

Reinforcement Learning

Introduction

Antonio Pio Ricciardi, PhD



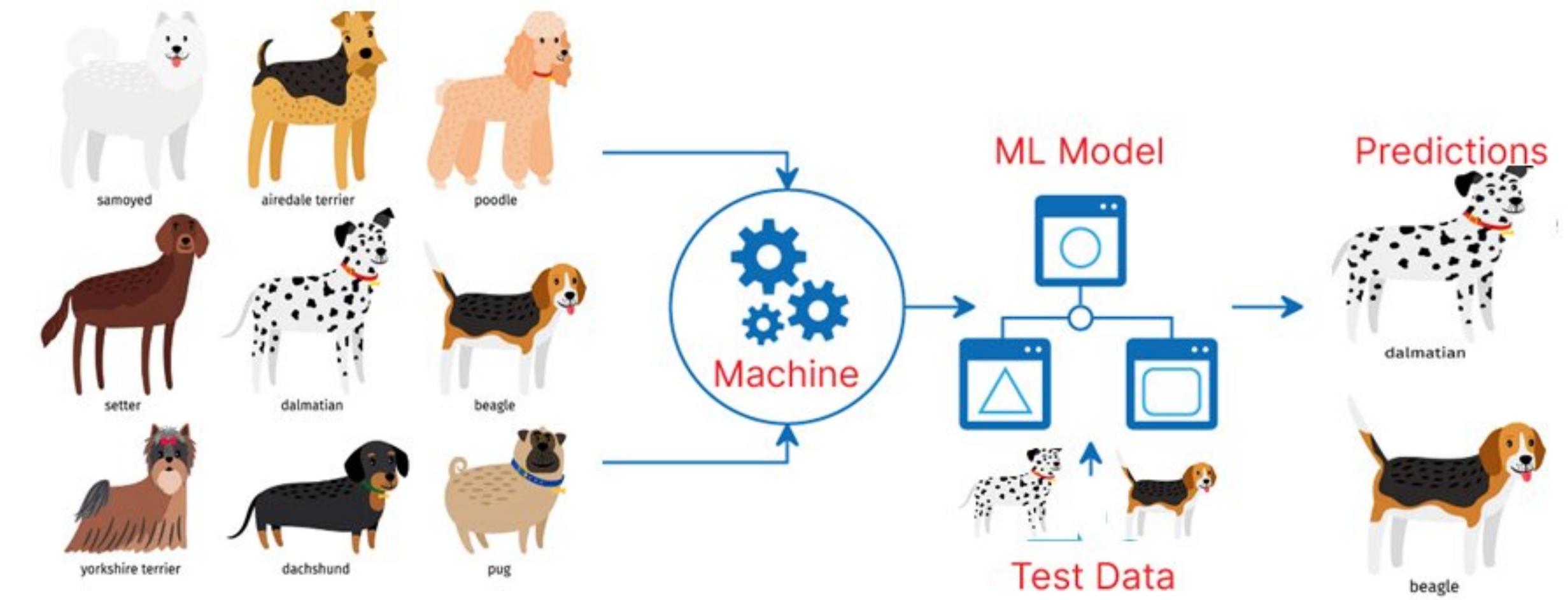
SAPIENZA
UNIVERSITÀ DI ROMA

Supervised Learning vs Reinforcement Learning

Supervised Learning

Learn using labeled data

Need pairs $\{x_i, y_i\}$



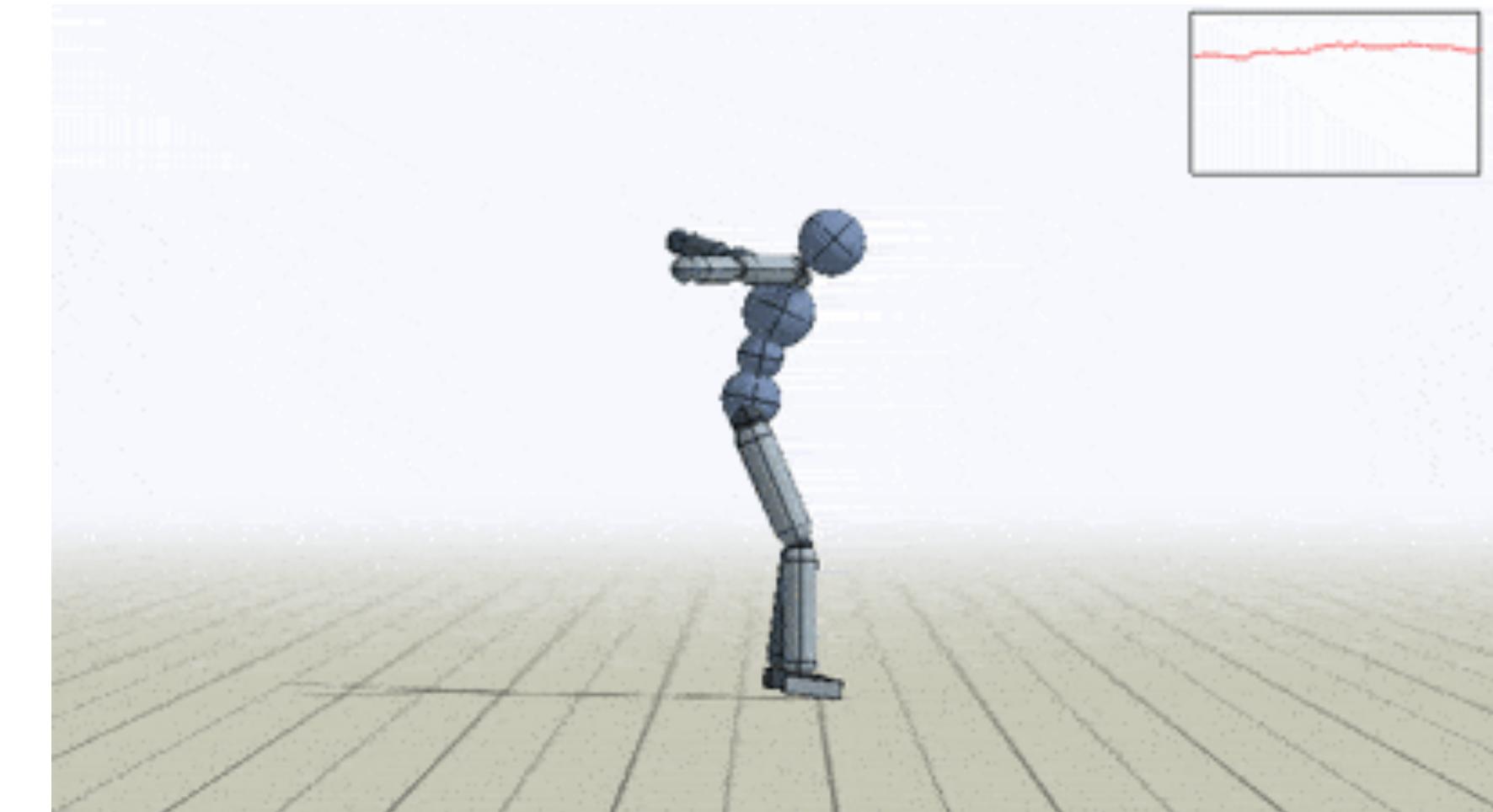
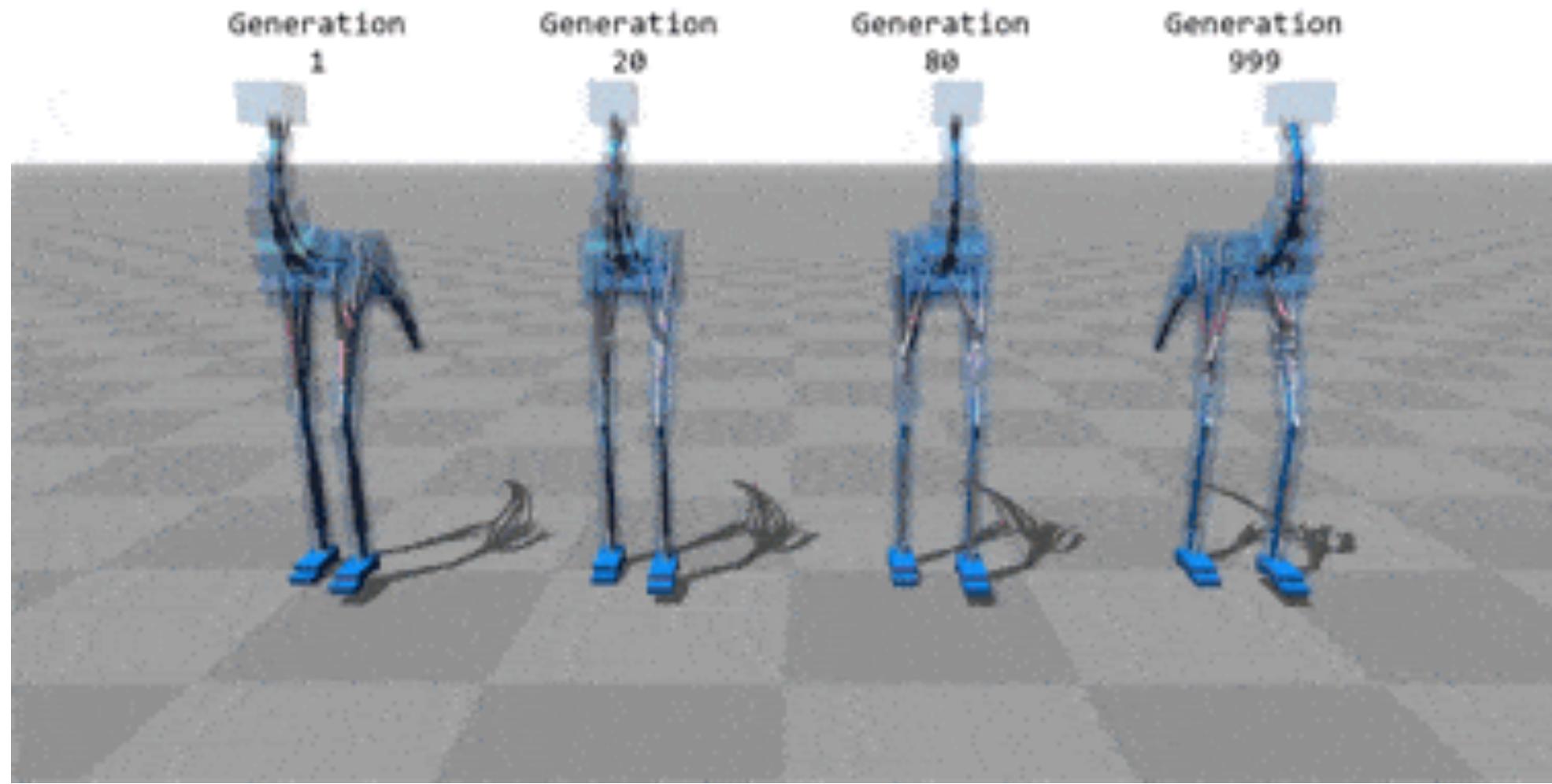
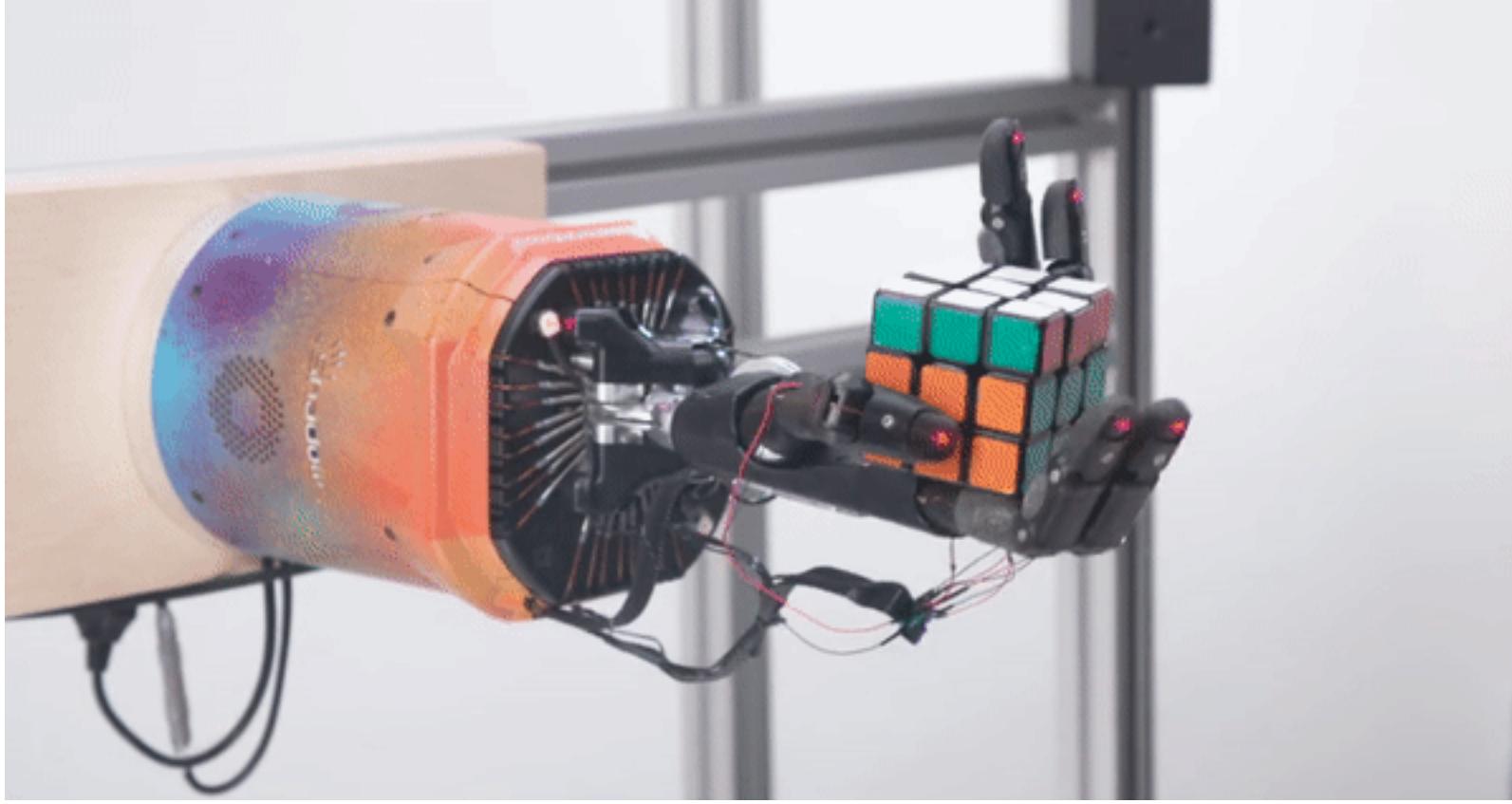
Reinforcement Learning

Framework for solving control tasks

Learn from the environment through trial and error

Receive rewards as feedback

Reinforcement Learning



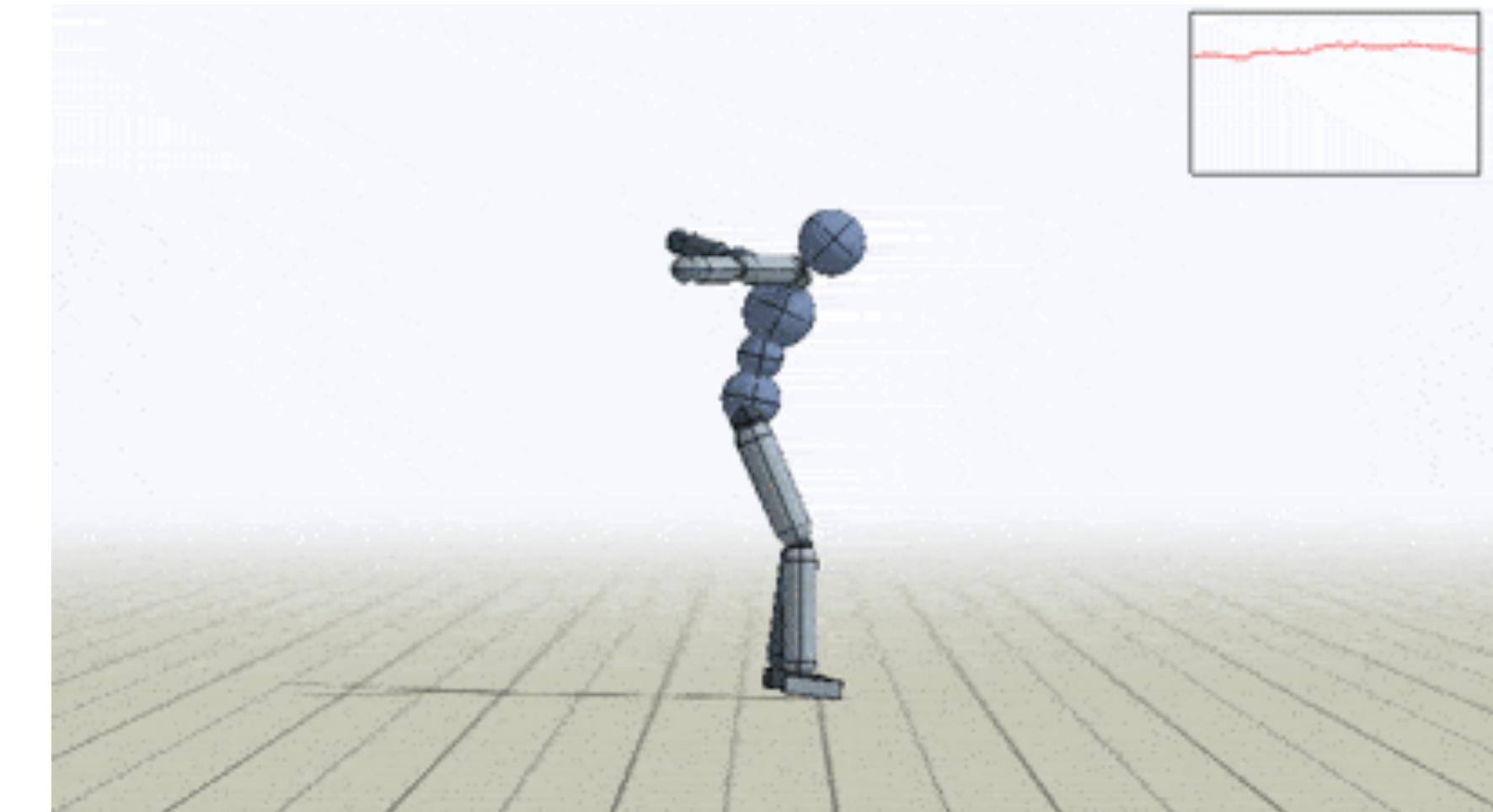
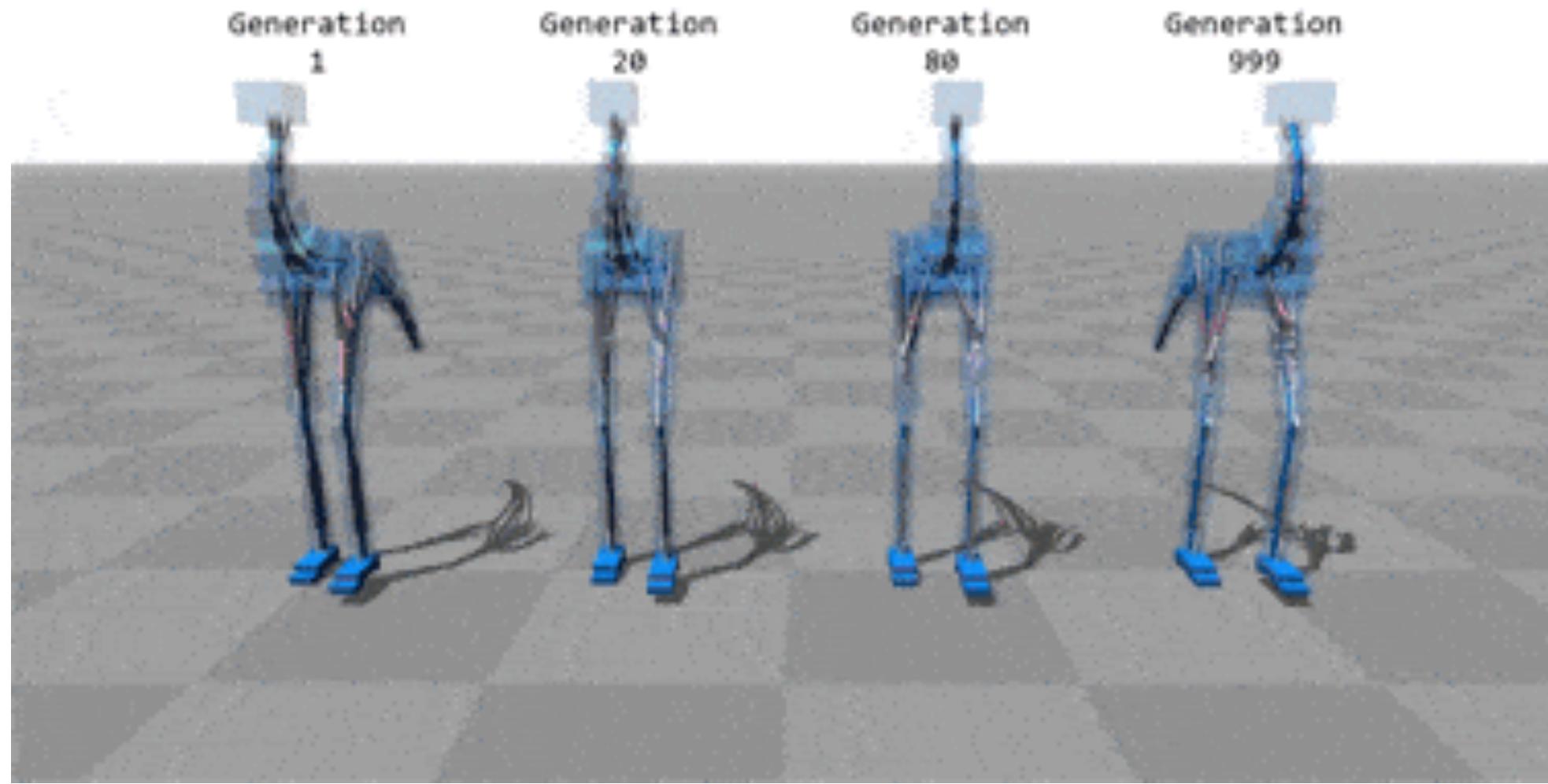
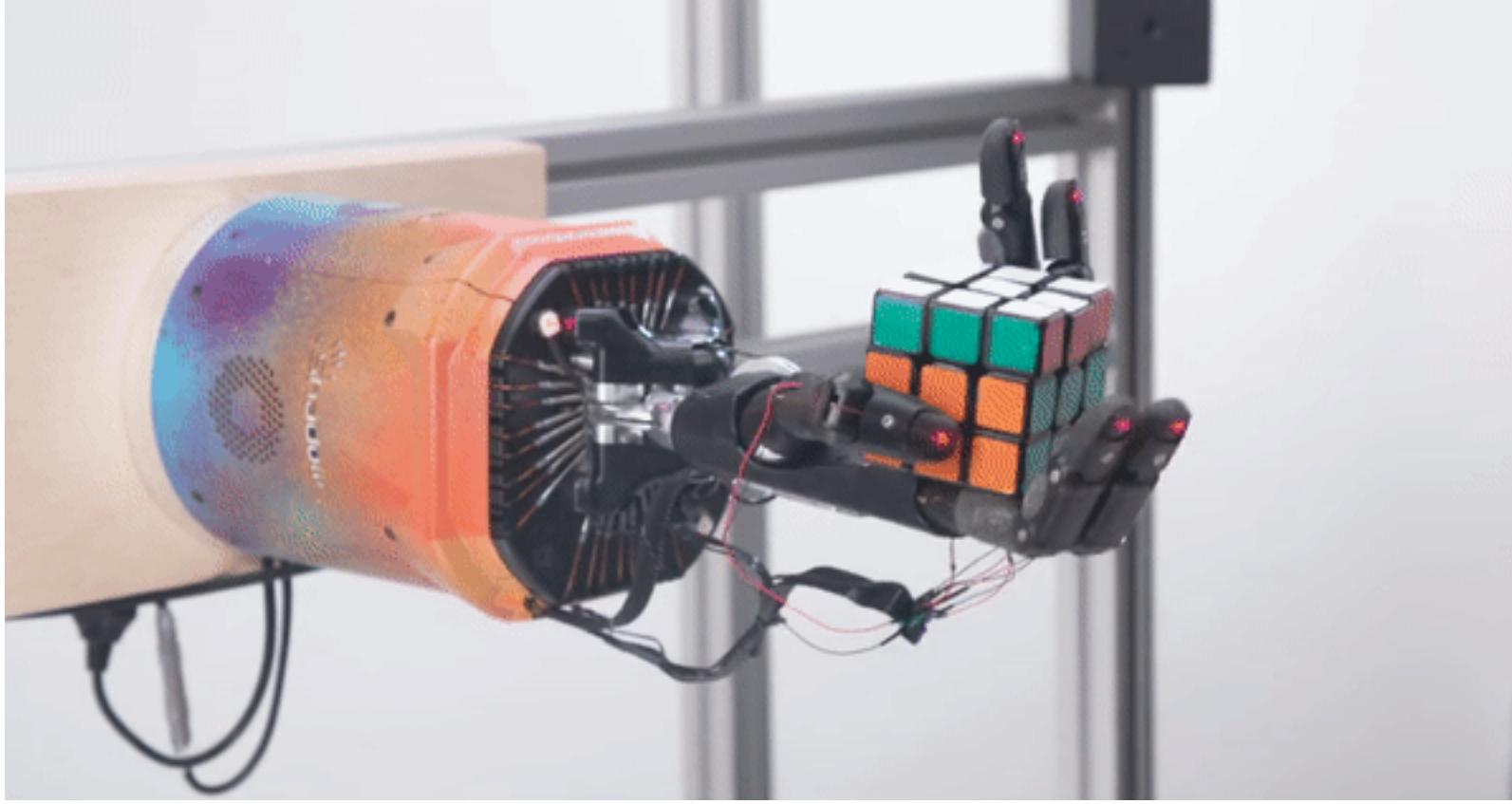
Akkaya, Ilge, et al. "Solving rubik's cube with a robot hand." arXiv preprint arXiv:1910.07113 (2019). (<https://openai.com/research/solving-rubiks-cube>)

Mnih, V., Kavukcuoglu, K., Silver, D. et al. Human-level control through deep reinforcement learning. Nature **518**, 529–533 (2015).

Geijtenbeek, Thomas, Michiel Van De Panne, and A. Frank Van Der Stappen. "Flexible muscle-based locomotion for bipedal creatures." ACM Transactions on Graphics (TOG) 32.6 (2013): 1-11.

Peng, Xue Bin, et al. "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills." ACM Transactions On Graphics (TOG) 37.4 (2018): 1-14.

Reinforcement Learning



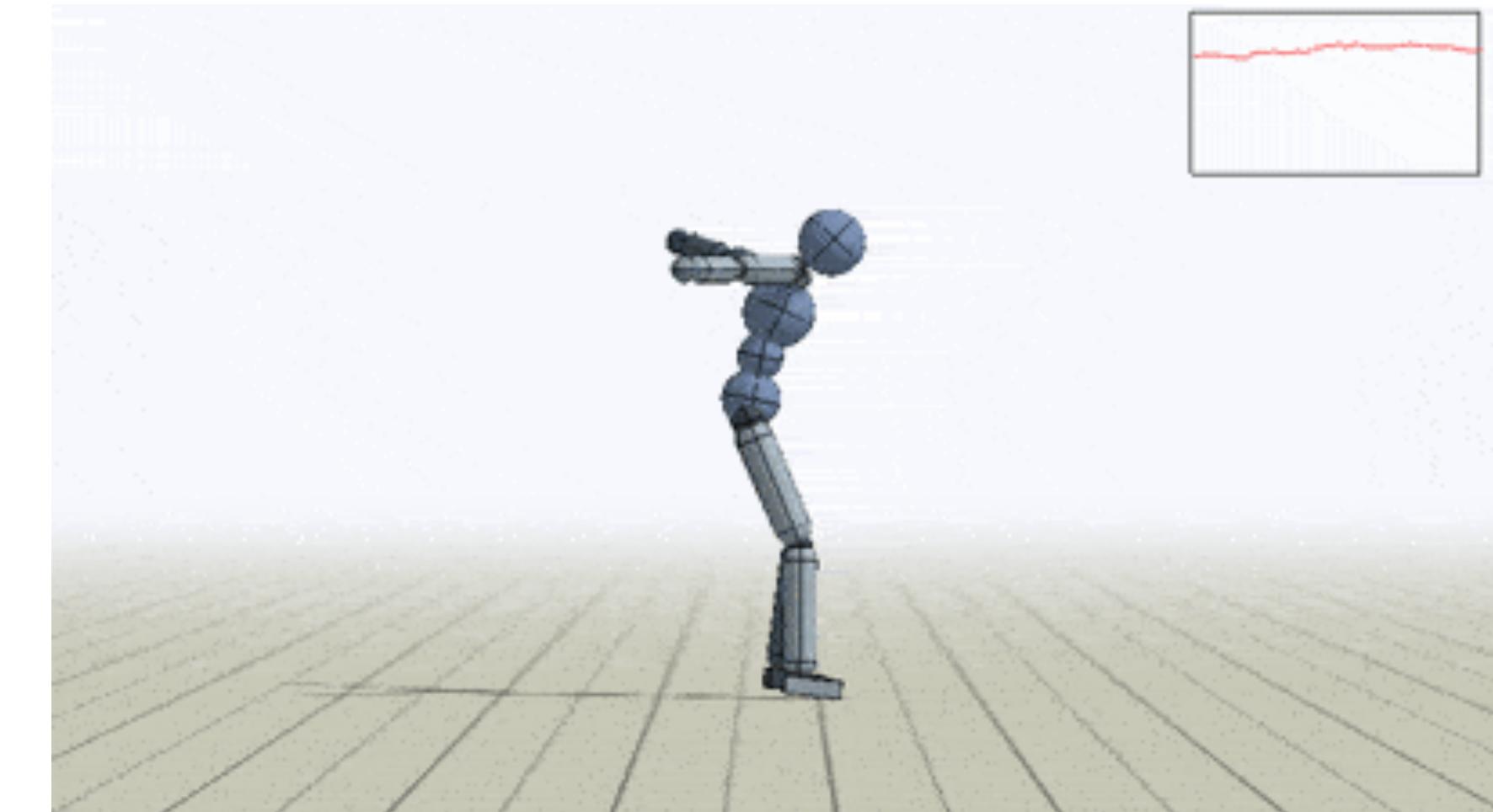
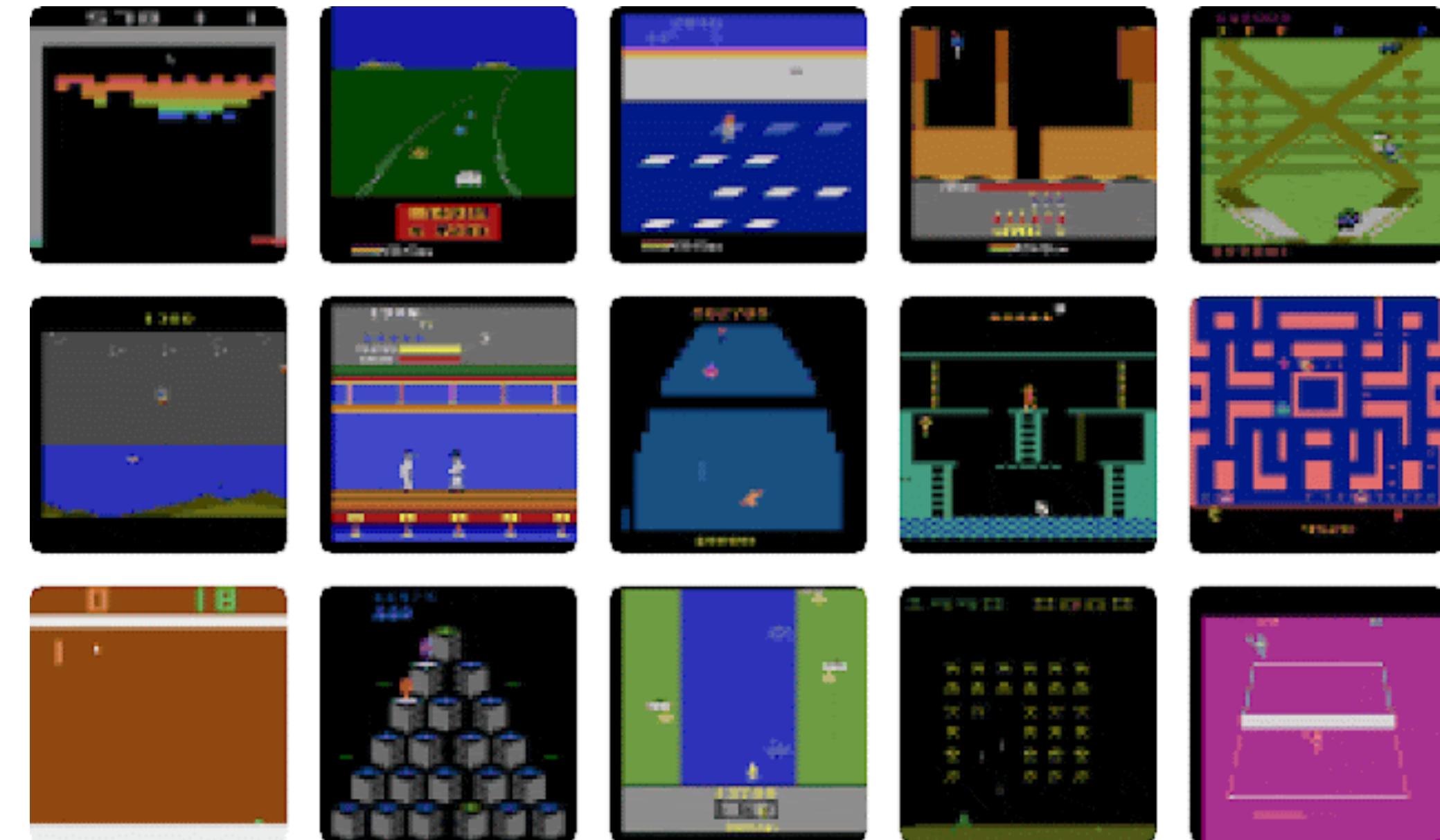
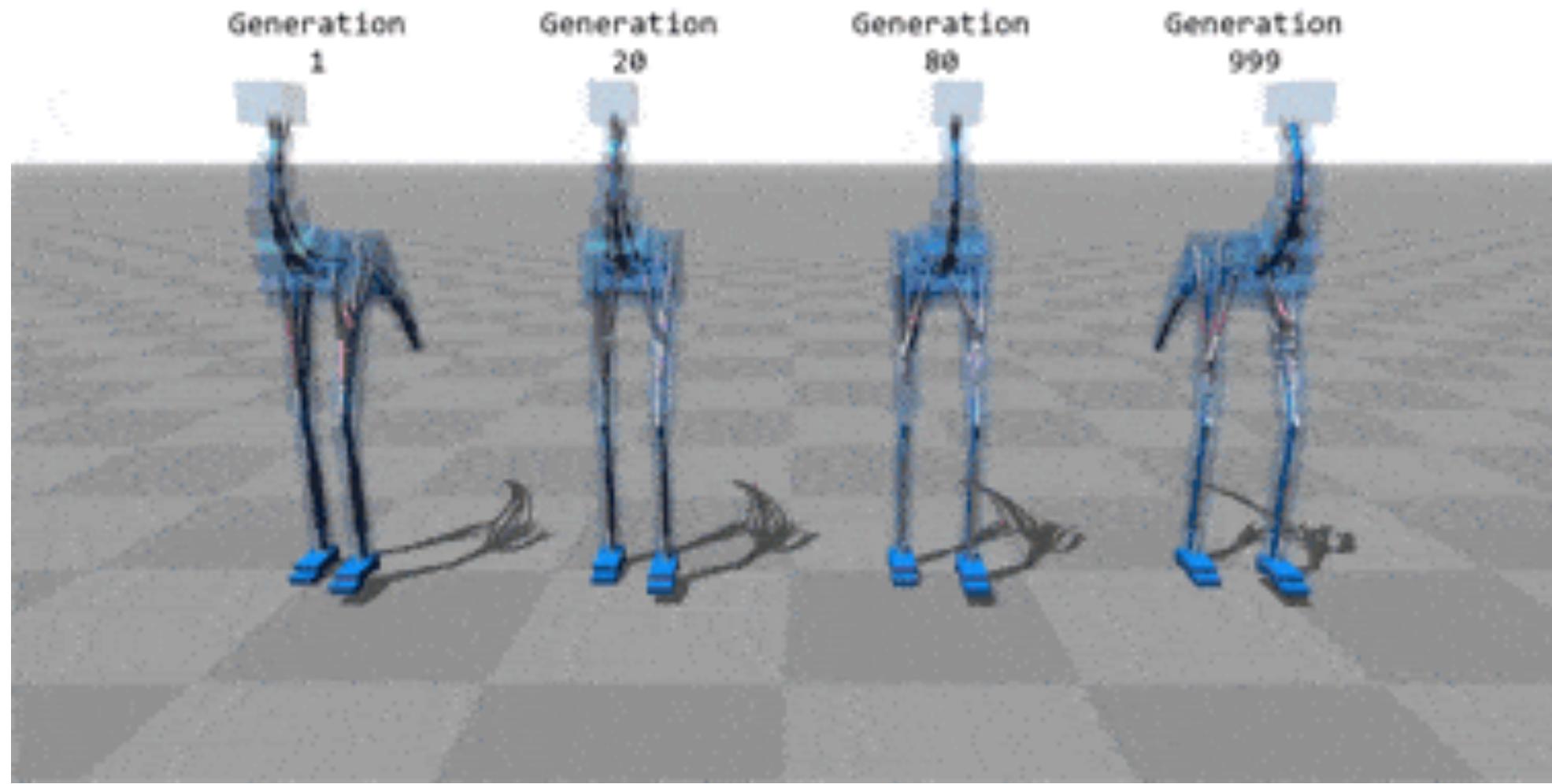
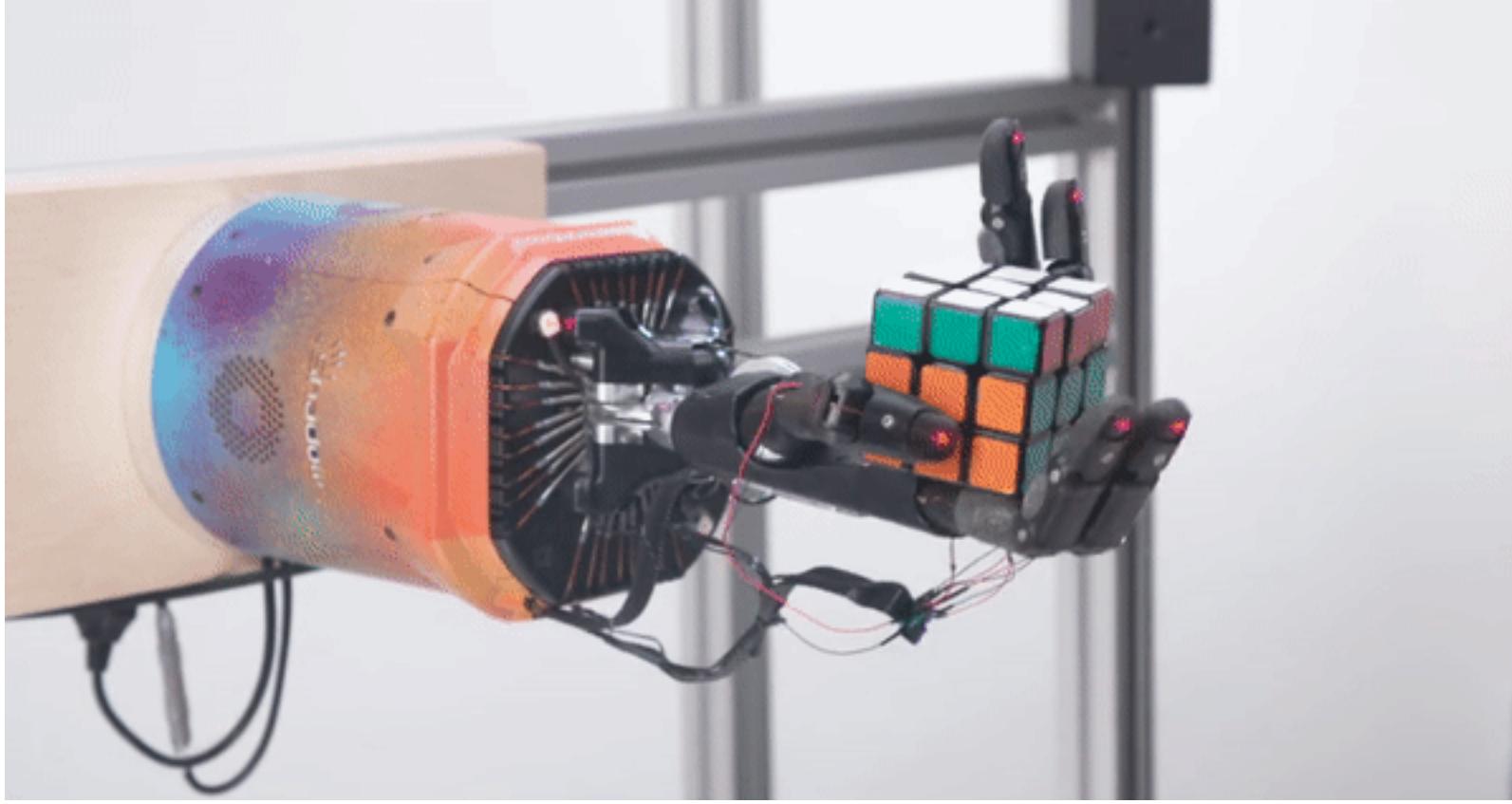
Akkaya, Ilge, et al. "Solving rubik's cube with a robot hand." arXiv preprint arXiv:1910.07113 (2019). (<https://openai.com/research/solving-rubiks-cube>)

Mnih, V., Kavukcuoglu, K., Silver, D. et al. Human-level control through deep reinforcement learning. Nature **518**, 529–533 (2015).

