Tutorial on amortized optimization

github.com/facebookresearch/amortized-optimization-tutorial

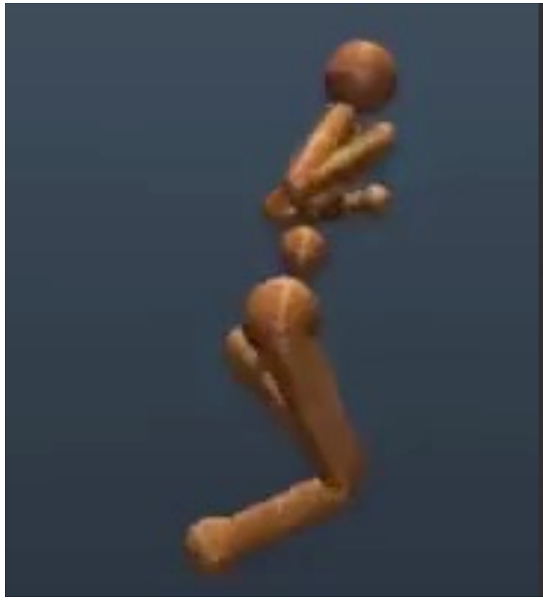
Brandon Amos
Meta AI NYC, Fundamental AI Research (FAIR)

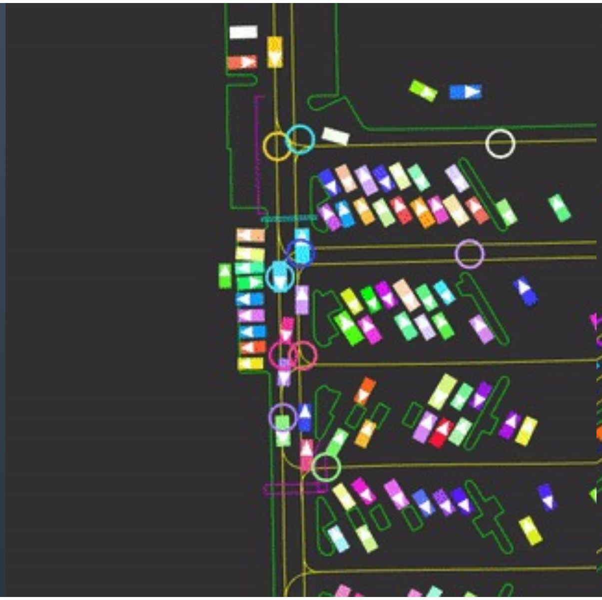
brandondamos bamos.github.io

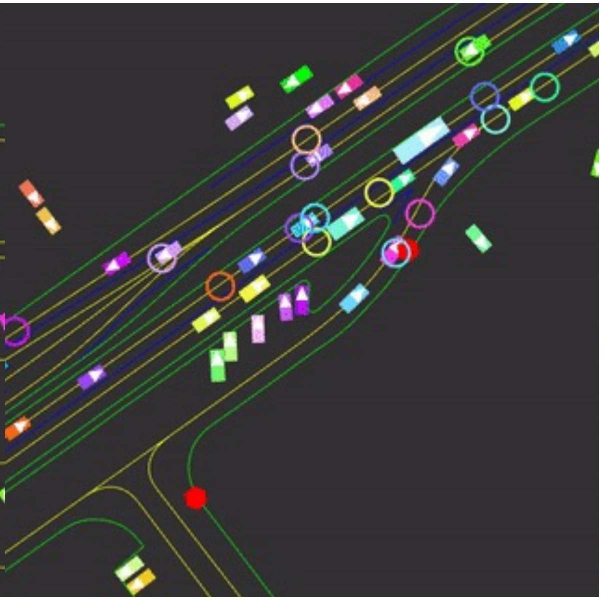
Collaborators: Noam Brown, Caroline Chen, Samuel Cohen, Arnaud Fickinger, Hengyuan Hu, Yann LeCun, Zeming Lin, Jason Liu, Giulia Luise, Joshua Meier, Ievgen Redko, Tom Sercu, Alexander Rives, Samuel Stanton, Shoba Venkataraman, Stuart Russel, Robert Verkuil, Andrew Gordon Wilson, Denis Yarats

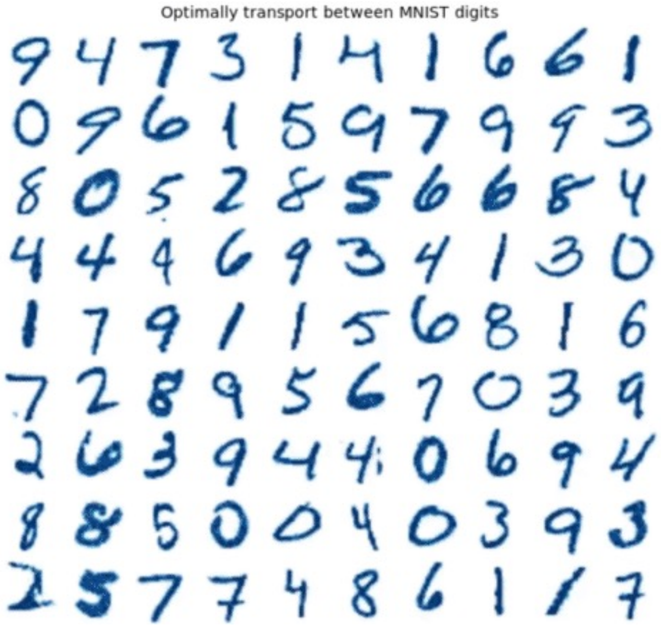
Optimization is a powerful modeling tool

Continuous optimization expresses many **non-trivial operations**
Control, reinforcement learning, robotics, geometry (projections), variational inference, finance (portfolio optimization), sparse coding, meta-learning, deep equilibrium networks, optimal transport, game and market equilibrium









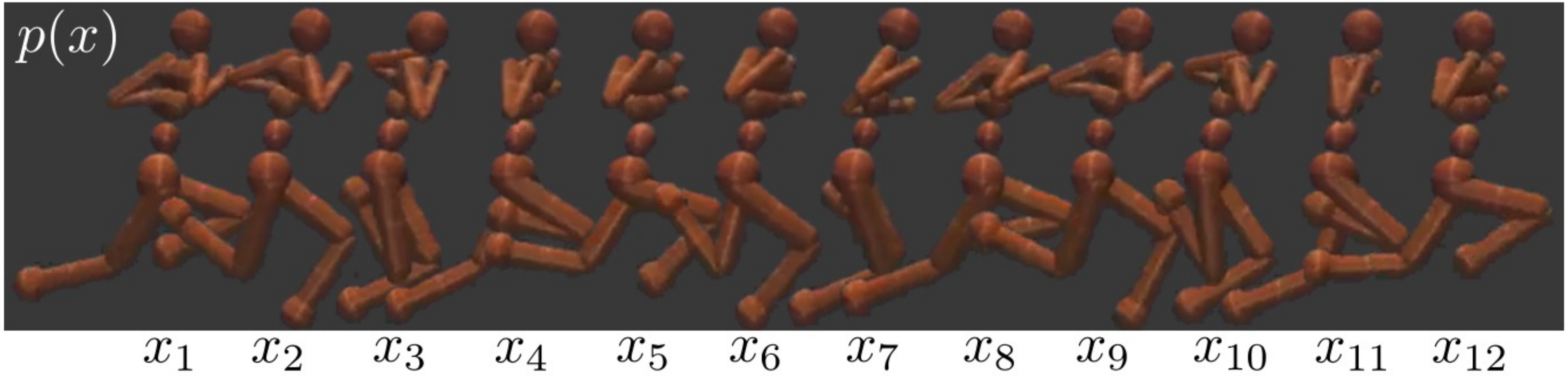
Repeatedly solving optimization problems

Optimization problems often **do not live in isolation** and are often **repeatedly solved** in deployment

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

Optimal solution | Optimization variable | Objective | Context

In control, x is the **system state**, y is the **action**, $f(y; x)$ is a **cost**, and $y^\star(x)$ is an **optimal action**



Difficulty: optimization is computationally expensive

Sometimes solving just **once** may be difficult
Exasperated when **repeatedly solving** during deployment

