Deep Reinforcement Learning

Alexander Amini

MIT Introduction to Deep Learning

January 11, 2023

Learning in Dynamic Environments

Reinforcement Learning: Robots, Games, the World

Robotics

Game Play and Strategy


Oriol Vinyals
Co-Lead, AlphaStar Project, DeepMind

Classes of Learning Problems

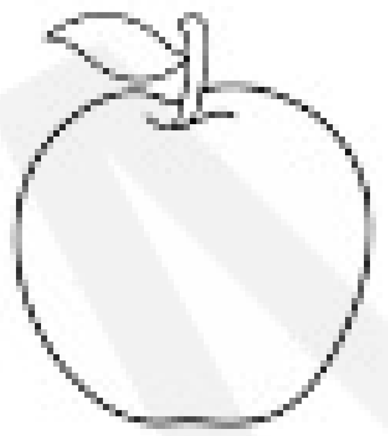
Supervised Learning

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

Goal: Learn function to map

$$x \rightarrow y$$

Apple example:



This thing is an apple.

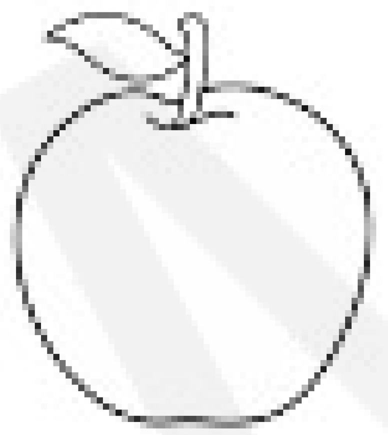
Classes of Learning Problems

Supervised Learning

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

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

Apple example:



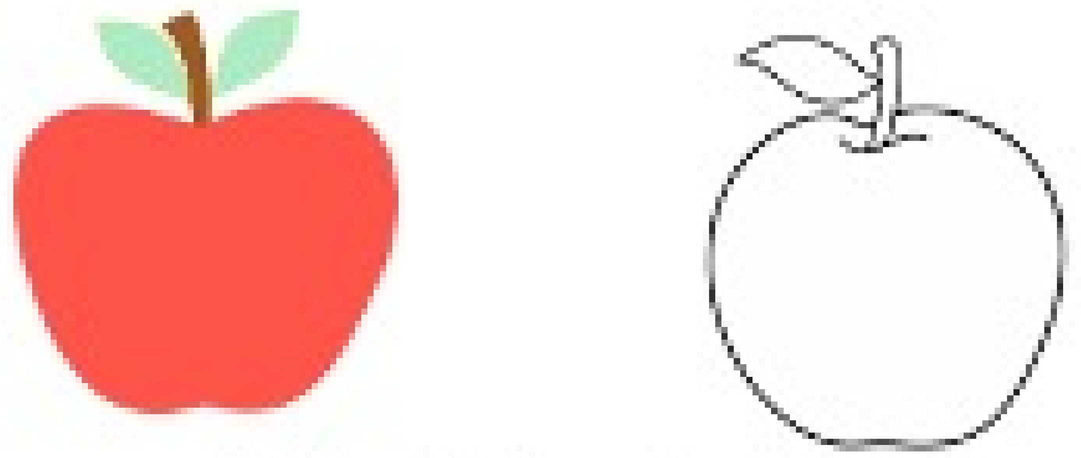
This thing is an apple.

Unsupervised Learning

Data: x
x is data, no labels!

Goal: Learn underlying structure

Apple example:



This thing is like the other thing.

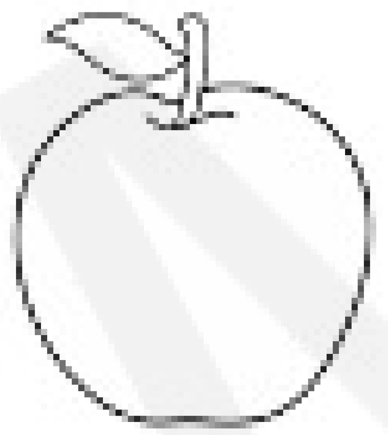
Classes of Learning Problems

Supervised Learning

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

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

Apple example:



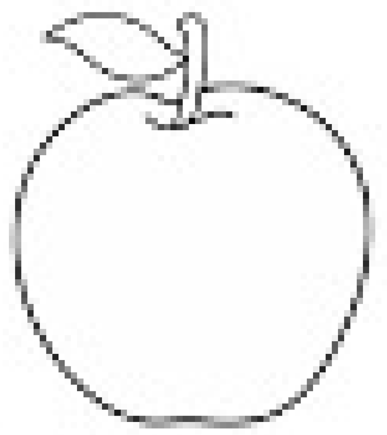
This thing is an apple.

Unsupervised Learning

Data: x
x is data, no labels!

Goal: Learn underlying structure

Apple example:



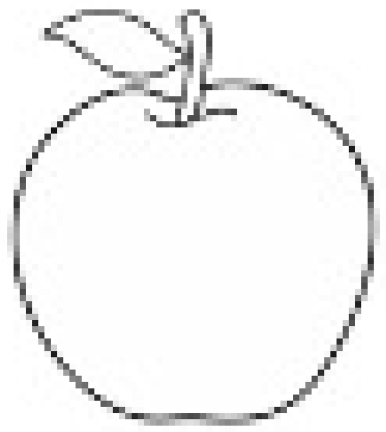
This thing is like the other thing.

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:

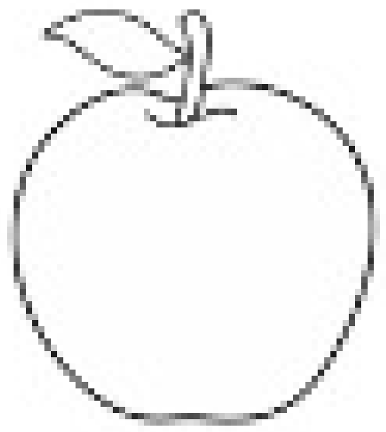
This thing is like the other thing.

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

Agent: takes actions.

Reinforcement Learning (RL): Key Concepts

AGENT

ENVIRONMENT

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

Total Reward (Return)

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

Reinforcement Learning (RL): Key Concepts

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

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 $\pi(s)$**, 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 $Q(s,a)$

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

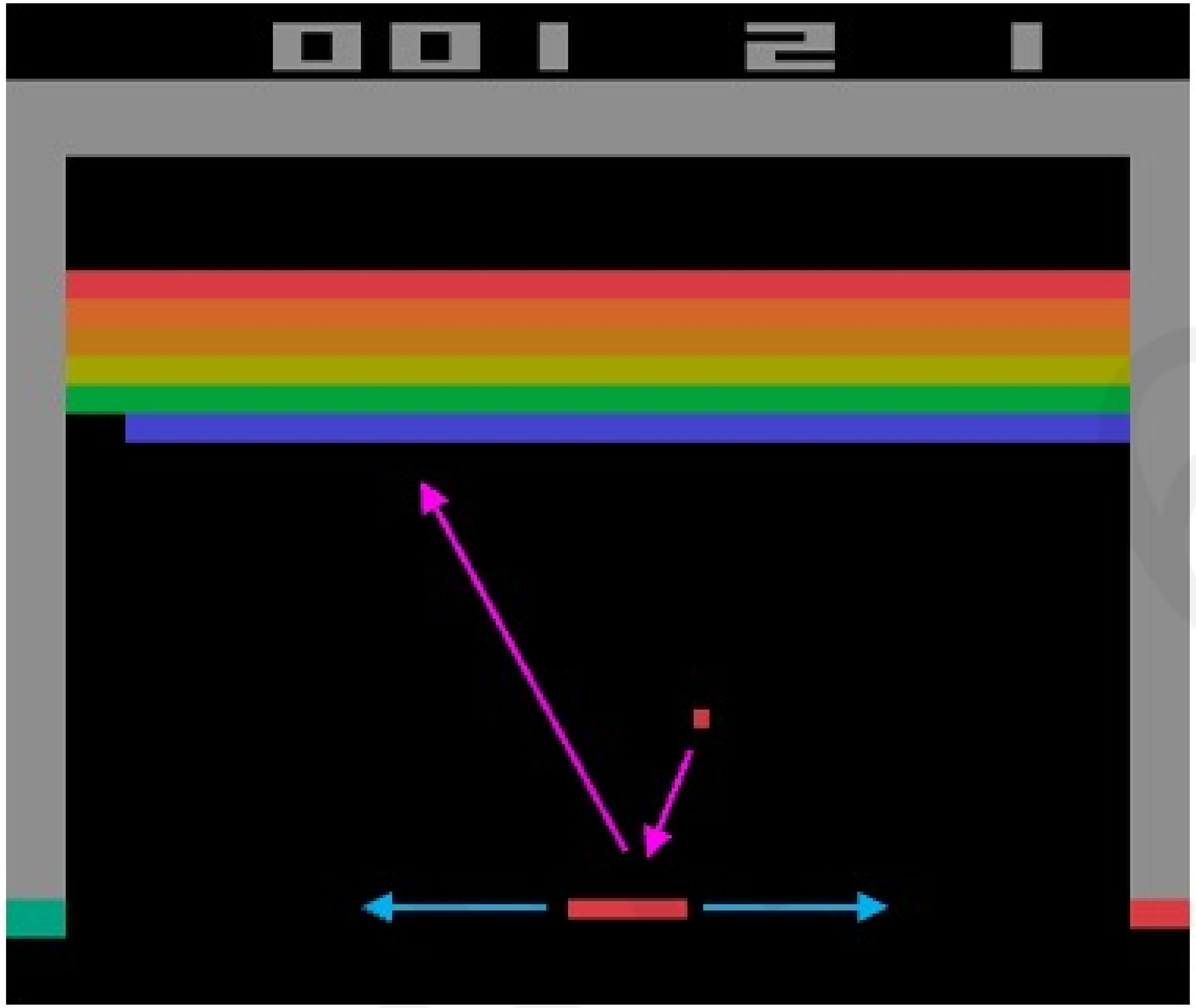
Policy Learning

Find $\pi(s)$

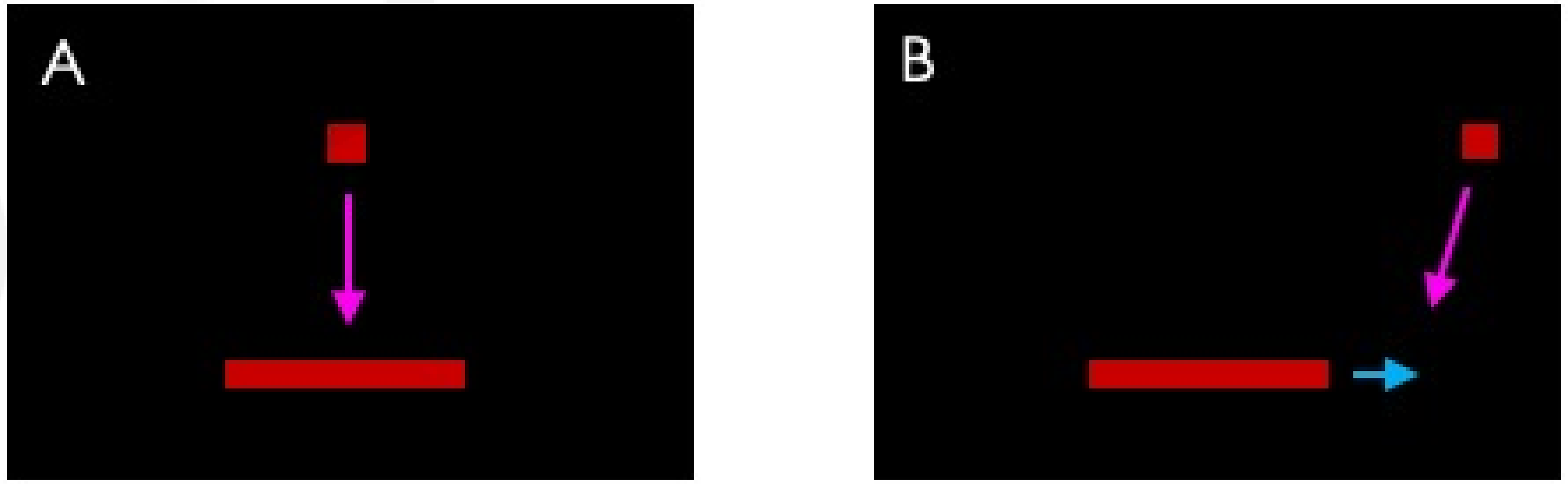
Sample $a \sim \pi(s)$

Digging deeper into the Q-function

Example: Atari Breakout



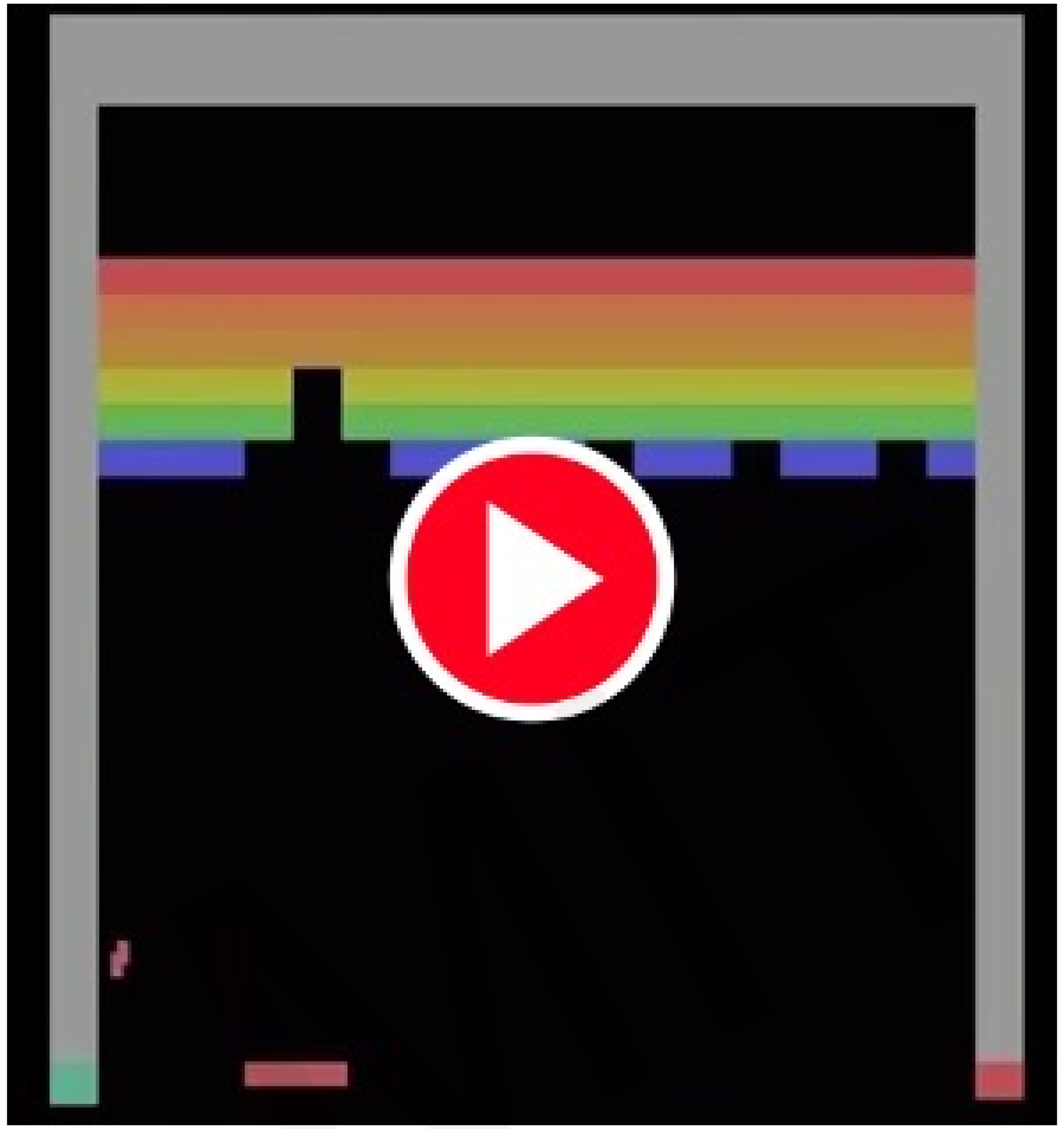
It can be very difficult for humans to accurately estimate Q-values