Geijtenbeek, Thomas, Michiel Van De Panne, and A. Frank Van Der Stappen. "Flexible muscle-based locomotion for bipedal creatures." ACM Transactions on Graphics (TOG) 32.6 (2013): 1-11.

Peng, Xue Bin, et al. "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills." ACM Transactions On Graphics (TOG) 37.4 (2018): 1-14.

Reinforcement Learning



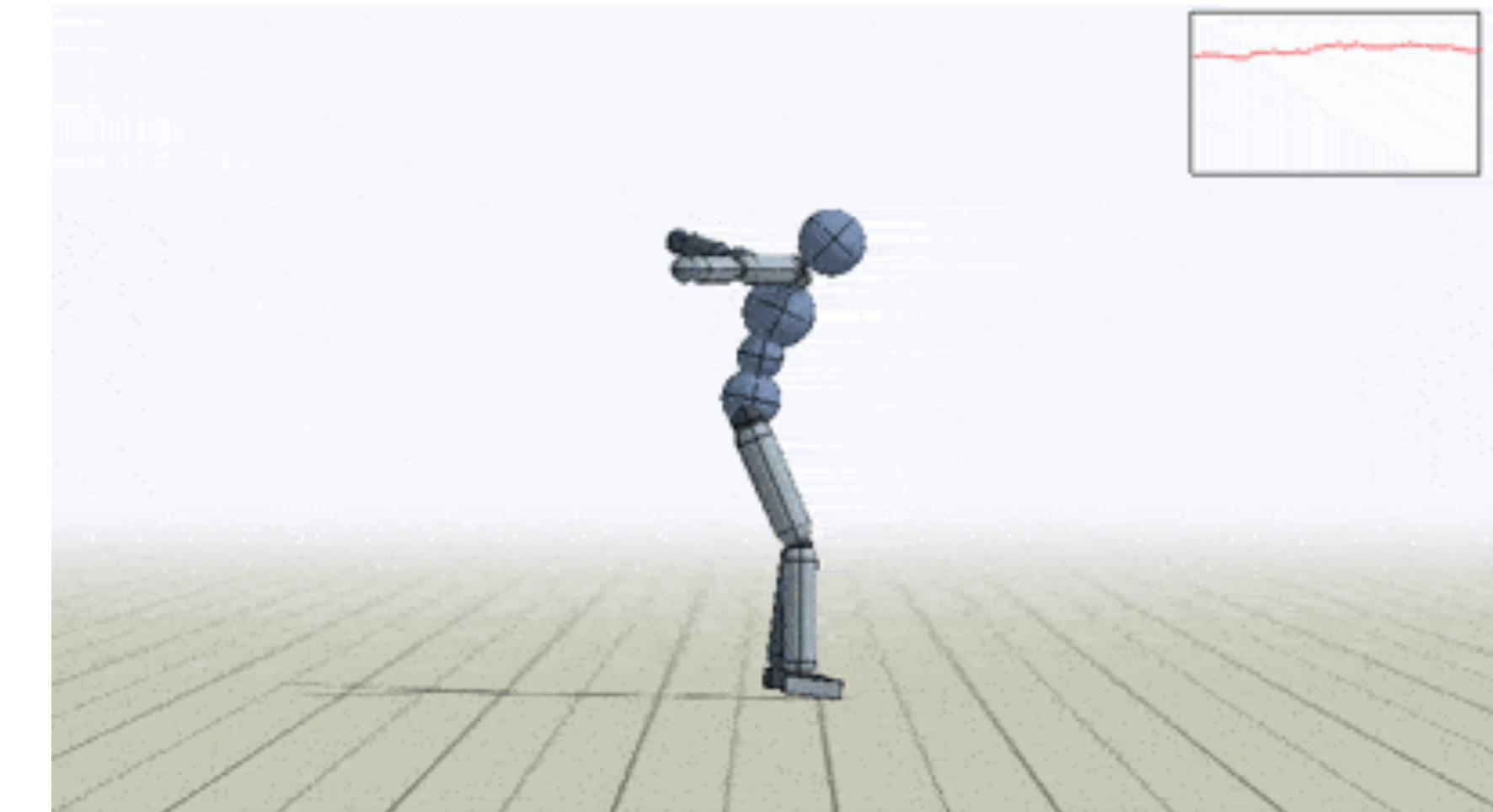
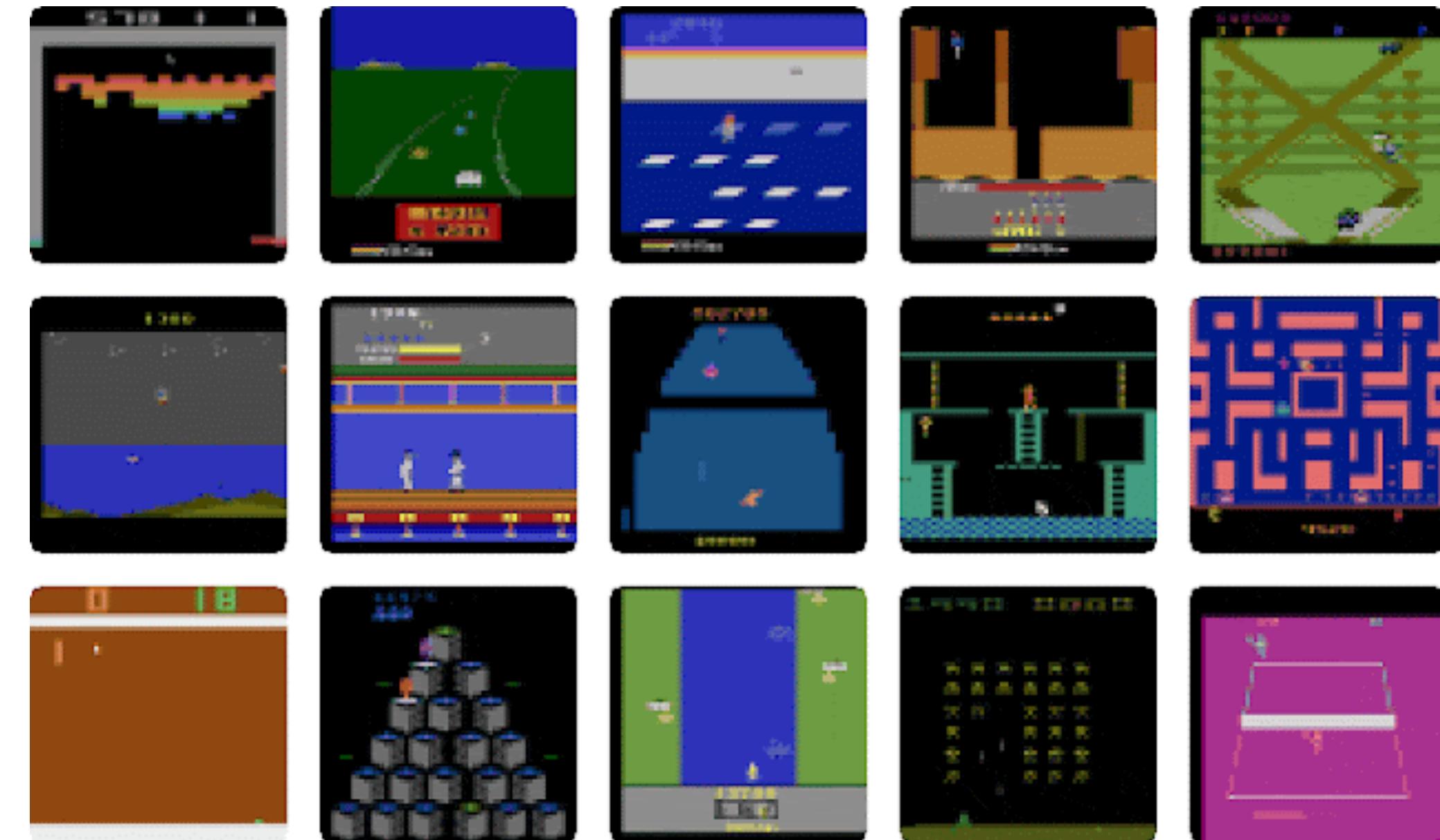
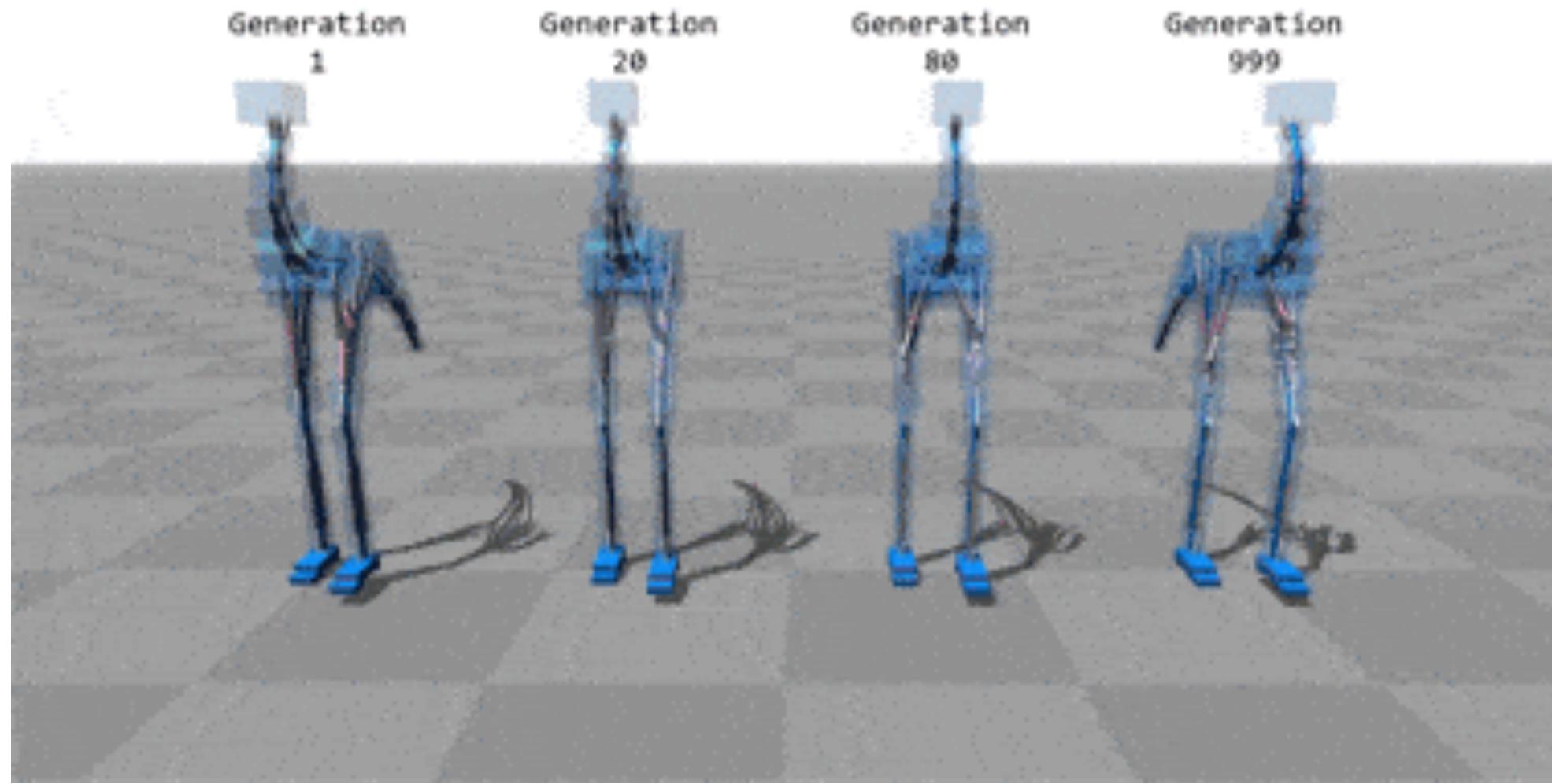
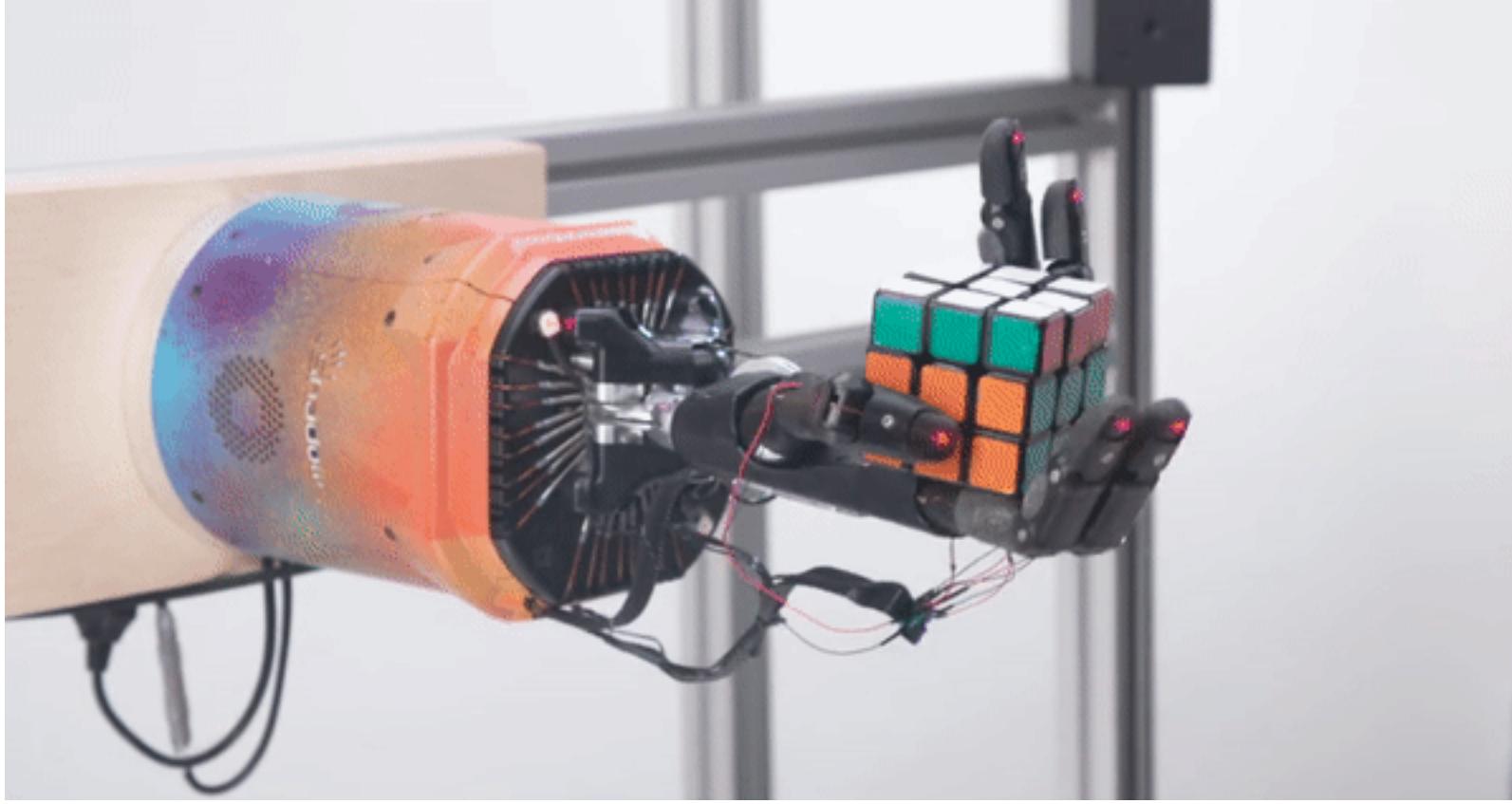
Akkaya, Ilge, et al. "Solving rubik's cube with a robot hand." arXiv preprint arXiv:1910.07113 (2019). (<https://openai.com/research/solving-rubiks-cube>)

Mnih, V., Kavukcuoglu, K., Silver, D. et al. Human-level control through deep reinforcement learning. Nature **518**, 529–533 (2015).

Geijtenbeek, Thomas, Michiel Van De Panne, and A. Frank Van Der Stappen. "Flexible muscle-based locomotion for bipedal creatures." ACM Transactions on Graphics (TOG) 32.6 (2013): 1-11.

Peng, Xue Bin, et al. "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills." ACM Transactions On Graphics (TOG) 37.4 (2018): 1-14.

Reinforcement Learning



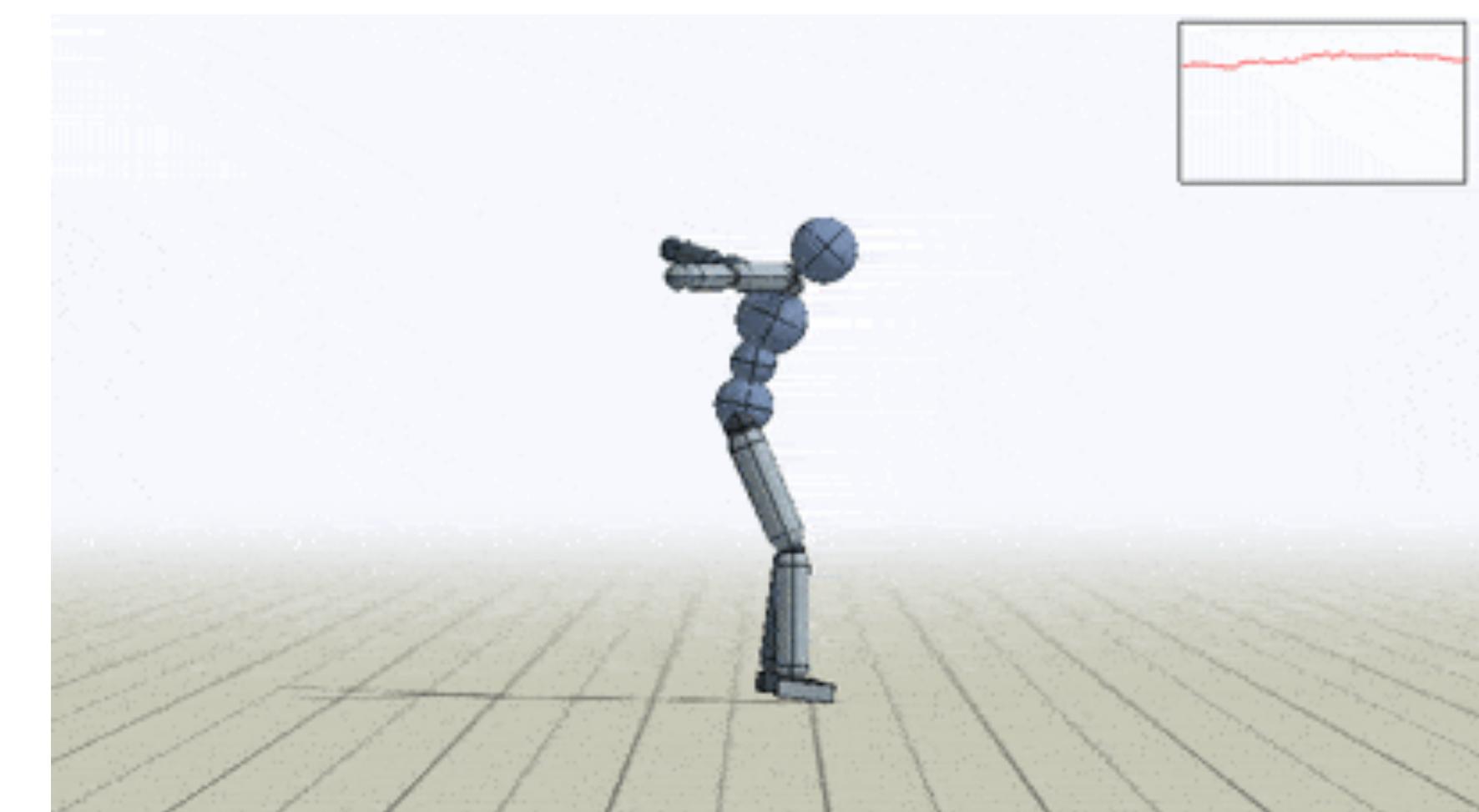
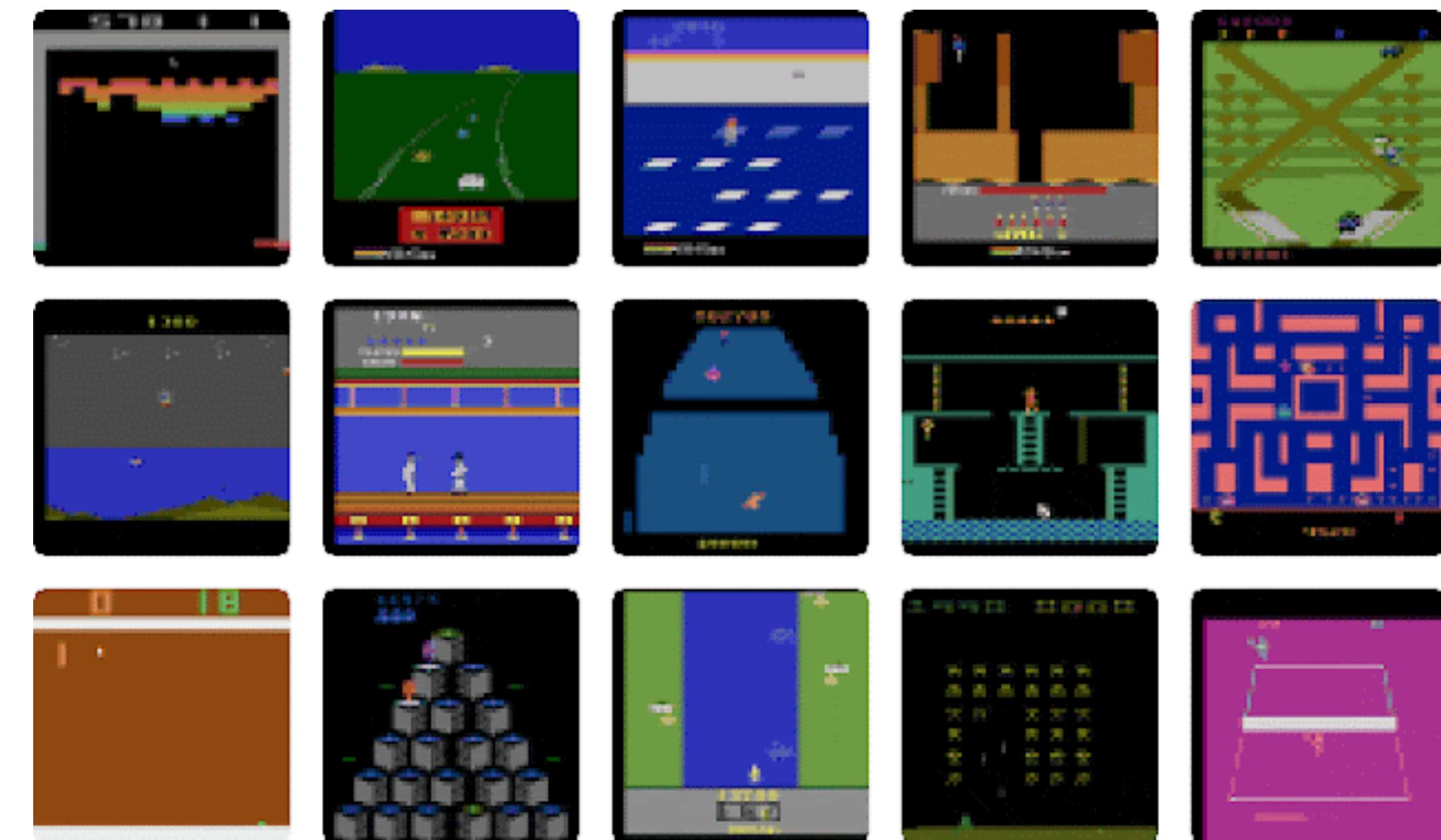
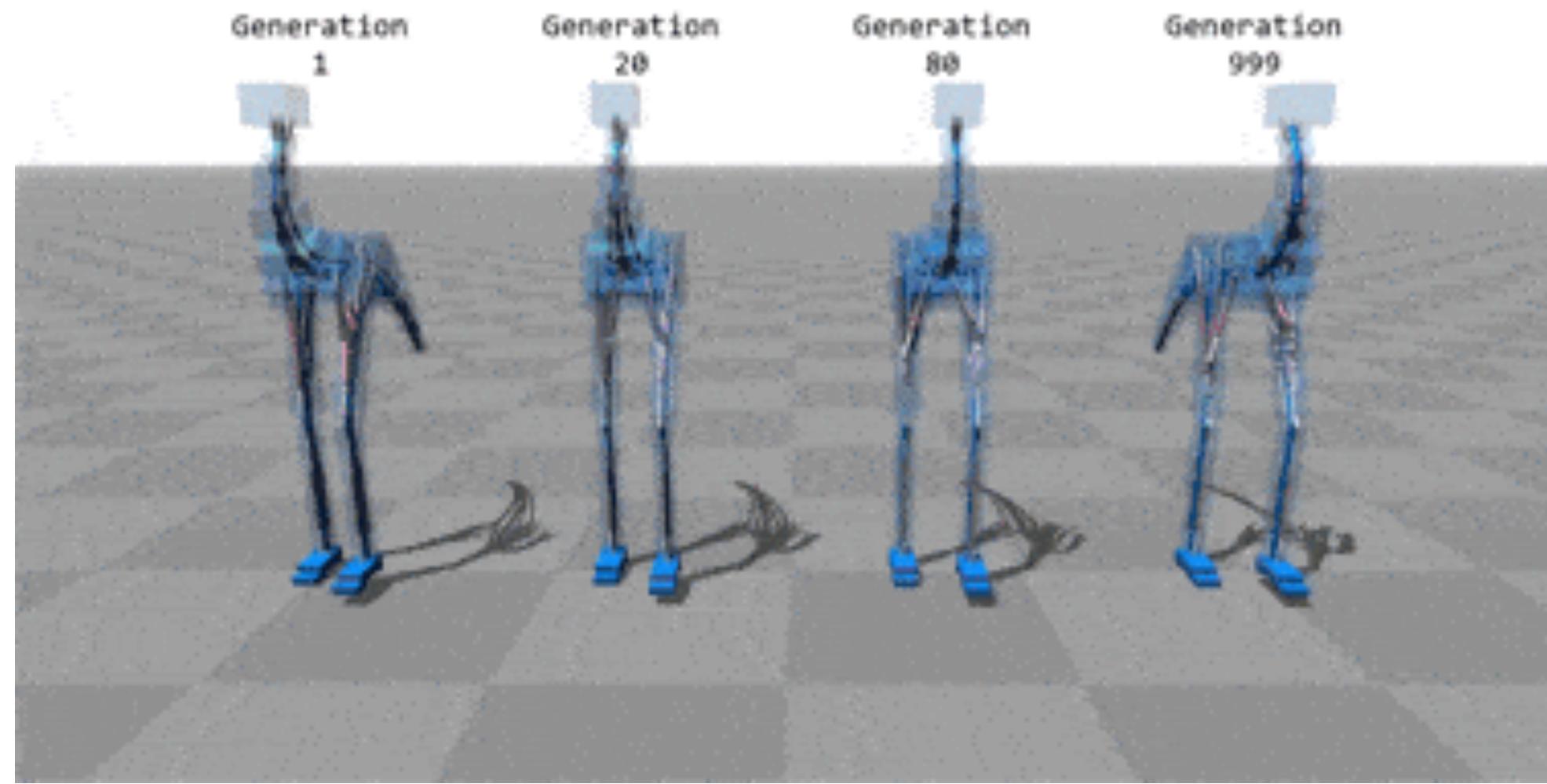
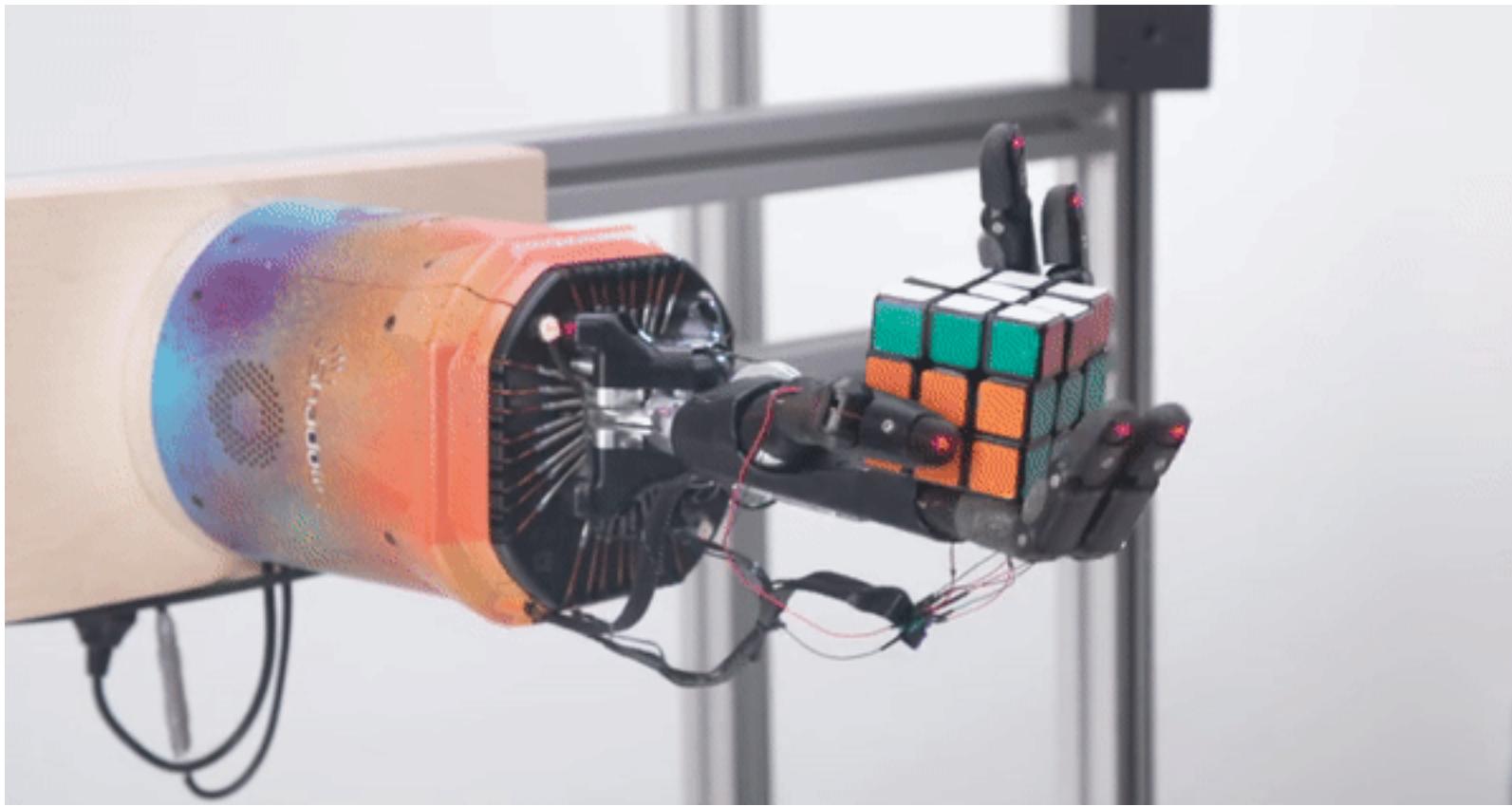
Akkaya, Ilge, et al. "Solving rubik's cube with a robot hand." arXiv preprint arXiv:1910.07113 (2019). (<https://openai.com/research/solving-rubiks-cube>)

Mnih, V., Kavukcuoglu, K., Silver, D. et al. Human-level control through deep reinforcement learning. Nature **518**, 529–533 (2015).

Geijtenbeek, Thomas, Michiel Van De Panne, and A. Frank Van Der Stappen. "Flexible muscle-based locomotion for bipedal creatures." ACM Transactions on Graphics (TOG) 32.6 (2013): 1-11.

Peng, Xue Bin, et al. "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills." ACM Transactions On Graphics (TOG) 37.4 (2018): 1-14.

Reinforcement Learning



Akkaya, Ilge, et al. "Solving rubik's cube with a robot hand." arXiv preprint arXiv:1910.07113 (2019). (<https://openai.com/research/solving-rubiks-cube>)

Mnih, V., Kavukcuoglu, K., Silver, D. et al. Human-level control through deep reinforcement learning. Nature **518**, 529–533 (2015).

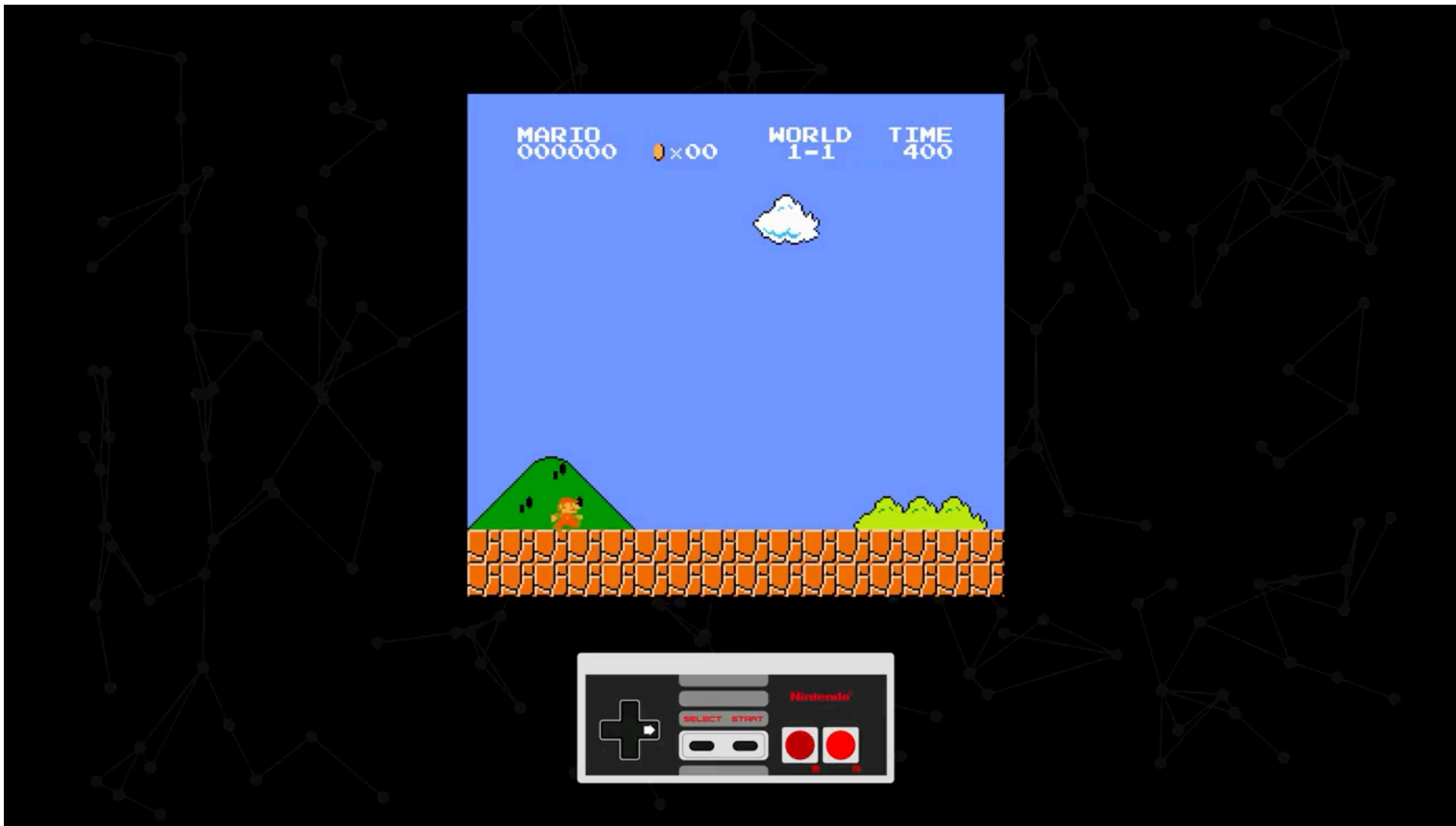
Geijtenbeek, Thomas, Michiel Van De Panne, and A. Frank Van Der Stappen. "Flexible muscle-based locomotion for bipedal creatures." ACM Transactions on Graphics (TOG) 32.6 (2013): 1-11.

Peng, Xue Bin, et al. "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills." ACM Transactions On Graphics (TOG) 37.4 (2018): 1-14.