Insight: optimal solutions share structure

Optimization problems **share structure** and don't live in isolation

Solution: amortized optimization

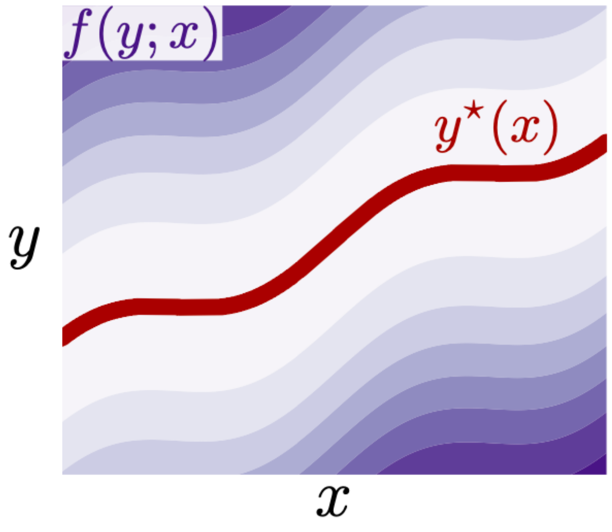
Use **machine learning** to uncover the shared structure
Create **learning-augmented** versions of classical optimization solvers
Far surpasses average or worst-case convergence rates

Also referred to as learning to optimize or data-driven optimization

Amortized optimization

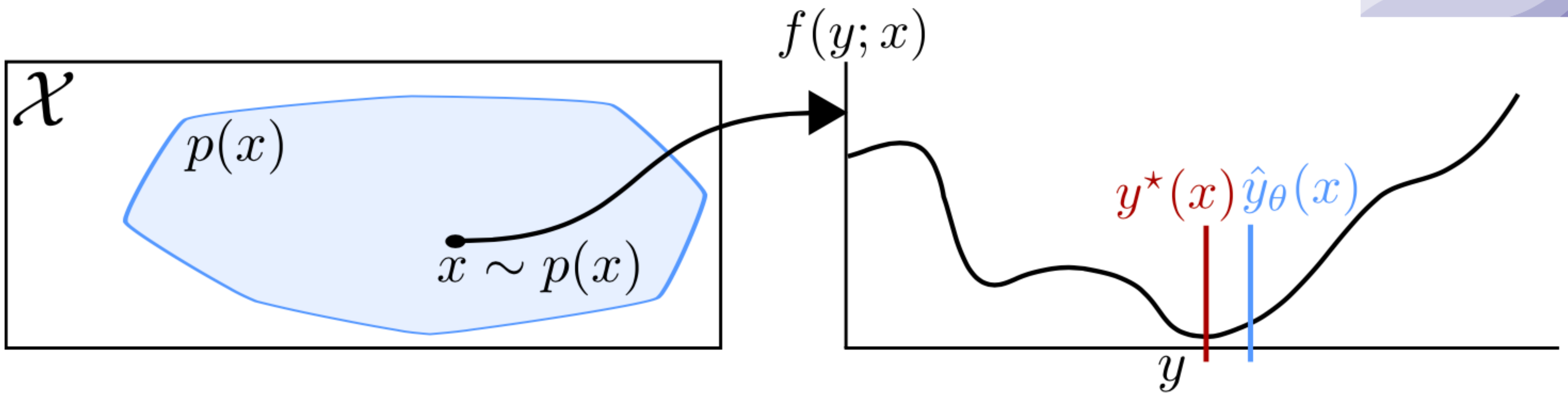
This talk: Explore **foundations** and **applications**

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



Amortization model $\hat{y}_\theta(x)$ tries to approximate $y^\star(x)$
Example: A neural network mapping from x to the solution

Loss $\mathcal{L}$ measures how well $\hat{y}$ fits $y^\star$ and optimized with $\min_\theta \mathcal{L}(\hat{y}_\theta)$
Example: $\mathcal{L}(\hat{y}_\theta) \coloneqq \mathbb{E}_{p(x)} \|\hat{y}_\theta(x) - y^\star(x)\|_2^2$



Amortized optimization is well-explored

My goal: characterize and **connect** applications **previously developed independently**

Variational inference	6.1	VAE	– ELBO	variational posterior	data	full	$\mathcal{L}_{\rm obj}$
		SAVAE/IVAE	$\vert$	$\vert$	$\vert$	semi	$\vert$
Sparse coding	6.2	PSD	reconstruction	sparse code	data	full	$\mathcal{L}_{\rm reg}$
		LISTA	$\vert$	$\vert$	$\vert$	semi	$\vert$
Meta-learning	6.3	HyperNets	task loss	model parameters	tasks	full	$\mathcal{L}_{\rm obj}$
		LM	$\vert$	$\vert$	$\vert$	semi	$\mathcal{L}_{\rm obj}^{\rm RL}$
		MAML	$\vert$	$\vert$	$\vert$	$\vert$	$\mathcal{L}_{\rm obj}$
		Neural Potts	pseudo-likelihood	$\vert$	protein sequences	full	$\mathcal{L}_{\rm obj}$
Fixed-points and convex optimization	6.4	NeuralFP	FP residual	FP iterates	FP contexts	semi	$\mathcal{L}_{\rm obj}^{\Sigma}$
		HyperAA	$\vert$	$\vert$	$\vert$	$\vert$	$\mathcal{L}_{\rm reg}^{\Sigma}$
		NeuralSCS	CP residual	CP iterates	CP parameters	$\vert$	$\mathcal{L}_{\rm obj}^{\Sigma}$
		HyperDEQ	DEQ residual	DEQ iterates	DEQ parameters	$\vert$	$\mathcal{L}_{\rm reg}^{\Sigma}$
		NeuralNMF	NMF residual	factorizations	input matrices	$\vert$	$\mathcal{L}_{\rm obj}^{\Sigma}$
		RLQP	$R_{\rm RLQP}$	QP iterates	QP parameters	$\vert$	$\mathcal{L}_{\rm obj}^{\rm RL}$
Optimal transport	6.5	AmorConj	c-transform obj	supp(α)	supp(β)	full	$\mathcal{L}_{\rm obj}$
		$\mathcal{A}$-SW	max-sliced dist	slices Θ	mini-batches	$\vert$	$\mathcal{L}_{\rm obj}$
		Meta OT	dual OT cost	optimal couplings	input measures	$\vert$	$\mathcal{L}_{\rm obj}$
Reinforcement learning	6.6	BC/IL	$-Q$-value	controls	state space	full	$\mathcal{L}_{\rm reg}$
		(D)DPG/TD3	$\vert$	$\vert$	$\vert$	$\vert$	$\mathcal{L}_{\rm obj}$
		PILCO	$\vert$	$\vert$	$\vert$	$\vert$	$\mathcal{L}_{\rm obj}$
		POPLIN	$\vert$	$\vert$	$\vert$	full or semi	$\mathcal{L}_{\rm reg}$
		DCEM	$\vert$	$\vert$	$\vert$	semi	$\mathcal{L}_{\rm reg}$
		IAPO	$\vert$	$\vert$	$\vert$	$\vert$	$\mathcal{L}_{\rm obj}$
		SVG	D_Q or $-\mathcal{E}_Q$	control dists	$\vert$	full	$\mathcal{L}_{\rm obj}$
		SAC	$\vert$	$\vert$	$\vert$	$\vert$	$\mathcal{L}_{\rm obj}$
		GPS	$\vert$	$\vert$	$\vert$	$\vert$	$\mathcal{L}_{\rm KL}$

This talk: amortized optimization

Design decisions

Modeling paradigms for $\hat{y}_\theta$ (fully-amortized and semi-amortized models)
Learning paradigms for $\mathcal{L}$ (objective-based and regression-based)

Selected applications

Reinforcement learning
Neural Potts Model for protein modeling
Meta optimal transport

This talk: amortized optimization

Design decisions

Modeling paradigms for $\hat{y}_\theta$ (fully-amortized and semi-amortized models)
Learning paradigms for $\mathcal{L}$ (objective-based and regression-based)

Selected applications

Reinforcement learning
Neural Potts Model for protein modeling
Meta optimal transport

Modeling paradigms for $\hat{y}_\theta$

How to best-predict the solution?

Fully-amortized models: Map from the context x to the solution **without** accessing the objective f
Example: Neural network mapping from x to the solution
Most of our applications will focus on these

Semi-amortized models: Internally access the objective f
Example: Gradient-based meta-learning models such as MAML

$$\hat{y}_\theta^0 \rightarrow \hat{y}_\theta^1 \rightarrow \cdots \rightarrow \hat{y}_\theta^K =: \hat{y}_\theta(x)$$

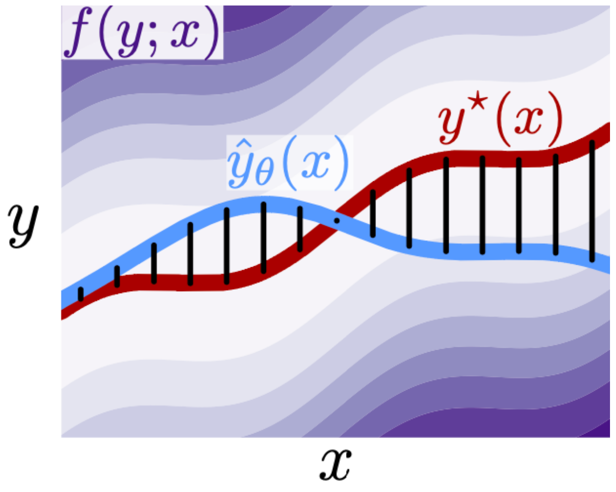
Learning paradigms for $\mathcal{L}$

What should the model $\hat{y}_\theta$ optimize for?

