Human-level control through deep reinforcement learning

Jiang Guo

2016.04.19

Towards General Artificial Intelligence

- **Playing Atari with Deep Reinforcement Learning**. *ArXiv (2013)*
  - 7 Atari games
  - The first step towards “General Artificial Intelligence”

- DeepMind got acquired by @Google (2014)

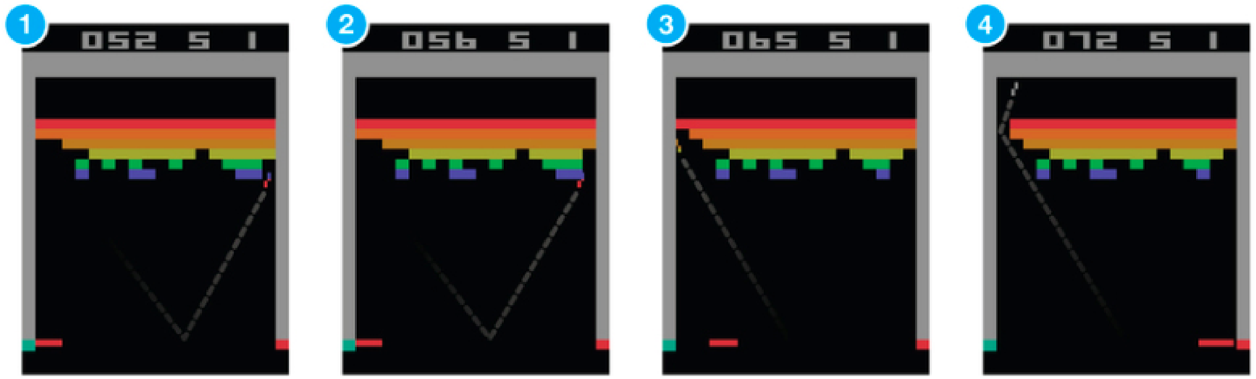
- **Human-level control through deep reinforcement learning**. *Nature (2015)*
  - 49 Atari games
  - Google patented “Deep Reinforcement Learning”

Key Concepts

- Reinforcement Learning
- Markov Decision Process
- Discounted Future Reward
- Q-Learning
- Deep Q Network
- Exploration-Exploitation
- Experience Replay
- Deep Q-learning Algorithm

Reinforcement Learning

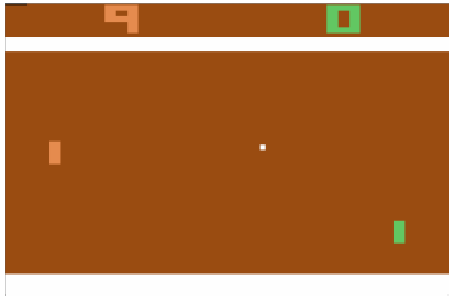
- Example: breakout (one of the Atari games)



- Suppose you want to teach an agent (e.g. NN) to play this game
  - Supervised training (expert players play a million times) *That's not how we learn!*
  - **Reinforcement learning**

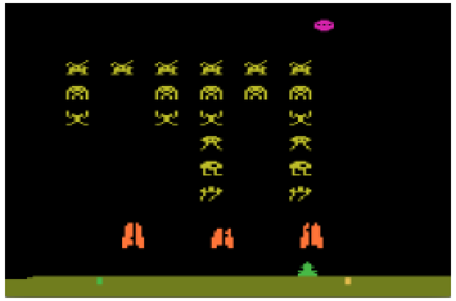
Reinforcement Learning

ML		
	Supervised Learning	Target label for each training example
	Reinforcement Learning	Sparse and time-delayed labels
	Unsupervised Learning	No label at all



