Building **Autonomous** Reinforcement Learning Agents 🤖

Archit Sharma
IRIS Lab @ Stanford

CS224R: Deep Reinforcement Learning
May 22, 2023

Plan for Today

- Why aren't robots autonomous already?
- Defining the problem: **autonomous RL**
- Developing the algorithms
    - Forward-backward RL, MEDAL
    - QWALE/single-life RL

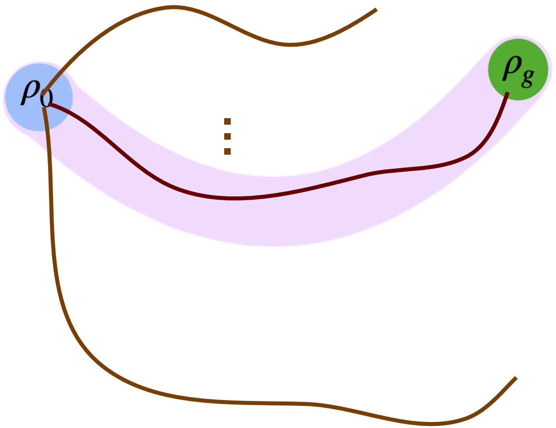
Goal: Build autonomous agents that can learn in and interact with the real world

Plan for Today

- Why aren't robots autonomous already?
- Defining the problem: autonomous RL
- Developing the algorithms
    - Forward-backward RL, MEDAL
    - QWALE/single-life RL

Reinforcement Learning = Trial-and-Error

Learn a policy π to go from ρ_0 to ρ_g



Repeat:

- Execute actions from the policy π
- Observe data from the environment
- Update the policy π

Standard Reinforcement Learning

$$s_0, a_0, s_1, a_1 \ldots s_H$$

$$s'_0, a'_0, s'_1, a'_1 \ldots$$

How does this happen?

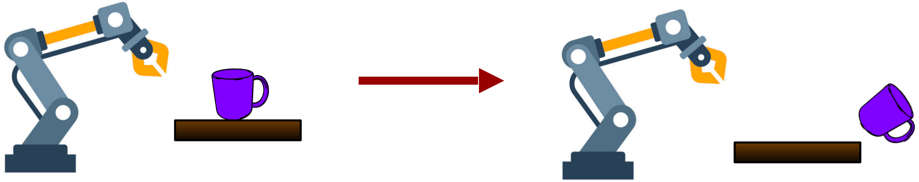
```
import gym
env = gym.make("CartPole-v1")
observation = env.reset()
for _ in range(1000):
  env.render()
  action = env.action_space.sample() # y
  observation, reward, done, info = env.

  if done:
    observation = env.reset()
env.close()
```

[Code snippet from https://gym.openai.com/]

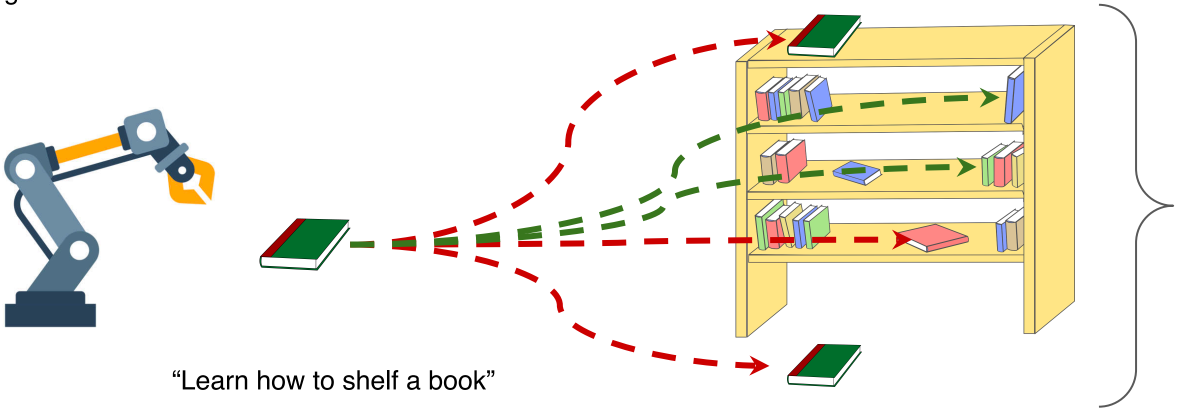
Only in simulation!

The Continual Real World

“Navigate to the basketball court”

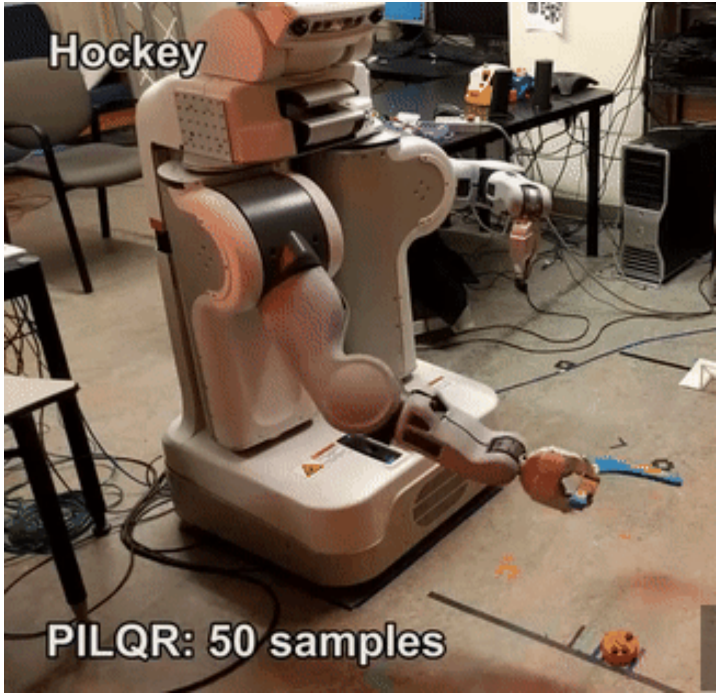


“Grasp the mug”

