

# Meta Reinforcement Learning

CS 224R

# Reminders

Midterm this **Friday morning** during normal lecture slot

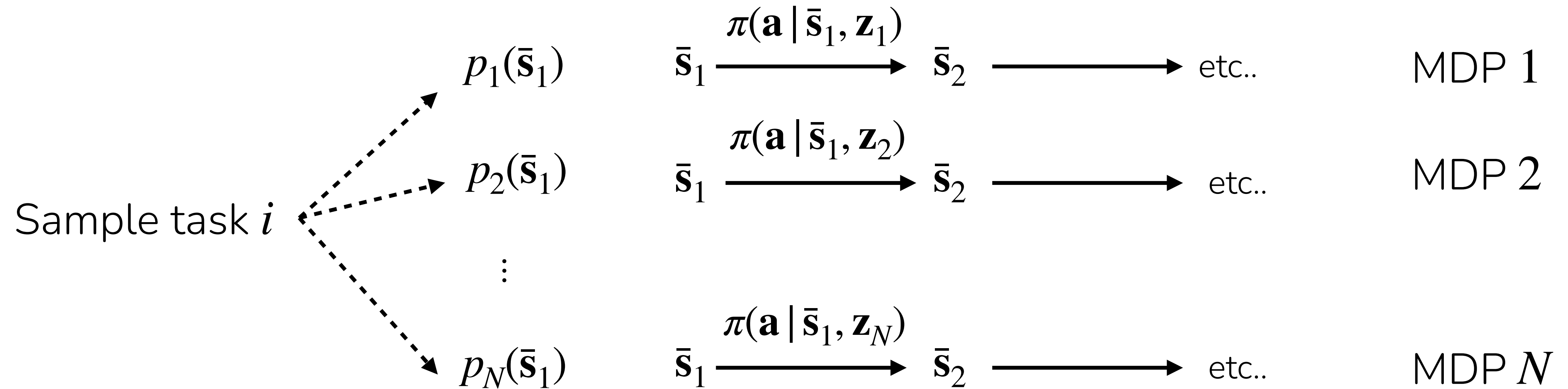
- Part of AIWG proctoring pilot.
- Please arrive to assigned location 5-10 minutes early, so we can start on-time
- Bring your ID
- Sign out on roster sheet when handing completed exam to CA

Project milestone due **next Friday**

# Summary: Multi-task learning

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

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



# Summary: 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 fairly hard to train

# Summary: Multi-task & goal-conditioned RL

## Weight sharing

Train network to do all tasks, conditioned on  $\mathbf{z}_i$

## Data sharing

Add data collected for one task to buffer for another by **relabeling** the reward and task identifier

Requires:

- same dynamics across tasks
- evaluatable reward functions
- off-policy learning algorithm

Directly applicable to goal-conditioned RL setting

# Plan for Today

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

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

# 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			

$D_{\text{train}}$



$x$

$\hat{y} = \text{Braque}$

In-context learning in LLMs

```
Poor English input: I ate 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.
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.
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.
```

$D_{\text{train}}$

$x$

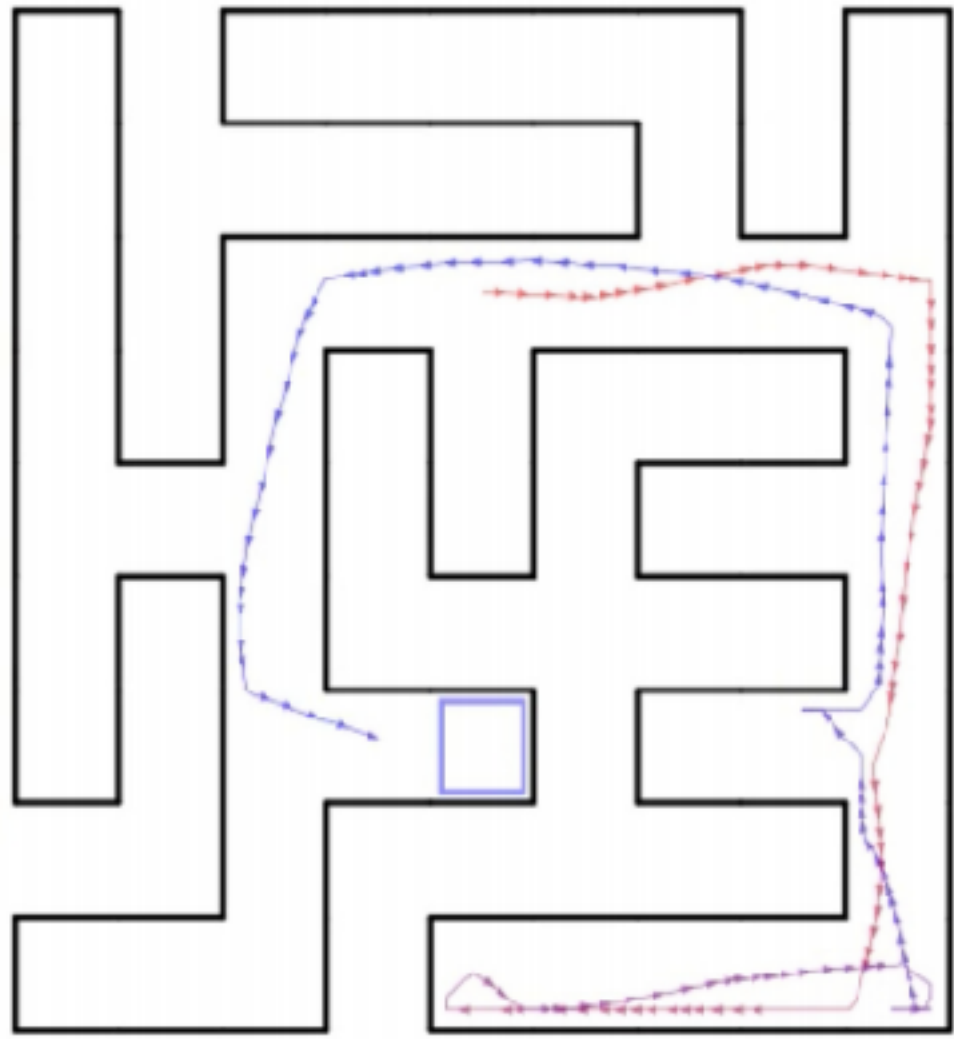
$\hat{y}$

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

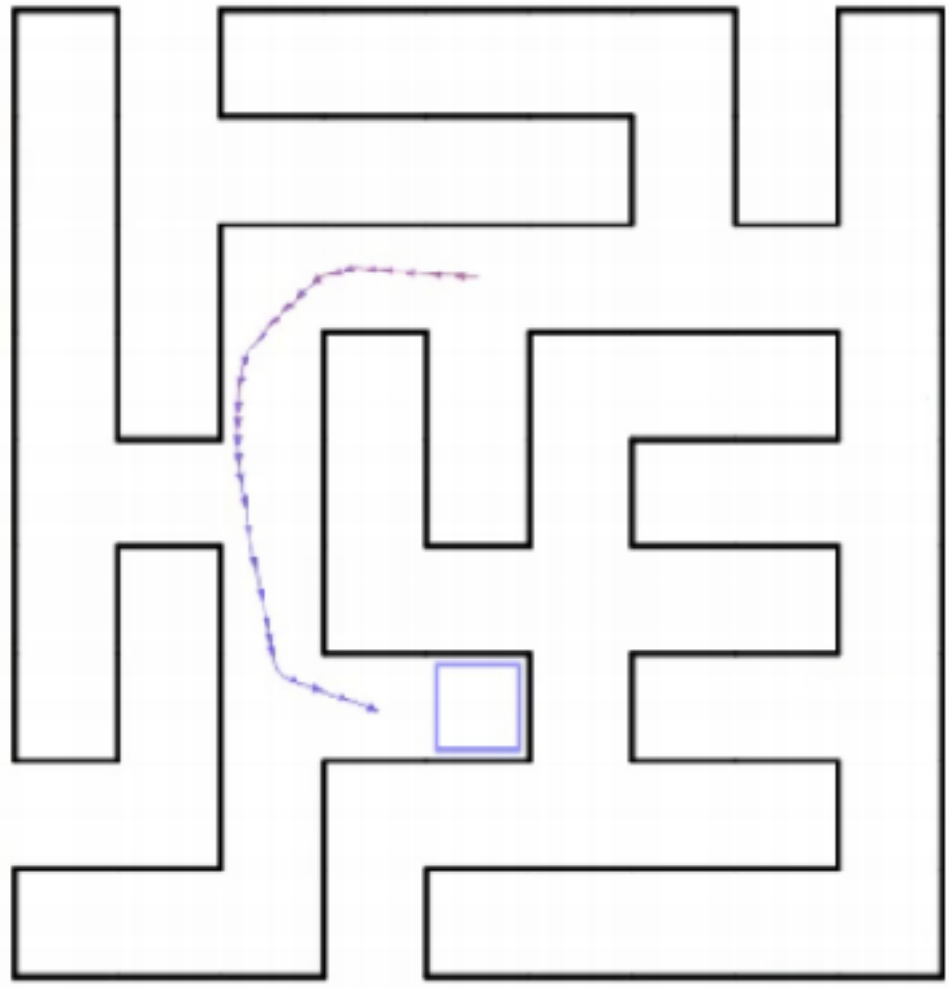
Collect small amount of experience in new MDP

Goal:



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

Learn policy that solves that MDP



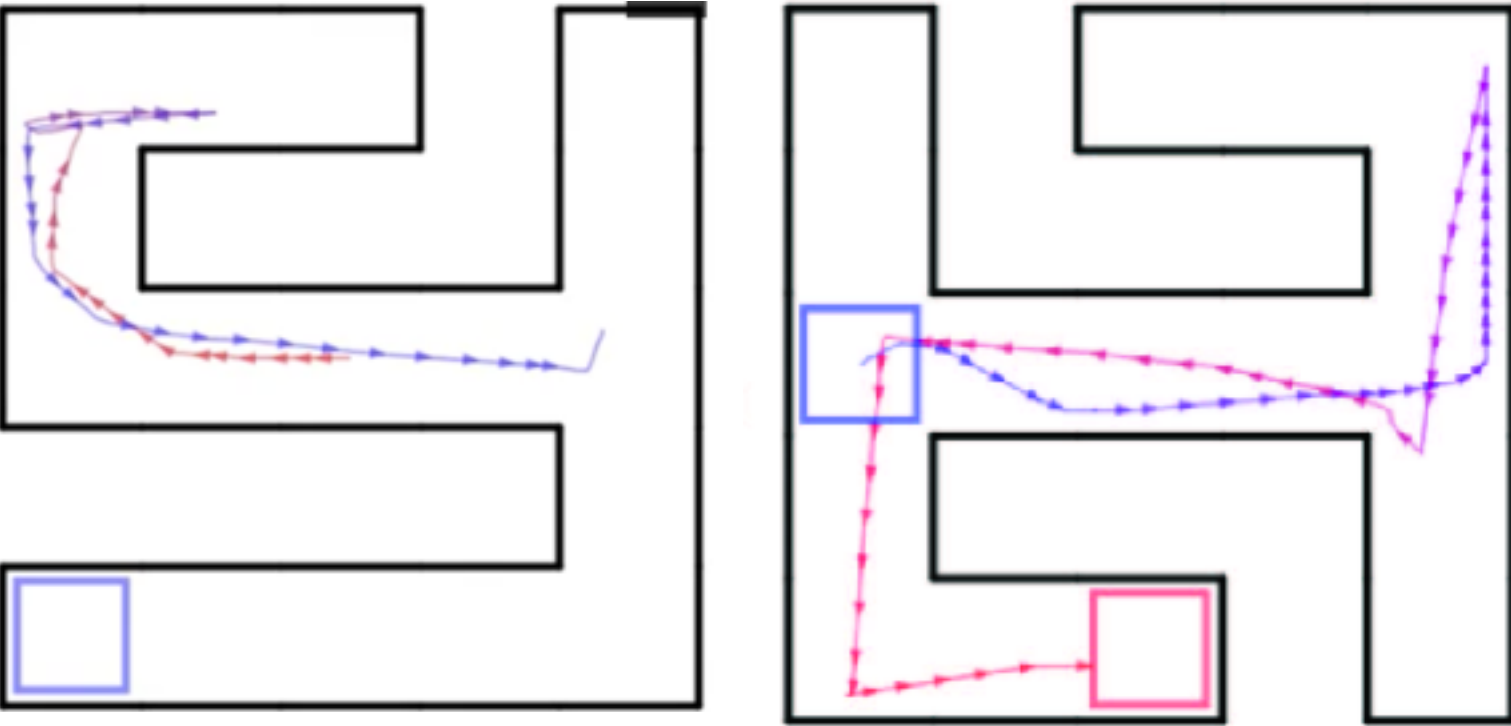
$\mathcal{D}_{tr} \rightarrow \pi^{task}$

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

# Meta-RL Example: Maze Navigation

## Meta-Train Time:

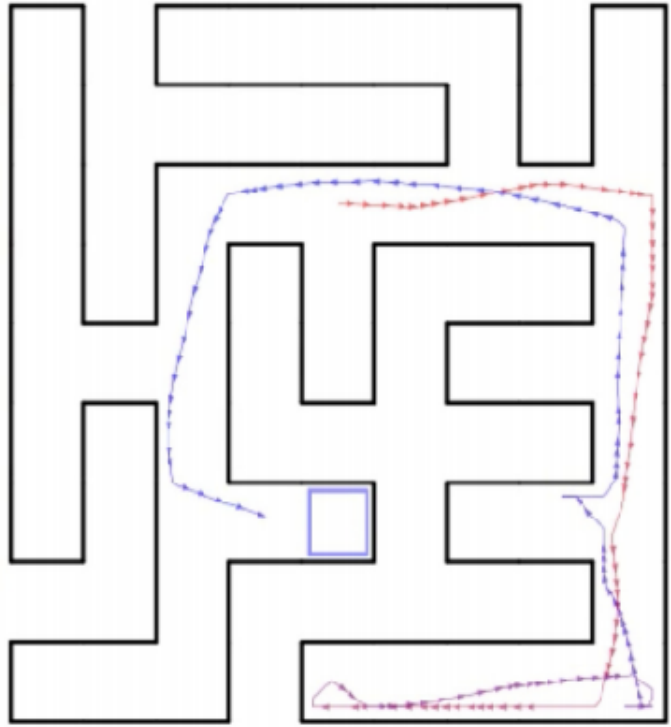
Learn how to efficiently explore & solve many MDPs:



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

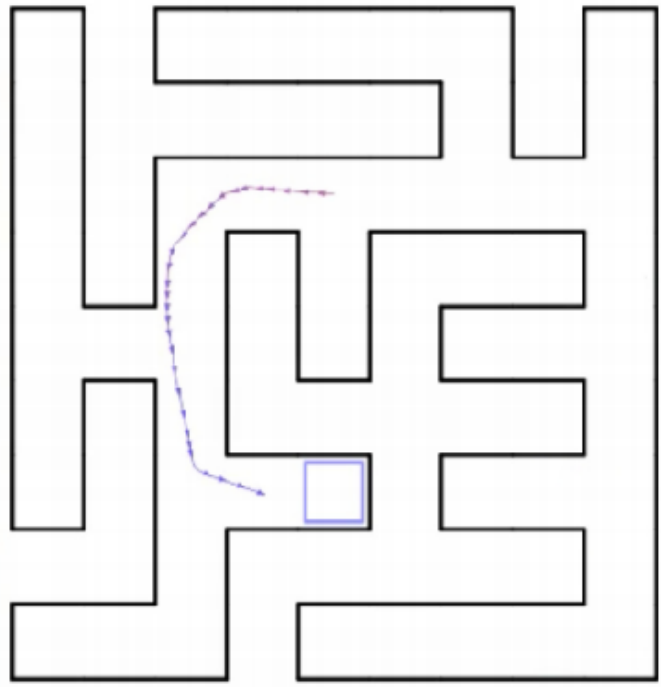
## Meta-Test Time:

