Reinforcement Learning

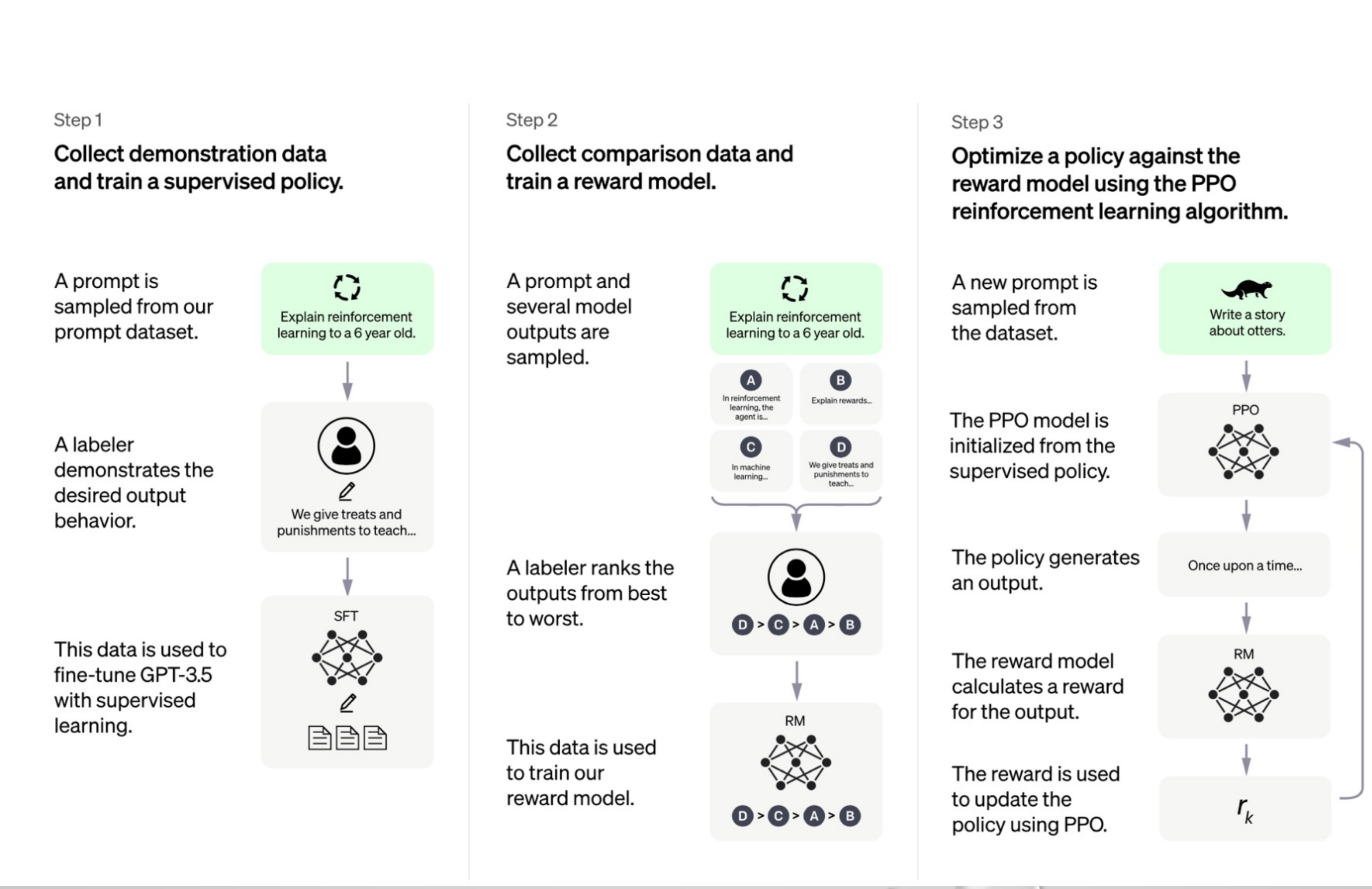
Computer Engineering Department
Sharif University of Technology

Mohammad Hossein Rohban, Ph.D.

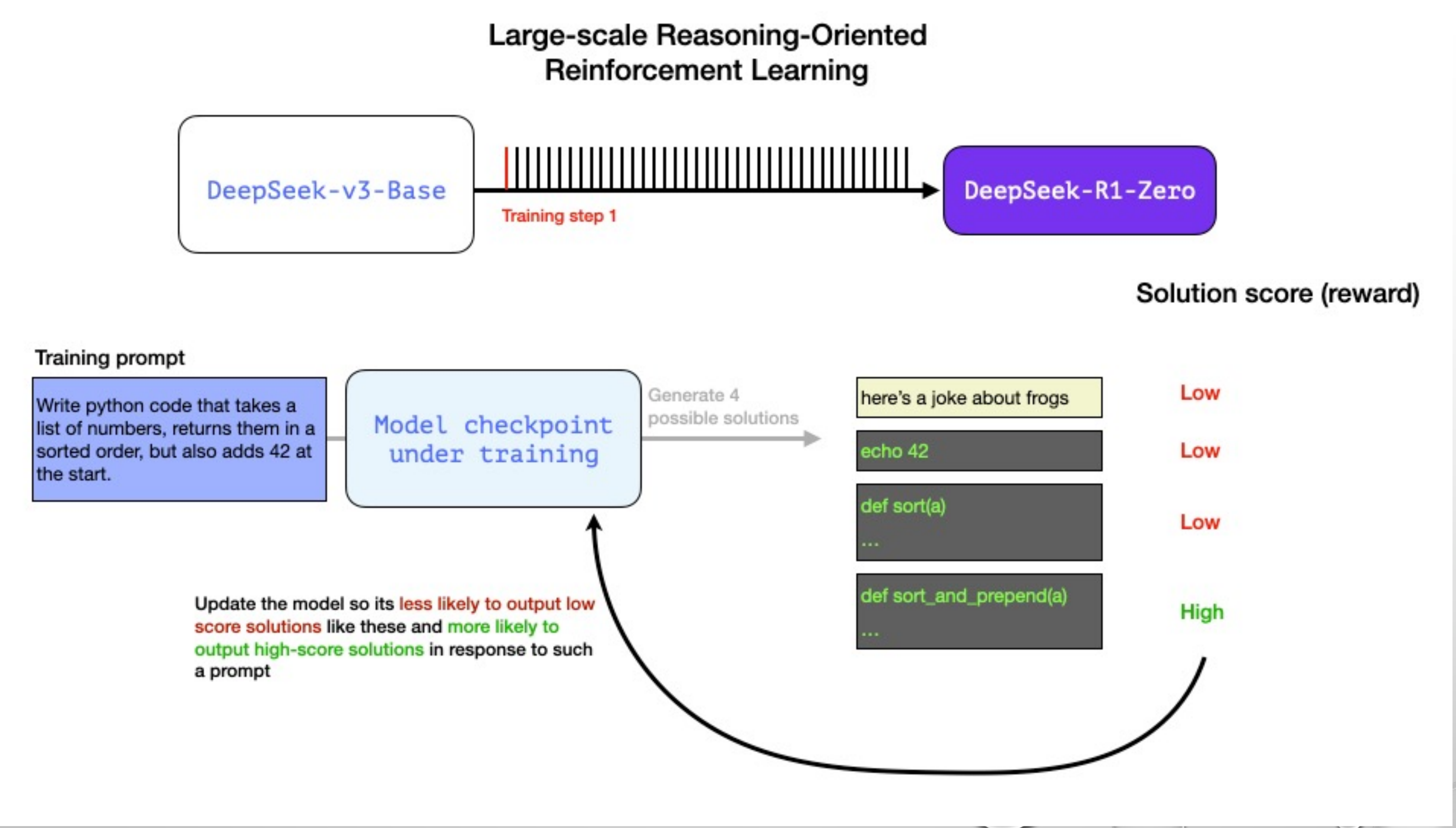
Spring 2025

Courtesy: Some slides are adopted from CS 285 Berkeley, and CS 234 Stanford, and Pieter Abbeel's compact series on RL.

