Use Offline data in RL

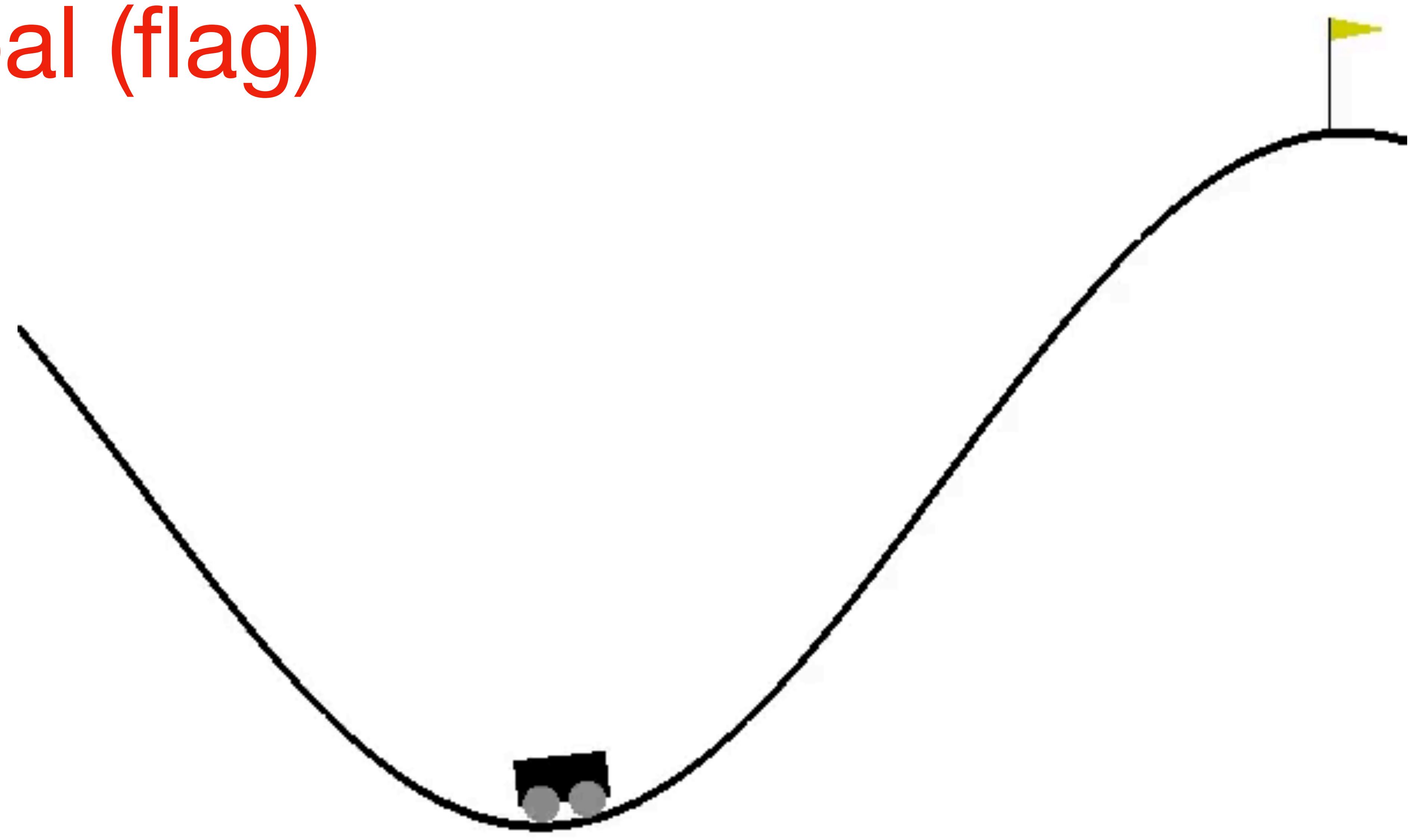
Annoucements

1. PA3 will be released today, due in three weeks
2. Almost done grading HW2 and Prelim exam
3. No office hour tmr

Failure mode of Policy Gradient

The mountainCar Example (i.e., the sparse reward problem)

We have reward zero everywhere except at the goal (flag)



The prob of a random policy hitting the goal is exponentially small $\approx 2^{-H}$

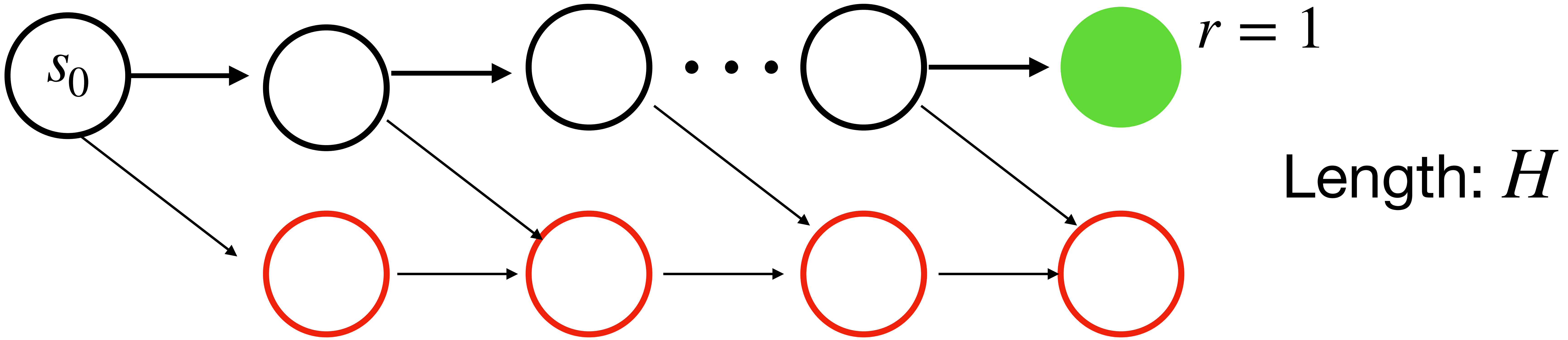
$$\text{PG} := R(\tau) \sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \mid s_h) \approx 0$$

i.e., a random policy is a perfect locally optimal policy

Failure model of Policy Gradient

The Combination Lock Example (i.e., the sparse reward problem)

(1) We have reward zero everywhere except at the goal (the right end);
(2) Every black node, one of the two actions will lead the agent to the dead state (red)



What is the probability of a random policy generating a trajectory that hits the goal?

Question Today:

