Reinforcement Learning

Introduction

Antonio Ricciardi, PhD

GLADIA

SAPIENZA
UNIVERSITÀ DI ROMA

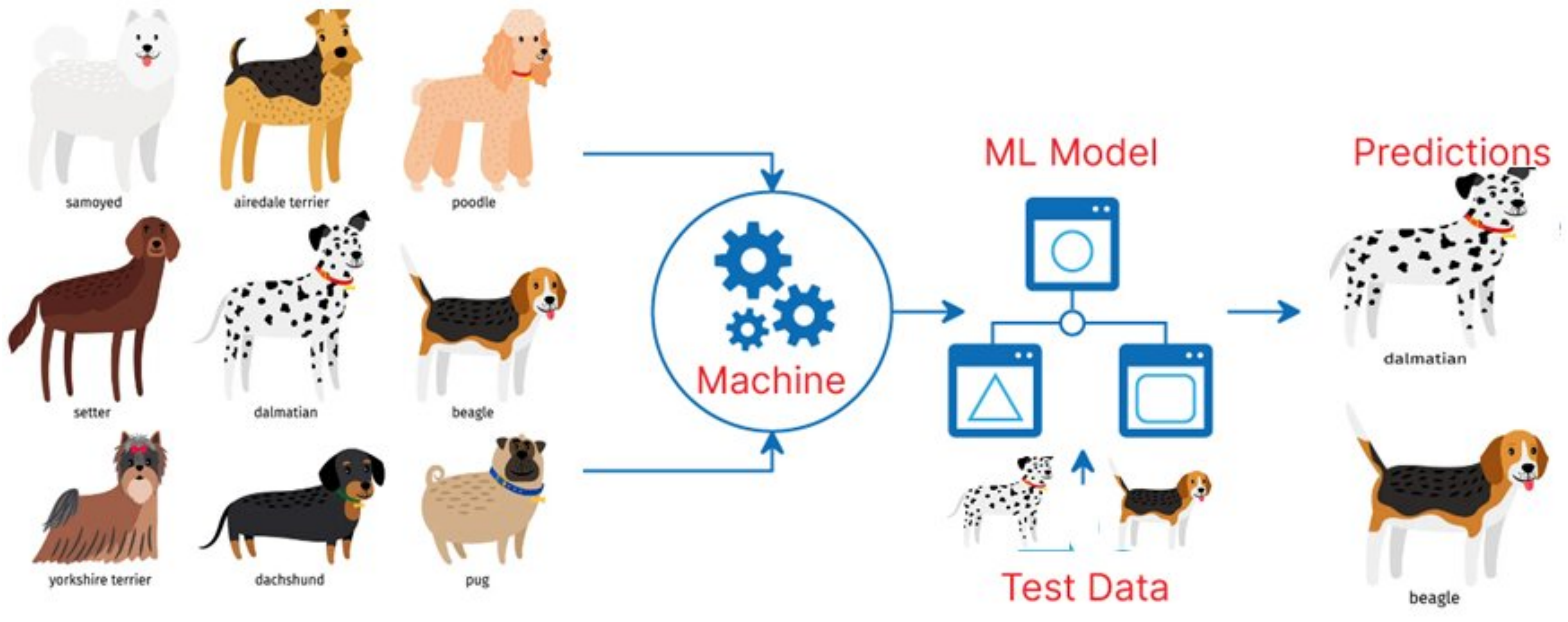
Machine Learning @ Mathematical Sciences for AI

Supervised Learning vs Reinforcement Learning

Supervised Learning

Learn using labeled data

Need pairs $\{x_i, y_i\}$



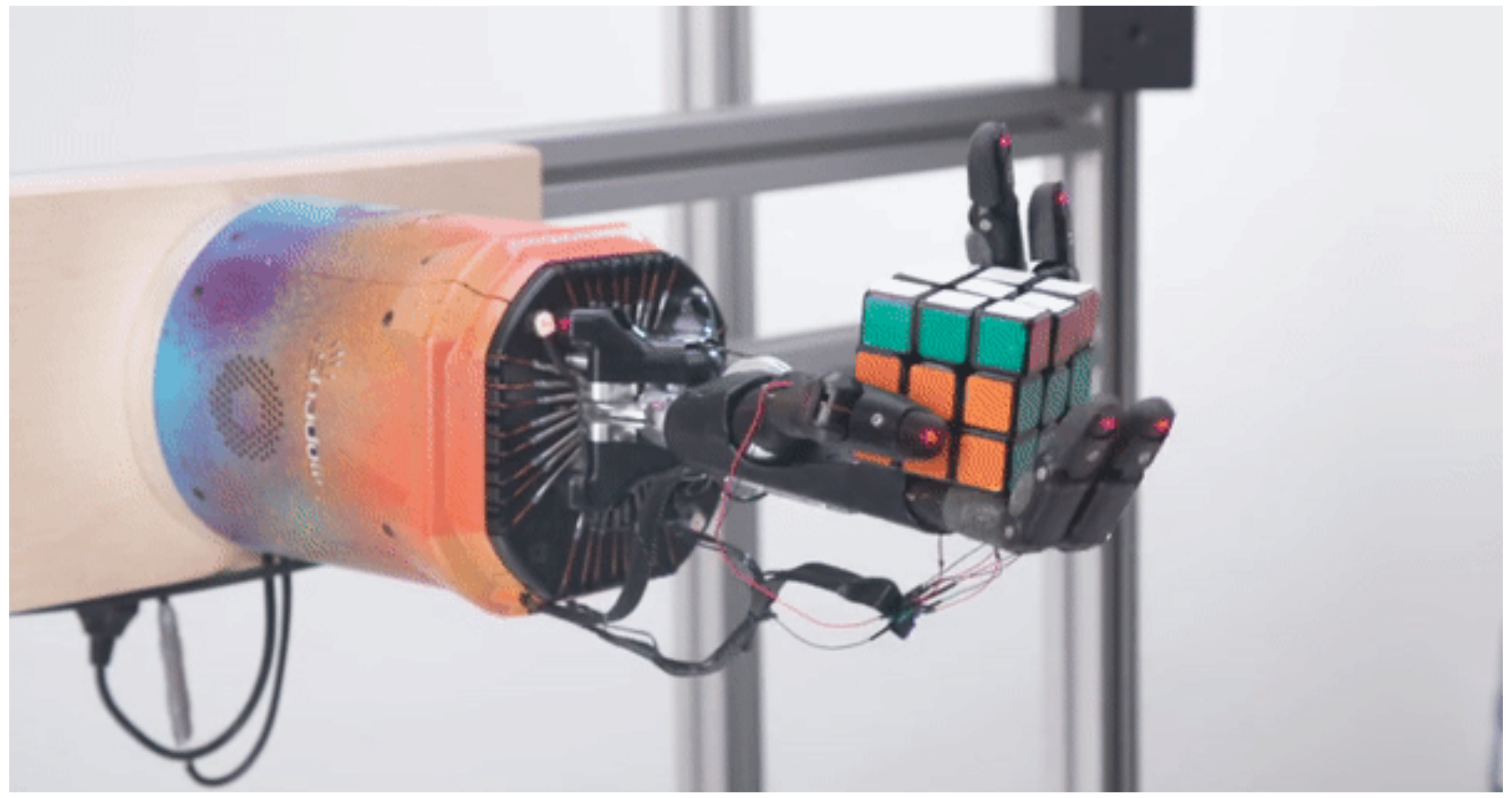
Reinforcement Learning

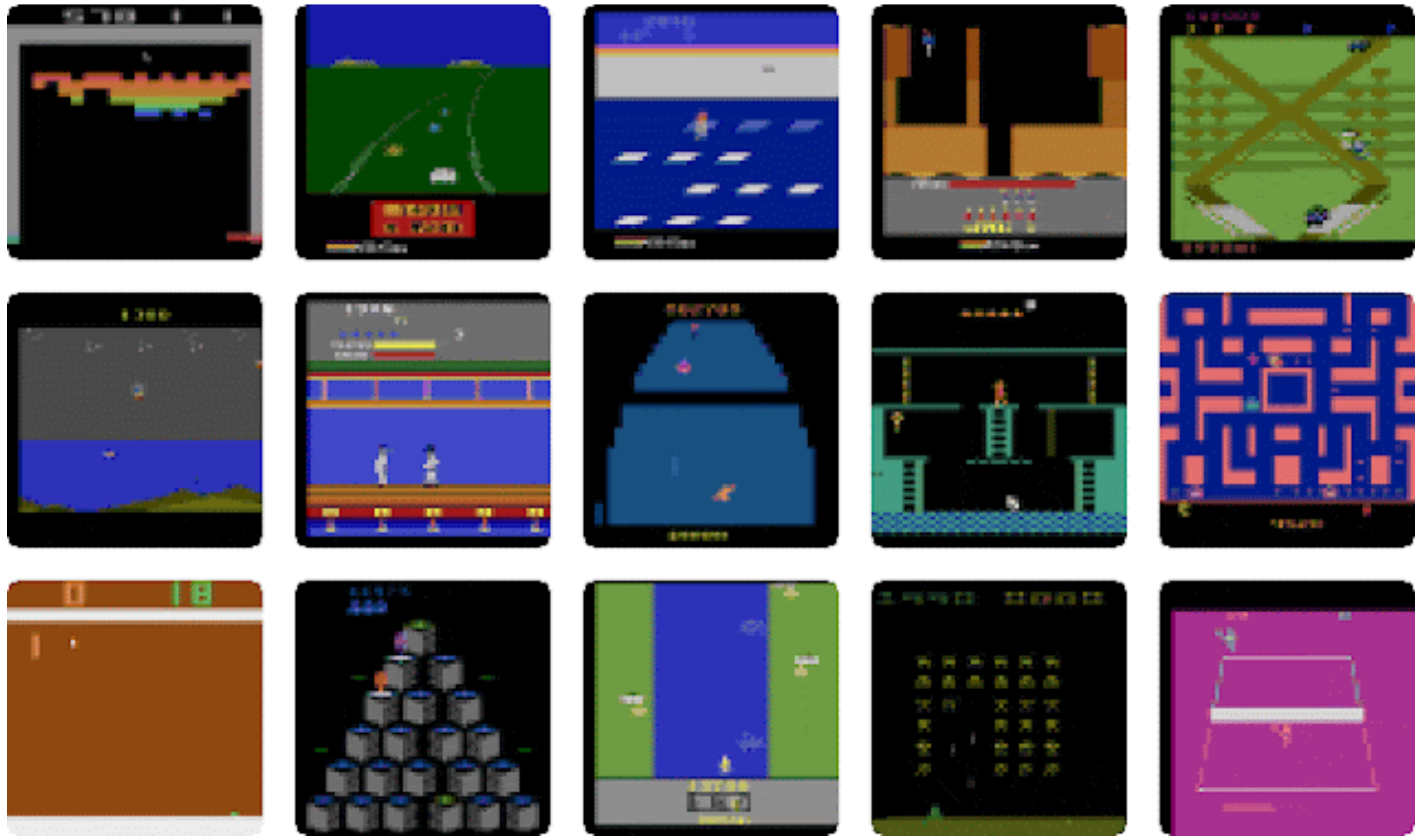
Framework for solving control tasks

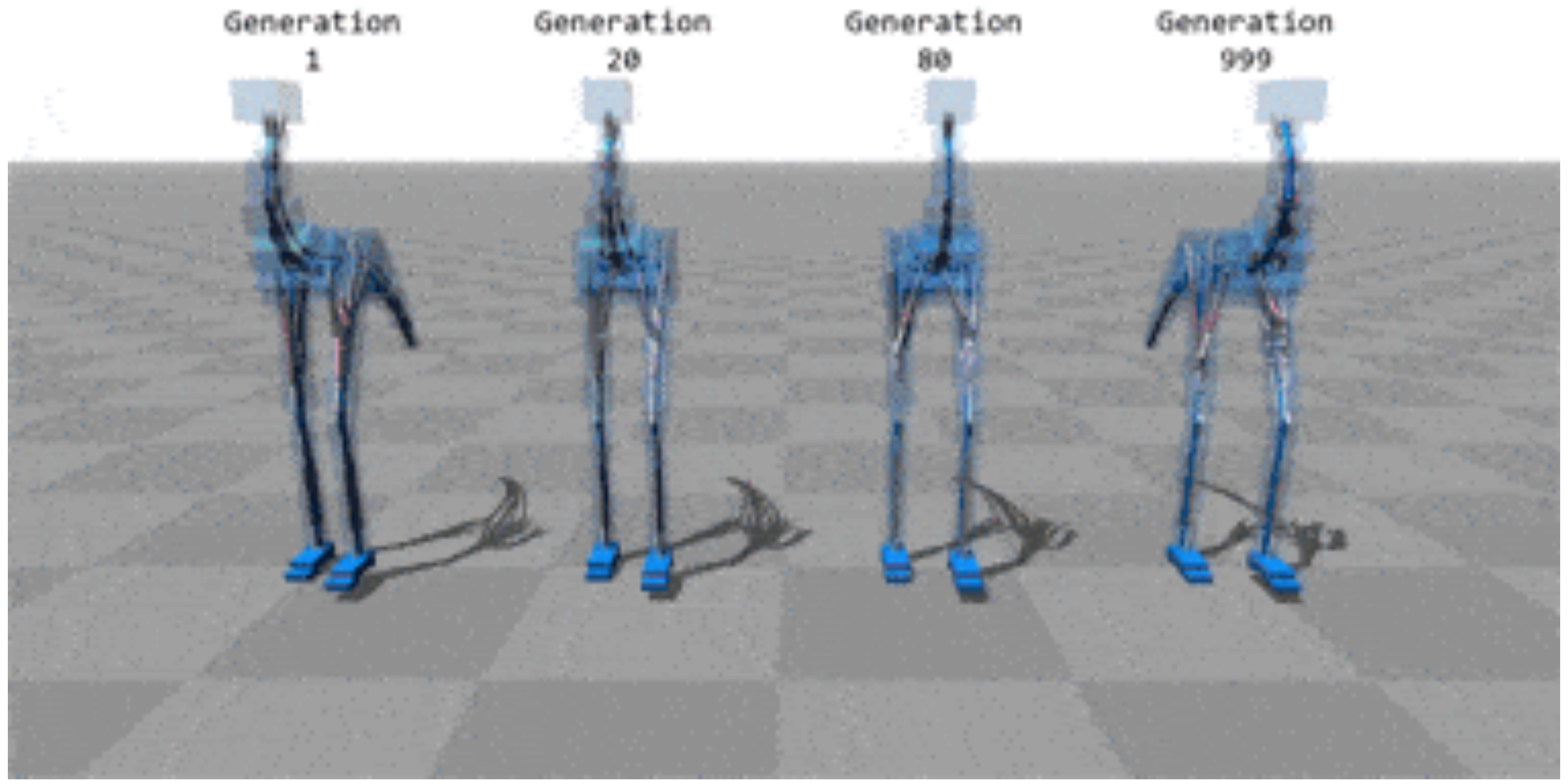
Learn from the environment through trial and error

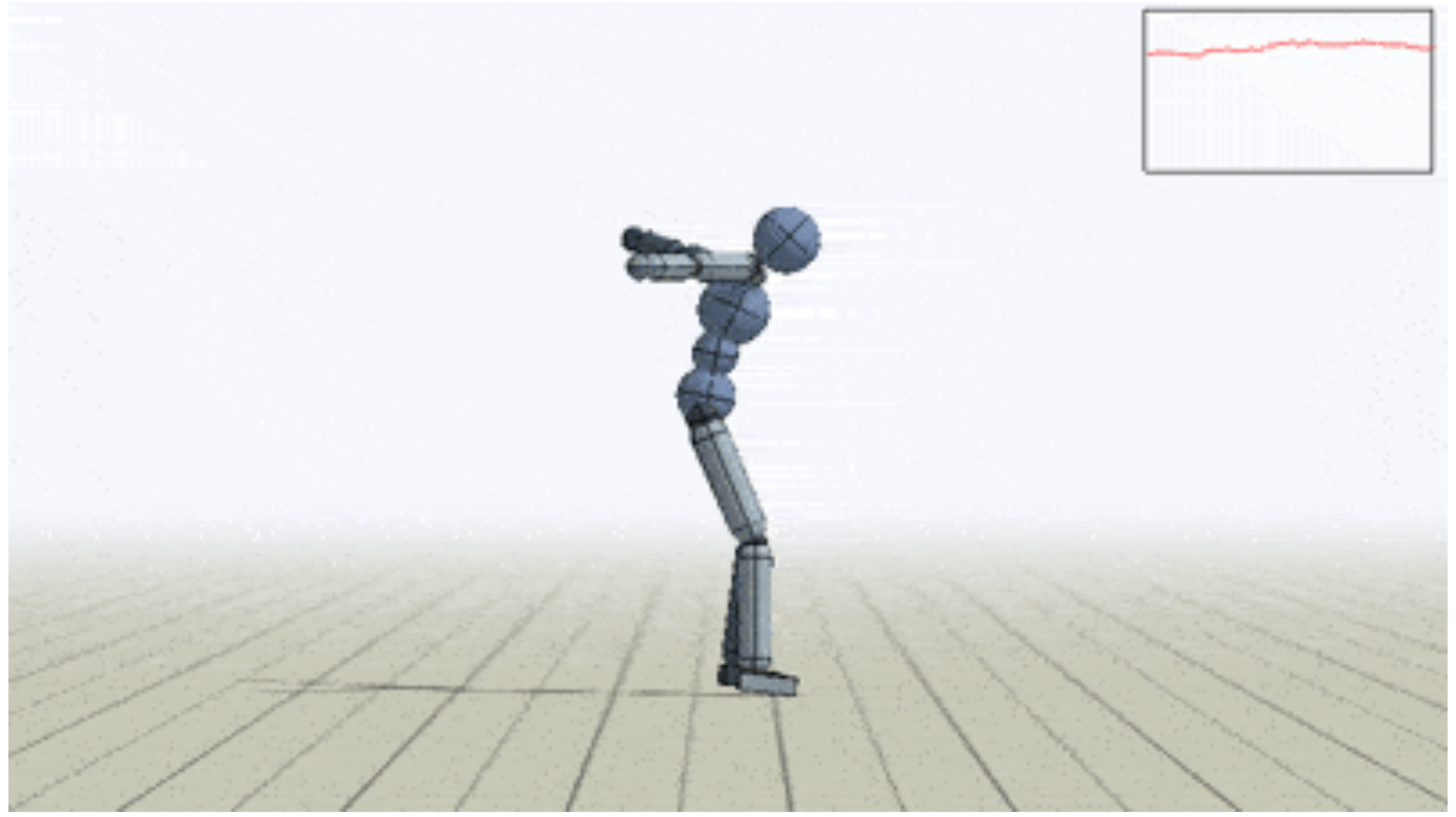
Receive rewards as feedback

Reinforcement Learning









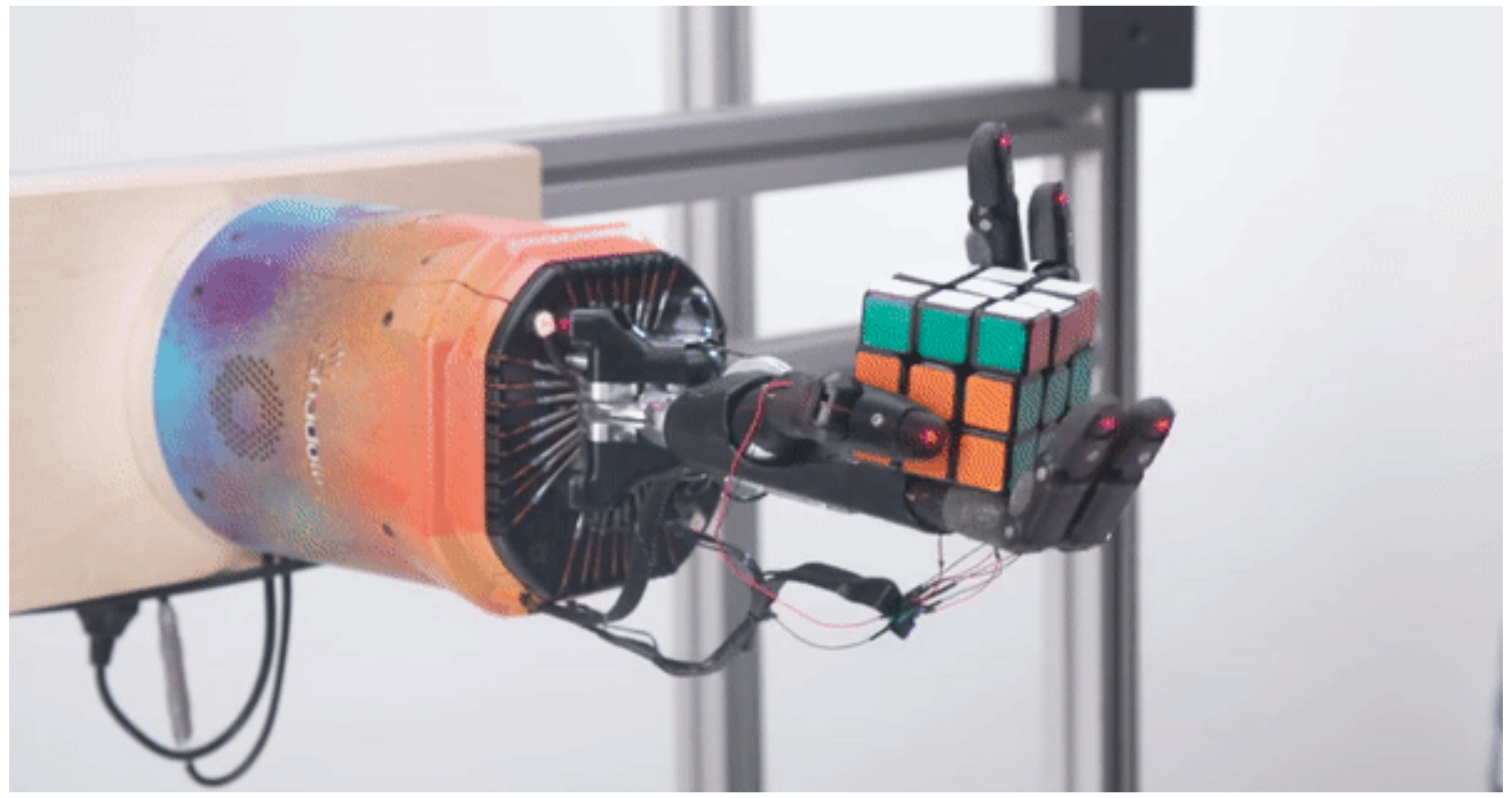
Akkaya, Ilge, et al. "Solving rubik's cube with a robot hand." arXiv preprint arXiv:1910.07113 (2019). (https://openai.com/research/solving-rubiks-cube)

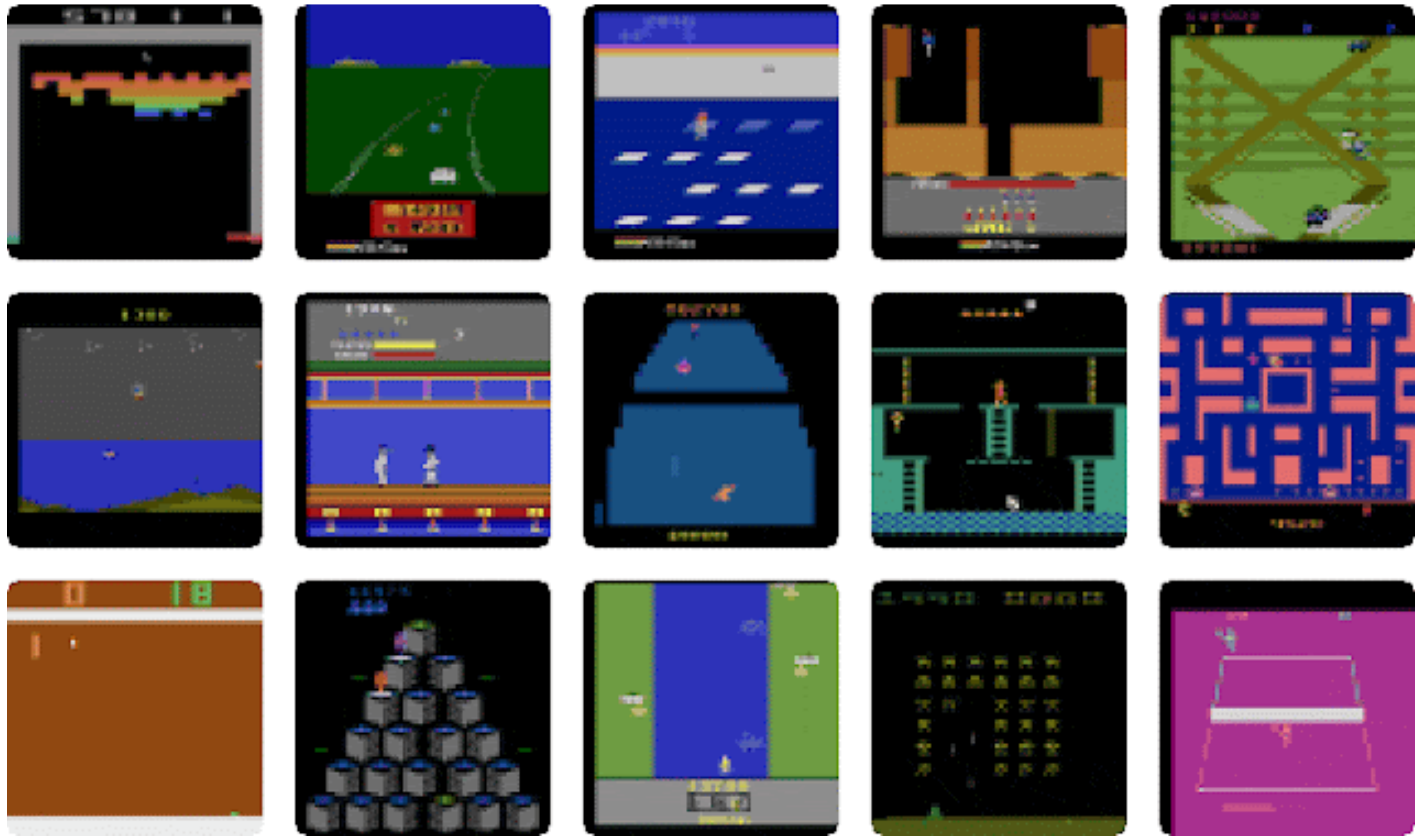
Mnih, V., Kavukcuoglu, K., Silver, D. et al. Human-level control through deep reinforcement learning. Nature **518**, 529–533 (2015).

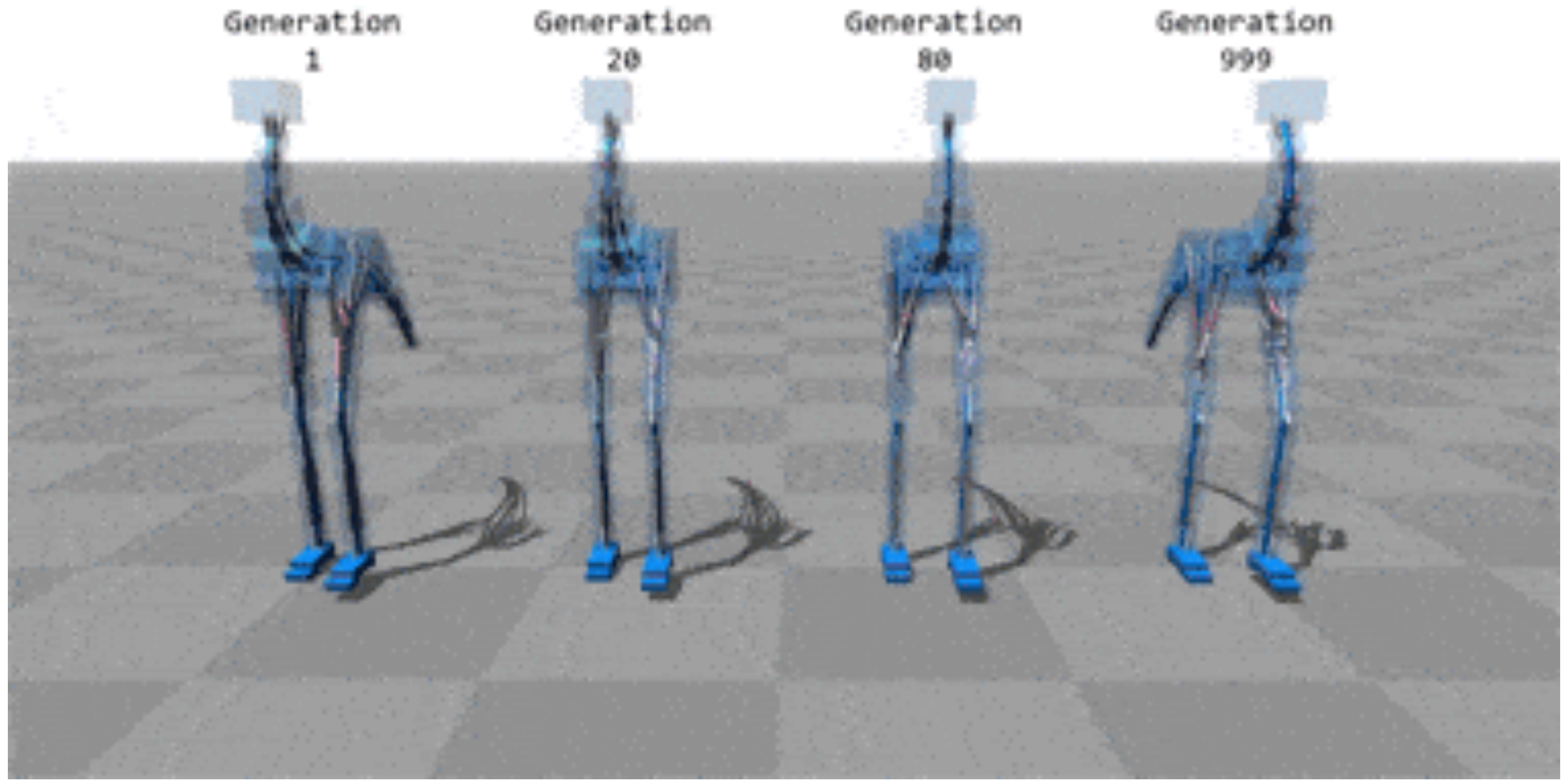
Geijtenbeek, Thomas, Michiel Van De Panne, and A. Frank Van Der Stappen. "Flexible muscle-based locomotion for bipedal creatures." ACM Transactions on Graphics (TOG) 32.6 (2013): 1-11.

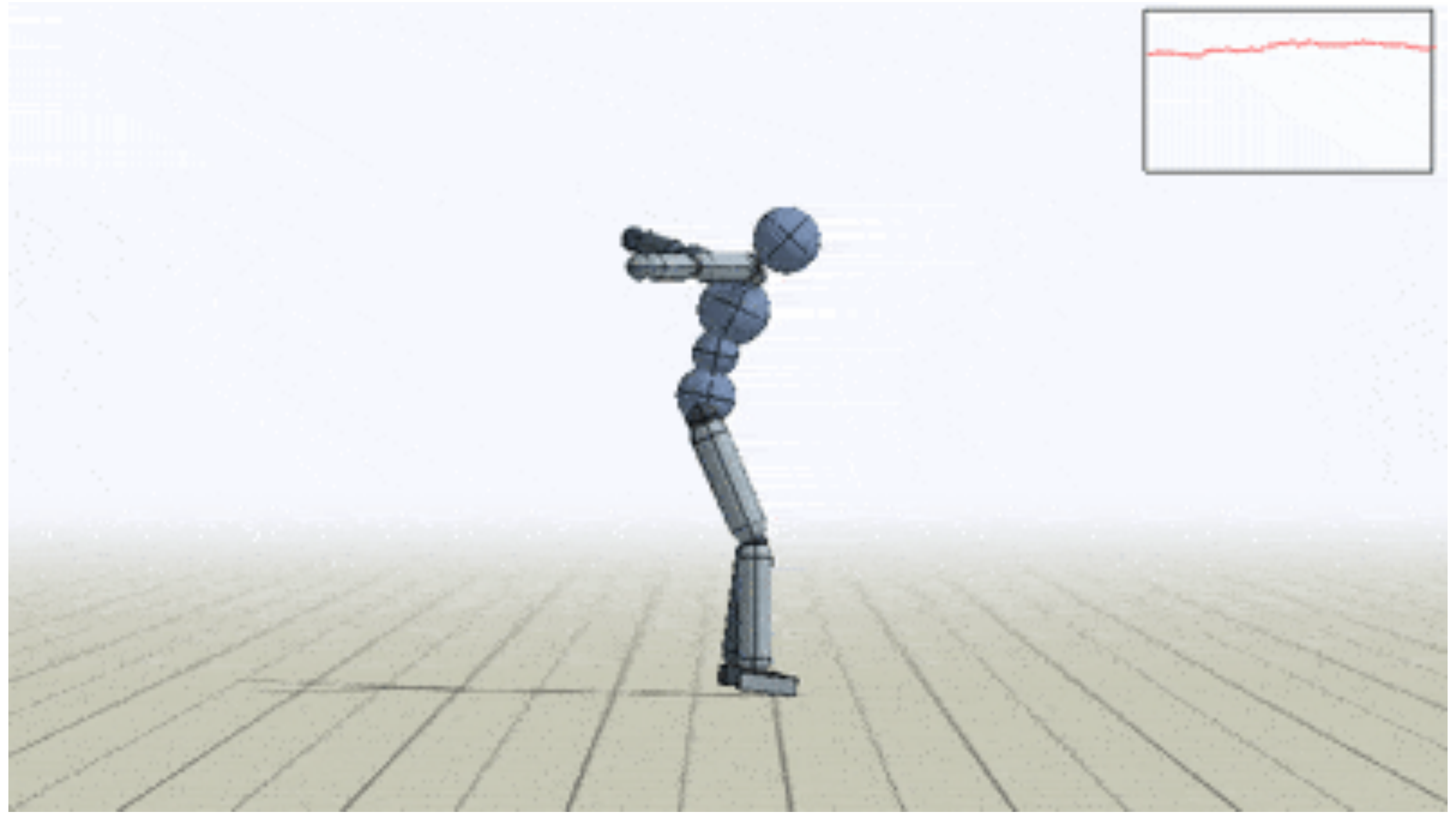
Peng, Xue Bin, et al. "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills." ACM Transactions On Graphics (TOG) 37.4 (2018): 1-14.

