Unlocking Dis^con^tinuities in Flow Models:

Jumps, Control Flow, Insertions, Deletions, etc

Ricky T. Q. Chen

Meta

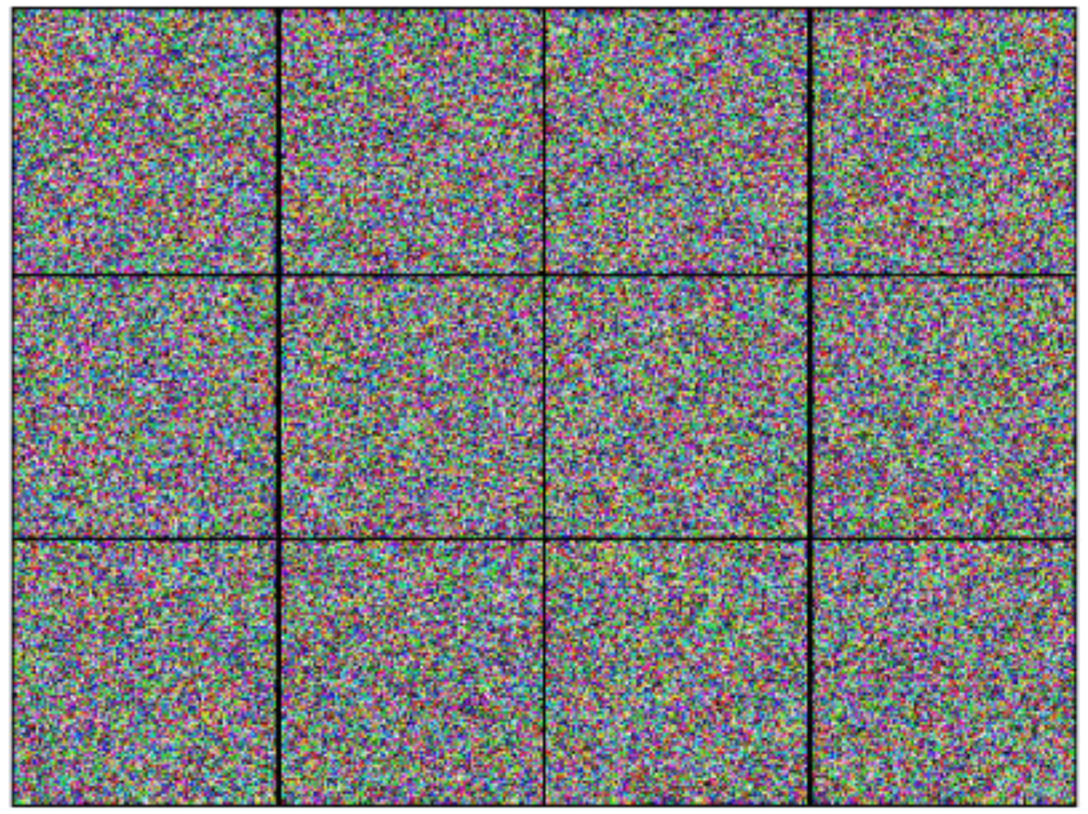
🤔

@ CVPR25 Workshop on ~~Visual~~ Generative Modeling: **What’s After Diffusion?**

About me and this talk

Data-driven

- fit to data
- simple & scalable



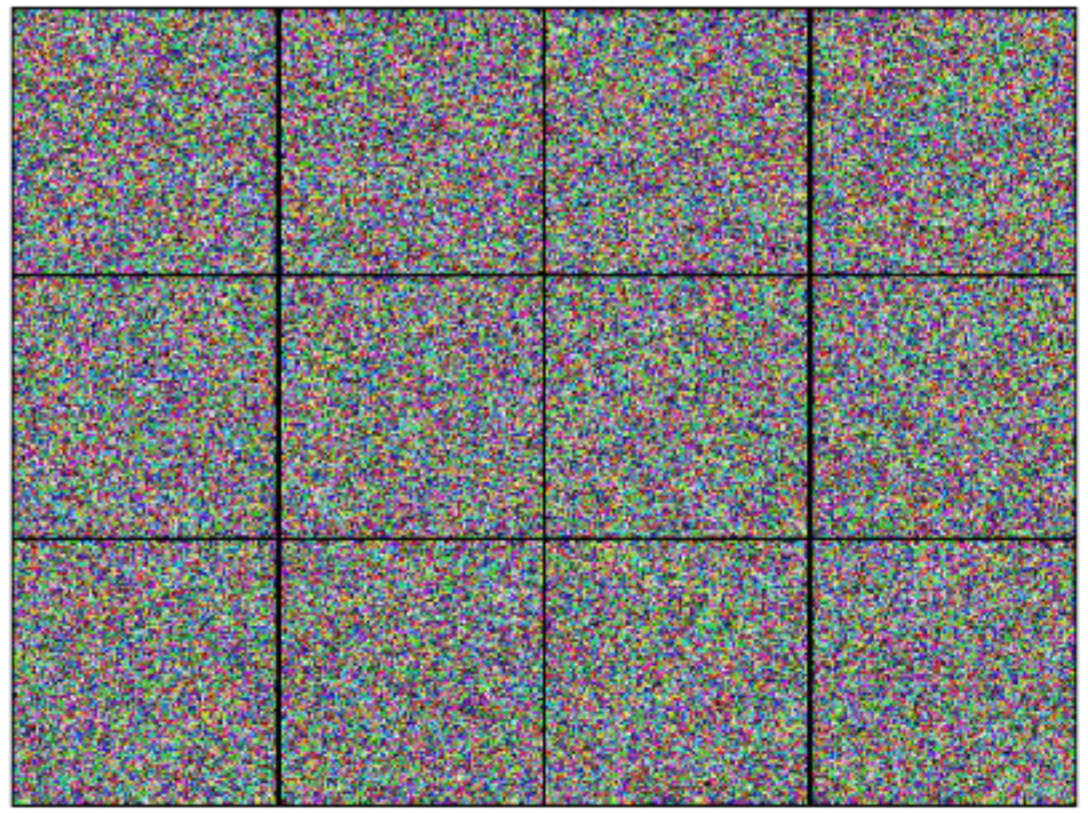
Flow Matching

Meta MovieGen

About me and this talk

Data-driven

- fit to data
- simple & scalable



Flow Matching

Meta MovieGen

Reward-driven

- maximize reward
- no data available

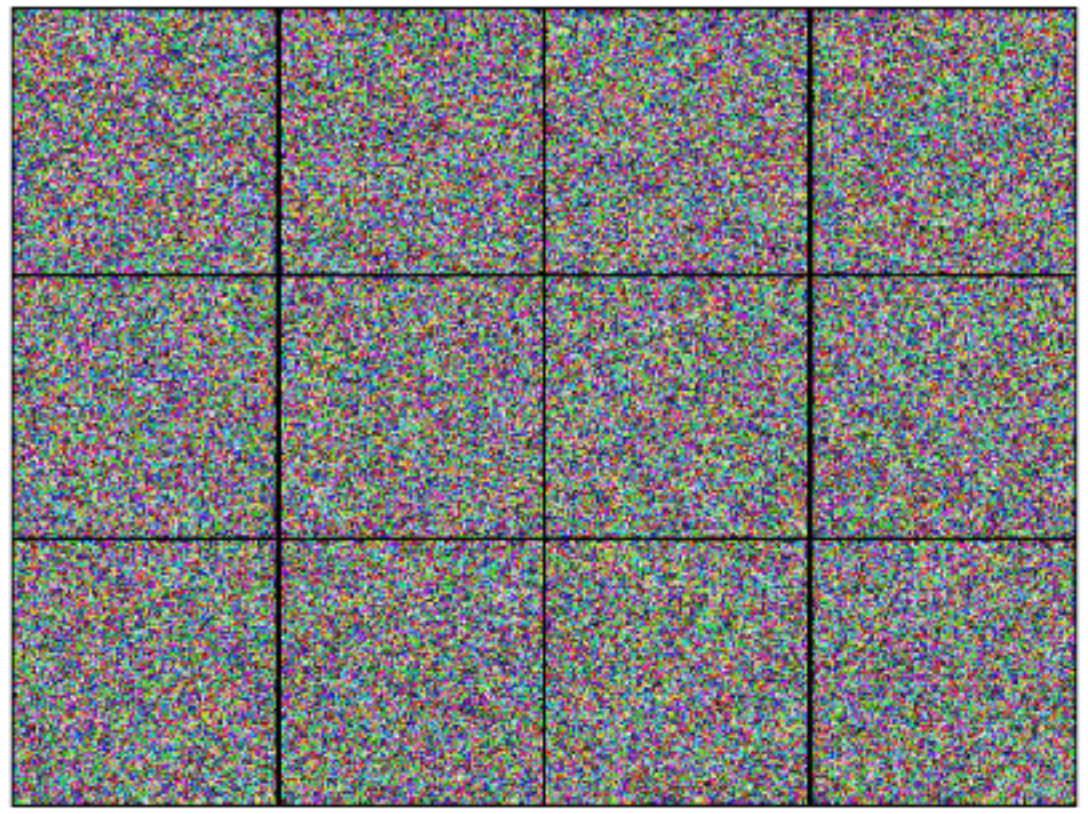
Adjoint Matching

Adjoint Sampling

About me and this talk

Data-driven

- fit to data
- simple & scalable

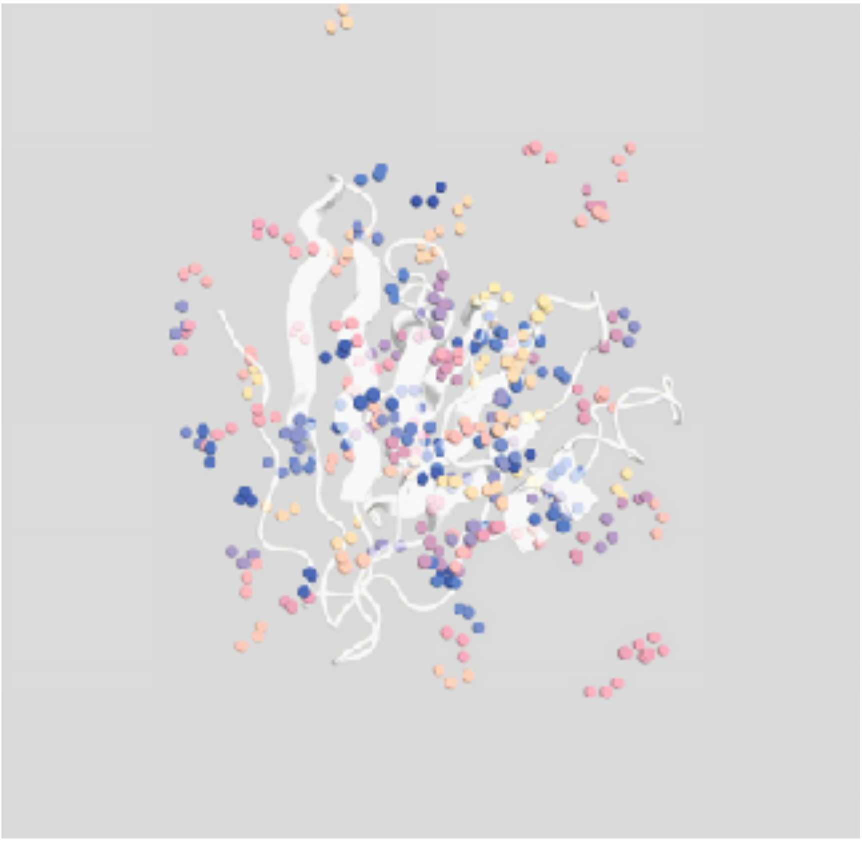


Flow Matching

Meta MovieGen

This Talk

- **Generator Matching**
- **Discrete Flow Matching**
- **Edit Flows**



```
def is_prime(n: int) -> bool:
```

