Meta Reinforcement Learning

CS 224R

Reminders

Homework 3 due **Friday**

Project milestone due **next Friday**

Announcements

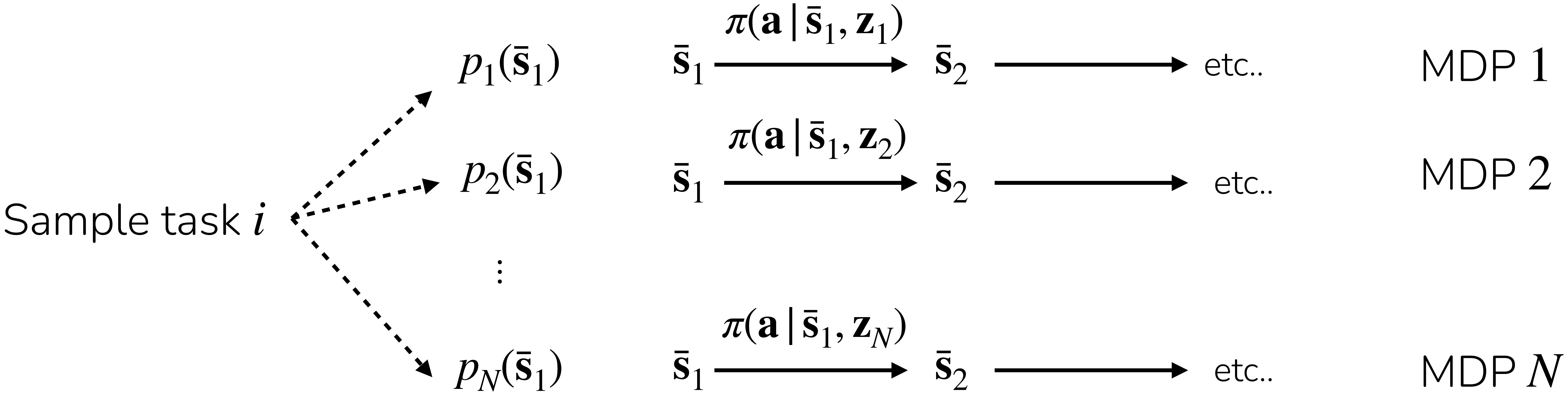
More evening office hours!

(Andy and Marcel moving to evening times)

Recap: Multi-task learning

Multi-task RL as single-task RL in a joint MDP

$$\mathbf{s} = (\bar{\mathbf{s}}, \mathbf{z}_i)$$



Sample task i

$p_1(\bar{\mathbf{s}}_1)$ $\quad \bar{\mathbf{s}}_1 \xrightarrow{\pi(\mathbf{a} \mid \bar{\mathbf{s}}_1, \mathbf{z}_1)} \bar{\mathbf{s}}_2 \longrightarrow$ etc.. MDP 1

$p_2(\bar{\mathbf{s}}_1)$ $\quad \bar{\mathbf{s}}_1 \xrightarrow{\pi(\mathbf{a} \mid \bar{\mathbf{s}}_1, \mathbf{z}_2)} \bar{\mathbf{s}}_2 \longrightarrow$ etc.. MDP 2

$\vdots$

$p_N(\bar{\mathbf{s}}_1)$ $\quad \bar{\mathbf{s}}_1 \xrightarrow{\pi(\mathbf{a} \mid \bar{\mathbf{s}}_1, \mathbf{z}_N)} \bar{\mathbf{s}}_2 \longrightarrow$ etc.. MDP N

This is going to be a bit different for meta-RL!

Recap: Goal-conditioned learning

Special case of multi-task RL, where $\mathbf{z}_i = \mathbf{s}_g$

Each task aims to reach some goal state $\mathbf{s}_g$

$r(\mathbf{s}, \mathbf{a}, \mathbf{s}_g) = \delta(\mathbf{s} = \mathbf{s}_g)$ for discrete states

$r(\mathbf{s}, \mathbf{a}, \mathbf{s}_g) = \delta(||\mathbf{s} - \mathbf{s}_g|| \leq \epsilon)$ for continuous states

+ No need to define reward (self-supervised!)

+ Many tasks can be formed as goal reaching tasks

- Can be pretty hard to train

Plan for Today

1. Meta-RL problem statement
2. Black-box meta-RL methods

Next time: Exploration + Learning to explore.

part of HW4

Lecture goals:

- Understand the meta-RL problem statement & set-up
- Understand the basics & challenges of black-box meta RL algorithms

What is the problem?

Say you wanted to learn to make coffee with a new espresso machine.

Training a robot to do this with PPO
would take **millions of attempts**.

With some instruction or prior coffee experience,
a person could learn in **minutes**.

People aren't starting from scratch!

Experience with similar tasks, either espresso-making on other machines or general motor control.

Can also learn to solve a challenging math problem more quickly using past problem solving experience!

Can RL algorithms leverage experience from *previous* tasks when learning a ***new*** task?

Framing transfer learning problems

1. Forward transfer: learn policies that transfer effectively
    a) Train on source task, then fine-tune on target task
2. Multi-task transfer: train on many tasks, transfer to a new task
    a) Task prompt/descriptor $\mathbf{z}_i$ needs to capture task structure for zero-shot transfer
3. Meta-learning: learn to *learn* on many tasks
    a) Accounts for the fact that we'll be adapting to a new task during training!
    b) The "task descriptor" is a few data examples provided in-context

For (1): Source and target tasks must be similar

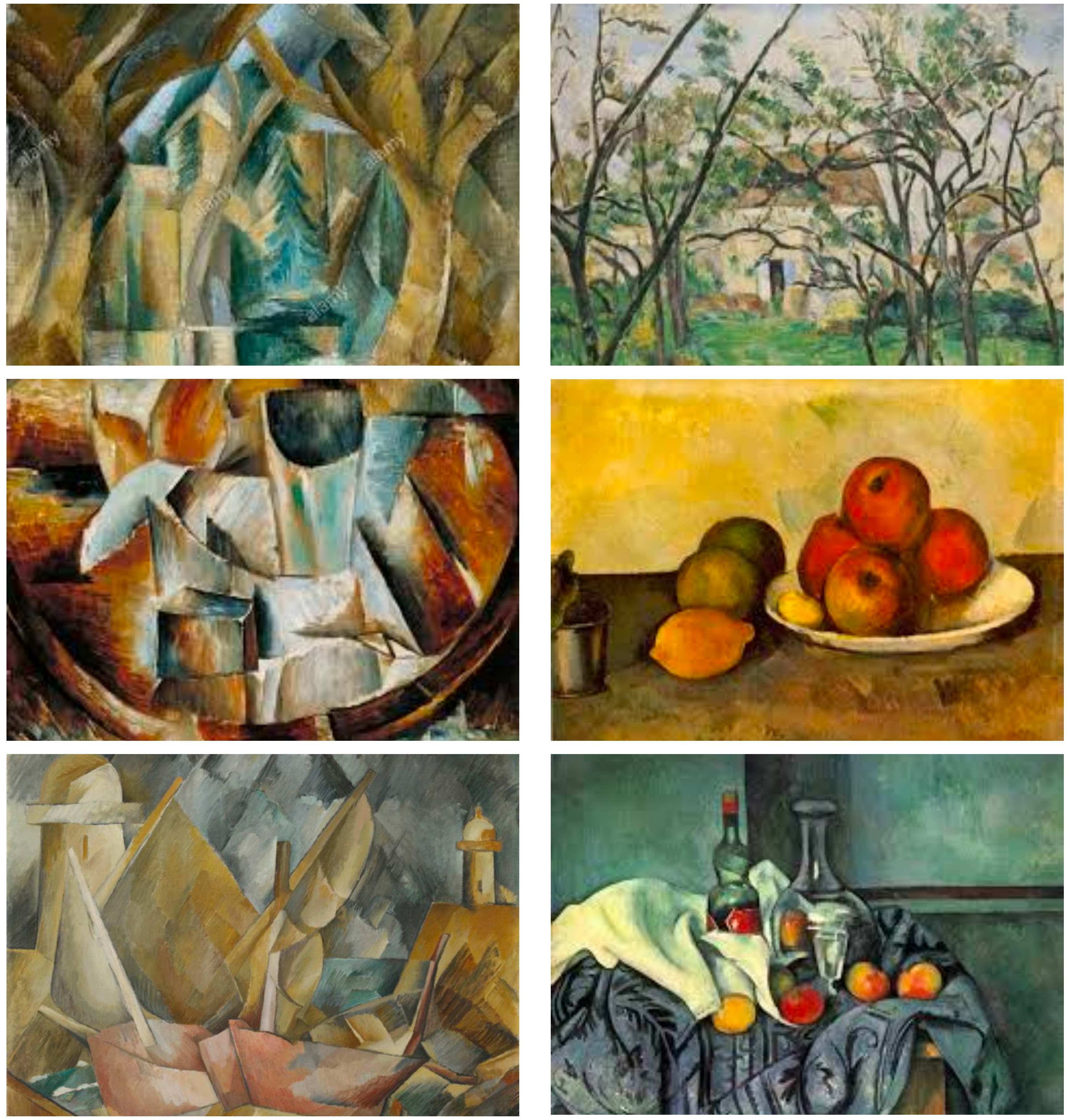
For (2) and (3): Target task must be similar to the *distribution* of training tasks

Slide adapted from Sergey Levine

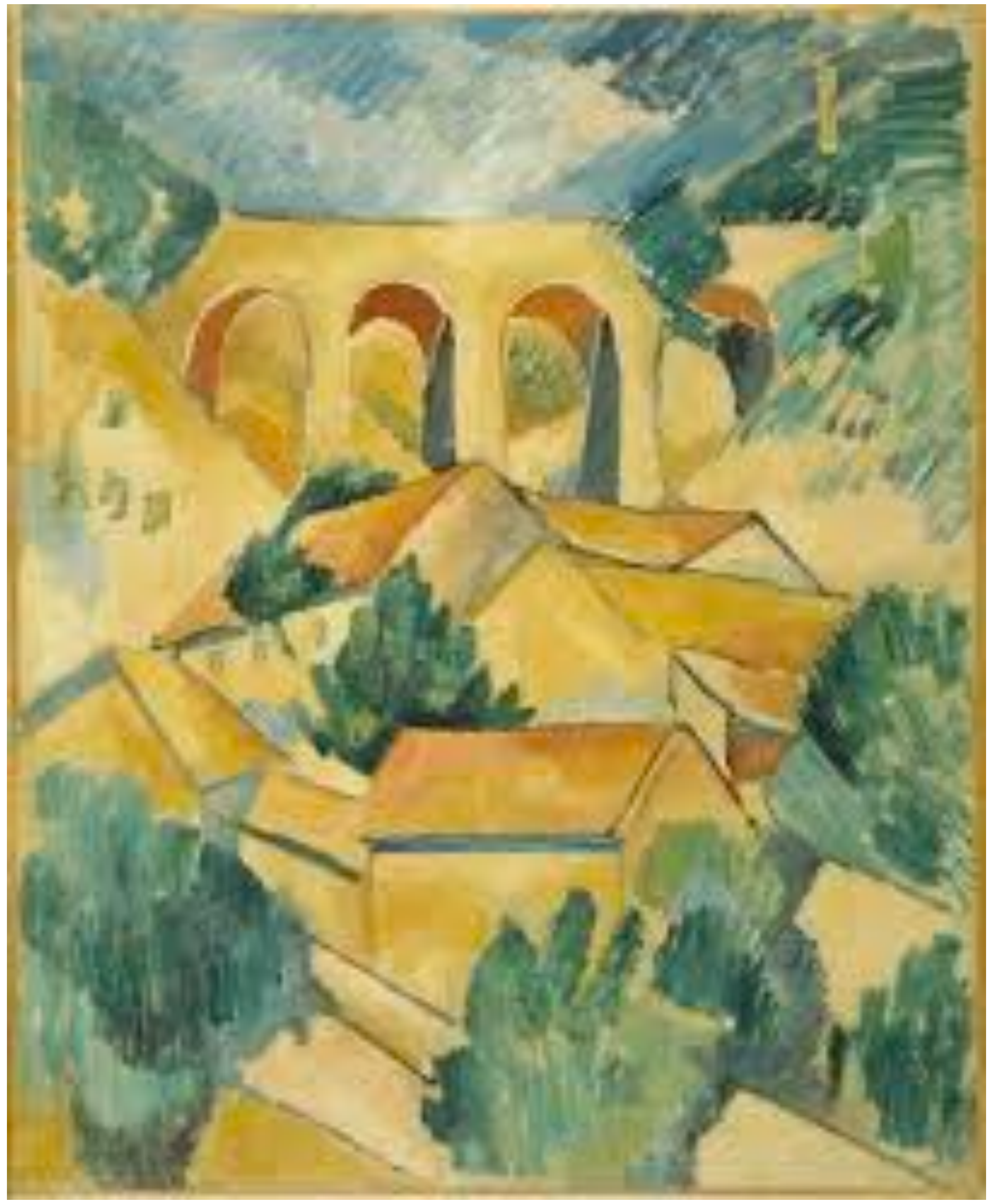
What does few-shot learning look like outside of RL?

training data

Braque　　Cezanne



test datapoint



By Braque or Cezanne?

What does few-shot learning look like outside of RL?

Few-shot image classification

Braque

Cezanne

D_{train}

x

$\hat{y}$ = Braque

In-context learning in LLMs

Poor English input: I eated the purple berries.
Good English output: I ate the purple berries.
Poor English input: Thank you for picking me as your designer. I'd appreciate it.
Good English output: Thank you for choosing me as your designer. I appreciate it.
Poor English input: The mentioned changes have done. or I did the alteration that you requested. or I changed things you wanted and did the modifications.
Good English output: The requested changes have been made. or I made the alteration that you requested. or I changed things you wanted and made the modifications.

D_{train}

Poor English input: Please provide me with a short brief of the design you're looking for and that'd be nice if you could share some examples or project you did before.

x

Good English output: Please provide me with a brief description of the design you're looking for and that would be nice if you could share some examples or projects you have done before.