Collect small amount of experience in new MDP



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

Learn policy that solves that MDP



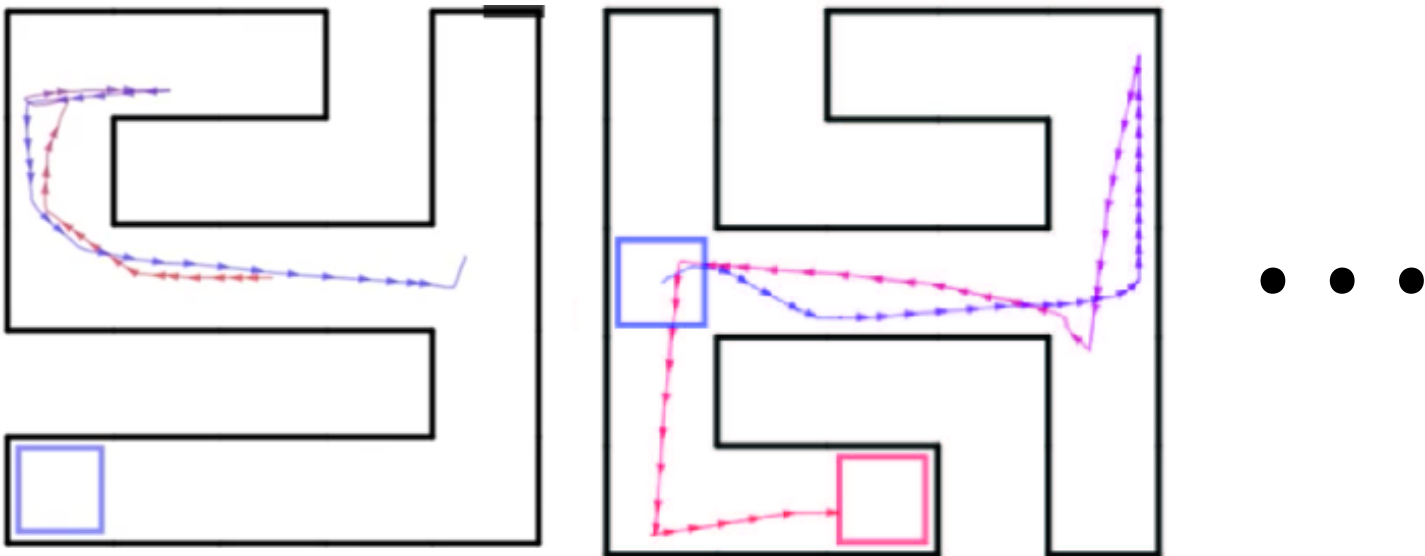
$\mathcal{D}_{\text{tr}} \rightarrow \pi^{\text{task}}$

Key assumption: Meta-testing MDPs come from same task distribution as meta-training MDPs.  
(so that we can expect generalization)

# Meta-reinforcement learning task distributions

## Examples of meta-RL tasks

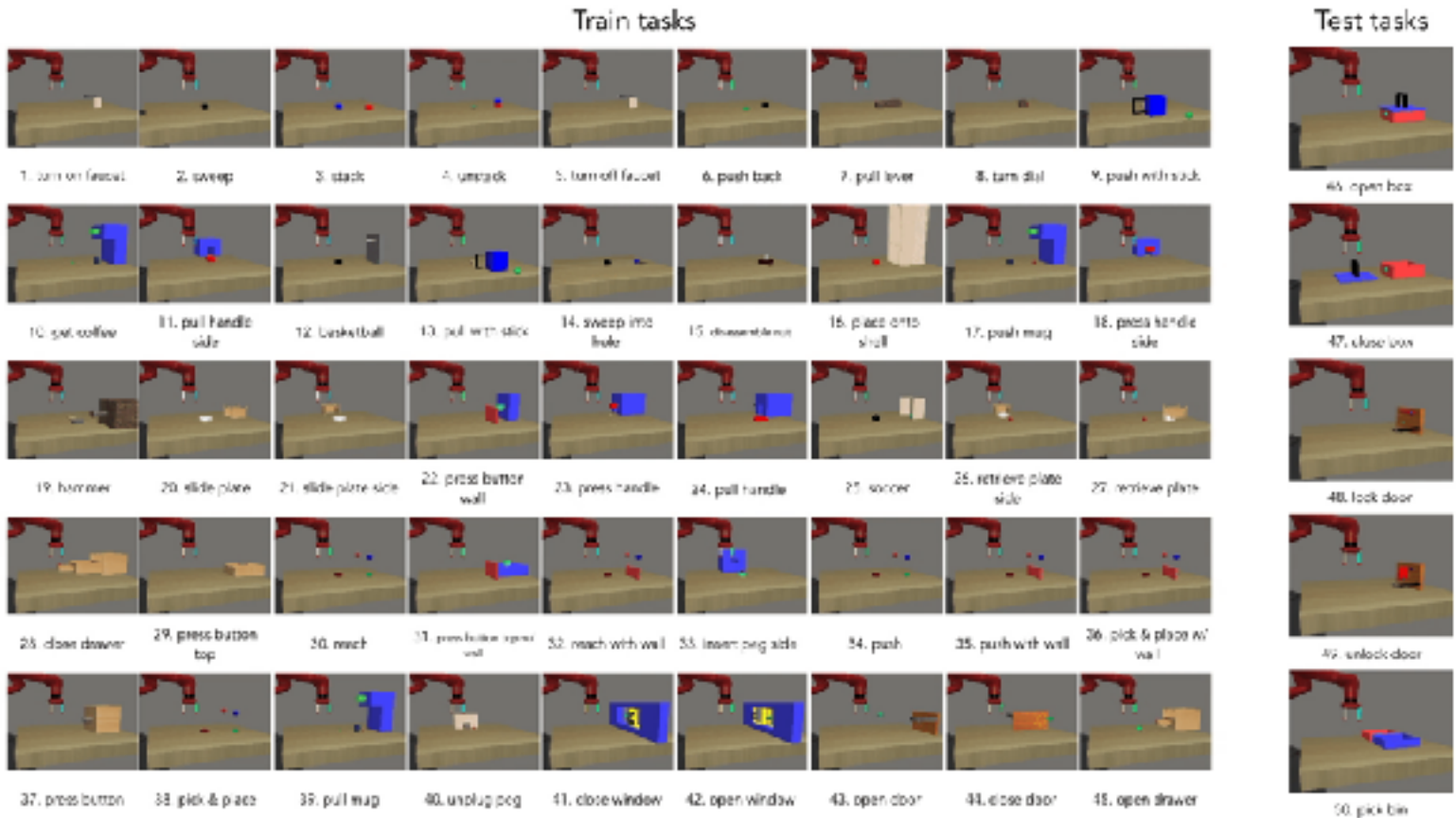
### Navigation through different mazes



### 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  $\mathcal{T}_1, \dots, \mathcal{T}_T$  at once.

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

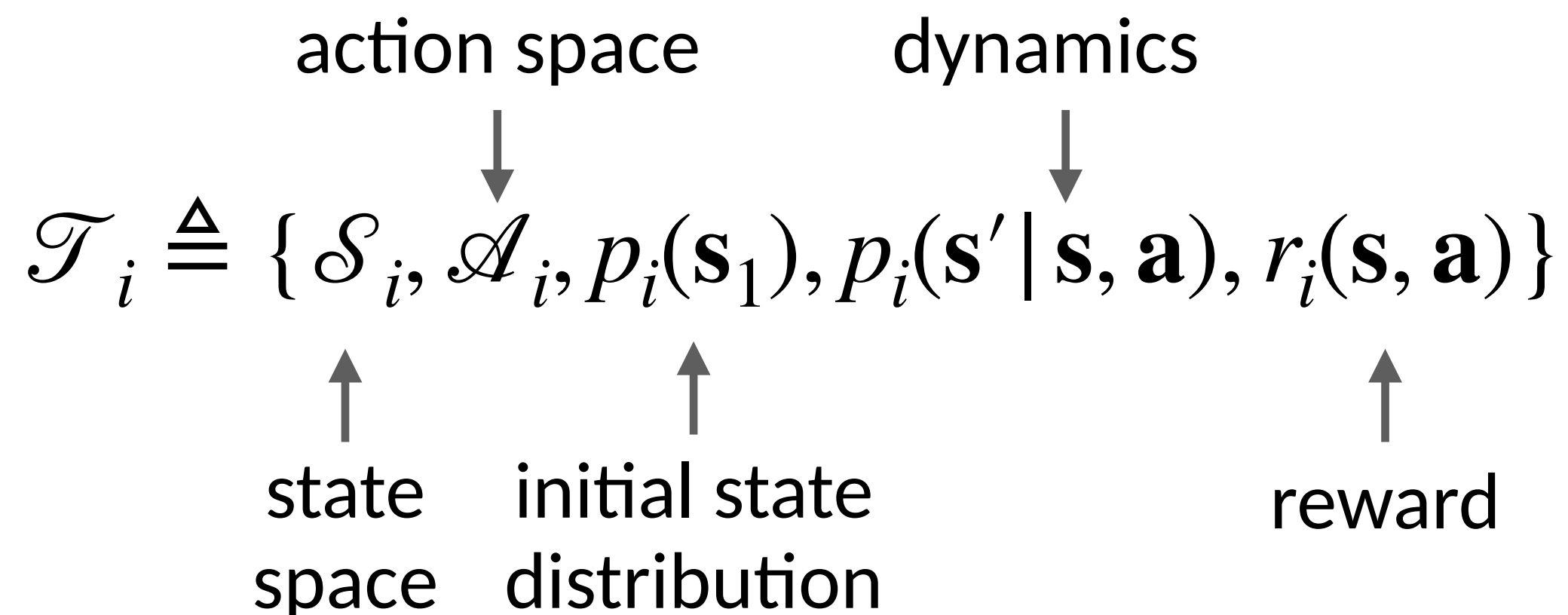
## Transfer Learning

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  $\mathcal{T}_1, \dots, \mathcal{T}_n$ , quickly solve new task  $\mathcal{T}_{\text{test}}$

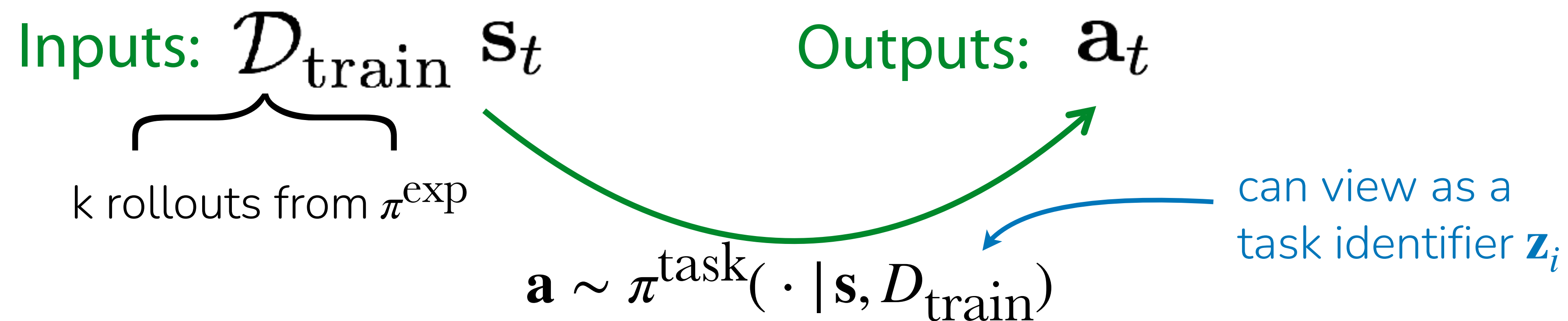
A reinforcement learning **task**:



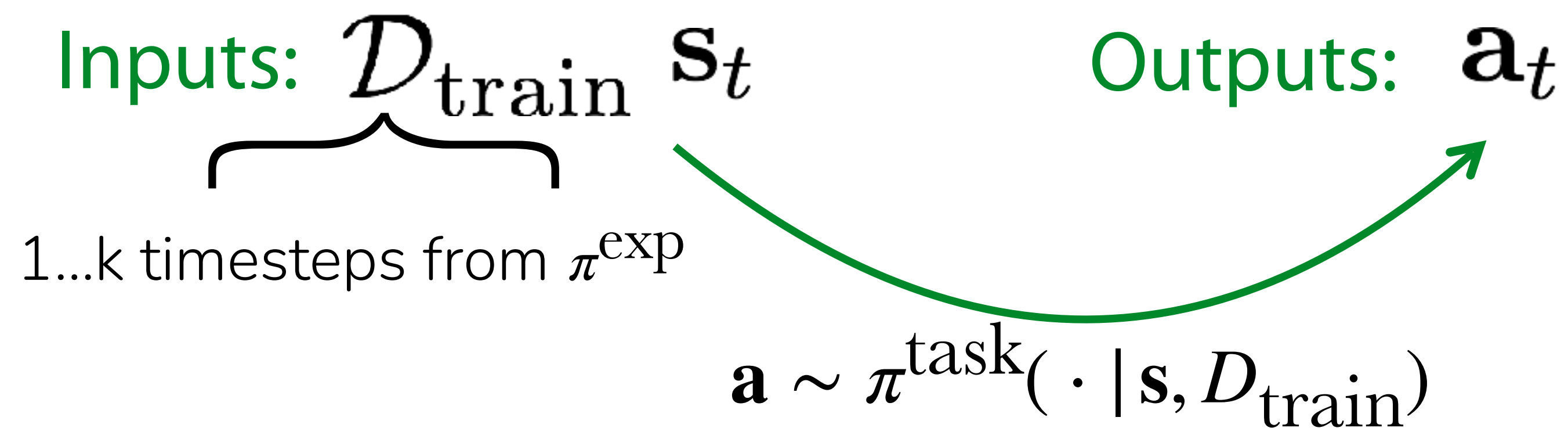
Meta-reinforcement learning  
= **meta-learning** with RL tasks

