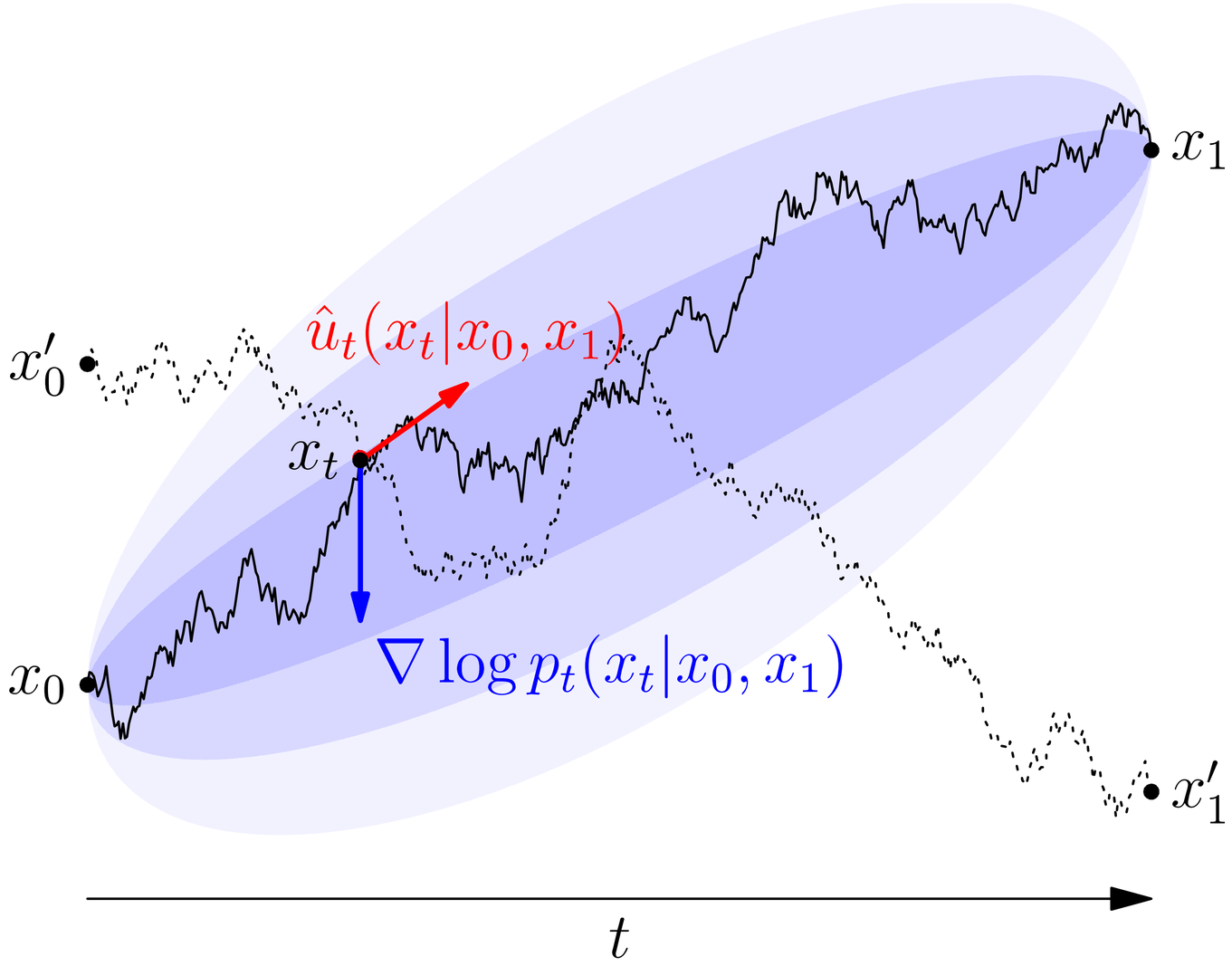
Score + Flow Matching!

