# Flow Models with Applications to Cell Trajectories and Protein Design

Alex Tong

March 21, 2024

# Stable Diffusion 3

“Stable Diffusion 3 combines a diffusion transformer architecture and flow matching.” – March 5, 2024

## Scaling Rectified Flow Transformers for High-Resolution Image Synthesis

Patrick Esser * Sumith Kulal Andreas Blattmann Rahim Entezari Jonas Müller Harry Saini Yam Levi
Dominik Lorenz Axel Sauer Frederic Boesel Dustin Podell Tim Dockhorn Zion English
Kyle Lacey Alex Goodwin Yannik Marek Robin Rombach *
Stability AI

# Useful for a range of applications

- **Image generation**
- **Cell trajectories**
- **Protein design**
- **Molecule generation**

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

atong01/conditional-flow-matching Public

TorchCFM: a Conditional Flow Matching library

Python ☆ 513 38

Tong et al. TMLR 2024

# Why Flow Matching vs. Score Matching?

More general framework:

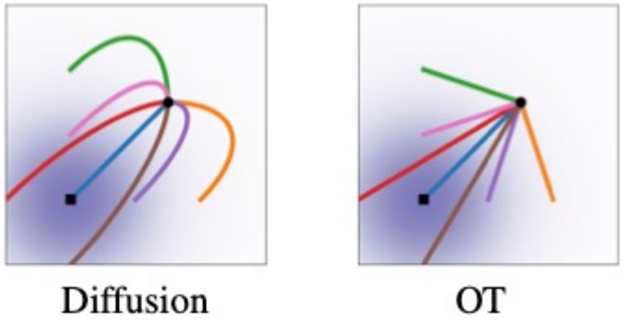
- Reduced variance in the objective via optimal transport leads to **faster training**
- Straighter inference paths via optimal transport leads to **faster inference**
- Flows are **easier to implement** avoiding defining diffusion on manifolds

Score Matching Loss

$$\mathbb{E}_{t,q(z),p_t(x|z)} \|s_\theta(t,x) - \nabla_x \log p_t(x|z)\|_2^2$$

Flow Matching Loss

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




Lipman et al. 2023

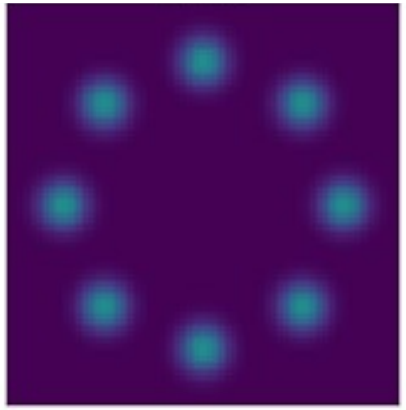
# The Problem

Given samples from a source and target distribution learn a function which **flows** one samples from one distribution to the other.

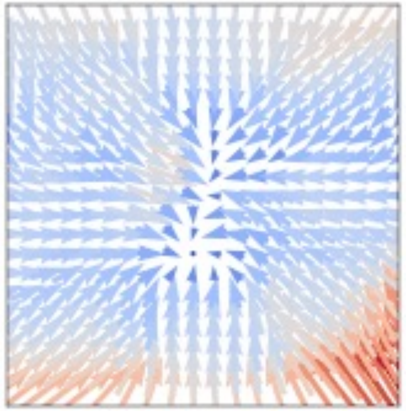
$$U_t(x_0) = x_0 + \int_0^t u_s(x_s)ds$$

$$p_t = [U_t]_\# p_0$$

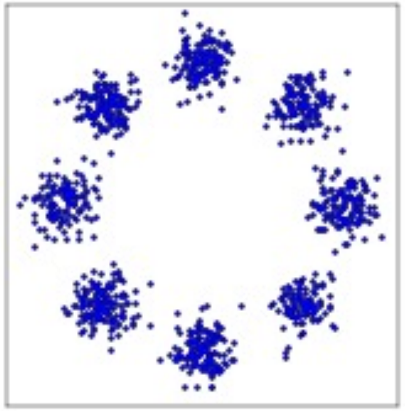
Marginal Probability $p_t(x)$



Marginal field $u_t(x)$



Time dependent flow $U_t(x_0)$

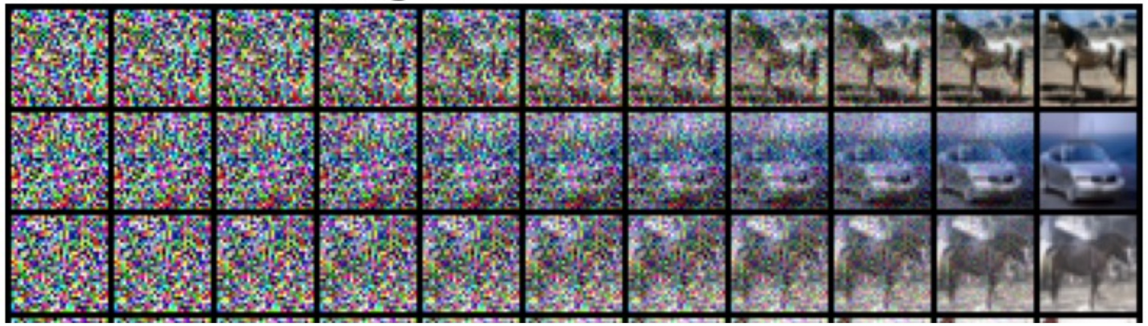


# The Problem

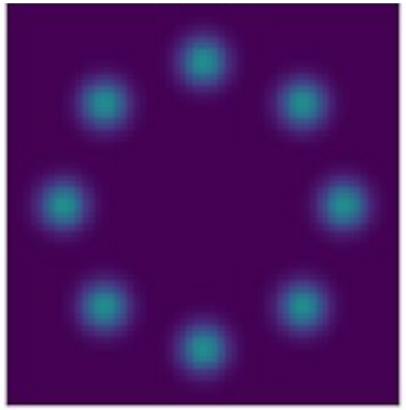
Given samples from a source and target distribution learn a function which **flows** one samples from one distribution to the other.

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

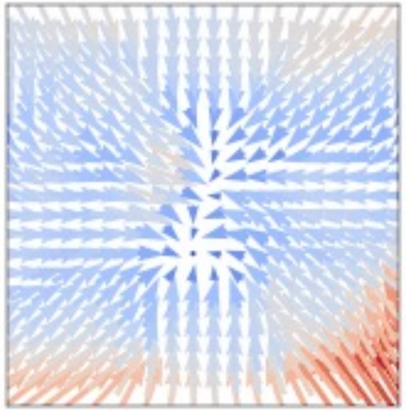
$$p_t = [U_t]_{\#}p_0$$



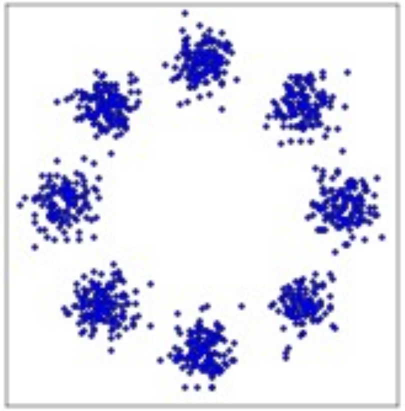
Marginal Probability $p_t(x)$



Marginal field $u_t(x)$



Time dependent flow $U_t(x_0)$

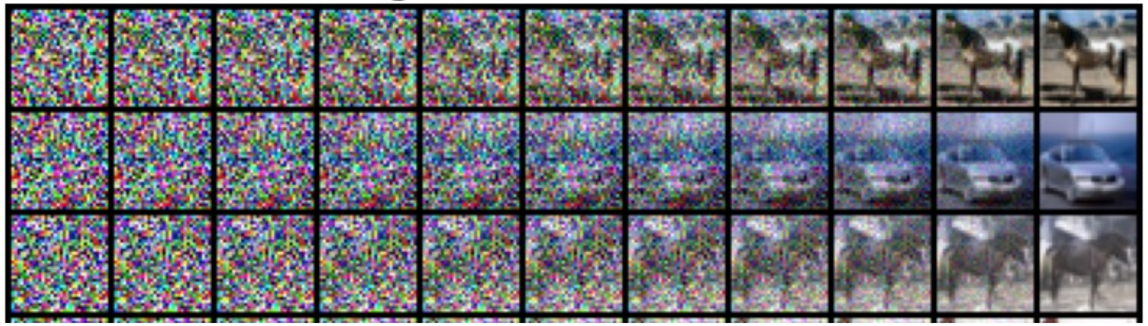


# The Problem

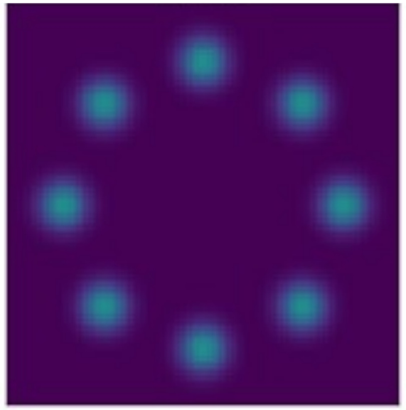
Given samples from a source and target distribution learn a function which **flows** one samples from one distribution to the other.

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

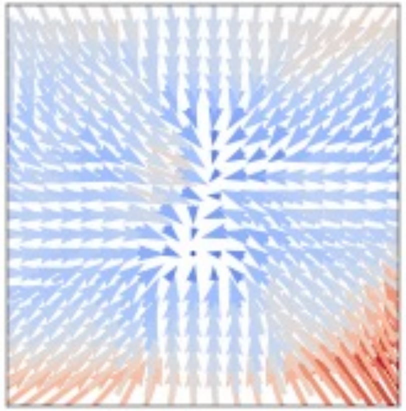
$$p_t = [U_t]_\# p_0$$



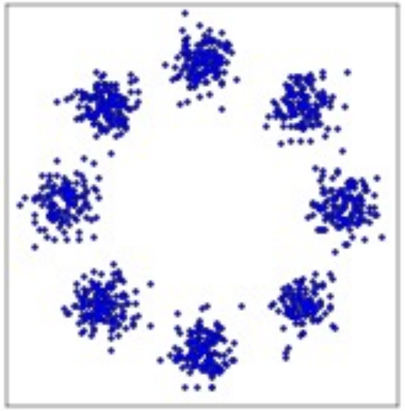
Marginal Probability $p_t(x)$



Marginal field $u_t(x)$



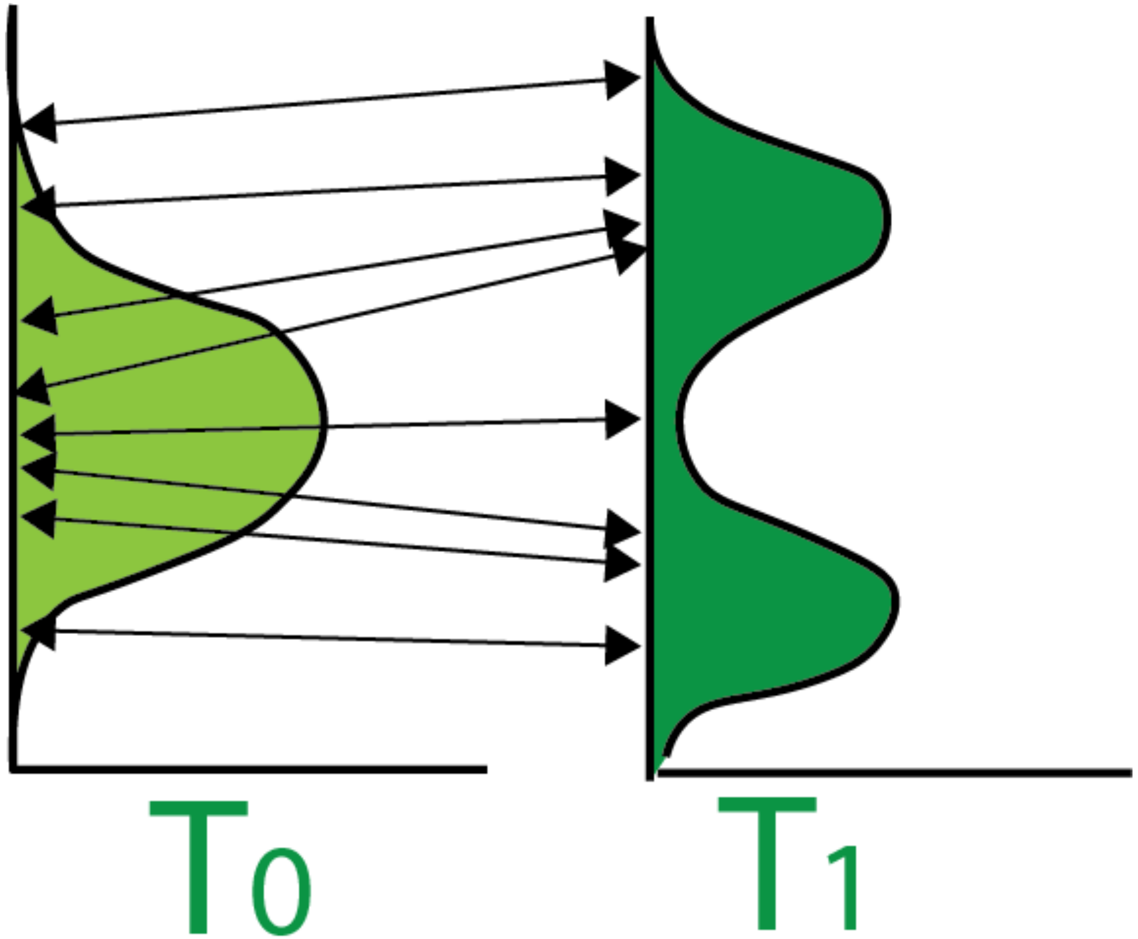
Time dependent flow $U_t(x_0)$



# Normalizing Flows (NFs)

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

- Begin with a simple distribution $p_{t_0}(x) \sim \mathcal{N}(0, 1)$



# Normalizing Flows (NFs)

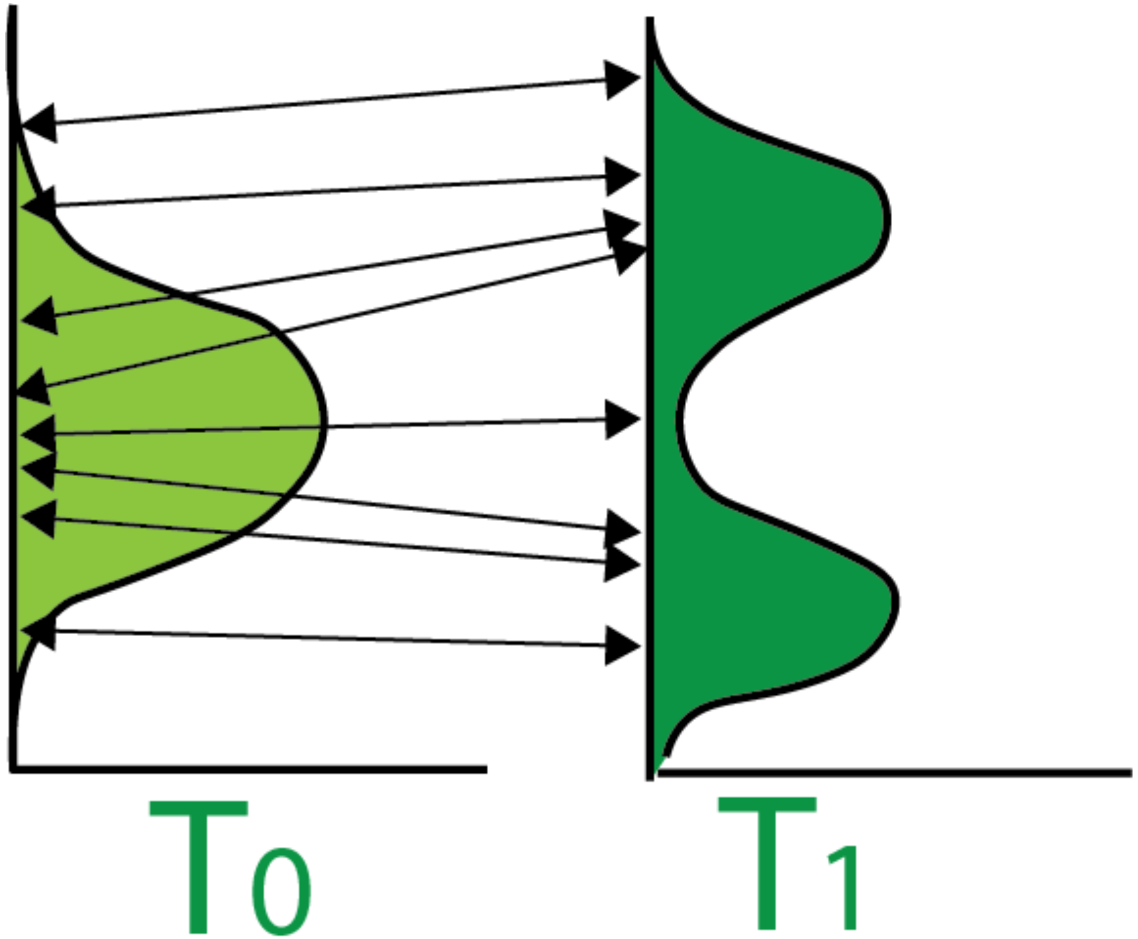
Sample from some complicated distribution by sampling from a simple distribution then applying U

- Begin with a simple distribution

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

- Apply an invertible transformation(s)

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



# Normalizing Flows (NFs)

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

- Begin with a simple distribution

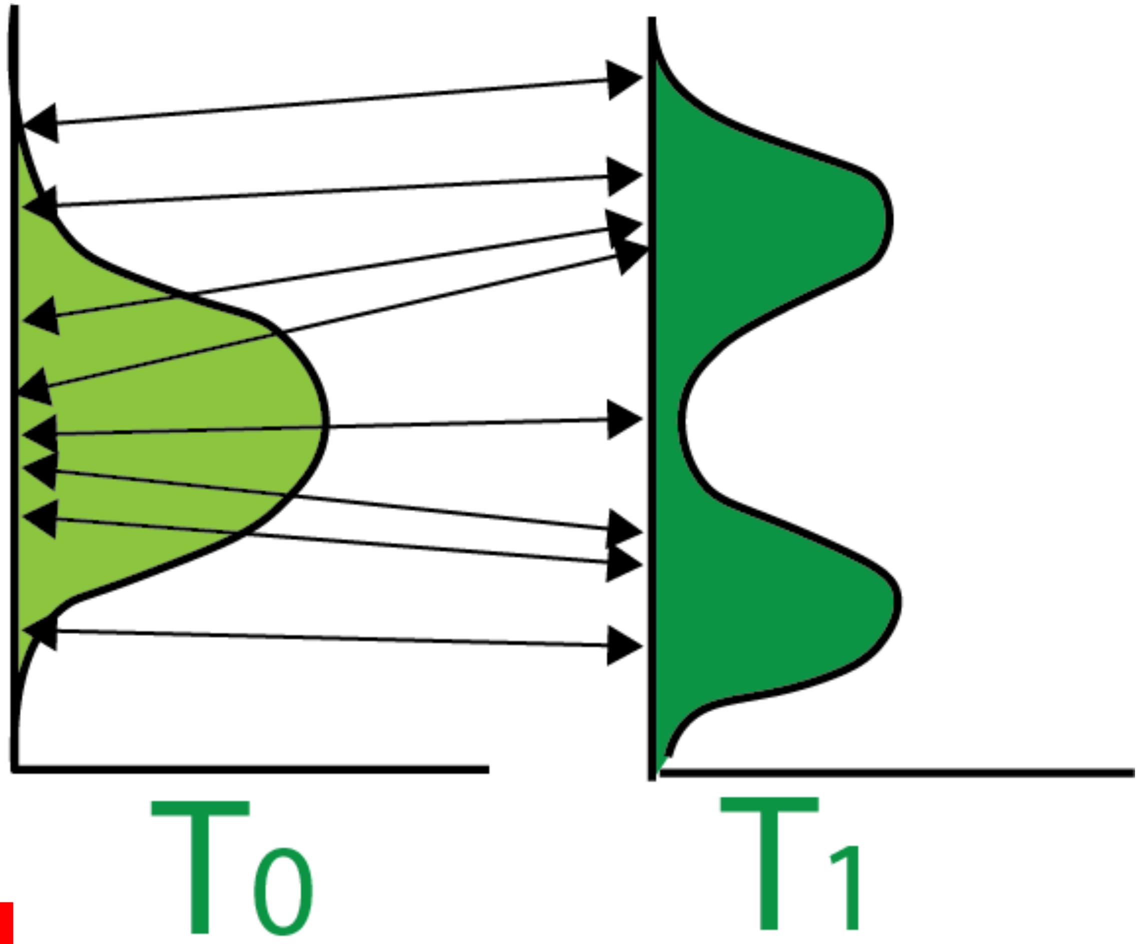
  $p_{t_0}(x) \sim \mathcal{N}(0,1)$

- Apply an invertible transformation(s)