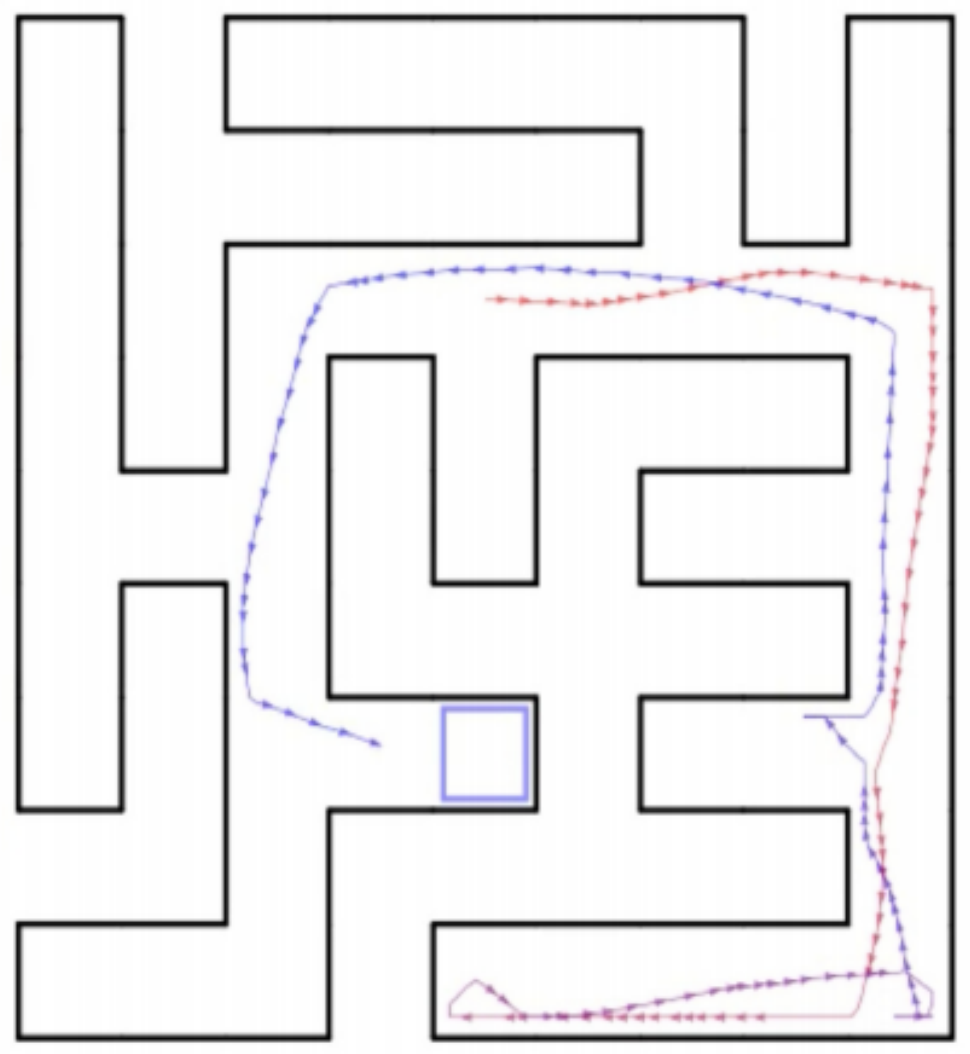
$\hat{y}$

Train a model $\hat{y} = f(x, D_{\text{train}})$ to be able to make predictions using on a few examples

Meta-RL Example: Maze Navigation

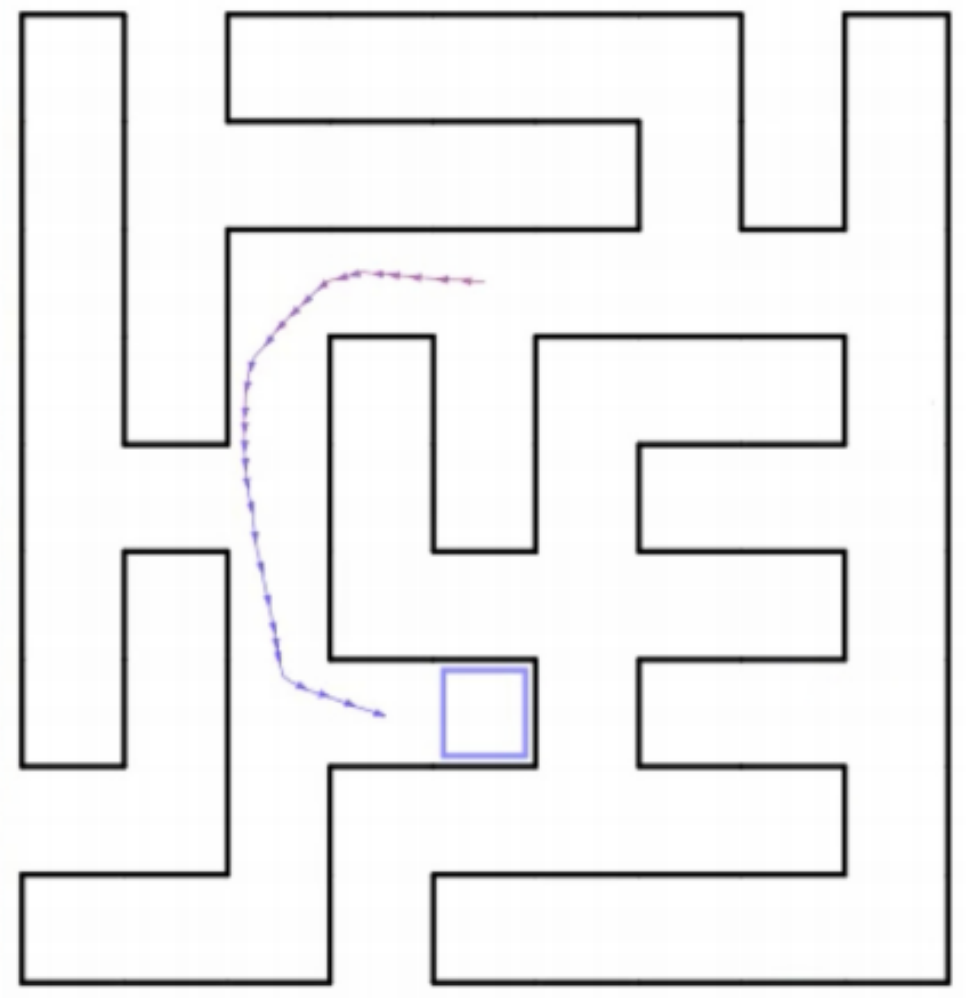
Goal:

Collect small amount of experience in new MDP



Collect $\mathscr{D}_{\text{tr}} \sim \pi^{\text{exp}}$

Learn policy that solves that MDP



$\mathscr{D}_{\text{tr}} \rightarrow \pi^{\text{task}}$

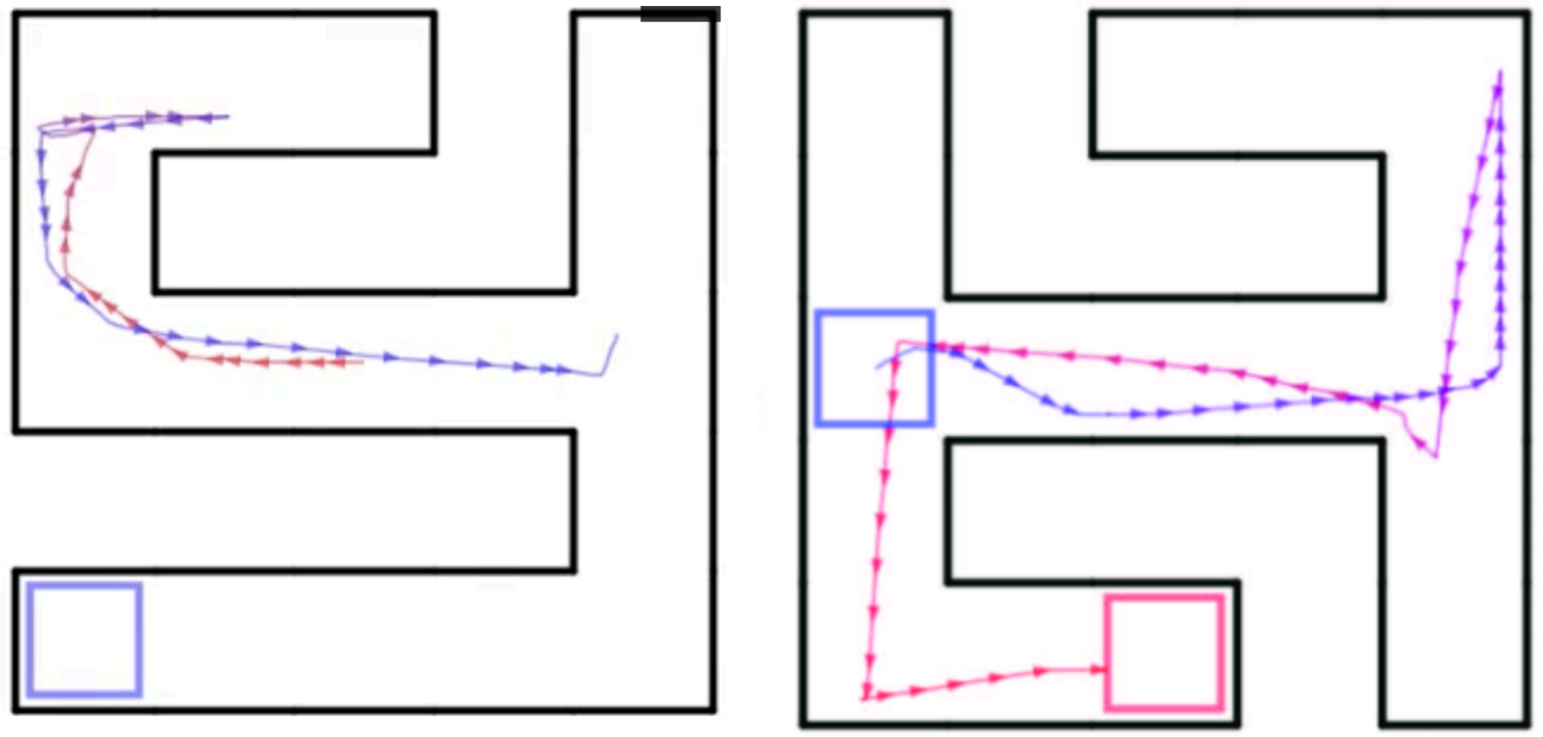
Exploration-exploitation trade-off is a unique part of meta-RL!

diagram adapted from Duan et al. ‘17

Meta-RL Example: Maze Navigation

Meta-Train Time:

Learn how to efficiently explore & solve many MDPs:

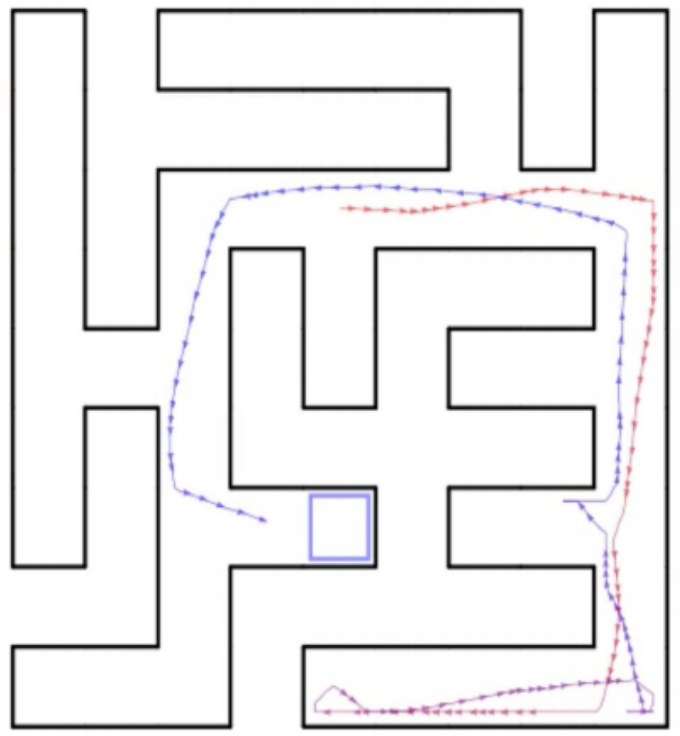


. . . meta-training tasks

Meta-train $\pi^{\text{exp}}, \pi^{\text{task}}$

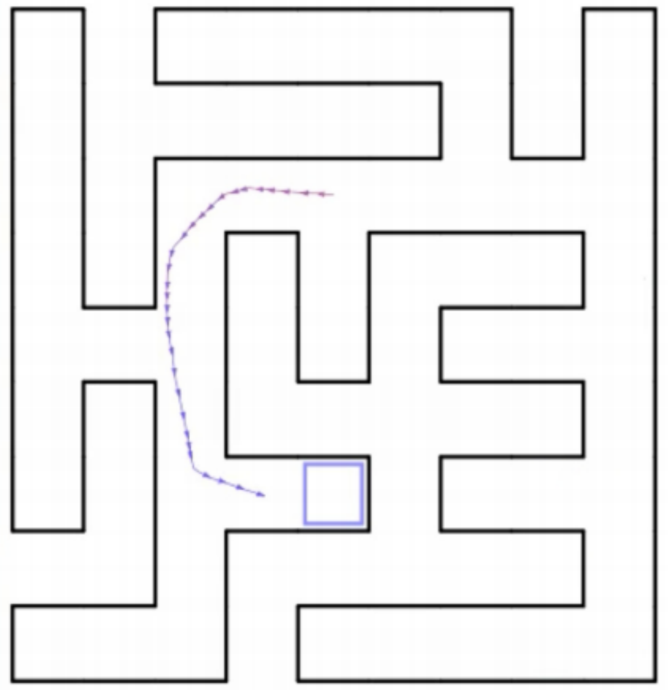
Meta-Test Time:

Collect small amount of experience in new MDP



Collect $\mathscr{D}_{\text{tr}} \sim \pi^{\text{exp}}$

Learn policy that solves that MDP



$\mathscr{D}_{\text{tr}} \rightarrow \pi^{\text{task}}$

Key assumption: Meta-testing MDPs come from same task distribution as meta-training MDPs.

