Decision Transformers

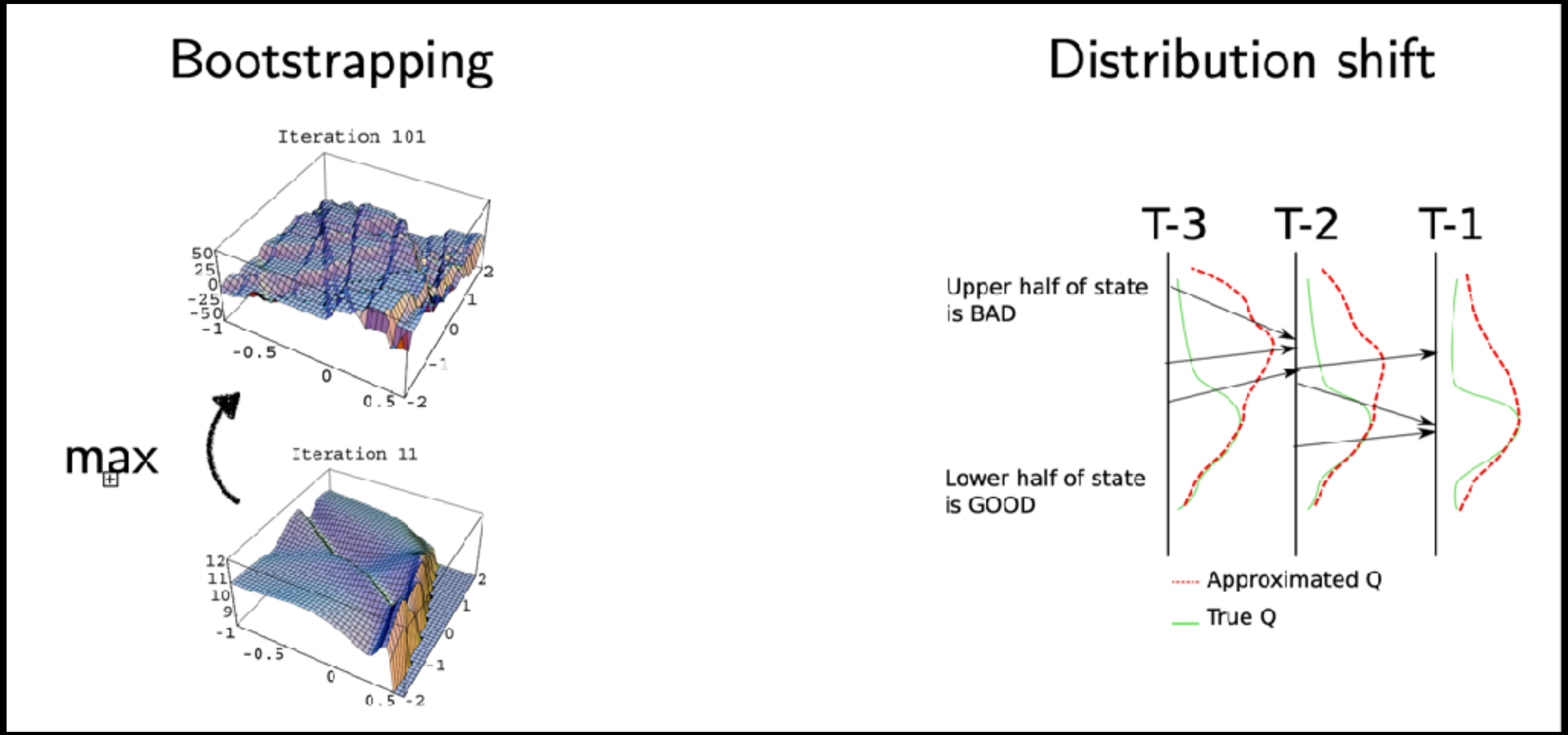
Sanjiban Choudhury

Cornell Bowers CIS
Computer Science

Reinforcement Learning is Hard ...

Many horror stories of RL!

Nightmares of
Policy Optimization



The deadly triad

From Sutton and Barto

- The risk of divergence arises whenever we combine three things:

  1. Function approximation

     significantly generalizing from large numbers of examples

  2. Bootstrapping

     learning value estimates from other value estimates,
     as in dynamic programming and temporal-difference learning

  3. Off-policy learning (Why is dynamic programming off-policy?)

     learning about a policy from data not due to that policy,
     as in Q-learning, where we learn about the greedy policy from
     data with a necessarily more exploratory policy

- Any two without the third is ok

Need many tricks to make RL work in practice!

Rainbow: Combining Improvements in Deep Reinforcement Learning

Matteo Hessel DeepMind
Joseph Modayil DeepMind
Hado van Hasselt DeepMind
Tom Schaul DeepMind
Georg Ostrovski DeepMind
Will Dabney DeepMind
Dan Horgan DeepMind
Bilal Piot DeepMind
Mohammad Azar DeepMind
David Silver DeepMind

Double Q Learning

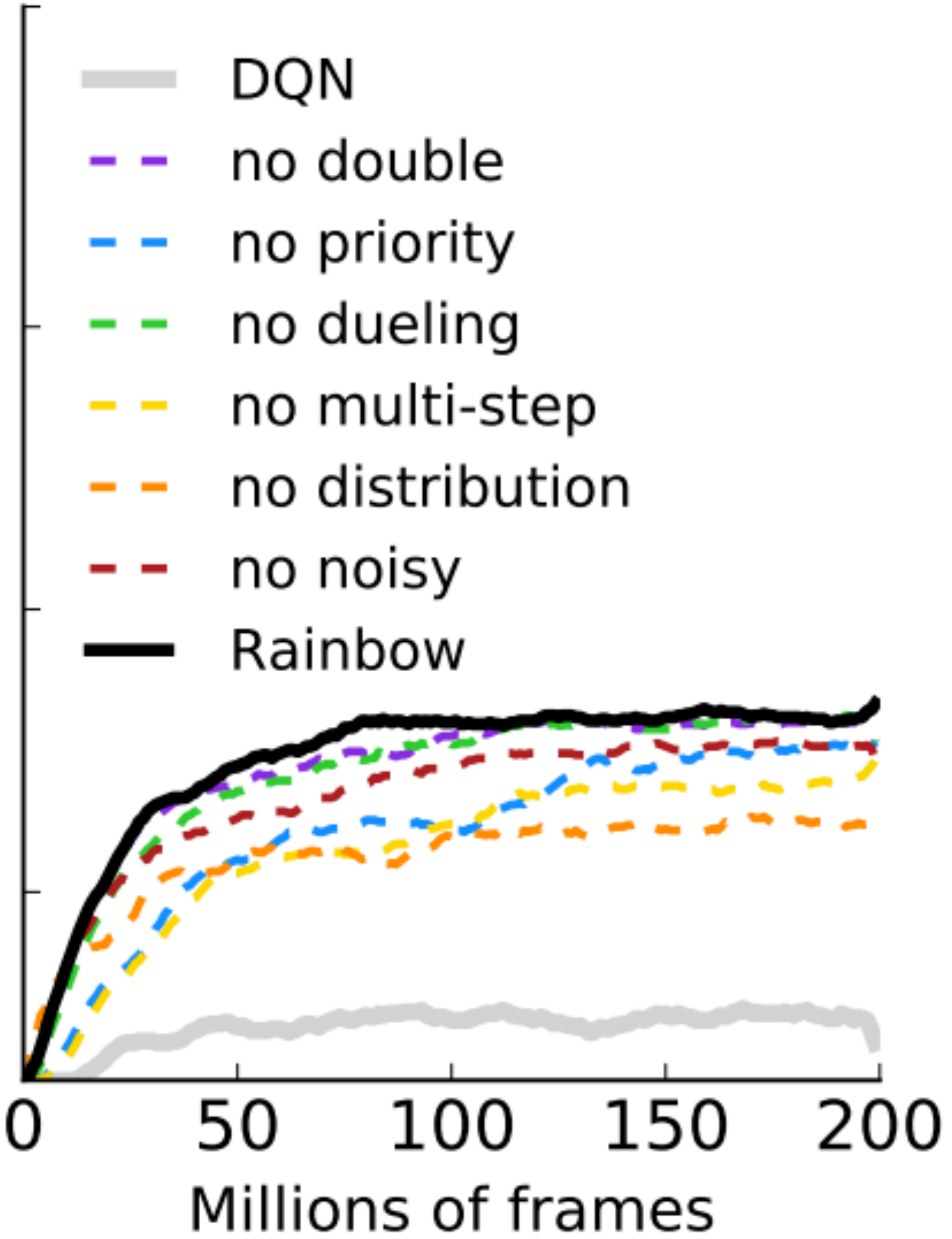
Prioritized Replay

Dueling Networks

Multi-step Learning

Distributional RL

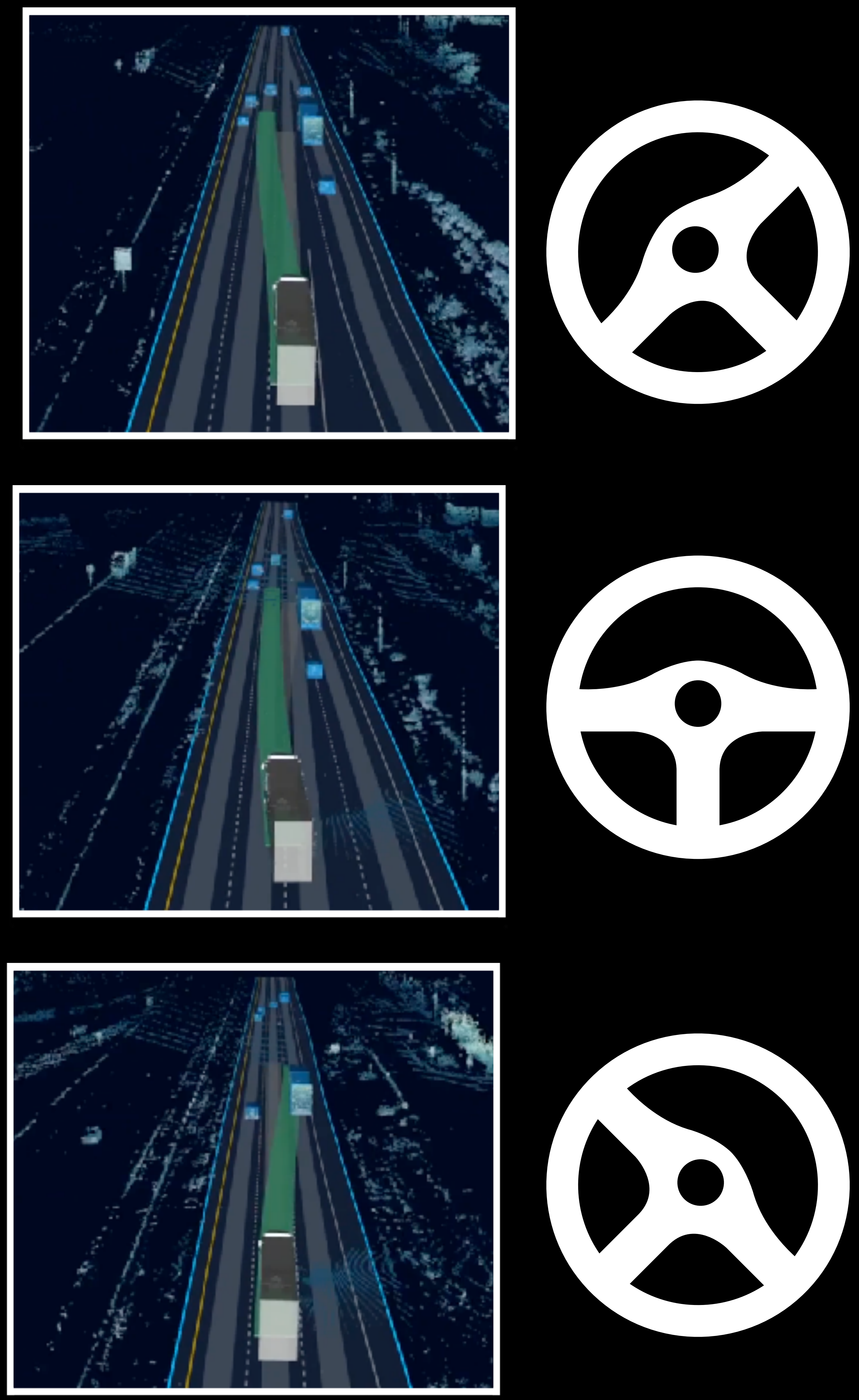
Noisy Nets



Is there any hope?

