Offline Reinforcement Learning

Sanjiban Choudhury

Cornell Bowers CIS
Computer Science

The story thus far ...

✅ Decision-making

✅ Perception

✅ Models of humans

☐ Practical Robot Learning

Today-> ☐ Offline RL

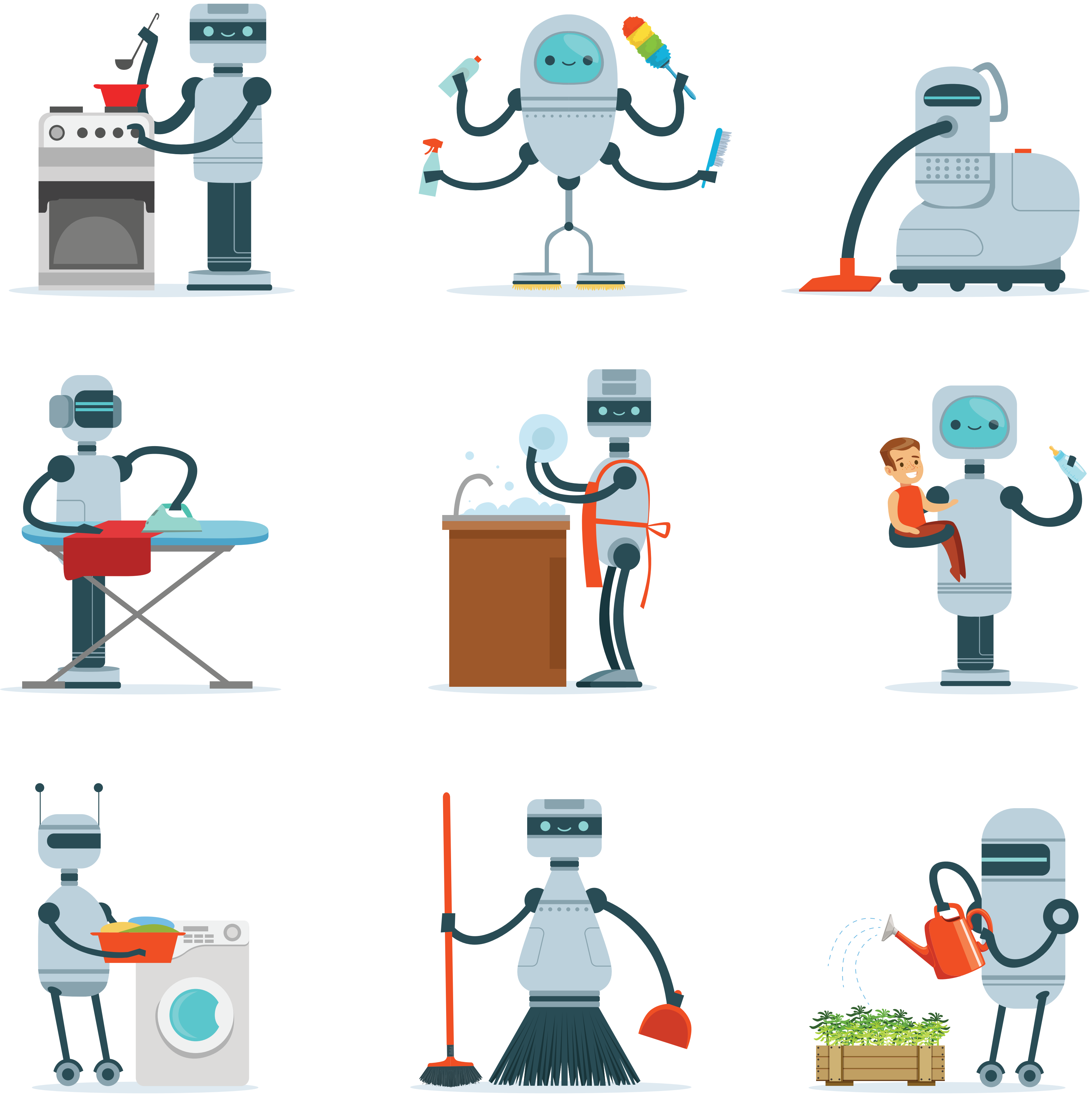
☐ Sim-to-Real

Today's class

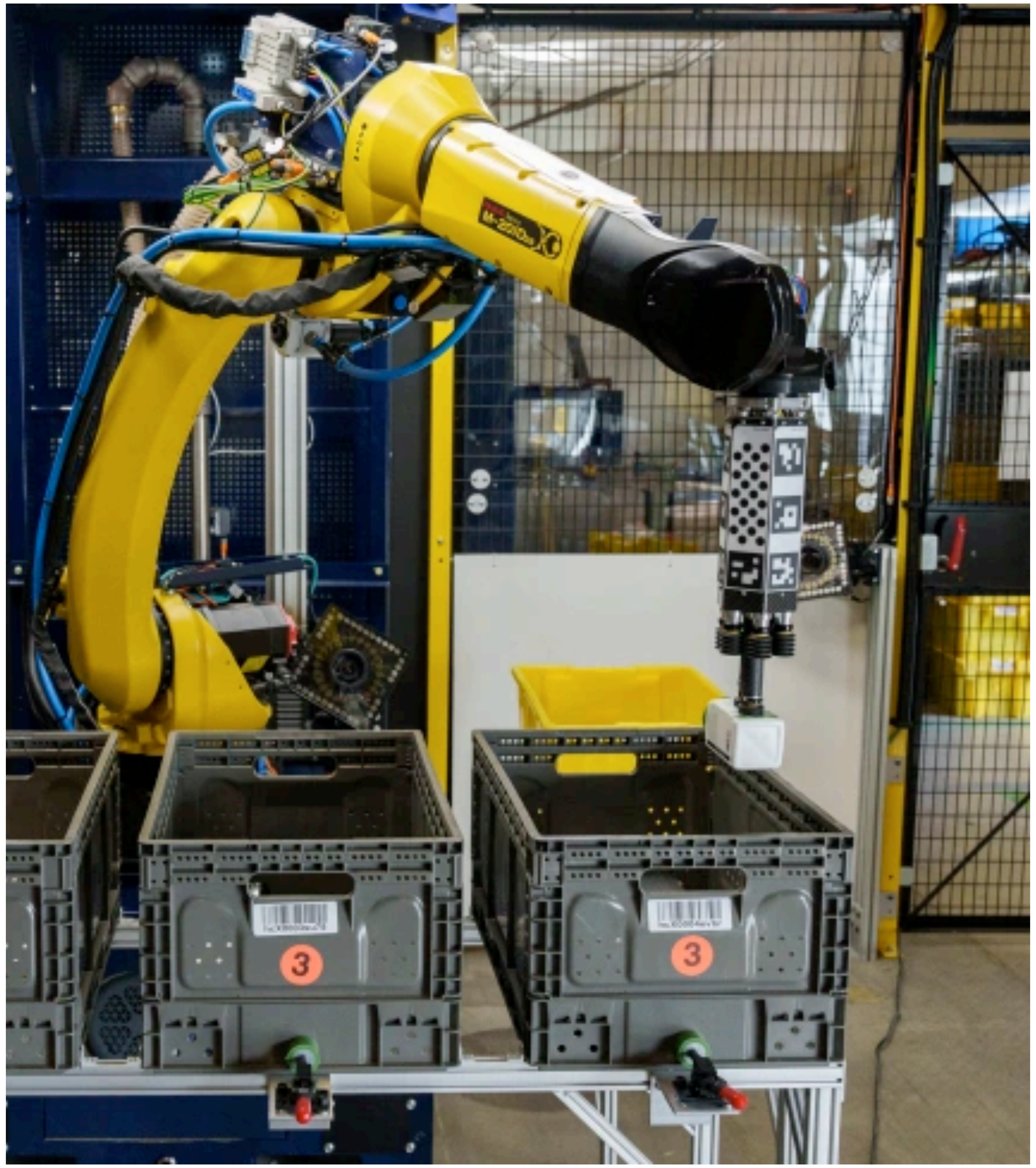
- What is offline RL? Why do we need it for robots?
- Paradigm 1: Offline RL via Pessimism
  - Problem with Q-learning
  - Pessimism to the rescue
- Paradigm 2: RL via Supervised Learning
  - Return-conditioned Supervised Learning
  - Problem in Stochastic MDPs

Why do we need offline RL for robots?

Robots today still only work in CLOSED world

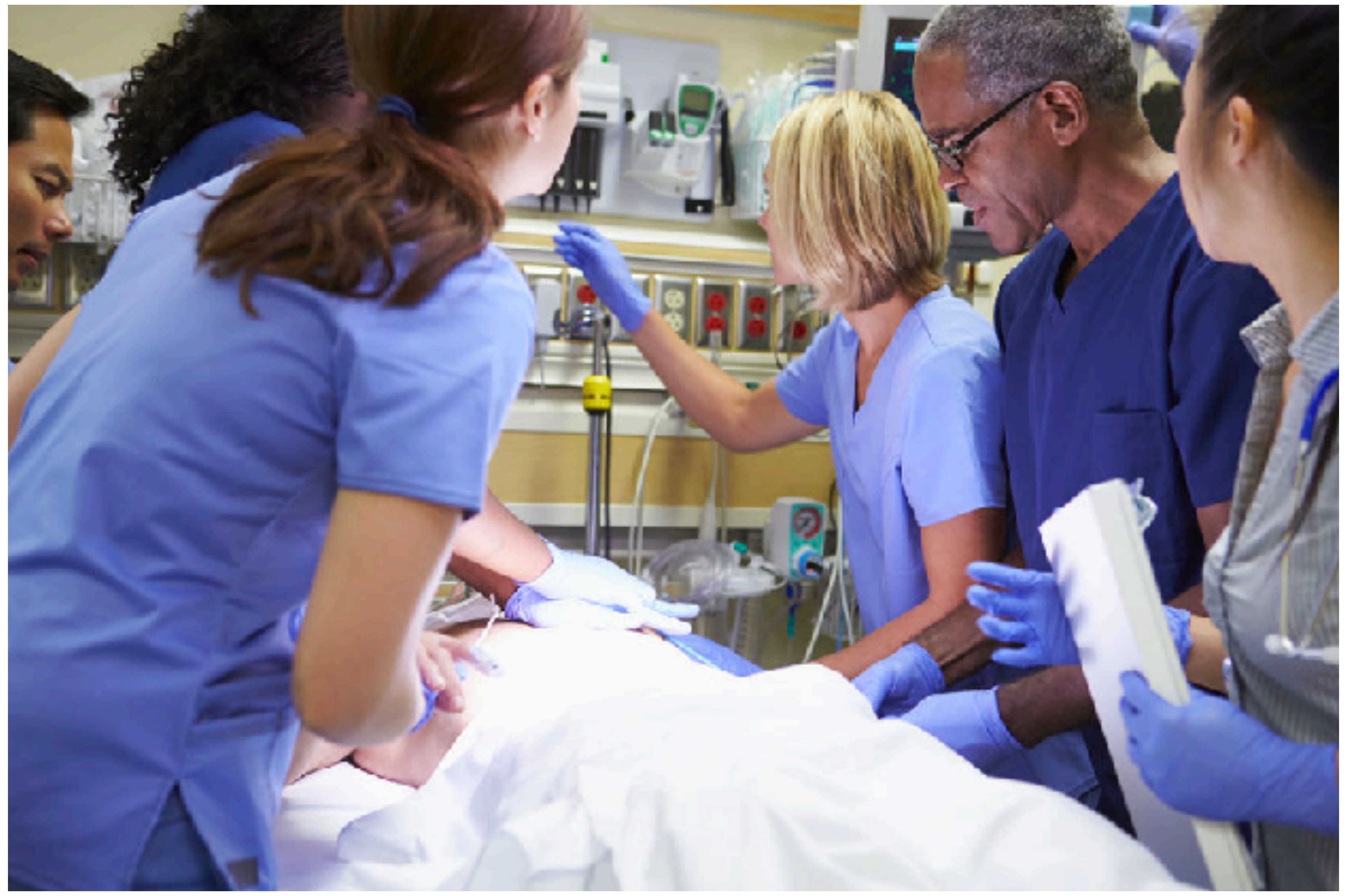


The Dream



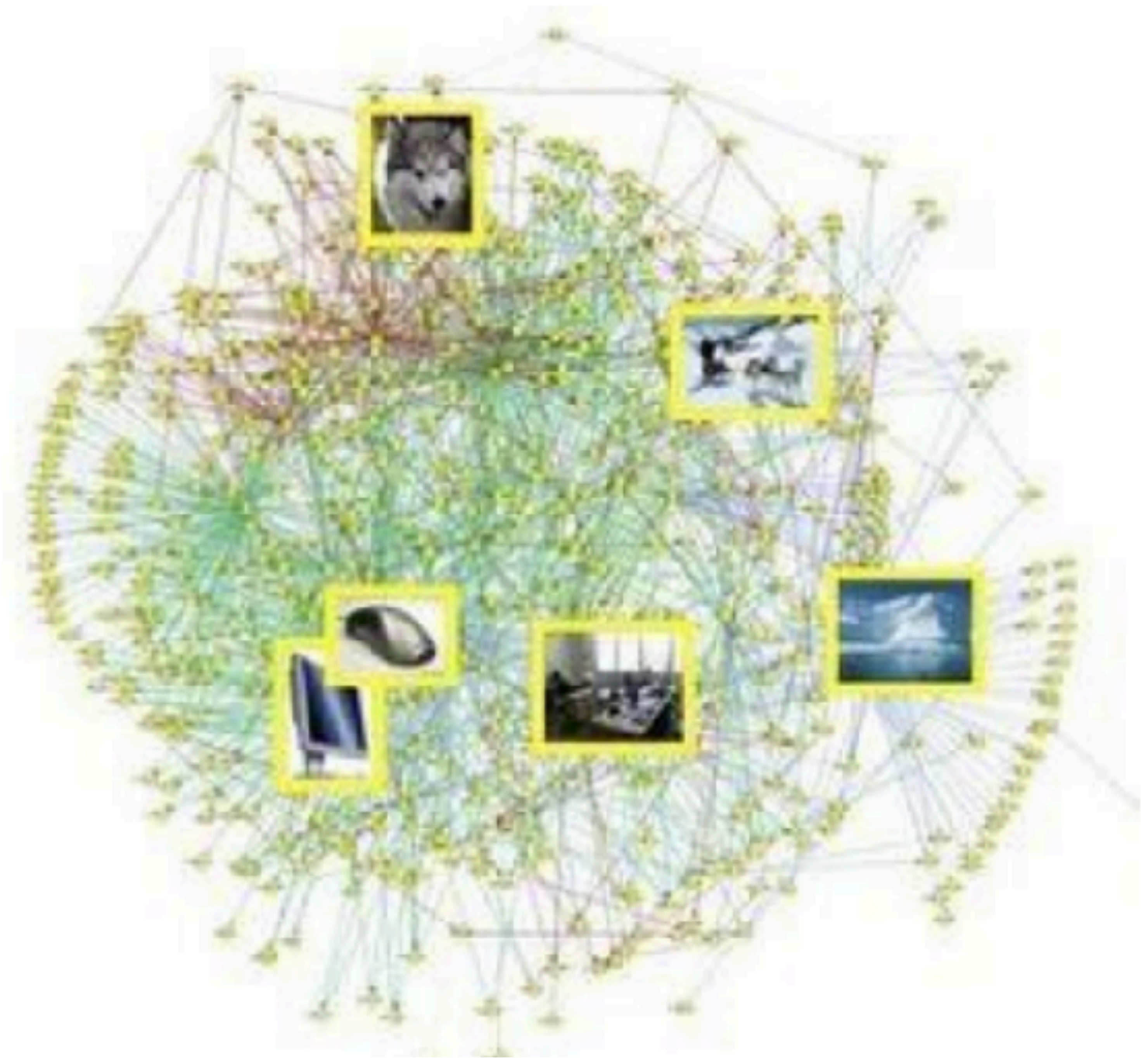
Reality

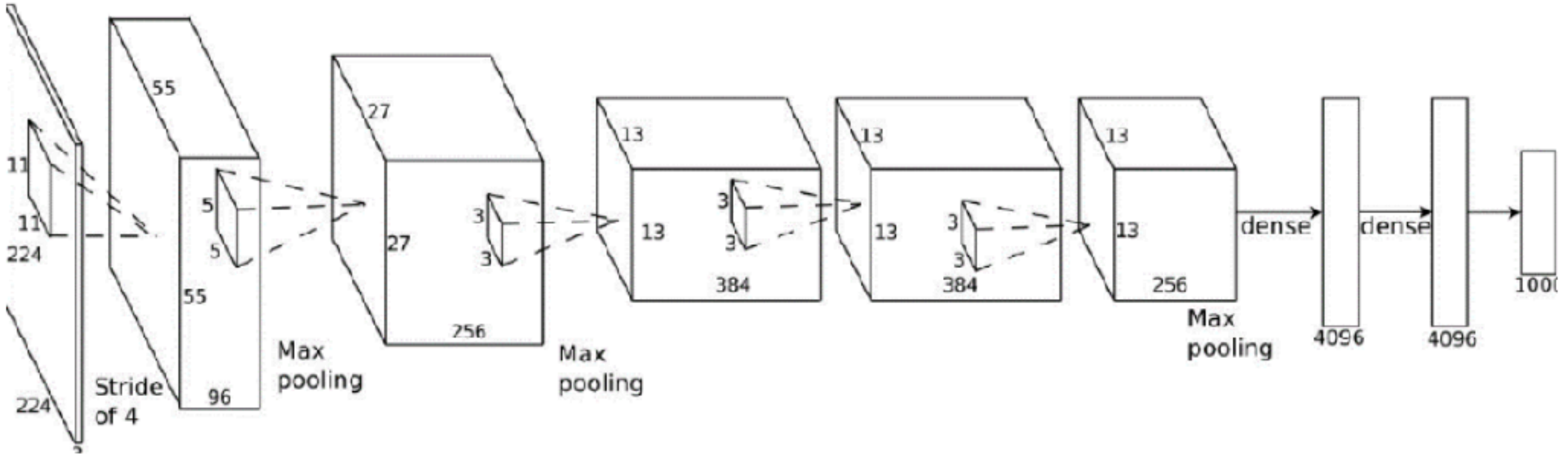
Generalize to variations of the OPEN world?



Why can't we do RL with robots in the real world?

Machine learning's answer!