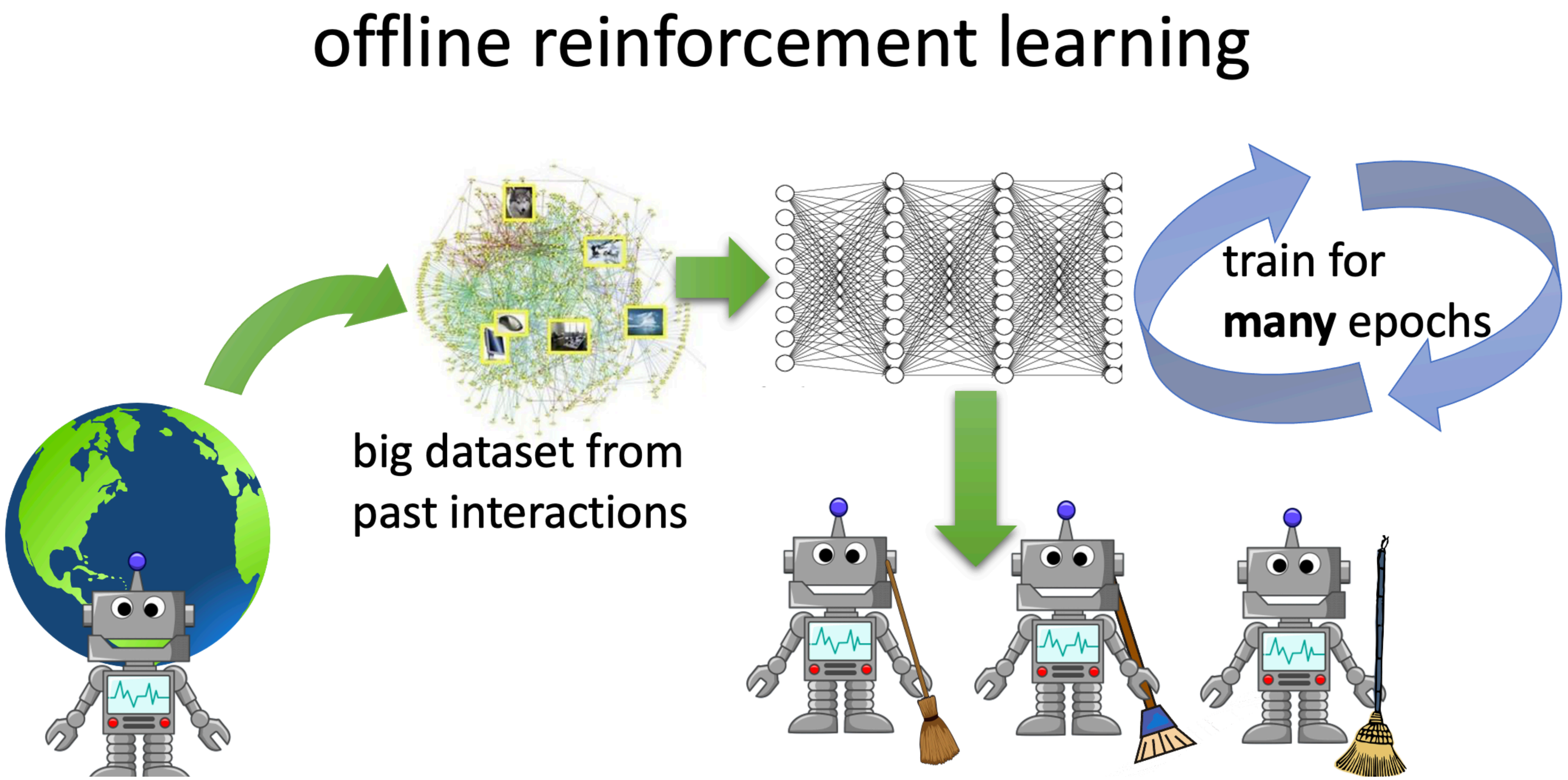
Make RL (DQN and PG/PPO) more efficient by leveraging offline data

Outline

1. Using offline data in the DQN framework

2. Using offline data in PG via Reset

Detour: Offline RL, i.e., RL with only pre-collected dataset



Note here loop is
not closed!

The hope:

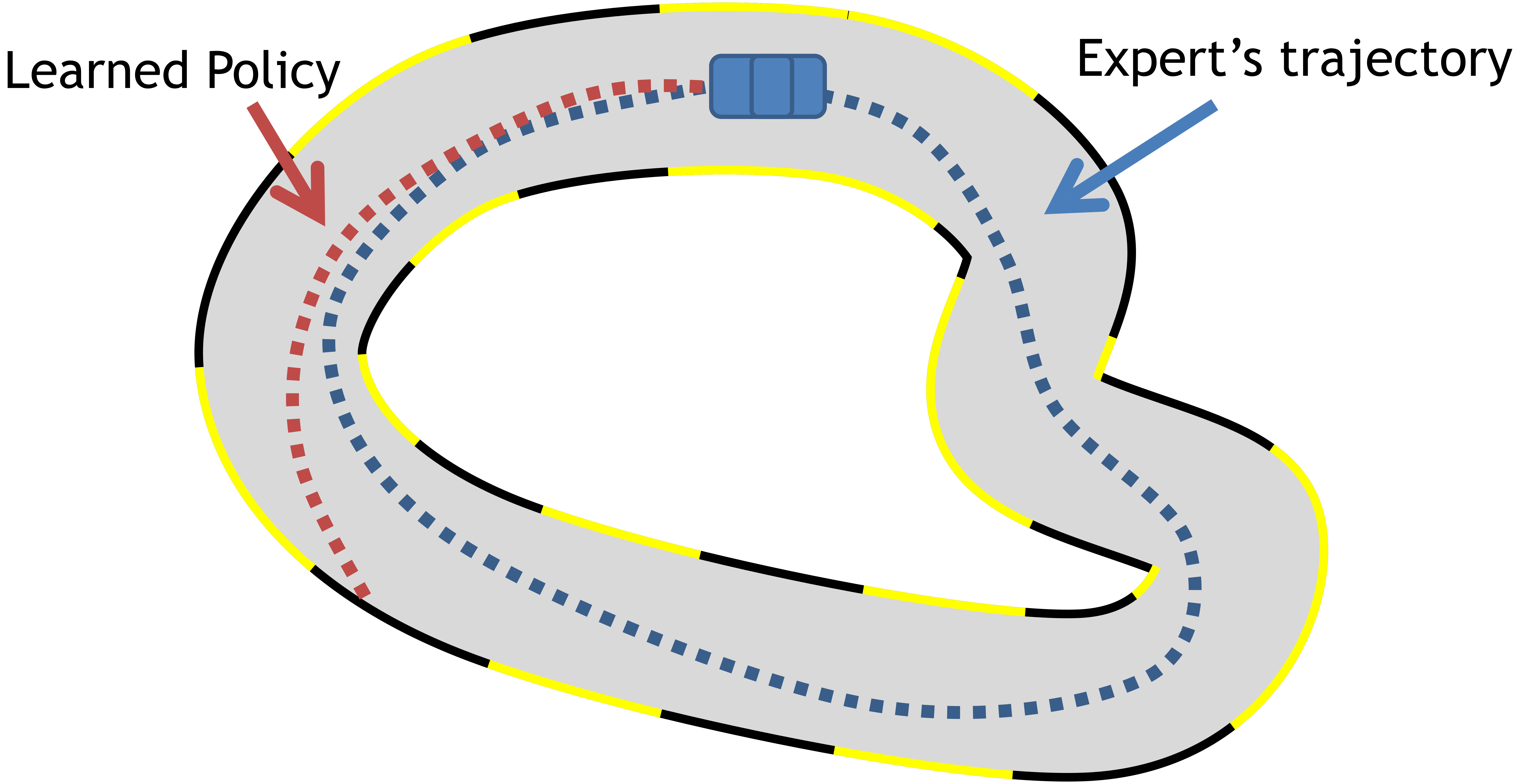
We can pre-train RL on
large logged datasets

[Image from BAIR blog post: https://bair.berkeley.edu/blog/2020/12/07/offline/]

What could go wrong?

