Tutorial on amortized optimization for learning to optimize over continuous spaces

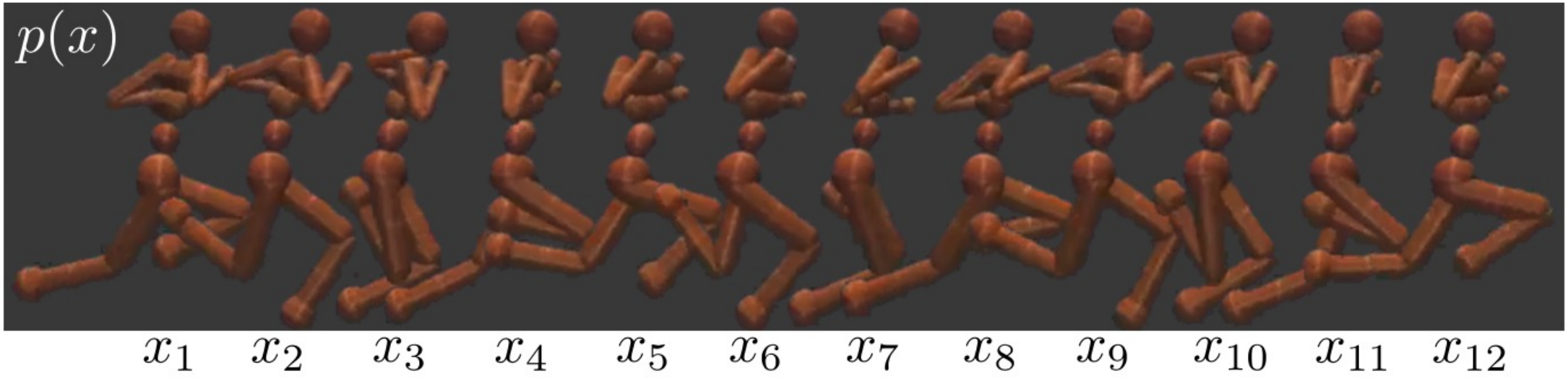
Brandon Amos, Meta AI (FAIR) brandondamos bamos.github.io

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

Application: Continuous control of a robotic system:



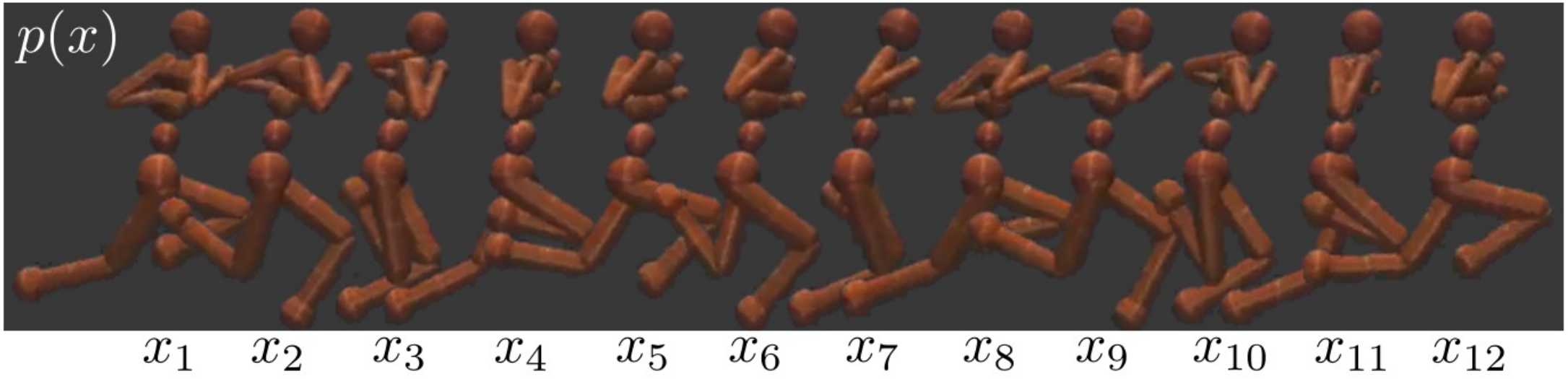
Repeatedly solving optimization problems

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

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

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: Shared structure between problems

