#DLUPC

Day 7 Lecture 2

Reinforcement Learning

[course site]

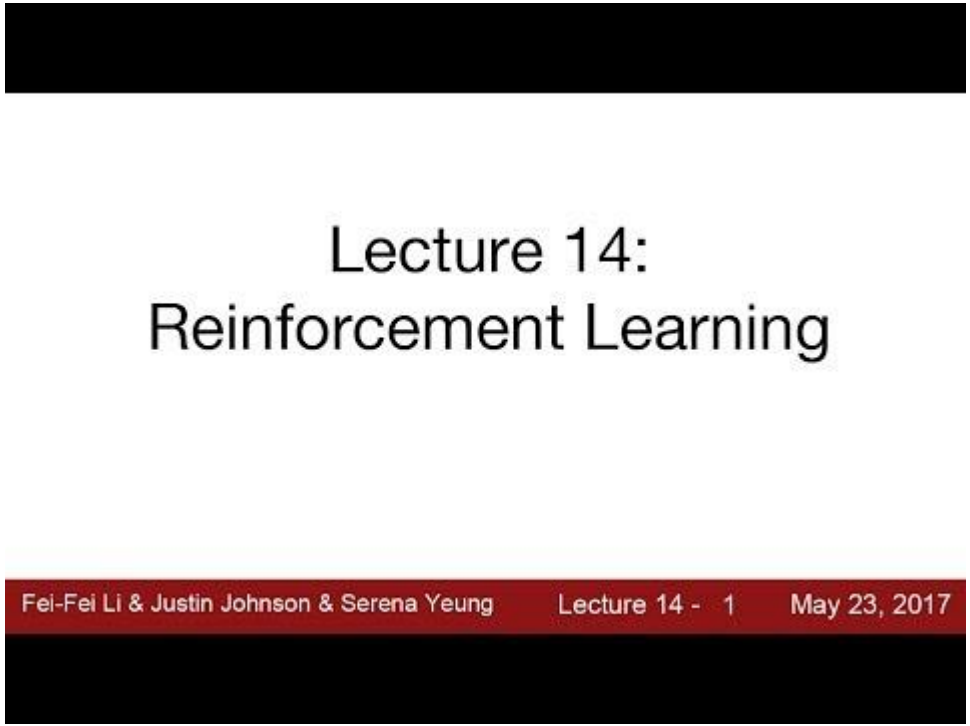
Xavier Giro-i-Nieto
xavier.giro@upc.edu

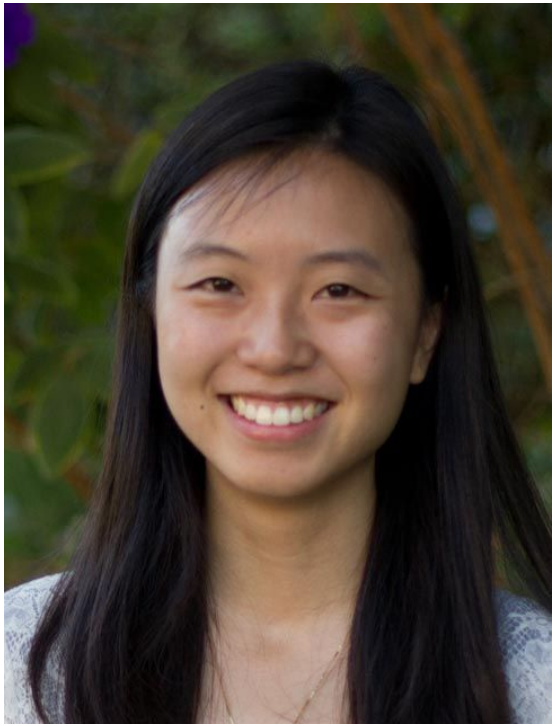
Associate Professor
Universitat Politecnica de Catalunya
Technical University of Catalonia

Acknowledegments

Bellver M, Giró-i-Nieto X, Marqués F, Torres J. Hierarchical Object Detection with Deep Reinforcement Learning. In Deep Reinforcement Learning Workshop, NIPS 2016. 2016.

Acknowledegments





Serena Yeung, “Deep Reinforcement Learning”. Stanford University CS231n, 2017.

Outline

1. Motivation
2. Architecture
3. Markov Decision Process (MDP)
4. Deep Q-learning
5. RL Frameworks
6. Learn more

Outline

1. **Motivation**
2. Architecture
3. Markov Decision Process (MDP)
4. Deep Q-learning
5. RL Frameworks
6. Learn more

Motivation

What is Reinforcement Learning ?

“a way of programming agents by reward and punishment without needing to specify how the task is to be achieved”

[Kaelbling, Littman, & Moore, 96]

Kaelbling, Leslie Pack, Michael L. Littman, and Andrew W. Moore. "Reinforcement learning: A survey." Journal of artificial intelligence research 4 (1996): 237-285.

Motivation

Yann Lecun's Black Forest cake

- "Pure" Reinforcement Learning (cherry)
  - The machine predicts a scalar reward given once in a while.
  - **A few bits for some samples**
- Supervised Learning (icing)
  - The machine predicts a category or a few numbers for each input
  - Predicting human-supplied data
  - **10→10,000 bits per sample**
- Unsupervised/Predictive Learning (cake)
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**

- (Yes, I know, this picture is slightly offensive to RL folks. But I'll make it up)

Motivation

We can categorize three types of learning procedures:

1. Supervised Learning:

$$\mathbf{y} = f(\mathbf{x})$$

Predict label y corresponding to observation x

2. Unsupervised Learning:

$$f(\mathbf{x})$$

Estimate the distribution of observation x

3. Reinforcement Learning (RL):

$$\mathbf{y} = f(\mathbf{x})$$

$$\mathbf{z}$$

Predict action y based on observation x, to maximize a future reward z

Motivation

We can categorize three types of learning procedures:

1. Supervised Learning:

$$\mathbf{y} = f(\mathbf{x})$$

2. Unsupervised Learning:

$$f(\mathbf{x})$$

3. Reinforcement Learning (RL):

$$\mathbf{y} = f(\mathbf{x})$$

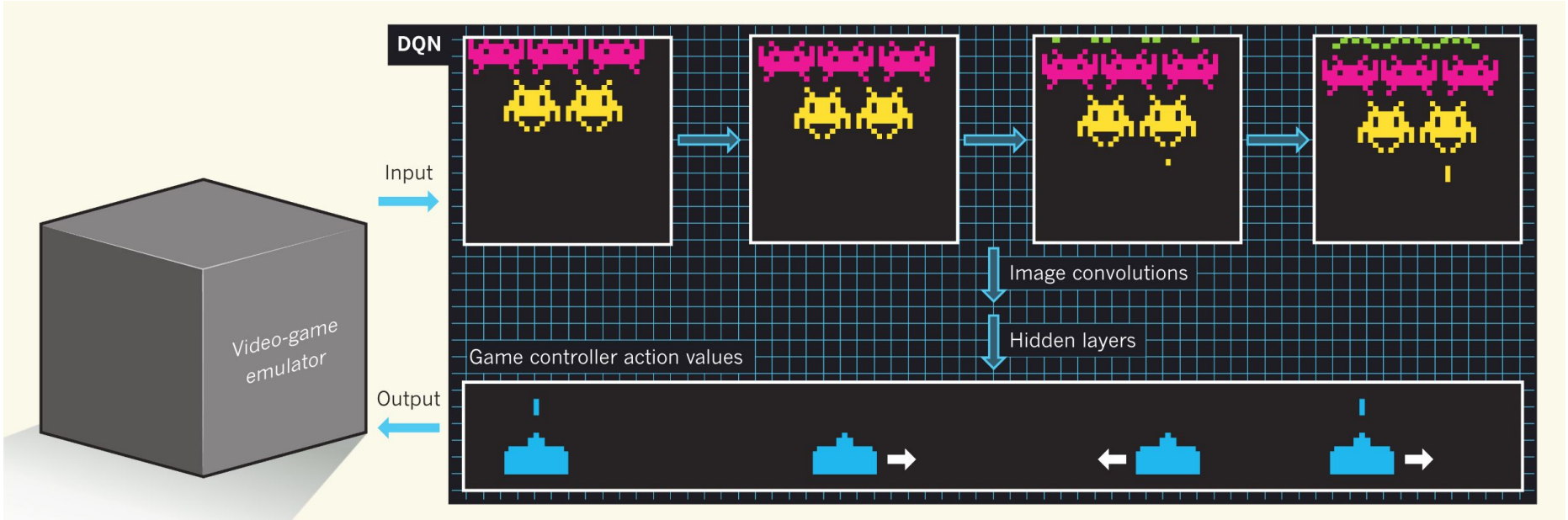
$$\mathbf{z}$$

Outline

1. Motivation
2. **Architecture**
3. Markov Decision Process (MDP)
4. Deep Q-learning
5. RL Frameworks
6. Learn more

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. "Playing atari with deep reinforcement learning." arXiv preprint arXiv:1312.5602 (2013).

Motivation



Bernhard Schölkkopf, "Learning to see and act" Nature 2015.

Outline

1. Motivation
2. Architecture
3. Markov Decision Process (MDP)
    - Policy
    - Optimal Policy
    - Value Function
    - Q-value function
    - Optimal Q-value function
    - Bellman equation
    - Value iteration algorithm
4. Deep Q-learning
5. RL Frameworks
6. Learn more

Architecture

Figure: UCL Course on RL by David Silver

Architecture

Figure: UCL Course on RL by David Silver

Architecture

state (s_t)

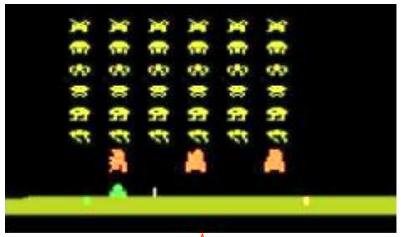
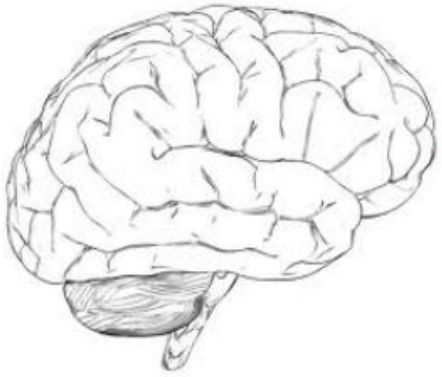


Figure: UCL Course on RL by David Silver

Architecture



state (s_t)