Pong

Breakout

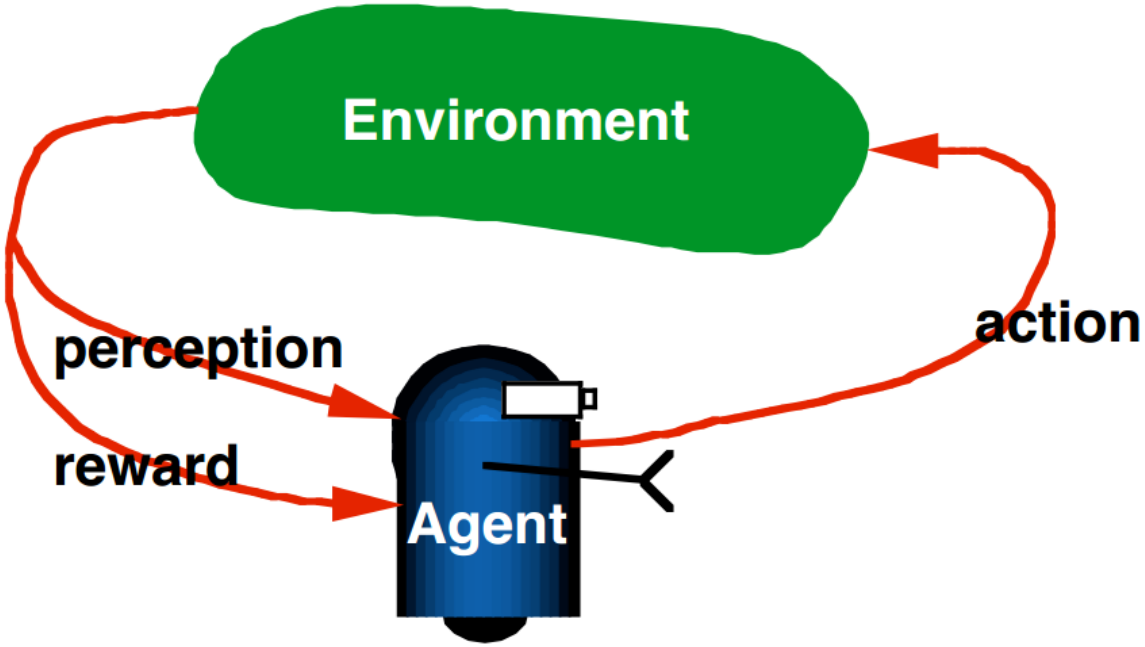


Space Invaders

Seaquest

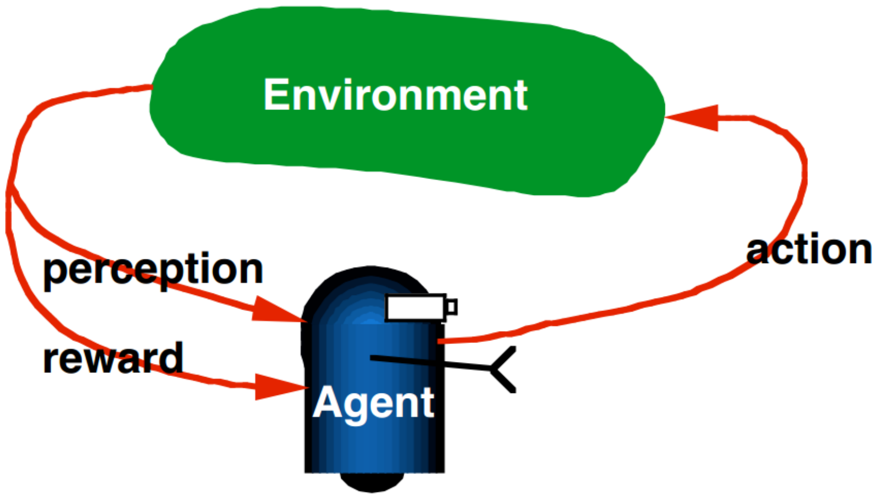
Beam Rider

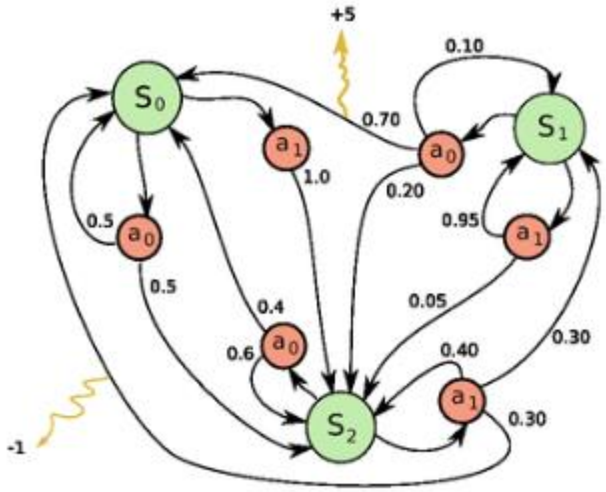
RL is Learning from Interaction



RL is like Life!

Markov Decision Process





$$s_0, a_0, r_1, s_1, a_1, r_2, \ldots, s_{n-1}, a_{n-1}, r_n, s_n$$

state

action

reward

Terminal state

State Representation

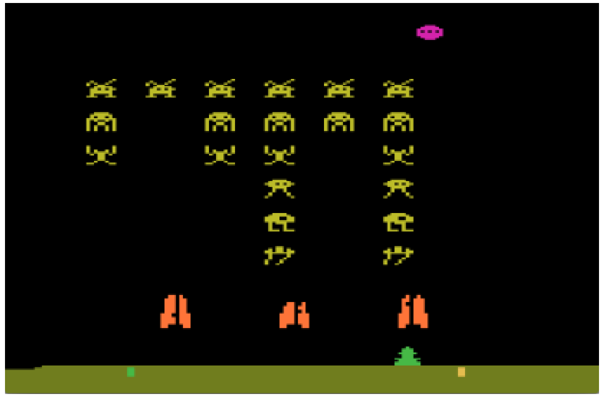
Think about the **Breakout** game

- How to define a state?

- Location of the paddle
- Location/direction of the ball
- Presence/absence of each individual brick

Let's make it more universal!

Screen pixels



Value Function

MDP

$$s_0, a_0, r_1, s_1, a_1, r_2, \ldots, s_{n-1}, a_{n-1}, r_n, s_n$$

- Future reward

$$R = r_1 + r_2 + r_3 + \cdots + r_n$$

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

- Discounted future reward (environment is stochastic)

$$\begin{aligned} R_t &= r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^{n-t} r_n \\ &= r_t + \gamma(r_{t+1} + \gamma(r_{t+2} + \cdots)) \\ &= r_t + \gamma R_{t+1} \end{aligned}$$

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

Value-Action Function

- We define a $Q(s, a)$ representing the maximum discounted future reward when we perform action <u>a</u> in state <u>s</u>:

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

  - **Q-function**: represents the "**Quality**" of a certain action in a given state
- Imagine you have the magical Q-function

$$\pi(s) = arg\max_{a} Q(s, a)$$

  - π is the policy

Q-Learning

- How do we get the Q-function?
  - Bellman Equation （贝尔曼公式）

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

```
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
```

Value Iteration

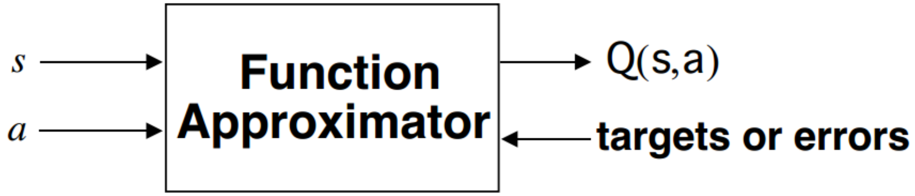
Q-Learning

- In practice, Value Iteration is impractical
  - Very limited states/actions
  - Cannot generalize to unobserved states

- Think about the **Breakout** game
  - State: screen pixels
    - Image size: $\mathbf{84 \times 84}$ (resized)
    - Consecutive **4** images
    - Grayscale with **256** gray levels

$\mathbf{256^{84\times84\times4}}$ rows in the Q-table!

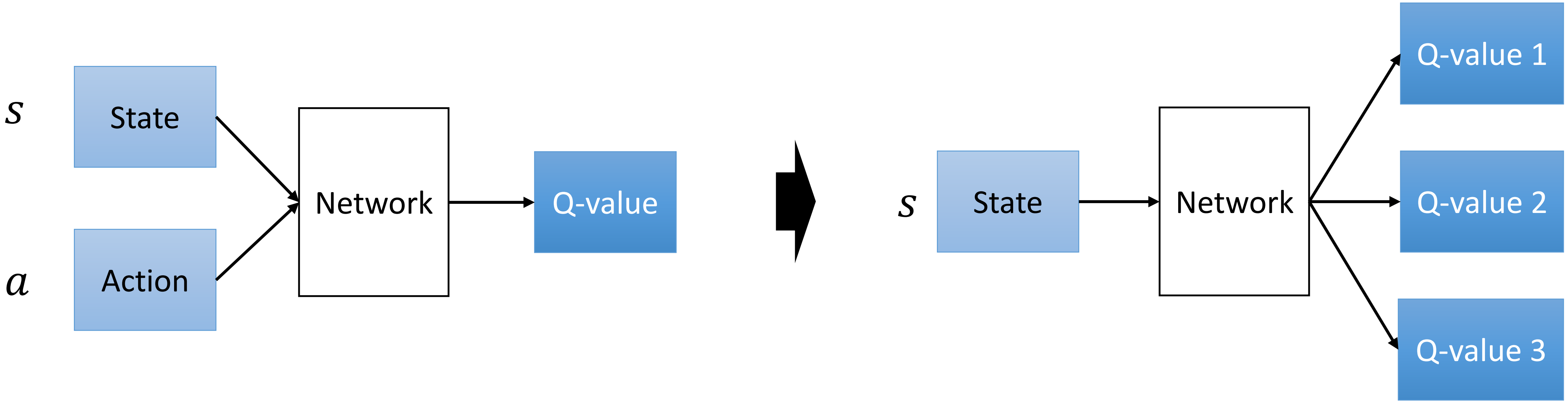
Function Approximator



- Use a function (with parameters) to approximate the Q-function

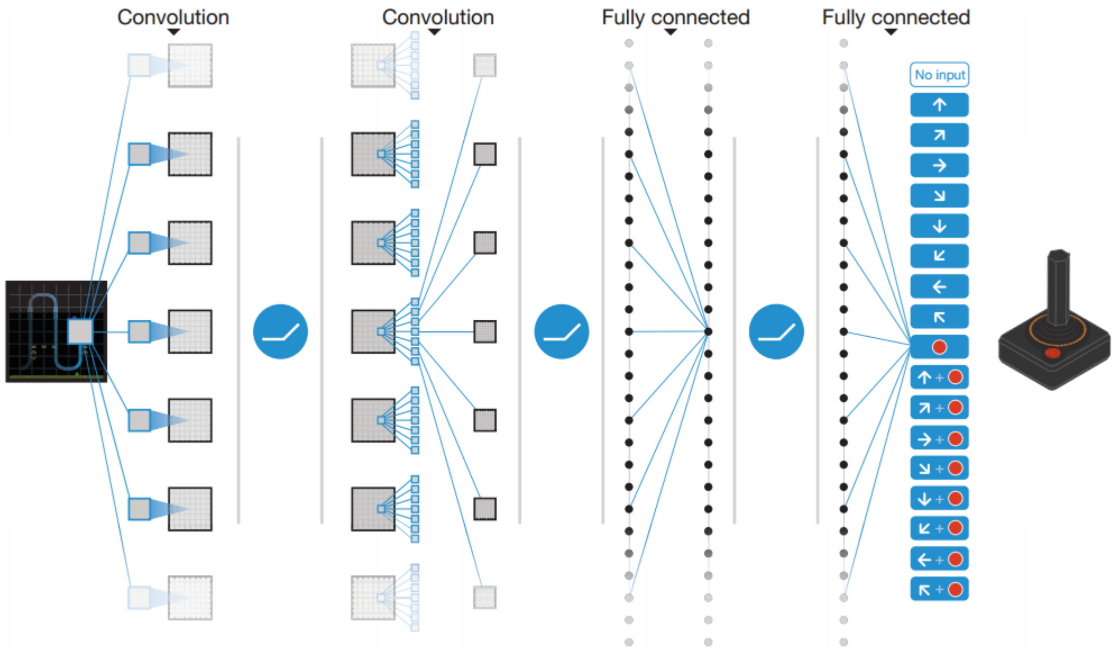
$$Q(s,a;\boldsymbol{\theta}) \approx Q^*(s,a)$$

  - Linear
  - Non-linear: **Q-network**



Deep Q-Network

Deep Q-Network used in the DeepMind paper:



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

Note: No Pooling Layer!

Estimating the Q-Network

- Objective Function
  - Recall the Bellman Equation: $Q(s,a) = r + \gamma max_{a'} Q(s',a')$
  - Here, we use simple squared error:

$$L = \mathbb{E}[(\underbrace{\boldsymbol{r + \gamma max_{a'} Q(s',a')}}_{\textbf{target}} - Q(s,a))^2]$$

  - Leading to the following Q-learning gradient

$$\frac{\partial L(w)}{\partial w} = \mathbb{E}[(\boldsymbol{r + \gamma max_{a'} Q(s',a')} - Q(s,a)) \frac{\partial Q(s,a,w)}{\partial w}]$$

  - Optimize objective end-to-end by **SGD**

Learning Stability

- Non-linear function approximator (Q-Network) is not very stable

Deep Learning		**Reinforcement Learning**
Data samples are I.I.D.	*vs.*	States are highly correlated
Underlying data distribution is fixed		Data distribution changes

1. Exploration-Exploitation

2. Experience Replay

Exploration-Exploitation Dilemma (探索-利用 困境)

- During training, how do we choose an action at time t?
  - （探索）Exploration: random guessing
  - （利用）Exploitation: choose the best one according to the Q-value
- ϵ-greedy policy
  - With probability ϵ select a random action (Exploration)
  - Otherwise select $a = argmax_{a'} Q(s, a')$ (Exploitation)

Experience Replay

- To remove correlations, build data-set from agent's own experience

1. Take action a_t according to **ϵ-greedy policy**
2. During gameplay, store transition $< s_t, a_t, r_{t+1}, s_{t+1} >$ in replay memory D
3. Sample random mini-batch of transitions $< s, a, r, s' >$ from D
4. Optimize MSE between Q-network and Q-learning targets

$$L = \mathbb{E}_{s,a,r,s' \sim D} \frac{1}{2} [\boldsymbol{r + \gamma max_{a'} Q(s', a')} - Q(s, a)]^2$$

Algorithm 1: deep Q-learning with experience replay.

Initialize replay memory D to capacity N

Initialize action-value function Q with random weights θ

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**

ϵ-greedy policy With probability ε 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 $\phi_{t+1} = \phi(s_{t+1})$

Experience memory 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

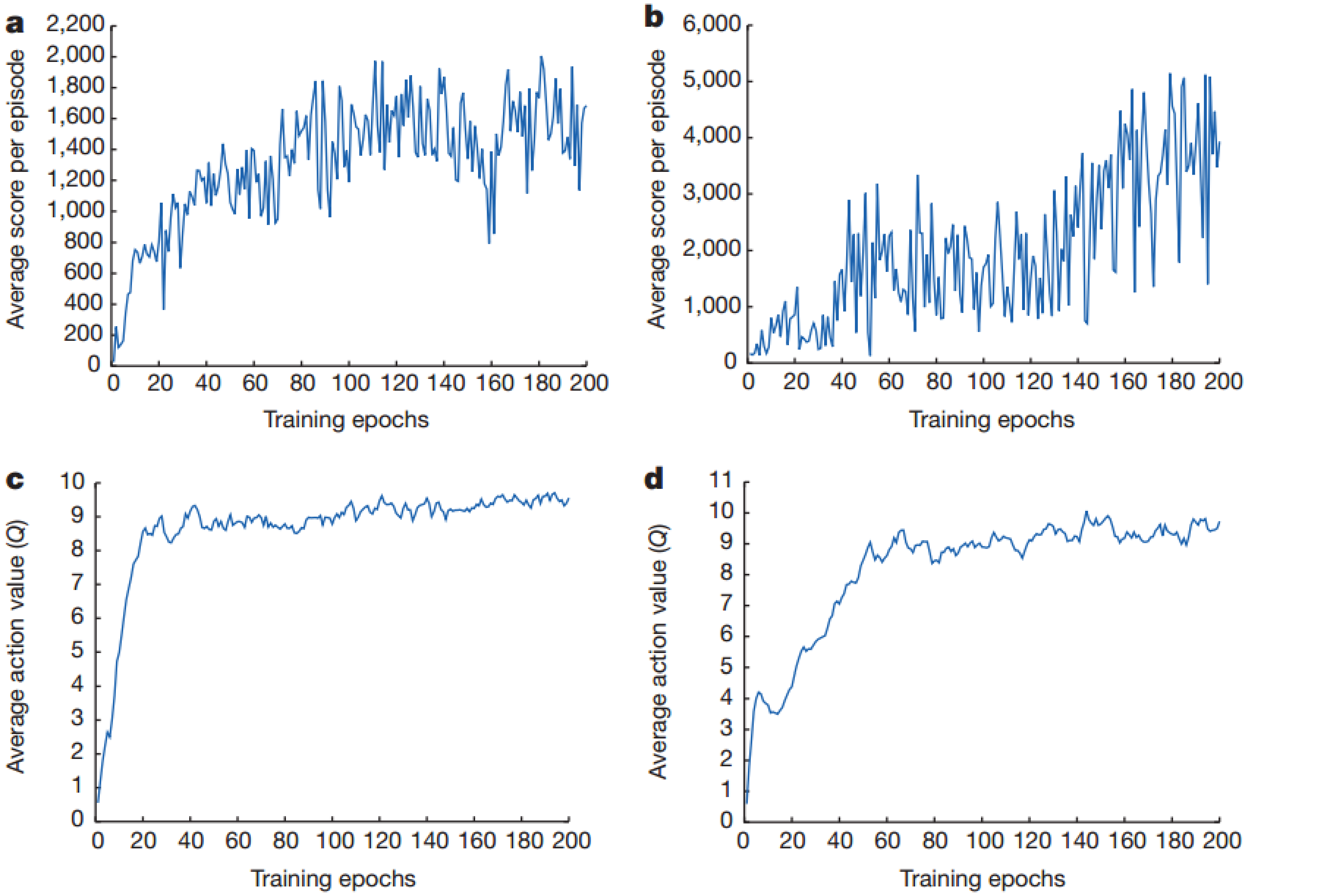
$$\text{Set } y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \boxed{\hat{Q}}\left(\phi_{j+1}, a'; \theta^{-}\right) & \text{otherwise} \end{cases}$$

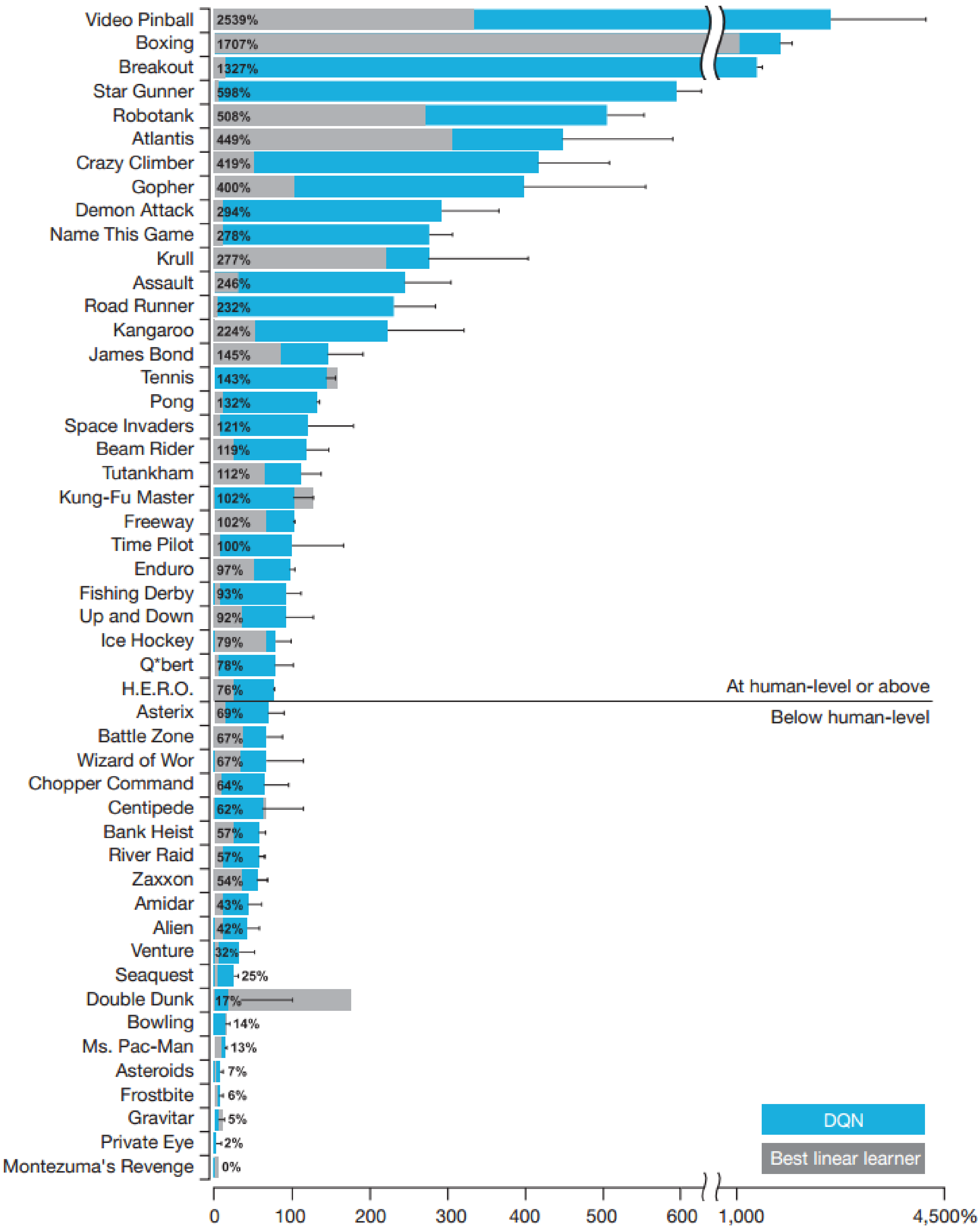
Perform a gradient descent step on $\left(y_j - \boxed{Q}\left(\phi_j, a_j; \theta\right)\right)^2$ with respect to the network parameters θ

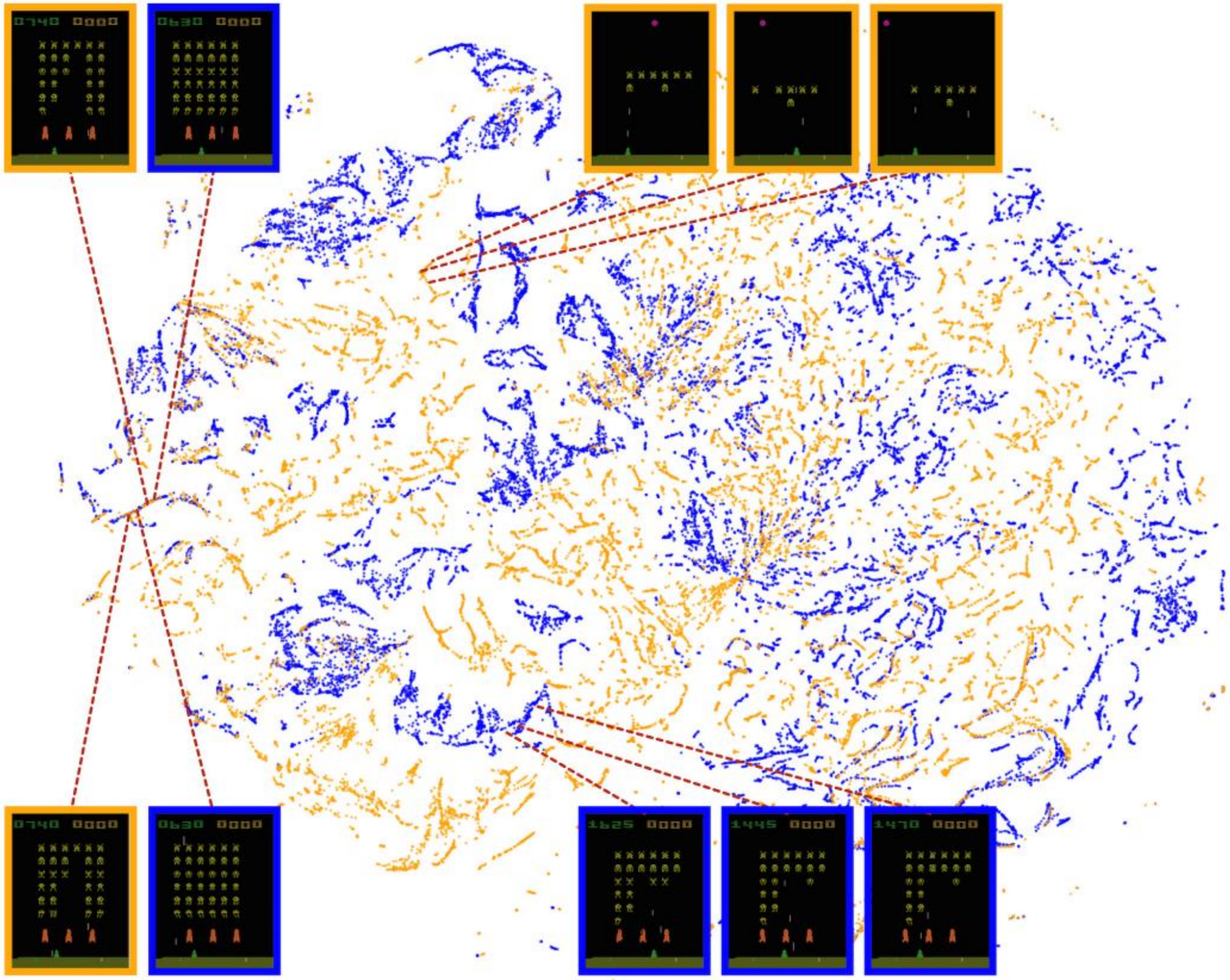
Target network Every C steps reset $\hat{Q} = Q$

End For

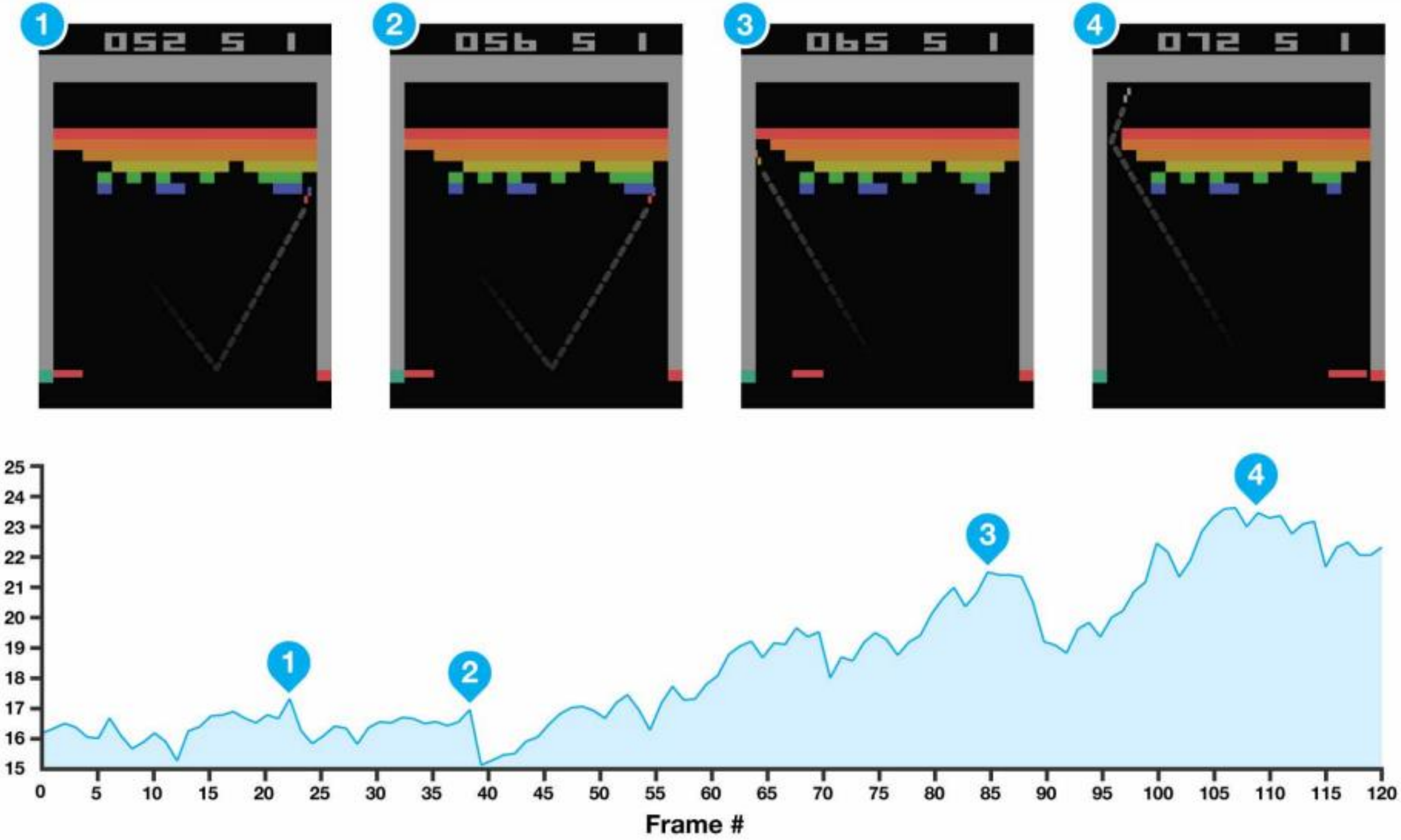
End For

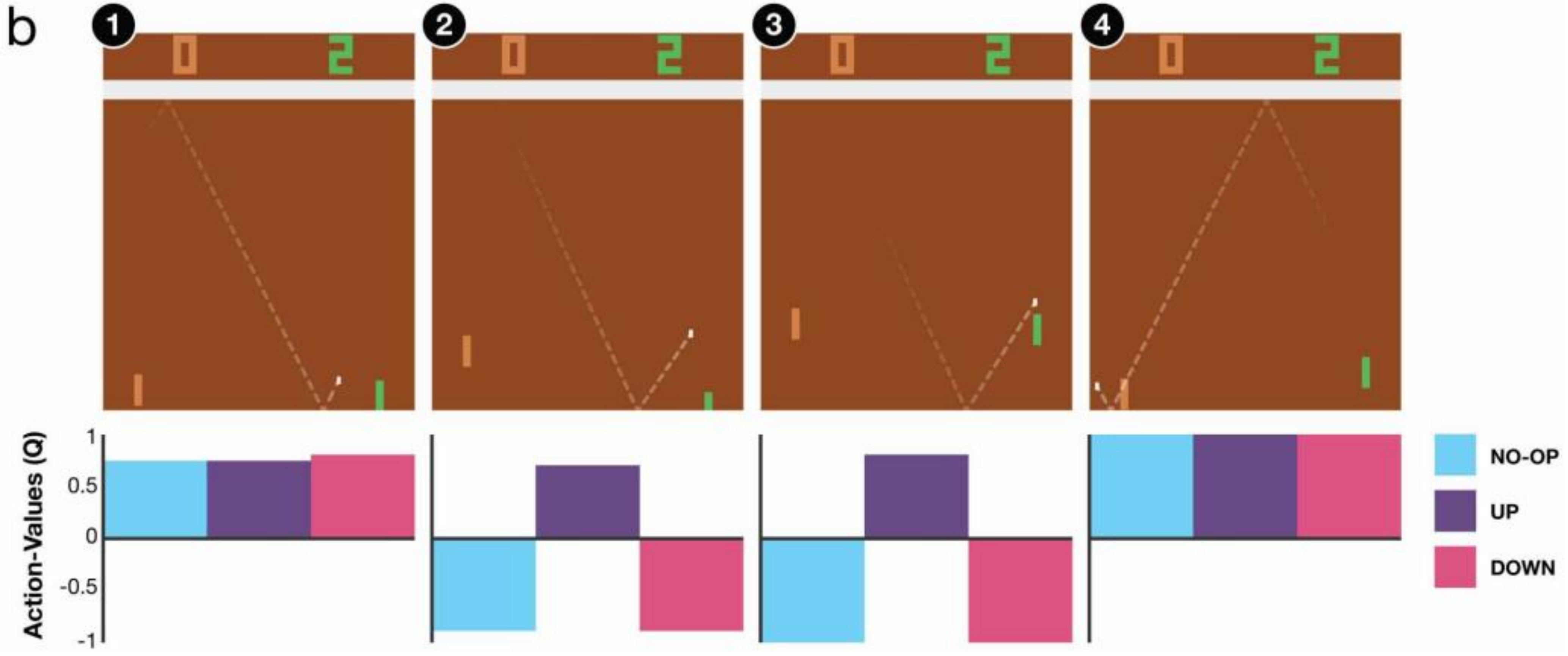

a
b
c
d
Average score per episode
Average action value (Q)
Training epochs
0
20
40
60
80
100
120
140
160
180
200
2,200
2,000
1,800
1,600
1,400
1,200
1,000
800
600
400
200
6,000
5,000
4,000
3,000
2,000
1,000
10
9
8
7
6
5
4
3
2
1
11


Video Pinball
2539%
Boxing
1707%
Breakout
1327%
Star Gunner
598%
Robotank
508%
Atlantis
449%
Crazy Climber
419%
Gopher
400%
Demon Attack
294%
Name This Game
278%
Krull
277%
Assault
246%
Road Runner
232%
Kangaroo
224%
James Bond
145%
Tennis
143%
Pong
132%
Space Invaders
121%
Beam Rider
119%
Tutankham
112%
Kung-Fu Master
102%
Freeway
102%
Time Pilot
100%
Enduro
97%
Fishing Derby
93%
Up and Down
92%
Ice Hockey
79%
Q*bert
78%
H.E.R.O.
76%
Asterix
69%
Battle Zone
67%
Wizard of Wor
67%
Chopper Command
64%
Centipede
62%
Bank Heist
57%
River Raid
57%
Zaxxon
54%
Amidar
43%
Alien
42%
Venture
32%
Seaquest
25%
Double Dunk
17%
Bowling
14%
Ms. Pac-Man
13%
Asteroids
7%
Frostbite
6%
Gravitar
5%
Private Eye
2%
Montezuma's Revenge
0%
At human-level or above
Below human-level
DQN
Best linear learner
0
100
200
300
400
500
600
1,000
4,500%


0740 0000
0630 0000
0740 0000
0630 0000
1625 0000
1445 0000
1470 0000

a





b
1
2
3
4
0
2
Action-Values (Q)
1
0.5
0
-0.5
-1
NO-OP
UP
DOWN

Effect of Experience Replay and Target Q-Network

Extended Data Table 3 | The effects of replay and separating the target Q-network

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

A short review

- Reinforcement Learning
  - Function approximators for end-to-end Q-learning
- Deep Learning
  - Extract high-level feature representations from high-dimensional raw sensory data

Reinforcement Learning + Deep Learning = AI

by David Silver