Problems **share structure** and don't live in isolation

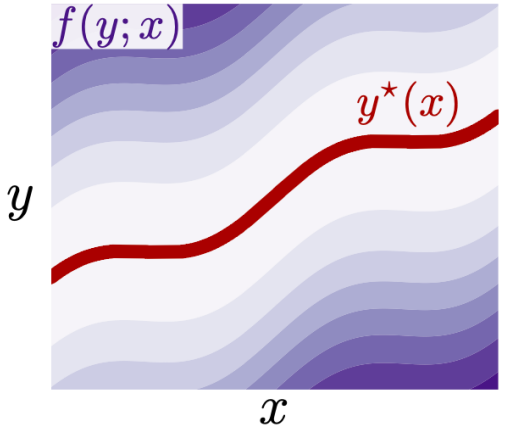
Solution: Amortized optimization

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

Amortized optimization foundations

This talk: Explore foundations and connect applications

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

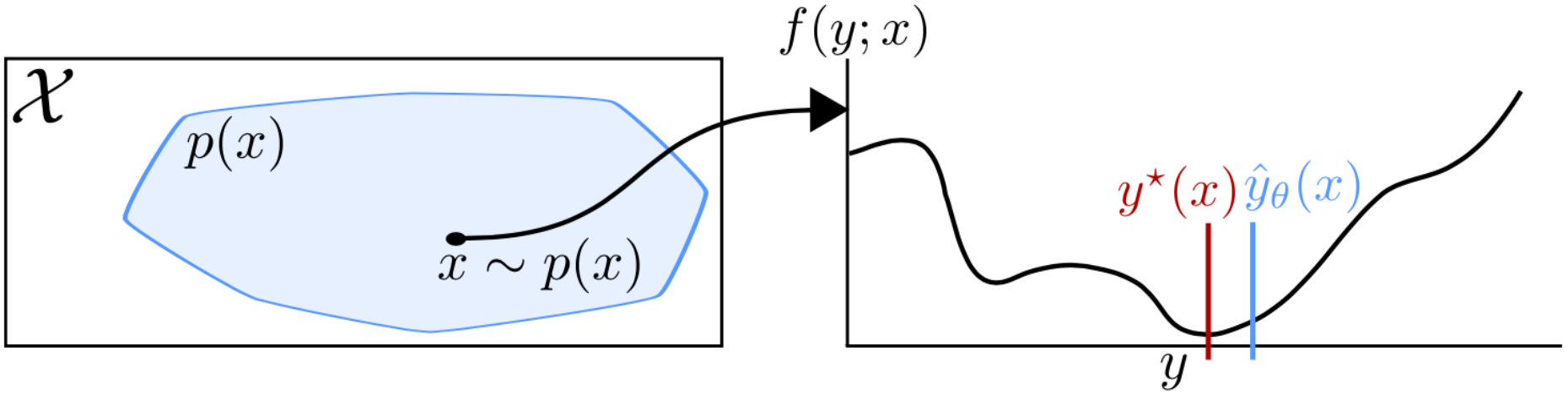


Amortization model $\hat{y}_\theta(x)$ tries to approximate $y^\star(x)$

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



This talk: Amortized optimization

Motivation and context

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

Applications
- VAEs, hyper-networks, MAML, reinforcement learning, meta optimal transport

Extensions and limitations

This talk: Amortized optimization

Motivation and context

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

Applications
- VAEs, hyper-networks, MAML, reinforcement learning, meta optimal transport

Extensions and limitations

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

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

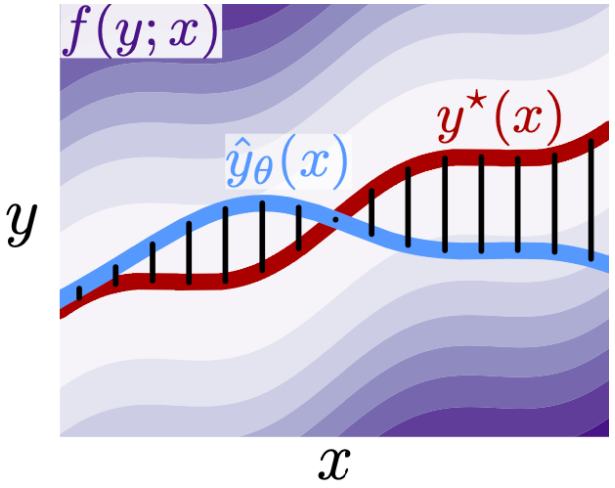
$$\hat{y}_\theta^0 \rightarrow \hat{y}_\theta^1 \rightarrow \cdots \rightarrow \hat{y}_\theta^K =: \hat{y}_\theta(x)$$

Learning paradigms for $\mathcal{L}$

Regression-based

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

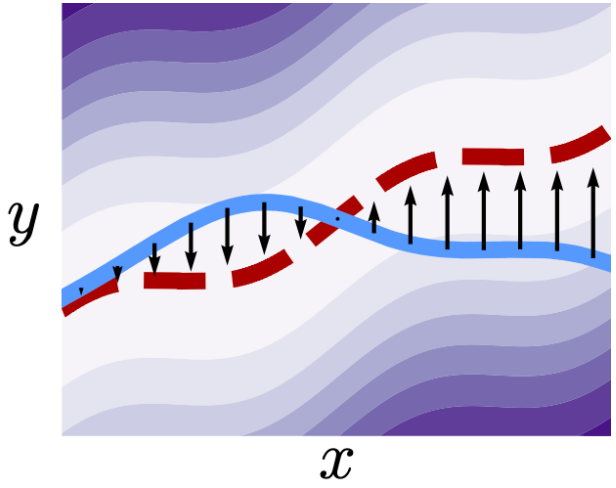
- $-$ Does not consider $f(y; x)$
- $+$ Uses global information with $y^\star(x)$
- $-$ Expensive to compute $y^\star(x)$
- $+$ Does not compute $\nabla_y f(y; x)$
- $-$ Hard to learn non-unique $y^\star(x)$



Objective-based:

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

- $+$ 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

Motivation and context

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

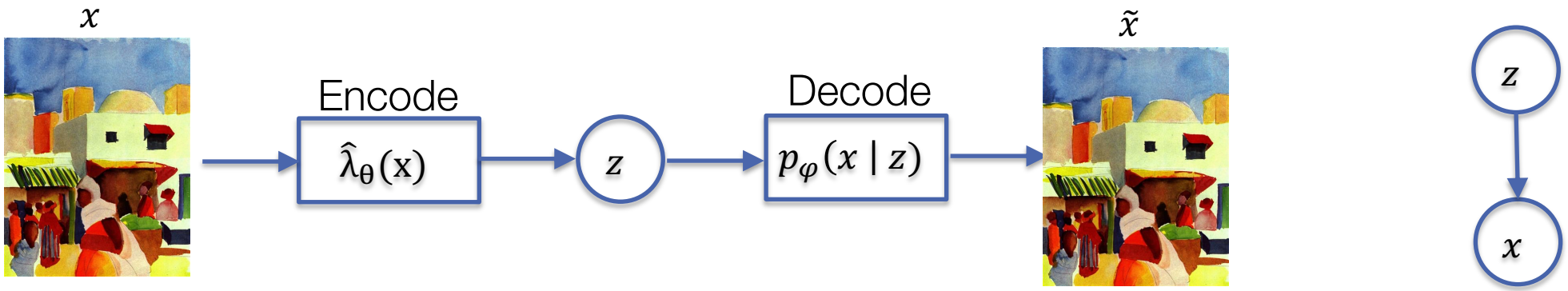
Applications
- VAEs, hyper-networks, MAML, reinforcement learning, meta optimal transport

Extensions and limitations

Warning: Quick overview of separate applications

Goal: Give a broad overview of **existing uses of amortized optimization**
Will try to focus on the amortization components, but assumes some context

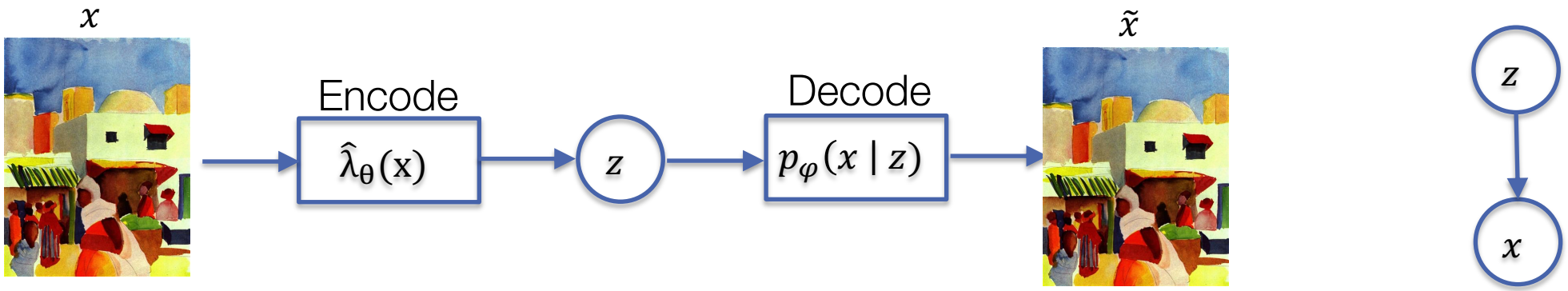
Amortized variational inference and auto-encoders



VAEs amortize the evidence lower bound (ELBO)

model: fully-amortized $\hat{\lambda}_\theta(x)$ as a neural network
loss: objective-based
objective: ELBO
learning: $\max_{\theta,\varphi} \mathbb{E}_{p(x)} \, \text{ELBO}(\hat{\lambda}_\theta(x); x, \varphi)$

Semi-amortized variational autoencoders



Incorporates knowledge of the ELBO into the encoder

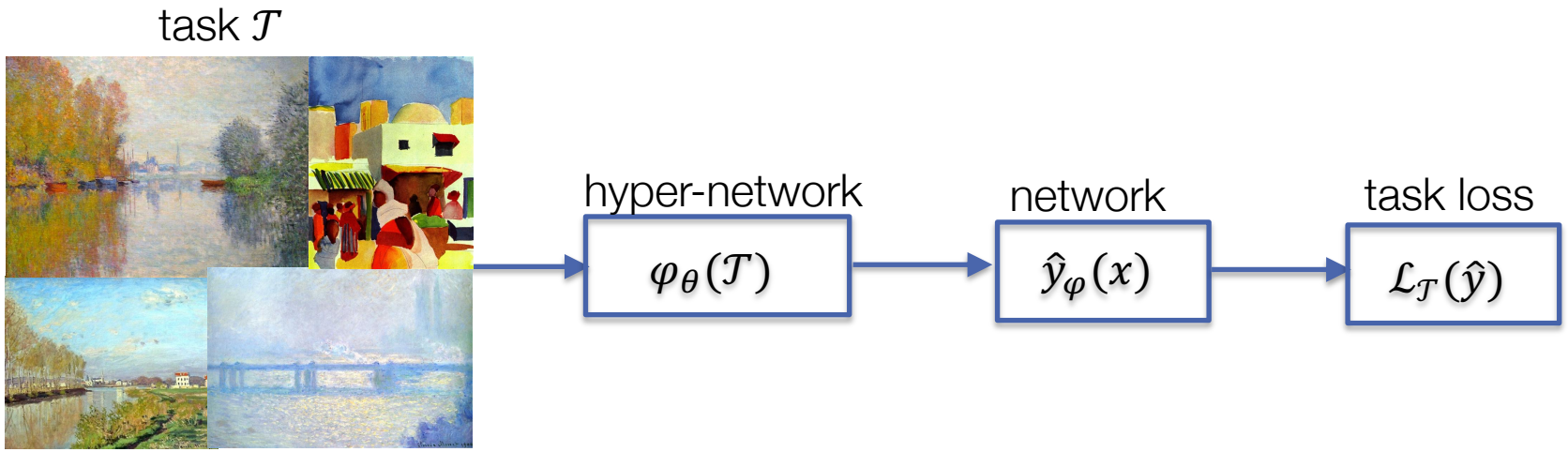
model: semi-amortized $\hat{\lambda}_\theta(x) \approx \underset{\lambda}{\operatorname{argmin}}\, \mathrm{ELBO}(\lambda; x, \varphi)$, approximated by gradient updates

