Deep Q Networks

Saasha Nair

Deep Q Networks (DQN)

TL; DR

Does not learn an explicit map/model of the environment

During training, the policy being learnt is different from the one collecting learning data from the environment

An off-policy, value-based, model-free RL algorithm. It learns to act in an environment with discrete action space by estimating Q-values.

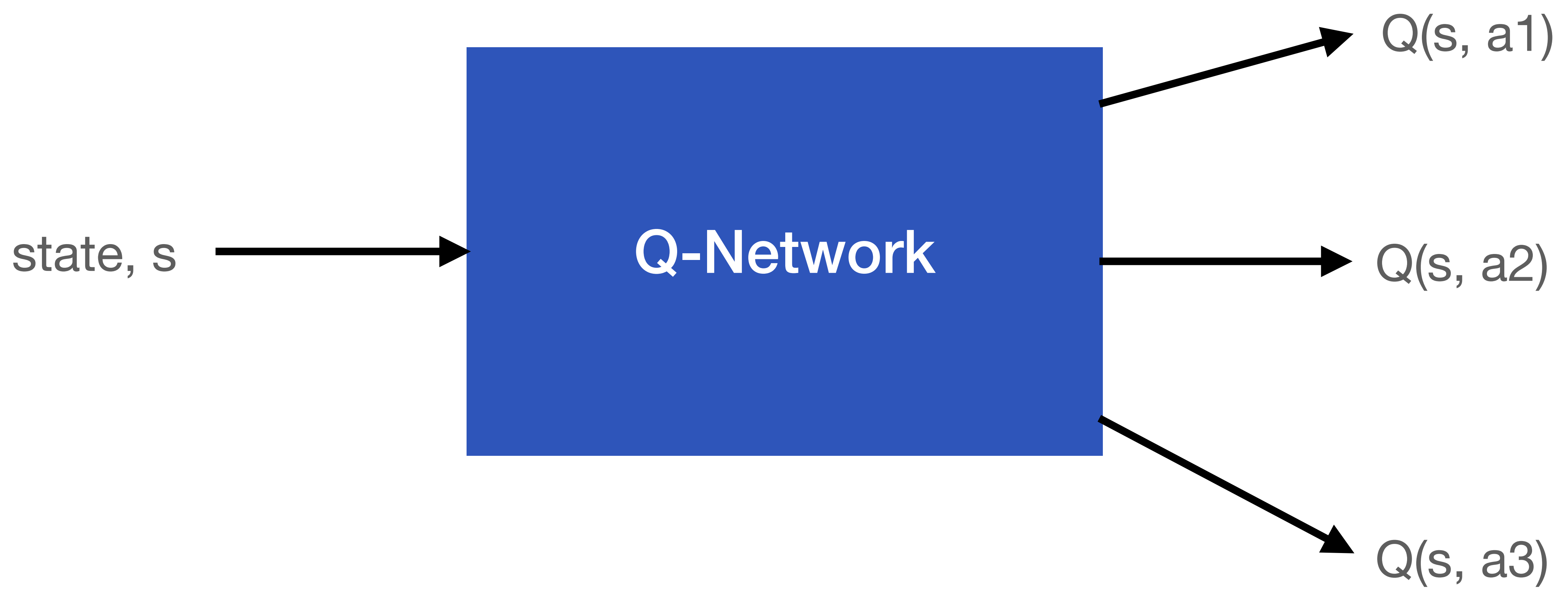
How good is a given state-action pair?

Policy not explicitly learnt, but derived from the learnt value function

Idea behind DQN

Q(s, a) = expected return from following a particular policy after performing action a in state s

- Allow a neural network to learn Q-values



Idea behind DQN

reward + discount * max_a' Q(s', a')

$$\text{Loss} = (\text{TargetQ} - \text{PredictQ})^2$$

Q(s, a)

But this naive implementation suffers from instability…

Non Stationary Targets

Challenge #1

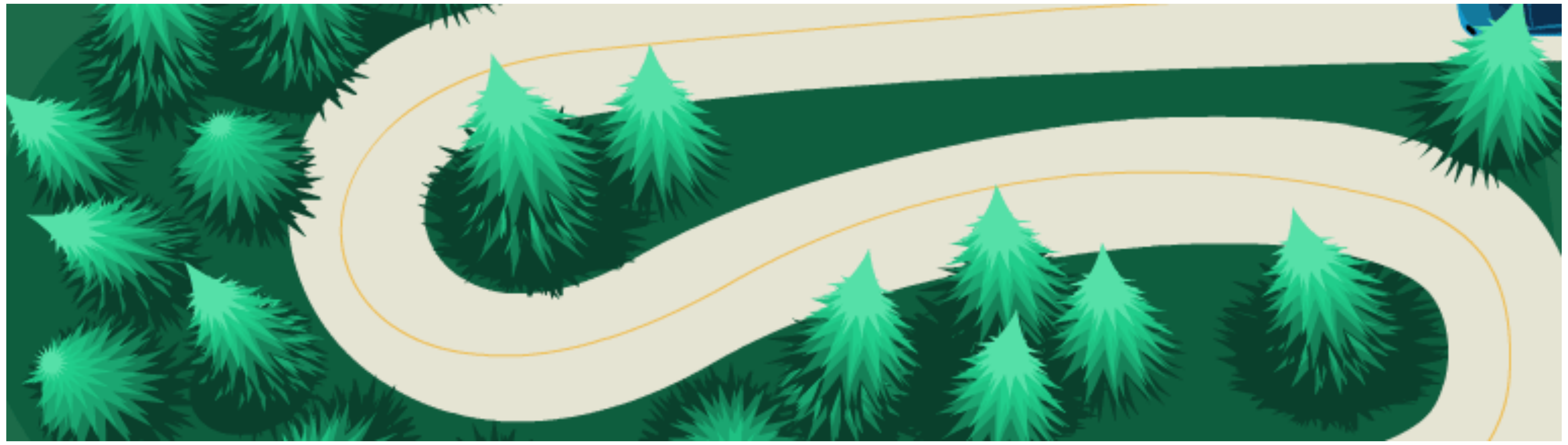
The problem:

The solution: *Target Networks*

GIF from: https://steamcommunity.com/sharedfiles/filedetails/?id=433115079

Non IID Data

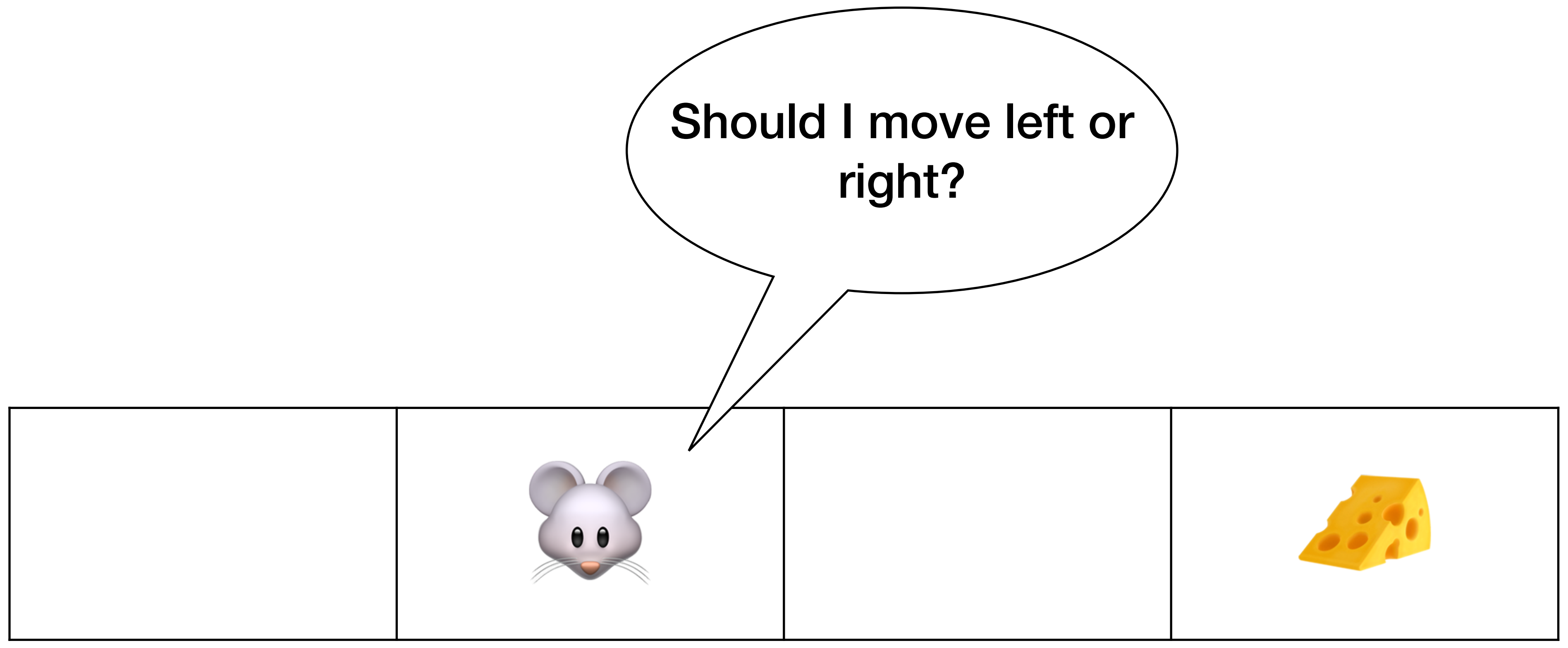
Challenge #2