Regression-based

$\mathcal{L}_{\mathrm{reg}}(\hat{y}_\theta) \coloneqq \mathbb{E}_{p(x)} \|\hat{y}_\theta(x) - y^\star(x)\|_2^2$

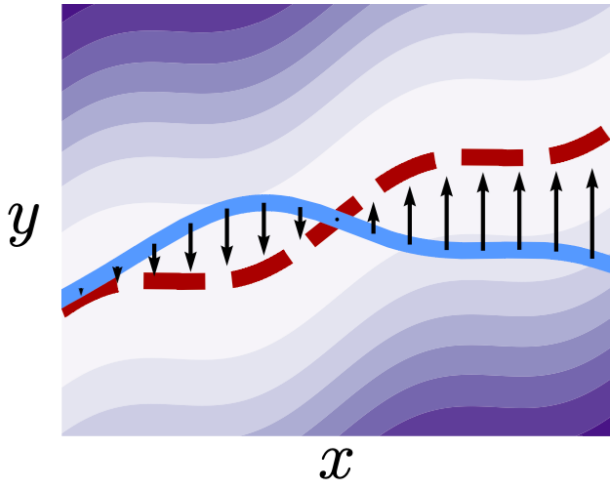
- $-$ Does not consider $f(y; x)$
- $+$ Uses global information with $y^\star(x)$
- $-$ Expensive to compute $y^\star(x)$
- $+$ Does not compute $\nabla_y f(y; x)$
- $-$ Hard to learn non-unique $y^\star(x)$



Objective-based:

$\mathcal{L}_{\mathrm{obj}}(\hat{y}_\theta) \coloneqq \mathbb{E}_{p(x)} f(\hat{y}_\theta(x); x)$

- $+$ Uses objective information of $f(y; x)$
- $-$ Can get stuck in local optima of $f(y; x)$
- $+$ Faster, does not require $y^\star(x)$
- $-$ Often requires computing $\nabla_y f(y; x)$
- $+$ Easily learns non-unique $y^\star(x)$



This talk: amortized optimization

Design decisions

- **Modeling** paradigms for $\hat{y}_\theta$ (fully-amortized and semi-amortized models)
- **Learning** paradigms for $\mathcal{L}$ (objective-based and regression-based)

Selected applications

- Reinforcement learning
- Neural Potts Model for protein modeling
- Meta optimal transport

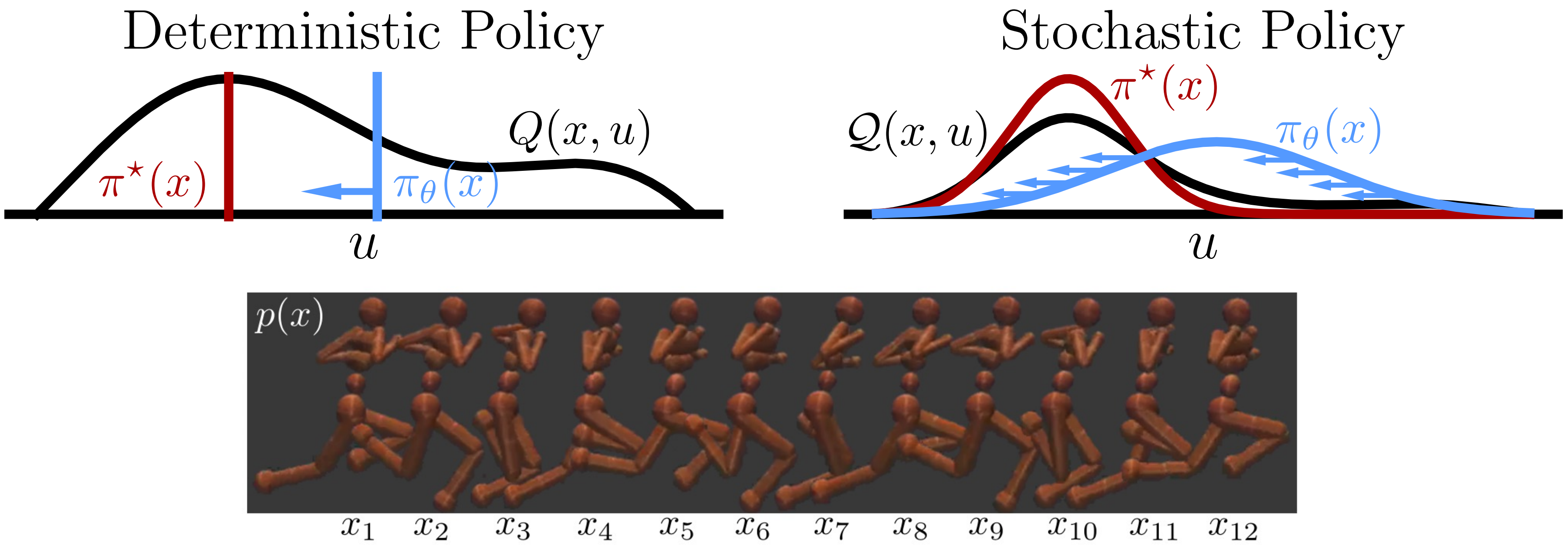
RL policy learning is amortized optimization

Setup: controlling a **continuous MDP** with a **model-free policy** $\pi_\theta(x)$

Review: Learning a policy with a **value gradient** amortizes over the Q-value:

$$\operatorname*{argmax}_\theta \mathbb{E}_{p(x)}\, Q(x, \pi_\theta(x))$$

The **amortization perspective** easily enables **expanding beyond this fully-amortized setting**



Semi-amortized policy learning

Iterative Amortized Policy Optimization

Joseph Marino*
California Institute of Technology

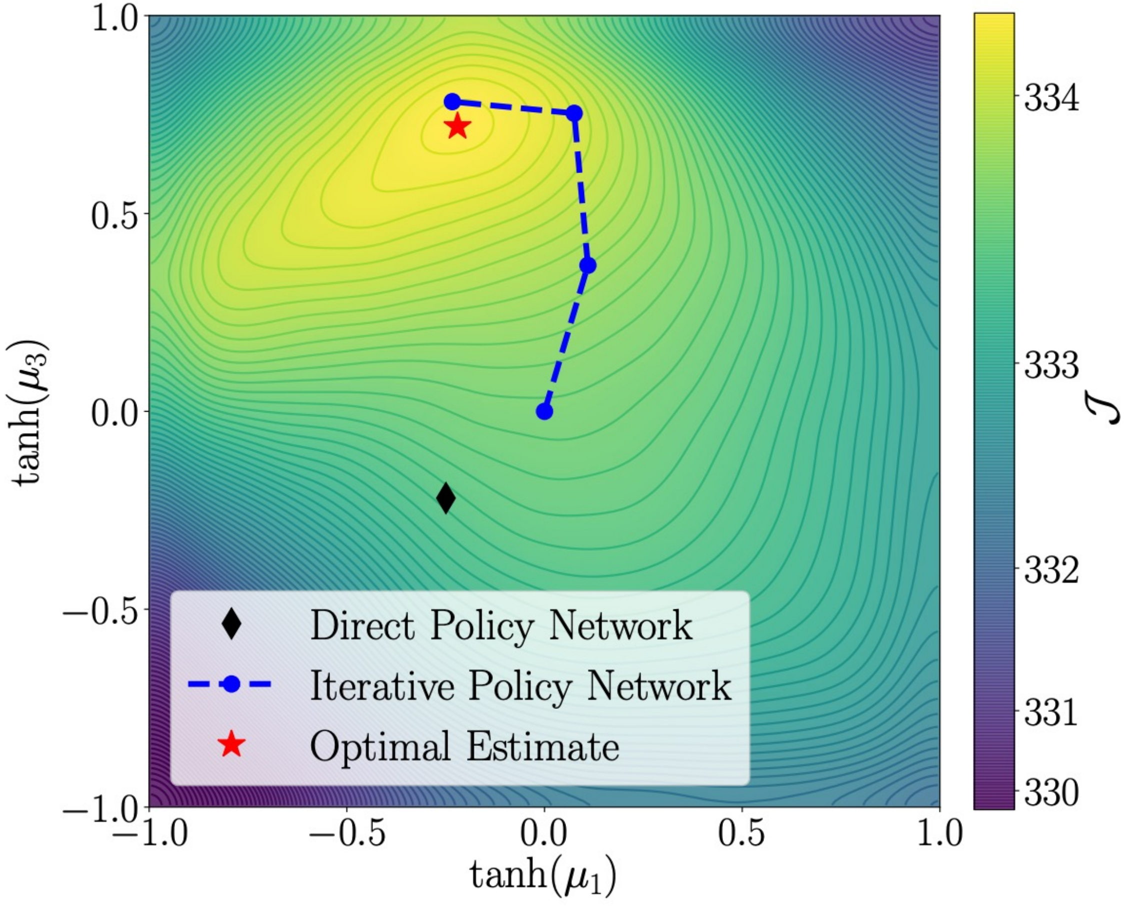
Alexandre Piché
Mila, Université de Montréal

Alessandro Davide Ialongo
University of Cambridge

Yisong Yue
California Institute of Technology

Abstract

Policy networks are a central feature of deep reinforcement learning (RL) algorithms for continuous control, enabling the estimation and sampling of high-value actions. From the variational inference perspective on RL, policy networks, when used with entropy or KL regularization, are a form of *amortized optimization*, optimizing network parameters rather than the policy distributions directly. However, *direct* amortized mappings can yield suboptimal policy estimates and restricted distributions, limiting performance and exploration. Given this perspective, we consider the more flexible class of *iterative* amortized optimizers. We demonstrate that the resulting technique, iterative amortized policy optimization, yields performance improvements over direct amortization on benchmark continuous control tasks. Accompanying code: `github.com/joelouismarino/variational_rl`.



