Deep Reinforcement Learning

Alexander Amini

MIT Introduction to Deep Learning

January 10, 2024

Learning in Dynamic Environments

Reinforcement Learning: Robots, Games, the World

Robotics

Game Play and Strategy

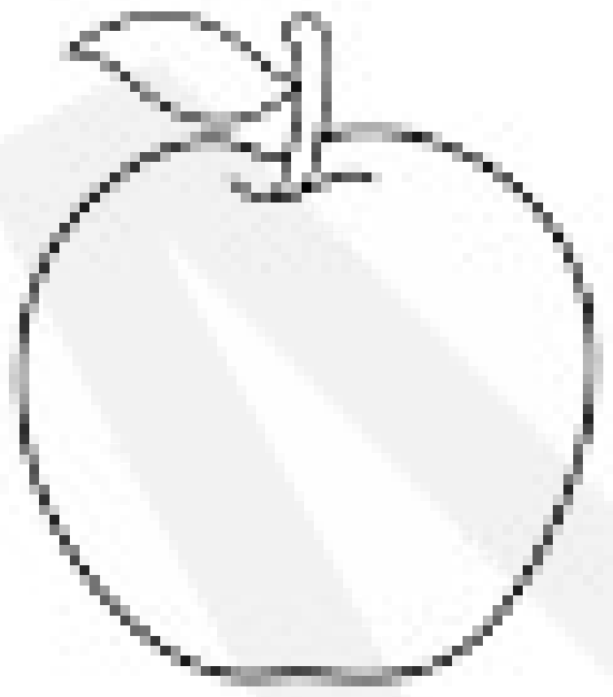
Classes of Learning Problems

Supervised Learning

Data: (x, y)
x is data, y is label

Goal: Learn function to map
$$x \rightarrow y$$

Apple example:



This thing is an apple.

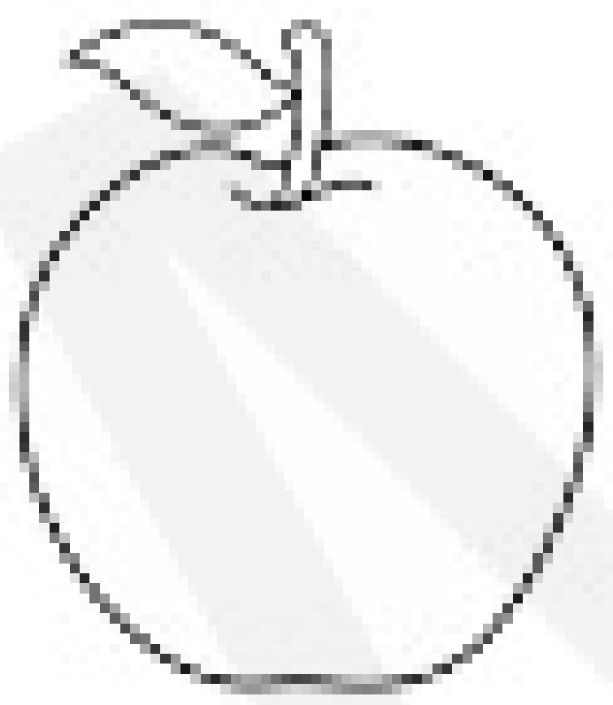
Classes of Learning Problems

Supervised Learning

Data: (x, y)
x is data, y is label

Goal: Learn function to map
$$x \rightarrow y$$

Apple example:



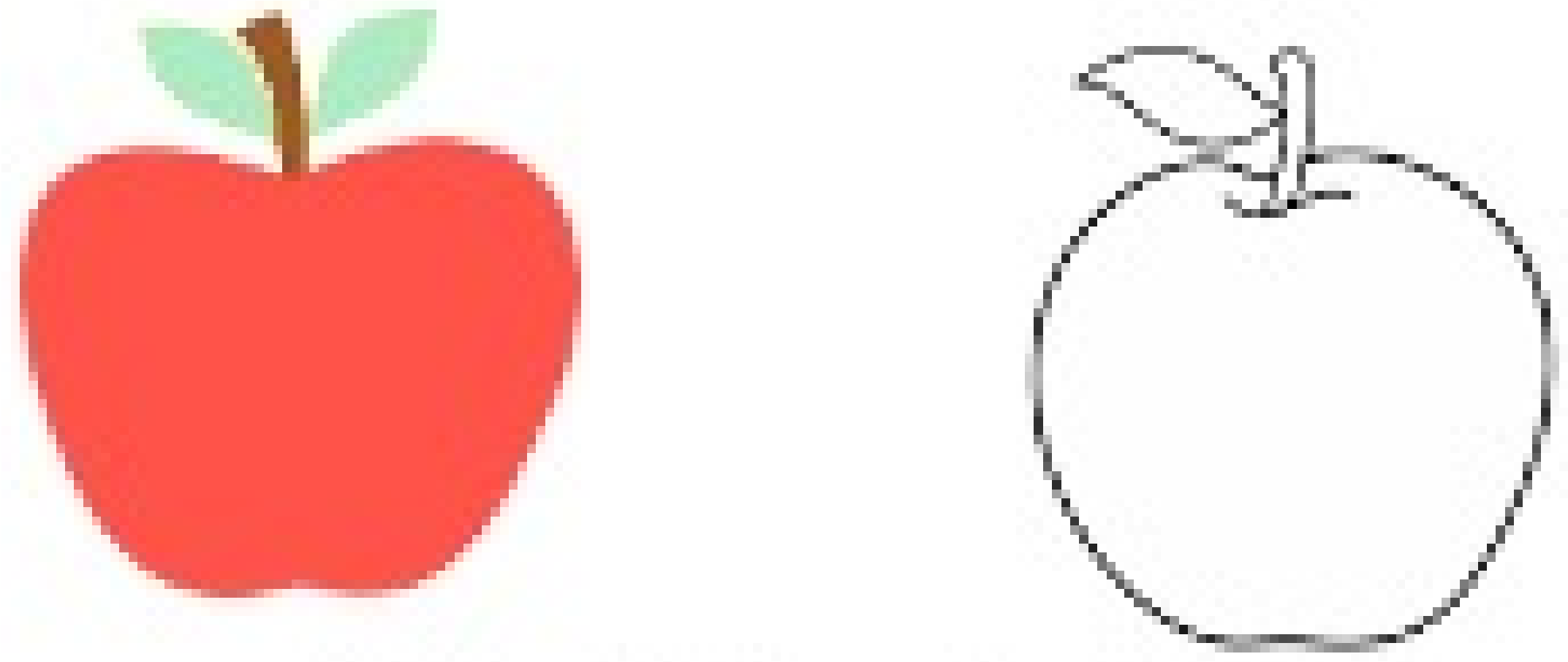
This thing is an apple.

Unsupervised Learning

Data: x
x is data, no labels!

Goal: Learn underlying structure

Apple example:



This thing is like the other thing.

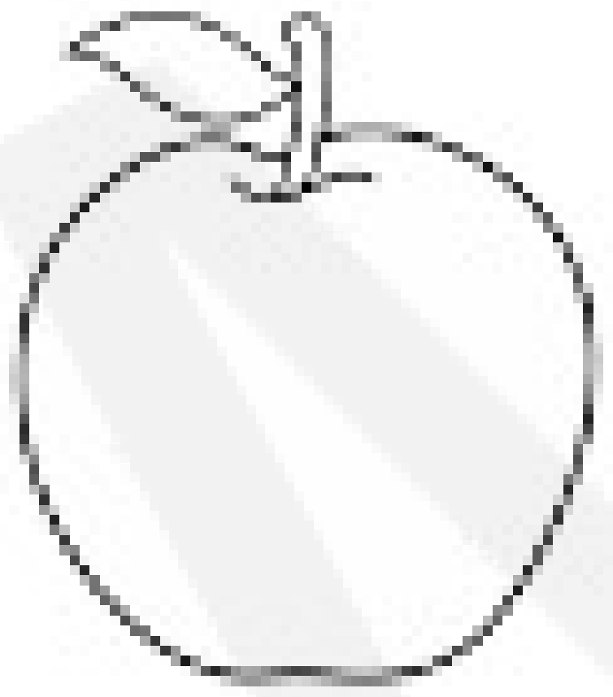
Classes of Learning Problems

Supervised Learning

Data: (x, y)
x is data, y is label

Goal: Learn function to map
$x \rightarrow y$

Apple example:



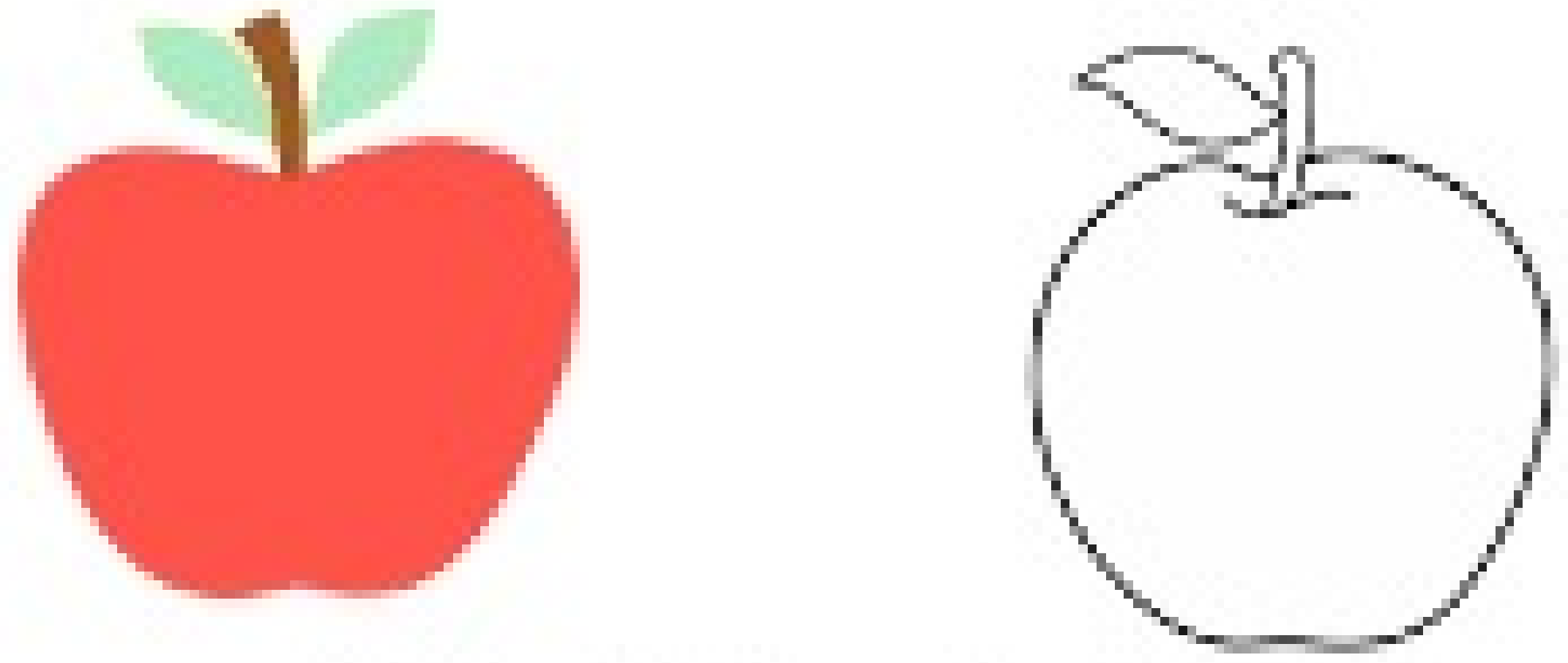
This thing is an apple.

Unsupervised Learning

Data: x
x is data, no labels!

Goal: Learn underlying structure

Apple example:



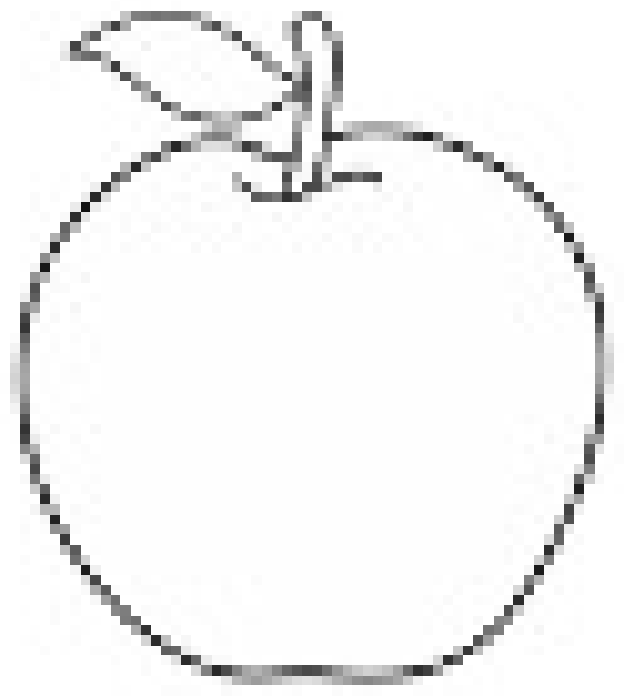
This thing is like
the other thing.

Reinforcement Learning

Data: state-action pairs

Goal: Maximize future rewards over many time steps

Apple example:



Eat this thing because it
will keep you alive.

Classes of Learning Problems

Supervised Learning

Data: (x, y)
x is data, y is label

Goal: Learn function to map
$x \to y$

Apple example:

This is an apple.

Unsupervised Learning

Data: x
x is data, no labels!

Goal: Learn underlying structure

Apple example:

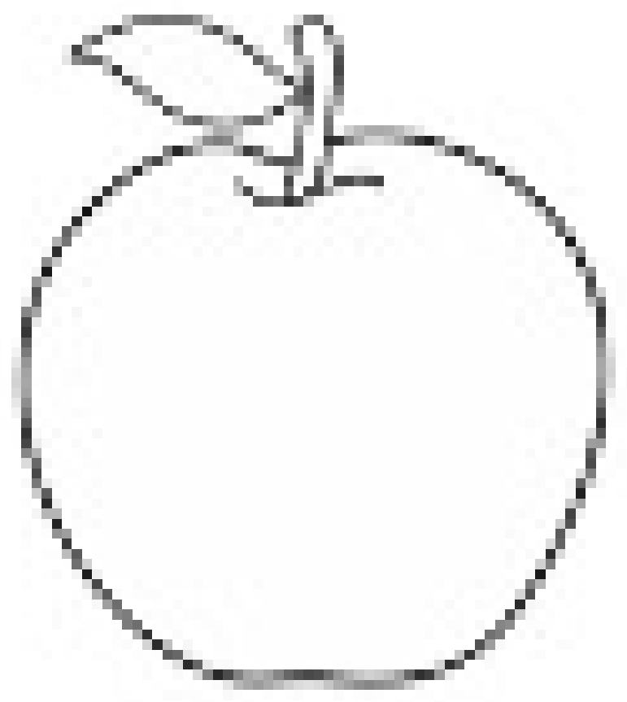
This thing is like the other thing.

Reinforcement Learning

Data: state-action pairs

Goal: Maximize future rewards over many time steps

Apple example:



Eat this thing because it will keep you alive.

RL: our focus today.

Reinforcement Learning (RL): Key Concepts

Agent: takes actions.

Reinforcement Learning (RL): Key Concepts

Environment: the world in which the agent exists and operates.

Reinforcement Learning (RL): Key Concepts

Action: a move the agent can make in the environment.

Action space A: the set of possible actions an agent can make in the environment

Reinforcement Learning (RL): Key Concepts

Observations: of the environment after taking actions.

Reinforcement Learning (RL): Key Concepts

State: a situation which the agent perceives.

Reinforcement Learning (RL): Key Concepts

Reward: feedback that measures the success or failure of the agent's action.