Reinforcement Learning









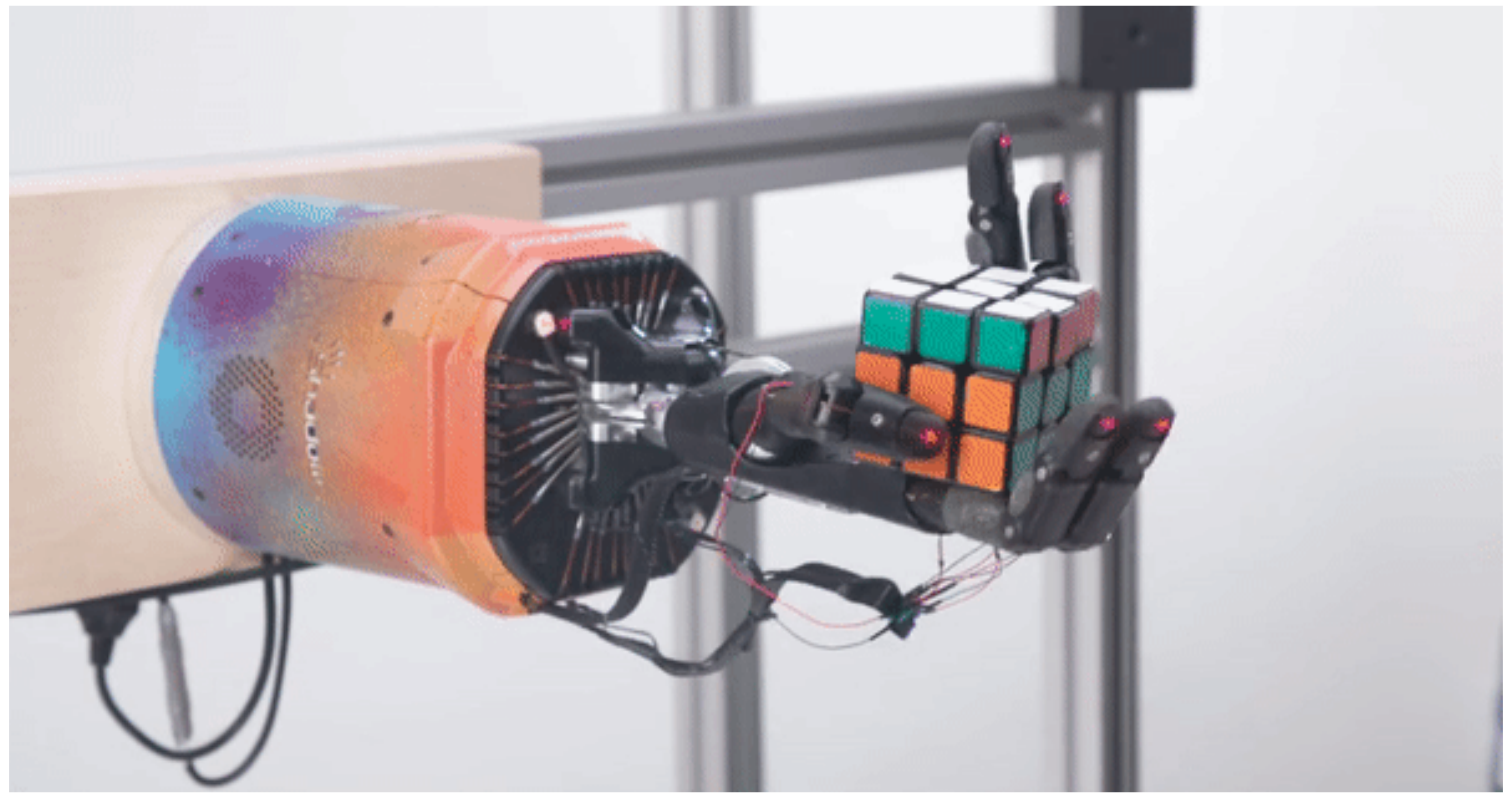
Akkaya, Ilge, et al. "Solving rubik's cube with a robot hand." arXiv preprint arXiv:1910.07113 (2019). (https://openai.com/research/solving-rubiks-cube)

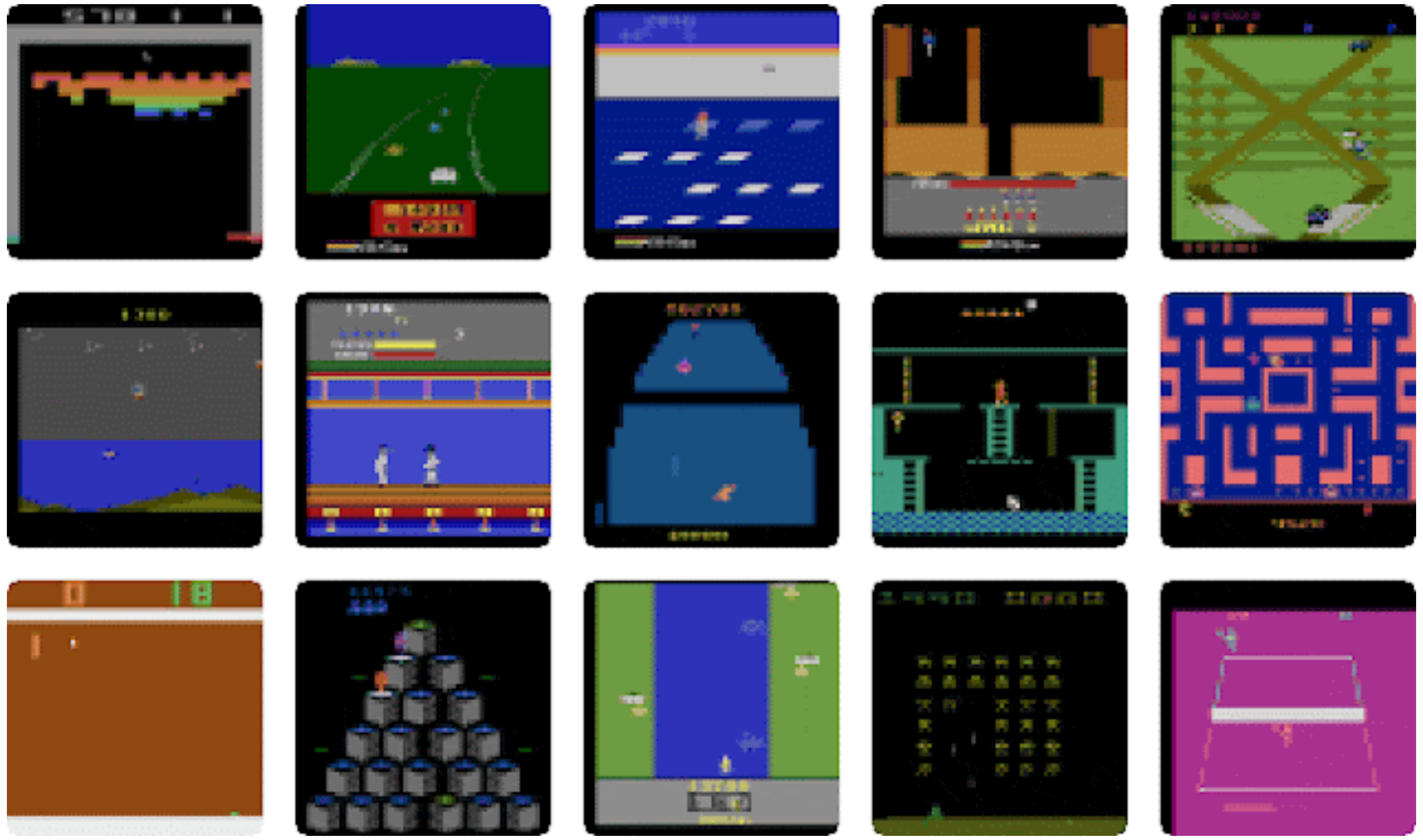
Mnih, V., Kavukcuoglu, K., Silver, D. et al. Human-level control through deep reinforcement learning. Nature **518**, 529–533 (2015).

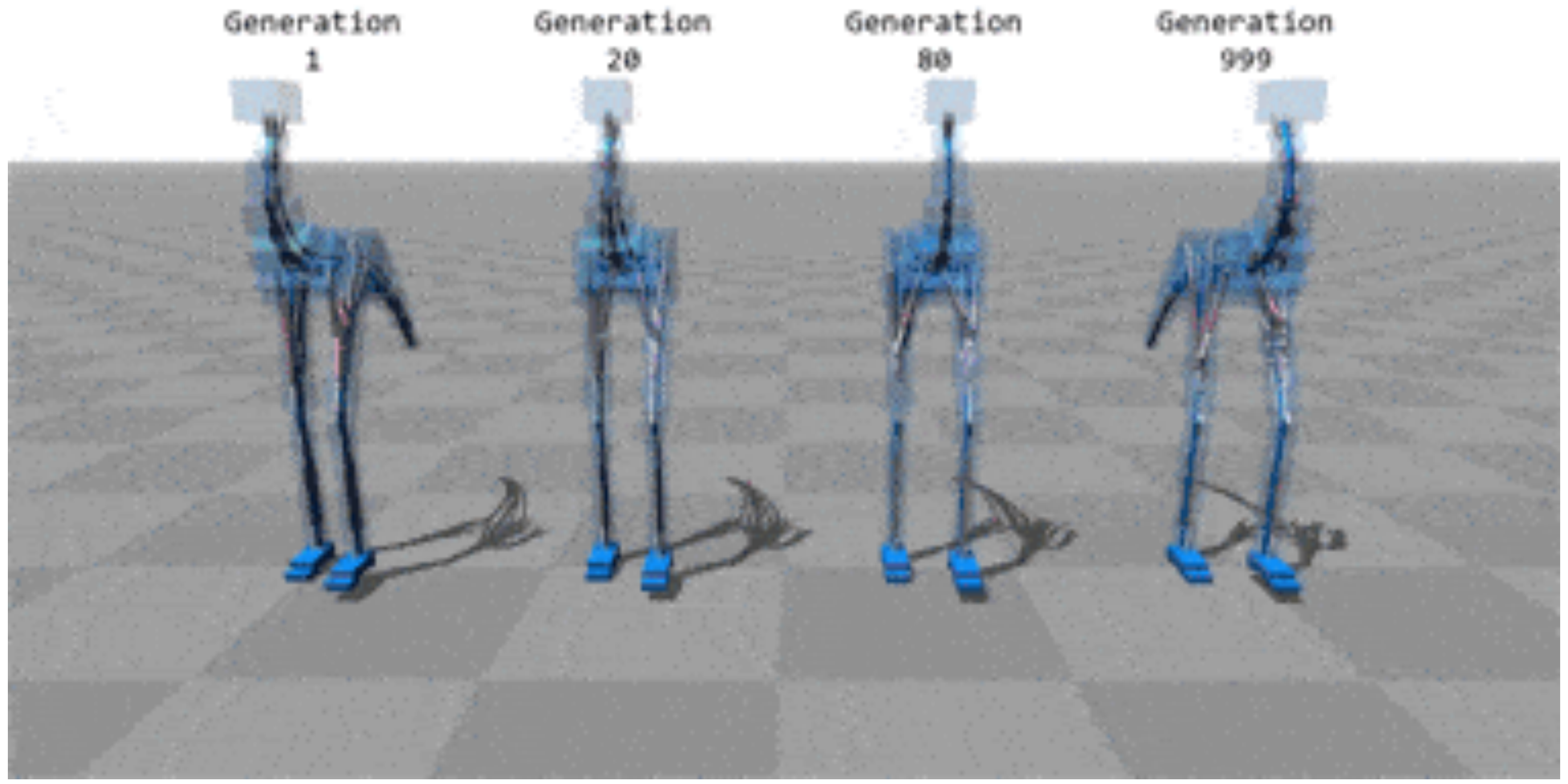
Geijtenbeek, Thomas, Michiel Van De Panne, and A. Frank Van Der Stappen. "Flexible muscle-based locomotion for bipedal creatures." ACM Transactions on Graphics (TOG) 32.6 (2013): 1-11.

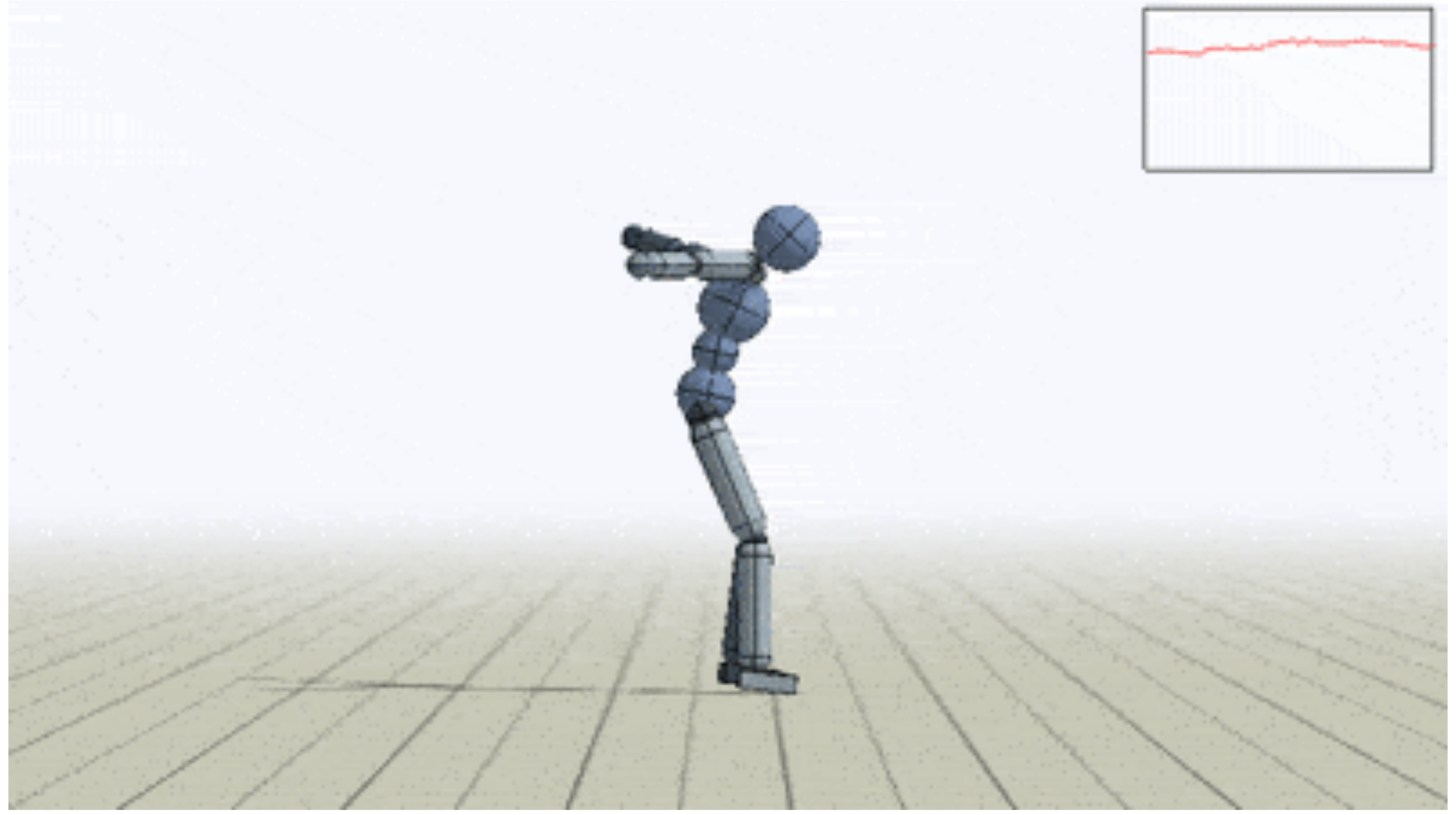
Peng, Xue Bin, et al. "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills." ACM Transactions On Graphics (TOG) 37.4 (2018): 1-14.

Reinforcement Learning









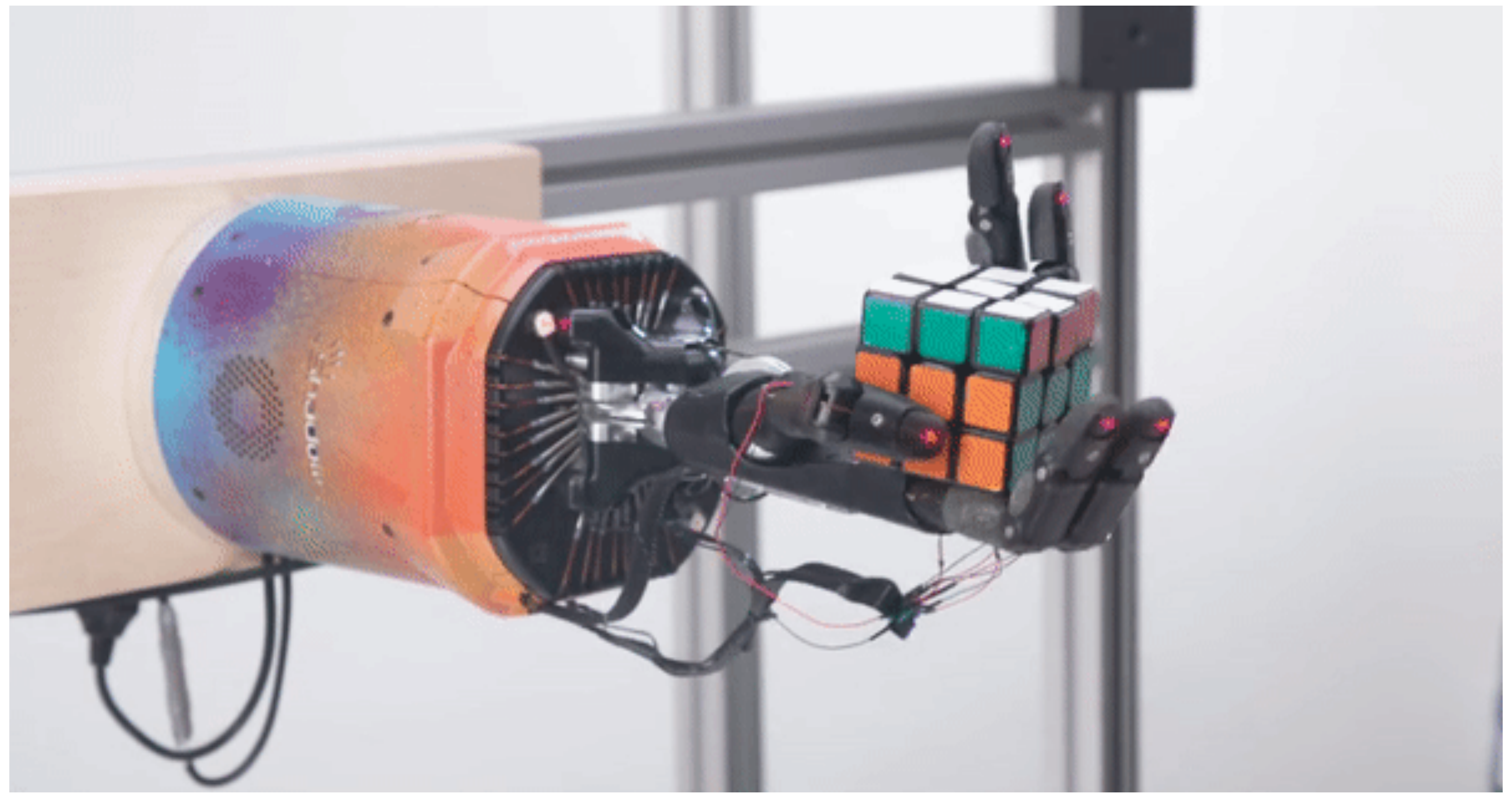
Akkaya, Ilge, et al. "Solving rubik's cube with a robot hand." arXiv preprint arXiv:1910.07113 (2019). (https://openai.com/research/solving-rubiks-cube)

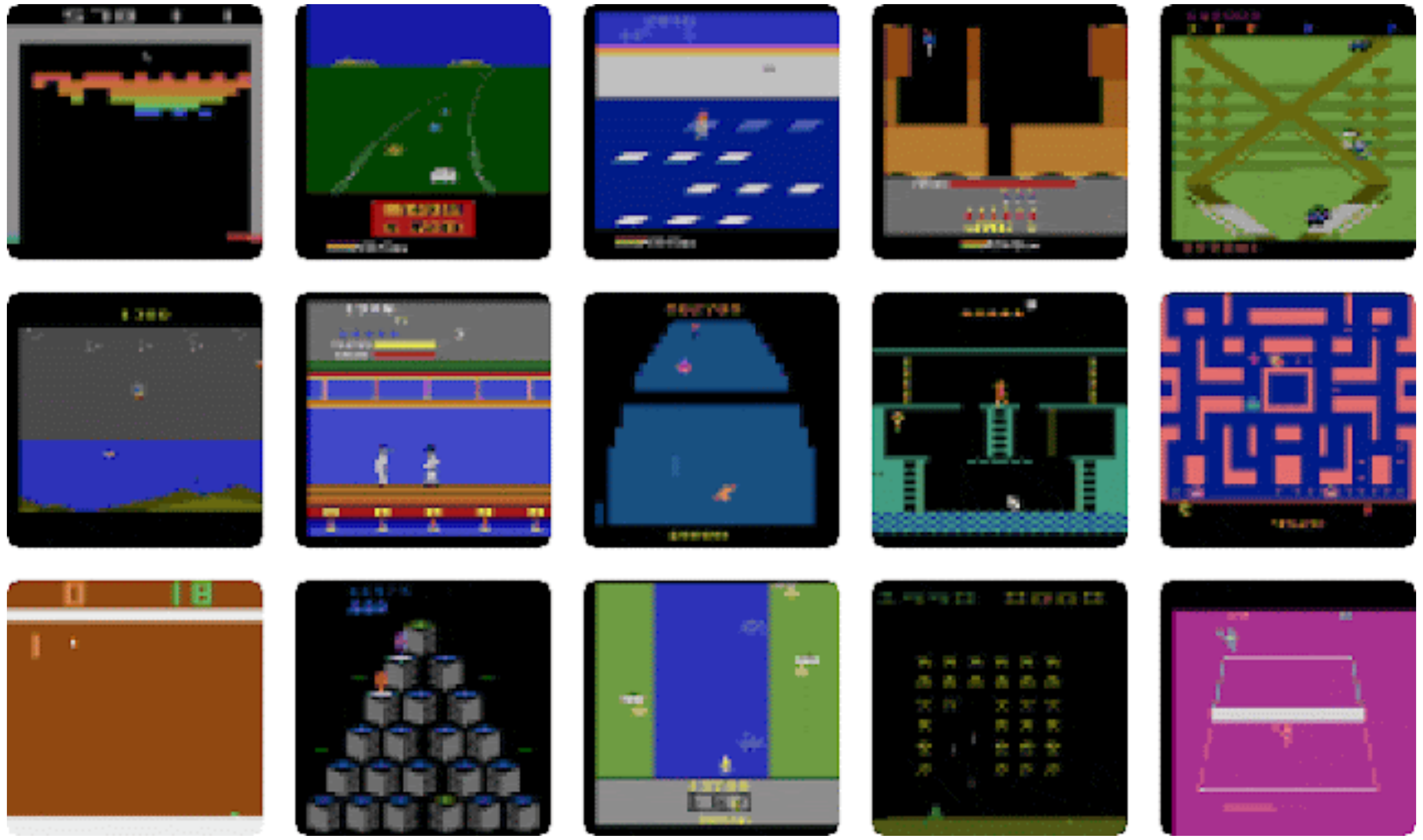
Mnih, V., Kavukcuoglu, K., Silver, D. et al. Human-level control through deep reinforcement learning. Nature **518**, 529–533 (2015).

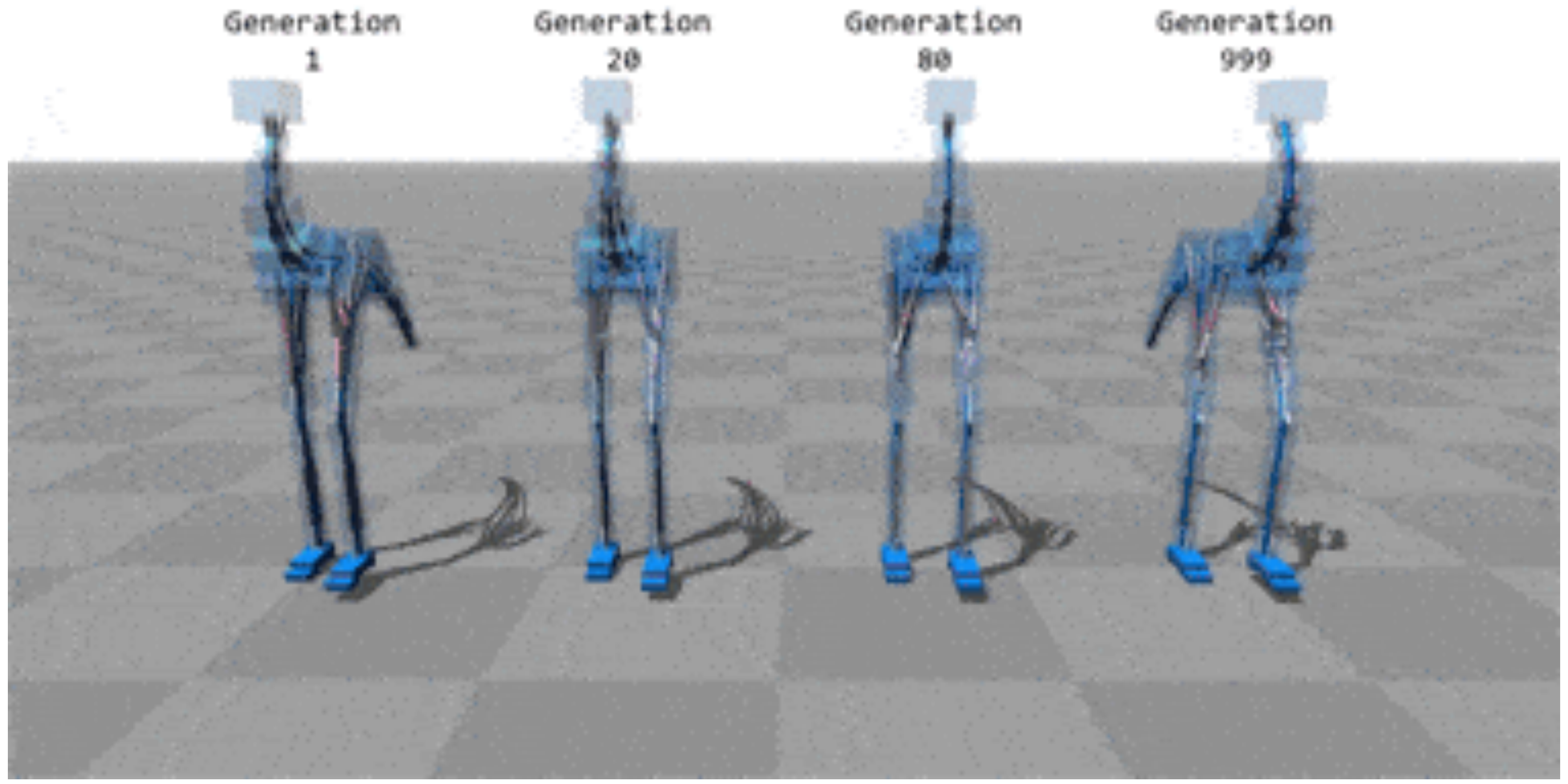
Geijtenbeek, Thomas, Michiel Van De Panne, and A. Frank Van Der Stappen. "Flexible muscle-based locomotion for bipedal creatures." ACM Transactions on Graphics (TOG) 32.6 (2013): 1-11.

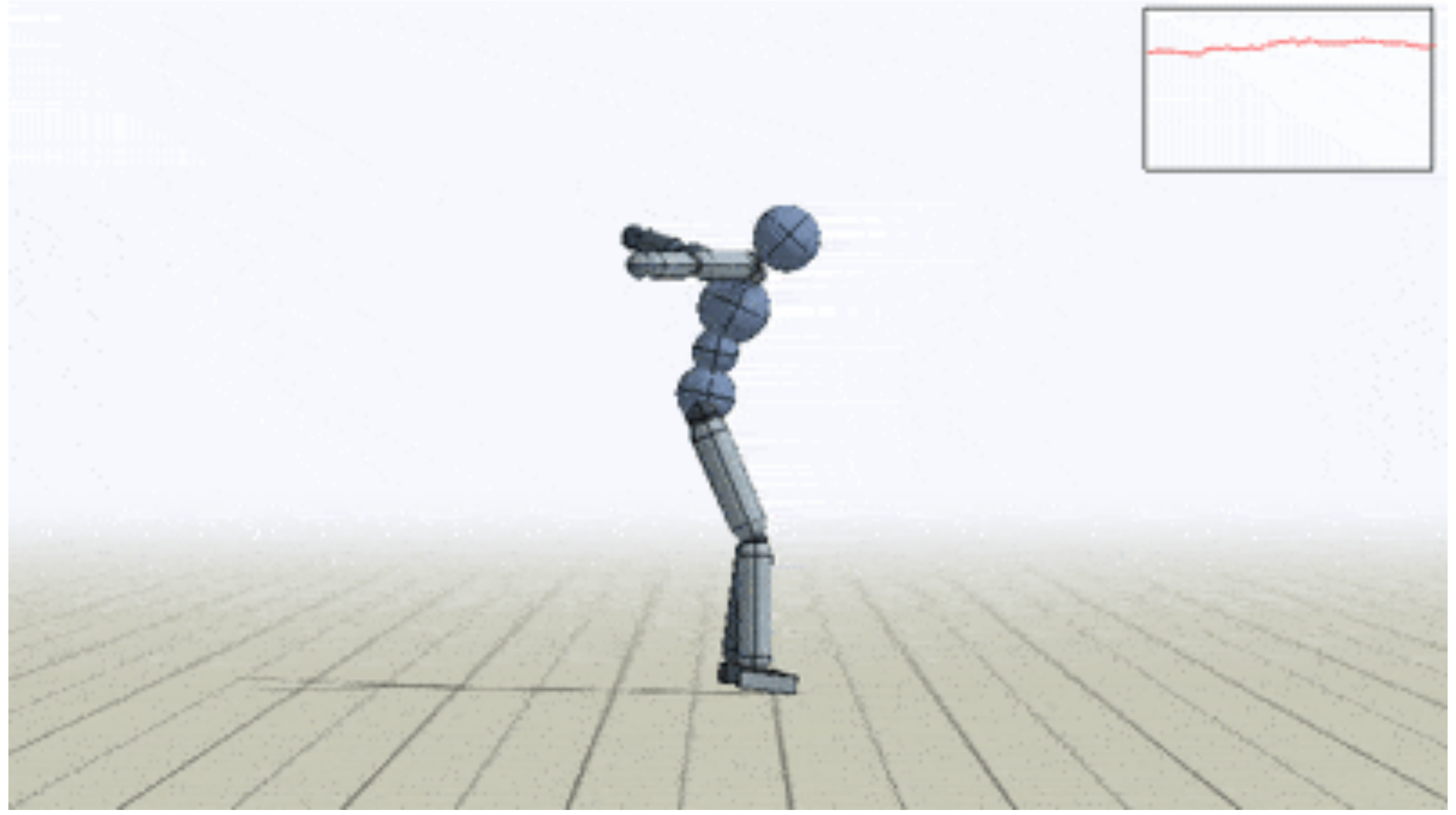
Peng, Xue Bin, et al. "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills." ACM Transactions On Graphics (TOG) 37.4 (2018): 1-14.

