Deep Reinforcement Learning

on Car Racing Game

Jiahao Zhang, Minghe Ren, Weixuan Jiang, Zhonghao Guo

BOSTON UNIVERSITY

Outline

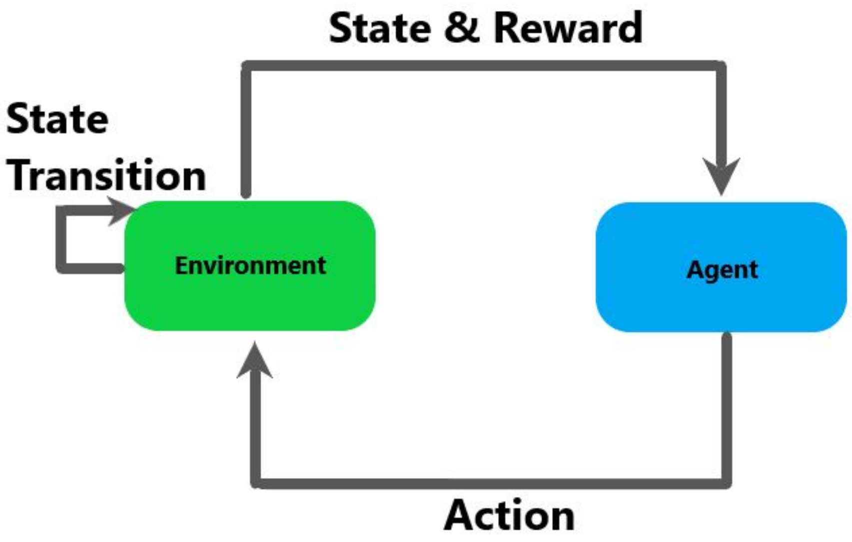
- Background
- Game Environment
- Deep Q-Network (DQN)
- Double DQN (DDQN)
- Dueling DQN
- Experimental Result

Outline

- Background
- Environment
- Deep Q-Network (DQN)
- Double DQN (DDQN)
- Dueling DQN
- Experimental Result

Reinforcement Learning Review

- Enables an **agent** to learn in an interactive **environment**
- Each **action** influences the agent's future state
- Success is measured by **reward**
- Goal: select actions to **maximize** future reward



Outline

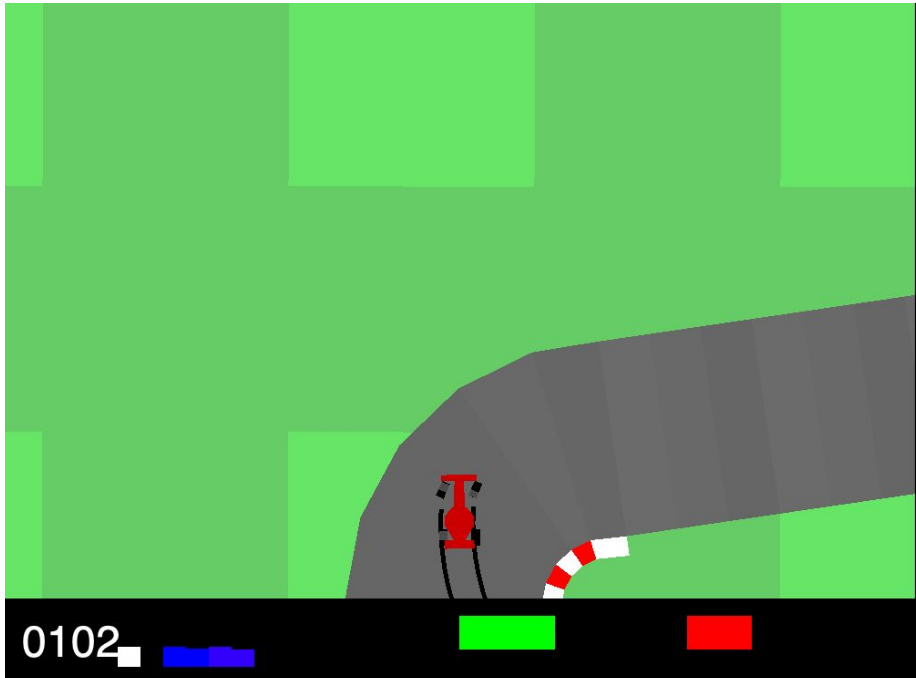
- Background
- Game Environment
- Deep Q-Network (DQN)
- Double DQN (DDQN)
- Dueling DQN
- Experimental Result

