Flow Matching Tutorial

Heli Ben-Hamu

Ricky T. Q. Chen

Yaron Lipman

WEIZMANN INSTITUTE OF SCIENCE

Meta

Agenda

[40 mins]	01 **Flow Matching Basics**
[35 mins]	02 **Flow Matching Advanced Designs**
[35 mins]	03 **Model Adaptation**
[30 mins]	04 **Generator Matching and Discrete Flows**
[10 mins]	05 **Codebase demo**

Flow Matching Guide and Code

a Meta FAIR release

Paper

Github

Yaron Lipman

Marton Havasi

Peter Holderrieth

Neta Shaul

Matt Le

Brian Karrer

Ricky T. Q. Chen

David Lopez-Paz

Heli Ben-Hamu

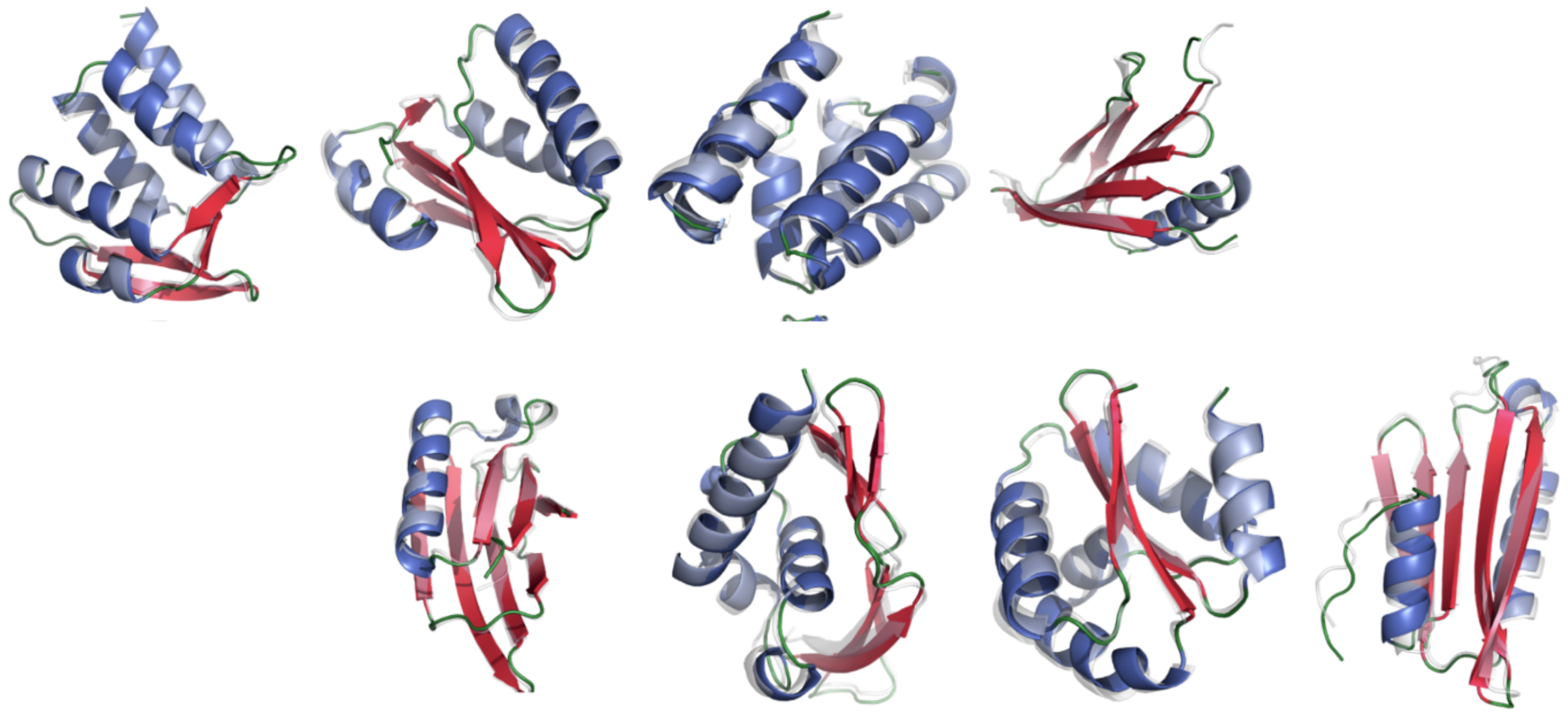
Itai Gat

01 Flow Matching Basics

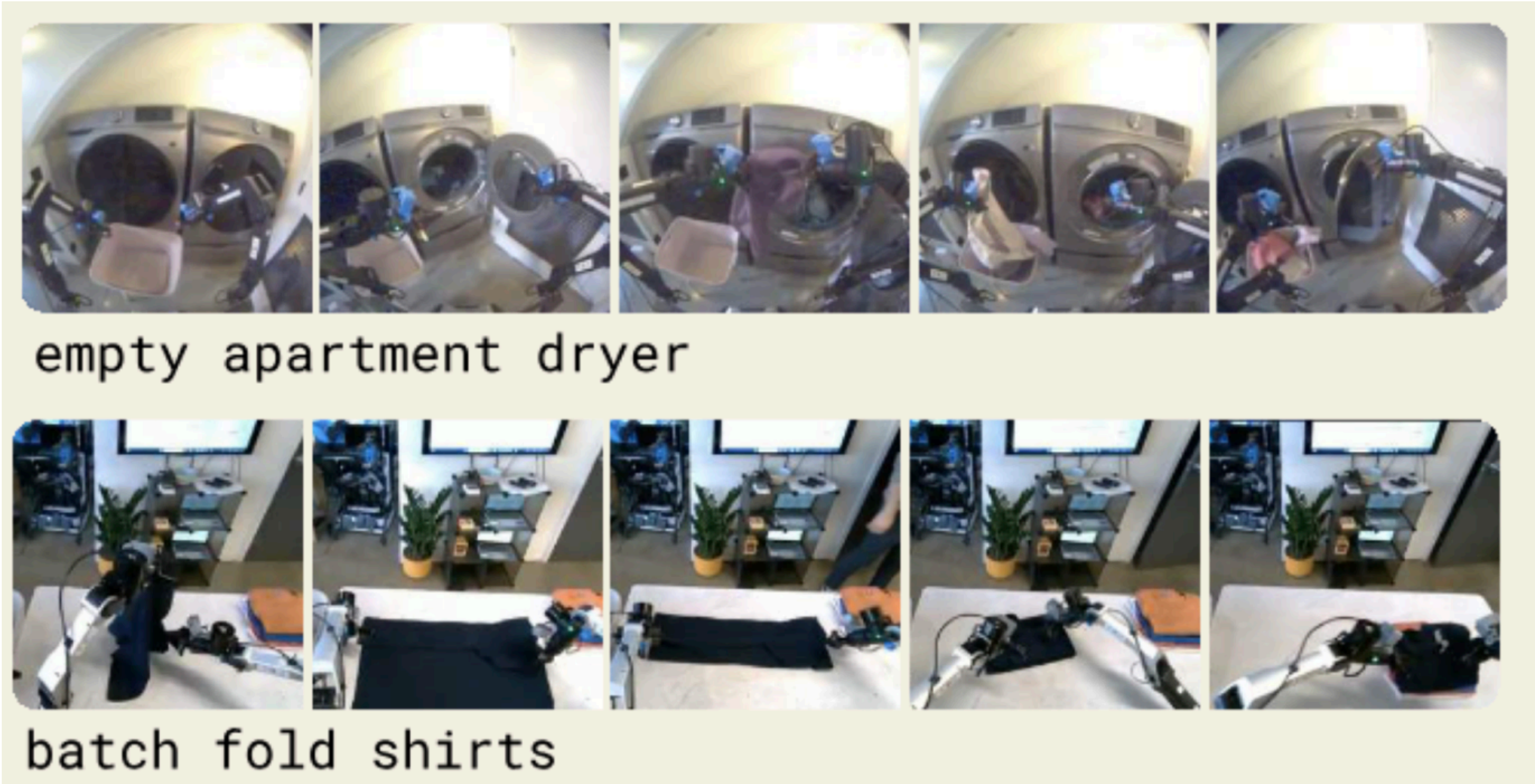
Flow Matching at SCALE

Text-2-Video
MovieGen, Meta

Text-2-Image
Stable Diffusion 3



Protein Generation
Huguet el a. 24



Robot Action Model
Black et a. 24

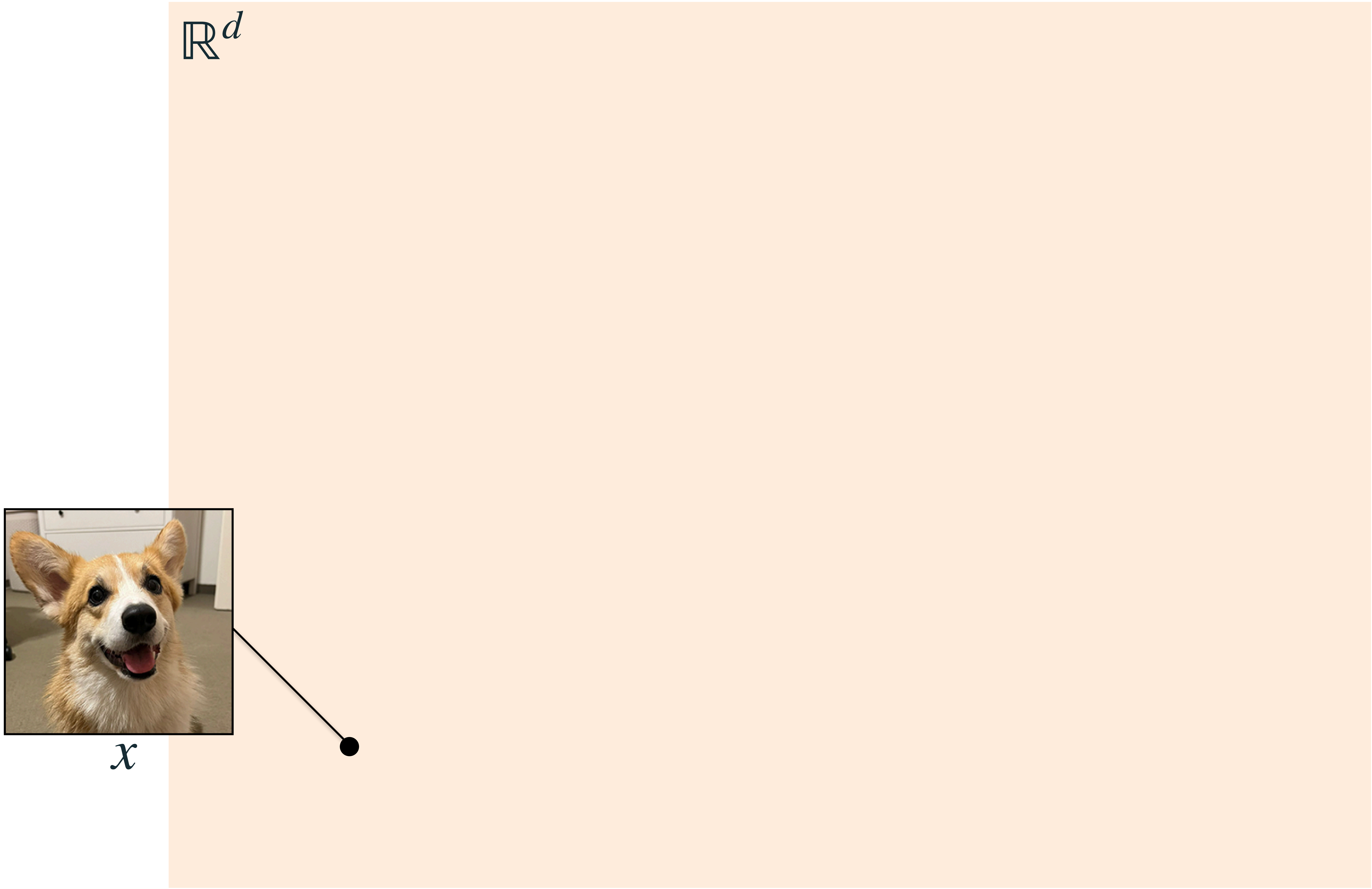
WHAT IS FLOW MATCHING?

A scalable method to train flow generative models.

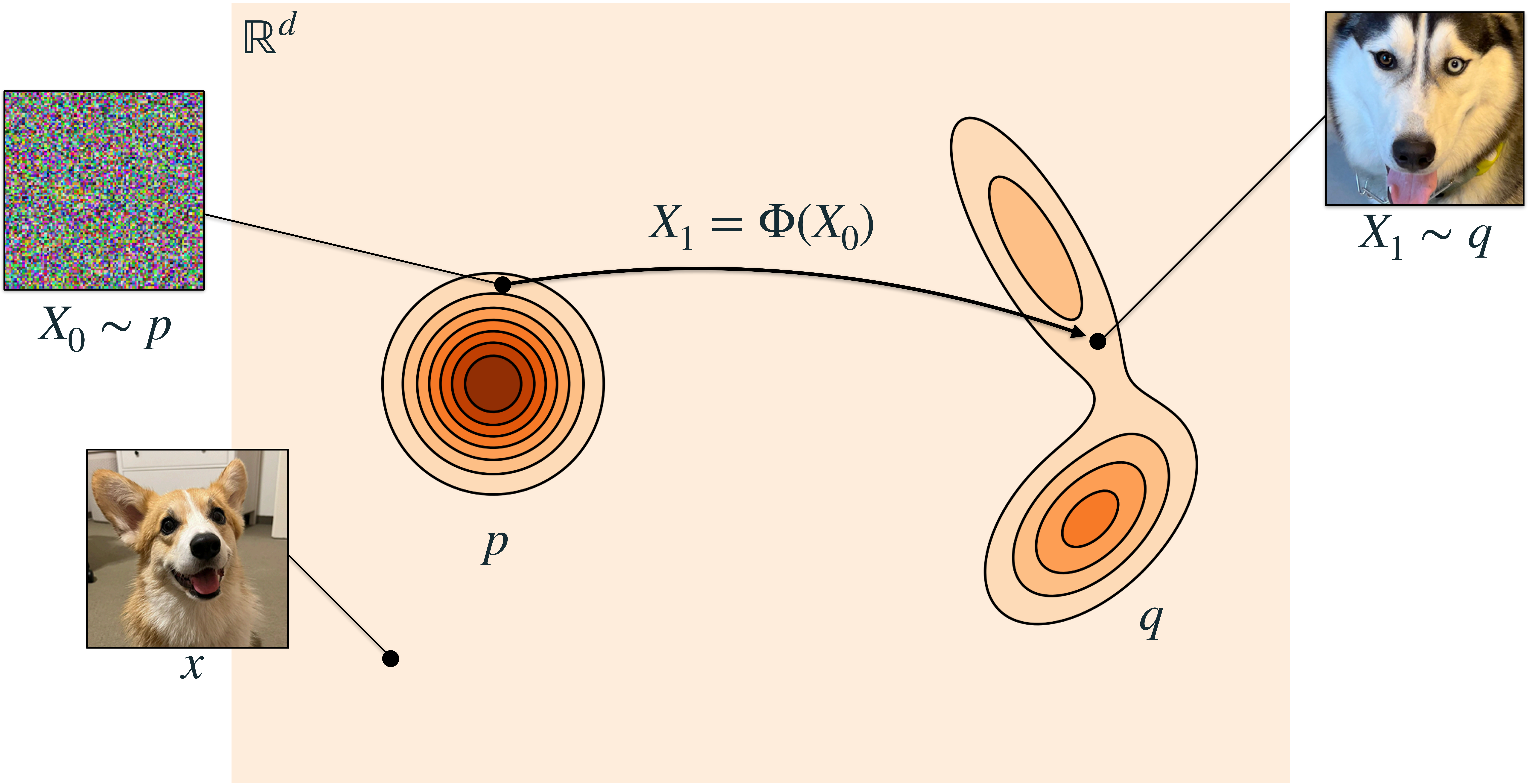
HOW DOES IT WORK?

Train by regressing a velocity, sample by following the velocity

The Generative Modeling Problem



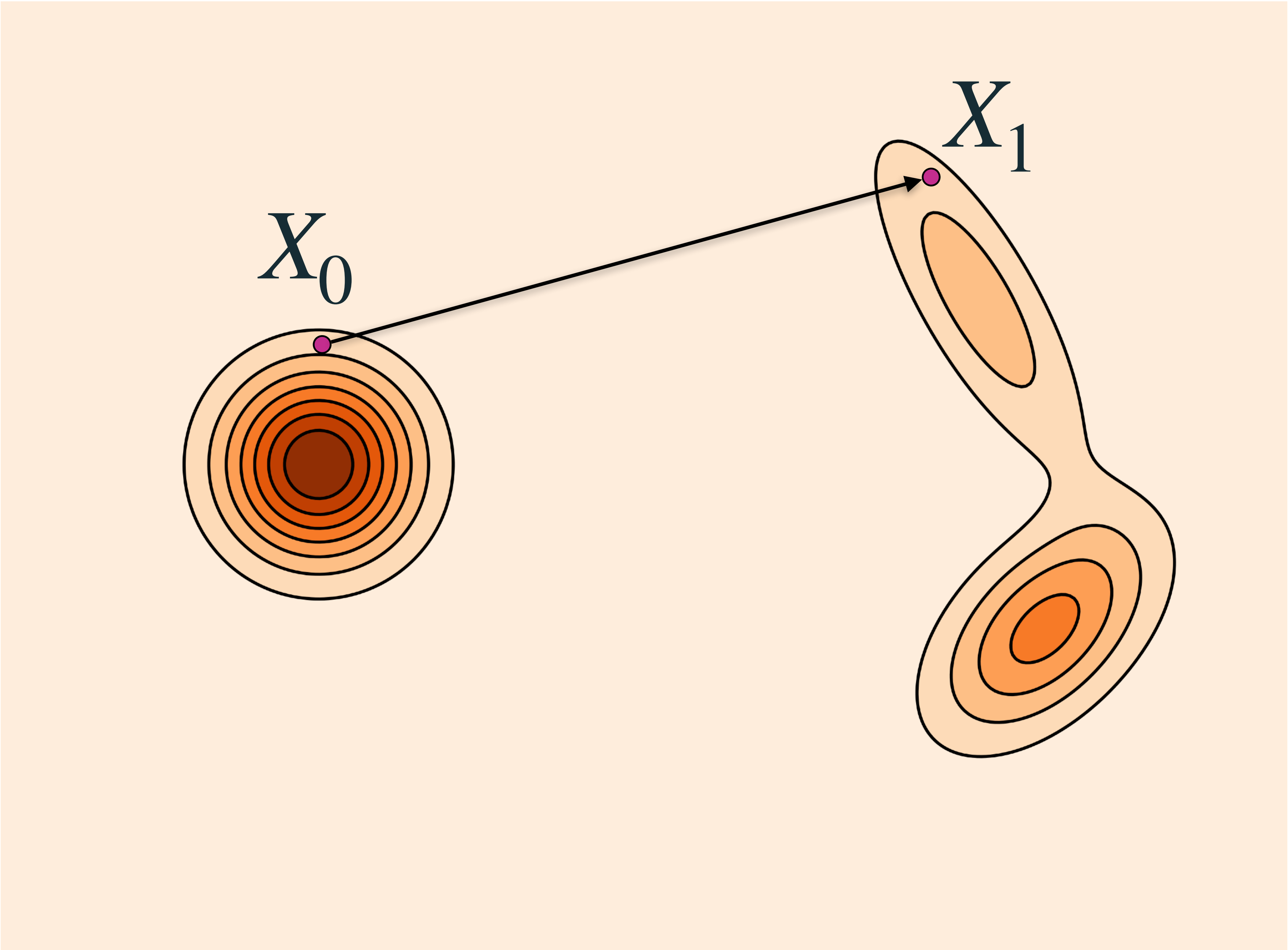
The Generative Modeling Problem



Model

- Direct map

$$X_1 = \Phi(X_0)$$



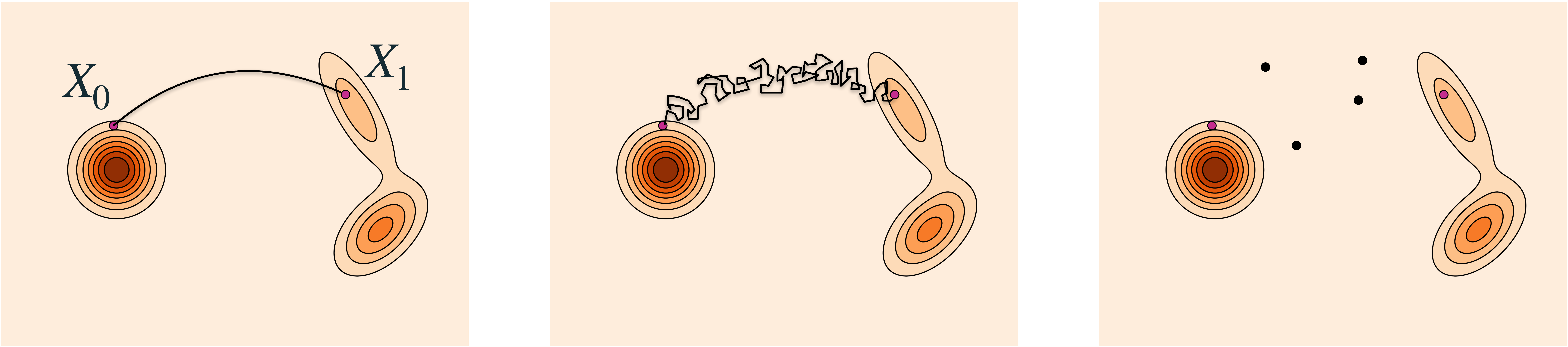
- **Pros**: efficient sampling
- **Cons**: not probabilistic
- **Cons**: min-max loss

"Generative Adversarial Networks" Goodfellow et al. (2014)

Model

- Continuous-time Markov process $(X_t)_{0 \leq t \leq 1}$

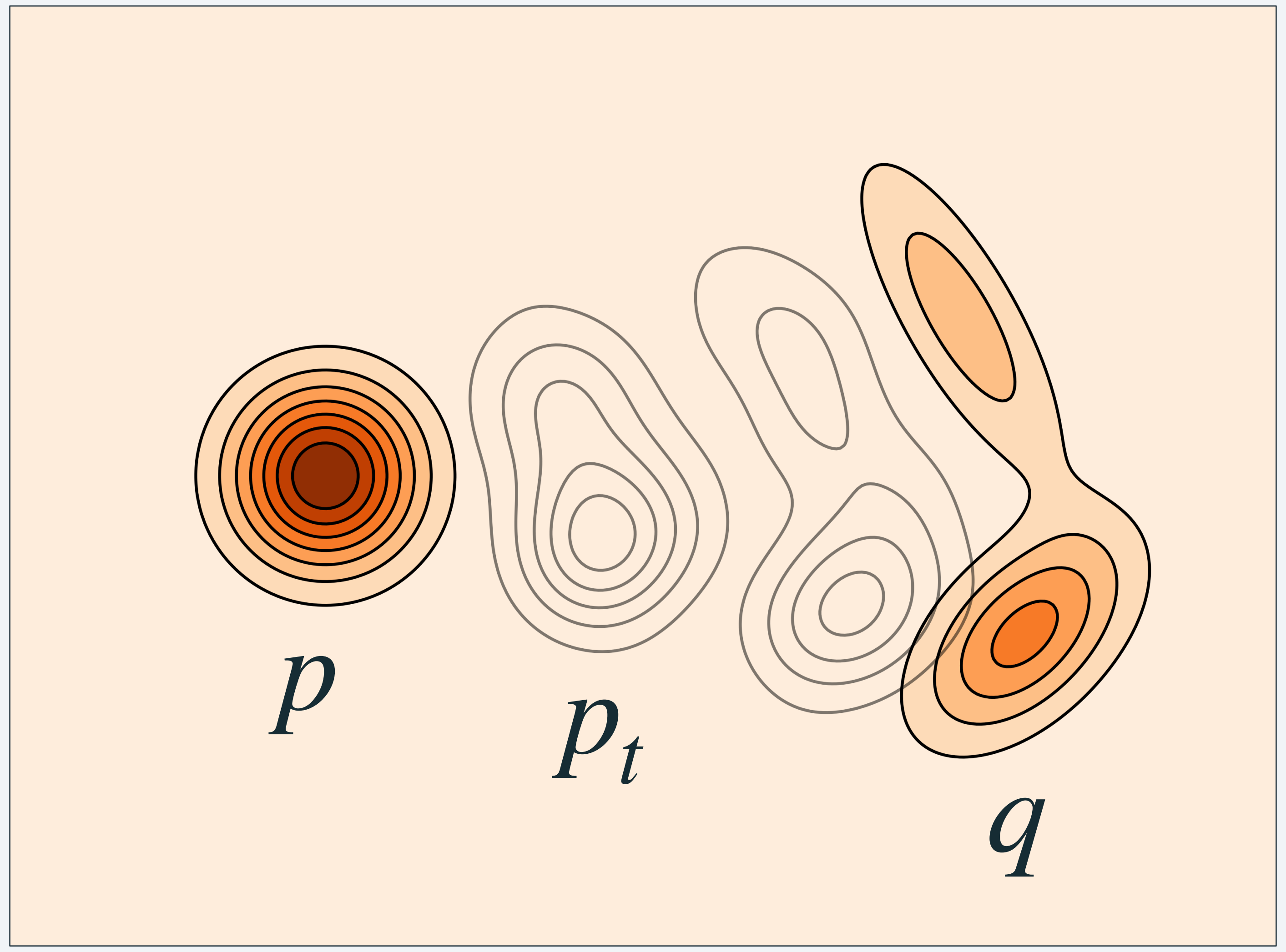
$$X_{t+h} \leftarrow \boxed{\Phi_{t+h|t}}(X_t)$$

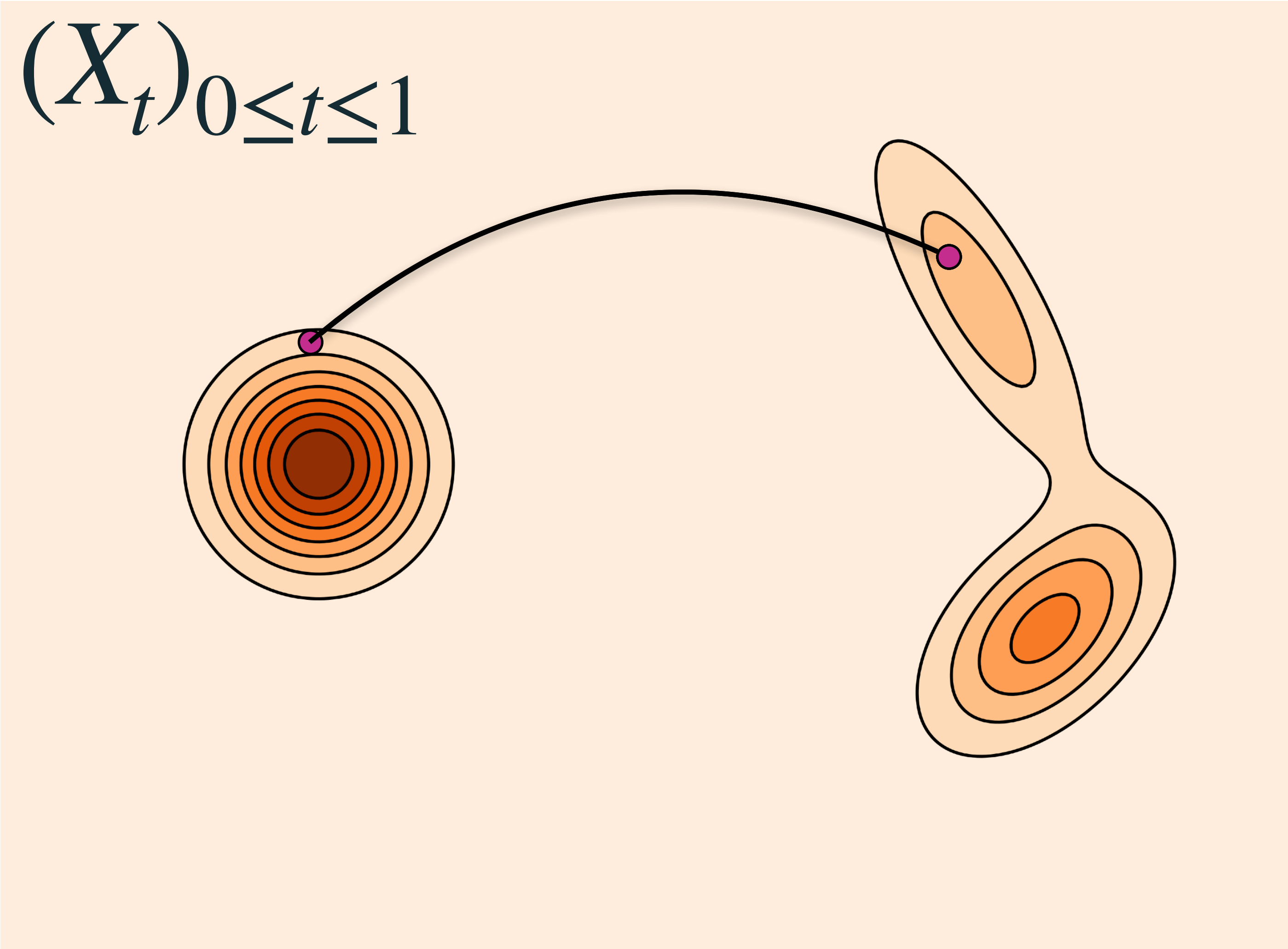


Flow **Diffusion** **Jump**

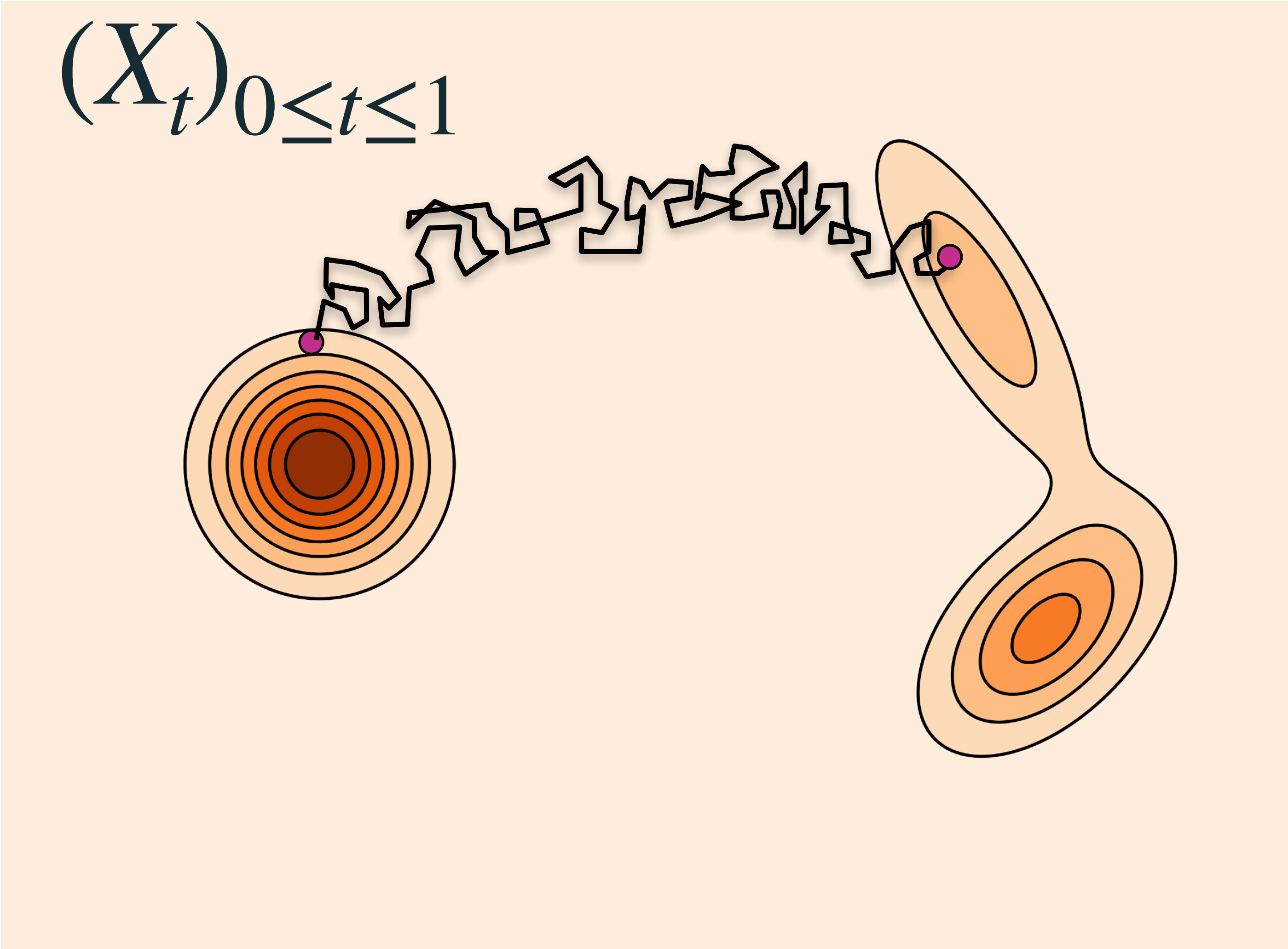
Marginal probability path

$$X_t \sim p_t$$

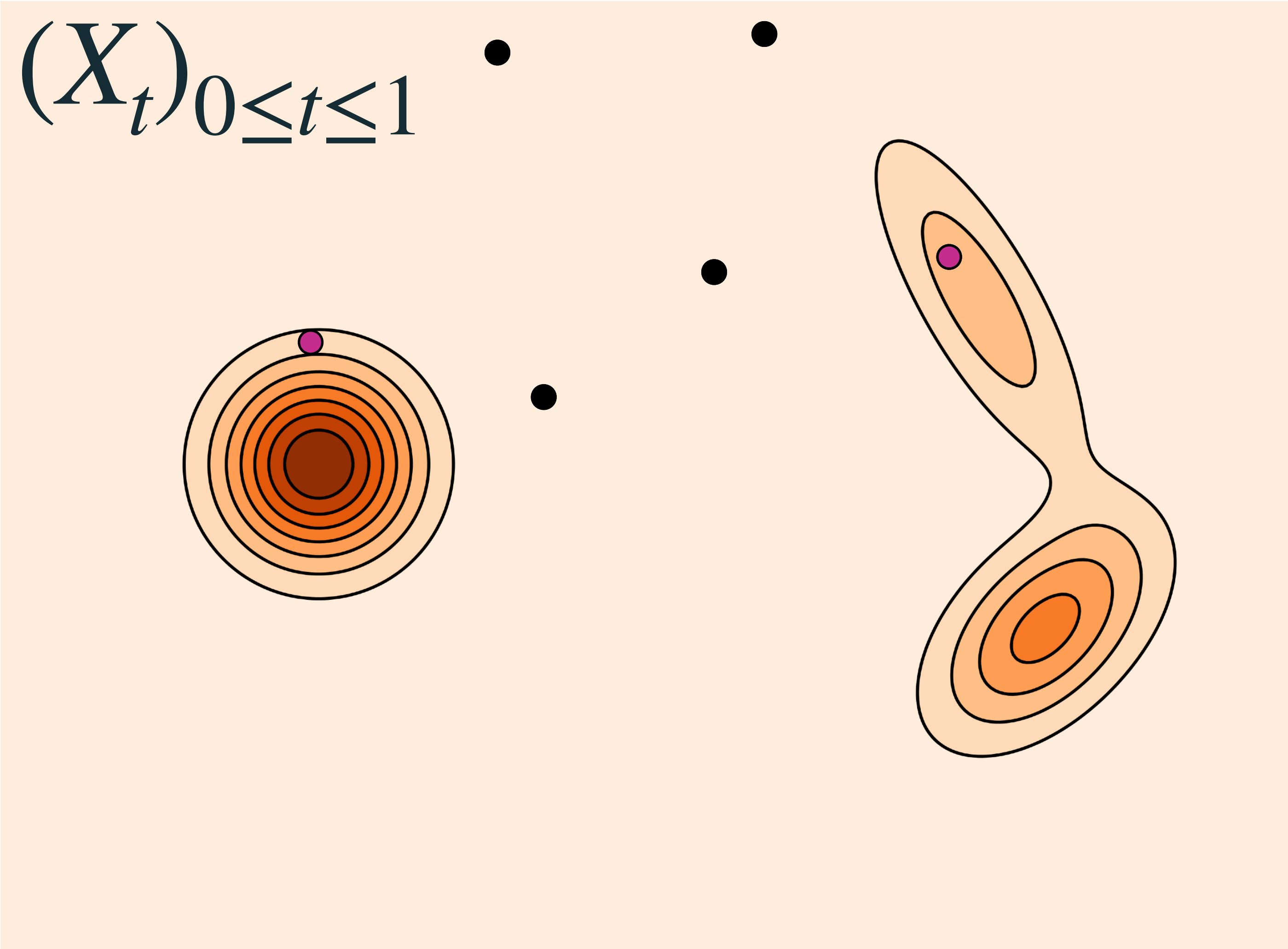




Flow

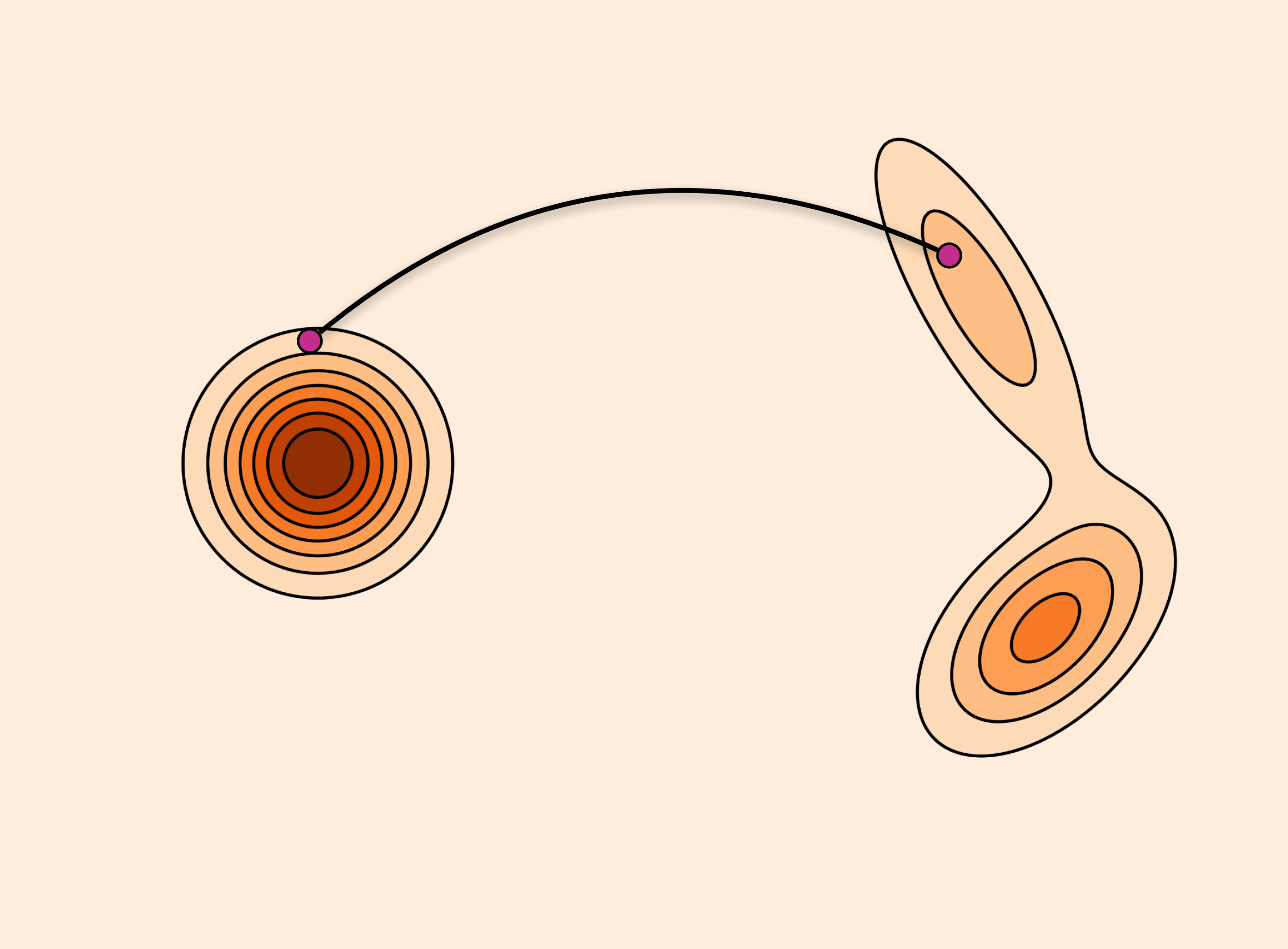


Diffusion



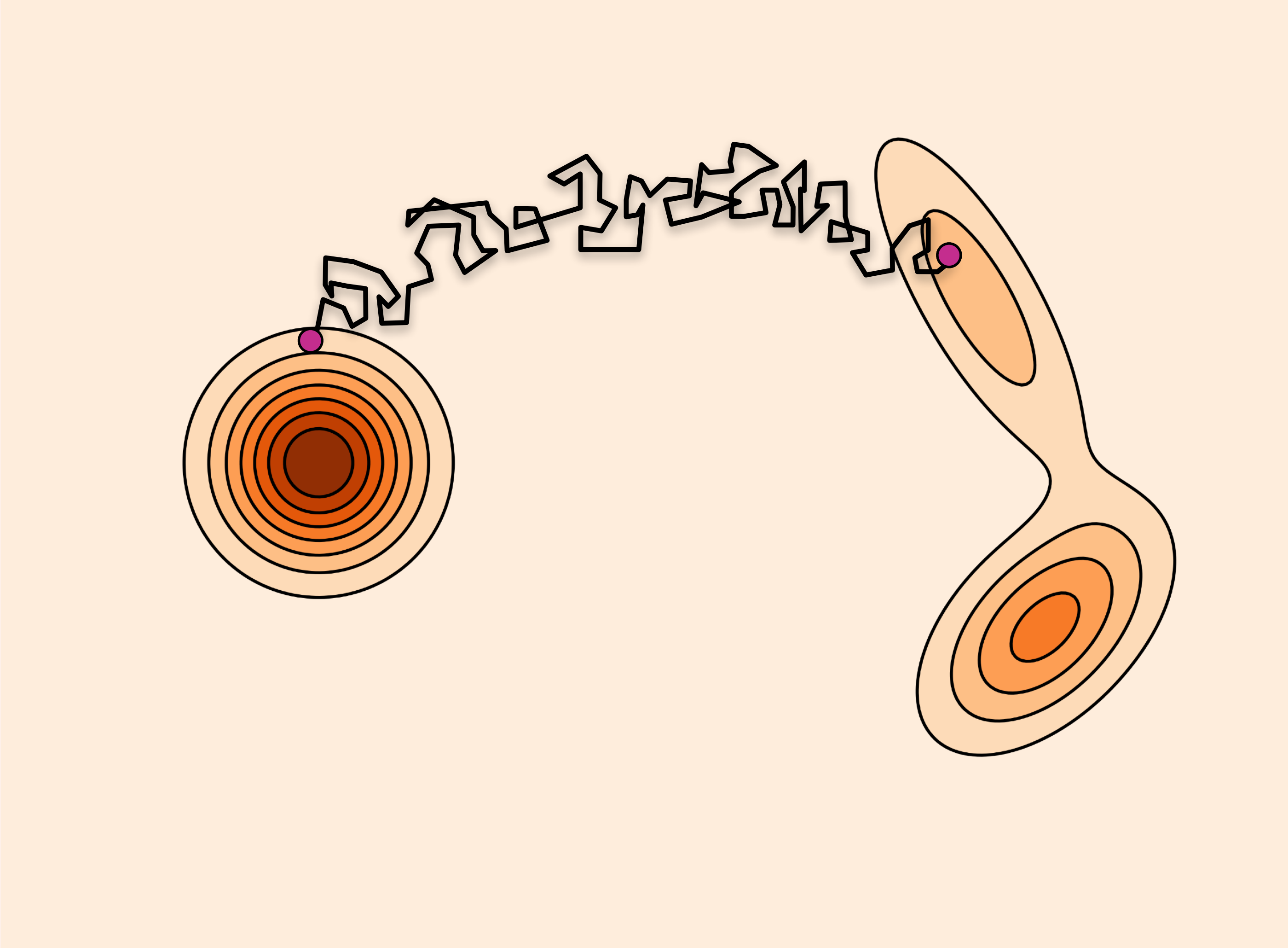
Jump

- For now, we focus on flows...

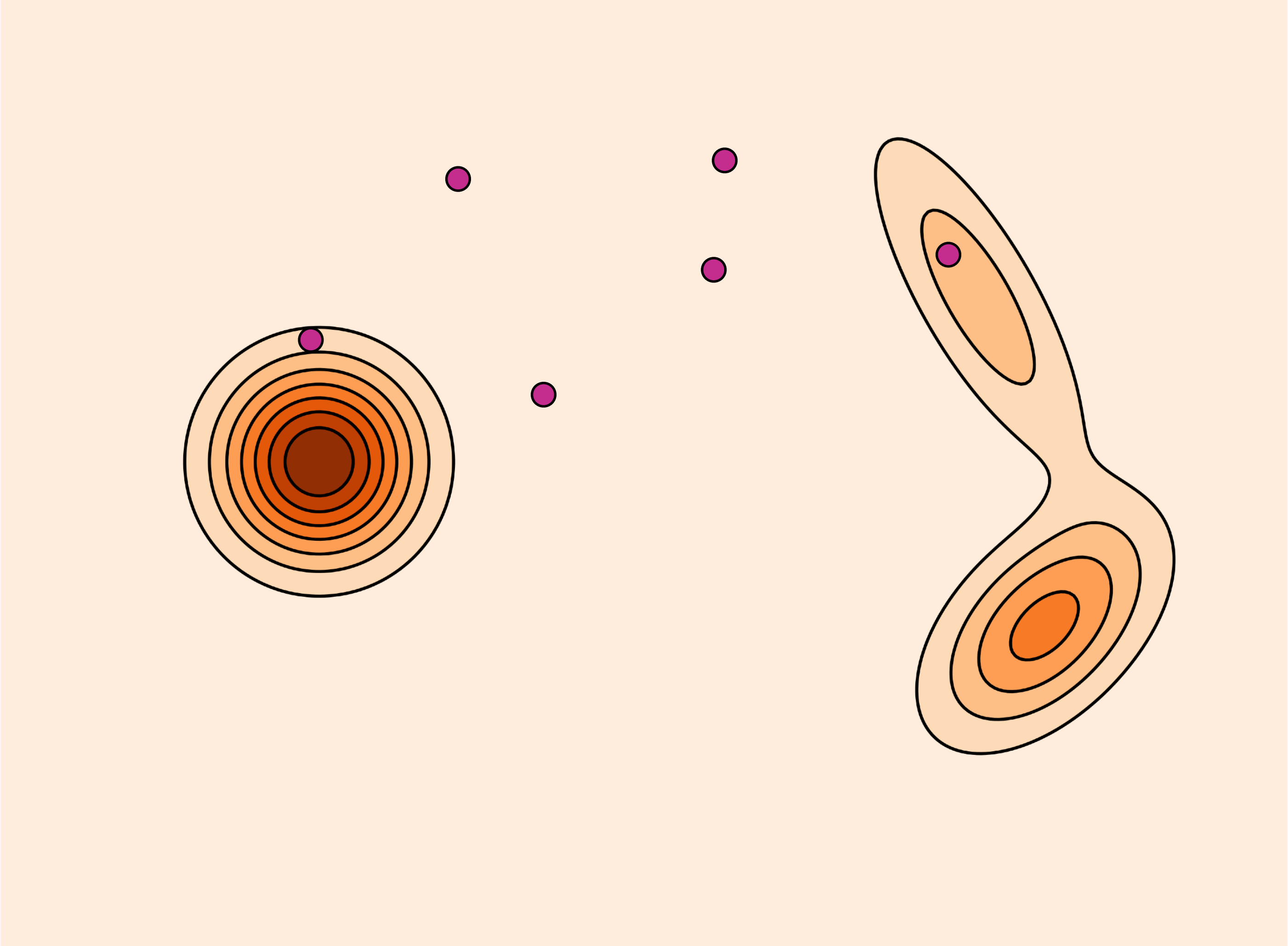


Flow

- Simple
- Faster sampling
- Exact likelihood estimator
- Flexible, easier to build



Diffusion



Jump

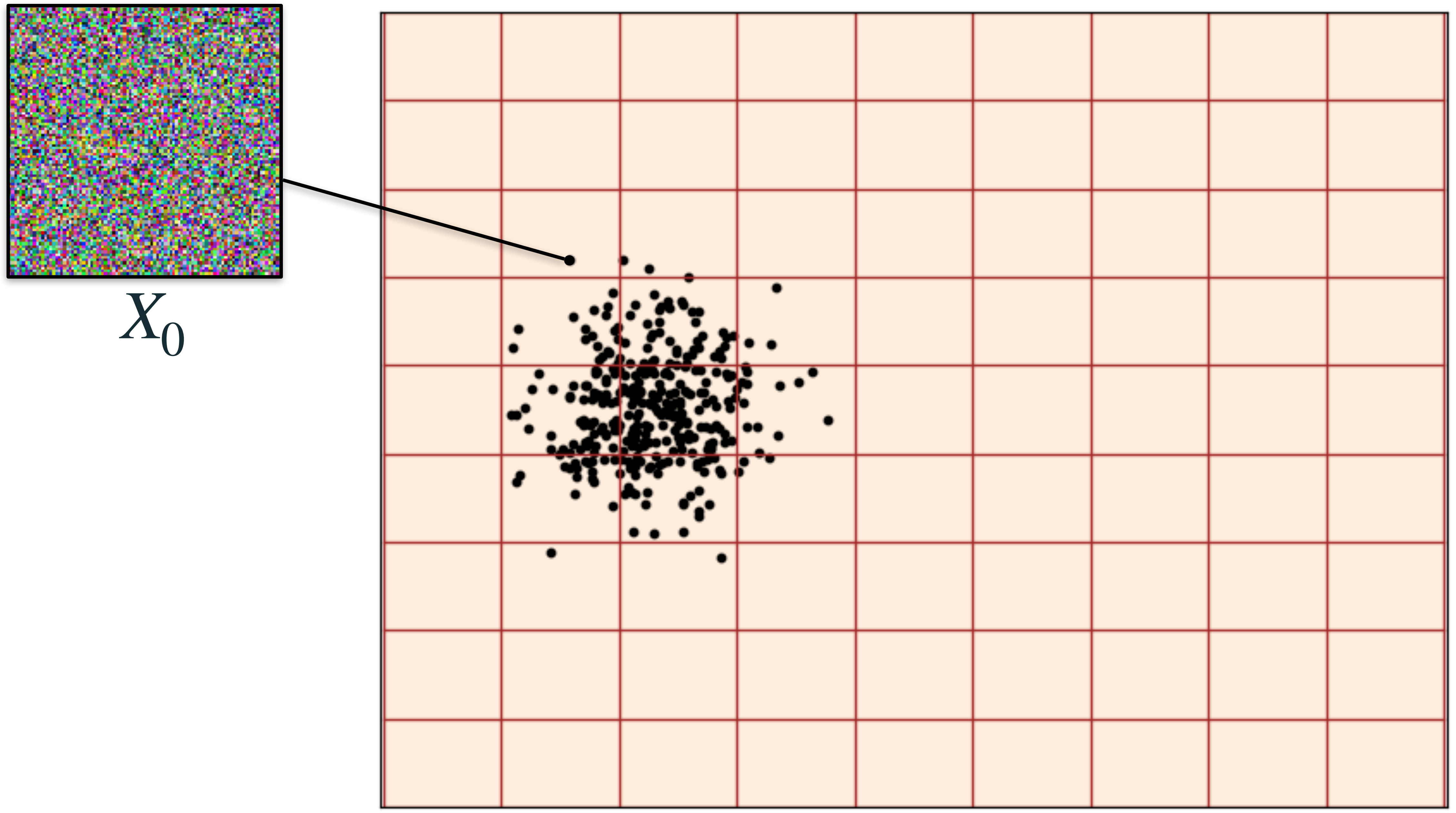
- Larger design space
- Slower sampling
- ELBO

Flow as a generative model

$$X_t = \psi_t(X_0)\ , \quad t \in [0,1]$$

Warping

Source $X_0 \sim p$



- Markov: $X_{t+h} = \psi_{t+h|t}(X_t)$

Flow = Velocity

$$\frac{\mathrm{d}}{\mathrm{d}t}\psi_t(x) = u_t(\psi_t(x))$$

Flow

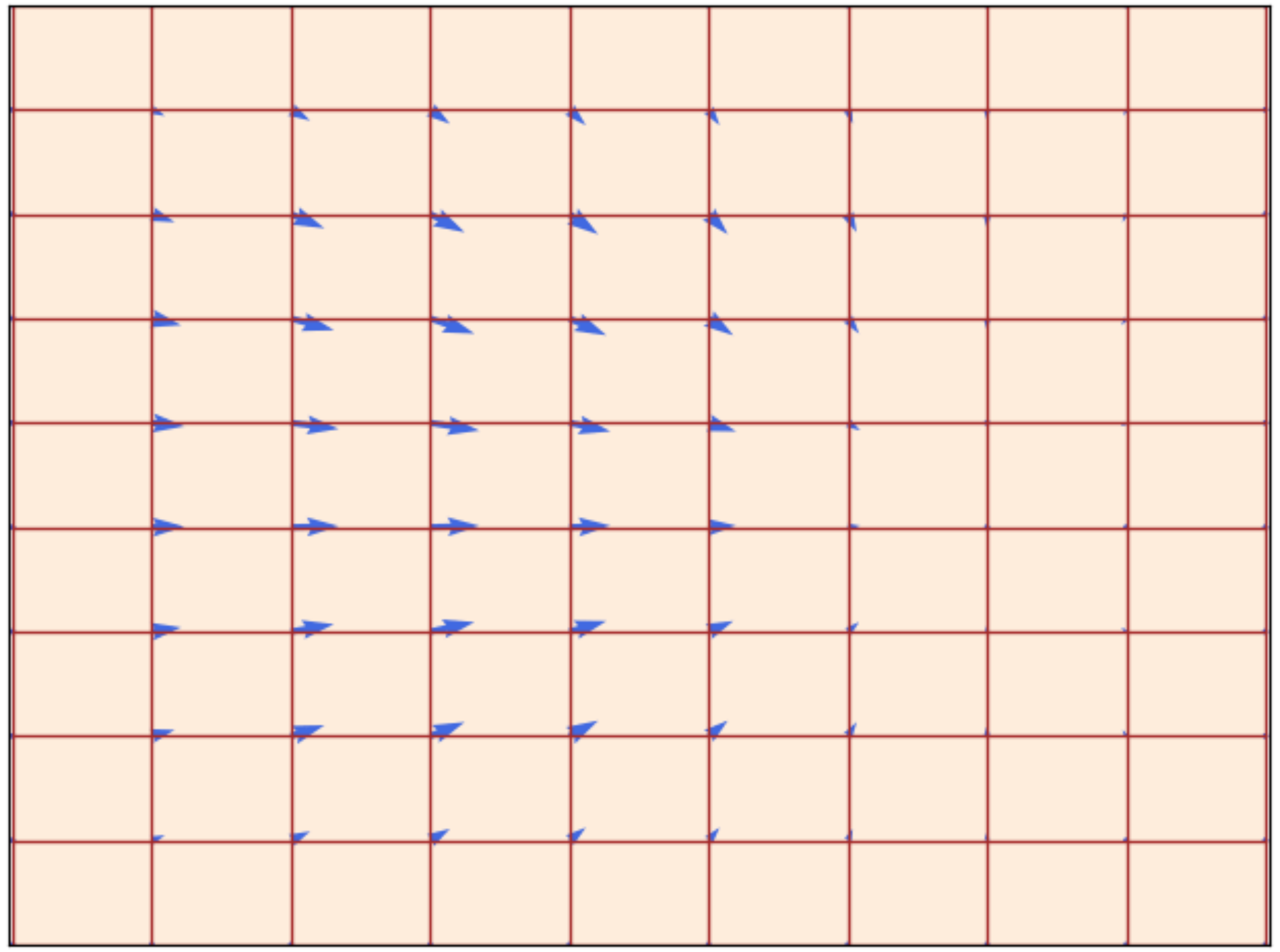
$$\psi_t(x)$$

Solve ODE

Differentiate

$$u_t(x)$$

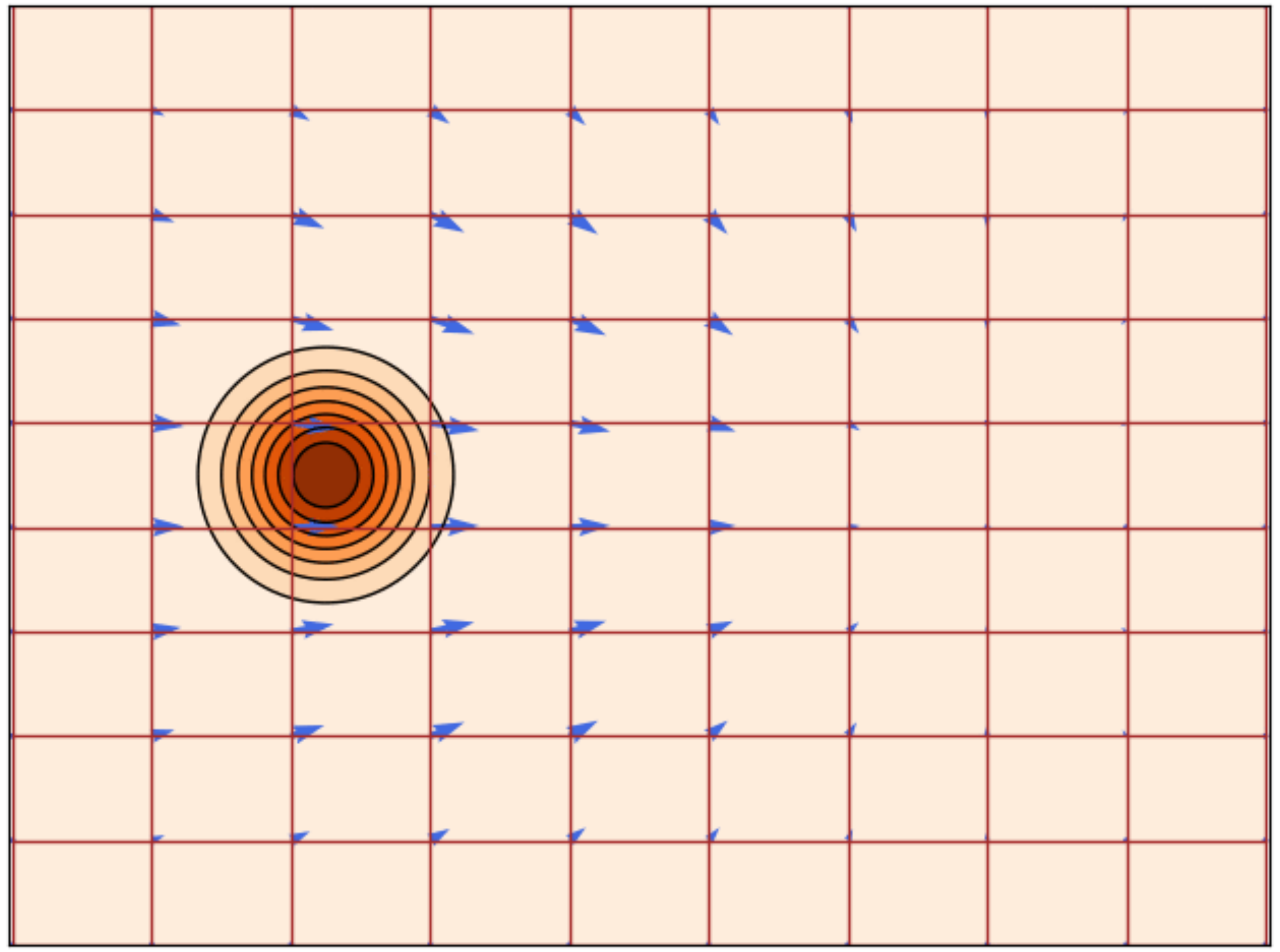
Velocity



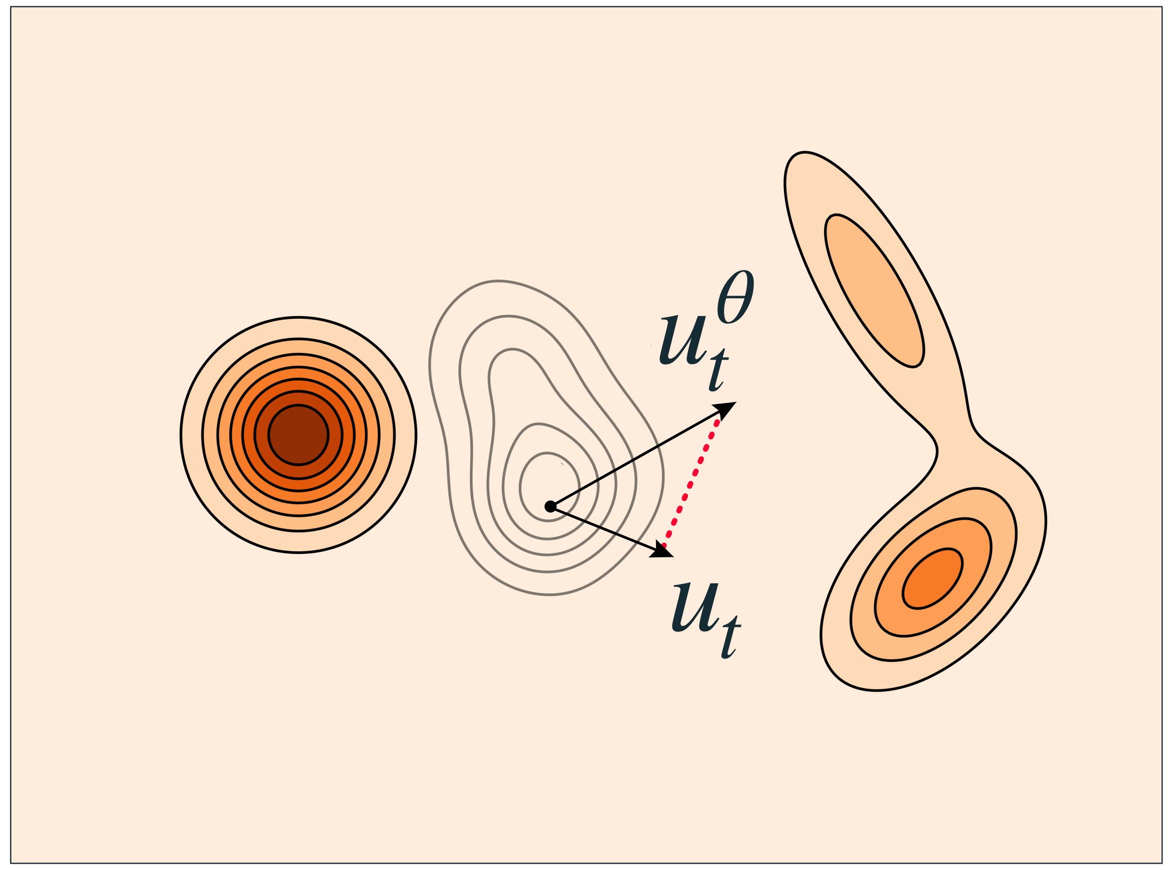
- **Pros**: velocities are **linear**
- **Cons**: simulate to sample

Velocity u_t **generates** p_t if

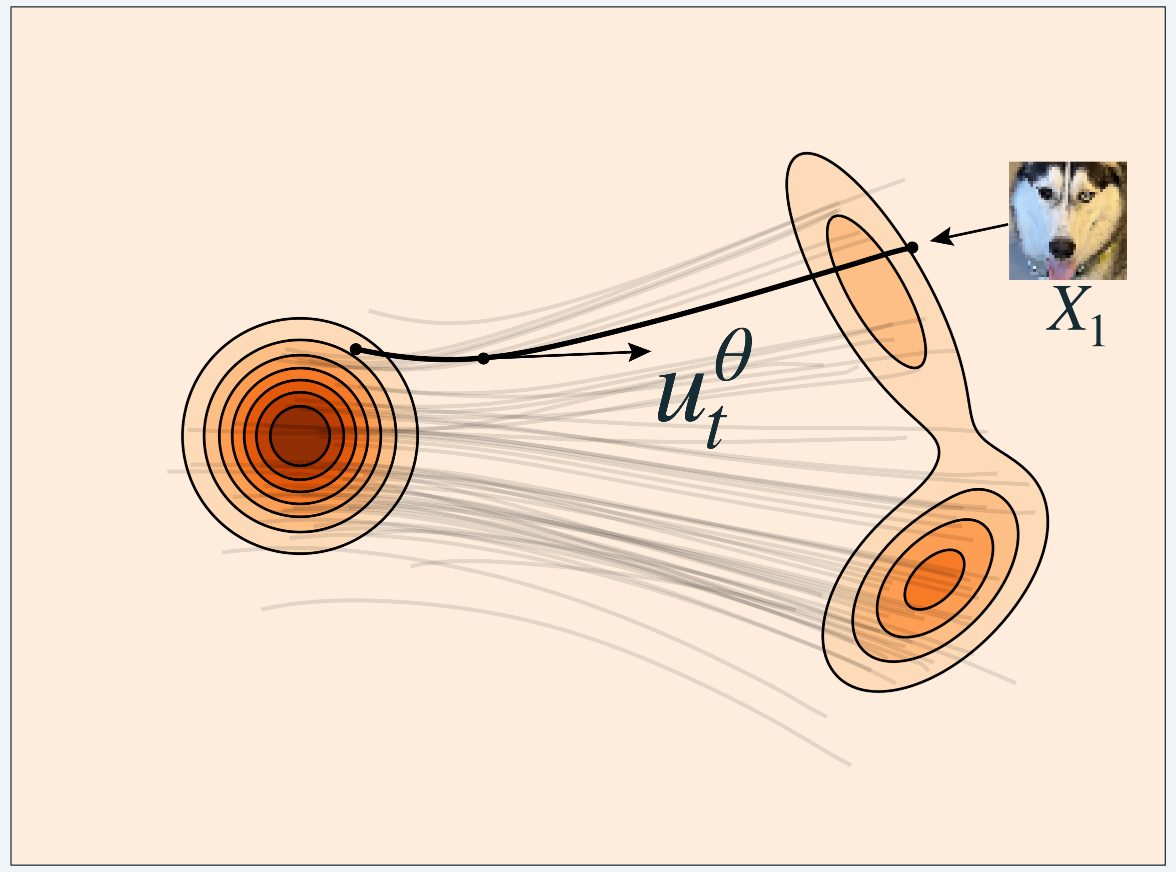
$$X_t = \psi_t(X_0) \sim p_t$$



Flow Matching



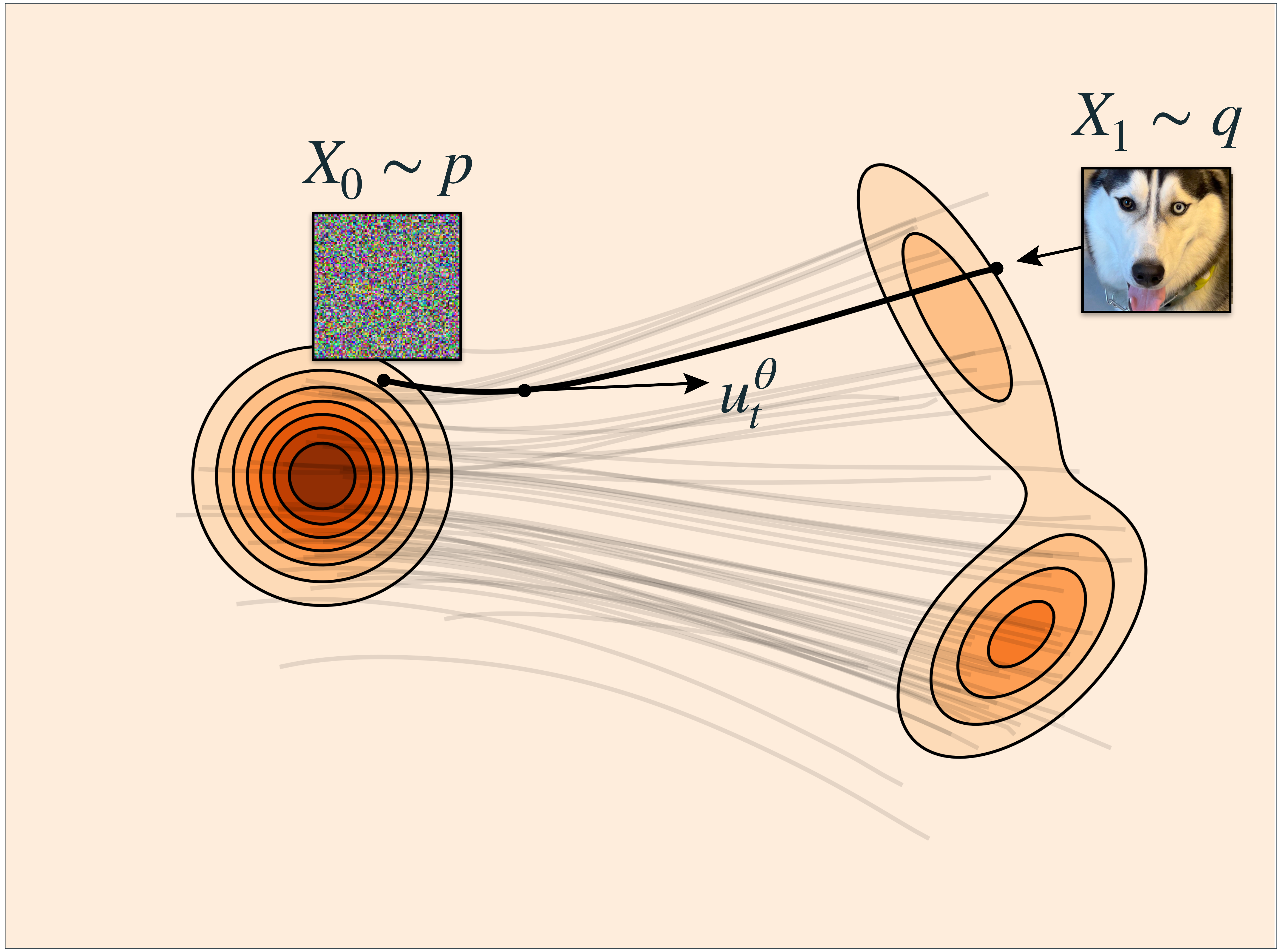
Train a velocity generating p_t with $p_0 = p$ and $p_1 = q$



Sample

from $X_0 \sim p$

Sampling a flow model

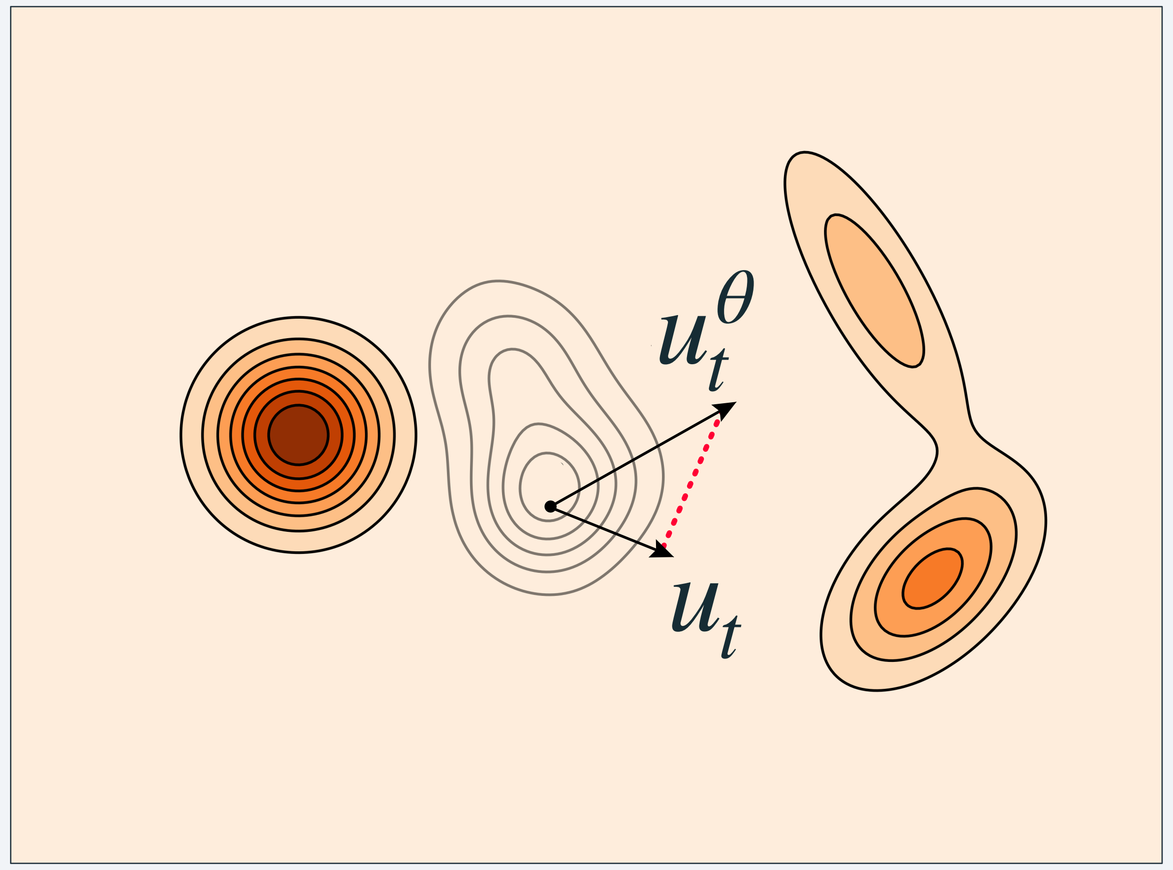


$$\frac{\mathrm{d}}{\mathrm{d}t} X_t = u_t^\theta(X_t)$$

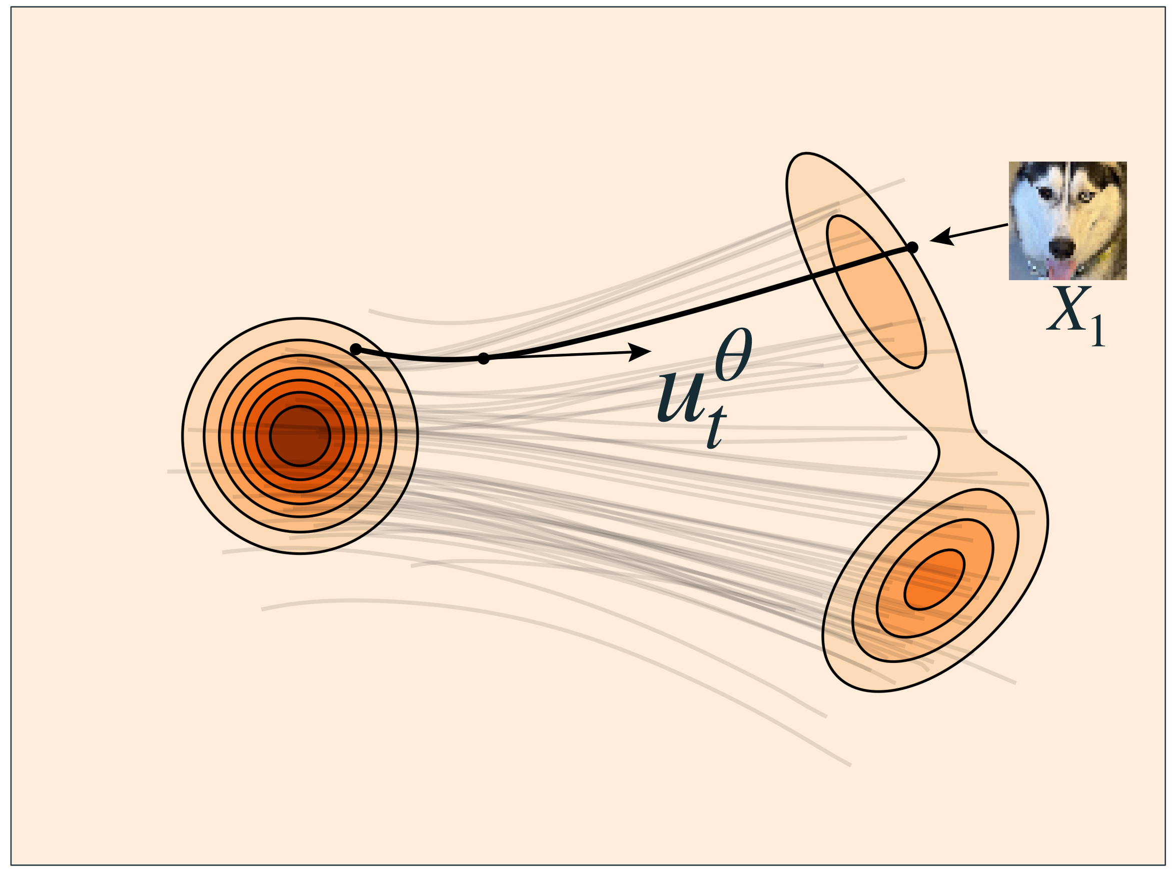
Use any ODE numerical solver.

One that works well: **Midpoint**

Flow Matching



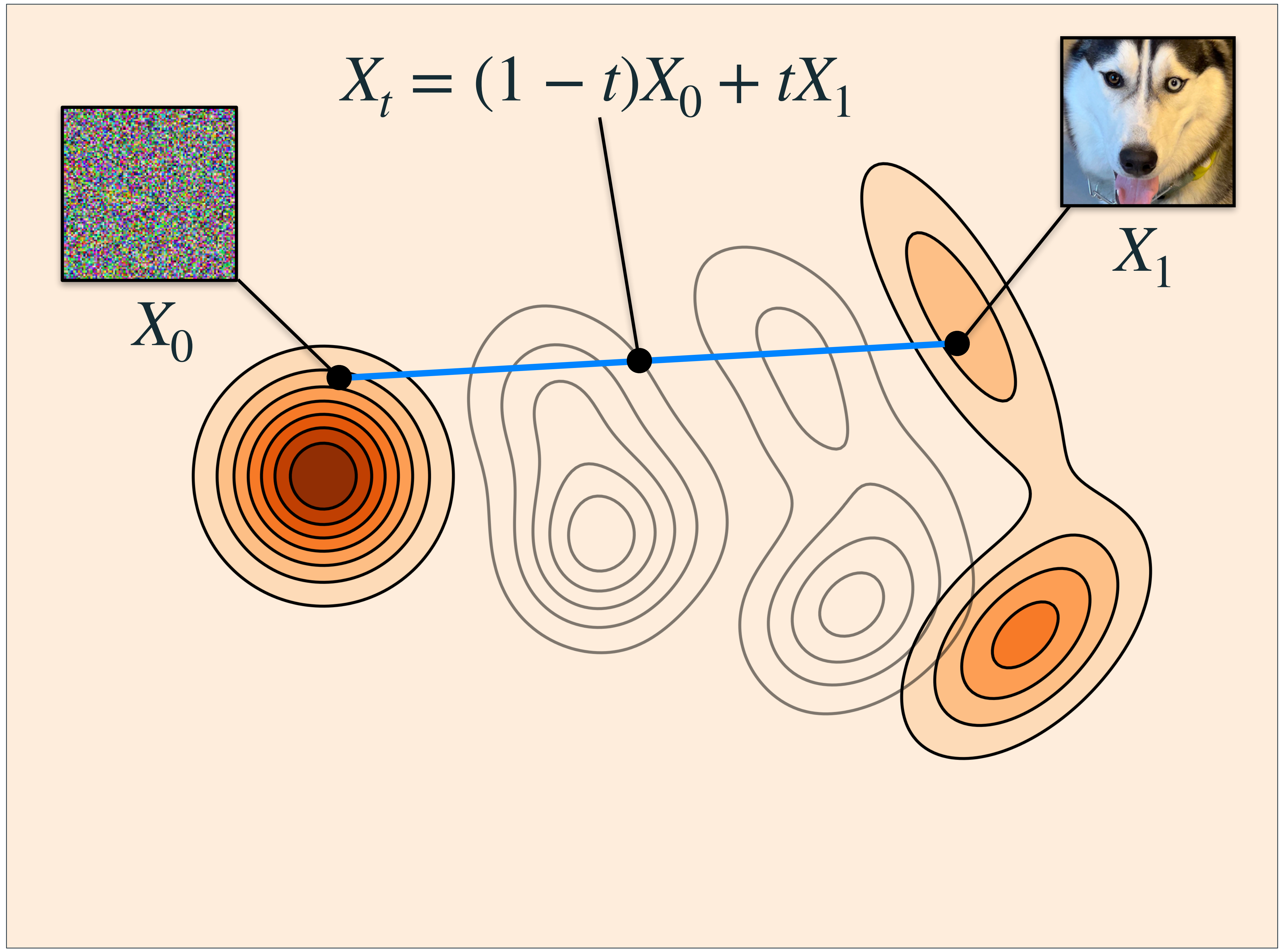
Train a velocity generating p_t with $p_0 = p$ and $p_1 = q$



Sample from $X_0 \sim p$

Simplest version of Flow Matching

```python
for _ in range(10000):
  x_1  = Tensor(make_moons(256, noise=0.05)[0])
  x_0  = torch.randn_like(x_1)
  t    = torch.rand(len(x_1), 1)
  x_t  = (1 - t) * x_0 + t * x_1
  dx_t = x_1 - x_0
  optimizer.zero_grad()
  loss_fn(flow(x_t, t), dx_t).backward()
  optimizer.step()
```



$$\mathbb{E}_{t,X_0,X_1}\left\| u_t^\theta(X_t) - (X_1 - X_0) \right\|^2$$

"Flow Matching for Generative Modeling" Lipman el al. (2022)

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

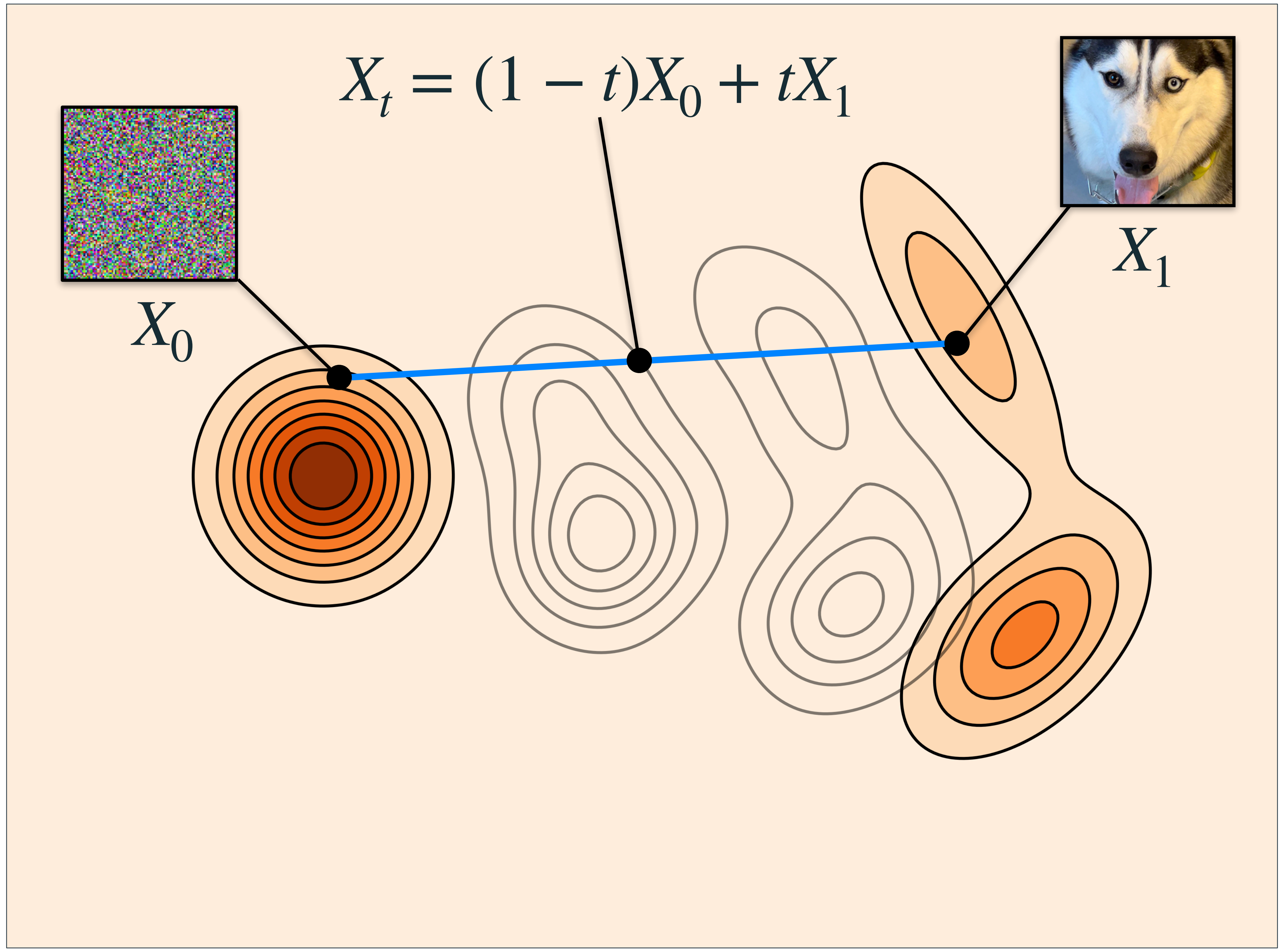
"Building Normalizing Flows with Stochastic Interpolants" Albergo et al. (2022)

Simplest version of Flow Matching

- Arbitrary $X_0 \sim p, X_1 \sim q$
- Arbitrary coupling $(X_0, X_1) \sim \pi_{0,1}$

Why does it work?