SUPERVISED LEARNING

#1 Get Data



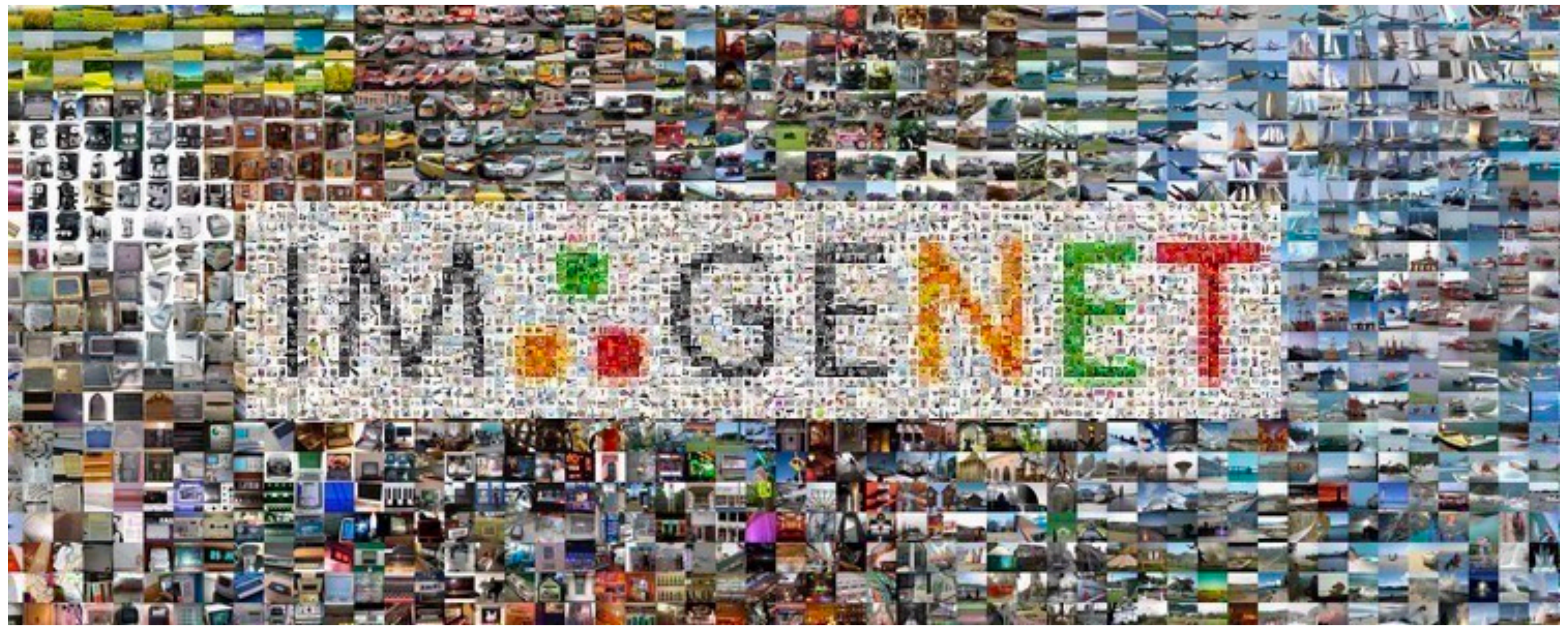
Input (s)   Output (a)

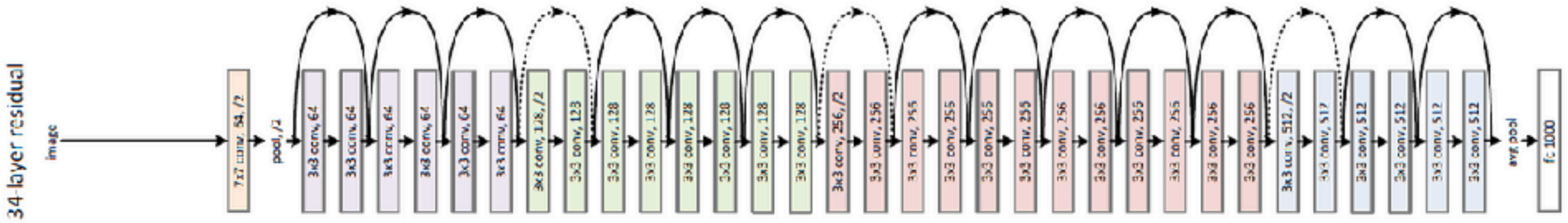
#2
Train
Policy

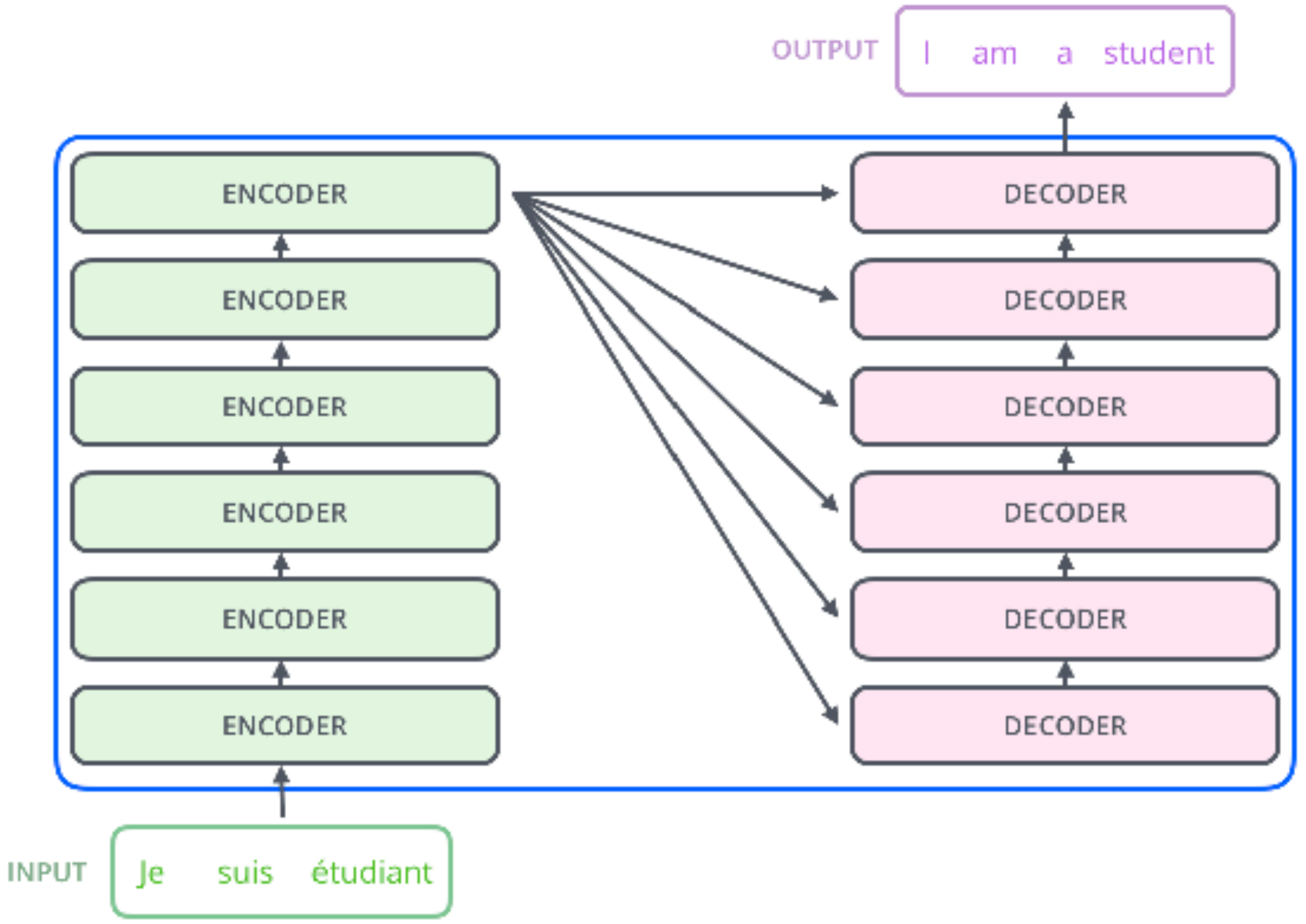
$$\pi : s \rightarrow a$$

#3 Deploy!

Supervised Learning success stories







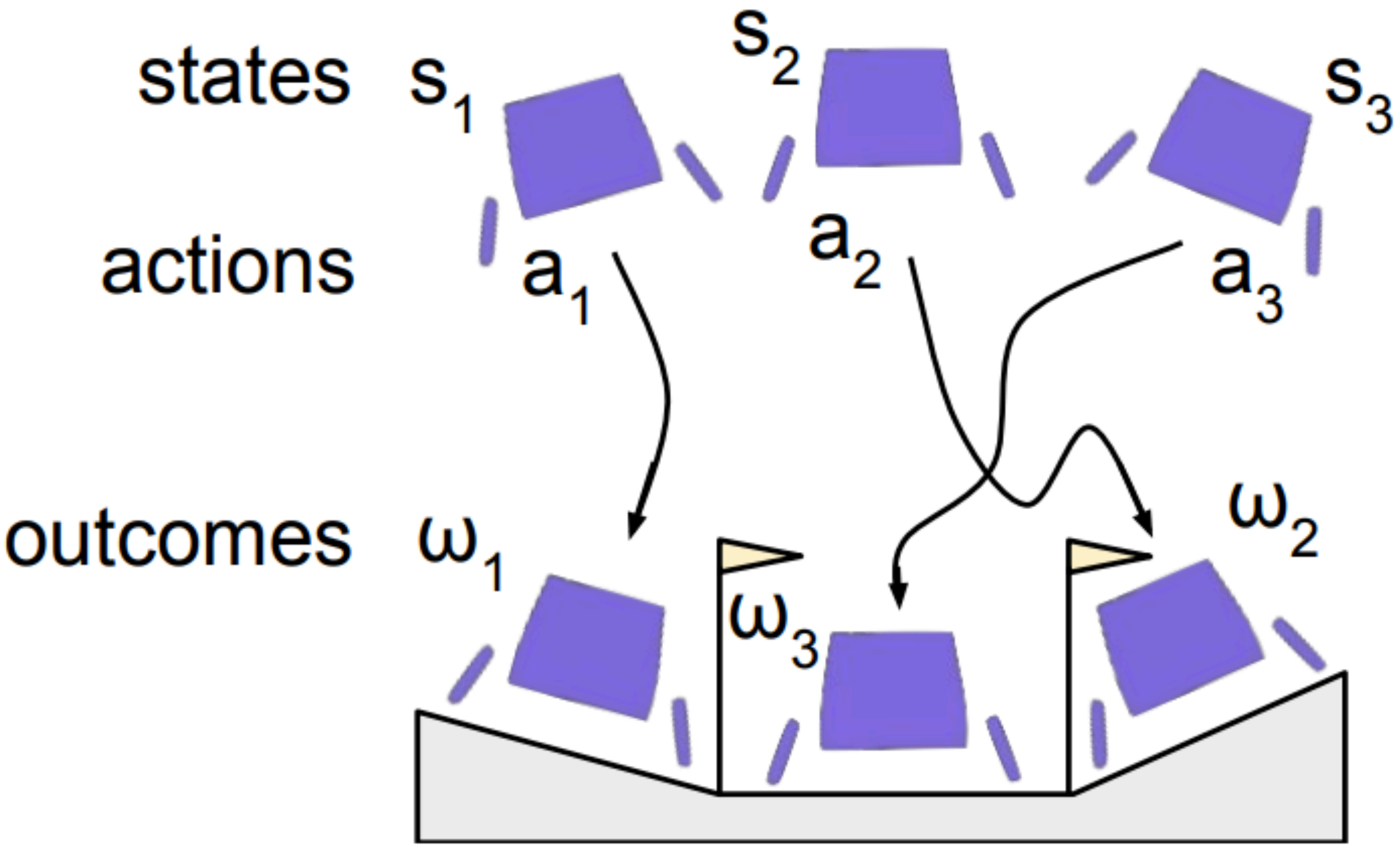
IDEA:

Can we make
Reinforcement Learning
look like
Supervised Learning?

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

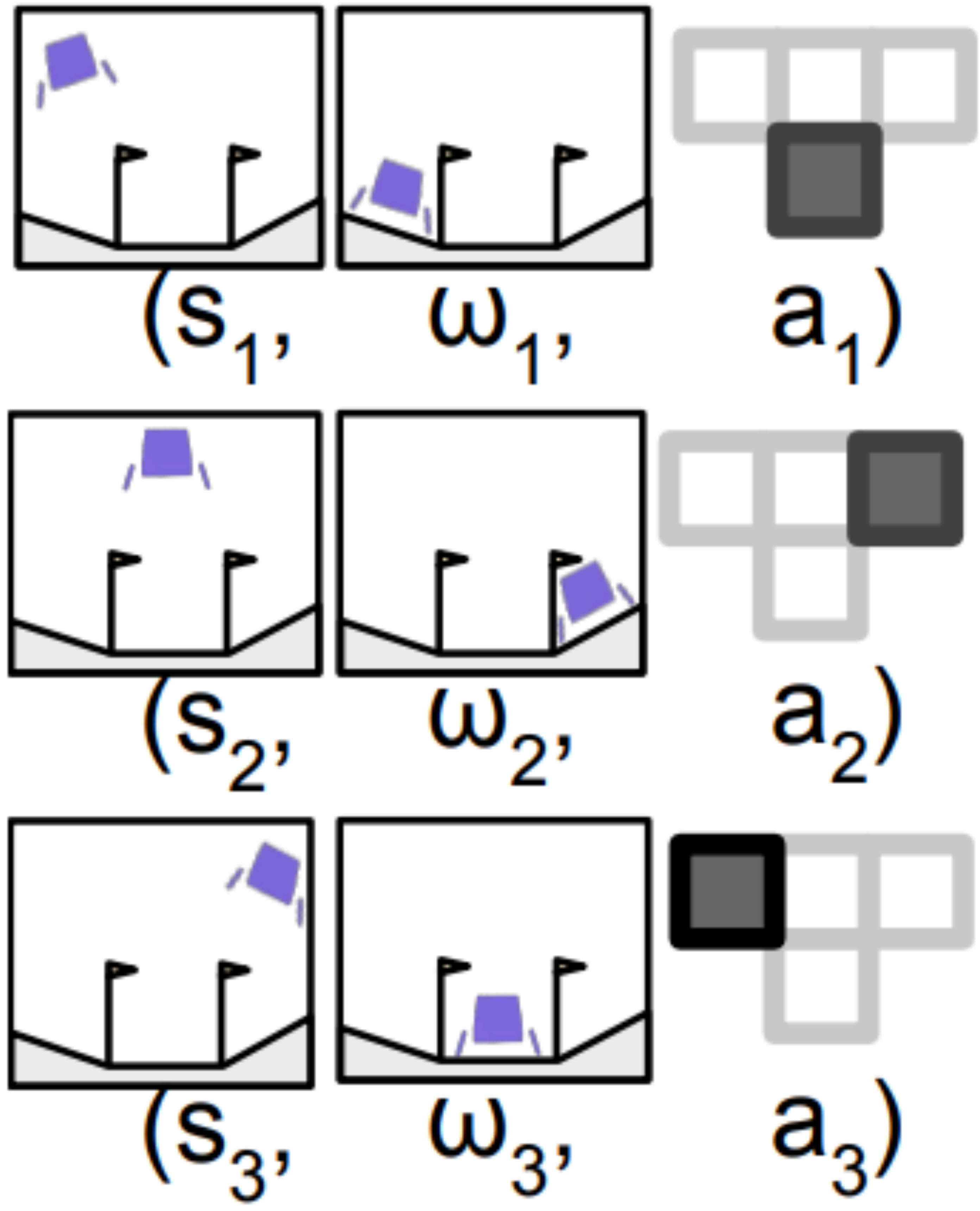
Scott Emmons[1], **Benjamin Eysenbach**[2], **Ilya Kostrikov**[1], **Sergey Levine**[1]
[1]UC Berkeley, [2]Carnegie Mellon University
emmons@berkeley.edu