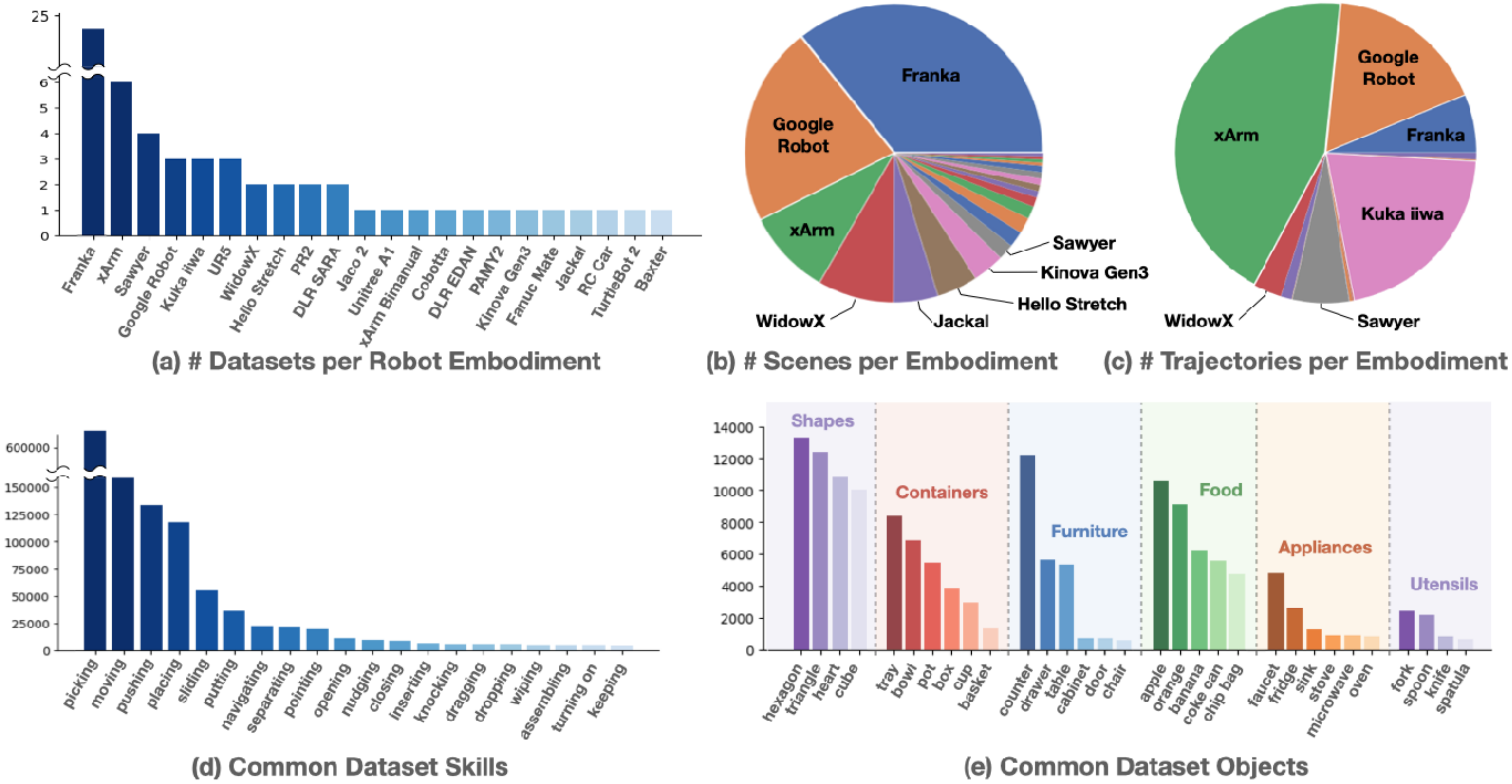
Big Data

Big Models

Credit: **Sergey Levine** "Offline RL lecture"

Efforts underway to scale up robotics data!

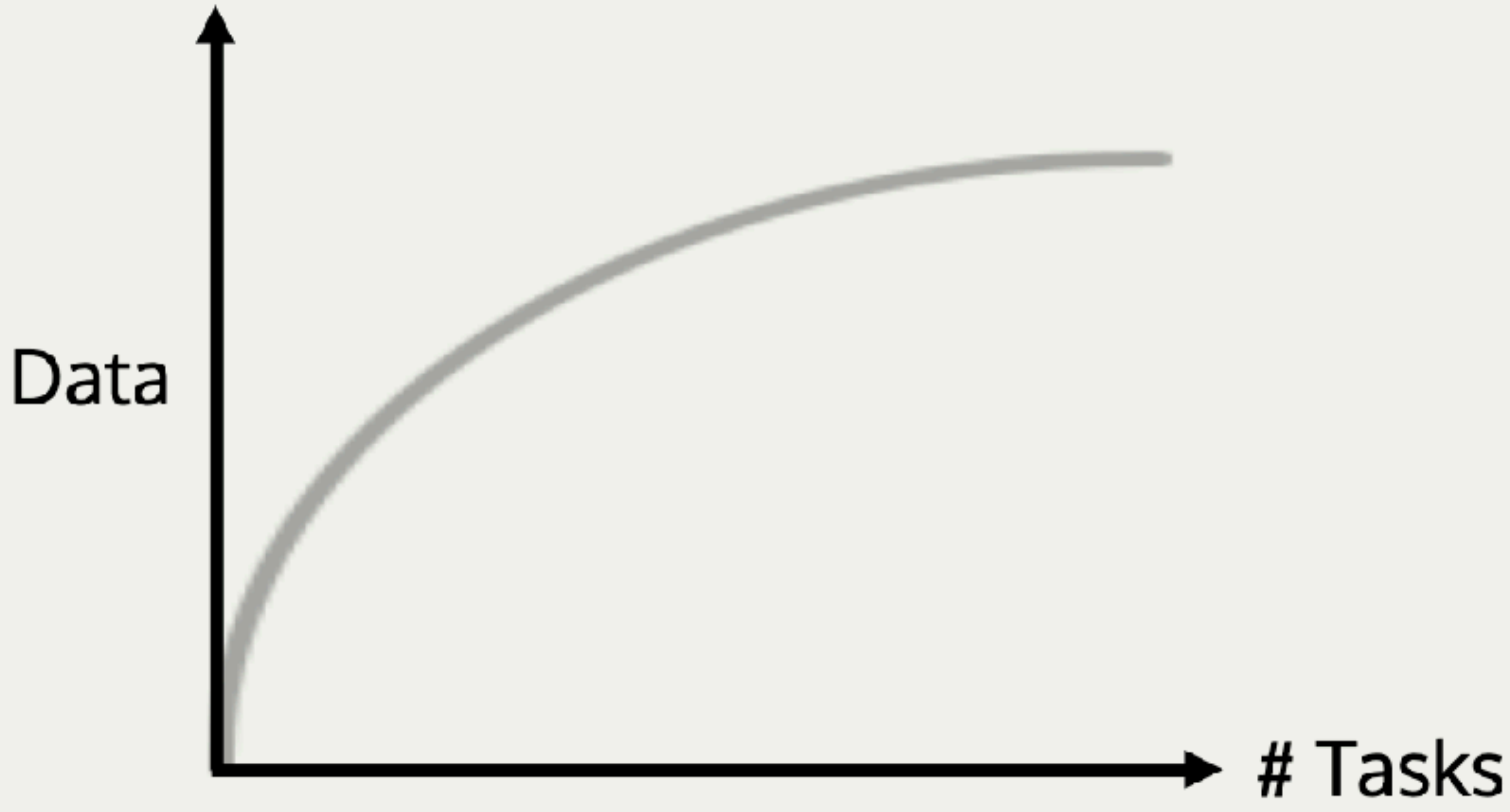
1M trajectories, 22 robots, 21 different institutions



Open-X Embodiment Dataset

Hope: Data grows logarithmically with tasks

On the quest for shared priors
w/ machine learning



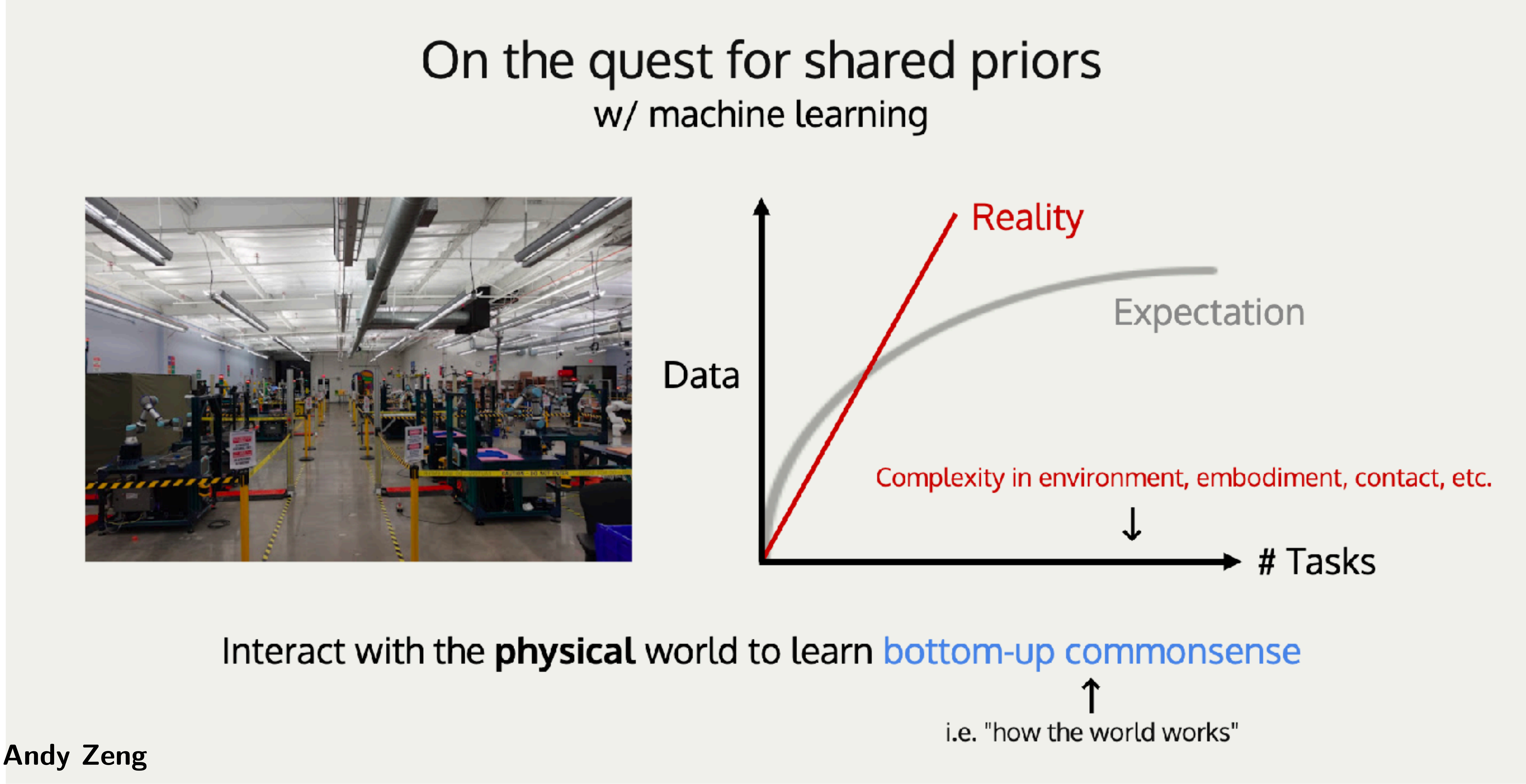
Interact with the **physical** world to learn bottom-up commonsense

↑

i.e. "how the world works"

Credit: **Andy Zeng**

Reality: Data grows linearly with tasks

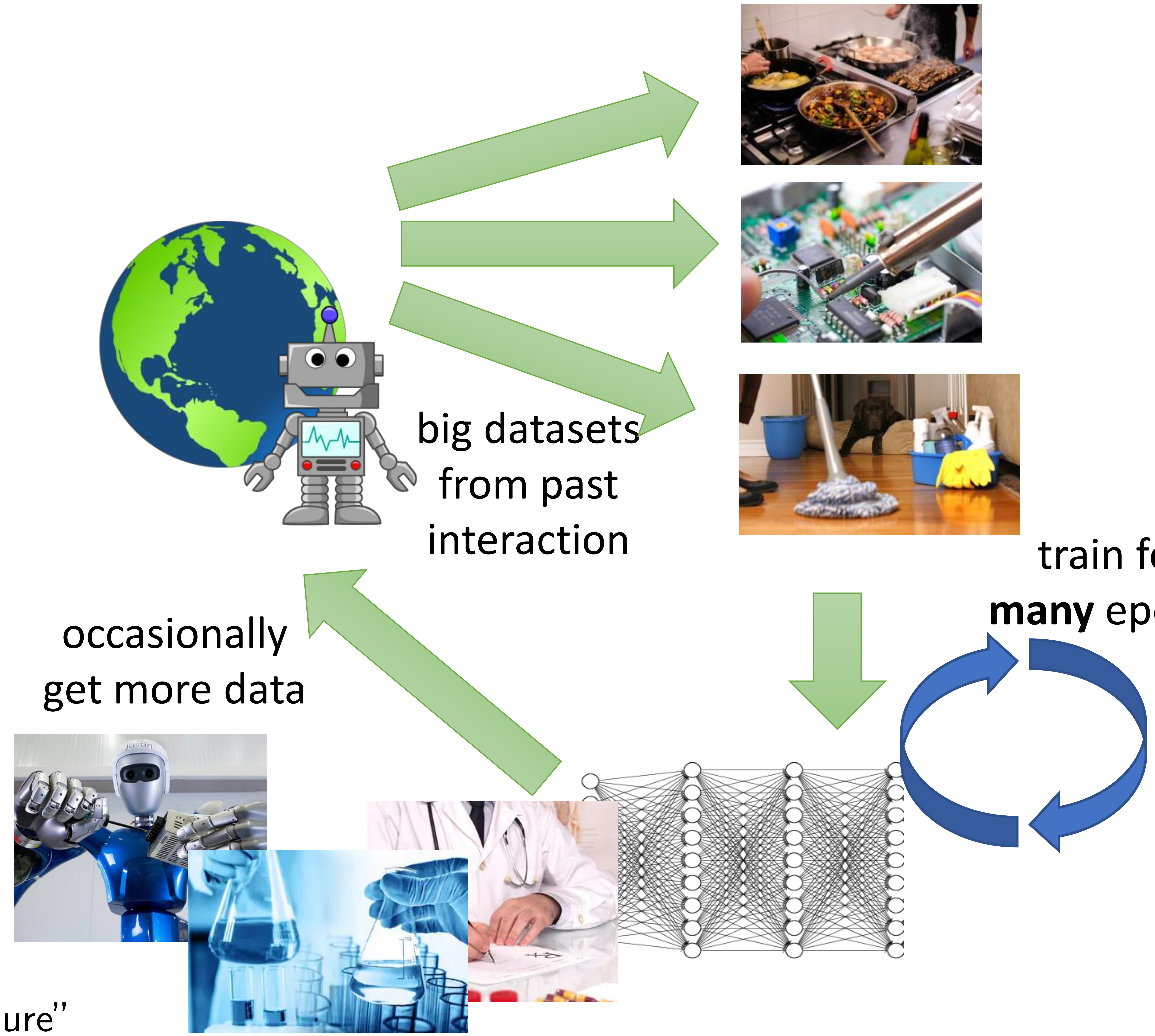


Credit: **Andy Zeng**

But for today, let’s pretend we can collect a ton of data
that “covers” tasks we care about

How can we learn optimal from large data collected by *any* policy?

Goal: Offline Reinforcement Learning



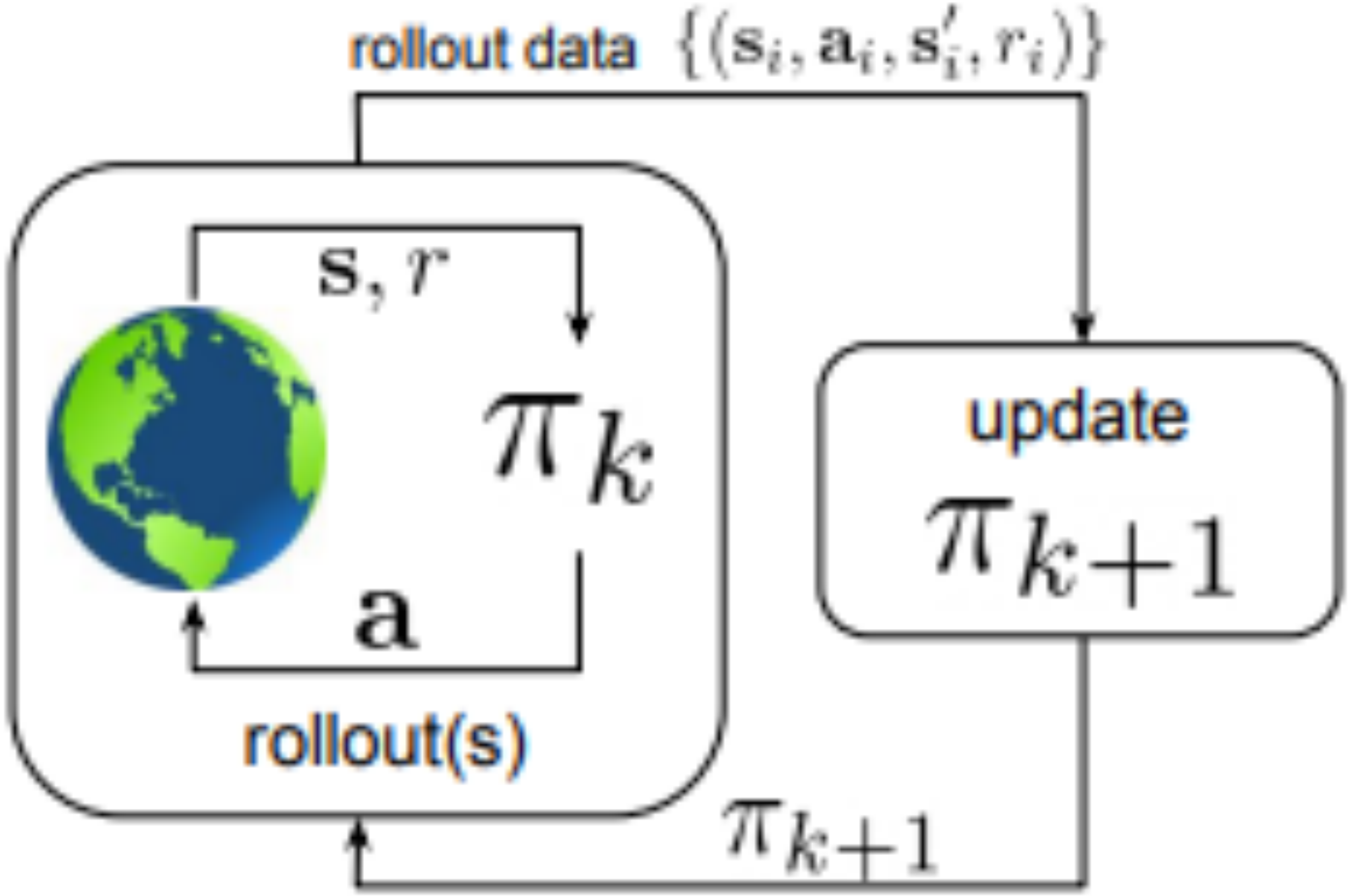
Credit: **Sergey Levine** "Offline RL lecture"