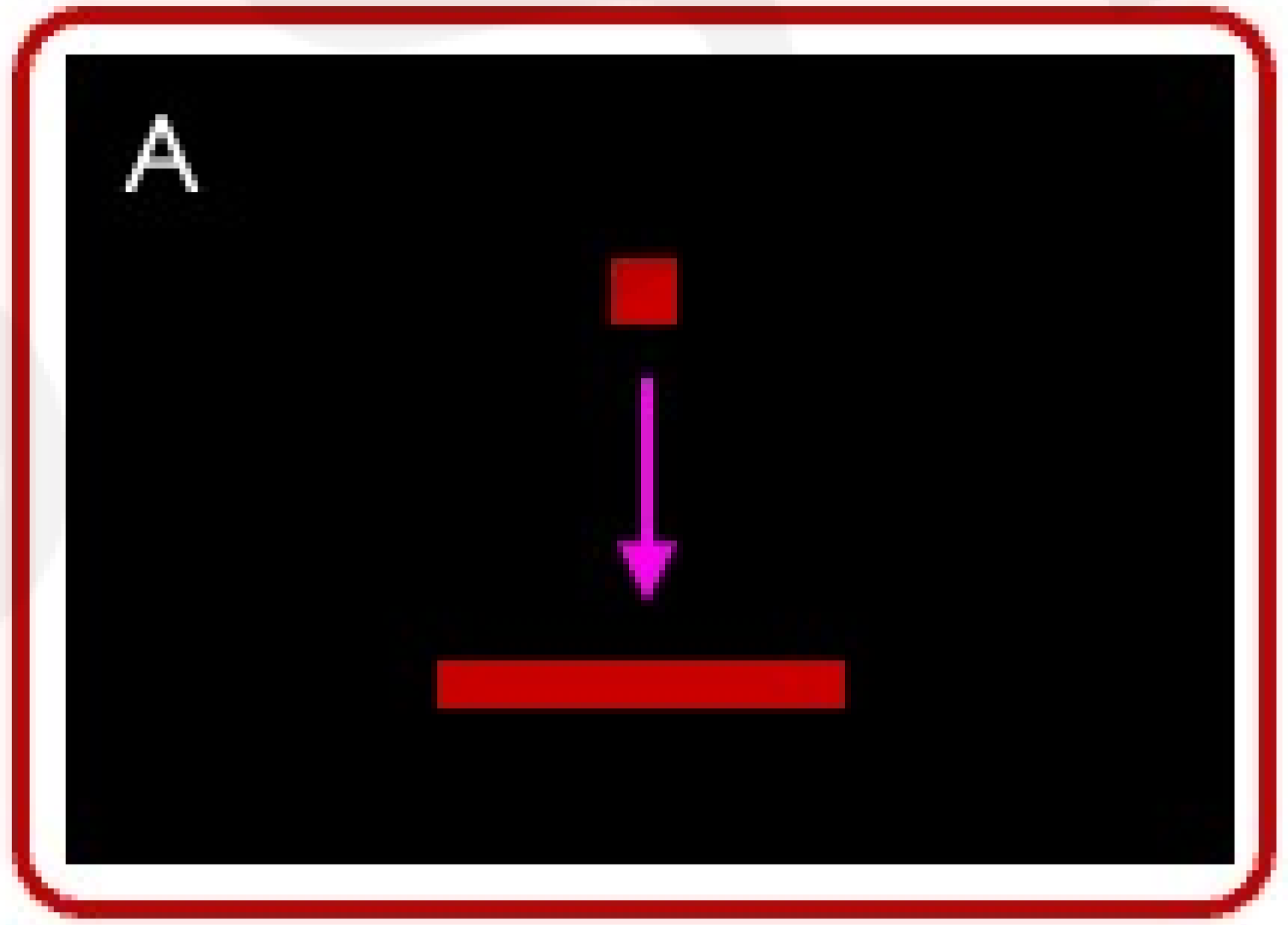
Which (s, a) pair has a higher Q-value?

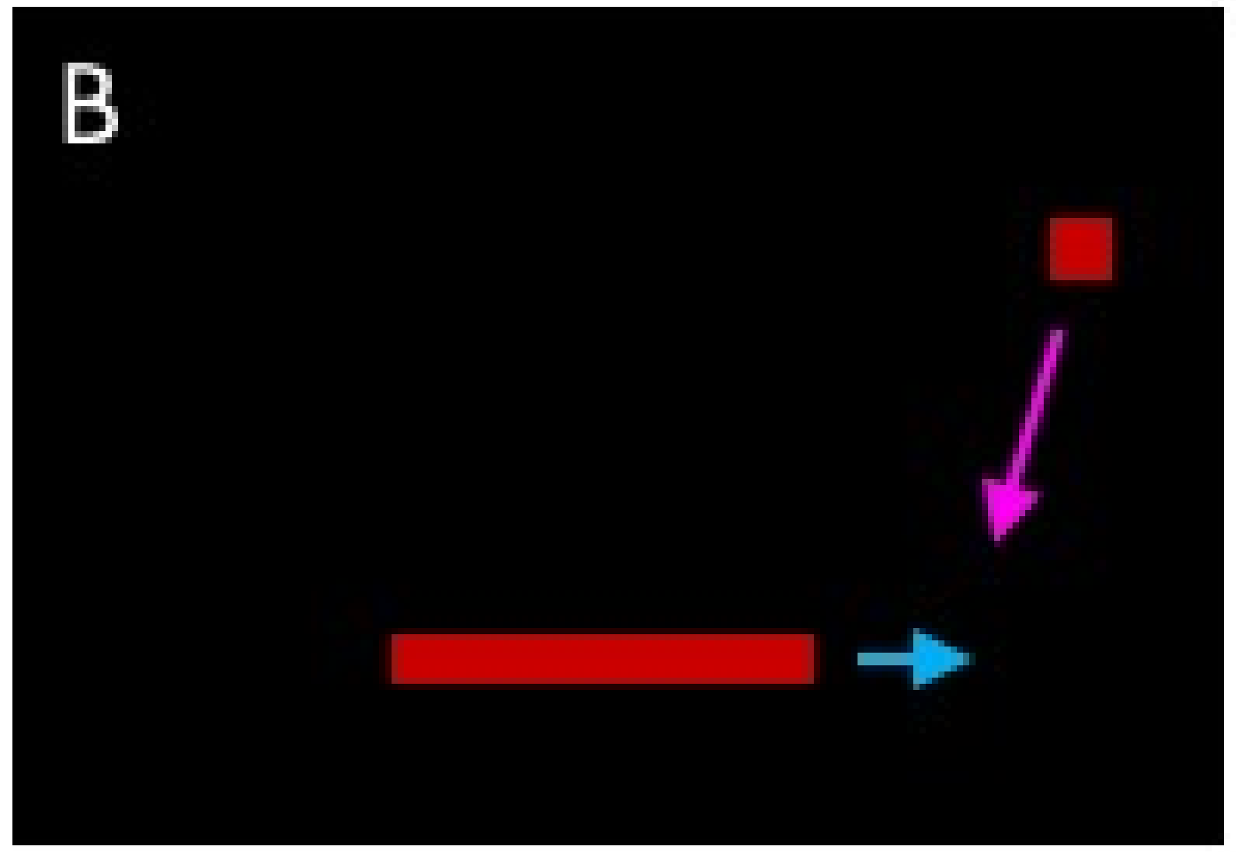
Digging deeper into the Q-function

Example: Atari Breakout - Middle



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



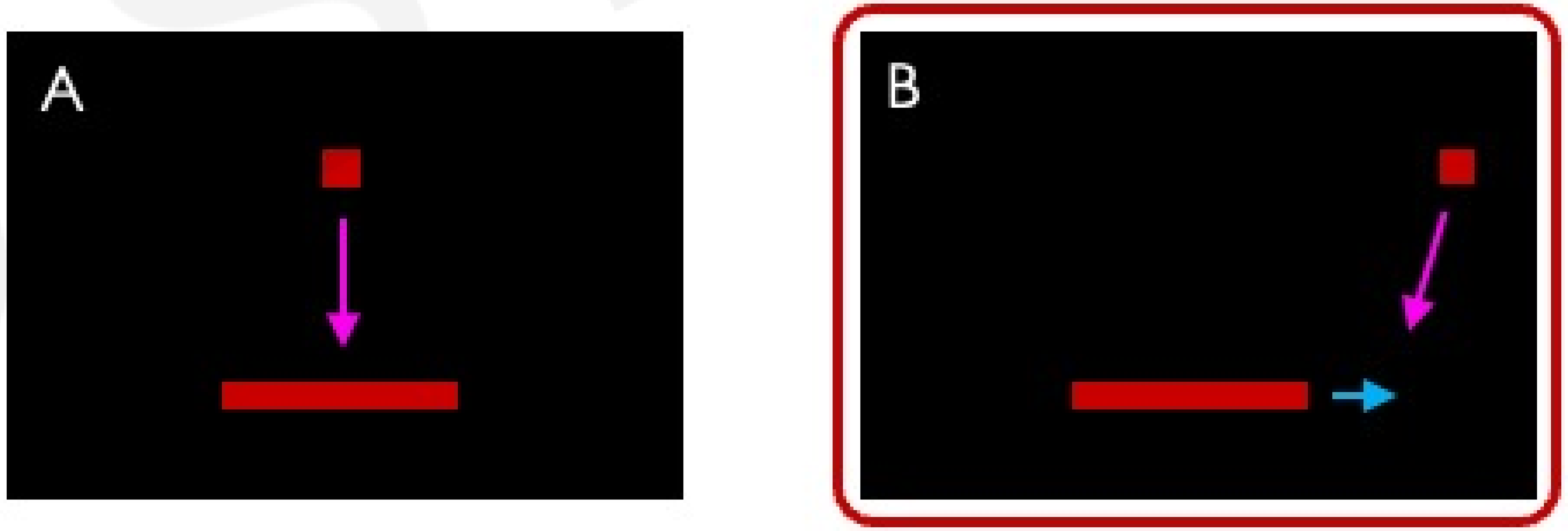


Which $(\mathbf{s}, \mathbf{a})$ pair has a higher Q-value?

Digging deeper into the Q-function

Example: Atari Breakout - Side

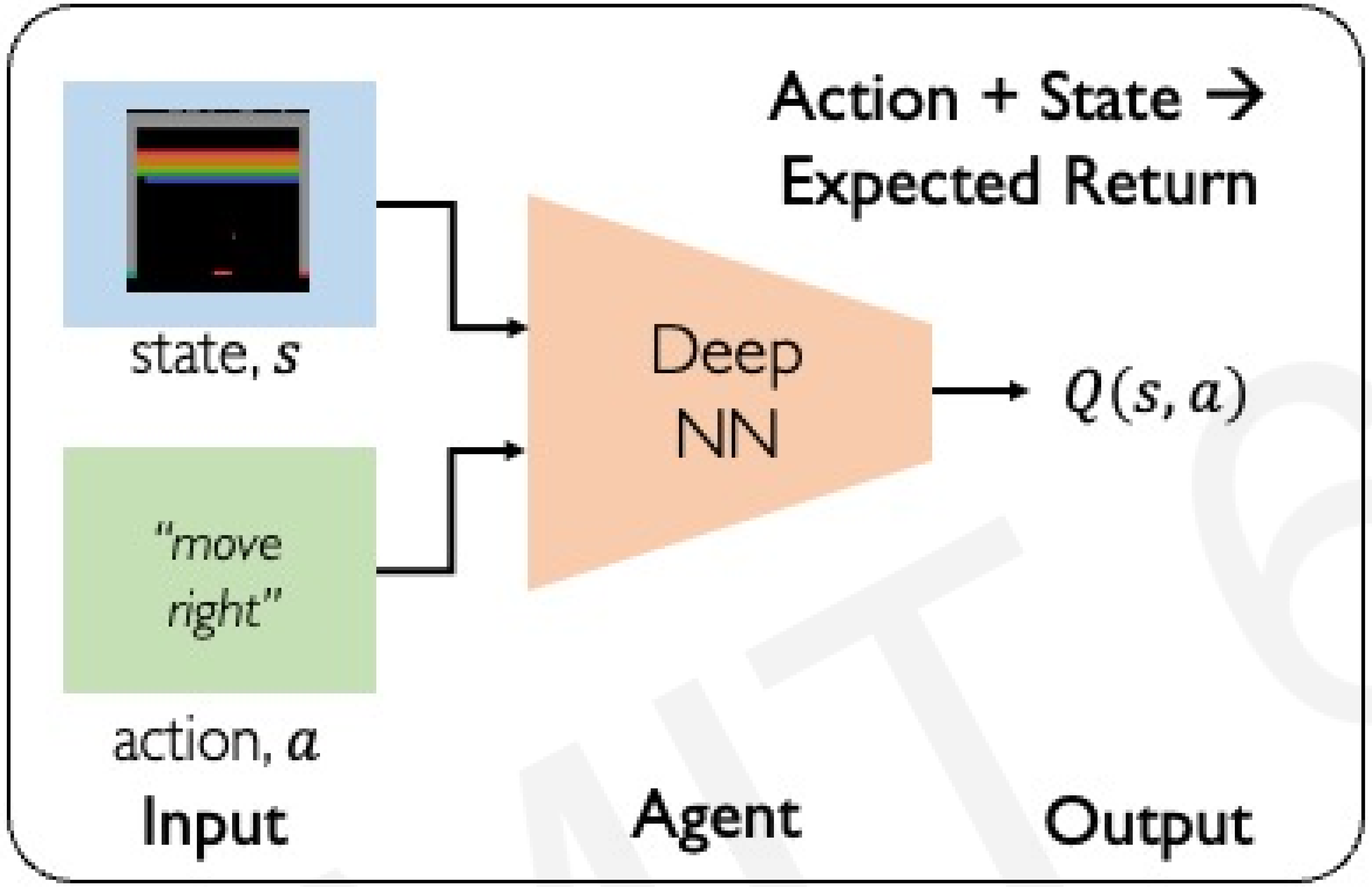
It can be very difficult for humans to accurately estimate Q-values



Which $(\mathbf{s}, \mathbf{a})$ pair has a higher Q-value?