The Idea



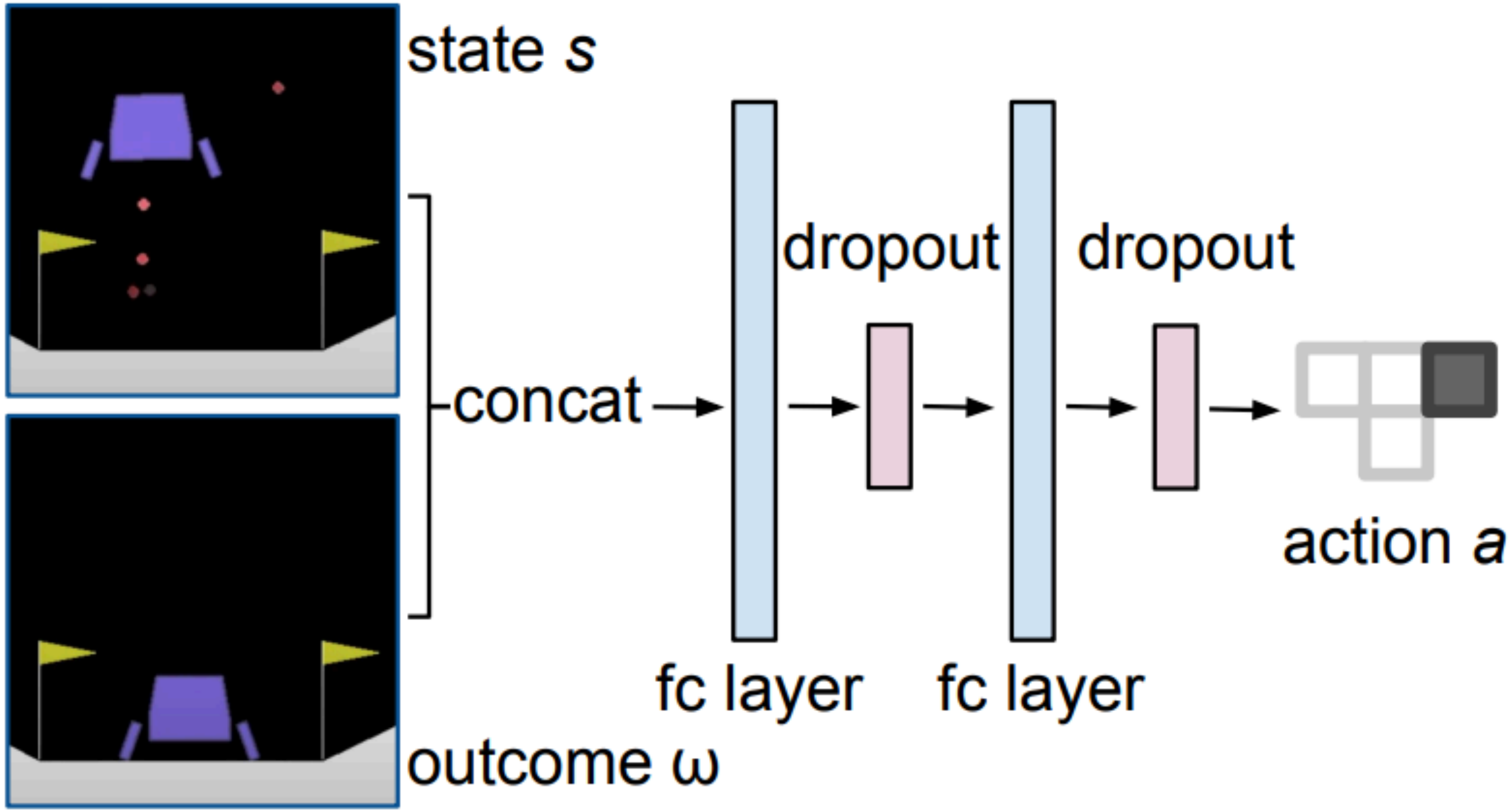
(a) replay buffer

The Idea



(b) training dataset

The Idea



(c) network architecture

The Algorithm

$$\max_{\theta} \overset{\text{For all trajectories:}}{\sum_{\tau \in \mathcal{D}}} \overset{\text{For all timesteps in that trajectory:}}{\sum_{1 \leq t \leq |\tau|}} \overset{\text{For all achieved outcomes:}}{\mathbb{E}_{\omega \sim f(\omega \mid \tau_{t:H})}} [\log \pi_\theta(a_t \mid s_t, \omega)].$$

Algorithm 1 RvS-Learning

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

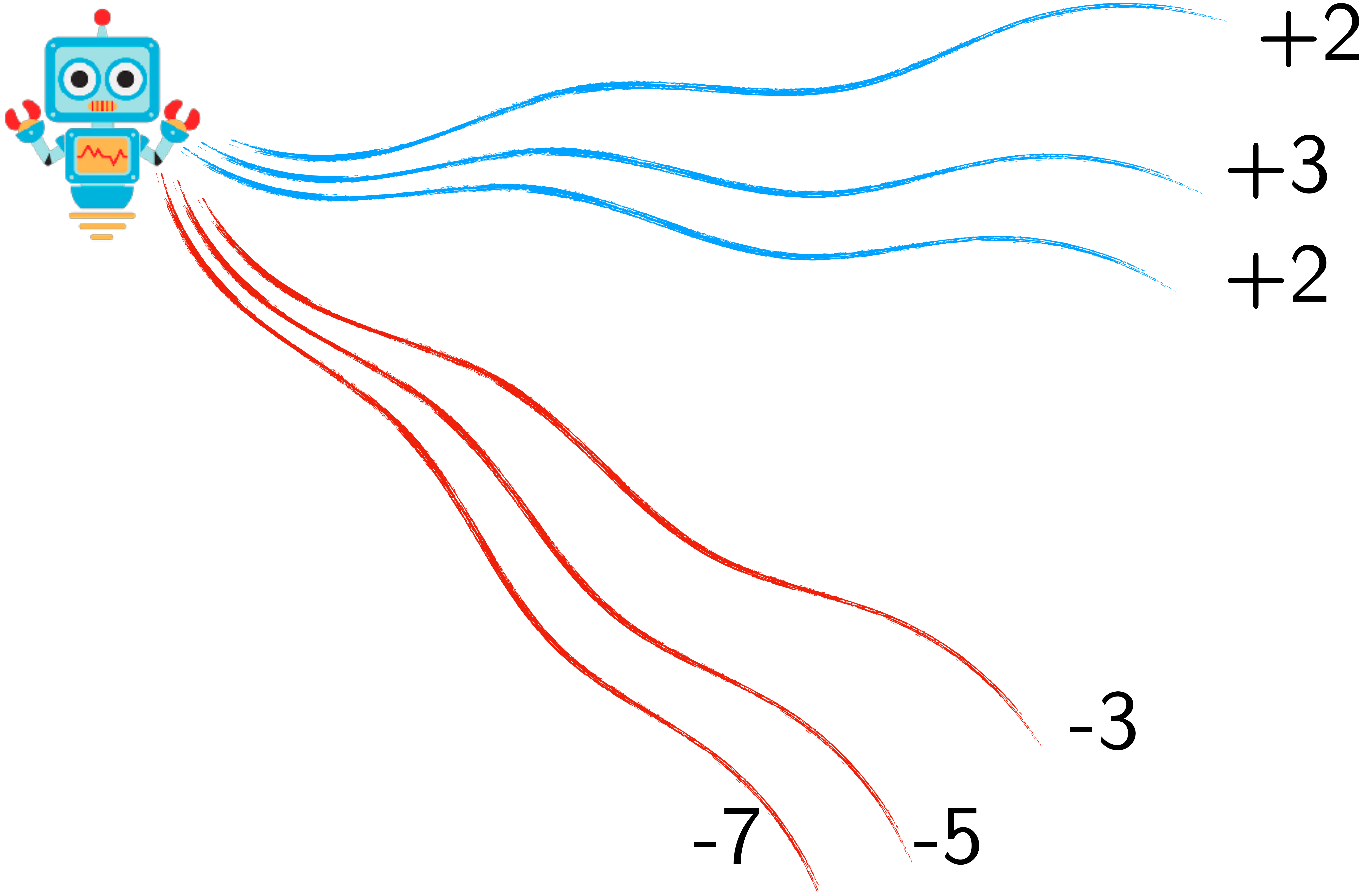
Aviral Kumar, Xue Bin Peng, and Sergey Levine. **Reward-conditioned policies**

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

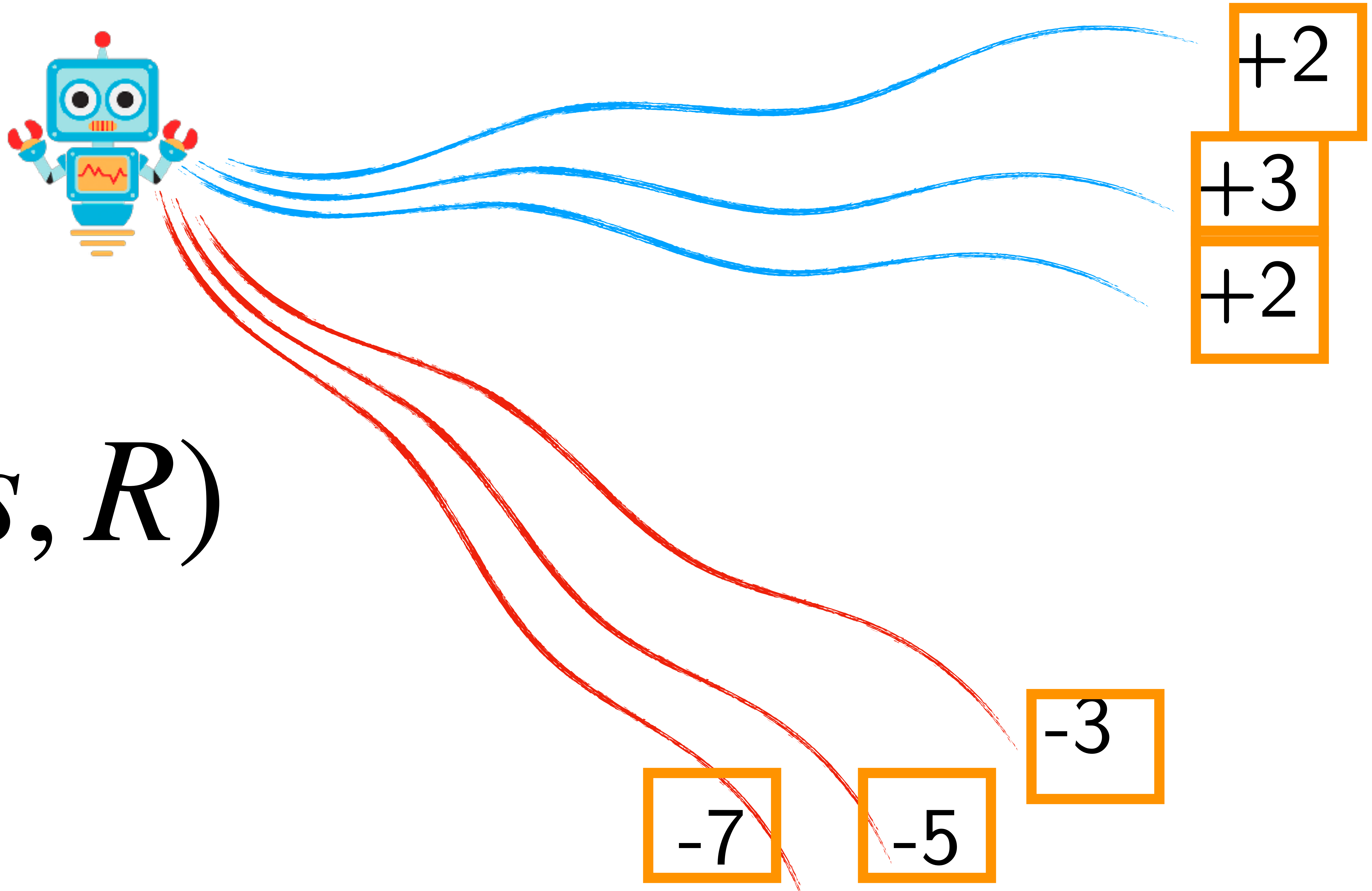
Rupesh Kumar Srivastava, Pranav Shyam, Filipe Mutz, Wojciech Jaskowski, and Jurgen Schmidhuber. ¨ **Training agents using upside-down reinforcement learning**

Do I really need to condition?