Reinforcement Learning (RL): Key Concepts

OBSERVATIONS

State changes: s_{t+1}

Reward: r_t

AGENT

ENVIRONMENT

Action: a_t

ACTIONS

Total Reward (Return)

$$R_t = \sum_{i=t}^{\infty} r_i$$

Reinforcement Learning (RL): Key Concepts

OBSERVATIONS

State changes: s_{t+1}

Reward: r_t

AGENT

ENVIRONMENT

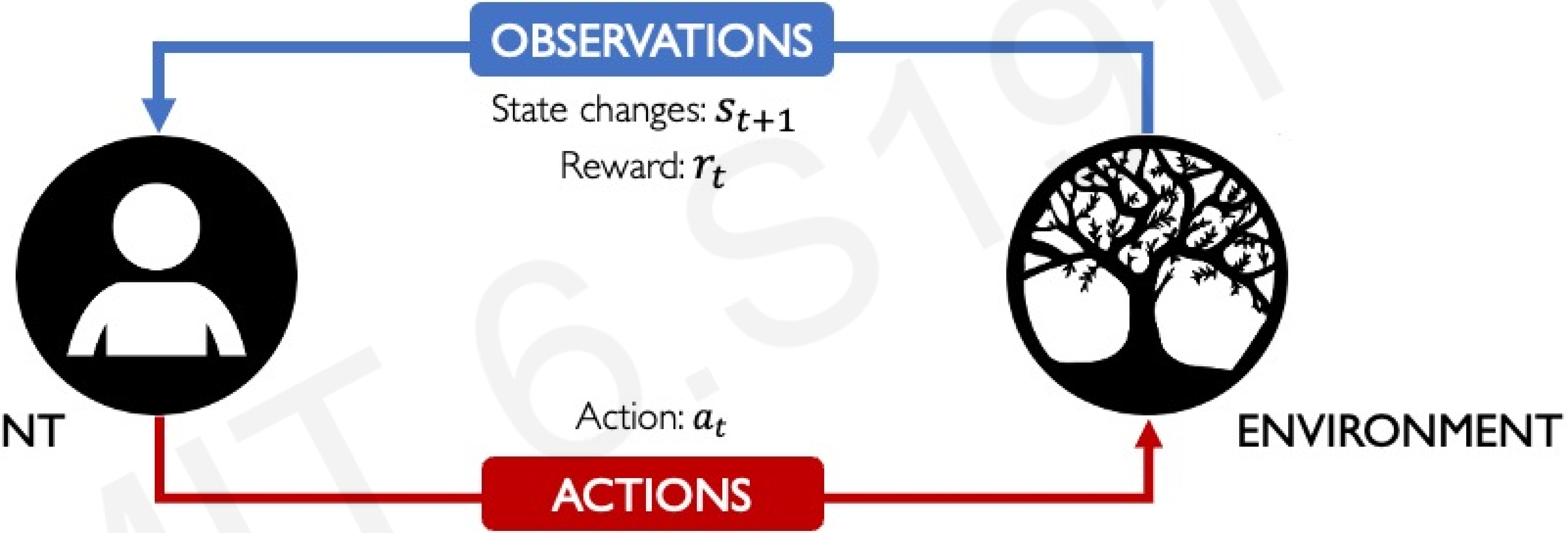
Action: a_t

ACTIONS

Total Reward (Return)

$$R_t = \sum_{i=t}^{\infty} r_i = r_t + r_{t+1} \ldots + r_{t+n} + \cdots$$

Reinforcement Learning (RL): Key Concepts



Discounted Total Reward (Return)

$$R_t = \sum_{i=t}^{\infty} \gamma^i r_i$$

Reinforcement Learning (RL): Key Concepts

OBSERVATIONS

State changes: s_{t+1}

Reward: r_t

AGENT

ENVIRONMENT

Action: a_t

ACTIONS

Discounted Total Reward (Return)

$$R_t = \sum_{i=t}^{\infty} \gamma^i r_i = \gamma^t r_t + \gamma^{t+1} r_{t+1} \ldots + \gamma^{t+n} r_{t+n} + \cdots$$

γ: discount factor; $0 < \gamma < 1$

Defining the Q-function

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots$$

Total reward, R_t, is the discounted sum of all rewards obtained from time t

$$Q(s_t, a_t) = \mathbb{E}[R_t | s_t, a_t]$$

The Q-function captures the **expected total future reward** an agent in state, s, can receive by executing a certain action, a

How to take actions given a Q-function?

$$Q(s_t, a_t) = \mathbb{E}[R_t | s_t, a_t]$$

(state, action)

Ultimately, the agent needs a **policy** $\boldsymbol{\pi(s)}$, to infer the **best action to take** at its state, s

Strategy: the policy should choose an action that maximizes future reward

$$\pi^*(s) = \underset{a}{\operatorname{argmax}}\, Q(s, a)$$

Deep Reinforcement Learning Algorithms

Value Learning

Find $Q(s, a)$

$$a = \underset{a}{\operatorname{argmax}}\, Q(s, a)$$

Policy Learning

Find $\pi(s)$

Sample $a \sim \pi(s)$

Deep Reinforcement Learning Algorithms

Value Learning

Find $Q(s,a)$

$$a = \underset{a}{\operatorname{argmax}}\, Q(s,a)$$

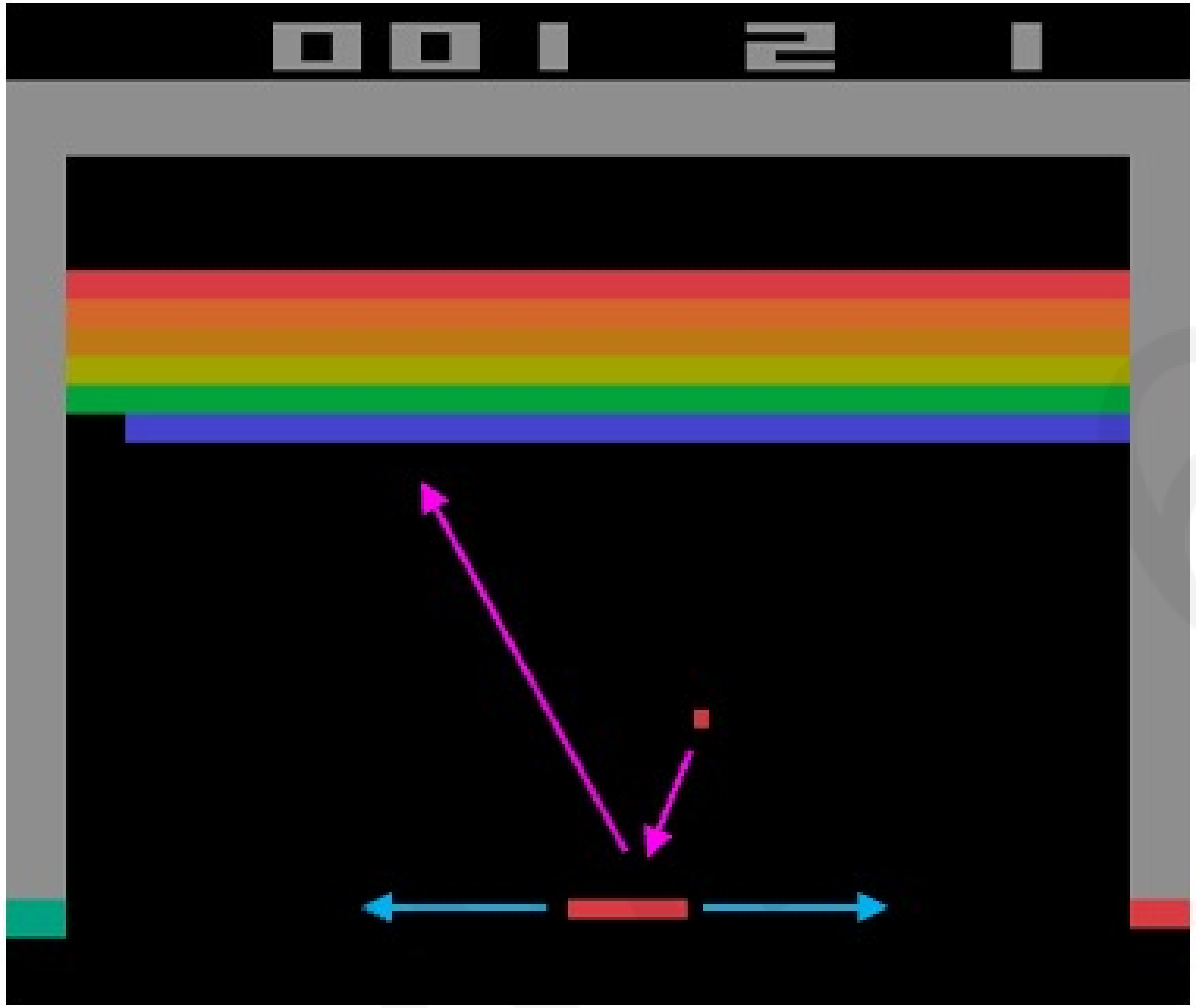
Policy Learning

Find $\pi(s)$

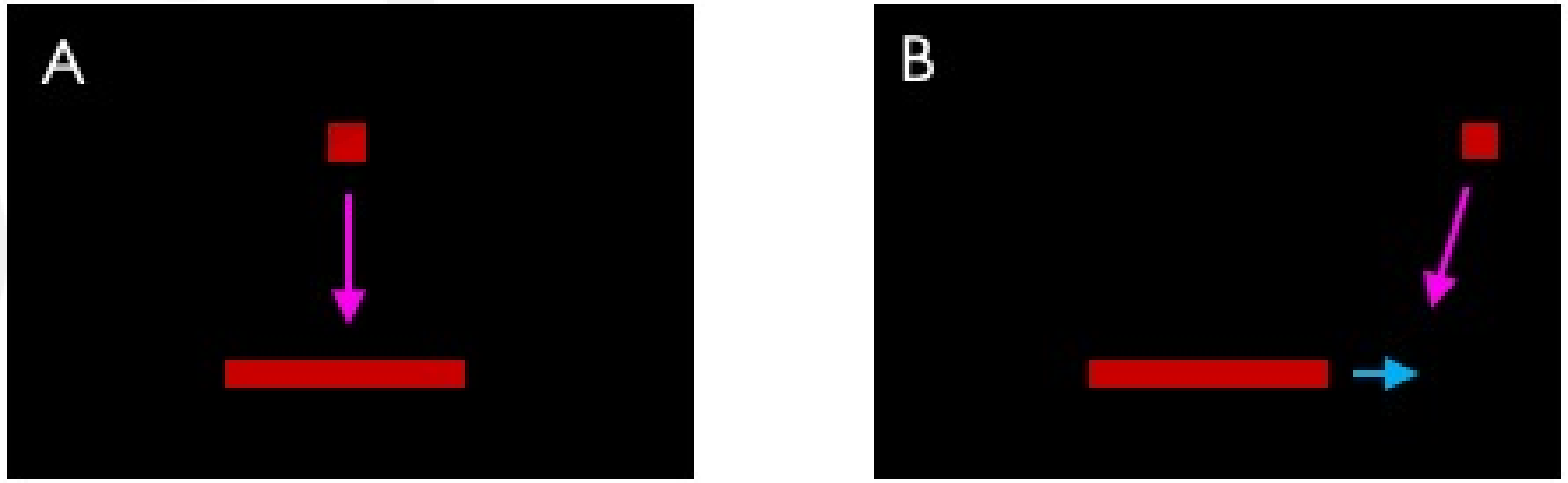
Sample $a \sim \pi(s)$

Digging deeper into the Q-function

Example: Atari Breakout



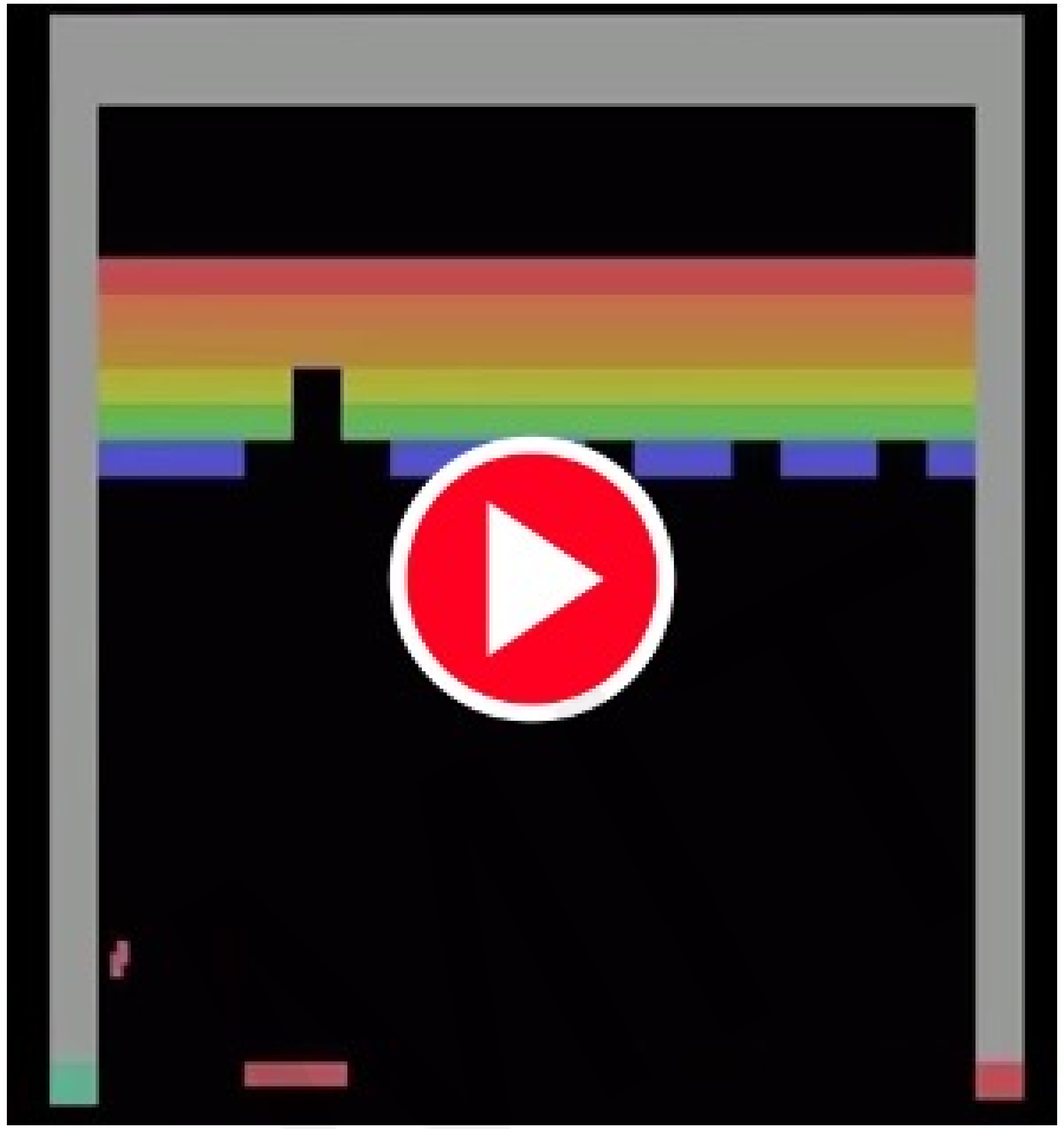
It can be very difficult for humans to accurately estimate Q-values



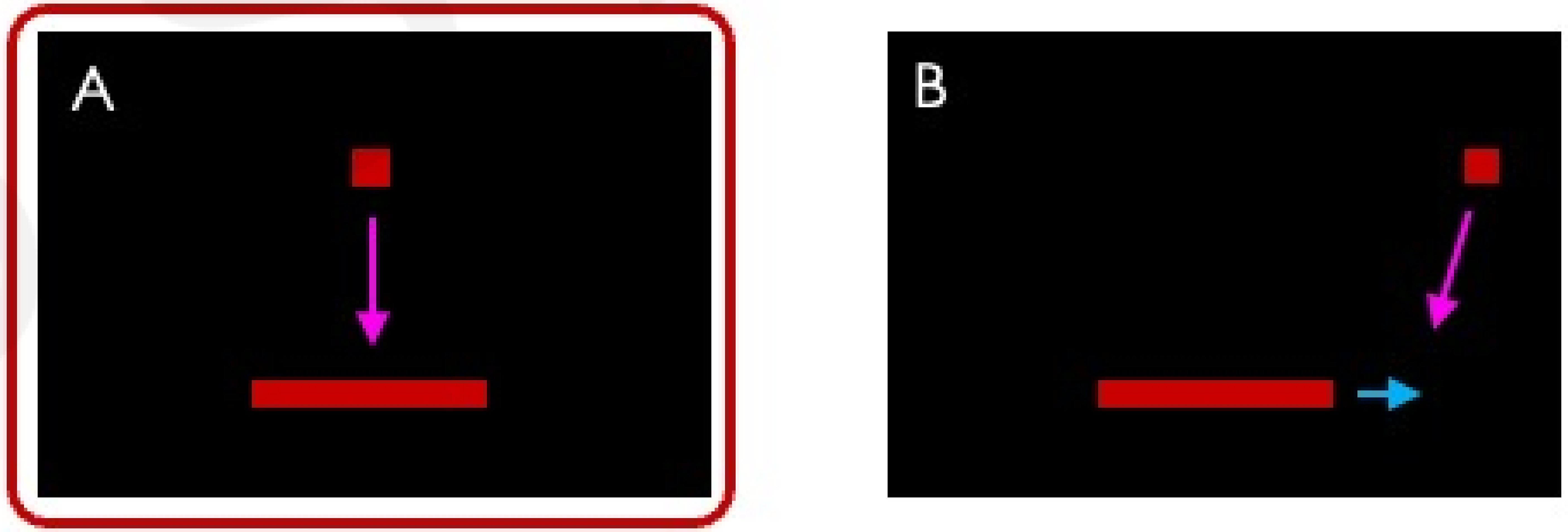
Which (s, a) pair has a higher Q-value? 🤔