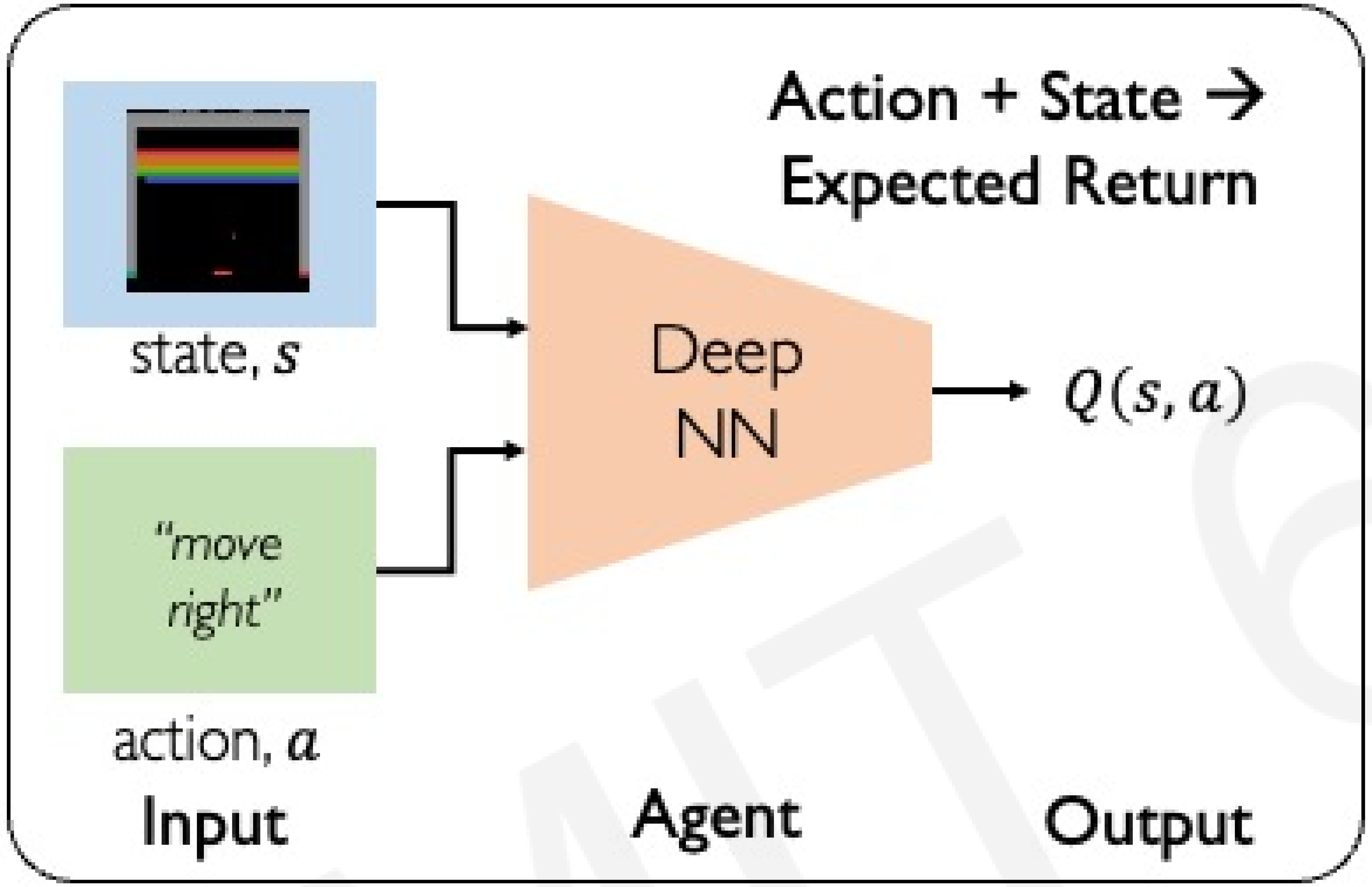
Deep Q Networks (DQN)

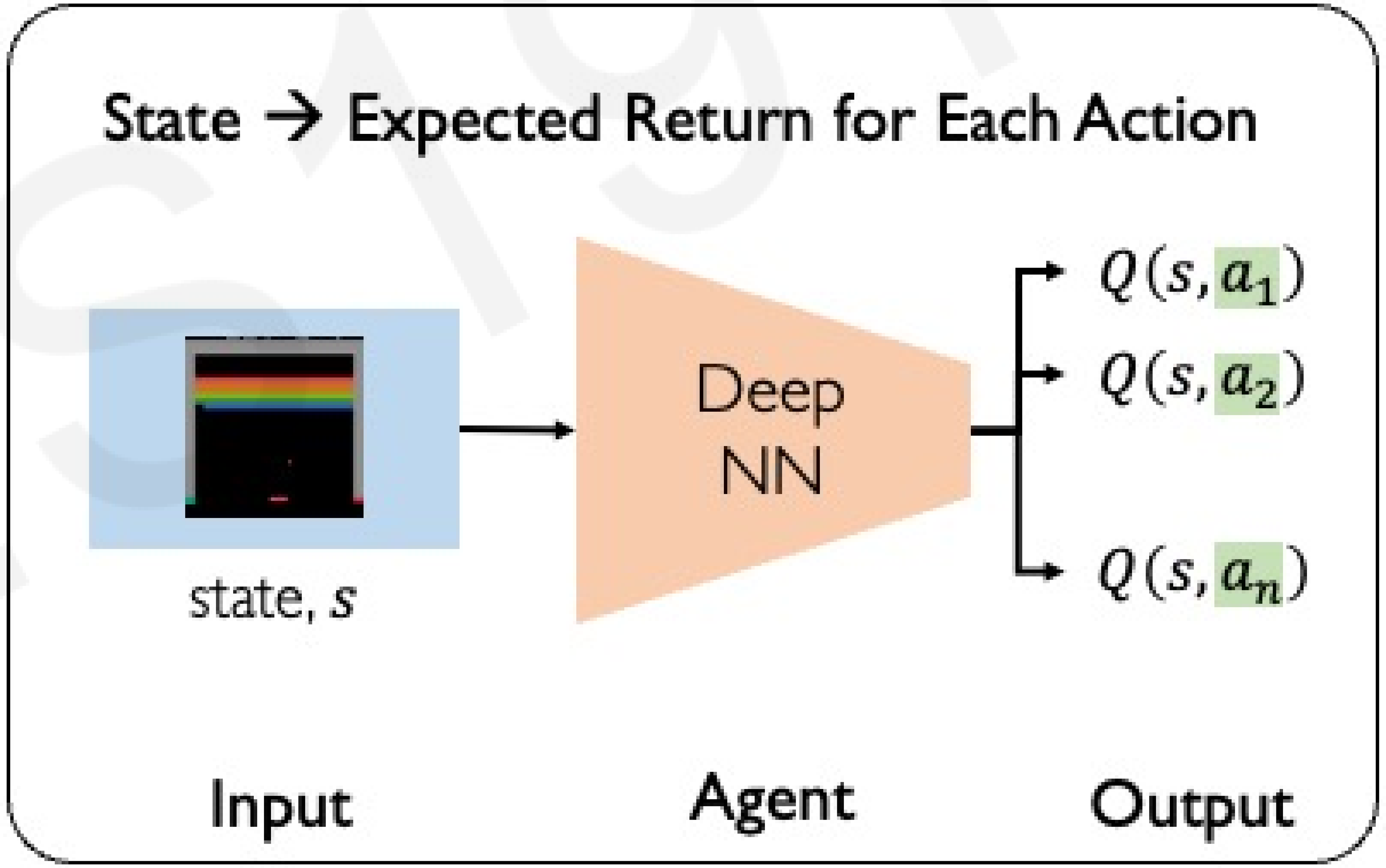
How can we use deep neural networks to model Q-functions?



Deep Q Networks (DQN)

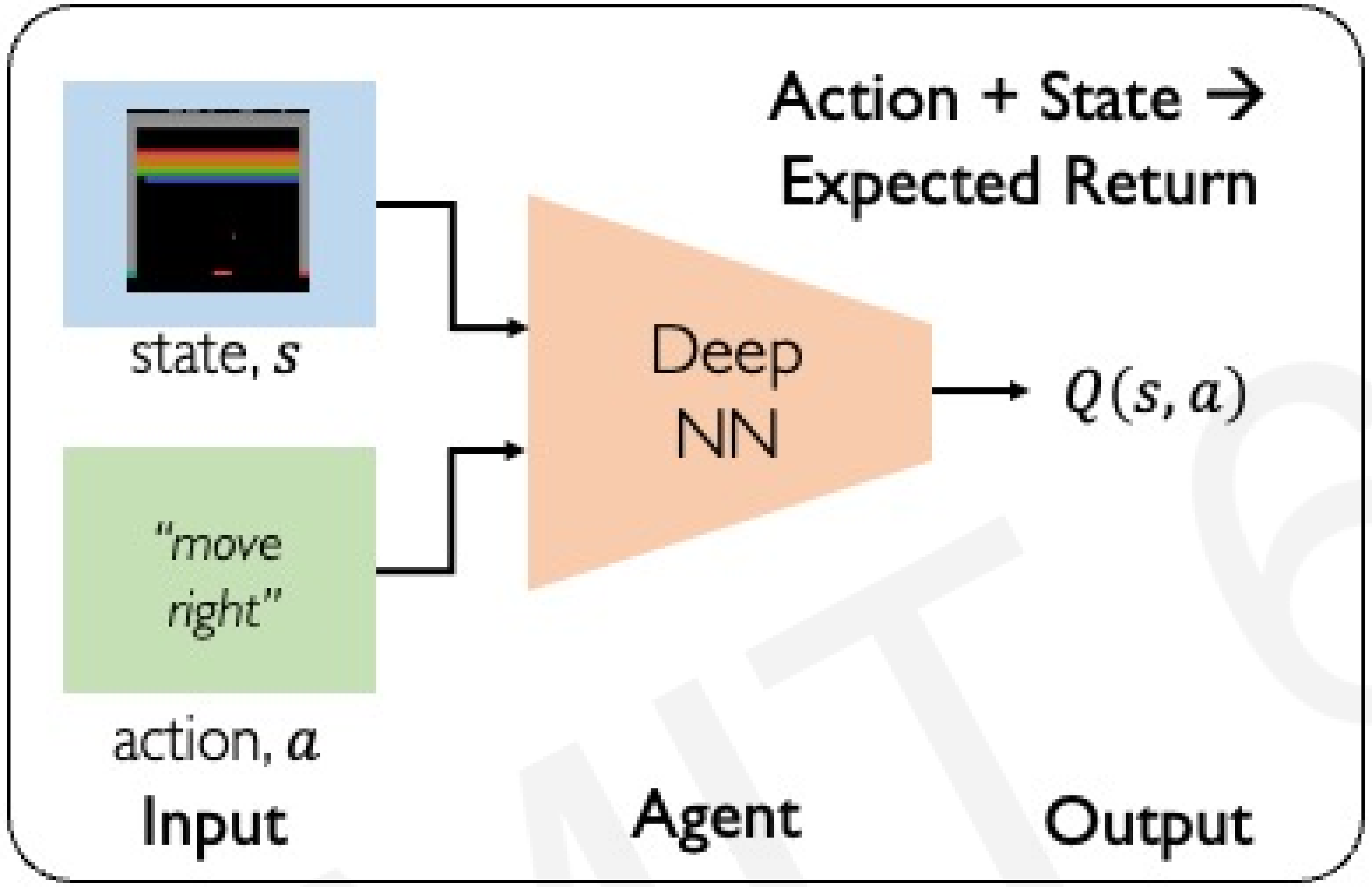
How can we use deep neural networks to model Q-functions?

