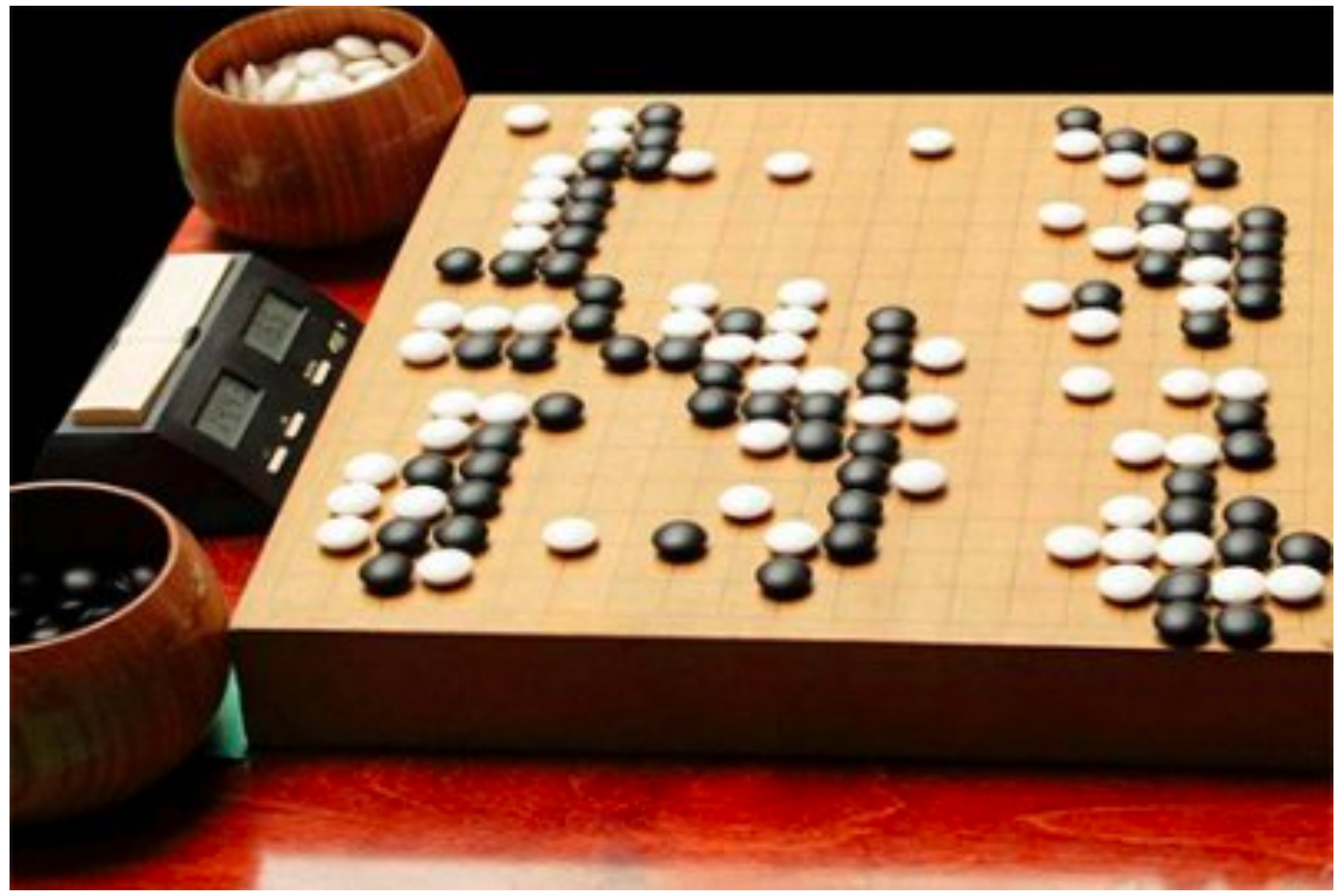
Deep Reinforcement Learning: Foundations and Recent Advances

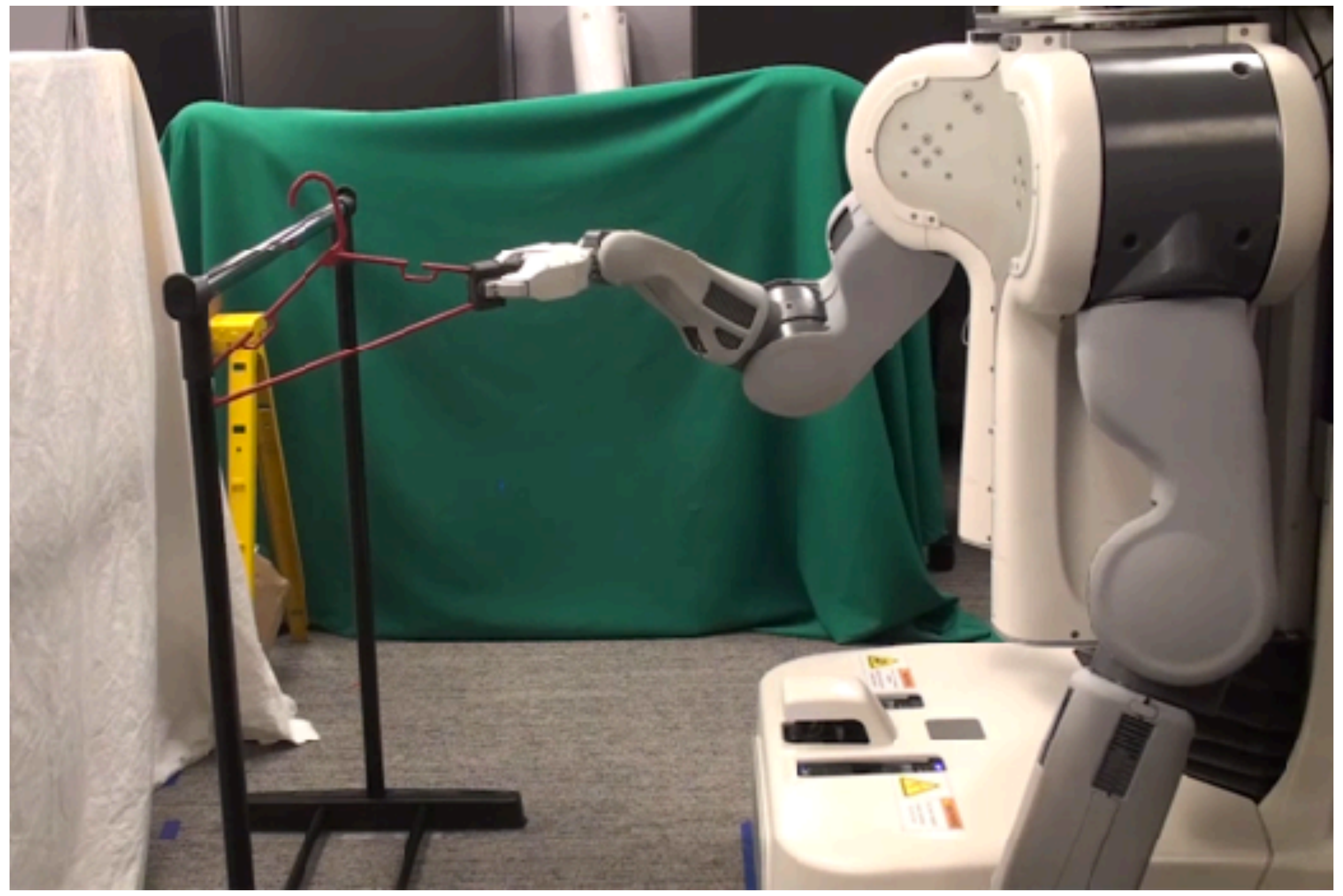
Rocky Duan
OpenAI / UC Berkeley

Recent Breakthroughs in AI

[Mnih et al, 2013]



[Silver et al, 2016]

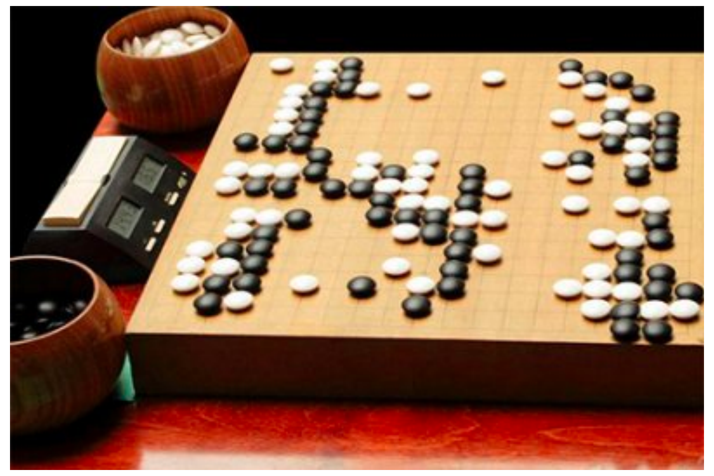
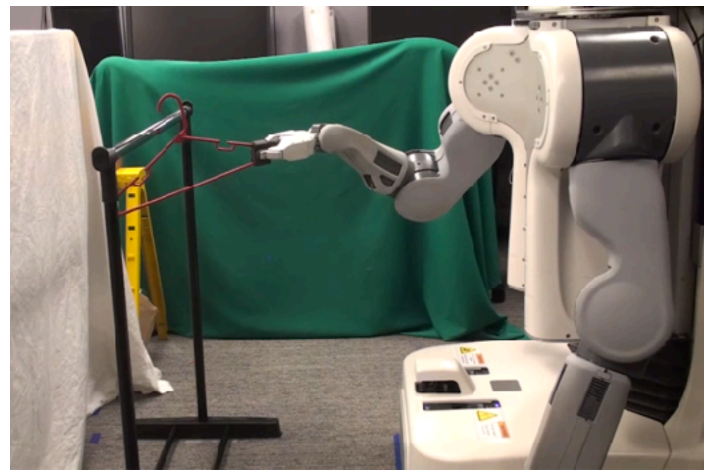


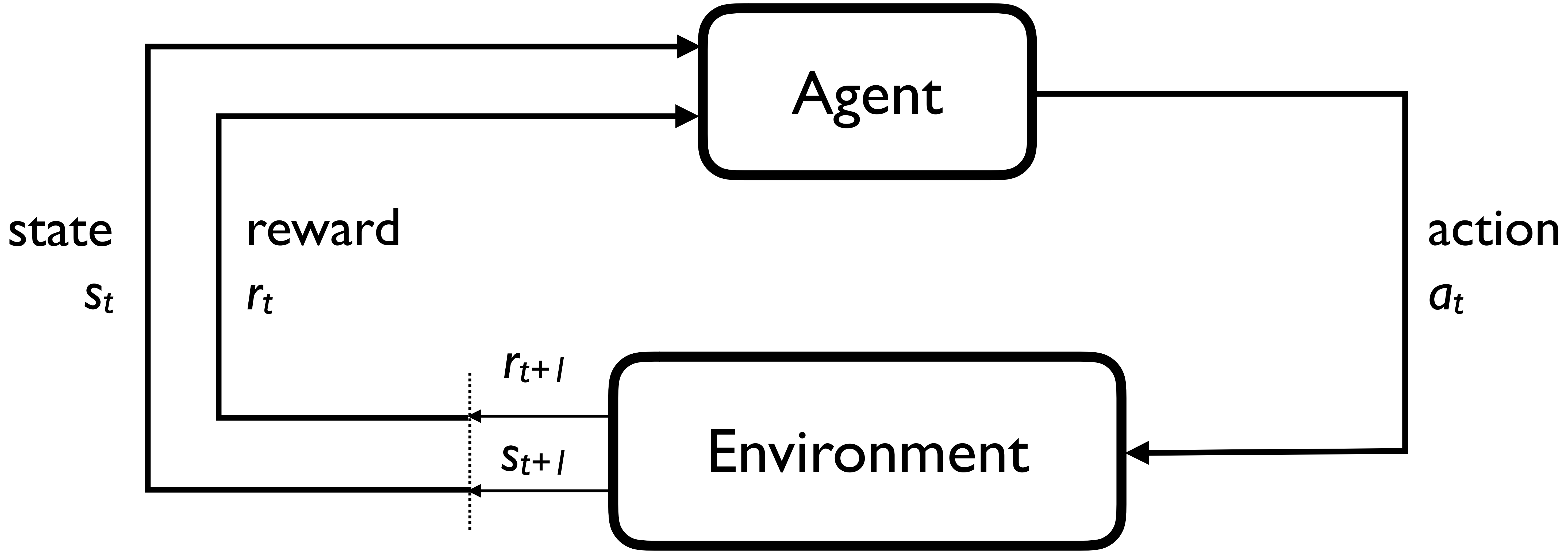
[Levine et al, 2015; Finn et al, 2016]

[Google]

[Rocky Duan, OpenAI / UC Berkeley]

Reinforcement Learning

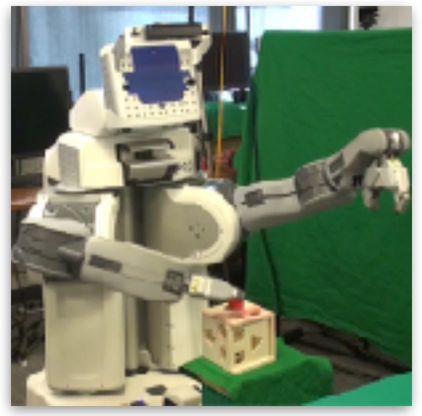



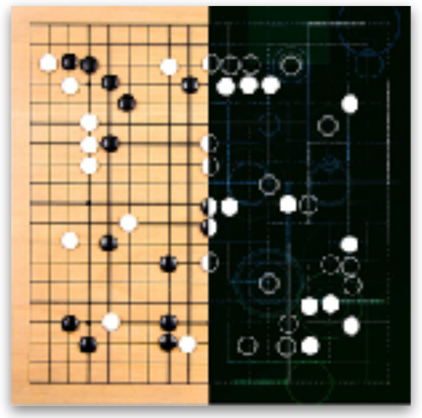
Goal: maximize expected reward

[Rocky Duan, OpenAI / UC Berkeley]

Outline

- Basics of Reinforcement Learning
- Model-Free RL
  - Value-Based Methods
  - Policy-Based Methods
- Model-Based RL
  - Guided Policy Search
  - AlphaGo





[Rocky Duan, OpenAI / UC Berkeley]

Outline

- Basics of Reinforcement Learning
- Model-Free RL
  - Value-Based Methods

  - Policy-Based Methods

- Model-Based RL
  - Guided Policy Search

  - AlphaGo

[Rocky Duan, OpenAI / UC Berkeley]

Markov Decision Processes (MDPs)

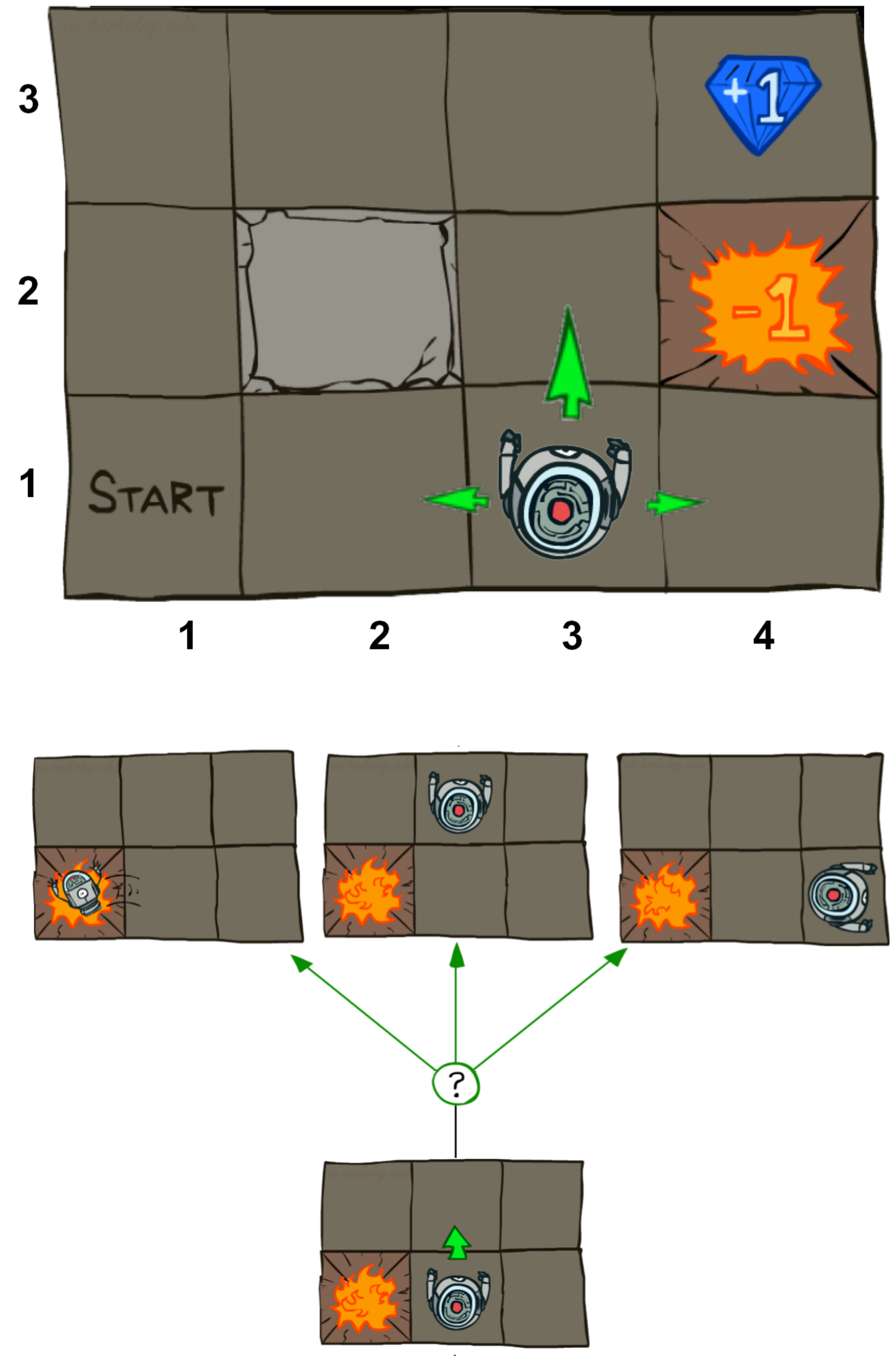
Markov property: the agent's future is independent of its past history conditioned on the current state.

[Rocky Duan, OpenAI / UC Berkeley]

Markov Decision Processes (MDPs)

An MDP is defined by:

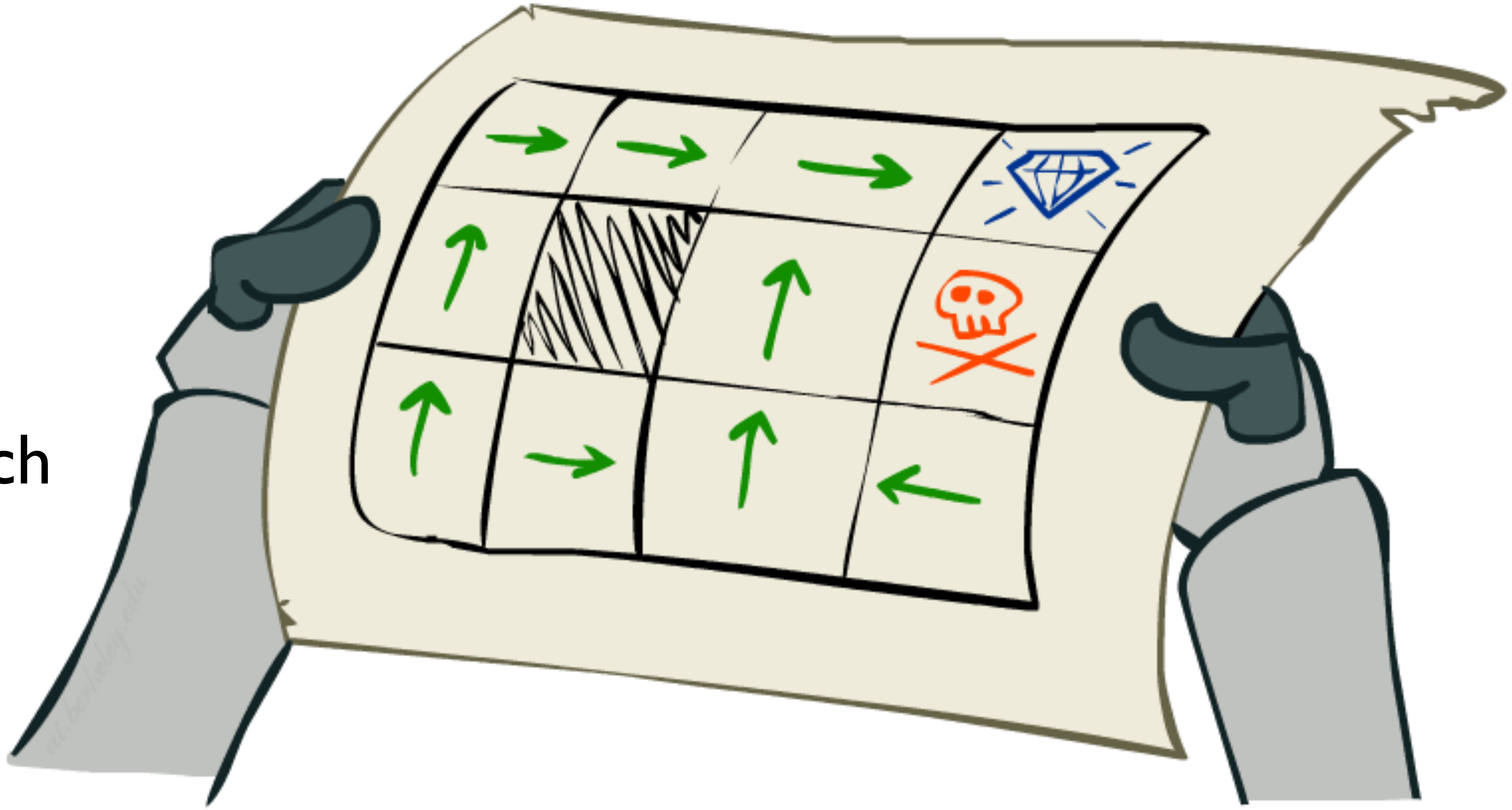
- A set of states $s \in \mathcal{S}$
- A set of actions $a \in \mathcal{A}$
- A transition function $P(s'|s,a)$
- A reward function $R(s,a,s')$
- A start state s_0
- Maybe a terminal state



[Image credit: CS188 at Berkeley]

Policies

A policy is a mapping $\pi : \mathcal{S} \to \mathcal{A}$ specifying what action to take at each state.



