Exploration & Meta-Exploration

CS224R

Reminders

Homework 3 due **tonight**

(and HW4 out today)

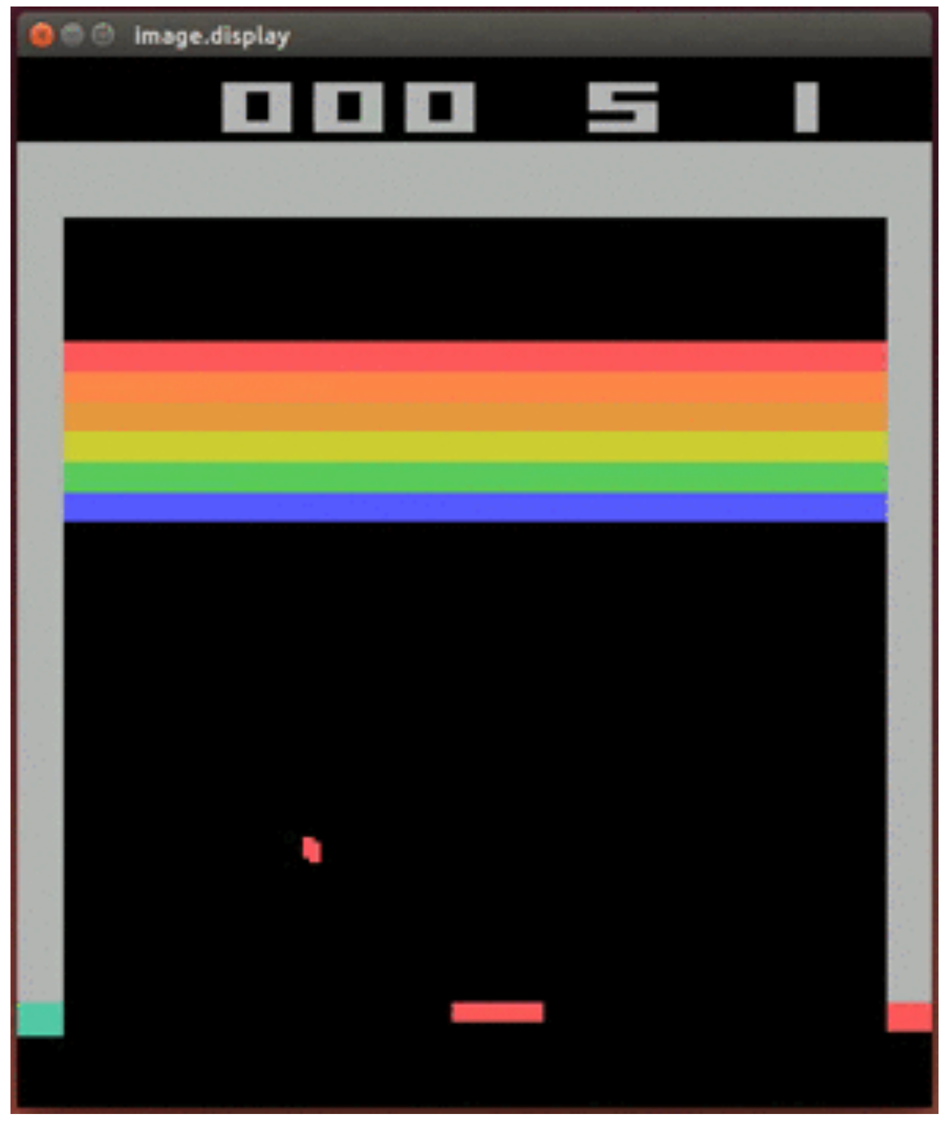
Project milestone due **next Friday**

The plan for today

1. **Exploration**
   a. Why is exploration hard?
   b. Algorithms for exploration in bandits
   c. Exploration in robotics, LLMs
2. **Learning to Explore via Meta-Learning**
   a. Efficiently Meta-Learning Optimal Exploration <- Part of HW 4
   b. Meta-Exploration in for CS Education

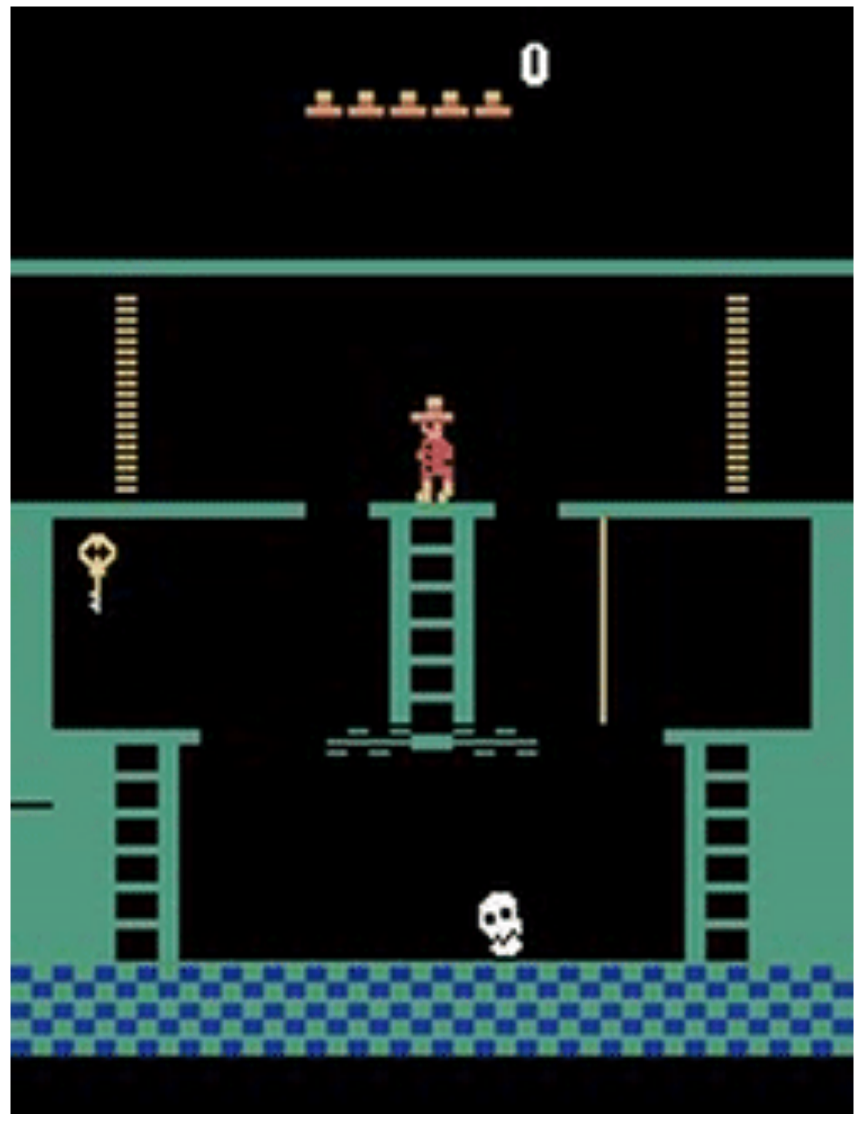
What's the problem?

this is easy (mostly)



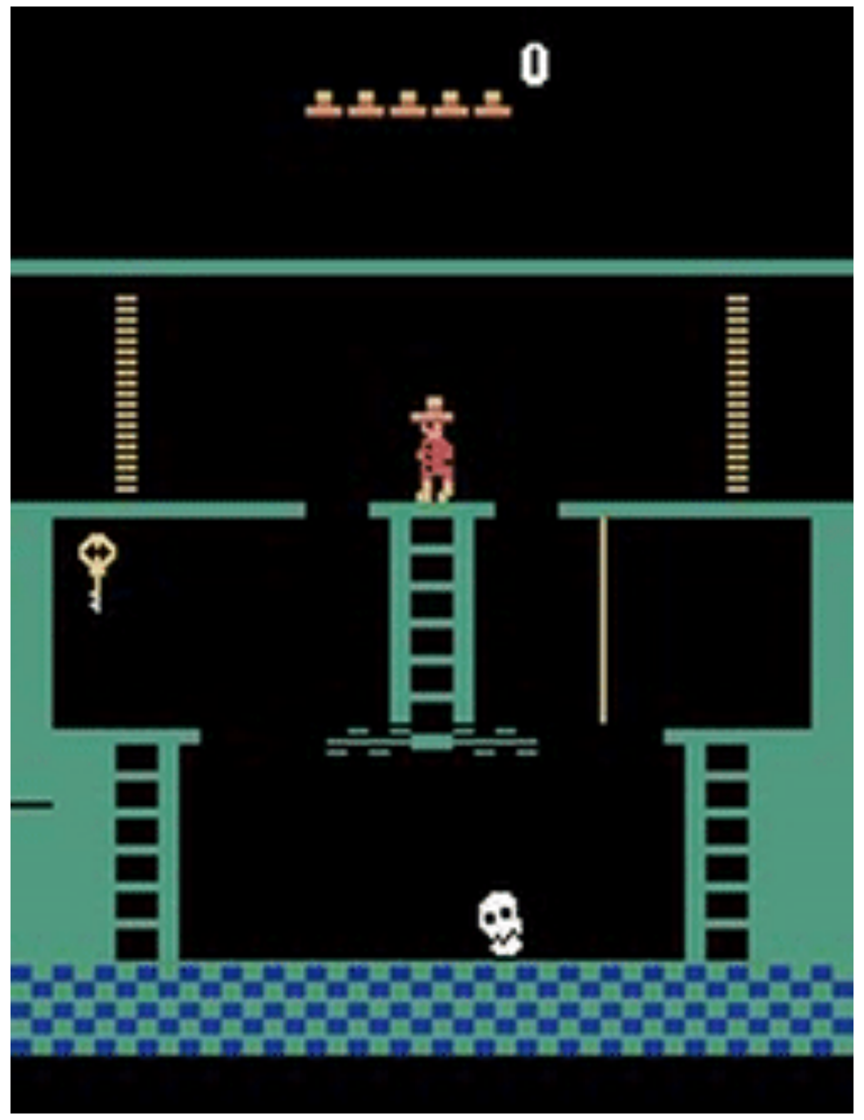
Why?

this is impossible



Slide adapted from Sergey Levine

Montezuma's revenge



- Getting key = reward
- Opening door = reward
- Getting killed by skull = nothing (is it good? bad?)
- Finishing the game only weakly correlates with rewarding events
- We know what to do because we **understand** what these sprites mean!