Game Environment

- Car-Racing-v0 from gym library as our basic environment
- Agent: car
- States: pics taken every 4 frames of 96×96 pixels
- Actions: forward, brake, left, right
- Reward:

  if action detected, reward -= 0.1

  if running out of playfield, reward -= 150



Outline

- Background
- Gaming Environment
- Deep Q-Network (DQN)
- Double DQN (DDQN)
- Dueling DQN
- Experimental Result

DQN -- The Combination of Q-learning and Neural Network

- Turn Q-table into Q-Network

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right]$$

$$Q_t = r_t + \gamma \max_{a'} Q(S_{t+1}, a'; \theta)$$

Reward

The action that maximize Q-value

DQN

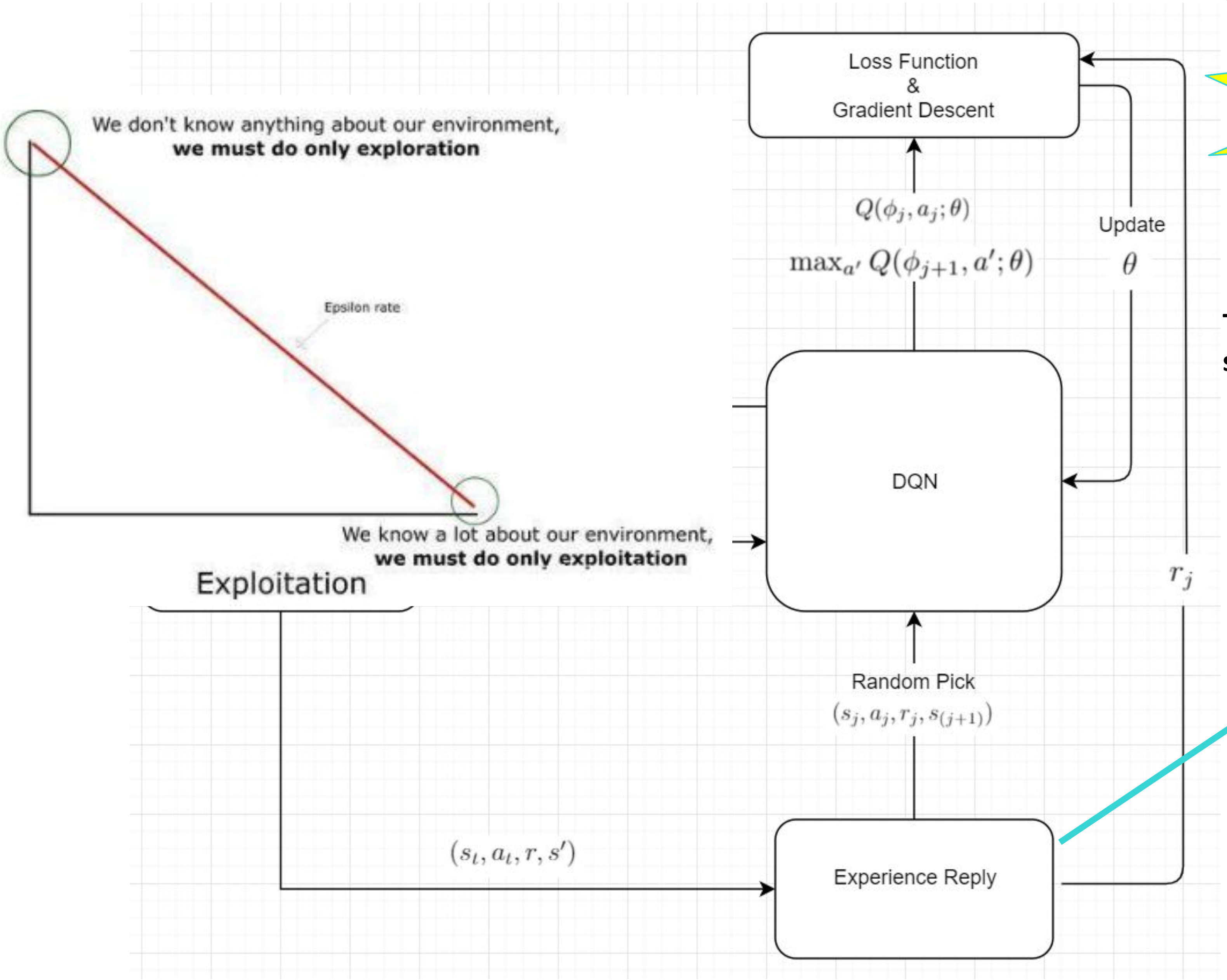
Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
    Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
    for $t = 1, T$ **do**
        With probability ϵ select a random action a_t
        otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
        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 $\mathcal{D}$
        Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
        Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
        Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
    end for
end for

