Reinforcement Learning: From Games to Robotics

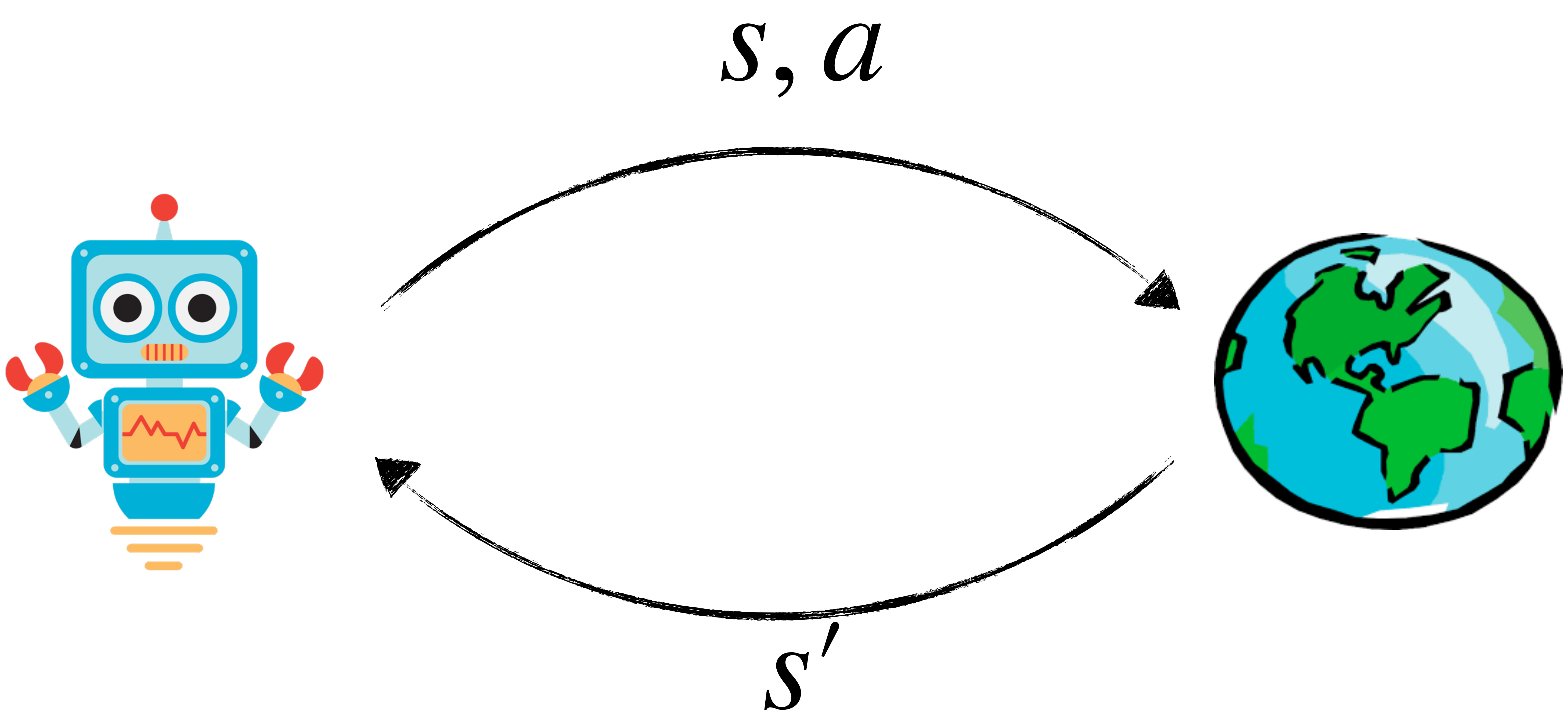
Sanjiban Choudhury

Cornell Bowers CIS
Computer Science

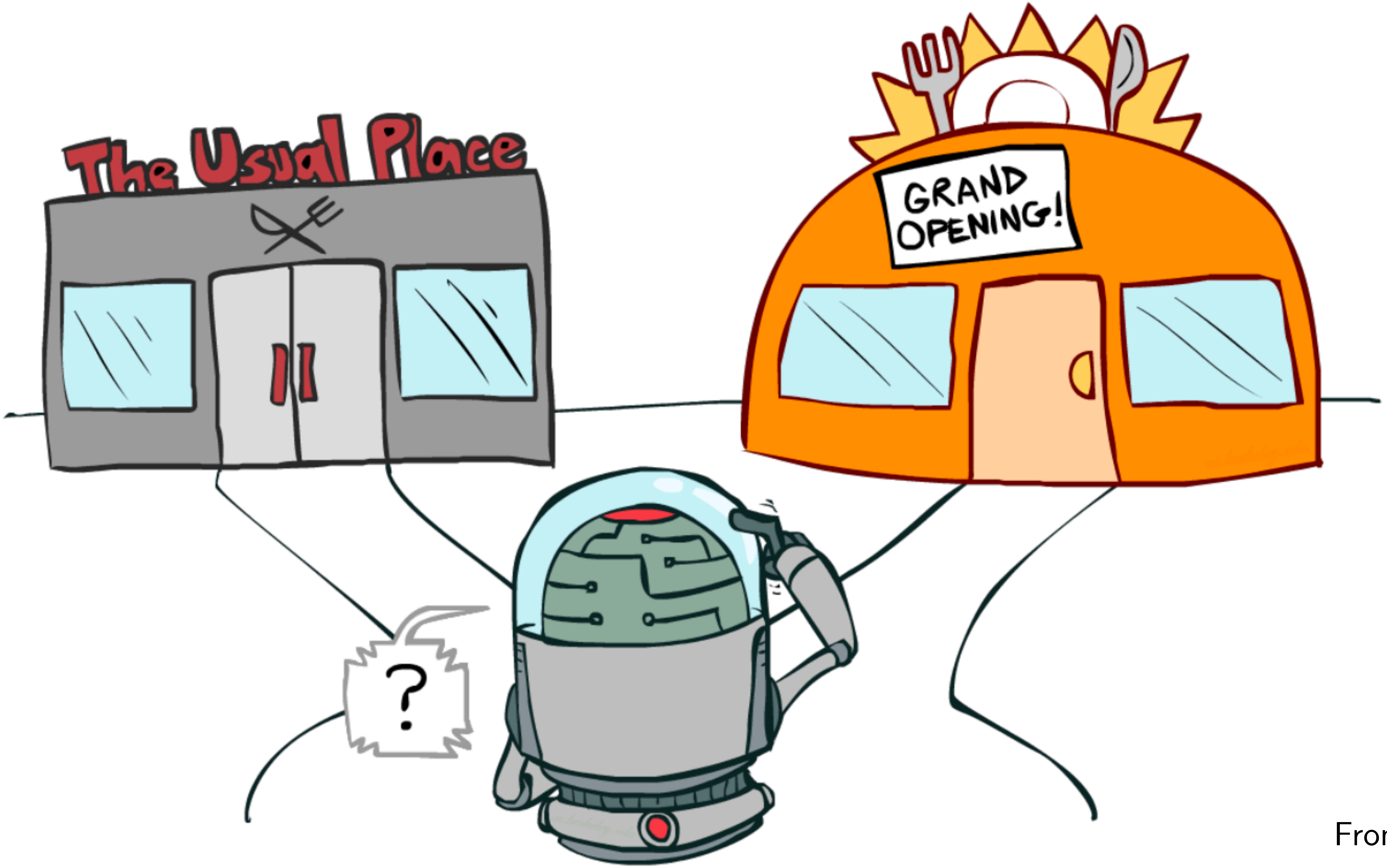
The story thus far ...

But what if the dynamics are unknown?

$$< S \, , A \, , C \, , \mathcal{T} >$$

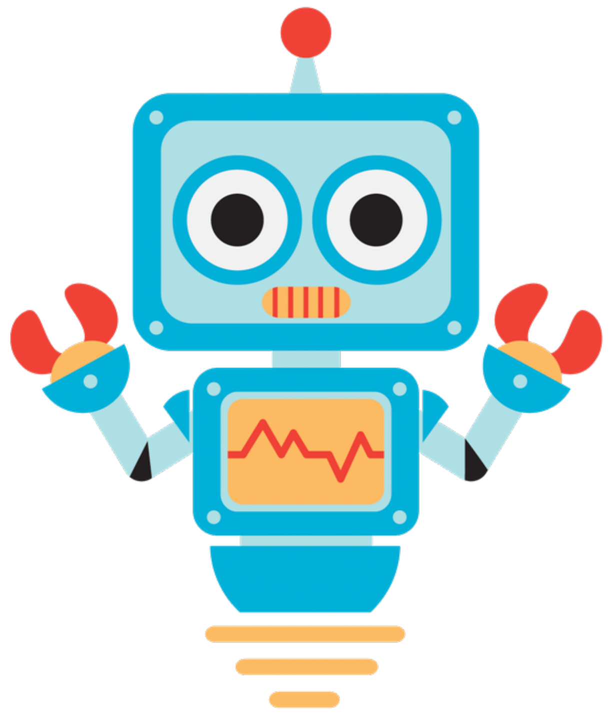


Exploration vs Exploitation



From Dan Klein

Doors



a^1 ?

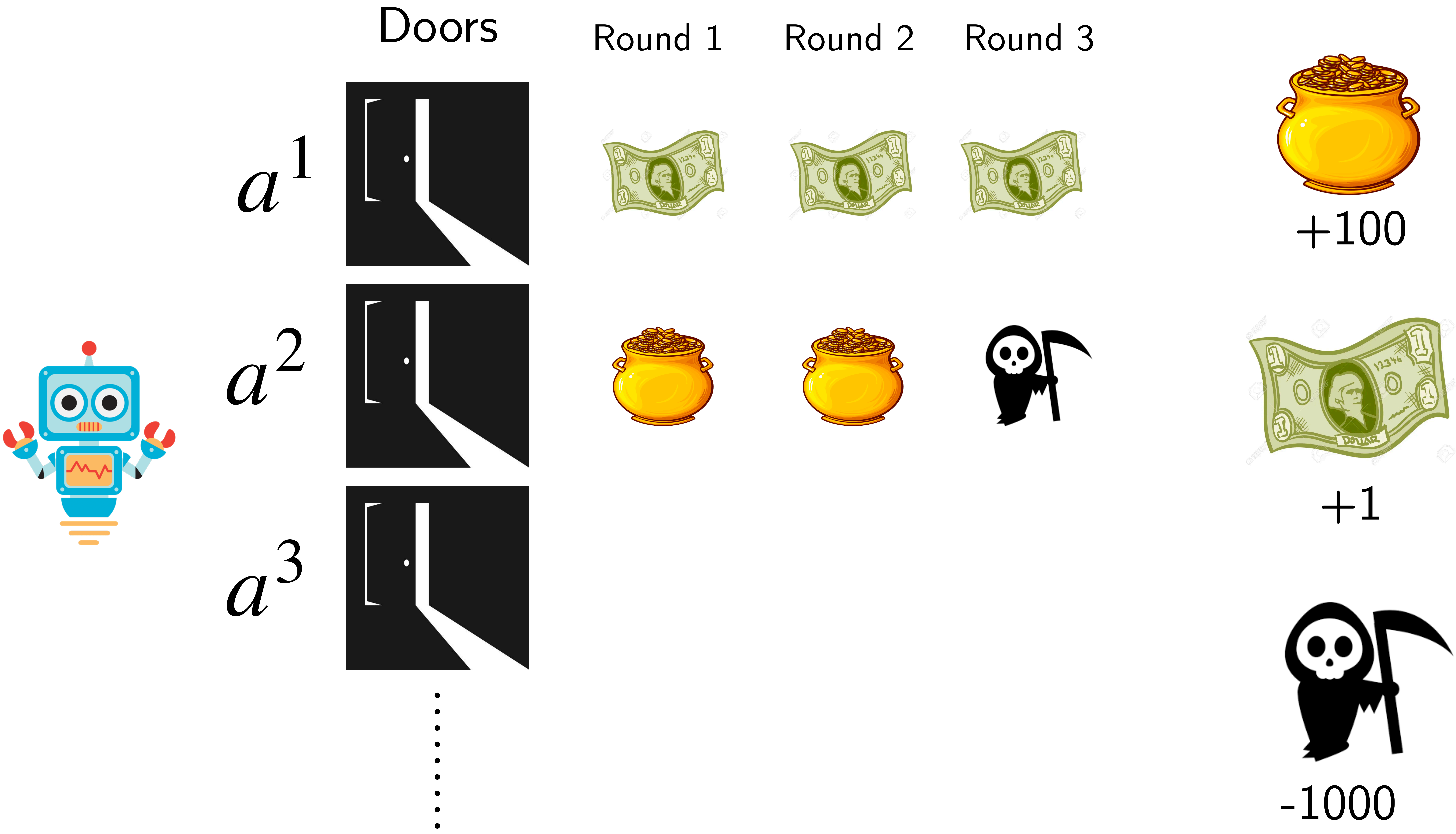
a^2 ?

a^3 ?