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

- Use change of variables to calculate probability

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



# Deep Normalizing Flows (NFs)

Apply a series of transformations

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

Use change of variables to calculate probability

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

# Continuous Normalizing Flows

Apply a series of transformations

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

Use change of variables to calculate probability

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

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

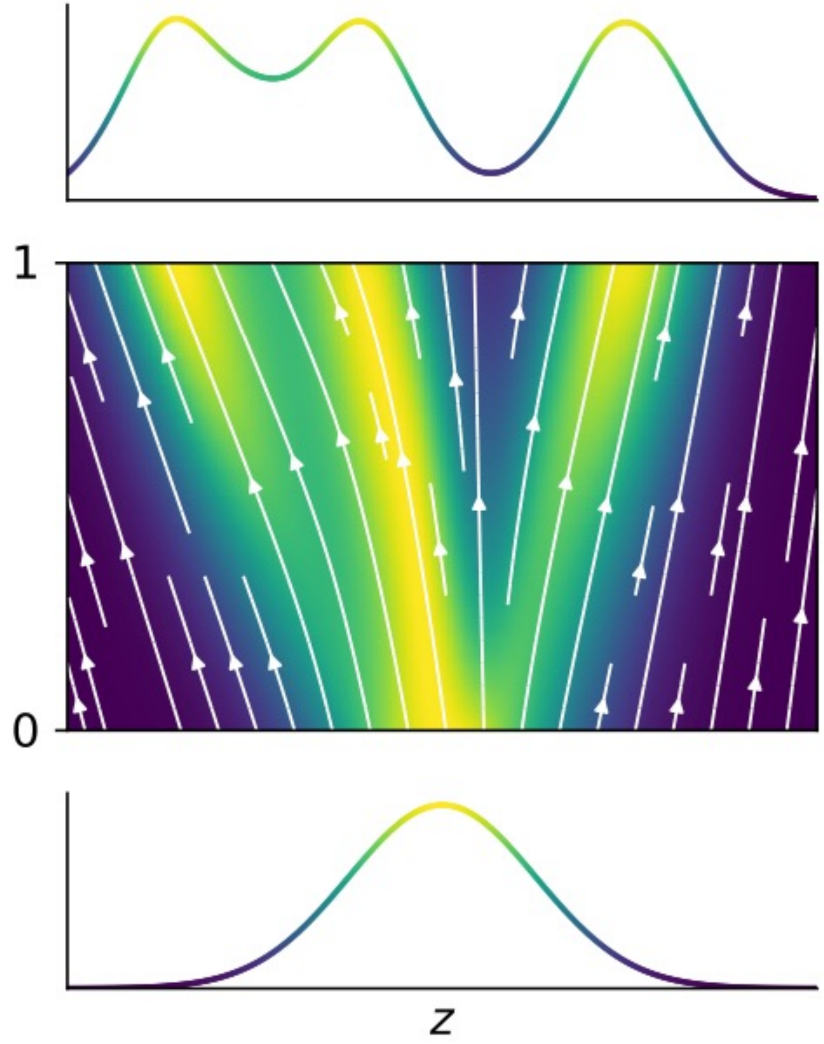
# Continuous Normalizing Flows

A ons

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

U lculate probability

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



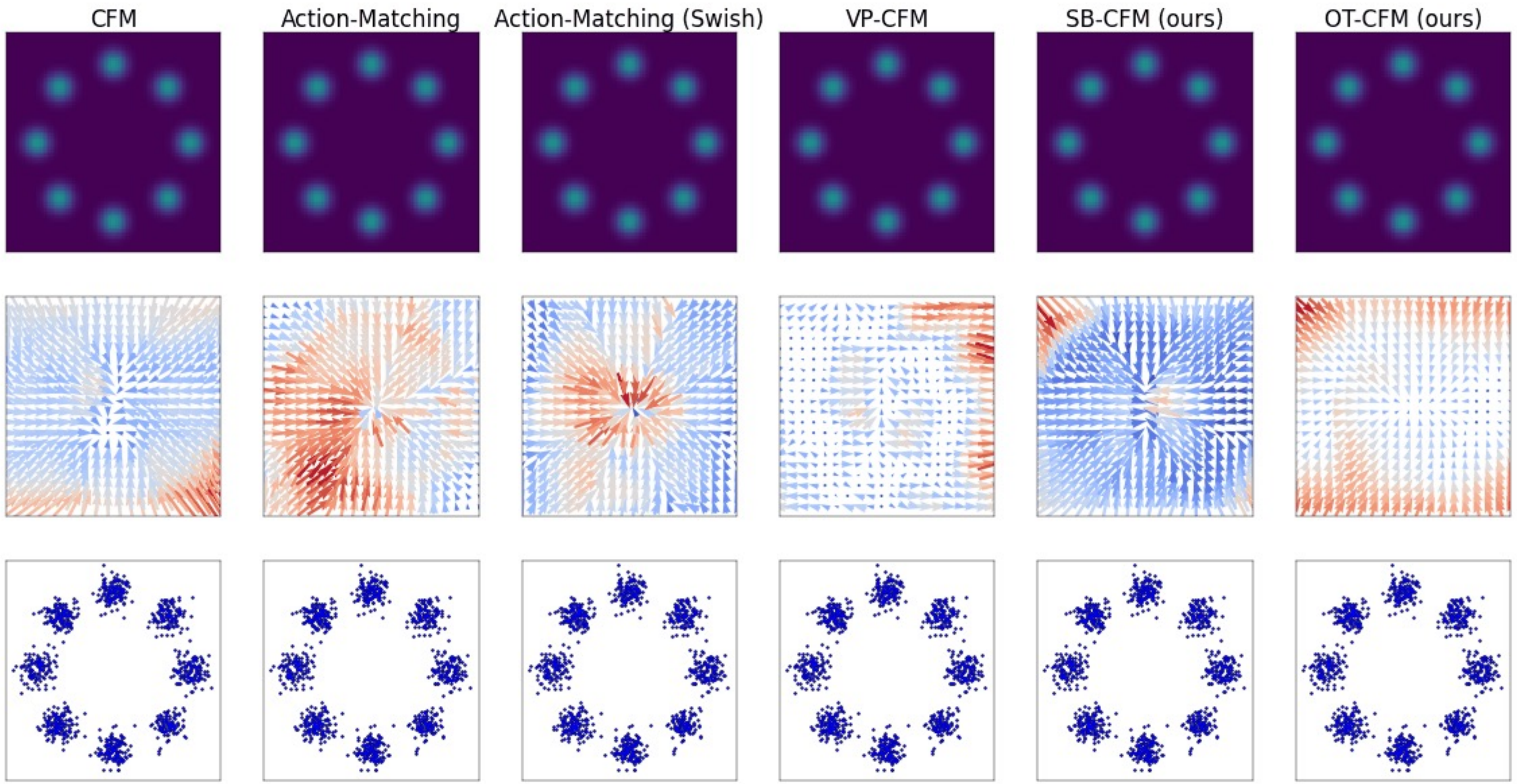

[Chen et al. 2018; Grathwohl et al. 2018]

# If we knew $u_t(x)$, $p_t(x)$ we could directly regress

Marginal Probability $p_t(x)$

Marginal field $u_t(x)$

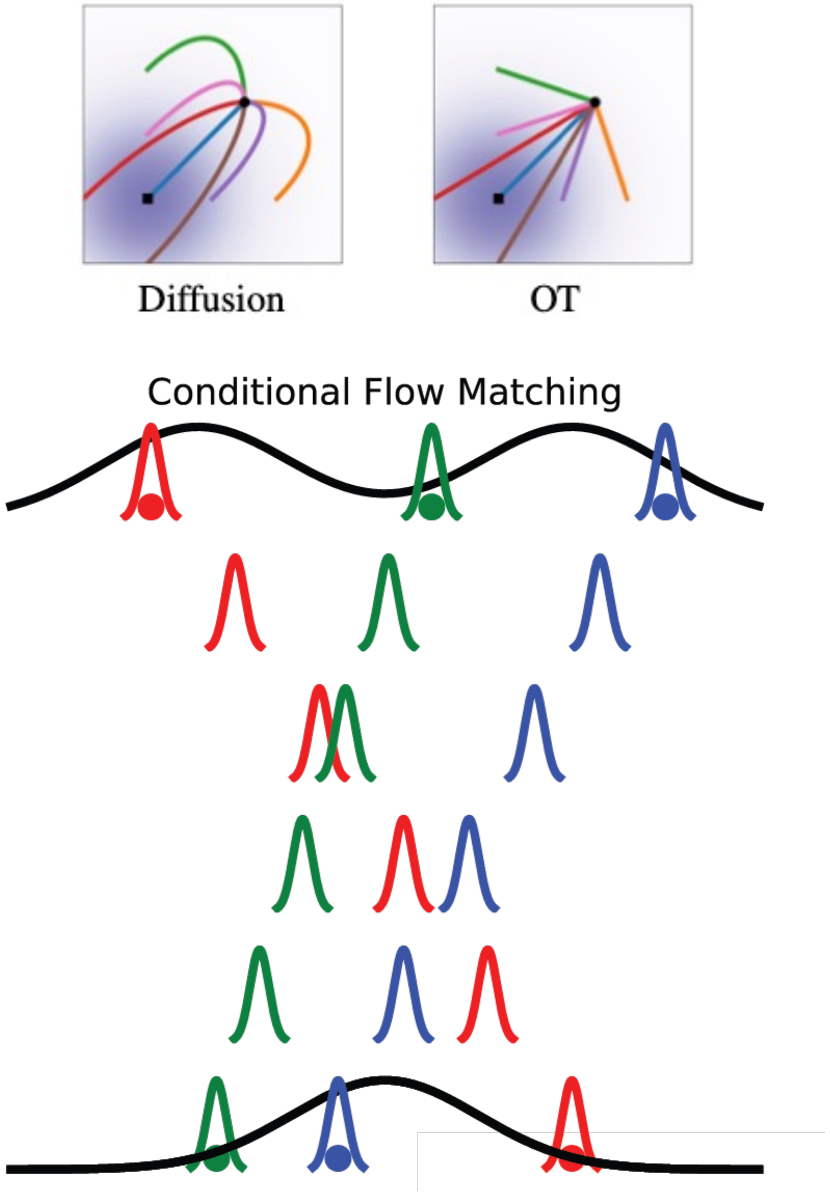
Time dependent flow $U_t(x_0)$




Flow Matching loss: $L_{FM}(\theta) = \mathbb{E}_{t,p_t(x)}||v_\theta(t,x) - u_t(x)||_2^2$

# Conditional Flow Matching

- Flows between Gaussians are simple
- Any distribution can be modeled as an infinite mixture of Gaussians
- Flow matching is the “law of total probability” for vector fields



# Main idea

Regressing against conditional flows is equivalent to regressing against the marginal flow in expectation.

# Main idea

Regressing against conditional flows is equivalent to regressing against the marginal flow in expectation.

***Theorem**: (Informally) Let $u_t(x)$ generate $p_t(x)$ and be of the form*

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

*then*

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

# Main idea

Regressing against conditional flows is equivalent to regressing against the marginal flow in expectation.

***Theorem**: (Informally) Let $u_t(x)$ generate $p_t(x)$ and be of the form*

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

*then*

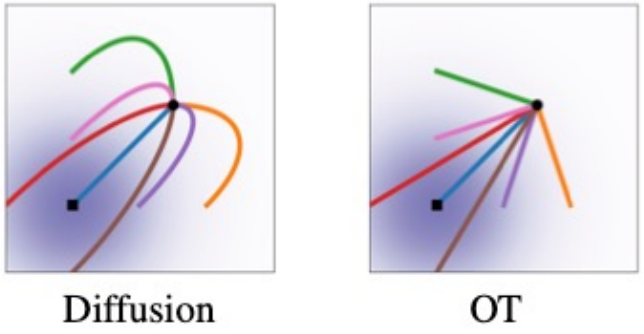
$$\nabla_\theta \mathbb{E}_{t,p_t(x)} ||v_\theta(t,x) - u_t(x)||_2^2 = \nabla_\theta \mathbb{E}_{t,q(z),p_t(x|z)} ||v_\theta(t,x) - u_t(x|z)||_2^2$$

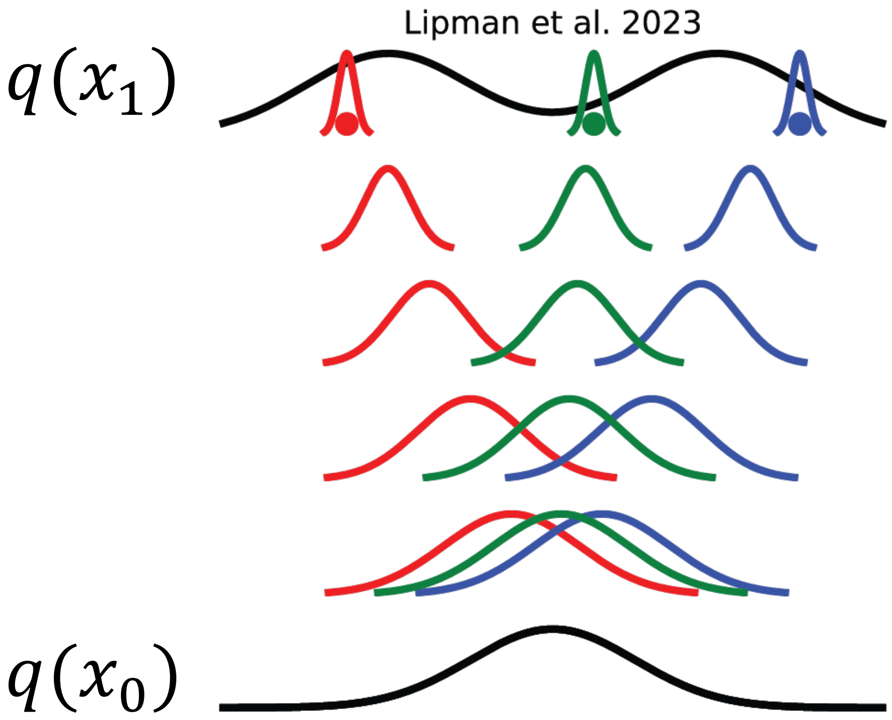
***Intuition:***

$$||v_\theta - u_t(x)||_2^2 = ||v_\theta||_2^2 - \left(v_\theta^T u_t(x)\right) + ||u_t(x)||_2^2$$

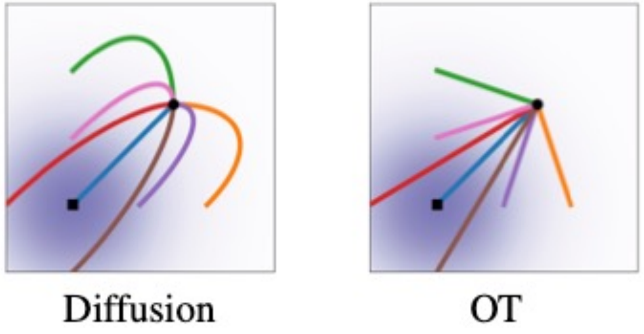
$$||v_\theta - u_t(x|z)||_2^2 = ||v_\theta||_2^2 - \left(v_\theta^T u_t(x|z)\right) + ||u_t(x|z)||_2^2$$

# Flow Matching



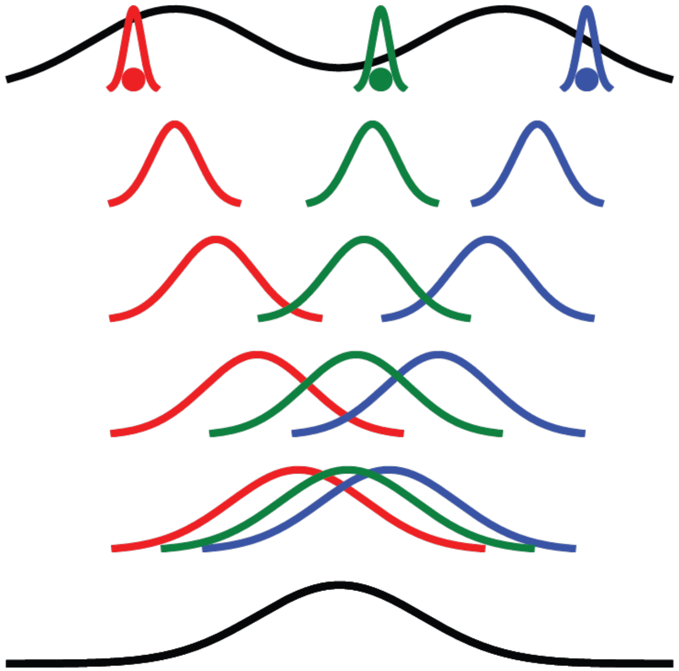
# Flow Matching




$p_t(x|z^1)$ $p_t(x|z^3)$

Lipman et al. 2023

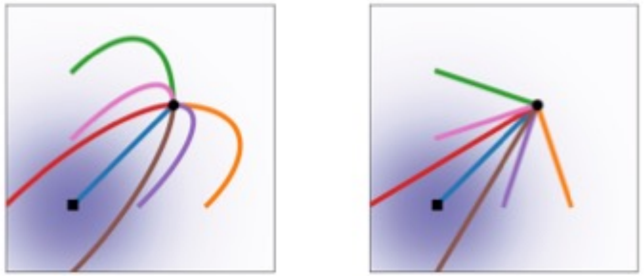
$q(x_1)$



$q(x_0)$

$q(z) \coloneqq q(x_1)\, p_t(x|z^2)$

# Flow Matching

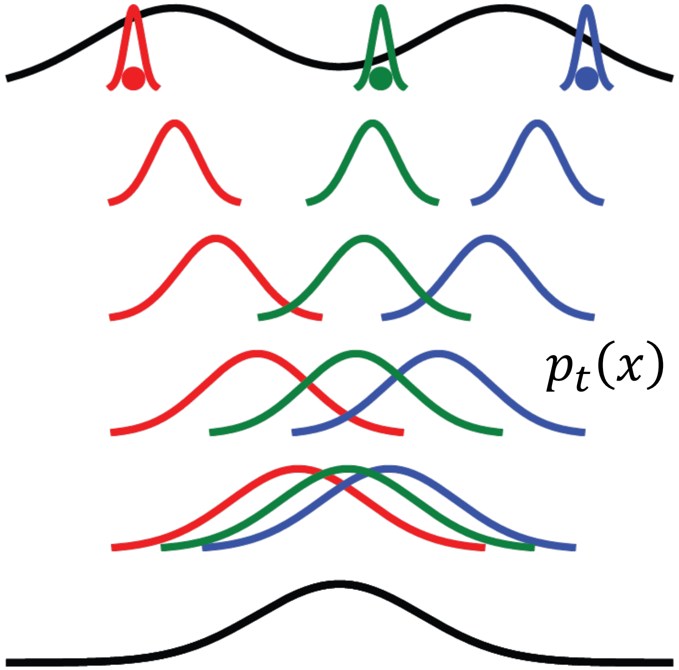


Diffusion  OT

$p_t(x|z^1)$  $p_t(x|z^3)$

Lipman et al. 2023

$q(x_1)$

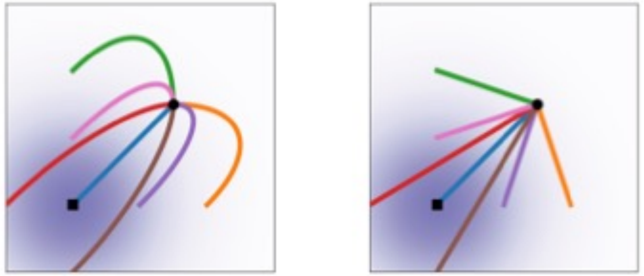


$$p_t(x) = \sum_z p_t(x|z)$$

$q(x_0)$

$q(z) \coloneqq q(x_1)\, p_t(x|z^2)$

# Flow Matching

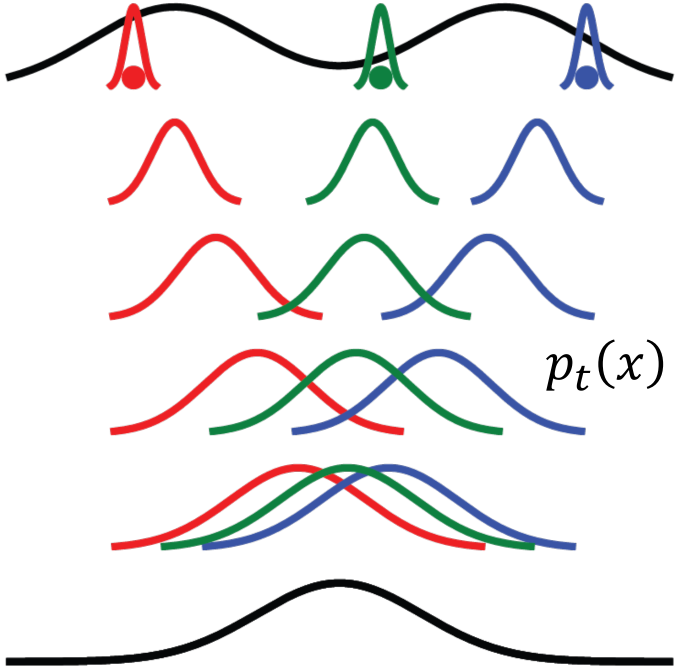


Diffusion OT

$p_t(x|z^1)$ $p_t(x|z^3)$

Lipman et al. 2023

$q(x_1)$



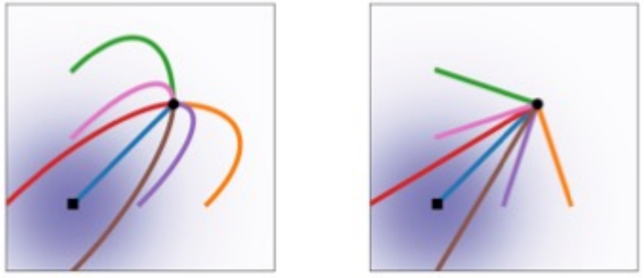
$$p_t(x) = \sum_z p_t(x|z)$$

$q(x_0)$

$q(z) \coloneqq q(x_1)$ $p_t(x|z^2)$

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

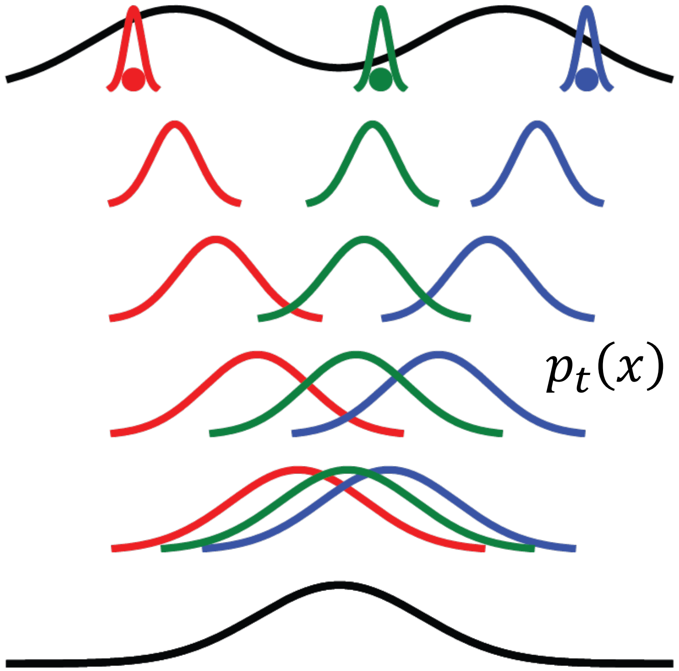
# Flow Matching




$p_t(x|z^1)$ $\quad$ $p_t(x|z^3)$

Lipman et al. 2023

$q(x_1)$



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

$q(x_0)$

$q(z) \coloneqq q(x_1)\, p_t(x|z^2)$

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

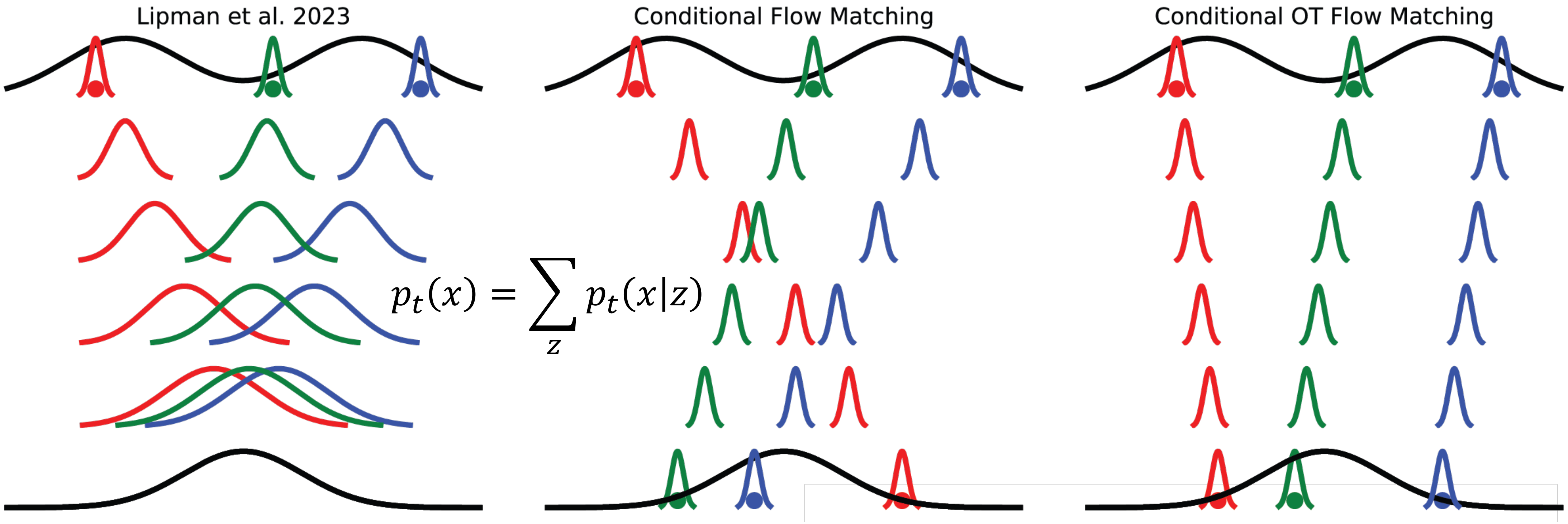
$$u_t(x) = \int \frac{u_t(x|z)p_t(x|z)q(z)}{p_t(x)} dz$$

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

# Conditional Flow Matching

$p_t(x|z^1)$ $\quad$ $p_t(x|z^3)$

$q(x_1)$



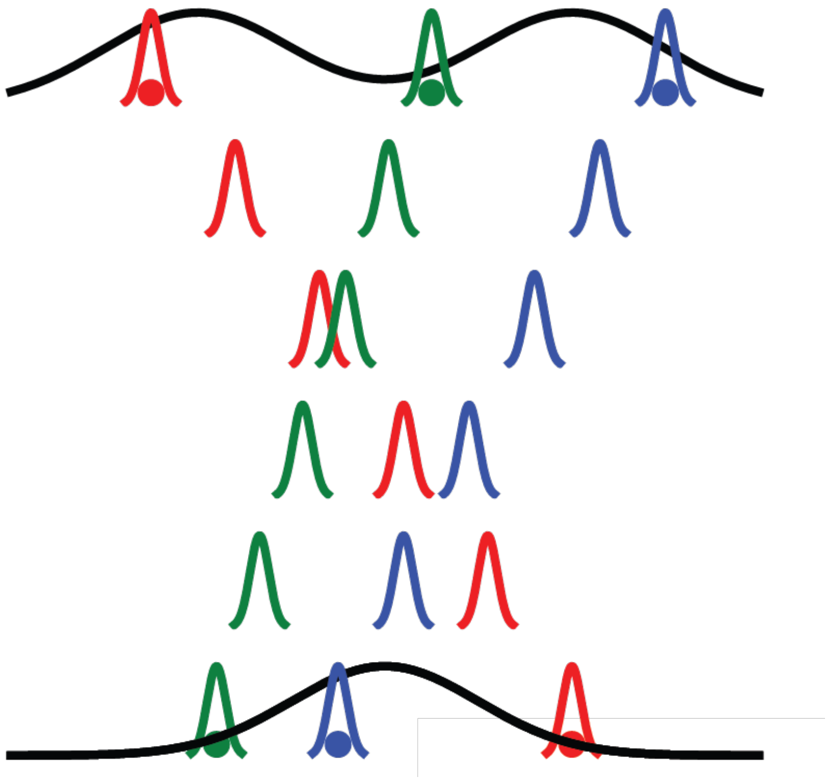

$$p_t(x) = \sum_z p_t(x|z)$$

$q(x_0)$

$q(z) := q(x_1)p_t(x|z^2)$ $\quad$ $q(z) := q(x_0)q(x_1)$ $\quad$ $q(z) := \pi(x_0, x_1)$

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

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




**Algorithm 1** Simplified Conditional Flow Matching

**Input:** Sample-able distributions $\boldsymbol{X}_0, \boldsymbol{X}_1$, bandwidth $\sigma$, batchsize $b$, initial network $v_\theta$.

