Human-level control through deep reinforcement learning

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg & Demis Hassabis

Presented by Guanheng Luo

Images from David Silver’s lecture slides on Reinforcement Learning

M.U.L.E.
COPYRIGHT (C) 1983
OZARK SOFTSCAPE
STANDBY TO LAND ON
THE PLANET IRATA
ATARI 800
ATARI 810

Overview

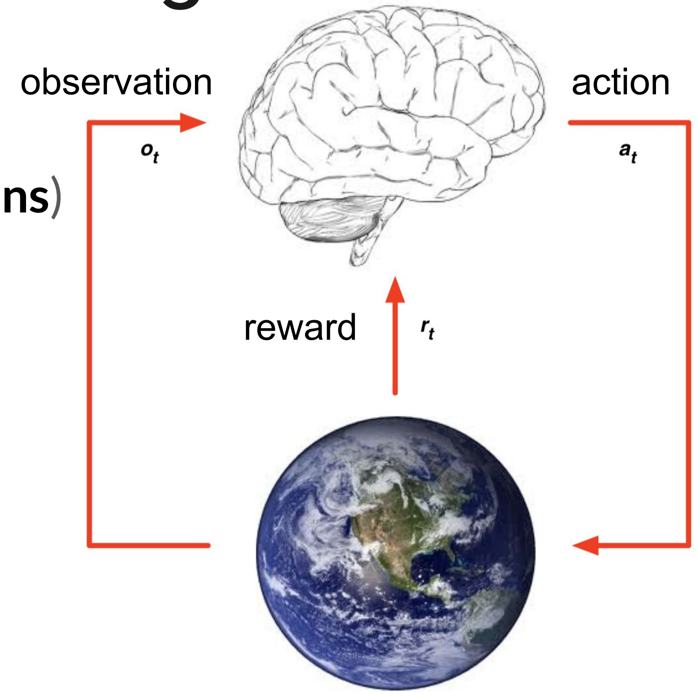
- Combine reinforcement learning with deep neural networks
  - Traditional reinforcement learning is limited to low dimensional input
  - Deep neural network can extract abstract representation from high dimensional input
- Overcome the convergence issue using the following techniques
  - Experience replay
  - Fixed Q target

What is deep reinforcement learning

What is deep reinforcement learning

Goal:

To train an **agent** that interacts (performs **actions**) with the **environment** given the **observation** such that it will receive the maximum accumulated **reward** at the end.



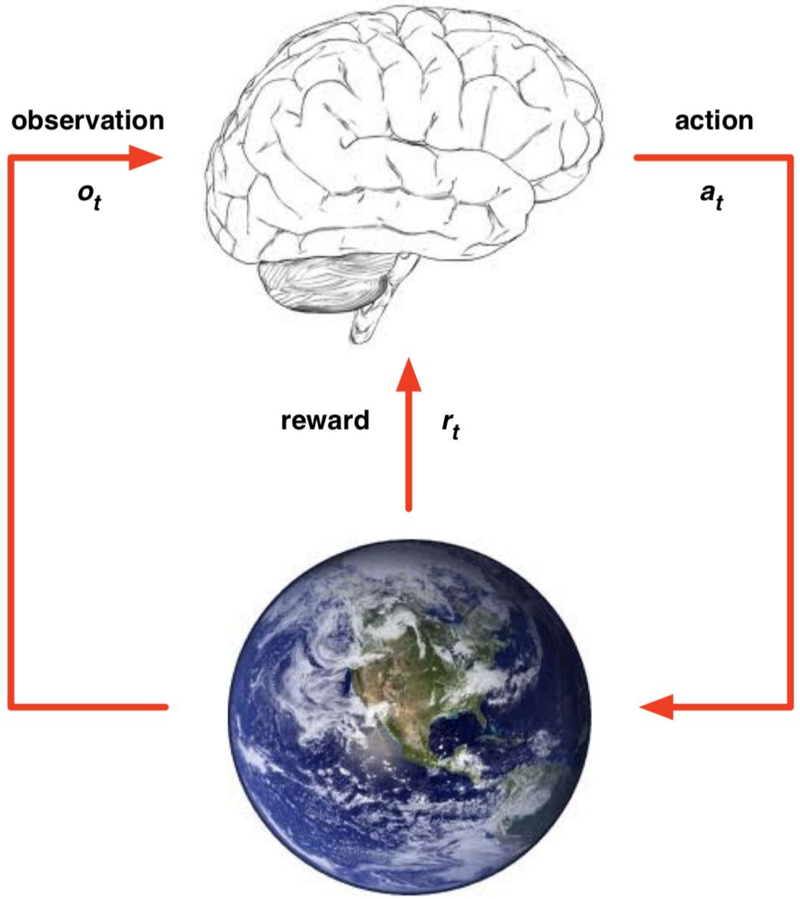
Settings: At each step t,

The agent:

- Receives observation o_t
- Receives scalar reward r_t
- Executes action a_t

The environment:

- Receives action a_t
- Emits observation o_{t+1}
- Emits scalar reward r_{t+1}



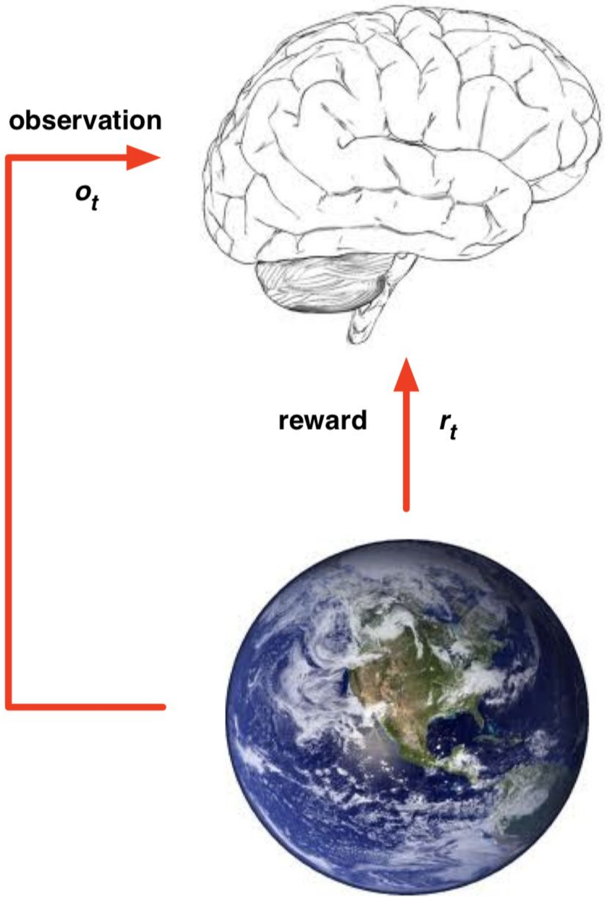
Settings: At each step t,

The agent:

- Receives observation O_t
- Receives scalar reward r_t
- Executes action a_t

The environment:

- Receives action a_t
- Emits observation O_{t+1}
- Emits scalar reward r_{t+1}



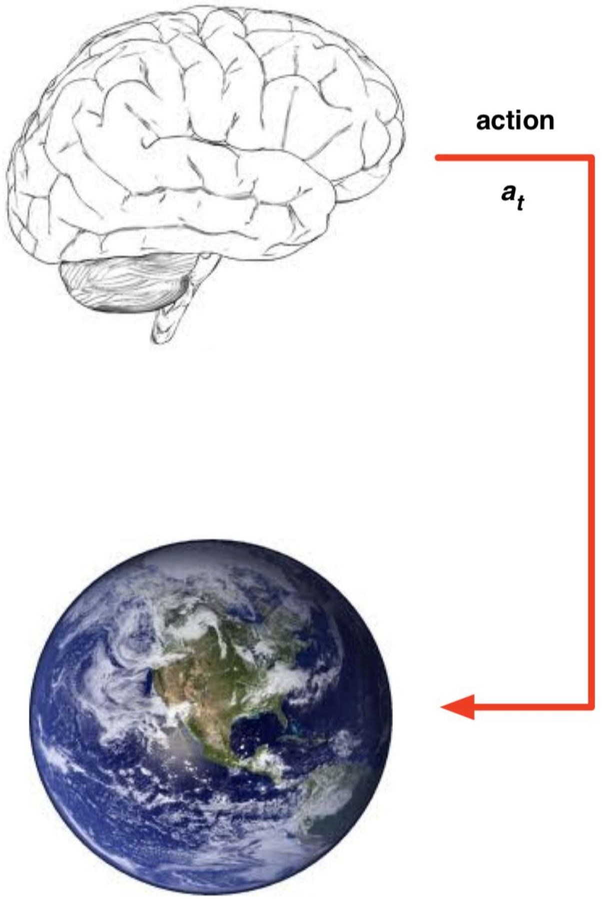
Settings: At each step t,

The agent:

- Receives observation O_t
- Receives scalar reward r_t
- Executes action a_t

The environment:

- Receives action a_t
- Emits observation O_{t+1}
- Emits scalar reward r_{t+1}



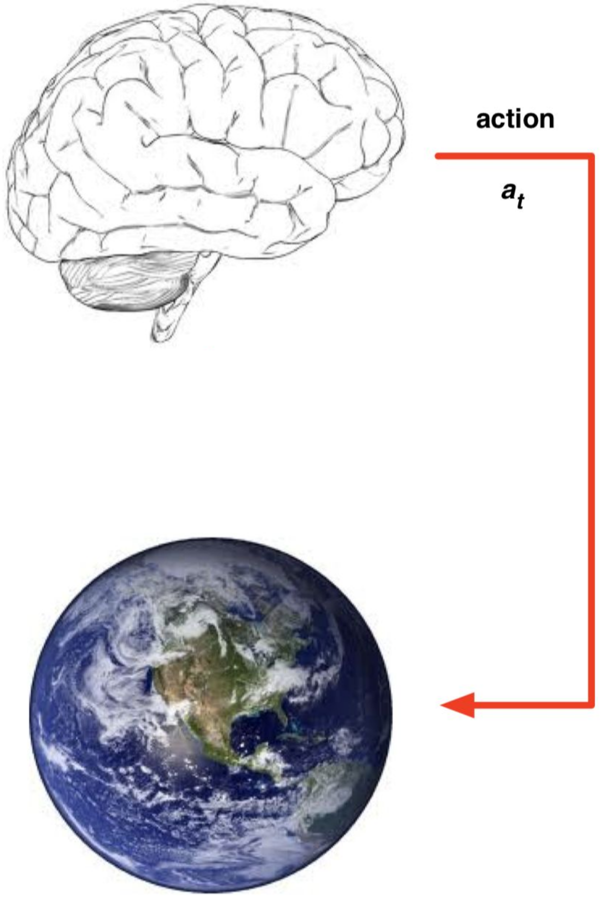
Settings: At each step t,

The agent:

- Receives observation O_t
- Receives scalar reward r_t
- Executes action a_t

The environment:

- Receives action a_t
- Emits observation O_{t+1}
- Emits scalar reward r_{t+1}



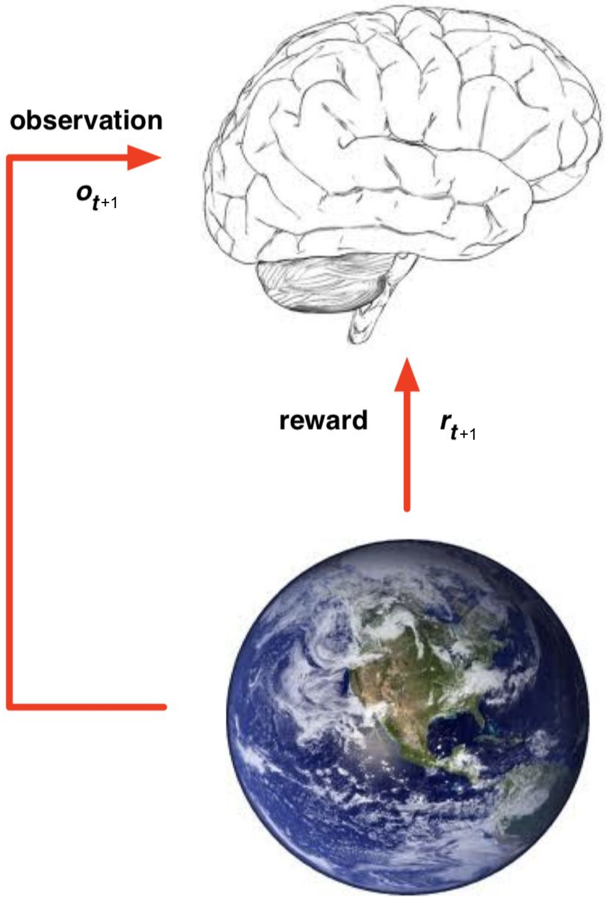
Settings: At each step t,

The agent:

- Receives observation O_t
- Receives scalar reward r_t
- Executes action a_t

The environment:

- Receives action a_t
- Emits observation O_{t+1}
- Emits scalar reward r_{t+1}



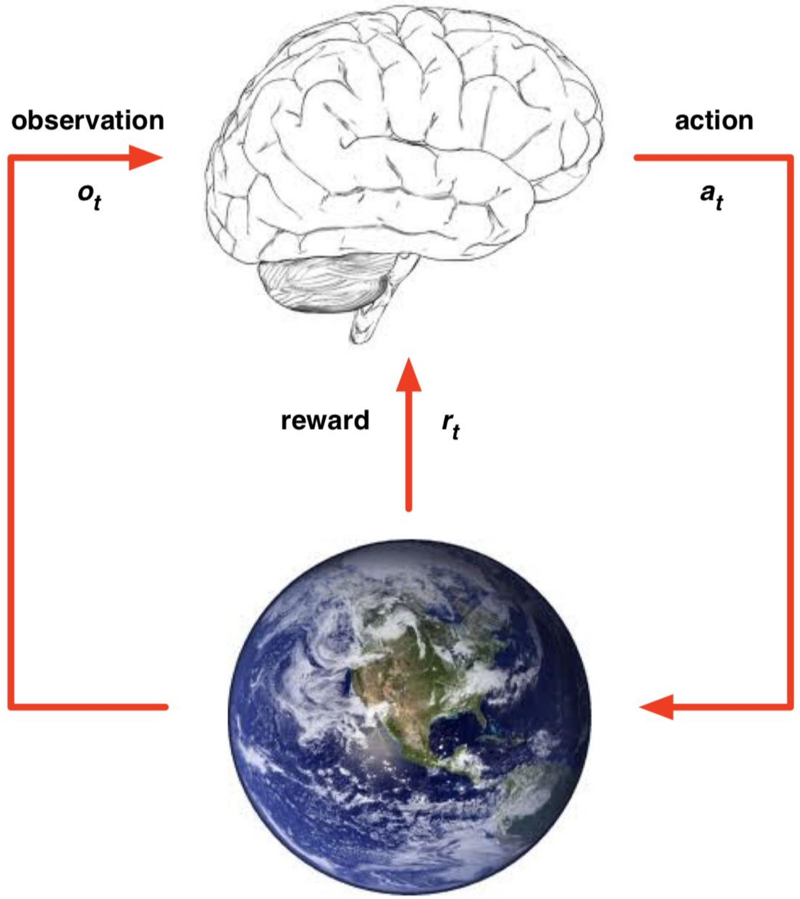
Settings: At each step t,

The agent:

- Receives observation o_t
- Receives scalar reward r_t
- Executes action a_t

The environment:

- Receives action a_t
- Emits observation o_{t+1}
- Emits scalar reward r_{t+1}



What is deep reinforcement learning

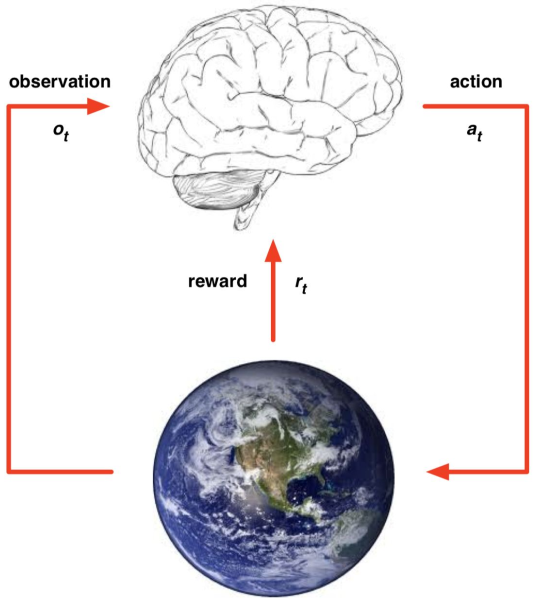
- "Experience" is a sequence of observation, rewards and actions

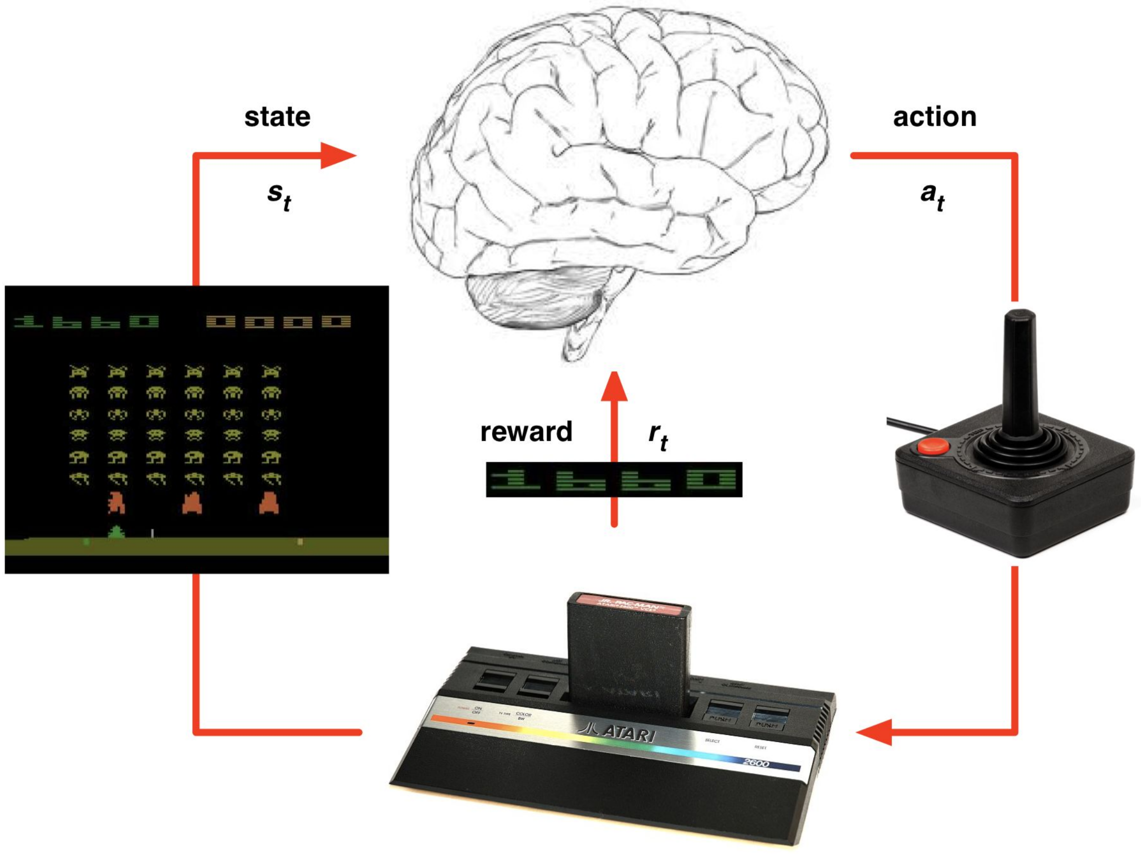
$$o_1, r_1, a_1, \ldots, a_{t-1}, o_t, r_t$$

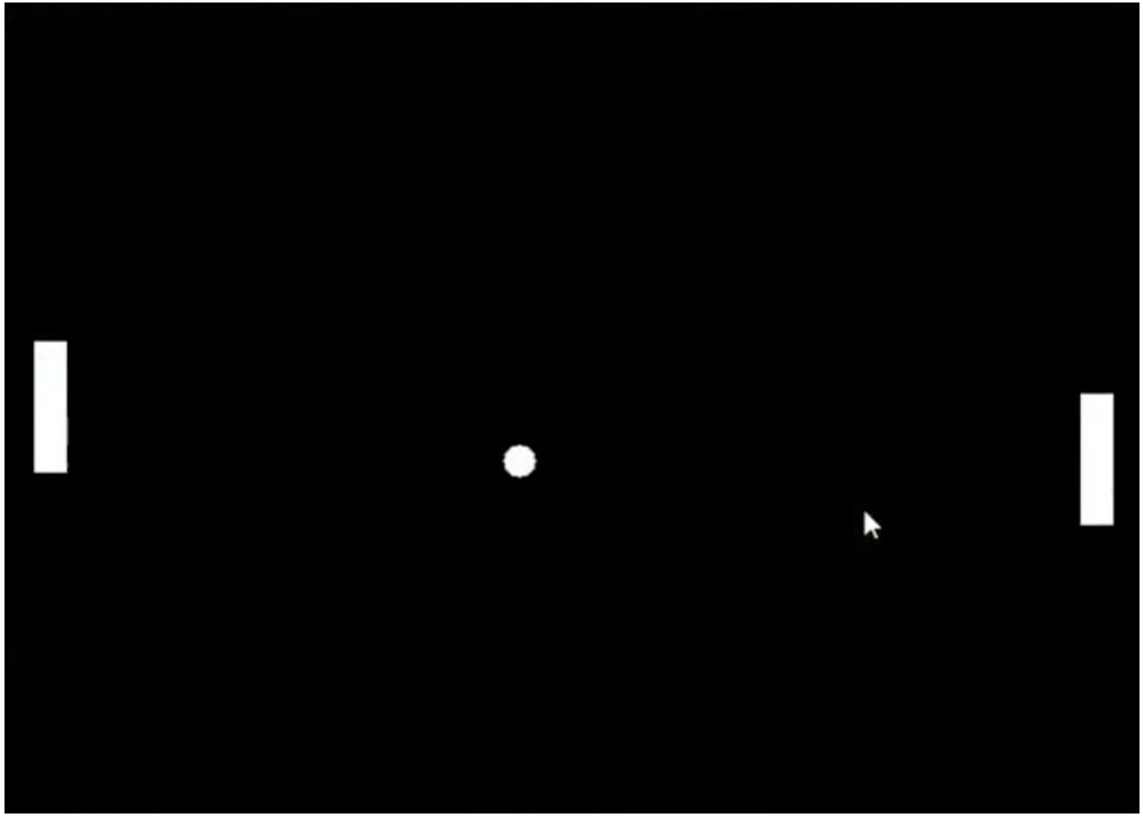
- "State" is a summary of the experience

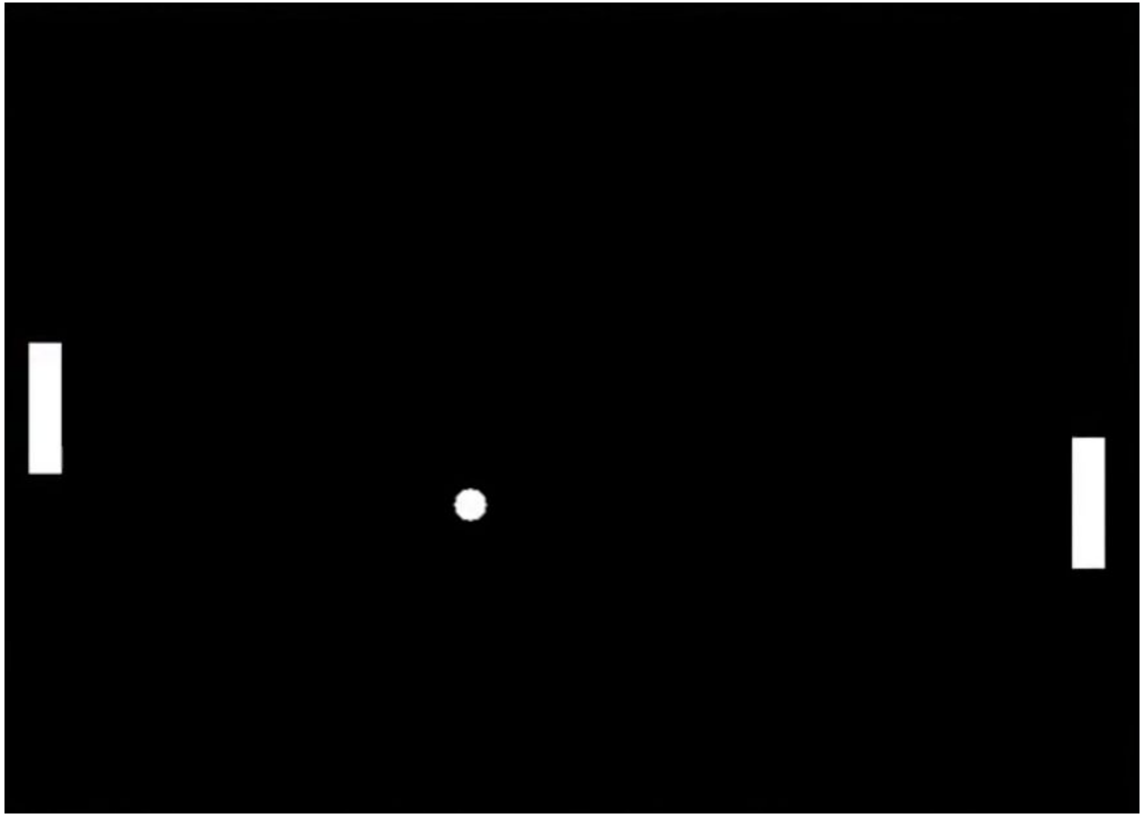
$$s_t = f(o_1, r_1, a_1, \ldots, a_{t-1}, o_t, r_t)$$

(actual input to the agent)




state
s_t
action
a_t
reward
r_t





What is deep reinforcement learning

An agent can have one or more of the following components:

- Policy
  - What should I do?
  - (can be derived from value function)
- Value Function
  - Am I in a good state?
  - (What is the accumulated reward after this state)
- Model
  - The representation of the environment in the agent’s perspective
  - (usually used for planning)

What is deep reinforcement learning

Value Function (Q-value function):

- Expected accumulated reward from state s and action a
- π is the policy
- γ is the discount factor

$$Q^{\pi}(s, a) = \mathbb{E}\left[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots \mid s, a\right]$$

What is deep reinforcement learning

Value Function (Q-value function):

- Expected accumulated reward from state s and action a
- s' is the state arrived after performing a and a' is the action picked at s'

$$Q^{\pi}(s,a) = \mathbb{E}\left[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots \mid s,a\right]$$

$$\downarrow$$

$$Q^{\pi}(s,a) = \mathbb{E}_{s',a'}\left[r + \gamma Q^{\pi}(s',a') \mid s,a\right]$$

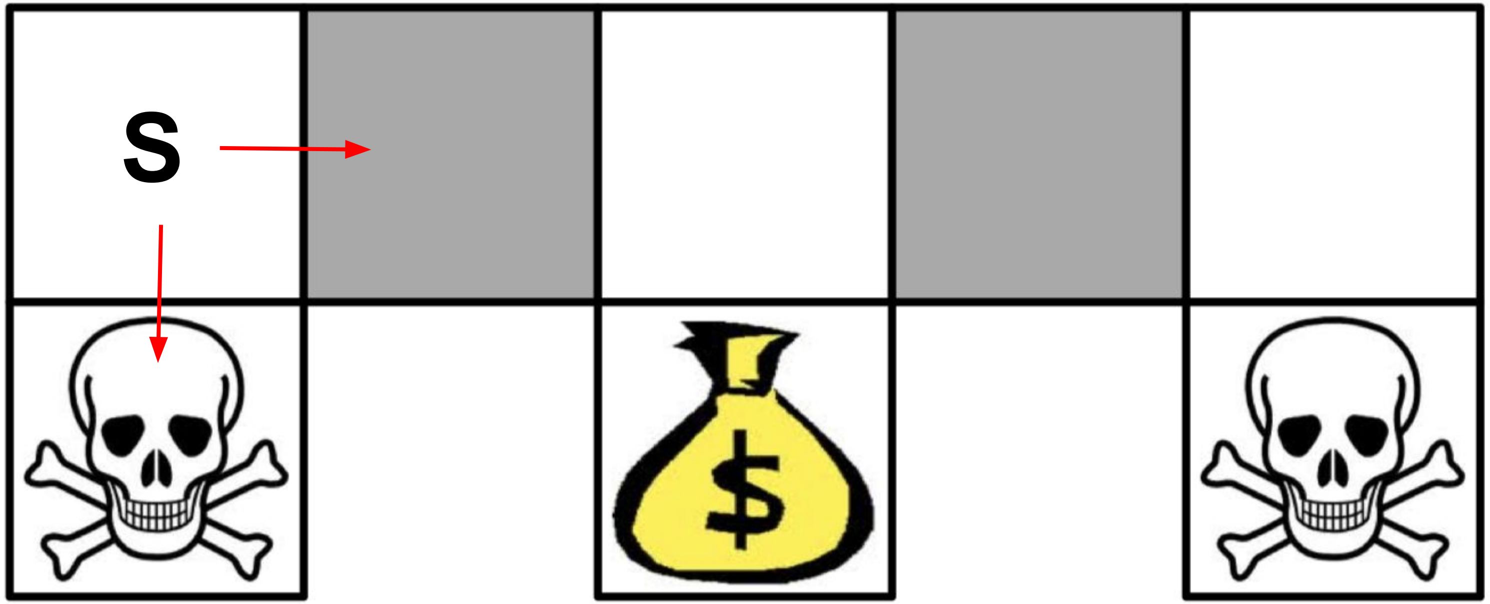
What is deep reinforcement learning

Optimal Value Function (oracle):

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

Optimal Policy:

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



What is Q*(S, down)?

What is Q*(S, right)?

Q*(S, down) = -1000

Q*(S, right) = 1000

$\pi^*(s)$ = right

What is deep reinforcement learning

To obtain Optimal Value Function:

Update our value function iteratively

$$Q(a,s) \leftarrow Q(a,s) + \alpha \cdot \left(\underbrace{\underbrace{r_s + \gamma \max_{a'} Q(a',s')}_{\text{target}} - \underbrace{Q(a,s)}_{\text{prediction}}}_{\text{loss}} \right)$$

What is deep reinforcement learning

To obtain Optimal Value Function:

Update our value function iteratively

$$Q(a,s) \leftarrow Q(a,s) + \alpha \cdot \Big(\underbrace{\underbrace{r_s + \gamma \max_{a'} Q(a',s')}_{\text{target}} - \underbrace{Q(a,s)}_{\text{prediction}}}_{\text{loss}} \Big)$$

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

What is deep reinforcement learning

Issues:

- Can’t derive efficient representations of the environment from high-dimensional sensory inputs
  - Atari 2600: 210 x 160 pixel images with a 128-colour palette
- Can’t generalize past experience to new situations

What is deep reinforcement learning

Solutions:

- Approximate the value function with linear function
  - Require well-handcrafted feature
- Approximate the value function with non-linear function
  - End to end

What is deep reinforcement learning

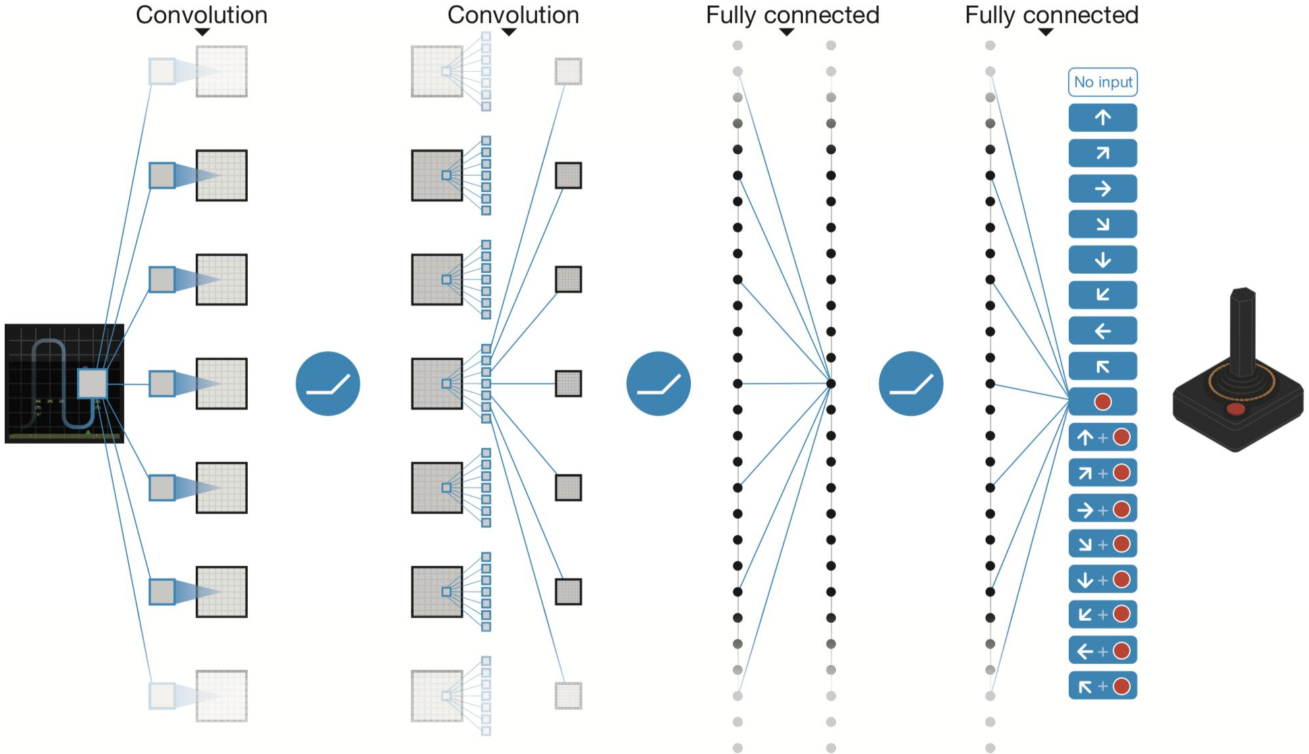
Approximating the value function with deep neural network, i.e. DQN (Deep Q-Network)

$$Q(s,a;\theta) \approx Q^*(s,a)$$

Detail

Network Structure

Input:
    84 * 84 * 4 image
First:
    32 filters, 8 * 8, stride 4
Second:
    64 filters, 4 * 4, stride 2
Third:
    64 filters, 3 * 3, stride 1
Last:
    512 rectifier units
Output:
    Q-value of 18 actions



In deep reinforcement learning

$$Q(a,s) \leftarrow Q(a,s) + \alpha \cdot \underbrace{\left(\underbrace{r_s + \gamma \max_{a'} Q(a',s')}_{\text{target}} - \underbrace{Q(a,s)}_{\text{prediction}} \right)}_{\text{loss}}$$

In deep reinforcement learning

$$Q(a,s) \leftarrow Q(a,s) + \alpha \cdot \underbrace{\left(\underbrace{r_s + \gamma \max_{a'} Q(a',s')}_{\text{target}} - \underbrace{Q(a,s)}_{\text{prediction}} \right)}_{\text{loss}}$$

Let θ_i be the weight of the network at iteration i.

Then:

Target y = $r + \gamma \max_{a'} Q(s',a';\theta_i)$ prediction $Q(s,a;\theta_i)$

Detail

The loss of the network at i-th iteration (mean-squared error)

$$\begin{aligned} L_i(\theta_i) &= \mathbb{E}_{s,a,r}\left[\left(\mathbb{E}_{s'}\left[y|s,a\right] - Q(s,a;\theta_i)\right)^2\right] \\ &= \mathbb{E}_{s,a,r,s'}\left[\left(y - Q(s,a;\theta_i)\right)^2\right] + \mathbb{E}_{s,a,r}\left[\mathbb{V}_{s'}\left[y\right]\right] \end{aligned}$$

Detail

Then it is just performing gradient descent on

$$\nabla_{\theta_i} L(\theta_i) = \mathbb{E}_{s,a,r,s'}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i)\right)\nabla_{\theta_i} Q(s, a; \theta_i)\right]$$

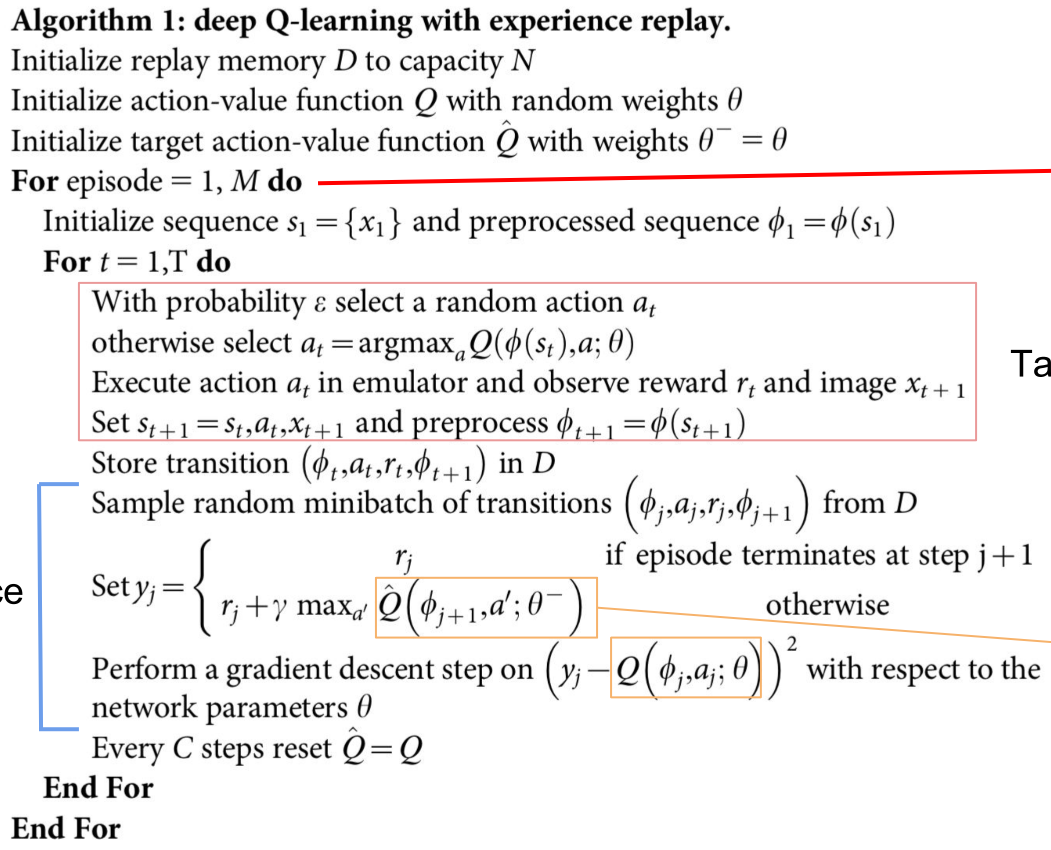
However,

Reinforcement learning using non-linear function may not converge, due to:

- The correlations present in the sequence of observations, the fact that small updates to Q may significantly change the policy and therefore change the data distribution
  - Solution: experience replay

- The correlations between the action-values $Q(s,a;\theta_i)$ and the target values $r+\gamma \max_{a'} Q(s',a';\theta_i\ )$
  - Solution: fixed Q targets

Detail



Algorithm 1: deep Q-learning with experience replay.

```
Initialize replay memory $D$ to capacity $N$
Initialize action-value function $Q$ with random weights $\theta$
Initialize target action-value function $\hat{Q}$ with weights $\theta^- = \theta$
For episode = 1, $M$ do
    Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
    For $t$ = 1,T do
        With probability $\varepsilon$ select a random action $a_t$
        otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$
        Execute action $a_t$ in emulator and observe reward $r_t$ and image $x_{t+1}$
        Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
        Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $D$
        Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $D$
        Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \boxed{\hat{Q}(\phi_{j+1}, a'; \theta^-)} & \text{otherwise} \end{cases}$
        Perform a gradient descent step on $\left(y_j - \boxed{Q(\phi_j, a_j; \theta)}\right)^2$ with respect to the
        network parameters $\theta$
        Every $C$ steps reset $\hat{Q} = Q$
    End For
End For
```

One game

Take a step

Experience replay

Fix Q target

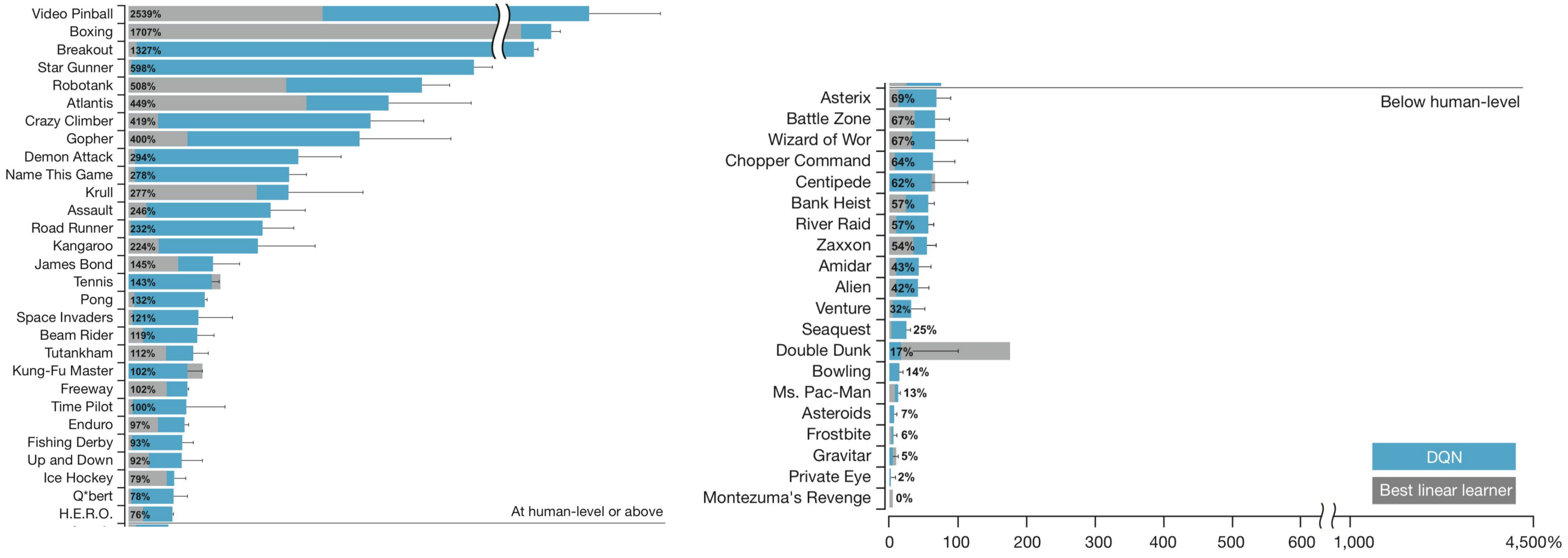
In deep reinforcement learning

$$Q(a,s) \leftarrow Q(a,s) + \alpha \cdot \underbrace{\left(\underbrace{r_s + \gamma \max_{a'} Q(a',s')}_{\text{target}} - \underbrace{Q(a,s)}_{\text{prediction}} \right)}_{\text{loss}}$$

Detail

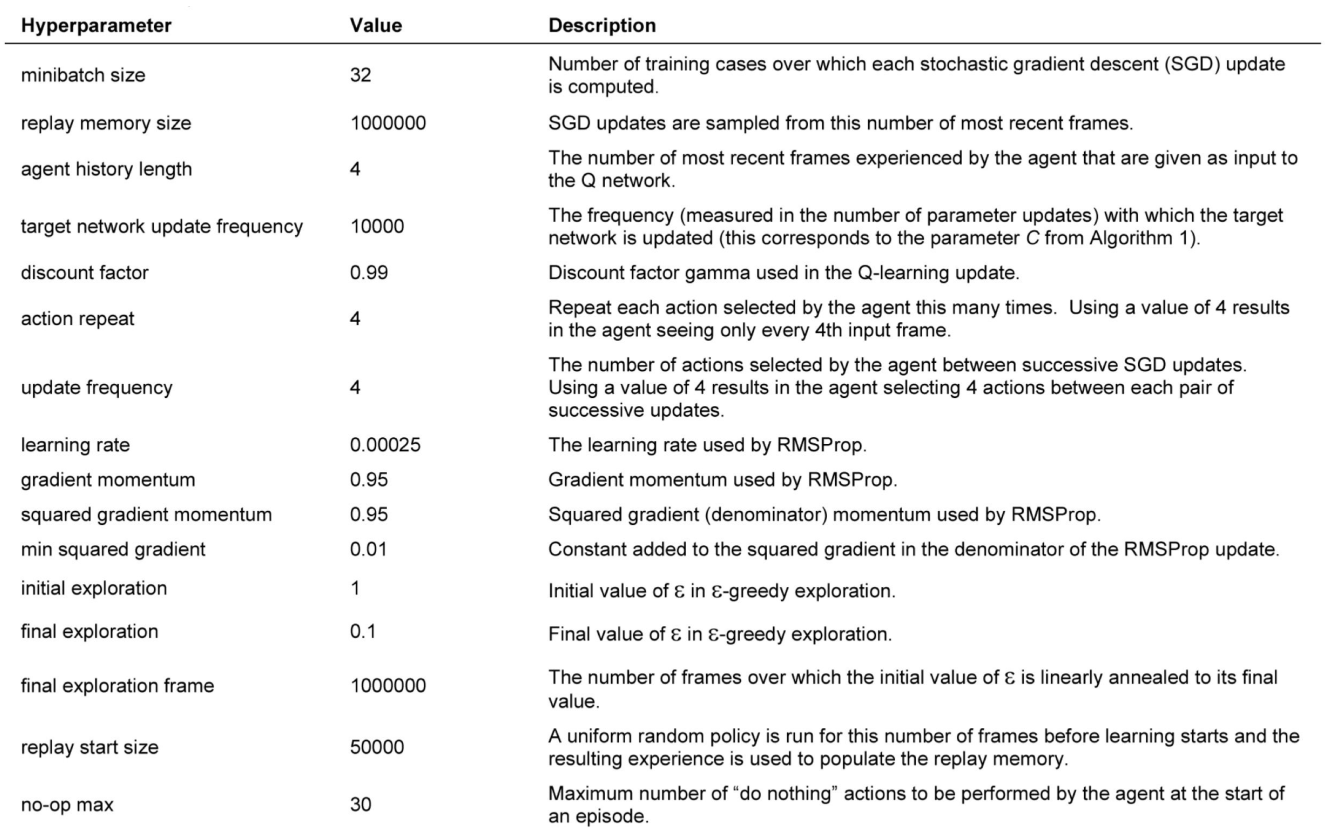
Game	With replay, with target Q	With replay, without target Q	Without replay, with target Q	Without replay, without target Q
Breakout	316.8	240.7	10.2	3.2
Enduro	1006.3	831.4	141.9	29.1
River Raid	7446.6	4102.8	2867.7	1453.0
Seaquest	2894.4	822.6	1003.0	275.8
Space Invaders	1088.9	826.3	373.2	302.0

Result



100 * (DQN score - random play score)/ (human score - random play score)

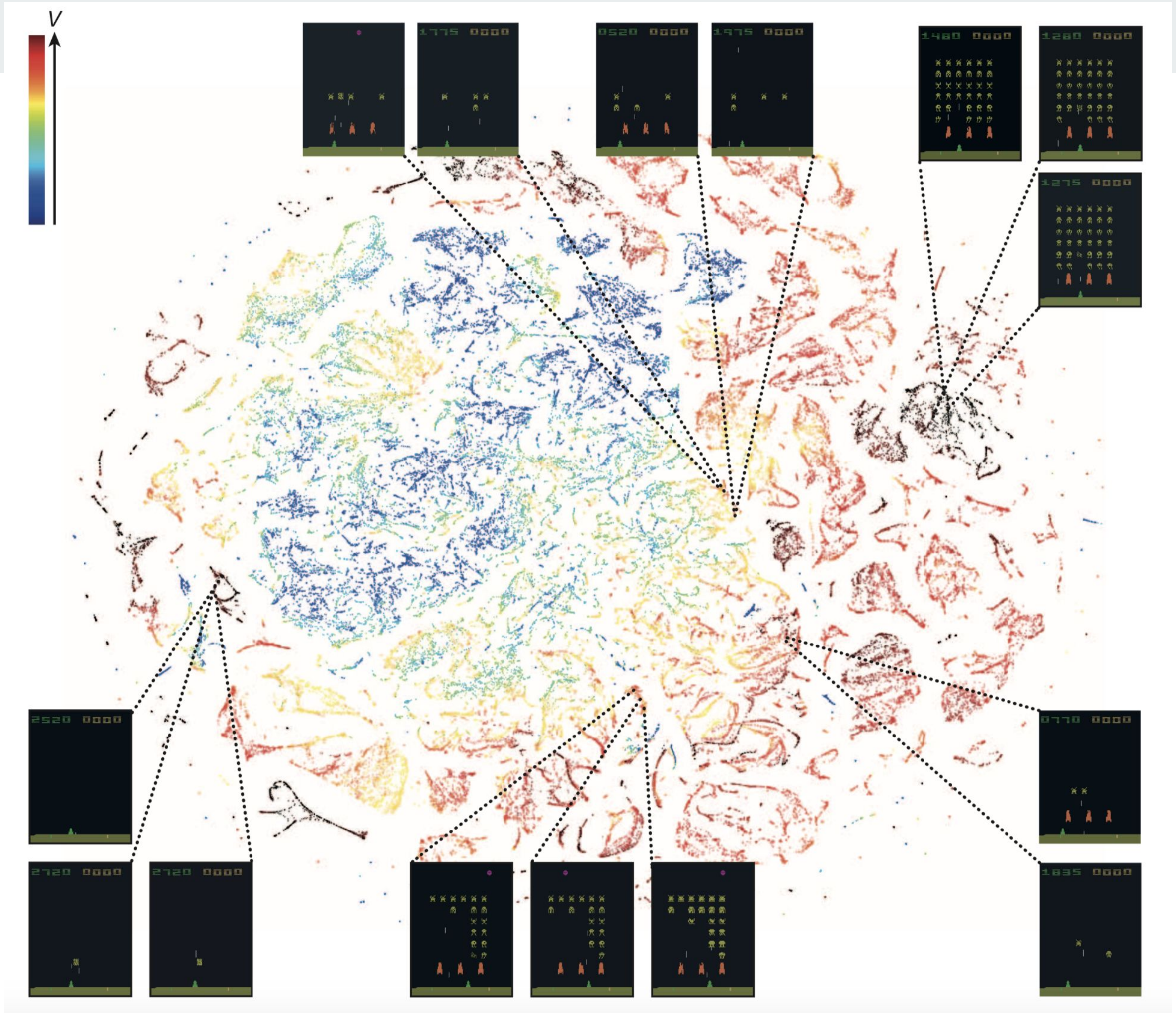
Parameters



Hyperparameter	Value	Description
minibatch size	32	Number of training cases over which each stochastic gradient descent (SGD) update is computed.
replay memory size	1000000	SGD updates are sampled from this number of most recent frames.
agent history length	4	The number of most recent frames experienced by the agent that are given as input to the Q network.
target network update frequency	10000	The frequency (measured in the number of parameter updates) with which the target network is updated (this corresponds to the parameter *C* from Algorithm 1).
discount factor	0.99	Discount factor gamma used in the Q-learning update.
action repeat	4	Repeat each action selected by the agent this many times. Using a value of 4 results in the agent seeing only every 4th input frame.
update frequency	4	The number of actions selected by the agent between successive SGD updates. Using a value of 4 results in the agent selecting 4 actions between each pair of successive updates.
learning rate	0.00025	The learning rate used by RMSProp.
gradient momentum	0.95	Gradient momentum used by RMSProp.
squared gradient momentum	0.95	Squared gradient (denominator) momentum used by RMSProp.
min squared gradient	0.01	Constant added to the squared gradient in the denominator of the RMSProp update.
initial exploration	1	Initial value of ε in ε-greedy exploration.
final exploration	0.1	Final value of ε in ε-greedy exploration.
final exploration frame	1000000	The number of frames over which the initial value of ε is linearly annealed to its final value.
replay start size	50000	A uniform random policy is run for this number of frames before learning starts and the resulting experience is used to populate the replay memory.
no-op max	30	Maximum number of “do nothing” actions to be performed by the agent at the start of an episode.

https://media.nature.com/original/nature-assets/nature/journal/v518/n7540/extref/nature14236-sv2.mov

Thanks for watching



t-SNE embedding