# The Meta Reinforcement Learning Problem

*Episodic Variant*



*Online Variant*



**Note:** exploration policy  $\pi^{\text{exp}}$  and adaptation policy  $\pi^{\text{task}}$  could share parameters.

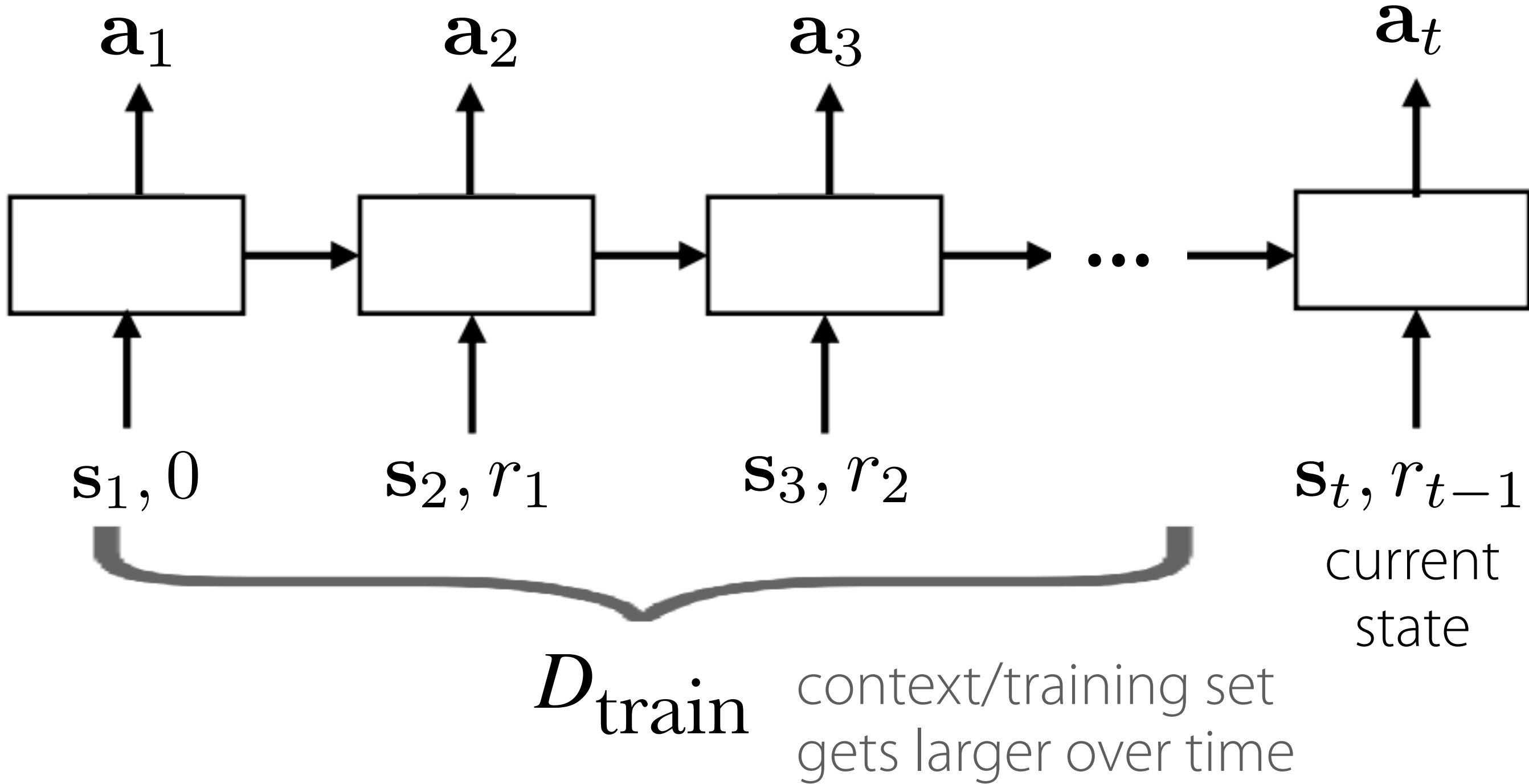
# Plan for Today

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

# Black-Box Meta-RL: Overview

**Black-box neural net**  
(Transformer, NN with memory)

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

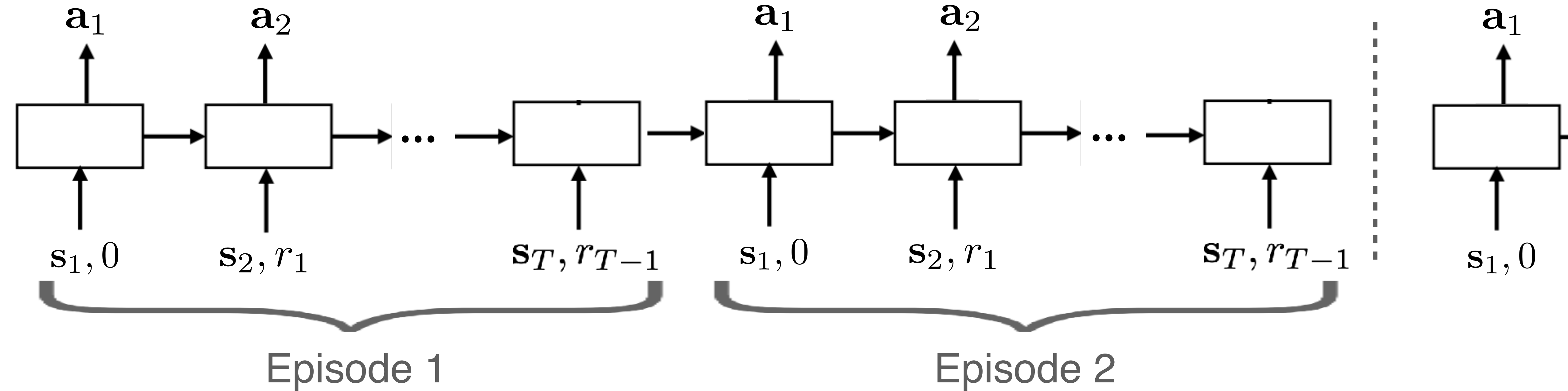


**Question:** How is this different from simply doing RL with a recurrent policy?

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

Hidden state maintained  
**across episodes** within a task!

# Black-Box Meta-RL: Algorithm



1. Sample task  $\mathcal{T}_i$

2. Roll-out policy  $\pi(a|s, \mathcal{D}_i^{\text{tr}})$  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  $p_i(s'|s, a)$  and reward  $r_i(s, a)$ )

# Black-Box Meta-RL: Algorithm

## Meta-Training

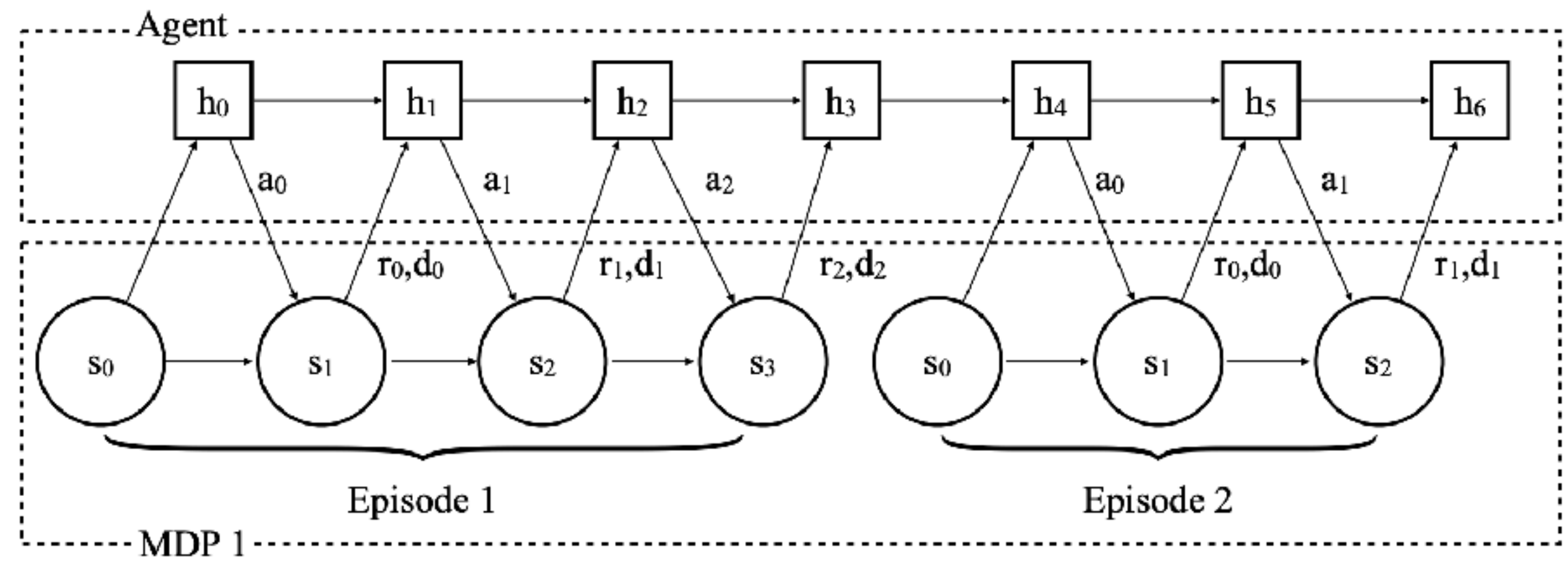
1. Sample task  $\mathcal{T}_i$
  2. Roll-out policy  $\pi(a | s, \mathcal{D}_i^{\text{tr}})$  for  $N$  episodes (under dynamics  $p_i(s' | s, a)$  and reward  $r_i(s, a)$ )
  3. Store sequence in replay buffer for task  $\mathcal{T}_i$ .
  4. Update policy to maximize discounted return for all tasks.
- 

## Meta-Test Time

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

# Black-Box Meta-RL: Architectures & Optimizers

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

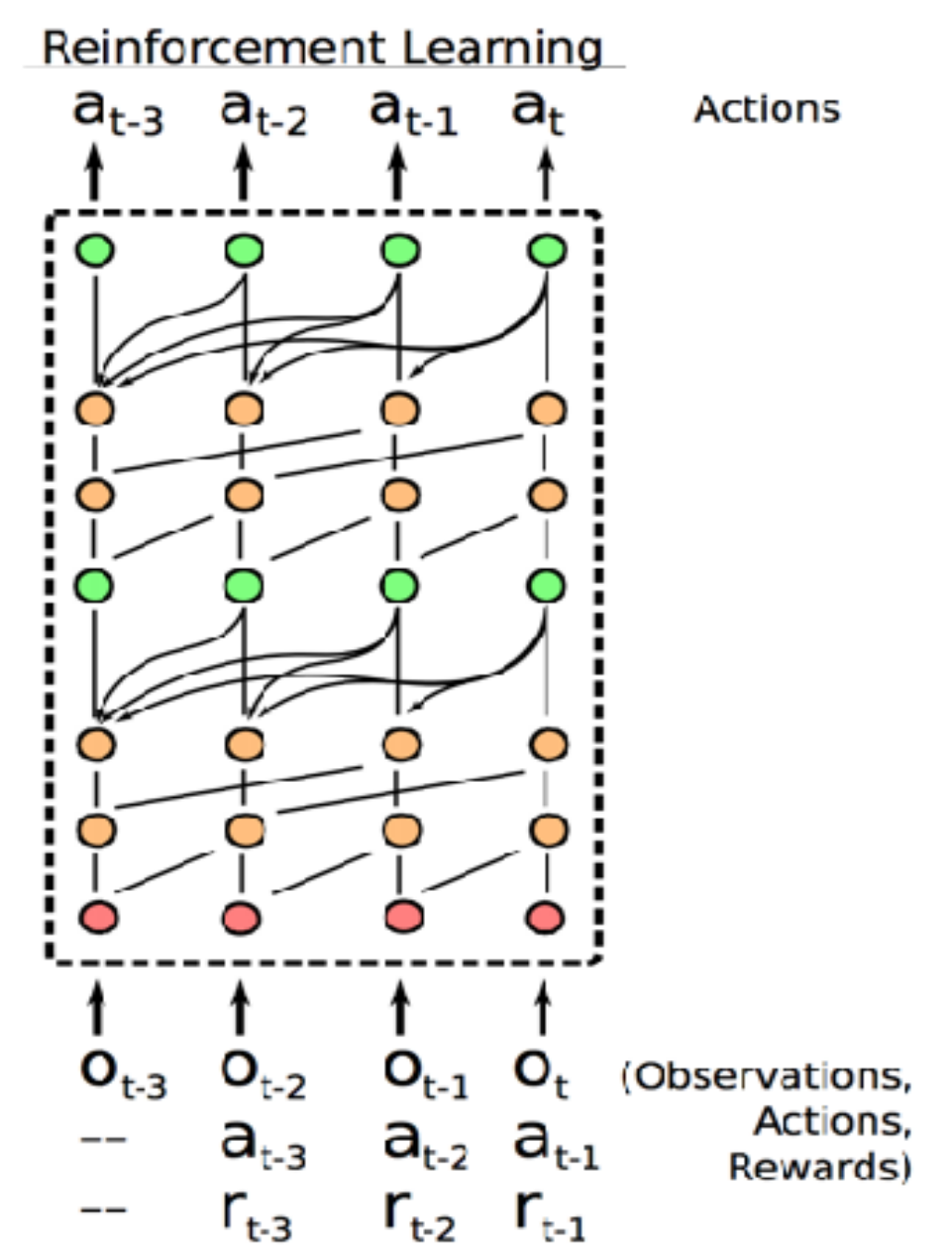


Duan, Schulman, Chen, Bartlett, Sutskever, Abbeel. *RL<sup>2</sup>: Fast Reinforcement Learning via Slow Reinforcement Learning*. 2017

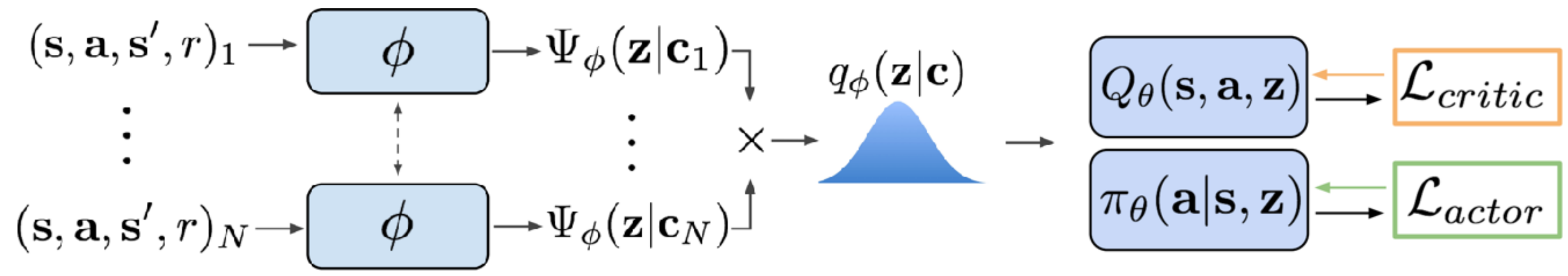
Wang, Kurth-Nelson, Tirumala, Soyer, Leibo, Munos, Blundell, Kumaran, Botvinick. *Learning to Reinforcement Learn*. CogSci 2017

