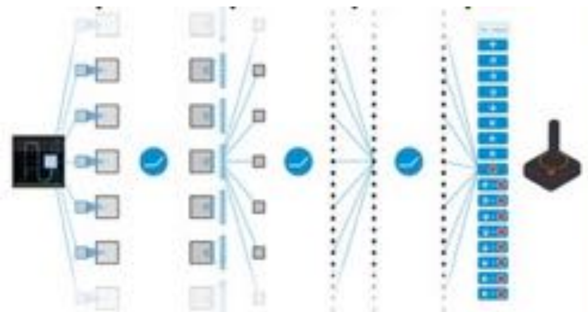
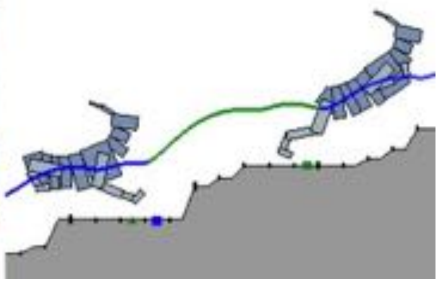
BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

Learning from Images

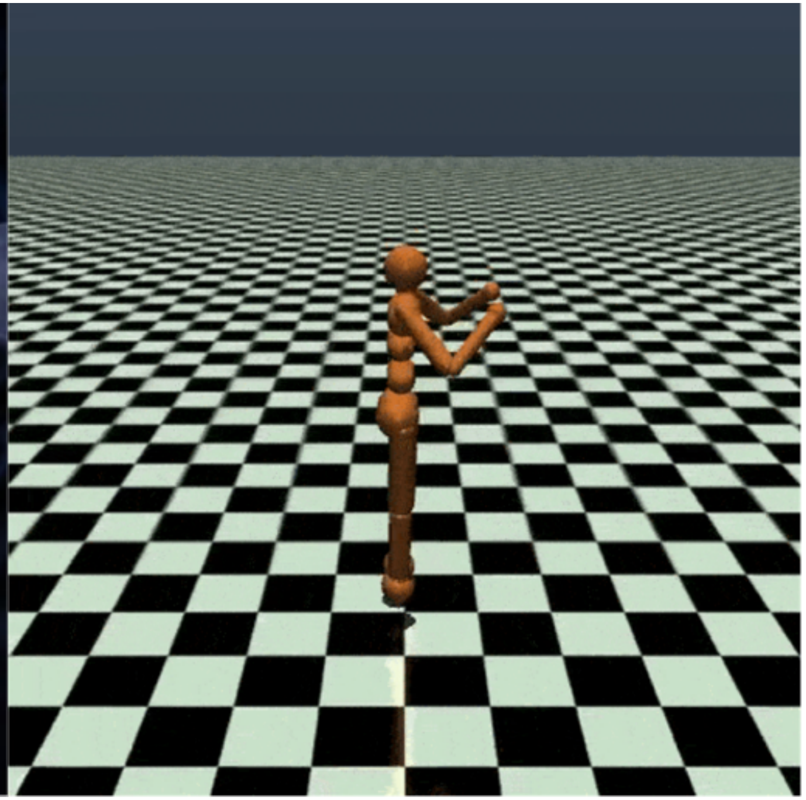
Introduction to Reinforcement Learning



Master DataScience
Winter term 2019/20

Prof. Dr. Kristian Hildebrand
khildebrand@beuth-hochschule.de

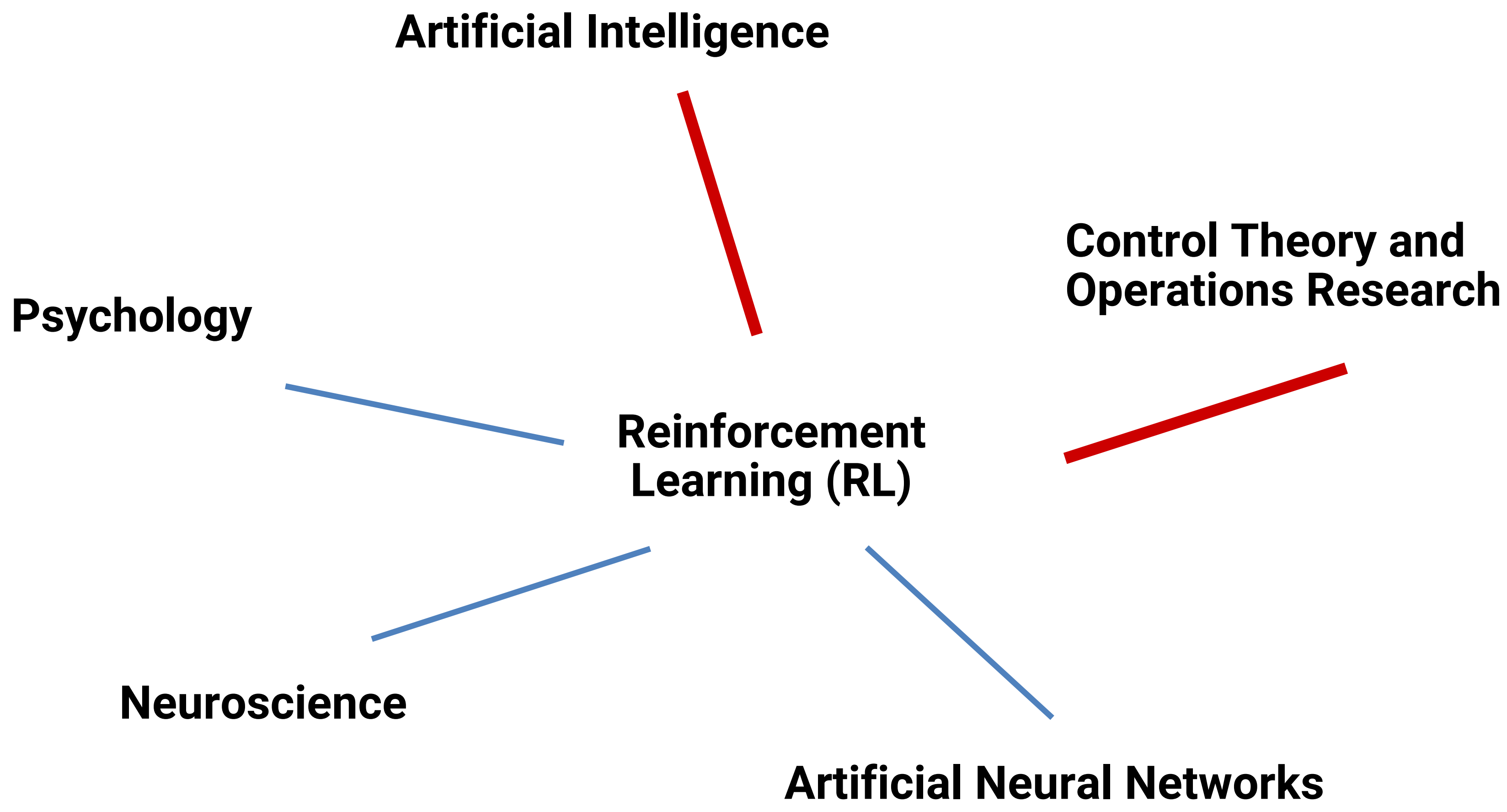
Learning to walk - https://www.youtube.com/watch?v=gn4nRCC9TwQ&feature=youtu.be

Learning to play go - https://www.youtube.com/watch?v=8tq1C8spV_g

"The Game of Go is the holy grail of artificial intelligence. Everything we've ever tried in AI, it just falls over when you try the game of Go."