+100

+1

-1000


Doors
Round 1
Round 2
Round 3
a^1
a^2
a^3
+100
+1
-1000

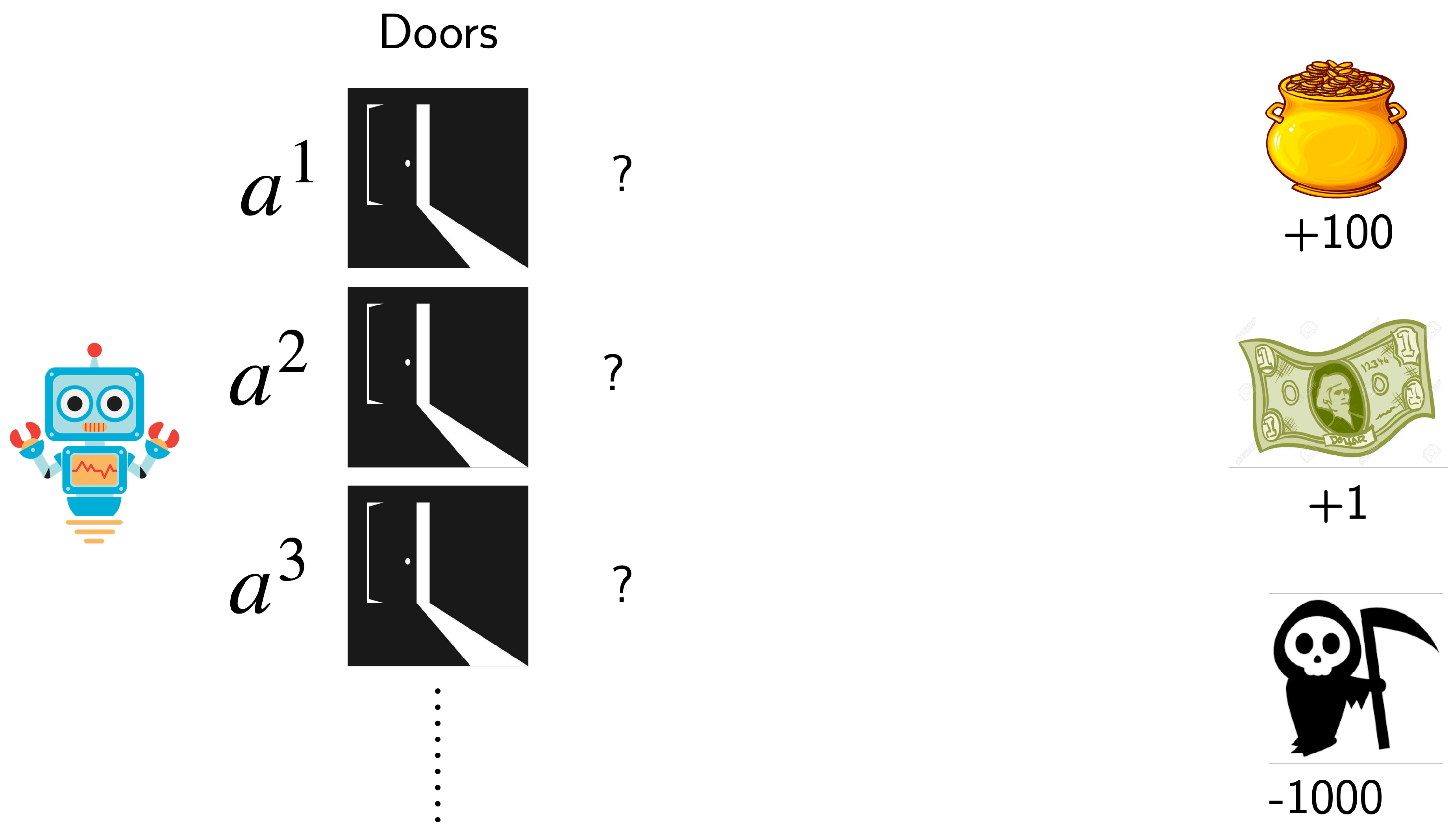
Activity!

Think-Pair-Share

Think (30 sec): What strategy would you pick doors?

Pair: Find a partner

Share (45 sec): Partners exchange ideas



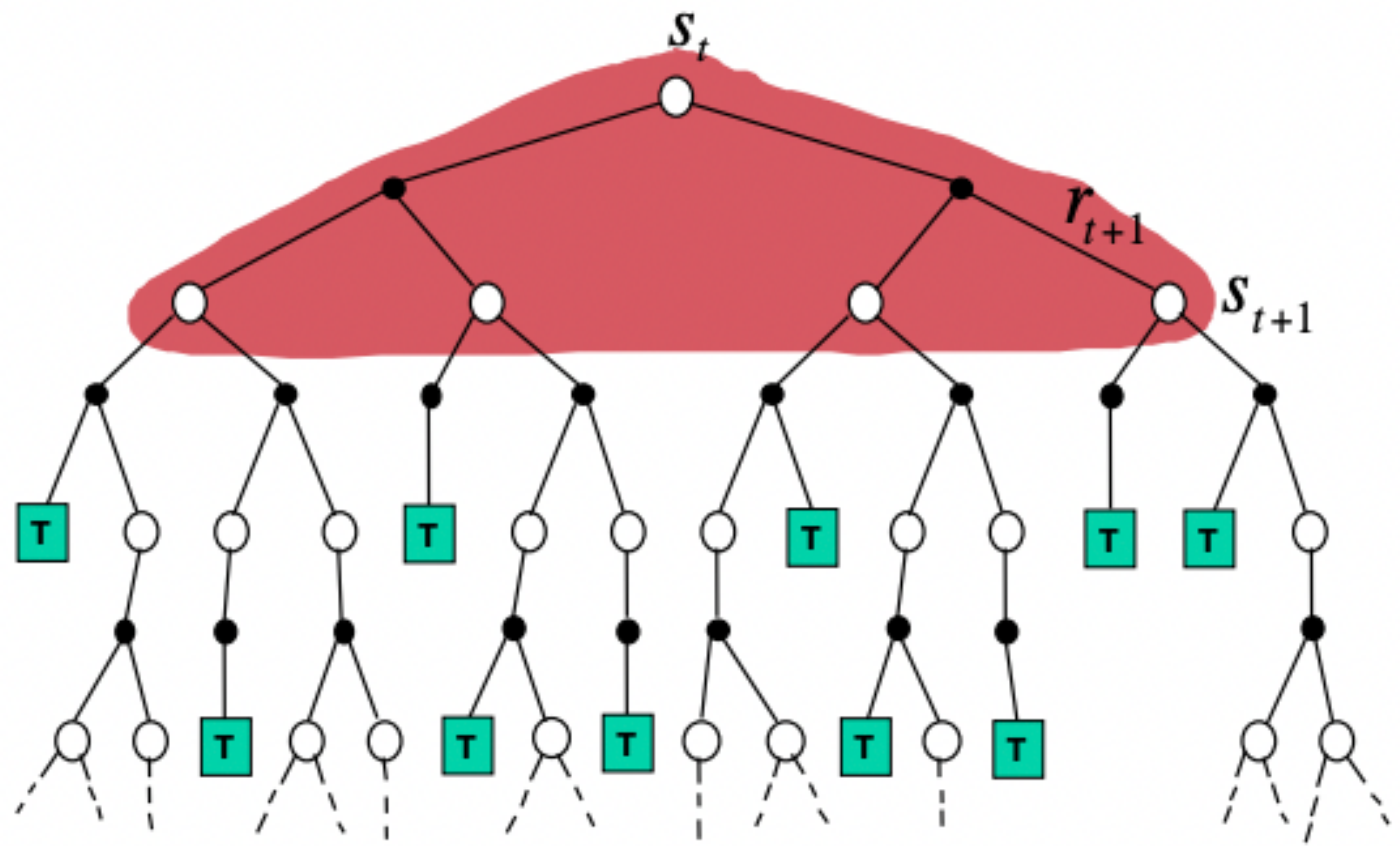
What if we played the game over multiple time steps?

+100
+1
$t = 1$
$t = 2$

Don't see how good a door is until the end of the episode

When we know the MDP: Dynamic Programming!

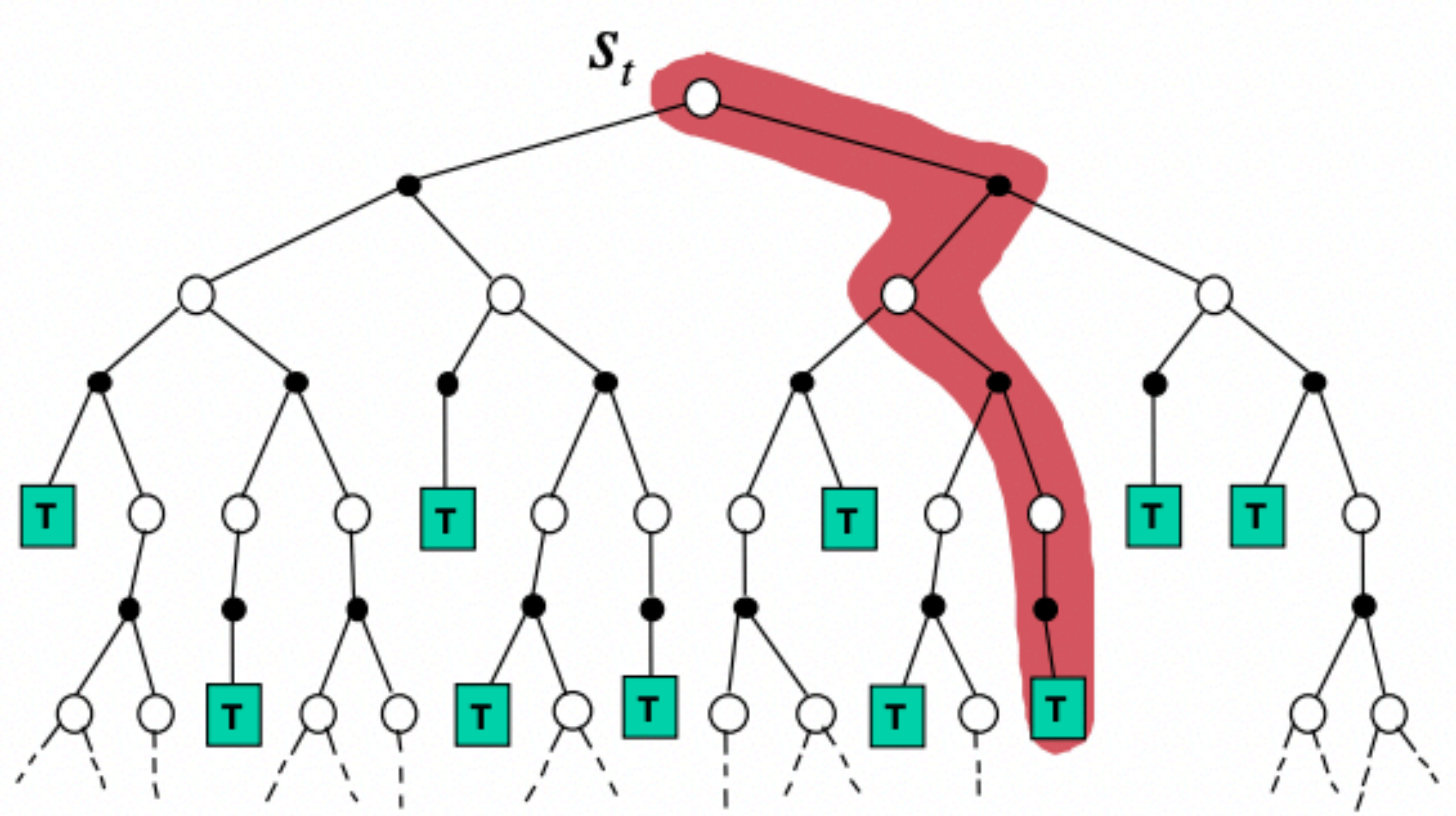
$$V(S_t) \leftarrow \mathbb{E}_\pi \left[R_{t+1} + \gamma V(S_{t+1})\right]$$