Reverse Probability Flow ODE

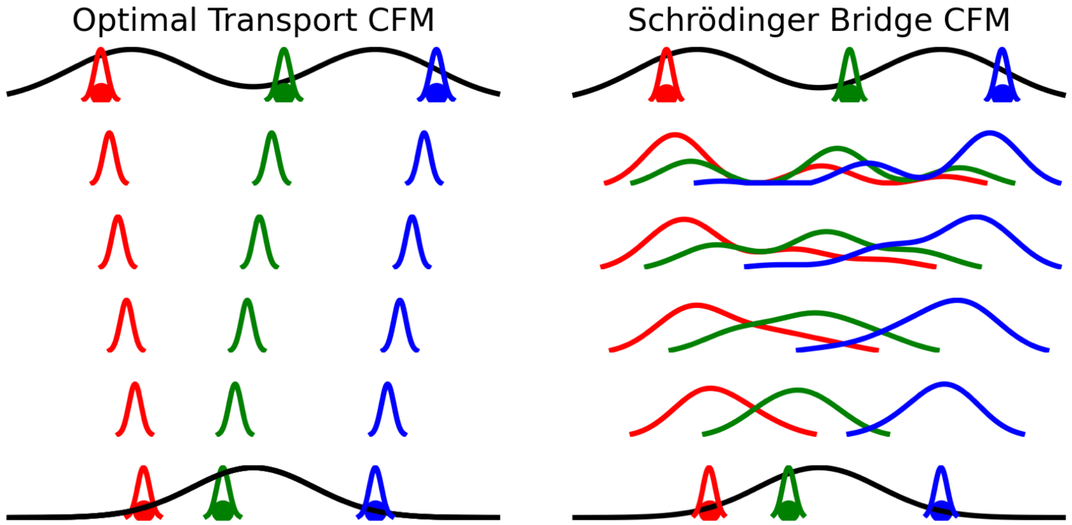
$$d\bar{x}_t = - u_t(x_t) dt$$

Noise → Data

A Geometric Intuition



Schrödinger Bridges



Problem statement

$$\mathbb{P}^\star = \min_{\mathbb{P}: p_0 = q_0, p_1 = q_1} \mathrm{KL}(\mathbb{P} \,\|\, \mathbb{Q})$$

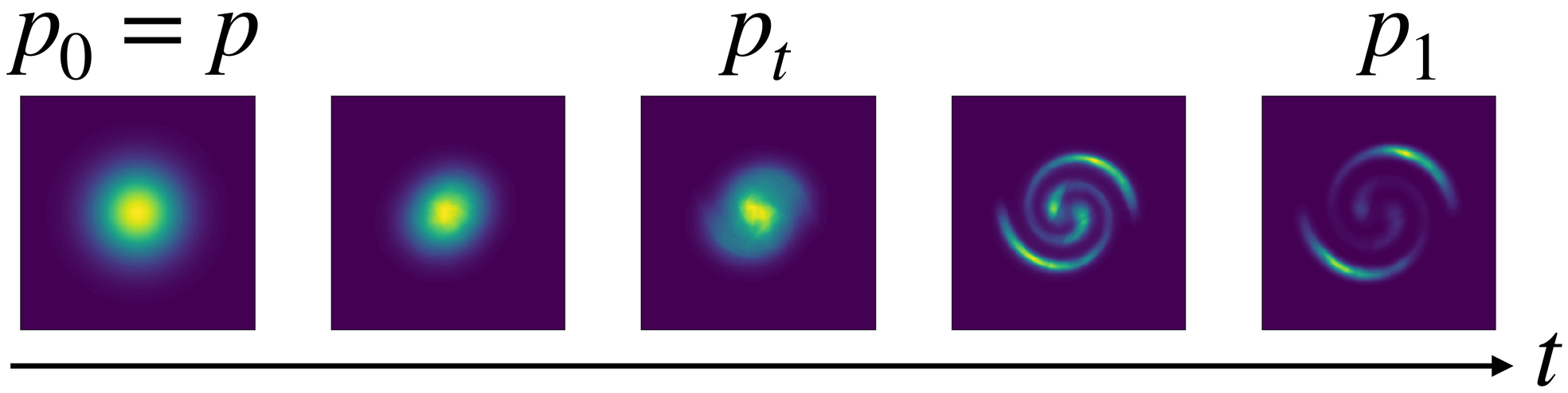
“Most likely stochastic process under observation”

Diffusion Schrödinger Bridges as mixtures of Brownian bridges

$$p_t(x) = \int p_t(x|x_0, x_1)\, d\pi^\star_{2\sigma^2}(x_0, x_1)$$

$$p_t(x|x_0, x_1) = \mathcal{N}(x; (1-t)x_0 + tx_1, \sigma^2 t(1-t))$$

How do we speed up Inference?



