Human-level Control Through Deep Reinforcement Learning

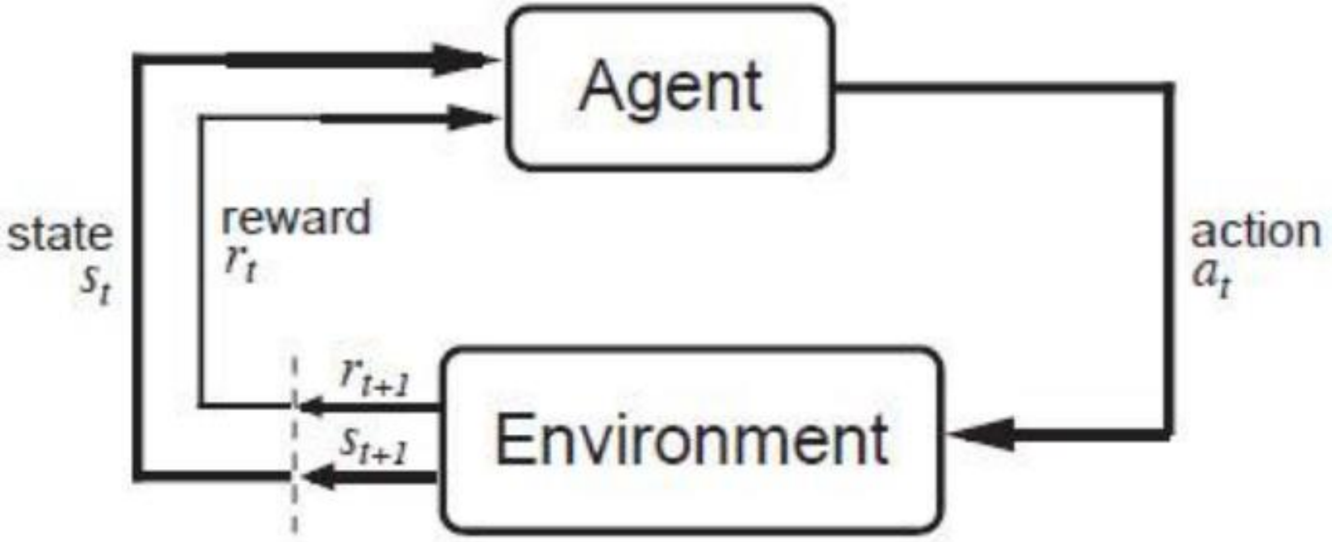
Google DeepMind

Nature 2015

Introduction

Markov Decision Process

- State
- Action
- Reward



Introduction

- Action value function
  - Discounted future reward (environment is stochastic)

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

$$Q^\pi(s,a) = E_\pi\{R_t | s_t = s, a_t = a\} = E_\pi\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\}$$