Put yourself in the algorithm's shoes

Mao

- "the only rule you may be told is this one"
- Incur a penalty when you break a rule
- Can only discover rules through trial and error
- Rules don't always make sense to you

- Temporally extended tasks like Montezuma's revenge become increasingly difficult based on
  - How long the task is
  - How little you know about the rules
- Imagine if your goal in life was to win 50 games of Mao…
- (and you didn't know this in advance)

Exploration and exploitation

- Two potential definitions of exploration problem
    - How can an agent discover high-reward strategies that require a temporally extended sequence of complex behaviors that, individually, are not rewarding?
    - How can an agent decide whether to attempt new behaviors (to discover ones with higher reward) or continue to do the best thing it knows so far?
- Actually the same problem:
    - Exploitation: doing what you know will yield highest reward
    - Exploration: doing things you haven't done before, in the hopes of getting even higher reward

Exploration and exploitation examples

- Restaurant selection
  - Exploitation: go to your favorite restaurant
  - Exploration: try a new restaurant
- Online ad placement
  - Exploitation: show the most successful advertisement
  - Exploration: show a different random advertisement
- Oil drilling
  - Exploitation: drill at the best known location
  - Exploration: drill at a new location

Examples from D. Silver lecture notes: http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching_files/XX.pdf

Exploration is hard

Can we derive an **optimal** exploration strategy?

what does optimal even mean?

Let’s talk about these for a minute.

multi-armed bandits (1-step stateless RL problems)

contextual bandits (1-step RL problems)

small, tabular MDPs

large MDPs with known low-rank features

large MDPs without known special structure

theoretically tractable $\longleftrightarrow$ theoretically intractable

Slide adapted from Sergey Levine

How do we measure optimality in bandits?

regret after doing RL for T episodes

$$\underbrace{\mathrm{Reg}(T)}_{} = T\underbrace{E[r(a^\star)]}_{} - \sum_{t=1}^{T} \underbrace{r(a_t)}_{}$$

expected reward of best action
(the best we can hope for in expectation)

reward of action
actually taken

- Variety of relatively simple strategies
- Often can provide theoretical guarantees on regret
  - Variety of optimal algorithms (up to a constant factor)
  - But empirical performance may vary…

Exploration in Bandits

Optimism when uncertain

(e.g. upper confidence bound (UCB))

Track avg reward $\bar{r}_a$ for all actions

Exploit: pick $a_{\text{exploit}} = \arg\max_a \bar{r}_a$

Explore: pick $a_{\text{explore}} = \arg\max_a \bar{r}_a + C\sigma_a$

some kind of variance estimate

Intuition: assume unknown = good

Probability matching

(e.g. posterior sampling)

Assume $r(a_i) \sim p_{\psi_i}(r)$ for each arm a_i

Defines a POMDP with $\mathbf{s} := [\psi_1, \ldots, \psi_n]$

1. Form beliefs $\hat{p}(\psi_1, \ldots, \psi_n)$ from data
2. Sample an MDP $\psi_1, \ldots, \psi_n \sim \hat{p}(\cdot)$
3. Take optimal action a assuming MDP is correct.

Intuition: iteratively improve model of bandit, act according to one model

These methods are widely used/studied in industry for recommenders, ad placement!

Exploration game

Exploration in Robotics and LLMs

Lots of research translating these exploration methods to large MDPs like robot control.

But, at the end of the day, **exploration from scratch in large MDPs is intractable**.

These models are *excellent* at exploration!

Instead:

- rely on demonstrations, or on base models pre-trained on demonstrations
- use shaped rewards wherever feasible

(e.g. hand-coded in simulation for legged robots, single step rewards from preferences for LLMs)

The plan for today

1. **Exploration**
    a. Why is exploration hard?
    b. Algorithms for exploration in bandits
    c. Exploration in robotics, LLMs
2. **Learning to Explore via Meta-Learning**
    a. Efficiently Meta-Learning Optimal Exploration <- Part of HW 4
    b. Meta-Exploration in for CS Education

What if we need to explore at test time?