Reinforcement Learning









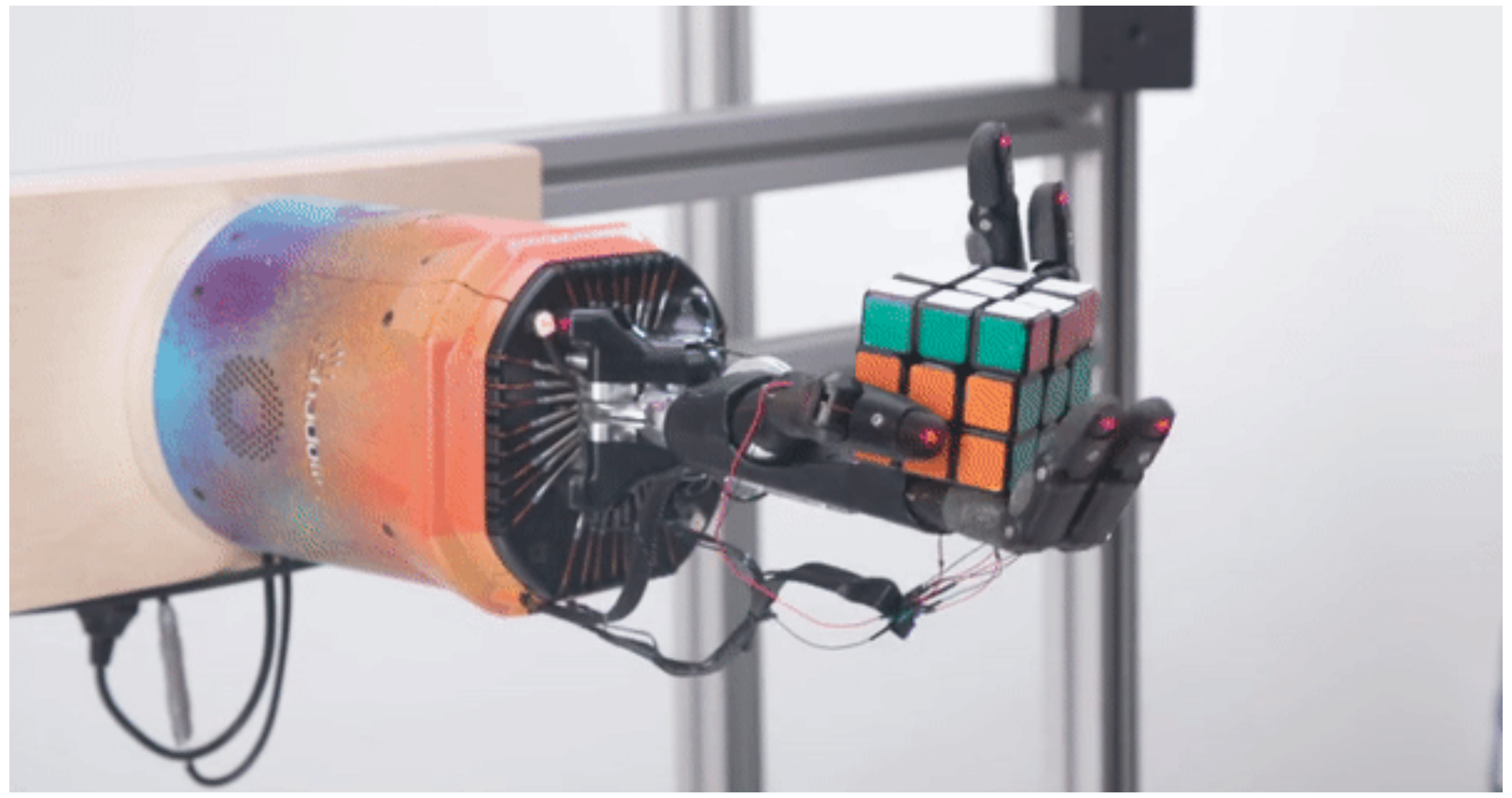
Akkaya, Ilge, et al. "Solving rubik's cube with a robot hand." arXiv preprint arXiv:1910.07113 (2019). (https://openai.com/research/solving-rubiks-cube)

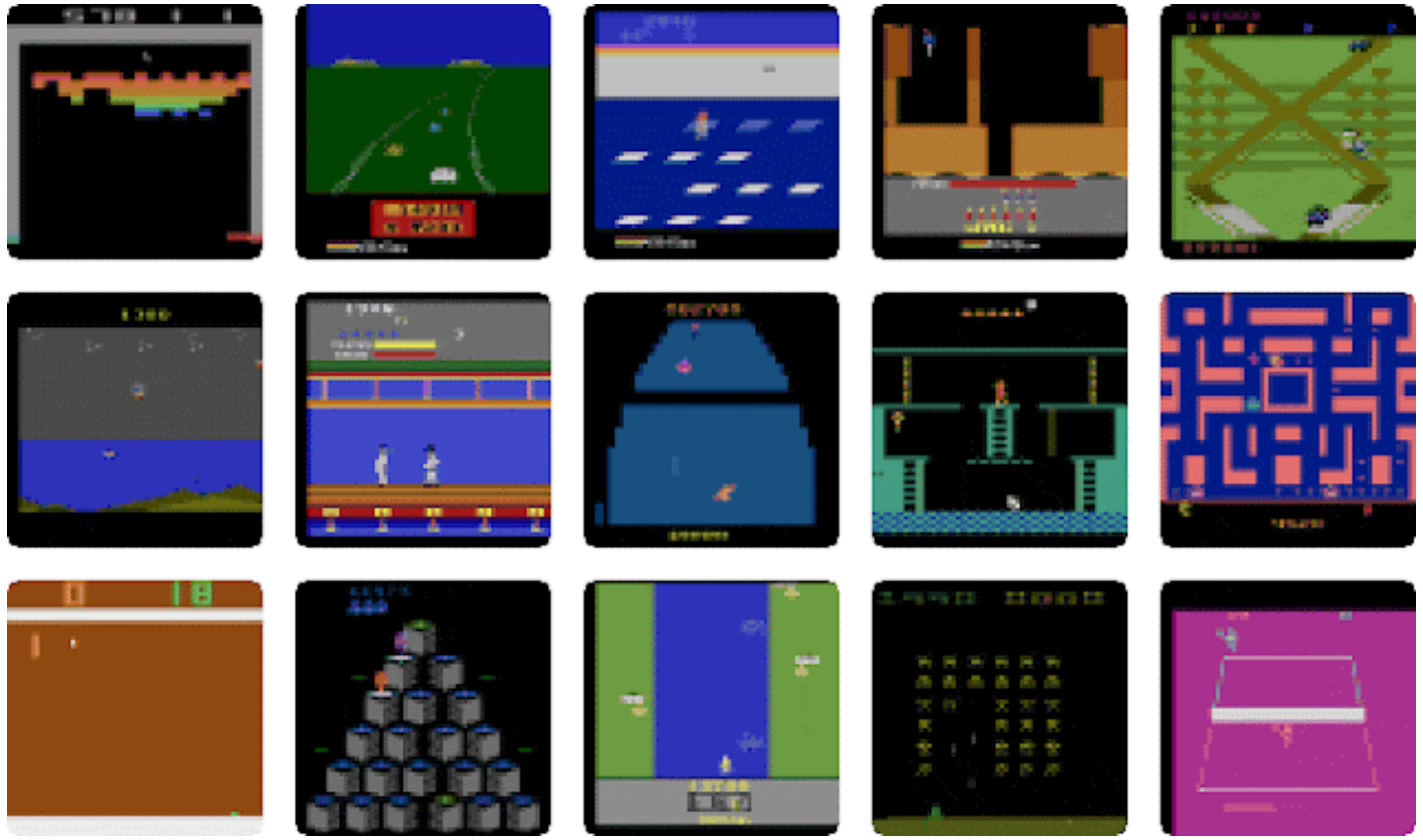
Mnih, V., Kavukcuoglu, K., Silver, D. et al. Human-level control through deep reinforcement learning. Nature **518**, 529–533 (2015).

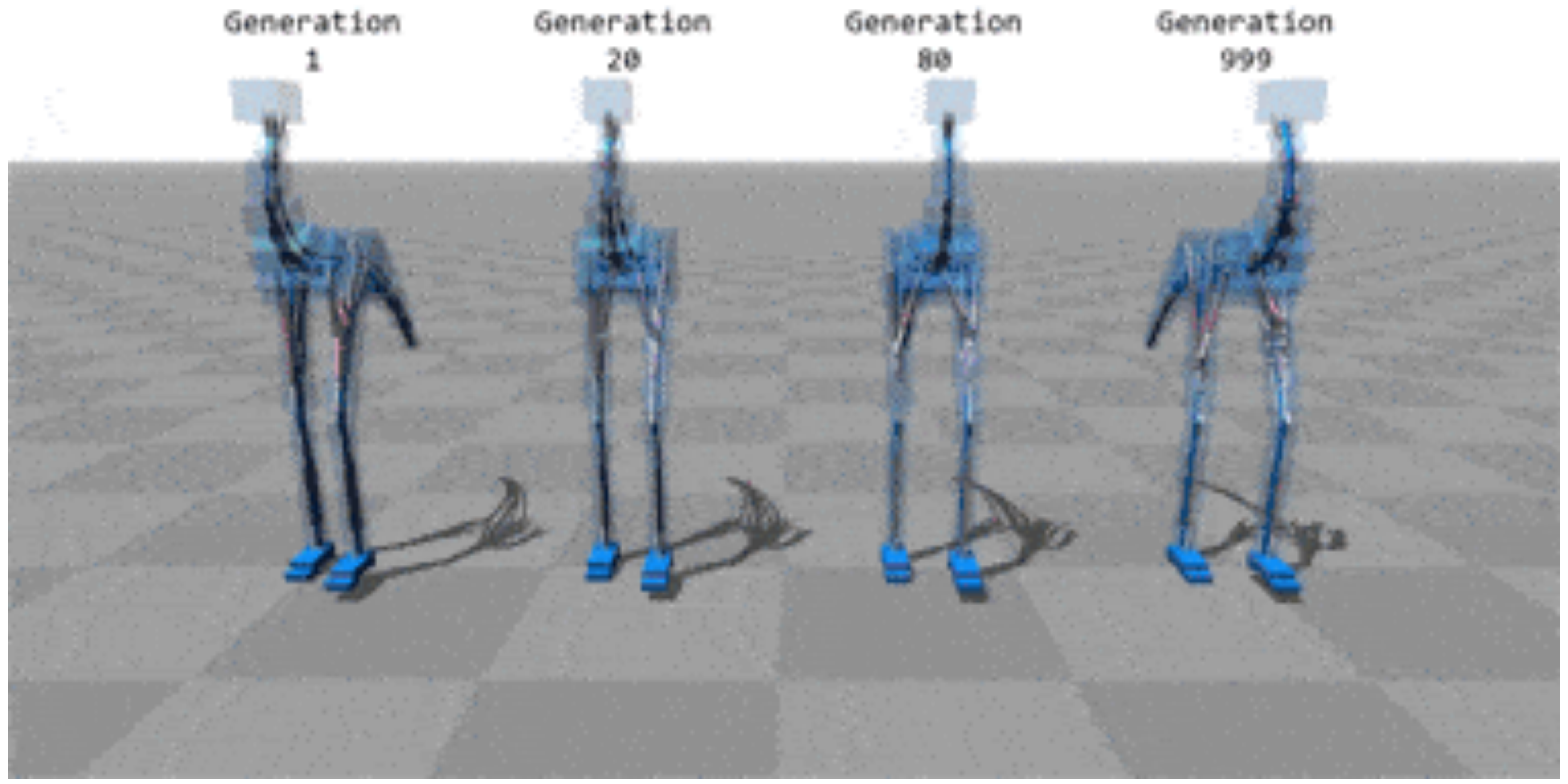
Geijtenbeek, Thomas, Michiel Van De Panne, and A. Frank Van Der Stappen. "Flexible muscle-based locomotion for bipedal creatures." ACM Transactions on Graphics (TOG) 32.6 (2013): 1-11.

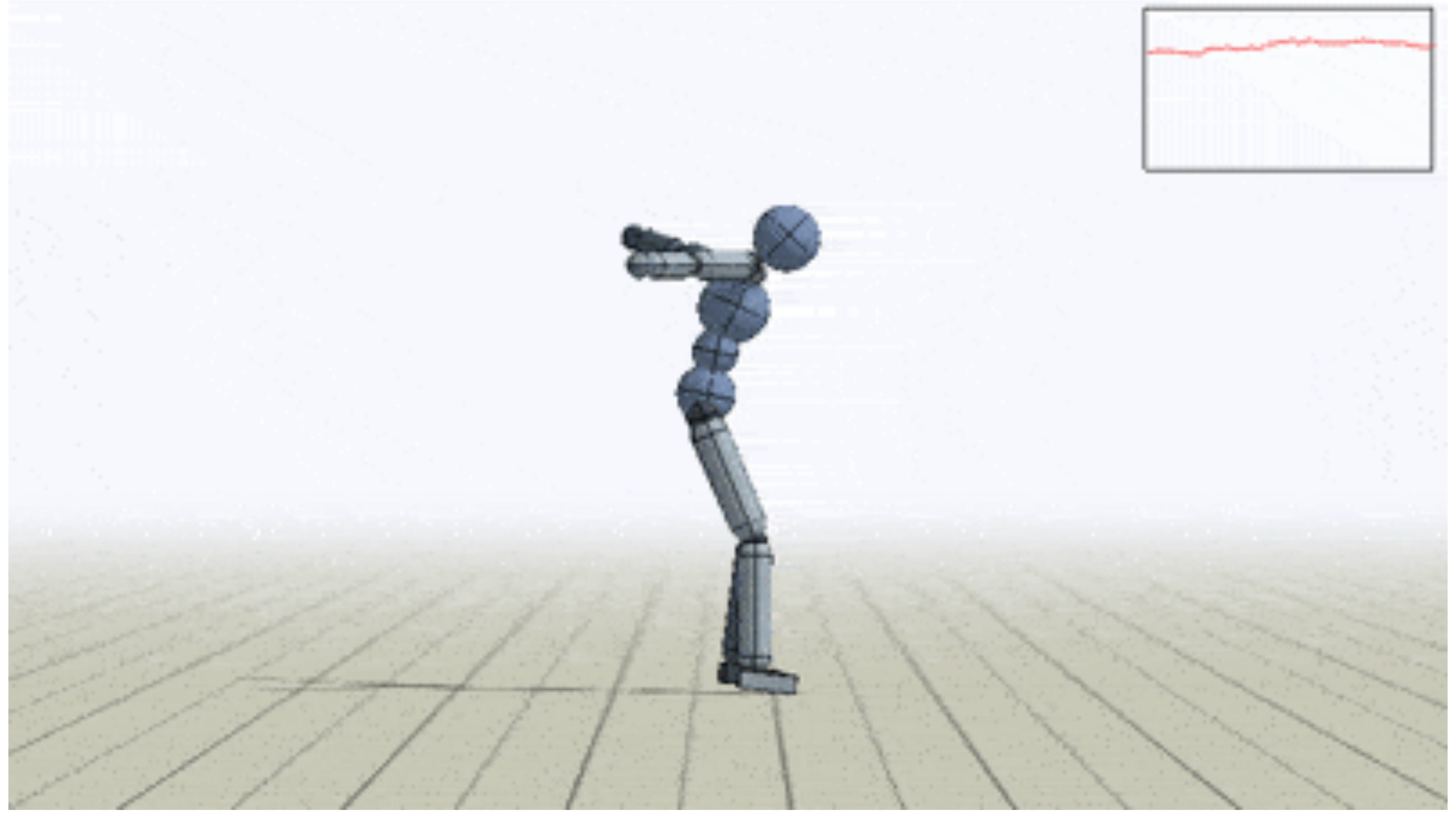
Peng, Xue Bin, et al. "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills." ACM Transactions On Graphics (TOG) 37.4 (2018): 1-14.

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

https://www.youtube.com/watch?v=PxoG0A2QoFs

Reinforcement Learning

https://www.youtube.com/watch?v=PxoG0A2QoFs

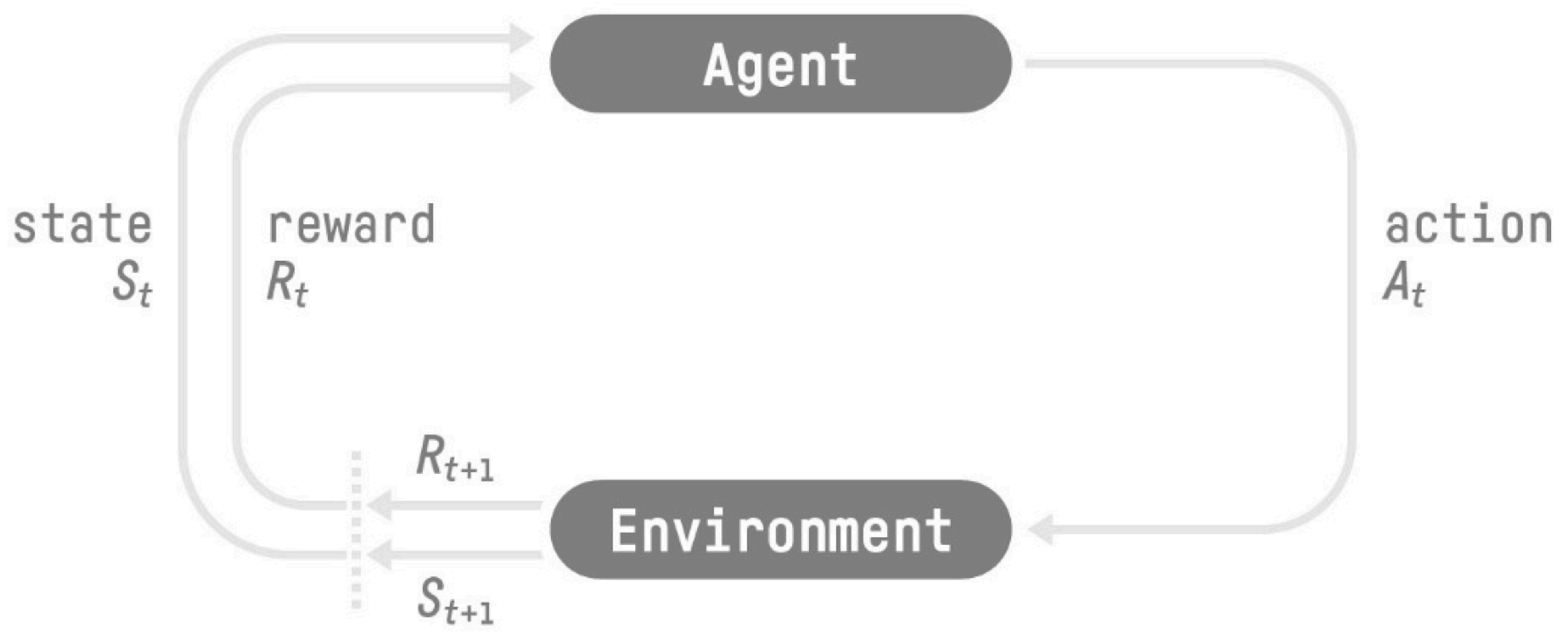
What you will learn

Basic concepts of RL

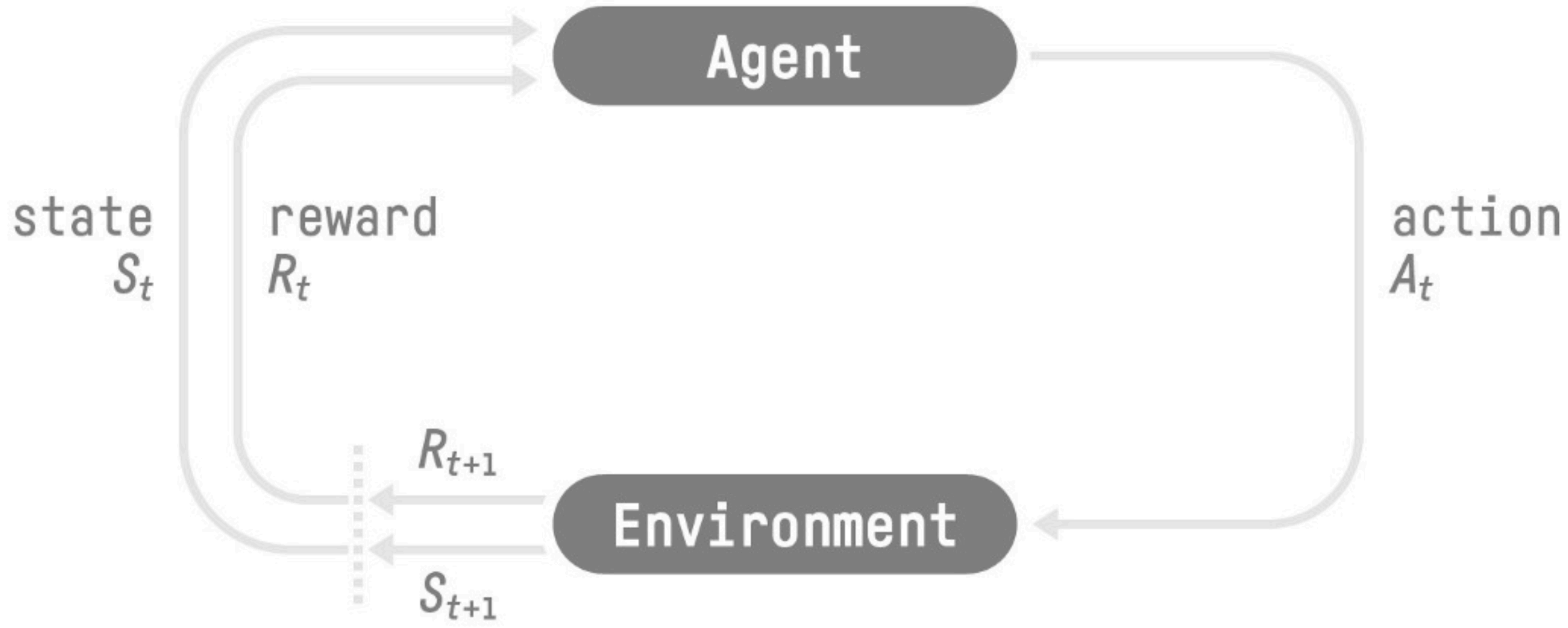
Q-Learning

Deep Q-Learning

Markov Decision Process (MDP) (finite)

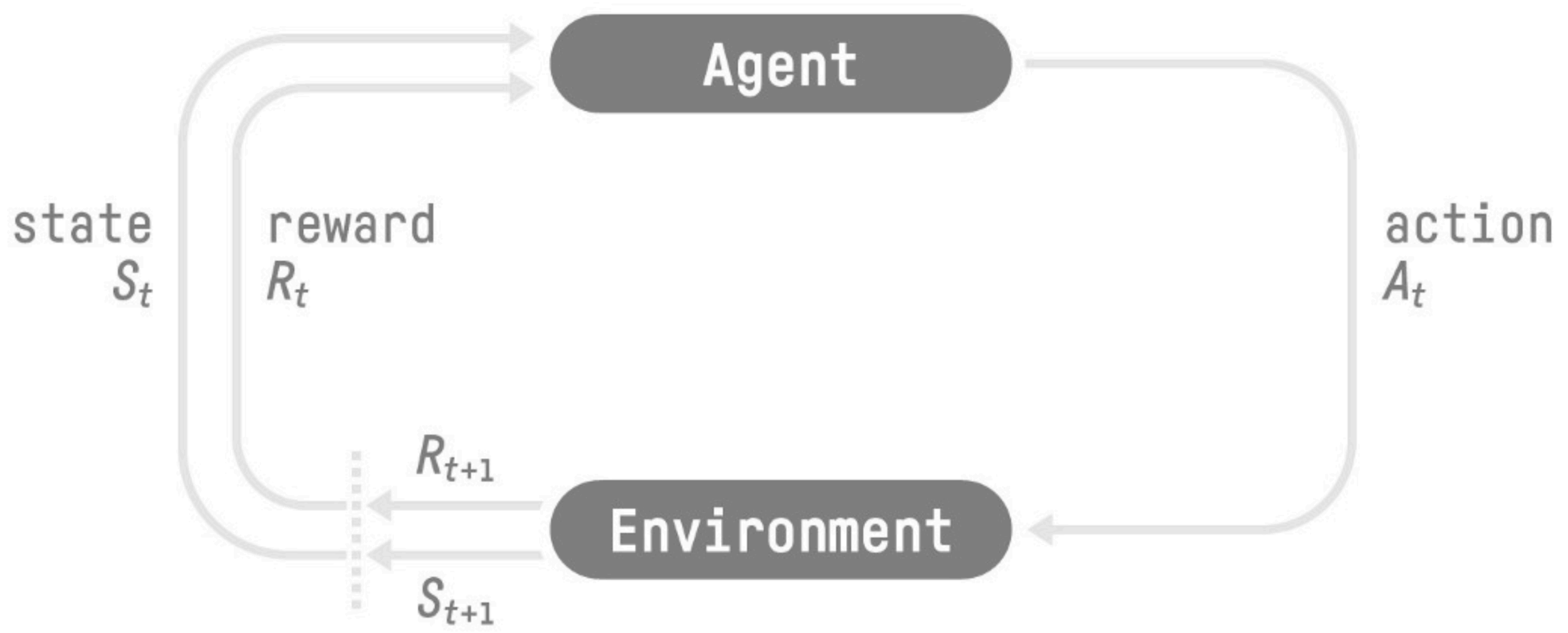


Markov Decision Process (MDP) (finite)



For t = 0, 1, 2, 3 …

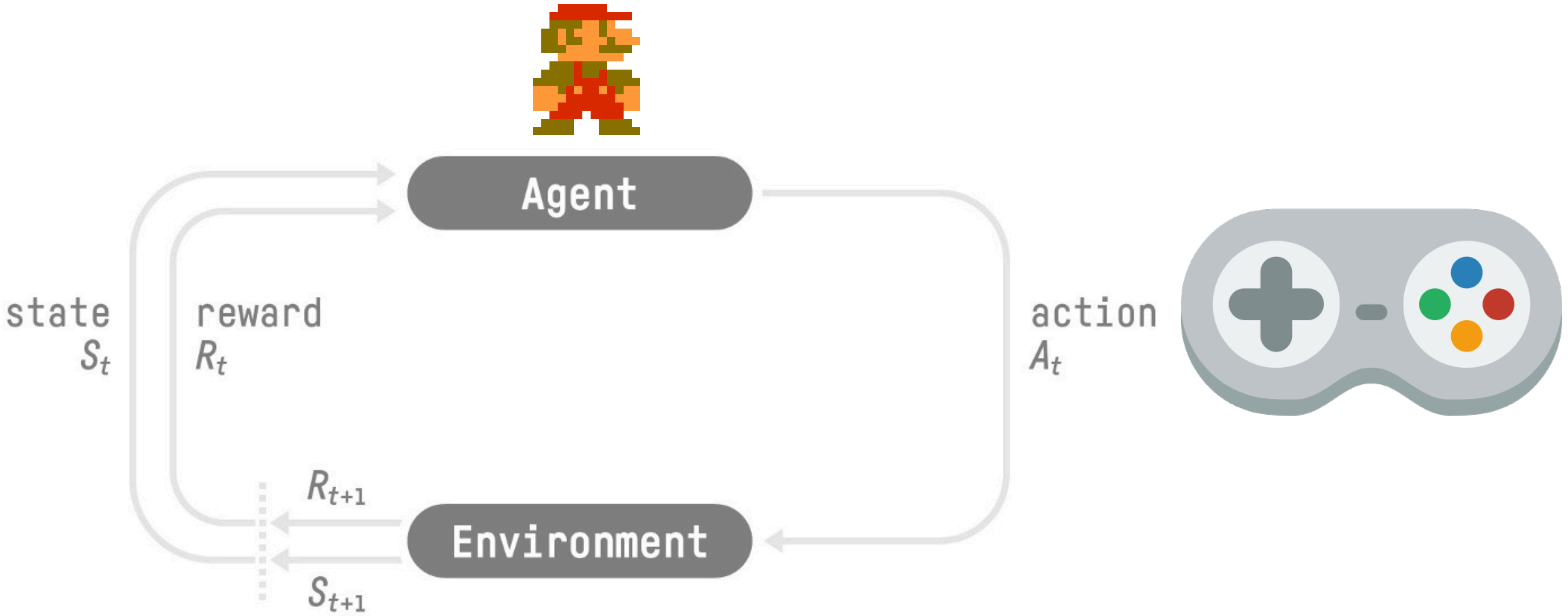
Markov Decision Process (MDP) (finite)



For t = 0, 1, 2, 3 …

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

Markov Decision Process (MDP)



For t = 0, 1, 2, 3 …

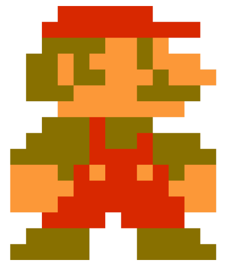
Paradigm

Paradigm

S_0

Paradigm

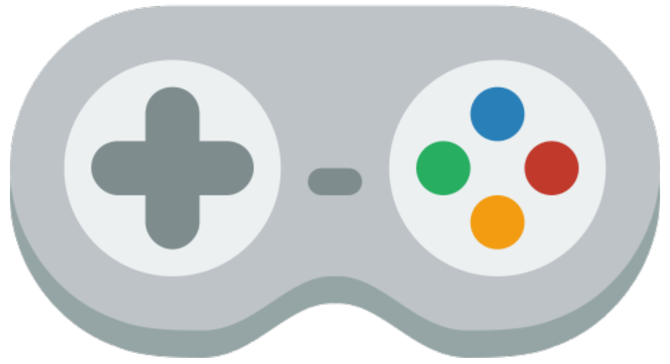
S_0

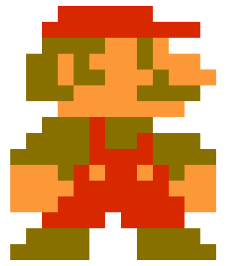


Policy π

Paradigm

S_0



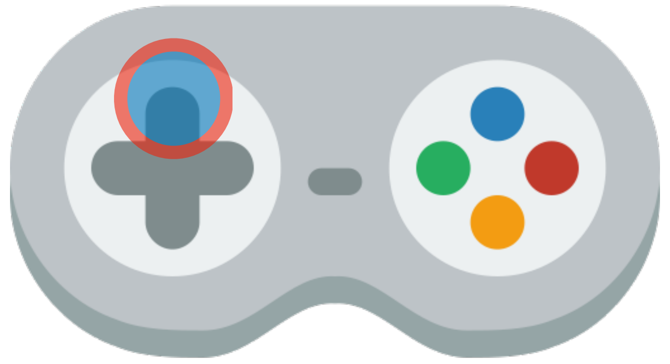


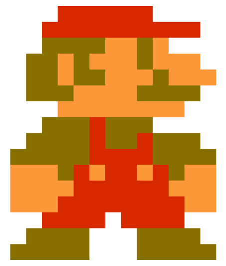
Policy π

Paradigm

S_0

A_0



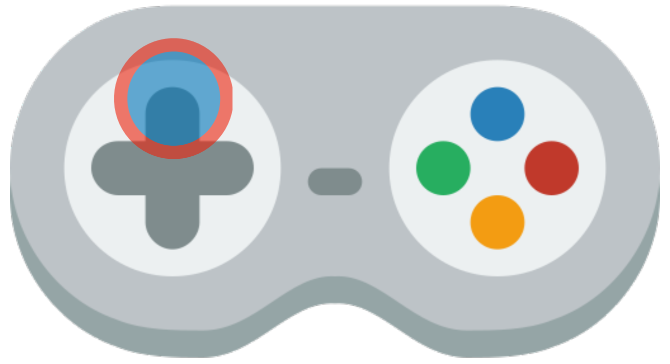


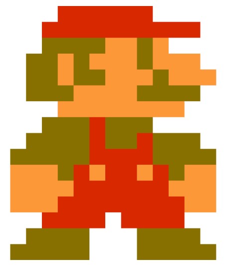
Policy π

Paradigm

S_0

S_1 $r_1 = 0$

A_0 

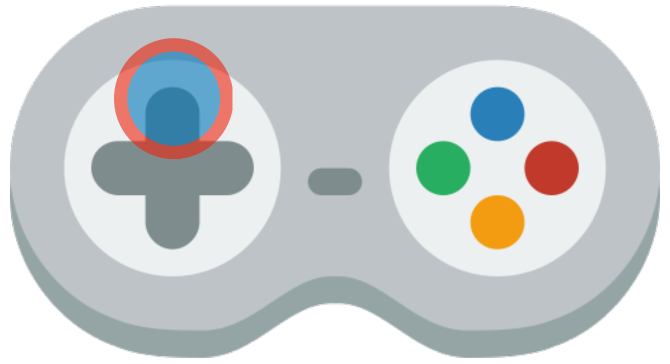
 Policy π

Paradigm

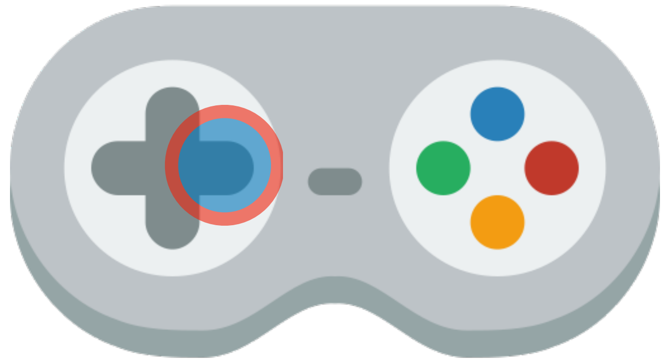
S_0

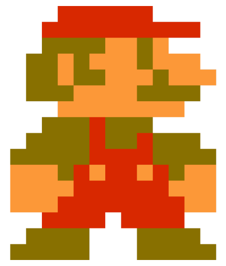
S_1 $r_1 = 0$

A_0



A_1





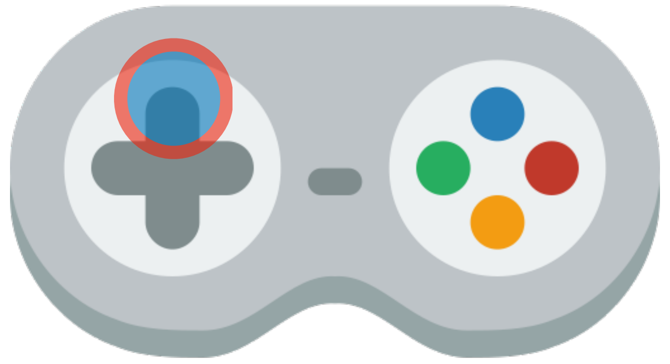
Policy π

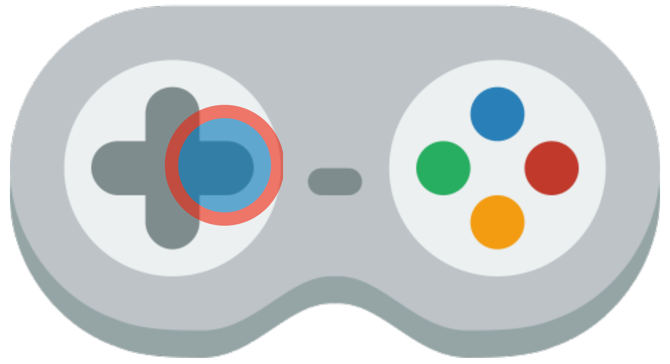
Paradigm

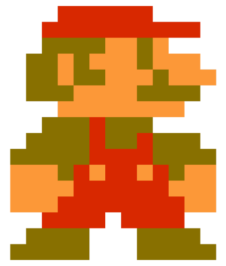
S_0

S_1 $r_1 = 0$

S_2 $r_2 = 100$

A_0 

A_1 

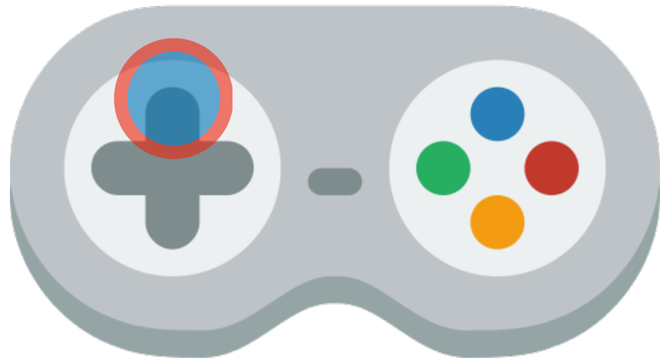
 Policy π

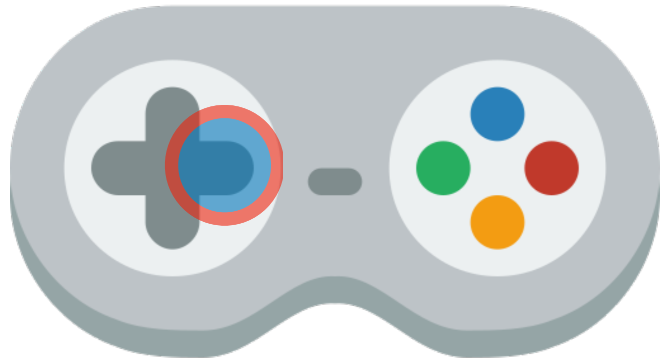
Paradigm

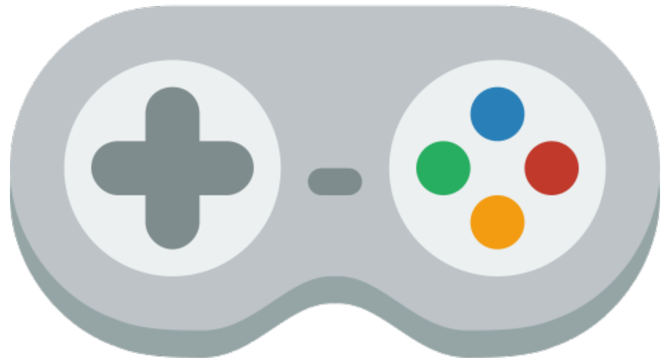
S_0

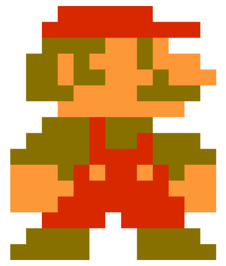
S_1 $r_1 = 0$

S_2 $r_2 = 100$

A_0 

A_1 

A_2 



Policy π

Agent Inputs

Agent Inputs

How do agents get information from the environment?