**while** Training **do**

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

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

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

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

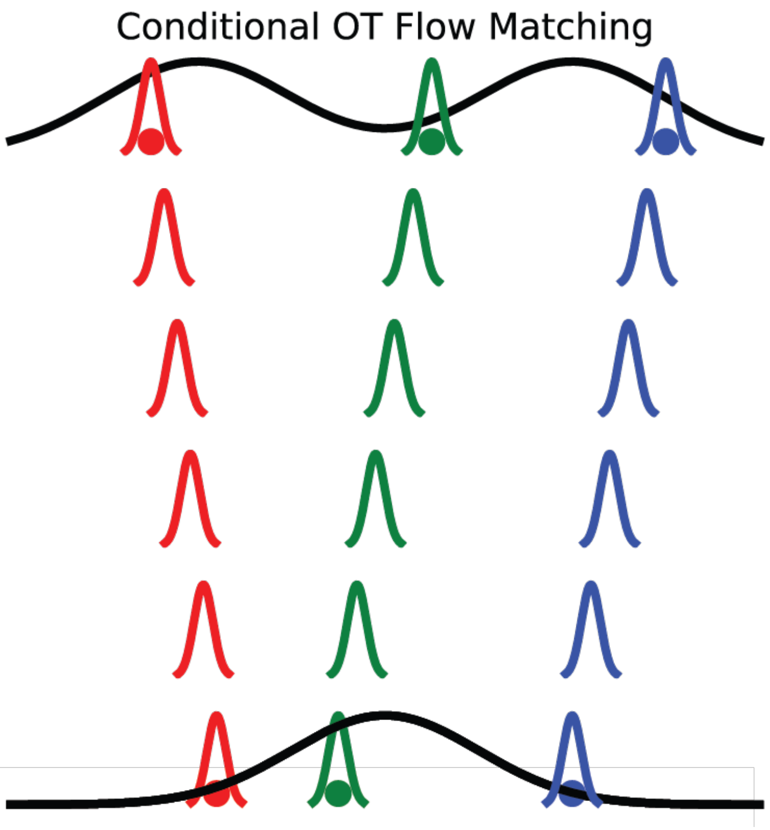
$x \sim \mathcal{N}(\mu_t, \sigma^2 I)$

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

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

**return** $v_\theta$

# + Static (mini-batch) OT Resampling Step




**Algorithm 2** Minibatch OT Conditional Flow Matching

**Input:** Sample-able distributions $\boldsymbol{X}_0, \boldsymbol{X}_1$, bandwidth $\sigma$, batch size $b$, initial network $v_\theta$.

**while** Training **do**

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

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

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

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

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

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

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

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

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

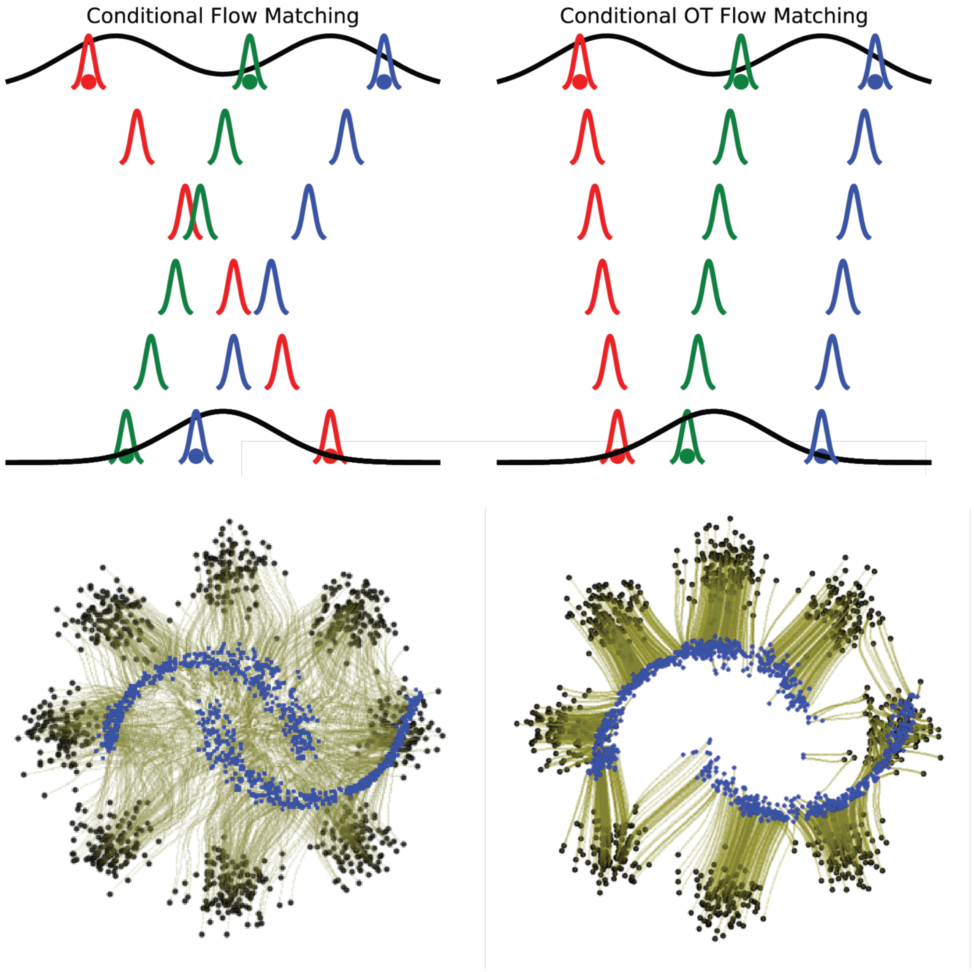
**return** $v_\theta$

Tong et al. TMLR 2024

# Why use optimal transport in flow matching?

More general framework:

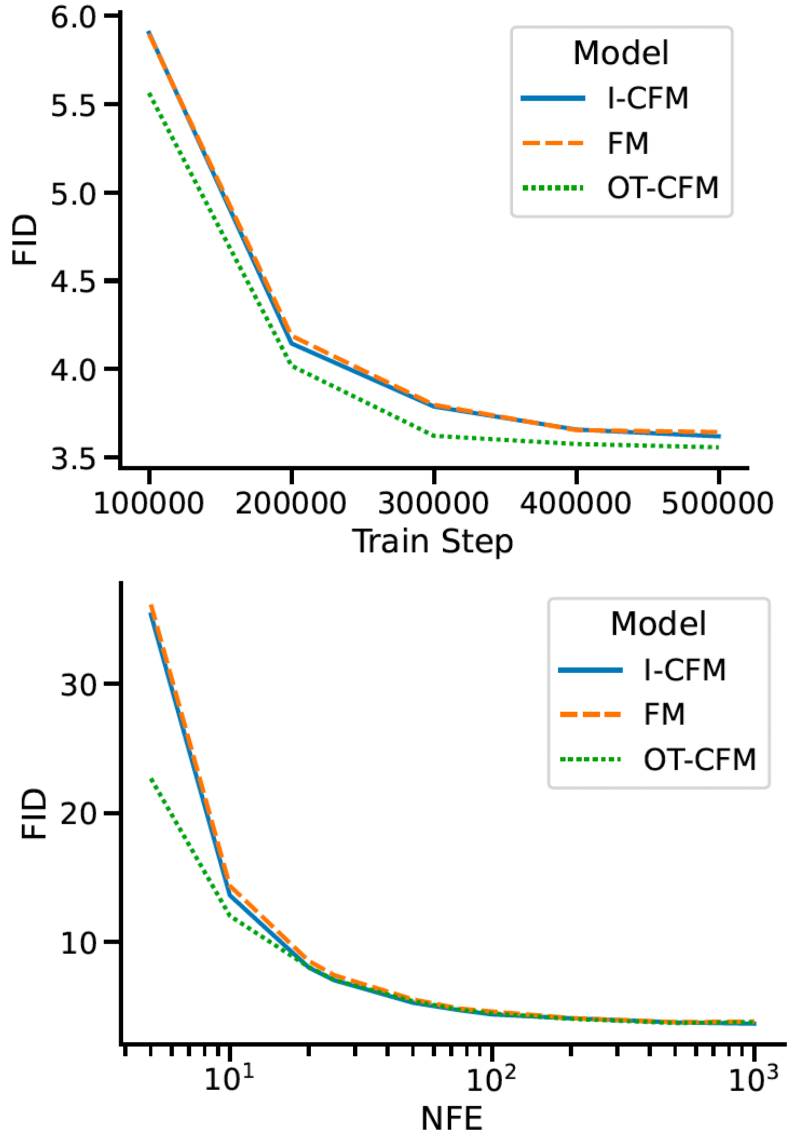
- Reduced variance in the objective via optimal transport leads to **faster training**
- Straighter inference paths via optimal transport leads to **faster inference**
- Can be applied to new problems where we care about the paths



# Why use optimal transport in flow matching?

More general framework:

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

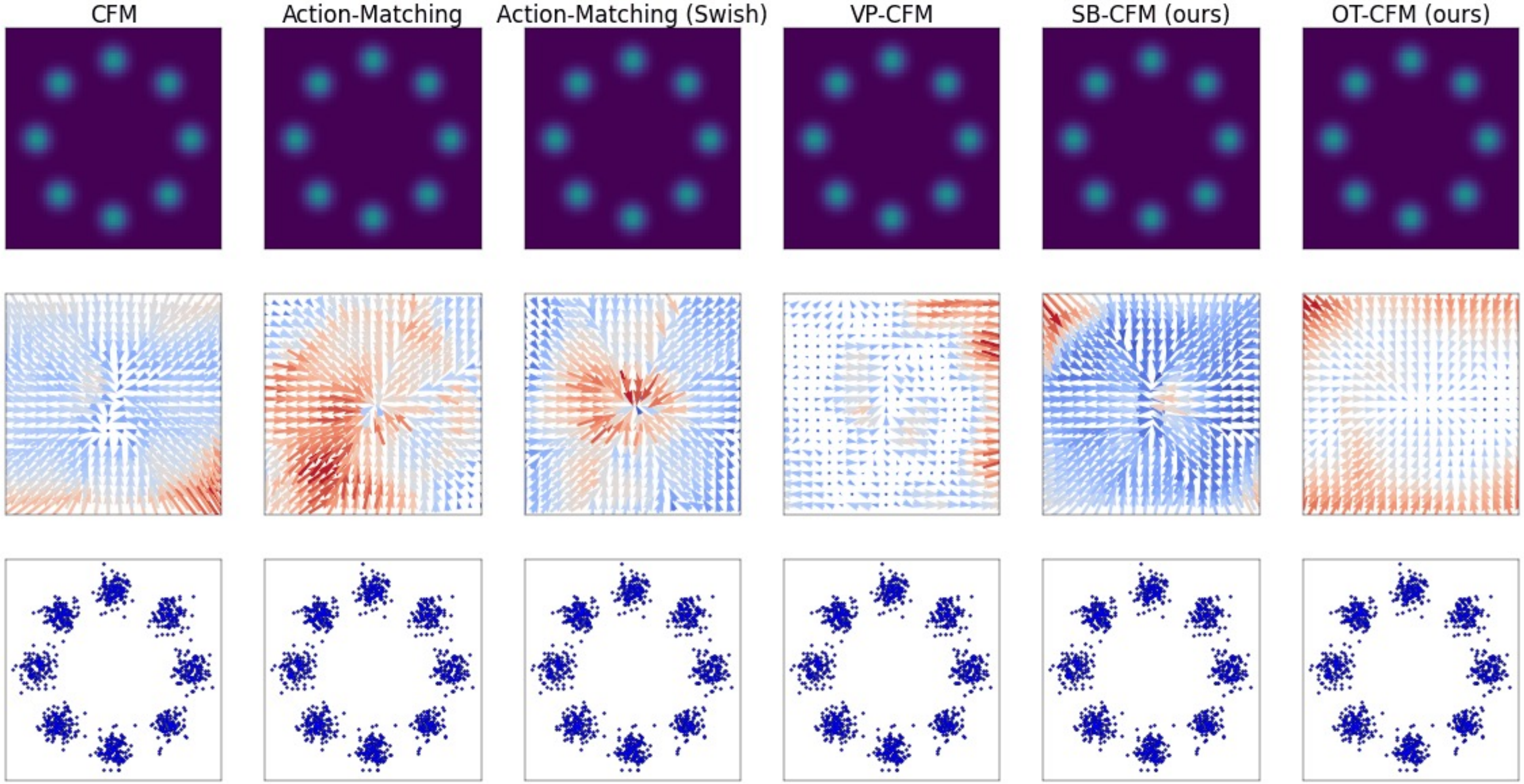


# Comparing choices of $u_t(x|z)$, $p_t(x|z)$, and $q_t(z)$

Marginal Probability $p_t(x)$

Marginal field $u_t(x)$

Time dependent flow $U_t(x_0)$

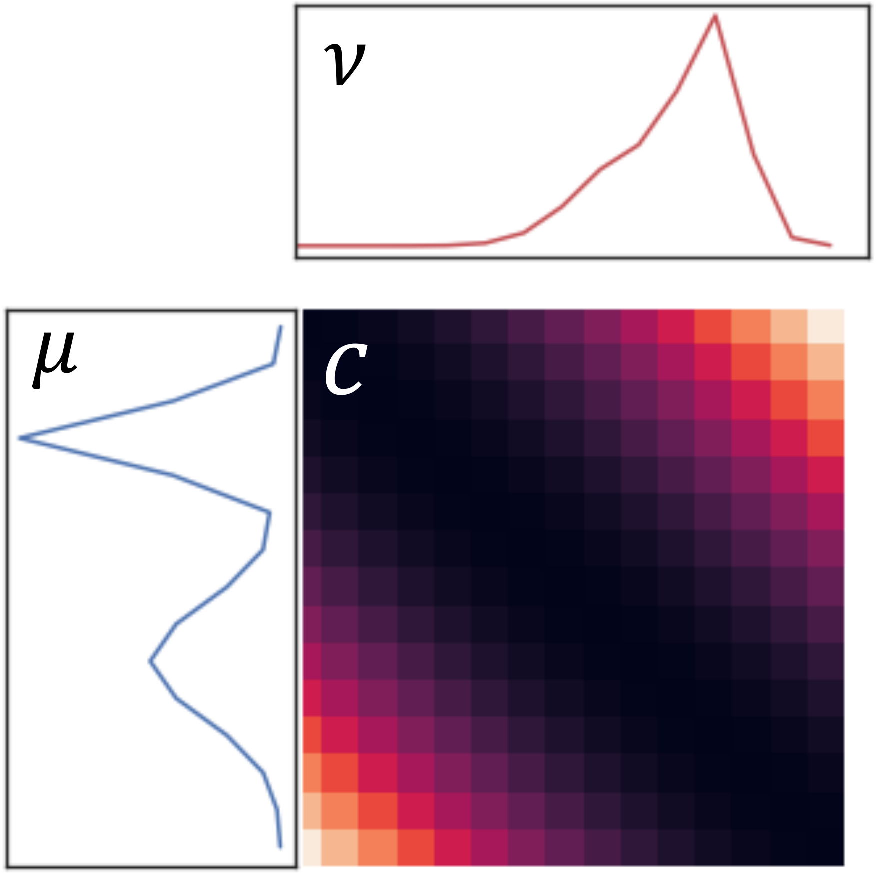


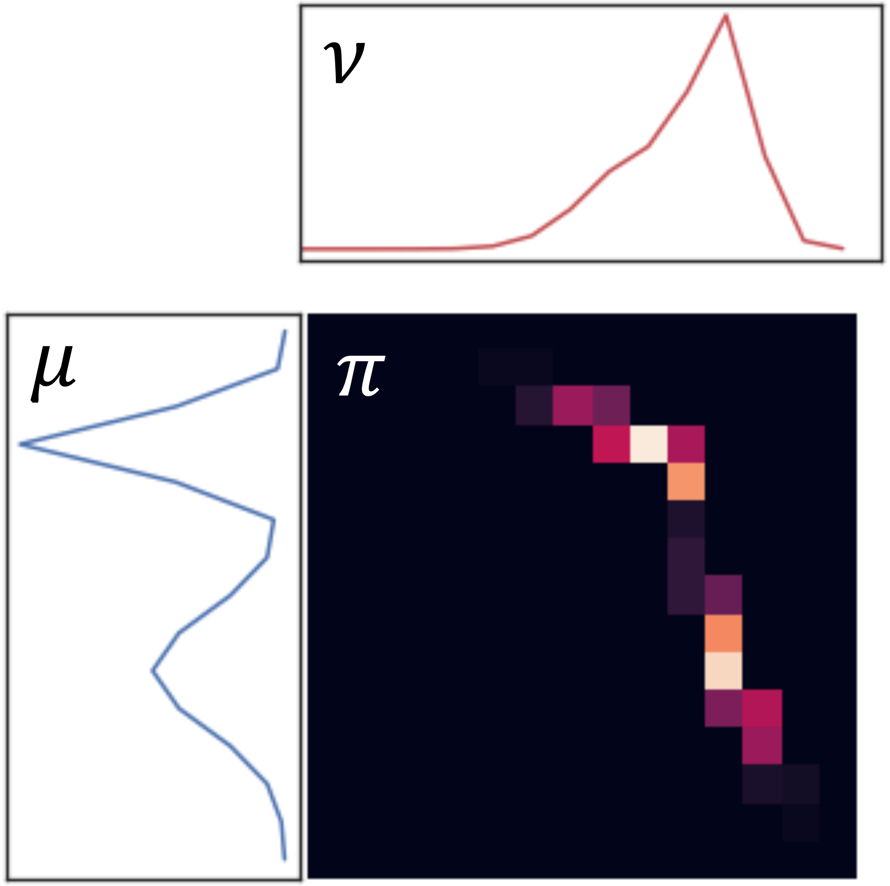
# Comparing choices of $u_t(x|z)$, $p_t(x|z)$, and $q_t(z)$

| Probability Path | $q(z)$ | $\mu_t(z)$ | $\sigma_t$ | Cond. OT | Marginal OT | General source |
|---|---|---|---|---|---|---|
| Var. Exploding (Song & Ermon, 2019) | $q(x_1)$ | $x_1$ | $\sigma_{1-t}$ | × | × | × |
| Var. Preserving (Ho et al., 2020) | $q(x_1)$ | $\alpha_{1-t}x_1$ | $\sqrt{1-\alpha_{1-t}^2}$ | × | × | × |
| Flow Matching (Lipman et al., 2023) | $q(x_1)$ | $tx_1$ | $t\sigma - t + 1$ | ✓ | × | × |
| Rectified Flow Liu (2022) | $q(x_0)q(x_1)$ | $tx_1 + (1-t)x_0$ | $0$ | ✓ | × | ✓ |
| Var. Pres. Stochastic Interpolant Albergo & Vanden-Eijnden (2023) | $q(x_0)q(x_1)$ | $\cos(\frac{1}{2}\pi t)x_0 + \sin(\frac{1}{2}\pi t)x_1$ | $0$ | ✓ | × | ✓ |
| Independent CFM | $q(x_0)q(x_1)$ | $tx_1 + (1-t)x_0$ | $\sigma$ | ✓ | × | ✓ |
| (Ours) Optimal Transport CFM | $\pi(x_0, x_1)$ | $tx_1 + (1-t)x_0$ | $\sigma$ | ✓ | ✓ | ✓ |
| (Ours) Schrödinger Bridge CFM | $\pi_{2\sigma^2}(x_0, x_1)$ | $tx_1 + (1-t)x_0$ | $\sigma\sqrt{t(1-t)}$ | ✓ | ✓ | ✓ |