When we don't know MDP: *Estimate* Values

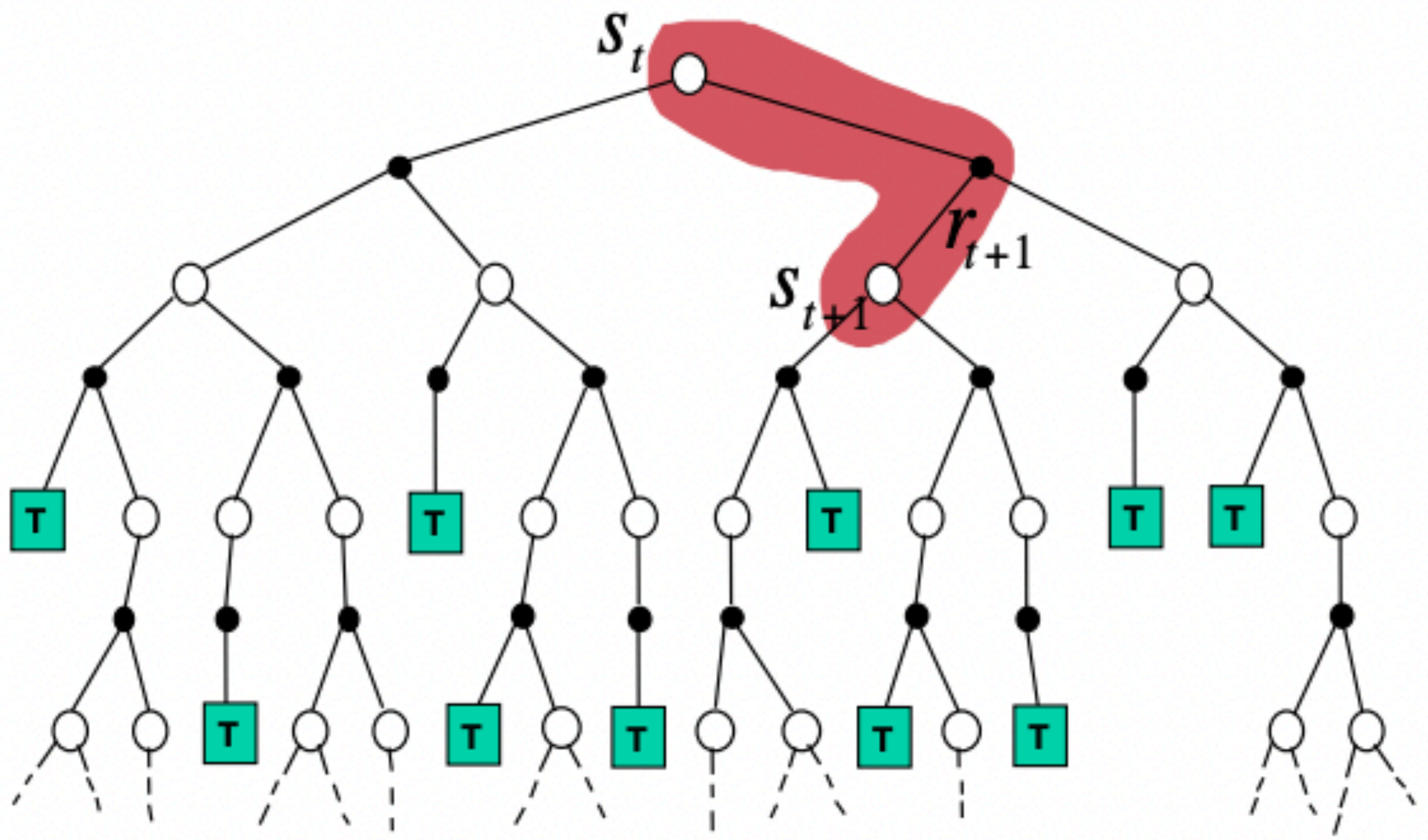
Monte Carlo

$$V(S_t) \leftarrow V(S_t) + \alpha\left(G_t - V(S_t)\right)$$

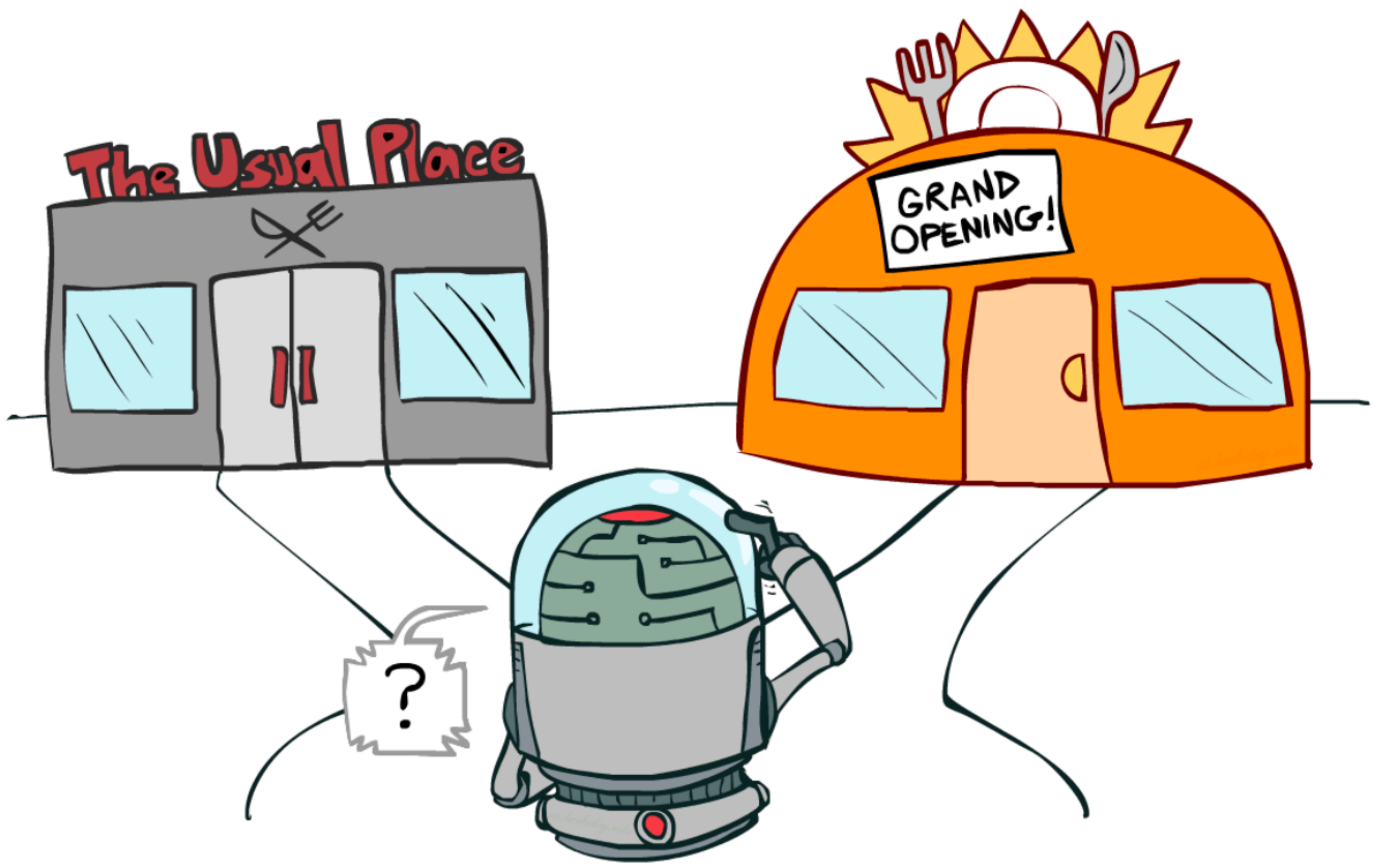


Temporal Difference

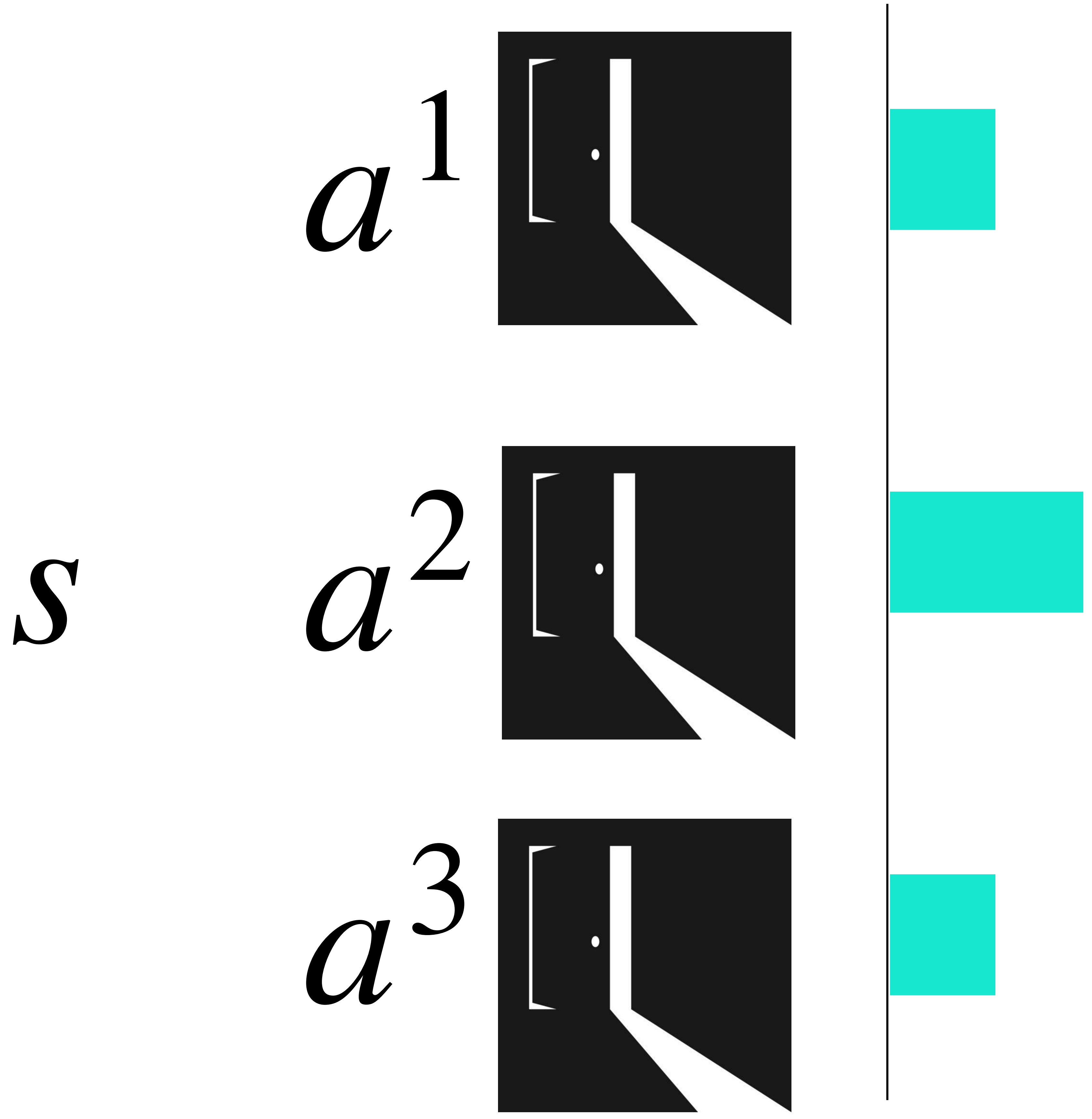
$$V(S_t) \leftarrow V(S_t) + \alpha\left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right)$$



Two Ingredients of RL



Exploration Exploitation



Estimate Values $Q(s, a)$

What is an application where we *really* need RL?


Games!

Why are Games perfect for RL?

You know the cost function
* +1 for winning and - 1 for losing

You don't know the transition function
* Don't know what opponent will do, stochasticity in dice rolls etc.

Perfect for learning from trial and error (by playing itself!)

Name the Game!

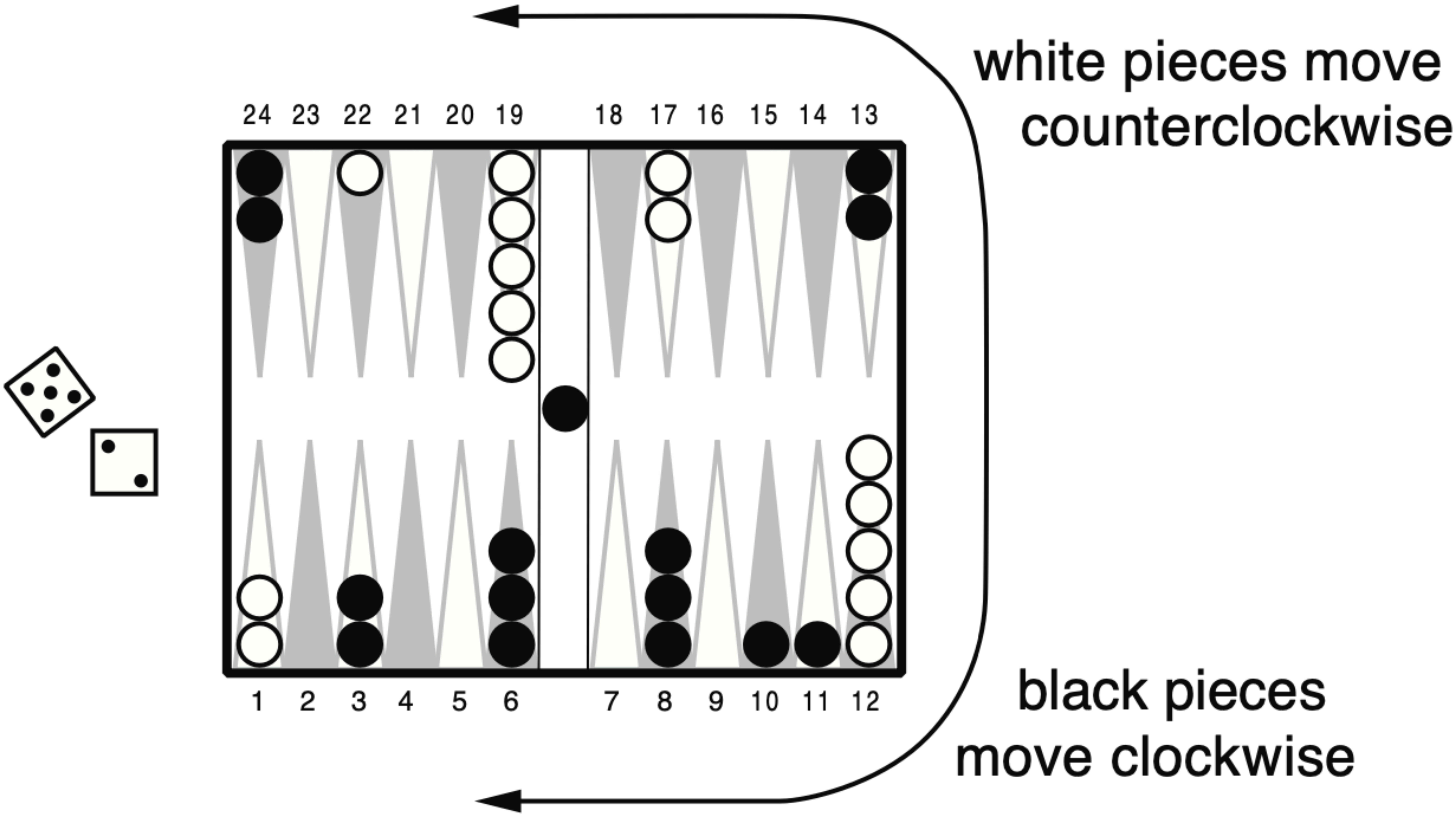
One of the biggest success stories of RL