[Image credit: CS188 at Berkeley]

Utility and Discounting

- Utility of a trajectory $\tau := (s_0, a_0, r_0, s_1, a_1, r_1, \ldots)$
- It's reasonable to maximize the sum of rewards: $U(\tau) = \sum_t r_t$
- It's also reasonable to prefer rewards now to rewards later
- One solution: value of rewards decay exponentially $U(\tau) = \sum_t \gamma^t r_t$

$\gamma \in (0, 1]$

1

Worth Now

γ

Worth Next Step

γ^2

Worth In Two Steps

[Image credit: CS188 at Berkeley]

Values of States

V*(s) = expected utility starting in s and acting optimally

Bellman Equation:

$$V^*(s) = \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V^*(s'))$$

Value Iteration:

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

[Rocky Duan, OpenAI / UC Berkeley]

Value Iteration

$$V_0(s) \leftarrow 0$$

k = 0

[Image credit: CS188 at Berkeley]

Noise = 0.2
Discount = 0.9

[Rocky Duan, OpenAI / UC Berkeley]

Value Iteration

$$V_1(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_0(s'))$$

k = 0

Noise = 0.2
Discount = 0.9

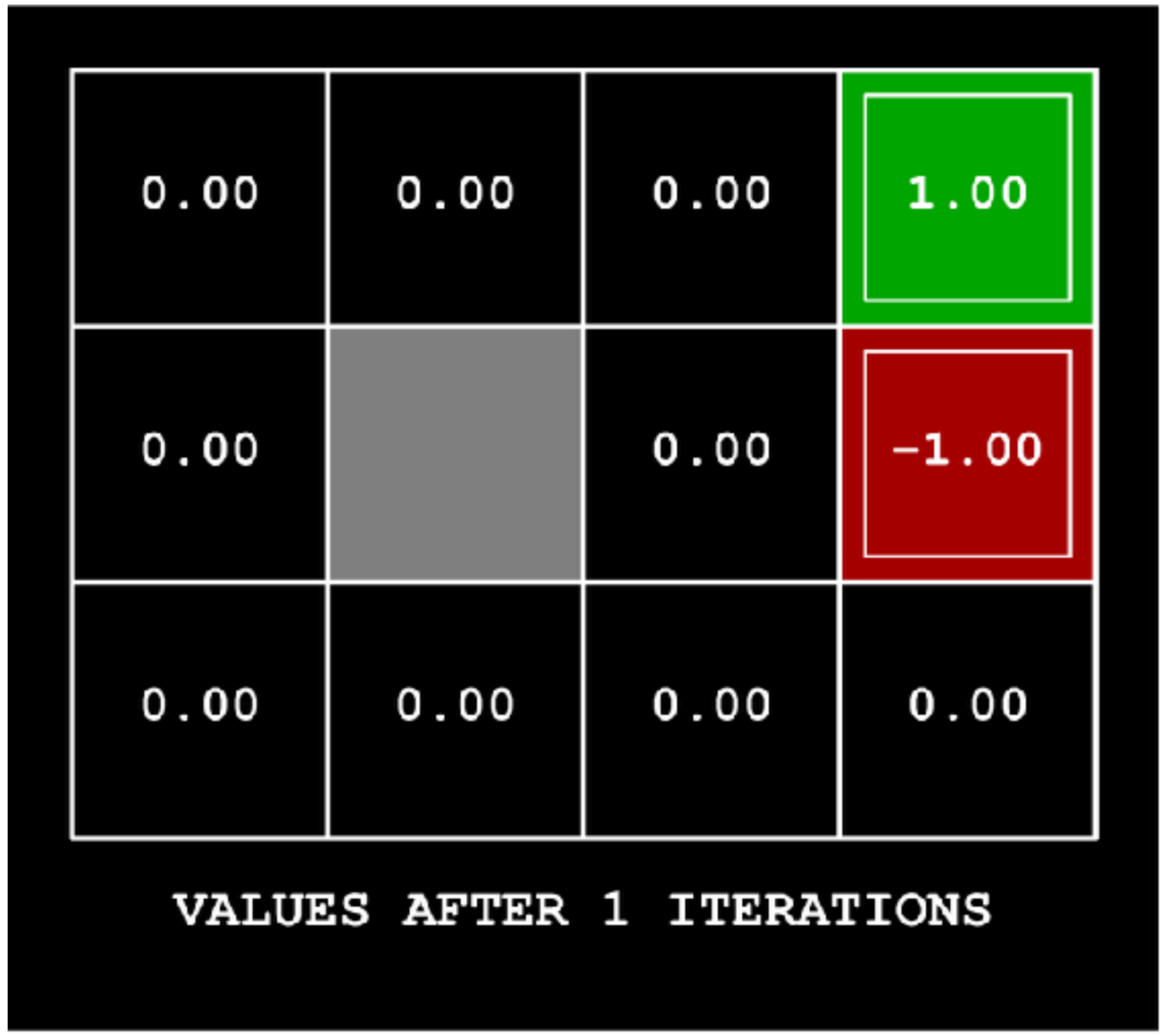
[Image credit: CS188 at Berkeley]

[Rocky Duan, OpenAI / UC Berkeley]

Value Iteration

$$V_1(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_0(s'))$$

k = 1



[Image credit: CS188 at Berkeley]

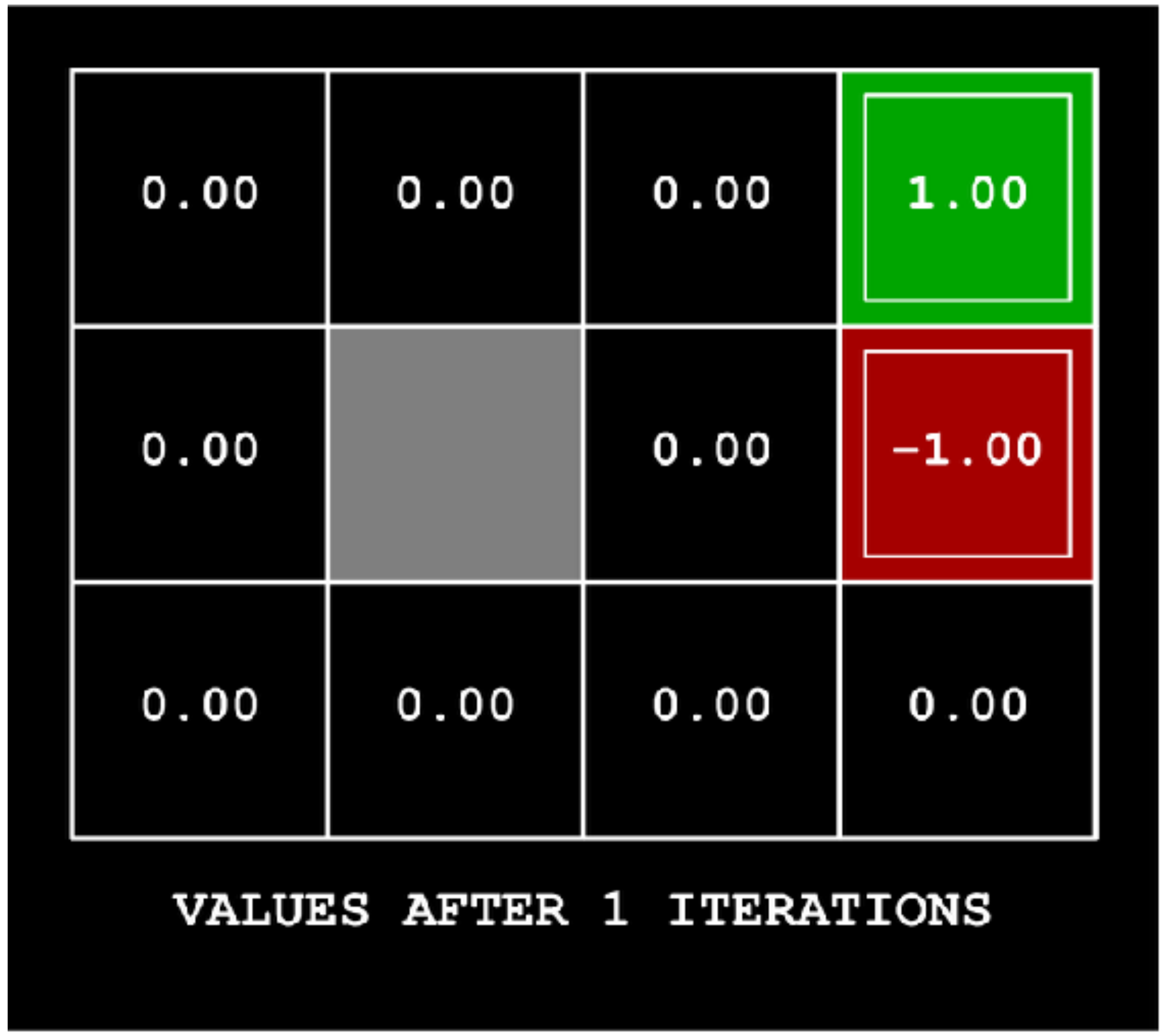
Noise = 0.2
Discount = 0.9

[Rocky Duan, OpenAI / UC Berkeley]

Value Iteration

$$V_2(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_1(s'))$$

k = 1



[Image credit: CS188 at Berkeley]

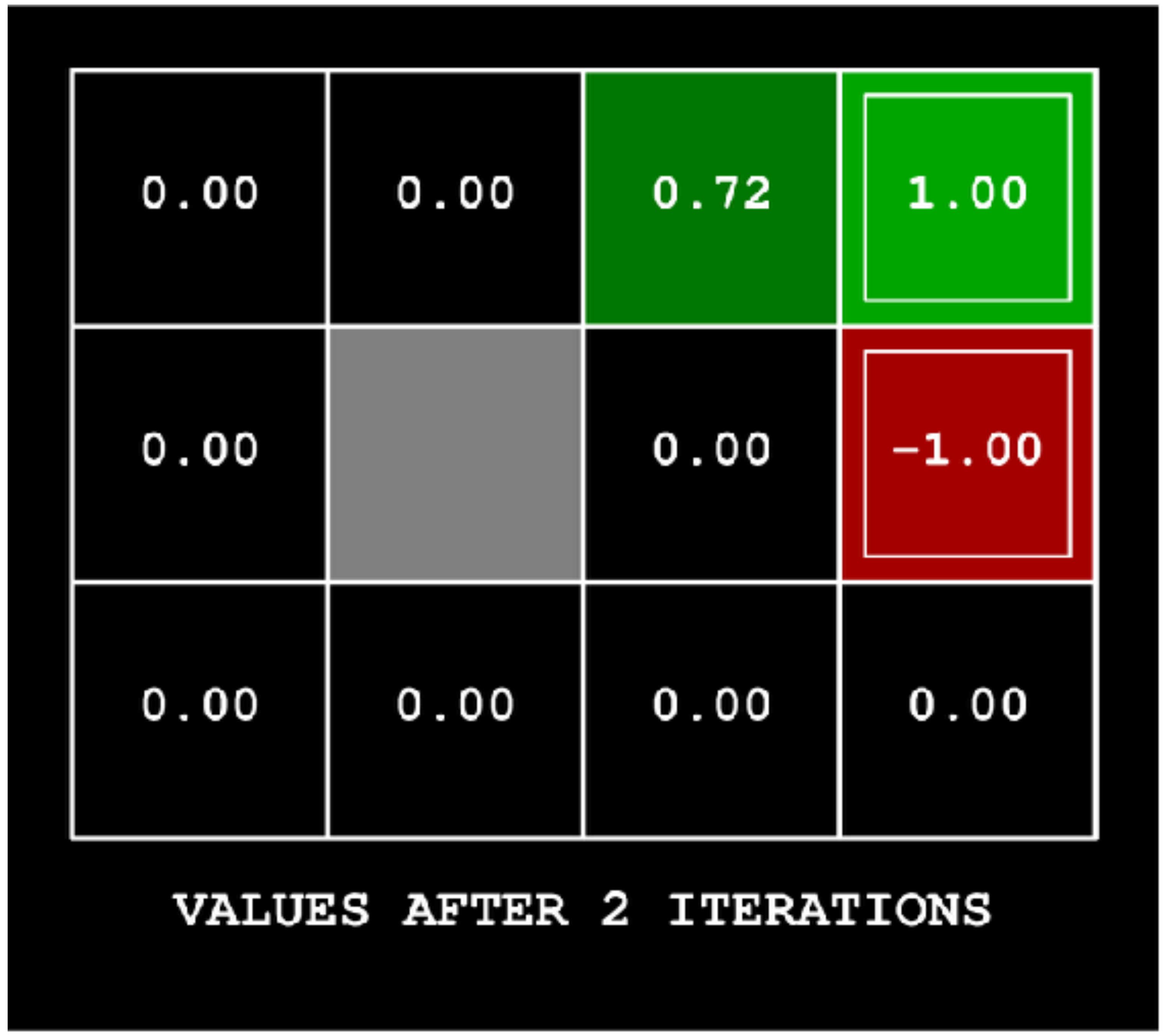
Noise = 0.2
Discount = 0.9

[Rocky Duan, OpenAI / UC Berkeley]

Value Iteration

$$V_2(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_1(s'))$$

k = 2



[Image credit: CS188 at Berkeley]

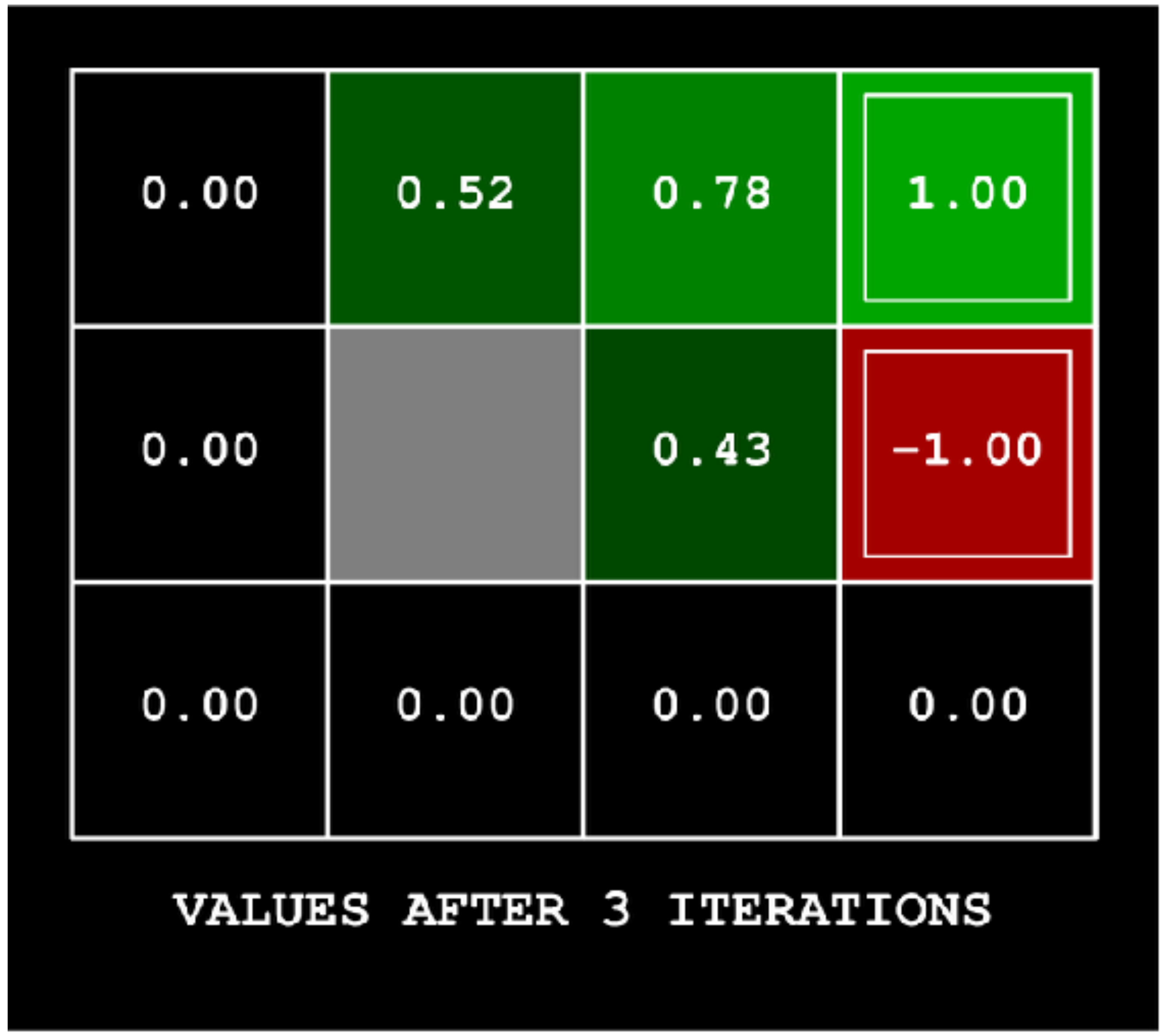
Noise = 0.2
Discount = 0.9

[Rocky Duan, OpenAI / UC Berkeley]

Value Iteration

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 3



[Image credit: CS188 at Berkeley]

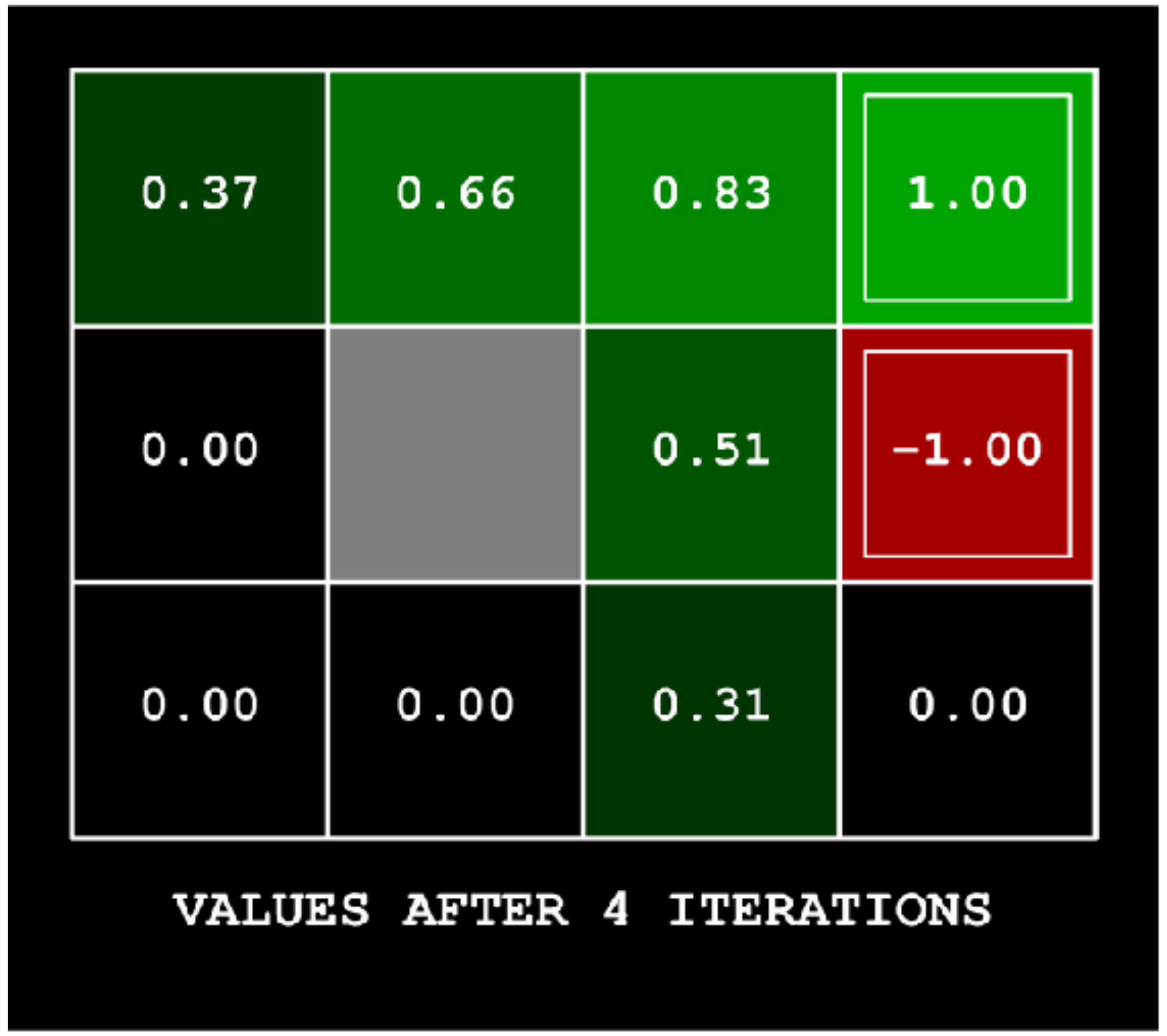
Noise = 0.2
Discount = 0.9

[Rocky Duan, OpenAI / UC Berkeley]

Value Iteration

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 4



[Image credit: CS188 at Berkeley]

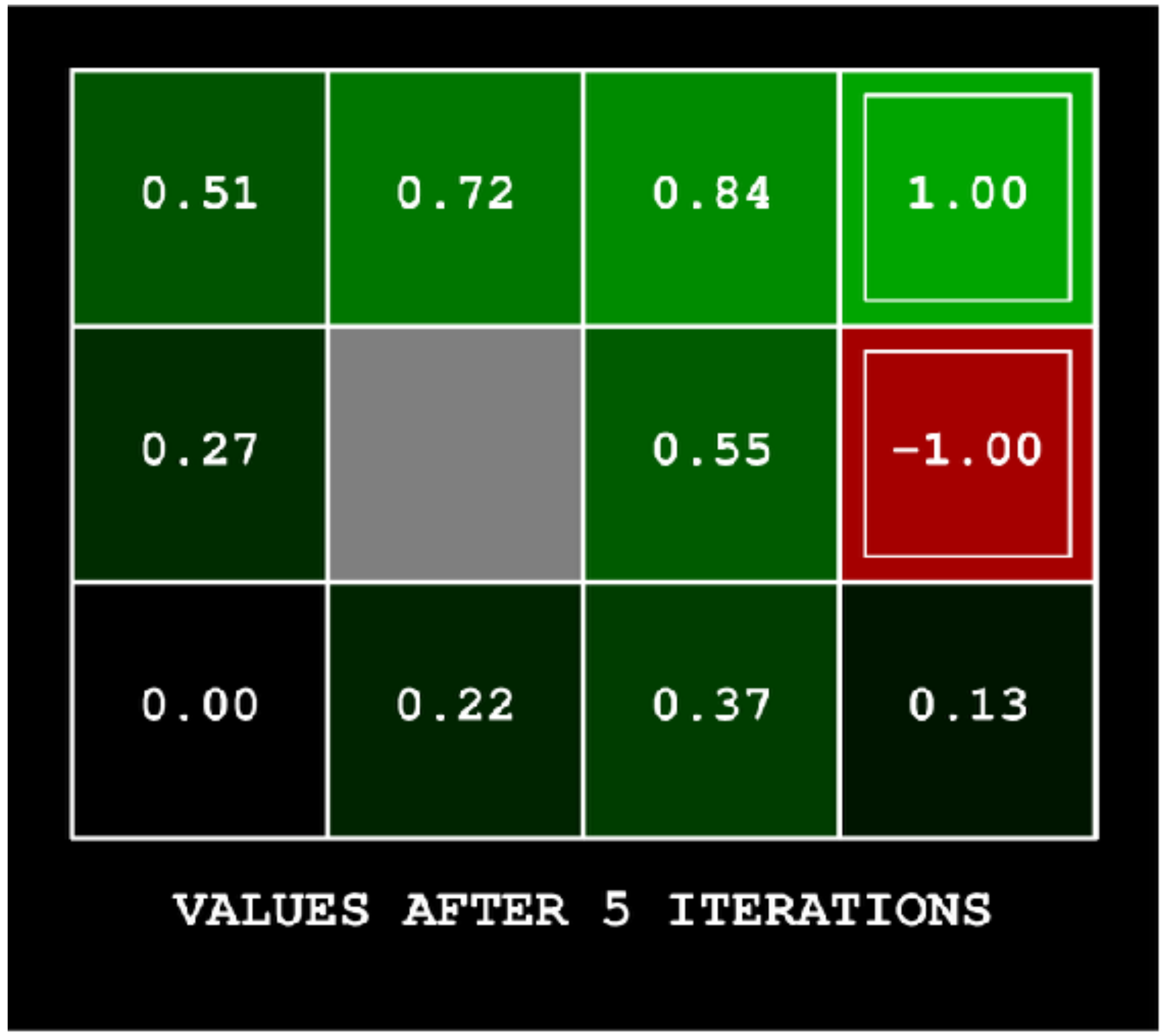
Noise = 0.2
Discount = 0.9

[Rocky Duan, OpenAI / UC Berkeley]