loss: objective-based

objective: ELBO (same as before)

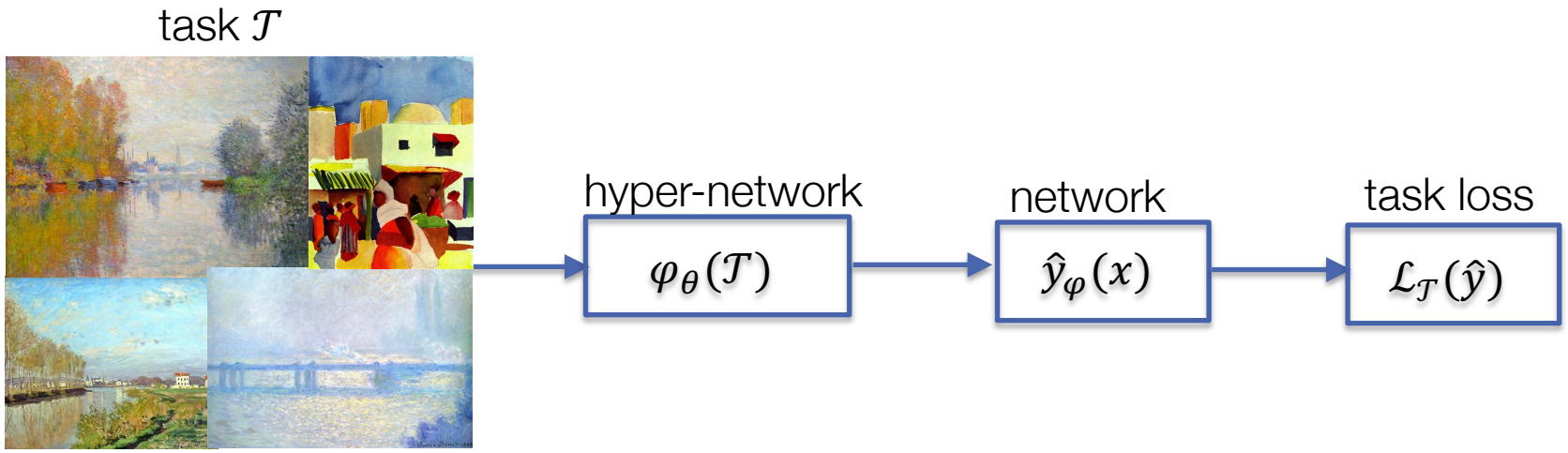
learning: $\max_{\theta,\varphi} \mathbb{E}_{p(x)}\, \mathrm{ELBO}(\hat{\lambda}_\theta(x); x, \varphi)$

Hyper-networks as amortized optimization



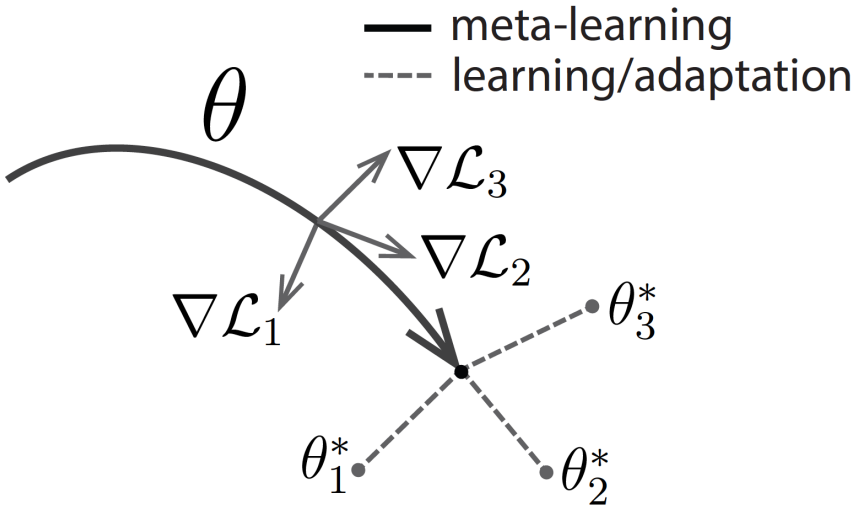
model: fully-amortized
loss: objective-based
objective: task loss
learning: $\underset{\theta}{\text{argmin}}\, \mathbb{E}_{p(\mathcal{T})}\, \mathcal{L}_\mathcal{T}(\hat{y})$