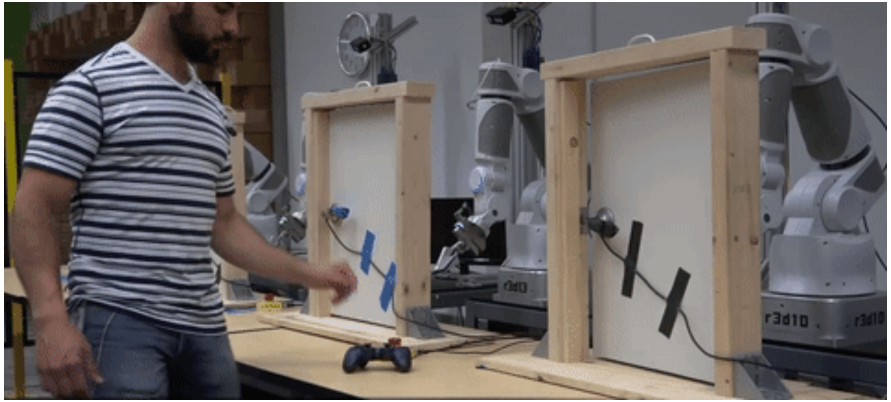


“Learn how to shelf a book”

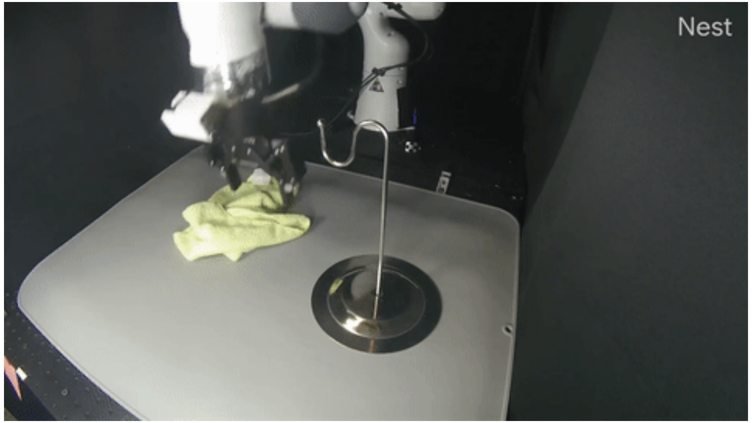
Several thousands of trials!



[Combining model-based and model-free updates for trajectory-centric reinforcement learning, Chebotar et al. 2017]



[Collective Robot Reinforcement Learning with Distributed Asynchronous Guided Policy Search, Yahya et al. 2016]



[Self-Improving Robots: End-to-End Autonomous Visuomotor Reinforcement Learning, Sharma et al. 2023]

Plan for Today

- Why aren't robots autonomous already?
- Defining the problem: autonomous RL
- Developing the algorithms
    - Forward-backward RL, MEDAL
    - QWALE/single-life RL

$$s_0, a_0, s_1, a_1 \ldots s_H$$

Reset Environment

$$s'_0, a'_0, s'_1, a'_1 \ldots$$

🚩 Problem: this requires human supervision

What if we increase *H*?

✔ Fewer environment resets, less human supervision

Autonomous RL: Definition

Agent interacts autonomously

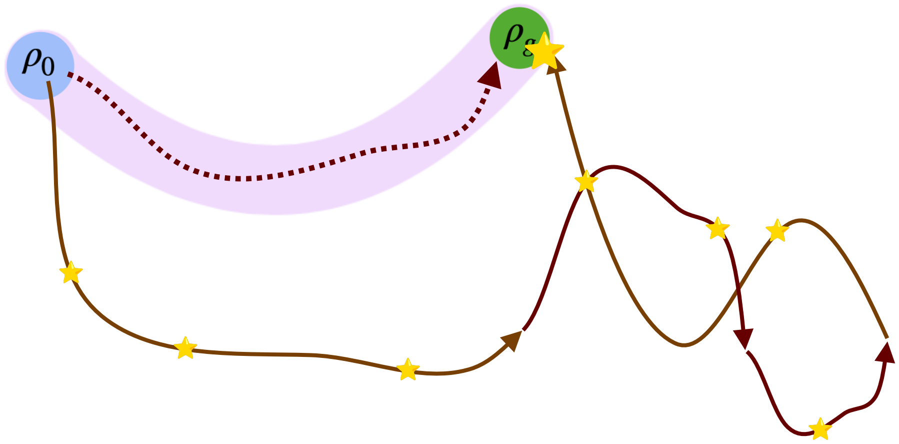
$$\boxed{s_0}, a_0, s_1, a_1 \ldots$$

No environment resets*

Initialize *once* at the beginning

*can relax this constraint to reset occasionally, or at low frequency

Autonomous RL: Evaluation



We might care about two different things:

- The amount of reward recovered over the course of its life (ex: mars rover)
- The quality of the policy learned (ex: robot chef)

Autonomous RL: Evaluation

Deployed Policy Evaluation

= Quality of the policy learned

$$J(\pi_t) = \mathbb{E}_{s_0 \sim \rho_0, a_t \sim \pi_t}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$$

Start from the initial state distribution

Take actions according to π_t

Total reward over the episode

Continuing Policy Evaluation*

= Reward accumulated over lifetime

$$\lim_{h \to \infty} \mathbb{E}\left[\frac{1}{h}\sum_{t=0}^{h} r(s_t, a_t)\right]$$

Average over reward accumulated in the lifetime

We'll take a look at algorithms for both!

*average-reward RL

Why is autonomous RL important?

Robotics < — > Autonomy

- We want robots to operate with minimal human supervision ✓
- We want robots to *train* with minimal human supervision ✓

Autonomy is important to build generalist robots

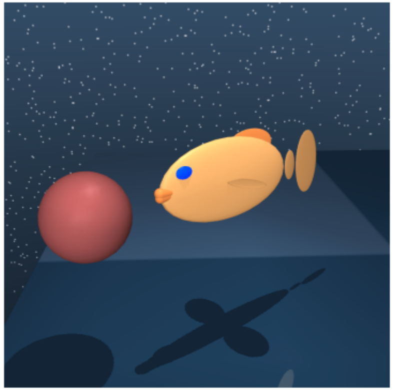
- generalization requires data
- robot interaction data is bottlenecked by human supervision
- less supervision => more data => better generalization?