Digging deeper into the Q-function

Example: Atari Breakout - Middle



It can be very difficult for humans to accurately estimate Q-values

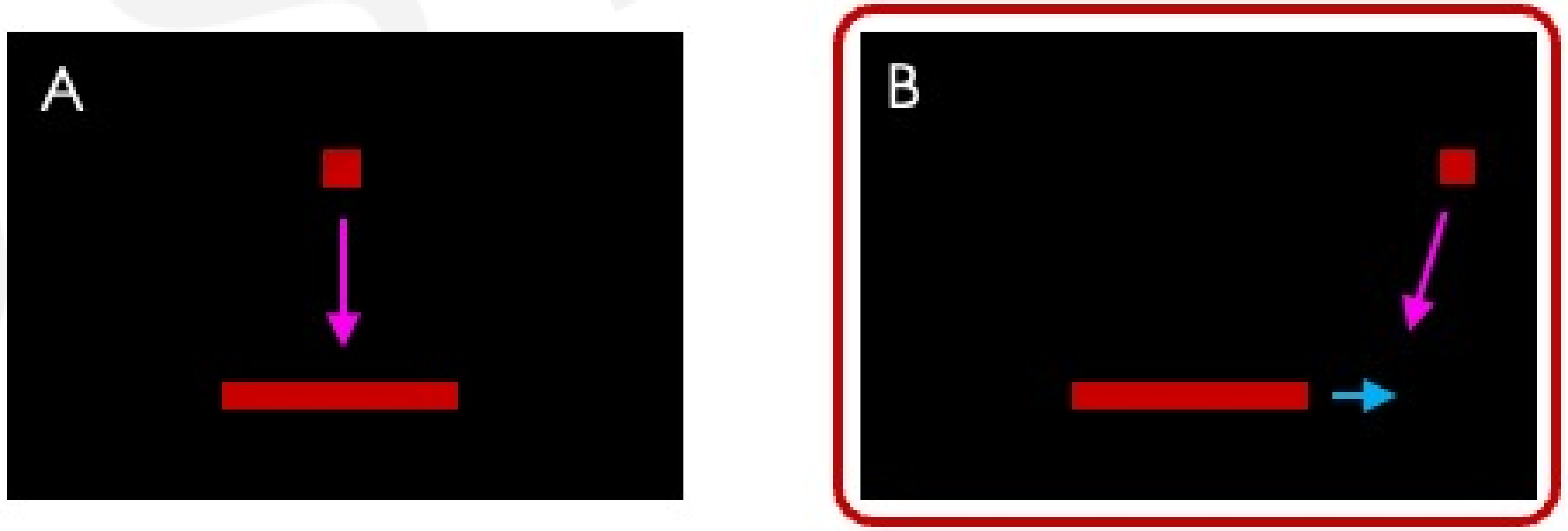


Which (**s**, **a**) pair has a higher Q-value? 🤔

Digging deeper into the Q-function

Example: Atari Breakout - Side

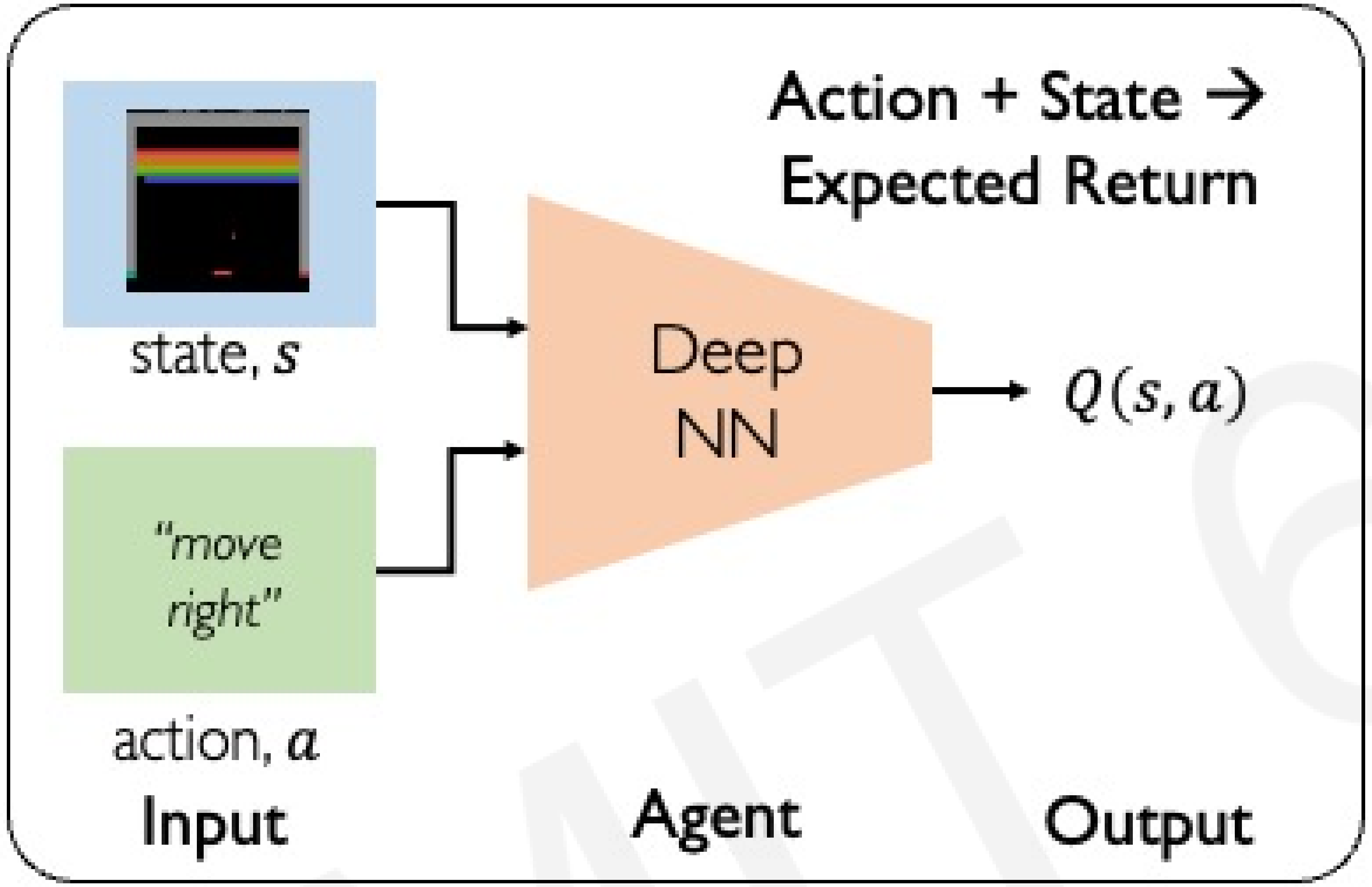
It can be very difficult for humans to accurately estimate Q-values



Which (s, a) pair has a higher Q-value? 🤔

Deep Q Networks (DQN)

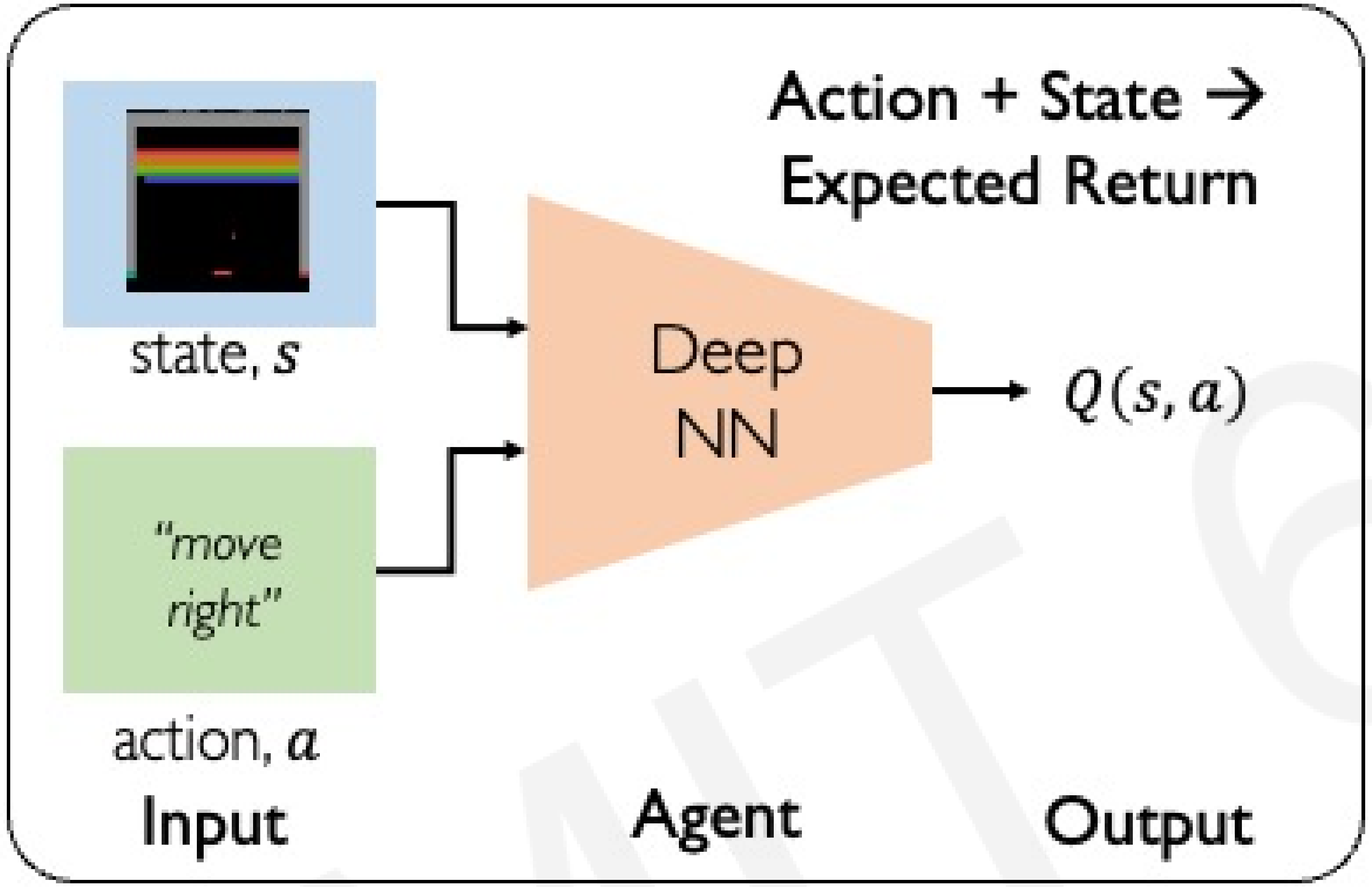
How can we use deep neural networks to model Q-functions?

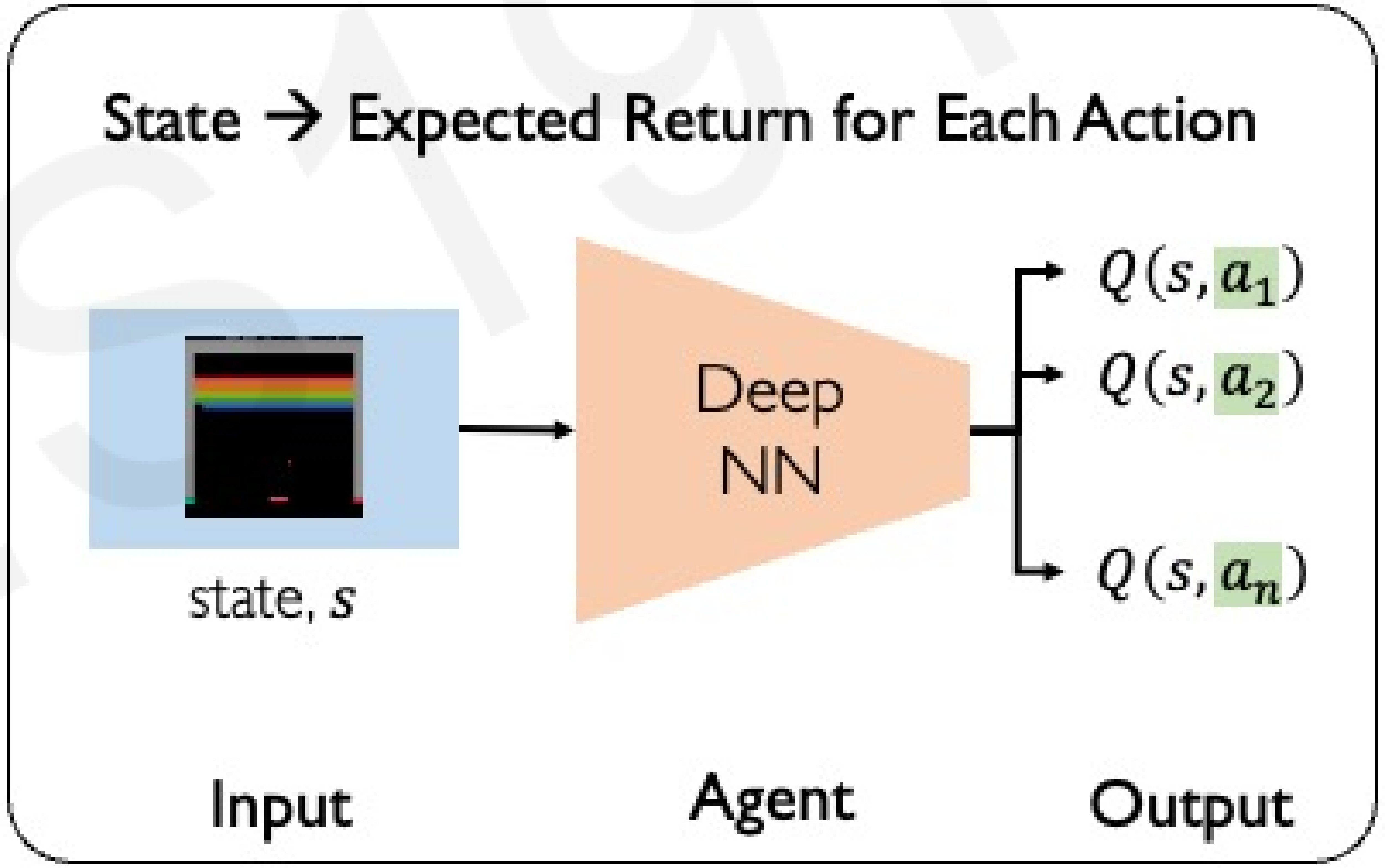


Mnih+ *Nature* 2015.

Deep Q Networks (DQN)

How can we use deep neural networks to model Q-functions?

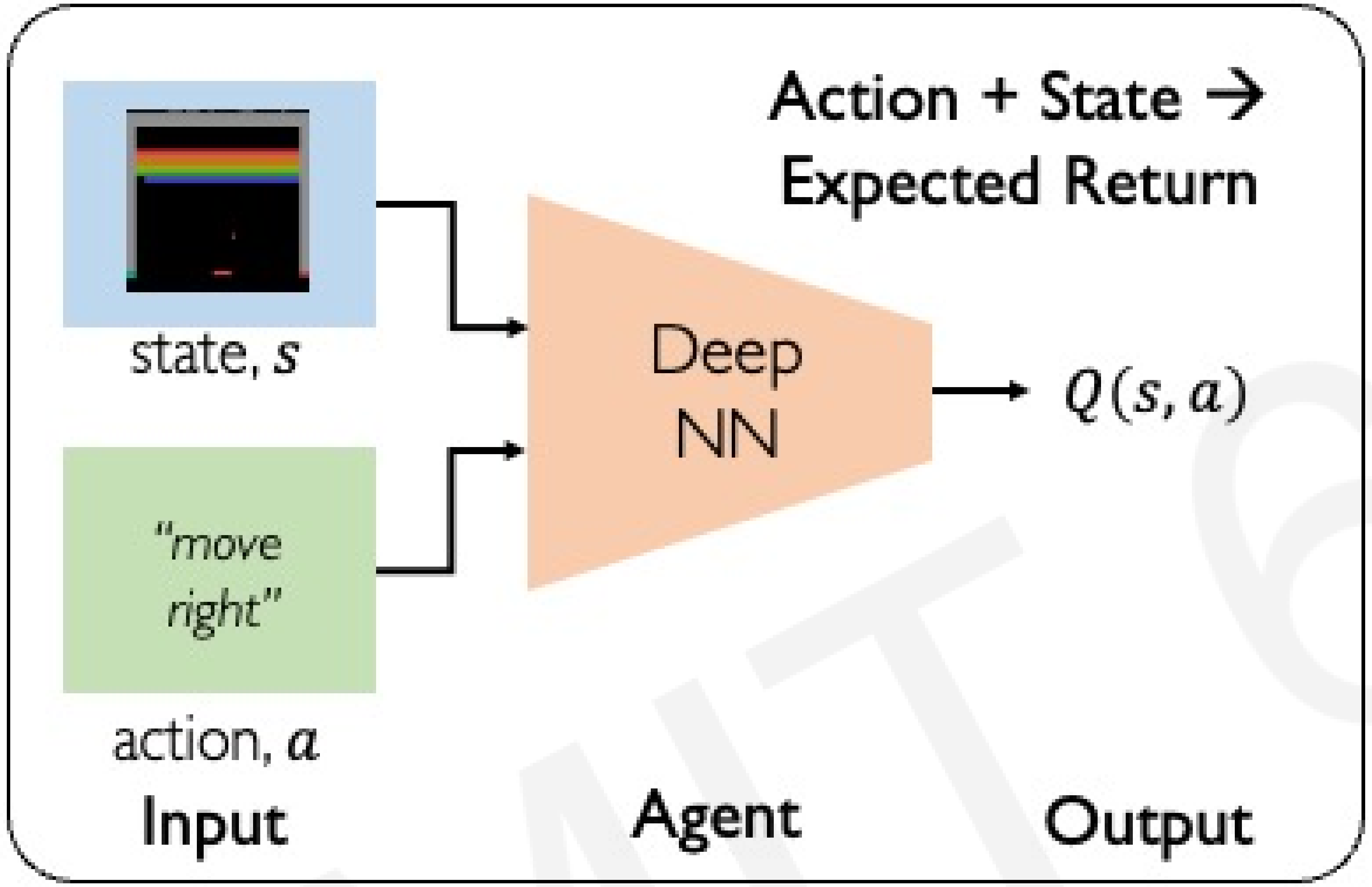




Mnih+ *Nature* 2015.