David Silver, Lead Researcher for AlphaGo

Learning from Experience Plays a Role in …



Supervised learning:

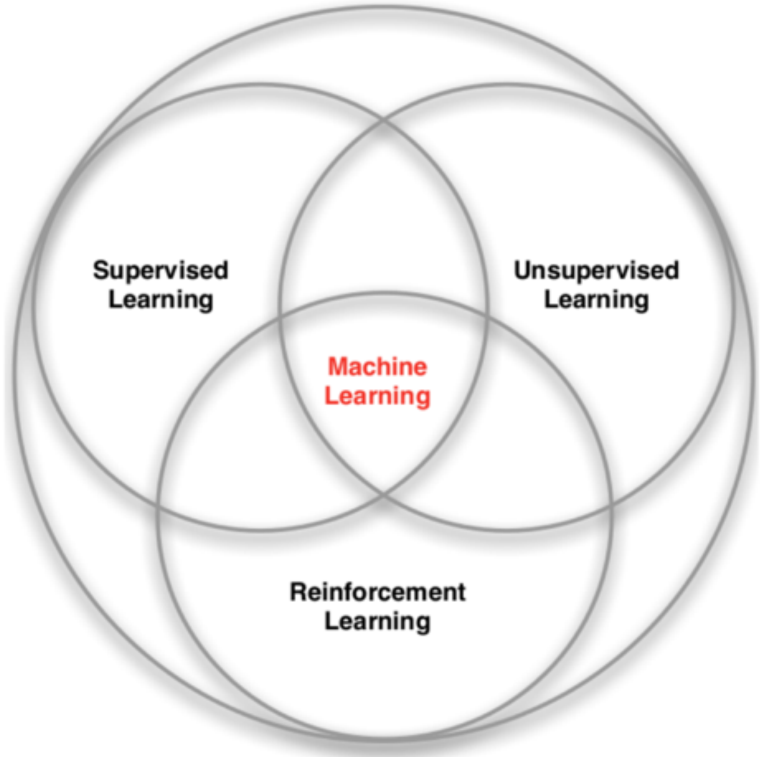
- learn from “labelled” data $(x_i, y_i)_{i=0}^{N}$

Unsupervised learning:

- learn from “unlabelled” data $(x_i)_{i=0}^{N}$

Semi-supervised learning:

- many unlabelled data, few labelled data



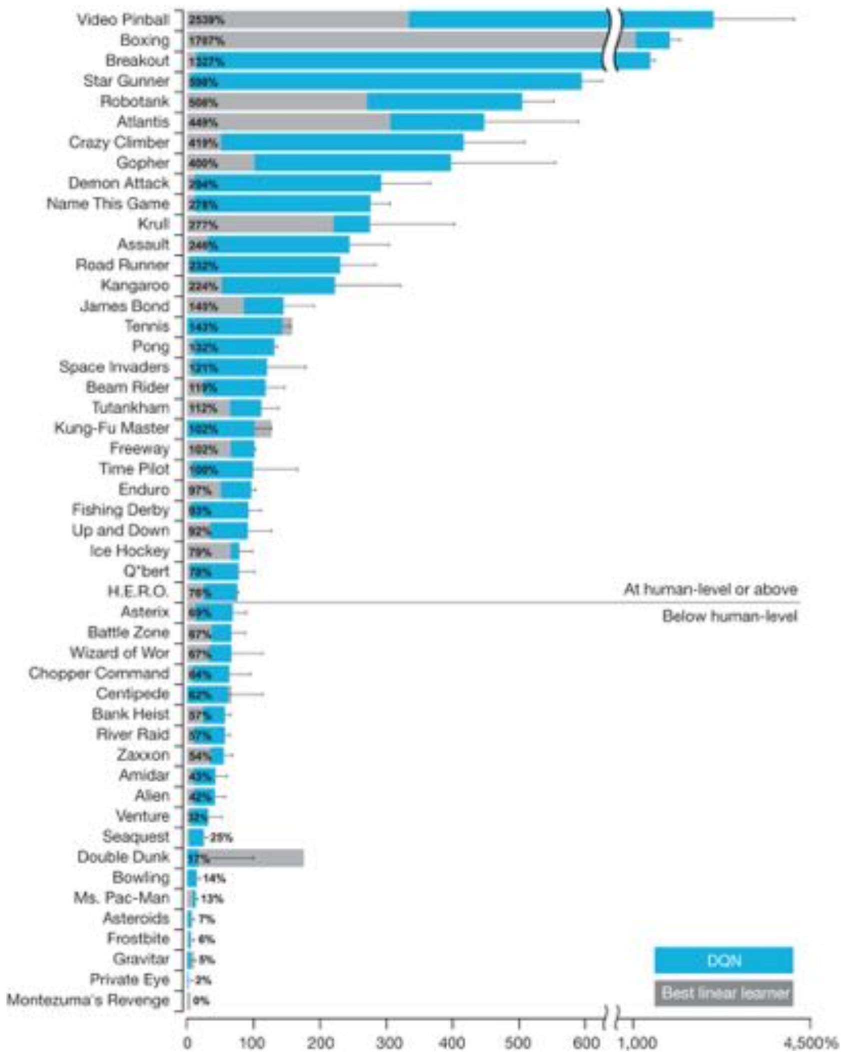
David Silver’s reinforcement learning lecture

Reinforcement learning:

State Action Reward

- learn from data (s_t, a_t, r_t, s_{t+1})
- learn a predictive model
- learn to predict reward
- learn a behavior (s → a) that maximizes the expected total reward

Comparison of the DQN agent with the best reinforcement learning methods[15] in the literature.



V Mnih *et al.* *Nature* **518**, 529-533 (2015) doi:10.1038/nature14236

nature

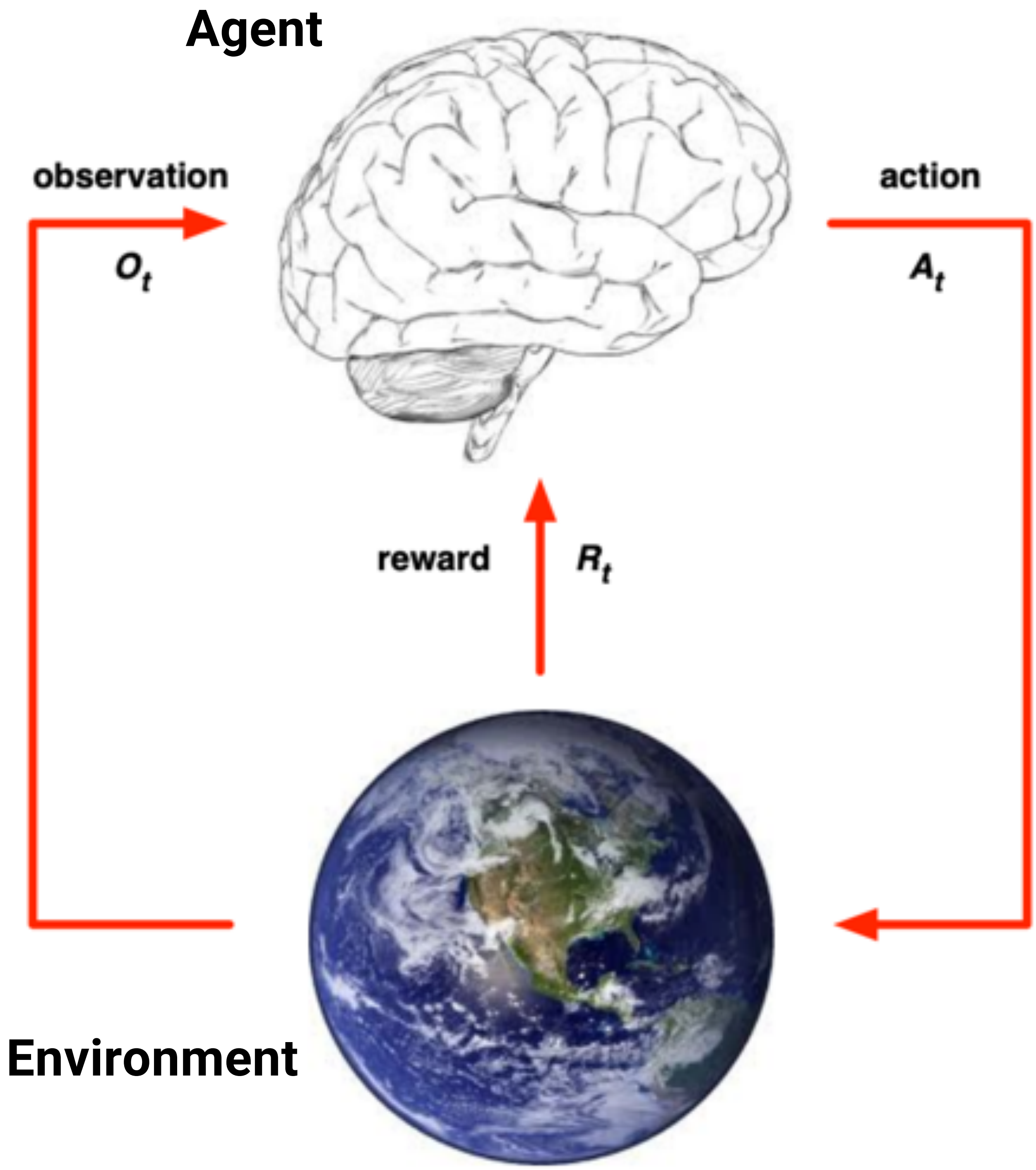
What makes reinforcement learning different from other machine learning paradigms?