Reinforcement Learning



Reinforcement Learning



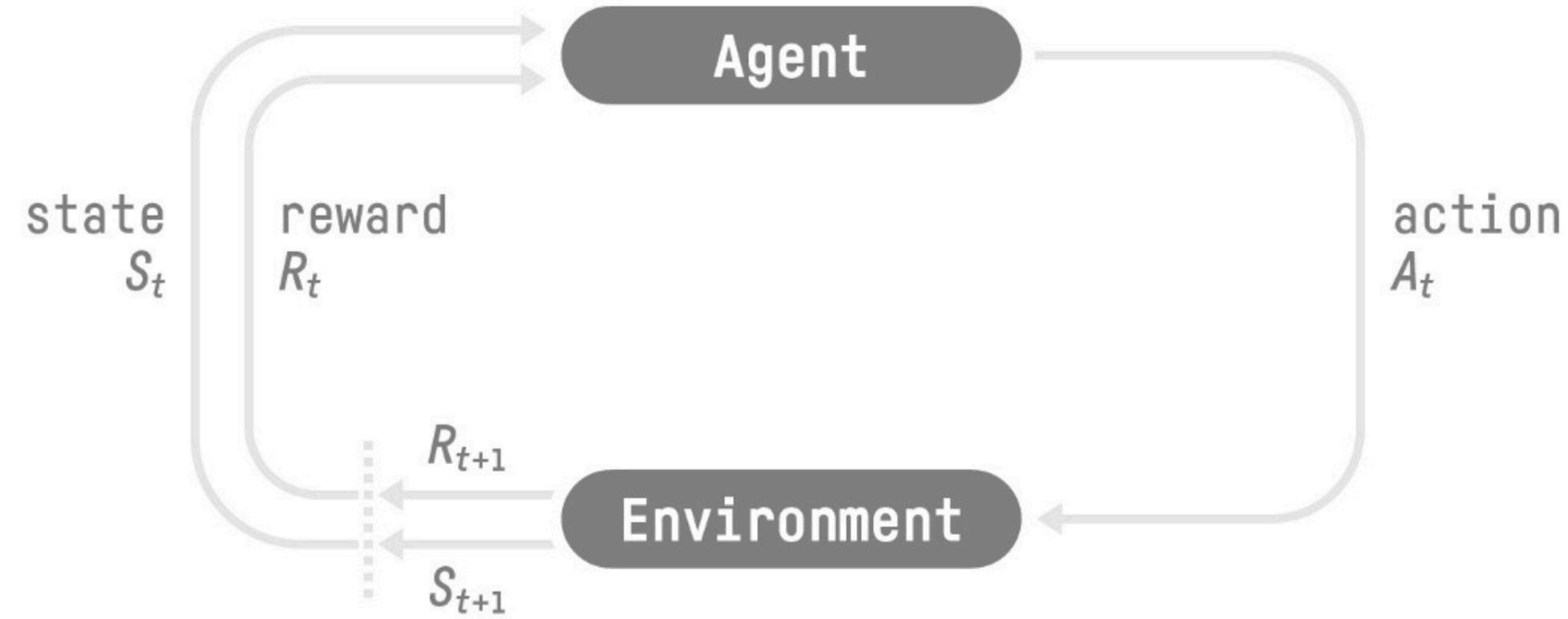
What you will learn

Basic concepts of RL

Q-Learning

Deep Q-Learning

Markov Decision Process (MDP) (finite)



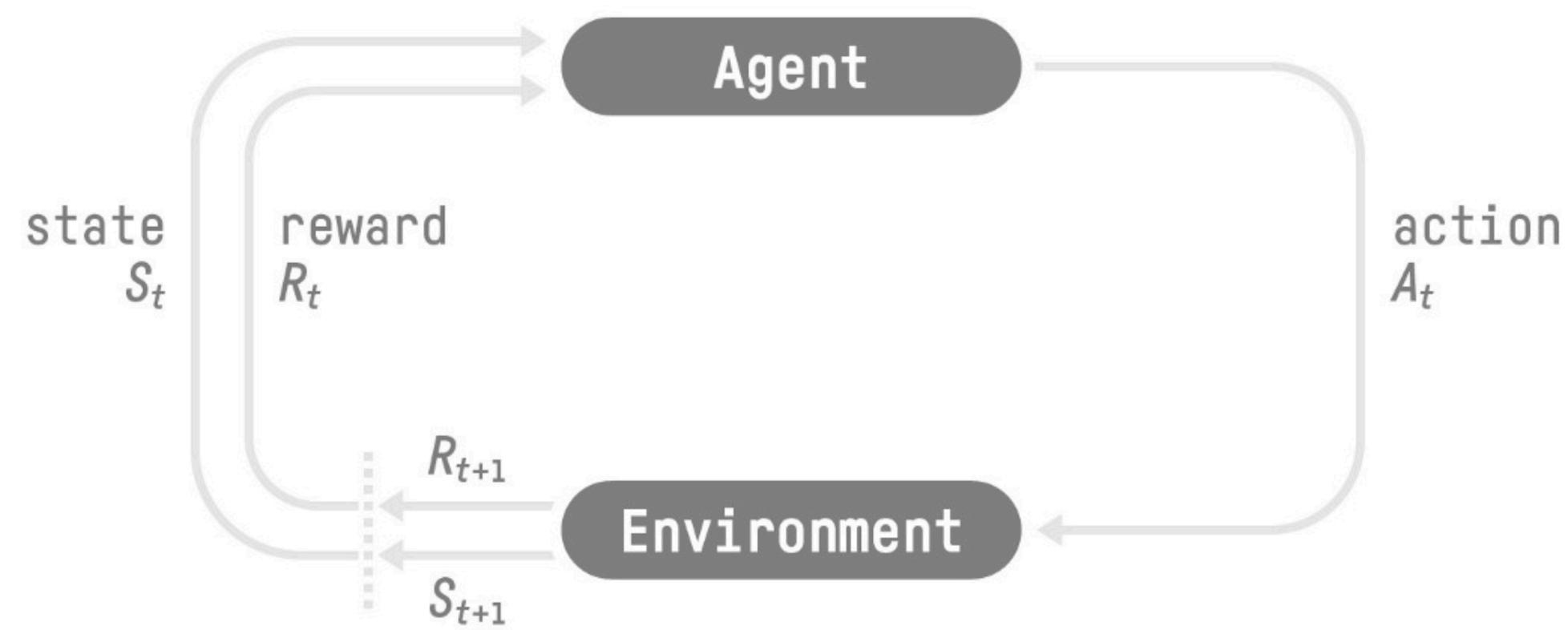
Markov Decision Process (MDP) (finite)



Markov Decision Process (MDP) (finite)



Markov Decision Process (MDP) (finite)

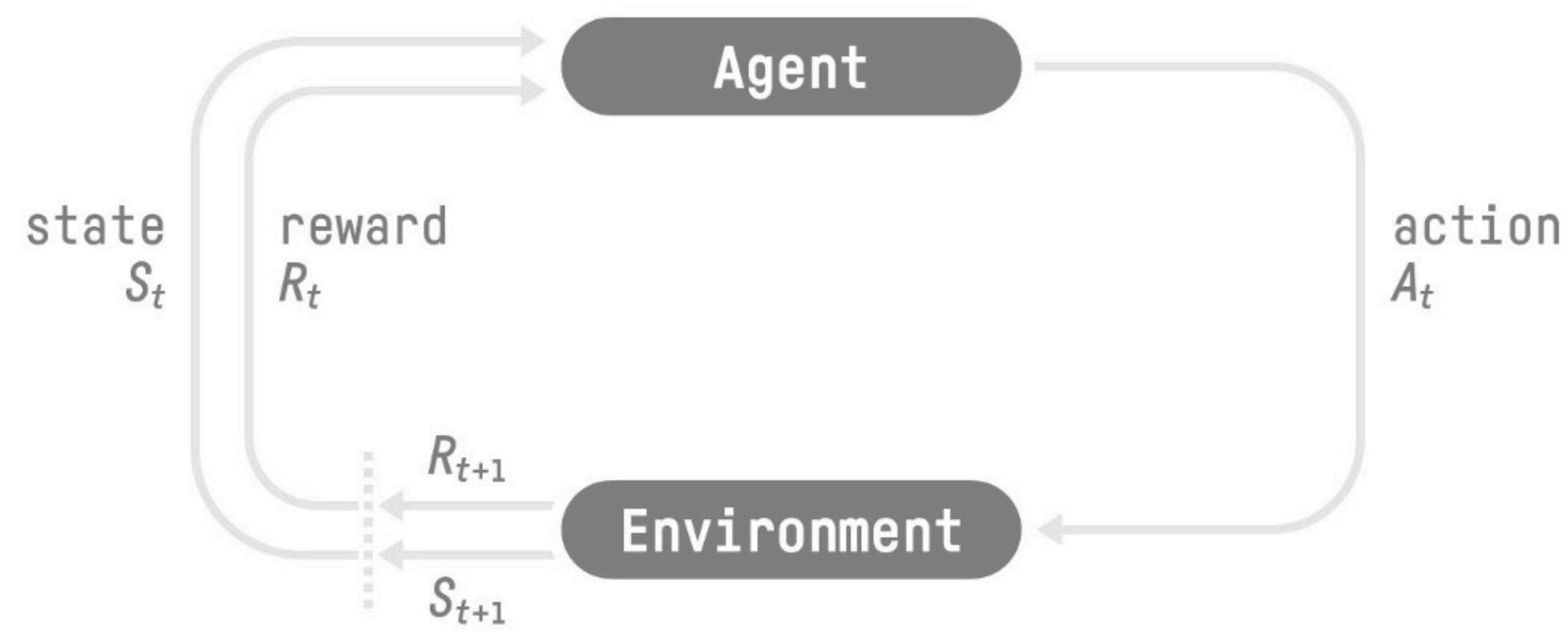


State $s \in \mathcal{S}$

Action $a \in \mathcal{A}$

Reward $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$

Markov Decision Process (MDP) (finite)



State $s \in \mathcal{S}$

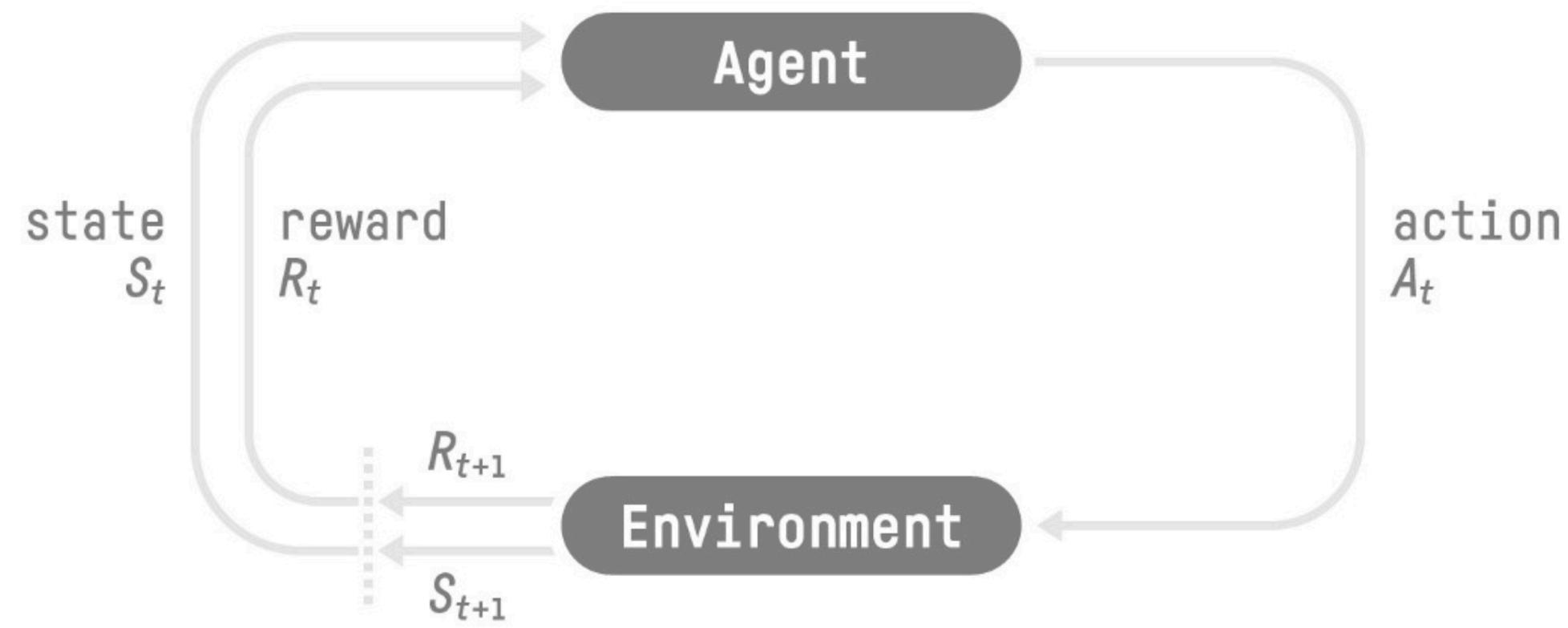
Action $a \in \mathcal{A}$

Reward $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$

Transition function

$p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathcal{A} \rightarrow [0,1]$

Markov Decision Process (MDP) (finite)



State $s \in \mathcal{S}$

Action $a \in \mathcal{A}$

Reward $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$

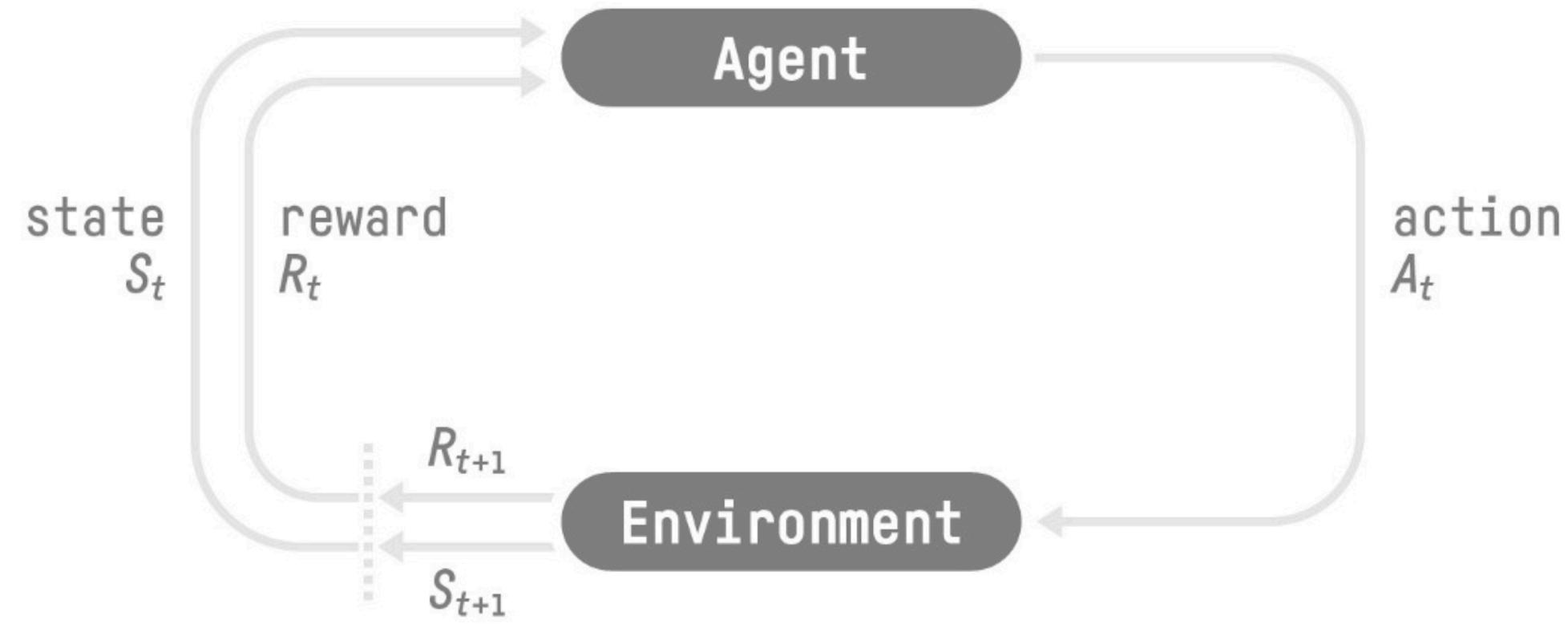
Transition function

$p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathcal{A} \rightarrow [0,1]$

$p(s', r | s, a) =$

$\Pr\{S_t = s', R_t = r | S_{t-1} = s, A_{t-1} = a\}$

Markov Decision Process (MDP) (finite)



State $s \in \mathcal{S}$

Action $a \in \mathcal{A}$

Reward $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$

Transition function

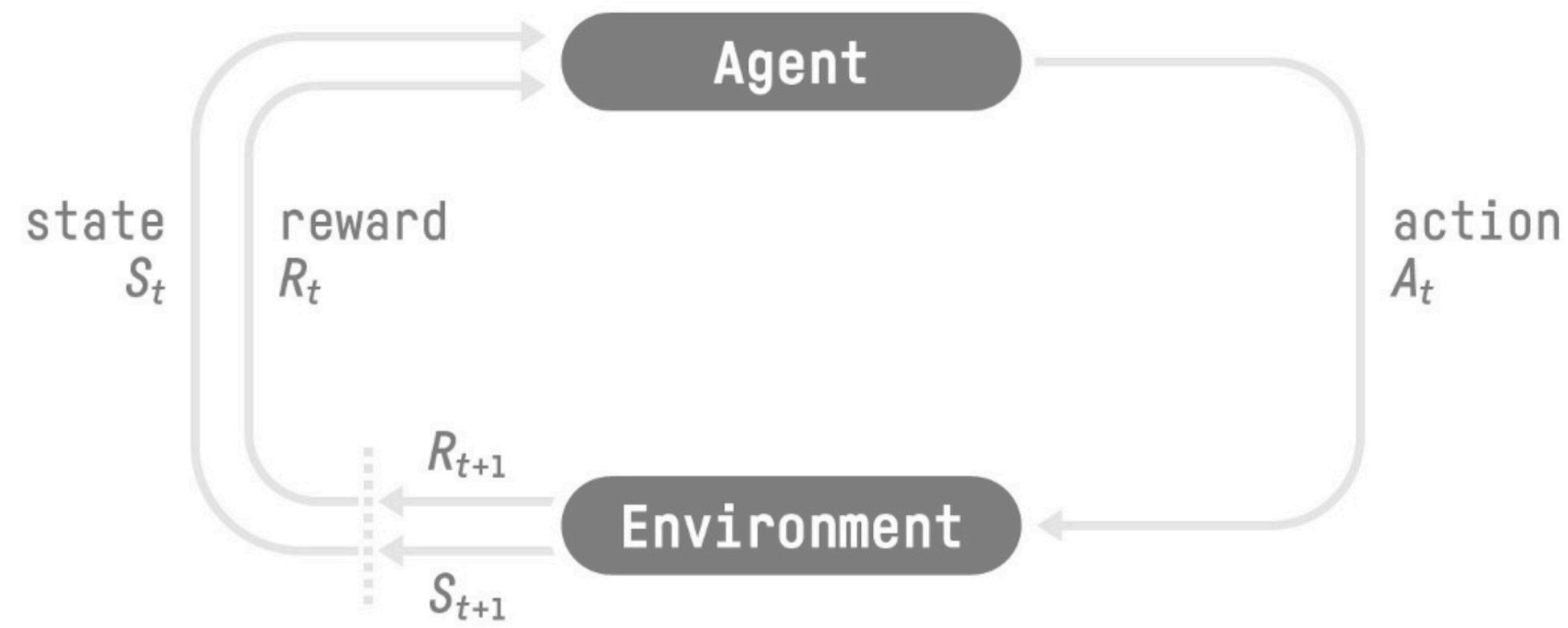
$p : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \rightarrow [0,1]$

$p(s', r | s, a) =$

$\Pr\{S_t = s', R_t = r | S_{t-1} = s, A_{t-1} = a\}$

$$\sum_{s' \in \mathcal{S}} \sum_{r \in R} p(s', r | s, a) = 1, \text{ for all } s \in \mathcal{S}, a \in \mathcal{A}(s)$$

Markov Decision Process (MDP) (finite)



For $t = 0, 1, 2, 3 \dots$

State $s \in \mathcal{S}$

Action $a \in \mathcal{A}$

Reward $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$

Transition function

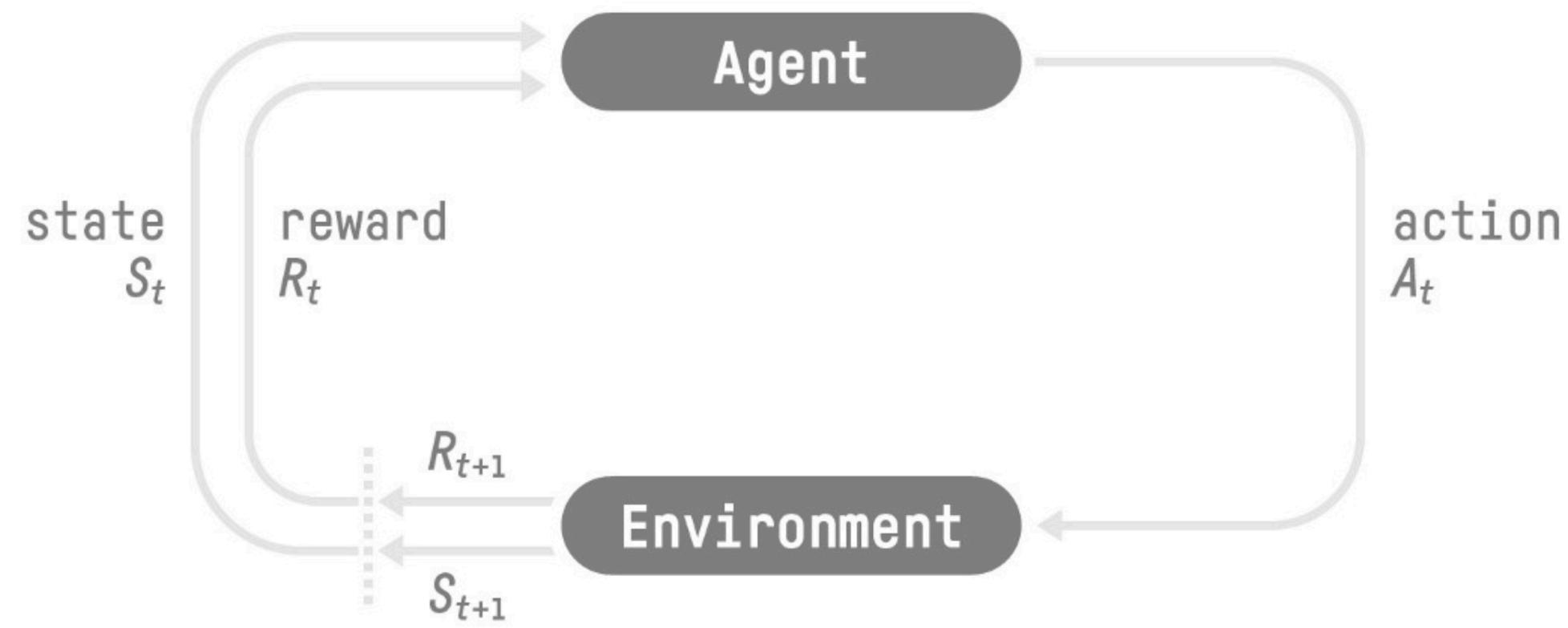
$p : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \rightarrow [0,1]$

$p(s', r | s, a) =$

$\Pr\{S_t = s', R_t = r | S_{t-1} = s, A_{t-1} = a\}$

$$\sum_{s' \in \mathcal{S}} \sum_{r \in R} p(s', r | s, a) = 1, \text{ for all } s \in \mathcal{S}, a \in \mathcal{A}(s)$$

Markov Decision Process (MDP) (finite)



For $t = 0, 1, 2, 3 \dots$

Trajectory: $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \dots$

State $s \in \mathcal{S}$

Action $a \in \mathcal{A}$

Reward $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$

Transition function

$p : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \rightarrow [0,1]$

$p(s', r | s, a) =$

$\Pr\{S_t = s', R_t = r | S_{t-1} = s, A_{t-1} = a\}$

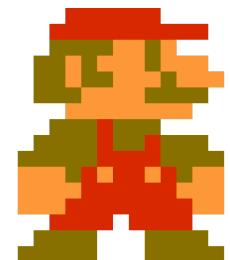
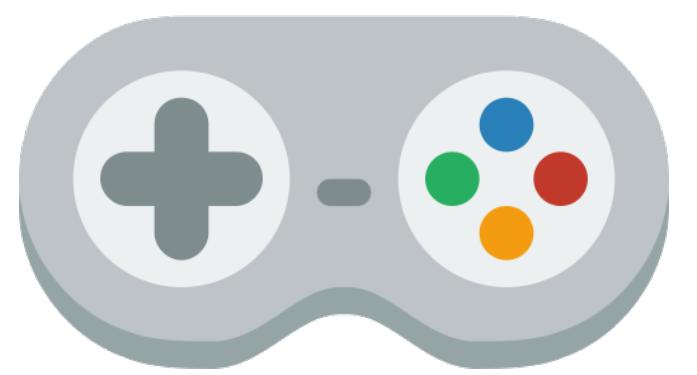
$$\sum_{s' \in \mathcal{S}} \sum_{r \in R} p(s', r | s, a) = 1, \text{ for all } s \in \mathcal{S}, a \in \mathcal{A}(s)$$

Markov Decision Process (MDP)



For $t = 0, 1, 2, 3 \dots$

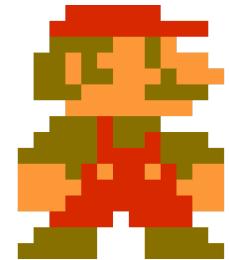
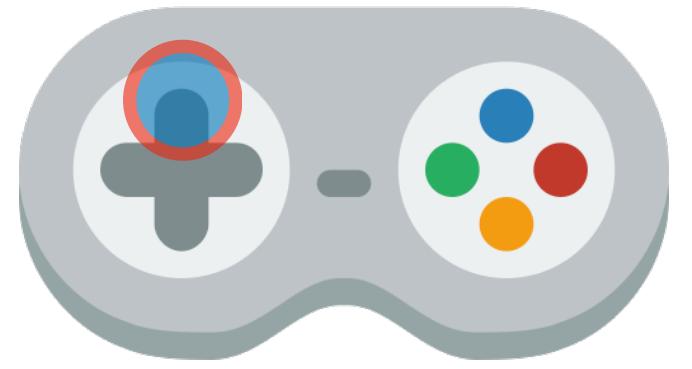
Paradigm



Policy π

A policy defines the learning agent's way of behaving at a given time. Roughly speaking, a policy is a mapping from perceived states of the environment to actions to be taken when in those states. It corresponds to what in psychology would be called a set of stimulus-response rules or associations. In some cases the policy may be a simple function or lookup table, whereas in others it may involve extensive computation such as a search process.

Paradigm

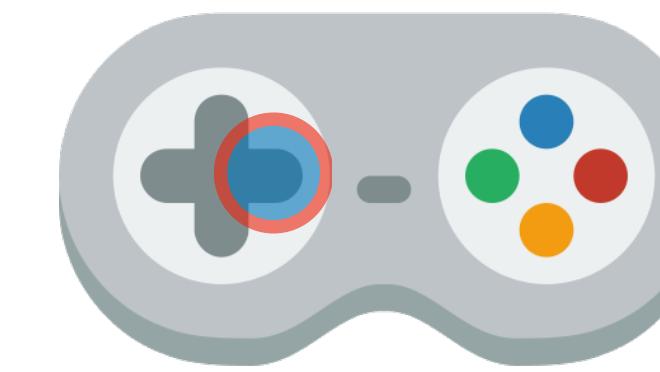
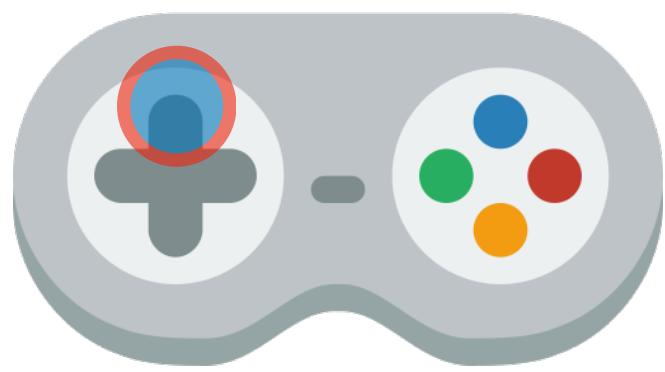


Policy π

A policy defines the learning agent's way of behaving at a given time. Roughly speaking, a policy is a mapping from perceived states of the environment to actions to be taken when in those states. It corresponds to what in psychology would be called a set of stimulus-response rules or associations. In some cases the policy may be a simple function or lookup table, whereas in others it may involve extensive computation such as a search process.

Paradigm

$$r = 0$$



Policy π

A policy defines the learning agent's way of behaving at a given time. Roughly speaking, a policy is a mapping from perceived states of the environment to actions to be taken when in those states. It corresponds to what in psychology would be called a set of stimulus-response rules or associations. In some cases the policy may be a simple function or lookup table, whereas in others it may involve extensive computation such as a search process.

Paradigm



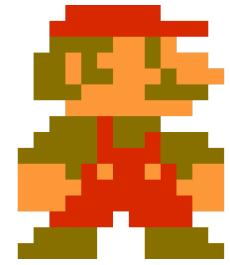
Paradigm

S_0



Paradigm

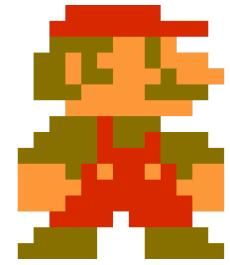
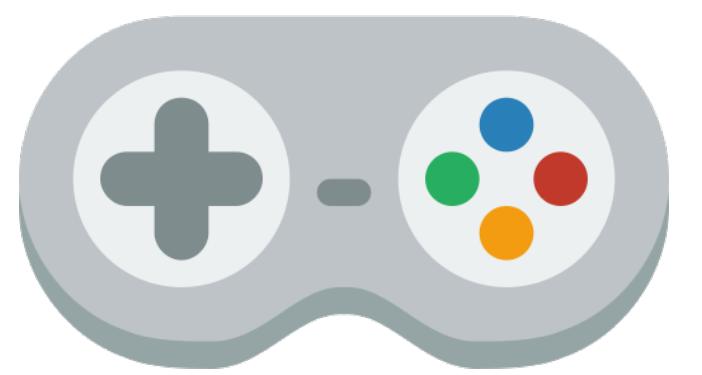
S_0



Policy π

Paradigm

S_0



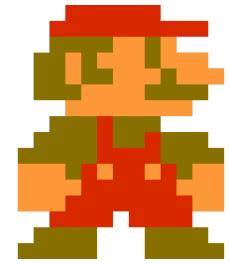
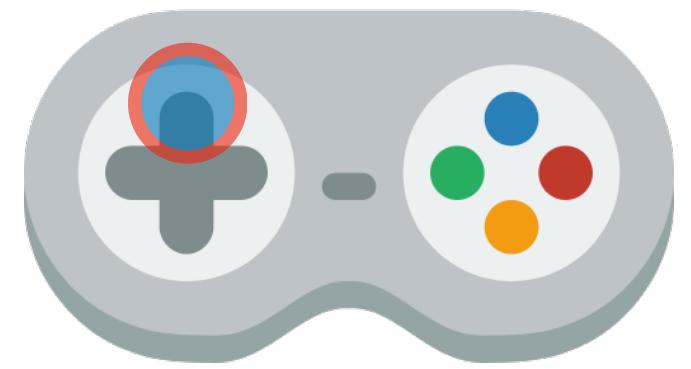
Policy π

Paradigm

S_0



A_0



Policy π

Paradigm

S_0



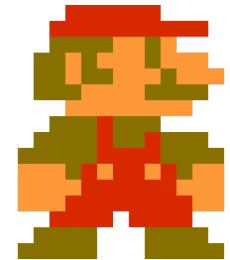
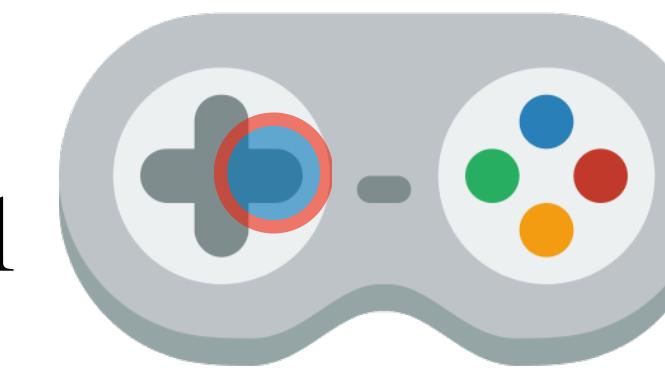
S_1



A_0

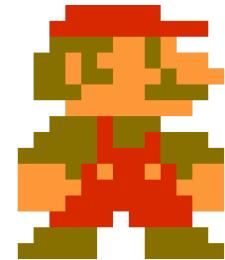
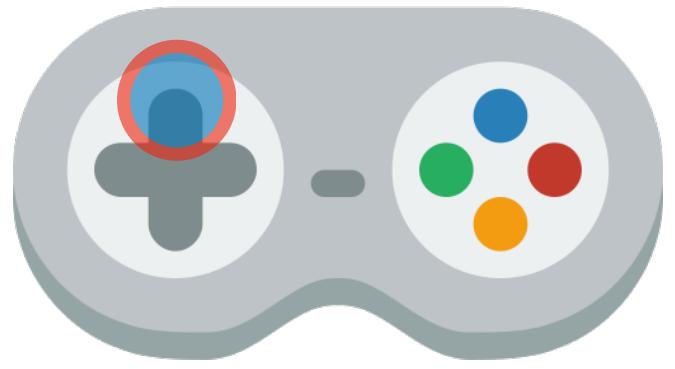


A_1

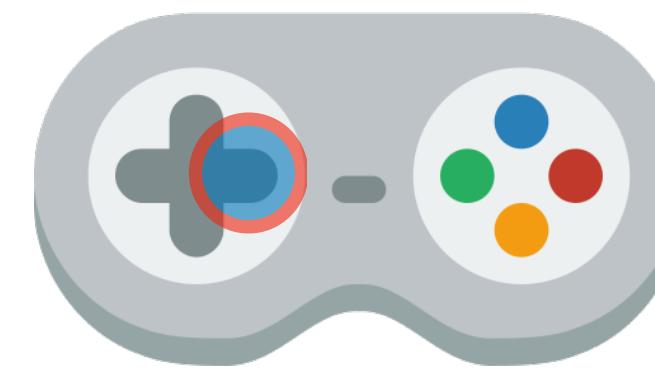


Policy π

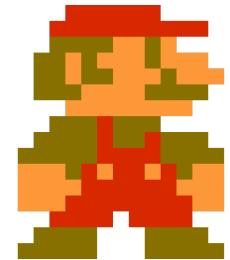
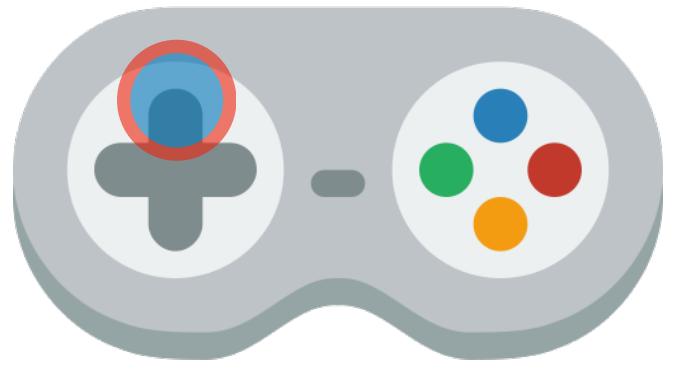
Paradigm

 S_0  S_1  $r_1 = 0$ A_0 

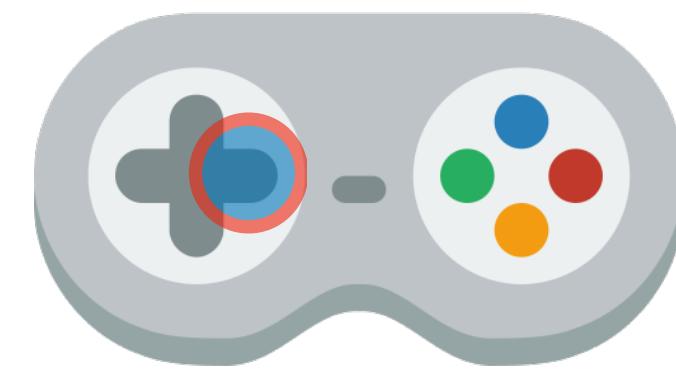
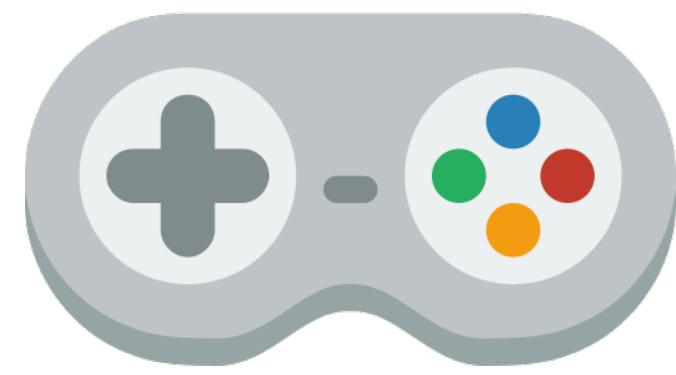
Policy π

 A_1 

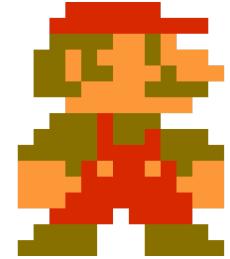
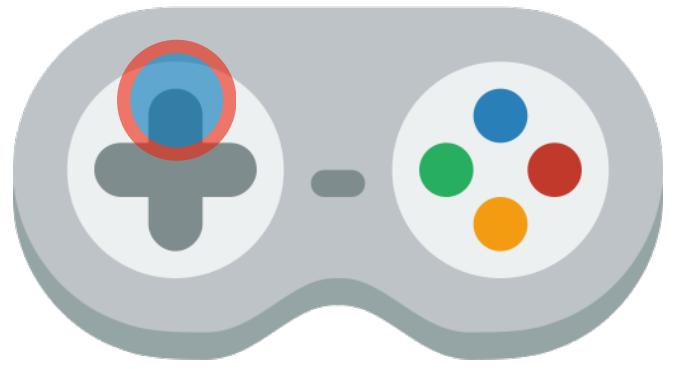
Paradigm

 S_0  S_1  S_2  A_0 

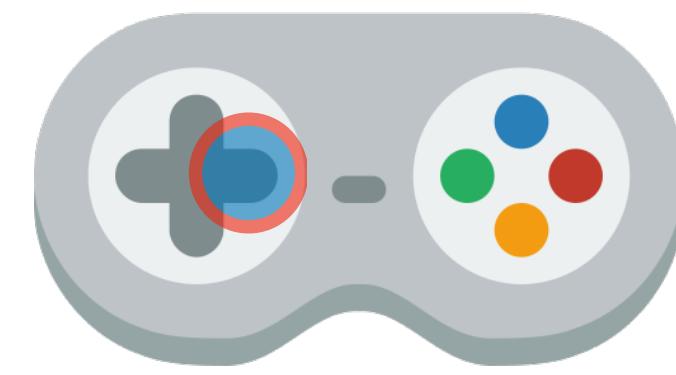
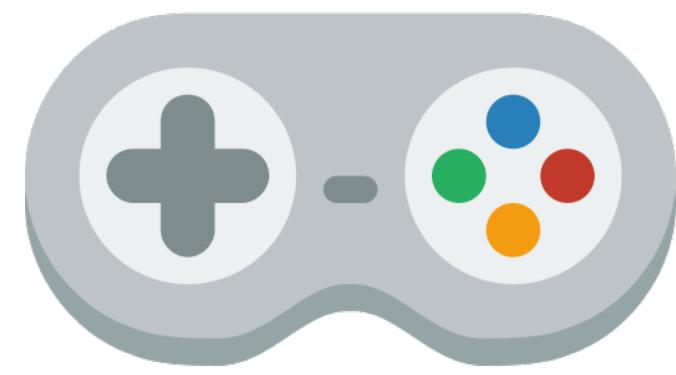
Policy π

 A_1  A_2 

Paradigm

 S_0  S_1  $r_1 = 0$ S_2  $r_2 = 100$ A_0 

Policy π

 A_1  A_2 

Agent Inputs

How do agents get information from the environment?

States

Complete description of the state of the world
(no hidden information)



Observations

Partial description of the state of the world



Agent Outputs

How do agents interact with the environment?

Action space

Agent Outputs

How do agents interact with the environment?

Action space Set of all possible actions in an environment.

Agent Outputs

How do agents interact with the environment?

Action space Set of all possible actions in an environment.

Discrete

Agent Outputs

How do agents interact with the environment?

Action space Set of all possible actions in an environment.

Discrete

Continuous

Agent Outputs

How do agents interact with the environment?

Action space Set of all possible actions in an environment.

Discrete

The number of possible actions is *finite*

Continuous

Agent Outputs

How do agents interact with the environment?

Action space Set of all possible actions in an environment.

Discrete

The number of possible actions is *finite*



Continuous

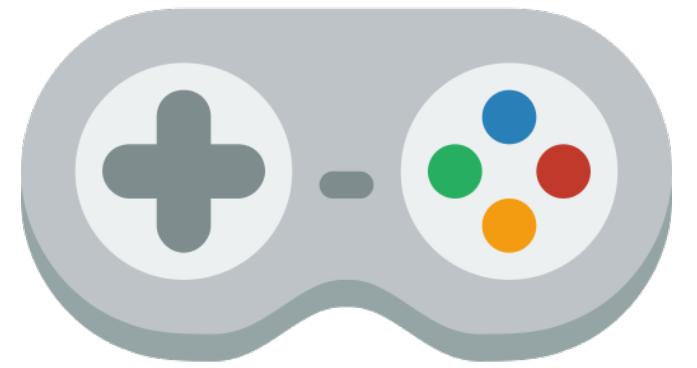
Agent Outputs

How do agents interact with the environment?