Agent Inputs

How do agents get information from the environment?

States

Agent Inputs

How do agents get information from the environment?

States

Observations

Agent Inputs

How do agents get information from the environment?

States

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

Observations

Agent Inputs

How do agents get information from the environment?

States

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

Observations

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 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

Agent Outputs

How do agents interact with the environment?

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

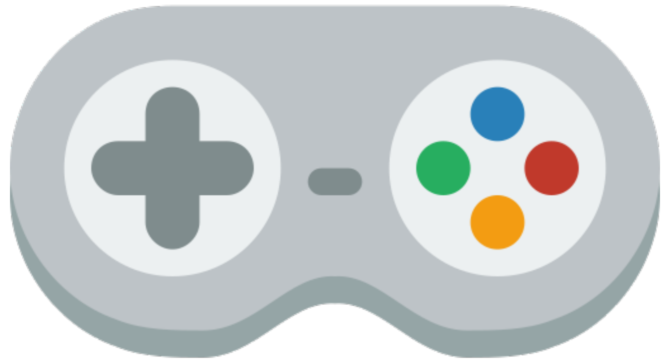
Agent Outputs

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

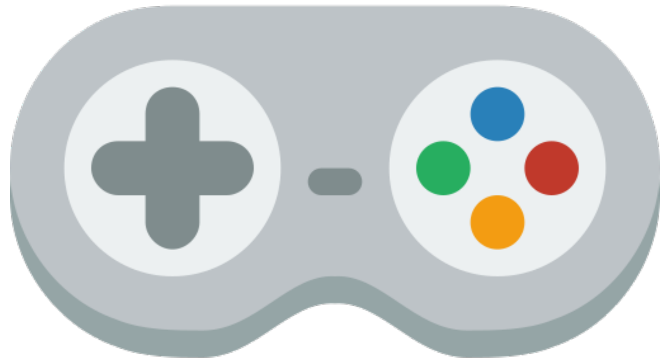
Agent Outputs

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*

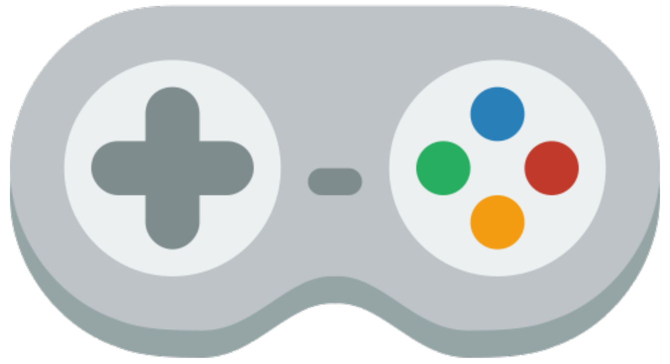
Agent Outputs

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*

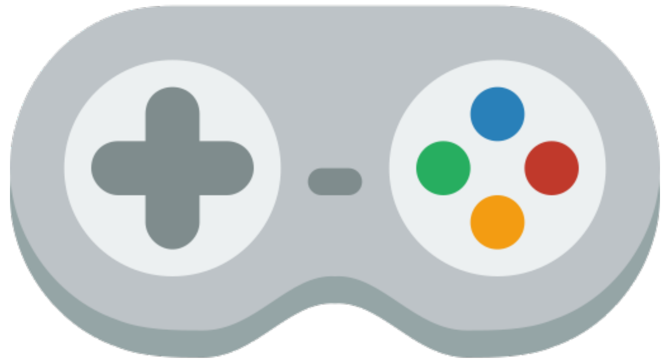
Agent Outputs

How do agents interact with the environment?

Action space Set of all possible actions in an environment.

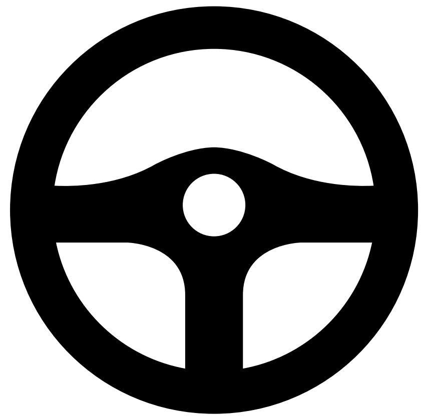
Discrete

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

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

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*)

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

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*)

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

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 $0 < \gamma < 1$

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 $0 < \gamma < 1$

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

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

$$0 < \gamma < 1$$

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

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

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

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

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

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

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

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

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

Action-value function for policy π

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

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

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

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

Action-value function for policy π

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

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

v_π and q_π can be estimated from experience

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. $\varepsilon -$ 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 $Q(\mathit{terminal}, \cdot) = 0$

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'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  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 $Q(\mathit{terminal}, \cdot) = 0$

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'
        $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
        $S \leftarrow S'$
    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 $Q(\mathit{terminal}, \cdot) = 0$

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'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  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 $Q(\mathit{terminal}, \cdot) = 0$

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'
        $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
        $S \leftarrow S'$
    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 $Q(terminal, \cdot) = 0$

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

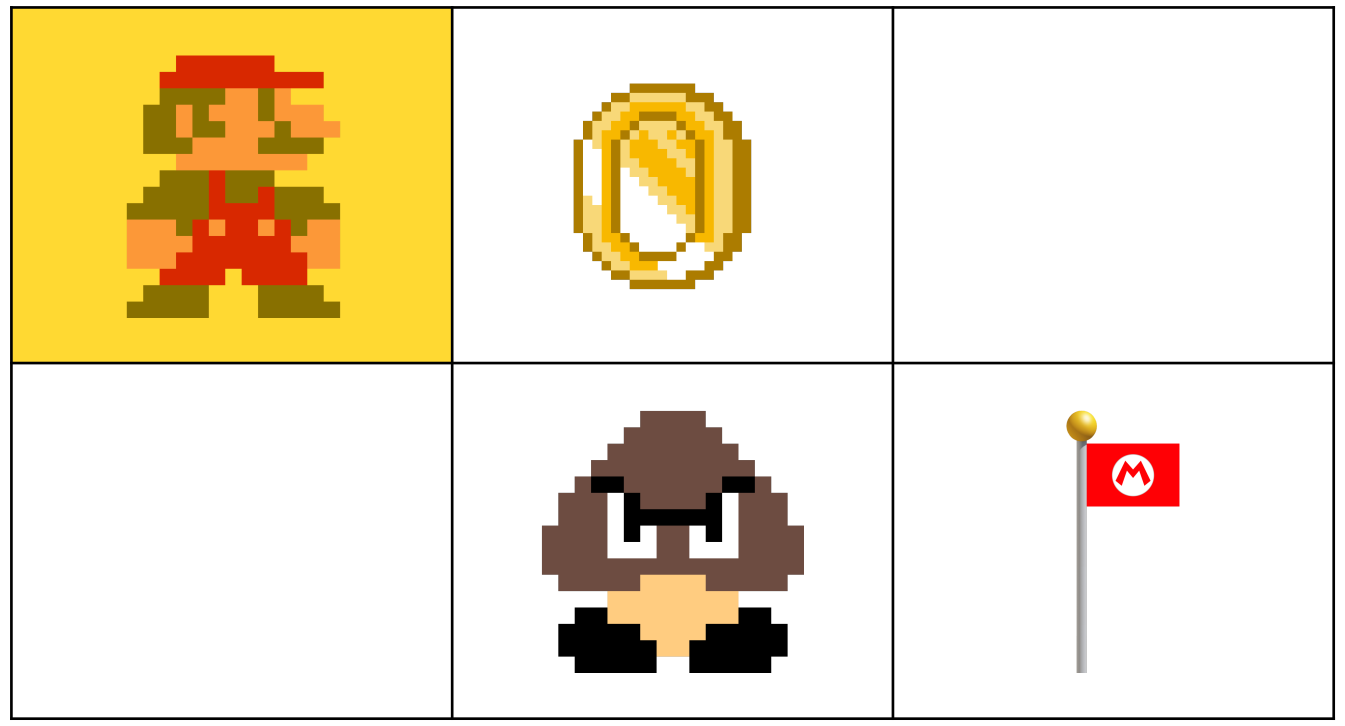
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma\ max_{a'}\ Q(S_{t+1}, a') - Q(S_t, A_t))$$

Bellman equation for state-action values

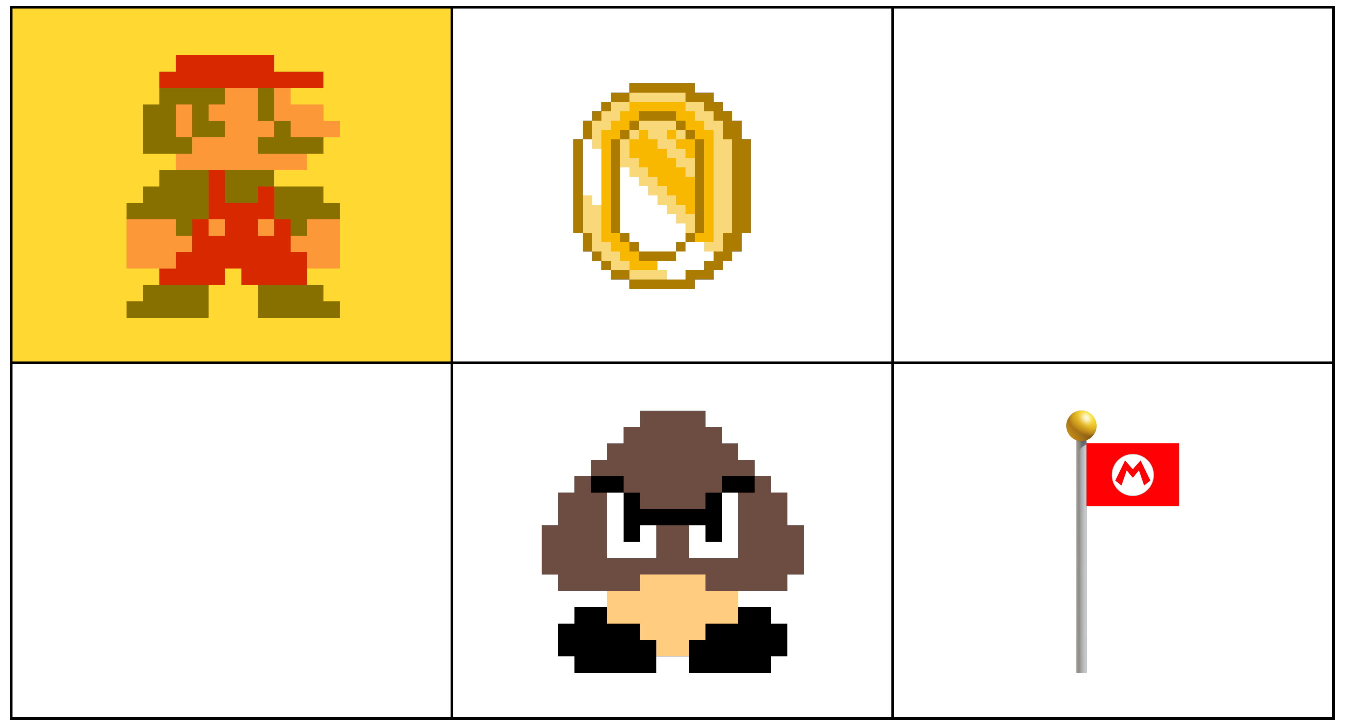
What is $Q(S, A)$?

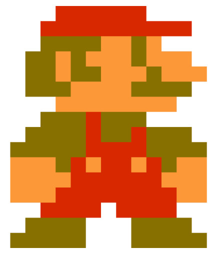
Encode Q-values in a table, each cell corresponds to a state-action pair value.

Q-Learning example



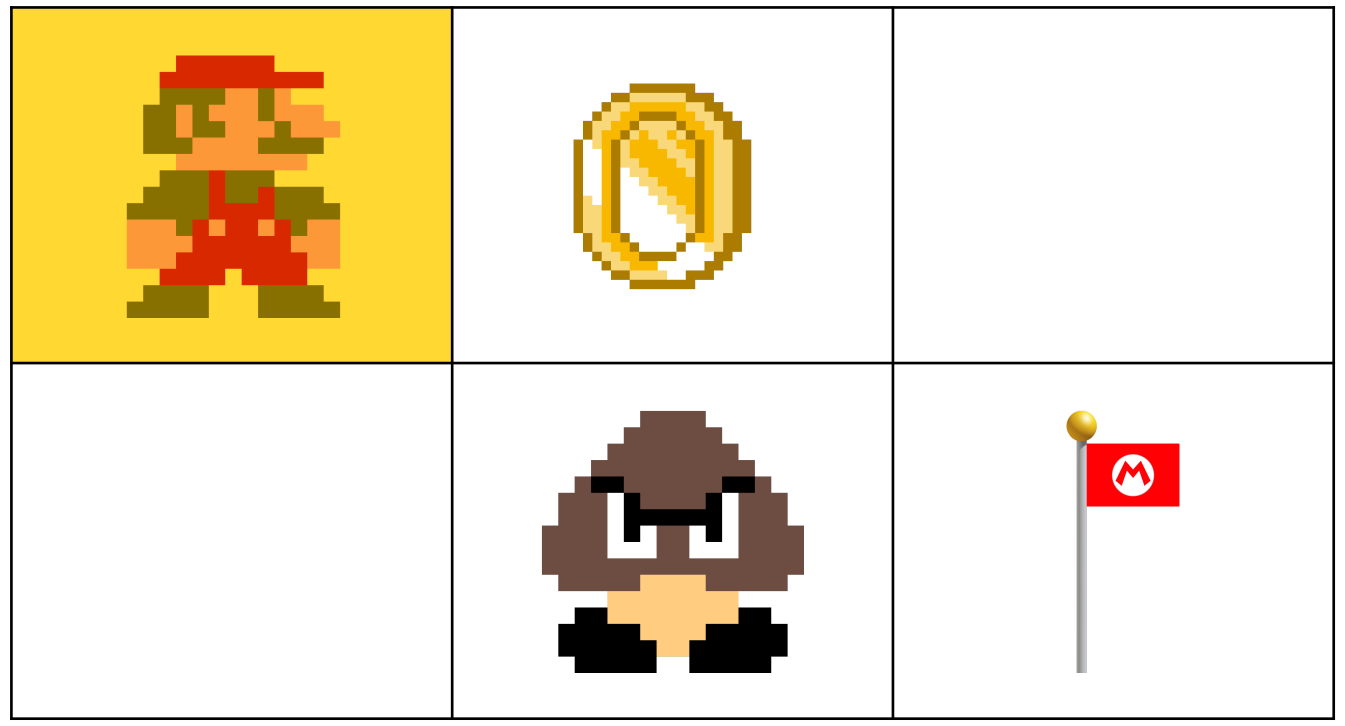
Q-Learning example

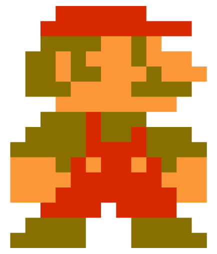




Policy π

Q-Learning example

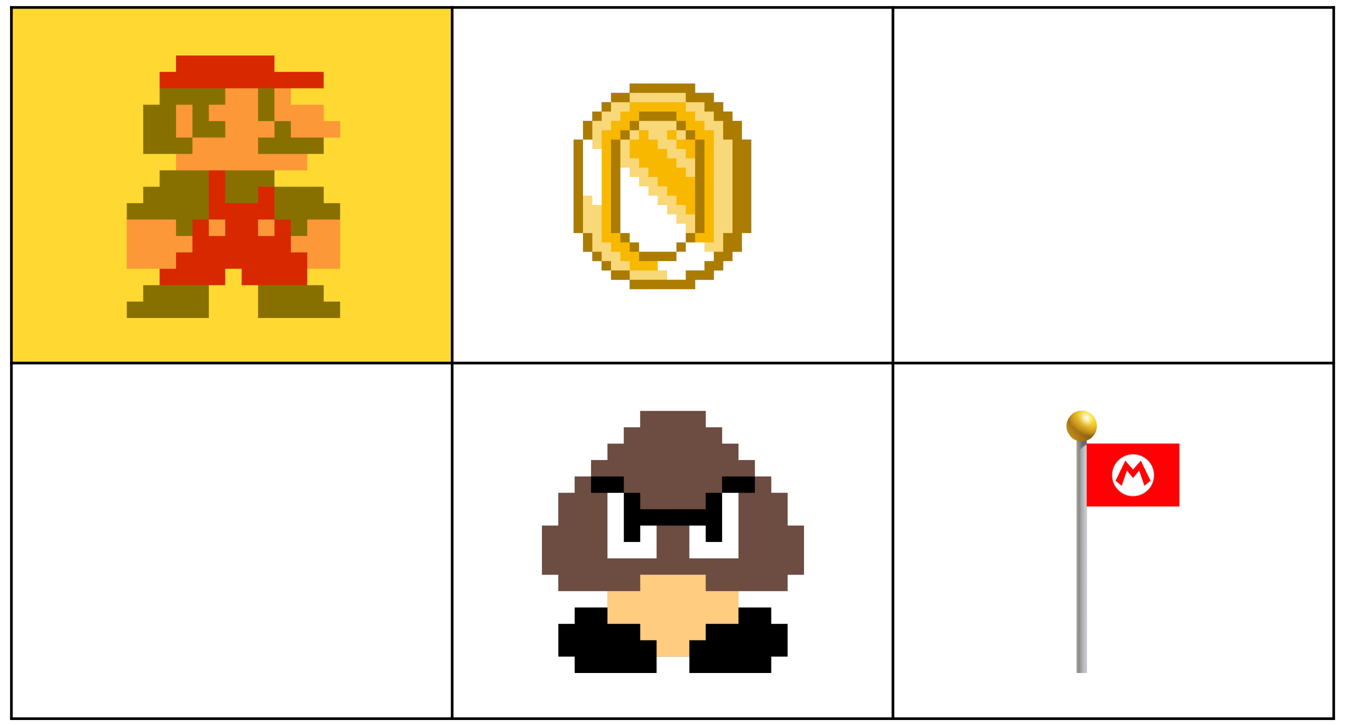


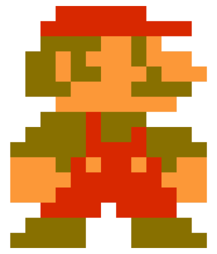


Policy π

Actions $\uparrow \downarrow \leftarrow \rightarrow$

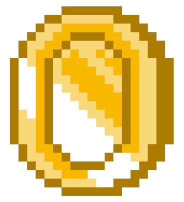
Q-Learning example





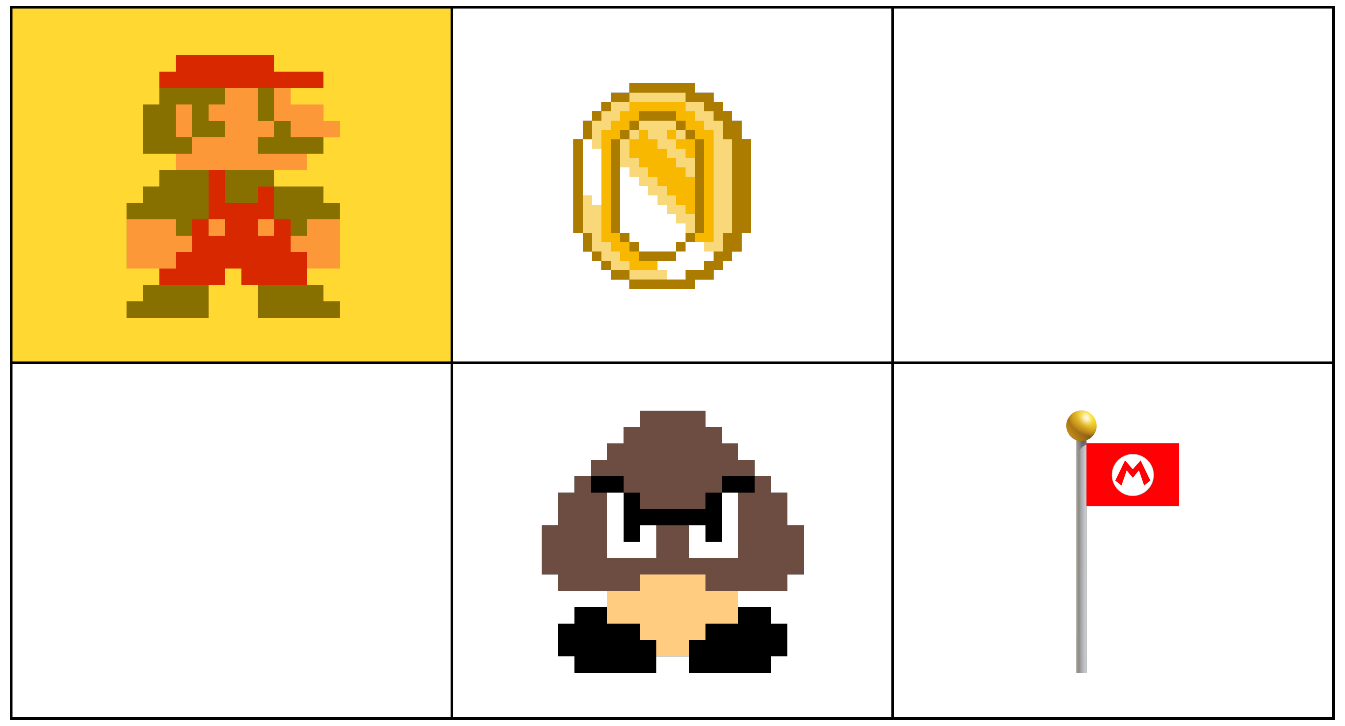
Policy π

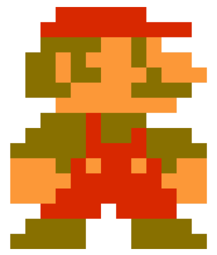
Actions $\uparrow \downarrow \leftarrow \rightarrow$

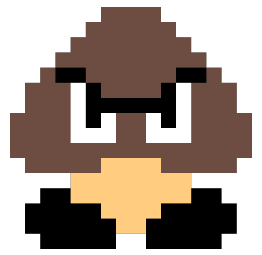


+1

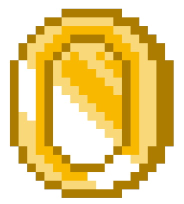
Q-Learning example



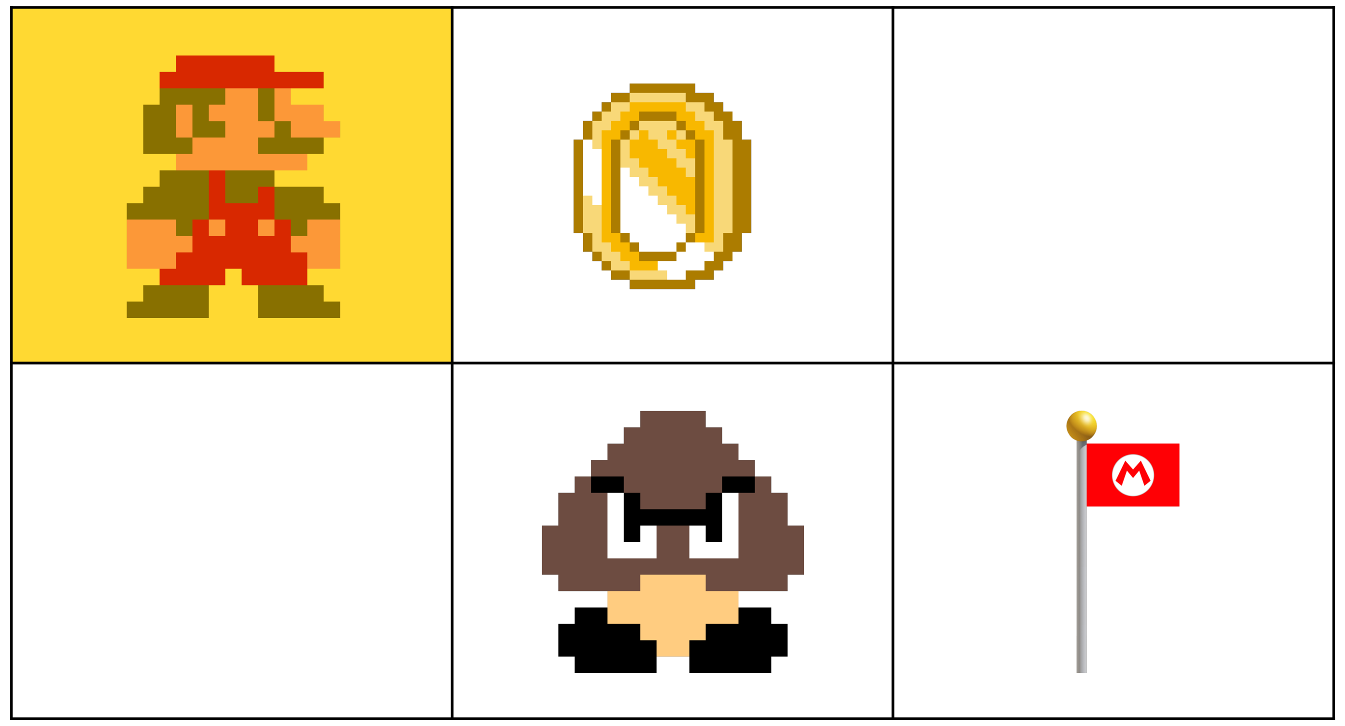
 Policy π

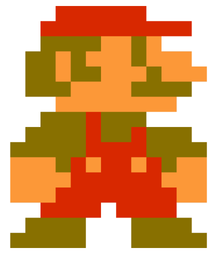
 -10

Actions $\uparrow \downarrow \leftarrow \rightarrow$

 +1

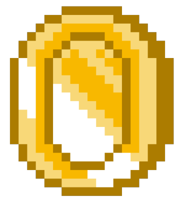
Q-Learning example



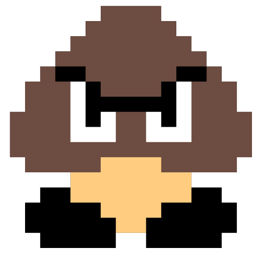


Policy π

Actions ↑ ↓ ← →



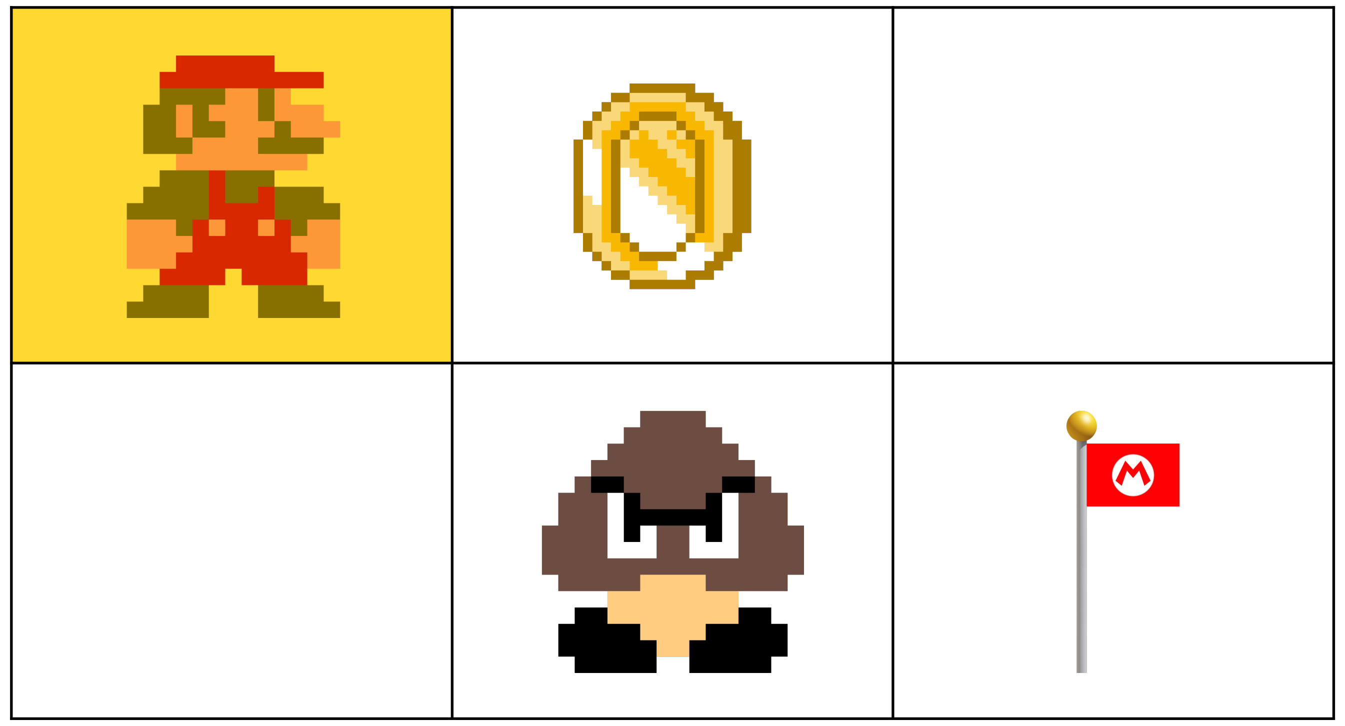
+1

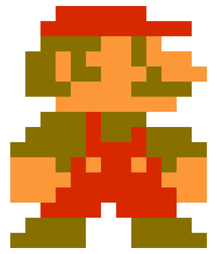


-10

+10
(terminal)

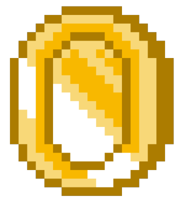
Q-Learning example



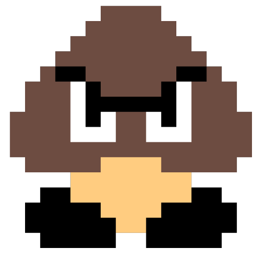


Policy π

Actions $\uparrow \downarrow \leftarrow \rightarrow$



+1

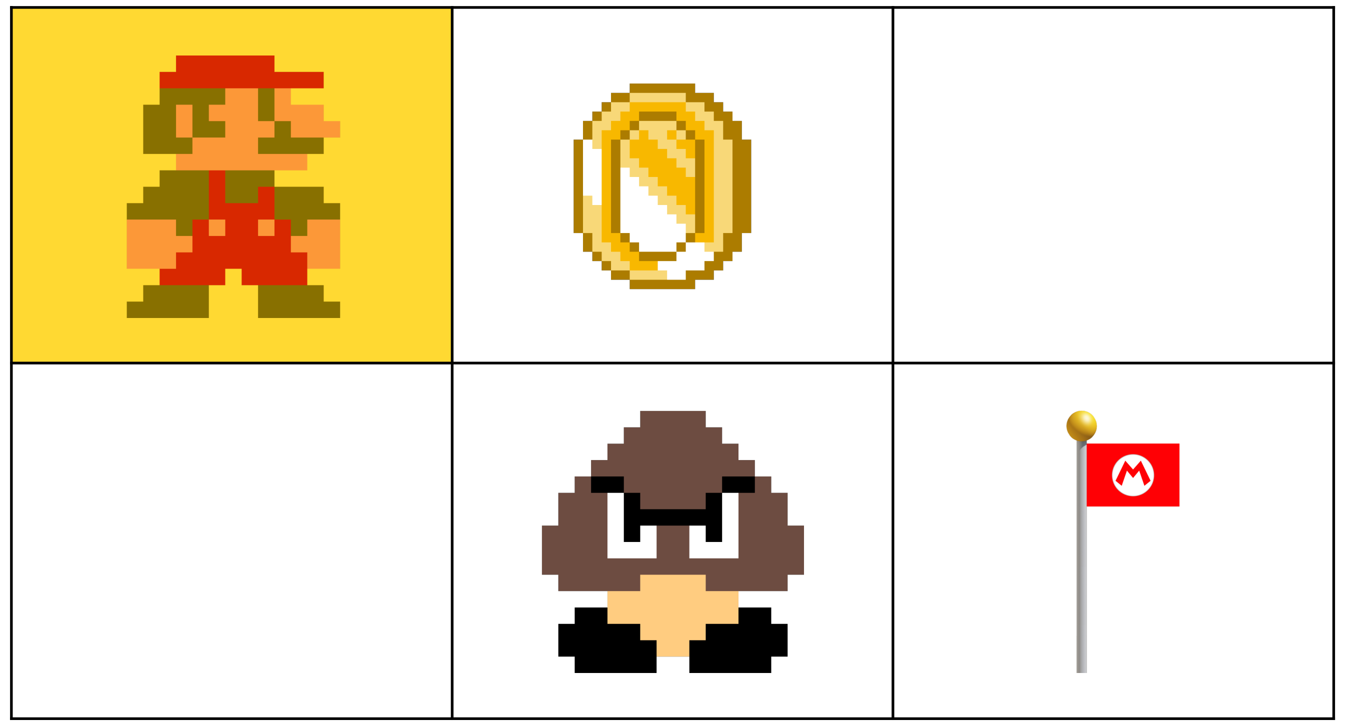


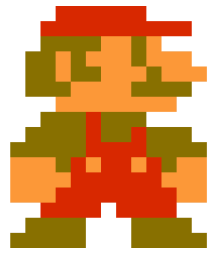
-10

+10
(terminal)