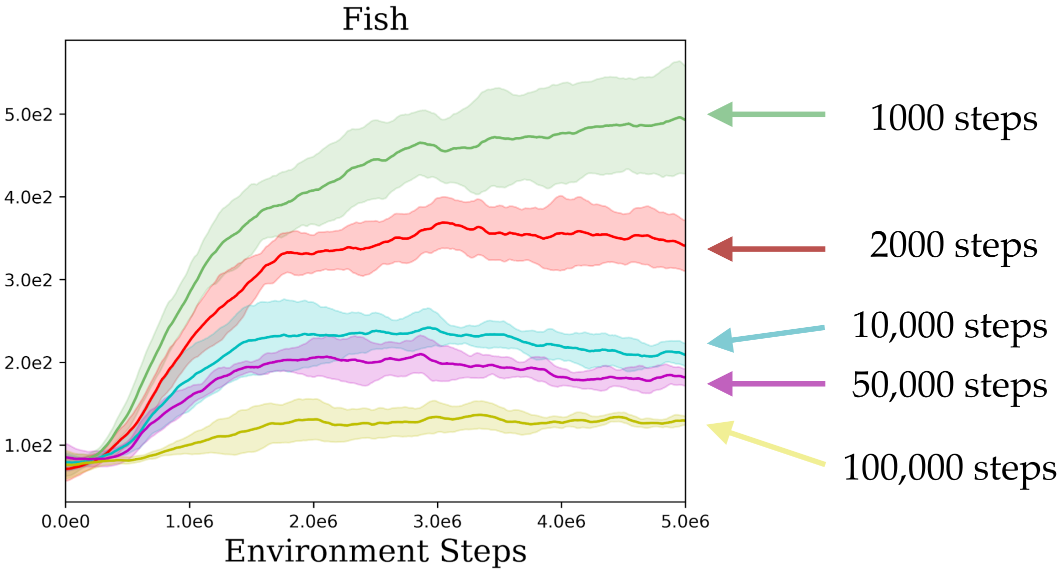
Plan for Today

- Why aren't robots autonomous already?
- Defining the problem: autonomous RL
- Developing the algorithms
  - Forward-backward RL, MEDAL
  - QWALE/single-life RL

Standard RL Algorithms Fail Autonomously

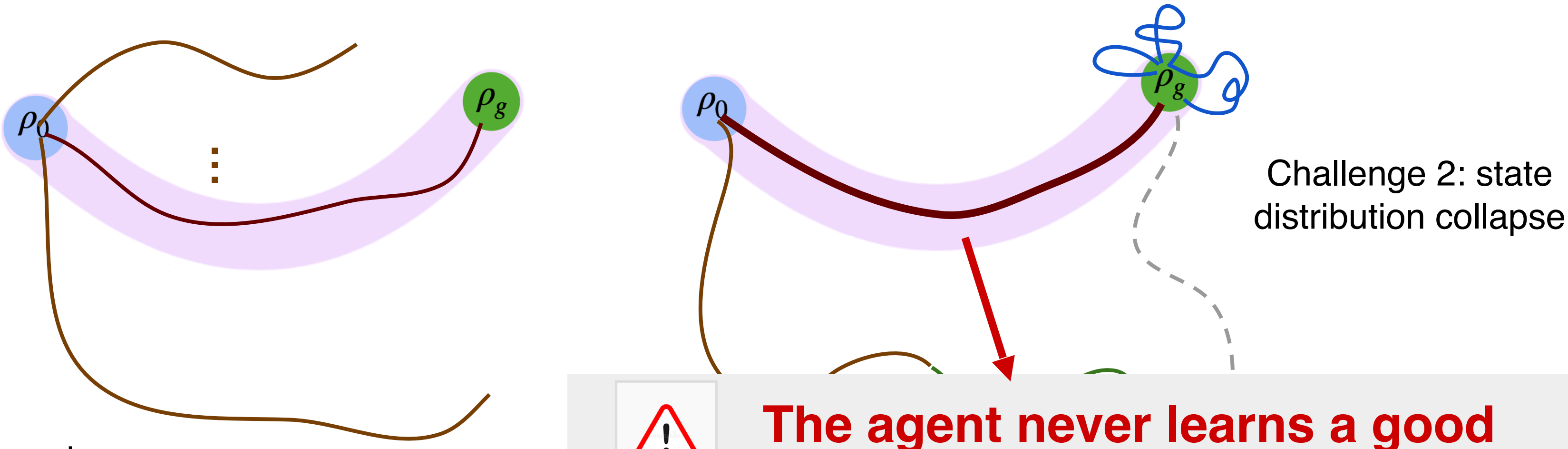
What happens when the episode length is increased?





Note: this measures **deployed policy evaluation**

The Challenge of Learning Autonomously

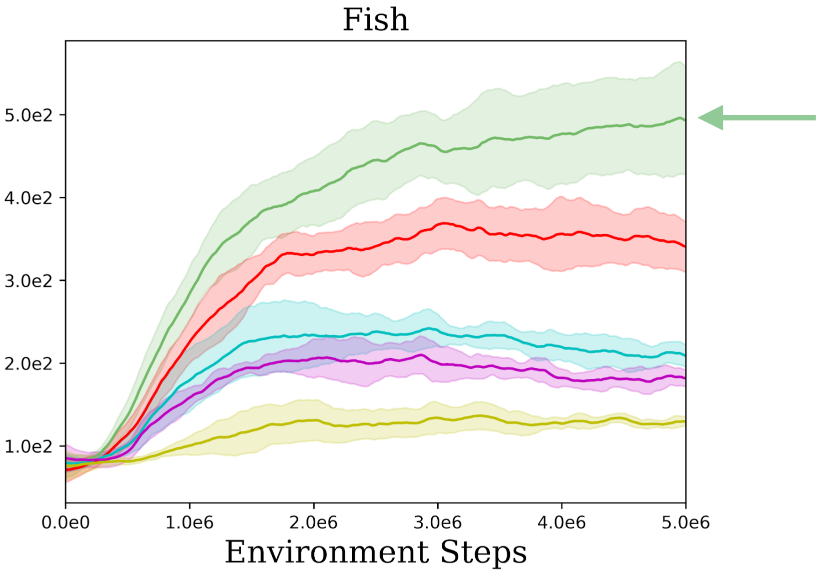


Can always retry the task from initial state distribution

Ch
can cause the agent to drift far away

⚠ **The agent never learns a good policy**

Learn a Backward Policy!

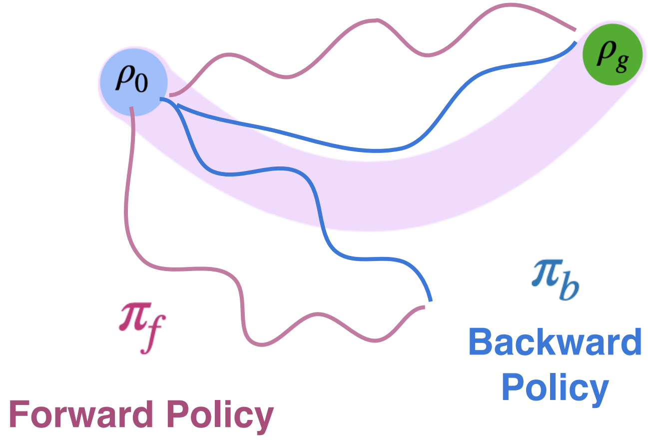


learns successfully if allowed to retry from the *initial state distribution frequently*

key idea: learn a policy to reset?