Value Iteration

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 5



[Image credit: CS188 at Berkeley]

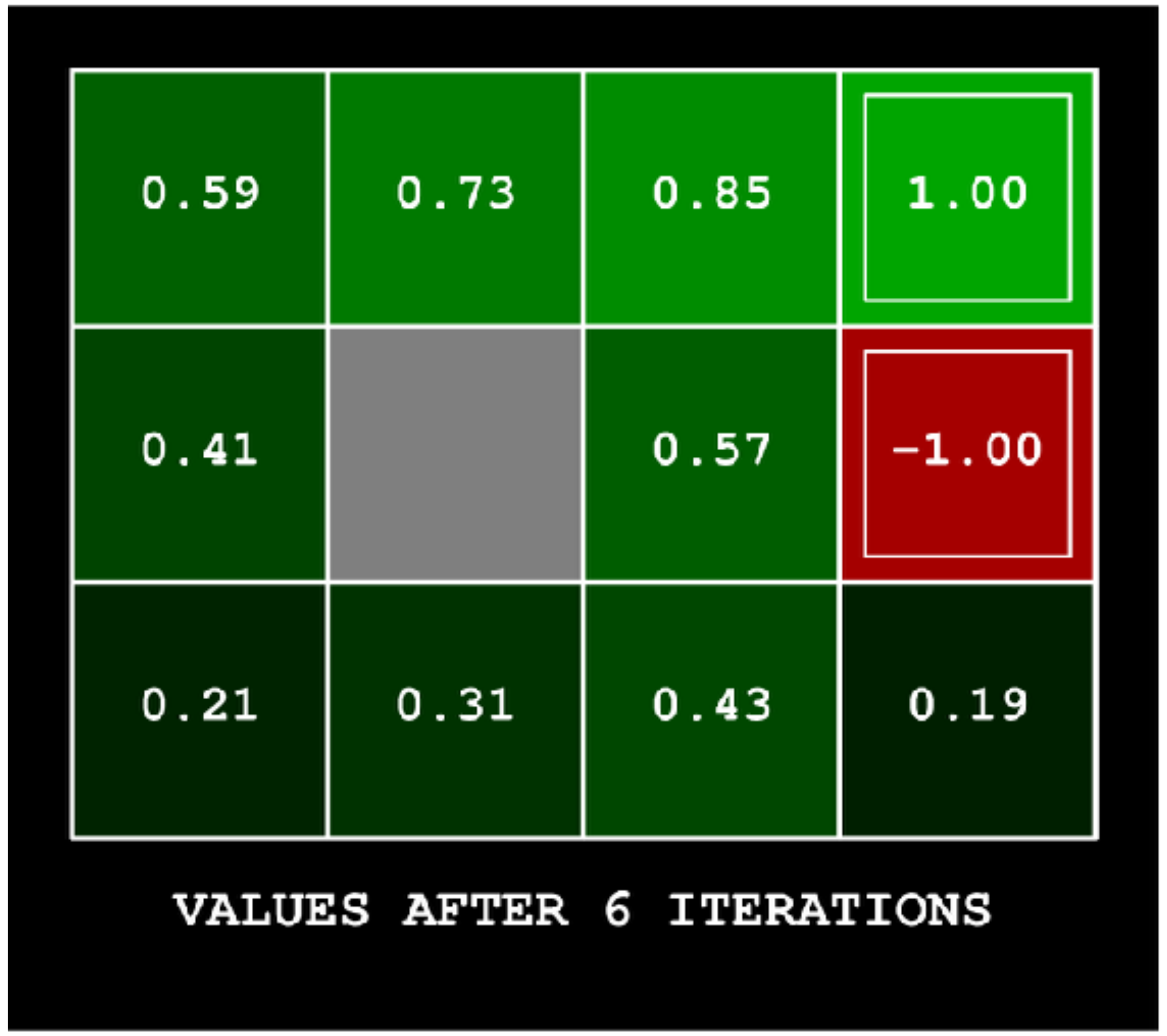
Noise = 0.2
Discount = 0.9

[Rocky Duan, OpenAI / UC Berkeley]

Value Iteration

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 6



Noise = 0.2
Discount = 0.9

[Image credit: CS188 at Berkeley]

[Rocky Duan, OpenAI / UC Berkeley]

Value Iteration

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 7

[Image credit: CS188 at Berkeley]

Noise = 0.2
Discount = 0.9

[Rocky Duan, OpenAI / UC Berkeley]

Value Iteration

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 8

Noise = 0.2
Discount = 0.9

[Image credit: CS188 at Berkeley]

[Rocky Duan, OpenAI / UC Berkeley]

Value Iteration

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 9

[Image credit: CS188 at Berkeley]

Noise = 0.2
Discount = 0.9

[Rocky Duan, OpenAI / UC Berkeley]

Value Iteration

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 10

[Image credit: CS188 at Berkeley]

Noise = 0.2
Discount = 0.9

[Rocky Duan, OpenAI / UC Berkeley]

Value Iteration

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 11

[Image credit: CS188 at Berkeley]

Noise = 0.2
Discount = 0.9

[Rocky Duan, OpenAI / UC Berkeley]

Value Iteration

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 12

Noise = 0.2
Discount = 0.9

[Image credit: CS188 at Berkeley]

[Rocky Duan, OpenAI / UC Berkeley]

Value Iteration

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 100

[Image credit: CS188 at Berkeley]

Noise = 0.2
Discount = 0.9

[Rocky Duan, OpenAI / UC Berkeley]

Q-Values

$Q^*(s, a)$ = expected utility starting in s, taking action a, and (thereafter) acting optimally

Bellman Equation:

$$Q^*(s,a) = \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma \max_{a'} Q^*(s',a'))$$

Q-Value Iteration:

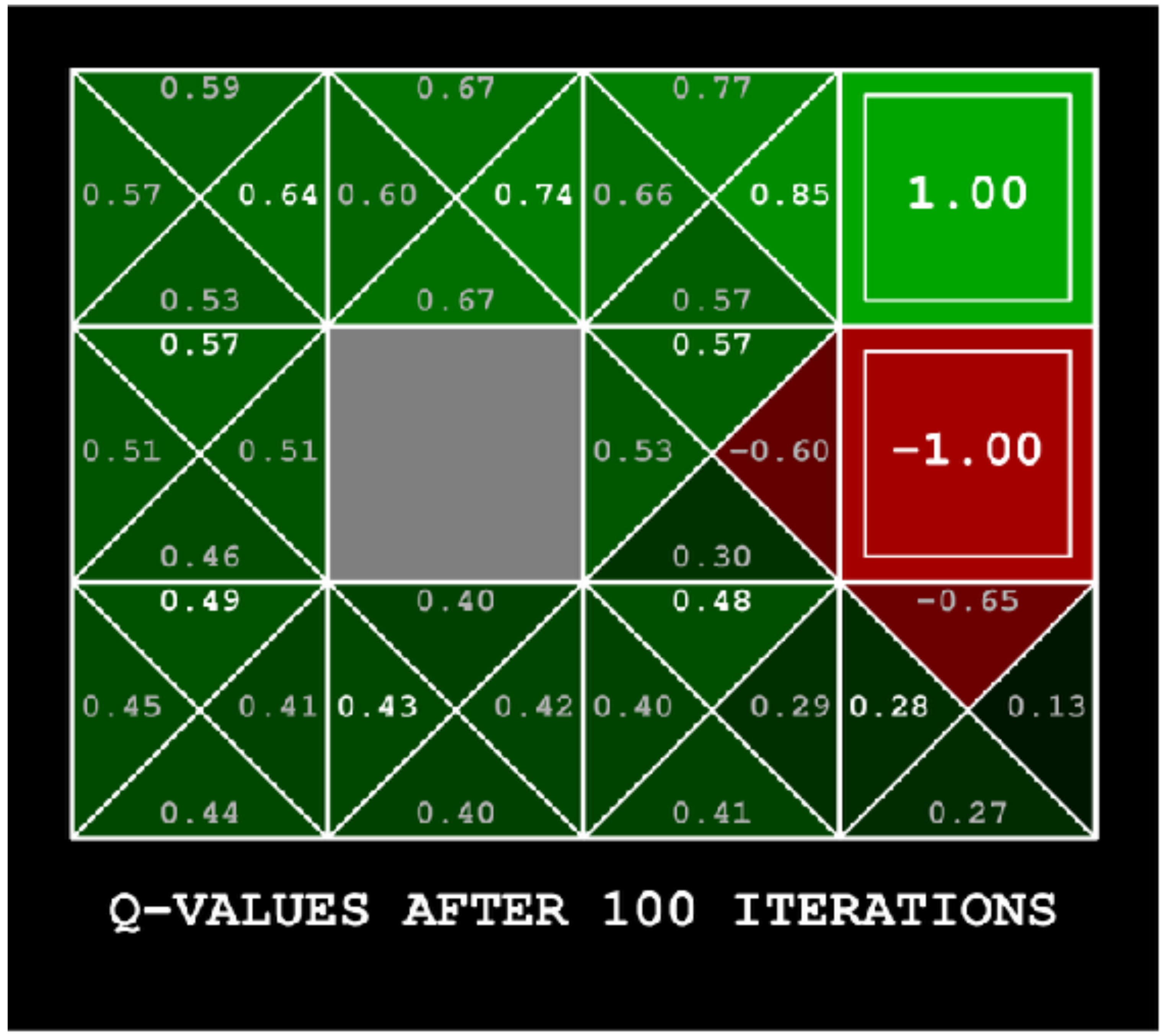
$$Q_{k+1}(s,a) \leftarrow \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma \max_{a'} Q_k(s',a'))$$

[Rocky Duan, OpenAI / UC Berkeley]

Q-Value Iteration

$$Q_{k+1}(s,a) \leftarrow \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma \max_{a'} Q_k(s',a'))$$

k = 100



[Image credit: CS188 at Berkeley]

Noise = 0.2
Discount = 0.9

[Rocky Duan, OpenAI / UC Berkeley]

Outline

- Basics of Reinforcement Learning
- Model-Free RL
    - Value-Based Methods
    - Policy-Based Methods
- Model-Based RL
    - Guided Policy Search
    - AlphaGo

[Rocky Duan, OpenAI / UC Berkeley]

Q-Learning

Want:

$$Q_{k+1}(s,a) \leftarrow \sum_{s'} P(s'|s,a)(R(s,a) + \gamma \max_{a'} Q_k(s',a'))$$

But no access to $P(s'|s,a)$

Q-Learning: Collect samples and learn Q(s, a) values as you go.

- Record sample (s, a, s', r)
- Consider your old estimate: $Q(s,a)$
- Consider your new sample estimate: $sample = r + \gamma \max_{a'} Q(s',a')$
- Incorporate the new estimate into a running average

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + (\alpha)[sample]$$

[Rocky Duan, OpenAI / UC Berkeley]

Q-Learning

Theorem: Q-Learning converges to the optimal value, as long as you visit each state-action pair infinitely often.

DQN

Problem: For very large state spaces, cannot store Q-values explicitly

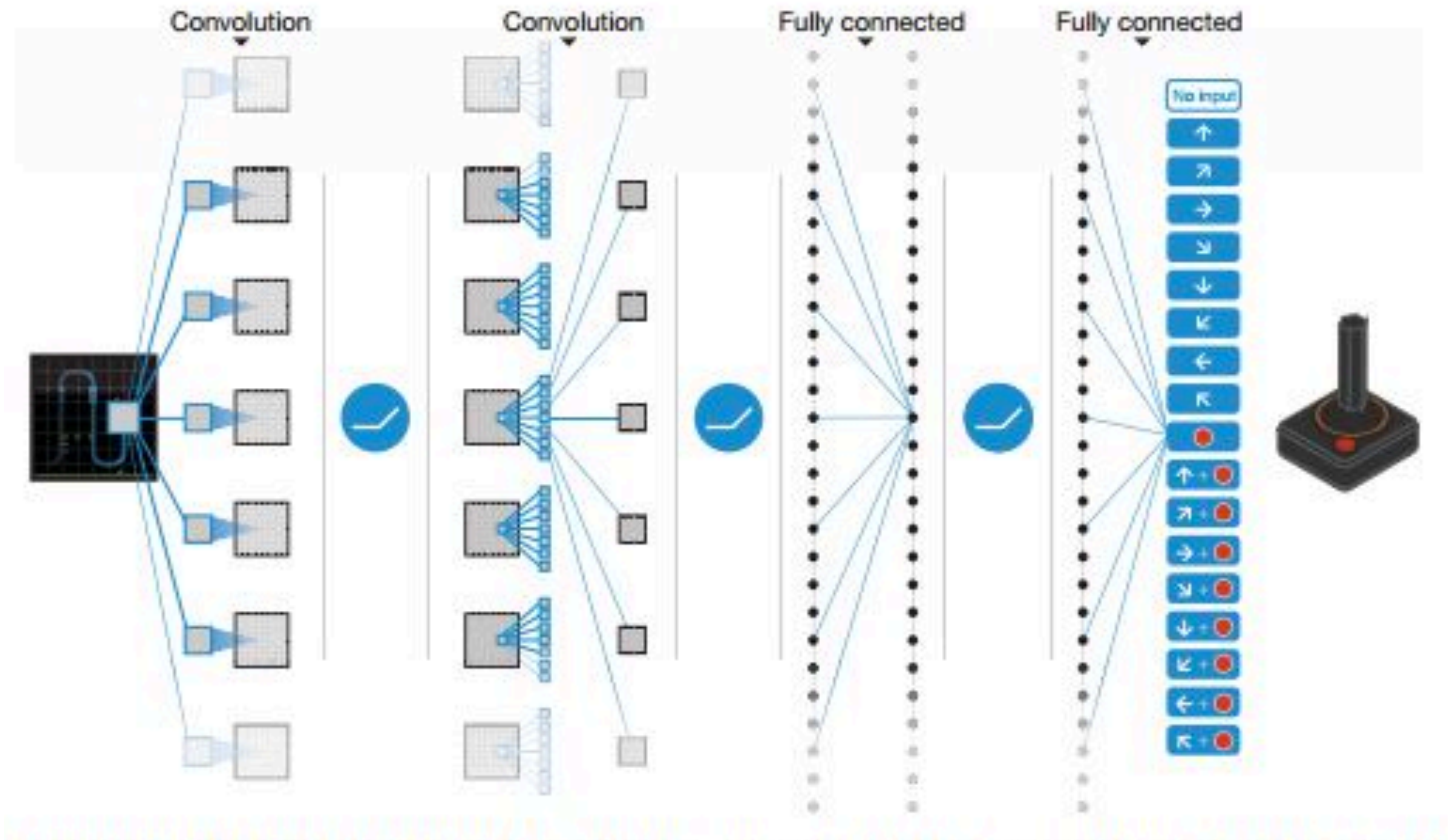
Solution: Use function approximation (e.g. deep neural networks!)

- Receive a sample (s, a, s', r)
- Consider your old estimate: $Q_\theta(s, a)$
- Consider your new sample estimate:

$$sample = r + \gamma \max_{a'} Q_\theta(s', a')$$

- Perform *gradient descent* on squared error

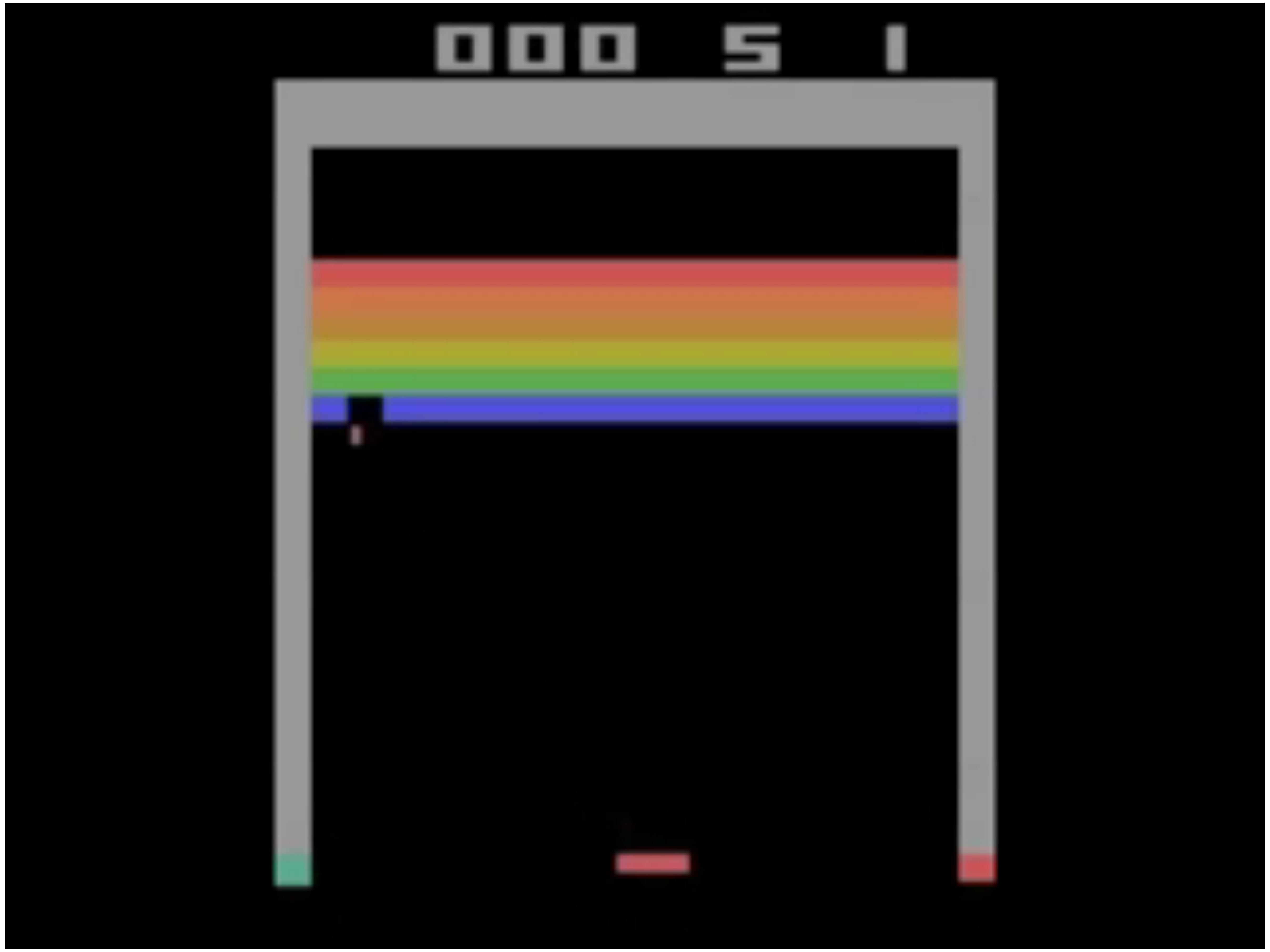
$$\theta \leftarrow \theta - \alpha \nabla_\theta \left(Q_\theta(s, a) - sample\right)^2$$



[Image credit: Mnih et al, 2015]

[Rocky Duan, OpenAI / UC Berkeley]

DQN: Success Stories



[Mnih et al, NIPS 2013 / Nature 2015]

[Rocky Duan, OpenAI / UC Berkeley]

DQN: Success Stories

[Mnih et al, NIPS 2013 / Nature 2015]