Deep Q Networks (DQN): Training

How can we use deep neural networks to model Q-functions?



Action + State → Expected Return

state, s

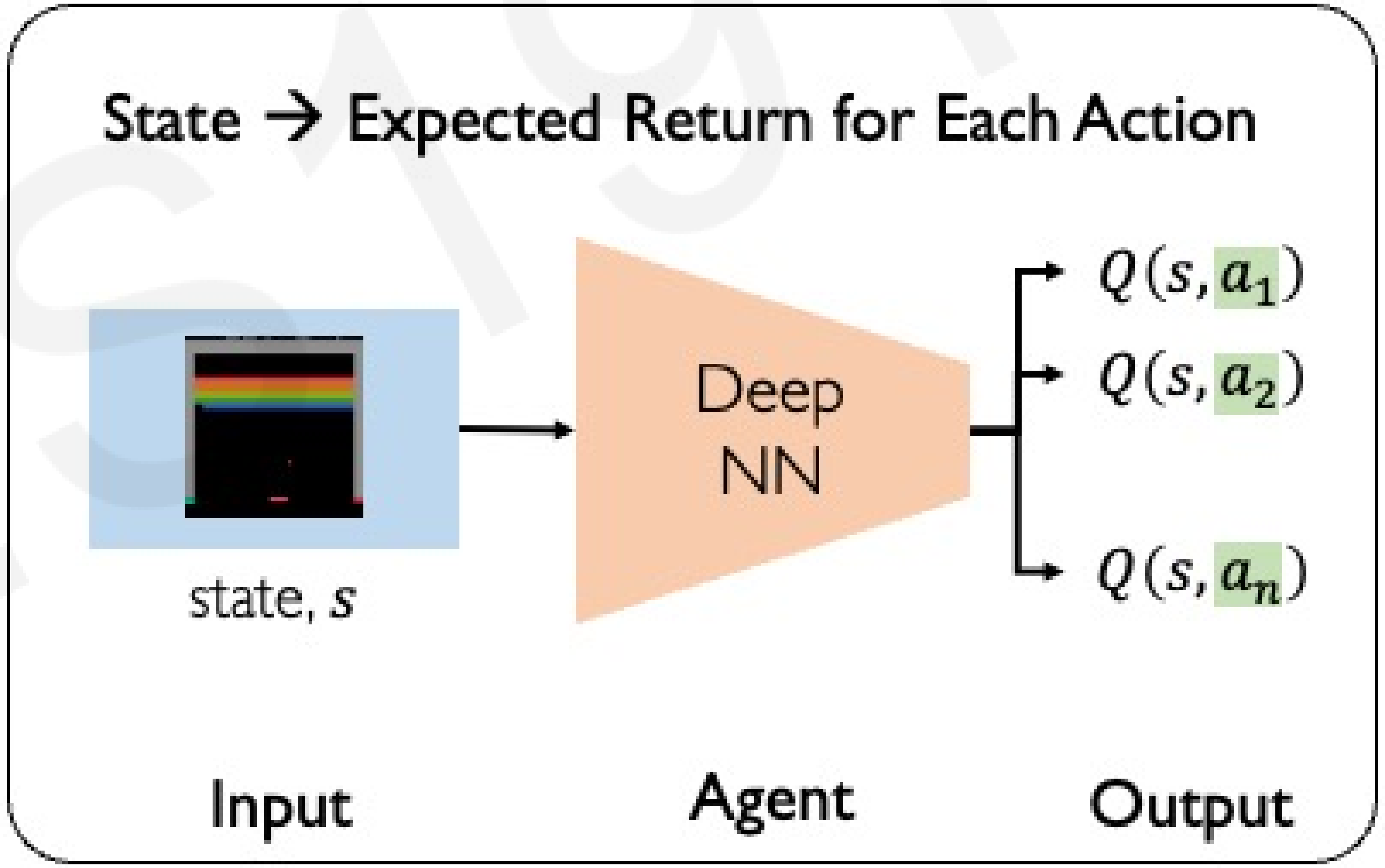
"move right"

action, a

Deep NN

$Q(s, a)$

Input **Agent** **Output**



State → Expected Return for Each Action

state, s

Deep NN

$Q(s, a_1)$

$Q(s, a_2)$

$Q(s, a_n)$

Input **Agent** **Output**

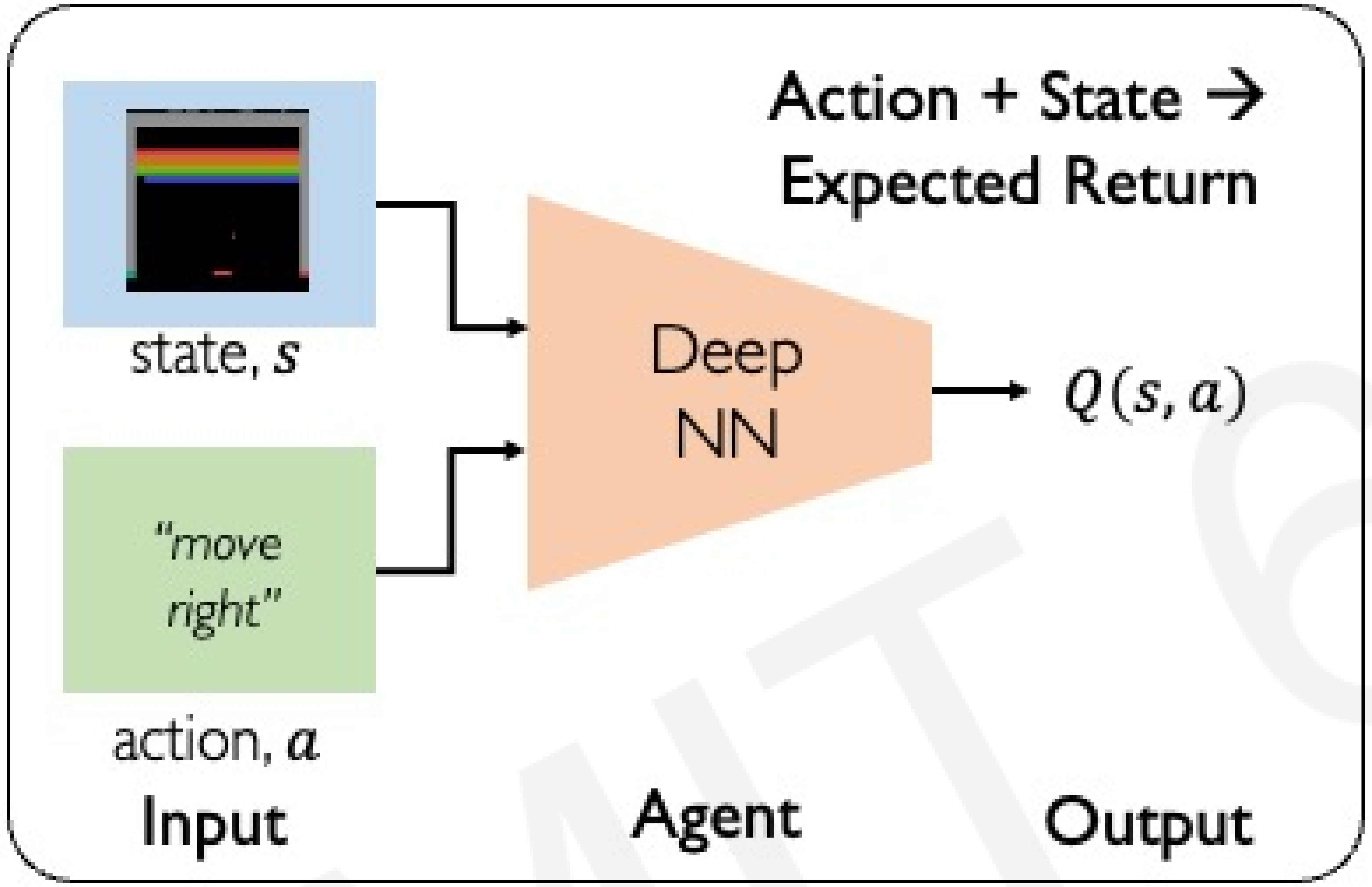
What happens if we take all the best actions?

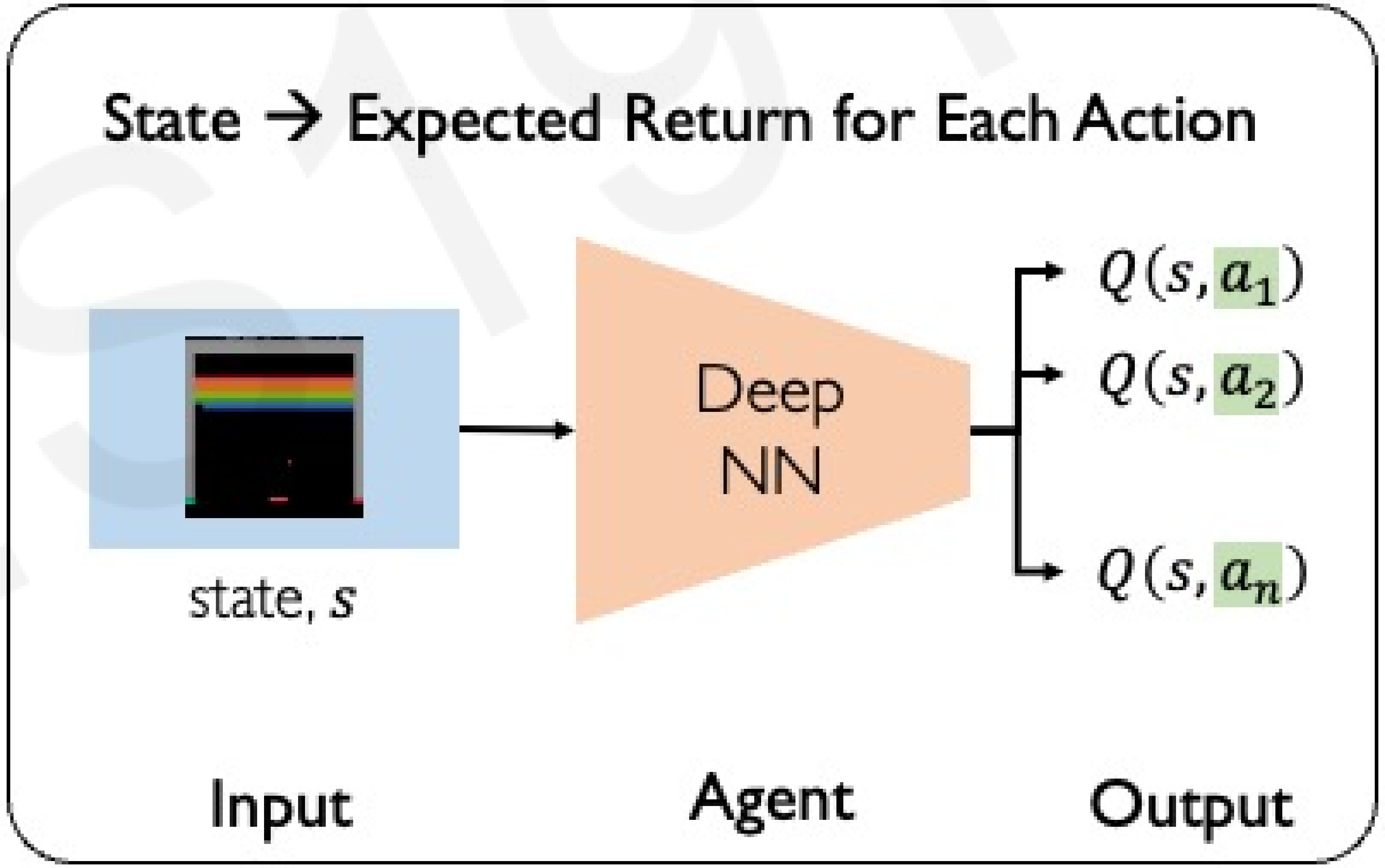
Maximize target return → train the agent

Mnih+ *Nature* 2015.

Deep Q Networks (DQN): Training

How can we use deep neural networks to model Q-functions?





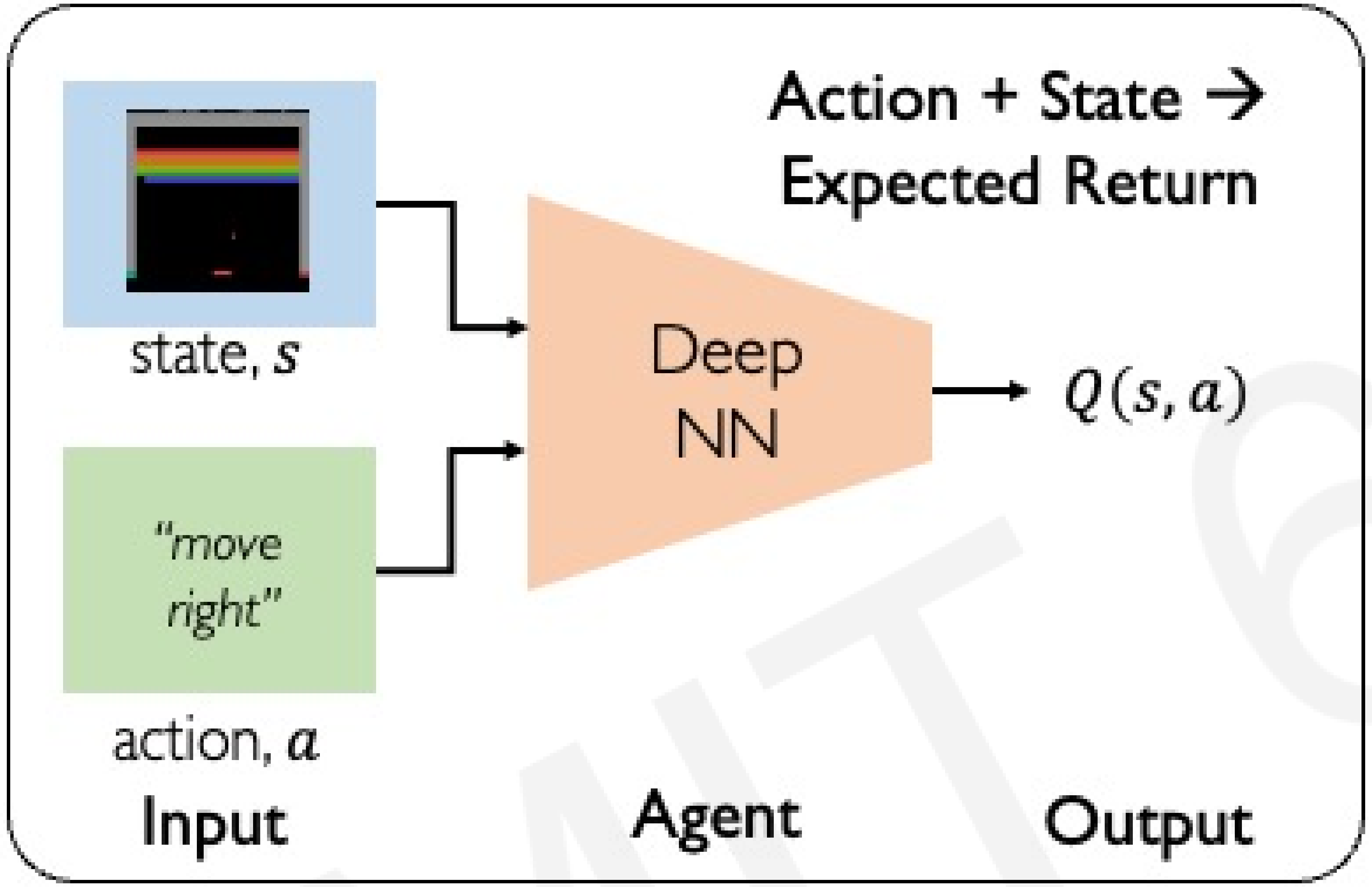
$$\overbrace{\left(r + \gamma \max_{a'} Q(s',a')\right)}^{\text{target}}$$