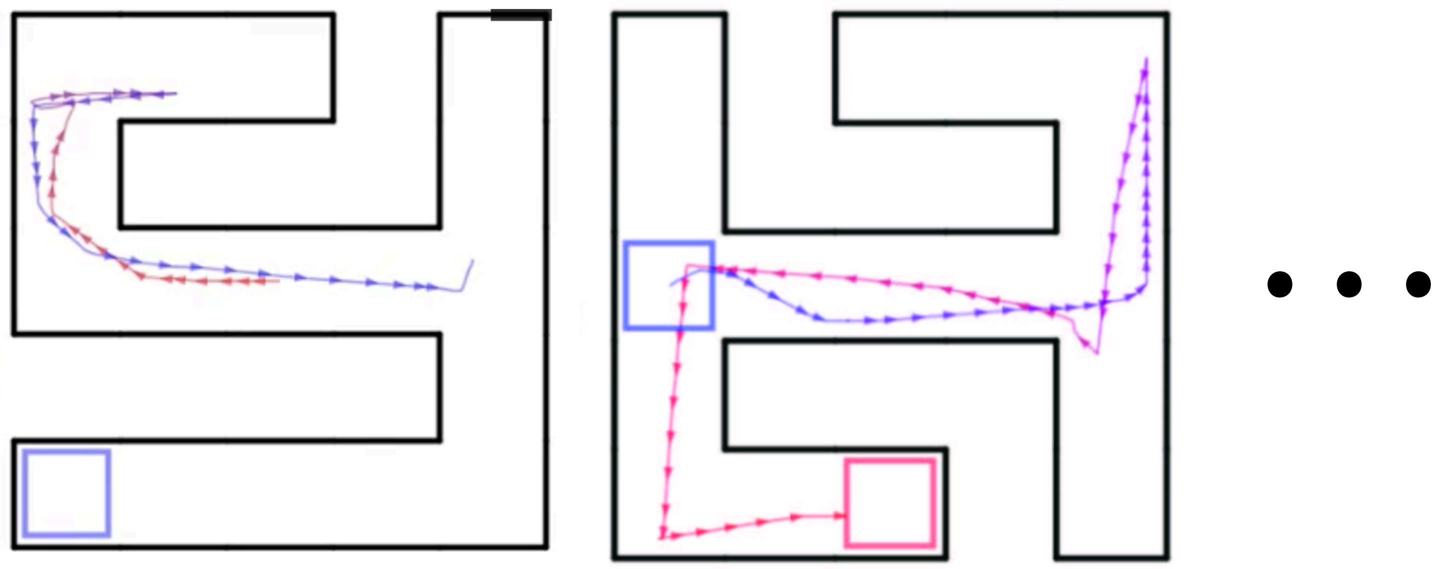
(so that we can expect generalization)

diagram adapted from Duan et al. '17

Meta-reinforcement learning task distributions

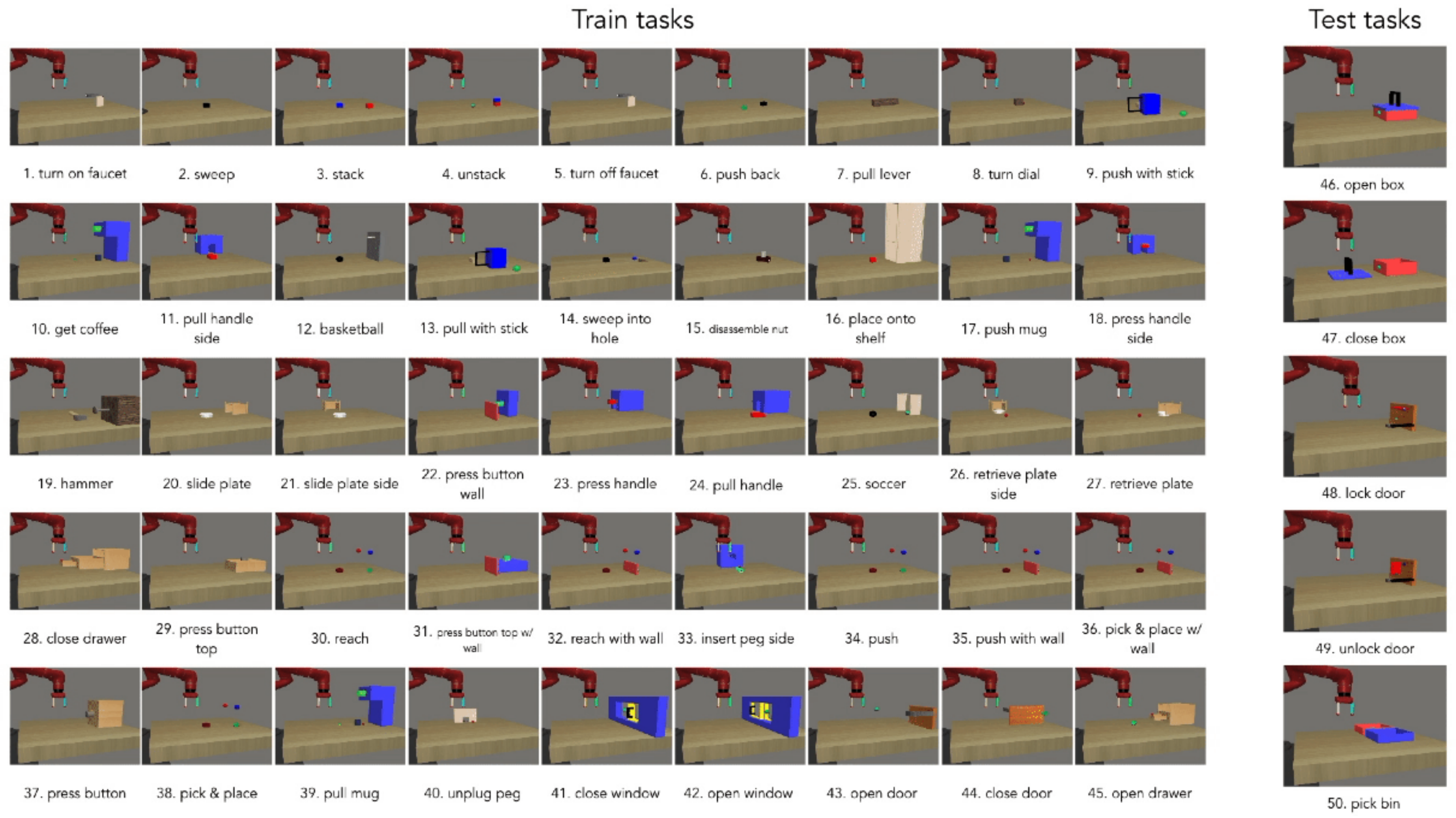
Examples of meta-RL tasks

Navigation through different mazes



Locomotion on different terrains, slopes

Object manipulation with different objects, goals



Dialog with different users w/ different preferences

More Formal Problem Settings

Multi-Task Learning

Solve multiple tasks $\mathcal{T}_1, \cdots, \mathcal{T}_T$ at once.

$$\min_{\theta} \sum_{i=1}^{T} \mathcal{L}_i(\theta, \mathcal{D}_i)$$

Transfer Learning

Solve target task $\mathcal{T}_b$ after solving source task $\mathcal{T}_a$ by *transferring* knowledge learned from $\mathcal{T}_a$

The Meta-Learning Problem

Given data from $\mathcal{T}_1, \ldots, \mathcal{T}_n$, quickly solve new task $\mathcal{T}_{\text{test}}$

A reinforcement learning task:

$$\mathcal{T}_i \triangleq \{\underset{\substack{\uparrow \\ \text{state} \\ \text{space}}}{\mathcal{S}_i}, \overset{\substack{\text{action space} \\ \downarrow}}{\mathcal{A}_i}, \underset{\substack{\uparrow \\ \text{initial state} \\ \text{distribution}}}{p_i(\mathbf{s}_1)}, \overset{\substack{\text{dynamics} \\ \downarrow}}{p_i(\mathbf{s}' \mid \mathbf{s}, \mathbf{a})}, \underset{\substack{\uparrow \\ \text{reward}}}{r_i(\mathbf{s}, \mathbf{a})}\}$$

Meta-reinforcement learning = meta-learning with RL tasks

The Meta Reinforcement Learning Problem

Episodic Variant

Inputs: $\underbrace{\mathcal{D}_{\text{train}}}_{\text{k rollouts from } \pi^{\text{exp}}}$ $\mathbf{s}_t$ Outputs: $\mathbf{a}_t$

$$\mathbf{a} \sim \pi^{\text{task}}(\,\cdot \,|\, \mathbf{s}, D_{\text{train}})$$

can view as a task identifier $\mathbf{z}_i$

Online Variant

Inputs: $\underbrace{\mathcal{D}_{\text{train}}}_{\text{1...k timesteps from } \pi^{\text{exp}}}$ $\mathbf{s}_t$ Outputs: $\mathbf{a}_t$

$$\mathbf{a} \sim \pi^{\text{task}}(\,\cdot \,|\, \mathbf{s}, D_{\text{train}})$$

Note: exploration policy π^{exp} and adaptation policy π^{task} could share parameters.

Plan for Today

1. Meta-RL problem statement
2. **Black-box meta-RL methods**

Next time: Exploration + Learning to explore.

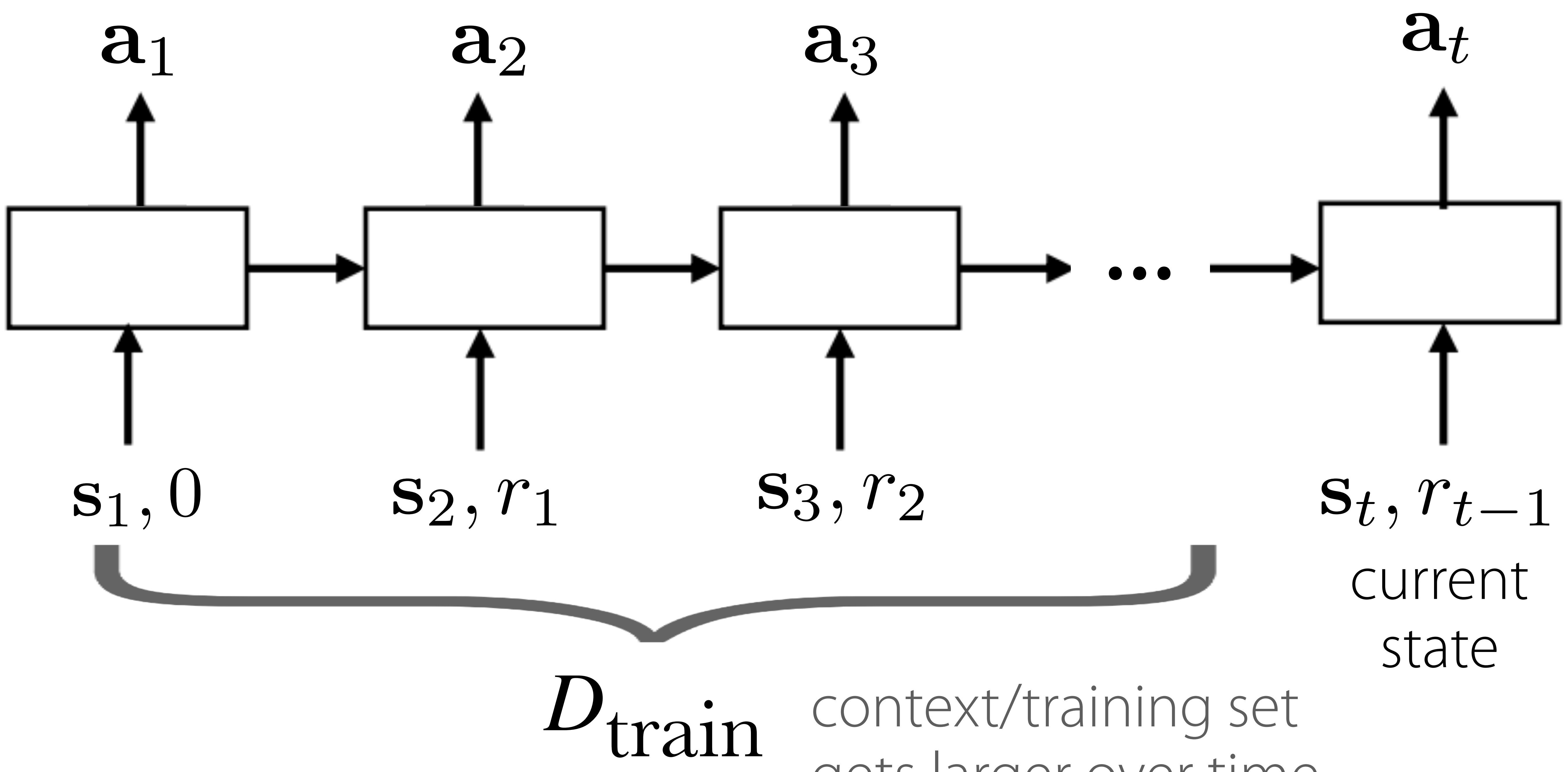
part of HW4

Black-Box Meta-RL: Overview

Black-box neural net

(Transformer, NN with memory)

$$\mathbf{a} \sim \pi^{\text{task}}(\,\cdot\,|\,\mathbf{s}, D_{\text{train}})$$



context/training set gets larger over time

Question: How is this different from simply doing RL with a recurrent policy?

Reward is passed as input
(& trained across multiple MDPs)

Hidden state maintained
across episodes within a task!

Black-Box Meta-RL: Algorithm

$\mathbf{a}_1$
$\mathbf{a}_2$
$\mathbf{a}_1$
$\mathbf{a}_2$
$\mathbf{a}_1$
...
...
$\mathbf{s}_1, 0$
$\mathbf{s}_2, r_1$
$\mathbf{s}_T, r_{T-1}$
$\mathbf{s}_1, 0$
$\mathbf{s}_2, r_1$
$\mathbf{s}_T, r_{T-1}$
$\mathbf{s}_1, 0$
Episode 1
Episode 2

1. Sample task $\mathcal{T}_i$
2. Roll-out policy $\pi(a \mid s, \mathcal{D}_i^{\text{tr}})$ for N episodes
3. Store sequence in replay buffer for task $\mathcal{T}_i$.
4. Update policy to maximize discounted return for all tasks.

(under dynamics $p_i(s' \mid s, a)$ and reward $r_i(s, a)$)

Black-Box Meta-RL: Algorithm

Meta-Training

1. Sample task $\mathcal{T}_i$
2. Roll-out policy $\pi(a \mid s, \mathcal{D}_i^{\mathrm{tr}})$ for N episodes
3. Store sequence in replay buffer for task $\mathcal{T}_i$.
4. Update policy to maximize discounted return for all tasks.