Algorithm from *Playing Atari with Deep Reinforcement Learning*

DQN

Exploration



The prev_state and new_state won't change too much.

The correlation between state and state is too strong!

Solve:

Random sampling to minimize the correlation between states

Q-net Architecture

Conv1 = 7*7	filter = 8	stride = 3
max_pooling1	filter = 2	stride = 2
Conv2 = 3*3	filter = 16	stride = 2
max_pooling2	filter = 2	stride = 2
fc1	node = 256	
fc2	node = 4	

Outline

- Background
- Gaming Environment
- Deep Q-Network (DQN)
- Double DQN (DDQN)
- Dueling DQN
- Experimental Result

Double DQN(DDQN)

Improvement in Q value estimation

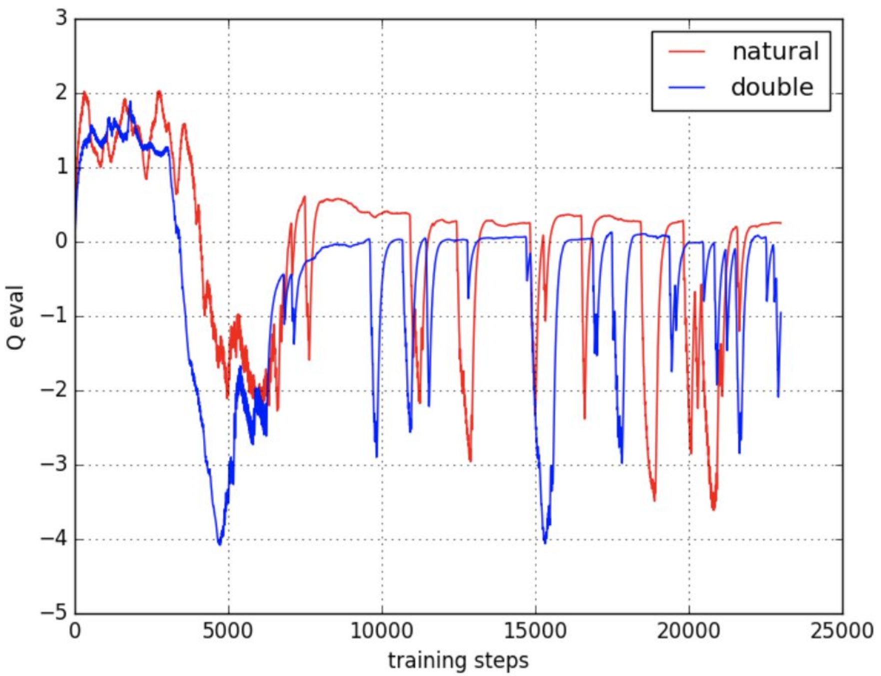
How Double DQN works?