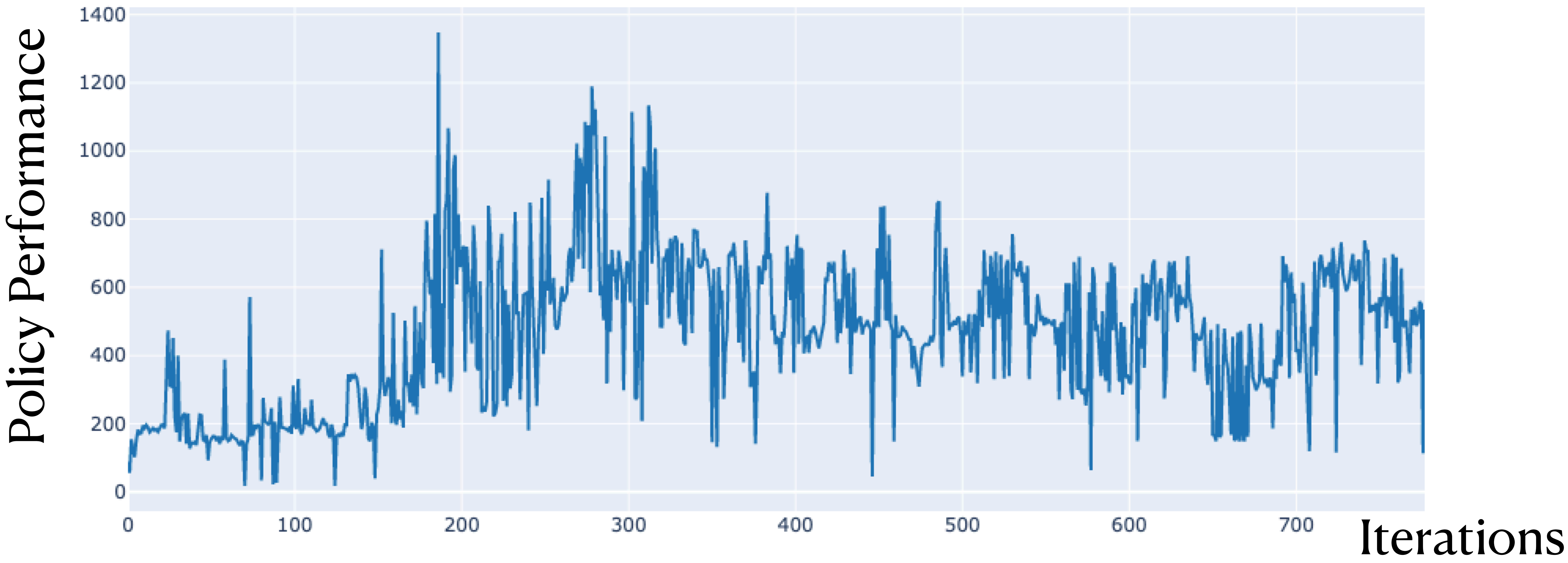
[Pomerleau89,Daume09]

- Distribution shift



Detour: Offline RL, i.e., RL with only pre-collected dataset

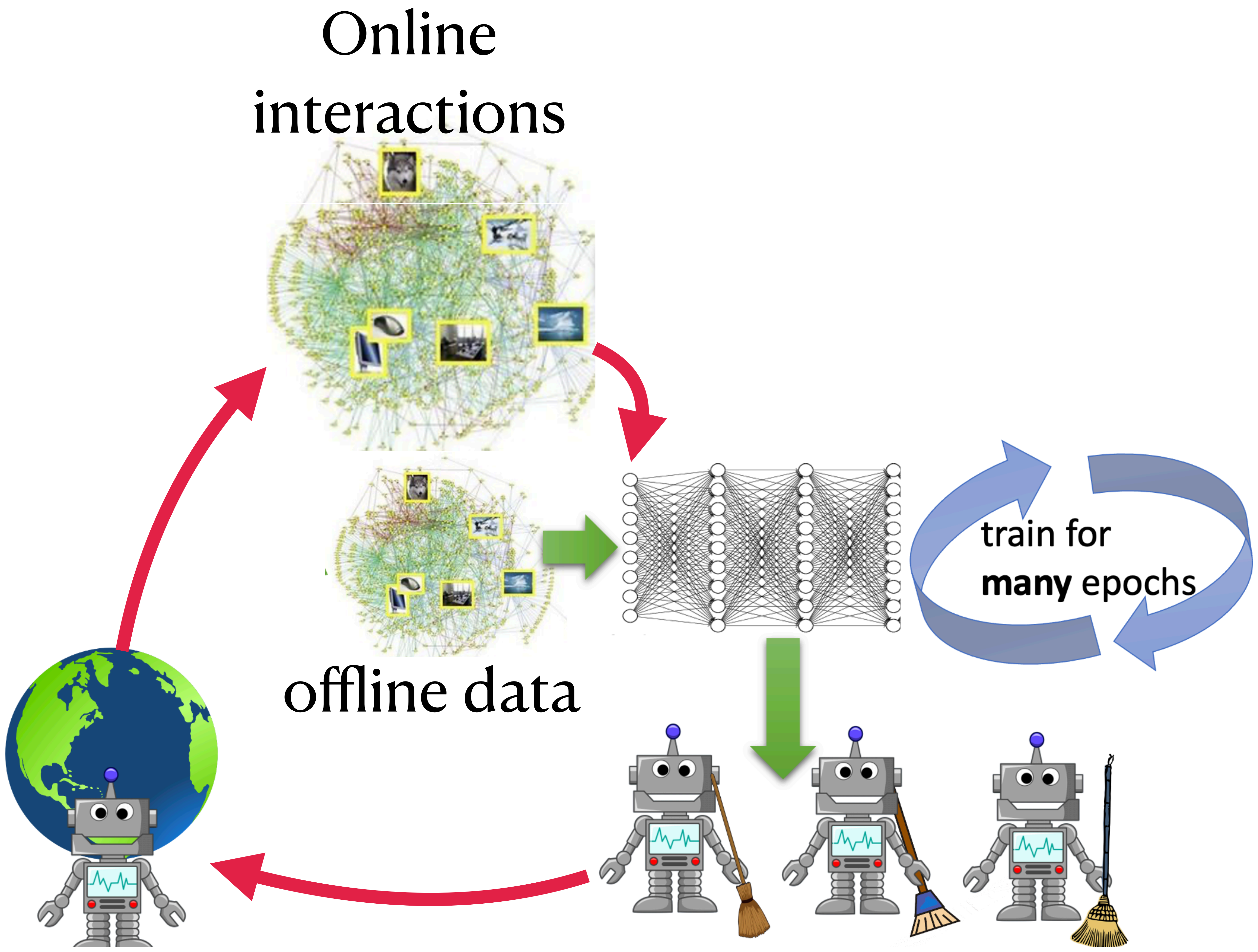
The reality: Making offline RL work reliably is hard...



A typical learning curve of some popular offline deep RL baseline tested under a standard D4RL benchmark

The rescue:

Offline data + Online Interaction



Offline data + Online is widely used in practice

1. In robotics, we typically combine offline expert demonstration with online interaction [e.g., Rajeswaran et al 17, Nair et al., 20, Zhu et al., 19]

2. In games, we combine human demonstrations with online interaction, e.g., first version of AlphaGo [deepmind], playing Hanabi [Meta AI, Hu et al, 22]

Offline data distribution

Offline data is sampled from offline distributions ν

$$\mathcal{D}_{off} = \{s, a, r, s'\}_{i=1}^{m}, \text{ where } s, a \sim \nu, s' \sim P(\cdot \mid s, a)$$

We assume offline distributions "cover" some high quality policy's traces

Algorithm: Hybrid (Deep) Q Learning (Hy-Q)

In high level, it iteratively runs DQN on combination of offline and online data

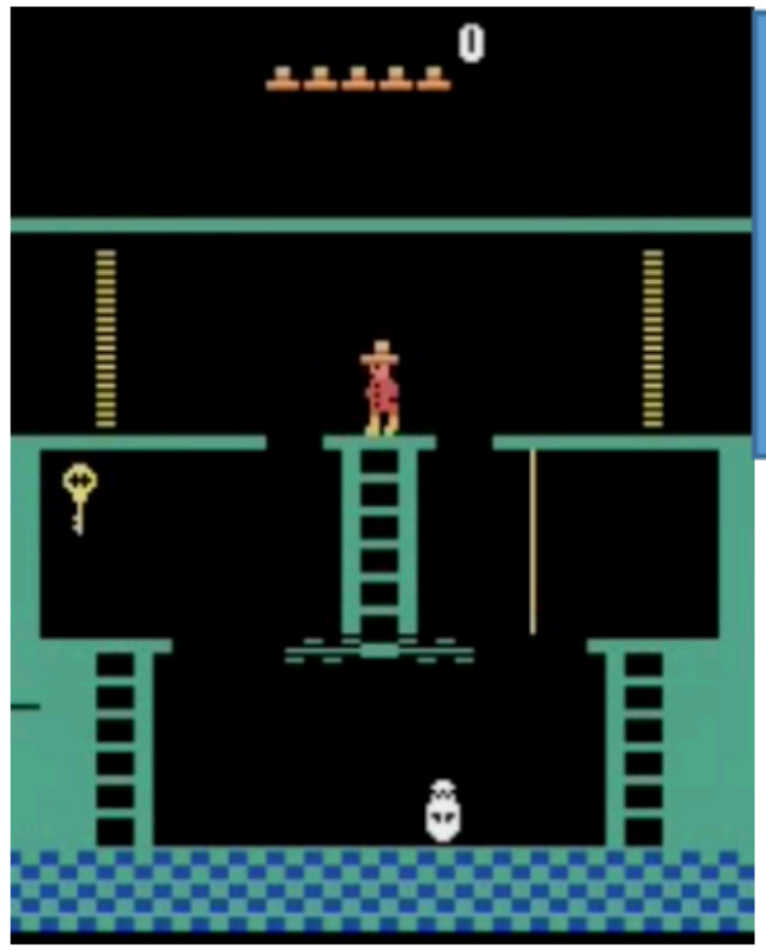
Initialize Q_{θ_0}, online replay buffer $\mathcal{D}_{on} = \emptyset$, initial state s, set target network $\widetilde{Q} = Q_{\theta_0}$