Decision-time fine-tuning and planning

Rely on **standard policy methods** such as PPO to **amortize the solution to a game**
Fine-tune at decision time by **constraining to the initial state** and **continuing policy optimization**

Scalable Online Planning via Reinforcement Learning Fine-Tuning

Arnaud Fickinger* Facebook AI Research arnaudfickinger@fb.com

Hengyuan Hu* Facebook AI Research hengyuan@fb.com

Brandon Amos Facebook AI Research bda@fb.com

Stuart Russell UC Berkeley russell@berkeley.edu

Noam Brown Facebook AI Research noambrown@fb.com

Hanabi scores

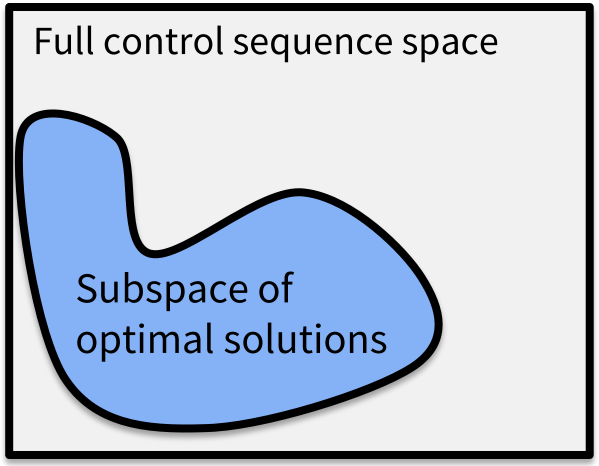
Variant	Blueprint	SPARTA (Single)	SPARTA (Multi)	RL Search (Single)	RL Search (Multi)
Normal	24.23 ± 0.04 63.20%	24.57 ± 0.03 73.90%	24.61 ± 0.02 75.46%	24.59 ± 0.02 75.05%	**24.62 ± 0.03** **75.93%**
2 Hints	22.99 ± 0.04 17.50%	23.60 ± 0.03 25.85%	23.67 ± 0.03 26.87%	23.61 ± 0.03 27.85%	**23.76 ± 0.04** **31.01%**

Ms. Pacman scores

Additional Samples	0	3.10^5	4.10^5	8.10^5
RL Fine-Tuning	1880	**3940**	**4580**	**5510**
PPO Training	1880	1900	1900	1920

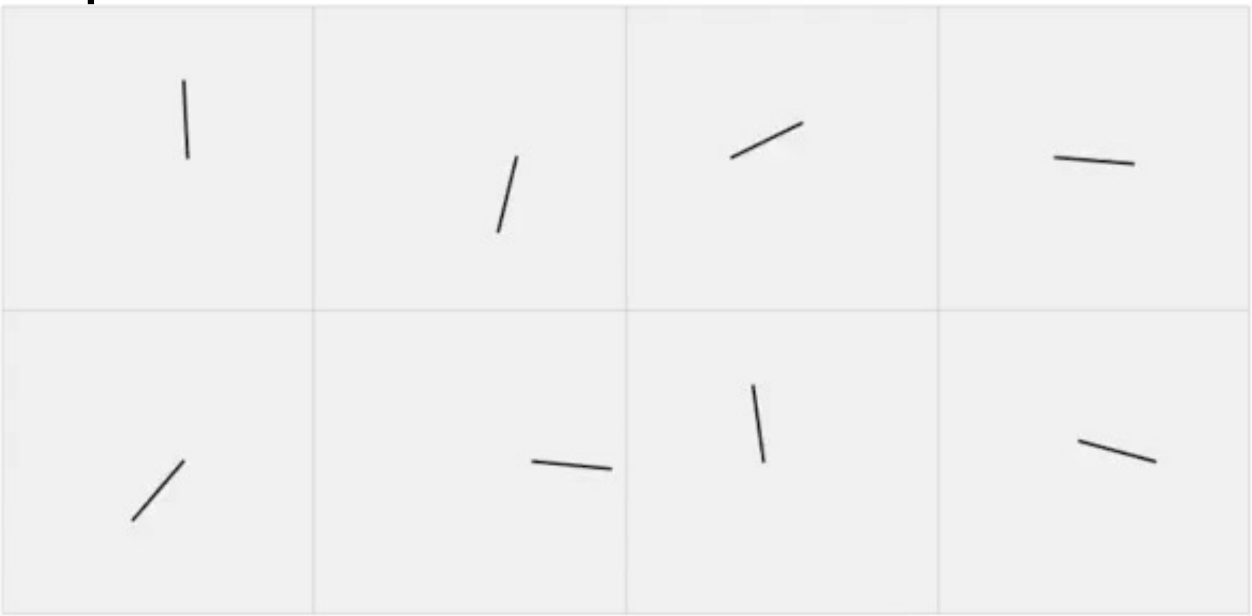
Amortization via learning latent subspaces

Amortize the problem by **learning a latent subspace** of optimal solutions
Only search over optimal solutions rather than the entire space



$$x^\star_{1:T}, u^\star_{1:T} \in \underset{x_{1:T}, u_{1:T}}{\mathrm{argmin}} \sum_t \underbrace{\boxed{C_\theta(x_t, u_t)}}_{\text{cost}} \text{ s.t. } \underbrace{\boxed{x_1 = x_{\rm init}}}_{\text{initial state}} \; \underbrace{\boxed{x_{t+1} = f_\theta(x_t, u_t)}}_{\text{dynamics}} \; \underbrace{\boxed{u_t \in \mathcal{U}}}_{\text{constraints}}$$