# Exact Solution

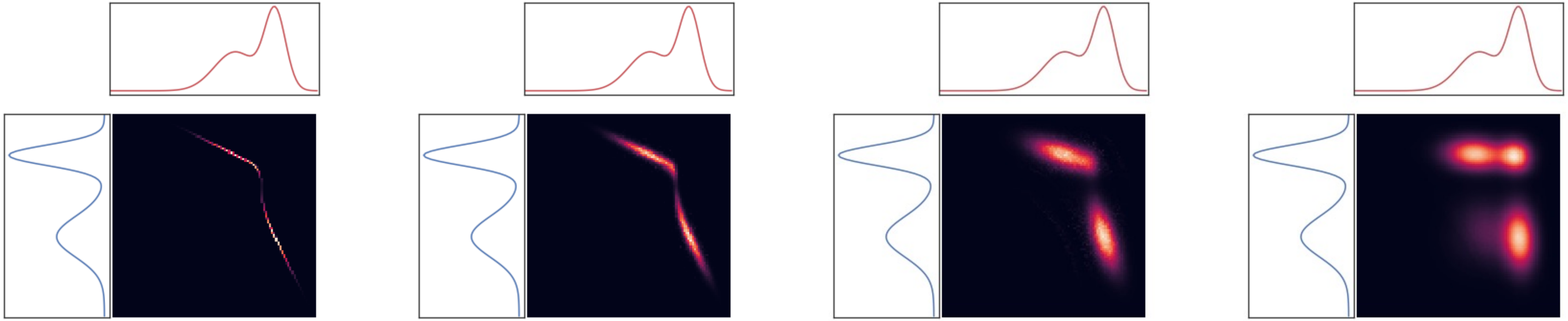
$$W(\mu,\nu) = \min_{\pi} \sum c \cdot \pi \quad s.t. \quad \sum_j \pi_{ij} = \mu_i \; and \; \sum_i \pi_{ij} = \nu_j$$



# Entropic Regularization

$$S(\mu, \nu) = \min_{\pi} \sum c \cdot \pi - \epsilon H(\pi) \;\; where\; H(\pi) = -\sum_{ij} \pi_{ij} (\log \pi_{ij} - 1)$$

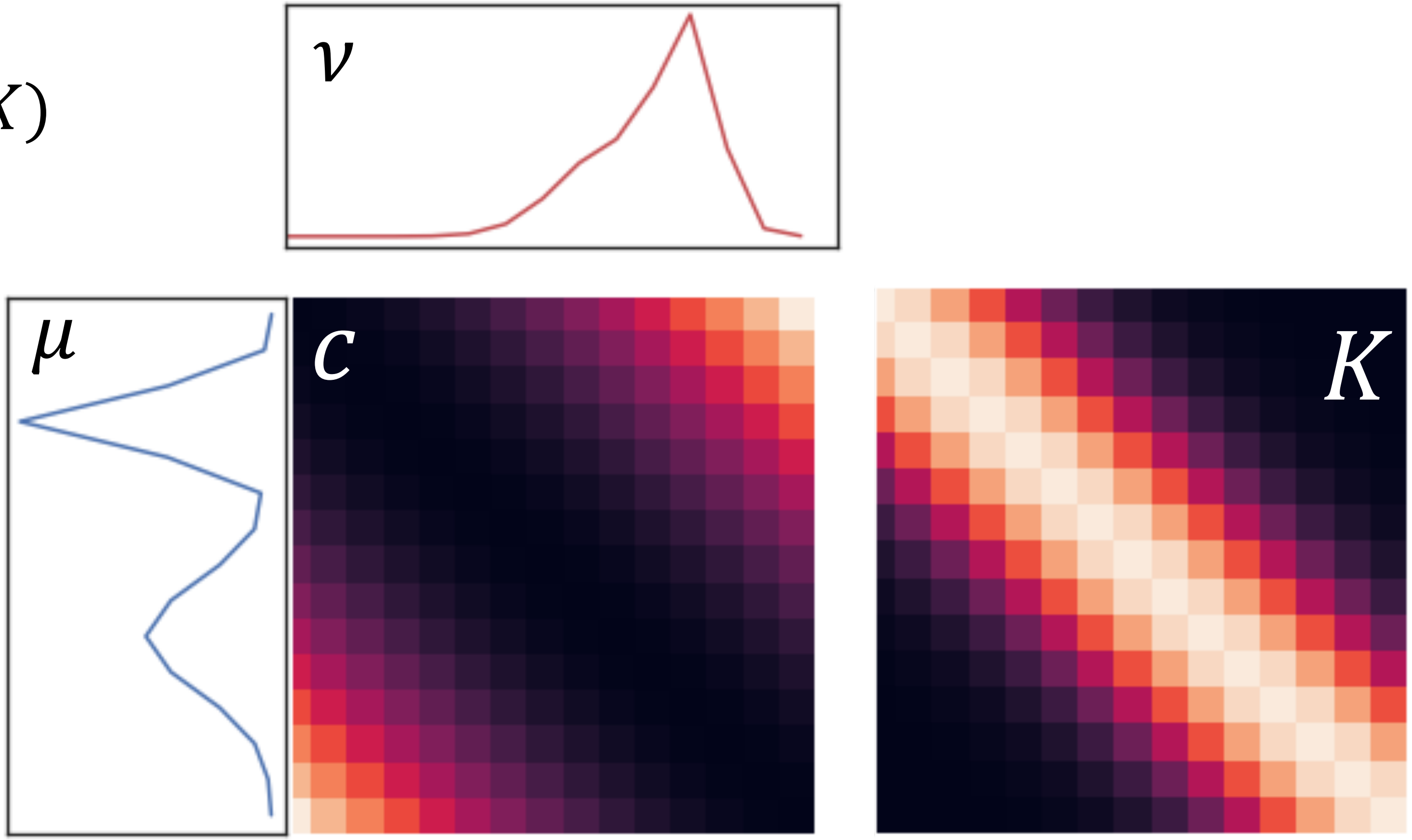


# Entropic Regularization with Matrix Scaling

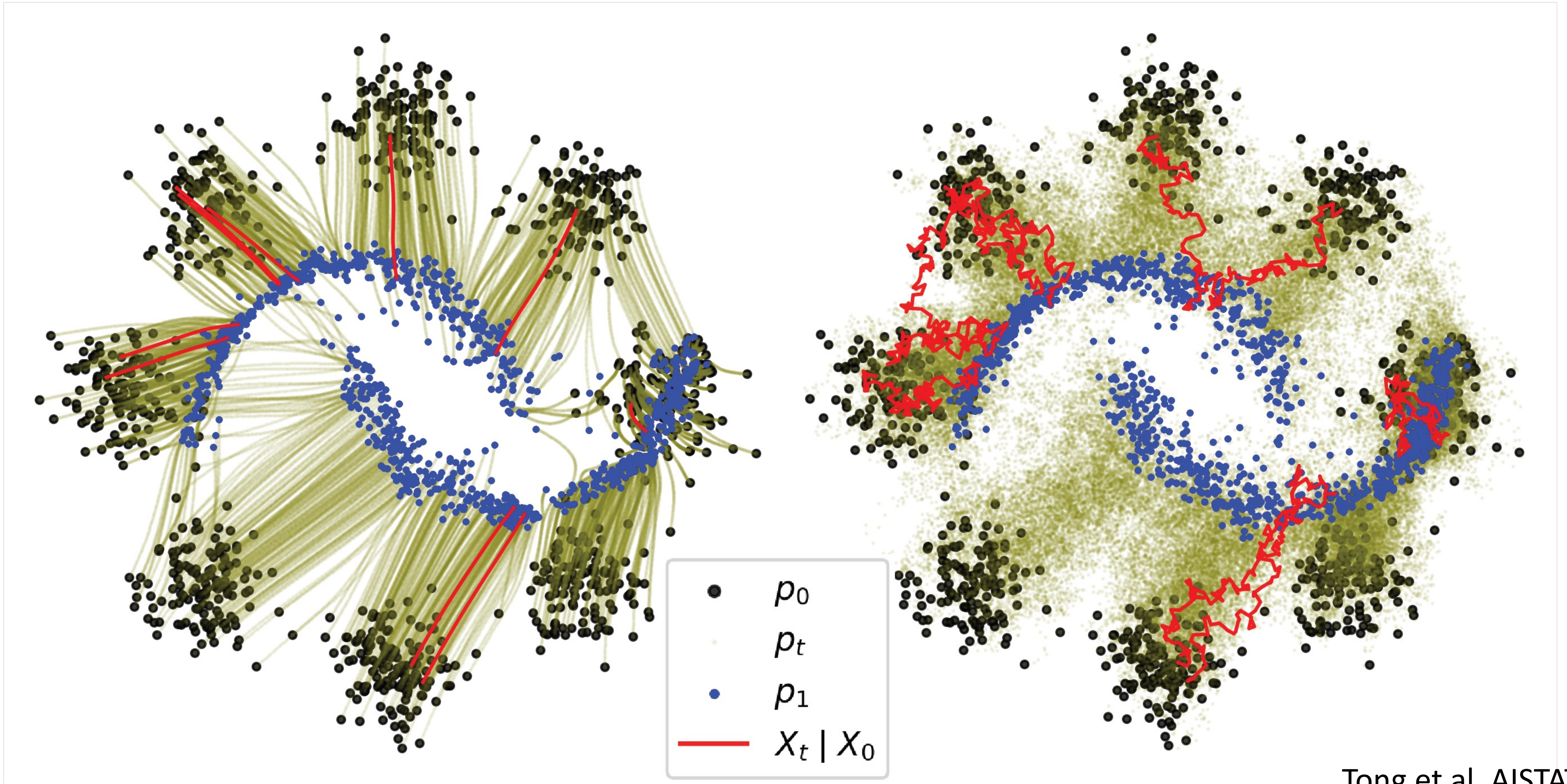
- Gibbs kernel
  - $K_{ij} = e^{\frac{-c_{ij}}{\lambda}}$
- $S(\mu, \nu) = argmin_P \; KL(P \mid K)$

- P is of the form
  - $P = diag(a)\; K\; diag(b)$
  
  For vectors a, b

  And this can be solved with
  the Sinkhorn algorithm!
  i.e. iterative proportional fitting



# Simulation-free Score and Flow Matching $(SF)^2M$




Tong et al. AISTATS 2024

# Stochastic Differential Equations

- Stochastic Differential Equation

$$dx = u_t(x)\,dt + g(t)\,dw_t$$

- Fokker-Plank and continuity equation

$$\frac{\partial p}{\partial t} = -\nabla \cdot (p_t u_t) + \frac{g^2(t)}{2} \Delta p_t$$

- Probability flow ODE

$$dx = \underbrace{\left[ u_t(x) - \frac{g(t)^2}{2} \nabla \log p_t(x) \right]}_{\hat{u}_t(x)} dt$$

# Score and Flow Matching

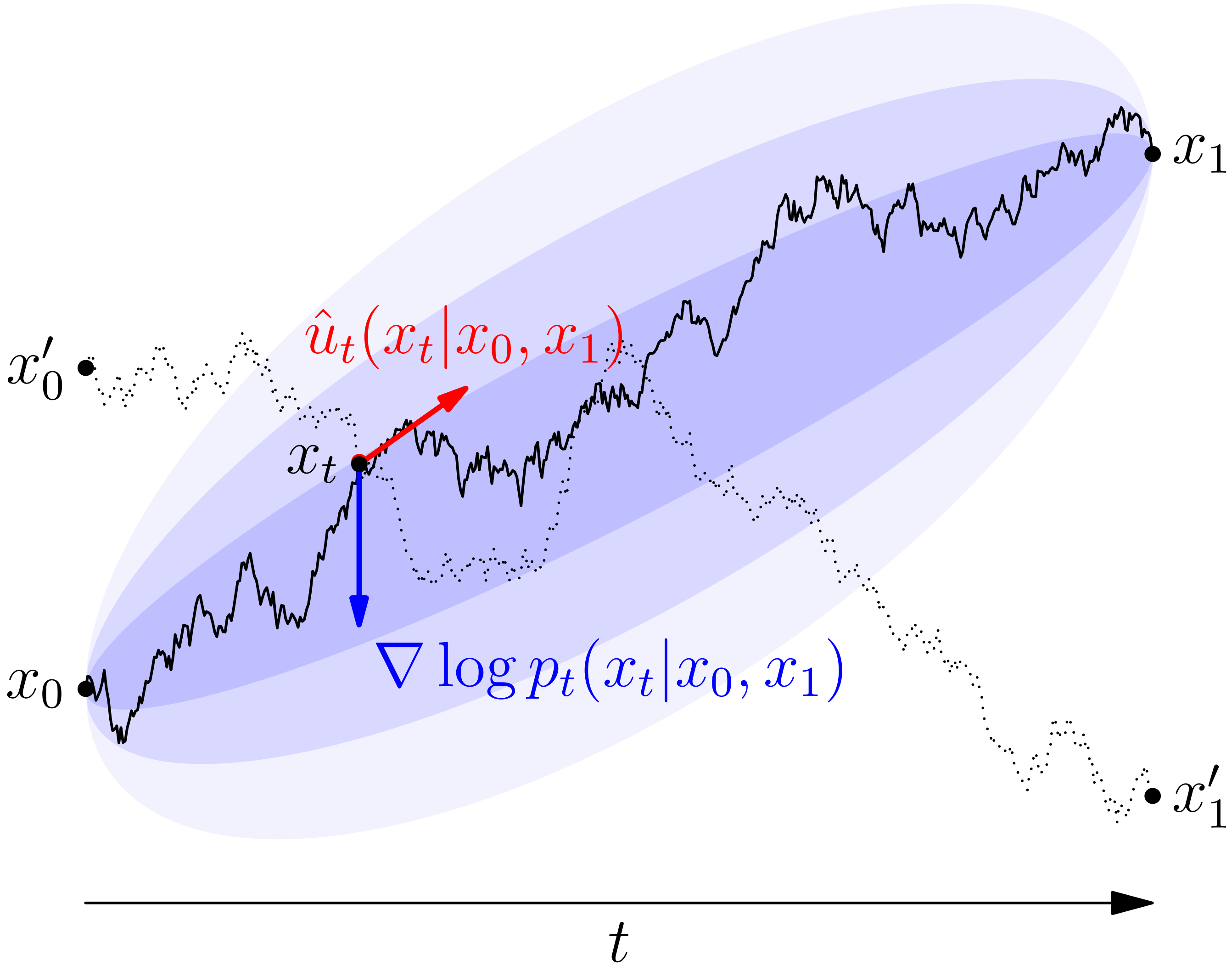
## Score and Flow Matching

$$\mathcal{L}_{\mathrm{U[SF]^2M}}(\theta) = \mathbb{E}_{t\sim\mathcal{U}(0,1), x\sim p_t(x)} \Big[ \underbrace{\|v_\theta(t,x) - \hat{u}_t(x)\|^2}_{\text{flow matching loss}} + \lambda(t) \underbrace{\|s_\theta(t,x) - \nabla \log p_t(x)\|^2}_{\text{score matching loss}} \Big]$$

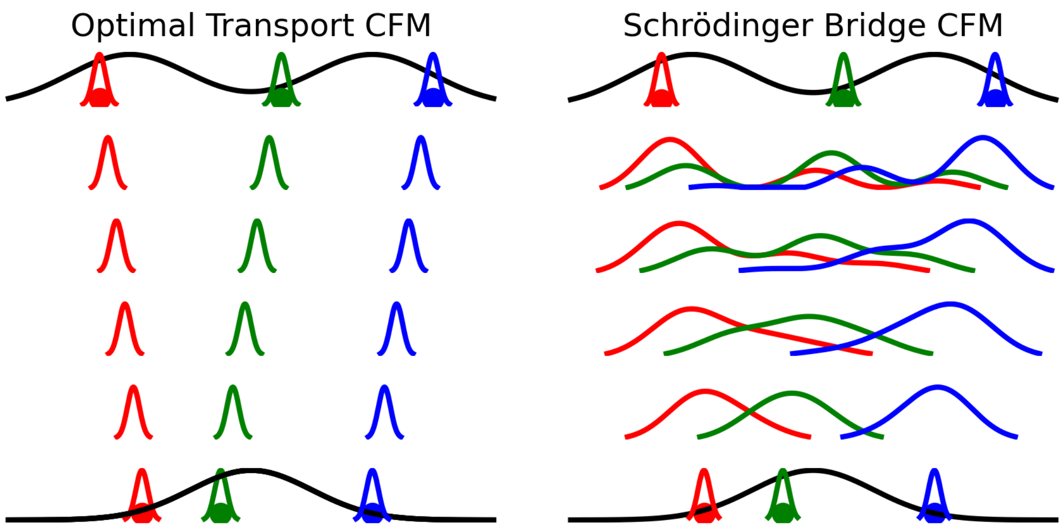
## Conditional Score and Flow Matching

$$\mathcal{L}_{\mathrm{[SF]^2M}}(\theta) = \mathbb{E}_{t\sim\mathcal{U}(0,1), z\sim q(z), x\sim p_t(x|z)} \Big[ \underbrace{\|v_\theta(t,x) - \hat{u}_t(x|z)\|^2}_{\text{conditional flow matching loss}} + \lambda(t) \underbrace{\|s_\theta(t,x) - \nabla \log p_t(x|z)\|^2}_{\text{conditional score matching loss}} \Big]$$

# A Geometric Intuition



# Schrödinger Bridges




Problem statement

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

"Most likely stochastic process under observation"

Diffusion Schrödinger Bridges as mixtures of Brownian bridges

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

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

# Determinstic vs. Stochastic

**OT-CFM**

- Learns dynamic OT paths between distributions
- Faster inference time by creating simpler flows

**SF2M**

- Learns entropic OT paths between distributions
- Slightly "more robust" in high dimensions

$$\mathcal{L}_{[\mathrm{SF}]^2\mathrm{M}}(\theta) = \mathbb{E}_{t\sim\mathcal{U}(0,1),z\sim q(z),x\sim p_t(x|z)}\Big[\underbrace{\|v_\theta(t,x) - \hat{u}_t(x|z)\|^2}_{\text{conditional flow matching loss}} + \lambda(t)\underbrace{\|s_\theta(t,x) - \nabla\log p_t(x|z)\|^2}_{\text{conditional score matching loss}}\Big]$$

# Applications to Biology

**Disease Dynamics**

- Focus: Dynamics of Metastasis
- Biological result: Characterize and disrupt the metastasis of triple negative breast cancer in mouse model
- Computational result: Learn gene regulatory networks from single-cell time series

**Protein Design**

**Molecule Sampling (in progress)**

# Applications to Biology

**Disease Dynamics**

- Focus: Dynamics of Metastasis
- Biological result: Characterize and disrupt the metastasis of triple negative breast cancer in mouse model
- Computational result: Learn gene regulatory networks from single-cell time series

**Protein Design**

- Focus: Designing de novo binders of difficult targets
- Biological goal: Faster, more effective, biologic drug candidates
- Computational goal: AlphaFold generates only one of many possible structures, we want all of them

**Molecule Sampling (in progress)**

# Applications to Biology

**Disease Dynamics**

- Focus: Dynamics of Metastasis
- Biological result: Characterize and disrupt the metastasis of triple negative breast cancer in mouse model
- Computational result: Learn gene regulatory networks from single-cell time series

**Protein Design**

- Focus: Designing de novo binders of difficult targets
- Biological goal: Faster, more effective, biologic drug candidates
- Computational goal: AlphaFold generates only one of many possible structures, we want all of them

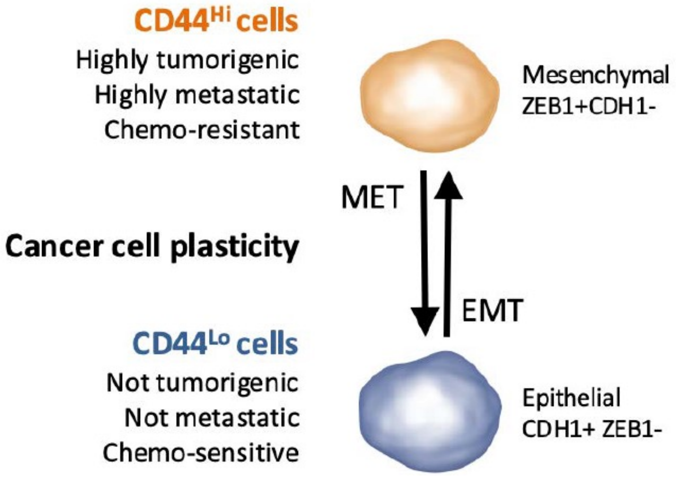
**Molecule Sampling (in progress)**

- Focus: Learning to sample the space of valid conformations
- Biological goal: Better sampling of the dynamics of molecules
- Computational goal: Learning flow with access to energy but no data