Motivation (cont.) ChatGPT; Why RL?!



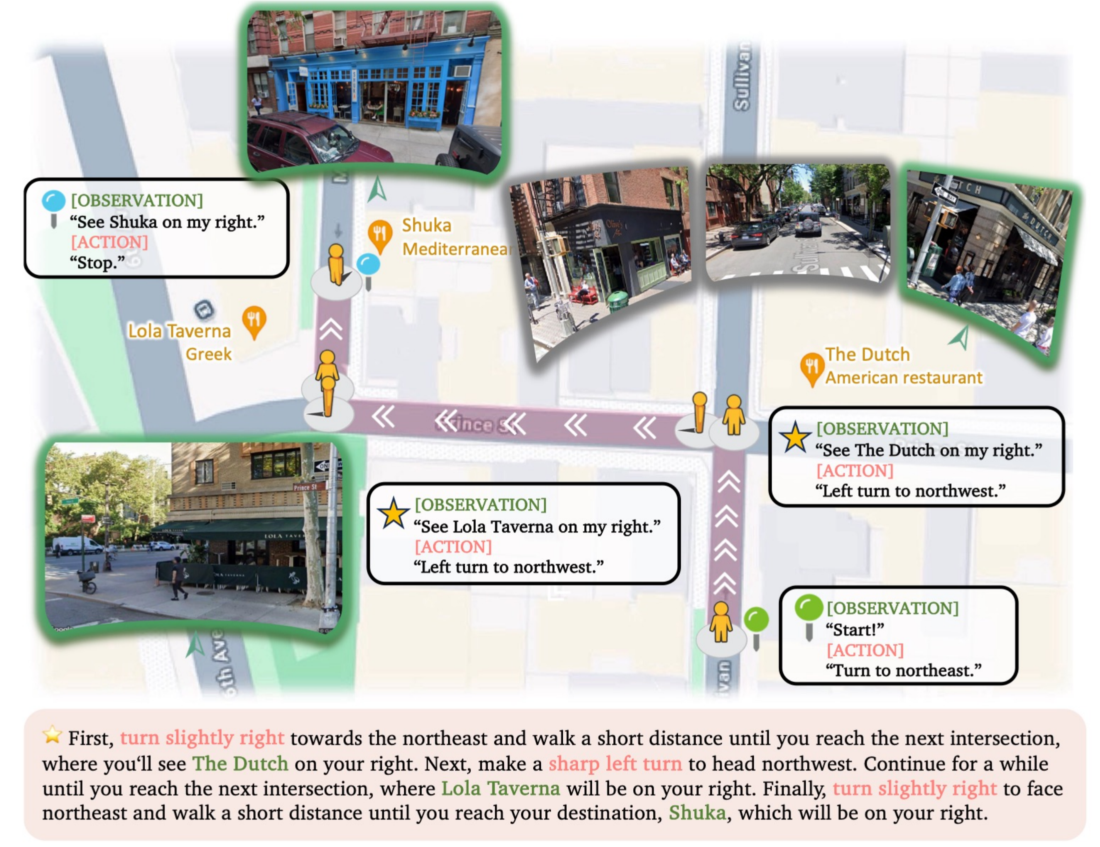
Courtesy: OpenAI Blog

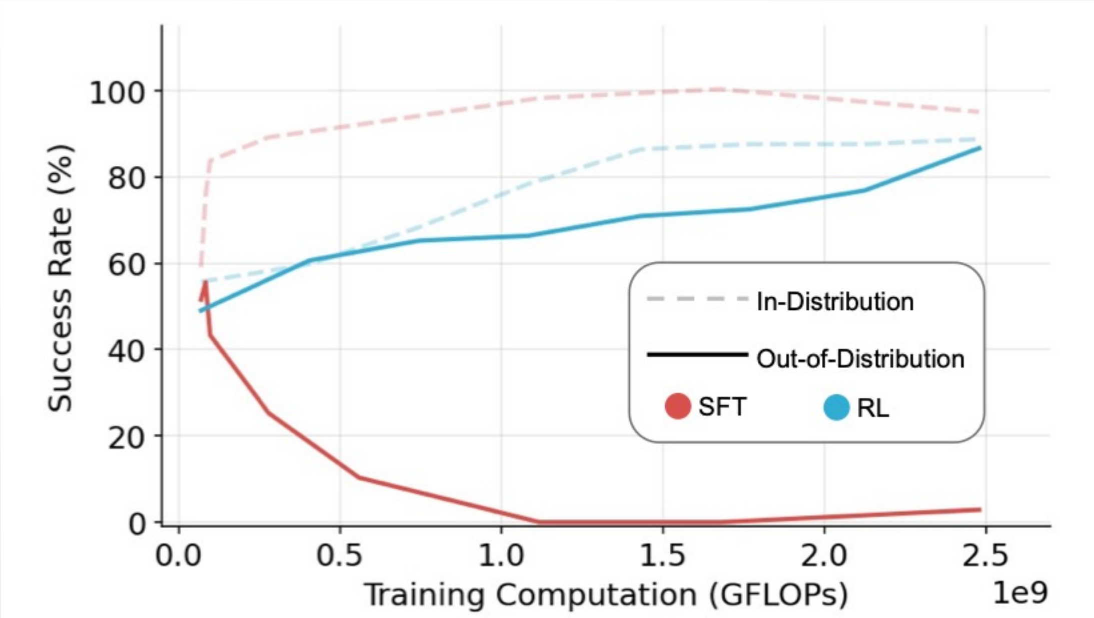


Courtesy: J. Alammar, The Illustrated DeepSeek R1

Motivation (cont.)