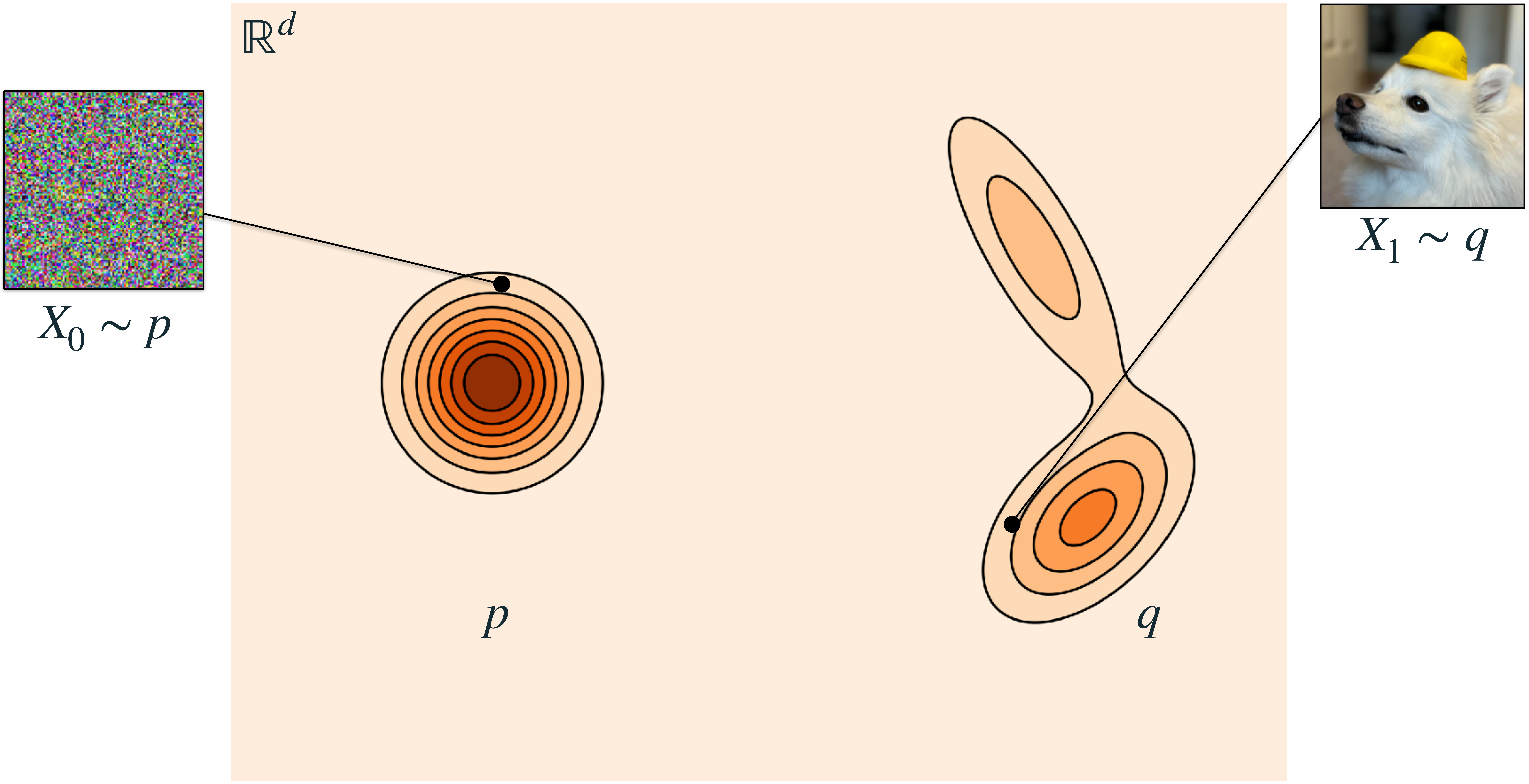
Generator Matching

Extending the Flow Matching recipe to jump Markov processes

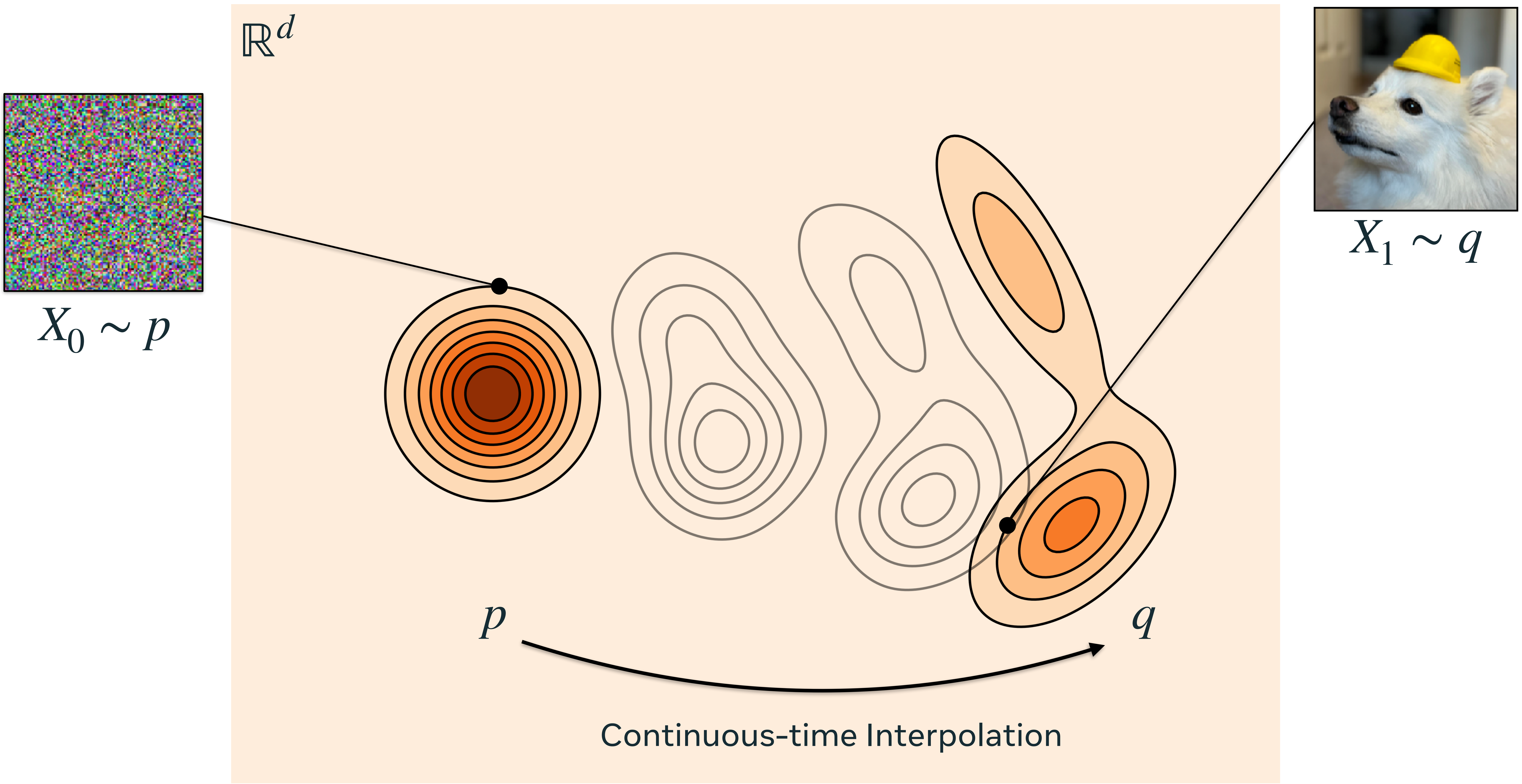
The Generative Modeling problem

$\mathbb{R}^d$

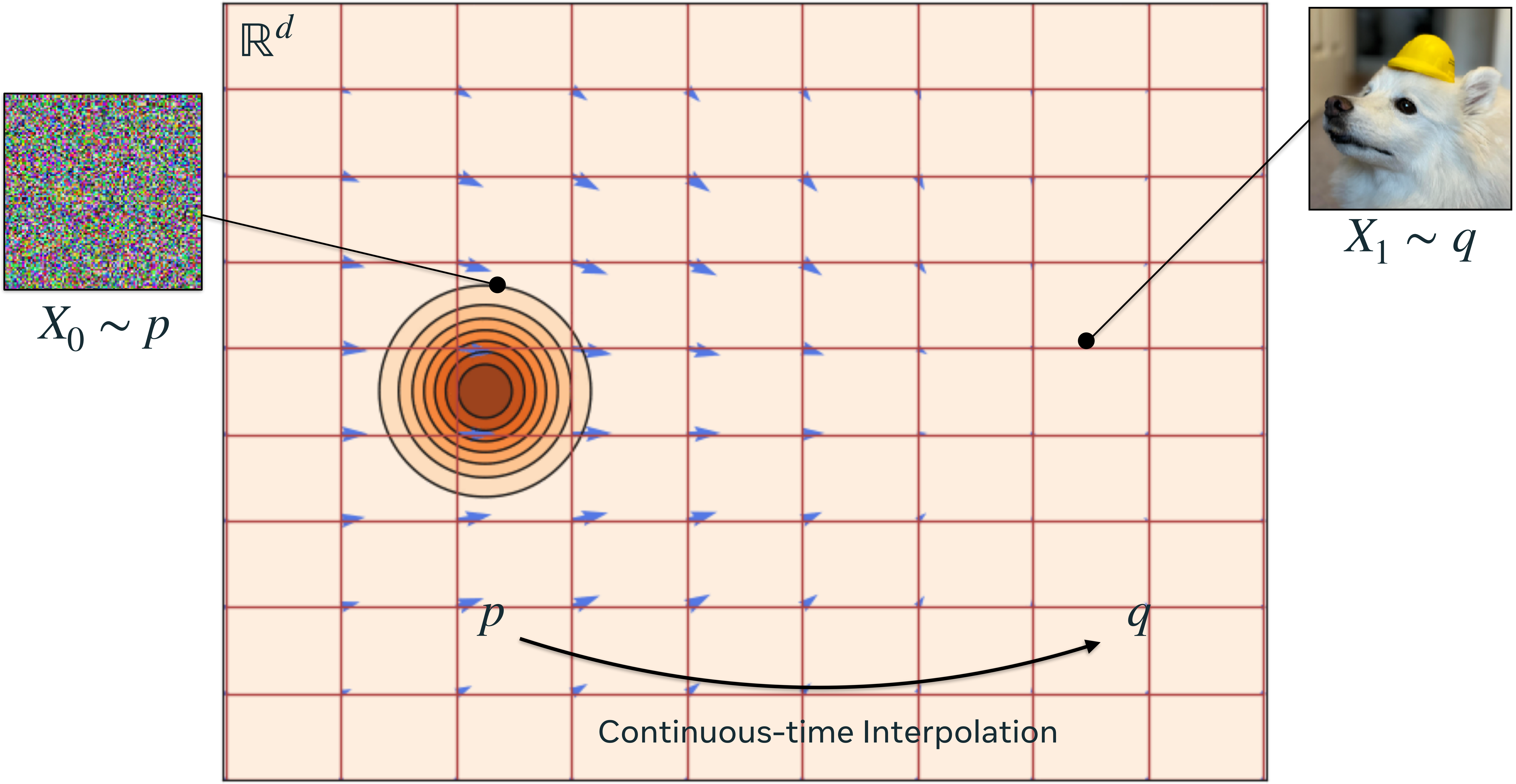
The Generative Modeling problem



The Generative Modeling problem



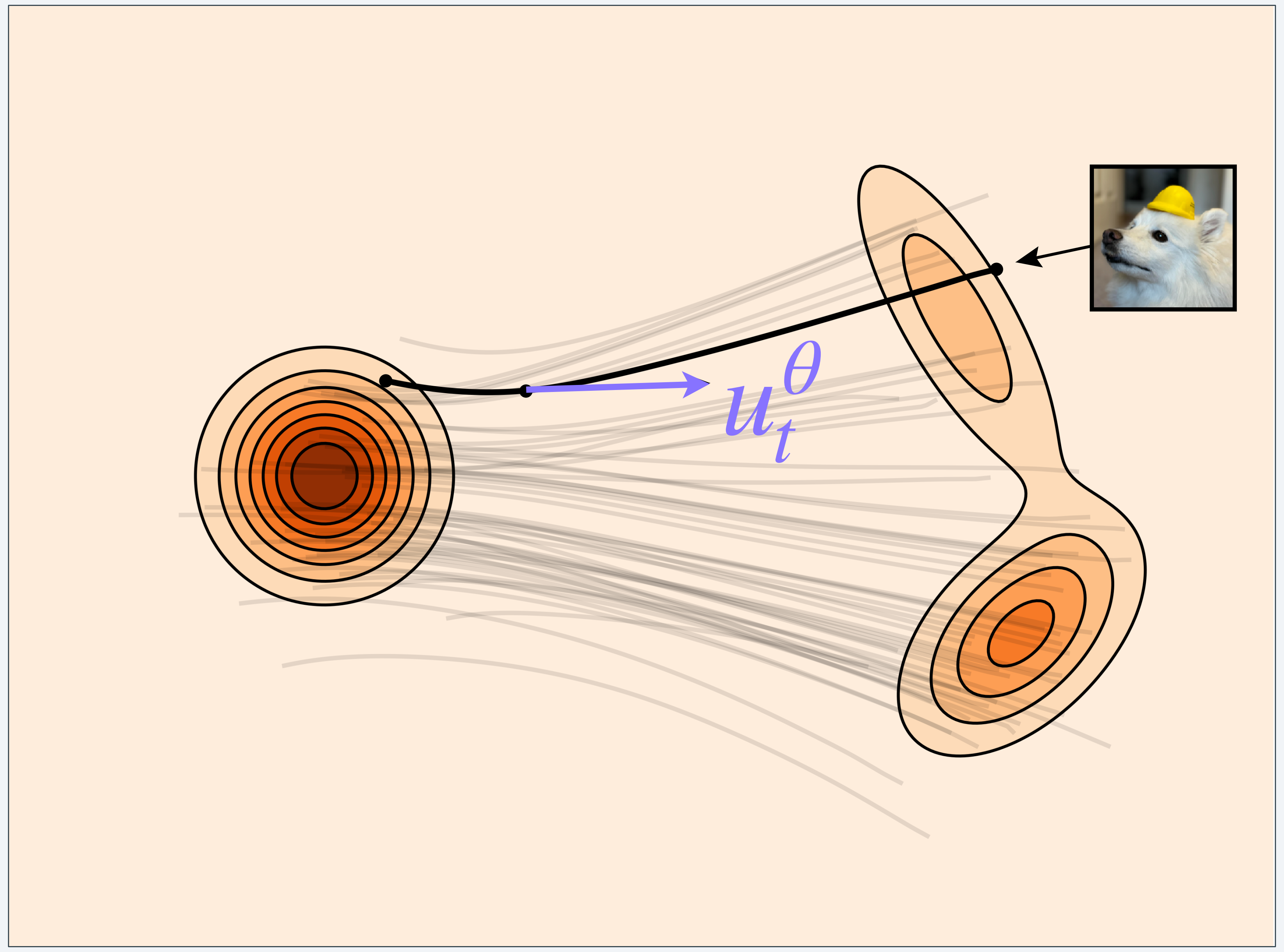
The Generative Modeling problem



Continuous-time Markov Processes

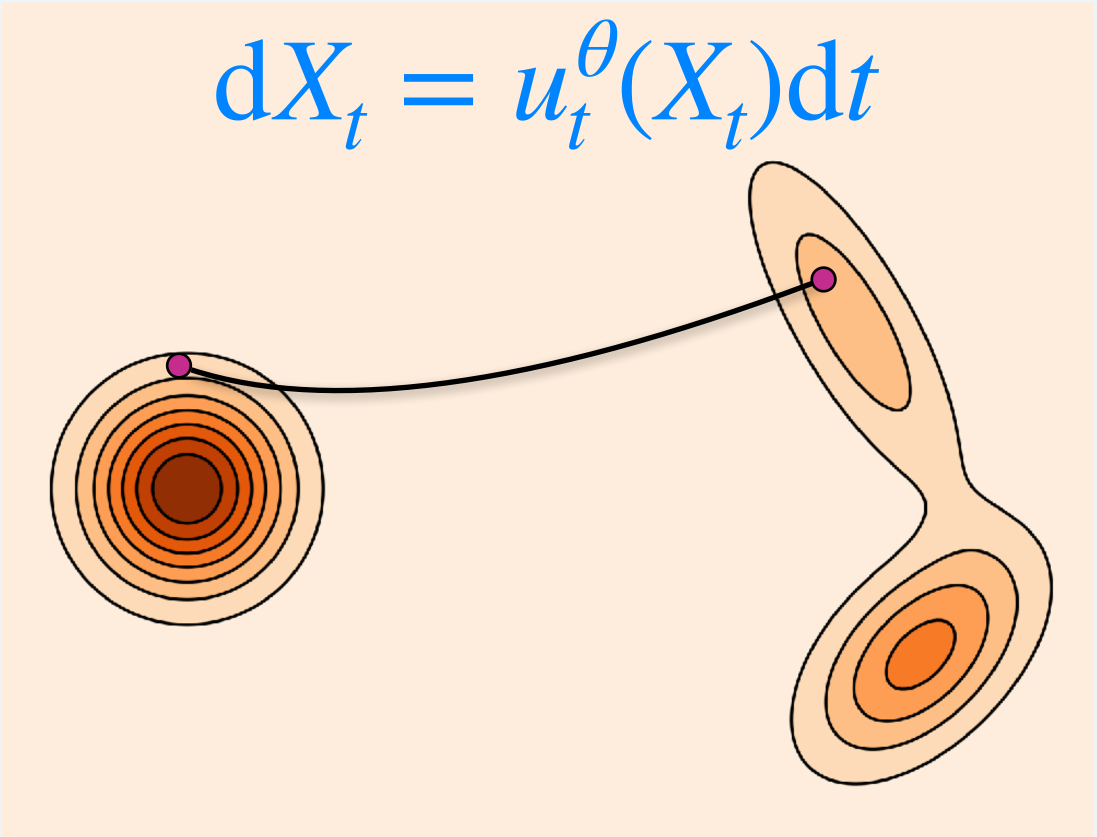
Parameterization

A vector field u_t^θ that points towards the data

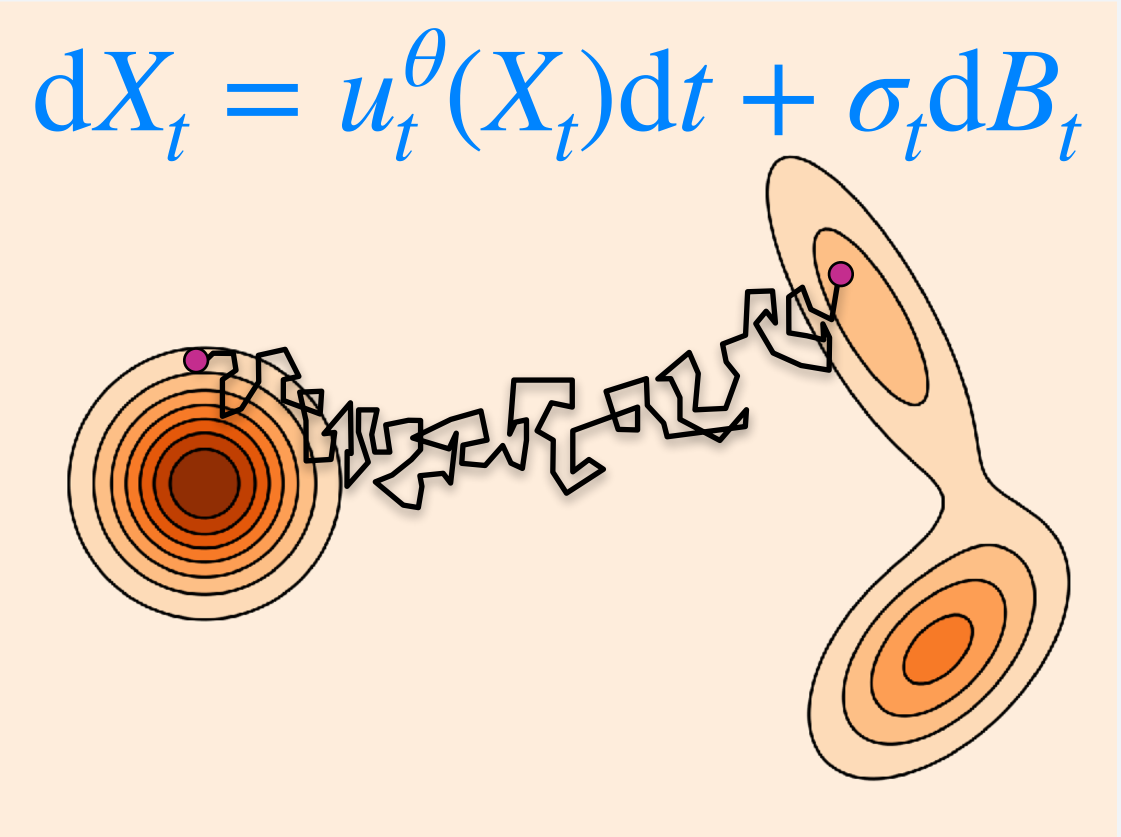


Sampling

A diffusion process X_t with (optional) diffusion coefficient σ_t



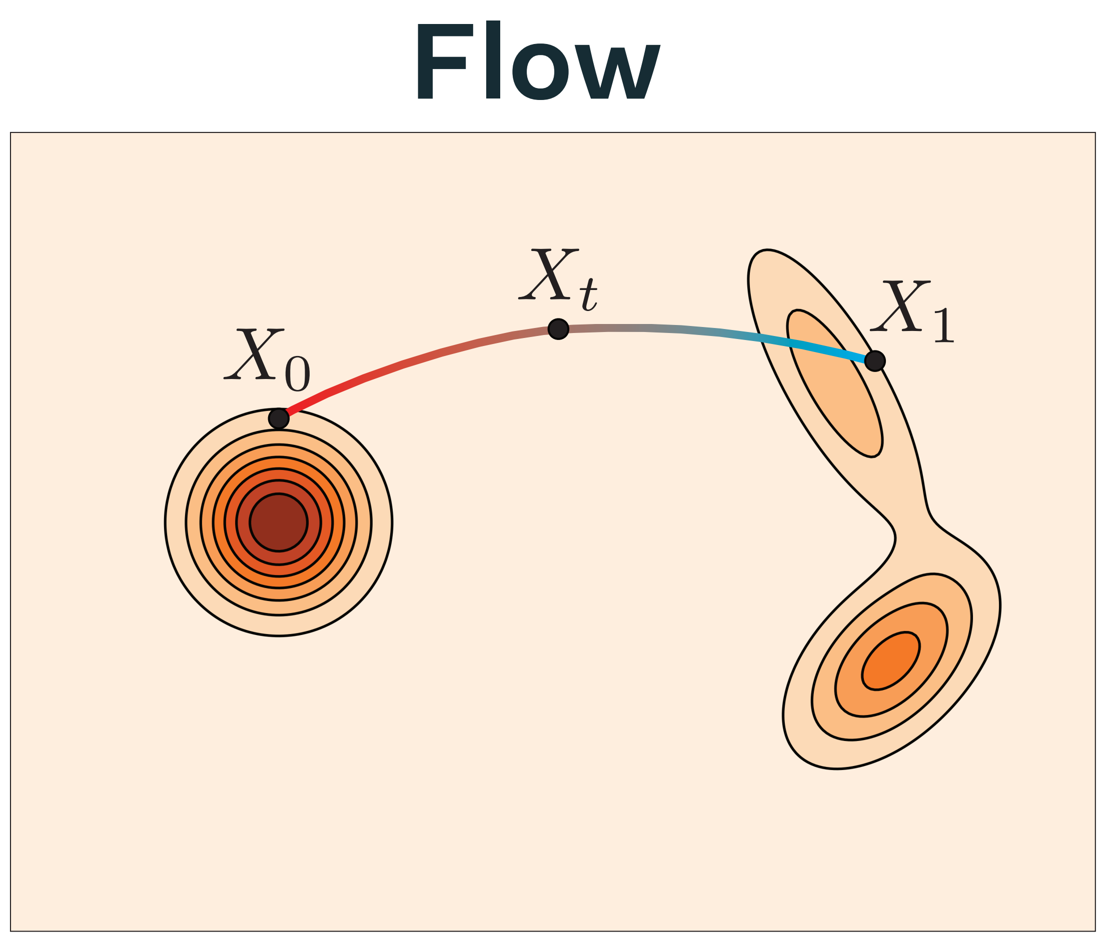
Ordinary Differential Equation

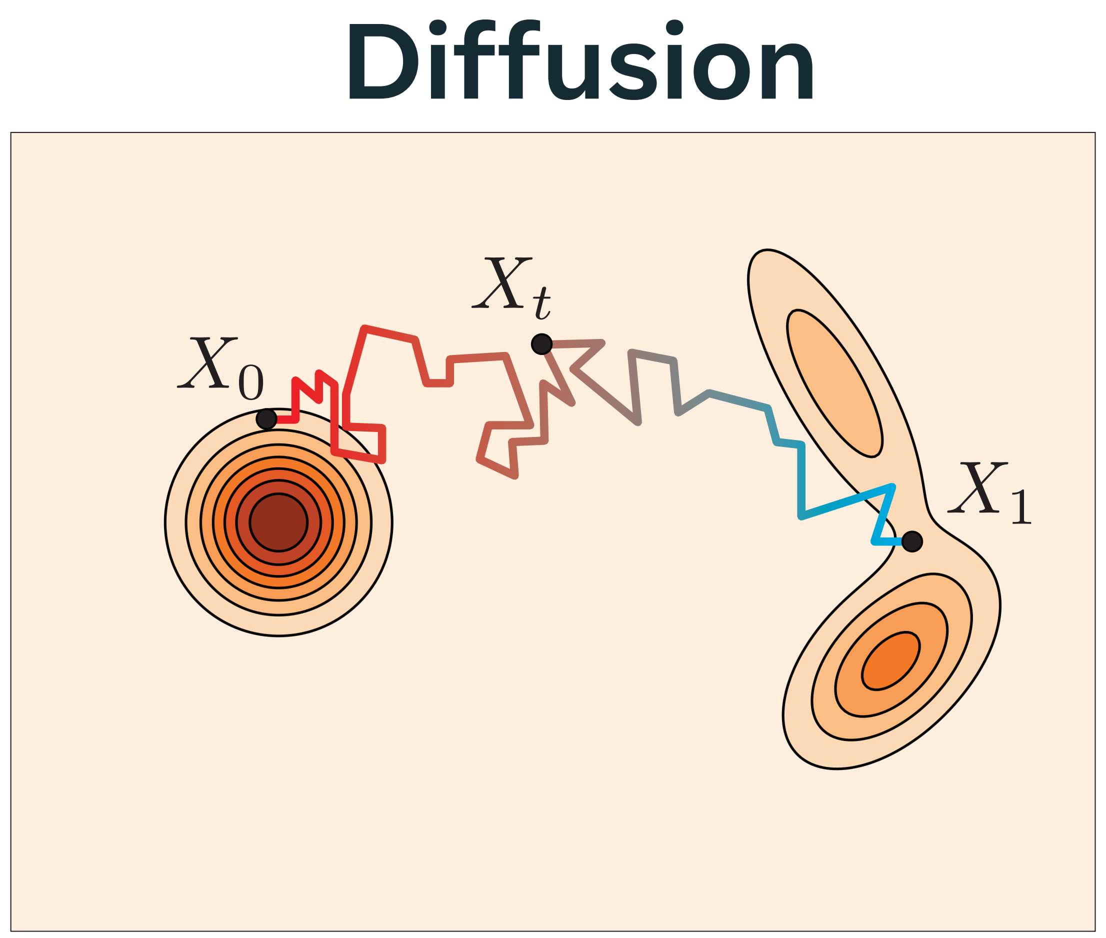


Stochastic Differential Equation

Continuous-time Markov Processes

Local perturbations only





Transition kernel

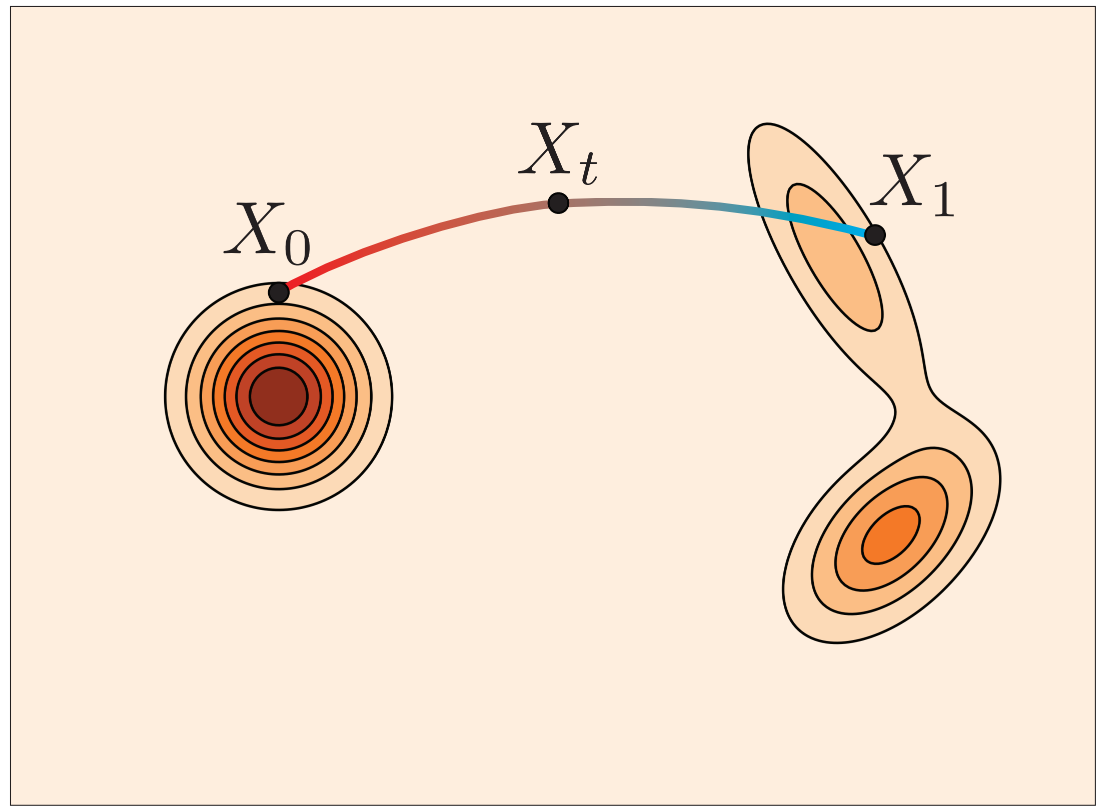
$$X_{t+h} \sim p_{t+h|t}(\,\cdot\,, X_t)$$