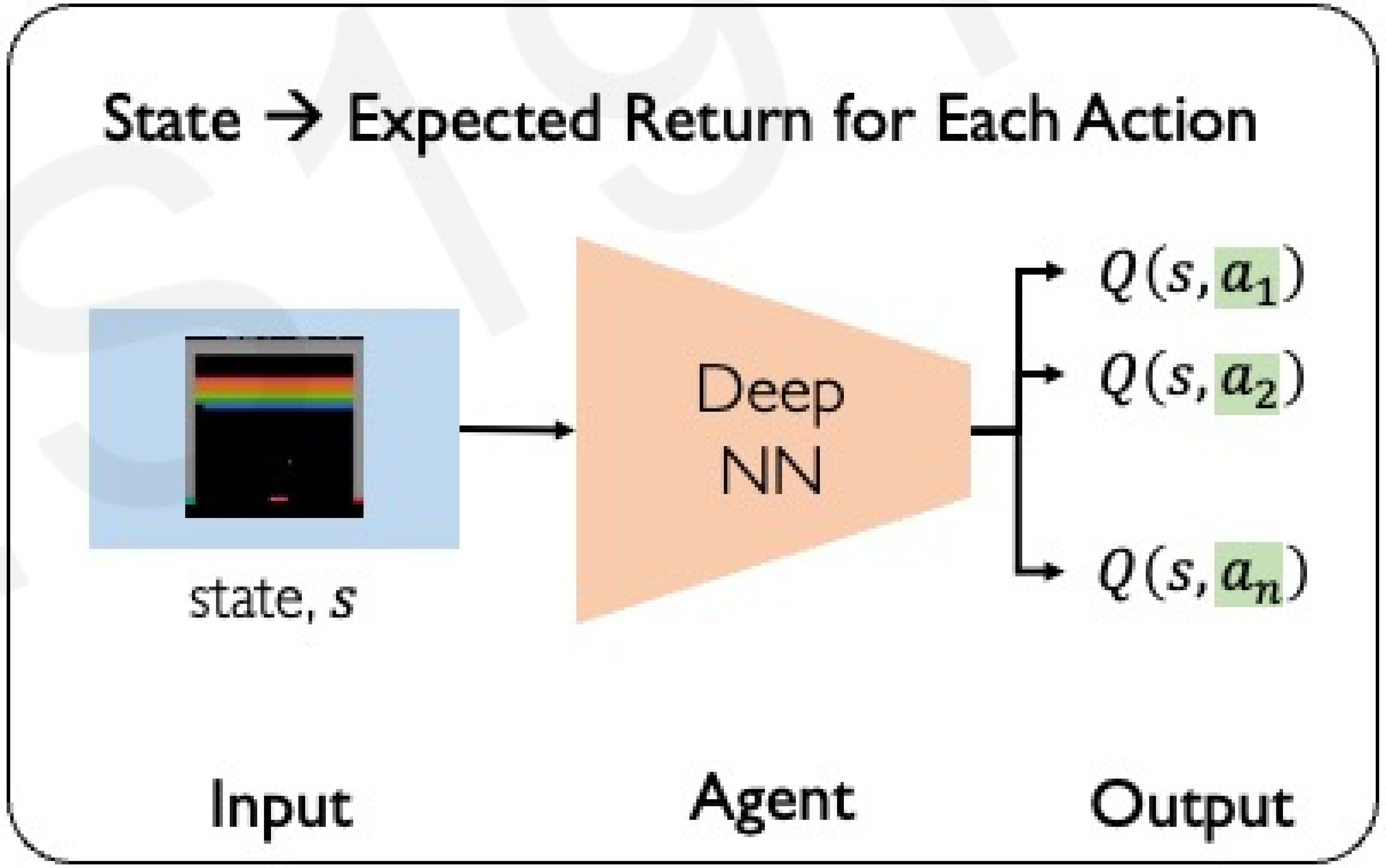




Deep Q Networks (DQN): Training

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

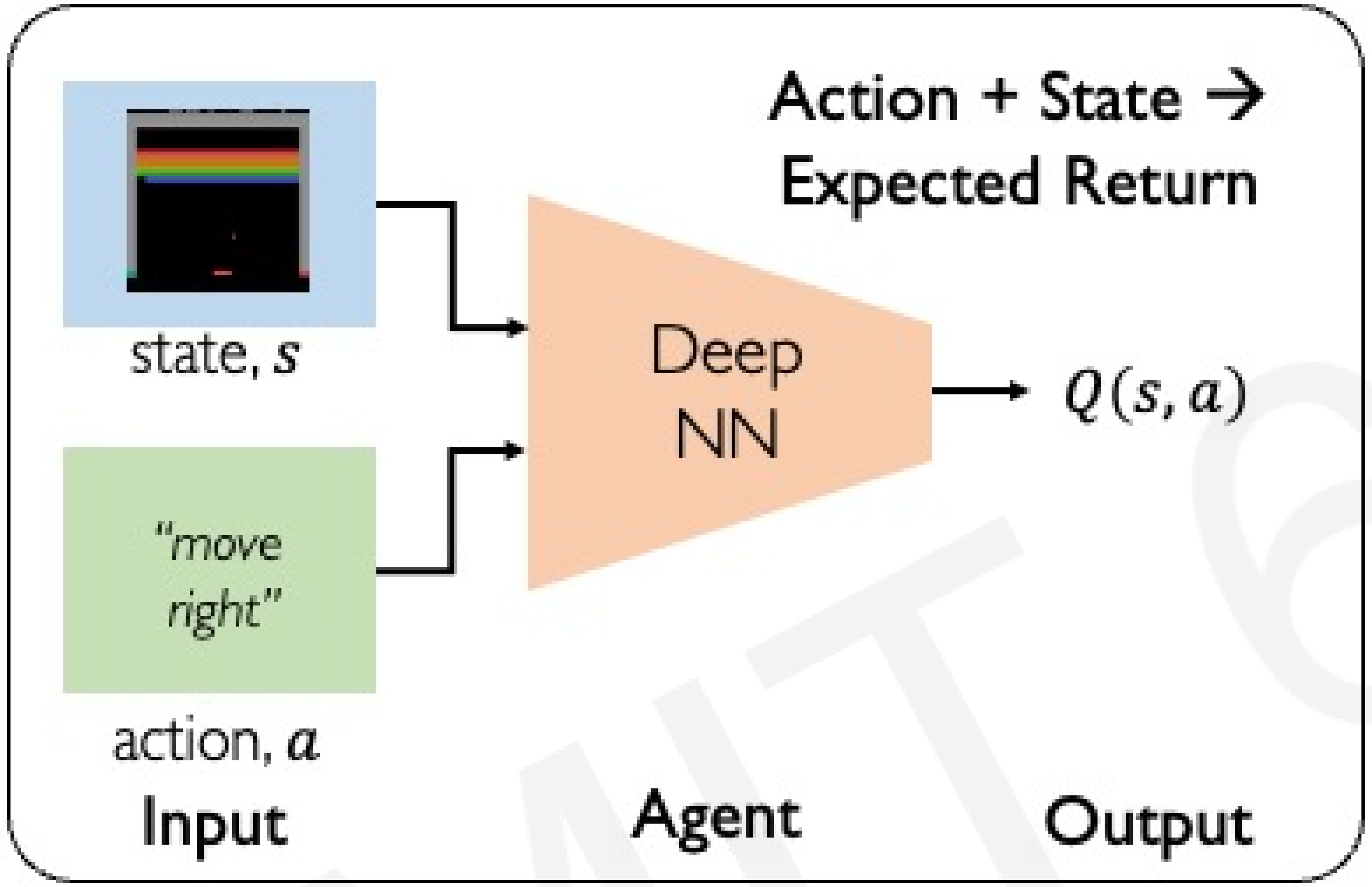


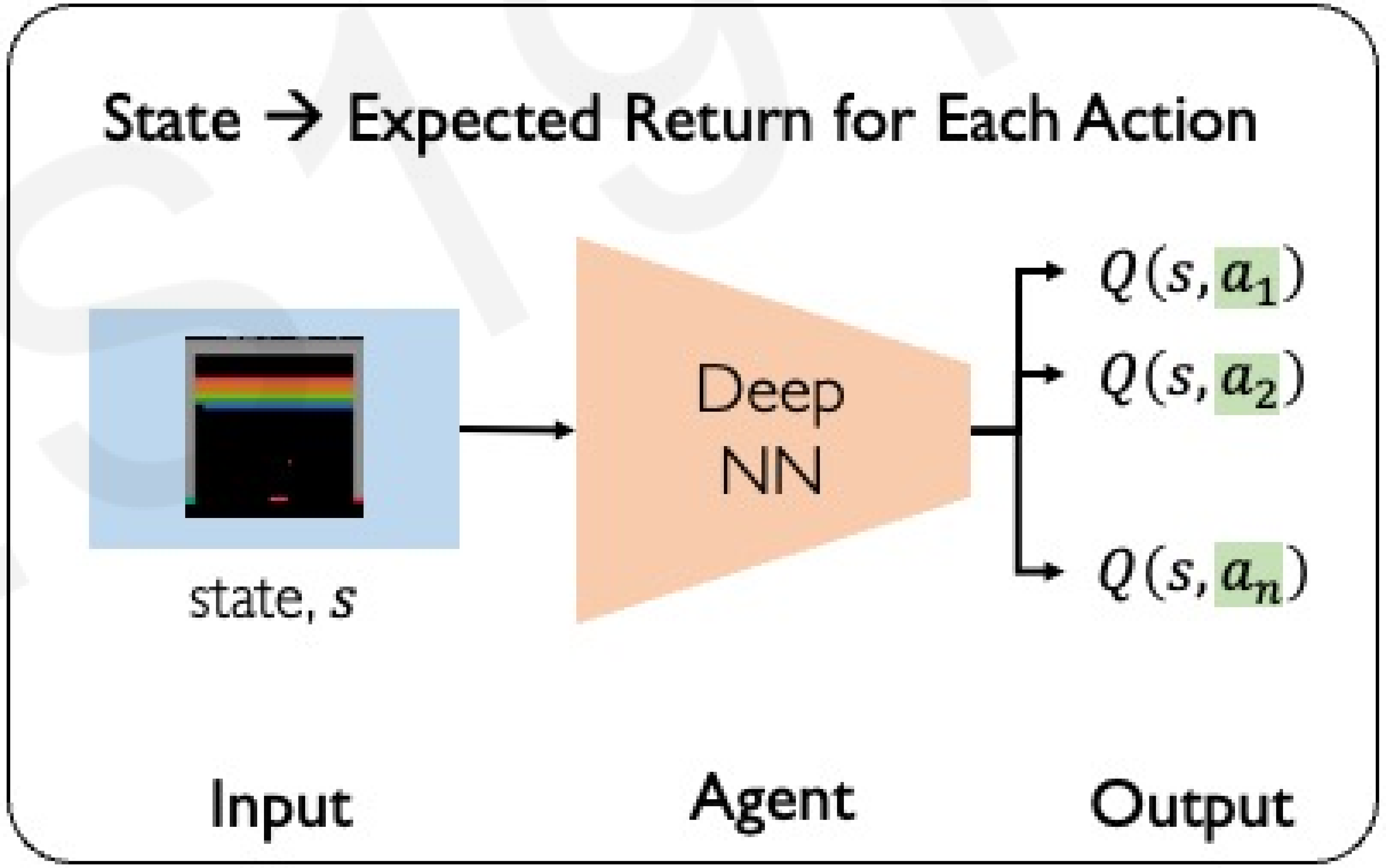


What happens if we take all the best actions?
Maximize target return → train the agent

Deep Q Networks (DQN): Training

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



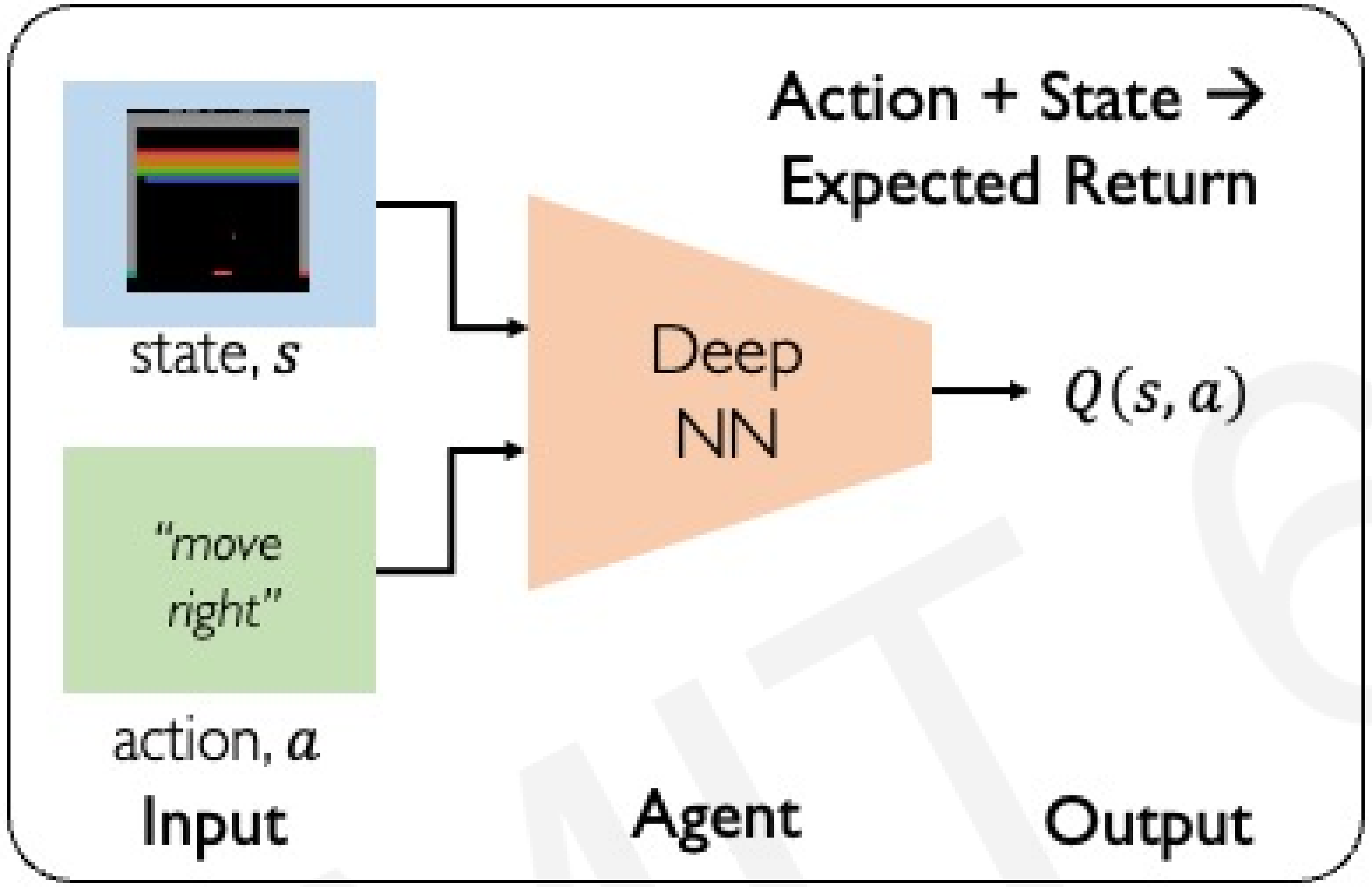


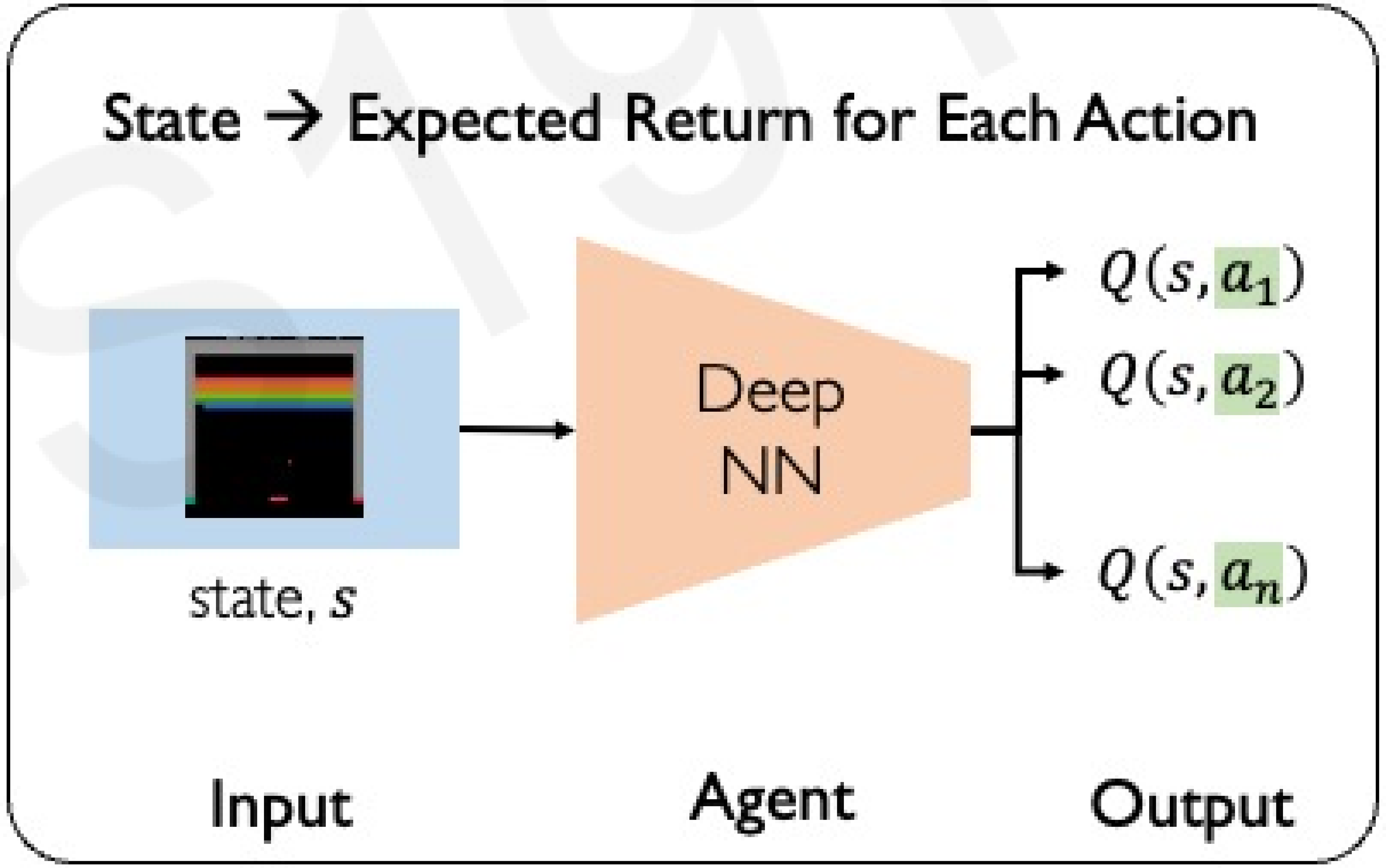
$$\overbrace{\left(r + \gamma \max_{a'} Q(s', a')\right)}^{\text{target}}$$