- There is no supervisor, only a reward signal
- Feedback is delayed, not instantaneous
- Time really matters (sequential, non i.i.d data)
- Agent's actions affect the subsequent data it receives

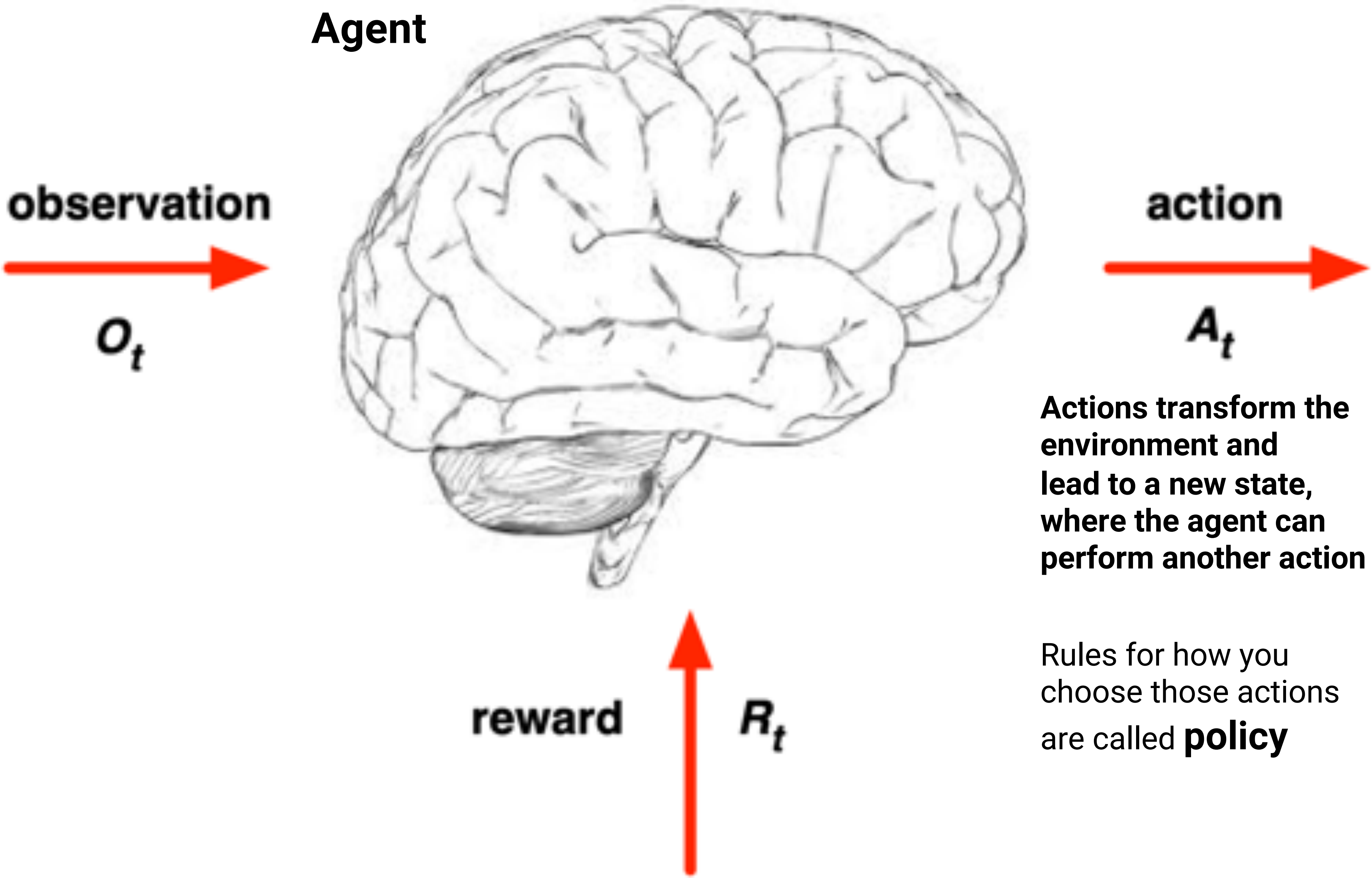
Examples of Reinforcement Learning

- Defeat the world champion at Backgammon
- Manage an investment portfolio
- Control a power station
- Make a humanoid robot walk
- Play many different Atari games better than humans
- SpaceX landing (Modern inverted pendulum)

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


Agent
observation
O_t
action
A_t
reward
R_t
Environment

What are those in the breakout game?



Actions transform the environment and lead to a new state, where the agent can perform another action

Rules for how you choose those actions are called **policy**

Rewards

- A reward R_t is a scalar feedback signal
- Indicates how well agent is doing at step t
- The agent's job is to maximise cumulative reward

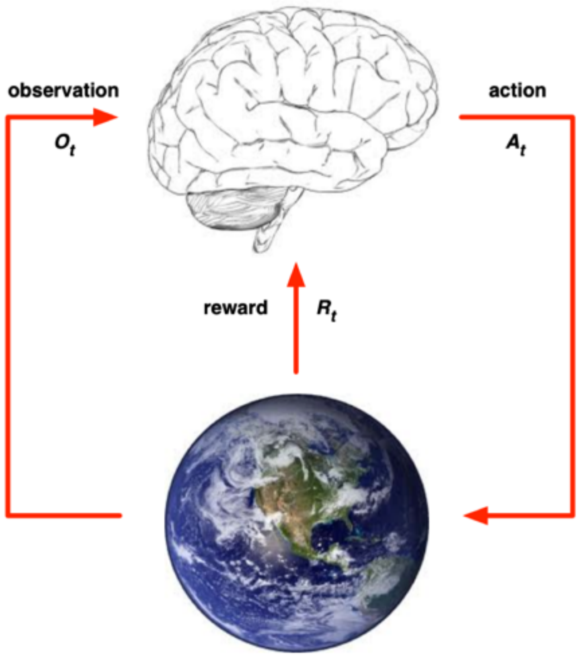
- Reinforcement learning is based on the reward hypothesis/function

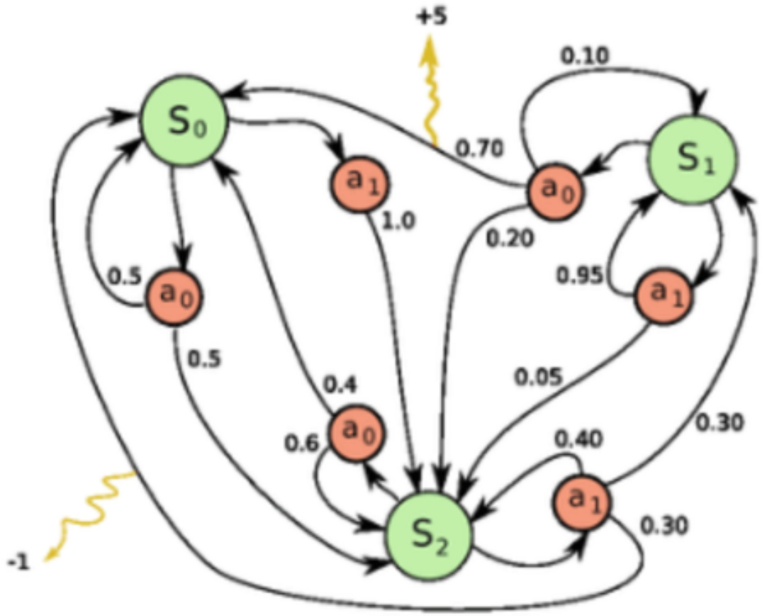
- All goals can be described by the **maximization of expected cumulative reward**

Examples of Rewards

- Defeat the world champion at Backgammon
  - +/−ve reward for winning/losing a game
- Manage an investment portfolio
  - +ve reward for each $ in bank
- Control a power station
  - +ve reward for producing power
  - −ve reward for exceeding safety thresholds
- Make a humanoid robot walk
  - +ve reward for forward motion
  - −ve reward for falling over
- Play many different Atari games better than humans
  - +/−ve reward for increasing/decreasing score

Markov Decision Process (Stochastic environment)





- set of states and actions, together with rules for transitioning from one state to another, make up a **Markov decision process**
- One **episode** of this process (e.g. one game) forms a finite sequence of states, actions and rewards
- episode ends with **terminal** state s_n (e.g. “game over” screen)

Discounted future reward

- To perform well in long-term, we need to take into account not only the immediate rewards, but also the future rewards
  - How should we go about that?