Consistency Models

Yang Song [1] **Prafulla Dhariwal** [1] **Mark Chen** [1] **Ilya Sutskever** [1]

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

ONE STEP DIFFUSION VIA SHORTCUT MODELS

Kevin Frans
UC Berkeley
kvfrans@berkeley.edu

Danijar Hafner
UC Berkeley

Sergey Levine
UC Berkeley

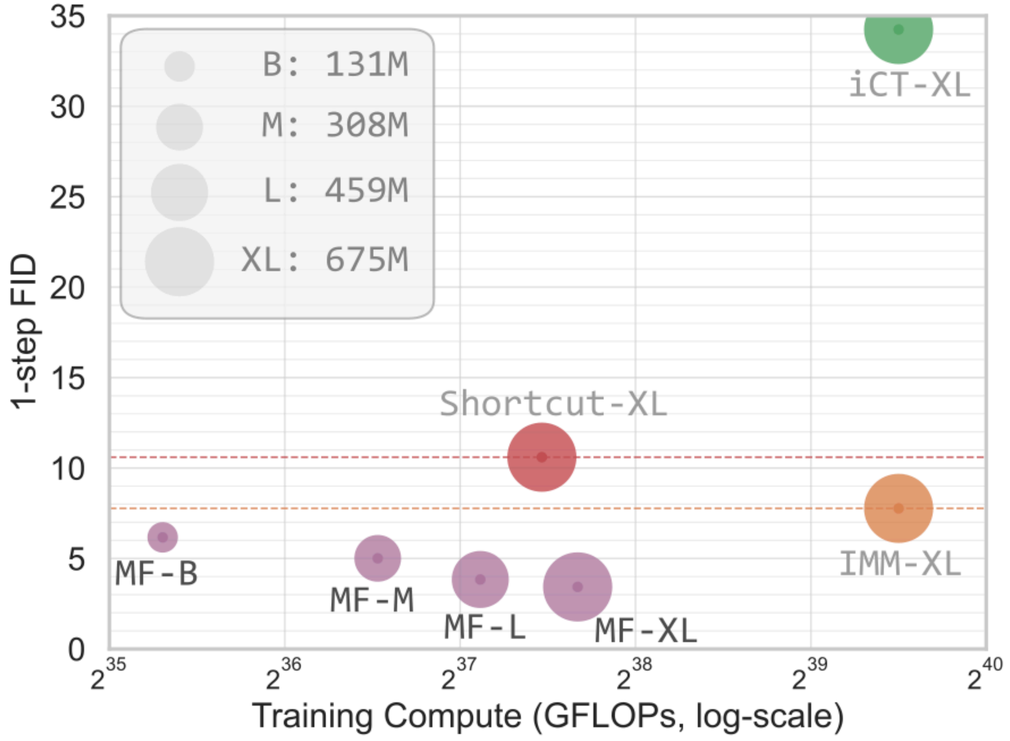
Pieter Abbeel
UC Berkeley

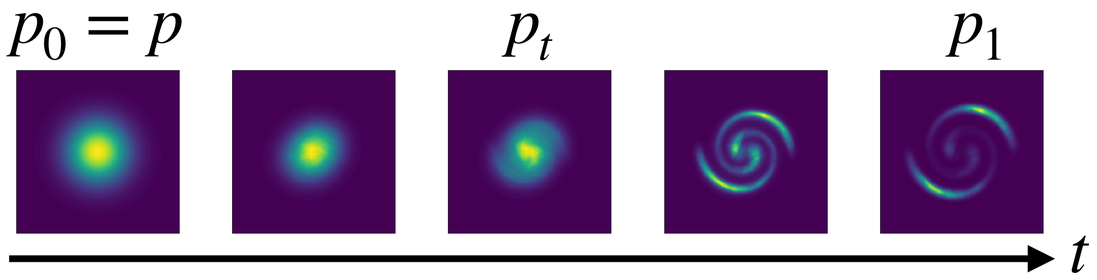
How do we speed up Inference? — Mean Flows