Take all the best actions → target return

Deep Q Networks (DQN): Training

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



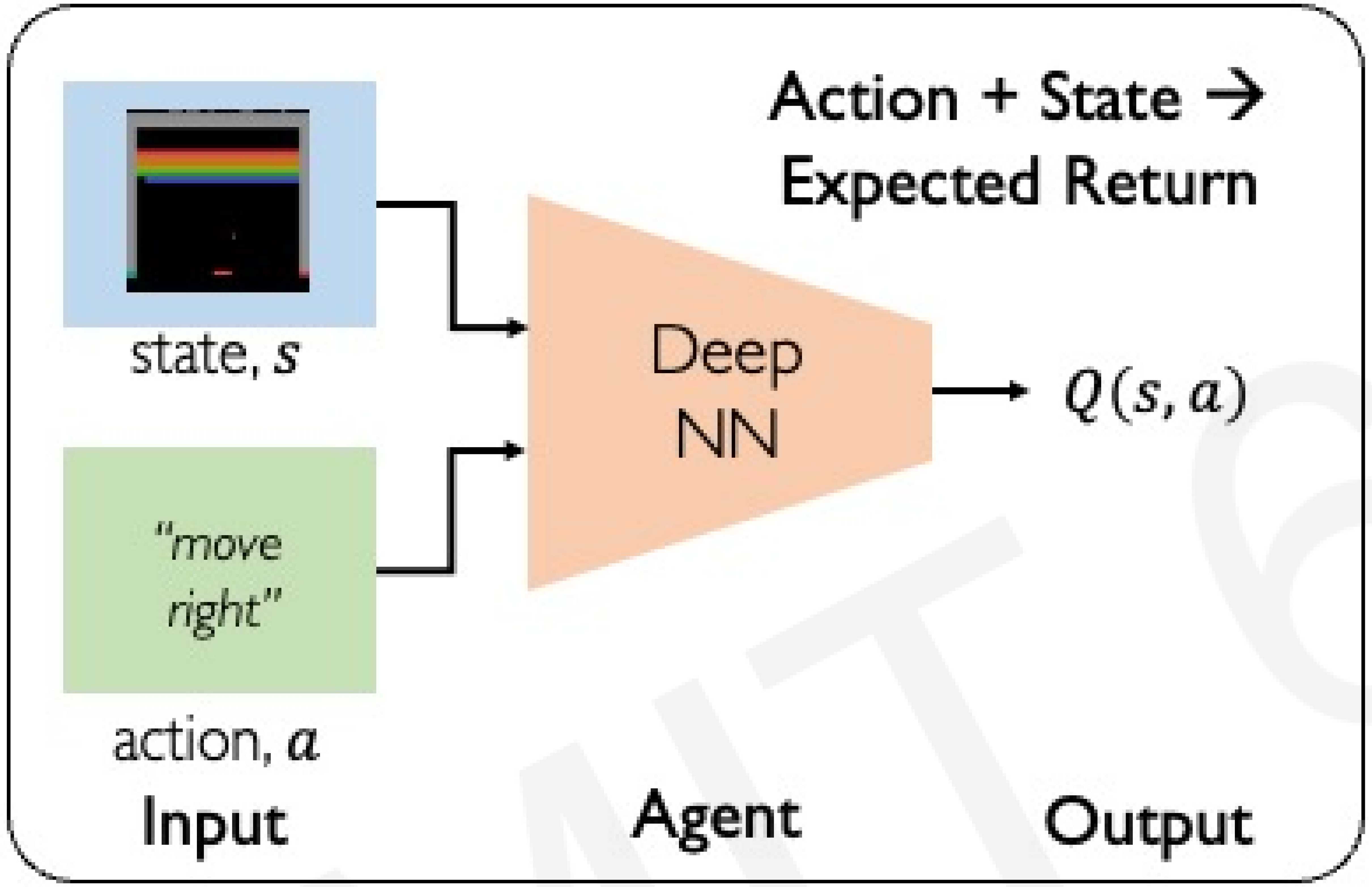


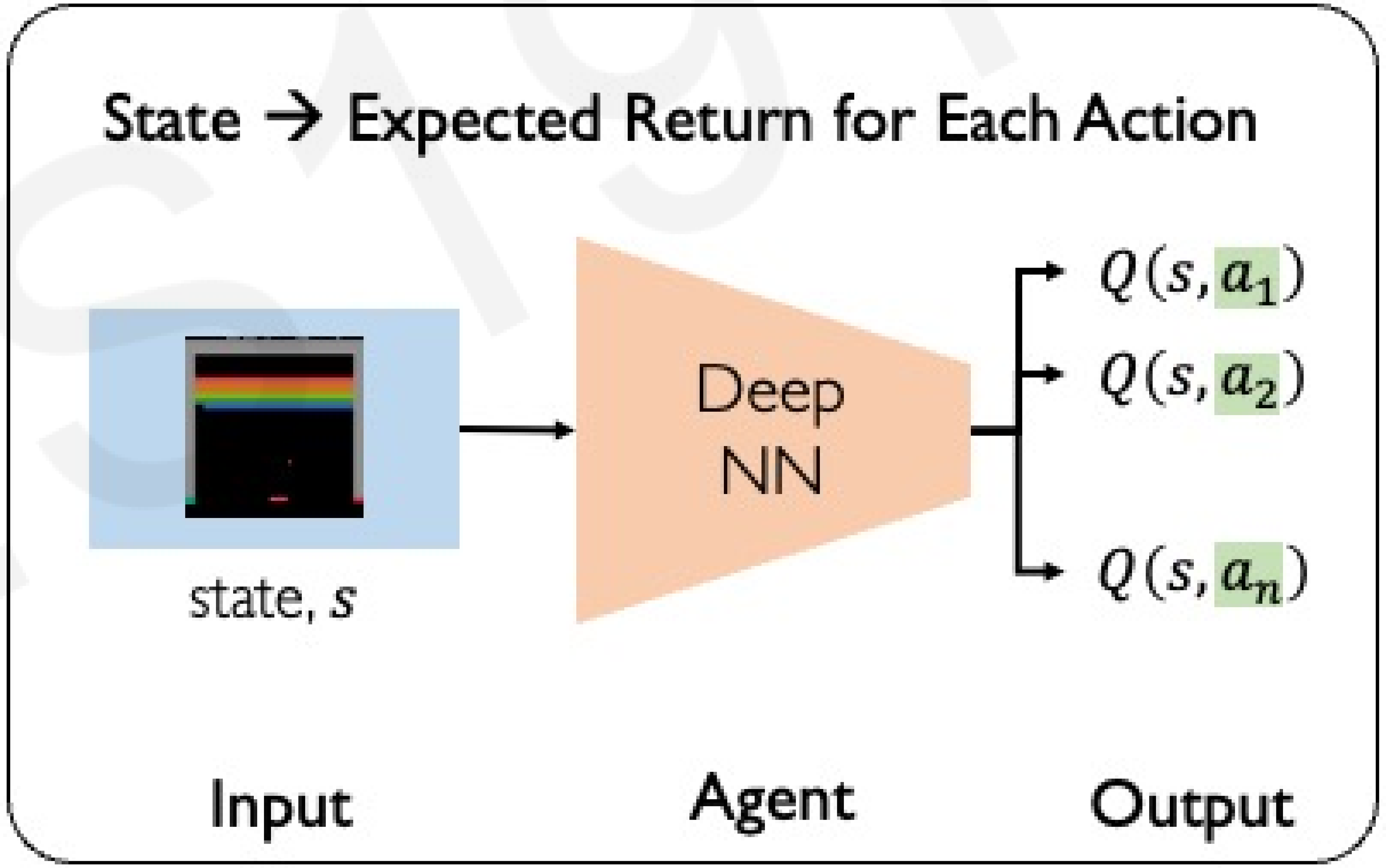
$$\underbrace{\left(r + \gamma \max_{a'} Q(s', a')\right)}_{\text{target}} \quad \underbrace{Q(s,a)}_{\text{predicted}}$$

Network prediction

Deep Q Networks (DQN): Training

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



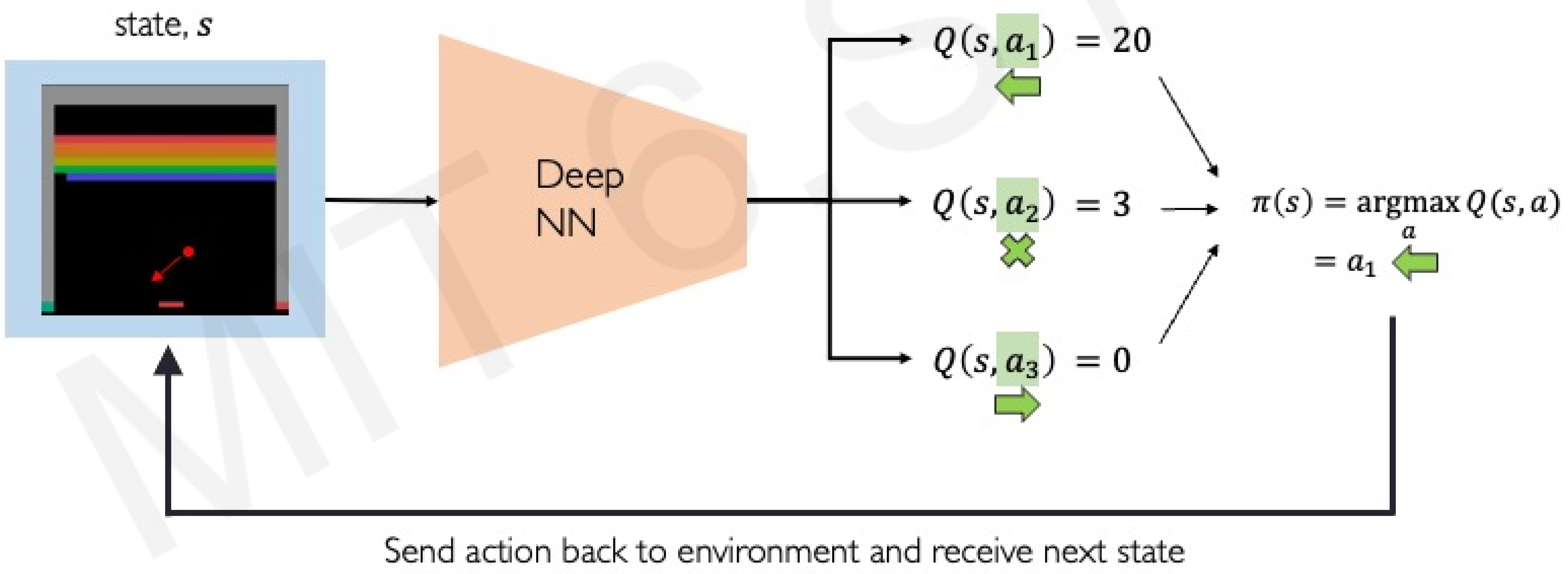


$$\mathcal{L} = \mathbb{E}\left[\left\| \overbrace{\left(r + \gamma \max_{a'} Q(s', a')\right)}^{\text{target}} - \overbrace{Q(s, a)}^{\text{predicted}} \right\|^2\right]$$