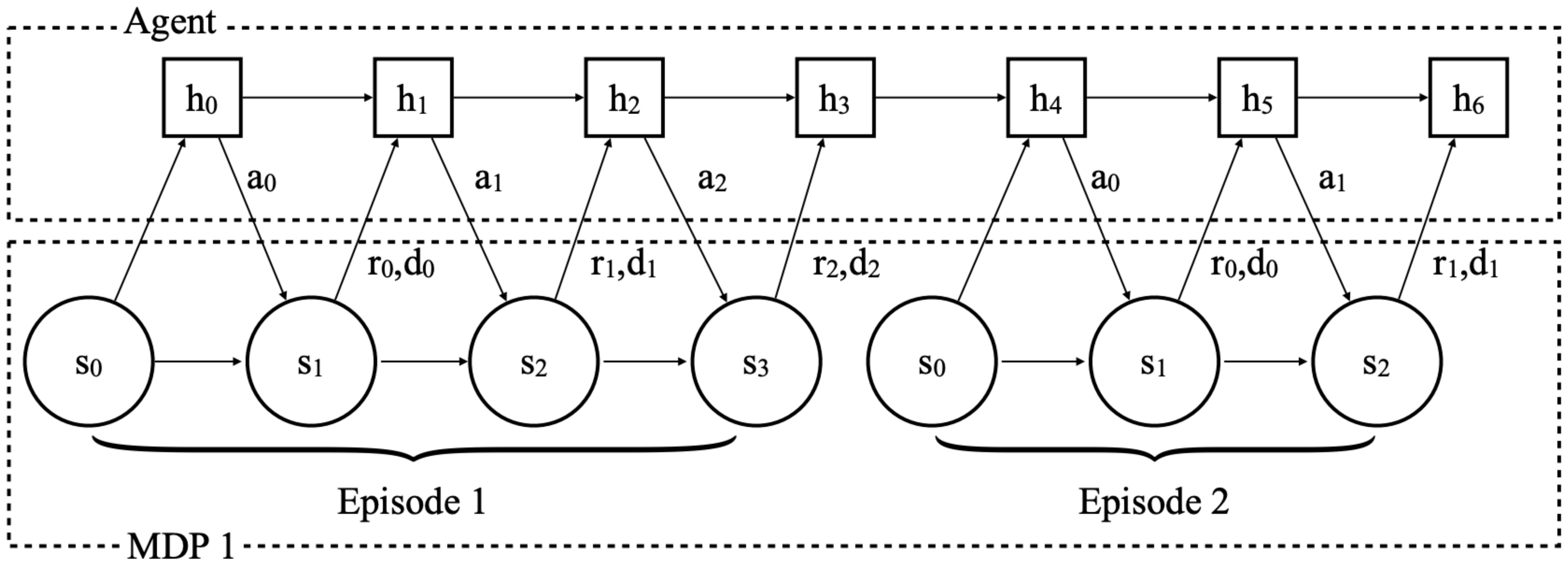
(under dynamics $p_i(s' \mid s, a)$ and
reward $r_i(s, a)$)

Meta-Test Time

1. Sample *new* task $\mathcal{T}_j$
2. Roll-out policy $\pi(a \mid s, \mathcal{D}_j^{\mathrm{tr}})$ for up to N episodes

Black-Box Meta-RL: Architectures & Optimizers

RNN architecture **TRPO/A3C (similar to PPO)**



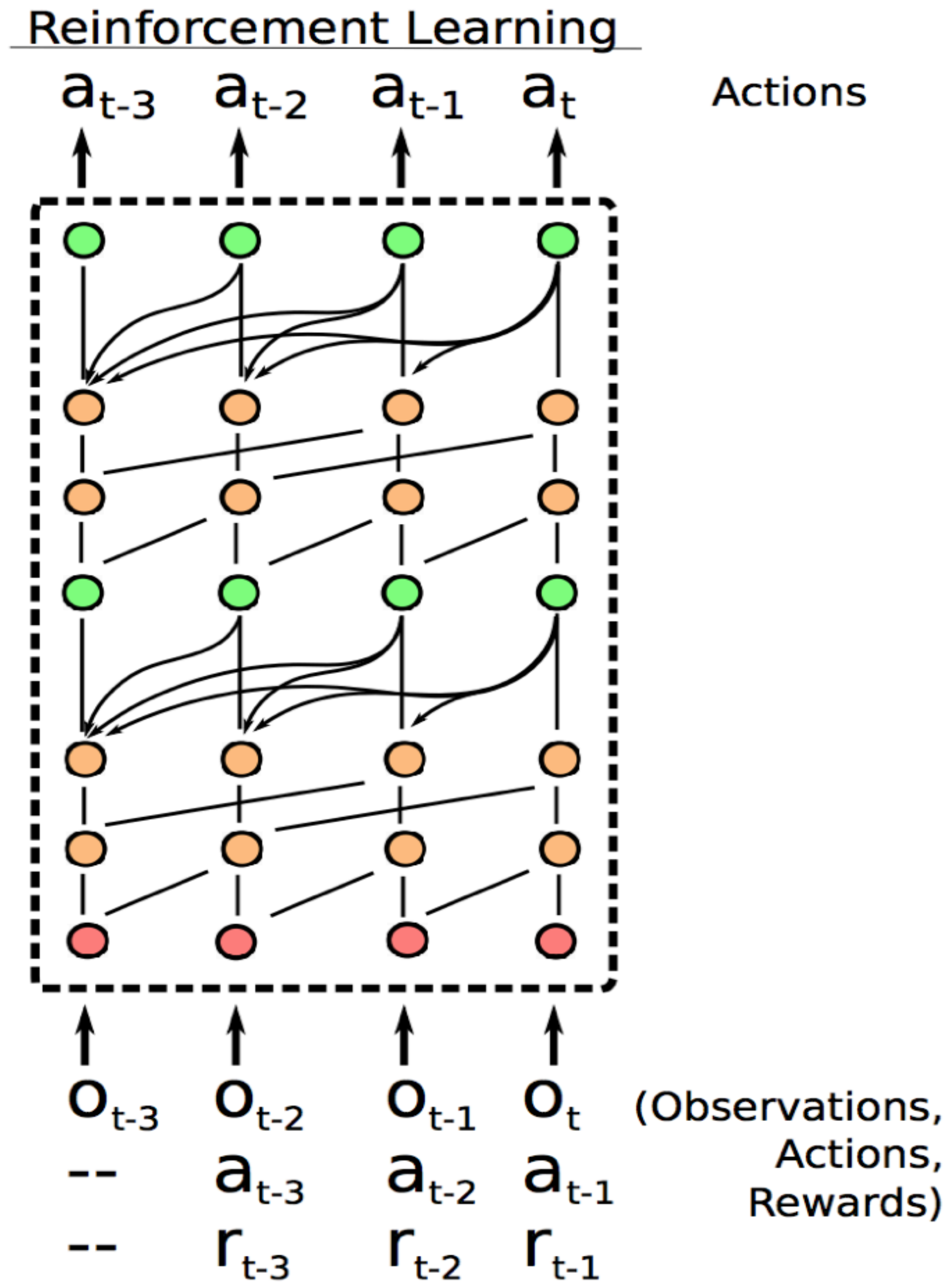
Duan, Schulman, Chen, Bartlett, Sutskever, Abbeel. *RL^2: Fast Reinforcement Learning via Slow Reinforcement Learning*. 2017

Wang, Kurth-Nelson, Tirumala, Soyer, Leibo, Munos, Blundell, Kumaran, Botvinick. *Learning to Reinforcement Learn*. CogSci 2017

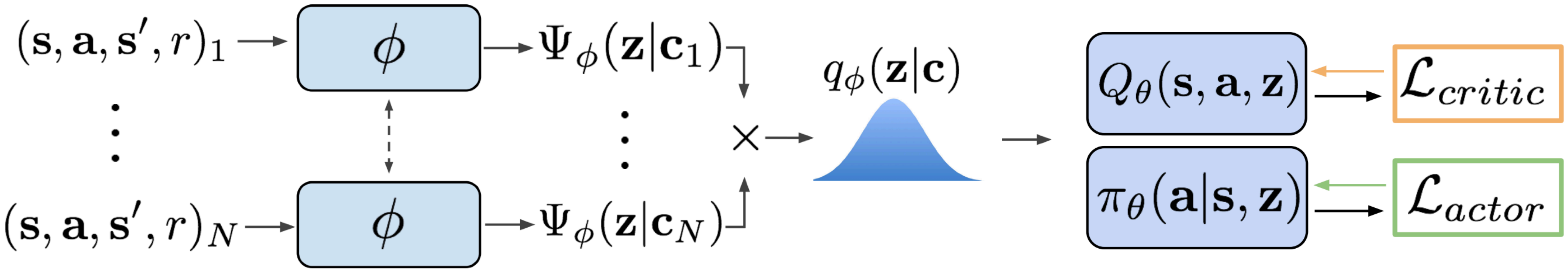
Attention + 1D conv

TRPO (similar to PPO)

Mishra, Rohaninejad, Chen, Abbeel. *A Simple Neural Attentive Meta-Learner*. ICLR 2018



Feedforward + average **SAC (off-policy with replay buffer)**


$(\mathbf{s},\mathbf{a},\mathbf{s}',r)_1$
$(\mathbf{s},\mathbf{a},\mathbf{s}',r)_N$
ϕ
ϕ
$\Psi_\phi(\mathbf{z}|\mathbf{c}_1)$
$\Psi_\phi(\mathbf{z}|\mathbf{c}_N)$
$\times$
$q_\phi(\mathbf{z}|\mathbf{c})$
$Q_\theta(\mathbf{s},\mathbf{a},\mathbf{z})$
$\pi_\theta(\mathbf{a}|\mathbf{s},\mathbf{z})$
$\mathcal{L}_{critic}$
$\mathcal{L}_{actor}$

Rakelly, Zhou, Quillen, Finn, Levine. *Efficient Off-Policy Meta-Reinforcement Learning via Probabilistic Context Variables*. ICML 2019.

Meta-RL Example #1

From: Mishra, Rohaninejad, Chen, Abbeel. ***A Simple Neural Attentive Meta-Learner***. ICLR 2018

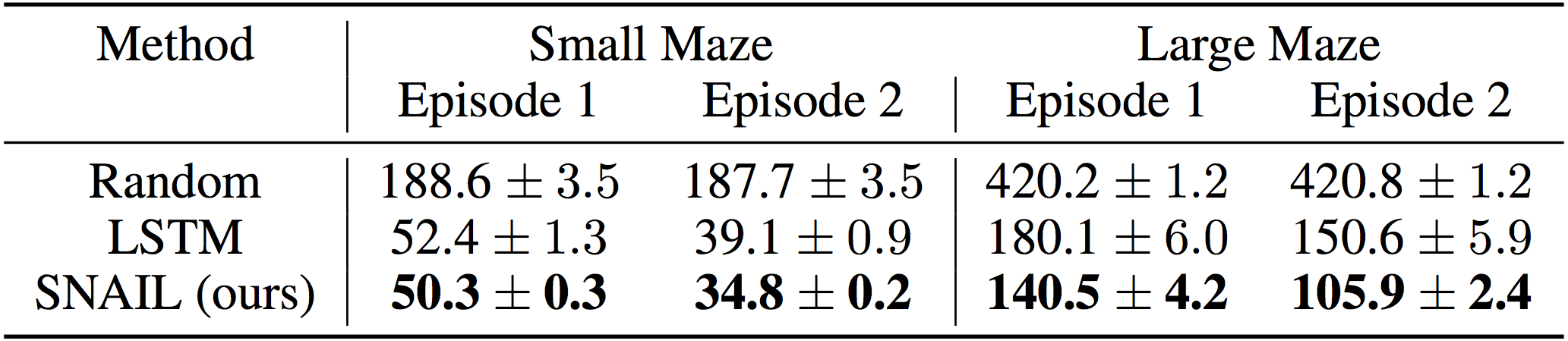
Experiment: Learning to visually navigate a maze
- train on 1000 small mazes
- test on held-out small mazes and large mazes

Meta-RL Example #1

From: Mishra, Rohaninejad, Chen, Abbeel. ***A Simple Neural Attentive Meta-Learner***. ICLR 2018

Experiment: Learning to visually navigate a maze

- train on 1000 small mazes
- test on held-out small mazes and large mazes



Method	Small Maze		Large Maze	
	Episode 1	Episode 2	Episode 1	Episode 2
Random	188.6 ± 3.5	187.7 ± 3.5	420.2 ± 1.2	420.8 ± 1.2
LSTM	52.4 ± 1.3	39.1 ± 0.9	180.1 ± 6.0	150.6 ± 5.9
SNAIL (ours)	**50.3 ± 0.3**	**34.8 ± 0.2**	**140.5 ± 4.2**	**105.9 ± 2.4**

Table 5: Average time to find the goal on each episode

Meta-RL Example #2

Qu, Yang, Setlur, Tunstall, Beeching, Salakhutdinov, Kumar. *Optimizing Test-Time Compute via Meta Reinforcement Finetuning*. ICML 2019.