While true:

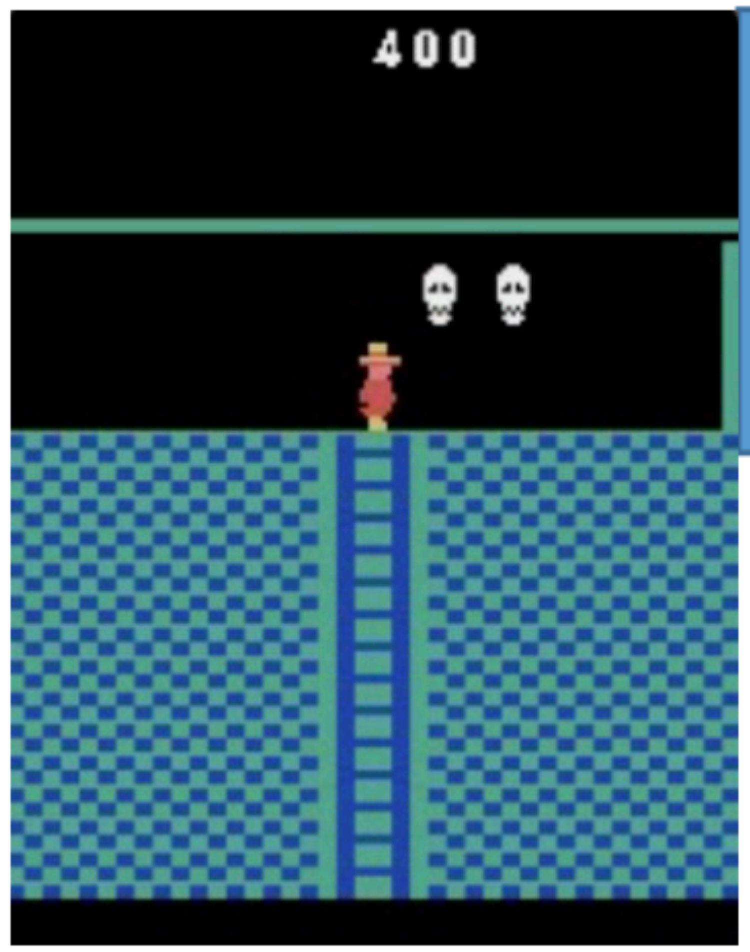
1. Run ϵ-greedy of Q_{θ_t} to collect a transition data (s, a, r, s'), $s' \sim P(s, a)$
2. Add (s, a, r, s') to online buffer $\mathcal{D}_{on}$
3. **W/ prob 0.5, sample batch $\mathcal{B}$ from $\mathcal{D}_{on}$, and otherwise from $\mathcal{D}_{off}$**
4. Q-update: $\theta_{t+1} \Leftarrow \theta_t - \eta \sum_{s,a,r,s' \in \mathcal{B}} \left(Q_{\theta_t}(s, a) - r - \gamma \max_{a'} \tilde{Q}(s', a') \right) \nabla_{\theta_t} Q_{\theta_t}(s, a)$
5. Set $s \Leftarrow s'$, and update target network once a while

How does such a simple algorithm work in practice?