Consider the following MDP

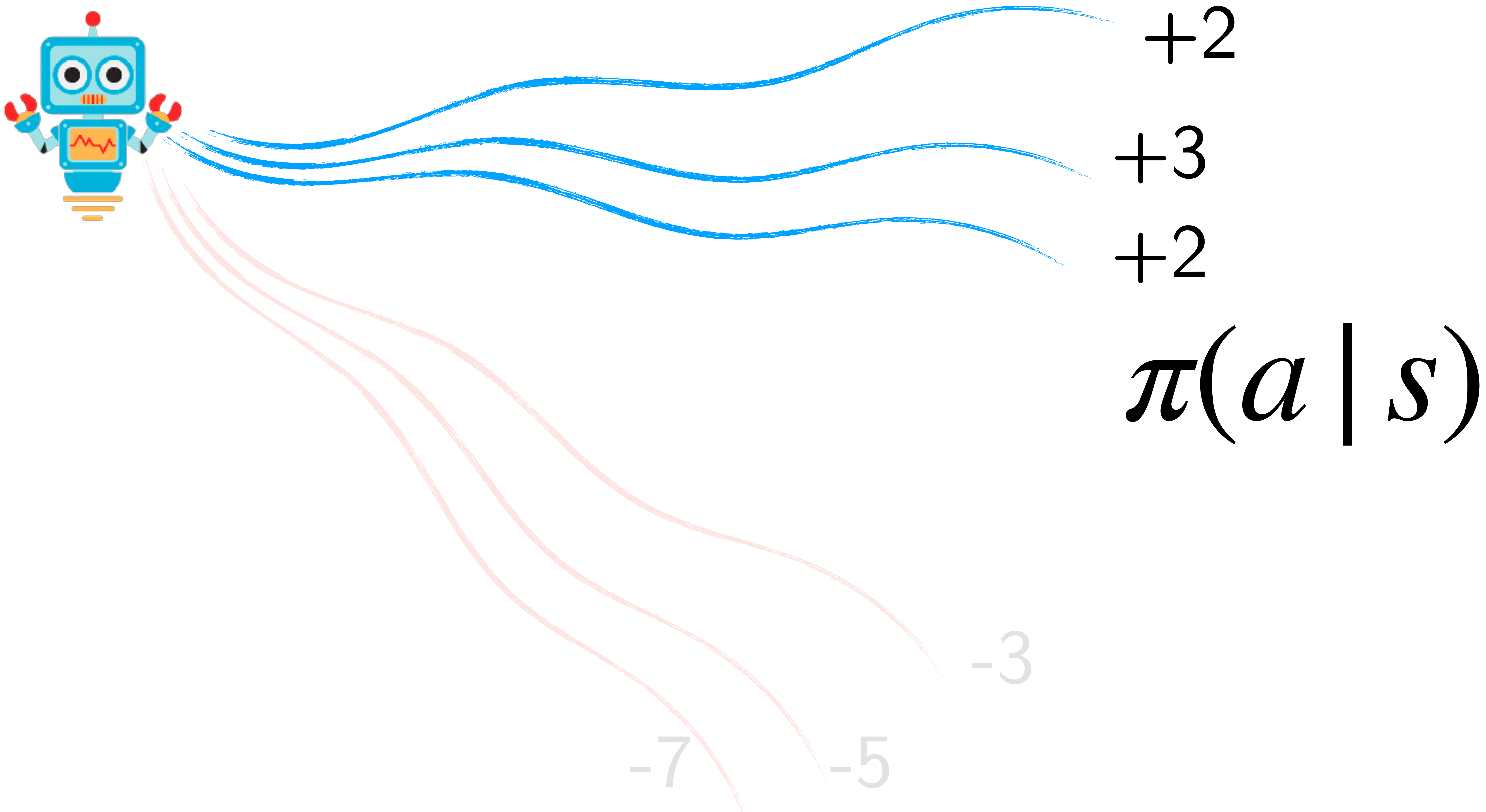


Option 1: Return Conditioned Policy



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

Option 2: Train a policy on top returns!



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

Can we make this a bit more fancier?

1. Handle noisy returns
2. Collect data on-policy

From Policy Gradient to Policy Search

Algorithm 1 Advantage-Weighted Regression

$\pi_1 \leftarrow$ random policy
$\mathcal{D} \leftarrow \emptyset$
for iteration $k = 1, ..., k_{\max}$ **do**
    add trajectories $\{\tau_i\}$ sampled via π_k to $\mathcal{D}$
    $V_k^{\mathcal{D}} \leftarrow \arg\min_V \ \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}} \left[\left|\left| \mathcal{R}_{\mathbf{s},\mathbf{a}}^{\mathcal{D}} - V(\mathbf{s}) \right|\right|^2 \right]$ Supervised Learning!
    $\pi_{k+1} \leftarrow \arg\max_\pi \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}} \left[\log \pi(\mathbf{a}|\mathbf{s}) \exp\left(\frac{1}{\beta} \left(\mathcal{R}_{\mathbf{s},\mathbf{a}}^{\mathcal{D}} - V_k^{\mathcal{D}}(\mathbf{s}) \right) \right) \right]$ Supervised Learning!
end for

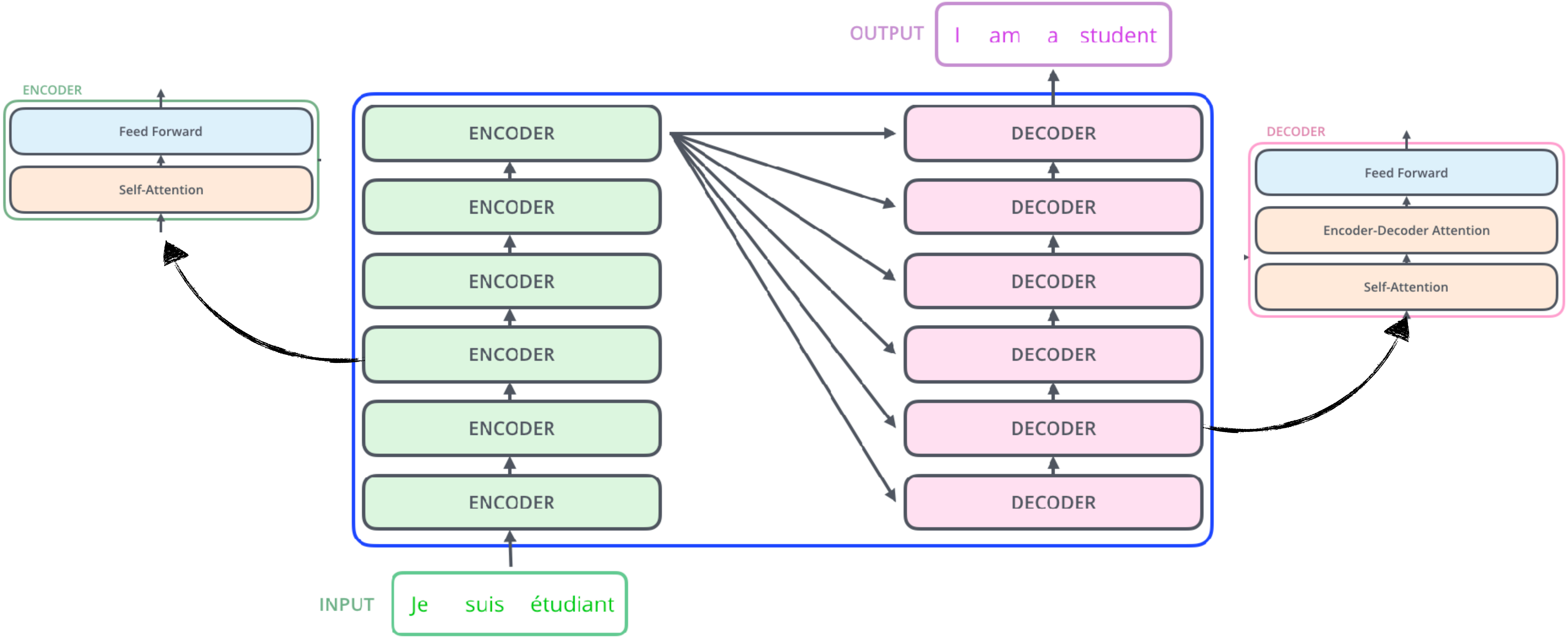
Peng et al, 2019

I thought we were going to talk about transformers?

Transformers

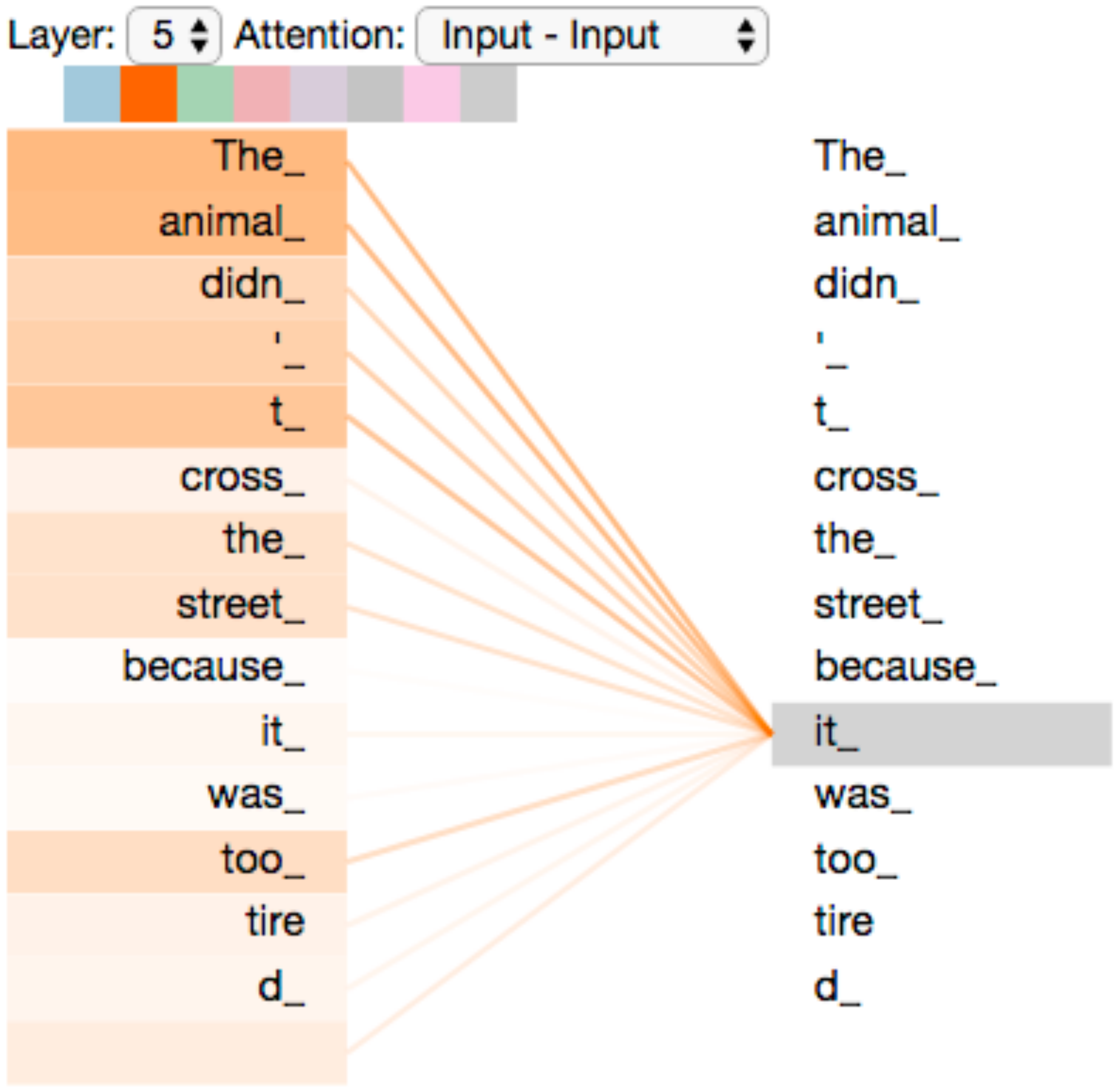
Given sequence of English words, predict sequence of French