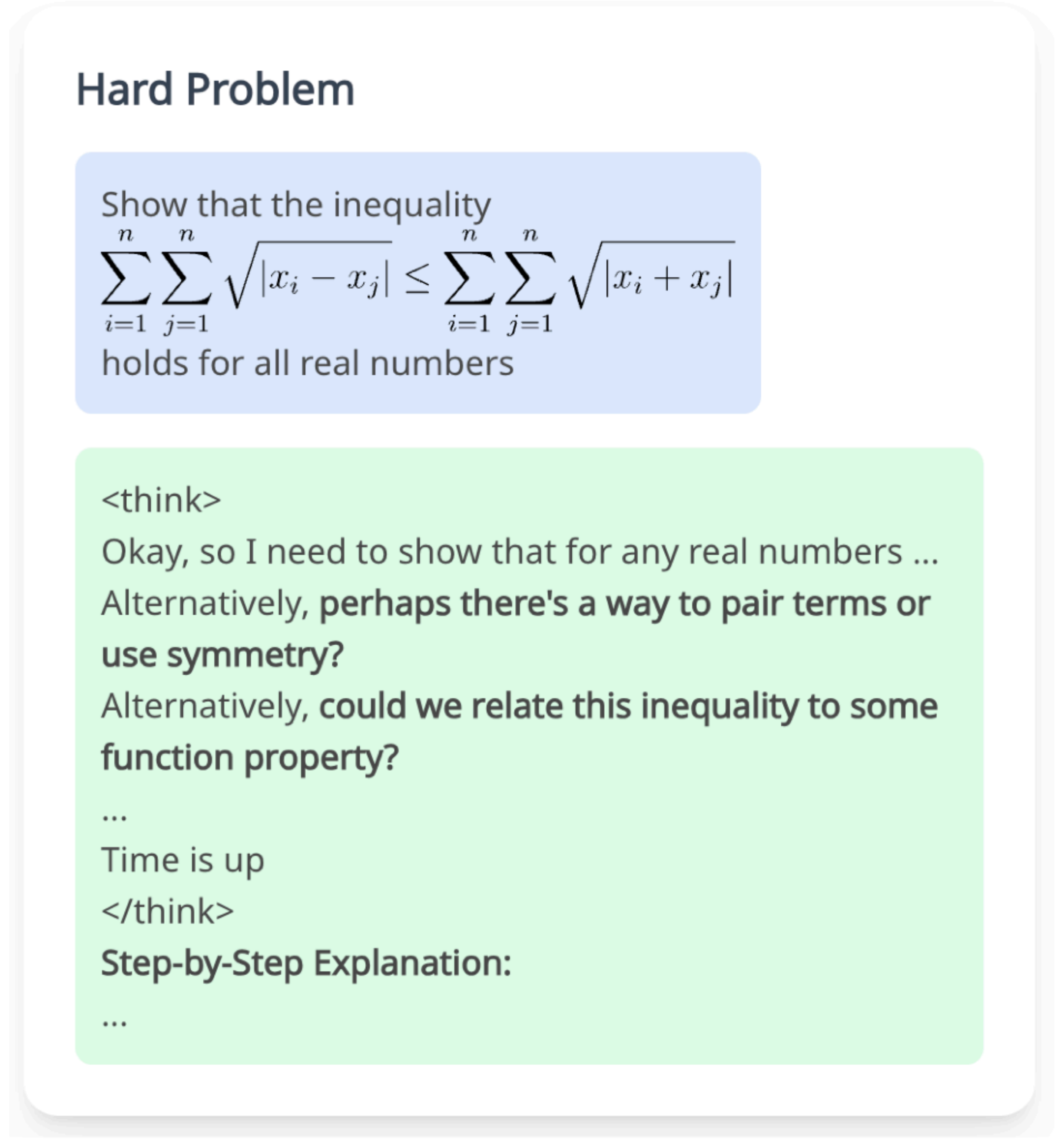
Let's revisit exploration in meta-RL

Partial observability

e.g. finding ingredients in a kitchen

LLM reasoning

e.g. identifying strategies for solving a complex problem



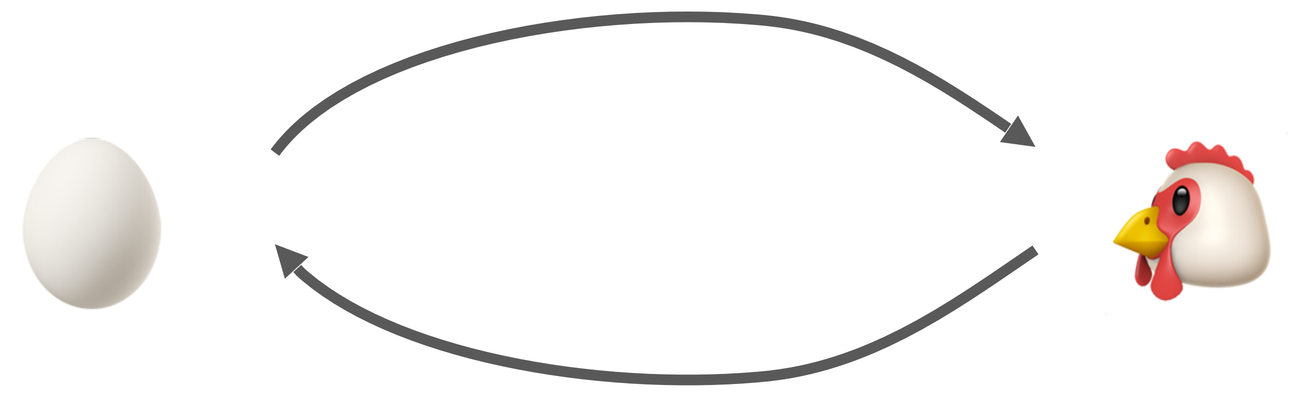
Hard Problem

Show that the inequality
$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sqrt{|x_i - x_j|} \le \sum_{i=1}^{n}\sum_{j=1}^{n}\sqrt{|x_i + x_j|}$$
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:
...

Recall: Why is End-to-End Training Hard for Meta-Learning with Exploration?

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

Alternative: 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 $\pi(\mathbf{a} | \mathbf{s}, \mathbf{z})$

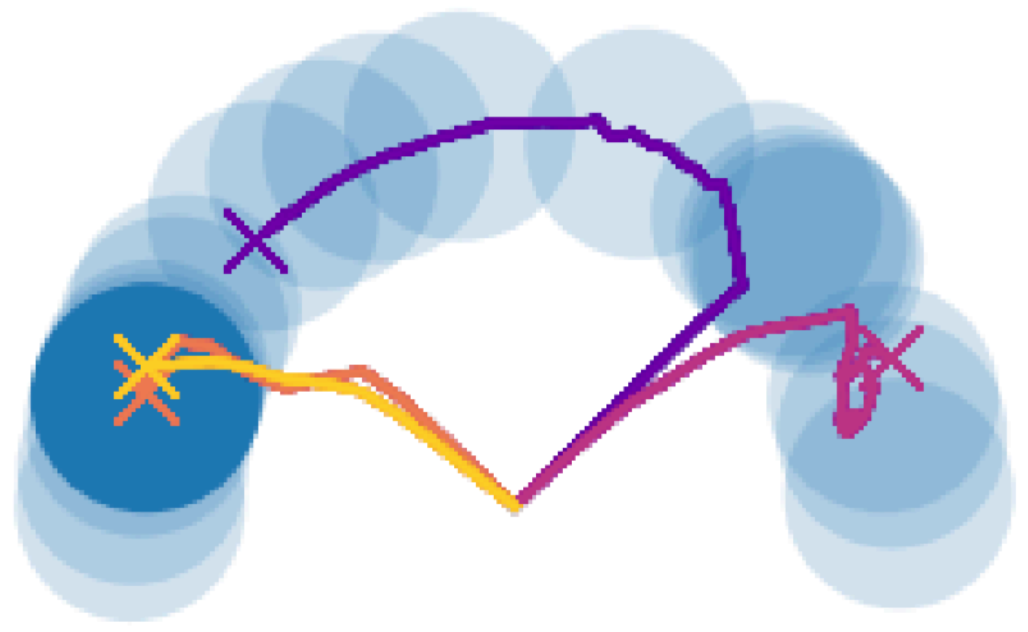
ii. Sample $\mathbf{z}$ from current *posterior* and sample from policy $\pi(\mathbf{a} | \mathbf{s}, \mathbf{z})$

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.



When might this be bad?

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

Can we avoid the chicken-and-egg problem *without* sacrificing optimality?

(best of both worlds?)

Yes!

Solution #3

Idea from solution #2c: Train model $f(\mathbf{s}', r \mid \mathbf{s}, \mathbf{a}, \mathscr{D}_{\text{tr}})$ & collect $\mathscr{D}_{\text{tr}}$ so that model is accurate.

Do we have to learn a *full dynamics & reward model*?

Idea 3.0: Label each training task with a unique ID μ

Meta training

Exploration policy: train policy $\pi^{\text{exp}}(\mathbf{a} \mid \mathbf{s})$ and task identification model $q(\mu \mid \mathscr{D}_{\text{tr}})$

such that $\mathscr{D}_{\text{tr}} \sim \pi^{\text{exp}}$ allows accurate task prediction from f

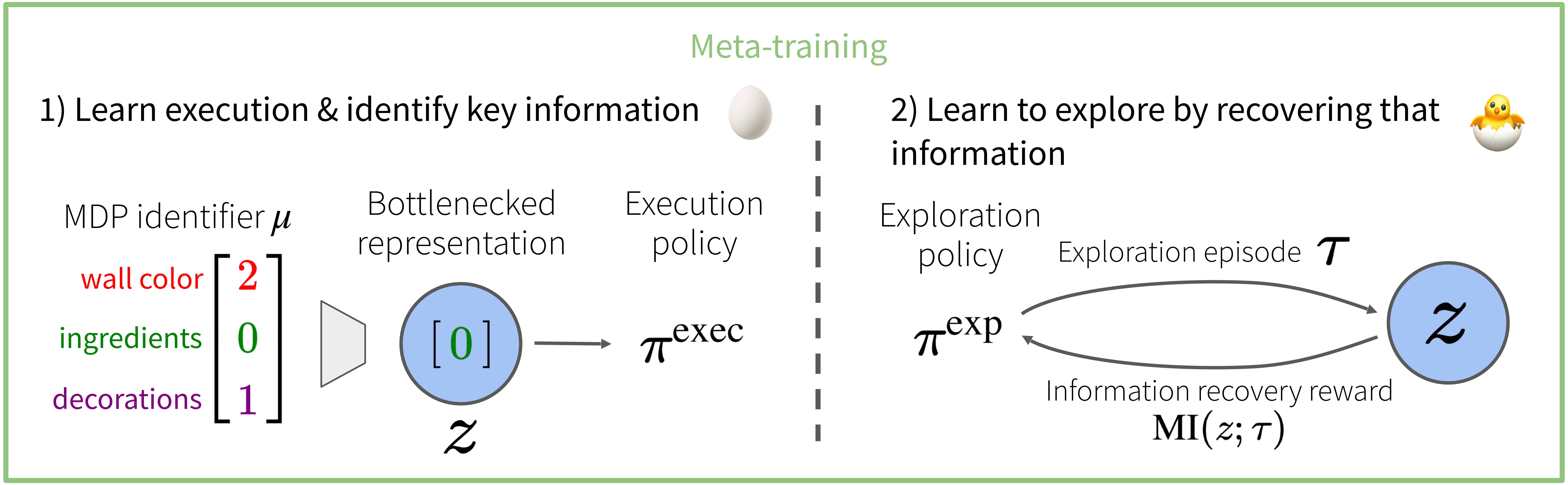
Execution policy: train ID-conditioned policy $\pi^{\text{exec}}(\mathbf{a} \mid \mathbf{s}, \mu_i)$

Meta testing

Explore: $\mathscr{D}_{\text{tr}} \sim \pi^{\text{exp}}(\mathbf{a} \mid \mathbf{s})$ Infer task: $\hat{\mu} \sim q(\mu \mid \mathscr{D}_{\text{tr}})$ Perform task: $\pi^{\text{exec}}(\mathbf{a} \mid \mathbf{s}, \hat{\mu})$

+ no longer need to model dynamics, rewards — may not generalize well for one-hot μ

Solution #3: Decouple by acquiring representation of task relevant information


Meta-training
1) Learn execution & identify key information
2) Learn to explore by recovering that information
MDP identifier μ
wall color
ingredients
decorations
$\begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix}$
Bottlenecked representation
$[0]$
z
Execution policy
π^{exec}
Exploration policy
π^{exp}
Exploration episode τ
z
Information recovery reward
$\mathbf{MI}(z; \tau)$

Aside: How can we bottleneck the information in a neural net's representation?

V0: Add noise the representation.