$$X_0 \sim p$$

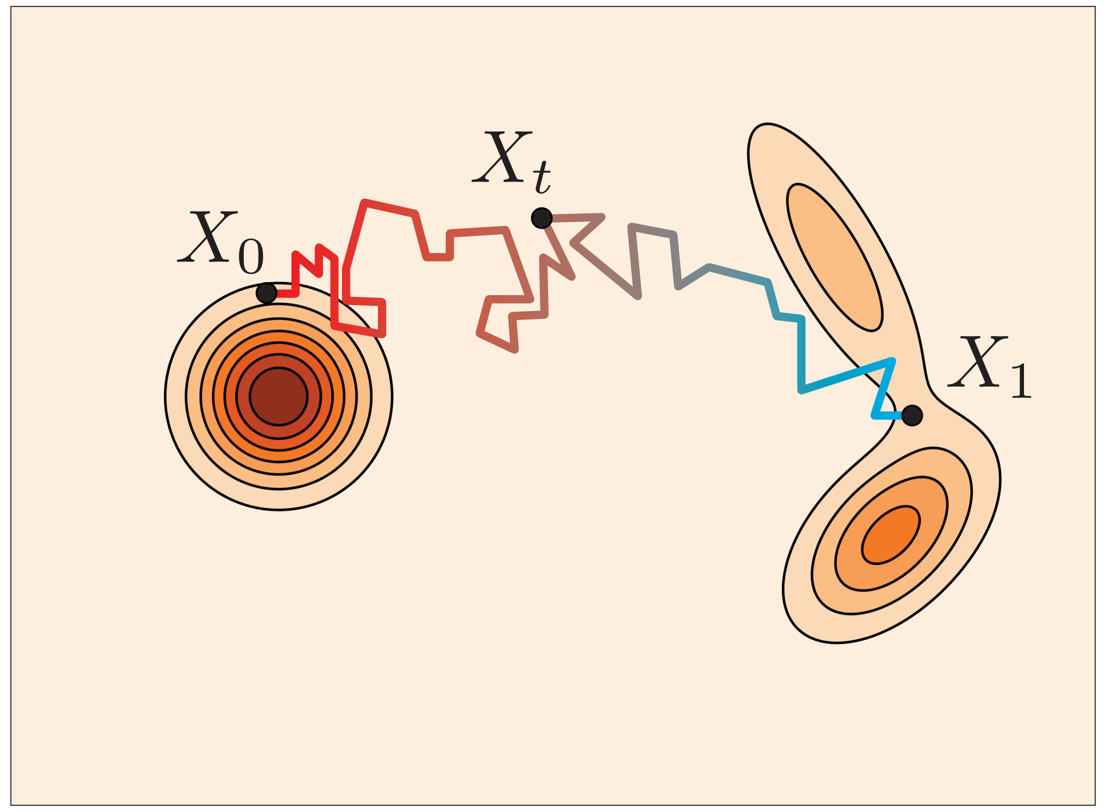
Generalizing beyond local perturbations

Local perturbations only

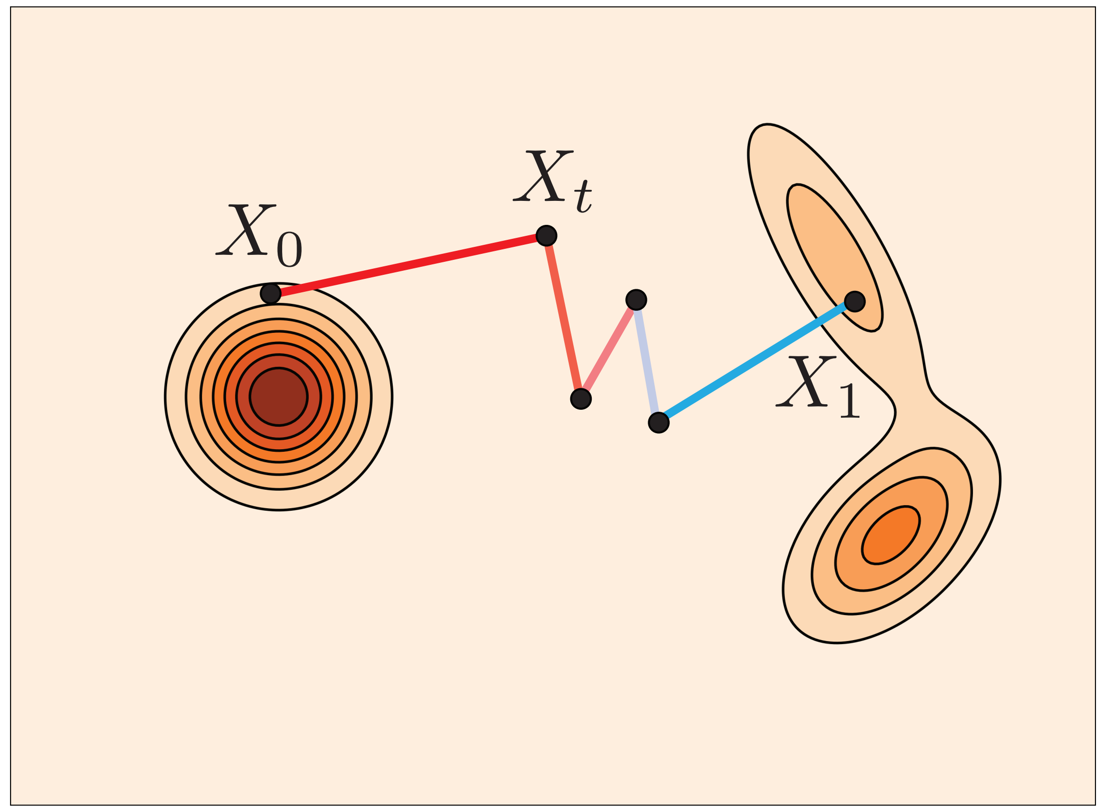
Flow



Diffusion

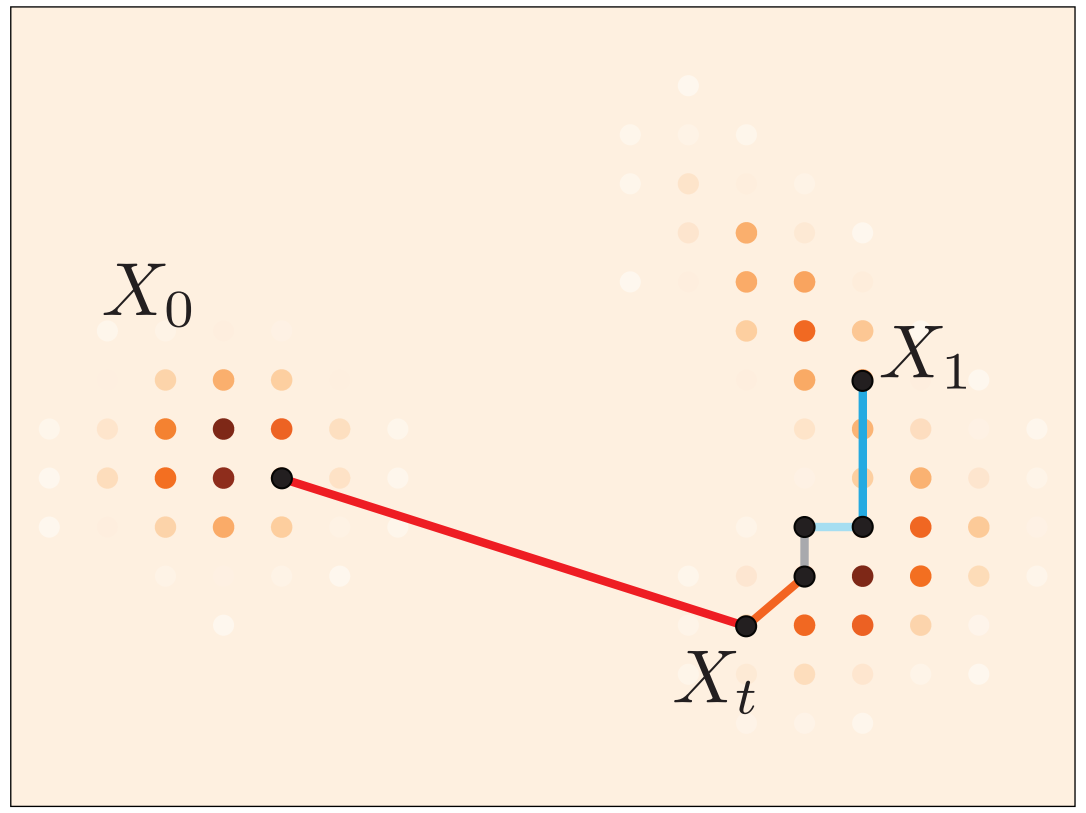


Jump



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

CTMC



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

Transition kernel

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

$$X_0 \sim p$$

Infinitesimal Generator

Generalize the notion of **first-order characterization**

Generator

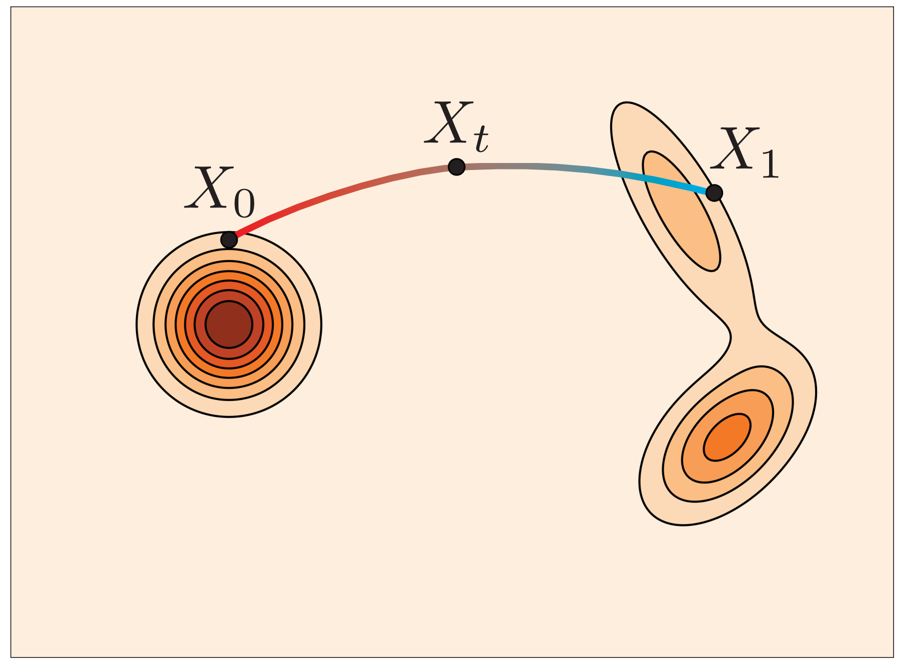
$$\underbrace{p_{t+h|t}(\,\cdot\,, X_t) = \delta_{X_t}}_{0^{\text{th}}\ \textbf{order}} + h \underbrace{\boxed{\frac{\mathrm{d}}{\mathrm{d}h}\bigg|_{h=0} p_{t+h|t}(\,\cdot\,, X_t)}}_{1^{\text{st}}\ \textbf{order}} + \underbrace{o(h)}_{\textbf{error}}$$

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

First-order characterizations

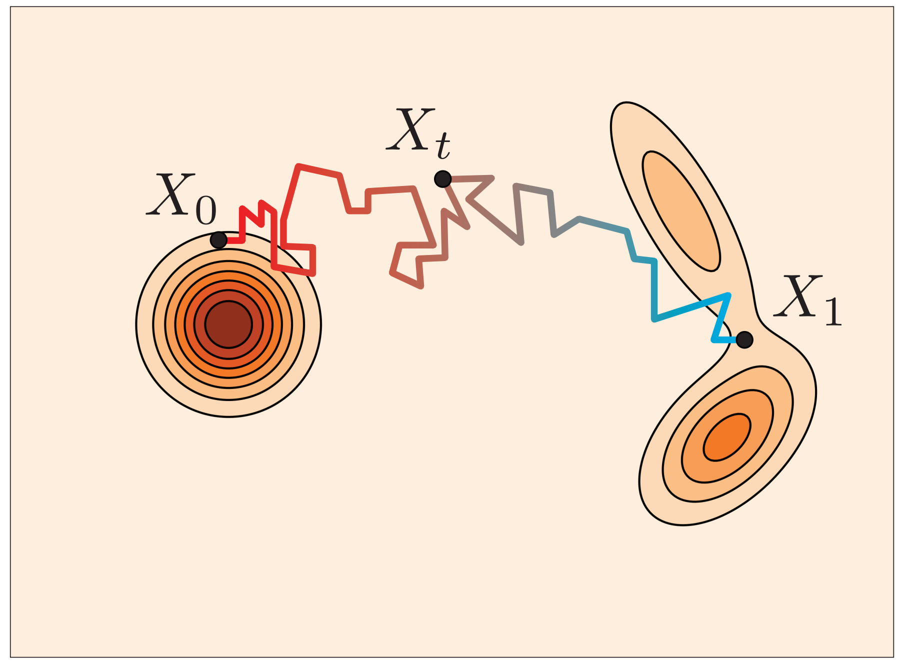
ODE:

$$X_{t+h} = \underbrace{X_t}_{0^{\text{th}}\text{ order}} + \underbrace{h u_t(X_t)}_{1^{\text{st}}\text{ order}} + \underbrace{o(h)}_{\text{error}}$$



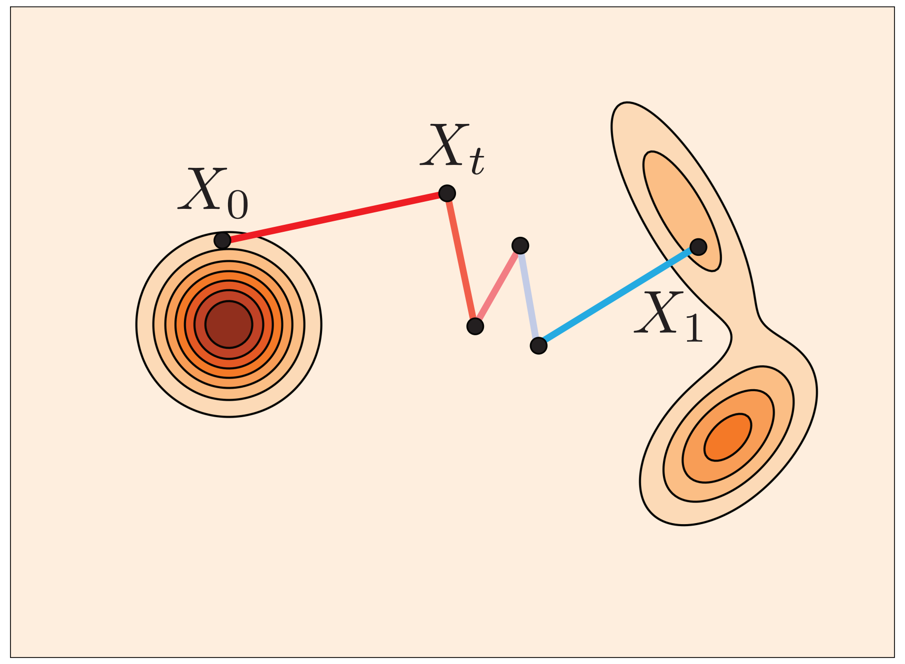
SDE:

$$X_{t+h} = \underbrace{X_t}_{0^{\text{th}}\text{ order}} + \underbrace{h u_t(X_t) + \sqrt{h\sigma_t}\varepsilon}_{1^{\text{st}}\text{ order}} + \underbrace{o(h)}_{\text{error}}$$



Jump & CTMC:

$$X_{t+h} \sim \underbrace{\delta_{X_t}(\,\cdot\,)}_{0^{\text{th}}\text{ order}} + \underbrace{h u_t(\,\cdot \mid X_t)}_{1^{\text{st}}\text{ order}} + \underbrace{o(h)}_{\text{error}}$$

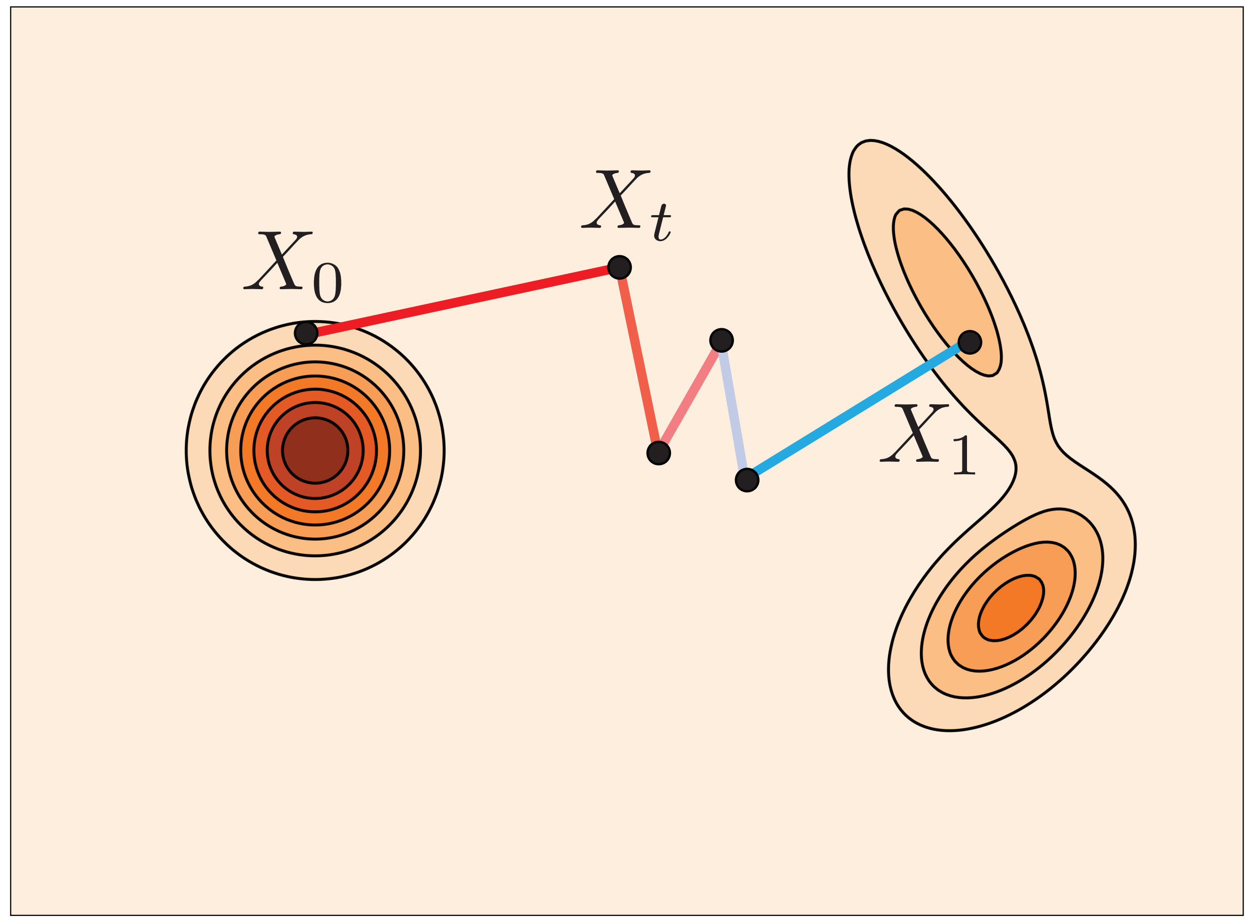


"Generator Matching: Generative modeling with arbitrary Markov processes" Holderrieth et al. (2024)

Jump Markov process

$$X_{t+h} \sim \delta_{X_t}(\cdot) + h\boxed{u_t(\cdot \mid X_t)}$$

Unnormalized distribution



Interpret $u_t(\cdot \mid X_t) = \boxed{\lambda_t(X_t)}\boxed{Q_t(\cdot \mid X_t)}$

Jump rate

Normalized

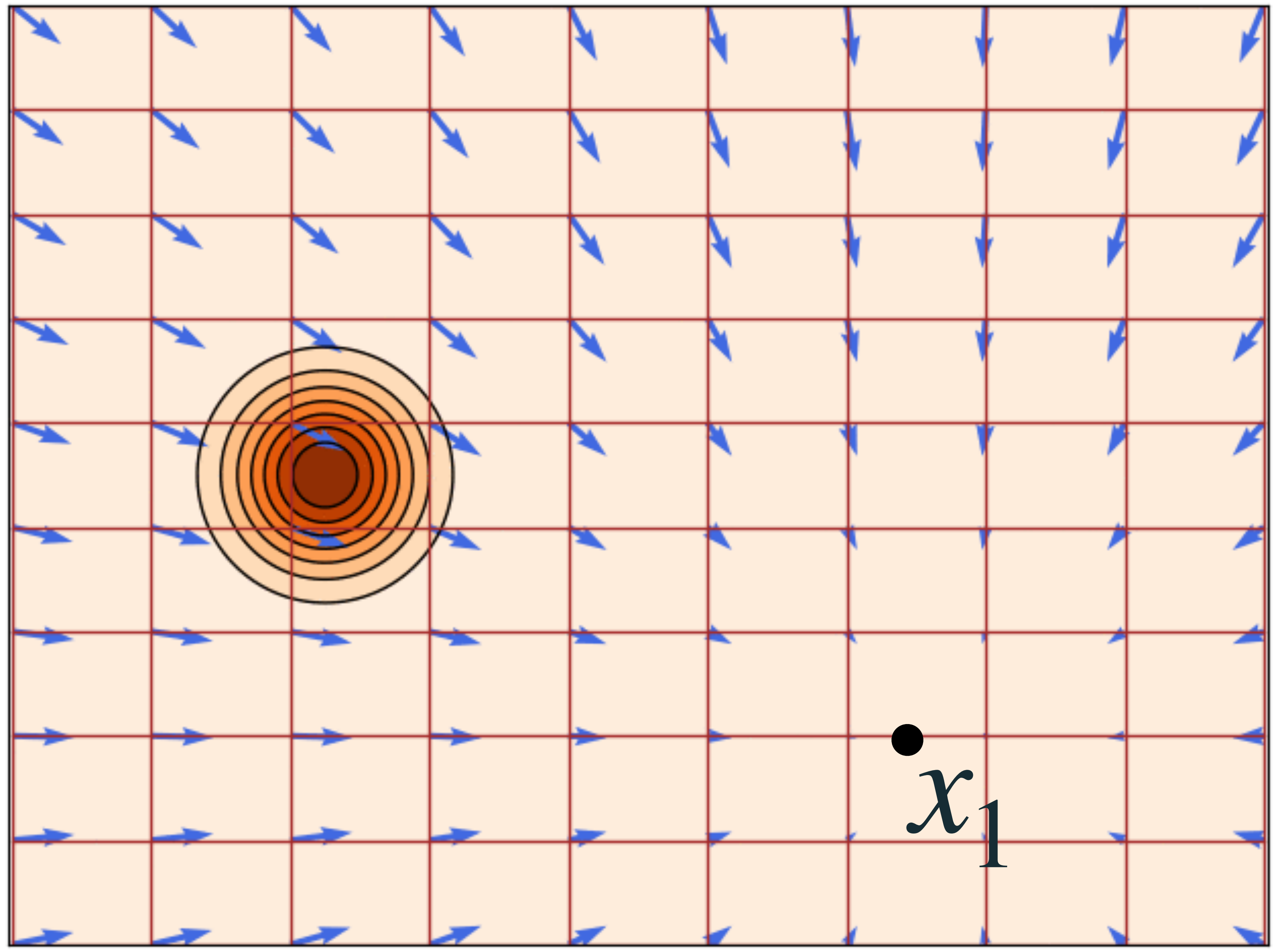
Jump process (sampling)