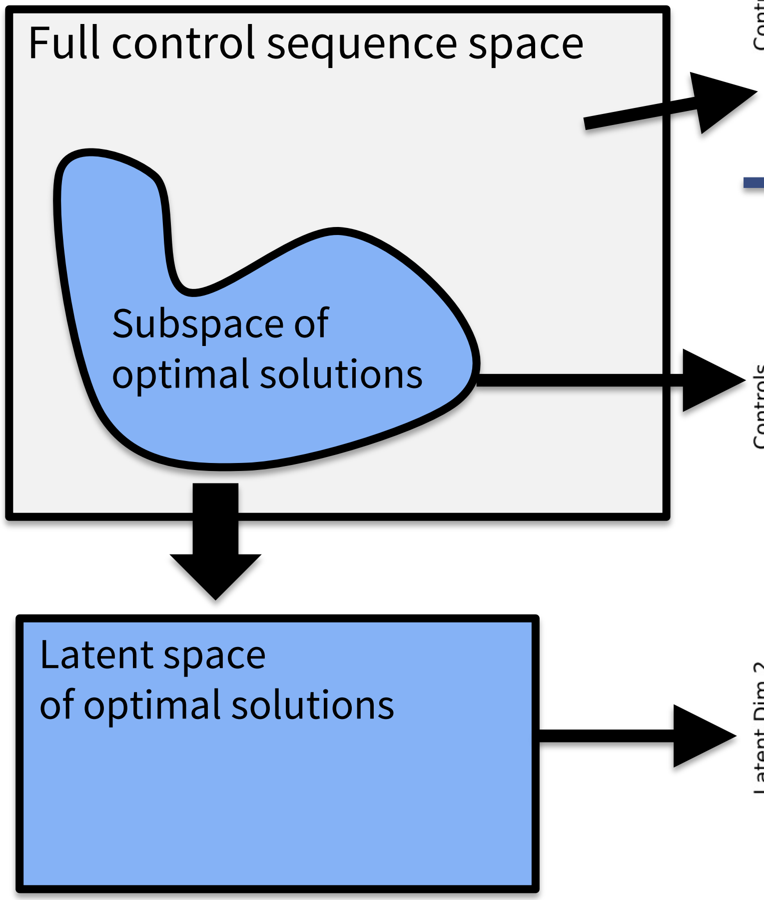
Cartpole videos

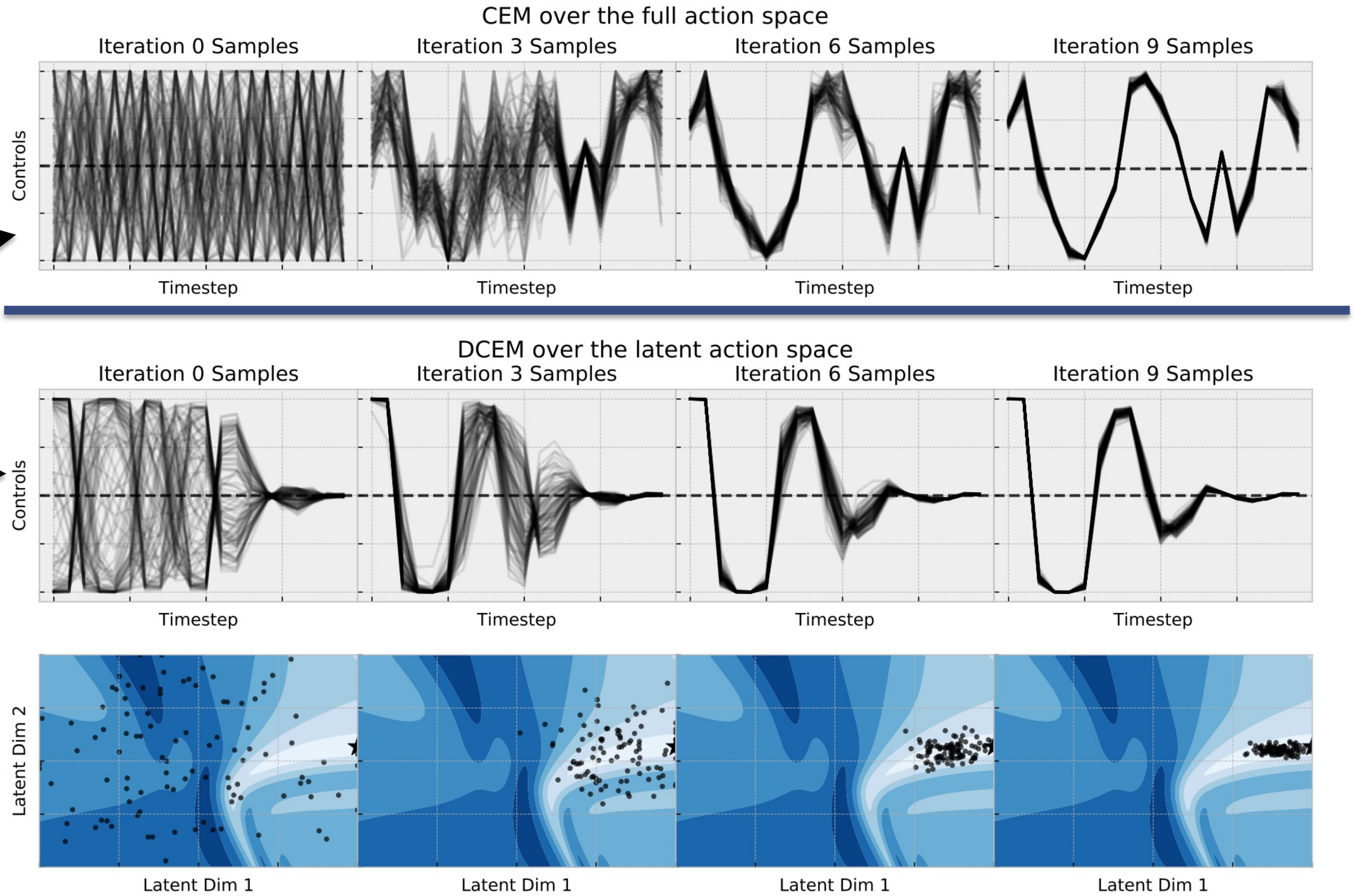


Optimal controls over time — force on the cartpole

Amortization via learning latent subspaces

$$u^\star = \underset{u \in [0,1]^N}{\operatorname{argmin}} \; f(u)$$

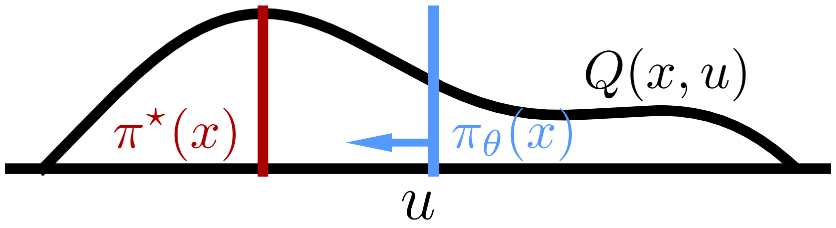




VAE amortization is conceptually the same as RL

Value gradient amortization in RL

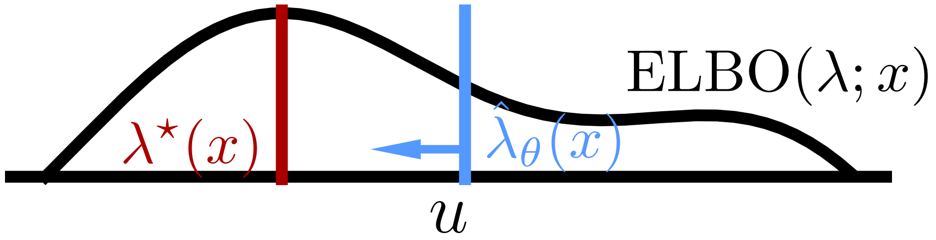
$$\underset{\theta}{\operatorname{argmax}}\ \mathbb{E}_{p(x)}\ Q(x, \pi_\theta(x))$$



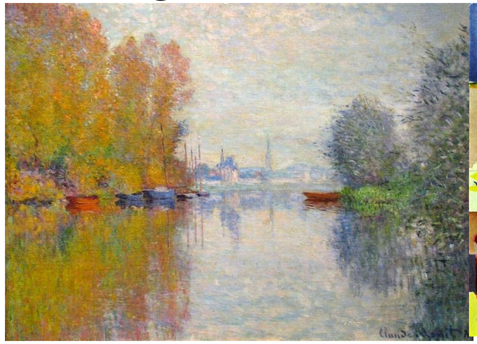
x: states from system

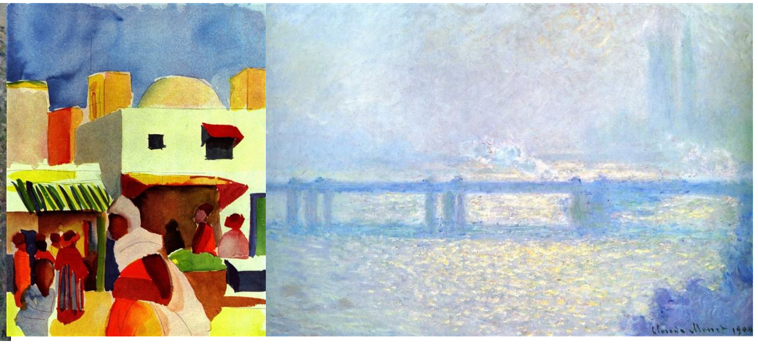
VAE posterior amortization

$$\underset{\theta}{\operatorname{argmax}}\ \mathbb{E}_{p(x)}\ \mathrm{ELBO}(\hat{\lambda}_\theta(x); x)$$



x: images from dataset





This talk: amortized optimization

Design decisions

- **Modeling** paradigms for $\hat{y}_\theta$ (fully-amortized and semi-amortized models)
- **Learning** paradigms for $\mathcal{L}$ (objective-based and regression-based)

Selected applications

- Reinforcement learning
- Neural Potts Model for protein modeling
- Meta optimal transport

Neural Potts Model

Potts model: A **Gibbs distribution** over a **protein sequence** x:

$$-E(x) \overset{\text{def}}{=} \sum_i h_i(x_i) + \sum_{ij} J_{ij}(x_i, x_j)$$

parameterized by $W = \{h, J\}$

Standard Potts (baseline)