- Given one run of the Markov decision process, we can easily calculate the **total reward** for one episode:

$$R = r_1 + r_2 + r_3 + ... + r_n$$

- Given that, the **total future reward** from time point *t* onward can be expressed as:

$$R = r_t + r_{t+1} + r_{t+2} + ... + r_n$$

Discounted future reward

- Because environment is stochastic, rewards might not be the same when performing the same actions. The more into the future we go, the more it may diverge. For that reason it is common to use **discounted future reward** instead:

$$R = r_t + \lambda r_{t+1} + \lambda^2 r_{t+2} + ... + \lambda^{n-t} r_n$$

- Discount factor between 0 and 1 – the more into the future the reward is, the less we take it into consideration.

$$R_t = r_t + \lambda(r_{t+1} + \lambda r_{t+2} + ...)) = r_t + \lambda R_{t+1}$$

- A good strategy for an agent would be to **always choose an action that maximizes the (discounted) future reward**.

Q Learning

Q-Learning

- define a function *Q(s, a)* representing **the maximum discounted future reward when we perform action a in state s, and continue optimally from that point on**

$$Q(s_t, a_t) = \max R_{t+1}$$

- best possible score at the end of the game after performing action a **in state** s
- **It is called Q-function, because it represents the "quality" of a certain action in a given state.**

$$\pi = \arg\max_a Q(s, a)$$

π represents the policy, the rule how we choose an action in each state

Q-Learning

- Bellman Equation

$$Q(s,a) = r + \lambda \max_{a'} Q(s',a')$$

- Store Q-Values in Table *np.matrix(num_states, num_actions)*
- Q-learning: **iteratively approximate the Q-function using the Bellman equation**

Frozen lake example whiteboard

```
initialize Q[num_states,num_actions] arbitrarily
observe initial state s
repeat
      select and carry out an action a
      observe reward r and new state s'
      Q[s,a] = Q[s,a] + α(r + γ max_a' Q[s',a']) - Q[s,a])
      s = s'
until terminated
```

Over steps in the game

Deep Q Network (DQN)

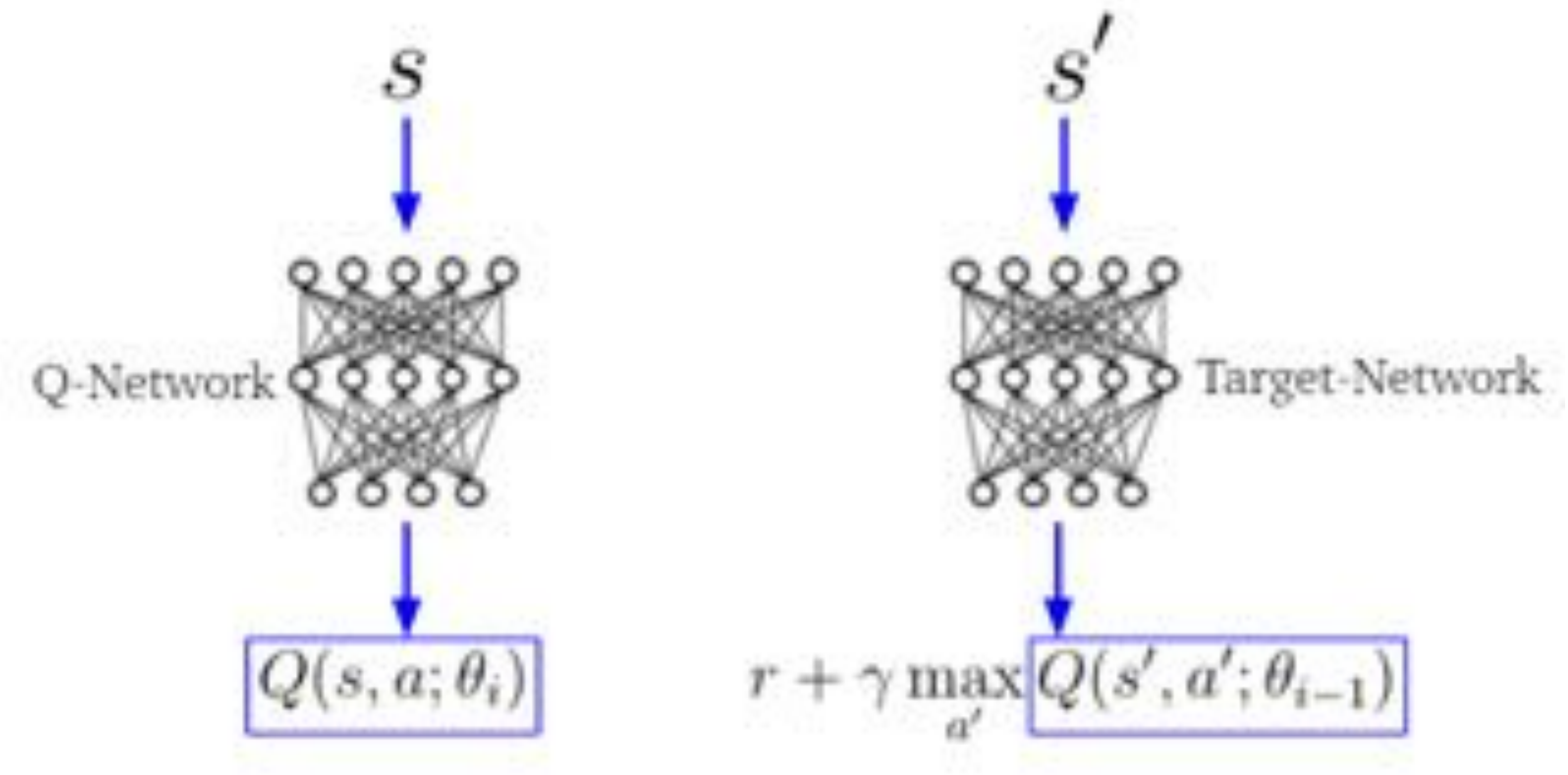
Deep Q Network

- State of the environment in Breakout game:
  - location of the paddle
  - location and direction of the ball
  - presence or absence of each individual brick

- Game specific – not a general good idea

- Better:
  - Screen pixels!
  - Game independent BUT many
  - 84×84 pixel gray scale -> 256^{84x84}
  - Q-Table does not scale

Deep Q-Network



$$L = \frac{1}{2}[r + \lambda \max_{a'} Q(s', a') - Q(s, a)]^2$$

target

prediction

Deep Q-Network Architecture

Layer	Input	Filter size	Stride	Num filters	Activation	Output
conv1	84x84x4	8x8	4	32	ReLU	20x20x32
conv2	20x20x32	4x4	2	64	ReLU	9x9x64
conv3	9x9x64	3x3	1	64	ReLU	7x7x64
fc4	7x7x64			512	ReLU	512
fc5	512			18	Linear	18

What is the difference to other ConvNet architectures?