Environment

Figure: UCL Course on RL by David Silver

Architecture

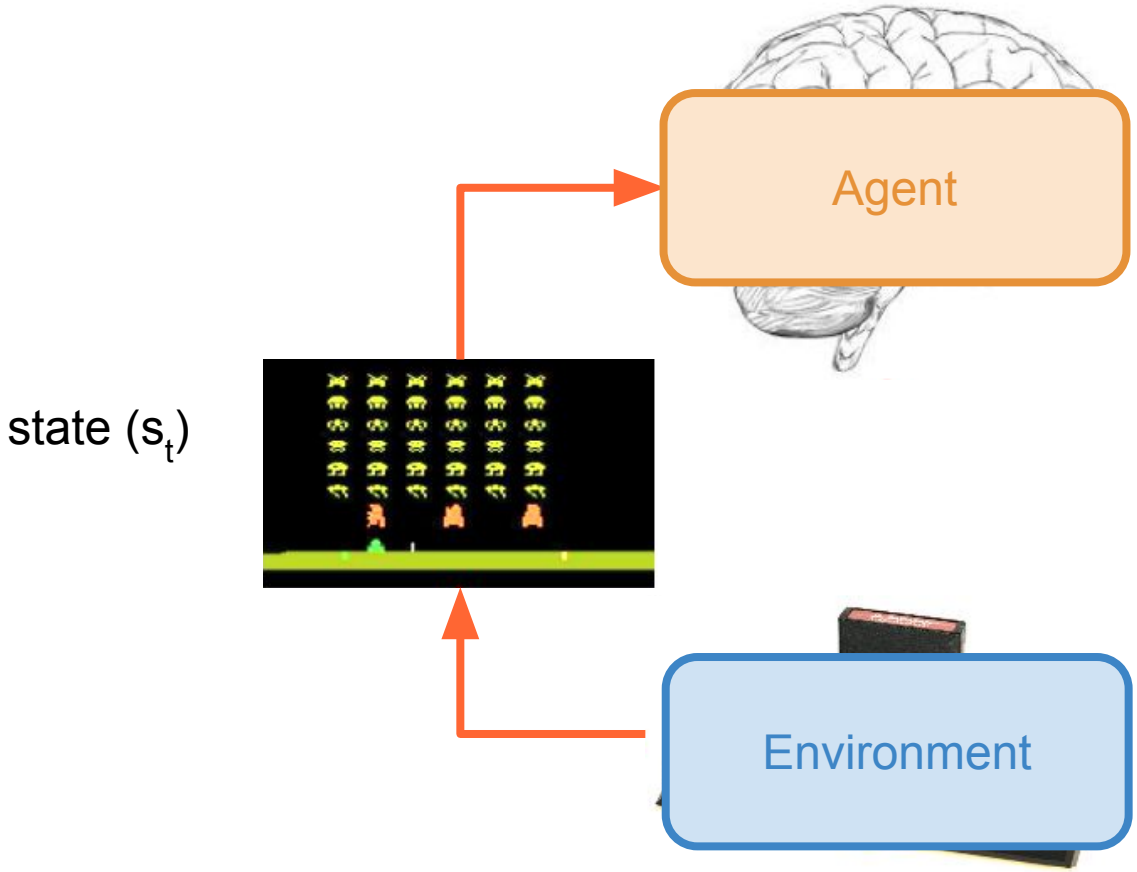


Figure: UCL Course on RL by David Silver

Architecture

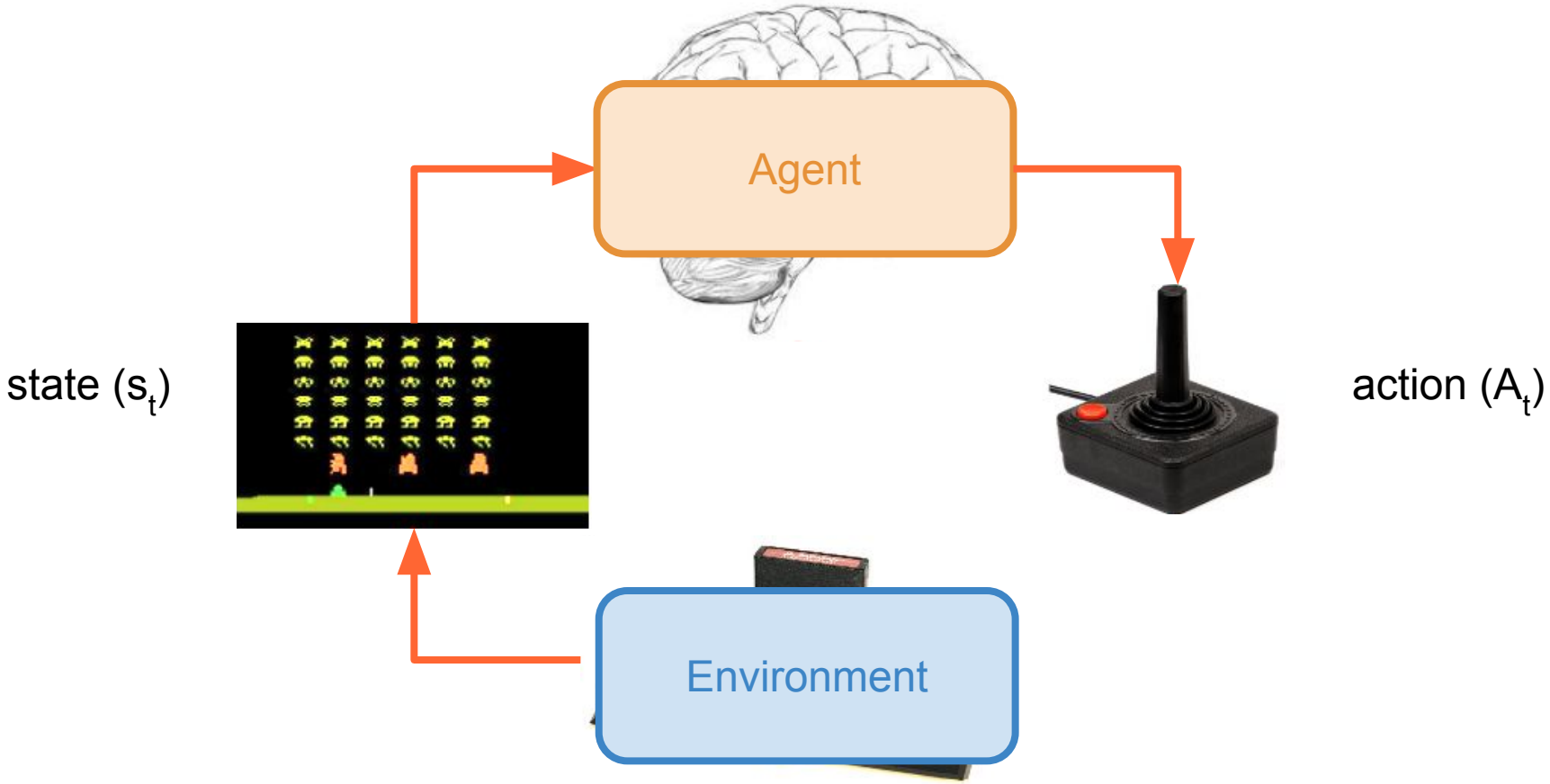


Figure: UCL Course on RL by David Silver

Architecture

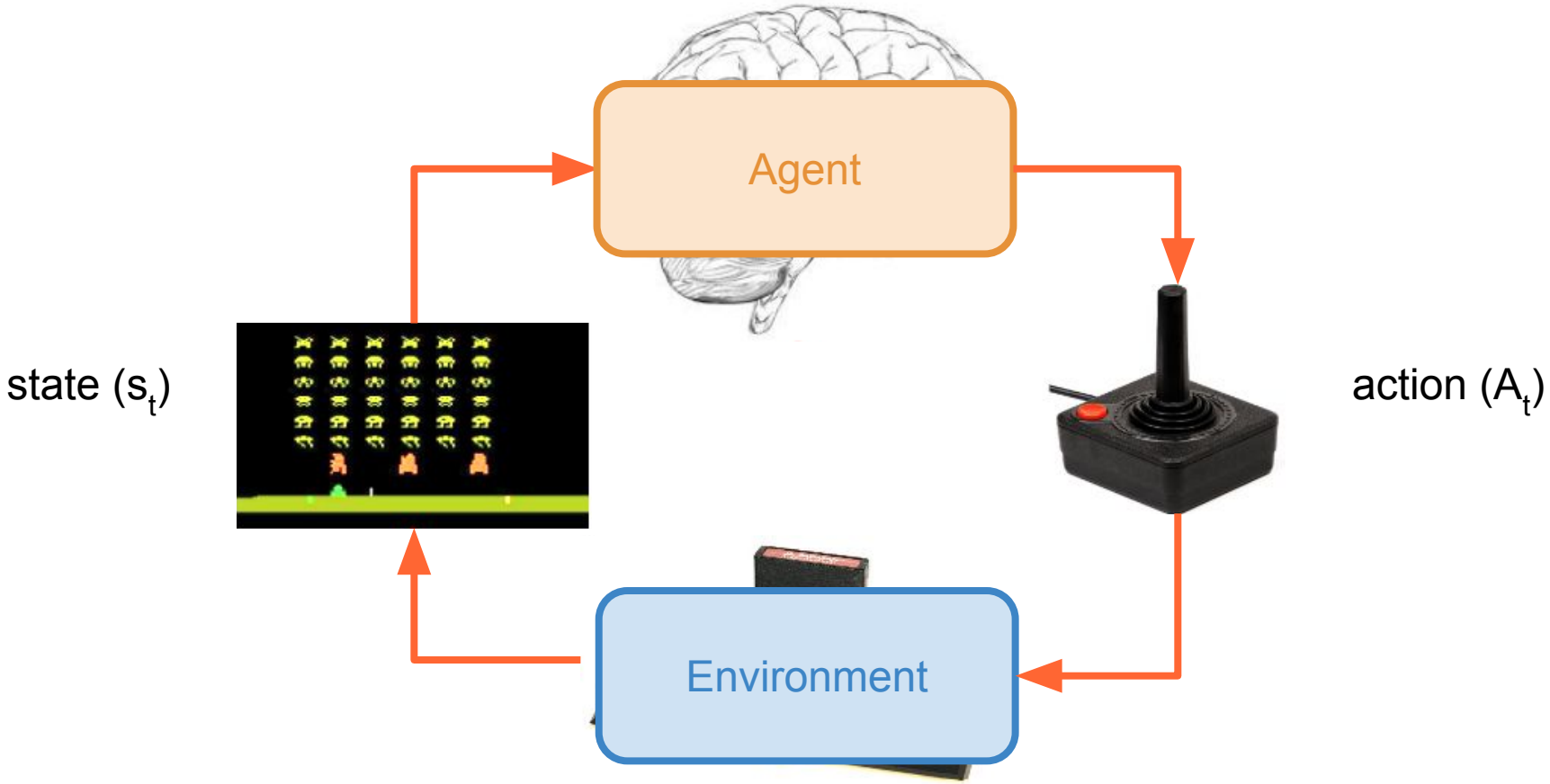


Figure: UCL Course on RL by David Silver

Architecture

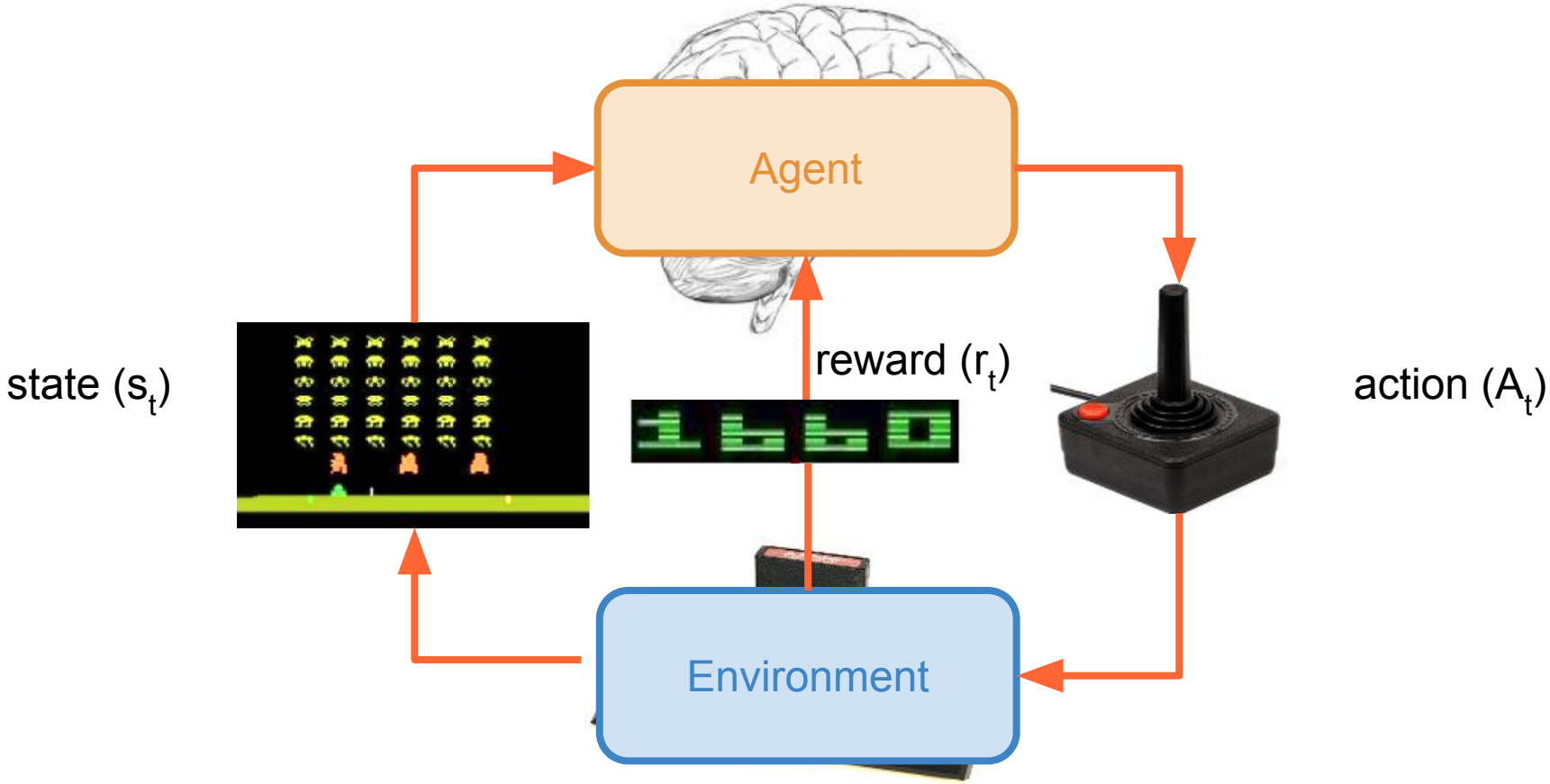
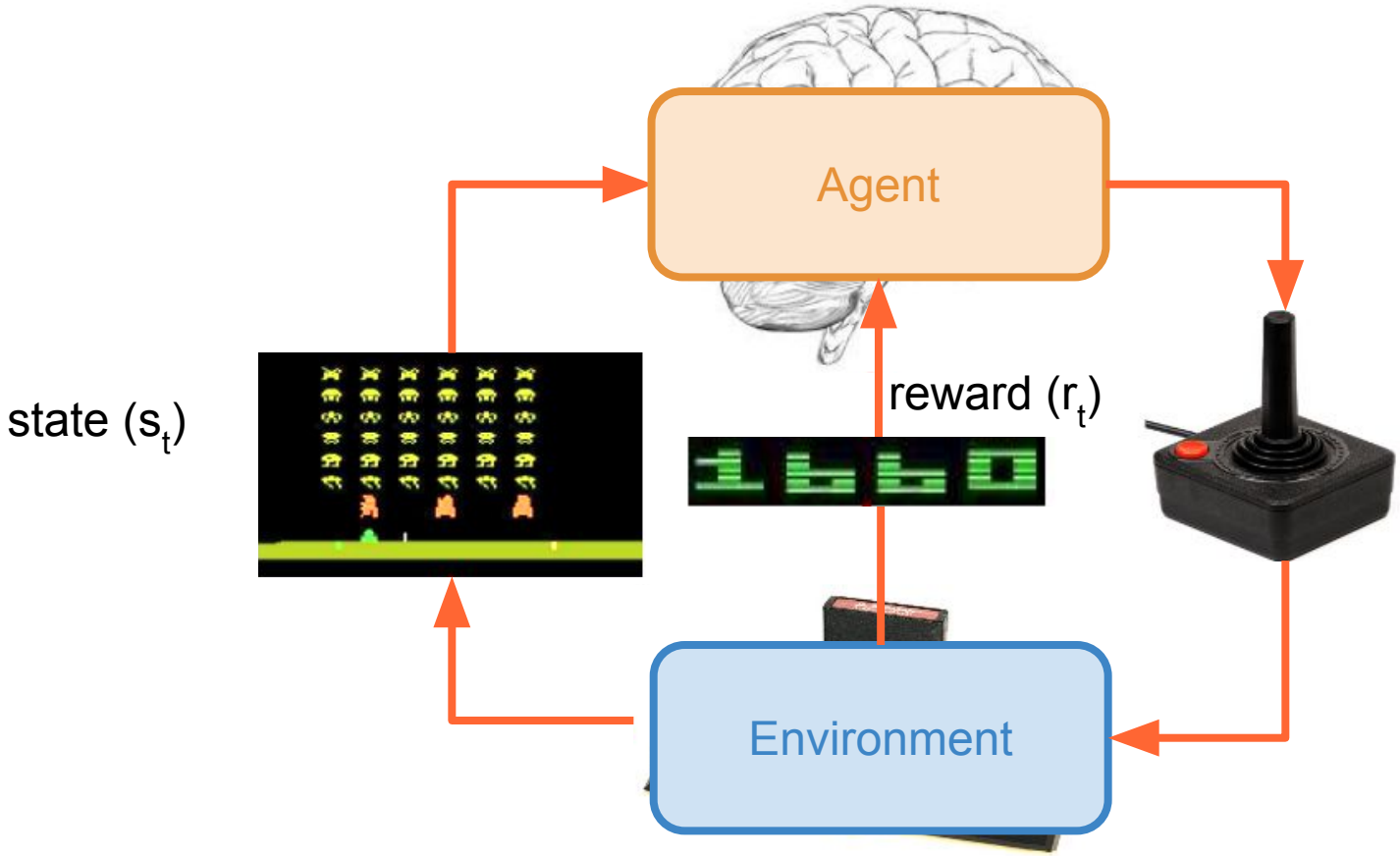


Figure: UCL Course on RL by David Silver

Architecture



Reward is given to the agent delayed with respect to previous states and actions !