Original DQN target Y: $Y_t^{\text{DQN}} \equiv R_t + \gamma \max_a Q(S_{t+1}, a; \boldsymbol{\theta}_t^-)$

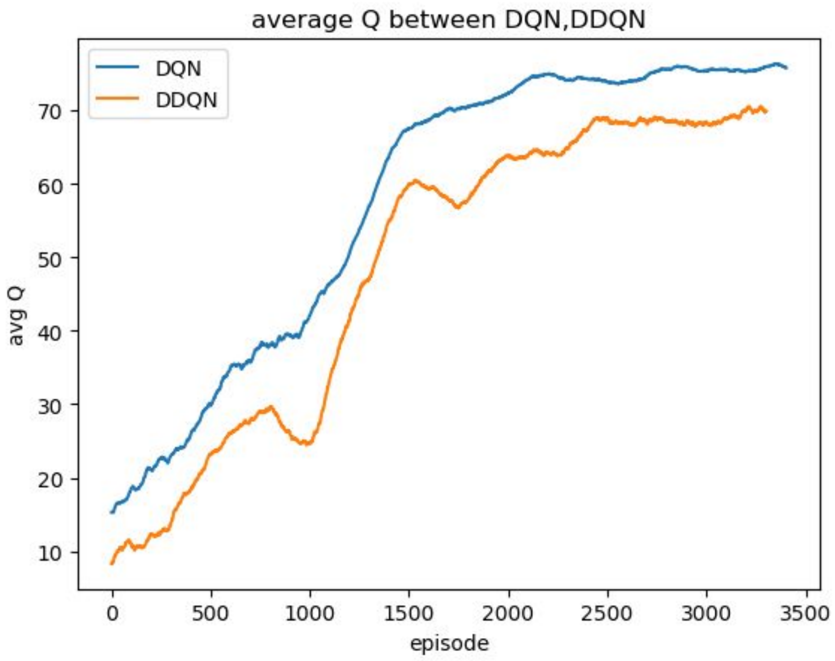
Double DQN target Y: $Y_t^{\text{DoubleQ}} \equiv R_t + \gamma Q(S_{t+1}, \operatorname{argmax}_a Q(S_{t+1}, a; \boldsymbol{\theta}_t); \boldsymbol{\theta}_t')$

In standard Q-learning and DQN, they both use same values to select and to evaluate an action. This may cause overestimated values, resulting overoptimistic value estimates. In double DQN, we use another second set of weights to evaluate the greedy policy estimating by the first set of weights (original DQN weights).

Double DQN(DDQN)



An example showing that the double DQN tends to reduce the overestimation (online source)