Different paradigms of RL

on-policy RL

Collect data with most recently policy π_k

Train on only this data



Credit: **Sergey Levine** "Offline RL lecture"

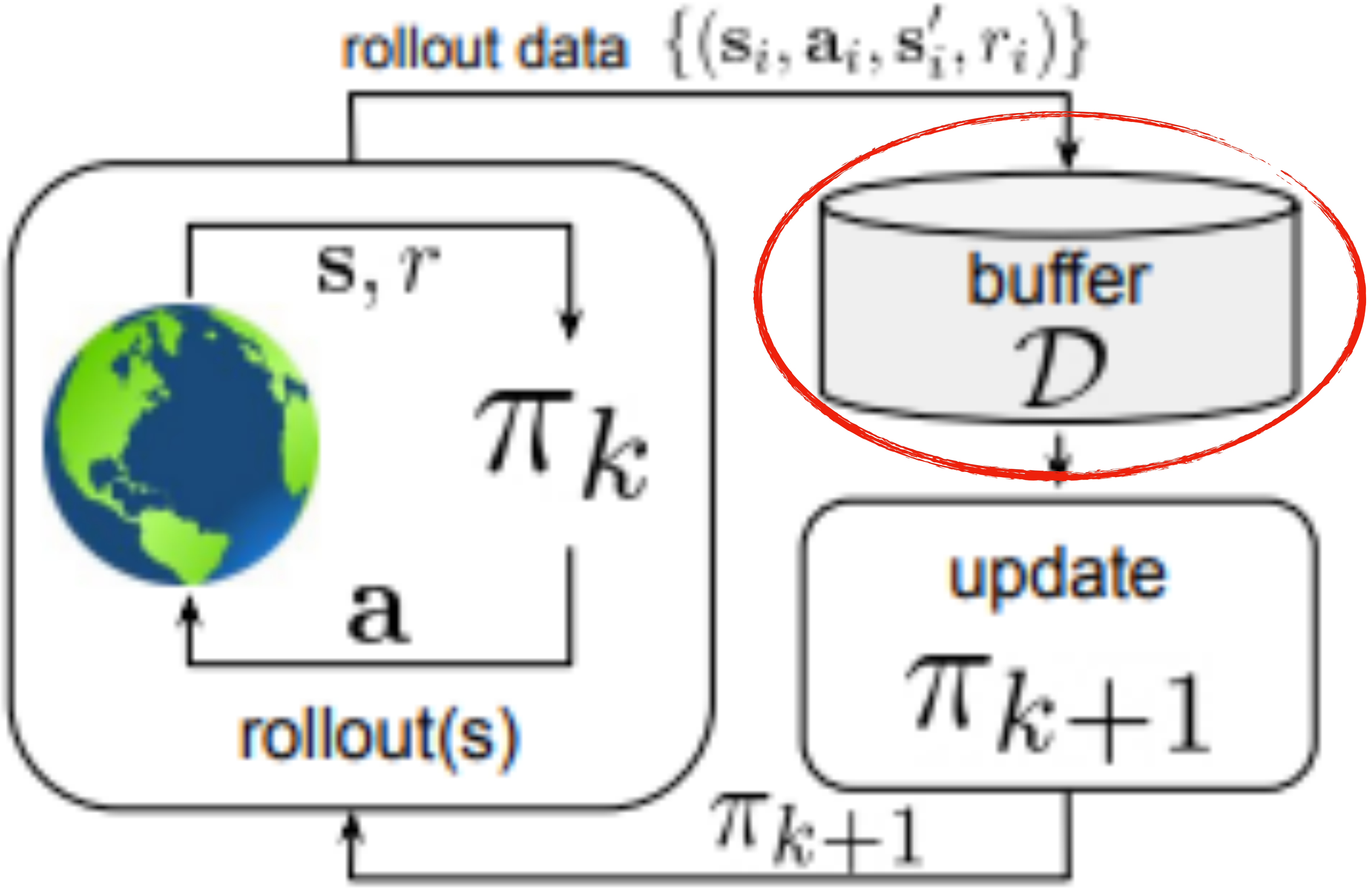
Different paradigms of RL

Collect data with most recently policy π_k

Aggregate this data in a buffer $\mathscr{D}$

Train on buffer

off-policy RL



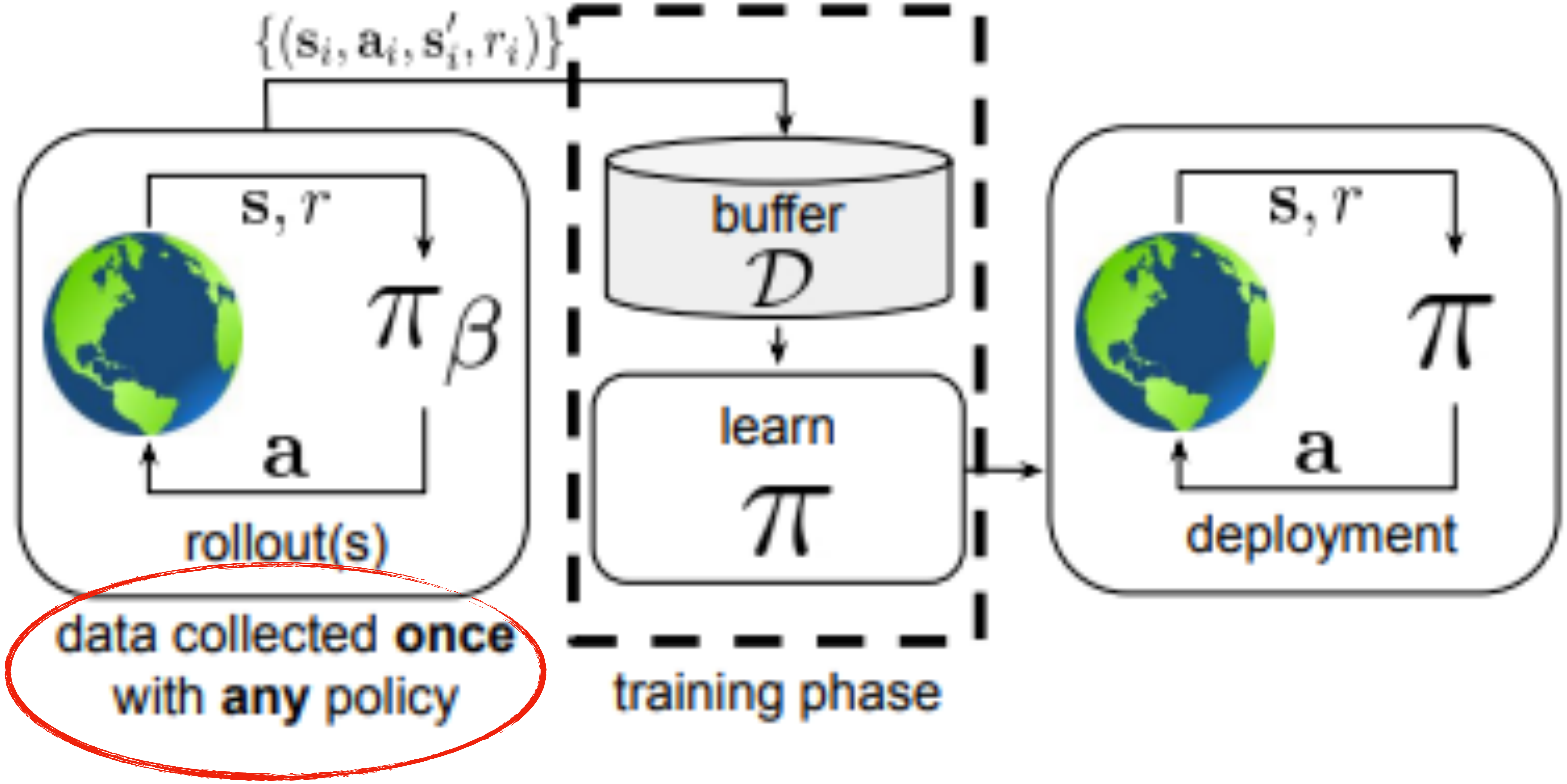
Credit: **Sergey Levine** "Offline RL lecture"

Different paradigms of RL

offline reinforcement learning

Data collected just once from any policy in buffer $\mathscr{D}$

Train on buffer



Credit: **Sergey Levine** "Offline RL lecture"

Different paradigms of RL

Offline RL enables robots to learn:
from pre-collected datasets
without real-time interaction,
enabling safer training
and leveraging diverse experiences.

Credit: **Sergey Levine** "Offline RL lecture"

Today's class

- ☑ What is offline RL? Why do we need it for robots?

  (Enables safer training, leverages diverse experience)

- ☐ Paradigm 1: Offline RL via Pessimism
  - ☐ Problem with Q-learning
  - ☐ Pessimism to the rescue

- ☐ Paradigm 2: RL via Supervised Learning
  - ☐ Return-conditioned Supervised Learning
  - ☐ Problem in Stochastic MDPs

Join by Web **PollEv.com/sc2582**

Join by Text Send **sc2582** to **22333**

Let's begin with a simple "offline" RL algorithm

We have already covered a fundamental algorithm in class that can learn from offline data.

What is it?

Q-learning for Offline RL

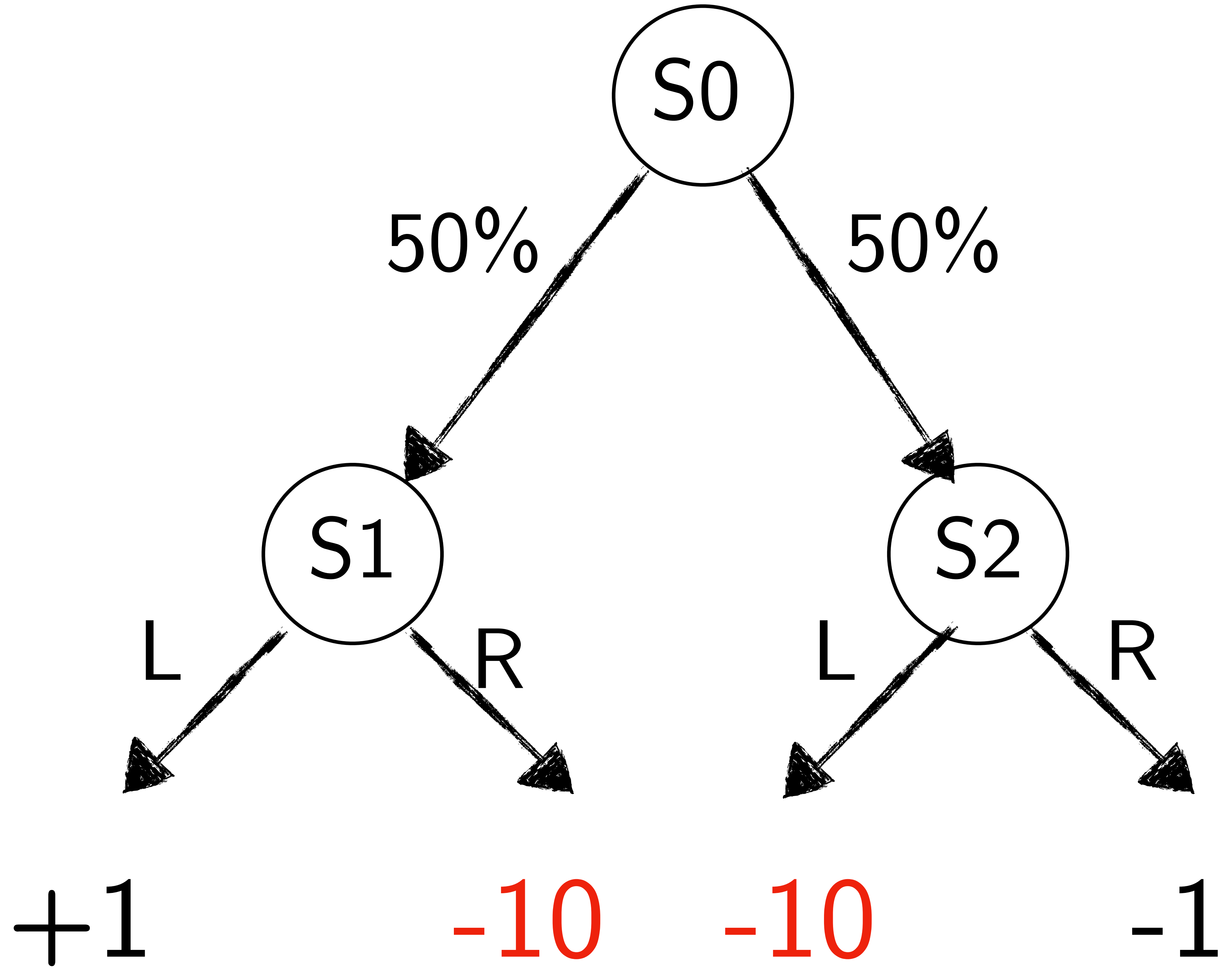
Collect data with a policy π_β and store in $\mathcal{D}$

For every $\left(s_t, a_t, r_t, s_{t+1}\right) \in \mathcal{D}$

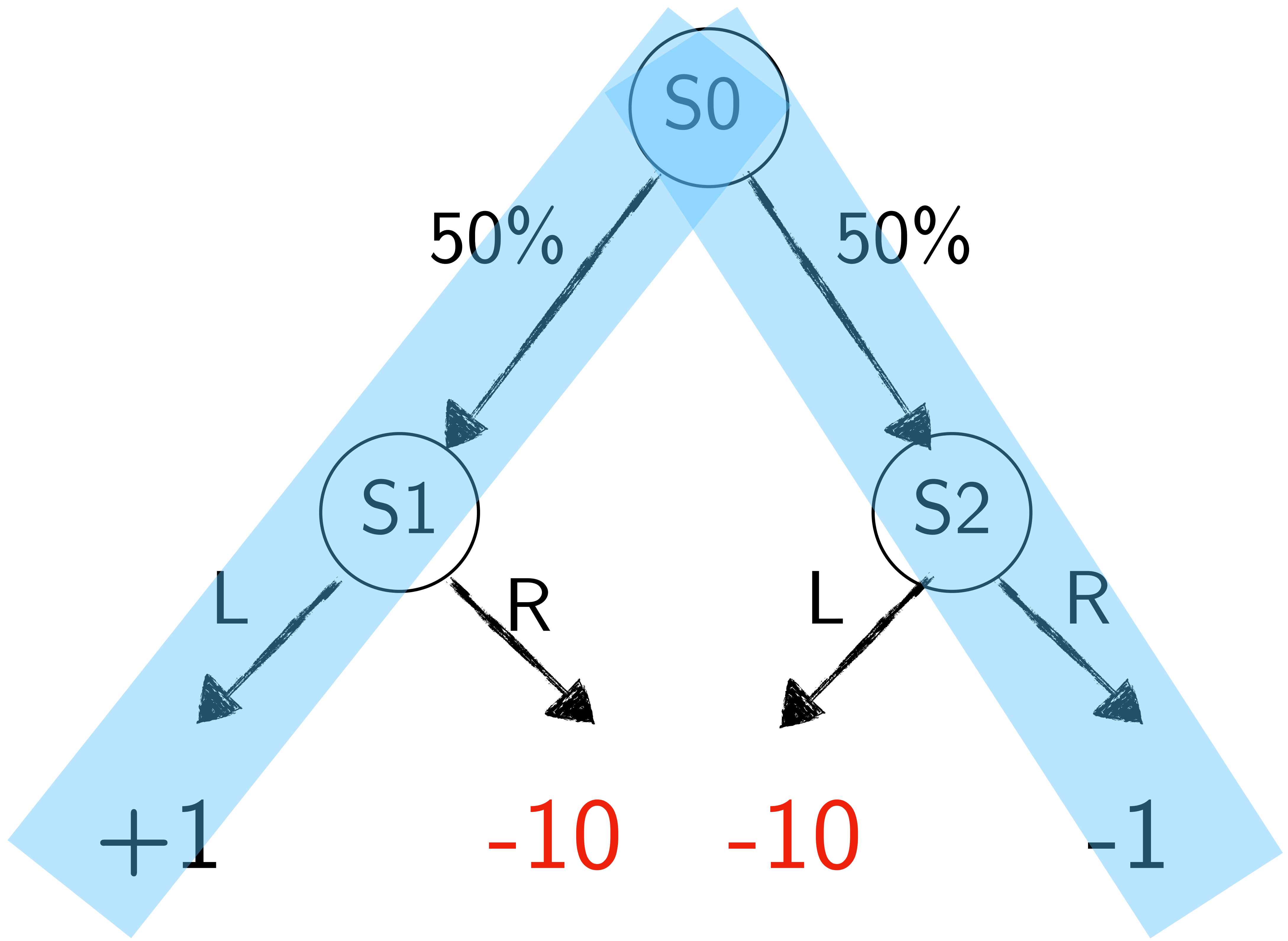
$$Q^*(s_t, a_t) = Q^*(s_t, a_t) + \alpha(r(s_t, a_t) + \gamma \max_{a'} Q^*(s_{t+1}, a') - Q^*(s_t, a_t))$$

Activity!