Tasks: solving different math problems

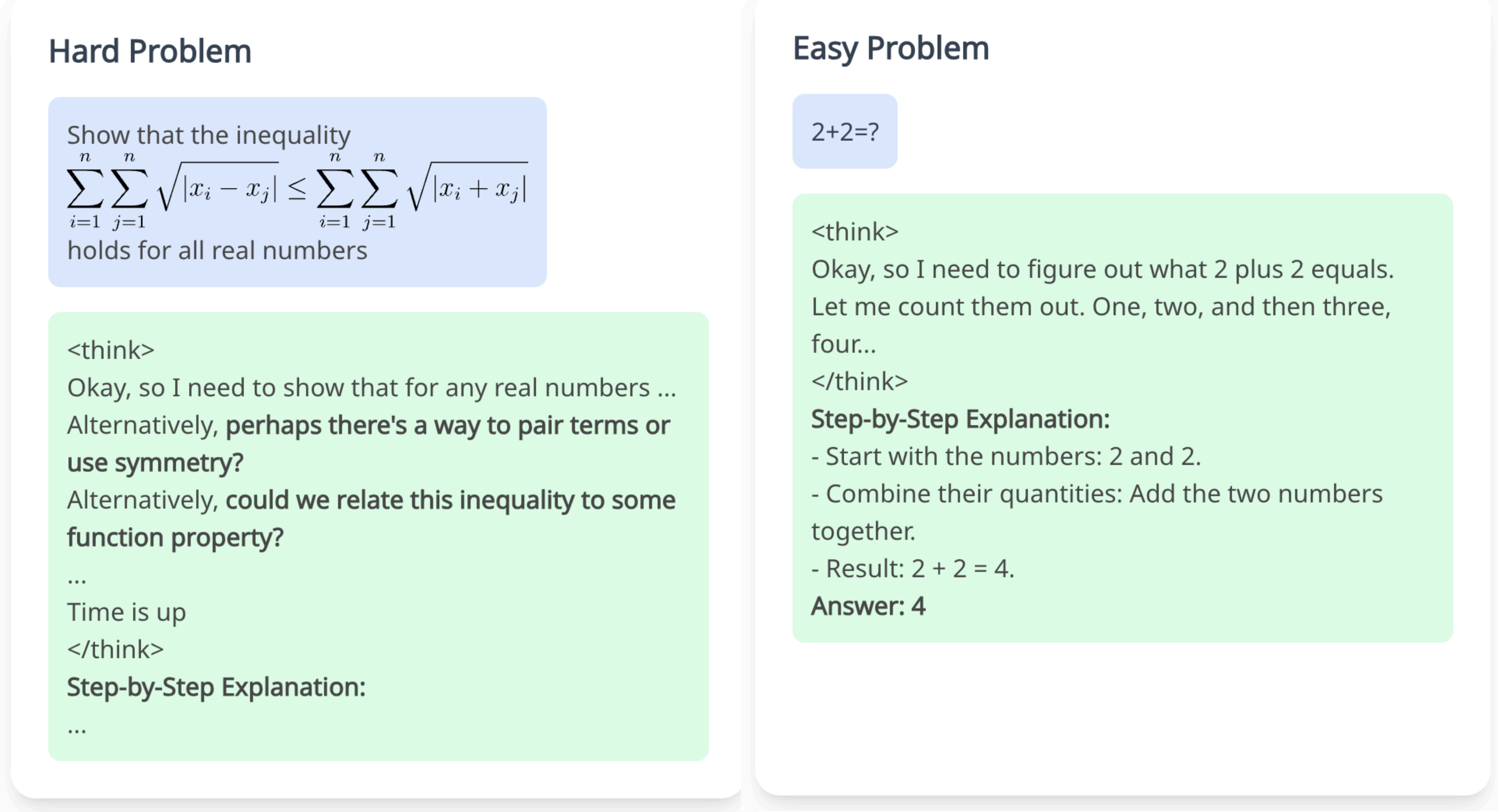
Show that the inequality

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sqrt{|x_i - x_j|} \leq \sum_{i=1}^{n}\sum_{j=1}^{n}\sqrt{|x_i + x_j|}$$

holds for all real numbers

Explore different strategies for solving the problem & then produce answer based on best strategy.

DeepSeek R1



Hard Problem

Show that the inequality

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sqrt{|x_i - x_j|} \leq \sum_{i=1}^{n}\sum_{j=1}^{n}\sqrt{|x_i + x_j|}$$

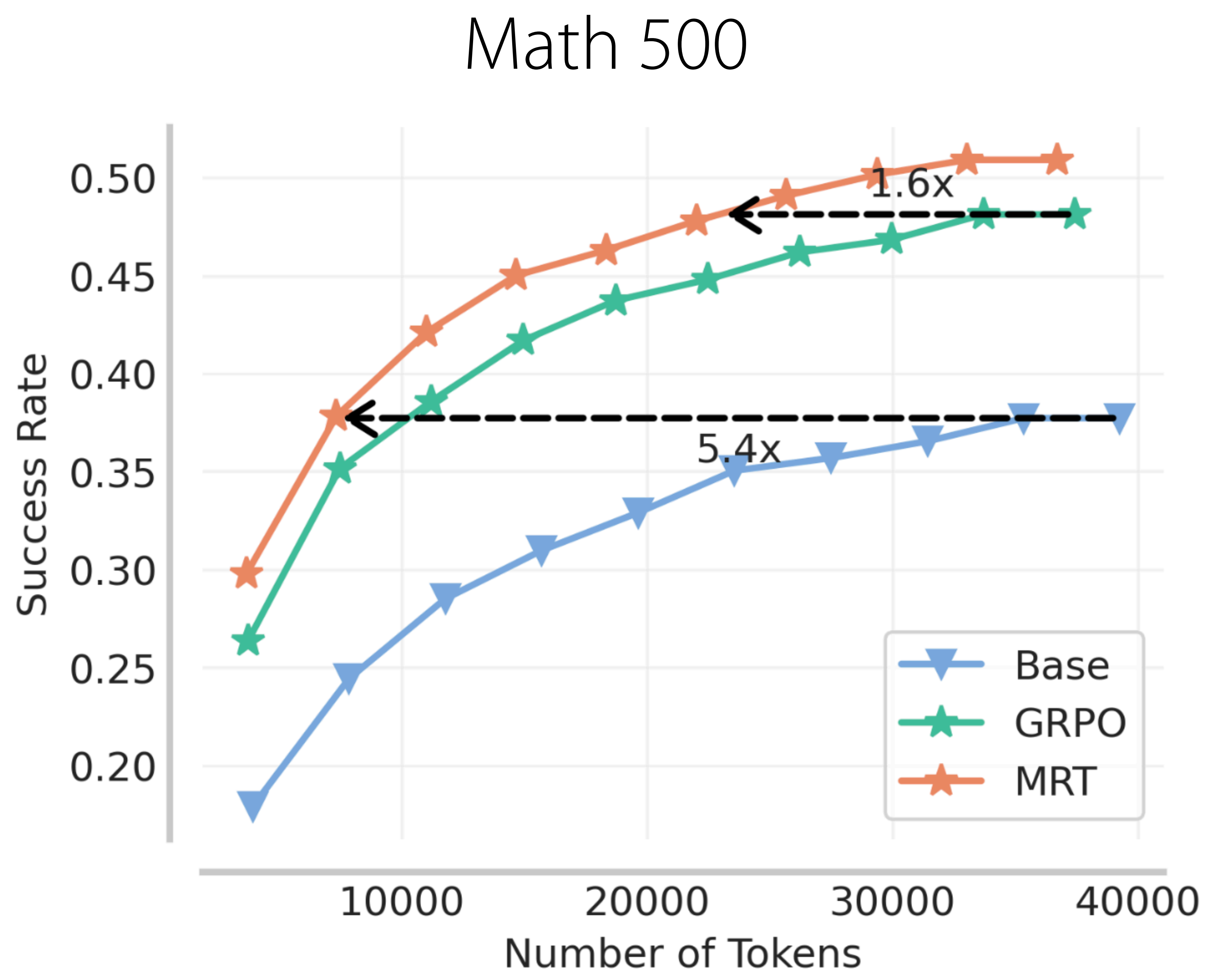
holds for all real numbers

<think>
Okay, so I need to show that for any real numbers ...
Alternatively, **perhaps there's a way to pair terms or use symmetry?**
Alternatively, **could we relate this inequality to some function property?**
...
Time is up
</think>
Step-by-Step Explanation:
...

Easy Problem

2+2=?

<think>
Okay, so I need to figure out what 2 plus 2 equals. Let me count them out. One, two, and then three, four...
</think>
Step-by-Step Explanation:
- Start with the numbers: 2 and 2.
- Combine their quantities: Add the two numbers together.
- Result: 2 + 2 = 4.
Answer: 4

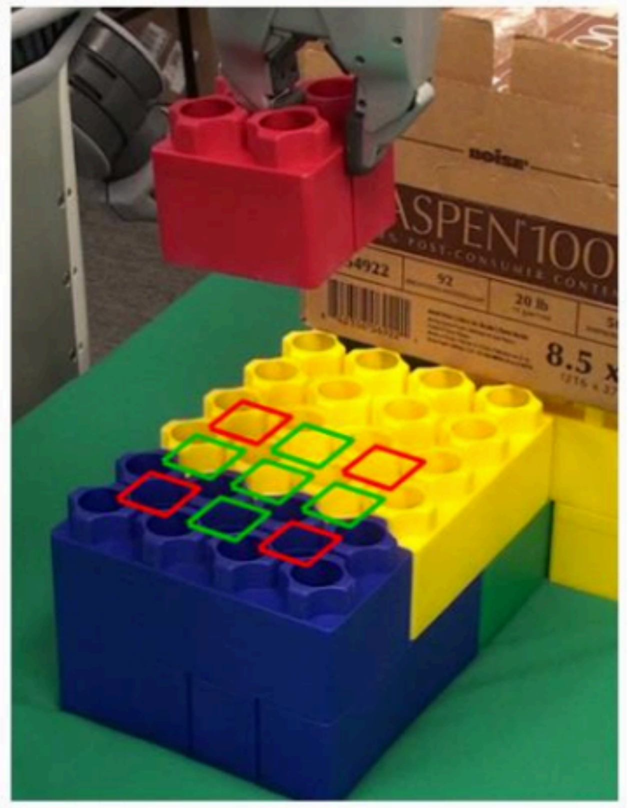


Higher performance with same # of tokens
Same accuracy with 1.6x fewer tokens

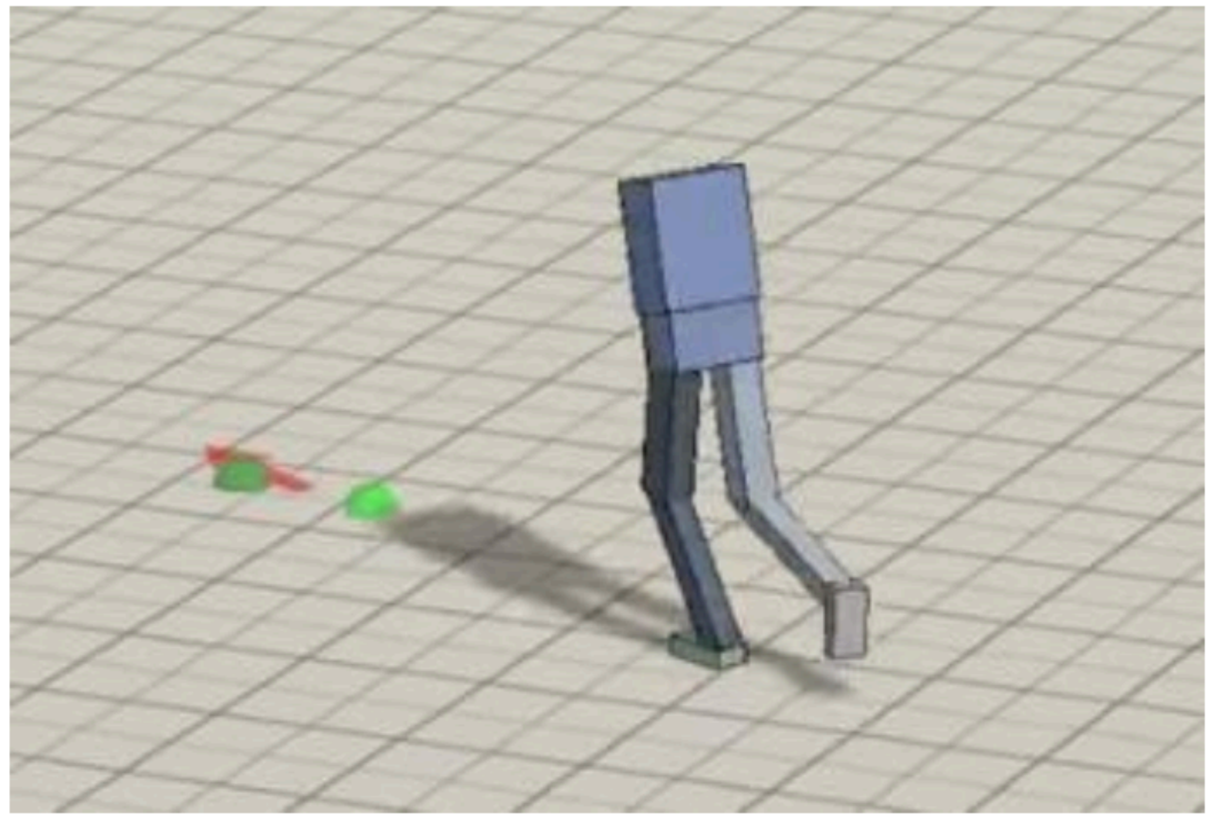
Key idea: Can we optimize test-time compute?

Digression: Connection to Multi-Task Policies

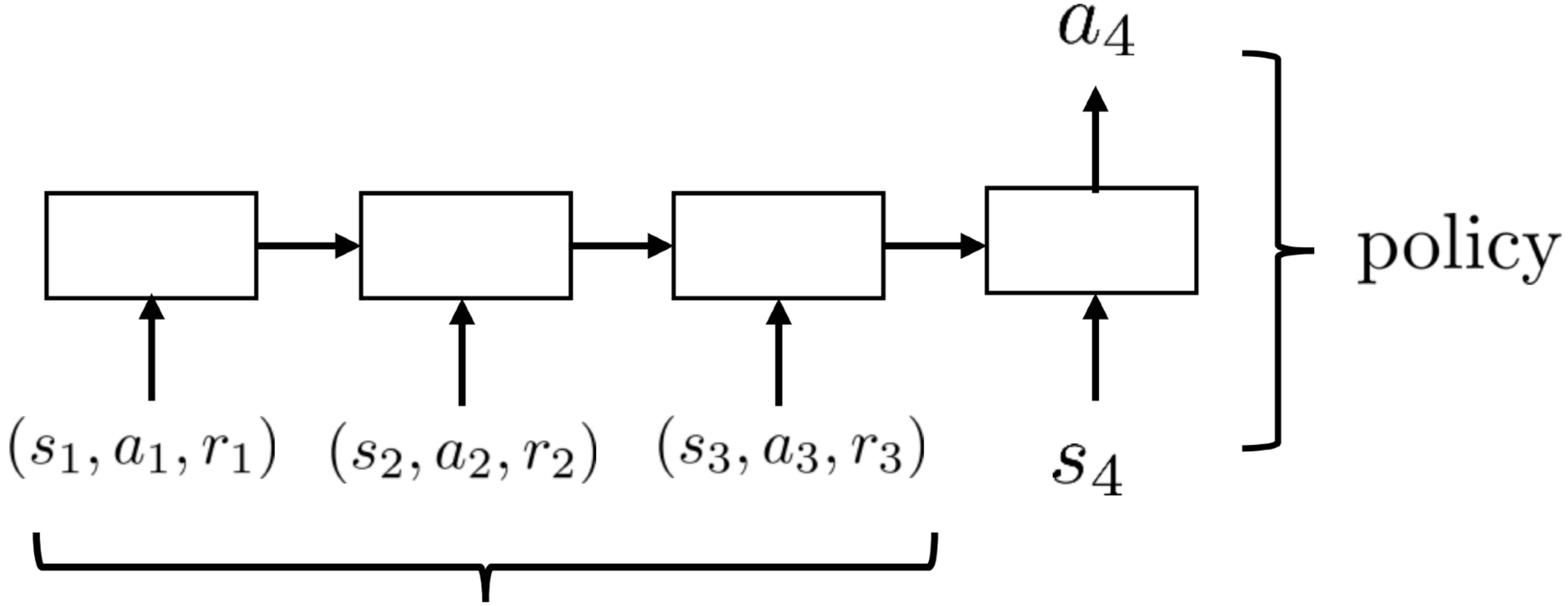
multi-task policy: $\pi_\theta(\mathbf{a} \mid \mathbf{s}, \mathbf{z}_i)$