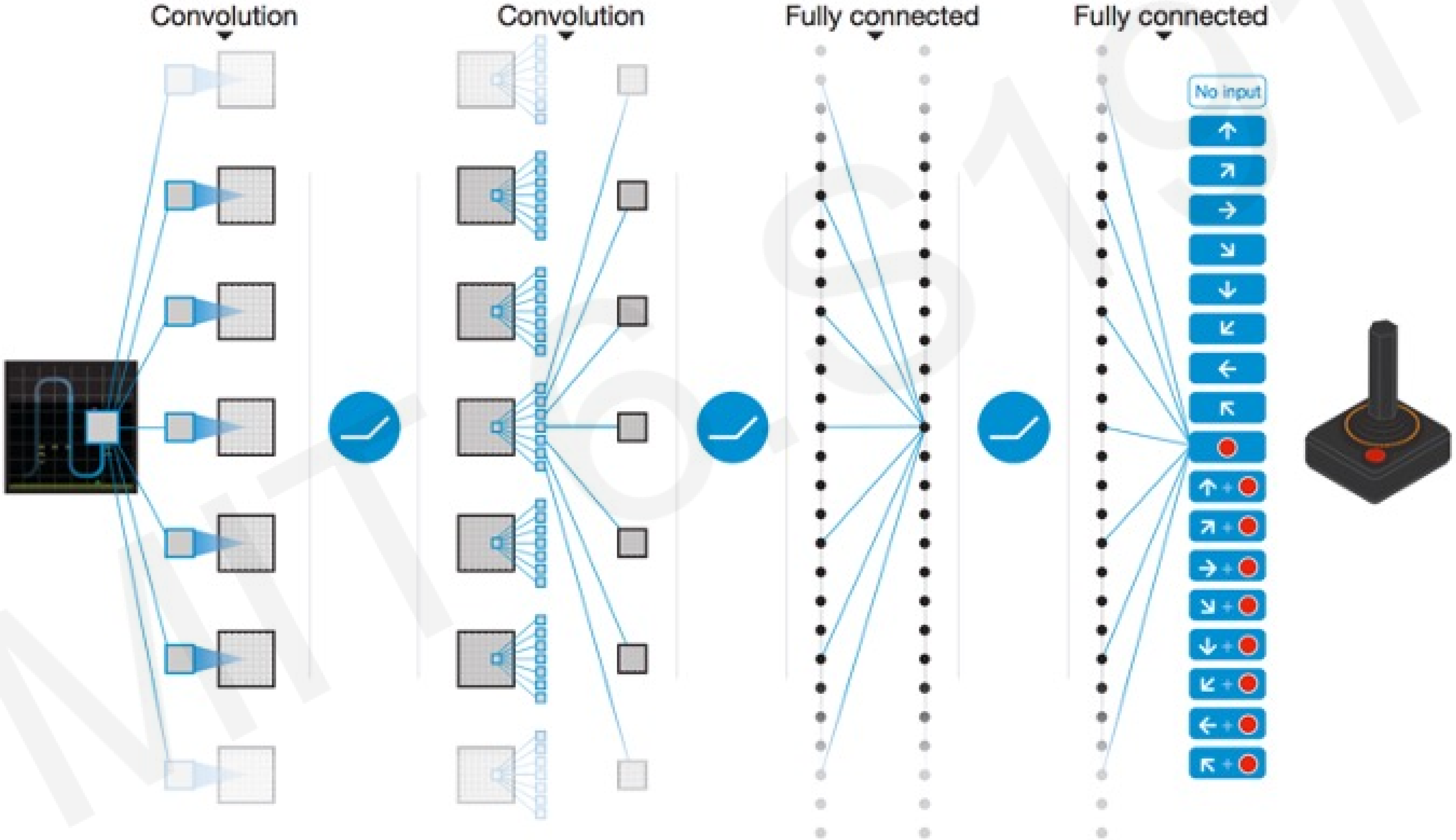
Q-Loss

Deep Q Network Summary

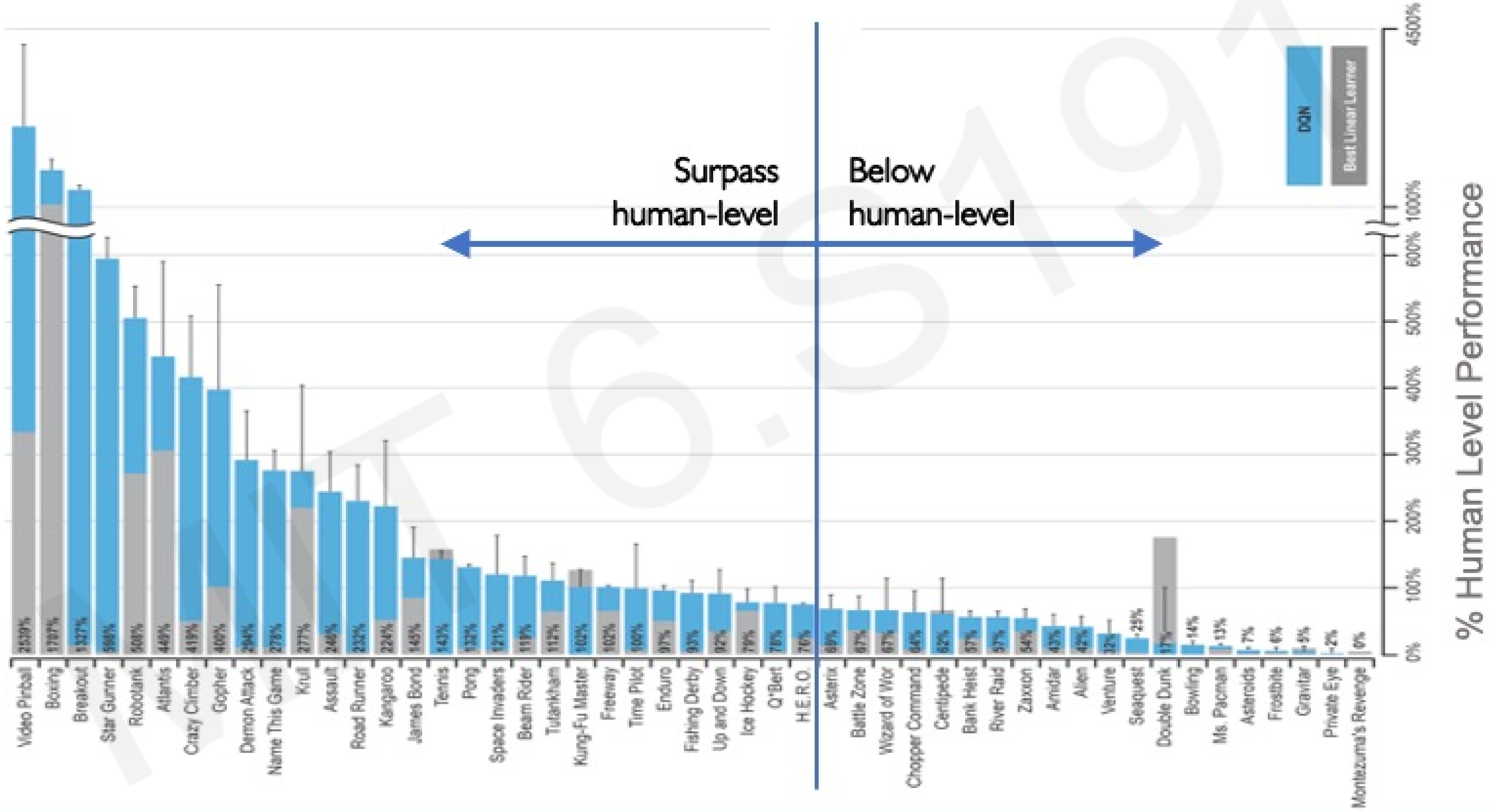
Use NN to learn Q-function and then use to infer the optimal policy, $\pi(s)$



DQN Atari Results



DQN Atari Results



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 $Q(s,a)$

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

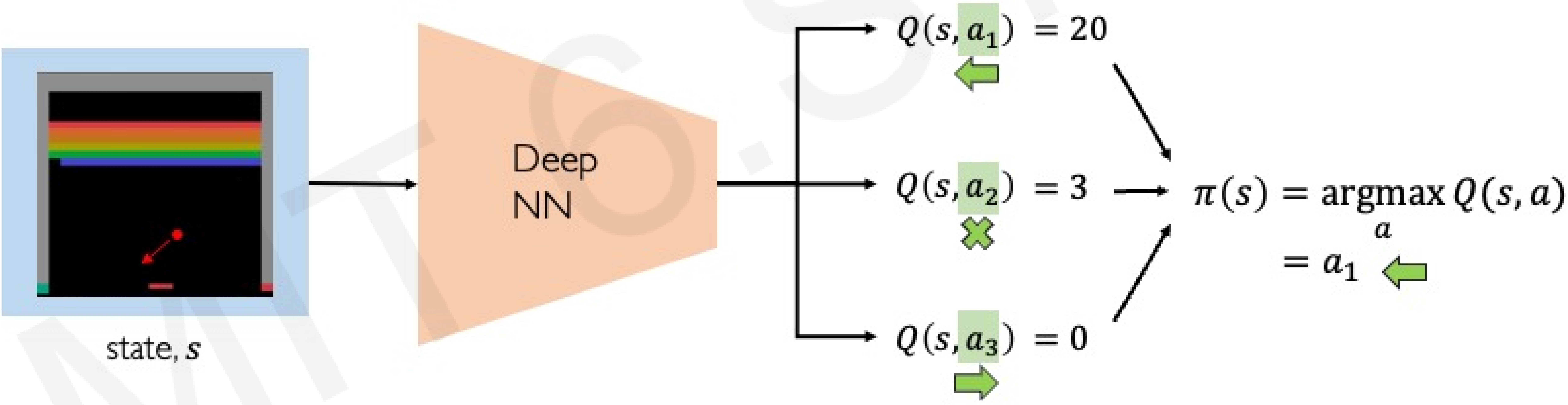
Policy Learning

Find $\pi(s)$

Sample $a \sim \pi(s)$

Deep Q Networks (DQN)

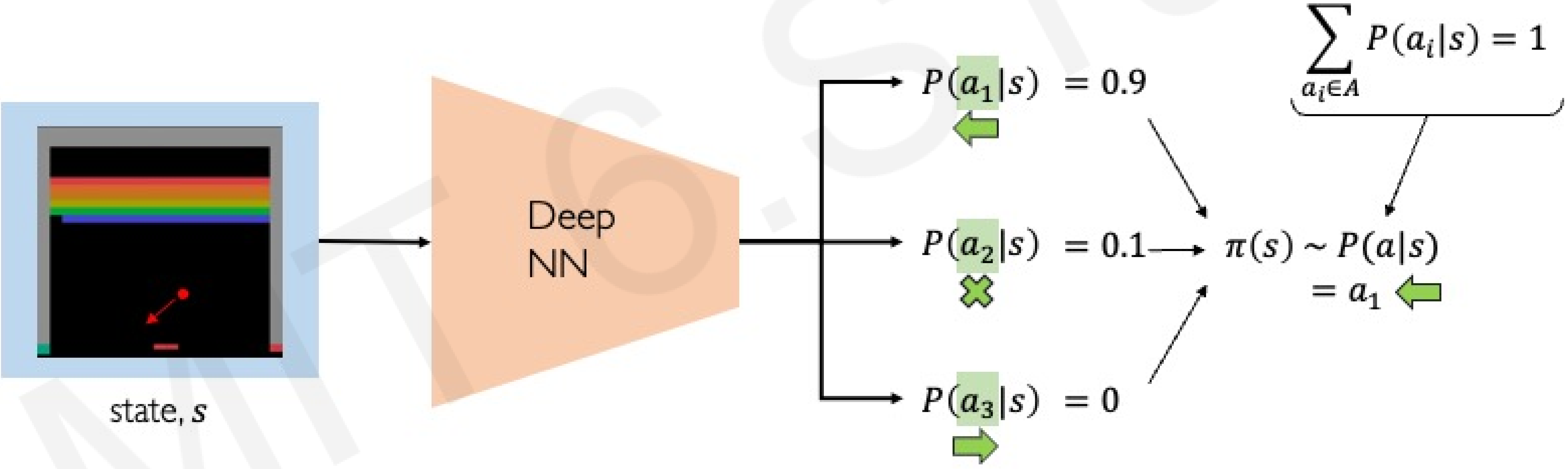
DQN: Approximate Q-function and use to infer the optimal policy, $\pi(s)$



Policy Gradient (PG): Key Idea

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

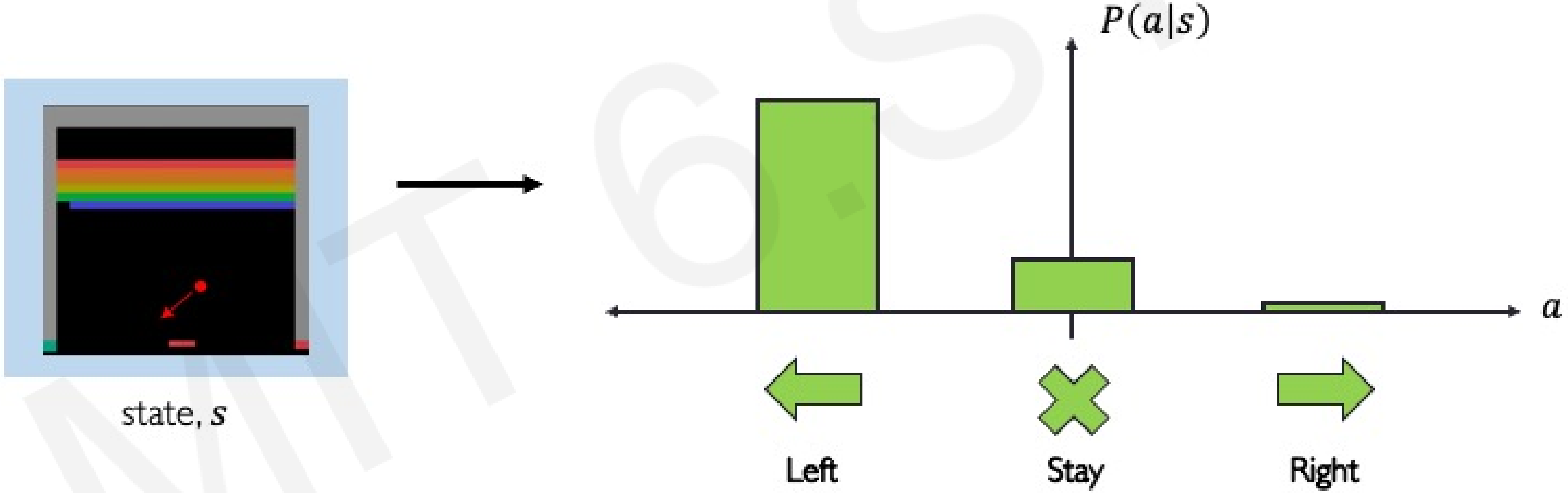
Policy Gradient: Directly optimize the policy $\pi(s)$



What are some advantages of this formulation?

Discrete vs Continuous Action Spaces

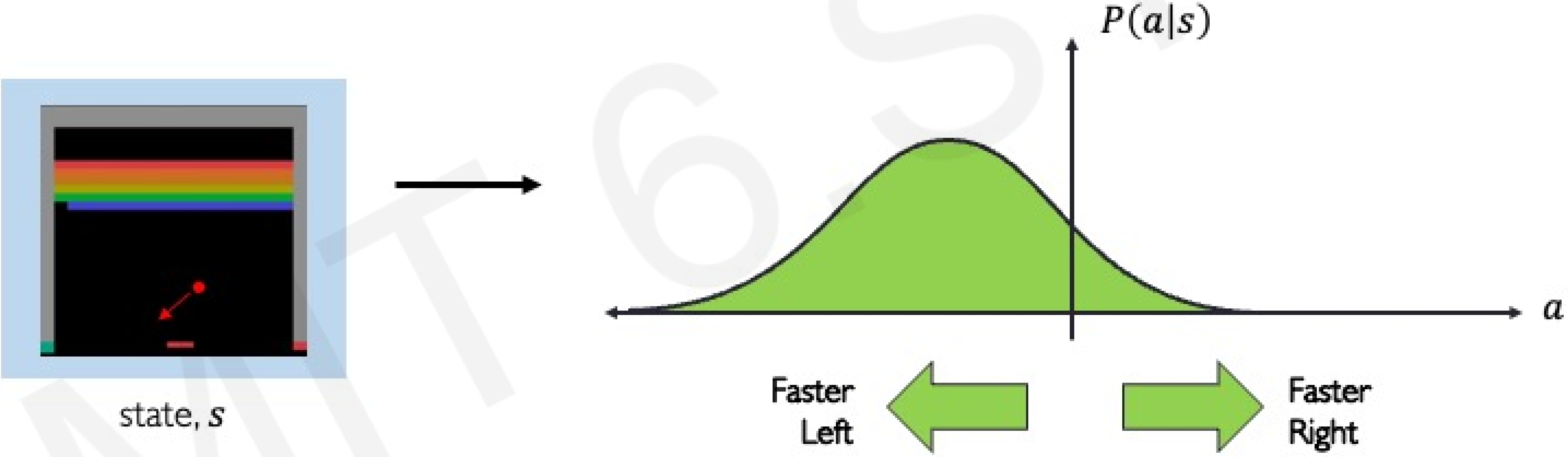
Discrete action space: which direction should I move?



Discrete vs Continuous Action Spaces

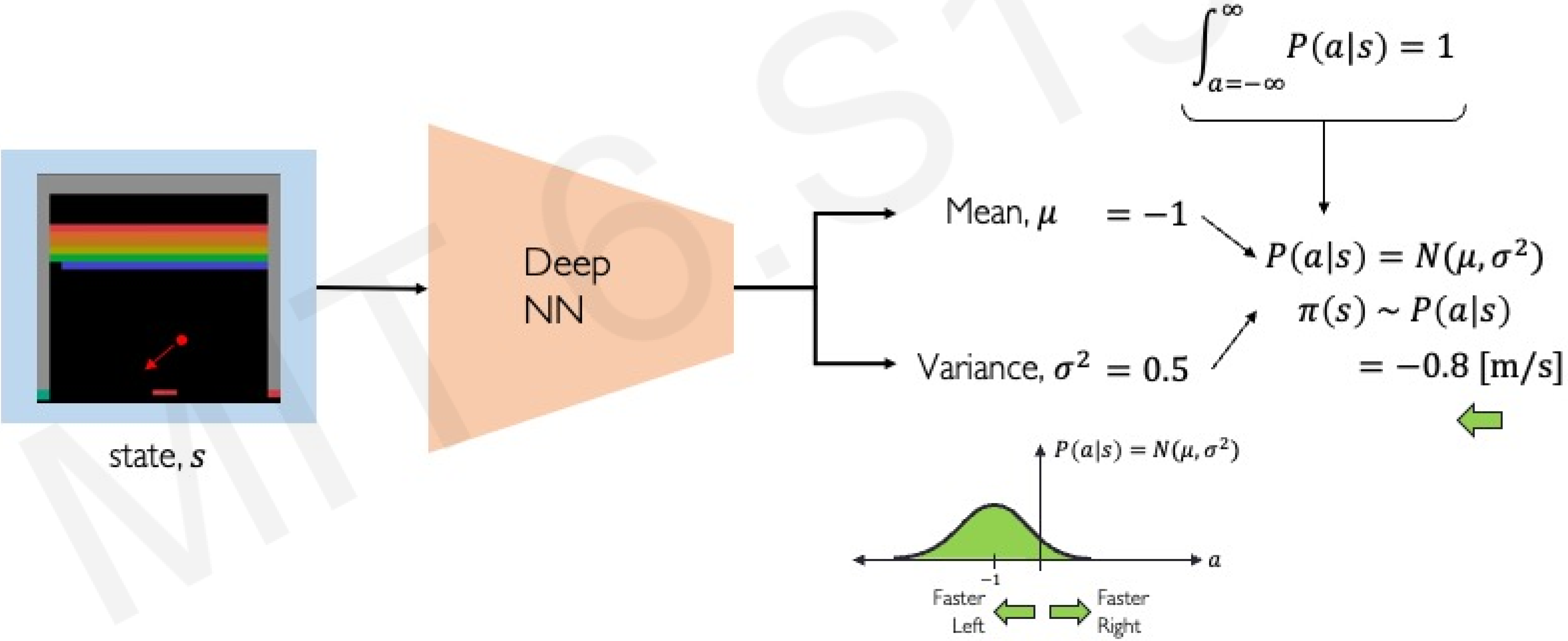
Discrete action space: which direction should I move?