# Framing the biological problem: Triple Negative Breast Cancer Models of the Mesenchymal to Epithelial transition

Christine Chaffer

Shabarni Gupta



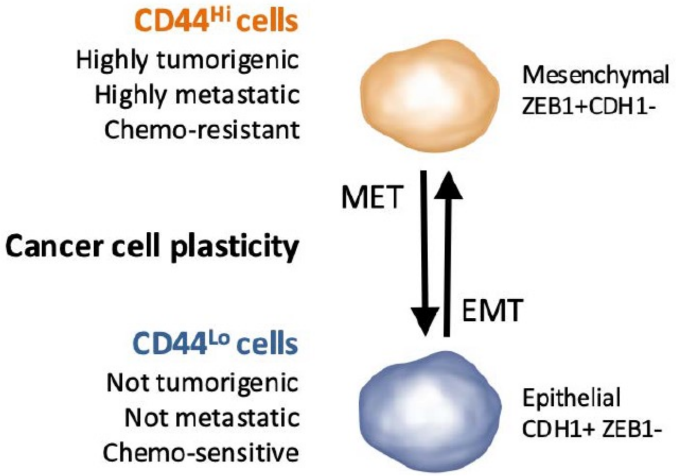

- TNBC has no know targeted therapies

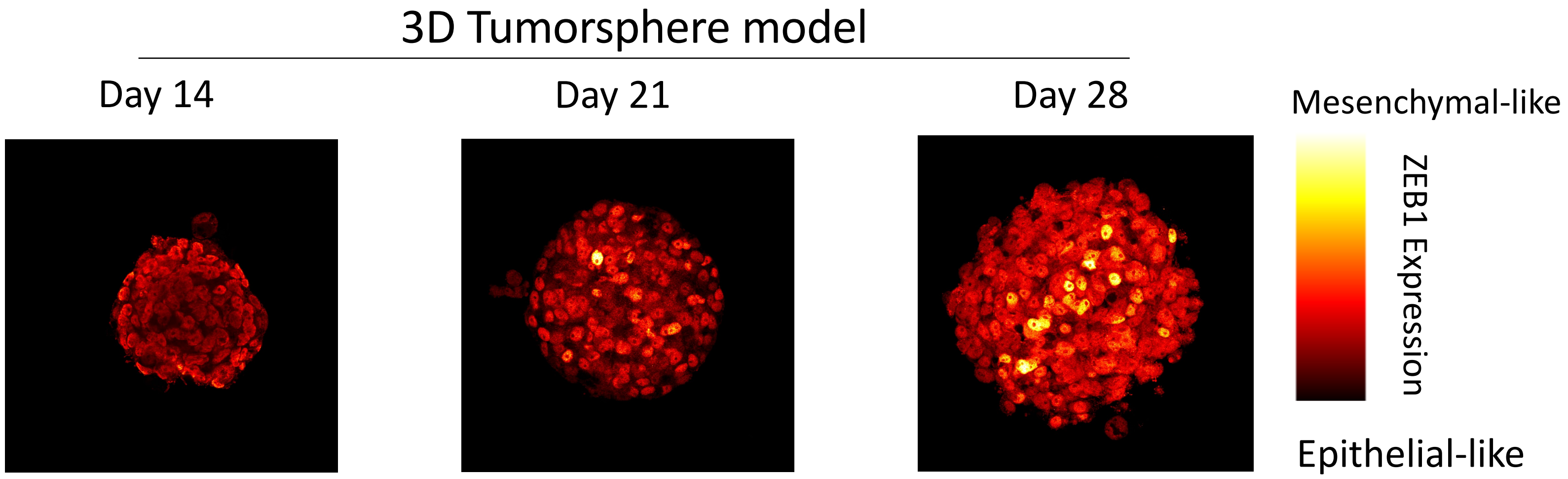
Tong*, Kuchroo*, et al. 2023

# Framing the biological problem: Triple Negative Breast Cancer Models of the Mesenchymal to Epithelial transition


Christine Chaffer


Shabarni Gupta

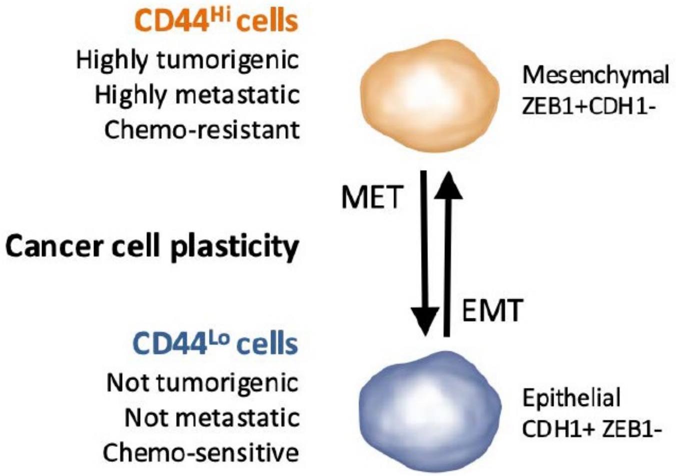


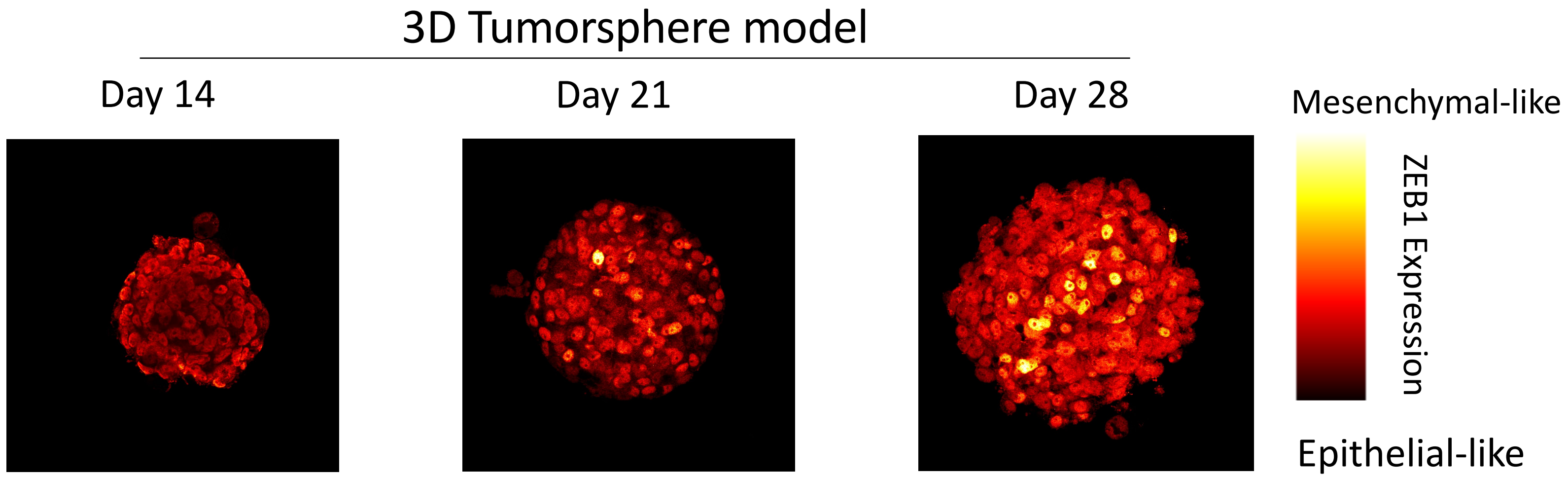




- TNBC has no know targeted therapies
- MET requires 3D cultures

Tong*, Kuchroo*, et al. 2023

# Framing the biological problem: Triple Negative Breast Cancer Models of the Mesenchymal to Epithelial transition


- TNBC has no know targeted therapies
- MET requires 3D cultures

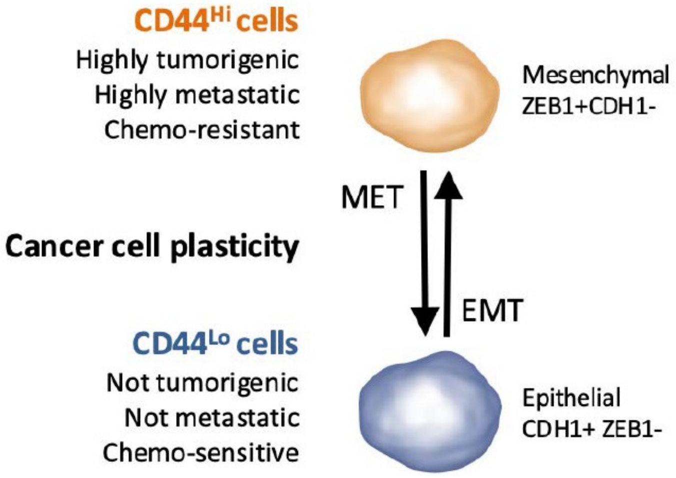
**What drives the MET cell state transition in TNBC?**

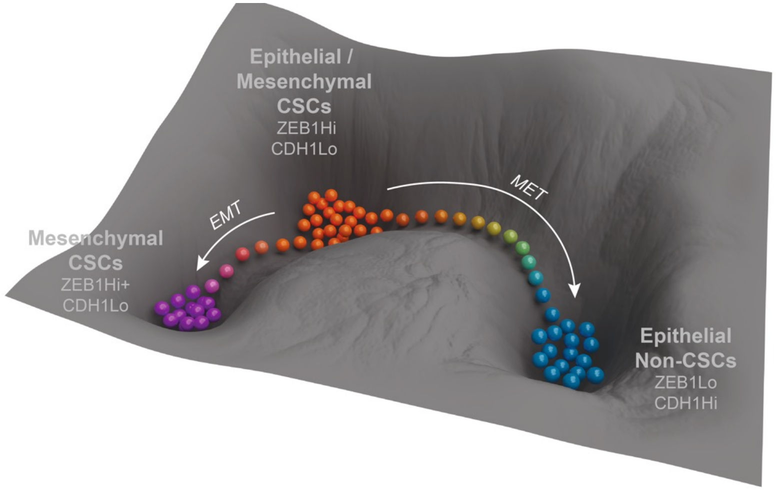
Tong*, Kuchroo*, et al. 2023

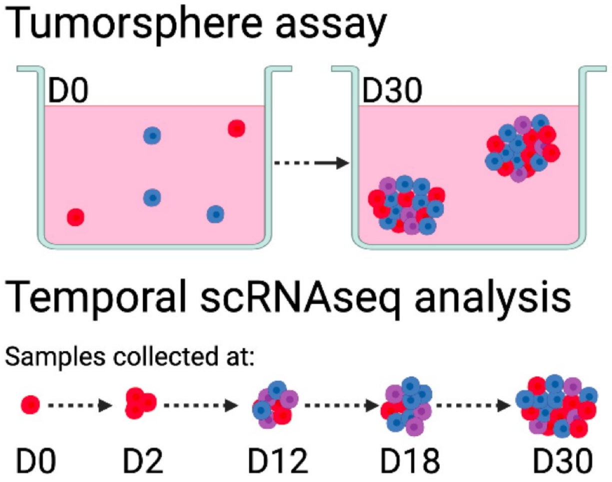
# Framing the biological problem: Triple Negative Breast Cancer Models of the Mesenchymal to Epithelial transition


Christine Chaffer


Shabarni Gupta



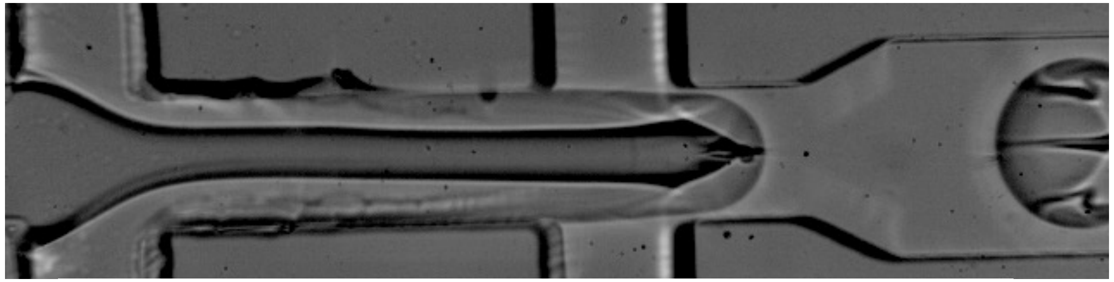

- TNBC has no know targeted therapies
- MET requires 3D cultures

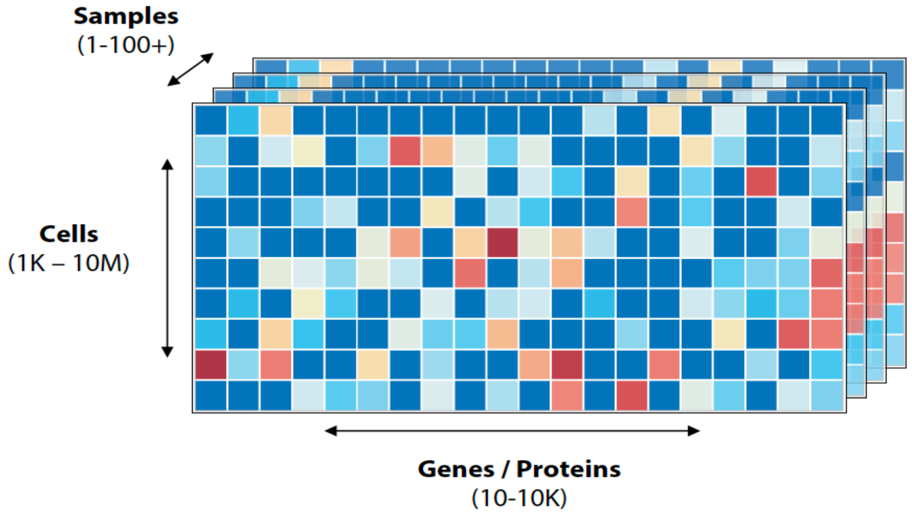
**What drives the MET cell state transition in TNBC?**

Tong*, Kuchroo*, et al. 2023

# A bit about single-cell data

- Transcriptomics is cheap!
- Destructive
- DNA → **RNA** → Protein
- Each cell is a vector in $\mathbb{R}^d_{\geq 0}$
- 1k to 10M cells 10-10k features

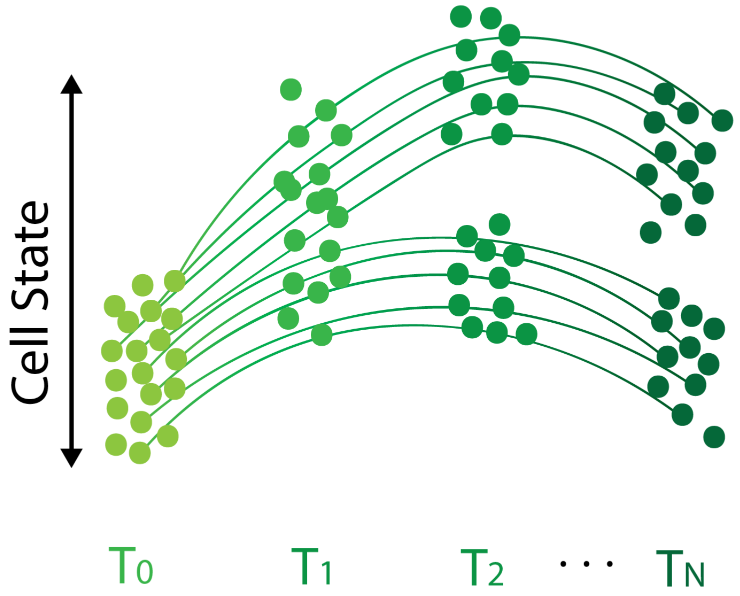




# The general biological problem – Destructive Measurements

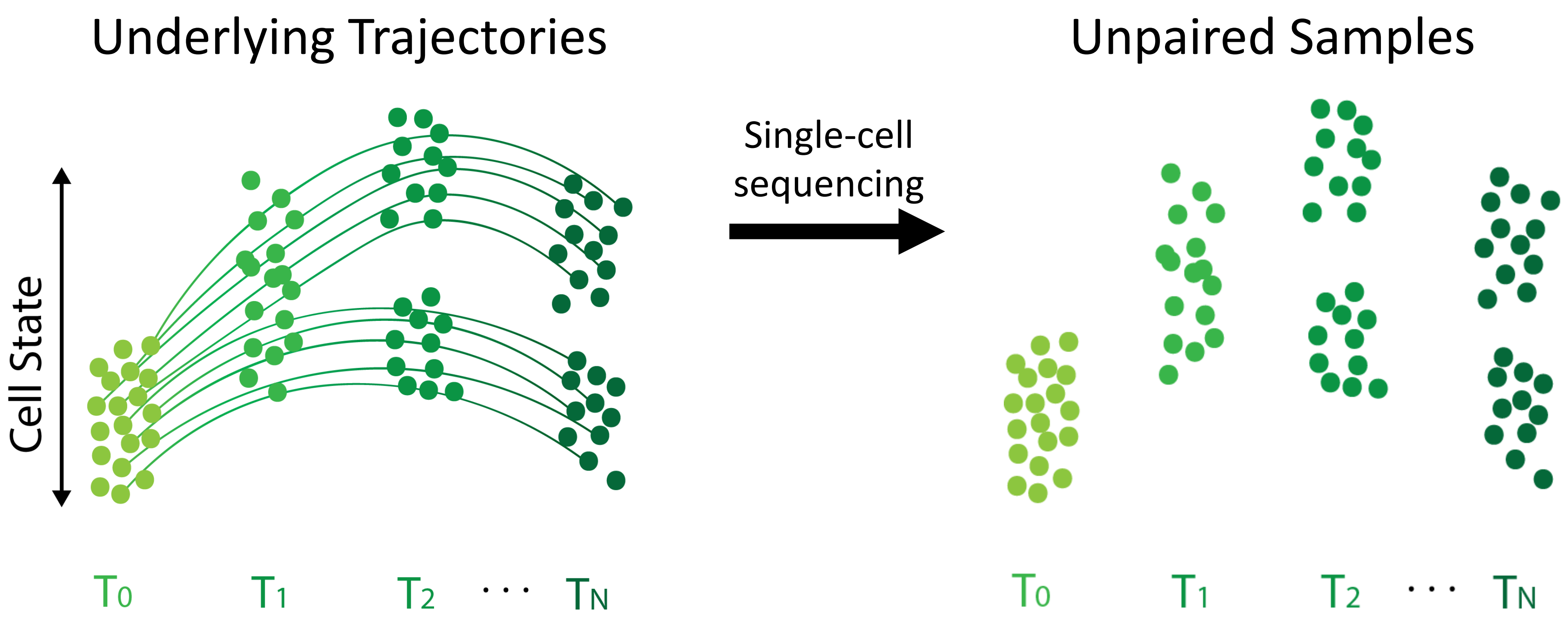
What drives the MET cell state transition in TNBC?

Underlying Trajectories



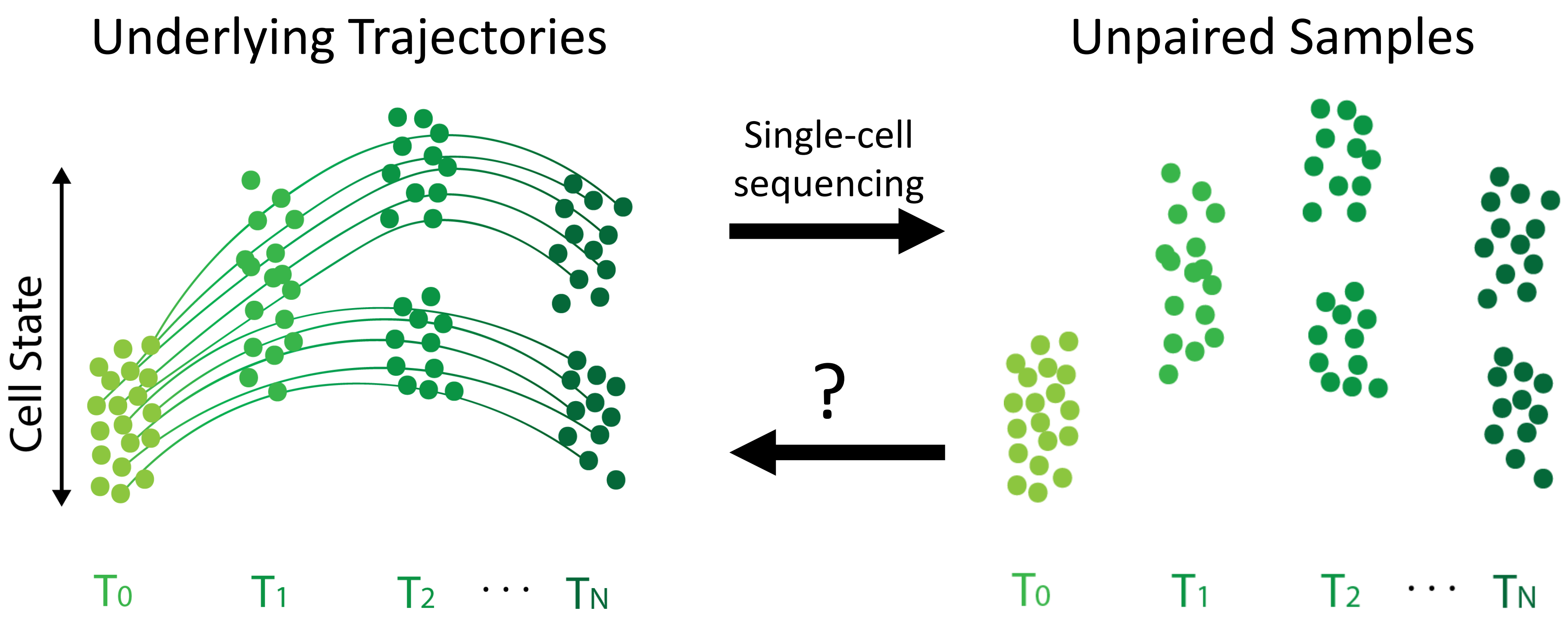
# The general biological problem – Destructive Measurements

What drives the MET cell state transition in TNBC?



# The general biological problem – Destructive Measurements

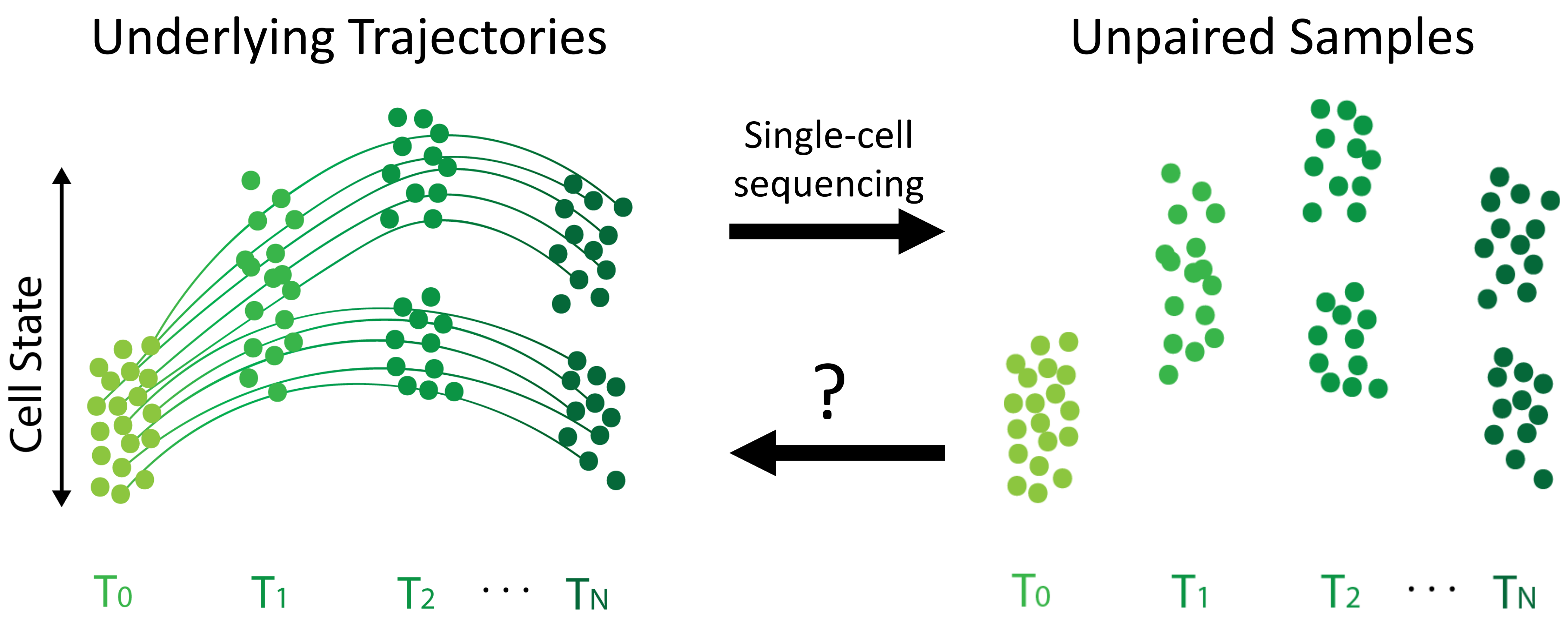
What drives the MET cell state transition in TNBC?