An agent trained via
self-play and neural networks

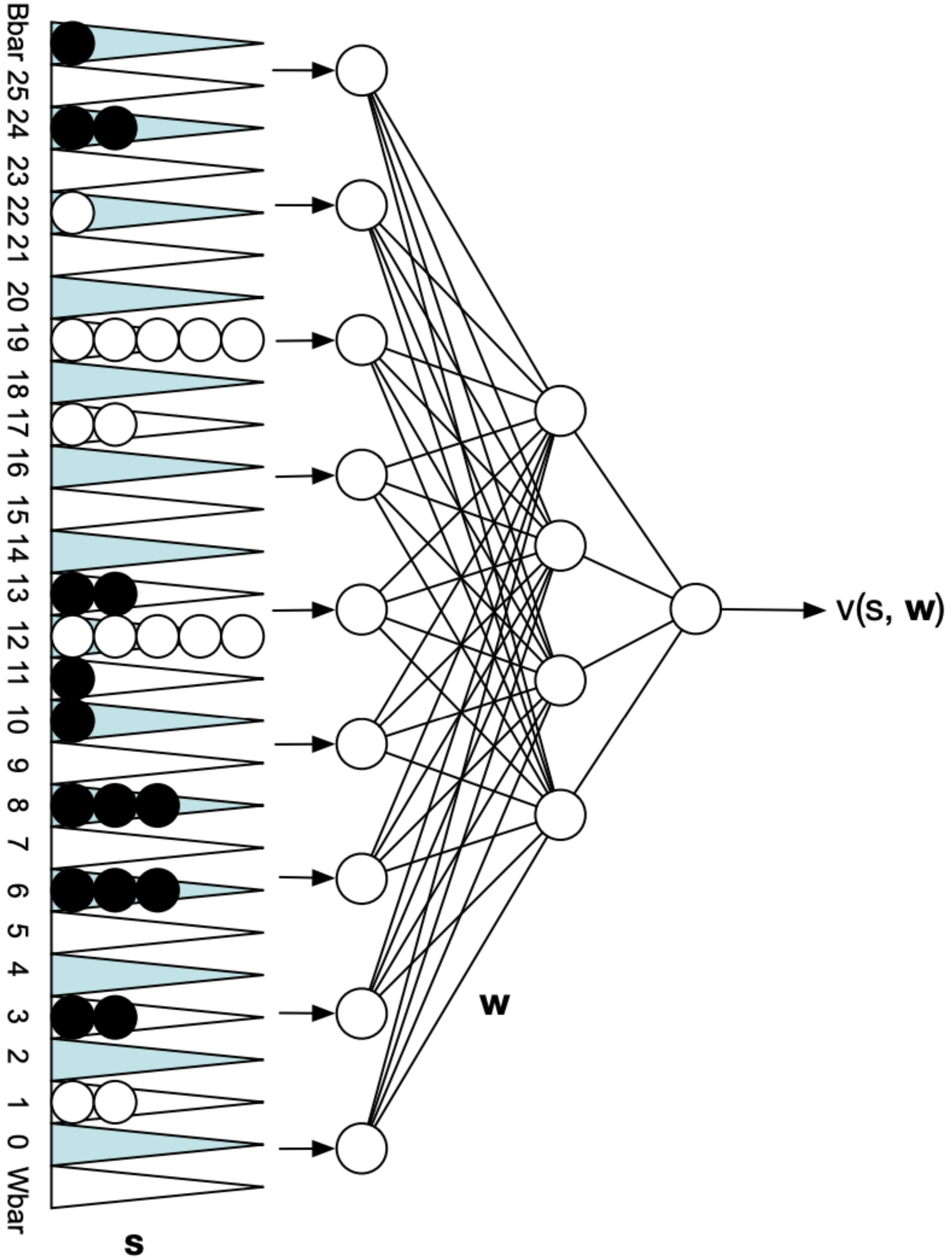
Beat the world champion

Discovered totally new moves

The game of Backgammon



A backgammon position



Temporal Difference Learning and TD-Gammon

By Gerald Tesauro

TD-Gammon

Initialised with random weights

Trained by games of self-play

Using non-linear temporal-difference learning

$$\delta_t = v(S_{t+1}, \mathbf{w}) - v(S_t, \mathbf{w})$$
$$\Delta \mathbf{w} = \alpha \delta_t \nabla_{\mathbf{w}} v(S_t, \mathbf{w})$$

Greedy policy improvement (no exploration)

Algorithm always converged in practice

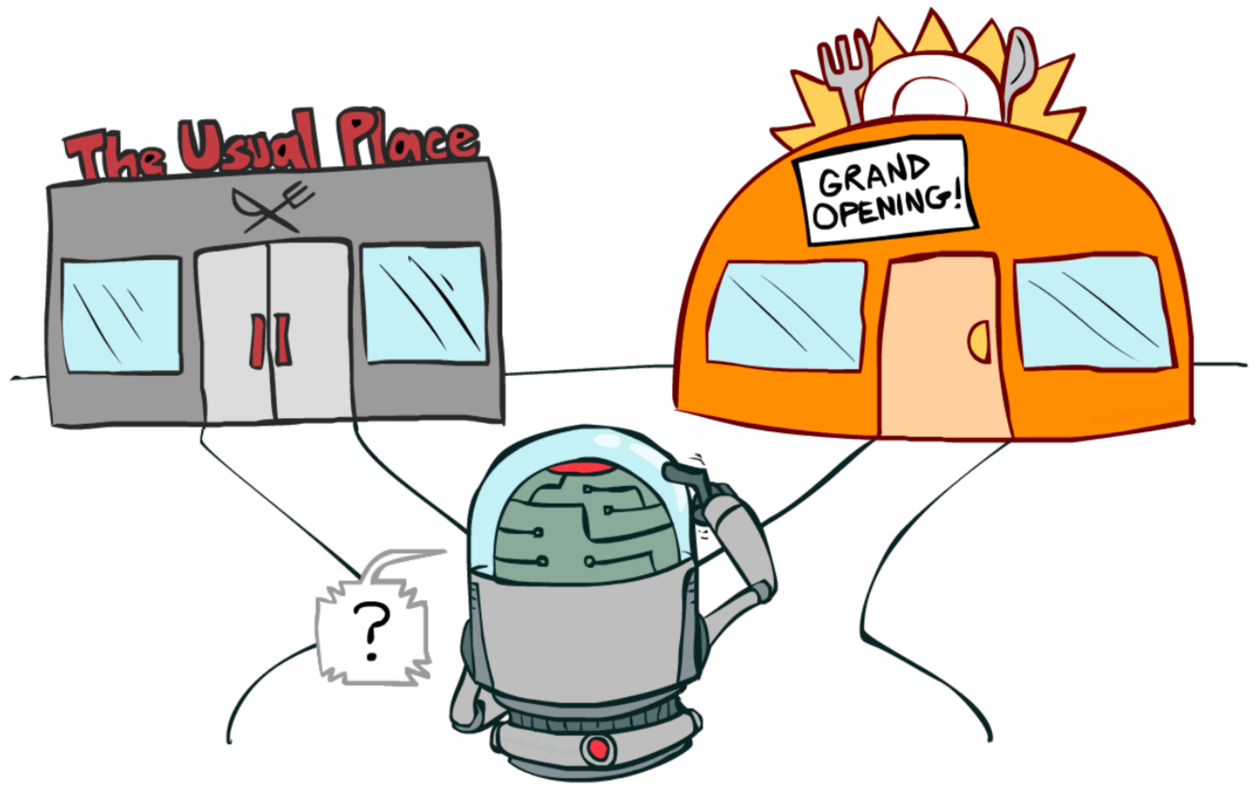
From David Silver

Why was TD Gammon such a big deal?

1) Power of self- play
2) First time system could deal with stochastic uncertainty in dynamics (this broke everything in the deep blue style chess engines that were first super human)
3) Actually learned a value function: first "human" like behavior rather than just deep search with some heuristic values.
4) Changed elements of how backgammon is played because it demonstrated that certain positions were more valuable than self play
5) Demonstrated the TD idea could be scaled to super human performance at a game previously unaddressable.
6) Use a "deep" (i.e. Neural) representation within RL algorithms
7) Saw the benefit of imitation learning, but eventually got better than imitation
8) Showed boosting performance by some explicit forward search

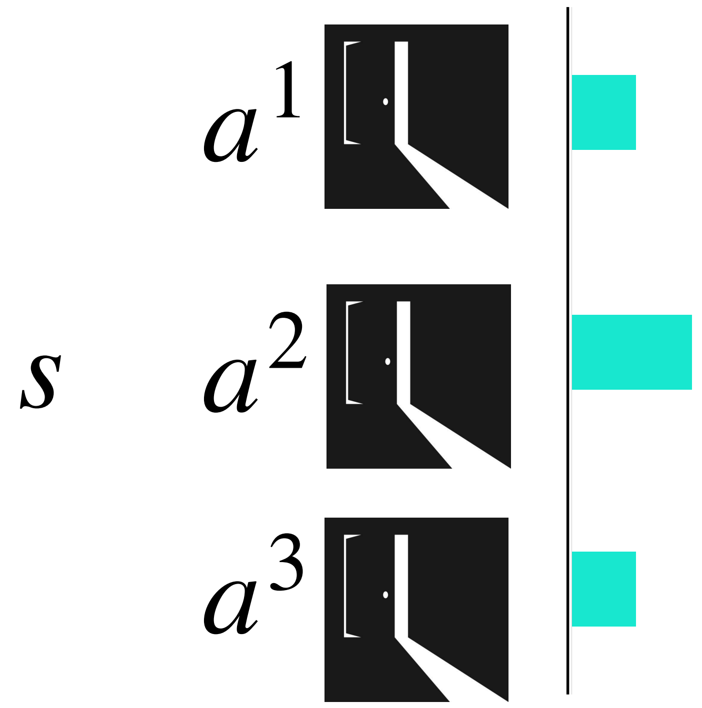
Two Ingredients of RL: TD-Gammon

No exploration,
was just greedy



Exploration Exploitation

Used temporal difference
to estimate values



Estimate Values

Okay, but what about games with more complex representations?

Circa 2013

Playing Atari with Deep Reinforcement Learning

Volodymyr Mnih **Koray Kavukcuoglu** **David Silver** **Alex Graves** **Ioannis Antonoglou**

Daan Wierstra **Martin Riedmiller**

DeepMind Technologies

{vlad,koray,david,alex.graves,ioannis,daan,martin.riedmiller} @ deepmind.com


ima...
021 3 1

Use NN to learn Q-function and then use to infer the optimal policy, $\pi(s)$

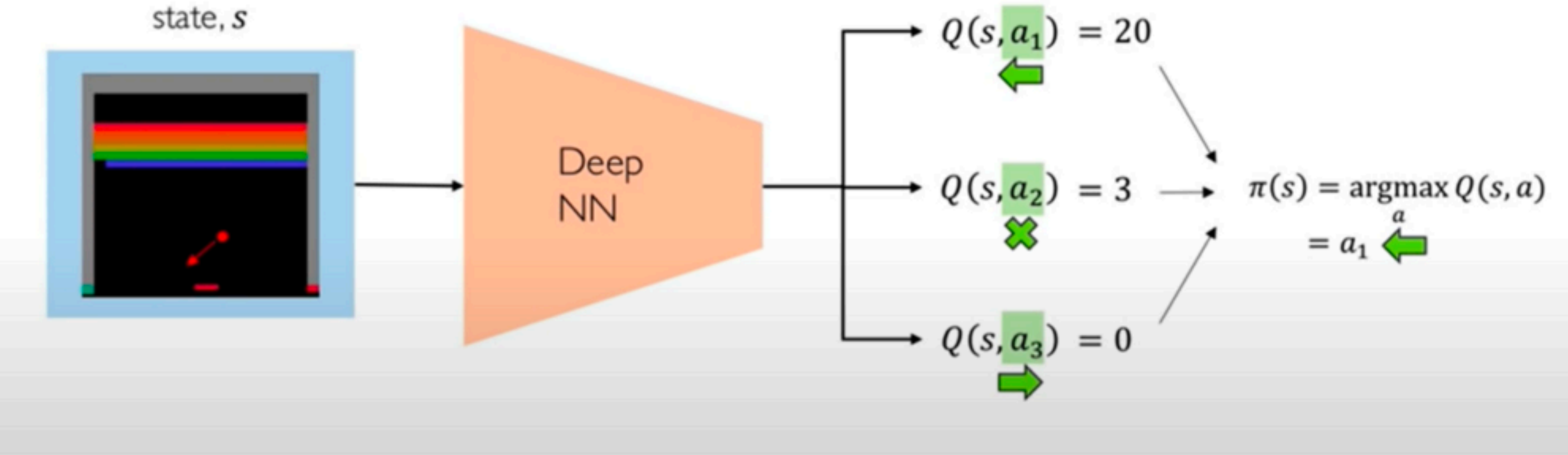
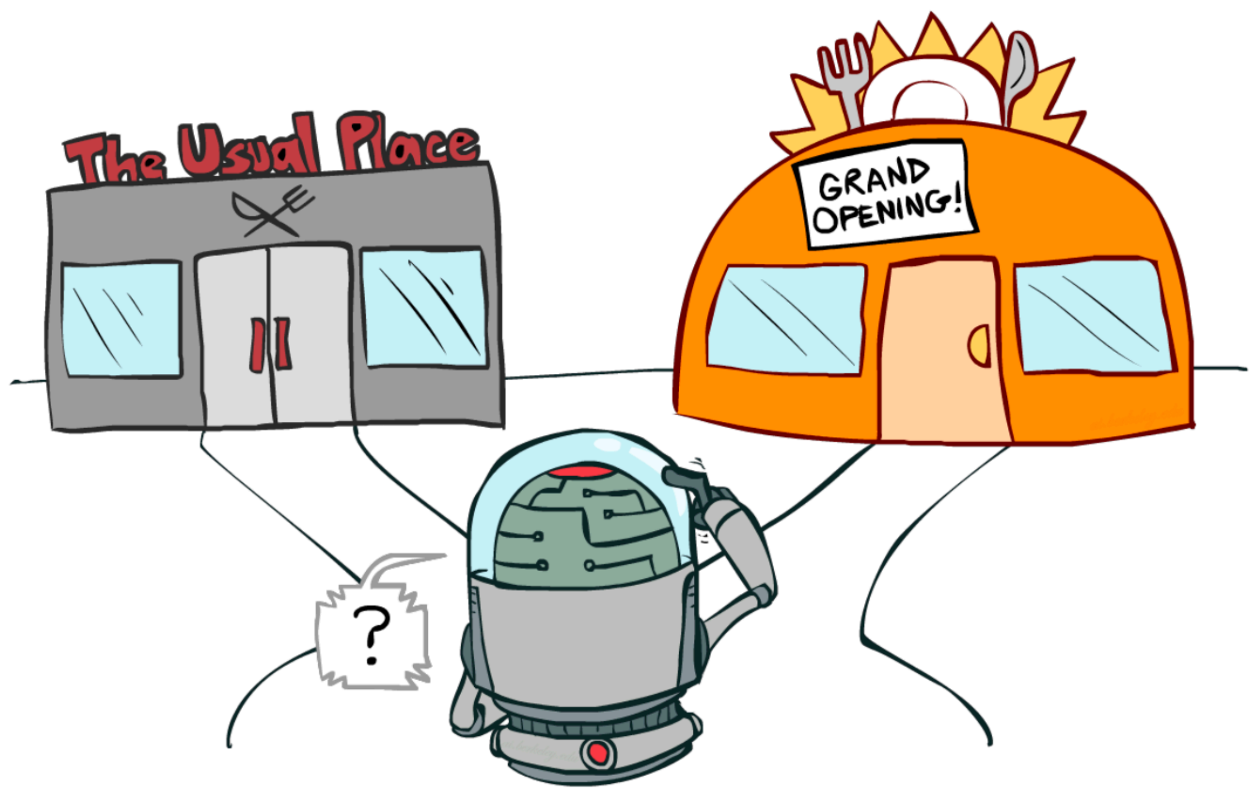