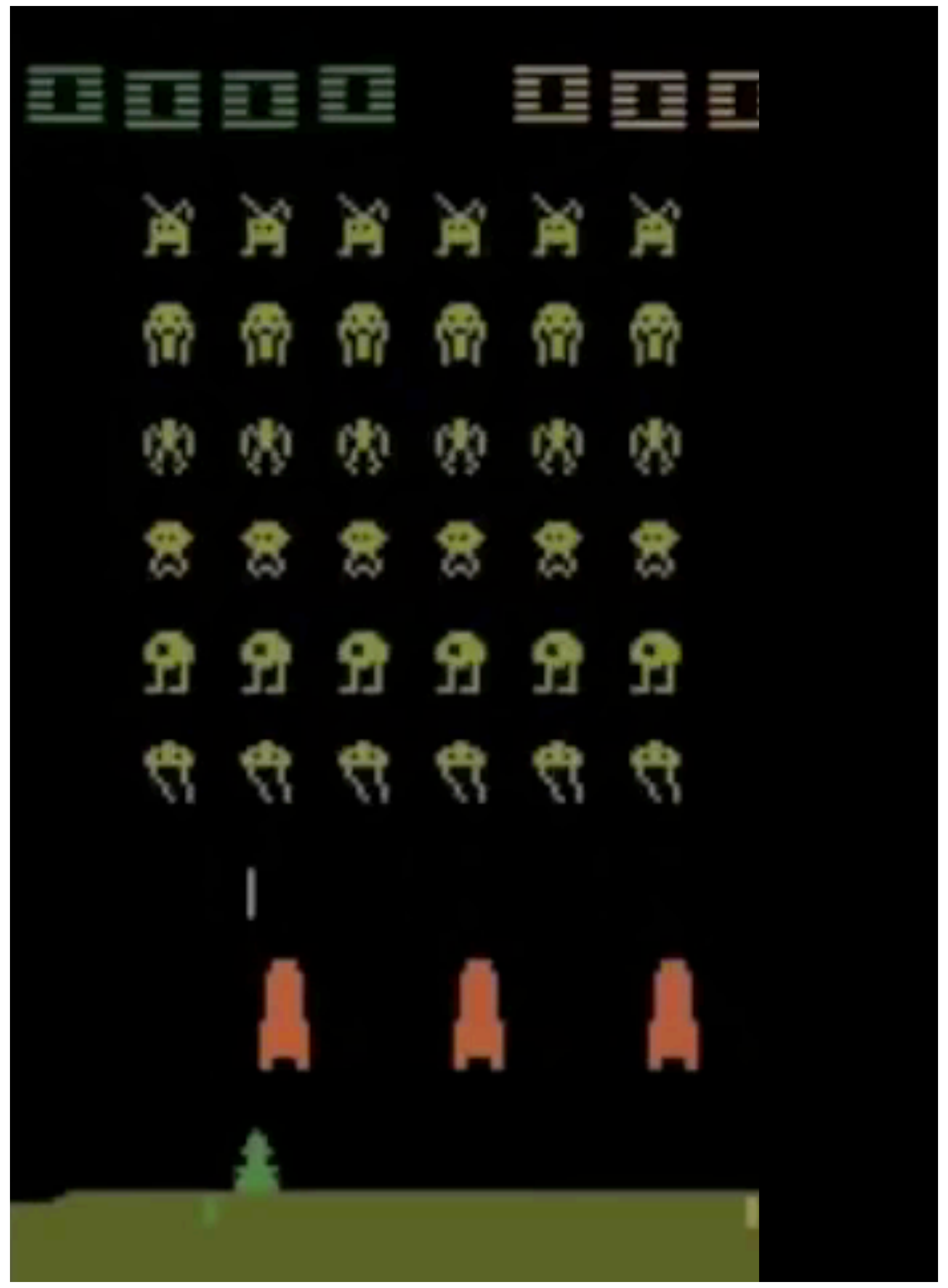
[Rocky Duan, OpenAI / UC Berkeley]

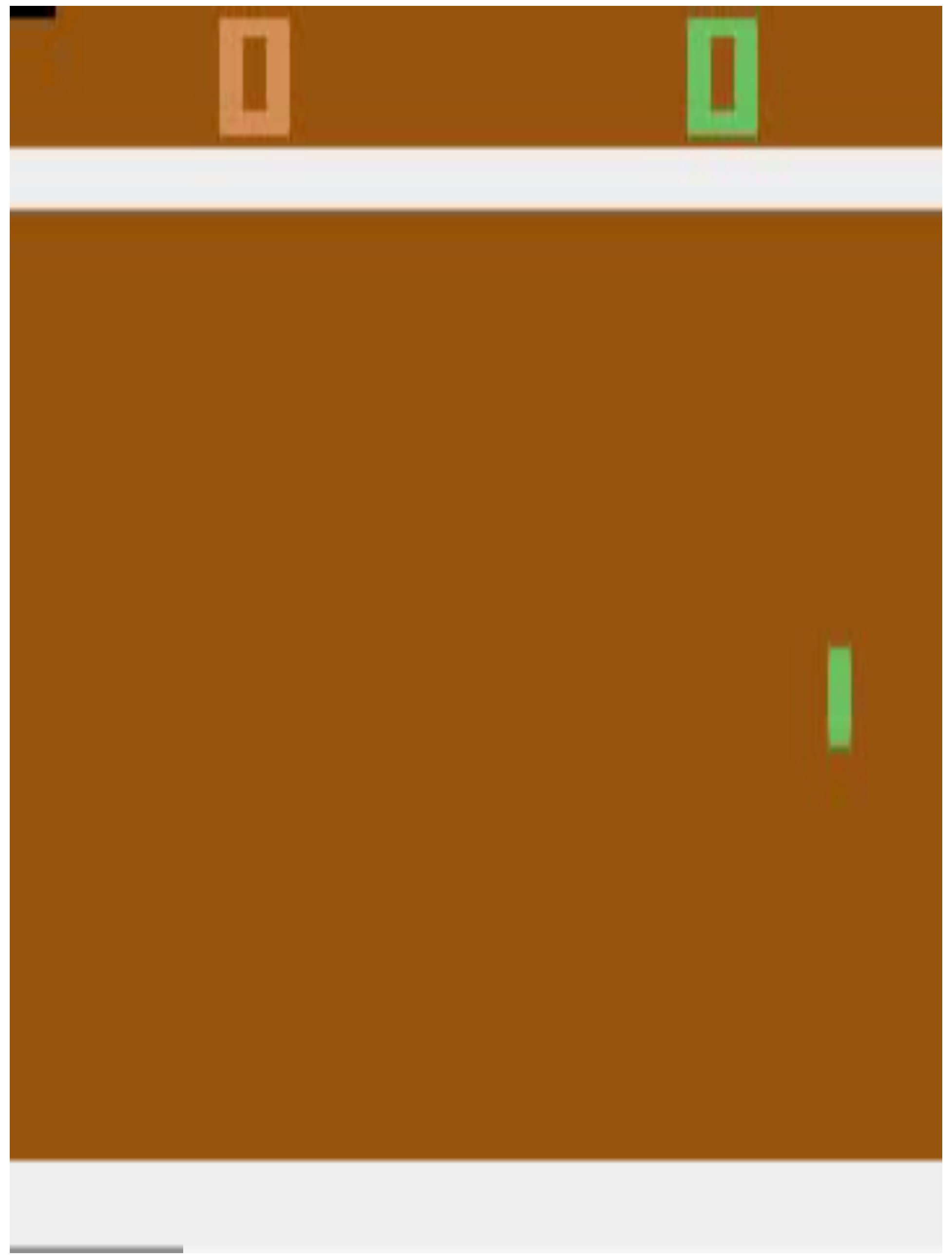
DQN: Success Stories

[Mnih et al, NIPS 2013 / Nature 2015]

[Rocky Duan, OpenAI / UC Berkeley]

DQN: Success Stories



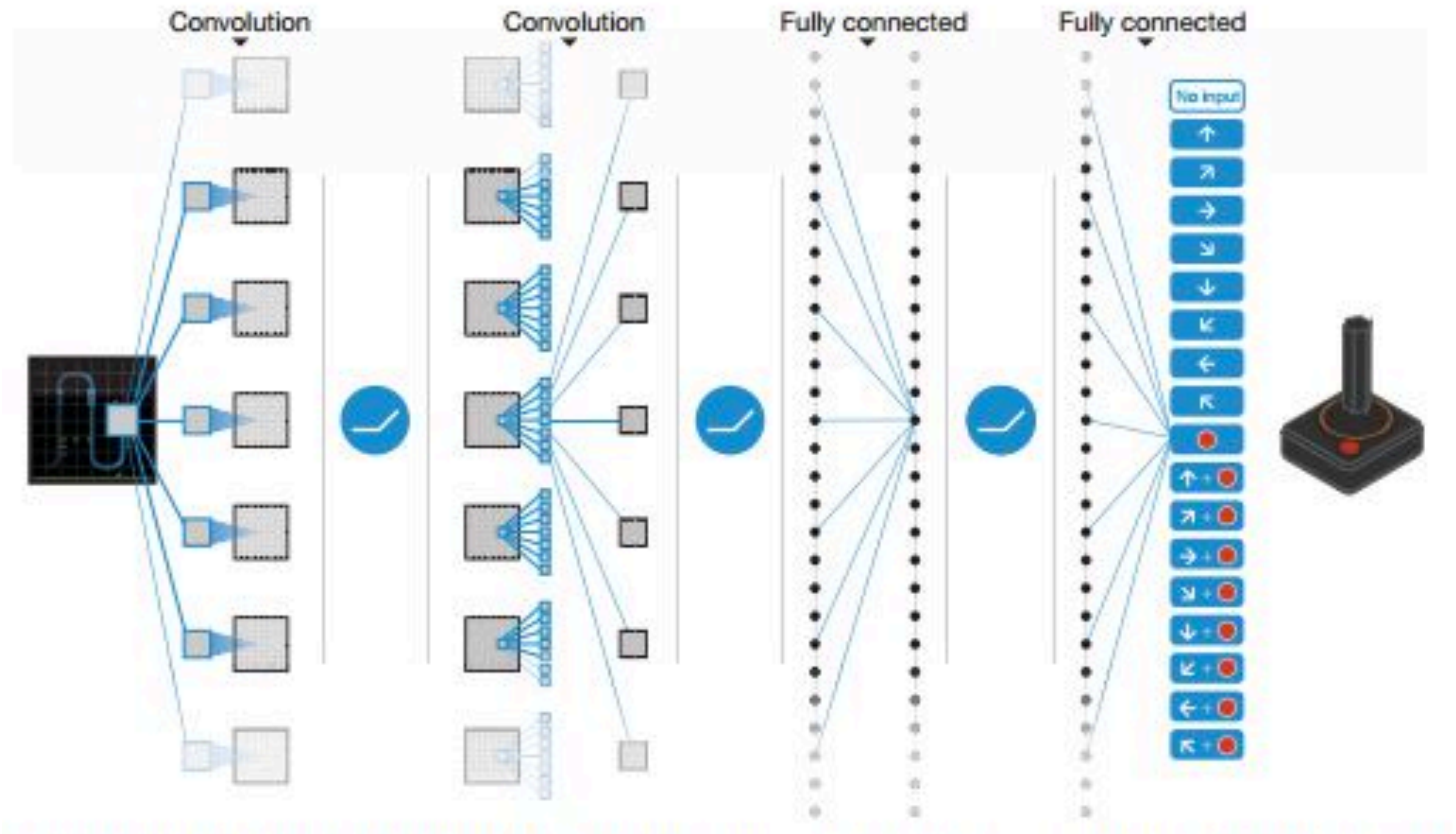


[Mnih et al, NIPS 2013 / Nature 2015]

[Rocky Duan, OpenAI / UC Berkeley]

Caveats

- No (known) guarantees of performance improvement when using function approximation
- Hard to work with continuous actions since we can't have an output for every possible action



[Rocky Duan, OpenAI / UC Berkeley]

Outline

- Basics of Reinforcement Learning
- Model-Free RL
  - Value-Based Methods

  - Policy-Based Methods

- Model-Based RL
  - Guided Policy Search

  - AlphaGo

[Rocky Duan, OpenAI / UC Berkeley]

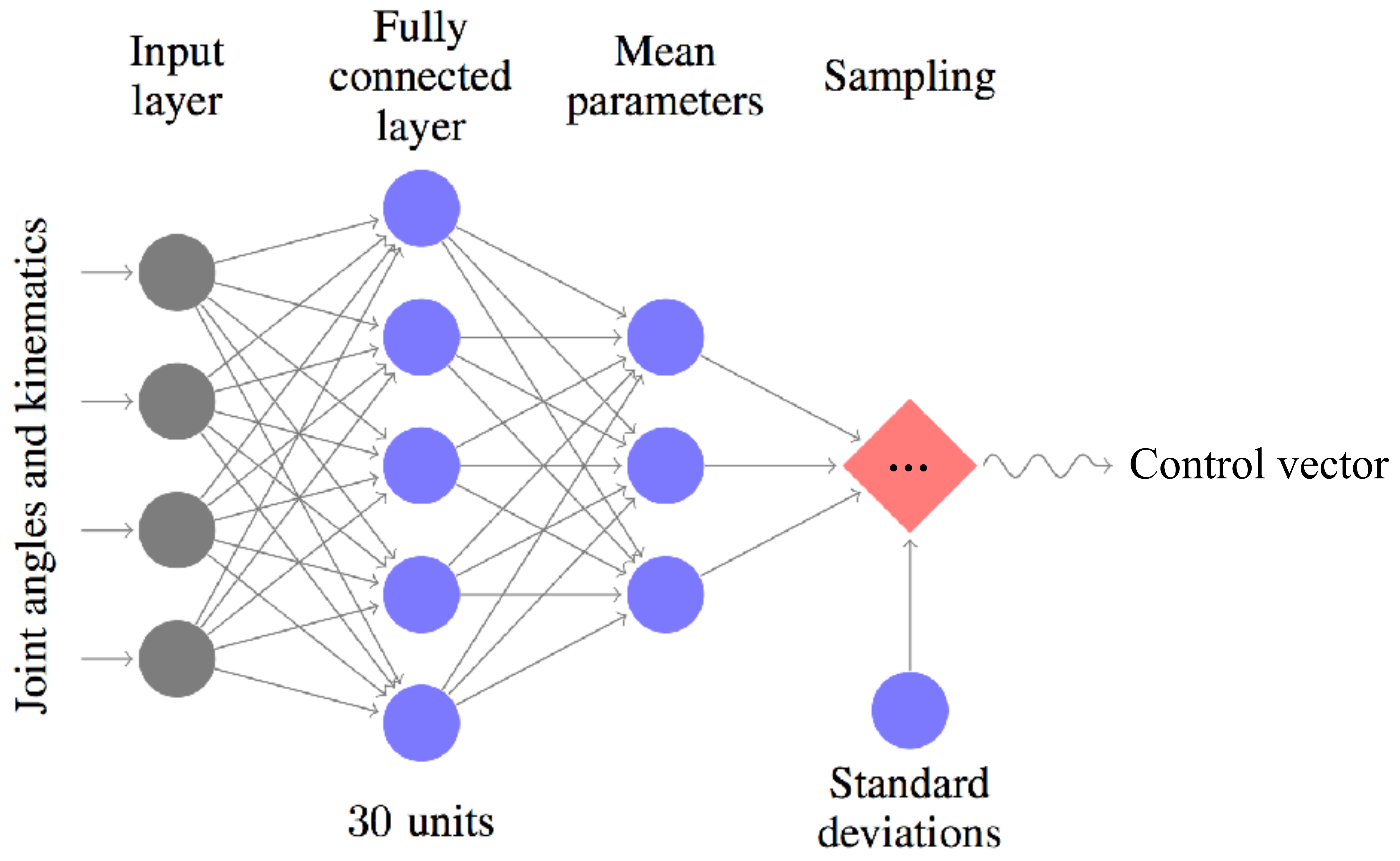
Policy Optimization

Objective: $\max_{\theta} U(\theta)$ where $U(\theta) := \mathbb{E}_{\tau|\pi_\theta}[U(\tau)] = \mathbb{E}_{\pi_\theta}[\sum_t \gamma^t r_t]$

[Rocky Duan, OpenAI / UC Berkeley]

Policy Gradient

Assume a stochastic policy $\pi_\theta : \mathcal{S} \times \mathcal{A} \to [0, 1]$

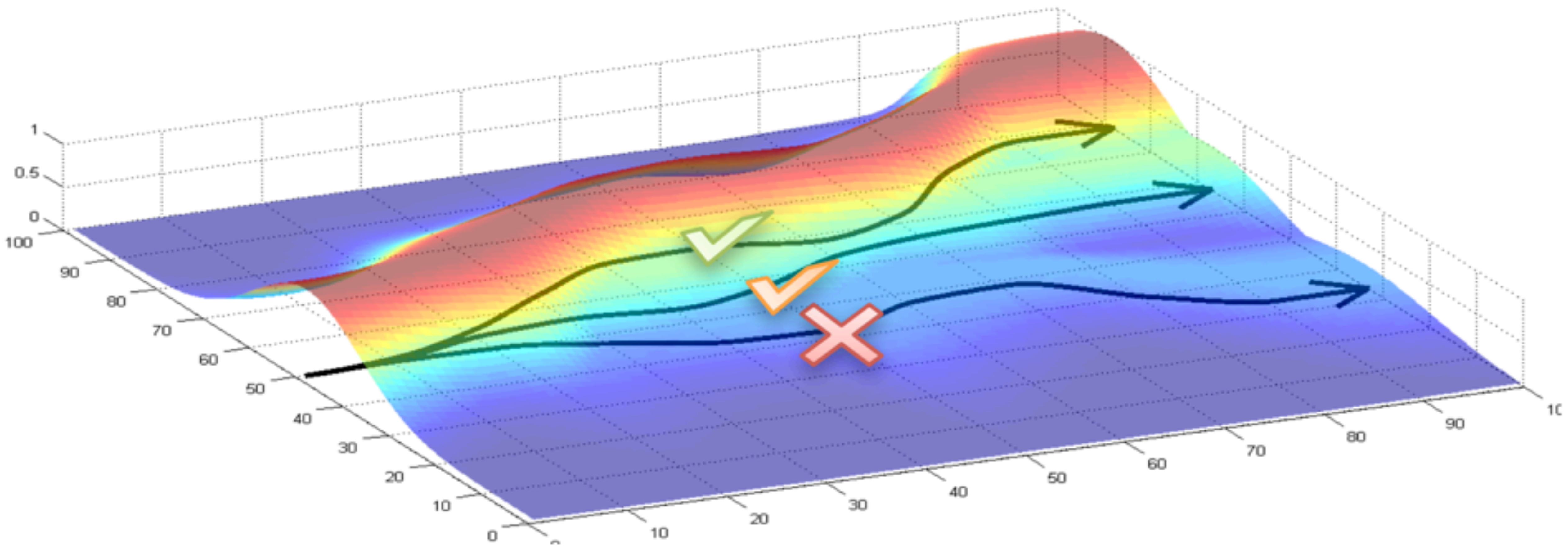


[Image credit: Schulman et al, 2015]

[Rocky Duan, OpenAI / UC Berkeley]

Policy Gradient

$$\nabla_\theta U(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t^i | s_t^i) \sum_{t'=t}^{T} \gamma^{t'} r_{t'}^i$$



[Rocky Duan, OpenAI / UC Berkeley]

Policy Gradient

Improving efficiency using baselines:

$$\nabla_\theta U(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t^i | s_t^i) \left(\sum_{t'=t}^{T} \gamma^{t'} r_{t'}^i - V^{\pi_\theta}(s_t^i) \right)$$

$V^{\pi_\theta}(s)$: average performance starting in state s and following policy π_θ

[Rocky Duan, OpenAI / UC Berkeley]

Choosing step size

Given the gradient, how to choose a step size?

- Supervised learning
  - Step too large: next update will correct for it
- Reinforcement Learning
  - Step too large: terrible policy
    - Next mini batch will be collected under this terrible policy!
    - Unclear how to recover

[Rocky Duan, OpenAI / UC Berkeley]

Trust Region Methods

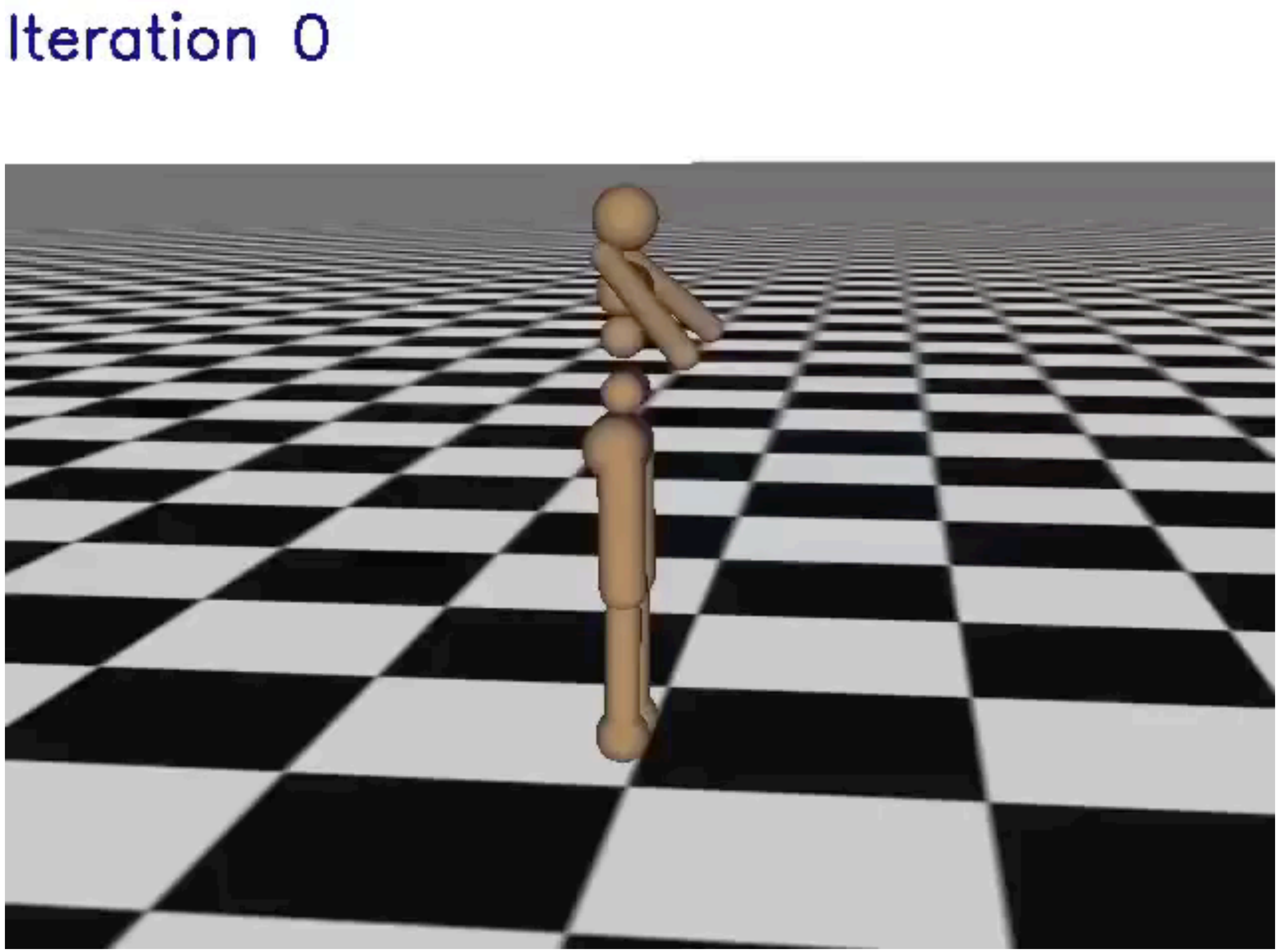
Formulate as constrained optimization problem, controlling how much the policy can change.