Action space Set of all possible actions in an environment.

Discrete

The number of possible actions is *finite*



Continuous

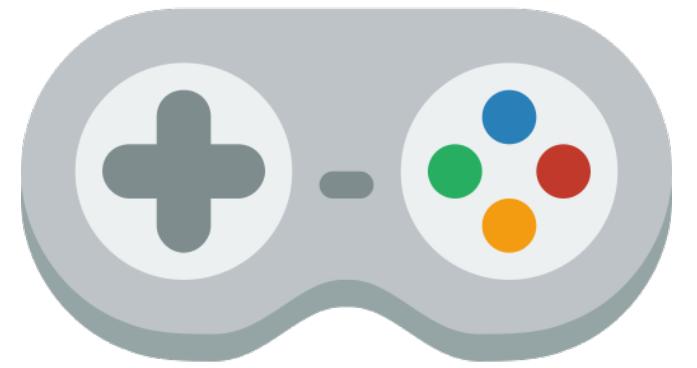
Agent Outputs

How do agents interact with the environment?

Action space Set of all possible actions in an environment.

Discrete

The number of possible actions is *finite*



Continuous

The number of possible actions is *continuous*

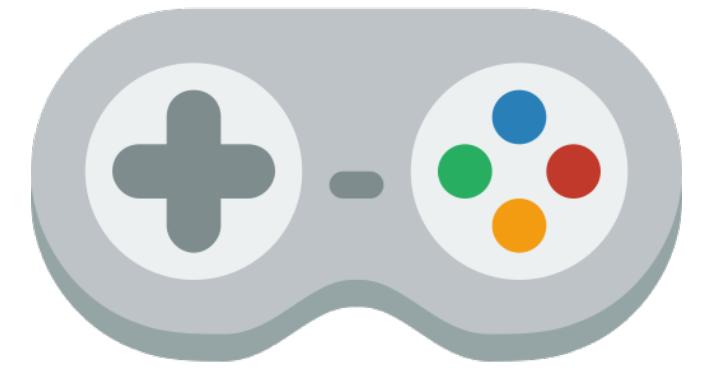
Agent Outputs

How do agents interact with the environment?

Action space Set of all possible actions in an environment.

Discrete

The number of possible actions is *finite*



Continuous

The number of possible actions is *continuous*



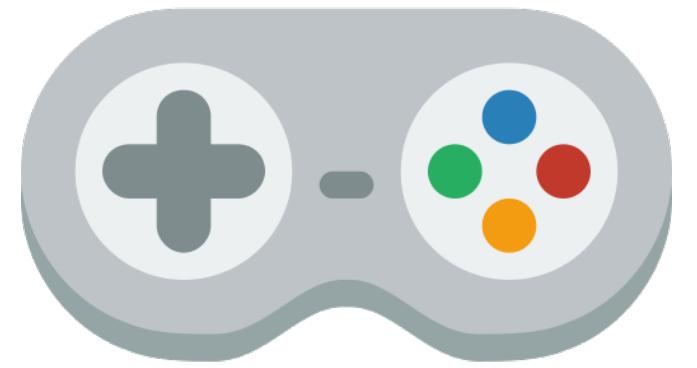
Agent Outputs

How do agents interact with the environment?

Action space Set of all possible actions in an environment.

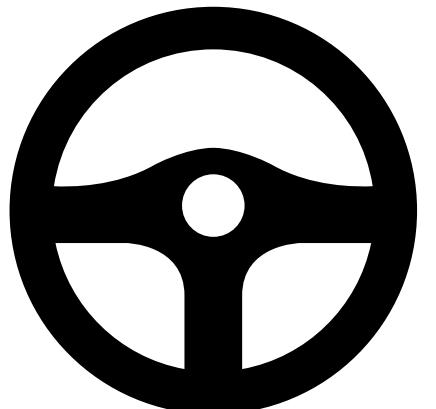
Discrete

The number of possible actions is *finite*



Continuous

The number of possible actions is *continuous*



Goal

Goal

Complete tasks successfully

Goal

Complete tasks successfully

Use rewards

Goal

Complete tasks successfully

Use rewards

Rewards communicate an agent *what* we want to achieve (NOT *how* to do it)

Goal

Complete tasks successfully

Use rewards

Rewards communicate an agent *what* we want to achieve (NOT *how* to do it)

Only feedback the agent receives

Goal

Complete tasks successfully

Use rewards

Rewards communicate an agent *what* we want to achieve (NOT *how* to do it)

Only feedback the agent receives

Maximize total amount of reward received

Goal

Complete tasks successfully

Use rewards

Rewards communicate an agent *what* we want to achieve (NOT *how* to do it)

Only feedback the agent receives

Maximize total amount of reward received

$$R_{t+1} + R_{t+2} + R_{t+3}, \dots + R_T$$

Goal

Complete tasks successfully

Use rewards

Rewards communicate an agent *what* we want to achieve (NOT *how* to do it)

Only feedback the agent receives

Maximize total amount of reward received (*return*)

$$R_{t+1} + R_{t+2} + R_{t+3}, \dots + R_T$$

Goal

Complete tasks successfully

Use rewards

Rewards communicate an agent *what* we want to achieve (NOT *how* to do it)

Only feedback the agent receives

Maximize total amount of reward received (*return*)

$$G_t = R_{t+1} + R_{t+2} + R_{t+3}, \dots + R_T$$

Episodic tasks

Continuing tasks

Episodic tasks

T is the final step. RL breaks into *episodes*.

Continuing tasks

Episodic tasks

T is the final step. RL breaks into *episodes*.

At the end of an episode, final step is reached

Continuing tasks

Episodic tasks

T is the final step. RL breaks into *episodes*.

At the end of an episode, final step is reached

The simulation resets to starting state

Continuing tasks

Episodic tasks

T is the final step. RL breaks into *episodes*.

At the end of an episode, final step is reached

The simulation resets to starting state

Continuing tasks

$T = \infty$, return could be infinite

Episodic tasks

T is the final step. RL breaks into *episodes*.

At the end of an episode, final step is reached

The simulation resets to starting state

Continuing tasks

$T = \infty$, return could be infinite

Use *discount rate* γ for convergence

Episodic tasks

T is the final step. RL breaks into *episodes*.

At the end of an episode, final step is reached

The simulation resets to starting state

Continuing tasks

$T = \infty$, return could be infinite

Use *discount rate* γ for convergence

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots$$

Episodic tasks

T is the final step. RL breaks into *episodes*.

At the end of an episode, final step is reached

The simulation resets to starting state

Continuing tasks

$T = \infty$, return could be infinite

Use *discount rate* γ for convergence

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Episodic tasks

T is the final step. RL breaks into *episodes*.

At the end of an episode, final step is reached

The simulation resets to starting state

Continuing tasks

$T = \infty$, return could be infinite

Use *discount rate* γ for convergence

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

$$0 < \gamma < 1$$

Policies and Value functions

Policies and Value functions

RL involves evaluating value functions

Policies and Value functions

RL involves evaluating value functions

Functions of states or state-action pairs

Policies and Value functions

RL involves evaluating value functions

Functions of states or state-action pairs

- How good is it to be in a certain state (or to take an action in a certain state)

Policies and Value functions

RL involves evaluating value functions

Functions of states or state-action pairs

- How good is it to be in a certain state (or to take an action in a certain state)

Policies π decide actions, maps states to actions

Policies and Value functions

RL involves evaluating value functions

Functions of states or state-action pairs

- How good is it to be in a certain state (or to take an action in a certain state)

Policies π decide actions, maps states to actions

At time t, $\pi(a | s)$ is the probability that $A_t = a$ given $S_t = s$

Policies and Value functions

RL involves evaluating value functions

Functions of states or state-action pairs

- How good is it to be in a certain state (or to take an action in a certain state)

Policies π decide actions, maps states to actions

At time t, $\pi(a | s)$ is the probability that $A_t = a$ given $S_t = s$

RL is about changing the policy as a result of its experience

Policies and Value functions

Policies and Value functions

Value function of state s under policy π

Expected return when starting in state s and following policy π

Policies and Value functions

Value function of state s under policy π

Expected return when starting in state s and following policy π

State-value function for policy π

Policies and Value functions

Value function of state s under policy π

Expected return when starting in state s and following policy π

State-value function for policy π

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t | S_t = s] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s \right], \quad \text{for all } s \in \mathcal{S}$$

Policies and Value functions

Value function of state s under policy π

Expected return when starting in state s and following policy π

State-value function for policy π

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t | S_t = s] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s \right], \quad \text{for all } s \in \mathcal{S}$$

Action-value function for policy π

Policies and Value functions

Value function of state s under policy π

Expected return when starting in state s and following policy π

State-value function for policy π

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t | S_t = s] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s \right], \quad \text{for all } s \in \mathcal{S}$$

Action-value function for policy π

Expected return when starting in state s and taking action a , according to policy π

Policies and Value functions

Value function of state s under policy π

Expected return when starting in state s and following policy π

State-value function for policy π

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t | S_t = s] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s \right], \quad \text{for all } s \in \mathcal{S}$$

Action-value function for policy π

Expected return when starting in state s and taking action a , according to policy π

$$q_\pi(s, a) \doteq \mathbb{E}_\pi[G_t | S_t = s, A_t = a] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a \right], \quad \text{for all } s \in \mathcal{S}$$

Policies and Value functions

Value function of state s under policy π

Expected return when starting in state s and following policy π

State-value function for policy π

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t | S_t = s] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s \right], \quad \text{for all } s \in \mathcal{S}$$

Action-value function for policy π

Expected return when starting in state s and taking action a , according to policy π

$$q_\pi(s, a) \doteq \mathbb{E}_\pi[G_t | S_t = s, A_t = a] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a \right], \quad \text{for all } s \in \mathcal{S}$$

v_π and q_π can be estimated from experience

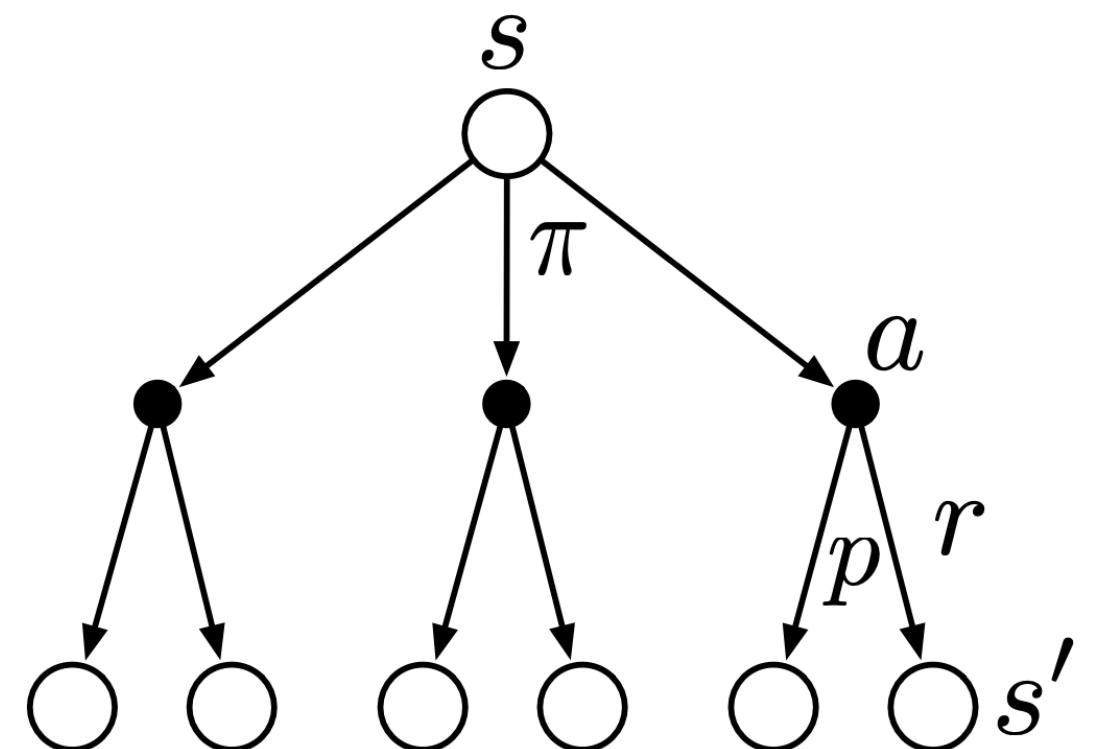
Bellman Equation for v_π

State-value function for policy π

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s]$$

$$= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s]$$

$$= \sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a) [r + \gamma v_\pi(s')], \quad \text{for all } s \in \mathcal{S}$$



Backup diagram

Monte Carlo methods

Monte Carlo methods

Monte Carlo method for prediction and control

Monte Carlo methods

Monte Carlo method for prediction and control

Often used for estimation that relies on repeated random sampling

Monte Carlo methods

Monte Carlo method for prediction and control

Often used for estimation that relies on repeated random sampling

Monte Carlo in RL

Monte Carlo methods

Monte Carlo method for prediction and control

Often used for estimation that relies on repeated random sampling

Monte Carlo in RL

Estimate values directly from experience (sequence of S, A, R)

Monte Carlo methods

Monte Carlo method for prediction and control

Often used for estimation that relies on repeated random sampling

Monte Carlo in RL

Estimate values directly from experience (sequence of S, A, R)

Learning from experience

Monte Carlo methods

Monte Carlo method for prediction and control

Often used for estimation that relies on repeated random sampling

Monte Carlo in RL

Estimate values directly from experience (sequence of S, A, R)

Learning from experience

Agent can accurately estimate a value function without prior knowledge of environment dynamics

Monte Carlo methods

Monte Carlo method for prediction and control

Often used for estimation that relies on repeated random sampling

Monte Carlo in RL

Estimate values directly from experience (sequence of S, A, R)

Learning from experience

Agent can accurately estimate a value function without prior knowledge of environment dynamics

No need to know transition probability p

Monte Carlo methods

Monte Carlo method for prediction and control

Often used for estimation that relies on repeated random sampling

Monte Carlo in RL

Estimate values directly from experience (sequence of S, A, R)

Learning from experience

Agent can accurately estimate a value function without prior knowledge of environment dynamics

No need to know transition probability p

We can use it to estimate $v_\pi(s)$

Monte Carlo Prediction

Monte Carlo Prediction

We want to estimate $v_\pi(s)$

Monte Carlo Prediction

We want to estimate $v_\pi(s)$

First-visit MC prediction, for estimating $V \approx v_\pi$

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$

$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Loop forever (for each episode):

Generate an episode following π : $S_0, A_0, R_1, S_1, A_1, R_2, \dots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \dots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless S_t appears in S_0, S_1, \dots, S_{t-1} :

Append G to $Returns(S_t)$

$V(S_t) \leftarrow \text{average}(Returns(S_t))$

Monte Carlo Prediction

We want to estimate $v_{\pi}(s)$

First-visit MC prediction, for estimating $V \approx v_{\pi}$

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$

$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Loop forever (for each episode):

Generate an episode following π : $S_0, A_0, R_1, S_1, A_1, R_2, \dots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

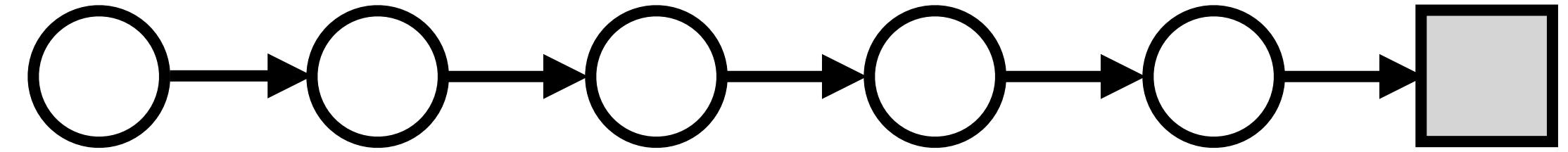
Loop for each step of episode, $t = T-1, T-2, \dots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless S_t appears in S_0, S_1, \dots, S_{t-1} :

Append G to $Returns(S_t)$

$V(S_t) \leftarrow \text{average}(Returns(S_t))$



Monte Carlo Prediction

We want to estimate $v_{\pi}(s)$

First-visit MC prediction, for estimating $V \approx v_{\pi}$

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$

$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Loop forever (for each episode):

Generate an episode following π : $S_0, A_0, R_1, S_1, A_1, R_2, \dots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

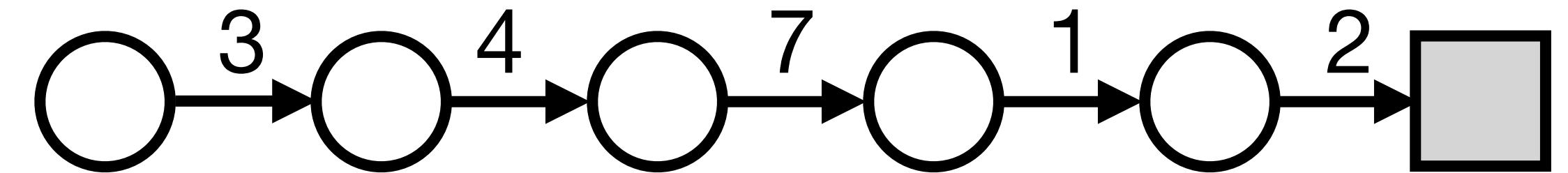
Loop for each step of episode, $t = T-1, T-2, \dots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless S_t appears in S_0, S_1, \dots, S_{t-1} :

Append G to $Returns(S_t)$

$V(S_t) \leftarrow \text{average}(Returns(S_t))$



Monte Carlo Prediction

We want to estimate $v_{\pi}(s)$

First-visit MC prediction, for estimating $V \approx v_{\pi}$

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$

$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Loop forever (for each episode):

Generate an episode following π : $S_0, A_0, R_1, S_1, A_1, R_2, \dots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

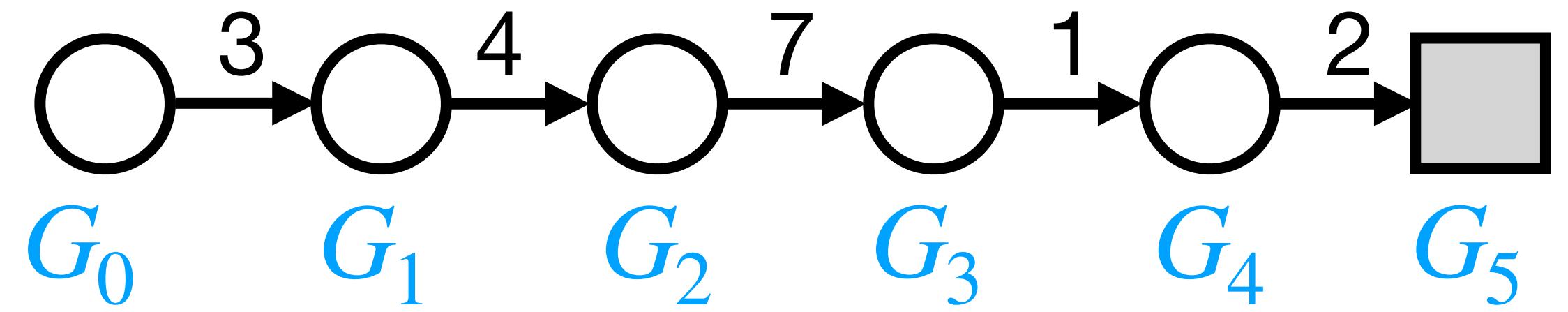
Loop for each step of episode, $t = T-1, T-2, \dots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless S_t appears in S_0, S_1, \dots, S_{t-1} :

Append G to $Returns(S_t)$

$V(S_t) \leftarrow \text{average}(Returns(S_t))$



Monte Carlo Prediction

We want to estimate $v_{\pi}(s)$

First-visit MC prediction, for estimating $V \approx v_{\pi}$

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$

$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Loop forever (for each episode):

Generate an episode following π : $S_0, A_0, R_1, S_1, A_1, R_2, \dots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \dots, 0$:

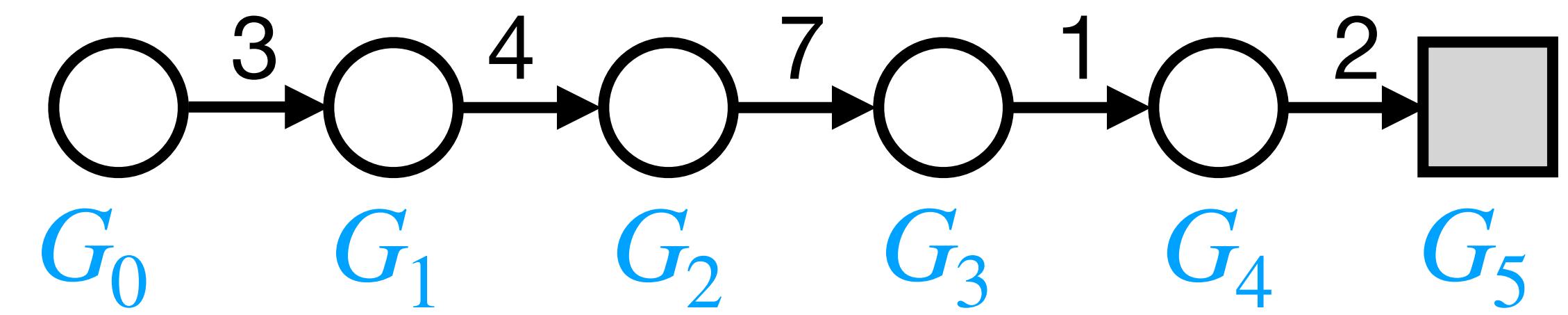
$G \leftarrow \gamma G + R_{t+1}$

Unless S_t appears in S_0, S_1, \dots, S_{t-1} :

Append G to $Returns(S_t)$

$V(S_t) \leftarrow \text{average}(Returns(S_t))$

$$\gamma = 0.5$$



Monte Carlo Prediction

We want to estimate $v_{\pi}(s)$

First-visit MC prediction, for estimating $V \approx v_{\pi}$

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$

$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Loop forever (for each episode):

Generate an episode following π : $S_0, A_0, R_1, S_1, A_1, R_2, \dots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \dots, 0$:

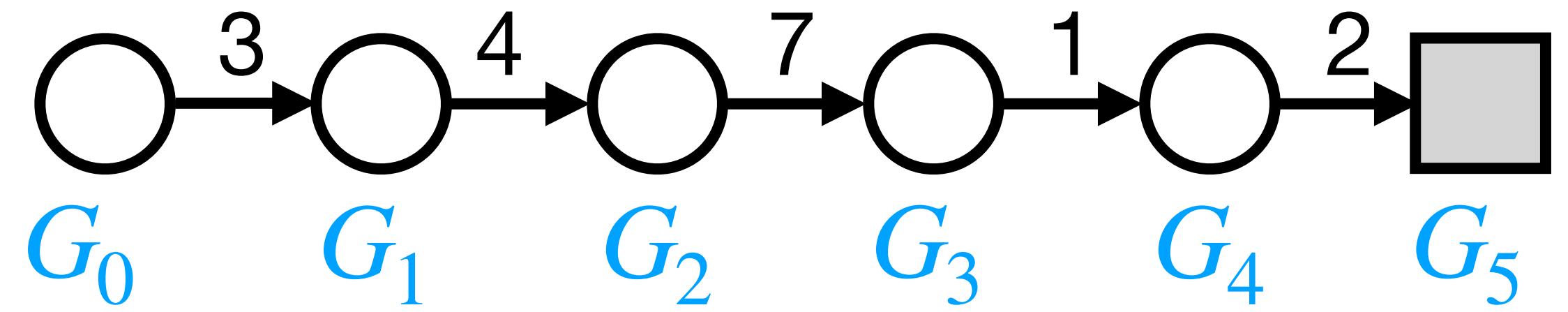
$G \leftarrow \gamma G + R_{t+1}$

Unless S_t appears in S_0, S_1, \dots, S_{t-1} :

Append G to $Returns(S_t)$

$V(S_t) \leftarrow \text{average}(Returns(S_t))$

$$\gamma = 0.5$$



Monte Carlo Prediction

We want to estimate $v_{\pi}(s)$

First-visit MC prediction, for estimating $V \approx v_{\pi}$

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$

$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Loop forever (for each episode):

Generate an episode following π : $S_0, A_0, R_1, S_1, A_1, R_2, \dots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \dots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

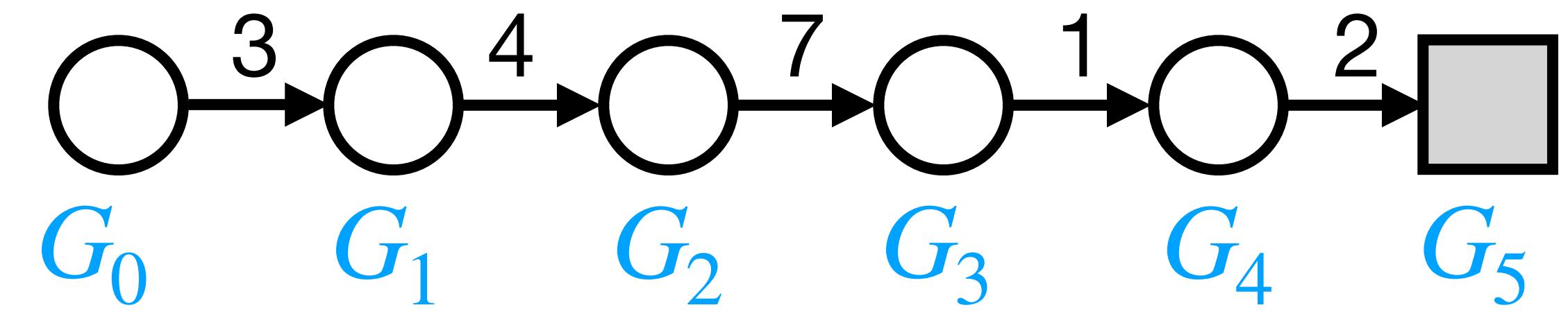
Unless S_t appears in S_0, S_1, \dots, S_{t-1} :

Append G to $Returns(S_t)$

$V(S_t) \leftarrow \text{average}(Returns(S_t))$

$$G_4 = R_5 = 2$$

$$\gamma = 0.5$$



Monte Carlo Prediction

We want to estimate $v_{\pi}(s)$

First-visit MC prediction, for estimating $V \approx v_{\pi}$

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$

$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Loop forever (for each episode):

Generate an episode following π : $S_0, A_0, R_1, S_1, A_1, R_2, \dots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \dots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless S_t appears in S_0, S_1, \dots, S_{t-1} :

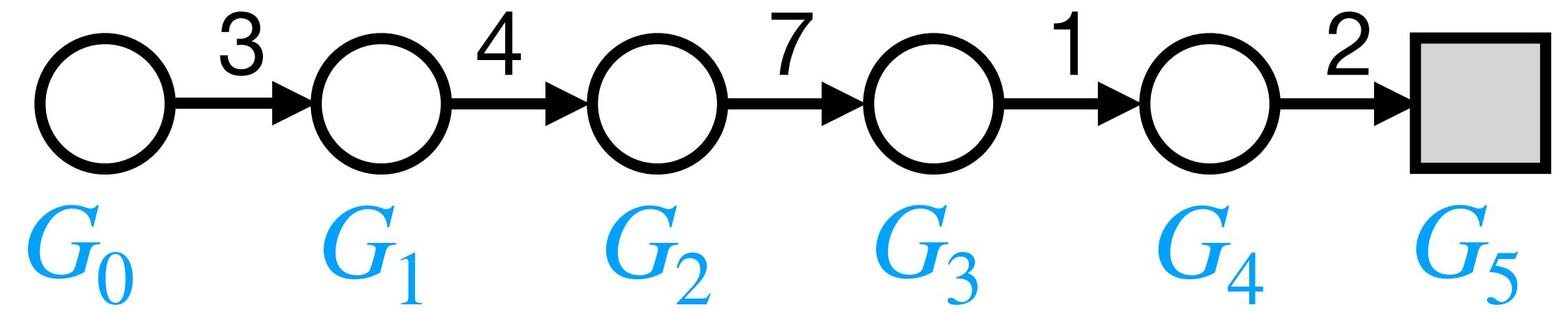
Append G to $Returns(S_t)$

$V(S_t) \leftarrow \text{average}(Returns(S_t))$

$$G_4 = R_5 = 2$$

$$G_3 = R_4 + \gamma G_4 = 1 + (0.5 * 2) = 2$$

$$\gamma = 0.5$$



Monte Carlo Prediction

We want to estimate $v_{\pi}(s)$

First-visit MC prediction, for estimating $V \approx v_{\pi}$

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$

$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Loop forever (for each episode):

Generate an episode following π : $S_0, A_0, R_1, S_1, A_1, R_2, \dots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \dots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless S_t appears in S_0, S_1, \dots, S_{t-1} :

Append G to $Returns(S_t)$

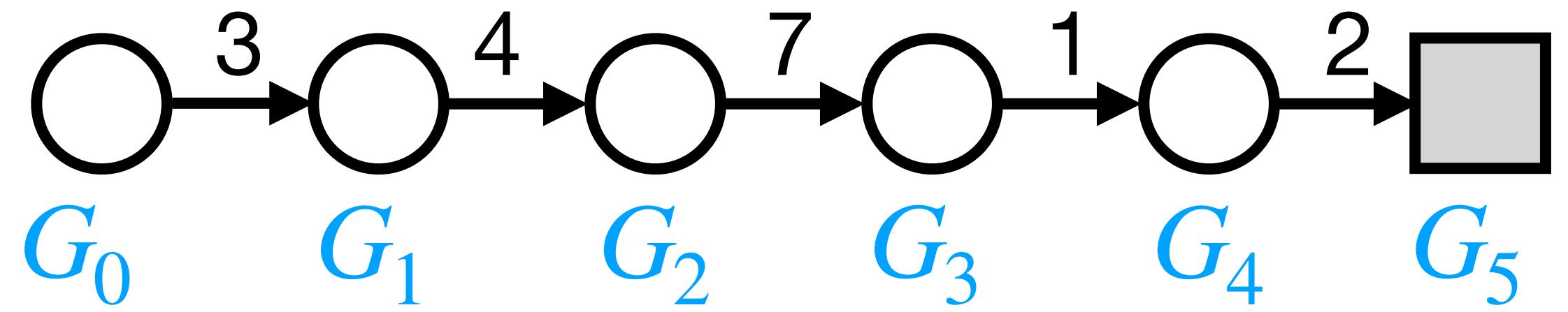
$V(S_t) \leftarrow \text{average}(Returns(S_t))$

$$G_4 = R_5 = 2$$

$$G_3 = R_4 + \gamma G_4 = 1 + (0.5 * 2) = 2$$

$$G_2 = R_3 + \gamma G_3 = 7 + (0.5 * 2) = 8$$

$$\gamma = 0.5$$



Monte Carlo Prediction

We want to estimate $v_{\pi}(s)$

First-visit MC prediction, for estimating $V \approx v_{\pi}$

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$

$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Loop forever (for each episode):

Generate an episode following π : $S_0, A_0, R_1, S_1, A_1, R_2, \dots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \dots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless S_t appears in S_0, S_1, \dots, S_{t-1} :

Append G to $Returns(S_t)$

$V(S_t) \leftarrow \text{average}(Returns(S_t))$

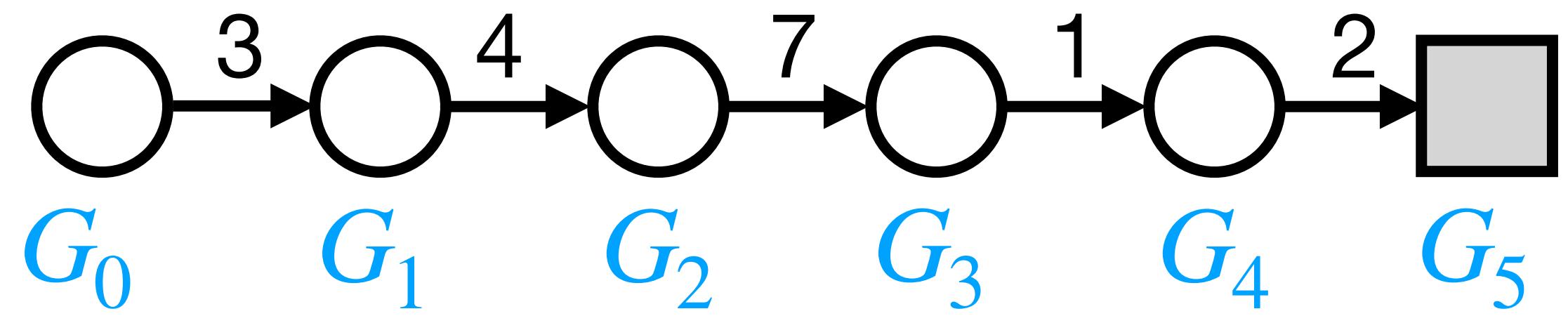
$$G_4 = R_5 = 2$$

$$G_3 = R_4 + \gamma G_4 = 1 + (0.5 * 2) = 2$$

$$G_2 = R_3 + \gamma G_3 = 7 + (0.5 * 2) = 8$$

$$G_1 = R_2 + \gamma G_4 = 4 + (0.5 * 8) = 8$$

$$\gamma = 0.5$$



Monte Carlo Prediction

We want to estimate $v_{\pi}(s)$

First-visit MC prediction, for estimating $V \approx v_{\pi}$

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$

$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Loop forever (for each episode):

Generate an episode following π : $S_0, A_0, R_1, S_1, A_1, R_2, \dots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \dots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless S_t appears in S_0, S_1, \dots, S_{t-1} :

Append G to $Returns(S_t)$

$V(S_t) \leftarrow \text{average}(Returns(S_t))$

$$G_4 = R_5 = 2$$

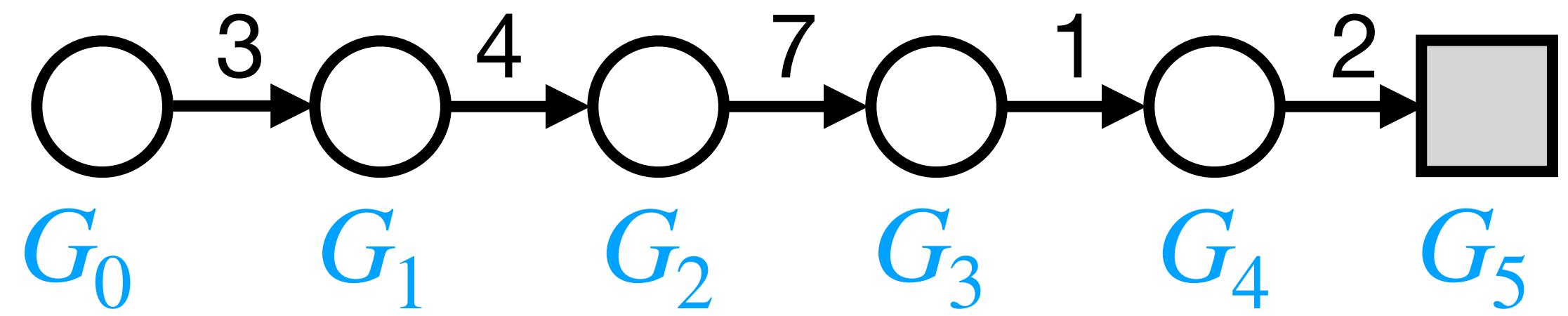
$$G_3 = R_4 + \gamma G_4 = 1 + (0.5 * 2) = 2$$

$$G_2 = R_3 + \gamma G_3 = 7 + (0.5 * 2) = 8$$

$$G_1 = R_2 + \gamma G_4 = 4 + (0.5 * 8) = 8$$

$$G_0 = R_1 + \gamma G_3 = 3 + (0.5 * 8) = 7$$

$$\gamma = 0.5$$



Monte Carlo Prediction

We want to estimate $v_{\pi}(s)$

First-visit MC prediction, for estimating $V \approx v_{\pi}$

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$

$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Loop forever (for each episode):

Generate an episode following π : $S_0, A_0, R_1, S_1, A_1, R_2, \dots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \dots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless S_t appears in S_0, S_1, \dots, S_{t-1} :

Append G to $Returns(S_t)$

$V(S_t) \leftarrow \text{average}(Returns(S_t))$

$$G_4 = R_5 = 2$$

$$G_3 = R_4 + \gamma G_4 = 1 + (0.5 * 2) = 2$$

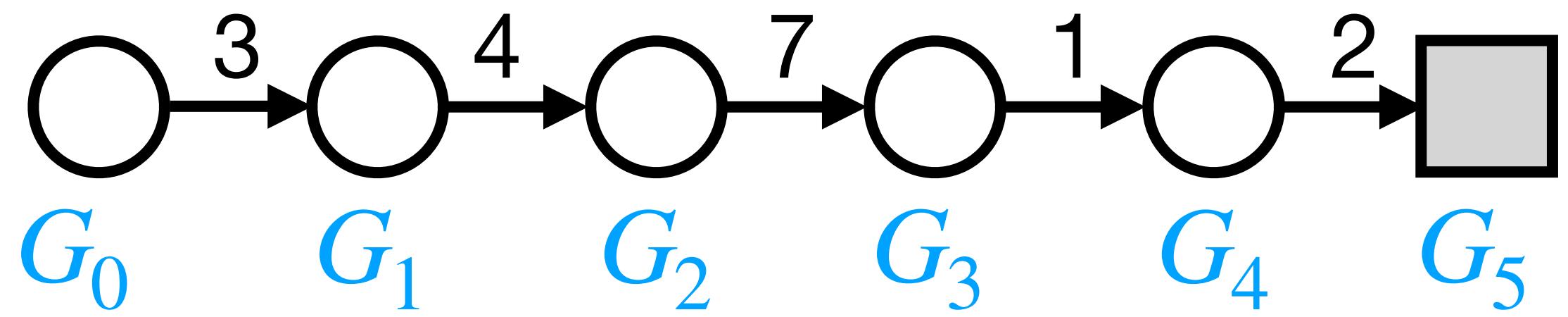
$$G_2 = R_3 + \gamma G_3 = 7 + (0.5 * 2) = 8$$

$$G_1 = R_2 + \gamma G_4 = 4 + (0.5 * 8) = 8$$

$$G_0 = R_1 + \gamma G_3 = 3 + (0.5 * 8) = 7$$

$$Returns(S_0) = [G_1, G_2, G_3, G_4, G_5] = [8, 8, 2, 2, 0]$$

$$\gamma = 0.5$$



Monte Carlo Prediction

We want to estimate $v_{\pi}(s)$

First-visit MC prediction, for estimating $V \approx v_{\pi}$

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$

$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Loop forever (for each episode):

Generate an episode following π : $S_0, A_0, R_1, S_1, A_1, R_2, \dots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \dots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless S_t appears in S_0, S_1, \dots, S_{t-1} :

