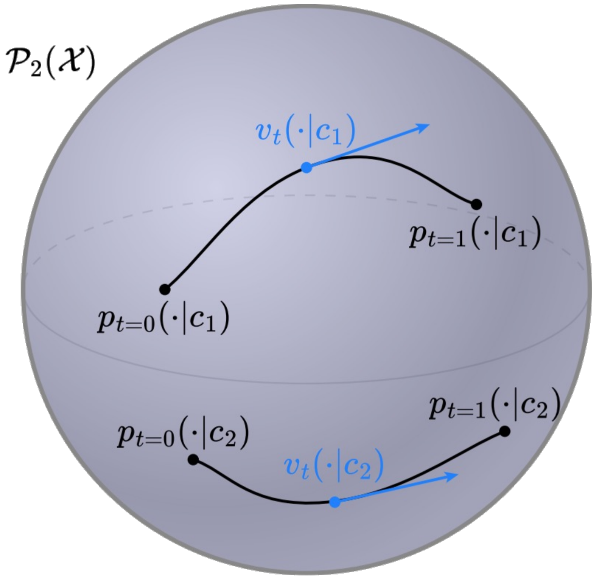
Transport and flows between distributions over distributions

Brandon Amos • Meta, NYC



slides

bamos.github.io/presentations

Generative models and media generation

📚 *Movie Gen: A Cast of Media Foundation Models.* Meta, Oct 2024.

Prompt: *A red-faced monkey with white fur is bathing in a natural hot spring. The monkey is playing in the water with a miniature sail ship in front of it, made of wood with a white sail and a small rudder. The hot spring is surrounded by lush greenery, with rocks and trees.*

Flows: how we got here

(a non-exhaustive list)

(many extensions/applications)

- 2022 Building Normalizing Flows with Stochastic Interpolants
- 2022 Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow
- 2022 Flow Matching for Generative Modeling
- 2021 Score-Based Generative Modeling through Stochastic Differential Equations
- 2021 Denoising Diffusion Implicit Models
- 2020 Improved techniques for training score-based generative models
- 2020 Denoising Diffusion Probabilistic Models
- 2019 Generative modeling by estimating gradients of the data distribution
- 2018 FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models
- 2018 Neural Ordinary Differential Equations
- 2017 Masked autoregressive flow for density estimation
- 2016 Density estimation using real NVP
- 2015 Variational inference with normalizing flows
- 2015 Deep Unsupervised Learning using Nonequilibrium Thermodynamics

Flows: how we got here

(a non-exhaustive list)

(many extensions/applications)

- 2022 Building Normalizing Flows with Stochastic Interpolants
- 2022 Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow
- 2022 Flow Matching for Generative Modeling
- 2021 Score-Based Generative Modeling through Stochastic Differential Equations
- 2021 Denoising Diffusion Implicit Models
- 2020 Improved techniques for training score-based generative models
- 2020 Denoising Diffusion Probabilistic Models
- 2019 Generative modeling by estimating gradients of the data distribution
- 2018 FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models
- 2018 Neural Ordinary Differential Equations
- 2017 Masked autoregressive flow for density estimation
- 2016 Density estimation using real NVP
- 2015 Variational inference with normalizing flows
- 2015 Deep Unsupervised Learning using Nonequilibrium Thermodynamics

$$\underbrace{p_Y(y)}_{\text{(data)}} = \underbrace{p_X}_{\text{(base)}}(f^{-1}(y)) \left| \frac{\partial f^{-1}(y)}{\partial y} \right|$$

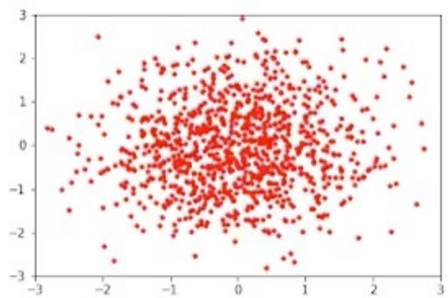
Flows: how we got here

(a non-exhaustive list)

(many extensions/applications)

- 2022 Building Normalizing Flows with Stochastic Interpolants
- 2022 Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow
- 2022 Flow Matching for Generative Modeling
- 2021 Score-Based Generative Modeling through Stochastic Differential Equations
- 2021 Denoising Diffusion Implicit Models
- 2020 Improved techniques for training score-based generative models
- 2020 Denoising Diffusion Probabilistic Models
- 2019 Generative modeling by estimating gradients of the data distribution
- 2018 FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models
- 2018 Neural Ordinary Differential Equations
- 2017 Masked autoregressive flow for density estimation
- 2016 Density estimation using real NVP
- 2015 Variational inference with normalizing flows
- 2015 Deep Unsupervised Learning using Nonequilibrium Thermodynamics

$$\underbrace{p_Y(y)}_{\text{(data)}} = \underbrace{p_X}_{\text{(base)}}(f^{-1}(y)) \left| \frac{\partial f^{-1}(y)}{\partial y} \right|$$



Source: Normalizing Flows in 100 Lines of JAX

parameterize with an invertible function
learn with likelihood

Flows: how we got here

(a non-exhaustive list)

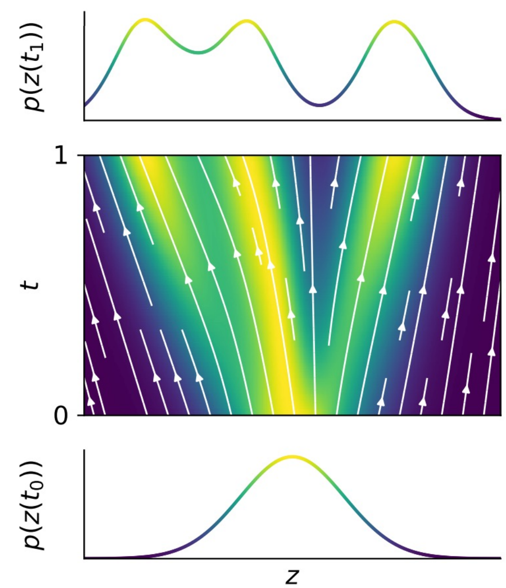
(many extensions/applications)

- 2022 Building Normalizing Flows with Stochastic Interpolants
- 2022 Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow
- 2022 Flow Matching for Generative Modeling
- 2021 Score-Based Generative Modeling through Stochastic Differential Equations
- 2021 Denoising Diffusion Implicit Models
- 2020 Improved techniques for training score-based generative models
- 2020 Denoising Diffusion Probabilistic Models
- 2019 Generative modeling by estimating gradients of the data distribution
- 2018 FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models
- 2018 Neural Ordinary Differential Equations
- 2017 Masked autoregressive flow for density estimation
- 2016 Density estimation using real NVP
- 2015 Variational inference with normalizing flows
- 2015 Deep Unsupervised Learning using Nonequilibrium Thermodynamics

$$\underbrace{p_Y(y)}_{\text{(data)}} = \underbrace{p_X}_{\text{(base)}}(f^{-1}(y)) \left| \frac{\partial f^{-1}(y)}{\partial y} \right|$$

parameterize with an ODE
learn with likelihood

$$\dot{z}_t = g(z_t) \quad z_0 \sim p_X$$



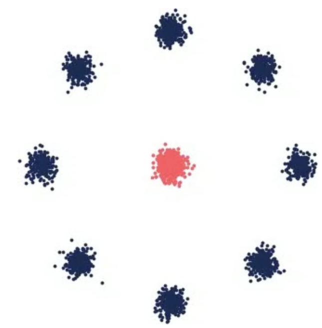
Flows: how we got here

(a non-exhaustive list)

(many extensions/applications)

- 2022 Building Normalizing Flows with Stochastic Interpolants
- 2022 Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow
- 2022 Flow Matching for Generative Modeling
- 2021 Score-Based Generative Modeling through Stochastic Differential Equations
- 2021 Denoising Diffusion Implicit Models
- 2020 Improved techniques for training score-based generative models
- 2020 Denoising Diffusion Probabilistic Models
- 2019 Generative modeling by estimating gradients of the data distribution
- 2018 FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models
- 2018 Neural Ordinary Differential Equations
- 2017 Masked autoregressive flow for density estimation
- 2016 Density estimation using real NVP
- 2015 Variational inference with normalizing flows
- 2015 Deep Unsupervised Learning using Nonequilibrium Thermodynamics

DDPM



reference path SDE known
score decomposes nicely, learn via a decomposition

$$dX(t) = \nabla \log p\big(X(t)\big)dt + 2^{1/2}dB_t$$

$$X_{t+1} = X_t + \epsilon\nabla_x \log p(X_t) + \mathcal{N}(0, 2\epsilon)$$

Flows: how we got here

(a non-exhaustive list)

(many extensions/applications)

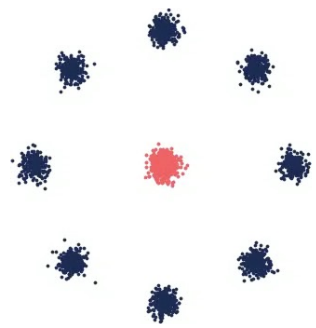
- 2022 Building Normalizing Flows with Stochastic Interpolants
- 2022 Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow
- 2022 Flow Matching for Generative Modeling
- 2021 Score-Based Generative Modeling through Stochastic Differential Equations
- 2021 Denoising Diffusion Implicit Models
- 2020 Improved techniques for training score-based generative models
- 2020 Denoising Diffusion Probabilistic Models
- 2019 Generative modeling by estimating gradients of the data distribution
- 2018 FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models
- 2018 Neural Ordinary Differential Equations
- 2017 Masked autoregressive flow for density estimation
- 2016 Density estimation using real NVP
- 2015 Variational inference with normalizing flows
- 2015 Deep Unsupervised Learning using Nonequilibrium Thermodynamics

$$\underbrace{p_Y(y)}_{\text{(data)}} = \underbrace{p_X}_{\text{(base)}}(f^{-1}(y)) \left| \frac{\partial f^{-1}(y)}{\partial y} \right|$$

back to an ODE/flow with the same marginals!!

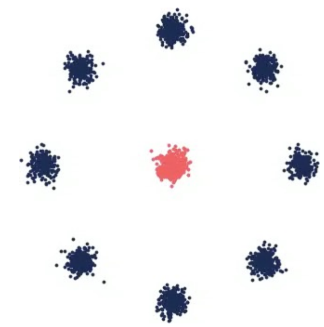
(probability flow ODE, implicit models)

DDPM



$$dX(t) = \nabla \log p\big(X(t)\big)dt + 2^{1/2}dB_t$$

DDIM



$$\dot{z}_t = g(z_t) \quad z_0 \sim p_X$$

Flows: how we got here

(a non-exhaustive list)

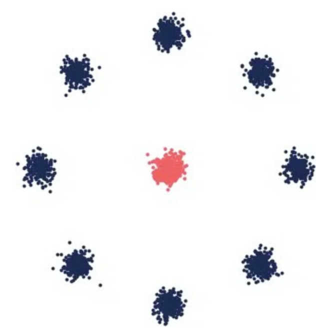
(many extensions/applications)

- 2022 Building Normalizing Flows with Stochastic Interpolants
- 2022 Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow
- 2022 Flow Matching for Generative Modeling
- 2021 Score-Based Generative Modeling through Stochastic Differential Equations
- 2021 Denoising Diffusion Implicit Models
- 2020 Improved techniques for training score-based generative models
- 2020 Denoising Diffusion Probabilistic Models
- 2019 Generative modeling by estimating gradients of the data distribution
- 2018 FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models
- 2018 Neural Ordinary Differential Equations
- 2017 Masked autoregressive flow for density estimation
- 2016 Density estimation using real NVP
- 2015 Variational inference with normalizing flows
- 2015 Deep Unsupervised Learning using Nonequilibrium Thermodynamics

Match the ODE directly, generalize diffusion path

Still use a ground-truth reference path (or interpolant)
Parameterize flow with an unconstrained neural network
(invertibility comes because the reference transport is invertible!!)

$$\underbrace{p_Y(y)}_{\text{(data)}} = \underbrace{p_X}_{\text{(base)}}(f^{-1}(y)) \left| \frac{\partial f^{-1}(y)}{\partial y} \right|$$



$$\dot{z}_t = g(z_t) \quad z_0 \sim p_X$$

Flows: how we got here

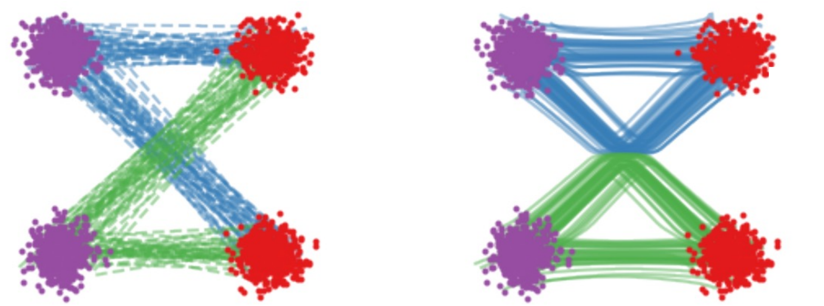
(a non-exhaustive list)

(many extensions/applications)

- 2022 Building Normalizing Flows with Stochastic Interpolants
- 2022 Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow
- 2022 Flow Matching for Generative Modeling
- 2021 Score-Based Generative Modeling through Stochastic Differential Equations
- 2021 Denoising Diffusion Implicit Models
- 2020 Improved techniques for training score-based generative models
- 2020 Denoising Diffusion Probabilistic Models
- 2019 Generative modeling by estimating gradients of the data distribution
- 2018 FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models
- 2018 Neural Ordinary Differential Equations
- 2017 Masked autoregressive flow for density estimation
- 2016 Density estimation using real NVP
- 2015 Variational inference with normalizing flows
- 2015 Deep Unsupervised Learning using Nonequilibrium Thermodynamics

Match the ODE directly, generalize diffusion path

Still use a ground-truth reference path (or interpolant)
Parameterize flow with an unconstrained neural network
(invertibility comes because the reference path is invertible!)



(a) Linear interpolation $X_t = tX_1 + (1-t)X_0$

(b) Rectified flow Z_t induced by (X_0, X_1)

What is optimal transport?

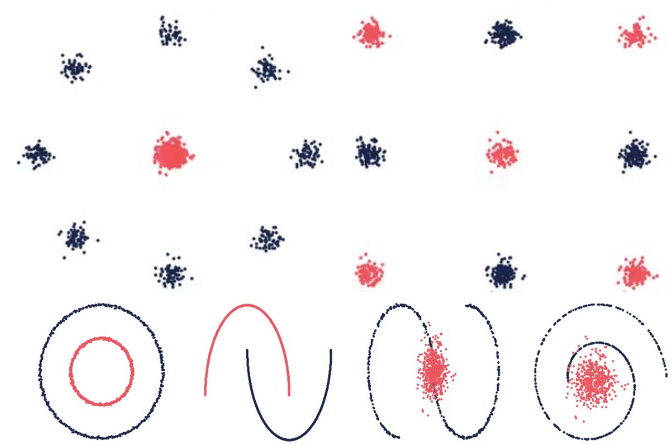
another way of **connecting probability measures**

- *Optimal transport: old and new.* Villani, 2009.
- *Optimal Transport in Learning, Control, and Dynamical Systems.* Bunne and Cuturi, ICML 2023 Tutorial.
- *Computational Optimal Transport*. Peyré and Cuturi, Foundations and Trends in Machine Learning, 2019.
- *Optimal Transport for Applied Mathematicians.* Santambrogio, Birkhäuser, 2015
- *Optimal Transport in Systems and Control.* Chen, Georgiou, and Pavon, Annual Review of Control, Robotics, and Autonomous Systems, 2021.
- *Optimal mass transport: Signal processing and machine-learning applications.* Kolouri et al., 2017.

Monge's problem (squared Euclidean)

$$\boxed{\inf_{T \in \mathcal{T}(\alpha,\beta)} \int_{\mathcal{X}} \|T(x) - x\|_2^2 \mathrm{d}\alpha(x)}$$

find a map connecting α and β that minimally displaces mass

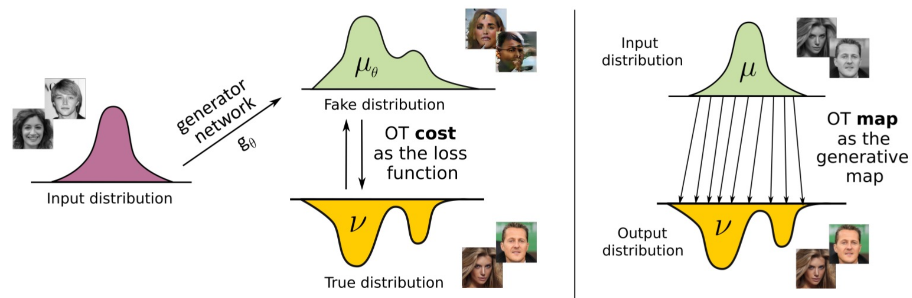


On amortizing convex conjugates for optimal transport. Amos, ICLR 2023.