$\epsilon \sim \mathcal{N}(0,I) \quad \bar{\mathbf{z}} = \mathbf{z} + \epsilon$

+ will discard information 😊

- if done at test time, my discard good info

- if done during training, model can increase magnitude of $\mathbf{z}$

Key ideas:

1. Add Gaussian noise during training
2. Prevent the model from increasing magnitude

V1: Variational information bottleneck

Add noise before passing representation to next layer: $\epsilon \sim \mathcal{N}(0,I) \quad \bar{\mathbf{z}} = \mathbf{z} + \epsilon$

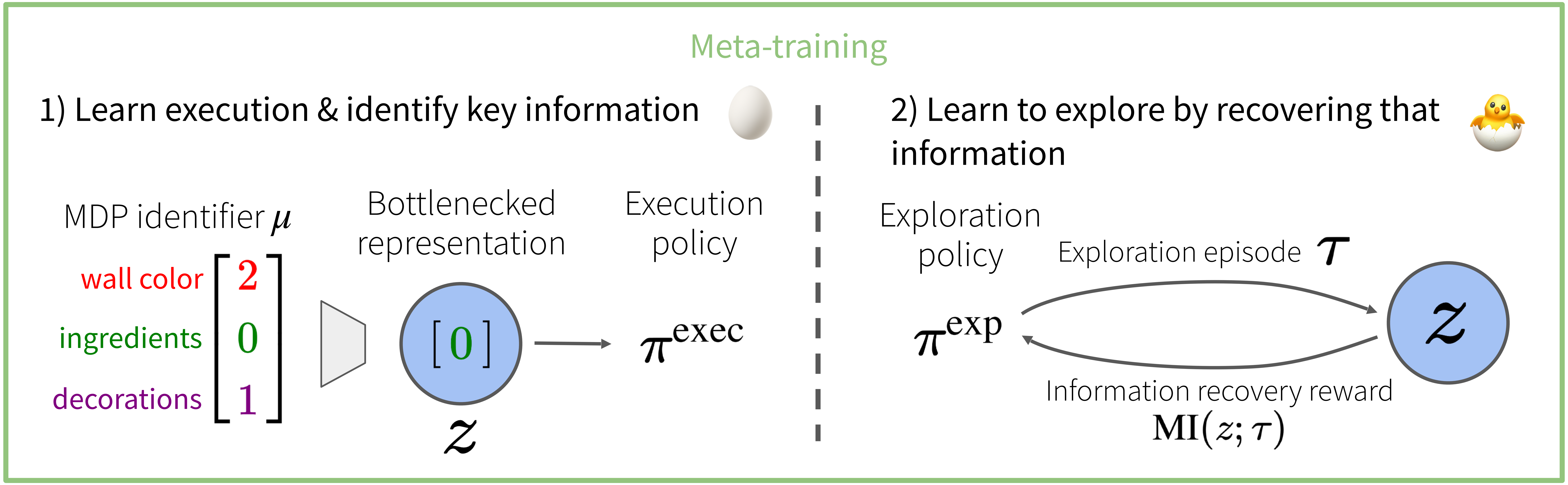
Modify loss term:

$L_{\text{tr}} + ||\mathbf{z}||^2$

-> equivalent to $D_{KL}\left(F(z \mid \mu_i) || \mathcal{N}(0,1)\right)$.

Alemi, Fischer, Dillon, Murphy. *Deep Variational Information Bottleneck*. ICLR 2017.

Solution #3: Decouple by acquiring representation of task relevant information



Meta-training
1) Learn execution & identify key information
2) Learn to explore by recovering that information
MDP identifier μ
wall color
ingredients
decorations
$\begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix}$
Bottlenecked representation
$[0]$
z
Execution policy
π^{exec}
Exploration policy
π^{exp}
Exploration episode τ
z
Information recovery reward
$\mathbf{MI}(z;\tau)$

Train $\pi^{\text{exec}}(\mathbf{a}\,|\,\mathbf{s}, z_i)$ and encoder $F(z_i\,|\,\mu_i)$ to:

$$\max \sum_i \mathbb{E}_{\pi^{\text{exec}}}[r_i] - D_{\text{KL}}\left(F(z_i|\mu_i) \| \mathcal{N}(0,1)\right)$$

Train π^{exp} such that collected $\mathscr{D}_{\text{tr}}$ is predictive of z_i.

In practice: (1) and (2) can be trained simultaneously.

Solution #3: Decouple by acquiring representation of task relevant information

Meta-training

1) Learn execution & identify key information 🥚

2) Learn to explore by recovering that information 🐣

Train $\pi^{\text{exec}}(\mathbf{a} \mid \mathbf{s}, z_i)$ and encoder $F(z_i \mid \mu_i)$ to:

$$\max \sum_i \mathbb{E}_{\pi^{\text{exec}}}[r_i] - D_{\text{KL}}\left(F(z_i|\mu_i) \| \mathcal{N}(0,1)\right)$$

Train π^{exp} such that collected $\mathcal{D}_{\text{tr}}$ is predictive of z_i.

How to formulate the *reward function* for π^{exp}?

(a) Train model $q(z_i \mid \mathcal{D}_{\text{tr}})$ (b) r_t = per-step information gain

$r_t =$ prediction error from $\tau_{1:t-1}$ — prediction error from $\tau_{1:t}$

Decoupled **R**eward-free **E**xplor**A**tion and Execution in **M**eta-Reinforcement Learning (DREAM)

Solution #3: Decouple by acquiring representation of task relevant information

(Informal) Theoretical Analysis

(1) DREAM objective is *consistent* with end-to-end optimization. [under mild assumptions]

-> can in principle recover the optimal exploration strategy

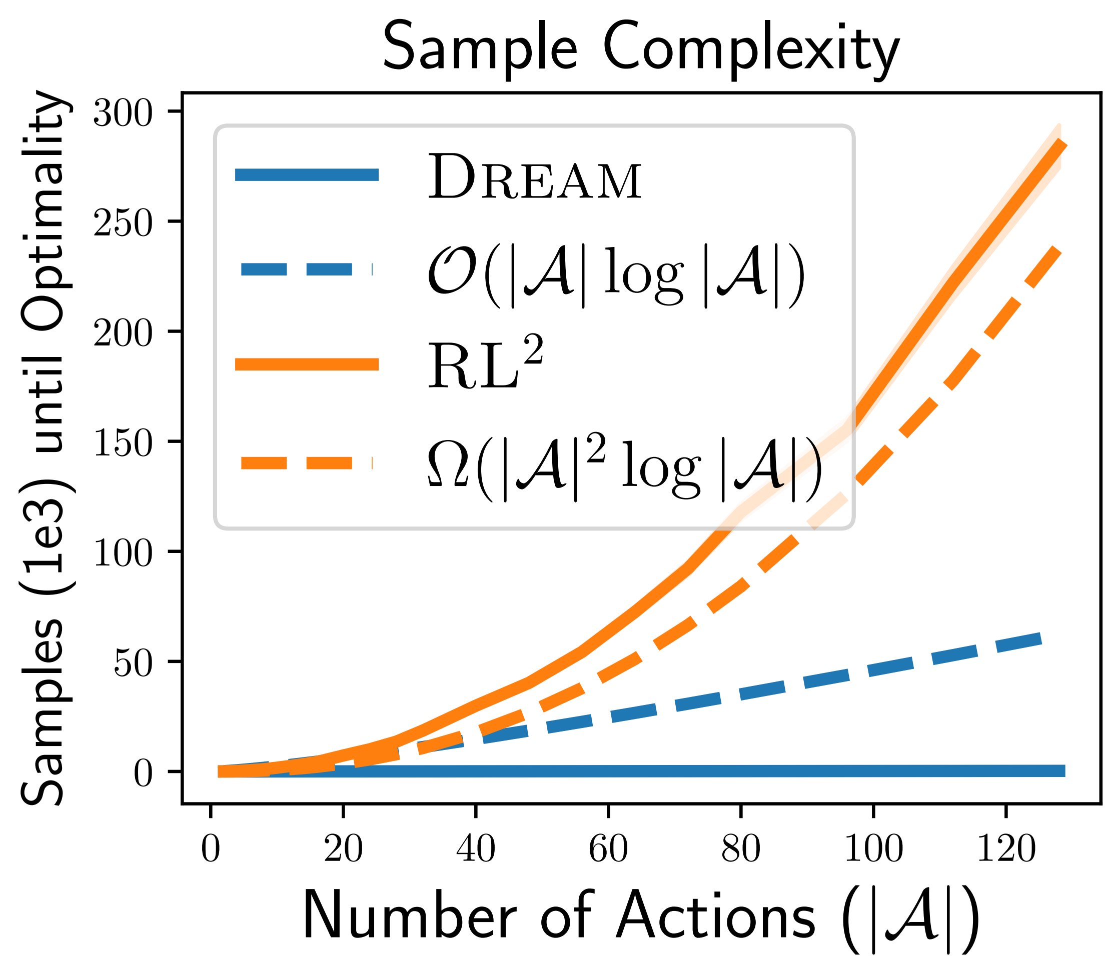
(2) Consider a bandit-like setting with $|\mathcal{A}|$ arms.

In MDP i, arm i yields reward. In all MDPs, arm 0 reveals the rewarding arm.