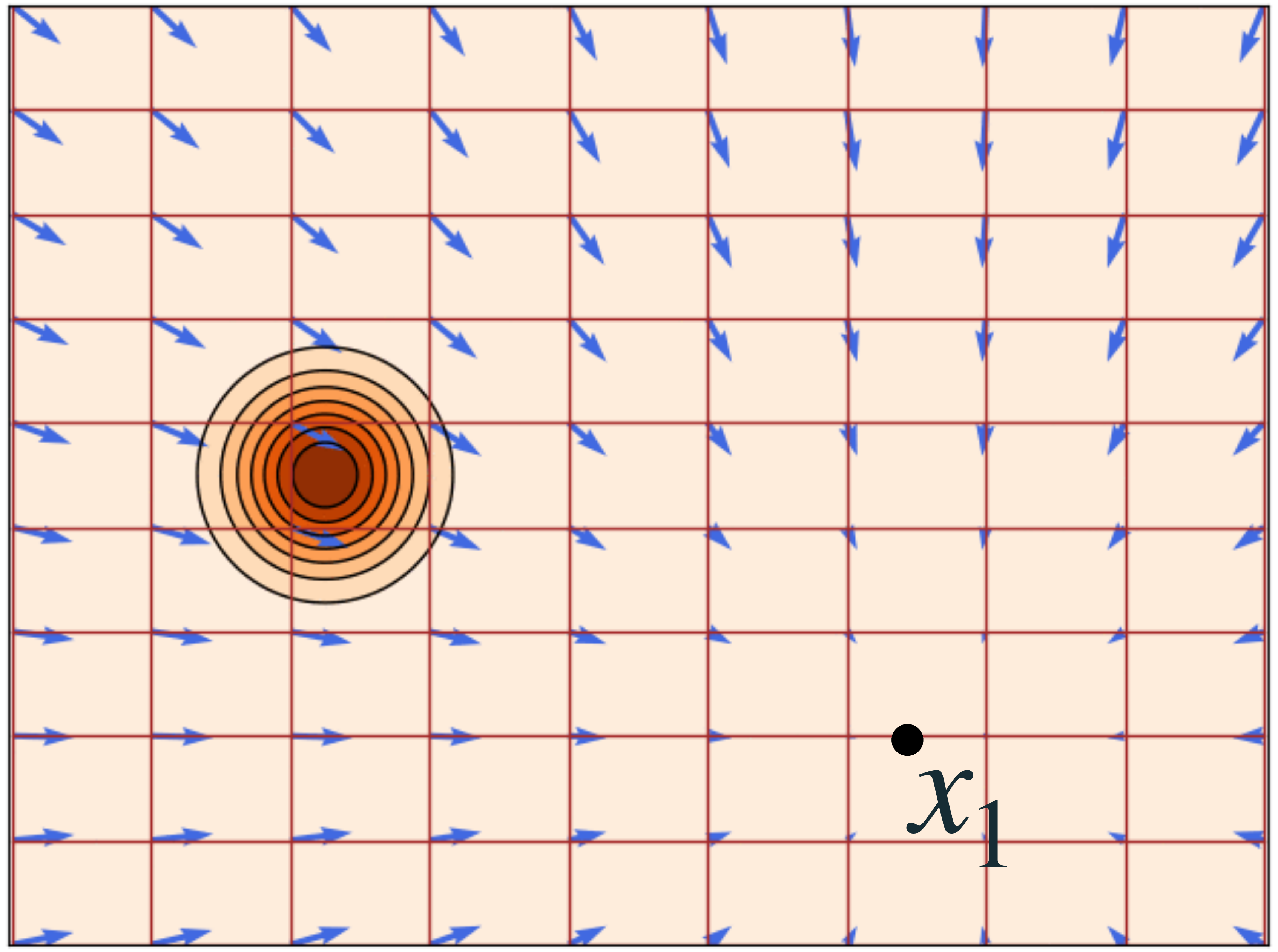
- Build flow from conditional flows
- Regress conditional flows



$$\mathbb{E}_{t,X_0,X_1}\left\| u_t^\theta(X_t) - (X_1 - X_0) \right\|^2$$

Build flow from conditional flows

Generate a single target point

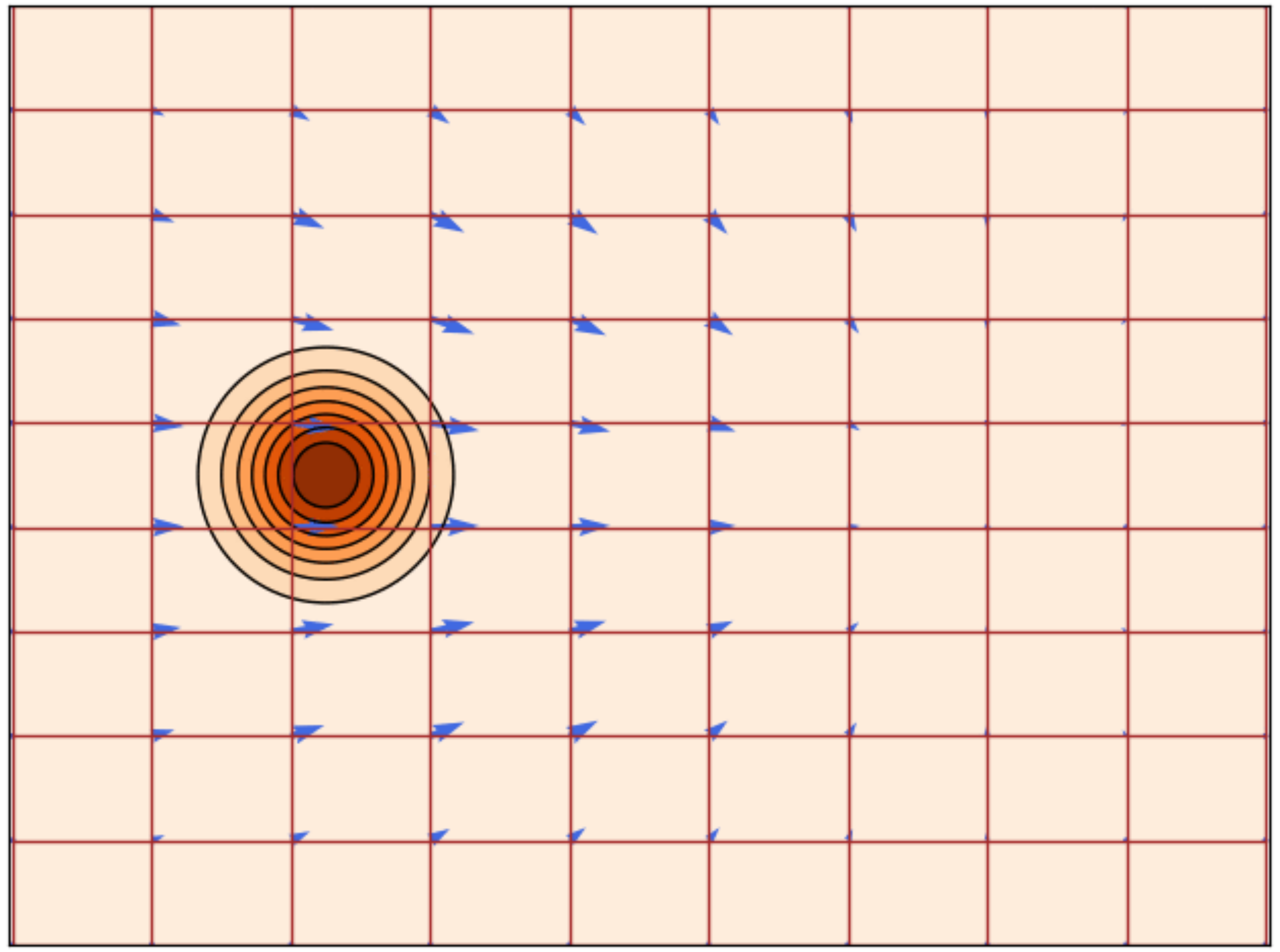


$$X_t = \psi_t(X_0 \,|\, x_1) = (1-t)X_0 + tx_1$$

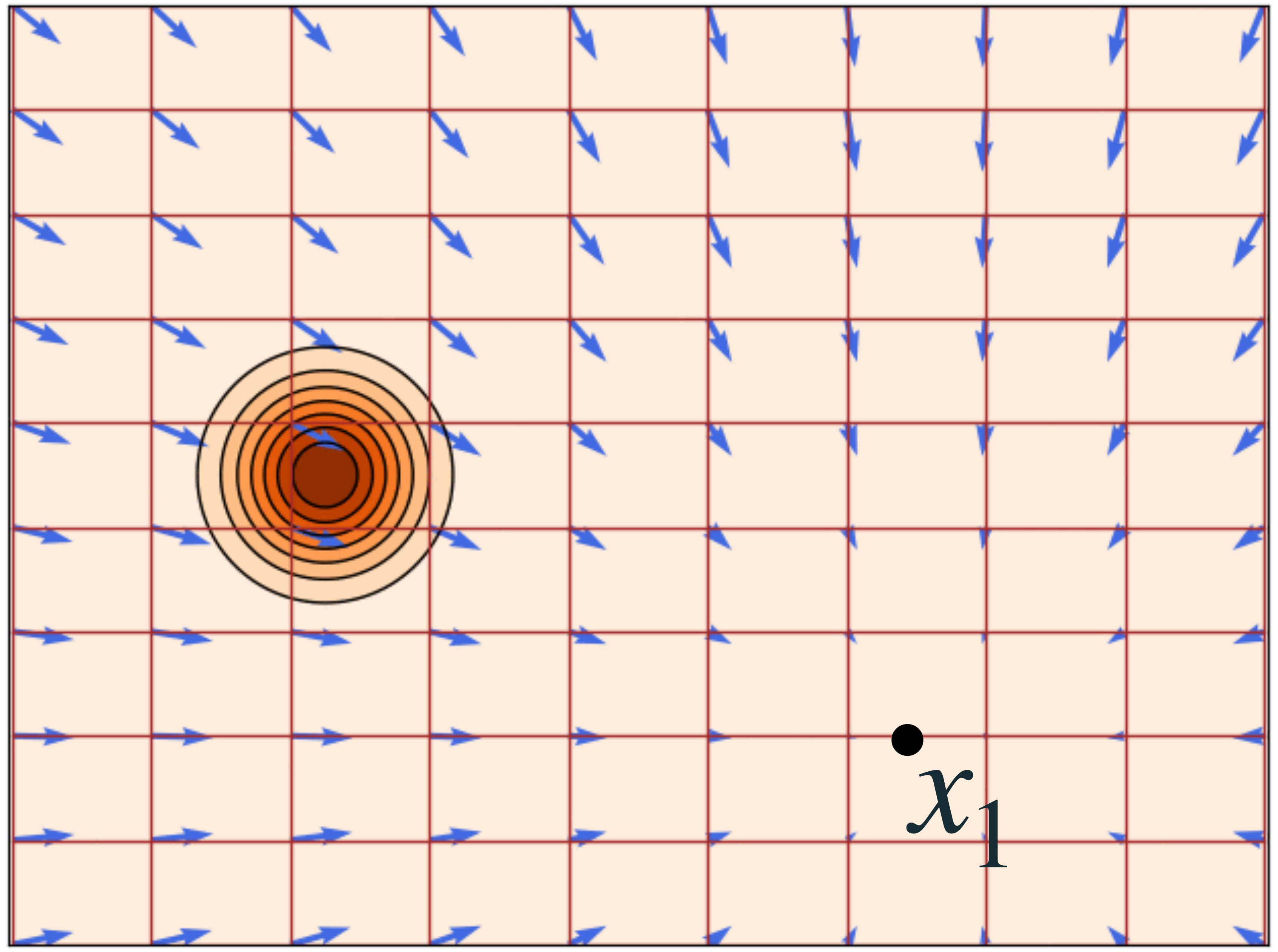
$p_{t|1}(x \,|\, x_1)$ conditional probability

$u_t(x \,|\, x_1)$ conditional velocity

Build flow from conditional flows



Generate a single target point



$$X_t = \psi_t(X_0 \,|\, x_1) = (1-t)X_0 + tx_1$$

$$p_t(x) = \mathbb{E}_{X_1} p_{t|1}(x \,|\, X_1) \longleftarrow p_{t|1}(x \,|\, x_1)$$ conditional probability

average

$$u_t(x) = \mathbb{E}\left[u_t(X_t \,|\, X_1) \,\middle|\, X_t = x\right] \longleftarrow u_t(x \,|\, x_1)$$ conditional velocity

The Marginalization Trick

Theorem*: The marginal velocity generates the marginal probability path.

$$u_t(x) = \mathbb{E}\left[u_t(X_t \,|\, X_1) \,|\, X_t = x\right] \qquad p_t(x) = \mathbb{E}_{X_1} p_{t|1}(x \,|\, X_1)$$

"Flow Matching for Generative Modeling" Lipman el al. (2022)

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

"Building Normalizing Flows with Stochastic Interpolants" Albergo et al. (2022)

Flow Matching Loss

- **Flow Matching loss:**

$$\mathscr{L}_{\mathrm{FM}}(\theta) = \mathbb{E}_{t,X_t} \left\| u_t^\theta(X_t) - u_t(X_t) \right\|^2$$

- **Conditional Flow Matching loss:**

$$\mathscr{L}_{\mathrm{CFM}}(\theta) = \mathbb{E}_{t,X_1,X_t} \left\| u_t^\theta(X_t) - u_t(X_t \mid X_1) \right\|^2$$

Theorem: Losses are equivalent,

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

Generalized Flow Matching Loss

- **Flow Matching loss:**

$$\mathscr{L}_{\text{FM}}(\theta) = \mathbb{E}_{t,X_t} \boxed{D}\left(u_t(X_t)\,,u_t^\theta(X_t)\right)$$

- **Conditional Flow Matching loss:**

$$\mathscr{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t,X_1,X_t} \boxed{D}\left(u_t(X_t\,|\,X_1)\,,u_t^\theta(X_t)\right)$$

Theorem: Losses are equivalent **iff** D is a **Bregman divergence**.

$$\nabla_\theta \mathscr{L}_{\text{FM}}(\theta) = \nabla_\theta \mathscr{L}_{\text{CFM}}(\theta)$$

"Generator Matching: Generative modeling with arbitrary Markov processes" Holderrieth et al. (2024)

Generalized Matching Loss

Theorem: Losses are equivalent **iff** D is a **Bregman divergence**.

$$\nabla_\theta \mathbb{E}_{X,Y} D(Y, g^\theta(X)) = \nabla_\theta \mathbb{E}_X D(\mathbb{E}[Y \mid X], g^\theta(X))$$

"Generator Matching: Generative modeling with arbitrary Markov processes" Holderrieth et al. (2024)

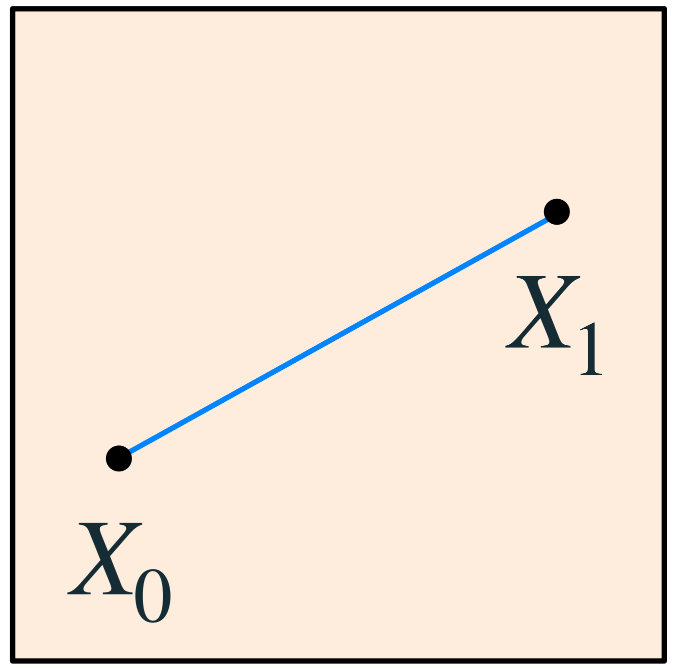
How to choose $\psi_t(x \mid x_1)$?

- **Optimal Transport** minimizes **Kinetic Energy**:

$$\int_0^1 \mathbb{E}_{X_t \sim p_t} \|u_t(X_t)\|^2 \mathrm{d}t \;\leq\; \mathbb{E}_{X_0, X_1} \int_0^1 \|\dot{\psi}_t(X_0 \mid X_1)\|^2 \mathrm{d}t$$

Linear conditional flow:

- Minimizes bound
- Reduces KE of initial coupling
- Exact OT for single data points
- <u>**Not**</u> Optimal Transport
  (but in high dim straighter)



$$\psi_t(x \mid x_1) = t x_1 + (1 - t) x$$

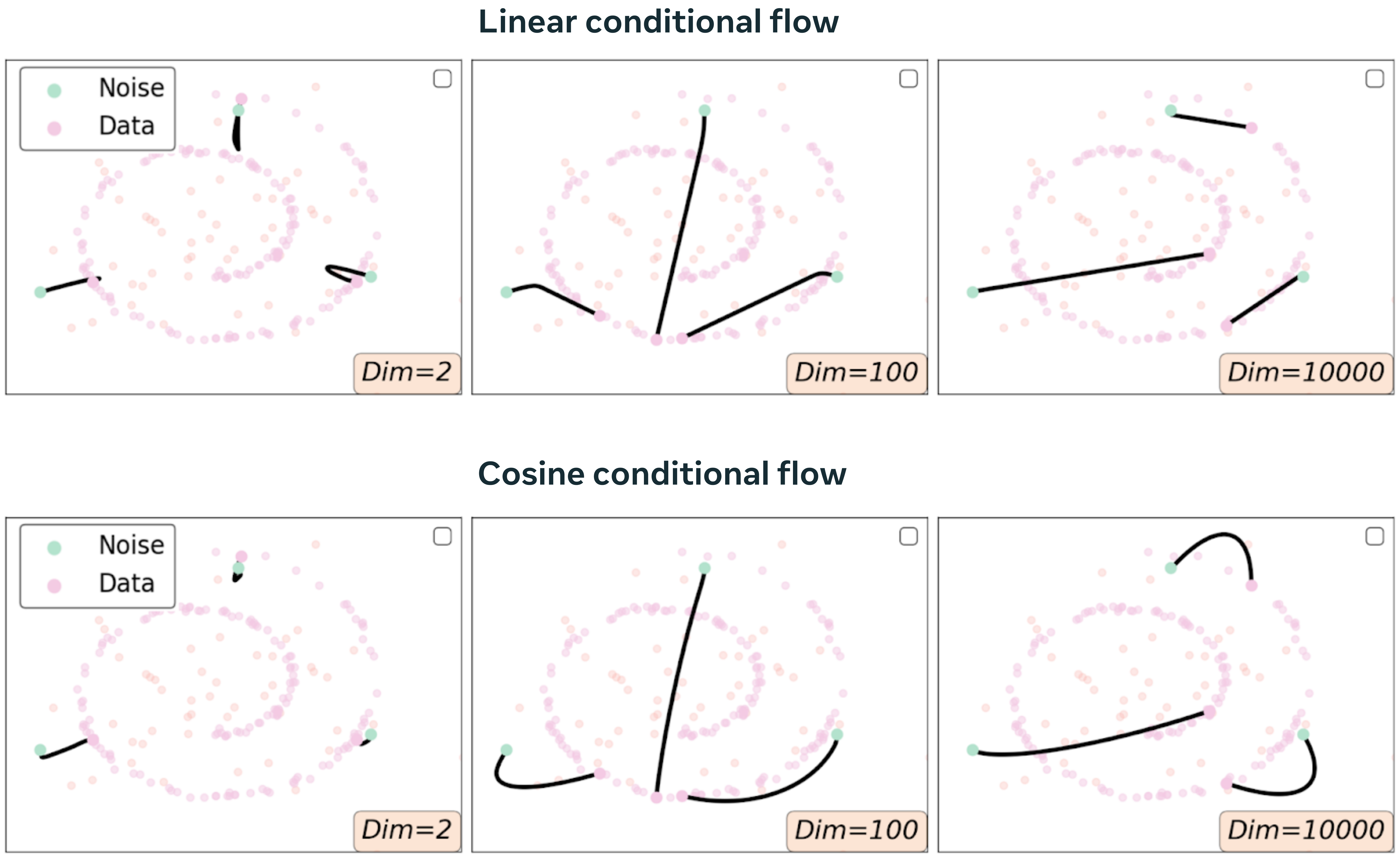
"Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow" Liu et al. (2022)
"On Kinetic Optimal Probability Paths for Generative Models" Shaul et al. (2023)

How to choose $\psi_t(x \mid x_1)$?

- Dynamic Optimal Transport minimizes Kinetic Energy:

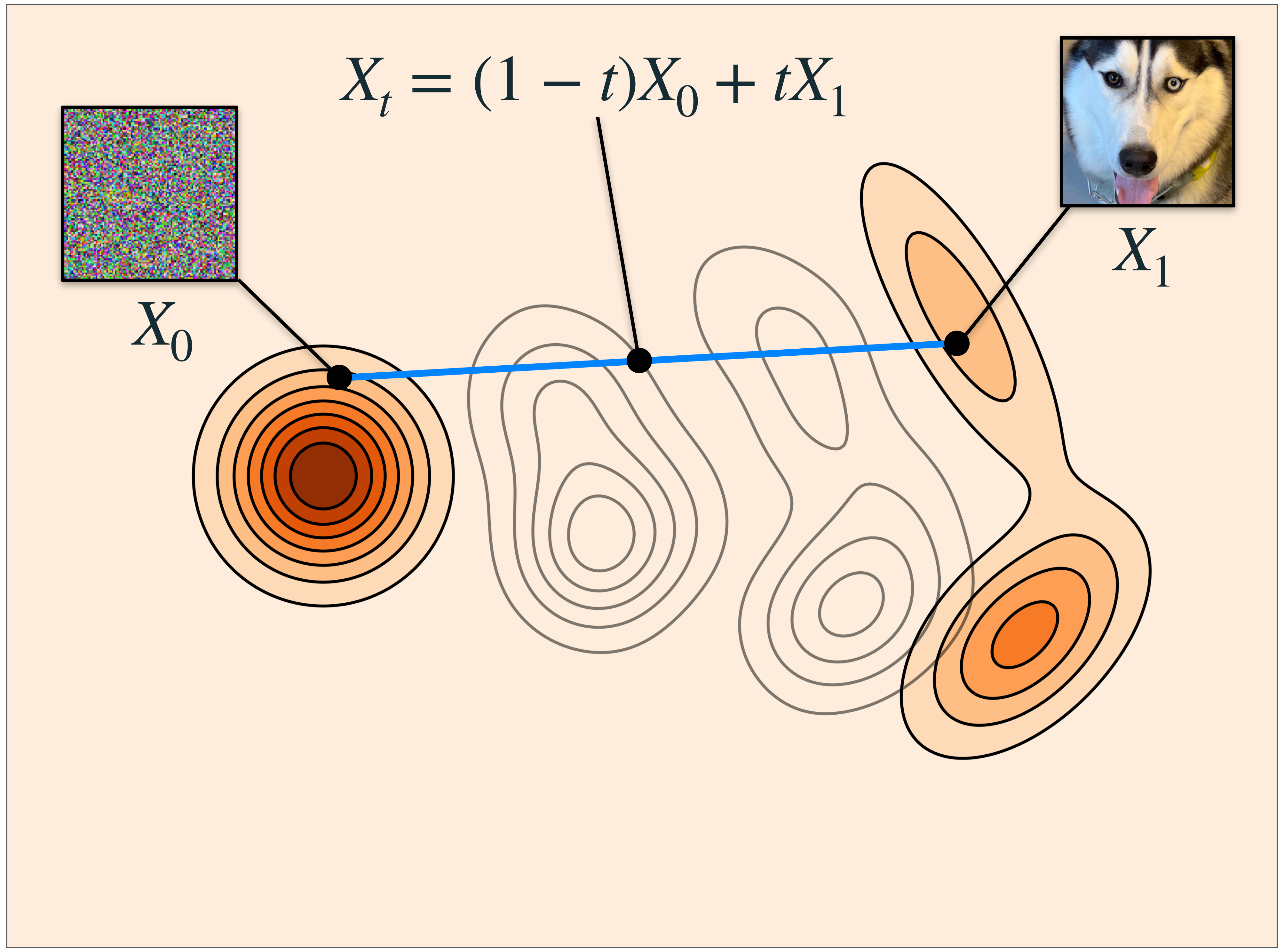
Linear conditional flow:

- Minimizes bound
- Reduces KE of initial coupling
- Exact OT for single data points
- <u>**Not**</u> Optimal Transport (but in high dim straighter)



"Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow" Liu et al. (2022)
"On Kinetic Optimal Probability Paths for Generative Models" Shaul et al. (2023)

Flow Matching with Cond-OT

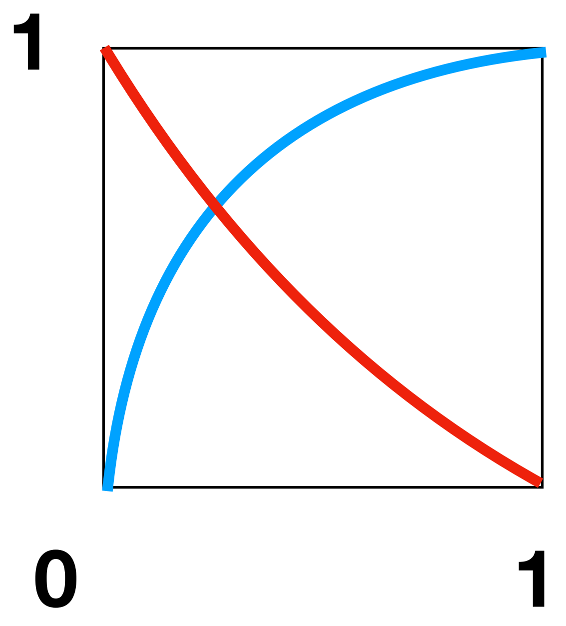


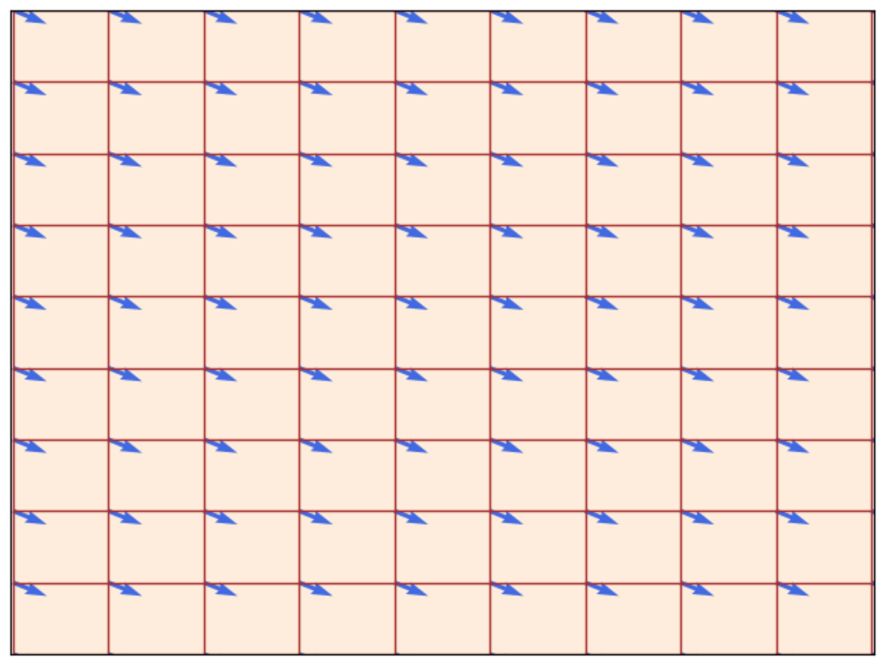
$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}\, D\Big(u_t^\theta(X_t)\,,\, u_t(X_t\,|\,X_1)\Big)$$

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}\left\| u_t^\theta\big(X_t\big) - \big(X_1 - X_0\big)\right\|^2$$

Affine paths

$$\psi_t(x \mid x_1) = \alpha_t x_1 + \sigma_t x$$





$$u_t(x) = \mathbb{E}\left[\dot{\alpha}_t X_1 + \dot{\sigma}_t X_0 \mid X_t = x\right]$$ **Velocity** prediction

Singular at $t = 0$ $$= a_t x + b_t \mathbb{E}\left[X_0 \mid X_t = x\right]$$ **Source** x_0 prediction

Singular at $t = 1$ $$= c_t \mathbb{E}\left[X_1 \mid X_t = x\right] + d_t$$ **Target** x_1 prediction

Gaussian paths

$$p(x) = \mathcal{N}(x \,|\, 0 \,, I) \qquad \pi_{0,1}(x_0, x_1) = p(x_0)q(x_1)$$

$$u_t(x) = \mathbb{E}\left[\dot{\alpha}_t X_1 + \dot{\sigma}_t X_0 \,|\, X_t = x\right]$$ **Velocity** prediction

$$= a_t x + b_t \mathbb{E}\left[X_0 \,|\, X_t = x\right]$$ **Source** x_0 prediction — ϵ **(noise) prediction**

Probability Flow ODE

$$= c_t \mathbb{E}\left[X_1 \,|\, X_t = x\right] + d_t$$ **Target** x_1 prediction — x_1 **prediction (denoiser)**

"Score-Based Generative Modeling through Stochastic Differential Equations" Song et al. 2020

Affine and Gaussian paths

B \ A	velocity	x_1-prediction	x_0-prediction	score
velocity	$0, 1$	$\frac{\dot{\sigma}_t}{\sigma_t}, \frac{\dot{\alpha}_t\sigma_t - \dot{\sigma}_t\alpha_t}{\sigma_t}$	$\frac{\dot{\alpha}_t}{\alpha_t}, \frac{\dot{\sigma}_t\alpha_t - \dot{\alpha}_t\sigma_t}{\alpha_t}$	$\frac{\dot{\alpha}_t}{\alpha_t}, -\frac{\dot{\sigma}_t\sigma_t\alpha_t - \dot{\alpha}_t\sigma_t^2}{\alpha_t}$
x_1-prediction		$0, 1$	$\frac{1}{\alpha_t}, -\frac{\sigma_t}{\alpha_t}$	$\frac{1}{\alpha_t}, \frac{\sigma_t^2}{\alpha_t}$
x_0-prediction			$0, 1$	$0, -\sigma_t$
score				$0, 1$

Affine paths

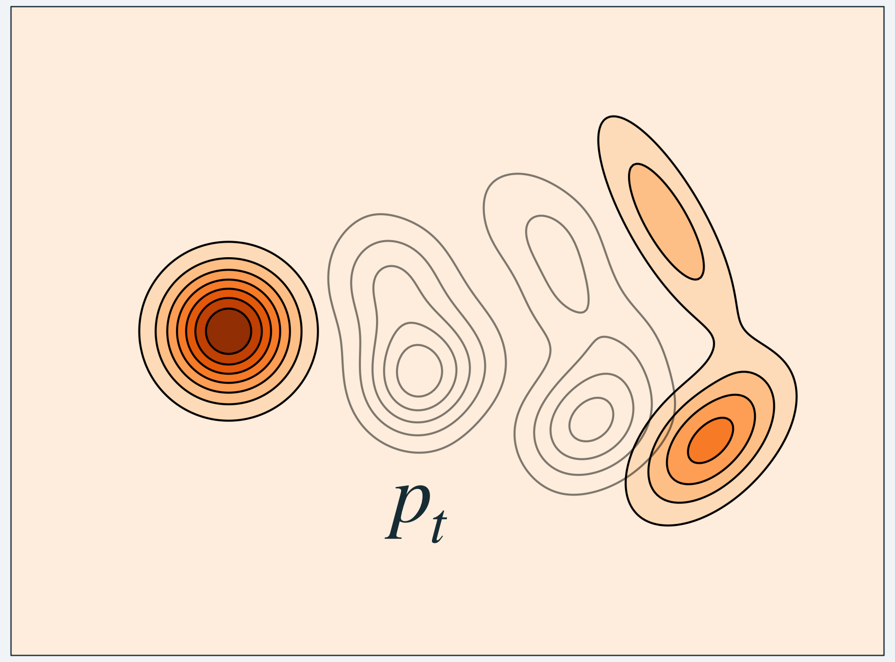
Gaussian paths

Flow Matching

Diffusion models (deterministic)

Defined by any $p_{t|1}$ we know how to generate

$X_t \sim p_t$



Defined by forward process (data $\to$ noise)

- arbitrary p, q and coupling
- $p_{t|1}$
- u_t, X_0, X_1
- u_t, cond-OT/linear $\Rightarrow$ performance, efficiency

Data

Conditional paths

Velocity Param

Gaussian affine paths

- $p = \mathcal{N}$, independent coupling
- $\mathcal{N}(\alpha_t x_1, \sigma_t^2 I)$ singularity
- ϵ, X_1, v

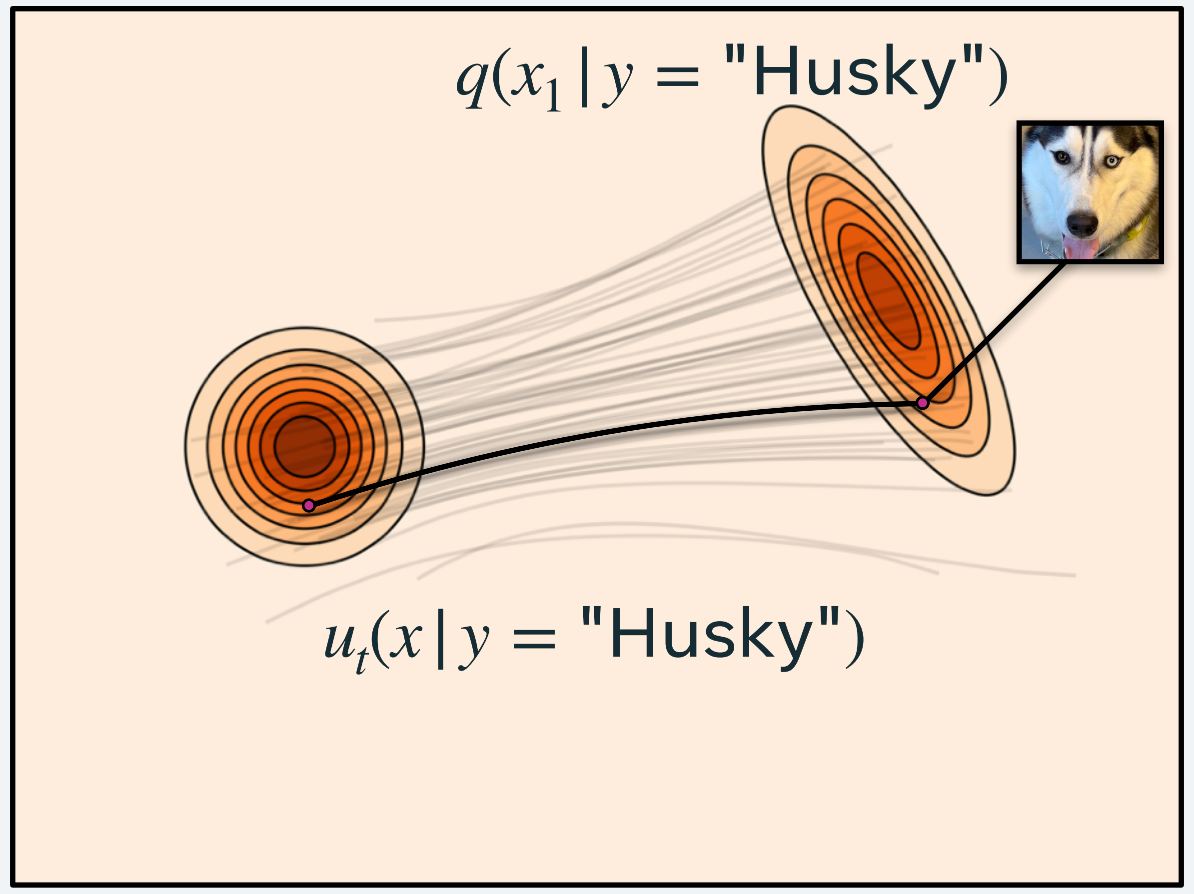
CTMP

Process

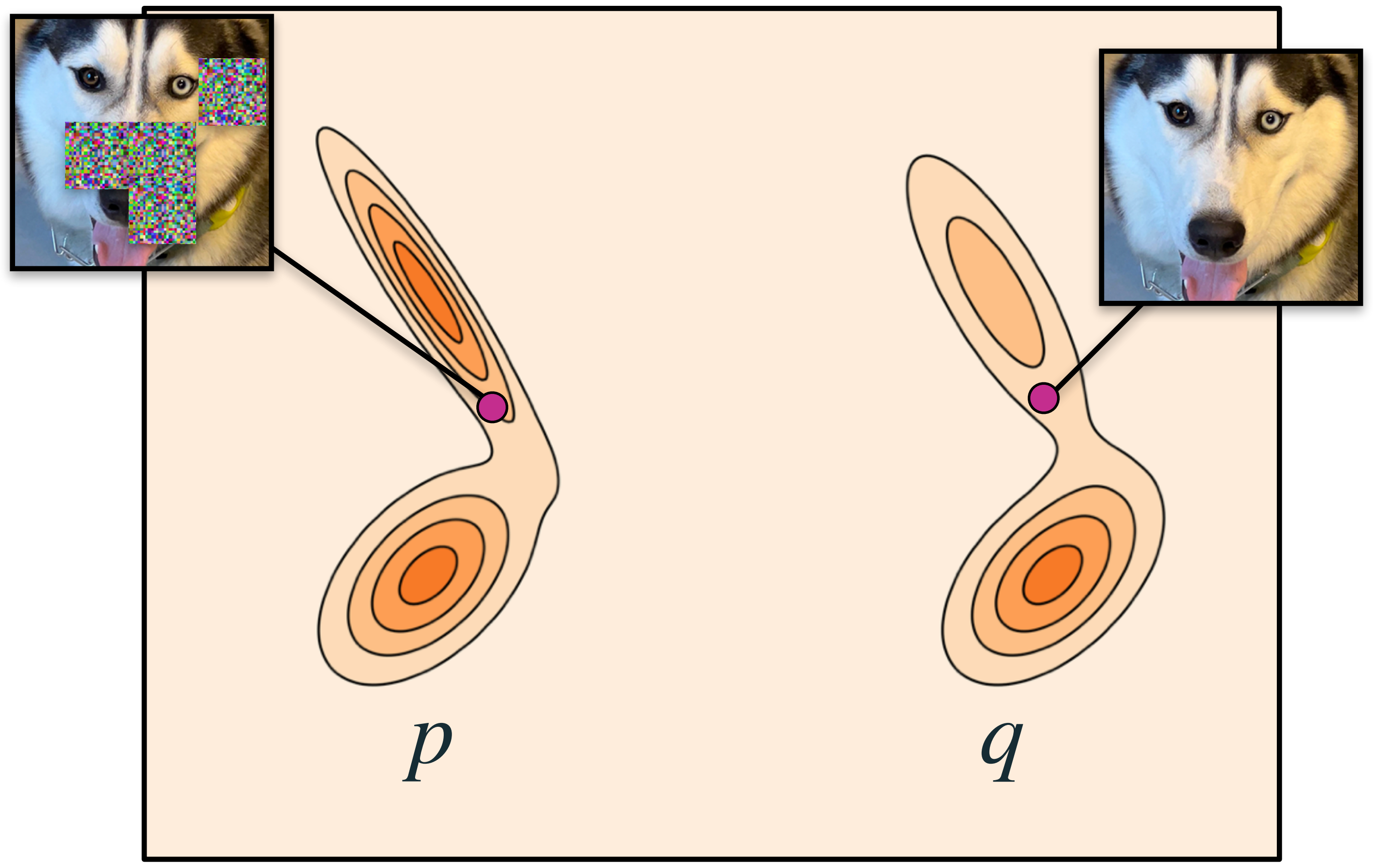
Flow, Diffusion, CTMC

02 Flow Matching Advanced Designs

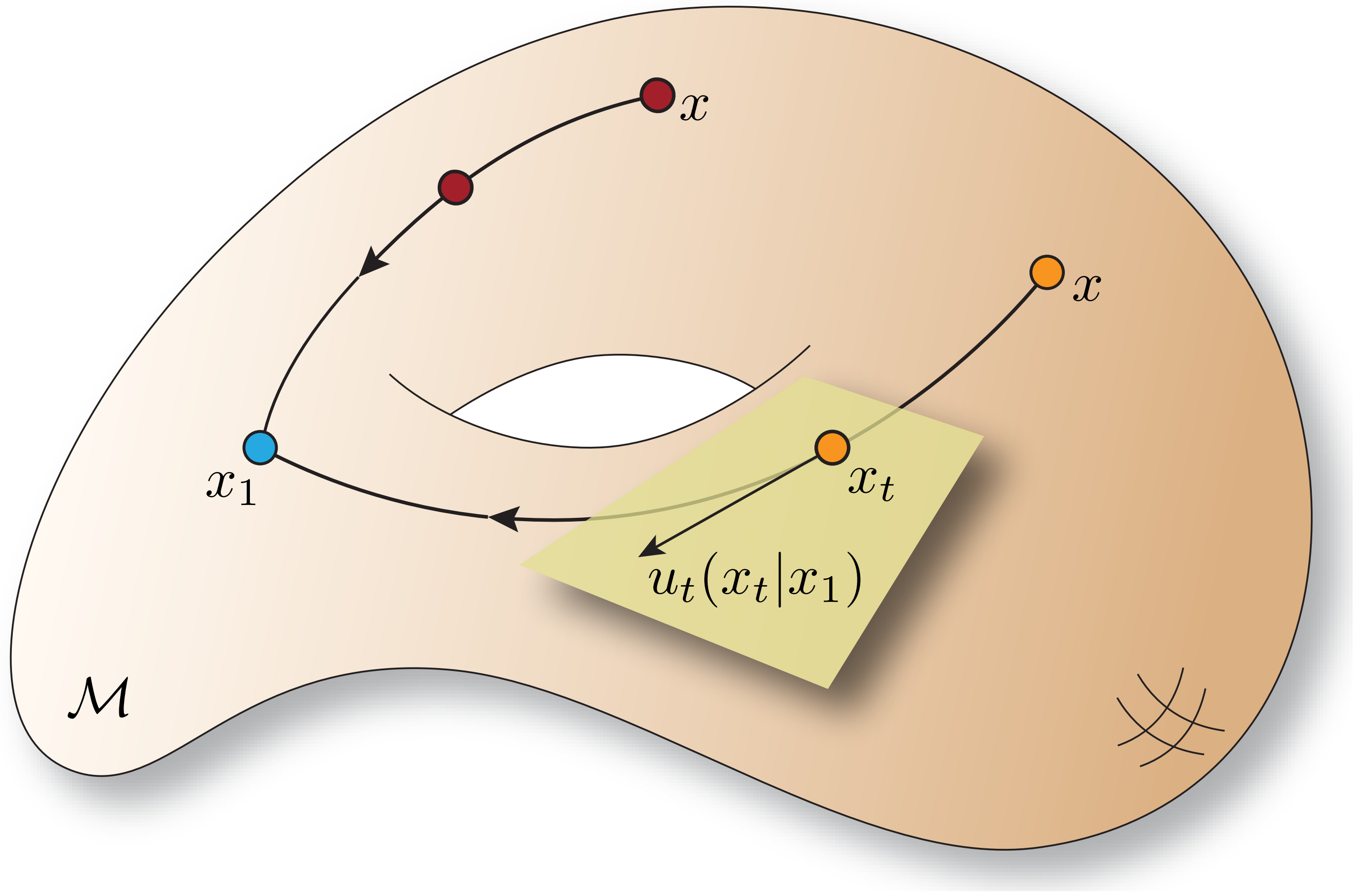
Conditioning and Guidance



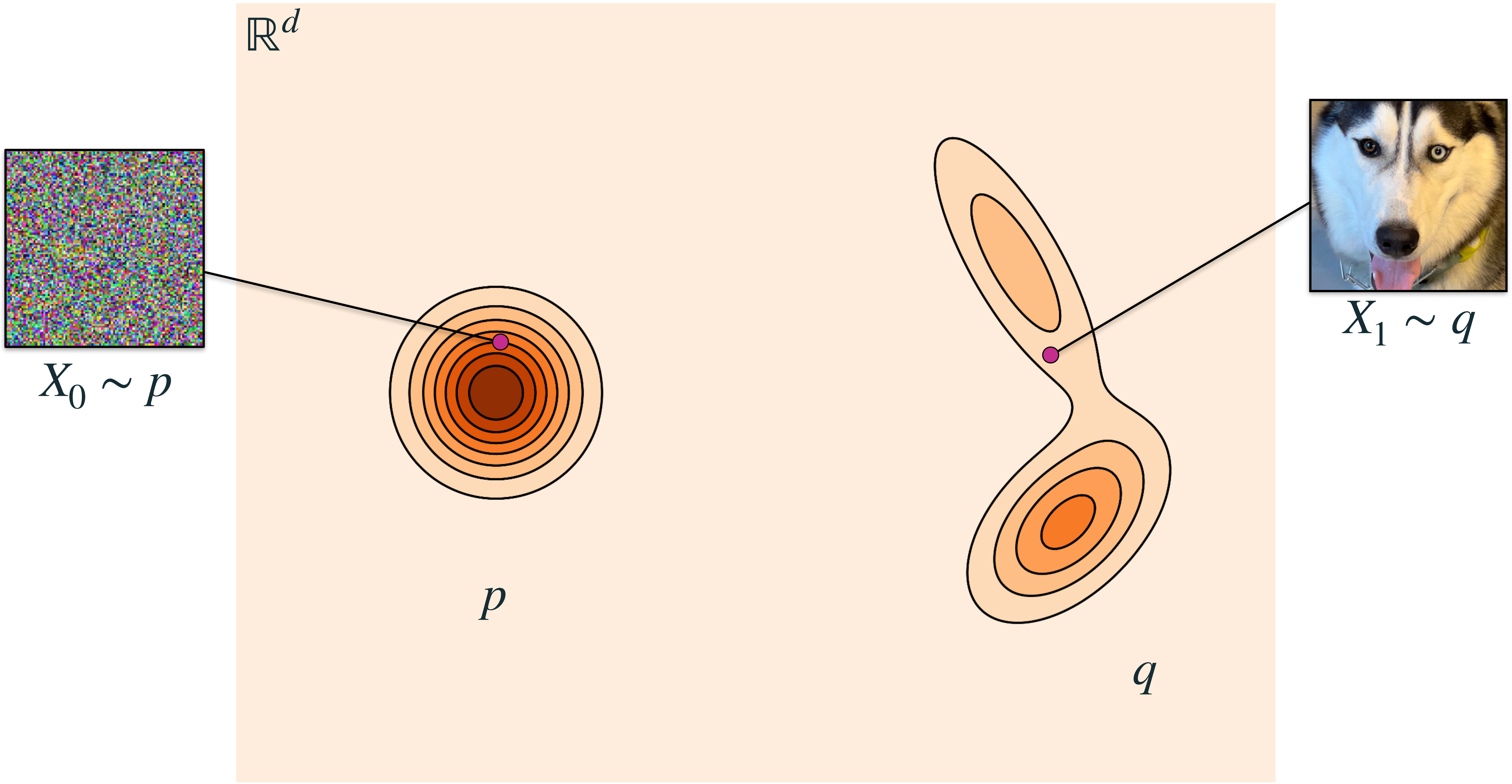
Data Couplings



Geometric Flow Matching

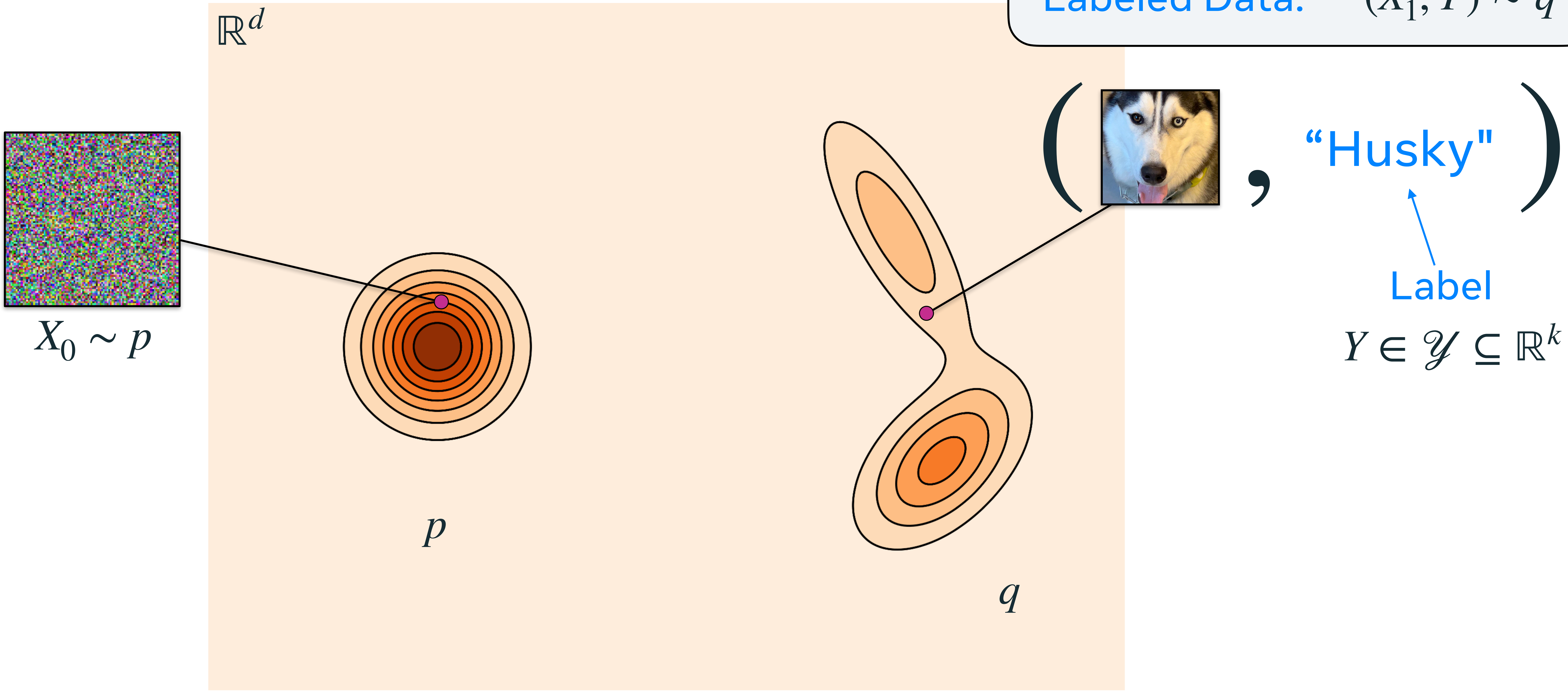


Conditioning and Guidance



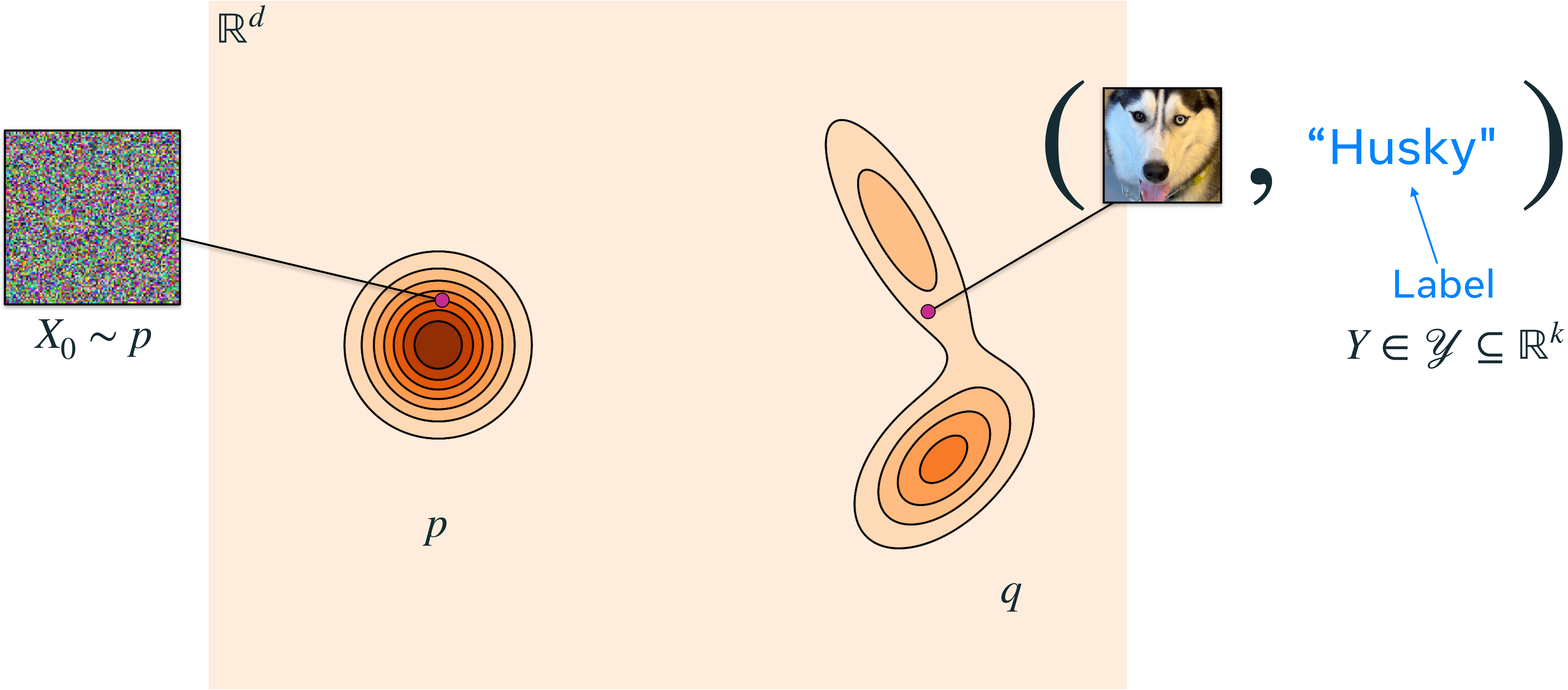
Conditioning and Guidance

Labeled Data: $(X_1, Y) \sim q$



Conditioning and Guidance

Goal: learn $q(x_1 \mid y)$

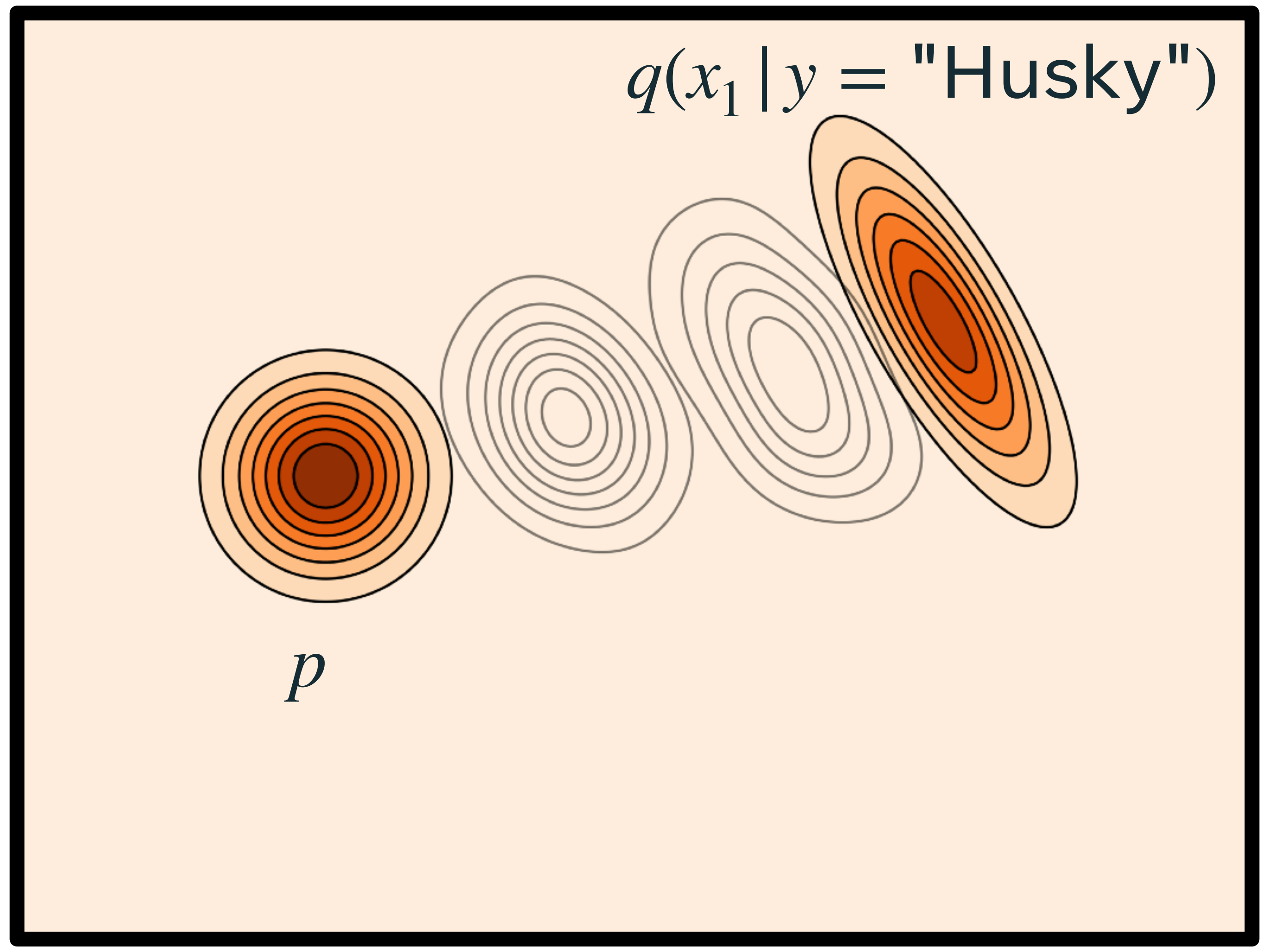


Conditional Models

$$p_{t,1|Y}(x, x_1 \mid y) = p_{t|1}(x \mid x_1) q(x_1 \mid y)$$

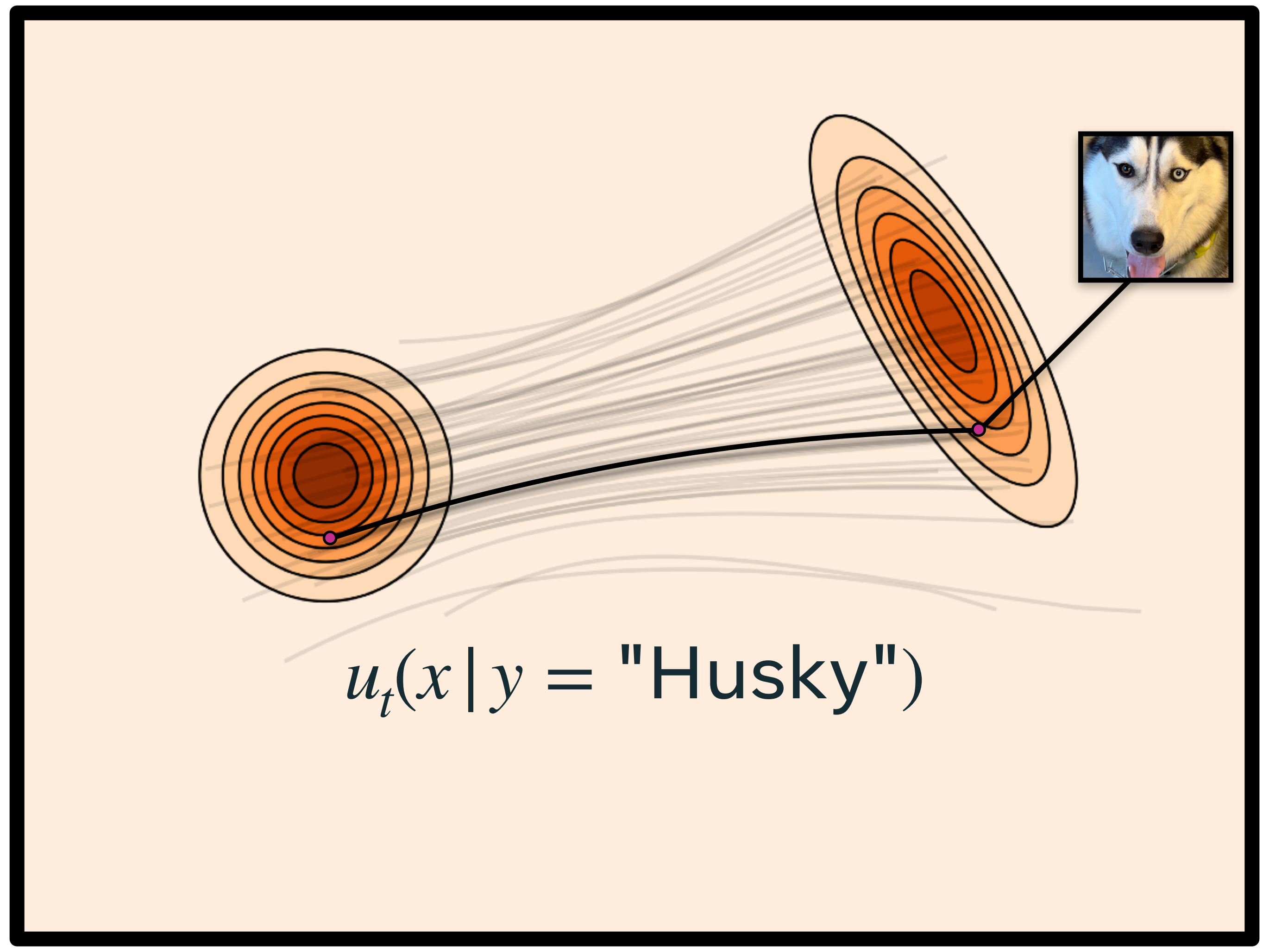
Marginal probability path

$$p_{t|Y}(x \mid y) = \mathbb{E}_{X_1}\left[p_{t|1}(x \mid X_1) \mid Y = y\right]$$



Marginal velocity

$$u_t(x \mid y) = \mathbb{E}\left[u_t(X_t \mid X_1) \,\middle|\, X_t = x, Y = y\right]$$

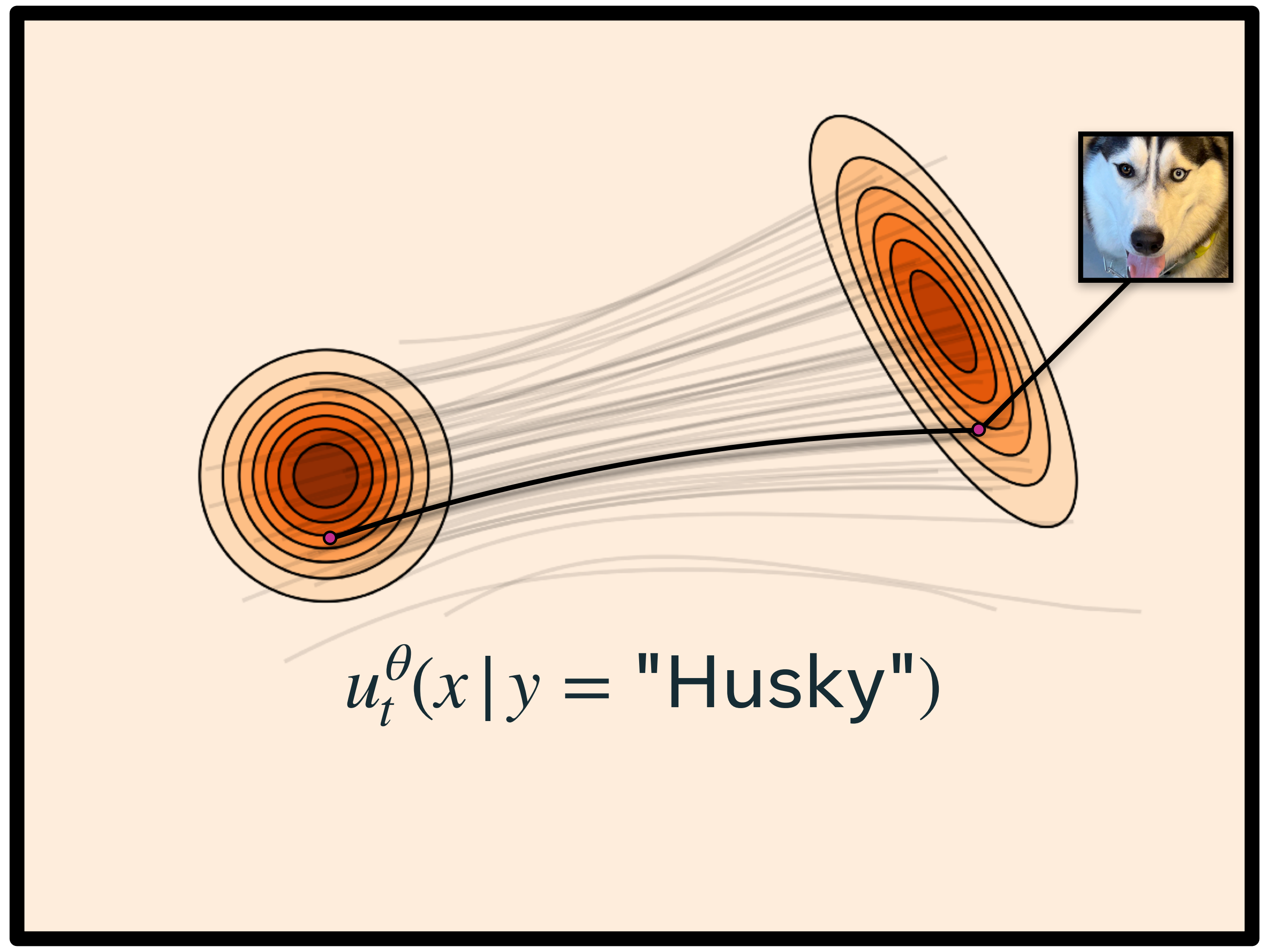


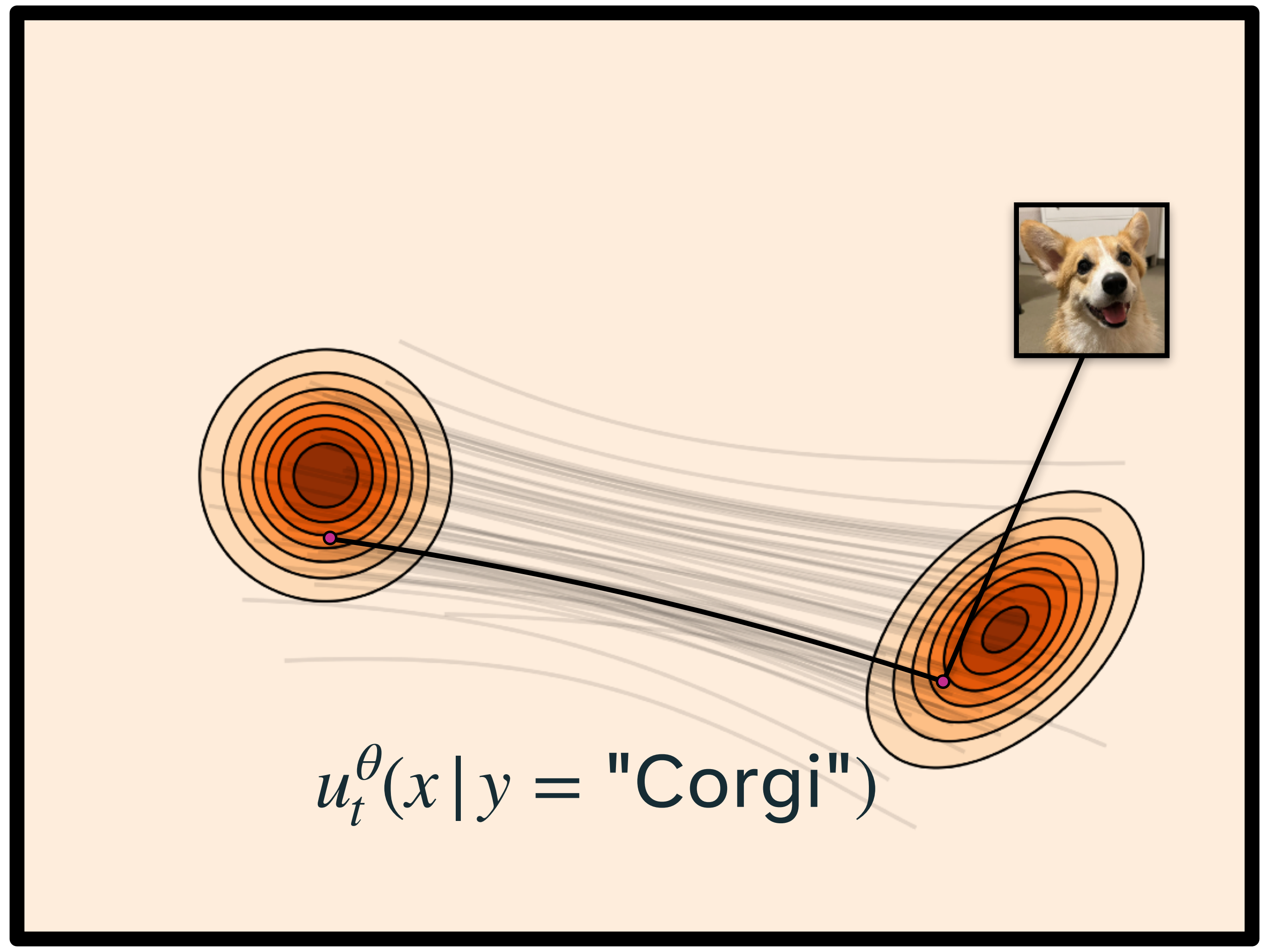
Conditional Models

Train same neural network on all conditions:

$$\mathscr{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t,(X_0,X_1,Y)\sim\pi_{0,1,Y}} \| u_t(X_t \,|\, X_1) - u_t^\theta(X_t \,|\, Y) \|^2$$

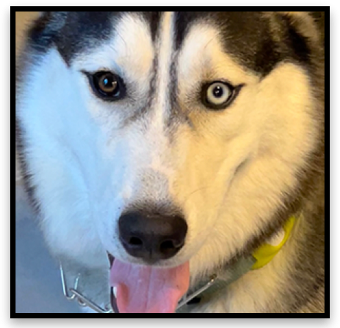
where $\quad u_t^\theta(x \,|\, y) : [0,1] \times \mathbb{R}^d \times \mathbb{R}^k \to \mathbb{R}^d$





Conditional Models - Examples

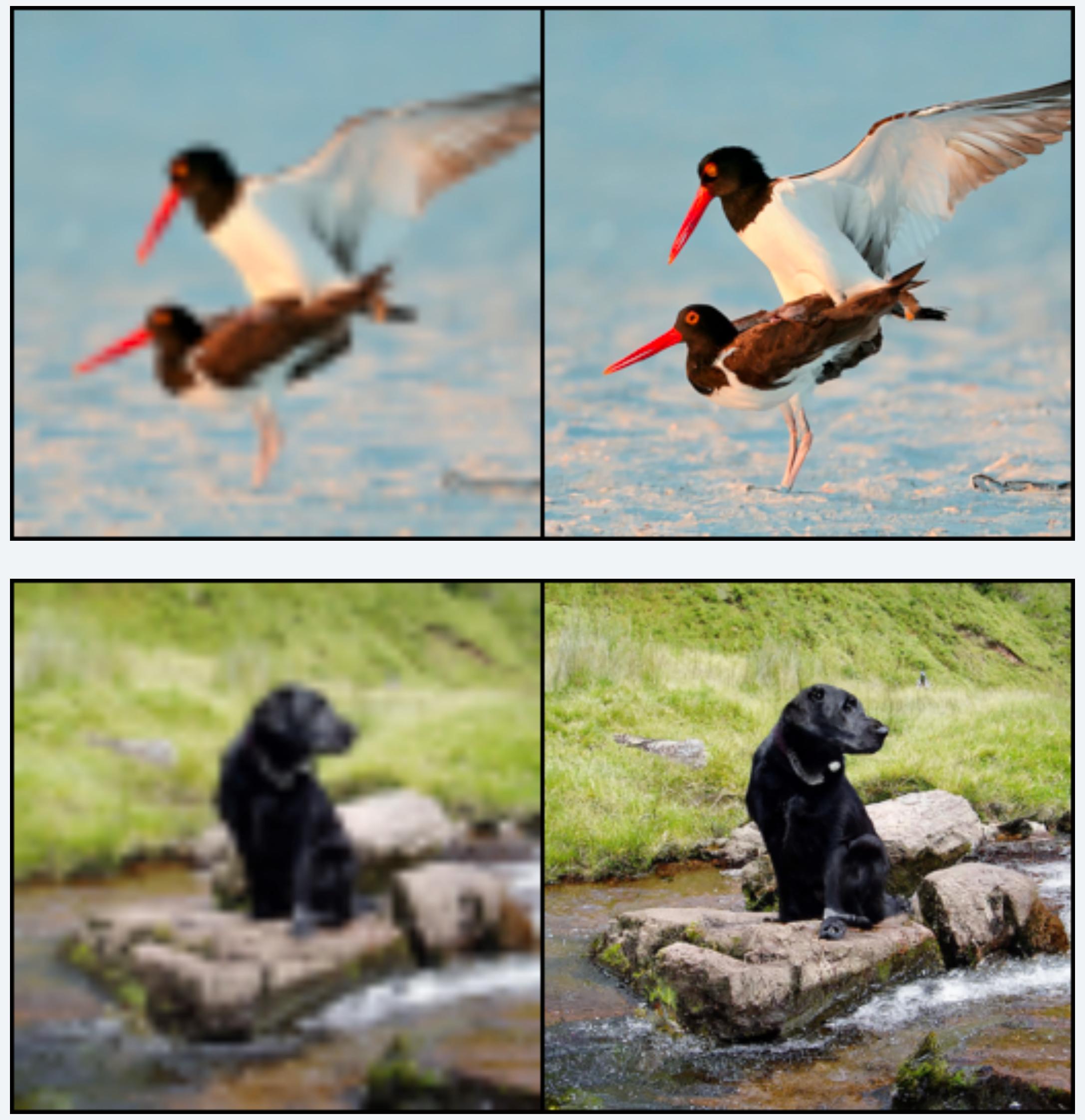
Class Conditioning

( , “Husky")

Inverse Problems

Super-resolution

$64 \times 64 \rightarrow 256 \times 256$



[Lipman et al. ’22]

Text-2-Image

“A cozy living room with a painting of a corgi on the wall above a
couch and a round coffee table in front of a couch and a vase of flowers on a coffee table”

[Nichol et al. ’22]

“Flow Matching for Generative Modeling” Lipman et al. (2022)
“GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models” Nichol et al. (2021)

Guidance for Score Matching

Guide unconditional model with classifier to sample from conditional distribution

Classifier Guidance $\nabla_x \log \tilde{p}_{t|Y}(x|y) = \nabla_x \log p_t(x) + w \nabla_x \log p_{Y|t}(y|x)$

Classifier-Free Guidance $\nabla_x \log \tilde{p}_{t|Y}(x|y) = (1-w)\nabla_x \log p_t(x) + w \nabla_x \log p_{t|Y}(x|y)$

Guidance for Flow Matching

Assume a velocity field trained with Gaussian paths.

Classifier Guidance

$$\tilde{u}_t(x|y) = u_t(x) + w b_t \nabla_x \log p_{Y|t}(y|x)$$

Classifier-Free Guidance

$$\tilde{u}_t(x|y) = (1-w)u_t(x) + w u_t(x|y)$$

Guidance for Flow Matching

Assume a velocity field trained with Gaussian paths.

Flow Matching with Classifier-Free guidance:

"Guided Flows for Generative Modeling and Decision Making" Zheng et al. (2023)

"Mosaic-SDF for 3D Generative Models" Yariv et al. (2023)

"Audiobox: Unified Audio Generation with Natural Language Prompts" Vyas et al. (2023)

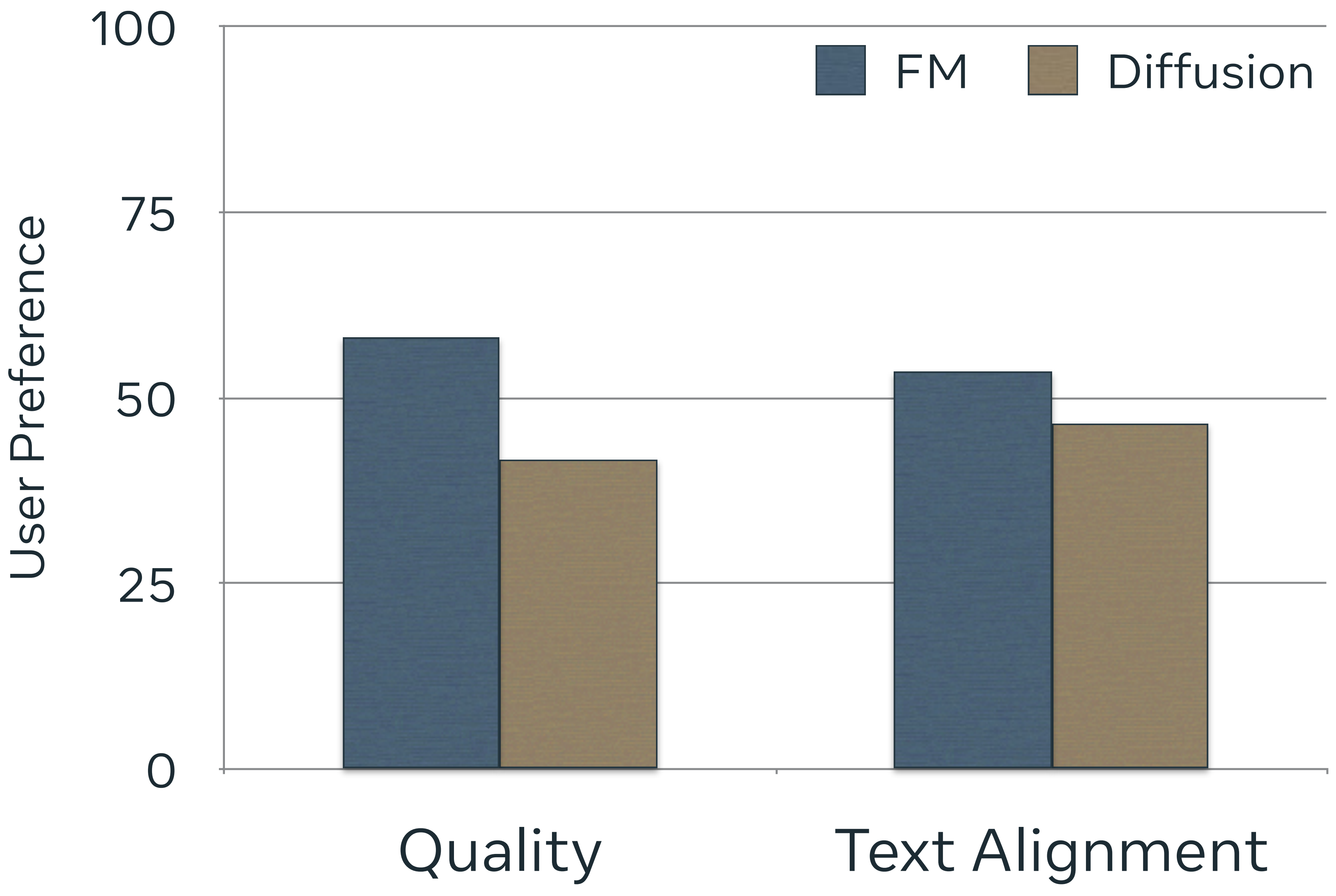
"Scaling Rectified Flow Transformers for High-Resolution Image Synthesis" Esser et al. (2024)

"Movie Gen: A Cast of Media Foundation Models" Polyak et al. (2024)

Guidance for Flow Matching

"Movie Gen: A Cast of Media Foundation Models" Polyak et al. (2024)

Ablation: text-2-video generation
FM vs. Diffusion [Polyak et al. '24]



A girl is running across a beach and holding a kite. She's wearing jean shorts and a yellow t-shirt. The sun is shining down.

A sloth with pink sunglasses lays on a donut float in a pool. The sloth is holding a tropical drink. The world is tropical. The sunlight casts a shadow.

Guidance for Flow Matching

Assume a velocity field trained with Gaussian paths.