Gives rise to natural policy gradient and TRPO

[Kakade 2002; Bagnell & Schneider 2003; Peters & Schaal 2003; Schulman et al 2015]

[Rocky Duan, OpenAI / UC Berkeley]

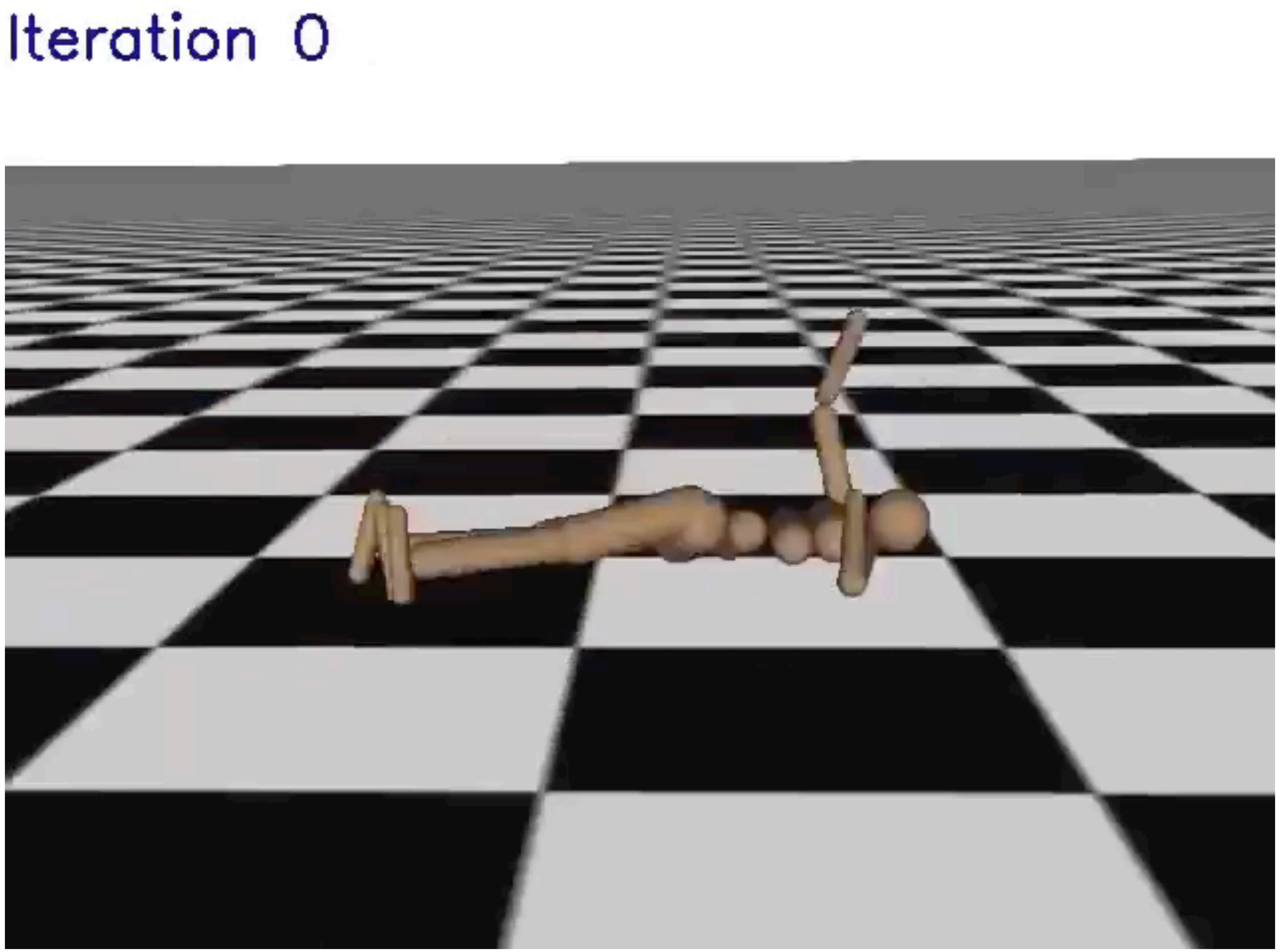
Trust Region Methods: Success Stories



[Schulman et al, 2015]

[Rocky Duan, OpenAI / UC Berkeley]

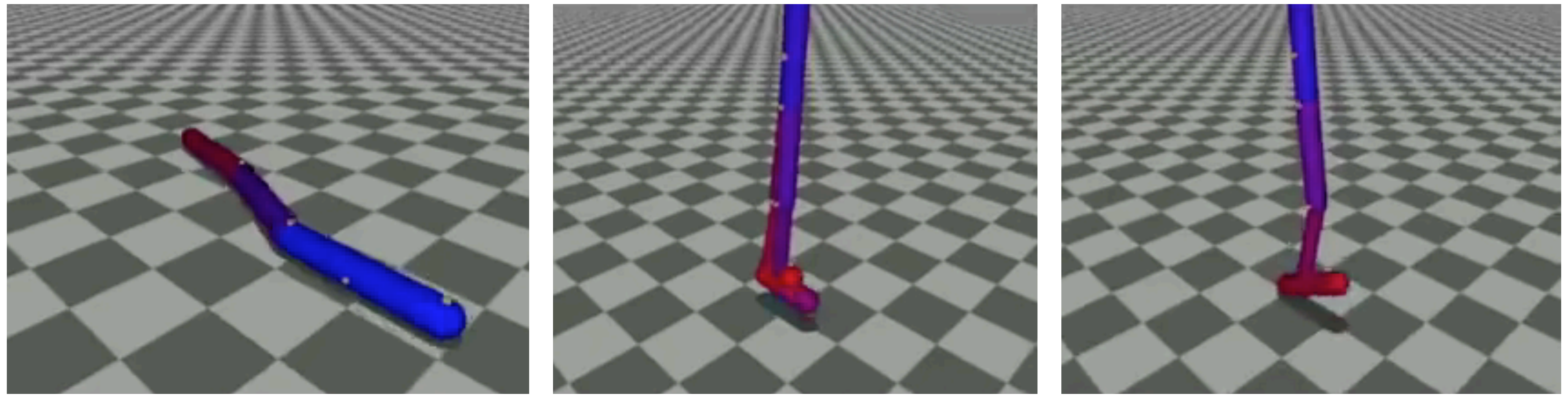
Trust Region Methods: Success Stories

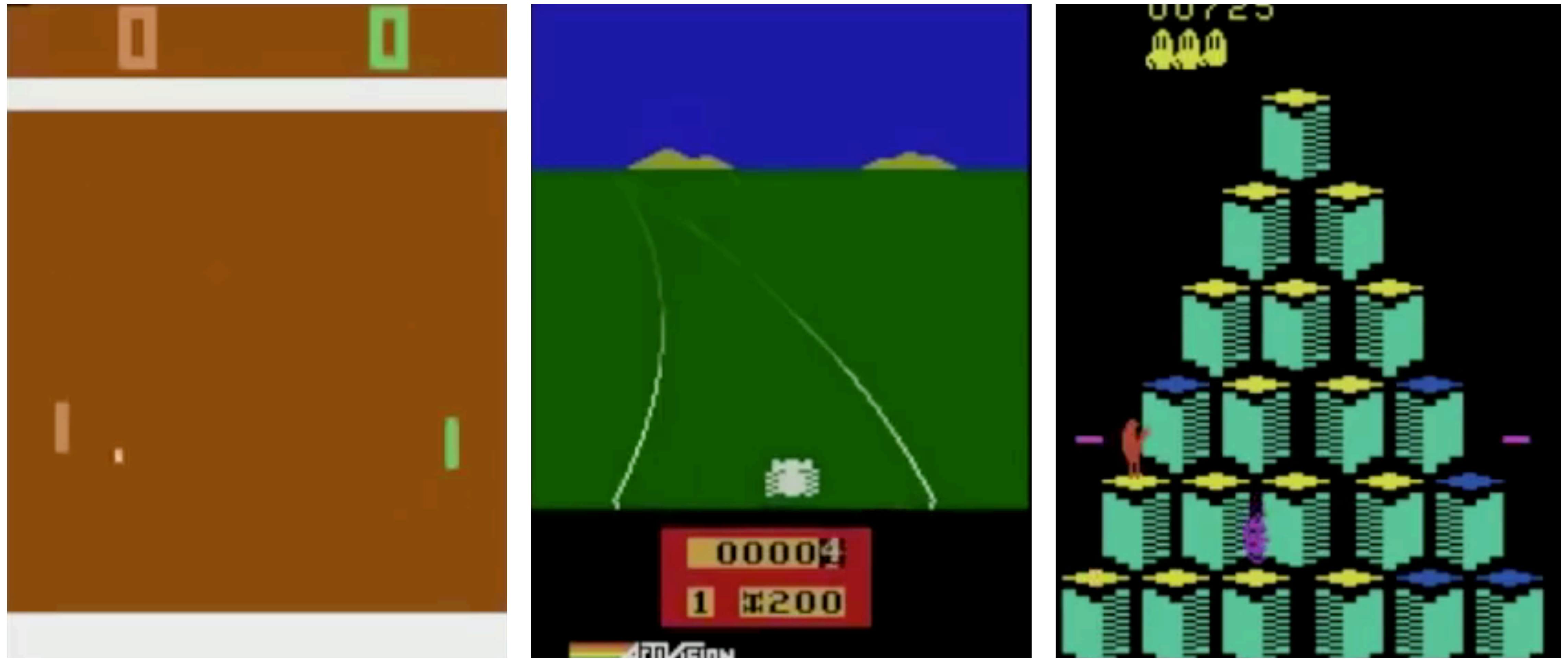


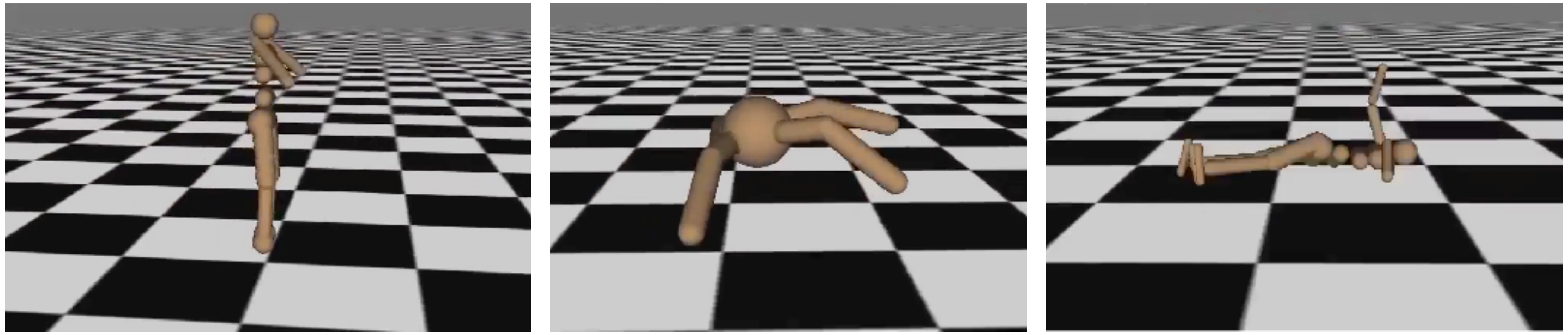
[Schulman et al, 2015]

[Rocky Duan, OpenAI / UC Berkeley]

Trust Region Methods: Success Stories







[Schulman et al, 2015]

[Rocky Duan, OpenAI / UC Berkeley]

Policy Optimization

Objective: $\max_{\theta} U(\theta)$ where $U(\theta) := \mathbb{E}_{\tau|\pi_\theta}[U(\tau)] = \mathbb{E}_{\pi_\theta}[\sum_t \gamma^t r_t]$

Black-Box Optimization?

[Rocky Duan, OpenAI / UC Berkeley]

Cross-Entropy Method

Start with initial parameter distribution, $P_{\mu^{(1)}}(\theta)$ say $\mathcal{N}(0, I)$

for iter i = 1, 2, …

for population member e = 1, 2, …

Sample $\theta^{(e)} \sim P_{\mu^{(i)}}(\theta)$

Collect trajectories under $\pi_{\theta^{(e)}} : \tau_1, \ldots, \tau_N$

Store $(\theta^{(e)}, \hat{U}(\theta^{(e)}))$ where $\hat{U}(\theta^{(e)}) := \frac{1}{N} \sum_{j=1}^{N} U(\tau_j)$

end for

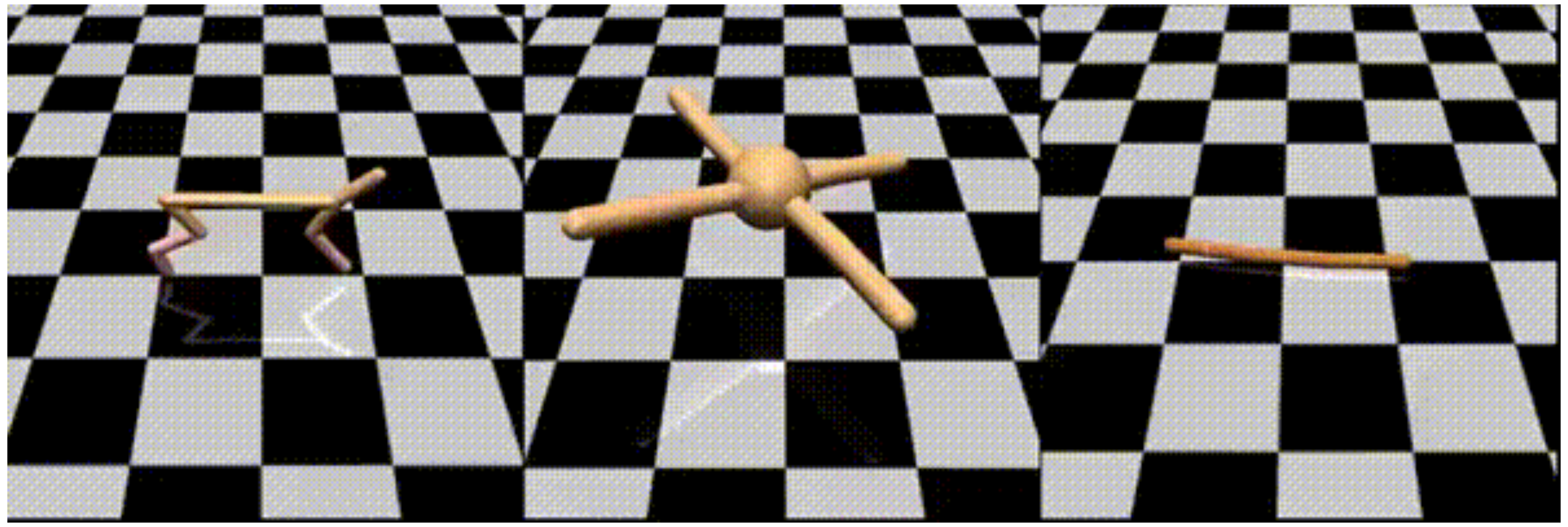
$\mu^{(i+1)} \leftarrow \arg\max_{\mu} \sum_{\bar{e}} \log P_{\mu}(\theta^{\bar{e}})$

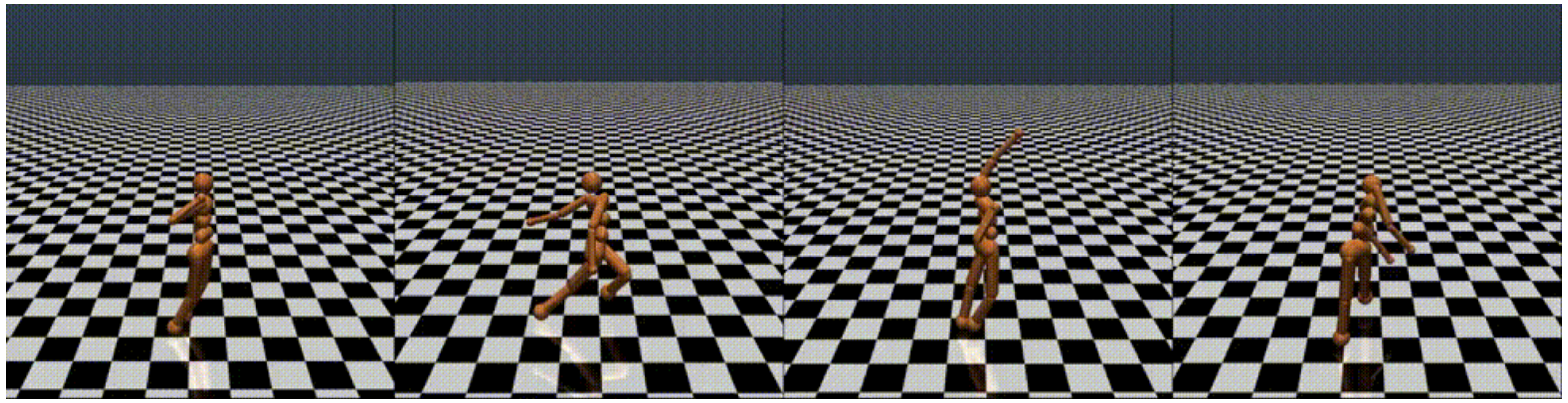
where ē indexes over top p% performance

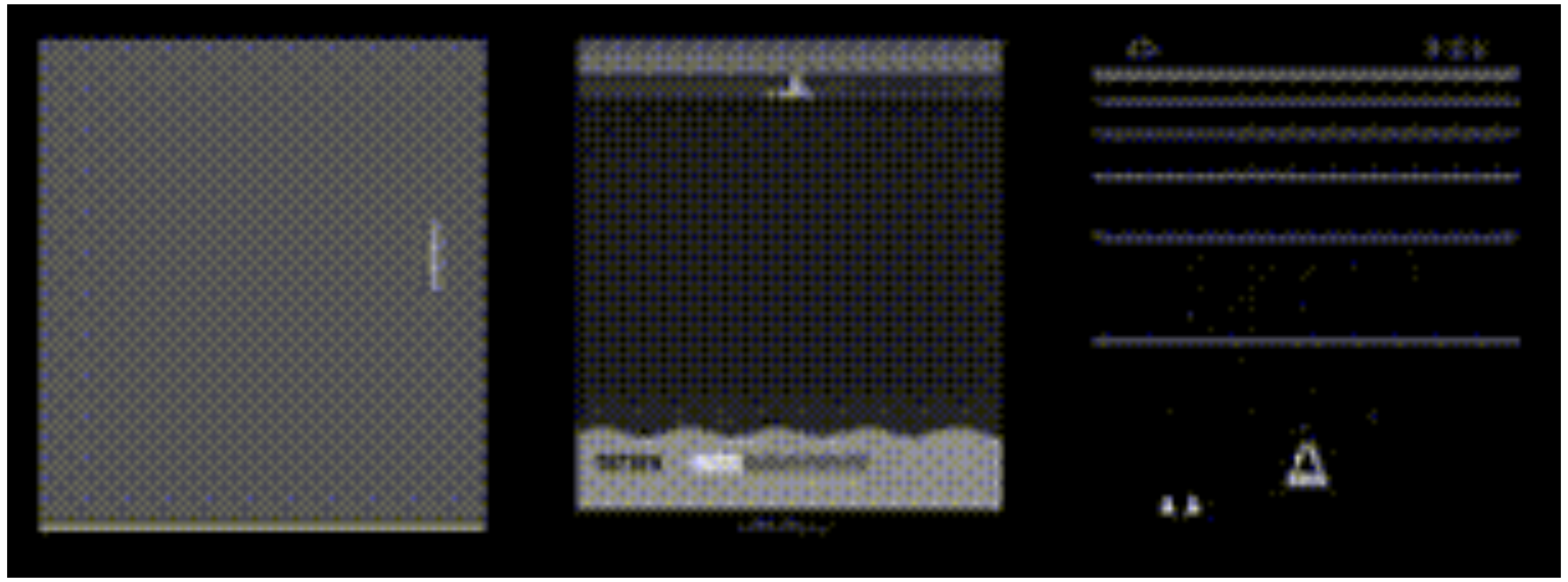
end for

[Rocky Duan, OpenAI / UC Berkeley]

Evolution Strategies: Success Stories







[Salimans et al, 2017]

[Rocky Duan, OpenAI / UC Berkeley]

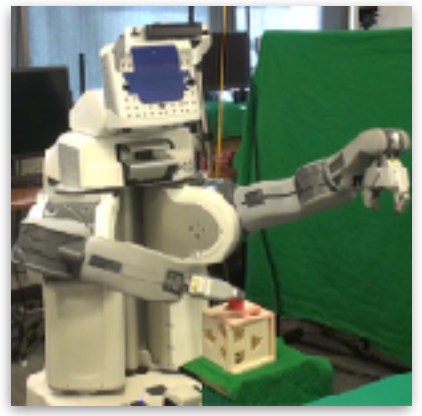
Policy Gradient vs. Evolution Strategies

Policy Gradient	Evolution Strategies
More sample efficient ☺	Less sample efficient ☹
Trickier to parallelize ☹	Easier to parallelize ☺
Requires differentiable policies ☹	Can work with non-differentiable policies ☺

[Rocky Duan, OpenAI / UC Berkeley]

Outline

- Basics of Reinforcement Learning
- Model-Free RL
  - Value-Based Methods
  - Policy-Based Methods
- Model-Based RL
  - Guided Policy Search
  - AlphaGo



[Rocky Duan, OpenAI / UC Berkeley]

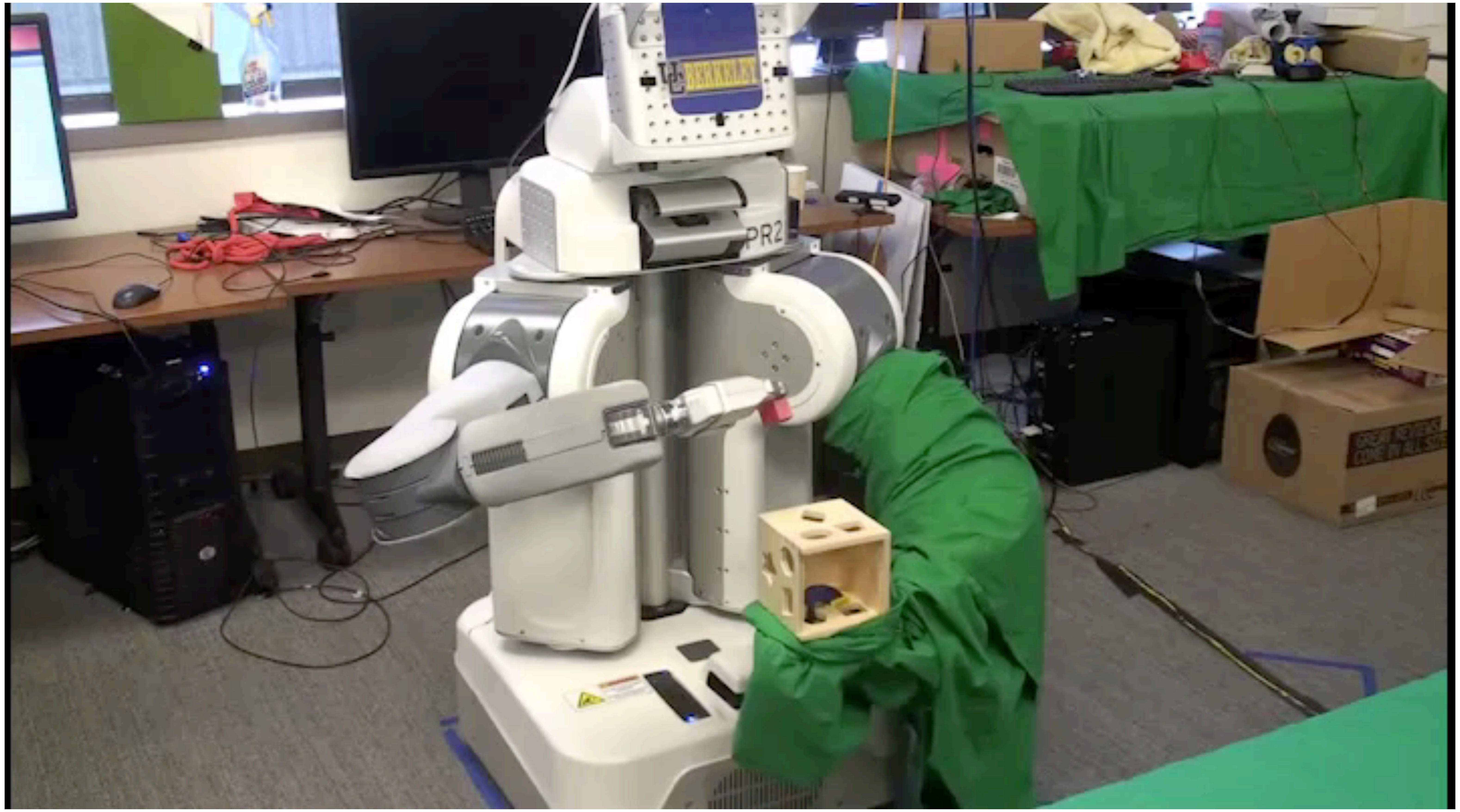
Guided Policy Search

Key idea:

- During training, allow robot to try from the exact same starting state several times
- For such consistent scenarios, iLQR from optimal control theory can be leveraged to help find a solution
- Train a neural network to match the iLQR controllers which generalizes to new situations

[Rocky Duan, OpenAI / UC Berkeley]

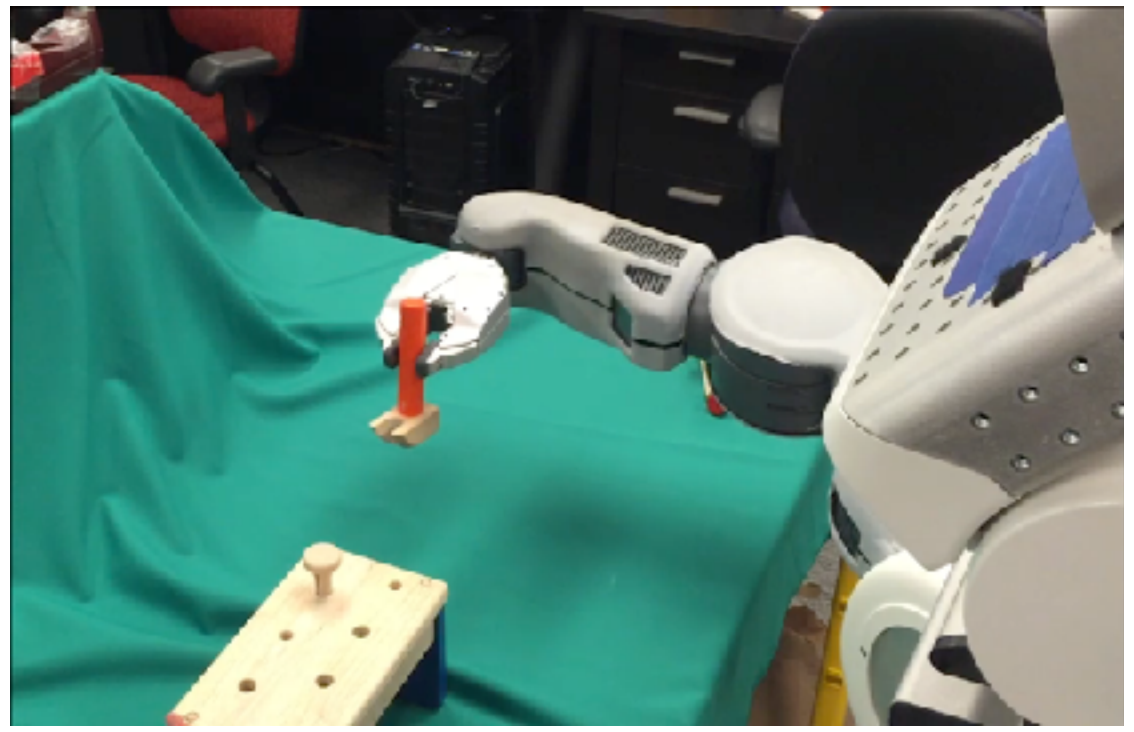
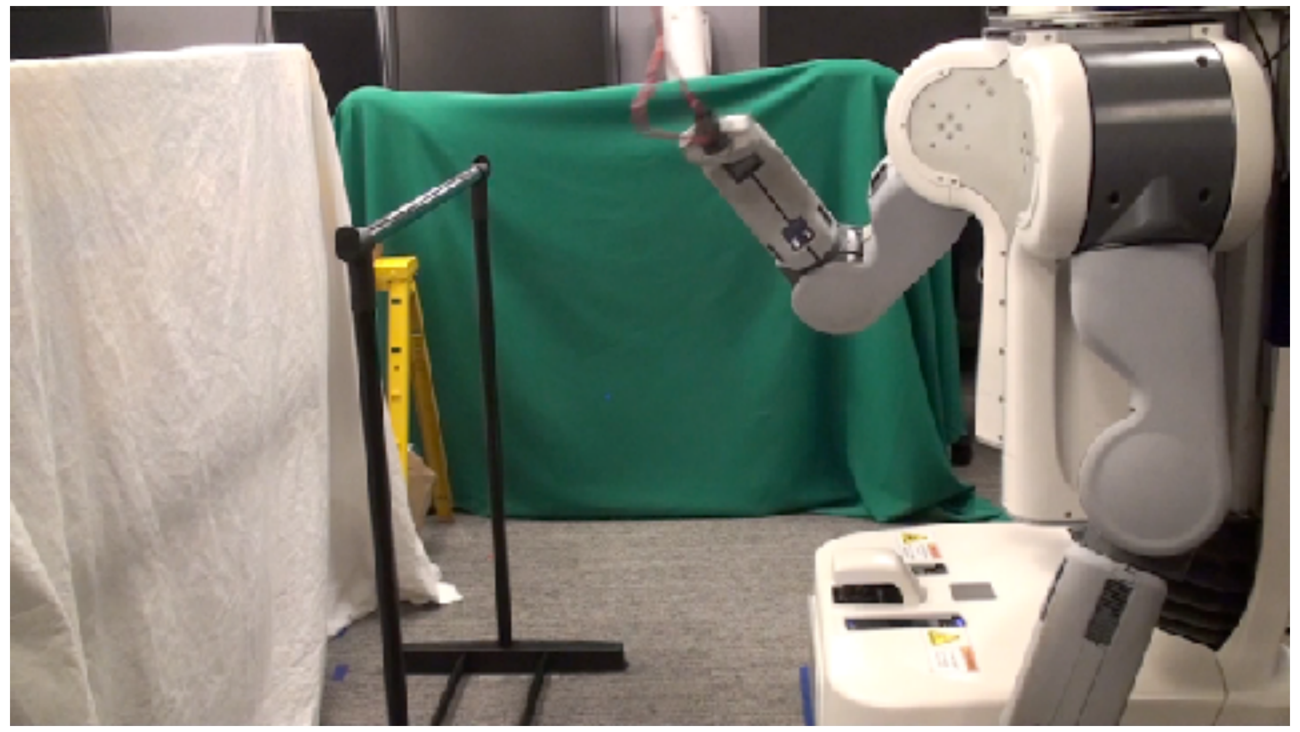
Guided Policy Search: Learning on Real Robot

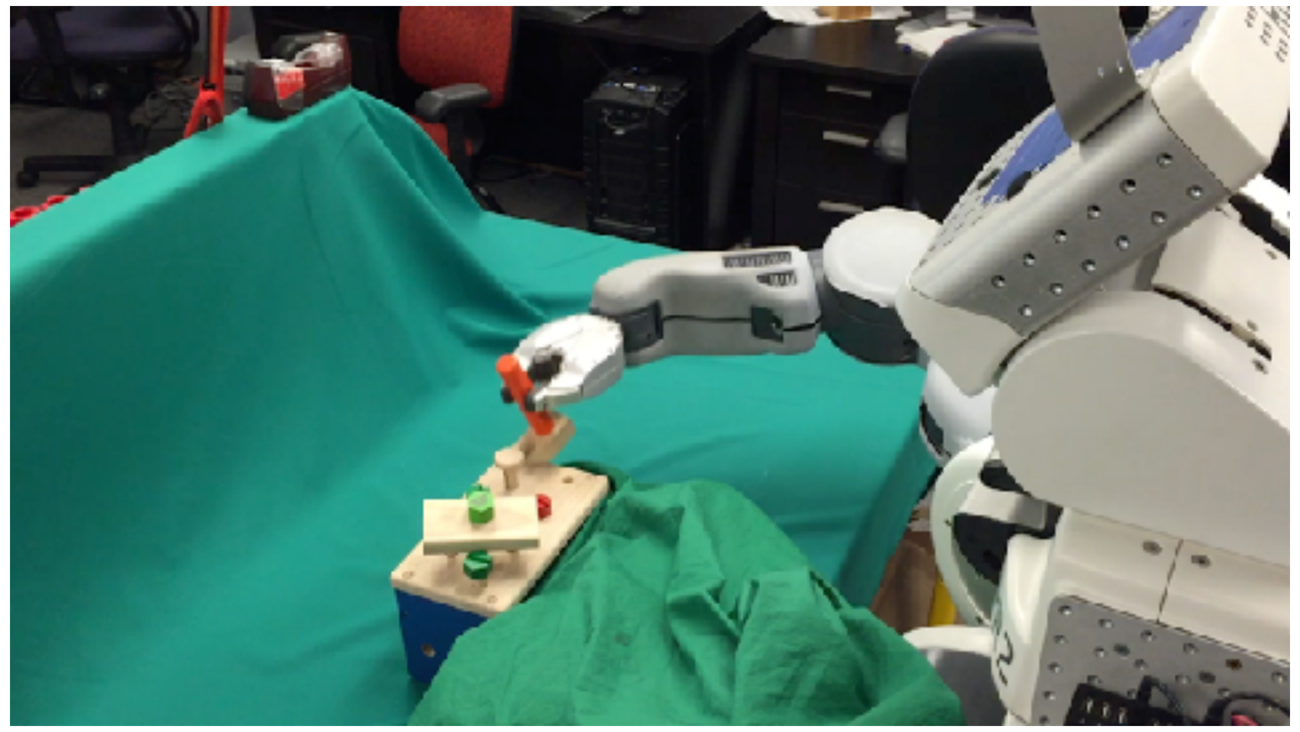
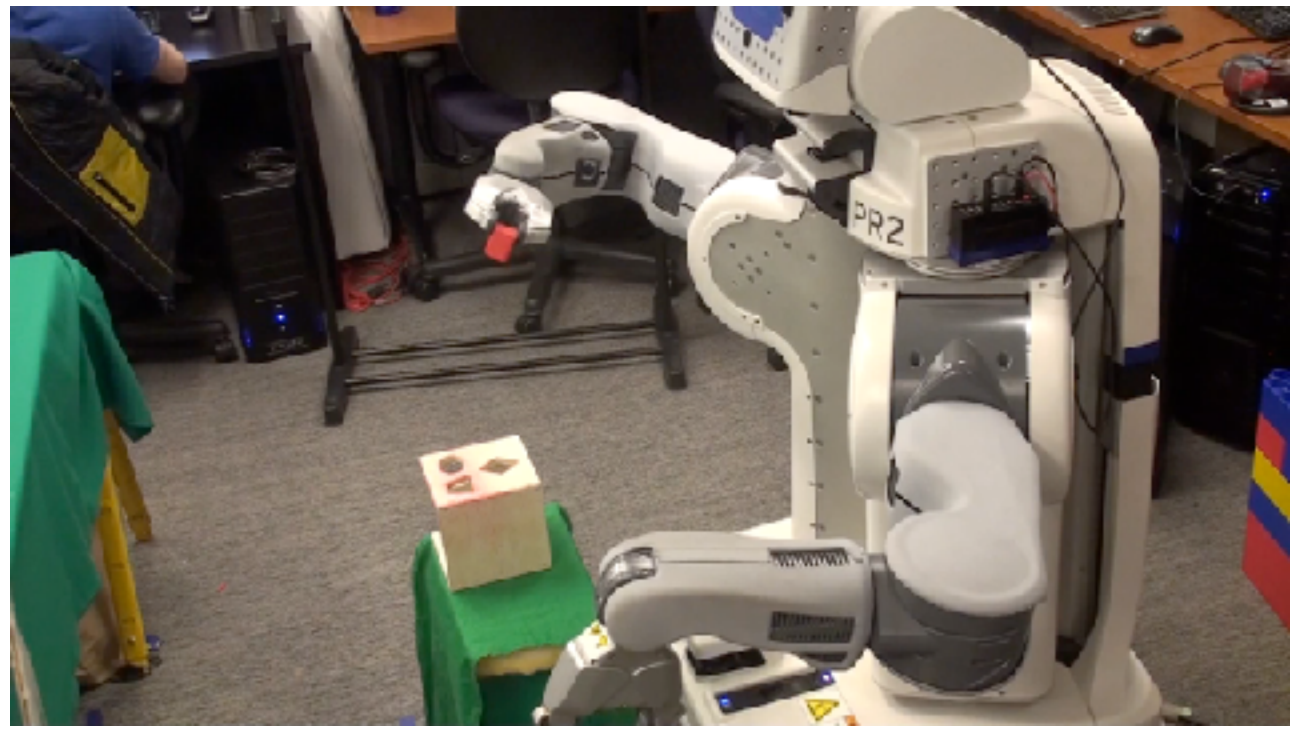


[Levine et al, 2015]

[Rocky Duan, OpenAI / UC Berkeley]

Guided Policy Search: Success Stories

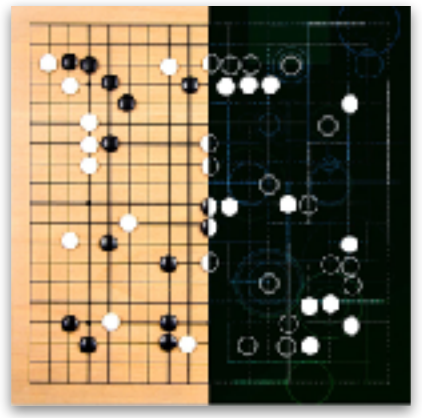



[Levine et al, 2015]

[Rocky Duan, OpenAI / UC Berkeley]

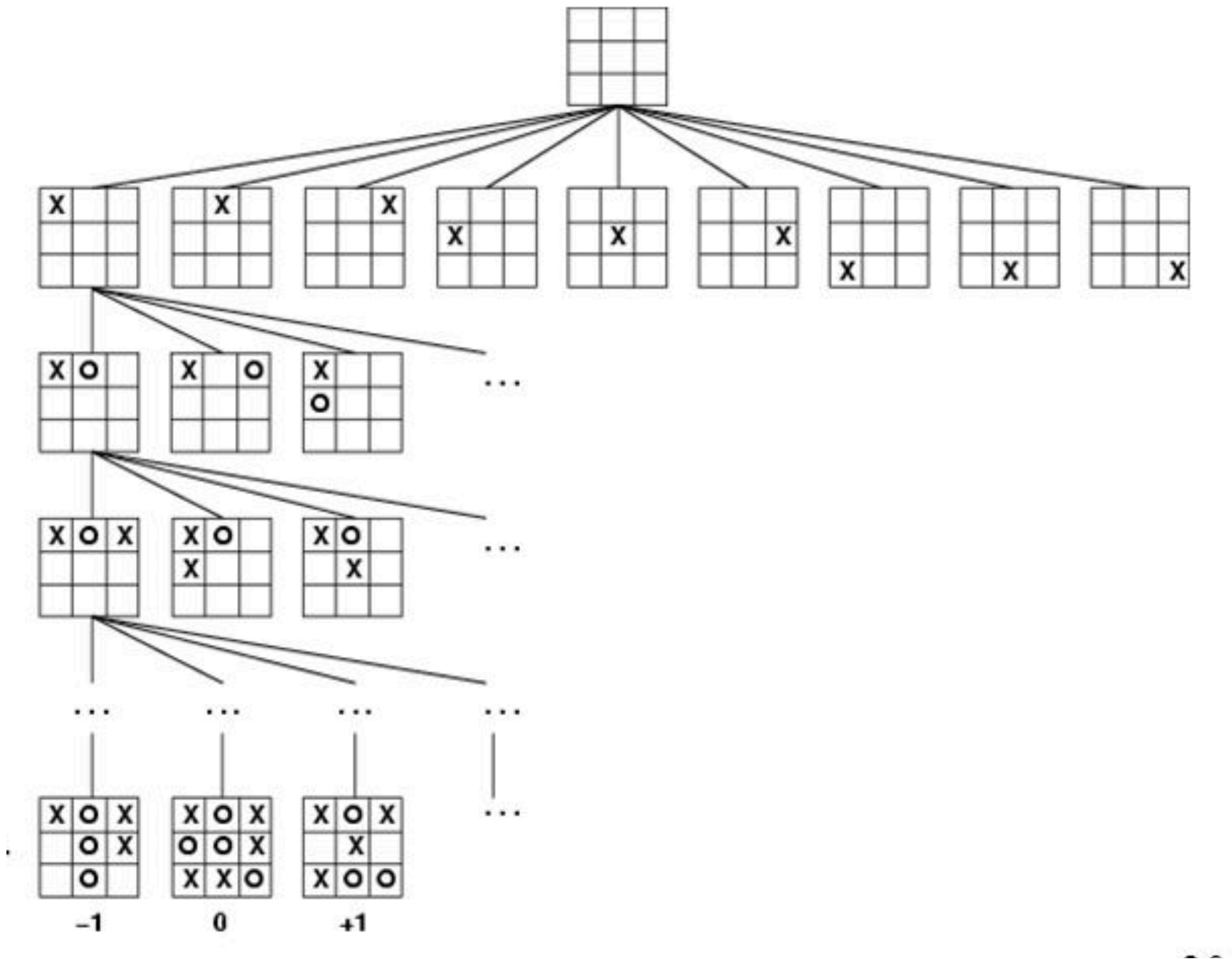
Outline

- Basics of Reinforcement Learning
- Model-Free RL
  - Value-Based Methods
  - Policy-Based Methods
- Model-Based RL
  - Guided Policy Search
  - AlphaGo



[Rocky Duan, OpenAI / UC Berkeley]

Background: Monte-Carlo Tree Search



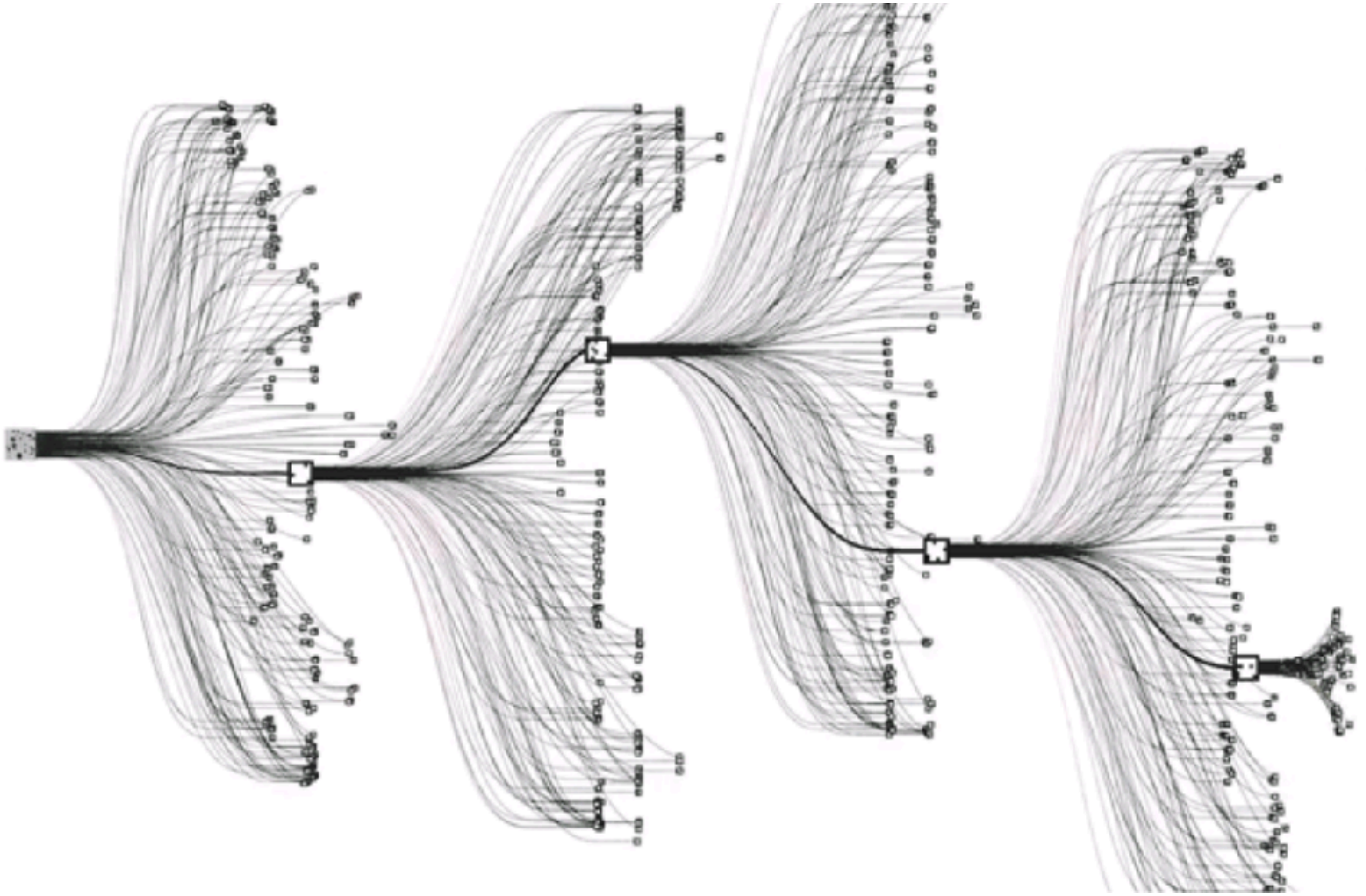
Partial game tree for tic-tac-toe

[Image credit: Wikimedia]

Number of states: less than 10^5

[Rocky Duan, OpenAI / UC Berkeley]

Background: Monte-Carlo Tree Search



Small snapshot of game tree for Go

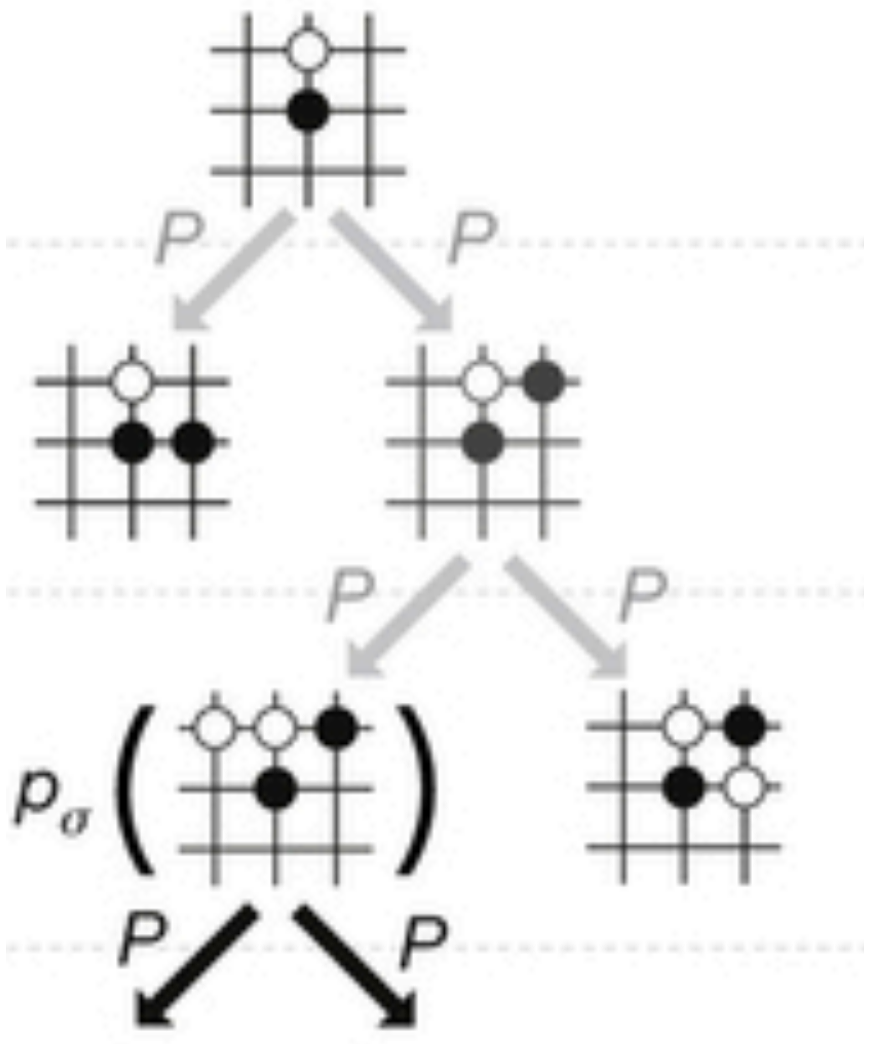
[Image credit: Deepmind]

Number of states: more than 10^{170}!