Montezuma's Revenge

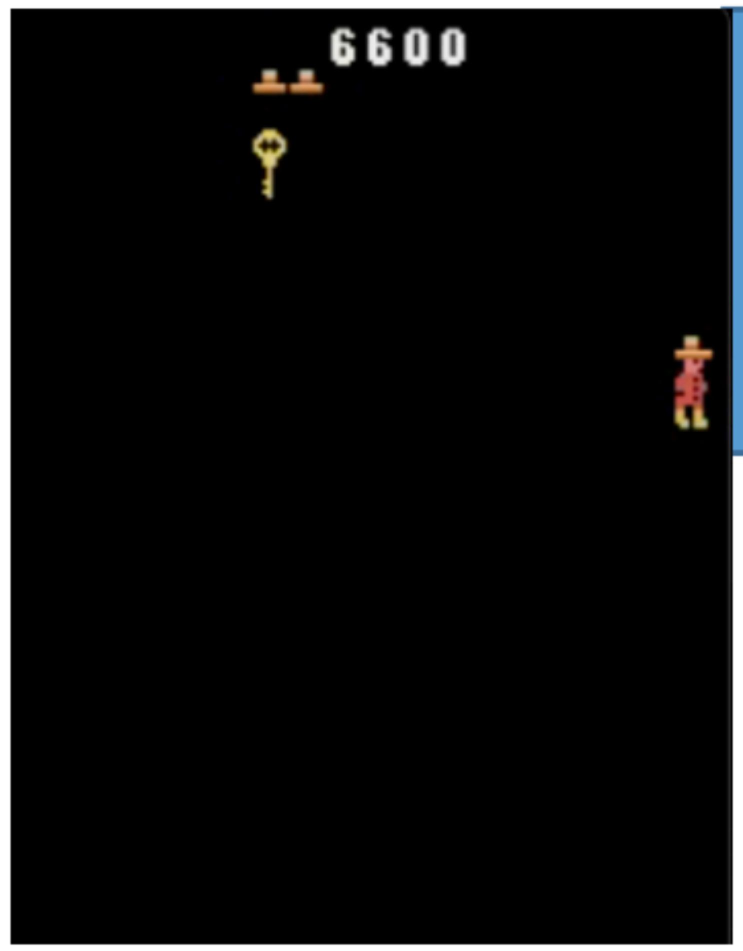


Reward
0
Room #
1



Reward
400
Room #
2

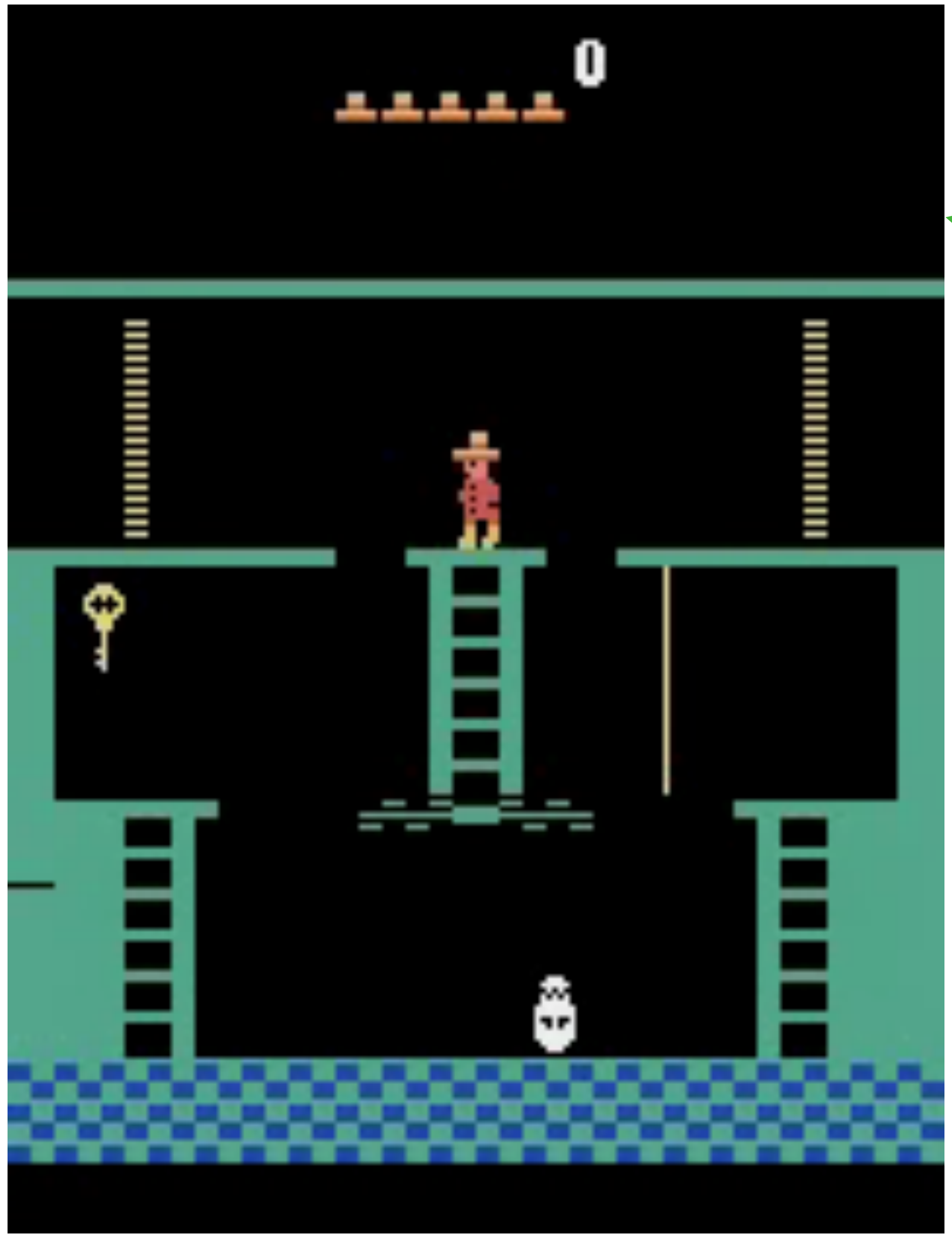
Reward
3600
Room #
14

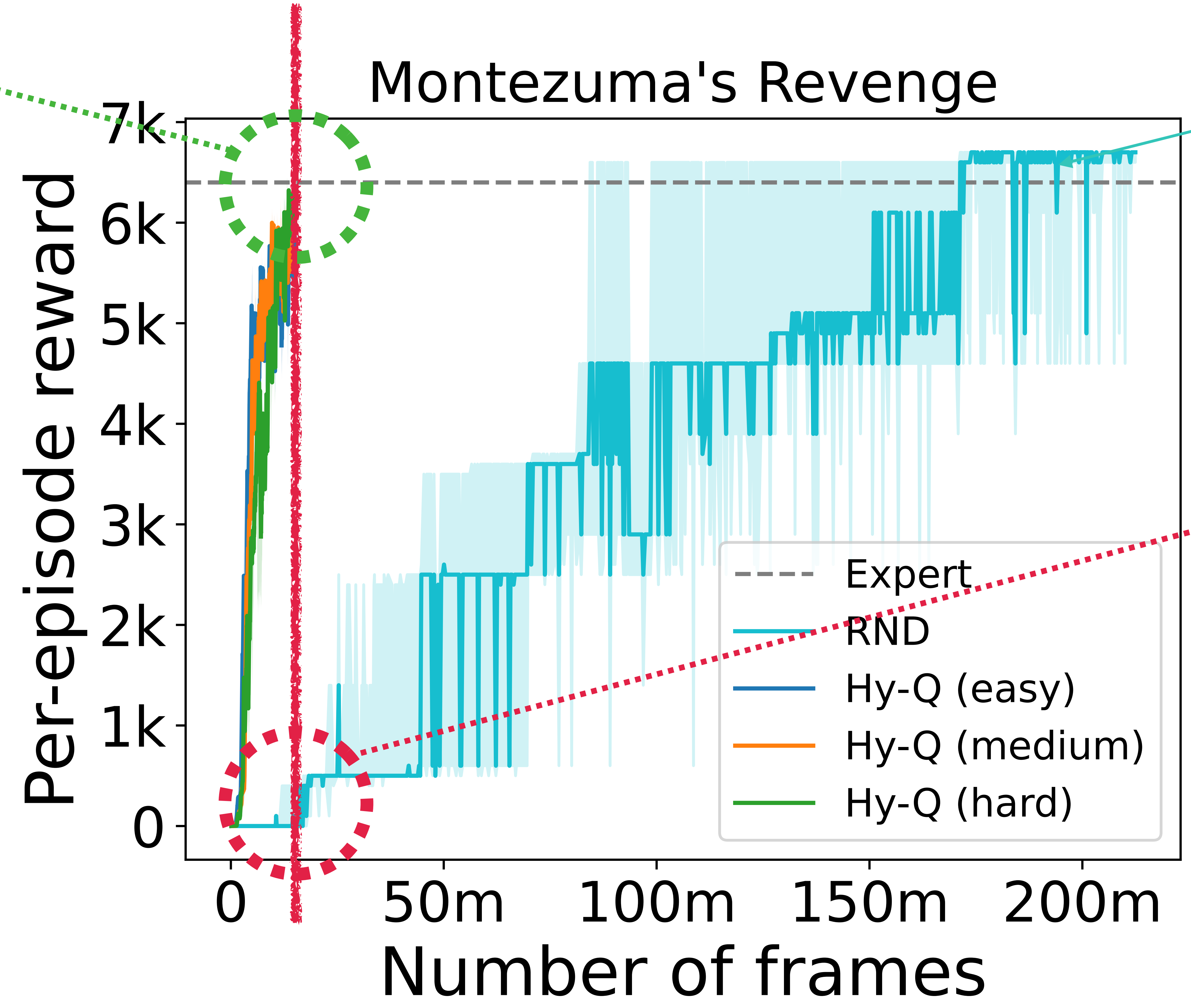


Reward
6600
Room #
23

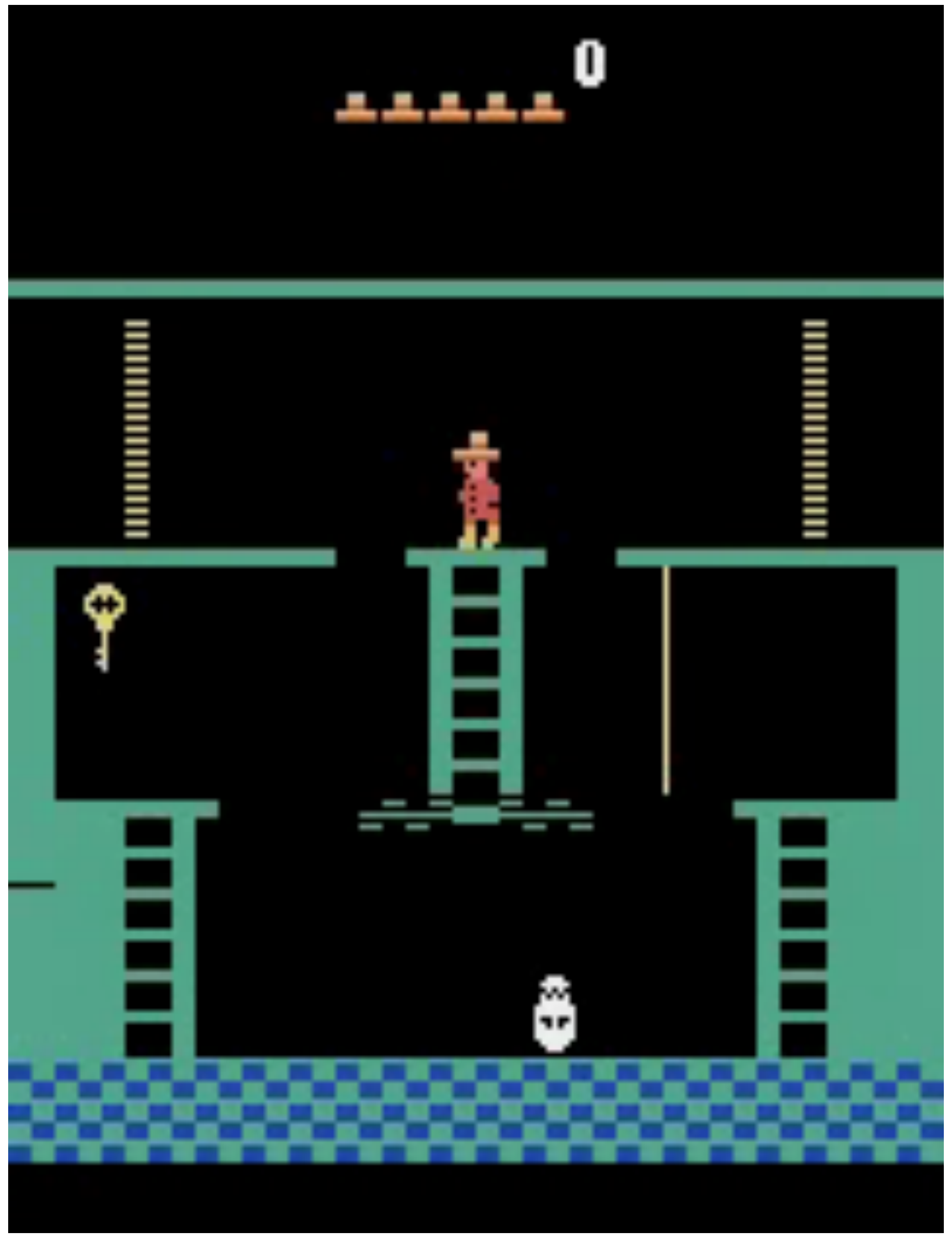
Comparison to Empirical Deep RL baseline

We construct offline dataset by mixing data from an expert policy (50%) and a low-quality policy (a random policy), w/ total 0.1 m samples

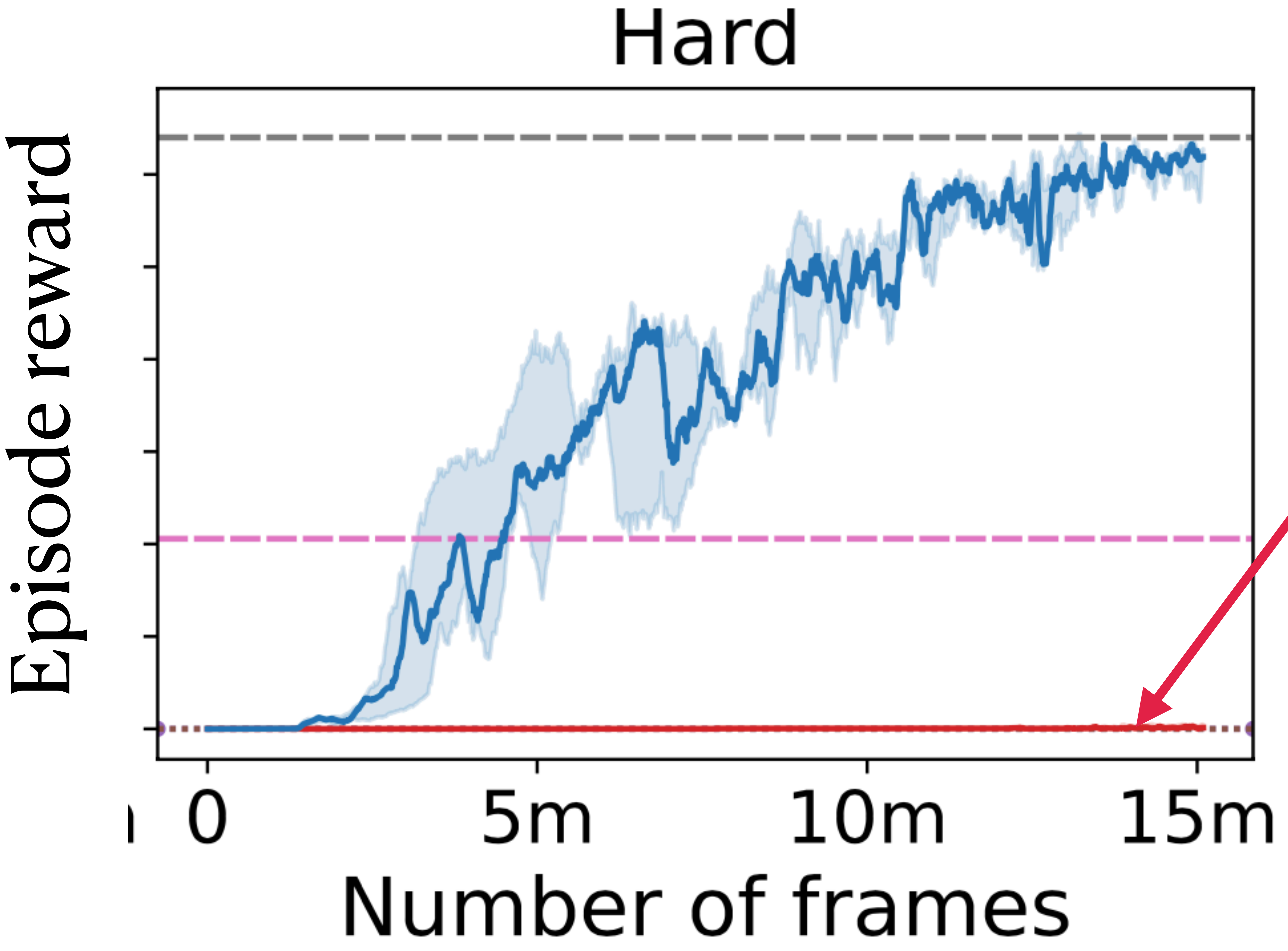




RND [BESK]: a method designed for M-revenge (openAI)



Comparison to Pure Offline RL & Imitation Learning baselines