SFT Memorizes, RL Generalizes:
A Comparative Study of Foundation Model Post-training



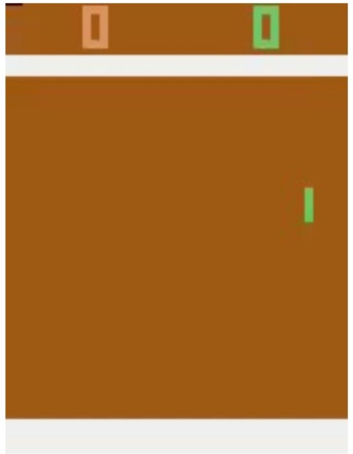


Courtesy: T. Chu, et al, SFT Memorizes, RL Generalizes: A Comparative Study of Foundation Model Post-training, 2025

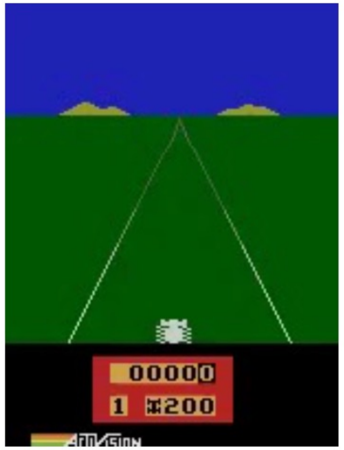
History

2013

Atari (DQN)
[Deepmind]

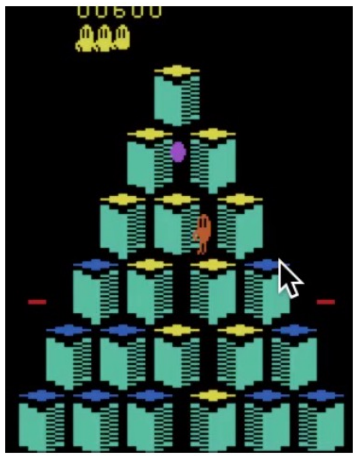


Pong



Enduro

Beamrider

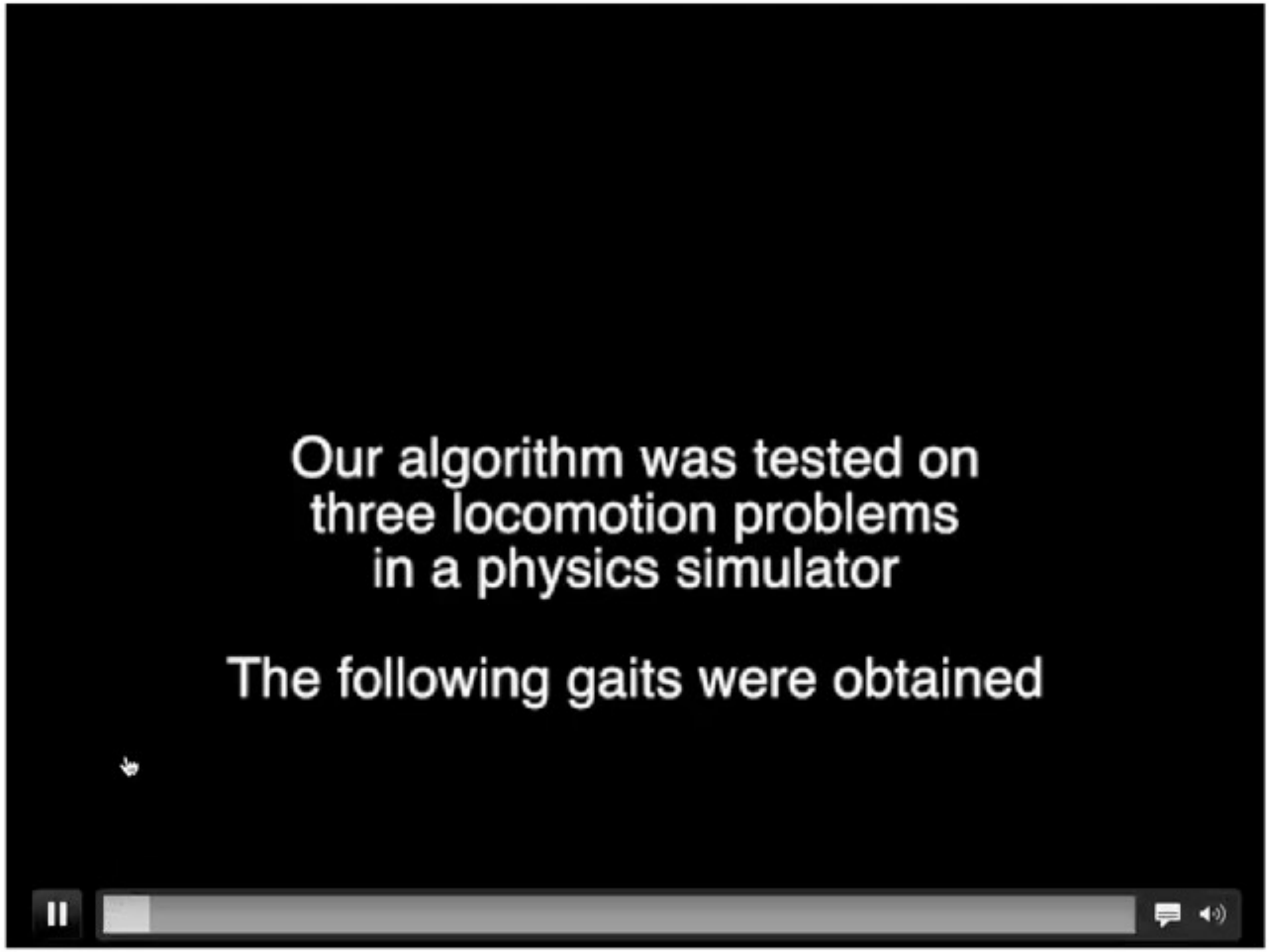


Q*bert

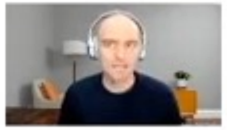
A Few Deep RL Highlights

2013 Atari (DQN)
[Deepmind]

2014 **2D locomotion (TRPO)**
[Berkeley]