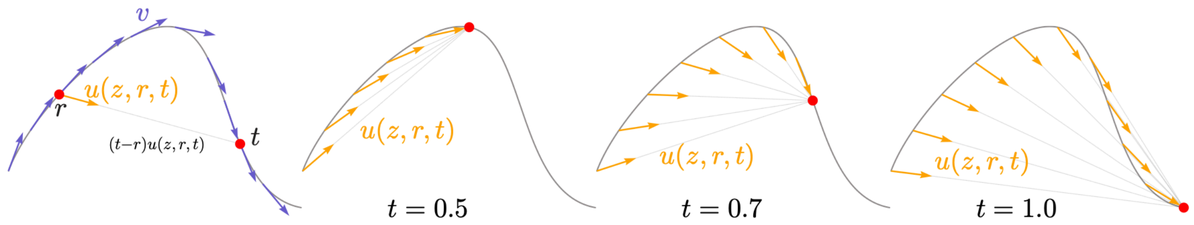
Mean Flows for One-step Generative Modeling

Zhengyang Geng[1]* Mingyang Deng[2] Xingjian Bai[2] J. Zico Kolter[1] Kaiming He[2]

[1]CMU [2]MIT







Idea:
Condition on current and target time!

$$u_\theta(x_t, t)$$

$$u_\theta(x_t, r, t)$$

How do we learn $\boldsymbol{u_\theta(x_t, r, t)}$?

$$(t-r)u(x_t, r, t) = \int_r^t v(x_t, \tau)d\tau$$

Derivative w.r.t. t

$$u(x_t, r, t) + (t-r)\frac{d}{dt}u(x_t, r, t) = v(x_t, t)$$

Rearrange

$$u_{target}(x_t, r, t) = v(x_t, t) - (t-r)\frac{d}{dt}u(x_t, r, t)$$

$$L(\theta) = \min \mathbb{E}_{t,\, q(z),\, p_t(x|z)} \| u_t^\theta(x, r, t) - stopgrad(u_{target}) \|^2$$

How do we learn $u_\theta(x_t, r, t)$?

$$(t-r)u(x_t, r, t) = \int_r^t v(x_t, \tau)d\tau$$

Derivative w.r.t. t

$$u(x_t, r, t) + (t-r)\frac{d}{dt}u(x_t, r, t) = v(x_t, t)$$

Rearrange

$$u_{target}(x_t, r, t) = v(x_t, t) - (t$$

$$L(\theta) = \min \mathbb{E}_{t,\, q(z),\, p_t(x|z)} \| u_t^\theta(x, r, t) -$$

Algorithm 1 MeanFlow: Training.

Note: in PyTorch and JAX, `jvp` returns the function output and JVP.

```
# fn(z, r, t): function to predict u
# x: training batch

t, r = sample_t_r()
e = randn_like(x)

z = (1 - t) * x + t * e
v = e - x

u, dudt = jvp(fn, (z, r, t), (v, 0, 1))

u_tgt = v - (t - r) * dudt
error = u - stopgrad(u_tgt)

loss = metric(error)
```

Algorithm 2 MeanFlow: 1-step Sampling

```
e = randn(x_shape)
x = e - fn(e, r=0, t=1)
```