Flow Matching with Classifier-Free guidance:

“Guided Flows for Generative Modeling and Decision Making ” Zheng et al. (2023)

"Mosaic-SDF for 3D Generative Models" Yariv et al. (2023)

"Audiobox: Unified Audio Generation with Natural Language Prompts" Vyas et al. (2023)

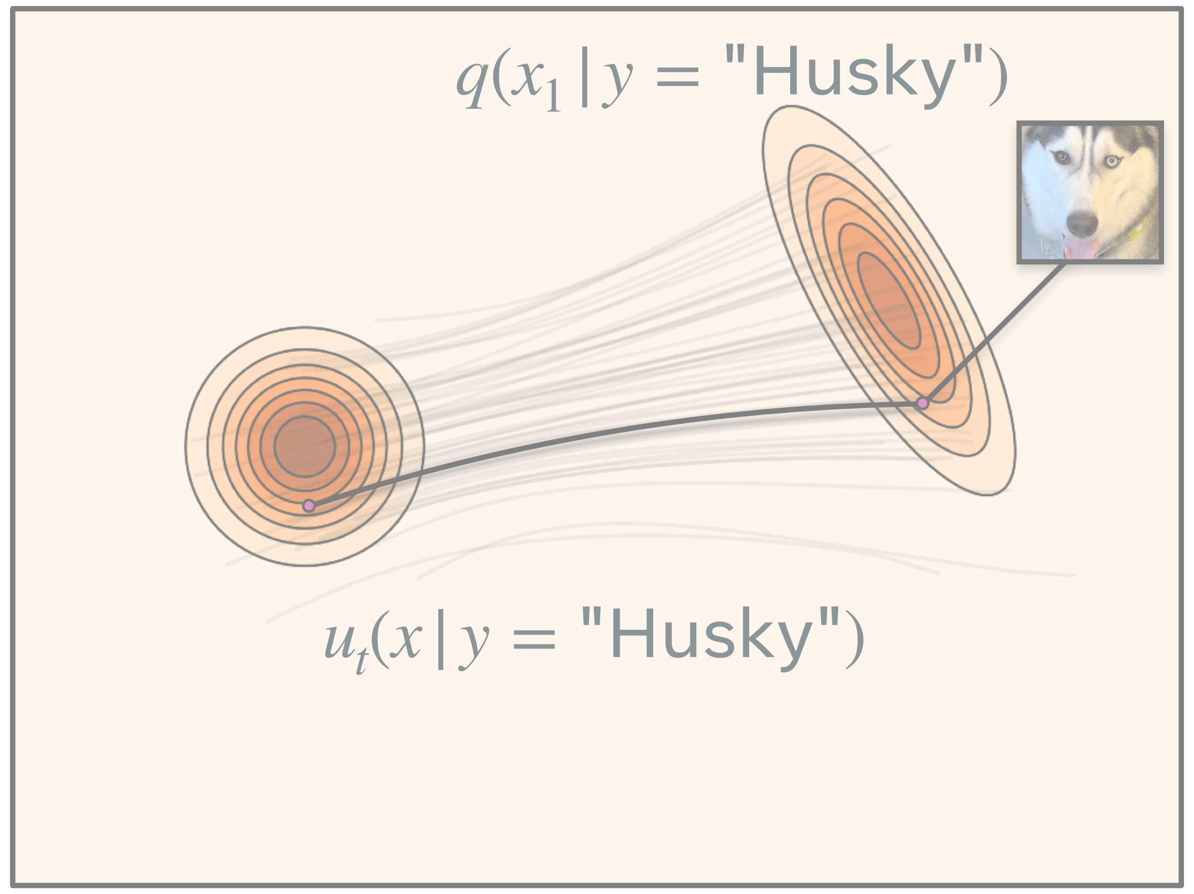
"Scaling Rectified Flow Transformers for High-Resolution Image Synthesis" Esser et al. (2024)

"Movie Gen: A Cast of Media Foundation Models" Polyak et al. (2024)

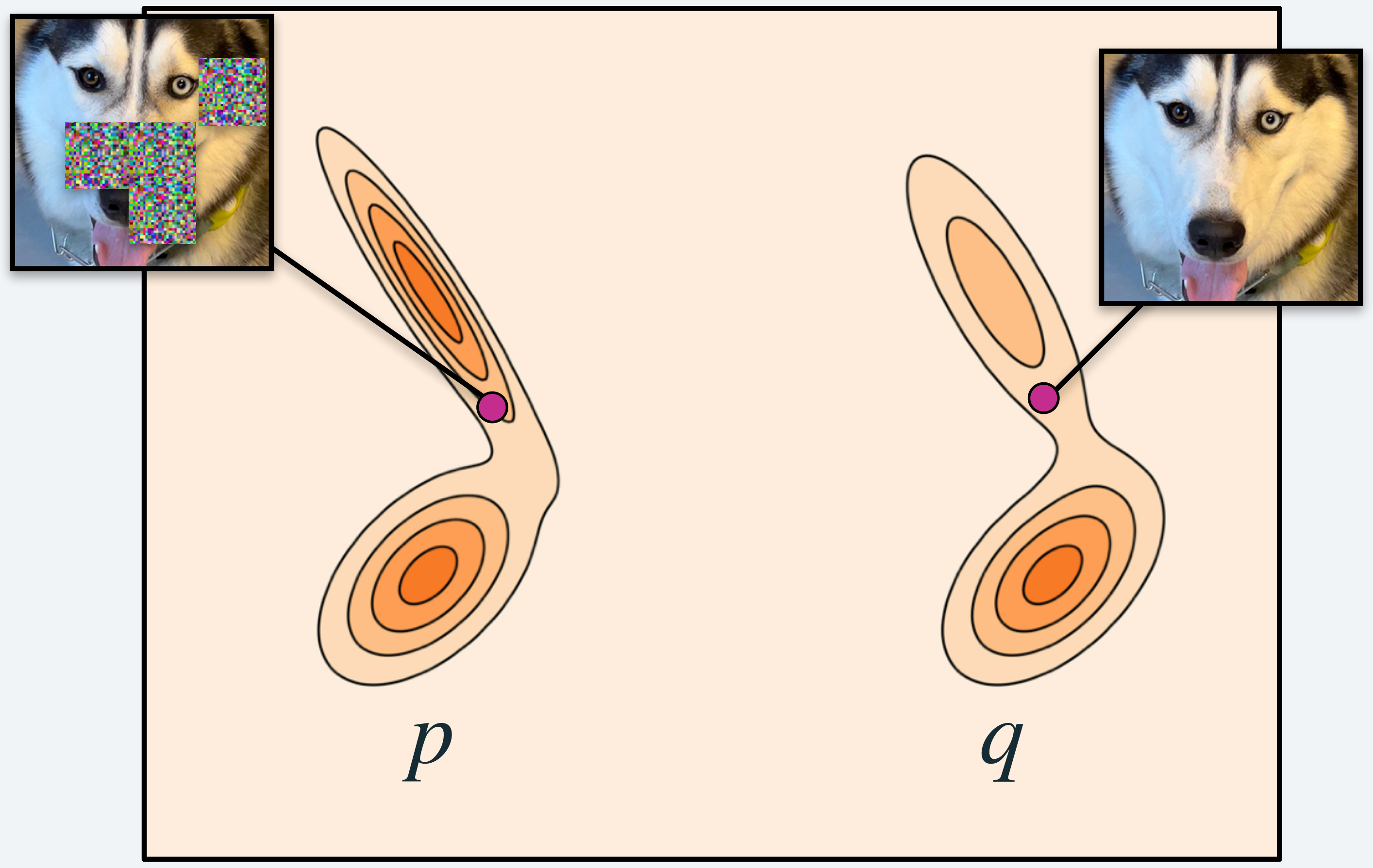
Open Problem

How to guide FM with non-Gaussian paths?

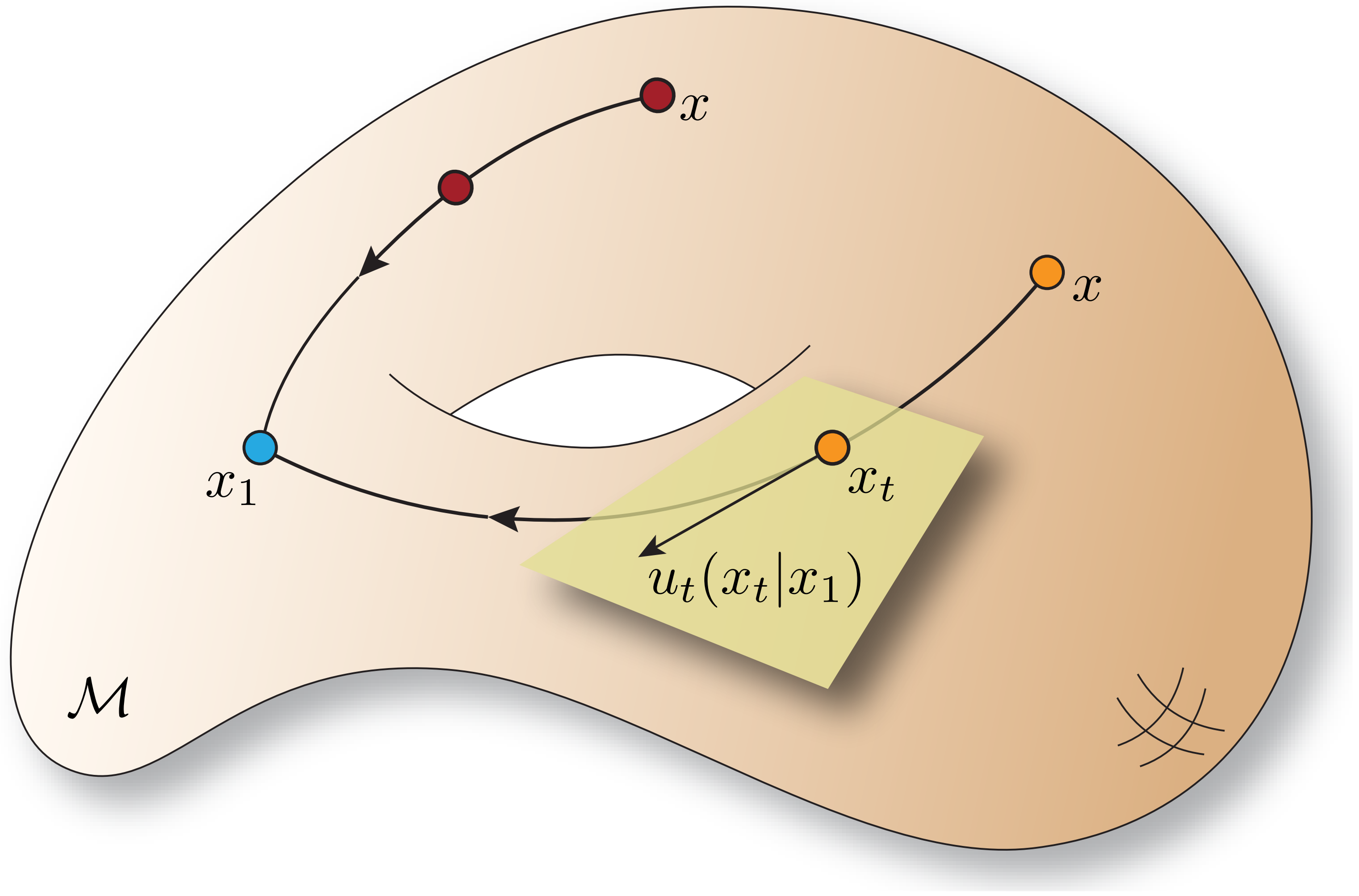
Conditioning and Guidance



Data Couplings



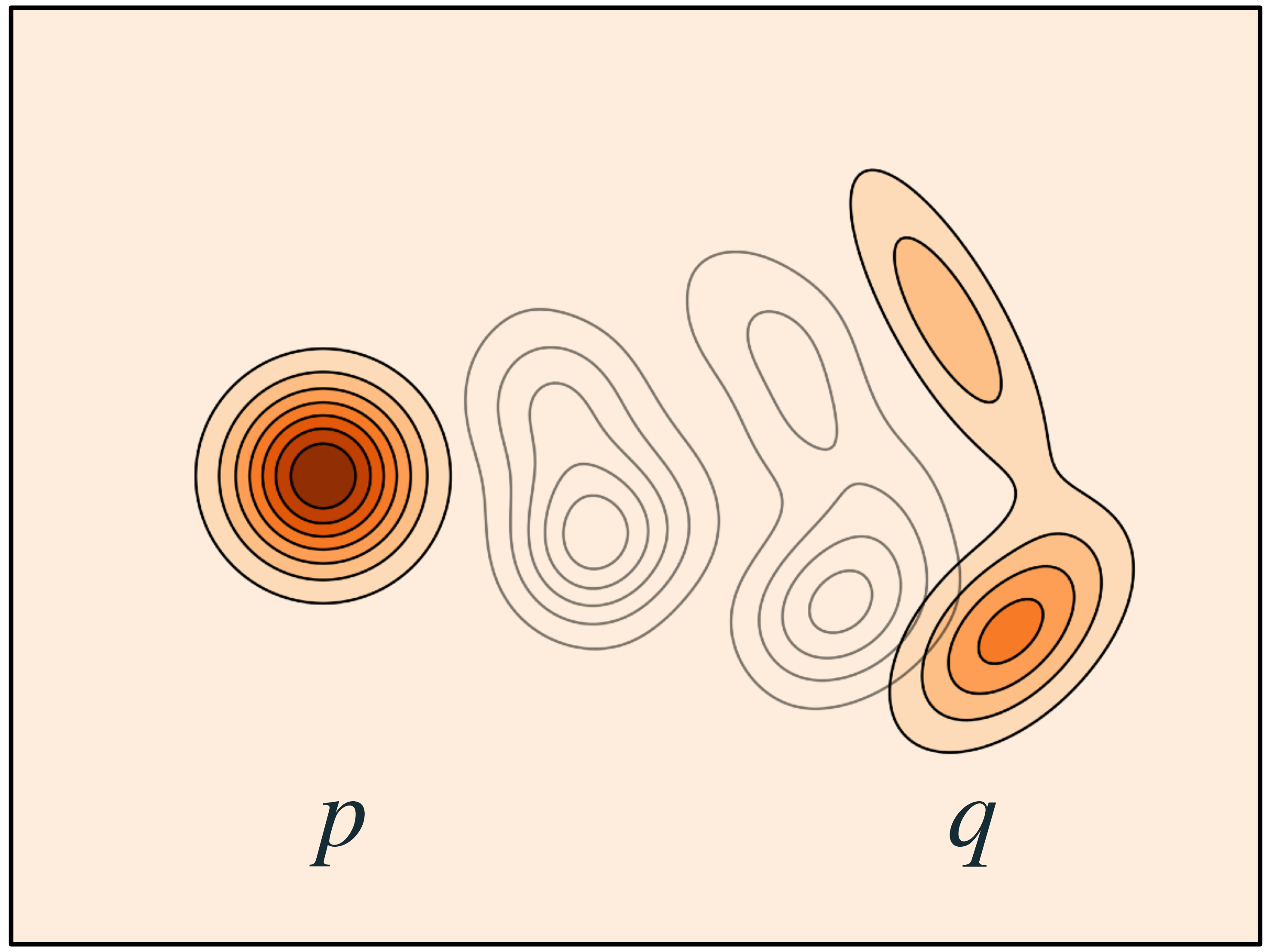
Geometric Flow Matching



Data Couplings

Until now: focused on successfully transforming p to q.

$$\mathscr{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t,X_1,X_t} \| u_t(X_t \mid X_1) - u_t^\theta(X_t) \|^2$$



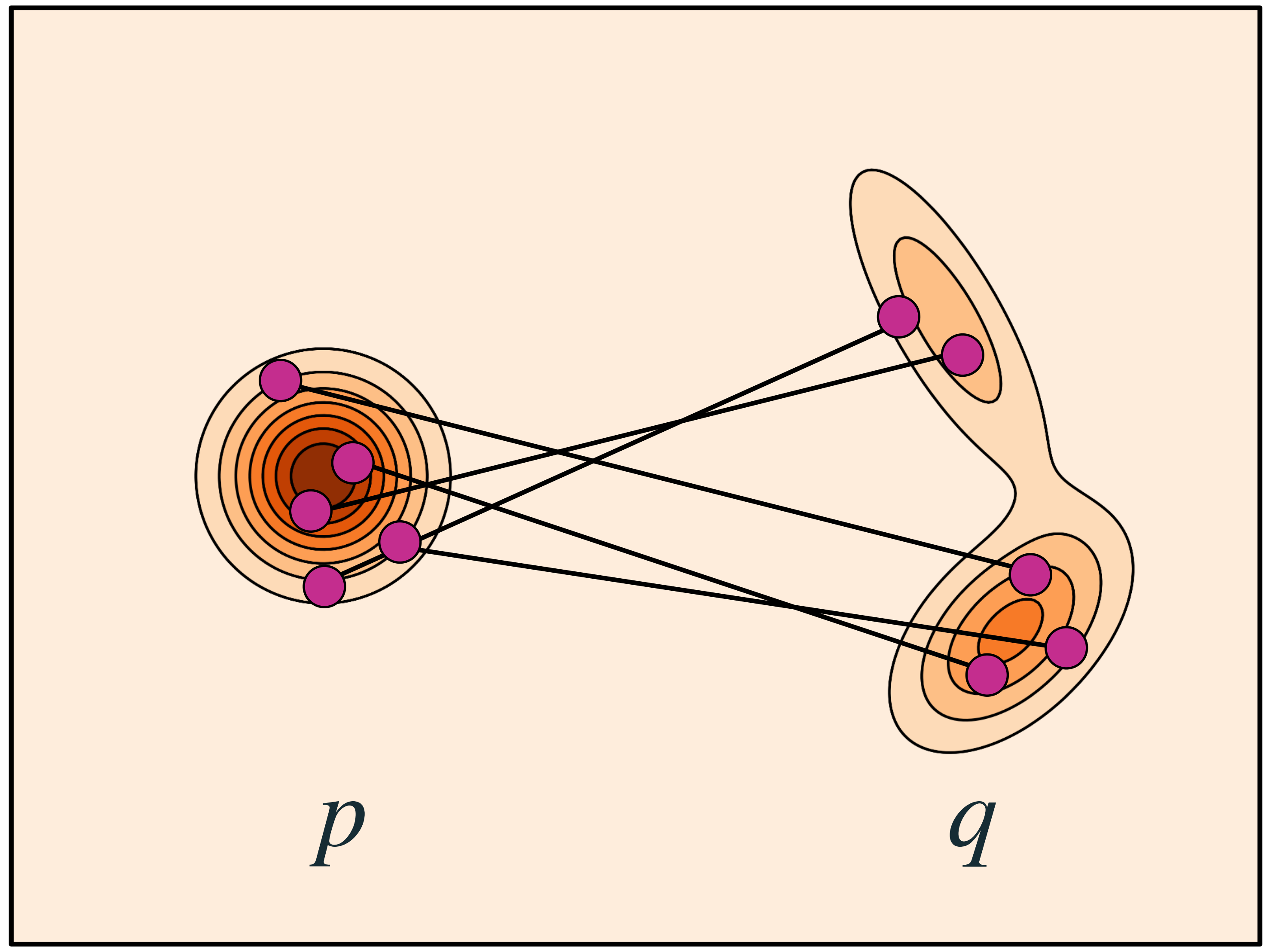
$X_t = \psi_t(X_0 | X_1)$

Data Couplings

Until now: focused on successfully transforming p to q.

$$\mathscr{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t,X_1,X_t} \| u_t(X_t | X_1) - u_t^\theta(X_t) \|^2$$

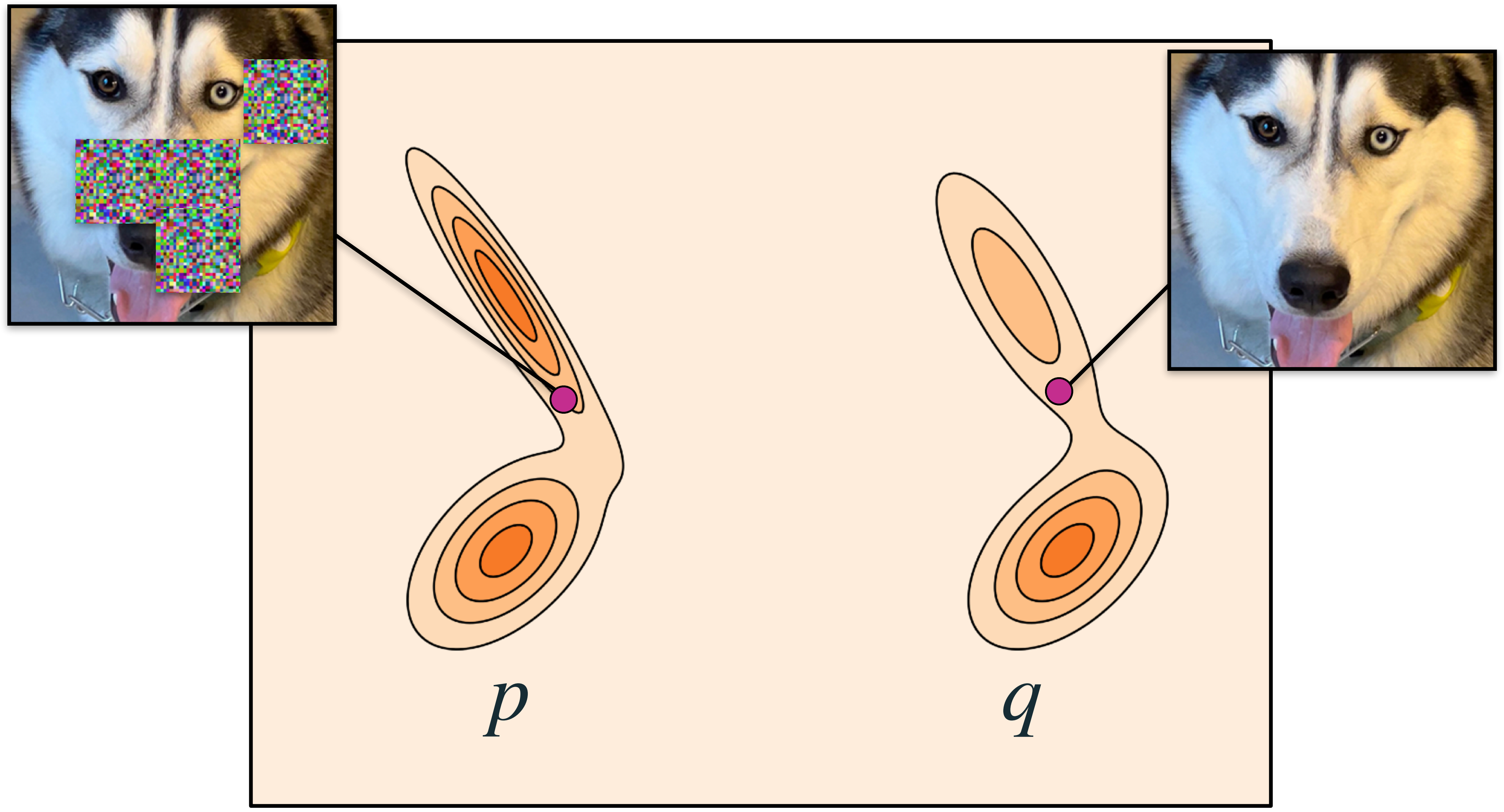
$$(X_0, X_1) \sim \pi_{0,1} = p(X_0)q(X_1)$$



What about dependent couplings?

Data Couplings

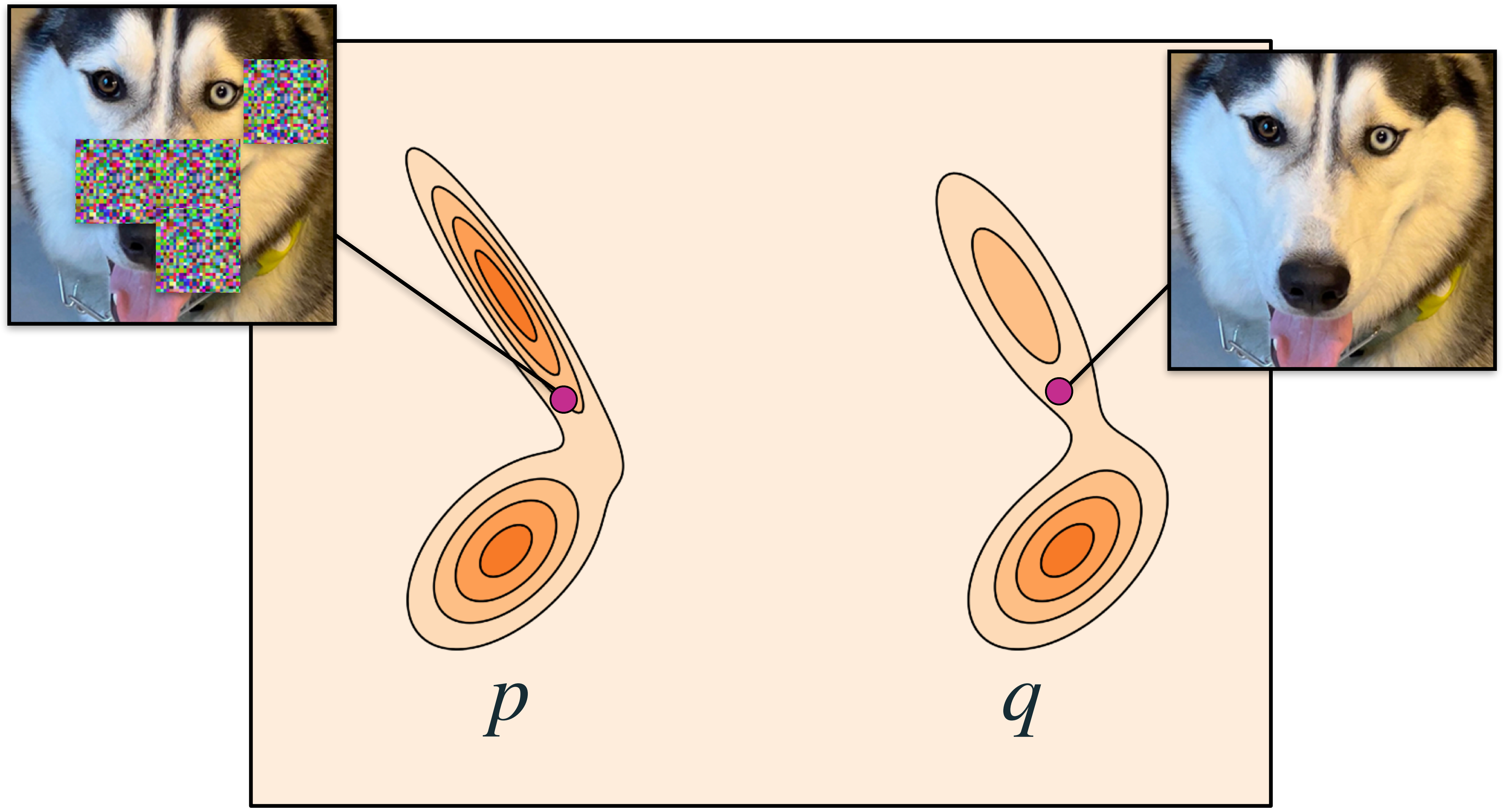
Paired Data



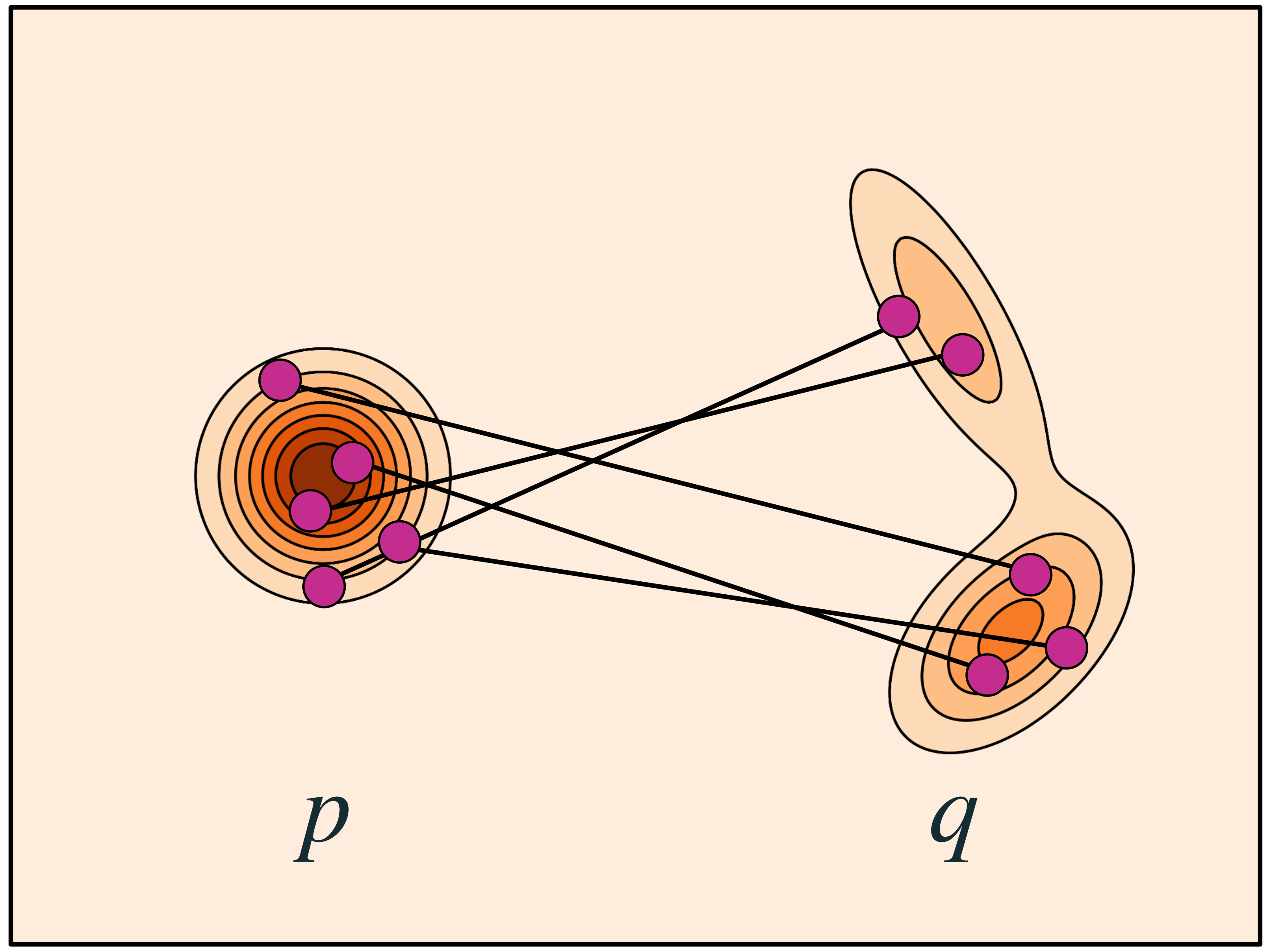
- Non-Gaussian source distribution
- Alternative conditioning approach
- Inverse problems

Data Couplings

Paired Data



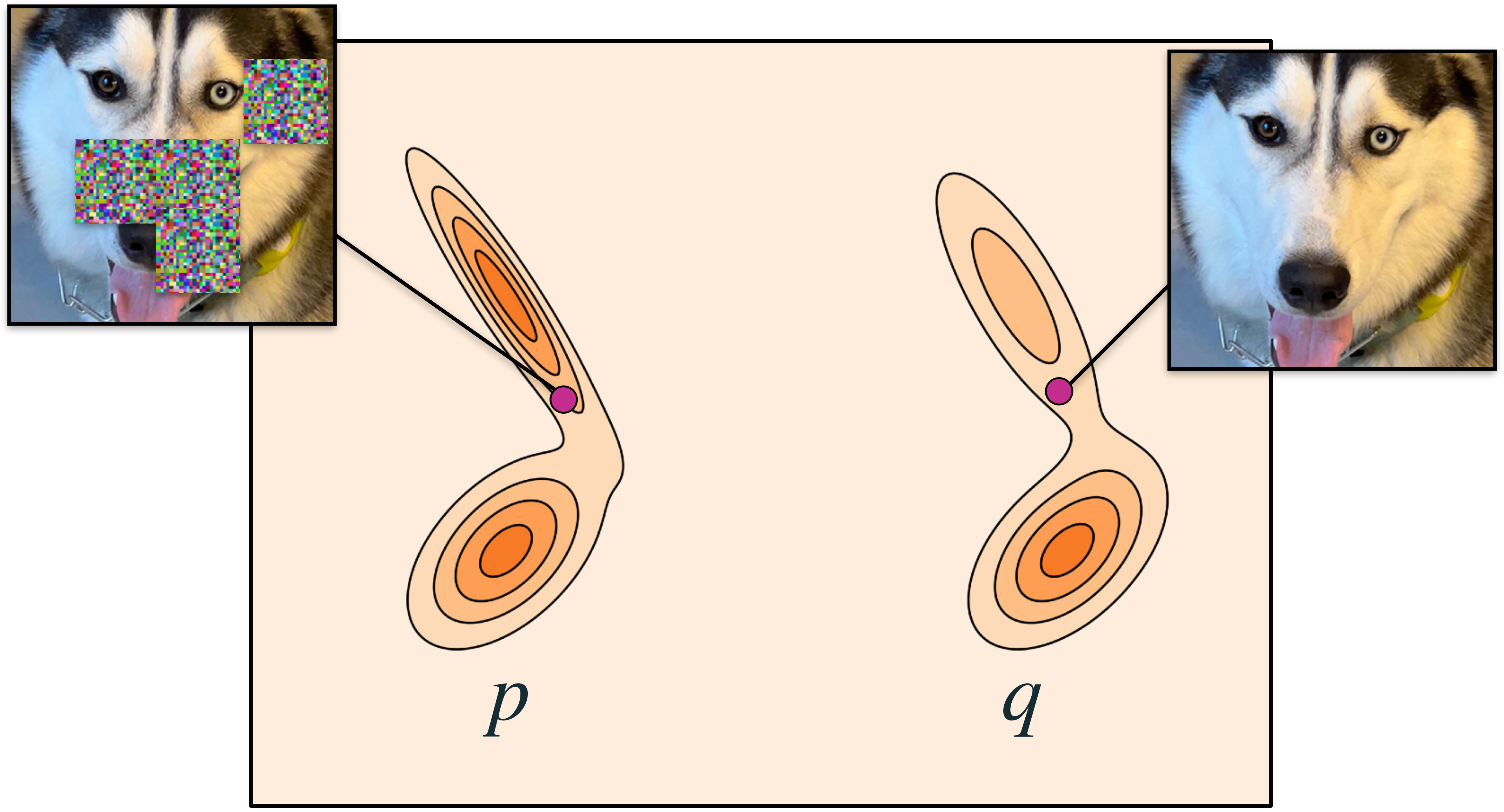
Multisample Couplings



- Non-Gaussian source distribution
- Alternative conditioning approach
- Inverse problems

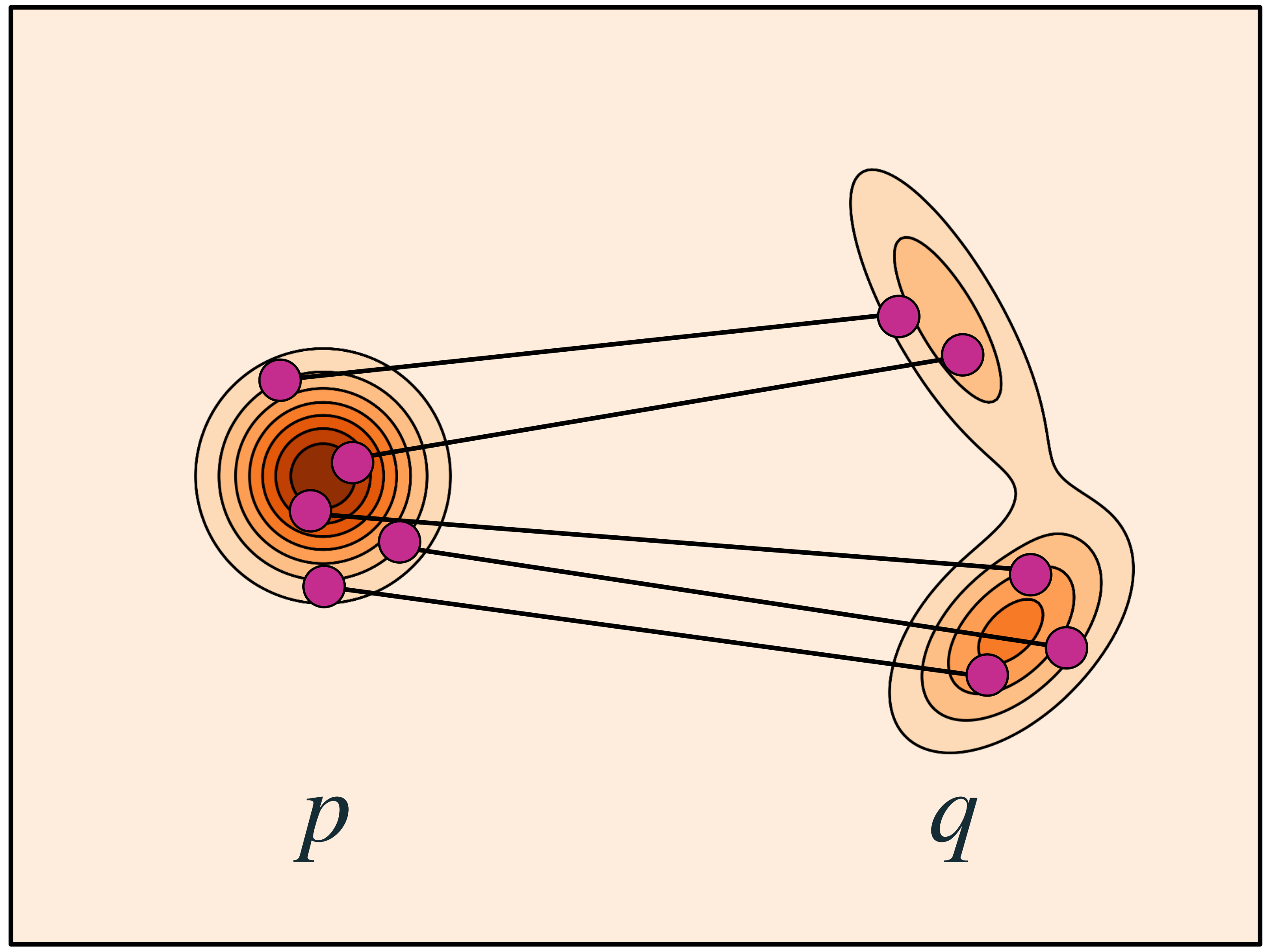
Data Couplings

Paired Data



- Non-Gaussian source distribution
- Alternative conditioning approach
- Inverse problems

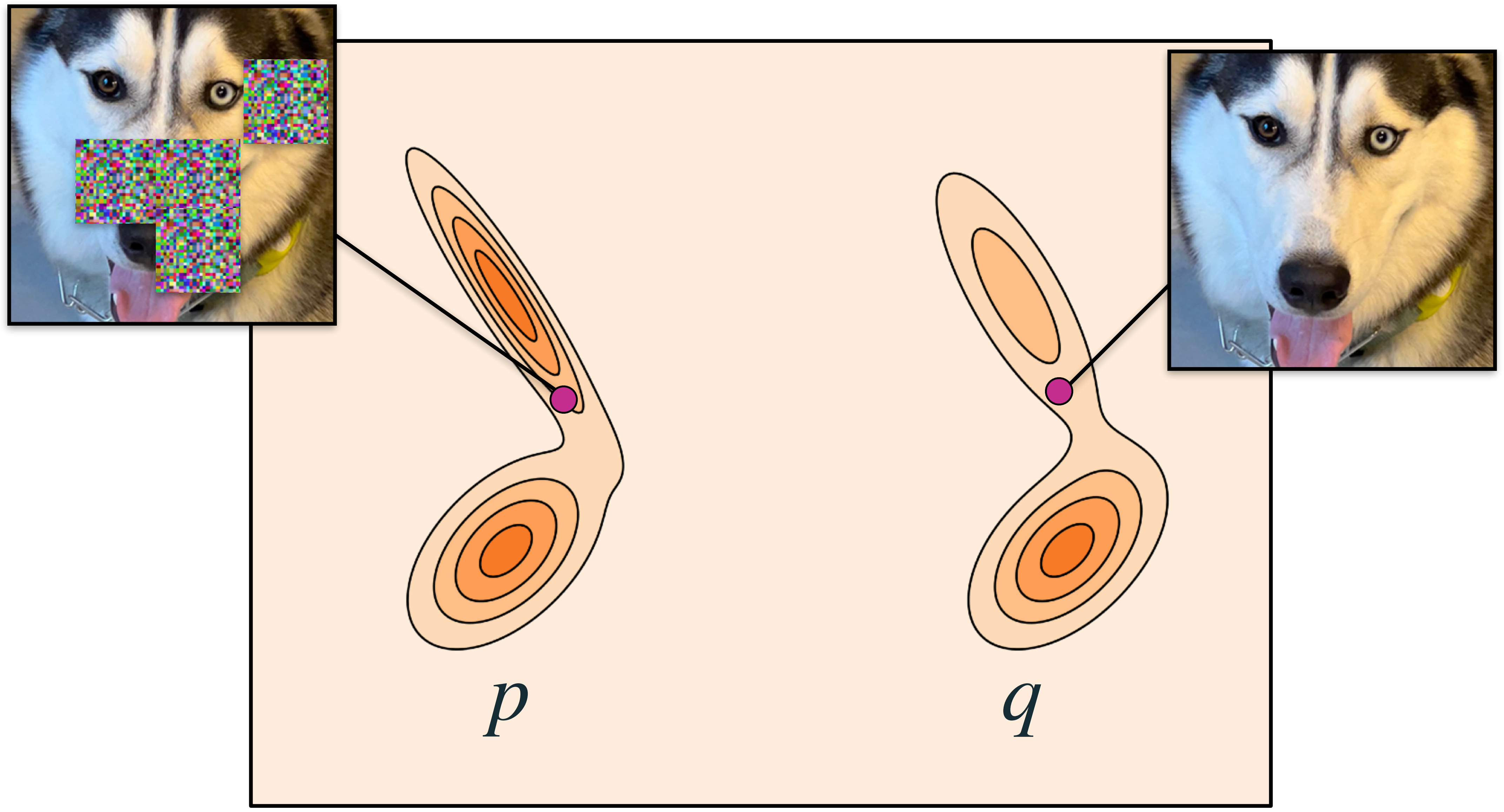
Multisample Couplings



- Applications to Optimal Transport
- Efficiency: straighter trajectories

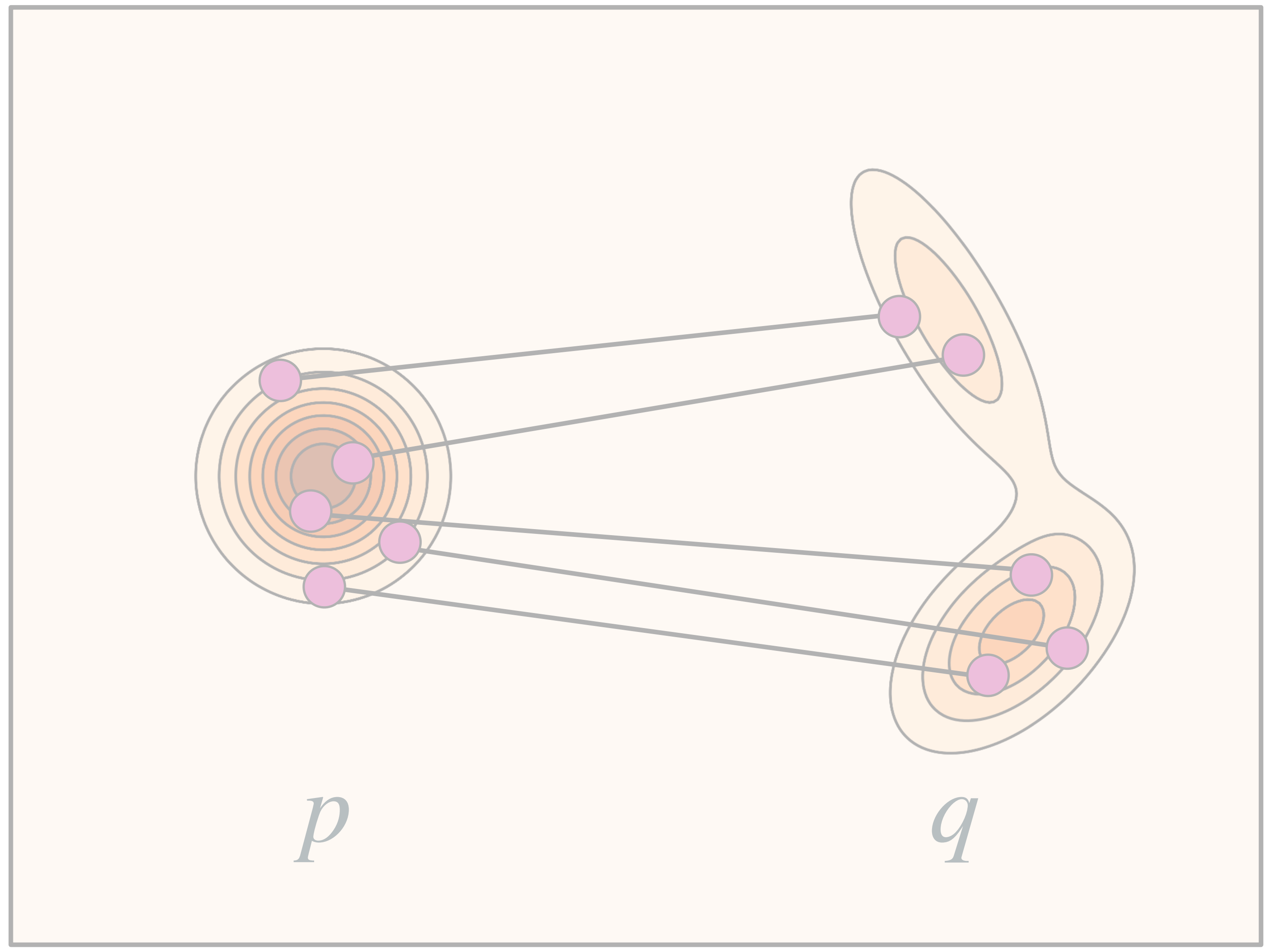
Data Couplings

Paired Data



- Non-Gaussian source distribution
- Alternative conditioning approach
- Inverse problems

Multisample Couplings



- Applications to Optimal Transport
- Efficiency: straighter trajectories

Paired Data

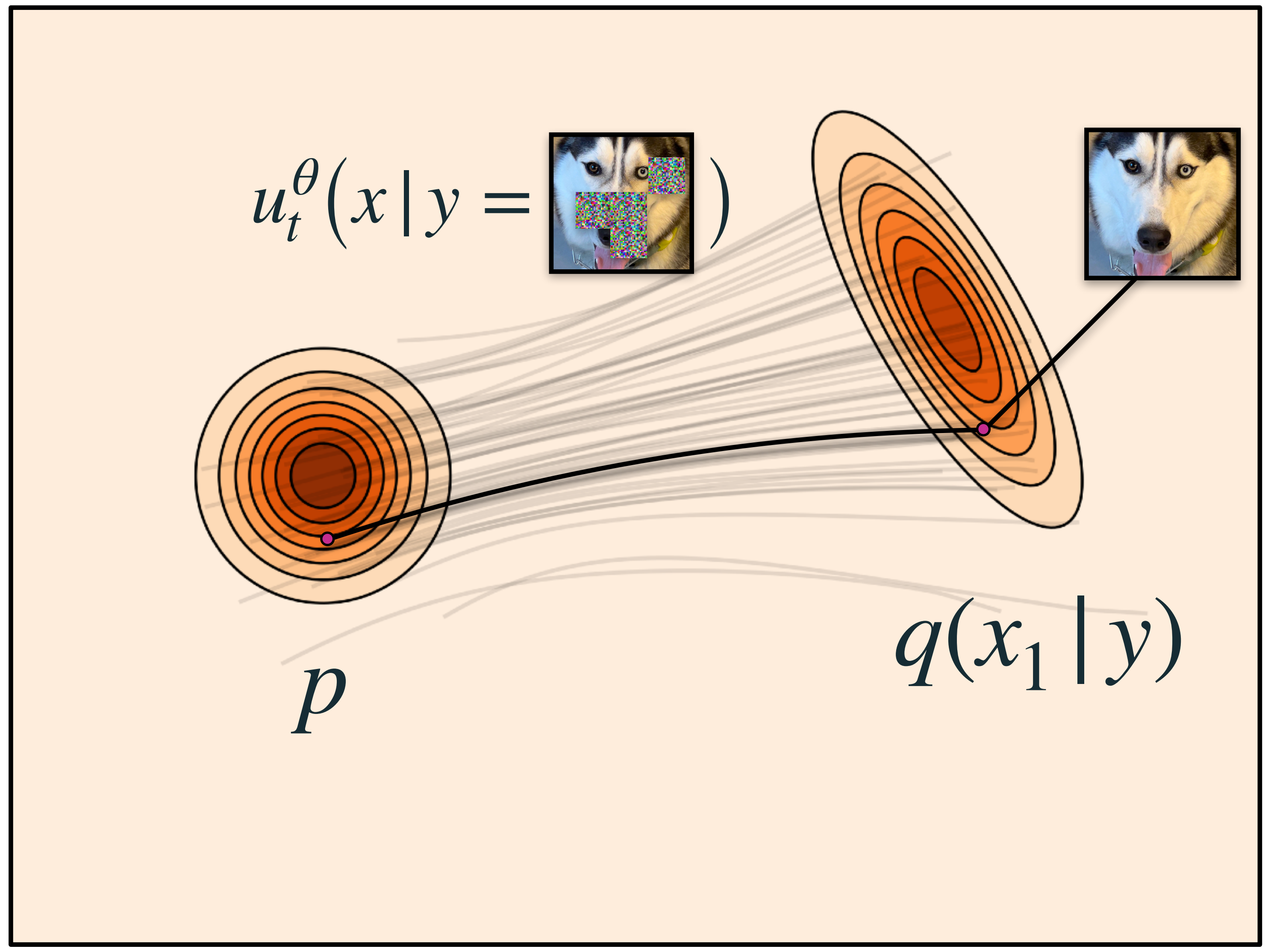
Labeled Data: $(X_1, Y) \sim q$ → Goal: learn $q(x_1 \mid y)$

Paired Data

Labeled Data: $(X_1, Y) \sim q$

Goal: learn $q(x_1 \mid y)$

Conditional Model

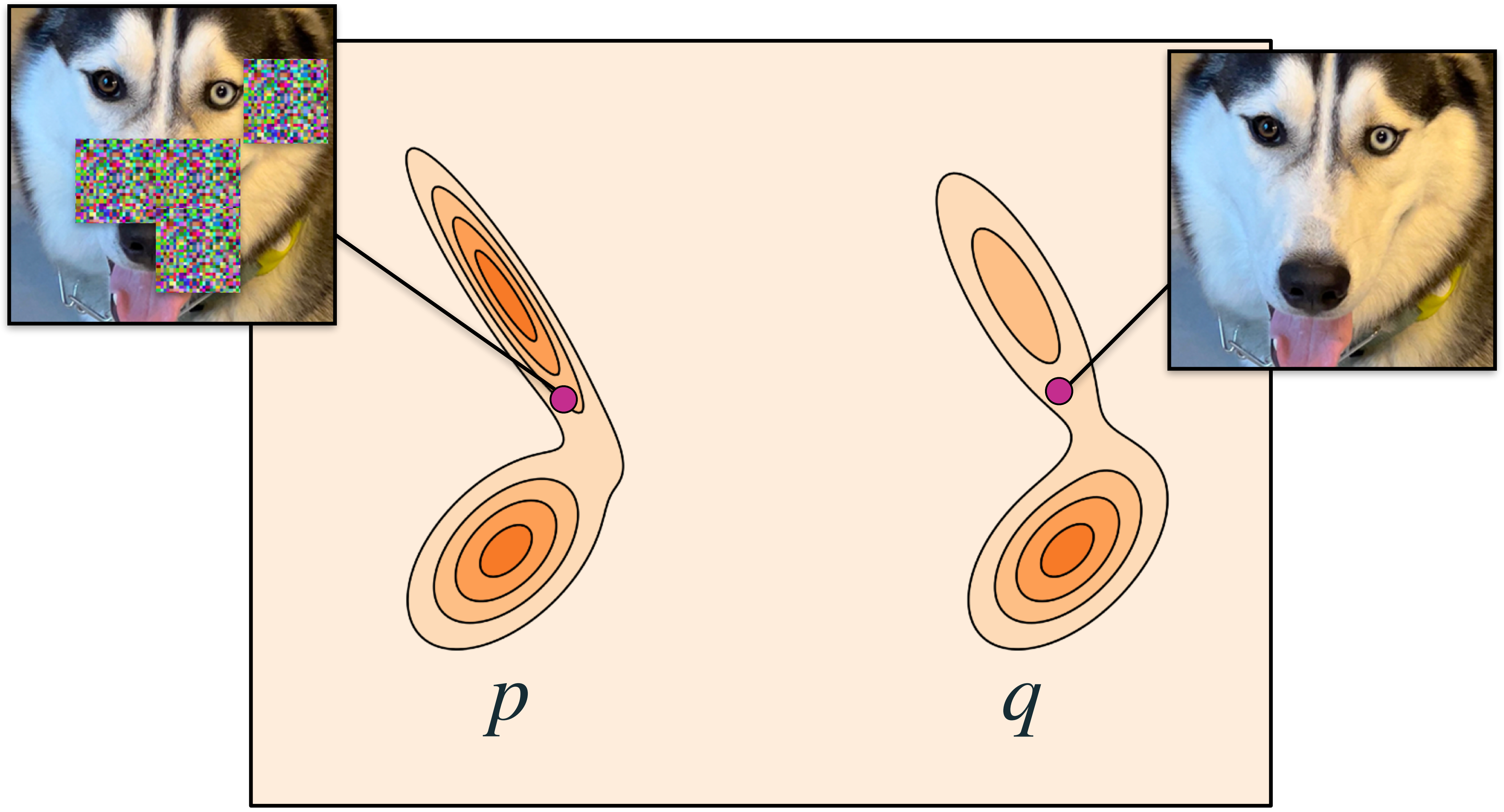


$u_t(x \mid y)$ generates $p_{t|Y}(x \mid y)$

Paired Data

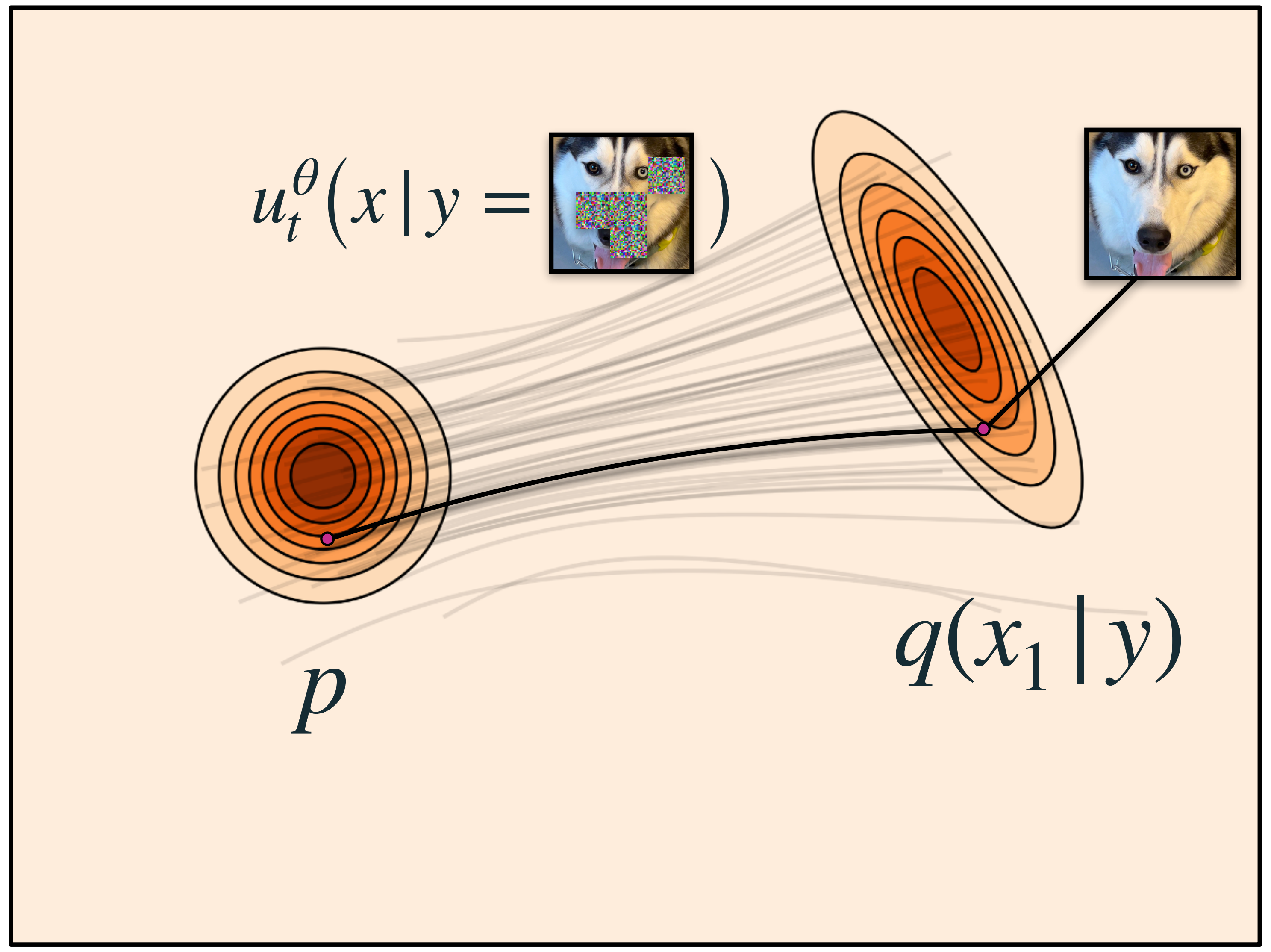
Labeled Data: $(X_1, Y) \sim q$ → **Goal:** learn $q(x_1 \mid y)$

Dependent Couplings



$X_0 = Y + \epsilon \quad \sim p$

Conditional Model



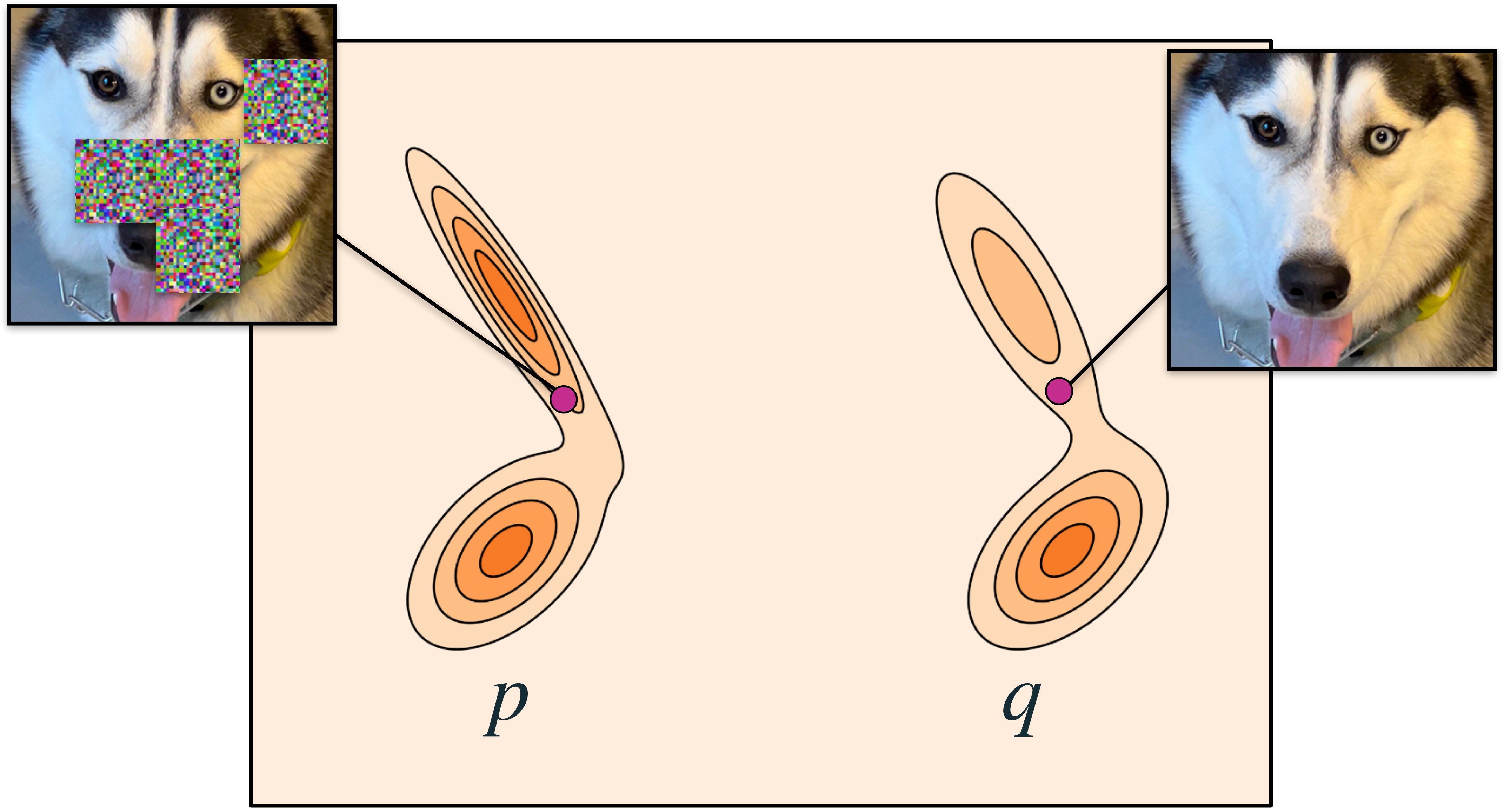
$u_t(x \mid y)$ generates $p_{t|Y}(x \mid y)$

"I2SB: Image-to-Image Schrödinger Bridge" Liu et al. (2023)
"Stochastic interpolants with data-dependent couplings" Albergo et al. (2024)

Paired Data

Labeled Data: $(X_1, Y) \sim q$ → Goal: learn $q(x_1 \mid y)$

Dependent Couplings



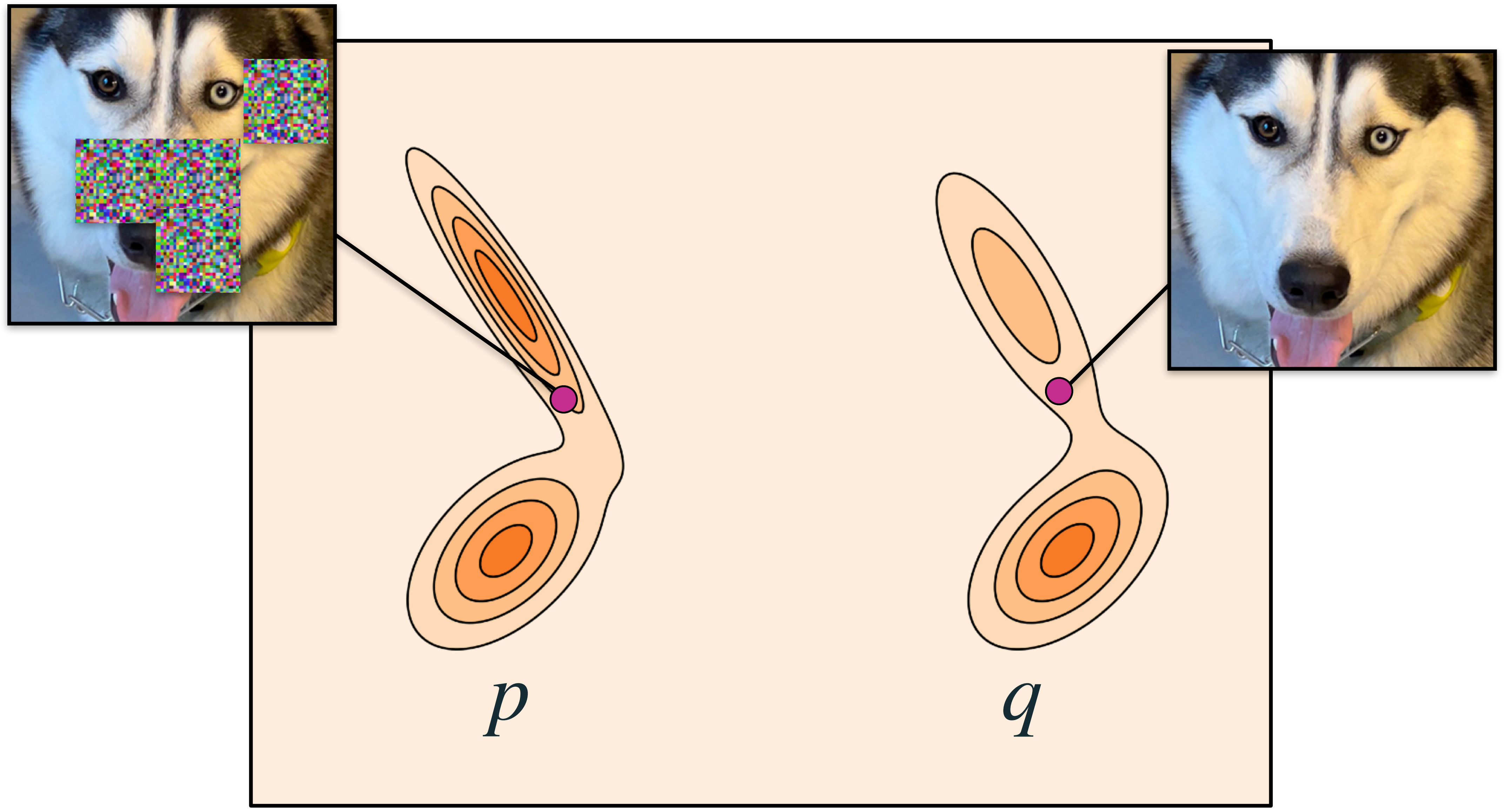
$$X_0 = Y + \epsilon \quad \sim p$$

"I2SB: Image-to-Image Schrödinger Bridge" Liu et al. (2023)
"Stochastic interpolants with data-dependent couplings" Albergo et al. (2024)

Paired Data

Labeled Data: $(X_1, Y) \sim q$ → **Goal:** learn $q(x_1 \mid y)$

Dependent Couplings



$$X_0 = Y + \epsilon \quad \sim p$$

Alter source distribution and coupling instead of adding condition

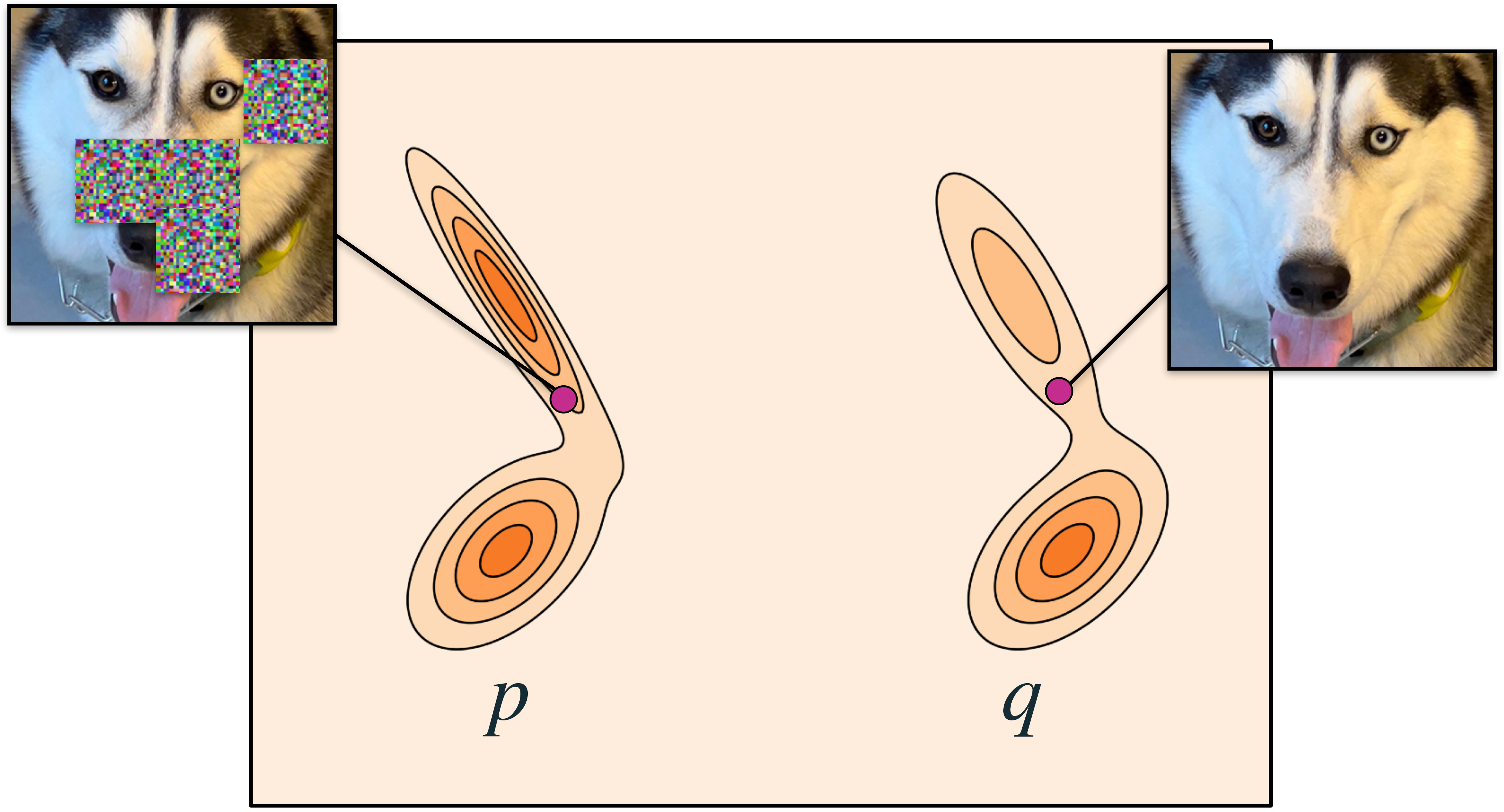
"I2SB: Image-to-Image Schrödinger Bridge" Liu et al. (2023)
"Stochastic interpolants with data-dependent couplings" Albergo et al. (2024)

$X_t = \psi_t(X_0 | X_1)$

Paired Data

Labeled Data: $(X_1, Y) \sim q$ → Goal: learn $q(x_1 | y)$

Dependent Couplings



$$X_0 = Y + \epsilon \quad \sim p$$

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t,X_1,X_t} \| u_t(X_t | X_1) - u_t^\theta(X_t) \|^2$$

$$(X_0, X_1, Y) \sim \pi_{0|1}(x_0 | x_1, y) q(x_1, y)$$

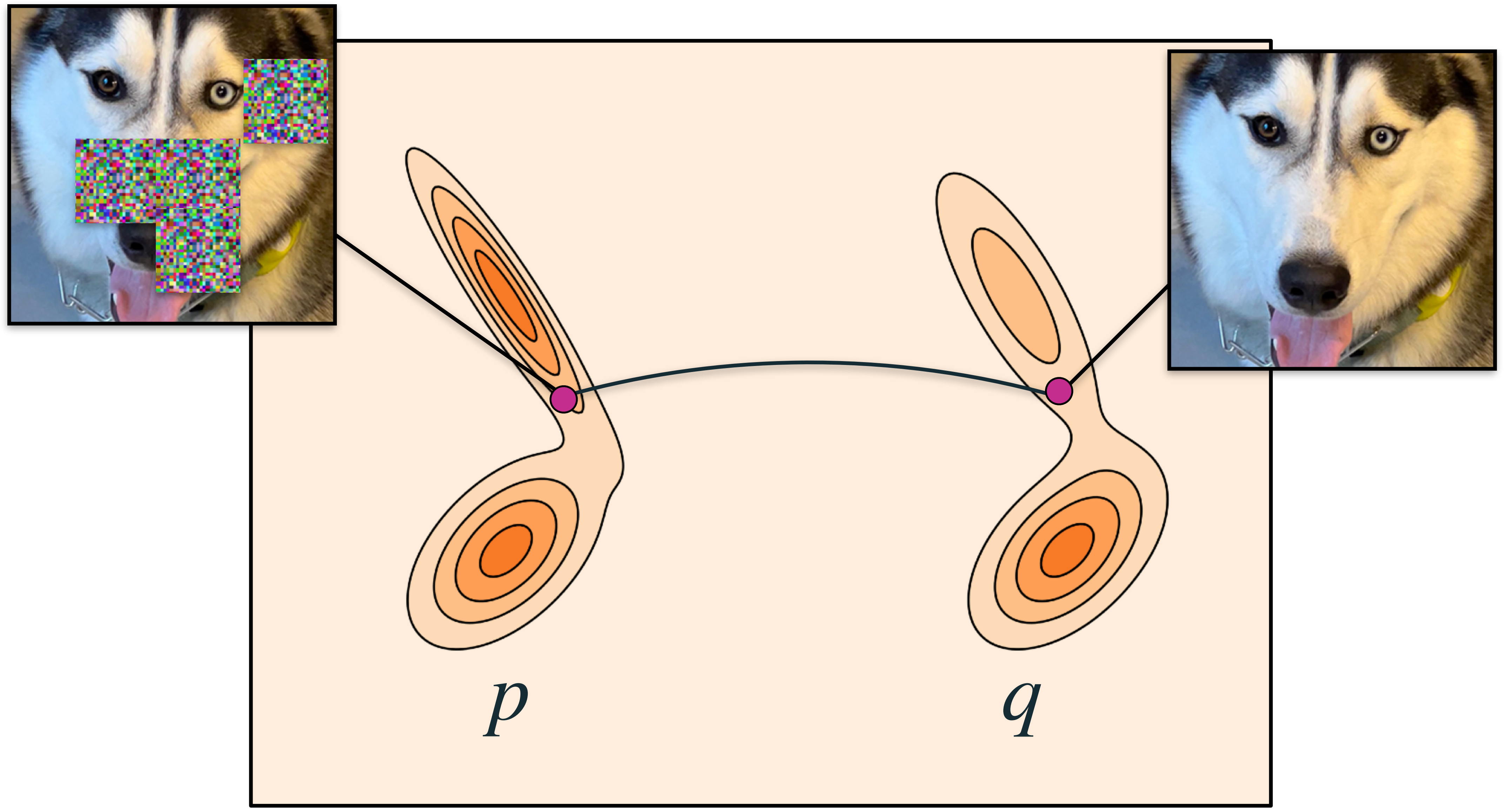
"I2SB: Image-to-Image Schrödinger Bridge" Liu et al. (2023)
"Stochastic interpolants with data-dependent couplings" Albergo et al. (2024)

$X_t = \psi_t(X_0 \mid X_1)$

Paired Data

Labeled Data: $(X_1, Y) \sim q$ → Goal: learn $q(x_1 \mid y)$

Dependent Couplings



$X_0 = Y + \epsilon \quad \sim p$

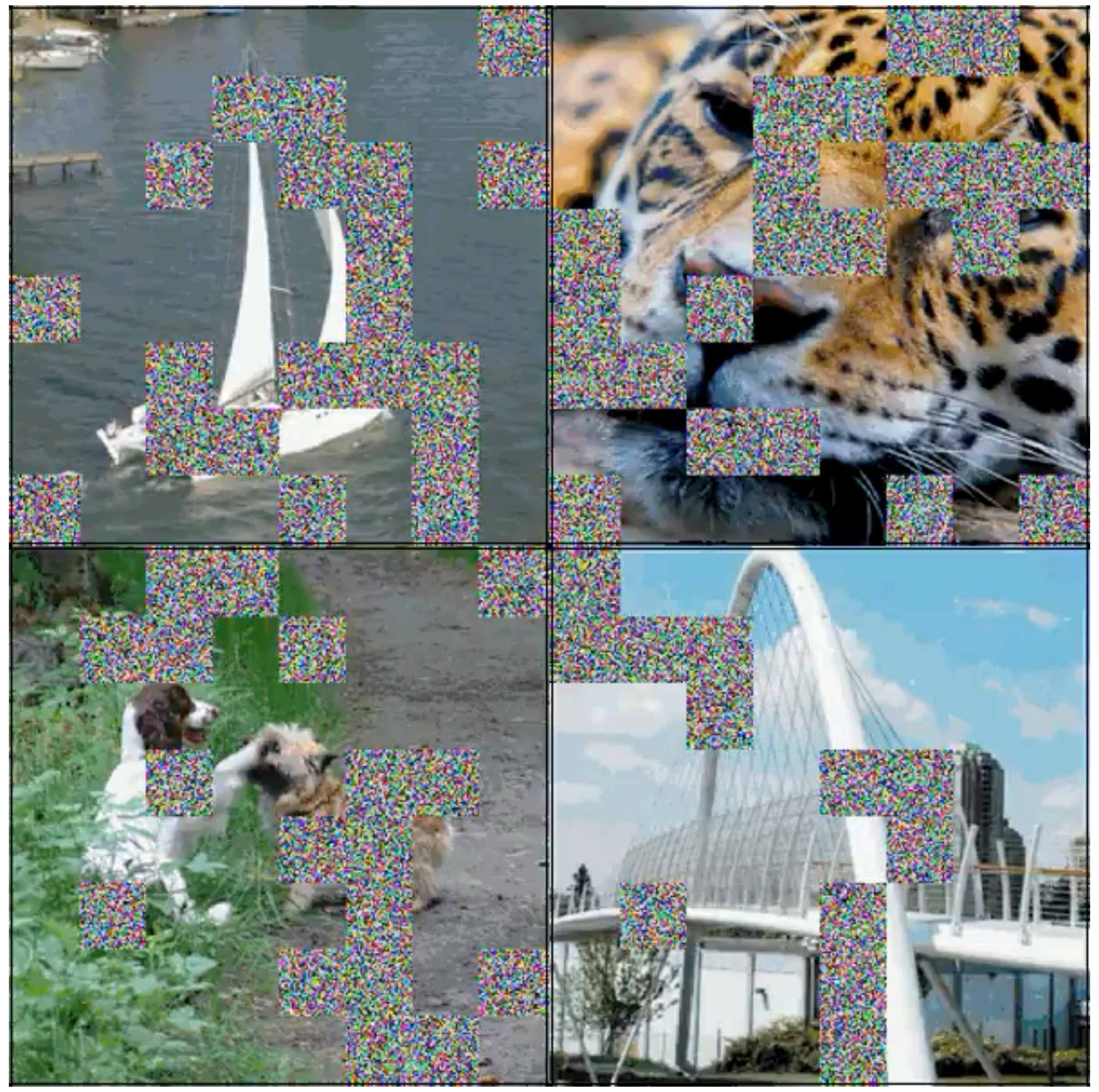
$$\mathscr{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t,X_1,X_t} \| u_t(X_t \mid X_1) - u_t^\theta(X_t) \|^2$$

$$(X_0, X_1, Y) \sim \pi_{0|1}(x_0 \mid x_1, y) q(x_1, y)$$

$$\psi_1(X_0 \mid Y = y) \sim q(x_1 \mid y)$$

"I2SB: Image-to-Image Schrödinger Bridge" Liu et al. (2023)
"Stochastic interpolants with data-dependent couplings" Albergo et al. (2024)

Paired Data



Infilling

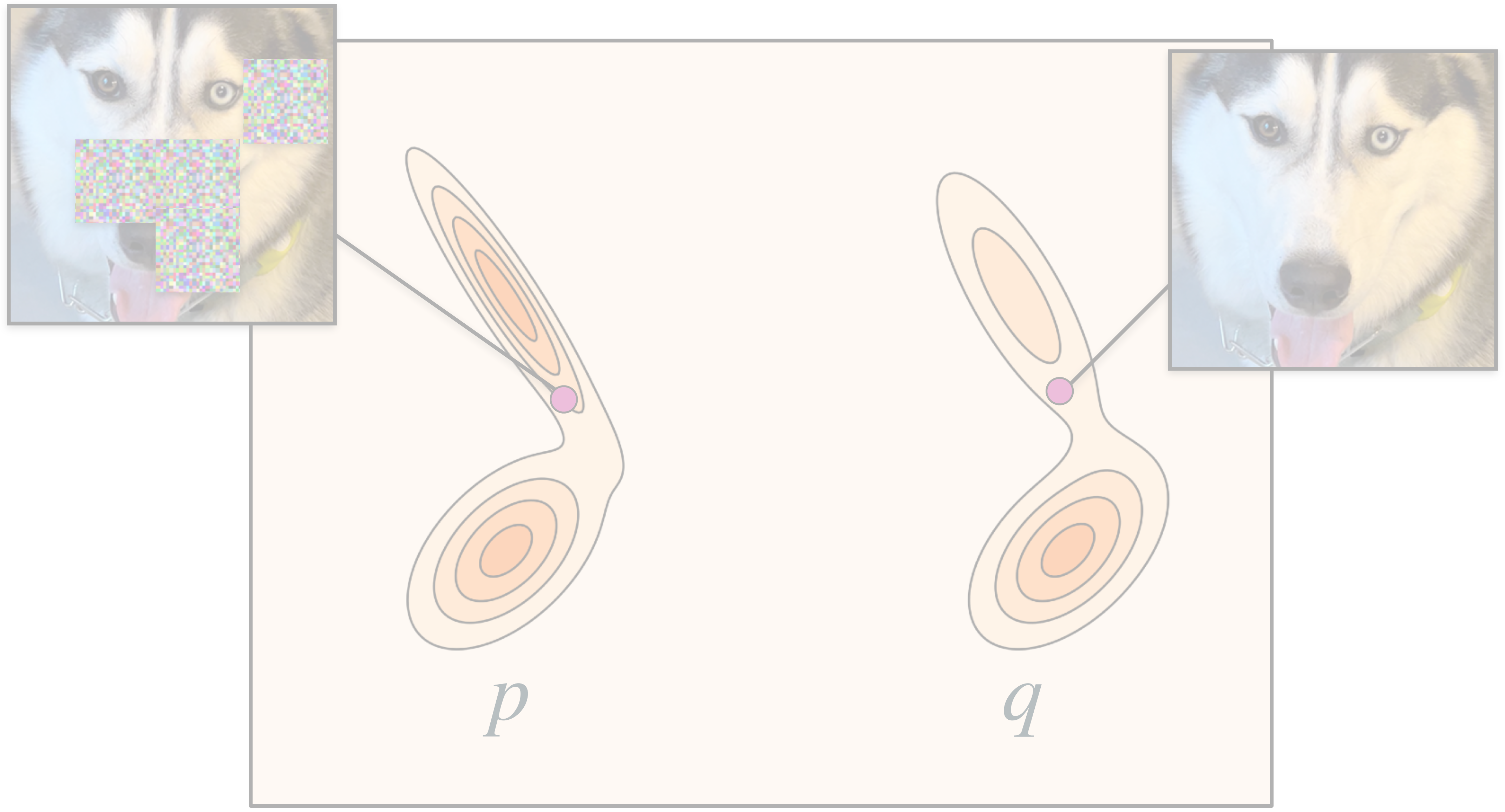
Model	FID-50K
Improved DDPM (Nichol & Dhariwal, 2021)	12.26
SR3 (Saharia et al., 2022)	11.3
ADM (Dhariwal & Nichol, 2021)	7.49
Cascaded Diffusion (Ho et al., 2022a)	4.88
I^2SB (Liu et al., 2023a)	2.70
Dependent Coupling (**Ours**)	**2.13**

Super-resolution $64 \times 64 \rightarrow 256 \times 256$

"Stochastic interpolants with data-dependent couplings" Albergo et al. (2024)

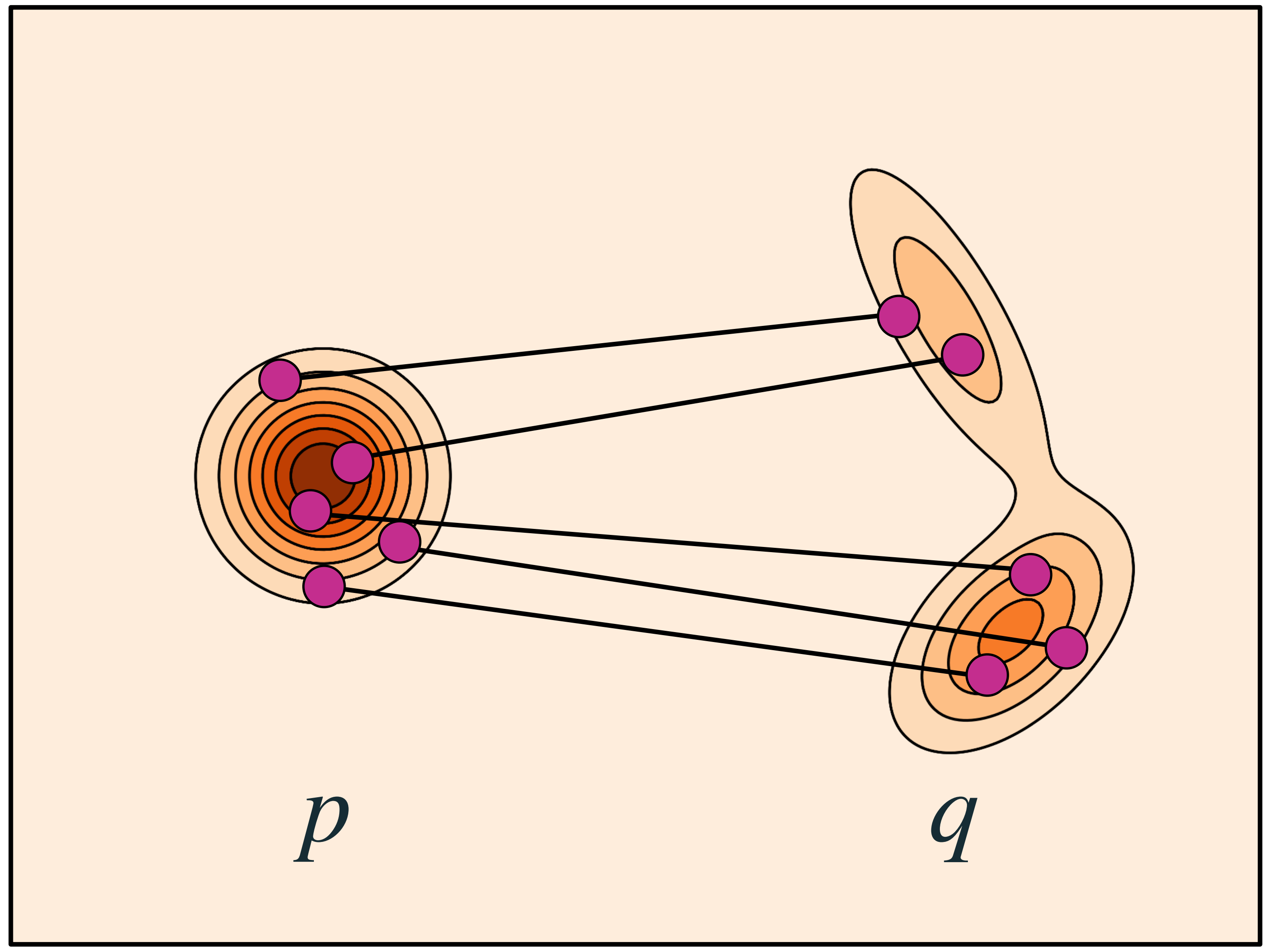
Data Couplings

Paired Data



- Non-Gaussian source distribution
- Alternative conditioning approach
- Inverse problems

Multisample Couplings



- Applications to Optimal Transport
- Efficiency: straighter trajectories

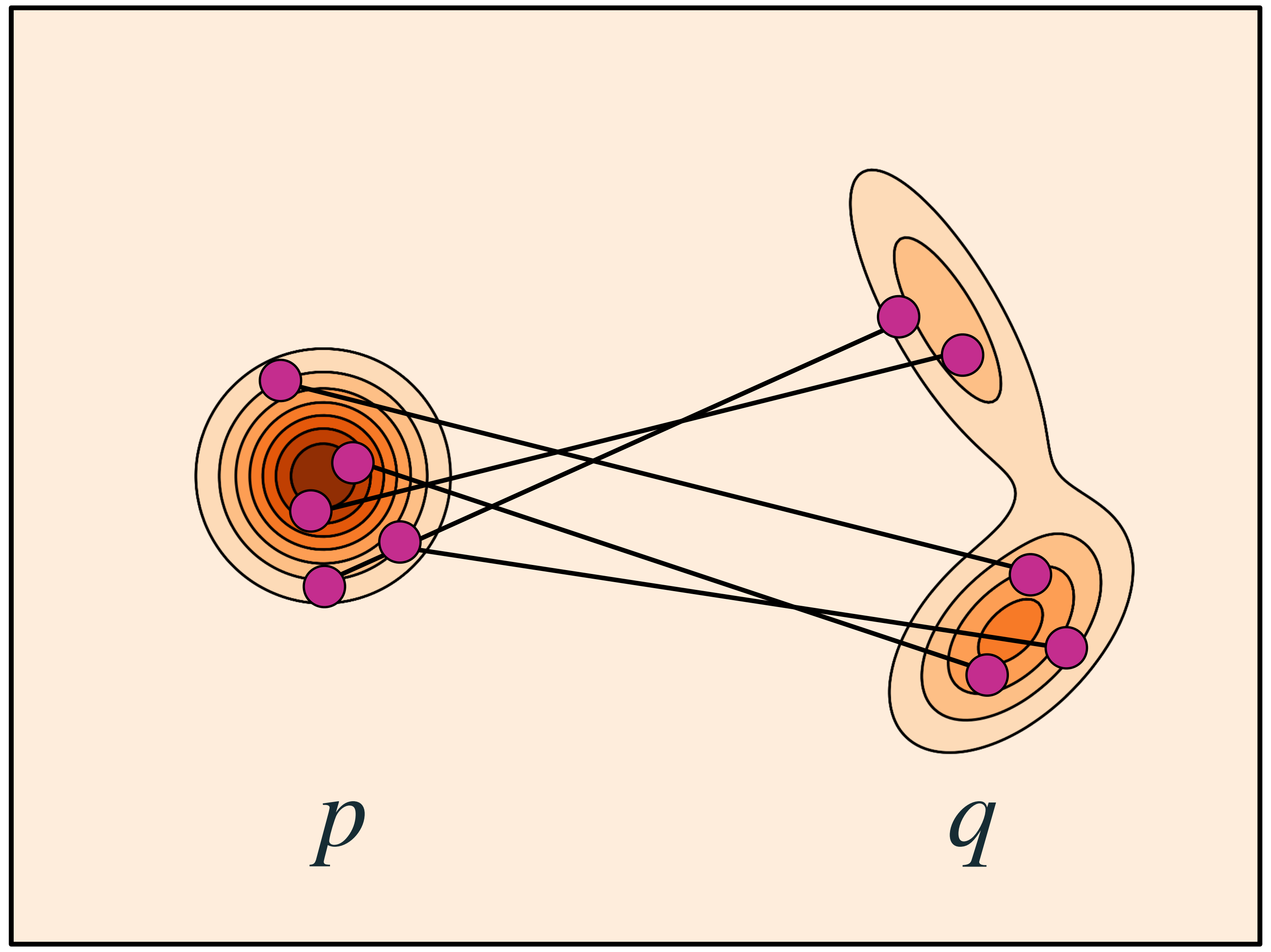
Multisample Couplings

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t,X_1,X_t} \|u_t(X_t|X_1) - u_t^\theta(X_t)\|^2$$

$$(X_0, X_1) \sim \pi_{0,1} = \,?$$

Given uncoupled source and target distributions,

can we build a coupling to induce straighter paths?