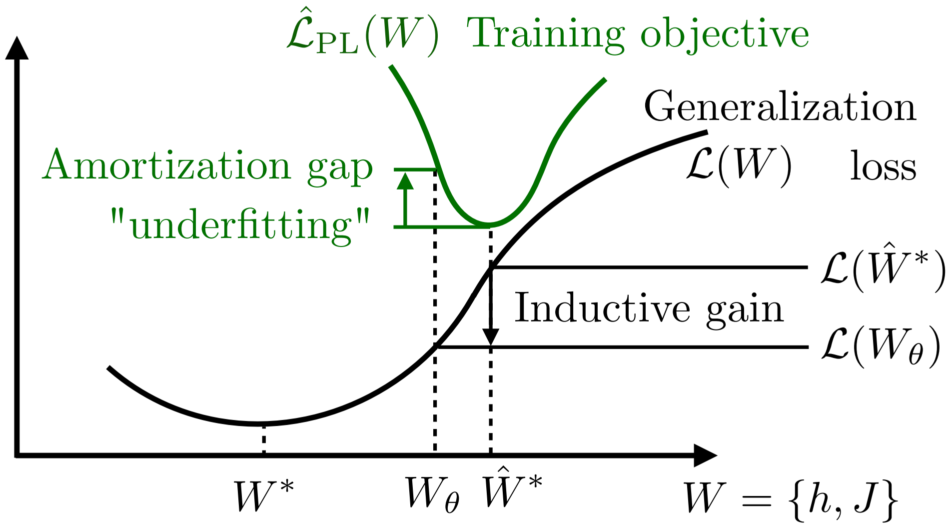
Independently optimize the likelihood for every sequence

$\boldsymbol{x}$ → Search & align → MSA → Optimize Potts model → $\hat{W}^*$

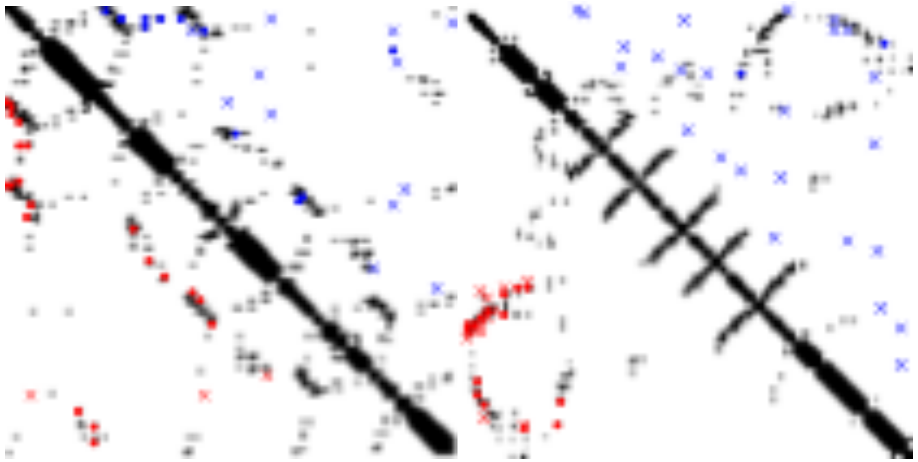
Neural Potts

Amortize and jointly optimize for all sequences at once

$\boldsymbol{x}$ → NPM forward pass → $W_\theta(\boldsymbol{x})$



Contact predictions



This talk: amortized optimization

Design decisions

Modeling paradigms for $\hat{y}_\theta$ (fully-amortized and semi-amortized models)
Learning paradigms for $\mathcal{L}$ (objective-based and regression-based)

Selected applications

Reinforcement learning
Neural Potts Model for protein modeling
Meta optimal transport

Meta Optimal Transport

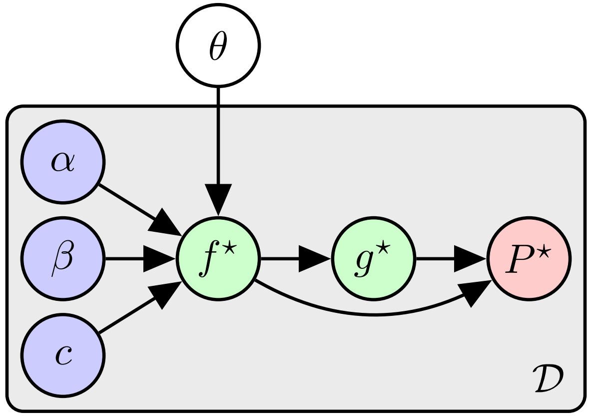
Goal: optimally transport mass between measures α to β

$$\pi^\star(\alpha, \beta, c) \in \operatorname*{argmin}_{\pi \in \mathcal{U}(\alpha,\beta)} \int_{\mathcal{X}\times\mathcal{Y}} c(x,y) \mathrm{d}\pi(x,y)$$

Use **amortization** when **repeatedly coupling measures**
e.g., between pairs of images or physical transport

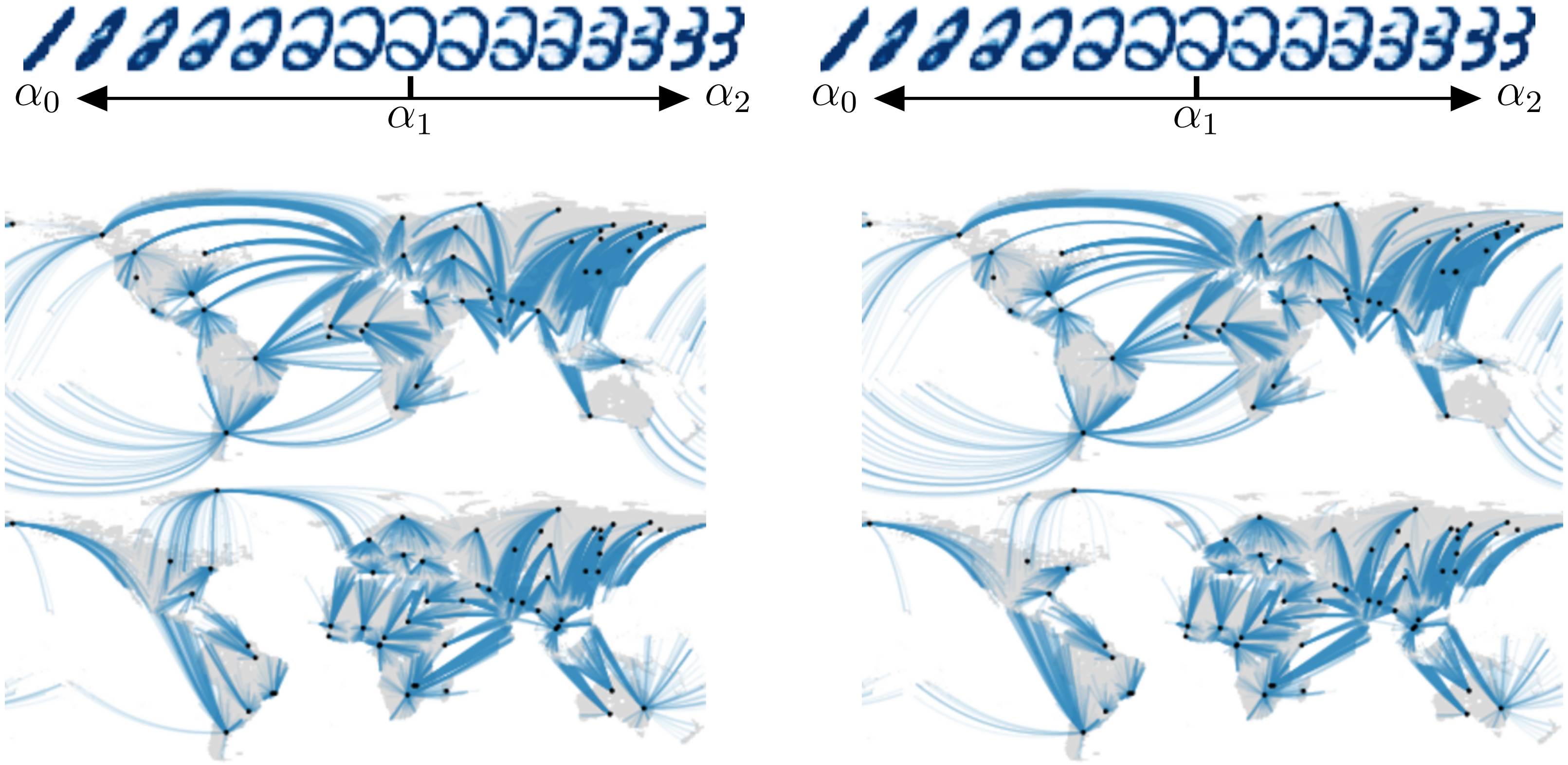
Often amortize over **unconstrained dual potentials**