## 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)



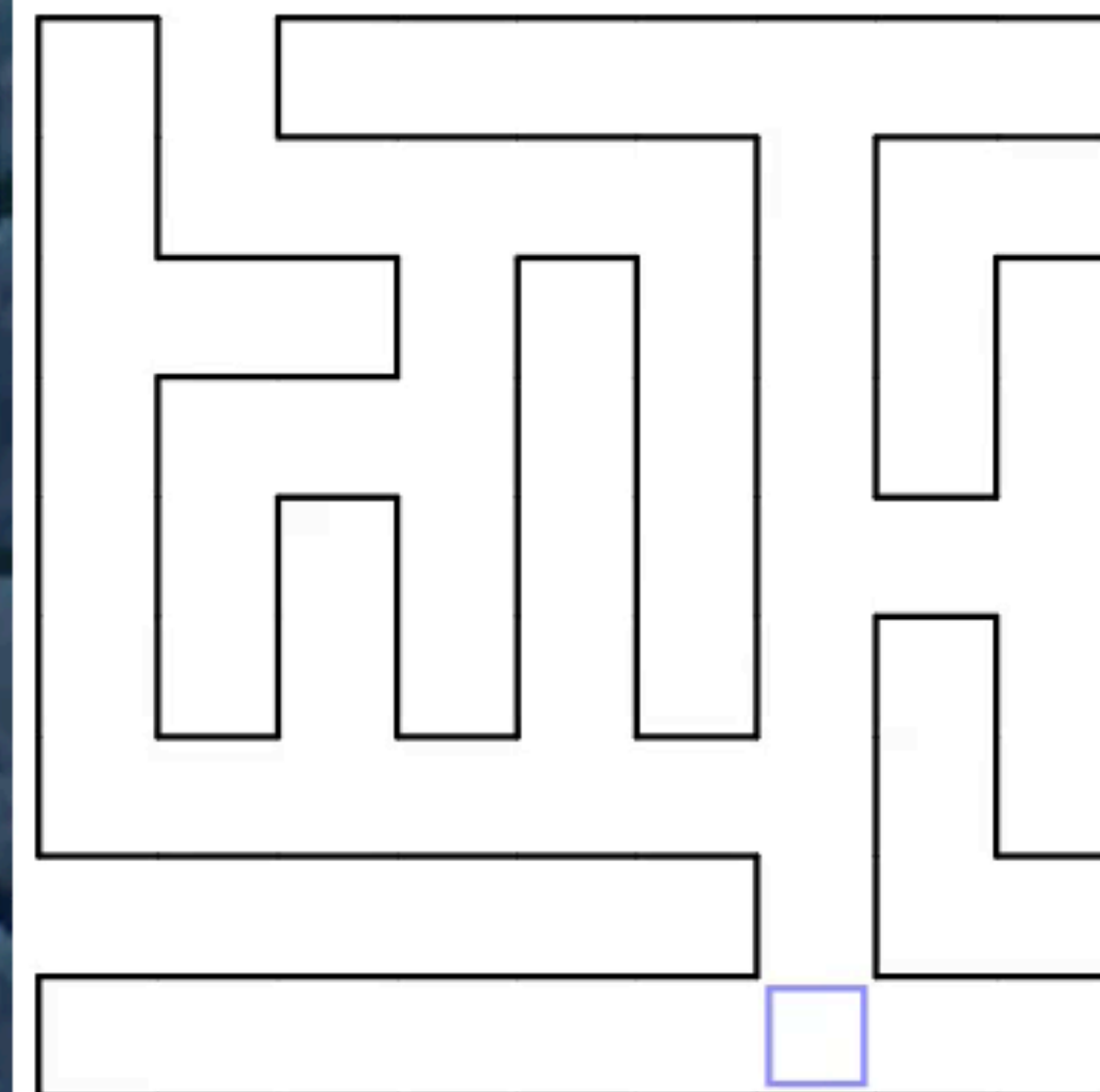
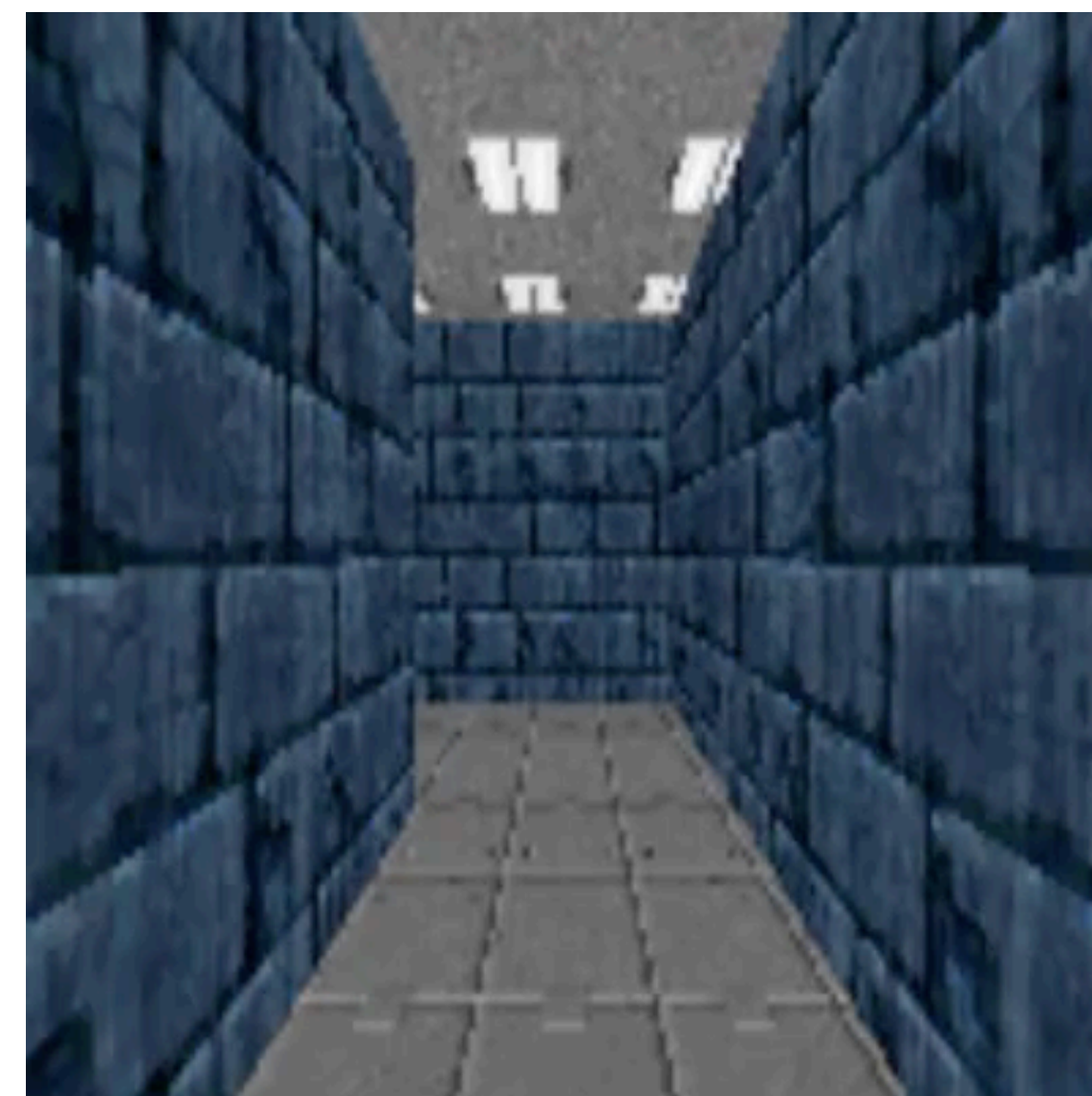
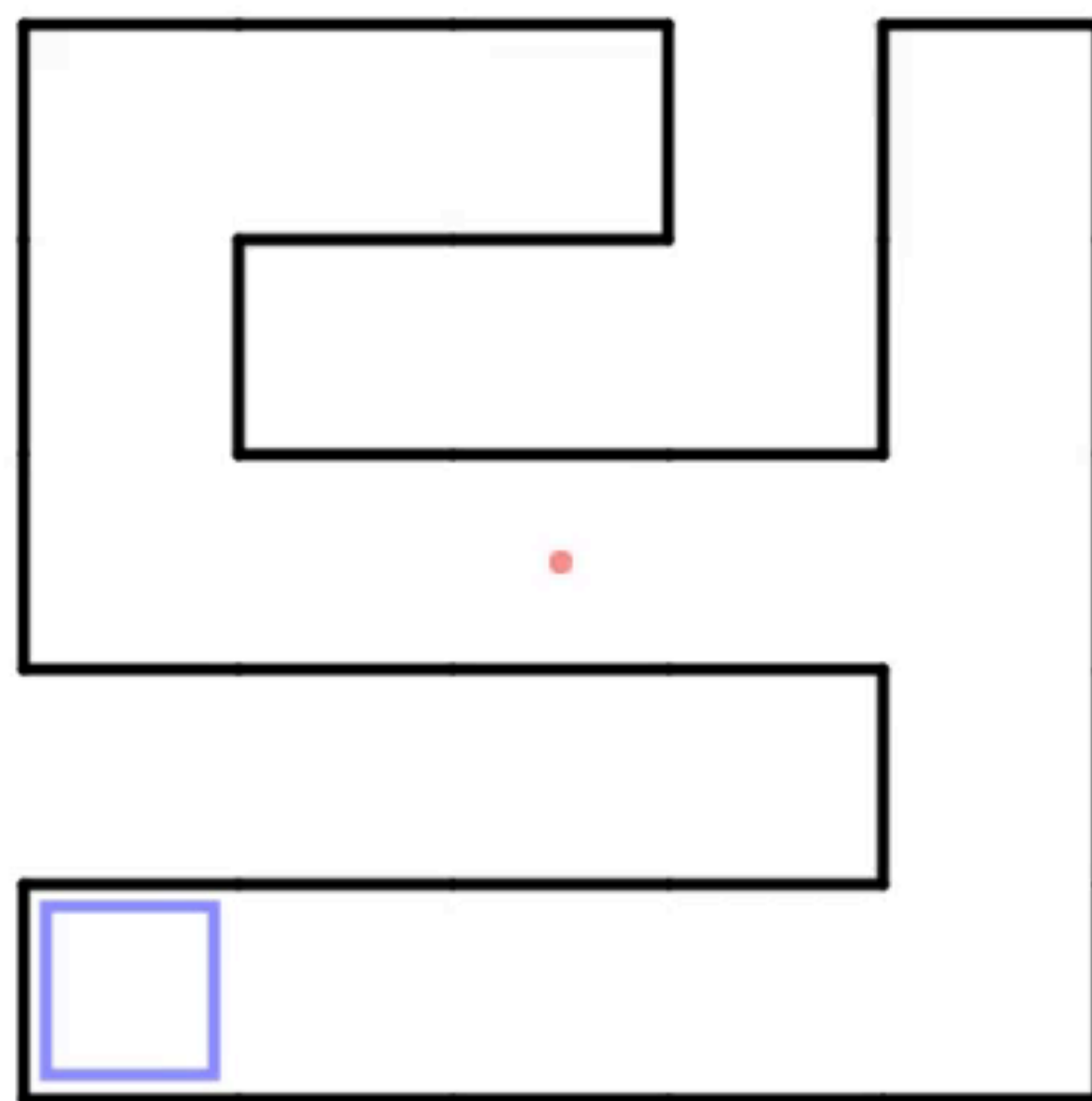
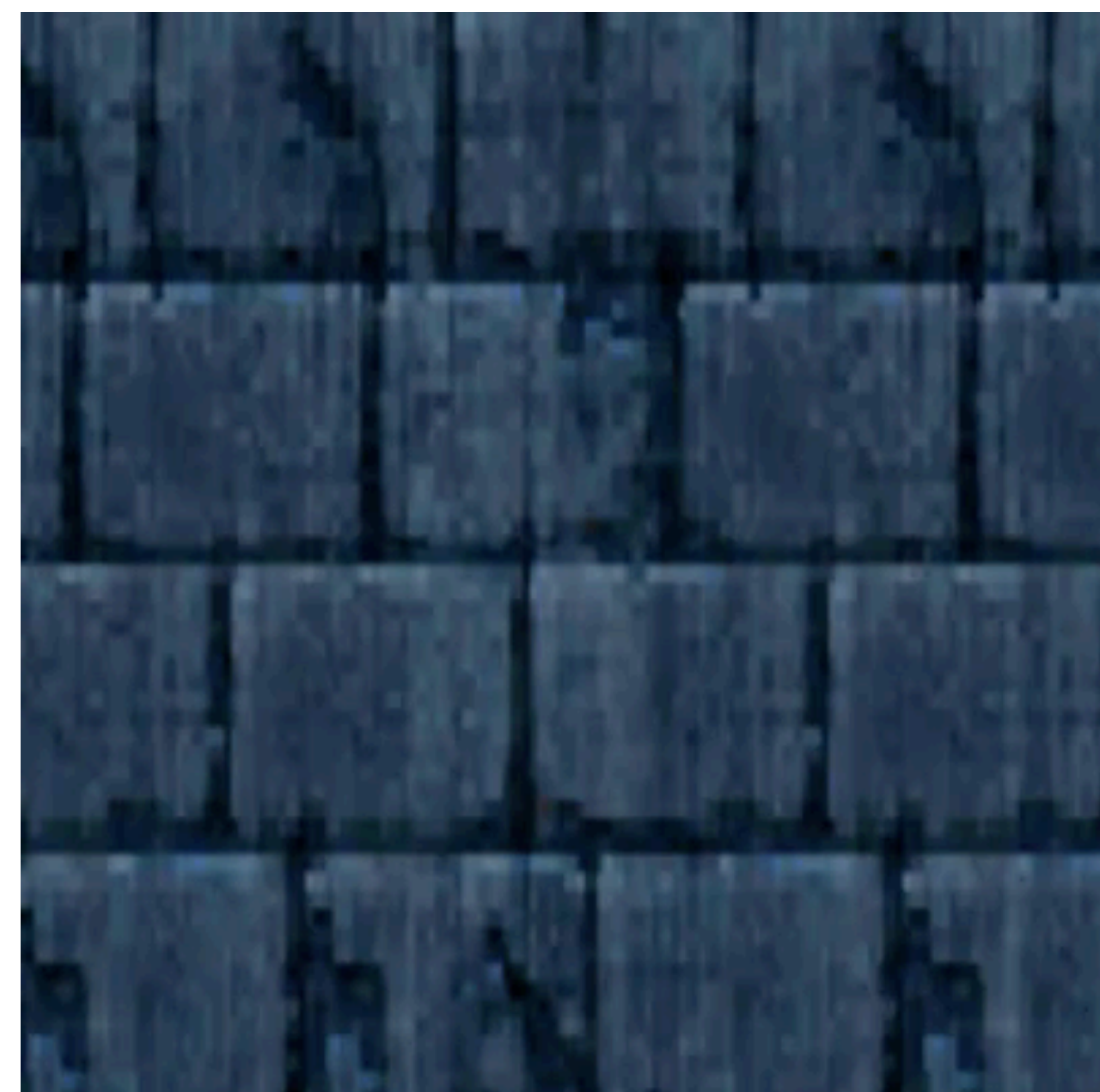
Rakelly, Zhou, Quillen, Finn, Levine. *Efficient Off-Policy Meta-Reinforcement Learning via Probabilistic Context Variables*. ICML 2019.