Figure: UCL Course on RL by David Silver

Architecture

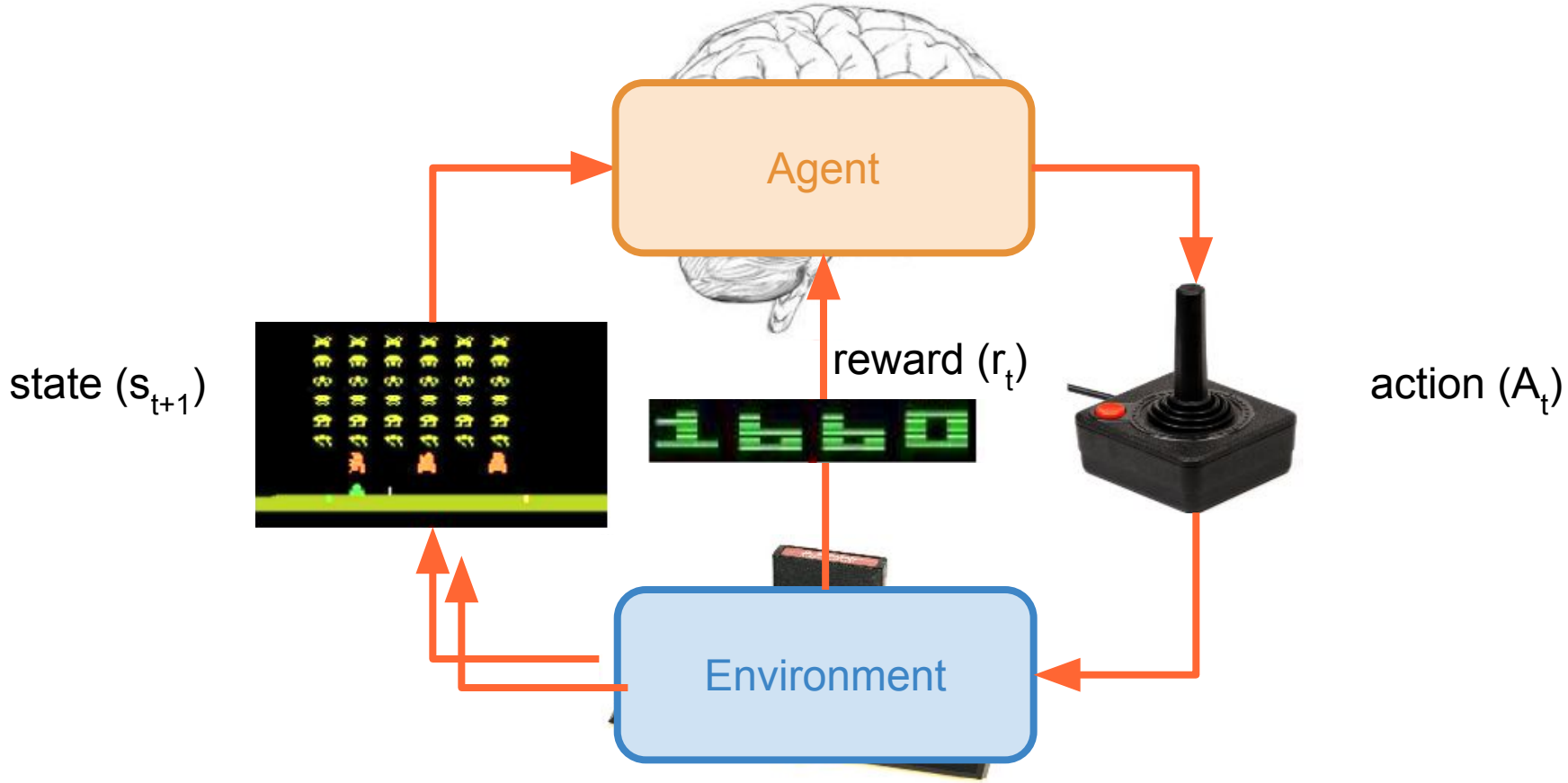


Figure: UCL Course on RL by David Silver

Architecture

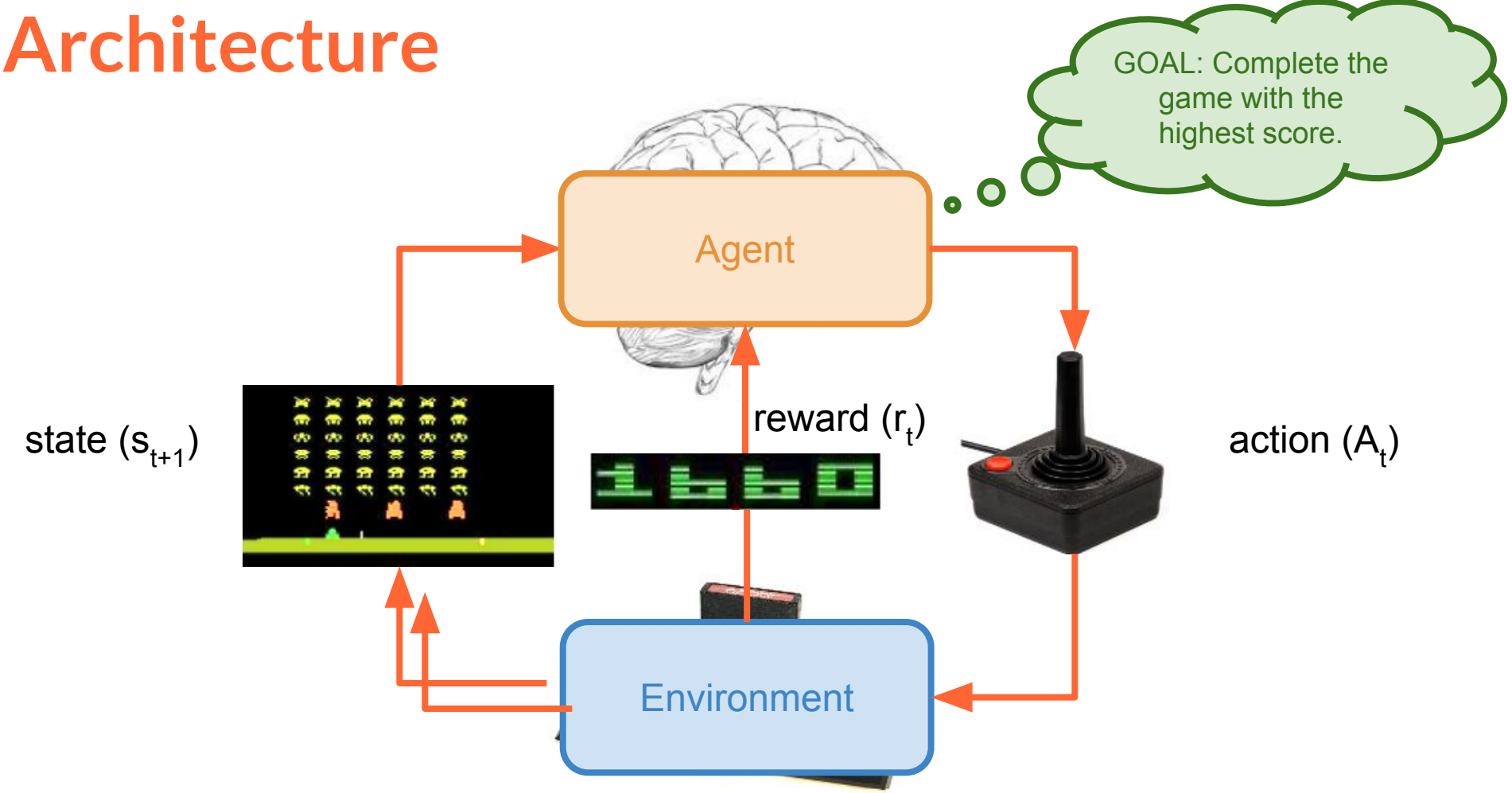


Figure: UCL Course on RL by David Silver

Architecture

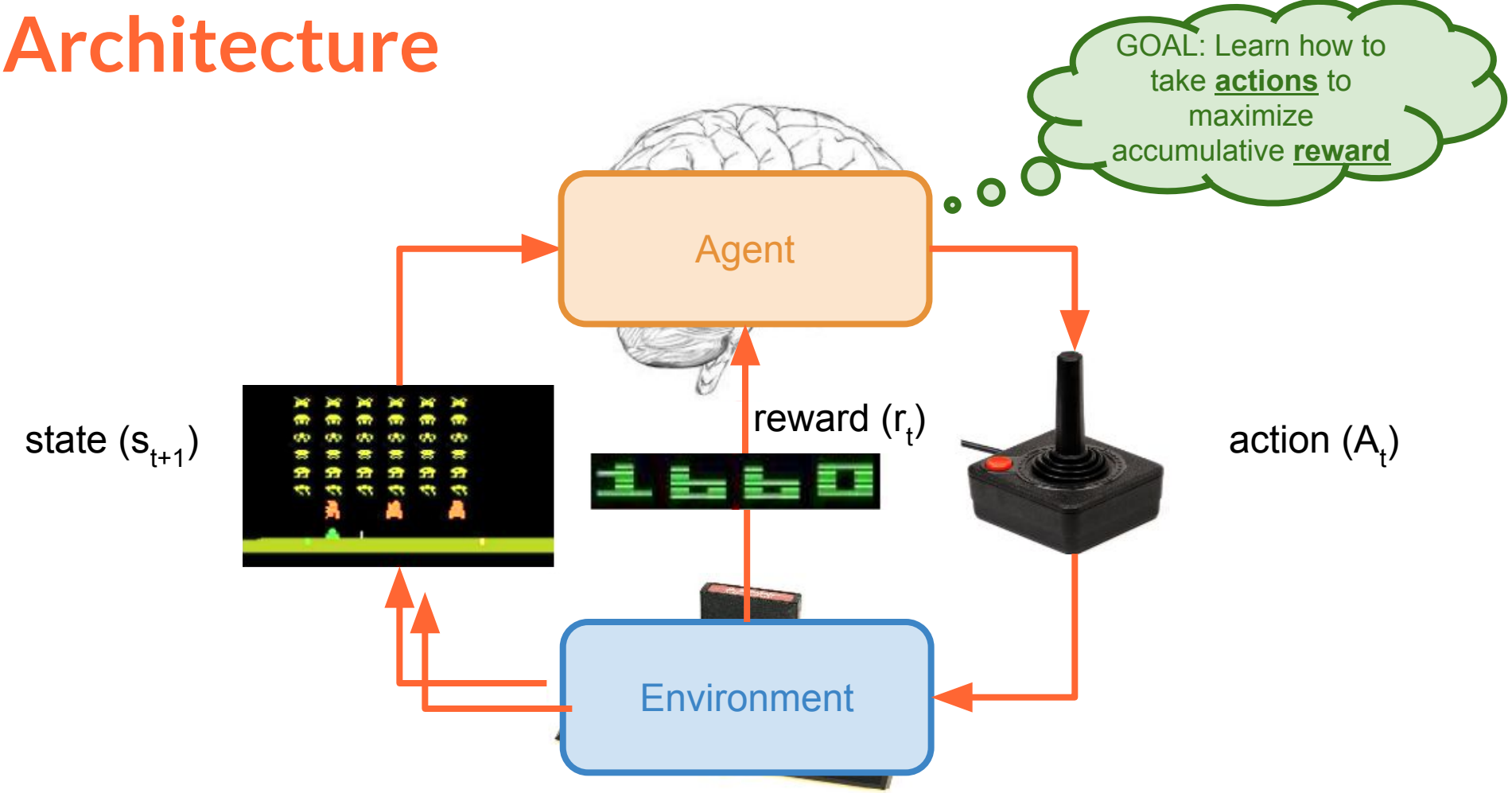


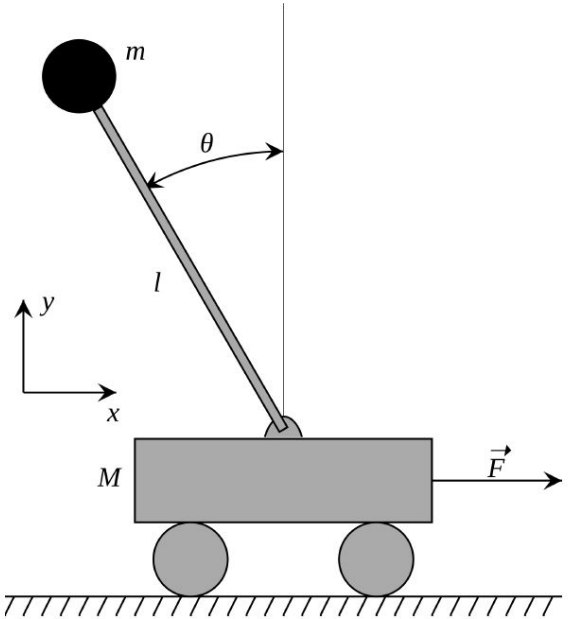
Figure: UCL Course on RL by David Silver

Architecture

Other problems that can be formulated with a RL architecture.

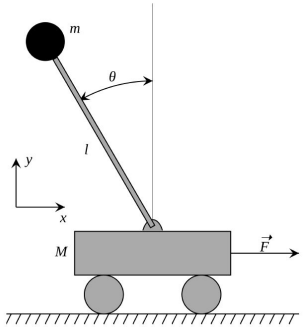
Cart-Pole Problem

Objective: Balance a pole on top of a movable car



Slide credit: Serena Yeung, "Deep Reinforcement Learning". Stanford University CS231n, 2017.

Architecture



Agent

state (s_t)

Angle
Angular speed
Position
Horizontal velocity

reward (r_t)

1 at each time step if the pole is upright

Horizontal force applied in the car

action (A_t)

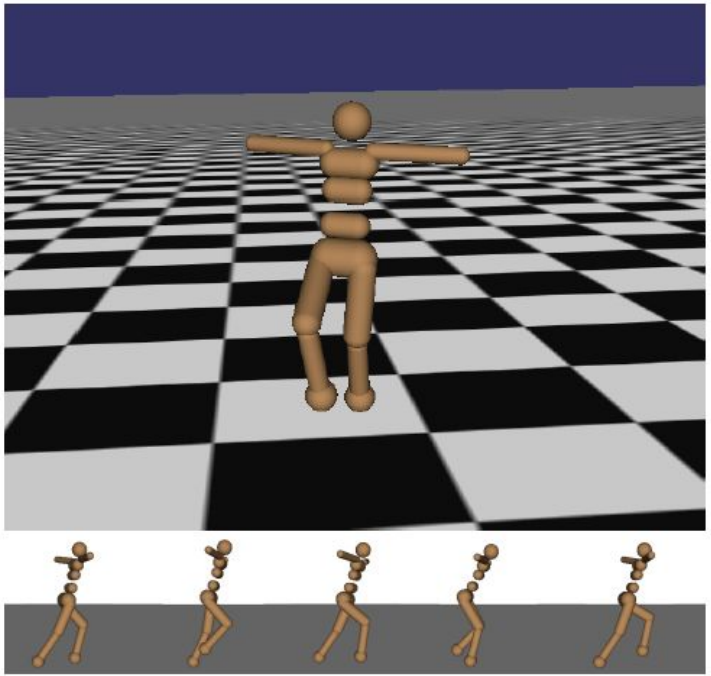
Environment

Slide concept: Serena Yeung, "Deep Reinforcement Learning". Stanford University CS231n, 2017.

Architecture

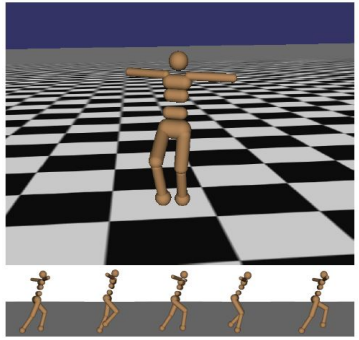
Other problems that can be formulated with a RL architecture.

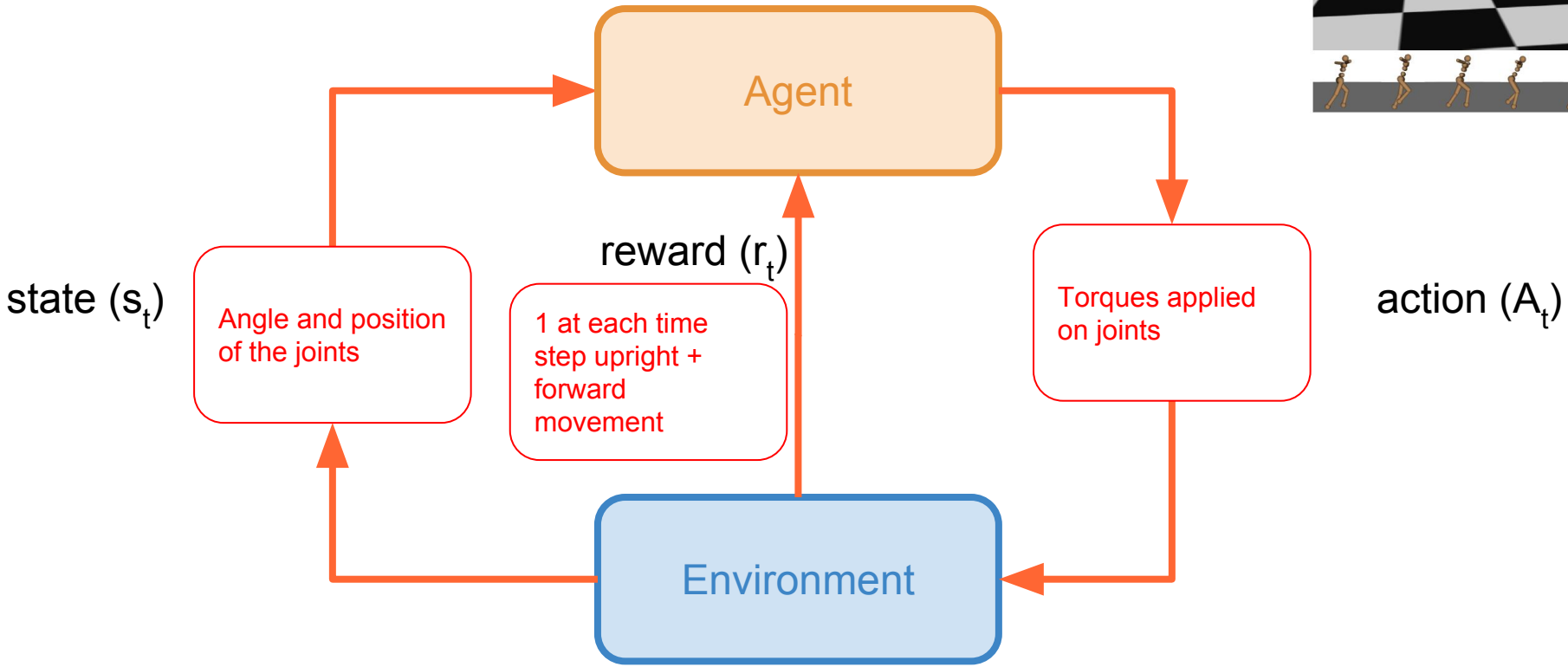
Robot Locomotion
Objective: Make the robot move forward