Append G to $Returns(S_t)$

$V(S_t) \leftarrow \text{average}(Returns(S_t))$

$$G_4 = R_5 = 2$$

$$G_3 = R_4 + \gamma G_4 = 1 + (0.5 * 2) = 2$$

$$G_2 = R_3 + \gamma G_3 = 7 + (0.5 * 2) = 8$$

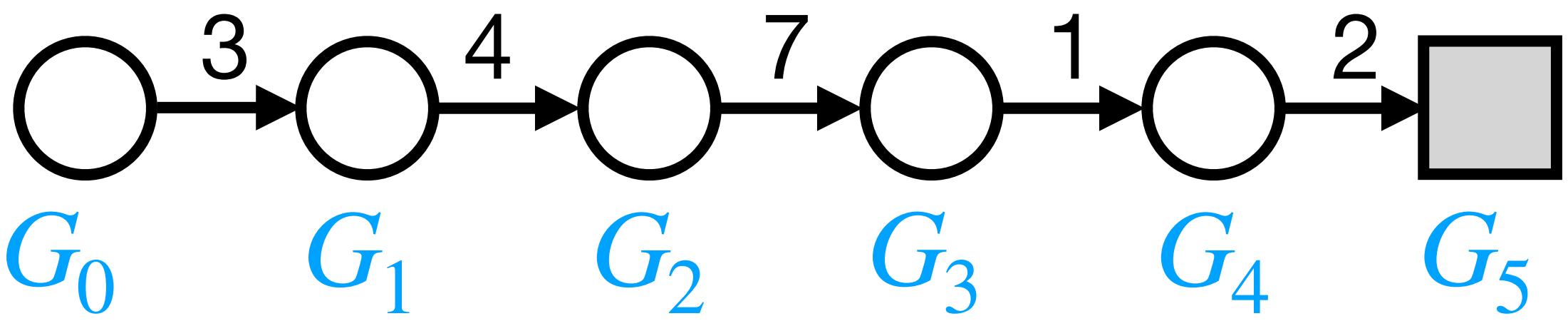
$$G_1 = R_2 + \gamma G_4 = 4 + (0.5 * 8) = 8$$

$$G_0 = R_1 + \gamma G_3 = 3 + (0.5 * 8) = 7$$

$$Returns(S_0) = [G_1, G_2, G_3, G_4, G_5] = [8, 8, 2, 2, 0]$$

$$V(S_0) = 4$$

$$\gamma = 0.5$$



Monte Carlo Prediction

We want to estimate $v_{\pi}(s)$

First-visit MC prediction, for estimating $V \approx v_{\pi}$

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$

$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Loop forever (for each episode):

Generate an episode following π : $S_0, A_0, R_1, S_1, A_1, R_2, \dots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \dots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless S_t appears in S_0, S_1, \dots, S_{t-1} :

Append G to $Returns(S_t)$

$V(S_t) \leftarrow \text{average}(Returns(S_t))$

$$G_4 = R_5 = 2$$

$$G_3 = R_4 + \gamma G_4 = 1 + (0.5 * 2) = 2$$

$$G_2 = R_3 + \gamma G_3 = 7 + (0.5 * 2) = 8$$

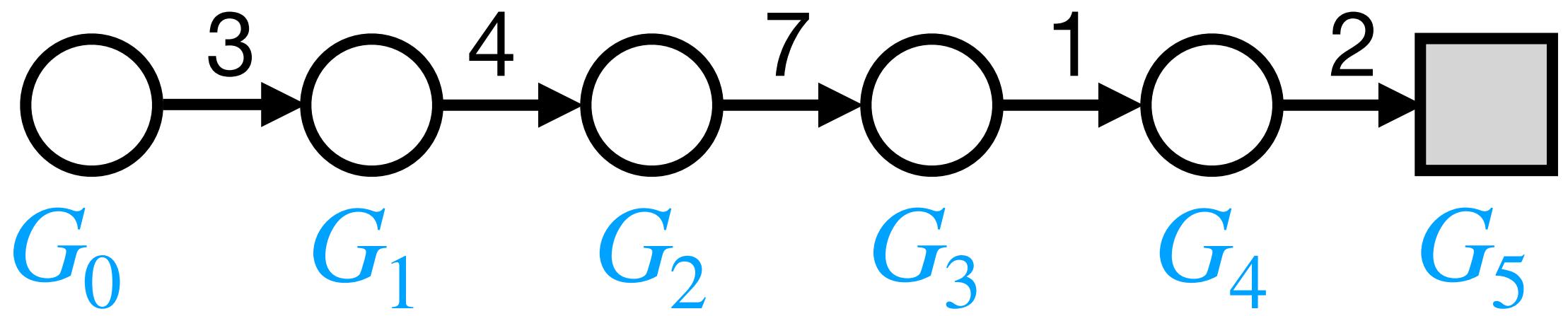
$$G_1 = R_2 + \gamma G_4 = 4 + (0.5 * 8) = 8$$

$$G_0 = R_1 + \gamma G_3 = 3 + (0.5 * 8) = 7$$

$$Returns(S_0) = [G_1, G_2, G_3, G_4, G_5] = [8, 8, 2, 2, 0]$$

$$V(S_0) = 4$$

$$\gamma = 0.5$$



We can avoid keeping values in a list

Monte Carlo Prediction

We want to estimate $v_{\pi}(s)$

First-visit MC prediction, for estimating $V \approx v_{\pi}$

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$

$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Loop forever (for each episode):

Generate an episode following π : $S_0, A_0, R_1, S_1, A_1, R_2, \dots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \dots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless S_t appears in S_0, S_1, \dots, S_{t-1} :

Append G to $Returns(S_t)$

$V(S_t) \leftarrow \text{average}(Returns(S_t))$

$$G_4 = R_5 = 2$$

$$G_3 = R_4 + \gamma G_4 = 1 + (0.5 * 2) = 2$$

$$G_2 = R_3 + \gamma G_3 = 7 + (0.5 * 2) = 8$$

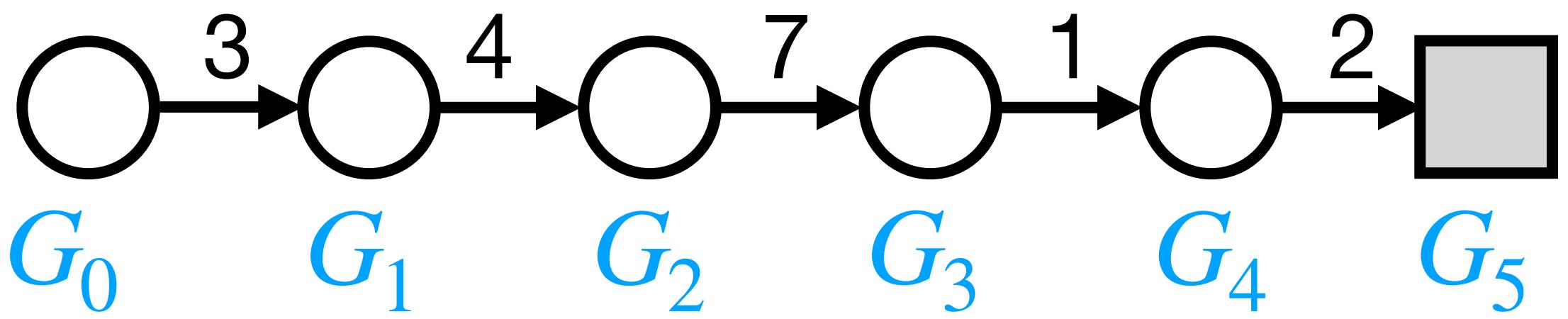
$$G_1 = R_2 + \gamma G_4 = 4 + (0.5 * 8) = 8$$

$$G_0 = R_1 + \gamma G_3 = 3 + (0.5 * 8) = 7$$

$$Returns(S_0) = [G_1, G_2, G_3, G_4, G_5] = [8, 8, 2, 2, 0]$$

$$V(S_0) = 4$$

$$\gamma = 0.5$$



We can avoid keeping values in a list

`new_est <- old_est + stepsize[target - Old est]`

Q-Learning

Q-Learning

One of the first major *online* RL algorithms

Q-Learning

One of the first major *online* RL algorithms

Bellman Equation for Q-Learning

Q-Learning

One of the first major *online* RL algorithms

Bellman Equation for Q-Learning

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t))$$

Q-Learning

One of the first major *online* RL algorithms

Bellman Equation for Q-Learning

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t))$$

Use samples from the environment

Q-Learning

One of the first major *online* RL algorithms

Bellman Equation for Q-Learning

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t))$$

Use samples from the environment

Q-Learning converges to the optimal value function, as long as the agent continues to explore and samples all areas at the state-action space

Q-Learning

One of the first major *online* RL algorithms

Bellman Equation for Q-Learning

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t))$$

Use samples from the environment

Q-Learning converges to the optimal value function, as long as the agent continues to explore and samples all areas at the state-action space

Q-Learning target policy is Greedy wrt its current values (max operator)

Q-Learning

One of the first major *online* RL algorithms

Bellman Equation for Q-Learning

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t))$$

Use samples from the environment

Q-Learning converges to the optimal value function, as long as the agent continues to explore and samples all areas at the state-action space

Q-Learning target policy is Greedy wrt its current values (max operator)

Its behaviour policy can be anything that continues to visit all state actions pairs during learning (e.g. ε – greedy)

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$

Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\text{terminal}, \cdot) = 0$

Loop for each episode:

 Initialize S

 Loop for each step of episode:

 Choose A from S using policy derived from Q (e.g., ε -greedy)

 Take action A , observe R, S'

$$Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', a) - Q(S, A)]$$

$S \leftarrow S'$

 until S is terminal

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$

Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\text{terminal}, \cdot) = 0$

Loop for each episode:

 Initialize S

 Loop for each step of episode:

 Choose A from S using policy derived from Q (e.g., ε -greedy)

 Take action A , observe R, S'

$$Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', a) - Q(S, A)]$$

$$S \leftarrow S'$$

 until S is terminal

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t))$$

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$

Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\text{terminal}, \cdot) = 0$

Loop for each episode:

 Initialize S

 Loop for each step of episode:

 Choose A from S using policy derived from Q (e.g., ε -greedy)

 Take action A , observe R, S'

$$Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', a) - Q(S, A)]$$

$S \leftarrow S'$

 until S is terminal

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t))$$

Bellman equation for state-action values

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$

Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\text{terminal}, \cdot) = 0$

Loop for each episode:

 Initialize S

 Loop for each step of episode:

 Choose A from S using policy derived from Q (e.g., ε -greedy)

 Take action A , observe R, S'

$$Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', a) - Q(S, A)]$$

$S \leftarrow S'$

 until S is terminal

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t))$$

Bellman equation for state-action values

What is $Q(S, A)$?

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$

Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\text{terminal}, \cdot) = 0$

Loop for each episode:

 Initialize S

 Loop for each step of episode:

 Choose A from S using policy derived from Q (e.g., ε -greedy)

 Take action A , observe R, S'

$$Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', a) - Q(S, A)]$$

$S \leftarrow S'$

 until S is terminal

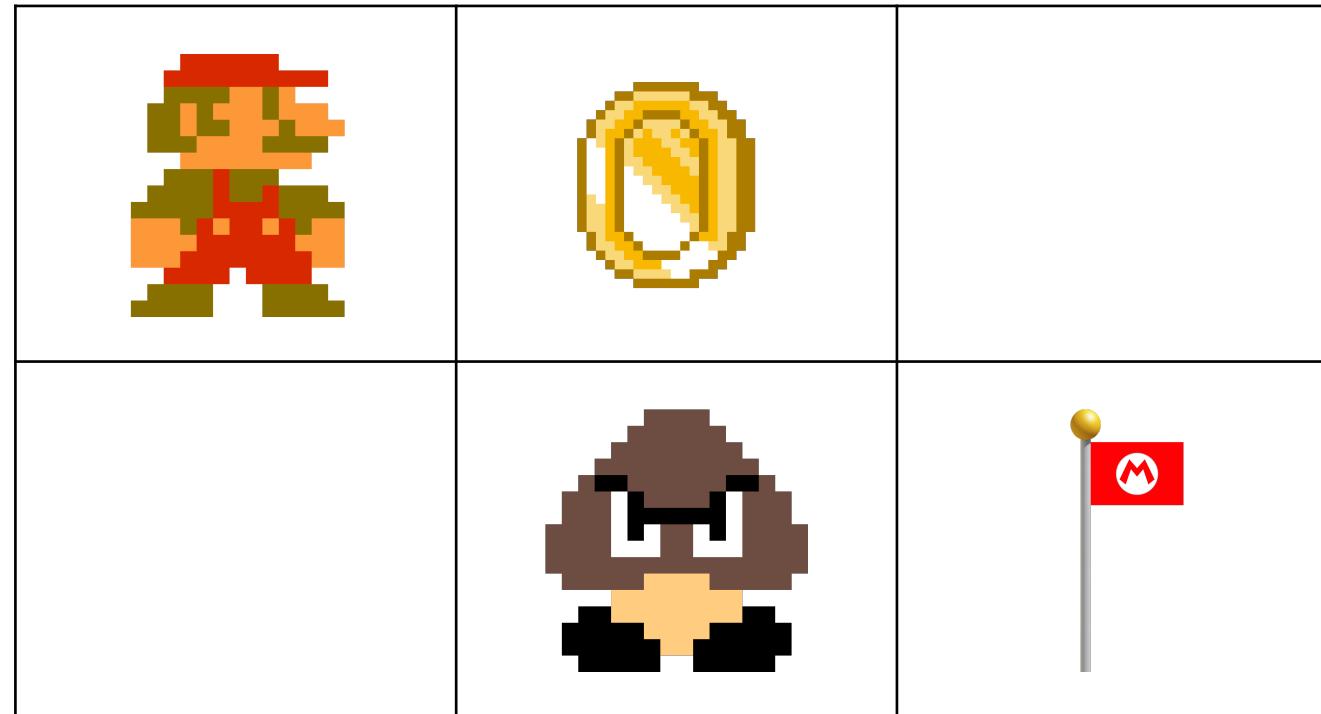
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t))$$

Bellman equation for state-action values

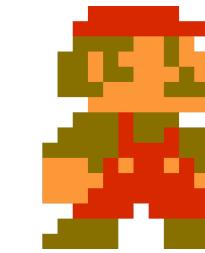
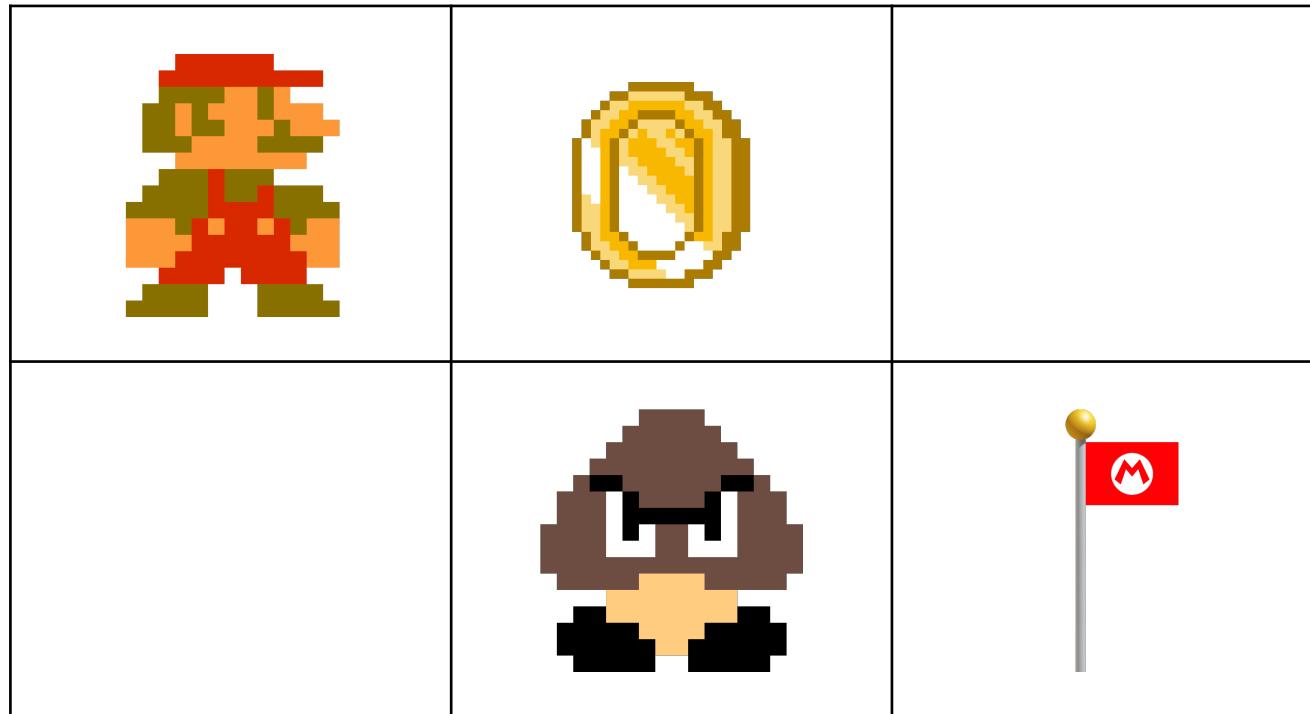
What is $Q(S, A)$?

Encode Q-values in a table, each cell corresponds to a state-action pair value.

Q-Learning example

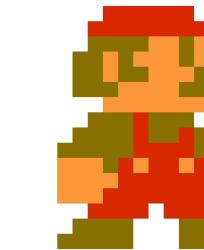
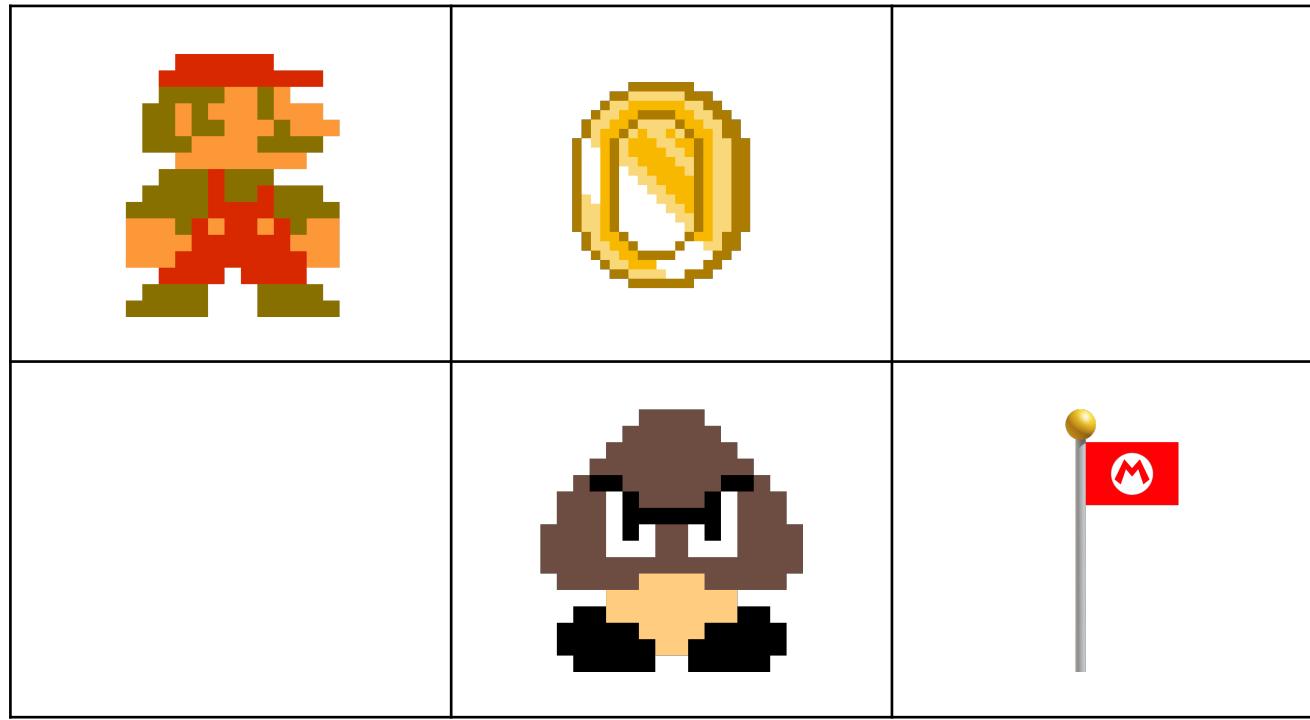


Q-Learning example



Policy π

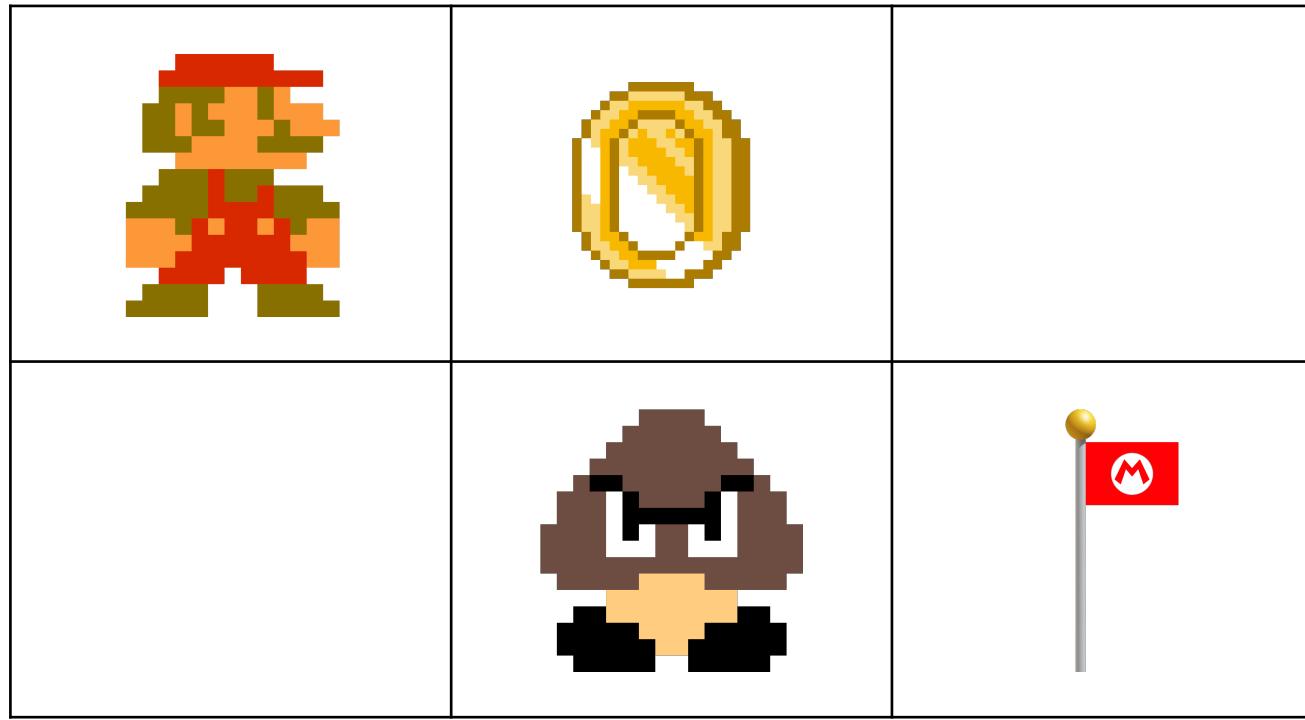
Q-Learning example

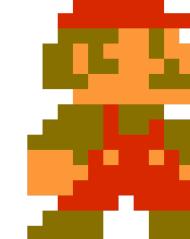


Policy π

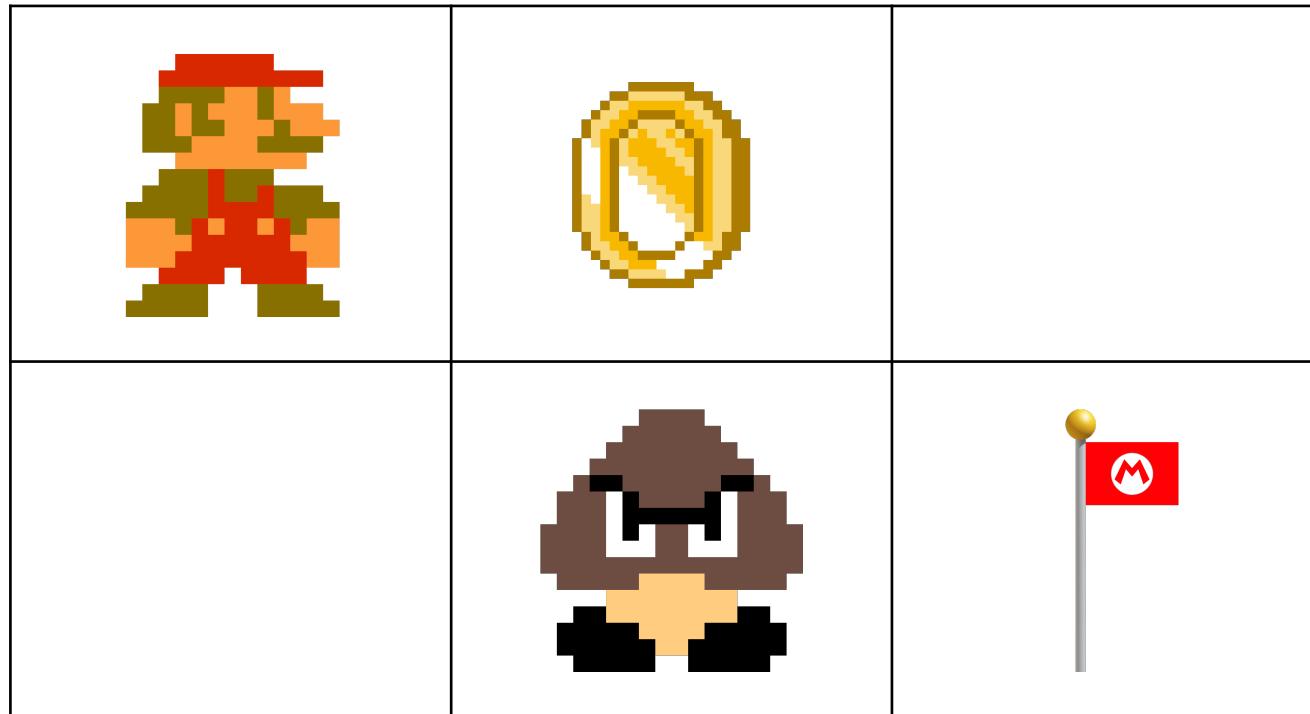
Actions $\uparrow \downarrow \leftarrow \rightarrow$