Algorithm: Forward-Backward RL

Assume we have $r_f(s,a)$ to reach ρ_g
and $r_b(s,a)$ to reach ρ_0



Forward-Backward RL

1. Initialize forward policy π_f and backward policy π_b
2. Rollout π_f for H steps (+ *update* π_f on $r_f(s,a)$)
3. Rollout π_b for H steps (+ *update* π_b on $r_b(s,a)$)
4. Rollout π_f for H steps (+ *update* π_f on $r_f(s,a)$)
5. Rollout π_b for H steps (+ *update* π_b on $r_b(s,a)$)

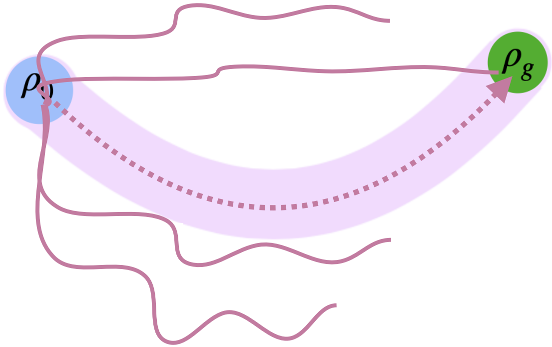
(repeat…)

- requires an additional $r_b(s,a)$
+ simple to train

[Han et al. Learning Compound Multi-Step Controllers under Unknown Dynamics]
[Eysenbach et al. Leave no Trace: Learning to Reset for Safe and Autonomous Reinforcement Learning]

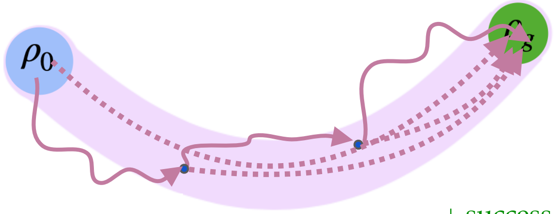
Can we do better?

Consider learning from the forward policy's perspective:



cannot change the initial state distribution for forward policy π_f in *episodic setting*

What if the forward policy can practice from the easier states:



+ success from easier states can make it faster to learn from harder states

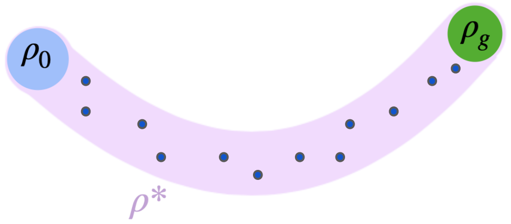
backward policy π_b controls the initial state distribution for forward policy π_f in *autonomous RL*

Matching Expert States

Key insight: train backward policy π_b to match expert states ρ^*

How? Minimize $\mathbb{D}_{\mathrm{JS}}(\rho^b(s) \,||\, \rho^*(s))$

State distribution of the backward policy π_b



we want to initialize forward policy π_f at states an *optimal policy would visit* (ρ^*) [1]

Problem: we don't have either distribution

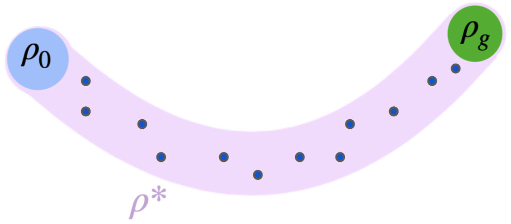
Assume we have access to a (small) set of demonstrations

expert demonstrations

+ we can sample distributions now
(rolling out π_b is approximately sampling ρ^b)

[1] Kakade & Langford. *Approximately Optimal Approximate Reinforcement Learning*. ICML 2002.

Matching Expert States



we want to initialize forward policy π_f at states an *optimal policy would visit* (ρ^*)

Key insight: train backward policy π_b to match expert states ρ^*

How? Minimize $\mathbb{D}_{\text{JS}}(\rho^b(s) \,||\, \rho^*(s))$

+ we can sample distributions now
(rolling out π_b is approximately sampling ρ^b)

How do we train π_b? *Train a classifier as a reward function!*
(already seen this in "Learning Rewards")

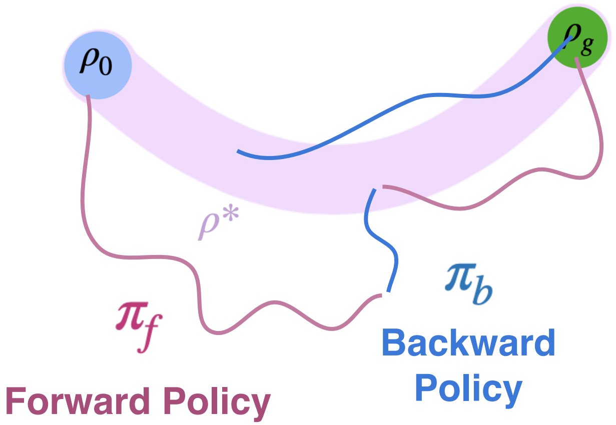
$$C(s) = \begin{cases} +1 & s \in \text{demos}, \\ -1 & s \sim \rho^b(\cdot) \end{cases}$$

forward policy π_f: $\max \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_f(s_t, a_t)\right]$

backward policy π_b: $\max -\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \log(1 - C(s_{t+1}))\right]$