$X_{t+h} \sim Q_t(\cdot \mid X_t)$ if $h\lambda_t(X_t) \geq U[0,1]$

$X_{t+h} = X_t$ otherwise

Build flow from conditional flows

Generate a single target point

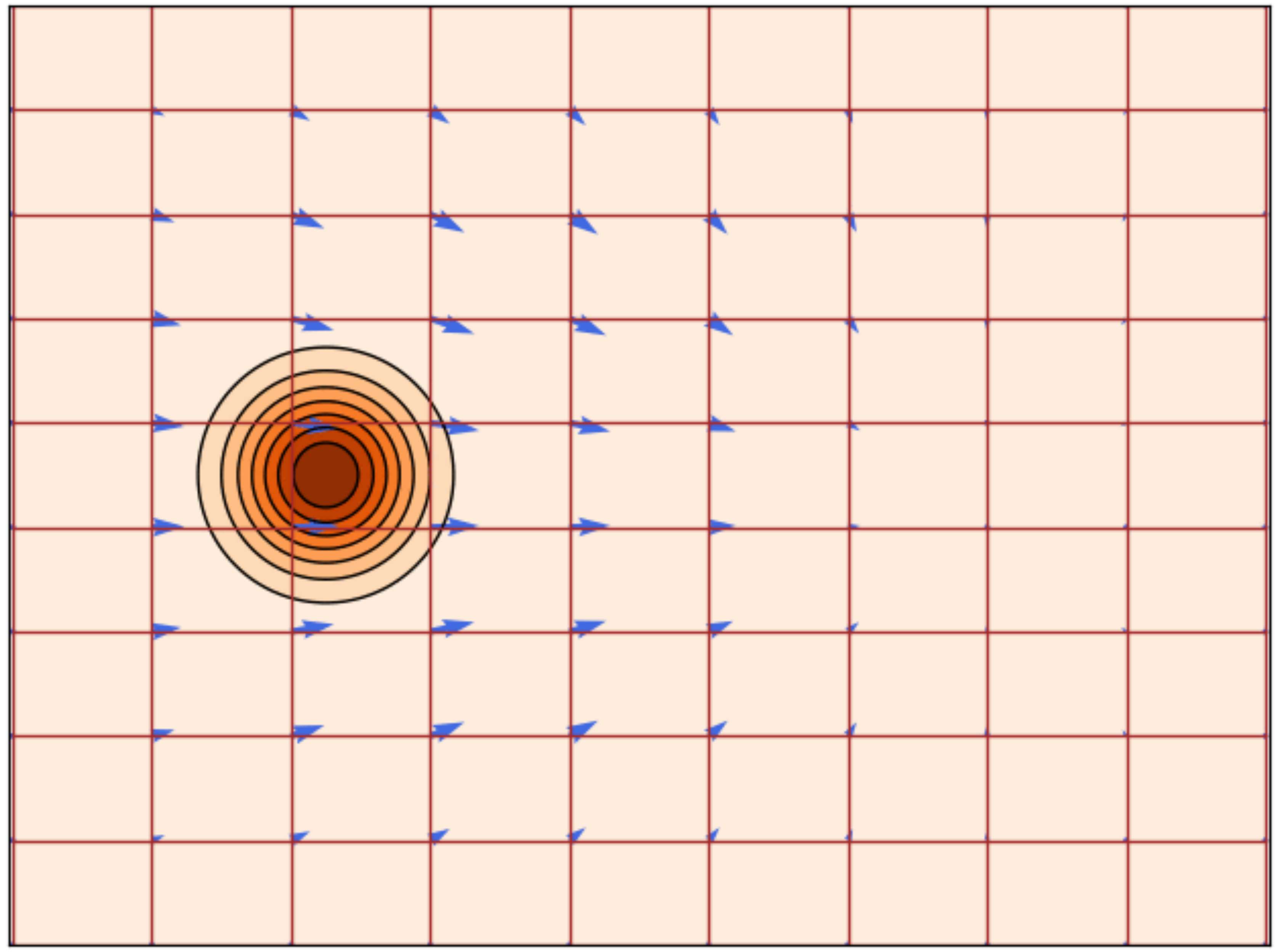


$$X_t = (1-t)X_0 + tX_1$$

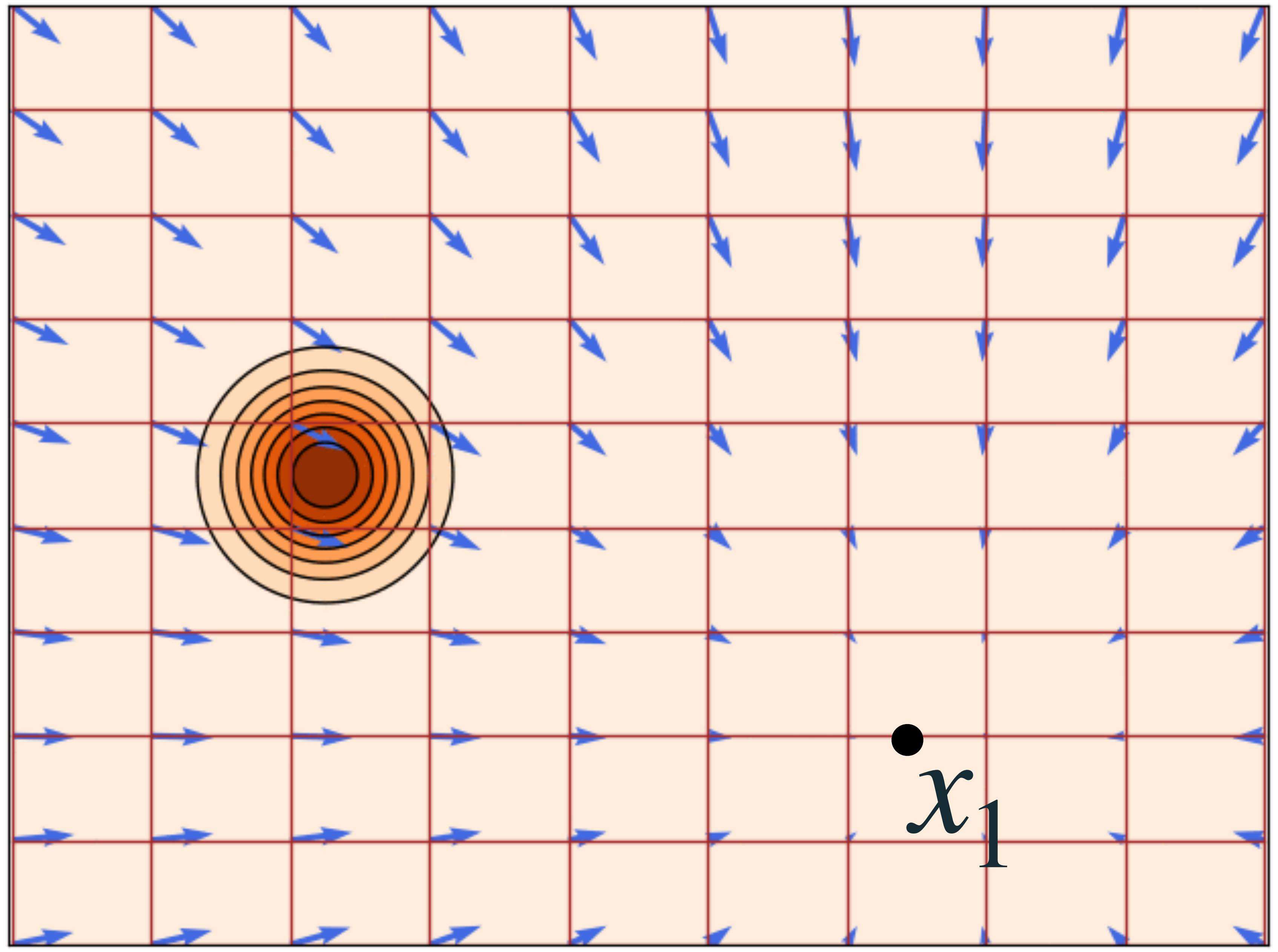
$p_{t|1}(x \mid x_1)$ conditional probability

$u_t(x \mid x_1)$ conditional velocity

Build flow from conditional flows



Generate a single target point



$$X_t = (1-t)X_0 + tX_1$$

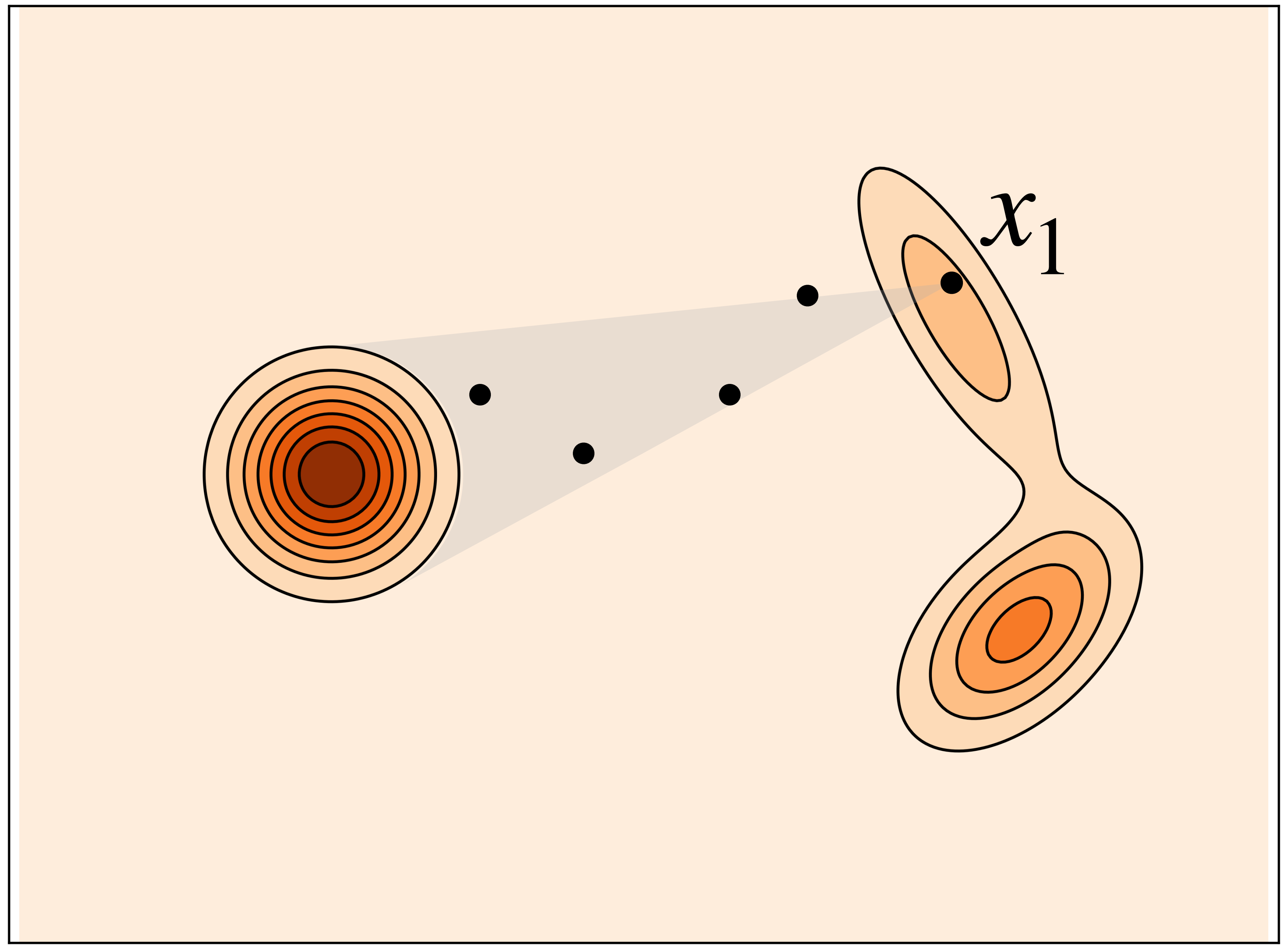
$$p_t(x) = \mathbb{E}_{X_1} p_{t|1}(x|X_1) \longleftarrow p_{t|1}(x|x_1)$$ conditional probability

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

Building **generator** from conditional **generators**

Identical recipe as the flow case...



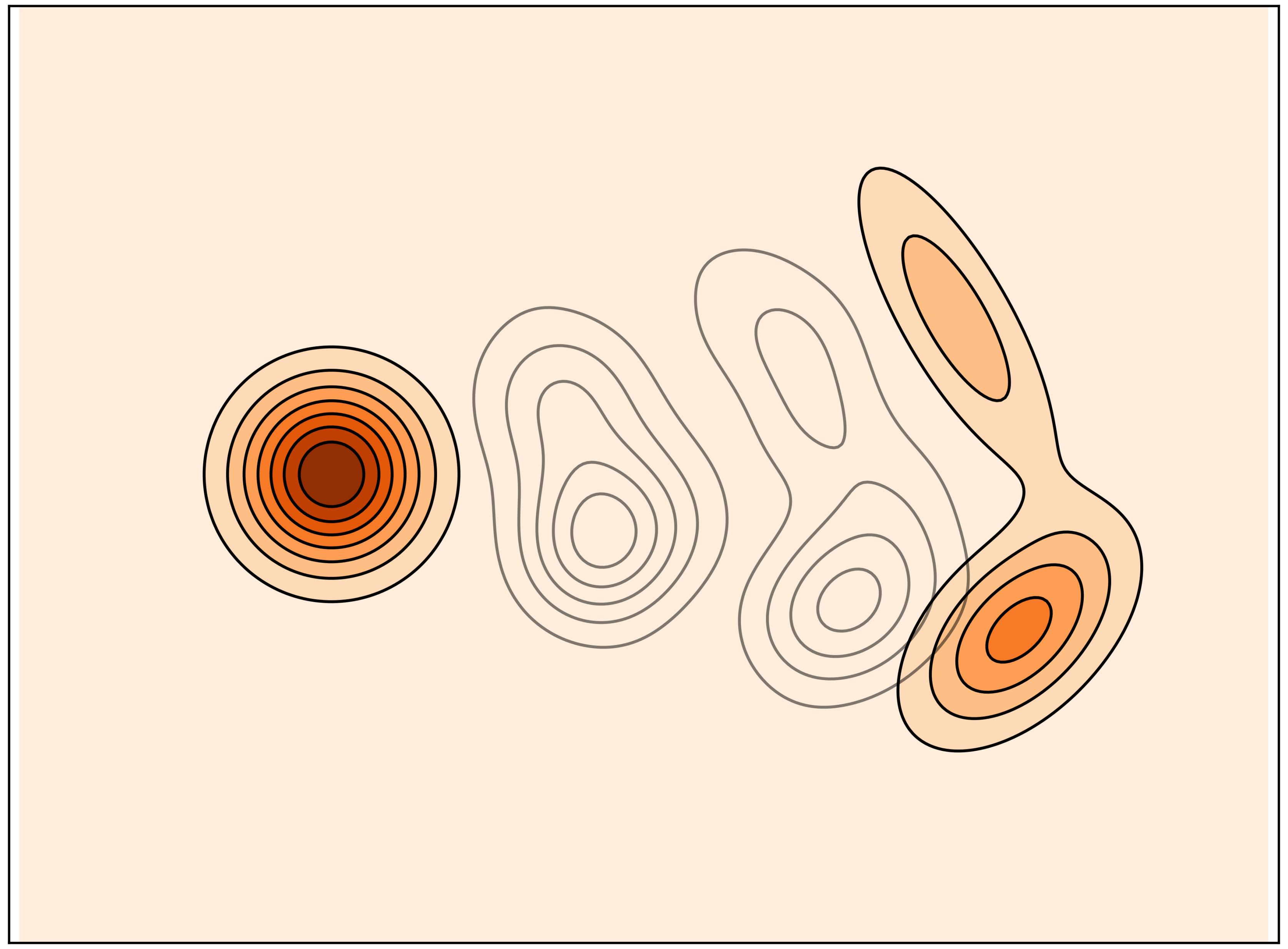
$p_{t|1}(x \,|\, x_1)$ conditional probability

$u_t(\,\cdot\,|\, x, x_1)$ conditional **generator**

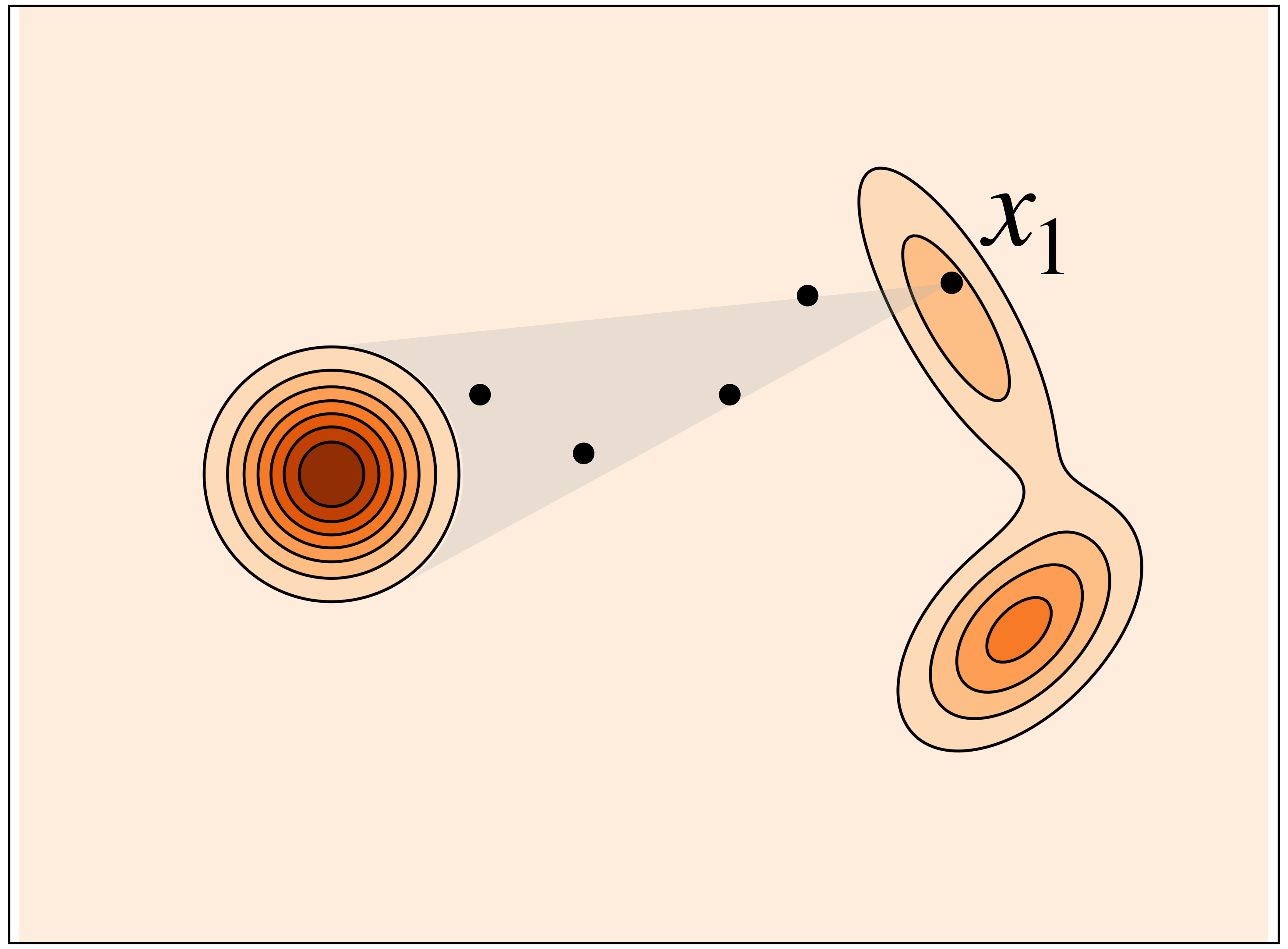
"Generator Matching: Generative modeling with arbitrary Markov processes" Holderrieth et al. (2024)

Building **generator** from conditional **generators**

Identical recipe as the flow case...



Generate a single target point



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

average

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

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

The Marginalization Trick

Theorem: The marginal generator generates the marginal probability path.

Flow and diffusion:

$$u_t(x) = \mathbb{E}\left[u_t(X_t \mid X_1) \mid X_t = x\right]$$

$$p_t(x) = \mathbb{E}_{X_1} p_{t|1}(x \mid X_1)$$

Jump and CTMC:

$$u_t(\,\cdot \mid x) = \mathbb{E}\left[u_t(\,\cdot \mid X_t, X_1) \mid X_t = x\right]$$

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

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

Flow Matching Loss

- **Flow Matching loss:**

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

- **Conditional Flow Matching loss:**

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

Theorem: Losses are equivalent,

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

Generalized Flow Matching Loss

- **Flow Matching loss:**

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

- **Conditional Flow Matching loss:**

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

Theorem: Losses are equivalent <u>if and only if</u> 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 Flow Matching Loss

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

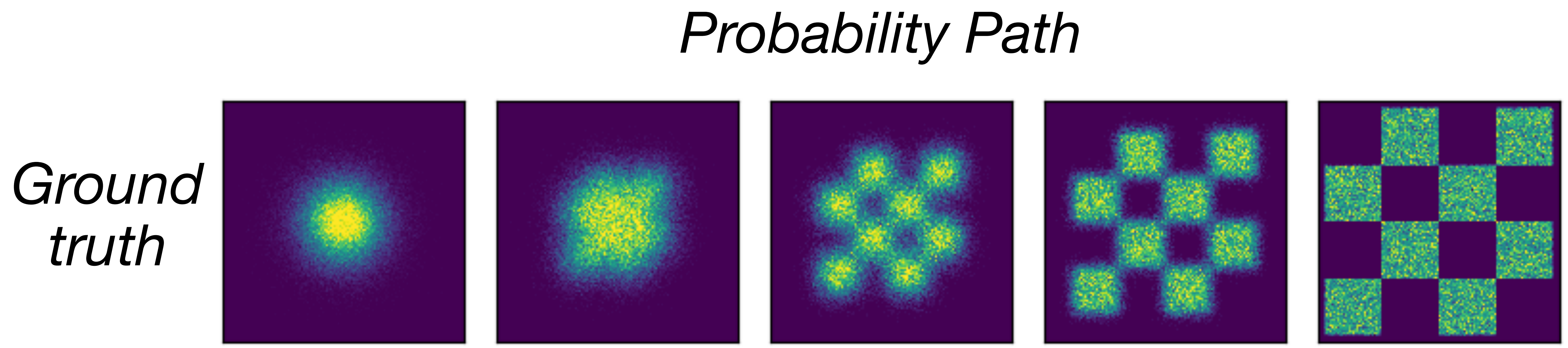
$$\nabla_\theta \mathbb{E}_{X,Y} D(Y, g^\theta(X)) = \nabla_\theta \mathbb{E}_X D(\mathbb{E}[Y \,|\, X], g^\theta(X))$$

$$D(a,b) = \phi(a) - [\phi(b) + \langle a - b, \nabla \phi(b) \rangle]$$

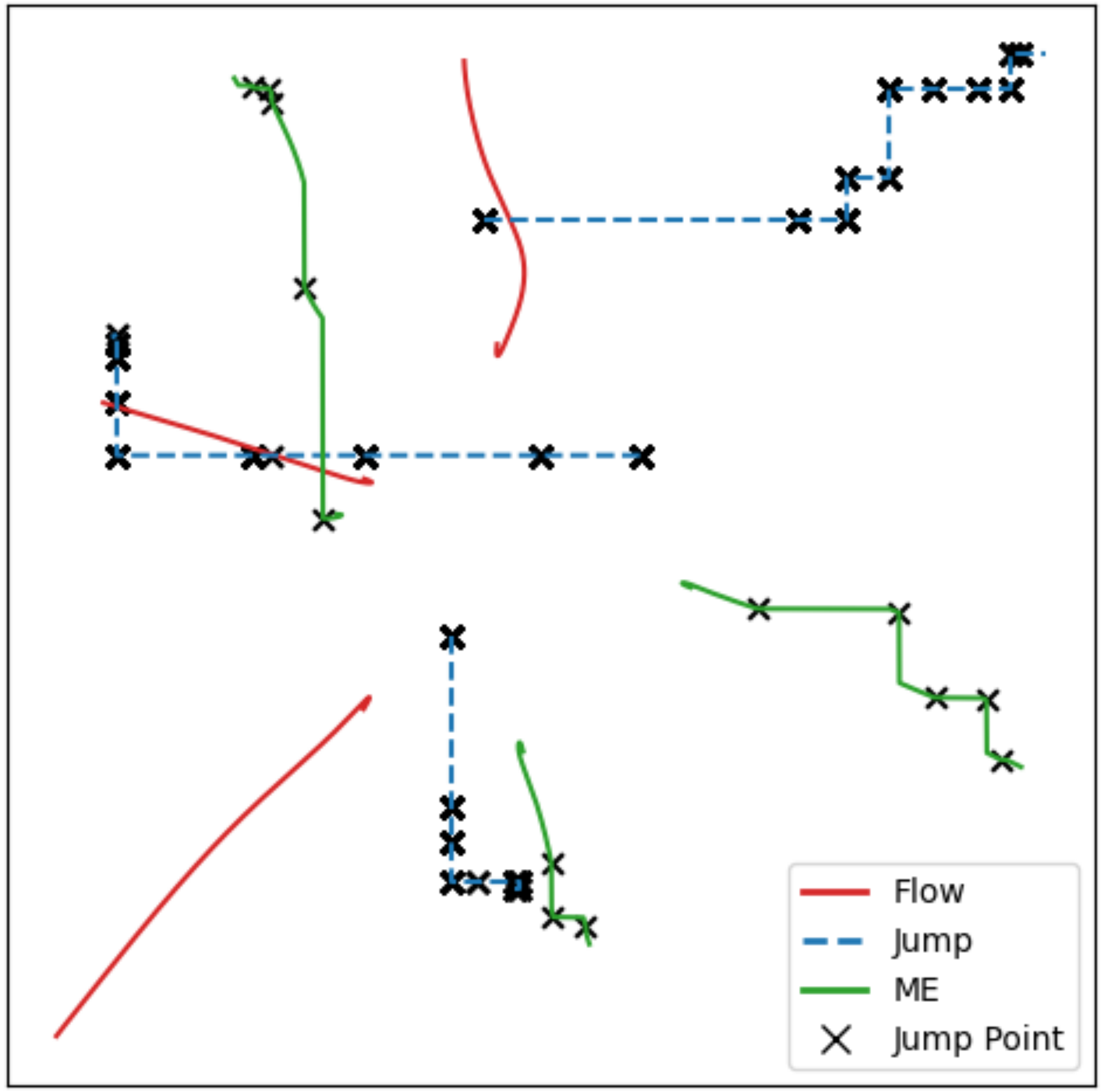
- **Includes MSE, ELBO, many many possible instantiations ...**