Transformer Architecture

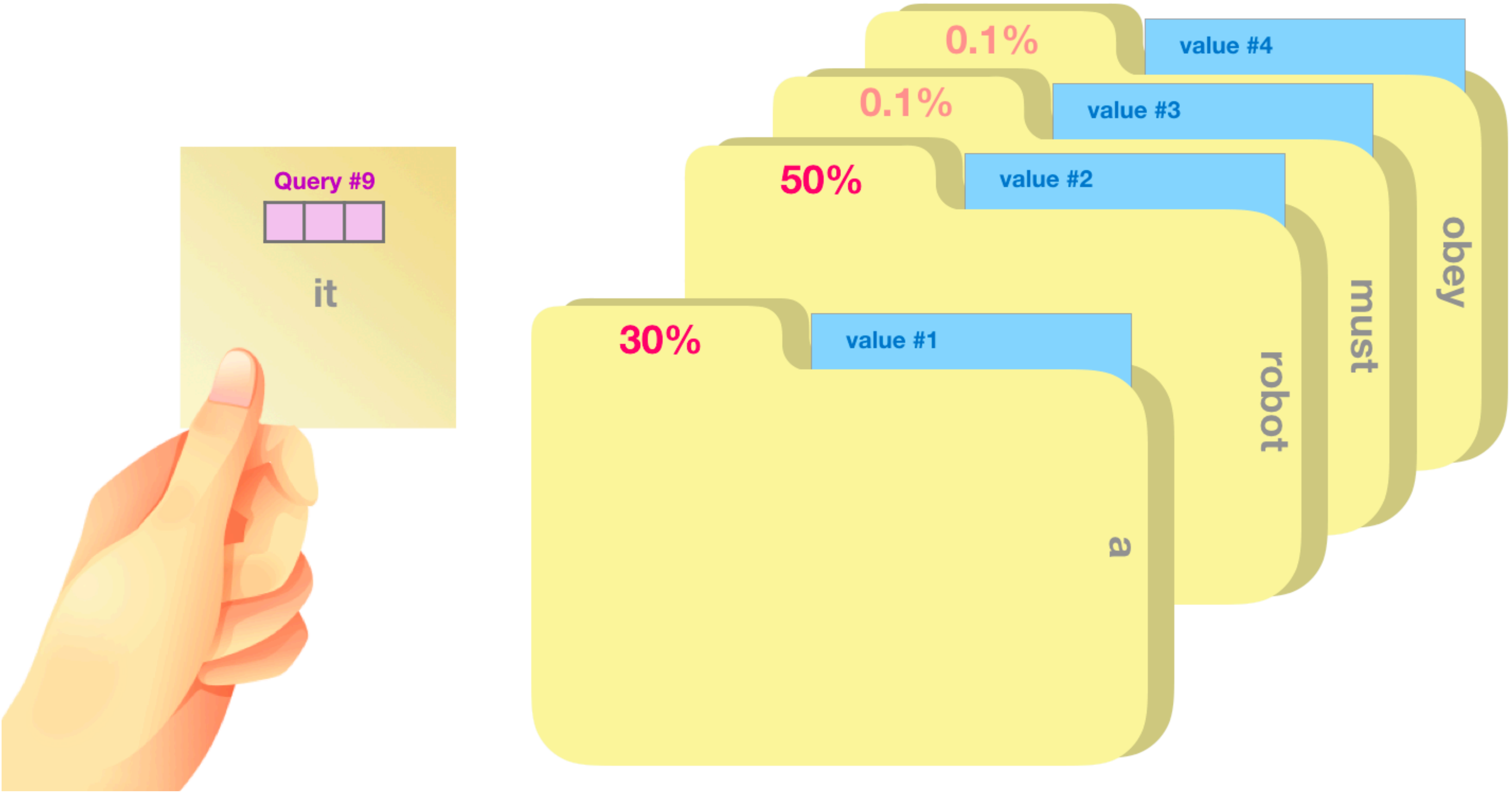


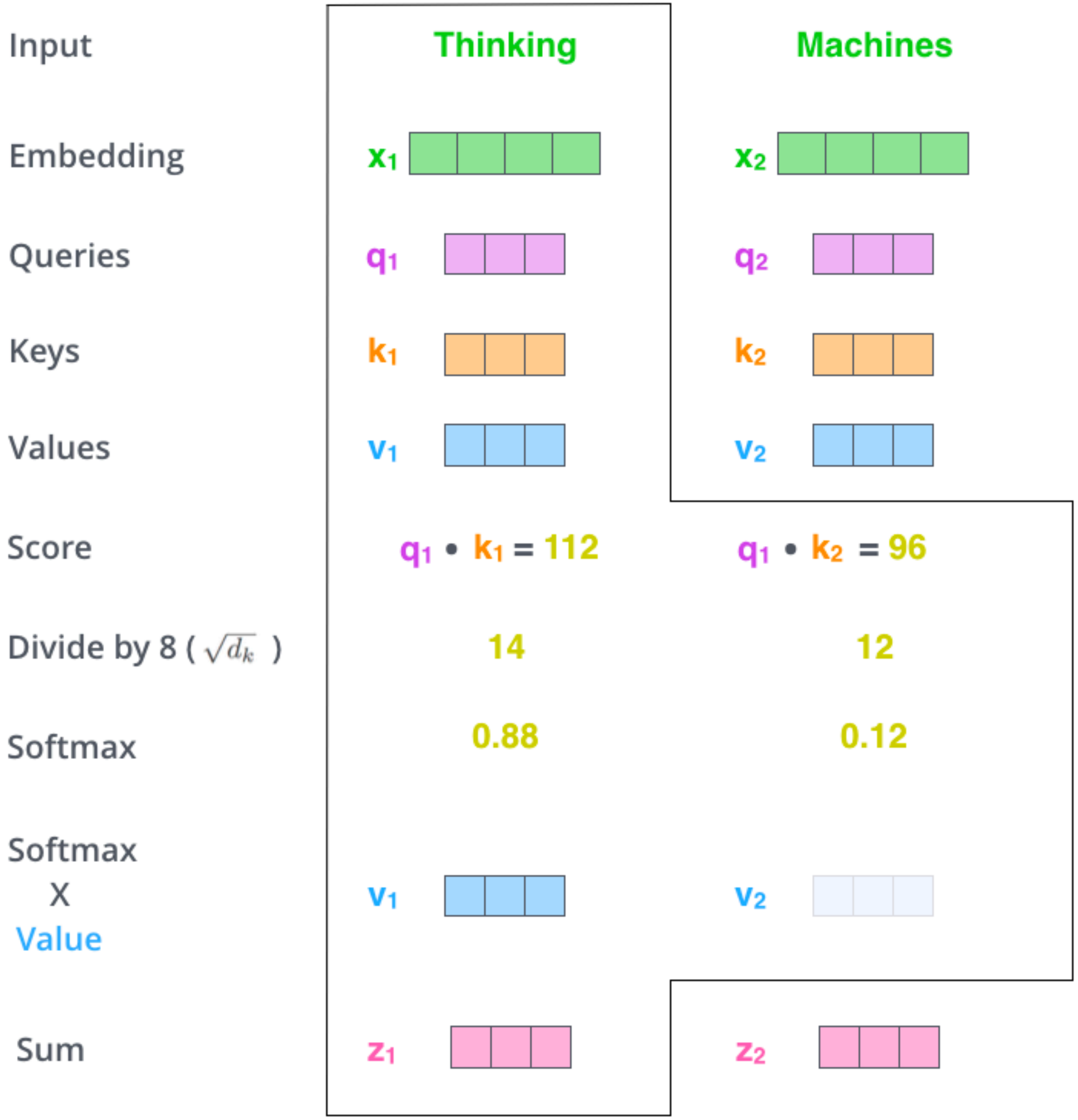
Visualizing attentions

”`The animal didn't cross the street because it was too tired`”



Attention as a soft-memory look up




Input
Thinking
Machines
Embedding
x1
x2
Queries
q1
q2
Keys
k1
k2
Values
v1
v2
Score
q1 • k1 = 112
q1 • k2 = 96
Divide by 8 ($\sqrt{d_k}$)
14
12
Softmax
0.88
0.12
Softmax
X
Value
v1
v2
Sum
z1
z2

Back to Decision Making

Decision Transformer: Reinforcement Learning via Sequence Modeling

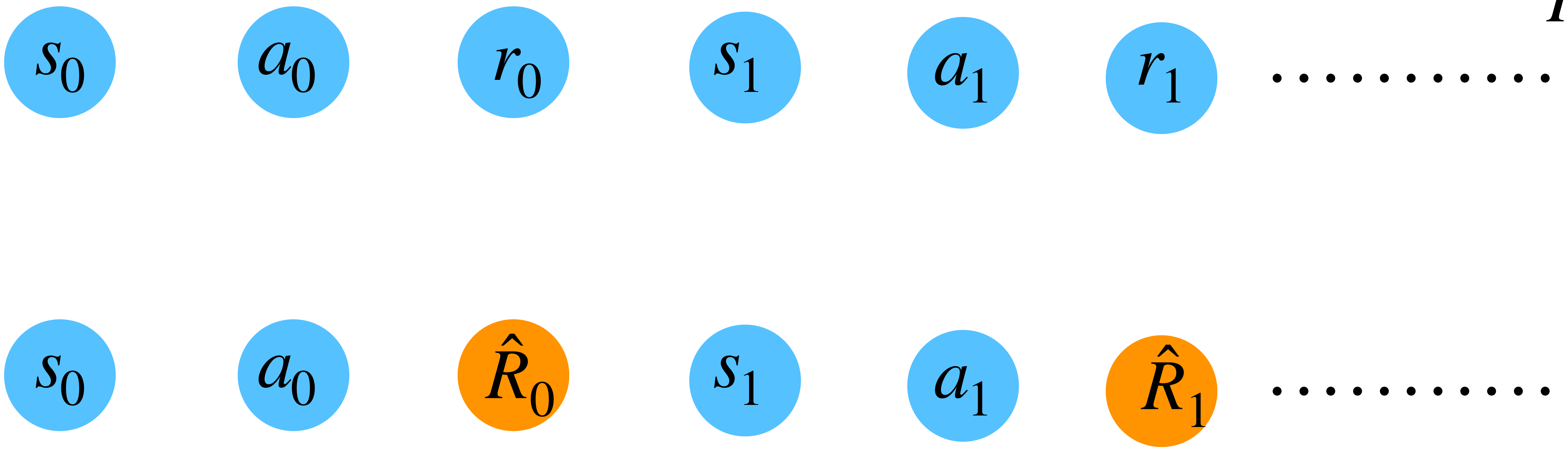
Lili Chen[*,1], **Kevin Lu**[*,1], **Aravind Rajeswaran**[2], **Kimin Lee**[1],
Aditya Grover[2], **Michael Laskin**[1], **Pieter Abbeel**[1], **Aravind Srinivas**[†,1], **Igor Mordatch**[†,3]

[*]equal contribution [†]equal advising

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

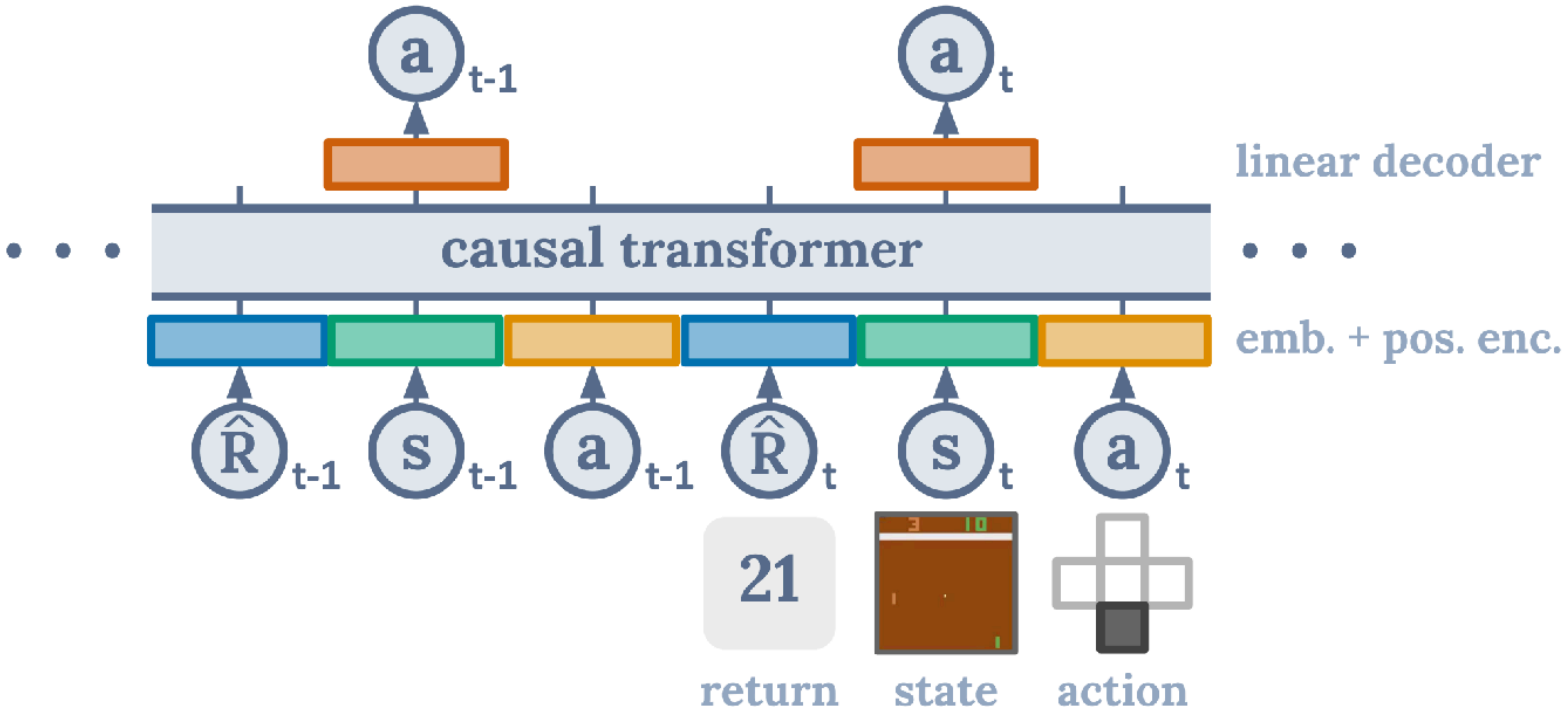
{lilichen, kzl}@berkeley.edu

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



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

• • • $\hat{R}_{t-1}$



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

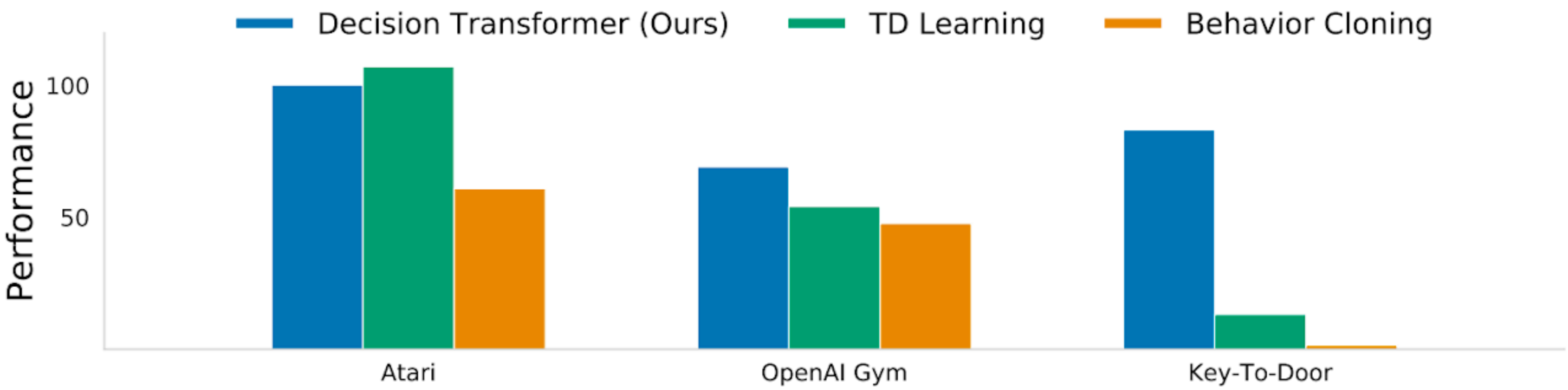
$a_0 = \text{Transformer}(R_0, s_0)$

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!



Seems to work!

Game	DT (Ours)	CQL	QR-DQN	REM	BC
Breakout	$\mathbf{267.5 \pm 97.5}$	211.1	17.1	8.9	138.9 ± 61.7
Qbert	15.4 ± 11.4	**104.2**	0.0	0.0	17.3 ± 14.7
Pong	106.1 ± 8.1	**111.9**	18.0	0.5	85.2 ± 20.0
Seaquest	$\mathbf{2.5 \pm 0.4}$	1.7	0.4	0.7	2.1 ± 0.3

Atari

Seems to work!