Consider the following MDP

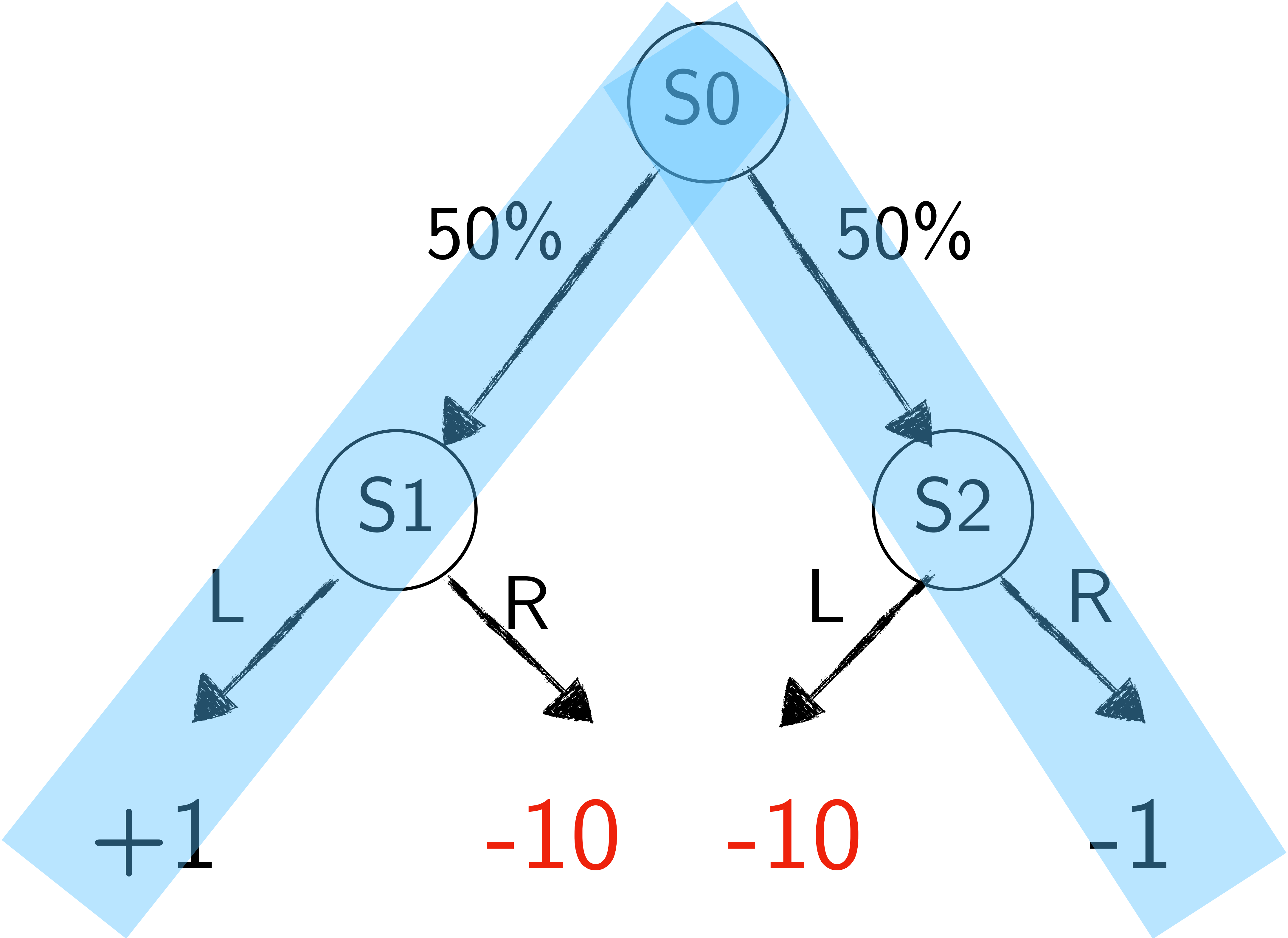


Let's say I collected some data from the MDP



Collect data with a policy π_β that is pretty good, and store in $\mathscr{D}$

What policy would Q-learning pick?



Assume we are in tabular case

Initialize Q values with 0's

For every $\left(s_t, a_t, r_t, s_{t+1}\right) \in \mathscr{D}$

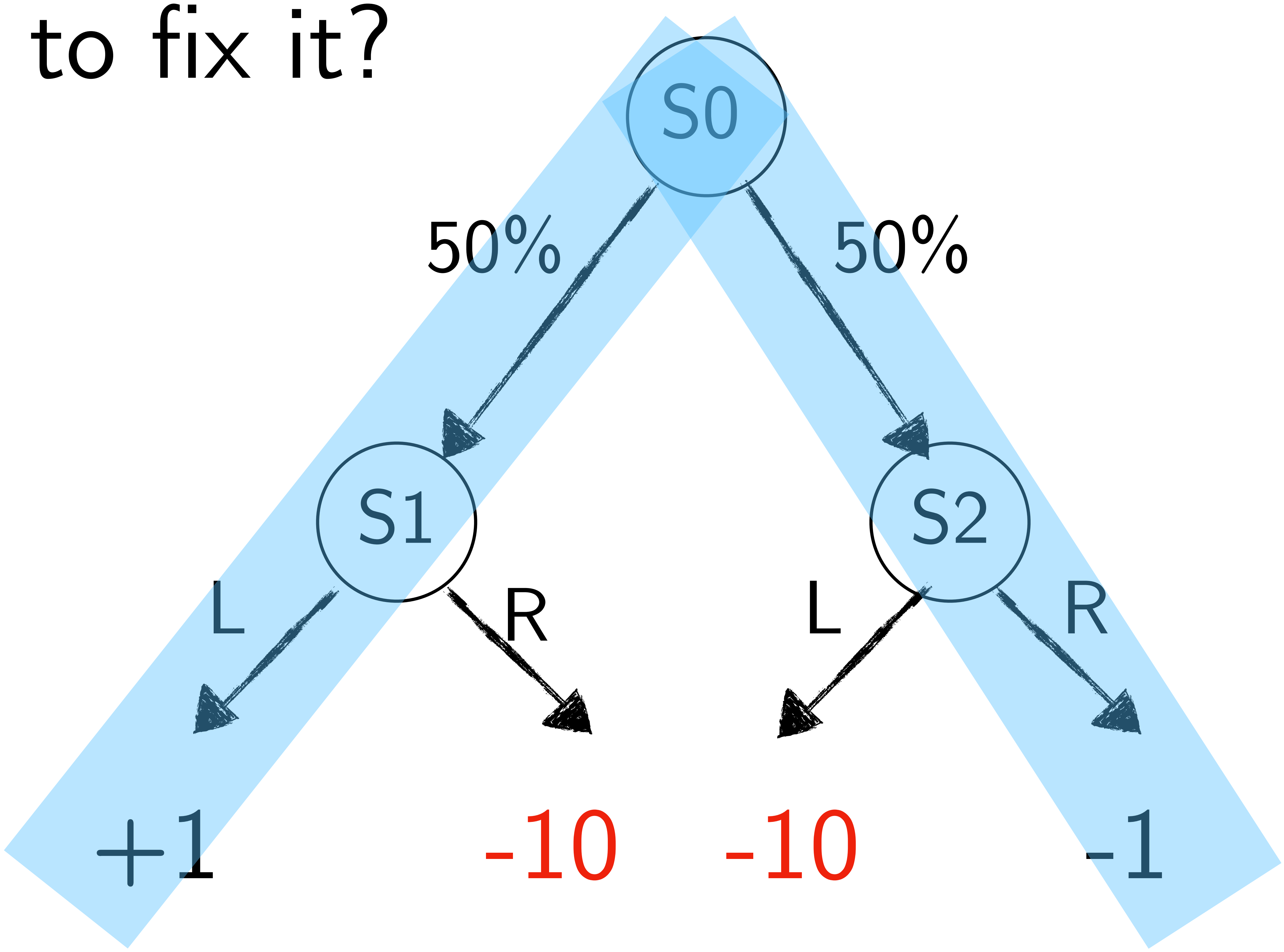
$$Q^*(s_t, a_t) = Q^*(s_t, a_t) + \alpha(r(s_t, a_t) + \gamma \max_{a'} Q^*(s_{t+1}, a') - Q^*(s_t, a_t))$$

Think-Pair-Share!

Think (30 sec): What policy would Q-learning pick in the tabular setting? Why? Ideas to fix it?

Pair: Find a partner

Share (45 sec): Partners exchange ideas



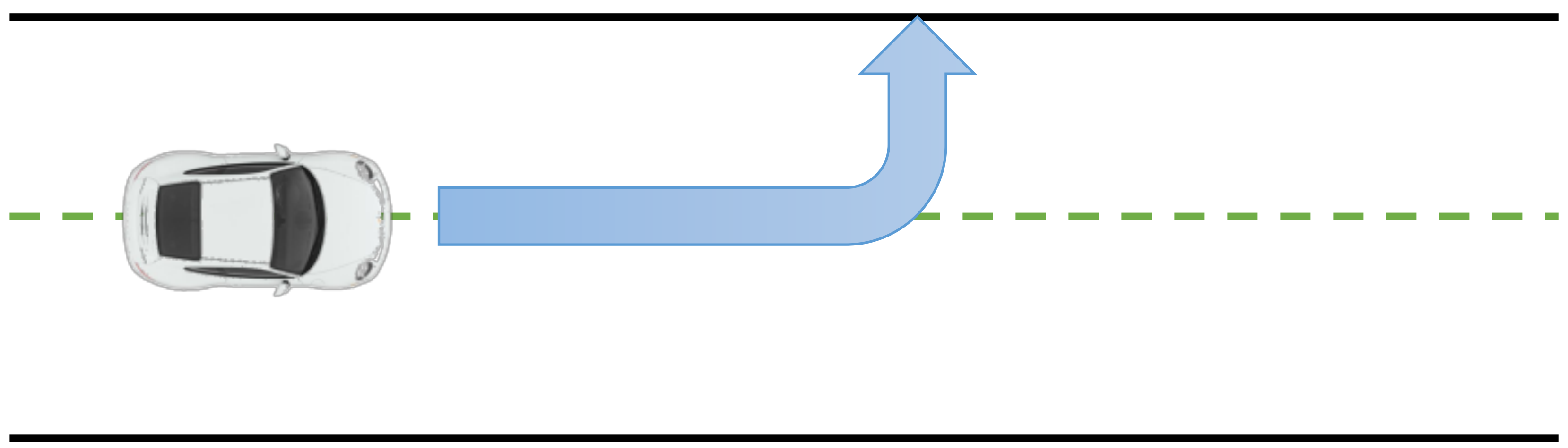
Credit: **Sergey Levine** “Offline RL lecture”

The Problem with Q-learning

Fundamental problem: counterfactual queries

Training data

What the policy wants to do



Is this good? Bad?

How do we know if we didn’t see it in the data?

Q-learning can be incorrectly optimistic about actions it has not see in the data

Today's class

- ☑ What is offline RL? Why do we need it for robots?

  (Enables safer training, leverages diverse experience)

- ☐ Paradigm 1: Offline RL via Pessimism
  - ☑ Problem with Q-learning (Incorrectly optimistic about unseen actions)
  - ☐ Pessimism to the rescue

- ☐ Paradigm 2: RL via Supervised Learning
  - ☐ Return-conditioned Supervised Learning
  - ☐ Problem in Stochastic MDPs

Pessimism

Pessimism as a policy constraint

Don't deviate too much from the data collecting policy

Pessimism as a policy constraint

"Don't deviate too much from the data collecting policy"

Collect data with a policy π_β and store in $\mathcal{D}$

For $(s, a, r, s') \in \mathcal{D}$

$$Q(\mathbf{s}, \mathbf{a}) \leftarrow r(\mathbf{s}, \mathbf{a}) + E_{\mathbf{a}' \sim \pi_{\text{new}}}[Q(\mathbf{s}', \mathbf{a}')]$$

$$\pi_{\text{new}}(\mathbf{a}|\mathbf{s}) = \arg\max_{\pi} E_{\mathbf{a} \sim \pi(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s}, \mathbf{a})]$$

Typical Q-learning

Credit: **Sergey Levine** "Offline RL lecture"

Pessimism as a policy constraint

"Don't deviate too much from the data collecting policy"

Collect data with a policy π_β and store in $\mathscr{D}$

For $(s, a, r, s') \in \mathscr{D}$

$$Q(\mathbf{s}, \mathbf{a}) \leftarrow r(\mathbf{s}, \mathbf{a}) + E_{\mathbf{a}' \sim \pi_{\text{new}}}[Q(\mathbf{s}', \mathbf{a}')]$$

$$\pi_{\text{new}}(\mathbf{a}|\mathbf{s}) = \arg\max_{\pi} E_{\mathbf{a} \sim \pi(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s}, \mathbf{a})] \text{ s.t. } \boxed{D_{\text{KL}}(\pi \| \pi_\beta) \leq \epsilon}$$

Typical Q-learning

Add a constraint on policy

TD3+BC: Most simple and effective offline RL!

A Minimalist Approach to Offline Reinforcement Learning

Scott Fujimoto[1,2] **Shixiang Shane Gu**[2]
[1]Mila, McGill University
[2]Google Research, Brain Team
scott.fujimoto@mail.mcgill.ca

$$\cancel{\pi = \operatorname*{argmax}_{\pi} \mathbb{E}_{(s,a)\sim\mathcal{D}}[Q(s, \pi(s))].}$$

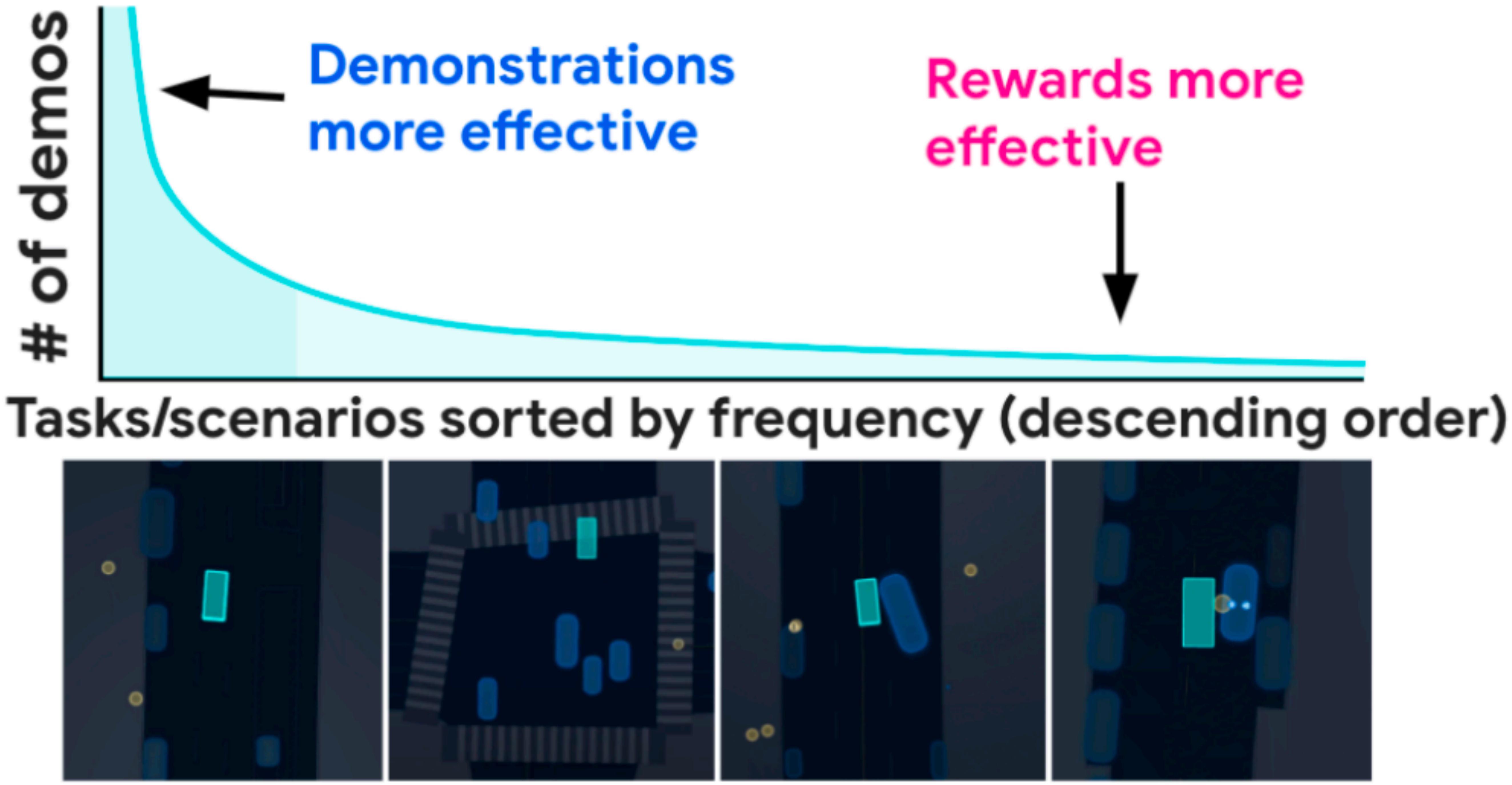
$$\pi = \operatorname*{argmax}_{\pi} \mathbb{E}_{(s,a)\sim\mathcal{D}} \left[\lambda Q(s, \pi(s)) - (\pi(s) - a)^2 \right],$$

		BC	BRAC-p	AWAC	CQL	Fisher-BRC	TD3+BC
Random	HalfCheetah	2.0 ±0.1	23.5	2.2	21.7 ±0.9	32.2 ±2.2	10.2 ±1.3
	Hopper	9.5 ±0.1	11.1	9.6	10.7 ±0.1	11.4 ±0.2	11.0 ±0.1
	Walker2d	1.2 ±0.2	0.8	5.1	2.7 ±1.2	0.6 ±0.6	1.4 ±1.6
Medium	HalfCheetah	36.6 ±0.6	44.0	37.4	37.2 ±0.3	41.3 ±0.5	42.8 ±0.3
	Hopper	30.0 ±0.5	31.2	72.0	44.2 ±10.8	99.4 ±0.4	99.5 ±1.0
	Walker2d	11.4 ±6.3	72.7	30.1	57.5 ±8.3	79.5 ±1.0	79.7 ±1.8
Medium Replay	HalfCheetah	34.7 ±1.8	45.6	-	41.9 ±1.1	43.3 ±0.9	43.3 ±0.5
	Hopper	19.7 ±5.9	0.7	-	28.6 ±0.9	35.6 ±2.5	31.4 ±3.0
	Walker2d	8.3 ±1.5	-0.3	-	15.8 ±2.6	42.6 ±7.0	25.2 ±5.1
Medium Expert	HalfCheetah	67.6 ±13.2	43.8	36.8	27.1 ±3.9	96.1 ±9.5	97.9 ±4.4
	Hopper	89.6 ±27.6	1.1	80.9	111.4 ±1.2	90.6 ±43.3	112.2 ±0.2
	Walker2d	12.0 ±5.8	-0.3	42.7	68.1 ±13.1	103.6 ±4.6	101.1 ±9.3
Expert	HalfCheetah	105.2 ±1.7	3.8	78.5	82.4 ±7.4	106.8 ±3.0	105.7 ±1.9
	Hopper	111.5 ±1.3	6.6	85.2	111.2 ±2.1	112.3 ±0.2	112.2 ±0.2
	Walker2d	56.0 ±24.9	-0.2	57.0	103.8 ±7.6	79.9 ±32.4	105.7 ±2.7
	Total	595.3 ±91.5	284.1	-	764.3 ±61.5	974.6 ±108.3	979.3 ±33.4

Works on real self-driving problems!