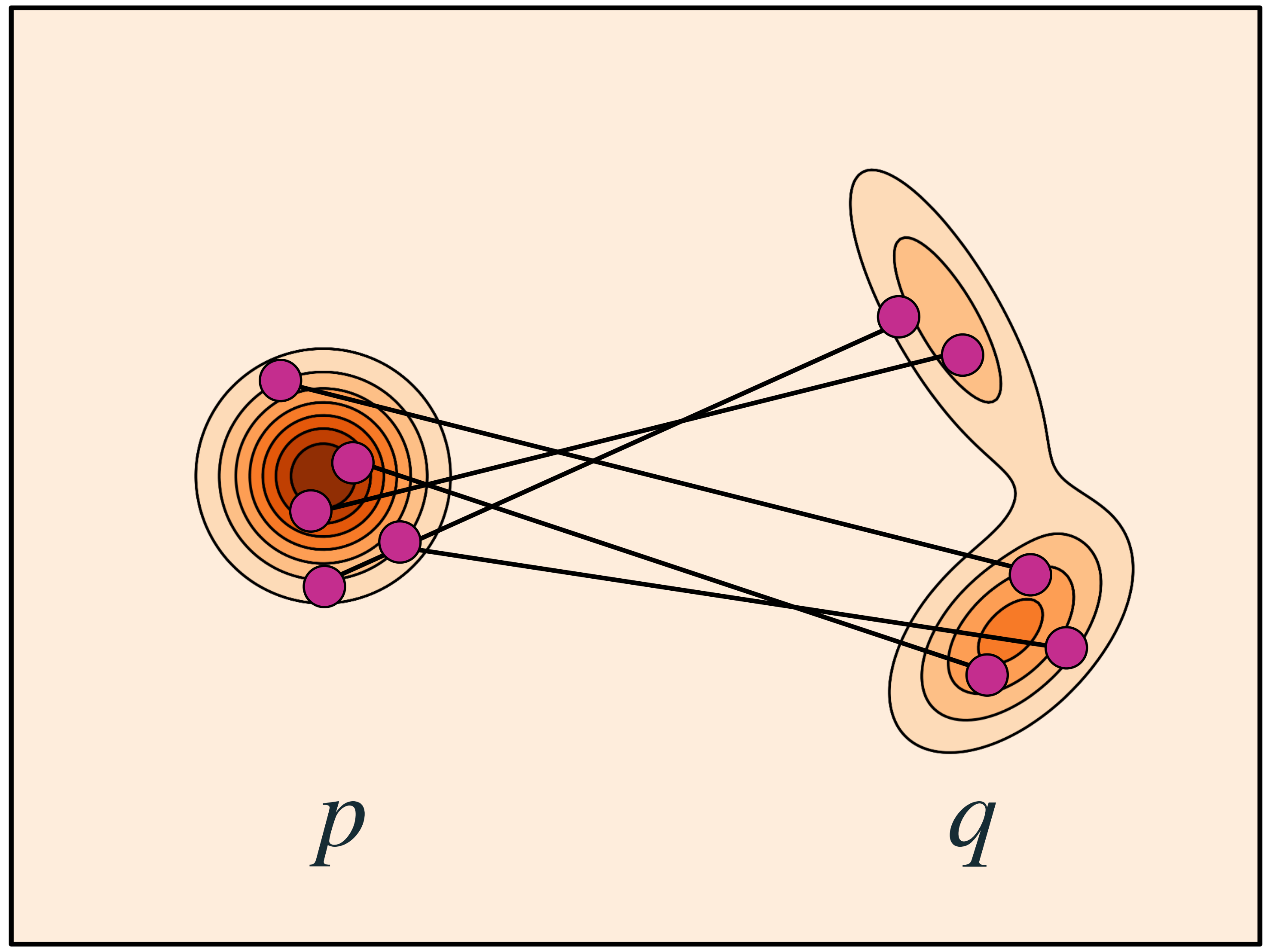


"Multisample Flow Matching: Straightening Flows with Minibatch Couplings" Pooladian et al. (2023)
"Improving and generalizing flow-based generative models with minibatch optimal transport" Tong et al. (2023)

Multisample Couplings

$$\underbrace{\mathrm{KE}(u_t)}_{\text{Kinetic Energy}} \le \underbrace{\mathbb{E}_{\pi_{0,1}} \|X_1 - X_0\|^2}_{\text{Coupling cost}}$$

Marginal u_t with cond-OT FM and $\pi_{0,1}$

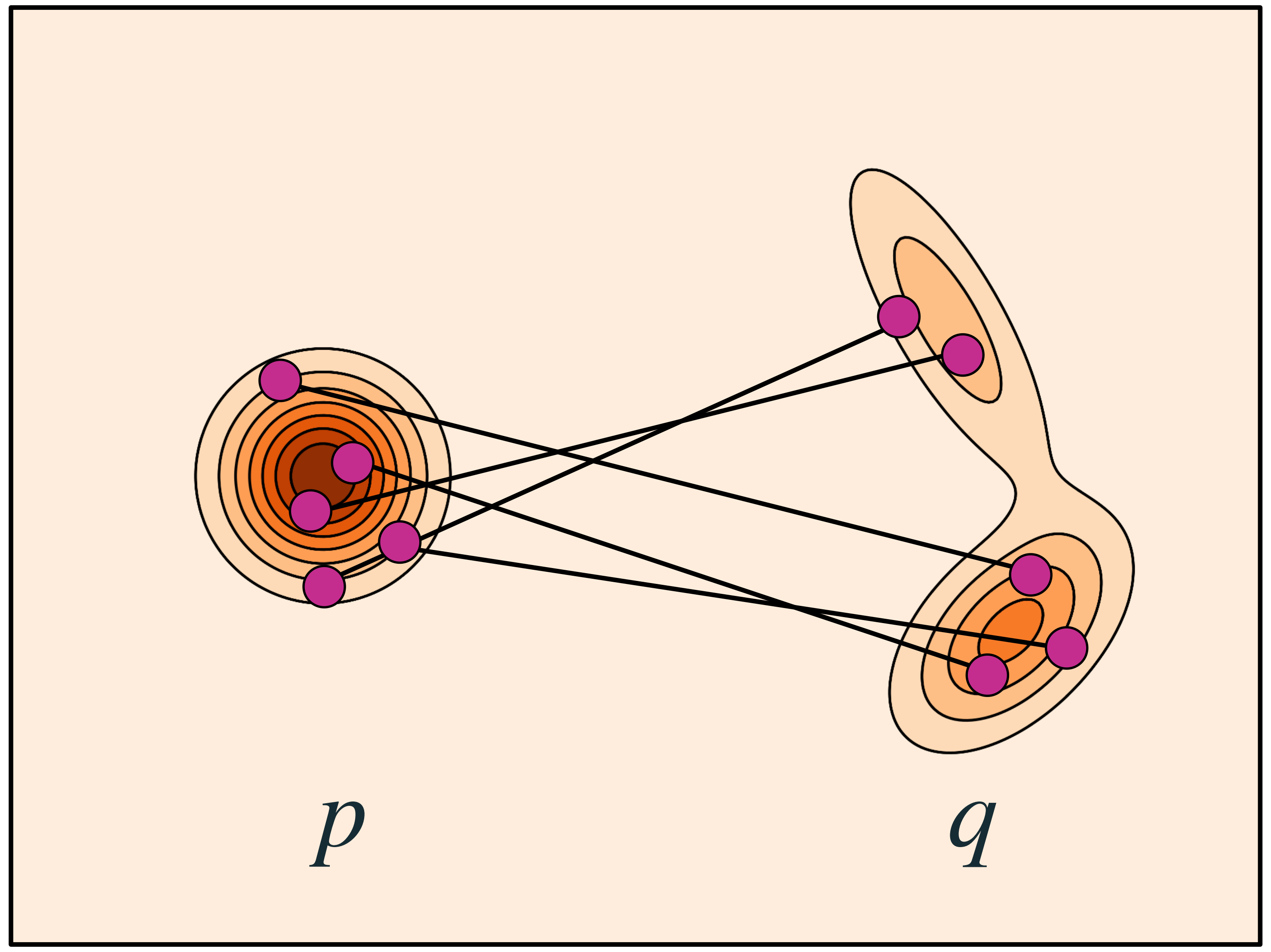


"Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow" Liu et al. (2022)
"Multisample Flow Matching: Straightening Flows with Minibatch Couplings" Pooladian et al. (2023)
"Improving and generalizing flow-based generative models with minibatch optimal transport" Tong et al. (2023)

Multisample Couplings

$$\mathrm{KE}(u_t) \leq \mathbb{E}_{\pi_{0,1}} \|X_1 - X_0\|^2$$

Kinetic Energy   Coupling cost



Use mini batch optimal transport couplings

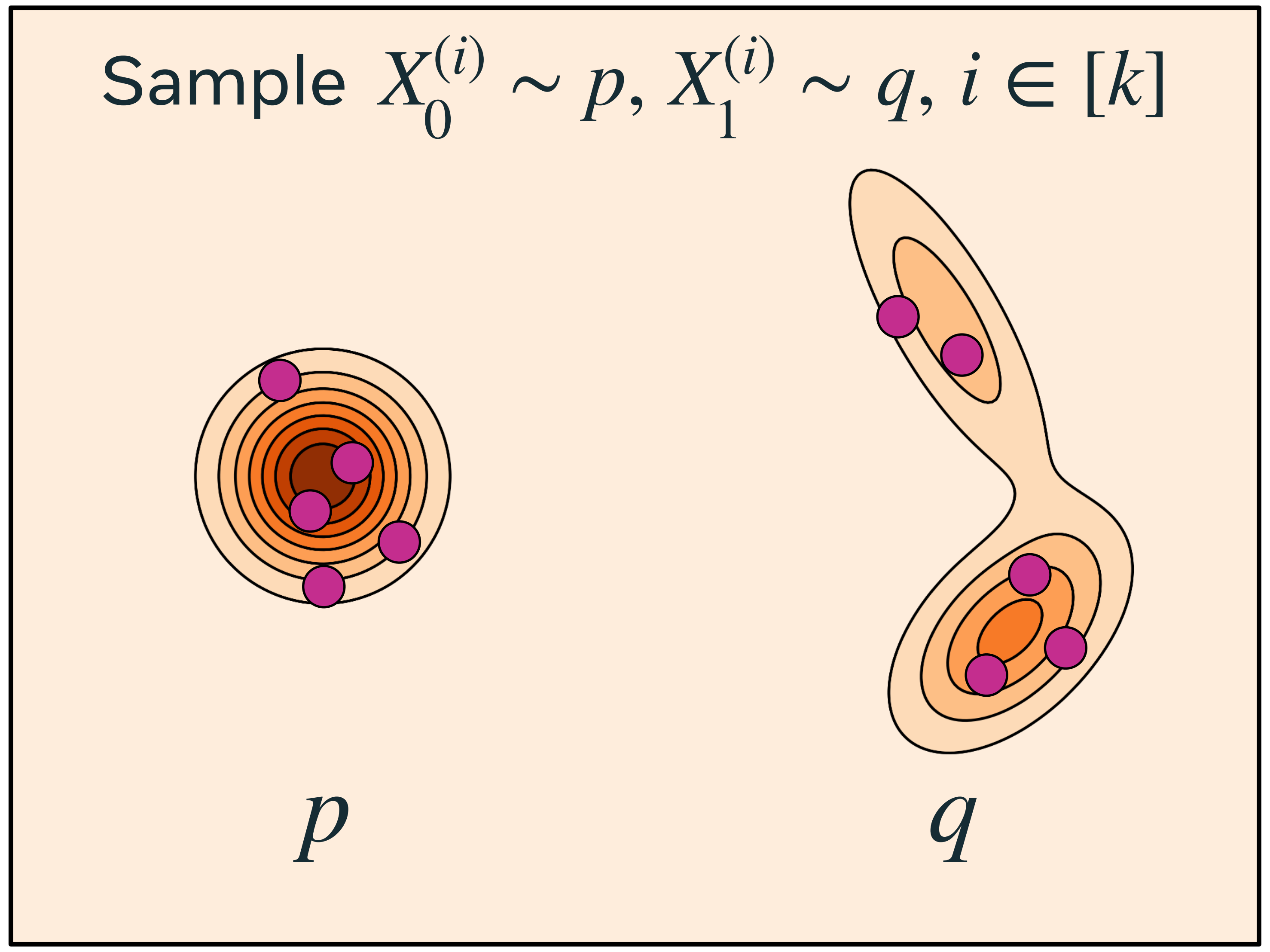
"Multisample Flow Matching: Straightening Flows with Minibatch Couplings" Pooladian et al. (2023)
"Improving and generalizing flow-based generative models with minibatch optimal transport" Tong et al. (2023)

Multisample Couplings

$$\mathrm{KE}(u_t) \le \mathbb{E}_{\pi_{0,1}} \|X_1 - X_0\|^2$$

Kinetic Energy Coupling cost

Use mini batch optimal transport couplings



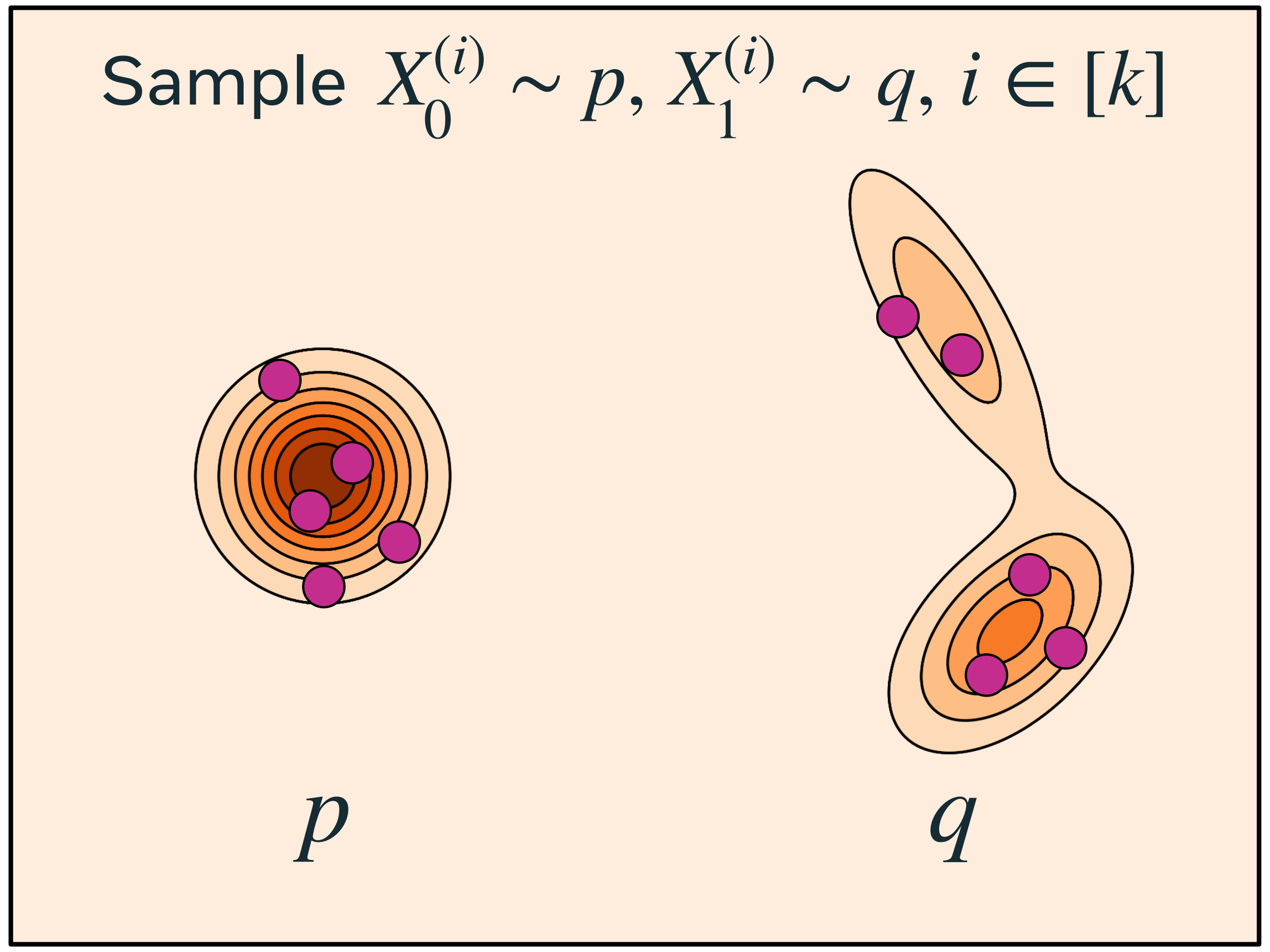
"Multisample Flow Matching: Straightening Flows with Minibatch Couplings" Pooladian et al. (2023)
"Improving and generalizing flow-based generative models with minibatch optimal transport" Tong et al. (2023)

Multisample Couplings

$$\mathrm{KE}(u_t) \leq \mathbb{E}_{\pi_{0,1}} \|X_1 - X_0\|^2$$

Kinetic Energy   Coupling cost

Use mini batch optimal transport couplings



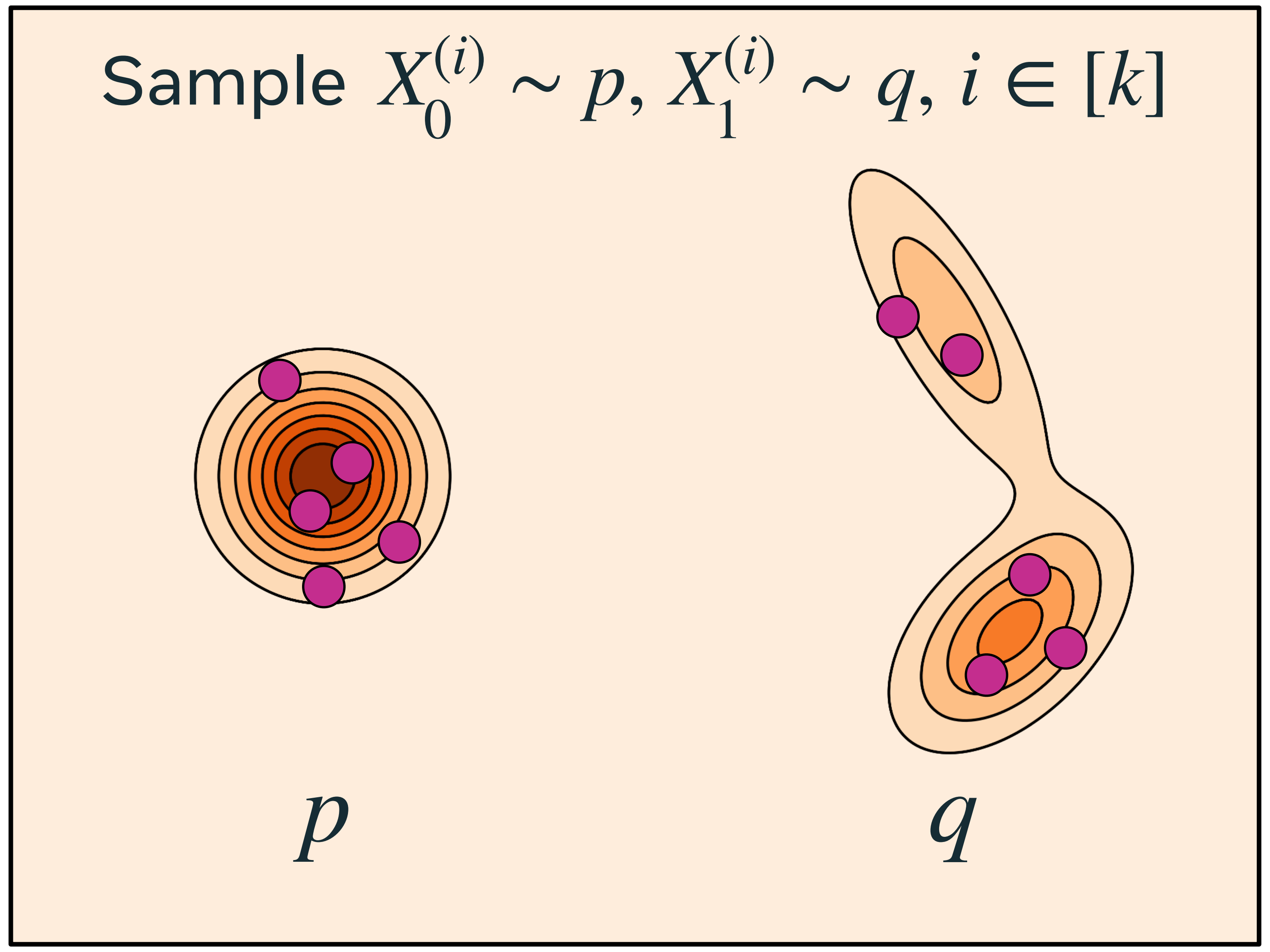
"Multisample Flow Matching: Straightening Flows with Minibatch Couplings" Pooladian et al. (2023)
"Improving and generalizing flow-based generative models with minibatch optimal transport" Tong et al. (2023)

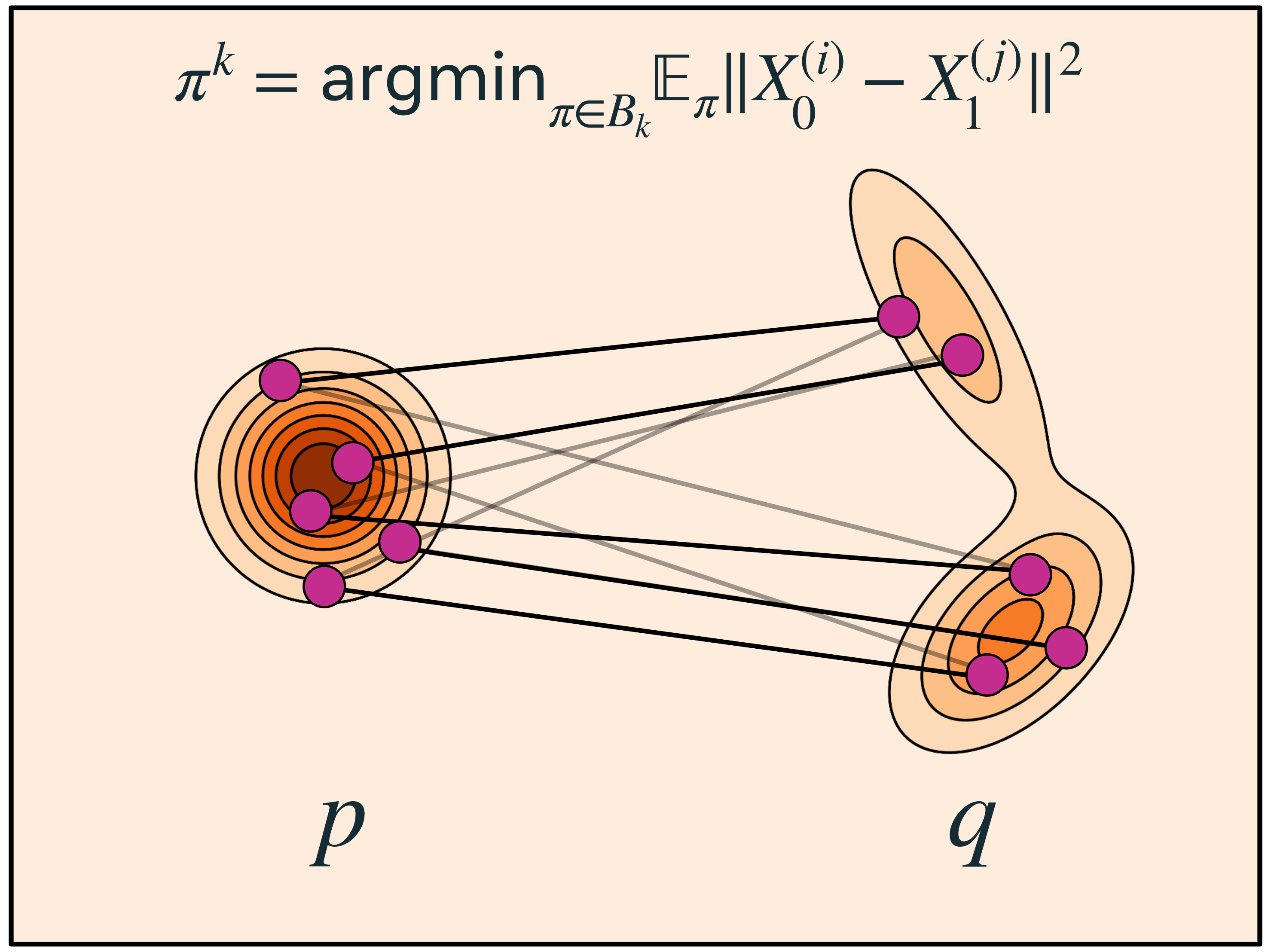
Multisample Couplings

$$\mathrm{KE}(u_t) \leq \mathbb{E}_{\pi_{0,1}} \|X_1 - X_0\|^2$$

Kinetic Energy Coupling cost

Use mini batch optimal transport couplings





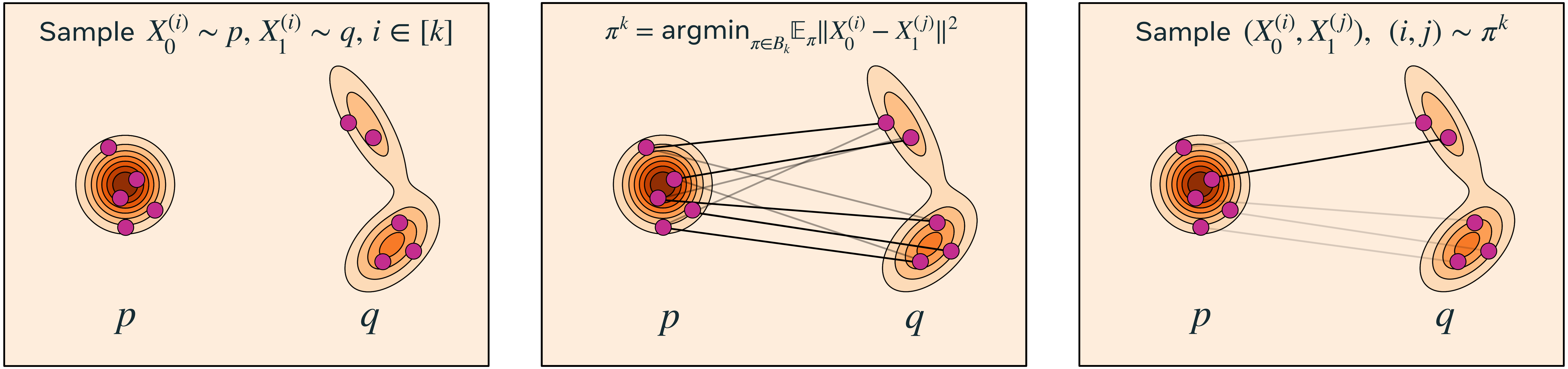
"Multisample Flow Matching: Straightening Flows with Minibatch Couplings" Pooladian et al. (2023)
"Improving and generalizing flow-based generative models with minibatch optimal transport" Tong et al. (2023)

Multisample Couplings

$$\mathrm{KE}(u_t) \leq \mathbb{E}_{\pi_{0,1}} \|X_1 - X_0\|^2$$

Kinetic Energy   Coupling cost

Use mini batch optimal transport couplings



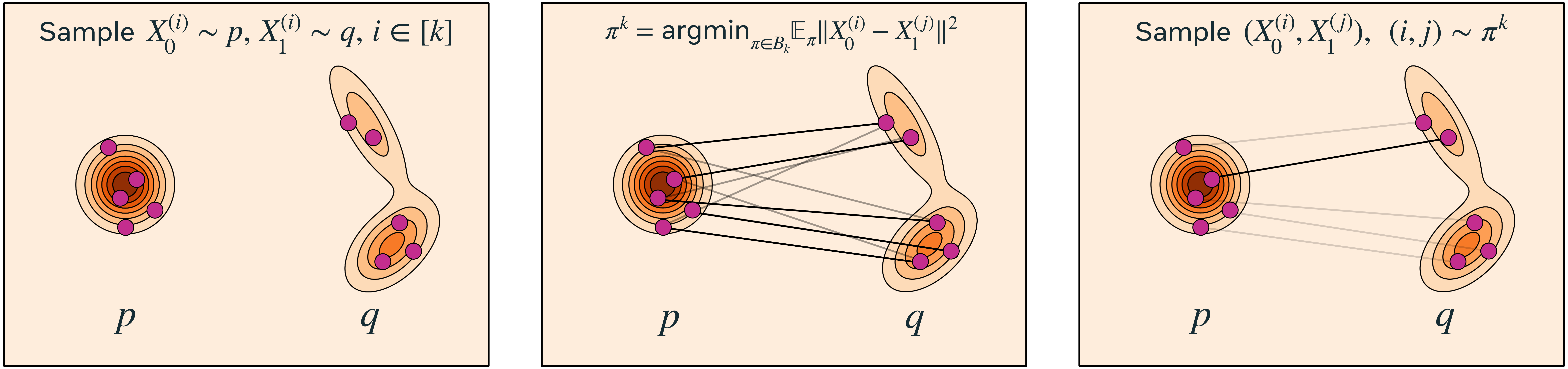
"Multisample Flow Matching: Straightening Flows with Minibatch Couplings" Pooladian et al. (2023)
"Improving and generalizing flow-based generative models with minibatch optimal transport" Tong et al. (2023)

Multisample Couplings

$$\mathrm{KE}(u_t) \leq \mathbb{E}_{\pi_{0,1}} \|X_1 - X_0\|^2$$

Kinetic Energy Coupling cost

When $k = 1 \rightarrow \pi_{0,1} = p(X_0)q(X_1)$



"Multisample Flow Matching: Straightening Flows with Minibatch Couplings" Pooladian et al. (2023)
"Improving and generalizing flow-based generative models with minibatch optimal transport" Tong et al. (2023)

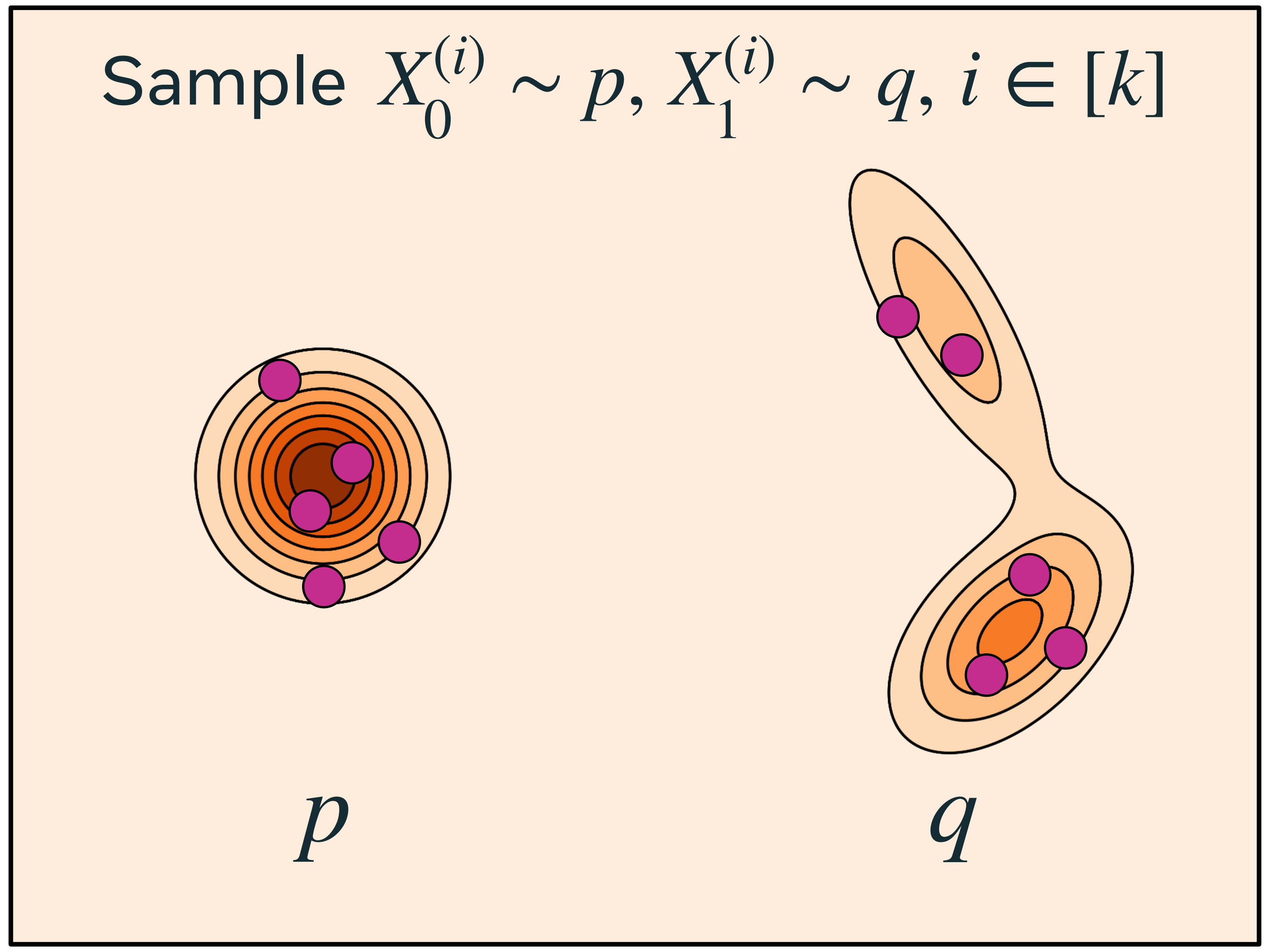
Multisample Couplings

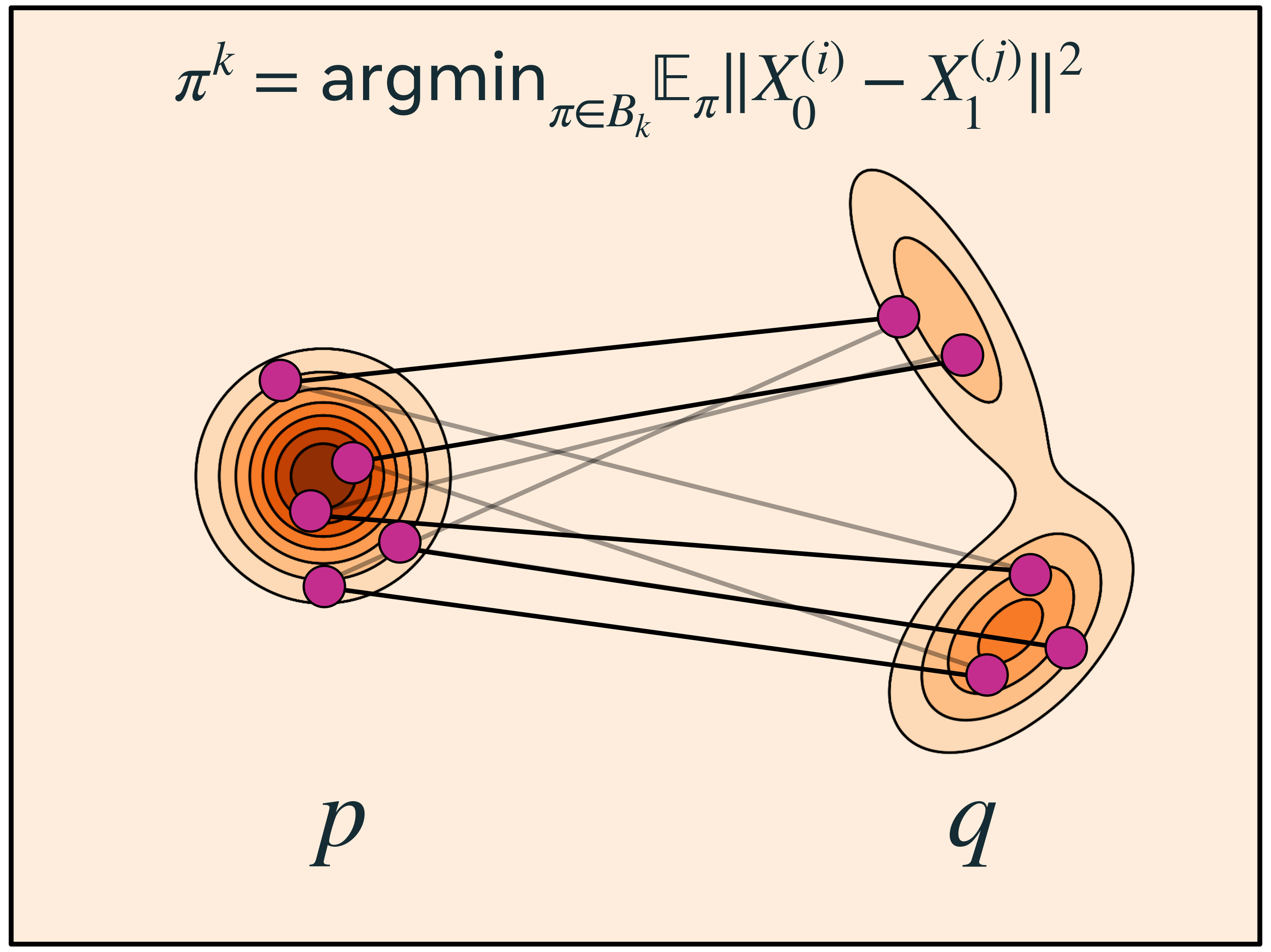
FM with cond-OT is not marginal OT!

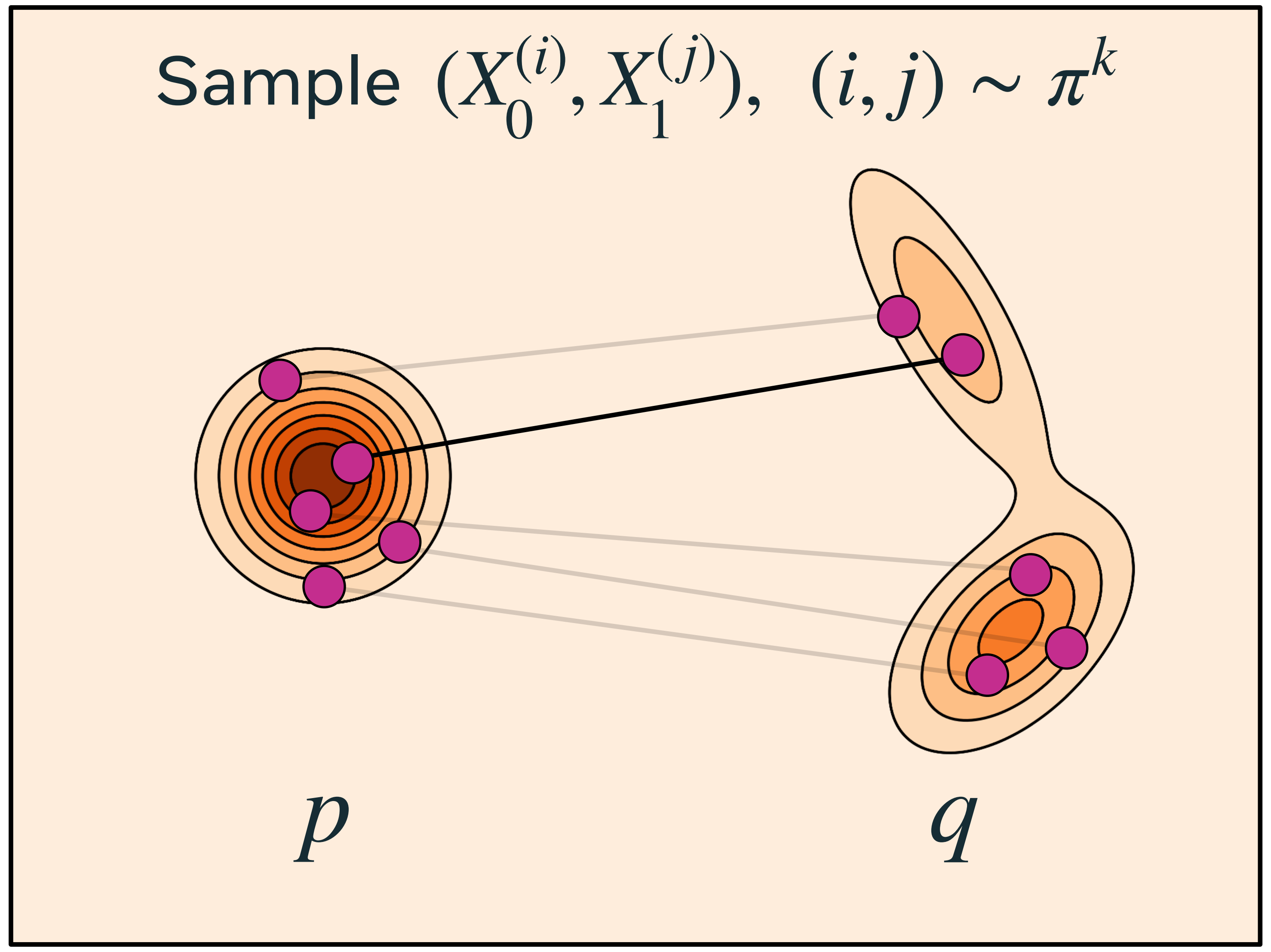
$$\mathrm{KE}(u_t) \leq \mathbb{E}_{\pi_{0,1}} \|X_1 - X_0\|^2$$

Kinetic Energy   Coupling cost

When $k \to \infty$, u_t generates the OT map

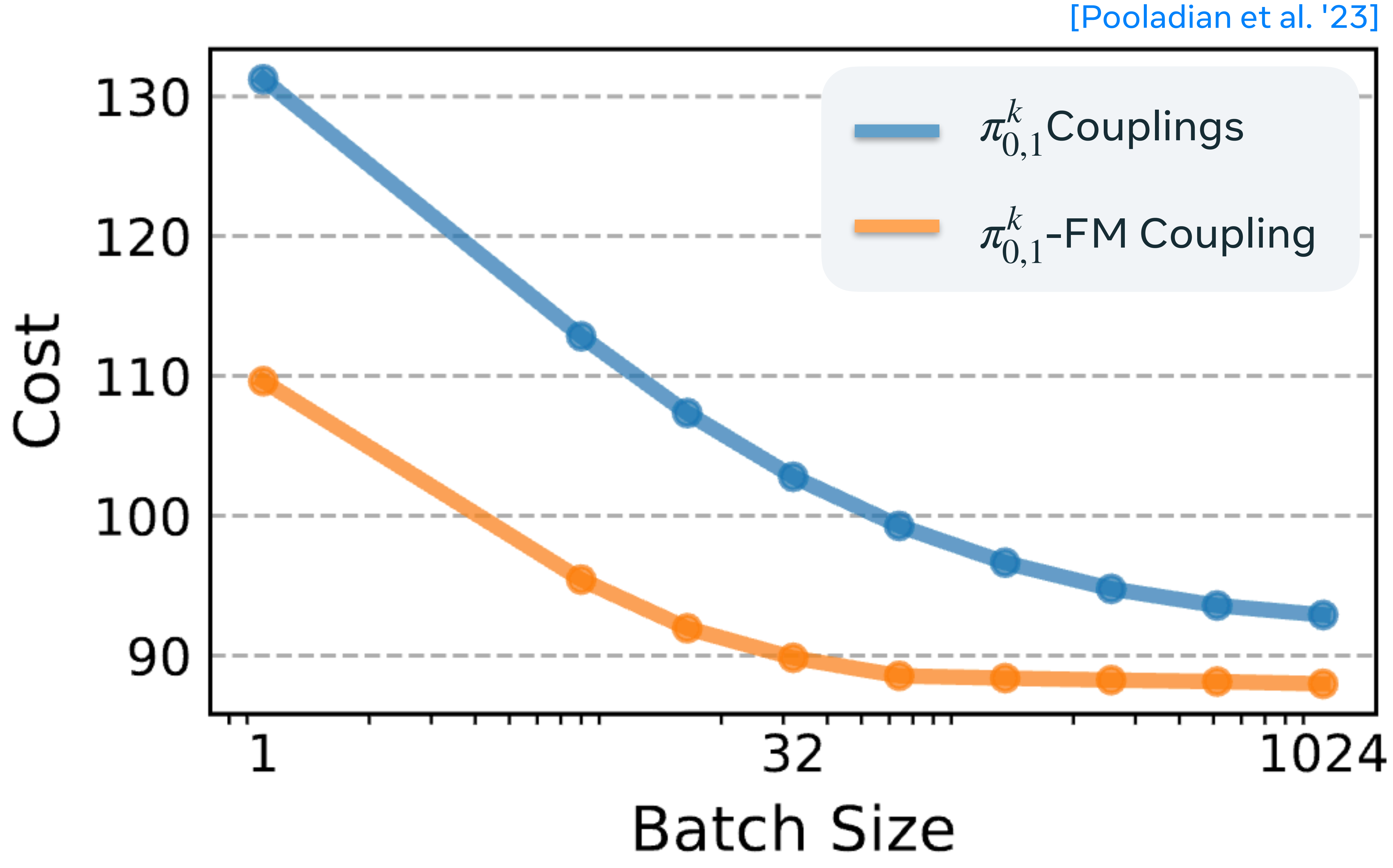






"Multisample Flow Matching: Straightening Flows with Minibatch Couplings" Pooladian et al. (2023)
"Improving and generalizing flow-based generative models with minibatch optimal transport" Tong et al. (2023)

Multisample Couplings



- High dimensions - minor improvement in sampling speed compared to tailored samplers.

- Shows promise in lower dimensional problems for scientific applications (e.g. protein backbone design [Bose et al. '23]).

"SE(3)-Stochastic Flow Matching for Protein Backbone Generation" Bose et al. (2023)
"Multisample Flow Matching: Straightening Flows with Minibatch Couplings" Pooladian et al. (2023)
"Improving and generalizing flow-based generative models with minibatch optimal transport" Tong et al. (2023)

Data Couplings

Paired data:

"I2SB: Image-to-Image Schrödinger Bridge" Liu et al. (2023)

"Stochastic interpolants with data-dependent couplings" Albergo et al. (2024)

"Simulation-Free Training of Neural ODEs on Paired Data" Kim et al. (2024)

Multisample couplings:

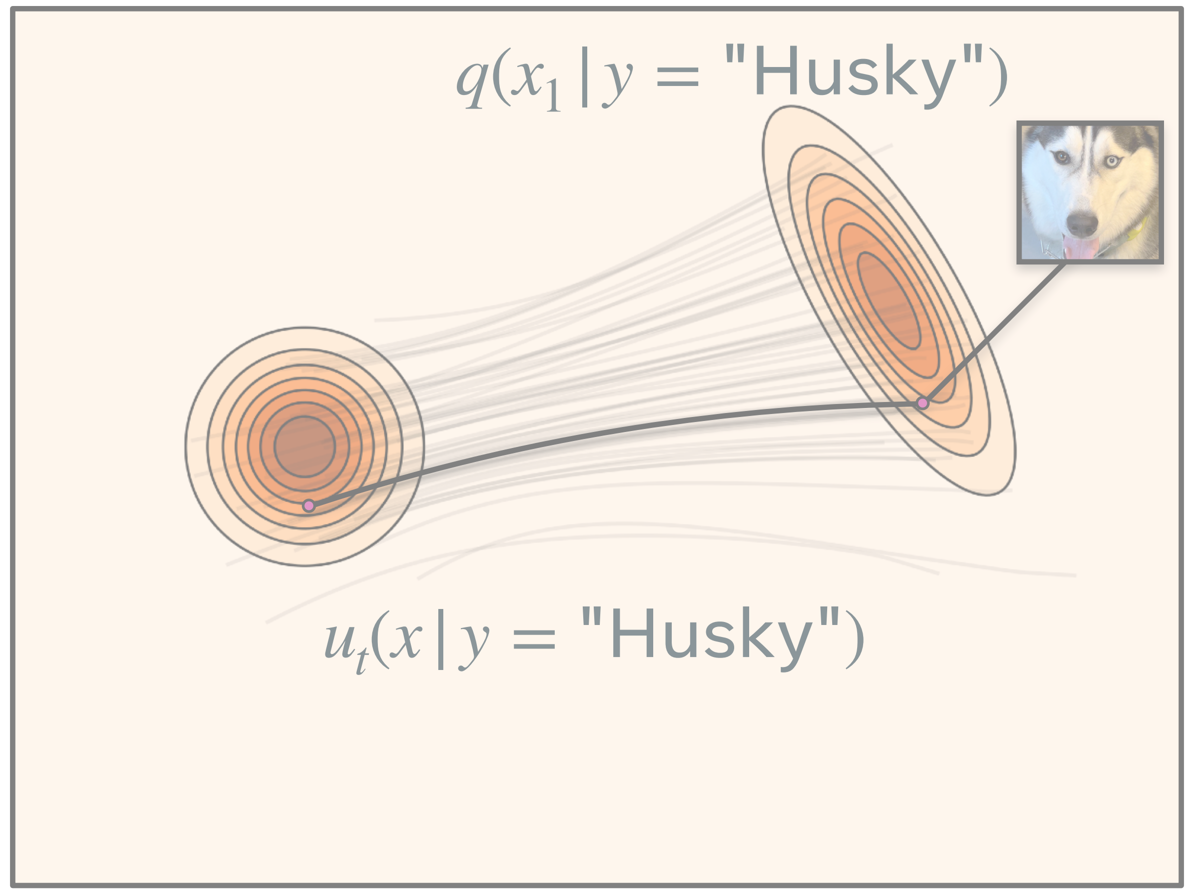
"Multisample Flow Matching: Straightening Flows with Minibatch Couplings" Pooladian et al. (2023)

"Improving and generalizing flow-based generative models with minibatch optimal transport" Tong et al. (2023)

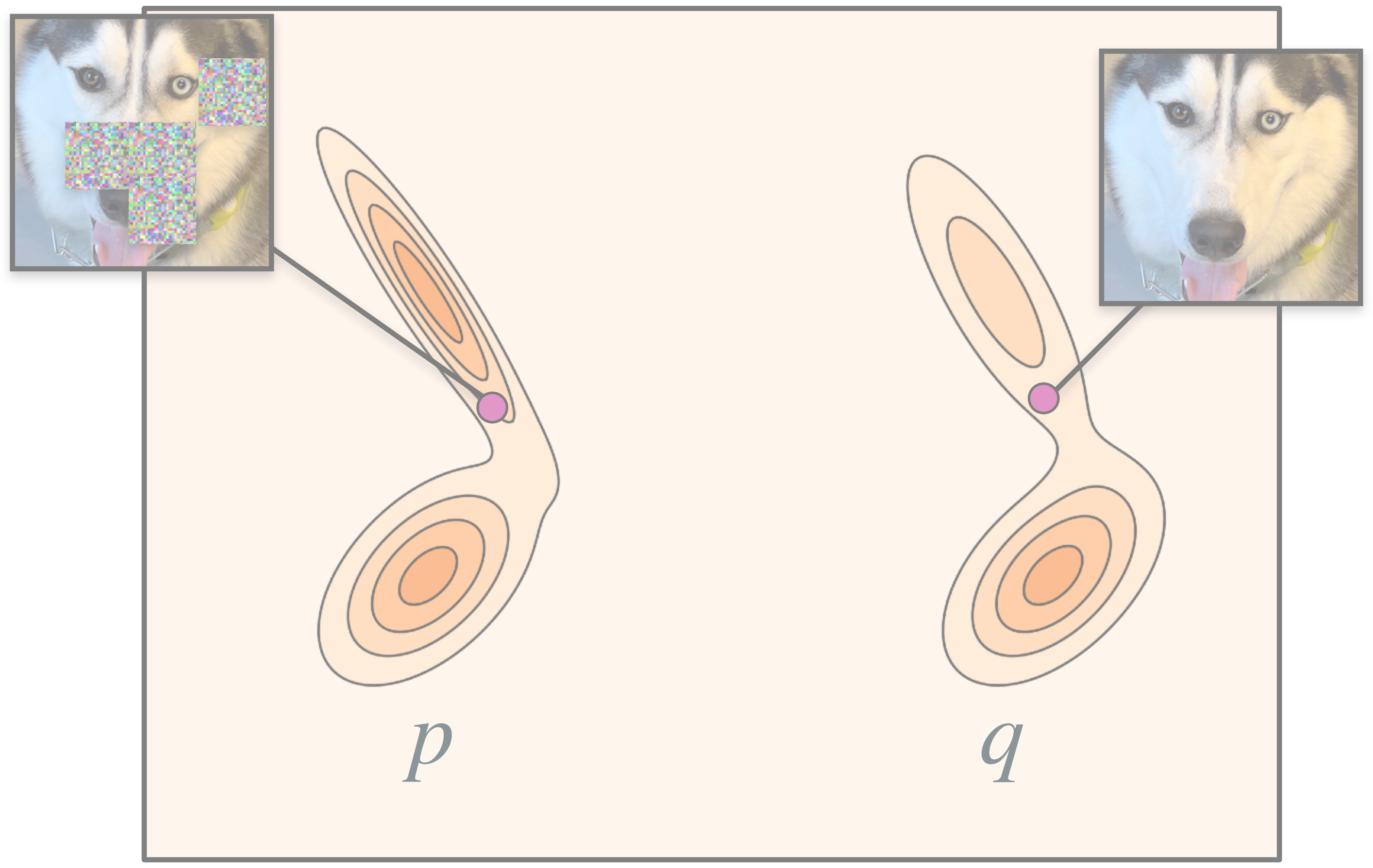
"SE(3)-Stochastic Flow Matching for Protein Backbone Generation" Bose et al. (2023)

"Sequence-Augmented SE(3)-Flow Matching For Conditional Protein Backbone Generation" Huguet et al. (2024)

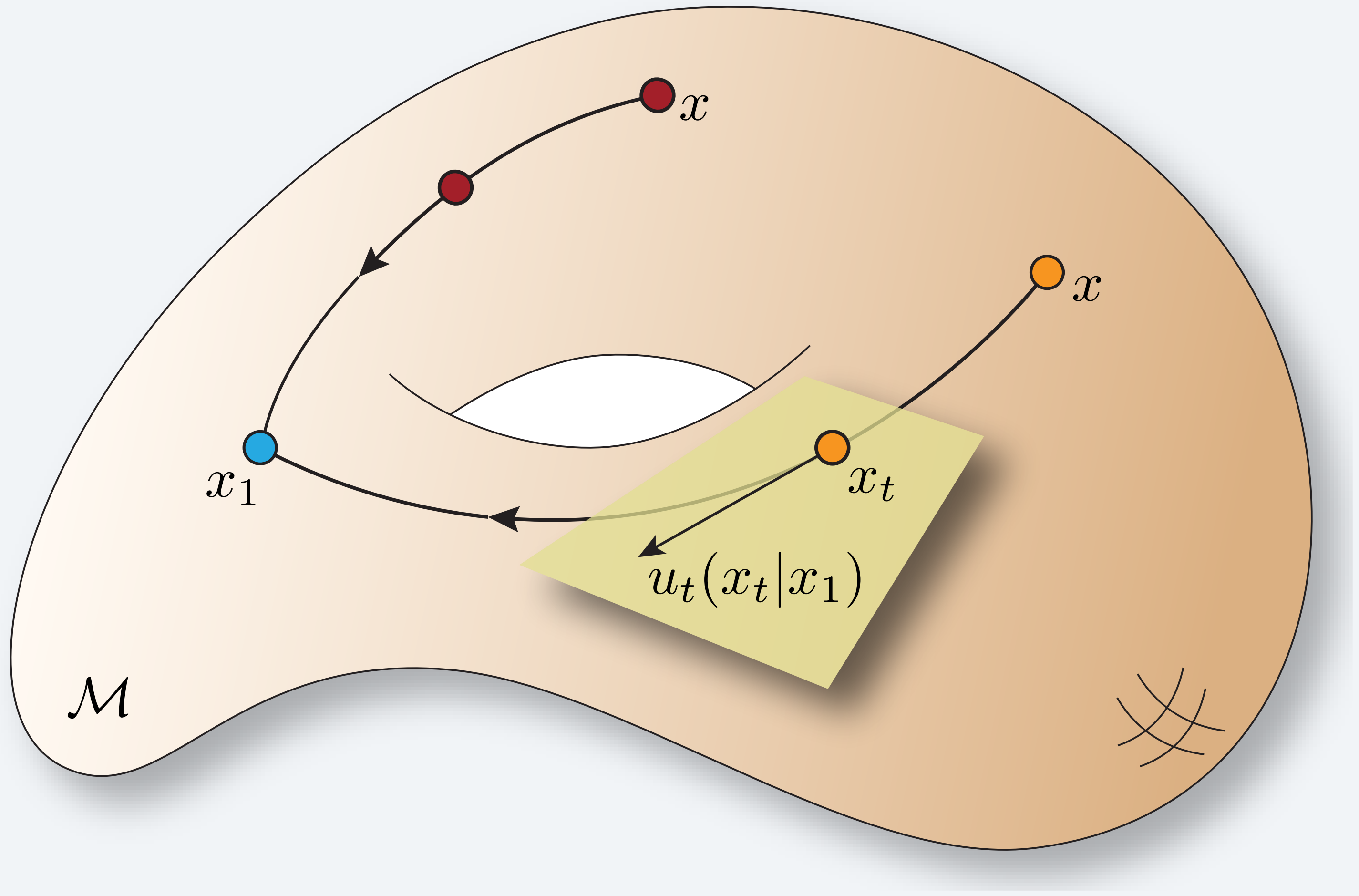
Conditioning and Guidance



Data Couplings

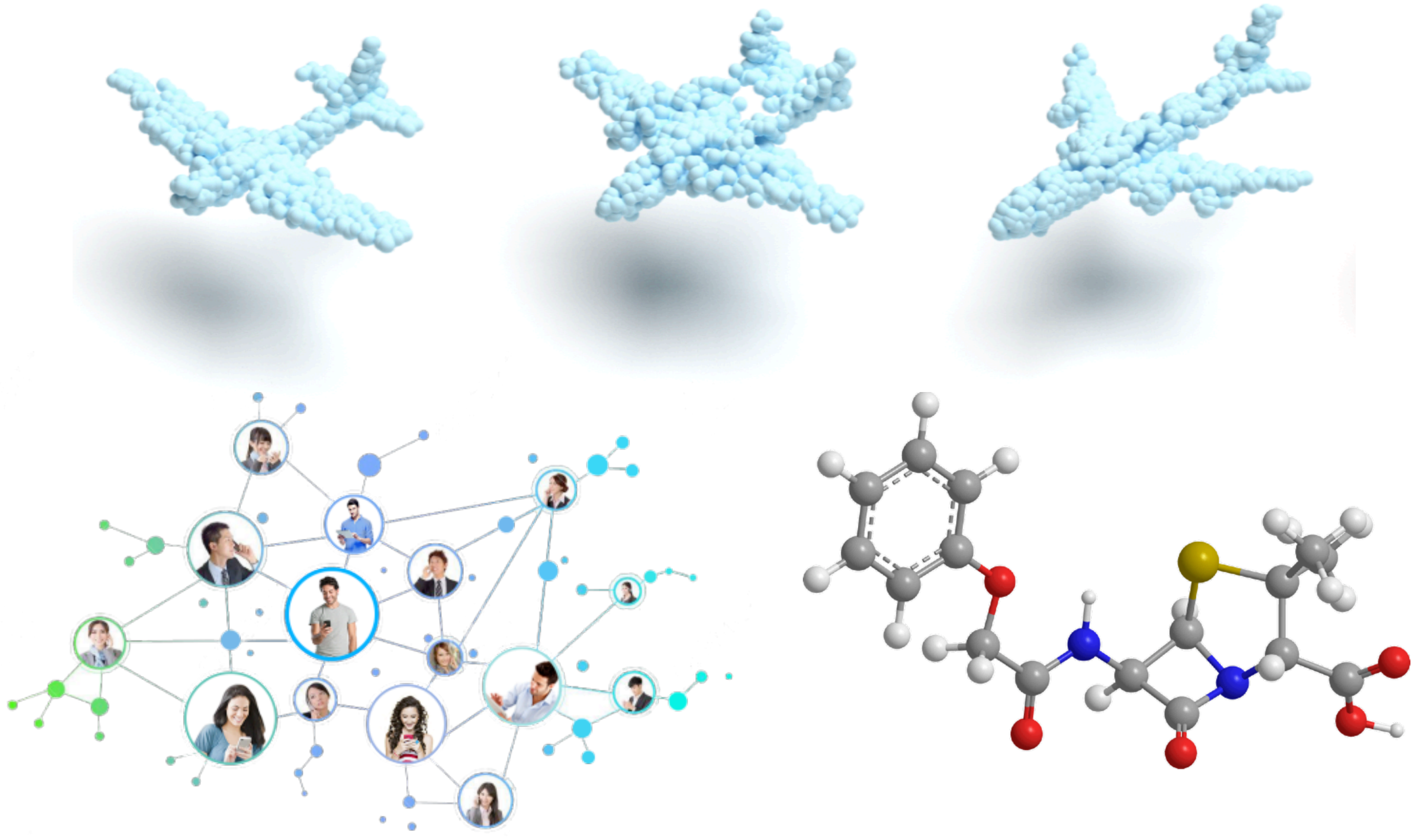


Geometric Flow Matching



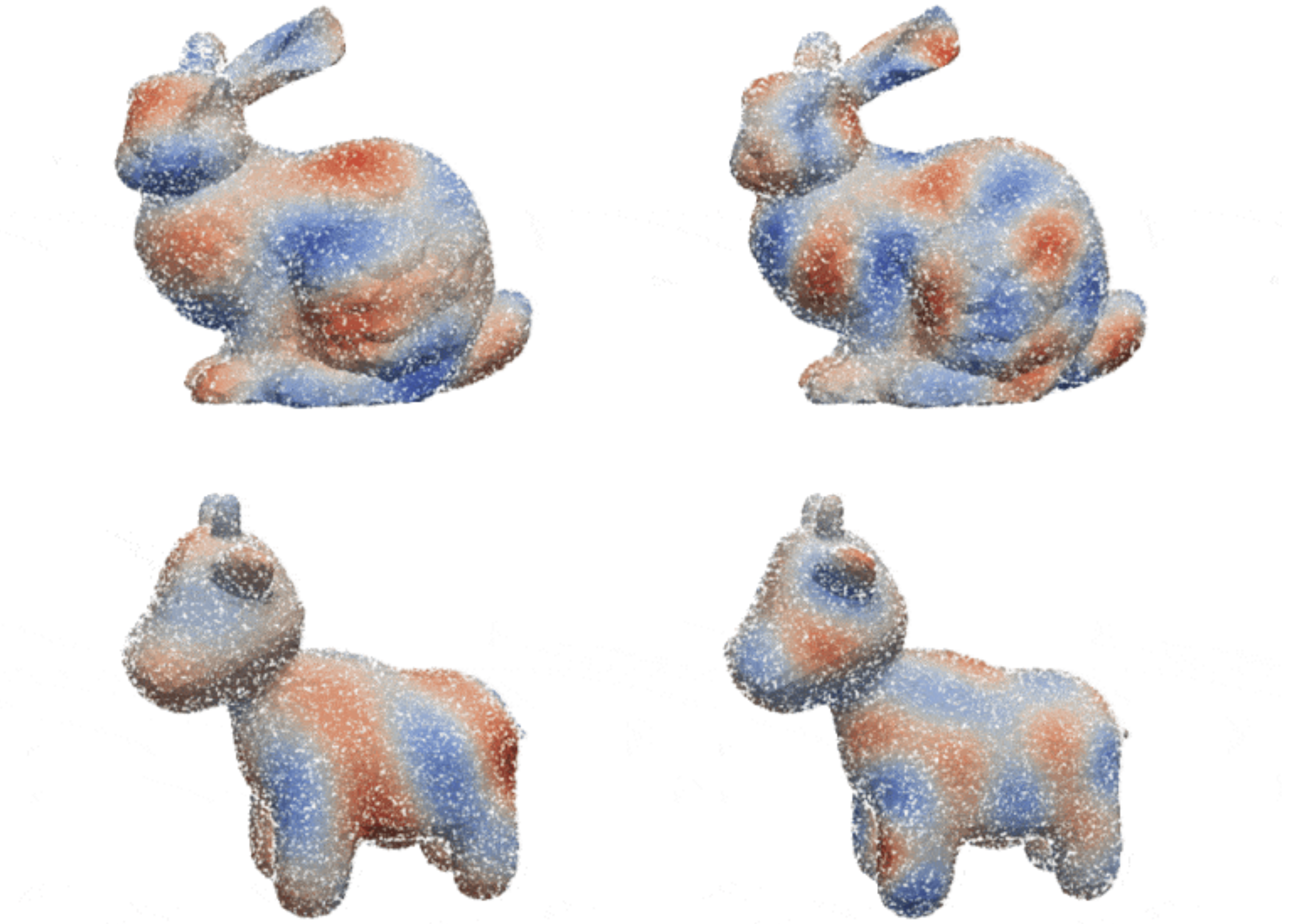
Geometric Flow Matching

Data with Symmetries



- Equivariant flows → invariant densities
- Alignment couplings

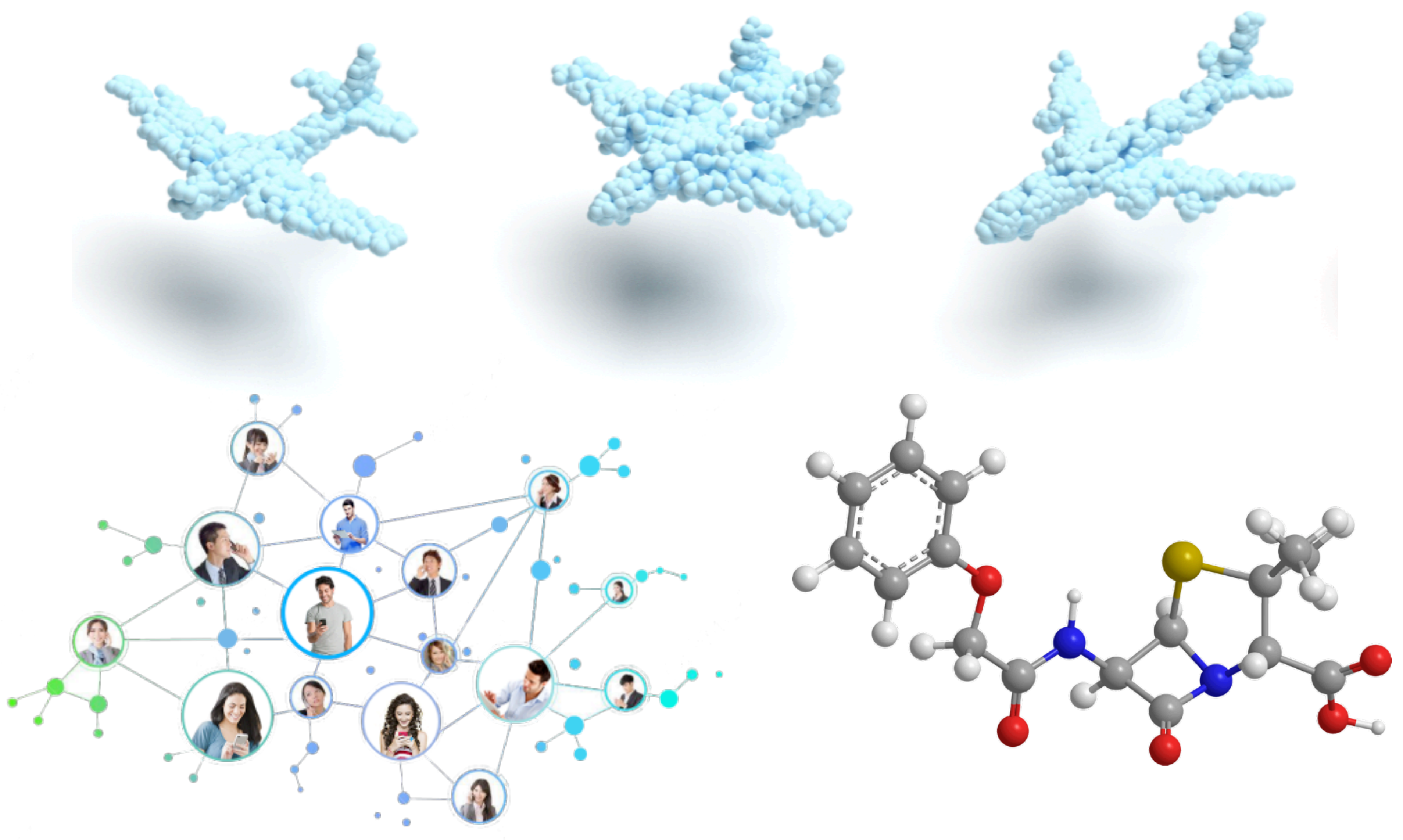
Riemannian Manifolds



- Simulation free on simple manifolds
- General geometries

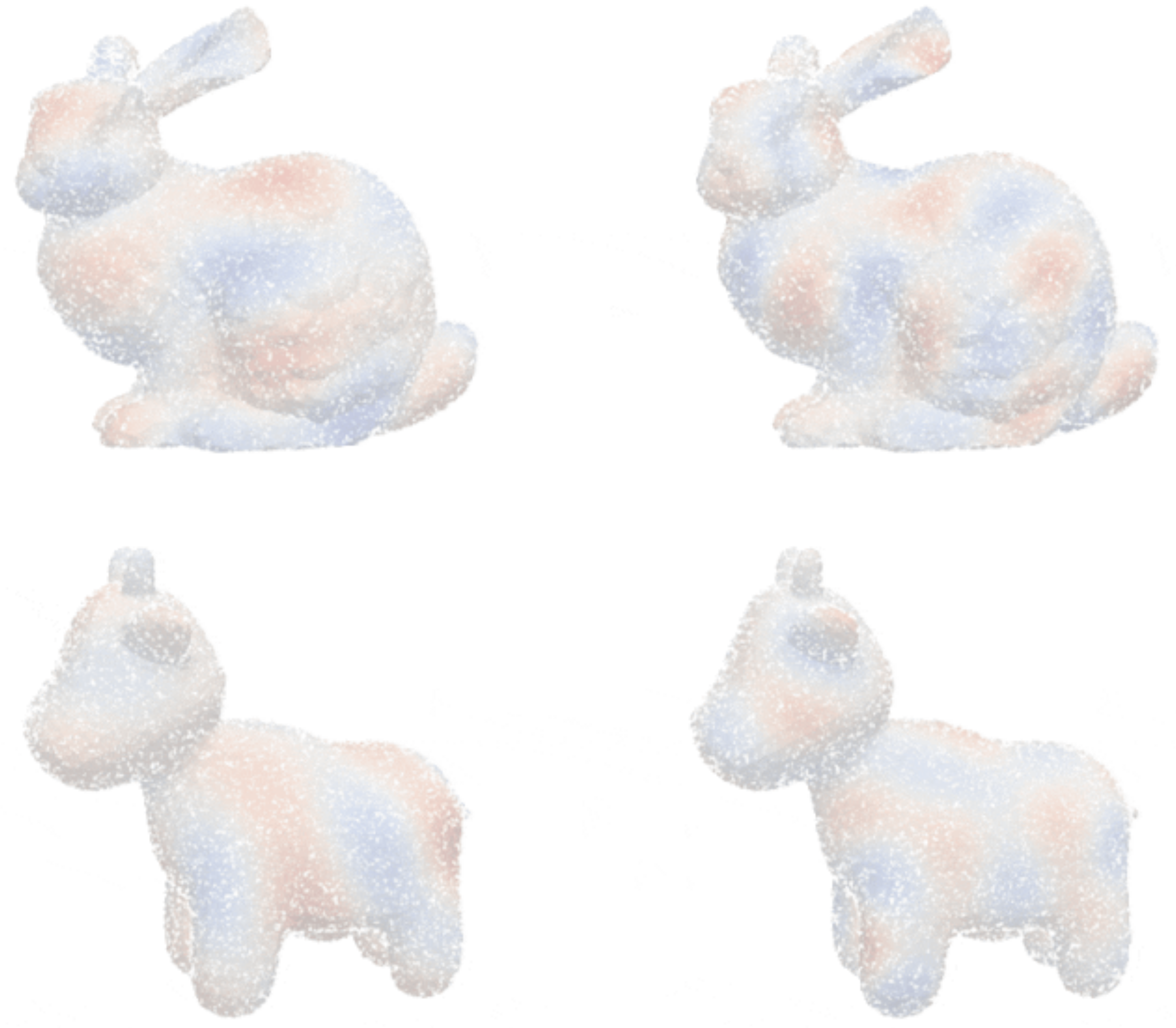
Geometric Flow Matching

Data with Symmetries



- Equivariant flows → invariant densities
- Alignment couplings

Riemannian Manifolds

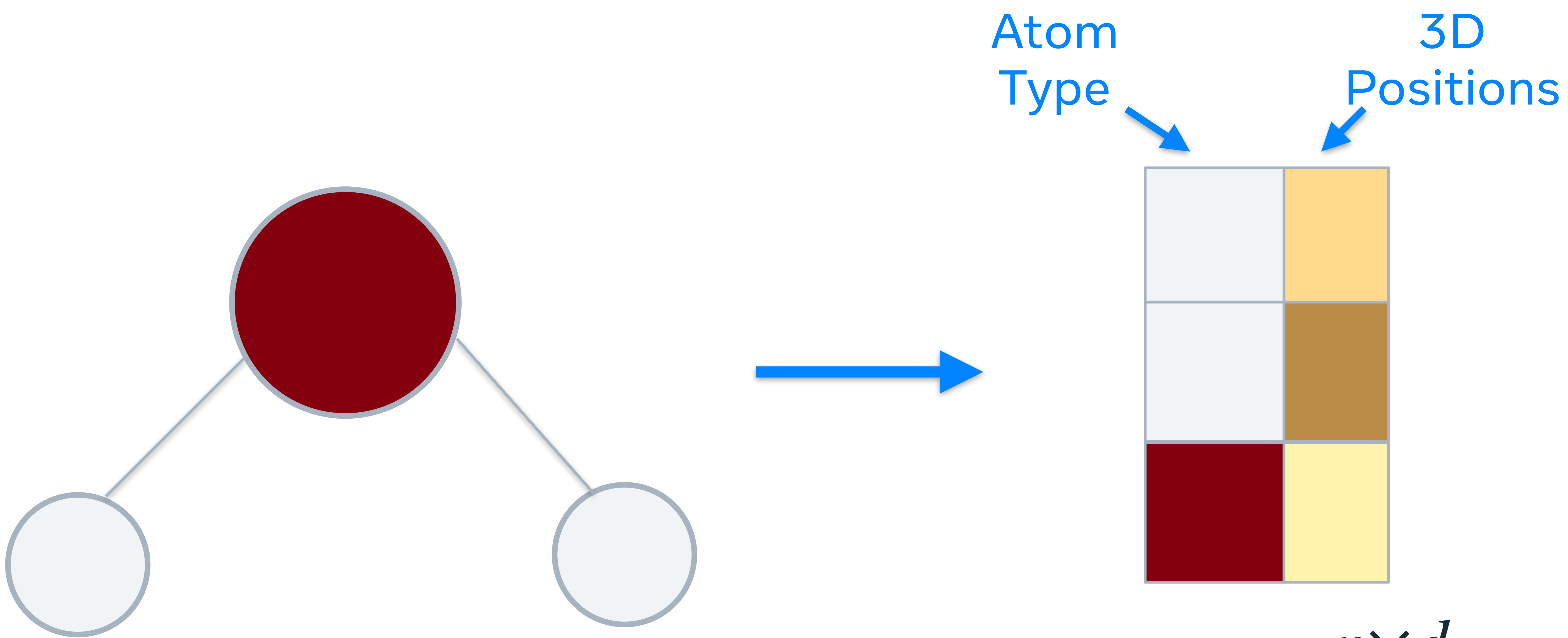


- Simulation free on simple manifolds
- General geometries

Data with Symmetries

Data

- Sets
- Graphs
- Hyper Graphs

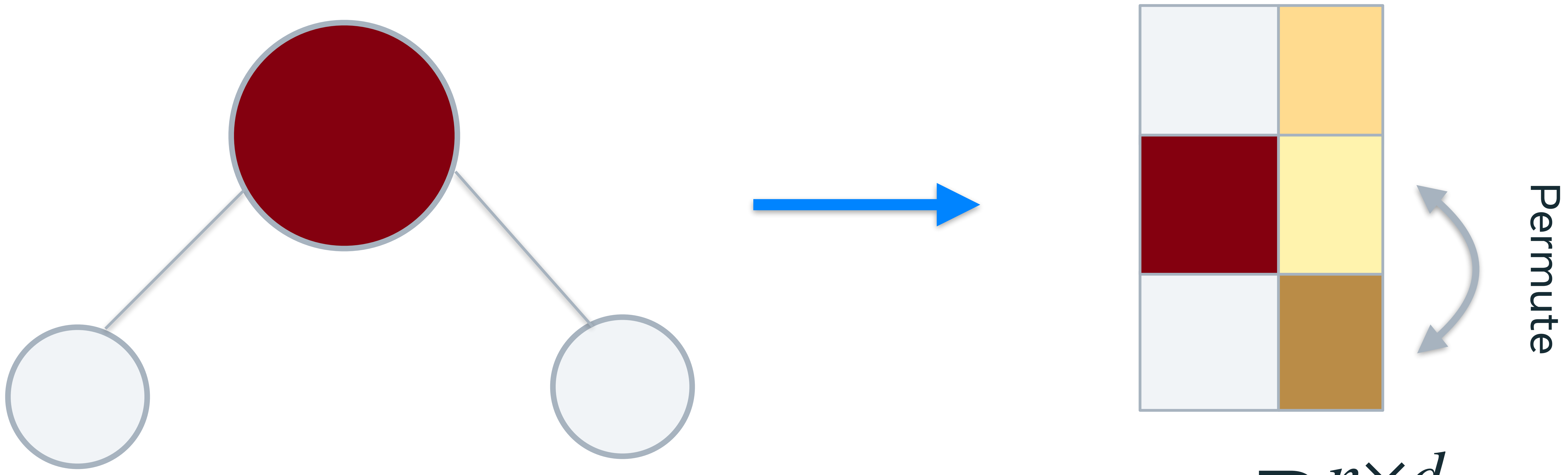


$$x \in \mathbb{R}^{n \times d}$$

Data with Symmetries

Data

- Sets
- Graphs
- Hyper Graphs



$x \in \mathbb{R}^{n \times d}$

Data with Symmetries

Symmetries are transformations under which an object is invariant.