Practical Considerations

Three practical considerations

- Model conditioning

EDM Kerras et al. 2022

		VP [49]	VE [49]	iDDPM [37] + DDIM [47]	Ours ("EDM")
Sampling (Section 3)					
ODE solver		Euler	Euler	Euler	2nd order Heun
Time steps	$t_{i<N}$	$1 + \frac{i}{N-1}(\epsilon_s - 1)$	$\sigma_{max}^2 \left(\sigma_{min}^2/\sigma_{max}^2\right)^{\frac{i}{N-1}}$	$u_{\lfloor j_0 + \frac{M-1-j_0}{N-1} i + \frac{1}{2} \rfloor}$, where $u_M = 0$, $u_{j-1} = \sqrt{\frac{u_j^2+1}{\max(\bar{\alpha}_{j-1}/\bar{\alpha}_j, C_1)} - 1}$	$\left(\sigma_{max}^{\frac{1}{\rho}} + \frac{i}{N-1}\left(\sigma_{min}^{\frac{1}{\rho}} - \sigma_{max}^{\frac{1}{\rho}}\right)\right)^{\rho}$
Schedule	$\sigma(t)$	$\sqrt{e^{\frac{1}{2}\beta_d t^2 + \beta_{min} t} - 1}$	$\sqrt{t}$	t	t
Scaling	$s(t)$	$1/\sqrt{e^{\frac{1}{2}\beta_d t^2 + \beta_{min} t}}$	1	1	1
Network and preconditioning (Section 5)					
Architecture of F_θ		DDPM++	NCSN++	DDPM	(any)
Skip scaling	$c_{skip}(\sigma)$	1	1	1	$\sigma_{data}^2/\left(\sigma^2 + \sigma_{data}^2\right)$
Output scaling	$c_{out}(\sigma)$	$-\sigma$	σ	$-\sigma$	$\sigma \cdot \sigma_{data}/\sqrt{\sigma_{data}^2 + \sigma^2}$
Input scaling	$c_{in}(\sigma)$	$1/\sqrt{\sigma^2 + 1}$	1	$1/\sqrt{\sigma^2 + 1}$	$1/\sqrt{\sigma^2 + \sigma_{data}^2}$
Noise cond.	$c_{noise}(\sigma)$	$(M-1)\,\sigma^{-1}(\sigma)$	$\ln(\frac{1}{2}\sigma)$	$M - 1 - \arg\min_j \lvert u_j - \sigma \rvert$	$\frac{1}{4}\ln(\sigma)$
Training (Section 5)					
Noise distribution		$\sigma^{-1}(\sigma) \sim \mathcal{U}(\epsilon_t, 1)$	$\ln(\sigma) \sim \mathcal{U}(\ln(\sigma_{min}), \ln(\sigma_{max}))$	$\sigma = u_j, \quad j \sim \mathcal{U}\{0, M-1\}$	$\ln(\sigma) \sim \mathcal{N}(P_{mean}, P_{std}^2)$
Loss weighting	$\lambda(\sigma)$	$1/\sigma^2$	$1/\sigma^2$	$1/\sigma^2$ (note: *)	$\left(\sigma^2 + \sigma_{data}^2\right)/(\sigma \cdot \sigma_{data})^2$
Parameters		$\beta_d = 19.9, \beta_{min} = 0.1$ $\epsilon_s = 10^{-3}, \epsilon_t = 10^{-5}$ $M = 1000$	$\sigma_{min} = 0.02$ $\sigma_{max} = 100$	$\bar{\alpha}_j = \sin^2(\frac{\pi}{2}\frac{j}{M(C_2+1)})$ $C_1 = 0.001, C_2 = 0.008$ $M = 1000, j_0 = 8^\dagger$	$\sigma_{min} = 0.002, \sigma_{max} = 80$ $\sigma_{data} = 0.5, \rho = 7$ $P_{mean} = -1.2, P_{std} = 1.2$

* iDDPM also employs a second loss term L_{vlb} † In our tests, $j_0 = 8$ yielded better FID than $j_0 = 0$ used by iDDPM

$\sigma(t)$	$\sqrt{e^{\frac{1}{2}\beta_{\mathrm{d}}t^2+\beta_{\mathrm{min}}t}-1}$	$\sqrt{t}$	t	t
$s(t)$	$1/\sqrt{e^{\frac{1}{2}\beta_{\mathrm{d}}t^2+\beta_{\mathrm{min}}t}}$	1	1	1

nd preconditioning (Section 5)

e of F_θ	DDPM++	NCSN++	DDPM	(any)
g $c_{\mathrm{skip}}(\sigma)$	1	1	1	$\sigma_{\mathrm{data}}^2/(\sigma^2+\sigma_{\mathrm{data}}^2)$
ing $c_{\mathrm{out}}(\sigma)$	$-\sigma$	σ	$-\sigma$	$\sigma\cdot\sigma_{\mathrm{data}}/\sqrt{\sigma_{\mathrm{data}}^2+\sigma^2}$
g $c_{\mathrm{in}}(\sigma)$	$1/\sqrt{\sigma^2+1}$	1	$1/\sqrt{\sigma^2+1}$	$1/\sqrt{\sigma^2+\sigma_{\mathrm{data}}^2}$
$c_{\mathrm{noise}}(\sigma)$	$(M-1)\,\sigma^{-1}(\sigma)$	$\ln(\frac{1}{2}\sigma)$	$M-1-\arg\min_j\lvert u_j-\sigma\rvert$	$\frac{1}{4}\ln(\sigma)$

ection 5)