Our results also showes that the double DQN tends to reduce the overestimation

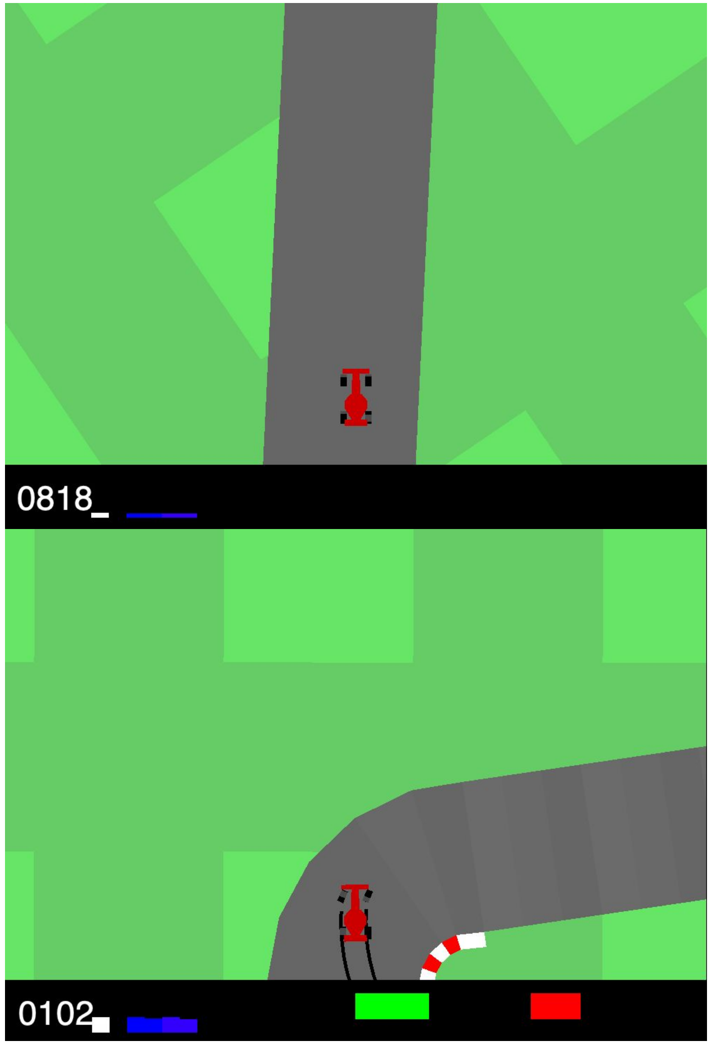
Outline

- Background
- Gaming Environment
- Deep Q-Network (DQN)
- Double DQN (DDQN)
- Dueling DQN
- Experimental Result

Dueling Network

- Why dueling network?
  There are many states where it's unnecessary to estimate the value of each action choice. Its actions do not affect the reward in any relevant way.