Introduction

- Q-learning

- Bellman Equation

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

```
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

-Limitation:

1.Very limited states/actions

2.Can not generalize to unobserved states

Introduction

Deep Q-network (DQN)

- Q learning plus
- Function approximator:

  Deep neural networks to approximate optimal action-value function

$$Q^*(s,a) = \max_{\pi} \mathbb{E}\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots \middle| s_t = s,\ a_t = a,\ \pi\right]$$

Problem

Stability issues with Deep RL

- Naïve Q-learning oscillates or diverges with neural nets

  1. Data is sequential
     - Successive samples are correlated, non-i.i.d.
  2. Policy changes rapidly with slight changes to Q-values
     - Policy may oscillate
     - Distribution of data can swing from one extreme to another

Experience Replay

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

- Take action a_t according to ε-greedy policy

  (Choose "best" action with probability 1- ε, and selects a random action with probability ε)

- Store transition (s_t, a_t, r_t, s_{t+1}) in replay memory $\mathcal{D}$ (Huge data base to store historical samples)
- Sample random mini-batch of transitions (s, a, r, s') from $\mathcal{D}$
- Optimize MSE between Q-network and Q-learning targets, e.g.

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

Fixed target Q-network

To avoid oscillations, fix parameters used in Q-learning target

- Compute Q-learning targets w.r.t. old, fixed parameters θ_i^-

$$r + \gamma \max_{a'} Q(s', a'; \theta_i^-)$$

- Optimize MSE between Q-network and Q-learning targets

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

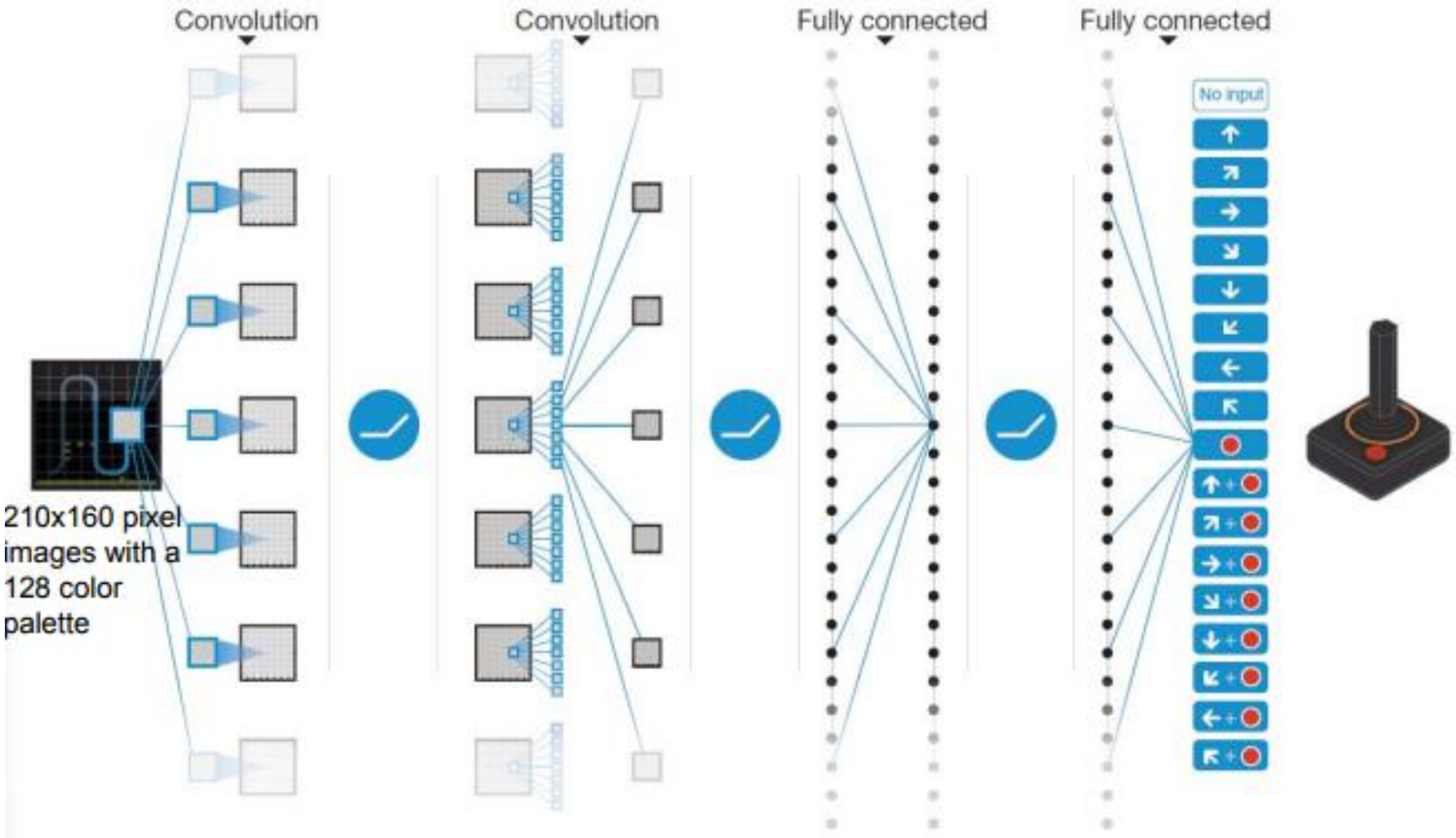
- Periodically update fixed parameters $\theta_i^- \leftarrow \theta_i$

Core components of DQN

Deep Q-Network provides a stable solution to deep value-based RL

1. Use experience replay