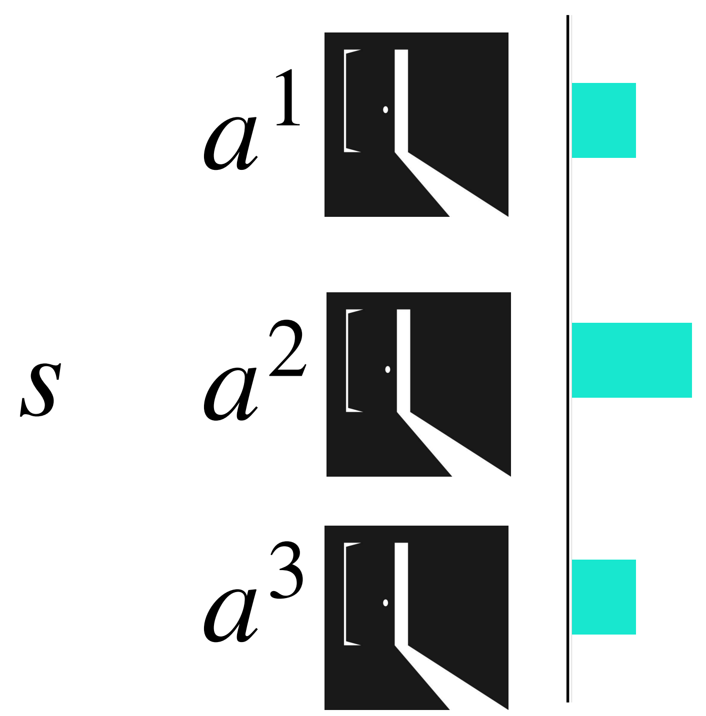
Image Source: MIT Introduction to Deep Learning

Two Ingredients of RL: DQN

Epsilon-Greedy

Used Q-learning with
Deep CNN





Exploration Exploitation

Estimate Values

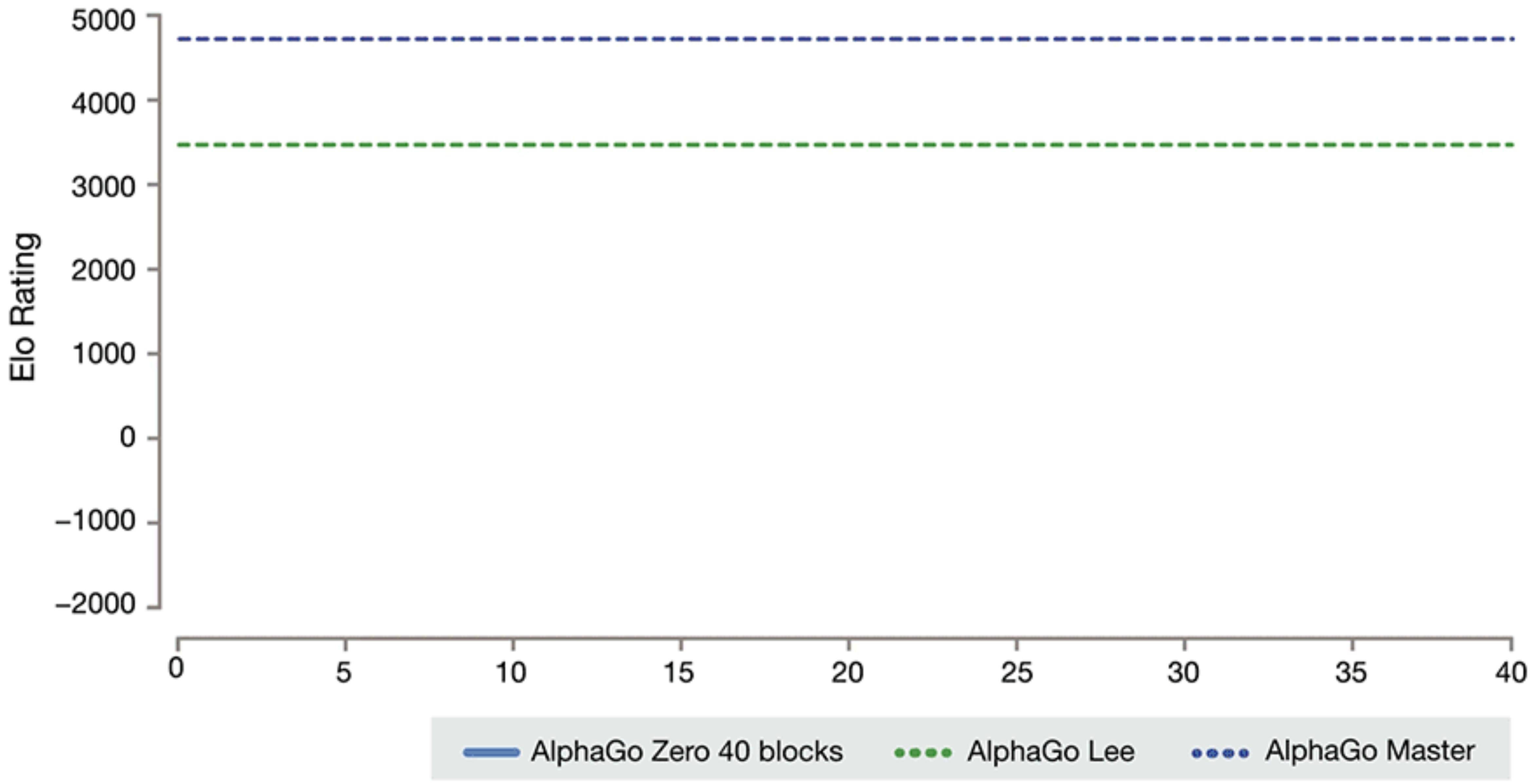
Okay, but what about games that require some level of planning?

AlphaGoZero

Mastering the game of Go without human knowledge

David Silver[1*], Julian Schrittwieser[1*], Karen Simonyan[1*], Ioannis Antonoglou[1], Aja Huang[1], Arthur Guez[1], Thomas Hubert[1], Lucas Baker[1], Matthew Lai[1], Adrian Bolton[1], Yutian Chen[1], Timothy Lillicrap[1], Fan Hui[1], Laurent Sifre[1], George van den Driessche[1], Thore Graepel[1] & Demis Hassabis[1]

AlphaGo



AlphaGoZero

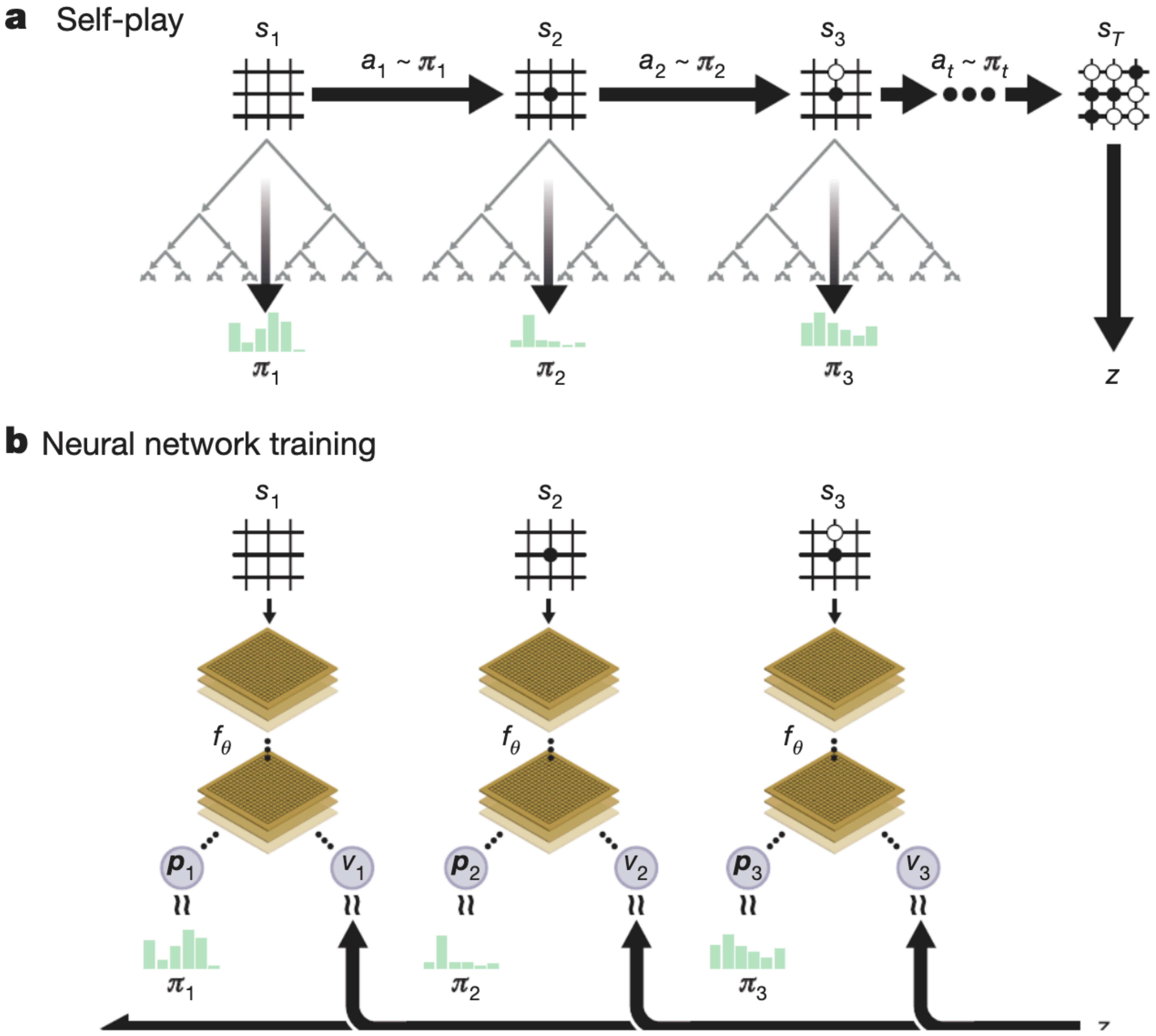


Figure 1 | Self-play reinforcement learning in AlphaGo Zero. a, The program plays a game $s_1, ..., s_T$ against itself. In each position s_t, an MCTS α_θ is executed (see Fig. 2) using the latest neural network f_θ. Moves are selected according to the search probabilities computed by the MCTS, $a_t \sim \boldsymbol{\pi}_t$. The terminal position s_T is scored according to the rules of the game to compute the game winner z. **b**, Neural network training in AlphaGo Zero. The neural network takes the raw board position s_t as its input, passes it through many convolutional layers with parameters θ, and outputs both a vector $\boldsymbol{p}_t$, representing a probability distribution over moves, and a scalar value v_t, representing the probability of the current player winning in position s_t. The neural network parameters θ are updated to maximize the similarity of the policy vector $\boldsymbol{p}_t$ to the search probabilities $\boldsymbol{\pi}_t$, and to minimize the error between the predicted winner v_t and the game winner z (see equation (1)). The new parameters are used in the next iteration of self-play as in **a**.