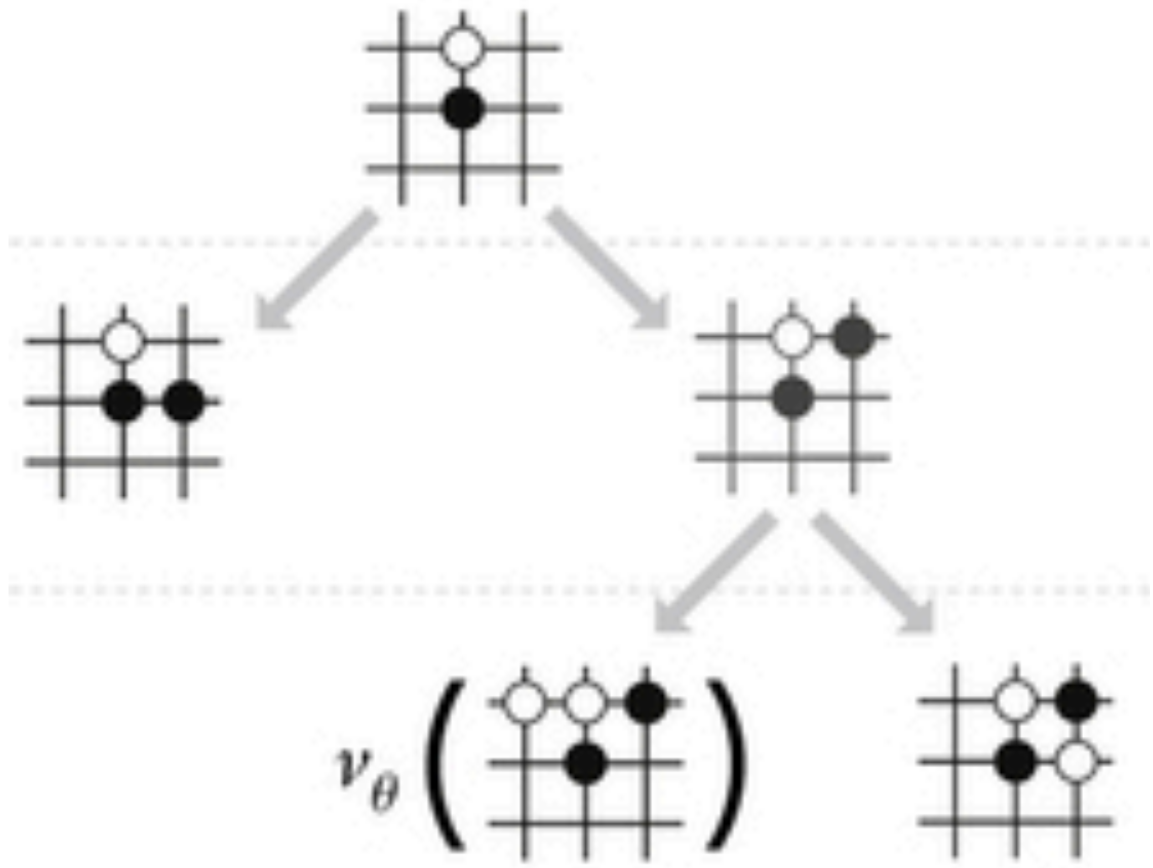
[Rocky Duan, OpenAI / UC Berkeley]

AlphaGo

Reducing breadth



Reducing depth



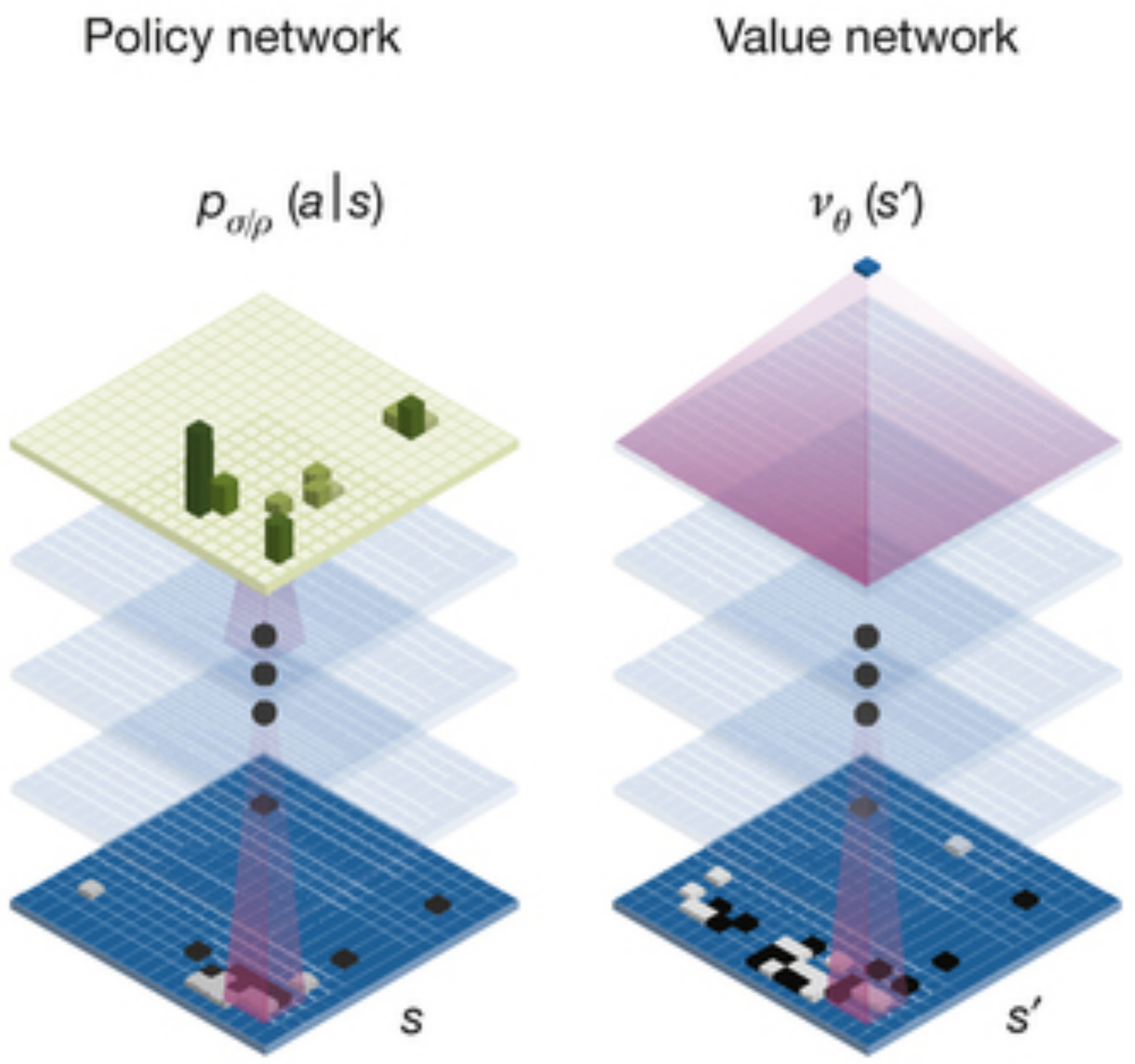
[Image credit: Silver et al, 2016]

[Rocky Duan, OpenAI / UC Berkeley]

AlphaGo

Learning the policy & value functions

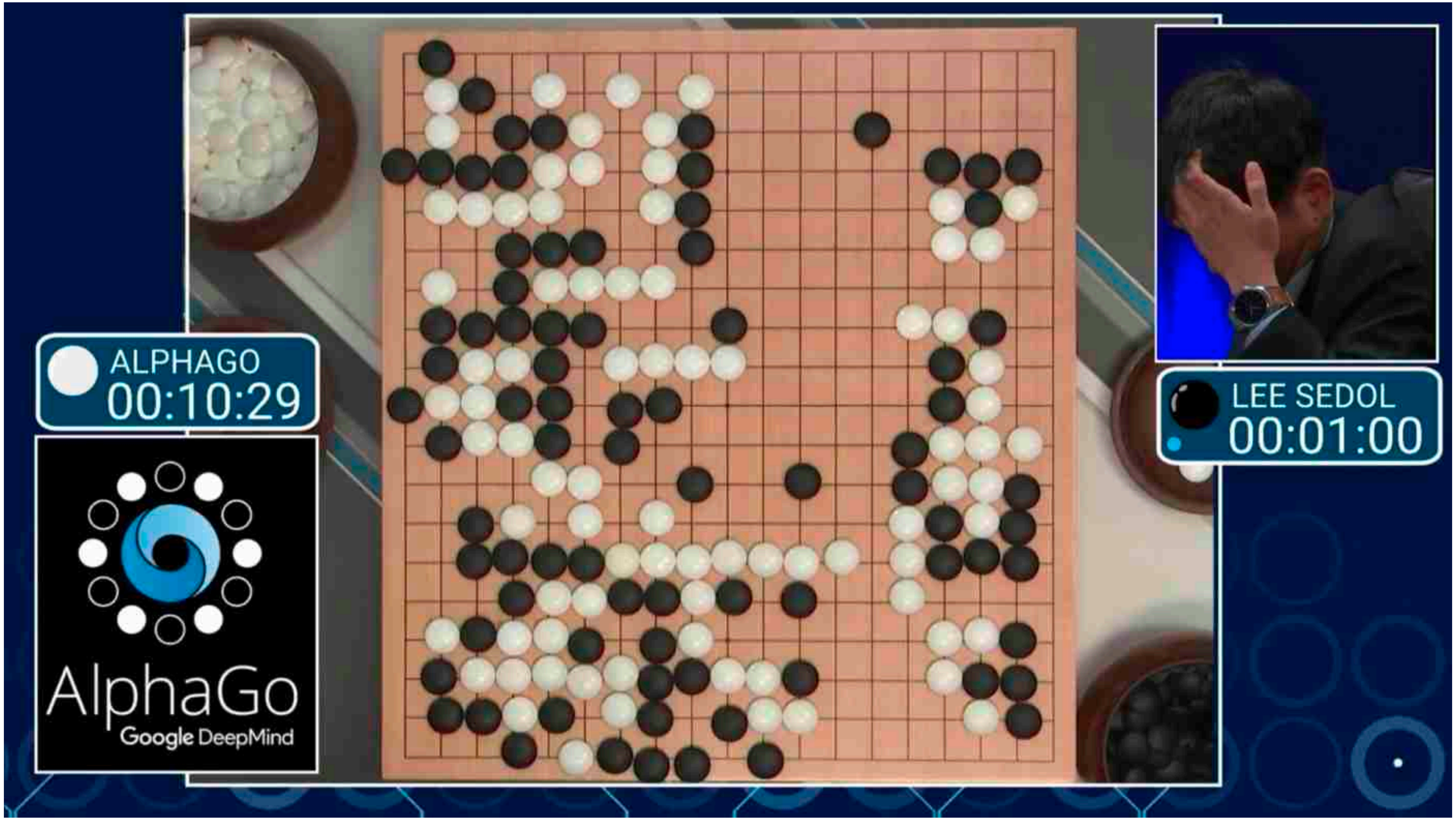
- Supervised pre-training
- Self-play



[Image credit: Silver et al, 2016]

[Rocky Duan, OpenAI / UC Berkeley]

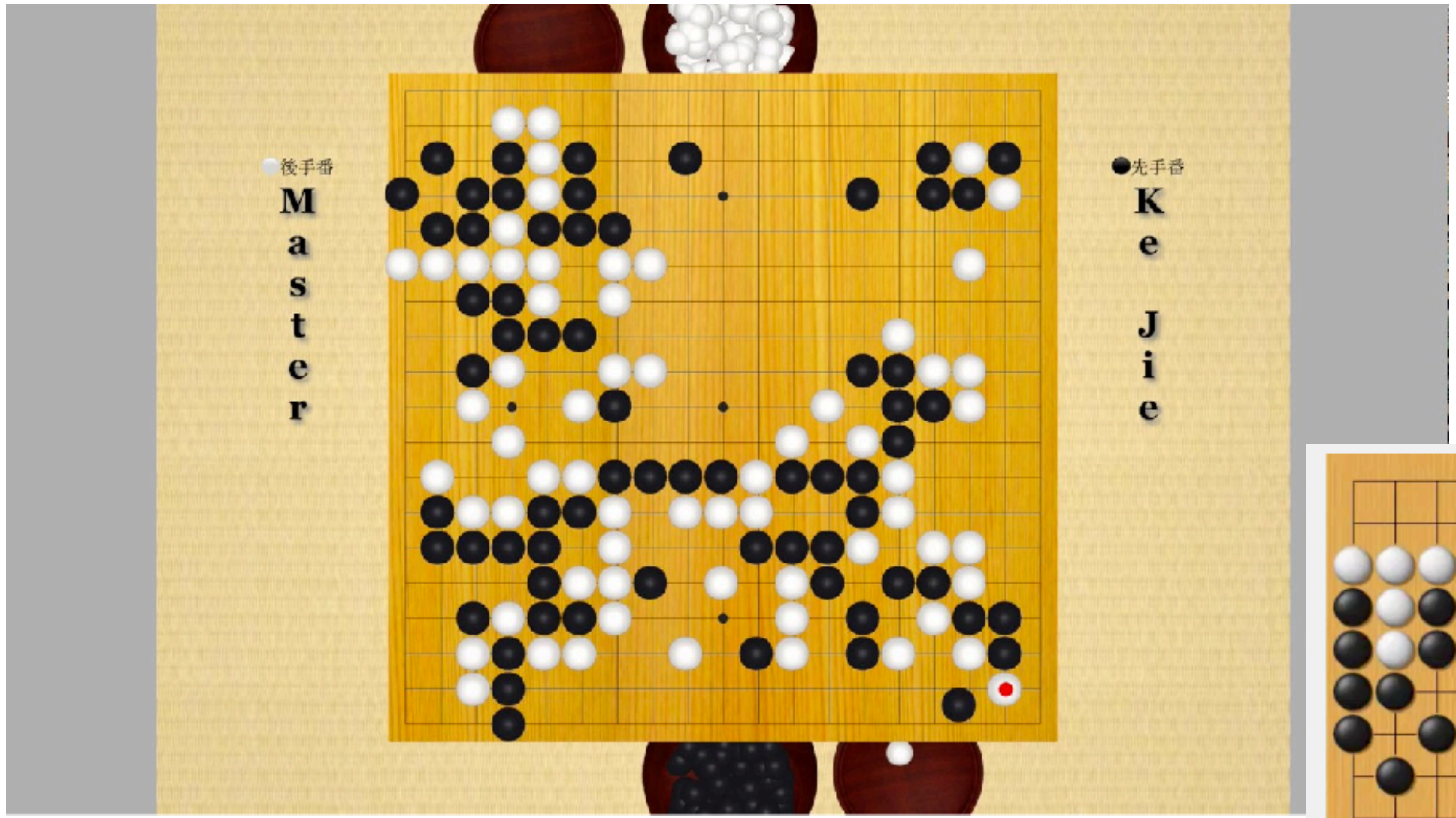
AlphaGo: Success Stories



[Silver et al, 2016]

[Rocky Duan, OpenAI / UC Berkeley]

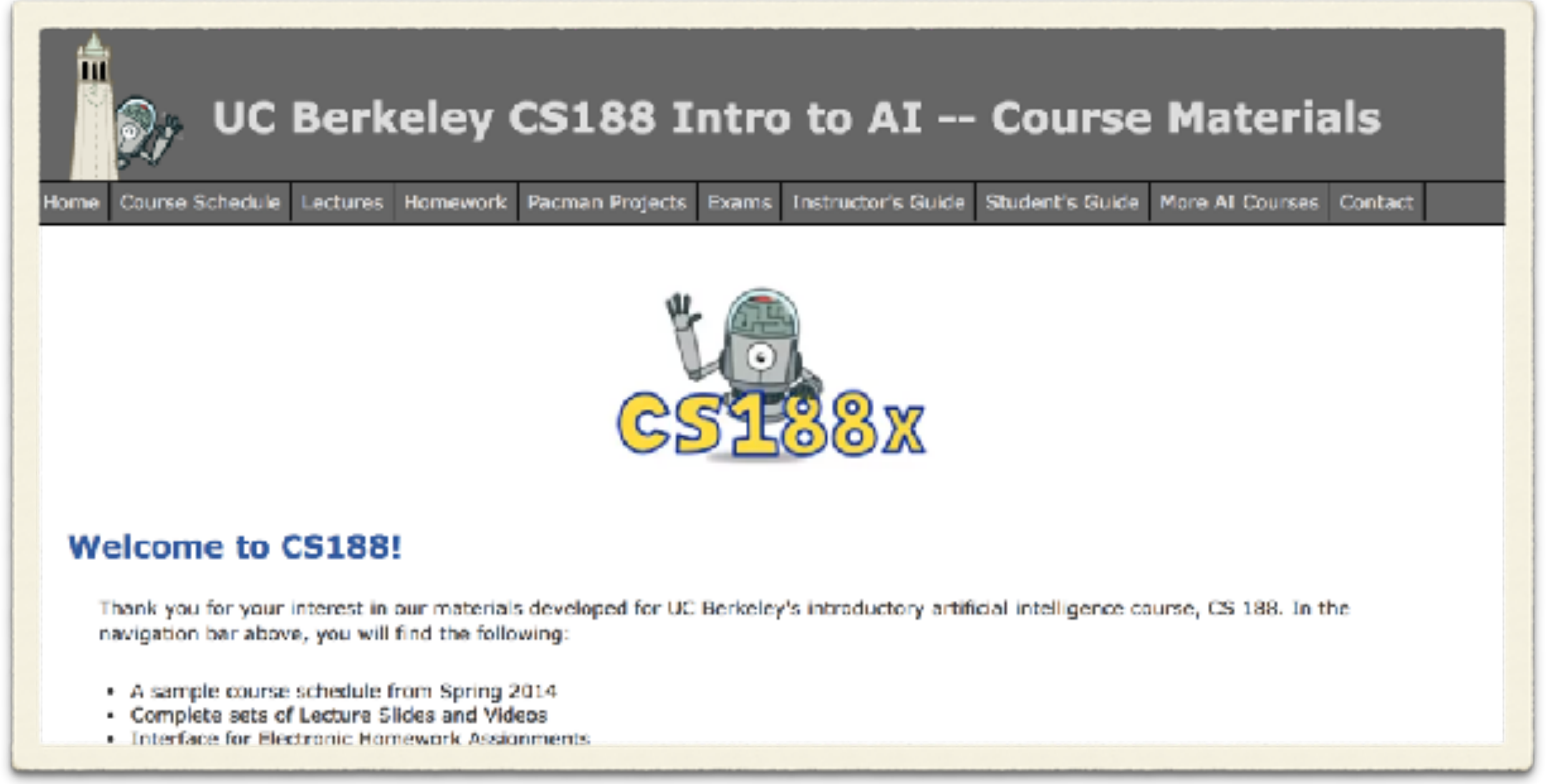
AlphaGo: Success Stories



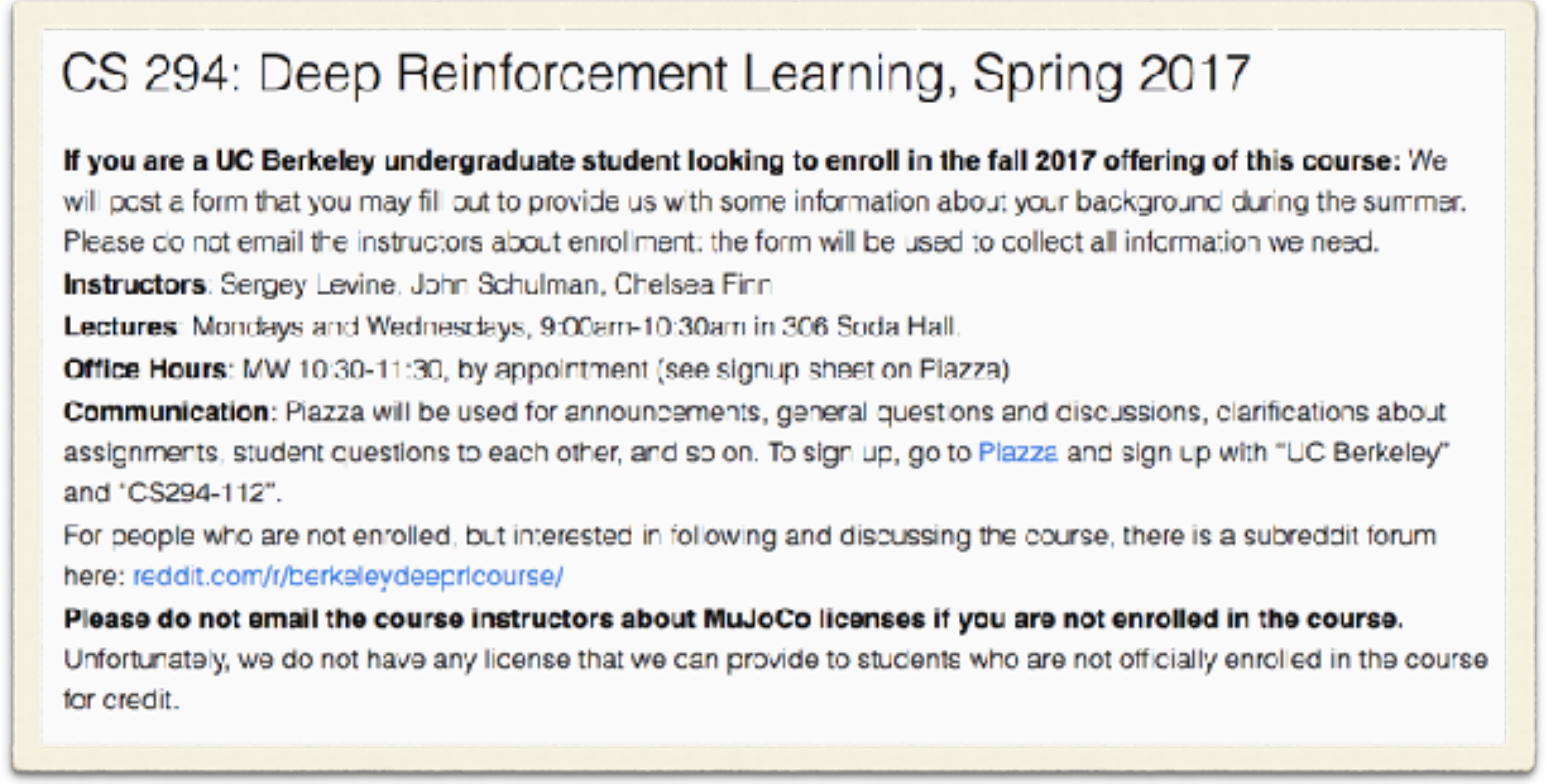
[Silver et al, 2016]