bution	$\sigma^{-1}(\sigma)\sim\mathcal{U}(\epsilon_{\mathrm{t}},1)$	$\ln(\sigma)\sim\mathcal{U}(\ln(\sigma_{\mathrm{min}}),\ln(\sigma_{\mathrm{max}}))$	$\sigma=u_j,\ \ j\sim\mathcal{U}\{0,M-1\}$	$\ln(\sigma)\sim\mathcal{N}(P_{\mathrm{mean}},P_{\mathrm{std}}^2)$
ting $\lambda(\sigma)$	$1/\sigma^2$	$1/\sigma^2$	$1/\sigma^2$ (note: *)	$(\sigma^2+\sigma_{\mathrm{data}}^2)/(\sigma\cdot\sigma_{\mathrm{data}})^2$
s	$\beta_{\mathrm{d}}=19.9,\beta_{\mathrm{min}}=0.1$	$\sigma_{\mathrm{min}}=0.02$	$\bar{\alpha}_j=\sin^2(\frac{\pi}{2}\frac{j}{M(C_2+1)})$	$\sigma_{\mathrm{min}}=0.002,\sigma_{\mathrm{max}}=80$
	$\epsilon_{\mathrm{s}}=10^{-3},\epsilon_{\mathrm{t}}=10^{-5}$	$\sigma_{\mathrm{max}}=100$	$C_1=0.001,C_2=0.008$	$\sigma_{\mathrm{data}}=0.5,\rho=7$
	$M=1000$		$M=1000,j_0=8^\dagger$	$P_{\mathrm{mean}}=-1.2,P_{\mathrm{std}}=1.2$

PM also employs a second loss term L_{vlb} † In our tests, $j_0=8$ yielded better FID than $j_0=0$ used by iDDPM

EDM Kerras et al. 2022

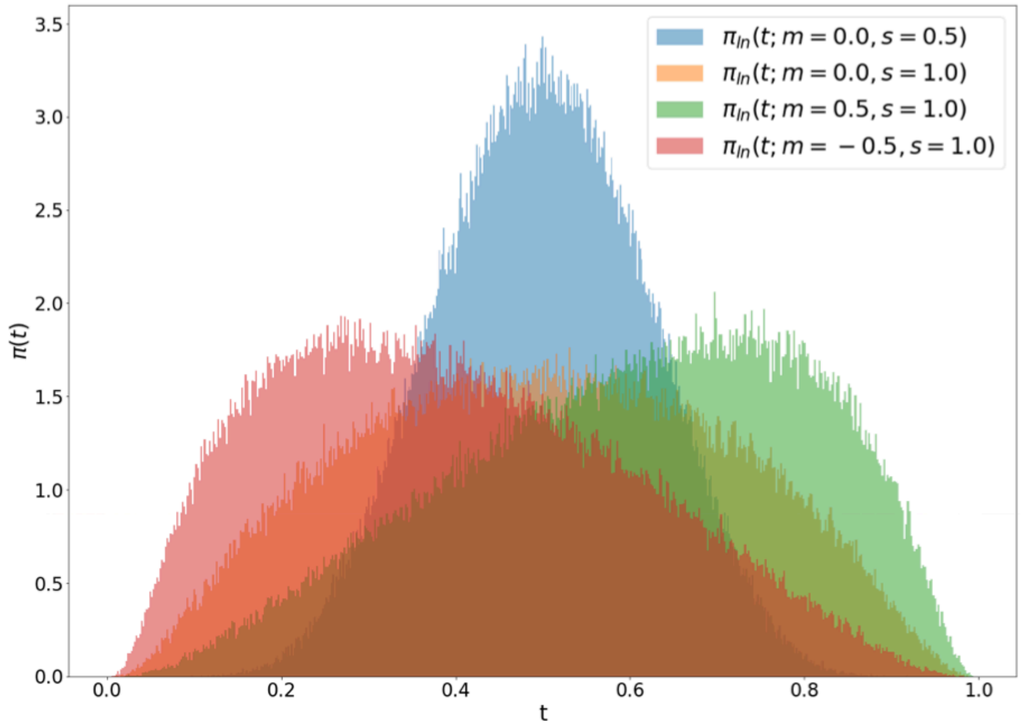
Actually is equivalent to a flow matching model!

Practical Considerations

Three practical considerations

- Model conditioning
- Sampling t

Stable diffusion 3 found orange was best



Practical Considerations

Three practical considerations

- Model conditioning
- Sampling t
- Exponential Moving Average on weights

$$\mathbf{x}_0^{\mathrm{EMA}} = \mathbf{x}_0, \text{ and } \mathbf{x}_{t+1}^{\mathrm{EMA}} = \alpha\, \mathbf{x}_t^{\mathrm{EMA}} + (1-\alpha)\, \mathbf{x}_{t+1}.$$

Practical Considerations

Three practical considerations

- Model conditioning
- Sampling t
- Exponential Moving Average on weights

Two unintuitive things / open questions

- Gaussian prior works way too well
- OT doesn't seem to help beyond a certain batch size