Meta OT for Sinkhorn

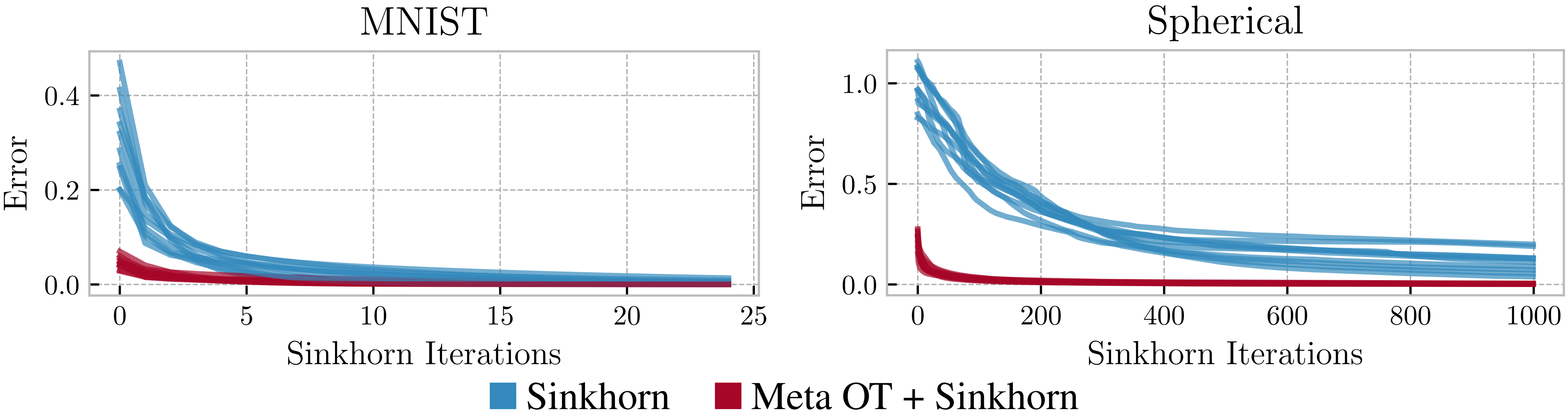


Discrete (Entropic)

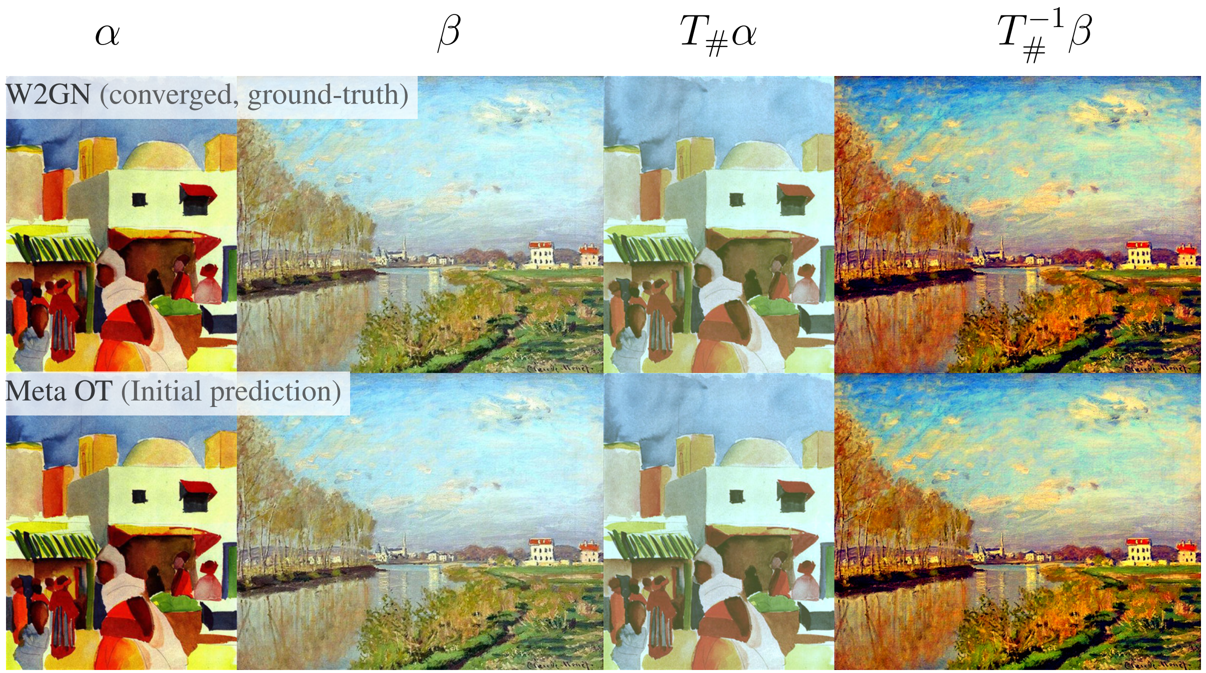
Sinkhorn (converged, ground-truth) | Meta OT (initial prediction)