Algorithm: MEDAL

Assume we have $r_f(s,a)$ to reach ρ_g
and expert demos ρ^*



Matching Expert Distributions for Autonomous Learning

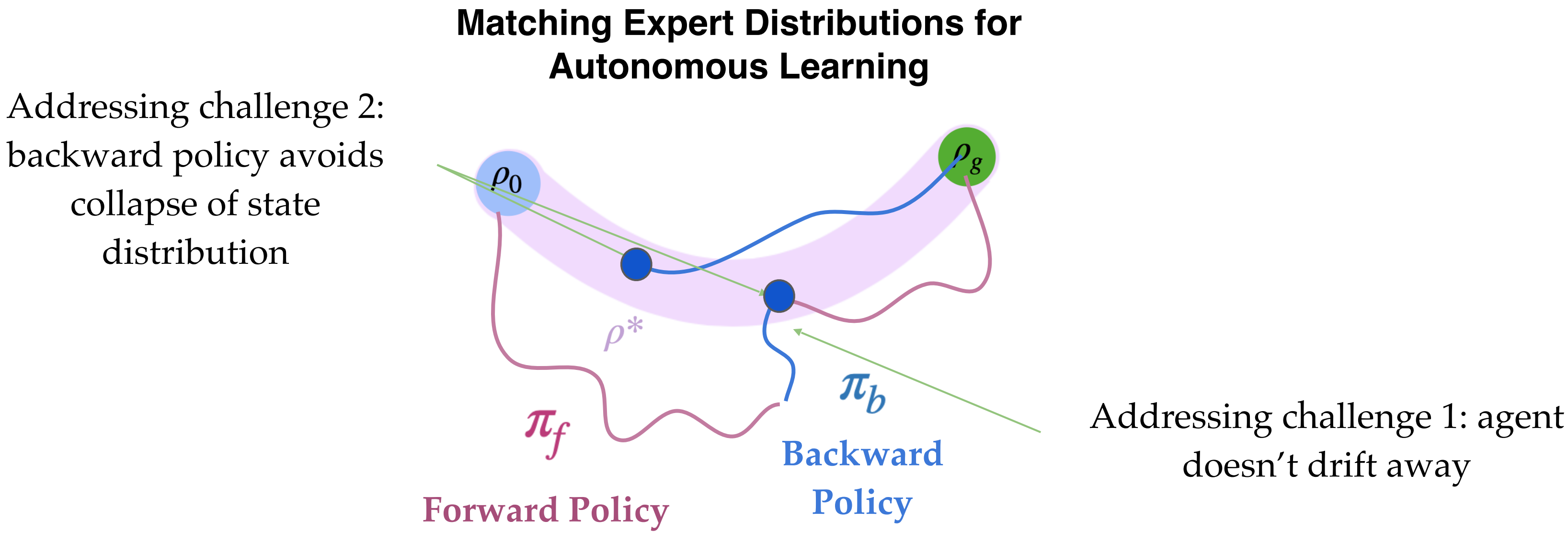
1. Initialize forward policy π_f and backward policy π_b
2. Rollout π_f for H steps (+ *update* π_f on $r_f(s,a)$)
3. Rollout π_b for H steps (+ *update* π_b on $\mathbb{D}_{JS}(\rho_b || \rho^*)$)
4. Rollout π_f for H steps (+ *update* π_f on $r_f(s,a)$)
5. Rollout π_b for H steps (+ *update* π_b on $\mathbb{D}_{JS}(\rho_b || \rho^*)$)

(repeat…)

- requires expert demos
- adversarial training can be tricky

\+ can be more efficient
\+ no additional reward functions

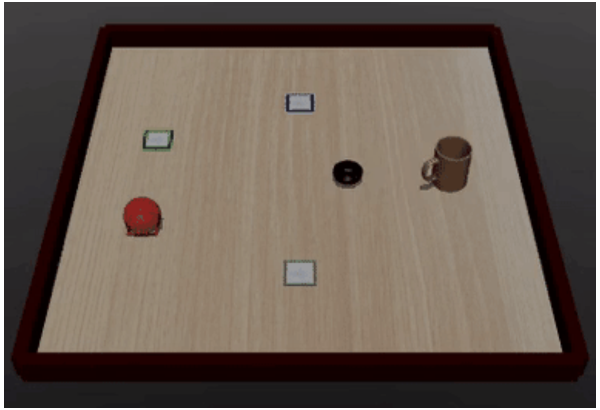
Autonomous Reinforcement Learning via MEDAL

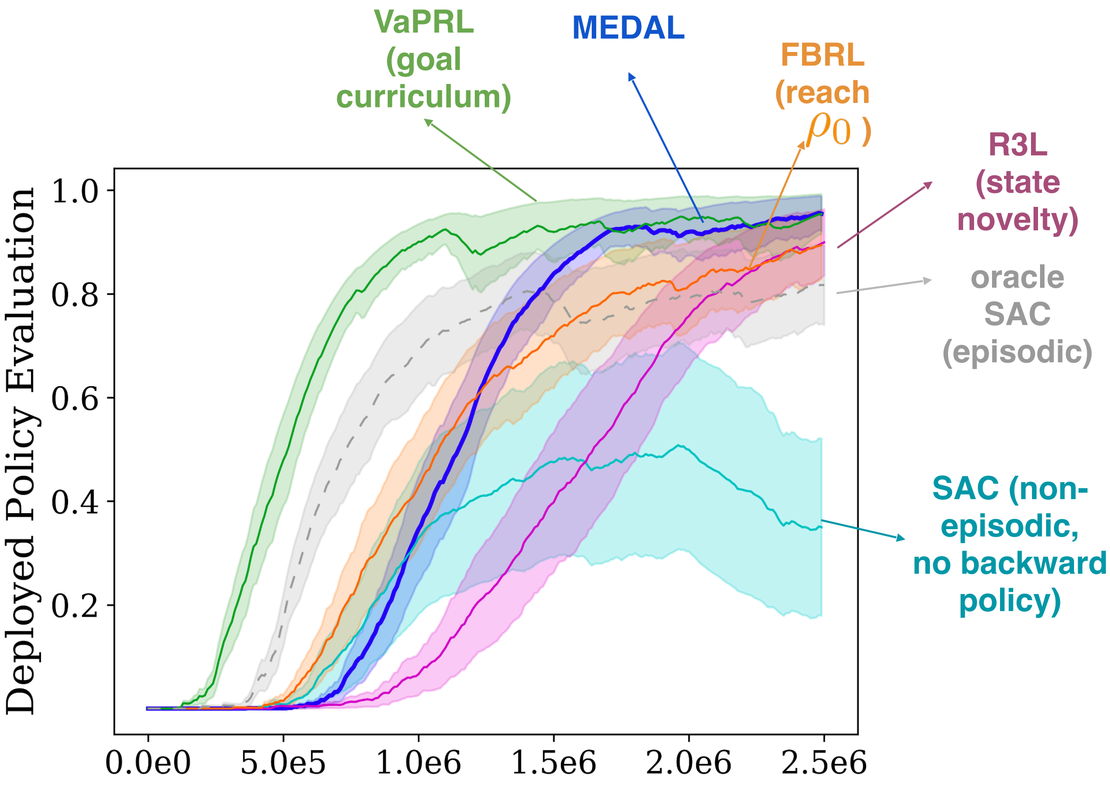


Pro: Forward policy tries the task from wide set of initial states, both easy and hard, improving the sample efficiency [1]