Take all the best actions → target return

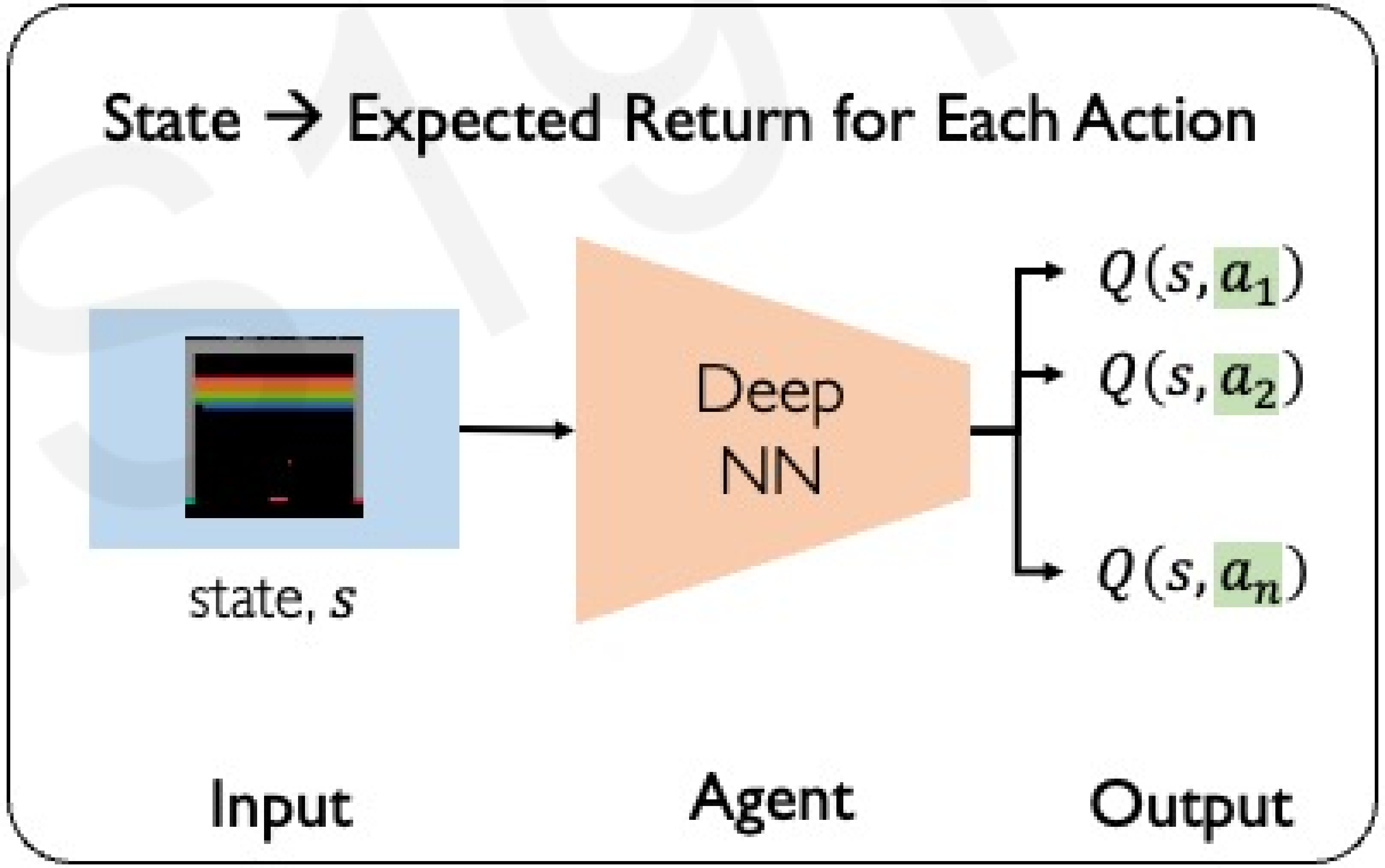
Mnih+ *Nature* 2015.

Deep Q Networks (DQN): Training

How can we use deep neural networks to model Q-functions?



Action + State → Expected Return



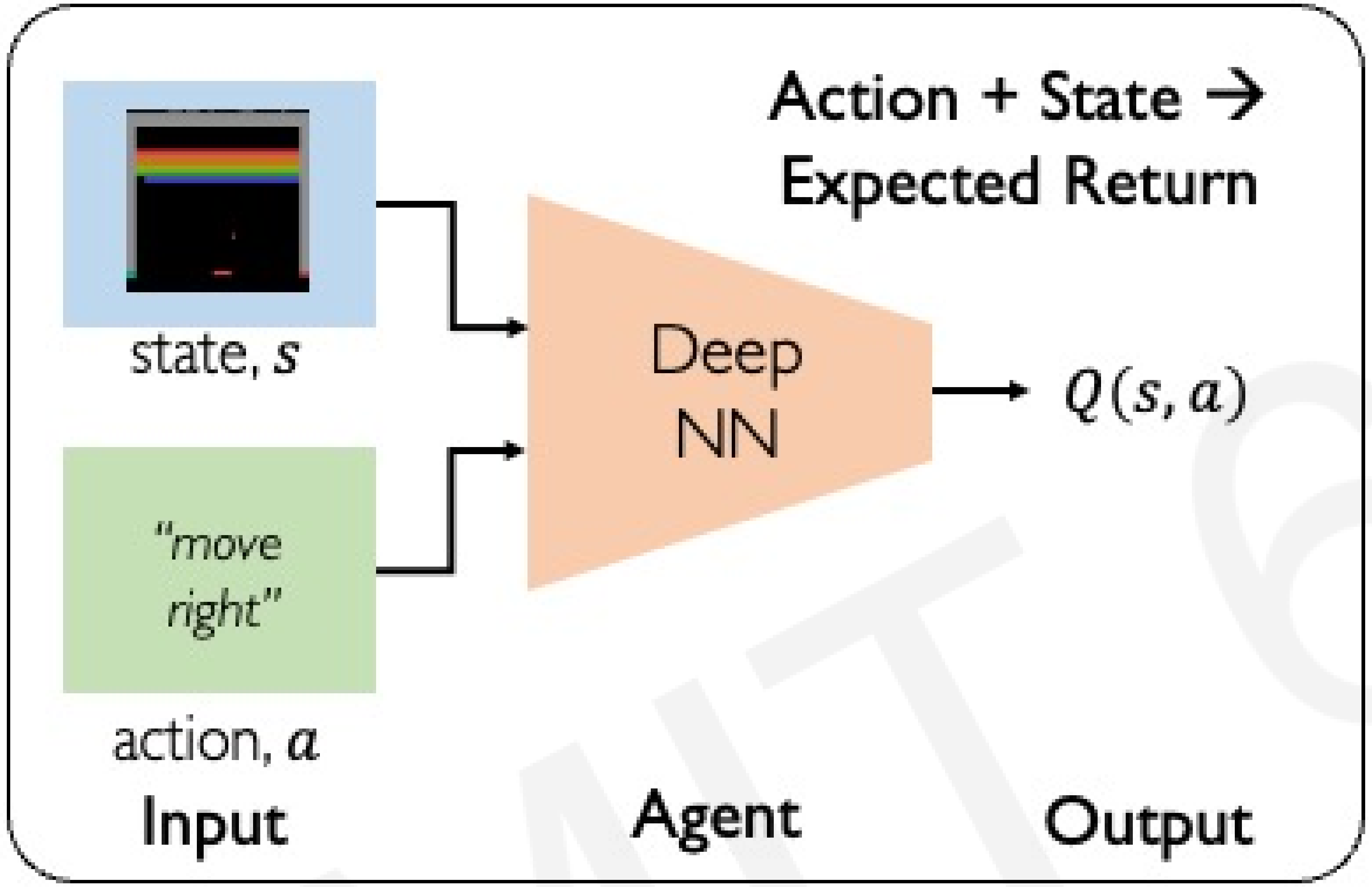
State → Expected Return for Each Action

$$\overbrace{\left(r+\gamma\max_{a'}Q(s',a')\right)}^{\text{target}} \qquad \overbrace{Q(s,a)}^{\text{predicted}}$$

Network prediction

Deep Q Networks (DQN): Training

How can we use deep neural networks to model Q-functions?



Action + State → Expected Return

state, s

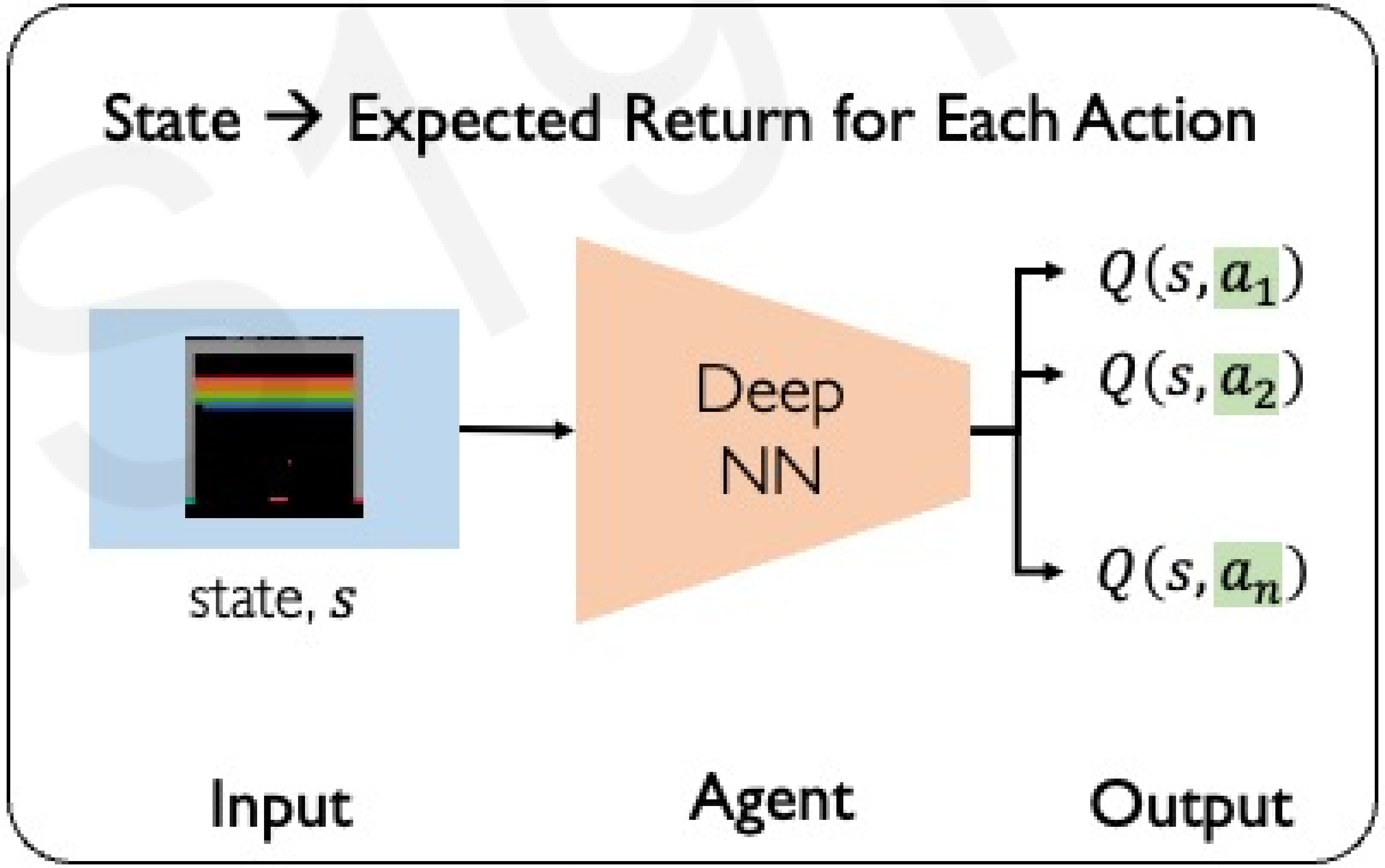
"move right"

action, a

Deep NN

$Q(s,a)$

Input **Agent** **Output**



State → Expected Return for Each Action

state, s

Deep NN

$Q(s,a_1)$
$Q(s,a_2)$
$Q(s,a_n)$

Input **Agent** **Output**

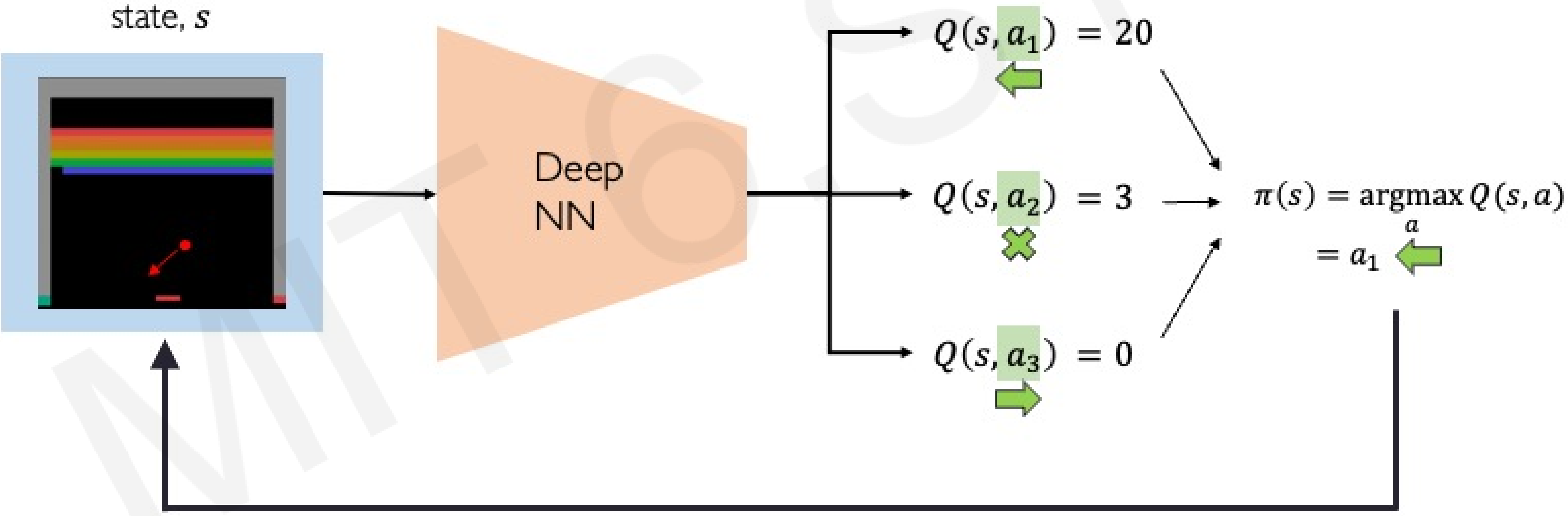
$$\mathcal{L} = \mathbb{E}\left[\left\| \overbrace{\left(r + \gamma \max_{a'} Q(s', a')\right)}^{\text{target}} - \overbrace{Q(s,a)}^{\text{predicted}} \right\|^2\right]$$