- Learning underlying trajectories from unpaired samples

# The general biological problem – Destructive Measurements

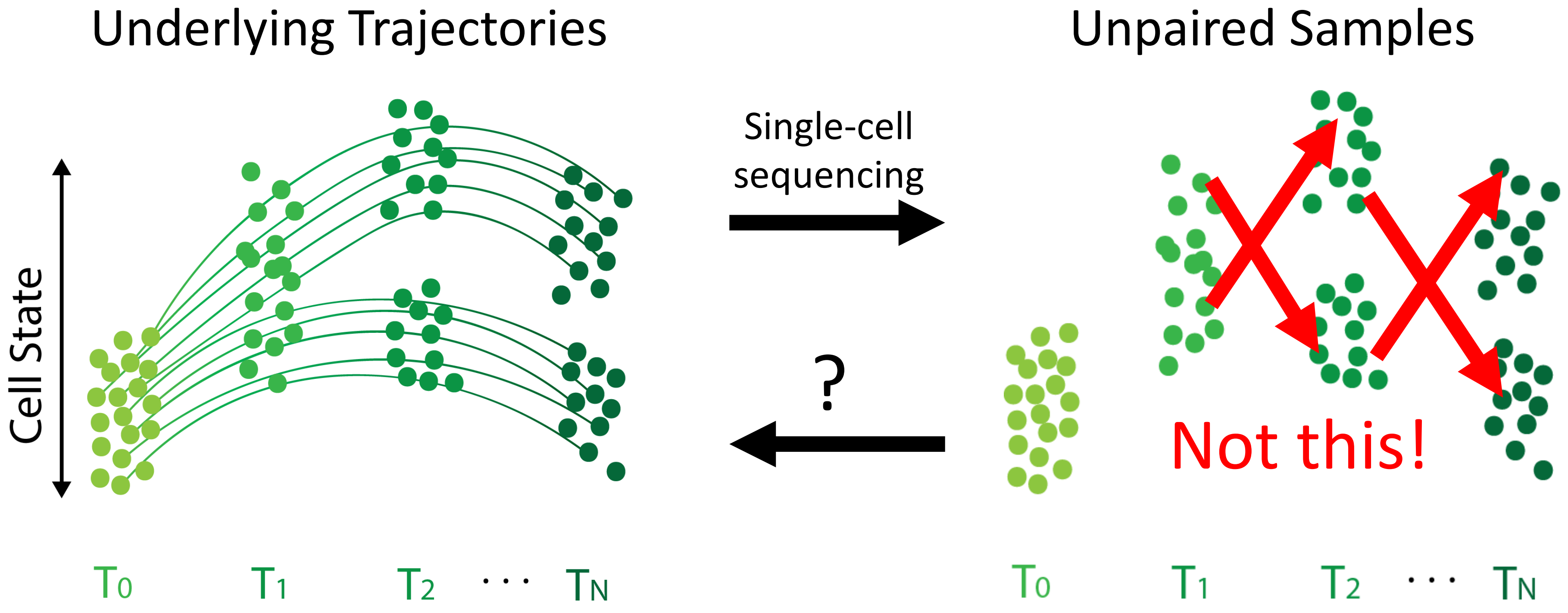
What drives the MET cell state transition in TNBC?




- Learning underlying trajectories from unpaired samples
- Many trajectories for unpaired samples but which should we pick?

# The general biological problem – Destructive Measurements

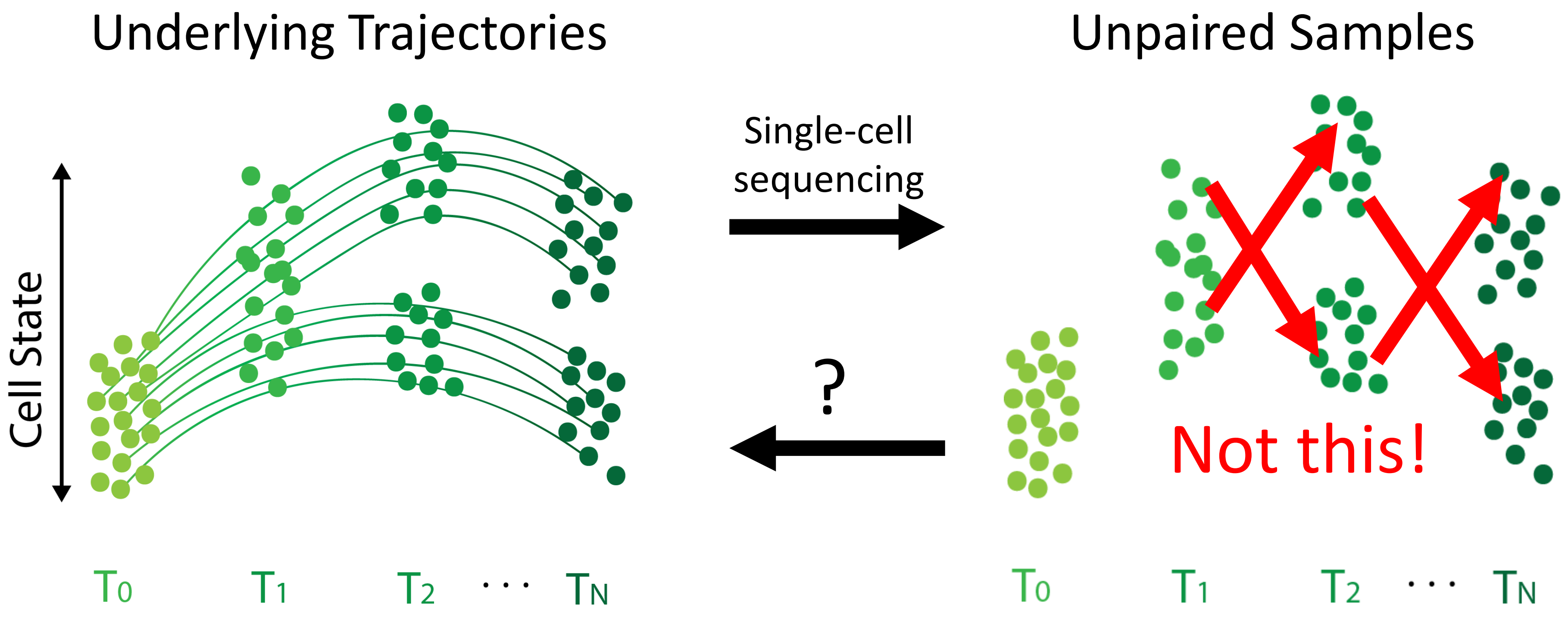
What drives the MET cell state transition in TNBC?




- Learning underlying trajectories from unpaired samples
- Many trajectories for unpaired samples but which should we pick?

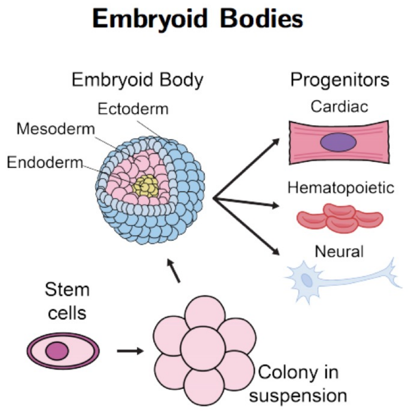
# The general biological problem – Destructive Measurements

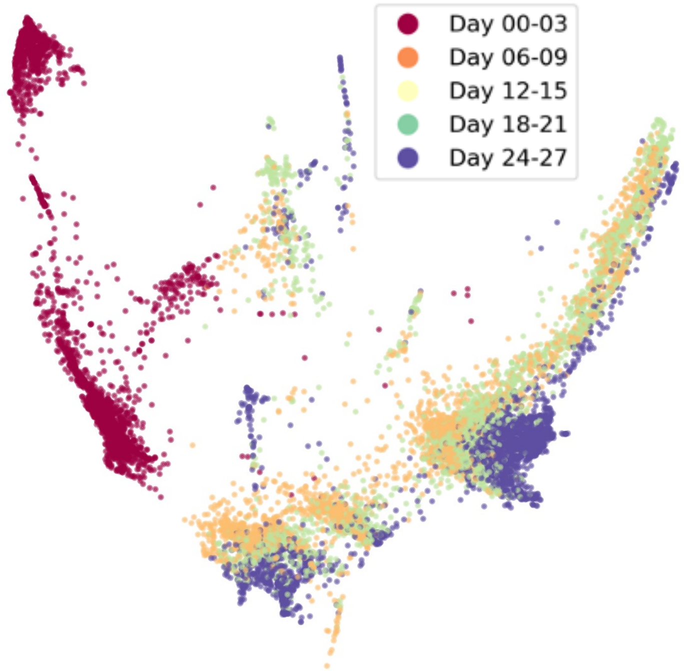
What drives the MET cell state transition in TNBC?

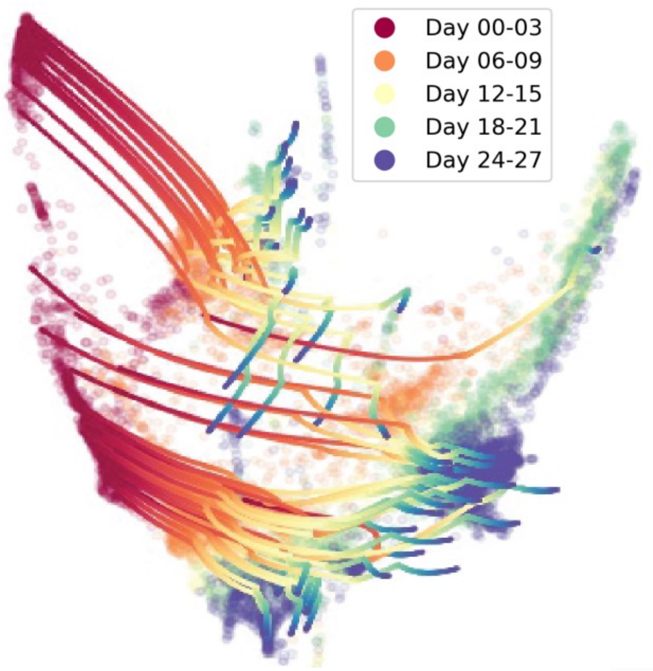


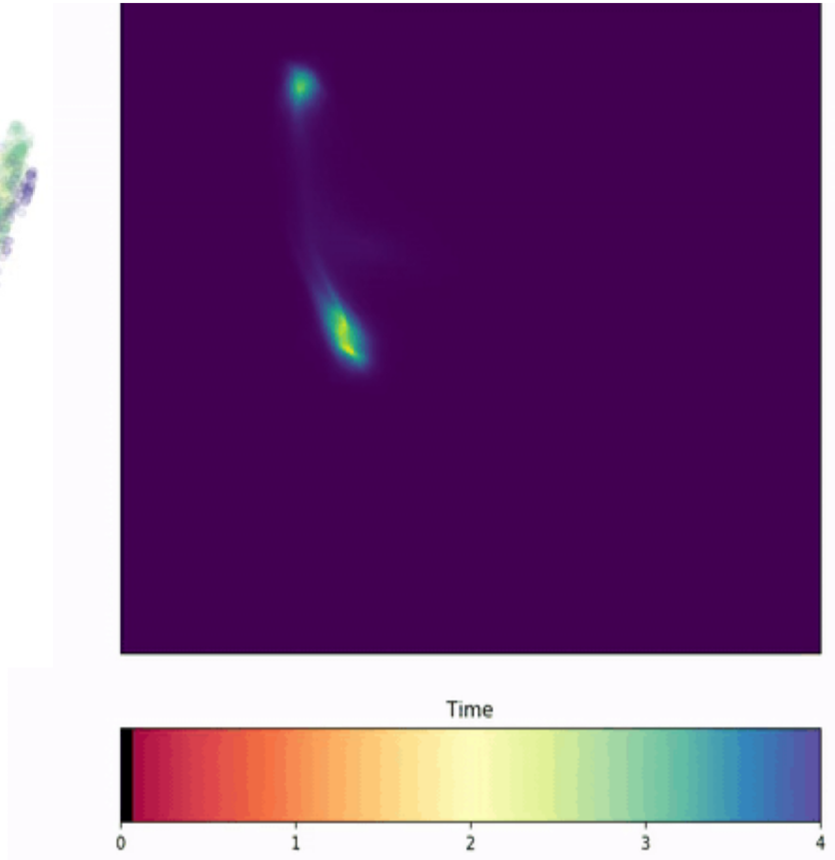

- Learning underlying trajectories from unpaired samples
- Many trajectories for unpaired samples but which should we pick?
- Conjecture: Paths minimize some energy / cost (evolutionary fitness)

# Computational Solution: TrajectoryNet



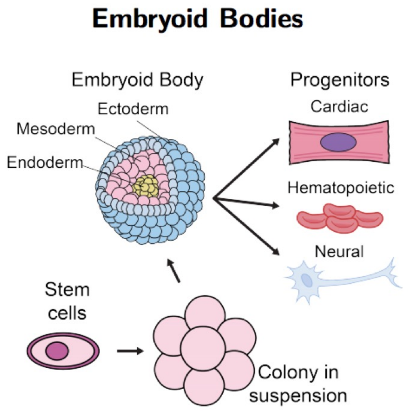



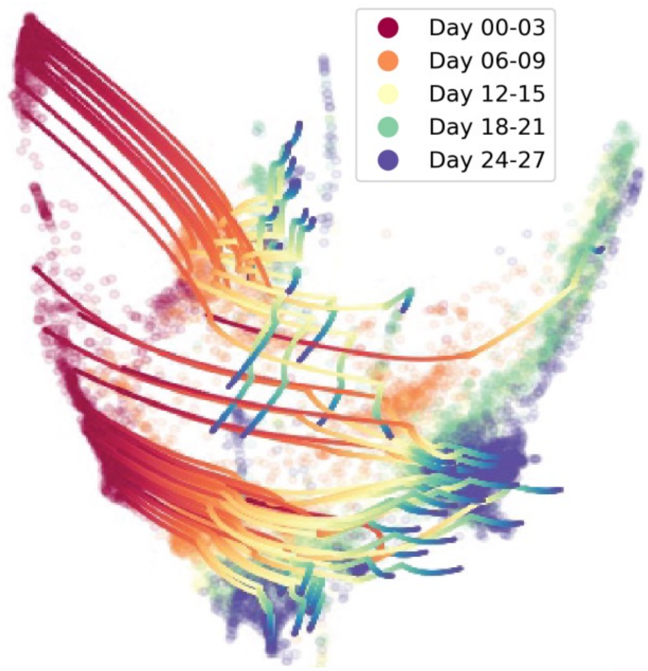

- Dots are cells colored by measurement time
- Lines indicate TrajectoryNet predicted trajectories
- Lighter indicates dense region of cells over time

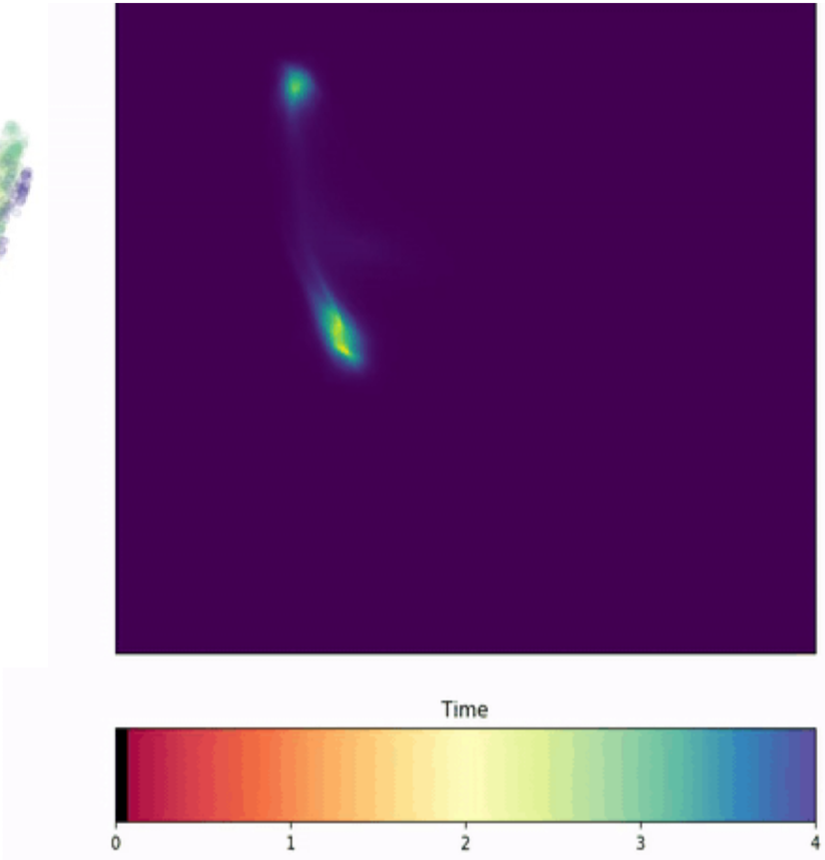
Tong et al. 2020 *ICML*

# Computational Solution: TrajectoryNet




- Recover “optimal” trajectories with respect to some distance
- Can be learned using **Flow Models**




Tong et al. 2020 *ICML*

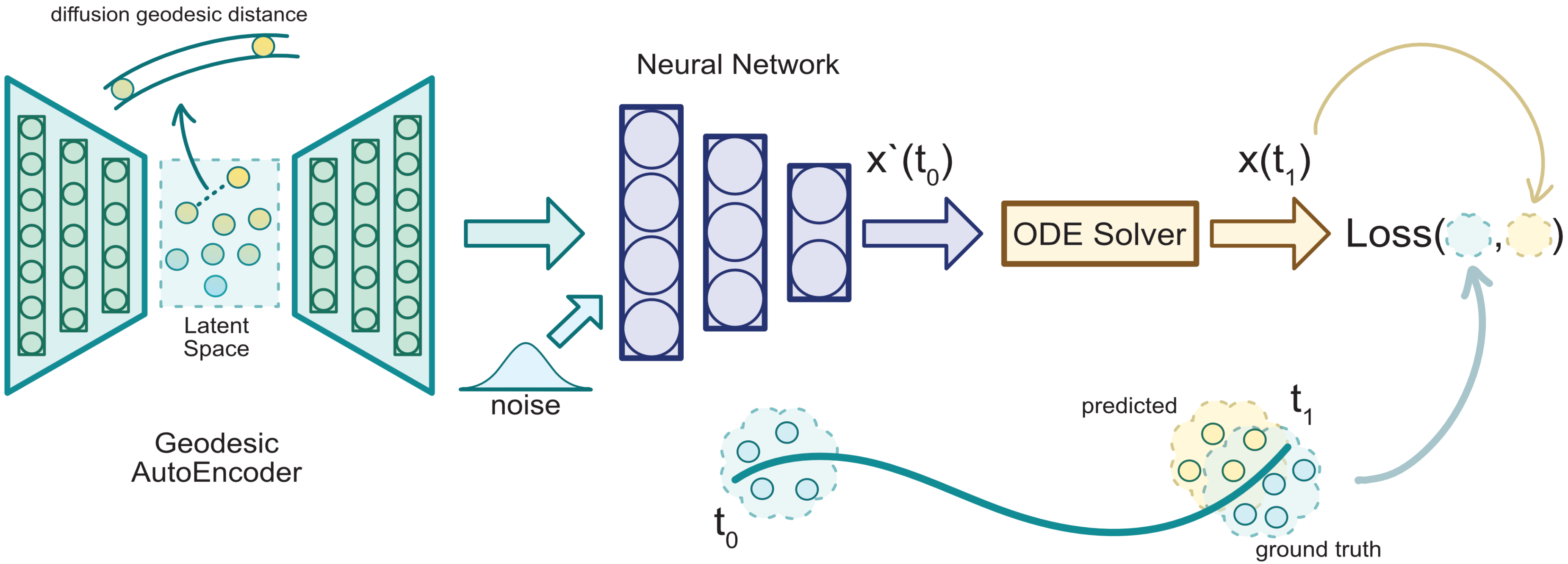
# Continuous normalizing flows for single-cell

**TrajectoryNet (Tong et al. 2020)**

- Maximum likelihood loss
- Ambient space

**MioFlow (Huguet et al. 2022)**

- Particle-based OT loss
- Geodesic Autoencoder

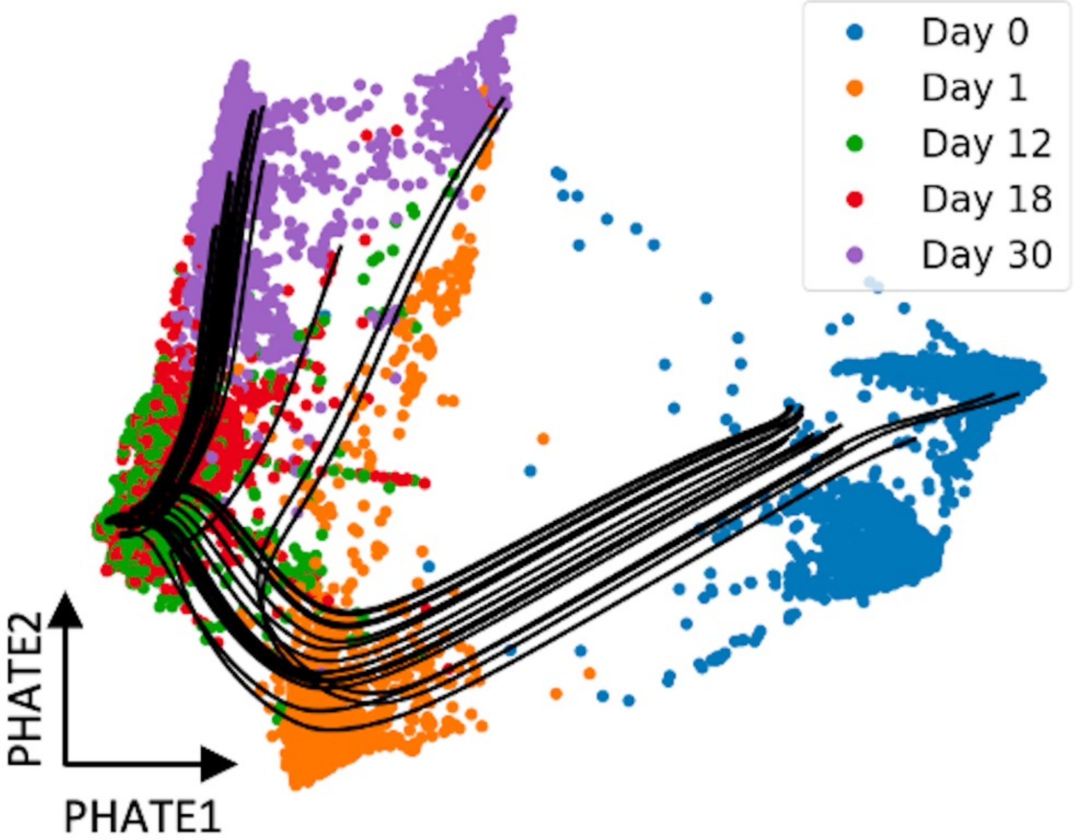


# Applying back to triple negative breast cancer

Q: What drives the MET cell state transition in breast cancer?

Method:

- Learn cell trajectories



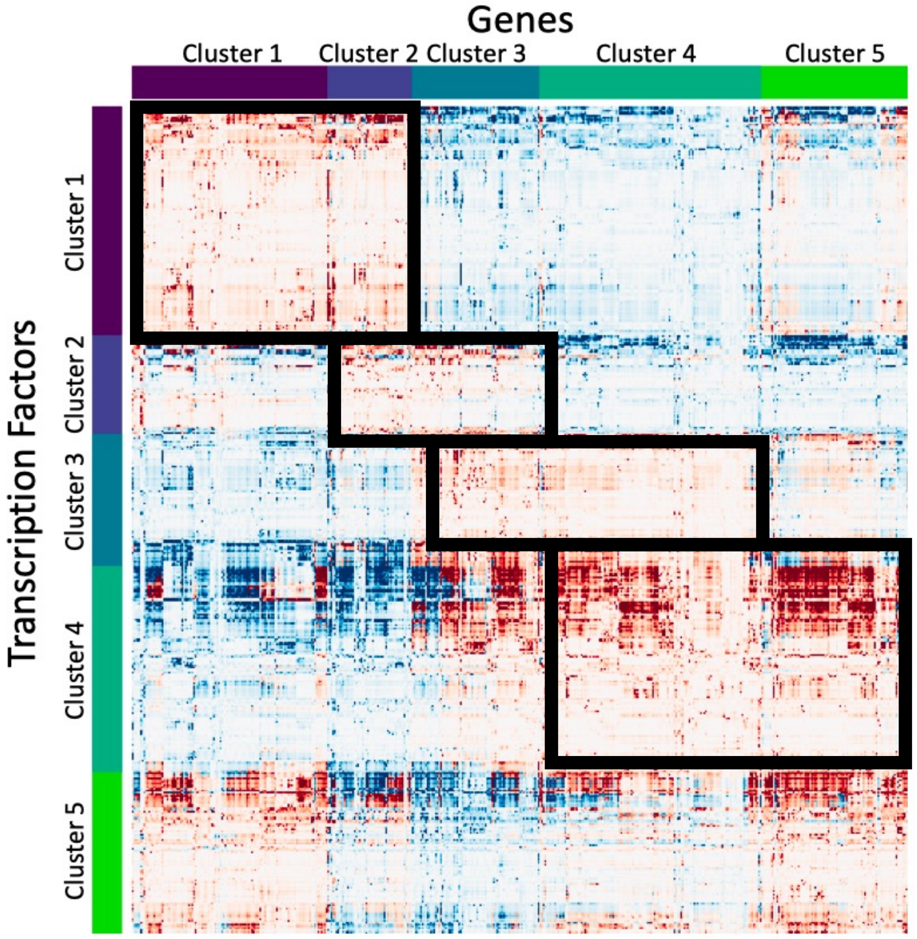

Tong*, Kuchroo*, et al. 2023

# Applying back to triple negative breast cancer

Q: What drives the MET cell state transition in breast cancer?