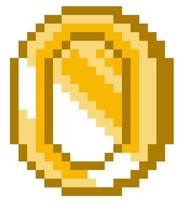
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma\ max_{a'}\ Q(S_{t+1}, a') - Q(S_t, A_t))$$

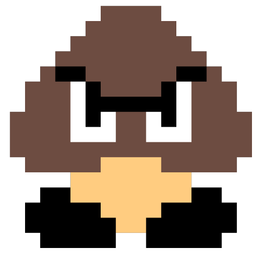
Q-Learning example



 Policy π

Actions $\uparrow \downarrow \leftarrow \rightarrow$

 +1

 -10

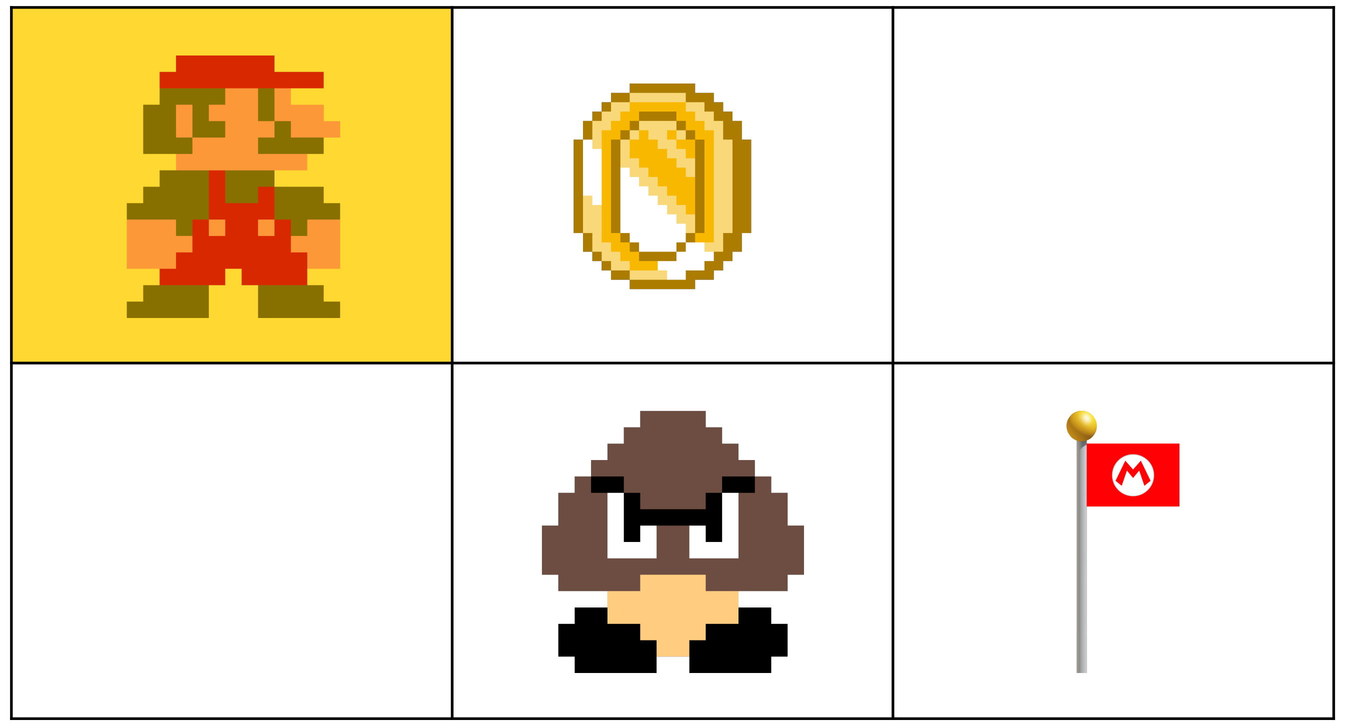
 +10 (terminal)

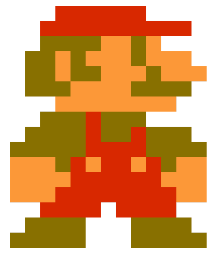
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \, max_{a'} \, Q(S_{t+1}, a') - Q(S_t, A_t))$$

$$\alpha = 0.1$$

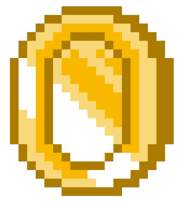
$$\gamma = 0.99$$

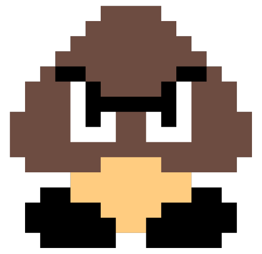
Q-Learning example



 Policy π

Actions $\uparrow \downarrow \leftarrow \rightarrow$

 +1

 -10

 +10 (terminal)

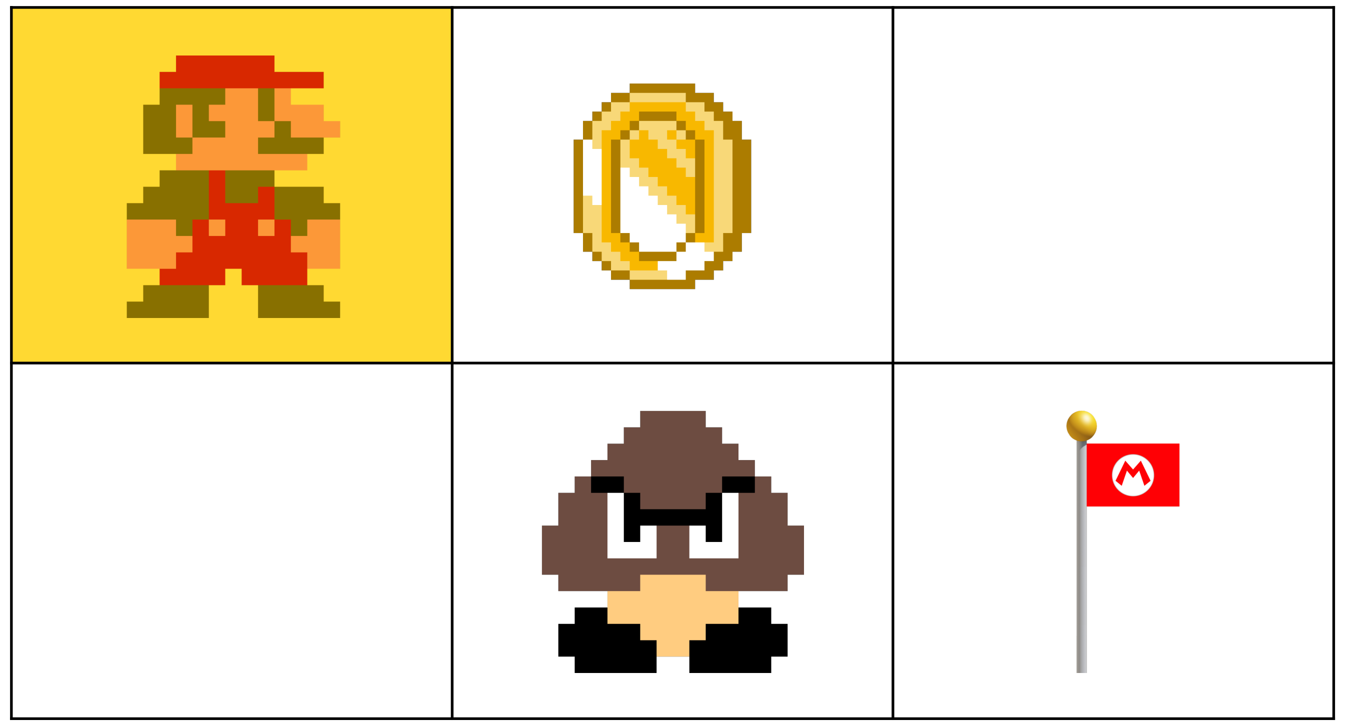
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \, max_{a'} \, Q(S_{t+1}, a') - Q(S_t, A_t))$$

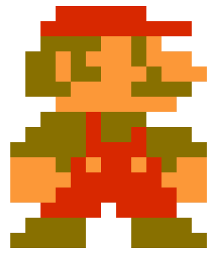
$$\alpha = 0.1$$

$$\gamma = 0.99$$

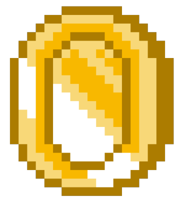
How to represent the Q-Table?

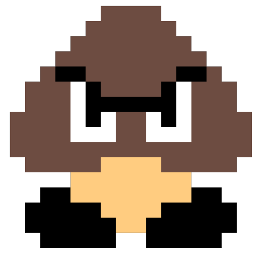
Q-Learning example



 Policy π

Actions $\uparrow \downarrow \leftarrow \rightarrow$

 +1

 -10

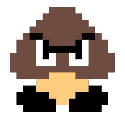
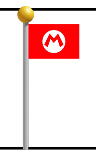
 +10 (terminal)

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

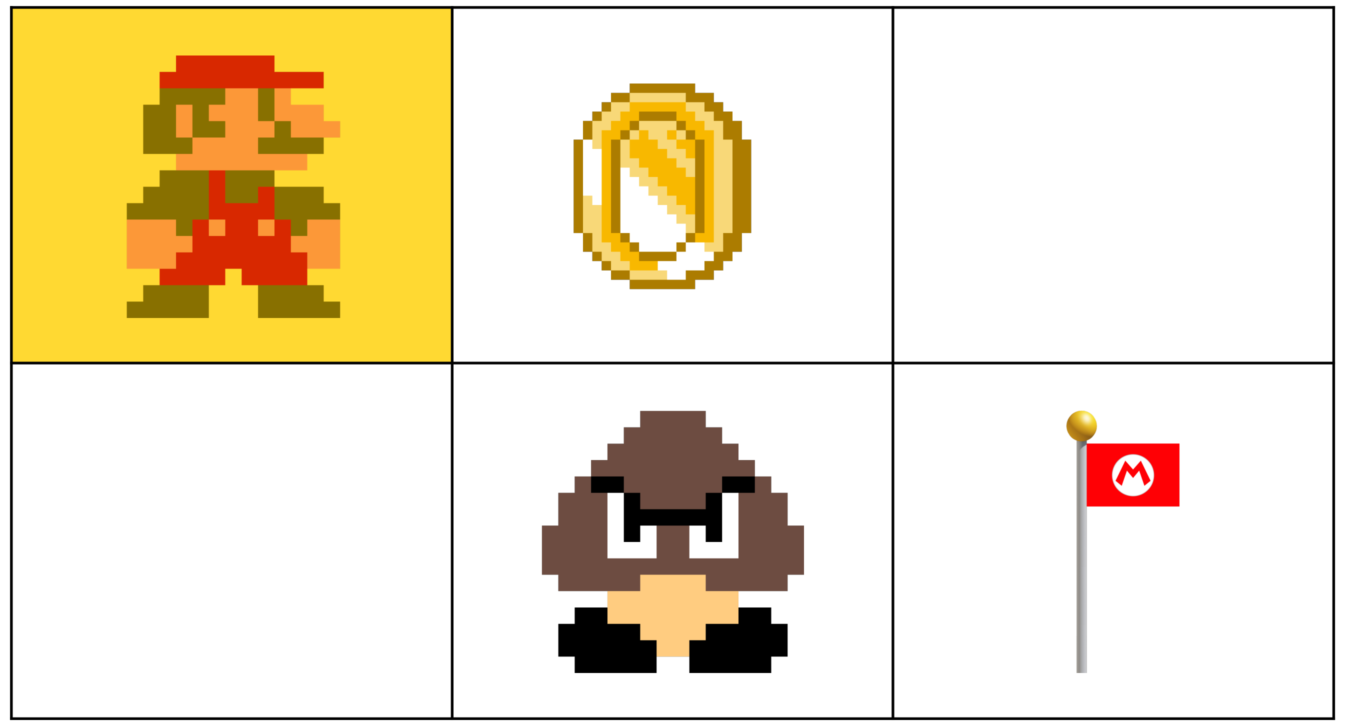
$$\alpha = 0.1$$

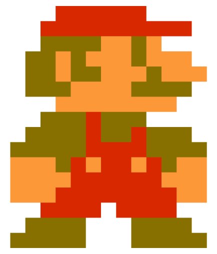
$$\gamma = 0.99$$

How to represent the Q-Table?

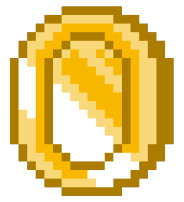
	$\leftarrow$	$\rightarrow$	$\uparrow$	$\downarrow$
				
				
				

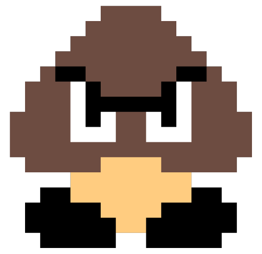
Q-Learning example



 Policy π

Actions $\uparrow \downarrow \leftarrow \rightarrow$

 +1

 -10

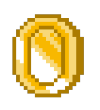
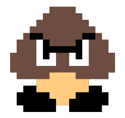
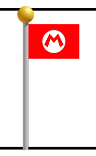
 +10 (terminal)

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

$$\alpha = 0.1$$

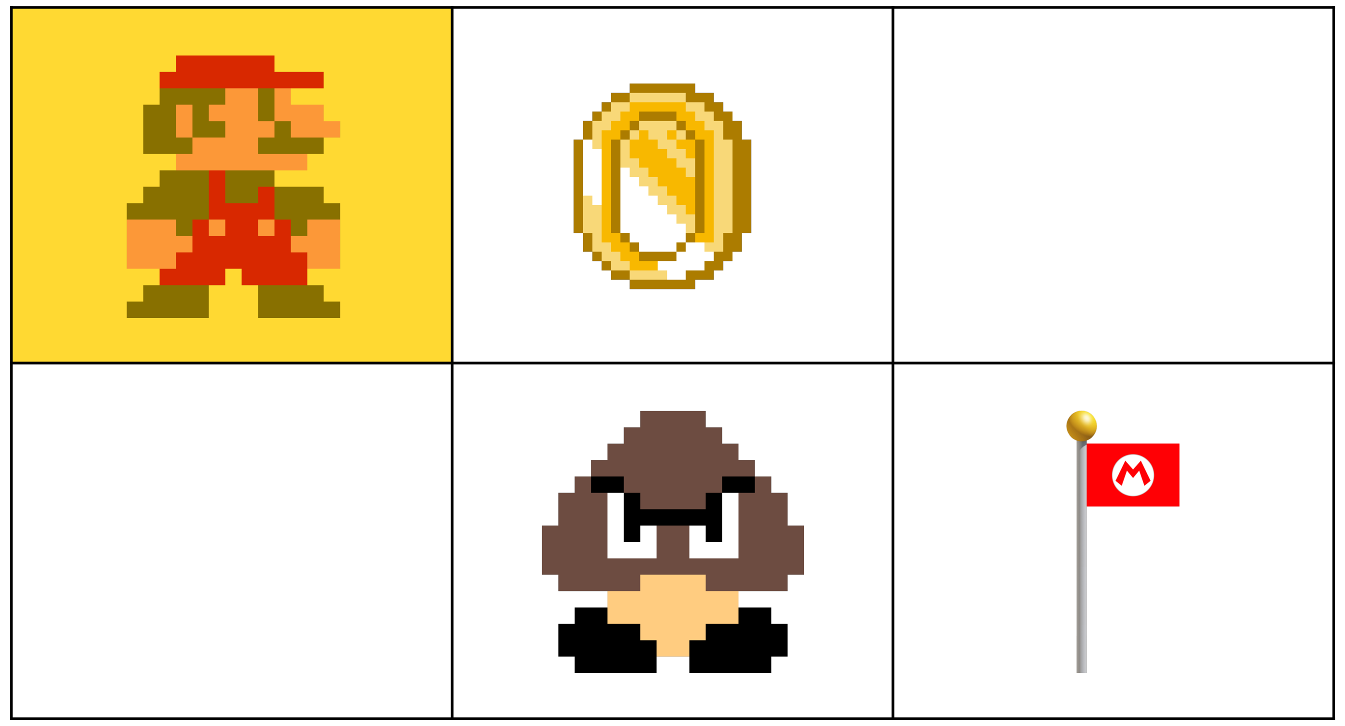
$$\gamma = 0.99$$

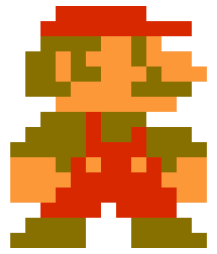
How to represent the Q-Table?

	$\leftarrow$	$\rightarrow$	$\uparrow$	$\downarrow$
				
				
				

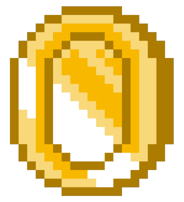
How to move?

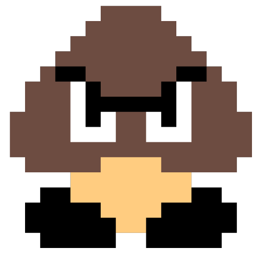
Q-Learning example



 Policy π

Actions $\uparrow \downarrow \leftarrow \rightarrow$

 +1

 -10

 +10 (terminal)

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

$$\alpha = 0.1$$

$$\gamma = 0.99$$

How to represent the Q-Table?

	$\leftarrow$	$\rightarrow$	$\uparrow$	$\downarrow$

How to move?

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

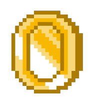
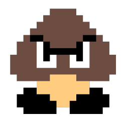
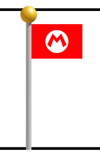
Q-Learning example

Q-Learning example

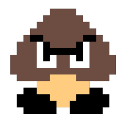
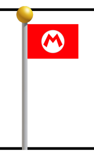
Initialize $Q(s,a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Q-Learning example

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

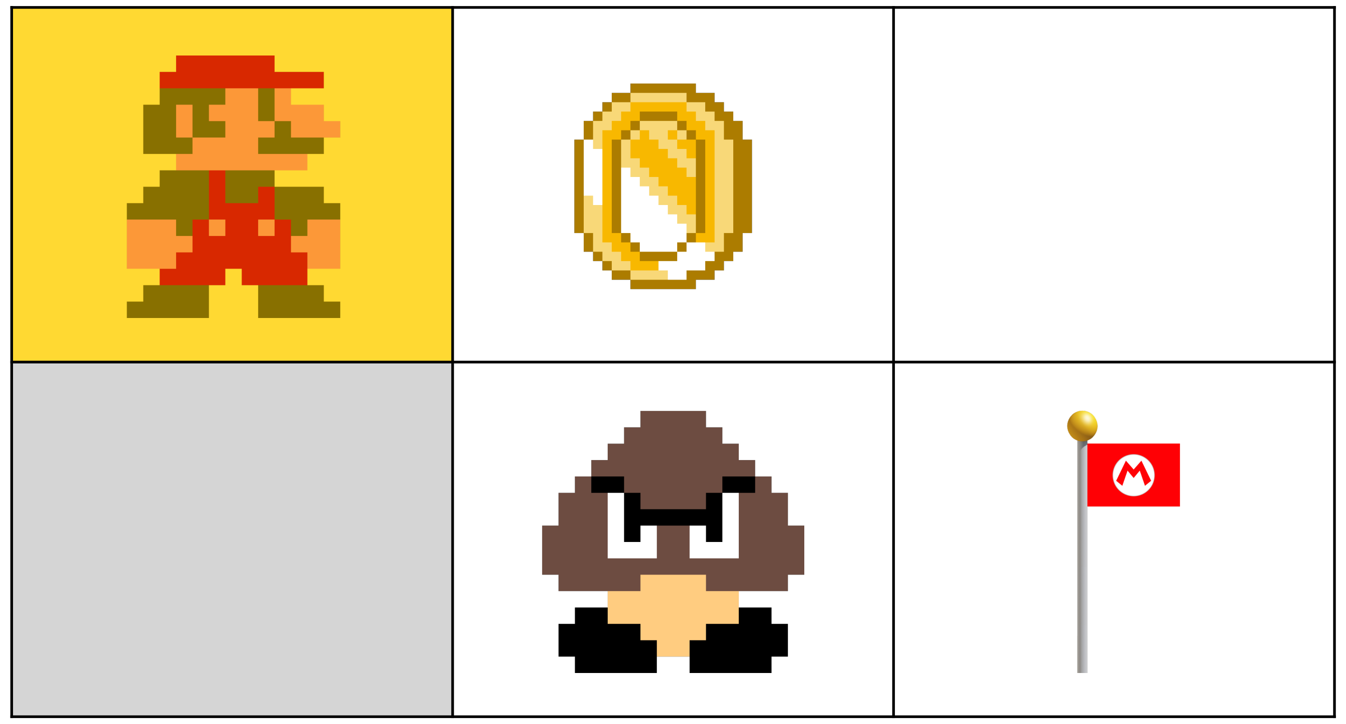
	$\leftarrow$	$\rightarrow$	$\uparrow$	$\downarrow$
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0

Q-Learning example

	←	→	↑	↓
	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

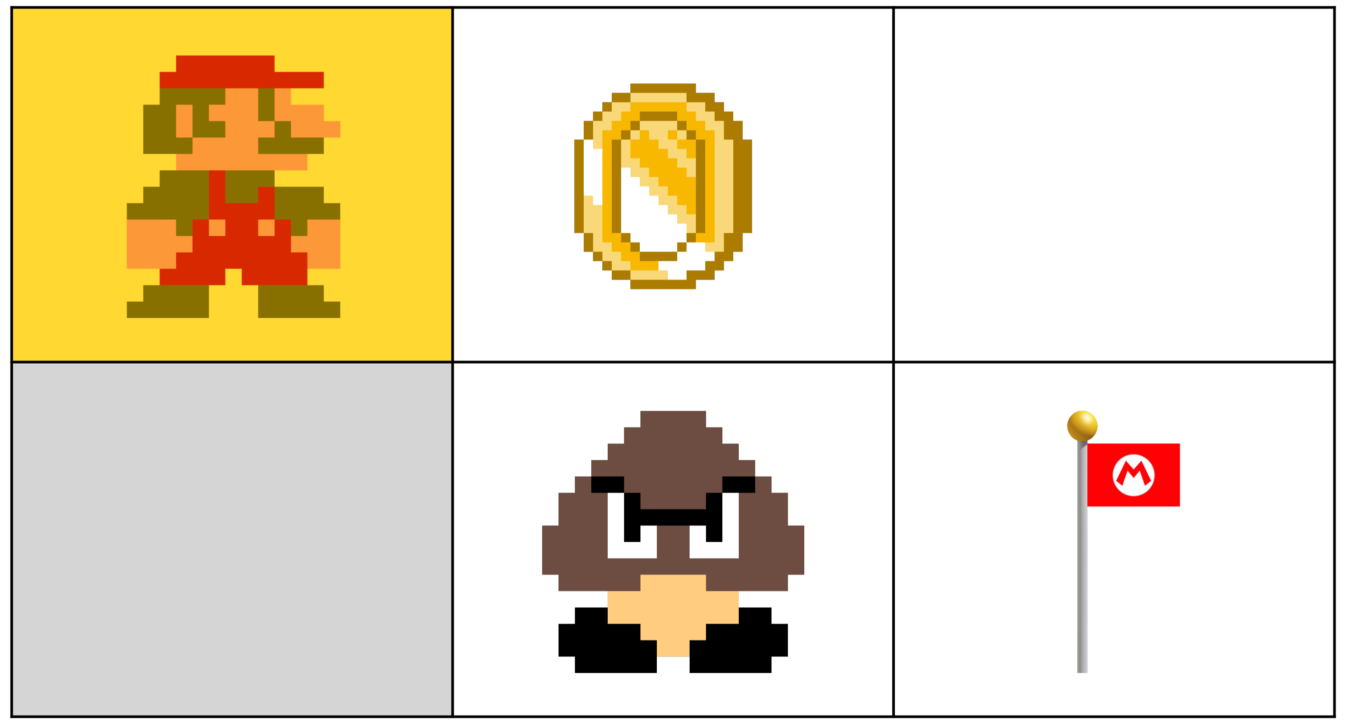
	←	→	↑	↓
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0



Q-Learning example

	←	→	↑	↓
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0

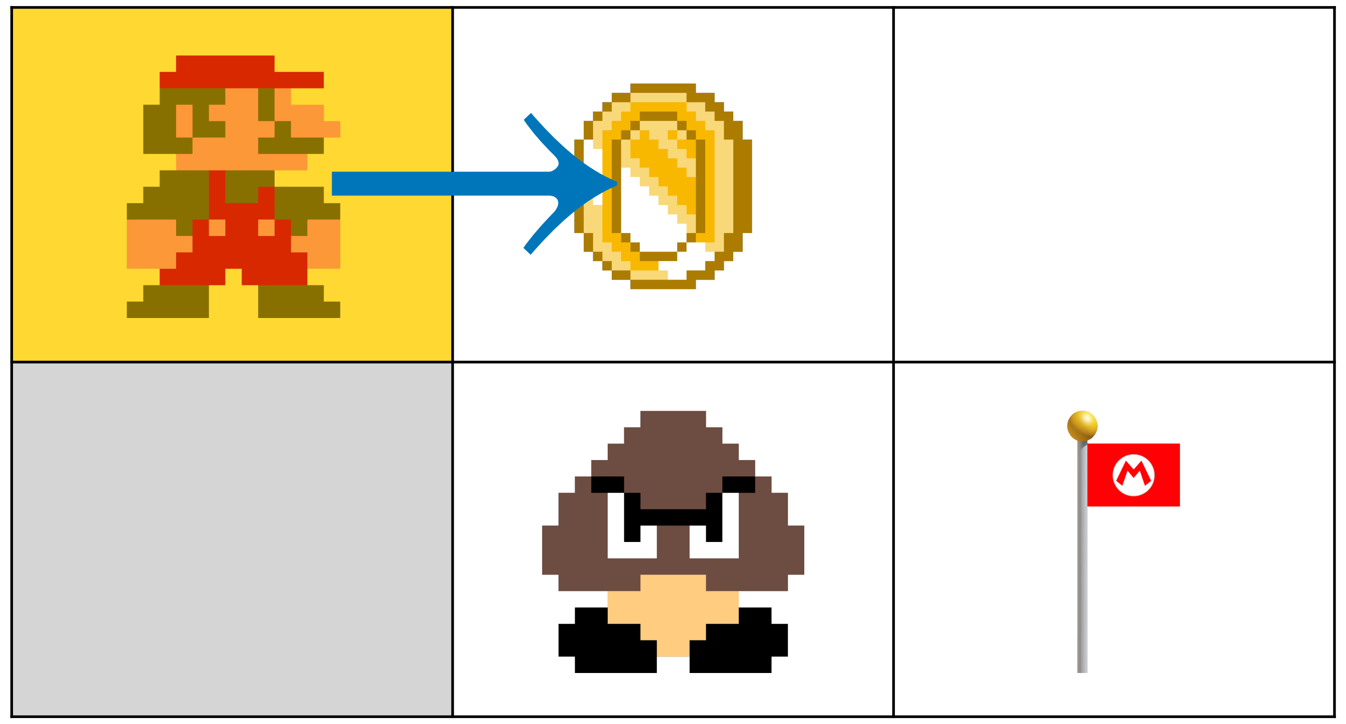
Choose A from S using policy derived from Q (e.g., ε-greedy)



Q-Learning example

	←	→	↑	↓
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0

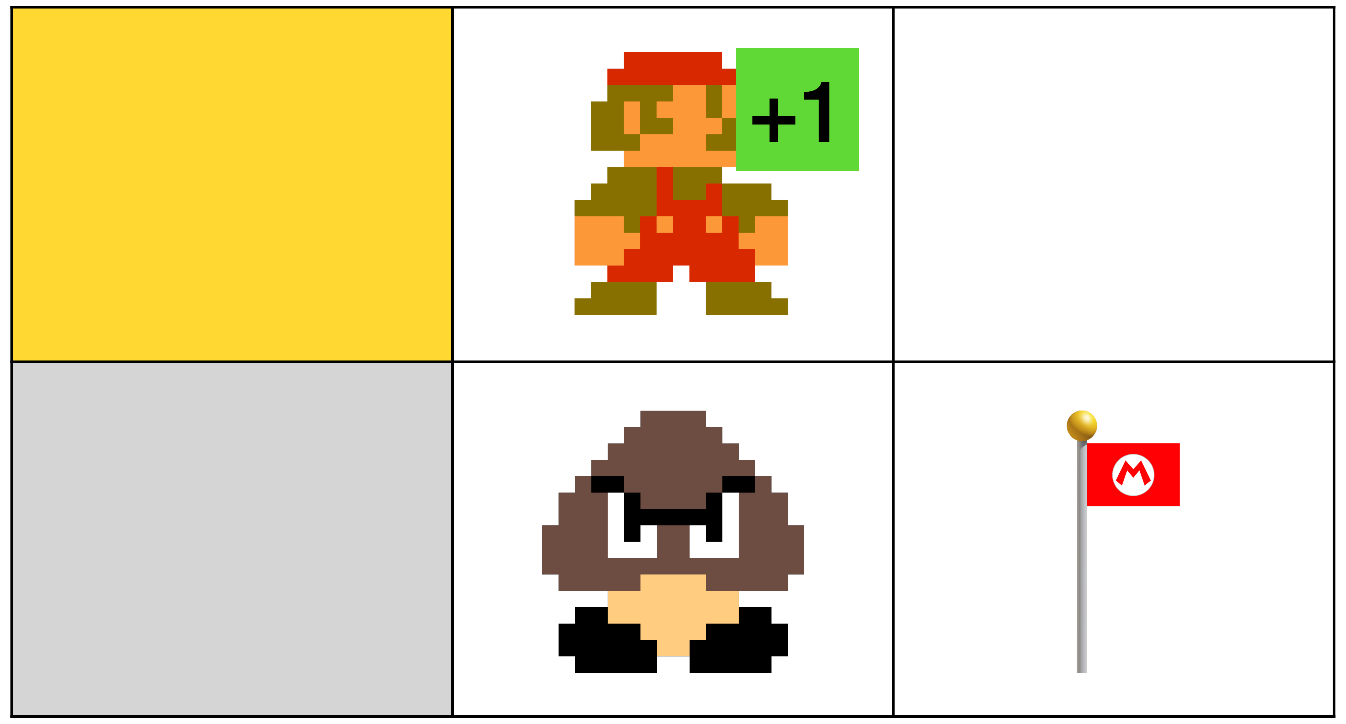
Choose A from S using policy derived from Q (e.g., ε-greedy)



Q-Learning example

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

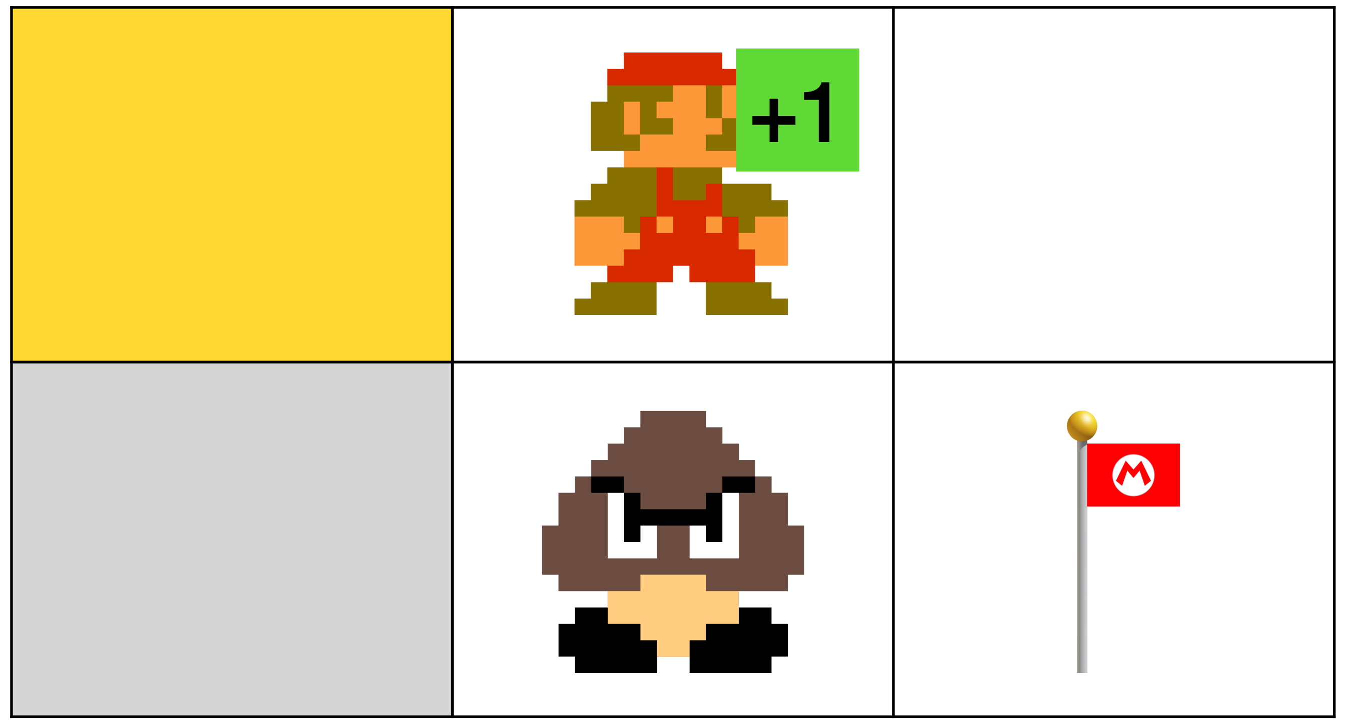
Take action A, observe R, S'