In theory we would think choosing our prior p_0 close to p_1 would be great. In practice $p_0 = \mathcal{N}(0,1)$ is almost always better

Practical Considerations

Three practical considerations

- Model conditioning
- Sampling t
- Exponential Moving Average on weights

Two unintuitive things / open questions

- Gaussian prior works way too well
- OT doesn't seem to help beyond a certain batch size

SIMPLE REFLOW: IMPROVED TECHNIQUES FOR FAST FLOW MODELS

Beomsu Kim
Apple and KAIST

Yu-Guan Hsieh
Apple

Michal Klein
Apple

Marco Cuturi
Apple

Jong Chul Ye
KAIST

Bahjat Kawar
Apple

James Thornton
Apple

Tested batch sizes up to 50k, didn't see improvement beyond 2-4k.

Method	CIFAR10			AFHQv2			FFHQ			ImageNet (cond.)			Reference
	NFE	FID	STN	NFE	FID	STN	NFE	FID	STN	NFE	FID	STN	
DM ODE													
EDM	35	1.97	14.19	79	1.96	28.41	79	2.39	27.15	79	2.30	26.76	(Karras et al., 2022)
	9	37.91	–	9	28.03	–	9	56.84	–	9	35.46	–	—〃—
DPM-Solver	9	4.98	–	–	–	–	9	9.26	–	9	6.64	–	(Lu et al., 2022)
AMED-Solver	9	2.63	–	–	–	–	9	4.24	–	9	5.60	–	(Zhou et al., 2024)
FM ODE													
MinCurv	9	8.76	5.87	9	13.63	10.45	9	10.44	10.49	–	–	–	(Lee et al., 2023)
FM-OT	142	6.35	–	–	–	–	–	–	–	138	14.45	–	(Lipman et al., 2023)
OT-CFM	100	4.44	–	–	–	–	–	–	–	–	–	–	(Tong et al., 2023)
MOT-50	–	–	–	–	–	–	–	–	–	132	11.82	–	(Pooladian et al., 2023)
FM*	100	2.96	10.73	100	2.73	16.20	100	3.30	16.71	–	–	–	Baseline
MOT-512*	100	3.29	8.77	100	5.53	13.45	100	4.69	14.29	–	–	–	—〃—
MOT-1024*	100	3.18	8.59	100	5.83	13.45	100	4.84	14.07	–	–	–	—〃—
MOT-4096*	100	3.16	8.34	100	6.18	12.68	100	4.92	13.47	–	–	–	—〃—
ReFlow	110	3.36	–	–	–	–	–	–	–	–	–	–	(Liu et al., 2022)
Simple ReFlow*	9	2.23	1.64	9	2.30	3.30	9	2.84	2.87	9	3.49	2.72	Ours
+ Guidance*	9	1.98	2.49	9	1.91	5.60	9	2.67	3.24	9	1.74	3.92	—〃—

Additional Resources

TorchCFM: a Conditional Flow Matching library

paper arxiv.2302.00482 | paper arxiv.2307.03672 | PyTorch 1.8+ | Lightning 1.6+

Config Hydra 1.2 | Code Style Black | Pre-commit enabled | TorchCFM Tests passing

codecov 34% | Code Quality Main failing | License MIT | Lightning-Hydra-Template

downloads 128k | downloads/month 14k

Flow Matching Guide and Code

Yaron Lipman[1], **Marton Havasi**[1], **Peter Holderrieth**[2], **Neta Shaul**[3], **Matt Le**[1], **Brian Karrer**[1], **Ricky T. Q. Chen**[1], **David Lopez-Paz**[1], **Heli Ben-Hamu**[3], **Itai Gat**[1]

[1]FAIR at Meta, [2]MIT CSAIL, [3]Weizmann Institute of Science

Flow Matching (FM) is a recent framework for generative modeling that has achieved state-of-the-art performance across various domains, including image, video, audio, speech, and biological structures. This guide offers a comprehensive and self-contained review of FM, covering its mathematical foundations, design choices, and extensions. By also providing a PyTorch package featuring relevant examples (*e.g.*, image and text generation), this work aims to serve as a resource for both novice and experienced researchers interested in understanding, applying and further developing FM.