Method:

- Learn cell trajectories
- Compute a pairwise gene regulatory network using granger causality



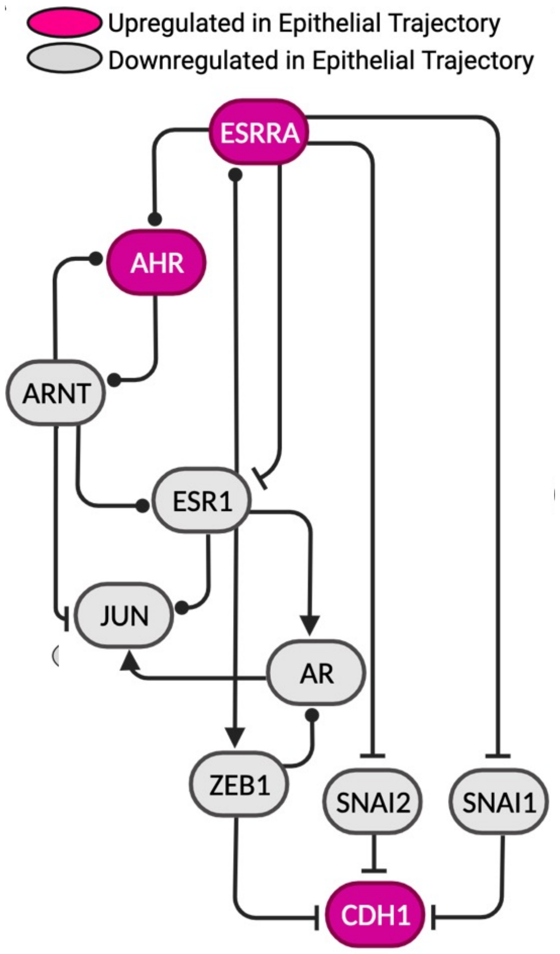

Tong*, Kuchroo*, et al. 2023

# Applying back to triple negative breast cancer

Q: What drives the MET cell state transition in breast cancer?

Method:

- Learn cell trajectories
- Compute a pairwise gene regulatory network using granger causality
- Filter based on prior knowledge




Tong*, Kuchroo*, et al. 2023

# Applying back to triple negative breast cancer

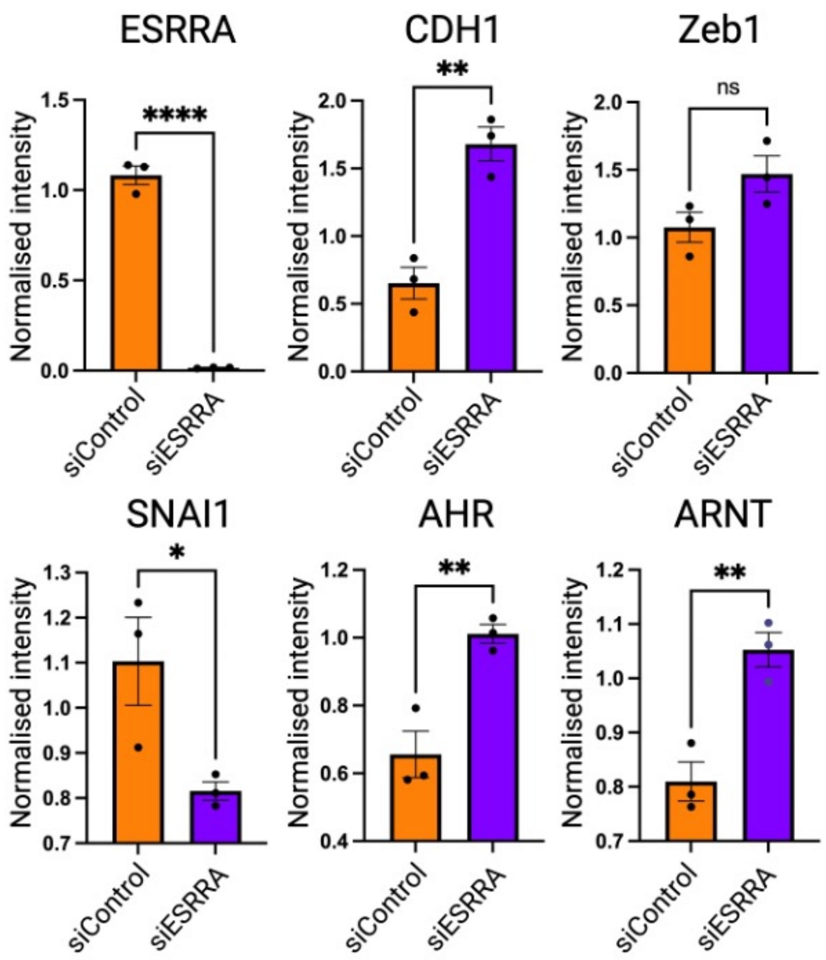
Q: What drives the MET cell state transition in breast cancer?

Method:

- Learn cell trajectories
- Compute a pairwise gene regulatory network using granger causality
- Filter based on prior knowledge

Validation:

- siRNA knockouts




Tong*, Kuchroo*, et al. 2023

# Applying back to triple negative breast cancer

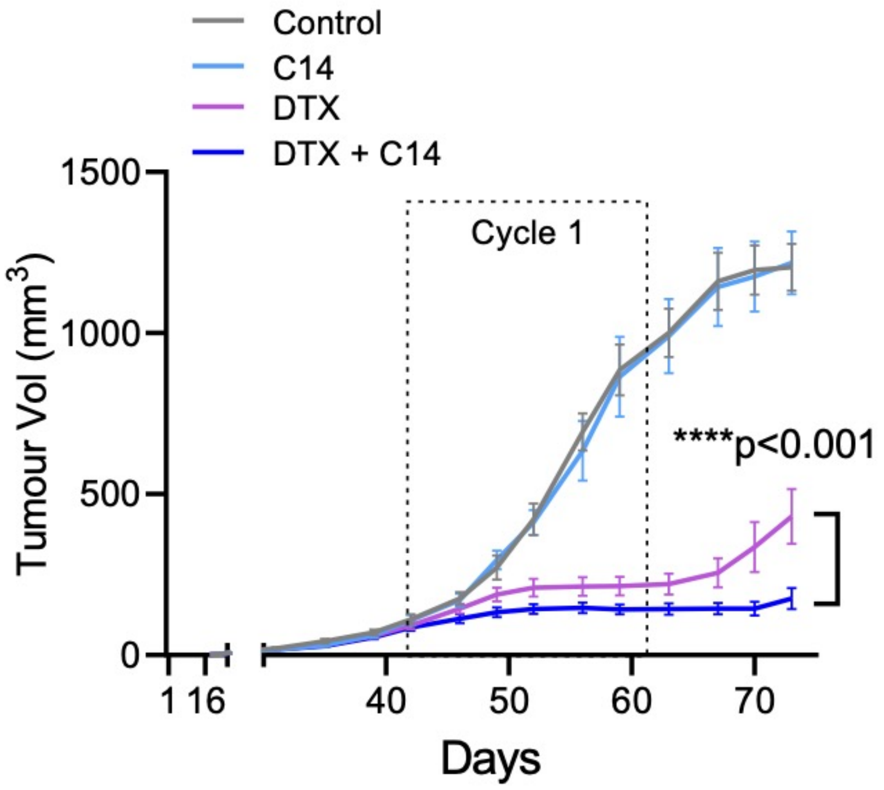
Q: What drives the MET cell state transition in breast cancer?

Method:

- Learn cell trajectories
- Compute a pairwise gene regulatory network using granger causality
- Filter based on prior knowledge

Validation:

- siRNA knockouts
- Knock out ESRRA driver in mice




Tong*, Kuchroo*, et al. 2023

# Applying back to triple negative breast cancer

Q: What drives the MET cell state transition in breast cancer?

Method:

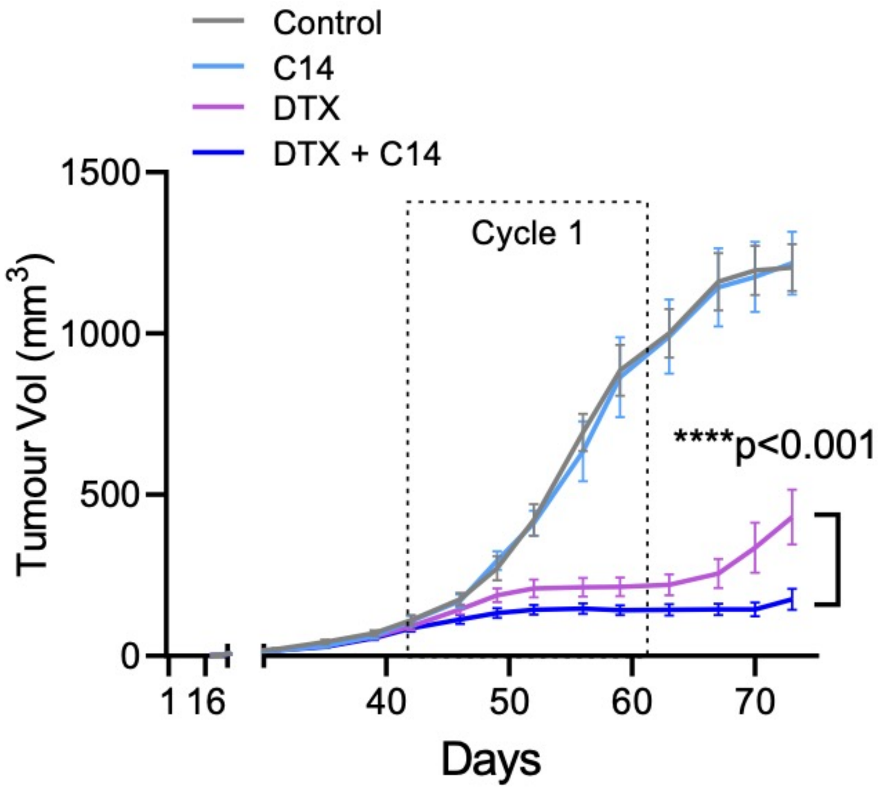
- Learn cell trajectories
- Compute a pairwise gene regulatory network using granger causality
- Filter based on prior knowledge

Validation:

- siRNA knockouts
- Knock out ESRRA driver in mice

Takeaways:

- Reducing ESRRA reduces metastasis in mouse
- TrajectoryNet + Granger causality can discover novel gene regulators




Tong*, Kuchroo*, et al. 2023

# Applications to Biology

**Disease Dynamics**

- Focus: Dynamics of Metastasis
- Biological result: Characterize and disrupt the metastasis of triple negative breast cancer in mouse model
- Computational result: Learn gene regulatory networks from single-cell time series

**Protein Design**

- Focus: Designing de novo binders of difficult targets
- Biological goal: Faster, more effective, biologic drug candidates
- Computational goal: AlphaFold generates only one of many possible structures, we want all of them

**Molecule Sampling (in progress)**

- Focus: Learning to sample the space of valid conformations
- Biological goal: Better sampling of the dynamics of molecules
- Computational goal: Learning flow with access to energy but no data

# Applications to Biology

**Disease Dynamics**

- Focus: Dynamics of Metastasis
- Biological result: Characterize and disrupt the metastasis of triple negative breast cancer in mouse model
- Computational result: Learn gene regulatory networks from single-cell time series

**Protein Design**

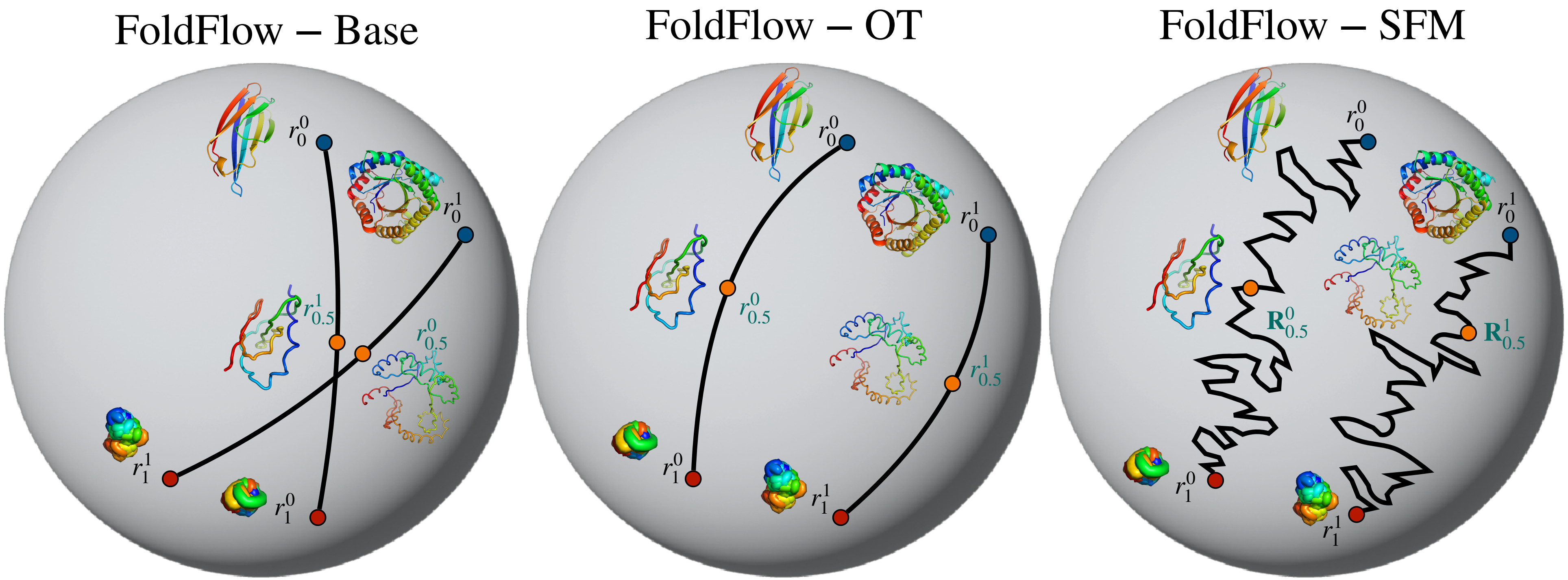
- Focus: Designing de novo binders of difficult targets
- Biological goal: Faster, more effective, biologic drug candidates
- Computational goal: AlphaFold generates only one of many possible structures, we want all of them

**Molecule Sampling (in progress)**

- Focus: Learning to sample the space of valid conformations
- Biological goal: Better sampling of the dynamics of molecules
- Computational goal: Learning flow with access to energy but no data

# FoldFlow – SE(3) Stochastic Flow Matching

## A flow matching method for protein backbone generation




Joint work with the *Dreamfold* team: Bose*, Akhound-Sadegh*, G. Huguet, J. Rector-Brooks, C. Liu, A. Nica, M. Korablyov, M. Bronstein, **A. Tong**

Accepted at ICLR 2024 (spotlight)

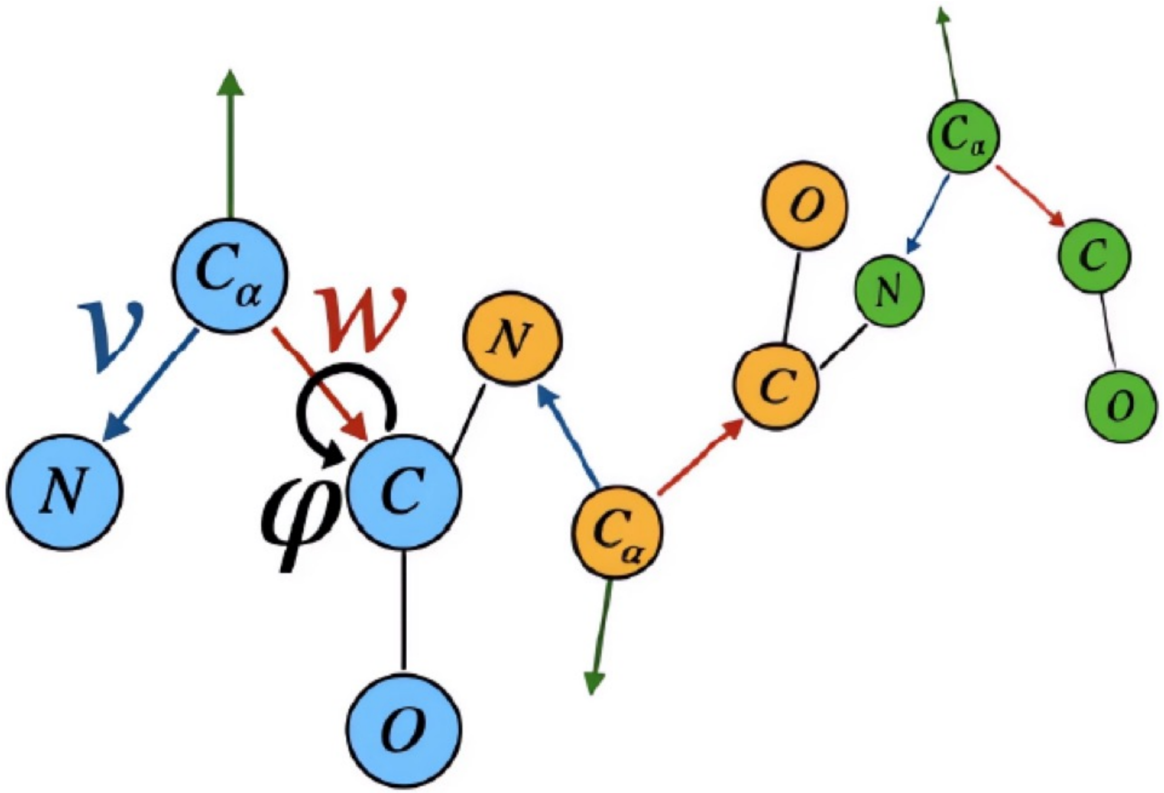
# The Problem: Protein Design

- Biological Problem:
  - Given a protein sequence its structure determines function
  - Given a sequence the number of 3D structures is enormous 10^300 (length 20)
  - Out of all possible folds, most are not stable. We would like to design 3D structures for which there exists a sequence (designable).

- Computational Problem:
  - Generate designable 3D conformations either unconditionally or conditioned on some function / sequence / structure.

# Flows for Proteins

- Build a flow over $SE(3)^N$ i.e. the manifold of N 3D translations and rotations
- Same space as AlphaFold2 folding module uses
- Leverages engineering in diffusion models for a flow model



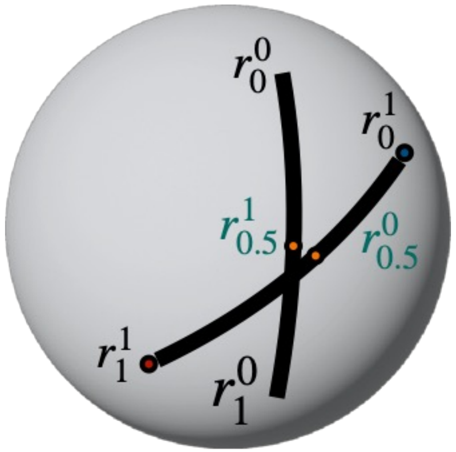
# Flows over SE(3)

- Flows in SE(3) are flows over Translations R(3) and Rotations SO(3)
- Recent models such as RFDiffusion and FrameDiff use the score of the isotropic Gaussian in SO(3) is

$$IGSO_3(\omega, t) = \sum_{l \in \mathcal{N}} (2l+1) e^{-l(l+1)t/2} \frac{\sin((l+1/2)\omega)}{\sin(\omega/2)}$$

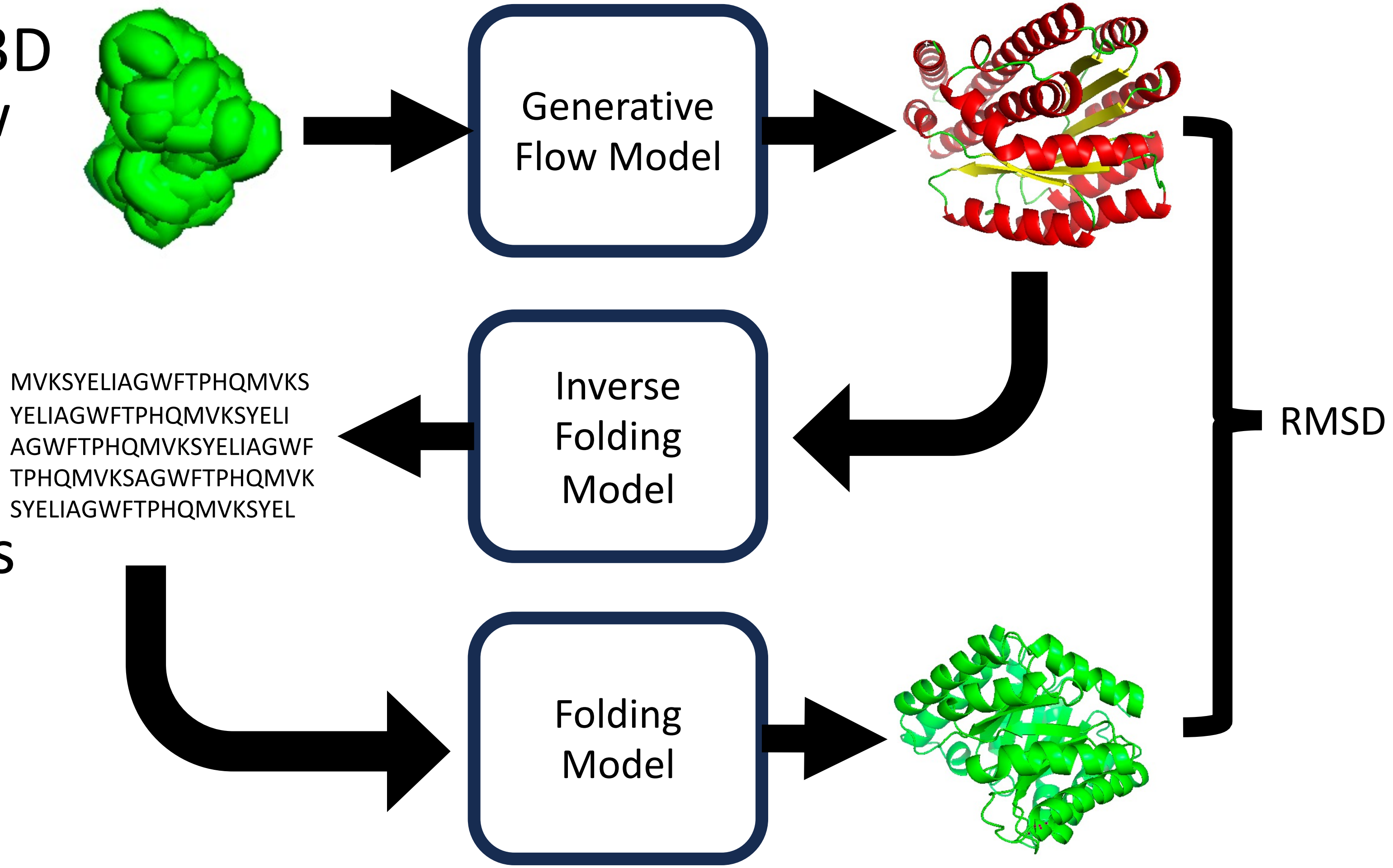
- Instead, only need flows over SO(3)

$$u_t(r_t|z) = \log_{r_t}(r_0)/t$$



# Structure-First Protein Design

1. Design a backbone in 3D using a generative flow model
2. Use Inverse-folding model to find many sequences for that structure
3. Refold those structures to find structures that fold to the generated conformation