No Pooling

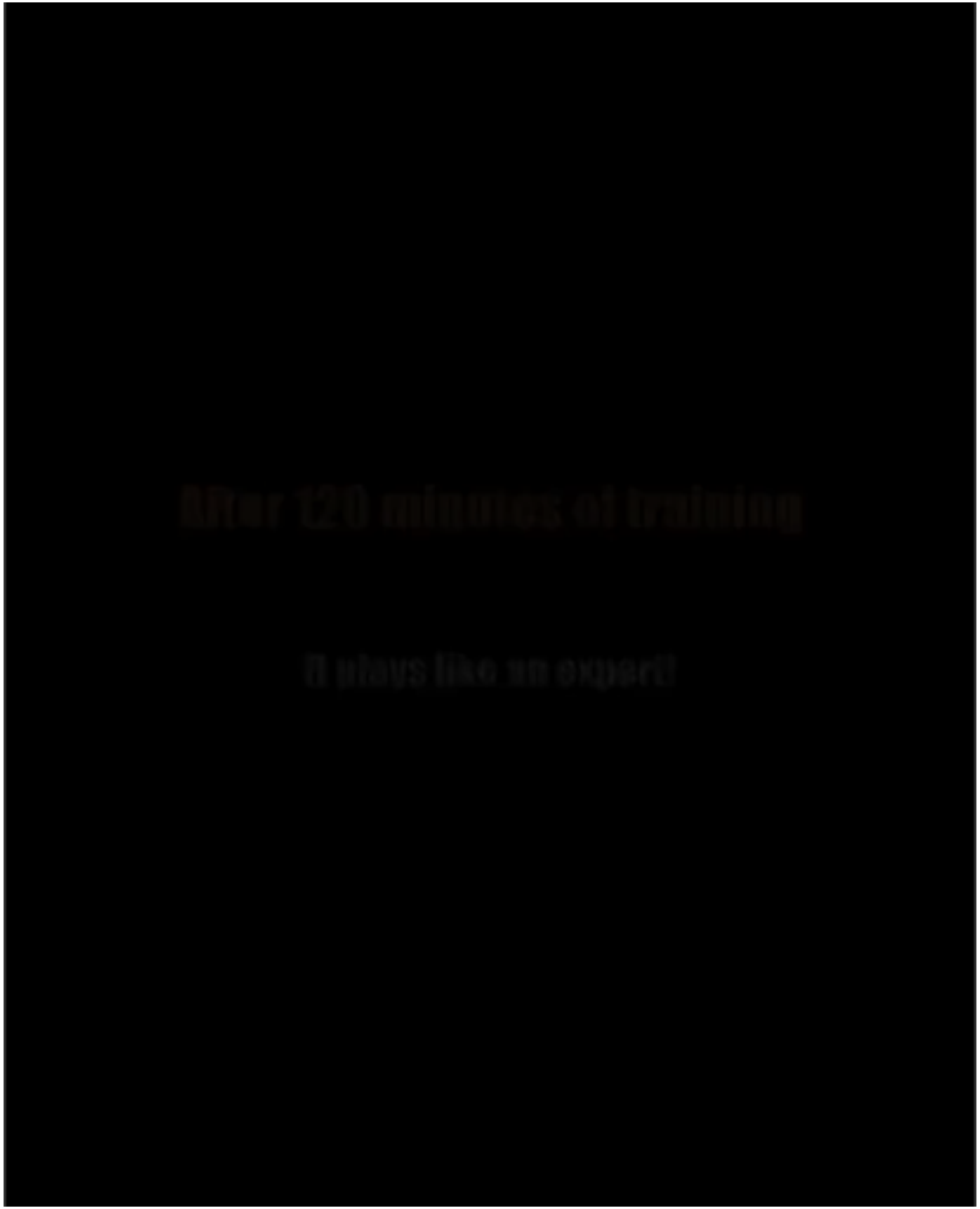
Why?

Locations are crucial in determining the potential reward

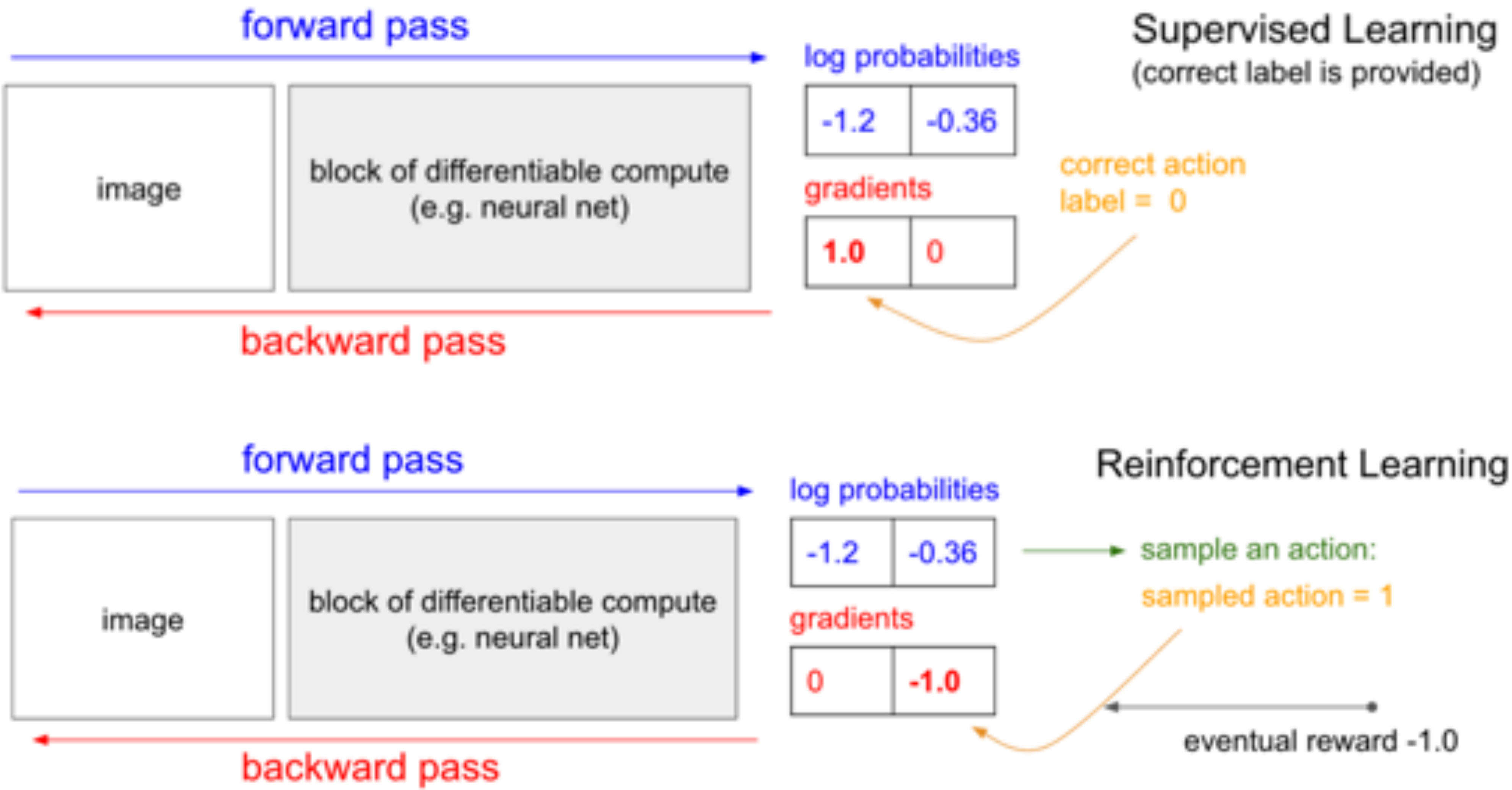
Reinforcement Learning can train computers to play ATARI games from raw game pixels

- Input frames to output action is called **policy network**
- method to train is called **policy gradients**
- Goal: **Improve the reward**