Offline RL (and imitation learning) baselines fail completely

Further reading:

Hybrid RL: Using Both Offline and Online Data Can Make RL Efficient

Yuda Song[*] Yifei Zhou[†] Ayush Sekhari[‡] J. Andrew Bagnell[§] Akshay Krishnamurthy[¶] Wen Sun[‖]

March 14, 2023

https://arxiv.org/pdf/2210.06718

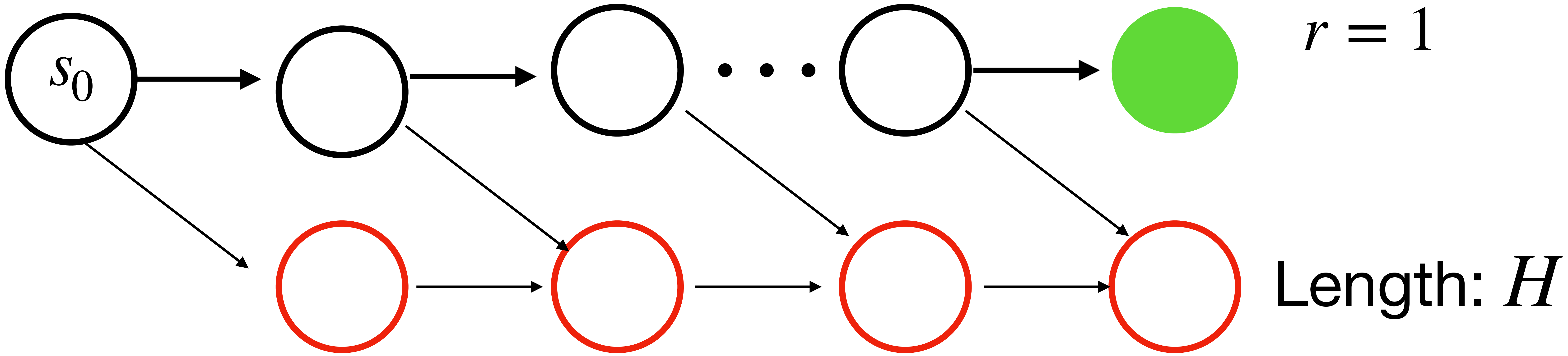
Outline

1. Using offline data in the DQN framework

2. Using offline data in PG via Reset

The Combination Lock Example (i.e., the sparse reward problem)

Instead of always starting from the s_0, what if we can start **everywhere**?