Dataset	Environment	DT (Ours)	CQL	BEAR	BRAC-v	AWR	BC
Medium-Expert	HalfCheetah	**86.8 ± 1.3**	62.4	53.4	41.9	52.7	59.9
Medium-Expert	Hopper	107.6 ± 1.8	**111.0**	96.3	0.8	27.1	79.6
Medium-Expert	Walker	**108.1 ± 0.2**	98.7	40.1	81.6	53.8	36.6
Medium-Expert	Reacher	**89.1 ± 1.3**	30.6	-	-	-	73.3
Medium	HalfCheetah	42.6 ± 0.1	44.4	41.7	**46.3**	37.4	43.1
Medium	Hopper	**67.6 ± 1.0**	58.0	52.1	31.1	35.9	63.9
Medium	Walker	74.0 ± 1.4	79.2	59.1	**81.1**	17.4	77.3
Medium	Reacher	**51.2 ± 3.4**	26.0	-	-	-	**48.9**
Medium-Replay	HalfCheetah	36.6 ± 0.8	46.2	38.6	**47.7**	40.3	4.3
Medium-Replay	Hopper	**82.7 ± 7.0**	48.6	33.7	0.6	28.4	27.6
Medium-Replay	Walker	**66.6 ± 3.0**	26.7	19.2	0.9	15.5	36.9
Medium-Replay	Reacher	**18.0 ± 2.4**	**19.0**	-	-	-	5.4
Average (Without Reacher)		**74.7**	63.9	48.2	36.9	34.3	46.4
Average (All Settings)		**69.2**	54.2	-	-	-	47.7

D4RL

Why does context length matter?

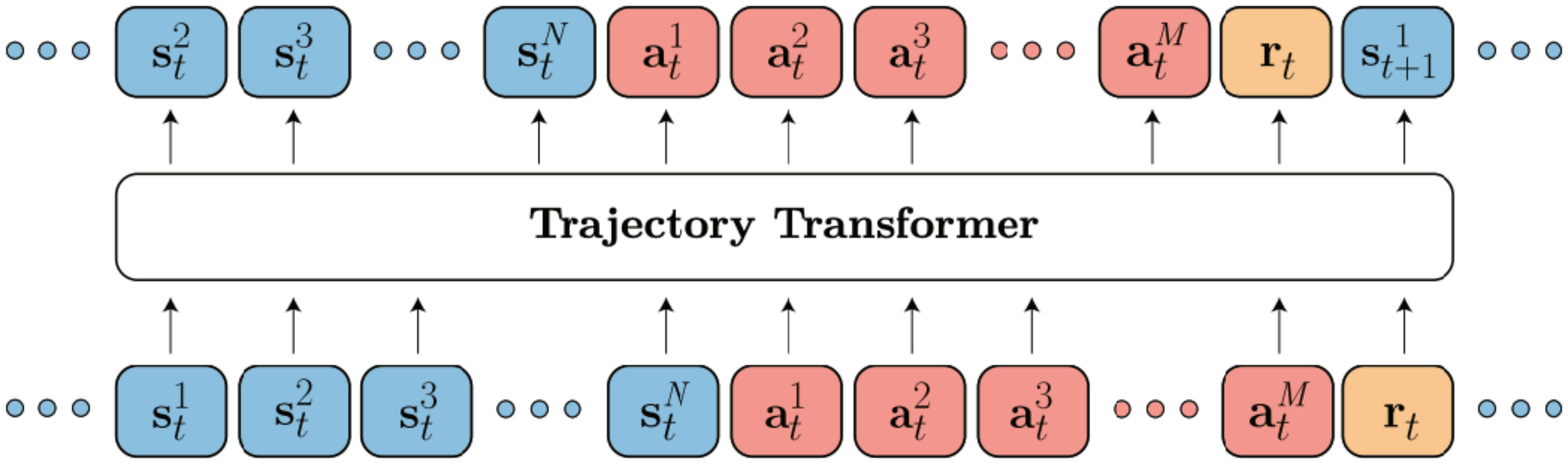
K=50

Game	DT (Ours)	DT with no context ($K = 1$)
Breakout	$\mathbf{267.5 \pm 97.5}$	73.9 ± 10
Qbert	$\mathbf{15.1 \pm 11.4}$	13.6 ± 11.3
Pong	$\mathbf{106.1 \pm 8.1}$	2.5 ± 0.2
Seaquest	$\mathbf{2.5 \pm 0.4}$	0.6 ± 0.1

Concurrent paper

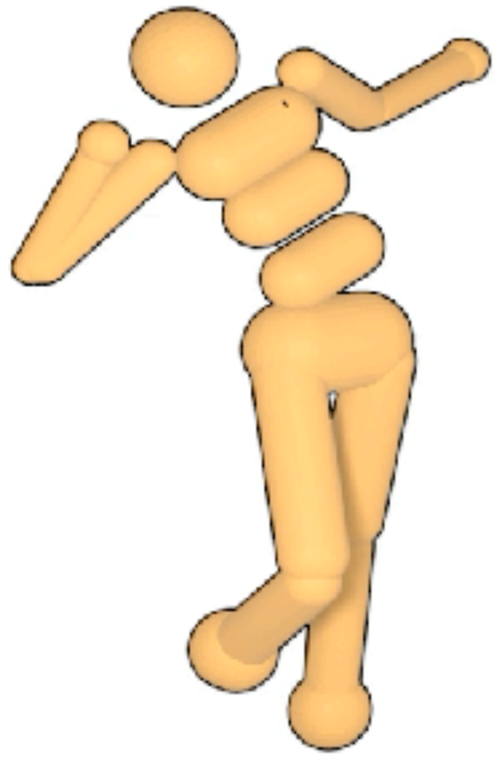
Offline Reinforcement Learning as One Big Sequence Modeling Problem

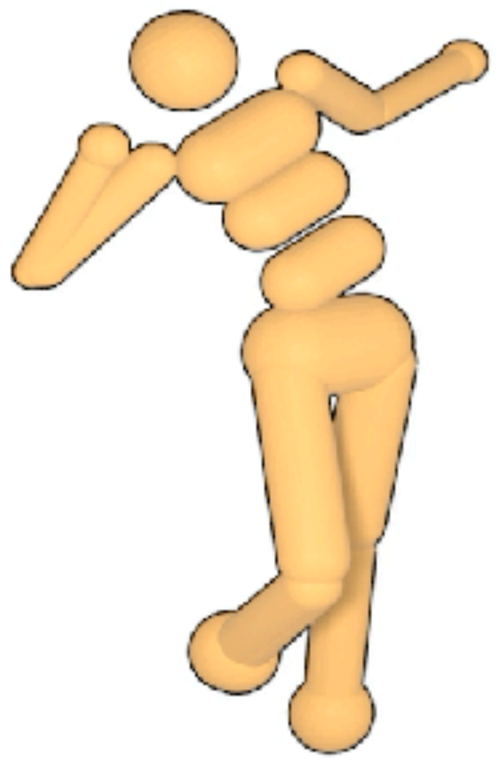
Michael Janner **Qiyang Li** **Sergey Levine**
University of California at Berkeley
{janner, qcli}@berkeley.edu svlevine@eecs.berkeley.edu