Play 0:06 – 0:25



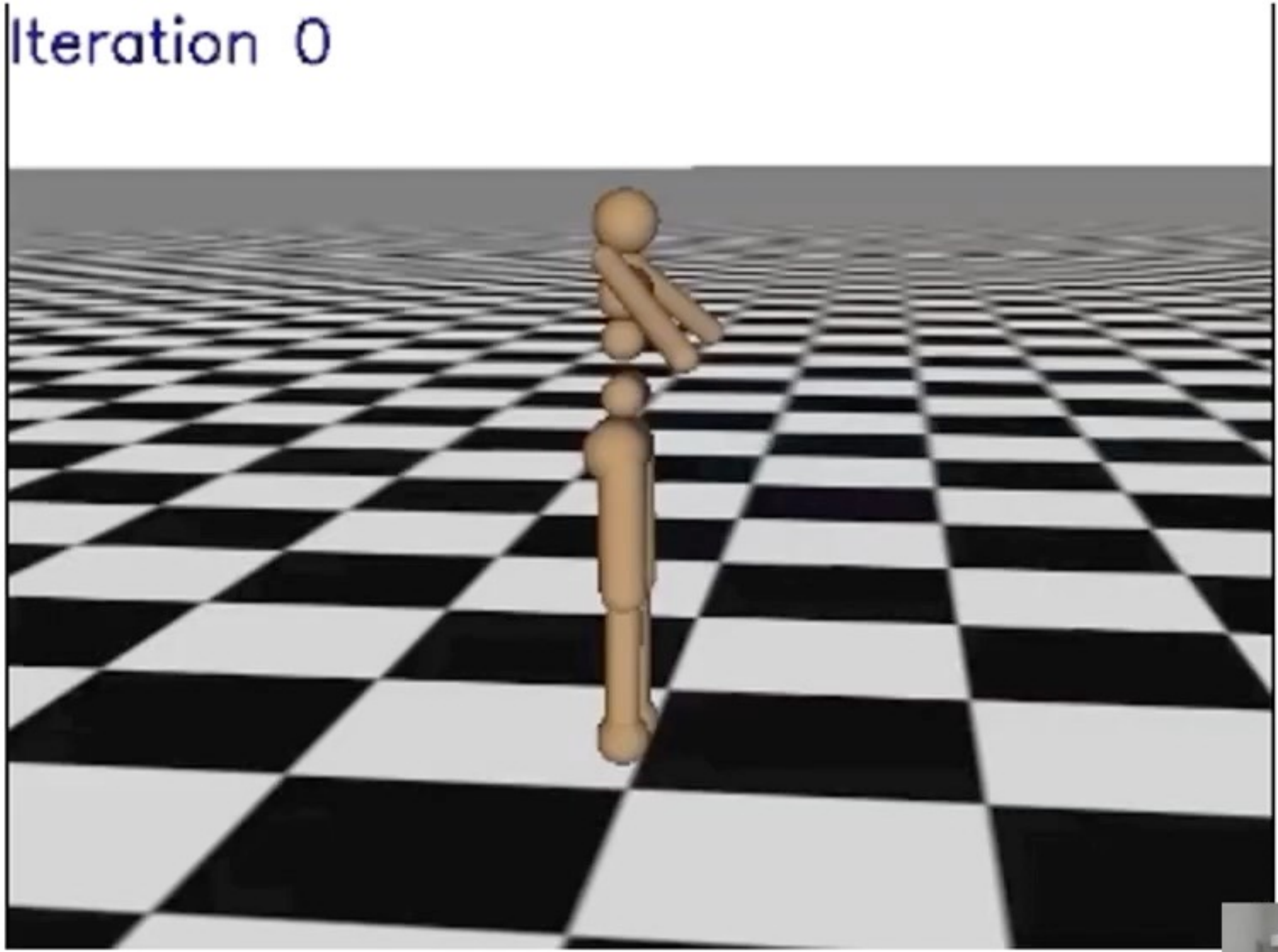
History

2013	Atari (DQN) [Deepmind]
2014	2D locomotion (TRPO) [Berkeley]
2015	**AlphaGo [Deepmind]**

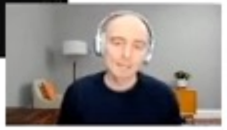
Tian et al, 2016; Maddison et al, 2014; Clark et al, 2015

A Few Deep RL Highlights

2013	Atari (DQN) [Deepmind]
2014	2D locomotion (TRPO) [Berkeley]
2015	AlphaGo [Deepmind]
2016	**3D locomotion (TRPO+GAE)** **[Berkeley]**



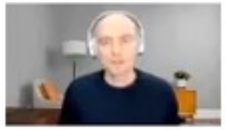
[Schulman, Moritz, Levine, Jordan, Abbeel, ICLR 2016]



A Few Deep RL Highlights

2013	Atari (DQN) [Deepmind]
2014	2D locomotion (TRPO) [Berkeley]
2015	AlphaGo [Deepmind]
2016	3D locomotion (TRPO+GAE) [Berkeley]
2016	**Real Robot Manipulation (GPS) [Berkeley]**

[Levine*, Finn*, Darrell, Abbeel, JMLR 2016]



History

2013 Atari (DQN) [Deepmind]

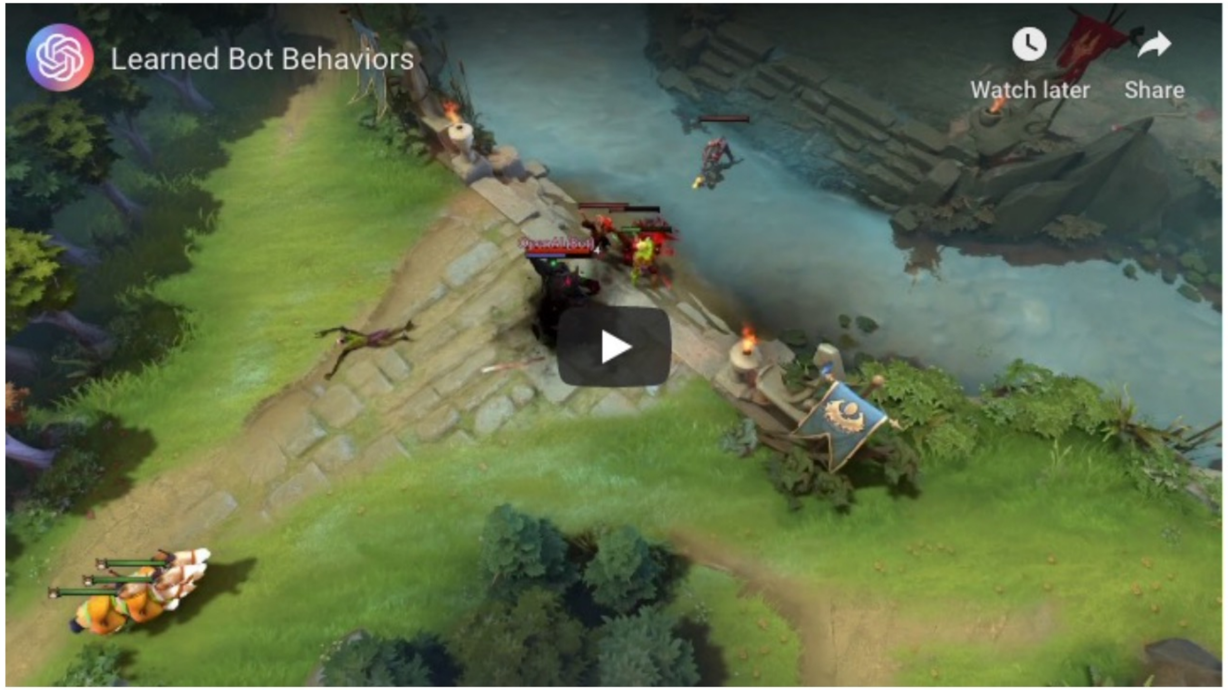
2014 2D locomotion (TRPO) [Berkeley]

2015 AlphaGo [Deepmind]

2016 3D locomotion (TRPO+GAE) [Berkeley]

2016 Real Robot Manipulation (GPS) [Berkeley, Google]

2017 Dota2 (PPO) [OpenAI]



OpenAI Dota Bot beat best humans 1:1 (Aug 2018)

A Few Deep RL Highlights

2013 Atari (DQN) [Deepmind]

2014 2D locomotion (TRPO) [Berkeley]