# 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	$188.6 \pm 3.5$	$187.7 \pm 3.5$	$420.2 \pm 1.2$	$420.8 \pm 1.2$
LSTM	$52.4 \pm 1.3$	$39.1 \pm 0.9$	$180.1 \pm 6.0$	$150.6 \pm 5.9$
<b>SNAIL (ours)</b>	<b><math>50.3 \pm 0.3</math></b>	<b><math>34.8 \pm 0.2</math></b>	<b><math>140.5 \pm 4.2</math></b>	<b><math>105.9 \pm 2.4</math></b>

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

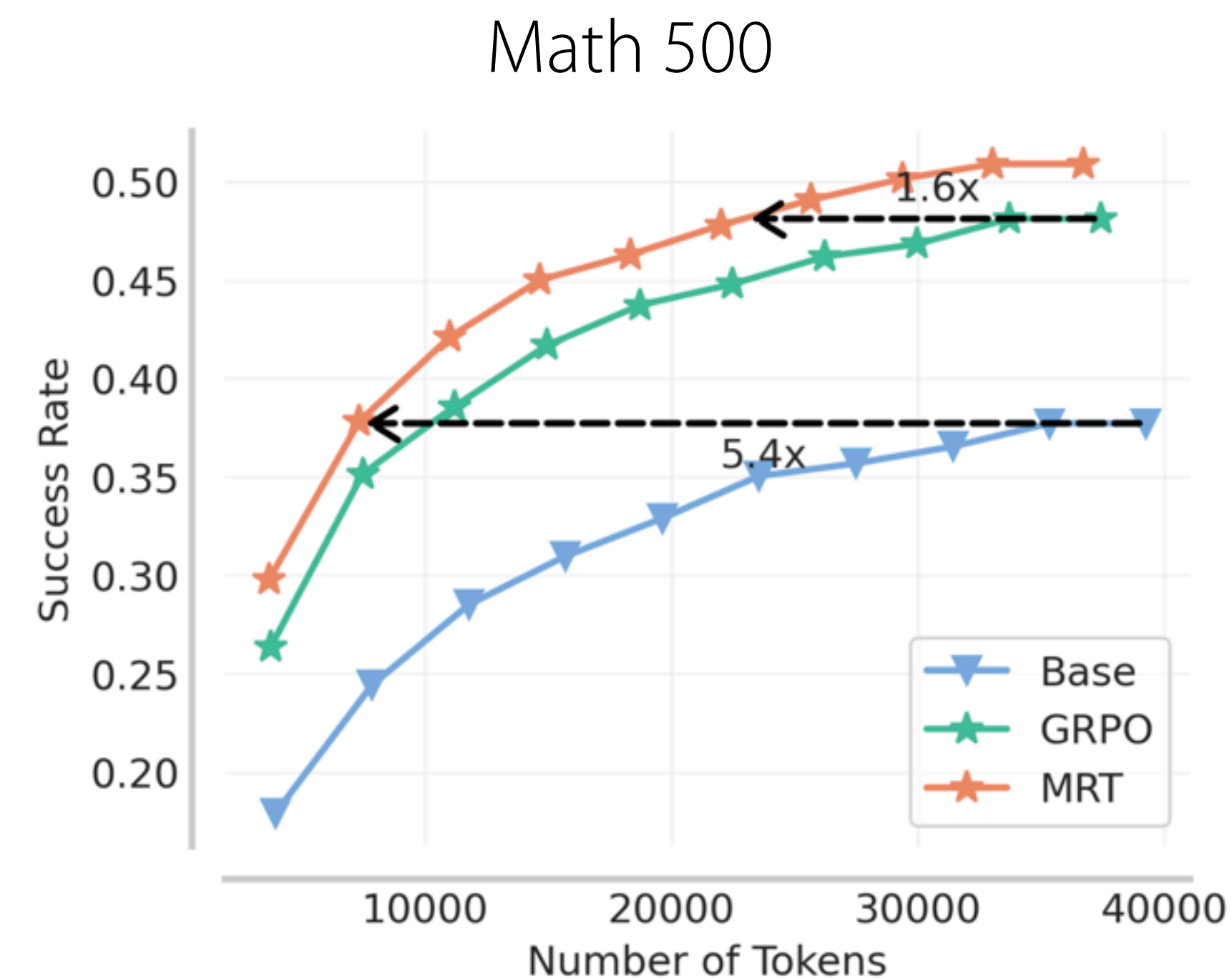
<think>  
 Okay, so I need to show that for any real numbers ...  
 Alternatively, perhaps there's a way to pair terms or use symmetry?  
 Alternatively, could we relate this inequality to some function property?  
 ...  
 Time is up  
 </think>  
 Step-by-Step Explanation:  
 ...

### 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**

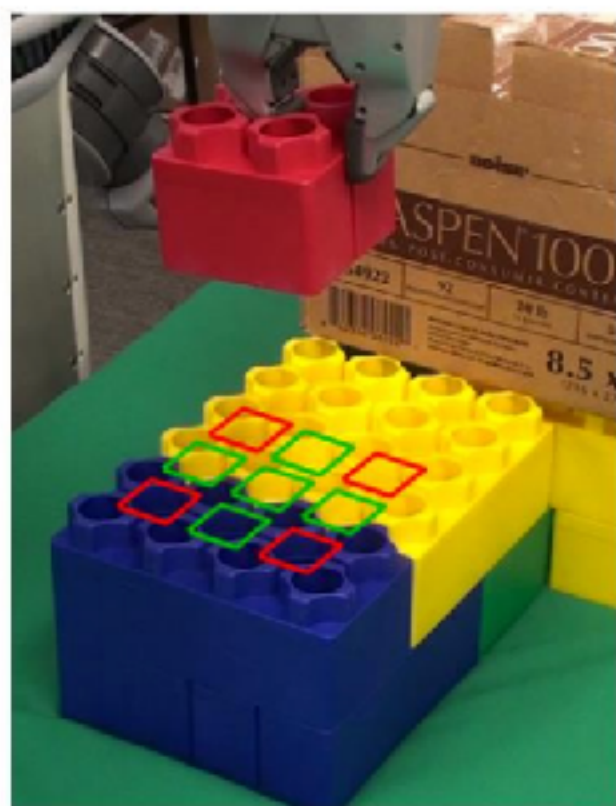
**Key idea:** Can we optimize test-time compute?



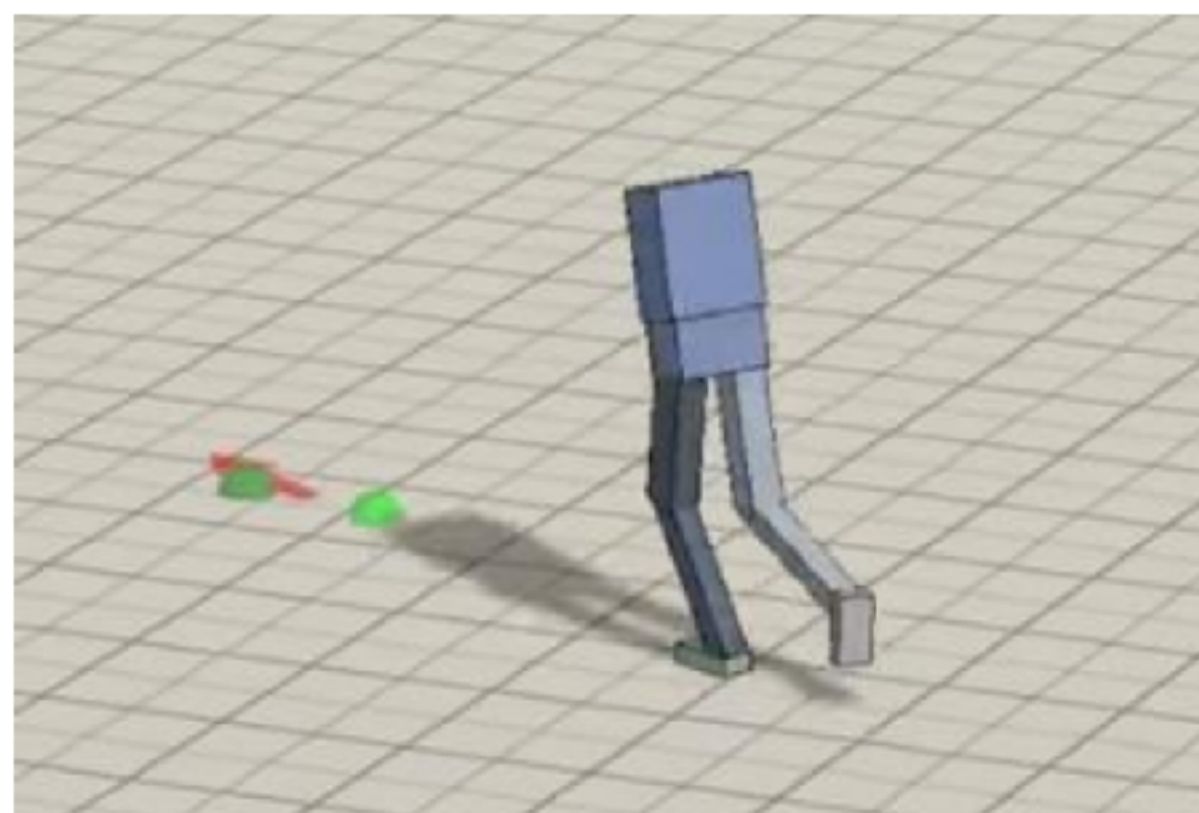
Higher performance with same # of tokens  
 Same accuracy with 1.6x fewer tokens

# Digression: Connection to Multi-Task Policies

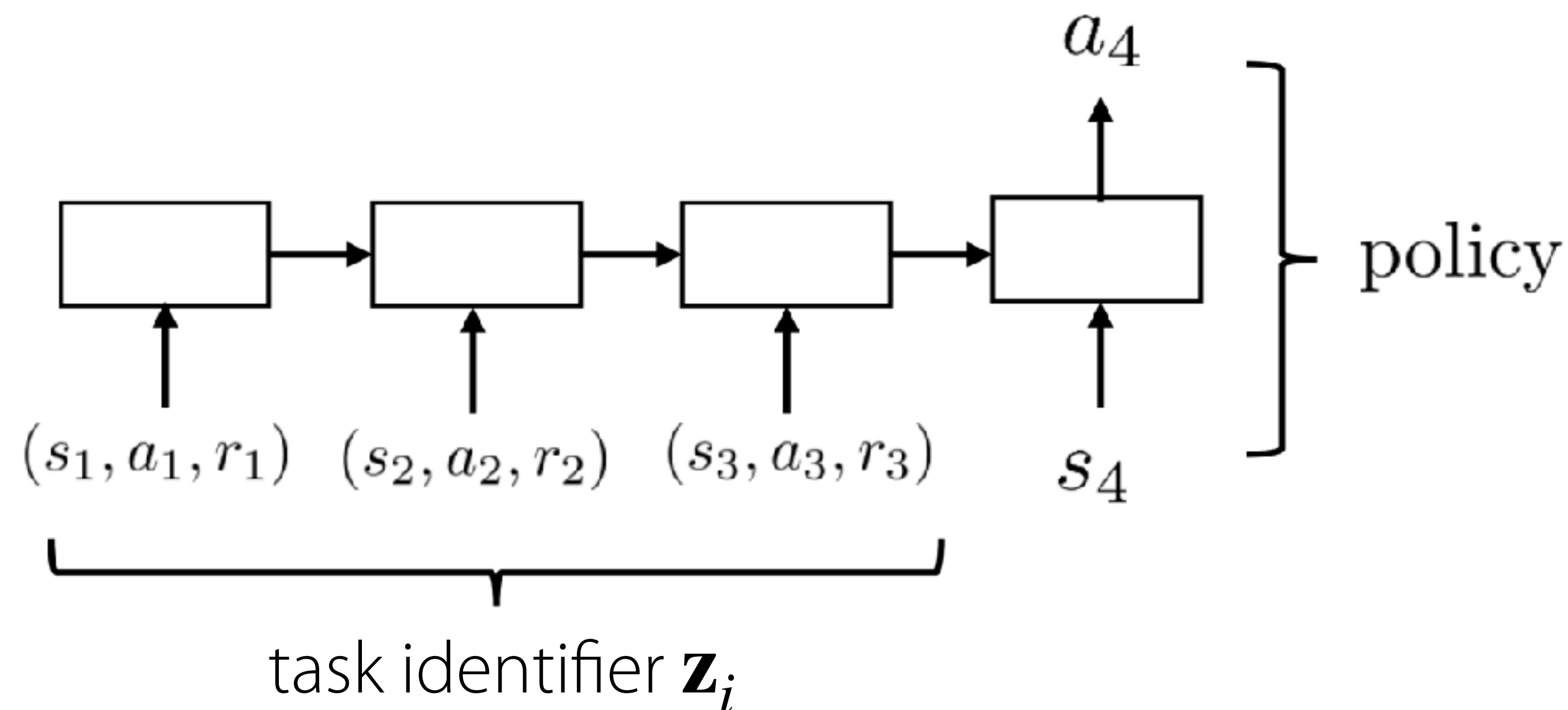
multi-task policy:  $\pi_{\theta}(\mathbf{a} \mid \mathbf{s}, \mathbf{z}_i)$



$\mathbf{z}_i$ : stack location



$\mathbf{z}_i$ : walking direction

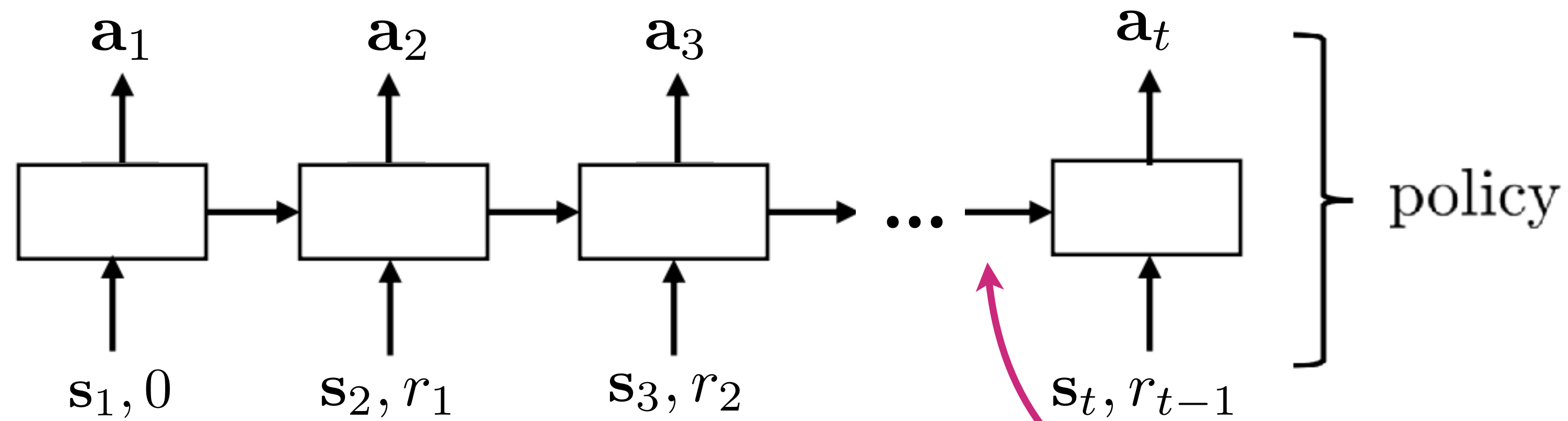


Multi-task policy with experience as task identifier.