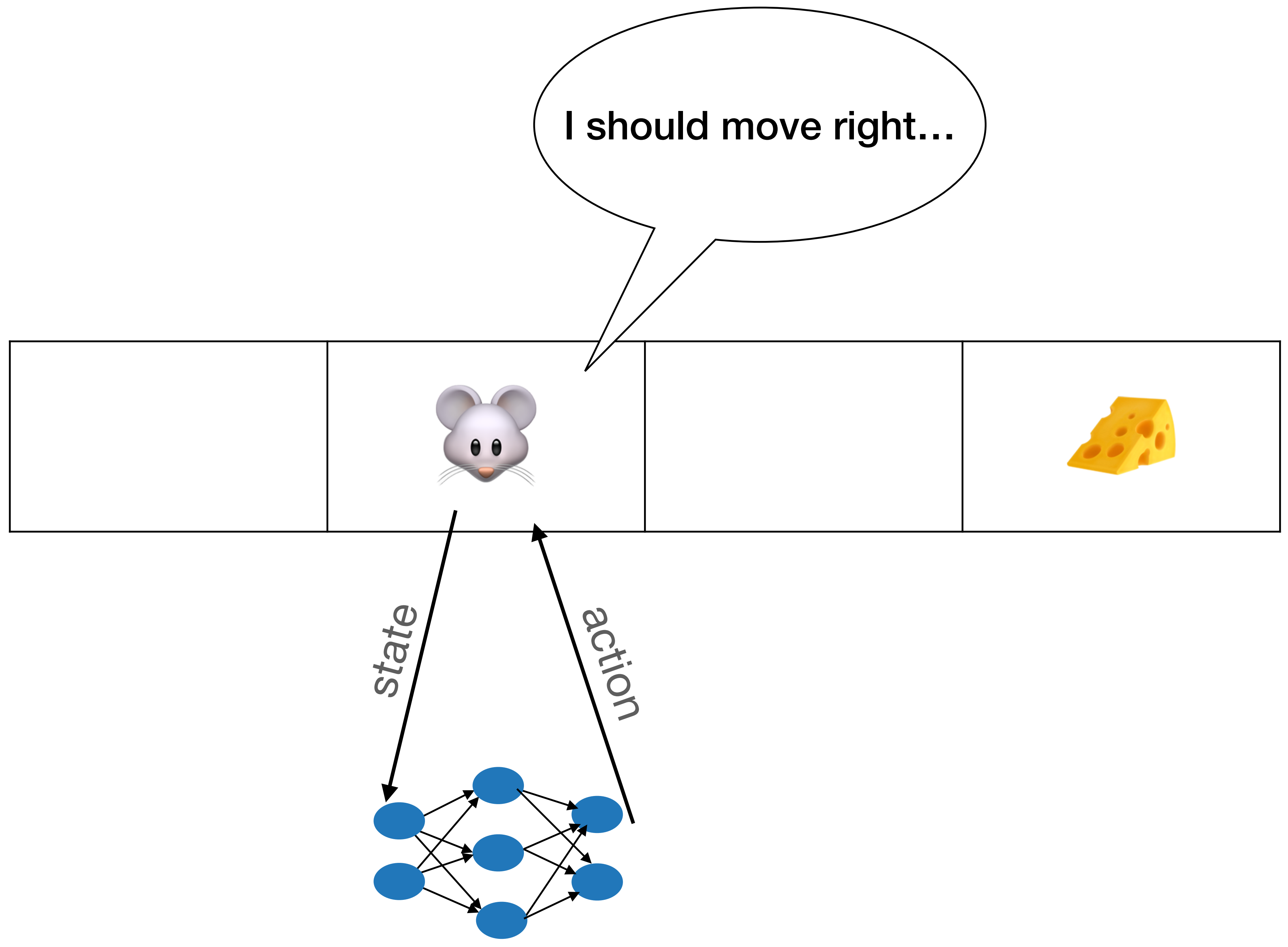
The problem: 

The solution: *Replay Buffer*

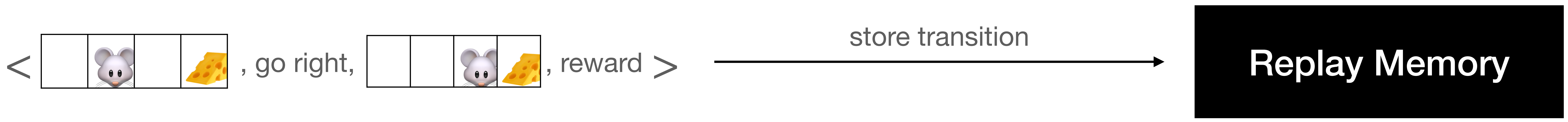
GIF from: https://coherent-labs.com/posts/create-motion-path-animation-animate/