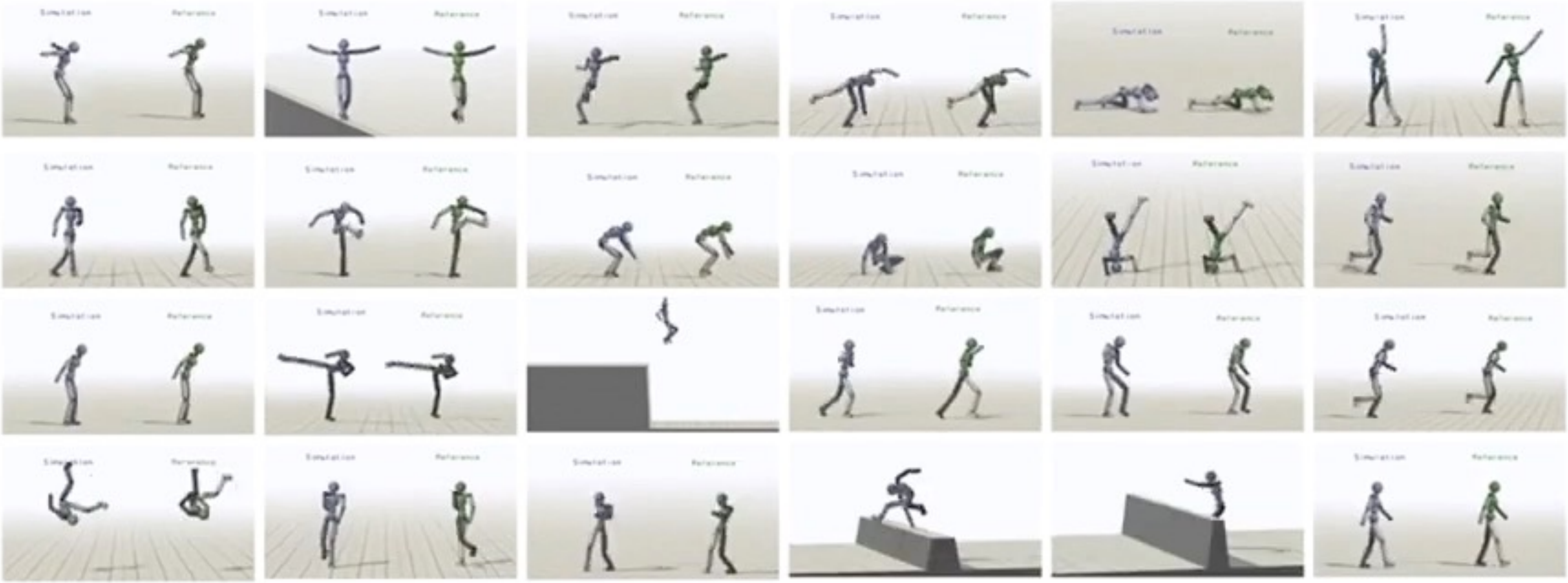
2015 AlphaGo [Deepmind]

2016 3D locomotion (TRPO+GAE) [Berkeley]

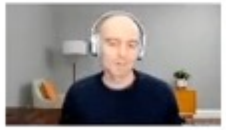
2016 Real Robot Manipulation (GPS) [Berkeley, Google]

2017 Dota2 (PPO) [OpenAI]

2018 DeepMimic [Berkeley]



[Peng, Abbeel, Levine, van de Panne, 2018]



History

2013 Atari (DQN) [Deepmind]

2014 2D locomotion (TRPO) [Berkeley]

2015 AlphaGo [Deepmind]

2016 3D locomotion (TRPO+GAE) [Berkeley]

2016 Real Robot Manipulation (GPS) [Berkeley, Google]

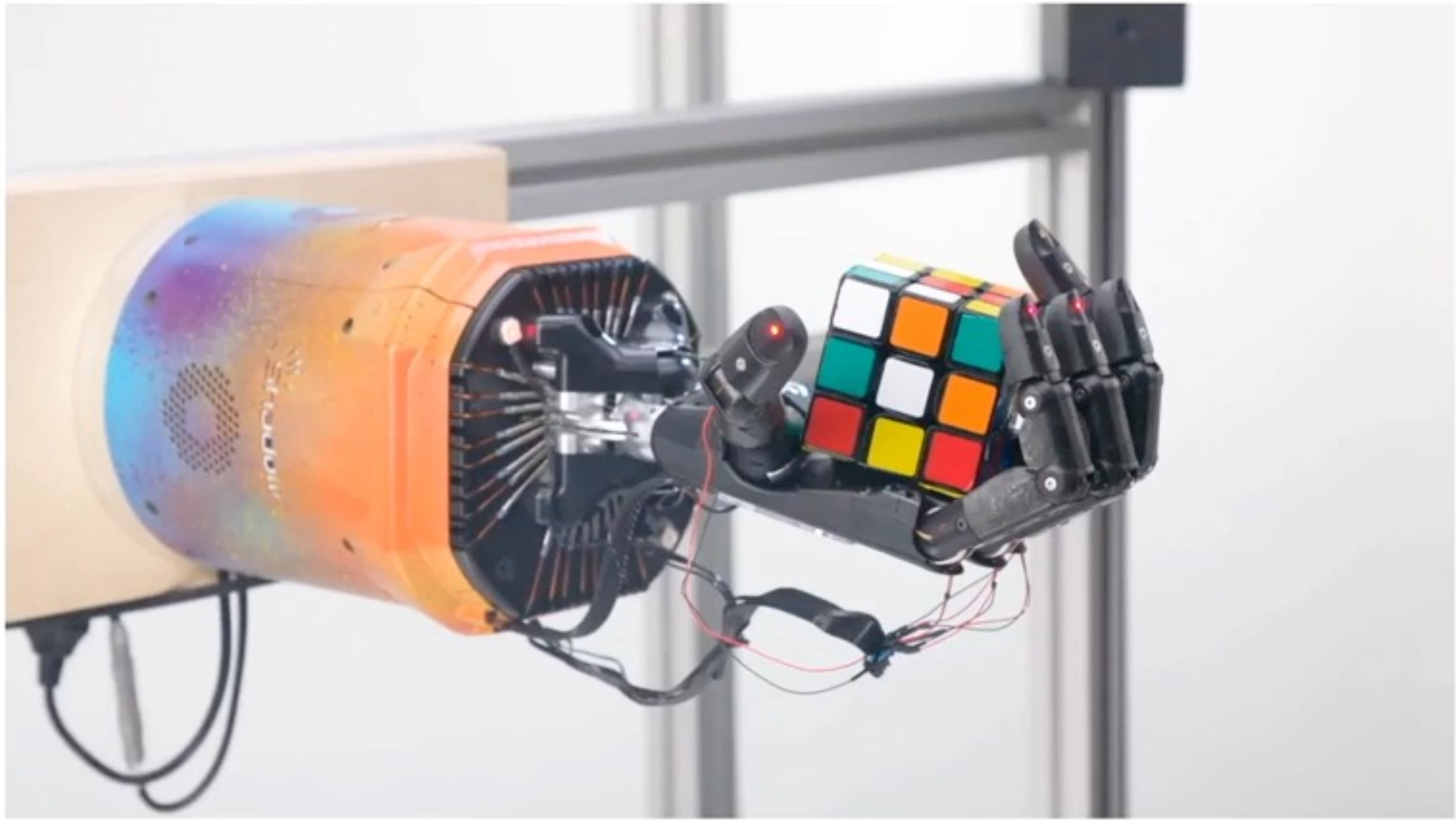
2017 Dota2 (PPO) [OpenAI]