Offline data distribution

We have some offline state distribution ν, where we have a dataset

$$\mathscr{D}_{off} = \{s\}_{i=1}^{m}, \text{ where } s \sim \nu$$

We again assume offline distribution "cover" some high quality policy's traces

Taking advantage of offline data via reset

In high level, let's run PPO with ν (offline data) as the new initial state distribution

Initialize θ_0 for the policy

For $t = 0 \rightarrow T$:

Run π_θ to collect multiple trajectories where **each traj's s_0 is randomly picked from $\mathcal{D}_{off}$**

Construct the policy loss and the value loss using the trajectories

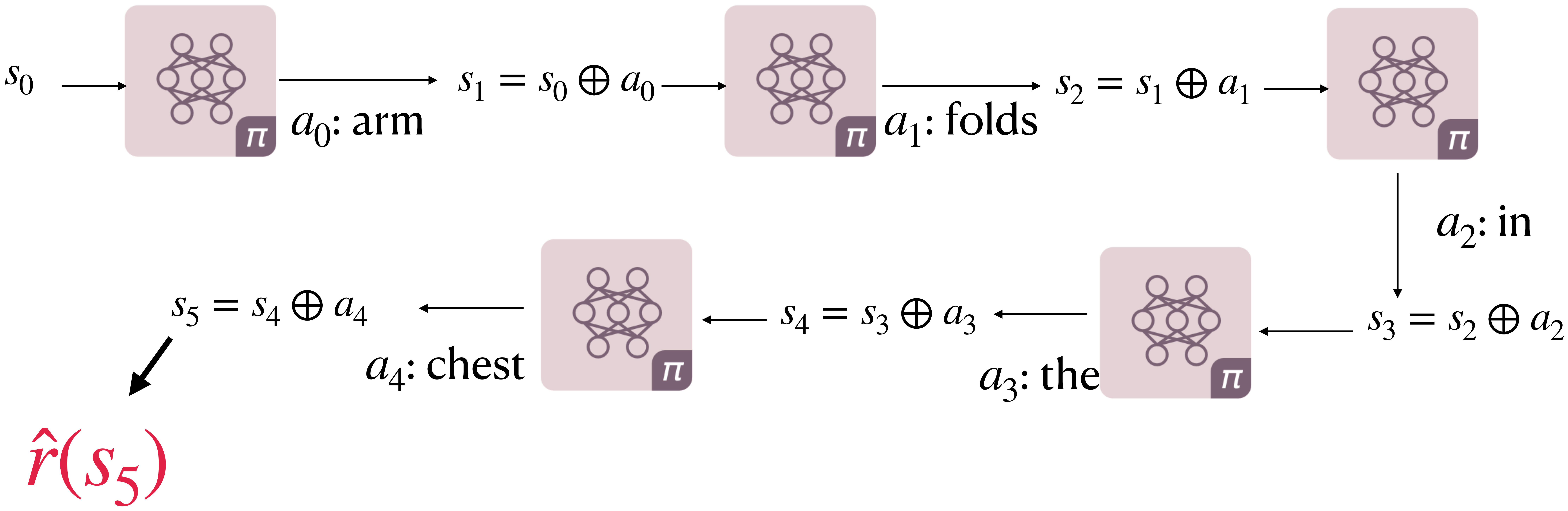
Update policy and value loss with gradient descents

Case study in post-training LLMs

Modeling text generation as an RL / MDP problem

Prompt = initial state s_0 e.g., *Generate a sentence with key words arm, chest, fold:*

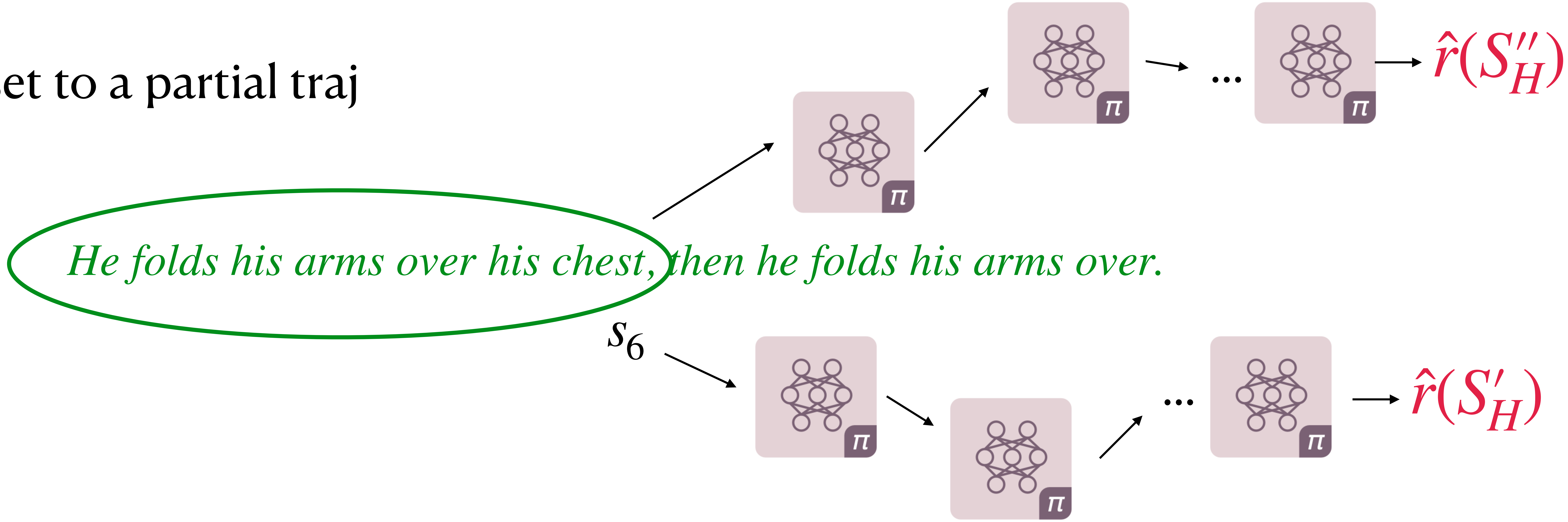
LLM as a policy π: **a sequence of tokens so far** => **the next token** (i.e., action)



Reset

Reset: we can rollout a policy π at any given partial sentence

e.g., reset to a partial traj



Reset is a game-changer in RL, both theory and practice (e.g., AlphaGo and MCTS)