Why optimal transport? (selected ML-focused highlights)

Defines a **metric on the space of measures**
(metricizes the space of weak convergence)

- *Wasserstein GAN.* Arjovsky, Chintala, Bottou, ICML 2017.
- *Generalized sliced Wasserstein distances.* Kolouri et al., NeurIPS 2019.
- *Sliced wasserstein distance for learning GMMs.* Kolouri et al., CVPR 2018.
- *Convolutional Wasserstein Distances on Geometric Domains*. Solomon et al., ToG 2015.



(a) OT cost as the loss for the generative model. (b) OT map as the generative model.

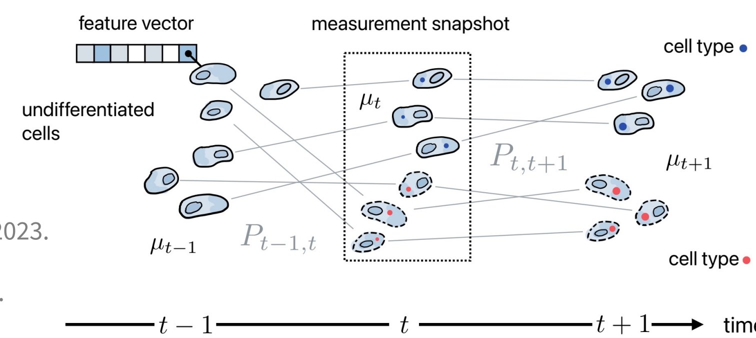
source: Generative Modeling with Optimal Transport Maps by Rout et al.

Couples measures without pairwise data
(e.g., for generative modeling, domain adaptation)

- *Generative modeling via OT maps.* Rout, Korotin, Burnaev. ICLR 2022.
- *Neural Optimal Transport.* Korotin et al., ICLR 2023
- *Neural Monge map estimation*. Jiaojiao Fan et al., TMLR 2023.
- *Joint distribution optimal transportation for domain adaptation.* Courty et al., NeurIPS 2017.
- *Geometric Dataset Distances via Optimal Transport.* Alvarez-Melis et al., NeurIPS 2020.

Finds interpolating paths between populations
(e.g., for cell populations or multi-agent systems)

- *Optimal-transport analysis of single-cell gene expression.* Schiebinger et al., Cell 2019.
- *Learning single-cell perturbation responses using neural optimal transport*. Bunne et al., Nature Methods 2023.
- *Likelihood Training of Schrödinger Bridge.* Liu, Horng, Theodorou. ICLR 2022.
- *Trajectorynet: A dynamic optimal transport network for modeling cellular dynamics.* Tong et al., ICML 2020.



source: Bunne and Cuturi

Flows = "suboptimal" transport

(many extensions/applications)

2022 Building Normalizing Flows with Stochastic Interpolants

2022 Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow

2022 Flow Matching for Generative Modeling

2021 Score-Based Generative Modeling through Stochastic Differential Equations

2021 Denoising Diffusion Implicit Models

2020 Improved techniques for training score-based generative models

2020 Denoising Diffusion Probabilistic Models

2019 Generative modeling by estimating gradients of the data distribution

2018 FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models

2018 Neural Ordinary Differential Equations

2017 Masked autoregressive flow for density estimation

2016 Density estimation using real NVP

2015 Variational inference with normalizing flows

2015 Deep Unsupervised Learning using Nonequilibrium Thermodynamics

$$\underbrace{p_Y}_{\text{(data)}}(y) = \underbrace{p_X}_{\text{(base)}}(f^{-1}(y)) \left| \frac{\partial f^{-1}(y)}{\partial y} \right|$$

Flows "just" care about **matching samples** from the data

Transport path usually chosen to be easy and decomposable

Optimal transport requires searching over the entire transport space

Challenging optimization problem, no nice decompositions

Monge's problem (squared Euclidean)

$$\boxed{\underbrace{\inf_{T\in\mathcal{T}(\alpha,\beta)}}_{\text{find a map connecting } \alpha \text{ and } \beta} \int_{\mathcal{X}} \underbrace{\|T(x)-x\|_2^2}_{\text{minimally displaces mass}} \mathrm{d}\alpha(x)}$$

find a map connecting α and β that minimally displaces mass

Flows = “suboptimal” transport

(many extensions/applications)

- 2022 Building Normalizing Flows with Stochastic Interpolants
- 2022 Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow
- 2022 Flow Matching for Generative Modeling
- 2021 Score-Based Generative Modeling through Stochastic Differential Equations
- 2021 Denoising Diffusion Implicit Models
- 2020 Improved techniques for training score-based generative models
- 2020 Denoisi
- 2019 Generat
- 2018 FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models
- 2018 Neural Ordinary Differential Equations
- 2017 Masked autoregressive flow for density estimation
- 2016 Density estimation using real NVP
- 2015 Variational inference with normalizing flows
- 2015 Deep Unsupervised Learning using Nonequilibrium Thermodynamics

$$\underbrace{p_Y(y)}_{\text{(data)}} = \underbrace{p_X}_{\text{(base)}}(f^{-1}(y)) \left| \frac{\partial f^{-1}(y)}{\partial y} \right|$$

Flows “just” care about **matching samples** from the data

Transport path usually chosen to b

all of this: single distribution to single distribution

g over the entire transport space

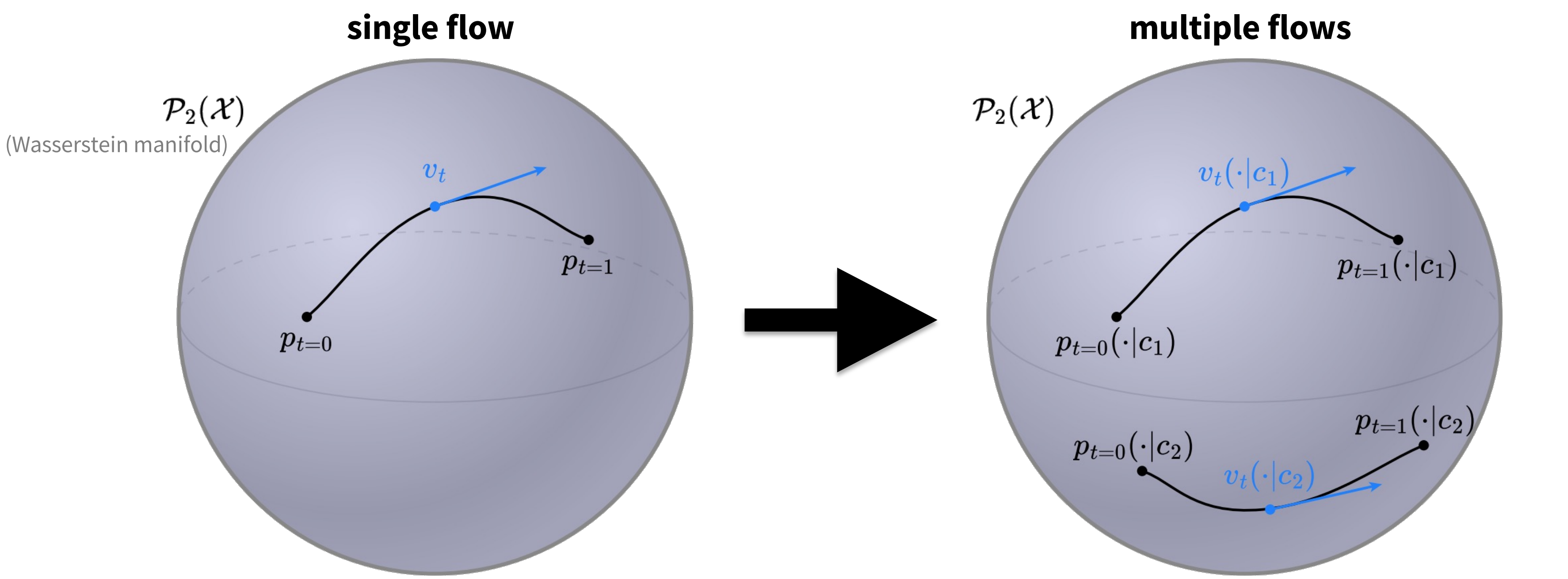
challenging optimization problem, no nice decompositions

Monge’s problem (squared Euclidean)

$$\boxed{\inf_{T \in \mathcal{T}(\alpha,\beta)} \int_{\mathcal{X}} \|T(x) - x\|_2^2 \mathrm{d}\alpha(x)}$$

find a map connecting α and β that minimally displaces mass

To distributions over distributions



$p_{t=0}$ and $p_{t=1}$ — distributions

$p(\cdot|c)$ — a distribution parameterized by c
$p(c)$ — a distribution over distributions

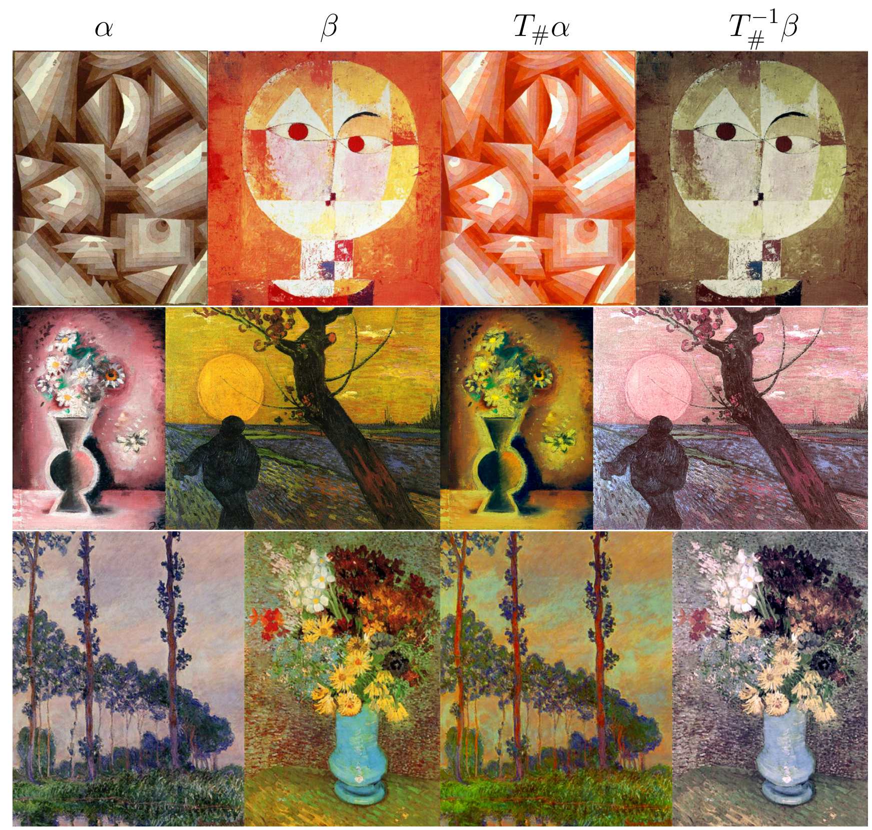
Why distributions over distributions?

1. **Text to image, video, or other media**
   between many text prompts

Why distributions over distributions?

1. **Text to image, video, or other media**
   between many text prompts

2. **Image editing**
   between many pairs of images



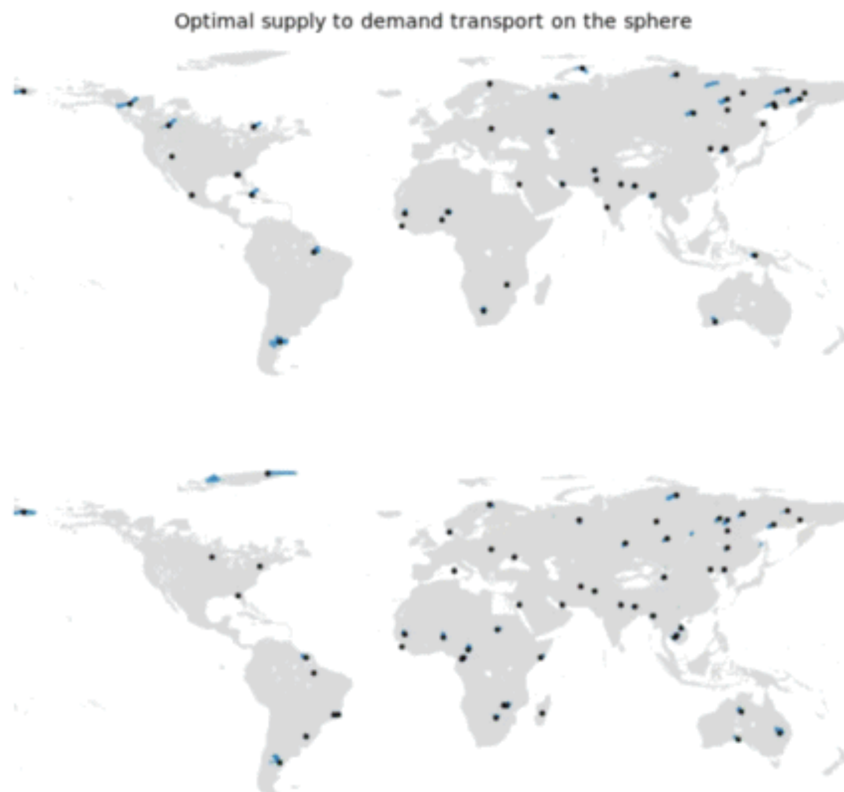
📚 *Meta Optimal Transport*. Amos et al., ICML 2023.

Why distributions over distributions?

1. **Text to image, video, or other media**
   between many text prompts

2. **Image editing**
   between many pairs of images

3. **Scheduling and supply-demand allocations**
   between many initial conditions



📚 *Meta Optimal Transport*. Amos et al., ICML 2023.

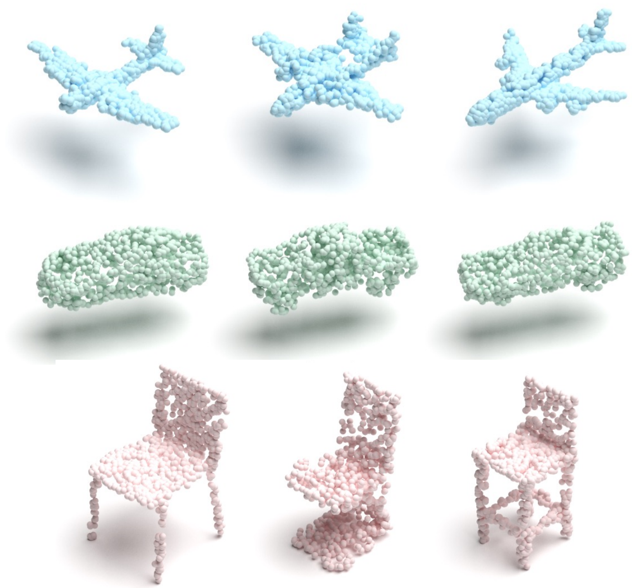
Why distributions over distributions?

1. **Text to image, video, or other media**
   between many text prompts

2. **Image editing**
   between many pairs of images

3. **Scheduling and supply-demand allocations**
   between many initial conditions

4. **Point cloud generation**
   each point cloud is an empirical distribution



📚 *Wasserstein Flow Matching.* Haviv*, Pooladian*, Pe'er, Amos. 2024.

Why distributions over distributions?