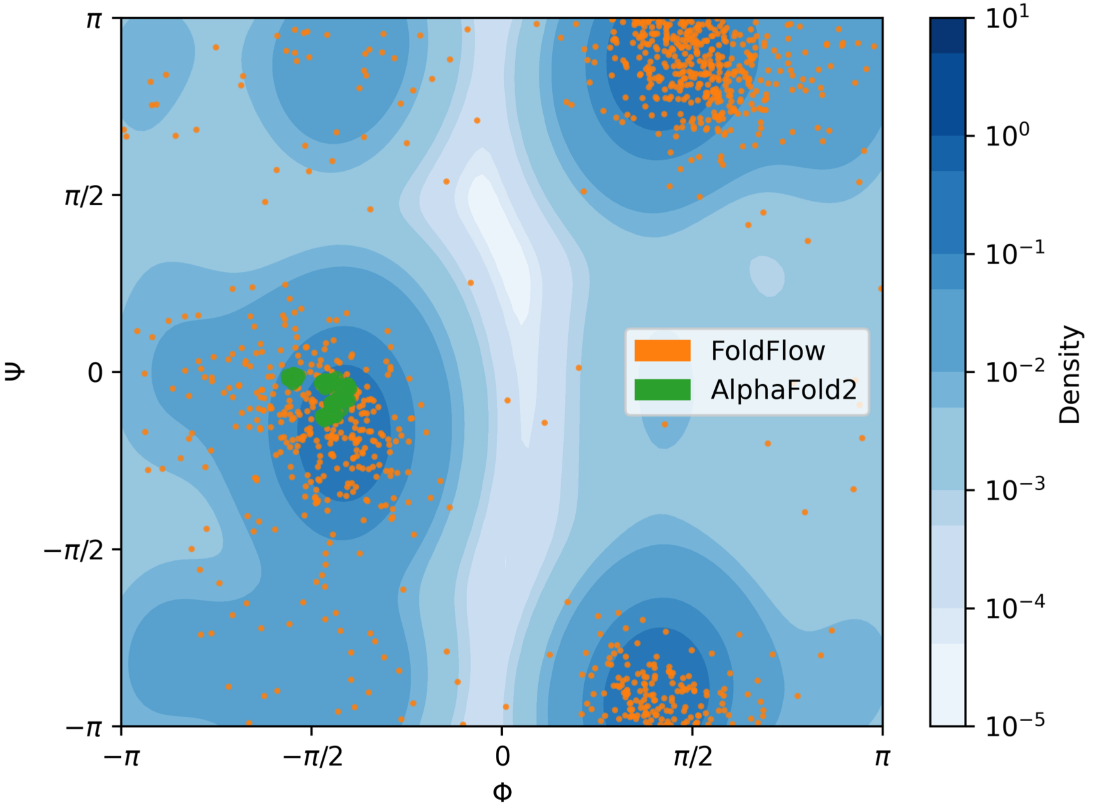


# Evaluation

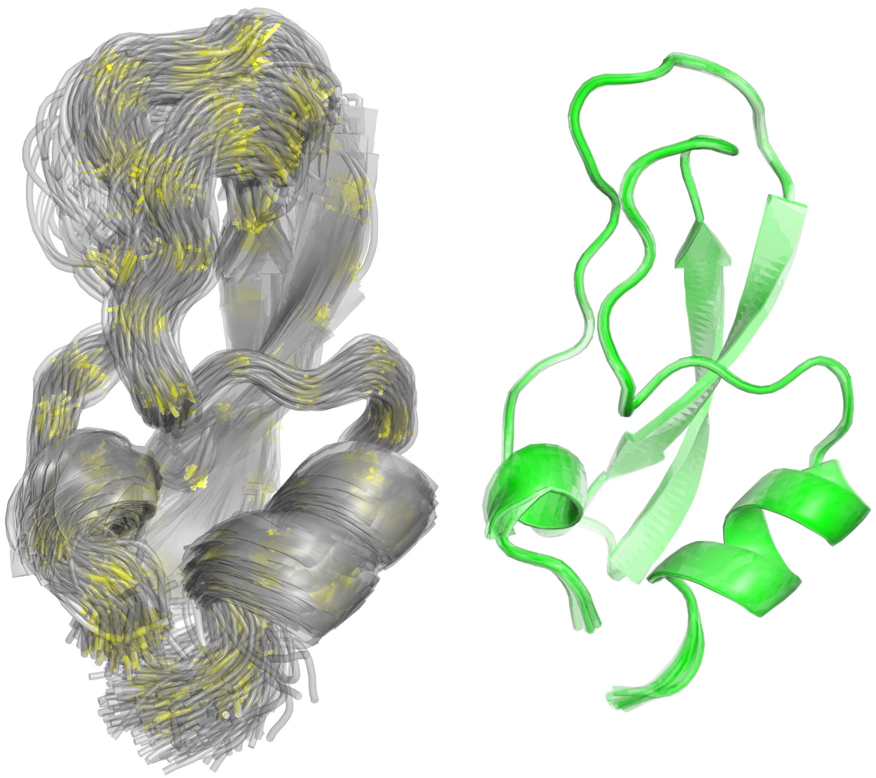
| | Designability | | Novelty | | Diversity (↓) | iters / sec (↑) |
|---|---|---|---|---|---|---|
| | Fraction (↑) | scRMSD (↓) | Fraction (↑) | avg. max TM (↓) | | |
| RFDiffusion | 0.969 ± 0.023 | 0.650 ± 0.136 | *0.708 ± 0.060 | *0.449 ± 0.012 | 0.256 | — |
| Genie | 0.581 ± 0.064 | 2.968 ± 0.344 | *0.556 ± 0.093 | *0.434 ± 0.016 | 0.228 | — |
| FrameDiff-ICML | 0.402 ± 0.062 | 3.885 ± 0.415 | 0.176 ± 0.124 | 0.542 ± 0.046 | 0.237 | — |
| FrameDiff-Improved | 0.555 ± 0.071 | 2.929 ± 0.354 | 0.296 ± 0.112 | 0.457 ± 0.026 | 0.278 | — |
| FrameDiff-Retrained | 0.612 ± 0.060 | 2.990 ± 0.307 | 0.108 ± 0.083 | 0.684 ± 0.032 | 0.403 | 1.278 |
| FoldFlow-Base | 0.657 ± 0.042 | 3.000 ± 0.271 | 0.432 ± 0.074 | 0.452 ± 0.024 | 0.264 | 2.674 |
| FoldFlow-OT | 0.820 ± 0.037 | 1.806 ± 0.249 | 0.484 ± 0.068 | 0.460 ± 0.020 | 0.247 | 2.673 |
| FoldFlow-SFM | 0.716 ± 0.040 | 2.296 ± 0.391 | 0.544 ± 0.061 | 0.411 ± 0.023 | 0.248 | 2.647 |

# Foldflow: Sampling conformations

- **Start from any distribution = use better inductive bias**
- **Learn conformation distribution, starting from a folded structure**




**Foldflow** **AlphaFold2**



# Flows for Protein Backbones

- Results in
  - Simpler code vs. diffusion
  - Faster training
  - Faster inference
- Future work
  - Conditioning on sequence
  - More controllable generation
  - More ideas from generative models

# Applications to Biology

**Disease Dynamics**

- Focus: Dynamics of Metastasis
- Biological result: Characterize and disrupt the metastasis of triple negative breast cancer in mouse model
- Computational result: Learn gene regulatory networks from single-cell time series

**Protein Design**

- Focus: Designing de novo binders of difficult targets
- Biological goal: Faster, more effective, biologic drug candidates
- Computational goal: AlphaFold generates only one of many possible structures, we want all of them

**Molecule Sampling (in progress)**

- Focus: Learning to sample the space of valid conformations
- Biological goal: Better sampling of the dynamics of molecules
- Computational goal: Learning flow with access to energy but no data

# Applications to Biology

**Disease Dynamics**

- Focus: Dynamics of Metastasis
- Biological result: Characterize and disrupt the metastasis of triple negative breast cancer in mouse model
- Computational result: Learn gene regulatory networks from single-cell time series

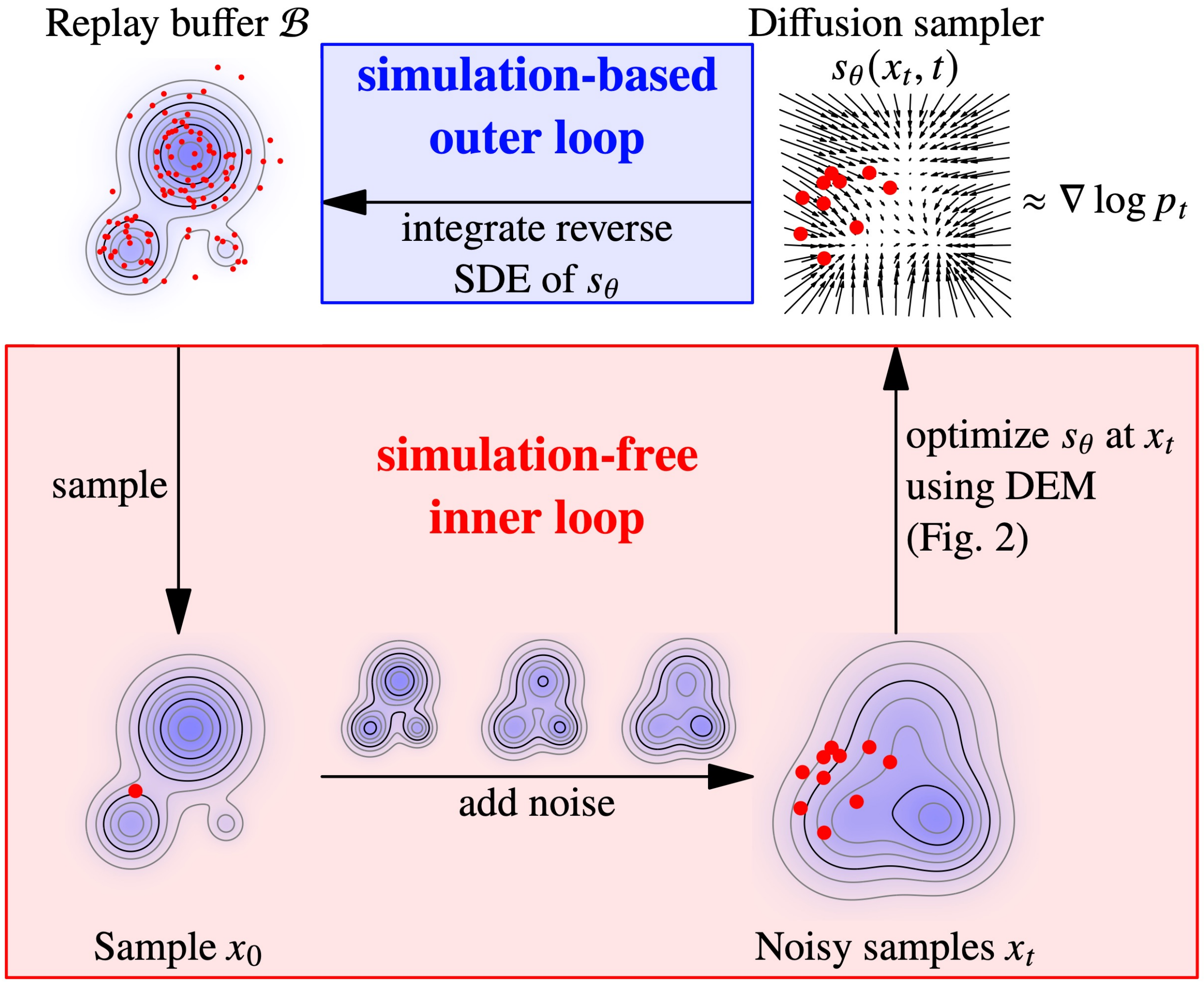
**Protein Design**

- Focus: Designing de novo binders of difficult targets
- Biological goal: Faster, more effective, biologic drug candidates
- Computational goal: AlphaFold generates only one of many possible structures, we want all of them

**Molecule Sampling (in progress)**

- Focus: Learning to sample the space of valid conformations
- Biological goal: Better sampling of the dynamics of molecules
- Computational goal: Learning flow with access to energy but no data

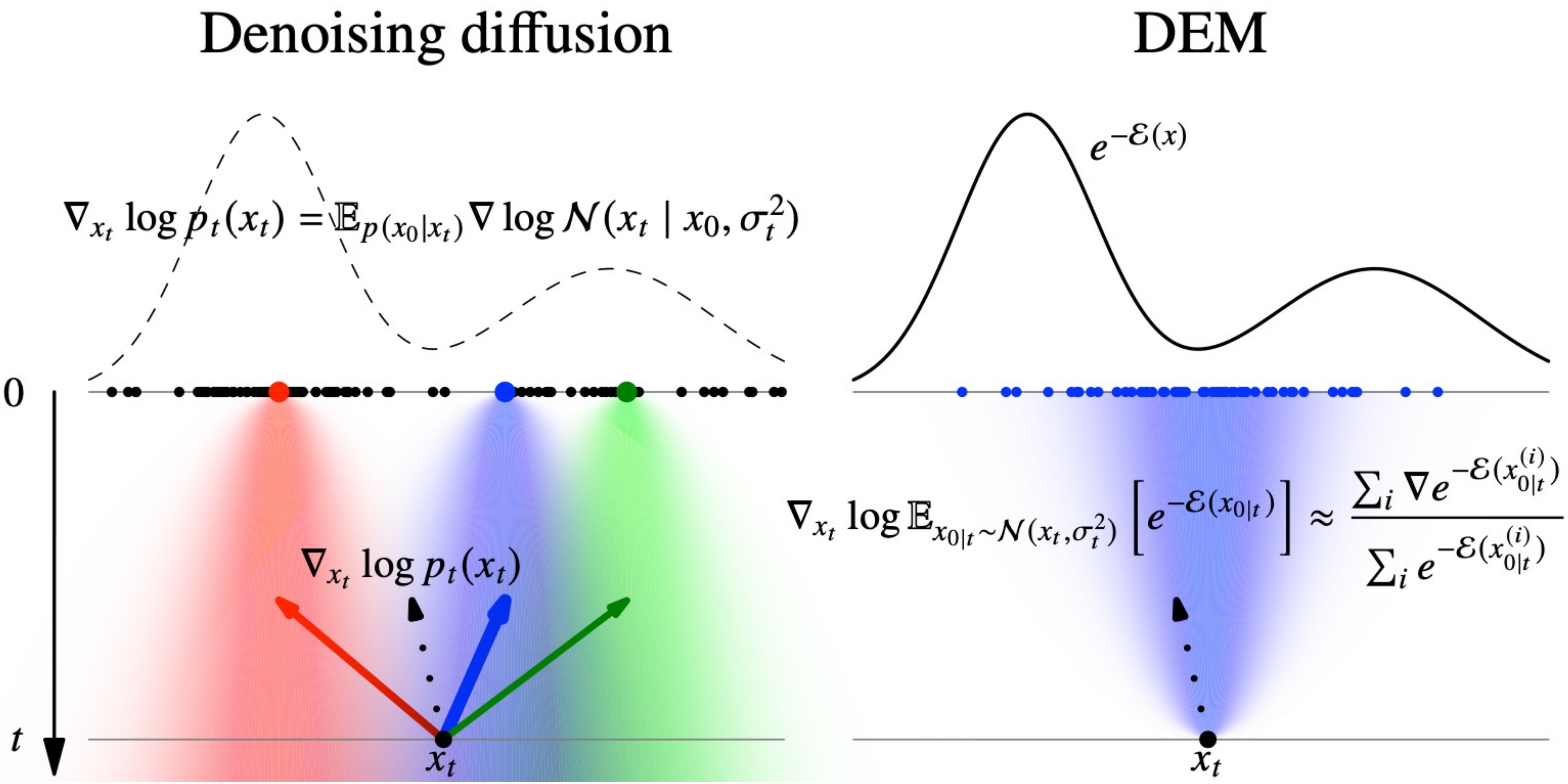
# iDEM: Iterated Denoising Energy Matching




- Goal: Sample proportional to an (unnormalized) energy
- Smooth energy over time using diffusion noising process
- $E_t(x) = \frac{E(x)^{1-t}}{E(x) * N(0, \sigma_t)}$
- First *simulation-free* training method using a matching loss to match the score

Akhound-Sadegh, Rector-Brooks, Bose, … Tong, 2024

# iDEM: Iterated Denoising Energy Matching



# Evaluation

- Faster and more scalable than previous methods
- Still useful to have some simulation for better training

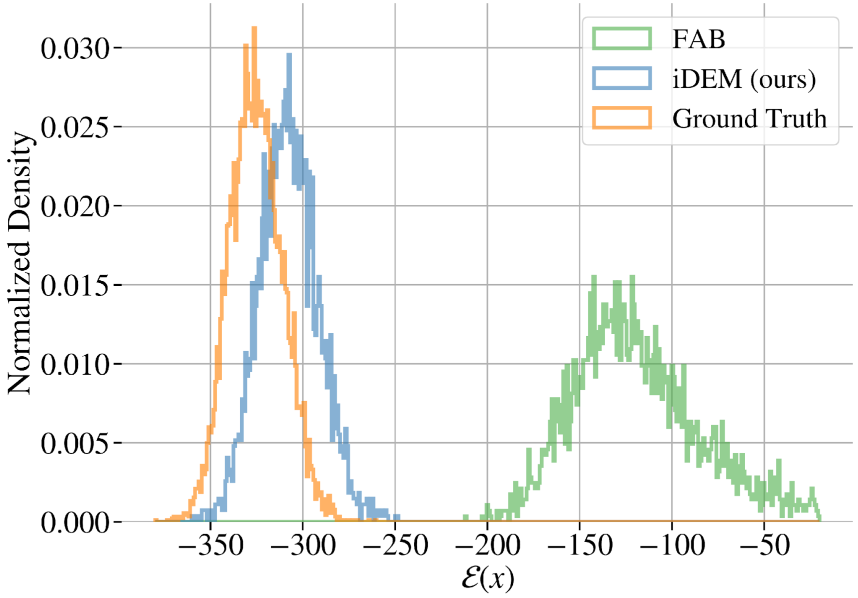
*Table 2.* Sampler performance with mean $\pm$ standard deviation over 3 seeds for negative log-likelihood (NLL), Total Variation (TV), and 2-Wasserstein metrics ($\mathcal{W}_2$). $*$ indicates divergent training. **Bold** via Welch's two sample t-test $p < 0.1$. See §F.2 for more details.

| Energy → | GMM ($d = 2$) | | | DW-4 ($d = 8$) | | | LJ-13 ($d = 39$) | | | LJ-55 ($d = 165$) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Algorithm ↓ | NLL | TV | $\mathcal{W}_2$ | NLL | TV | $\mathcal{W}_2$ | NLL | TV | $\mathcal{W}_2$ | NLL | TV | $\mathcal{W}_2$ |
| FAB (Midgley et al., 2023b) | 7.14±0.01 | 0.88±0.02 | 12.0±5.73 | **7.16**±0.01 | **0.09**±0.00 | 2.15±0.02 | **17.52**±0.17 | **0.04**±0.00 | 4.35±0.01 | 200.32±62.3 | 0.24±0.09 | 18.03±1.21 |
| PIS (Zhang & Chen, 2022) | 7.72±0.03 | 0.92±0.01 | **7.64**±0.92 | 7.19±0.01 | **0.09**±0.00 | **2.13**±0.02 | 47.05±12.46 | 0.25±0.01 | 4.67±0.11 | * | * | * |
| DDS (Vargas et al., 2023) | 7.43±0.46 | **0.82**±0.02 | 9.31±0.82 | 11.27±1.24 | 0.16±0.01 | 2.15±0.04 | * | * | * | * | * | * |
| pDEM (ours) | 7.10±0.02 | **0.82**±0.02 | 12.20±0.14 | 7.44±0.05 | 0.13±0.00 | **2.11**±0.03 | 18.80±0.48 | 0.06±0.02 | **4.21**±0.06 | * | * | * |
| iDEM (ours) | **6.96**±0.07 | **0.82**±0.01 | **7.42**±3.44 | **7.17**±0.00 | **0.10**±0.01 | **2.13**±0.04 | **17.68**±0.14 | **0.04**±0.01 | **4.26**±0.03 | **125.86**±18.03 | **0.09**±0.01 | **16.128**±0.071 |

# DEM: Learning to sample an energy function

- Extensible to incorporate symmetries: we can learn a $SE(3)x\ \mathbb{S}_n$ equivariant score network
- Scalable to high dimensions & much faster

LJ55 (165 dimensions)






T. Akhound Sadegh & J. Rector-Brooks et al., *under review.* arXiv:2402.06121

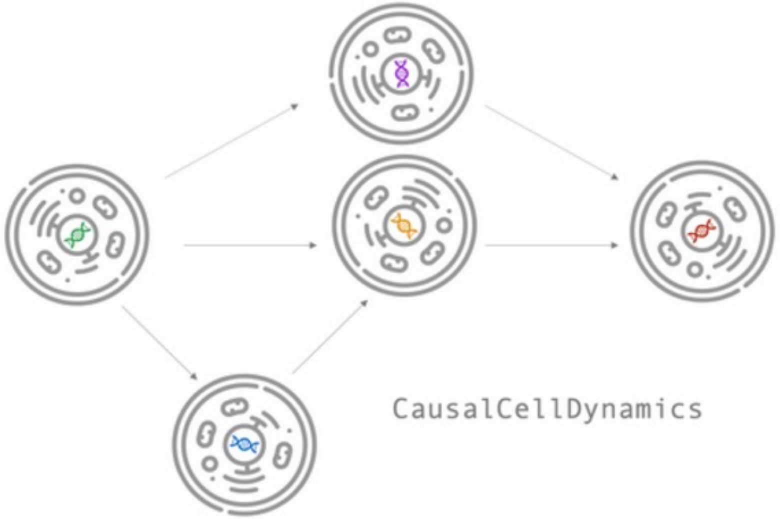
# Why Flow Matching vs. Score Matching?

More general framework:

- Reduced variance in the objective via optimal transport leads to **faster training**
- Straighter inference paths via optimal transport leads to **faster inference**
- Flows are **easier to implement** avoiding defining diffusion on manifolds

# Thank you!

- Bengio Lab
- Krishnaswamy Lab
- RAFALES group
- Theis Lab




CIFAR Mila

Yale Université de Montréal