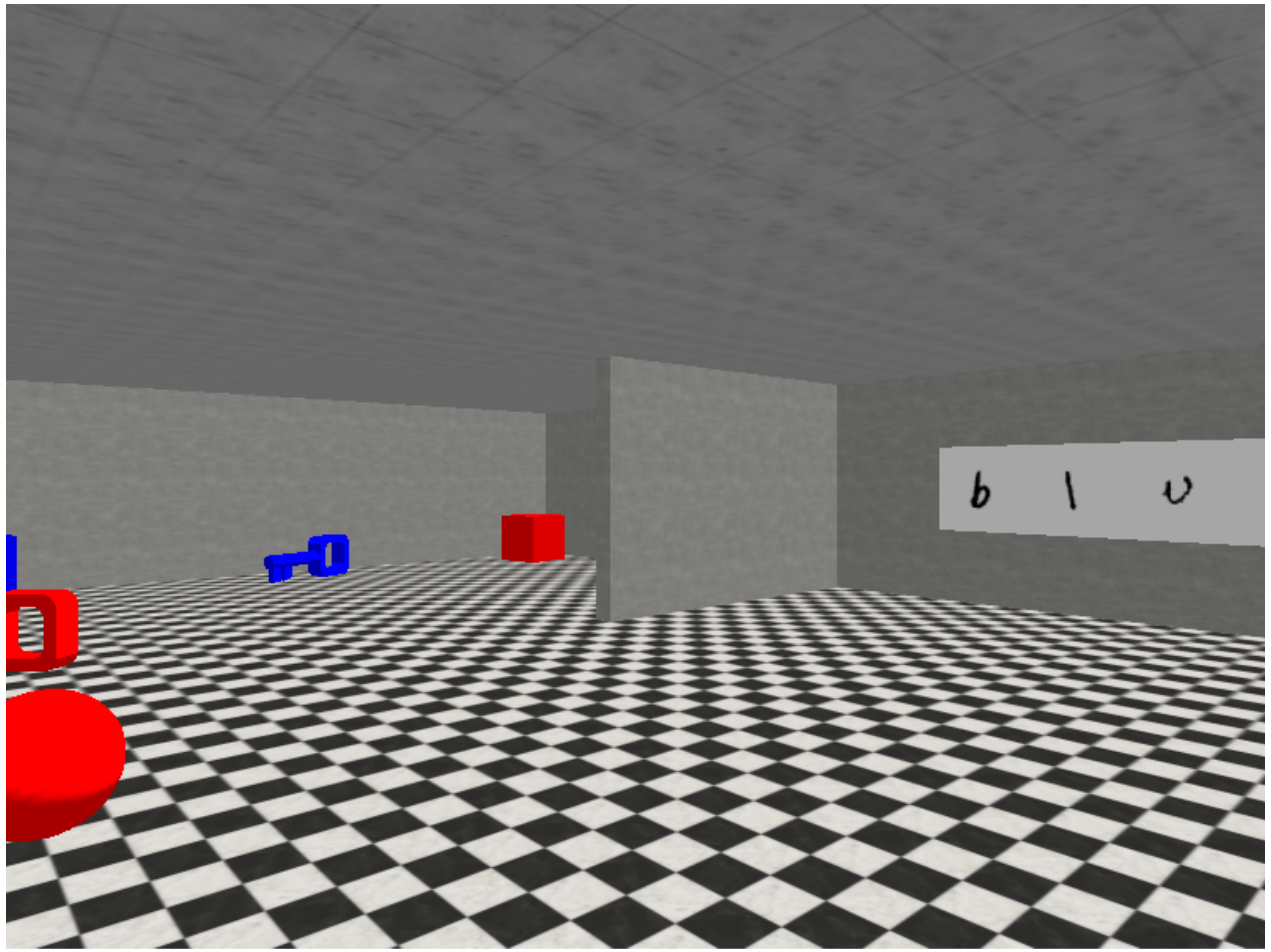
RL^2 requires $\Omega(|\mathcal{A}|^2 \log |\mathcal{A}|)$ samples for meta-optimization.

DREAM requires $\mathcal{O}(|\mathcal{A}| \log |\mathcal{A}|)$ samples for meta-optimization.

[assuming Q-learning with uniform outer-loop exploration]

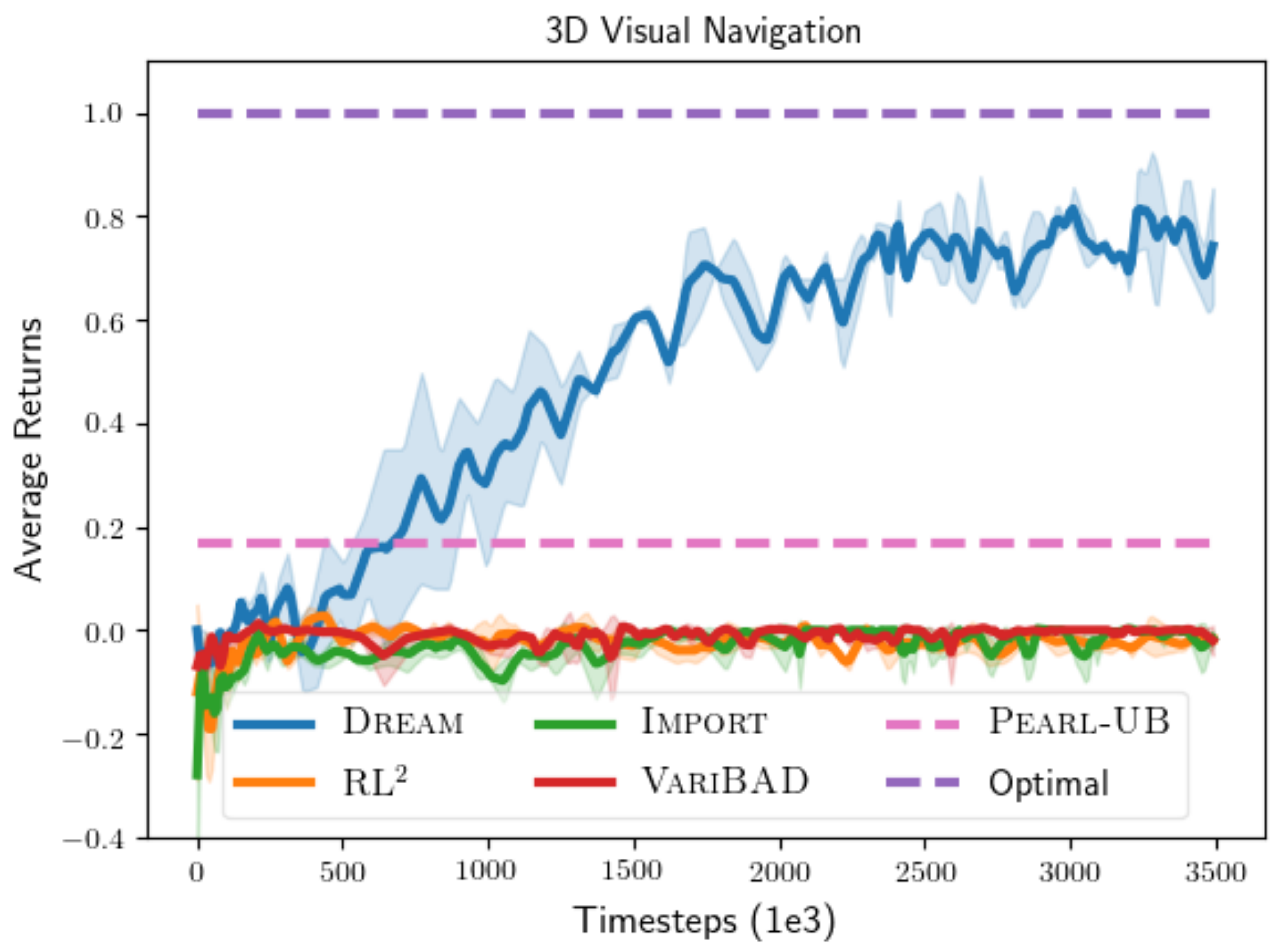


Empirical Comparison: Sparse Reward 3D Visual Navigation Problem



- Task: go to the (key / block / ball), color specified by the sign
- Agent starts on other side of barrier, must walk around to read the sign
- Pixels observations (80 x 60 RGB)
- Sparse binary reward

Quantitative Comparison

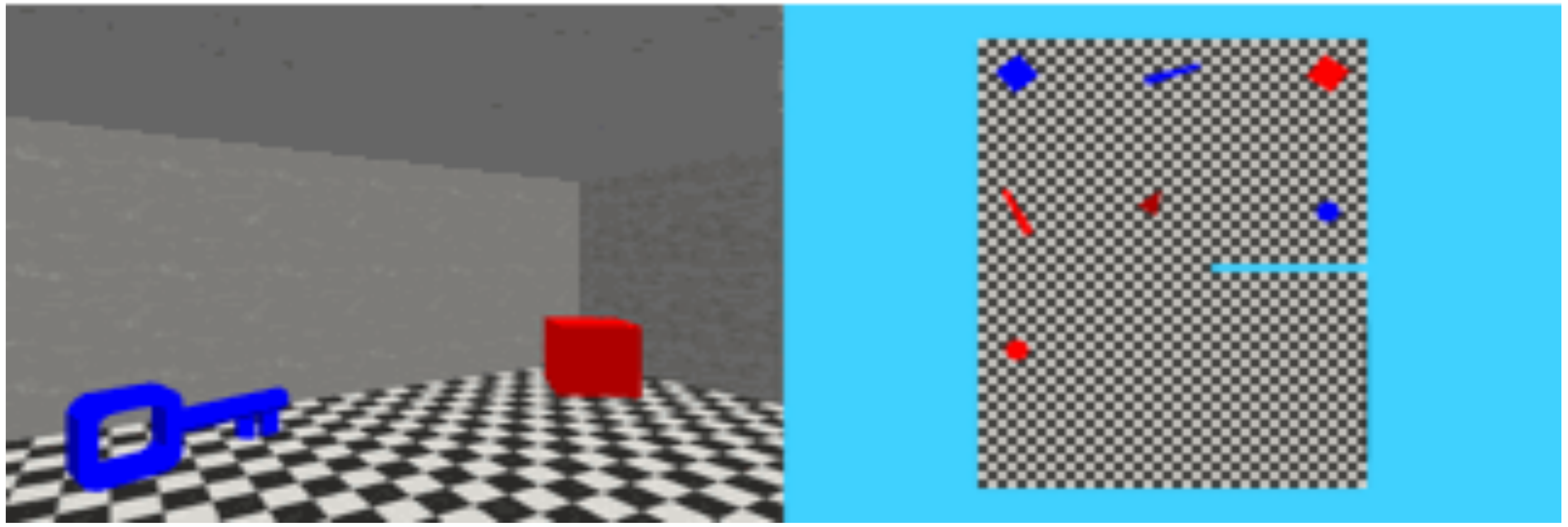


- End-to-end algorithms (RL², IMPORT, VARIBAD) perform poorly due to **coupling**
- PEARL-UB: Upper-bound on PEARL: optimal policy and Thompson-Sampling exploration, does not learn the optimal exploration strategy
- DREAM achieves near-optimal reward