   - Break correlations in data, bring us back to i.i.d. setting
   - Learn from all past policies
   - Using off-policy Q-learning
2. Freeze target Q-network
   - Avoid oscillations
   - Break correlations between Q-network and target

Model:Train this agent on Atari 2600 games



- The input to the neural network consists of an 84x84x4 image produced by the pre-processing map ϕ
- Input state is stack of raw pixels from last 4 frames

Model

- State: Sequences of action and observation

$$s_t = x_1, a_1, x_2, \ldots, a_{t-1}, x_t,$$

- Action: Legal game action set

$$\mathcal{A} = \{1, \ldots, K\}.$$

- Reward: Change in game score

How to train DQN

Loss function :

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

Differentiating the loss function w.r.t. the weights we arrive at following gradient :

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

Do gradient descent:

$$\theta_{i+1} = \theta_i + \alpha \cdot \nabla_{\theta_i} L_i(\theta_i)$$

Algorithm

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**

    With probability ε select a random action a_t

    otherwise select $a_t = \mathrm{argmax}_a Q(\phi(s_t), a; \theta)$

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

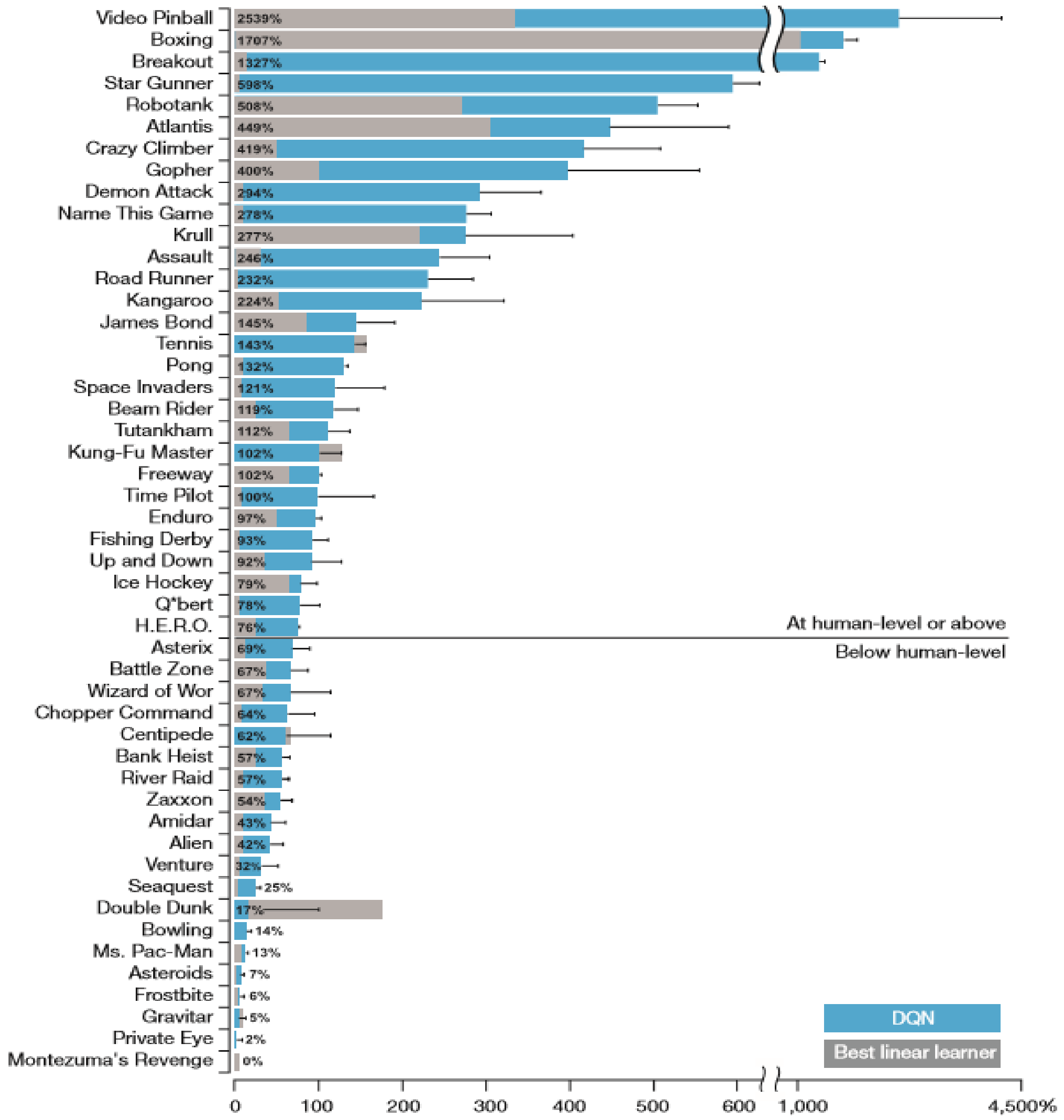
    Perform a gradient descent step on $\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2$ with respect to the network parameters θ

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

  End For

End For

Evaluation



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%

Evaluation

DQN

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

Conclusion

- A single architecture can successfully learn policy with only minimal prior knowledge
- And the successful integration of RL with deep neural network was mainly dependent on a replay algorithm

- However, games demanding more temporally extended planning strategy still are challenge for all existing agent including DQN
- Experience replay can be better