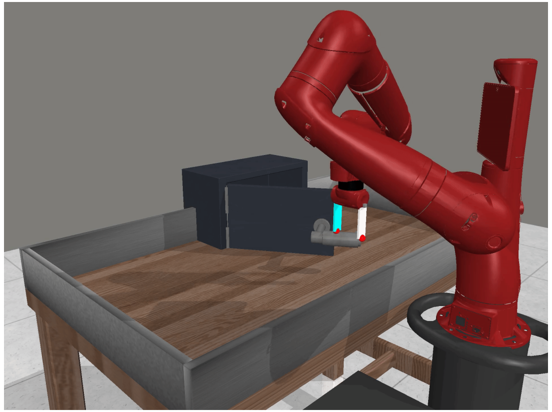
Results

EARL Benchmark
Training: reset every 200k steps
Evaluation: policy performance from ρ_0

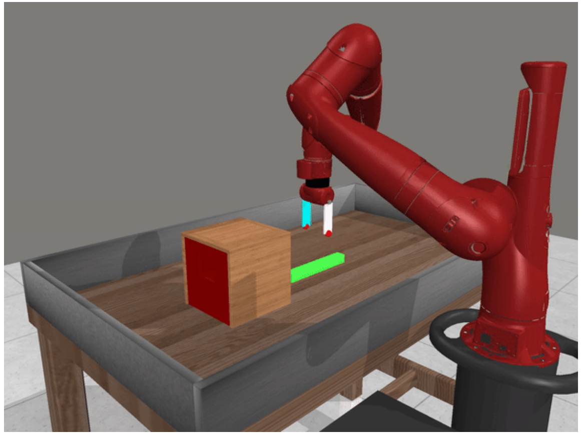




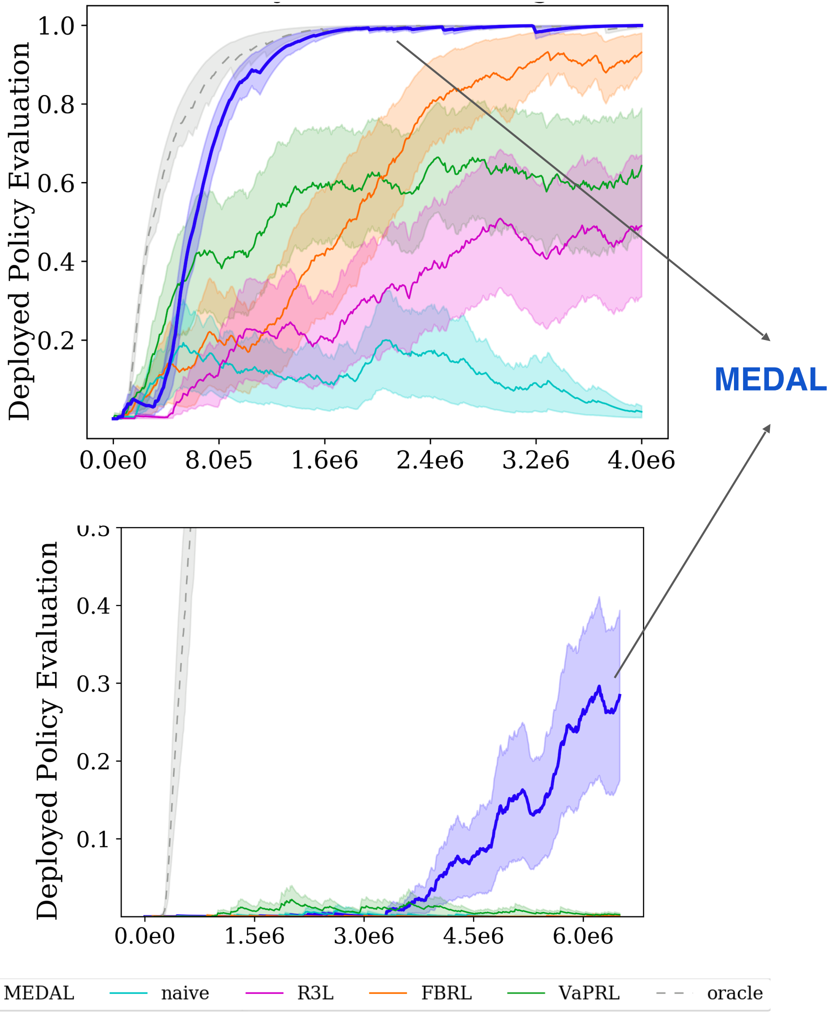
EARL: Sharma*, Xu* et al. Autonomous Reinforcement Learning: Formalism and Benchmarking, ICLR 2022.
VaPRL: Sharma et al. *Autonomous Reinforcement Learning via Subgoal Curricula*. NeurIPS 2021.
FBRL: Han et al. Learning Compound Multi-Step Controllers under Unknown Dynamics. IROS 2015.
R3L: Zhu et al. The Ingredients of Real-World Robotic Reinforcement Learning. ICLR 2020.



Door Closing



Peg Insertion

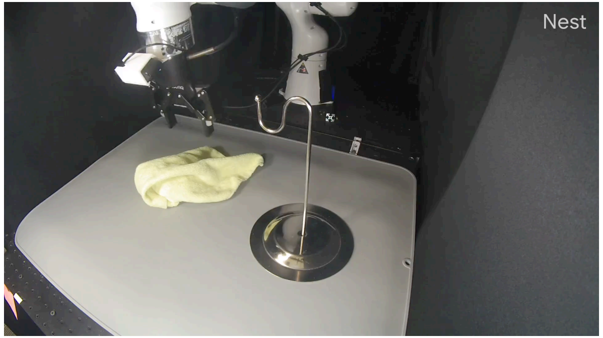


Putting it on the real world: MEDAL++

Challenge: we don't have a reward function $r_f(s, a)$ for the forward policy

Solution: *use classifier-based rewards again!*

Results:



Forward and backward policies in MEDAL++ learning to put a cloth through the hook

Timelapse of MEDAL++ on other tasks

[1] Sharma et al. Self-Improving Robots: End-to-End Autonomous Visuomotor Reinforcement Learning

Plan for Today

- Why aren't robots autonomous already?
- Defining the problem: autonomous RL
- Developing the algorithms
    - Forward-backward RL, MEDAL
    - QWALE/single-life RL

So far…

We Looked at FBRL, MEDAL

- We can improve the quality of the policy through *autonomous* practice

What happens after we deploy the policy?