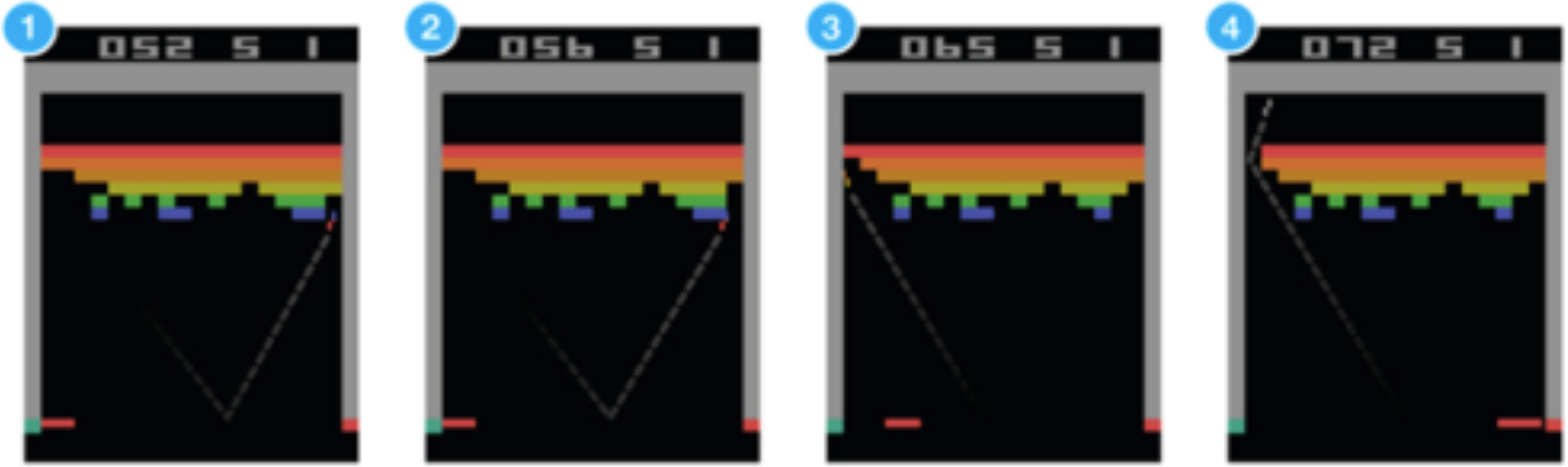
- Same model architecture was used to learn seven different games
- 3 of 7 performed even better than a human!



Reinforcement Learning



https://karpathy.github.io



DeepMind

Classification problem?

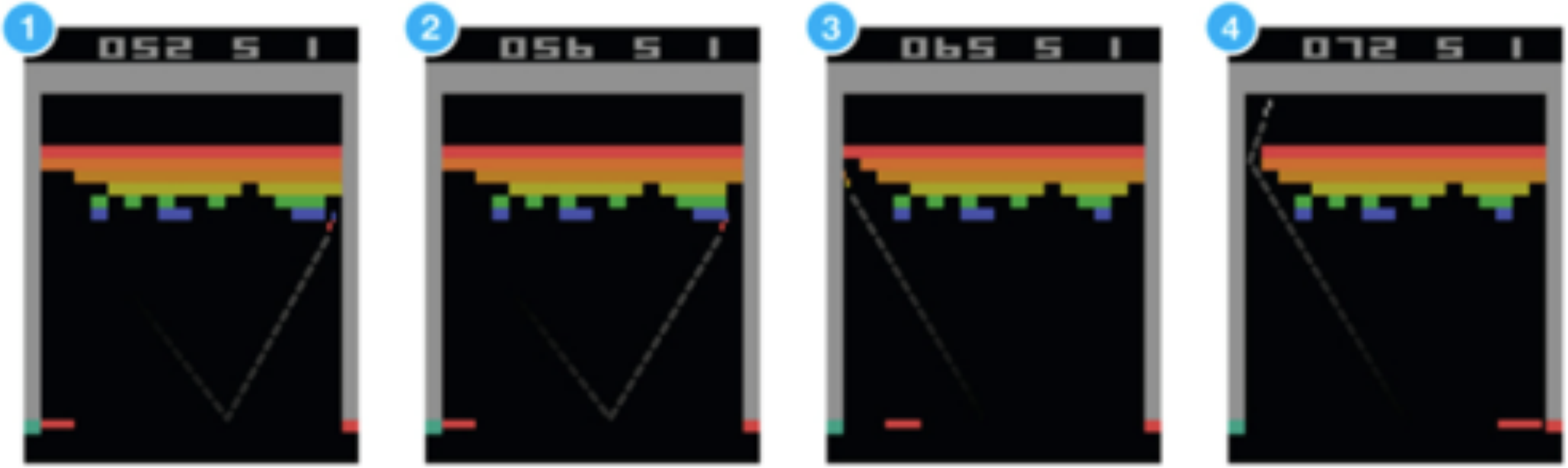
- For each game screen one decides whether to move left or right

What about training examples? By expert player? Bad Idea!

- In **supervised learning** one has a target label for each training example
- In **unsupervised learning** one has no labels at all
- In **reinforcement learning** one has sparse and time-delayed labels – **rewards**

Based only on those rewards the agent has to learn to behave in an environment.

Challenge: Which action belongs to which reward? **Credit assignment problem!**



DeepMind

Once we have a strategy to collect a number of rewards, should you stick with it or experiment with something that could result in even bigger rewards?

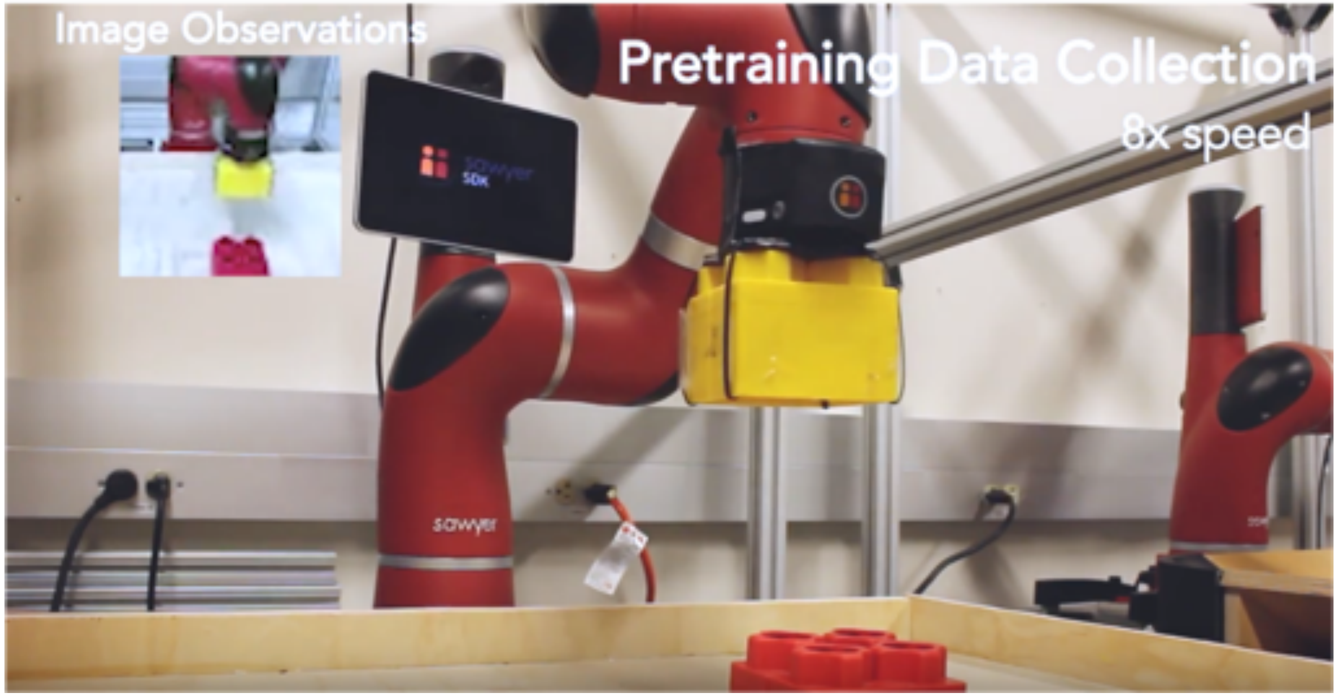
Should we exploit known working strategy or explore possibly better strategies?