Q-Learning example



 Policy π
Actions $\uparrow \downarrow \leftarrow \rightarrow$
 +1

Q-Learning example

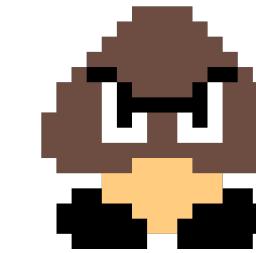


Policy π

Actions $\uparrow \downarrow \leftarrow \rightarrow$

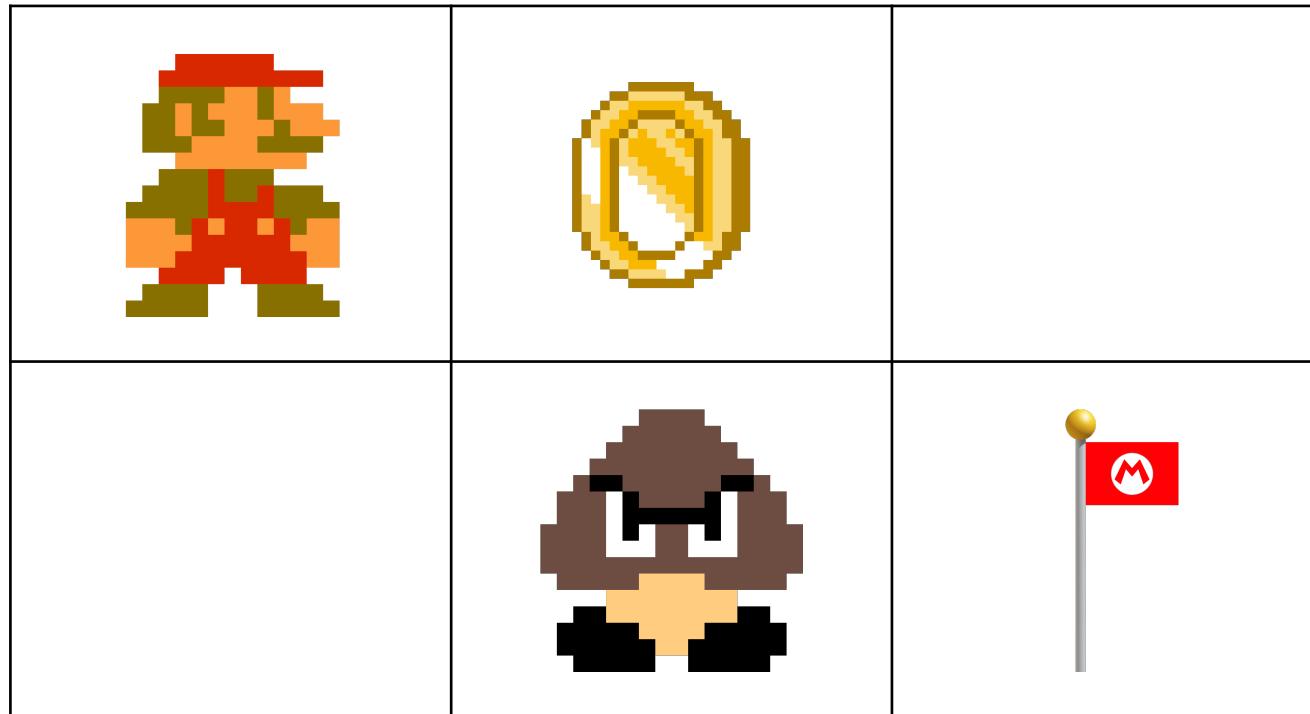


+1



-10

Q-Learning example



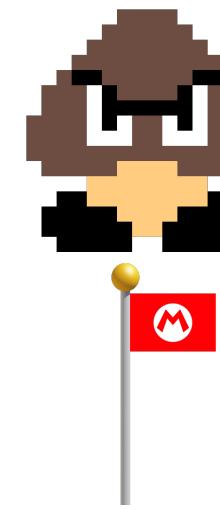
Policy π

Actions

$\uparrow \downarrow \leftarrow \rightarrow$

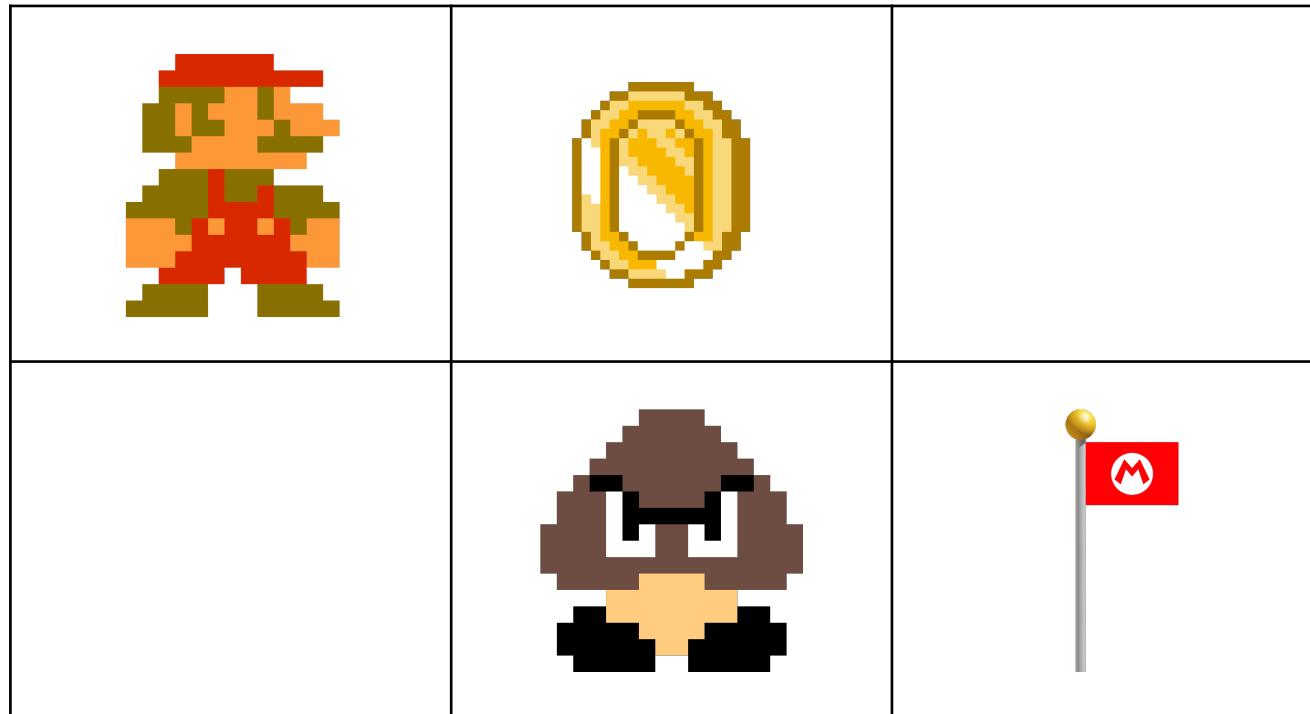


+1



-10
+10

Q-Learning example



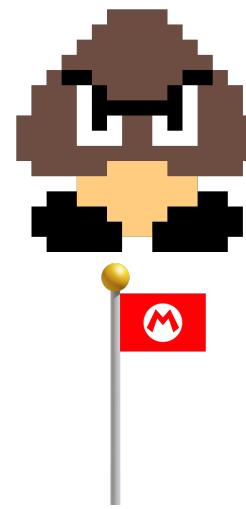
Policy π

Actions



$+1$

$\uparrow \downarrow \leftarrow \rightarrow$



-10

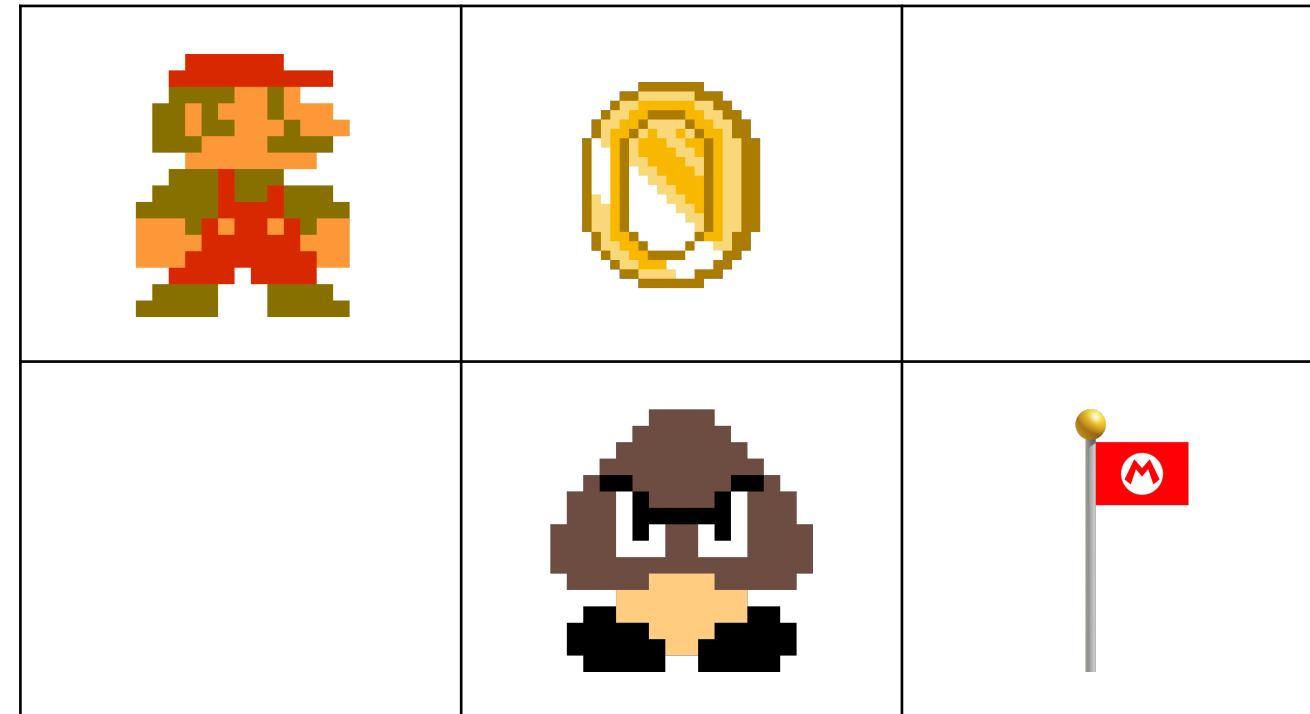
$+10$

> 5 steps

0



Q-Learning example



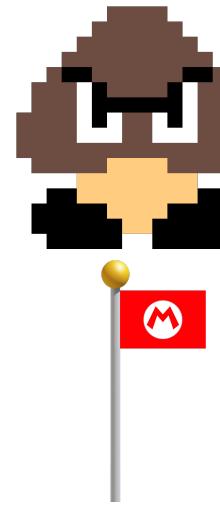
Policy π

Actions



+1

$\uparrow \downarrow \leftarrow \rightarrow$



-10

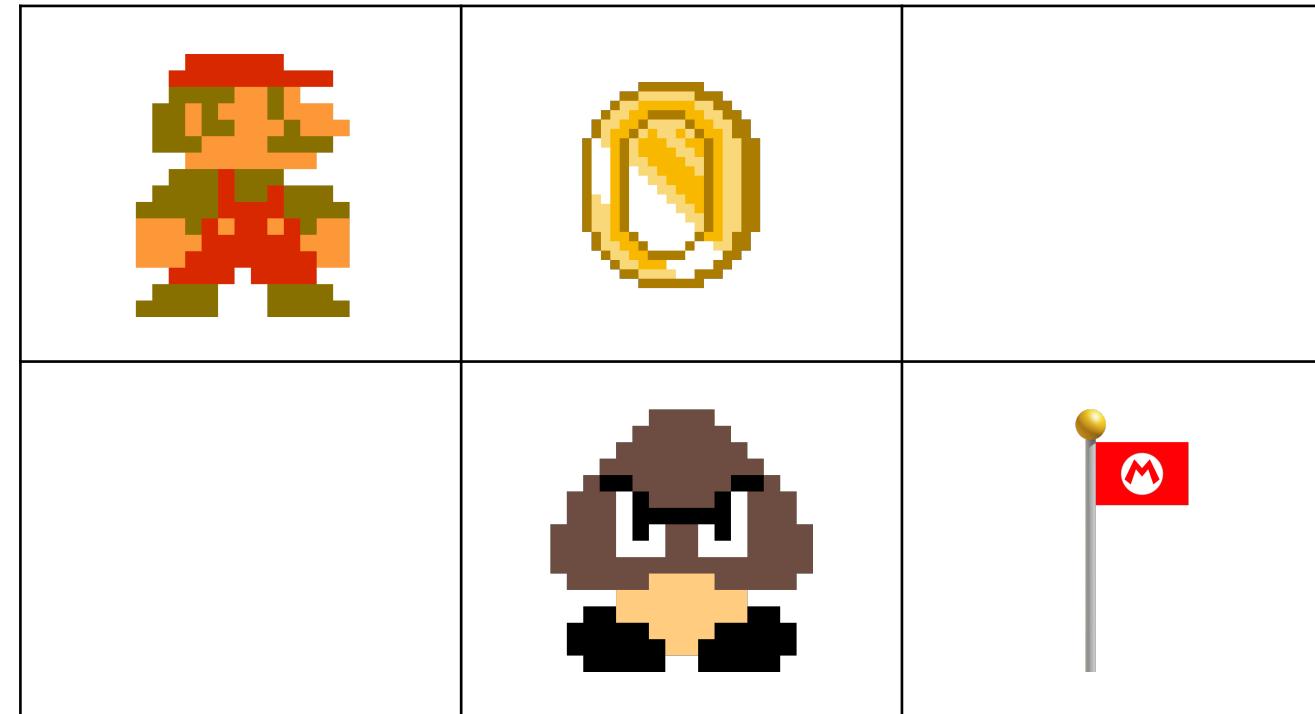
+10

> 5 steps

0

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t))$$

Q-Learning example



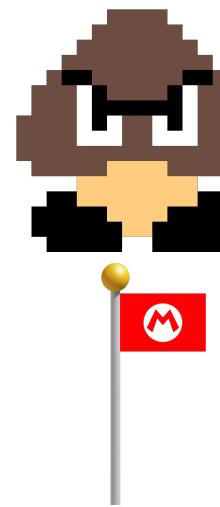
Policy π

Actions



+1

$\uparrow \downarrow \leftarrow \rightarrow$



-10

+10

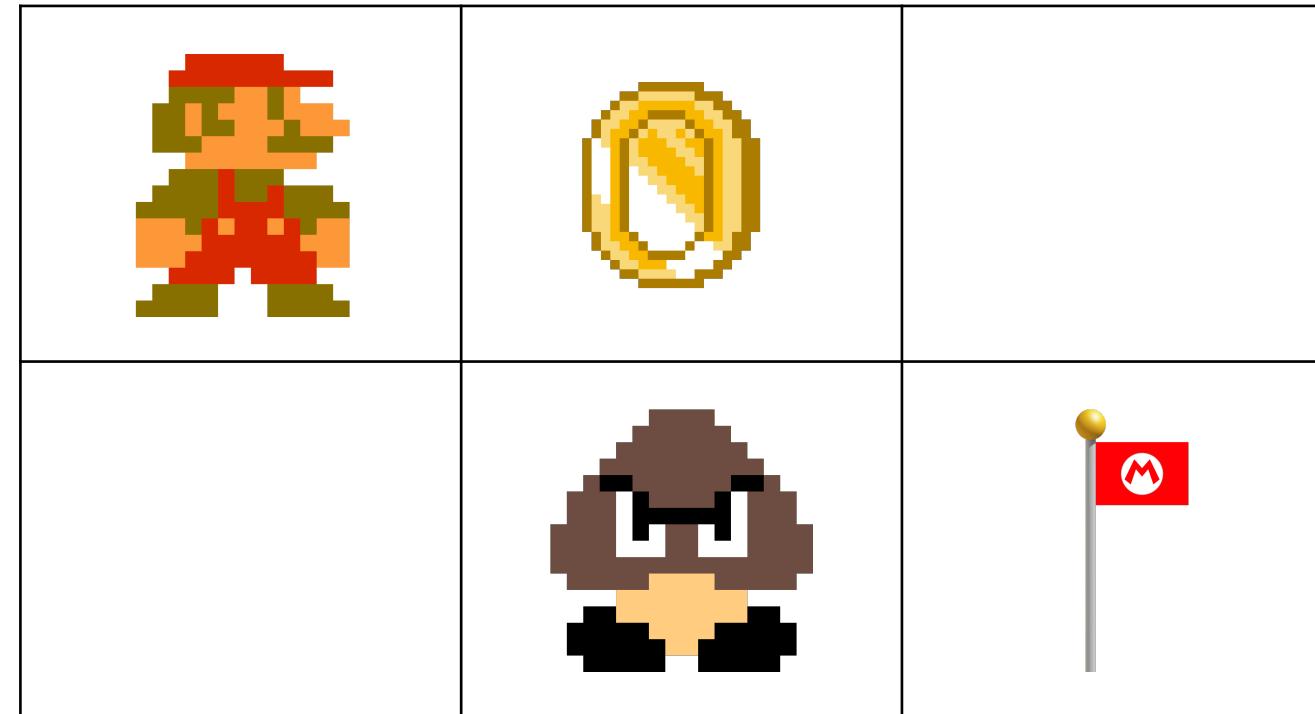
> 5 steps

0

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t))$$

$$\begin{aligned}\alpha &= 0.1 \\ \gamma &= 0.99\end{aligned}$$

Q-Learning example



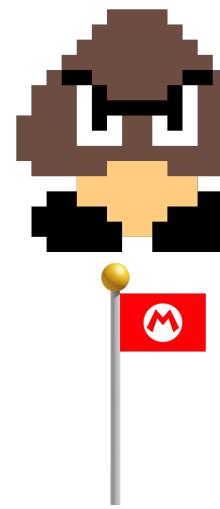
Policy π

Actions



+1

$\uparrow \downarrow \leftarrow \rightarrow$



-10

+10

> 5 steps

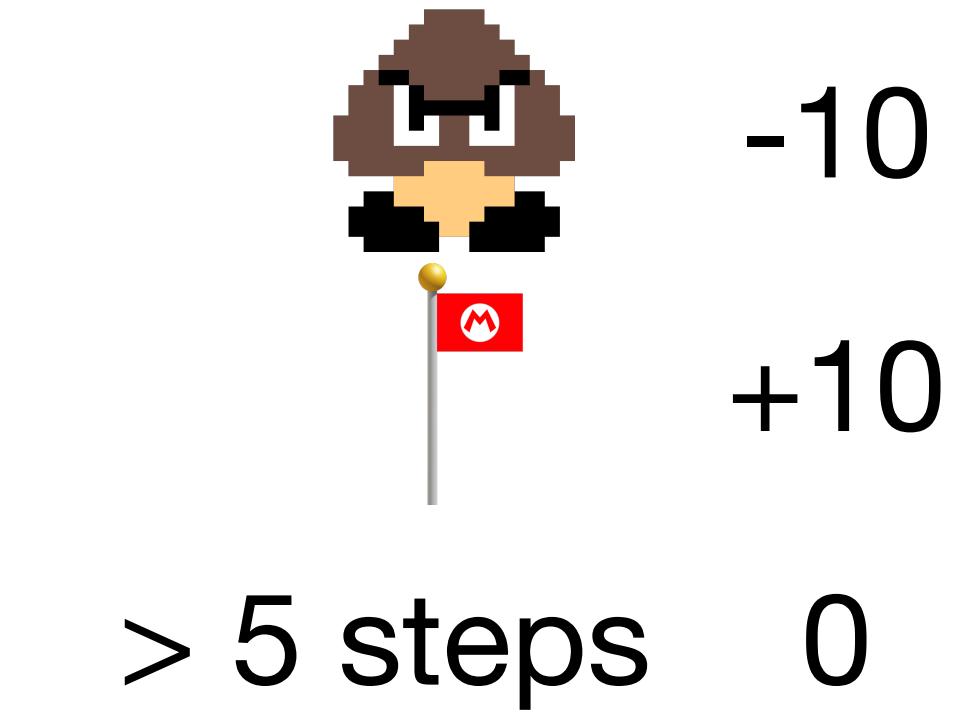
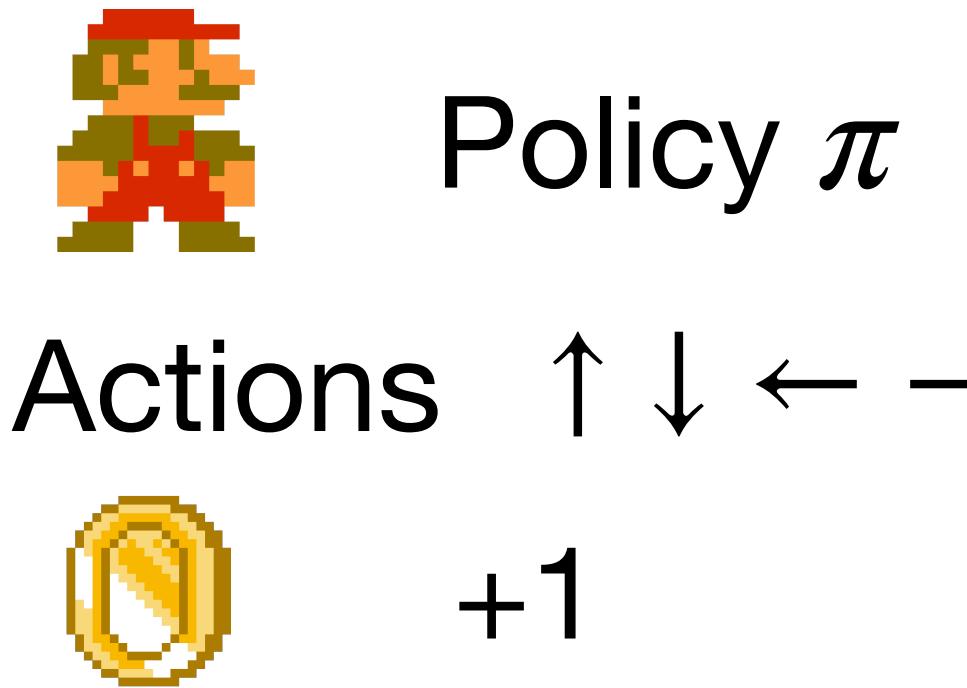
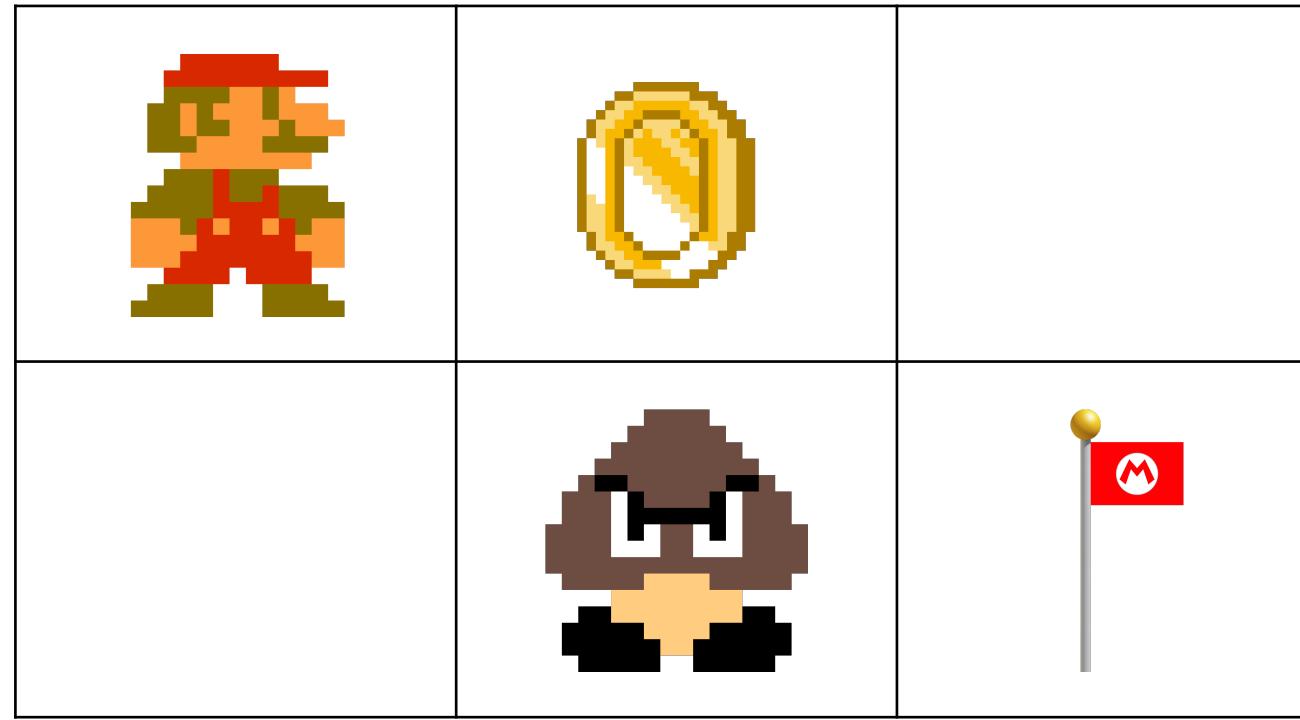
0

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t))$$

$$\begin{aligned}\alpha &= 0.1 \\ \gamma &= 0.99\end{aligned}$$

How to represent the Q-Table?

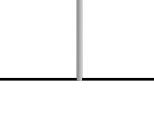
Q-Learning example



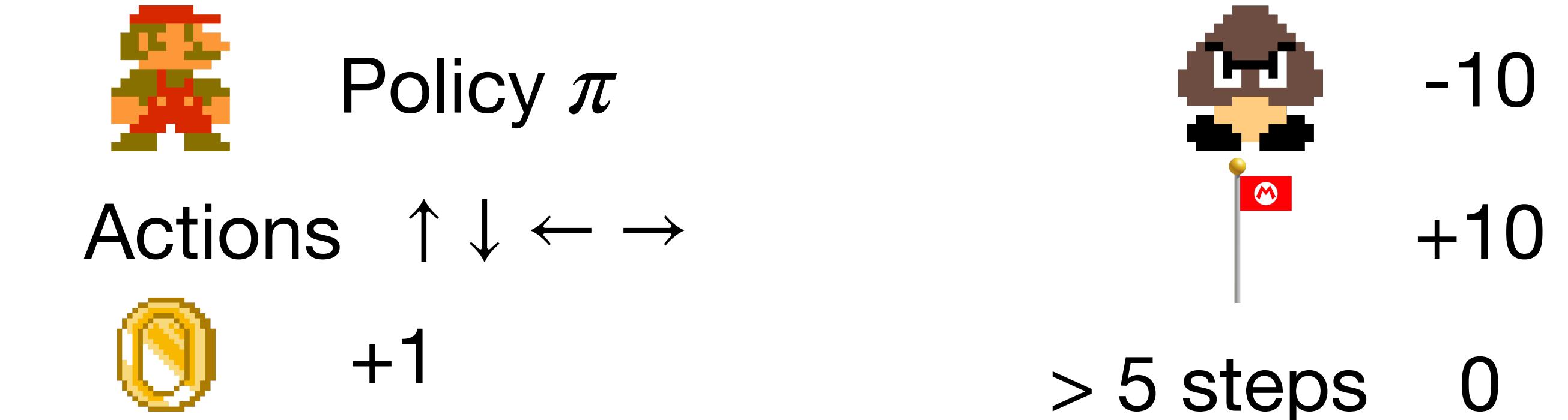
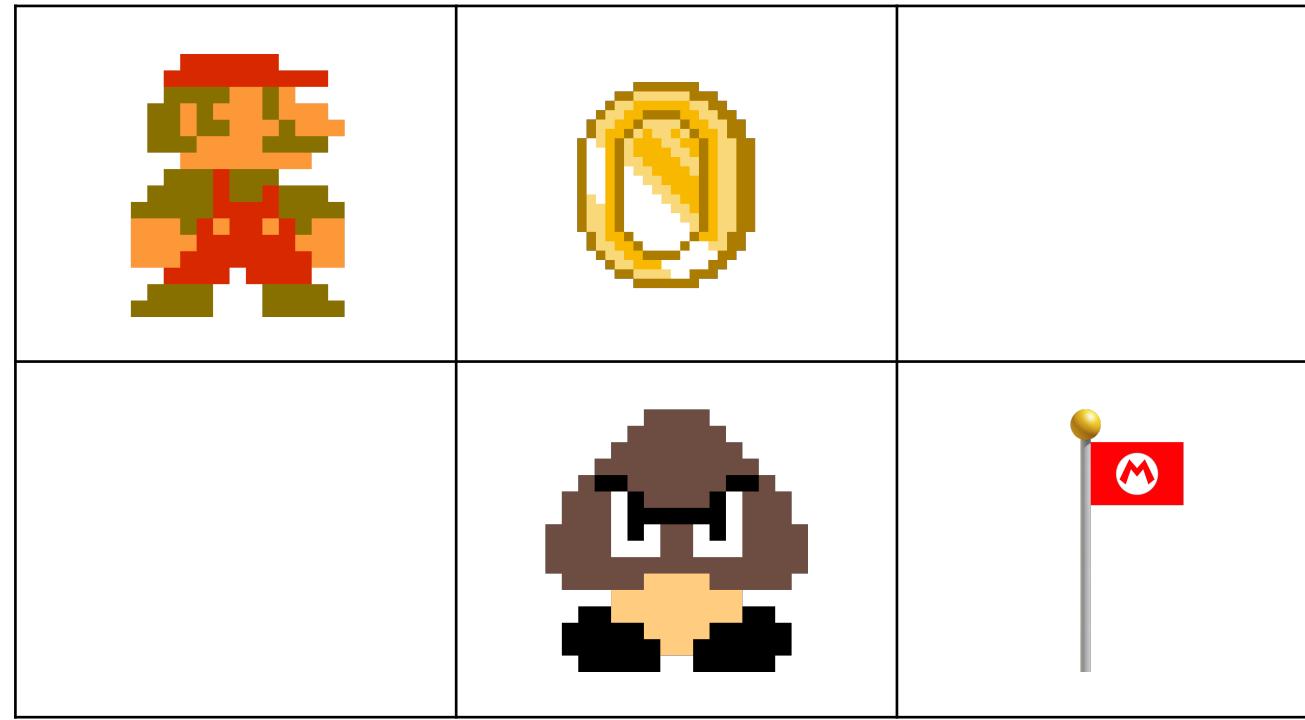
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t))$$

$$\begin{aligned}\alpha &= 0.1 \\ \gamma &= 0.99\end{aligned}$$

How to represent the Q-Table?

	\leftarrow	\rightarrow	\uparrow	\downarrow
				
				
				
				

Q-Learning example



$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t))$$

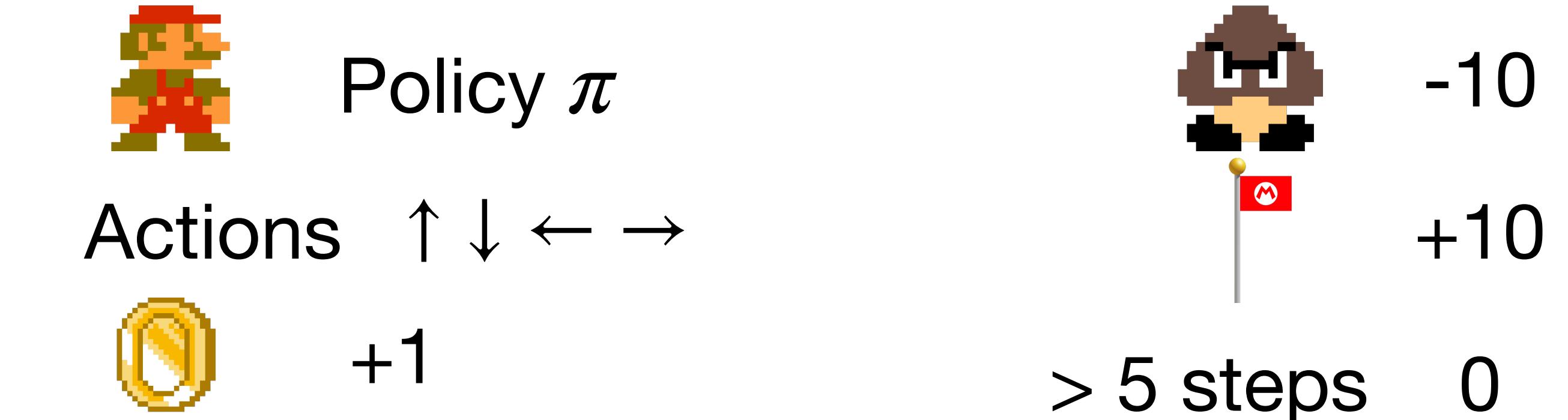
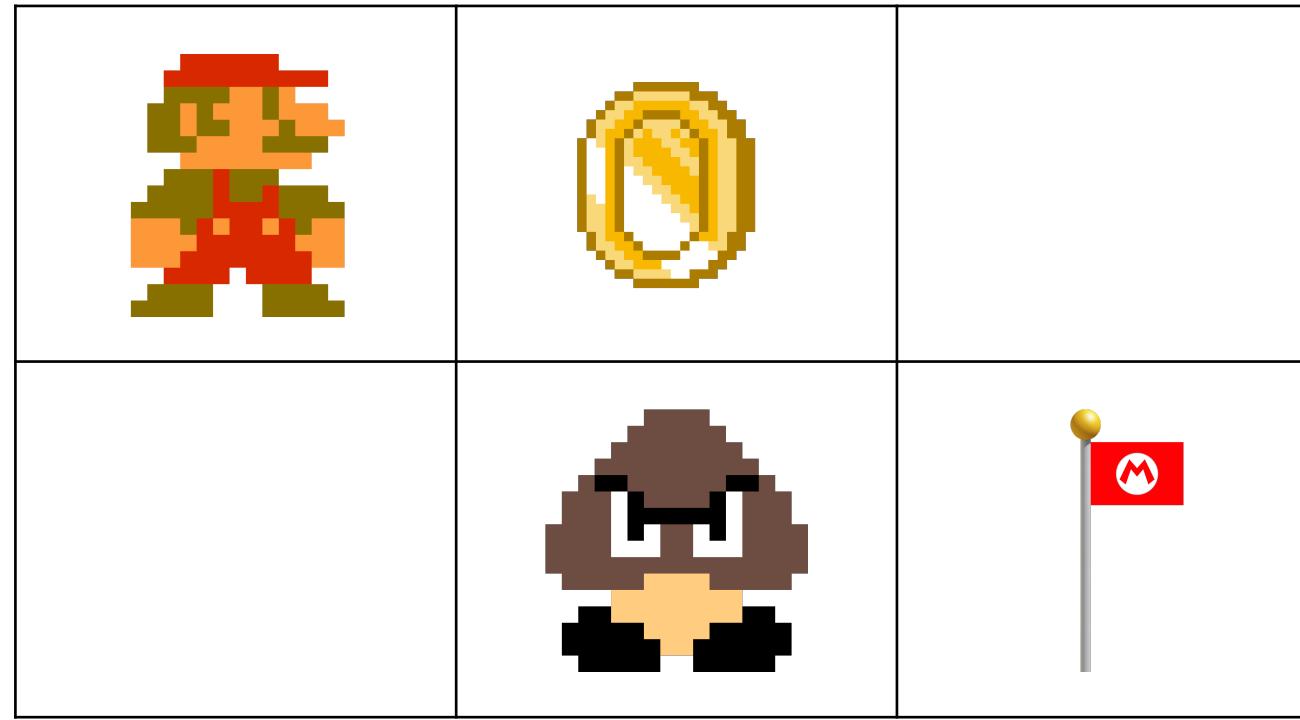
$$\begin{aligned}\alpha &= 0.1 \\ \gamma &= 0.99\end{aligned}$$

How to represent the Q-Table?

	\leftarrow	\rightarrow	\uparrow	\downarrow

How to move?

Q-Learning example



$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t))$$

$$\begin{aligned}\alpha &= 0.1 \\ \gamma &= 0.99\end{aligned}$$

How to represent the Q-Table?

	\leftarrow	\rightarrow	\uparrow	\downarrow

How to move?

$$\pi = \operatorname{argmax}_A Q(S, A)$$

Q-Learning example

Q-Learning example

Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+$, $a \in \mathcal{A}(s)$, arbitrarily except that $Q(\text{terminal}, \cdot) = 0$

Q-Learning example

Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+$, $a \in \mathcal{A}(s)$, arbitrarily except that $Q(\text{terminal}, \cdot) = 0$

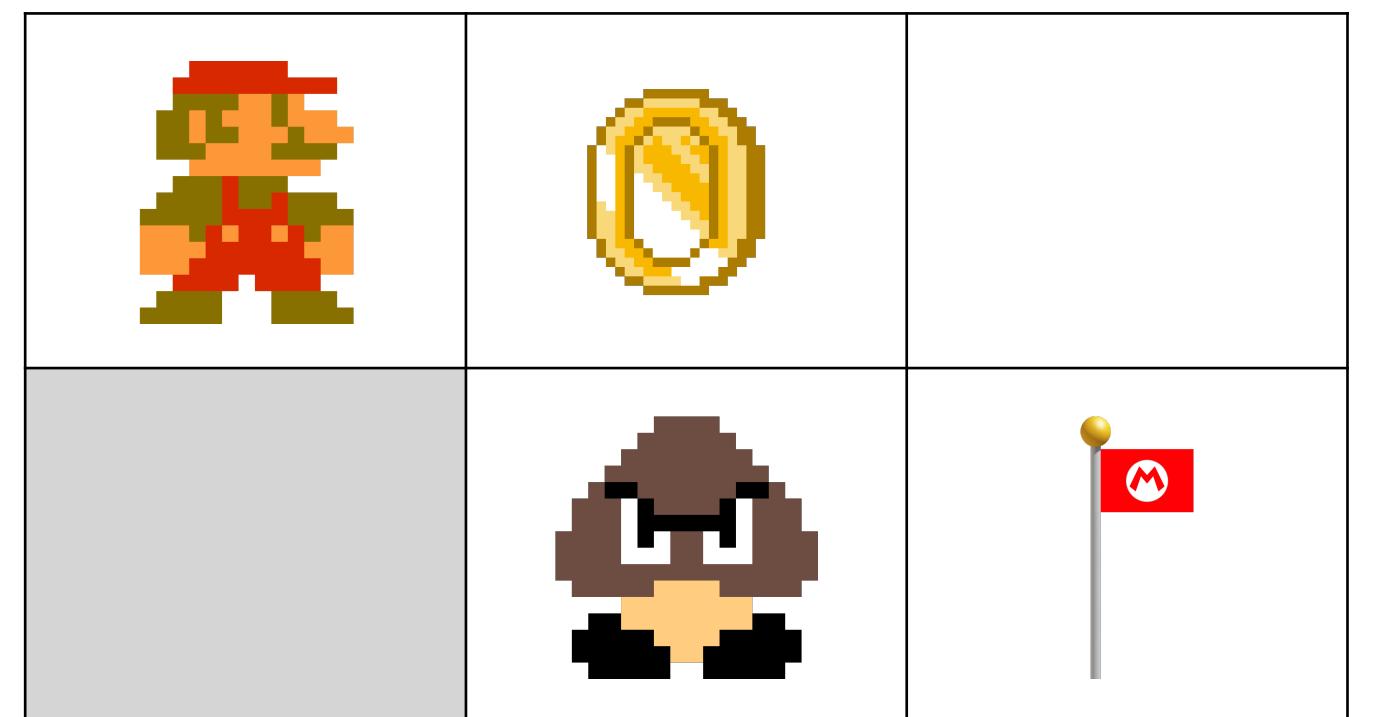
	\leftarrow	\rightarrow	\uparrow	\downarrow
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0

Q-Learning example

	←	→	↑	↓
Mario	0	0	0	0
Mushroom	0	0	0	0
Blank	0	0	0	0
Blank	0	0	0	0
Flag	0	0	0	0

Q-Learning example

	←	→	↑	↓
Mario	0	0	0	0
Gold Coin	0	0	0	0
Blank	0	0	0	0
Blank	0	0	0	0
Koopa Troopa	0	0	0	0
Flag	0	0	0	0



Q-Learning example

	←	→	↑	↓
Mario	0	0	0	0
Gold Coin	0	0	0	0
Blank	0	0	0	0
Blank	0	0	0	0
Koopa Troopa	0	0	0	0
Flag	0	0	0	0

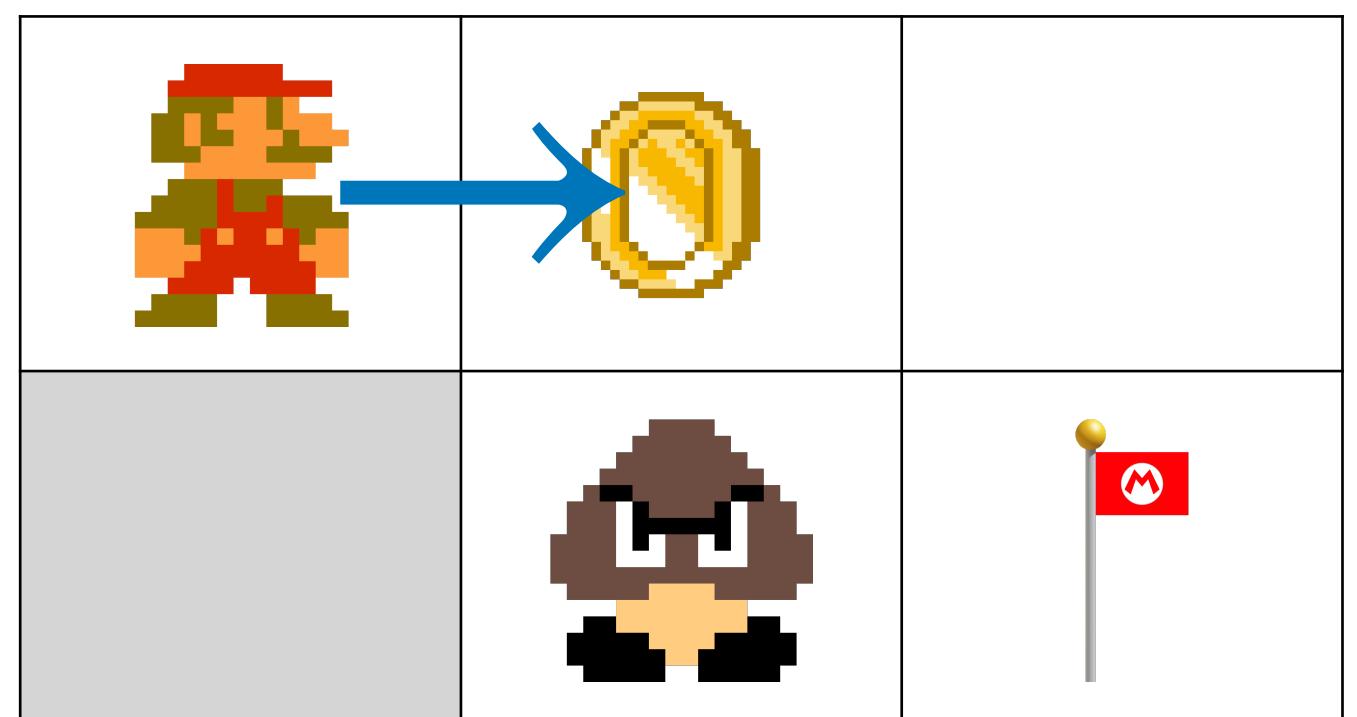
Choose A from S using policy derived from Q (e.g., ε -greedy)



Q-Learning example

	←	→	↑	↓
Luigi	0	0	0	0
Coin	0	0	0	0
Wall	0	0	0	0
Goomba	0	0	0	0
Flag	0	0	0	0

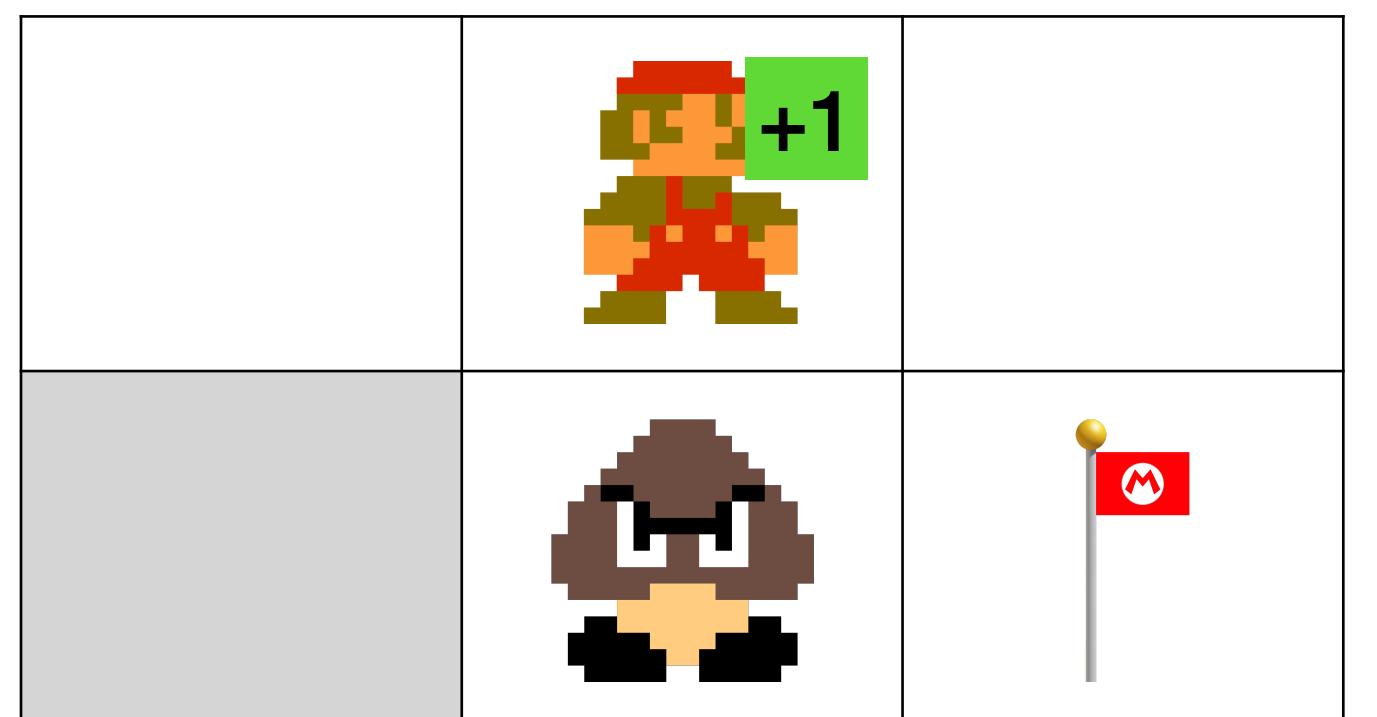
Choose A from S using policy derived from Q (e.g., ε -greedy)



Q-Learning example

	←	→	↑	↓
Luigi	0	0	0	0
Coin	0	0	0	0
Blank	0	0	0	0
Goomba	0	0	0	0
Flag	0	0	0	0

Take action A , observe R, S'

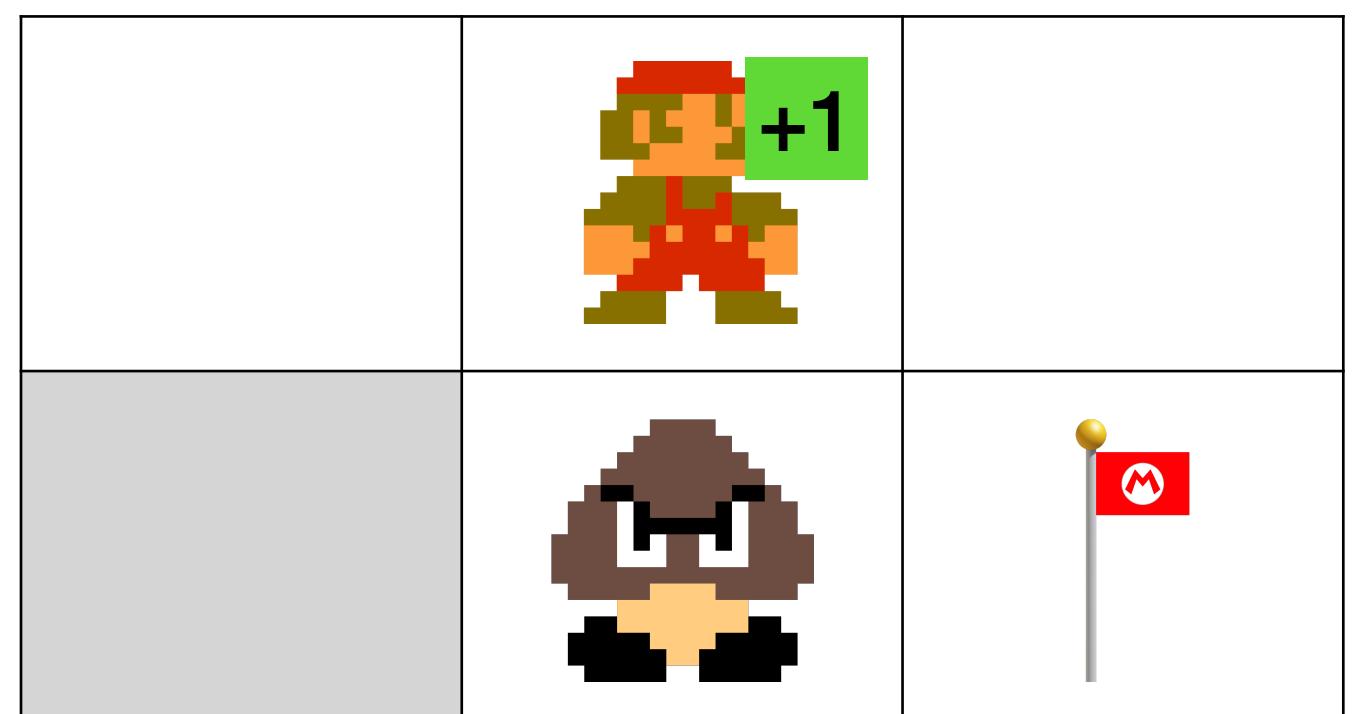


Q-Learning example

	←	→	↑	↓
Luigi	0	0	0	0
Goomba	0	0	0	0
Blank	0	0	0	0
Goomba	0	0	0	0
Flag	0	0	0	0

Update $Q(S, A)$

Take action A , observe R, S'



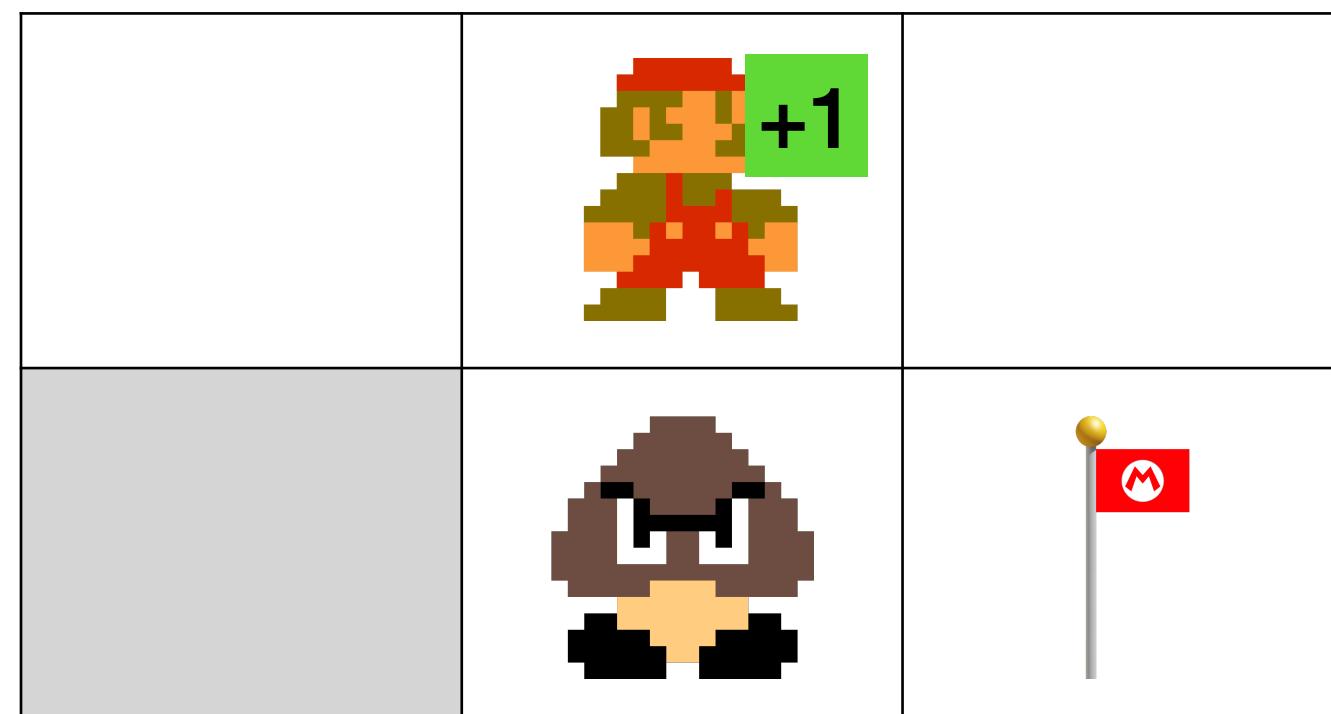
Q-Learning example

	←	→	↑	↓
Luigi	0	0	0	0
Goomba	0	0	0	0
Blank	0	0	0	0
Goomba	0	0	0	0
Flag	0	0	0	0

Update $Q(S, A)$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t))$$

Take action A , observe R, S'