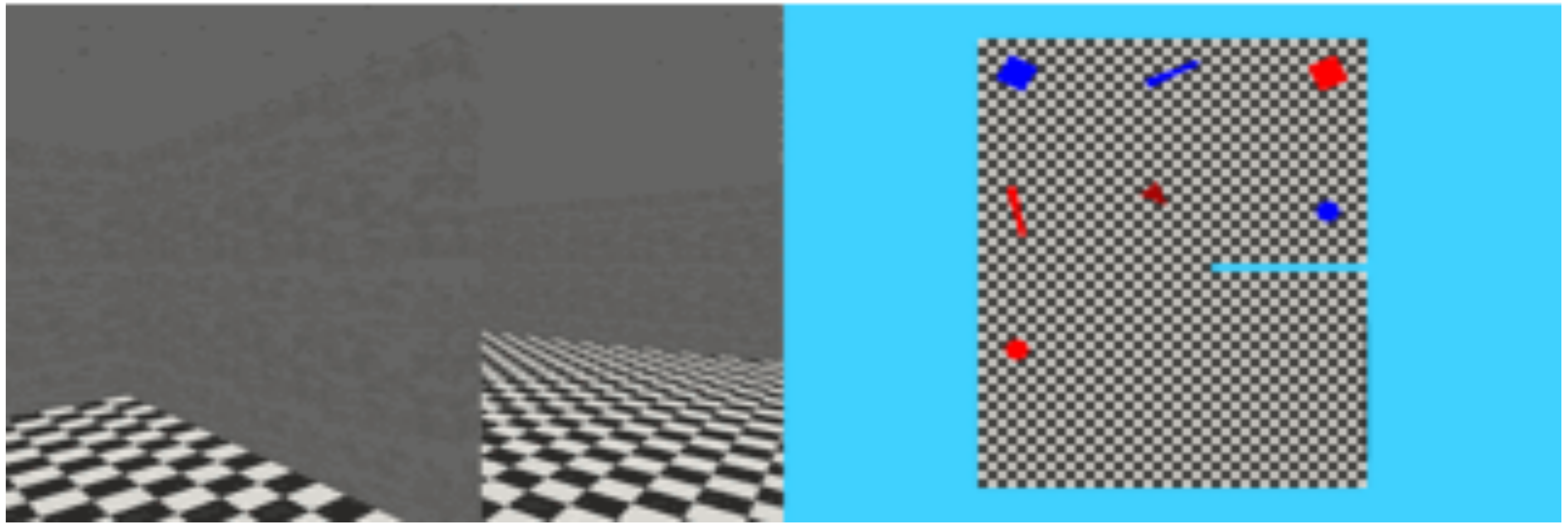
RL² (Duan et al., 2016), IMPORT (Kamienny et al., 2020), VARIBAD (Zintgraf et al., 2019), PEARL (Rakelly, et. al., 2019), Thompson, 1933

Qualitative Results for DREAM

Exploration episode



Execution episode
Goal: Go to key



How Do We Learn to Explore in Meta-RL?

End-to-End

- + leads to optimal strategy in principle
- -- challenging optimization when exploration is hard

Alternative Strategies

- + easy to optimize
- + many based on principled strategies
- -- suboptimal by arbitrarily large amount in some environments.

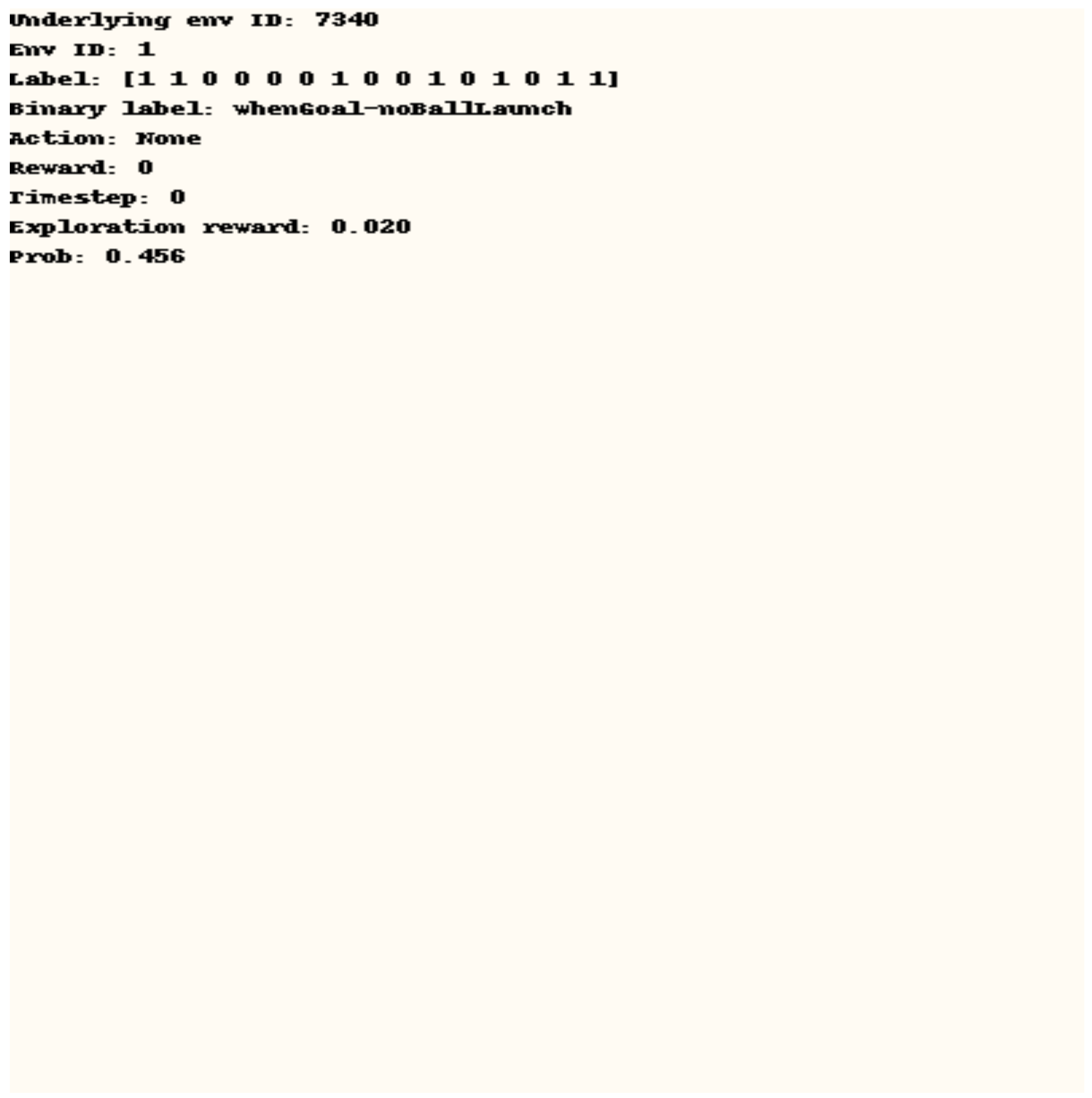
Decoupled Exploration & Execution

- + leads to optimal strategy in principle
- + easy to optimize in practice
- -- requires task identifier