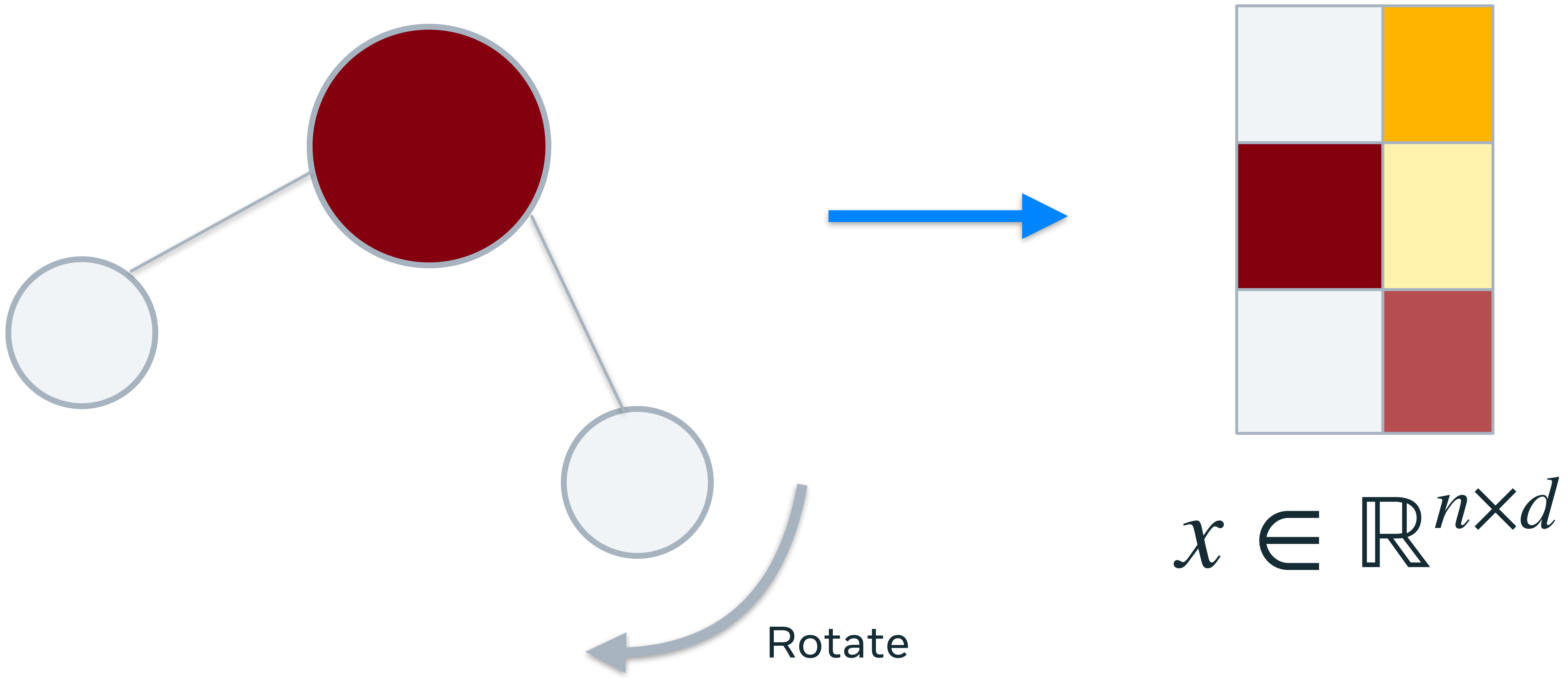
Data

- Sets
- Graphs
- Hyper Graphs

Symmetries

- S_n - permutations
- $SO(n)$ - rotations
- $SE(3)$ - rigid motions

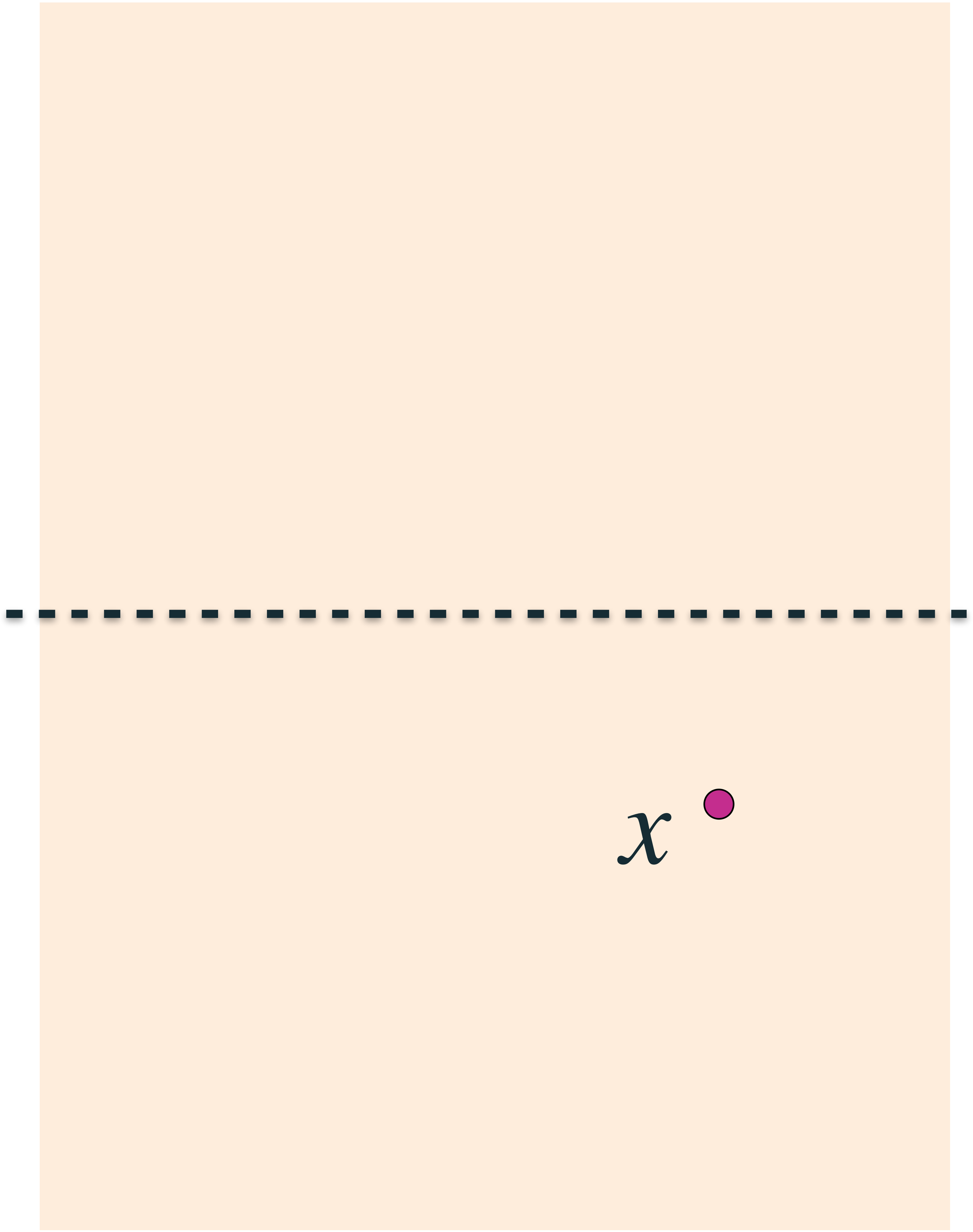
(rotations, reflections, translations)



Invariant densities

Symmetry Group G

Example: Reflection

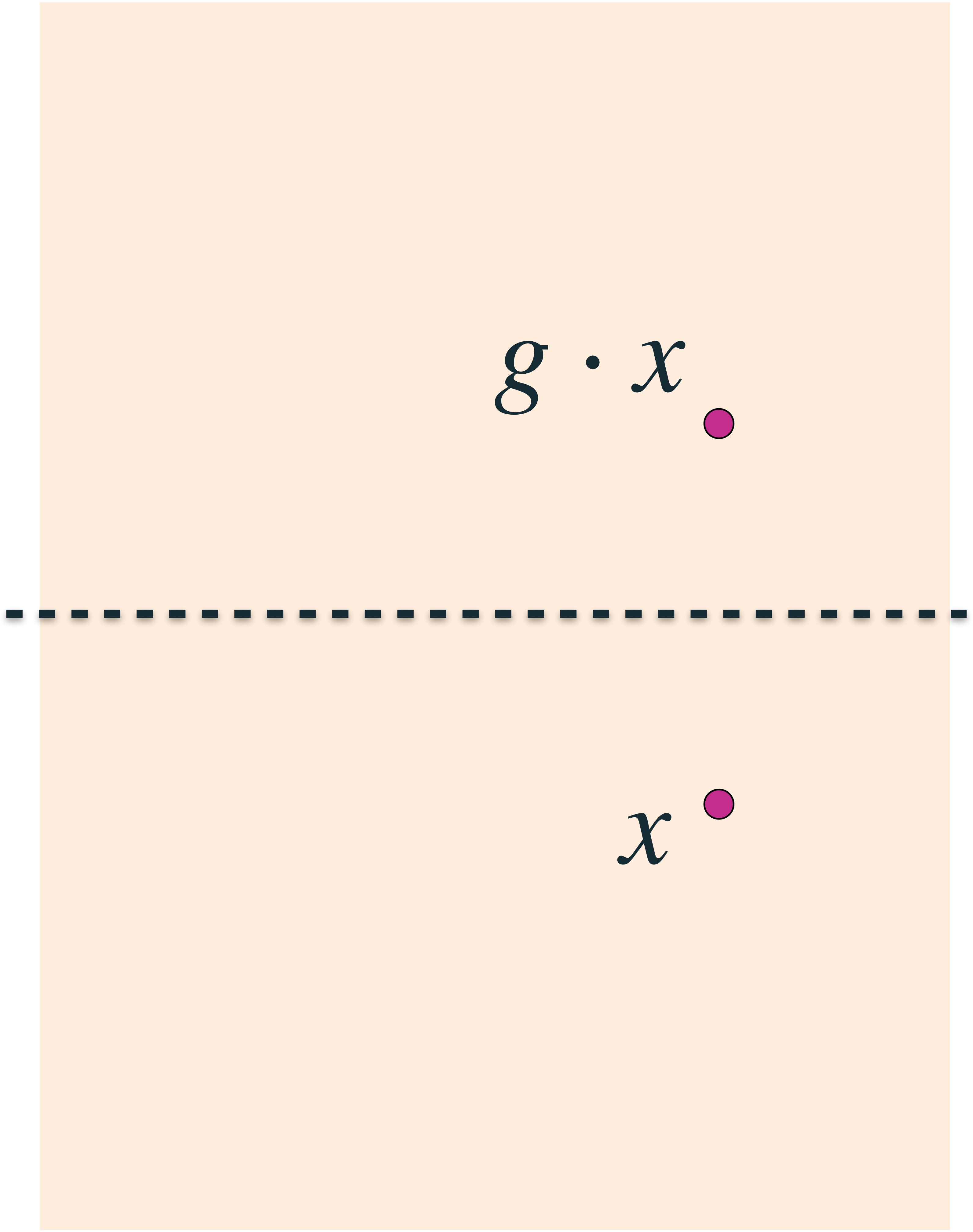


Invariant densities

Symmetry Group G

$$G = \{g, e\}$$

Example: Reflection



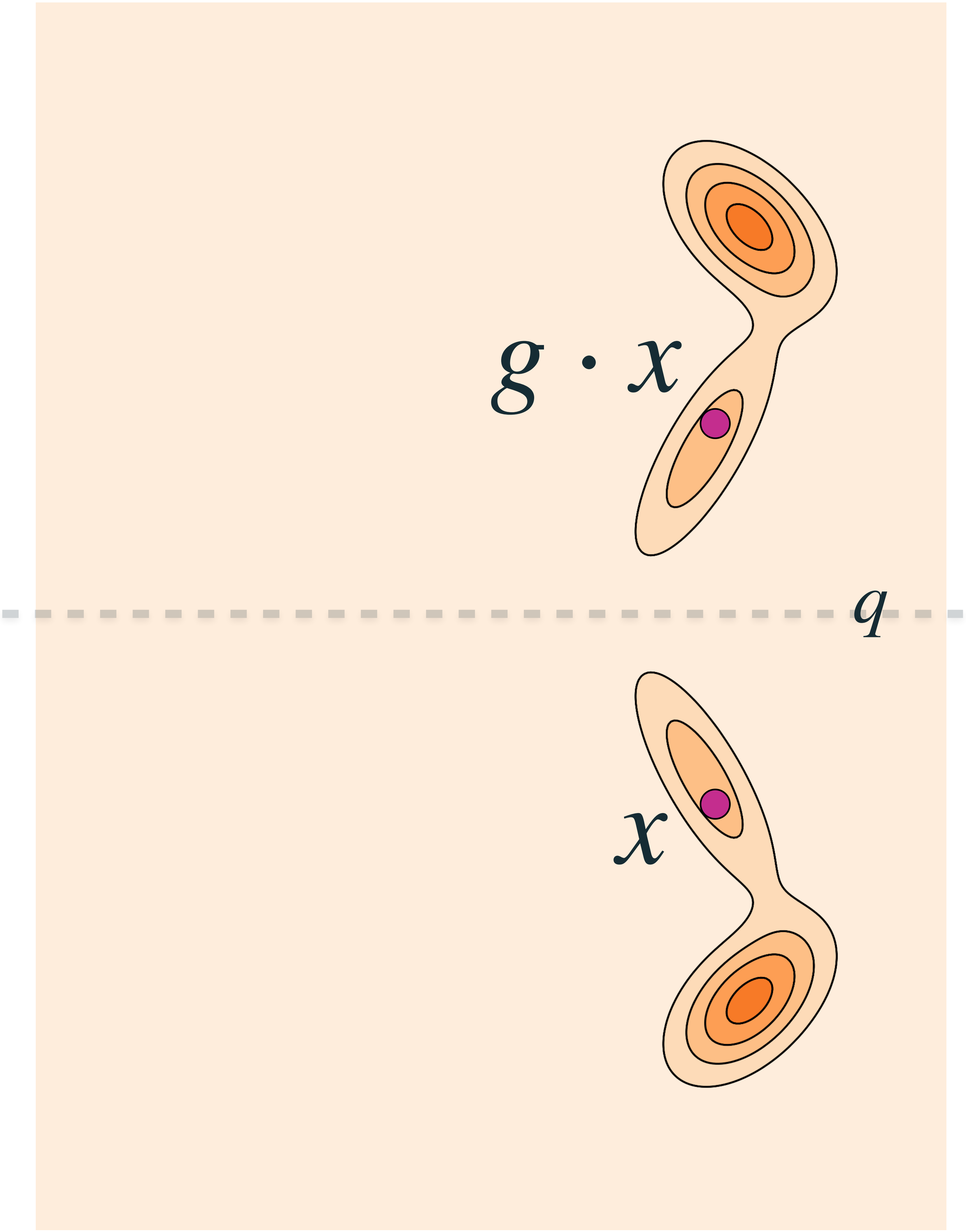
Invariant densities

Symmetry Group G

$$G = \{g, e\}$$

Invariant Density $\quad q(g \cdot x) = q(x)$

Example: Reflection



Equivariant Flows

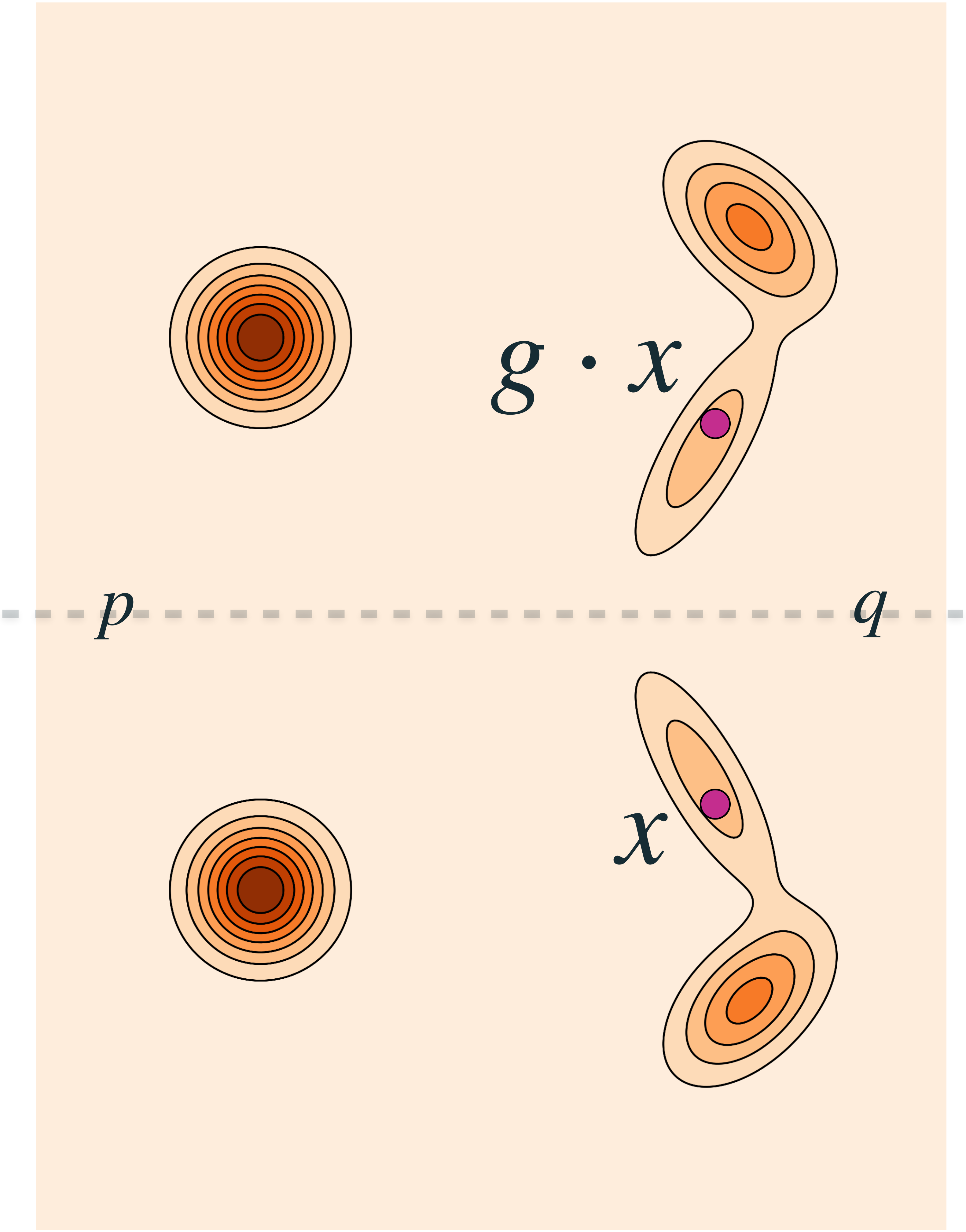
Symmetry Group G

$$G = \{g, e\}$$

Invariant Density

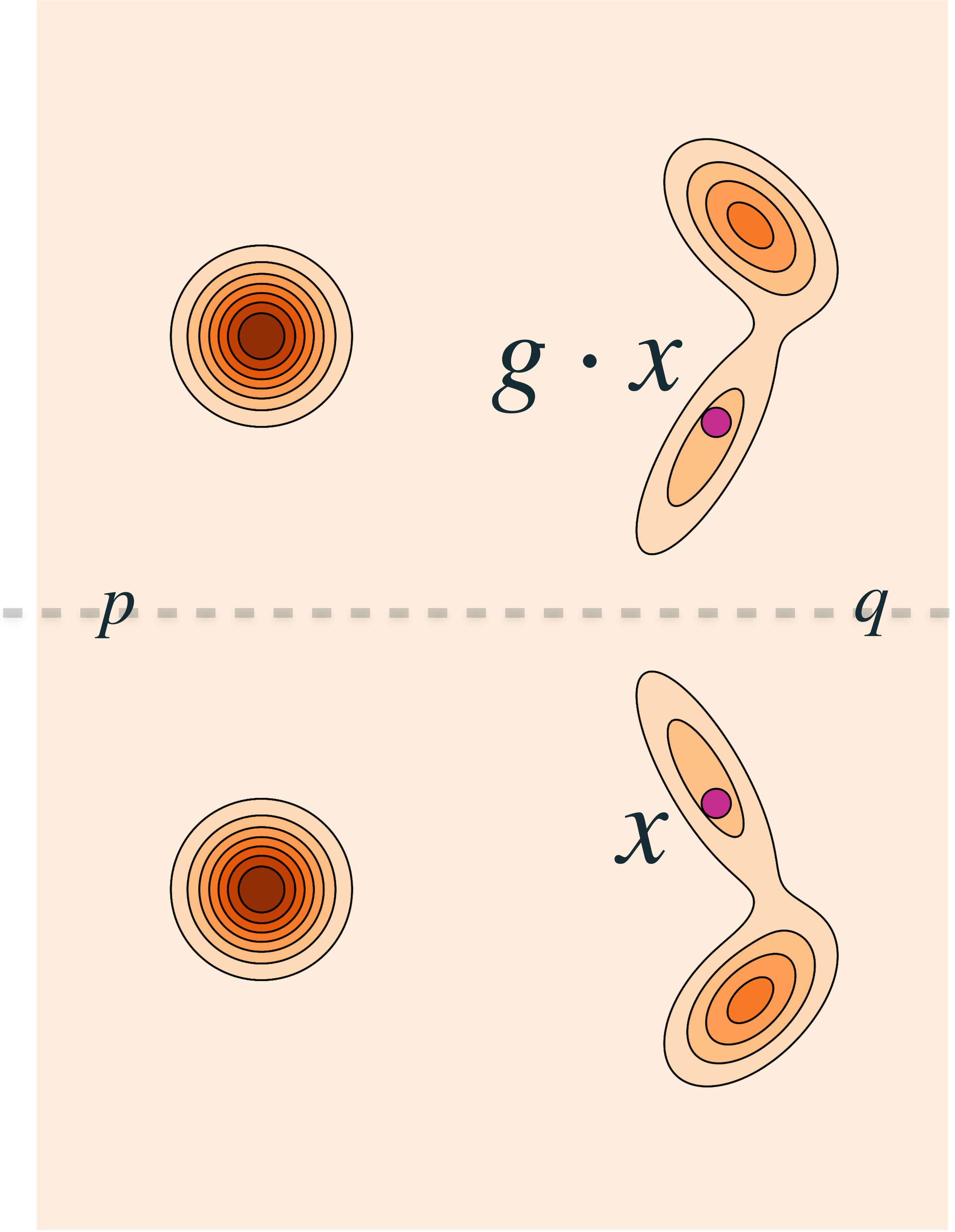
$$q(g \cdot x) = q(x)$$

Example: Reflection



Equivariant Flows

Example: Reflection

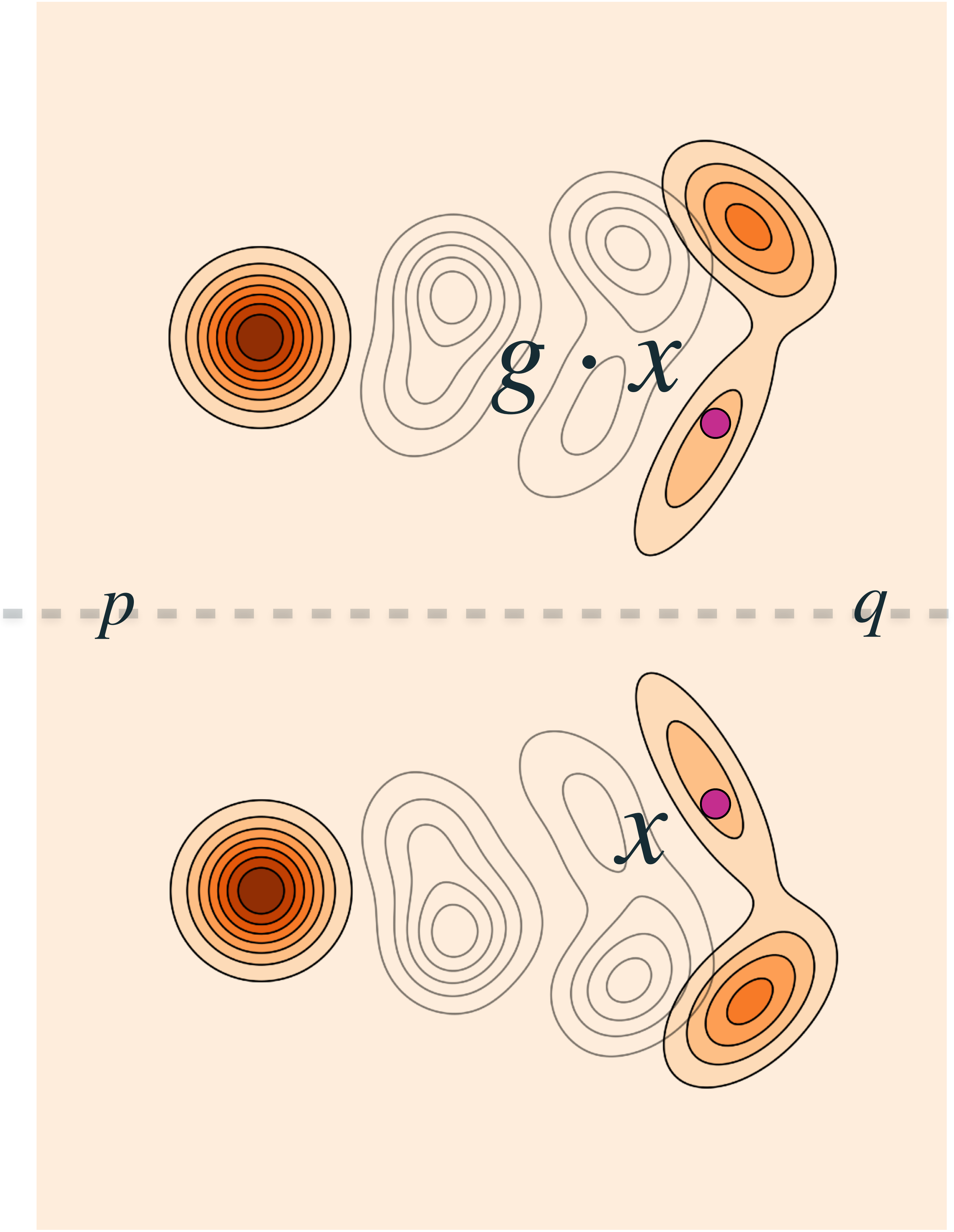


$$p_t(g \cdot x) = p_t(x)$$

Invariant probability path

Equivariant Flows

Example: Reflection



$$p_t(g \cdot x) = p_t(x)$$

Invariant probability path

Equivariant Flows

Equivariant Flow

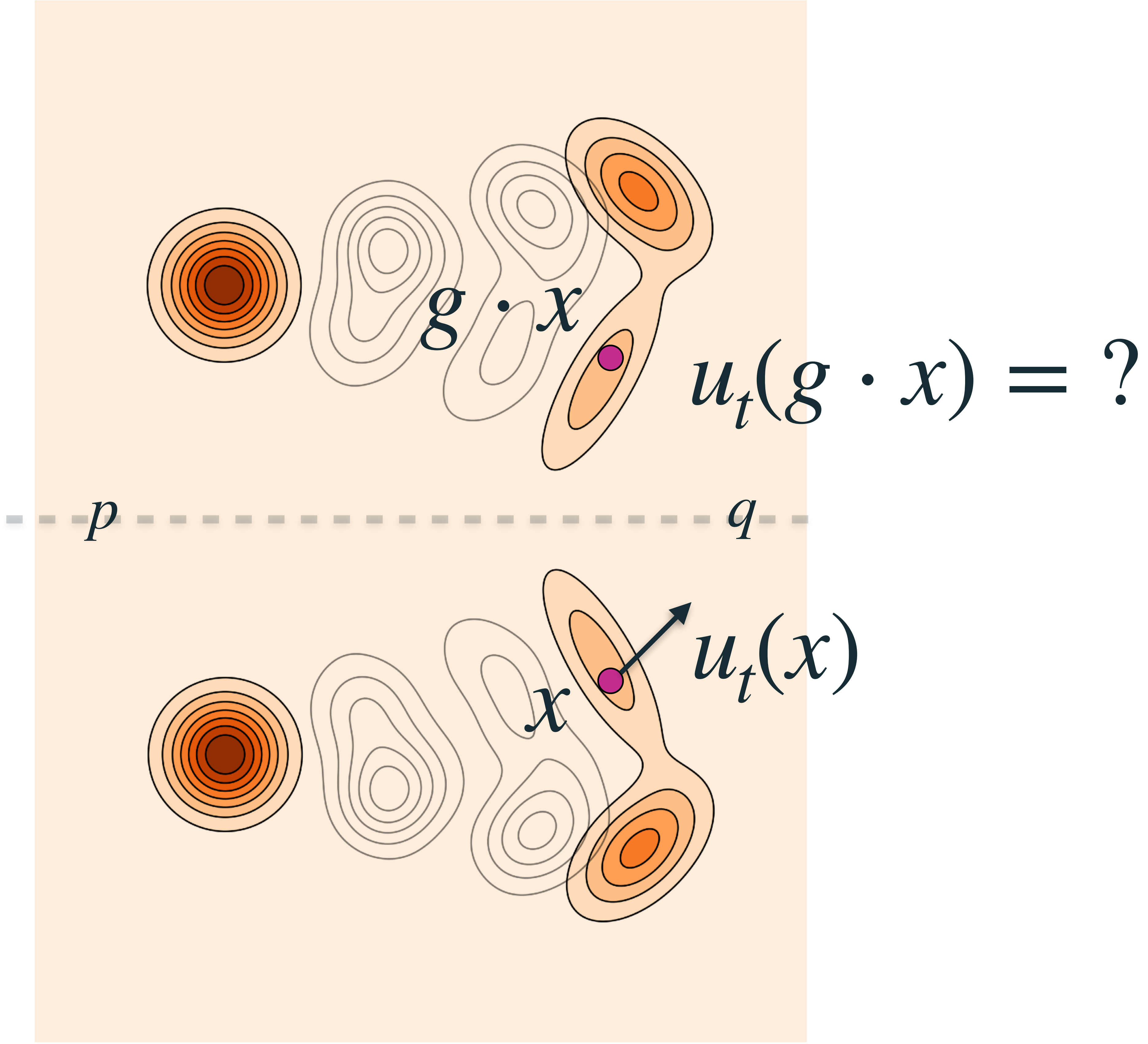
$$\psi_t(g \cdot x) = g \cdot \psi_t(x)$$

Solve ODE ↑ ↓ **Differentiate**

$$u_t(g \cdot x) = g \cdot u_t(x)$$

Equivariant Velocity

Example: Reflection



$$p_t(g \cdot x) = p_t(x)$$

Invariant probability path

"Equivariant Flows: Exact Likelihood Generative Learning for Symmetric Densities" Köhler et al. (2020)

Equivariant Flows

Equivariant Flow

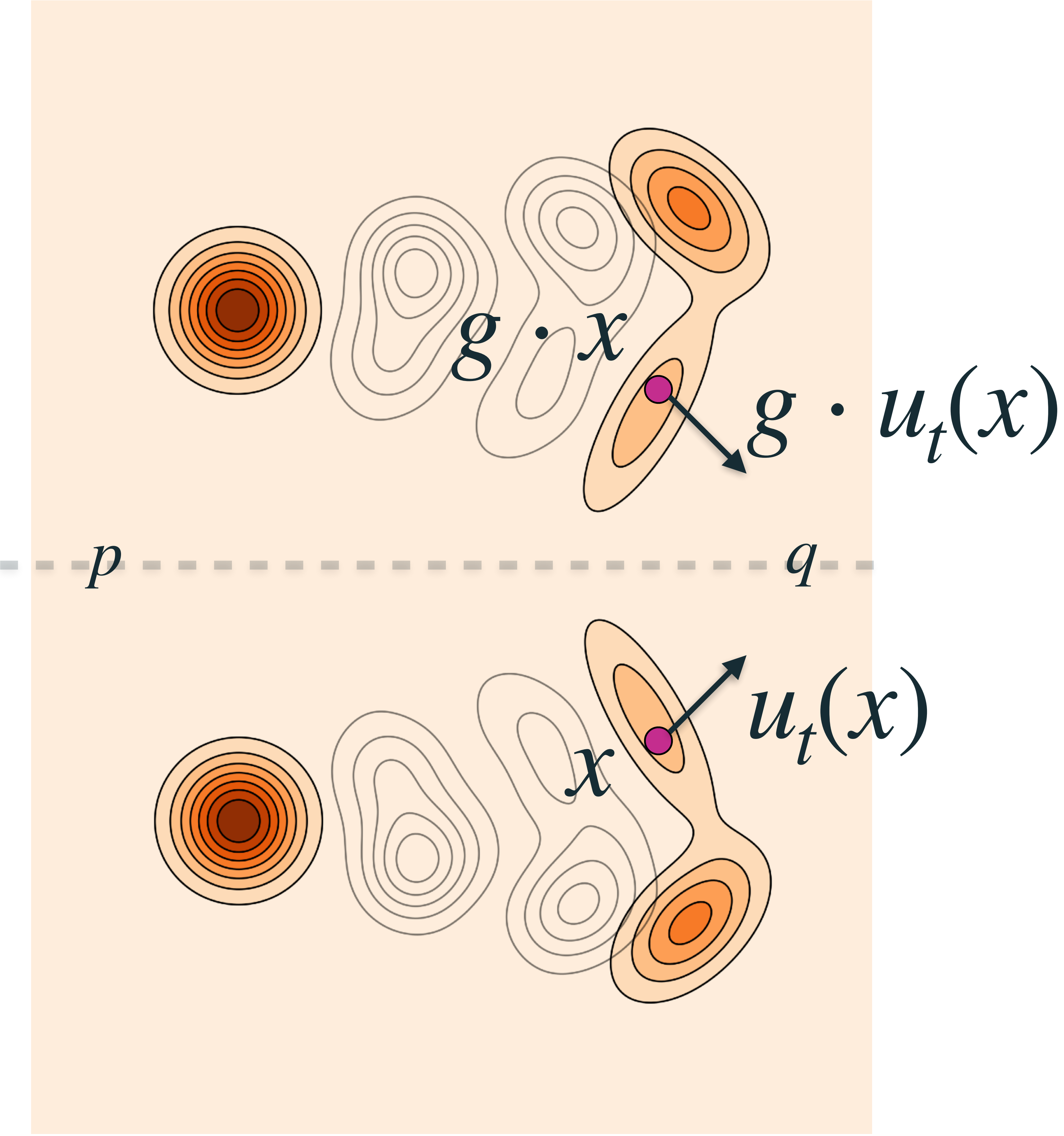
$$\psi_t(g \cdot x) = g \cdot \psi_t(x)$$

Solve ODE ↑ ↓ **Differentiate**

$$u_t(g \cdot x) = g \cdot u_t(x)$$

Equivariant Velocity

Example: Reflection



$$p_t(g \cdot x) = p_t(x)$$

Invariant probability path

"Equivariant Flows: Exact Likelihood Generative Learning for Symmetric Densities" Köhler et al. (2020)

Equivariant Flow Matching

Equivariant Velocity

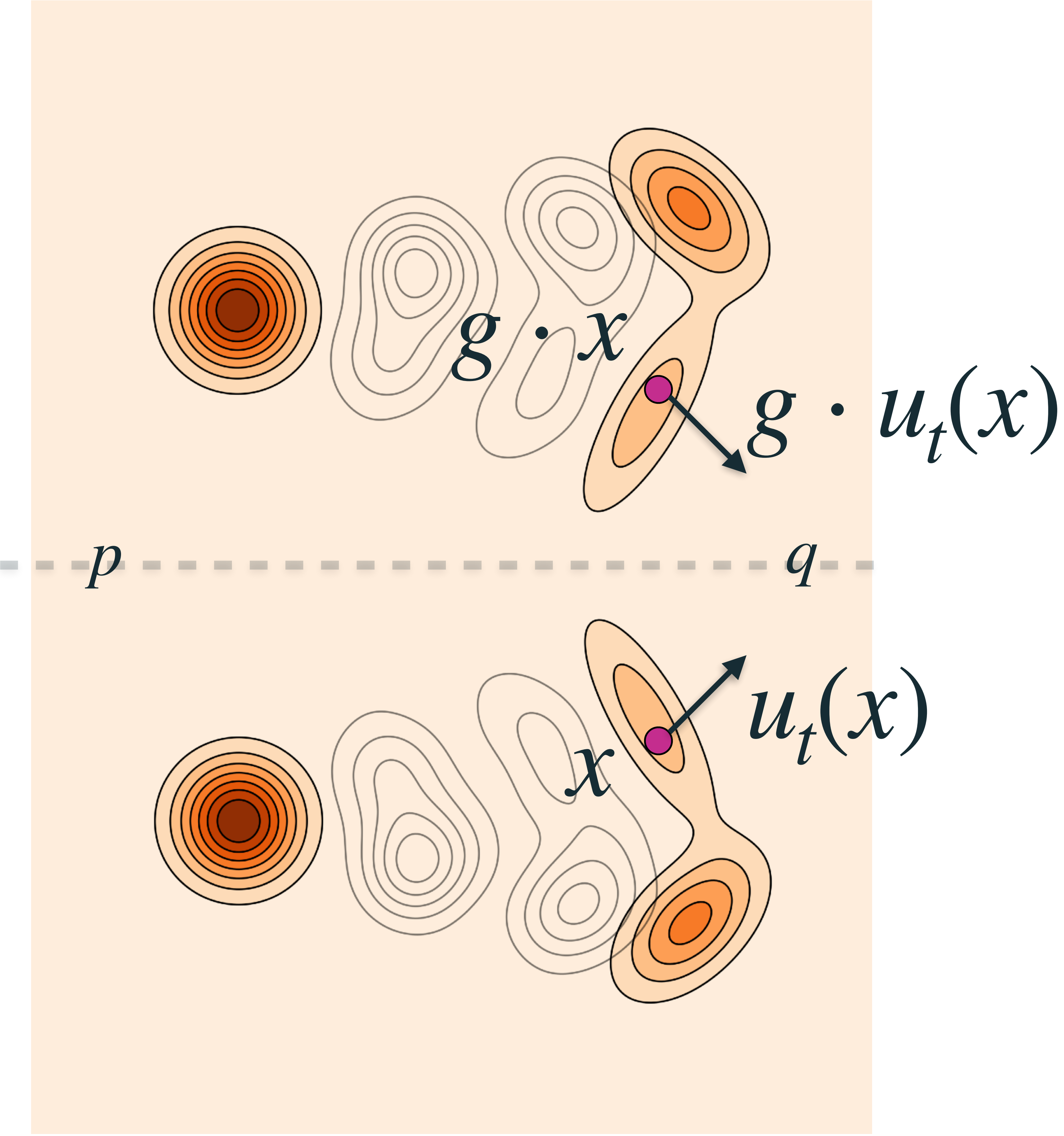
$$u_t^\theta(g \cdot x) = g \cdot u_t^\theta(x)$$

Train with CFM:

$$\mathscr{L}_{\mathrm{CFM}}(\theta) = \mathbb{E}_{t,X_1,X_t} \|u_t(X_t | X_1) - u_t^\theta(X_t)\|^2$$

$$(X_0, X_1) \sim \pi_{0,1} = p(X_0)q(X_1)$$

Example: Reflection



$$p_t(g \cdot x) = p_t(x)$$

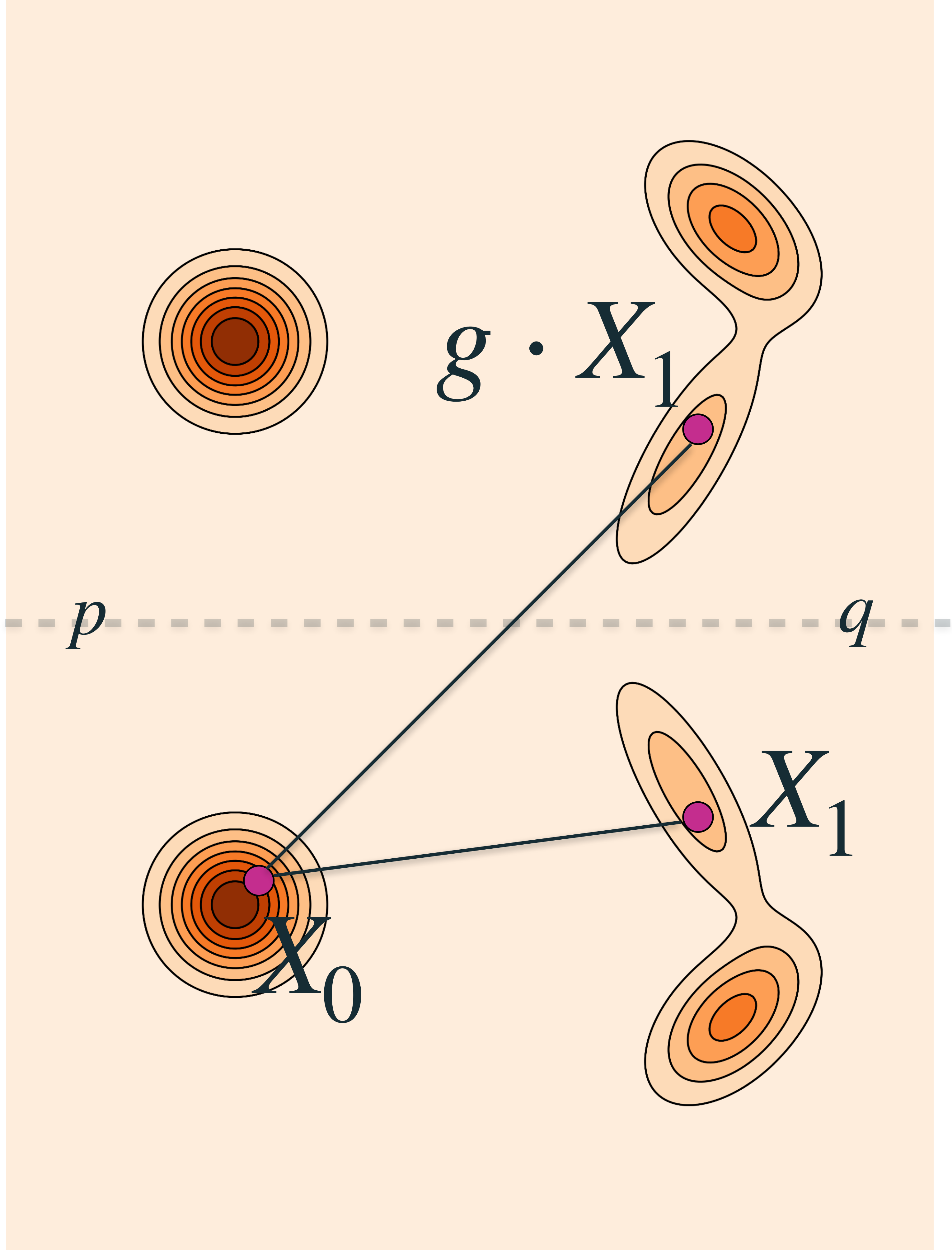
Invariant probability path

"Equivariant flow matching" Klein et al. (2023)

"Equivariant Flow Matching with Hybrid Probability Transport" Song et al. (2023)

Equivariant Flow Matching

Example: Reflection



Equivariant Velocity

$$u_t^\theta(g \cdot x) = g \cdot u_t^\theta(x)$$

Train with CFM:

$$\mathscr{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t,X_1,X_t} \|u_t(X_t | X_1) - u_t^\theta(X_t)\|^2$$

$$(X_0, X_1) \sim \pi_{0,1} = p(X_0)q(X_1)$$

"Equivariant flow matching" Klein et al. (2023)

"Equivariant Flow Matching with Hybrid Probability Transport" Song et al. (2023)

Equivariant Flow Matching

Equivariant Velocity $\quad u_t^\theta(g \cdot x) = g \cdot u_t^\theta(x)$

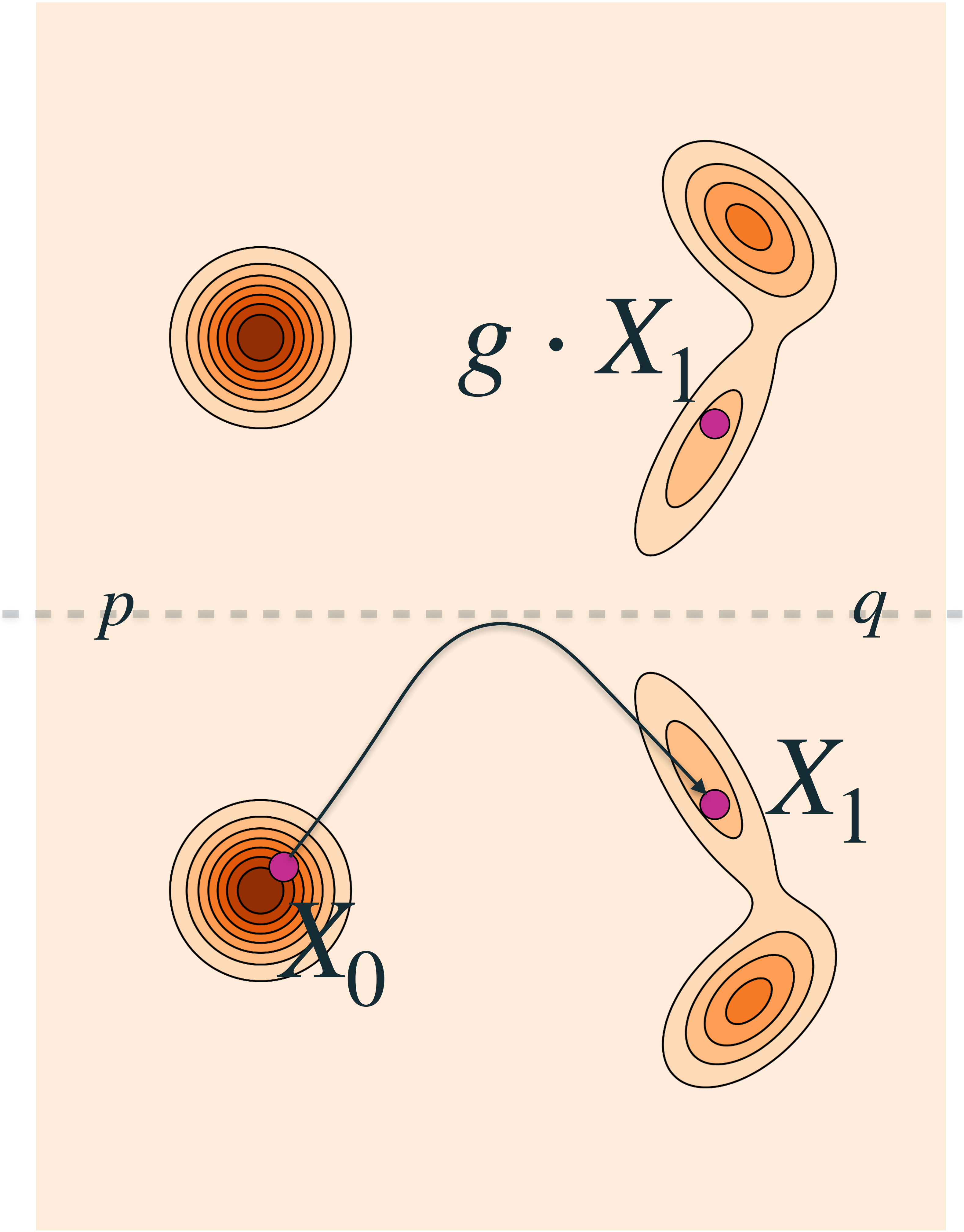
Train with CFM:

$$\mathscr{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t,X_1,X_t} \|u_t(X_t | X_1) - u_t^\theta(X_t)\|^2$$

$$(X_0, X_1) \sim \pi_{0,1} = p(X_0)q(X_1)$$

Coupling disregards symmetry → Curved trajectories

Example: Reflection



Alignment Couplings

Equivariant Velocity

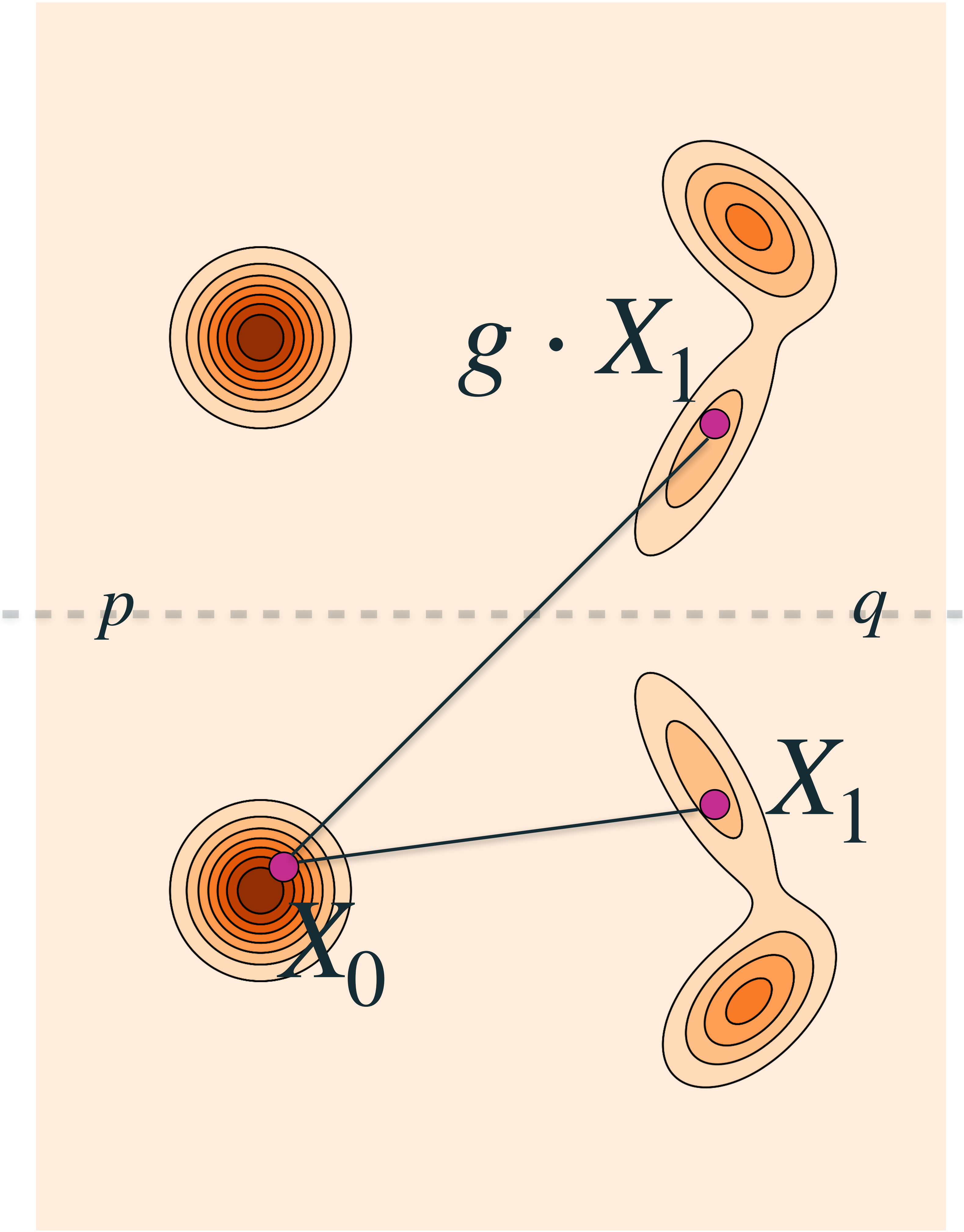
$$u_t^\theta(g \cdot x) = g \cdot u_t^\theta(x)$$

Train with CFM:

$$\mathscr{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t,X_1,X_t} \| u_t(X_t | X_1) - u_t^\theta(X_t) \|^2$$

$$(X_0, X_1) \sim \pi_{0,1}^{\text{Align}}$$

Example: Reflection

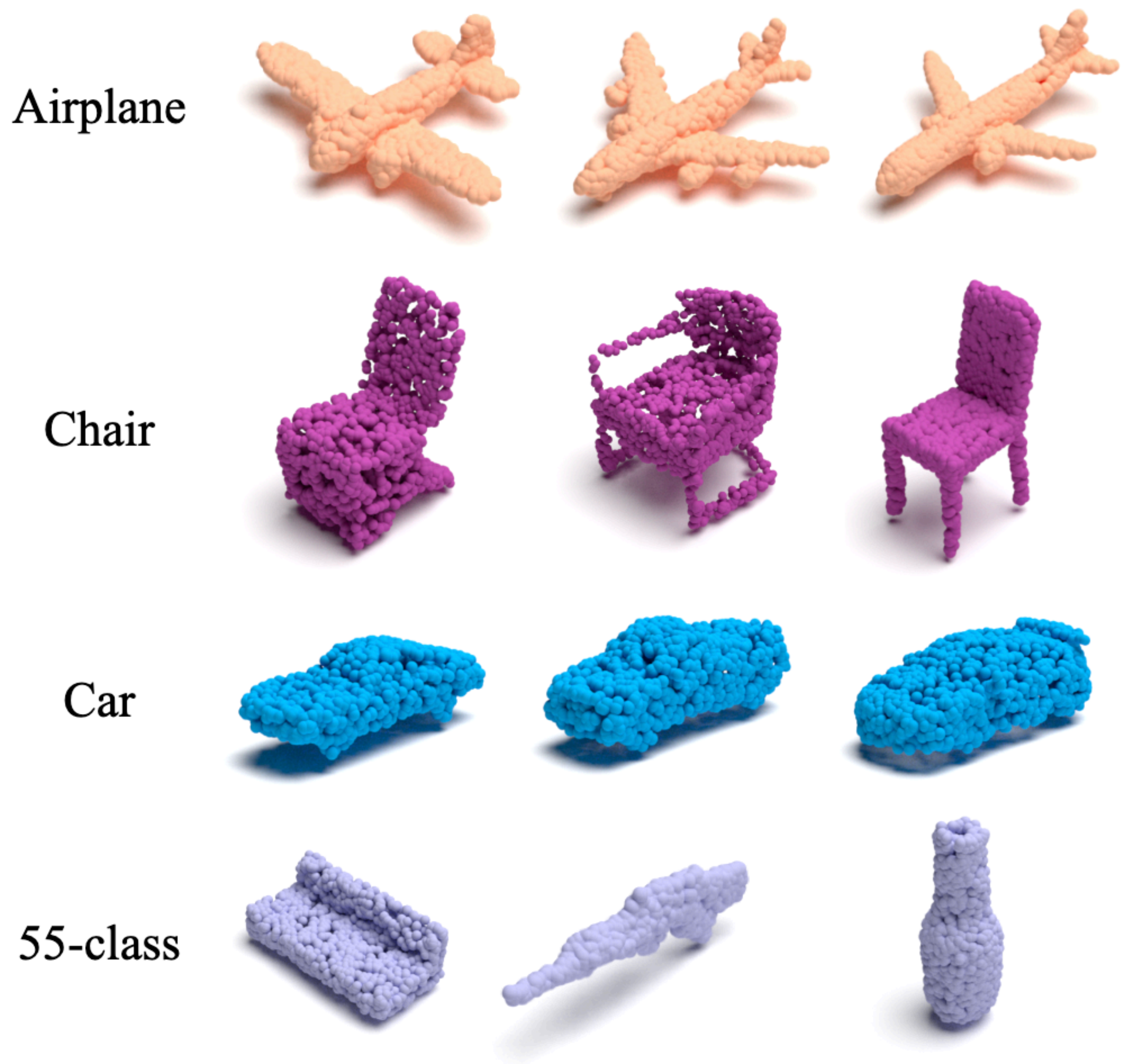


"Equivariant flow matching" Klein et al. (2023)

"Equivariant Flow Matching with Hybrid Probability Transport" Song et al. (2023)

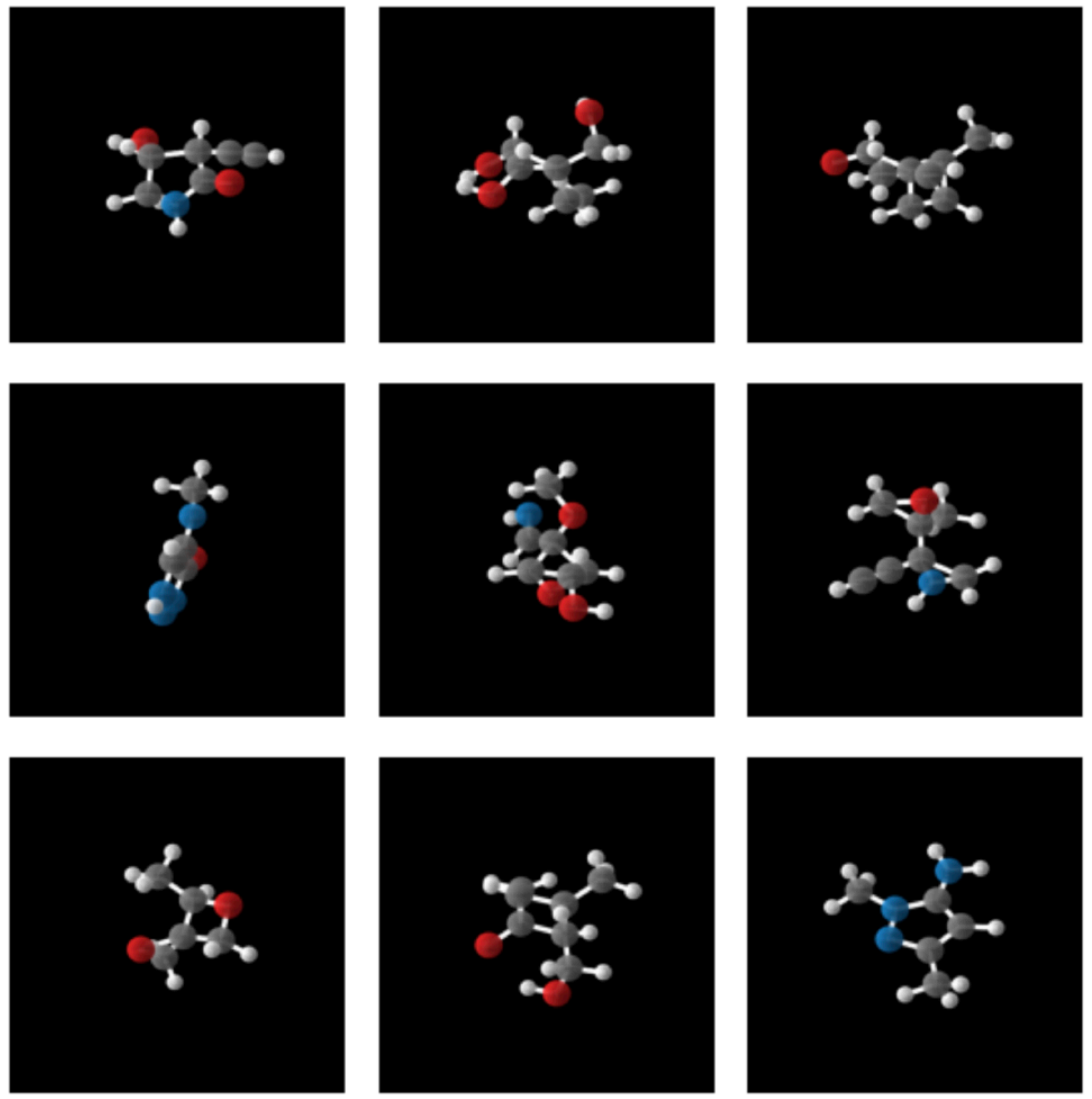
Equivariant Flow Matching

"Fast Point Cloud Generation with Straight Flows" Wu et al. (2022)



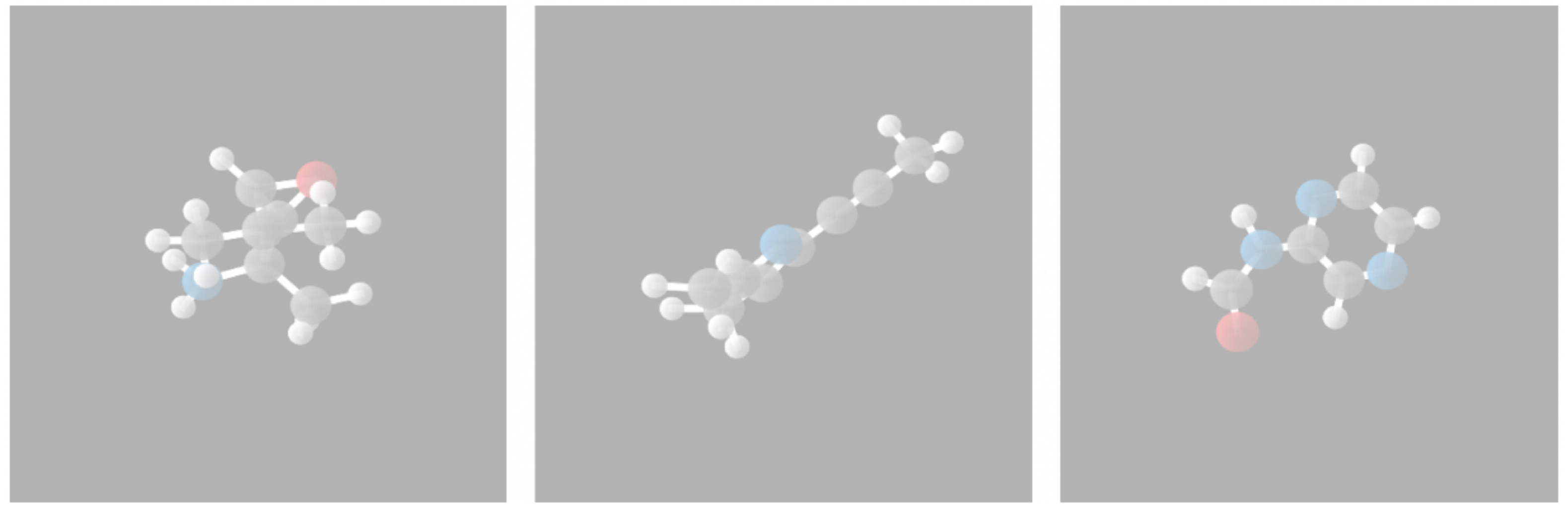
"Equivariant Flow Matching with Hybrid Probability Transport" Song et al. (2023)

"Equivariant flow matching" Klein et al. (2023)



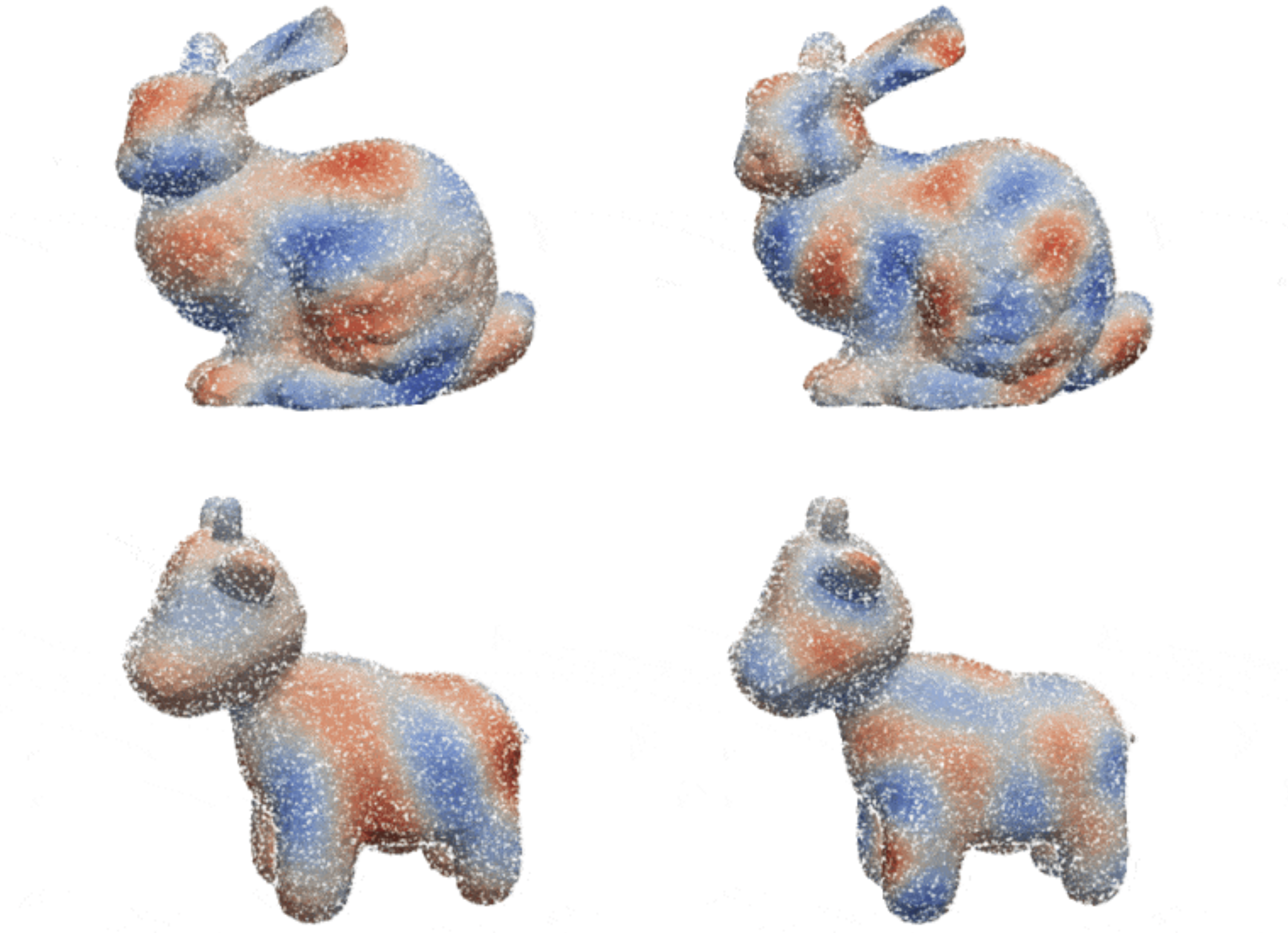
Geometric Flow Matching

Data with Symmetries



- Equivariant flows → invariant densities
- Alignment couplings

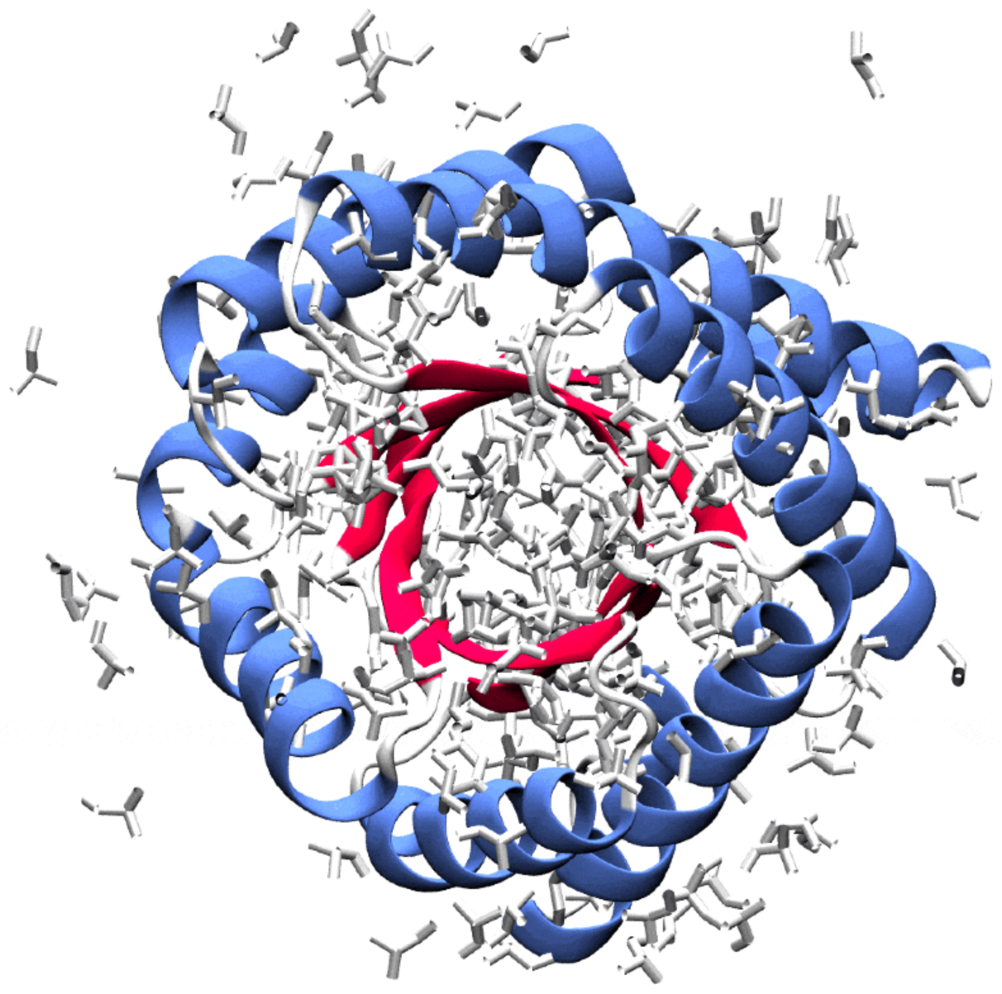
Riemannian Manifolds



- Simulation free on simple manifolds
- General geometries

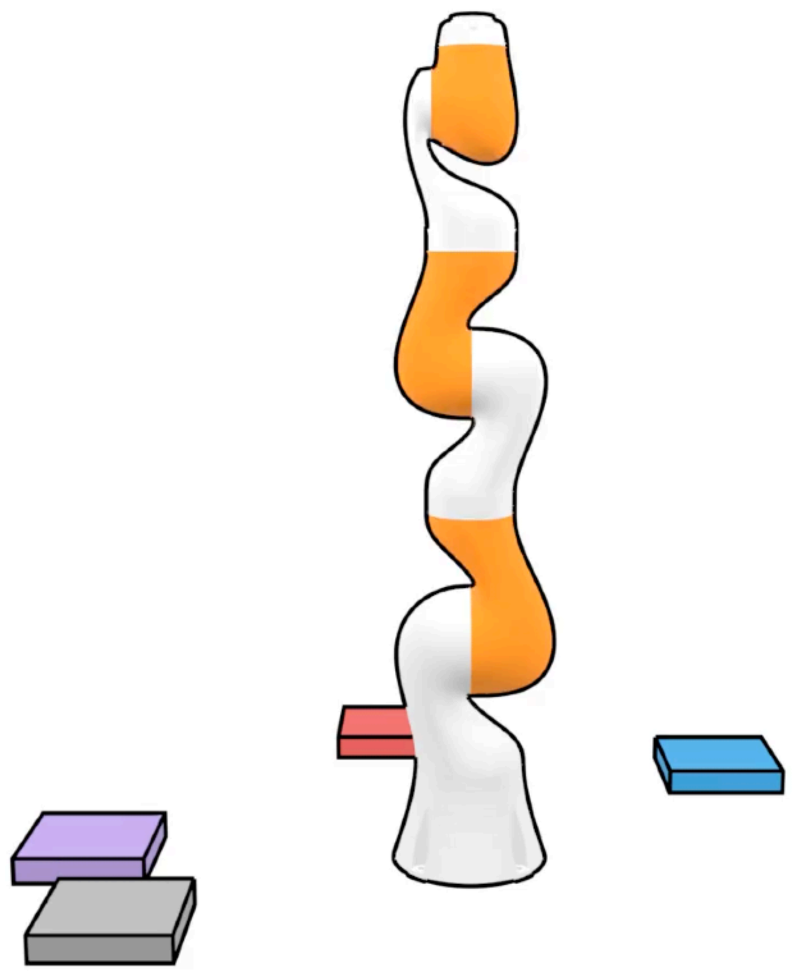
Generative Modeling on Manifolds

Scientific Data



SE(3) invariant
Protein structure generation

Robotics



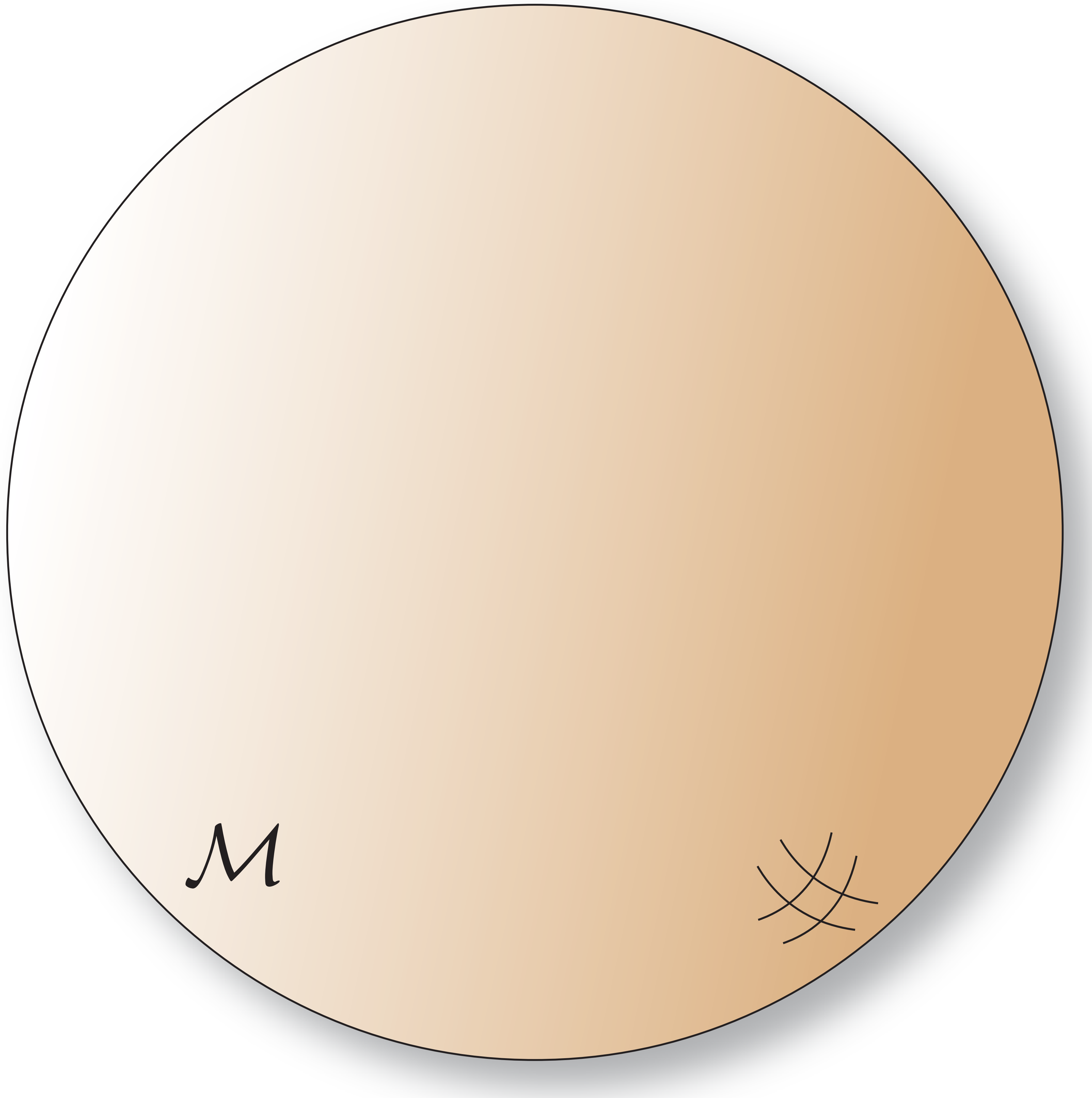
SO(2) invariant
Block stacking

Climate Modeling

Spherical Geometry $\mathbb{S}^2$

"Sequence-Augmented SE(3)-Flow Matching For Conditional Protein Backbone Generation" Huguet et al. (2024)
"Planning with Diffusion for Flexible Behavior Synthesis" Janner et al. (2022)

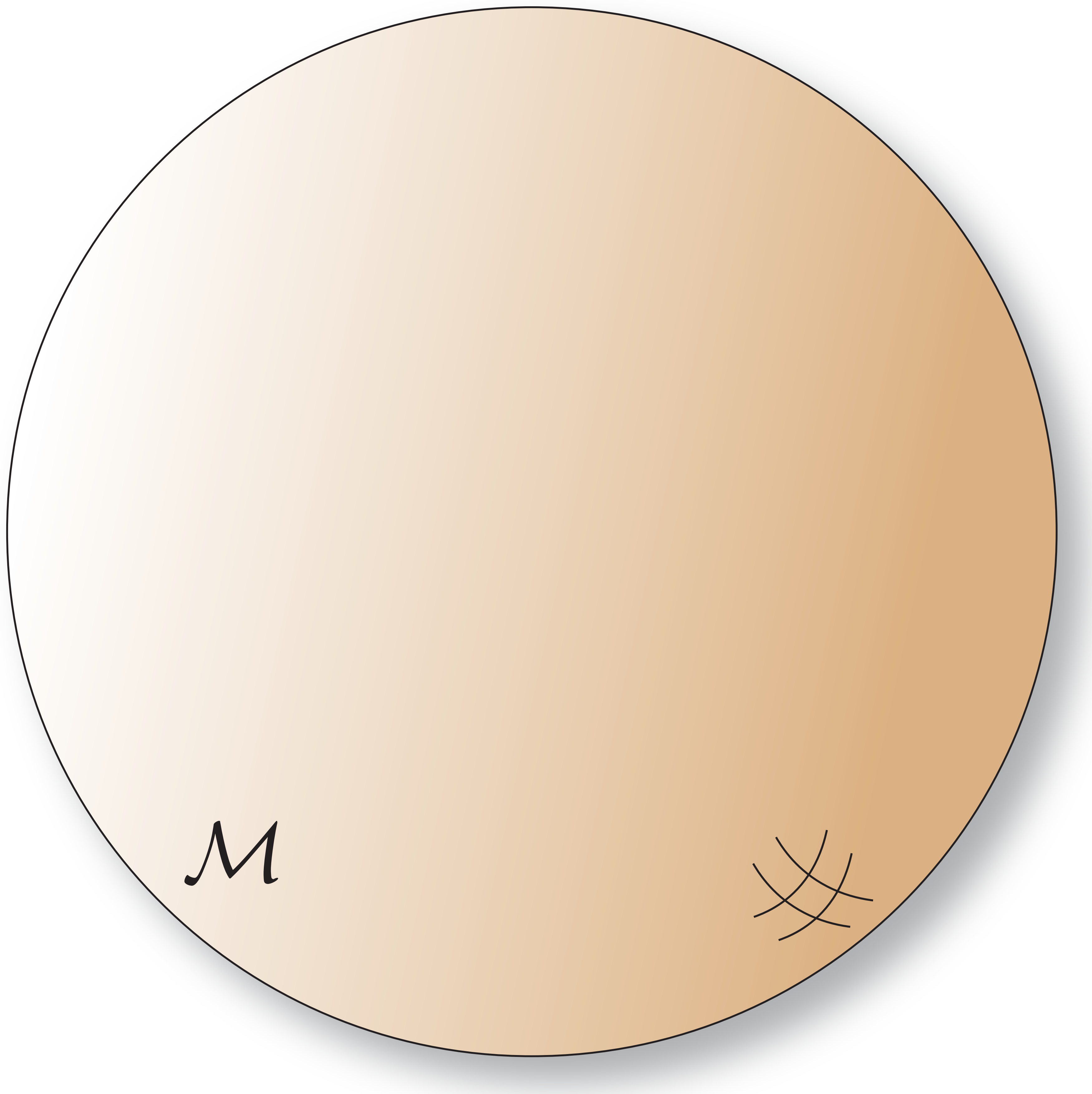
Need to re-define the geometric structures we have in Euclidean space.