Q-Loss

Mnih+ *Nature* 2015. 1/10/24

Deep Q Network Summary

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



state, s

Deep NN

$Q(s, a_1) = 20$ ⇦

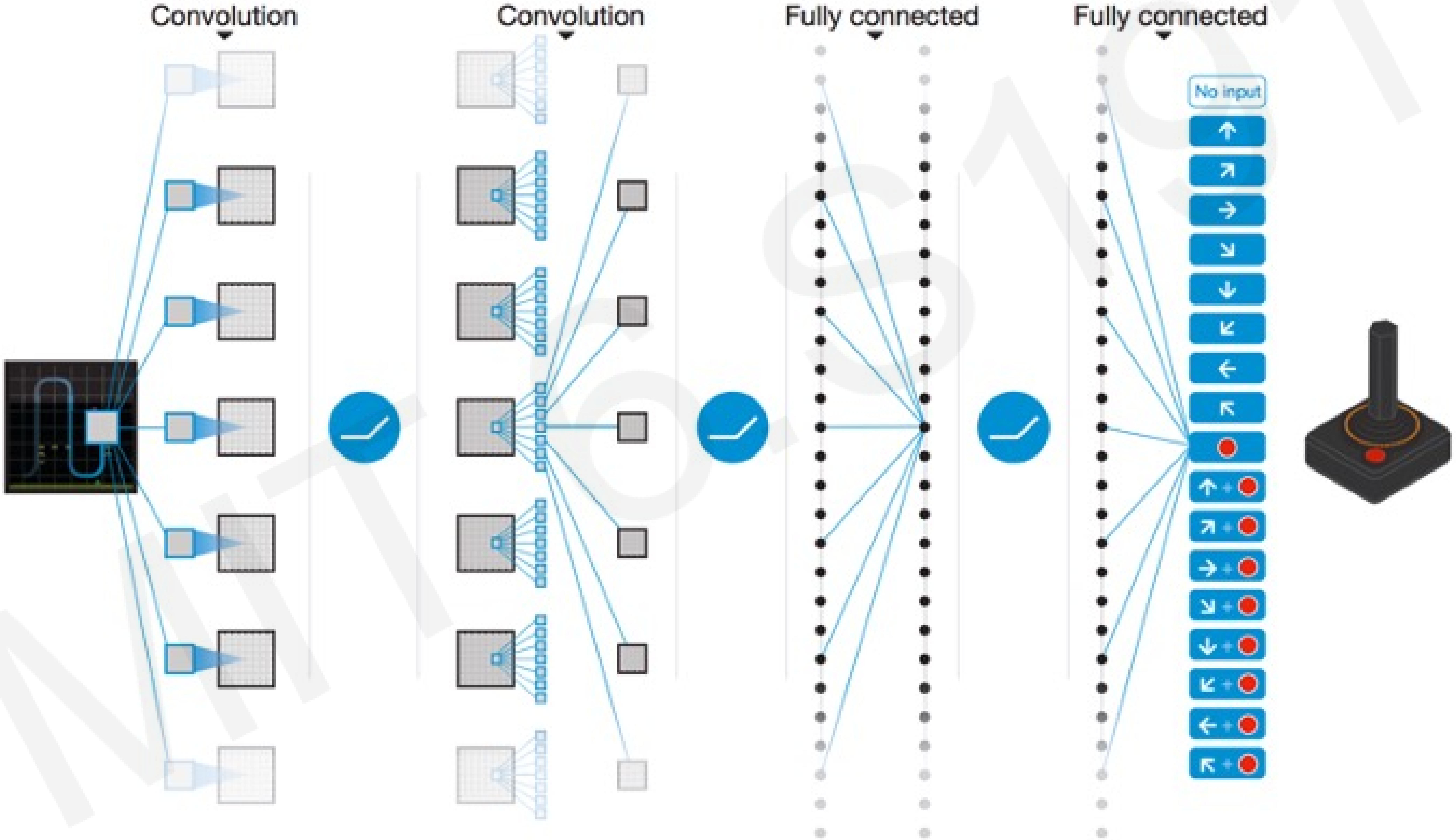
$Q(s, a_2) = 3$ ✖

$Q(s, a_3) = 0$ ⇨

$$\pi(s) = \underset{a}{\operatorname{argmax}}\, Q(s,a) = a_1$$ ⇦

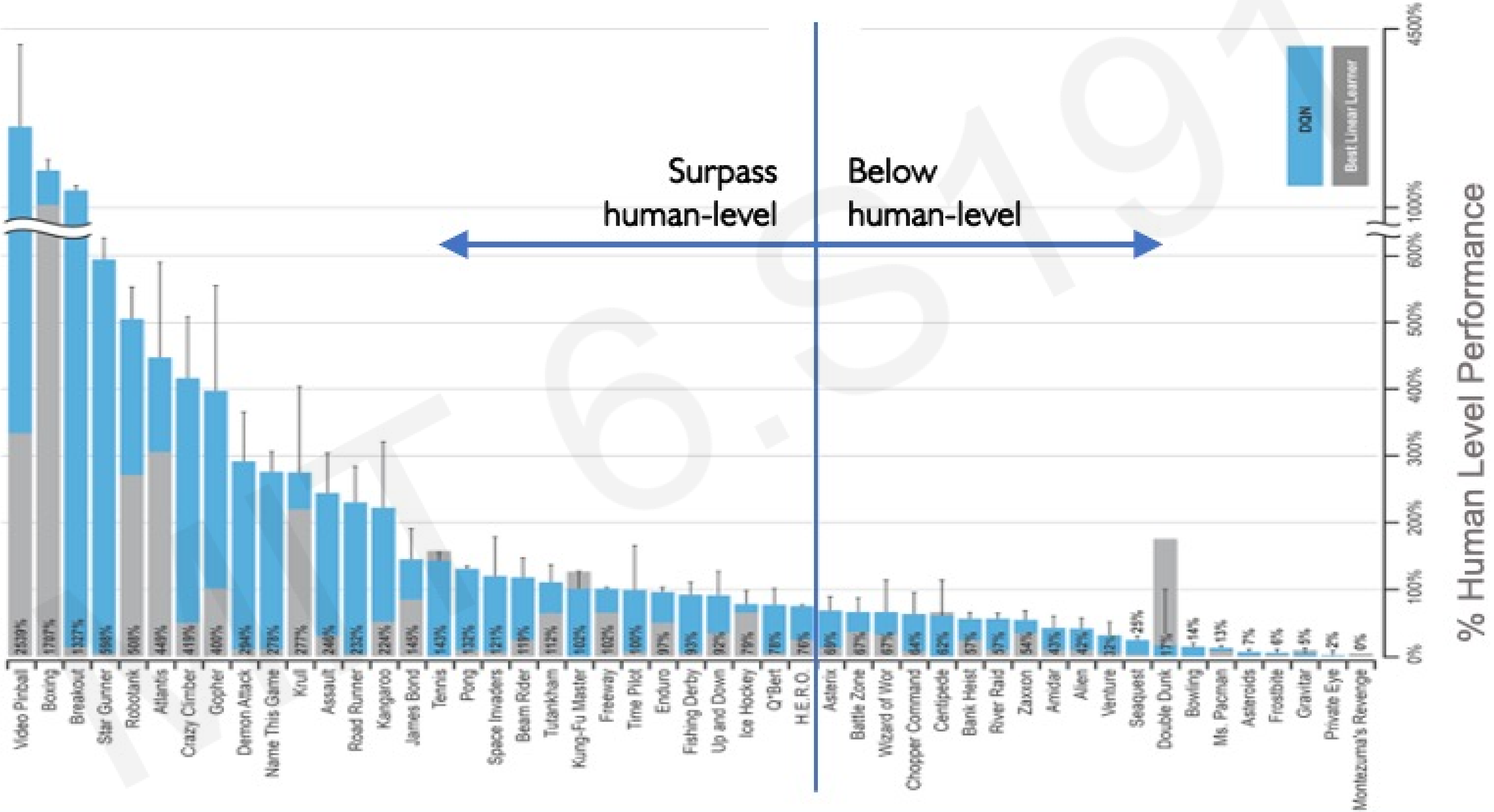
Send action back to environment and receive next state

DQN Atari Results



Mnih+ *Nature* 2015.

DQN Atari Results



Surpass human-level

Below human-level

Mnih+ *Nature* 2015.

Downsides of Q-learning

Complexity:
- Can model scenarios where the action space is discrete and small
- Cannot handle continuous action spaces

Flexibility:
- Policy is deterministically computed from the Q function by maximizing the reward → cannot learn stochastic policies

To address these, consider a new class of RL training algorithms:
Policy gradient methods

Deep Reinforcement Learning Algorithms

Value Learning

Find $Q(s, a)$

$$a = \underset{a}{\operatorname{argmax}}\, Q(s, a)$$

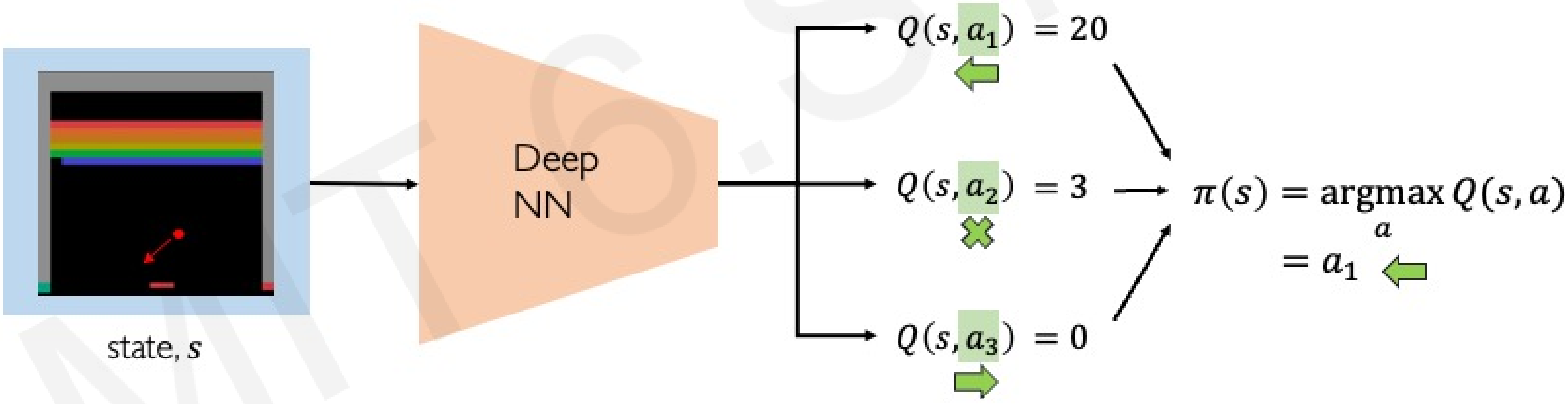
Policy Learning

Find $\pi(s)$

Sample $a \sim \pi(s)$

Deep Q Networks (DQN)

DQN: Approximate Q-function and use to infer the optimal policy, $\pi(s)$



state, s

Deep NN

$Q(s, a_1) = 20$ ⇦

$Q(s, a_2) = 3$ ✖

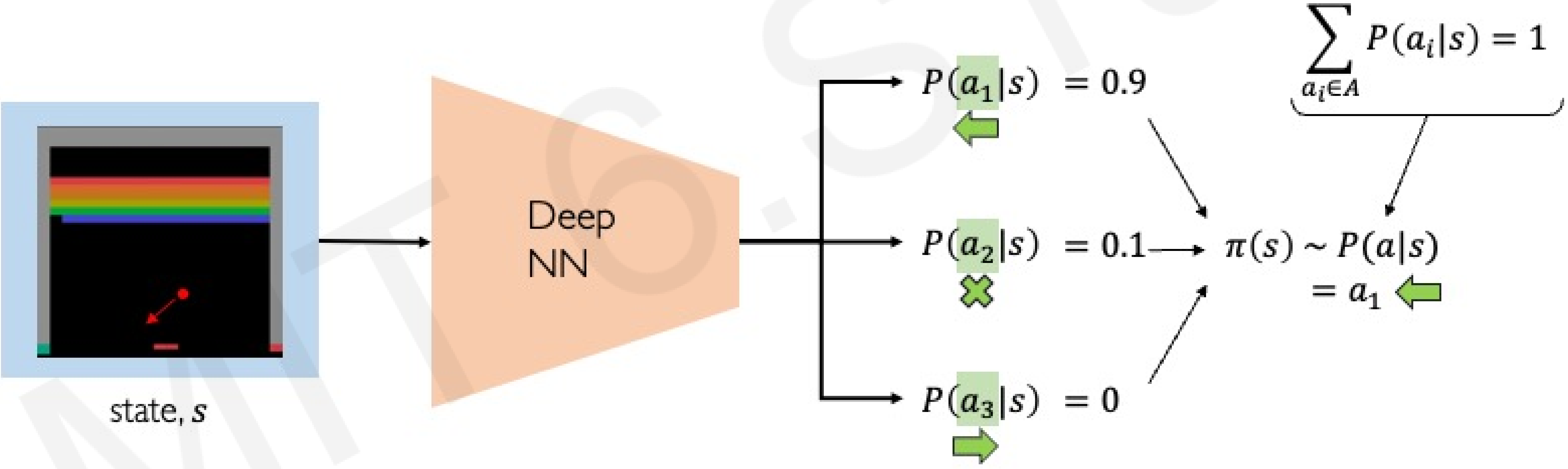
$Q(s, a_3) = 0$ ⇨

$$\pi(s) = \underset{a}{\operatorname{argmax}}\, Q(s,a)$$
$$= a_1$$ ⇦

Policy Gradient (PG): Key Idea

DQN: Approximate Q-function and use to infer the optimal policy, $\pi(s)$

Policy Gradient: Directly optimize the policy $\pi(s)$



state, s

Deep NN

$P(a_1|s) = 0.9$ ⇦

$P(a_2|s) = 0.1$ ✖

$P(a_3|s) = 0$ ⇨

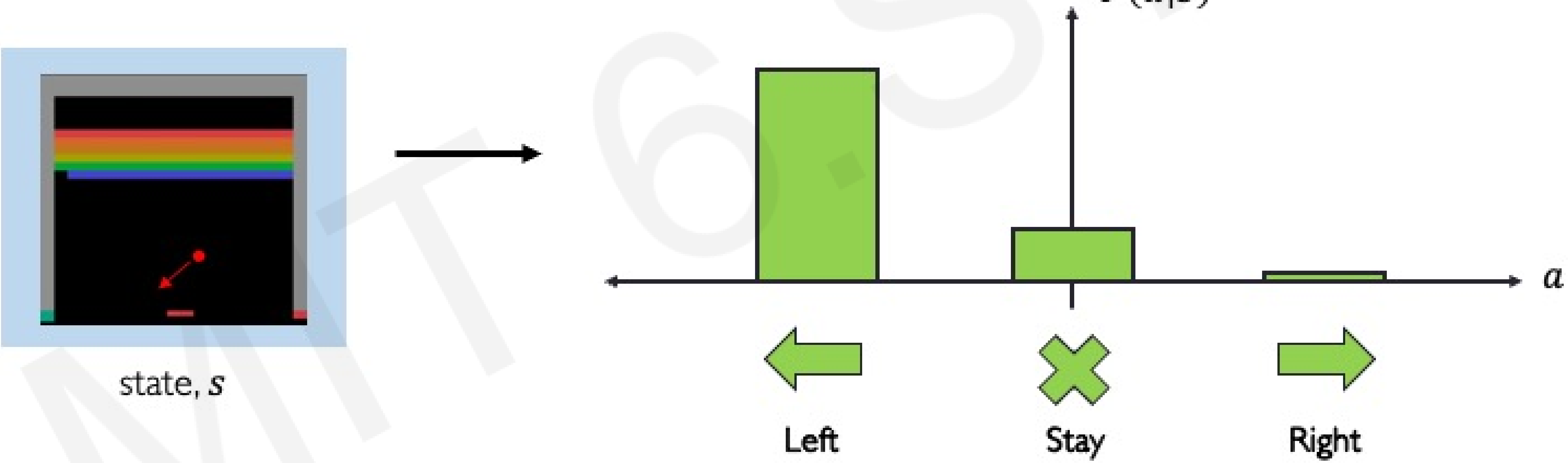
$$\underbrace{\sum_{a_i \in A} P(a_i|s) = 1}$$

$$\pi(s) \sim P(a|s) = a_1$$ ⇦

🤔 What are some advantages of this formulation?