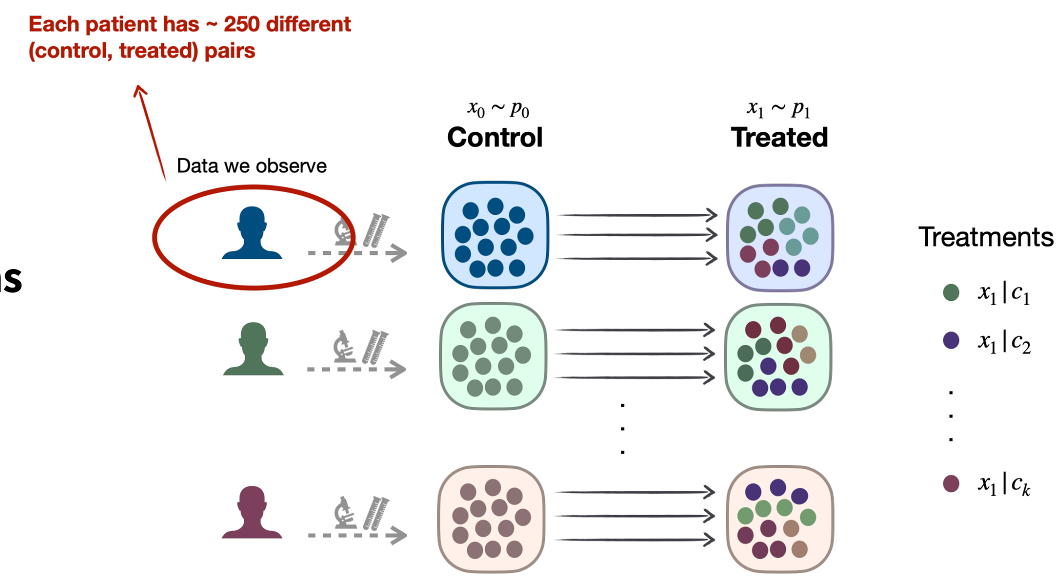
1. **Text to image, video, or other media**
   between many text prompts

2. **Image editing**
   between many pairs of images

3. **Scheduling and supply-demand allocations**
   between many initial conditions

4. **Point cloud generation**
   each point cloud is an empirical distribution

5. **Cellular transport**
   many pairs of untreated to treated populations

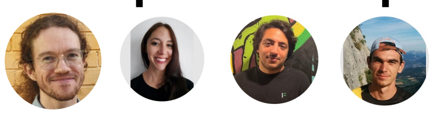


📚 *Meta Flow Matching.* Atanackovic et al., 2024.

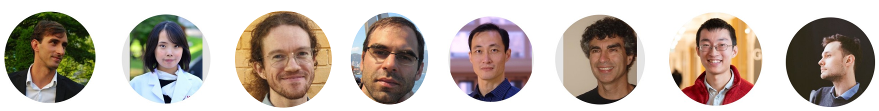
Talk overview

- Primer on **amortized optimization** [Foundations and Trends in ML, 2023]

- **Meta Optimal Transport** [ICML 2023]



- **Meta Flow Matching** [2024]



- **Wasserstein Flow Matching** [2024]

*also referred to as *learned* optimization

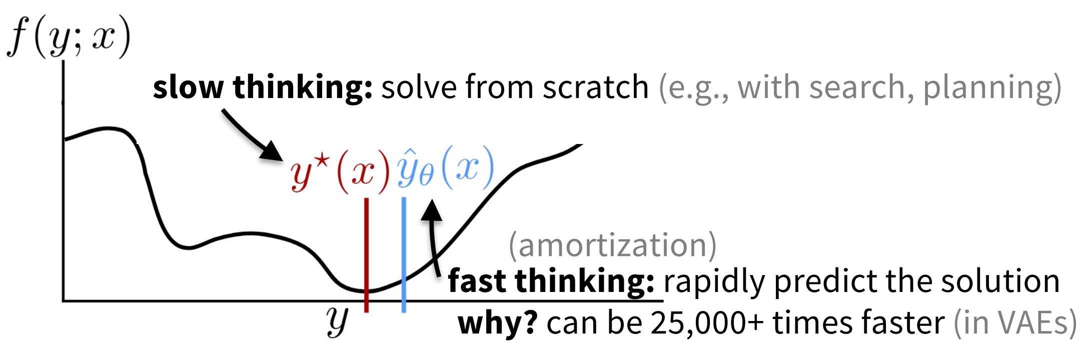
A crash course on amortized optimization

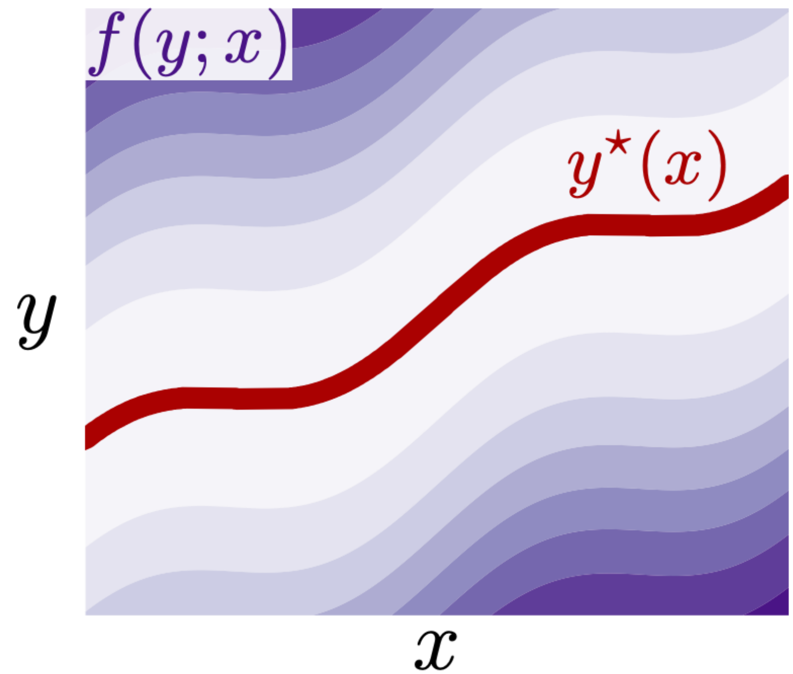
📚 *Tutorial on amortized optimization*. Amos, Foundations and Trends in Machine Learning 2023.

optimal solution — cost — context (state of the world)

$$y^\star(x) \in \operatorname*{argmin}_{y\in\mathcal{C}(x)} f(y;x)$$

domain





Existing, widely-deployed uses of amortization

Tutorial on amortized optimization. Amos, Foundations and Trends in Machine Learning 2023.

Reinforcement learning and **control** (actor-critic methods, SAC, DDPG, GPS, BC)

Variational inference (VAEs, semi-amortized VAEs)

Meta-learning (HyperNets, MAML)

Sparse coding (PSD, LISTA)

Roots, fixed points, and convex optimization (NeuralDEQs, RLQP, NeuralSCS)

Foundations and Trends® in Machine Learning

Tutorial on amortized optimization

Learning to optimize over continuous spaces

Brandon Amos, *Meta AI*

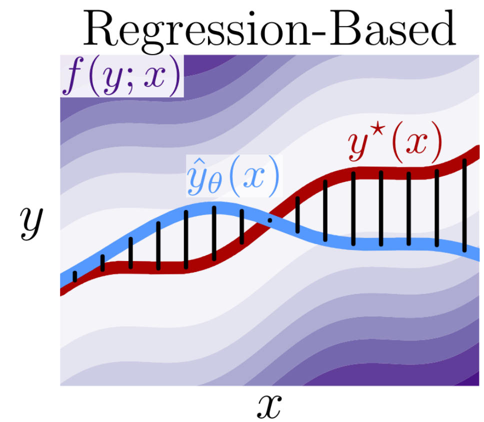
How to amortize?

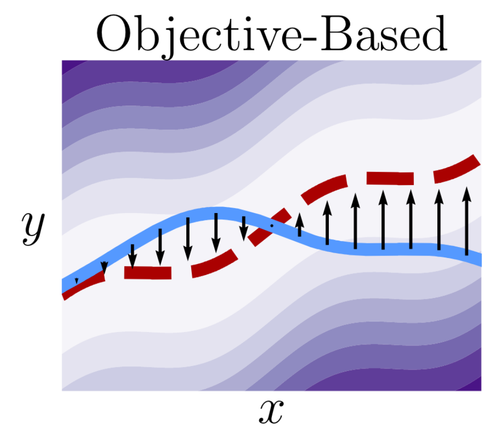
📚 *Tutorial on amortized optimization*. Amos, Foundations and Trends in Machine Learning 2023.

1. Define an **amortization model** $\hat{y}_\theta(x)$ to approximate $y^\star(x)$
   Example: a neural network mapping from x to the solution

2. Define a **loss** $\mathcal{L}$ that measures how well $\hat{y}$ fits $y^\star$
   Regression: $\mathcal{L}(\hat{y}_\theta) := \mathbb{E}_{p(x)} \; \|\hat{y}_\theta(x) - y^\star(x)\|_2^2$
   Objective: $\mathcal{L}(\hat{y}_\theta) := \mathbb{E}_{p(x)} f\big(\hat{y}_\theta(x)\big)$

3. Learn the model with $\min_\theta \; \mathcal{L}(\hat{y}_\theta)$





Why call it *amortized* optimization?

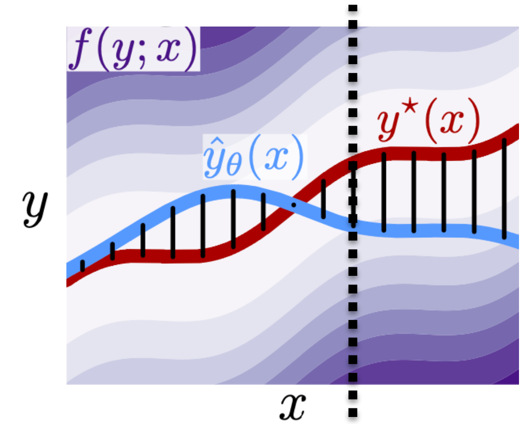
Tutorial on amortized optimization. Amos. FnT in ML, 2023.

to amortize: *to spread out an upfront cost over time*

$$\hat{y}_\theta(x) \approx y^\star(x) \in \operatorname*{argmin}_{y \in \mathcal{Y}(x)} f(y; x)$$

expensive upfront cost

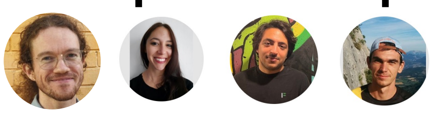
training the model → fast approximate solutions



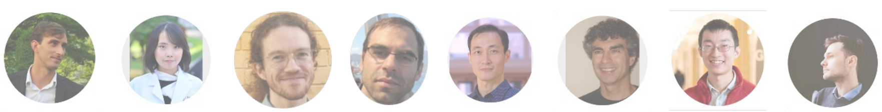
Talk overview

- Primer on **amortized optimization** [Foundations and Trends in ML, 2023]

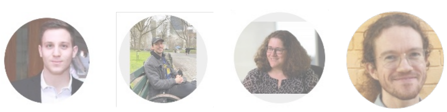
- **Meta Optimal Transport** [ICML 2023]



- **Meta Flow Matching** [2024]



- **Wasserstein Flow Matching** [2024]



Challenge: computing OT maps

Meta Optimal Transport. Amos et al., ICML 2023.

Monge (primal, Wasserstein-2)

$$T^\star(\alpha,\beta) \in \operatorname*{argmin}_{T\in\mathcal{T}(\alpha,\beta)} \mathbb{E}_{x\sim\alpha}\|x-T(x)\|_2^2$$

we also consider other/discrete OT formulations

Many OT problems are **numerically solved**
Improving OT solvers is active research

Solving multiple OT problems: even harder
Standard solution: independently solve

Meta Optimal Transport

Idea: predict the solution to OT problems with amortized optimization
Simultaneously solve many OT problems, sharing info between instances

(via amortization)
Why call it "meta"? Instead of solving a single OT problem, learn how to solve many

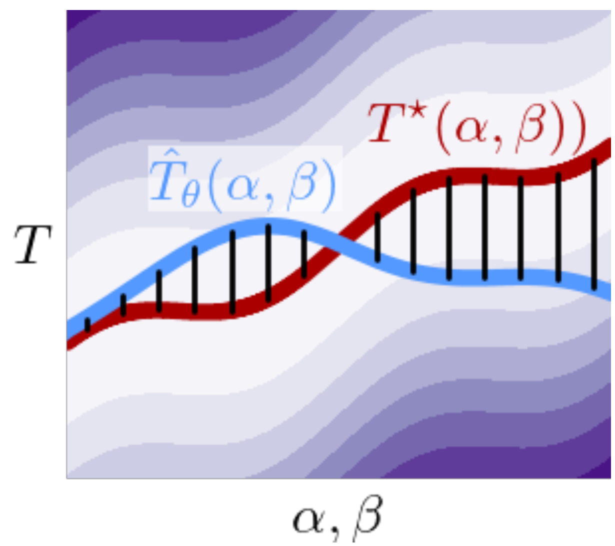
Monge (primal, Wasserstein-2)

$$T^\star(\alpha,\beta) \in \underset{T\in\mathcal{T}(\alpha,\beta)}{\operatorname{argmin}}\ \mathbb{E}_{x\sim\alpha}\|x-T(x)\|_2^2$$

$$\wr\wr$$

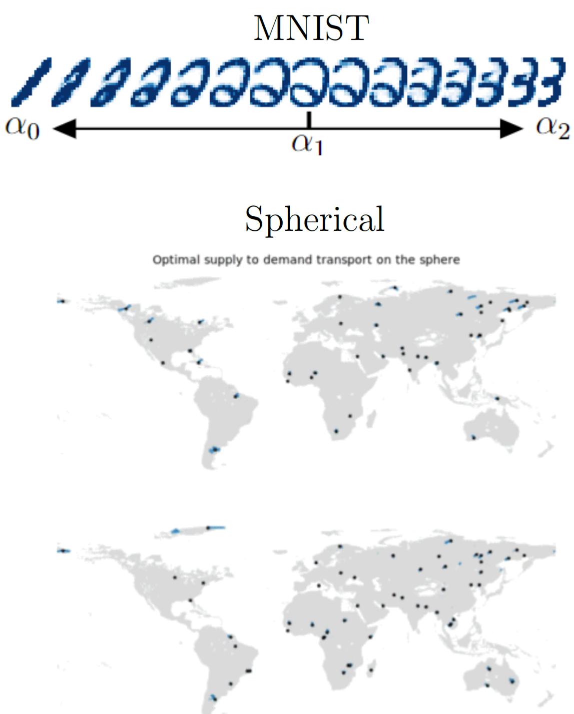
$\hat{T}_\theta(\alpha,\beta)$ (parameterize dual potential via an MLP)

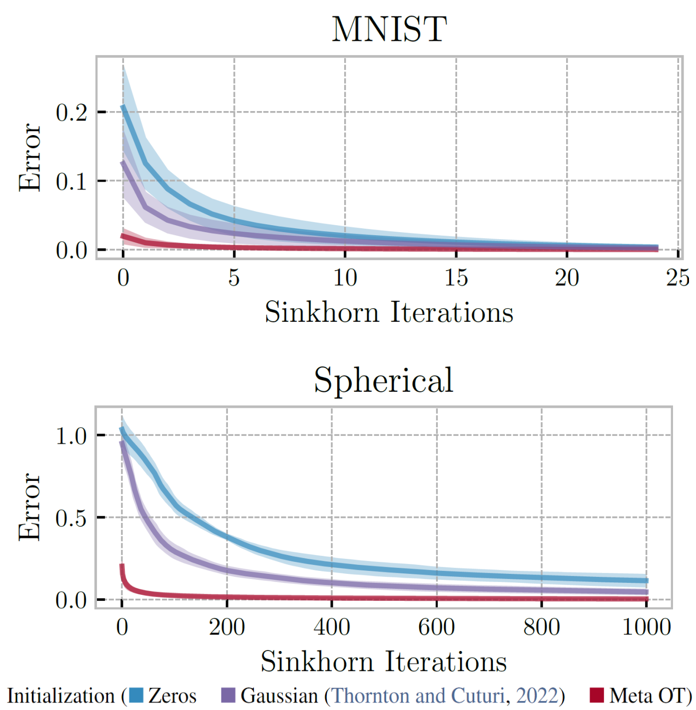
we also consider other/discrete OT formulations



Meta OT for Discrete OT (Sinkhorn)

📚 *Sinkhorn Distances: Lightspeed Computation of Optimal Transport.* Marco Cuturi, NeurIPS 2013.





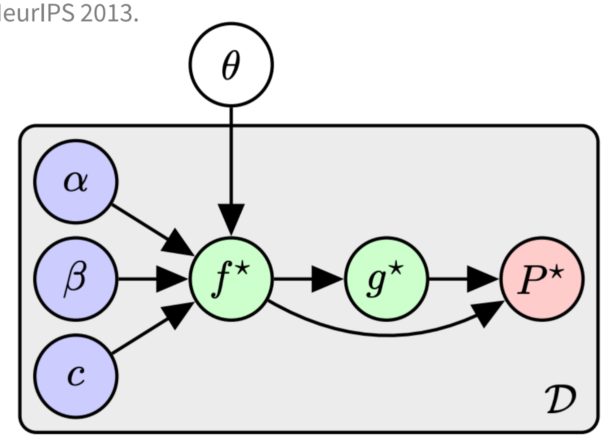


Table 1. Sinkhorn runtime (seconds) to reach a marginal error of 10^{-2}. Meta OT's initial prediction takes $\approx 5 \cdot 10^{-5}$ seconds. We report the mean and std across 10 test instances.