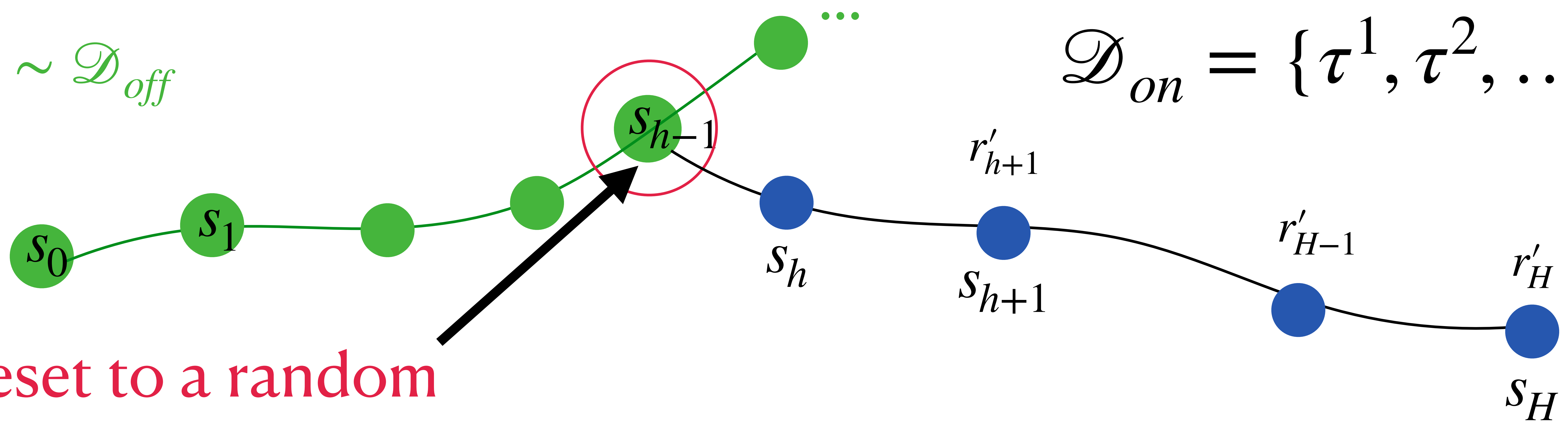
Alg: Dataset Reset Policy Optimization (DR-PO)

Reset to offline data + black-box Policy Optimization oracle (e.g., PPO)

Iteration t w/ the latest π_t:

1. Sample traj from $\mathscr{D}_{off}$

$\tau_{off} \sim \mathscr{D}_{off}$



2. Reset to a random step and rollout π_t

Repeat this sampling process n times,

$$\mathscr{D}_{on} = \{\tau^1, \tau^2, \ldots, \tau^n\}$$

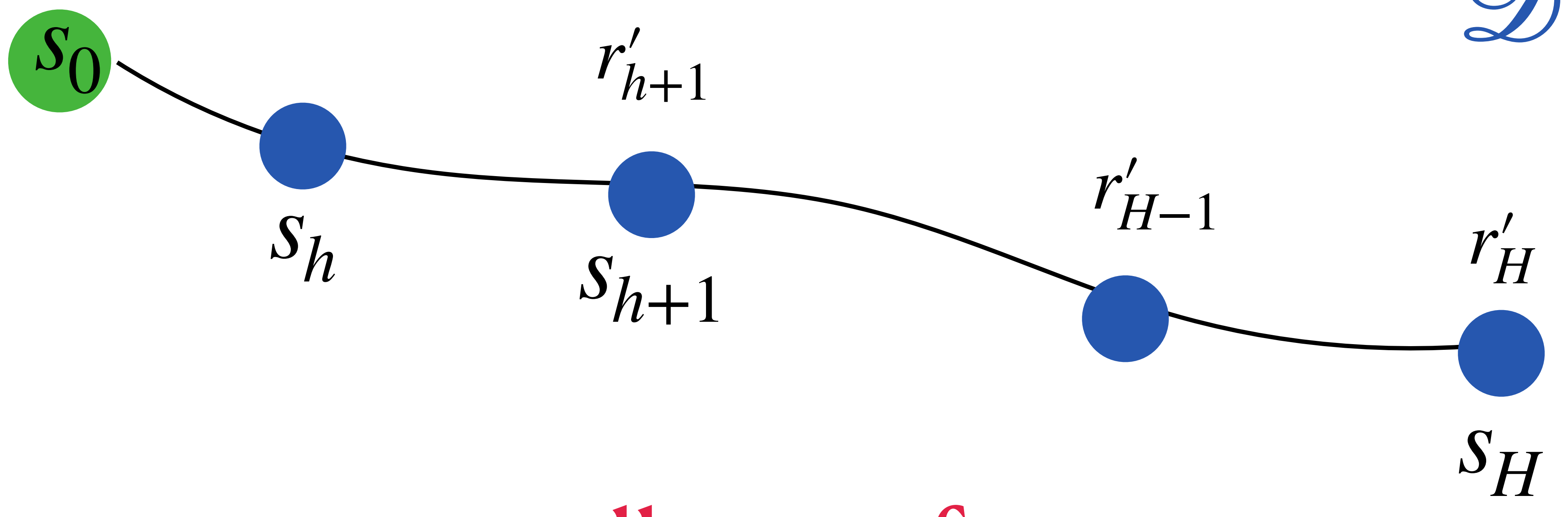
3. Policy update using $\mathscr{D}_{on}$

$$\pi_{t+1} \Leftarrow \text{policy-update}(\mathscr{D}_{on}, \pi_t)$$

What's the key difference to standard PPO

PPO collects online data by always resetting to s_0

1. Sample s_0



2. rollout π_t from s_0

Repeat sampling process n times,

$$\mathcal{D}_{on} = \{\tau^1, \tau^2, \ldots, \tau^n\}$$

3. Policy update using $\mathcal{D}_{on}$

$$\pi_{t+1} \Leftarrow \text{policy-update}(\mathcal{D}_{on}, \pi_t)$$

Task: TL;DR Summarization

Task Statement

Given a reddit post, write a TL;DR (short summary).

[Stiennon et.al, 17]

Dataset Composition

- 210K Prompts total
  - 117K Prompts with *Human written summaries*
  - 93K Prompts with *Human Preference Labels*

Performance again human

(Policy: 7B Pythia model + RoLA)

Algorithms	TL;DR Summarization					
	Win Rate (↑)	RM Score (↑)	$KL(\pi \| \| \pi_{ref})$ (↓)	Rouge 1 (↑)	Rouge 2 (↑)	RougeL (↑)
SFT	$31.6 \pm 0.2\%$	-0.51 ± 0.04	-	32.17 ± 1.01	12.27 ± 0.67	24.87 ± 1.22
DPO	$52.6 \pm 0.4\%$	-	37.33 ± 2.01	30.03 ± 3.23	7.93 ± 1.02	22.05 ± 0.83
PPO	$62.3 \pm 2.5\%$	1.17 ± 0.13	$\mathbf{16.32 \pm 1.46}$	$\mathbf{33.73 \pm 2.34}$	$\mathbf{11.97 \pm 0.91}$	24.97 ± 1.03
DR-PO	$\mathbf{70.2 \pm 1.7\%}$	$\mathbf{1.52 \pm 0.09}$	16.84 ± 0.83	33.68 ± 1.78	11.90 ± 0.06	$\mathbf{25.12 \pm 0.76}$

Message: DR-PO outperforms PPO *at no extra cost of computation or memory*

Would using offline data make DR-PO overfit?

Zero-shot transfer: evaluate trained models directly on CNN Daily mail news articles

Message 2: DR-PO's zero-shot > supervised learning model trained on CNN DM

Algorithms	CNN/DM Summarization			
	Win Rate (↑)	Rouge 1 (↑)	Rouge 2 (↑)	RougeL (↑)
SFT (CNN/DM)	10.5%	25.60	12.27	19.99
DPO	6.0%	20.71	9.47	15.70
PPO	8.5%	23.62	12.29	18.56
DR-PO	**12.0%**	**29.53**	**15.36**	**22.88**

Message 1: DR-PO > PPO

Further reading:

Dataset Reset Policy Optimization for RLHF

Jonathan D. Chang[*]
Department of Computer Science
Cornell University
jdc396@cornell.edu

Wenhao Zhan[*]
Department of Electrical and Computer Engineering
Princeton University
wenhao.zhan@princeton.edu

Owen Oertell
Department of Computer Science
Cornell University
ojo2@cornell.edu

Kianté Brantley
Department of Computer Science
Cornell University
kdb82@cornell.edu

Dipendra Misra
Microsoft Research New York
dimisra@microsoft.com

Jason D. Lee
Department of Electrical and Computer Engineering
Princeton University
jasonlee@princeton.edu

Wen Sun
Department of Computer Science
Cornell University
ws455@cornell.edu

https://arxiv.org/abs/2404.08495

Summary

1. Offline data can boost RL performance

2. Two approaches for taking advantage of offline data:

- Mixing offline data into a replay buffer (e.g., Hybird Q-learning)
- Resetting to the offline data in policy optimization (e.g., DR-PO)