$\mathbf{z}_i$: stack location



$\mathbf{z}_i$: walking direction



a_4

policy

(s_1, a_1, r_1) (s_2, a_2, r_2) (s_3, a_3, r_3) s_4

task identifier $\mathbf{z}_i$

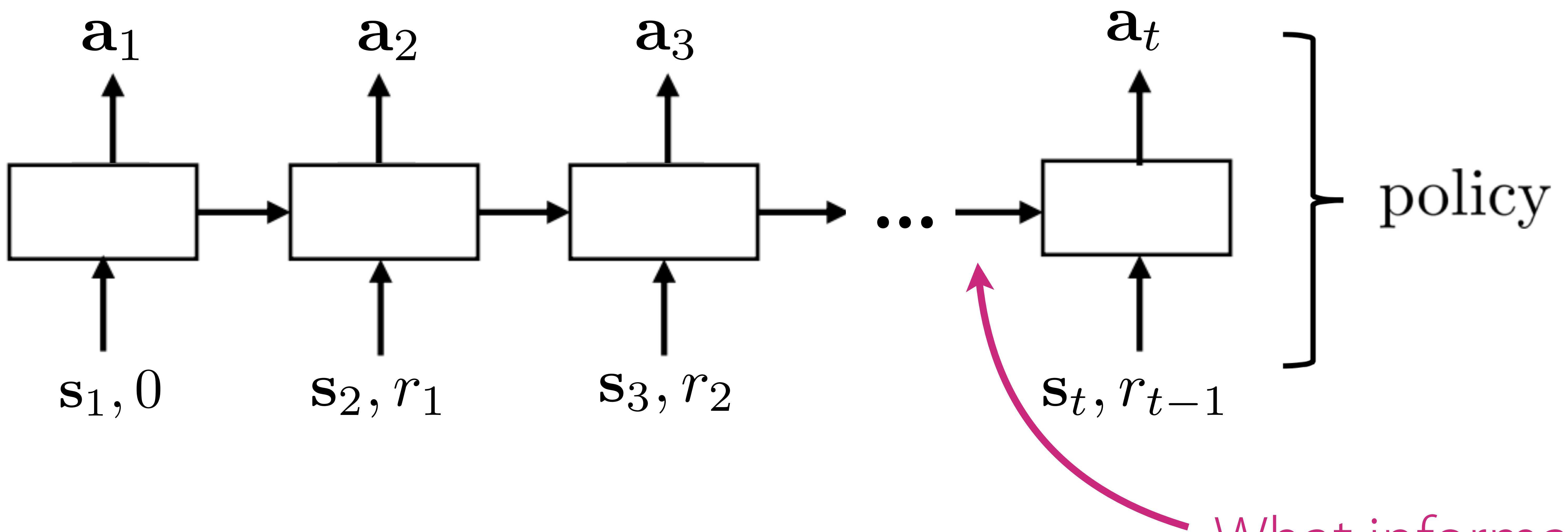
Multi-task policy with experience as task identifier.

What about **goal-conditioned policies**?

- rewards are a strict generalization of goals
- meta-RL objective is to *adapt* new tasks vs. ***generalize*** to new goals

(**k-shot** vs. **0-shot**)

Another way to look at it

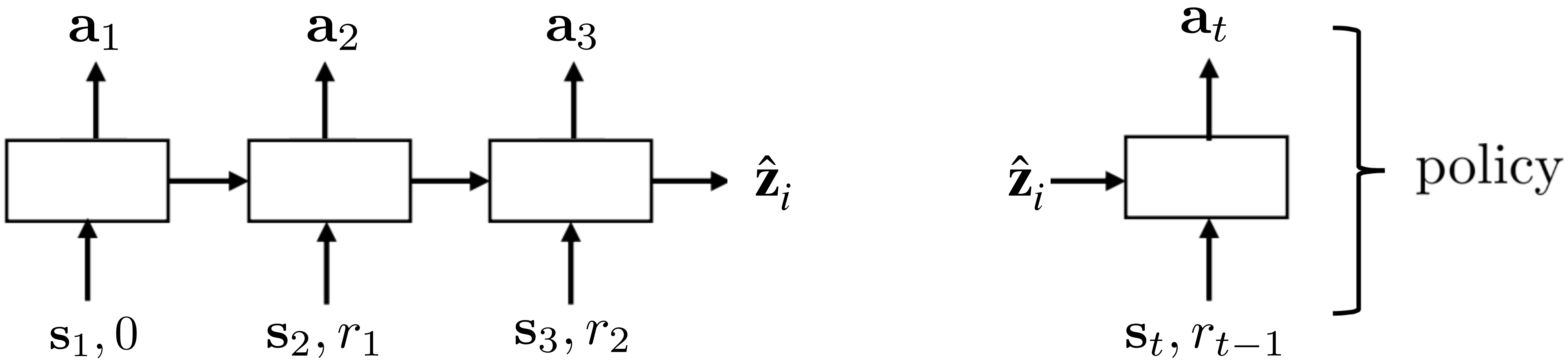


What information is passed here?

When $\mathbf{z}_i$ isn't in the state, there's ambiguity about the MDP.

Policy needs to *infer* $\mathbf{z}_i$ from its experience.

Another way to look at it



When $\mathbf{z}_i$ isn't in the state, there's ambiguity about the MDP.

Policy needs to *infer* $\mathbf{z}_i$ from its experience.

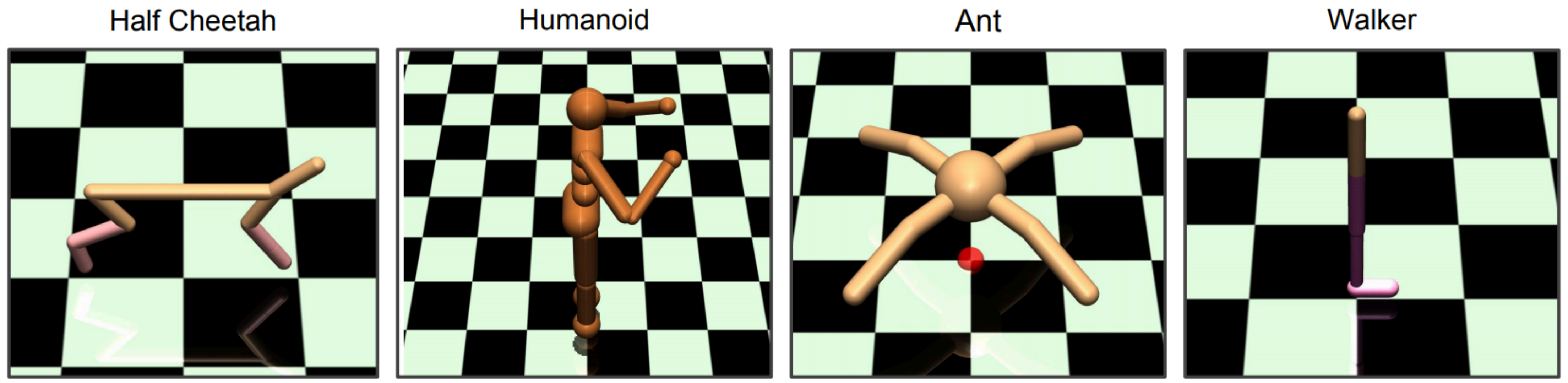
Meta-RL as exploring to infer the unknown task, and then performing the task.

A special kind of **partially observed** Markov decision process (POMDP)

Meta-RL Example #3

Rakelly, Zhou, Quillen, Finn, Levine. *Efficient Off-Policy Meta-Reinforcement Learning via Probabilistic Context Variables*. ICML 2019.

Experiment: Continuous control problems

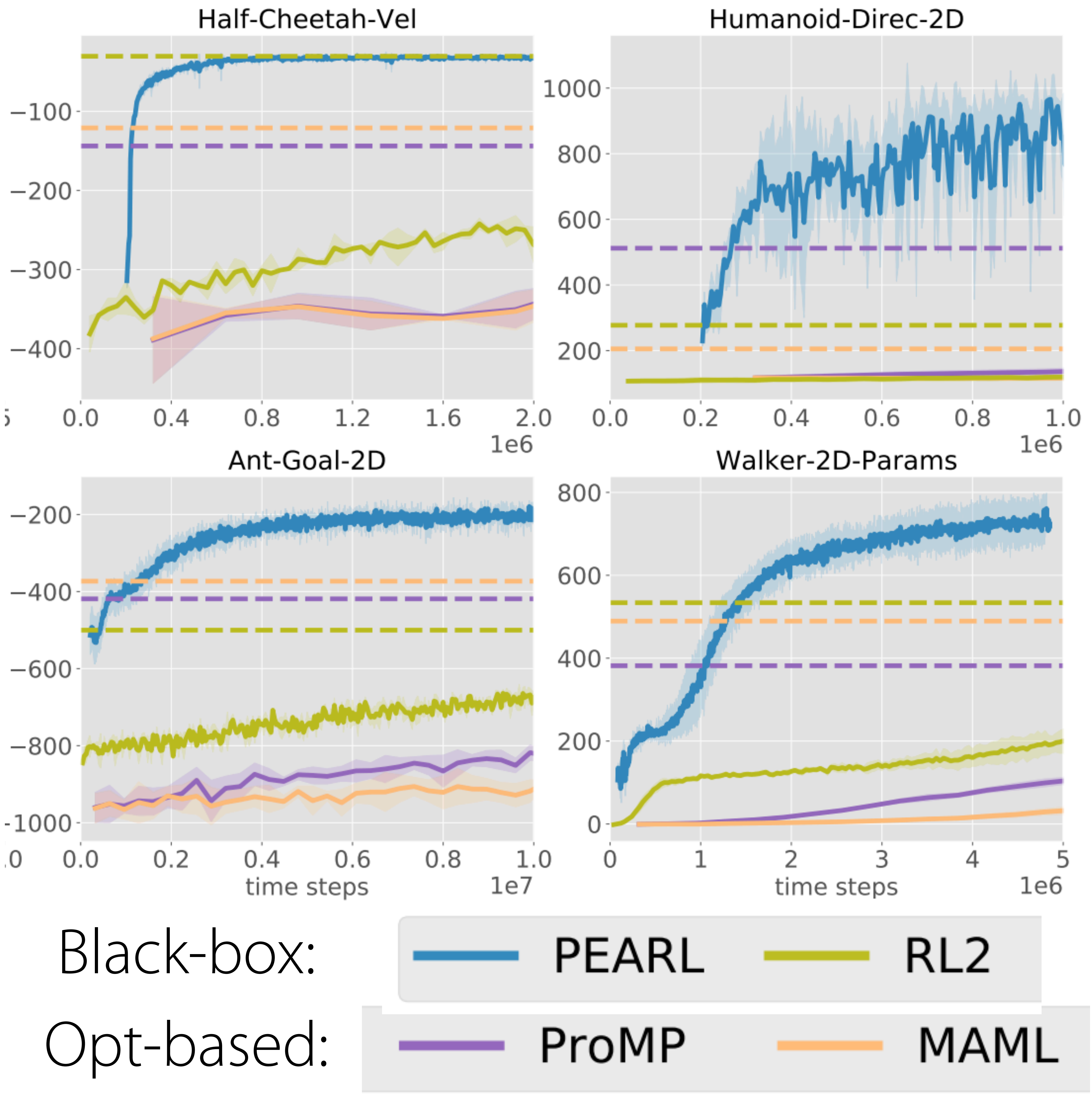


- different directions, velocities
- different physical dynamics

Meta-RL algos are very efficient at new tasks.

But, what about **meta-training efficiency**?

Question: Do you expect off-policy meta-RL to be more or less efficient than on-policy meta-RL?

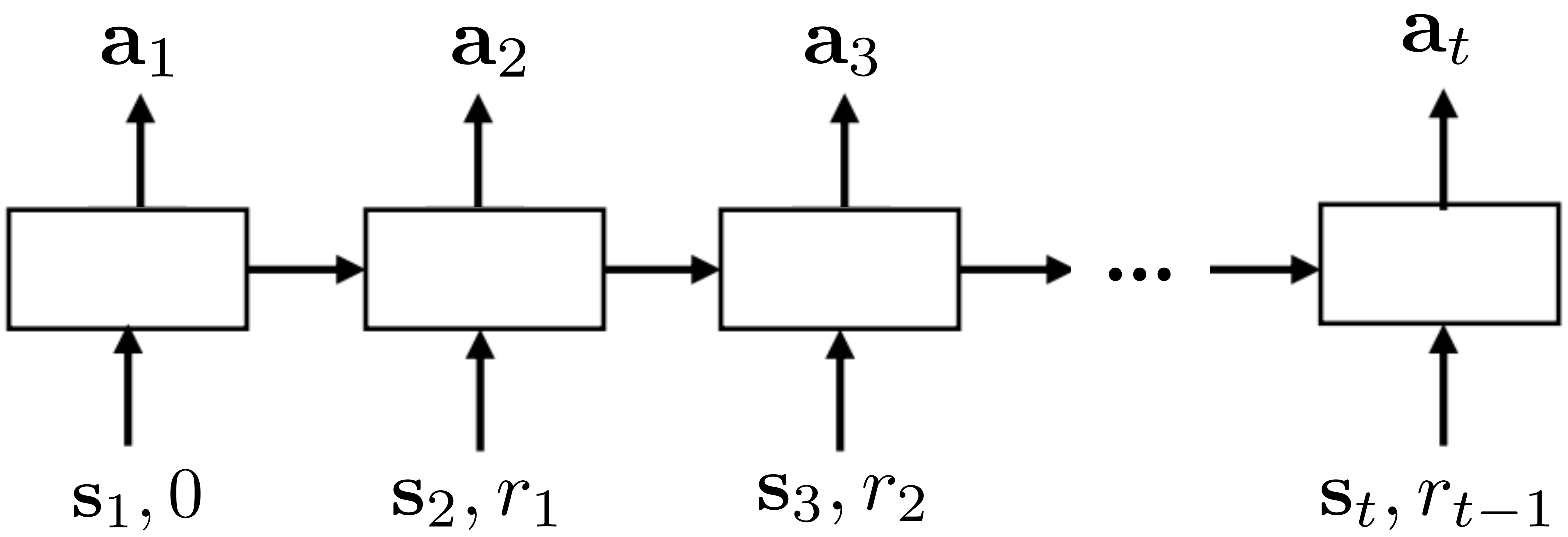


Summary So Far

Black-box neural net

(Transformer, NN with memory)

$$\mathbf{a} \sim \pi^{\text{task}}(\,\cdot\,|\,\mathbf{s}, D_{\text{train}})$$



+ general & expressive

+ a variety of design choices in architecture

-- hard to optimize

~ inherits sample efficiency from outer RL optimizer

How should we think about exploration in meta-RL?