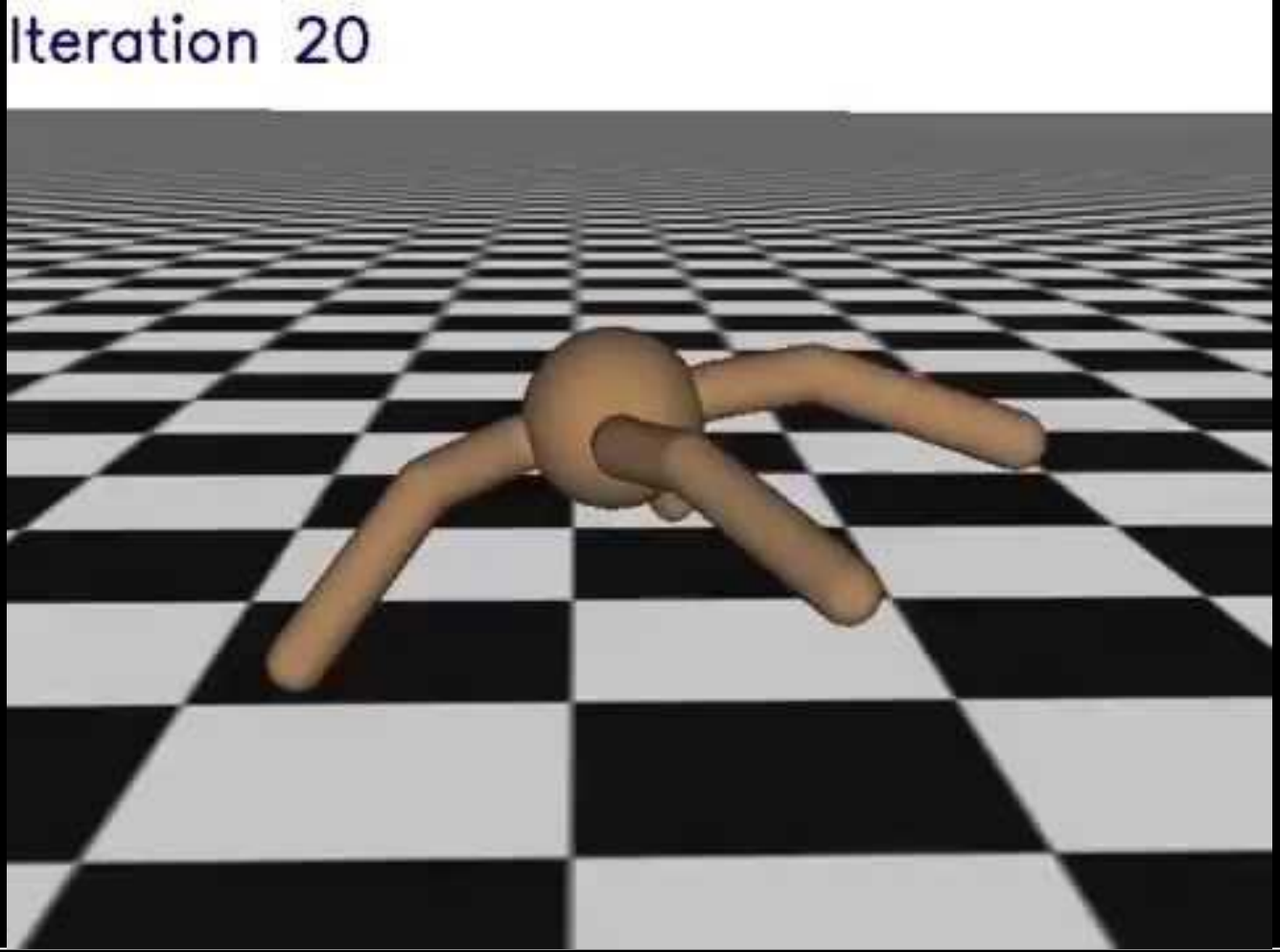
Schulman, John, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. "High-dimensional continuous control using generalized advantage estimation." ICLR 2016 [project page]

Architecture





Slide concept: Serena Yeung, "Deep Reinforcement Learning". Stanford University CS231n, 2017.



Schulman, John, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. "High-dimensional continuous control using generalized advantage estimation." ICLR 2016 [project page]

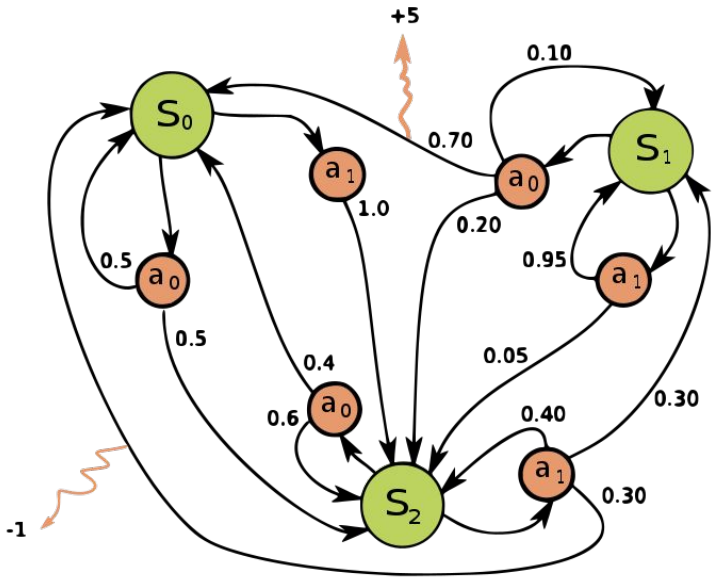
Outline

1. Motivation
2. Architecture
3. Markov Decision Process (MDP)
    - Policy
    - Optimal Policy
    - Value Function
    - Q-value function
    - Optimal Q-value function
    - Bellman equation
    - Value iteration algorithm
4. Deep Q-learning
5. RL Frameworks
6. Learn more

Markov Decision Processes (MDP)

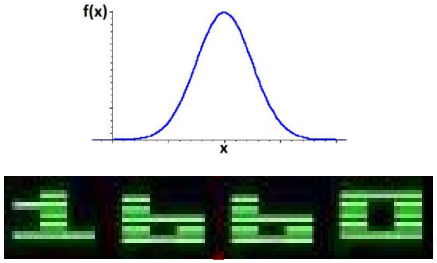
Markov Decision Processes provide a formalism for reinforcement learning problems.

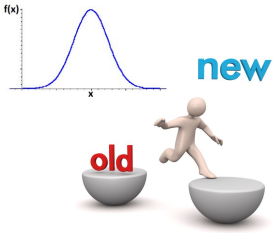
Markov property:
Current state completely characterises the state of the world.



Slide concept: Serena Yeung, "Deep Reinforcement Learning". Stanford University CS231n, 2017.

Markov Decision Processes (MDP)





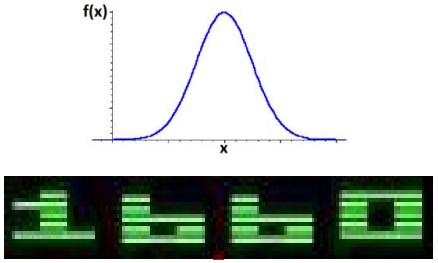
S A R P ɣ

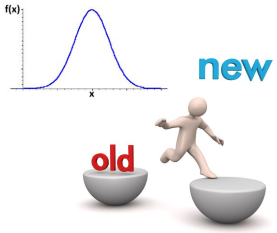
Defined by: $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathbb{P}, \gamma)$

$\mathcal{S}$: set of possible states
$\mathcal{A}$: set of possible actions
$\mathcal{R}$: distribution of reward given (state, action) pair
$\mathbb{P}$: transition probability i.e. distribution over next state given (state, action) pair
γ : discount factor

Slide concept: Serena Yeung, "Deep Reinforcement Learning". Stanford University CS231n, 2017.

Markov Decision Processes (MDP)

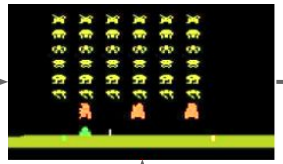




S A R P ɣ

Environment samples initial state $s_0 \sim p(s_0)$

state (s_t)

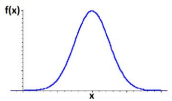


Agent selects action a_t

action (a_t)

Environment samples reward $r_t \sim R(. \mid s_t, a_t)$

reward (r_t)



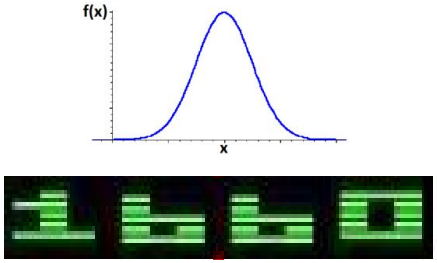
Environment samples next state $s_{t+1} \sim P(. \mid s_t, a_t)$

Slide concept: Serena Yeung, "Deep Reinforcement Learning". Stanford University CS231n, 2017.

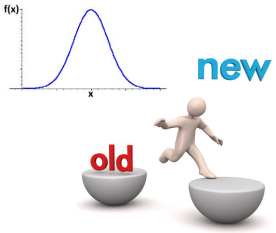
MDP: Policy

S

A



R



P

ɣ

A **Policy π** is a function S → A that specifies which action to take in each state.

Slide concept: Serena Yeung, "Deep Reinforcement Learning". Stanford University CS231n, 2017.

MDP: Policy

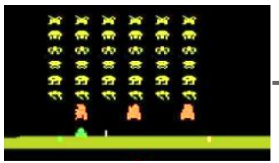
Agent

GOAL: Learn how to take **actions** to maximize **reward**

MDP

GOAL: Find **policy π*** that maximizes the cumulative discounted reward:

$$\sum_{t \geq 0} \gamma^t r_t$$



Agent selects action a_t

policy π

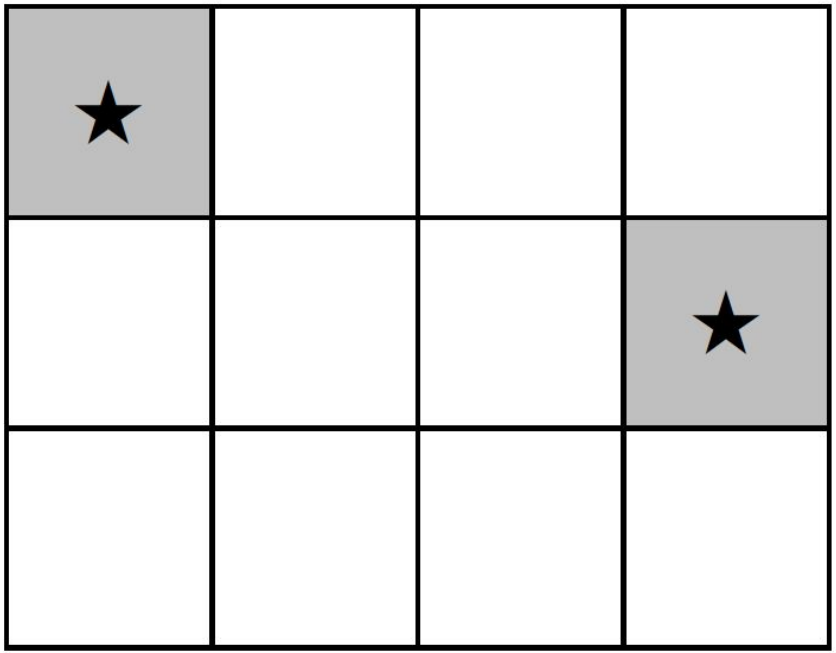
A **Policy π** is a function S → A that specifies which action to take in each state.

Slide concept: Serena Yeung, "Deep Reinforcement Learning". Stanford University CS231n, 2017.

MDP: Policy

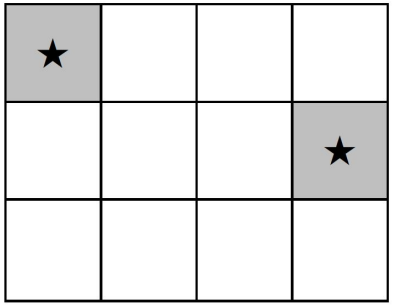
Other problems that can be formulated with a RL architecture.

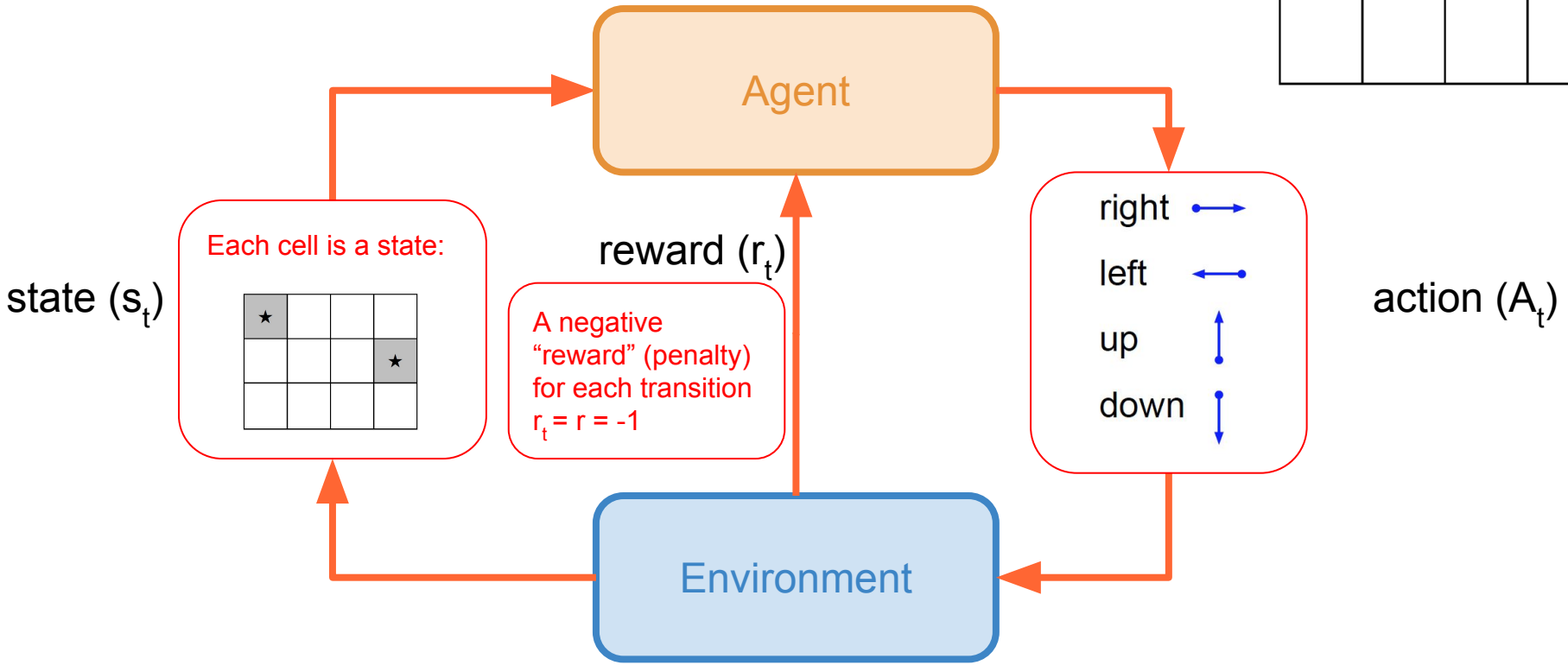
Grid World (a simple MDP)
Objective: reach one of the terminal states (greyed out) in least number of actions.



Slide concept: Serena Yeung, "Deep Reinforcement Learning". Stanford University CS231n, 2017.

MDP: Policy

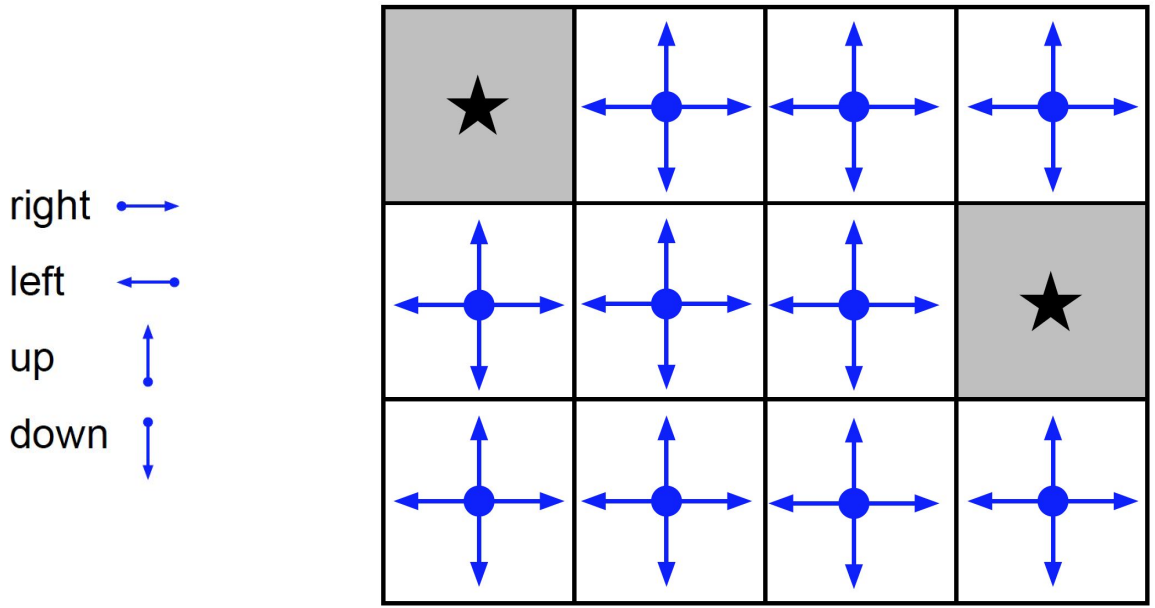




Slide concept: Serena Yeung, "Deep Reinforcement Learning". Stanford University CS231n, 2017.