Riemannian Manifolds

Smooth

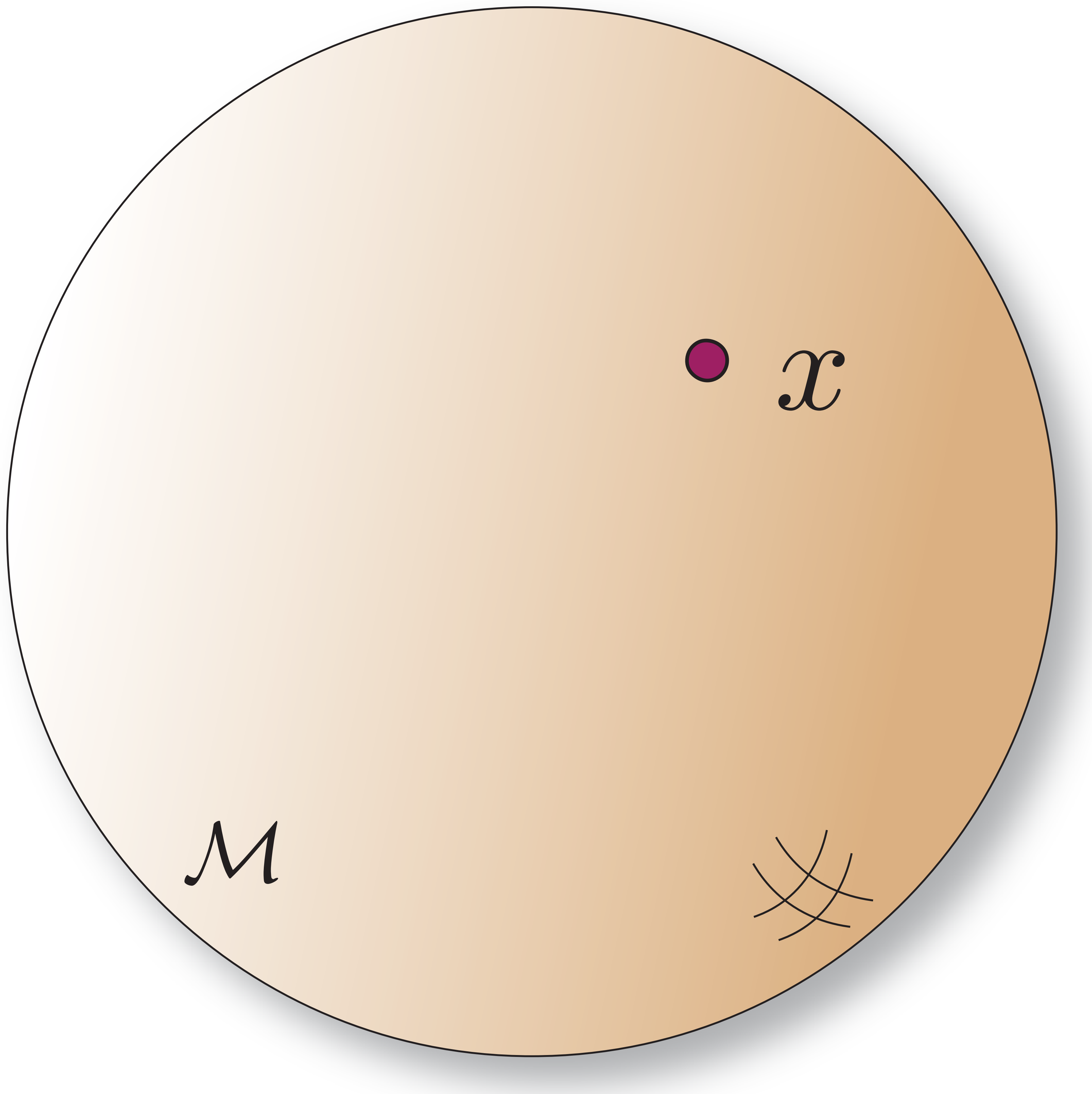
Global differential structure



Riemannian Manifolds

Smooth

Global differential structure



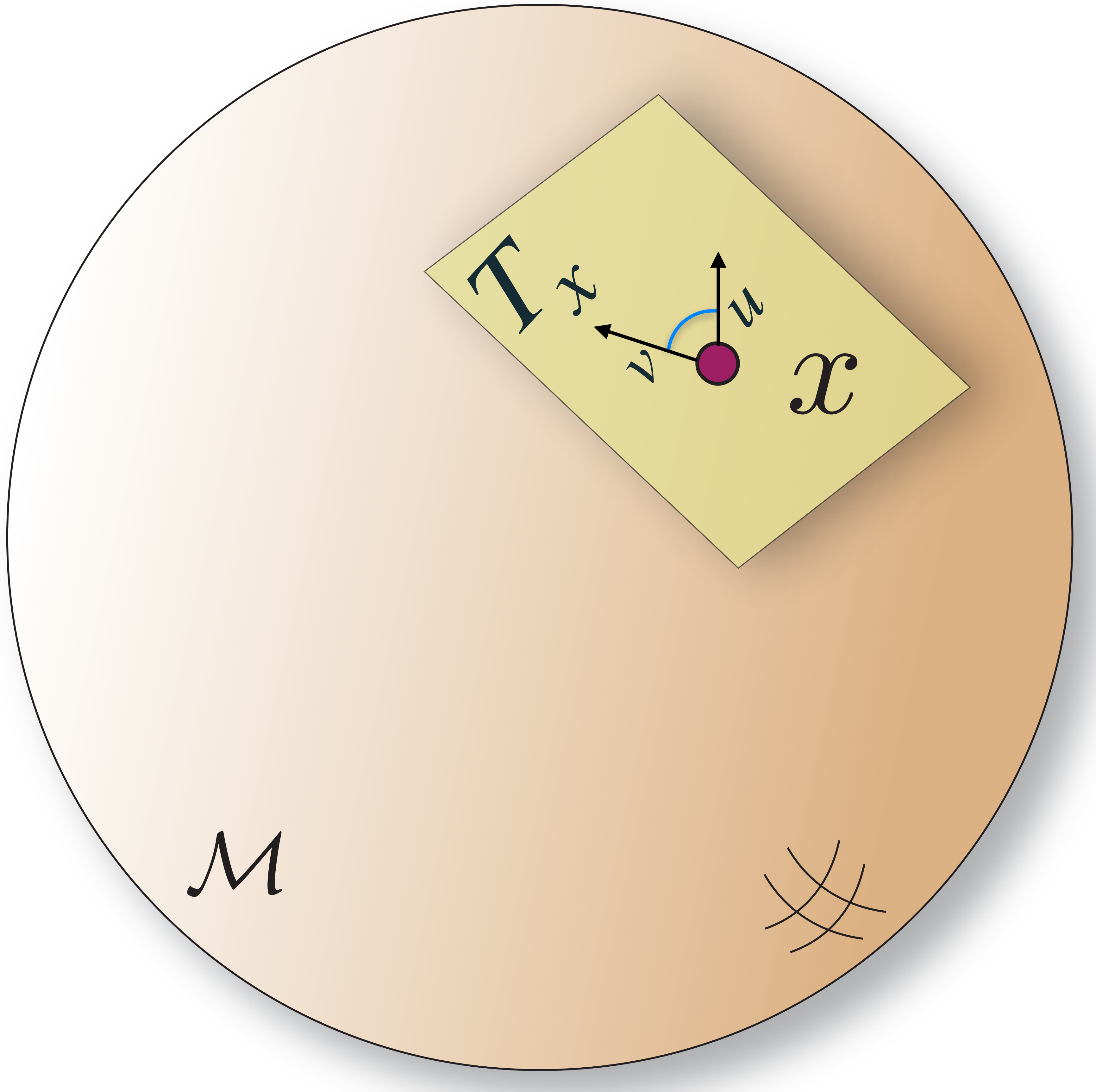
Riemannian Manifolds

Smooth | Global differential structure

+

Riemannian Metric | $\langle u, v \rangle_g$ $u, v \in T_x$

Riemannian Manifold



Flows on Manifolds

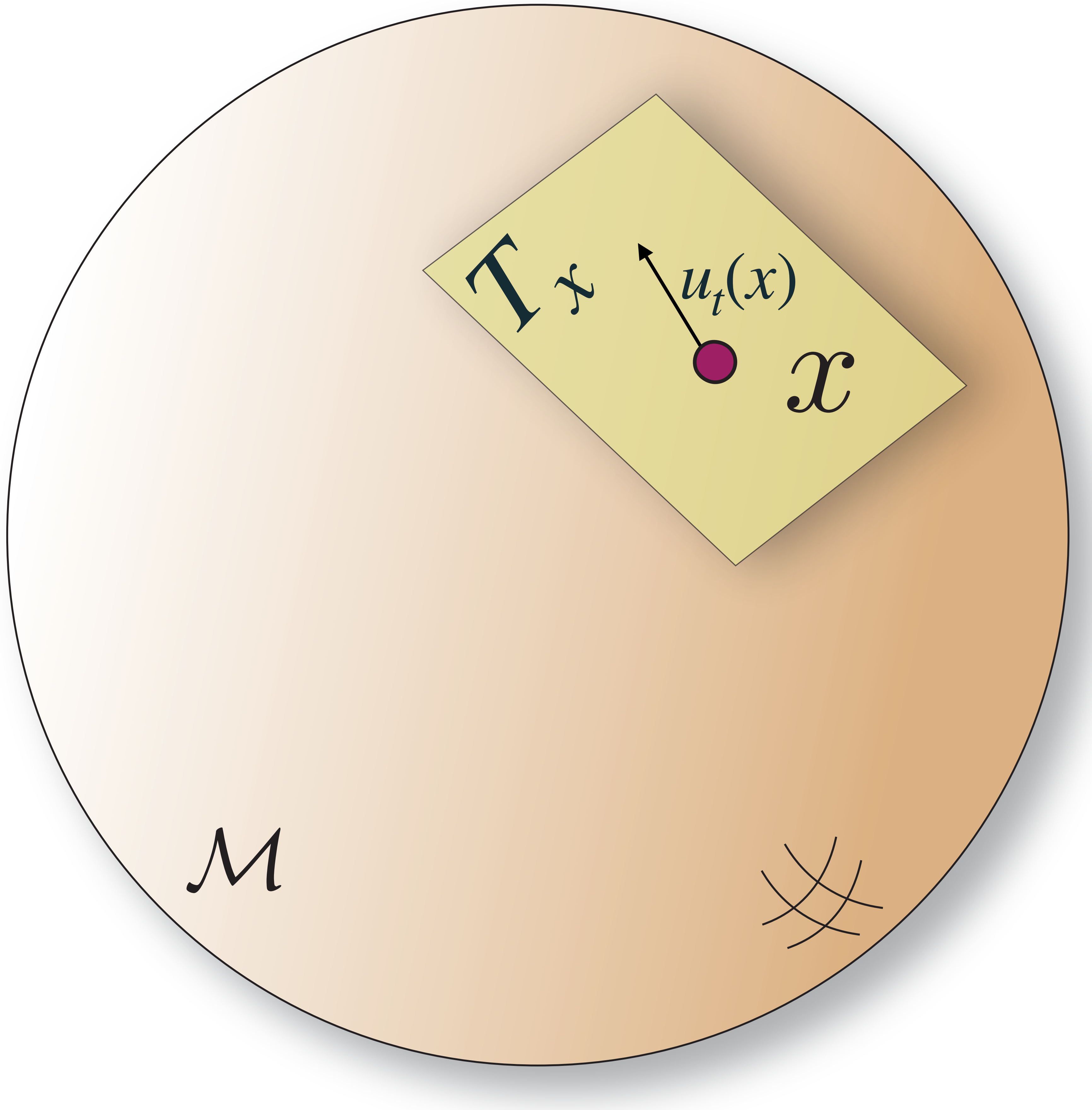
Flow

$$\psi_t(x)$$

Solve ODE ↑ ↓ **Differentiate**

$$u_t(x) \in T_x$$

Velocity



Flows on Manifolds

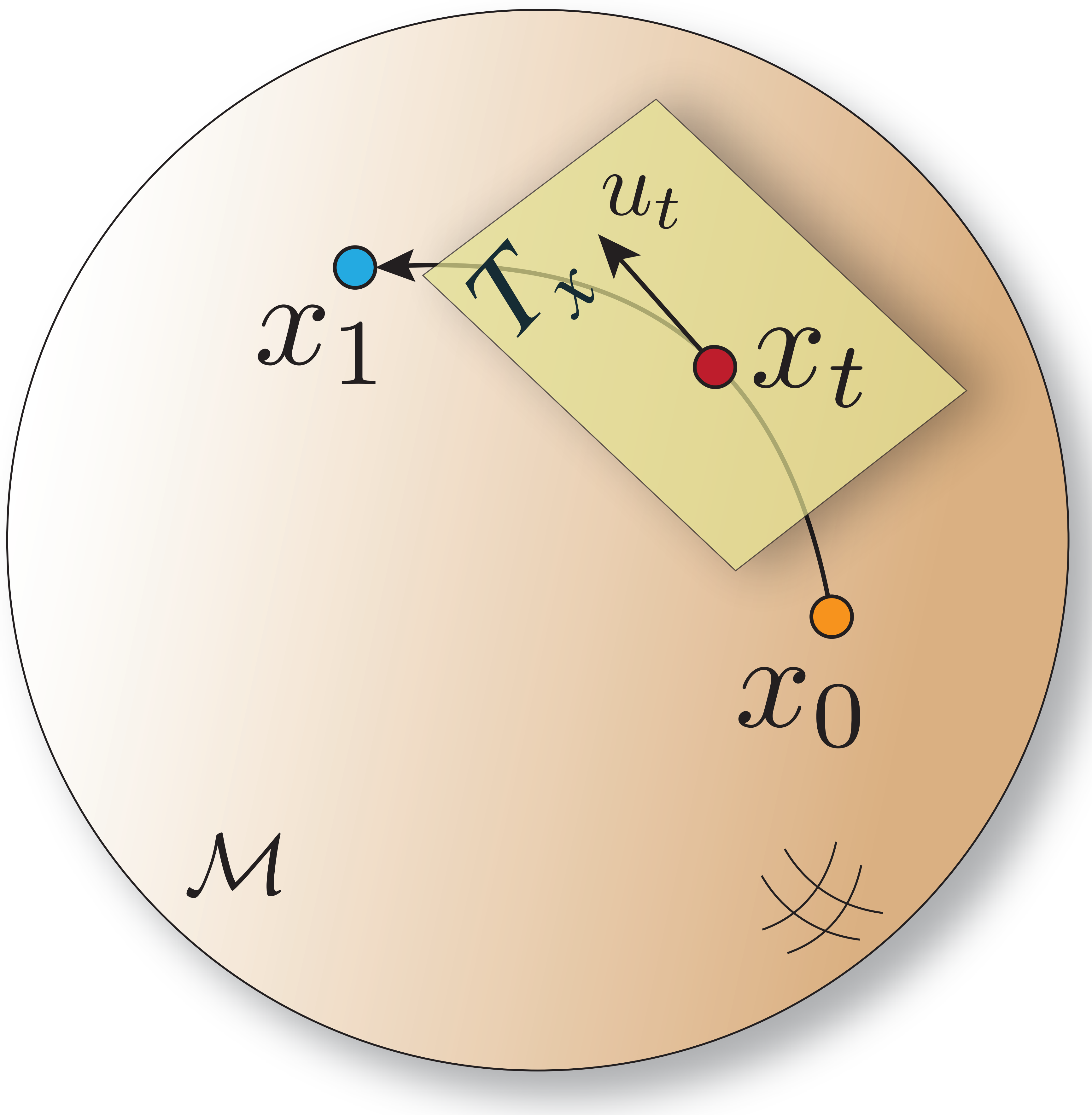
Flow

$$\psi_t(x)$$

Solve ODE ↑ ↓ **Differentiate**

$$u_t(x) \in T_x$$

Velocity



Riemannian Flow Matching

- **Riemannian Flow Matching loss:**

$$\mathscr{L}_{\mathrm{RFM}}(\theta) = \mathbb{E}_{t,X_t} \left\| u_t^\theta(X_t) - u_t(X_t) \right\|_g^2$$

Riemanninan

Metric

"Flow Matching on General Geometries" Chen & Lipman (2023)

Riemannian Flow Matching

- **Riemannian Flow Matching loss:**

$$\mathcal{L}_{\mathrm{RFM}}(\theta) = \mathbb{E}_{t,X_t} \left\| u_t^\theta(X_t) - u_t(X_t) \right\|_g^2$$

- **Riemannian Conditional Flow Matching loss:**

$$\mathcal{L}_{\mathrm{RCFM}}(\theta) = \mathbb{E}_{t,X_1,X_t} \left\| u_t^\theta(X_t) - \boxed{u_t(X_t \mid X_1)} \right\|_g^2$$

Theorem: Losses are equivalent,

$$\nabla_\theta \mathcal{L}_{\mathrm{RFM}}(\theta) = \nabla_\theta \mathcal{L}_{\mathrm{RCFM}}(\theta)$$

"Flow Matching on General Geometries" Chen & Lipman (2023)

Conditional Flows - Simple Geometries

Straight lines → Geodesics

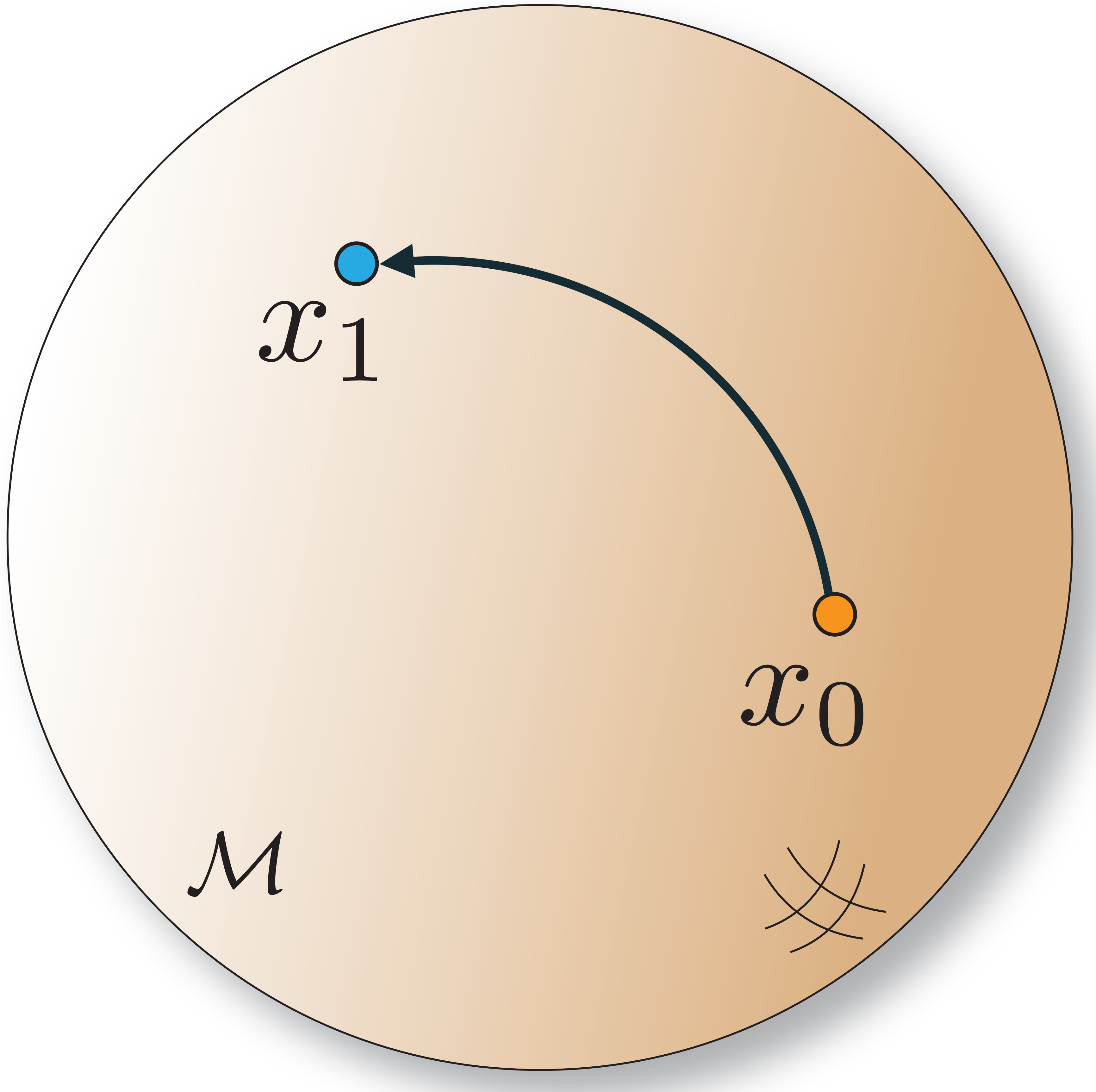
For simple manifolds (e.g. Euclidean, sphere, torus, hyperbolic)**:**

$$\psi_t(x_0 \,|\, x_1) = \exp_{x_0}(\kappa(t)\log_{x_0}(x_1)), \quad t \in [0,1]$$

Closed-form geodesic

Scheduler $\kappa(t)$: $\kappa(0) = 0$, $\kappa(1) = 1$

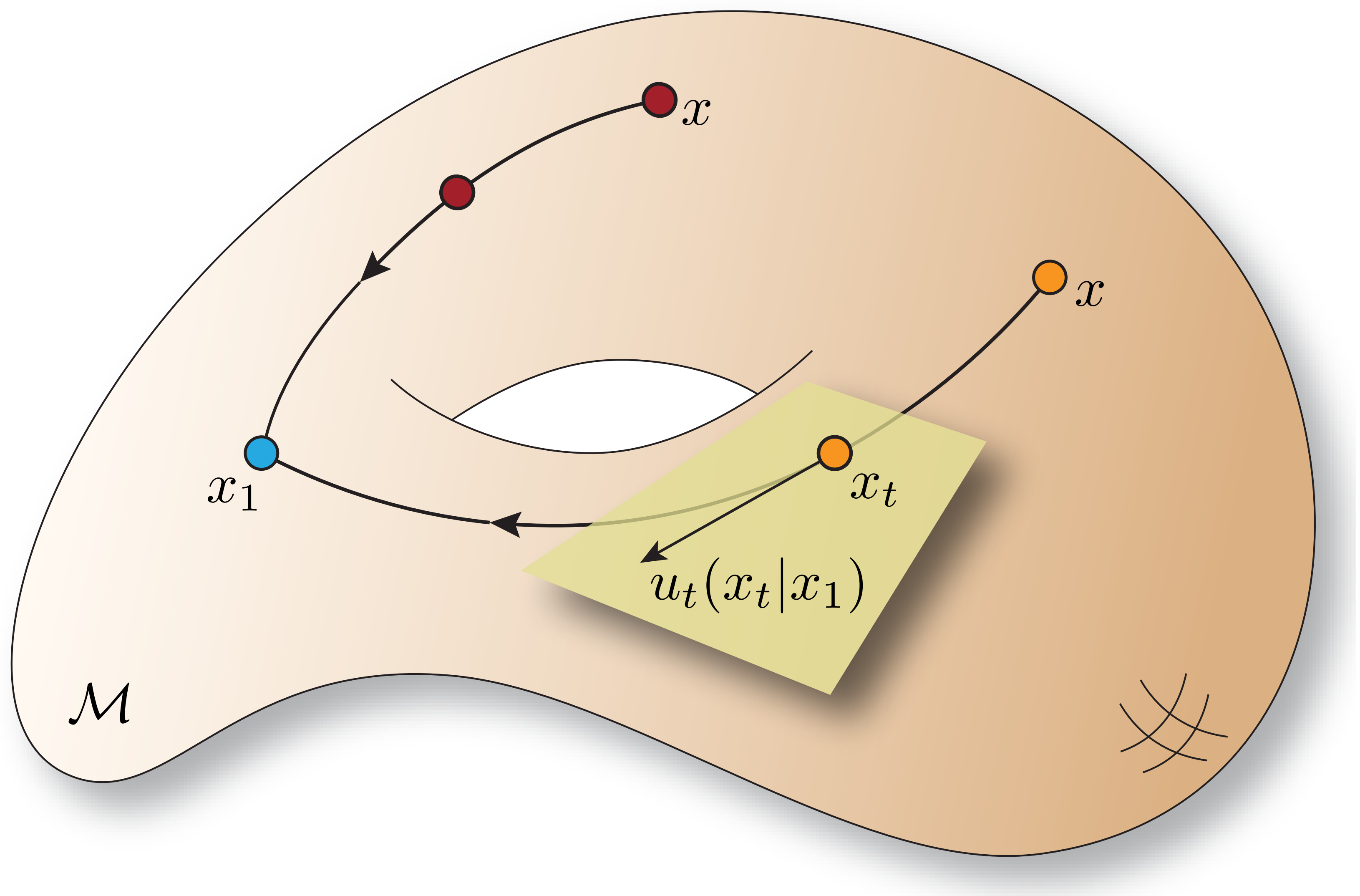
Simulation Free!



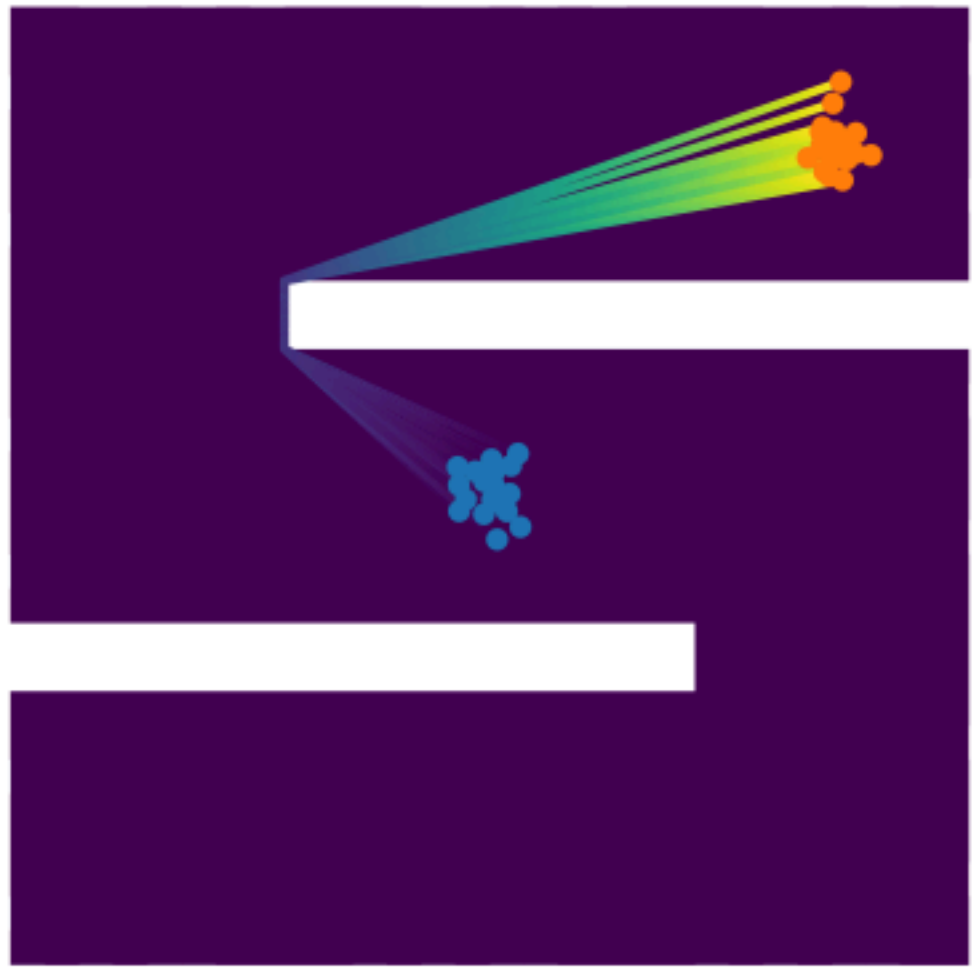
"Flow Matching on General Geometries" Chen & Lipman (2023)

Conditional Flows - General Geometries

Geodesics can be hard to compute



Concentrate probability at boundary

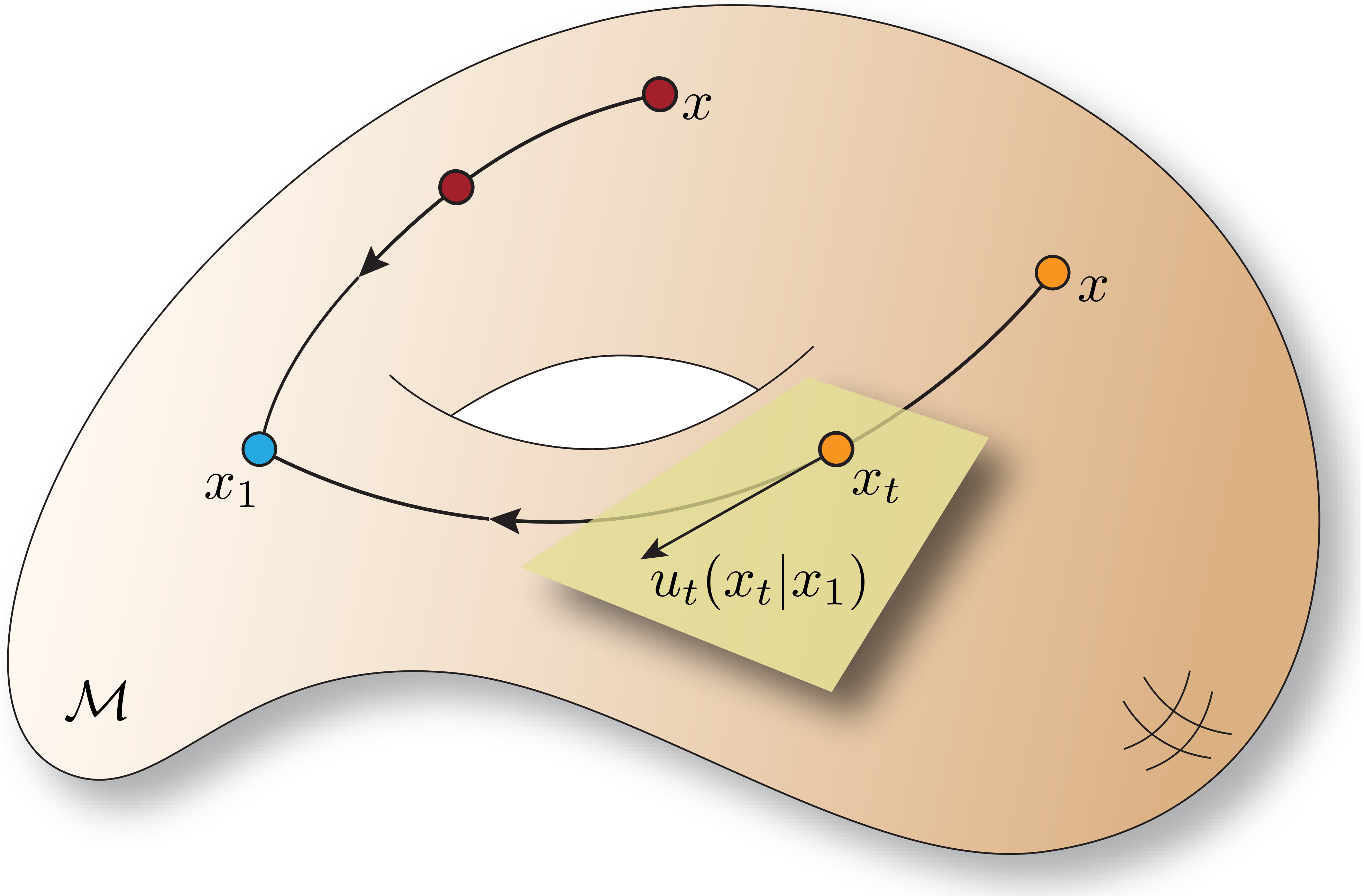


"Flow Matching on General Geometries" Chen & Lipman (2023)

Conditional Flows - General Geometries

Choose a premetric satisfying:

1. Non-negative: $\mathrm{d}(x, y) \geq 0$.
2. Positive: $\mathrm{d}(x, y) = 0$ iff $x = y$.
3. Non-degenerate: $\nabla \mathrm{d}(x, y) \neq 0$ iff $x \neq y$.



Build conditional flow satisfying:

$$\mathrm{d}(\psi_t(x_0 \,|\, x_1), x_1) = \bar{\kappa}(t)\mathrm{d}(x_0, x_1)$$

Scheduler $\bar{\kappa}(t) = 1 - \kappa(t)$

"Flow Matching on General Geometries" Chen & Lipman (2023)

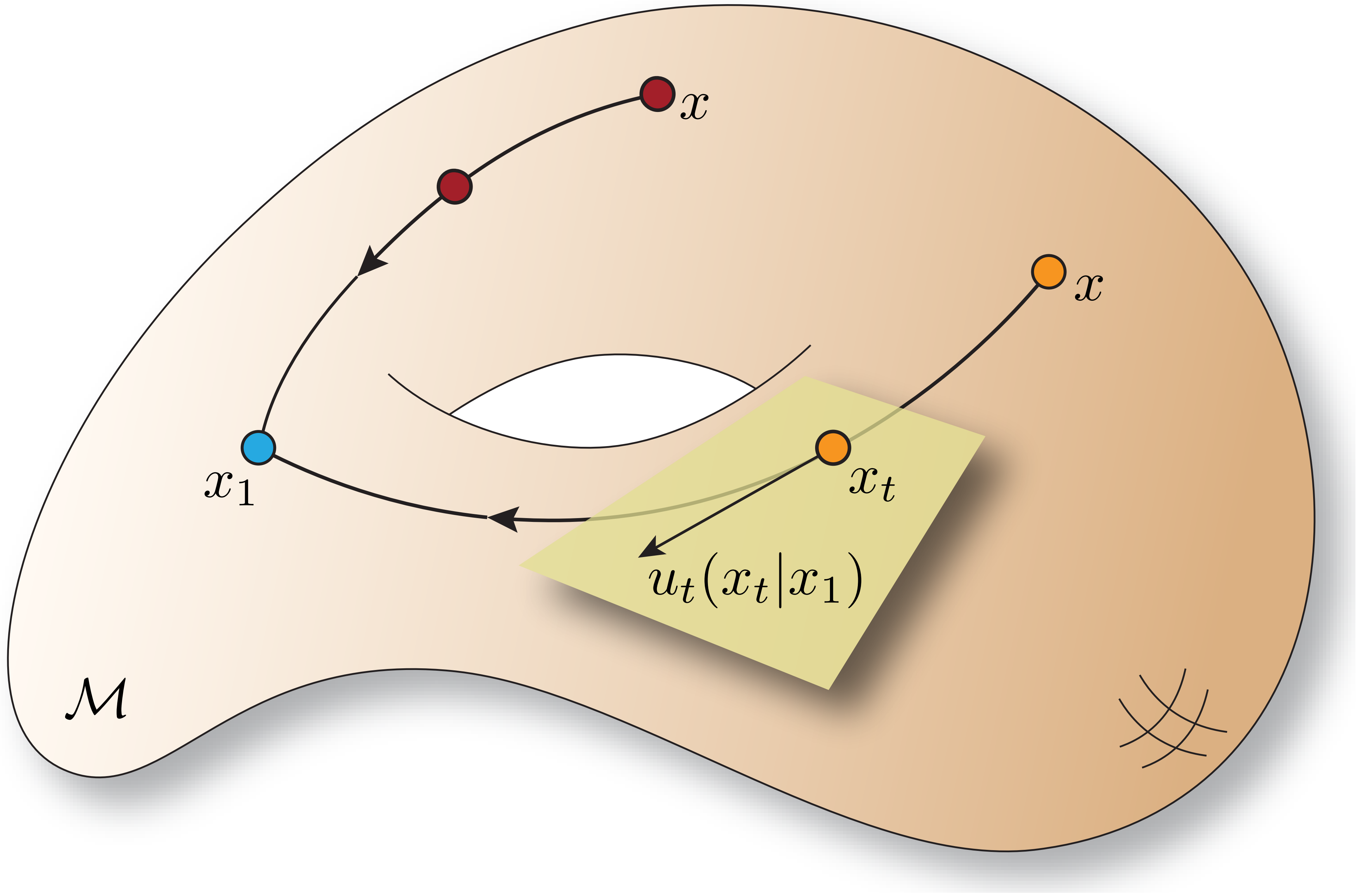
Conditional Flows - General Geometries

Build conditional flow satisfying:

$$\mathrm{d}(\psi_t(x_0 \mid x_1), x_1) = \bar{\kappa}(t)\mathrm{d}(x_0, x_1)$$

$$u_t(x \mid x_1) = \frac{d \log \bar{\kappa}(t)}{dt} \mathrm{d}(x, x_1) \frac{\nabla \mathrm{d}(x, x_1)}{\|\nabla \mathrm{d}(x, x_1)\|_g^2}$$

Requires simulation



"Flow Matching on General Geometries" Chen & Lipman (2023)

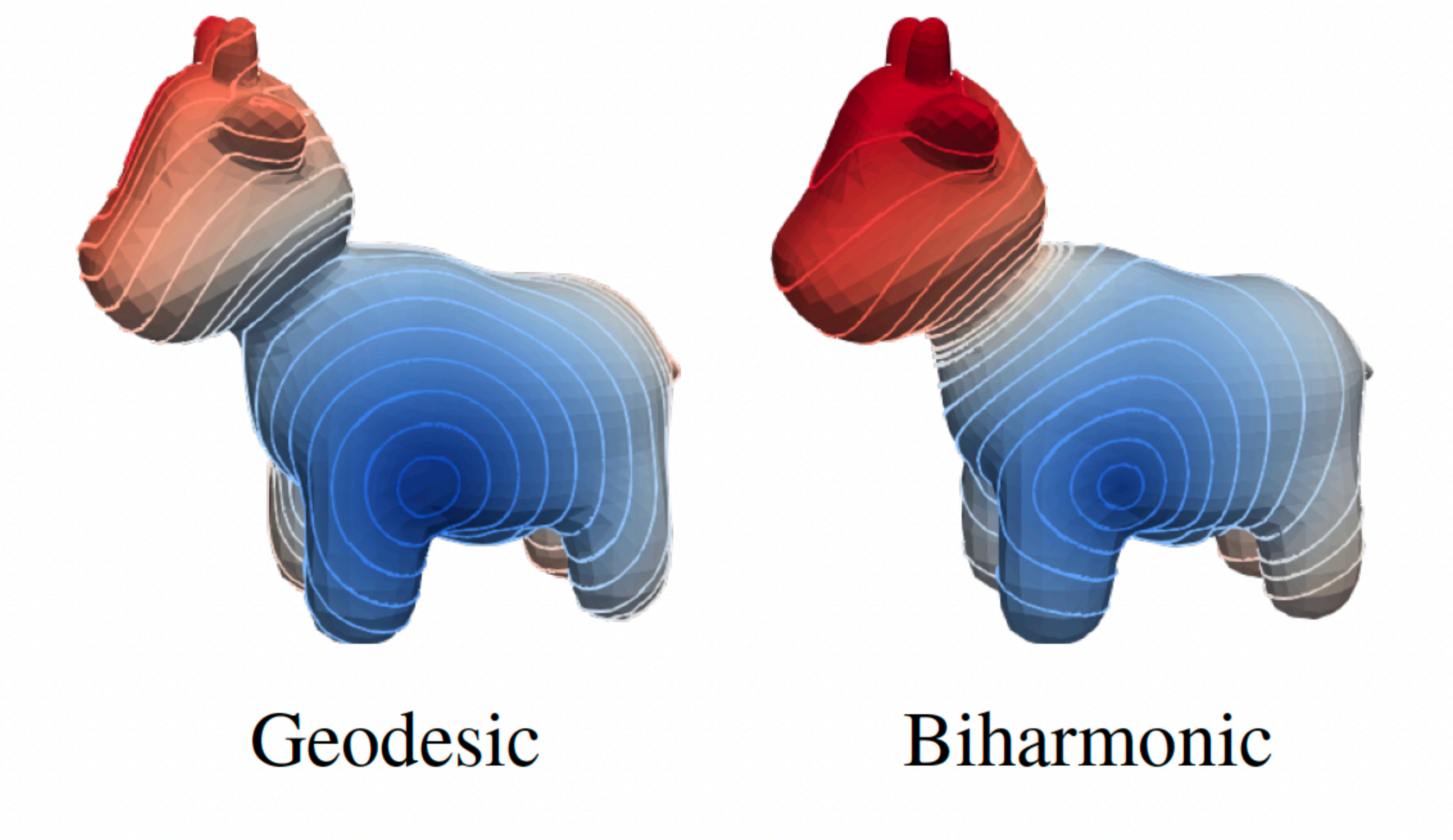
Conditional Flows - General Geometries

Build conditional flow satisfying:

$$\mathrm{d}(\psi_t(x_0 \mid x_1), x_1) = \bar{\kappa}(t)\mathrm{d}(x_0, x_1)$$

$$u_t(x \mid x_1) = \frac{d \log \bar{\kappa}(t)}{dt} \mathrm{d}(x, x_1) \frac{\nabla \mathrm{d}(x, x_1)}{\|\nabla \mathrm{d}(x, x_1)\|_g^2}$$

Requires simulation



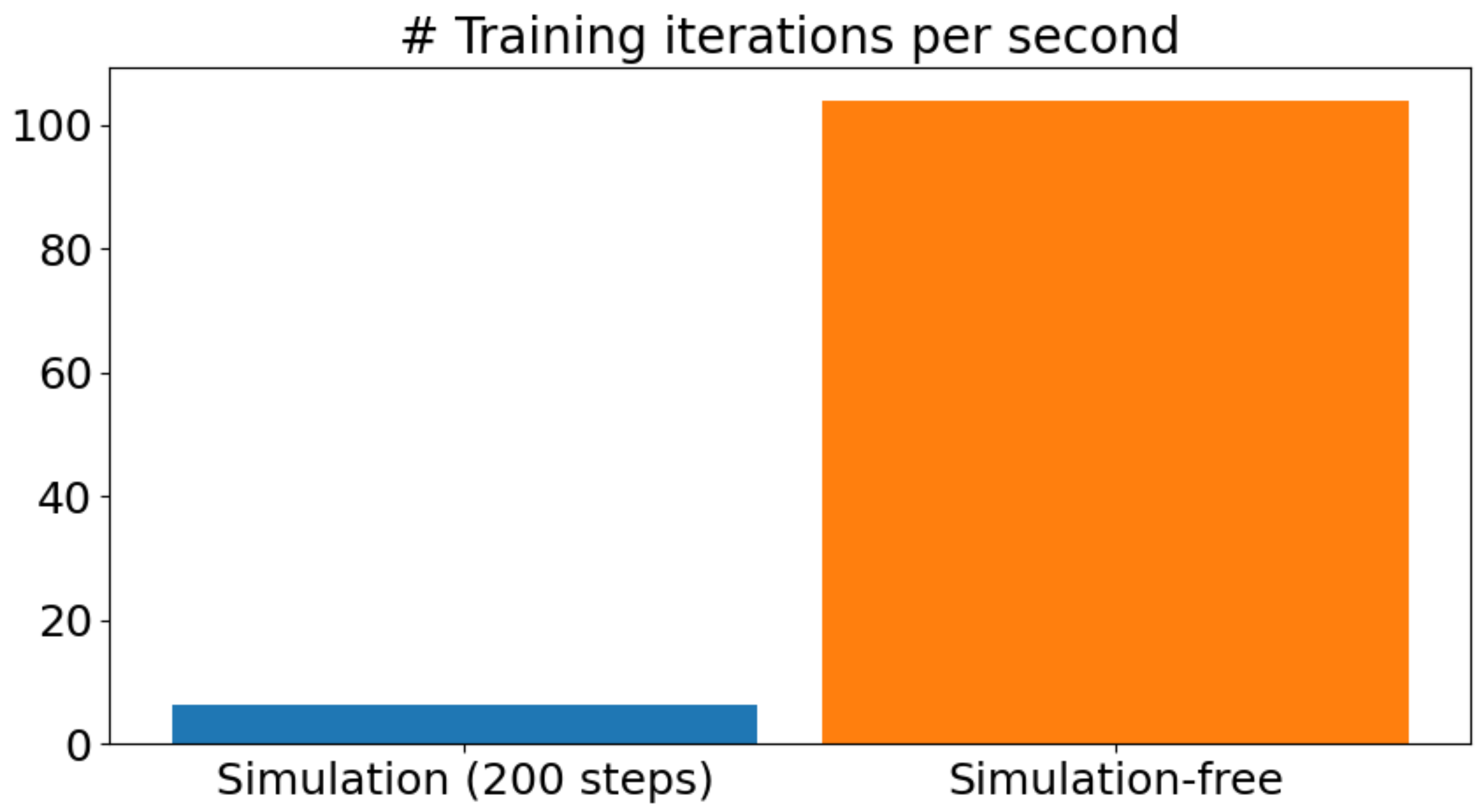
"Flow Matching on General Geometries" Chen & Lipman (2023)

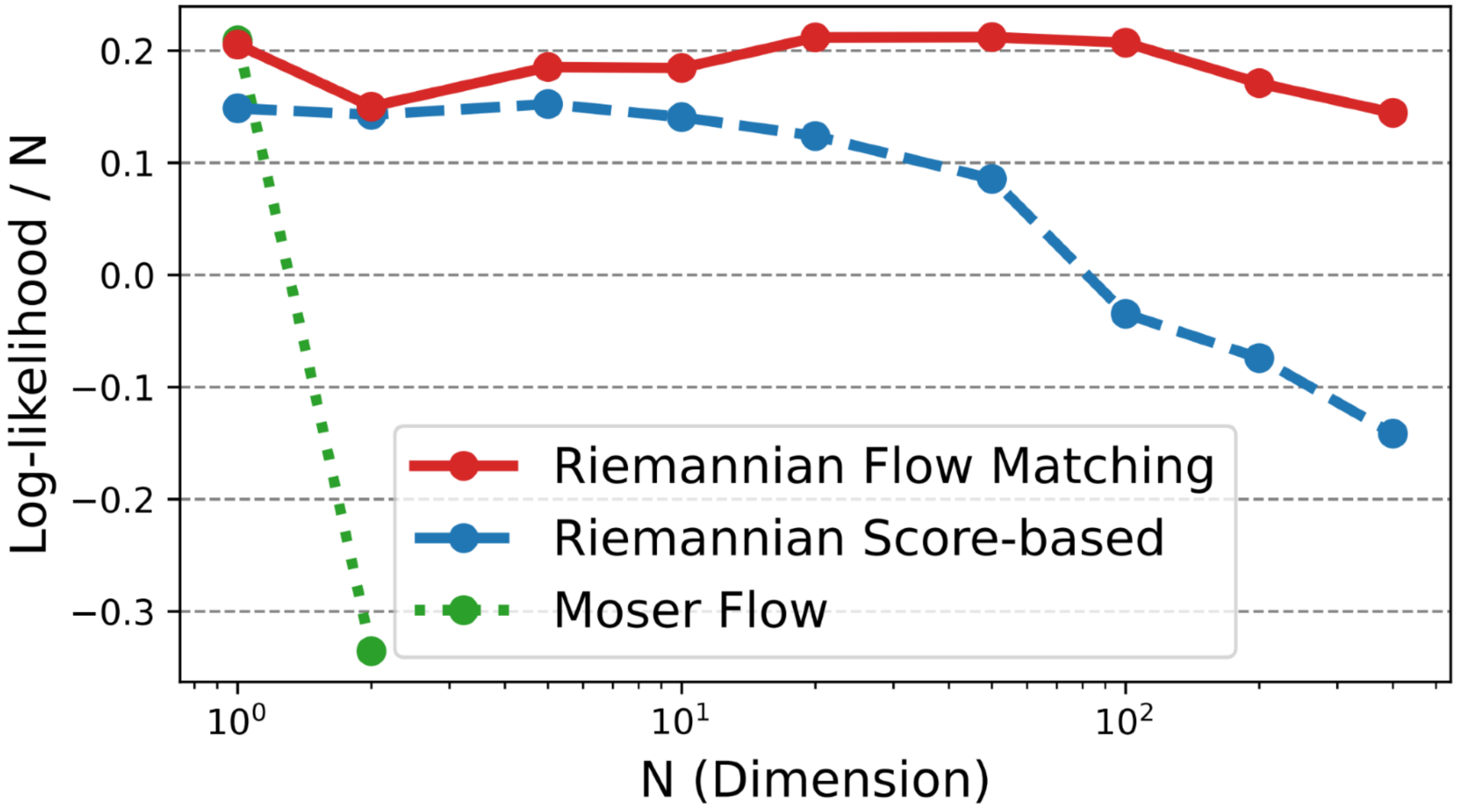
Riemannian Flow vs. Score Matching

	Riemannian Flow Matching	Riemannian Score Matching
Simple manifolds	Simulation Free!	Solve SDE
General manifolds	Solve ODE	Solve SDE
Regression target	$u_t(X_t \mid X_1)$	$\nabla \log p_t(x \mid x_0)$

"Riemannian Score-Based Generative Modelling" De Bortoli et al. (2022)
"Flow Matching on General Geometries" Chen & Lipman (2023)

Riemannian Flow vs. Score Matching





"Riemannian Score-Based Generative Modelling" De Bortoli et al. (2022)
"Flow Matching on General Geometries" Chen & Lipman (2023)

Geometric Flow Matching

Equivariant Flow Matching:

"Fast Point Cloud Generation with Straight Flows" Wu et al. (2022)

"Equivariant flow matching" Klein et al. (2023)

"Equivariant Flow Matching with Hybrid Probability Transport" Song et al. (2023)

"Mosaic-SDF for 3D Generative Models" Yariv et al. (2023)

Riemannian Flow Matching:

"Flow Matching on General Geometries" Chen & Lipman (2023)

"SE(3)-Stochastic Flow Matching for Protein Backbone Generation" Bose et al. (2023)

"Sequence-Augmented SE(3)-Flow Matching For Conditional Protein Backbone Generation" Huguet et al. (2024)

"FlowMM: Generating Materials with Riemannian Flow Matching" Miller et al. (2024)

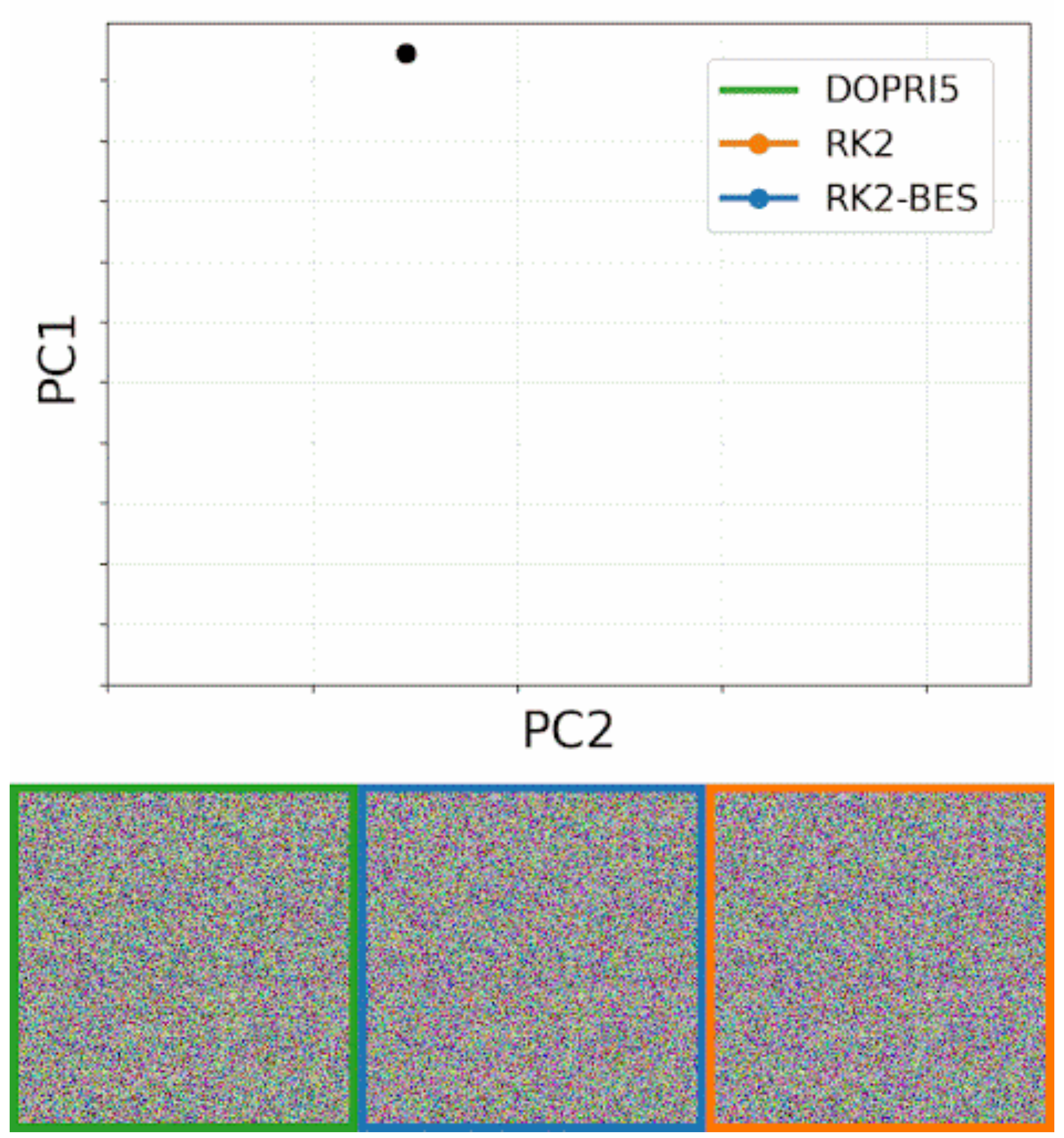
"FlowLLM: Flow Matching for Material Generation with Large Language Models as Base Distributions" Sriram et al. (2024)

"Metric Flow Matching for Smooth Interpolations on the Data Manifold" Kapuśniak et al. (2024)

03 Model Adaptation

You've trained a model. What next?

Faster Sampling

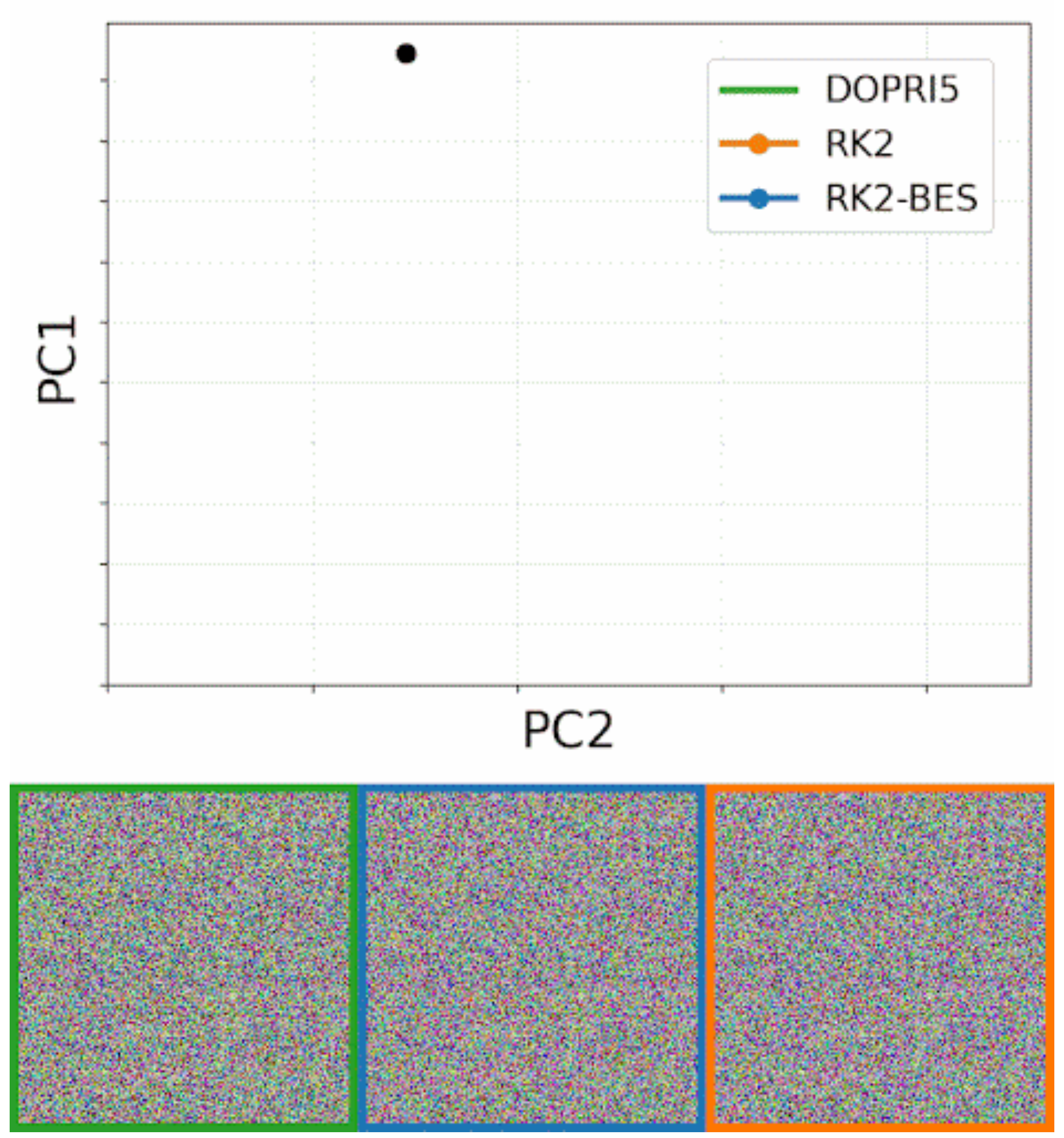


Inverse Problems (Training-Free)

Reward Fine-tuning

You've trained a model. What next?

Faster Sampling

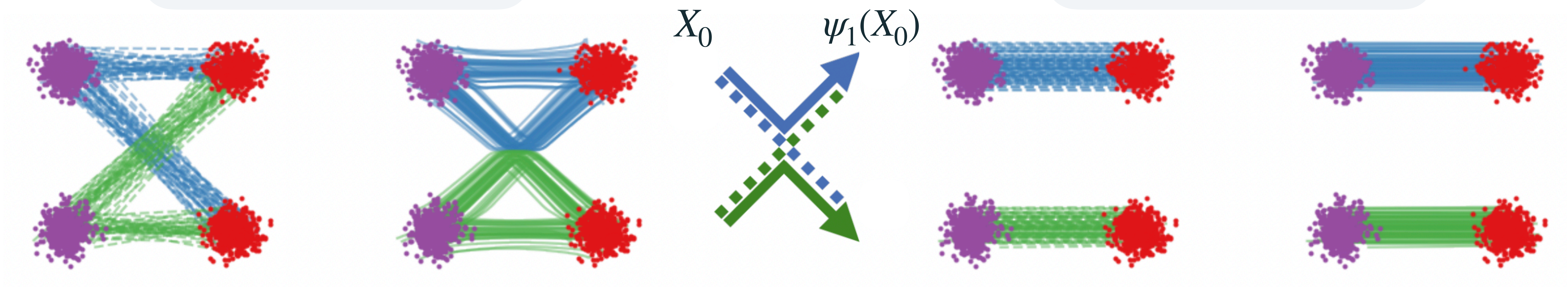


Faster sampling by **straightening the flow**

1-Rectified Flow

(Flow Matching)

2-Rectified Flow



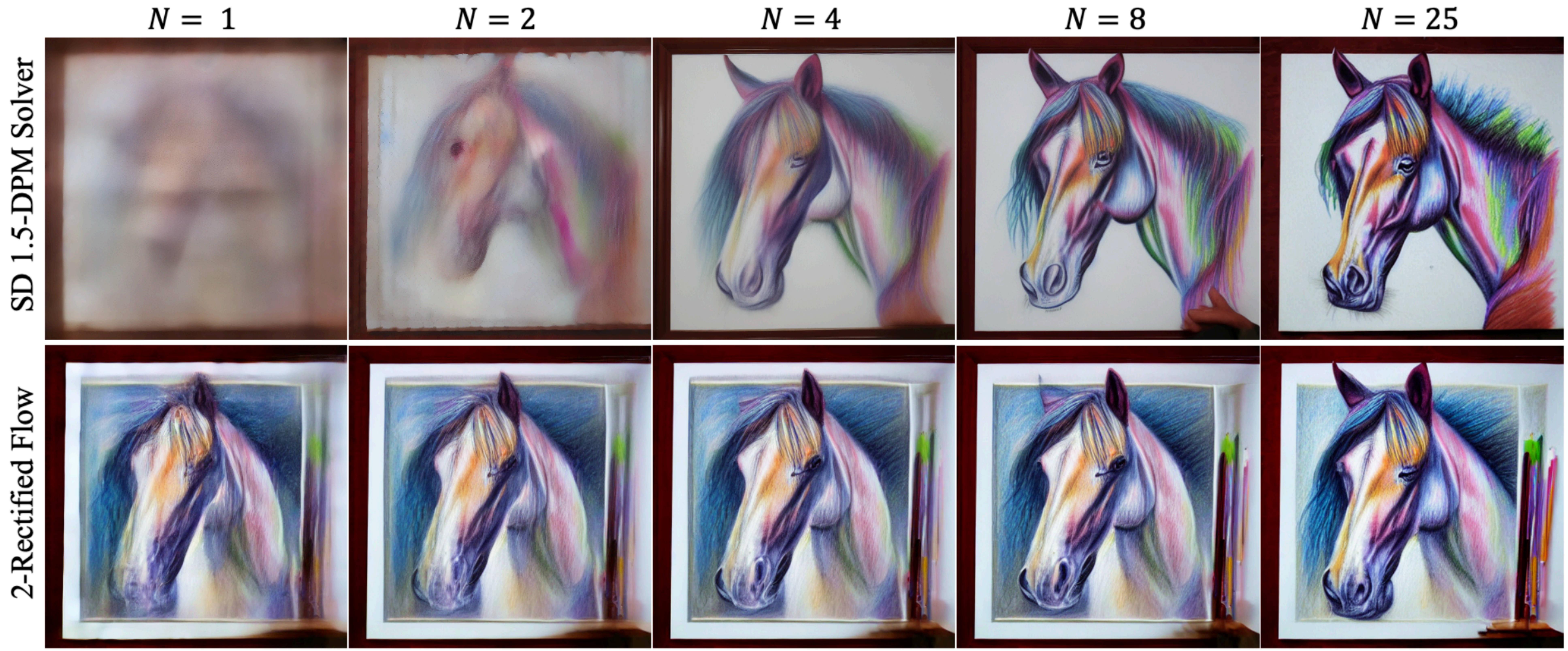
$$\mathcal{L}(\theta) = \mathbb{E}_{t,(X_0,X_1)\sim \pi^{\theta}_{0,1}} \|u^{\theta}_t(X_t) - (X_1 - X_0)\|^2$$

Rectified Flow refits using the pre-trained (noise, data) coupling.

Leads to straight flows.

"Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow" Liu et al. (2022)
"InstaFlow: One Step is Enough for High-Quality Diffusion-Based Text-to-Image Generation" Liu et al. (2022)

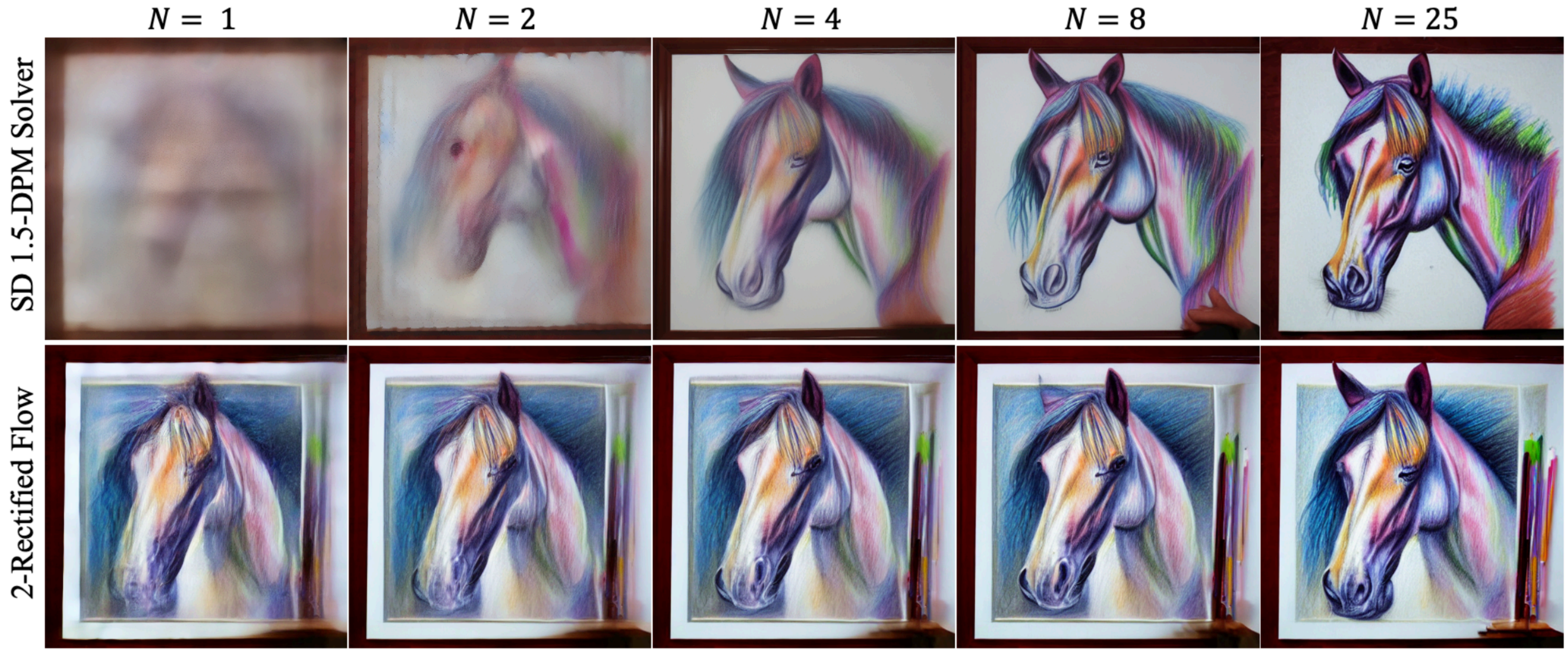
Faster sampling by **straightening the flow**



'Masterpiece color pencil drawing of a horse; bright vivid color'

"InstaFlow: One Step is Enough for High-Quality Diffusion-Based Text-to-Image Generation" Liu et al. (2022)

Faster sampling by **straightening the flow**



'Masterpiece color pencil drawing of a horse; bright vivid color'

Caveat

Enforcing **straightness restricts** the model. Often a **slight drop in sample quality**.

"InstaFlow: One Step is Enough for High-Quality Diffusion-Based Text-to-Image Generation" Liu et al. (2022)

Faster sampling by **self-consistency loss**

Velocity is defined as a limiting quantity:

Euler method is an approximation

$$u_t(x) := \lim_{h \to 0} \frac{X_{t+h} - X_t}{h} \quad \Longrightarrow \quad X_{t+h} \approx X_t + h u_t(X_t)$$

Instead, define **shortcuts** with step size h as **additional** argument:

$$X_{t+h} := X_t + h s_t(X_t, h)$$

Note: $\lim_{h \to 0} s_t(x, h) = u_t(x)$

“One Step Diffusion via Shortcut Models” Frans et al. (2024)

Faster sampling by **self-consistency loss**

Velocity is defined as a limiting quantity:

Euler method is an approximation

$$u_t(x) := \lim_{h \to 0} \frac{X_{t+h} - X_t}{h} \quad \Longrightarrow \quad X_{t+h} \approx X_t + h u_t(X_t)$$

Instead, define **shortcuts** with step size h as **additional** argument:

$$X_{t+h} := X_t + h s_t(X_t, h)$$

Note: $\lim_{h \to 0} s_t(x, h) = u_t(x)$

Shortcuts satisfy a **consistency** property:

$$\begin{aligned} X_{t+2h} &= X_{t+h} + h s_t(X_{t+h}, h) \\ &= X_t + \boxed{h s_t(X_t, h) + h s_t(X_{t+h}, h)} = X_t + \boxed{2h s_t(X_t, 2h)} \end{aligned}$$

"One Step Diffusion via Shortcut Models" Frans et al. (2024)

Faster sampling by **self-consistency loss**

Shortcuts satisfy a **consistency** property:

$$\boxed{s_t(X_t, h)/2 + s_t(X_{t+h}, h)/2 \; = s_t(X_t, 2h)}$$

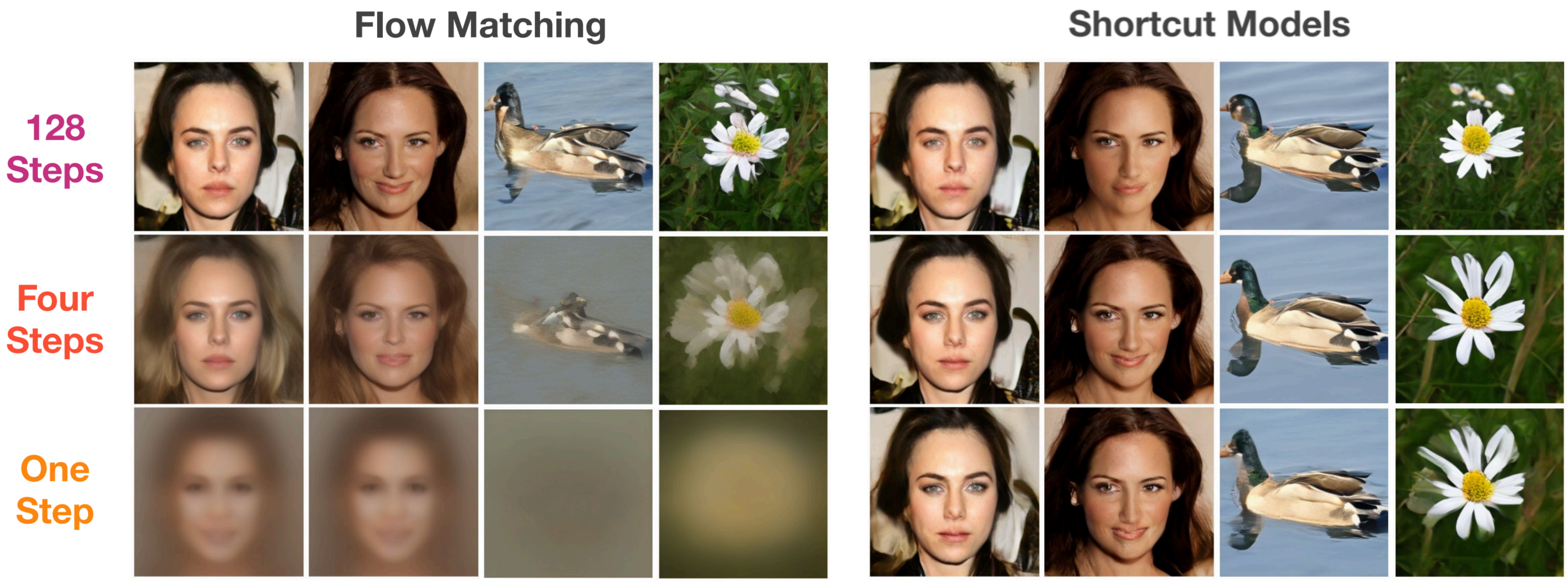
Shortcut models are trained by a mix of Flow Matching & consistency:

$$\mathcal{L}(\theta) = \mathbb{E}_{t,h,X_0,X_1}\left[\underbrace{\|s_t(X_t, 0) - (X_1 - X_0)\|^2}_{\textbf{Flow Matching}} + \underbrace{\|s_t(X_t, 2h) - s^{target}\|^2}_{\textbf{Self-consistency}}\right]$$

where $s^{target} = s_t(X_t, h)/2 + s_t(X_{t+h}, h)/2$

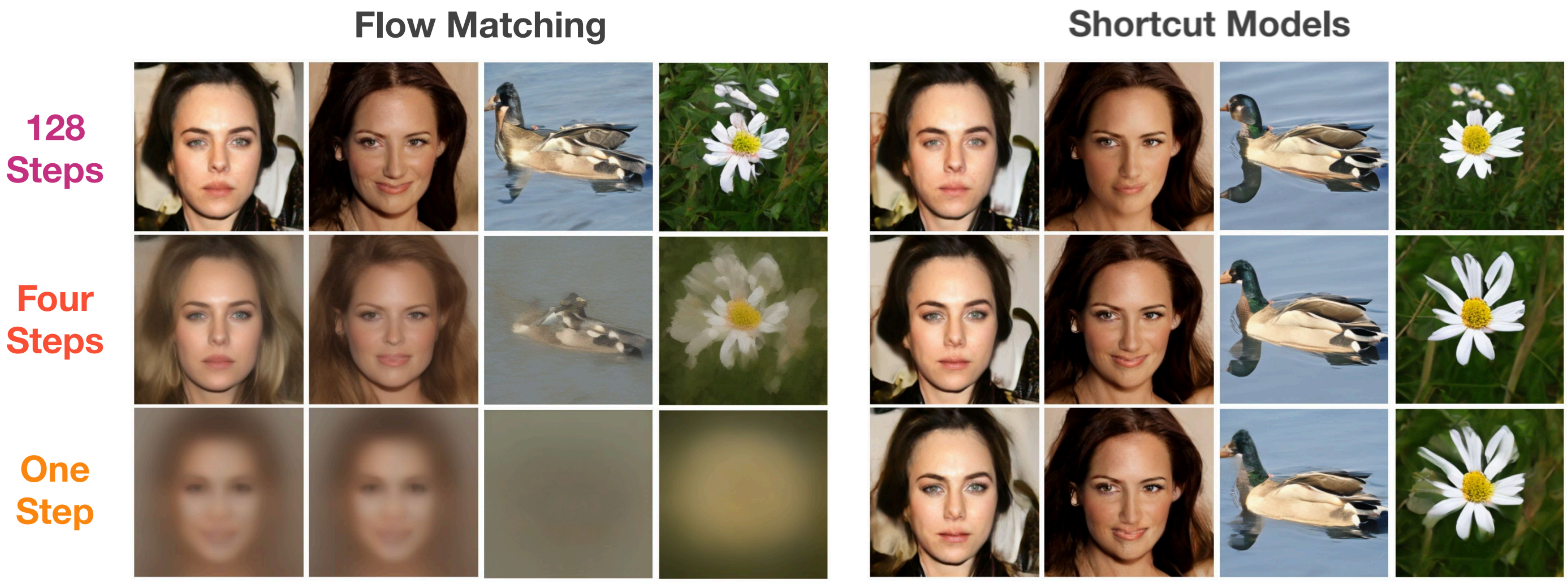
“One Step Diffusion via Shortcut Models” Frans et al. (2024)

Faster sampling by self-consistency loss



“One Step Diffusion via Shortcut Models” Frans et al. (2024)

Faster sampling by self-consistency loss



Caveats

Shortcuts with $h > 0$ **do not work with classifier-free guidance** (CFG).

CFG weight can & must be specified before training.

"One Step Diffusion via Shortcut Models" Frans et al. (2024)

Faster sampling by **only modifying the solver**

Faster sampling by **only modifying the solver**

Can adapt pre-trained models to **different schedulers**.

From original scheduler:

$$X_t = \alpha_t X_1 + \sigma_t X_0$$

To modified scheduler:

$$\bar{X}_t = \bar{\alpha}_t X_1 + \bar{\sigma}_t X_0$$

"Elucidating the design space of diffusion-based generative models" Karras et al. (2023)

"Bespoke Solvers for Generative Flow Models" Shaul et al. (2023)

Faster sampling by only modifying the solver

Can adapt pre-trained models to **different schedulers.**

From original scheduler:

$$X_t = \alpha_t X_1 + \sigma_t X_0$$

To modified scheduler:

$$\bar{X}_t = \bar{\alpha}_t X_1 + \bar{\sigma}_t X_0$$

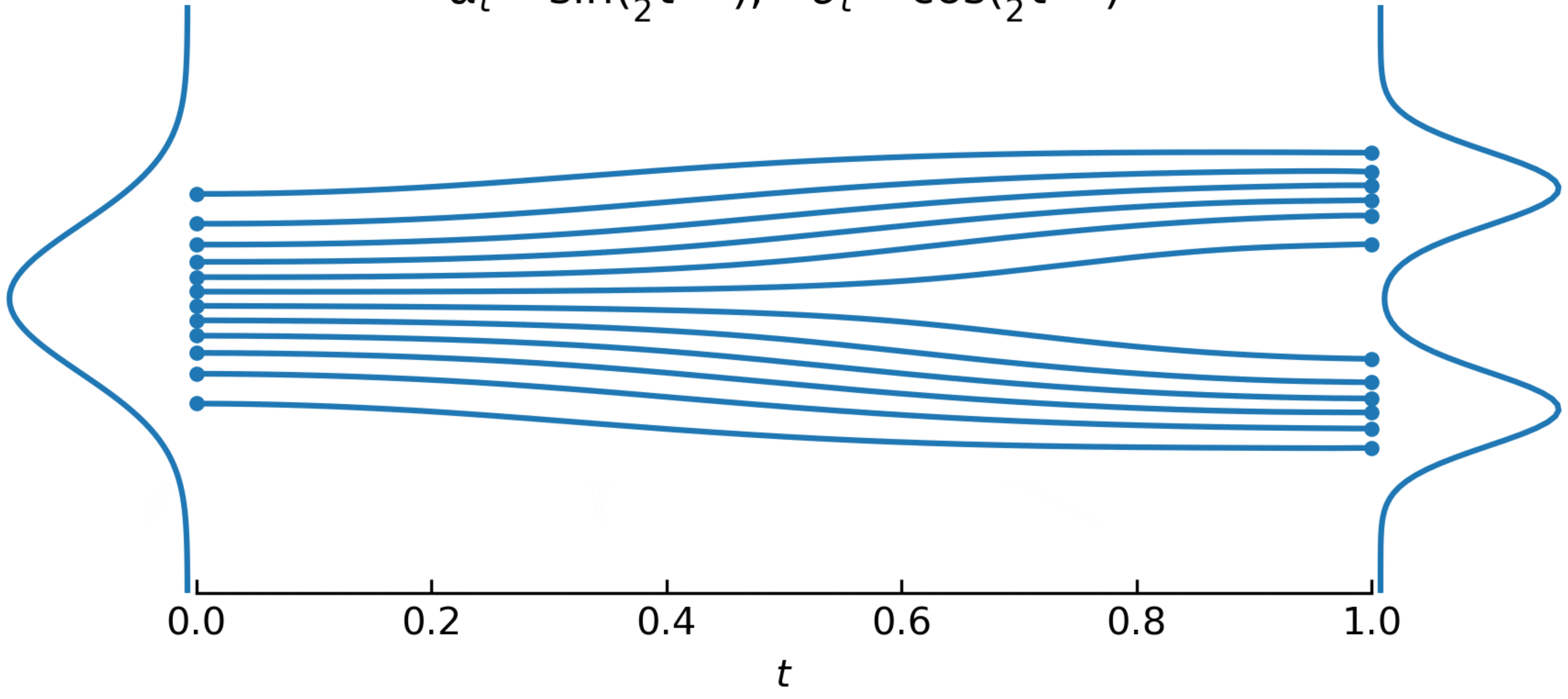
Related by a scaling & time transformation:

$$\bar{X}_r = s_r X_{t_r}$$

where

$$t_r = \mathrm{SNR}^{-1}(\overline{\mathrm{SNR}}(r))$$
$$s_r = \bar{\sigma}_r / \sigma_{t_r}$$

$\bar{\alpha}_t = \sin(\frac{\pi}{2}t^{1.0}), \quad \bar{\sigma}_t = \cos(\frac{\pi}{2}t^{1.0})$

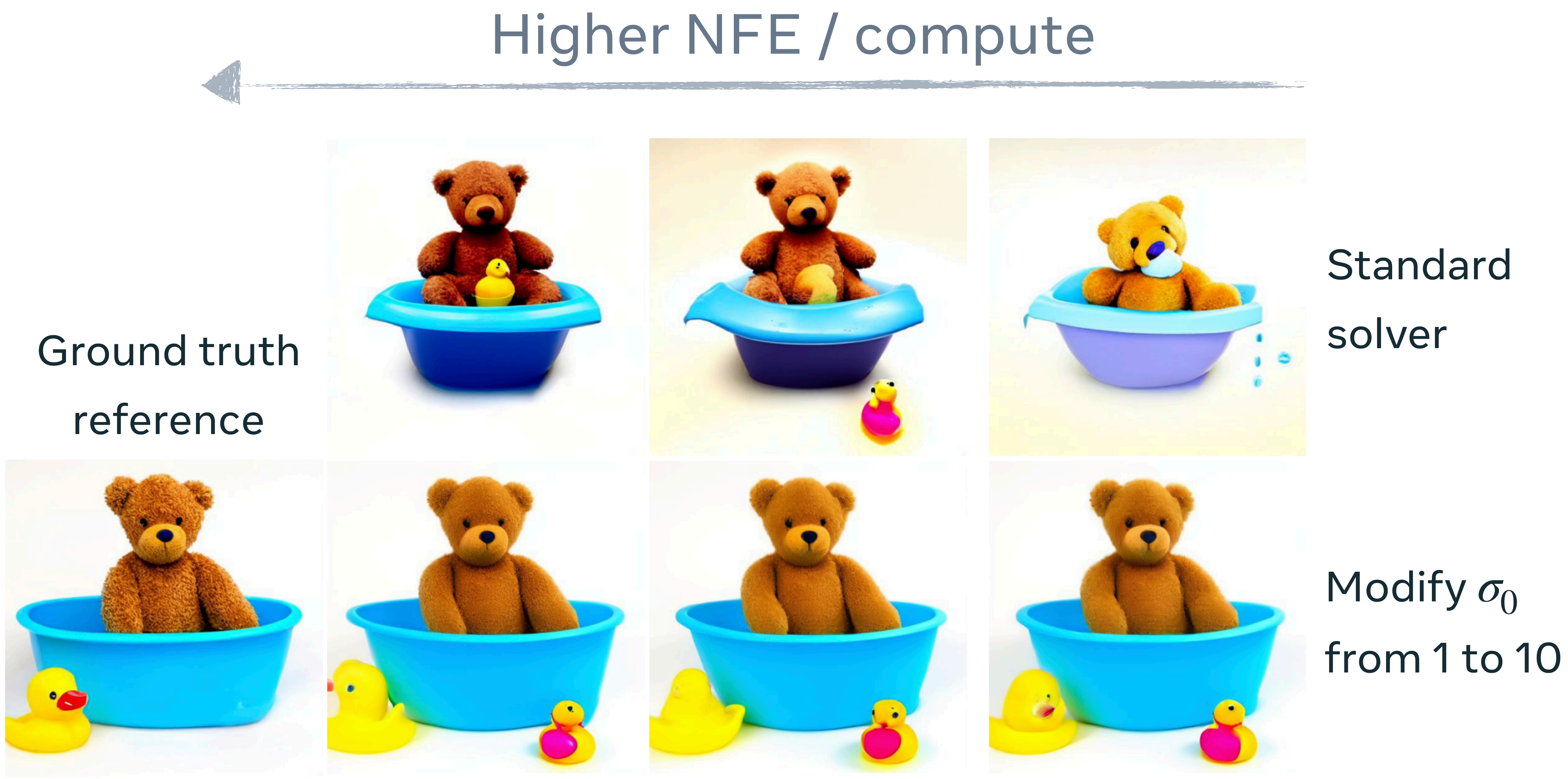


Note: (X_0, X_1) coupling does not change

“Elucidating the design space of diffusion-based generative models” Karras et al. (2023)

“Bespoke Solvers for Generative Flow Models” Shaul et al. (2023)

Faster sampling by **only modifying the solver**



“a teddy bear sitting in a fake bath tub with a rubber ducky””

“Bespoke Solvers for Generative Flow Models” Shaul et al. (2023)
“Bespoke Non-Stationary Solvers for Fast Sampling of Diffusion and Flow Models” Shaul et al. (2024)

Faster sampling by **only modifying the solver**

Bespoke solvers:

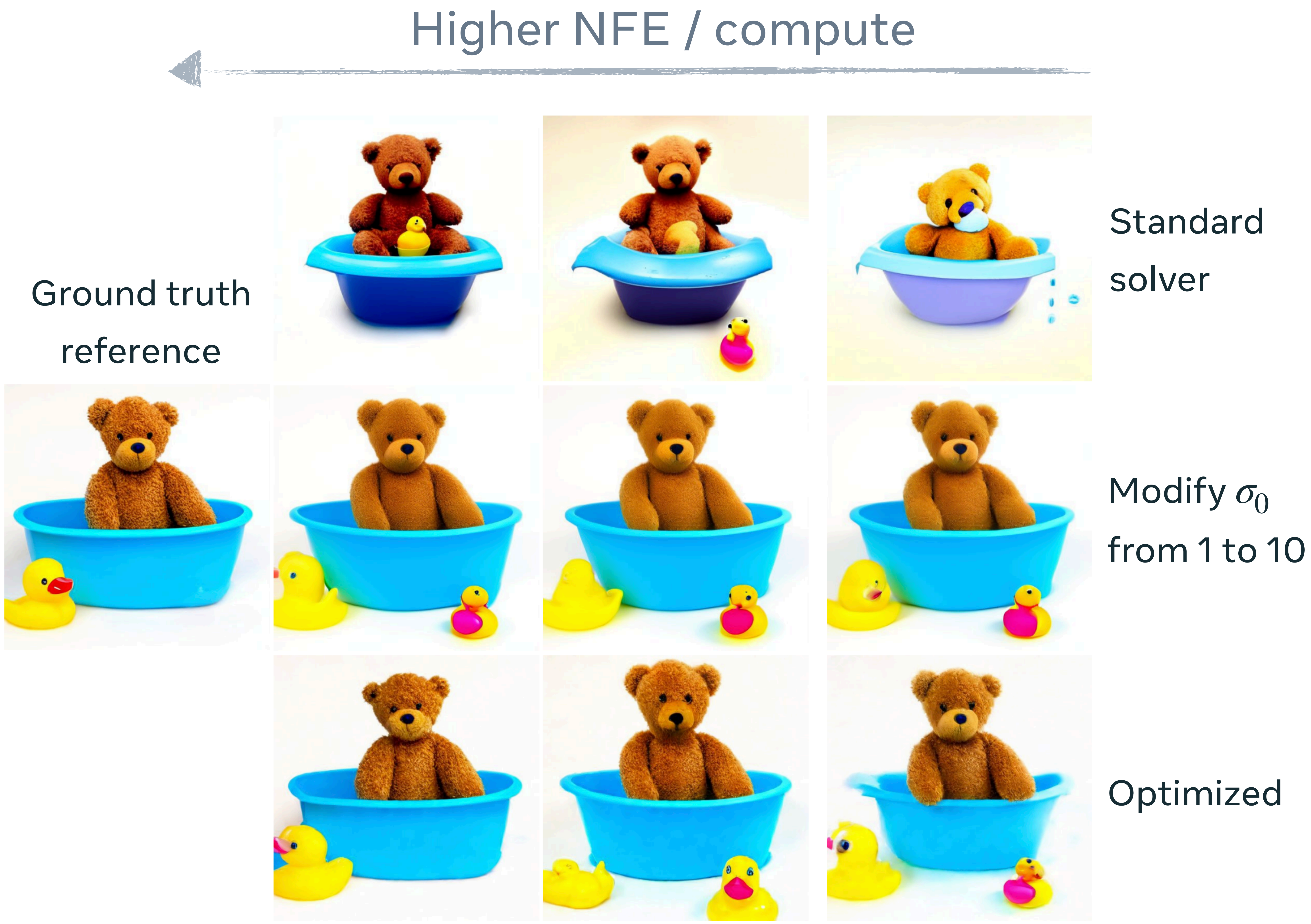
Decouples model & solver.

Model is left unchanged.

Parameterize solver and optimize.

Can be interpreted as finding best scheduler + more.

Solver consistency: sample quality is retained as NFE $\rightarrow \infty$.



“a teddy bear sitting in a fake bath tub with a rubber ducky””

“Bespoke Solvers for Generative Flow Models” Shaul et al. (2023)

“Bespoke Non-Stationary Solvers for Fast Sampling of Diffusion and Flow Models” Shaul et al. (2024)

Faster sampling by **only modifying the solver**

Bespoke solvers:

Decouples model & solver.