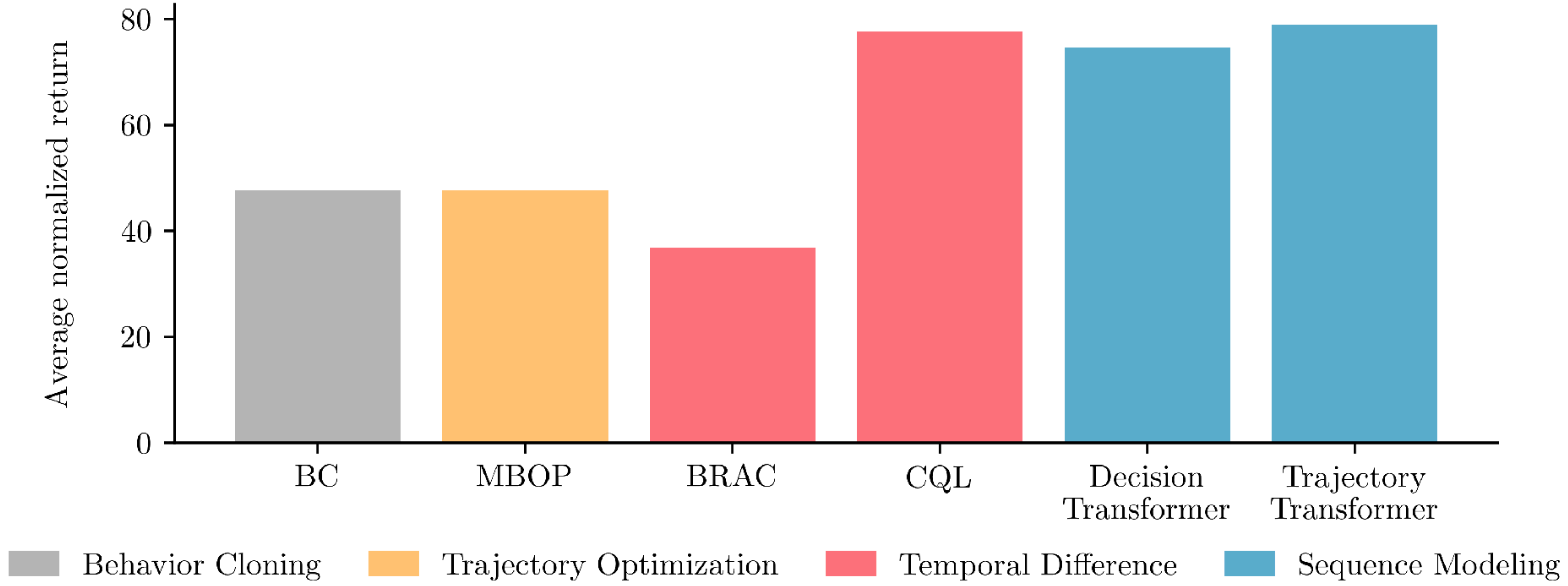
Think of it as model-based RL / IL

Trajectory Transformer





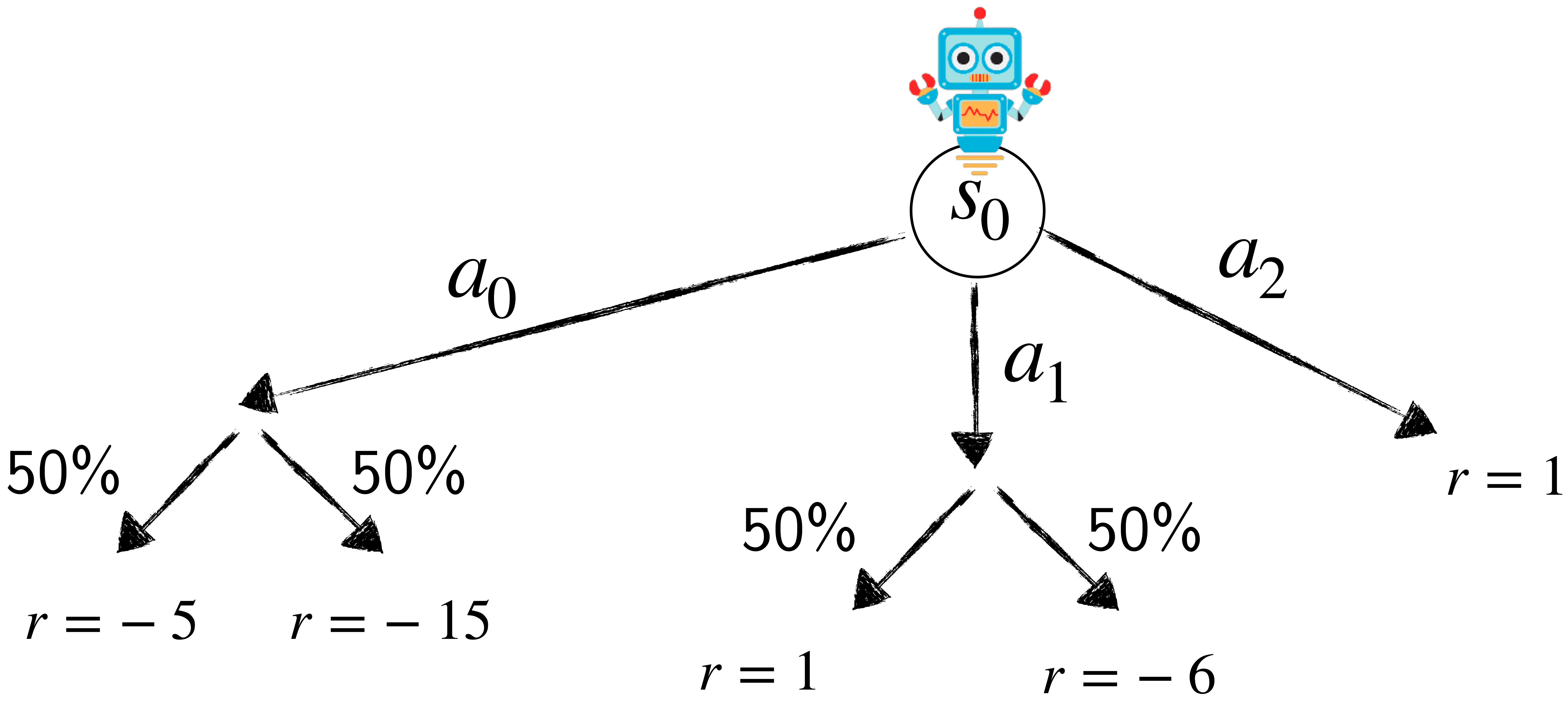
Performs comparably to DT



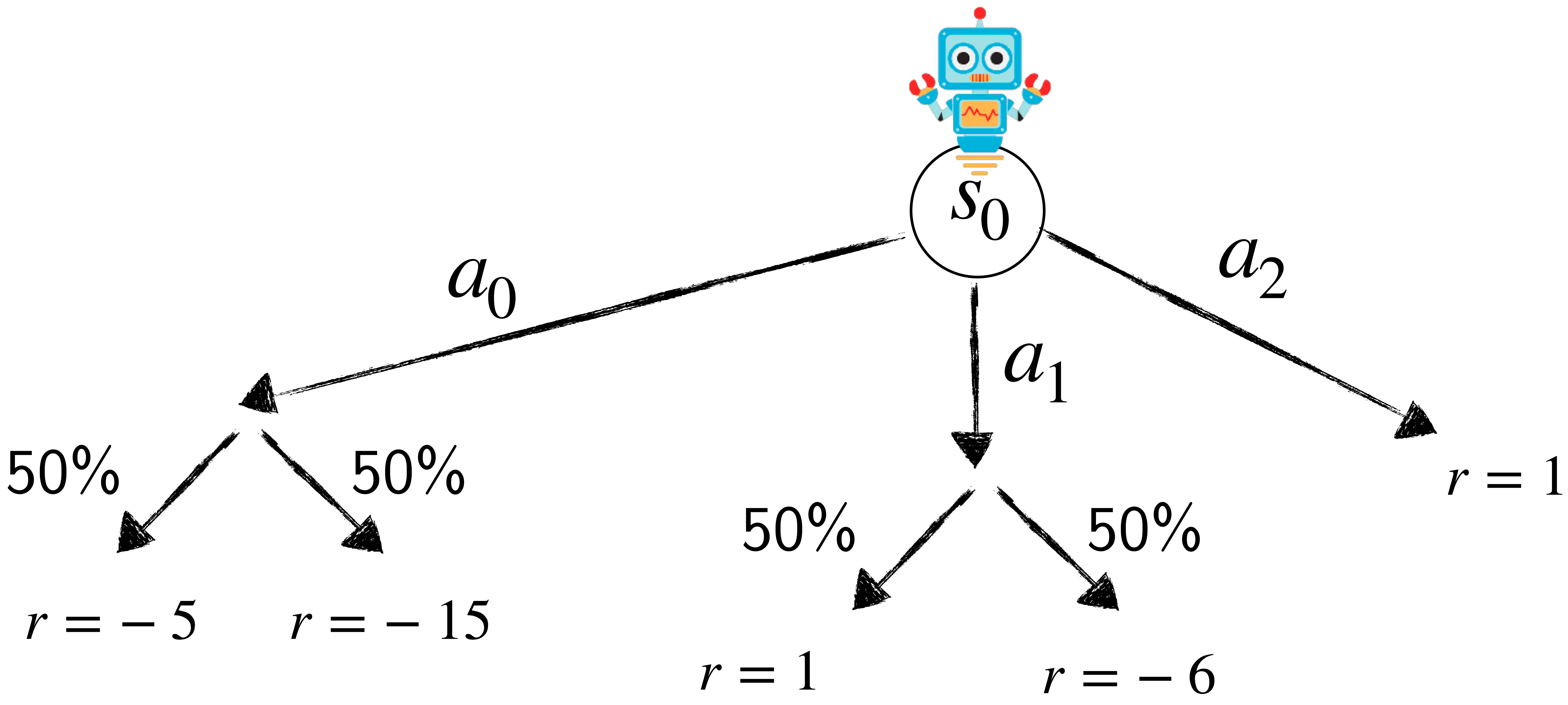
Are we done?

Activity!

Consider the following MDP



Consider the following MDP



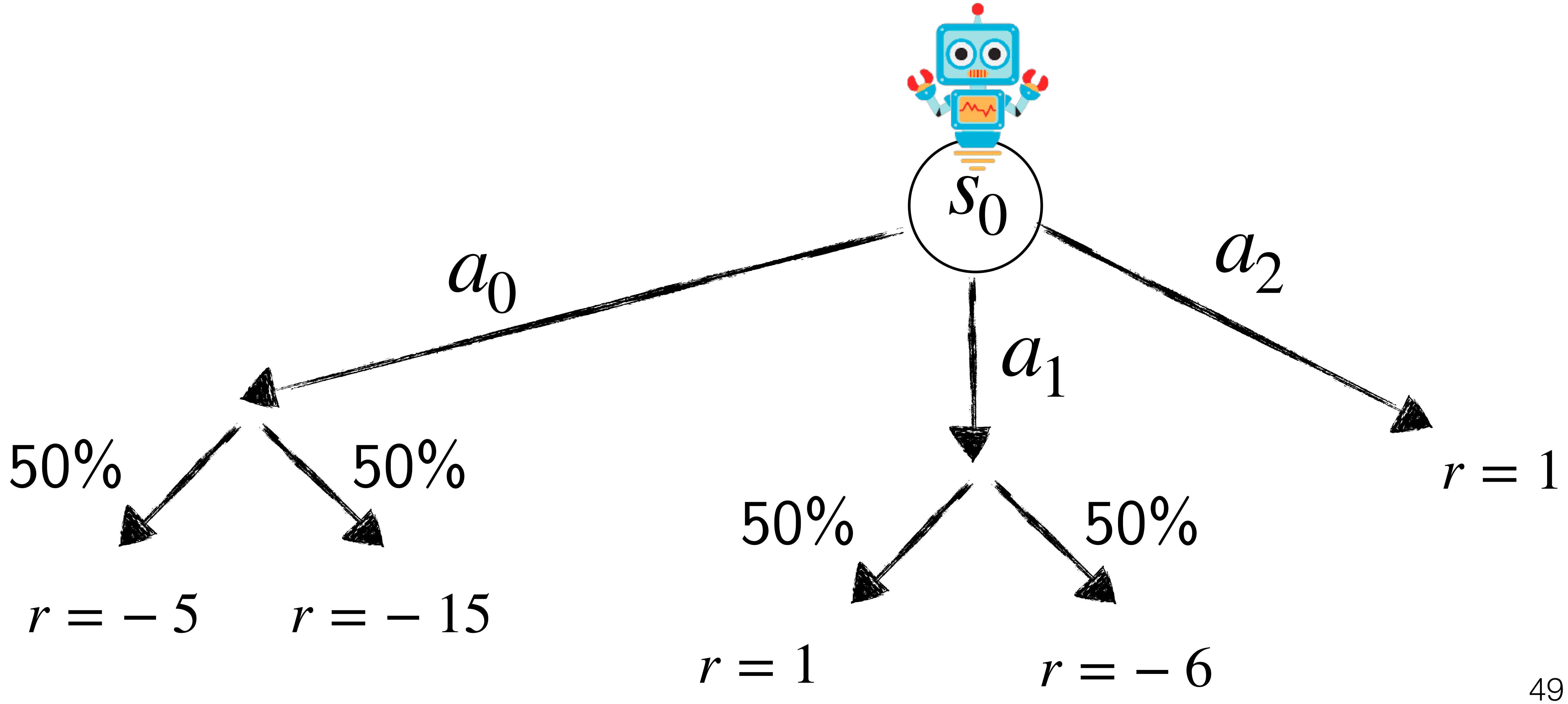
What is the optimal action? What will Decision Transformer play?

Think-Pair-Share!

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

Pair: Find a partner

Share (45 sec): Partners exchange ideas



You Can't Count on Luck:
Why Decision Transformers and RvS
Fail in Stochastic Environments

Keiran Paster
Department of Computer Science
University of Toronto, Vector Institute
keirp@cs.toronto.edu

Sheila A. McIlraith & Jimmy Ba
Department of Computer Science
University of Toronto, Vector Institute
{sheila, jba}@cs.toronto.edu

methods that condition on outcomes such as return can make incorrect decisions in stochastic environments regardless of scale or the amount of data they are trained on

You Can't Count on Luck: Why Decision Transformers and RvS Fail in Stochastic Environments

Keiran Paster
Department of Computer Science
University of Toronto, Vector Institute
keirp@cs.toronto.edu

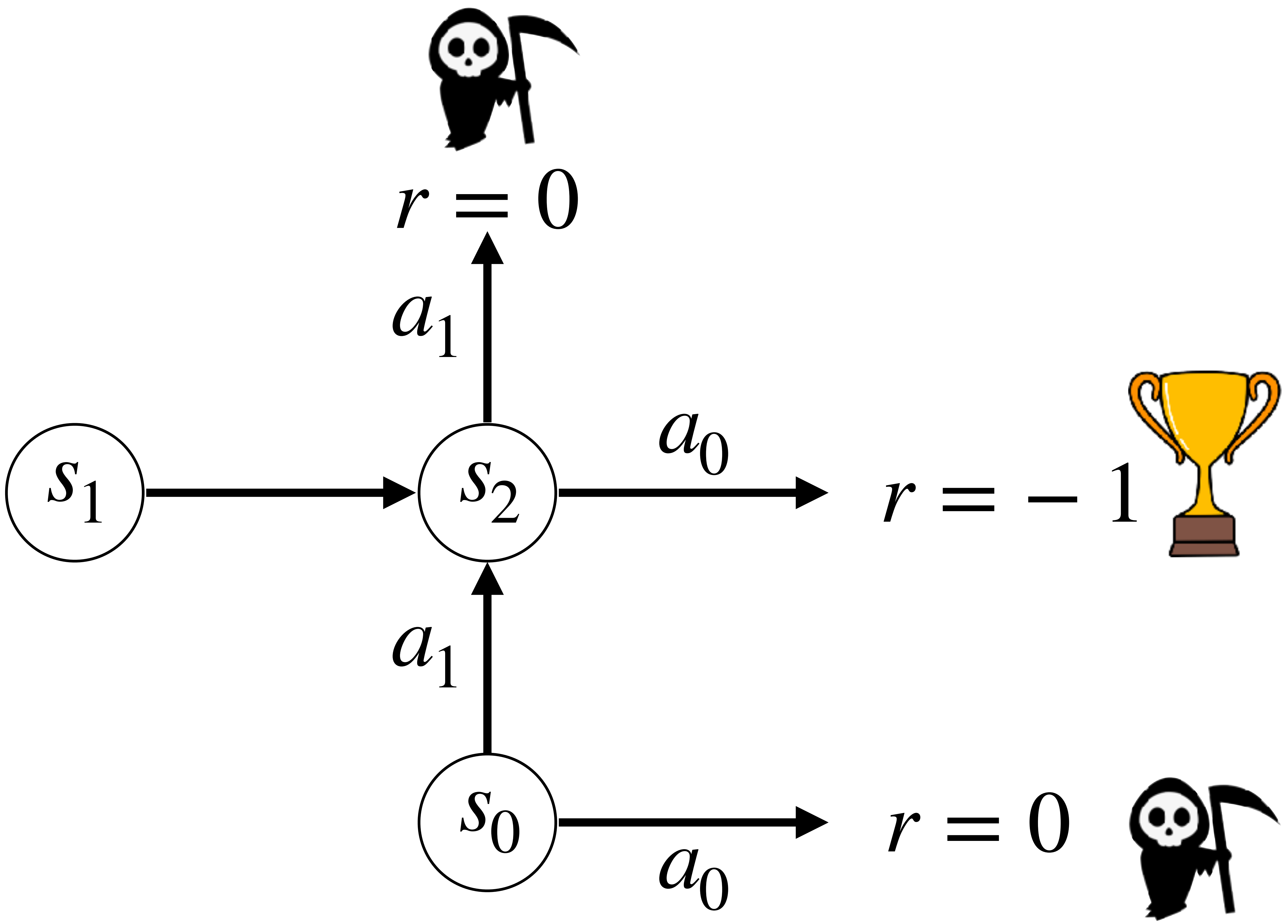
Sheila A. McIlraith & Jimmy Ba
Department of Computer Science
University of Toronto, Vector Institute
{sheila, jba}@cs.toronto.edu

methods that condition on outcomes such as return can make incorrect decisions in stochastic environments regardless of scale or the amount of data they are trained on

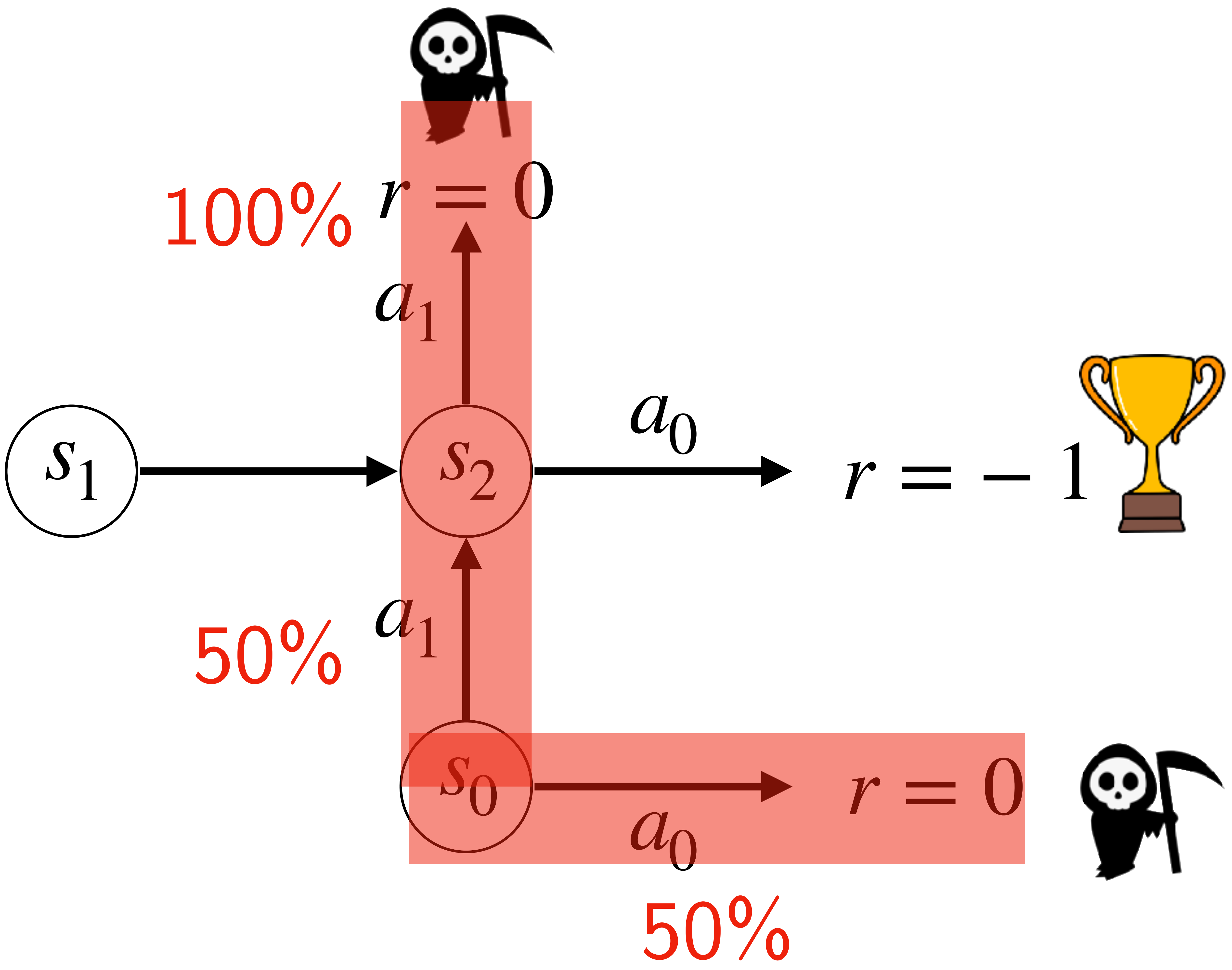
But does it work in deterministic environments?