Time Permitting: Applying Meta-RL to CS Education

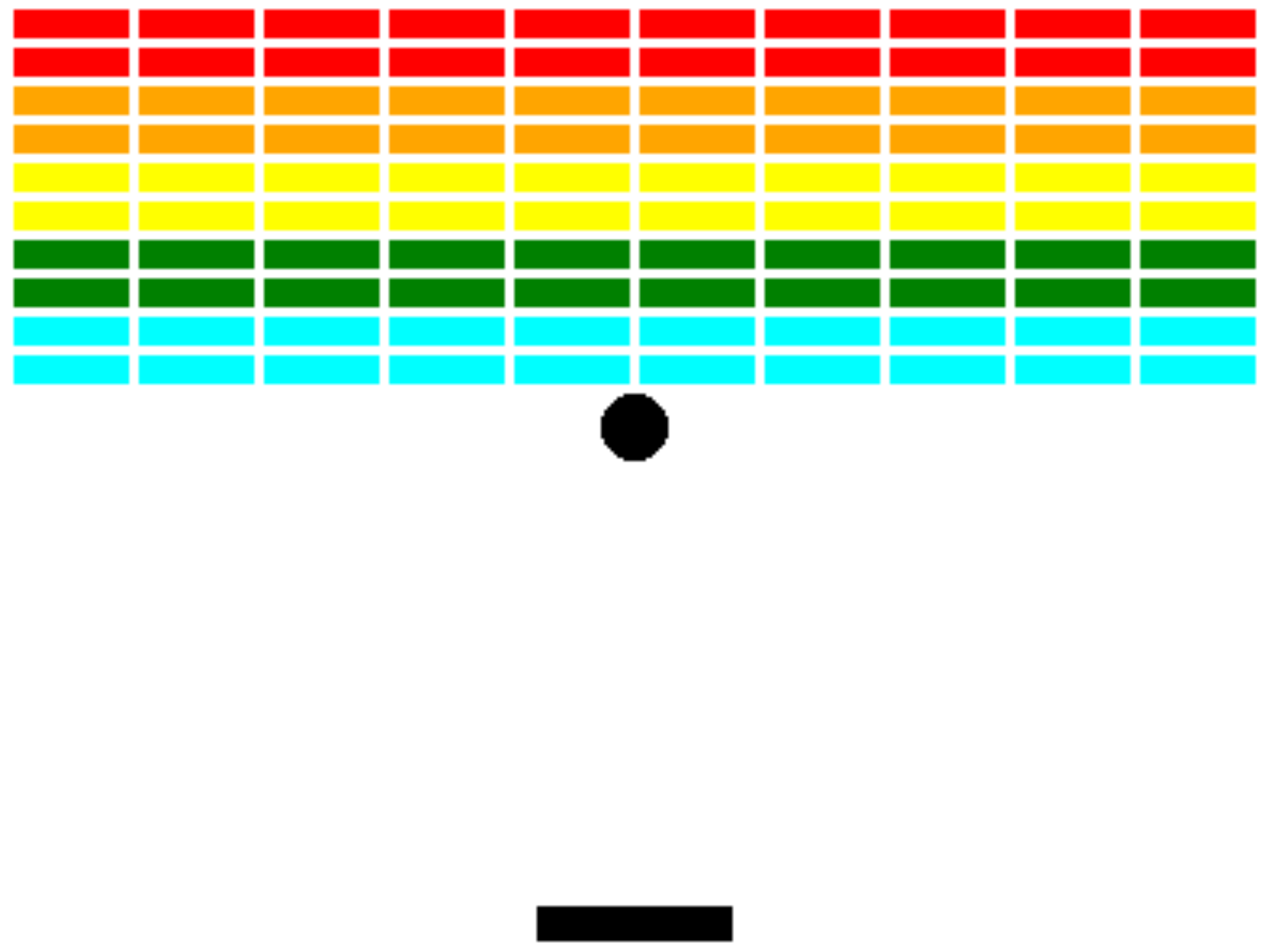
Example application: finding bugs & providing feedback in student programs

Bounce programming assignment
(Code.org)



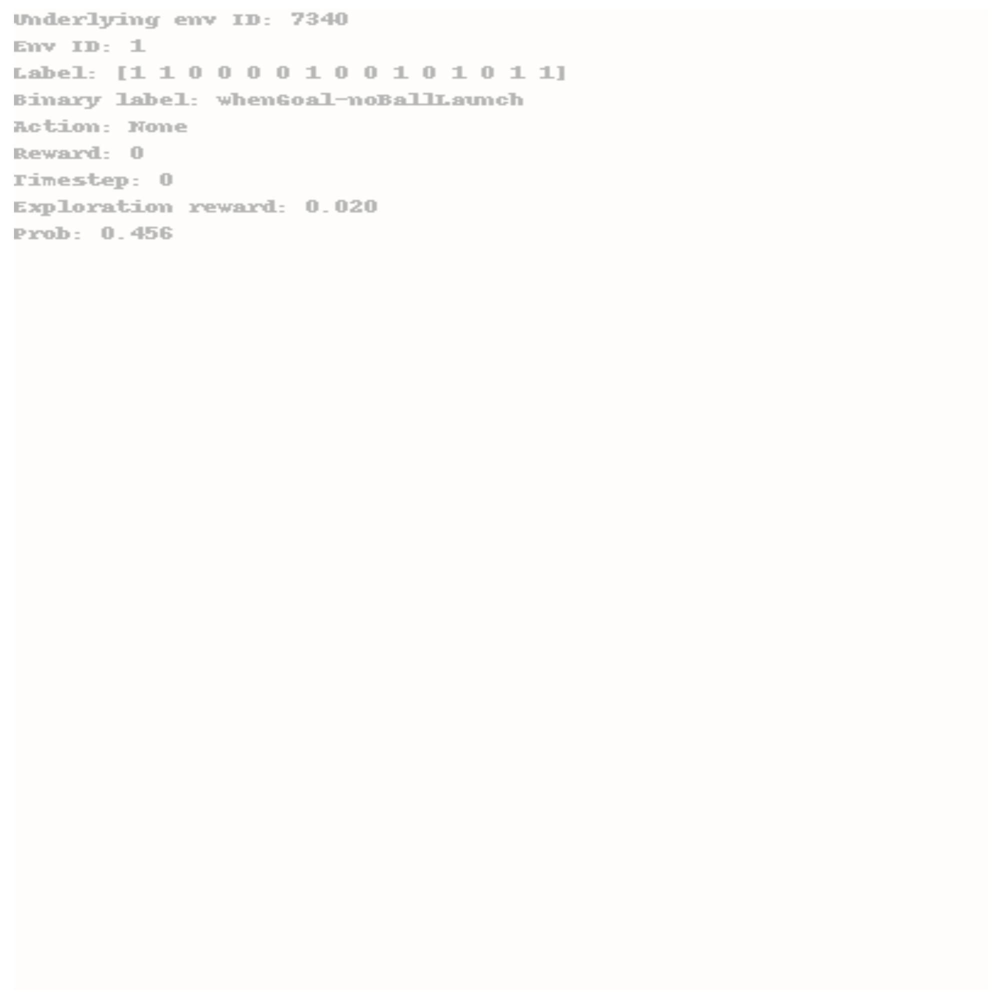
```
Underlying env ID: 7340
Env ID: 1
Label: [1 1 0 0 0 0 1 0 0 1 0 1 0 1 1]
Binary label: whenGoal-noBallLaunch
Action: None
Reward: 0
Timestep: 0
Exploration reward: 0.020
Prob: 0.456
```

Breakout assignment
(CS106A)

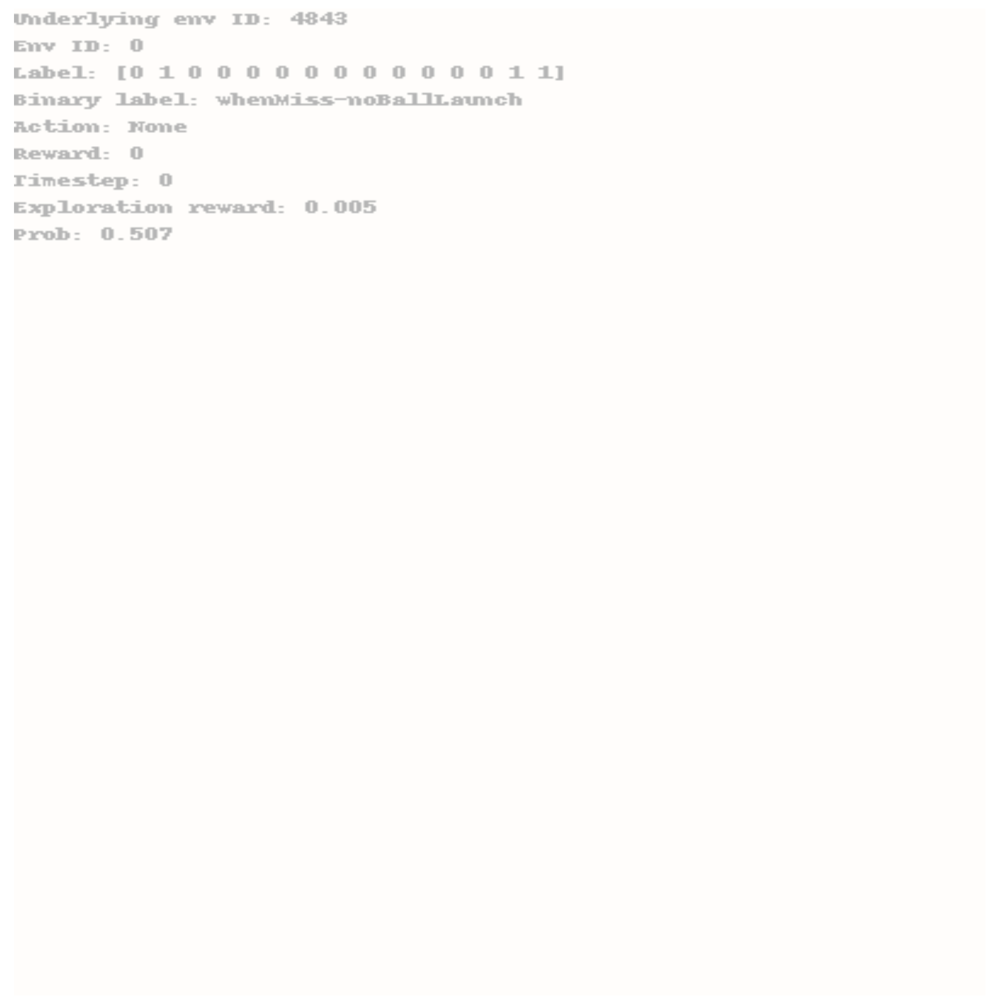


Time-consuming for instructors/TAs to give feedback, grades.
Use meta-RL to learn exploration!