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

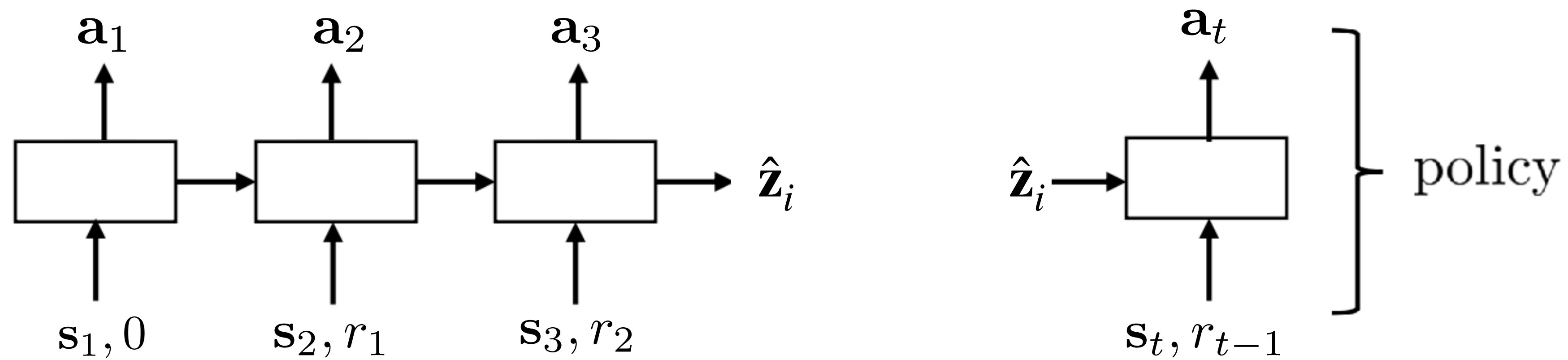


What information is passed here?

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.

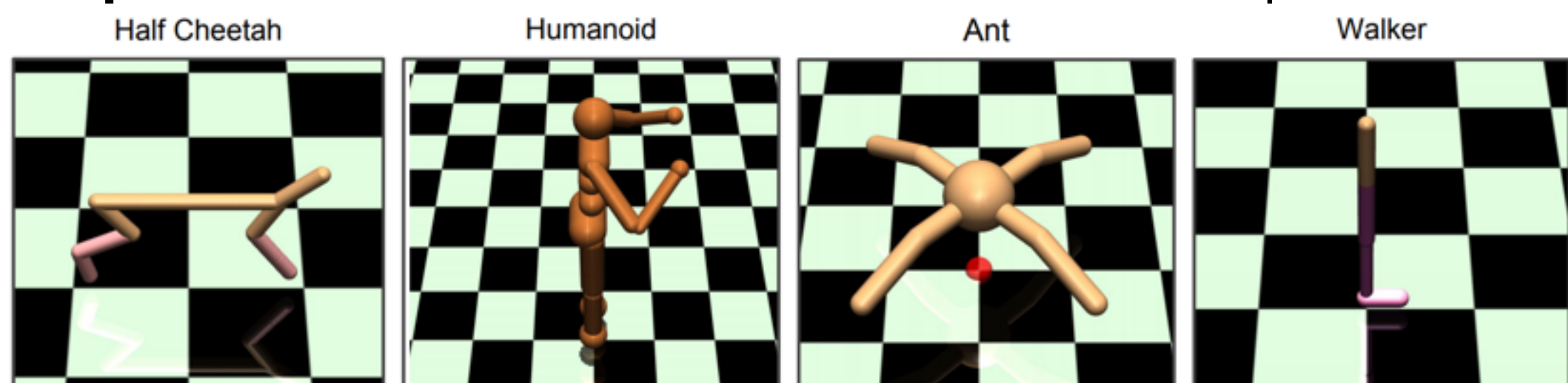
Meta-RL as exploring to infer the unknown task, and then performing the task.

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

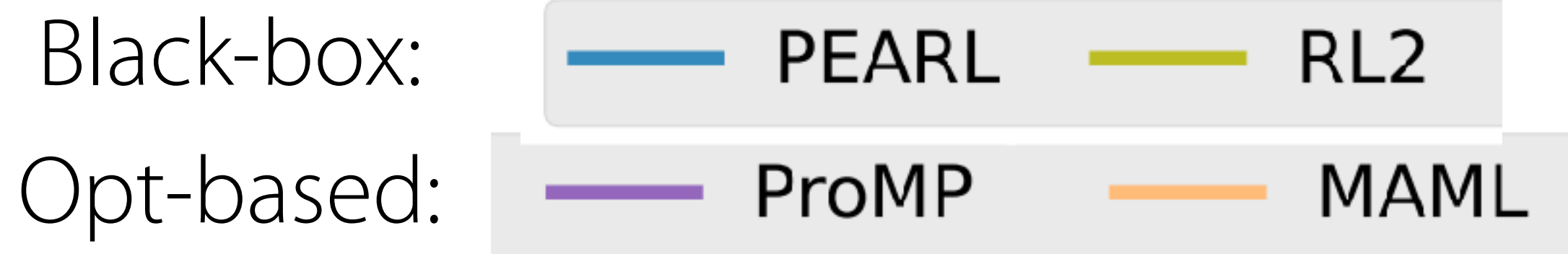
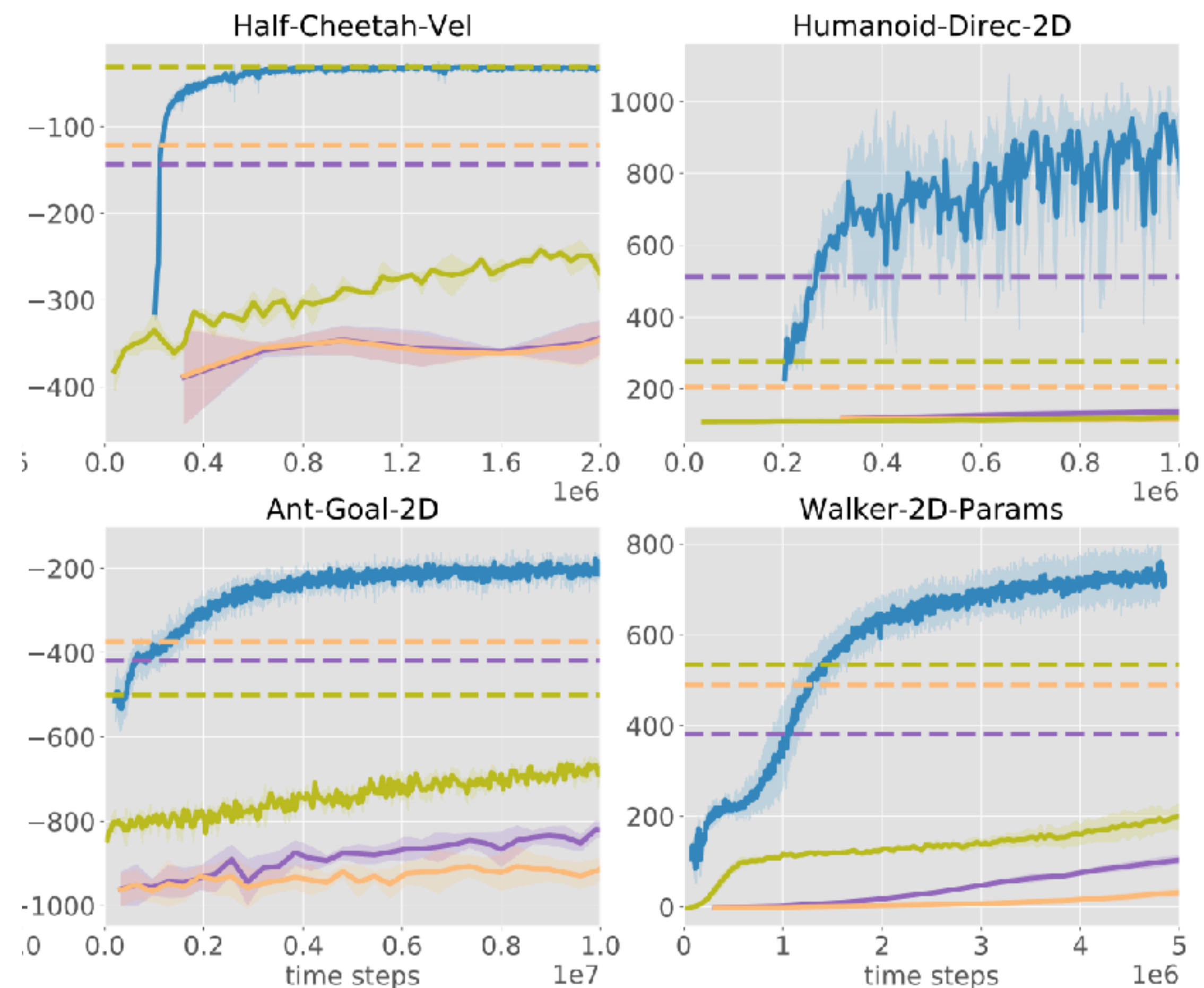


- different directions, velocities
- different physical dynamics

Meta-RL algos are very efficient at new tasks.

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

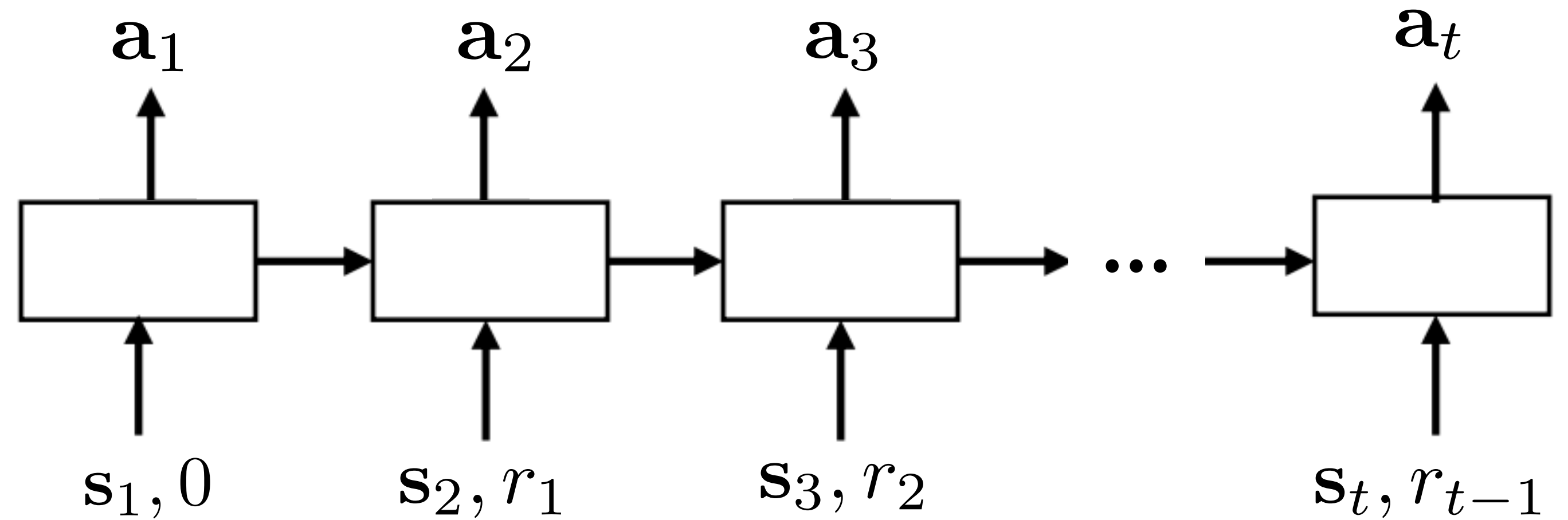
**Question:** Do you expect off-policy meta-RL to be more or less efficient than on-policy meta-RL?



# 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

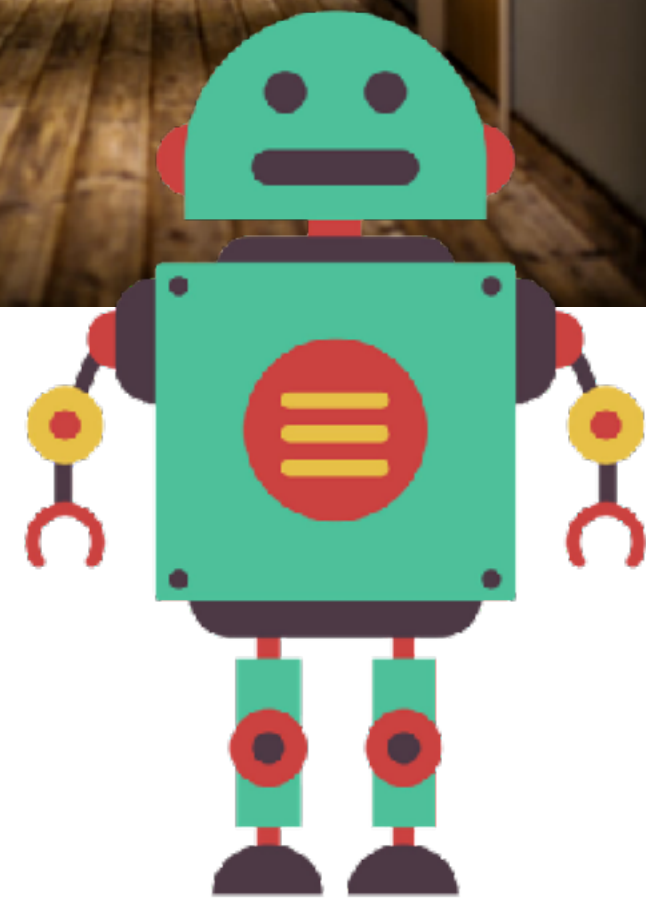


Hallway 2



...