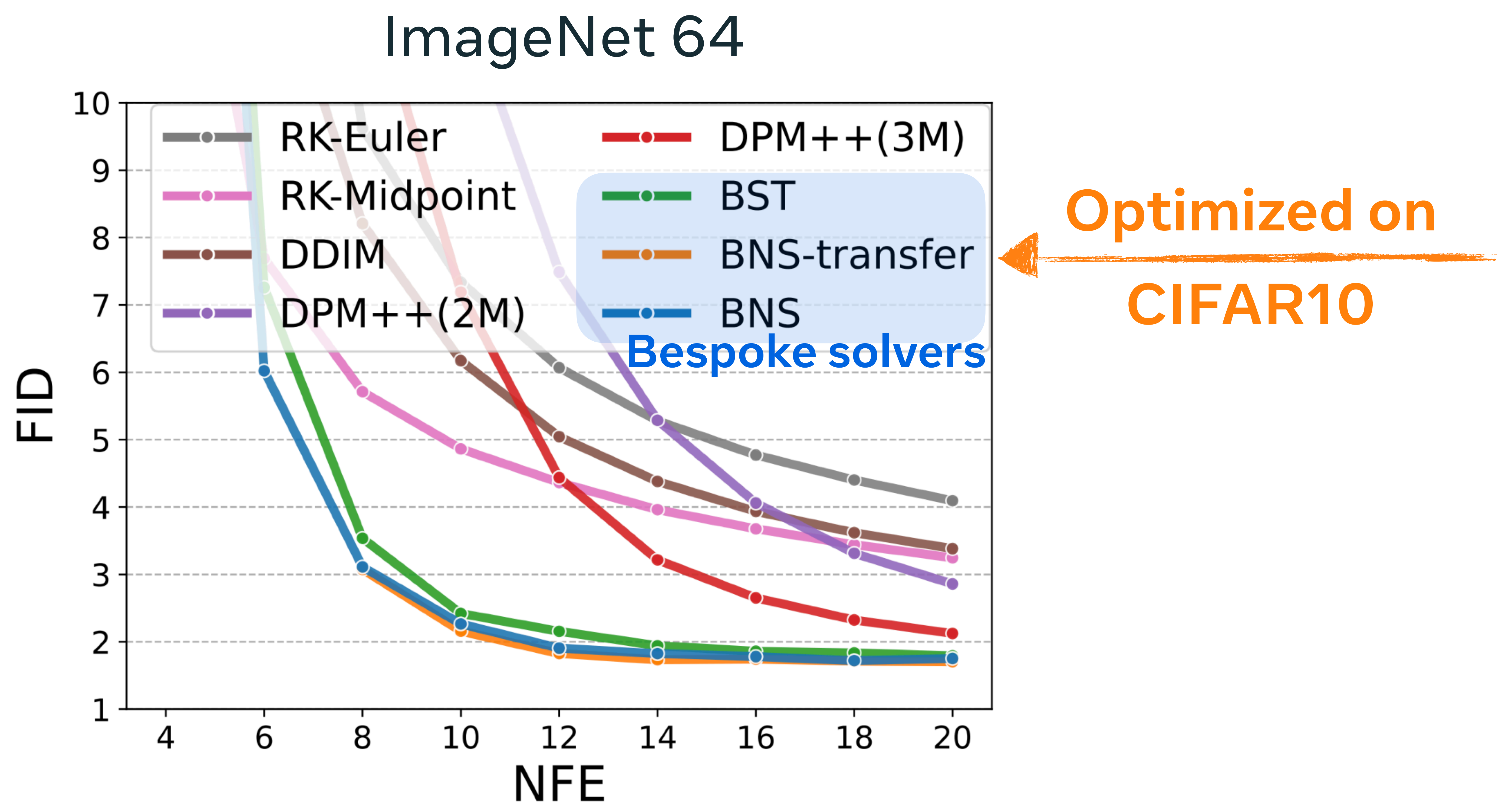
Model is left unchanged.

Parameterize solver and optimize.

Bespoke solvers can **transfer across different data sets and resolutions**.

Caveat

However, **does not reach distillation performance at extremely low NFEs**.



"Bespoke Solvers for Generative Flow Models" Shaul et al. (2023)

"Bespoke Non-Stationary Solvers for Fast Sampling of Diffusion and Flow Models" Shaul et al. (2024)

Faster sampling references

Rectified flows:

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

"InstaFlow: One Step is Enough for High-Quality Diffusion-Based Text-to-Image Generation" Liu et al. (2024)

"Improving the Training of Rectified Flows" Lee et al. (2024)

Consistency & shortcut models:

"Consistency Models" Song et al. (2023)

"Improved Techniques for Training Consistency Models" Song & Dhariwal (2023)

"One Step Diffusion via Shortcut Models" Frans et al. (2024)

Trained & bespoke solvers:

"DPM-Solver-v3: Improved Diffusion ODE Solver with Empirical Model Statistics" Zheng et al. (2023)

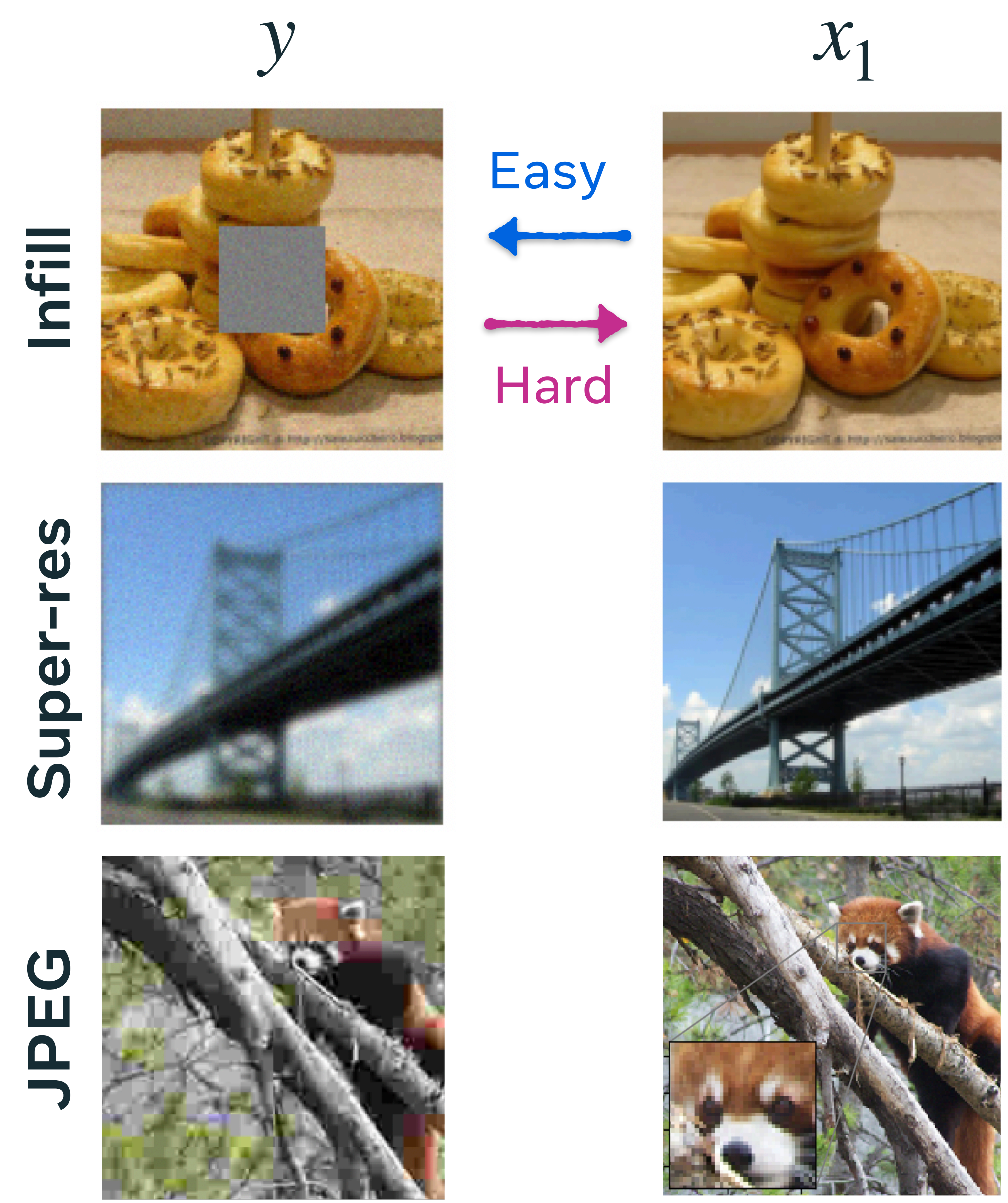
"Bespoke Solvers for Generative Flow Models" Shaul et al. (2023)

"Bespoke Non-Stationary Solvers for Fast Sampling of Diffusion and Flow Models" Shaul et al. (2024)

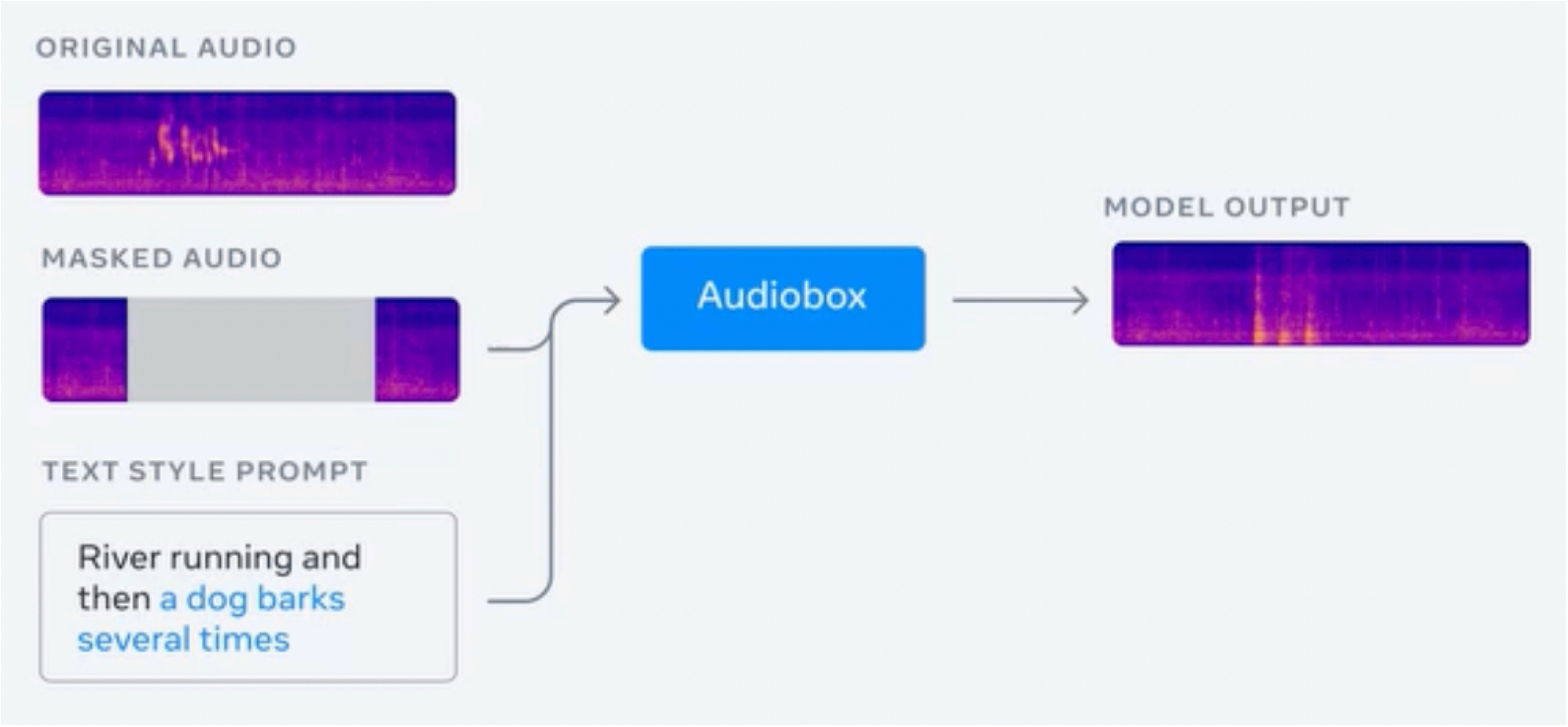
You’ve trained a model. What next?

Inverse Problems
(Training-Free)

Examples of inverse problems



Text-conditional audio infilling



“Audiobox: Unified Audio Generation with Natural Language Prompts” Vyas et al. (2023)

“Pseudoinverse-Guided Diffusion Models for Inverse Problems” Song et al. (2023)

“Training-free Linear Image Inverses via Flows” Pokle et al. (2024)

Solving inverse problems by posterior inference

Formulate as posterior inference given a **pretrained model** and a **known corruption**:

$$p_{1|Y}(x_1 \mid y) \propto p_1(x_1)\, p_{Y|1}(y \mid x_1)$$

“Pseudoinverse-Guided Diffusion Models for Inverse Problems” Song et al. (2023)

“Training-free Linear Image Inverses via Flows” Pokle et al. (2024)

Solving inverse problems by posterior inference

Formulate as posterior inference given a **pretrained model** and a **known corruption**:

$$p_{1|Y}(x_1 \mid y) \propto p_1(x_1)\, p_{Y|1}(y \mid x_1)$$

Velocity that generates the **posterior** can be constructed via “conditional guidance”:

$$u_t(x \mid y) = u_t(x) + \sigma_t^2 \frac{d \log(\alpha_t / \sigma_t)}{dt} \nabla_{x_t} \log p_{Y|t}(y \mid x_t)$$

(unknown)

“Pseudoinverse-Guided Diffusion Models for Inverse Problems” Song et al. (2023)

“Training-free Linear Image Inverses via Flows” Pokle et al. (2024)

Solving inverse problems by posterior inference

Formulate as posterior inference given a **pretrained model** and a **known corruption**:

$$p_{1|Y}(x_1 \mid y) \propto p_1(x_1)\, p_{Y|1}(y \mid x_1)$$

Velocity that generates the **posterior** can be constructed via "conditional guidance":

$$u_t(x \mid y) = u_t(x) + \sigma_t^2 \frac{d \log(\alpha_t/\sigma_t)}{dt} \underbrace{\boxed{\nabla_{x_t} \log p_{Y|t}(y \mid x_t)}}_{\text{(unknown)}}$$

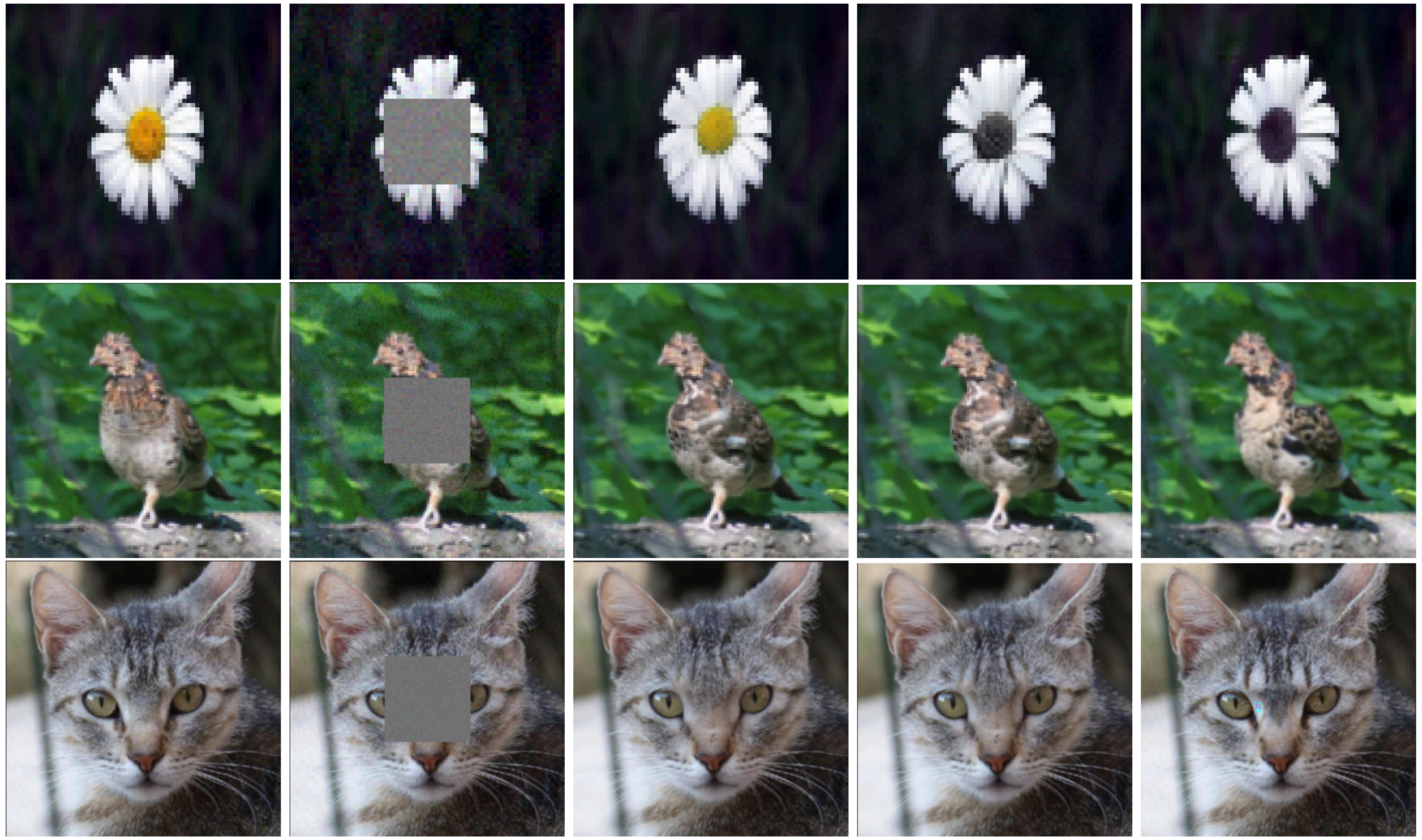
One idea is to replace the unknown score function with a **heuristic approximation**:

$$u_t(x \mid y) \approx u_t(x) + \sigma_t^2 \frac{d \log(\alpha_t/\sigma_t)}{dt} \boxed{\nabla_{x_t} \log p_{Y|t}^{approx}(y \mid x_t)}$$

"Pseudoinverse-Guided Diffusion Models for Inverse Problems" Song et al. (2023)

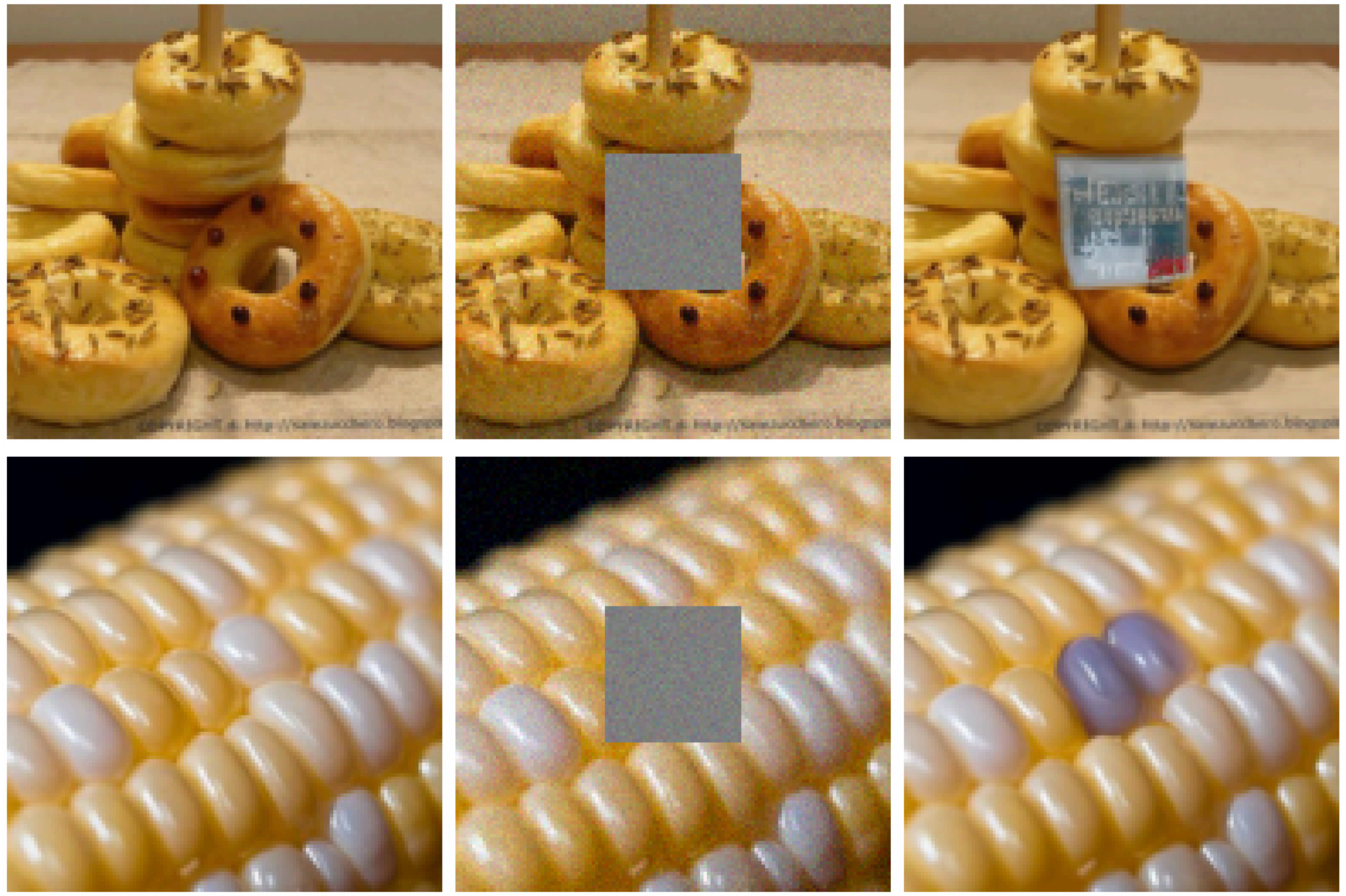
"Training-free Linear Image Inverses via Flows" Pokle et al. (2024)

Solving inverse problems by posterior inference



(a) Reference (b) Distorted (c) OT-ODE (d) VP-ODE (e) ΠGDM

Failure cases



Caveats

Typically requires **known** linear corruption and Gaussian prob path.

Can **randomly fail** due to the heuristic sampling.

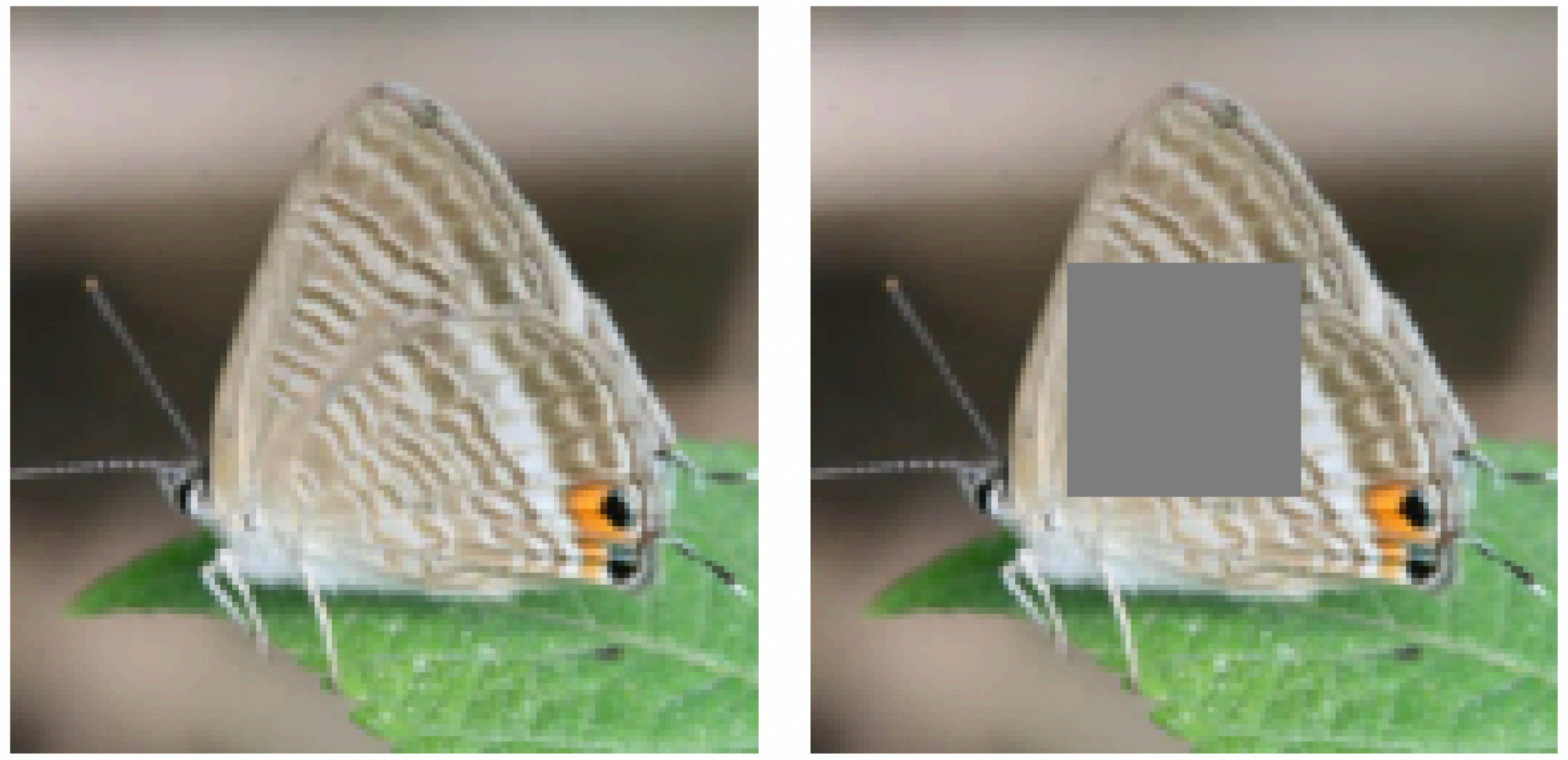
"Pseudoinverse-Guided Diffusion Models for Inverse Problems" Song et al. (2023)

"Training-free Linear Image Inverses via Flows" Pokle et al. (2024)

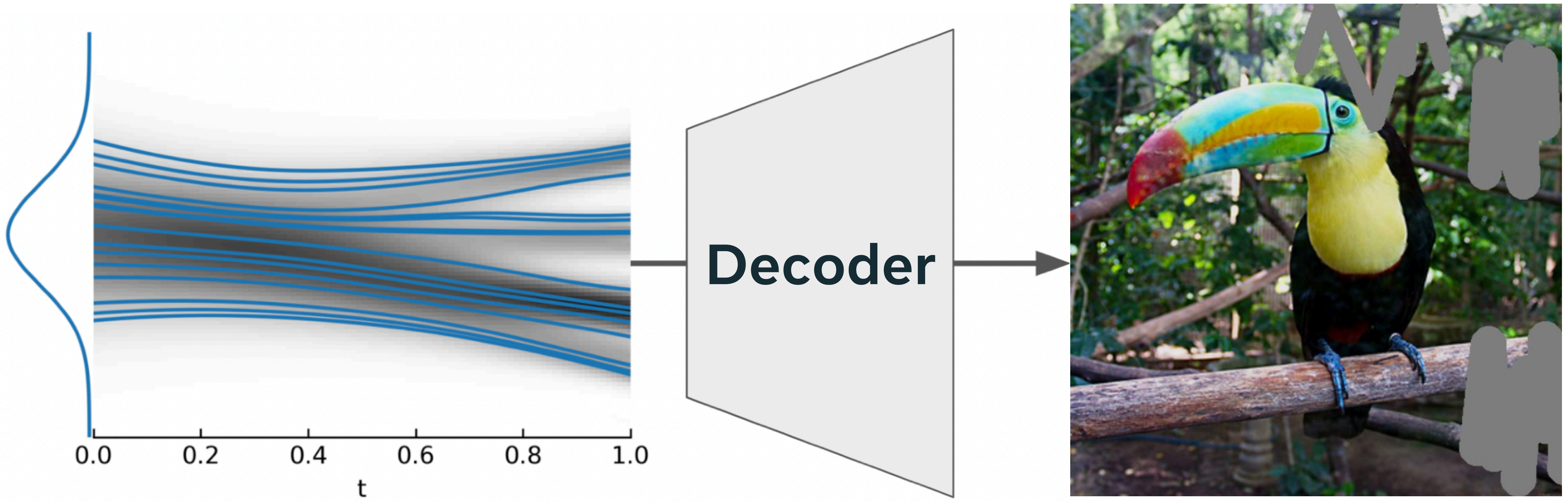
Solving inverse problems by optimizing the source

1. Don't want to rely on **likelihoods / densities**.
2. Have observation y being **nonlinear** in x_1.

Model density is unreliable



Latent FM decoders are nonlinear



"Do Deep Generative Models Know What They Don't Know?" Nalisnick et al. (2018)
"D-Flow: Differentiating through Flows for Controlled Generation" Ben-Hamu et al. (2024)

Solving inverse problems by **optimizing the source**

Inverse problems often formulated as optimization:

$$\min_{x_1} L(x_1) \qquad \textit{e.g.,} \quad L(x) = \| \underbrace{f(x)}_{\text{Corruption fn.}} - \underbrace{y}_{\text{Corrupted obs.}} \|^2$$

"D-Flow: Differentiating through Flows for Controlled Generation" Ben-Hamu et al. (2024)

Solving inverse problems by **optimizing the source**

Inverse problems often formulated as optimization:

$$\min_{x_1} L(x_1) \qquad \textit{e.g.,} \quad L(x) = \| f(x) - y \|^2$$

Corruption fn. Corrupted obs.

Simple idea of using a **pre-trained flow** ψ_1^θ as a diffeomorphism:

(smooth invertible fn.)

$$\min_{x_0} L(\psi_1^\theta(x_0))$$

and optimize the **source variable** x_0.

“D-Flow: Differentiating through Flows for Controlled Generation” Ben-Hamu et al. (2024)

Solving inverse problems by **optimizing the source**

Inverse problems often formulated as optimization:

$$\min_{x_1} L(x_1) \qquad \textit{e.g.,} \quad L(x) = \| f(x) - y \|^2$$

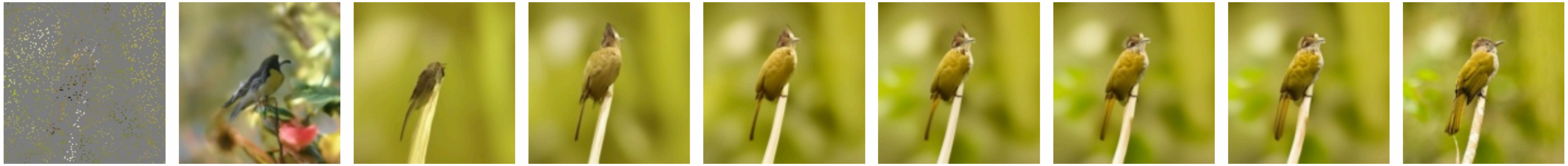
Corruption fn. Corrupted obs.

Simple idea of using a **pre-trained flow** ψ_1^θ as a diffeomorphism:
(smooth invertible fn.)

$$\min_{x_0} L(\psi_1^\theta(x_0))$$

and optimize the **source variable** x_0.

90% masked



Distorted | Initial $x(1)$ | step 2 | step 4 | step 6 | step 8 | step 10 | step 12 | GT

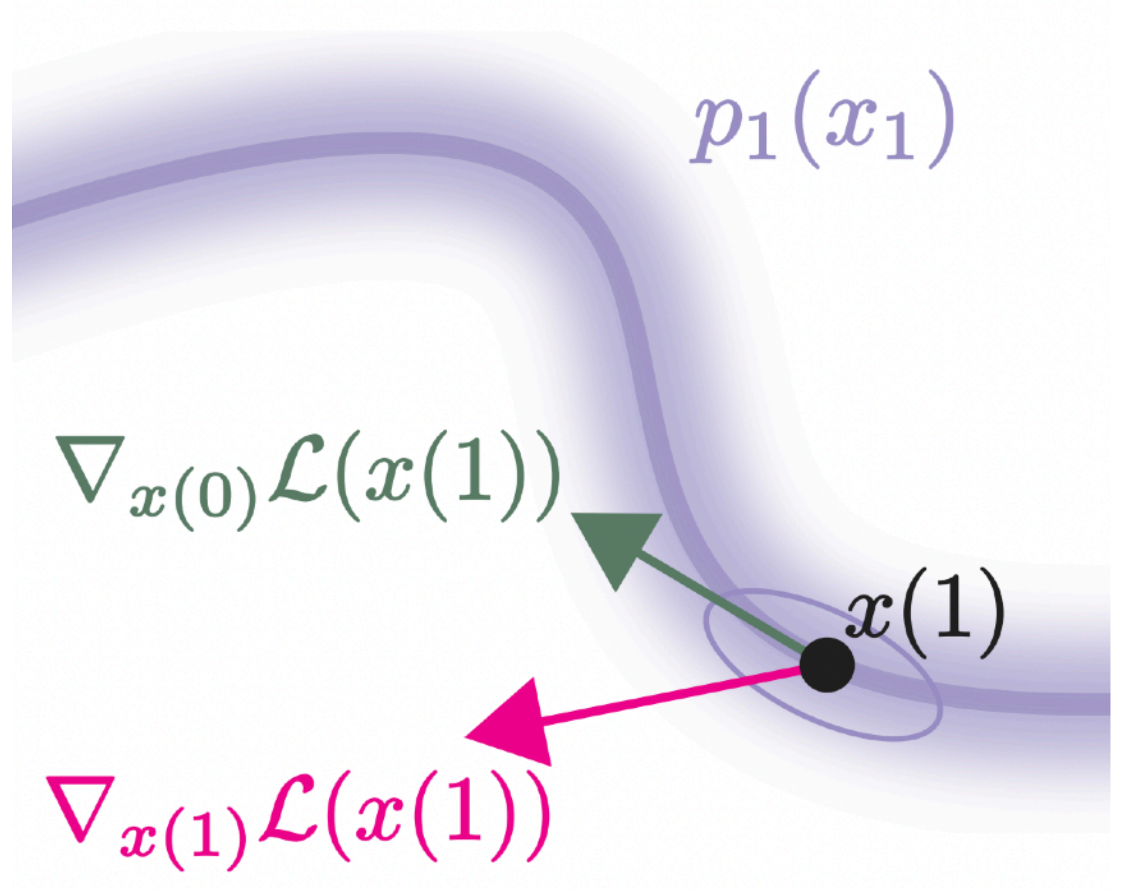
“D-Flow: Differentiating through Flows for Controlled Generation” Ben-Hamu et al. (2024)

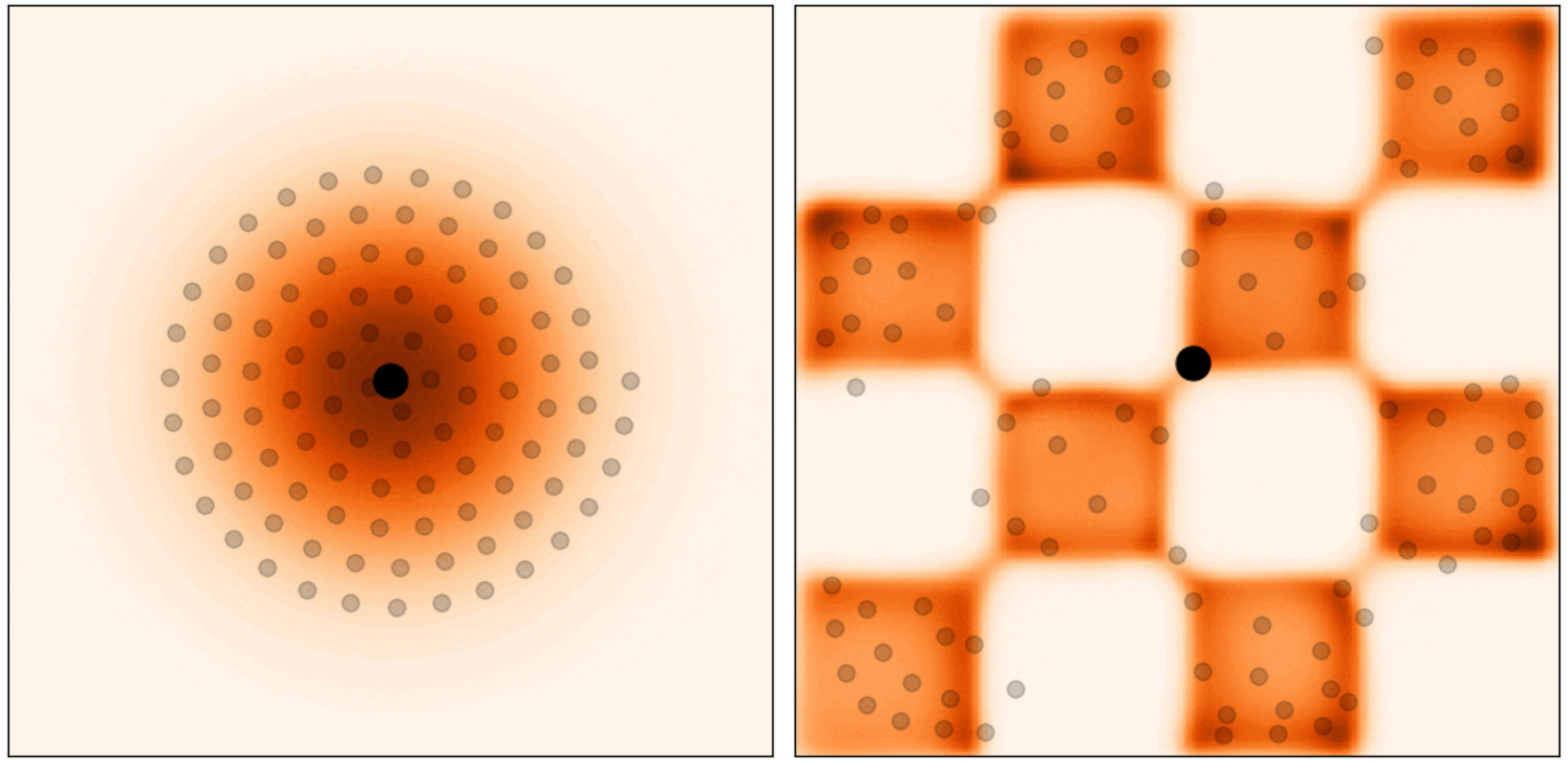
Solving inverse problems by optimizing the source

$$\min_{x_0} L(\psi_1^\theta(x_0))$$

Theory: Jacobian of the flow $\nabla_{x_0}\psi_1^\theta$ projects the gradient along the data manifold.

Intuition: Diffeomorphism enables **mode hopping**!



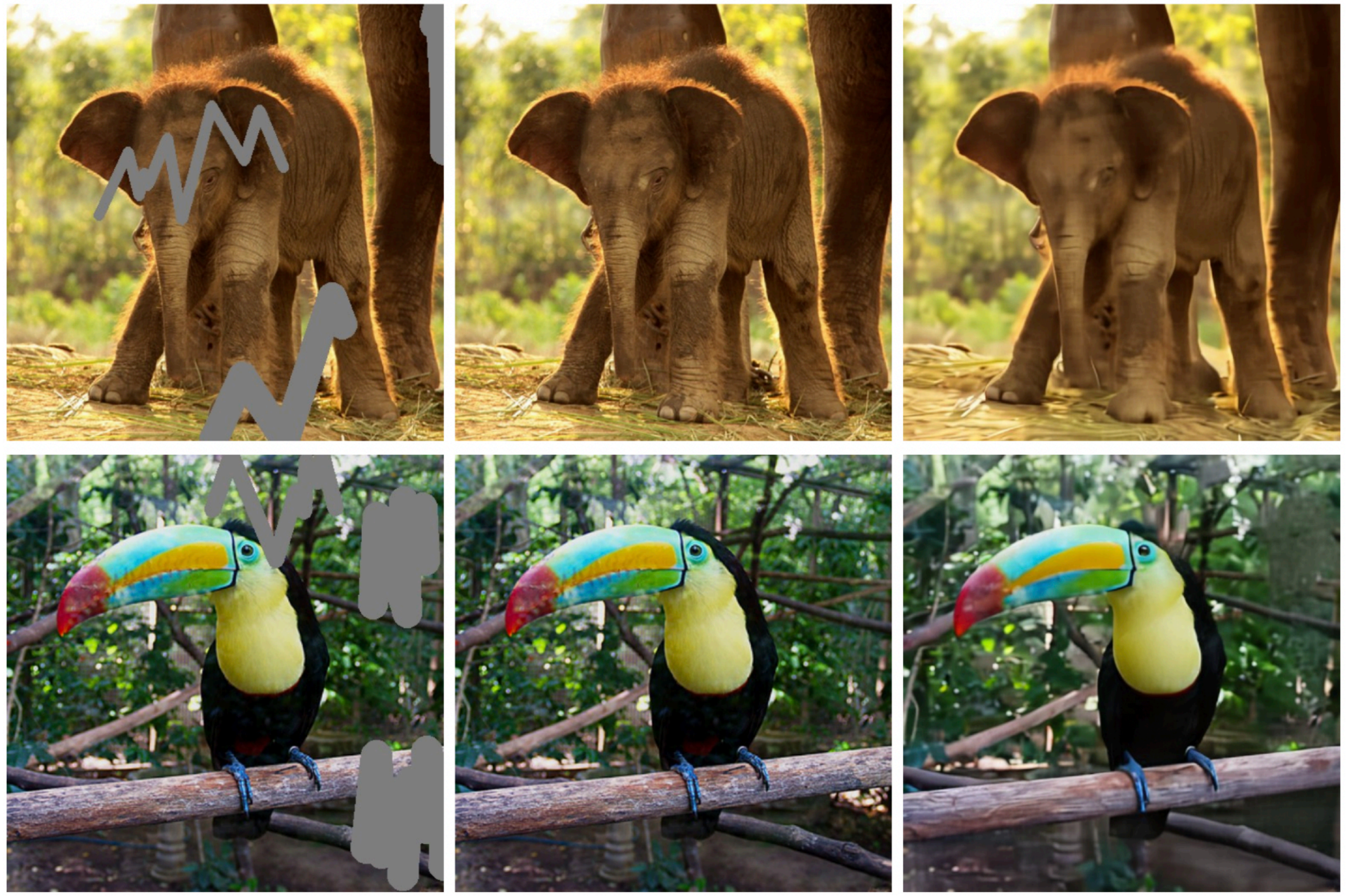


x_0 $\psi_1^\theta(x_0)$

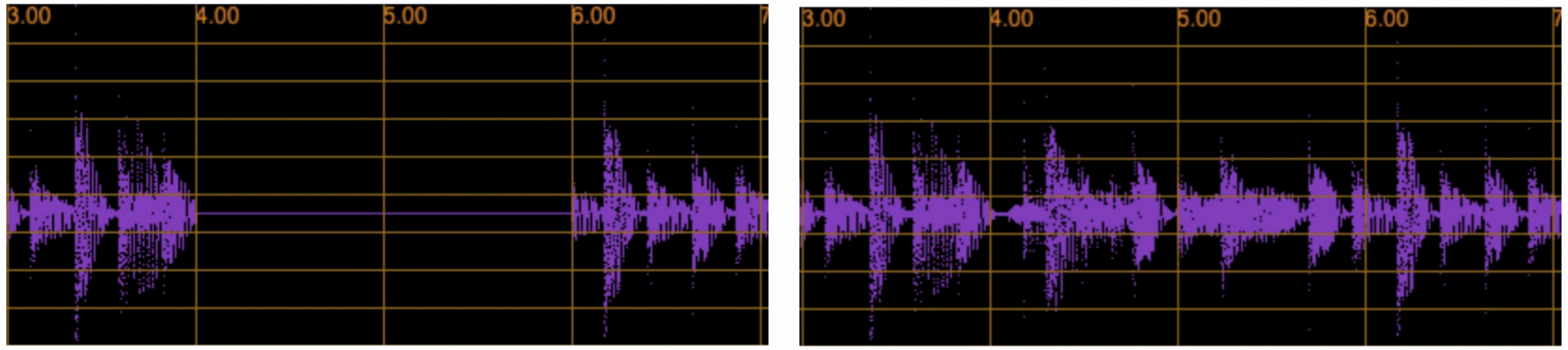
"D-Flow: Differentiating through Flows for Controlled Generation" Ben-Hamu et al. (2024)

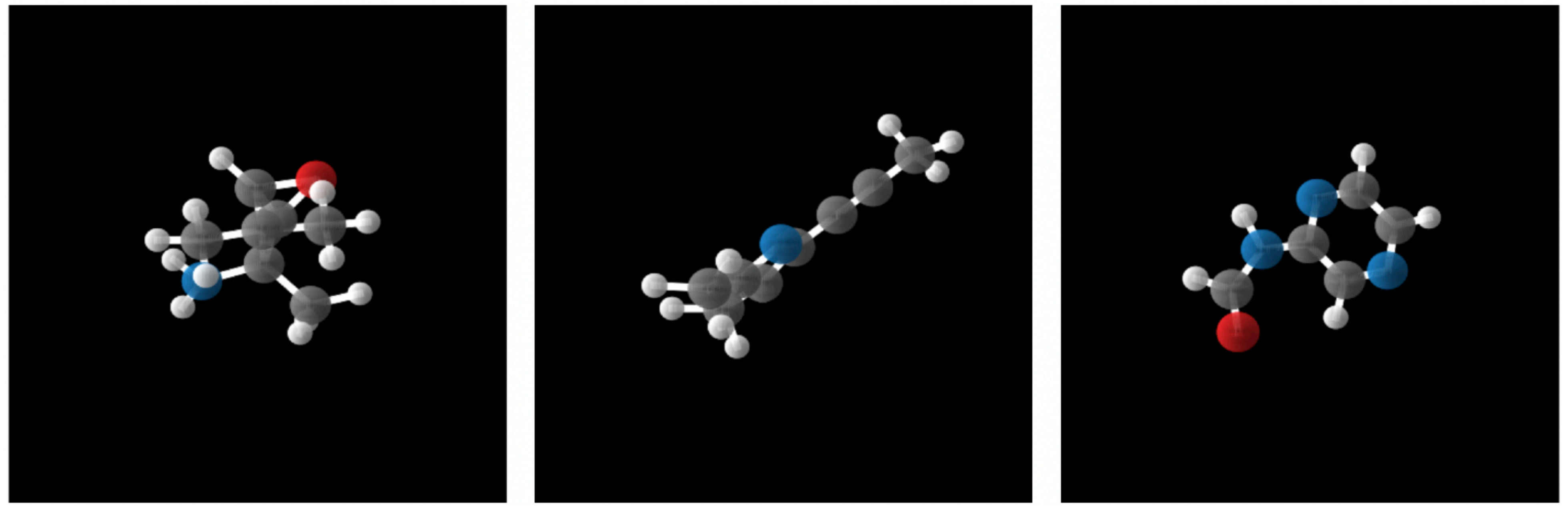
Solving inverse problems by **optimizing the source**

Conditioned on text & corrupted image



Conditioned on text & corrupted audio





Conditioned on molecular properties

Works with **latent** text-conditional models. **Simplicity** allows application in **multiple domains**.

Caveat: Requires **multiple simulations and differentiation of ψ_1^θ.**

"D-Flow: Differentiating through Flows for Controlled Generation" Ben-Hamu et al. (2024)

Inverse problems references

Online sampling methods inspired by posterior inference:

“Diffusion Posterior Sampling for General Noisy Inverse Problems” Chung et al. (2022)

“A Variational Perspective on Solving Inverse Problems with Diffusion Models” Mardani et al. (2023)

“Pseudoinverse-Guided Diffusion Models for Inverse Problems” Song et al. (2023)

“Training-free Linear Image Inverses via Flows” Pokle et al. (2023)

“Practical and Asymptotically Exact Conditional Sampling in Diffusion Models” Wu et al. (2023)

“Monte Carlo guided Diffusion for Bayesian linear inverse problems” Cardoso et al. (2023)

Source point optimization:

“Differentiable Gaussianization Layers for Inverse Problems Regularized by Deep Generative Models" Li (2021)

“End-to-End Diffusion Latent Optimization Improves Classifier Guidance” Wallace et al. (2023)

“D-Flow: Differentiating through Flows for Controlled Generation” Ben-Hamu et al. (2024)

You’ve trained a model. What next?

Reward Fine-tuning

Model fine-tuning drastically enhances quality

Pre-trained	Fine-tuned	Pre-trained	Fine-tuned

a rowboat

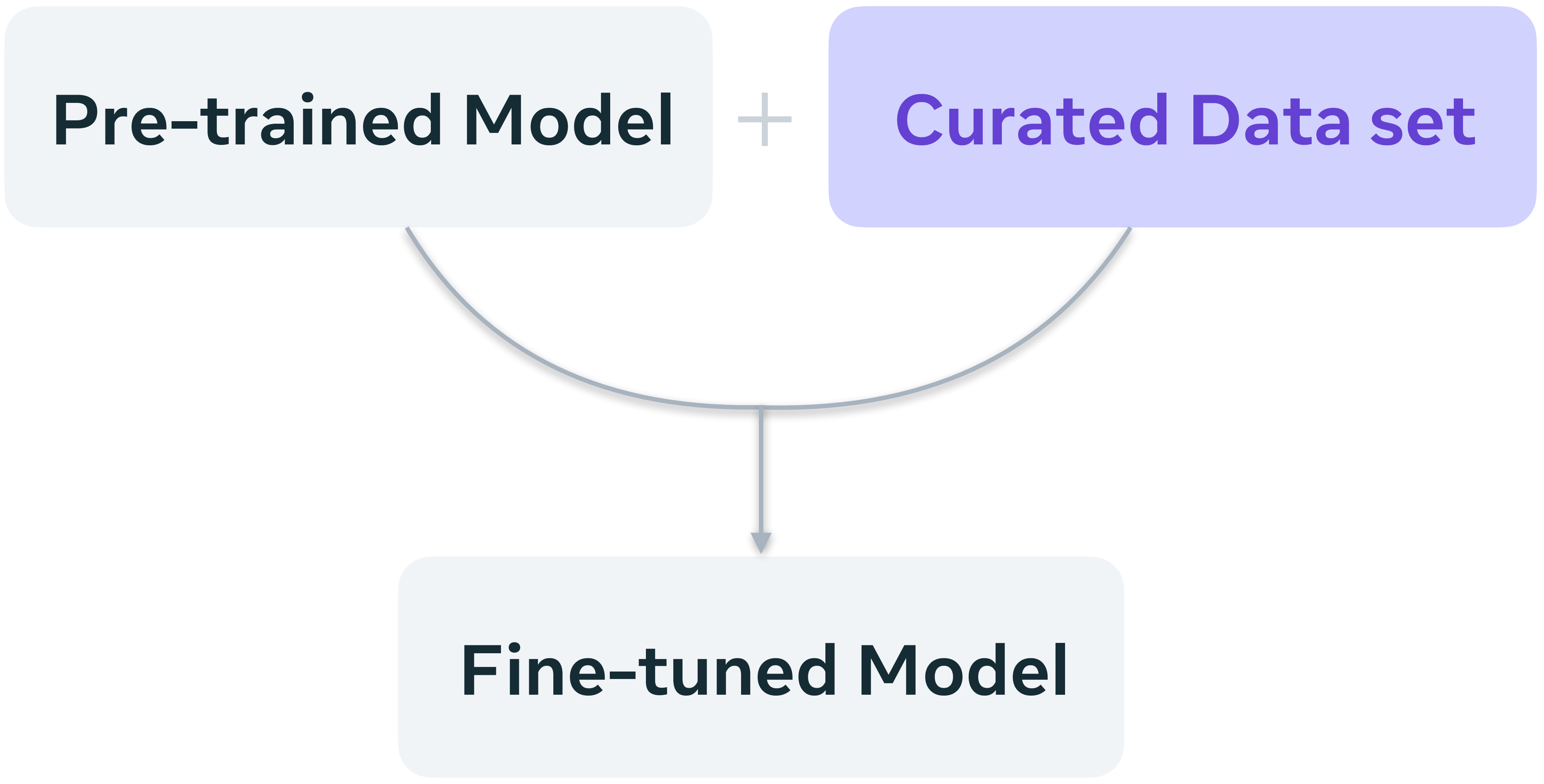
a book

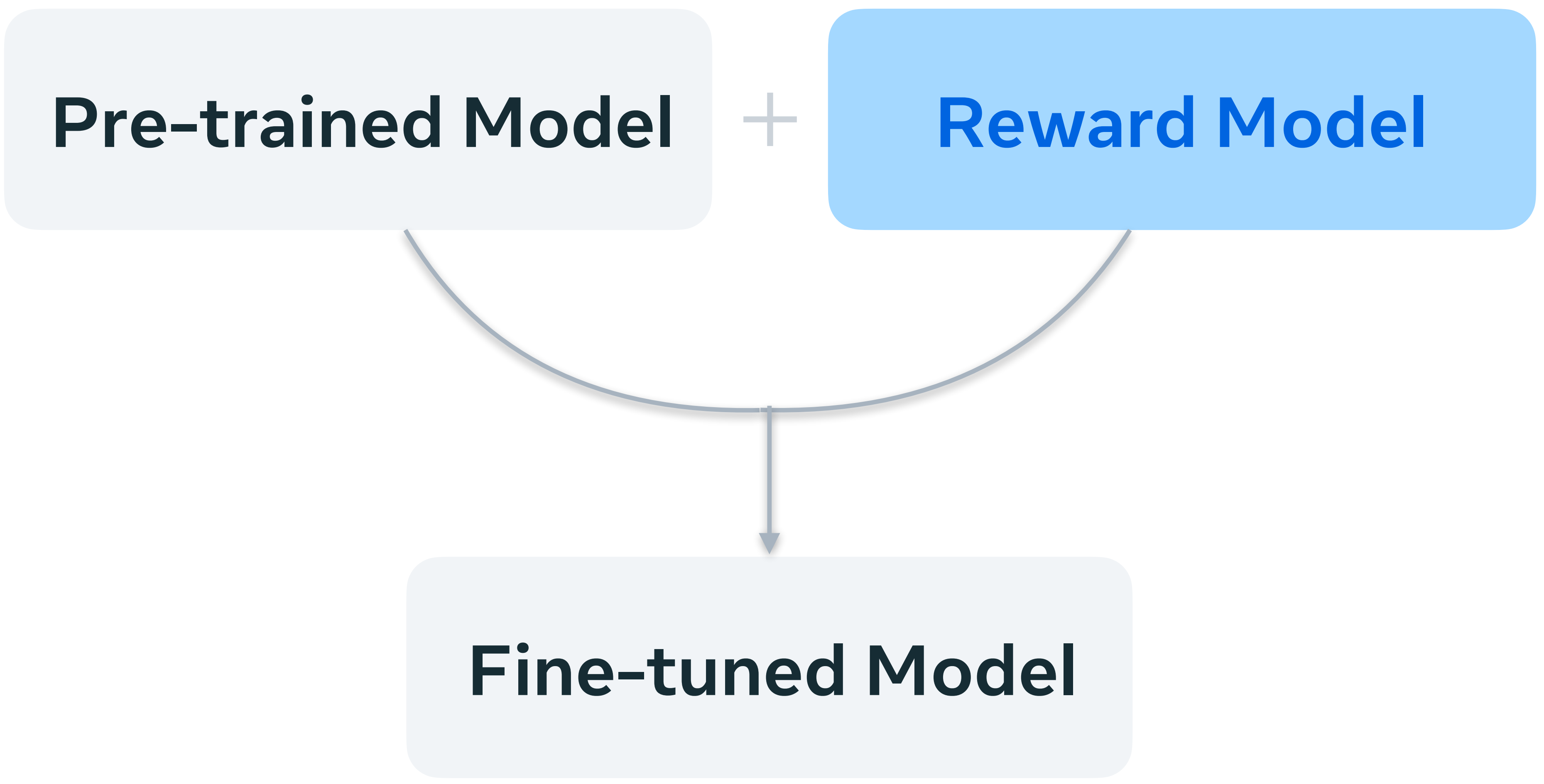
A classic cocktail is placed alongside a napkin

An ink-and-wash artwork depicting a tiger wearing a train conductor's hat and holding onto a skateboard

"Emu: Enhancing Image Generation Models Using Photogenic Needles in a Haystack" Dai et al. (2024)

Data-driven and reward-driven fine-tuning





A lot of focus put into **data set curation** through human filtering.

Can use **human preference models** or **text-to-image alignment**.

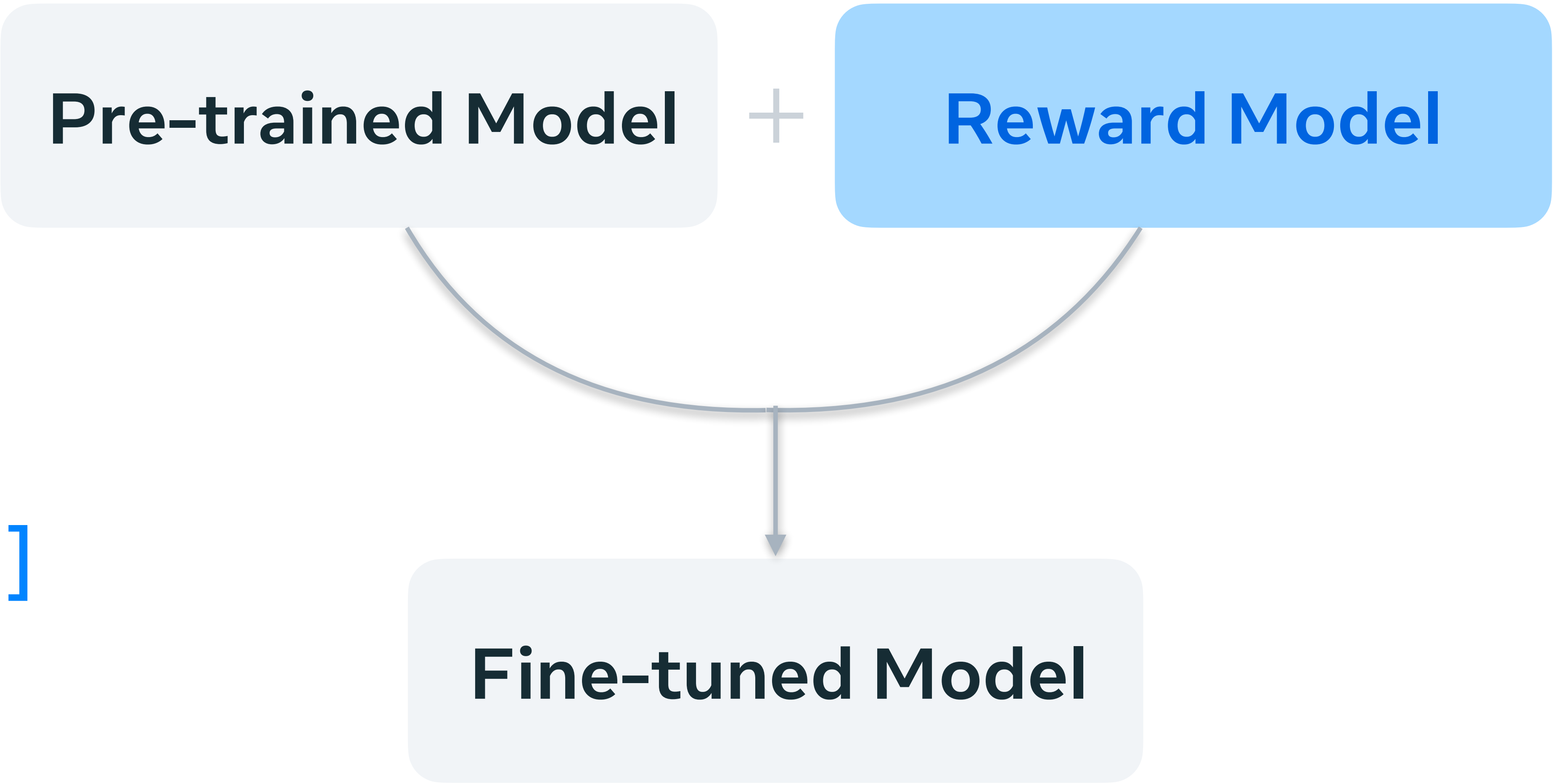
Reward fine-tuning by gradient descent

Initializing with a pre-trained flow model p^θ:

$$\max_{\theta} \mathbb{E}_{X_1 \sim p^\theta} \left[r(X_1) \right]$$

Optimize the reward model with RL [Black et al. 2023]

or direct gradients [Xu et al. 2023, Clark et al. 2024].



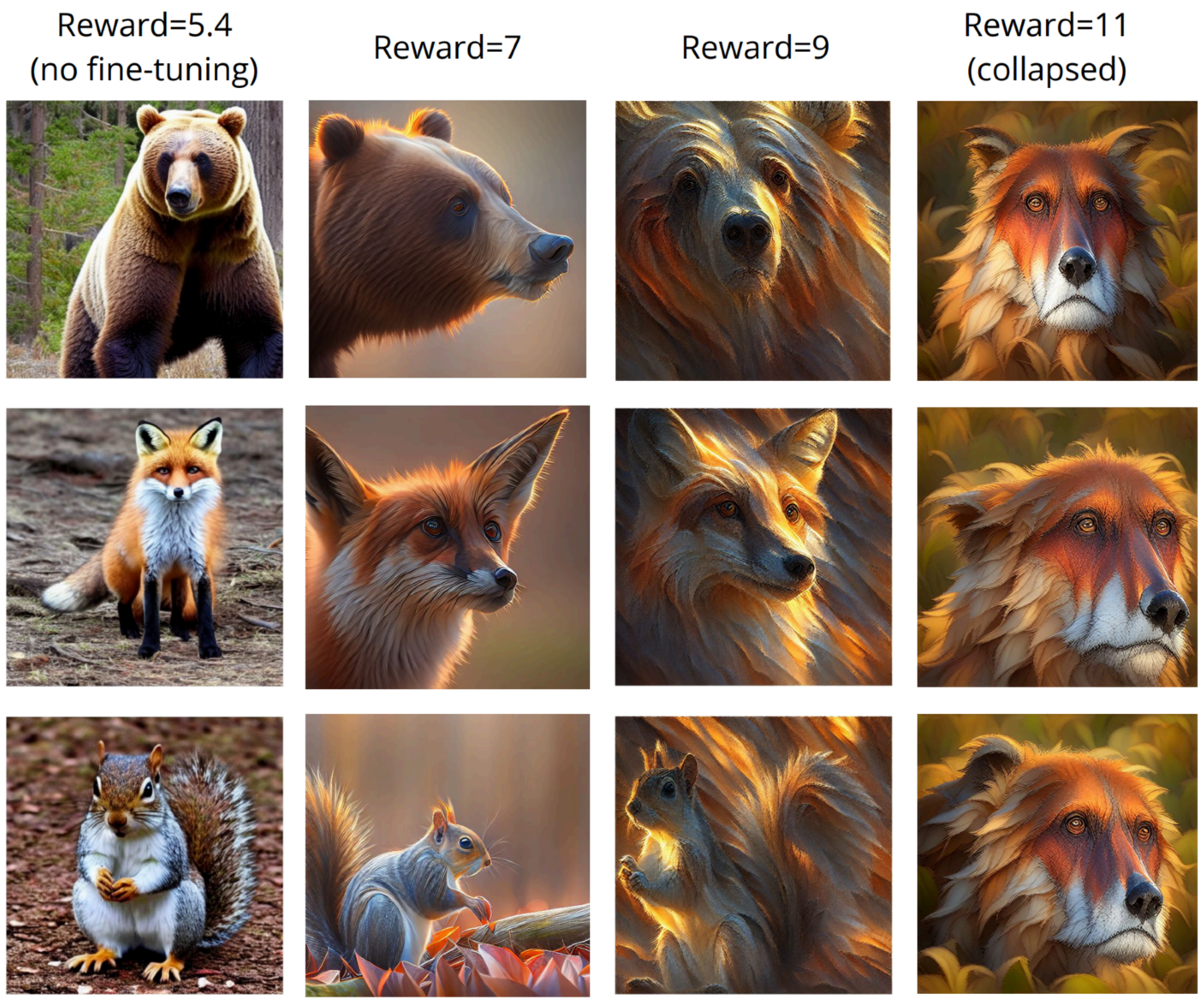
“A painting of a deer” [Clark et al. 2024]

“Training diffusion models with reinforcement learning” Black et al. (2023)

“Imagereward: Learning and evaluating human preferences for text-to-image generation.” Xu et al. (2023)

“Directly fine-tuning diffusion models on differentiable rewards.” Clark et al. (2024)

Reward fine-tuning by gradient descent



Caveats

Requires using **LoRA** to heuristically stay close to the original model.

Still relatively easy to **over-optimize** reward models; **"reward hacking"**.

"Directly fine-tuning diffusion models on differentiable rewards." Clark et al. (2024)

Reward fine-tuning by stochastic optimal control

Reinforcement learning from human feedback (RLHF) from **the LLM literature** typically target the **tilted distribution**:

$$\boxed{p^*(X_1) \propto p^{base}(X_1) \exp(r(X_1))}$$

Based on a **pre-trained (base) model** and a **reward model**.

“Fine-tuning of continuous-time diffusion models as entropy regularized control” Uehara et al. (2024)

“Adjoint matching: Fine-tuning flow and diffusion generative models with memoryless stochastic optimal control” Domingo-Enrich et al. (2024)

Reward fine-tuning by stochastic optimal control

Reinforcement learning from human feedback (RLHF) from **the LLM literature** typically target the **tilted distribution**:

$$\boxed{p^*(X_1) \propto p^{base}(X_1)\exp(r(X_1))}$$

Based on a **pre-trained (base) model** and a **reward model**.

One idea: use a **KL regularization** over the path $X_{(0,1)}$:

$$\max_\theta \mathbb{E}_{X_0 \sim p_0} \mathbb{E}_{X_{0:1} \sim p^\theta(X_1|X_0)} \left[r(X_1) - D_{KL}(p^\theta(X_{(0,1)} \,|\, X_0) \,||\, p^{base}(X_{(0,1)} \,|\, X_0)) \right]$$

“Fine-tuning of continuous-time diffusion models as entropy regularized control” Uehara et al. (2024)

“Adjoint matching: Fine-tuning flow and diffusion generative models with memoryless stochastic optimal control” Domingo-Enrich et al. (2024)

Reward fine-tuning by stochastic optimal control

Reinforcement learning from human feedback (RLHF) from **the LLM literature** typically target the **tilted distribution**:

$$\boxed{p^*(X_1) \propto p^{base}(X_1)\exp(r(X_1))}$$

Based on a **pre-trained (base) model** and a **reward model**.

One idea: use a **KL regularization** over the path $X_{(0,1)}$:

$$\max_{\theta} \mathbb{E}_{X_0 \sim p_0} \mathbb{E}_{X_{0:1} \sim p^{\theta}(X_1 | X_0)} \left[r(X_1) - D_{KL}(p^{\theta}(X_{(0,1)} | X_0) \,||\, p^{base}(X_{(0,1)} | X_0)) \right]$$

However, since X_0 and X_1 are **dependent**, this results in:

$$p^*(X_{(0,1)}) = p^{base}(X_{(0,1)})\exp(r(X_1) + V(X_0))$$

$$V(x) = \mathbb{E}_{p^{base}}[r(X_1) \,|\, X_0 = x]$$

"value function bias"

"Fine-tuning of continuous-time diffusion models as entropy regularized control" Uehara et al. (2024)

"Adjoint matching: Fine-tuning flow and diffusion generative models with memoryless stochastic optimal control" Domingo-Enrich et al. (2024)

Reward fine-tuning by stochastic optimal control

Reinforcement learning from human feedback (RLHF) from **the LLM literature** typically target the **tilted distribution**:

$$\boxed{p^*(X_1) \propto p^{base}(X_1)\exp(r(X_1))}$$

Based on a **pre-trained (base) model** and a **reward model**.

Intuition: Both initial noise $p(X_0)$ and the model u_t^{base} affect $p^{base}(X_1)$.

[Uehara et al. 2024] proposes to **learn the optimal source distribution** $p^*(X_0)$.

[Domingo-Enrich et al. 2024] proposes to **remove the dependency** between X_0, X_1.

$$p^*(X_{(0,1)}) = p^{base}(X_{(0,1)})\exp(r(X_1) + const.) \Longrightarrow p^*(X_1) \propto p^{base}(X_1)\exp(r(X_1))$$

"Fine-tuning of continuous-time diffusion models as entropy regularized control" Uehara et al. (2024)

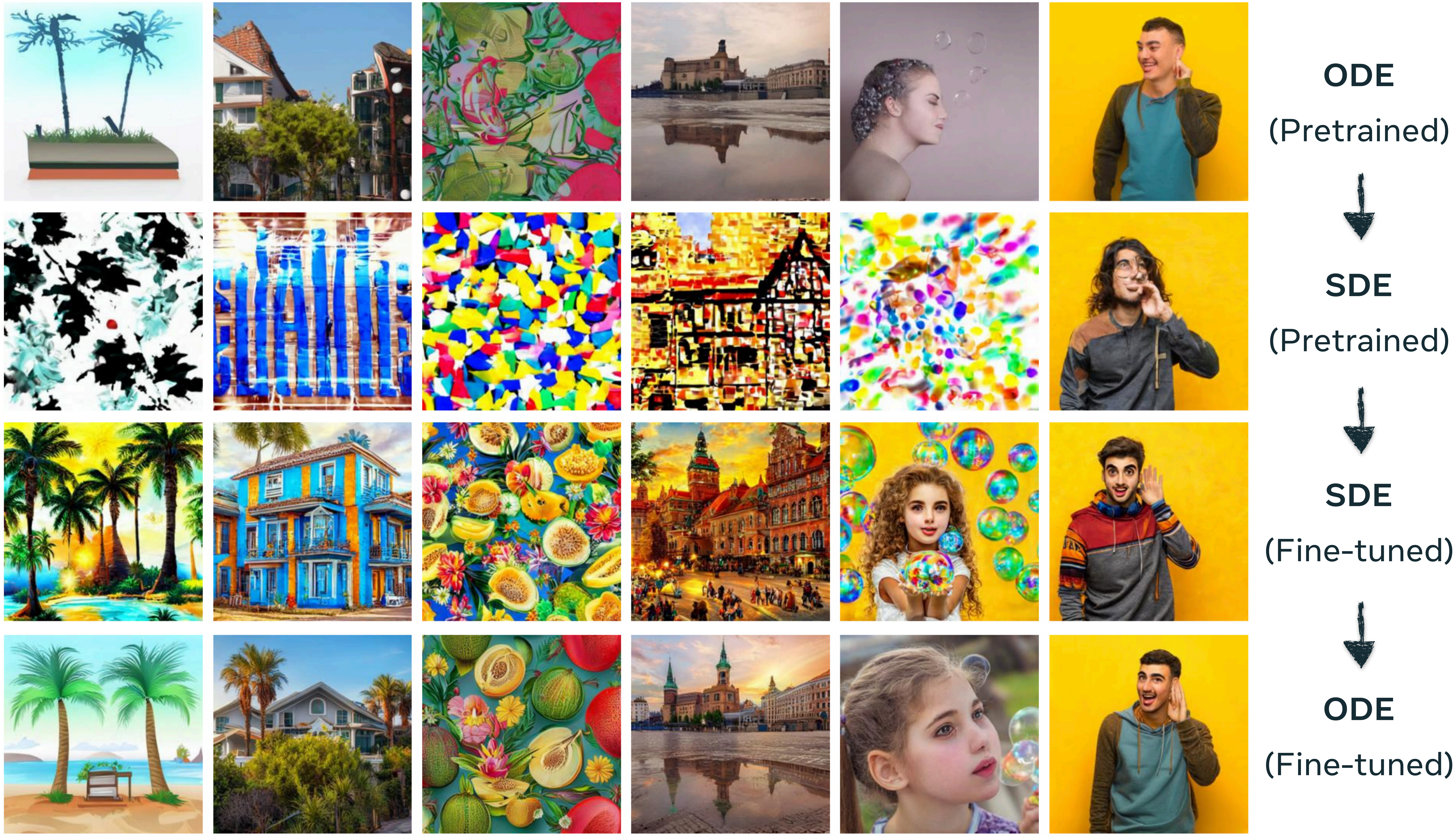
"Adjoint matching: Fine-tuning flow and diffusion generative models with memoryless stochastic optimal control" Domingo-Enrich et al. (2024)

Reward fine-tuning by stochastic optimal control

Memoryless SDE during fine-tuning.

Memoryless retains relation between **velocity & score.**

Uniquely allowing **conversion to ODE after fine-tuning.**



"Adjoint matching: Fine-tuning flow and diffusion generative models with memoryless stochastic optimal control" Domingo-Enrich et al. (2024)

Reward fine-tuning references

Gradient-based optimization:

"DPOK: Reinforcement Learning for Fine-tuning Text-to-Image Diffusion Models" Fan et al. (2023)

"Training diffusion models with reinforcement learning" Black et al. (2023)

"Imagereward: Learning and evaluating human preferences for text-to-image generation." Xu et al. (2023)

"Directly fine-tuning diffusion models on differentiable rewards." Clark et al. (2024)

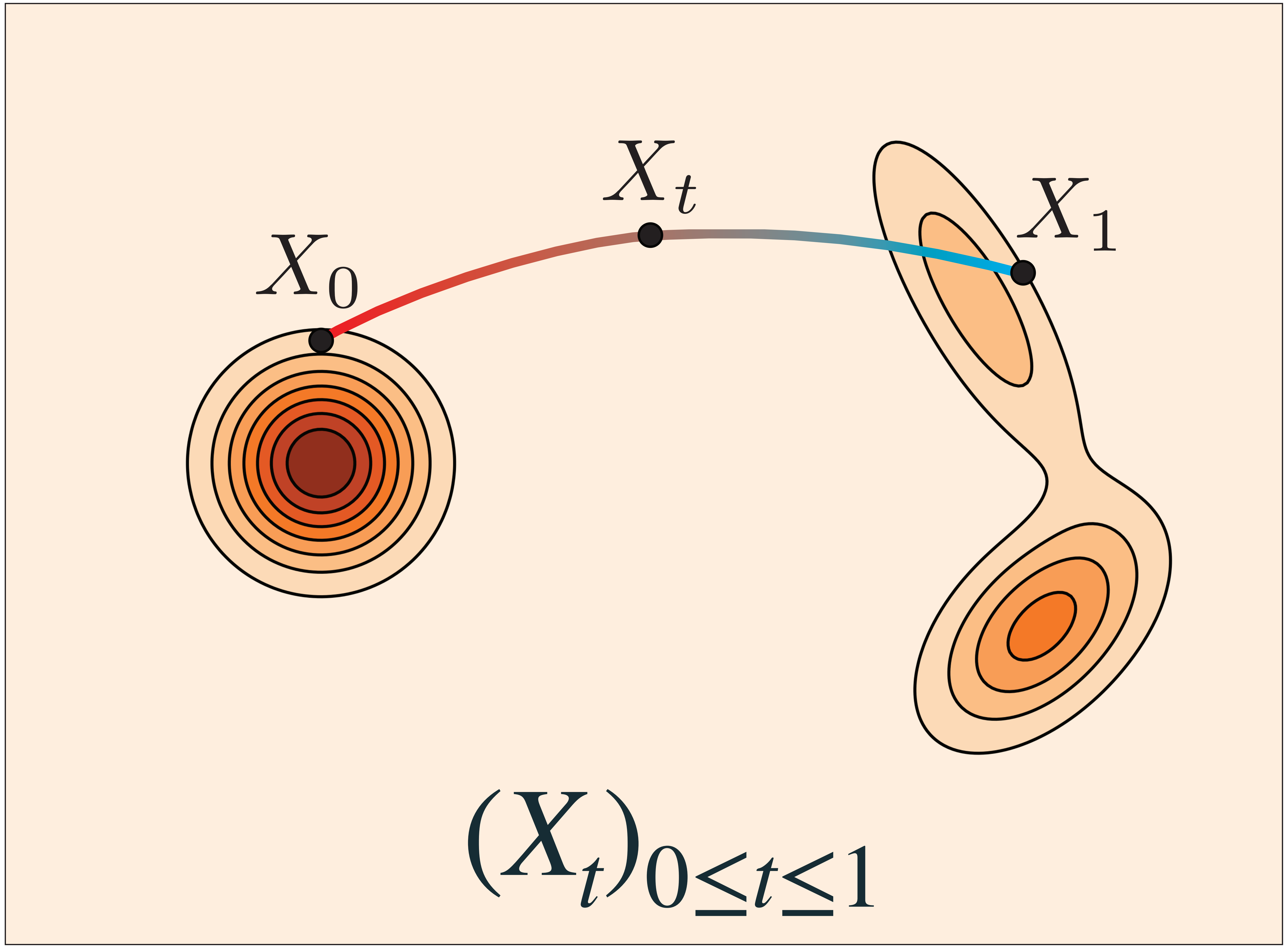
Stochastic optimal control:

"Fine-tuning of continuous-time diffusion models as entropy regularized control" Uehara et al. (2024)

"Adjoint matching: Fine-tuning flow and diffusion generative models with memoryless stochastic optimal control" Domingo-Enrich et al. (2024)

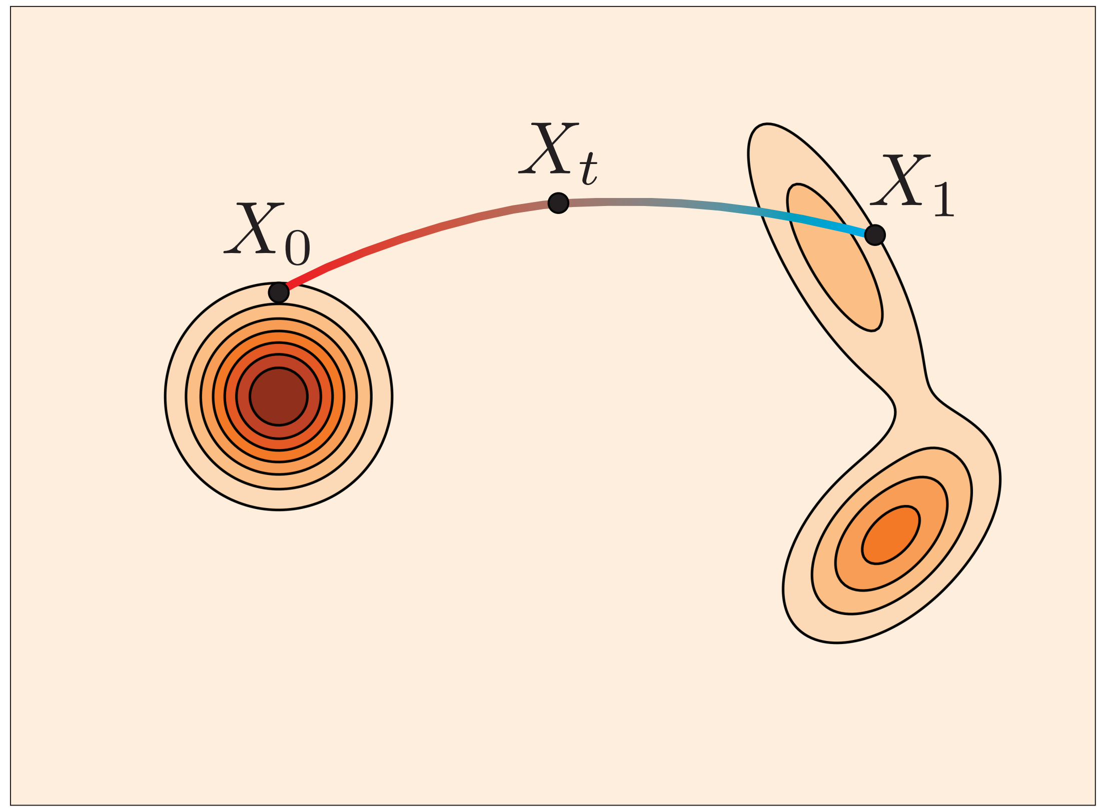
04 Generator Matching and Discrete Flows