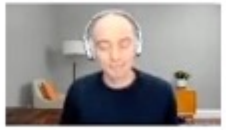
2018 DeepMimic [Berkeley]

2019 AlphaStar [Deepmind]

A Few Deep RL Highlights

2013	Atari (DQN) [Deepmind]
2014	2D locomotion (TRPO) [Berkeley]
2015	AlphaGo [Deepmind]
2016	3D locomotion (TRPO+GAE) [Berkeley]
2016	Real Robot Manipulation (GPS) [Berkeley, Google]
2017	Dota2 (PPO) [OpenAI]
2018	DeepMimic [Berkeley]
2019	AlphaStar [Deepmind]
2019	**Rubik's Cube (PPO+DR) [OpenAI]**

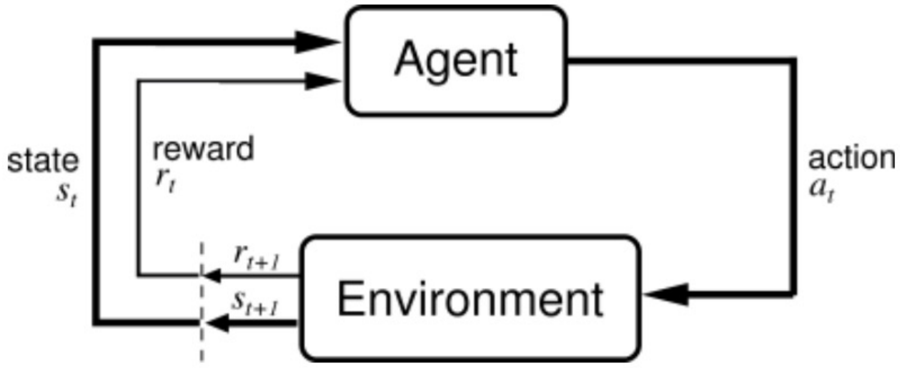




Let's Begin: Markov Decision Processes (MDPs)

An MDP is defined by:

- Set of states S
- Set of actions A
- Transition function $P(s' \mid s, a)$
- Reward function $R(s, a, s')$
- Start state s_0
- Discount factor γ
- Horizon H



The Goal

- The policy is $\pi_\theta : S \rightarrow A$ for infinite horizon or

  $\pi_\theta : S \times \{0, \dots, H\} \rightarrow A$ for finite horizon MDP.

MDP (S, A, T, R, γ, H), goal: $max_\pi \mathrm{E}[\sum_{t=0}^{H} \gamma^t R(S_t, A_t, S_{t+1}) | \pi]$

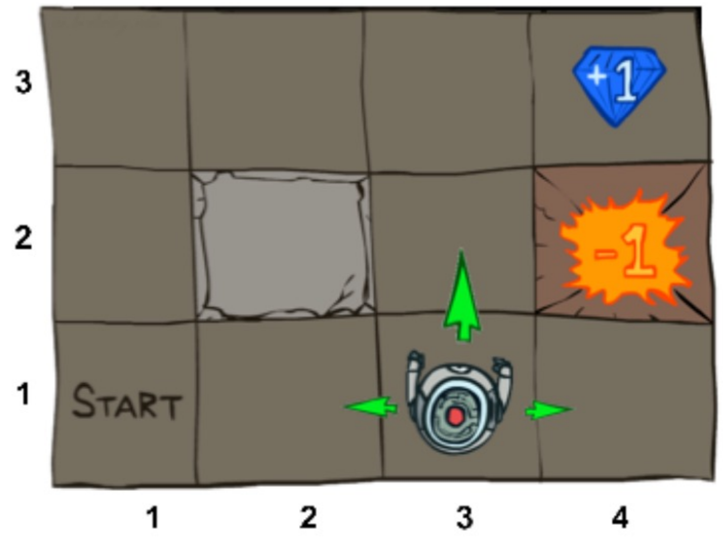
Sometimes the policy could be stochastic: $\pi : S \times A \rightarrow [0,1]$, which is

$\pi(a|s) = \Pr(A_t = a | S_t = s)$.

Example: Grid World

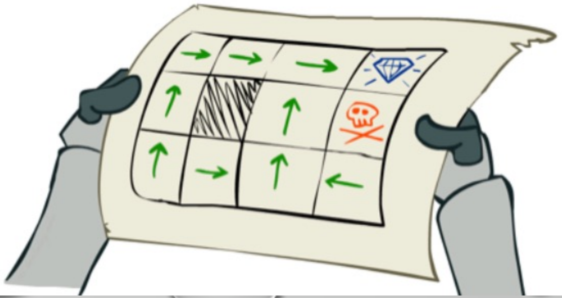
An MDP is defined by:

- Set of states S
- Set of actions A
- Transition function $P(s' \mid s, a)$
- Reward function $R(s, a, s')$
- Start state s_0
- Discount factor γ
- Horizon H

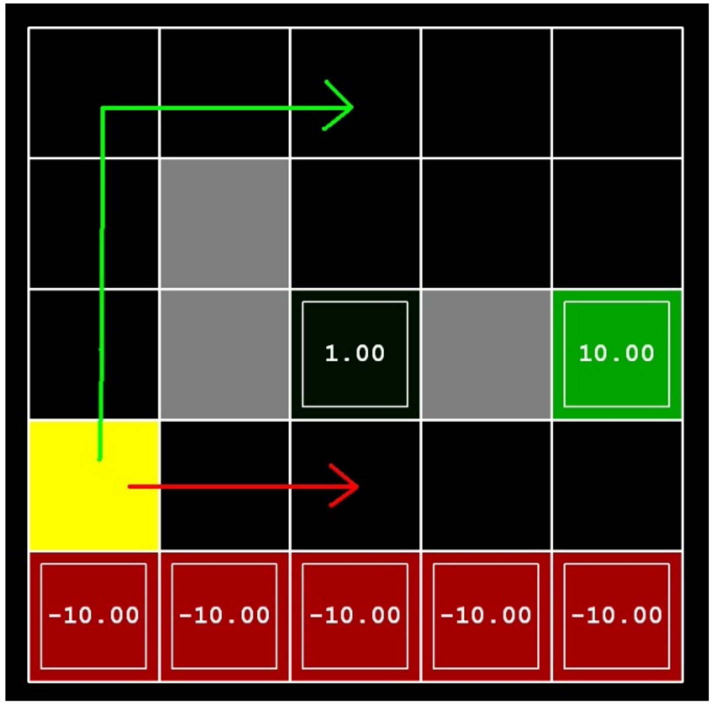


Goal: $$max_\pi \mathrm{E}[\sum_{t=0}^{H} \gamma^t R(S_t, A_t, S_{t+1})|\pi]$$

π:



Exercise



(a) Prefer the close exit (+1), risking the cliff (-10)

(b) Prefer the close exit (+1), but avoiding the cliff (-10)

(c) Prefer the distant exit (+10), risking the cliff (-10)

(d) Prefer the distant exit (+10), avoiding the cliff (-10)

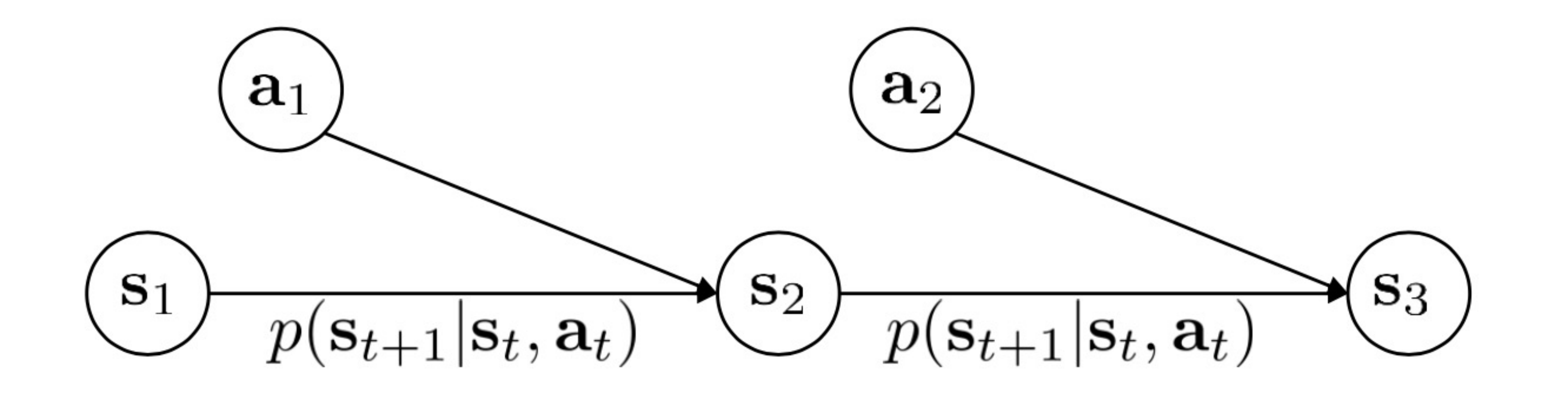
(1) $\gamma = 0.1$, noise = 0.5

(2) $\gamma = 0.99$, noise = 0

(3) $\gamma = 0.99$, noise = 0.5

(4) $\gamma = 0.1$, noise = 0

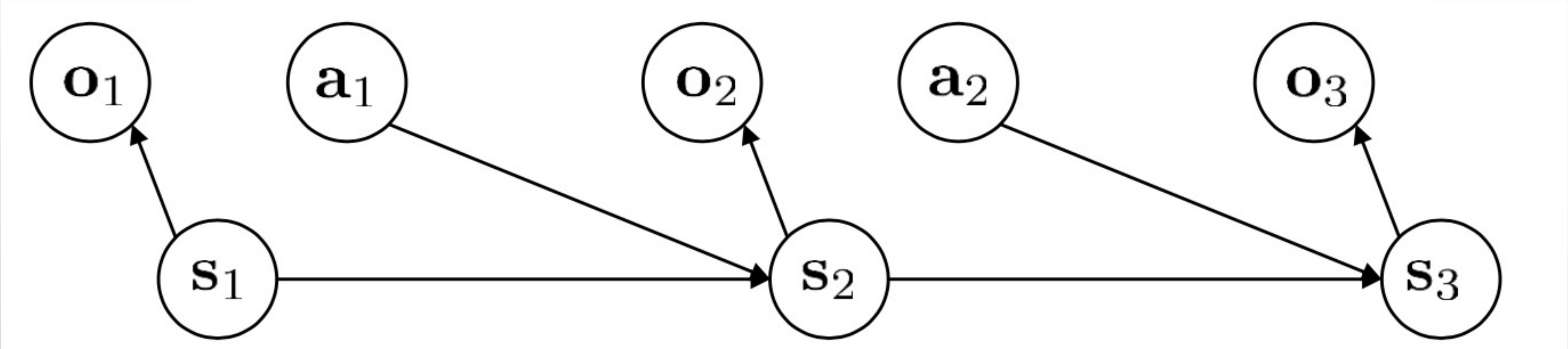
Graphical Model of MDPs



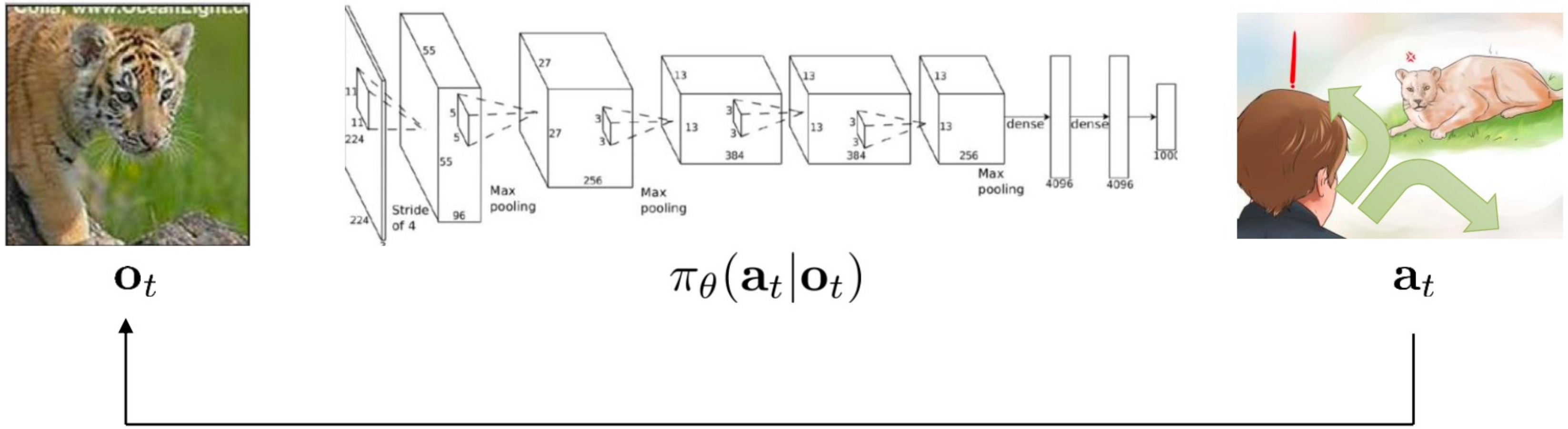
Partially Observable MDPs (POMDPs)

- Often times the state S_t is hidden from the agent,

  and only noisy or incomplete measurement of it is available O_t.