MAML as amortized optimization



Setup almost identical to hyper-networks, except for the model

model: semi-amortized
loss: objective-based
objective: task loss
learning: $\min_\theta \mathbb{E}_{p(\mathcal{T})} \mathcal{L}_\mathcal{T}(\hat{y}_\theta)$

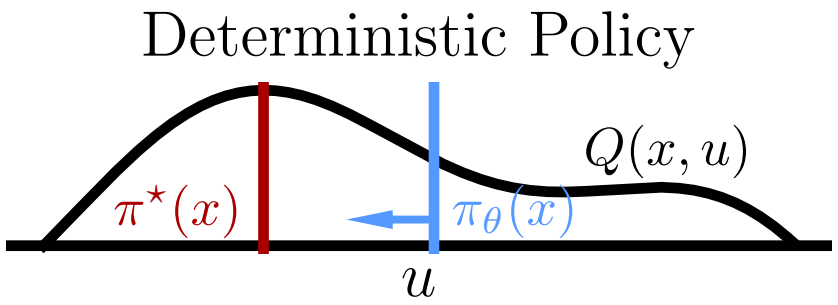


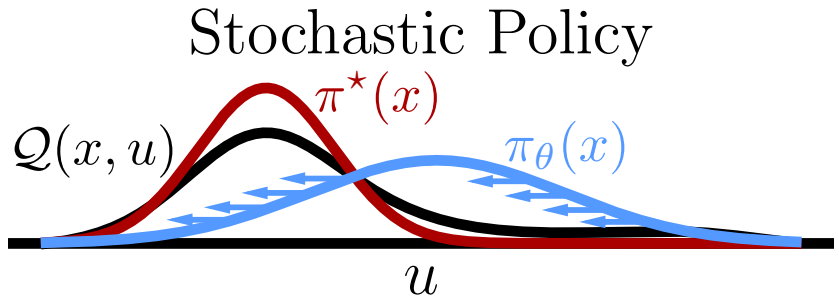
Amortized policy learning in reinforcement learning

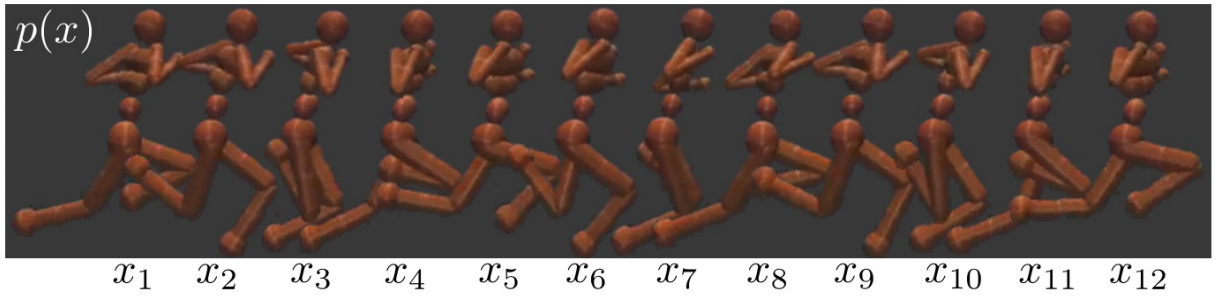
The **policy** $\pi_\theta(x)$ of almost every reinforcement learning method performs amortized optimization

setup: Controlling a continuous MDP
policy: $\pi_\theta(x)$
value (expected reward): $V(x) \coloneqq \mathbb{E}\, r(x_t, u_t)$
Q-value: $Q(x,u) \coloneqq \sum r(x,u) + \mathbb{E}_{x'} V^\pi(x')$

model: fully-amortized
loss: objective-based
objective: Q-value
learning: $\max_\theta \mathbb{E}_{p(x)}\, Q(x, \pi_\theta(x))$