Continuous action space: how fast should I move?



Policy Gradient (PG): Key Idea

Policy Gradient: Enables modeling of continuous action space



Training Policy Gradients: Case Study

Reinforcement Learning Loop:

Case Study – Self-Driving Cars

Agent: vehicle

State: camera, lidar, etc

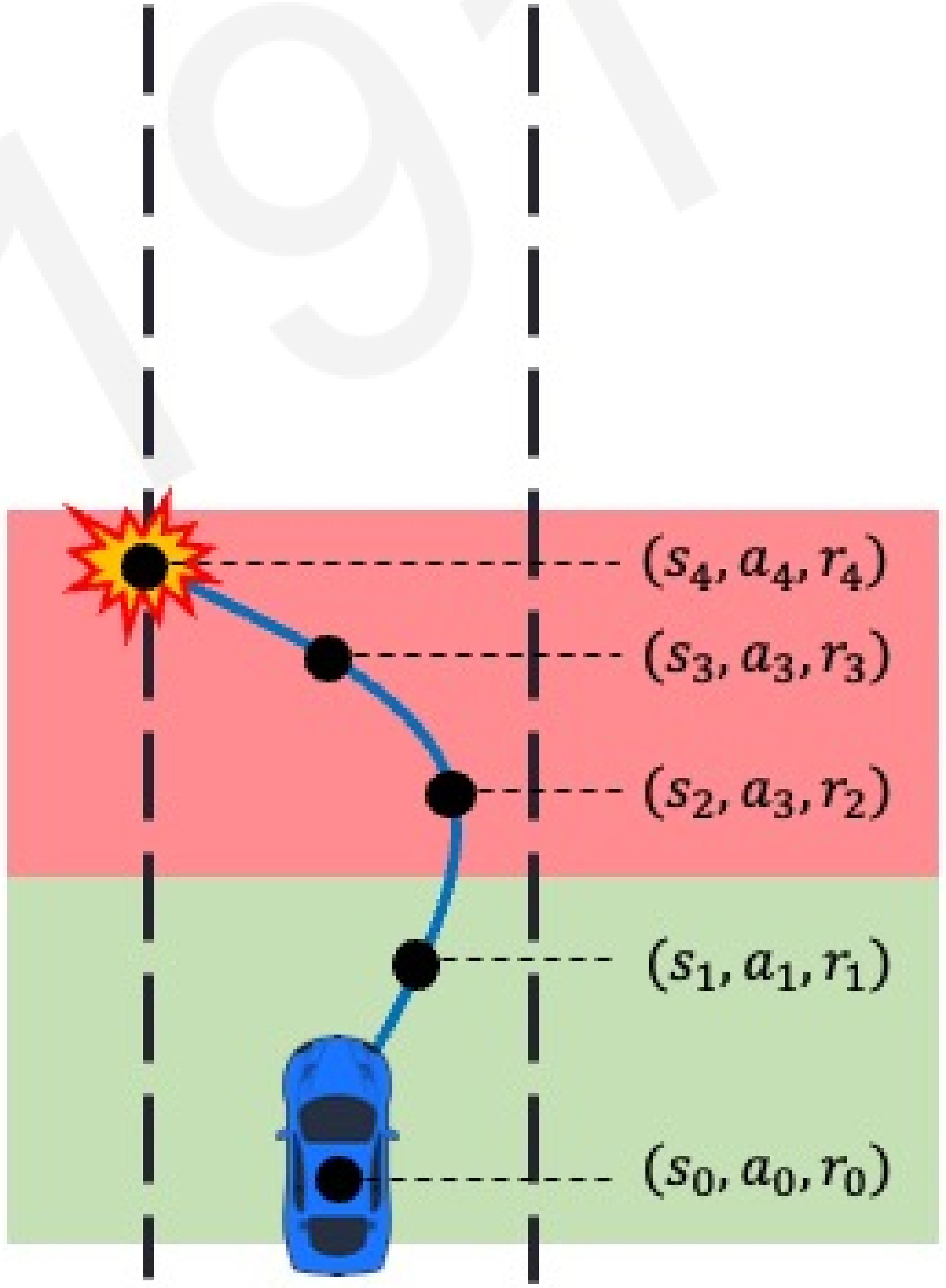
Action: steering wheel angle

Reward: distance traveled

Training Policy Gradients

Training Algorithm

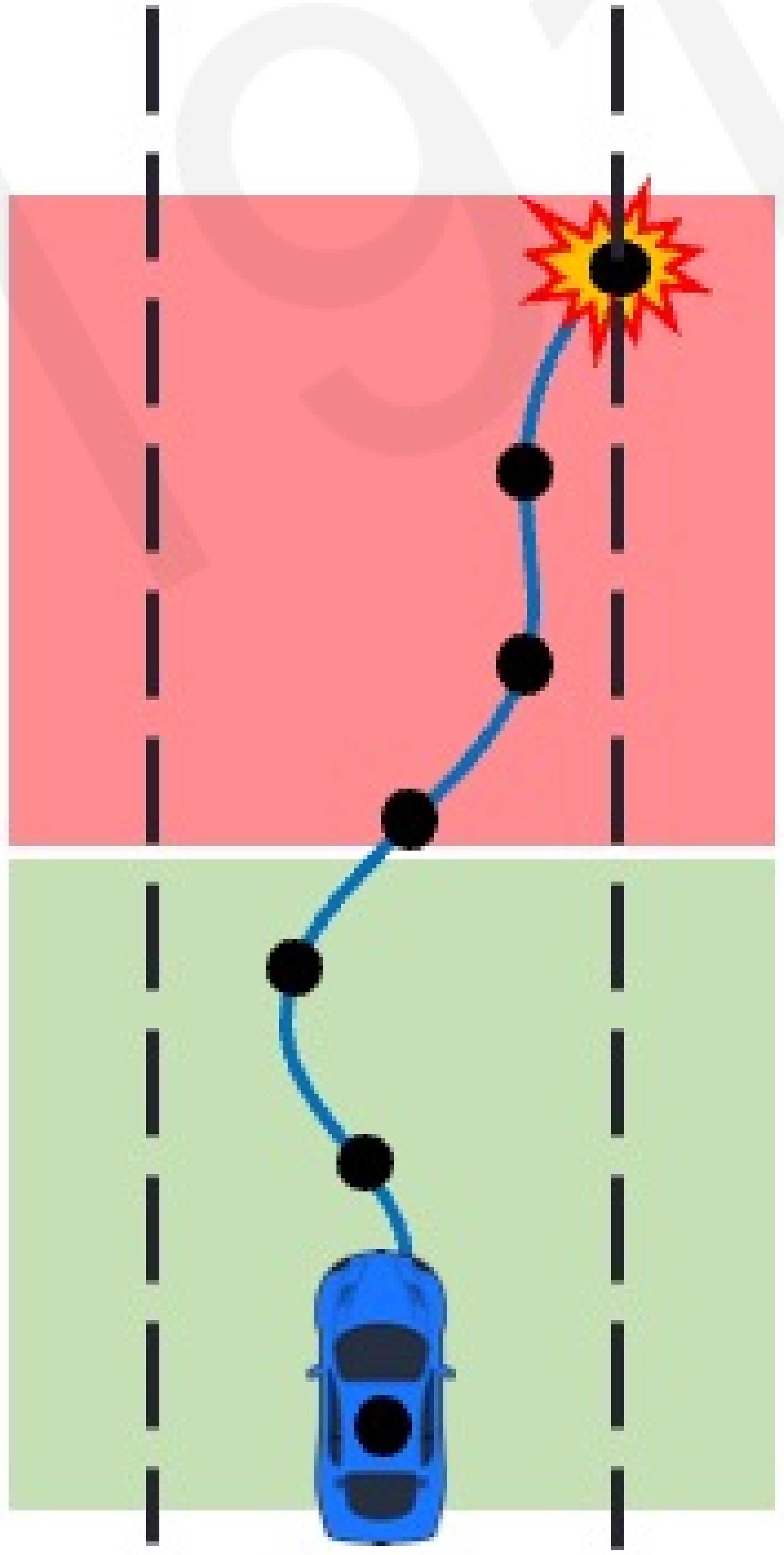
1. Initialize the agent
2. Run a policy until termination
3. Record all states, actions, rewards
4. Decrease probability of actions that resulted in low reward
5. Increase probability of actions that resulted in high reward



Training Policy Gradients

Training Algorithm

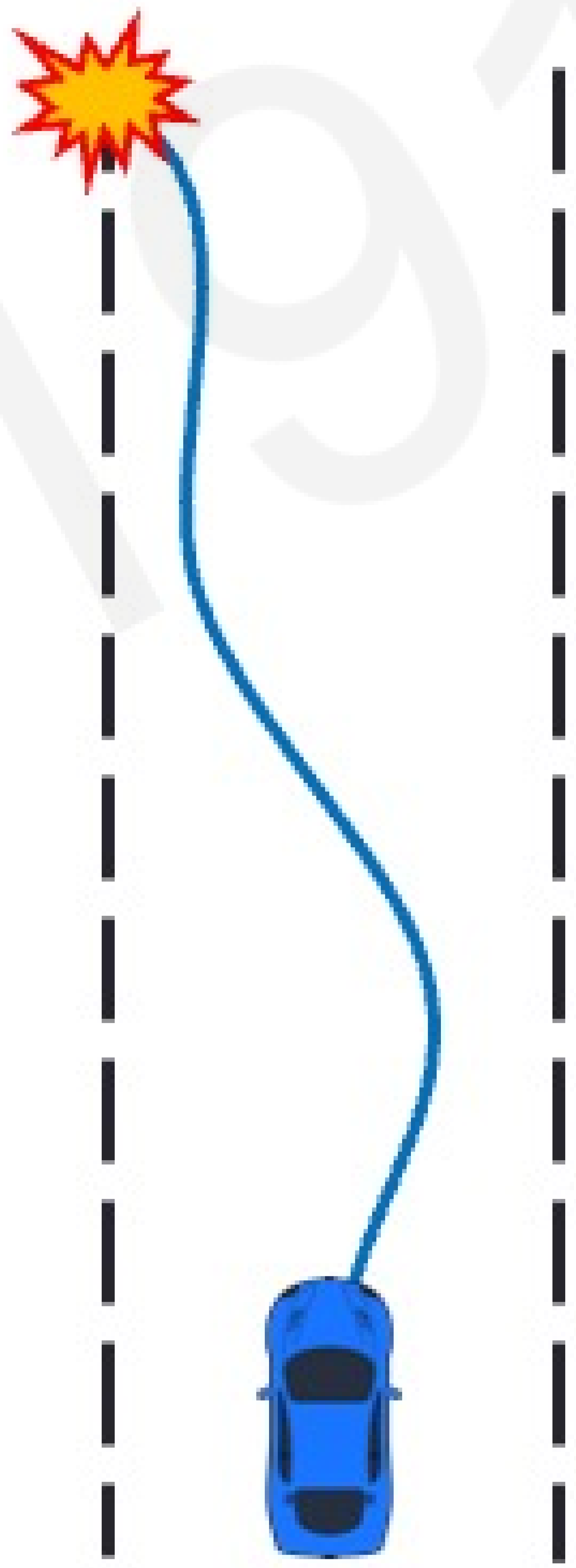
1. Initialize the agent
2. Run a policy until termination
3. Record all states, actions, rewards
4. Decrease probability of actions that resulted in low reward
5. Increase probability of actions that resulted in high reward



Training Policy Gradients

Training Algorithm

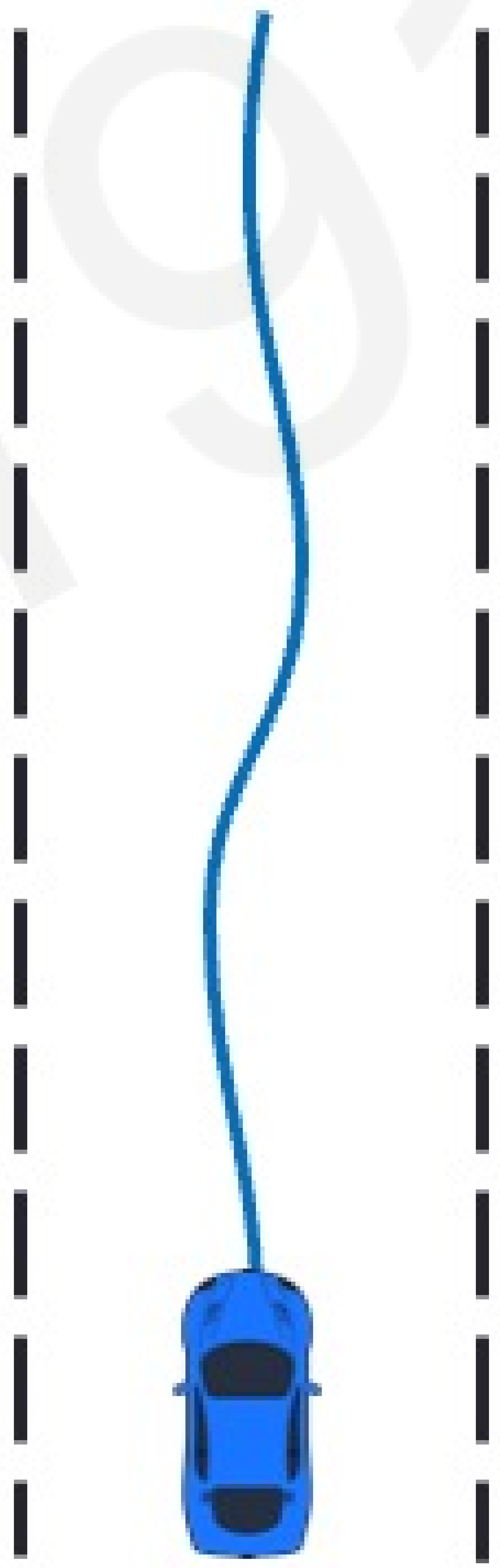
1. Initialize the agent
2. Run a policy until termination
3. Record all states, actions, rewards
4. Decrease probability of actions that resulted in low reward
5. Increase probability of actions that resulted in high reward



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

log-likelihood of action

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

reward

Gradient descent update:

$$w' = w - \nabla\, \textbf{loss}$$
$$w' = w + \boxed{\nabla \log \mathrm{P}(a_t|s_t)\, R_t}$$

Policy gradient!