Activity!

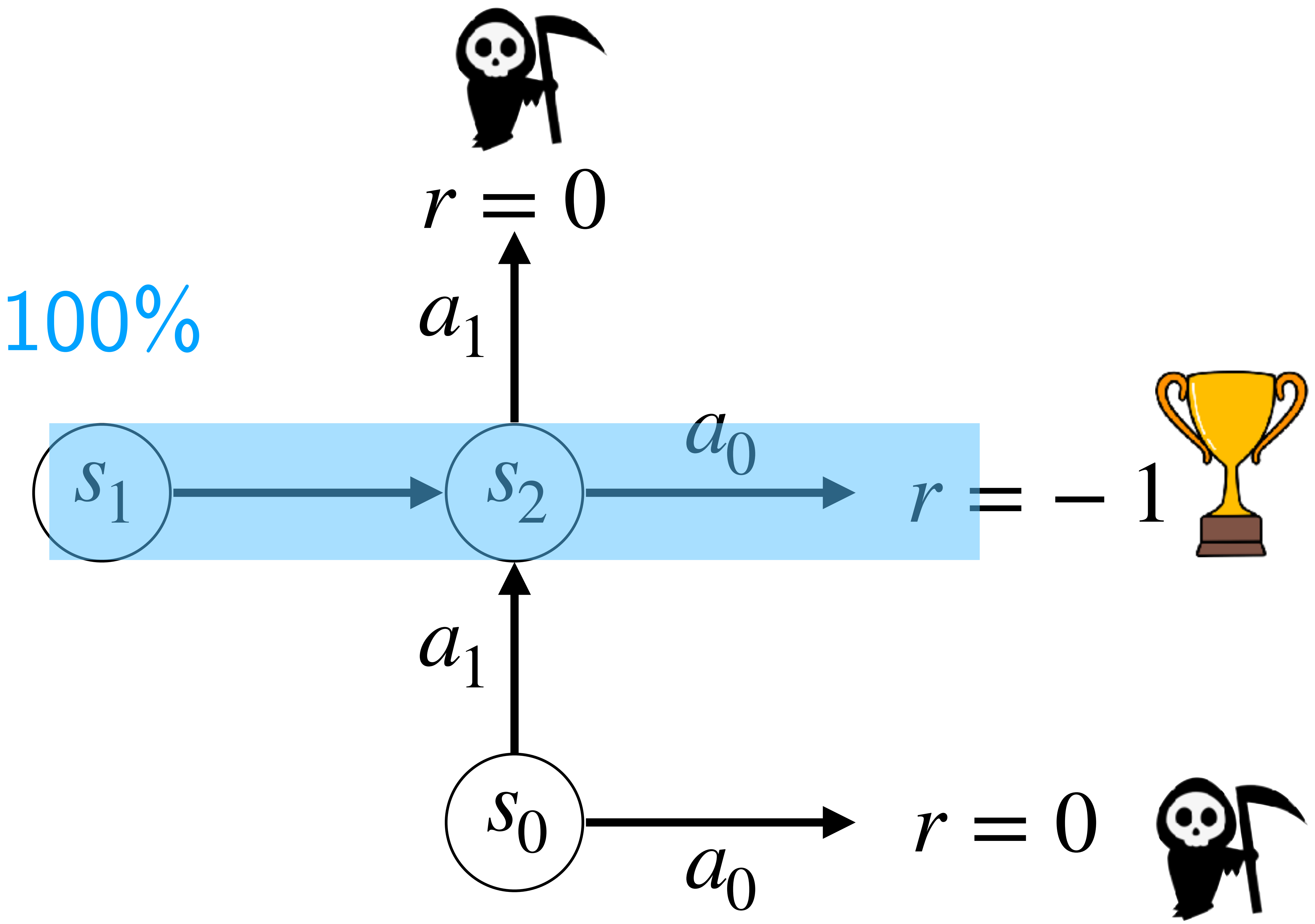
Consider the following deterministic MDP



Data collection 1



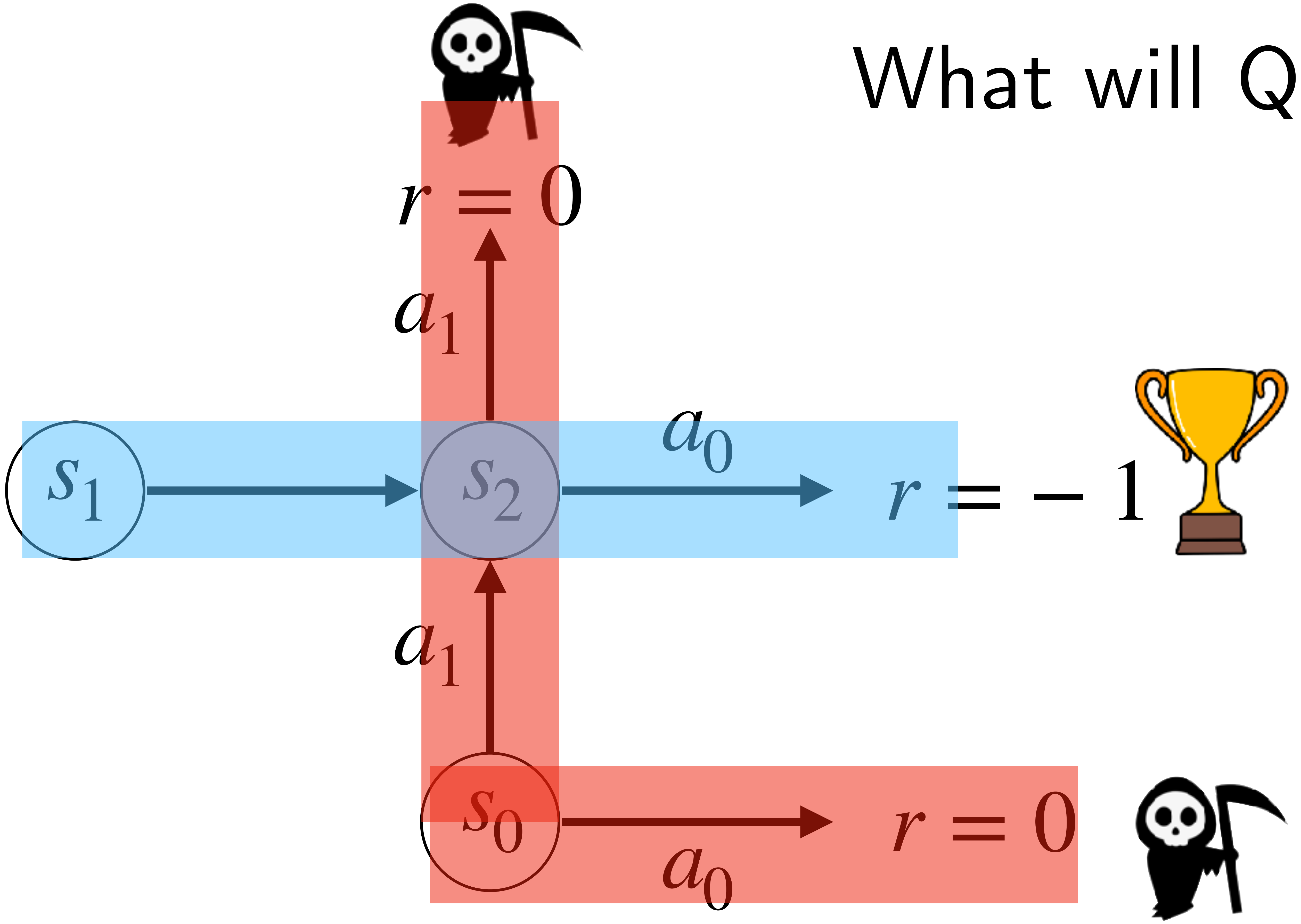
Data collection2



Let's say we start from s0

What will DT learn?

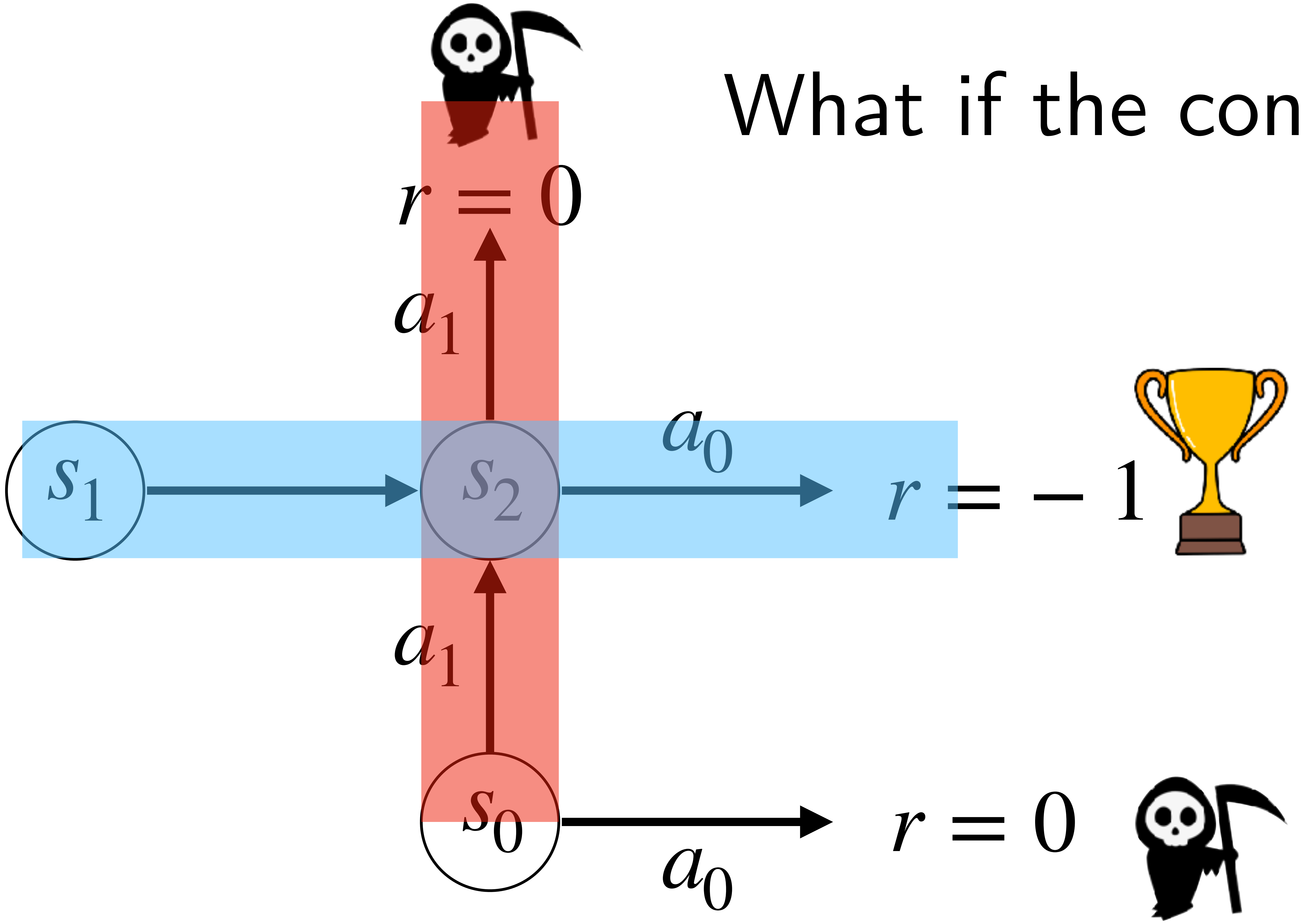
What will Q learning learn?



Let's say we start from s0

What will DT learn?

What if the context length =1?



When does return-conditioned supervised learning work for offline reinforcement learning?

David Brandfonbrener
New York University
david.brandfonbrener@nyu.edu

Alberto Bietti
New York University

Jacob Buckman
MILA

Romain Laroche
Microsoft Research

Joan Bruna
New York University

Sufficient conditions for DT to work

Let's data gathering policy be β, and $R^*(s)$ be the optimal return

Assume

1. Return coverage: $P_\beta(R = R^*(s_0) \,|\, s_0) \geq \alpha$ for all initial states s_0

   You will see all returns some fraction of the time from all initial states

2. Near determinism:

   $P(r \neq r(s,a) \text{ or } s' \neq T(s,a) \,|\, s,a) \leq \epsilon$ for all (s,a)

Then

$$J(\pi^*) - J(\pi_{DT}) \leq \epsilon \left(\frac{1}{\alpha} + 2\right) H^2$$

Research Questions

Can we condition on better alternatives to return?

Train a value estimator (critic)

DICHOTOMY OF CONTROL: SEPARATING WHAT YOU CAN CONTROL FROM WHAT YOU CANNOT

Mengjiao Yang
University of California, Berkeley
Google Research, Brain Team
sherryy@google.com

Dale Schuurmans
University of Alberta
Google Research, Brain Team

Pieter Abbeel
University of California, Berkeley

Ofir Nachum
Google Research, Brain Team

Addressing Optimism Bias in Sequence Modeling for Reinforcement Learning

Adam Villaflor [1] **Zhe Huang** [1] **Swapnil Pande** [1] **John Dolan** [1] **Jeff Schneider** [1]