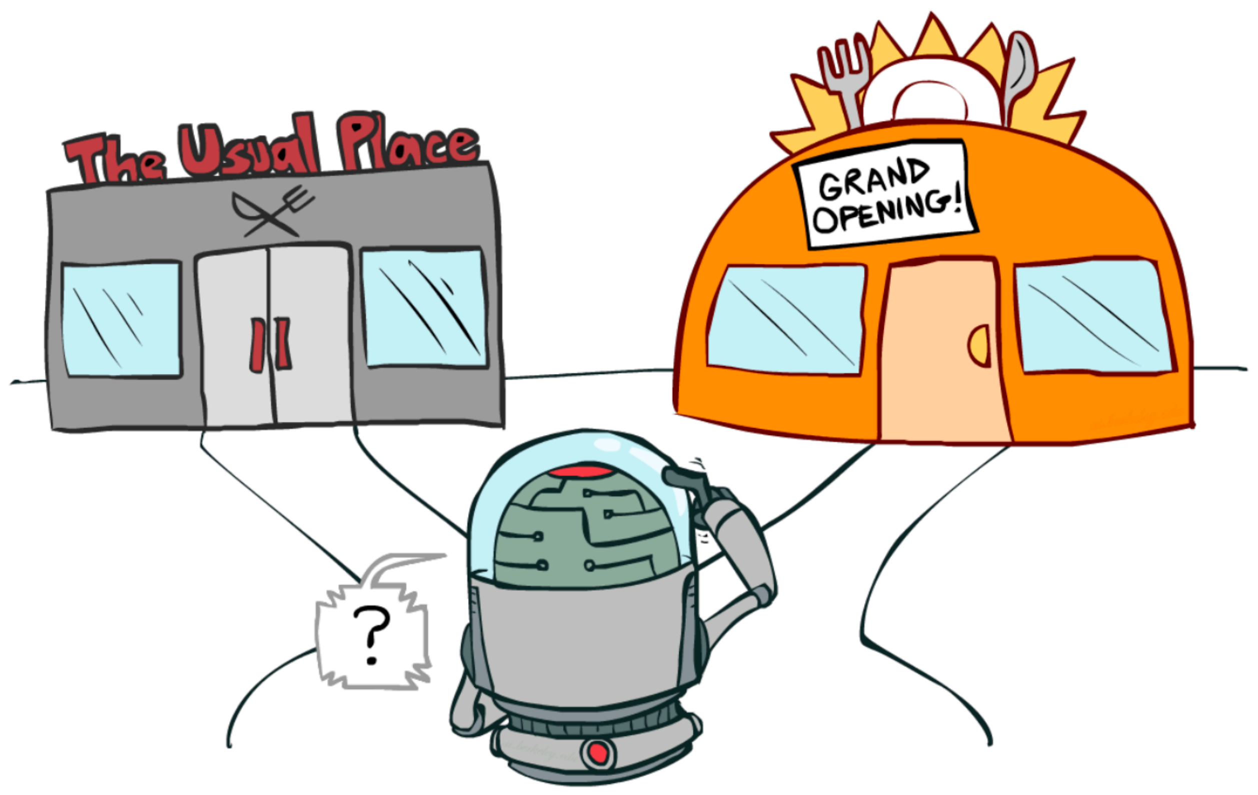
Two Ingredients of RL: AlphaGoZero

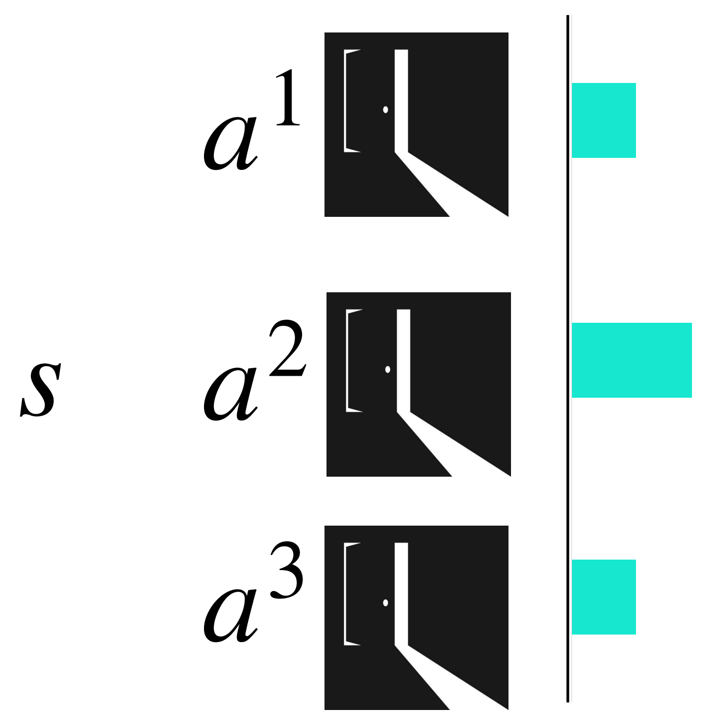
Monte Carlo Tree Search

Deep value network

+

Deep policy network





Exploration Exploitation

Estimate Values

So what happens when we replace the human with a robot?

photo: Axel Griesh/MPG, Munich

Learning Strategies in Table Tennis using Inverse Reinforcement Learning

Katharina Muelling · Abdeslam Boularias · Betty Mohler · Bernhard Schölkopf · Jan Peters

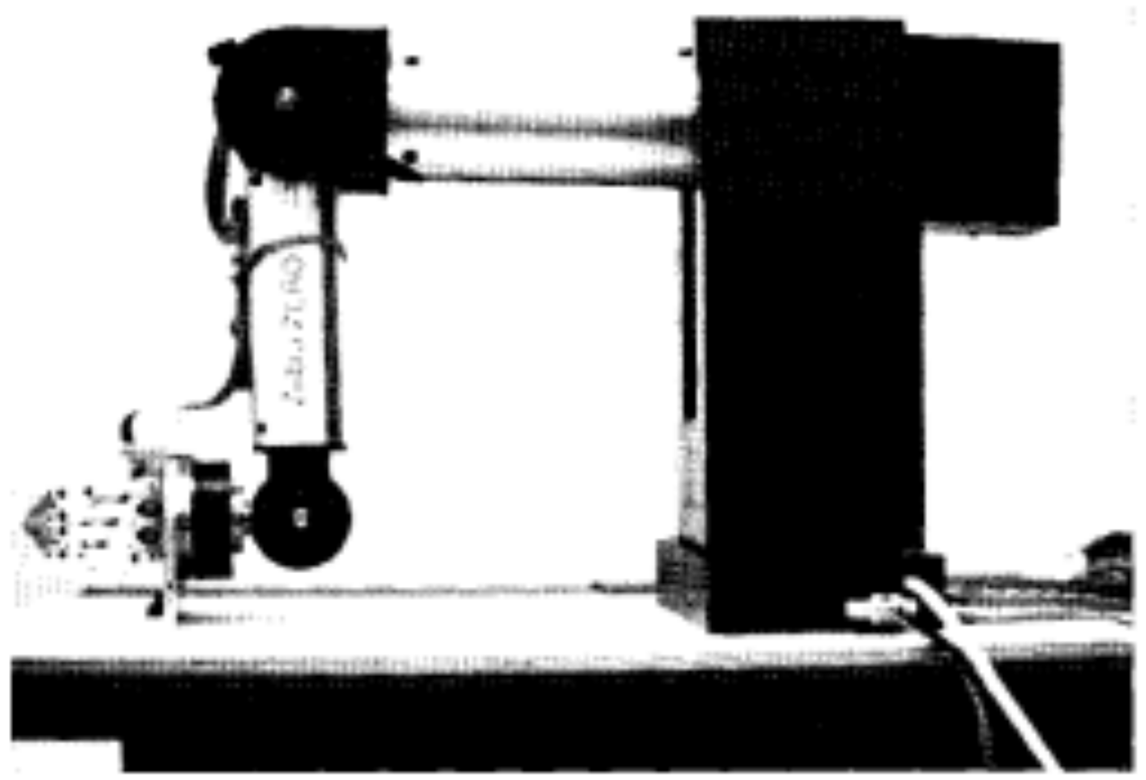
Reinforcement Learning in Robotics: A Survey

Jens Kober[*†] J. Andrew Bagnell[‡] Jan Peters[§¶]