Explore-Exploit Dilemma

Reinforcement learning is an important model of how we learn.

Credit assignment problems and exploration-exploitation dilemmas come up every day

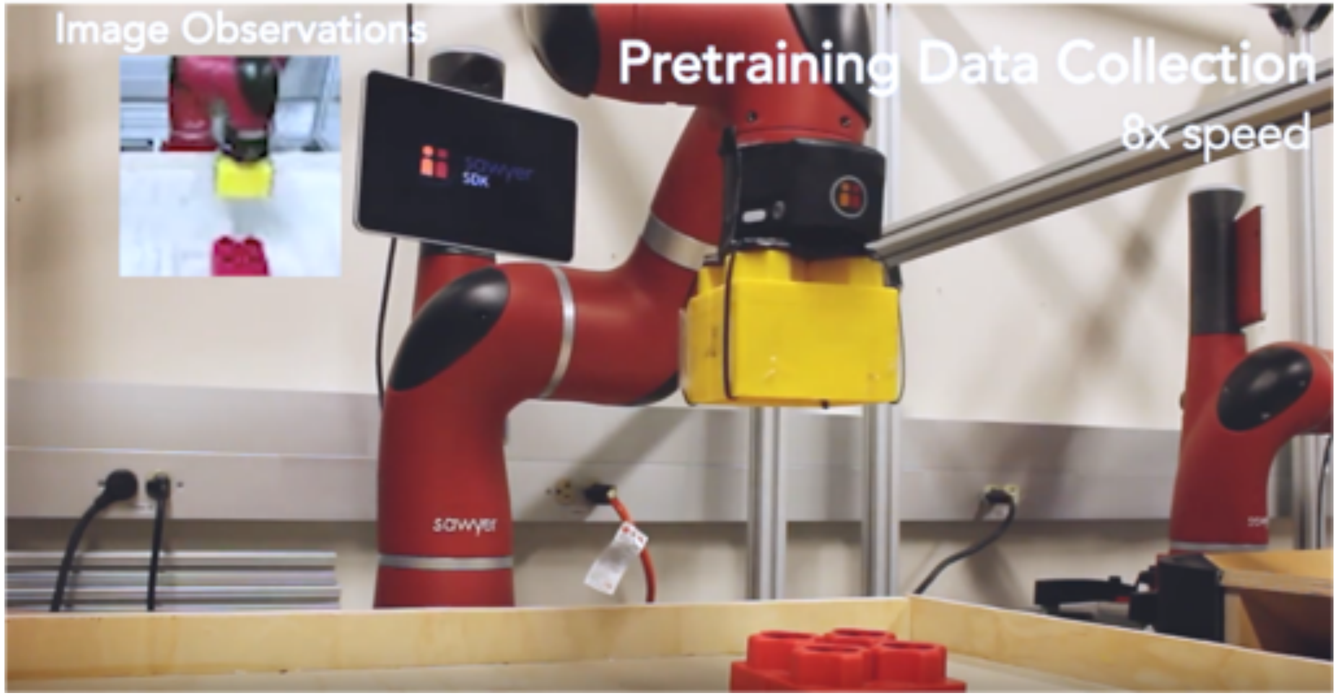
Games form a wonderful sandbox



https://www.youtube.com/watch?v=iymXaDdBdg0&feature=emb_logo

- RL algorithms are typically very sample inefficient
  - Long training time before learning useful behaviour
- Sometime set of actions is too complicated – so sparse reward setting fails completely

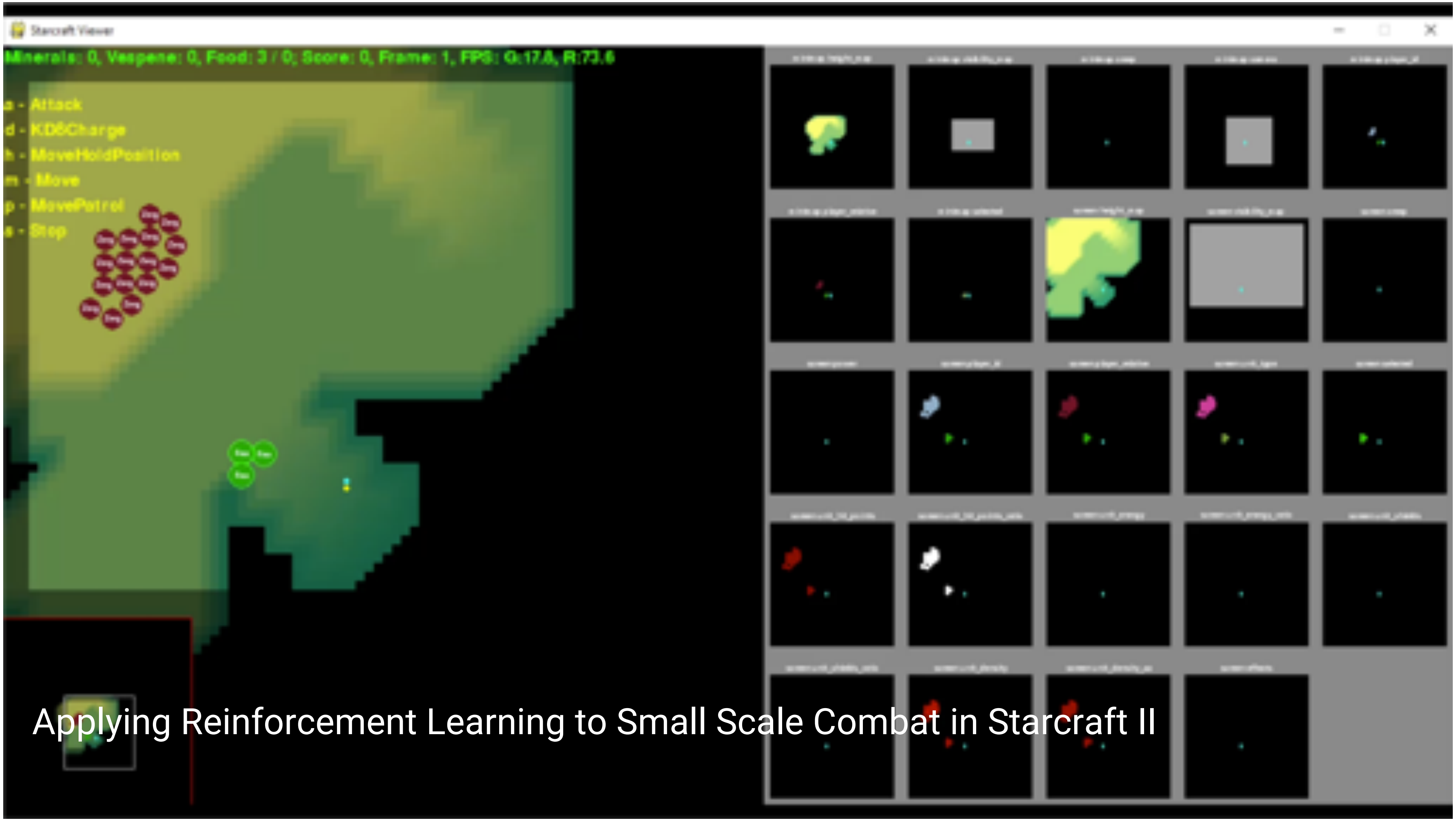
- **Reward shaping**
  - Designing a reward function that guides your policy to a desired behaviour
- Leads to **Alignment Problem**
  - Policy is just overffiting to that reward function you designed while not generalising to the intended bevaviour



https://www.youtube.com/watch?v=iymXaDdBdg0&feature=emb_logo

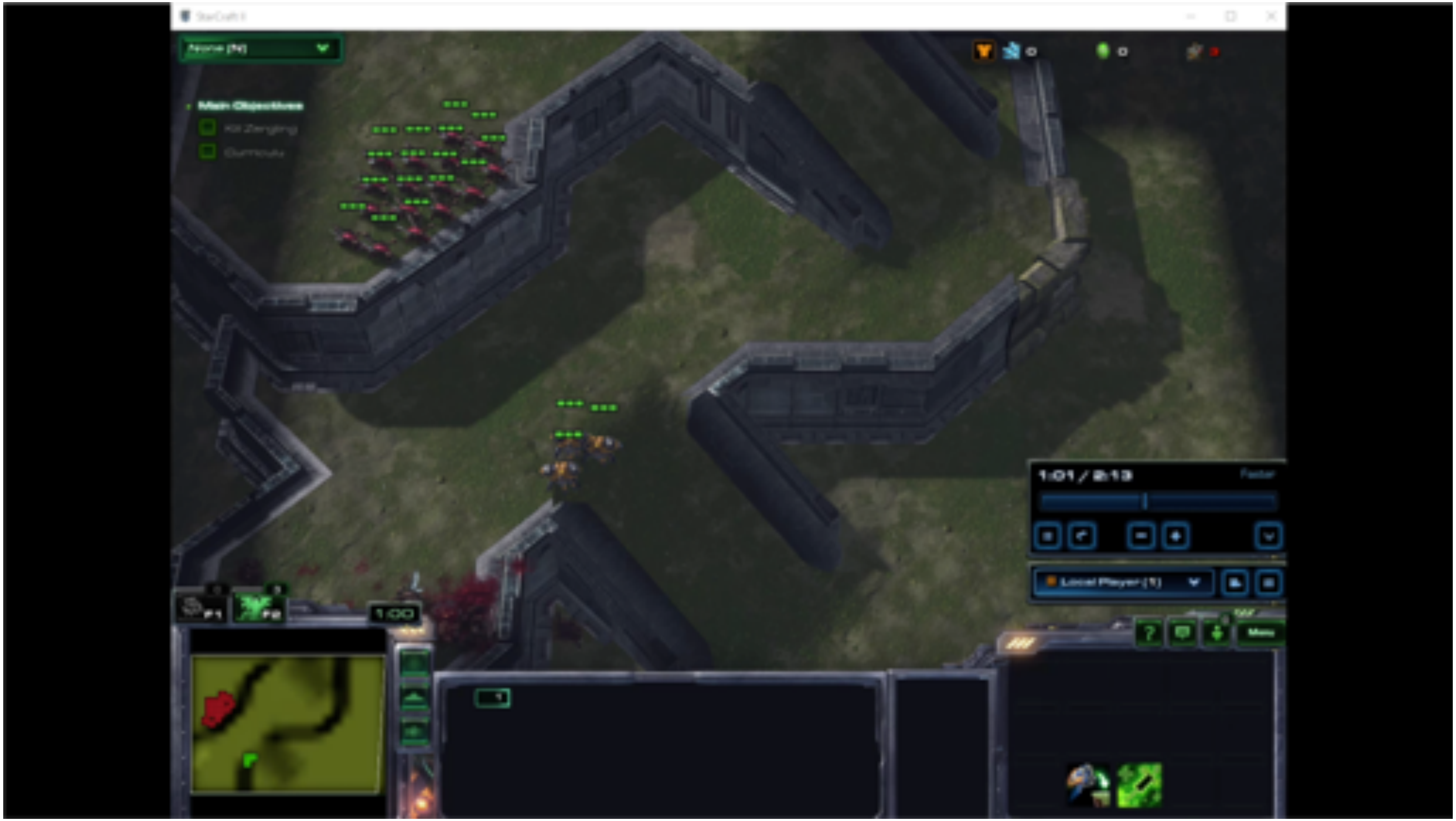
- Know it is hard to train in sparse reward setting
- Tricky to shape a reward function and not always optimal
- Solve problem of **sparse reward setting**
  - **Additional Auxiliary reward signals**
  - Curiosity driven exploration
  - Hindsight experience replay (HER)

Reinforcement Learning



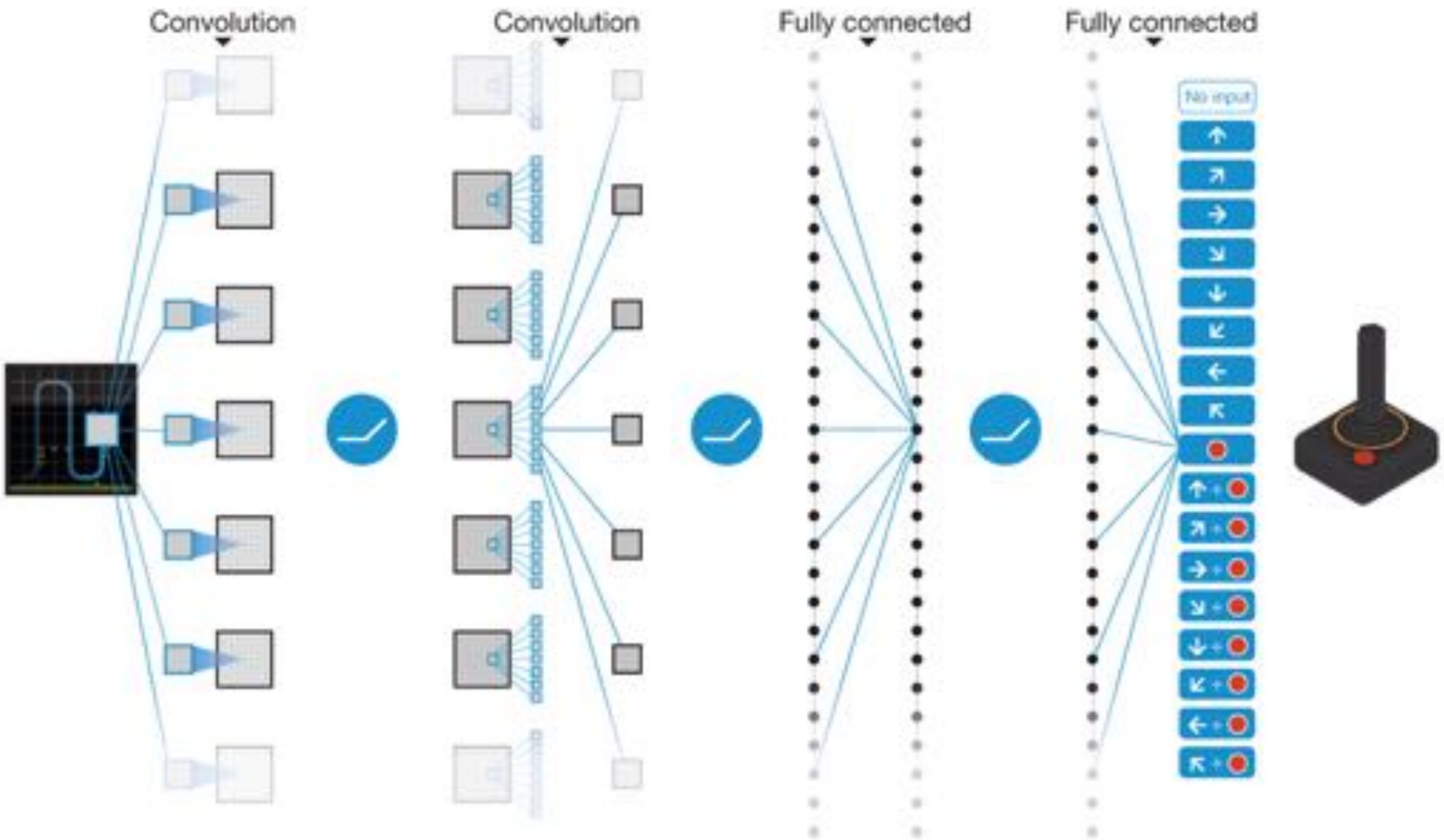

www.BANDICAM.com


Main Objectives
[All] Local Player: reset
2:35
Reaper
Menu



Main Objectives
1:01 / 2:13
1:00
Local Player (1)
Menu

Reinforcement Learning (DQN)



V Mnih *et al. Nature* **518**, 529-533 (2015) doi:10.1038/nature14236

nature

DQNViz: A Visual Analytics Approach to Understand Deep Q-Networks

Junpeng Wang,[1] Liang Gou,[2] Han-Wei Shen,[1] Hao Yang[2]

[1]The Ohio State University

[2]Visa Research