Hallway N



agent



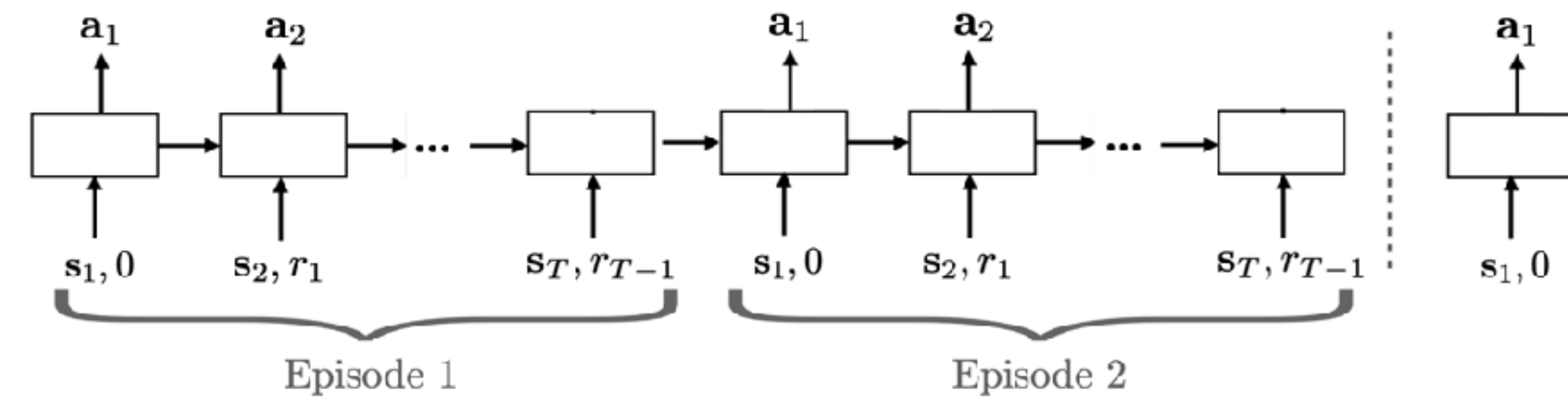
information on  
where to go

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

agent goes to wrong hallway then correct hallway

agent looks at the instructions

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

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

- 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

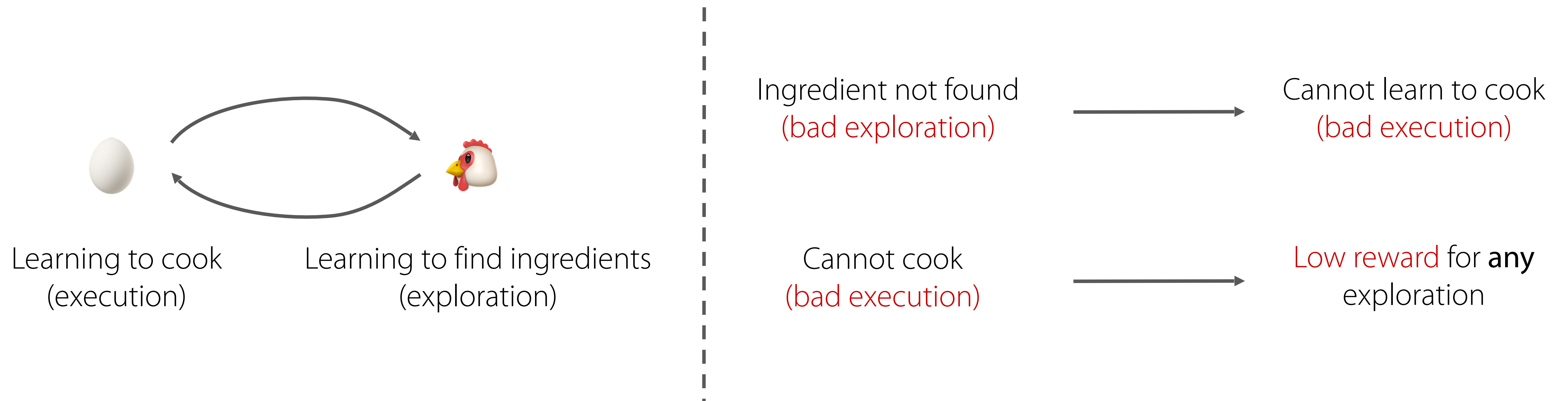
**Goal:** Quickly learn tasks in a new kitchen.



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



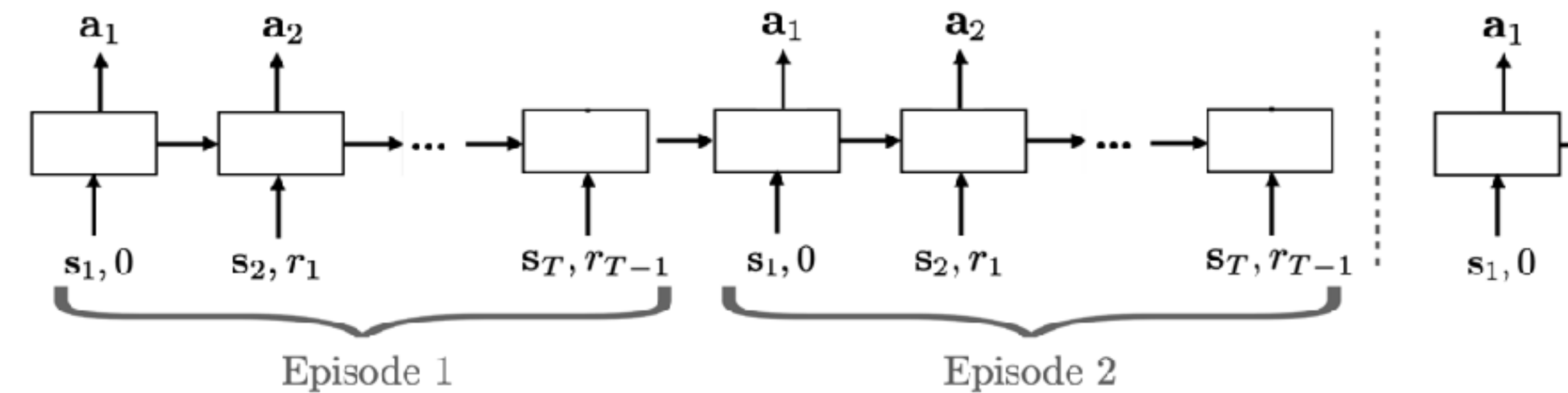
**Coupling problem:** learning exploration and execution depend on each other

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

# 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

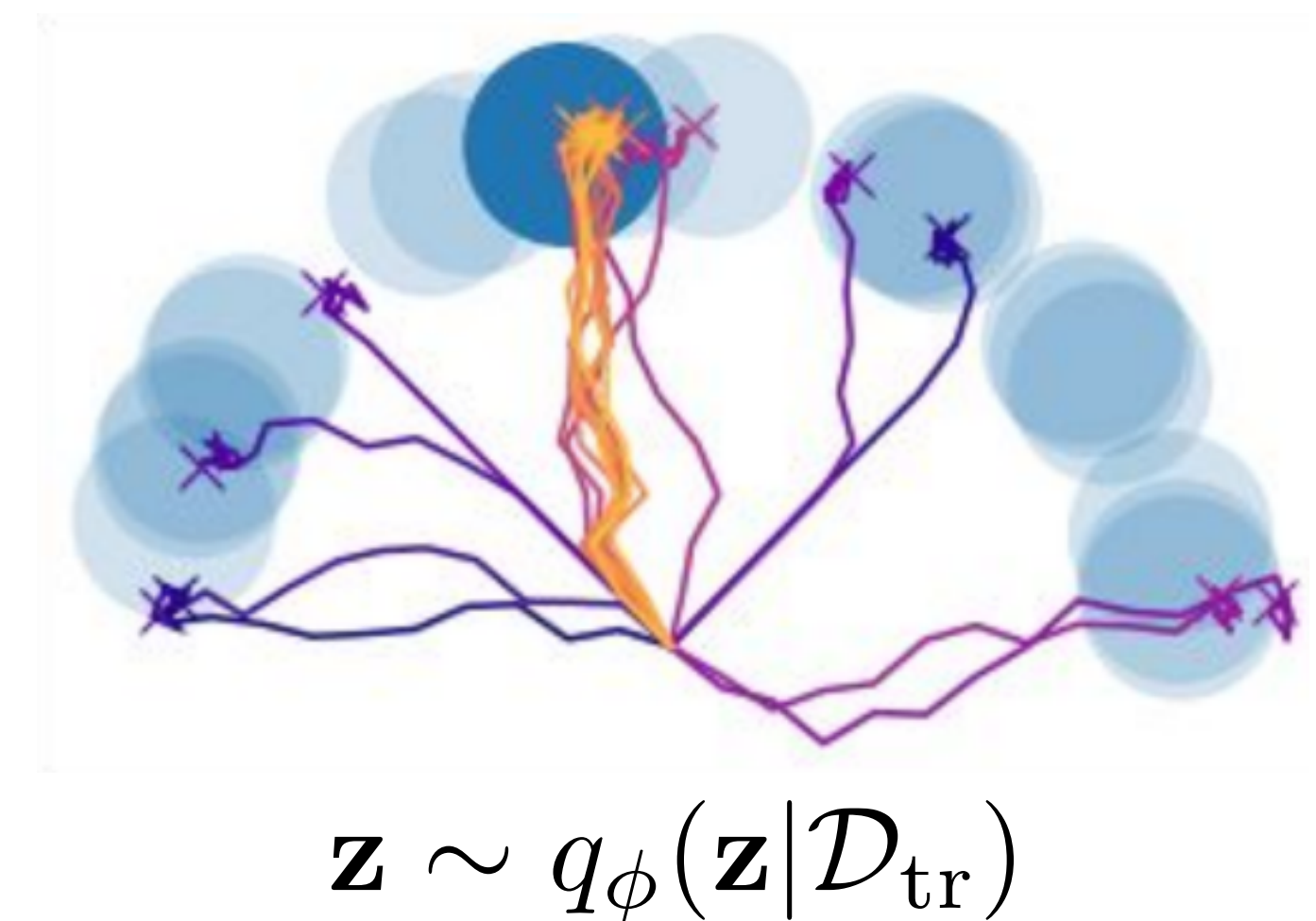
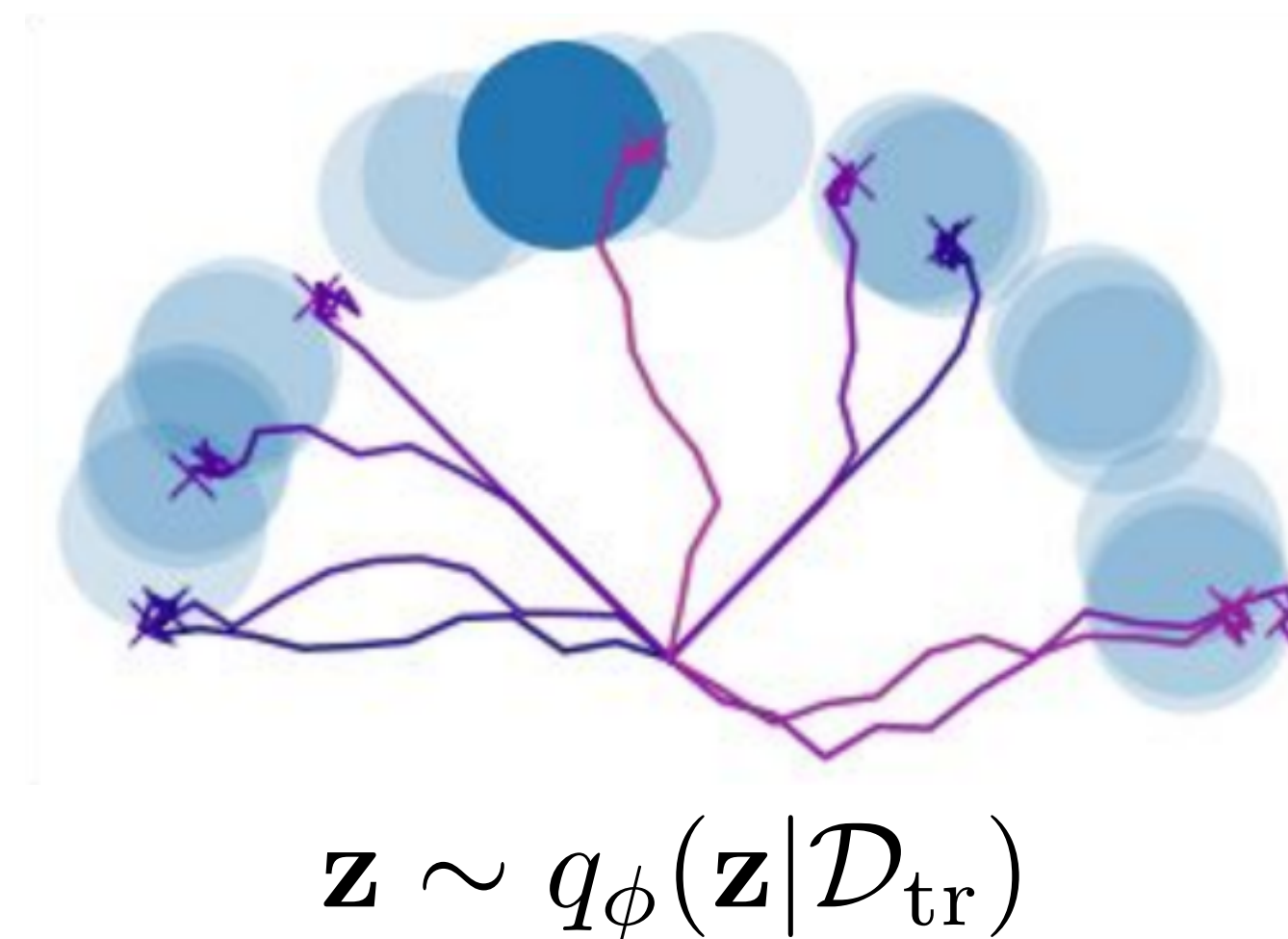
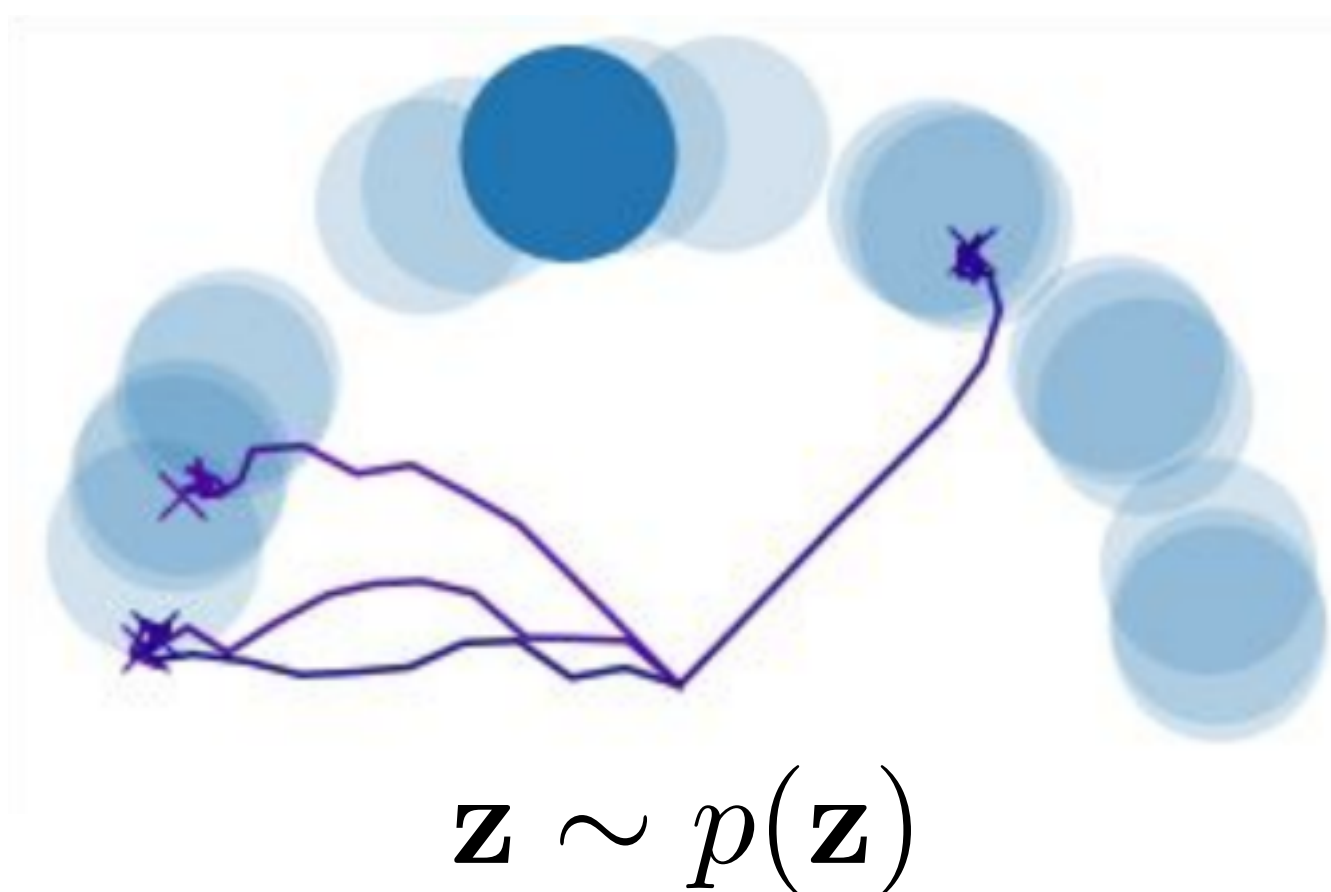
- challenging optimization when exploration is hard