Date: December 10, 2024

Code: `flow_matching` library at https://github.com/facebookresearch/flow_matching

Meta

Elucidating the Design Space of Diffusion-Based Generative Models

Tero Karras NVIDIA

Miika Aittala NVIDIA

Timo Aila NVIDIA

Samuli Laine NVIDIA

ICLR Blogposts 2025

A Visual Dive into Conditional Flow Matching

Conditional flow matching (CFM) was introduced by three simultaneous papers at ICLR 2023, through different approaches (conditional matching, rectifying flows and stochastic interpolants).
The main part of this post, Section 2, explains CFM by using both visual intuitions and insights on its probabilistic formulations. Section 1 introduces nomalizing flows; it can be skipped by reader familiar with the topic, or that wants to cover them later. Section 3 opens on the links between CFM and other approaches, and ends with a 'CFM playground'.

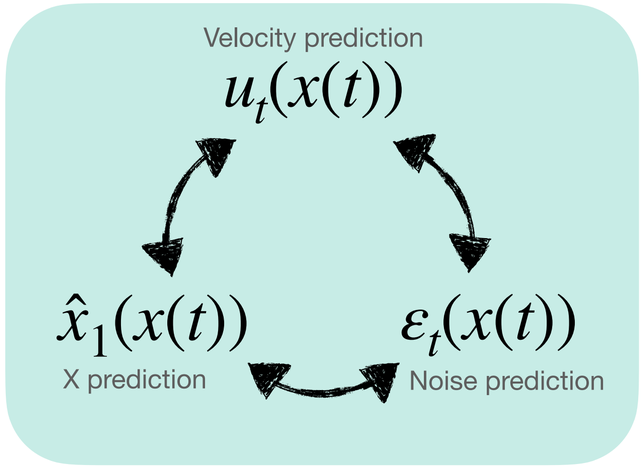
AUTHORS
Anne Gagneux
Ségolène Martin
Rémi Emonet
Quentin Bertrand
Mathurin Massias

AFFILIATIONS
INRIA, ENS de Lyon
Technische Universität Berlin
Inria, Université Jean Monnet
Inria, Université Jean Monnet
Inria, ENS de Lyon

PUBLISHED
April 28, 2025

Thank you and Questions?

Flow/Diffusion Models



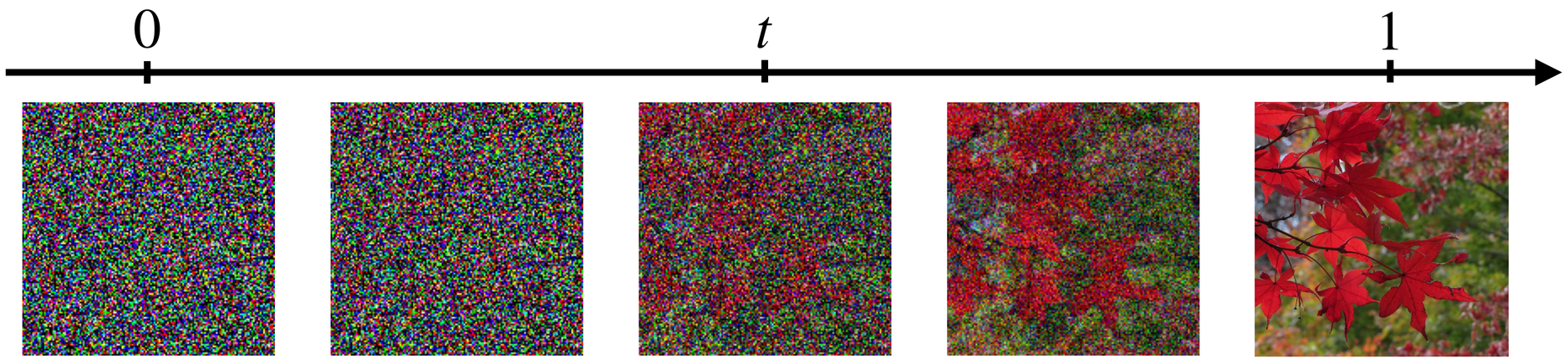
Neural network

$\dot{x}(t) = u_t(x(t))$

$x(1) = \phi(x) \sim q$

$x(t) \sim p_t$

$x(0) = x \sim p$



Ho J., Jain A., Abbeel P. *Denoising Diffusion Probabilistic Models* (NeurIPS 2020)
Lipman Y., Chen RTQ., **BH**, Nickel M., Le M. *Flow Matching for Generative Modeling* (ICLR 2023)