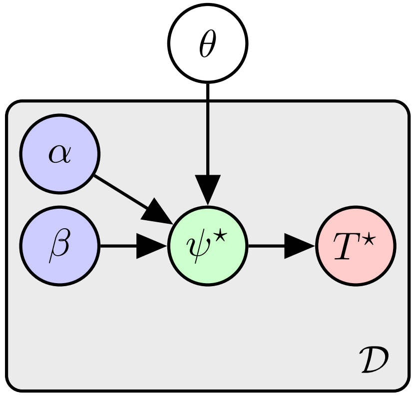


Meta OT significantly improves Sinkhorn

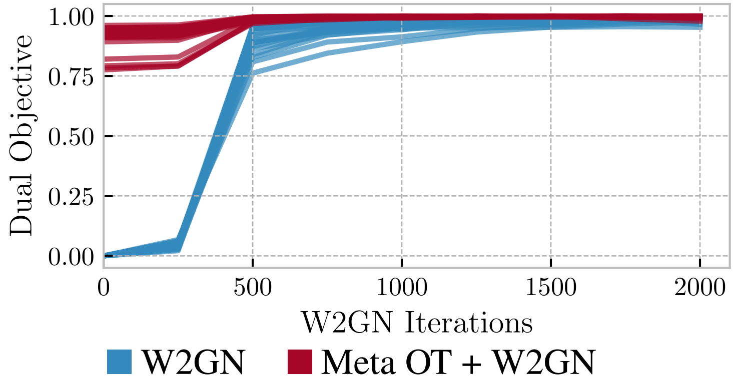


Meta OT for Continuous OT with Meta ICNNs

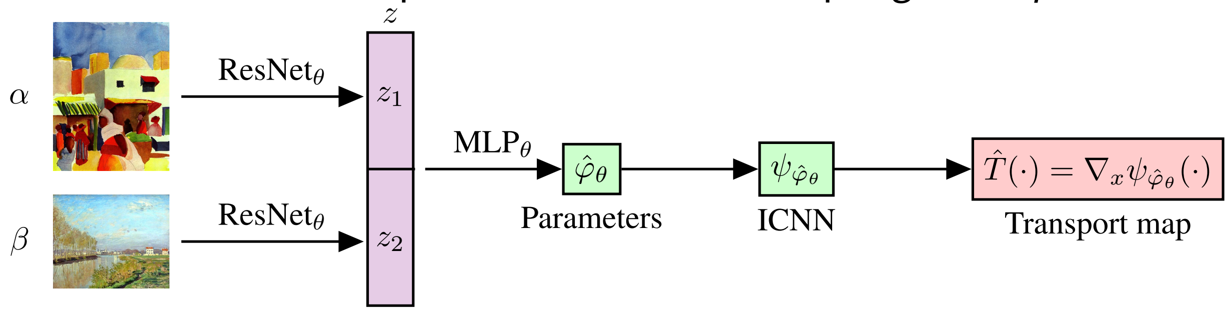




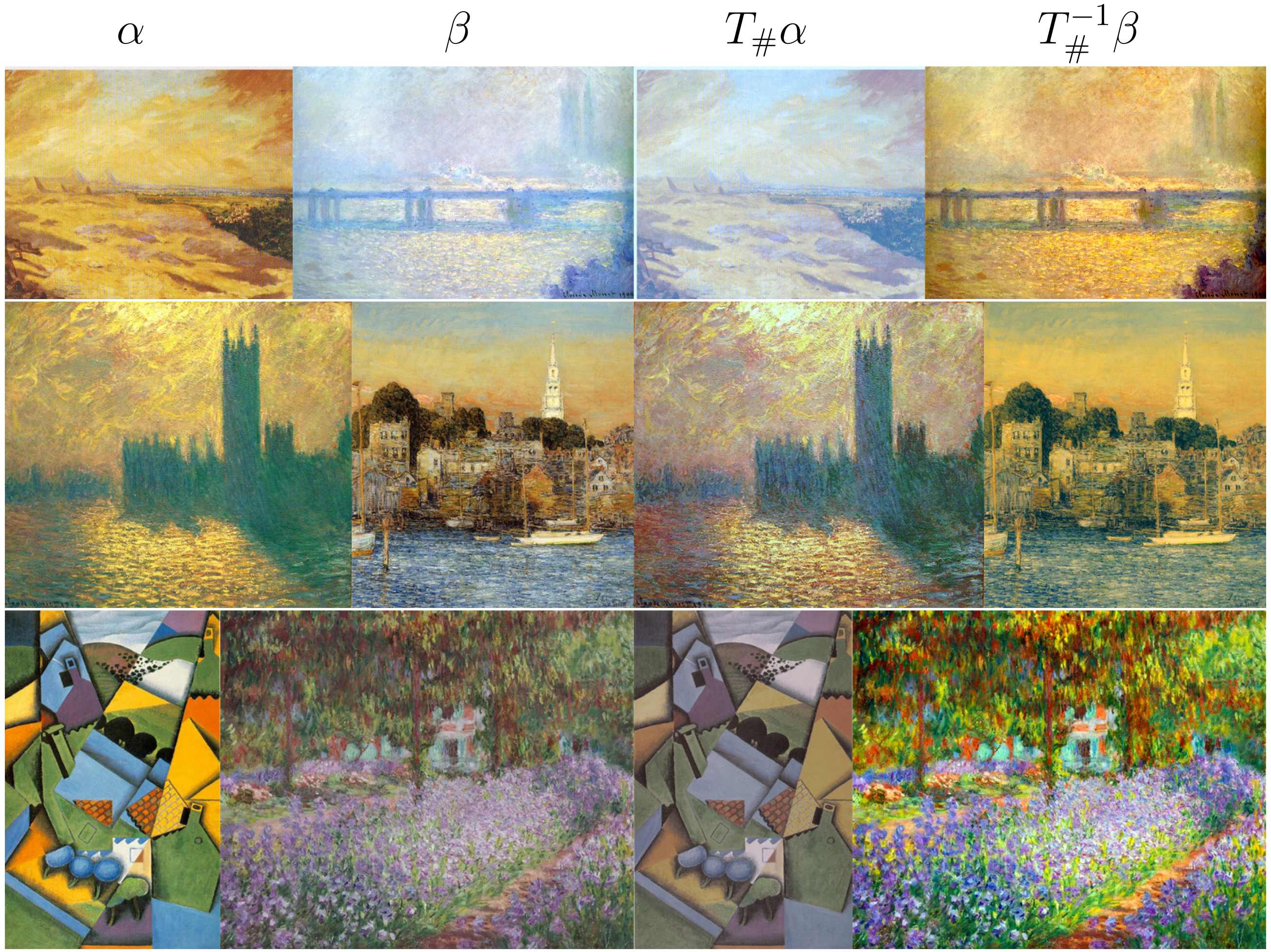
Continuous (Wasserstein-2)

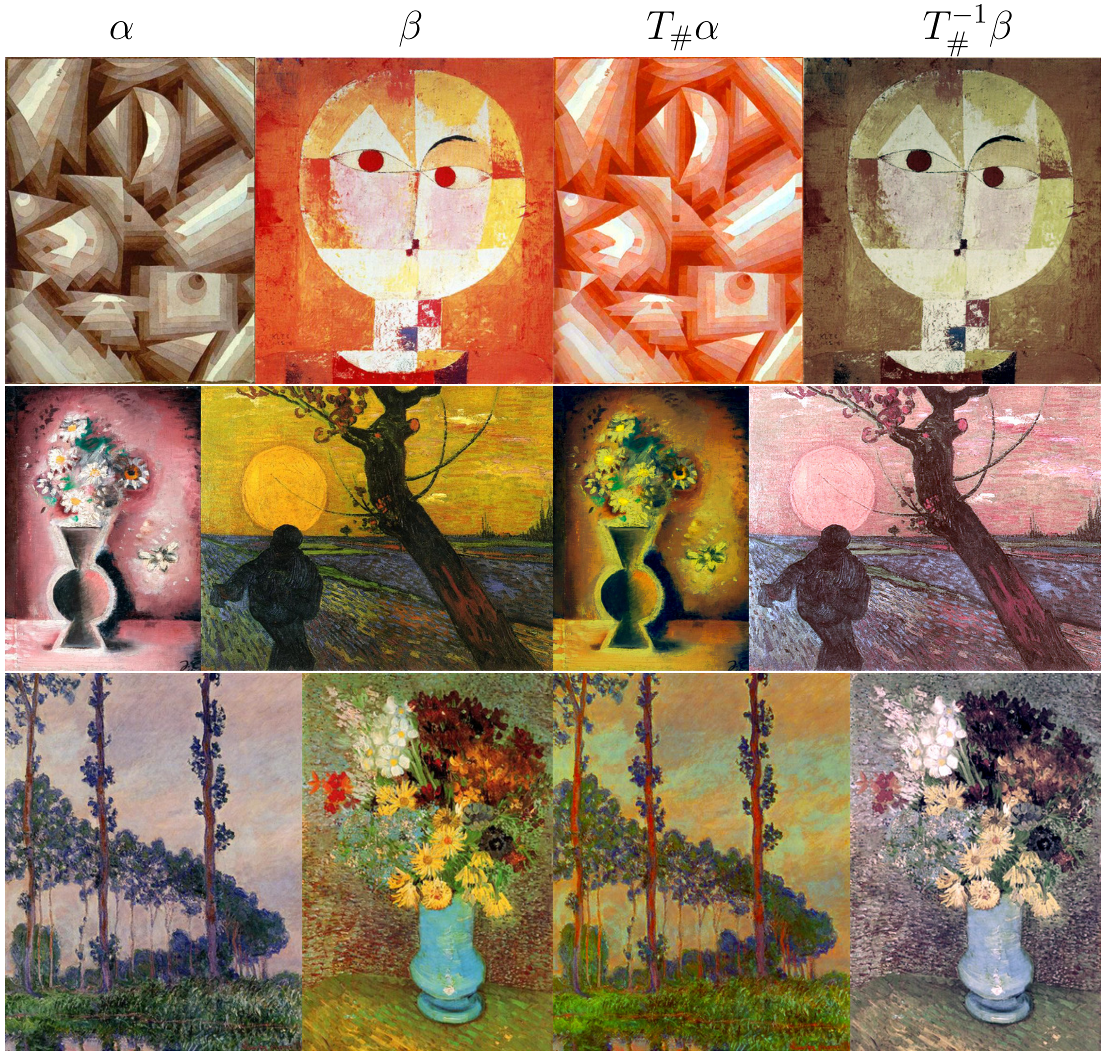


Meta ICNN: Predict parameters of ICNN coupling α and β



More Meta OT color transfer predictions

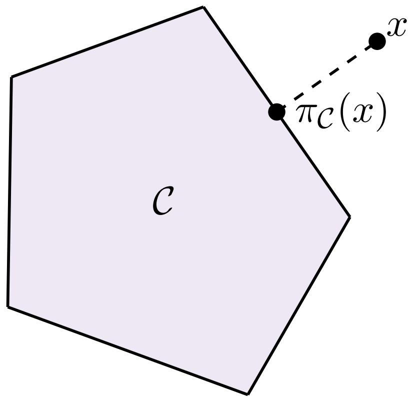




Future directions and limitations

Amortized optimization is established and budding with new methods and applications

Possible to expand far beyond **unconstrained continuous Euclidean optimization** settings:
1. **New applications and settings for semi-amortized modeling**
2. **Constrained domains** (e.g., with differentiable projections)
3. **Discrete optimization settings** (e.g., with differentiable discrete optimization)
4. **Non-Euclidean settings** (e.g., with Riemannian optimization)



Potential limitations:
1. Difficult in **out-of-domain settings** when the contexts significantly change
2. Generally difficult to **ensure stability or convergence**
3. Typically **does not solve previously intractable problems**
4. Can be **difficult to obtain high-accuracy solutions** without fine-tuning/semi-amortization

Tutorial on amortized optimization

github.com/facebookresearch/amortized-optimization-tutorial

Brandon Amos
Meta AI NYC, Fundamental AI Research (FAIR)

brandondamos bamos.github.io

The differentiable cross-entropy method [Amos and Yarats, ICML 2020]
Neural Potts Model [Sercu*, Verkuil*, et al., MLCB 2020]
On the model-based stochastic value gradient [Amos, Stanton, Yarats, Wilson, L4DC 2021]
Online planning via RL fine-tuning [Fickinger*, Hu*, et al., NeurIPS 2021]
Neural fixed-point acceleration [Venkataraman and Amos, ICML AutoML Workshop, 2021]
Meta Optimal Transport [Amos, Cohen, Luise, Redko, arXiv 2022]
Tutorial on amortized optimization [Amos, arXiv 2022]

Collaborators: Noam Brown, Caroline Chen, Samuel Cohen, Arnaud Fickinger, Hengyuan Hu, Yann LeCun, Zeming Lin, Jason Liu, Giulia Luise, Joshua Meier, Ievgen Redko, Tom Sercu, Alexander Rives, Samuel Stanton, Shoba Venkataraman, Stuart Russel, Robert Verkuil, Andrew Gordon Wilson, Denis Yarats