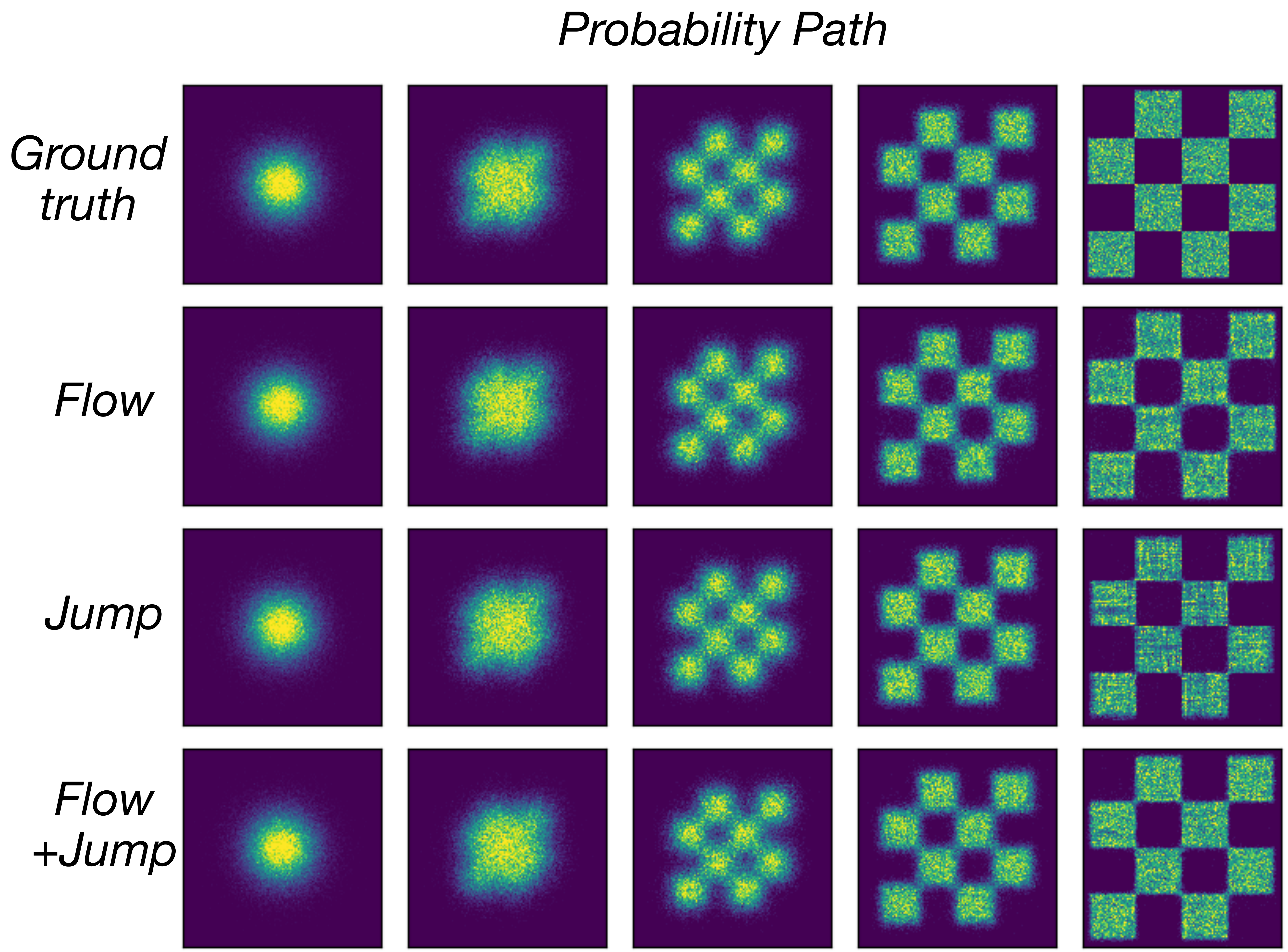
"Generator Matching: Generative modeling with arbitrary Markov processes" Holderrieth et al. (2024)

Toy problem illustration



Toy problem illustration





Different sample paths

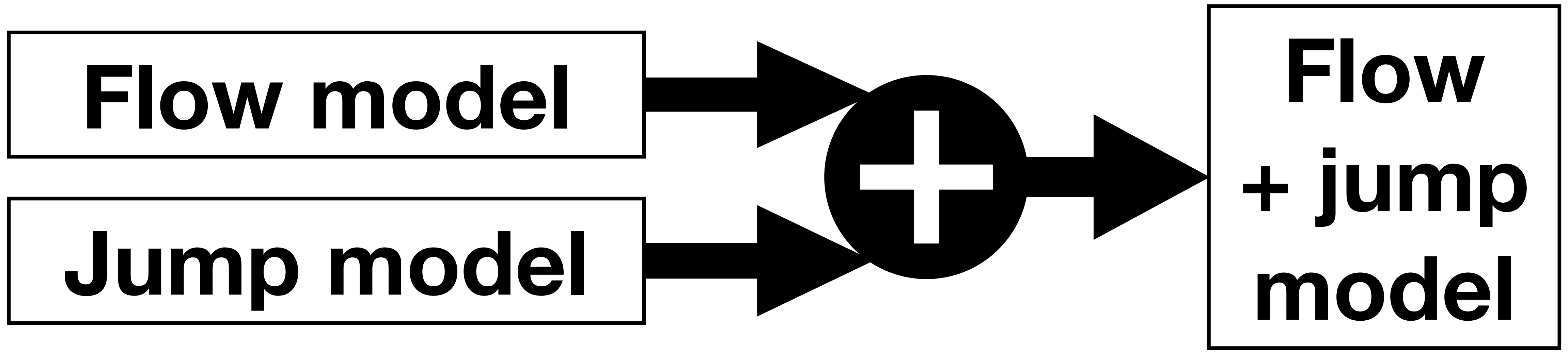
Same probability path

Image Generation

Markov jump model

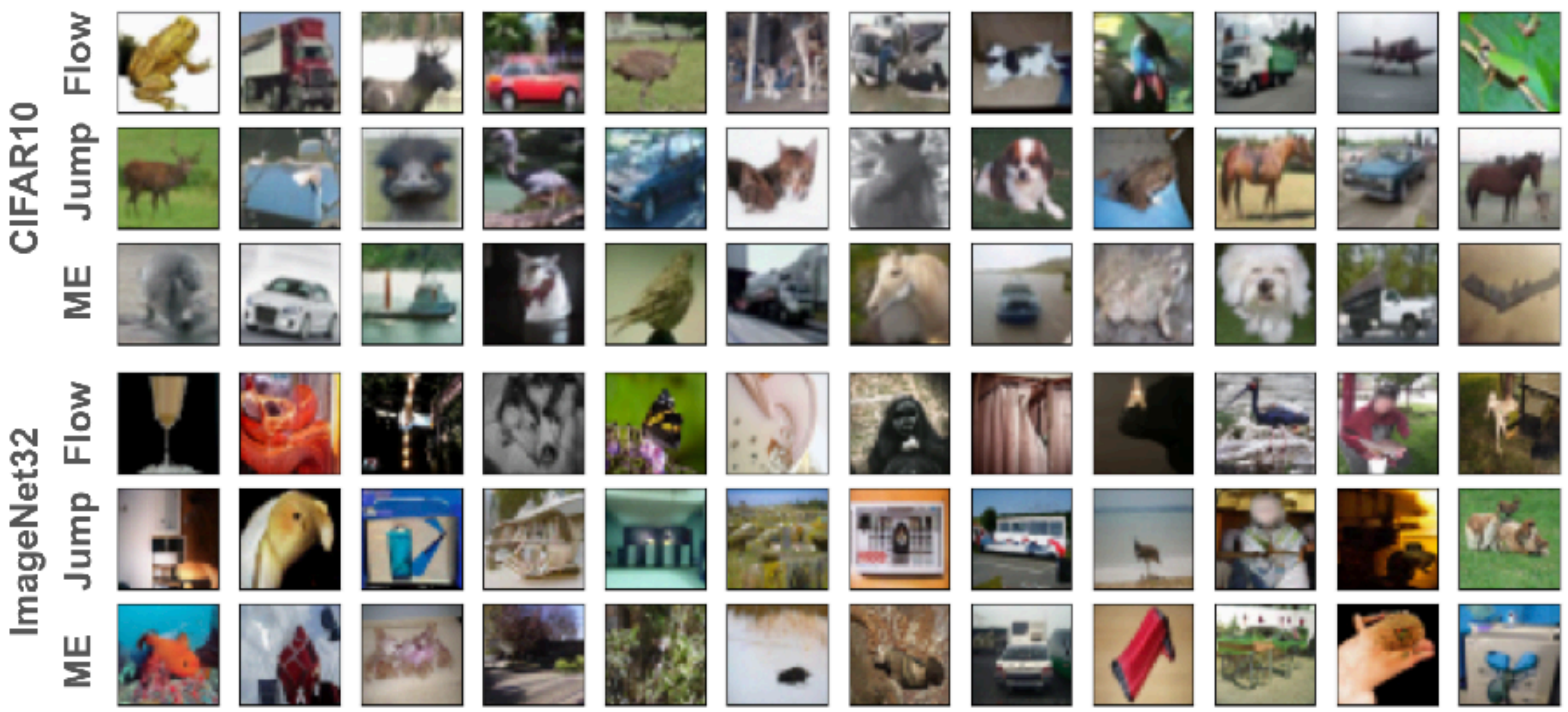
Large unexplored design space!

Markov superposition



Markov superpositions show synergistic effects of two models!

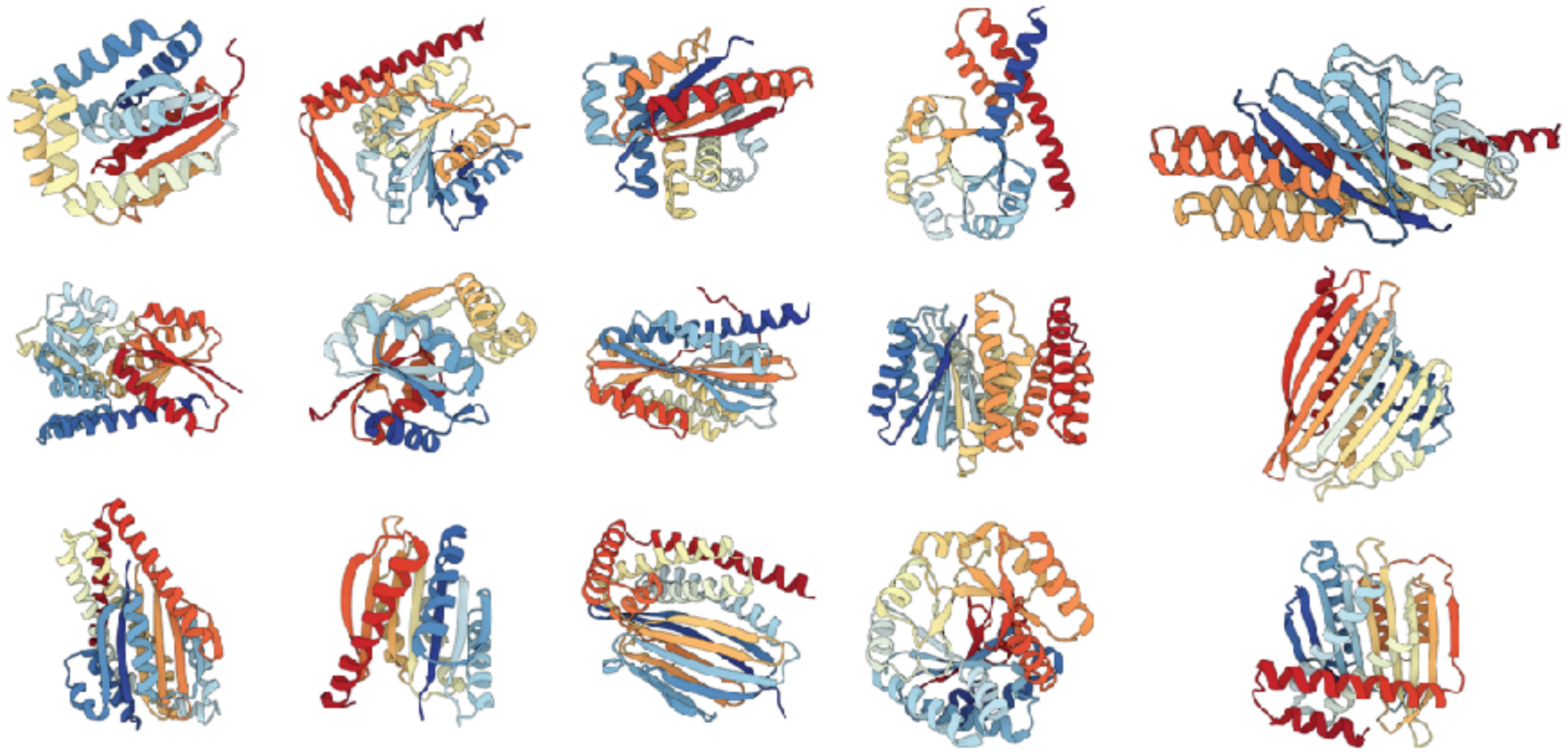
Method	CIFAR10	ImageNet
DDPM (Ho et al., 2020)	3.17	6.99
VP-SDE (Song et al., 2020)	3.01	6.84
EDM (Karras et al., 2022)	1.98	–
Flow model (Euler)	2.94	4.58
Jump model (Euler)	4.23	7.66
Jump + Flow MS (Euler)	**2.49**	**3.47**
Flow model (2nd order)	2.48	3.59
Jump + Flow MS (mixed)	**2.36**	**3.33**

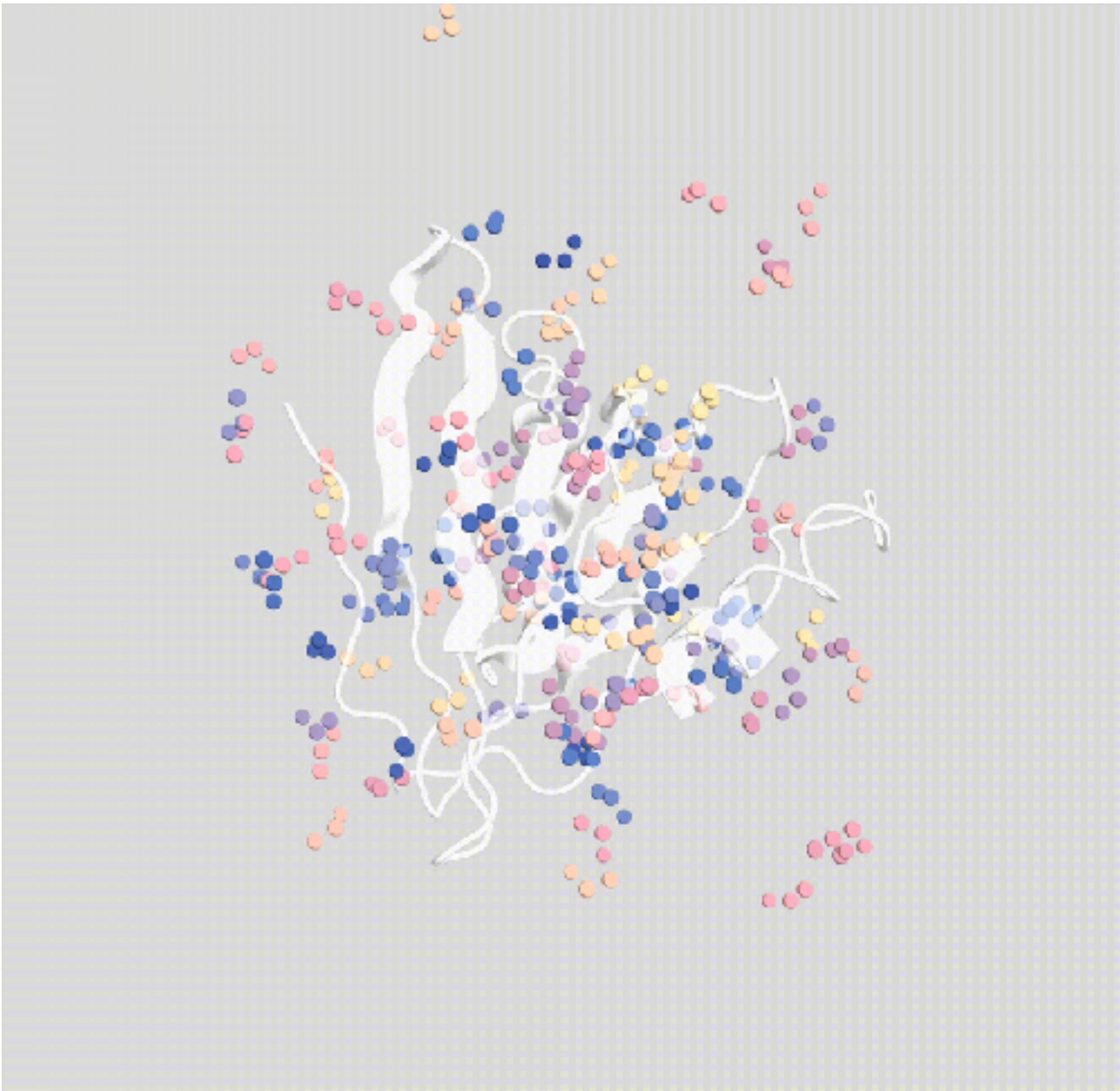


FrameJump : Protein Generation

Manifold jump model on SO(3).