email: jkober@cor-lab.uni-bielefeld.de, dbagnell@ri.cmu.edu, mail@jan-peters.net

1992

(a) OBELIX robot

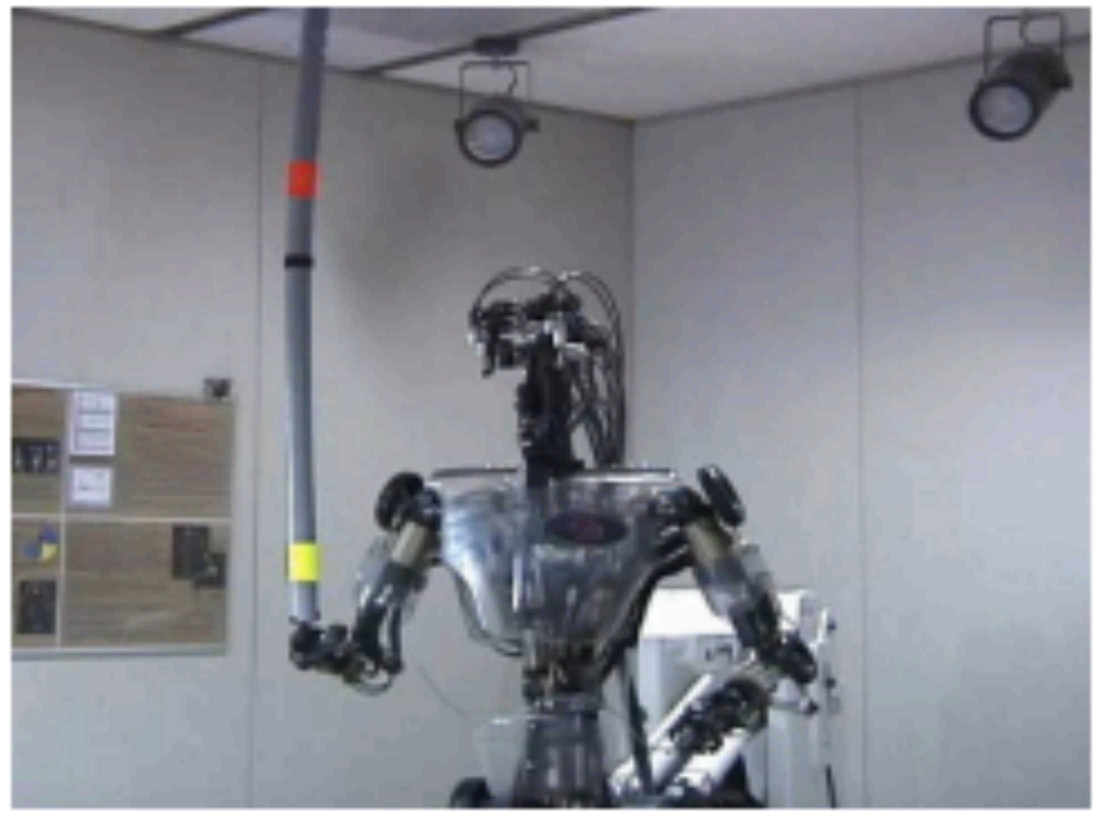


(b) Zebra Zero robot

1994

2001

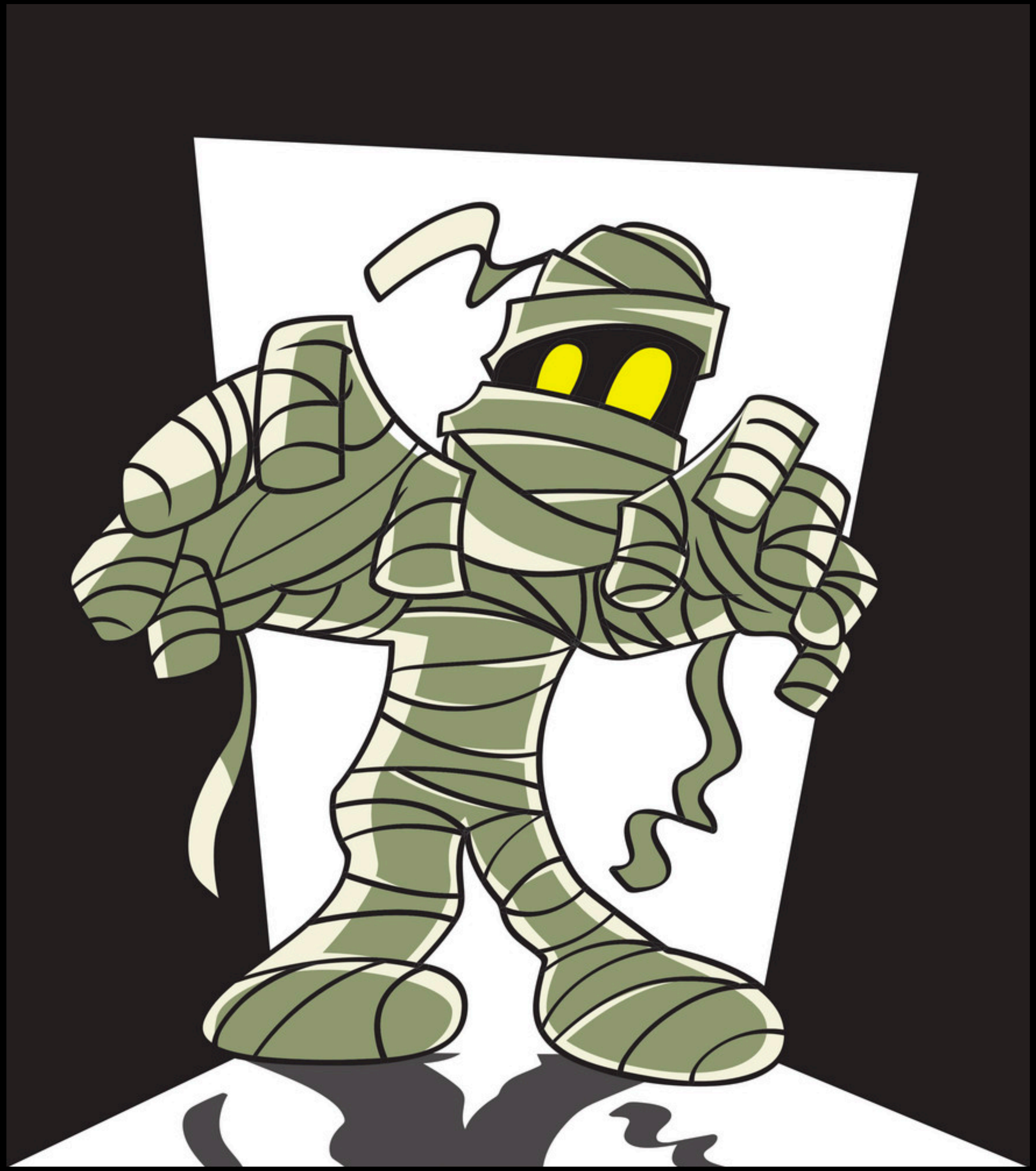
(c) Autonomous helicopter



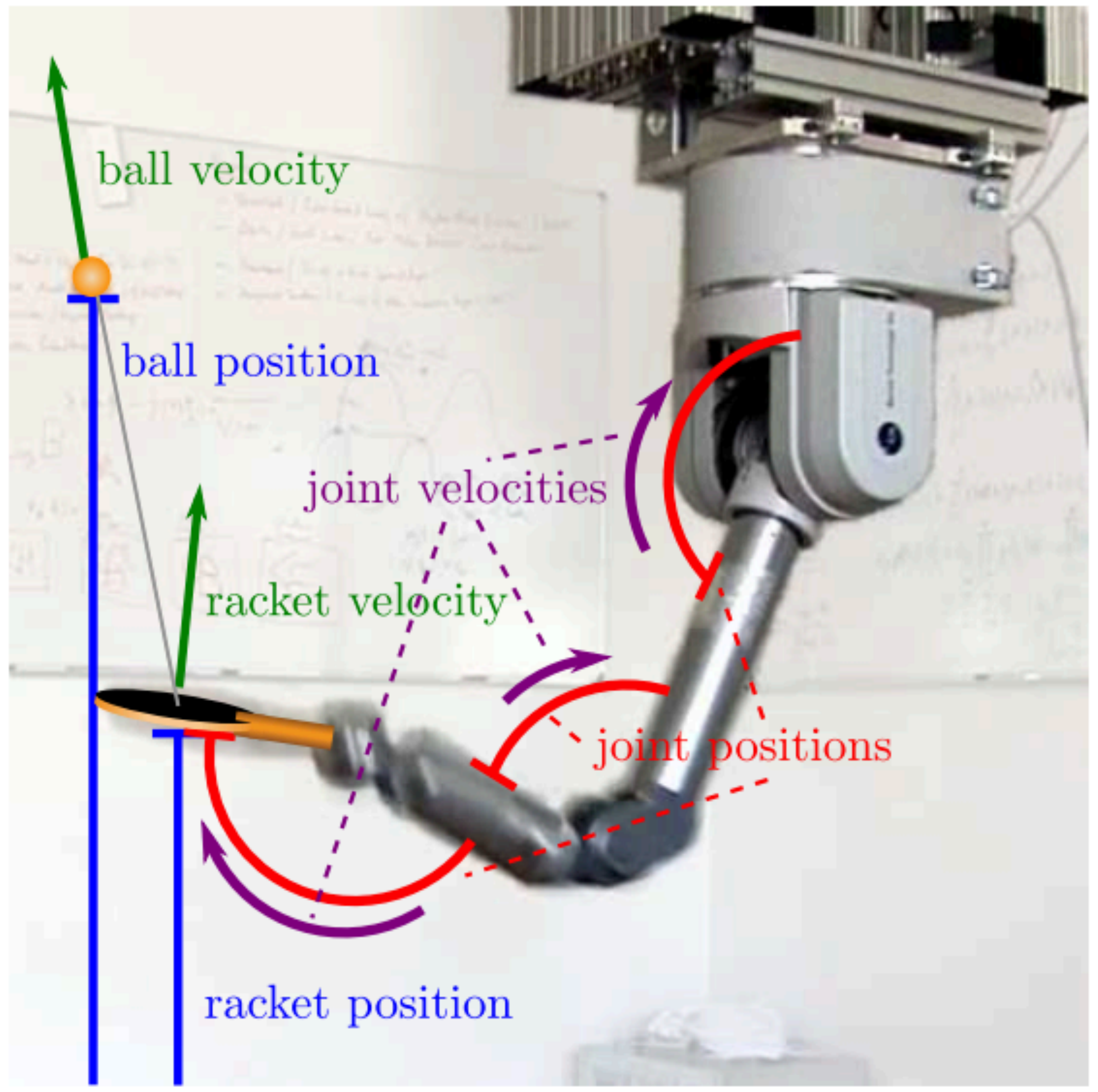
(d) Sarcos humanoid DB

1996

Curses in Robotics



Curses in RL for Robotics



Curse of Real-World Samples

Curse of Reward Specification

Curse of Dimensionality

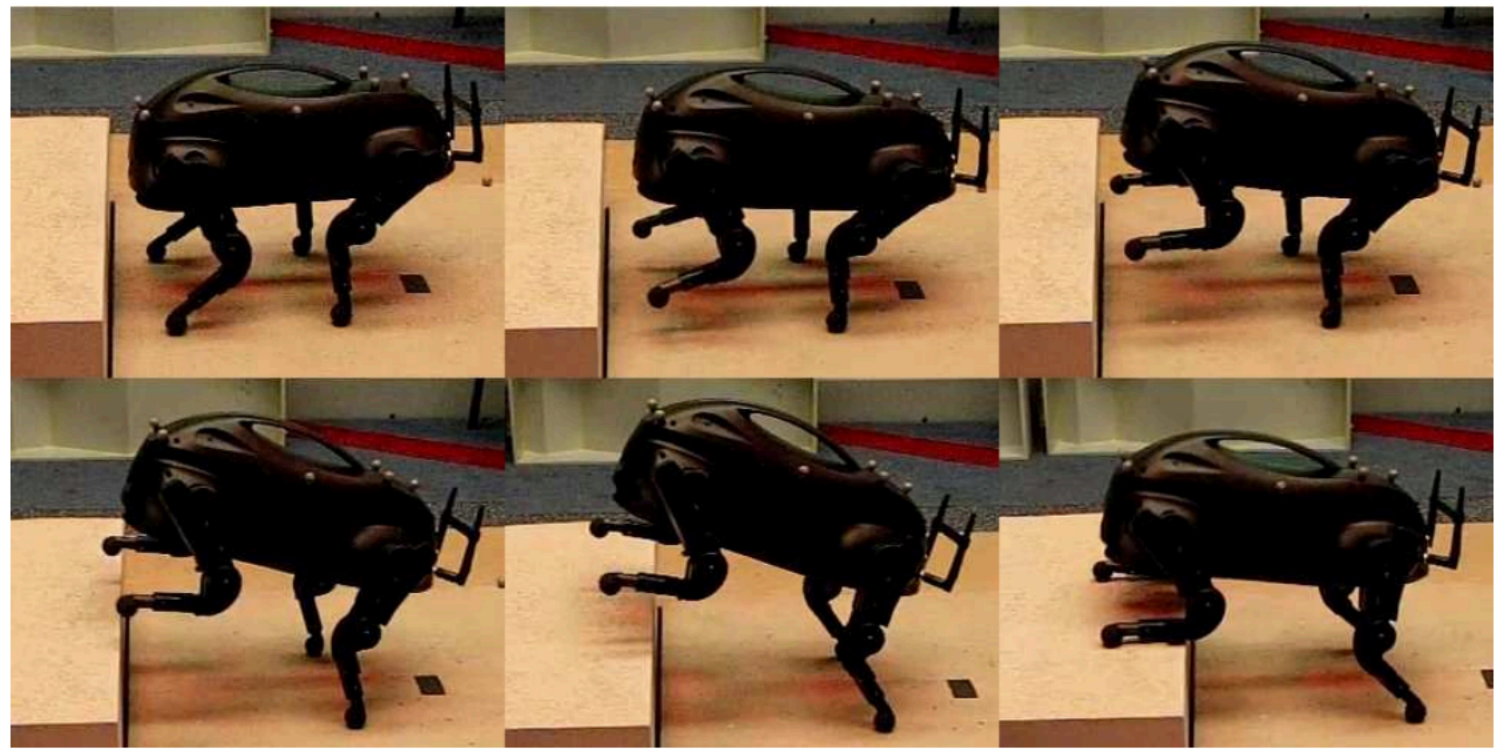
Curse of Model Uncertainty

No Silver Bullet!

Ingredients for Practical RL

Leveraging Demonstrations

Kakade et al, Bagnell and Scheider

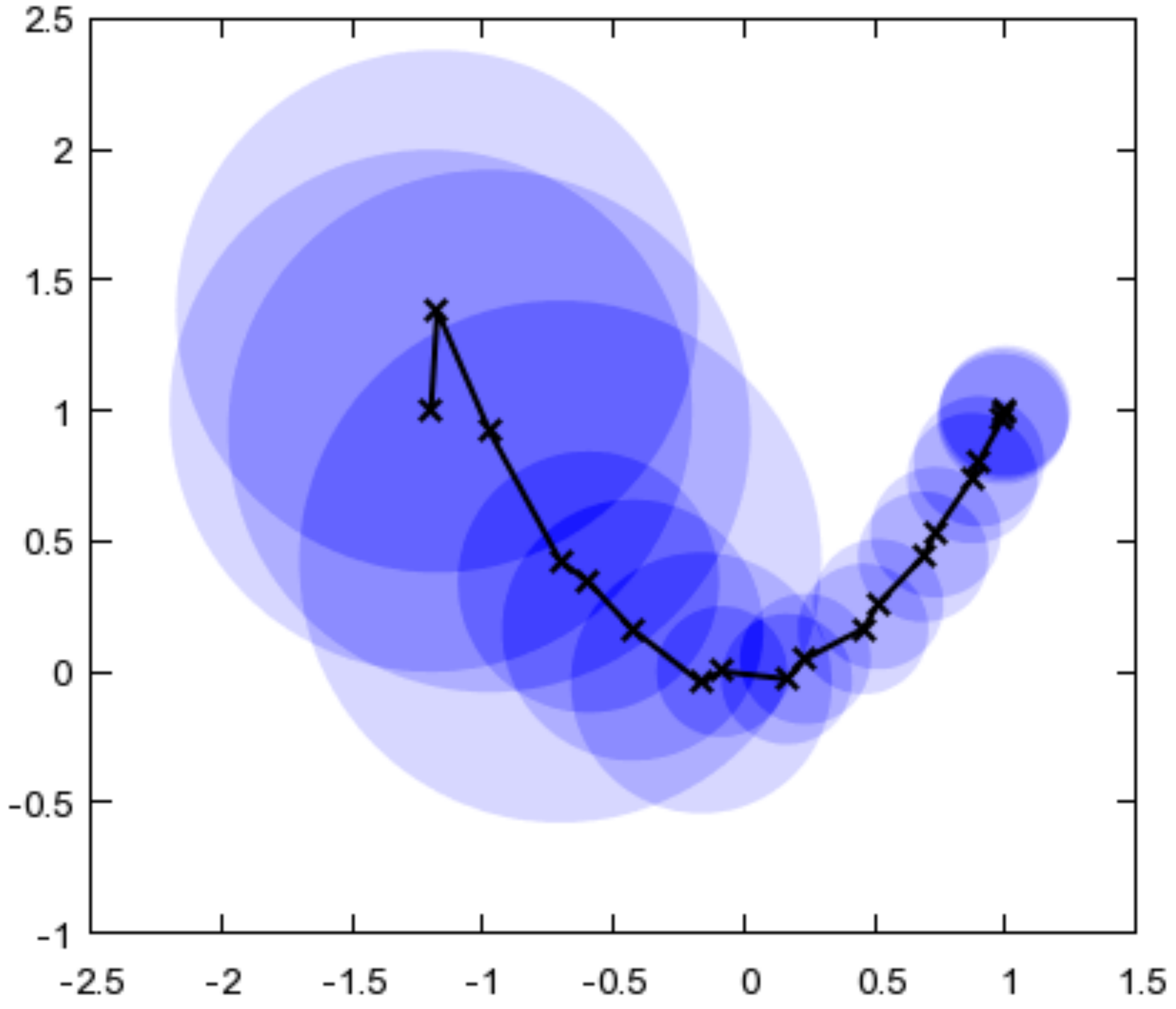


Appropriate Policy Representation

Locally linear (Kolter and Ng),
Dynamic Motor Primitives (Schaal)

Reward Shaping

Ng et al.



Conservatism and Trust

Kakade and Langford, Schulman et al.

Plannable Models

Abbeel and Ng

What about Deep RL?