Obviously, everything works perfectly and nothing goes wrong ✓

So far…

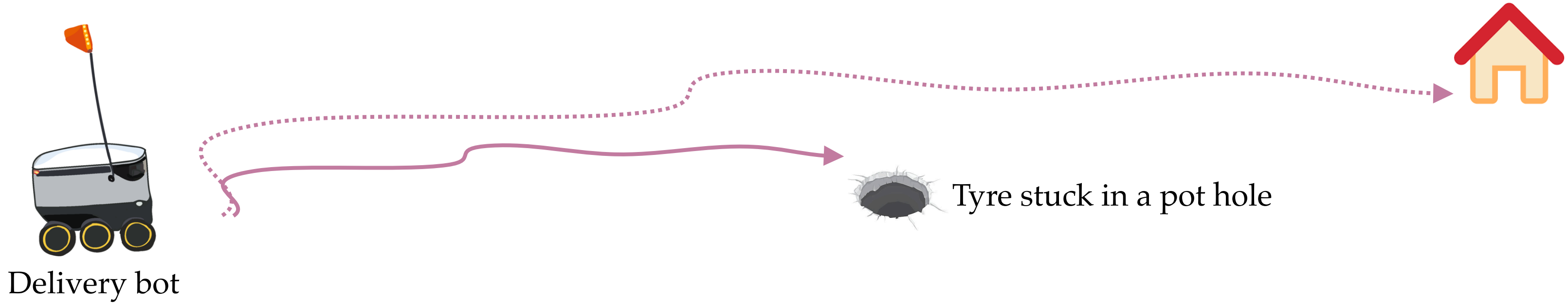
We Looked at FBRL, MEDAL
- We can improve the quality of the policy through *autonomous* practice

What happens after we deploy the policy?

~~Obviously, everything works perfectly and nothing goes wrong~~

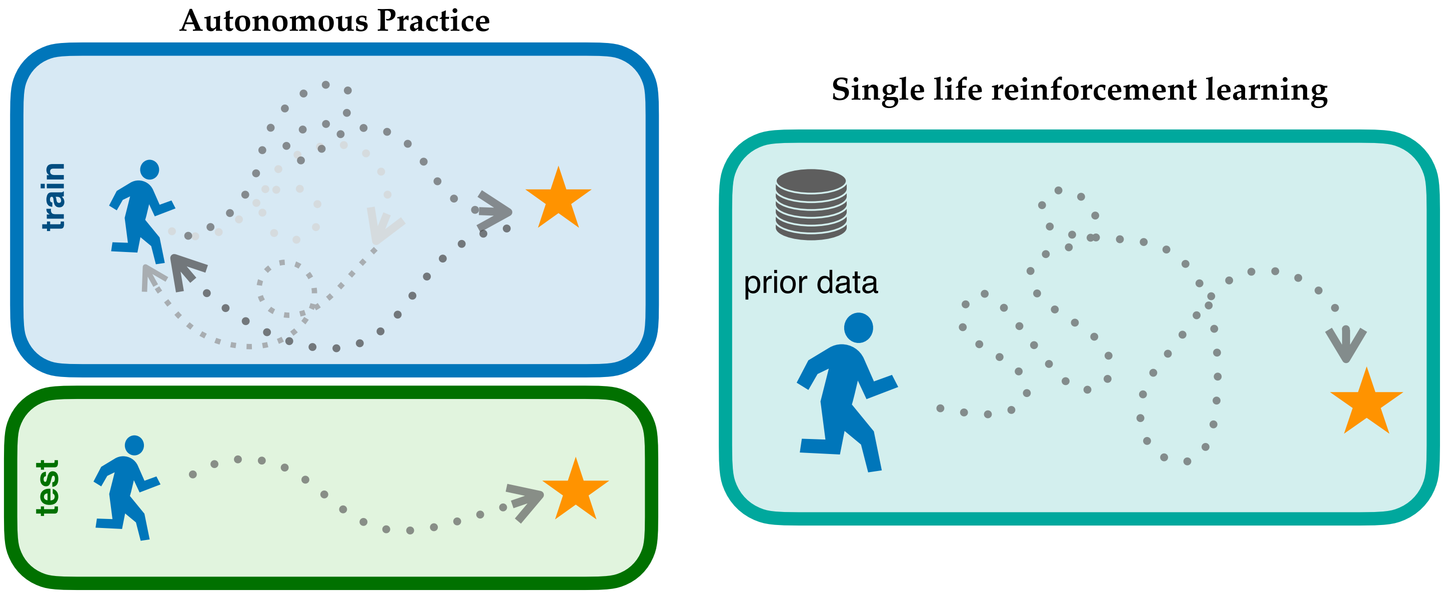
The natural world is complex, and something likely will go wrong, despite preparation :(

Single-life RL



This is a new scenario that has not been seen in training before:
- Finishing the delivery is more important
- Wait for human to intervene?
    - Consider the example of Mars Rover, how would humans even intervene?

Single-life RL



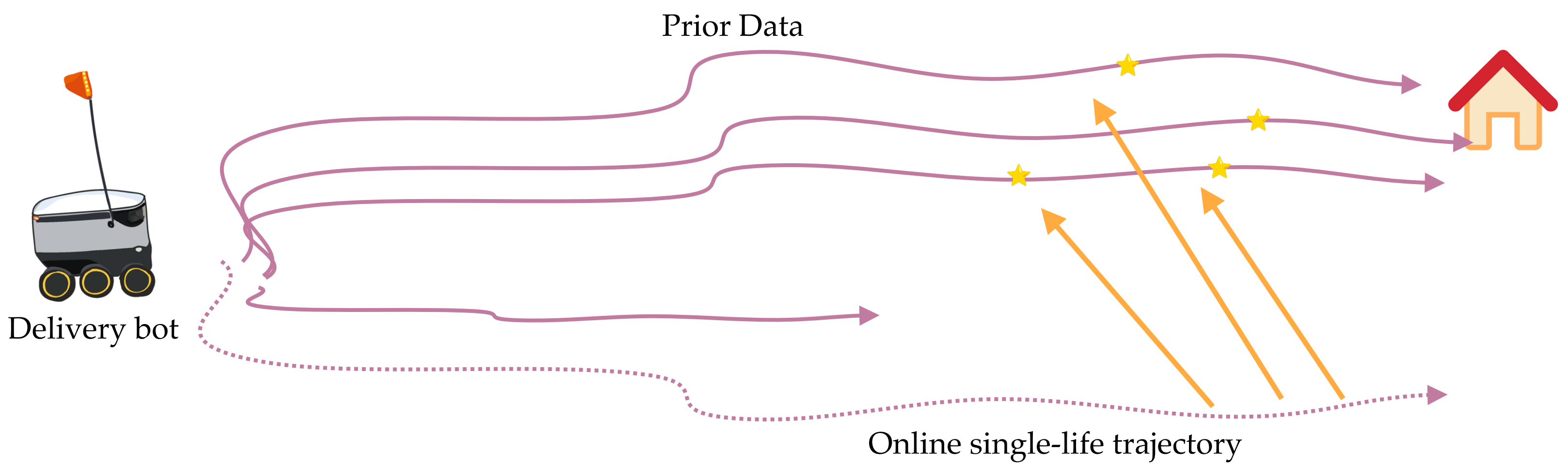
Given prior data, the agent has *one life* to autonomously complete the task in a novel scenario