Q-Learning example

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

Take action A, observe R, S'



Update $Q(S, A)$

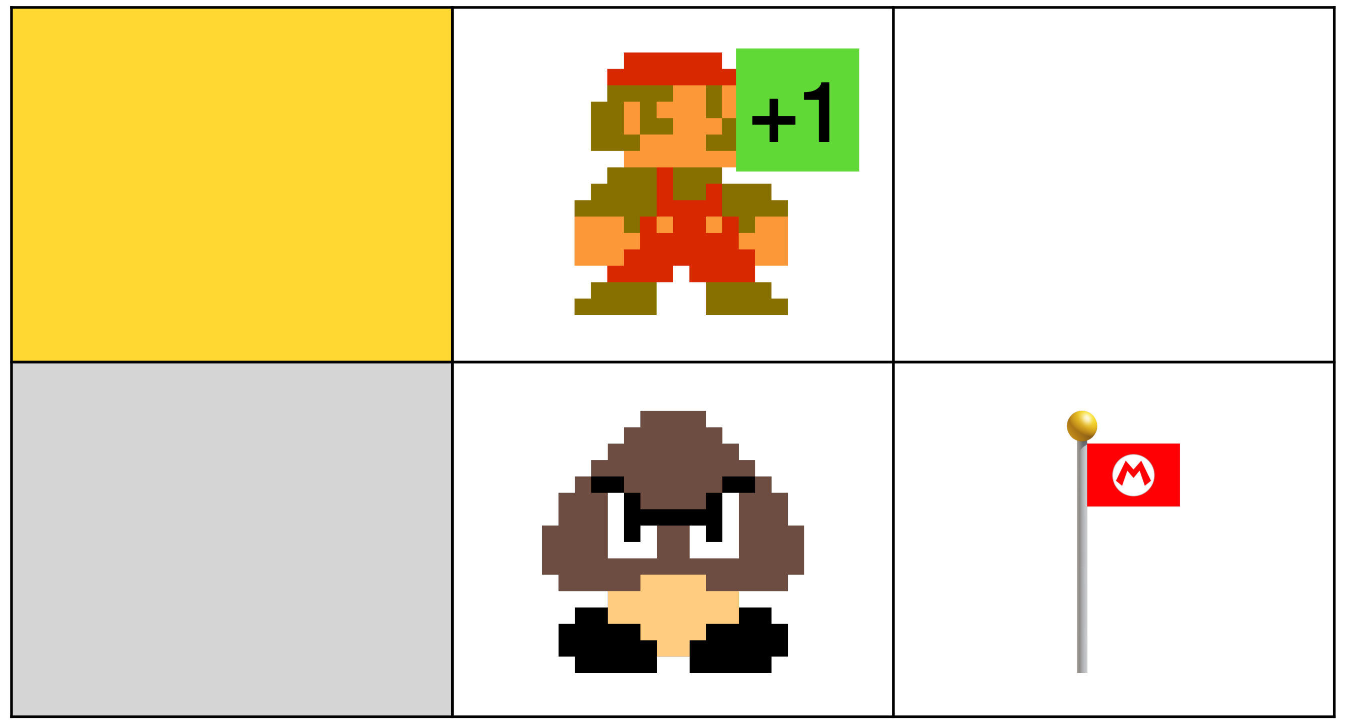
Q-Learning example

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

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

$\alpha = 0.1$

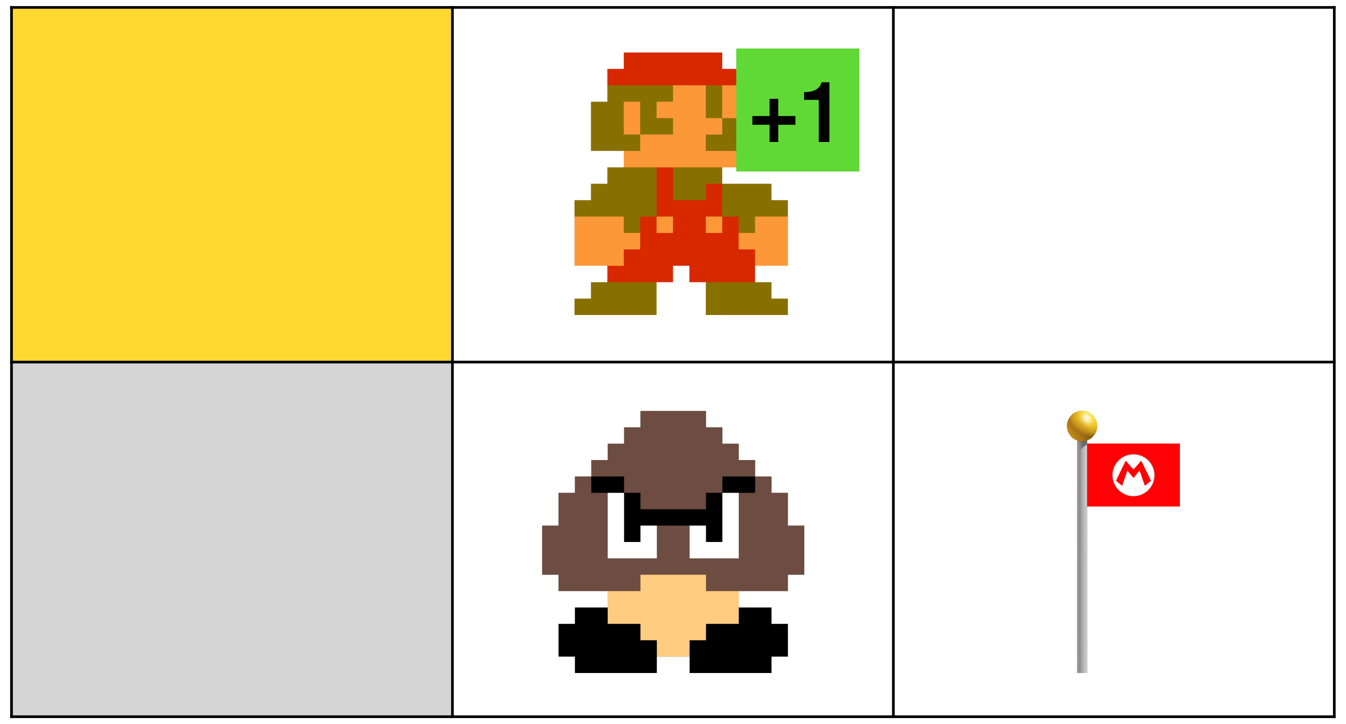
$\gamma = 0.99$

	←	→	↑	↓
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	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

$\alpha = 0.1$

$\gamma = 0.99$

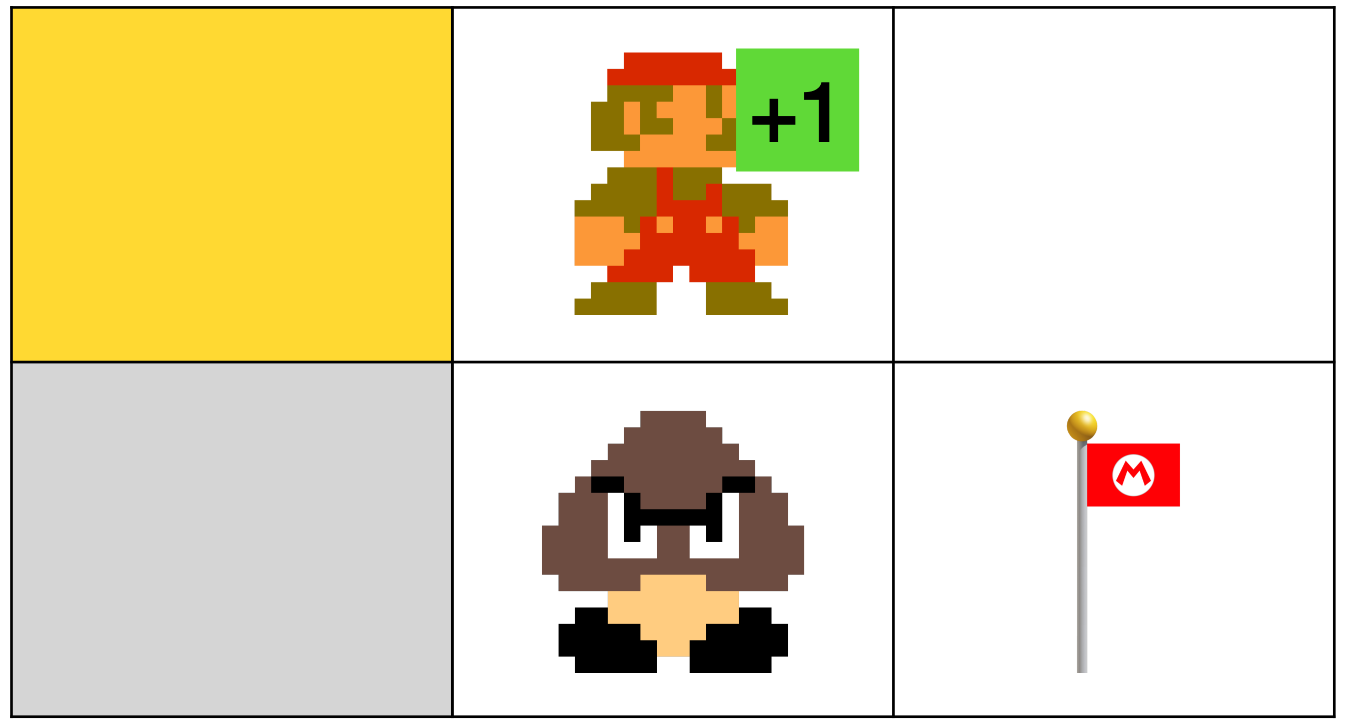
	$\leftarrow$	$\rightarrow$	$\uparrow$	$\downarrow$
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0

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(initial, →) = 0 + 0.1*[1 + 0.99*0 - 0] = 0.1

Take action A, observe R, S'



Q-Learning example

$\alpha = 0.1$

$\gamma = 0.99$

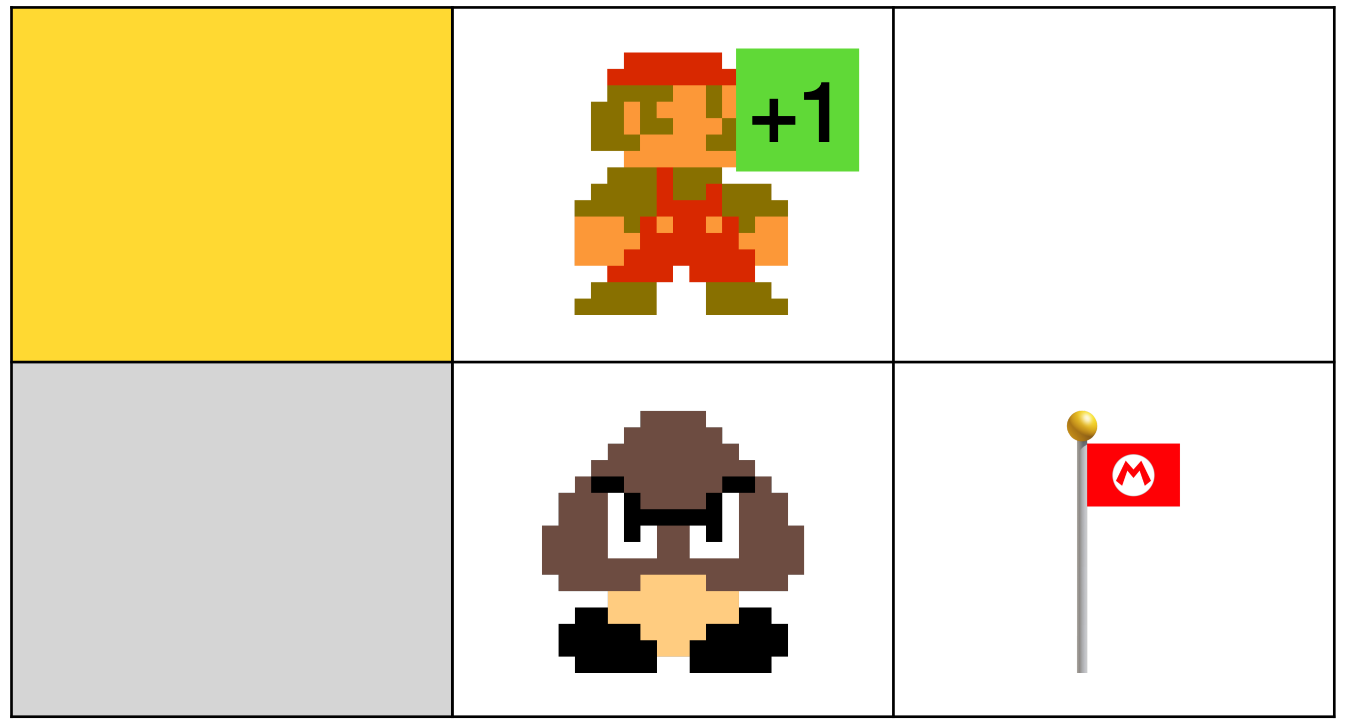
	$\leftarrow$	$\rightarrow$	$\uparrow$	$\downarrow$
	0	0.1	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0

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(initial, →) = 0 + 0.1*[1 + 0.99*0 - 0] = 0.1

Take action A, observe R, S'



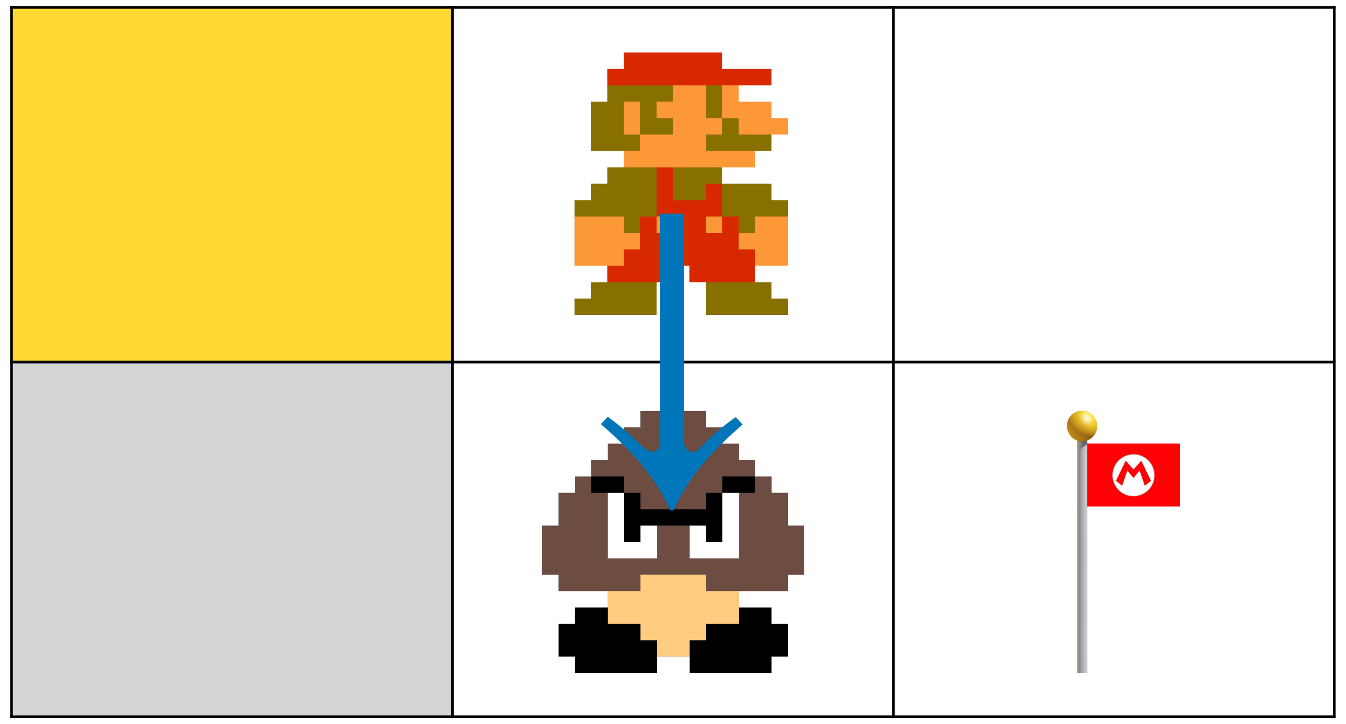
Q-Learning example

$\alpha = 0.1$

$\gamma = 0.99$

	←	→	↑	↓
	0	0.1	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0

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



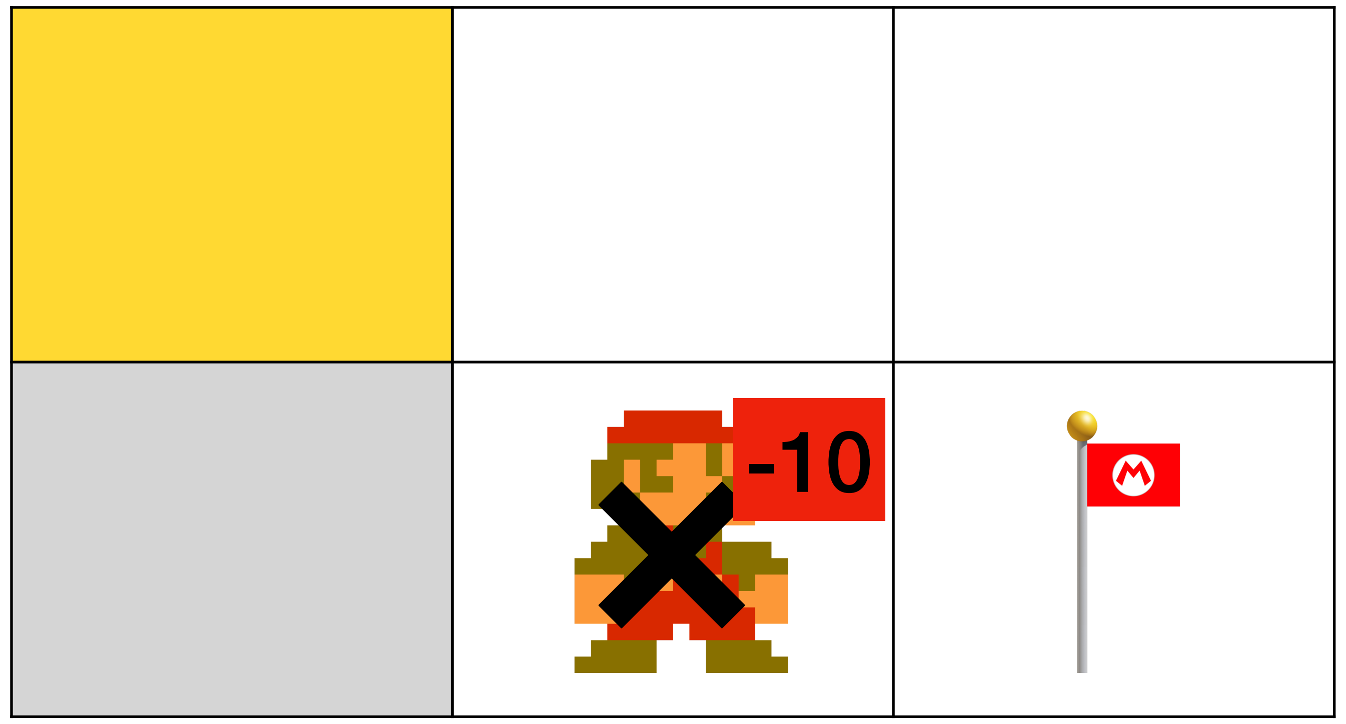
Q-Learning example

$\alpha = 0.1$

$\gamma = 0.99$

	$\leftarrow$	$\rightarrow$	$\uparrow$	$\downarrow$
	0	0.1	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0

Take action A, observe R, S'



Q-Learning example

$\alpha = 0.1$

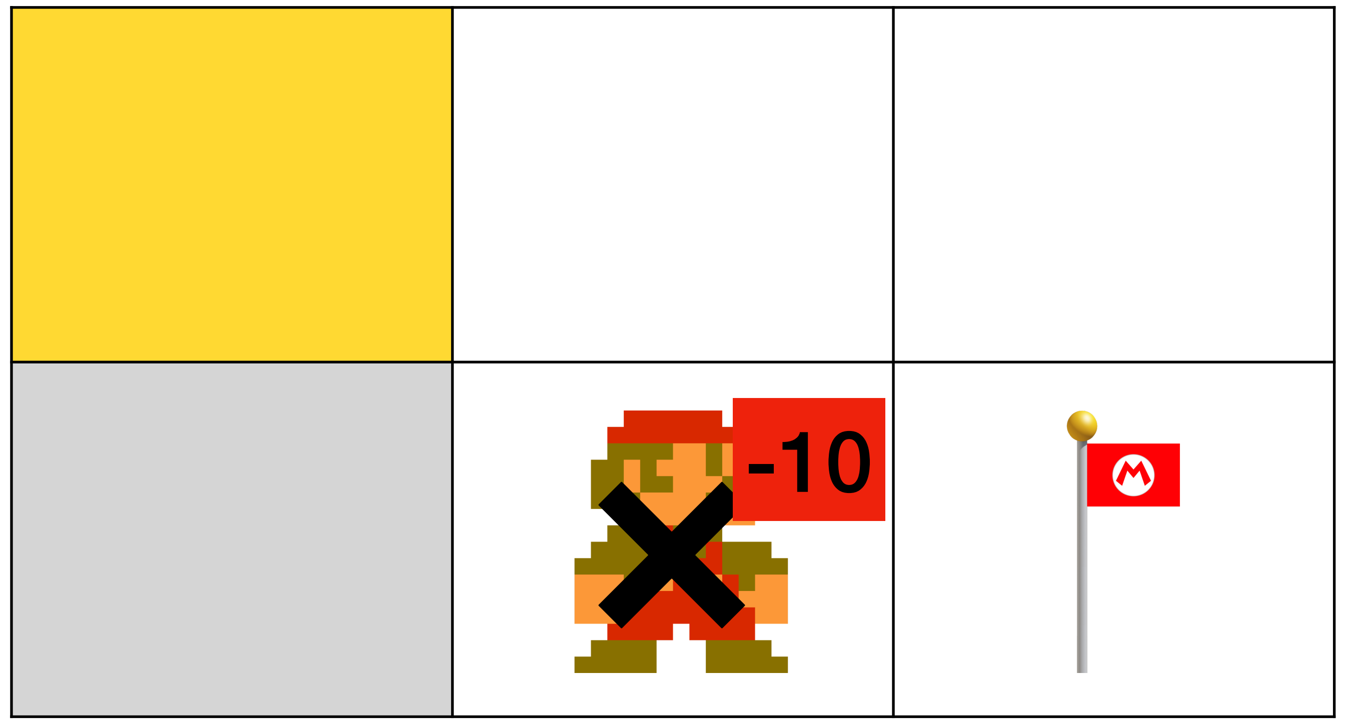
$\gamma = 0.99$

	$\leftarrow$	$\rightarrow$	$\uparrow$	$\downarrow$
	0	0.1	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	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

$\alpha = 0.1$

$\gamma = 0.99$

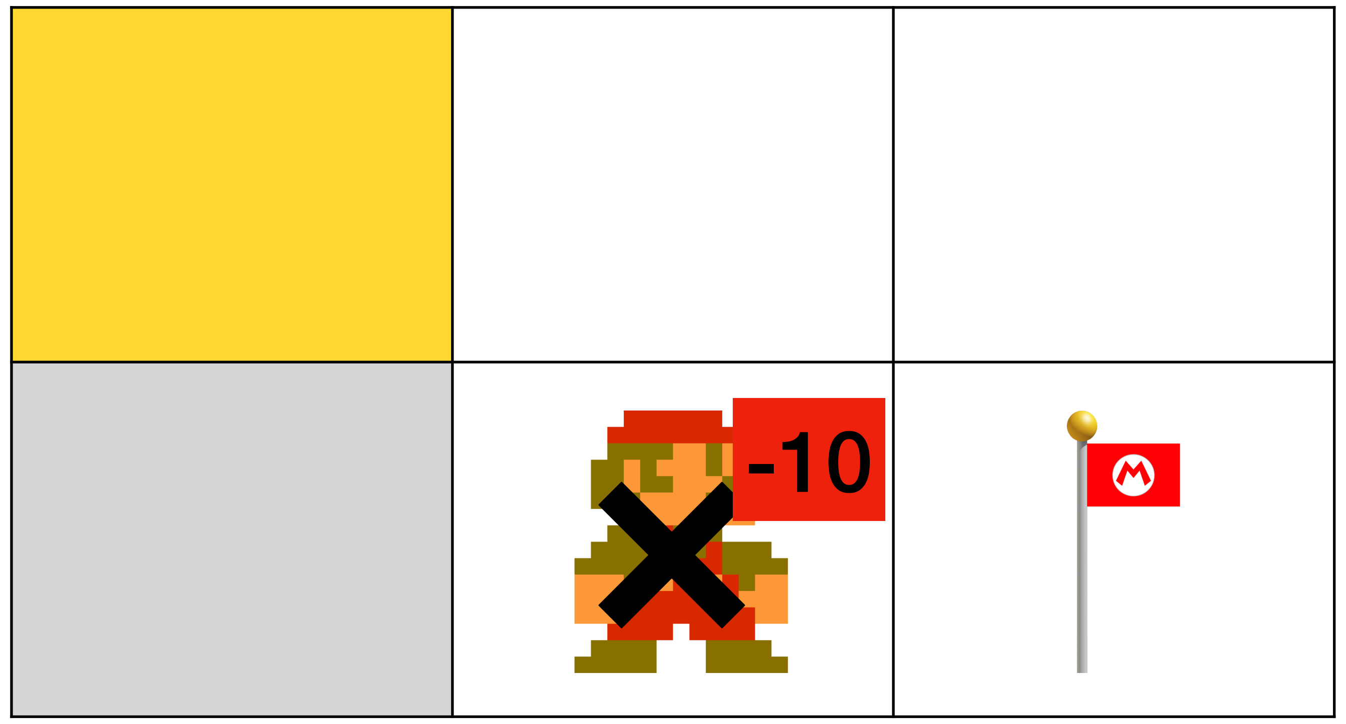
	$\leftarrow$	$\rightarrow$	$\uparrow$	$\downarrow$
	0	0.1	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0

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(state_2, ↓) = 0 + 0.1*[-10 + 0.99*0 - 0] = -1

Take action A, observe R, S'



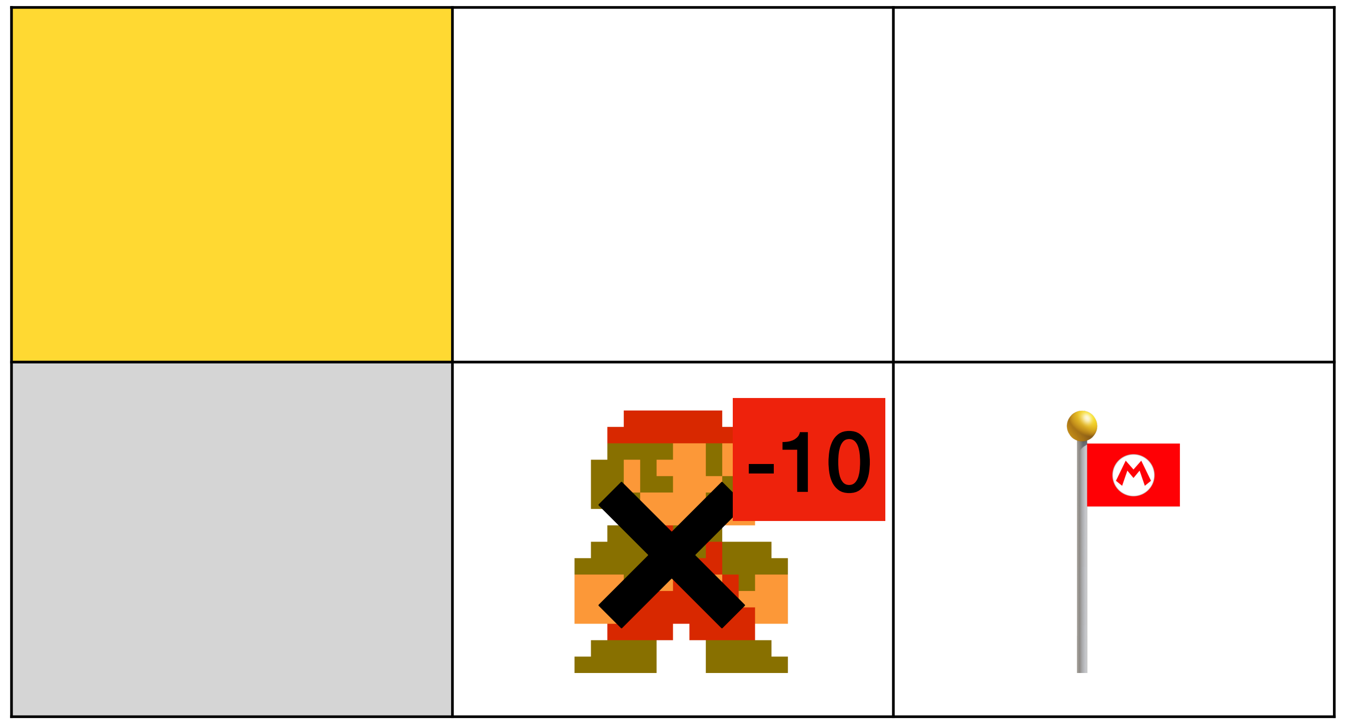
Q-Learning example

$\alpha = 0.1$

$\gamma = 0.99$

	$\leftarrow$	$\rightarrow$	$\uparrow$	$\downarrow$
	0	0.1	0	0
	0	0	0	-1
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0

Take action A, observe R, S'



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(state_2, ↓) = 0 + 0.1*[-10 + 0.99*0 - 0] = -1

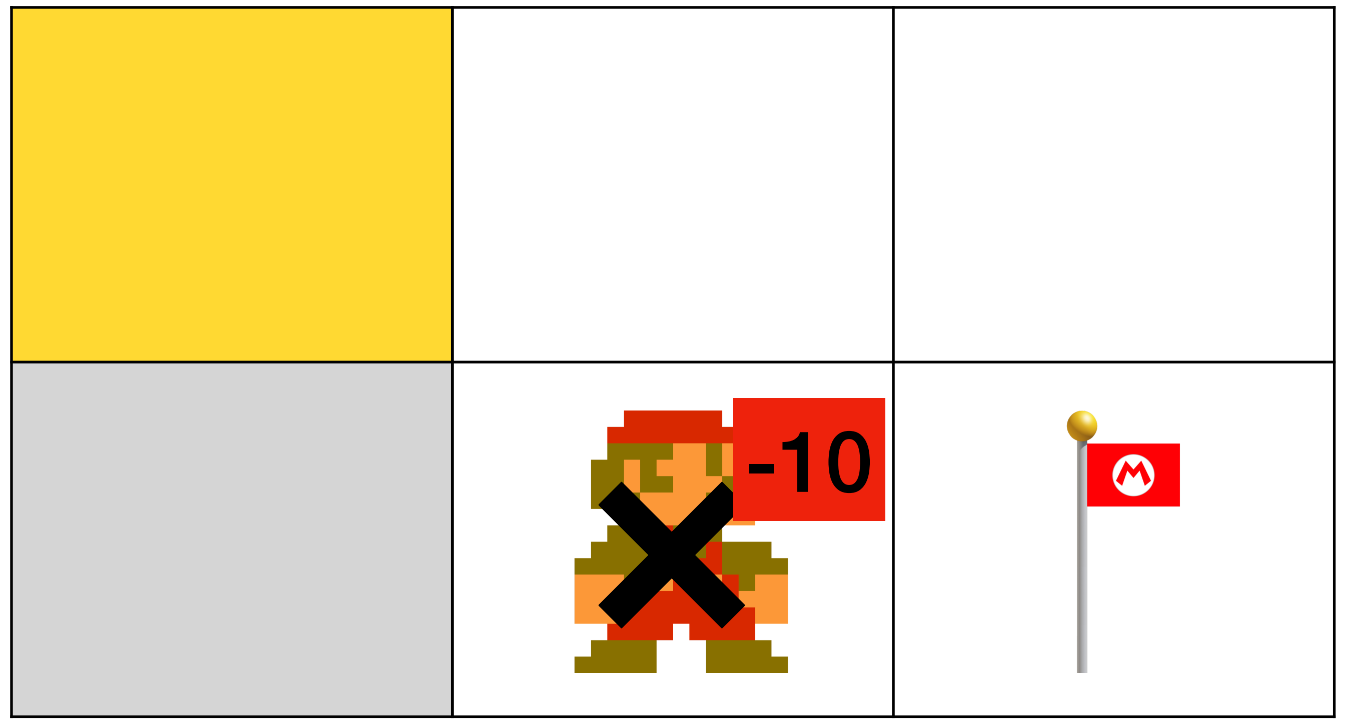
Q-Learning example

$\alpha = 0.1$

$\gamma = 0.99$

	$\leftarrow$	$\rightarrow$	$\uparrow$	$\downarrow$
	0	0.1	0	0
	0	0	0	-1
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0