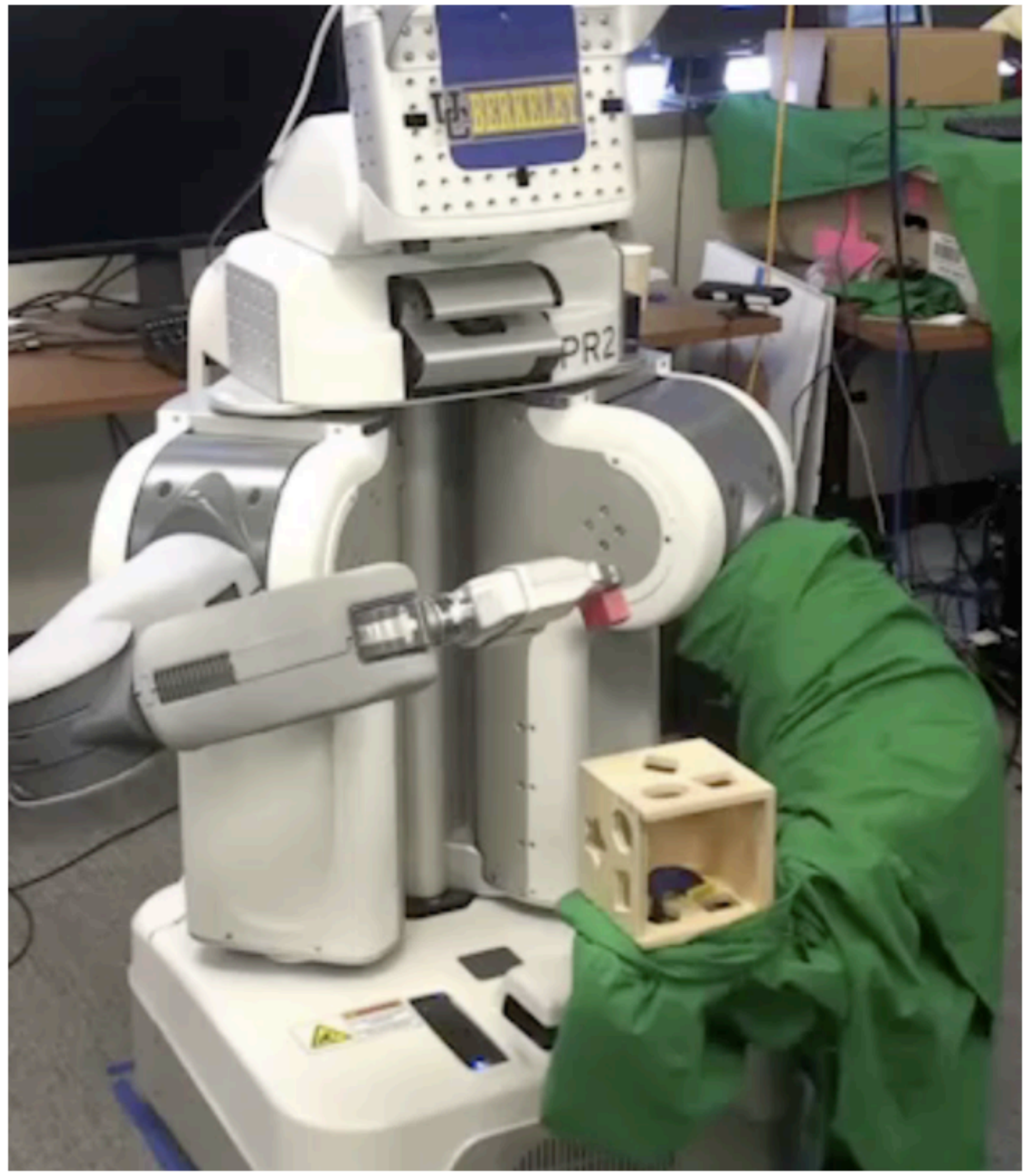

Position 2
real time
autonomous execution

How to Train Your Robot with Deep Reinforcement Learning – Lessons We've Learned

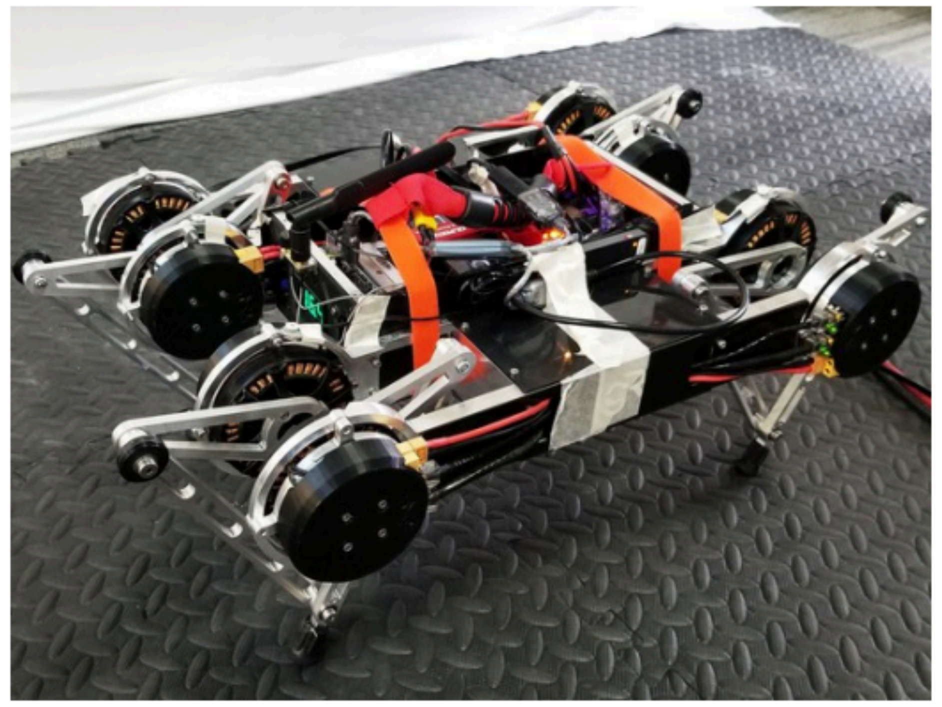
1–22

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

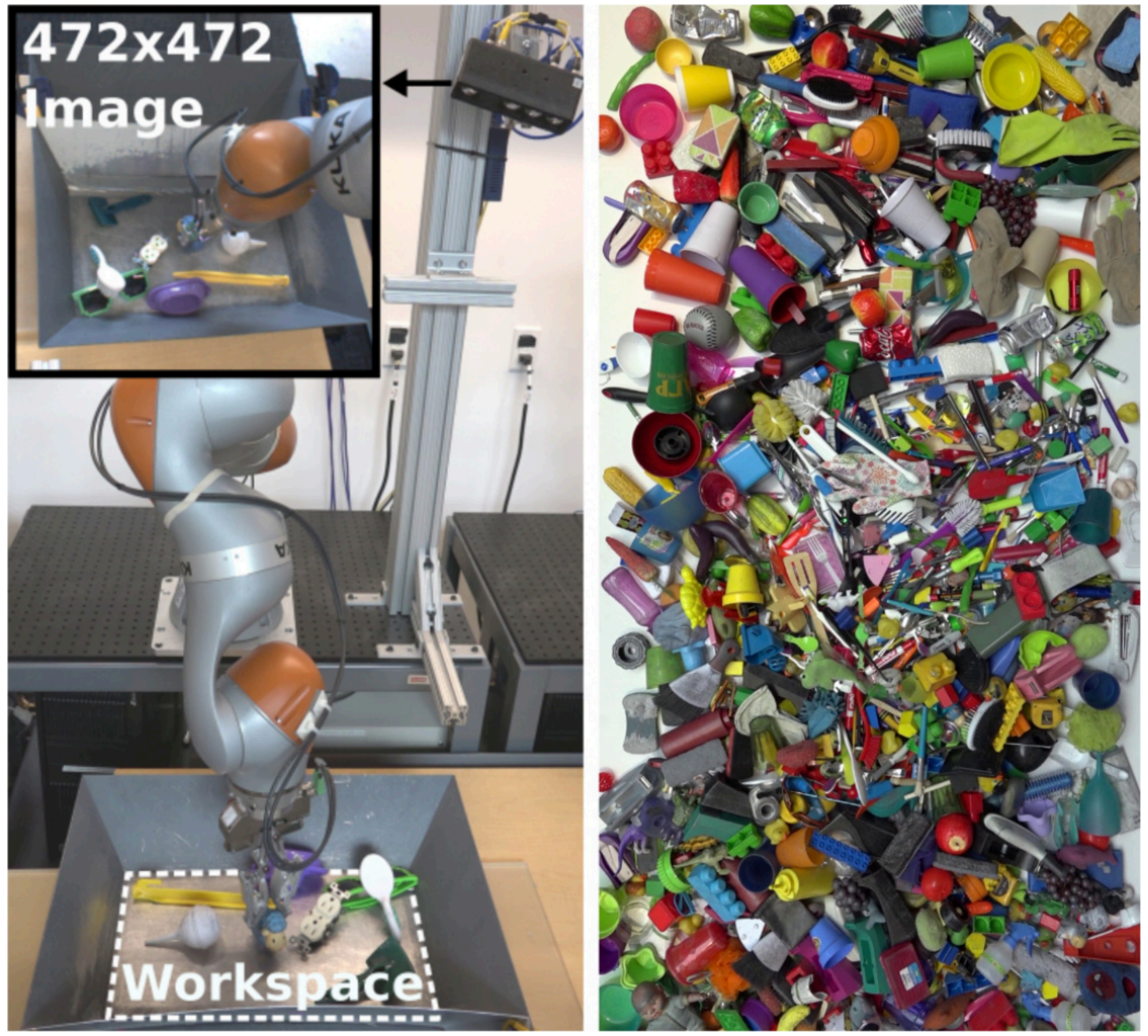
Julian Ibarz[1], Jie Tan[1], Chelsea Finn[1,3], Mrinal Kalakrishnan[2], Peter Pastor[2], Sergey Levine[1,4]



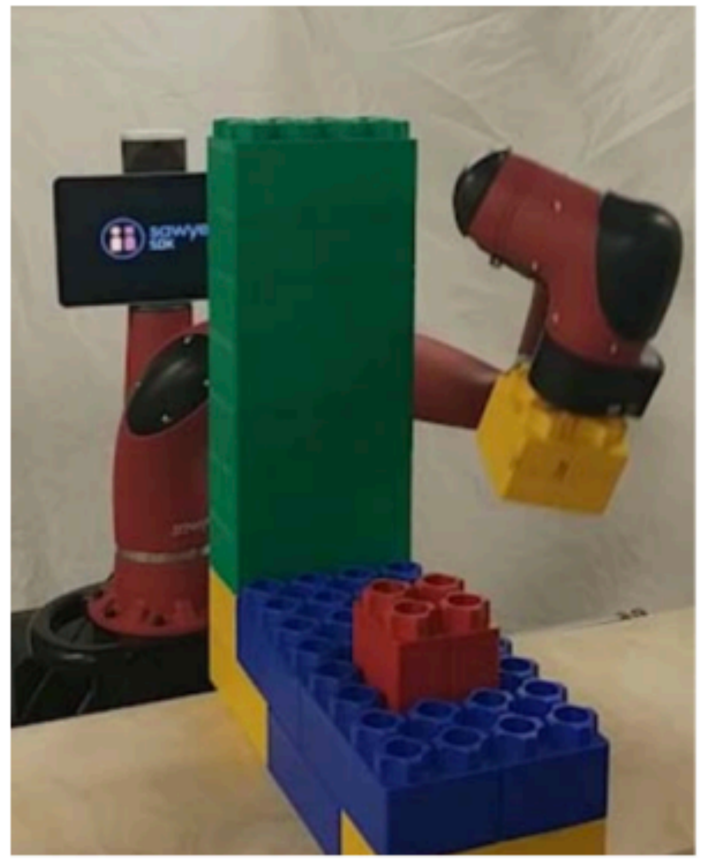
Levine et al 2016



Harnoja et al. 2019

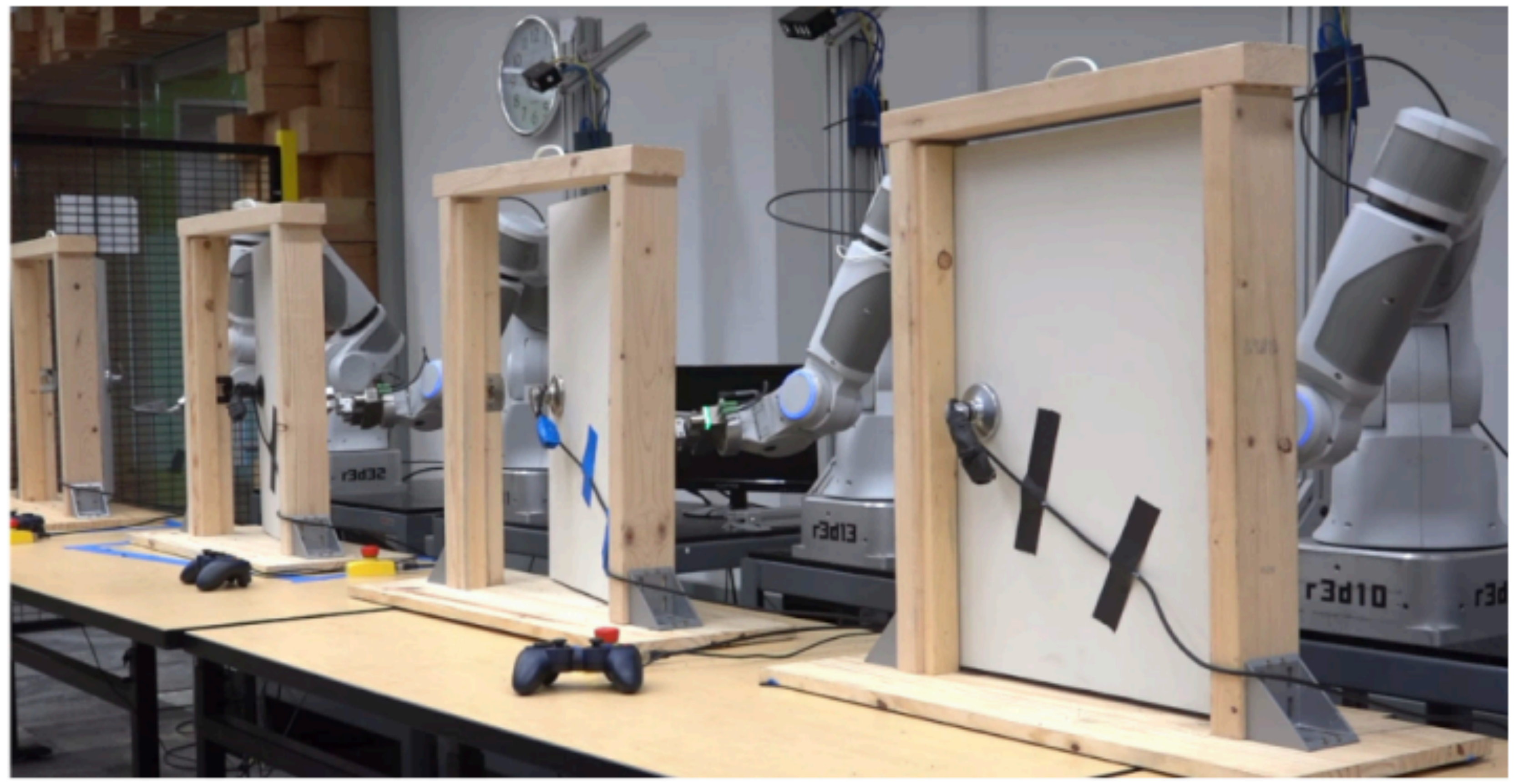


Kalashnikov et al. 2018



(a) block stacking

Harnoja et al 2018



(b) door opening

Gu et al 2017

There is
no Deep RL

There is no "Deep RL"

There is
Deep Learning

There is
Reinforcement Learning

Better representations for state / value function