Q-Learning example

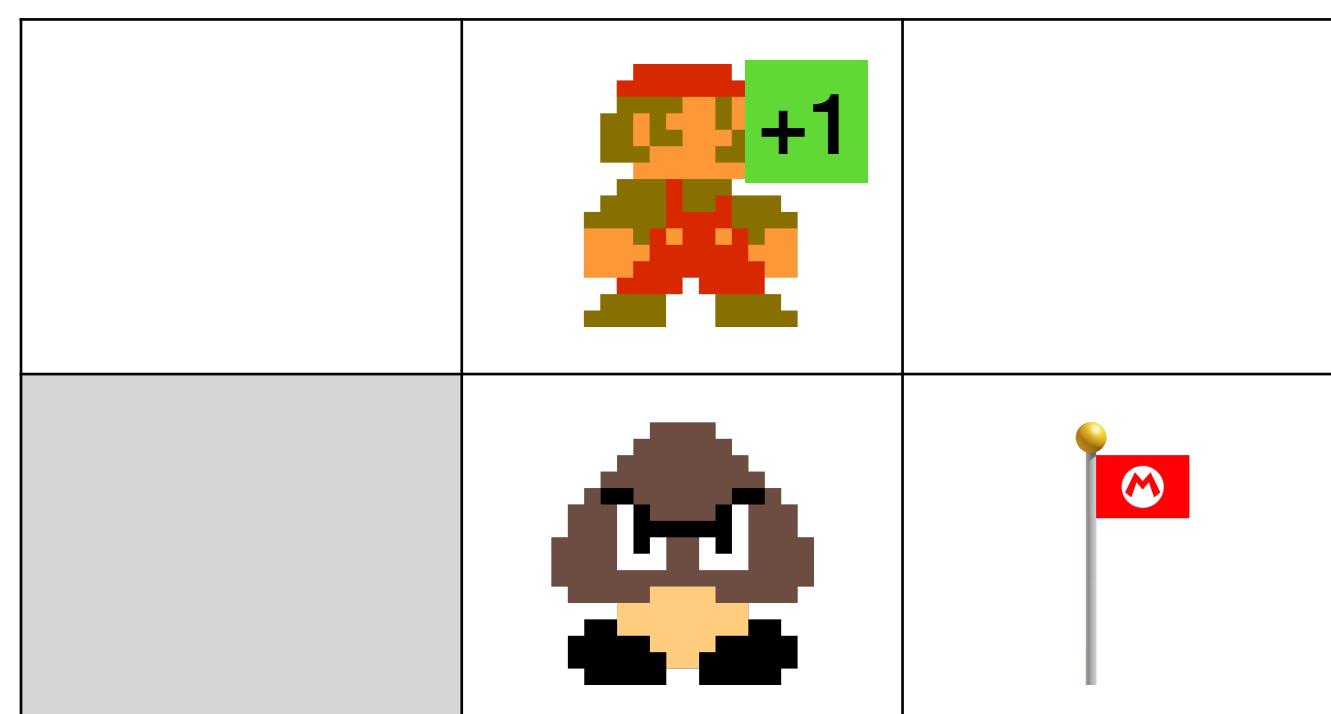
$$\alpha = 0.1$$
$$\gamma = 0.99$$

	←	→	↑	↓
Luigi	0	0	0	0
Goomba	0	0	0	0
Blank	0	0	0	0
Goomba	0	0	0	0
Flag	0	0	0	0

Update $Q(S, A)$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t))$$

Take action A , observe R, S'



Q-Learning example

$$\alpha = 0.1$$
$$\gamma = 0.99$$

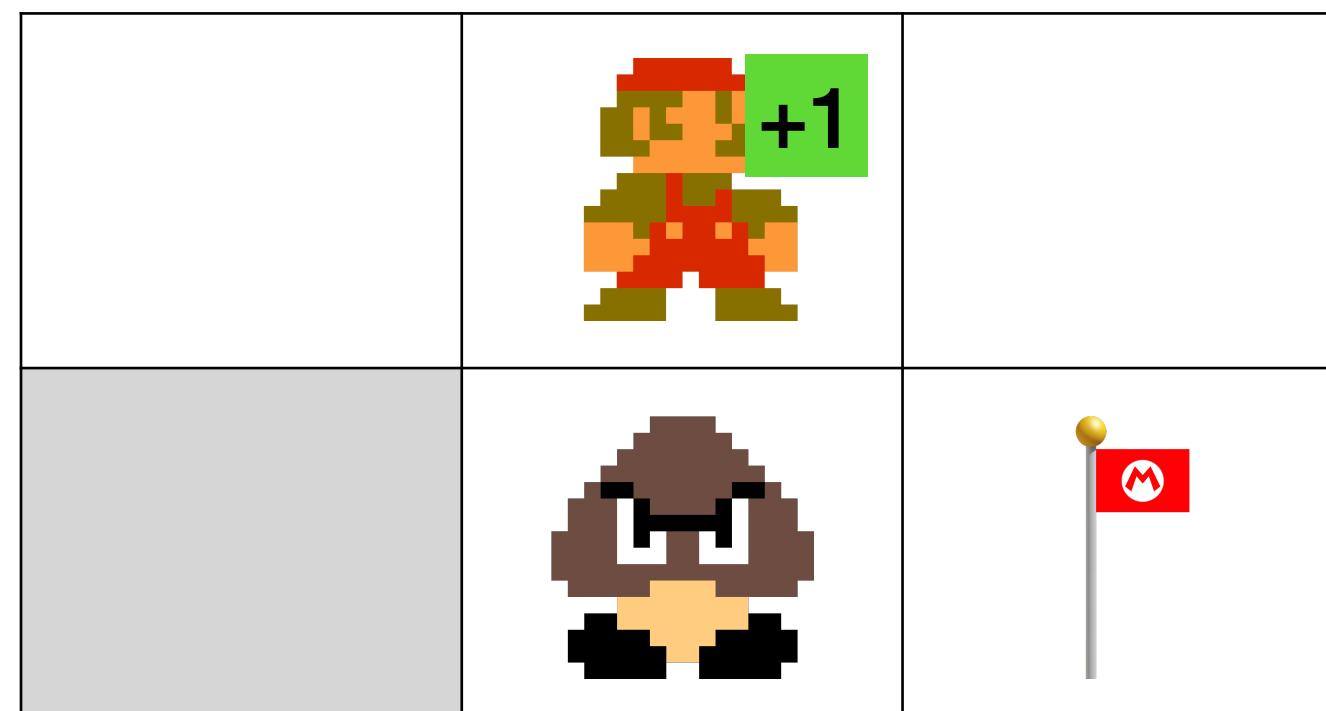
	←	→	↑	↓
Luigi	0	0	0	0
Goomba	0	0	0	0
Blank	0	0	0	0
Goomba	0	0	0	0
Flag	0	0	0	0

Update $Q(S, A)$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t))$$

$$Q(\text{initial}, \rightarrow) = 0 + 0.1 * [1 + 0.99 * 0 - 0] = 0.1$$

Take action A , observe R, S'



Q-Learning example

$$\alpha = 0.1$$
$$\gamma = 0.99$$

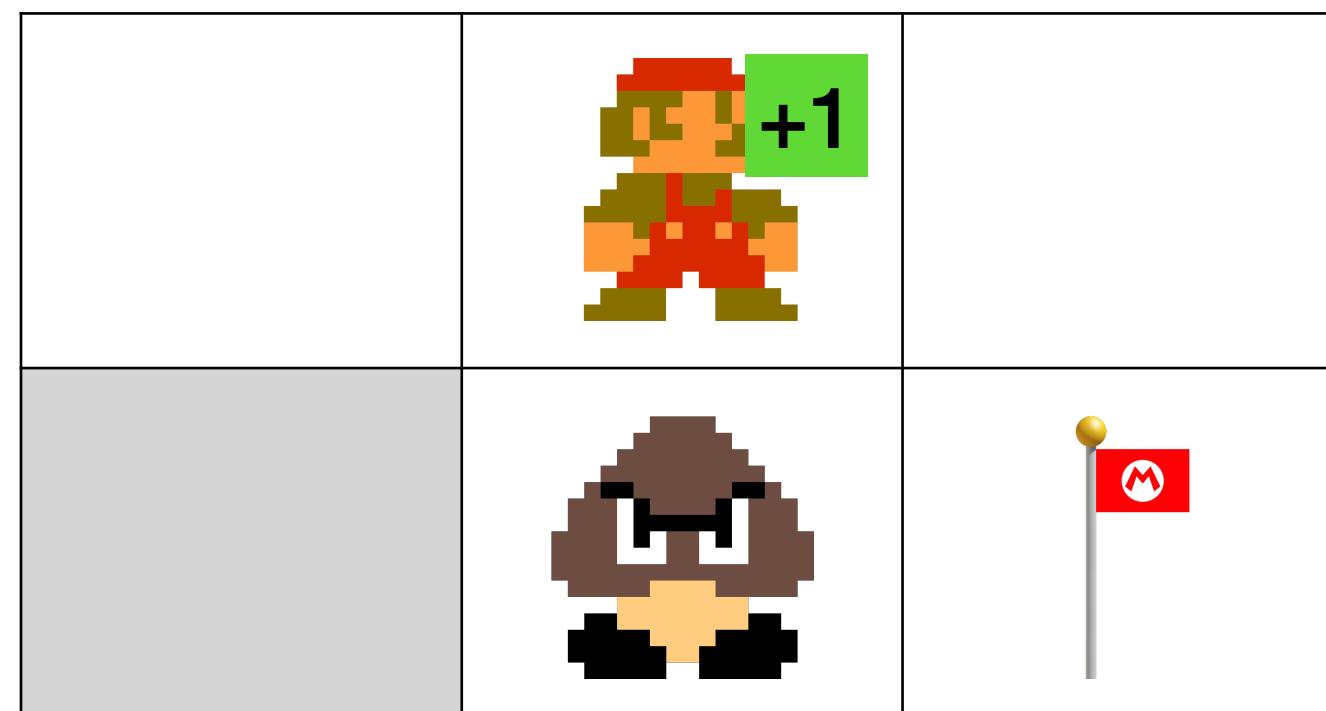
	←	→	↑	↓
Luigi	0	0.1	0	0
Goomba	0	0	0	0
Blank	0	0	0	0
Goomba	0	0	0	0
Flag	0	0	0	0

Update $Q(S, A)$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t))$$

$$Q(\text{initial}, \rightarrow) = 0 + 0.1 * [1 + 0.99 * 0 - 0] = 0.1$$

Take action A , observe R, S'

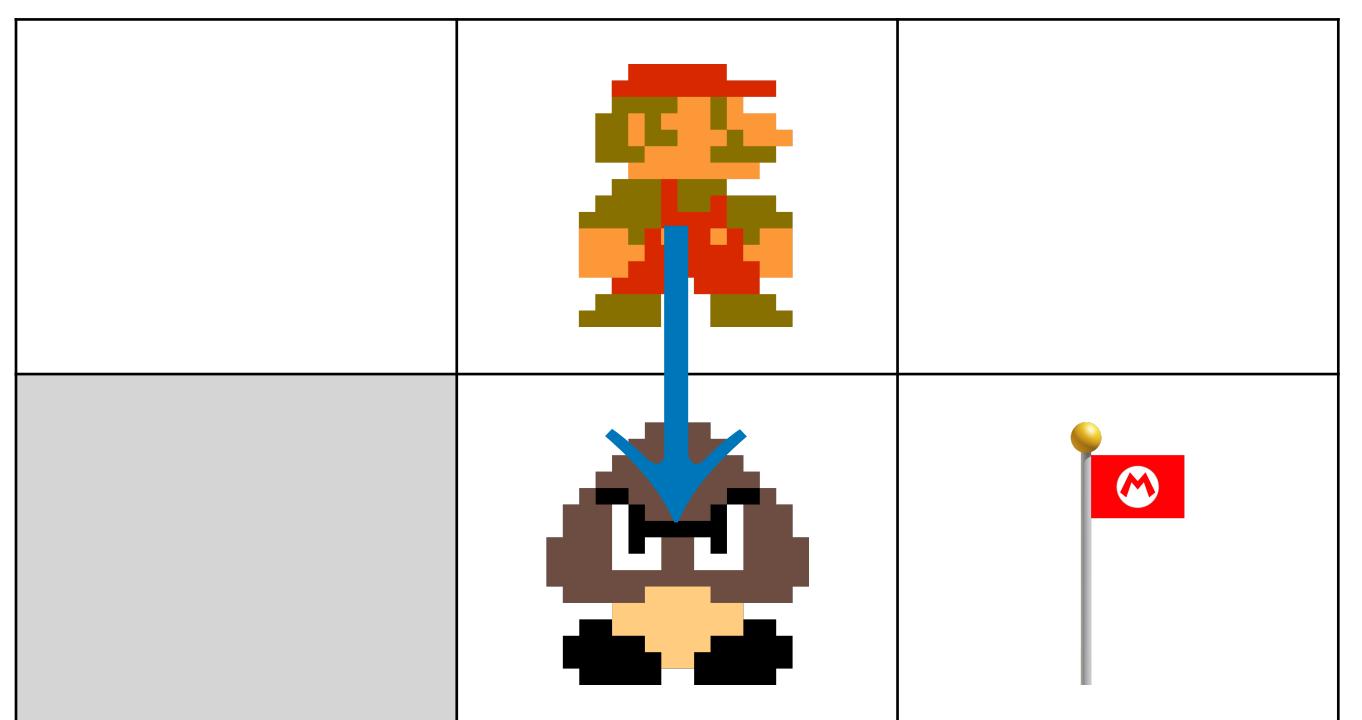


Q-Learning example

$$\alpha = 0.1$$
$$\gamma = 0.99$$

	←	→	↑	↓
Luigi	0	0.1	0	0
Goomba	0	0	0	0
Blank	0	0	0	0
Koopa	0	0	0	0
Flag	0	0	0	0

Choose A from S using policy derived from Q (e.g., ε -greedy)

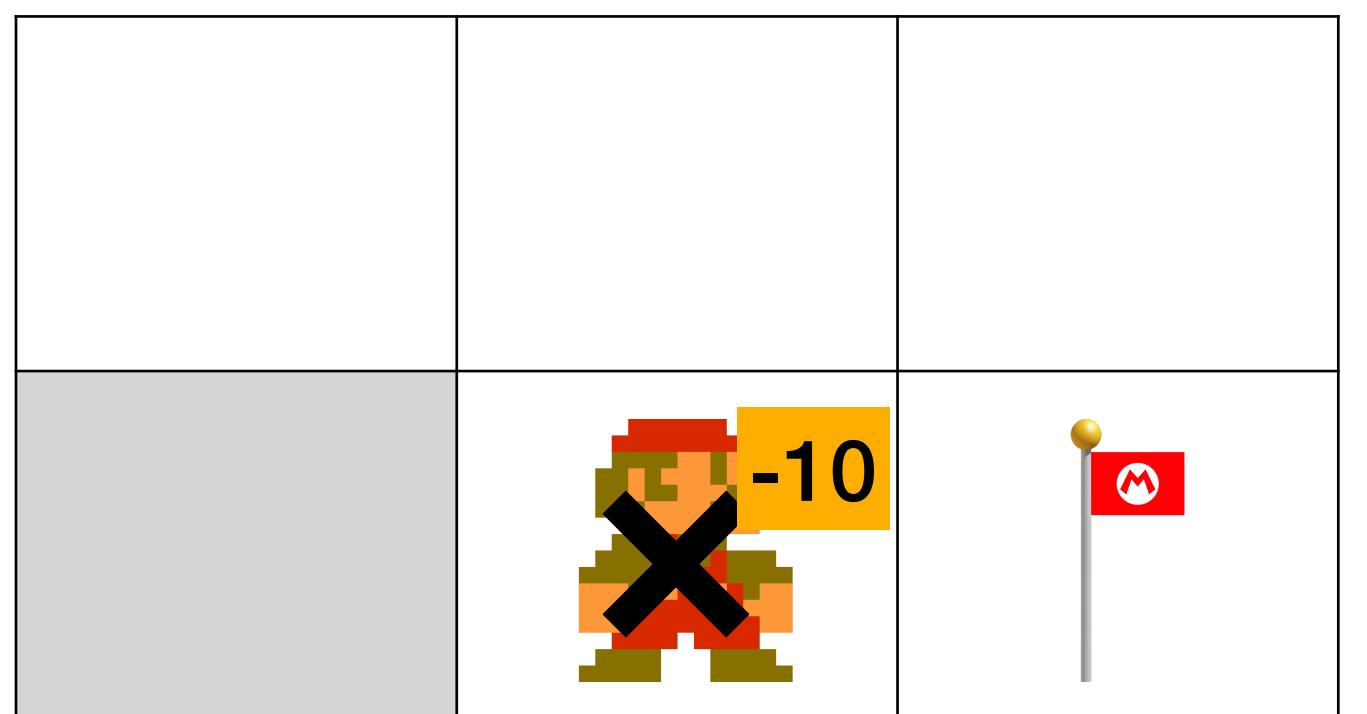


Q-Learning example

$$\alpha = 0.1$$
$$\gamma = 0.99$$

	←	→	↑	↓
Luigi	0	0.1	0	0
Coin	0	0	0	0
Blank	0	0	0	0
Goomba	0	0	0	0
Flag	0	0	0	0

Take action A , observe R, S'



Q-Learning example

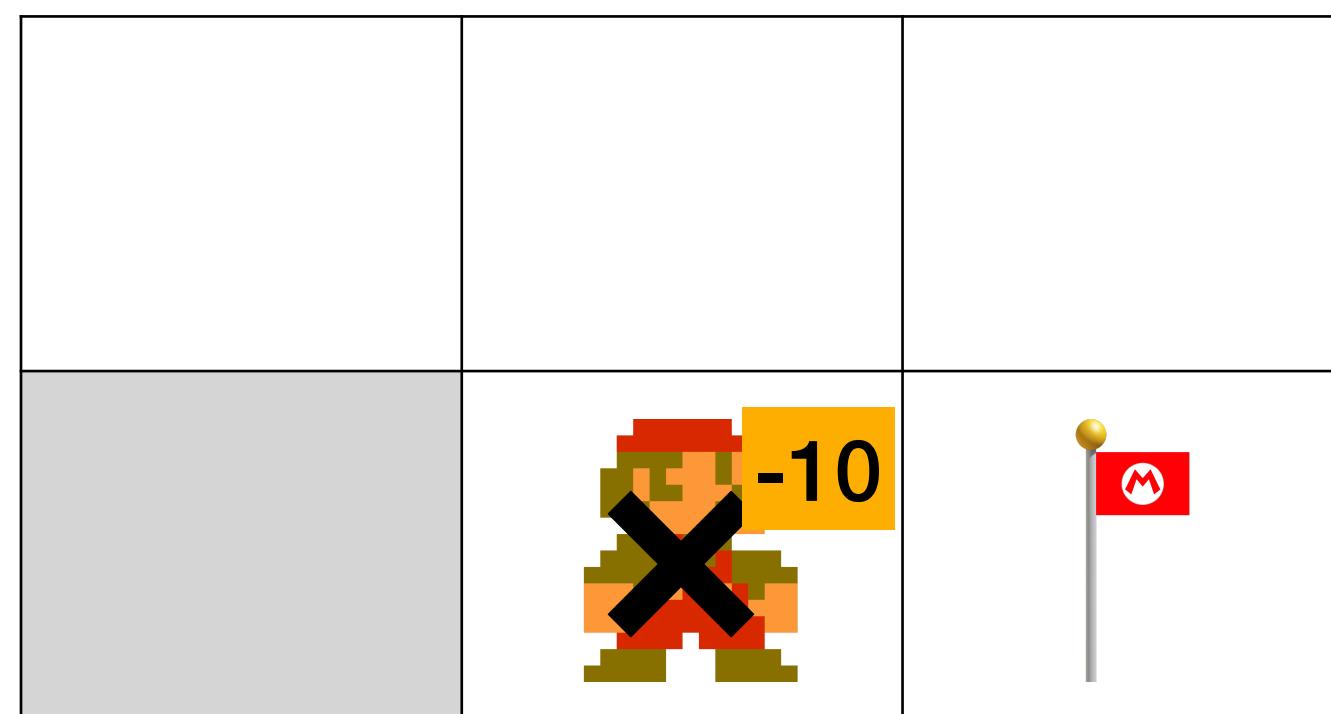
$$\alpha = 0.1$$
$$\gamma = 0.99$$

	←	→	↑	↓
Luigi	0	0.1	0	0
Coin	0	0	0	0
Blank	0	0	0	0
Goomba	0	0	0	0
Flag	0	0	0	0

Update $Q(S, A)$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t))$$

Take action A , observe R, S'



Q-Learning example

$$\alpha = 0.1$$
$$\gamma = 0.99$$

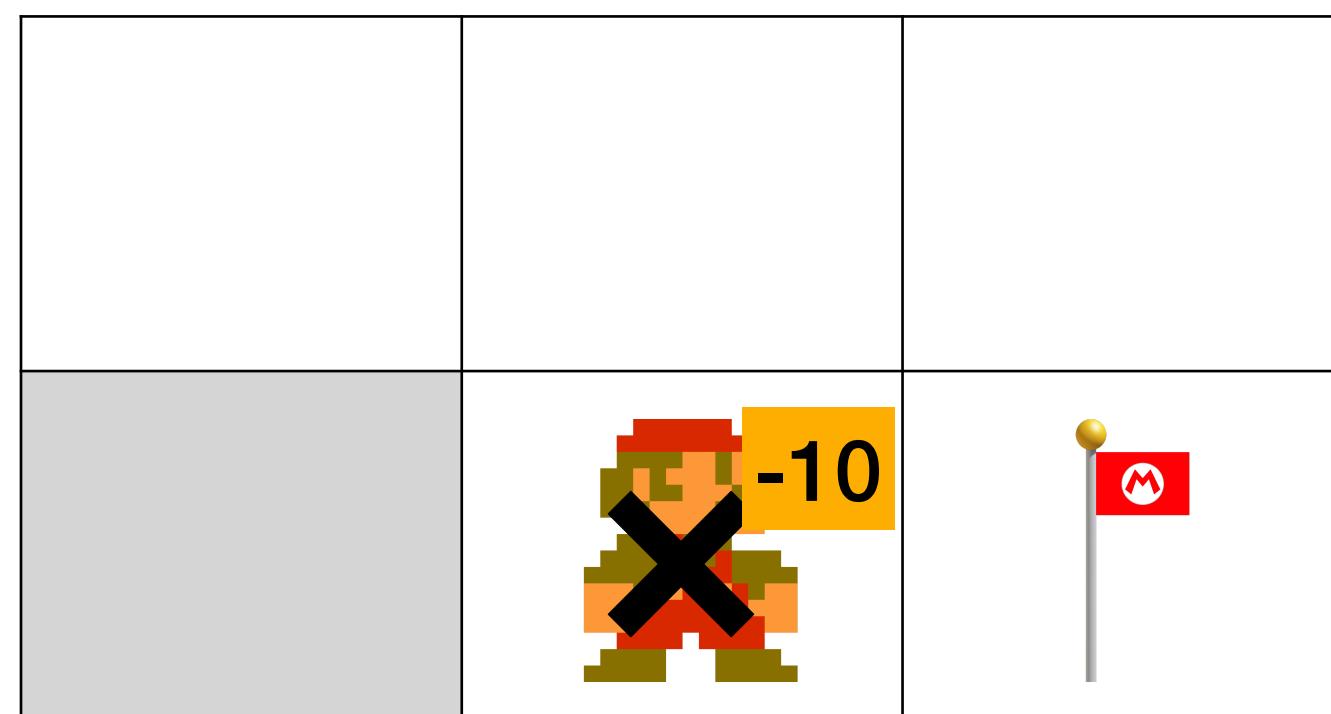
	←	→	↑	↓
Luigi	0	0.1	0	0
Coin	0	0	0	0
Wall	0	0	0	0
Goomba	0	0	0	0
Flag	0	0	0	0

Update $Q(S, A)$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t))$$

$$Q(\text{state_2}, \downarrow) = 0 + 0.1 * [-10 + 0.99 * 0 - 0] = -1$$

Take action A , observe R, S'



Q-Learning example

$$\alpha = 0.1$$
$$\gamma = 0.99$$

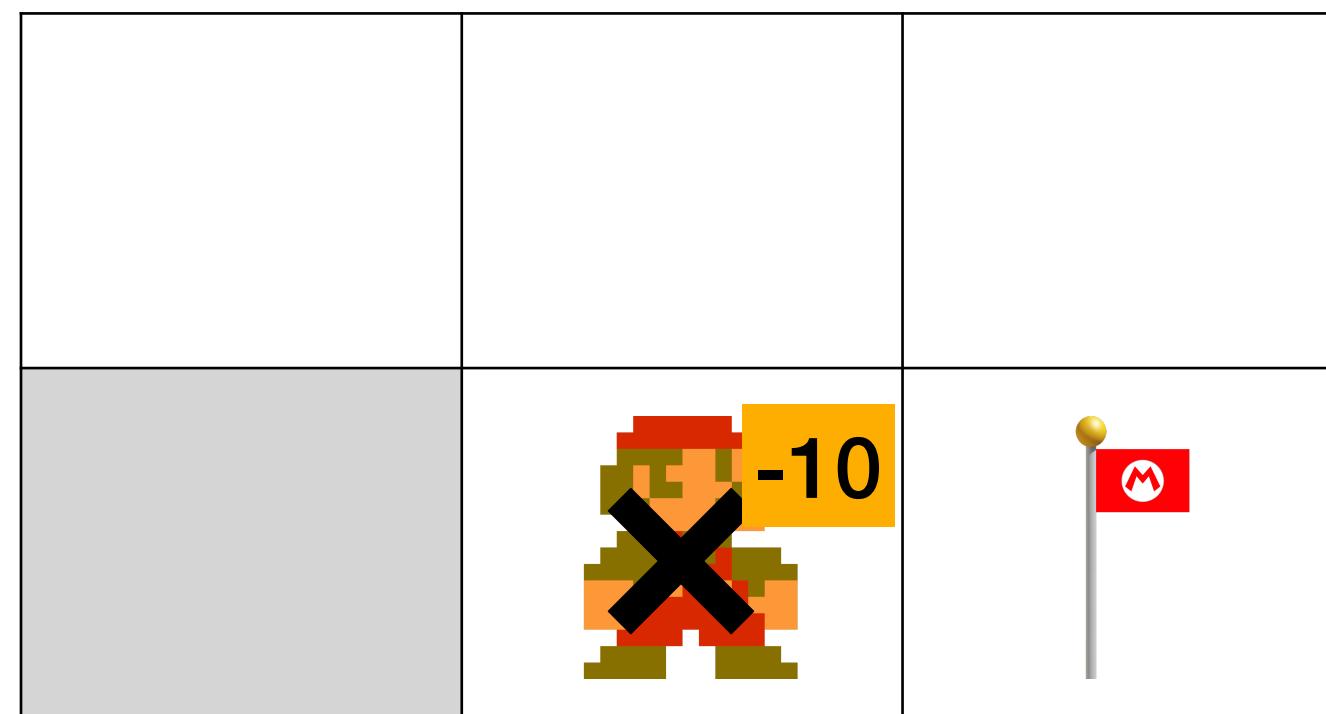
	←	→	↑	↓
Luigi	0	0.1	0	0
Goomba	0	0	0	-1
	0	0	0	0
	0	0	0	0
Koopa Troopa	0	0	0	0
Flag	0	0	0	0

Update $Q(S, A)$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t))$$

$$Q(\text{state_2}, \downarrow) = 0 + 0.1 * [-10 + 0.99 * 0 - 0] = -1$$

Take action A , observe R, S'



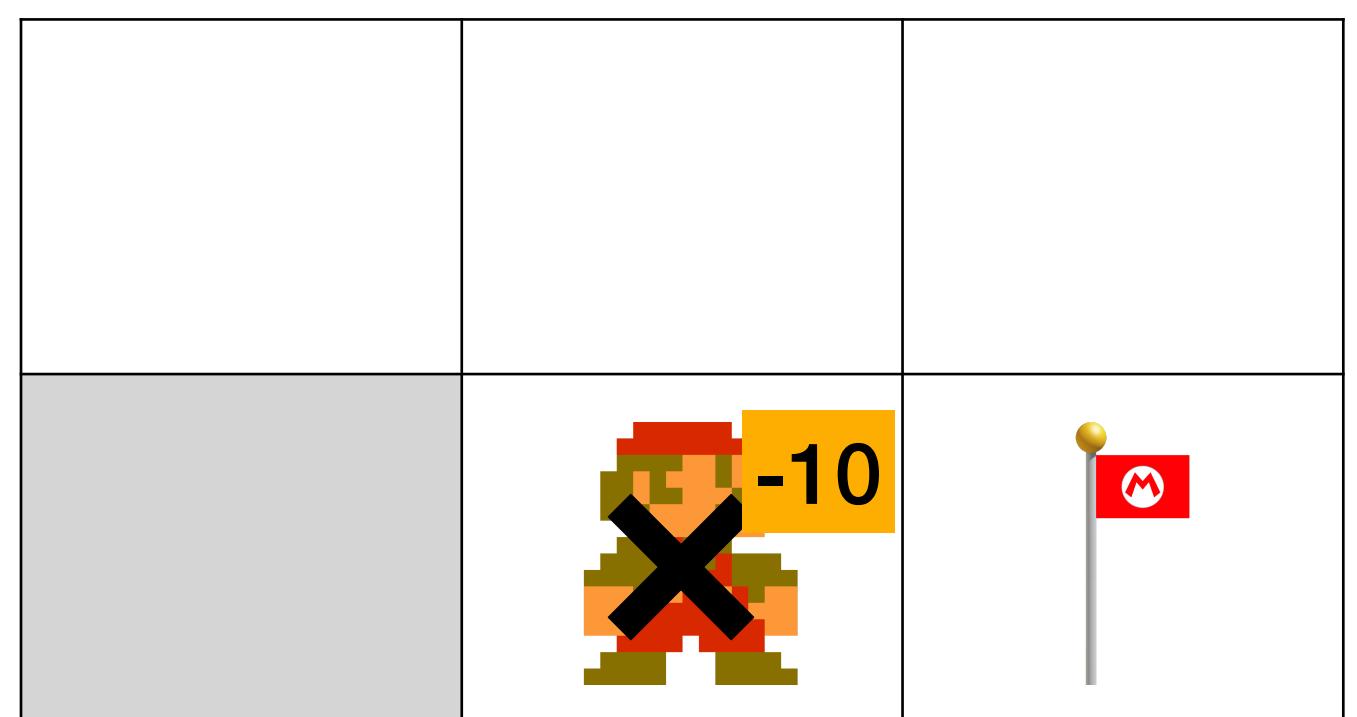
Q-Learning example

$$\alpha = 0.1$$
$$\gamma = 0.99$$

	←	→	↑	↓
Luigi	0	0.1	0	0
Coin	0	0	0	-1
Blank	0	0	0	0
Goomba	0	0	0	0
Flag	0	0	0	0

Terminal state, Episode resets

Take action A , observe R , S'



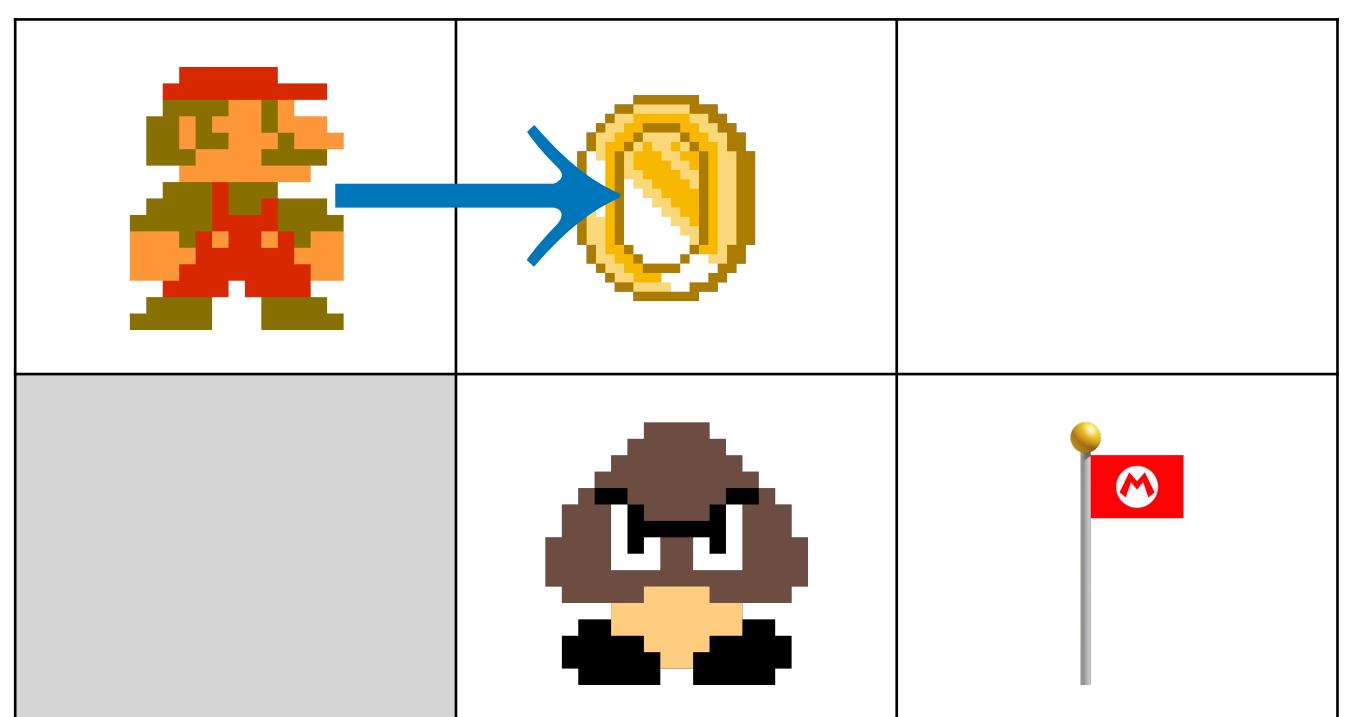
Q-Learning example

New episode

$$\alpha = 0.1$$
$$\gamma = 0.99$$

	←	→	↑	↓
Luigi	0	0.1	0	0
Coin	0	0	0	-1
Blank	0	0	0	0
Goomba	0	0	0	0
Flag	0	0	0	0

Choose A from S using policy derived from Q (e.g., ε -greedy)



Q-Learning example

$$\alpha = 0.1$$
$$\gamma = 0.99$$

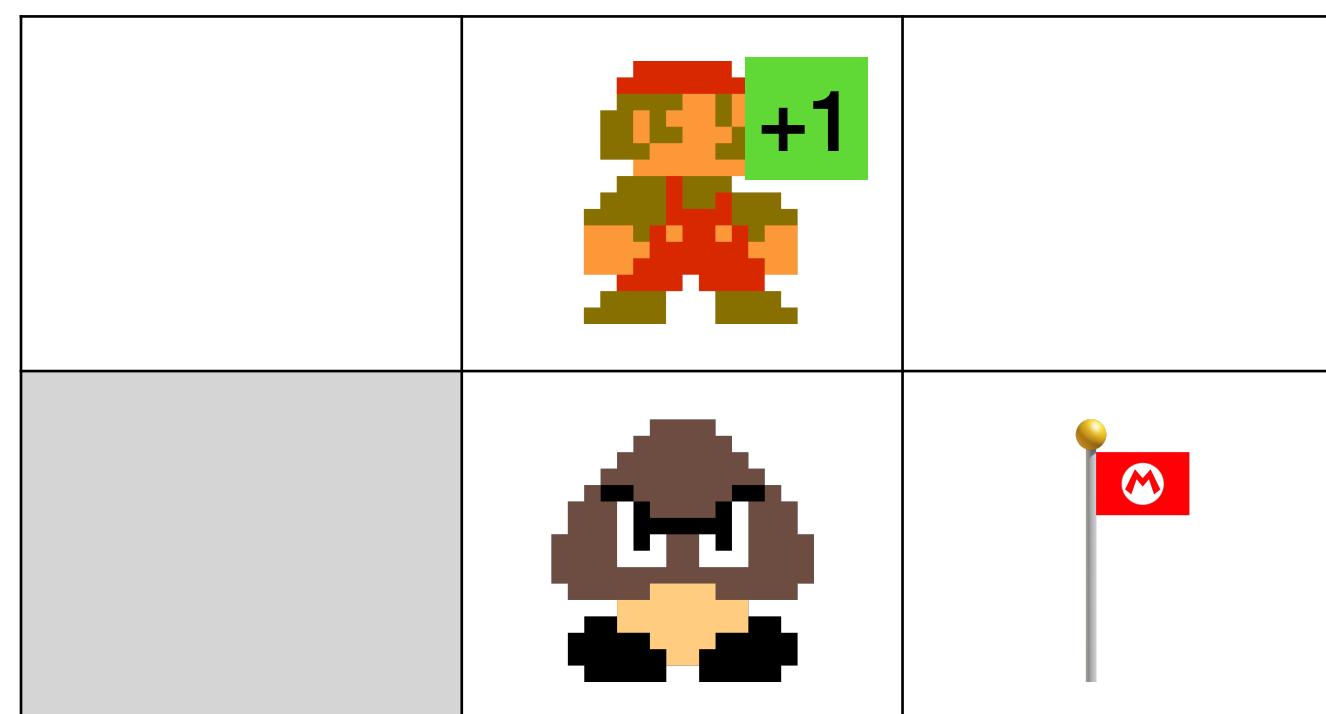
	←	→	↑	↓
Luigi	0	0.1	0	0
Goomba	0	0	0	-1
Blank	0	0	0	0
Blank	0	0	0	0
Blank	0	0	0	0
Flag	0	0	0	0

Update $Q(S, A)$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t))$$

$$Q(\text{initial}, \rightarrow) = 0.1 + 0.1 * [1 + 0.99 * 0 - 0] = 0.2$$

Take action A , observe R, S'



Q-Learning example

$$\alpha = 0.1$$
$$\gamma = 0.99$$

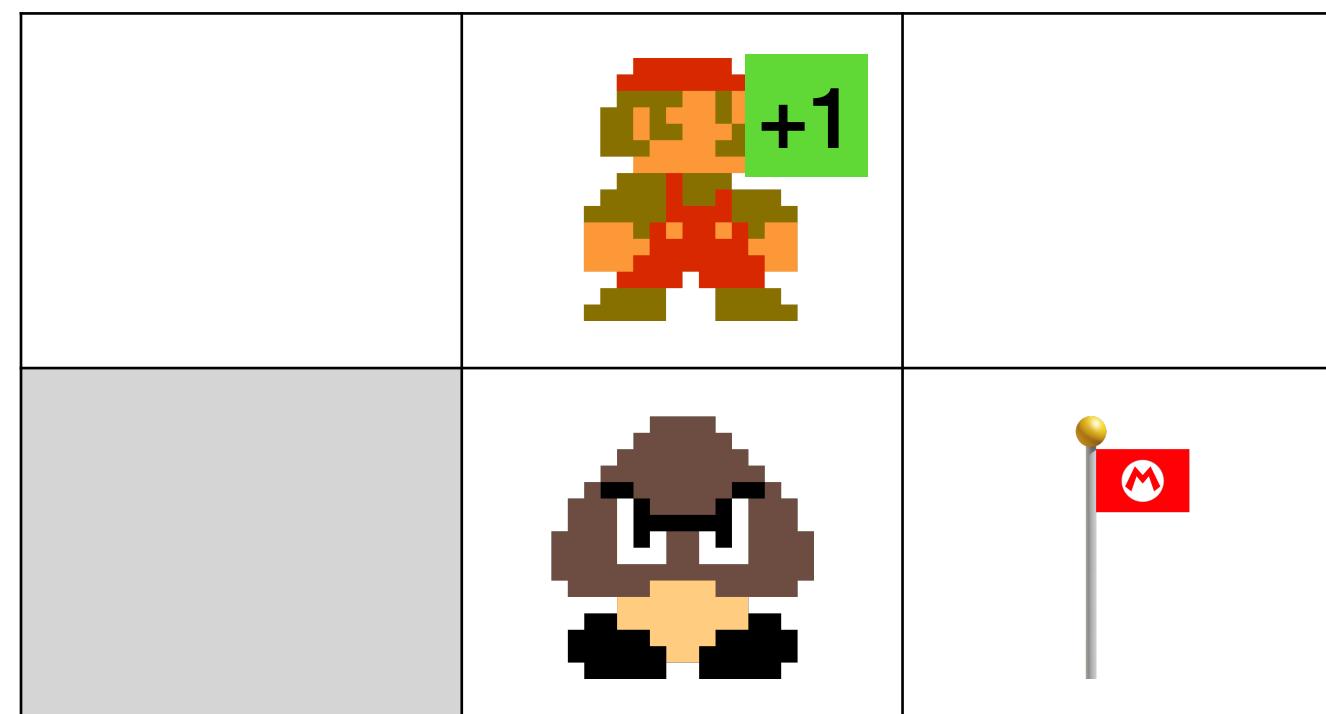
	←	→	↑	↓
Luigi	0	0.2	0	0
Goomba	0	0	0	-1
Blank	0	0	0	0
Blank	0	0	0	0
Flag	0	0	0	0

Update $Q(S, A)$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t))$$

$$Q(\text{initial}, \rightarrow) = 0.1 + 0.1 * [1 + 0.99 * 0 - 0] = 0.2$$

Take action A , observe R, S'