Imitation Is Not Enough: Robustifying Imitation with Reinforcement Learning for Challenging Driving Scenarios

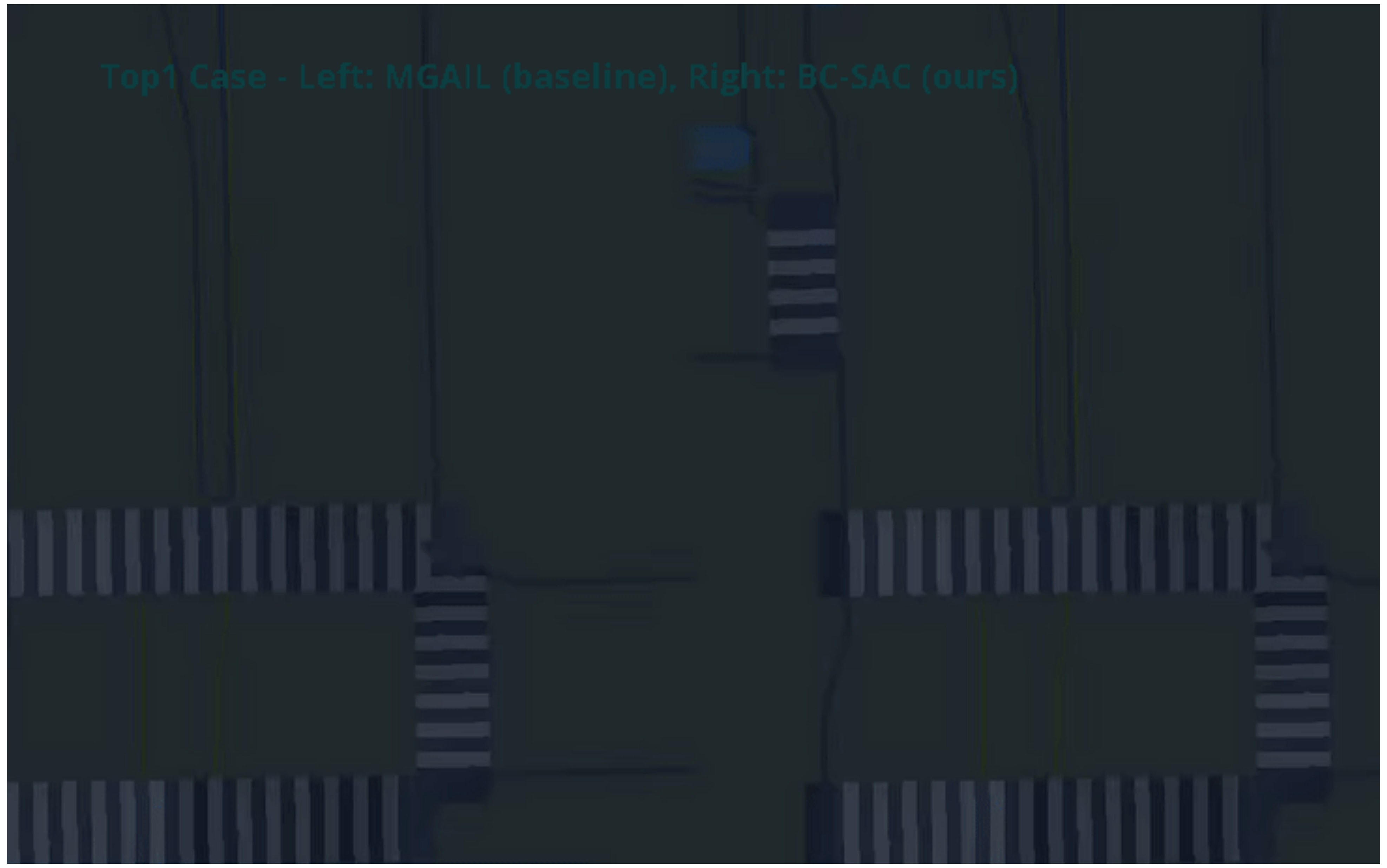
Yiren Lu[1], Justin Fu[1], George Tucker[2], Xinlei Pan[1], Eli Bronstein[1], Rebecca Roelofs[2], Benjamin Sapp[1], Brandyn White[1], Aleksandra Faust[2], Shimon Whiteson[1], Dragomir Anguelov[1], Sergey Levine[2,3]



Works on real self-driving problems!

Imitation Is Not Enough: Robustifying Imitation with Reinforcement Learning for Challenging Driving Scenarios

Yiren Lu[1], Justin Fu[1], George Tucker[2], Xinlei Pan[1], Eli Bronstein[1], Rebecca Roelofs[2], Benjamin Sapp[1], Brandyn White[1], Aleksandra Faust[2], Shimon Whiteson[1], Dragomir Anguelov[1], Sergey Levine[2,3]



https://waymo.com/research/imitation-is-not-enough-robustifying-imitation-with-reinforcement-learning/

Many more sophisticated offline RL methods

Conservative Q-Learning for Offline Reinforcement Learning

Aviral Kumar[1], Aurick Zhou[1], George Tucker[2], Sergey Levine[1,2]
[1]UC Berkeley, [2]Google Research, Brain Team
aviralk@berkeley.edu

Instead of constraining policy, compute pessimistic Q values

Adversarially Trained Actor Critic for Offline Reinforcement Learning

Ching-An Cheng[*1] Tengyang Xie[*2] Nan Jiang[2] Alekh Agarwal[3]

Optimize the best worst case performance

Today's class

- ☑ What is offline RL? Why do we need it for robots?
  (Enables safer training, leverages diverse experience)

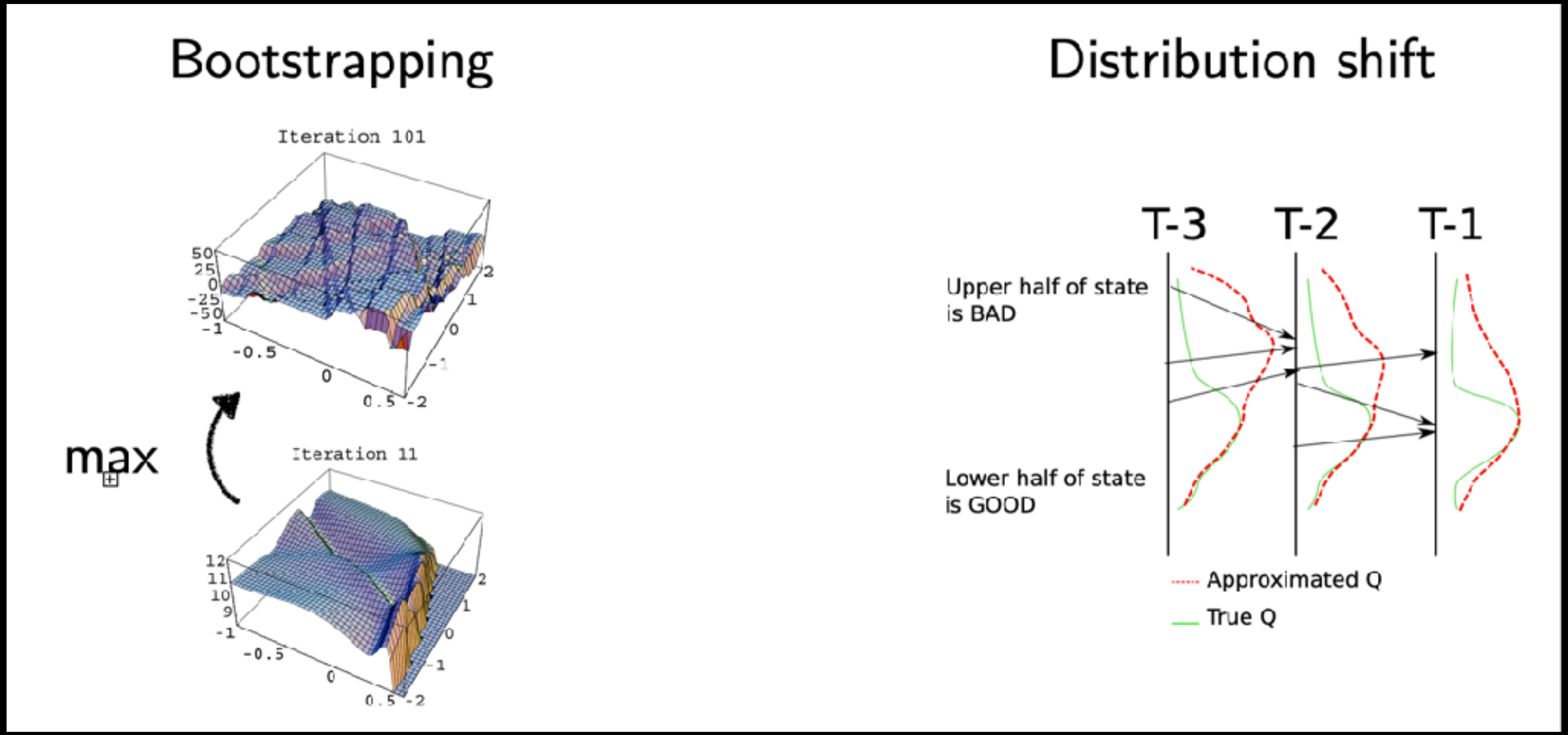
- ☑ Paradigm 1: Offline RL via Pessimism
  - ☑ Problem with Q-learning (Incorrectly optimistic about unseen actions)
  - ☑ Pessimism to the rescue (Constrain policy to not deviate from data)

- ☐ Paradigm 2: RL via Supervised Learning
  - ☐ Return-conditioned Supervised Learning
  - ☐ Problem in Stochastic MDPs

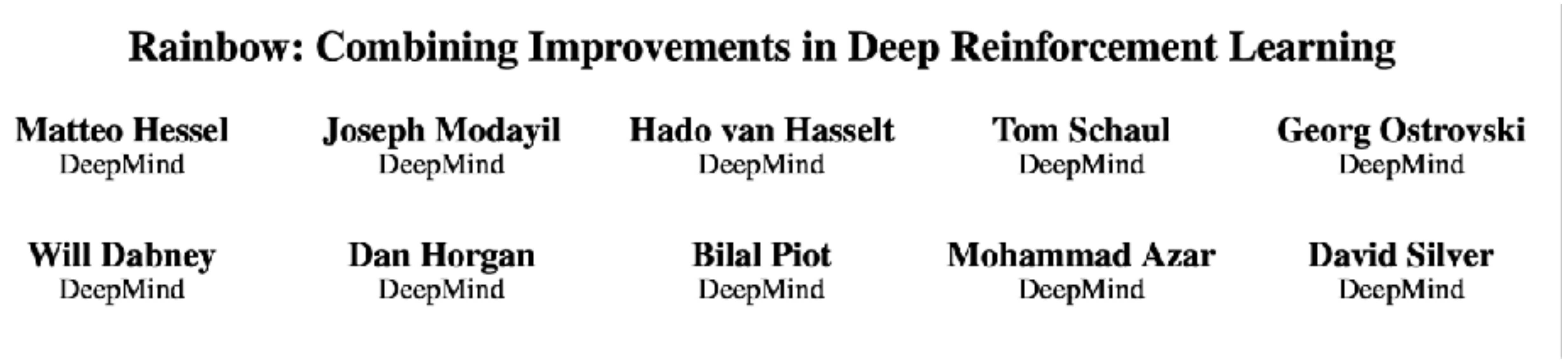
Reinforcement Learning is Hard ...

Many horror stories of RL!

Nightmares of
Policy Optimization



Need many tricks to make Q-learning work in practice!



Double Q Learning

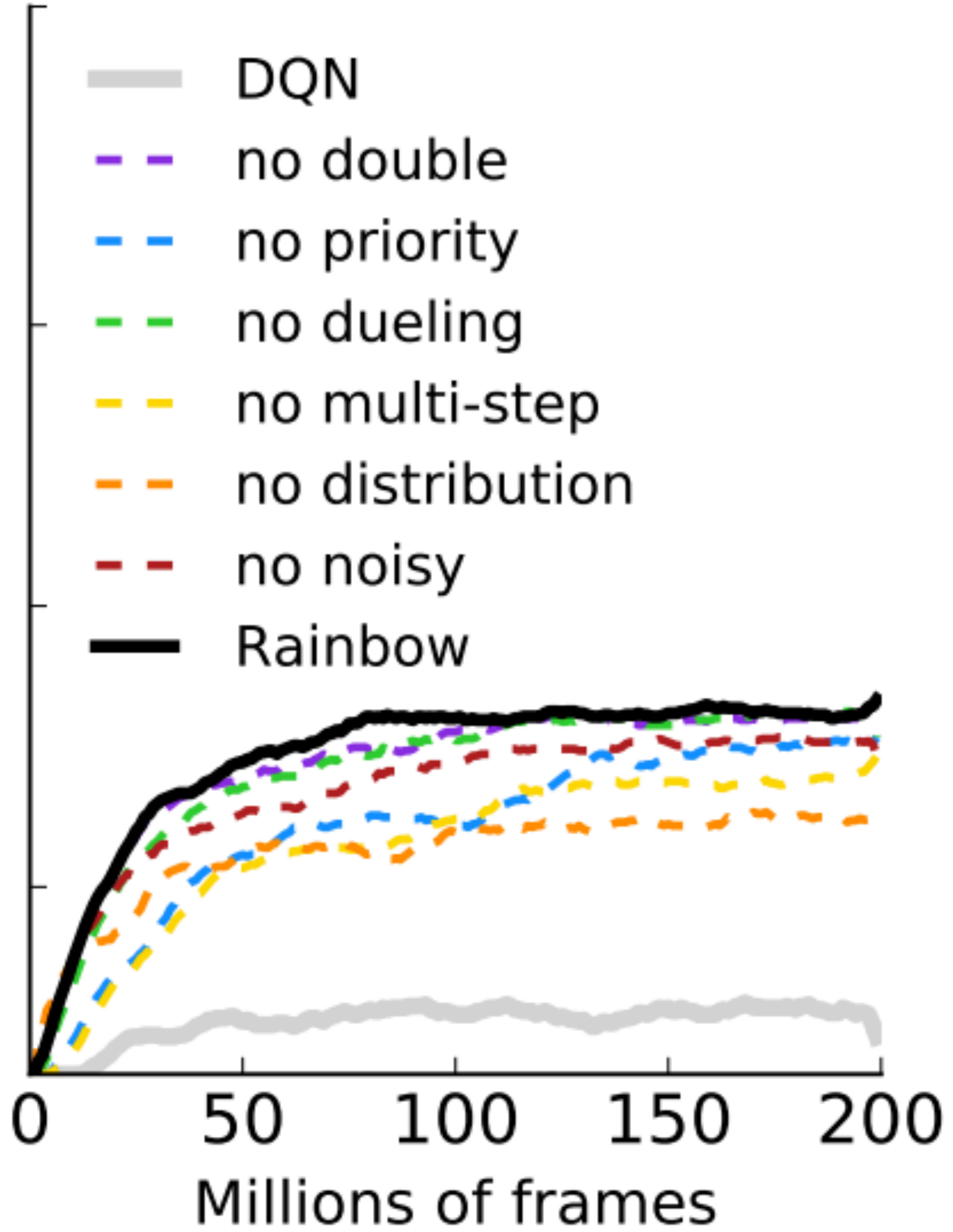
Prioritized Replay

Dueling Networks

Multi-step Learning

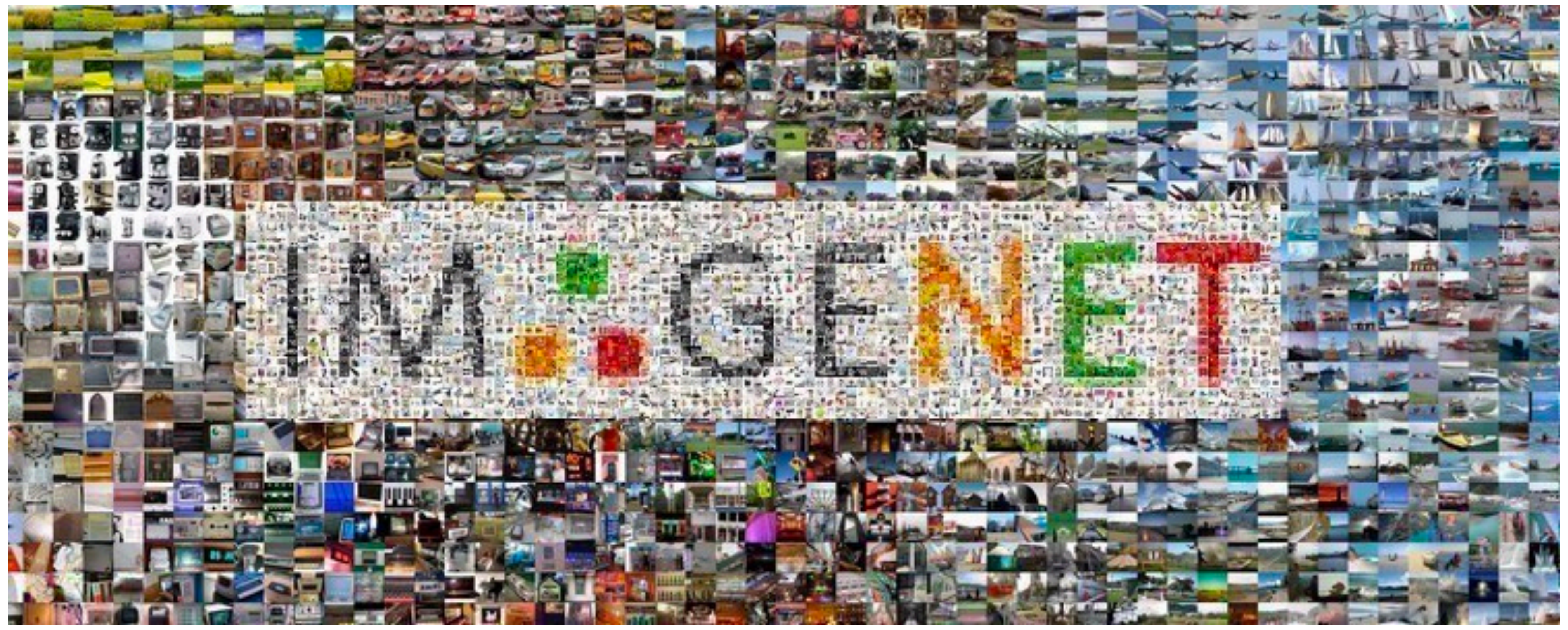
Distributional RL

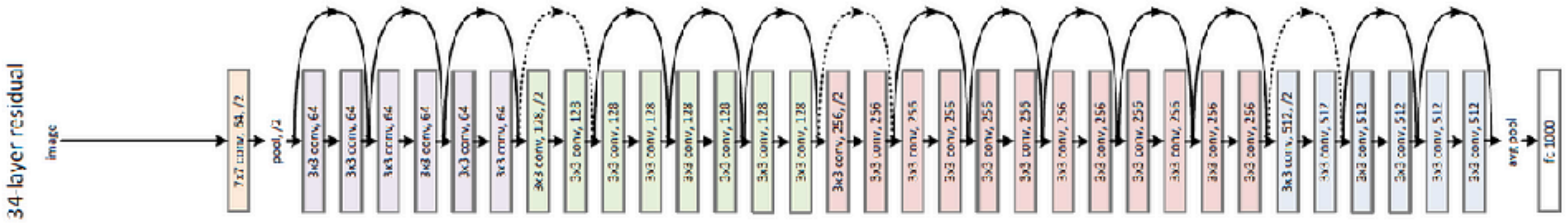
Noisy Nets

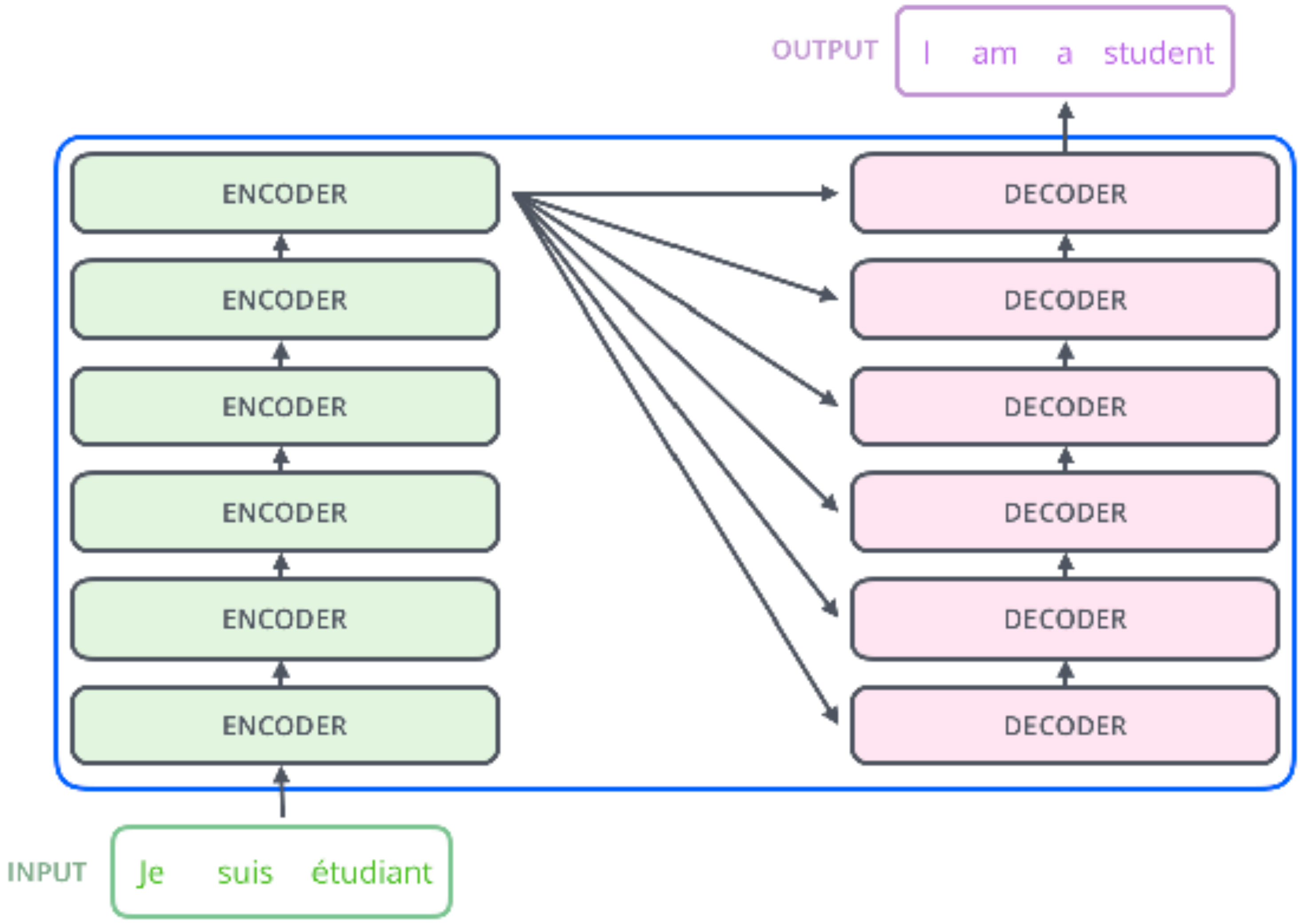


Can we just go back to good old supervised learning?

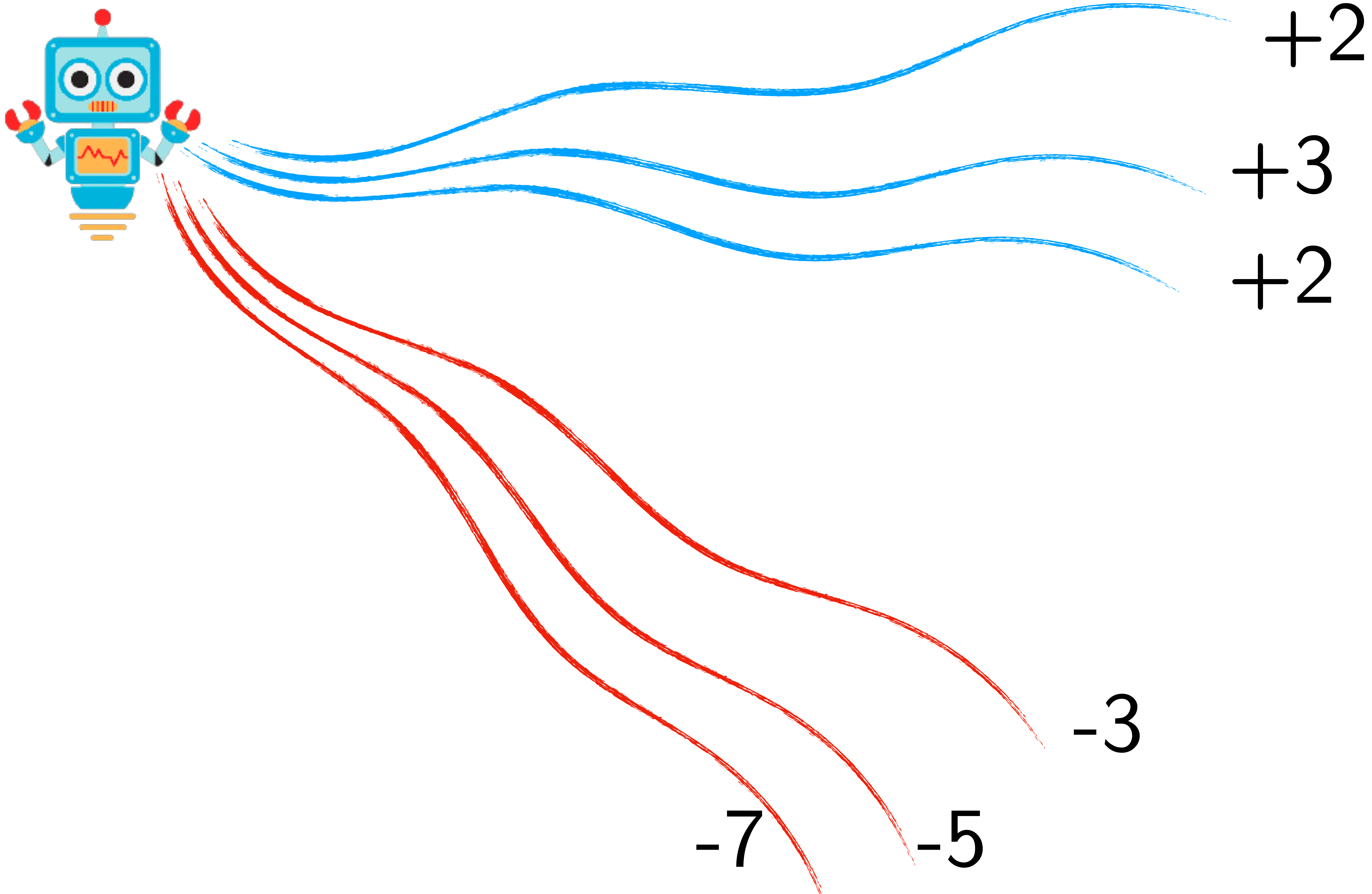
Supervised Learning success stories



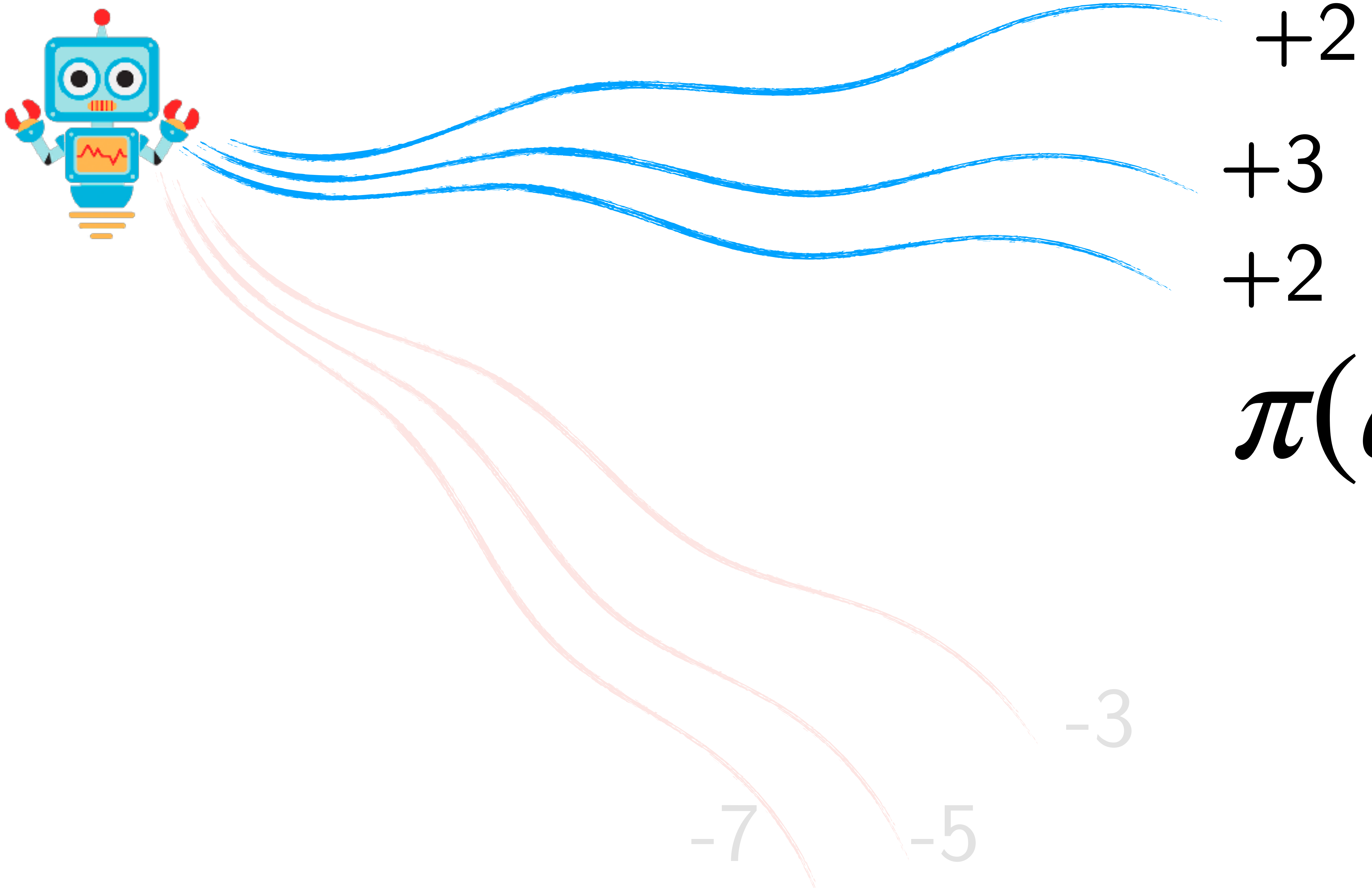




What if I did supervised learning (BC) here?



What if I did supervised learning (BC) only on the top % rollouts?



$$\pi(a \mid s)$$

An embarrassingly simply algorithm: BC%

1. Collect offline dataset using whatever behavior policy

2. Get the top % trajectories based on returns

3. Do BC on just that!

Does this even work ?!?

A legit Offline RL Algo

Dataset	Environment	10%BC	25%BC	40%BC	100%BC	CQL
Medium	HalfCheetah	42.9	43.0	43.1	43.1	**44.4**
Medium	Hopper	65.9	65.2	65.3	63.9	58.0
Medium	Walker	78.8	**80.9**	78.8	77.3	79.2
Medium	Reacher	51.0	48.9	58.2	**58.4**	26.0
Medium-Replay	HalfCheetah	40.8	40.9	41.1	4.3	**46.2**
Medium-Replay	Hopper	70.6	58.6	31.0	27.6	48.6
Medium-Replay	Walker	**70.4**	67.8	67.2	36.9	26.7
Medium-Replay	Reacher	**33.1**	16.2	10.7	5.4	19.0
Average		**56.7**	52.7	49.4	39.5	43.5

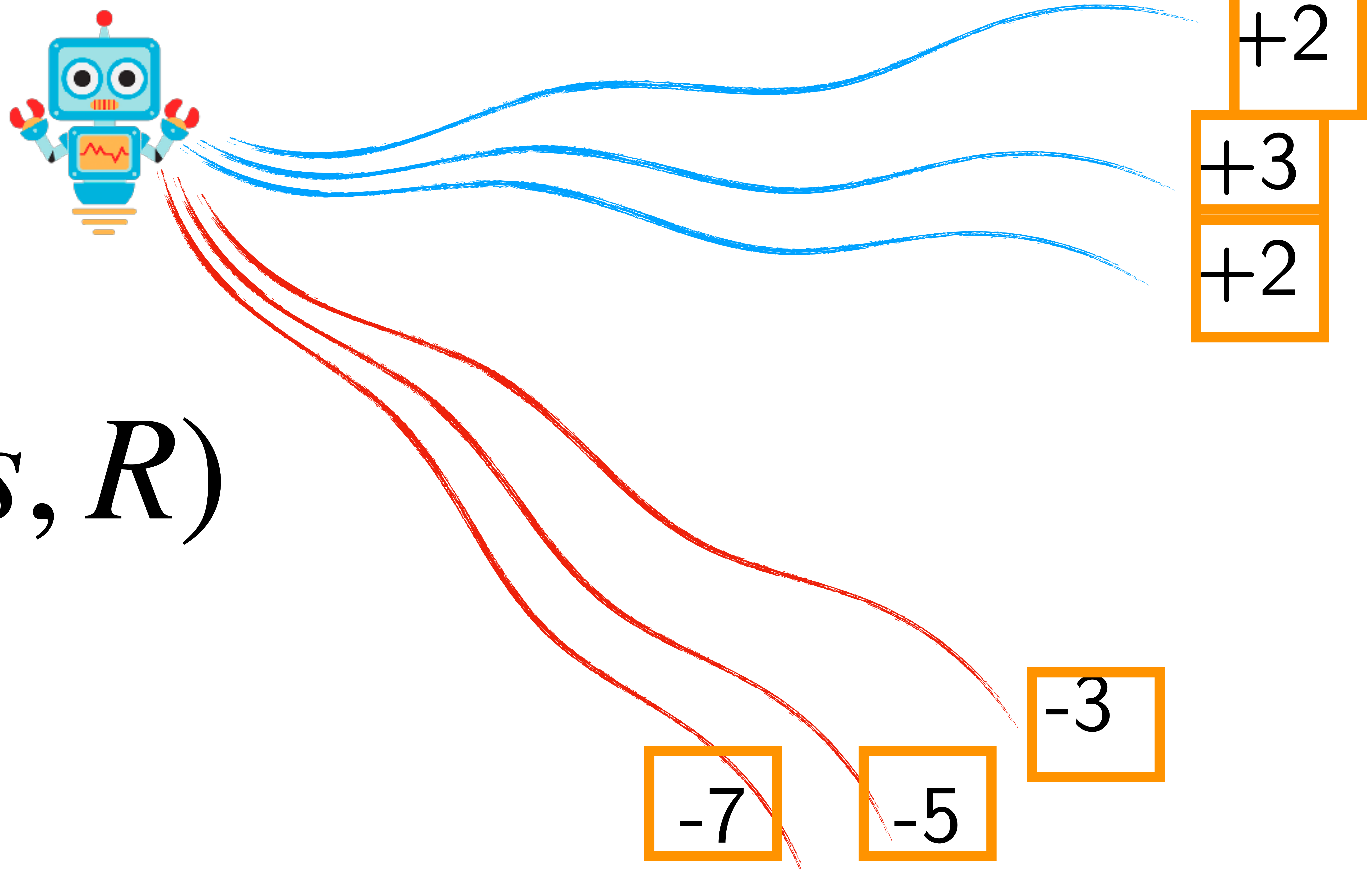
An embarrassingly simply algorithm: BC%

1. Collect offline dataset using whatever behavior policy
2. Get the top % trajectories based on returns
3. Do BC on just that!

Challenge with BC%:
What happens as I vary % from small to high values?

Can we have a more principled approach?

Idea: Train a policy *conditioned* on the returns

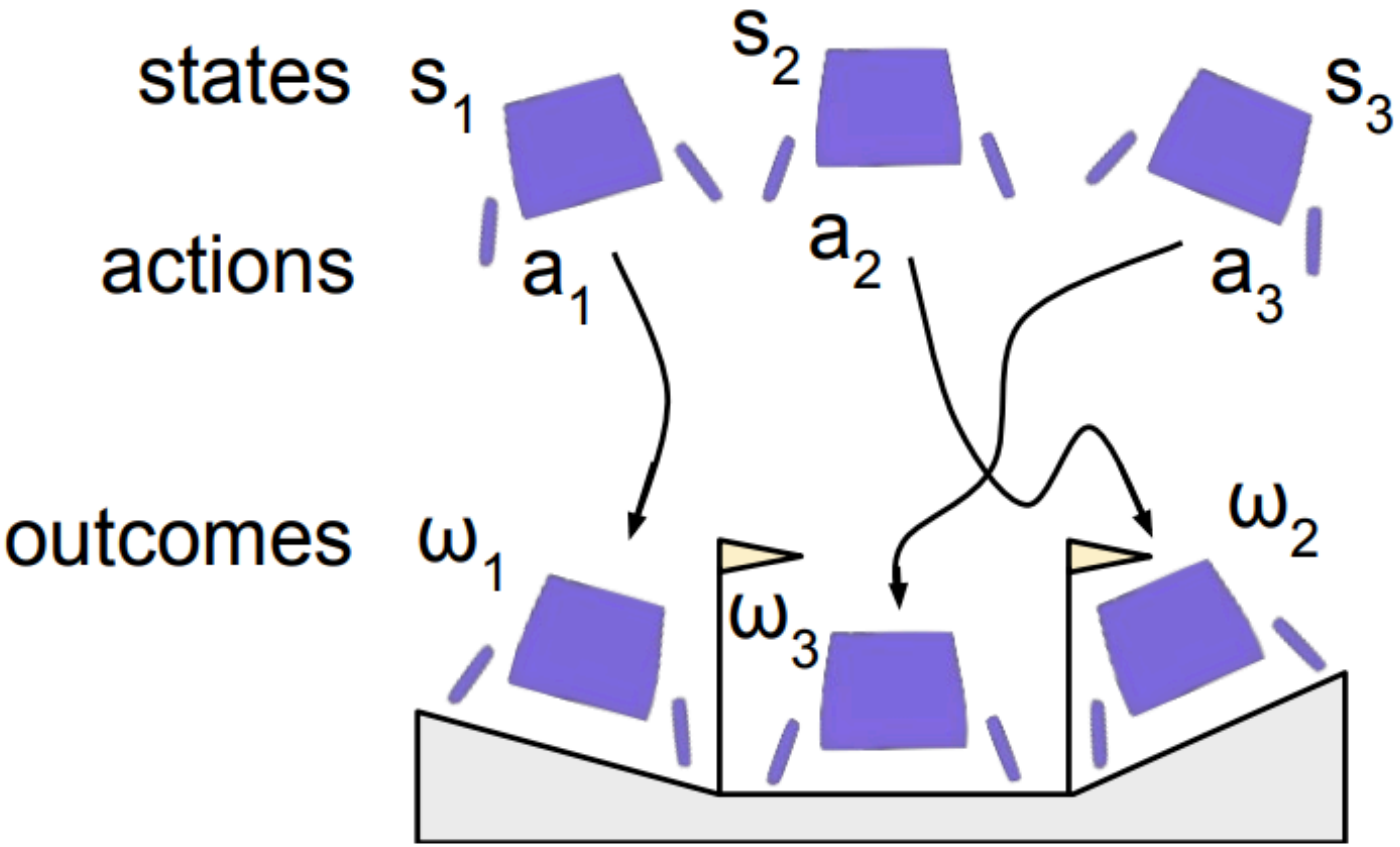


$$\pi(a \mid s, R)$$

RvS: What is Essential for Offline RL via Supervised Learning?

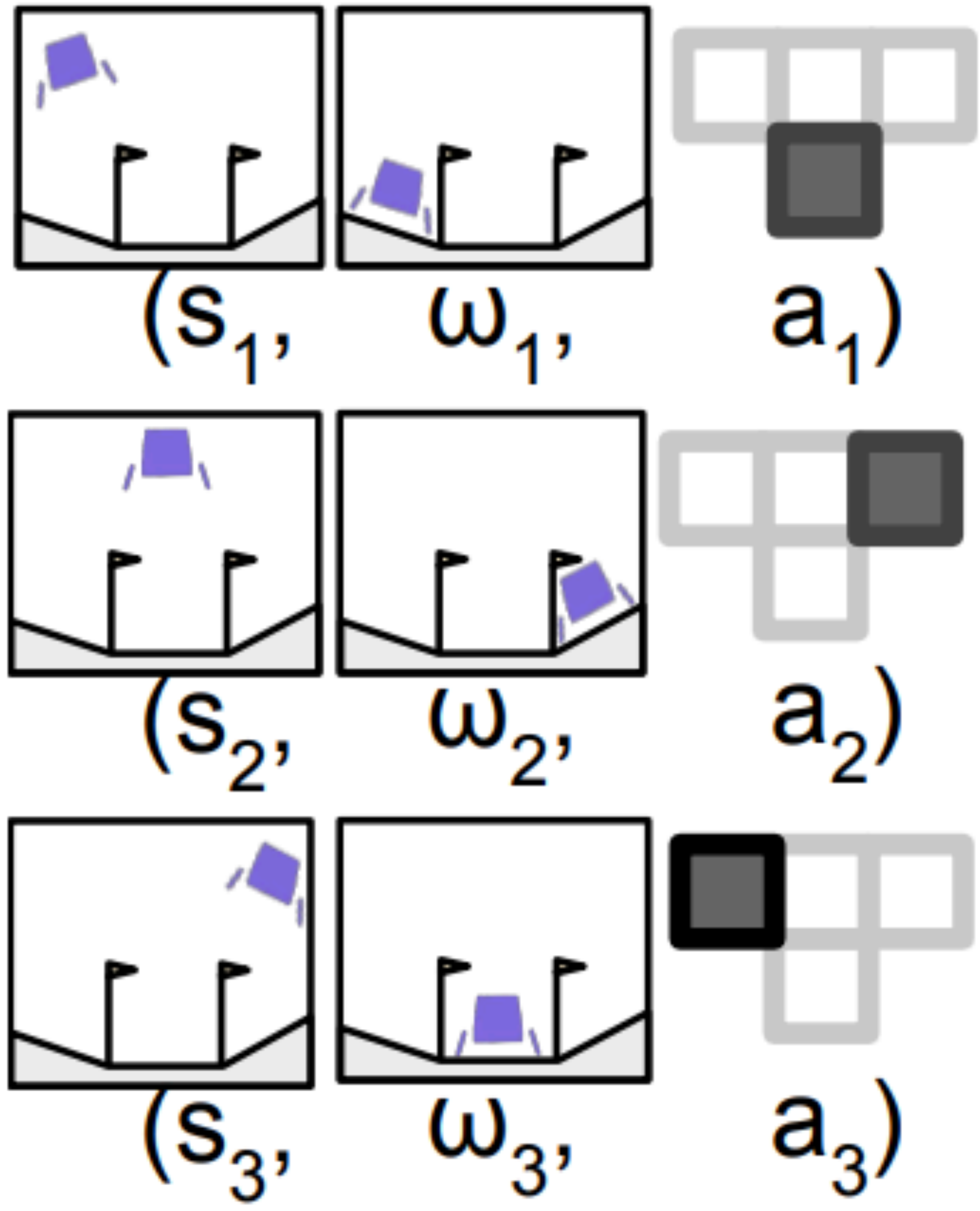
Scott Emmons[1], **Benjamin Eysenbach**[2], **Ilya Kostrikov**[1], **Sergey Levine**[1]
[1]UC Berkeley, [2]Carnegie Mellon University
emmons@berkeley.edu

The Idea



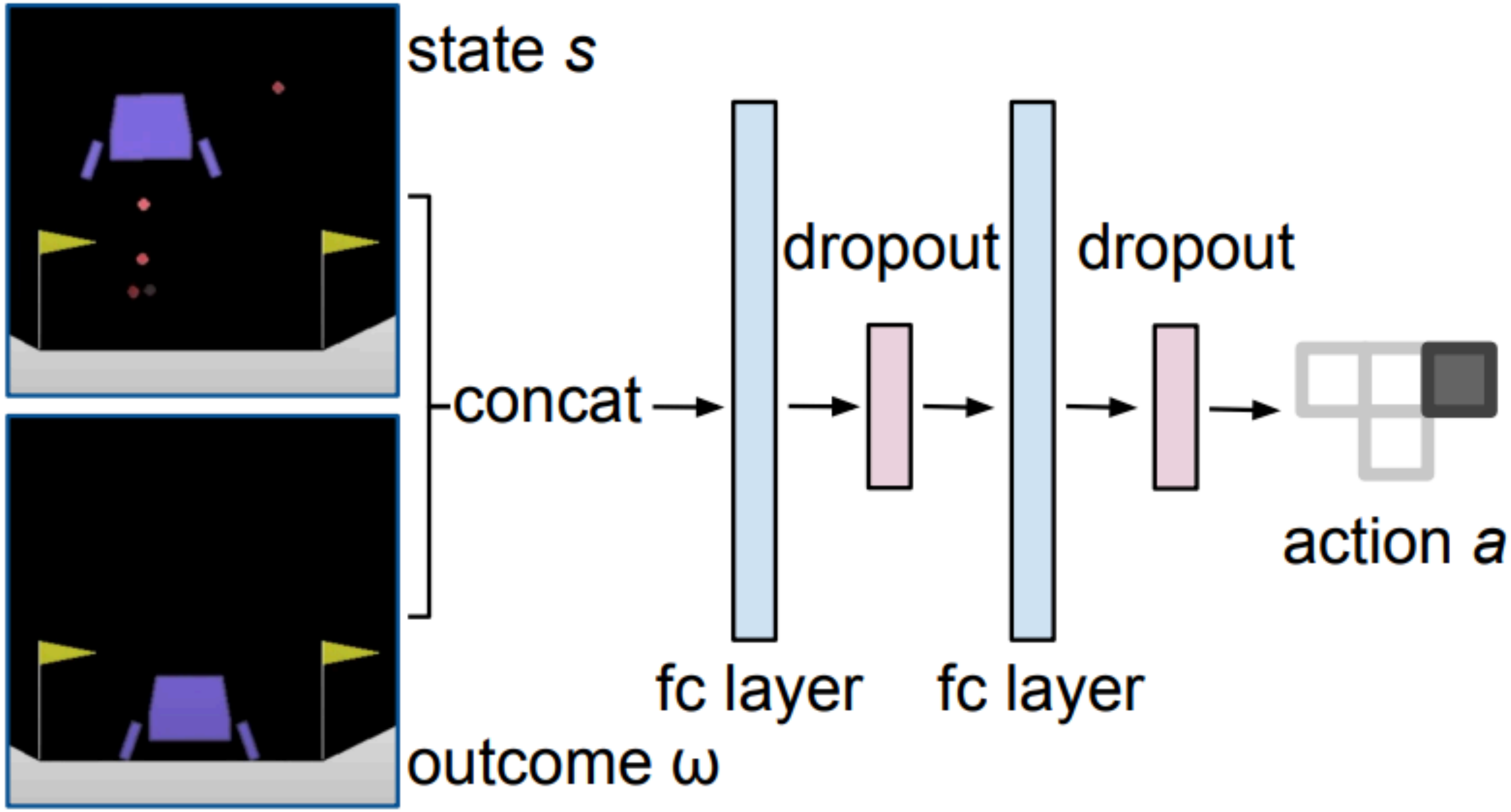
(a) replay buffer

The Idea



(b) training dataset

The Idea



(c) network architecture

The Algorithm

For all trajectories: For all timesteps in that trajectory: For all achieved outcomes:

$$\max_{\theta} \sum_{\tau \in \mathcal{D}} \sum_{1 \le t \le |\tau|} \mathbb{E}_{\omega \sim f(\omega \mid \tau_{t:H})} [\log \pi_\theta(a_t \mid s_t, \omega)].$$

Algorithm 1 RvS-Learning

1: **Input**: Dataset of trajectories, $\mathcal{D} = \{\tau\}$
2: Initialize policy $\pi_\theta(a \mid s, \omega)$.
3: **while** not converged **do**
4: Randomly sample trajectories: $\tau \sim \mathcal{D}$.
5: Sample time index for each trajetory, $t \sim [1, H]$, and sample a corresponding outcome: $\omega \sim f(\omega \mid \tau_{t:H})$.
6: Compute loss: $\mathcal{L}(\theta) \leftarrow \sum_{(s_t, a_t, \omega)} \log \pi_\theta(a_t \mid s_t, \omega)$
7: Update policy parameters: $\theta \leftarrow \theta + \eta \nabla_\theta \mathcal{L}(\theta)$
8: **end while**
9: **return** Conditional policy $\pi_\theta(a \mid s, \omega)$

What are some choices for “outcomes”?

Option 1: What is the future state the agent ended up at?

RvS-G (Goal conditioned)

Option 2: What is the total return that the agent got?

RvS-R (Return conditioned)

A very *popular* idea

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. **Decision transformer: Reinforcement learning via sequence modeling**

Felipe Codevilla, Matthias Muller, Antonio Lopez, Vladlen Koltun, and Alexey Dosovitskiy. **End-to-end driving via conditional imitation learning**

Yiming Ding, Carlos Florensa, Pieter Abbeel, and Mariano Phielipp. **Goal-conditioned imitation learning.**

Michael Janner, Qiyang Li, and Sergey Levine. **Offline reinforcement learning as one big sequence modeling problem**

Aviral Kumar, Xue Bin Peng, and Sergey Levine. **Reward-conditioned policies**

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. **Advantage-weighted regression: Simple and scalable off-policy reinforcement learning**

Rupesh Kumar Srivastava, Pranav Shyam, Filipe Mutz, Wojciech Jaskowski, and Jurgen Schmidhuber. ¨ **Training agents using upside-down reinforcement learning**

Decision Transformer: Reinforcement Learning via Sequence Modeling

Lili Chen[*,1], **Kevin Lu**[*,1], **Aravind Rajeswaran**[2], **Kimin Lee**[1],
Aditya Grover[2], **Michael Laskin**[1], **Pieter Abbeel**[1], **Aravind Srinivas**[†,1], **Igor Mordatch**[†,3]

[*]equal contribution [†]equal advising

[1]UC Berkeley [2]Facebook AI Research [3]Google Brain

{lilichen, kzl}@berkeley.edu

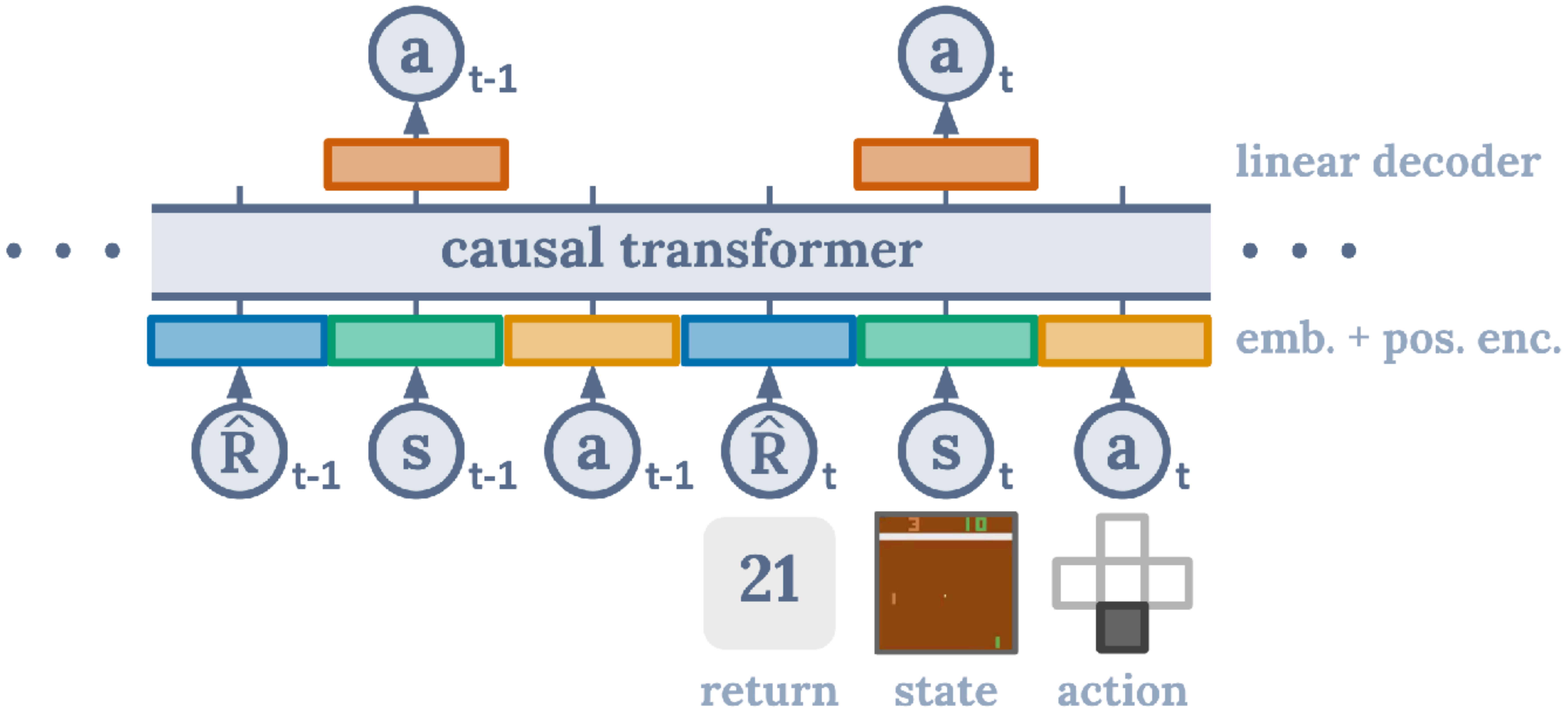
$$\hat{R} = \sum_{t=0}^{T-1} r_t$$

$s_0 \quad a_0 \quad r_0 \quad s_1 \quad a_1 \quad r_1 \quad \cdots\cdots\cdots$

$s_0 \quad a_0 \quad \hat{R}_0 \quad s_1 \quad a_1 \quad \hat{R}_1 \quad \cdots\cdots\cdots$

$$\hat{R}_0 = \sum_{t=0}^{T-1} r_t \qquad \hat{R}_1 = \sum_{t=1}^{T-1} r_t$$


$\hat{R}_{t-1}$




Introducing Decision Transformers on Hugging Face 🤗

Published March 28, 2022

Update on GitHub

edbeeching Edward Beeching

ThomasSimonini Thomas Simonini

Test Time

Start at initial state s_0

Specify the desired target return R_0

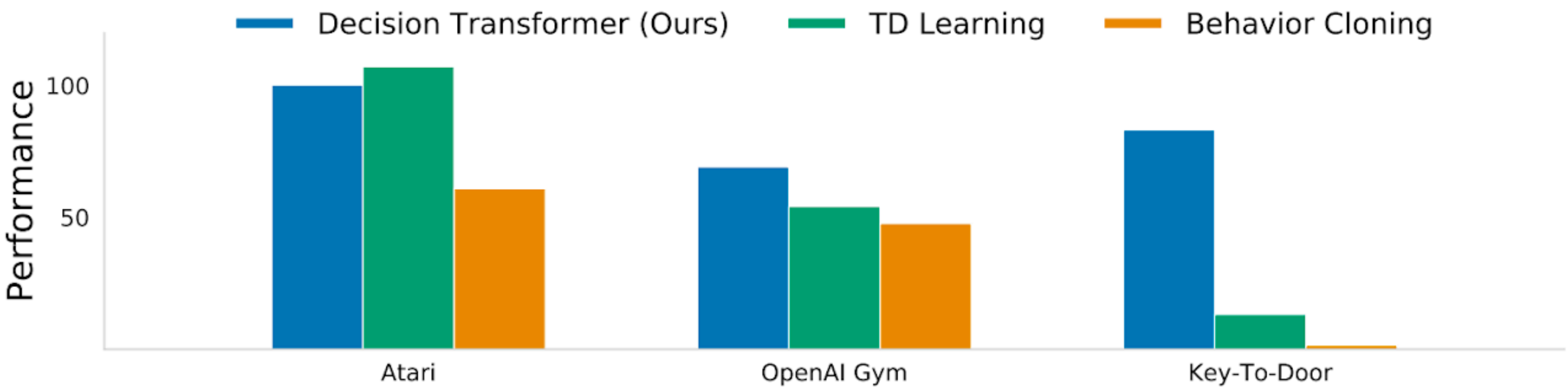
$a_0 = \text{Transformer}(R_0, s_0)$

Execute action, observe reward and next state (r_0, s_1)

Decrement the target return $R_1 = R_0 - r_0$

$a_1 = \text{Transformer}(R_0, s_0, a_0, R_1, s_1)$

Seems to work!



Today's class

- ☑ What is offline RL? Why do we need it for robots?

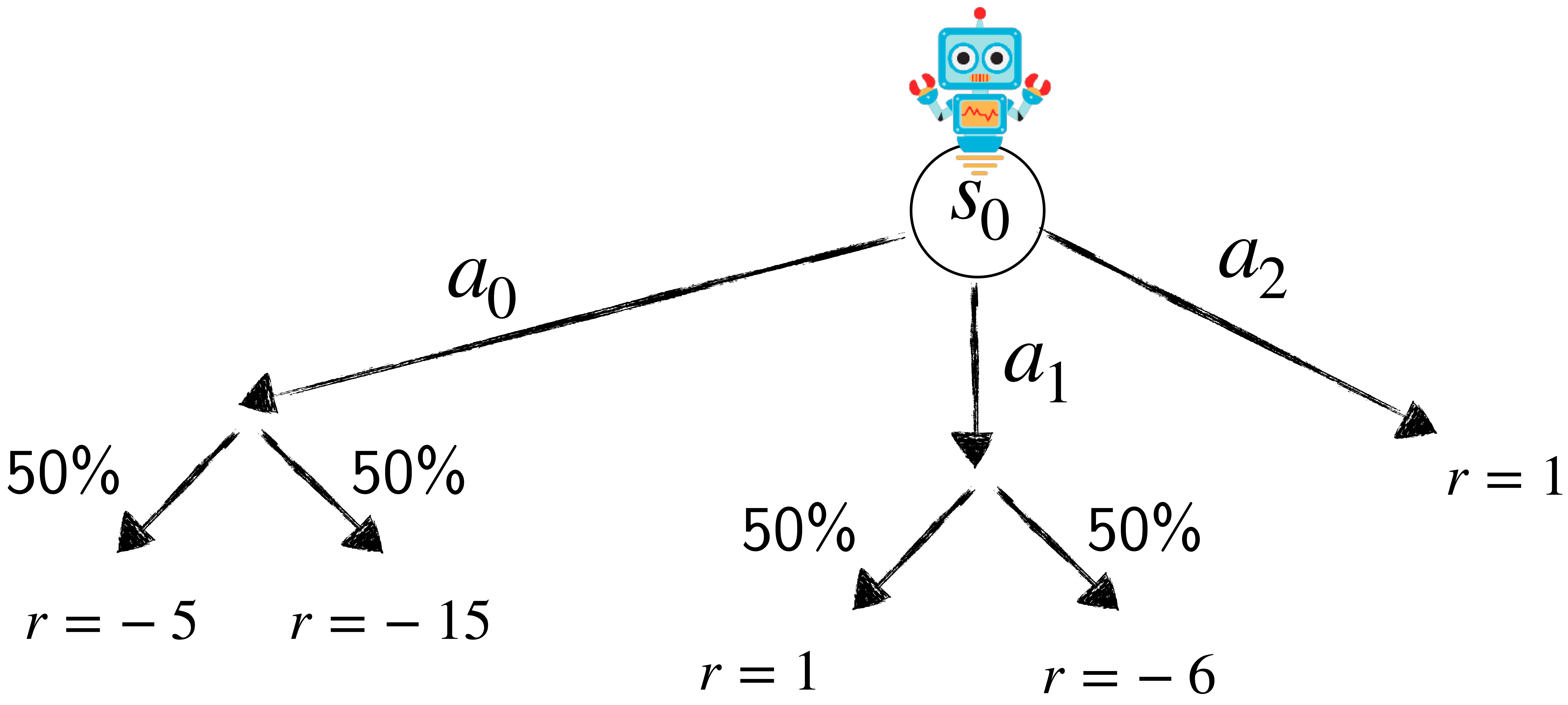
  (Enables safer training, leverages diverse experience)

- ☑ Paradigm 1: Offline RL via Pessimism
  - ☑ Problem with Q-learning (Incorrectly optimistic about unseen actions)
  - ☑ Pessimism to the rescue (Constrain policy to not deviate from data)

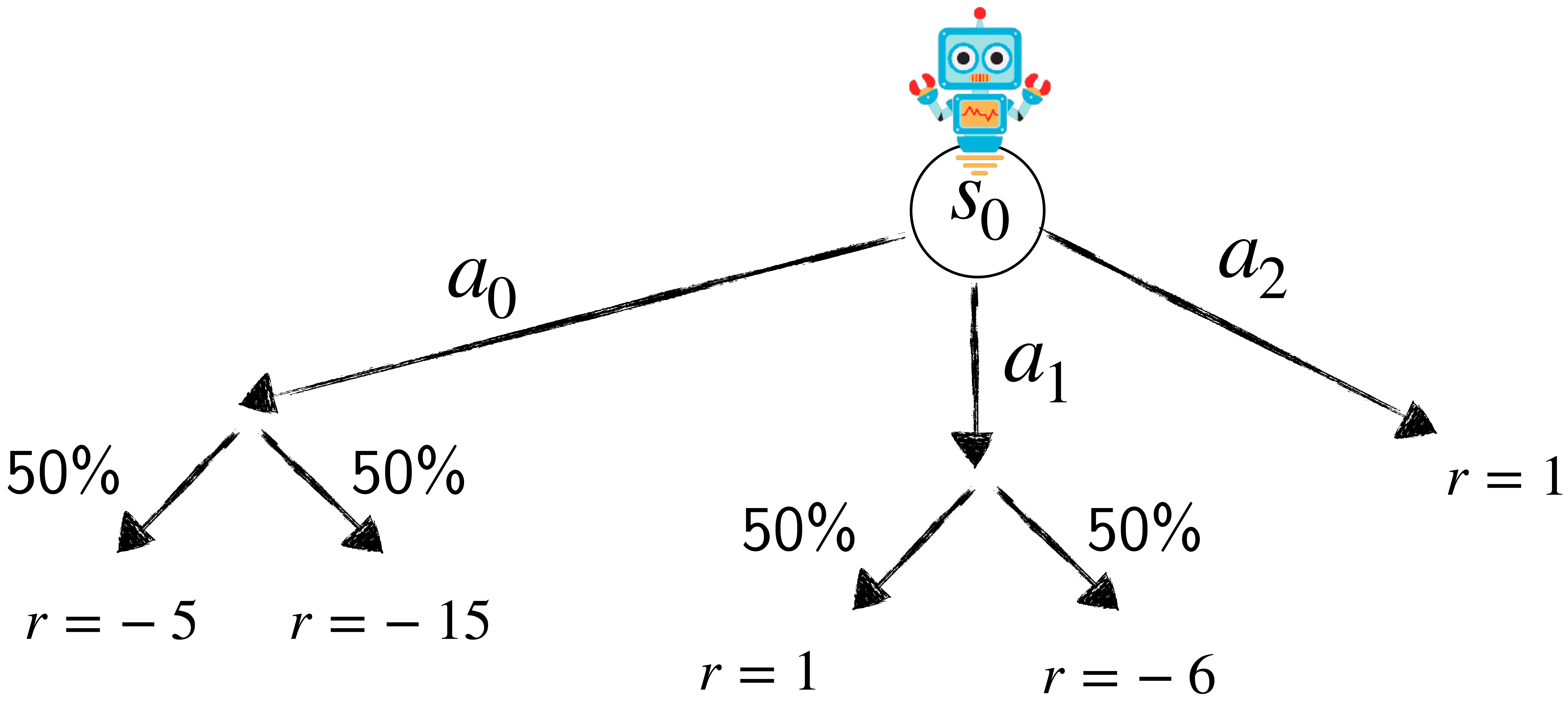
- ☐ Paradigm 2: RL via Supervised Learning
  - ☑ Return-conditioned Supervised Learning (Train policy to conditioned on return, Inference with a high return)
  - ☐ Problem in Stochastic MDPs

Activity!

Consider the following MDP



Consider the following MDP



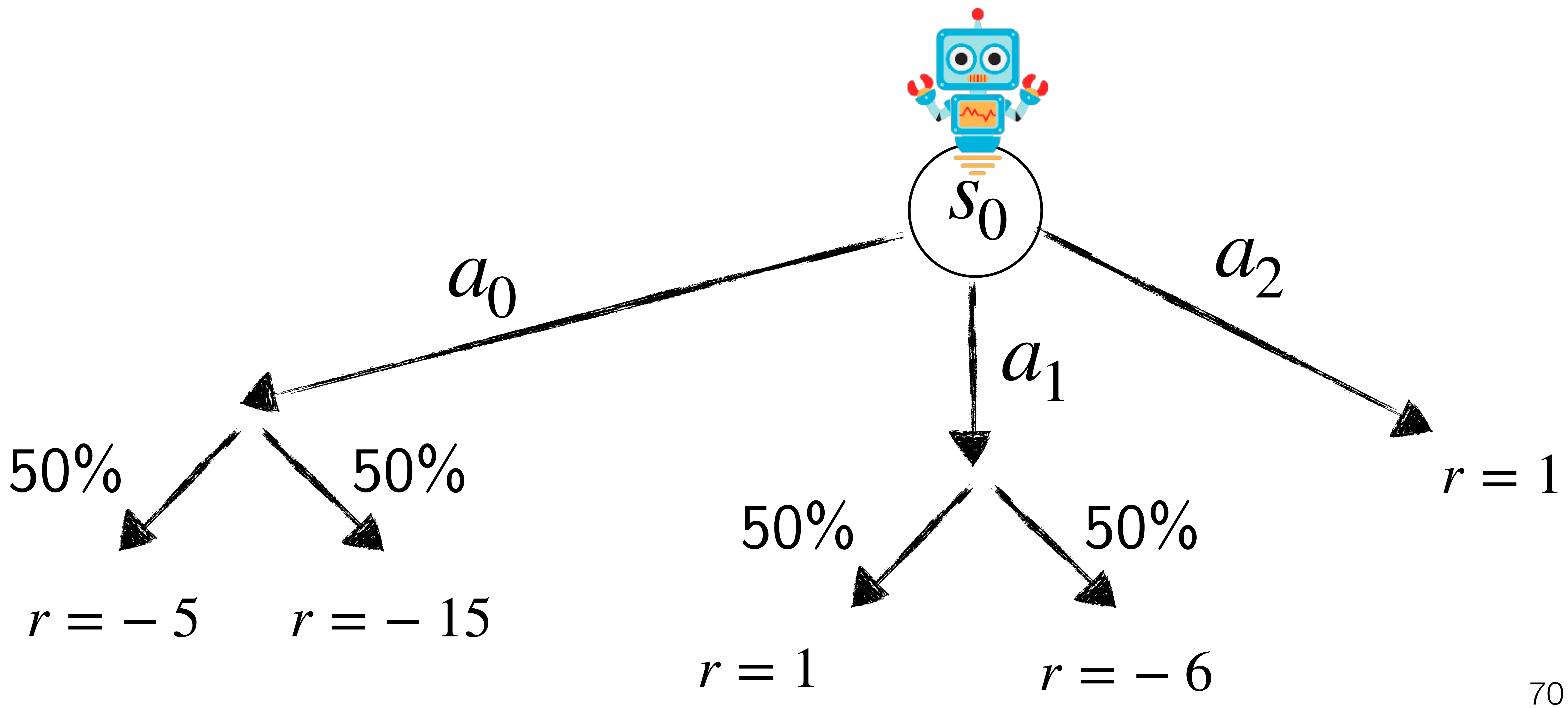
What is the optimal action? What will RvS pick?

Think-Pair-Share!

Think (30 sec): What is the optimal action? What would RvS play?

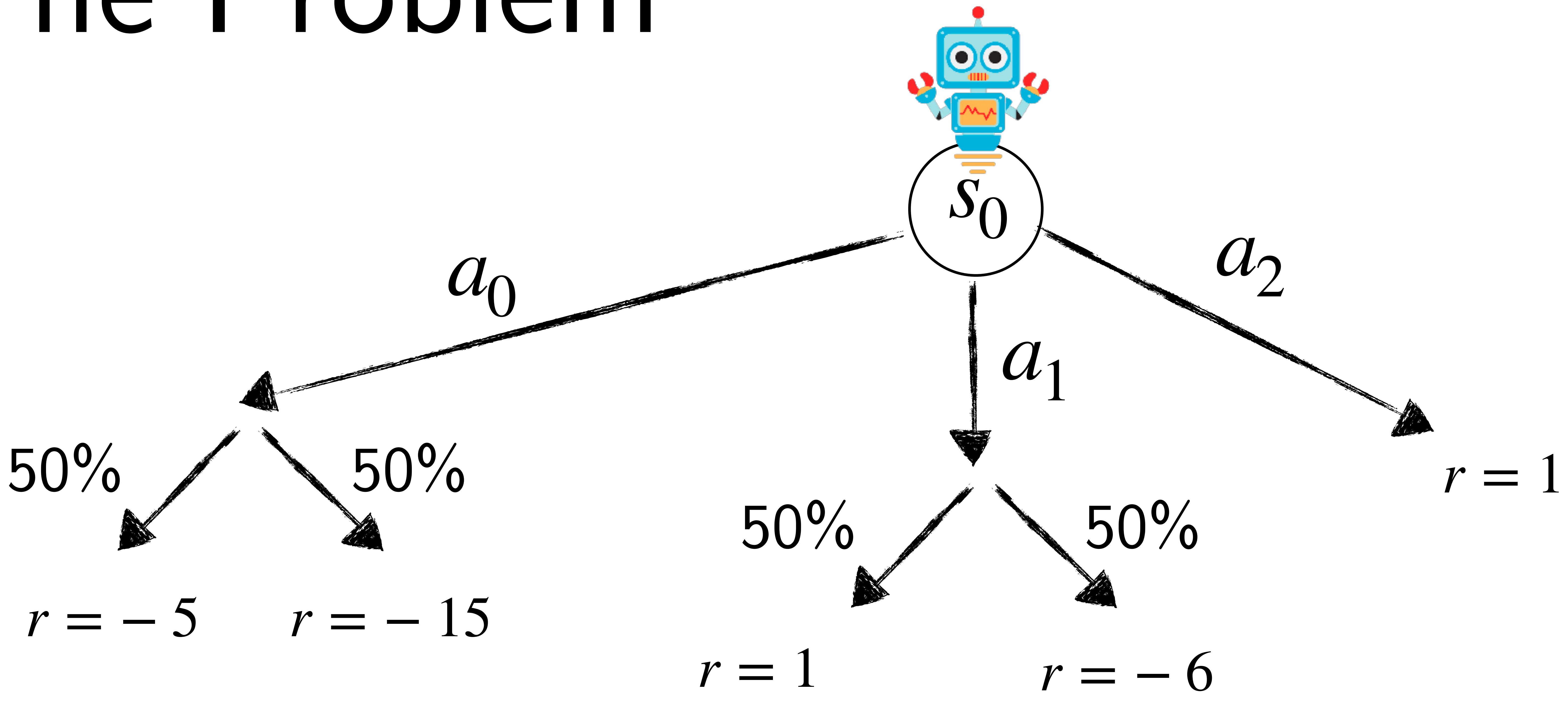
Pair: Find a partner

Share (45 sec): Partners exchange ideas



The Problem

No matter how much data it is trained on, RvS will always gamble and take a_1 some of the time rather than a_2 all of the time



Can prove that RvS fails to assign credit correctly when it got reward due to an action vs due to environment

Can't tell when it just got lucky

Today's class

- ☑ What is offline RL? Why do we need it for robots?

  (Enables safer training, leverages diverse experience)

- ☑ Paradigm 1: Offline RL via Pessimism
  - ☑ Problem with Q-learning (Incorrectly optimistic about unseen actions)
  - ☑ Pessimism to the rescue (Constrain policy to not deviate from data)

- ☑ Paradigm 2: RL via Supervised Learning
  - ☑ Return-conditioned Supervised Learning (Train policy to conditioned on return, Inference with a high return)
  - ☑ Problem in Stochastic MDPs (Fails to account for luck)