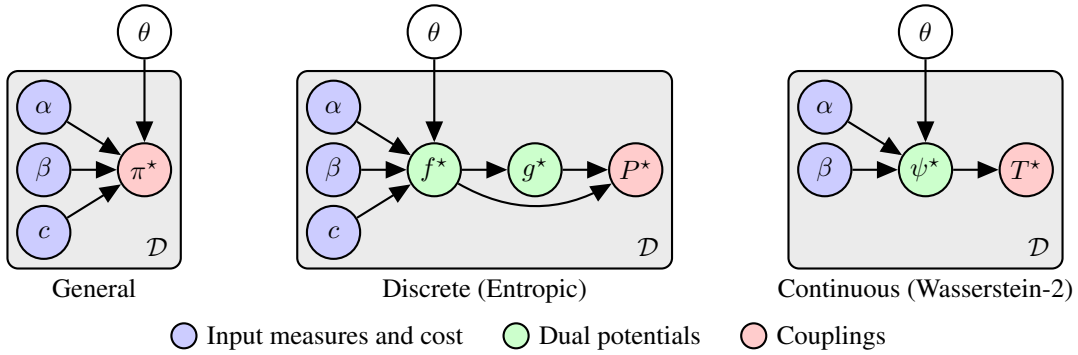


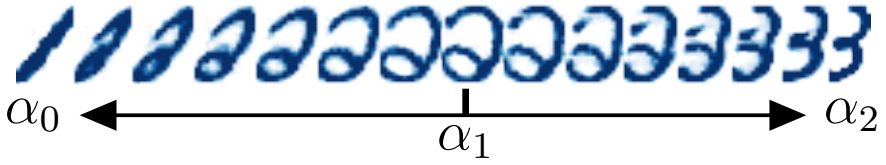


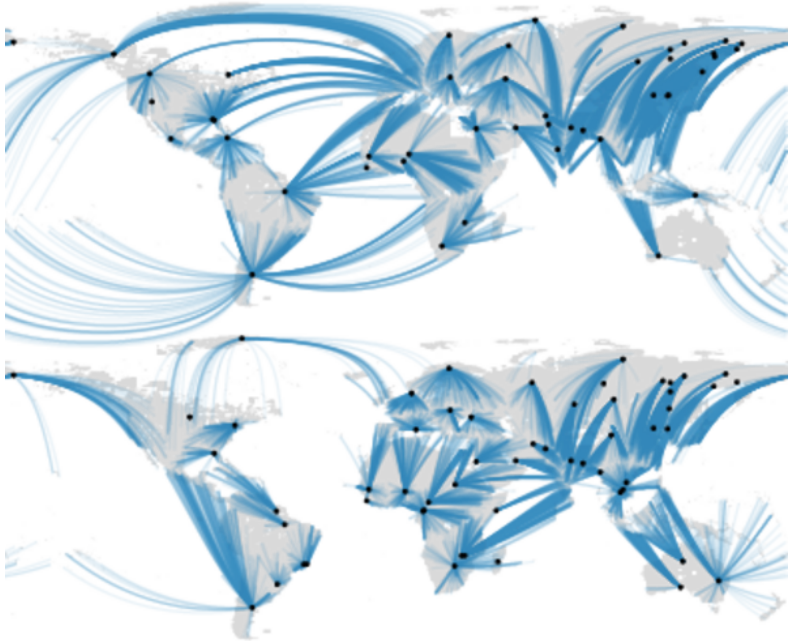


(Amortized) meta optimal transport

(Available on workplace)





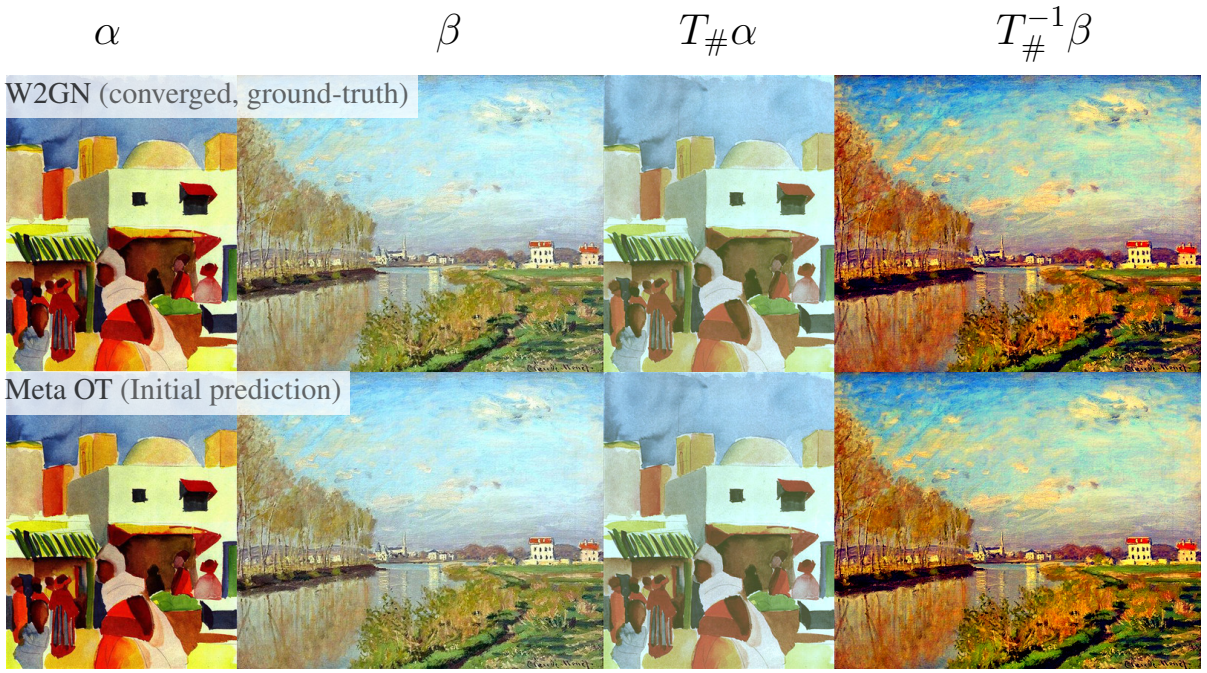


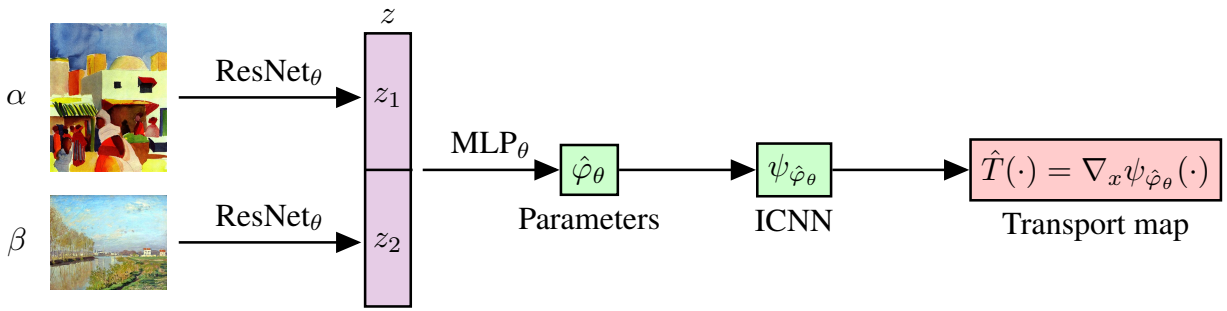
goal: Optimally transport mass between measures α to β