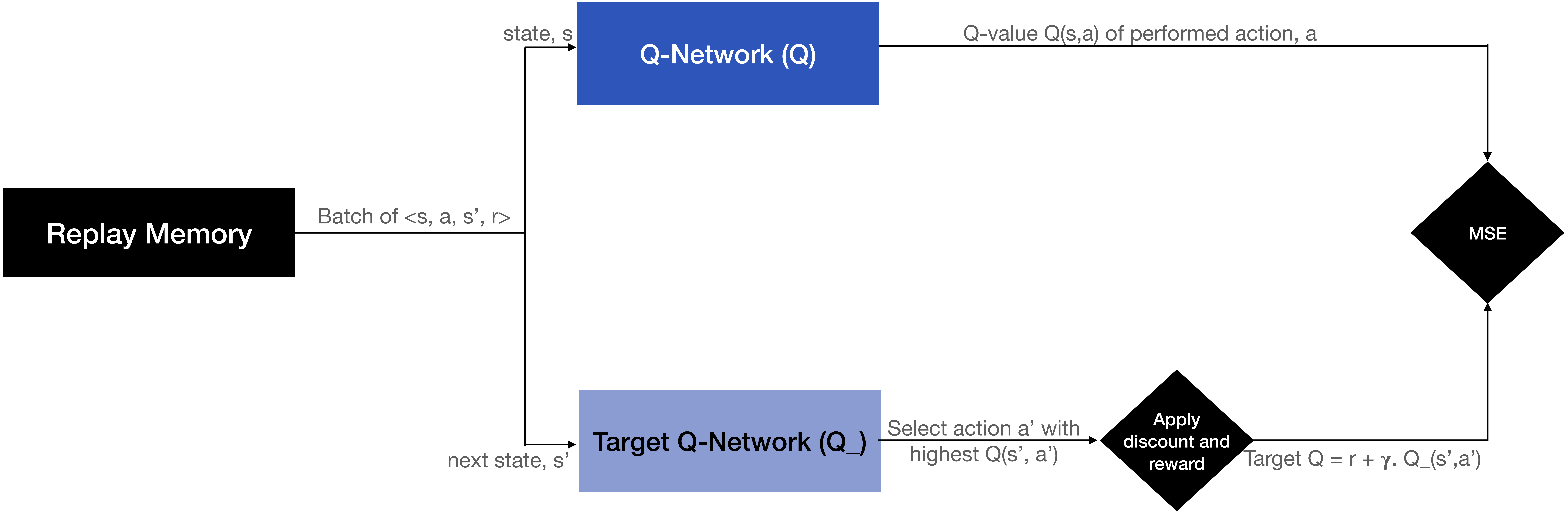
DQN: Intuition




I should move right…
state
action

		🐭	🧀

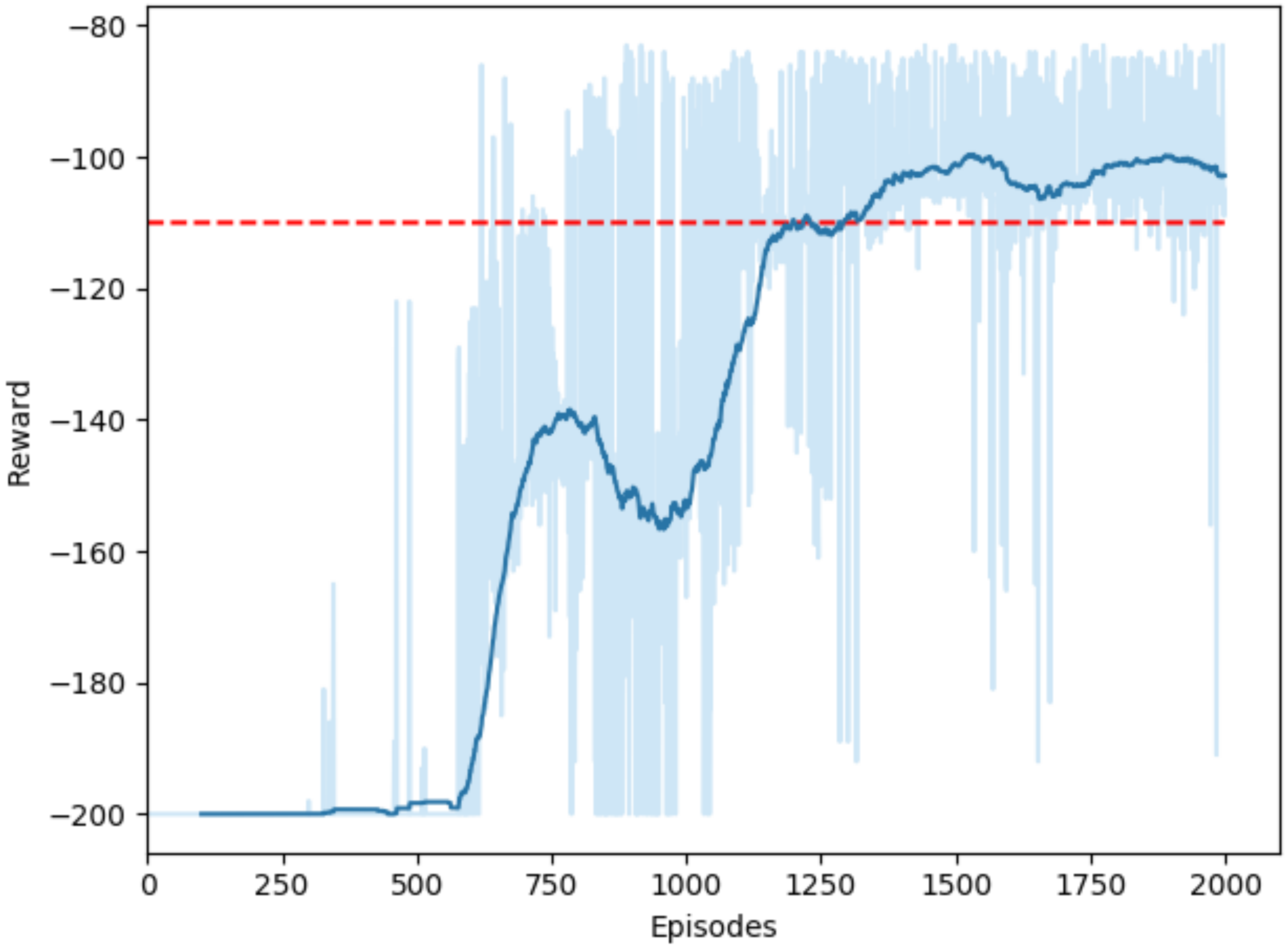

<
, go right,
, reward >
store transition
Replay Memory

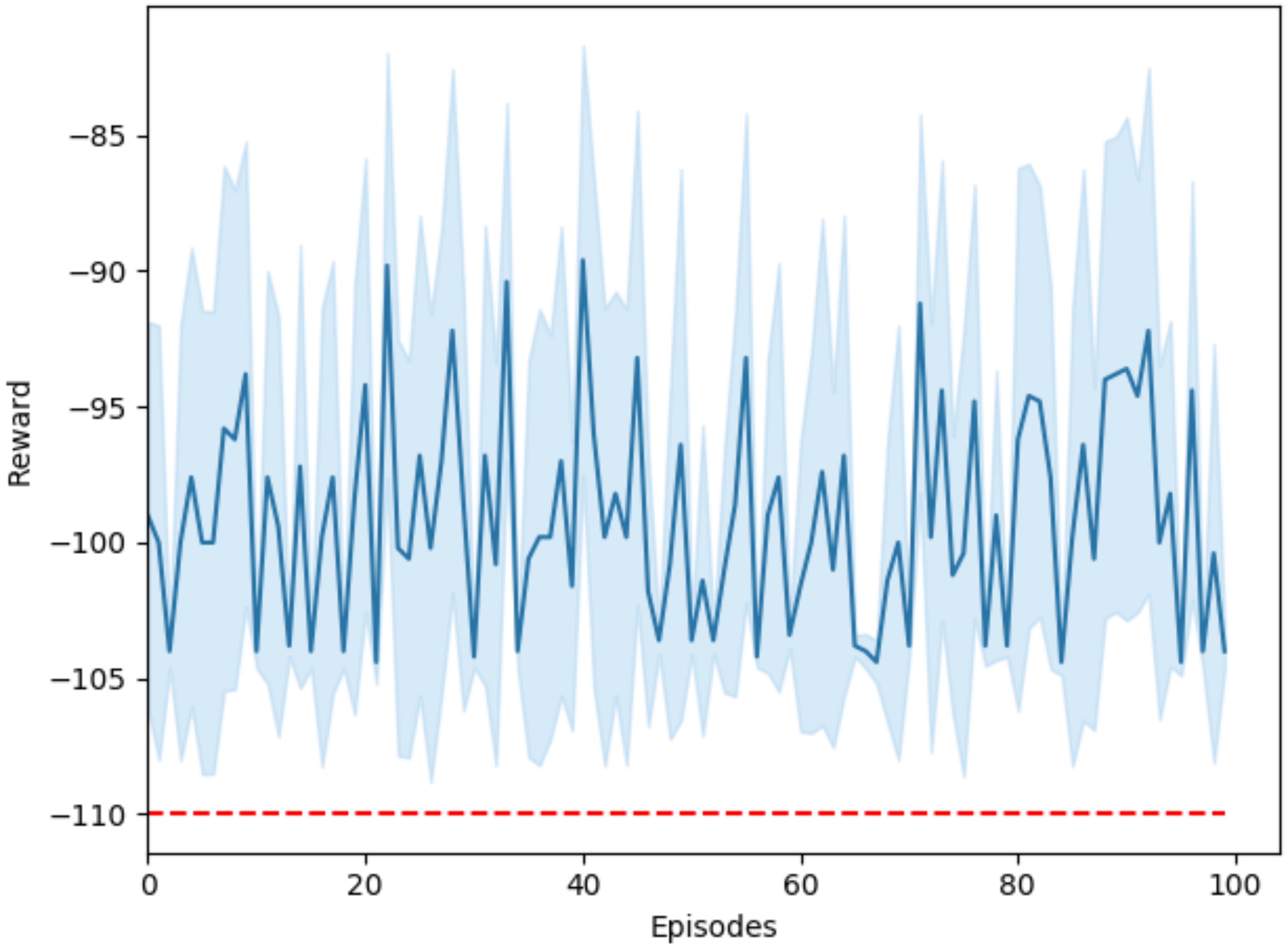


state, s
Q-Network (Q)
Q-value Q(s,a) of performed action, a
Replay Memory
Batch of <s, a, s', r>
MSE
Target Q-Network (Q_)
next state, s'
Select action a' with highest Q(s', a')
Apply discount and reward
Target Q = r + γ. Q_(s',a')