How Do We Learn to Explore?

Solution #1: Optimize for Exploration & Exploitation *End-to-End* w.r.t. Reward

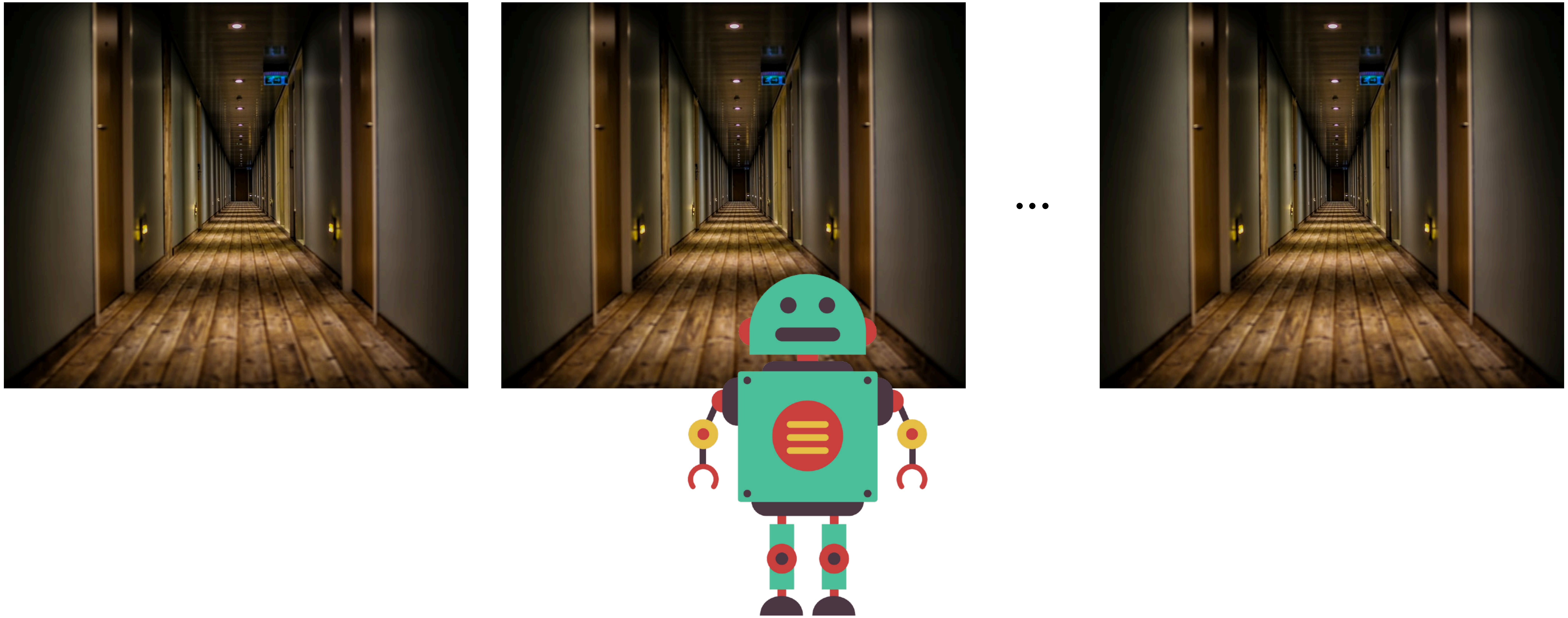
(Duan et al., 2016, Wang et al., 2016, Mishra et al., 2017, Stadie et al., 2018, Zintgraf et al., 2019, Kamienny et al., 2020)

+ simple

+ in principle, it yields optimal exploration-exploitation trade-off

-- challenging optimization when exploration is hard

A simple, running example



Hallway 1

Hallway 2

…

Hallway N

agent

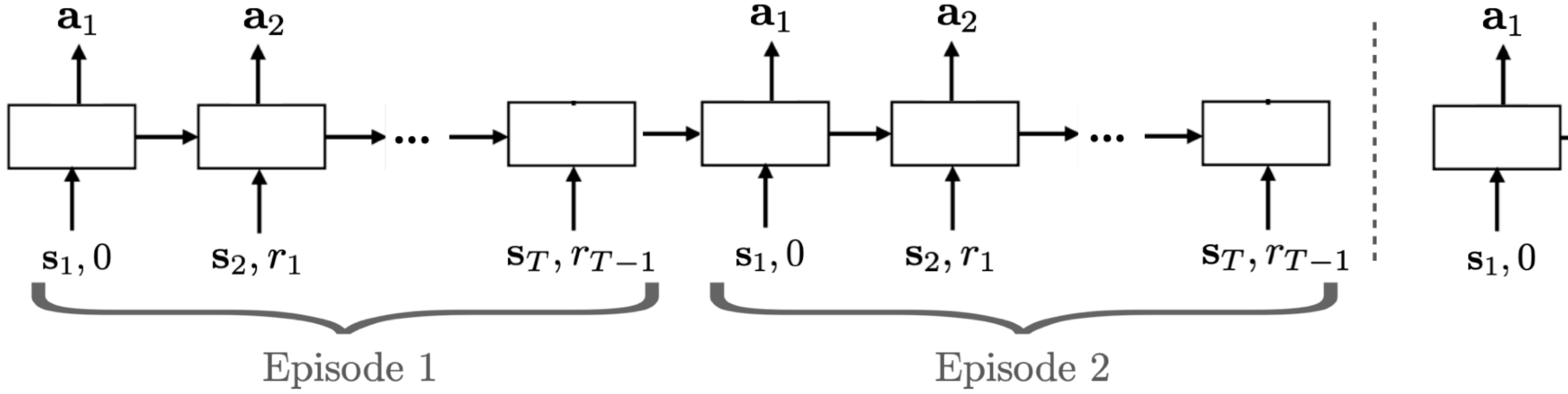
information on where to go

Different tasks: navigating to the ends of different hallways

How Do We Learn to Explore?

Solution #1: Optimize for Exploration & Exploitation *End-to-End* w.r.t. Task Reward

(Duan et al., 2016, Wang et al., 2016, Mishra et al., 2017, Stadie et al., 2018, Zintgraf et al., 2019, Kamienny et al., 2020)



Example episodes during meta-training:

agent goes to the end of the correct hallway

- gets positive reward for current task, but $\mathcal{D}_i^{\text{tr}}$ won't be different than for any other task

agent goes to wrong hallway then correct hallway

+/- provides signal on a **suboptimal** exploration + exploitation strategy

agent looks at the instructions

- good exploratory behavior, but won't get any reward for this behavior

It's hard to learn exploration & exploitation at the same time!

Another Example of a Hard Exploration Meta-RL Problem

Learned cooking tasks in previous kitchens

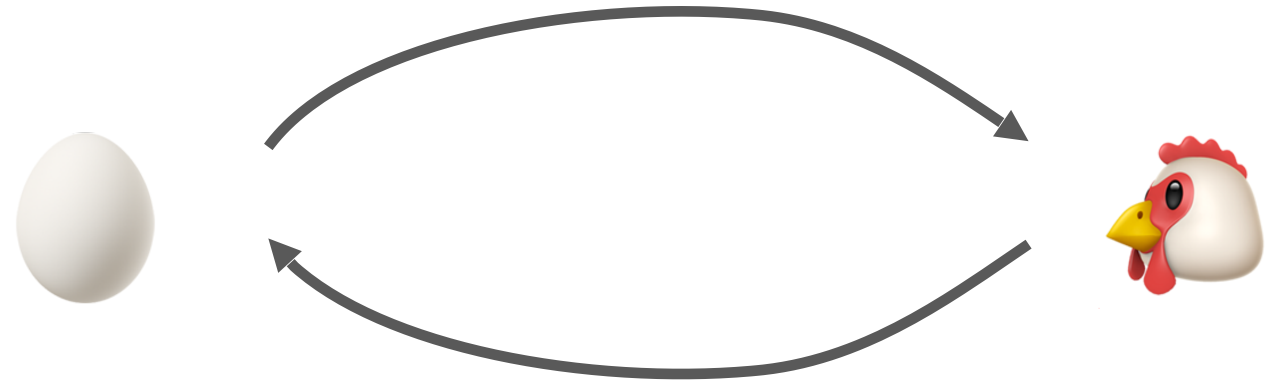
meta-training

Goal: Quickly learn tasks in a new kitchen.

meta-testing

Why is End-to-End Training Hard in This Example?

End-to-end approach: optimize exploration and execution episode behaviors end-to-end to maximize reward of execution



Learning to cook
(execution)

Learning to find ingredients
(exploration)

Ingredient not found
(bad exploration)
→
Cannot learn to cook
(bad execution)

Cannot cook
(bad execution)
→
Low reward for **any** exploration

Coupling problem: learning exploration and execution depend on each other

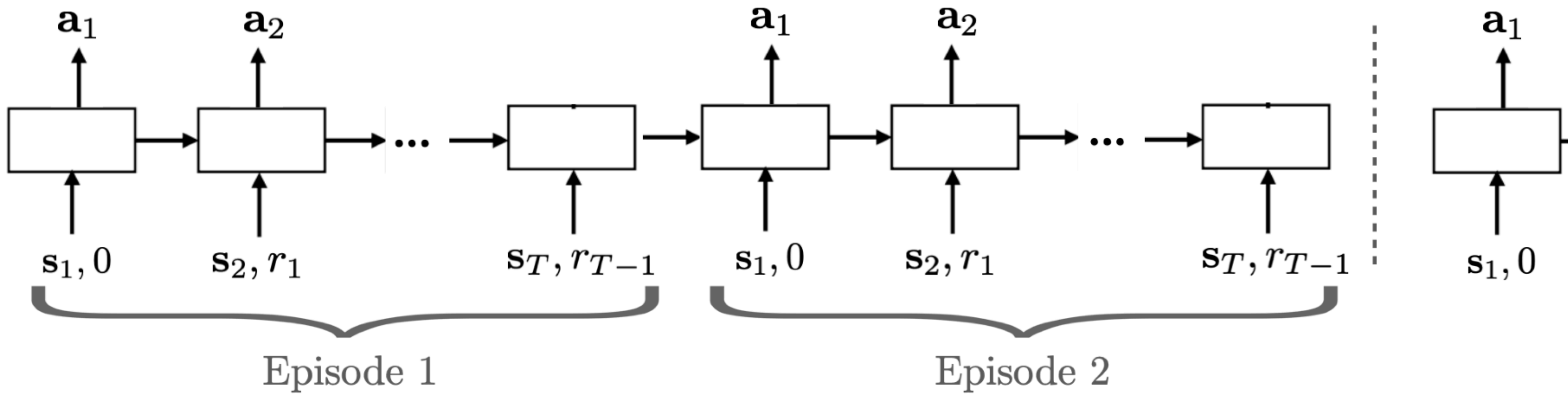
—> can lead to poor local optima, poor sample efficiency

Liu, Raghunathan, Liang, Finn. *Decoupling Exploration and Exploitation for Meta-Reinforcement Learning without Sacrifices*. ICML 2021

How Do We Learn to Explore?