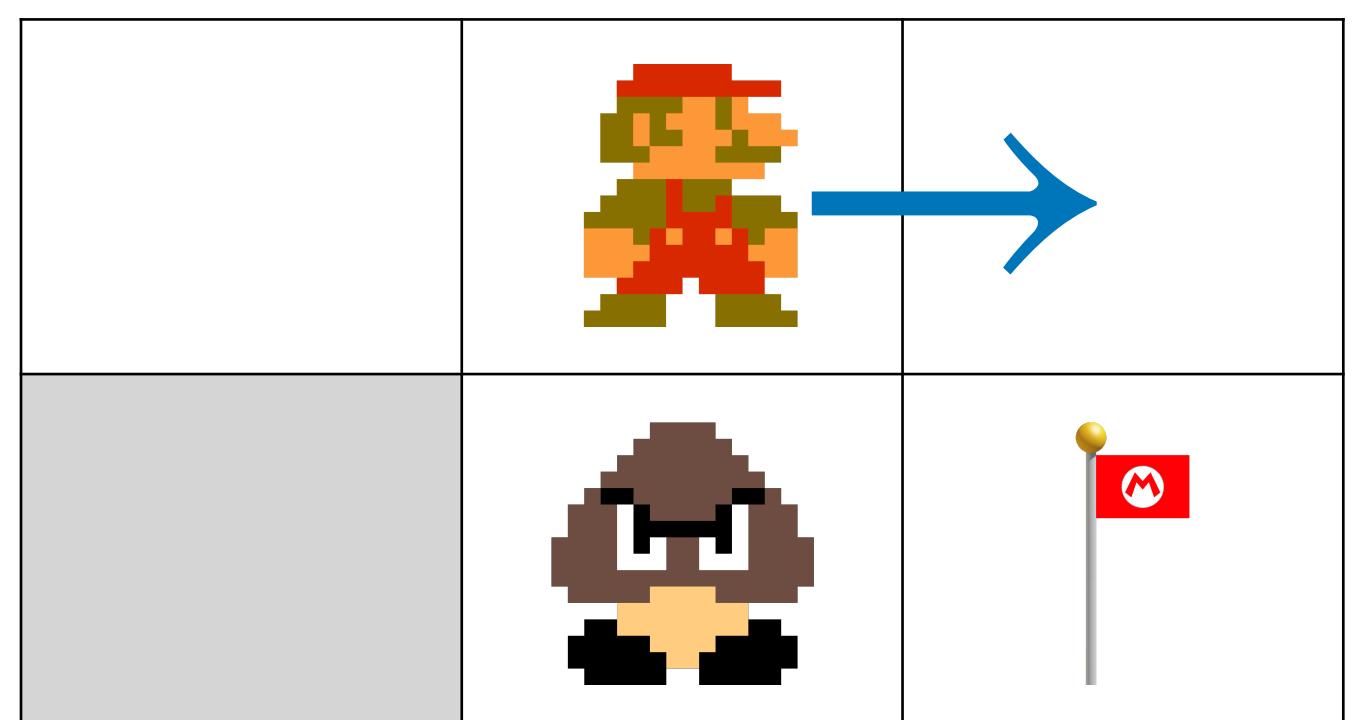


Q-Learning example

$$\alpha = 0.1$$
$$\gamma = 0.99$$

	←	→	↑	↓
Mario	0	0.2	0	0
Goomba	0	0	0	-1
Blank	0	0	0	0
Blank	0	0	0	0
Goomba	0	0	0	0
Flag	0	0	0	0

Choose A from S using policy derived from Q (e.g., ε -greedy)

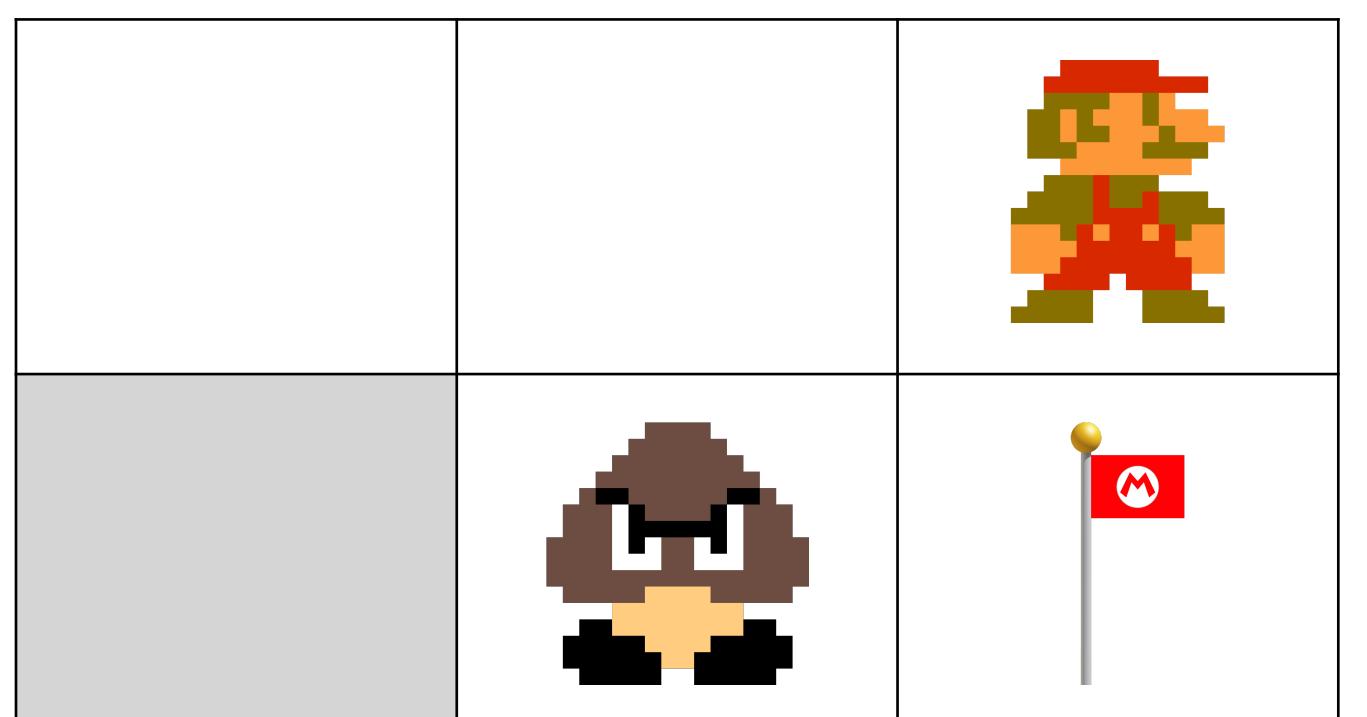


Q-Learning example

$$\alpha = 0.1$$
$$\gamma = 0.99$$

	←	→	↑	↓
Mario	0	0.2	0	0
Goomba	0	0	0	-1
Blank	0	0	0	0
Blank	0	0	0	0
Goomba	0	0	0	0
Flag	0	0	0	0

Take action A , observe R, S'



Q-Learning example

$$\alpha = 0.1$$
$$\gamma = 0.99$$

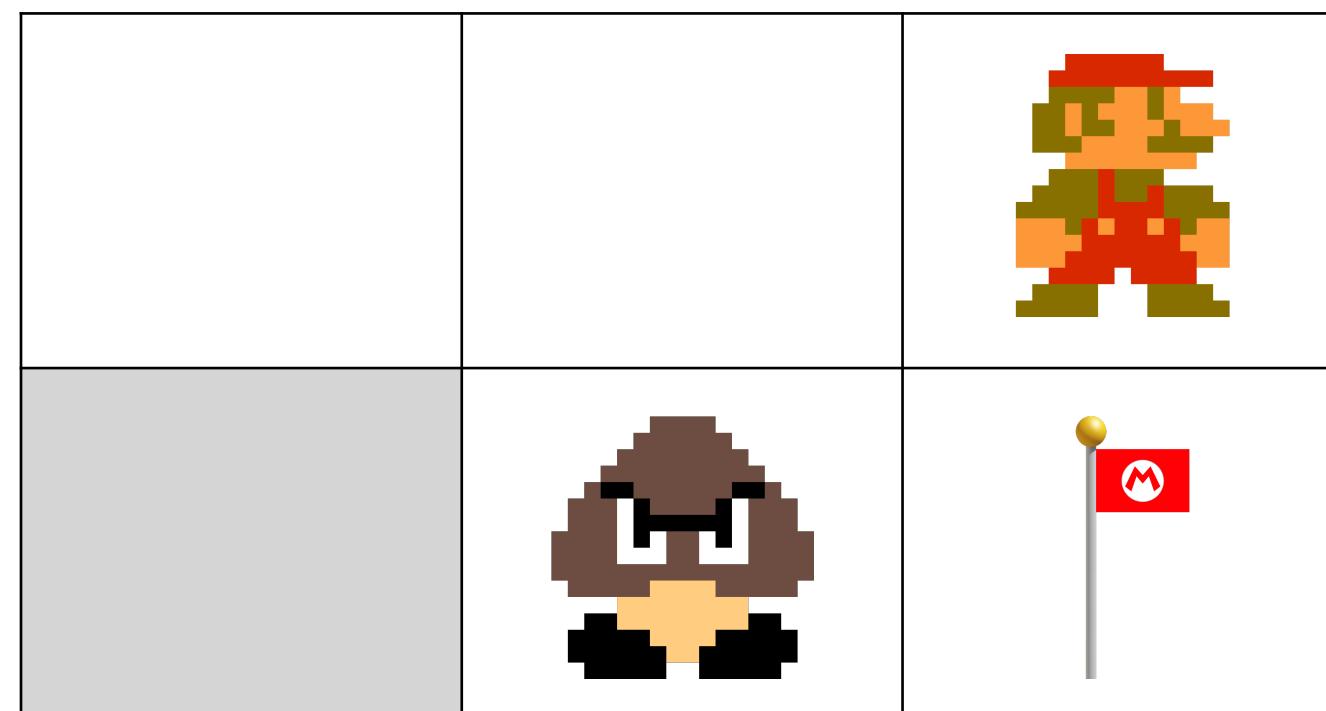
	←	→	↑	↓
Luigi	0	0.2	0	0
Goomba	0	0	0	-1
Blank	0	0	0	0
Luigi	0	0	0	0
Goomba	0	0	0	0
Flag	0	0	0	0

Update $Q(S, A)$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t))$$

$$Q(\text{Goomba}, \rightarrow) = 0 + 0.1 * [0 + 0.99 * 0 - 0] = 0$$

Take action A , observe R, S'



Q-Learning example

$$\alpha = 0.1$$
$$\gamma = 0.99$$

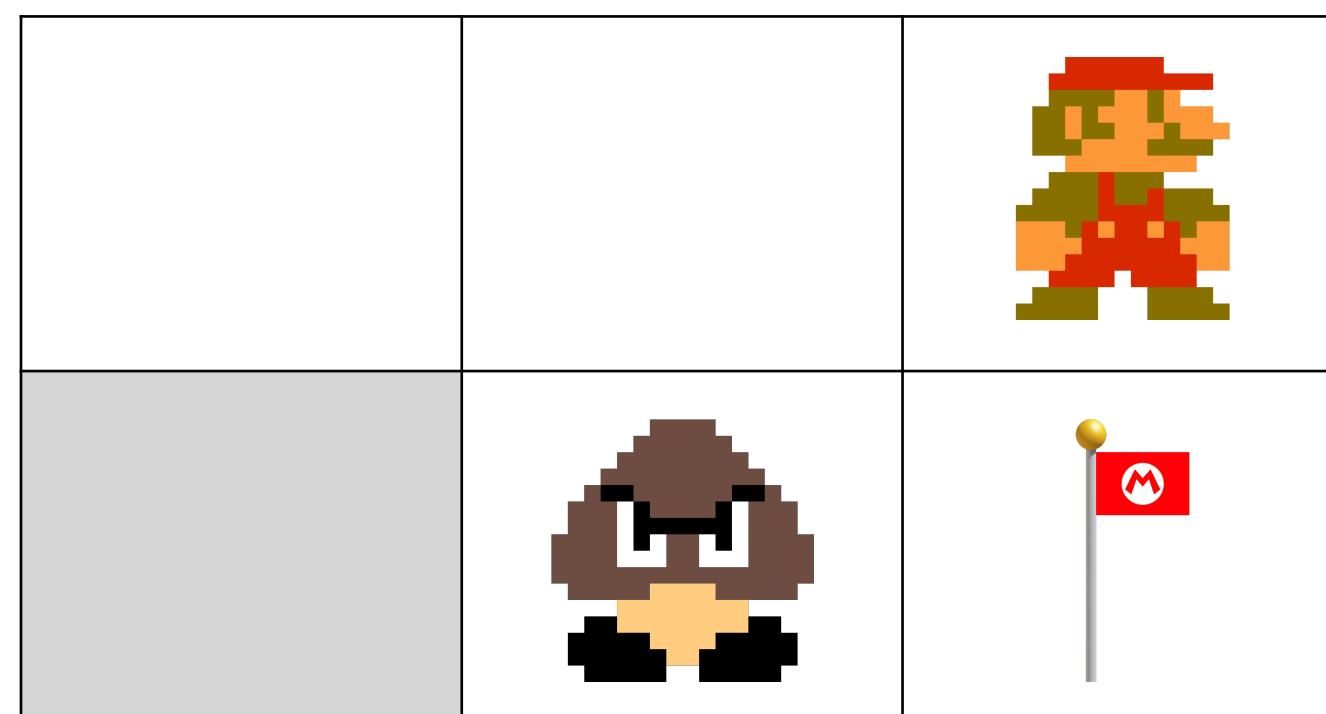
	←	→	↑	↓
Luigi	0	0.2	0	0
Goomba	0	0	0	-1
Blank	0	0	0	0
Luigi	0	0	0	0
Flag	0	0	0	0

Update $Q(S, A)$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t))$$

$$Q(\text{Goomba}, \rightarrow) = 0 + 0.1 * [0 + 0.99 * 0 - 0] = 0$$

Take action A , observe R, S'

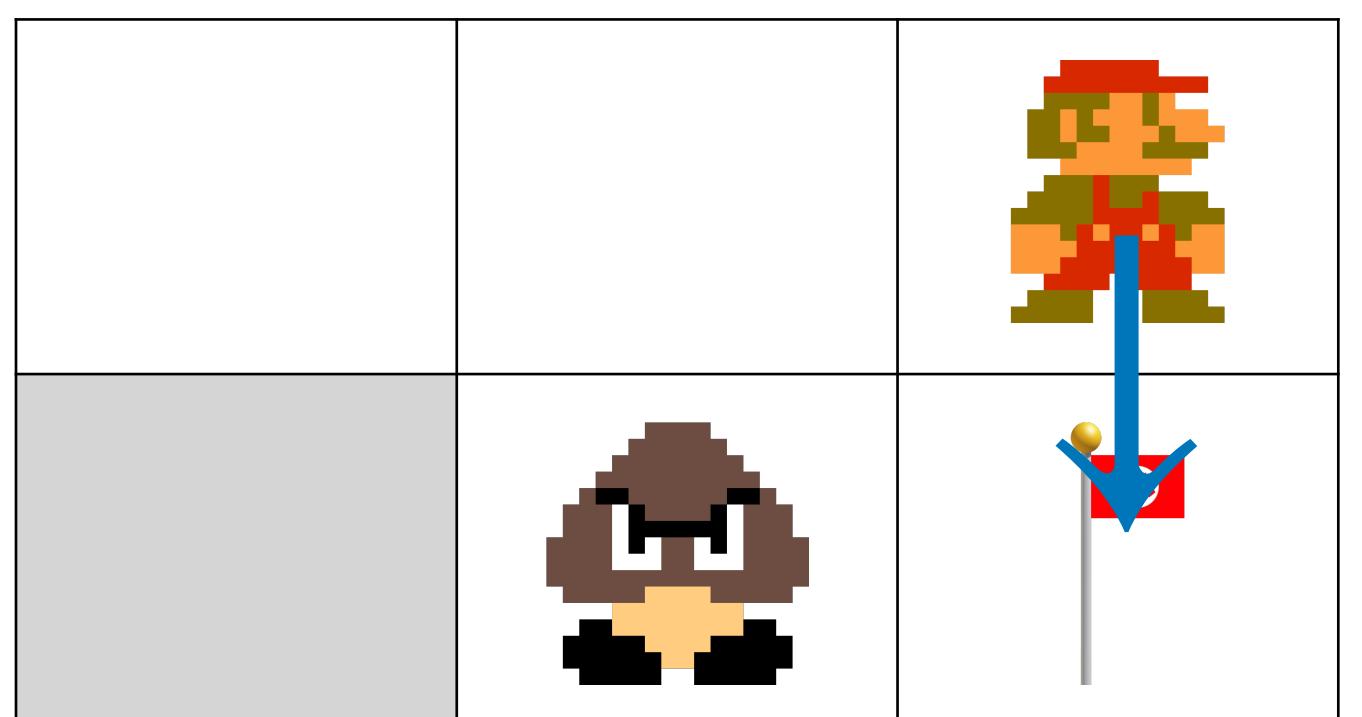


Q-Learning example

$$\alpha = 0.1$$
$$\gamma = 0.99$$

	←	→	↑	↓
Luigi	0	0.2	0	0
Goomba	0	0	0	-1
Blank	0	0	0	0
Blank	0	0	0	0
Goomba	0	0	0	0
Flag	0	0	0	0

Choose A from S using policy derived from Q (e.g., ε -greedy)

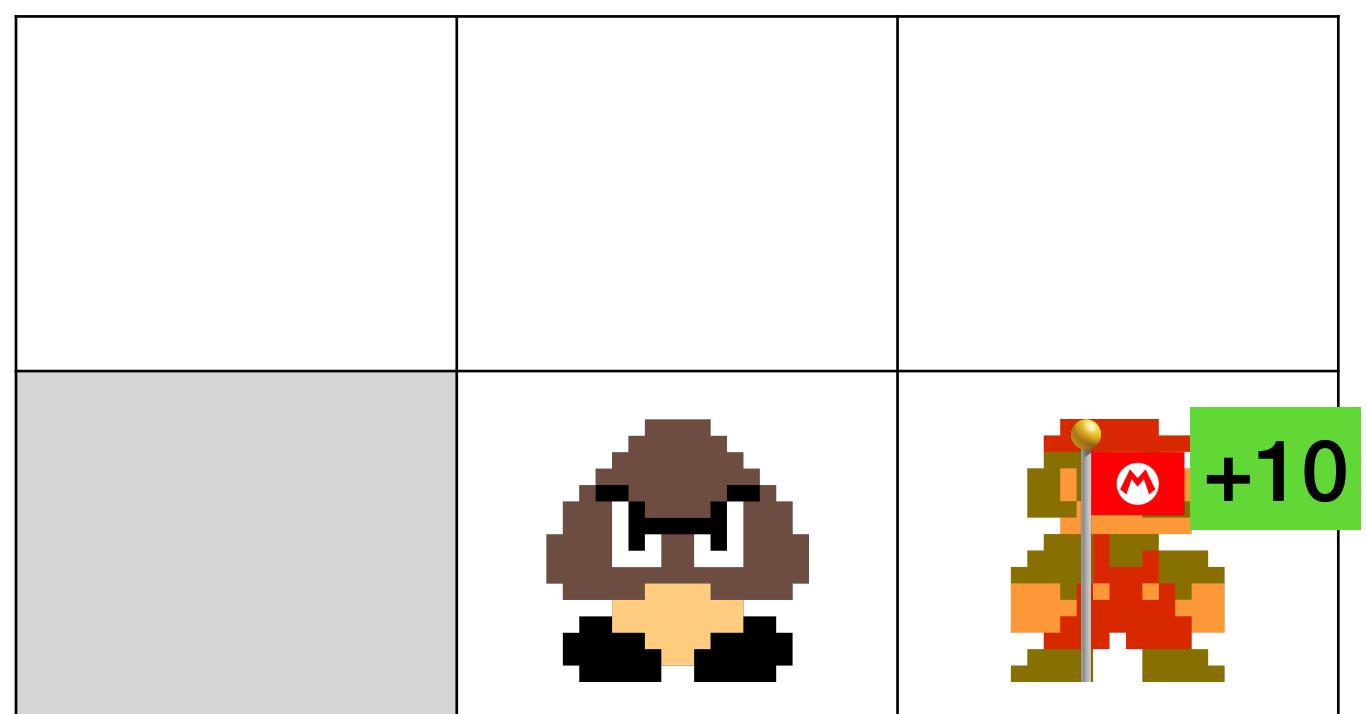


Q-Learning example

$$\alpha = 0.1$$
$$\gamma = 0.99$$

	←	→	↑	↓
Mario	0	0.2	0	0
Goomba	0	0	0	-1
Blank	0	0	0	0
Blank	0	0	0	0
Goomba	0	0	0	0
Flag	0	0	0	0

Take action A , observe R, S'



Q-Learning example

$$\alpha = 0.1$$
$$\gamma = 0.99$$

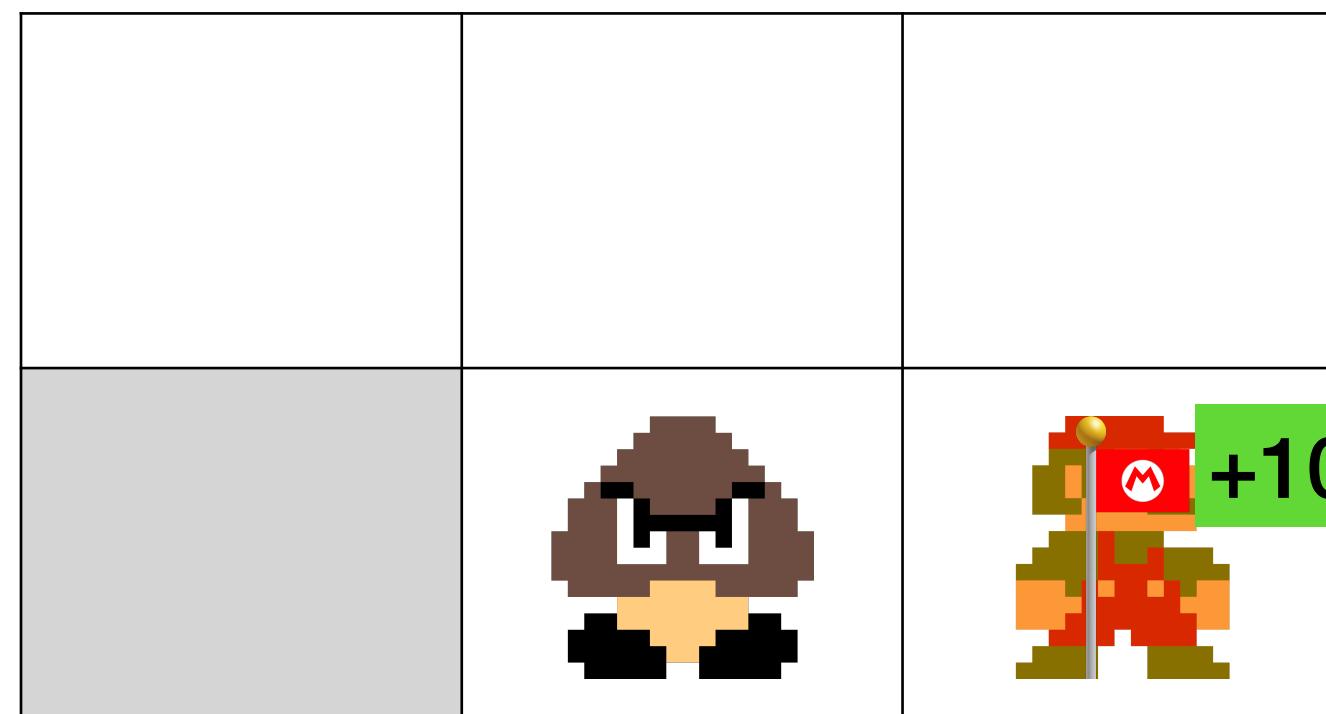
	←	→	↑	↓
Luigi	0	0.2	0	0
Coin	0	0	0	-1
	0	0	0	0
	0	0	0	0
Goomba	0	0	0	0
Flag	0	0	0	0

Update $Q(S, A)$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t))$$

$$Q(\text{ Luigi}, \downarrow) = 0 + 0.1 * [10 + 0.99 * 0 - 0] = 1$$

Take action A , observe R, S'



Q-Learning example

$$\alpha = 0.1$$
$$\gamma = 0.99$$

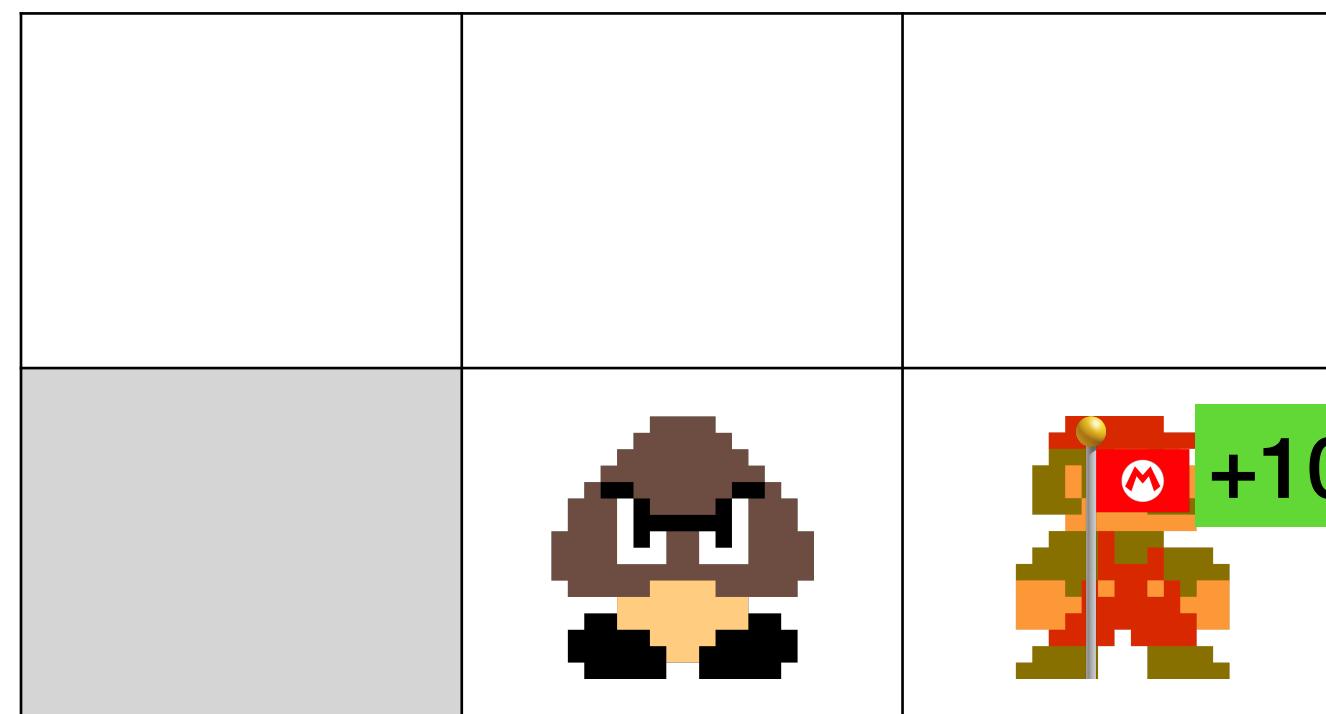
	←	→	↑	↓
Luigi	0	0.2	0	0
Coin	0	0	0	-1
	0	0	0	1
Goomba	0	0	0	0
Flag	0	0	0	0

Update $Q(S, A)$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t))$$

$$Q(\text{ Luigi}, \downarrow) = 0 + 0.1 * [10 + 0.99 * 0 - 0] = 1$$

Take action A , observe R, S'



Q-Learning example

$$\alpha = 0.1$$
$$\gamma = 0.99$$

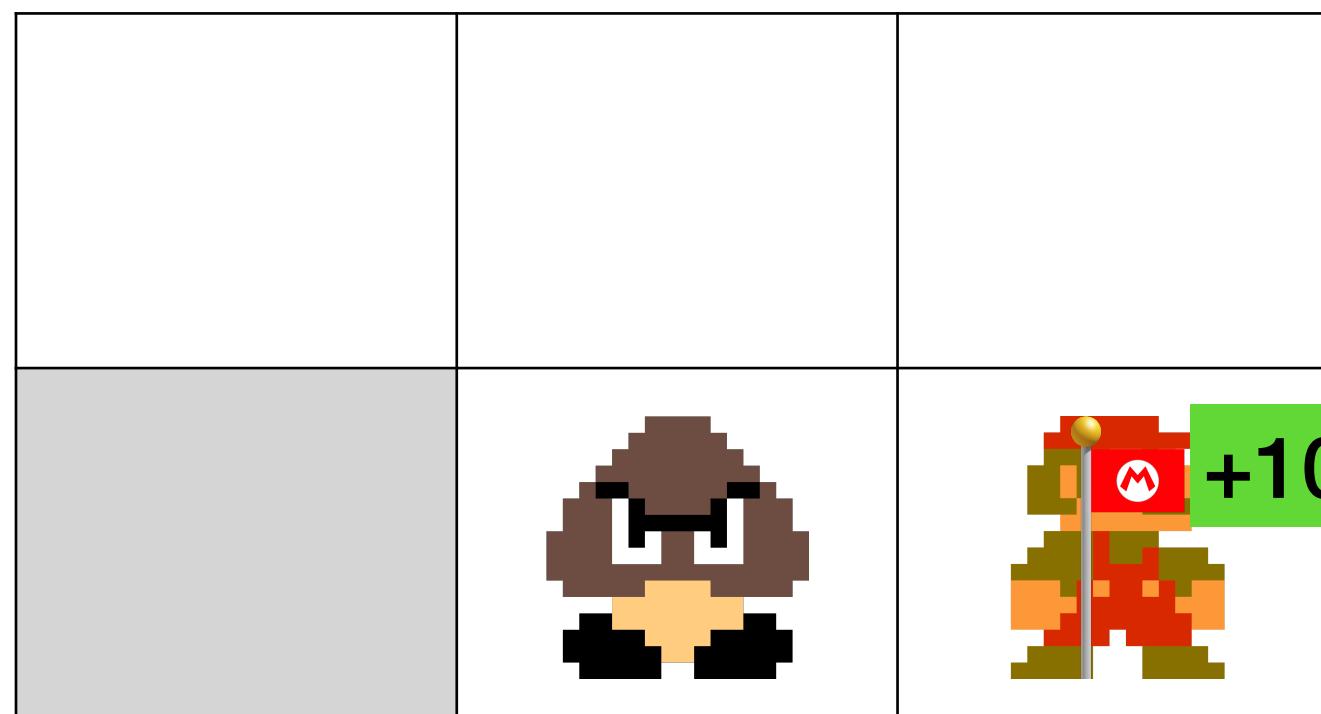
	←	→	↑	↓
Luigi	0	0.2	0	0
Goomba	0	0	0	-1
	0	0	0	1
Flag	0	0	0	0
Goomba	0	0	0	0
Flag	0	0	0	0

Update $Q(S, A)$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t))$$

$$Q(\text{ Luigi}, \downarrow) = 0 + 0.1 * [10 + 0.99 * 0 - 0] = 1$$

Take action A , observe R, S'



Episode won! Terminal state

Why ε -greedy?

Why ϵ -greedy?

Exploration-exploitation tradeoff

Why ε -greedy?

Exploration-exploitation tradeoff

Take greedy actions with probability $1 - \varepsilon$

Why ϵ -greedy?

Exploration-exploitation tradeoff

Take greedy actions with probability $1 - \epsilon$

Exploit what the agent already knows

Why ϵ -greedy?

Exploration-exploitation tradeoff

Take greedy actions with probability $1 - \epsilon$

Exploit what the agent already knows

Explore with probability ϵ

Why ϵ -greedy?

Exploration-exploitation tradeoff

Take greedy actions with probability $1 - \epsilon$

Exploit what the agent already knows

Explore with probability ϵ

Take random actions to visit new states you would not visit otherwise

Why ϵ -greedy?

Exploration-exploitation tradeoff

Take greedy actions with probability $1 - \epsilon$

Exploit what the agent already knows

Explore with probability ϵ

Take random actions to visit new states you would not visit otherwise

Decrease ϵ with time

Why ε -greedy?

Exploration-exploitation tradeoff

Take greedy actions with probability $1 - \varepsilon$

Exploit what the agent already knows

Explore with probability ε

Take random actions to visit new states you would not visit otherwise

Decrease ε with time

$$0 \leq \varepsilon \leq 1$$

Why ϵ -greedy?

Start	0	0	0	0	0	0
-1						0
10		1	1	1		0
1		1		1		0
1	1	1		1	1	Goal (100)

Why ϵ -greedy?

$$Q(s, a) = Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$$

Start	0	0	0	0	0	0
-1						0
10		1	1	1		0
1		1		1		0
1	1	1		1	1	Goal (100)

Why ϵ -greedy?

$$Q(s, a) = Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$$

If we take only greedy actions...

Start	0	0	0	0	0	0
-1						0
10		1	1	1		0
1		1		1		0
1	1	1		1	1	Goal (100)

Why ϵ -greedy?

$$Q(s, a) = Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$$

If we take only greedy actions...

Start	0	0	0	0	0	0
-1						0
10		1	1	1		0
1		1		1		0
1	1	1		1	1	Goal (100)

Why ϵ -greedy?

$$Q(s, a) = Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$$

If we take only greedy actions...

We end up with a suboptimal solution

Start	0	0	0	0	0	0
-1						0
10		1	1	1		0
1		1		1		0
1	1	1		1	1	Goal (100)

Why ϵ -greedy?

$$Q(s, a) = Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$$

If we take only greedy actions...

We end up with a suboptimal solution

Start	0	0	0	0	0	0
-1						0
10		1	1	1		0
1		1		1		0
1	1	1		1	1	Goal (100)

G = 100

Why ϵ -greedy?

$$Q(s, a) = Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$$

Start	0	0	0	0	0	0
-1						0
10		1	1	1		0
1		1		1		0
1	1	1		1	1	Goal (100)

Why ϵ -greedy?

$$Q(s, a) = Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$$

Start	0	0	0	0	0	0
-1						0
10		1	1	1		0
1		1		1		0
1	1	1		1	1	Goal (100)

Why ϵ -greedy?

$$Q(s, a) = Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$$

If we explore...

Start	0	0	0	0	0	0
-1						0
10		1	1	1		0
1		1		1		0
1	1	1		1	1	Goal (100)

Why ϵ -greedy?

$$Q(s, a) = Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$$

If we explore...

We might discover more valuable states

Start	0	0	0	0	0	0
-1						0
10		1	1	1		0
1		1		1		0
1	1	1		1	1	Goal (100)

Why ϵ -greedy?

$$Q(s, a) = Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$$

If we explore...

We might discover more valuable states

Start	0	0	0	0	0	0
-1						0
10		1	1	1		0
1		1		1		0
1	1	1		1	1	Goal (100)

Why ϵ -greedy?

$$Q(s, a) = Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$$

If we explore...

We might discover more valuable states

Start	0	0	0	0	0	0
-1						0
10		1	1	1		0
1		1		1		0
1	1	1		1	1	Goal (100)

That lead to higher returns

Why ϵ -greedy?

$$Q(s, a) = Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$$

If we explore...

We might discover more valuable states

Start	0	0	0	0	0	0
-1						0
10		1	1	1		0
1		1		1		0
1	1	1		1	1	Goal (100)

That lead to higher returns

Why ϵ -greedy?

$$Q(s, a) = Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$$

If we explore...

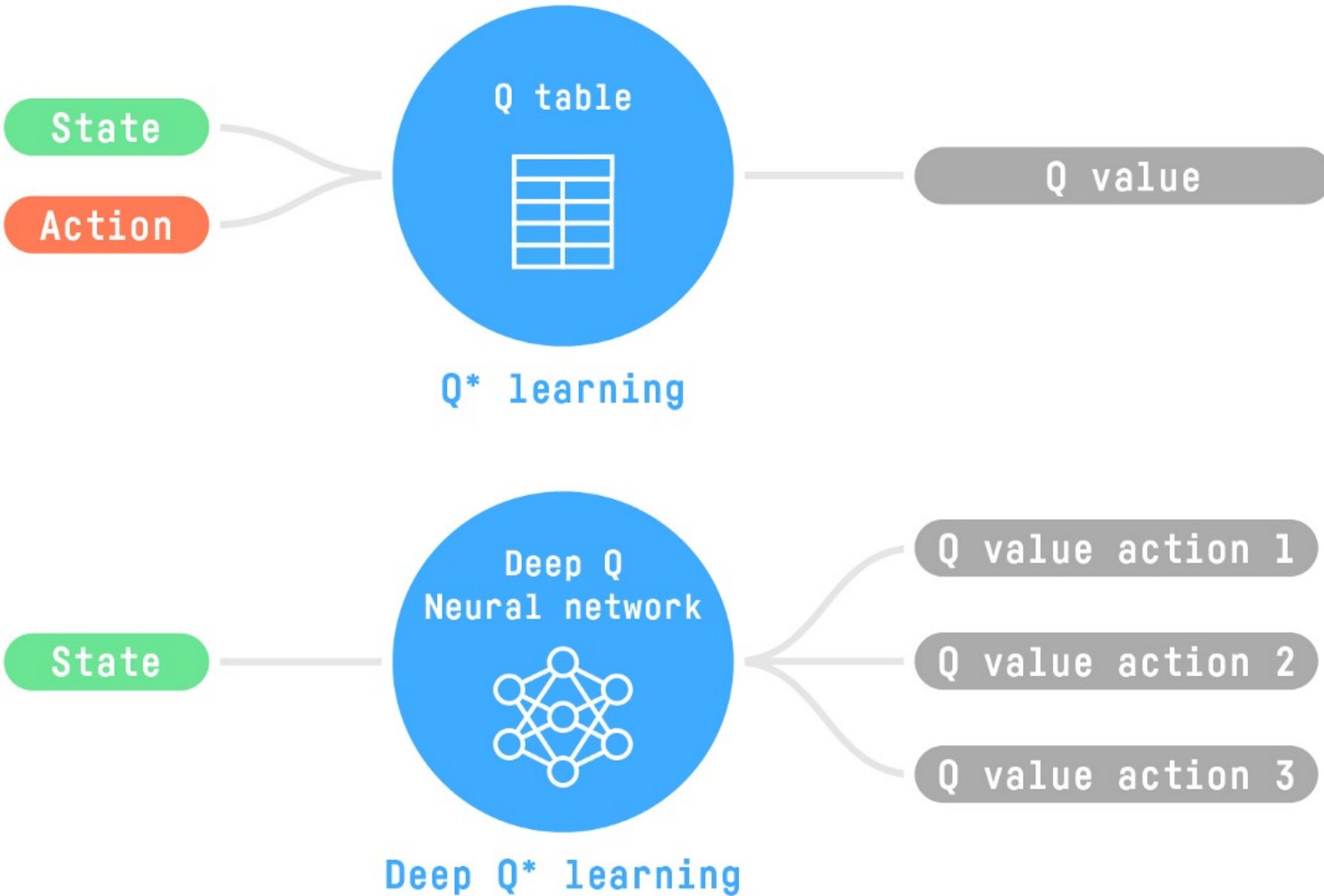
We might discover more valuable states

Start	0	0	0	0	0	0
-1						0
10		1	1	1		0
1		1		1		0
1	1	1		1	1	Goal (100)

That lead to higher returns

$$G = 121$$

Deep Q-Learning



Deep Q-Learning

Q-Learning works well for small state spaces

Atari games, however, have observation space of shape (210, 160, 3), values in [0, 255]

This gives us $256^{210 \times 160 \times 3} = 256^{100800}$ possible observations

Approximate Q-values using a neural network

Parametrized Q-function $Q_\theta(s, a)$

Deep Q-Learning

For episode = 1, M **do**

 Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$

For $t = 1, T$ **do**

 With probability ε select a random action a_t

 otherwise select $a_t = \operatorname{argmax}_a Q(\phi(s_t), a; \theta)$

 Execute action a_t in emulator and observe reward r_t and image x_{t+1}

 Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$

 Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D

 Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D

 Set $y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^-) & \text{otherwise} \end{cases}$

 Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ with respect to the network parameters θ

 Every C steps reset $\hat{Q} = Q$

End For

End For