# Approaches to Exploration in Meta-RL

2a. Use posterior sampling  
(also called Thompson sampling)

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

- i. Learn distribution over latent task variable  $p(\mathbf{z}), q(\mathbf{z} | \mathcal{D}_{\text{tr}})$  and corresponding task policies  $\pi(\mathbf{a} | \mathbf{s}, \mathbf{z})$
- ii. Sample  $\mathbf{z}$  from current *posterior* and sample from policy  $\pi(\mathbf{a} | \mathbf{s}, \mathbf{z})$



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

# Approaches to Exploration in Meta-RL

2a. Use posterior sampling  
(also called Thompson sampling)

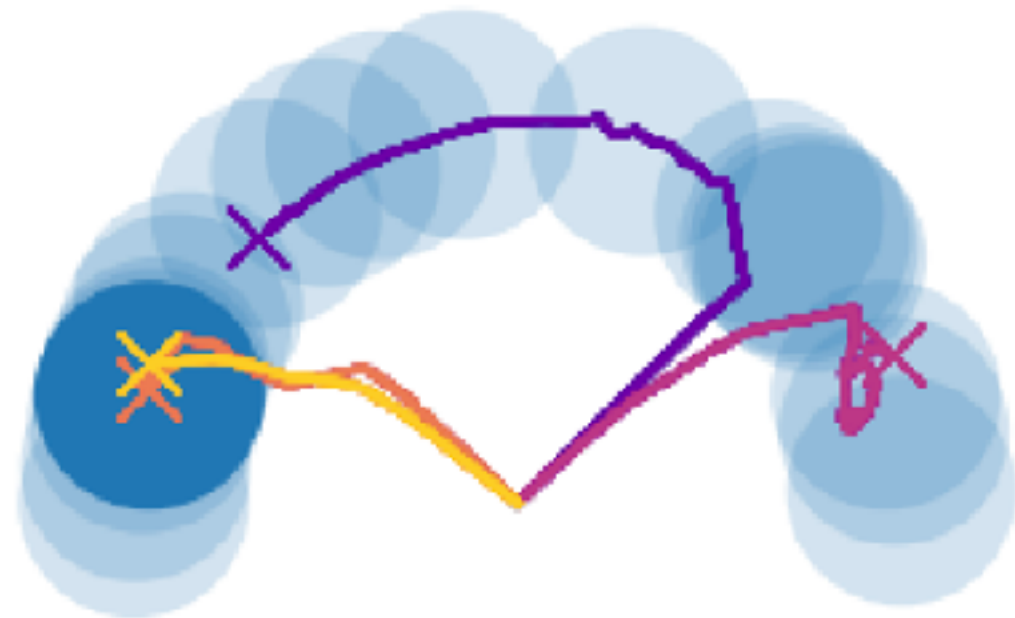
PEARL (Rakelly, Zhou, Quillen, Finn, Levine. ICML '19)

- i. Learn distribution over latent task variable  $p(\mathbf{z}), q(\mathbf{z} | \mathcal{D}_{\text{tr}})$  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. Task dynamics & reward prediction

MetaCURE (Zhang, Wang, Hu, Chen, Fan, Zhang. '20)

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



When might this be bad?

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

# Approaches to Exploration in Meta-RL

2a. Use posterior sampling  
(also called Thompson sampling)

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

- i. Learn distribution over latent task variable  $p(\mathbf{z}), q(\mathbf{z} | \mathcal{D}_{\text{tr}})$  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. Task dynamics & reward prediction

MetaCURE (Zhang, Wang, Hu, Chen, Fan, Zhang. '20)

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

+ easy to optimize  
+ many based on  
principled strategies

-- suboptimal by arbitrarily large  
amount in some environments.

# Approaches to Exploration in Meta-RL

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

- i. Learn distribution over latent task variable  $p(\mathbf{z}), q(\mathbf{z} | \mathcal{D}_{tr})$  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. Task dynamics & reward prediction MetaCURE (Zhang, Wang, Hu, Chen, Fan, Zhang. '20)

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

2c. Predict compressed task representation DREAM (Liu, Raghunathan, Liang, Finn. '21)

- i. Train model  $f(\mathbf{z}_{comp} | \mathcal{D}_{train})$
- ii. Collect  $\mathcal{D}_{train}$  so that task prediction is accurate.

+ leads to optimal strategy in principle  
+ easy to optimize in practice

-- requires task identifier

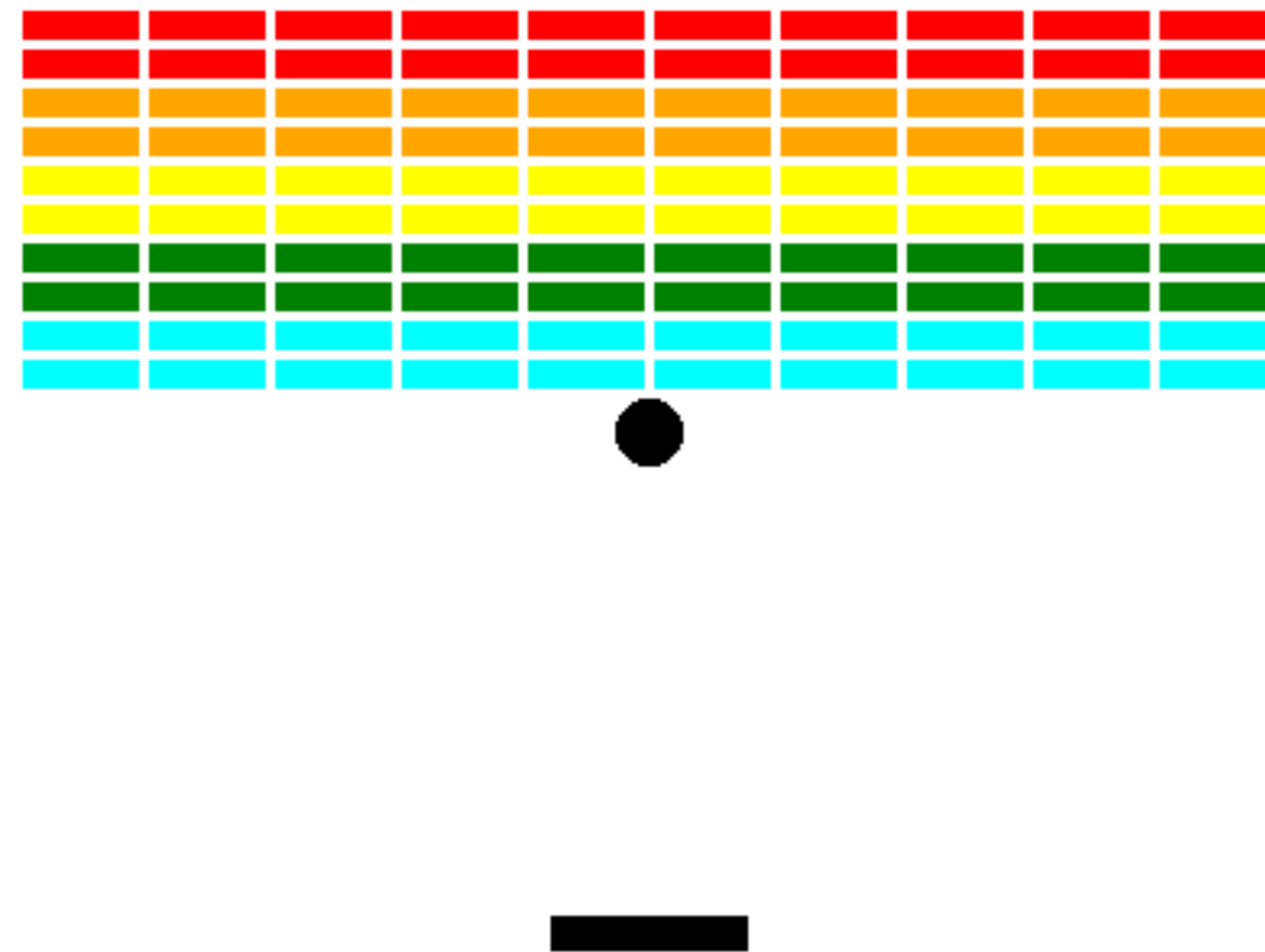
Time Permitting: Applying Meta-RL to CS Education

# Example application: finding bugs & providing feedback in student programs

Bounce programming assignment  
(Code.org)

```
Underlying env ID: 7340
Env ID: 1
Label: [1 1 0 0 0 0 1 0 0 1 0 1 0 1 1]
Binary label: whenGoal-noBallLaunch
Action: None
Reward: 0
Timestep: 0
Exploration reward: 0.020
Prob: 0.456
```

Breakout assignment  
(CS106A)



Time-consuming for instructors/TAs to give feedback, grades.  
*Use meta-RL to learn exploration!*

# Experiments: Learned Exploration Behavior on Bounce

```
Underlying env ID: 7340
Env ID: 1
Label: [1 1 0 0 0 0 1 0 0 1 0 1 0 1 1]
Binary label: whenGoal-noBallLaunch
Action: None
Reward: 0
Timestep: 0
Exploration reward: 0.020
Prob: 0.456
```

```
Underlying env ID: 4843
Env ID: 0
Label: [0 1 0 0 0 0 0 0 0 0 0 0 0 1 1]
Binary label: whenBall-noBallLaunch
Action: None
Reward: 0
Timestep: 0
Exploration reward: 0.005
Prob: 0.507
```

```
Underlying env ID: 2932
Env ID: 1
Label: [0 1 0 1 0 0 0 0 0 0 0 1 0 0 1]
Binary label: whenBall-illegal-moveRight
Action: None
Reward: 0
Timestep: 0
Exploration reward: 0.070
Prob: 0.221
```

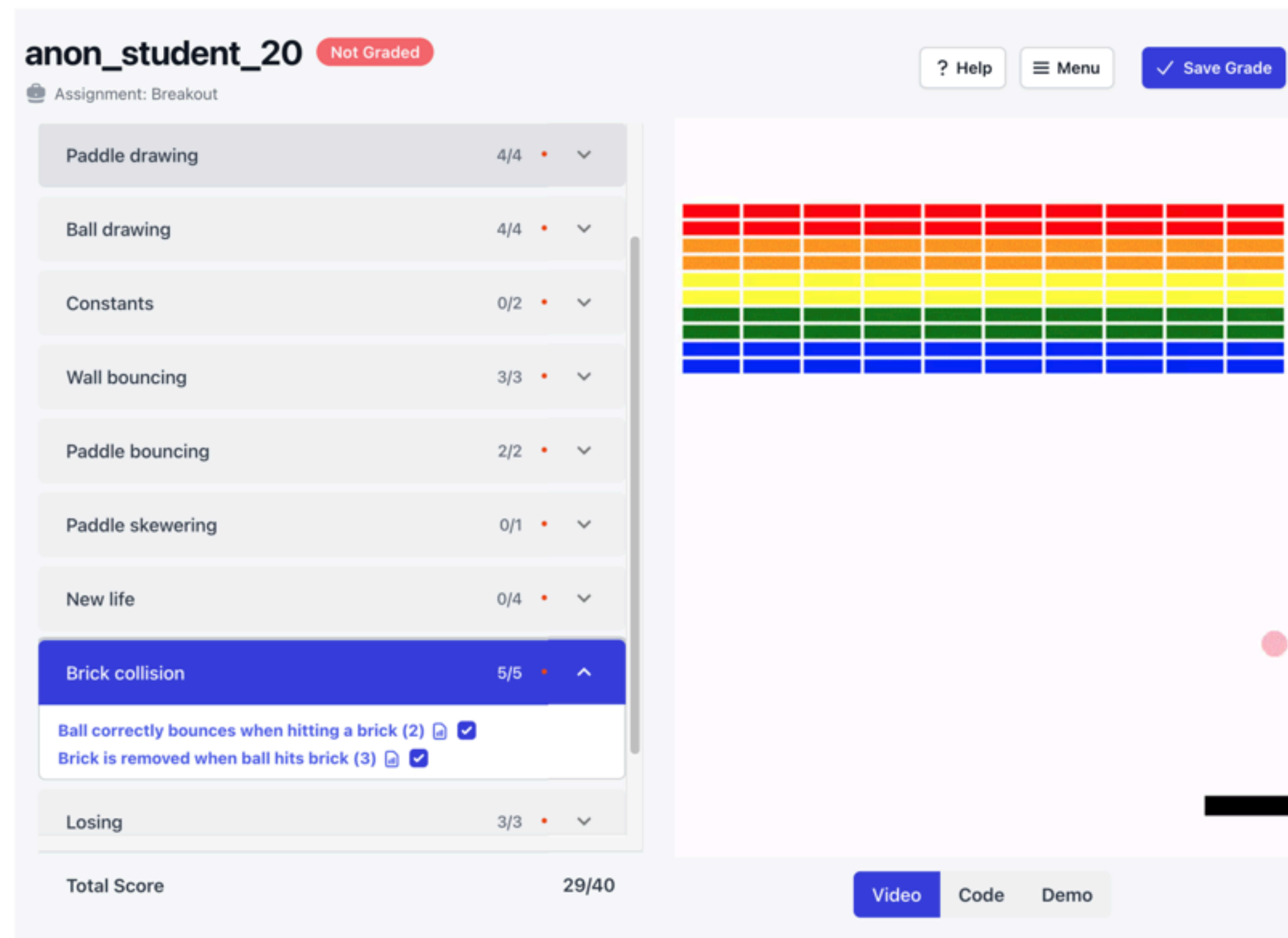
What happens when...

the ball hits the goal?

the ball hits the floor?

the ball hits the wall?

# Experiments: AI-Assisted Grading in CS106A (Spring 2023)



Autograder prepopulates rubric & shows videos.

Leads to 44% faster & 6% more accurate grading.

Grading Scheme	Human Grading Time	Grading Accuracy
Manual	8 min 35s ± 6 min 47s	86.4% ± 8.9%
Autograder with human	<b>4 min 49s ± 2 min 5s</b>	<b>92.3% ± 7.6%</b>
Autograder only	—	90.1% ± 11.0%

Stanford TAs like using it.

Likert Scale (Strongly Disagree = 1, Disagree = 2, Neutral = 3, Agree = 4, Strongly Agree = 5)	
Statement	Avg. Score
Using the autograder is easier than manually grading.	4.5
Using the autograder is faster than manually grading.	4.5
Using the autograder is more accurate than manually grading.	3.9
The autograder's grades were useful to me.	4.4
I enjoyed using the autograder.	4.6
Net Promoter Score (0 - 10 inclusive)	
How much would you recommend using the autograder over manually grading in the future?	9.0

# Plan for Today

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

**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  
**Next Weds:** hierarchy

Reminders

Midterm on **Friday**  
Project milestone due **next Friday**