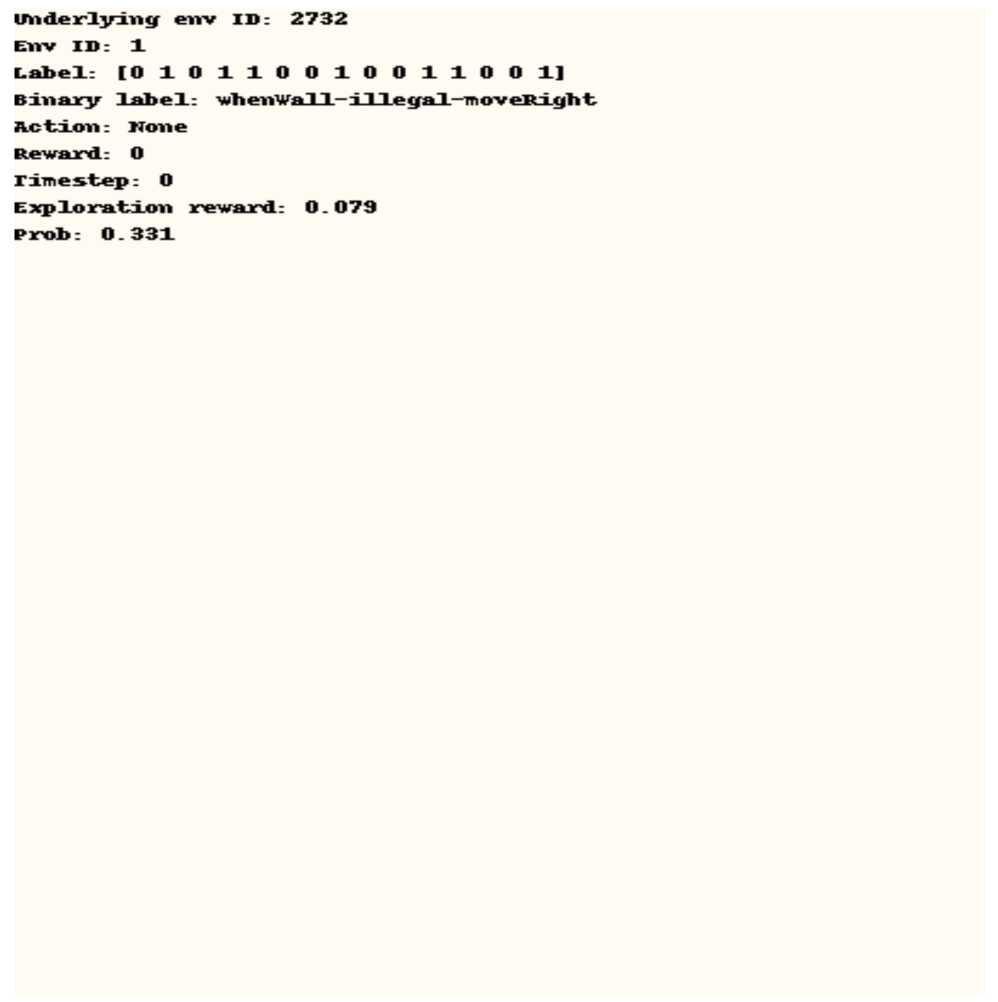
Experiments: Learned Exploration Behavior on Bounce



```
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
```



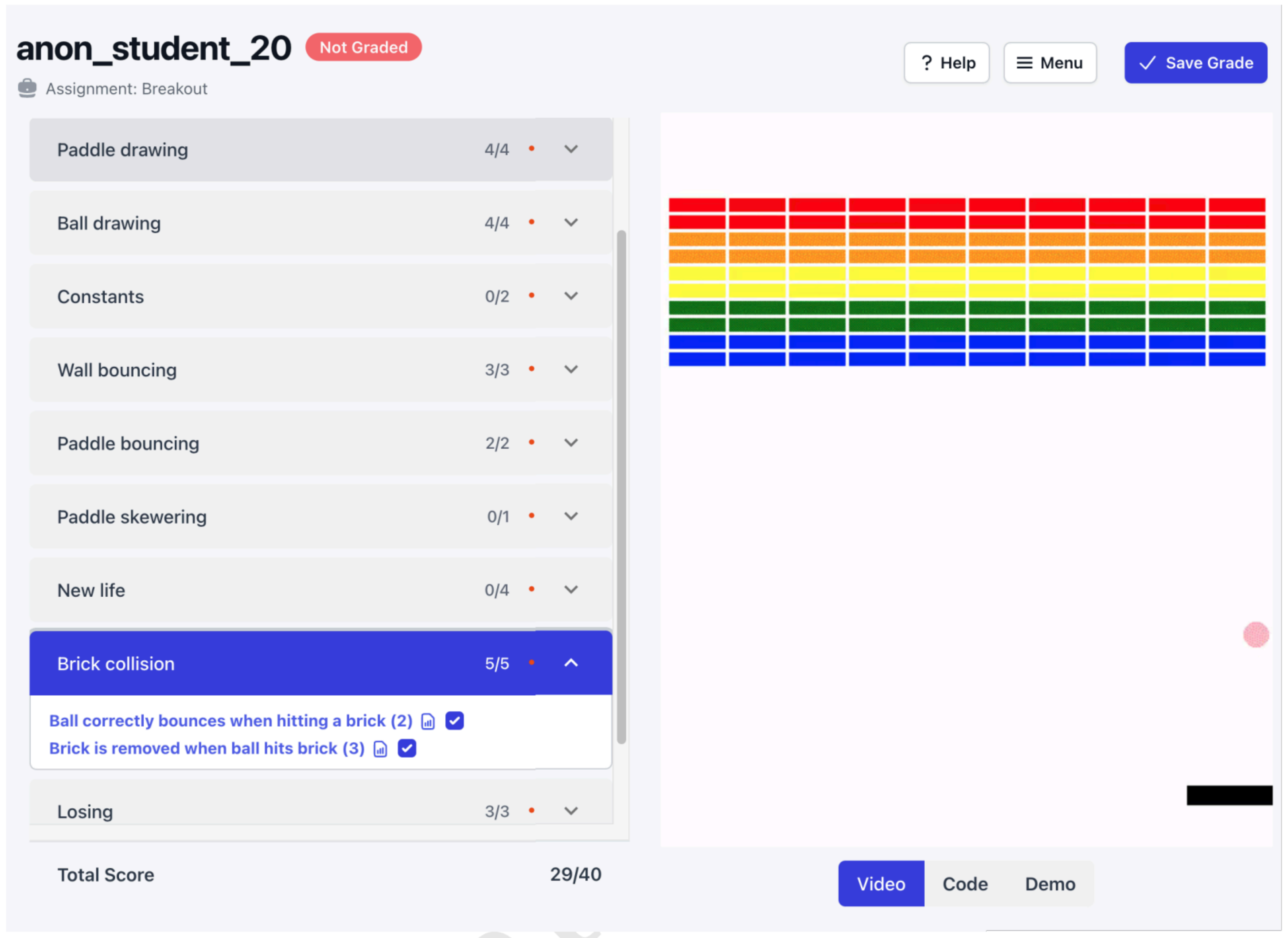
```
Underlying env ID: 4843
Env ID: 0
Label: [0 1 0 0 0 0 0 0 0 0 0 0 0 1 1]
Binary label: whenMiss-noBallLaunch
Action: None
Reward: 0
Timestep: 0
Exploration reward: 0.005
Prob: 0.507
```



```
Underlying env ID: 2732
Env ID: 1
Label: [0 1 0 1 1 0 0 1 0 0 1 1 0 0 1]
Binary label: whenWall-illegal-moveRight
Action: None
Reward: 0
Timestep: 0
Exploration reward: 0.079
Prob: 0.331
```

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)



Paddle drawing	4/4
Ball drawing	4/4
Constants	0/2
Wall bouncing	3/3
Paddle bouncing	2/2
Paddle skewering	0/1
New life	0/4
Brick collision	5/5
Losing	3/3
Total Score	29/40

Ball correctly bounces when hitting a brick (2) ✓
Brick is removed when ball hits brick (3) ✓

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	**4 min 49s ± 2 min 5s**	**92.3% ± 7.6%**
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

The plan for today

1. **Exploration**
   a. Why is exploration hard?
   b. Algorithms for exploration in bandits
   c. Exploration in robotics, LLMs
2. **Learning to Explore via Meta-Learning**
   a. Efficiently Meta-Learning Optimal Exploration <- Part of HW 4
   b. Meta-Exploration in for CS Education

Reminders

Homework 3 due **tonight**

(and HW4 out today)

Project milestone due **next Friday**

Next week: Can we make reinforcement learning more autonomous?

Can we do extra long-horizon tasks leveraging hierarchy?