Discrete vs Continuous Action Spaces

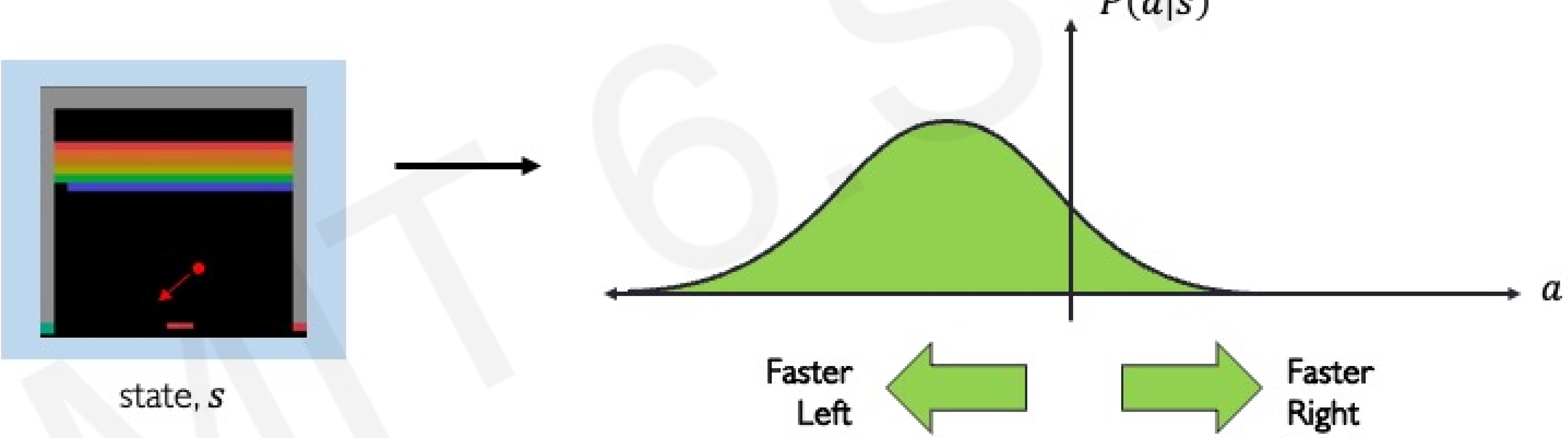
Discrete action space: which direction should I move? ⇦ ✖ ⇨



Discrete vs Continuous Action Spaces

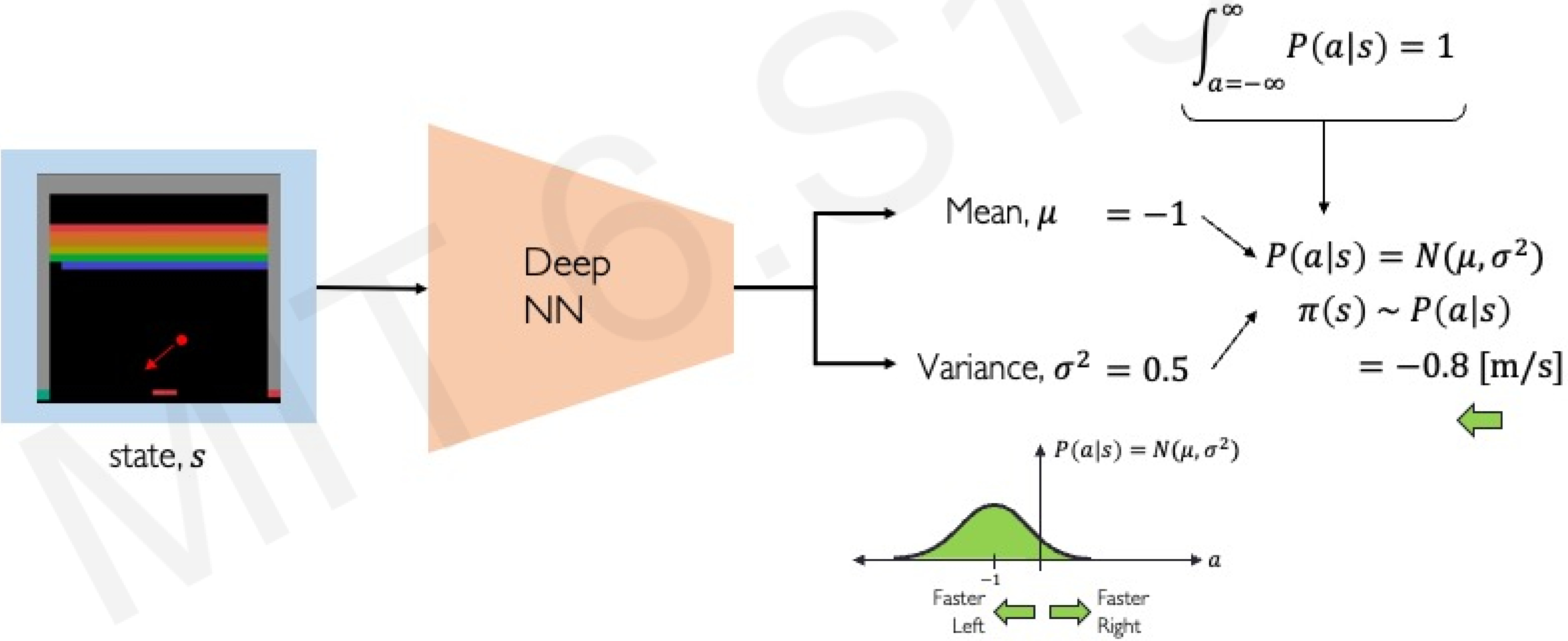
Discrete action space: which direction should I move? ⇦ ✖ ⇨

Continuous action space: how fast should I move? ⇦ 0.7 m/s



Policy Gradient (PG): Key Idea

Policy Gradient: Enables modeling of continuous action space



state, s

Deep NN

Mean, $\mu = -1$

Variance, $\sigma^2 = 0.5$

$$\underbrace{\int_{a=-\infty}^{\infty} P(a|s) = 1}$$

$$P(a|s) = N(\mu, \sigma^2)$$

$$\pi(s) \sim P(a|s)$$

$$= -0.8 \text{ [m/s]}$$

Training Policy Gradients: Case Study

Reinforcement Learning Loop:

Case Study – Self-Driving Cars

Agent: vehicle

State: camera, lidar, etc

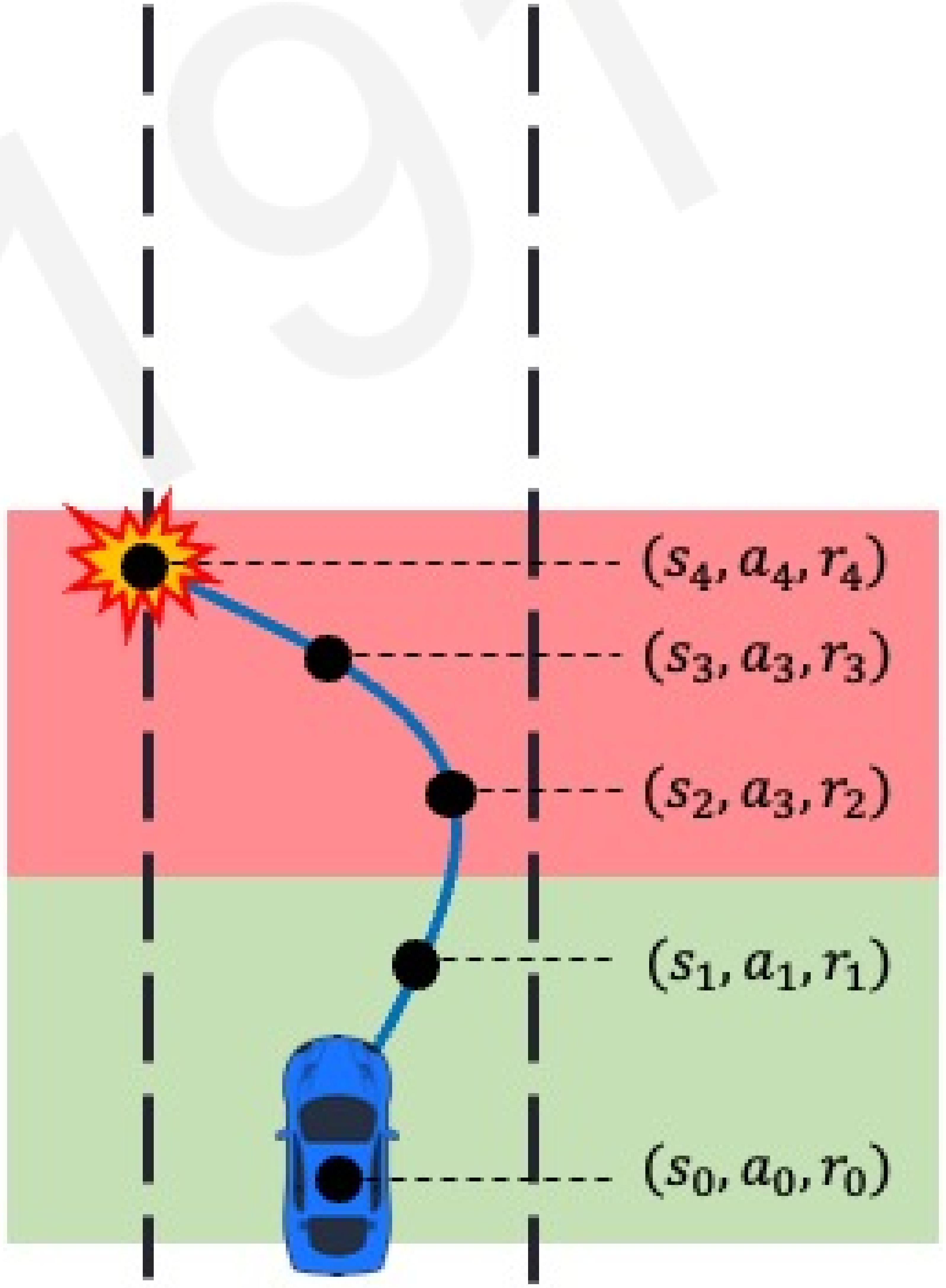
Action: steering wheel angle

Reward: distance traveled

Training Policy Gradients

Training Algorithm

1. Initialize the agent
2. Run a policy until termination
3. Record all states, actions, rewards
4. Decrease probability of actions that resulted in low reward
5. Increase probability of actions that resulted in high reward

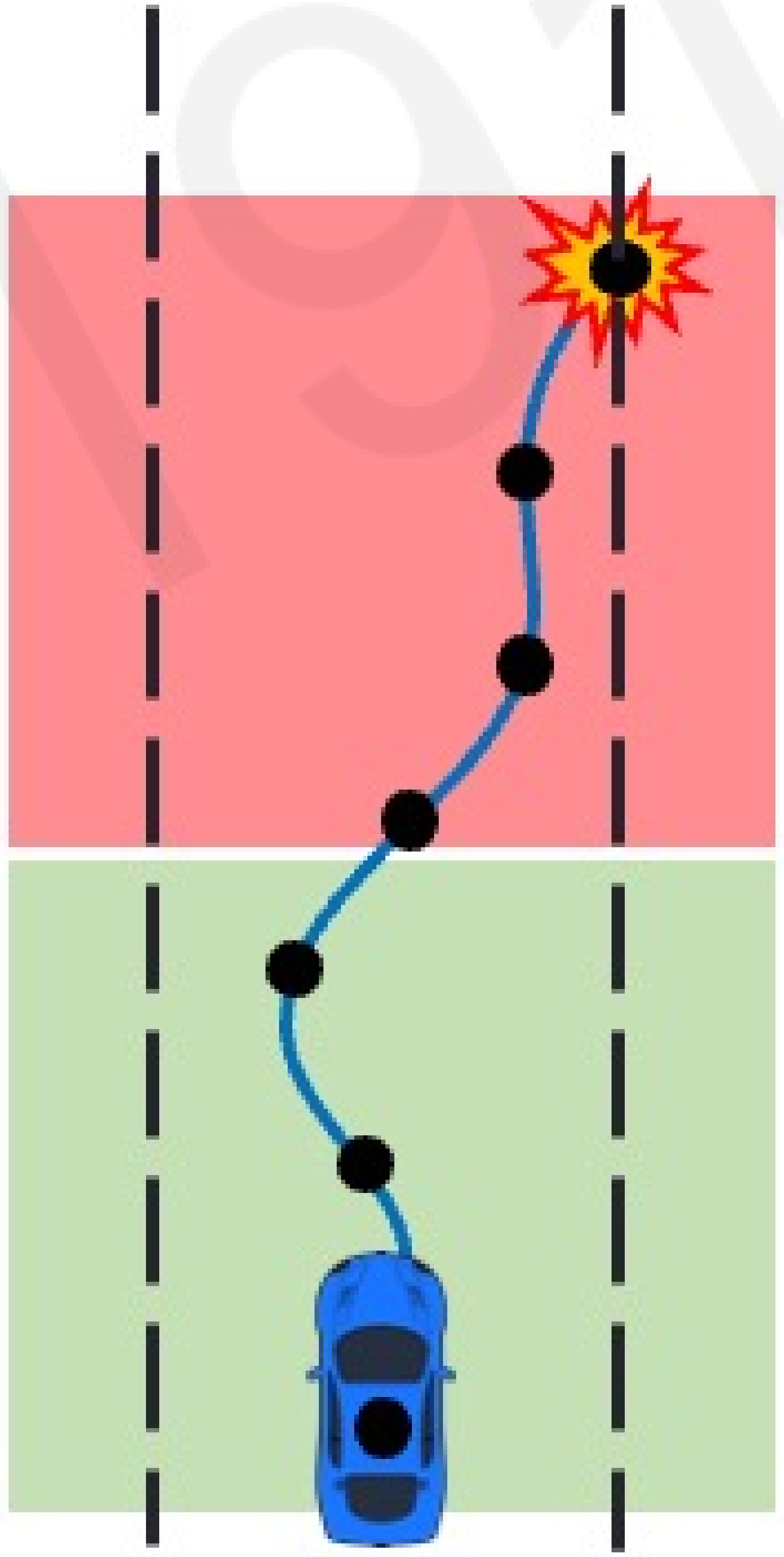


Williams *Reinforcement Learning* 1992.

Training Policy Gradients

Training Algorithm

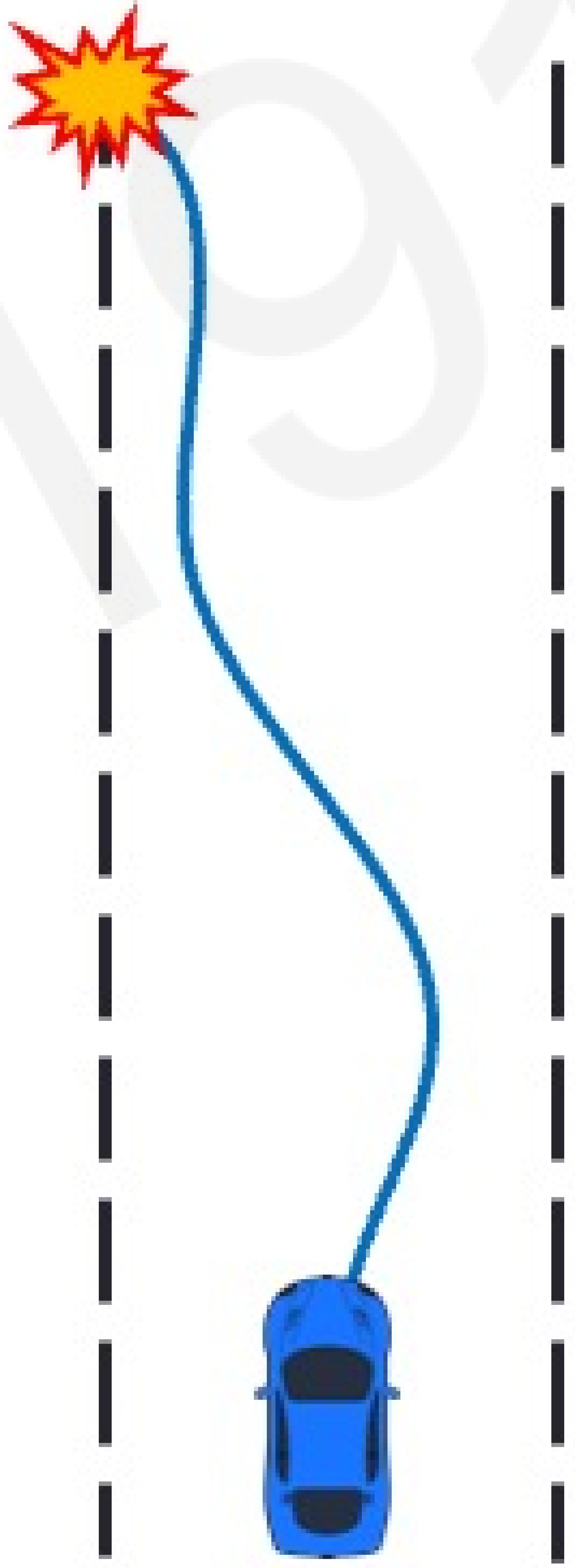
1. Initialize the agent
2. Run a policy until termination
3. Record all states, actions, rewards
4. Decrease probability of actions that resulted in low reward
5. Increase probability of actions that resulted in high reward