Challenge: Recovery from Unseen States

This agent was only trained to run, but has not seen hurdles before:

Deploying a policy, and even fine-tuning online does not encourage it to recover once it tumbles

Bias towards Prior Data



When the agent goes reaches an unseen state:
- bias towards states seen in prior data
- but not all states, as the data may have suboptimal states

Q-Weighted Adversarial Learning (QWALE)

Key Idea: Train the agent to stay close to states visited in prior data

Do we know of a technique to reach a state distribution? yes! *Train a reward classifier!*

$$C(s) = \begin{cases} +1 & s \in \text{prior data}, \\ -1 & s \in \text{online data} \end{cases}$$

But, how do differentiate good prior states from bad ones? Weigh all prior states when training the classifier by $\exp(Q(s,a))$ [1]

prior data bias

QWALE: train policy π with the reward: $\underbrace{r(s')}_{\text{task reward}} - \overbrace{\log(1 - C(s'))}^{\text{prior data bias}}$

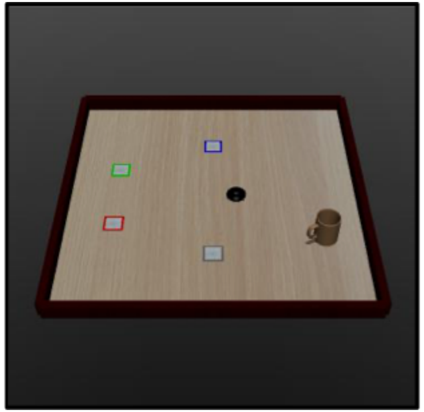
task reward

[1] Chen et al. You Only Live Once: Single-life Reinforcement Learning. NeurIPS, 2022

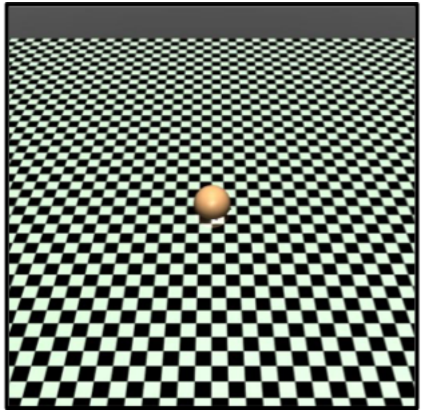
Can QWALE help agents handle novel, out-of-distribution situations?

QWALE helps the agent recover when it falls into out-of-distribution states.

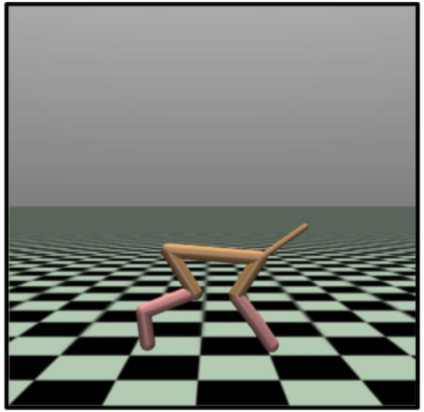
Experimental Domains



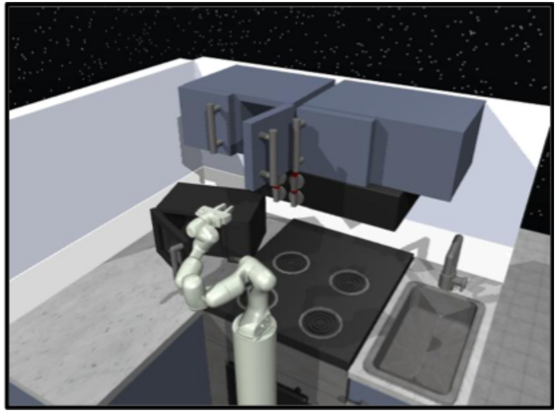
Tabletop-Organization
(+new initial mug pos)



Pointmass
(+wind)

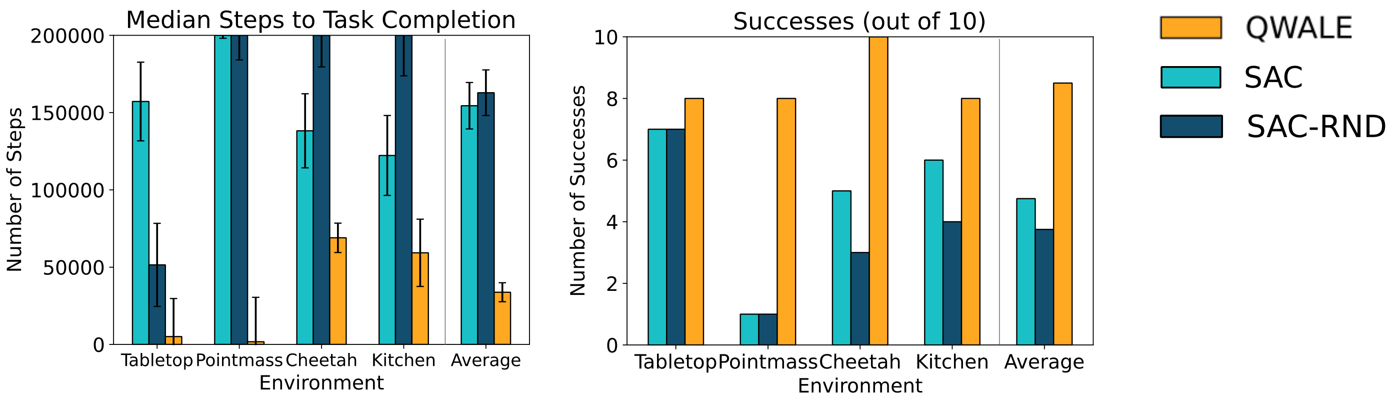


Cheetah
(+hurdles)



Franka-Kitchen
(+new combo of tasks)

How does QWALE compare to RL fine-tuning in SLRL settings?



QWALE significantly outperforms RL fine-tuning.

Summary

- Why aren't robots autonomous already?
- Defining the problem: autonomous RL
- Developing the algorithms
    - Forward-backward RL, MEDAL
    - QWALE/single-life RL

Things we did not cover:
- How to handle irreversible states [1]
- Autonomous agents are taking over the web! [2]

Questions?

[1] Xie* et al. When to Ask for Help: Proactive Interventions in Autonomous Reinforcement Learning
[2] https://github.com/Significant-Gravitas/Auto-GPT