[Rocky Duan, OpenAI / UC Berkeley]

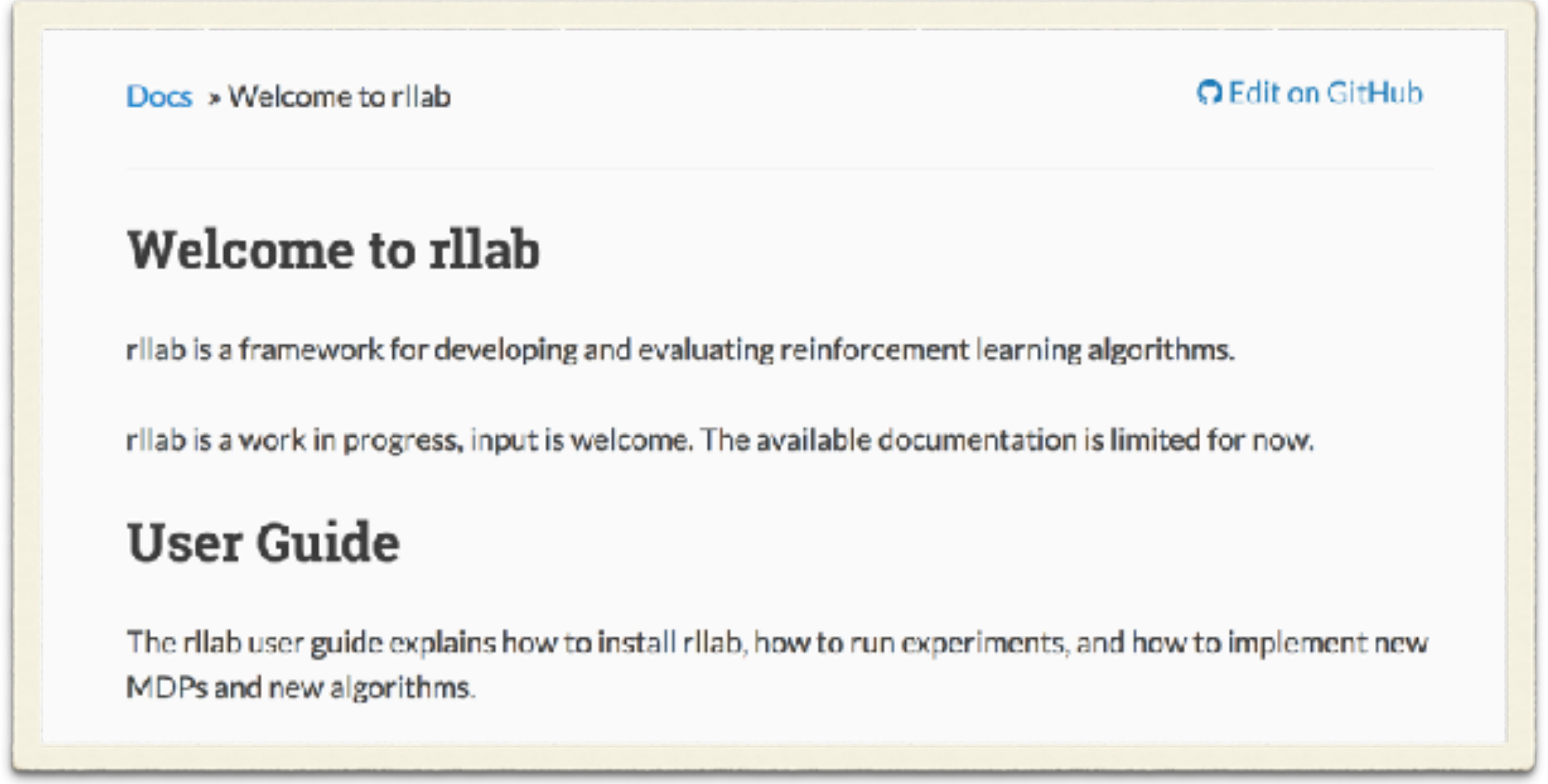
Want to Learn More?



CS188
Artificial Intelligence
bit.ly/fnal-cs188



CS294-112
Deep Reinforcement Learning
bit.ly/fnal-cs294



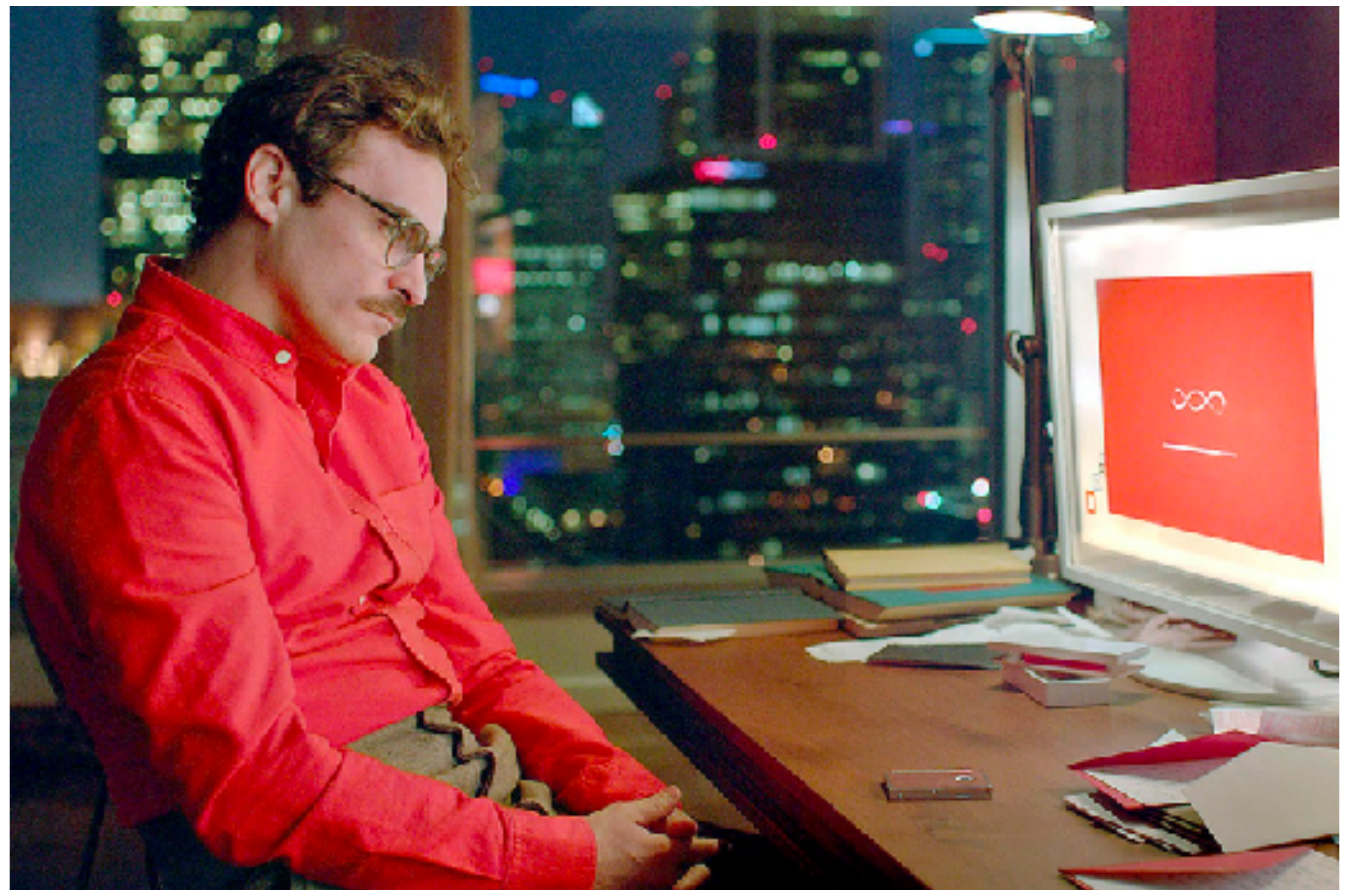
rllab
Reinforcement Learning Toolkit
bit.ly/fnal-rllab

[Rocky Duan, OpenAI / UC Berkeley]

Potential Applications

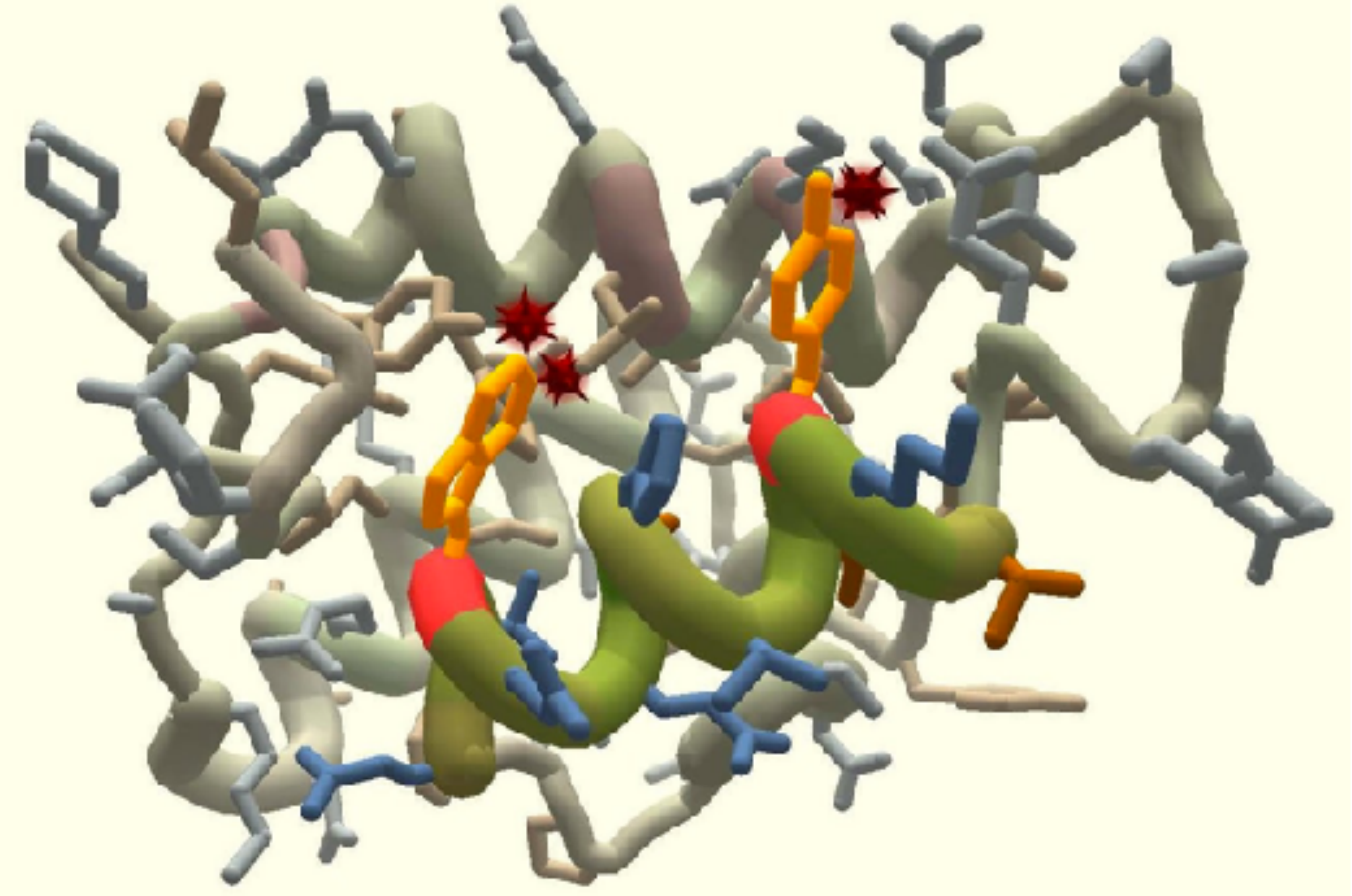
Autonomous Flight

Algorithmic Trading

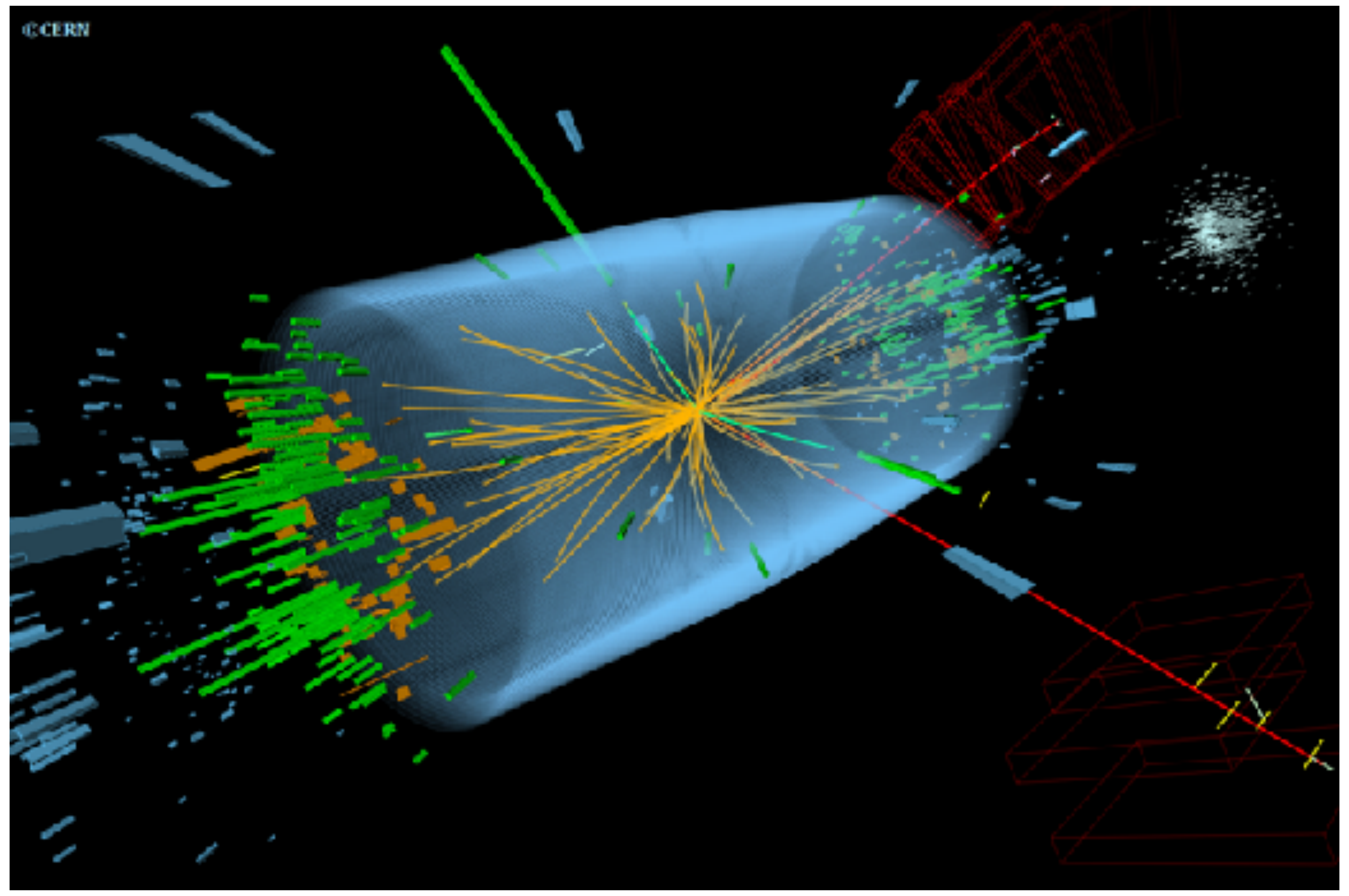


Virtual Assistant

Smart Grid



Protein Folding



Experiment Design

[Rocky Duan, OpenAI / UC Berkeley]

Thank you!

[Back-up Slides]

Target Network

$$sample = r + \gamma \max_{a'} Q_\theta(s', a')$$
$$\theta \leftarrow \theta - \alpha \nabla_\theta \left(Q_\theta(s,a) - sample\right)^2$$

Problem: constantly regressing against moving target since θ is used in computing sample estimates (and error accumulates).

Solution: use a snapshot of the parameter value to compute sample estimates, θ_{target} which is updated occasionally (once per ~10^4 updates)

$$sample = r + \gamma \max_{a'} Q_{\theta_{\text{target}}}(s', a')$$
$$\theta \leftarrow \theta - \alpha \nabla_\theta \left(Q_\theta(s,a) - sample\right)^2$$

[Rocky Duan, OpenAI / UC Berkeley]

Trust Region Methods

Problem: For high-dimensional θ building F_θ is impractical!

Trust Region Policy Optimization [Schulman 2015]

- Efficient scheme through conjugate gradient;
- Replace objective with surrogate loss, which is a better approximation yet equally efficient to evaluate.

[Rocky Duan, OpenAI / UC Berkeley]

Policy Gradient

Assumes a stochastic policy $\pi_\theta : \mathcal{S} \times \mathcal{A} \to [0, 1]$

$$
\begin{aligned}
U(\theta) &= \mathbb{E}_{\tau|\pi_\theta}[U(\tau)] \\
&= \int P_\theta(\tau) U(\tau) \mathrm{d}\tau
\end{aligned}
$$

where

$$
P_\theta(\tau) = P(s_0) \prod_t \pi_\theta(a_t|s_t) P(s_{t+1}|s_t, a_t)
$$

[Rocky Duan, OpenAI / UC Berkeley]

Policy Gradient

$$\begin{aligned}
\nabla_\theta U(\theta) &= \nabla_\theta \int P_\theta(\tau) U(\tau) \mathrm{d}\tau \\
&= \int \nabla_\theta P_\theta(\tau) U(\tau) \mathrm{d}\tau \\
&= \int P_\theta(\tau) \frac{\nabla_\theta P_\theta(\tau)}{P_\theta(\tau)} U(\tau) \mathrm{d}\tau \\
&= \int P_\theta(\tau) \nabla_\theta \log P_\theta(\tau) U(\tau) \mathrm{d}\tau \\
&= \mathbb{E}_{\tau|\pi_\theta}[\nabla_\theta \log P_\theta(\tau) U(\tau)] \\
&\approx \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta \log P_\theta(\tau_i) U(\tau_i)
\end{aligned}$$

[Rocky Duan, OpenAI / UC Berkeley]

Policy Gradient

Recall

$$P_\theta(\tau) = P(s_0) \prod_t \pi_\theta(a_t|s_t) P(s_{t+1}|s_t, a_t)$$

Hence $\nabla_\theta \log P_\theta(\tau) = \nabla_\theta \log P(s_0) + \sum_t \left(\nabla_\theta \log \pi_\theta(a_t|s_t) + \nabla_\theta \log P(s_{t+1}|s_t, a_t)\right)$

$$= \sum_t \nabla_\theta \log \pi_\theta(a_t|s_t)$$

No need to differentiate through dynamics!

[Rocky Duan, OpenAI / UC Berkeley]

Our Current / Future Directions

- Faster learning
  - Exploration [Stadie et al, 2015; Houthooft et al, 2016; Tang et al, 2016]
  - Meta-learning: RL^2 [Duan et al, 2016]; One-shot Imitation Learning [Duan et al, 2017]; MAML [Finn et al, 2017]
- Transfer learning
  - Modular networks [Devin et al, 2017]; Invariant feature spaces [Gupta et al, 2017]
  - Domain randomization [Tobin et al, 2017]
- Safe learning
  - [Kahn et al, 2017; Held et al, 2016]
- Unsupervised / Semi-supervised learning
  - InfoGAN [Chen et al, 2016]; VLAE [Chen et al, 2017]; Temporal segment models [Mishra et al]
- Grounded language / Multi-agent
  - "Inventing" language [Mordatch & Abbeel, 2017]
- Imitation
  - Generative Adversarial Imitation Learning [Ho et al, 2016]; Guided Cost Learning [Finn et al, 2016]; Third-person [Stadie et al, 2017]
- Value alignment / AI safety
  - CIRL [Hadfield-Menell et al, 2016]; Off-switch [Hadfield-Menell et al, 2016]
  - Communication [Huang et al, 2017]

[Rocky Duan, OpenAI / UC Berkeley]