MDP: Policy

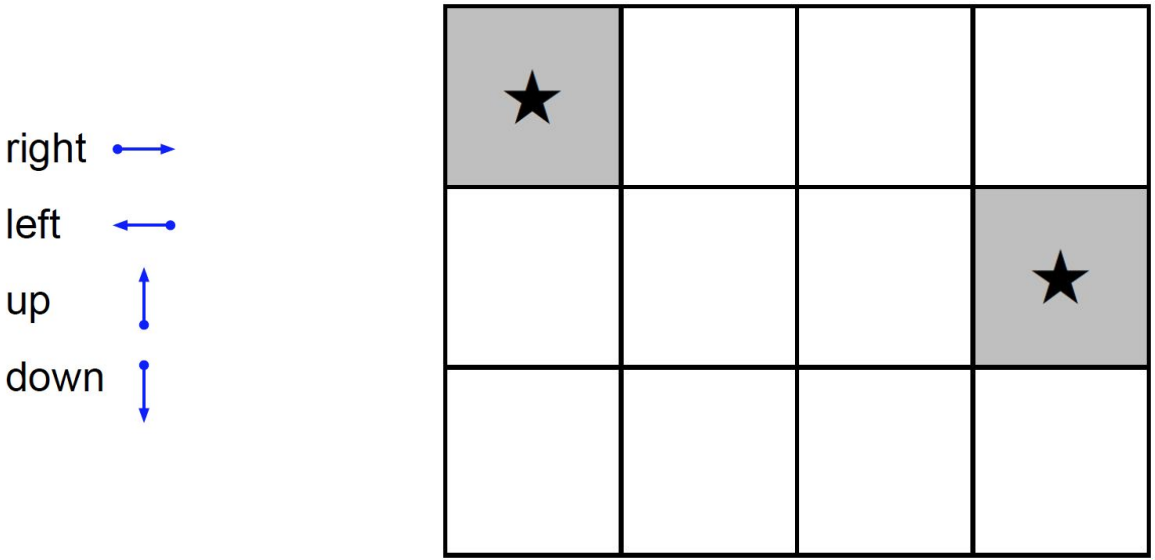
Example: Actions resulting from applying a **random policy** on this Grid World problem.



Slide concept: Serena Yeung, "Deep Reinforcement Learning". Stanford University CS231n, 2017.

MDP: Optimal Policy π*

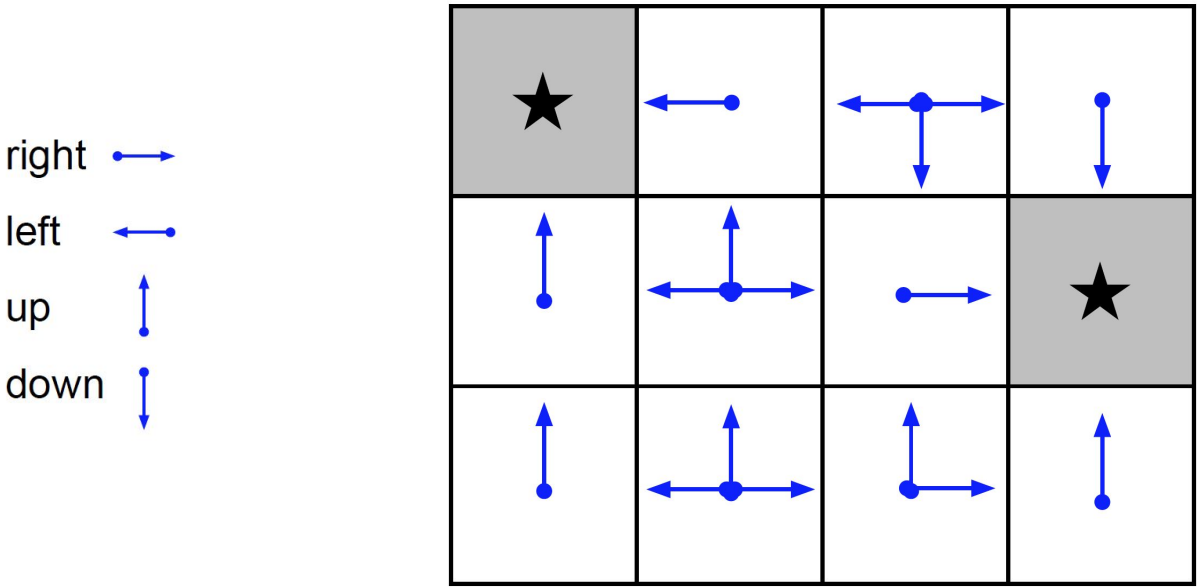
Exercise: Draw the actions resulting from applying an **optimal policy** in this Grid World problem.



Slide concept: Serena Yeung, "Deep Reinforcement Learning". Stanford University CS231n, 2017.

MDP: Optimal Policy π*

Solution: Draw the actions resulting from applying an optimal policy in this Grid World problem.



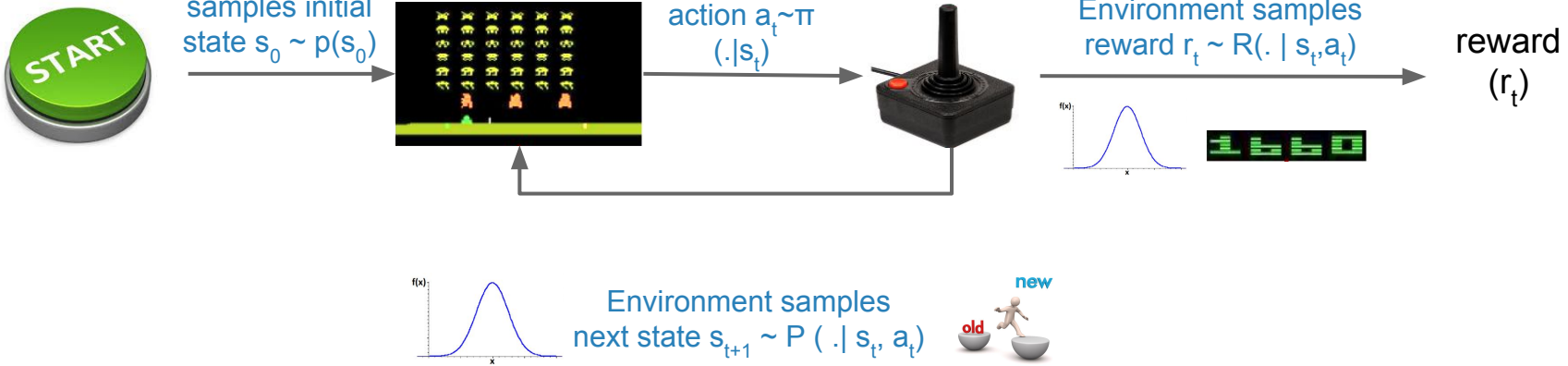
Slide concept: Serena Yeung, "Deep Reinforcement Learning". Stanford University CS231n, 2017.

MDP: Optimal Policy π*

How do we handle the randomness (initial state s_0, transition probabilities, action...) ?

GOAL: Find **policy π*** that maximizes the cumulative discounted reward:

$$\sum_{t \geq 0} \gamma^t r_t$$



MDP: Optimal Policy π*

How do we handle the randomness (initial state s_0, transition probabilities, action) ?

GOAL: Find **policy π*** that maximizes the cumulative discounted reward:

$$\sum_{t \geq 0} \gamma^t r_t$$

The **optimal policy π*** will maximize the expected sum of rewards:

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t \geq 0} \gamma^t r_t | \pi\right] \quad \text{with} \quad s_0 \sim p(s_0), a_t \sim \pi(\cdot|s_t), s_{t+1} \sim p(\cdot|s_t, a_t)$$

expected cumulative discounted reward

initial state

selected action at t

sampled state for t+1

Slide concept: Serena Yeung, "Deep Reinforcement Learning". Stanford University CS231n, 2017.

MDP: Policy: Value function $V^{\pi}(s)$

How to estimate how good state *s* is for a given policy π ?

With the **value function at state s, $V^{\pi}(s)$,** the expected cumulative reward from following policy π from state s.

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t \geq 0} \gamma^t r_t | s_0 = s, \pi\right]$$

"Expected cumulative reward…""

"...from following policy π from state s."

Slide concept: Serena Yeung, "Deep Reinforcement Learning". Stanford University CS231n, 2017.

MDP: Policy: Q-value function $Q^{\pi}(s,a)$

How to estimate how good a state-action pair (s,a) is for a given policy π ?

With the **Q-value function at state *s* and action *a*, $Q^{\pi}(s,a)$,** the expected cumulative reward from taking action *a* in state *s*, and then following policy π.

$$Q^{\pi}(s,a) = \mathbb{E}\left[\sum_{t \geq 0} \gamma^t r_t | s_0 = s, a_0 = a, \pi\right]$$

"Expected cumulative reward…""

"...from taking action a in state s and then following policy π."

Slide concept: Serena Yeung, "Deep Reinforcement Learning". Stanford University CS231n, 2017.

MDP: Policy: Optimal Q-value function $Q^*(s,a)$

(From the previous page)
Q-value function

$$Q^{\pi}(s,a) = \mathbb{E}\left[\sum_{t \geq 0} \gamma^t r_t | s_0 = s, a_0 = a, \pi\right]$$

The **optimal Q-value function at state s and action, $Q^*(s,a)$,** is the **maximum expected cumulative reward** achievable from a given (state, action) pair:

$$Q^*(s,a) = \max_{\pi} \mathbb{E}\left[\sum_{t \geq 0} \gamma^t r_t | s_0 = s, a_0 = a, \pi\right]$$

choose the **policy** that maximizes the expected cumulative reward

Slide concept: Serena Yeung, "Deep Reinforcement Learning". Stanford University CS231n, 2017.

MDP: Policy: Bellman equation

(From the previous page)
Optimal Q-value function

$$Q^*(s,a) = \max_\pi \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | s_0 = s, a_0 = a, \pi\right]$$

$Q^*(s,a)$ satisfies the following **Bellman equation:**

$$Q^*(s,a) = \mathbb{E}_{s'\sim\mathcal{E}}\left[r + \gamma \max_{a'} Q^*(s',a') | s,a\right]$$

- $Q^*(s,a)$: Maximum expected cumulative reward for considered pair (s,a)
- $\mathbb{E}_{s'\sim\mathcal{E}}$: Expectation across possible future states s' (randomness)
- r: reward for considered pair (s,a)
- γ: discount factor
- $\max_{a'} Q^*(s',a')$: Maximum expected cumulative reward for future pair (s',a')

FUTURE REWARD

Slide concept: Serena Yeung, "Deep Reinforcement Learning". Stanford University CS231n, 2017.

MDP: Policy: Bellman equation

GOAL: Find **policy π*** that maximizes the cumulative discounted reward:

$$\sum_{t \geq 0} \gamma^t r_t$$

$Q^*(s,a)$ satisfies the following **Bellman equation:**

$$Q^*(s,a) = \mathbb{E}_{s' \sim \mathcal{E}} \left[r + \gamma \max_{a'} Q^*(s', a') | s, a \right]$$

select action a' that maximizes expected cumulative reward

The **optimal policy π*** corresponds to taking the best action in any state according to Q*.

Slide concept: Serena Yeung, "Deep Reinforcement Learning". Stanford University CS231n, 2017.

MDP: Policy: Solving the Optimal Policy

(From the previous page)
Bellman Equation

$$Q^*(s,a) = \mathbb{E}_{s' \sim \mathcal{E}} \left[r + \gamma \max_{a'} Q^*(s',a') | s, a \right]$$

Value iteration algorithm: Estimate the Bellman equation with an iterative update.

$$Q_{i+1}(s,a) = \mathbb{E} \left[r + \gamma \max_{a'} Q_i(s',a') | s, a \right]$$

Updated Q-value function

Current Q-value for future pair (s',a')

The iterative estimation $Q_i(s,a)$ will converge to the optimal $Q^*(s,a)$ as $i \rightarrow \infty$.

Slide concept: Serena Yeung, "Deep Reinforcement Learning". Stanford University CS231n, 2017.

MDP: Policy: Solving the Optimal Policy

This iterative approach is **not scalable** because it requires computing $Q_i(s,a)$ for every state-action pair.
Eg. If state is current game pixels, computationally unfeasible to compute $Q_i(s,a)$ for the entire state space !

$$Q_{i+1}(s,a) = \mathbb{E}\left[r + \gamma \max_{a'} Q_i(s',a') | s,a\right]$$

Updated Q-value for current pair (s,a)

Current Q-value for next pair (s',a')

$Q_i(s,a)$ will converge to the optimal $Q^*(s,a)$ as $i \rightarrow \infty$.

Slide concept: Serena Yeung, "Deep Reinforcement Learning". Stanford University CS231n, 2017.

MDP: Policy: Solving the Optimal Policy

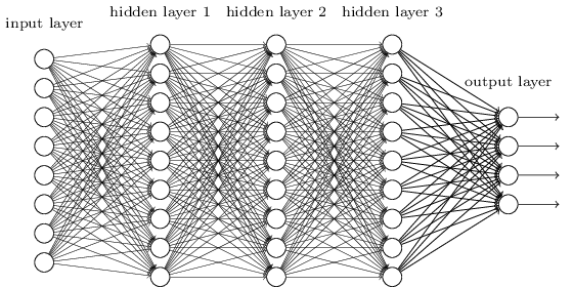
This iterative approach is **not scalable** because it requires computing Q(s,a) for every state-action pair.
Eg. If state is current game pixels, computationally unfeasible to compute Q(s,a) for the entire state space !

$$Q_{i+1}(s,a) = \mathbb{E}\left[r + \gamma \max_{a'} Q_i(s',a') | s,a\right]$$

Solution: Use a **deep neural network** as an function approximator of Q*(s,a).

$$Q(s,a,\Theta) \approx Q^*(s,a)$$

Neural Network parameters



Slide concept: Serena Yeung, "Deep Reinforcement Learning". Stanford University CS231n, 2017.

Outline

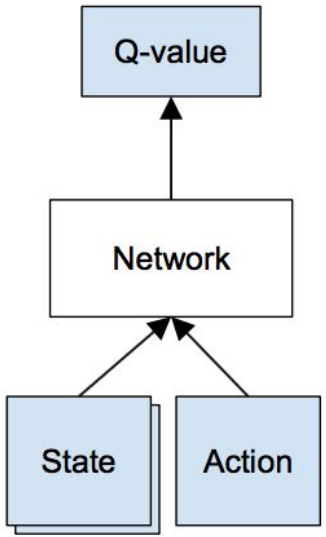
1. Motivation
2. Architecture
3. Markov Decision Process (MDP)
4. **Deep Q-learning**
    - Forward and Backward passes
    - DQN
    - Experience Replay
    - Examples
5. RL Frameworks
6. Learn more
    - Coming next…

Deep Q-learning

The function to approximate is a Q-function that satisfies the Bellman equation:

$Q(s,a,\Theta) \approx Q^*(s,a)$

$$Q^*(s,a) = \mathbb{E}_{s' \sim \mathcal{E}} \left[r + \gamma \max_{a'} Q^*(s',a') | s, a \right]$$



Slide concept: Serena Yeung, "Deep Reinforcement Learning". Stanford University CS231n, 2017.