Initialization	MNIST	Spherical
Zeros (t_{zeros})	$4.5 \cdot 10^{-3}$ $\pm 1.5 \cdot 10^{-3}$	0.88 ± 0.13
Gaussian	$4.1 \cdot 10^{-3}$ $\pm 1.2 \cdot 10^{-3}$	0.56 $\pm 9.9 \cdot 10^{-2}$
Meta OT (t_{Meta})	$2.3 \cdot 10^{-3}$ $\pm 9.2 \cdot 10^{-6}$	$7.8 \cdot 10^{-2}$ $\pm 3.4 \cdot 10^{-2}$
Improvement ($t_{\text{zeros}}/t_{\text{Meta}}$)	1.96	11.3

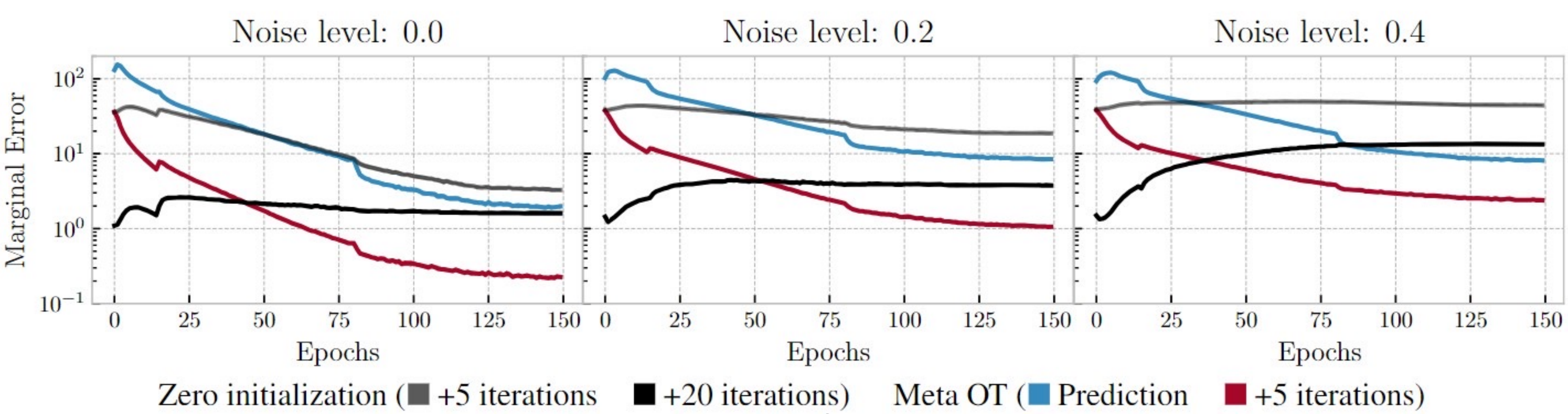
Wasserstein adversarial regularization

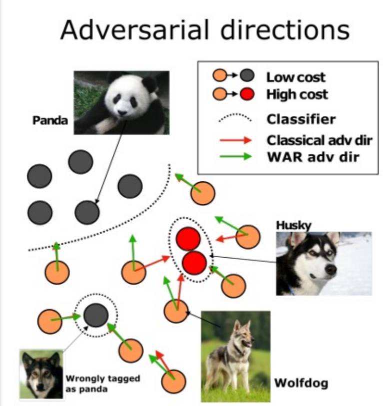
📚 *Wasserstein adversarial regularization for learning with label noise.* Kilian Fatras et al., TPAMI 2021.

Setting: discrete OT for classification with label noise

OT is **repeatedly solved** across minibatches
Use Meta OT to **learn better solutions**

Fig. 1: AR vs. WAR. Given a number of samples, both methods regularize along adversarial directions (arrows in the left panel), leading to updated decision functions (right panel). While both regularizations prevent the classifier to overfit on the noisy labelled sample, AR also tends over-smooth between similar classes (*wolfdog* and *husky*), while WAR preserves them by changing the adversarial direction.

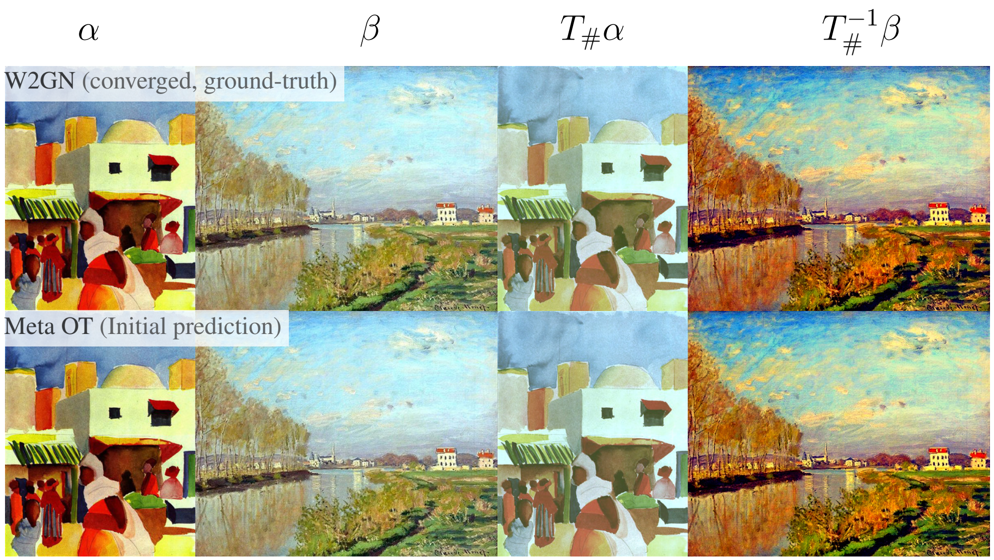


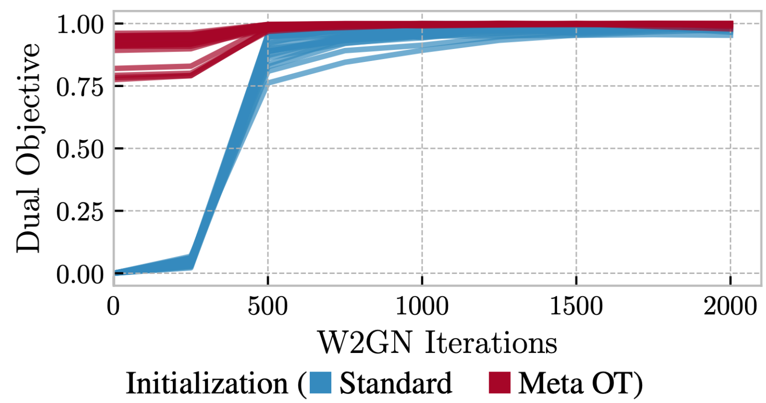


Meta OT in continuous settings (W2GN)

Wasserstein-2 Generative Networks. Alexander Korotin et al., ICLR 2021.

RGB color palette transport

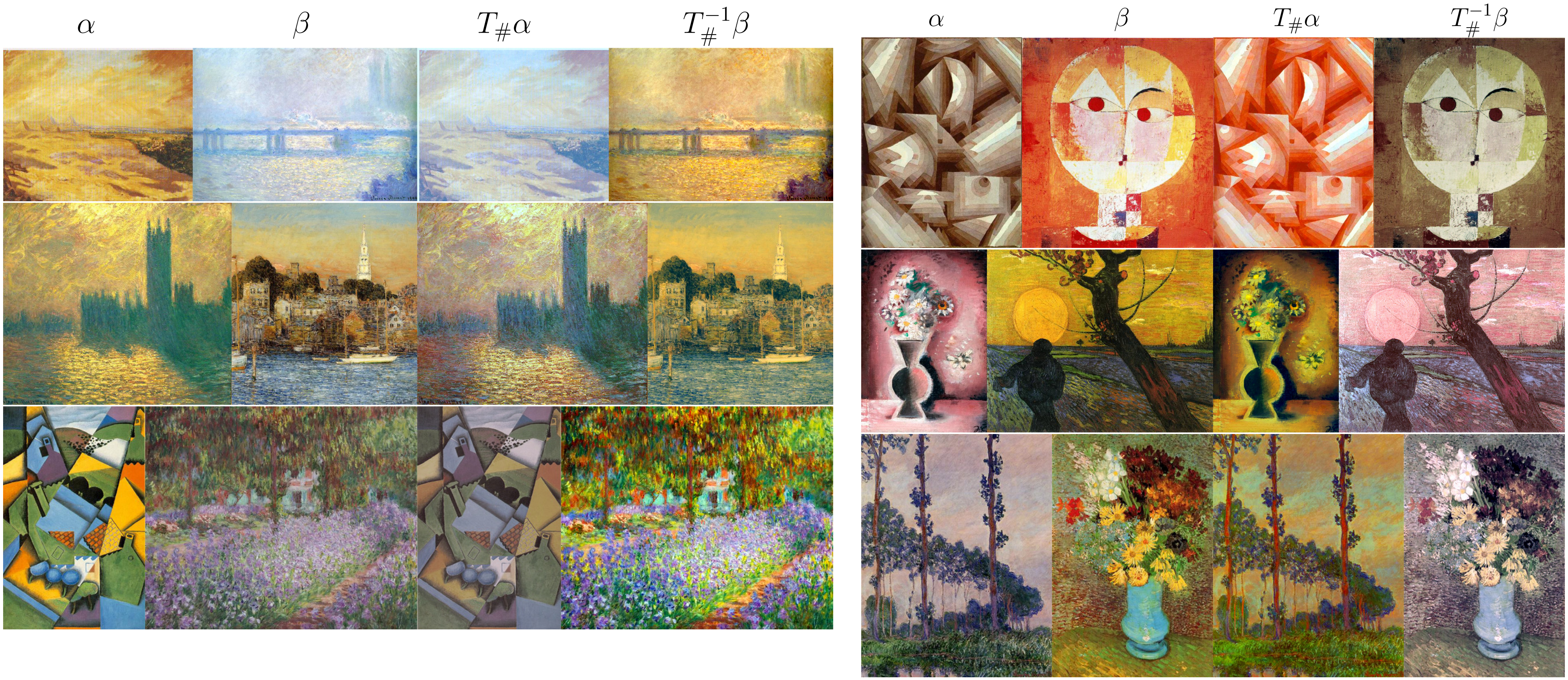




	Iter	Runtime (s)	Dual Value
Meta OT + W2GN	None	$3.5 \cdot 10^{-3} \pm 2.7 \cdot 10^{-4}$	$0.90 \pm 6.08 \cdot 10^{-2}$
	1k	$0.93 \pm 2.27 \cdot 10^{-2}$	$1.0 \pm 2.57 \cdot 10^{-3}$
	2k	$1.84 \pm 3.78 \cdot 10^{-2}$	$1.0 \pm 5.30 \cdot 10^{-3}$
W2GN	1k	$0.90 \pm 1.62 \cdot 10^{-2}$	$0.96 \pm 2.62 \cdot 10^{-2}$
	2k	$1.81 \pm 3.05 \cdot 10^{-2}$	$0.99 \pm 1.14 \cdot 10^{-2}$

More Meta OT color transfer predictions

Meta Optimal Transport. Amos et al., ICML 2023.



Conditional Monge Maps

📚 *Supervised Training of Conditional Monge Maps.* Bunne, Krause, Cuturi, NeurIPS 2022.

Focus: predicting drug treatments with OT
Idea: condition OT map on patient information

Methodological differences
Conditional Monge Maps ≈ Neural Processes
Predict conditioning inputs of the OT map

Meta OT ≈ Hyper-Networks
Predict parameters of an OT map

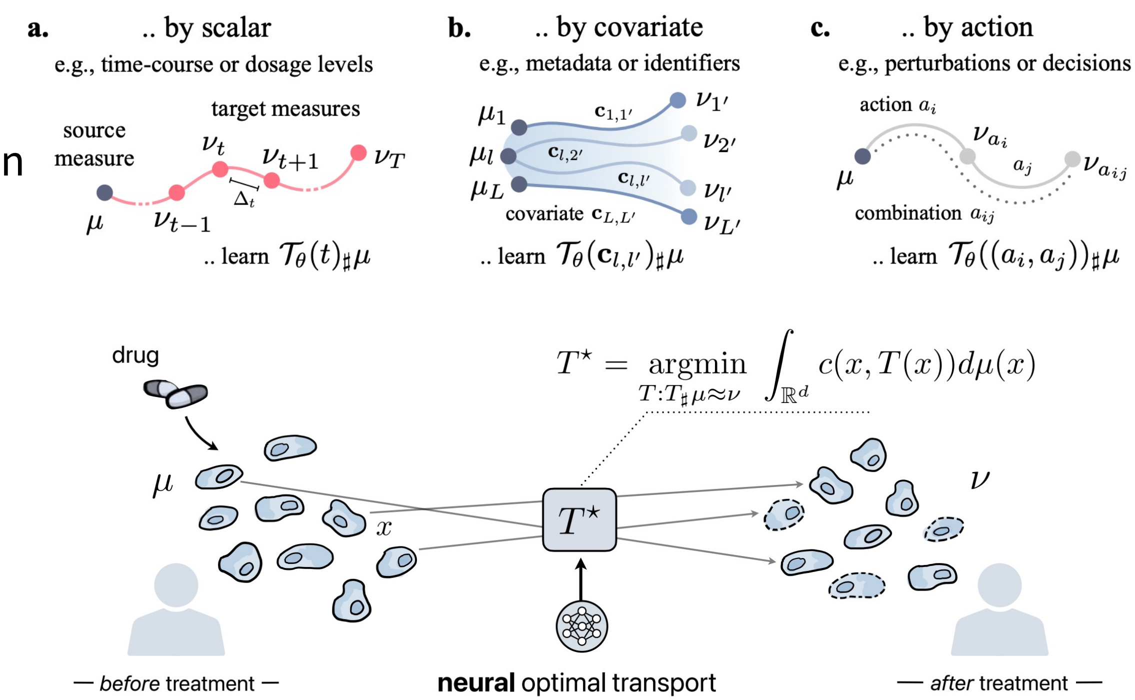
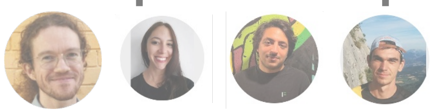


image sources: Bunne and Cuturi

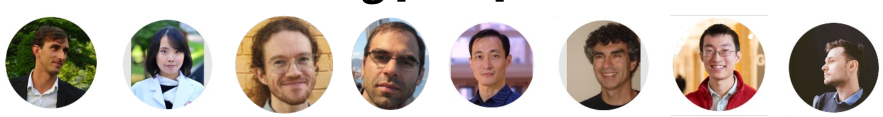
Talk overview

- Primer on **amortized optimization** [Foundations and Trends in ML, 2023]

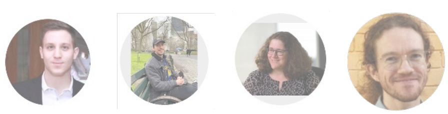
- **Meta Optimal Transport** [ICML 2023]



- **Meta Flow Matching** [2024]



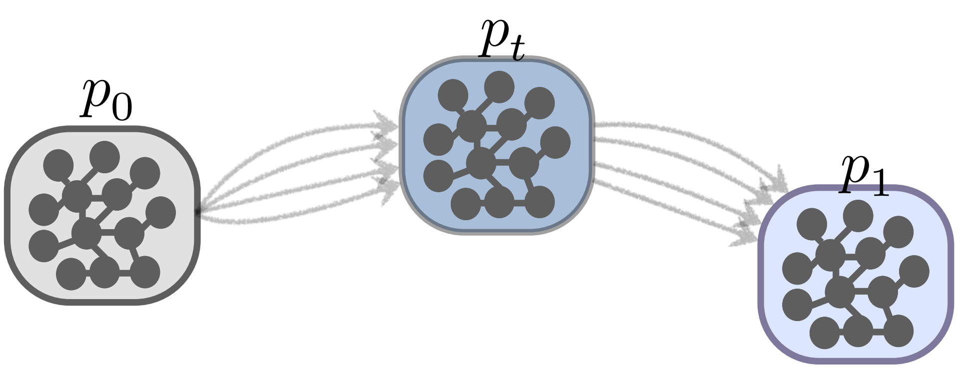
- **Wasserstein Flow Matching** [2024]



Background and Motivation

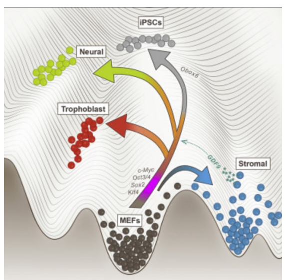
In many scientific problems, we want to understand the **dynamics of many-body problems**
(the dynamic evolution of interacting particles)

E.g. the dynamic processes **cells** undergo w.r.t. their **environment** and **interactions** with each other

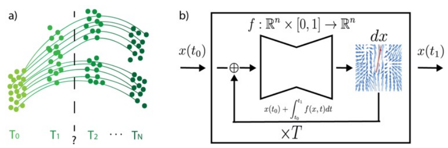


Background and Motivation

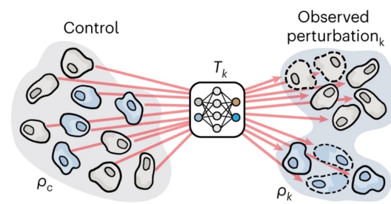
We want to model the dynamics of particles (or cells) at the **population level**. Many methods do this:



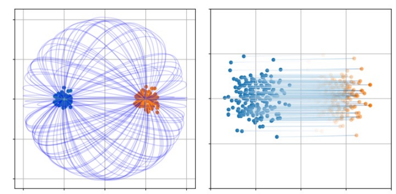
Schiebinger et al, *Cell*, 2019



Tong et al, *ICML*, 2020



Bunne et al, *Nature Methods*, 2023

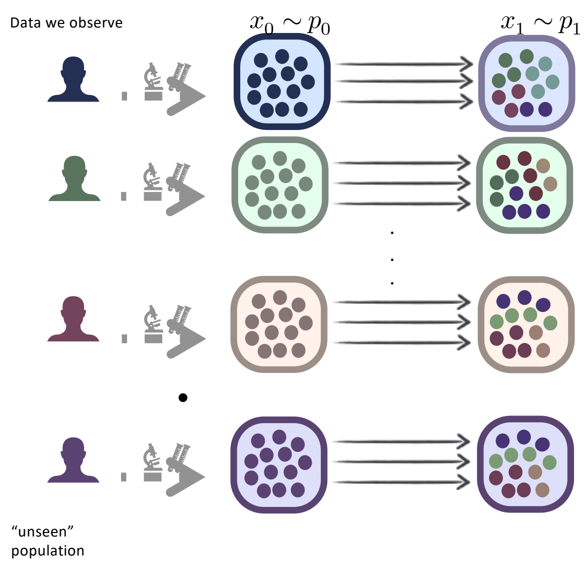


Neklydov et al, *ICML*, 2024

Existing methods typically only model the evolution of cells as independent particles.

Background and Motivation

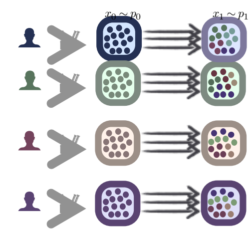
We would also like a model that can **generalize across measures** (populations)



Existing methods are typically restricted to a single measure (population, patient). At best can condition on different dynamics.

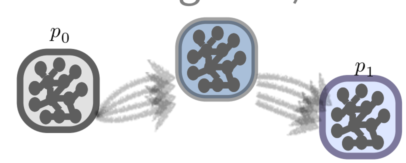
Problem setup

We want a model that can:
1. model the **evolution of particles** while taking into account their **interactions**
2. **generalize** across **unseen populations**



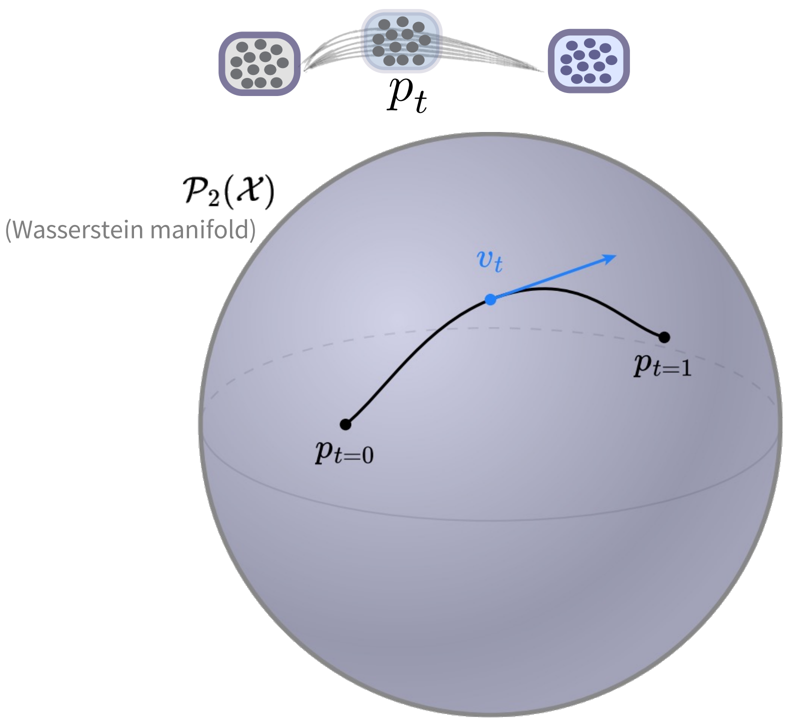
Main assumptions:

1. **Coupled distribution**/population pairs $\{(p_0(x_0|i), p_1(x_1|i))\}_{i=1}^{N}$
2. The collected data undergoes a **universal developmental process**
   depends only on the population itself (e.g., interacting particles or communicating cells)



From Flow Matching

$$\frac{\partial p_t(x)}{\partial t} = -\langle \nabla_x, p_t(x) v_t^*(x) \rangle$$

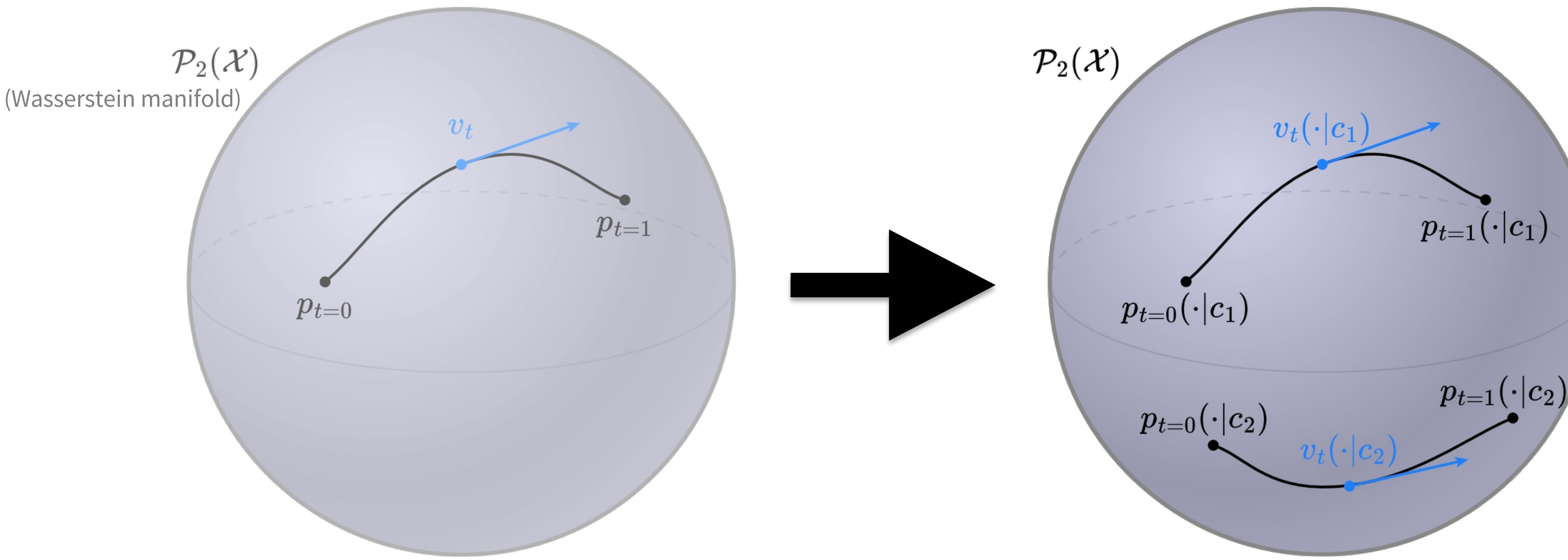


From Flow Matching to Meta Flow Matching

$$\frac{\partial p_t(x)}{\partial t} = -\langle \nabla_x, p_t(x) v_t^*(x) \rangle$$

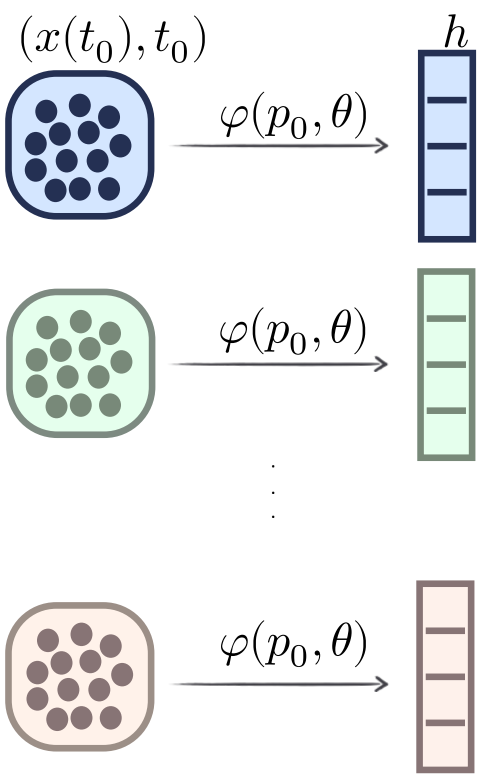
$$\frac{\partial p_t(x)}{\partial t} = -\langle \nabla_x, p_t(x) v_t^*(x, \boxed{p_t}) \rangle$$

(from assumptions: end up just conditioning on p_0)

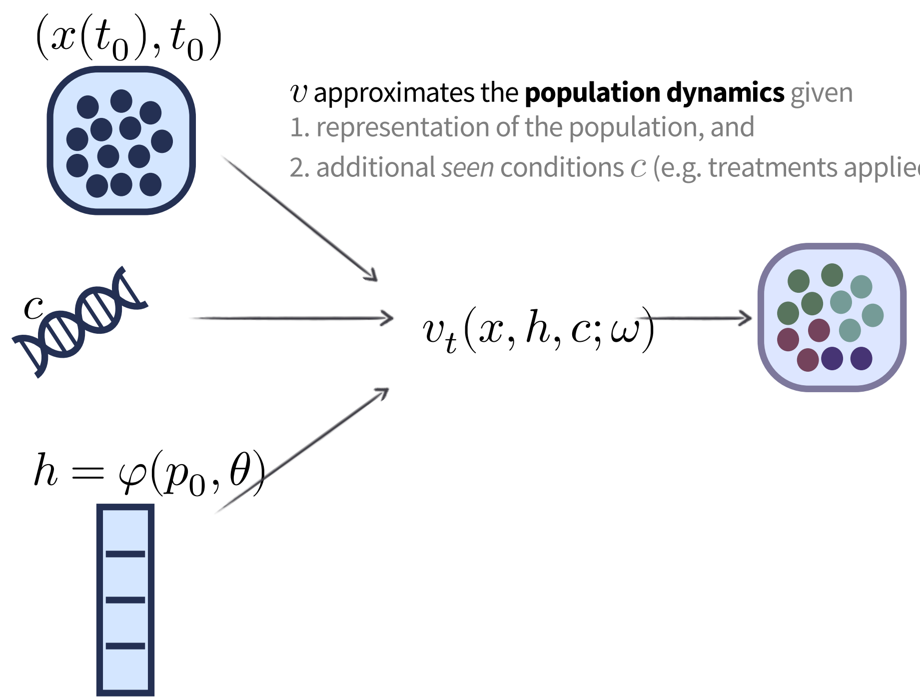


Meta Flow Matching

A ***model*** to learn to represent the population (GCN w/ *knn* edge pooling)



v approximates the **population dynamics** given
1. representation of the population, and
2. additional *seen* conditions c (e.g. treatments applied to population)



$\varphi(p_0, \theta)$ (GCN) captures **interactions** between particles

Algorithm 1: Meta Flow Matching (training)

Input : dataset of populations $\{(\pi(x_0, x_1 \mid i), c^i)\}_{i=1}^N$ and treatments c^i, and parametric models for the velocity, $v_t(\cdot;\omega)$, and population embedding $\varphi(\cdot;\theta)$.

for *training iterations* **do**

$i \sim \mathcal{U}_{\{1,N\}}(i)$ `// sample batch of` n `populations ids`

$(x_0^j, x_1^j, t^j) \sim \pi(x_0, x_1 \mid i)\mathcal{U}_{[0,1]}(t)$ `// sample` N_i `particles for every population` i

$f_t(x_0^j, x_1^j) \leftarrow (1-t^j)x_0^j + t^j x_1^j$

$h^i(\theta) \leftarrow \varphi\left(\{x_0^j\}_{j=1}^{N_i};\theta\right)$ `// embed population` $\{x_0^j\}_{j=1}^{N_i}$`. For CGFM` $h \leftarrow i$`, FM` $h \leftarrow \emptyset$`.`

$\mathcal{L}_{\text{MFM}}(\omega,\theta) \leftarrow \frac{1}{n}\sum_i \frac{1}{n_i}\sum_j \left\| \frac{d}{dt} f_t(x_0^j, x_1^j) - v_{t^j}\left(f_t(x_0^j, x_1^j) \mid h^i(\theta), c^i; \omega\right)\right\|^2$

$\omega' \leftarrow \text{Update}(\omega, \nabla_\omega \mathcal{L}_{\text{MFM}}(\omega,\theta))$ `// evaluate new parameters of the flow model`

$\theta' \leftarrow \text{Update}(\theta, \nabla_\theta \mathcal{L}_{\text{MFM}}(\omega,\theta))$ `// evaluate new parameters of the embedding model`

$\omega \leftarrow \omega', \quad \theta \leftarrow \theta'$ `// update both models`

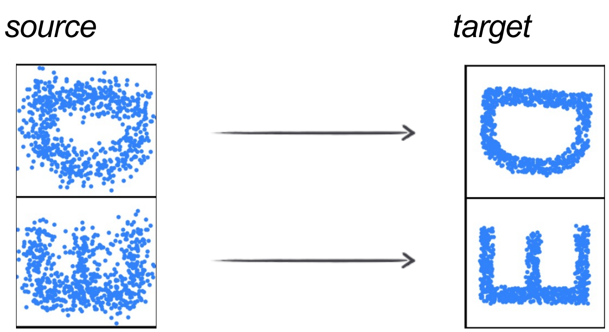
return $v_t(\cdot;\omega^*), \varphi(\cdot;\theta^*)$

Synthetic Example

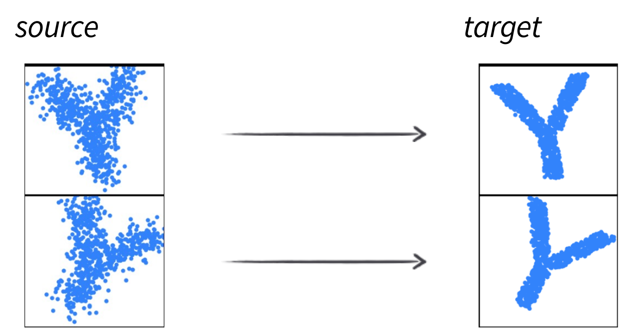
We create a **synthetic dataset** of paired joint distributions $\{(p_0(x_0|i), p_1(x_1|i))\}_{i=1}^N$

- We define a set of pre-defined **target** distributions $p_1(x_1|i)$ for $i = 1, \ldots, N$ (letter silhouettes)
- To get paired $p_0(x_0|i)$ we **simulate the forward diffusion process** without drift $x_0 \sim \mathcal{N}(x_1, \sigma)$
- We **reverse** the diffusion process and learn the push-forward map from $p_0(x_0|i)$ (***source***) to $p_1(x_1|i)$ (***target***) for every index i

Train: 24 letters (excluding 'Y' and 'X'), each in 10 orientations

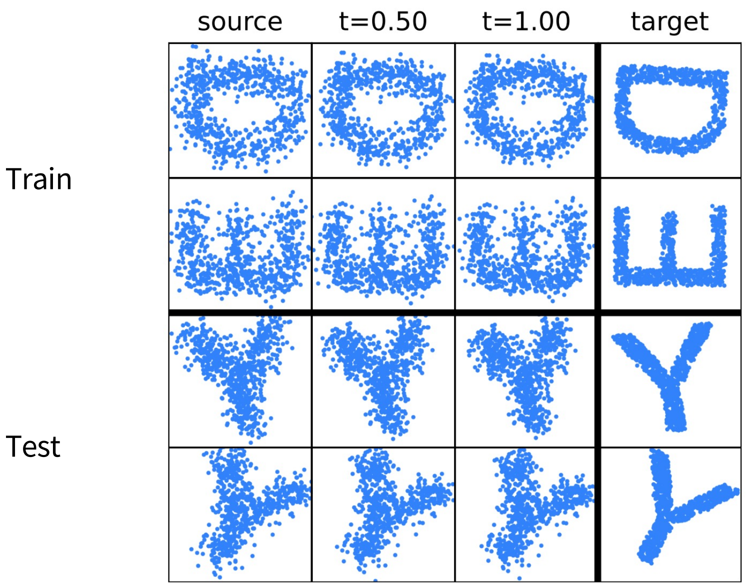


Test: 'Y' and 'X', each in 10 orientations



Synthetic Example (FM baseline)

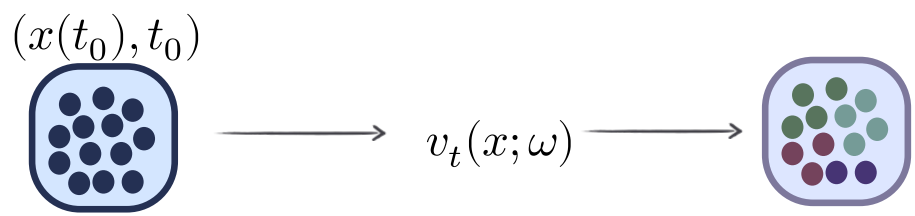
No population information



FM

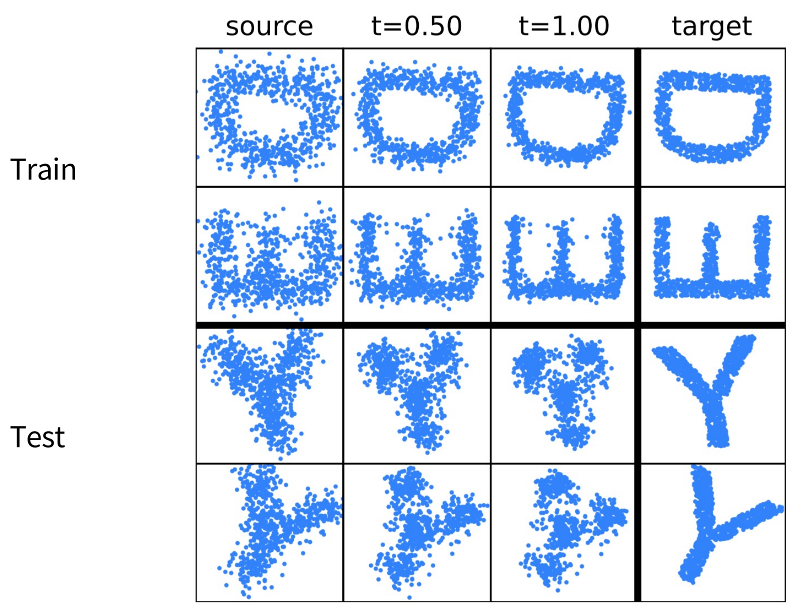
FM cannot fit the training data and cannot generalize to *unseen* populations

Predicts aggregate response over populations



Synthetic Example (CGFM baseline)

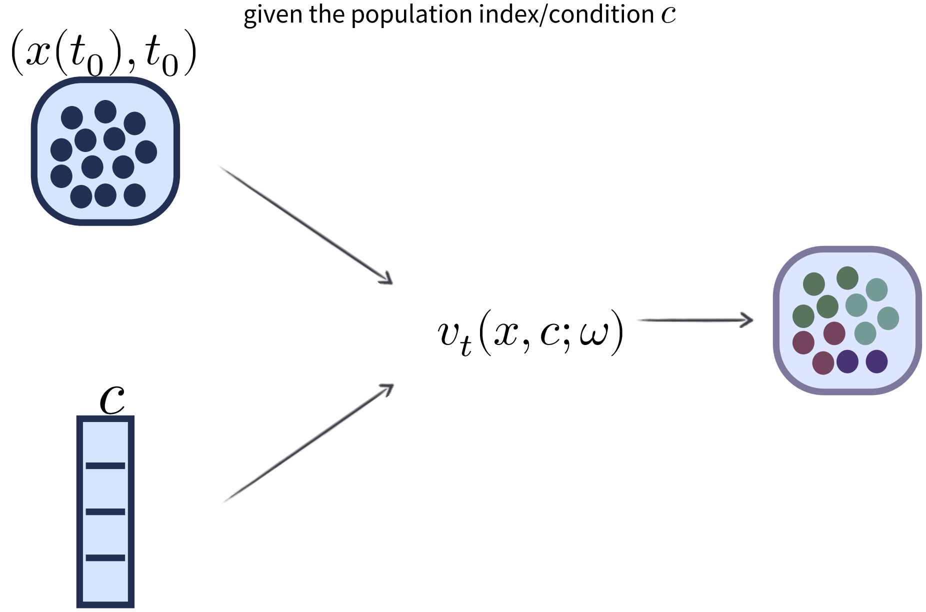
“Perfect” conditions (naive)



CGFM

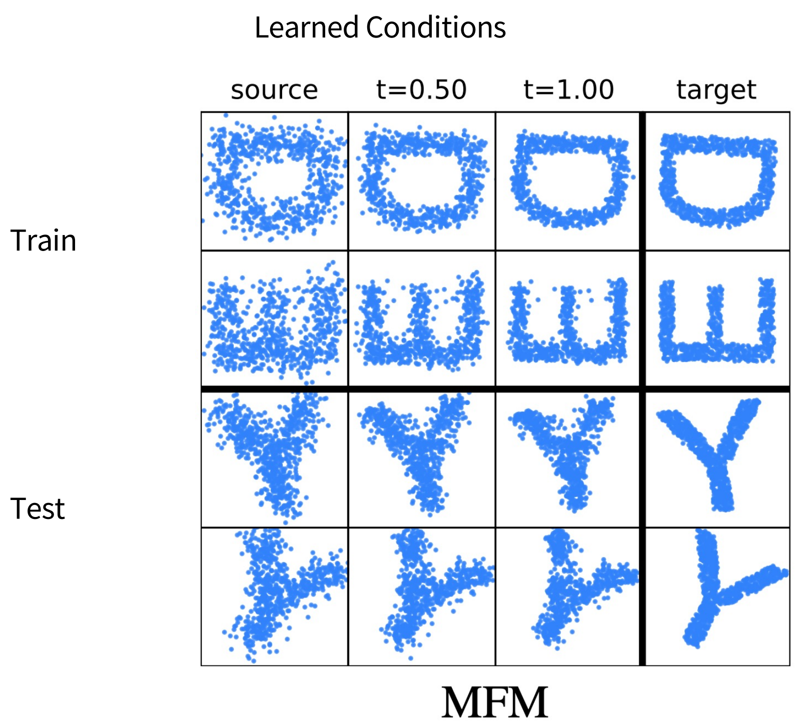
CGFM cannot generalize to the conditions of *unseen* populations

A model v to approximate the dynamical response given the population index/condition c

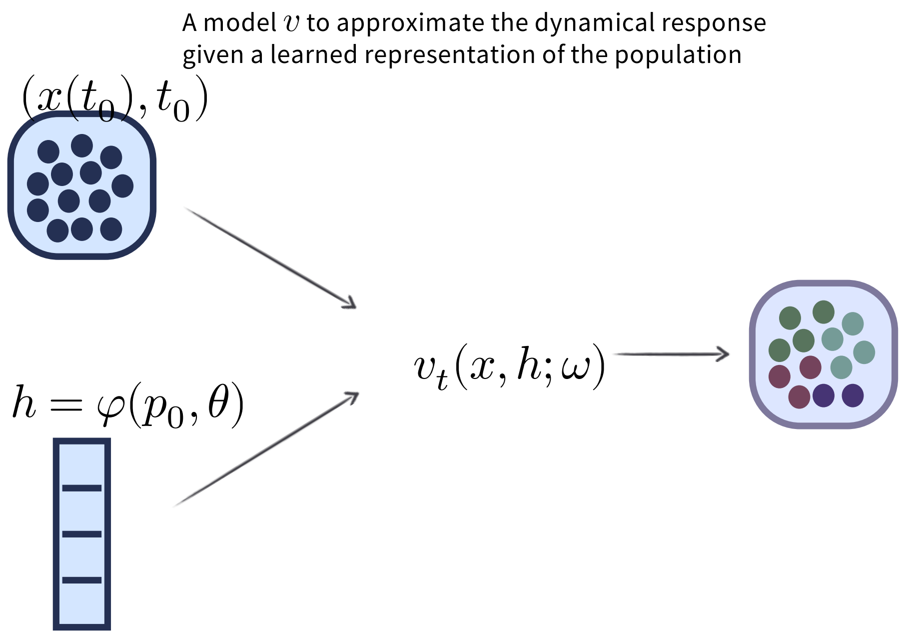


Perfect information on which environment the model is working with

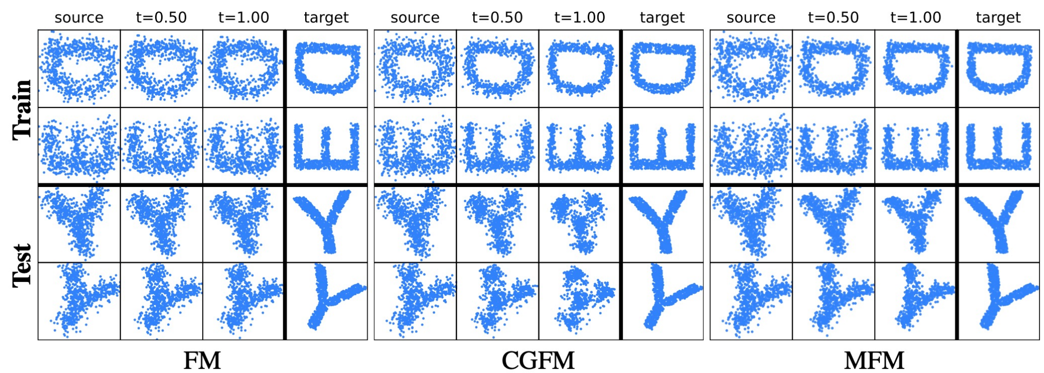
Synthetic Example (MFM)



MFM learns to represent entire populations, hence generalizes across *unseen* populations



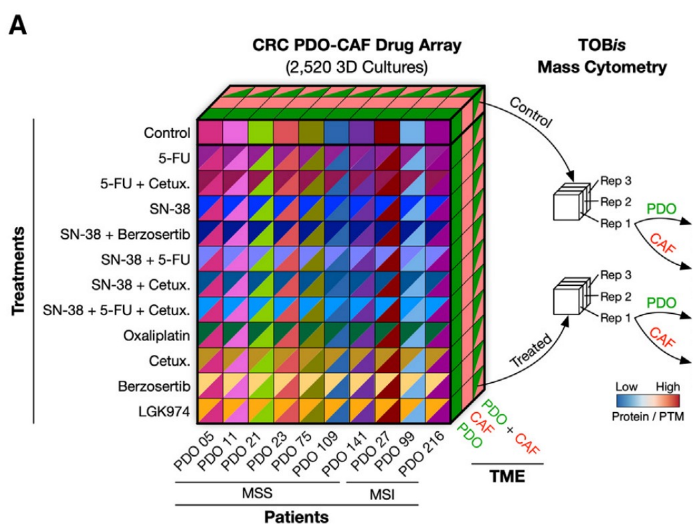
Synthetic Example



	Train			Test (X's)			Test (Y's)		
	$\mathcal{W}_1$	$\mathcal{W}_2$	MMD ($\times 10^{-3}$)	$\mathcal{W}_1$	$\mathcal{W}_2$	MMD ($\times 10^{-3}$)	$\mathcal{W}_1$	$\mathcal{W}_2$	MMD ($\times 10^{-3}$)
FM	0.209 ± 0.000	0.277 ± 0.000	2.54 ± 0.00	0.234 ± 0.000	0.309 ± 0.000	2.45 ± 0.00	0.238 ± 0.000	0.316 ± 0.000	3.32 ± 0.01
$\text{FM}^{\text{w/}}\mathcal{N}$	0.806 ± 0.000	0.960 ± 0.000	31.68 ± 0.00	0.764 ± 0.000	0.931 ± 0.000	25.04 ± 0.00	1.030 ± 0.000	1.228 ± 0.000	45.36 ± 0.00
CGFM	0.090 ± 0.000	0.113 ± 0.000	0.25 ± 0.00	0.334 ± 0.000	0.407 ± 0.000	5.55 ± 0.00	0.327 ± 0.000	0.405 ± 0.000	6.85 ± 0.00
$\text{CGFM}^{\text{w/}}\mathcal{N}$	0.156 ± 0.025	0.201 ± 0.027	1.02 ± 0.39	0.849 ± 0.004	0.993 ± 0.003	35.08 ± 0.75	1.062 ± 0.011	1.229 ± 0.010	55.66 ± 0.76
$\text{MFM}^{\text{w/}}\mathcal{N}$ ($k=0$)	0.148 ± 0.003	0.195 ± 0.010	0.94 ± 0.11	0.347 ± 0.011	0.431 ± 0.012	6.47 ± 0.44	0.402 ± 0.011	0.485 ± 0.010	10.92 ± 0.18
$\text{MFM}^{\text{w/}}\mathcal{N}$ ($k=1$)	0.154 ± 0.004	0.208 ± 0.010	0.91 ± 0.01	0.349 ± 0.023	0.433 ± 0.023	6.53 ± 0.52	0.391 ± 0.035	0.477 ± 0.041	10.71 ± 1.86
$\text{MFM}^{\text{w/}}\mathcal{N}$ ($k=10$)	0.151 ± 0.013	0.197 ± 0.015	0.94 ± 0.15	0.343 ± 0.020	0.427 ± 0.019	6.38 ± 0.67	0.413 ± 0.018	0.502 ± 0.024	11.93 ± 1.14
$\text{MFM}^{\text{w/}}\mathcal{N}$ ($k=50$)	0.174 ± 0.005	0.232 ± 0.006	1.40 ± 0.13	0.363 ± 0.010	0.449 ± 0.013	7.46 ± 0.44	0.446 ± 0.021	0.536 ± 0.028	13.40 ± 0.23
MFM ($k=0$)	$\mathbf{0.081 \pm 0.003}$	$\mathbf{0.100 \pm 0.004}$	$\mathbf{0.16 \pm 0.06}$	0.202 ± 0.002	0.249 ± 0.003	2.29 ± 0.05	0.218 ± 0.001	0.262 ± 0.002	3.79 ± 0.11
MFM ($k=1$)	0.082 ± 0.001	0.101 ± 0.002	$\mathbf{0.16 \pm 0.01}$	0.205 ± 0.008	0.251 ± 0.008	2.38 ± 0.22	0.215 ± 0.006	0.258 ± 0.007	3.78 ± 0.25
MFM ($k=10$)	0.088 ± 0.002	0.109 ± 0.003	0.21 ± 0.01	$\mathbf{0.201 \pm 0.006}$	$\mathbf{0.248 \pm 0.006}$	2.20 ± 0.15	0.208 ± 0.003	0.252 ± 0.002	3.55 ± 0.06
MFM ($k=50$)	0.092 ± 0.004	0.116 ± 0.004	0.25 ± 0.06	0.206 ± 0.008	0.257 ± 0.008	$\mathbf{2.18 \pm 0.25}$	$\mathbf{0.204 \pm 0.005}$	$\mathbf{0.249 \pm 0.006}$	$\mathbf{3.14 \pm 0.18}$

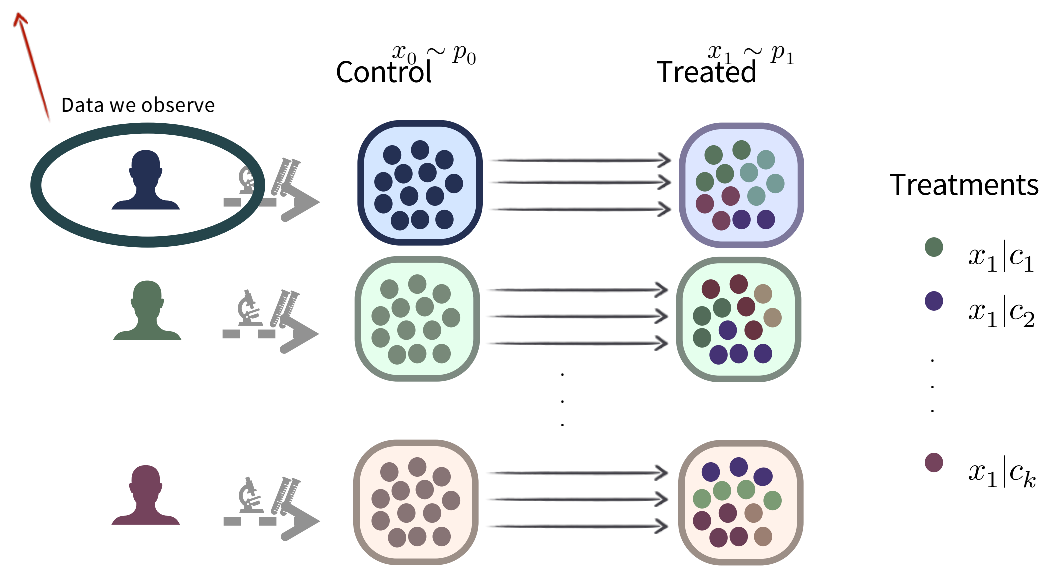
Biological data — patient-specific organoid drug screen dataset

Mass Cytometry



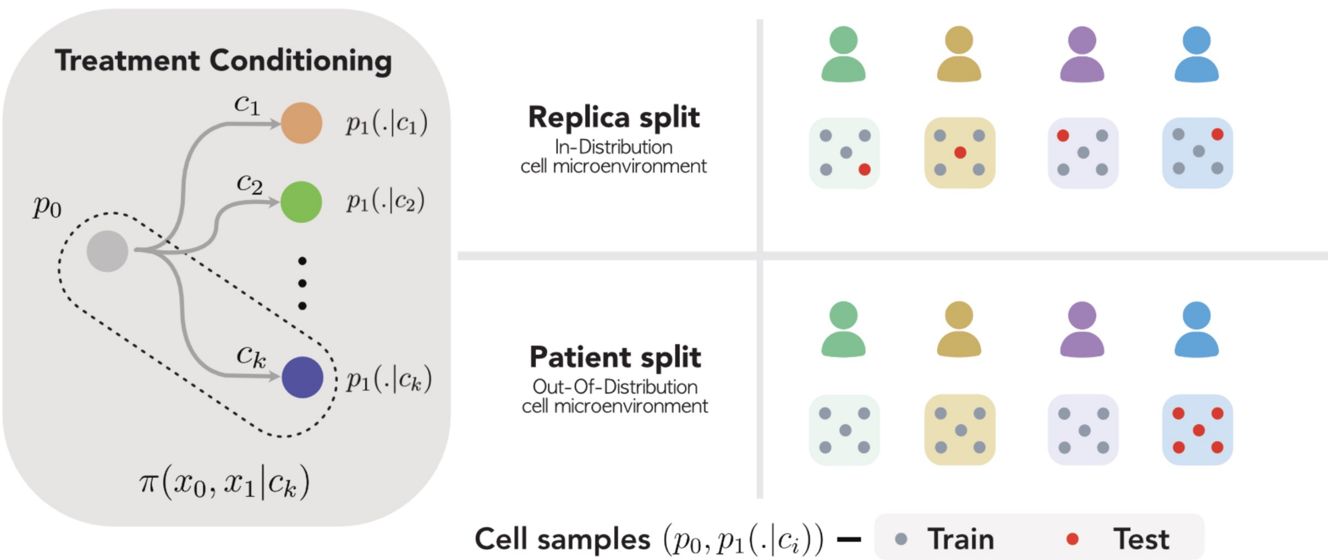
(Zapatero et al, *Cell*, 2023)

Each patient has ~ 250 different (control, treated) pairs



10 patients, 11 treatments, varying doses, 3 different cell cultures . . . ***up to 2500 different environmental conditions!***
(we use ~ 1000)

Organoid Drug Screen Data



"Replica" Split

	Train				Test			
	$\mathcal{W}_1(\downarrow)$	$\mathcal{W}_2(\downarrow)$	MMD $(\times 10^{-3})$ $(\downarrow)$	$r^2(\uparrow)$	$\mathcal{W}_1(\downarrow)$	$\mathcal{W}_2(\downarrow)$	MMD $(\times 10^{-3})$ $(\downarrow)$	$r^2(\uparrow)$
FM	3.925 ± 0.019	4.041 ± 0.023	3.76 ± 0.26	0.952 ± 0.007	3.961 ± 0.036	4.089 ± 0.042	5.90 ± 0.25	0.941 ± 0.010
FM$^{\text{w/}}\mathcal{N}$	6.908 ± 0.037	7.181 ± 0.033	57.70 ± 0.75	0.639 ± 0.005	6.972 ± 0.022	7.244 ± 0.022	60.39 ± 0.98	0.642 ± 0.007
CGFM	$\mathbf{3.864 \pm 0.064}$	3.975 ± 0.069	$\mathbf{3.16 \pm 0.89}$	0.964 ± 0.006	4.087 ± 0.063	4.211 ± 0.066	8.84 ± 0.75	0.938 ± 0.006
CGFM$^{\text{w/}}\mathcal{N}$	4.187 ± 0.008	4.340 ± 0.009	8.69 ± 0.50	0.936 ± 0.002	6.852 ± 0.045	7.114 ± 0.044	71.24 ± 3.71	0.666 ± 0.016
ICNN	4.286 ± 0.018	4.313 ± 0.112	38.6 ± 0.212	0.897 ± 0.031	4.194 ± 0.110	4.313 ± 0.112	37.9 ± 2.84	0.897 ± 0.008
MFM$^{\text{w/}}\mathcal{N}$ $(k=0)$	3.940 ± 0.022	4.047 ± 0.023	3.91 ± 0.18	0.959 ± 0.006	3.896 ± 0.026	4.002 ± 0.030	$\mathbf{4.35 \pm 0.18}$	0.950 ± 0.005
MFM$^{\text{w/}}\mathcal{N}$ $(k=10)$	3.976 ± 0.044	4.086 ± 0.049	4.52 ± 0.42	0.961 ± 0.002	3.943 ± 0.032	4.051 ± 0.034	5.28 ± 0.25	0.952 ± 0.001
MFM$^{\text{w/}}\mathcal{N}$ $(k=50)$	3.968 ± 0.013	4.075 ± 0.014	4.36 ± 0.44	0.961 ± 0.002	3.934 ± 0.007	4.041 ± 0.008	4.99 ± 0.35	0.954 ± 0.000
MFM$^{\text{w/}}\mathcal{N}$ $(k=100)$	3.937 ± 0.014	4.040 ± 0.015	3.94 ± 0.00	0.963 ± 0.001	3.908 ± 0.030	4.011 ± 0.033	4.68 ± 0.52	0.953 ± 0.002
MFM $(k=0)$	3.874 ± 0.015	$\mathbf{3.973 \pm 0.020}$	3.37 ± 0.14	$\mathbf{0.967 \pm 0.003}$	$\mathbf{3.880 \pm 0.009}$	$\mathbf{3.990 \pm 0.011}$	4.68 ± 0.16	0.955 ± 0.002
MFM $(k=10)$	3.896 ± 0.021	4.000 ± 0.021	3.82 ± 0.12	0.964 ± 0.001	3.899 ± 0.013	4.012 ± 0.011	5.13 ± 0.48	0.955 ± 0.001
MFM $(k=50)$	3.888 ± 0.038	3.991 ± 0.030	3.59 ± 0.41	0.963 ± 0.001	3.900 ± 0.038	4.013 ± 0.034	5.06 ± 0.22	0.954 ± 0.003
MFM $(k=100)$	3.906 ± 0.010	4.008 ± 0.005	4.05 ± 0.38	0.964 ± 0.002	3.898 ± 0.008	4.009 ± 0.009	5.19 ± 0.05	$\mathbf{0.957 \pm 0.000}$

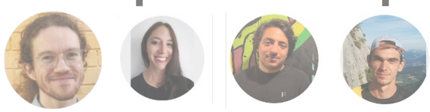
Patient Split

	Train				Test			
	$\mathcal{W}_1(\downarrow)$	$\mathcal{W}_2(\downarrow)$	MMD ($\times 10^{-3}$) ($\downarrow$)	$r^2(\uparrow)$	$\mathcal{W}_1(\downarrow)$	$\mathcal{W}_2(\downarrow)$	MMD ($\times 10^{-3}$) ($\downarrow$)	$r^2(\uparrow)$
FM	3.985 ± 0.054	4.115 ± 0.067	4.64 ± 0.43	0.938 ± 0.014	4.340 ± 0.078	4.564 ± 0.111	13.00 ± 0.67	0.865 ± 0.034
FM$^{\text{w/}}\mathcal{N}$	6.892 ± 0.027	7.164 ± 0.033	57.03 ± 1.00	0.655 ± 0.003	7.114 ± 0.100	7.404 ± 0.086	64.97 ± 3.79	0.613 ± 0.008
CGFM	$\mathbf{3.882 \pm 0.019}$	3.999 ± 0.020	$\mathbf{3.16 \pm 0.59}$	0.952 ± 0.004	4.443 ± 0.033	4.621 ± 0.041	17.00 ± 1.03	0.899 ± 0.008
CGFM$^{\text{w/}}\mathcal{N}$	4.313 ± 0.077	4.480 ± 0.081	11.51 ± 1.96	0.918 ± 0.004	7.135 ± 0.045	7.390 ± 0.037	79.78 ± 4.67	0.637 ± 0.010
ICNN	4.289 ± 0.020	4.382 ± 0.021	37.0 ± 2.84	0.913 ± 0.003	4.525 ± 0.051	4.681 ± 0.054	74.00 ± 0.57	0.862 ± 0.127
MFM$^{\text{w/}}\mathcal{N}$ ($k = 0$)	3.982 ± 0.014	4.095 ± 0.015	5.04 ± 0.36	0.951 ± 0.002	4.177 ± 0.042	4.355 ± 0.048	10.53 ± 0.59	0.911 ± 0.001
MFM$^{\text{w/}}\mathcal{N}$ ($k = 10$)	4.006 ± 0.008	4.119 ± 0.012	5.13 ± 0.30	0.948 ± 0.001	4.156 ± 0.065	4.324 ± 0.067	9.58 ± 1.63	0.912 ± 0.003
MFM$^{\text{w/}}\mathcal{N}$ ($k = 50$)	3.982 ± 0.018	4.095 ± 0.016	4.74 ± 0.21	0.951 ± 0.002	4.153 ± 0.069	4.324 ± 0.070	9.63 ± 1.45	0.912 ± 0.002
MFM$^{\text{w/}}\mathcal{N}$ ($k = 100$)	4.004 ± 0.012	4.119 ± 0.014	5.19 ± 0.43	0.949 ± 0.002	4.166 ± 0.001	4.341 ± 0.003	9.52 ± 0.33	0.915 ± 0.005
MFM ($k = 0$)	3.905 ± 0.005	4.012 ± 0.006	4.18 ± 0.25	$\mathbf{0.958 \pm 0.001}$	4.209 ± 0.007	4.380 ± 0.012	12.34 ± 0.50	$\mathbf{0.918 \pm 0.002}$
MFM ($k = 10$)	3.896 ± 0.033	4.005 ± 0.036	3.89 ± 0.44	0.957 ± 0.005	4.216 ± 0.090	4.395 ± 0.098	11.99 ± 2.36	0.917 ± 0.005
MFM ($k = 50$)	3.902 ± 0.018	4.008 ± 0.022	4.20 ± 0.17	$\mathbf{0.958 \pm 0.000}$	4.214 ± 0.017	4.396 ± 0.020	12.09 ± 0.75	0.916 ± 0.002
MFM ($k = 100$)	3.884 ± 0.039	$\mathbf{3.986 \pm 0.044}$	3.77 ± 0.49	0.955 ± 0.001	$\mathbf{4.100 \pm 0.093}$	$\mathbf{4.269 \pm 0.104}$	$\mathbf{8.96 \pm 1.88}$	0.917 ± 0.004

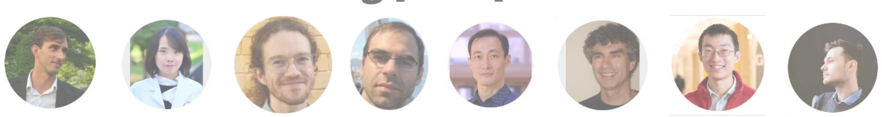
Talk overview

- Primer on **amortized optimization** [Foundations and Trends in ML, 2023]

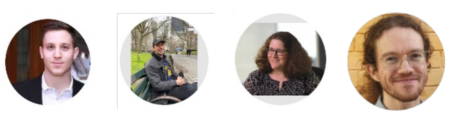
- **Meta Optimal Transport** [ICML 2023]



- **Meta Flow Matching** [2024]

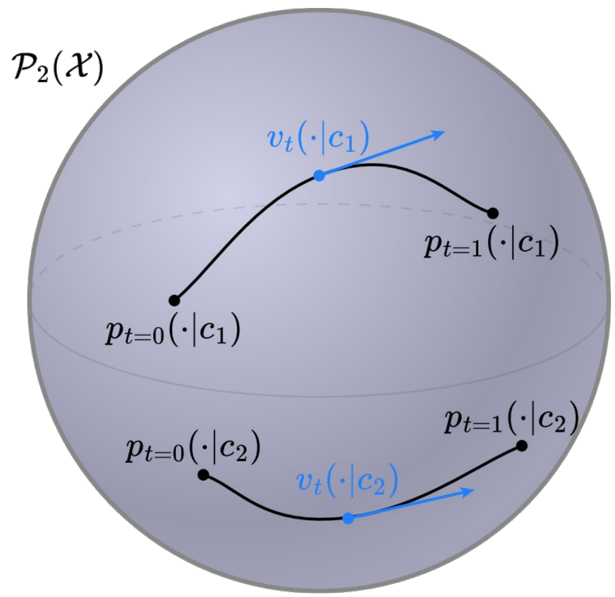


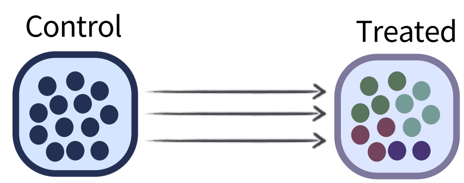
- **Wasserstein Flow Matching** [2024]

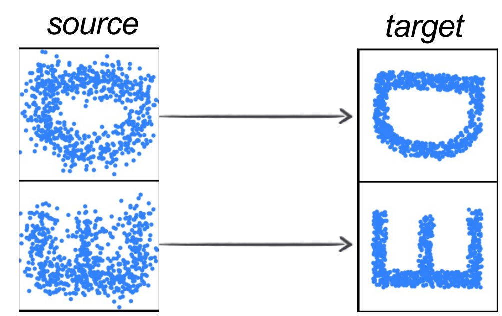


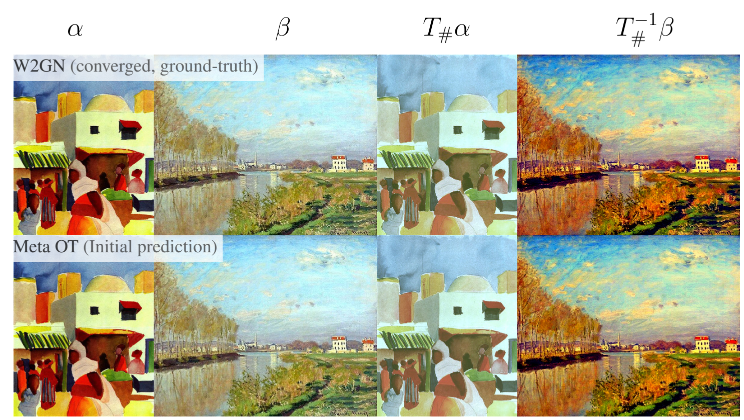
So far: distributions over pairs of distributions

Meta OT and Meta FM assume (coupled) **pairs of distributions**





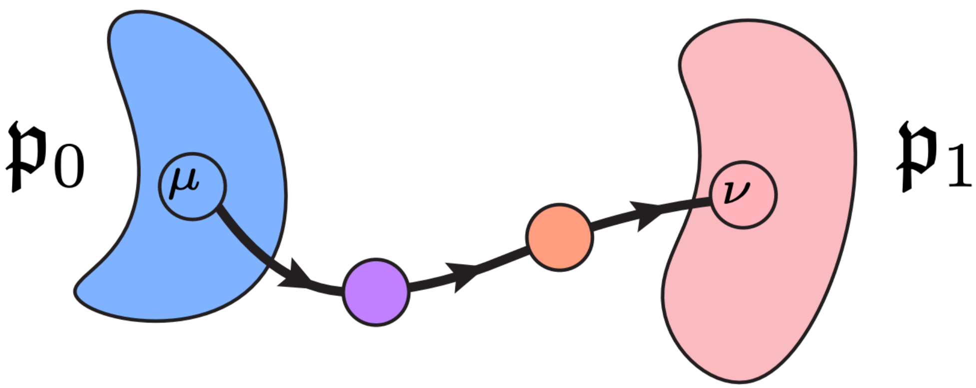


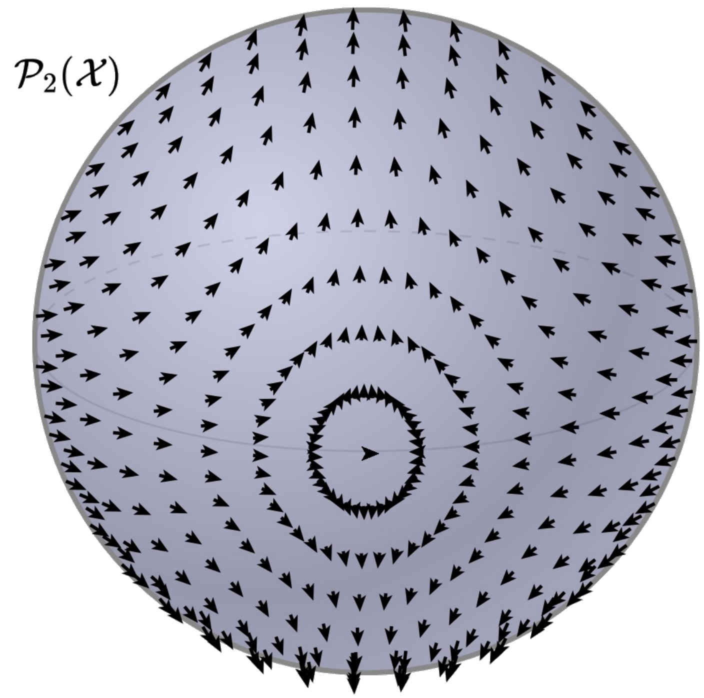


What if we have unpaired distributions?

📚 *Wasserstein Flow Matching.* Haviv, Pooladian, Pe'er, Amos. 2024.

Still can **learn a flow** between them

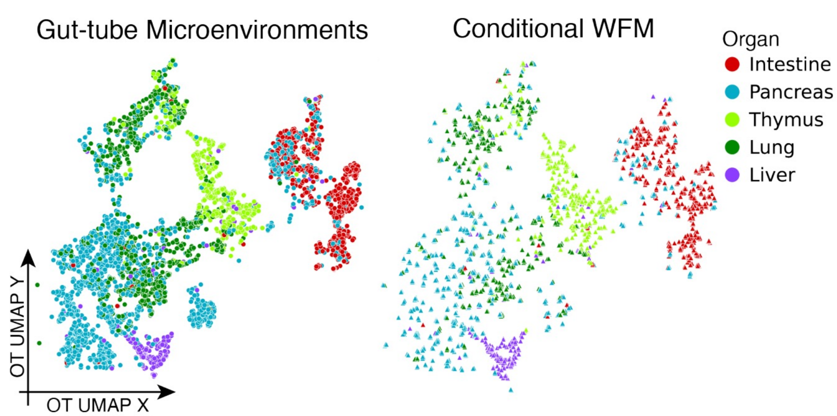




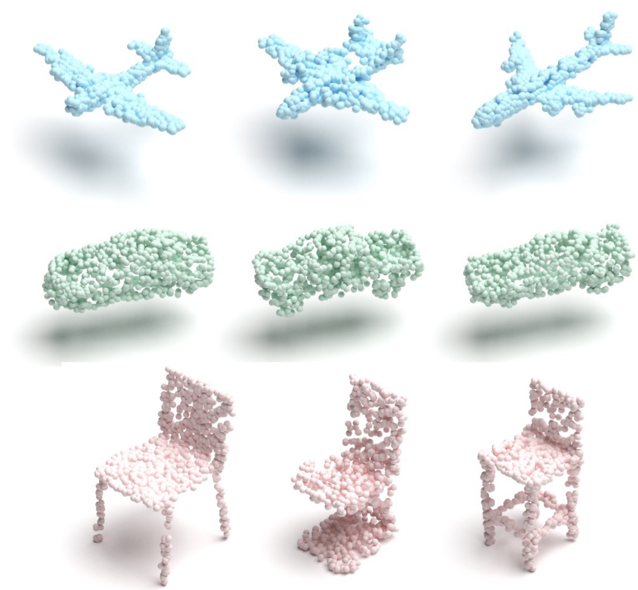
Why would we have unpaired distributions?

Want to do generative modeling where the data points are inherently distributions

biological data

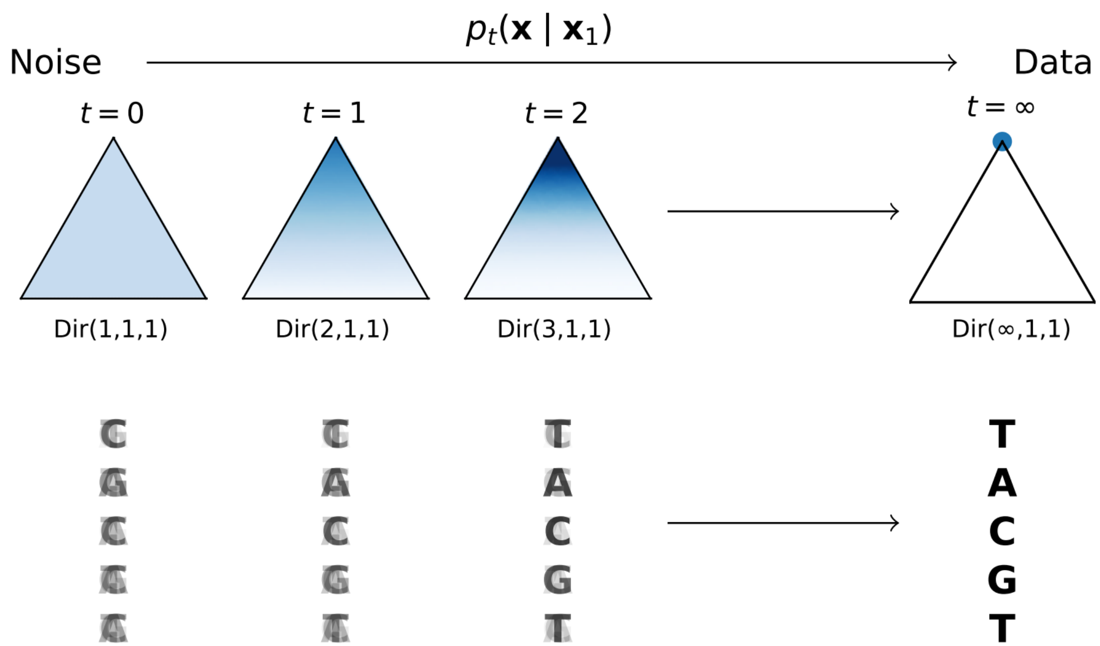


point clouds



Related: flows for categorical distributions

📚 *Dirichlet Flow Matching. Stark et al., NeurIPS 2024.*
📚 *Fisher Flow Matching. Davis et al., ICML 2024.*



How to flow on the Wasserstein manifold?

Riemannian Flow Matching. Chen and Lipman, ICLR 2024.

Use **Riemannian flow matching** with the geodesics (OT paths) on the Wasserstein manifold

Algorithm 1: Wasserstein FM Training

Data: base $\mathfrak{p}_0 \in \mathcal{P}(\mathcal{P}(\mathbb{R}^d))$, target $\mathfrak{p}_1 \in \mathcal{P}(\mathcal{P}(\mathbb{R}^d))$, $\texttt{geo} \in \{\mathrm{BW}, \mathrm{PC}\}$
Init: Parameters θ of $f_\theta^{\texttt{geo}}$
while *not converged* **do**
- Sample time $t \sim \mathcal{U}(0,1)$
- Sample source measure $\mu \sim \mathfrak{p}_0$
- Sample target measure $\nu \sim \mathfrak{p}_1$
- **if** *geo is BW* **then**
  - $\mu_t \leftarrow (m_t, \Sigma_t)$ via equation 8
  - $v_t \leftarrow (\dot{m}_t, \dot{\Sigma}_t^{\mathrm{BW}})$ via equation 9
- **else**
  - $\mu_t \leftarrow$ Approximate via equation 4
  - $v_t \leftarrow$ Approximate via equation 7
- $\ell(\theta) \leftarrow \|f_\theta^{\texttt{geo}}(\mu_t, t) - v_t\|^2_{L^2(\mu_t)}$
- $\theta \leftarrow \texttt{optimizer\_step}(\theta, \ell(\theta), \nabla_\theta \ell(\theta))$

Algorithm 2: $\mathrm{BW}(\mathbb{R}^d)$ generation

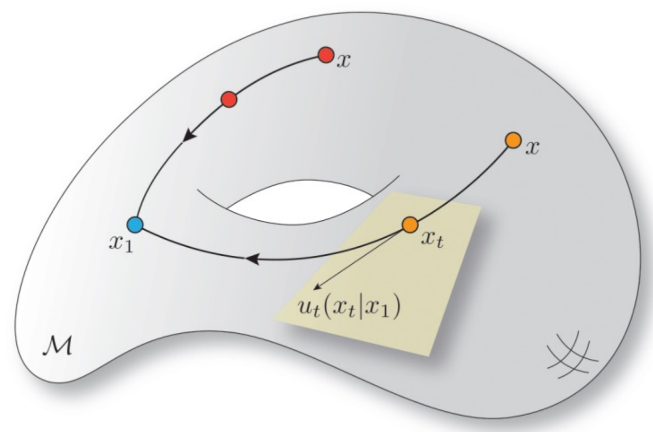
Data: Trained f_θ^{BW}, step size $h = 1/N$
Init: $\mathcal{N}(m_0, \Sigma_0) \sim \mathfrak{p}_0$
for $k = 0, \dots, N-1$ **do**
- $(s_k, S_k) \leftarrow f_\theta^{\mathrm{BW}}((m_{kh}, \Sigma_{kh}), kh)$
- $m_{(k+1)h} \leftarrow m_k + h s_k$
- $U_k \leftarrow (I + hS_k)$
- $\Sigma_{(k+1)h} \leftarrow U_k \Sigma_{kh} U_k$

Return: $\mathcal{N}(m_{Nh}, \Sigma_{Nh})$

Algorithm 3: Point-cloud generation

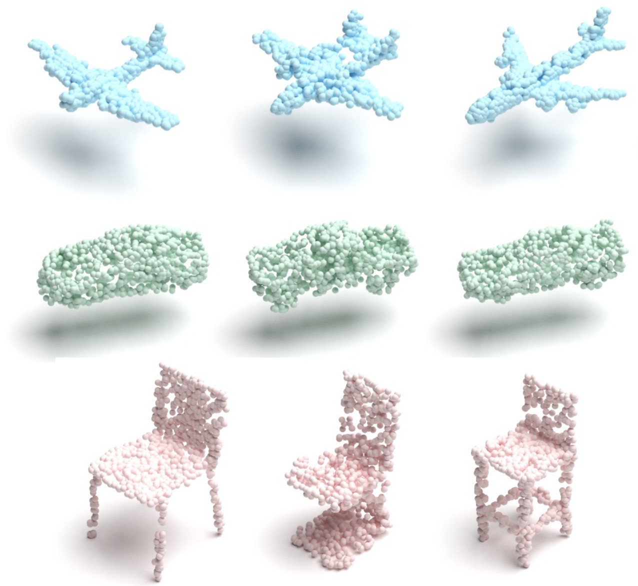
Data: Trained f_θ^{PC}, step size $h = 1/N$
Init: $\hat{\boldsymbol{X}}_0 = \{X_1, \dots, X_n\} \sim \mathfrak{p}_0$
for $k = 0, \dots, N-1$ **do**
- $\hat{\boldsymbol{X}}_{(k+1)h} \leftarrow \hat{\boldsymbol{X}}_{kh} + h f_\theta^{\mathrm{PC}}(\hat{\boldsymbol{X}}_{kh}, kh)$

Return: $\hat{\boldsymbol{X}}_{Nh}$



Point cloud generation results

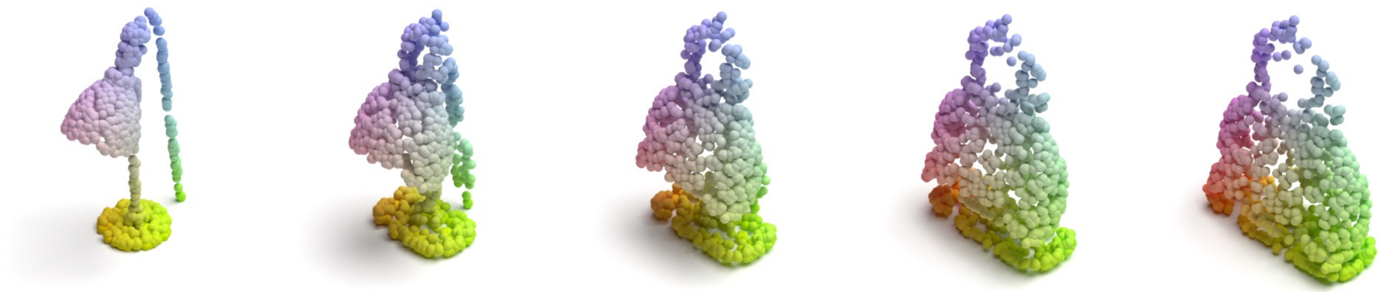
Competitive and **don't require spatially discretizing** the domain like most of the baselines



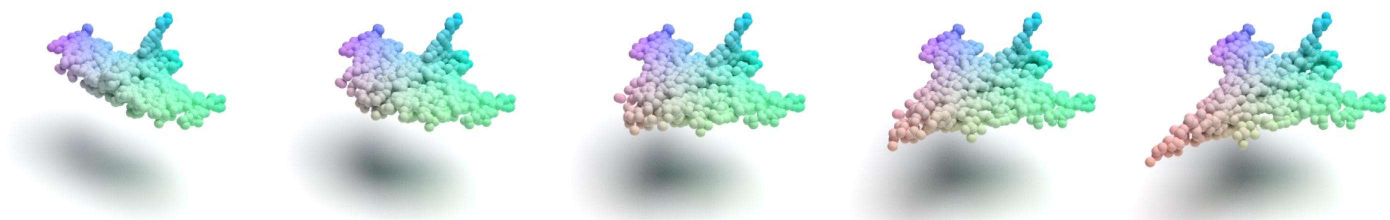
	Airplane		Chair		Car	
	CD ↓	EMD ↓	CD ↓	EMD ↓	CD ↓	EMD ↓
PointFlow	75.68	70.74	62.84	60.57	58.10	56.25
SoftFlow	76.05	65.80	59.21	60.05	64.77	60.09
DPF-Net	75.18	65.55	62.00	58.53	62.35	54.48
Shape-GF	80.00	76.17	68.96	65.48	63.20	56.53
PVD	73.82	64.81	56.26	53.32	54.55	53.83
PSF	71.11	61.09	58.92	54.45	57.19	56.07
WFM (ours)	73.45	71.72	58.98	57.77	56.53	57.95

WFM also does interpolations and completions

Flow from a **distribution over lamps** to a **distribution over handbags**



Completion using a flow trained over a **distribution of planes**

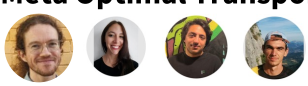


Transport and flows between distributions over distributions

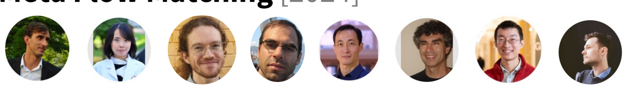
Brandon Amos • Meta, NYC

Primer on **amortized optimization** [Foundations and Trends in ML, 2023]

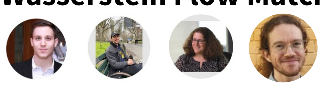
Meta Optimal Transport [ICML 2023]

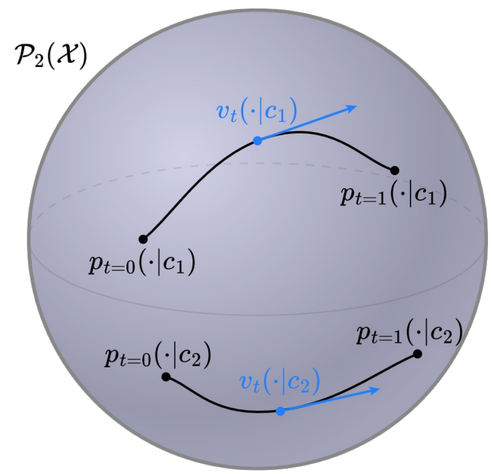


Meta Flow Matching [2024]



Wasserstein Flow Matching [2024]





slides

bamos.github.io/presentations