Flow

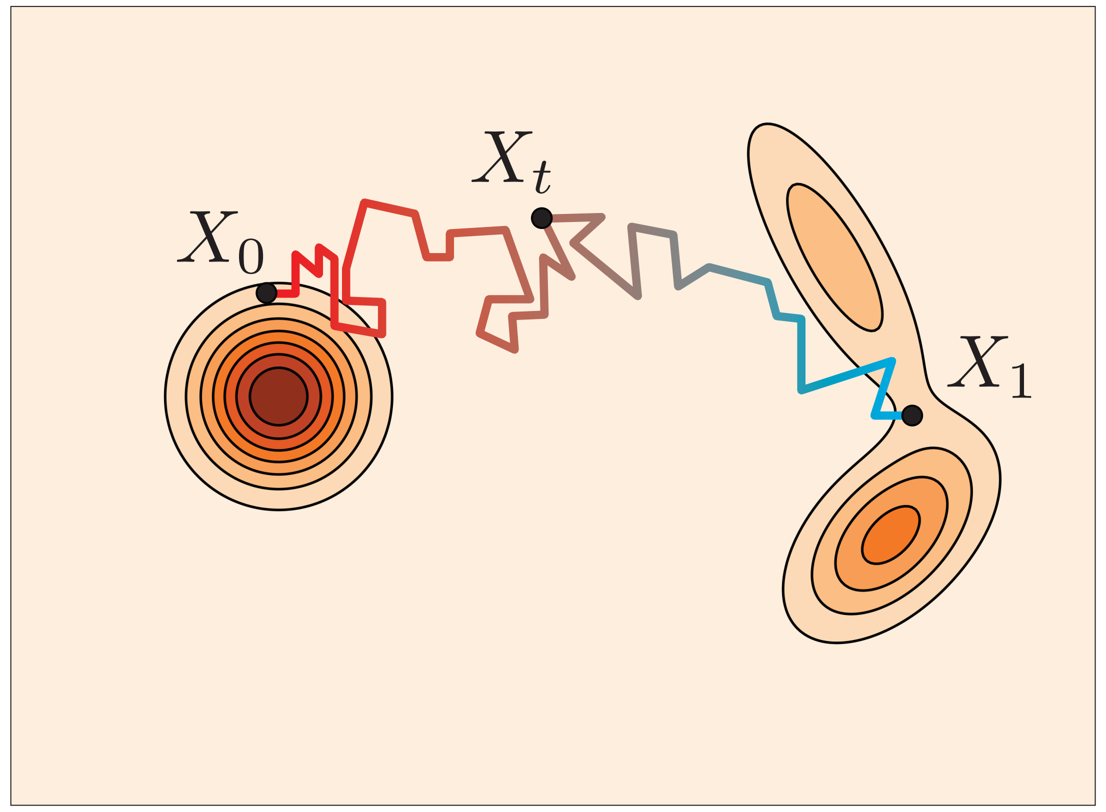


Continuous Time Markov Processes

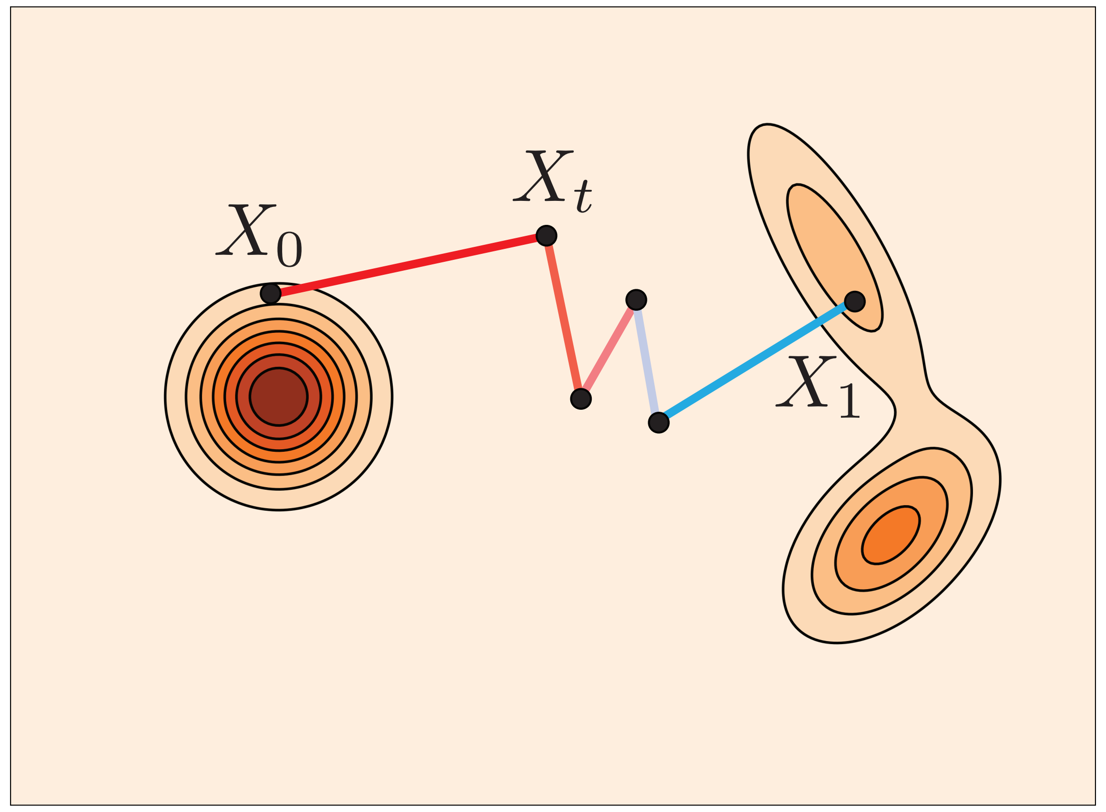
Flow



Diffusion

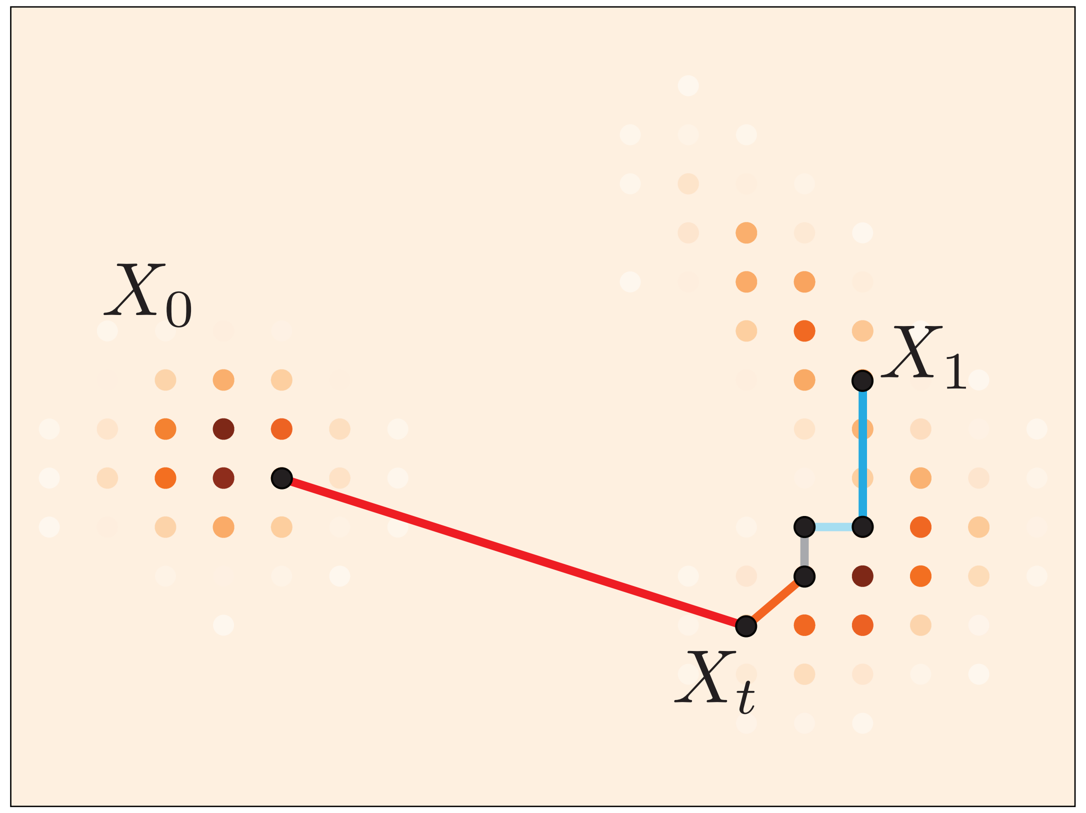


Jump



$$\mathcal{S} = \mathbb{R}^d$$

CTMC



$$\mathcal{S} = \mathscr{D}$$

Transition kernel

$$X_{t+h} \sim p_{t+h|t}(\,\cdot\,, X_t)$$

$$X_0 \sim p$$

Generator

Generalize the notion of **velocity** to arbitrary CTMP

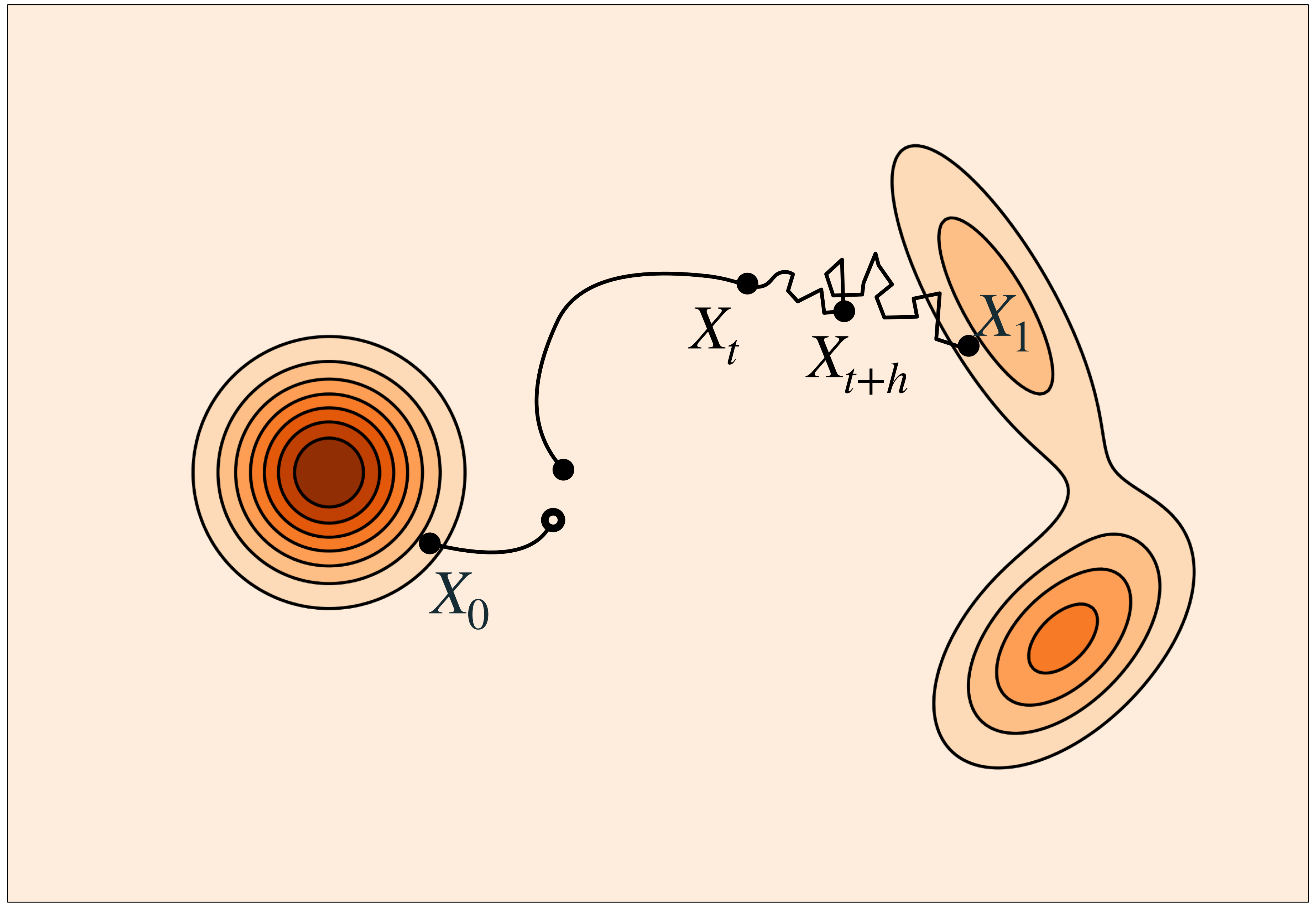
$$p_{t+h|t}(\,\cdot\,,x) = \underbrace{\delta_x}_{0^{\text{th}}\text{ order}} + \underbrace{h \overbrace{\boxed{\left.\frac{\mathrm{d}}{\mathrm{d}h}\right|_{h=0} p_{t+h|t}(\,\cdot\,,x)}}^{\mathscr{L}_t(\,\cdot\,,x)}}_{1^{\text{st}}\text{ order}} + \underbrace{o(h)}_{\text{error}}$$

"Generator Matching: Generative modeling with arbitrary Markov processes" Holderrieth et al. (2024)

CTMP via generator

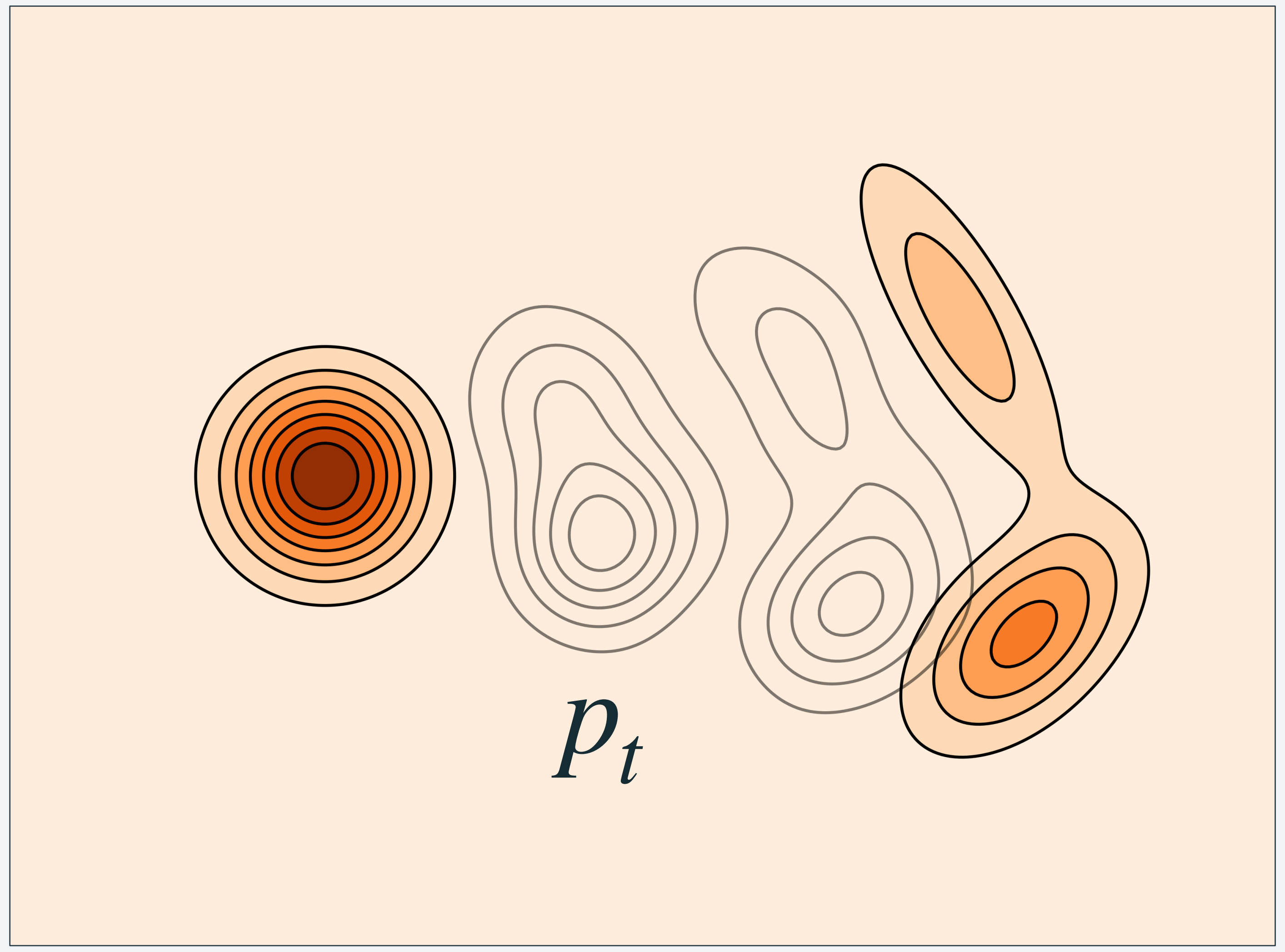
$$X_{t+h} \sim \delta_{X_t} + h\mathscr{L}_t(\,\cdot\,, X_t) + o(h)$$

$$X_0 \sim p$$

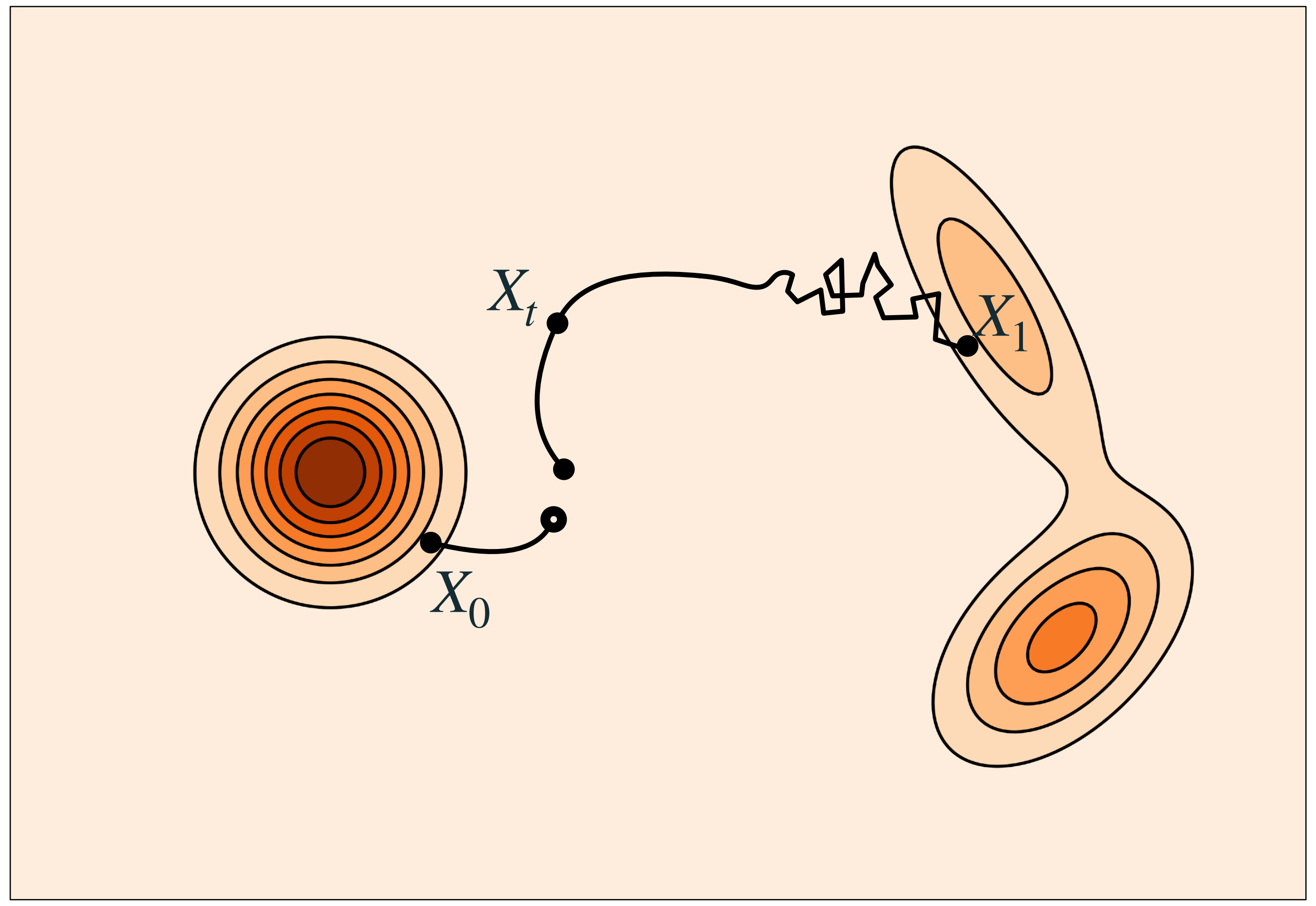


Marginal probability path

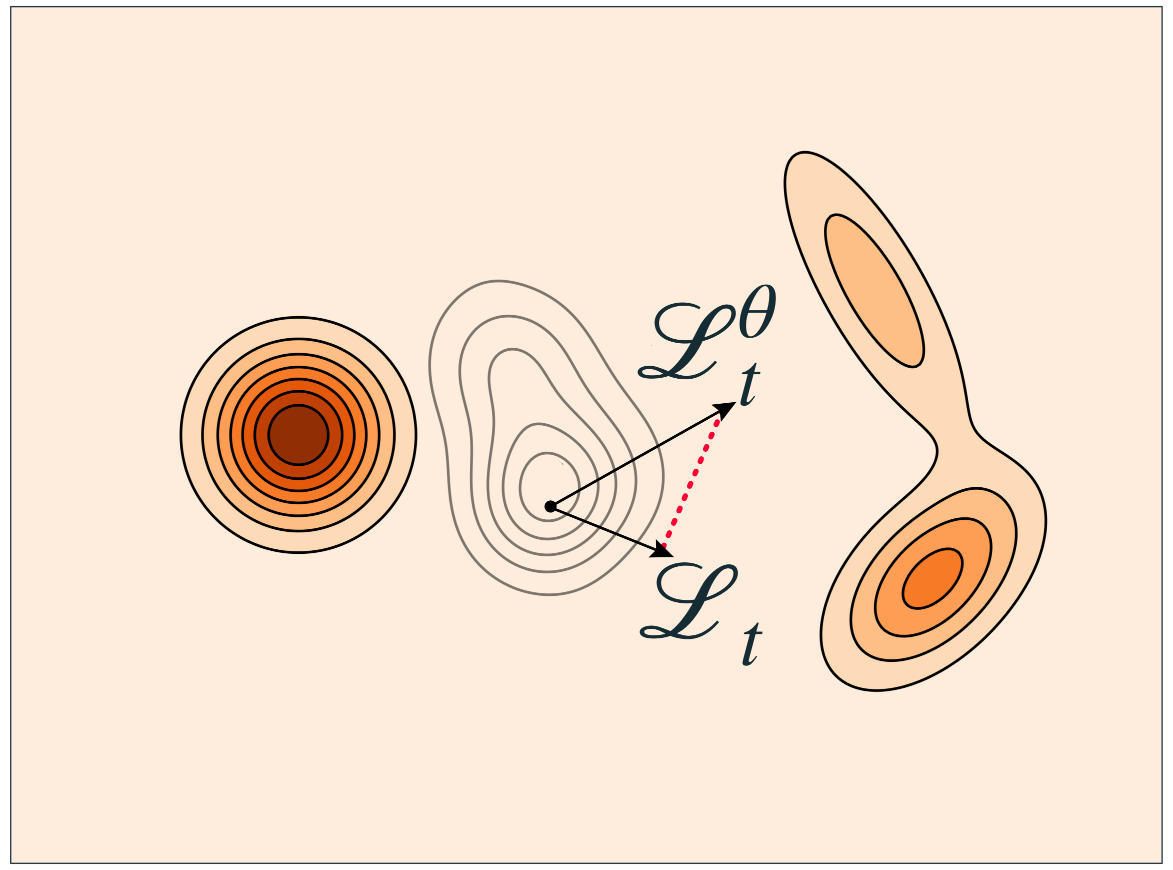
$$X_t \sim p_t$$



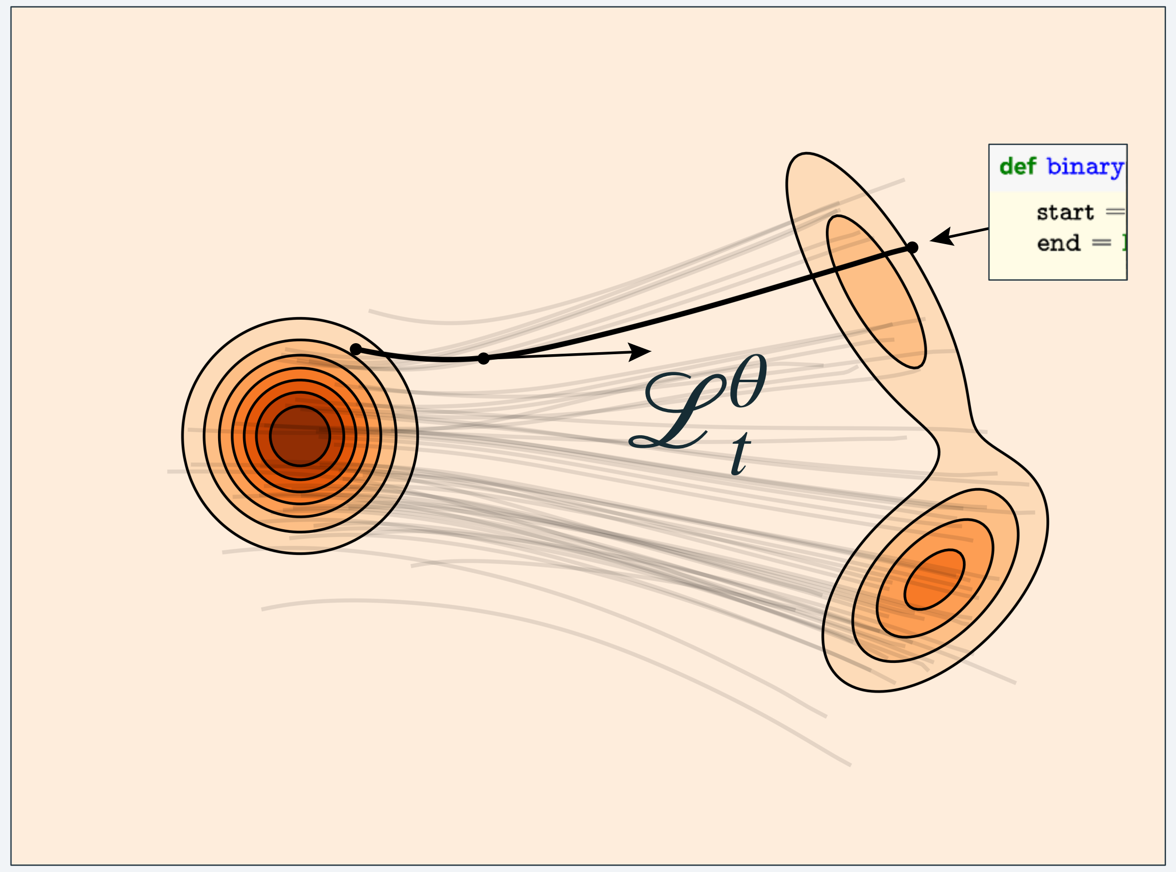
$$(X_t)_{0 \le t \le 1}$$



Generator Matching



Train a generator generating p_t with $p_0 = p$ and $p_1 = q$



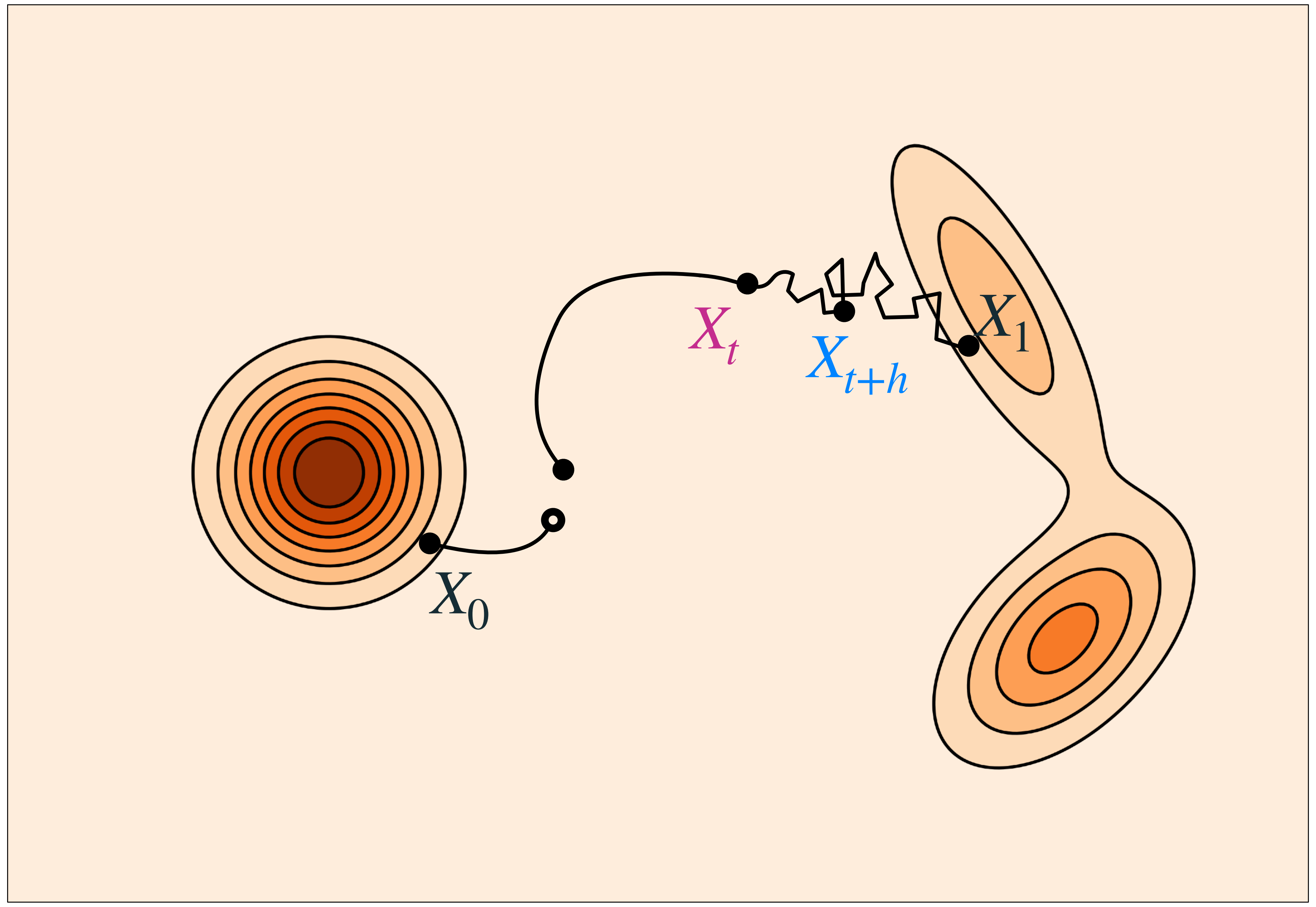
Sample from $X_0 \sim p$

Sampling

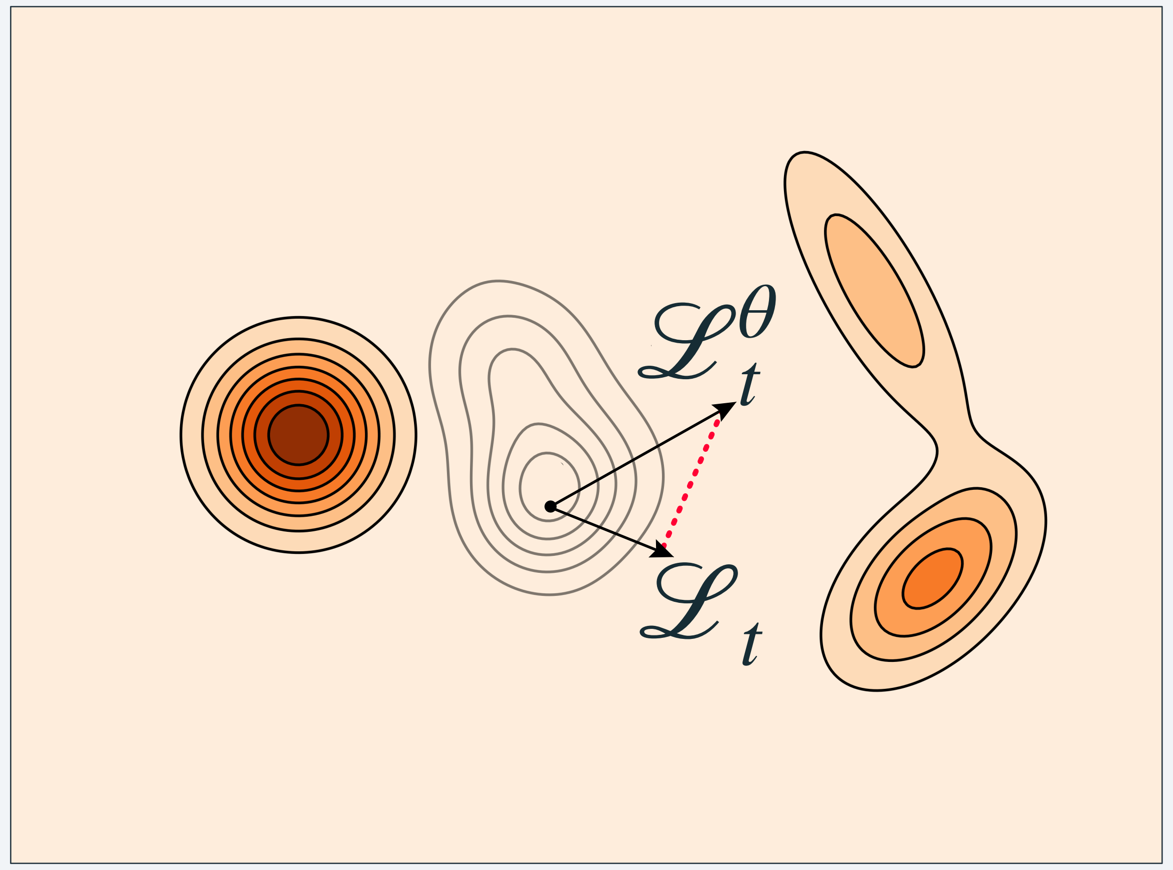
Euler method:

$$X_{t+h} \sim \delta_{X_t} + h\mathscr{L}_t^{\theta}(\,\cdot\,, X_t) + \cancel{o(h)}$$

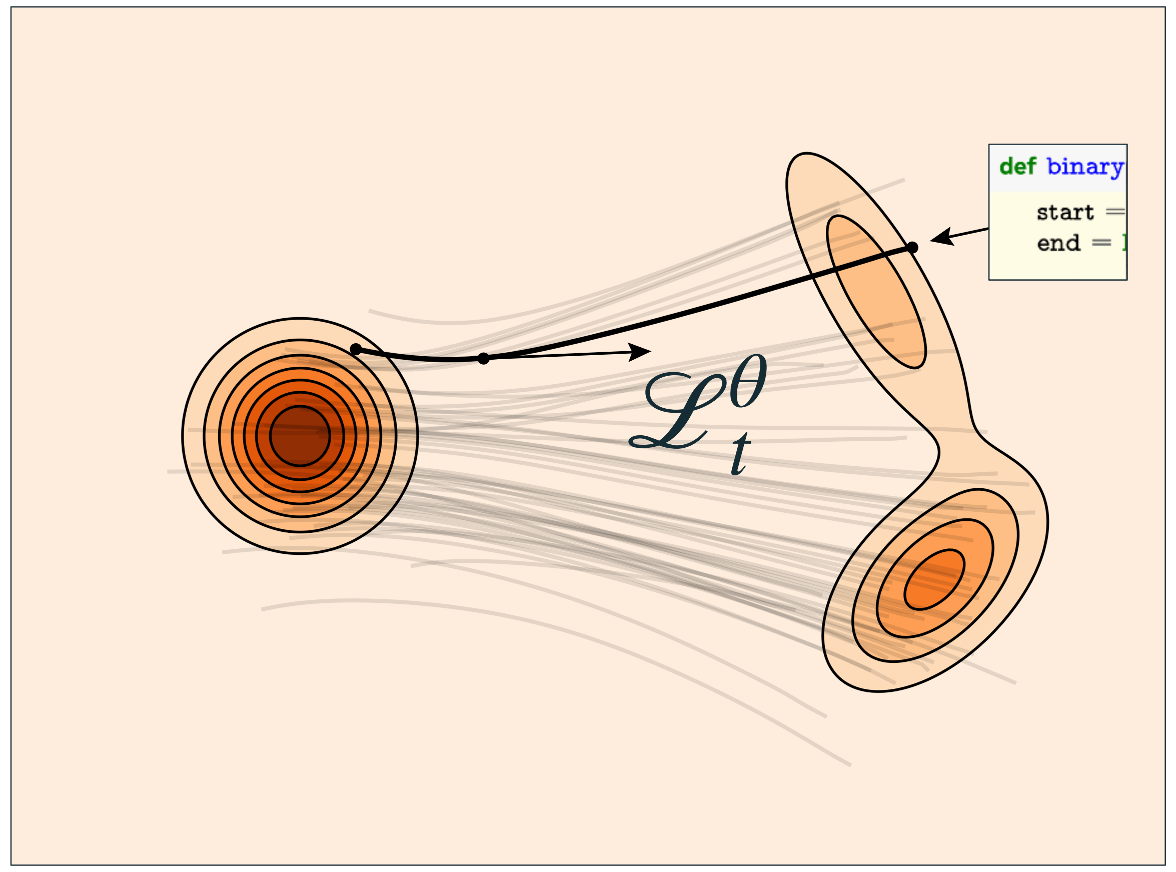
$$X_0 \sim p$$



Generator Matching



Train a generator generating p_t with $p_0 = p$ and $p_1 = q$

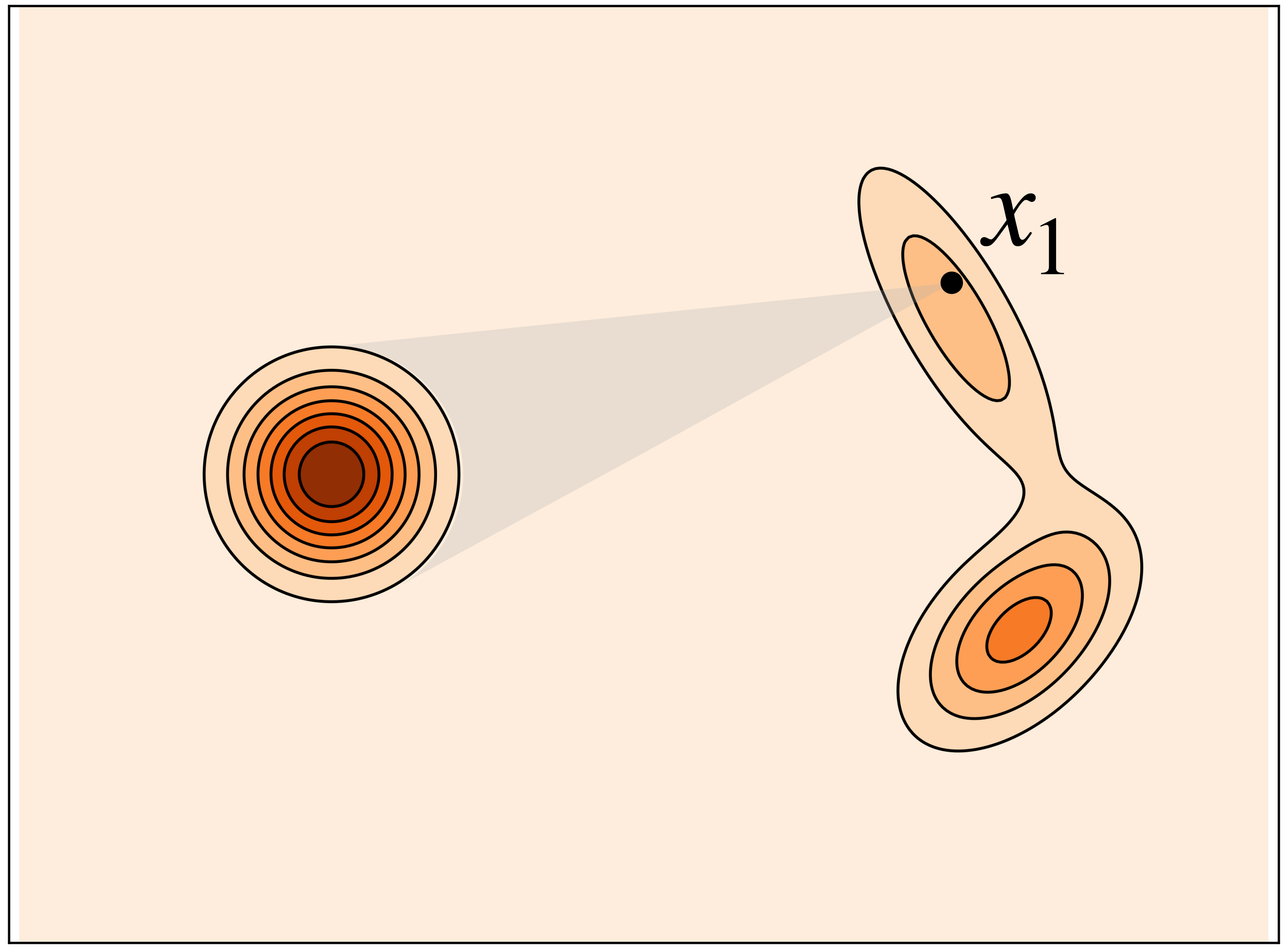


Sample

from $X_0 \sim p$

Building generator from conditional generators

Repeating the Kata from the continuous case.....



Kolmogorov Equation

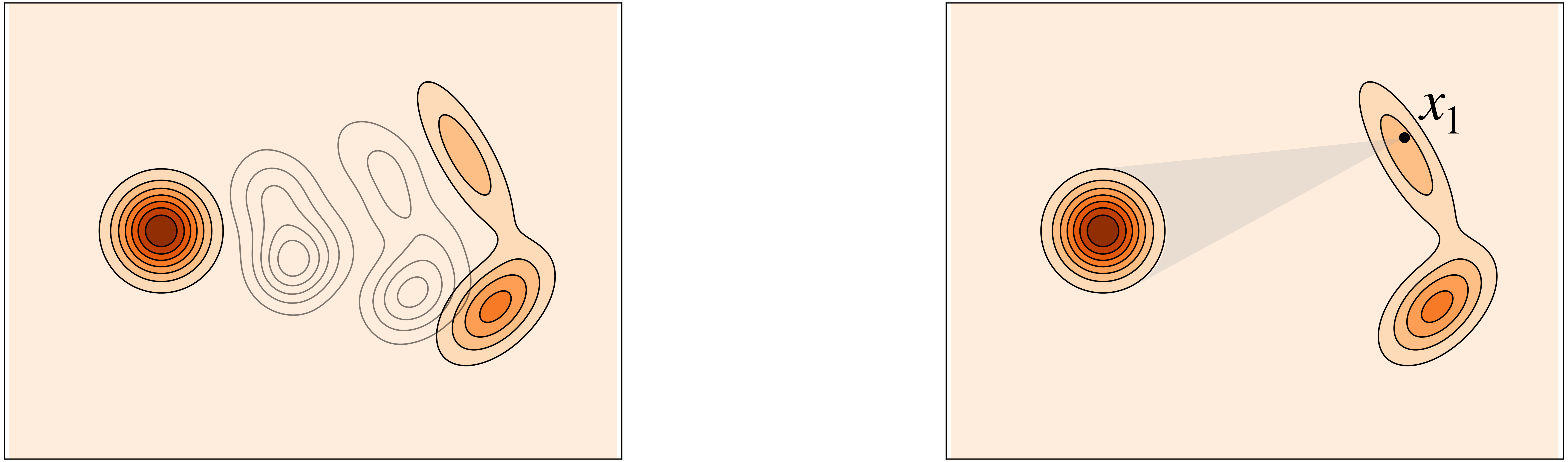
$p_{t|1}(x|x_1)$ conditional probability

$\mathscr{L}_t(\cdot, x|x_1)$ conditional generator

"Generator Matching: Generative modeling with arbitrary Markov processes" Holderrieth et al. (2024)

Building generator from conditional generators

Repeating the Kata from flows.....



$$p_t(x) = \mathbb{E}_{X_1} p_{t|1}(x \mid X_1) \longleftarrow p_{t|1}(x \mid x_1) \quad \text{conditional probability}$$

average

$$\mathcal{L}_t(\cdot \mid x) = \mathbb{E}\left[\mathcal{L}_t(\cdot\,, X_t \mid X_1) \,\middle|\, X_t = x\right] \longleftarrow \mathcal{L}_t(\cdot\,, x \mid x_1) \quad \text{conditional generator}$$

"Generator Matching: Generative modeling with arbitrary Markov processes" Holderrieth et al. (2024)

The Marginalization Trick

Theorem*: The marginal generator generates the marginal probability path.

$$\mathcal{L}_t(\cdot\,,x) = \mathbb{E}\left[\mathcal{L}_t(\cdot\,,X_t|X_1) \mid X_t = x\right] \qquad p_t(x) = \mathbb{E}_{X_1} p_{t|1}(x|X_1)$$

Train with Bregman divergence:

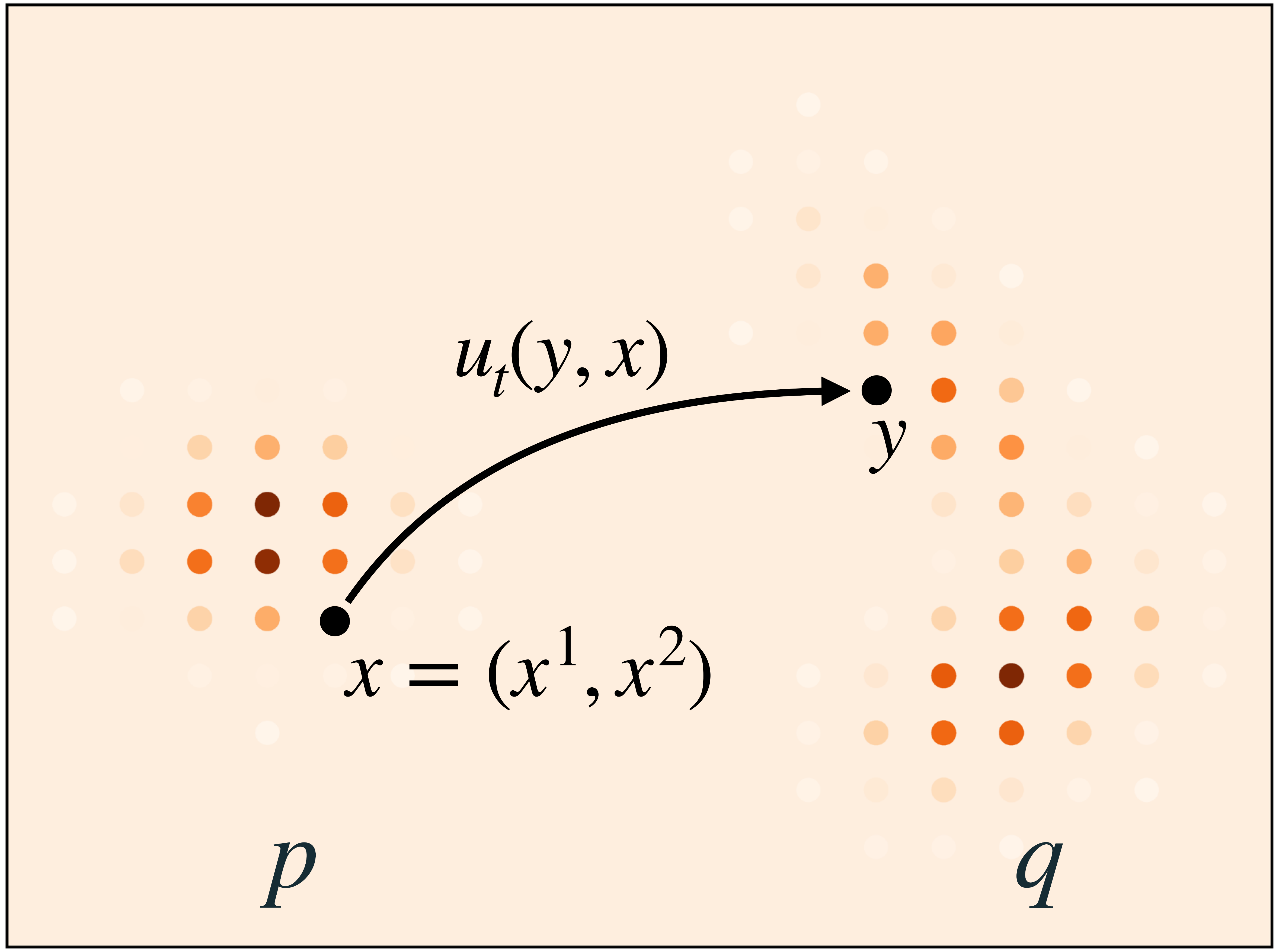
$$\mathcal{L}_{\text{CGM}}(\theta) = \mathbb{E}_{t,X_1,X_t} D_{X_t}\left(\mathcal{L}_t(\cdot\,,X_t|X_1), \mathcal{L}_t^{\theta}(\cdot\,,X_t)\right)$$

"Generator Matching: Generative modeling with arbitrary Markov processes" Holderrieth et al. (2024)

Discrete Flow Matching

- State space $\mathcal{T}^d$: **sequences of tokens**
- $x = (x^1, x^2, \ldots, x^d) \in \mathcal{S}$

$$p_{t+h|t}(y, x) = \delta(y, x) + h \boxed{u_t(y, x)} + o(h)$$



"Generative Flows on Discrete State-Spaces: Enabling Multimodal Flows with Applications to Protein Co-Design" Campbell et al. (2024)
"Discrete Flow Matching" Gat el al. (2024)

Factorized velocities

Similar to continuous case $\mathcal{S} = \mathbb{R}^d$:

$$u_t(x) = [u_t^1(x), \ldots, u_t^d(x)]$$

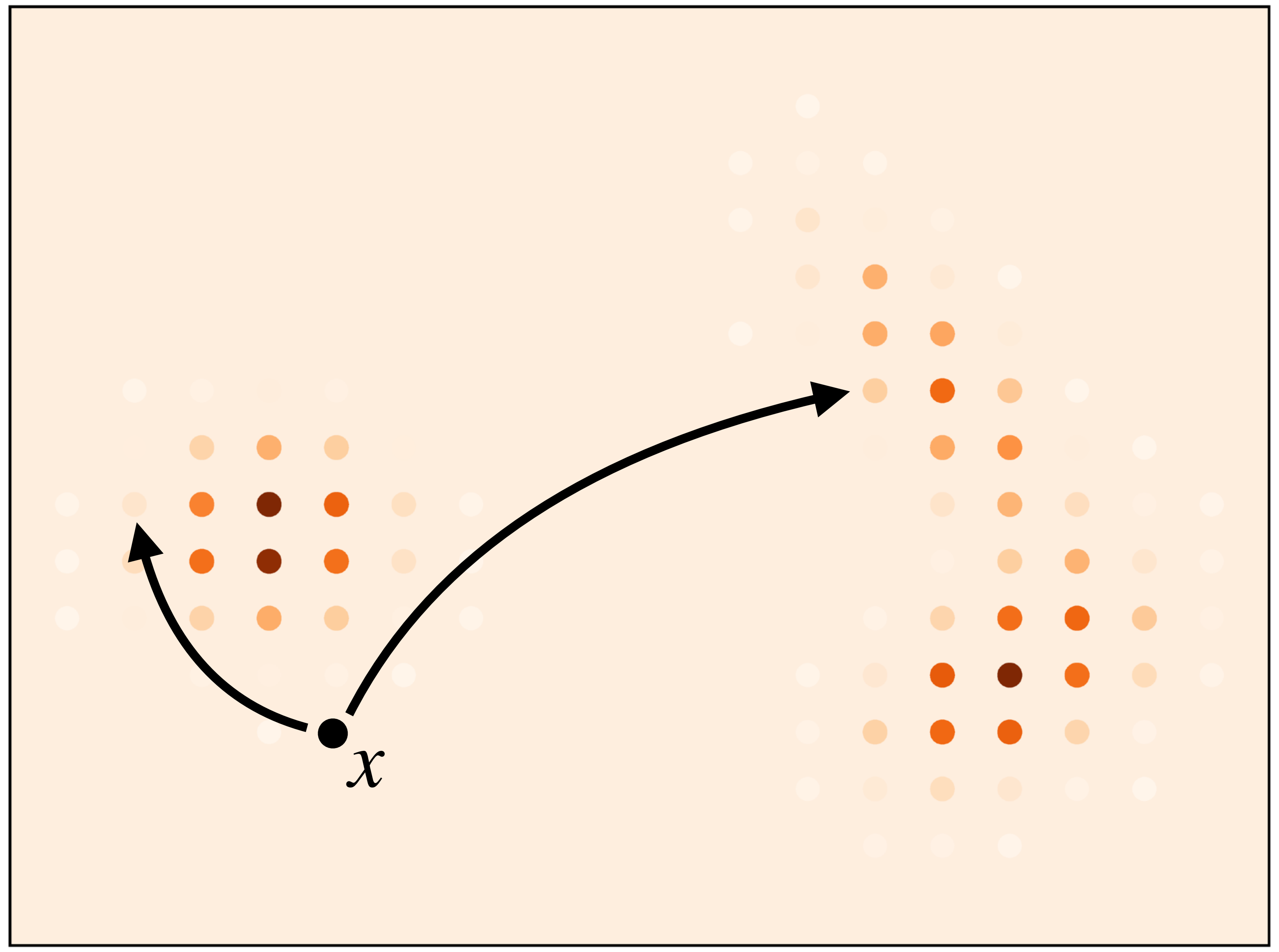
"Real life" case:

$d \approx 1000, |\mathcal{T}| \approx 50000$

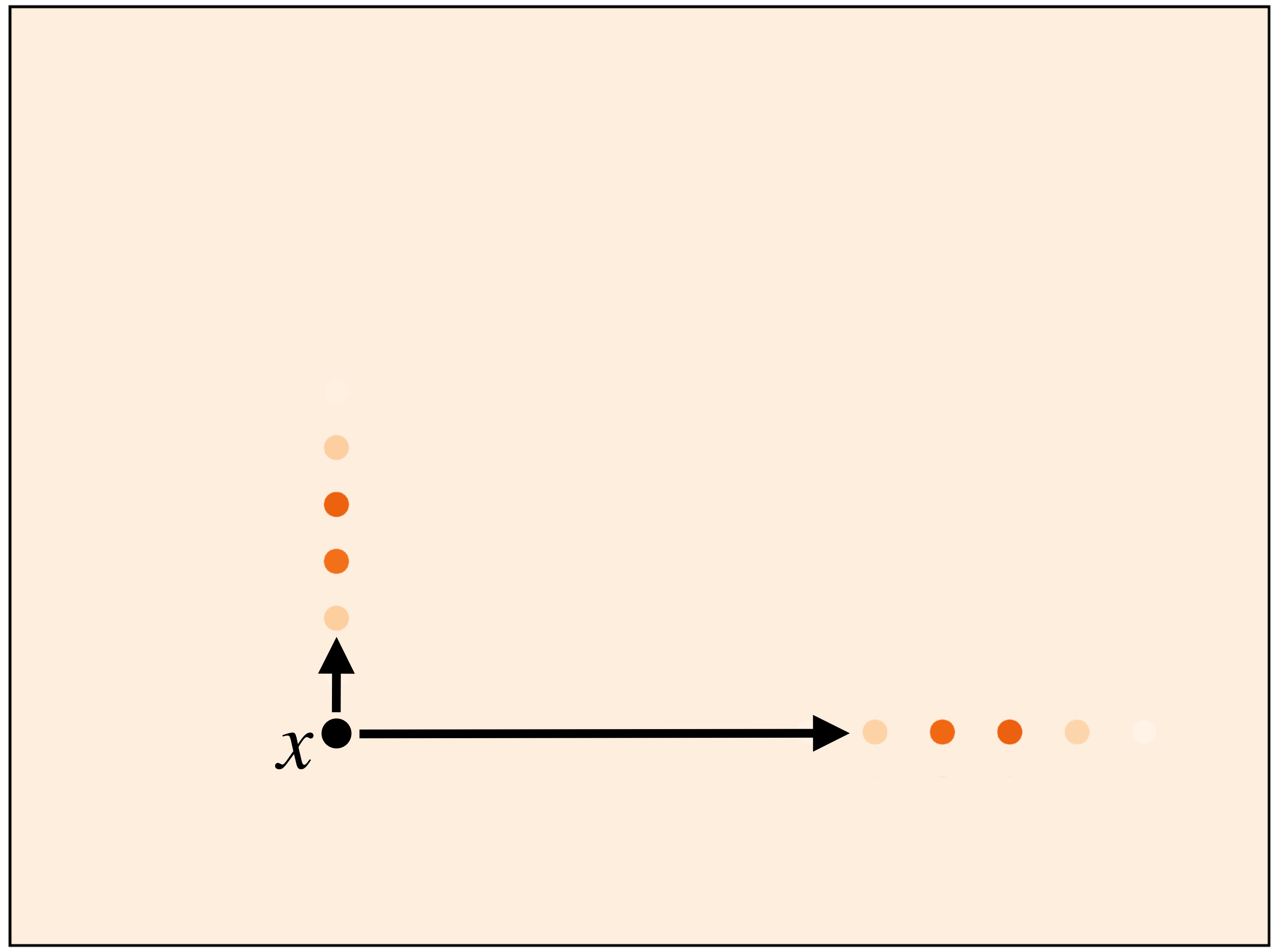
```python
def binary_search(arr, x):
    start = 0
    end = len(arr)-1

    # While performing binary search
    while start <= end:
        mid = (start + end) // 2
        # If x is greater
        if arr[mid] < x:
            start = mid + 1
        # If x is smaller
        elif arr[mid] > x:
            end = mid - 1
        else:
            return mid
    return -1
```

$u_t(\cdot, x) \in \mathbb{R}^{\mathcal{T}^d}$



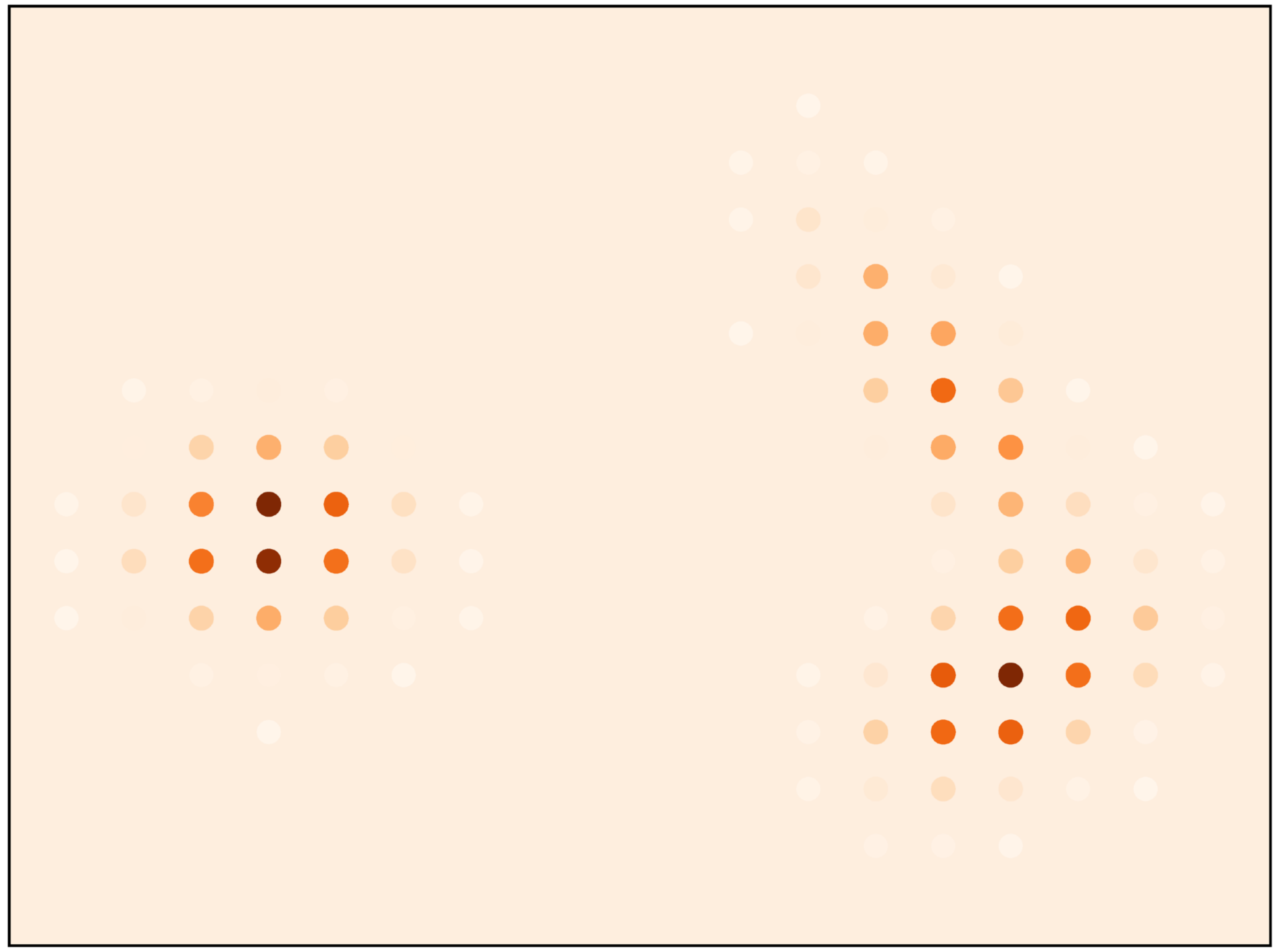
$u_t(\cdot, x) \in \mathbb{R}^{d|\mathcal{T}|}$

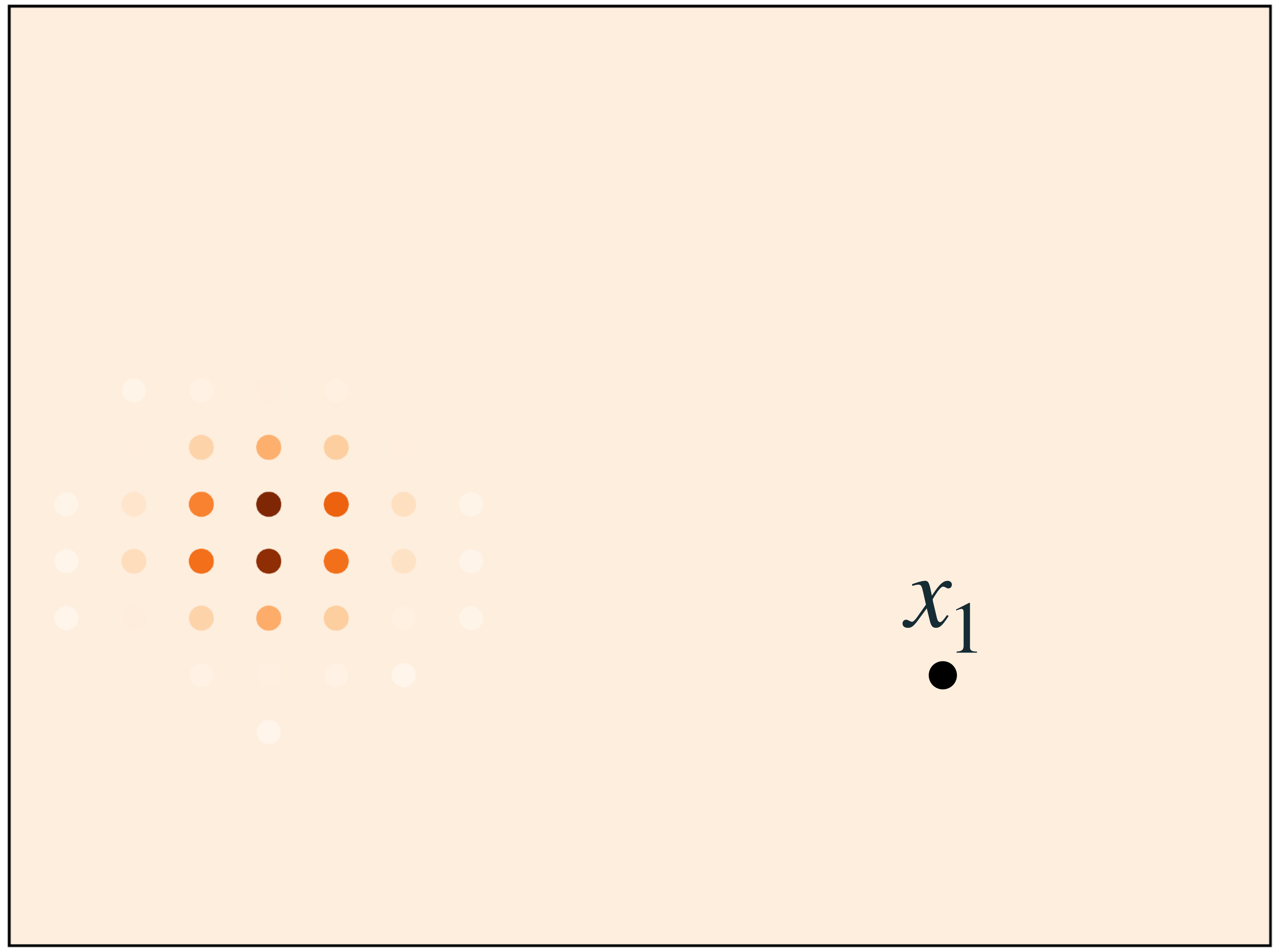


$$u_t^i(y^i, x)$$

"A Continuous Time Framework for Discrete Denoising Models" Campbell et al. (2022)

Build (factorized) velocities





Mixture path

$$p_t(x) = \dots \quad \longleftarrow \quad p^i_{t|1}(x^i \mid x_1) = (1-t)p(x^i) + t\delta(x^i, x^i_1)$$

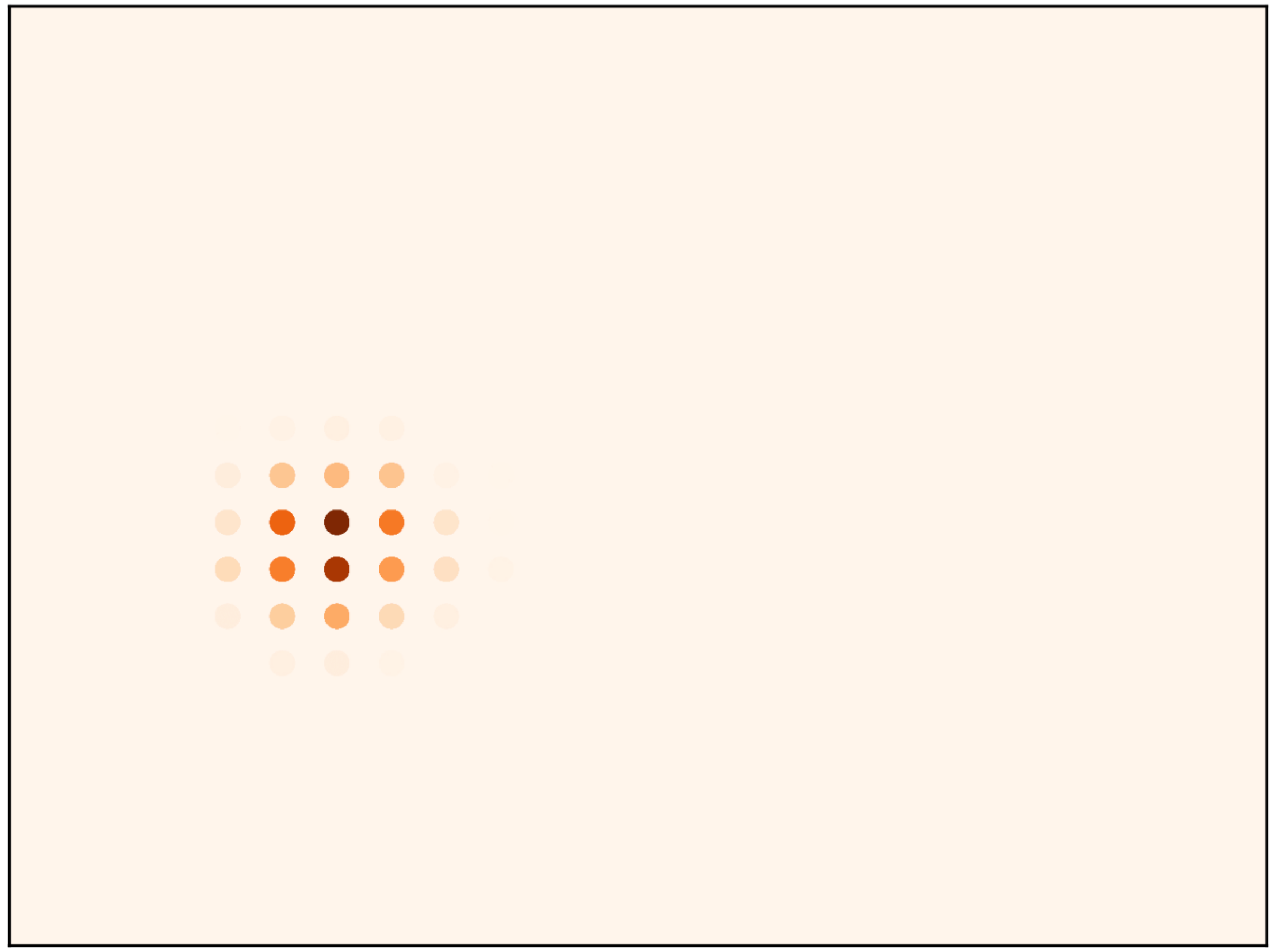
average

$$u^i_t(y^i, x) = \dots \quad \longleftarrow \quad u^i_t(y^i, x^i \mid x_1) = \frac{1}{1-t}\delta(y^i, x^i_1) \qquad y^i \neq x^i$$

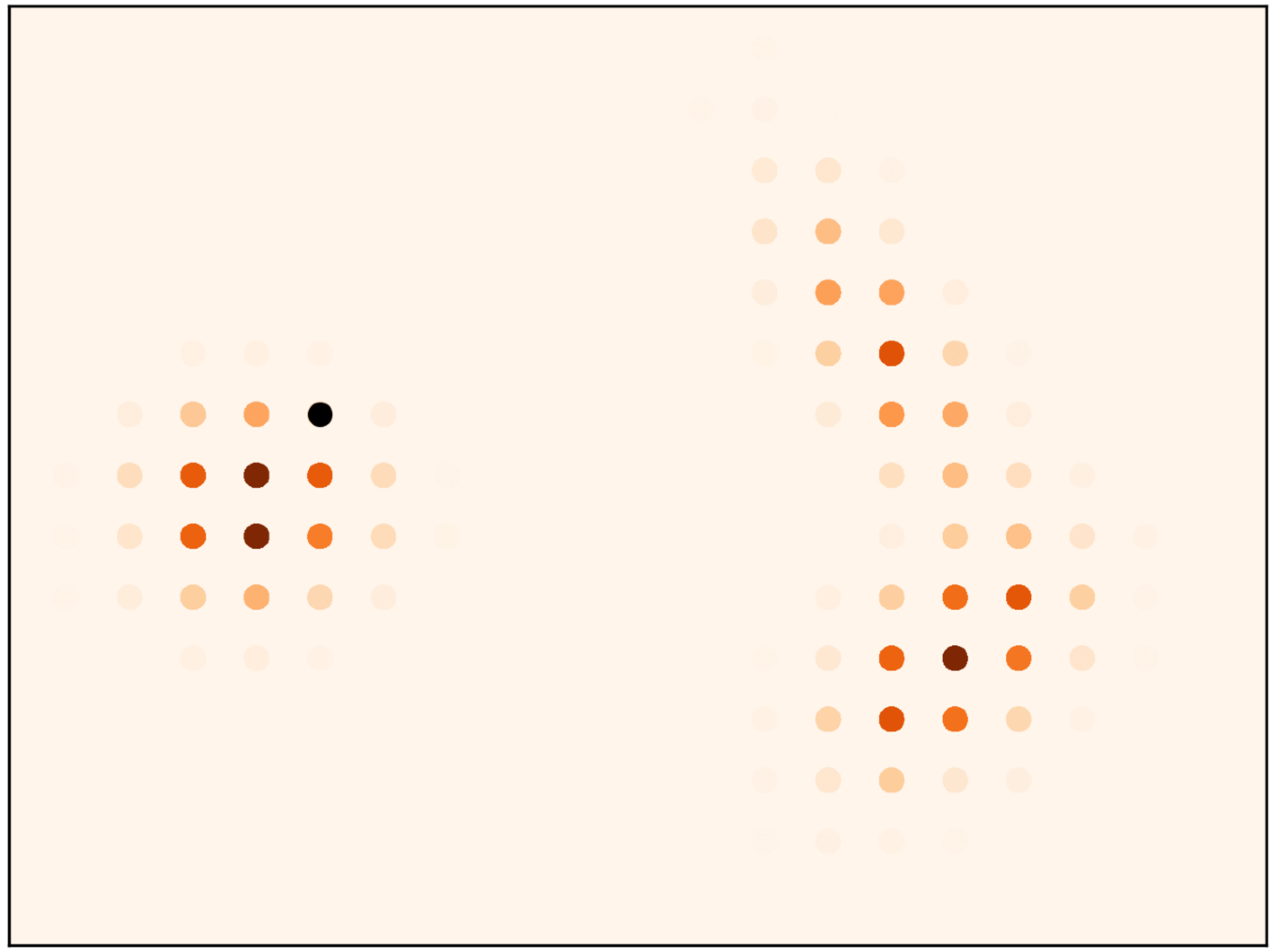
"Generative Flows on Discrete State-Spaces: Enabling Multimodal Flows with Applications to Protein Co-Design" Campbell et al. (2024)
"Discrete Flow Matching" Gat el al. (2024)

Discrete Flow Matching Loss

$$\mathcal{L}_{\text{CDFM}}(\theta) = \mathbb{E}_{t,X_1,X_t} \sum_i D_{X_t}\left(\frac{1}{1-t}\delta(\,\cdot\,,X_1^i), u_t^{\theta,i}(\,\cdot\,,X_t)\right)$$



$p_t(x)$



$(X_t)_{0\leq t\leq 1}$

“Discrete Flow Matching” Gat el al. (2024)

"Flow Matching with General Discrete Paths: A Kinetic-Optimal Perspective" Shaul et al. (2024)

“Discrete Diffusion Modeling by Estimating the Ratios of the Data Distribution” Lou et al. (2024)

Example: code generation model (1.7B)

Method	Data	HumanEval↑ Pass@1	HumanEval↑ Pass@10	HumanEval↑ Pass@25	MBPP (1-shot)↑ Pass@1	MBPP (1-shot)↑ Pass@10	MBPP (1-shot)↑ Pass@25
Autoregressive	Text	1.2	3.1	4.8	0.2	1.7	3.3
	Code	14.3	21.3	27.8	17.0	34.3	44.1
FM	Text	1.2	2.6	4.0	0.4	1.1	3.6
	Code	6.7	13.4	18.0	6.7	20.6	26.5
FM (Oracle length)	Code	11.6	18.3	20.6	13.1	28.4	34.2

Success case

```python
from typing import List


def below_zero(operations: List[int]) -> bool:
    """ You're given a list of deposit and withdrawal operations
     on a bank account that starts with
     zero balance. Your task is to detect if at any point the balance
     of account fallls below zero, and
     at that point function should return True.
     Otherwise it should return False.
     >>> below_zero([1, 2, 3])
     False
     >>> below_zero([1, 2, -4, 5])
     True
     """

    balance = 0
    for operation in operations:
        if balance + operation < 0:
            return True
        balance = balance + operation
    return False
```

Failure case

```python
def solve(s):
    """You are given a string s.
     if s[i] is a letter, reverse its case from lower to upper or vise versa,
     otherwise keep it as it is.
     If the string contains no letters, reverse the string.
     The function should return the resulted string.
     Examples
     solve("1234") = "4321"
     solve("ab") = "AB"
     solve("#a@C") = "#A@c"
     """

    if s is None:
        return ""
    if len(s) == 0:
        return s
    # result string
    res = str()
    # to lower case
    s = s.lower()
    for i in range(len(s)):
        ch = s[i]
        if ch.islower():
            res.append(ch)
        elif ch.isupper():
            res.append(ch.lower())
        else:
            res.append(ch)
    return res
```

“Discrete Flow Matching” Gat el al. (2024)

Example: code generation model (1.7B)

```python
def binary_search(arr, x):



        # If x is greater


        # If x is smaller


        else:

```

“Discrete Flow Matching” Gat el al. (2024)

OPEN PROBLEMS FOR DISCRETE FLOWS

How to go beyond the factorized velocity?

Better sampling?

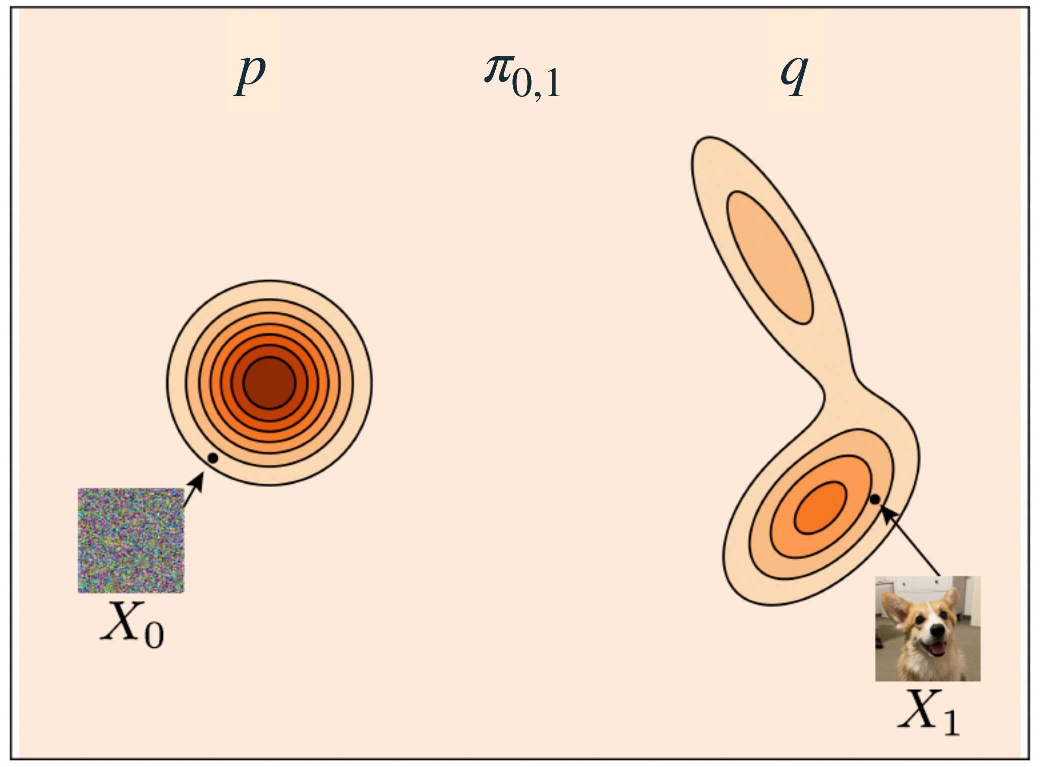
How to explore the (huge) design space?

Design choices:

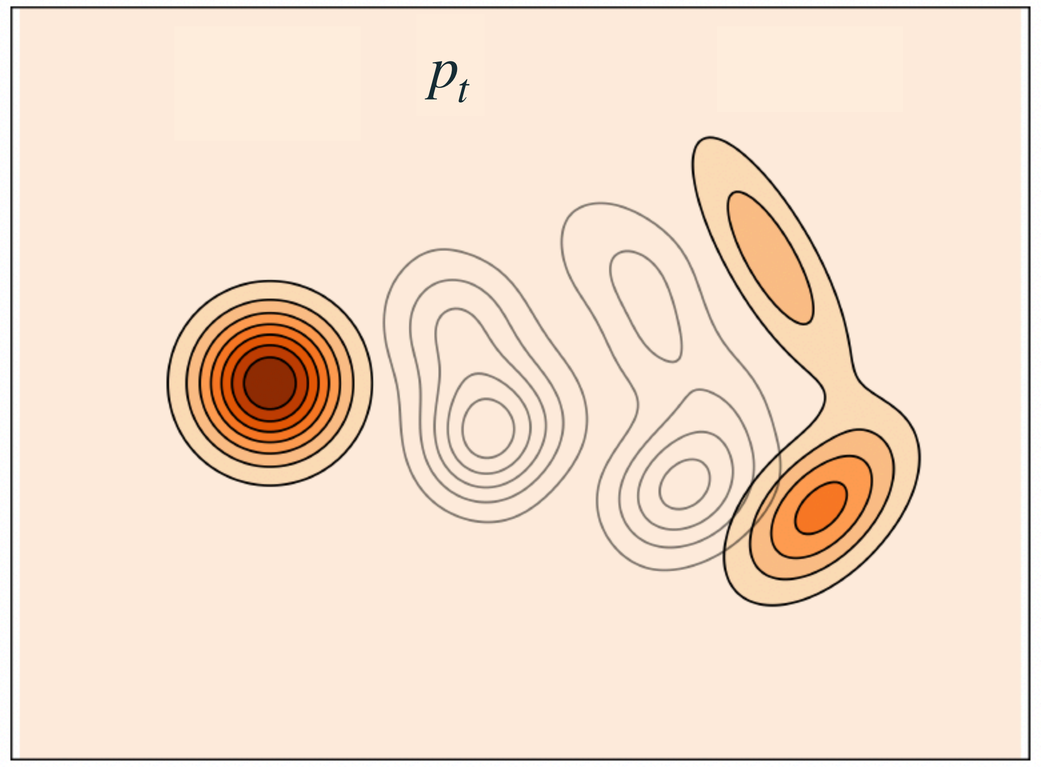
- Process
- Marginal Path
- Corrector steps
- Models superposition

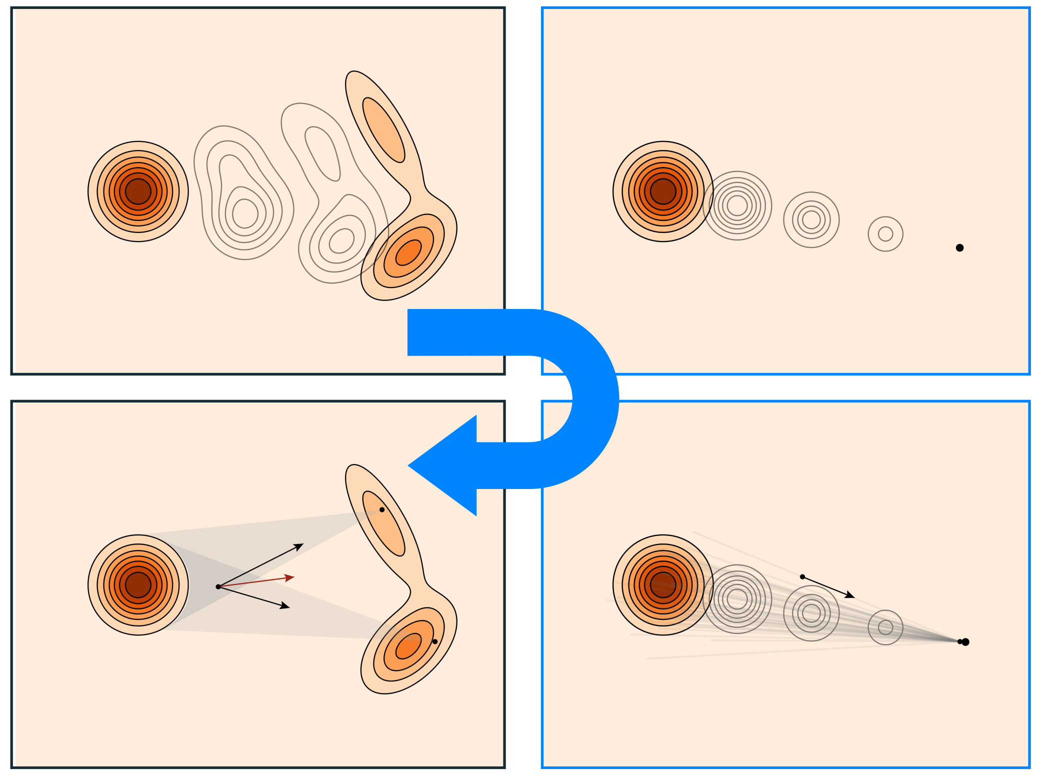
Flow Matching blueprint

Data

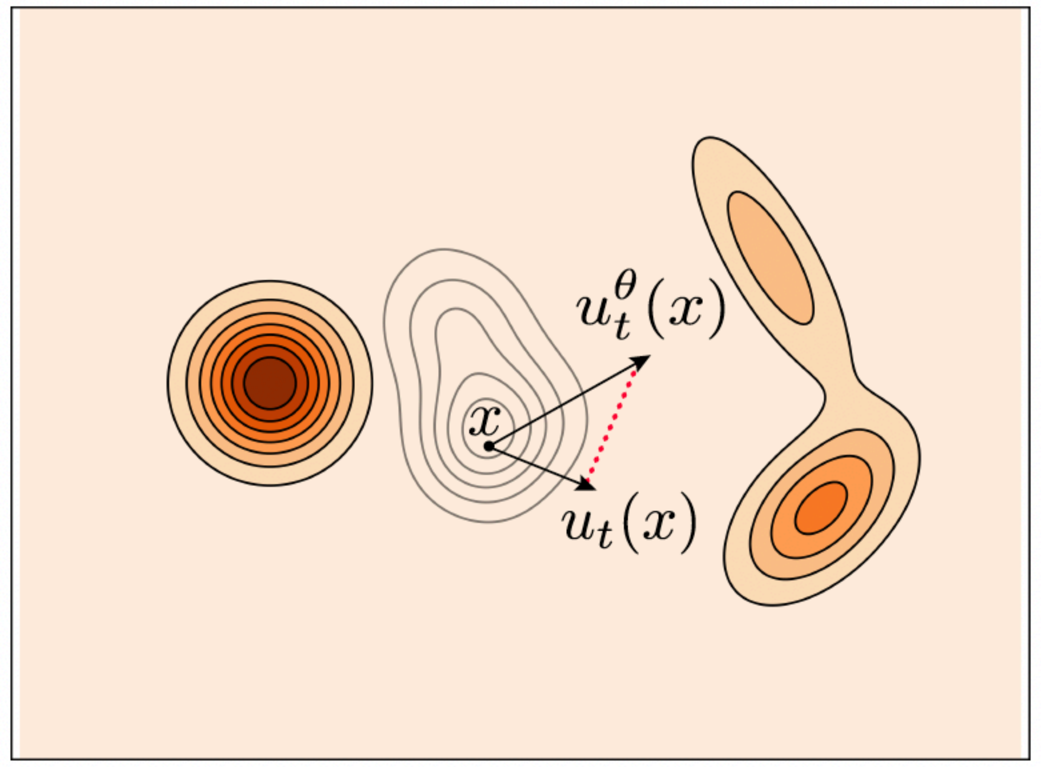


Path design

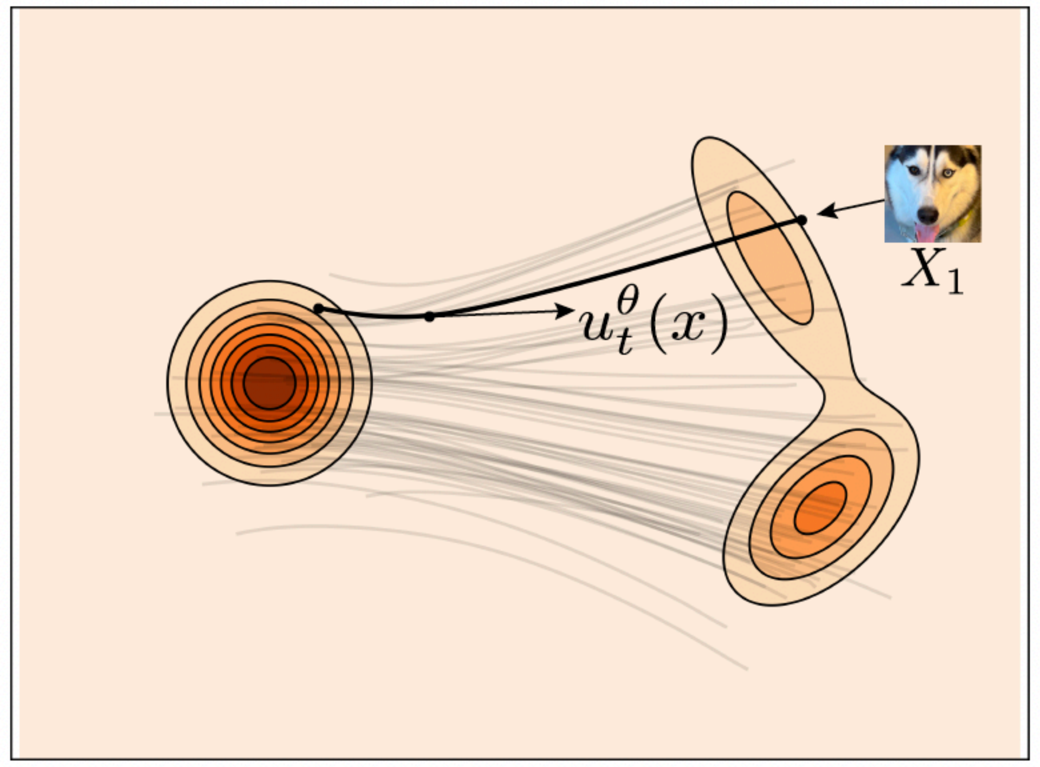




Training



Sampling



06 Demo

Flow Matching Codebase

Supports Discrete, Riemannian and Continuous FM!

Scalable training code

Discrete: FineWeb

Continuous: ImageNet 32, 64

Github