Deep Q-learning

The function to approximate is a Q-function that satisfies the Bellman equation:

$Q(s,a,\Theta) \approx Q^*(s,a)$

$$Q^*(s,a) = \mathbb{E}_{s' \sim \mathcal{E}} \left[r + \gamma \max_{a'} Q^*(s',a') | s,a \right]$$

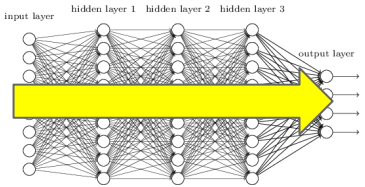
Forward Pass

Sample a (s,a) pair

Predicted Q-value with Θ_i

Loss function:

$$L_i(\theta_i) = \mathbb{E}_{s,a \sim \rho(\cdot)} \left[(y_i - Q(s,a;\theta_i))^2 \right]$$



$$y_i = \mathbb{E}_{s' \sim \mathcal{E}} \left[r + \gamma \max_{a'} Q(s',a';\theta_{i-1}) | s,a \right]$$

Sample a future state s'

Predict Q-value with Θ_{i-1}

Slide concept: Serena Yeung, "Deep Reinforcement Learning". Stanford University CS231n, 2017.

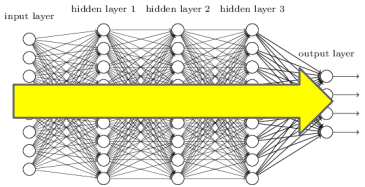
Deep Q-learning

Train the DNN to approximate a Q-value function that satisfies the Bellman equation

$$Q^*(s,a) = \mathbb{E}_{s' \sim \mathcal{E}} \left[r + \gamma \max_{a'} Q^*(s', a') | s, a \right]$$

$$L_i(\theta_i) = \mathbb{E}_{s,a \sim \rho(\cdot)} \left[(y_i - Q(s, a; \theta_i))^2 \right]$$

$$y_i = \mathbb{E}_{s' \sim \mathcal{E}} \left[r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) | s, a \right]$$

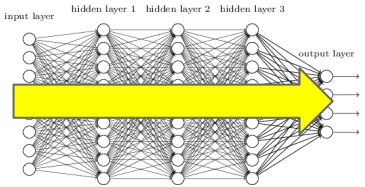


Slide concept: Serena Yeung, "Deep Reinforcement Learning". Stanford University CS231n, 2017.

Deep Q-learning

Must compute reward during training

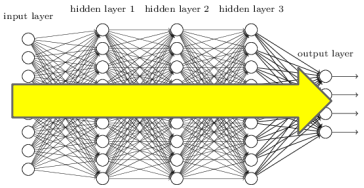
$$L_i(\theta_i) = \mathbb{E}_{s,a\sim\rho(\cdot)}\left[(y_i - Q(s,a;\theta_i))^2\right]$$



$$y_i = \mathbb{E}_{s'\sim\mathcal{E}}\left[r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) | s, a\right]$$

Slide concept: Serena Yeung, "Deep Reinforcement Learning". Stanford University CS231n, 2017.

Deep Q-learning



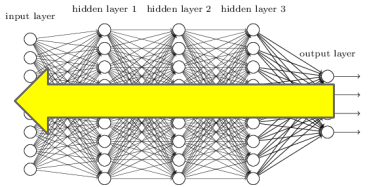
Forward Pass

Loss function: $$L_i(\theta_i) = \mathbb{E}_{s,a\sim\rho(\cdot)}\left[(y_i - Q(s,a;\theta_i))^2\right]$$

$$y_i = \mathbb{E}_{s'\sim\mathcal{E}}\left[r + \gamma \max_{a'} Q(s',a';\theta_{i-1})|s,a\right]$$

Backward Pass

Gradient update (with respect to Q-function parameters Θ):

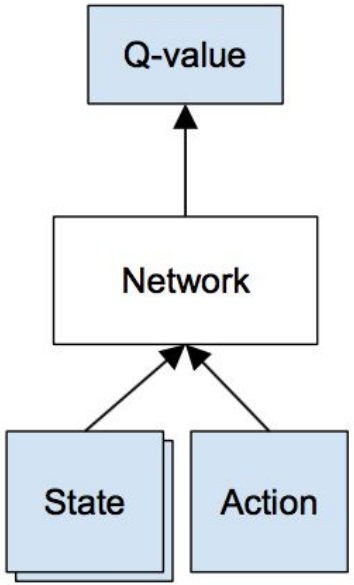


$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{s,a\sim\rho(\cdot);s'\sim\mathcal{E}}\left[r + \gamma \max_{a'} Q(s',a';\theta_{i-1}) - Q(s,a;\theta_i))\nabla_{\theta_i} Q(s,a;\theta_i)\right]$$

Slide concept: Serena Yeung, "Deep Reinforcement Learning". Stanford University CS231n, 2017.

Deep Q-learning: Deep Q-Network DQN

$Q(s,a,\Theta) \approx Q^*(s,a)$

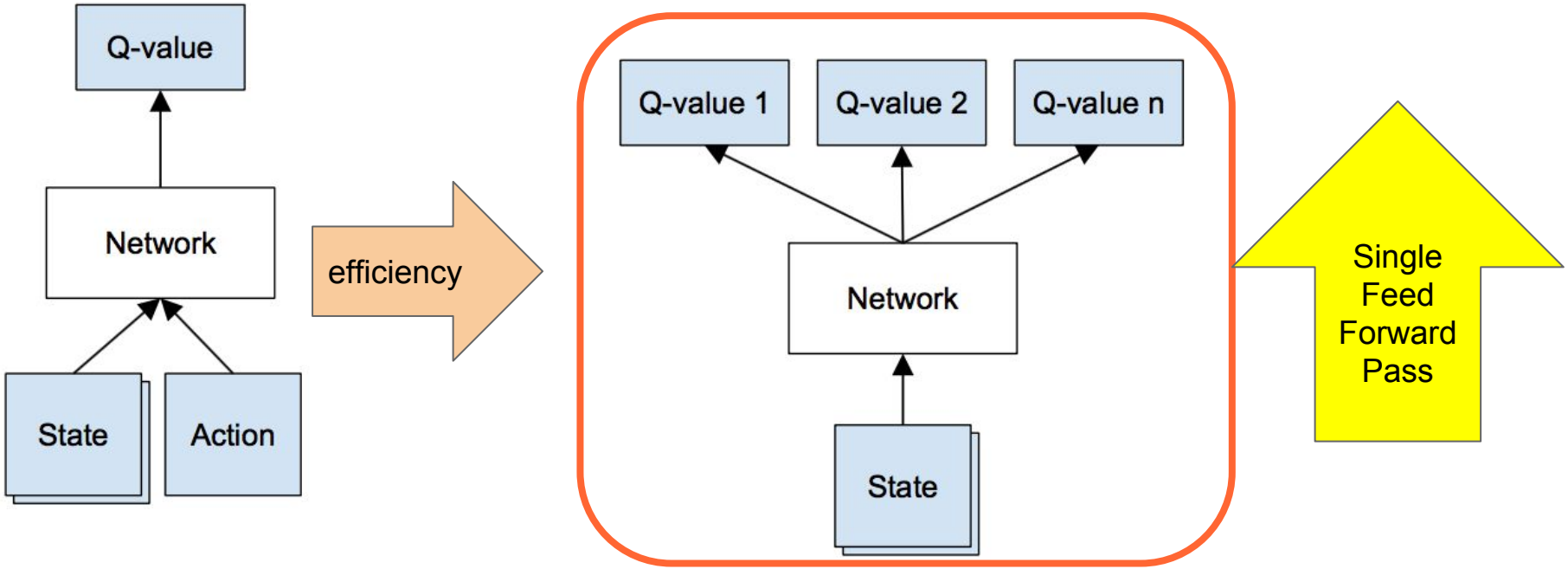


Source: Tambet Matiisen, Demystifying Deep Reinforcement Learning (Nervana)

Deep Q-learning: Deep Q-Network DQN

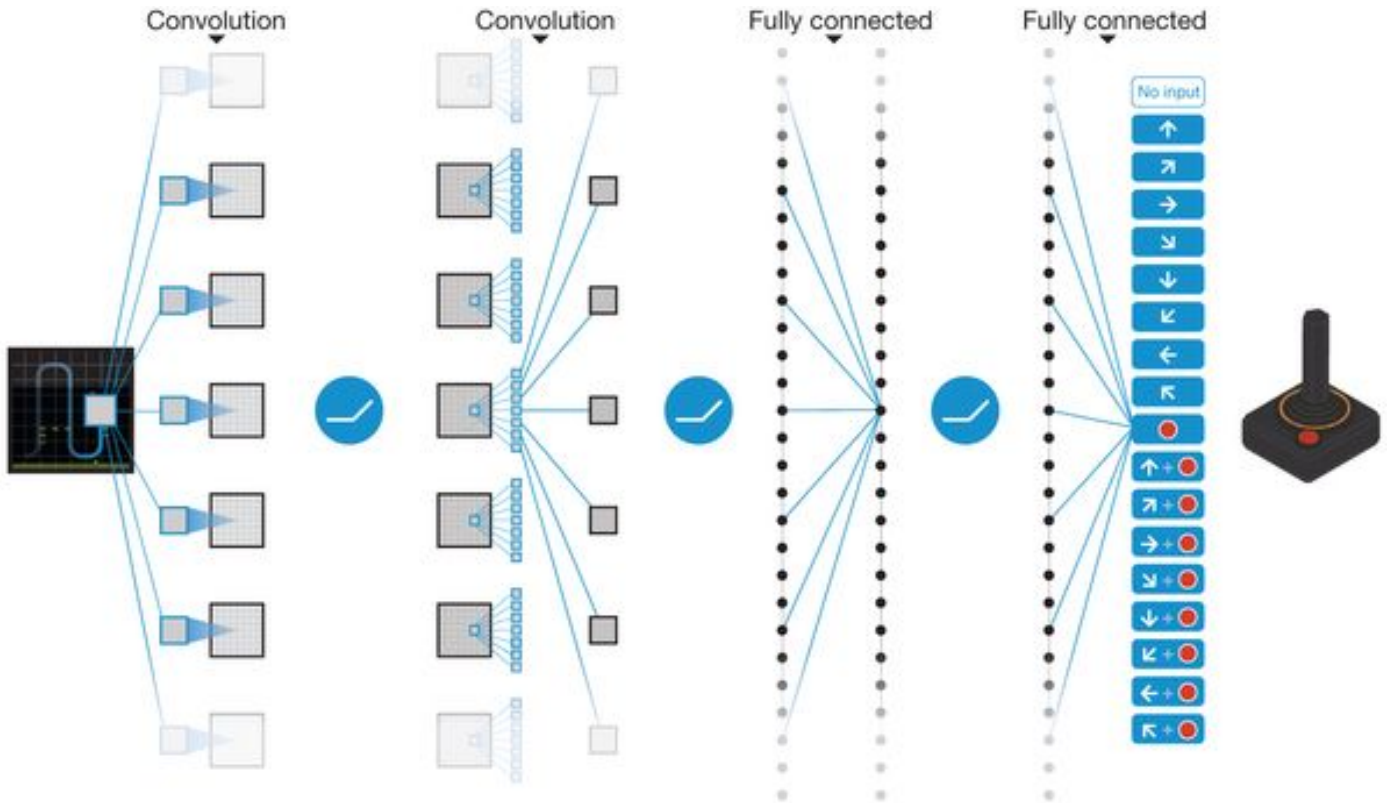
$Q(s,a,\Theta) \approx Q^*(s,a)$

A **single feedforward** pass to compute the Q-values for all actions from the current state (efficient)



Source: Tambet Matiisen, Demystifying Deep Reinforcement Learning (Nervana)

Deep Q-learning: Deep Q-Network DQN



Number of actions between 4-18, depending on the Atari game

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves et al. "Human-level control through deep reinforcement learning." *Nature* 518, no. 7540 (2015): 529-533.

Deep Q-learning: Deep Q-Network DQN

$Q(s_t, \leftarrow), Q(s_t, \rightarrow), Q(s_t, \uparrow), Q(s_t, \downarrow)$

FC-4 (Q-values)

FC-256

32 4x4 conv, stride 2

16 8x8 conv, stride 4

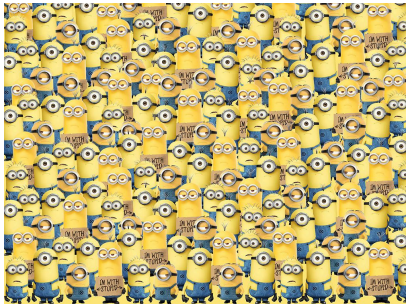
Current state s_t: 84x84x4 stack of last 4 frames
(after RGB->grayscale conversion, downsampling, and cropping)

Slide concept: Serena Yeung, "Deep Reinforcement Learning". Stanford University CS231n, 2017.

Deep Q-learning: Experience Replay

Learning from batches of **consecutive samples** is problematic:

- Samples are too correlated ➡ inefficient learning



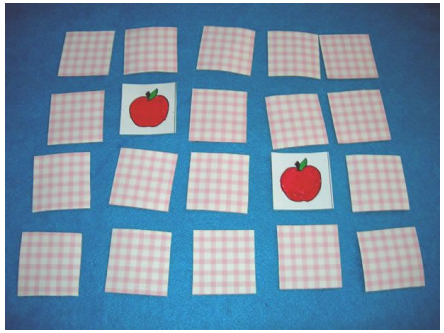
- Q-network parameters determine the next training samples ➡ can lead to bad feedback loops.

Slide concept: Serena Yeung, "Deep Reinforcement Learning". Stanford University CS231n, 2017.

Deep Q-learning: Experience Replay

Experience replay:

- Continually update a replay **memory table** of transitions (s_t, a_t, r_t, s_{t+1}) as game (experience) episodes are played.



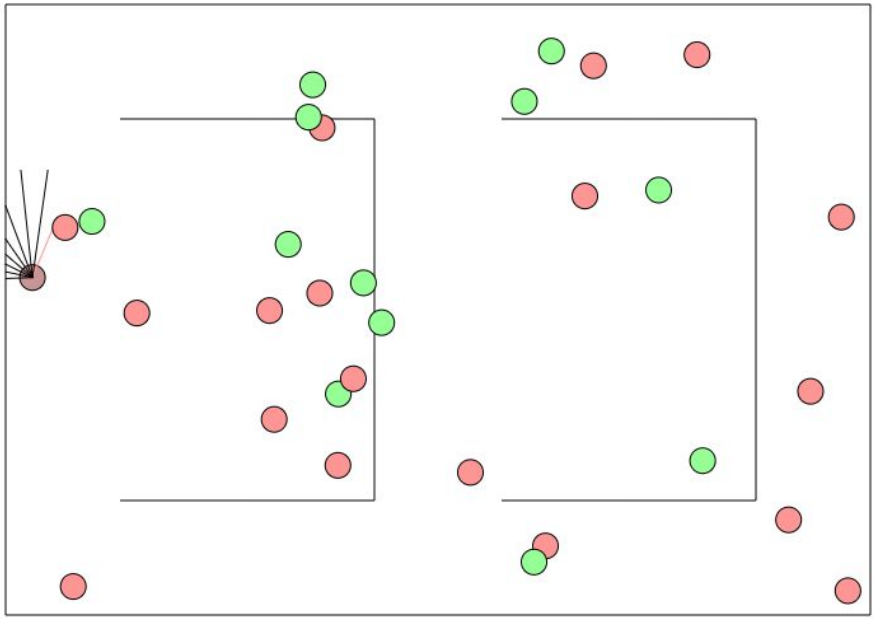
- Train a Q-network on **random minibatches** of transitions from the replay memory, instead of consecutiev samples.

Slide concept: Serena Yeung, "Deep Reinforcement Learning". Stanford University CS231n, 2017.

Deep Q-learning: Demo

ConvNetJS

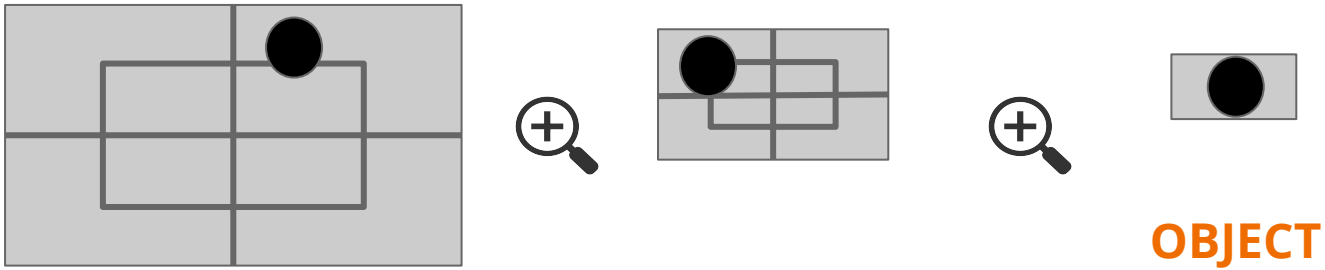
Deep Learning in your browser



Andrej Karpathy, [“ConvNetJS Deep Q Learning Demo”]

Deep Q-learning: DQN: Computer Vision

Method for performing hierarchical object detection in images guided by a **deep reinforcement learning agent**.