  Improvement in network structure

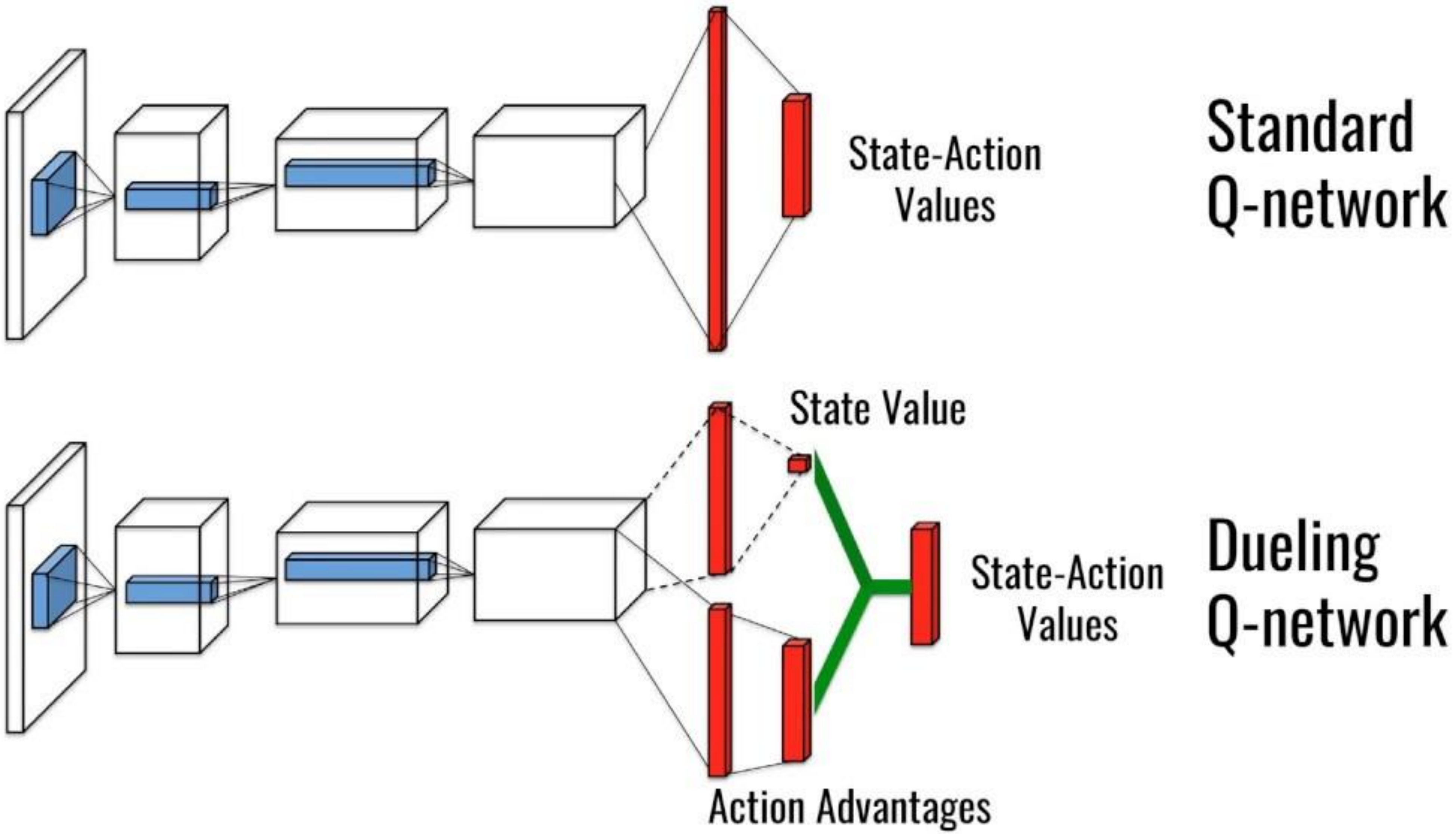


Dueling Network

- What is the new Q value here?
  state-value function $V(s;\theta,\beta)$, *advantage function* $A(s,a;\theta,\alpha)$

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + A(s,a;\theta,\alpha)$$

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + \left(A(s,a;\theta,\alpha) - \frac{1}{|A|}\sum_{a'} A(s,a';\theta,\alpha)\right)$$


State-Action
Values
Standard
Q-network
State Value
State-Action
Values
Dueling
Q-network
Action Advantages

Outline

- Background
- Gaming Environment
- Deep Q-Network (DQN)
- Double DQN (DDQN)
- Dueling DQN
- Experimental Result