Make **multimodal**: Combine continuous and discrete models





Method	Multimodal Div.	Multimodal Nov.	Unimodal Div.	Unimodal Nov.
RFdiffusion (Watson et al., 2023)	N/A		0.4	0.37
FrameFlow (Yim et al., 2024)	N/A		0.39	0.39
FoldFlow (Bose et al., 2023)	N/A		0.24	0.32
Protpardelle (Chu et al., 2024)	0.1	0.4	0.12	**0.41**
ProteinGenerator (Lisanza et al., 2023)	0.09	0.31	0.19	0.35
MultiFlow (Campbell et al., 2024b)	0.38	0.39	0.52	0.39
w/ $SO(3)$ jumps (ours)	**0.48**	**0.41**	**0.63**	**0.41**
w/ $SO(3)$ jumps + flow (ours)	0.47	0.4	0.59	0.40

Discrete Flow Matching

Discrete space diffusion, general corruption processes

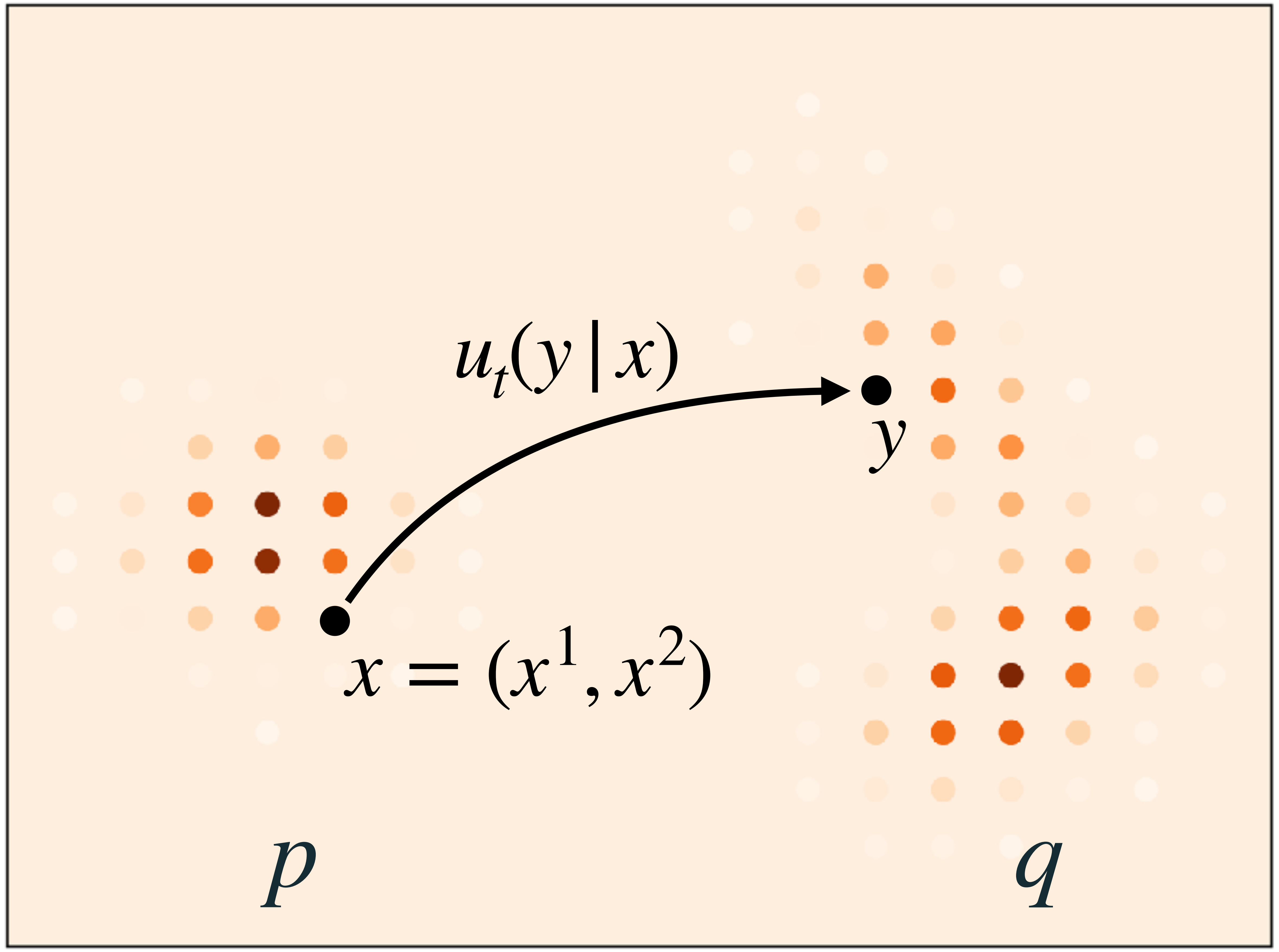
Continuous-time Markov Chains (CTMC)

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

$$p_{t+h|t}(\,\cdot\,, X_t) = \delta_{X_t}(\,\cdot\,) + h\,\boxed{u_t(\,\cdot\,|\,X_t)} + o(h)$$

Unnormalized

distribution over $\mathcal{X}$



“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 velocity

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

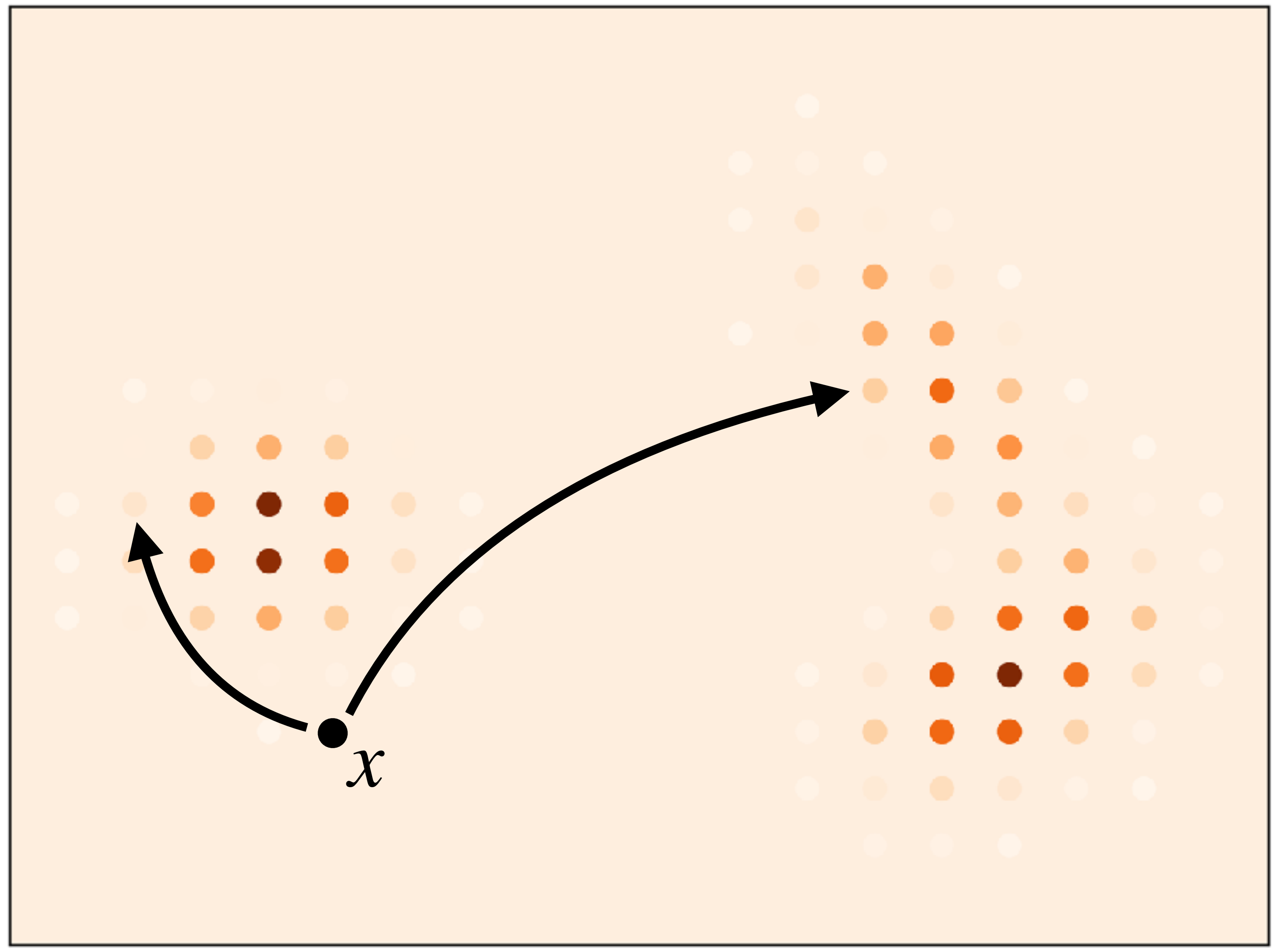
$$u_t(x) = [u_t^1(x), \ldots, u_t^d(x)]$$

Example:

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

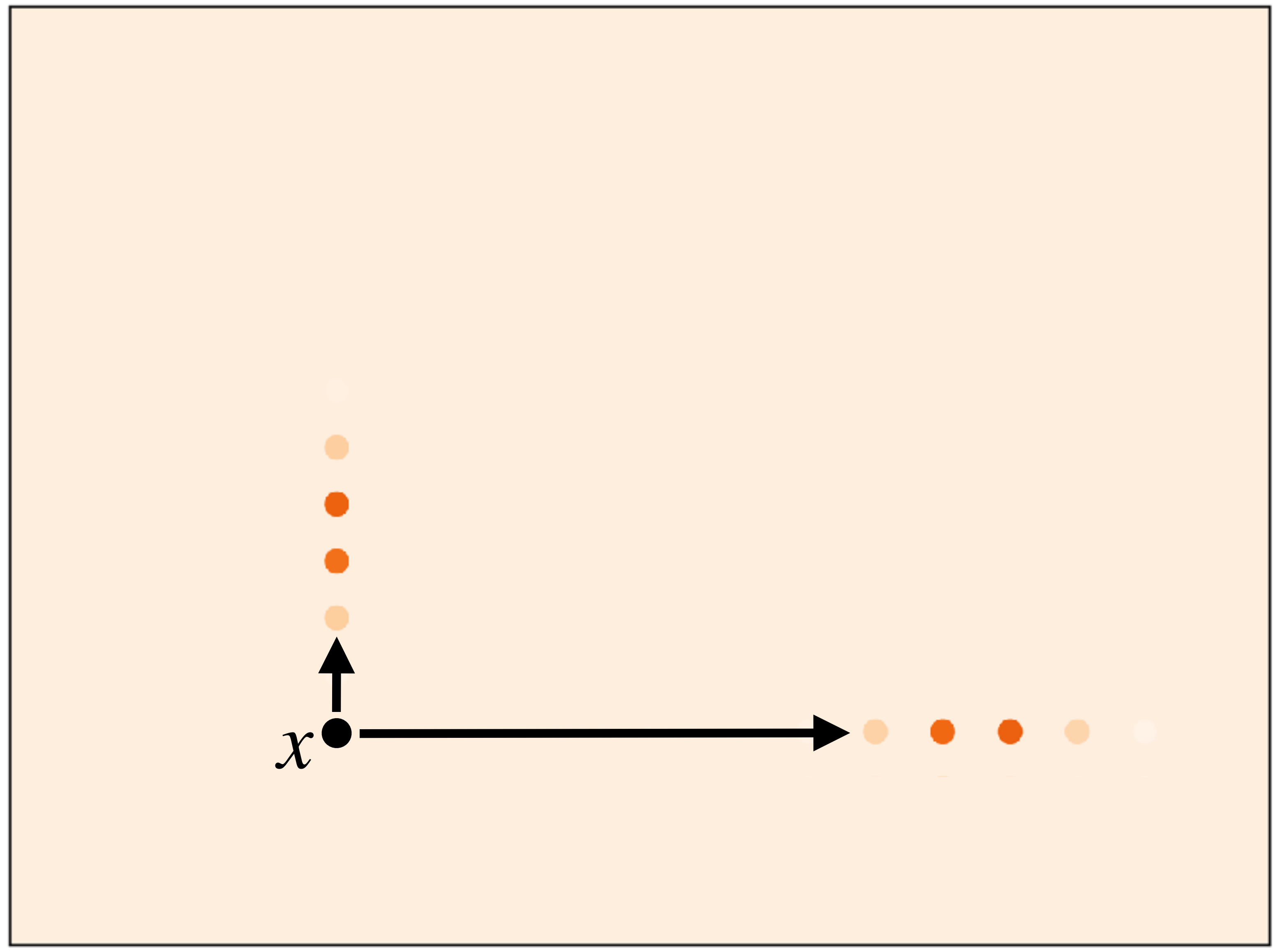
```
def fib(n: int):
    """Return n-th Fibonacci
    number.
    >>> fib(10)
    55
    >>> fib(1)
    1
    >>> fib(8)
    21
    """
    if n < 1: return 0
    if n < 2: return 1

    return fib(n-1) + fib(n-2)
```

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



Intractable

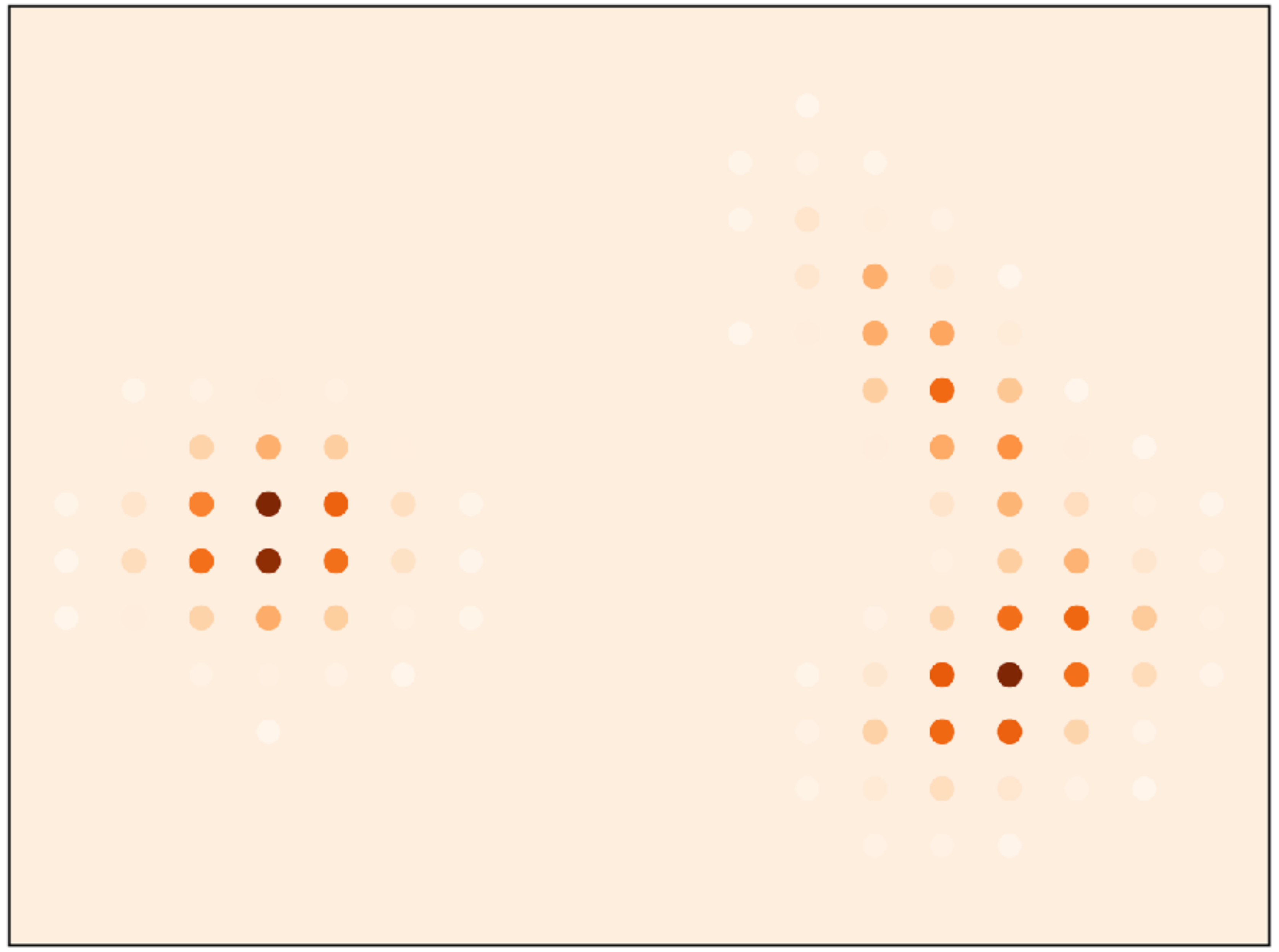
$u_t(\cdot \,|\, x) \in \mathbb{R}^{d|\mathcal{T}|}$