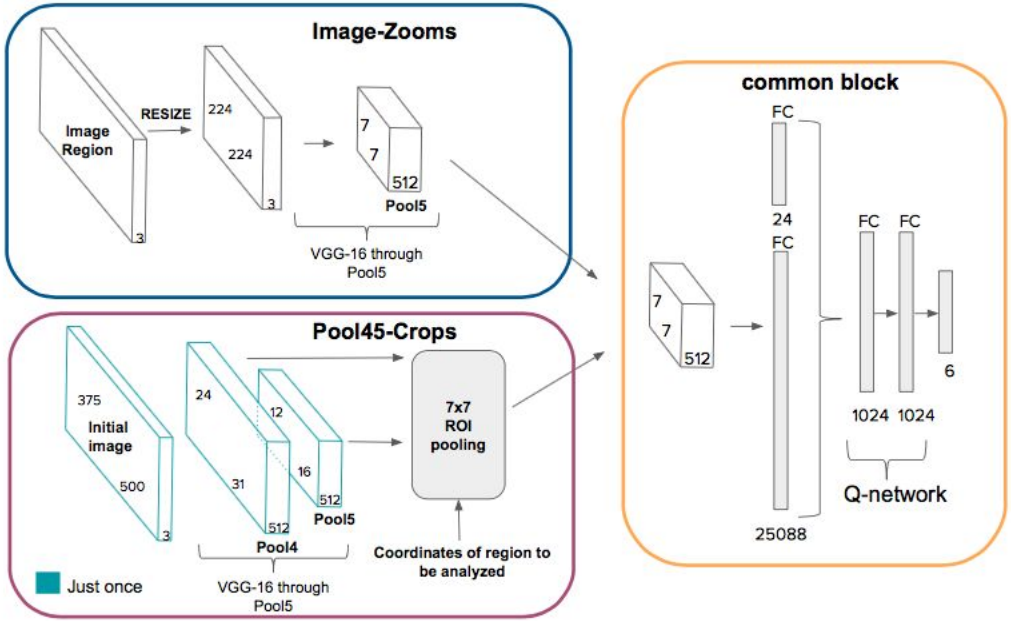


Miriam Bellver, Xavier Giro-i-Nieto, Ferran Marques, and Jordi Torres. "Hierarchical Object Detection with Deep Reinforcement Learning." Deep Reinforcement Learning Workshop NIPS 2016.

Deep Q-learning: DQN: Computer Vision

State: The agent will decide which action to choose based on:

- **visual description** of the current observed region
- **history vector** that maps past actions performed



Deep Q-learning: DQN: Computer Vision

Reward:

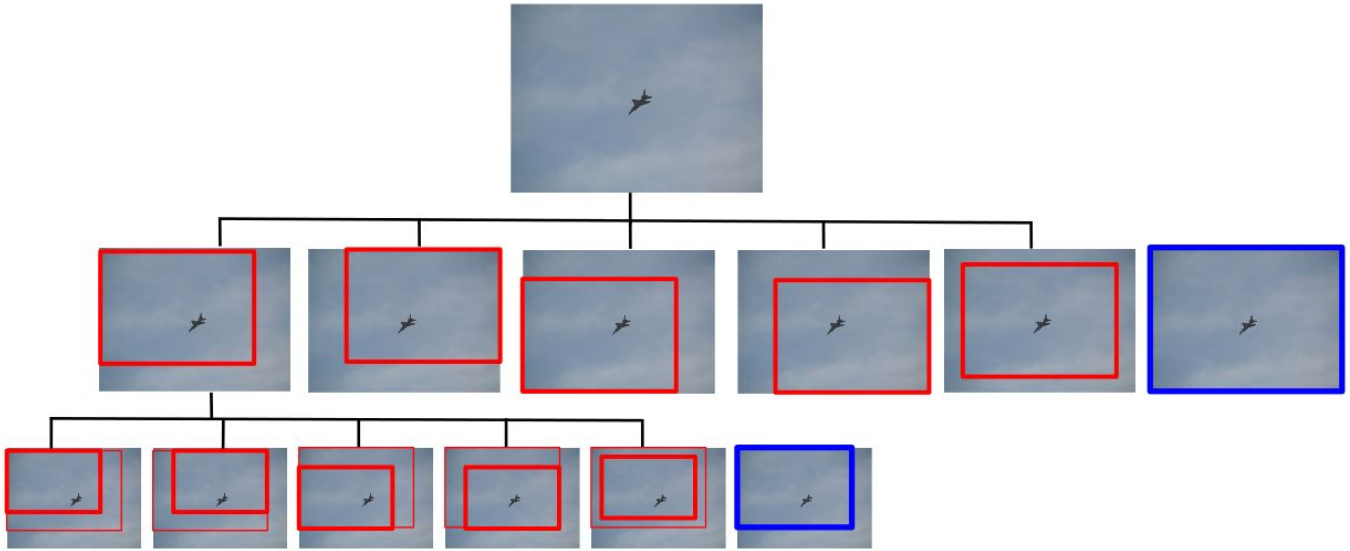
Reward for **movement actions** $$R_m(s, s') = sign(IoU(b', g) - IoU(b, g))$$

Reward for **terminal action** $$R_t(s, s') = \begin{cases} +\eta & if\ IoU(b, g) \geq \tau \\ -\eta & otherwise \end{cases}$$

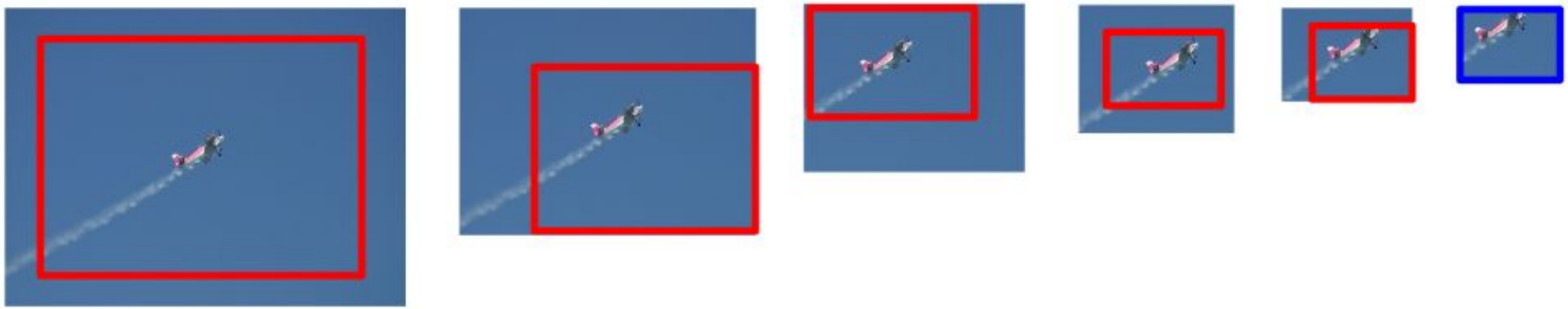
Deep Q-learning: DQN: Computer Vision

Actions: Two kind of actions:

- **movement actions**: to which of the 5 possible regions defined by the hierarchy to move
- **terminal action:** the agent indicates that the object has been found



Deep Q-learning: DQN: Computer Vision



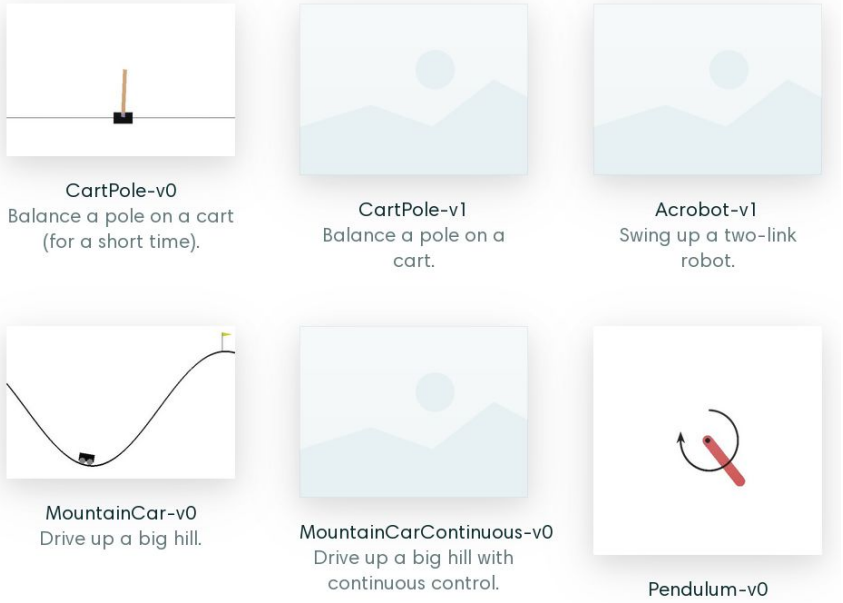
Miriam Bellver, Xavier Giro-i-Nieto, Ferran Marques, and Jordi Torres. "Hierarchical Object Detection with Deep Reinforcement Learning." Deep Reinforcement Learning Workshop NIPS 2016.

Outline

1. Motivation
2. Architecture
3. Markov Decision Process (MDP)
4. Deep Q-learning
5. **RL Frameworks**
6. Learn more

RL Frameworks

OpenAI Gym + keras-rl



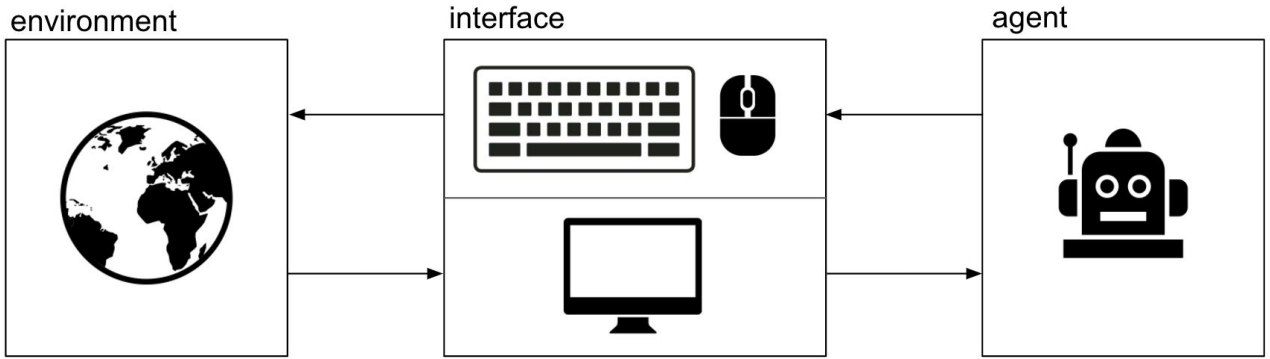
+

keras-rl

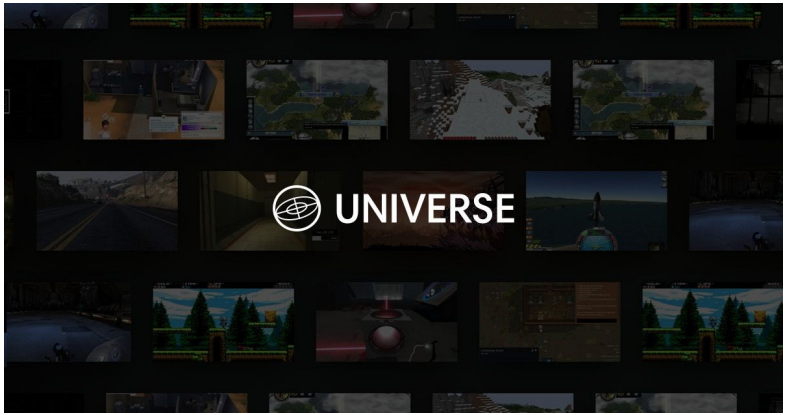
keras-rl implements some state-of-the art deep reinforcement learning algorithms in Python and seamlessly integrates with the deep learning library Keras. Just like Keras, it works with either Theano or TensorFlow, which means that you can train your algorithm efficiently either on CPU or GPU. Furthermore, keras-rl works with OpenAI Gym out of the box.

Slide credit: Míriam Bellver

RL Frameworks



OpenAI



OpenAI Universe environment

Outline

1. Motivation
2. Architecture
3. Markov Decision Process (MDP)
4. Deep Q-learning
5. RL Frameworks
6. **Learn more**

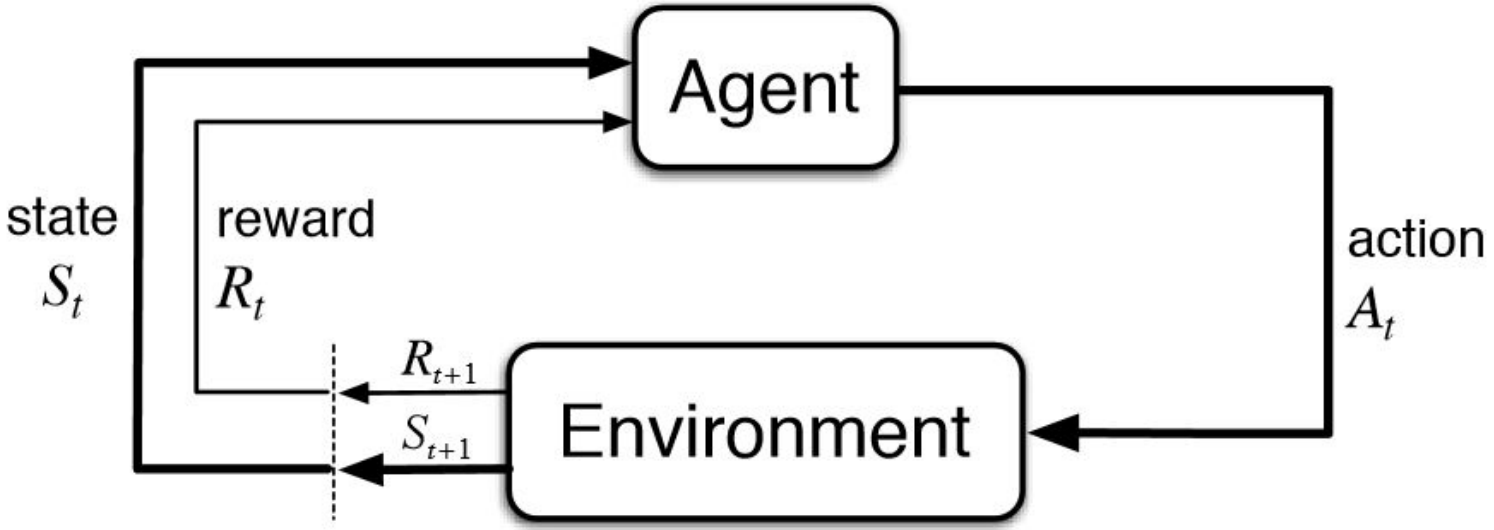
Learn more

Deep Learning TV,
"Reinforcement learning - Ep. 30"

Siraj Raval, Deep Q Learning for Video Games

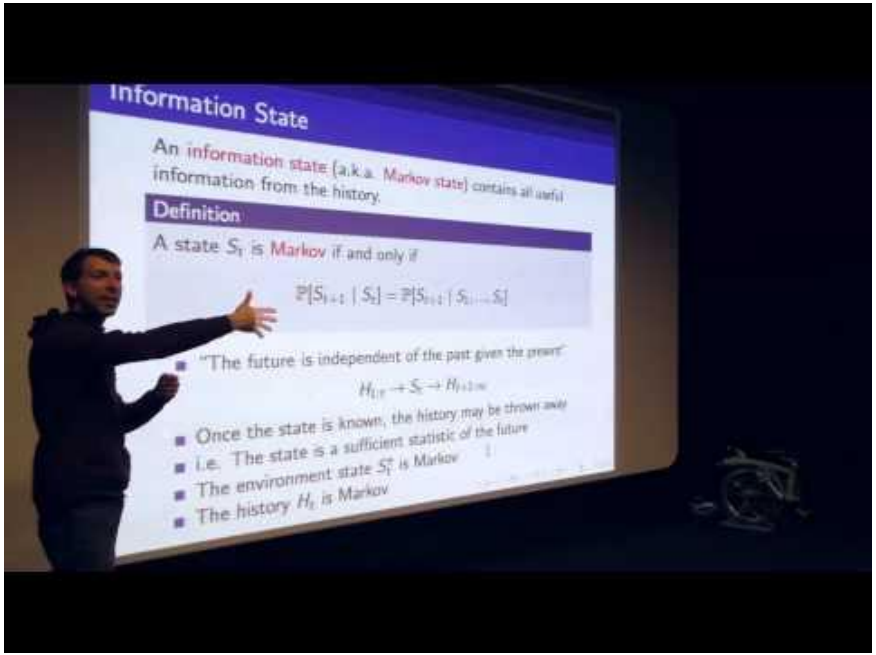
Learn more

Emma Brunskill, [Stanford CS234: Reinforcement Learning]