Solution #1: Optimize for Exploration & Exploitation *End-to-End* w.r.t. Task Reward

(Duan et al., 2016, Wang et al., 2016, Mishra et al., 2017, Stadie et al., 2018, Zintgraf et al., 2019, Kamienny et al., 2020)



+ simple
+ leads to optimal strategy in principle

-- challenging optimization when exploration is hard

Solution #2: Leverage Alternative Exploration Strategies

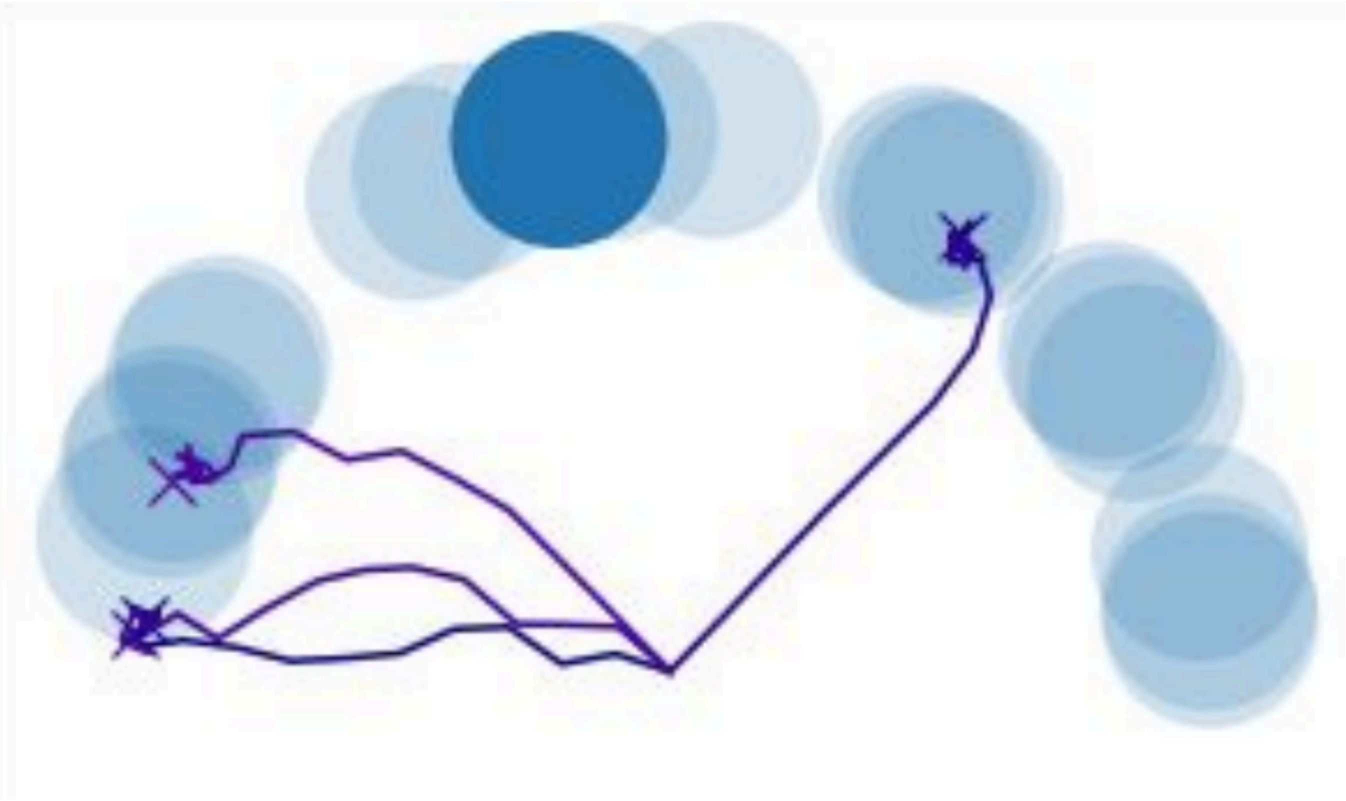
2a. Use posterior sampling

(also called Thompson sampling)

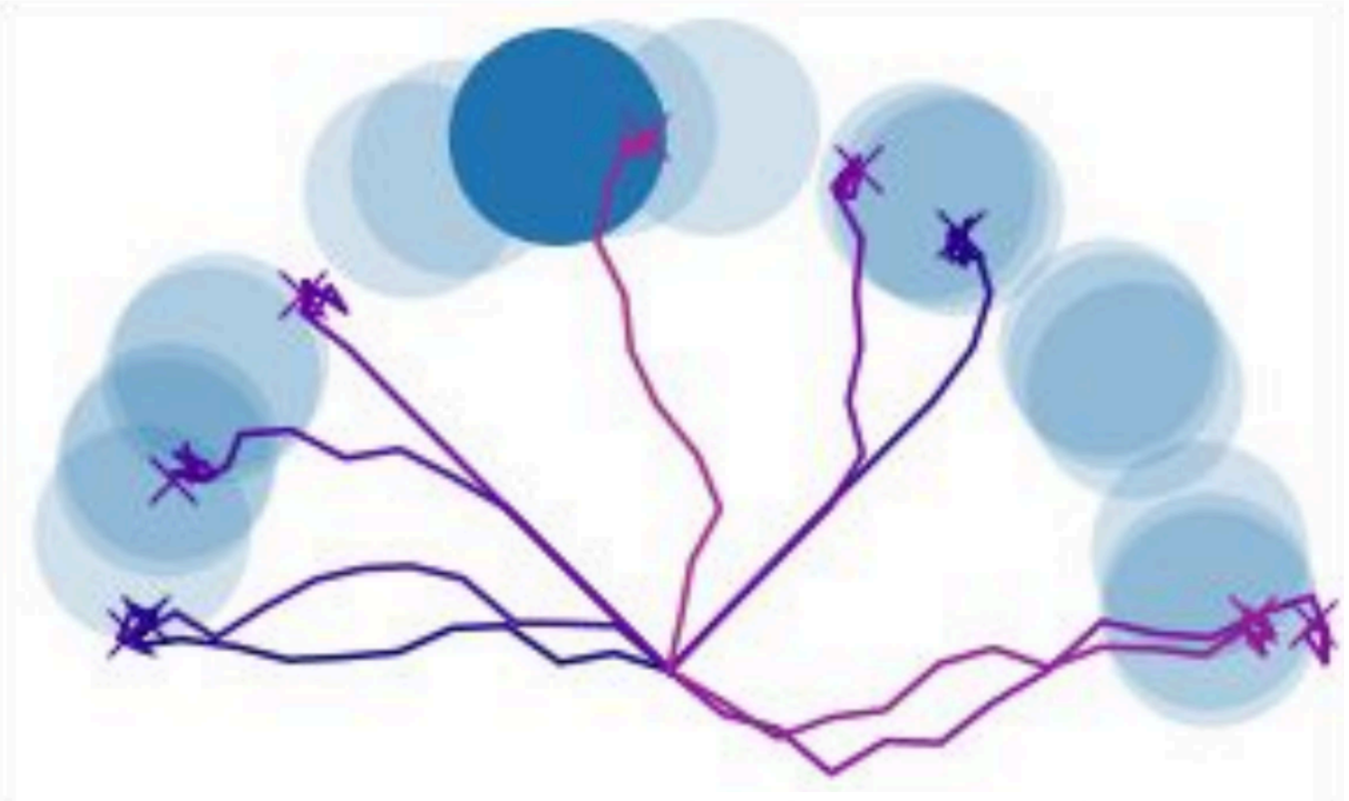
PEARL (Rakelly, Zhou, Quillen, Finn, Levine. ICML '19)

i. Learn distribution over latent task variable $p(\mathbf{z}), q(\mathbf{z} \mid \mathscr{D}_{\text{tr}})$ and corresponding task policies $\pi(\mathbf{a} \mid \mathbf{s}, \mathbf{z})$

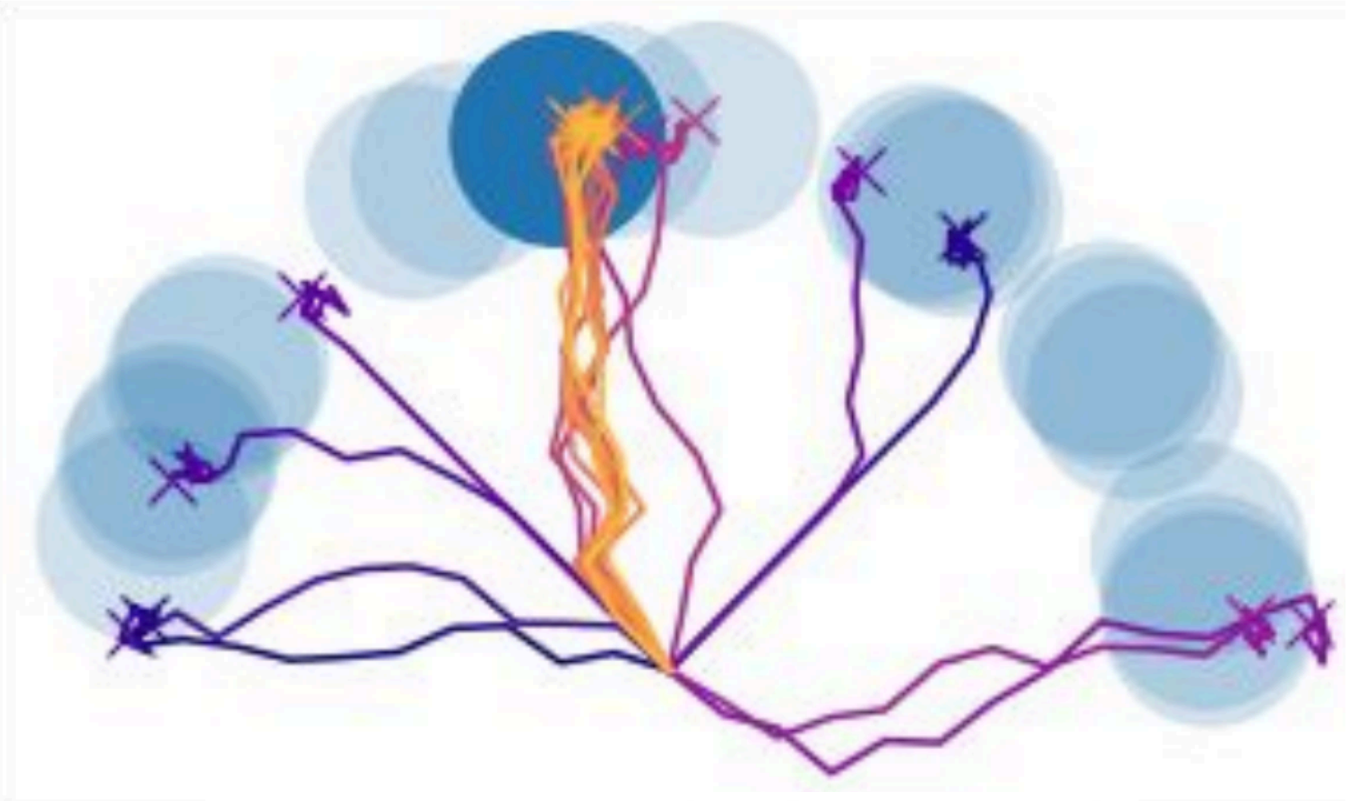
ii. Sample $\mathbf{z}$ from current *posterior* and sample from policy $\pi(\mathbf{a} \mid \mathbf{s}, \mathbf{z})$



$\mathbf{z} \sim p(\mathbf{z})$



$\mathbf{z} \sim q_\phi(\mathbf{z} | c_{1:10})$



$\mathbf{z} \sim q_\phi(\mathbf{z} | c_{1:30})$

When might posterior sampling be bad? Eg. Goals far away & sign on wall that tells you the correct goal.

Solution #2: Leverage Alternative Exploration Strategies

2a. Use posterior sampling PEARL (Rakelly, Zhou, Quillen, Finn, Levine. ICML '19)

(also called Thompson sampling)

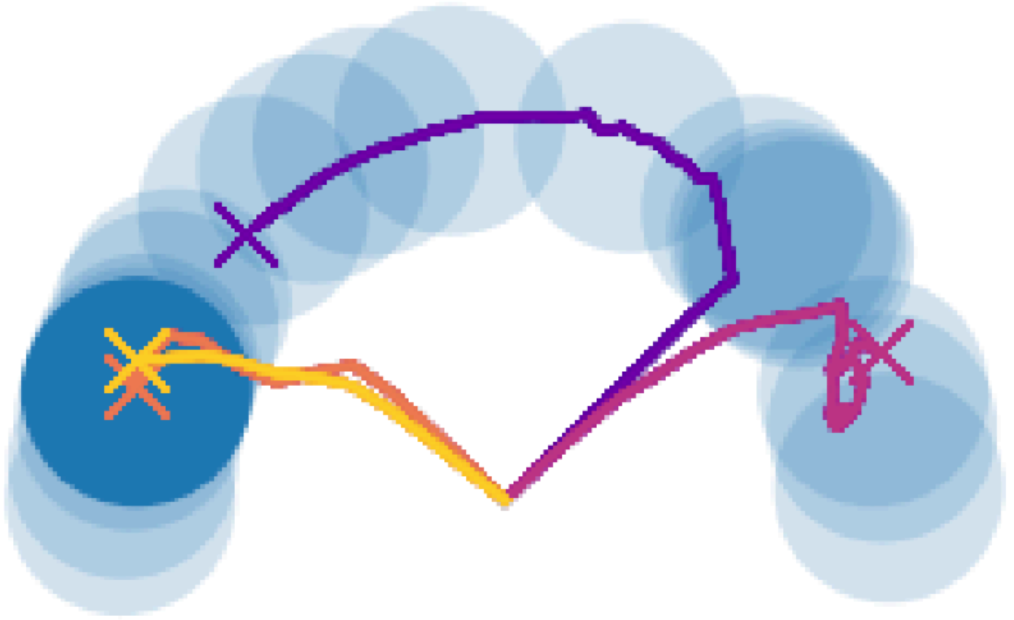
i. Learn distribution over latent task variable $p(\mathbf{z}), q(\mathbf{z} \mid \mathscr{D}_{\text{tr}})$ and corresponding task policies $\pi(\mathbf{a} \mid \mathbf{s}, \mathbf{z})$

ii. Sample $\mathbf{z}$ from current *posterior* and sample from policy $\pi(\mathbf{a} \mid \mathbf{s}, \mathbf{z})$

2b. Use intrinsic rewards MAME (Gurumurthy, Kumar, Sycara. CoRL '19)

2c. Task dynamics & reward prediction MetaCURE (Zhang, Wang, Hu, Chen, Fan, Zhang. '20)

i. Train model $f(\mathbf{s}', r \mid \mathbf{s}, \mathbf{a}, \mathscr{D}_{\text{train}})$ ii. Collect $\mathscr{D}_{\text{train}}$ so that model is accurate.



When might this be bad?

Lots of distractors,
or complex, high-dim state dynamics

Solution #2: Leverage Alternative Exploration Strategies

2a. Use posterior sampling PEARL (Rakelly, Zhou, Quillen, Finn, Levine. ICML '19)

(also called Thompson sampling)

i. Learn distribution over latent task variable $p(\mathbf{z}), q(\mathbf{z} | \mathscr{D}_{\text{tr}})$ and corresponding task policies $\pi(\mathbf{a} | \mathbf{s}, \mathbf{z})$

ii. Sample $\mathbf{z}$ from current *posterior* and sample from policy $\pi(\mathbf{a} | \mathbf{s}, \mathbf{z})$

2b. Use intrinsic rewards MAME (Gurumurthy, Kumar, Sycara. CoRL '19)

2c. Task dynamics & reward prediction MetaCURE (Zhang, Wang, Hu, Chen, Fan, Zhang. '20)

i. Train model $f(\mathbf{s}', r | \mathbf{s}, \mathbf{a}, \mathscr{D}_{\text{train}})$

ii. Collect $\mathscr{D}_{\text{train}}$ so that model is accurate.

+ easy to optimize

+ many based on principled strategies

-- suboptimal by arbitrarily large amount in some environments.

Plan for Today

1. Meta-RL problem statement
2. Black-box meta-RL methods

Next time: Exploration + Learning to explore.

part of HW4

Lecture goals:

- Understand the meta-RL problem statement & set-up
- Understand the basics & challenges of black-box meta RL algorithms

Next time

Today: meta-RL basics

Friday: exploration & meta-exploration

Reminders

Homework 3 due **Friday**

Project milestone due **next Friday**