Training Policy Gradients

Training Algorithm

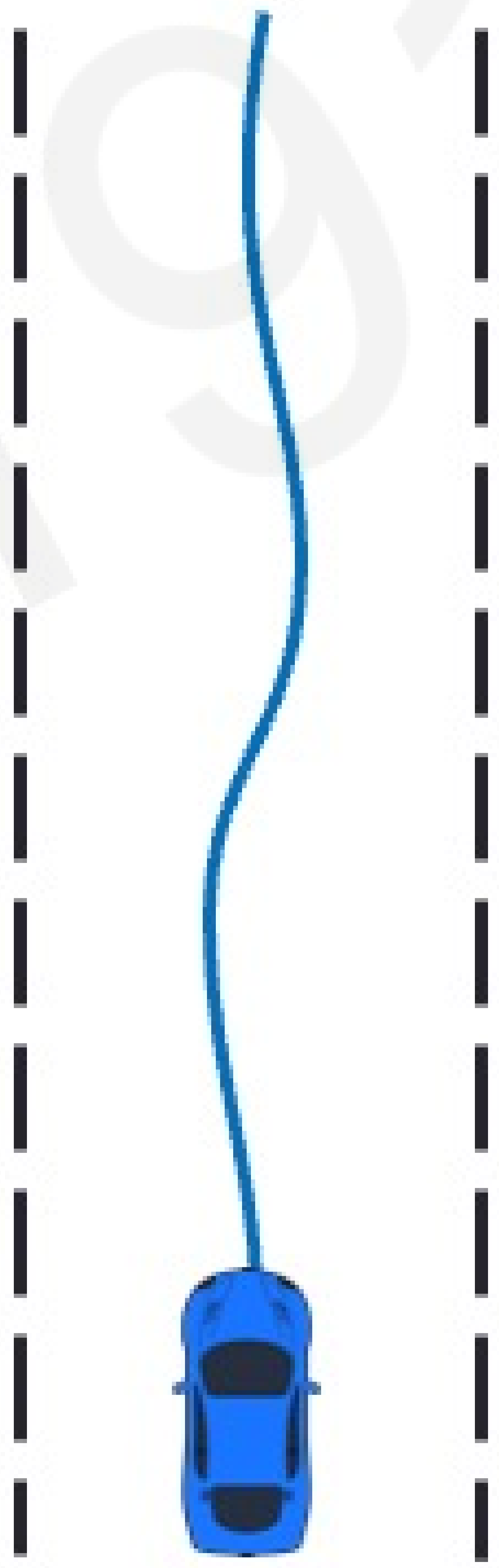
1. Initialize the agent
2. Run a policy until termination
3. Record all states, actions, rewards
4. Decrease probability of actions that resulted in low reward
5. Increase probability of actions that resulted in high reward



Training Policy Gradients

Training Algorithm

1. Initialize the agent
2. Run a policy until termination
3. Record all states, actions, rewards
4. Decrease probability of actions that resulted in low reward
5. Increase probability of actions that resulted in high reward



Training Policy Gradients

Training Algorithm

1. Initialize the agent
2. Run a policy until termination
3. Record all states, actions, rewards
4. Decrease probability of actions that resulted in low reward
5. Increase probability of actions that resulted in high reward

$$\mathbf{loss} = -\overbrace{\log \mathrm{P}(a_t|s_t)}^{\text{log-likelihood of action}}\,\underbrace{R_t}_{\text{reward}}$$

Gradient descent update:

$$w' = w - \nabla\,\mathbf{loss}$$
$$w' = w + \underbrace{\boxed{\nabla \log \mathrm{P}(a_t|s_t)\,R_t}}_{\text{Policy gradient!}}$$

Williams *Reinforcement Learning* 1992. 1/10/24

Reinforcement Learning in Real Life

Training Algorithm

1. Initialize the agent
2. Run a policy until termination
3. Record all states, actions, rewards
4. Decrease probability of actions that resulted in low reward
5. Increase probability of actions that resulted in high reward

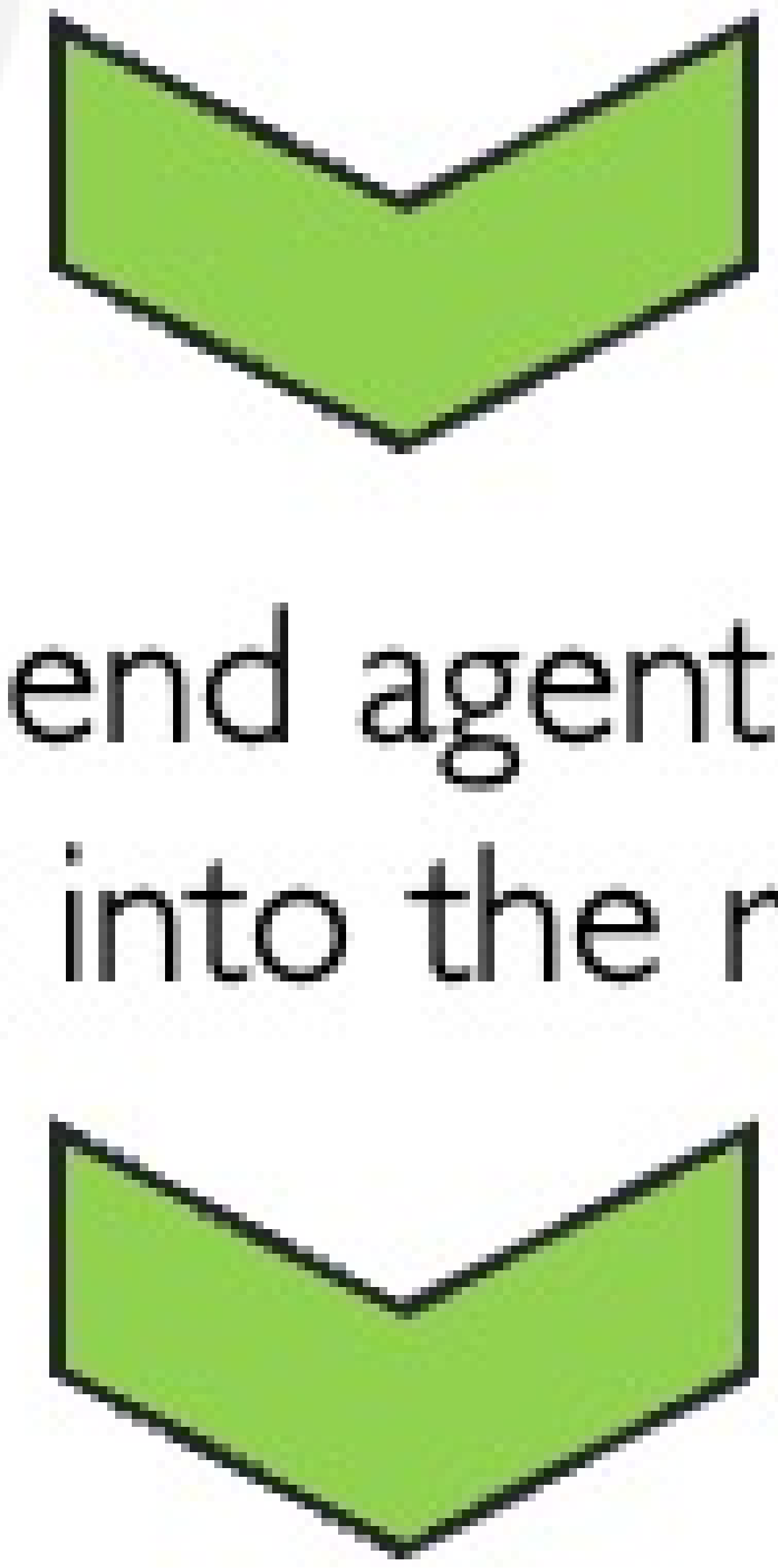
Amini+ *IEEE RA-L* 2020.

Data-driven Simulation for Autonomous Vehicles

VISTA: Photorealistic and high-fidelity simulator for training and testing self-driving cars

Amini+ *IEEE RA-L* 2020.

Deploying End-to-End RL for Autonomous Vehicles



Policy Gradient RL agent trained entirely within VISTA simulator

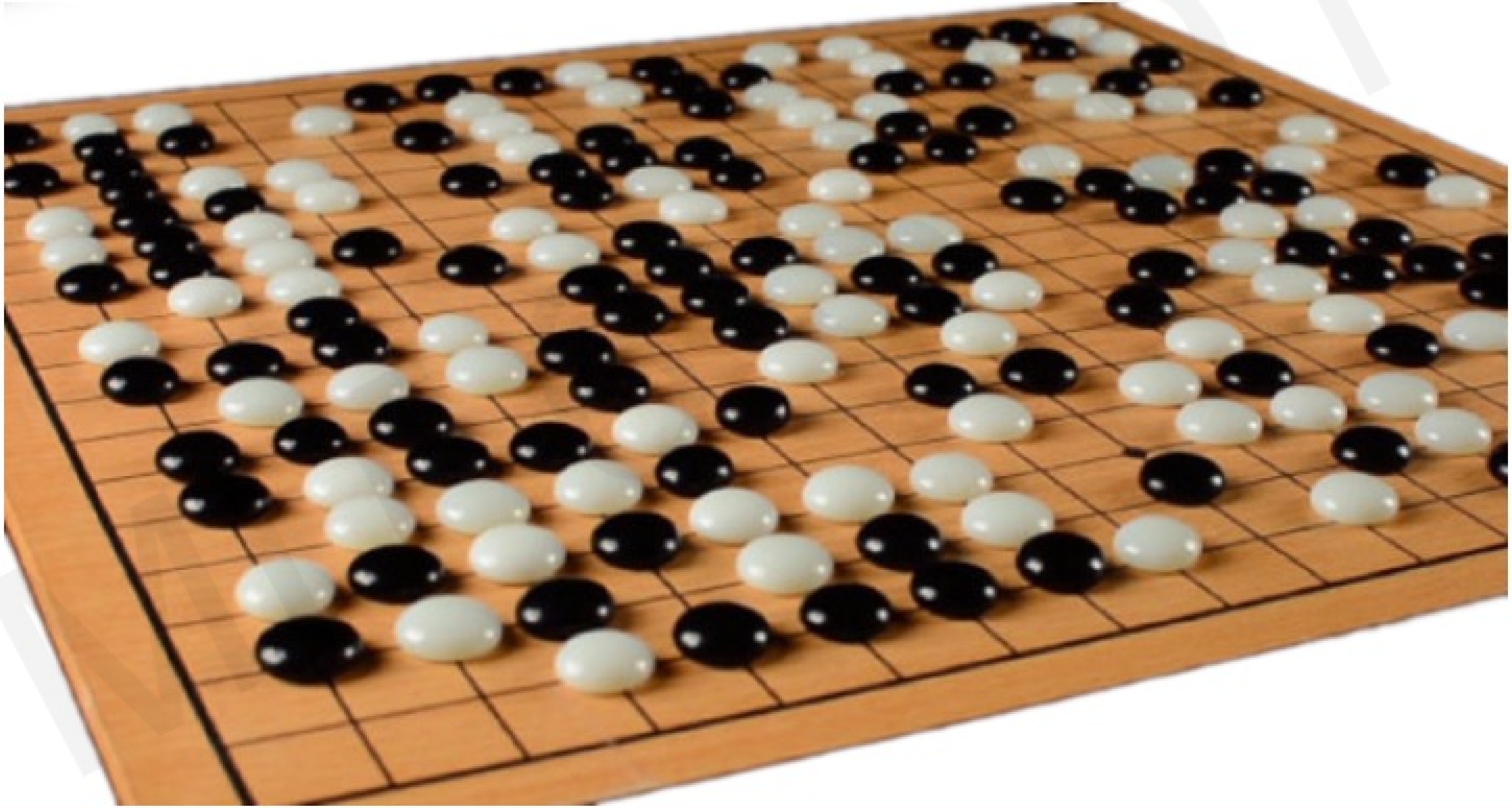
End-to-end agent directly deployed into the real-world

First full-scale autonomous vehicle trained using RL entirely in simulation and deployed in real life!

Amini+ *IEEE RA-L* 2020.

Deep Reinforcement Learning Applications

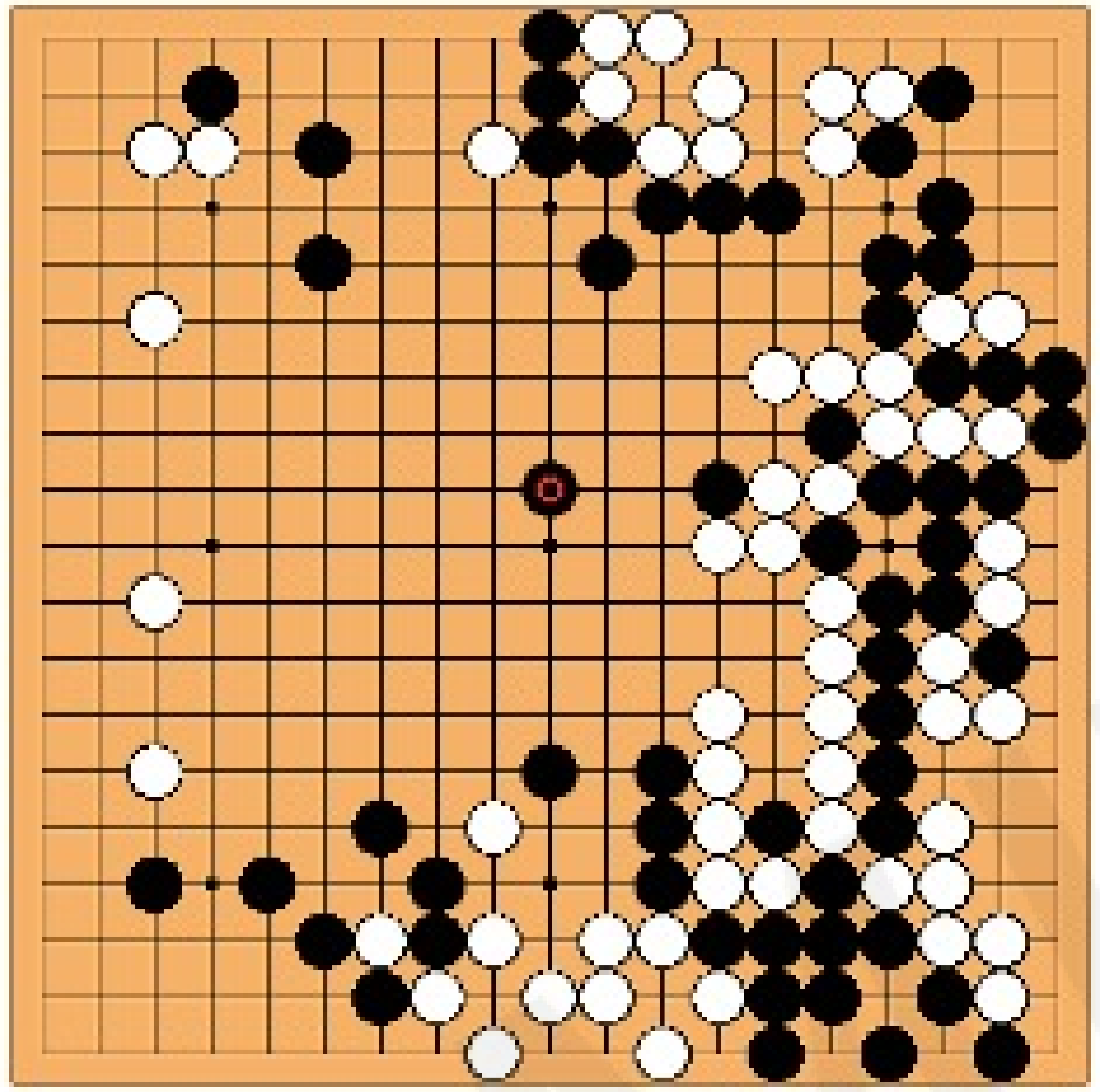
Reinforcement Learning and the Game of Go



Silver+ *Science* 2016; Silver+ *Nature* 2017.

The Game of Go

Aim: Get more board territory than your opponent.



Board Size n x n	Positions 3^{n^2}	% Legal	Legal Positions
1×1	3	33.33%	1
2×2	81	70.37%	57
3×3	19,683	64.40%	12,675
4×4	43,046,721	56.49%	24,318,165
5×5	847,288,609,443	48.90%	414,295,148,741
9×9	$4.434264882 \times 10^{38}$	23.44%	$1.039191487910 \times 10^{38}$
13×13	$4.300233593 \times 10^{80}$	8.66%	$3.72497923077 \times 10^{79}$
19×19	$1.740896506 \times 10^{172}$	1.20%	$2.08168199382 \times 10^{170}$

Greater number of legal board positions than atoms in the universe.

Source: Wikipedia.

AlphaGo Beats Top Human Player at Go

Silver+ Science 2016.

AlphaGo Beats Top Human Player at Go

1) Initial training: human data

Silver+ Science 2016.

AlphaGo Beats Top Human Player at Go

Human expert positions

Classification

Supervised Learning policy network

Self Play

RL policy network

Self Play

Self-play data

Regression

Value network

1) Initial training: human data

2) Self-play and reinforcement learning
→ super-human performance

Silver+ Science 2016.

AlphaGo Beats Top Human Player at Go

1) Initial training: human data

2) Self-play and reinforcement learning
→ super-human performance

3) "Intuition" about board state

Silver+ Science 2016.

AlphaZero: RL from Self-Play

Silver+ Science 2018.

Deep Reinforcement Learning: Summary

Foundations

- Agents acting in environment
- State-action pairs → maximize future rewards
- Discounting

Q-Learning

- Q function: expected total reward given **s, a**
- Policy determined by selecting action that maximizes Q function

Policy Gradients

- Learn and optimize the policy directly
- Applicable to continuous action spaces

T-Shirts Coming Tomorrow!

SYLLABUS: **bit.ly/6s191-syllabus**

1. Project sign-ups due **TODAY 1/10 11:00pm ET**

2. Lab competitions and prizes!
DEADLINE: Thursday 1/11 11:00pm ET

3. **Project and lab submission links on syllabus!**

4. **RESUME DROP** to sponsoring companies!
introtodeeplearning.com/jobs.html