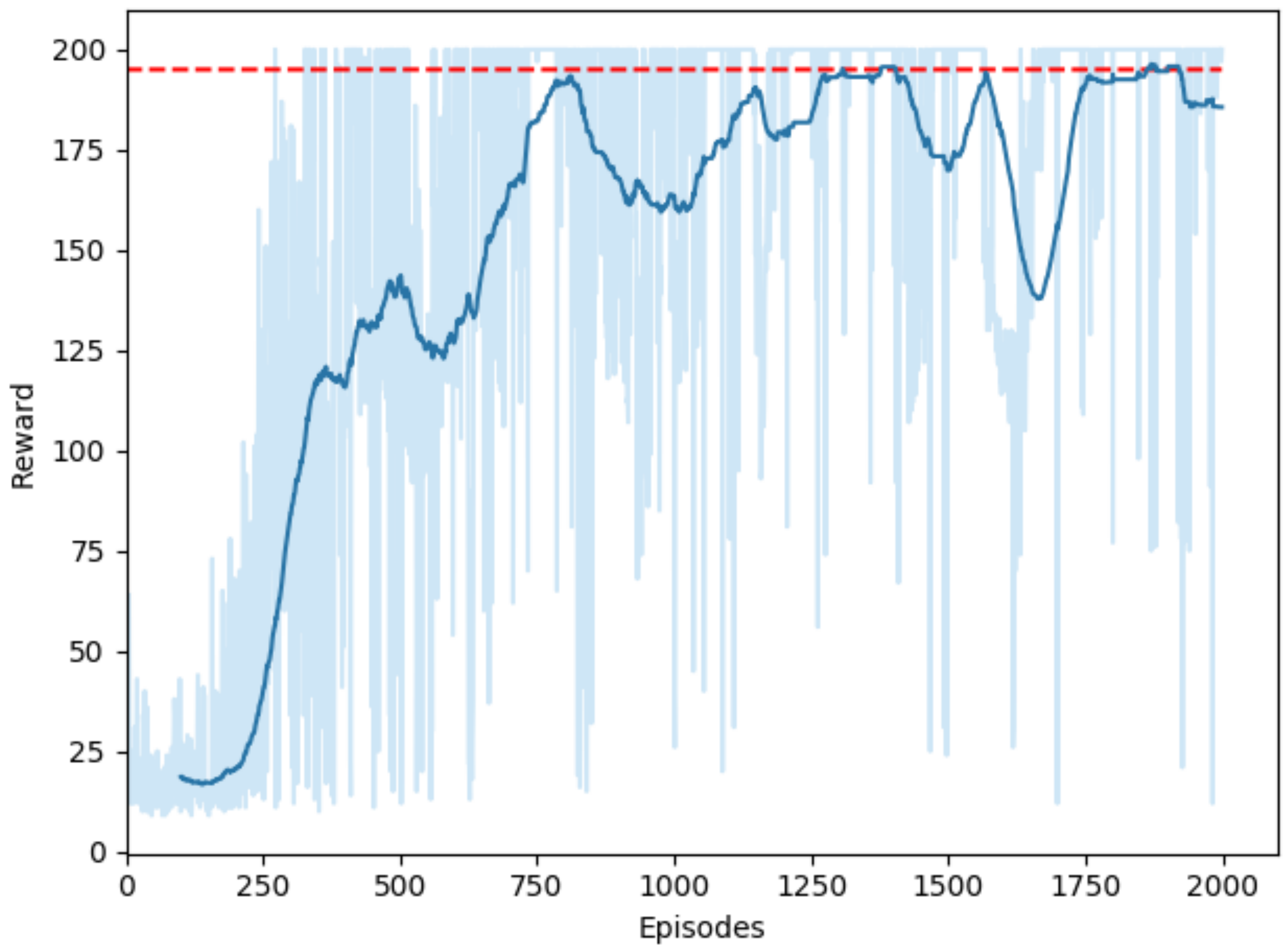
Results

MountainCar-v0

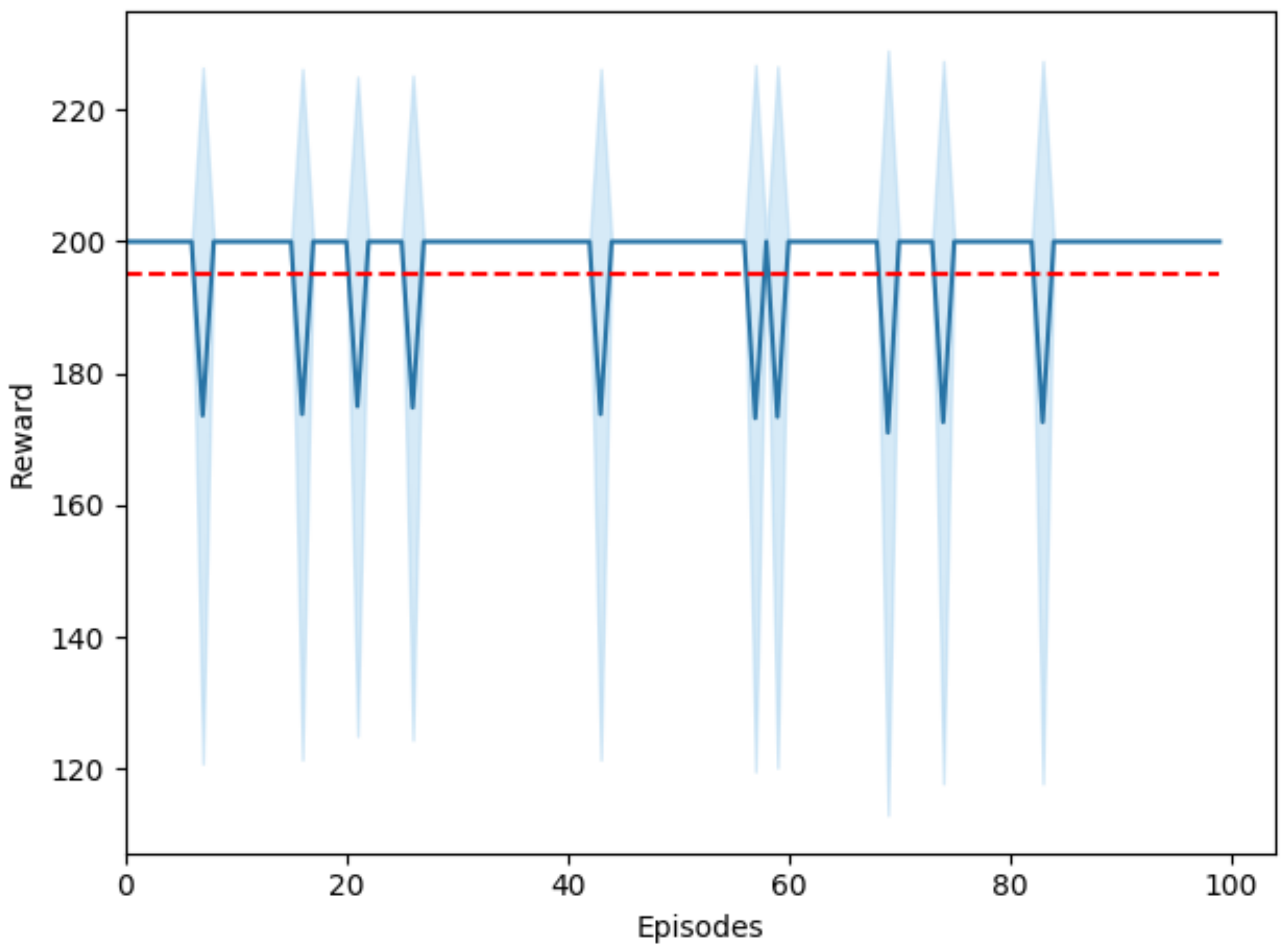




CartPole-v0

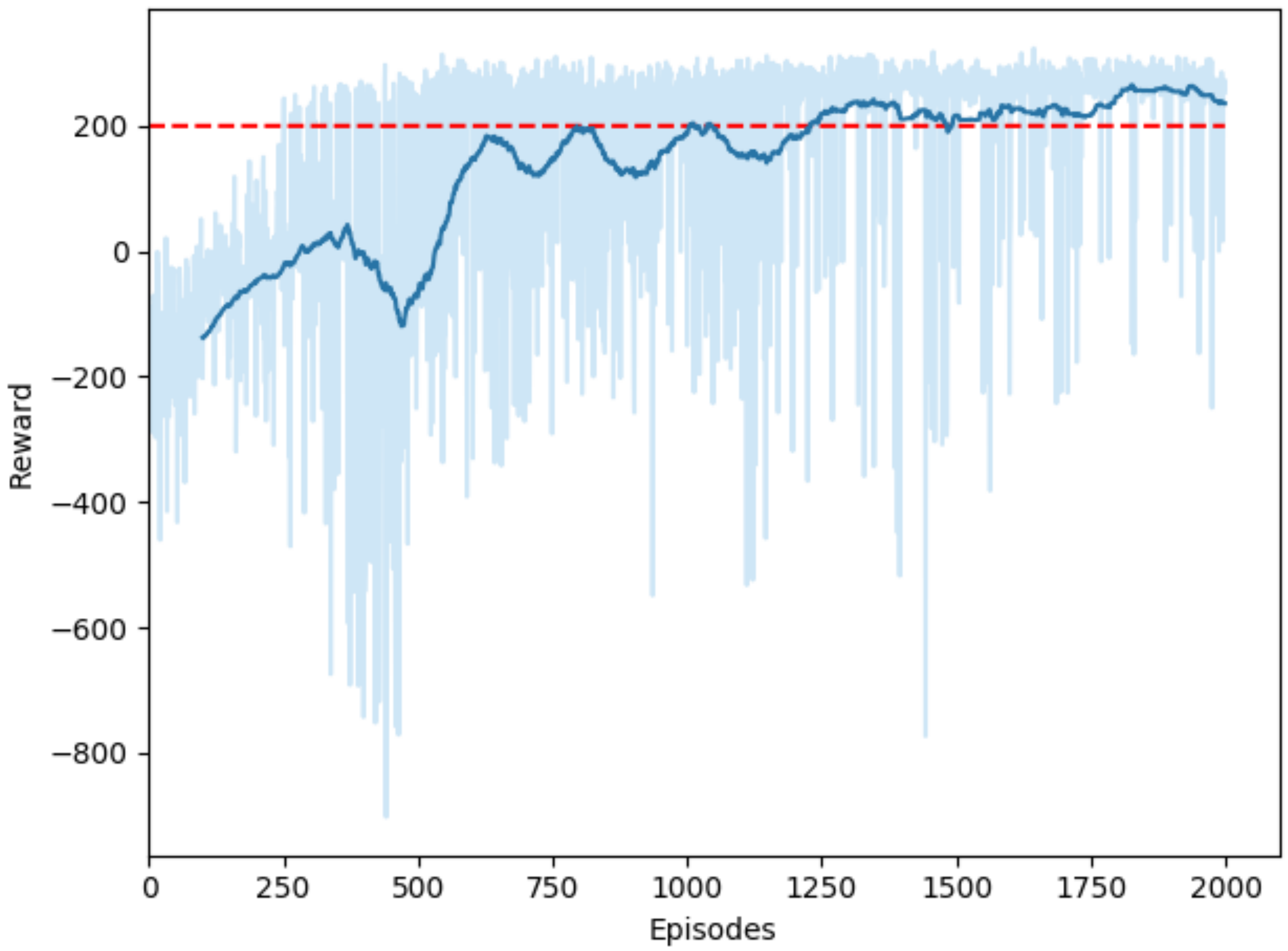


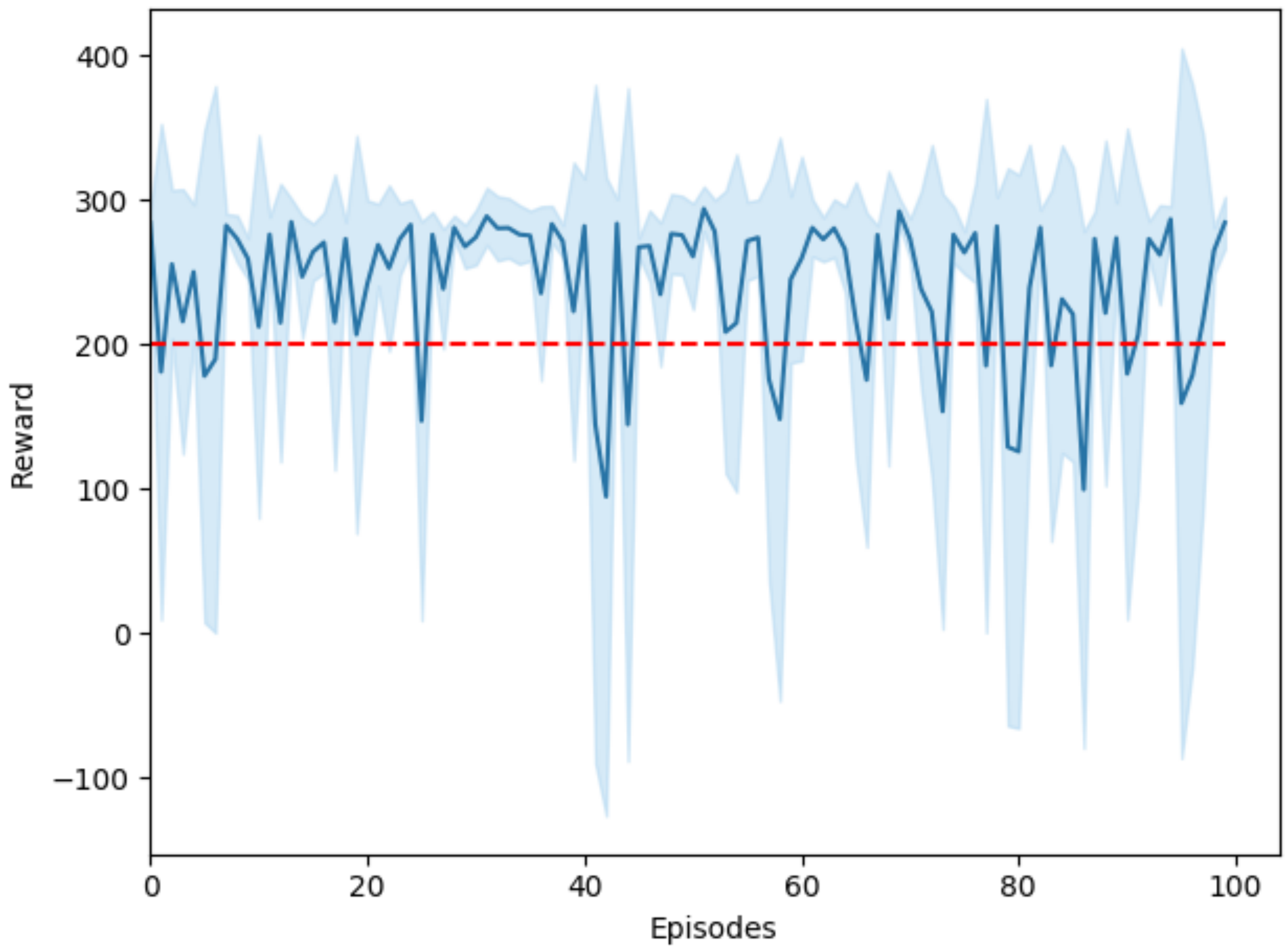
Reward
Episodes
0
25
50
75
100
125
150
175
200
0
250
500
750
1000
1250
1500
1750
2000



Reward
Episodes
120
140
160
180
200
220
0
20
40
60
80
100

LunarLander-v2





DQN: Shortcoming

- discrete action space only

- overestimation bias

    Target = reward + discount * max_a’ Q_(s’, a’)

Code

The 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 $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^-) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ with respect to the network parameters θ
    Every C steps reset $\hat{Q} = Q$
  End For
End For

Replay Memory
- store experiences
- sample n experiences

DQNNet
- network specifics

DQNAgent
- instances of policy and target networks
- select action
- update target
- update epsilon
- learn

Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." *nature* 518.7540 (2015): 529-533.

Overestimation Bias

Target = reward + discount * max_a’ Q(s’, a’)