Result

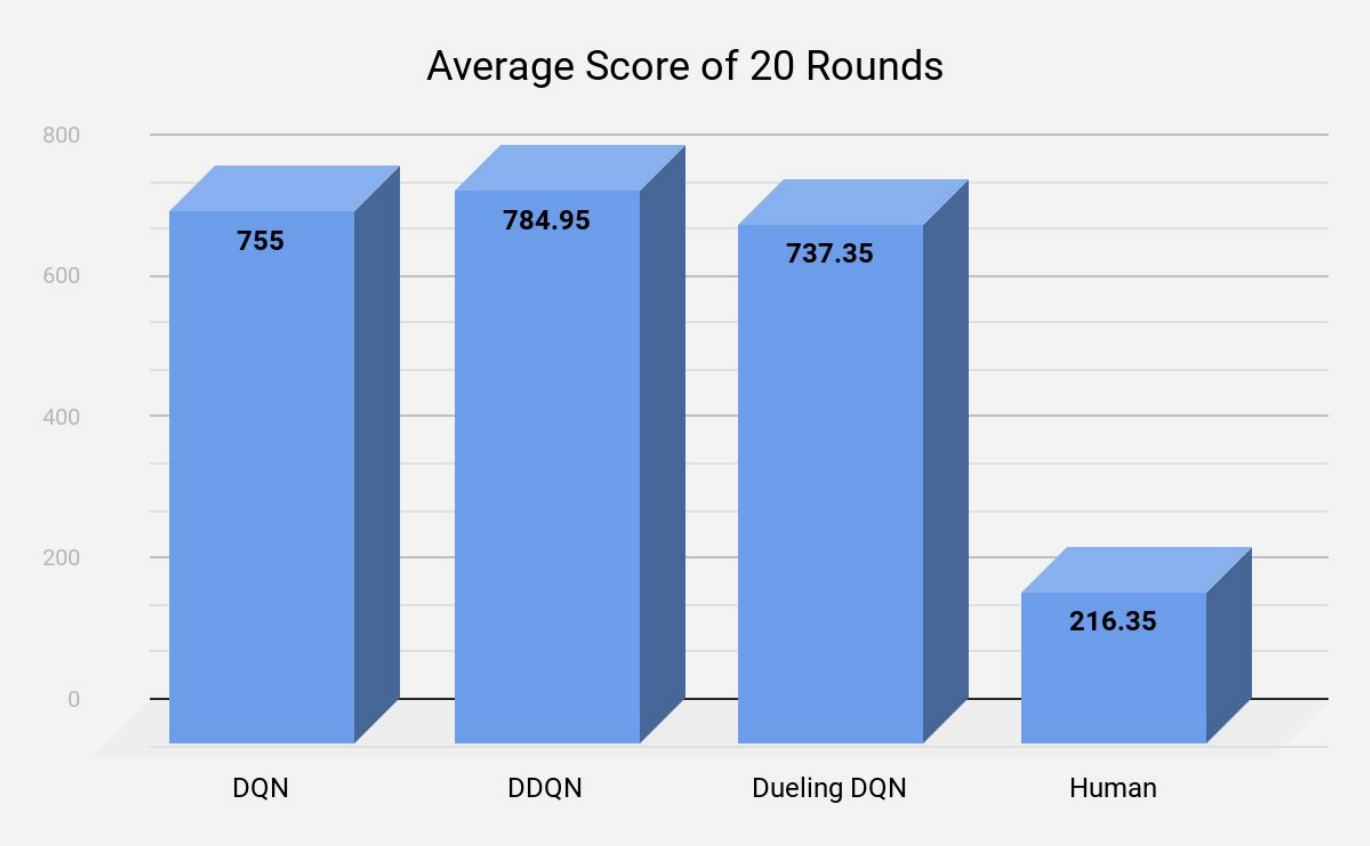
Before training

After 12h training

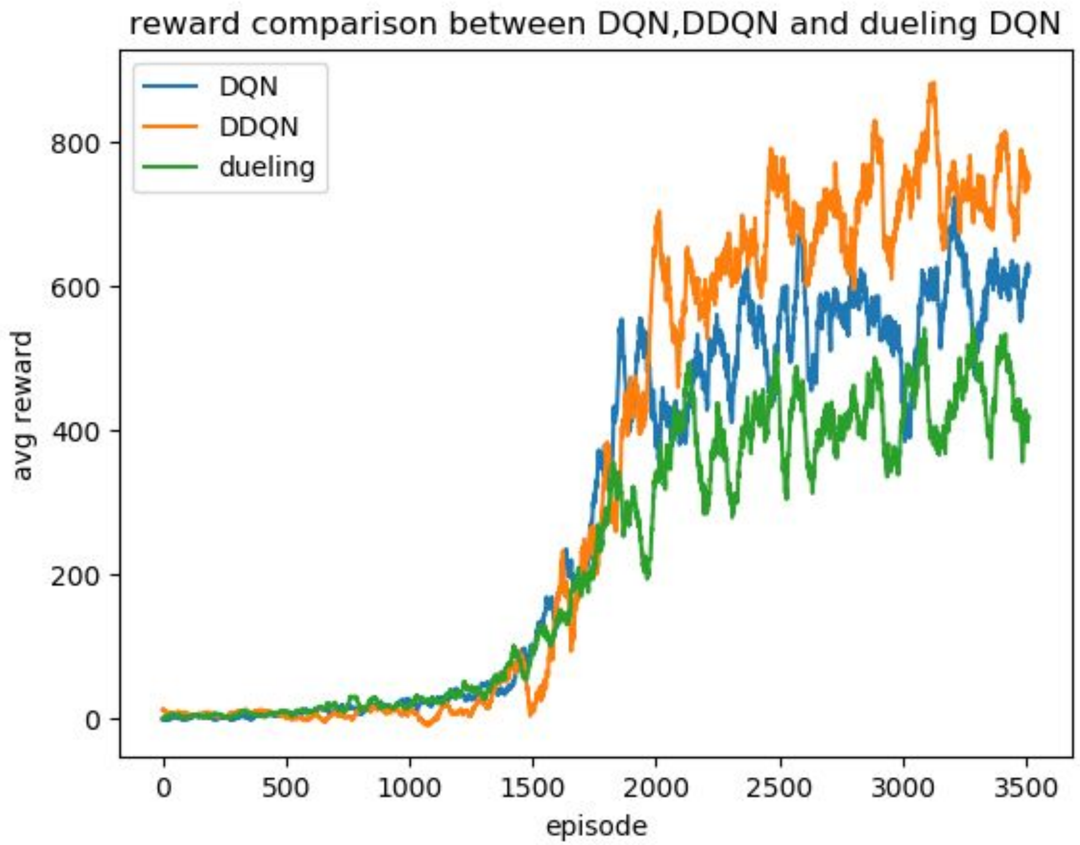
Test Results

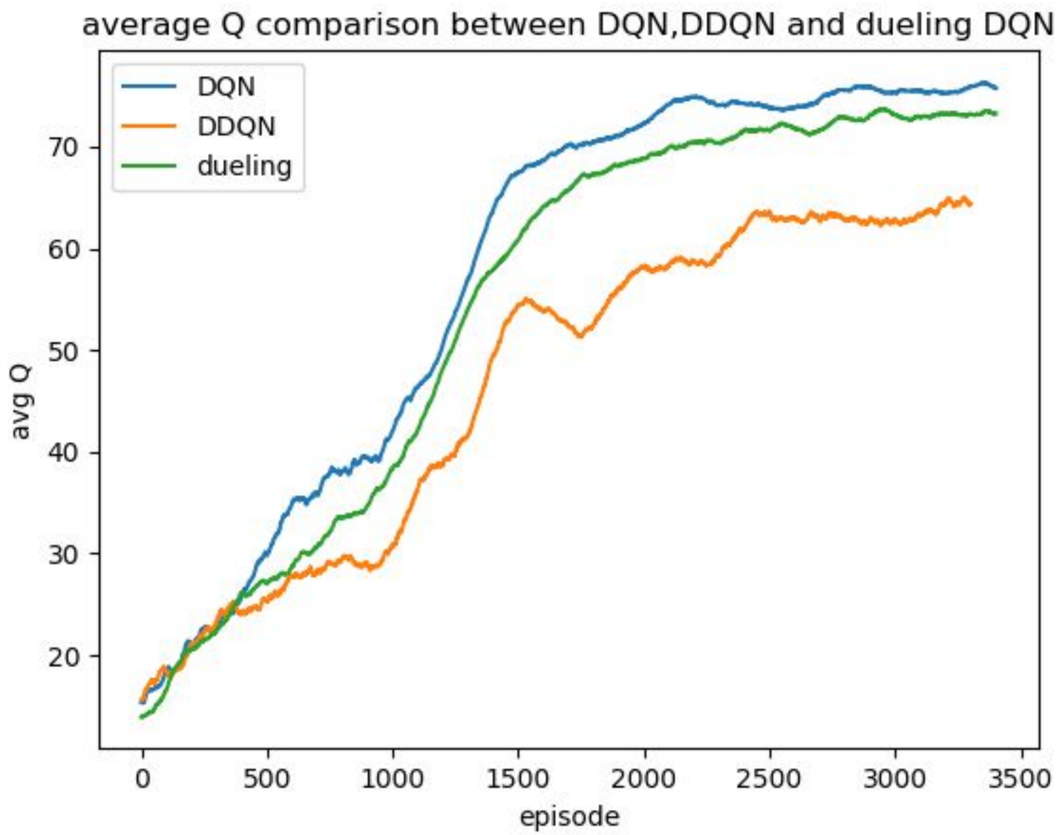
DQN	DDQN	Dueling DQN	Human
755	784.95	737.35	216.35

Average score of 20 rounds (episodes)



Training Results





Thoughts on Dueling DQN bad results

- The view of the car is narrow, every action value estimation is significant in every state.
- The number of nodes in fully-connected layer in dueling network is smaller. Thus, features extracted is not sufficient.

END