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

Data-driven Simulation for Autonomous Vehicles

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

Deploying End-to-End RL for Autonomous Vehicles

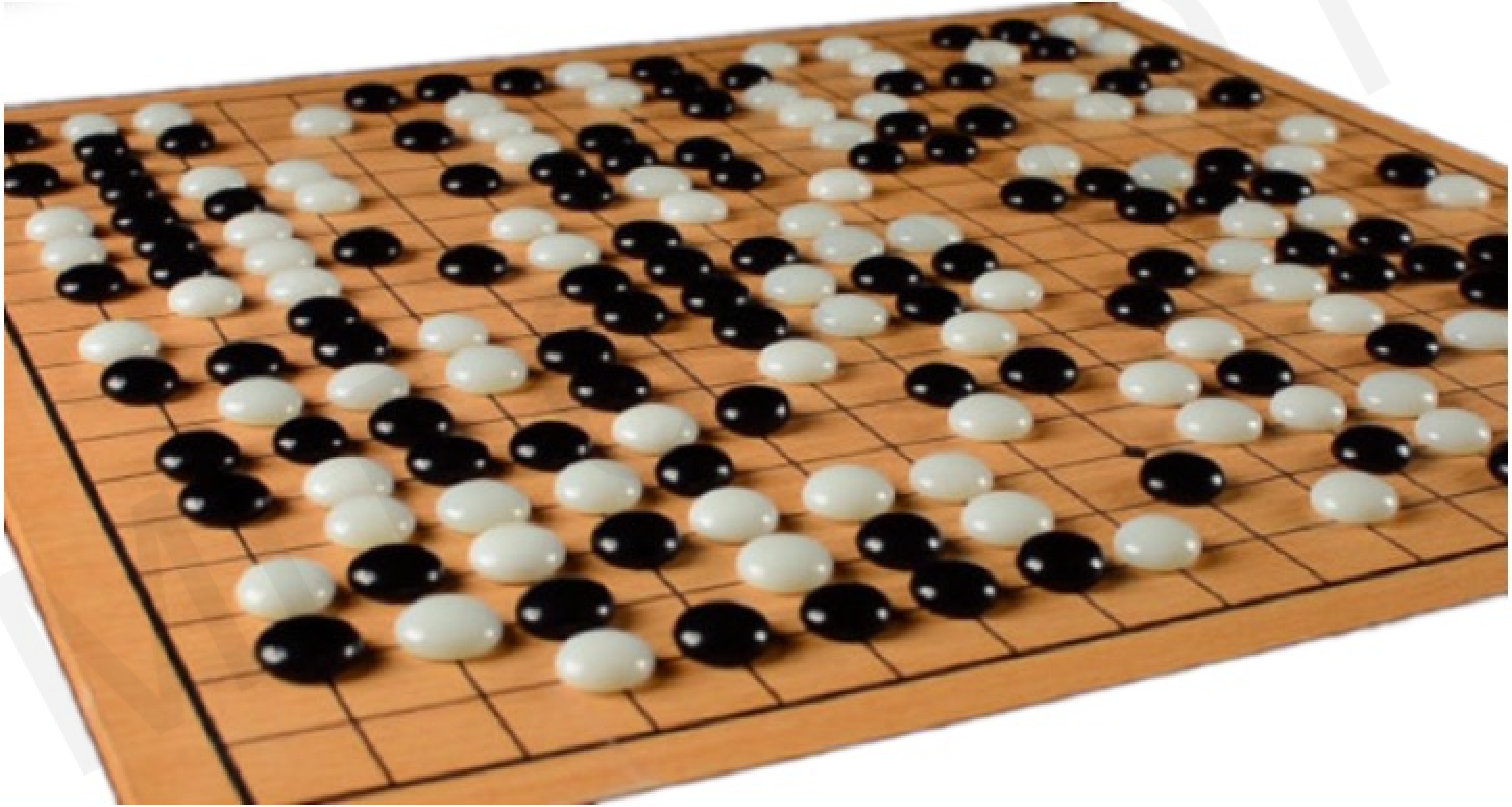
Policy Gradient RL agent trained entirely within VISTA simulator

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!

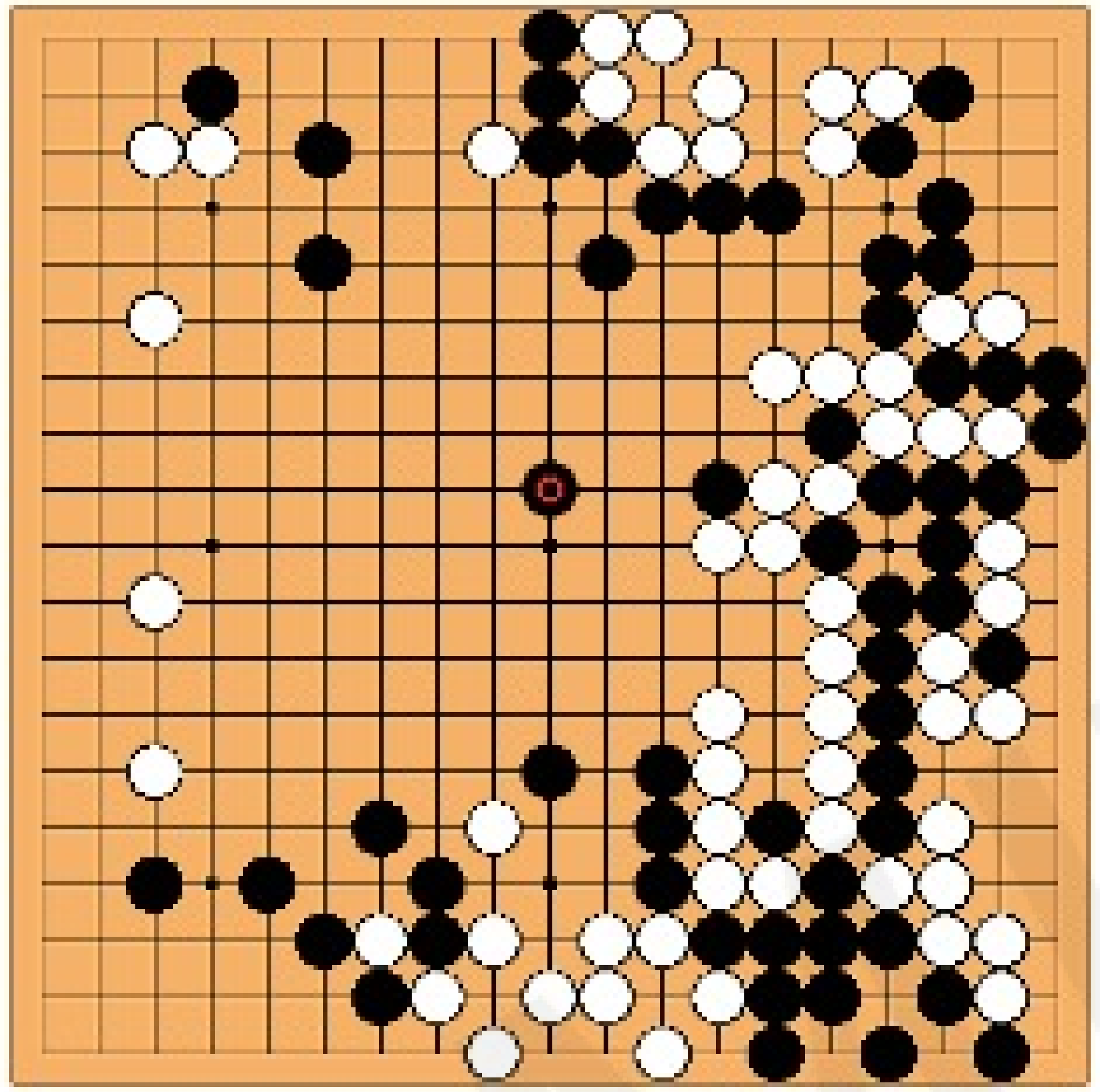
Deep Reinforcement Learning Applications

Reinforcement Learning and the Game of Go



The Game of Go

Aim: Get more board territory than your opponent.



Board Size $n \times n$	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%	$1.03919148791 \times 10^{38}$
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.

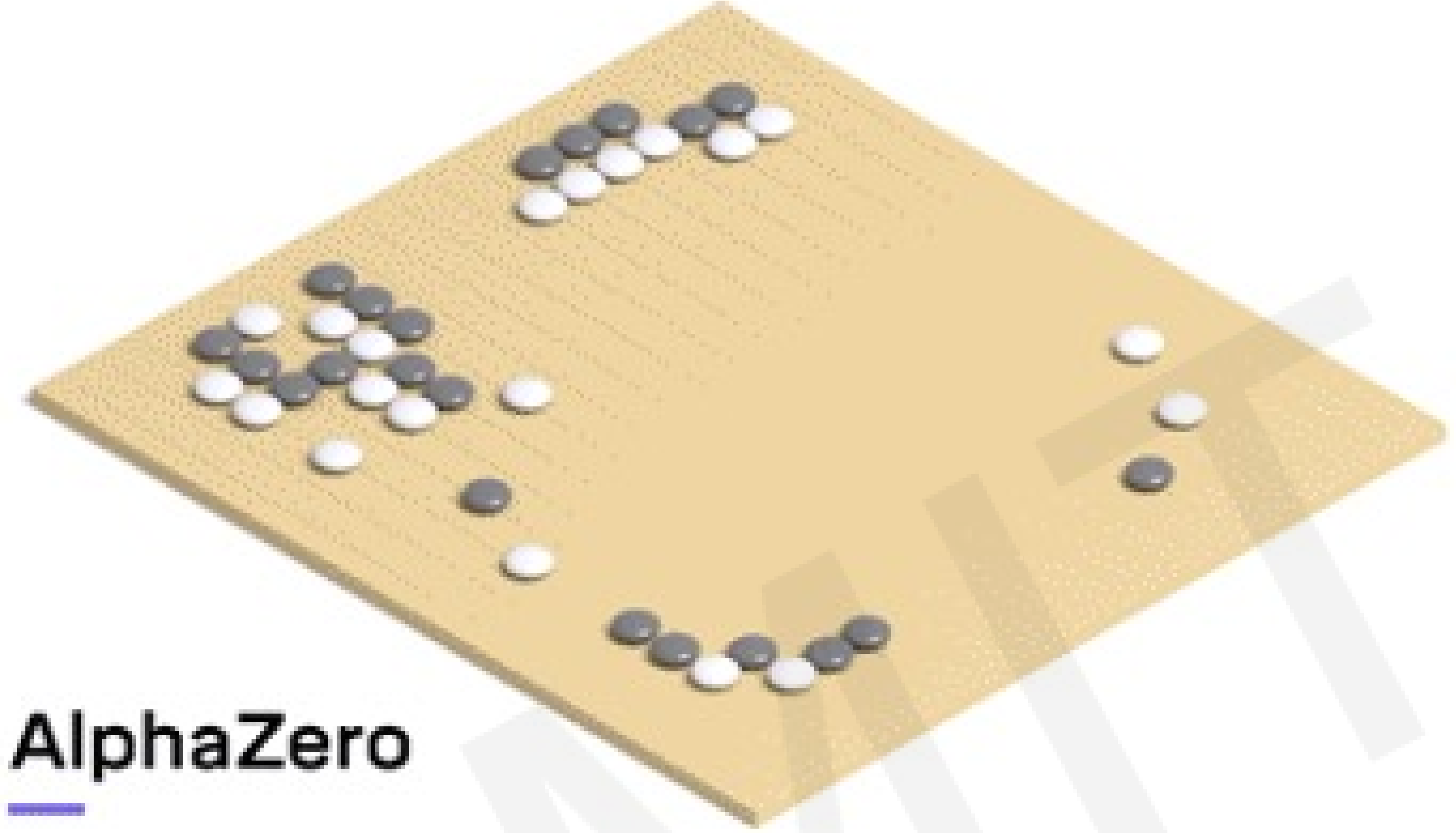
AlphaGo Beats Top Human Player at Go (2016)

AlphaGo Beats Top Human Player at Go (2016)

AlphaGo Beats Top Human Player at Go (2016)

AlphaGo Beats Top Human Player at Go (2016)

AlphaZero: RL from Self-Play (2018)

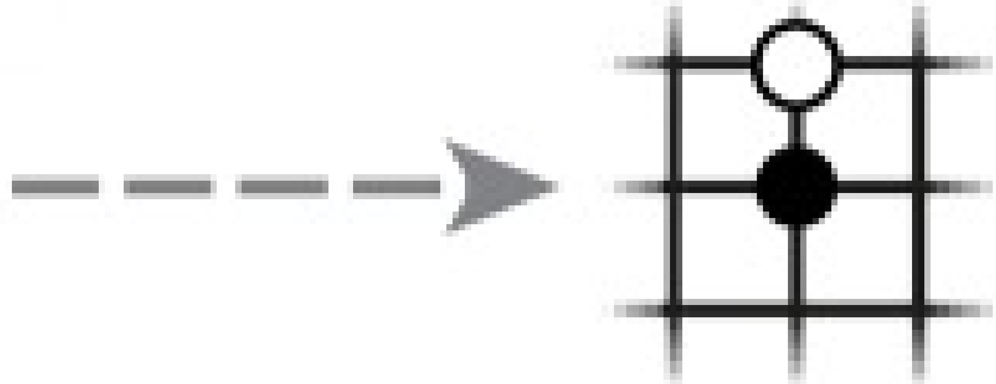


MuZero: Learning Dynamics for Planning (2020)

MuZero: Learning Dynamics for Planning (2020)

How MuZero acts in its environment:

1) Observe **2) Search** **3) Plan** **4) Act**



Deep Reinforcement Learning: Summary

Foundations

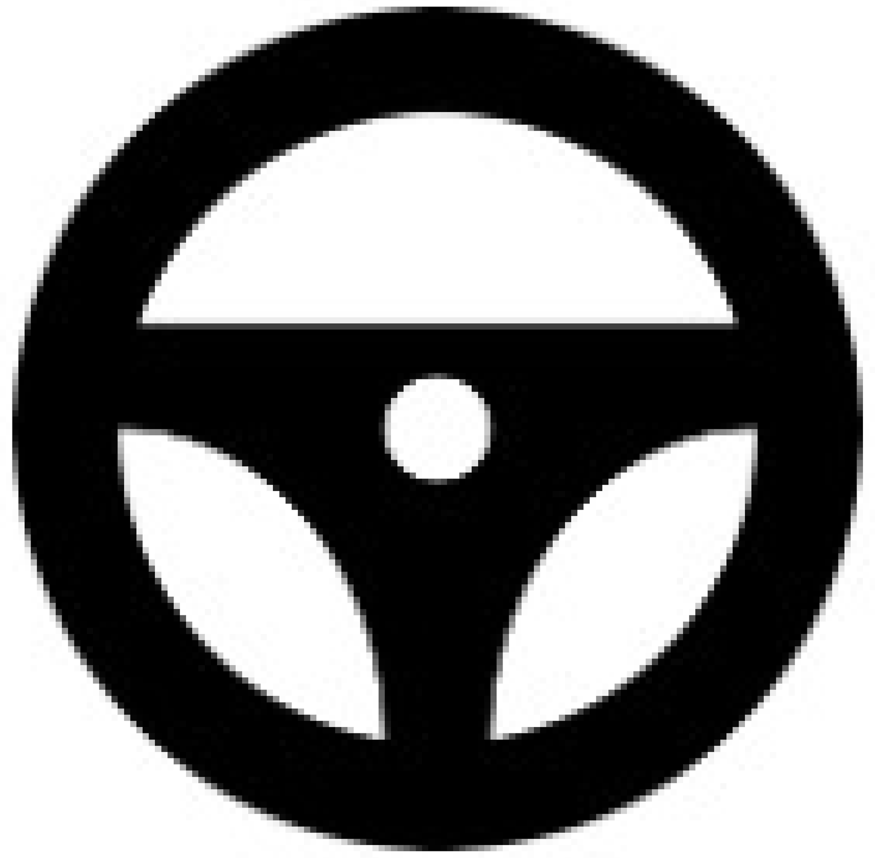
- Agents acting in environment
- State-action pairs → maximize future rewards
- Discounting

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



MIT
Introduction to Deep Learning

Lab 3: Debiasing, Uncertainty, and Robustness

Link to download labs:
http://introtodeeplearning.com#schedule

1. Open the lab in Google Colab
2. Start executing code blocks and filling in the #TODOs
3. Need help? Come to 32-123!