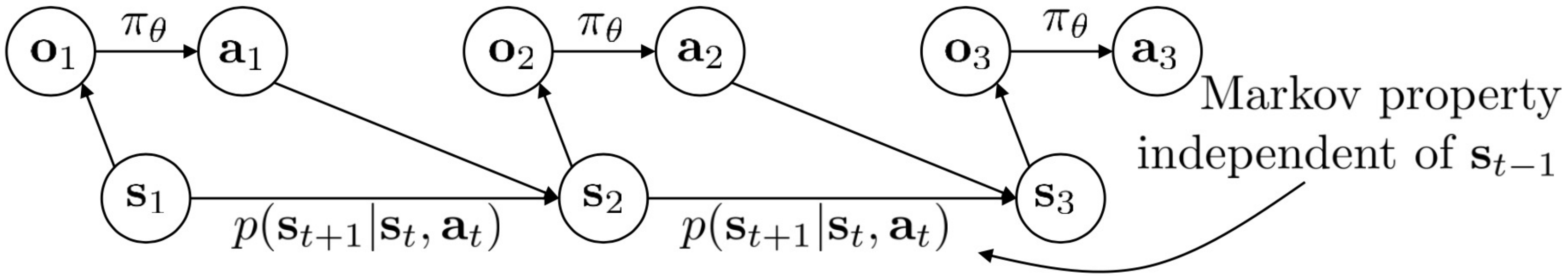


Policy as a function of S_t or O_t



$\mathbf{s}_t$ – state
$\mathbf{o}_t$ – observation
$\mathbf{a}_t$ – action

$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ – policy
$\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ – policy (fully observed)



Optimal Value Function

MDP (S, A, T, R, γ, H), goal: $max_{\pi} \mathrm{E}[\sum_{t=0}^{H} \gamma^t R(S_t, A_t, S_{t+1}) | \pi]$

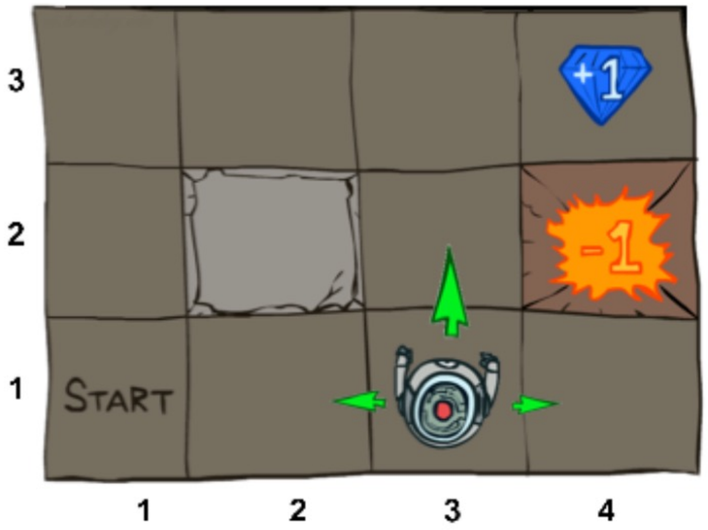
$$V^*(s) = \max_{\pi} \mathbb{E}\left[\sum_{t=0}^{H} \gamma^t R(s_t, a_t, s_{t+1}) \mid \pi, s_0 = s\right]$$

= sum of discounted rewards when starting from state s and acting optimally

Optimal Value Function

$$V^*(s) = \max_\pi \mathbb{E}\left[\sum_{t=0}^{H} \gamma^t R(s_t, a_t, s_{t+1}) \mid \pi, s_0 = s\right]$$

= sum of discounted rewards when starting from state s and acting optimally



Let's assume:
actions deterministically successful, gamma = 1, H = 100

V*(4,3) =

V*(3,3) =

V*(2,3) =

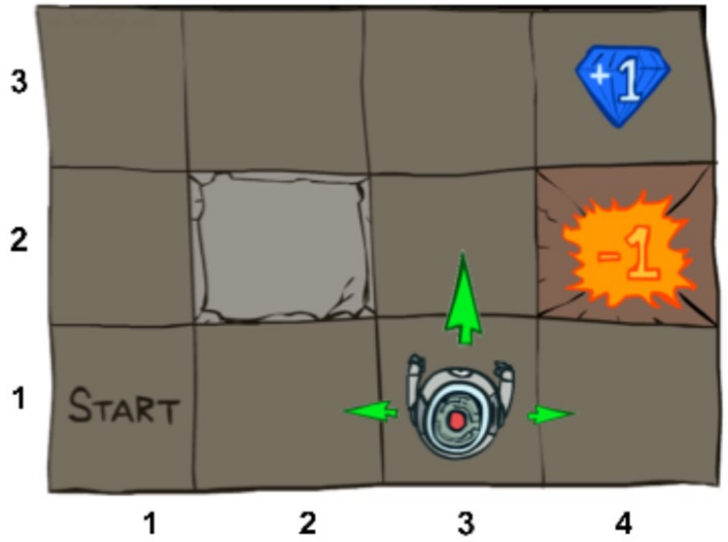
V*(1,1) =

V*(4,2) =

Optimal Value Function

$$V^*(s) = \max_{\pi} \mathbb{E}\left[\sum_{t=0}^{H} \gamma^t R(s_t, a_t, s_{t+1}) \mid \pi, s_0 = s\right]$$

= sum of discounted rewards when starting from state s and acting optimally



<u>Let's assume</u>:
actions deterministically successful, gamma = 0.9, H = 100

V*(4,3) =

V*(3,3) =

V*(2,3) =

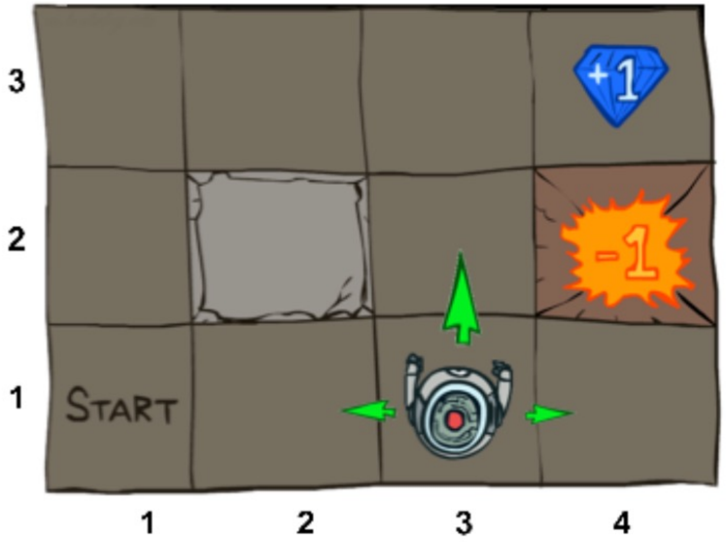
V*(1,1) =

V*(4,2) =

Optimal Value Function

$$V^*(s) = \max_{\pi} \mathbb{E}\left[\sum_{t=0}^{H} \gamma^t R(s_t, a_t, s_{t+1}) \mid \pi, s_0 = s\right]$$

= sum of discounted rewards when starting from state s and acting optimally



<u>Let's assume</u>:
actions successful w/probability 0.8, gamma = 0.9, H = 100

V*(4,3) =

V*(3,3) =

V*(2,3) =

V*(1,1) =

V*(4,2) =