model: fully-amortized (hyper-network)
loss: objective-based
objective: transport cost
learning: $\max_{\theta} \mathbb{E}_{p(\alpha,\beta)} C(\hat{\pi}(\alpha,\beta))$

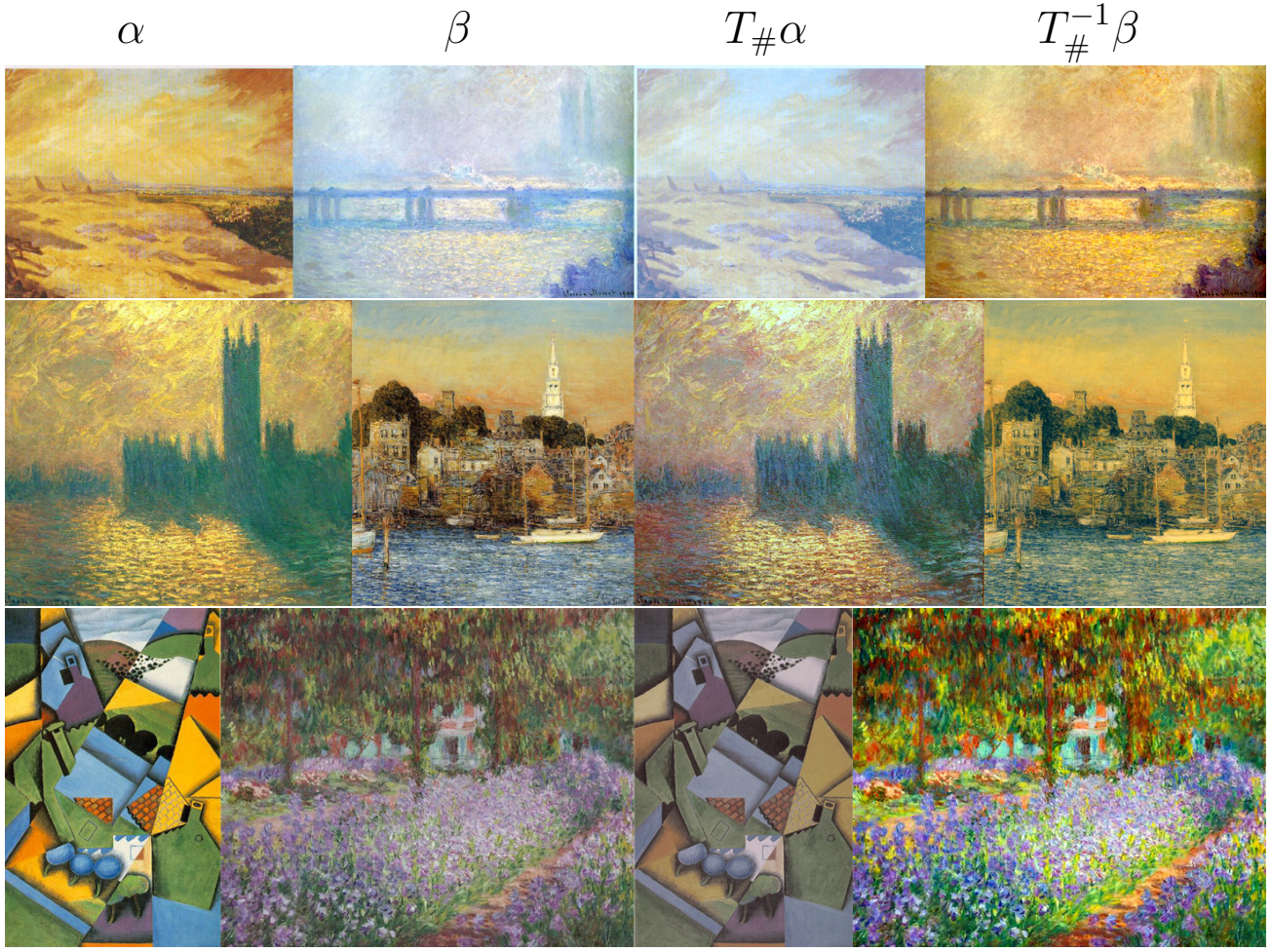
(Amortized) meta optimal transport

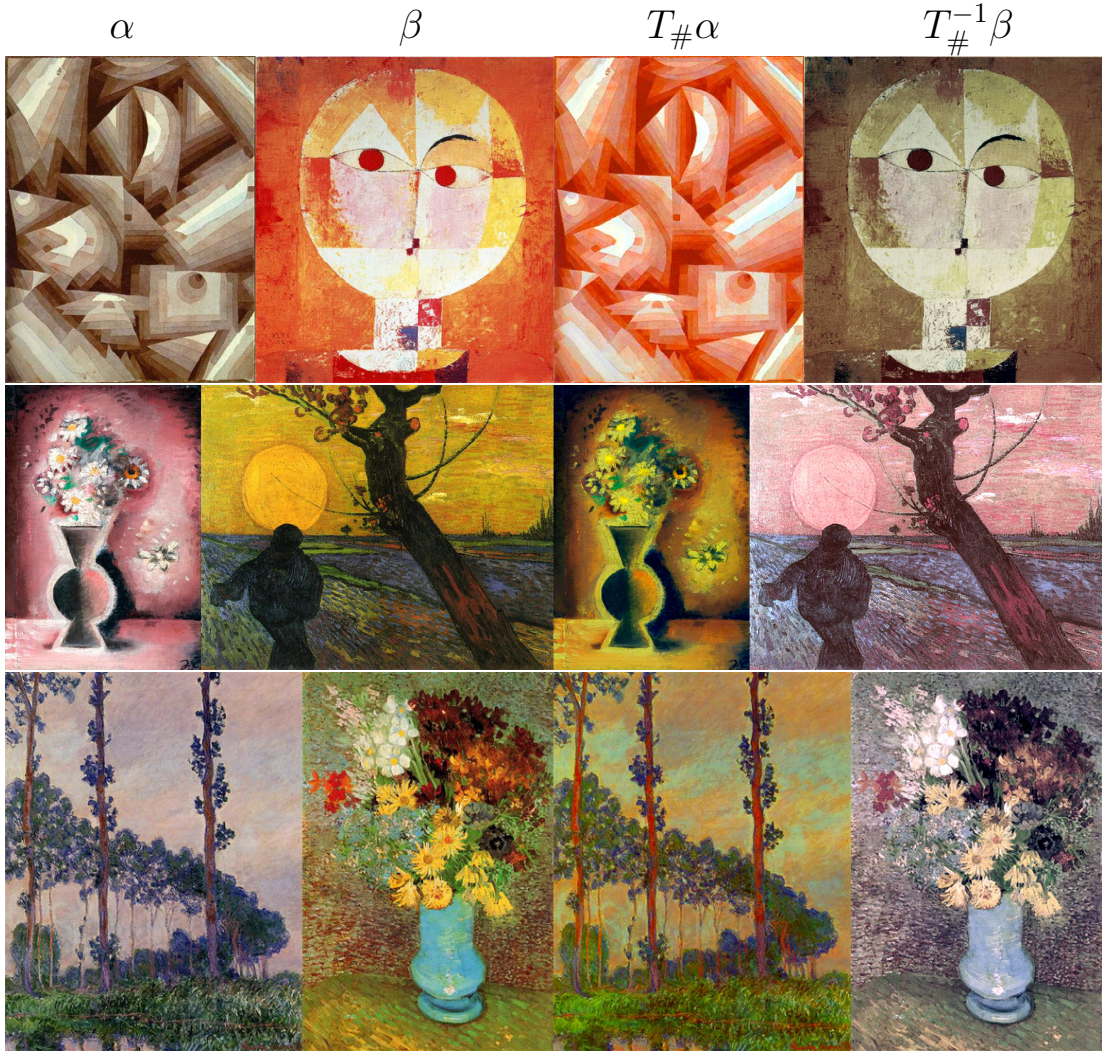
Predict couplings **10000x faster** than SOTA solvers that doesn't use learning





More meta color transfer predictions





This talk: Amortized optimization

Motivation and context

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

Applications
- VAEs, hyper-networks, MAML, reinforcement learning, meta optimal transport

Extensions and limitations

Limitations of amortized optimization

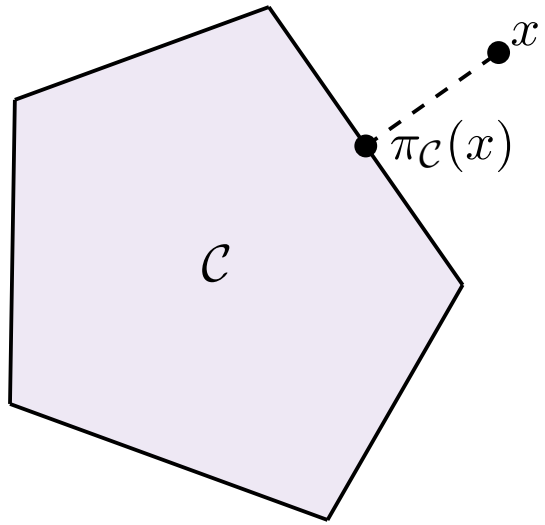
Gives extreme computational advantages in many settings
but may be detrimental and suboptimal in the wrong settings

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

Extensions

All settings here: **unconstrained continuous Euclidean optimization problems**
Possible to expand far beyond these:

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)



Tutorial on amortized optimization for learning to optimize over continuous spaces

Brandon Amos, Meta AI (FAIR) brandondamos bamos.github.io

Motivation and context

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

Applications
- VAEs, hyper-networks, MAML, reinforcement learning, meta optimal transport

Extensions and limitations

github.com/facebookresearch/amortized-optimization-tutorial