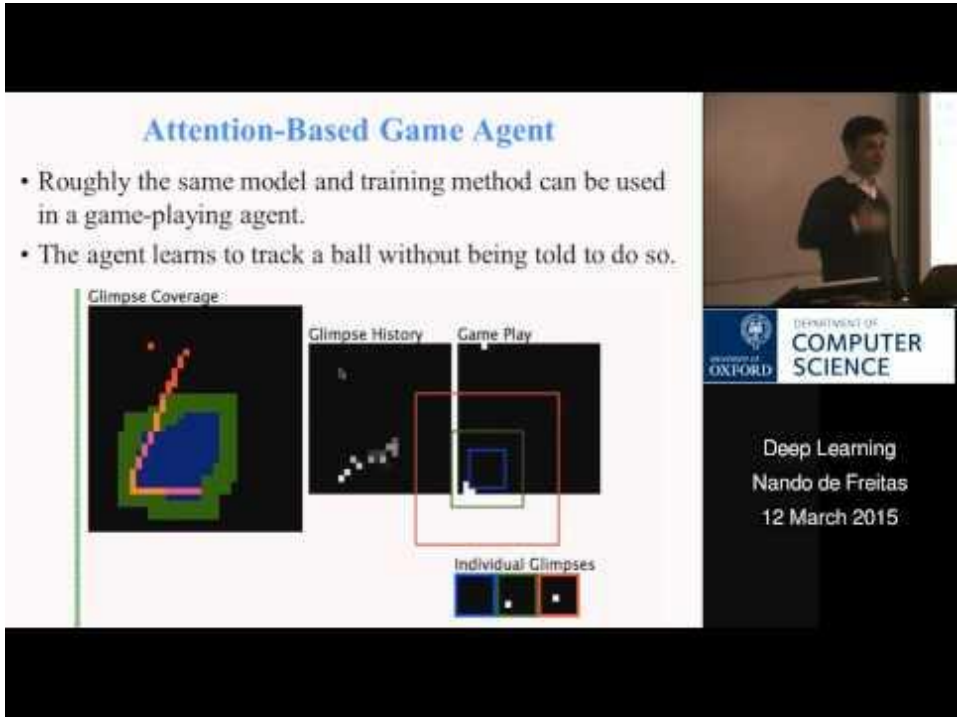
Learn more

David Silver, UCL COMP050, Reinforcement Learning



Learn more

Nando de Freitas, [“Machine Learning”](#) (University of Oxford)



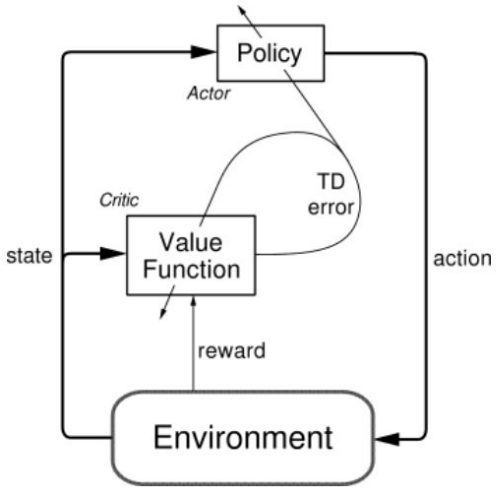
Learn more

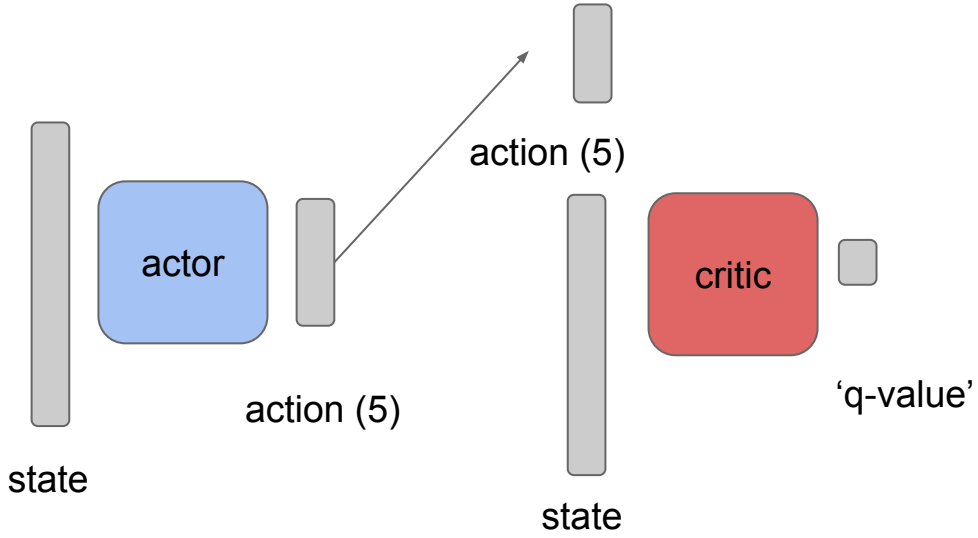
Pieter Abbeel and John Schulman, CS 294-112 Deep Reinforcement Learning, Berkeley.

Slides: "Reinforcement Learning - Policy Optimization" OpenAI / UC Berkeley (2017)

Learn more

Actor-Critic algorithm





actor performs an action

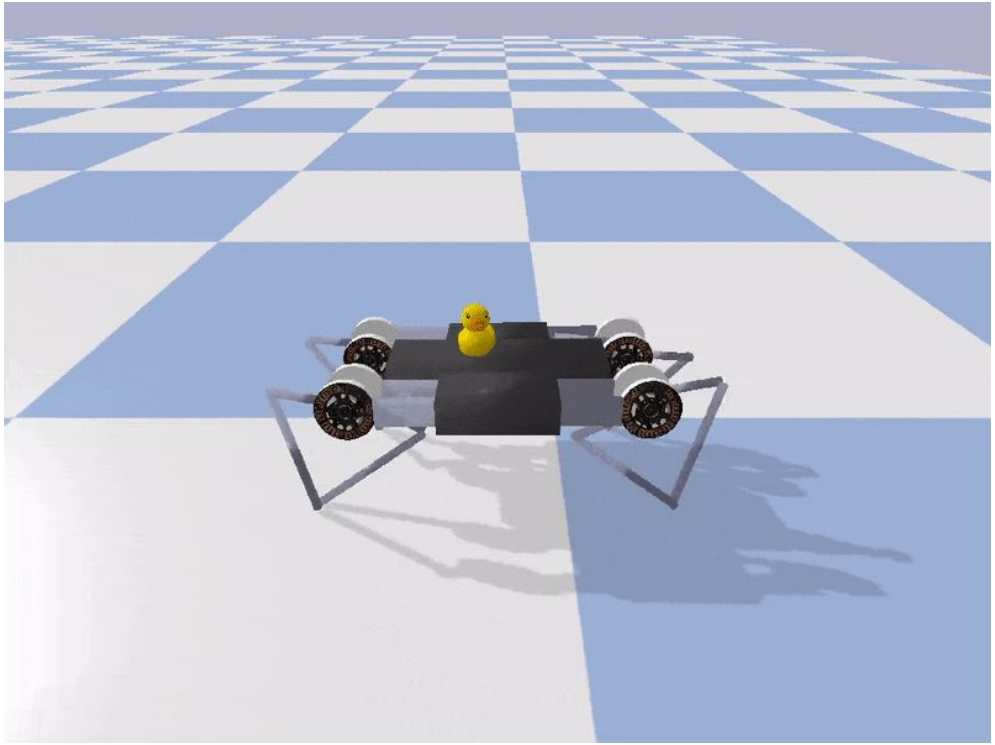
critic assesses how good the action was, and the gradients are used to train the actor and the critic

Slide credit: Míriam Bellver

Grondman, Ivo, Lucian Busoniu, Gabriel AD Lopes, and Robert Babuska. "A survey of actor-critic reinforcement learning: Standard and natural policy gradients." *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42, no. 6 (2012): 1291-1307.

Learn more

- Evolution Strategies



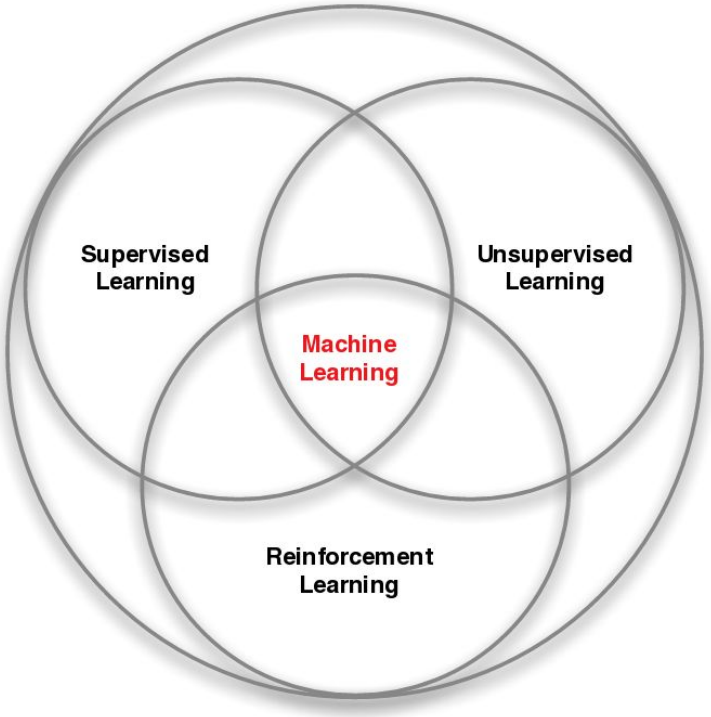
Outline

1. Motivation
2. Architecture
3. Markov Decision Process (MDP)
    - Policy
    - Optimal Policy
    - Value Function
    - Q-value function
    - Optimal Q-value function
    - Bellman equation
    - Value iteration algorithm
4. Deep Q-learning
    - Forward and Backward passes
    - DQN
    - Experience Replay
    - Examples
5. RL Frameworks
6. Learn more
    - Coming next…

Conclusions

Reinforcement Learning

- There is no supervisor, only reward signal
- Feedback is delayed, not instantaneous
- Time really matters (sequential, non i.i.d data)



Slide credit: UCL Course on RL by David Silver

Coming next...

Google DeepMind

Artificial intelligence (AI)

Google buys UK artificial intelligence startup Deepmind for £400m

Google makes its biggest EU purchase yet with the technology that aims to make computers think like humans

Samuel Gibbs

Monday 27 January 2014 13.23 GMT

This article is 2 years old

1046 186

https://www.theguardian.com/technology/2014/jan/27/google-acquires-uk-artificial-intelligence-startup-deepmind

Coming next...

AlphaGo

Google DeepMind

Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M. and Dieleman, S., 2016. Mastering the game of Go with deep neural networks and tree search. *Nature*, *529*(7587), pp.484-489

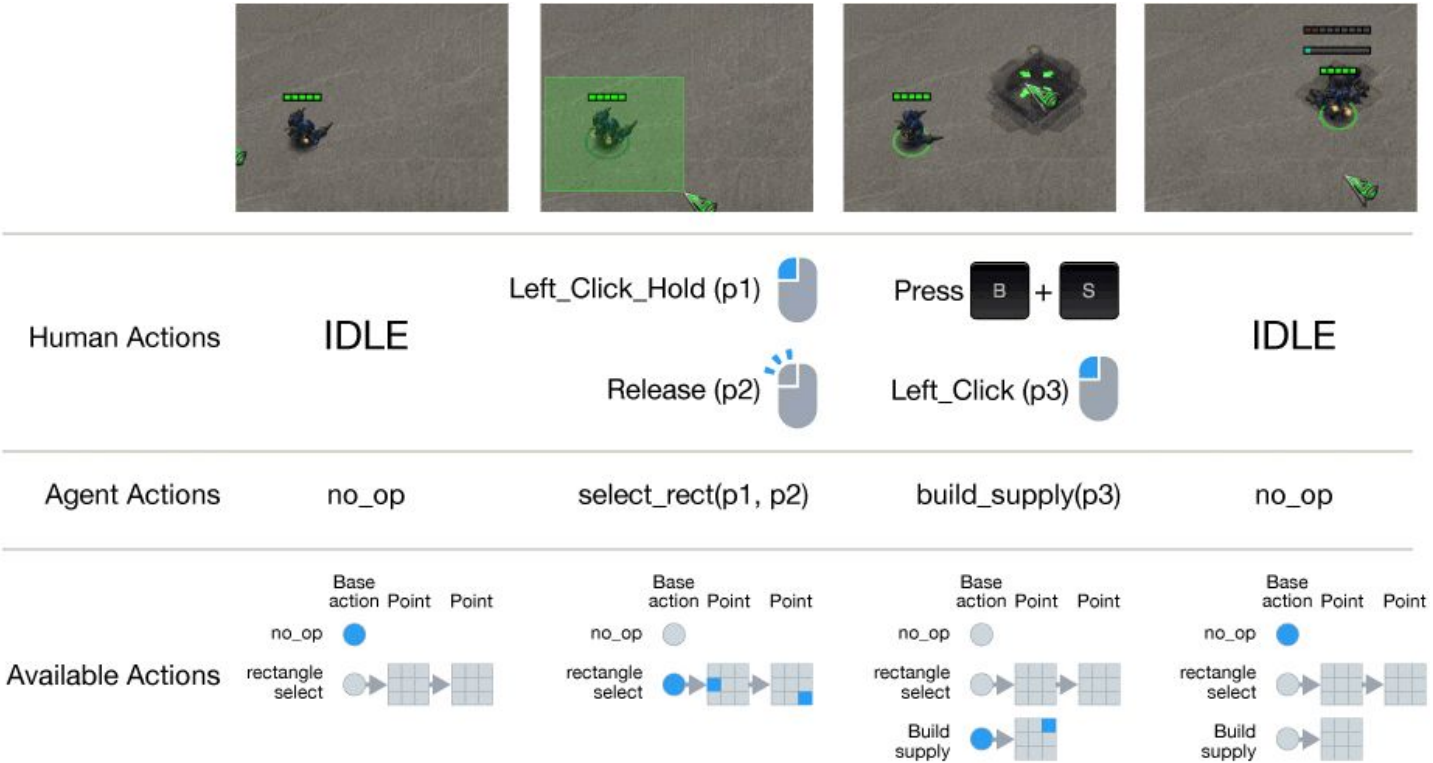
Coming next...

Greg Kohs, "AlphaGo" (2017)

Coming next...

Vinyals, O., Ewalds, T., Bartunov, S., Georgiev, P., Vezhnevets, A.S., Yeo, M., Makhzani, A., Küttler, H., Agapiou, J., Schrittwieser, J. and Quan, J., 2017. Starcraft ii: A new challenge for reinforcement learning. arXiv preprint arXiv:1708.04782. [Press release]

Coming next...



Vinyals, O., Ewalds, T., Bartunov, S., Georgiev, P., Vezhnevets, A.S., Yeo, M., Makhzani, A., Küttler, H., Agapiou, J., Schrittwieser, J. and Quan, J., 2017. Starcraft ii: A new challenge for reinforcement learning. arXiv preprint arXiv:1708.04782. [Press release]

Coming next...

telecom BCN

UPC

DEEP LEARNING FOR AI

Open Lecture

Monday 20 November 2017 @ Edifici Vèrtex (Auditori)

11:30am

Yannis Kalantidis

Facebook Research

12:00pm

Oriol Vinyals

Google Deepmind

Supported by

BSC Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA