# Double Deep Q-Network Solution

Yermek Kapushev

Skoltech

2017

# Q-learning

- Under a given policy $\pi$, the true value of action $a$ in a state $s$ is

$$Q_\pi(s, a) = \mathbb{E}[r_1 + \gamma r_2 + \gamma^2 r_3 + \ldots | s_0 = s, a_0 = a, \pi],$$

  where $\gamma \in [0, 1]$.
- The optimal value is

$$Q^*(s, a) = \max_\pi Q_\pi(s, a)$$

- An optimal policy is derived from $Q^*(s, a)$ by taking $\arg\max_a Q^*(s, a)$

# Learning optimal action-value function

Let $Q(s, a, \theta)$ be a model, that approximates Q-function. $\theta$ is a vector of its parameters.

1. Initialize $Q(s, a; \theta)$
2. Observe:

$$s' \leftarrow s, \quad a' \leftarrow a$$

   get new state $s$, get reward $r$.
3. Update weights

$$\theta \leftarrow \theta + \alpha(r + \gamma \max_{\tilde{a}} Q(s, \tilde{a}, \theta) - Q(s', a', \theta))\nabla Q(s', a', \theta)$$

4. Go to step 2.

RL is known to be unstable, or even to diverge when a nonlinear function approximator such as an ANN is used to represent Q-function:

- Correlations in the sequence of observations
- Small updates of Q may significantly change the policy $\Rightarrow$ change of data distribution
- Correlations between action values $Q(s, a)$ and target values $r + \gamma \max_a Q(s, a)$

Solution:

- Randomization over the data *experience replay* $\Rightarrow$ remove correlations in the observations sequence and smoothing over changes in the data distribution.
- *Iterative update* of action-values and target values $\Rightarrow$ reduce correlations between them

# Experience replay

- Store the agents experience $e_t = (s_t, a_t, r_t, s_{t+1})$ at each time step in a data set $D = \{e_1, \ldots, e_t\}$
- Apply Q-learning updates on mini-batches of experience $(s, a, r, s') \sim \mathit{Uniform}(D)$

# Double Q-learning

- The following loss function is used:

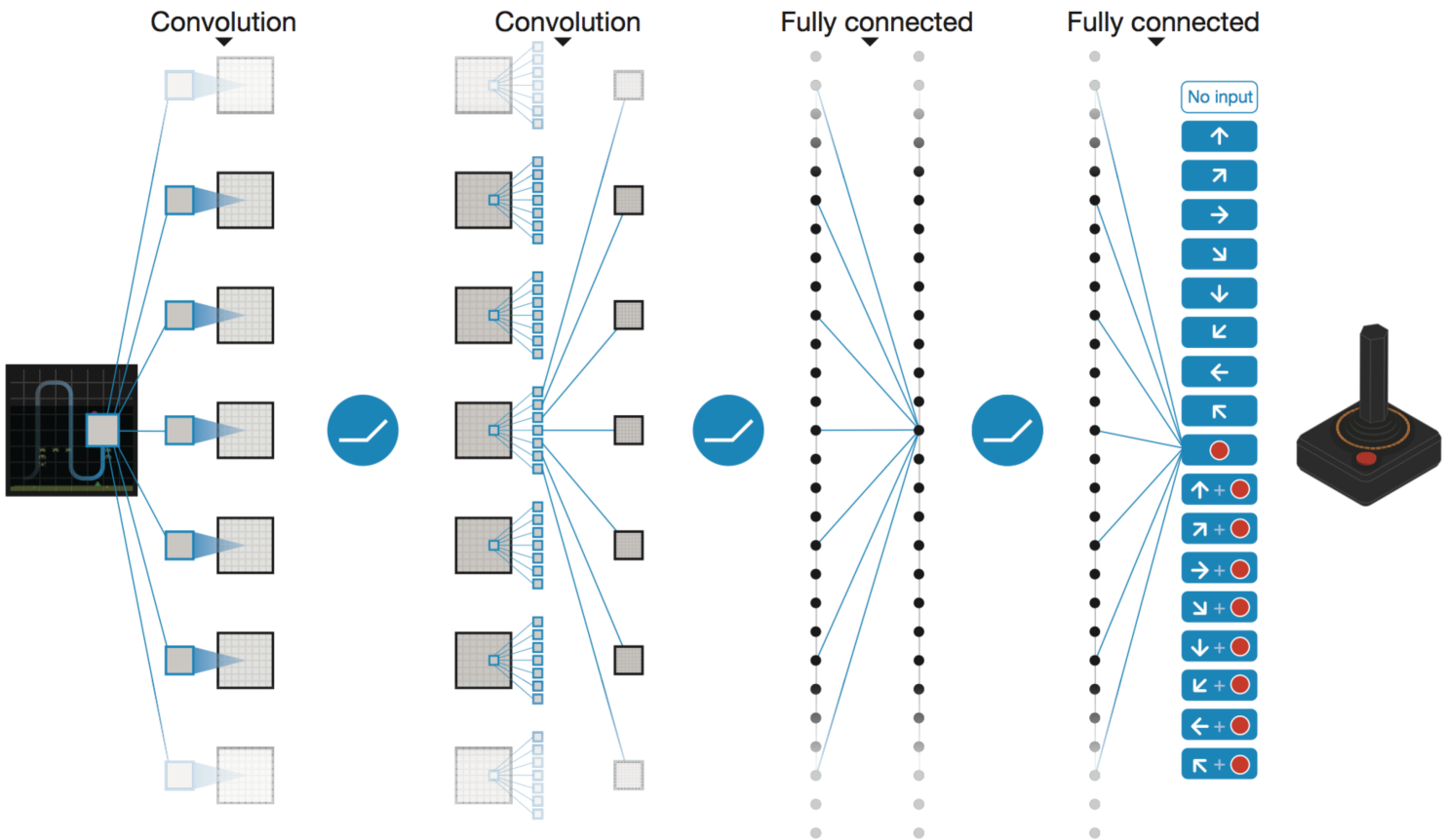
$$L(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim U(D)}\left[\left(r + \gamma \max_{a'} Q(s', a', \theta_i^-) - Q(s, a, \theta_i\right)^2\right],$$

  where $\theta_i^-$ — parameters used to compute action-value, $\theta_i$ — parameters used to compute target value.
- Parameters $\theta_i^-$ are updated every $C$ steps with the parameters $\theta_i$.

# Learning to play Atari games

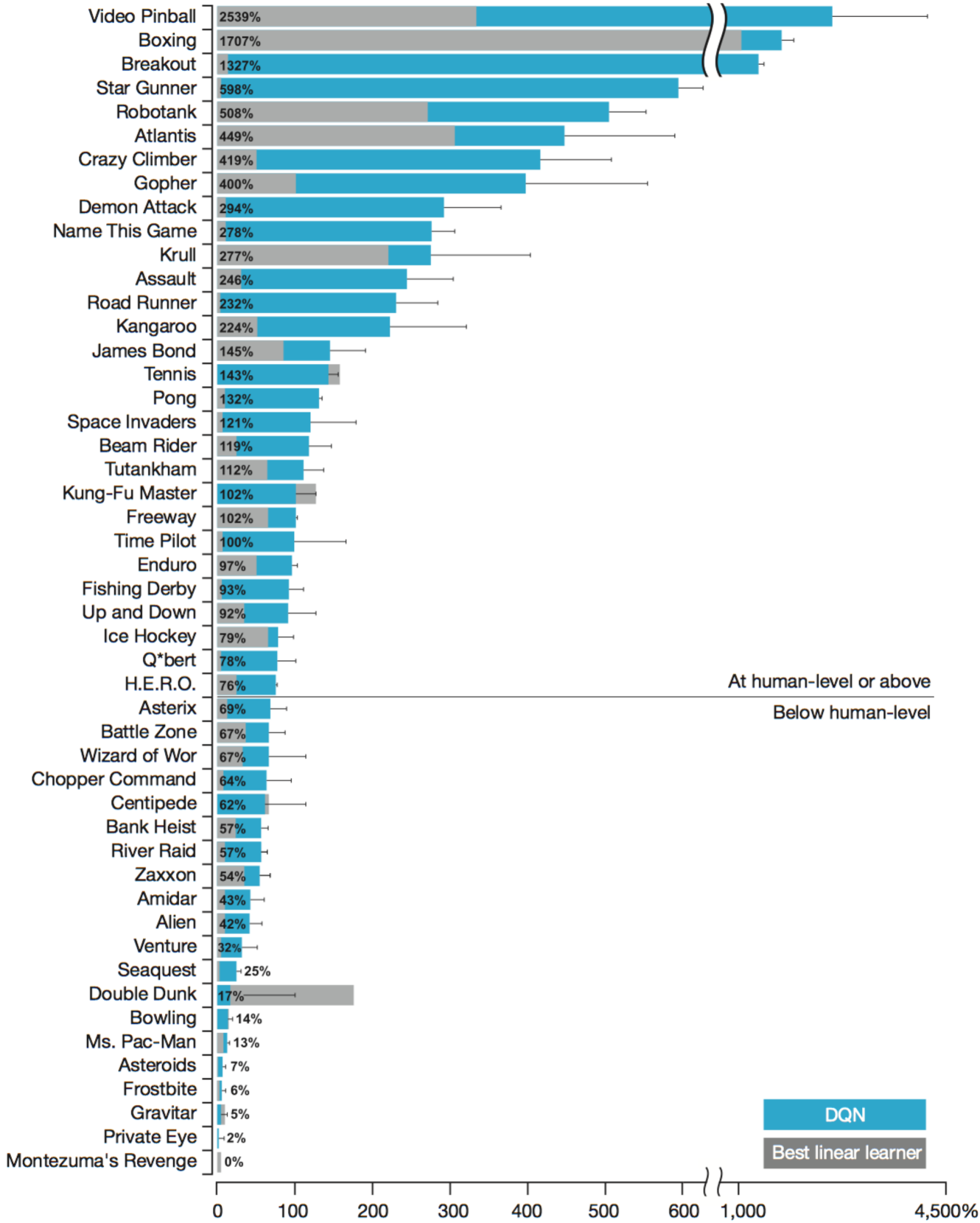
- Arcade Learning Environment (ALE) — python API to ROMs emulator.
- ANN is used for Double DQN
- Input data: 210 * 160 video at 60 Hz
- Output data: action, game score
- Internal game state is not observed. Only frames (images) are observed.

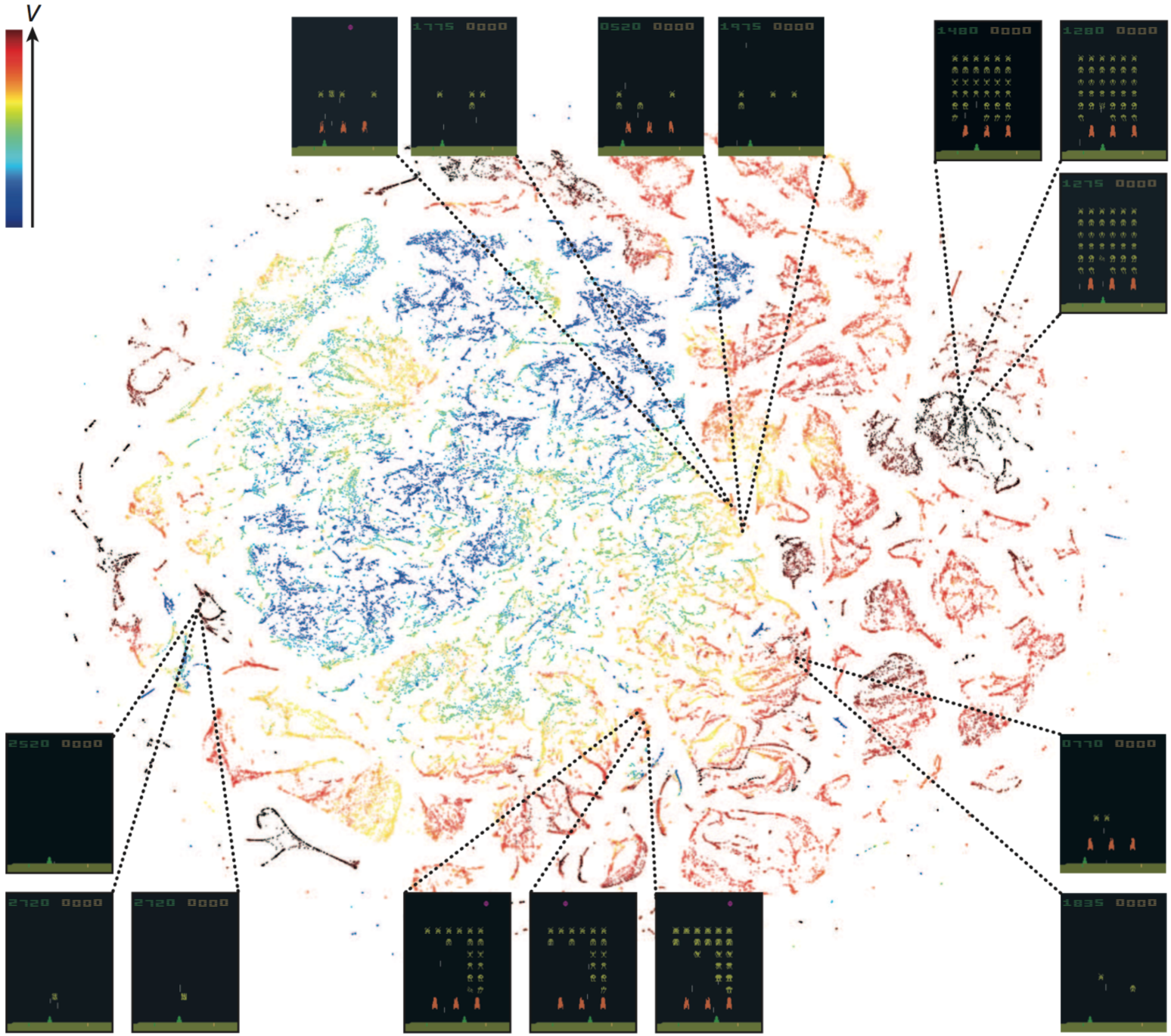
# Deep convolutional network architecture




Input: 4 frames of 84 * 84

Output: single output for each possible action


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


V