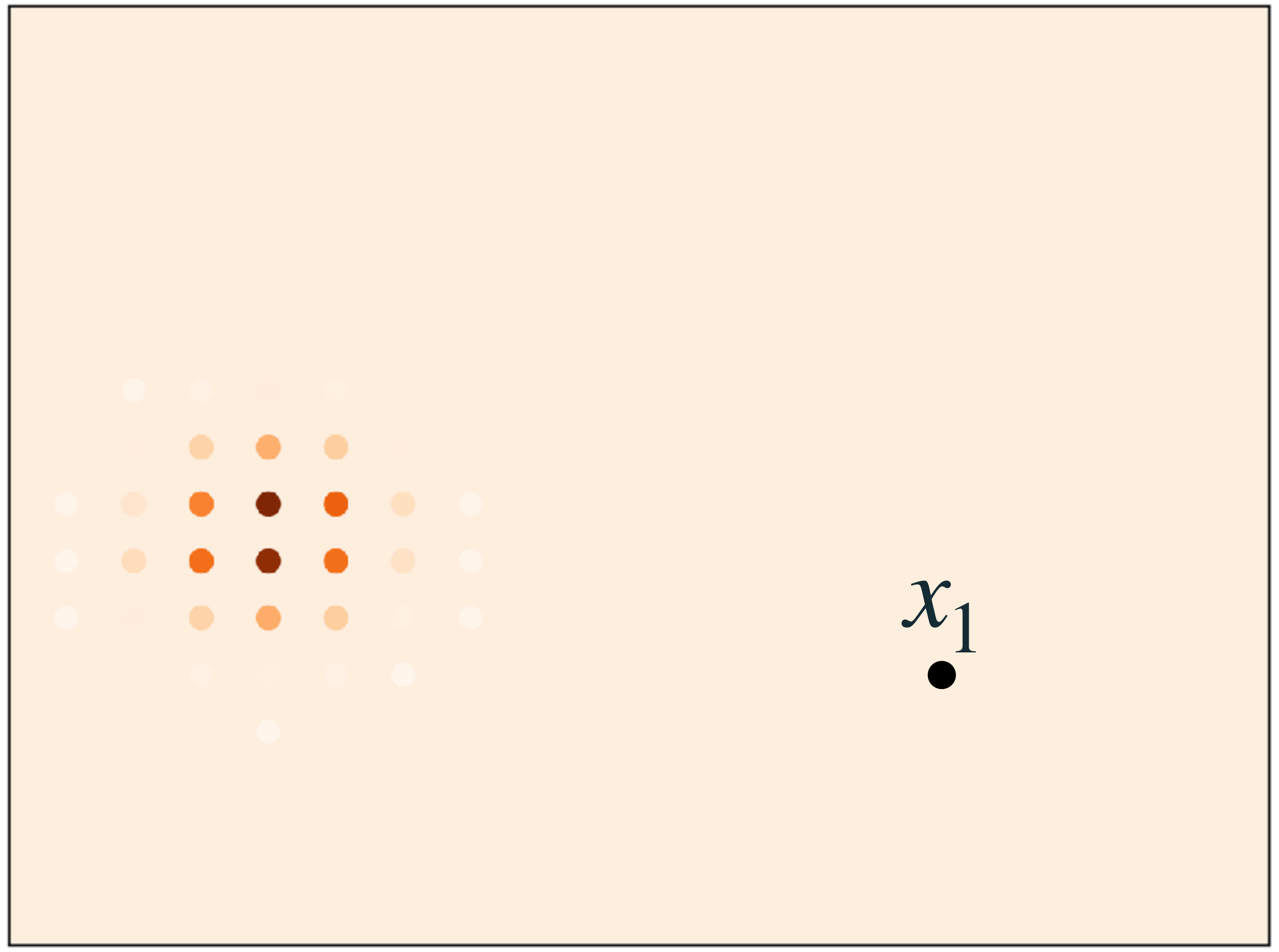


$$u_t^i(y^i \,|\, x)$$

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

Build (factorized) velocities





Mixture path

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

average

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

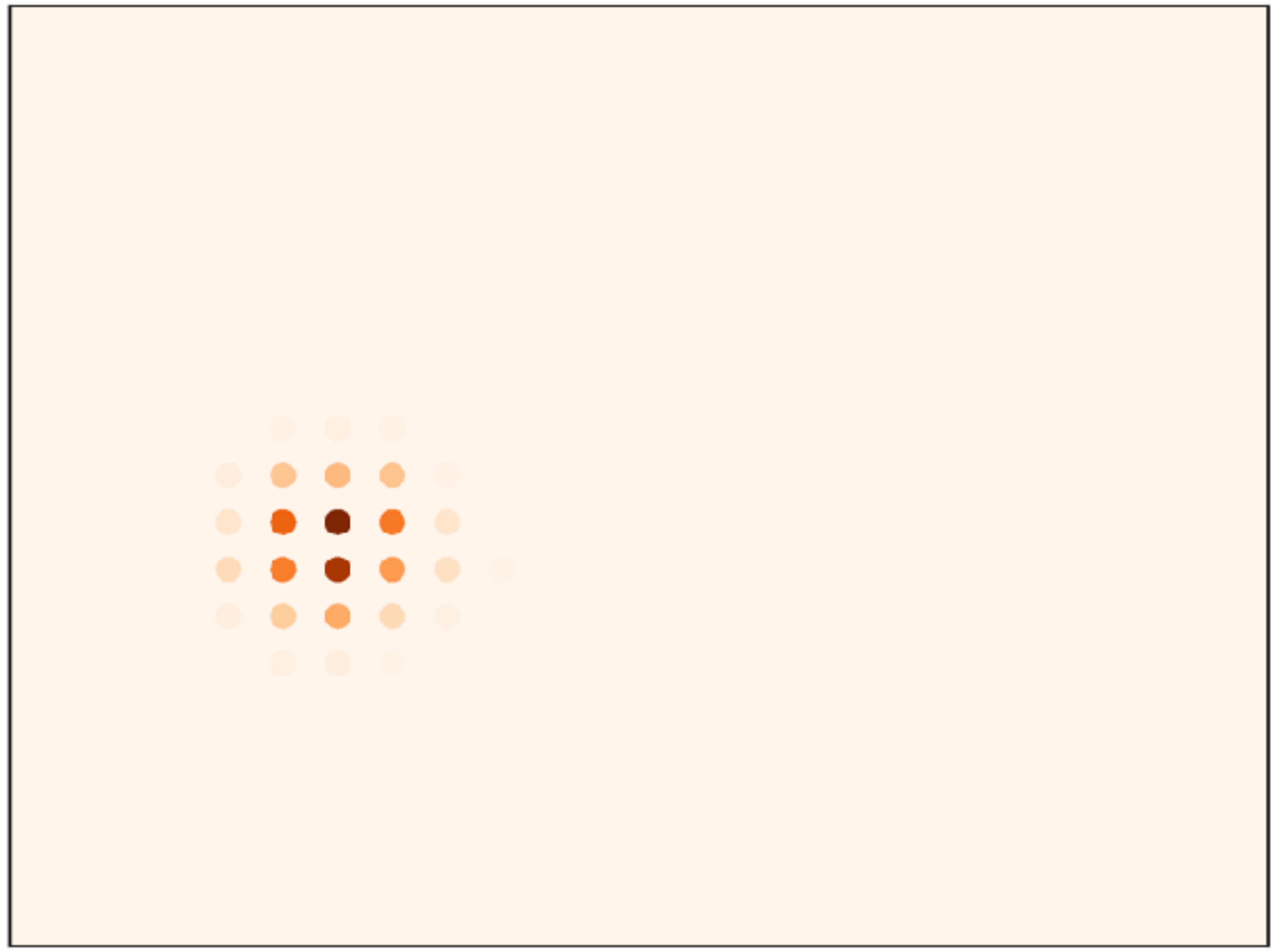
jump rate

where to jump

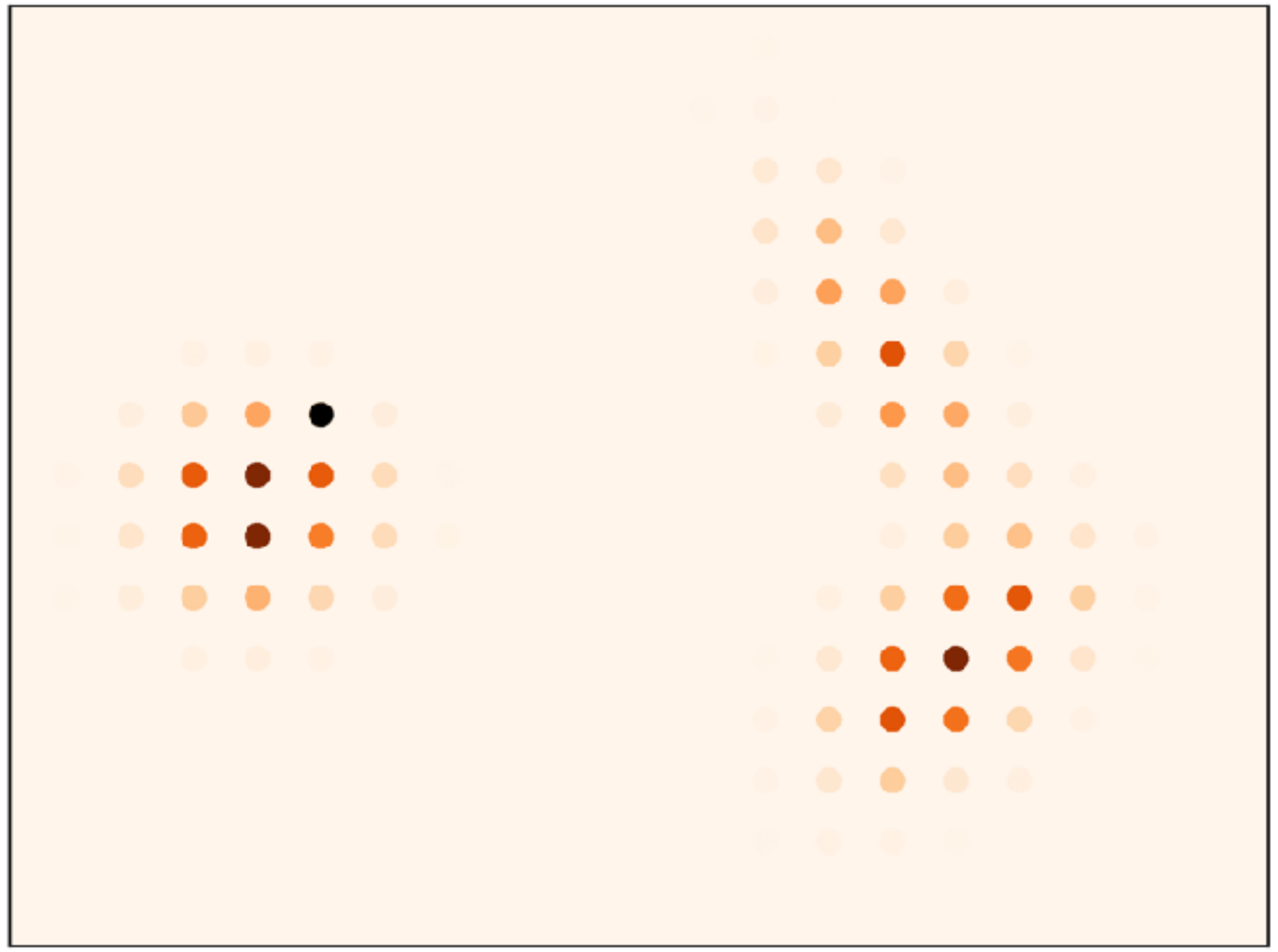
“Generative Flows on Discrete State-Spaces: Enabling Multimodal Flows with Applications to Protein Co-Design” Campbell et al. (2024)

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

Discrete Flow Matching



$p_t(x)$



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

"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 mask model (1.7B)

```
def binary_search(arr, x):






        # If x is greater


        # If x is smaller


        else:

```

"Simple and Effective Masked Diffusion Language Models" Sahoo et al. (2024)
"Simplified and Generalized Masked Diffusion for Discrete Data" Shi et al. (2024)
“Discrete Flow Matching” Gat el al. (2024)

General Discrete Probability Paths

Given any (conditional) probability path $p_t(x)$, a **kinetic optimal** (conditional) velocity is:

$$u_t(y|x) = \frac{\mathrm{ReLU}(p_t(x)\dot{p}_t(y) - \dot{p}_t(x)p_t(y))}{p_t(x)}$$

E.g. Metric paths:

$$p_{t|1}(x|x_1) = \mathrm{softmax}(-\beta_t d(x, x_1))$$

$$u_t(y|x, x_1) = p_{t|1}(x|x_1)\dot{\beta}_t \mathrm{ReLU}(d(x, x_1) - d(y, x_1))$$

CIFAR10

Method	FID ↓
D3PM (Austin et al., 2021)	7.34
CTDD (Nisonoff et al., 2024)	7.86
τLDR-10 (Campbell et al., 2022)	3.74
DFM w/ mask (Gat et al., 2024)	3.63
DFM w/ metric (Ours)	**3.43**

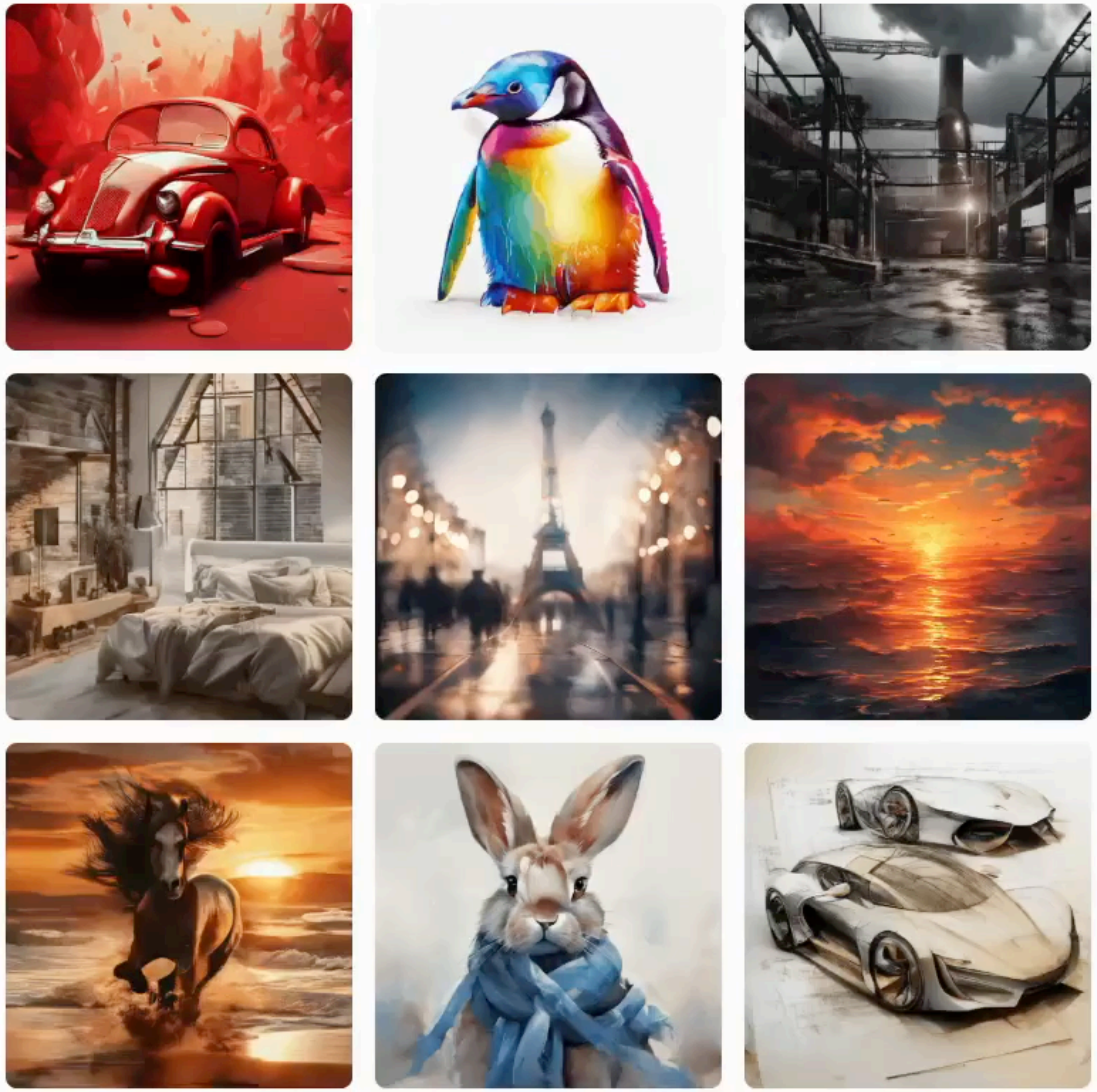
ImageNet 256x256

Method	NFE	FID
LlamaGen (AR) (Sun et al., 2024)	256	5.46
LlamaGen (AR)* (Sun et al., 2024)	256	4.81
DFM - Mask* (Gat et al., 2024)	100	5.72
DFM - Metric (Ours)	100	**4.50**

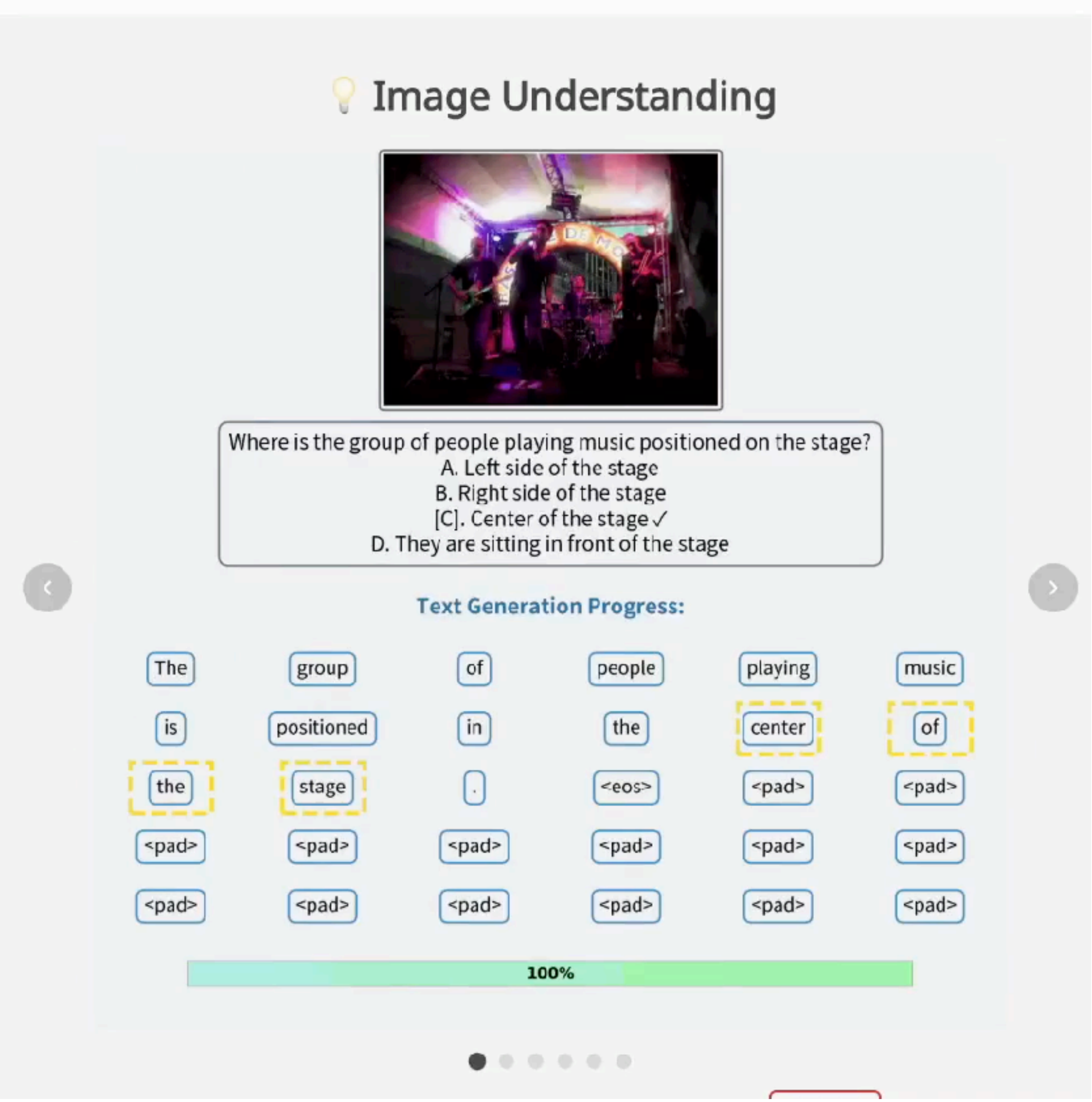
"Flow Matching with General Discrete Paths: A Kinetic-Optimal Perspective" Shaul el al. (2024)

FUDOKI: Multimodal w/ General Discrete Paths

Image Generation



Discrete flow matching denoises step by step, transforming noise into vivid images.



"FUDOKI: Discrete Flow-based Unified Understanding and Generation via Kinetic-Optimal Velocities" Wang et al. (2025)

Edit Flows

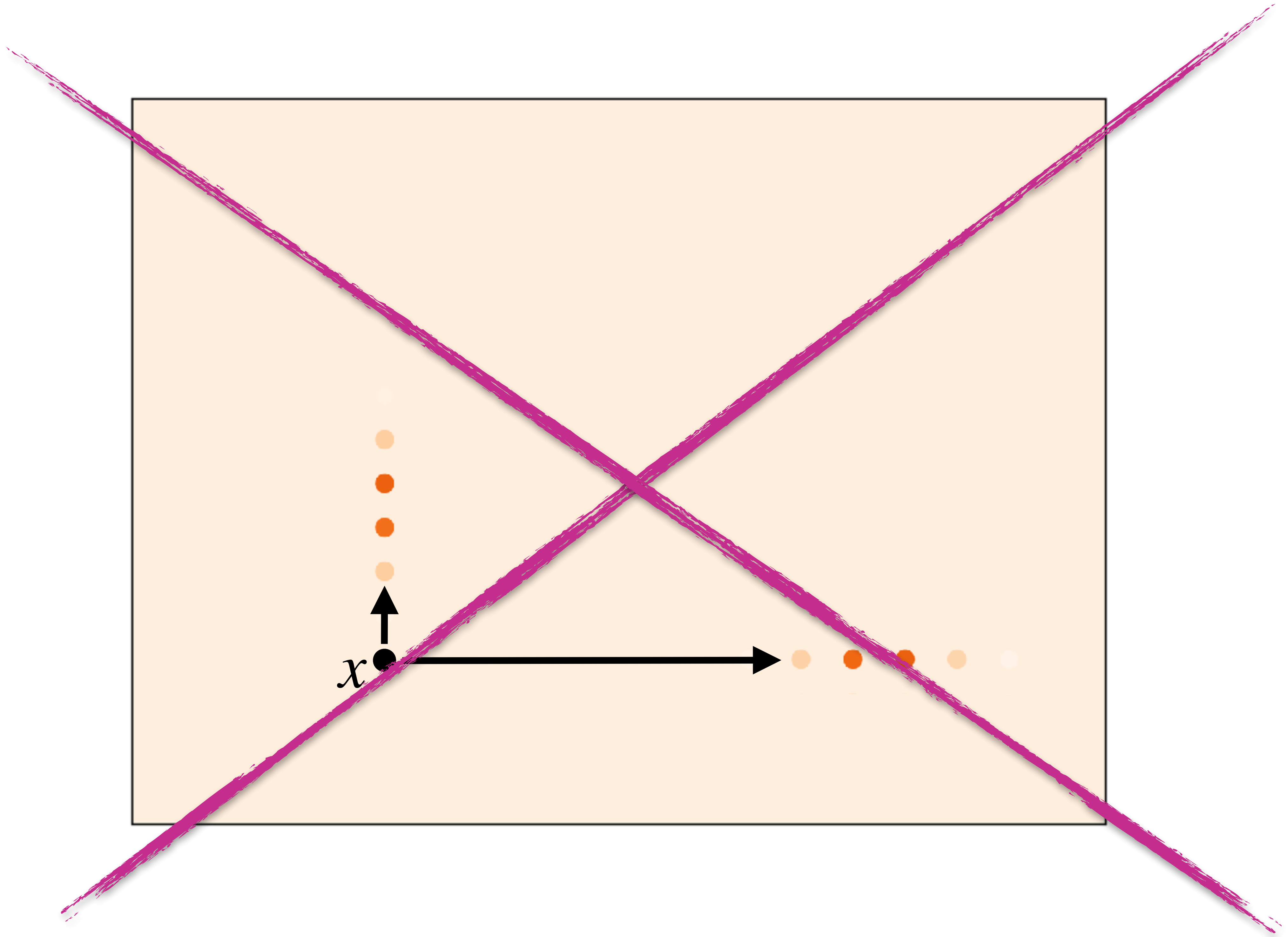
Imbuing diffusion with dimensional changes

Designing a model for variable length sequences

State space

$$X \in \bigcup_{n=0}^{N} \mathcal{T}^n$$

Data contains variable length sequences



Forcefully aligning tokens doesn’t make sense for sequence generation

- Requires padding with <EOS> to handle variable length generation
- Makes the model over-confident in predicting <EOS>

Designing a model for variable length sequences

**Insertion Transformer:
Flexible Sequence Generation via Insertion Operations**

Mitchell Stern [1,2] **William Chan** [1] **Jamie Kiros** [1] **Jakob Uszkoreit** [1]

State space

$$X \in \bigcup_{n=0}^{N} \mathscr{T}^n$$

Data contains variable length sequences

Levenshtein Transformer

Jiatao Gu[†], **Changhan Wang**[†], **and Jake Zhao (Junbo)**[‡◇]
[†]Facebook AI Research
[‡]New York University [◇]Tigerobo Inc.
[†]{jgu, changhan}@fb.com [‡]jakezhao@cs.nyu.edu

Preprint

DIFFUSER: DISCRETE DIFFUSION VIA EDIT-BASED RECONSTRUCTION

Machel Reid
Google Research*
machelreid@google.com

Vincent J. Hellendoorn
Software and Societal Systems Department
Carnegie Mellon University
vhellendoorn@cmu.edu

Graham Neubig
Language Technologies Institute,
Carnegie Mellon University
Inspired Cognition
gneubig@cs.cmu.edu

VERY rough description of some prior works:

- Train on an (ordered) sequence of edit operations
- 2-stage model to predict # edits, use causal mask model to predict tokens

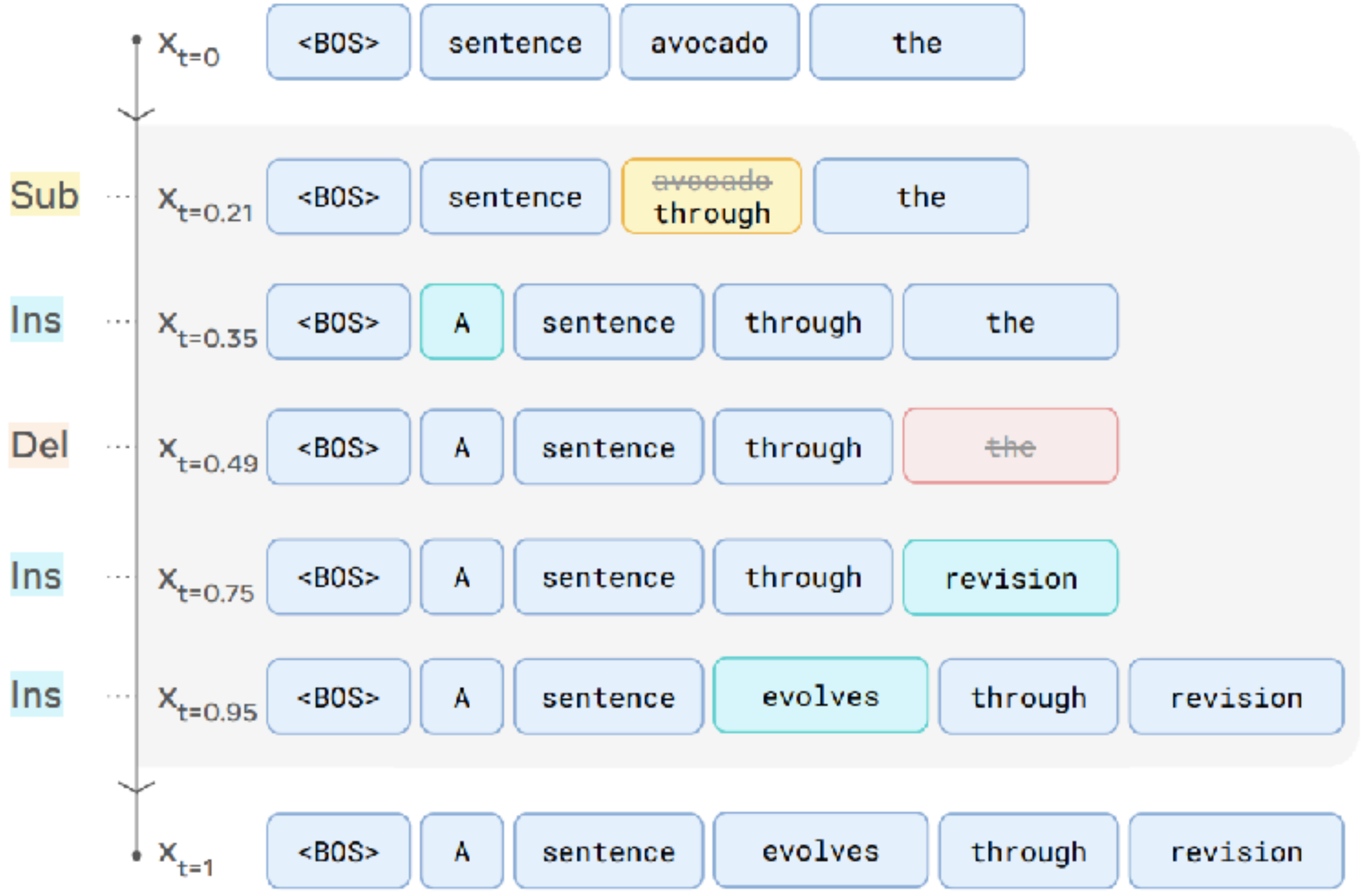
Designing a model for variable length sequences

State space

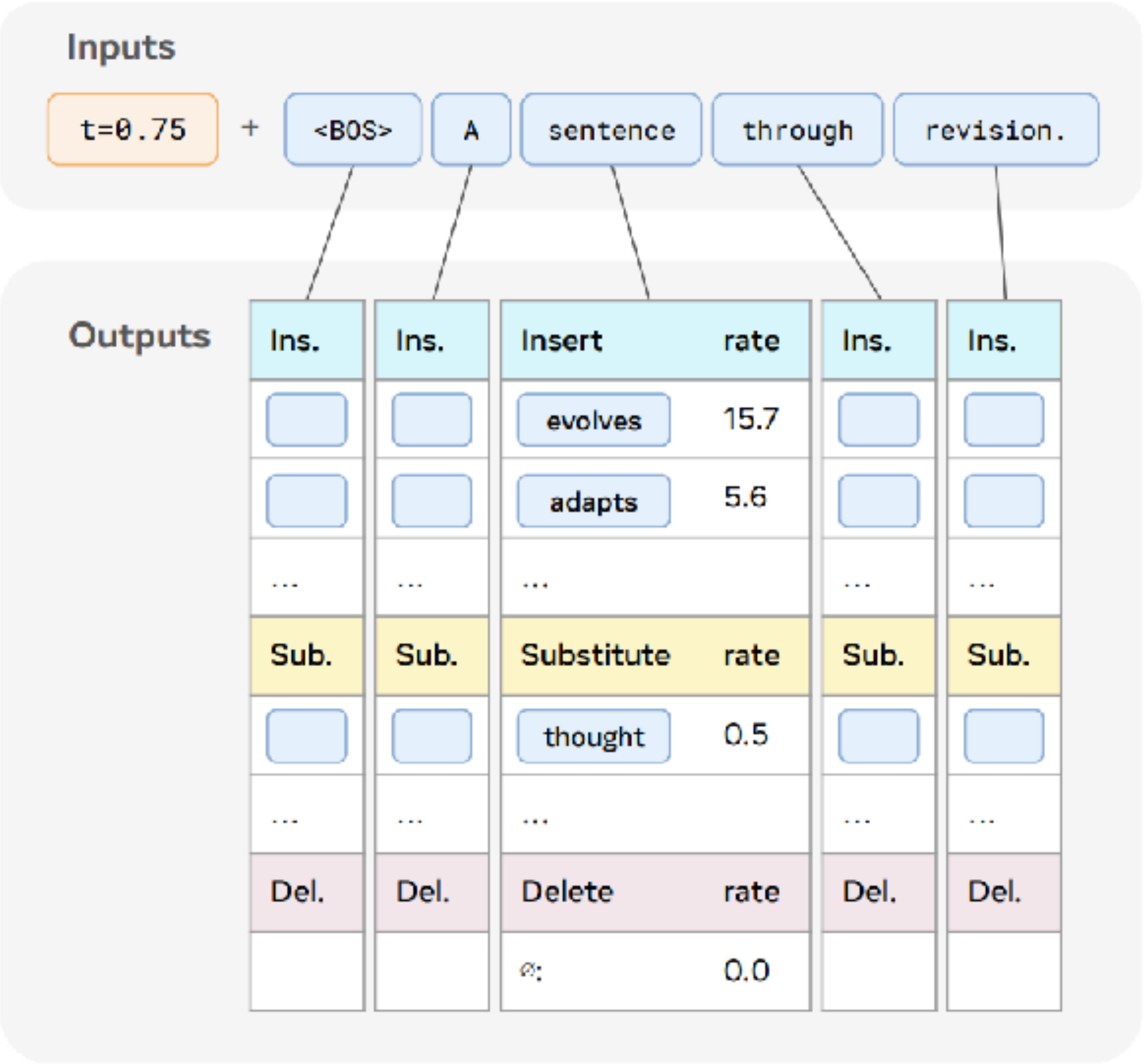
$$X \in \bigcup_{n=0}^{N} \mathcal{T}^n$$

Data contains variable length sequences

Sampling process



Inputs / Outputs



CTMC Model

$$u_t^\theta(\text{ins}(x,i,a)|x) = \lambda_{t,i}^{\text{ins}}(x) Q_{t,i}^{\text{ins}}(a|x) \qquad \text{for } i \in \{1,\dots,n(x)\}$$

$$u_t^\theta(\text{del}(x,i)|x) = \lambda_{t,i}^{\text{del}}(x) \qquad \text{for } i \in \{1,\dots,n(x)\}$$

$$u_t^\theta(\text{sub}(x,i,a)|x) = \lambda_{t,i}^{\text{sub}}(x) Q_{t,i}^{\text{sub}}(a|x) \qquad \text{for } i \in \{1,\dots,n(x)\}$$

Define unique set of edits via alignments

Multiple ways to map X_t to X_1 → Intractable to construct $u_t(x \mid X_t, X_1)$

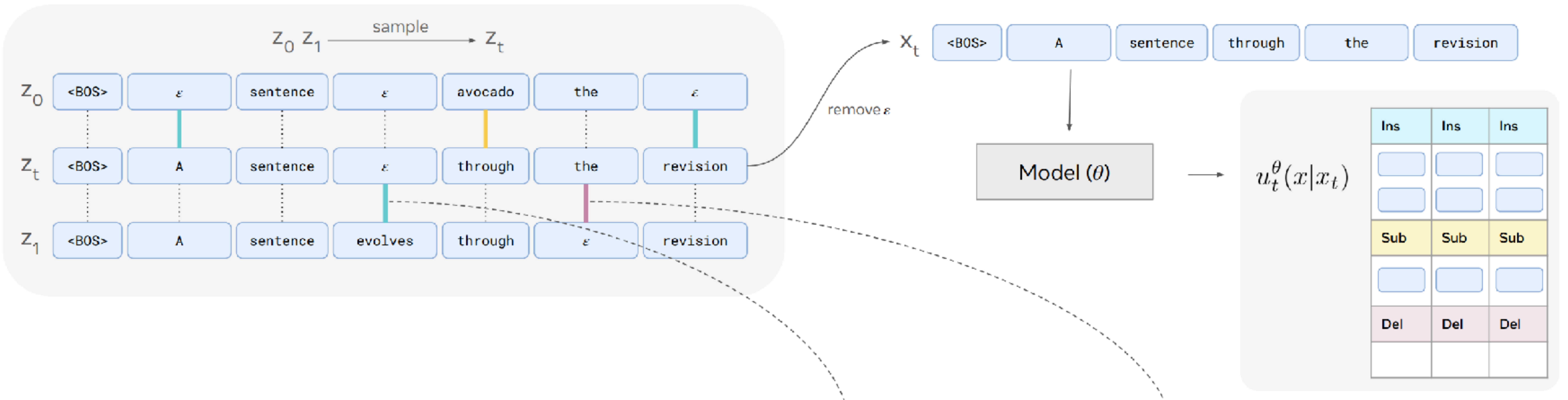
Alignments

$$Z \in \mathcal{T}^N$$

Fixed-length (padded) sequences

(Z_0, Z_1) defines a **unique set of edit operations** to map X_0 to X_1

Discrete Flow Matching with auxiliary variables

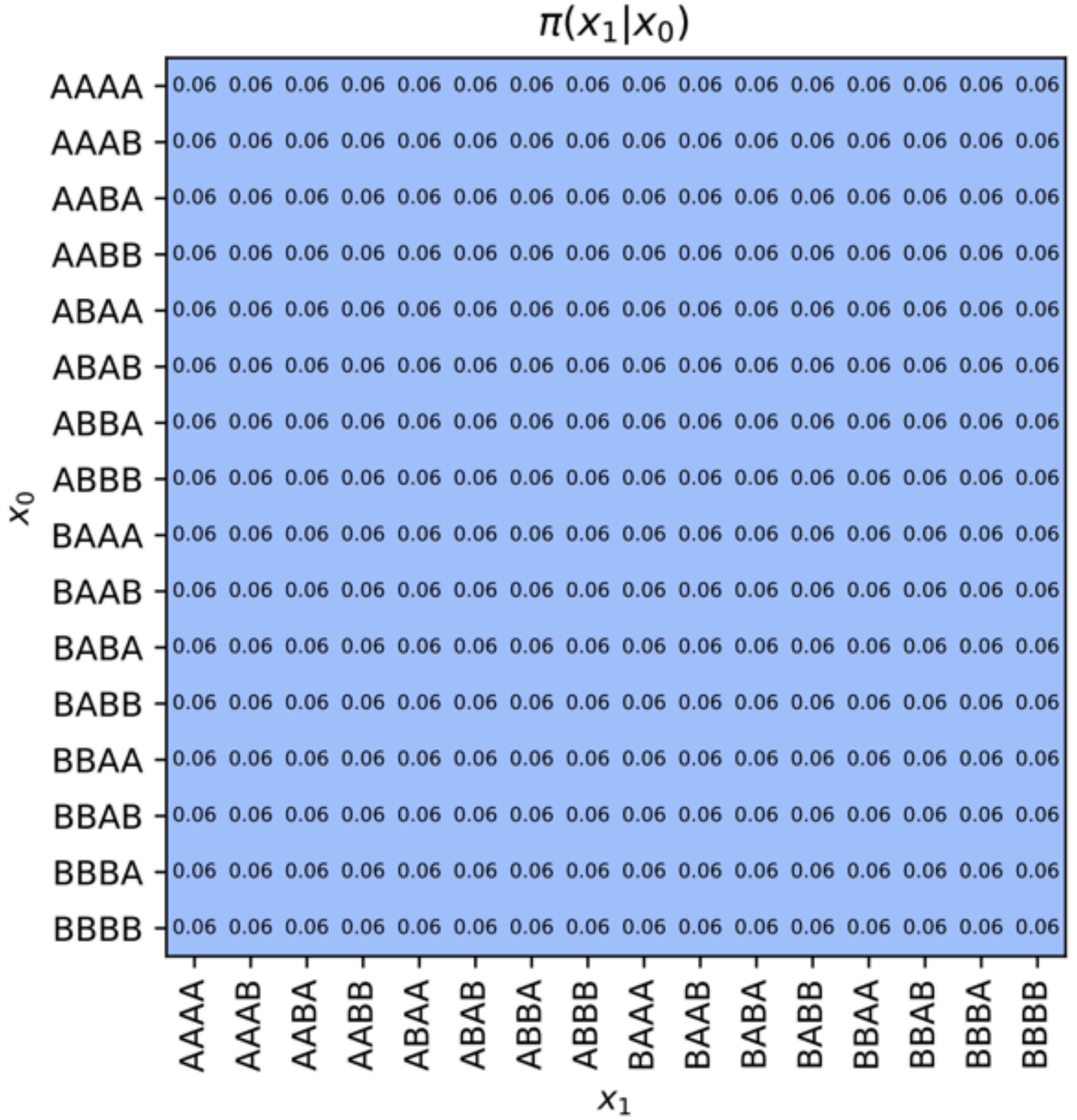


$$\mathcal{L}(\theta) \overset{\text{MC}}{\approx} \sum_{x \neq x_t} u_t^\theta(x|x_t) - \frac{\dot{\kappa}_t}{1-\kappa_t}\left[\log u_t^\theta(\text{Ins}\ \boxed{\text{evolves}}\ |x_t) + \log u_t^\theta(\text{Del}\ \boxed{\text{the}}\ |x_t)\right]$$

Inherits similar optimality as continuous space

Learned model will try to **minimize the number of edits**! Maps nearby (X_0, X_1).

Training (X_0, X_1) pairs



Generated (X_0, X_1) pairs

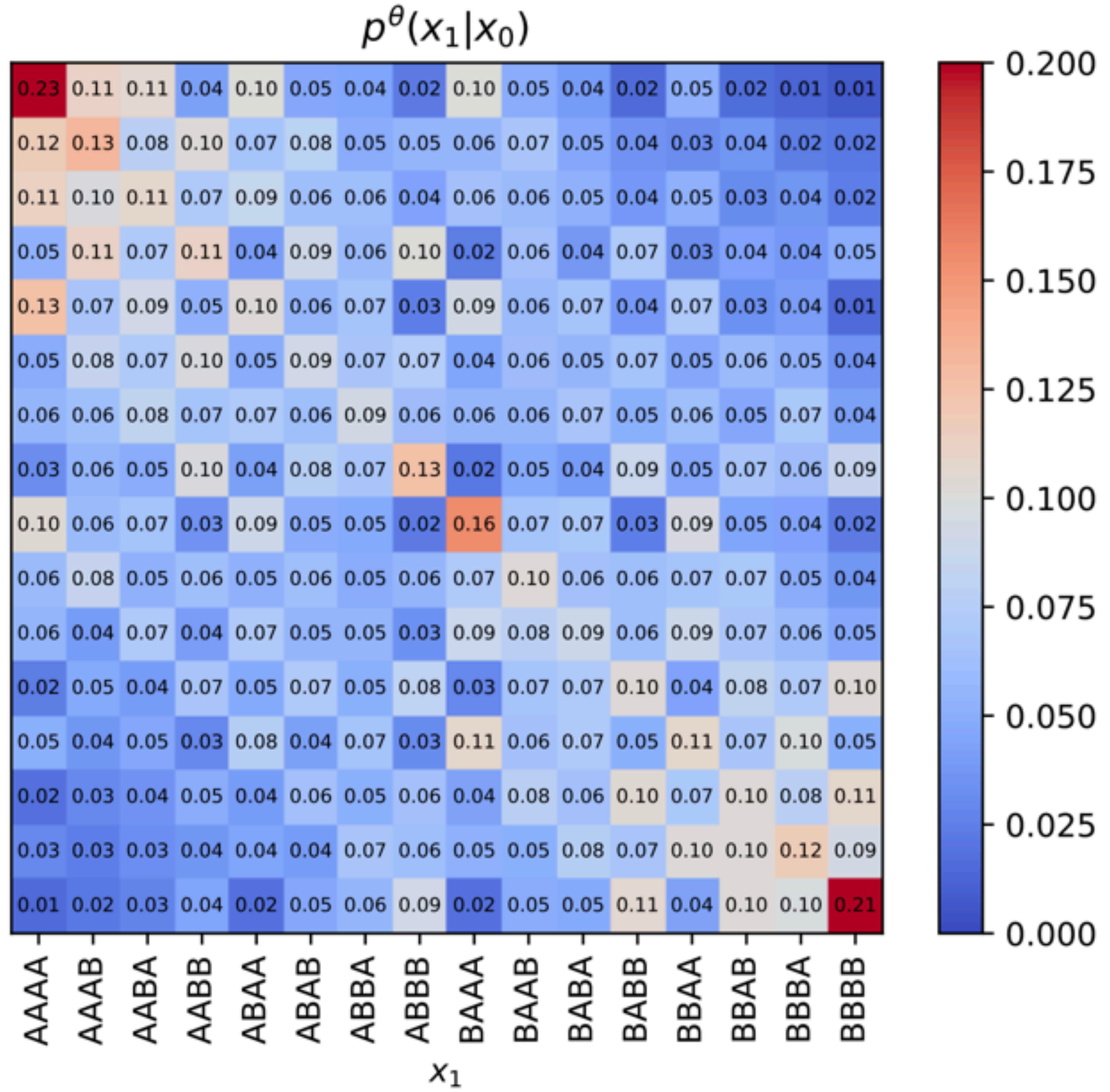


Image Captioning

Method	MS COCO			Image Captioning 3M		
	METEOR	CIDEr	SPICE	ROUGE-L	CIDEr	SPICE
VLP† (Zhou et al., 2020)	28.4	117.7	21.3	24.3	77.5	16.5
ClipCap† (Mokady et al., 2021)	27.1	108.3	20.1	26.7	87.2	18.5
Autoregressive	25.7	95.5	19.6	25.2	85.8	17.8
Mask DFM	25.3	95.6	19.2	27.4	96.2	20.3
Edit Flow (**Ours**)	27.4	108.1	21.1	29.0	101.9	21.7
Localized Edit Flow (**Ours**)	27.4	105.1	22.1	28.3	99.7	20.8

floor

tree

camera

Text & Code Generation (1.3B)

Insert-only

```
def is_prime(n: int) -> bool:
```

Insert + Delete + Substitute

```
def is_prime(n: int) -> bool:
    if n <= 1:
        return True
    for i in range(2, n):
        if i % n == 0:
            return True
    return False
```

Method	HellaSwag	ARC-E	ARC-C	PIQA	OBQA	WinoGrande
Autoregressive	49.5	71.0	36.3	76.0	30.4	62.1
Mask DFM	38.3	55.4	27.8	65.3	22.6	52.3
Edit Flow (CFG applied to u_t) (**Ours**)	49.0	63.1	33.0	68.8	28.6	53.6
Edit Flow (CFG applied to $\mathcal{L}$) (**Ours**)	54.5	61.0	34.0	65.0	37.2	54.3

Table 2 Zero-shot text benchmarks using 1.3B parameter models trained on DCLM-baseline 1.0 (Li et al., 2024). Colors show the best and second best among each metric.

	Method	HumanEval		HumanEval+		MBPP	
		Pass@1	Pass@10	Pass@1	Pass@10	Pass@1	Pass@10
	Autoregressive (Gat et al., 2024)	14.3	21.3			17.0	34.3
	Autoregressive†	17.0	34.7	14.0	28.6	25.6	45.4
Non-AR	Mask DFM (Gat et al., 2024)	6.7	13.4			6.7	20.6
Non-AR	Mask DFM (Oracle Length) (Gat et al., 2024)	11.6	18.3			13.1	28.4
Non-AR	Mask DFM†	9.1	17.6	7.9	13.4	6.2	25.0
Non-AR	Uniform X_0 + Edit Flow (**Ours**)	9.7	24.3	9.7	19.5	9.4	33.4
Non-AR	Edit Flow (**Ours**)	12.8	24.3	10.4	20.7	10.0	36.4
Non-AR	Localized Edit Flow (**Ours**)	14.0	22.6	10.4	18.9	14.8	34.0

Table 3 Code generation benchmarks using 1.3B parameter models trained on the CodeLlama (Roziere et al., 2023) datamix. †Superscript denotes our own implementation. We highlight the best non-autoregressive models, where colors show the best and second best among each metric.

Summary: Jumps with the Flow Matching recipe!

$$X_{t+h} \sim \delta_{X_t}(\cdot) + h\boxed{u_t(\cdot \mid X_t)}$$

Unnormalized distribution

$$u_t(\cdot \mid X_t) = \boxed{\lambda_t(X_t)}\boxed{Q_t(\cdot \mid X_t)}$$

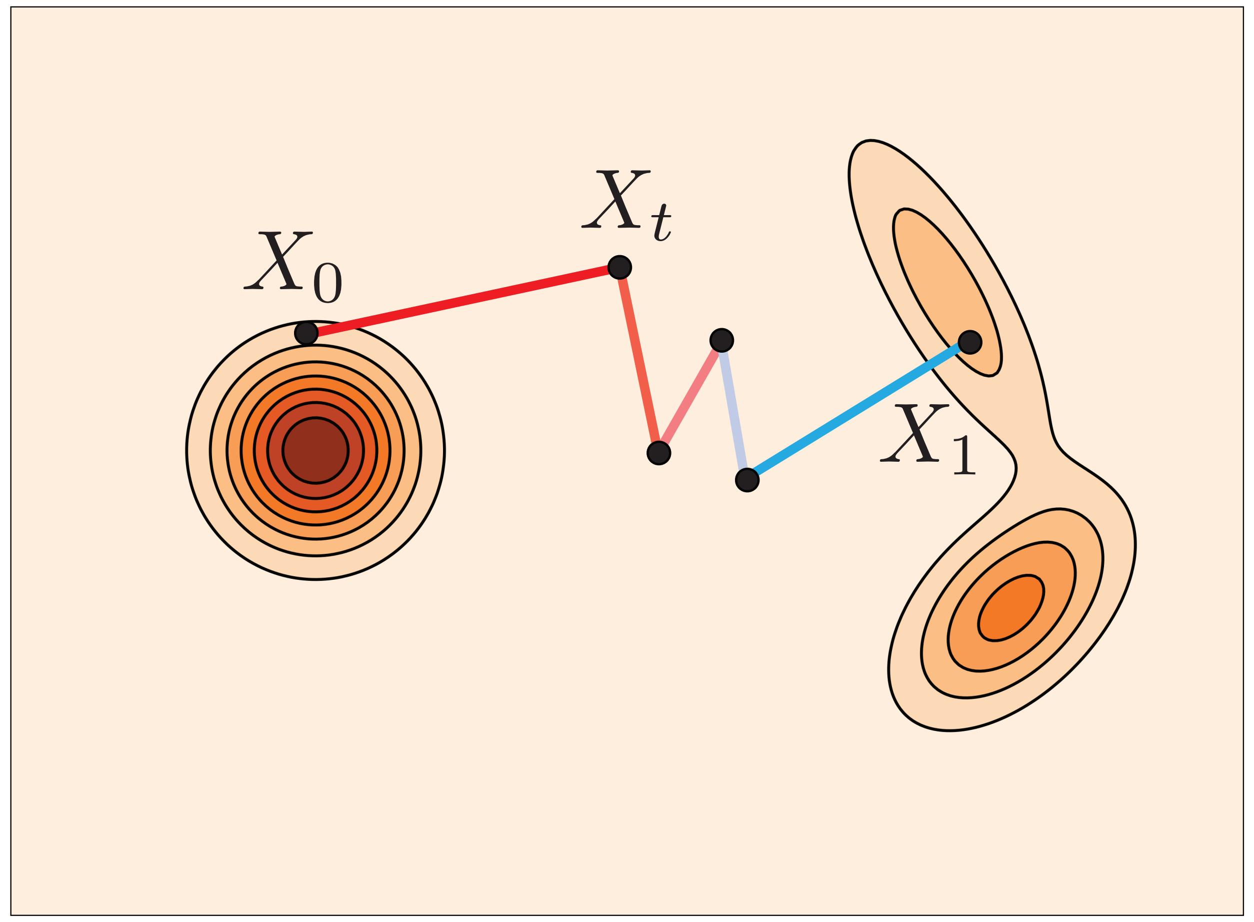
Jump rate

Normalized

Jump process (sampling)

$X_{t+h} \sim Q_t(\cdot \mid X_t)$ if $h\lambda_t(X_t) \leq U[0,1]$

$X_{t+h} = X_t$ otherwise



Learn conditional expectation

$$u_t(\cdot \mid x) = \mathbb{E}\left[u_t(\cdot \mid X_t, X_1) \mid X_t = x\right]$$

Collaborators


Marton Havasi


Itai Gat


Peter Holderrieth


Neta Shaul


Brian Karrer


Yaron Lipman


Tal Remez


Felix Kreuk


Gabriel Synnaeve


Daniel Severo


Anuroop Sriram


Ricky Chen