Take action A, observe R, S'



Terminal state, Episode resets

Q-Learning example

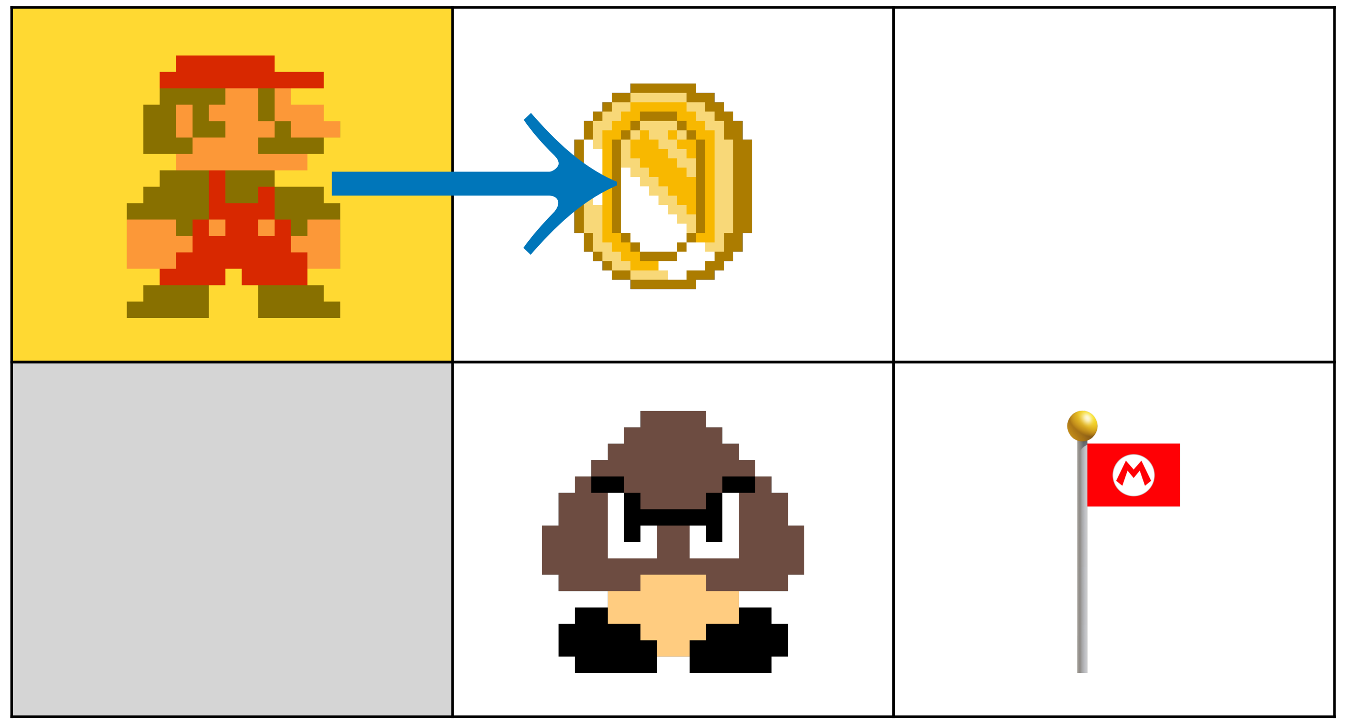
$\alpha = 0.1$

$\gamma = 0.99$

New episode

	$\leftarrow$	$\rightarrow$	$\uparrow$	$\downarrow$
	0	0.1	0	0
	0	0	0	-1
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0

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



Q-Learning example

$\alpha = 0.1$

$\gamma = 0.99$

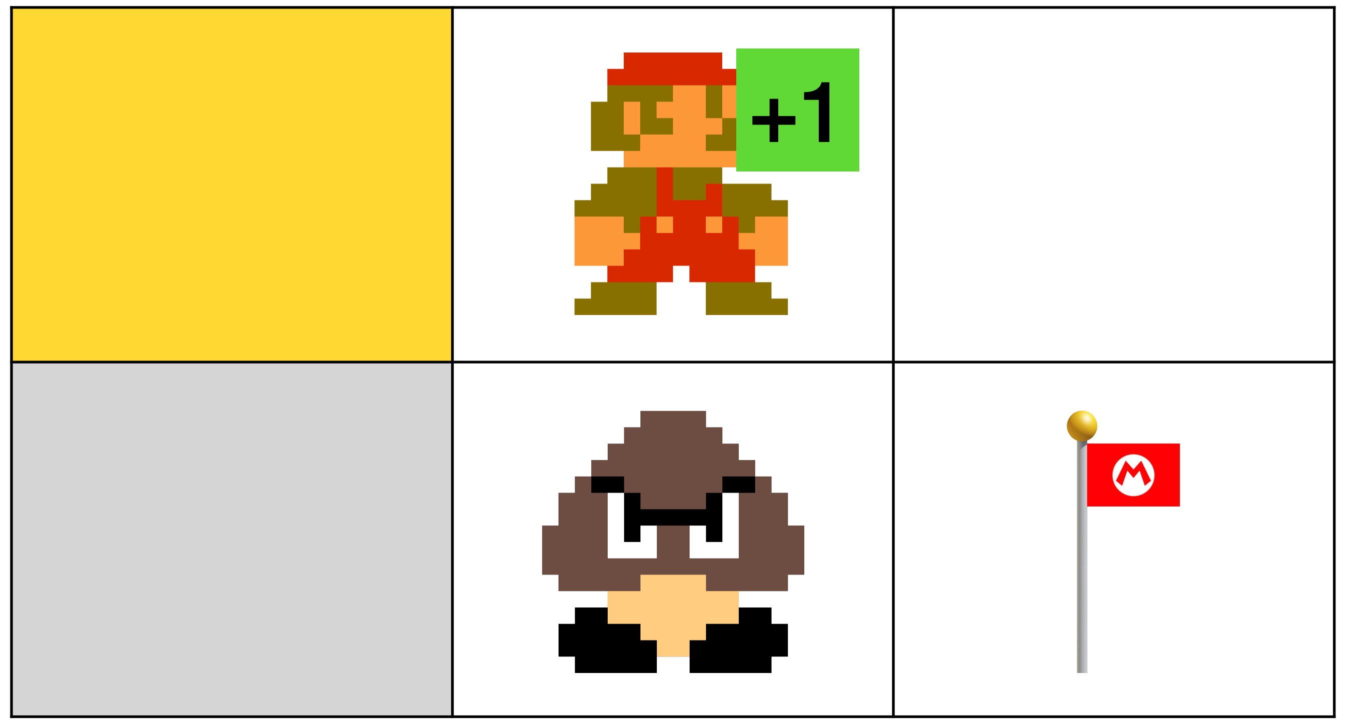
	←	→	↑	↓
	0	0.1	0	0
	0	0	0	-1
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0

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(initial, →) = 0.1 + 0.1*[1 + 0.99*0 - 0] = 0.2

Take action A, observe R, S'



Q-Learning example

$\alpha = 0.1$

$\gamma = 0.99$

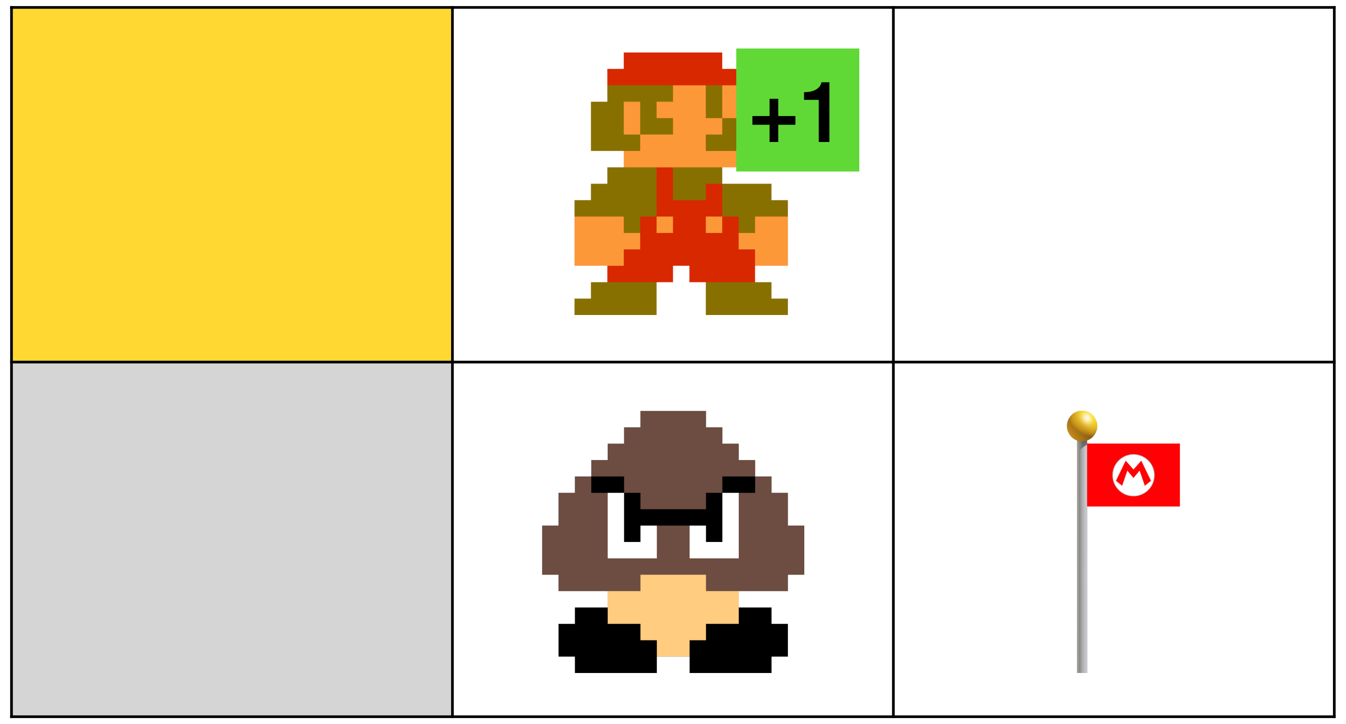
	$\leftarrow$	$\rightarrow$	$\uparrow$	$\downarrow$
	0	0.2	0	0
	0	0	0	-1
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0

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(initial, →) = 0.1 + 0.1*[1 + 0.99*0 - 0] = 0.2

Take action A, observe R, S'



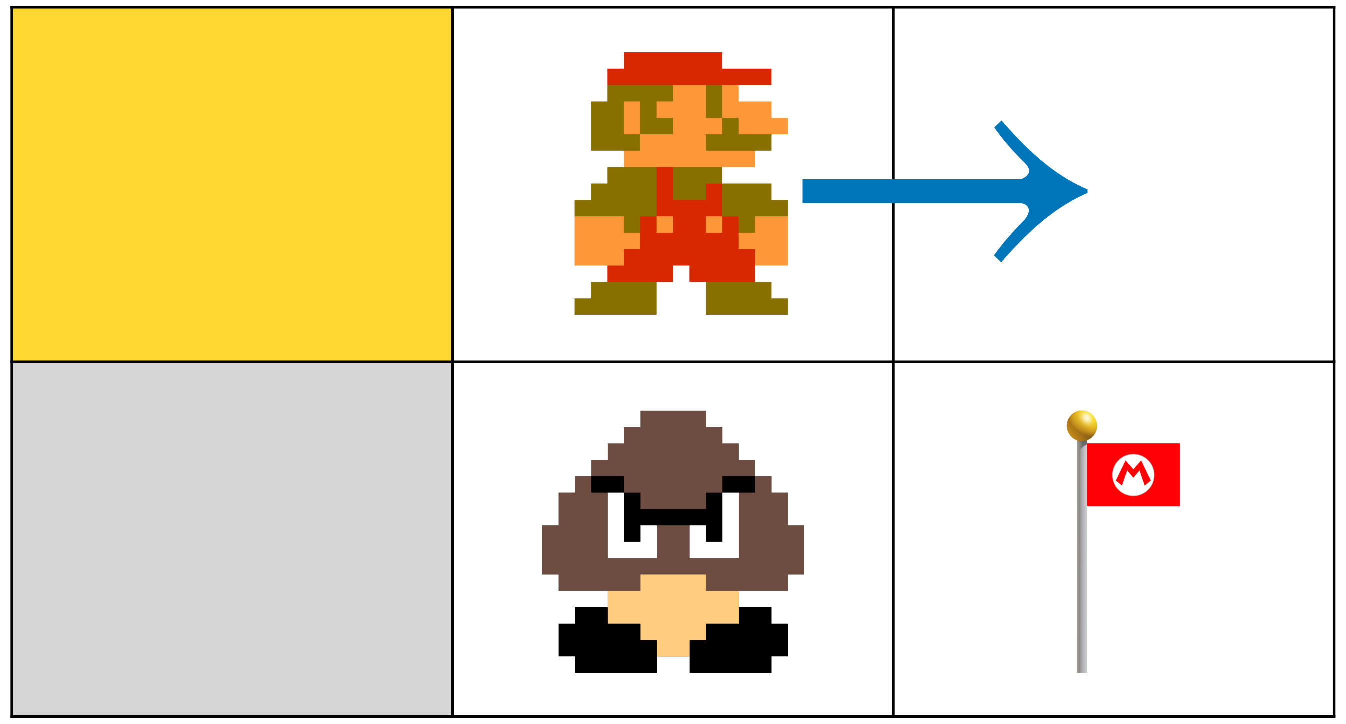
Q-Learning example

$\alpha = 0.1$

$\gamma = 0.99$

	$\leftarrow$	$\rightarrow$	$\uparrow$	$\downarrow$
	0	0.2	0	0
	0	0	0	-1
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0

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



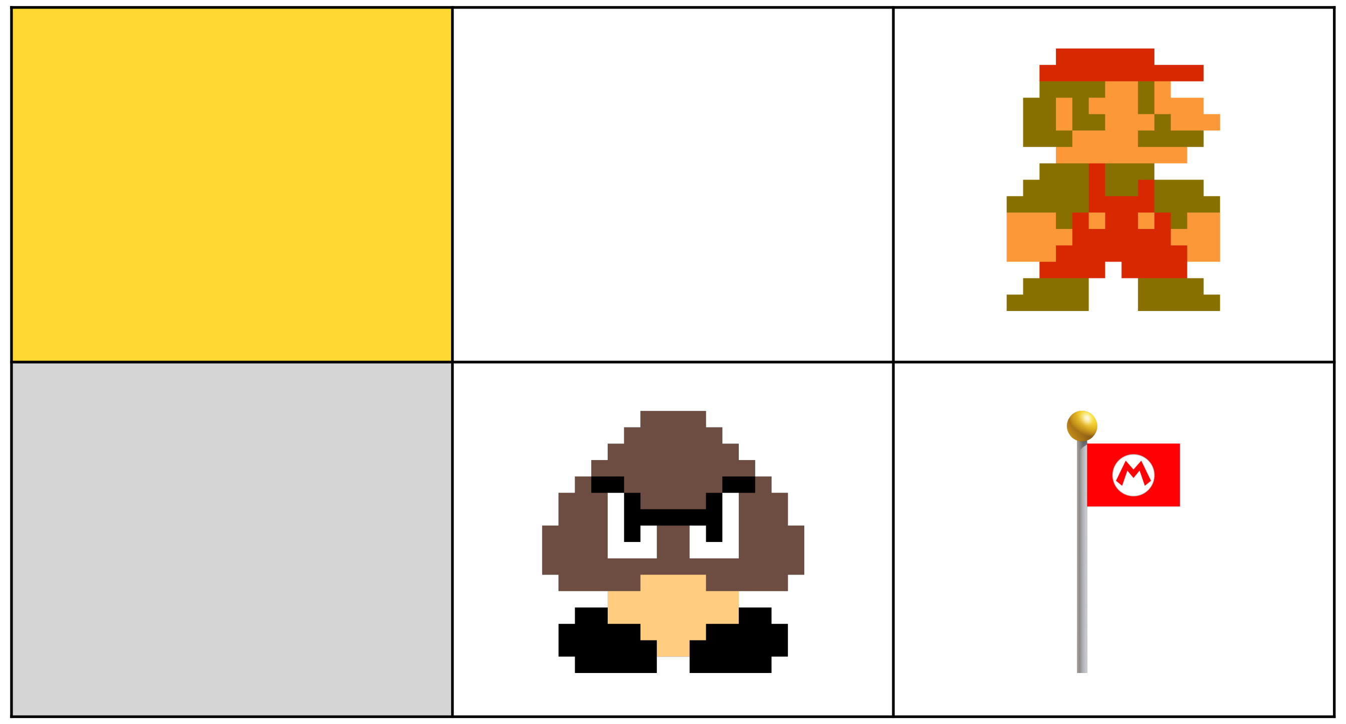
Q-Learning example

$\alpha = 0.1$

$\gamma = 0.99$

	$\leftarrow$	$\rightarrow$	$\uparrow$	$\downarrow$
	0	0.2	0	0
	0	0	0	-1
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0

Take action A, observe R, S'



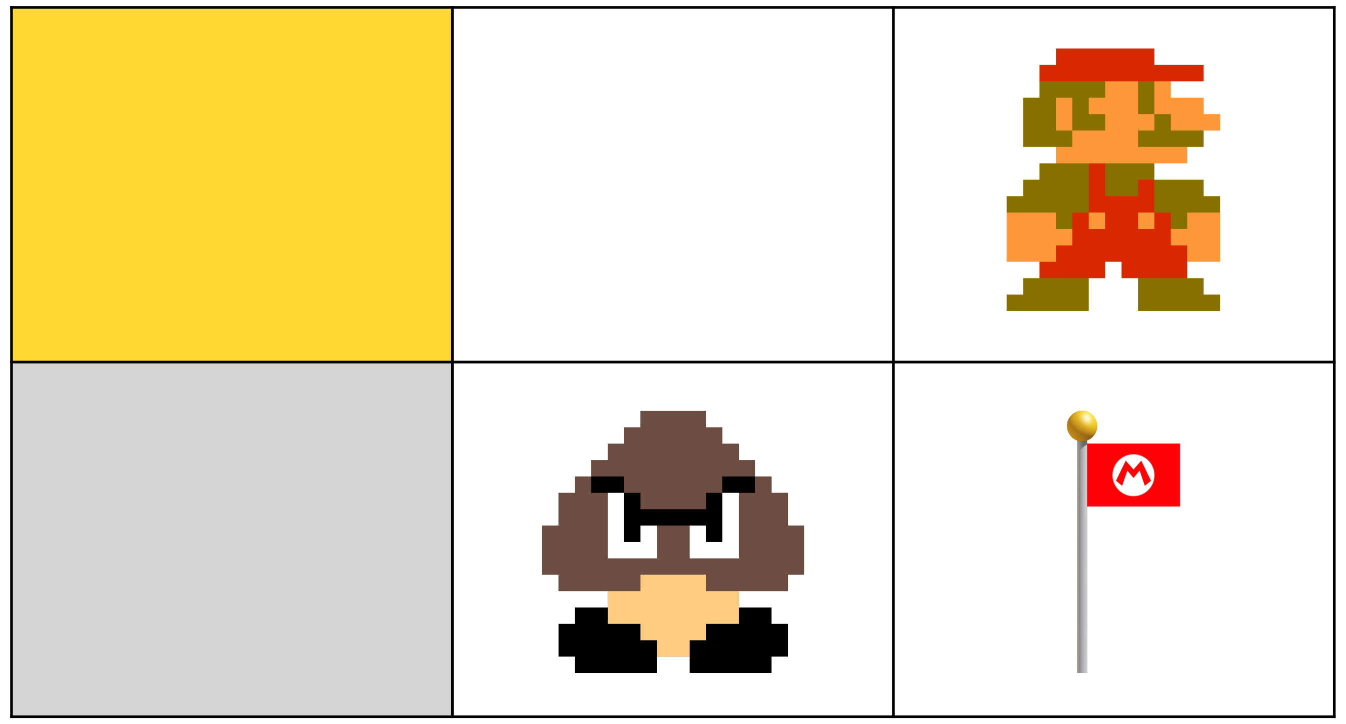
Q-Learning example

$\alpha = 0.1$

$\gamma = 0.99$

	←	→	↑	↓
	0	0.2	0	0
	0	0	0	-1
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0

Take action A, observe R, S'



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(, →) = 0 + 0.1*[0 + 0.99*0 - 0] = 0

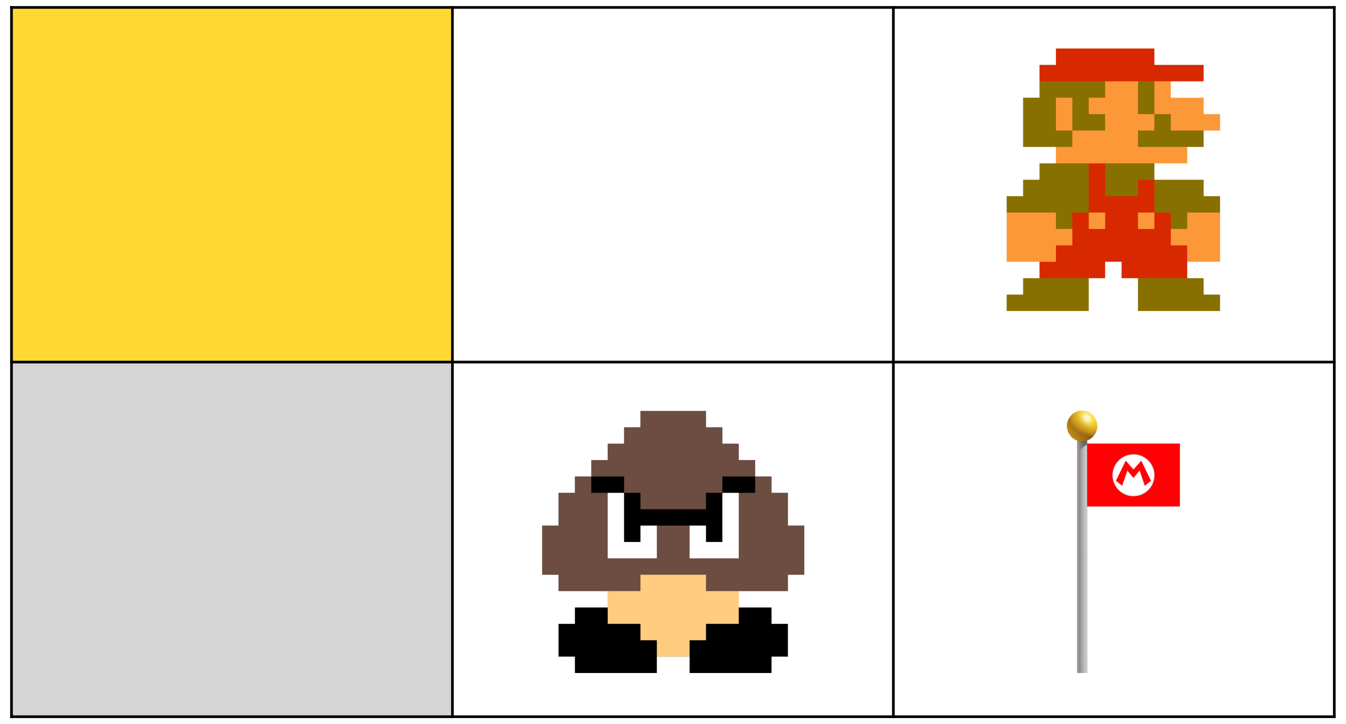
Q-Learning example

$\alpha = 0.1$

$\gamma = 0.99$

	$\leftarrow$	$\rightarrow$	$\uparrow$	$\downarrow$
	0	0.2	0	0
	0	0	0	-1
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0

Take action A, observe R, S'



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( , $\rightarrow$) = 0 + 0.1*[0 + 0.99*0 - 0] = 0

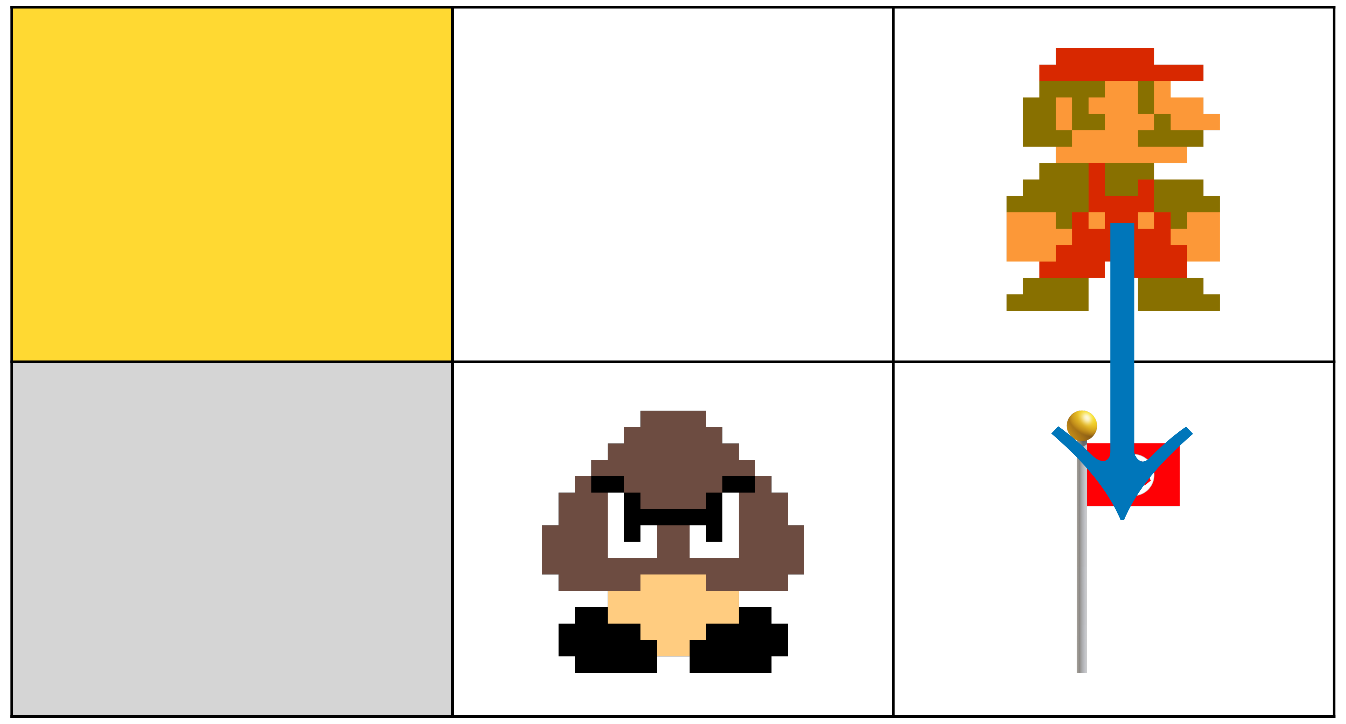
Q-Learning example

$\alpha = 0.1$

$\gamma = 0.99$

	$\leftarrow$	$\rightarrow$	$\uparrow$	$\downarrow$
	0	0.2	0	0
	0	0	0	-1
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0

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



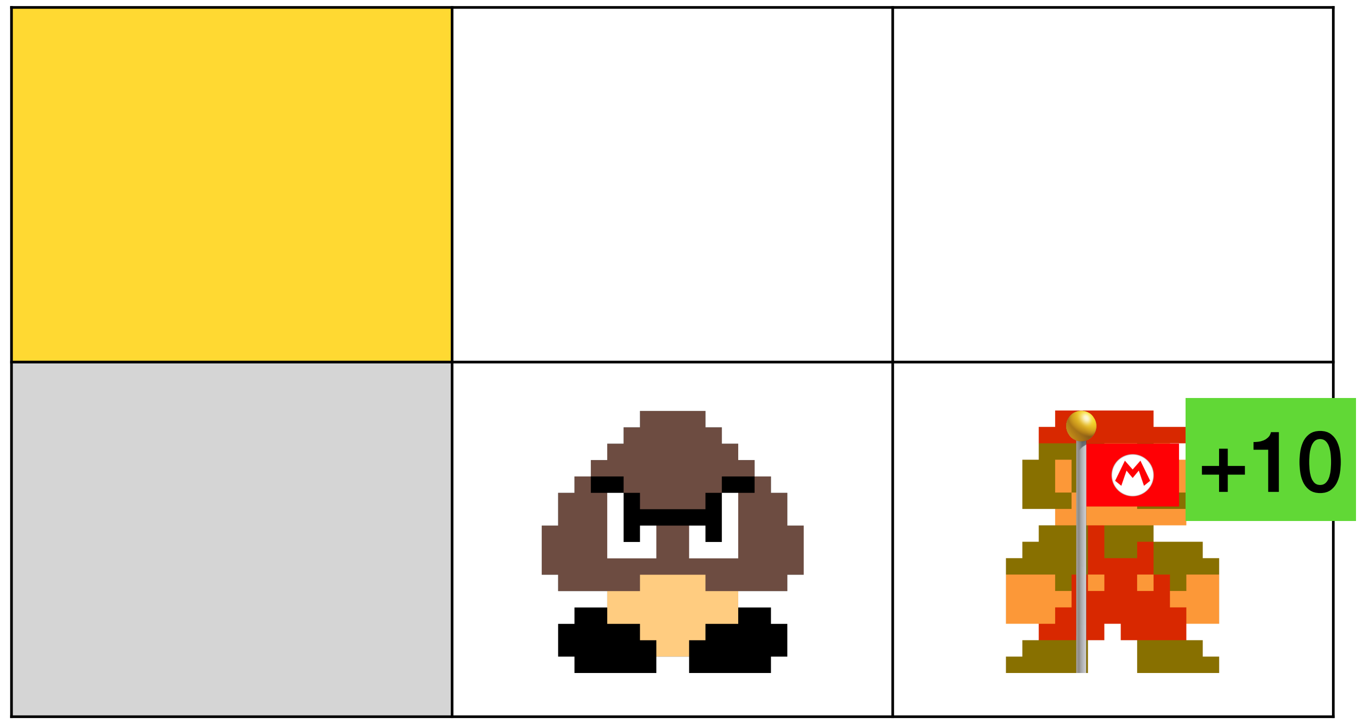
Q-Learning example

$\alpha = 0.1$

$\gamma = 0.99$

	←	→	↑	↓
	0	0.2	0	0
	0	0	0	-1
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0

Take action A, observe R, S'



Q-Learning example

$\alpha = 0.1$

$\gamma = 0.99$

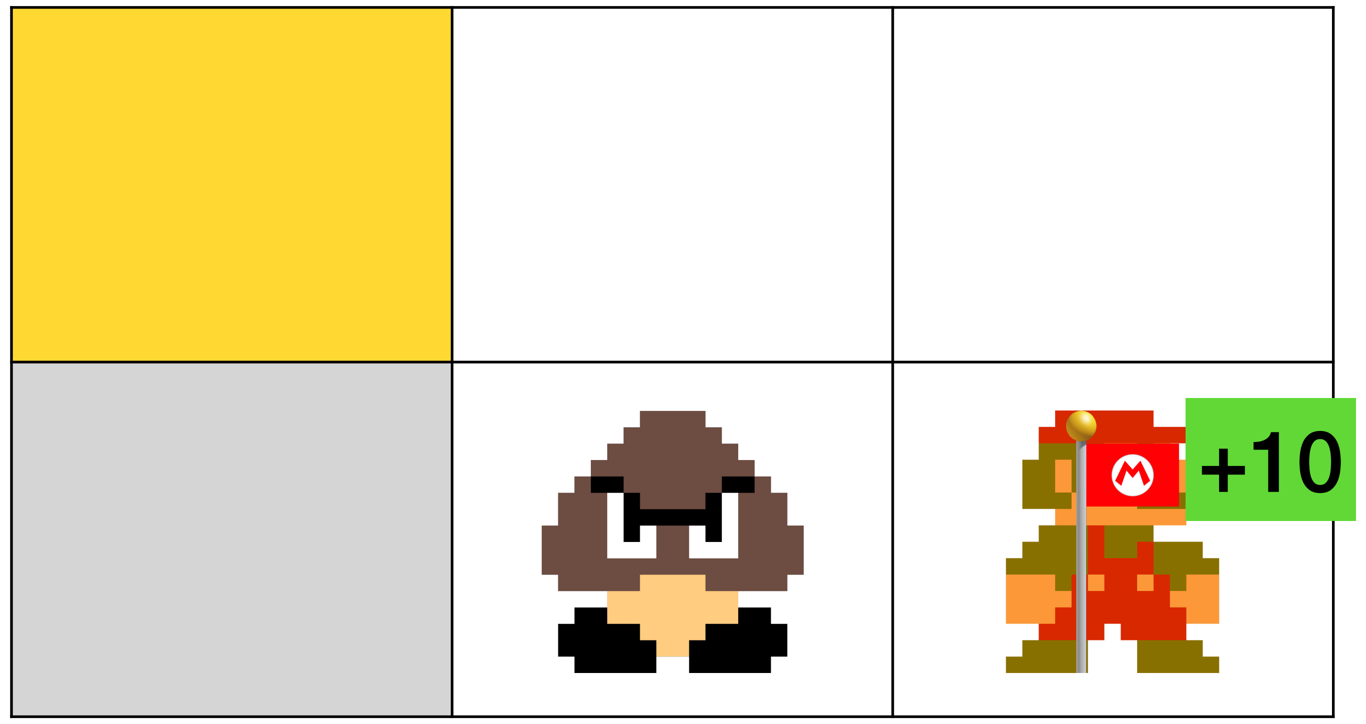
	←	→	↑	↓
	0	0.2	0	0
	0	0	0	-1
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0

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(, ↓) = 0 + 0.1*[10 + 0.99*0 - 0] = 1

Take action A, observe R, S'



Q-Learning example

$\alpha = 0.1$

$\gamma = 0.99$

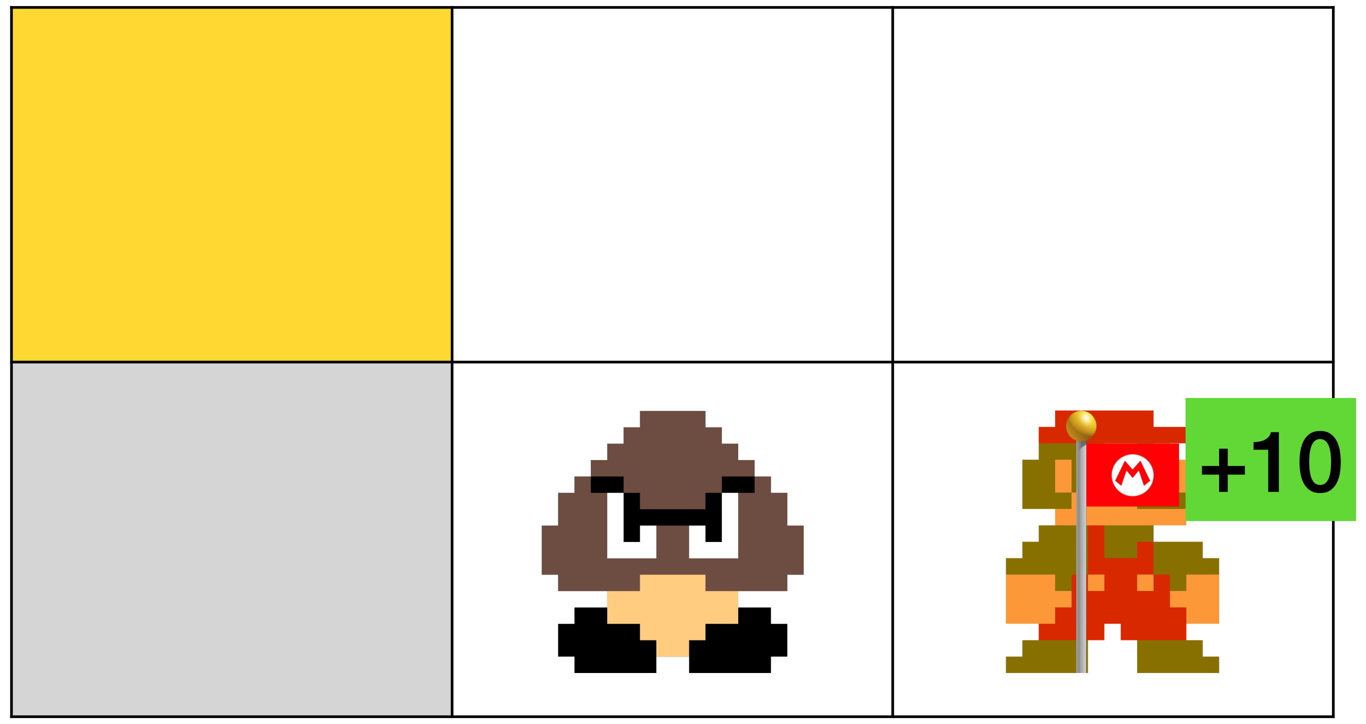
	←	→	↑	↓
	0	0.2	0	0
	0	0	0	-1
	0	0	0	1
	0	0	0	0
	0	0	0	0
	0	0	0	0

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(, ↓) = 0 + 0.1*[10 + 0.99*0 - 0] = 1

Take action A, observe R, S'



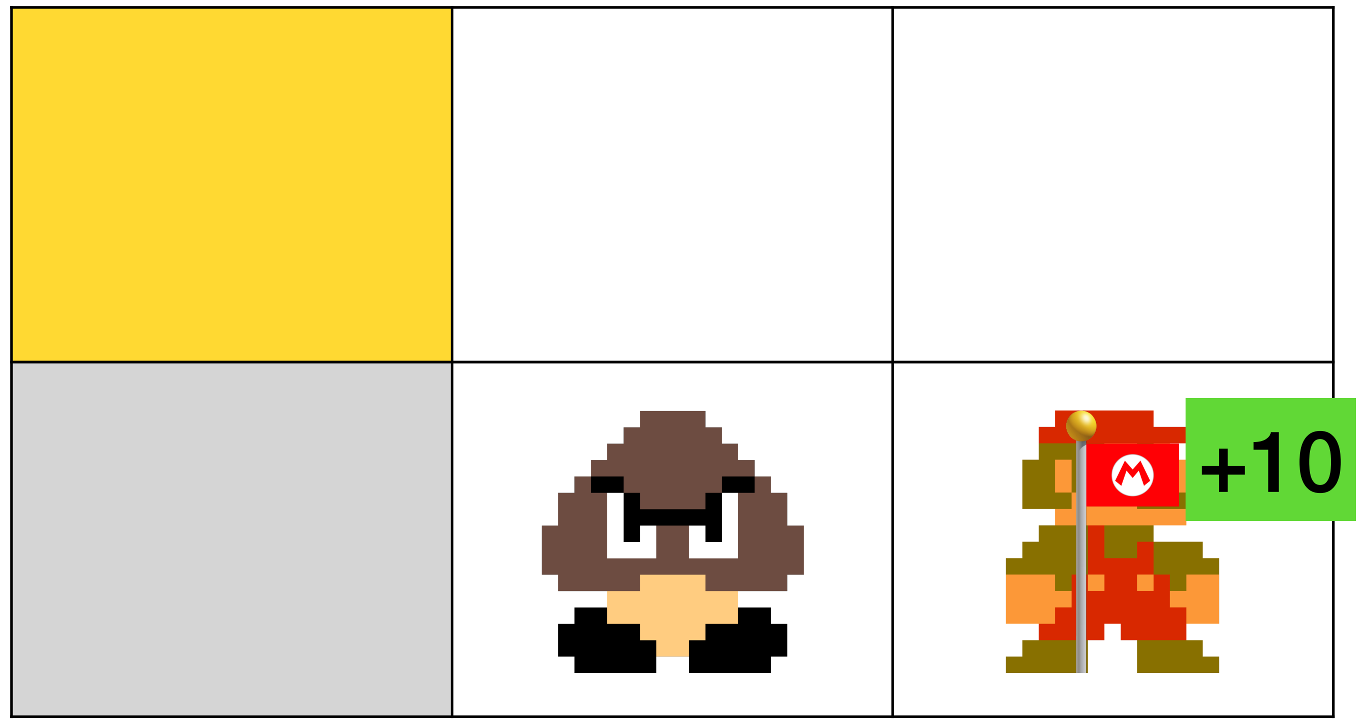
Q-Learning example

$\alpha = 0.1$

$\gamma = 0.99$

	←	→	↑	↓
	0	0.2	0	0
	0	0	0	-1
	0	0	0	1
	0	0	0	0
	0	0	0	0
	0	0	0	0

Take action A, observe R, S'



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(, ↓) = 0 + 0.1*[10 + 0.99*0 - 0] = 1

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 $1 - \varepsilon$

Exploit what the agent already knows

Why ε-greedy?

Exploration-exploitation tradeoff

Take greedy actions with probability $1 - \varepsilon$

Exploit what the agent already knows

Explore with probability ε

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

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

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				0
-1	1	0				0
0						0
0	10	1	1	1	1	Goal (10)

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				0
-1	1	0				0
0						0
0	10	1	1	1	1	Goal (10)

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				0
-1	1	0				0
0						0
0	10	1	1	1	1	Goal (10)

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				0
-1	1	0				0
0						0
0	10	1	1	1	1	Goal (10)

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				0
-1	1	0				0
0						0
0	10	1	1	1	1	Goal (10)

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				0
-1	1	0				0
0						0
0	10	1	1	1	1	Goal (10)

We end up with a suboptimal solution

G = 10

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				0
-1	1	0				0
0						0
0	10	1	1	1	1	Goal (10)

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				0
-1	1	0				0
0						0
0	10	1	1	1	1	Goal (10)

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				0
-1	1	0				0
0						0
0	10	1	1	1	1	Goal (10)

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				0
-1	1	0				0
0						0
0	10	1	1	1	1	Goal (10)

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				0
-1	1	0				0
0						0
0	10	1	1	1	1	Goal (10)

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				0
-1	1	0				0
0						0
0	10	1	1	1	1	Goal (10)

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				0
-1	1	0				0
0						0
0	10	1	1	1	1	Goal (10)

We might discover more valuable states

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…

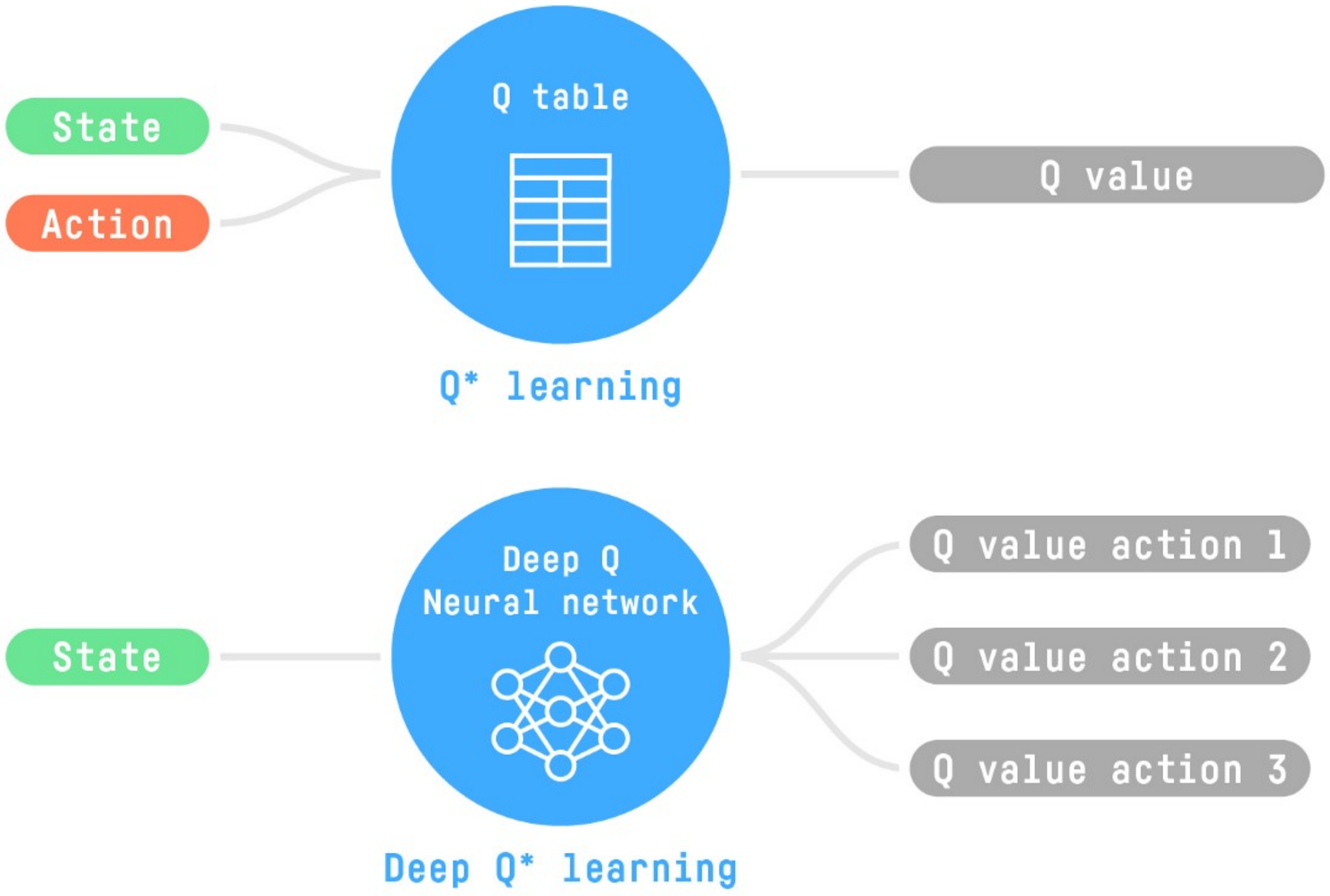
Start	0	0	0	0	0	0
-1		0				0
-1	1	0				0
0						0
0	10	1	1	1	1	Goal (10)

We might discover more valuable states

That lead to higher returns

G = 22

Deep Q-Learning


State
Action
Q table
Q value
Q* learning
State
Deep Q
Neural network
Q value action 1
Q value action 2
Q value action 3
Deep Q* learning

Deep Q-Learning

Deep Q-Learning

Q-Learning works well for small state spaces

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]

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

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

Deep Q-Learning

Q-Learning works well for small state spaces

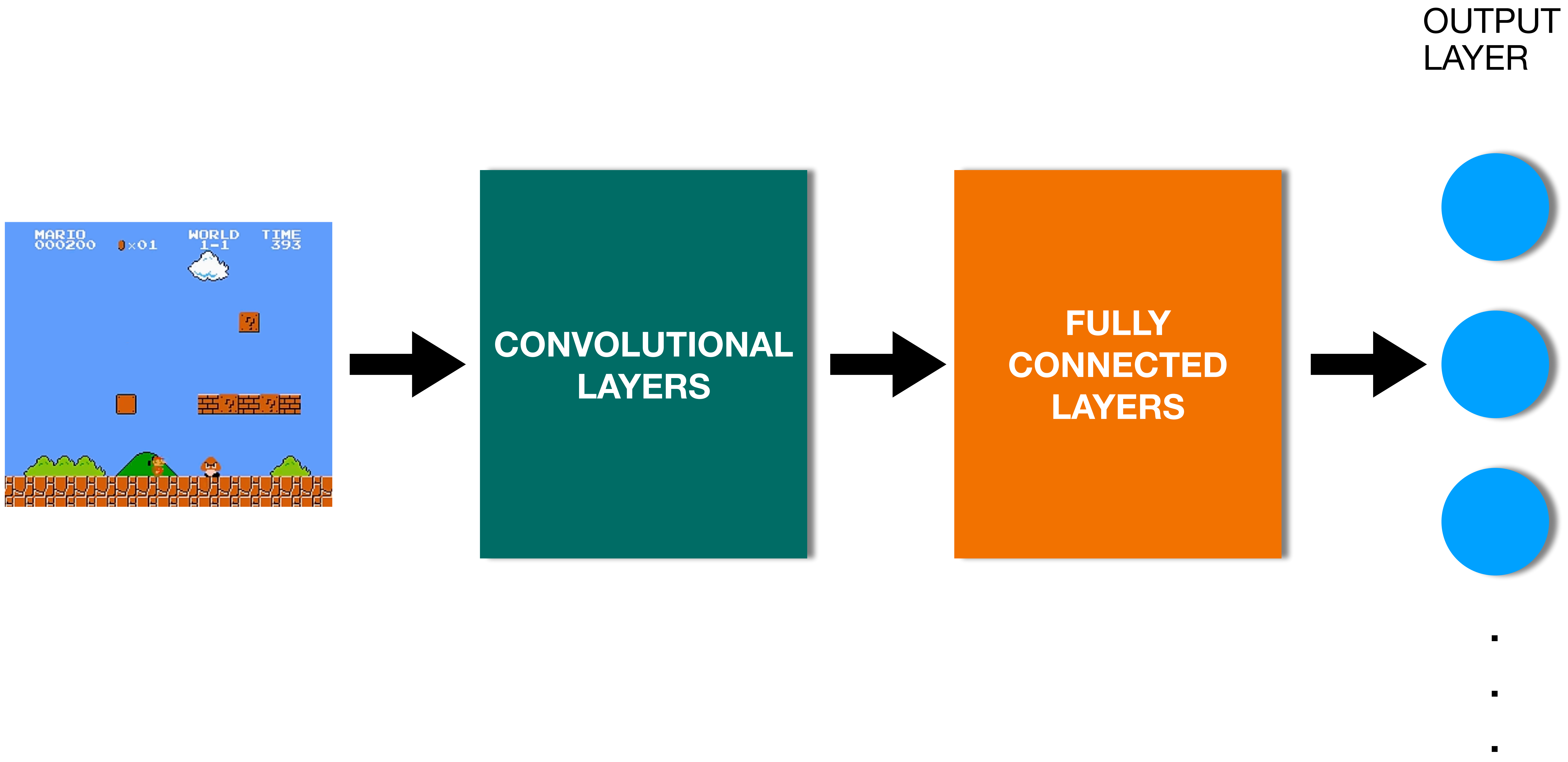
Atari games, however, have observation space of shape (210, 160, 3), values in [0, 255]

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

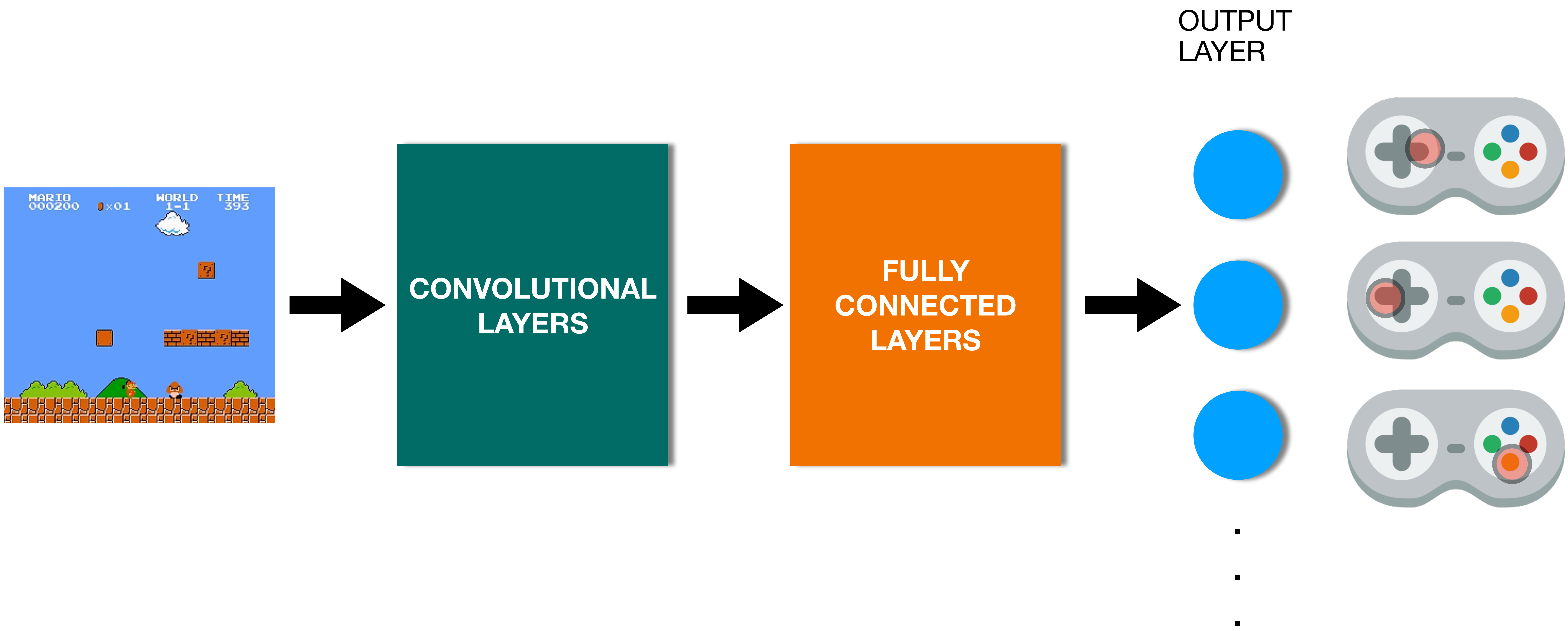
Approximate Q-values using a neural network

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

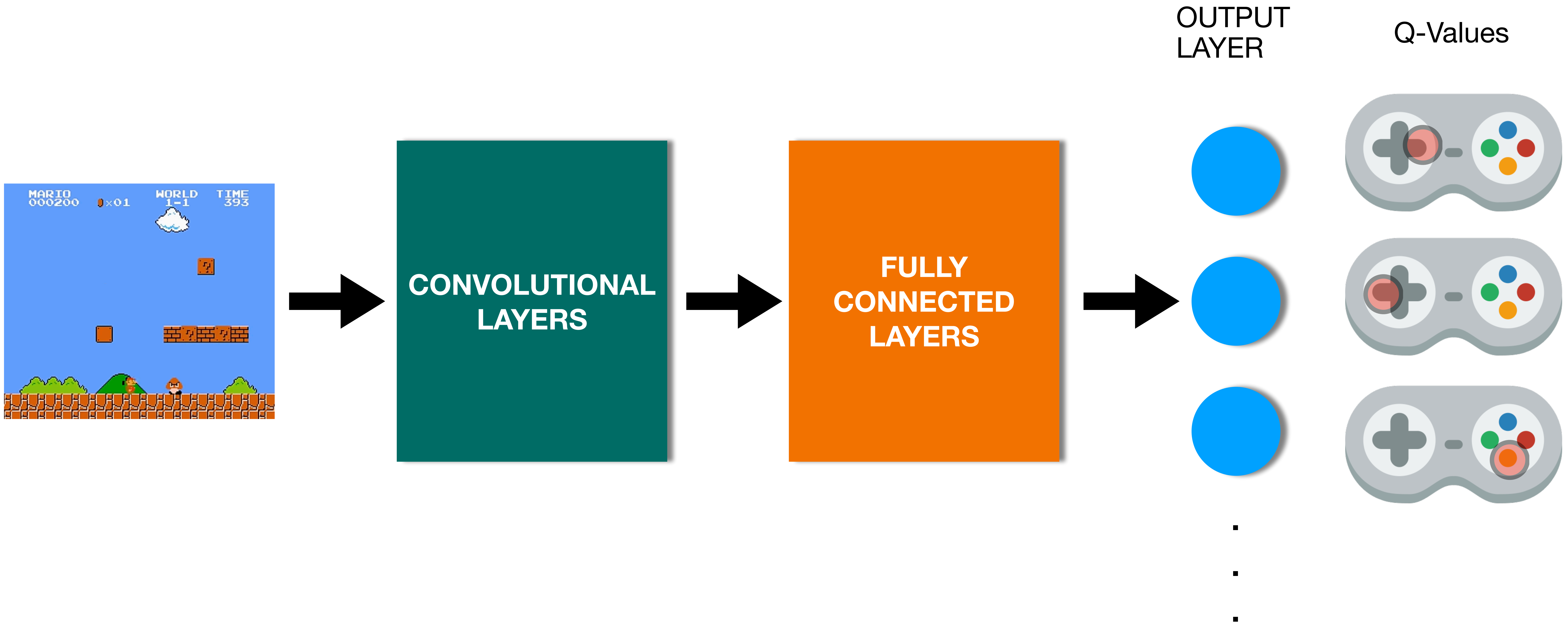
Deep Q-Learning



Deep Q-Learning



Deep Q-Learning



Deep Q-Learning algorithm

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

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

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

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

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

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

End For

End For

Deep Q-Learning algorithm

ϕ represents the NN

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 $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$
    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 $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^-\right) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2$ with respect to the network parameters θ
    Every C steps reset $\hat{Q} = Q$
  End For
End For

Deep Q-Learning algorithm

ϕ represents the NN

For episode $= 1, M$ **do**
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  For $t = 1$,T **do**
    $\boxed{\text{With probability } \varepsilon \text{ select a random action } a_t \text{ otherwise select } a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)}$
    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 $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^-\right) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2$ with respect to the network parameters θ
    Every C steps reset $\hat{Q} = Q$
  End For
End For

Deep Q-Learning algorithm

ϕ represents the NN

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

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 $\left(\phi_j, a_j, r_j, \phi_{j+1}\right)$ from D

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

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

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

End For

End For

Deep Q-Learning algorithm

ϕ represents the NN

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

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 $\left(\phi_j, a_j, r_j, \phi_{j+1}\right)$ from D

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

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

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

End For

End For

Deep Q-Learning algorithm

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 $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$
    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 $\left(\phi_t, a_t, r_t, \phi_{t+1}\right)$ in D
    Sample random minibatch of transitions $\left(\phi_j, a_j, r_j, \phi_{j+1}\right)$ from D
    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^-\right) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2$ with respect to the network parameters θ
    Every C steps reset $\hat{Q} = Q$
  End For
End For

ϕ represents the NN

Sampling (interaction with the environment)

Deep Q-Learning algorithm

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

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

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

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

End For

End For

ϕ represents the NN

Sampling (interaction with the environment)

Training

Deep Q-Learning algorithm

Tabular Q-Learning

Deep Q-Learning

Deep Q-Learning algorithm

Tabular Q-Learning

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

Deep Q-Learning

Deep Q-Learning algorithm

Tabular Q-Learning

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

Deep Q-Learning

$$y_j = \quad r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^-\right)$$

$$\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2$$

Deep Q-Learning algorithm

Tabular Q-Learning

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

New
Q-value

Deep Q-Learning

$$y_j = r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^-\right)$$

$$\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2$$

Deep Q-Learning algorithm

Tabular Q-Learning

$$\underbrace{Q(S_t, A_t)}_{\text{New Q-value}} \leftarrow \underbrace{Q(S_t, A_t)}_{\text{Former Q-value}} + \alpha(R_{t+1} + \gamma\, max_{a'}\, Q(S_{t+1}, a') - Q(S_t, A_t))$$

Deep Q-Learning

$$y_j = r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^{-}\right)$$

$$\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2$$

Deep Q-Learning algorithm

Tabular Q-Learning

$$\underbrace{Q(S_t, A_t)}_{\text{New Q-value}} \leftarrow \underbrace{Q(S_t, A_t)}_{\text{Former Q-value}} + \underbrace{\alpha}_{\text{step size}}(R_{t+1} + \gamma \, max_{a'} \, Q(S_{t+1}, a') - Q(S_t, A_t))$$

Deep Q-Learning

$$y_j = r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^-\right)$$

$$\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2$$

Deep Q-Learning algorithm

Tabular Q-Learning

$$\underbrace{Q(S_t, A_t)}_{\text{New Q-value}} \leftarrow \underbrace{Q(S_t, A_t)}_{\text{Former Q-value}} + \underbrace{\alpha}_{\text{step size}}(\underbrace{R_{t+1}}_{\text{Reward}} + \gamma \, max_{a'} \, Q(S_{t+1}, a') - Q(S_t, A_t))$$

Deep Q-Learning

$$y_j = \quad r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^-\right)$$

$$\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2$$

Deep Q-Learning algorithm

Tabular Q-Learning

$$\underbrace{Q(S_t, A_t)}_{\text{New Q-value}} \leftarrow \underbrace{Q(S_t, A_t)}_{\text{Former Q-value}} + \underbrace{\alpha}_{\text{step size}}(\underbrace{R_{t+1}}_{\text{Reward}} + \underbrace{\gamma\, max_{a'}\, Q(S_{t+1}, a')}_{\text{Discounted Estimate (optimal Q-value of next state)}} - Q(S_t, A_t))$$

Deep Q-Learning

$$y_j = \quad r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^-\right)$$

$$\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2$$

Deep Q-Learning algorithm

Tabular Q-Learning

$$\underbrace{Q(S_t, A_t)}_{\text{New Q-value}} \leftarrow \underbrace{Q(S_t, A_t)}_{\text{Former Q-value}} + \underbrace{\alpha}_{\text{step size}}(\underbrace{R_{t+1}}_{\text{Reward}} + \underbrace{\gamma\, max_{a'}\, Q(S_{t+1}, a')}_{\text{Discounted Estimate (optimal Q-value of next state)}} - \underbrace{Q(S_t, A_t)}_{\text{Former Q-value}})$$

Deep Q-Learning

$$y_j = r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^-\right)$$

$$\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2$$

Deep Q-Learning algorithm

Tabular Q-Learning

$$\underbrace{Q(S_t, A_t)}_{\text{New Q-value}} \leftarrow \underbrace{Q(S_t, A_t)}_{\text{Former Q-value}} + \underbrace{\alpha}_{\text{step size}}(\underbrace{\underbrace{R_{t+1}}_{\text{Reward}} + \underbrace{\gamma\ max_{a'}\ Q(S_{t+1}, a')}_{\text{Discounted Estimate (optimal Q-value of next state)}}}_{\text{TD-target}} - \underbrace{Q(S_t, A_t)}_{\text{Former Q-value}})$$

Deep Q-Learning

$$y_j = r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^-\right)$$

$$\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2$$

Deep Q-Learning algorithm

Tabular Q-Learning

$$\underbrace{Q(S_t, A_t)}_{\text{New Q-value}} \leftarrow \underbrace{Q(S_t, A_t)}_{\text{Former Q-value}} + \underbrace{\alpha}_{\text{step size}} (\underbrace{\underbrace{\underbrace{R_{t+1}}_{\text{Reward}} + \underbrace{\gamma \, max_{a'} \, Q(S_{t+1}, a')}_{\text{Discounted Estimate (optimal Q-value of next state)}}}_{\text{TD-target}} - \underbrace{Q(S_t, A_t)}_{\text{Former Q-value}}}_{\text{TD-error}})$$

Deep Q-Learning

$$y_j = \quad r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^{-}\right)$$

$$\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2$$

Deep Q-Learning algorithm

Tabular Q-Learning

$$\underbrace{Q(S_t, A_t)}_{\text{New Q-value}} \leftarrow \underbrace{Q(S_t, A_t)}_{\text{Former Q-value}} + \underbrace{\alpha}_{\text{step size}} \overbrace{\Big(\underbrace{\underbrace{R_{t+1}}_{\text{Reward}} + \underbrace{\gamma\, max_{a'}\, Q(S_{t+1}, a')}_{\text{Discounted Estimate (optimal Q-value of next state)}}}_{\text{TD-target}} - \underbrace{Q(S_t, A_t)}_{\text{Former Q-value}}\Big)}^{}$$

TD-error

Deep Q-Learning

Q-Target

$$y_j = r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^-\right)$$

$$\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2$$

Deep Q-Learning algorithm

Tabular Q-Learning

$$\underbrace{Q(S_t, A_t)}_{\text{New Q-value}} \leftarrow \underbrace{Q(S_t, A_t)}_{\text{Former Q-value}} + \underbrace{\alpha}_{\text{step size}}\Big(\underbrace{\underbrace{\underbrace{R_{t+1}}_{\text{Reward}} + \underbrace{\gamma\, max_{a'}\, Q(S_{t+1}, a')}_{\text{Discounted Estimate (optimal Q-value of next state)}}}_{\text{TD-target}} - \underbrace{Q(S_t, A_t)}_{\text{Former Q-value}}}_{\text{TD-error}}\Big)$$

Deep Q-Learning

Q-Target

$$y_j = r_j + \gamma\, \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^-\right)$$

Q-Loss

$$\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2$$

Deep Q-Learning algorithm

Tabular Q-Learning

$$\underbrace{Q(S_t, A_t)}_{\text{New Q-value}} \leftarrow \underbrace{Q(S_t, A_t)}_{\text{Former Q-value}} + \underbrace{\alpha}_{\text{step size}}(\underbrace{\underbrace{\underbrace{R_{t+1}}_{\text{Reward}} + \underbrace{\gamma\ max_{a'}\ Q(S_{t+1}, a')}_{\text{Discounted Estimate (optimal Q-value of next state)}}}_{\text{TD-target}} - \underbrace{Q(S_t, A_t)}_{\text{Former Q-value}}}_{\text{TD-error}})$$

Deep Q-Learning

Q-Target

$$y_j = \underbrace{r_j}_{\text{Reward}} + \gamma\ \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^-\right)$$

Q-Loss

$$\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2$$

Deep Q-Learning algorithm

Tabular Q-Learning

$$\underbrace{Q(S_t, A_t)}_{\text{New Q-value}} \leftarrow \underbrace{Q(S_t, A_t)}_{\text{Former Q-value}} + \underbrace{\alpha}_{\text{step size}} \Big(\underbrace{\underbrace{\underbrace{R_{t+1}}_{\text{Reward}} + \underbrace{\gamma \ max_{a'} \ Q(S_{t+1}, a')}_{\text{Discounted Estimate (optimal Q-value of next state)}}}_{\text{TD-target}} - \underbrace{Q(S_t, A_t)}_{\text{Former Q-value}}}_{\text{TD-error}}\Big)$$

Deep Q-Learning

Q-Target

$$y_j = \underbrace{r_j}_{\text{Reward}} + \gamma \ \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^-\right)$$

Discounted Estimate (optimal Q-value of next state)

Q-Loss

$$\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2$$

Deep Q-Learning algorithm

Tabular Q-Learning

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

New Q-value

Former Q-value

step size

Reward

Discounted Estimate (optimal Q-value of next state)

Former Q-value

TD-target

TD-error

Deep Q-Learning

Q-Target

$$y_j = r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^-\right)$$

Reward

Discounted Estimate (optimal Q-value of next state)

TD-target

Q-Loss

$$\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2$$

Deep Q-Learning algorithm

Tabular Q-Learning

$$\underbrace{Q(S_t, A_t)}_{\text{New Q-value}} \leftarrow \underbrace{Q(S_t, A_t)}_{\text{Former Q-value}} + \underbrace{\alpha}_{\text{step size}}(\underbrace{R_{t+1}}_{\text{Reward}} + \underbrace{\gamma\ max_{a'}\ Q(S_{t+1}, a')}_{\text{Discounted Estimate (optimal Q-value of next state)}} - \underbrace{Q(S_t, A_t)}_{\text{Former Q-value}})$$

TD-target: $R_{t+1} + \gamma\ max_{a'}\ Q(S_{t+1}, a')$

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

Deep Q-Learning

Q-Target

$$y_j = \underbrace{\underbrace{r_j}_{\text{Reward}} + \underbrace{\gamma\ \max_{a'}\ \hat{Q}\left(\phi_{j+1}, a'; \theta^-\right)}_{\text{Discounted Estimate (optimal Q-value of next state)}}}_{\text{TD-target}}$$

Q-Loss

$$\underbrace{\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2}_{\text{TD-error}}$$

To the notebook!