



AIL 722

Deep Q Networks

Deep Q Learning



Pong



Breakout



Space Invaders

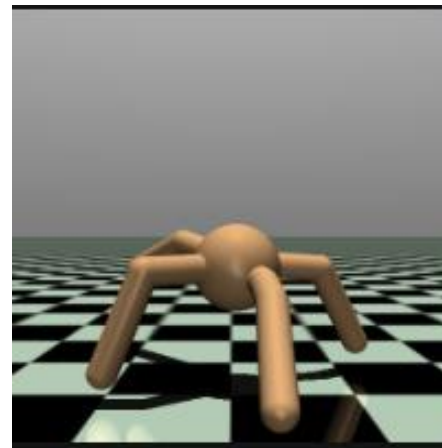


Seaquest



Beam Rider

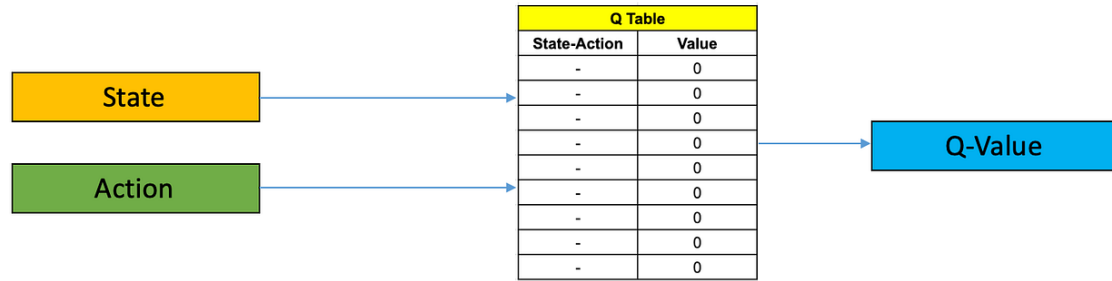
Atari2600 Games : Arcade video games, started in 1970s



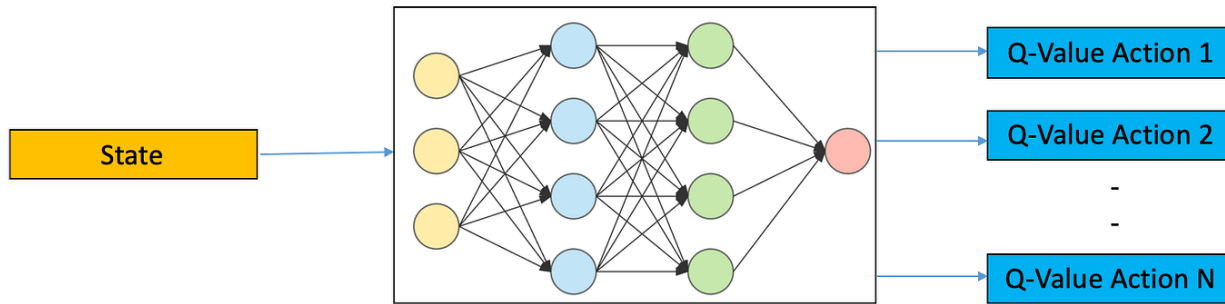
Locomotion Environments



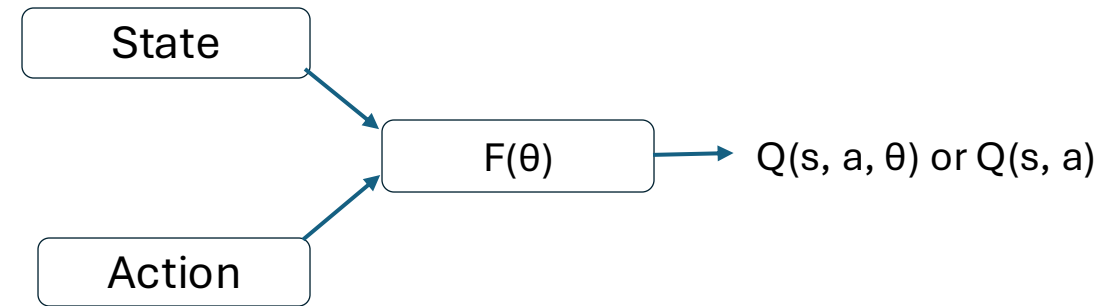
Deep Q Learning Overview



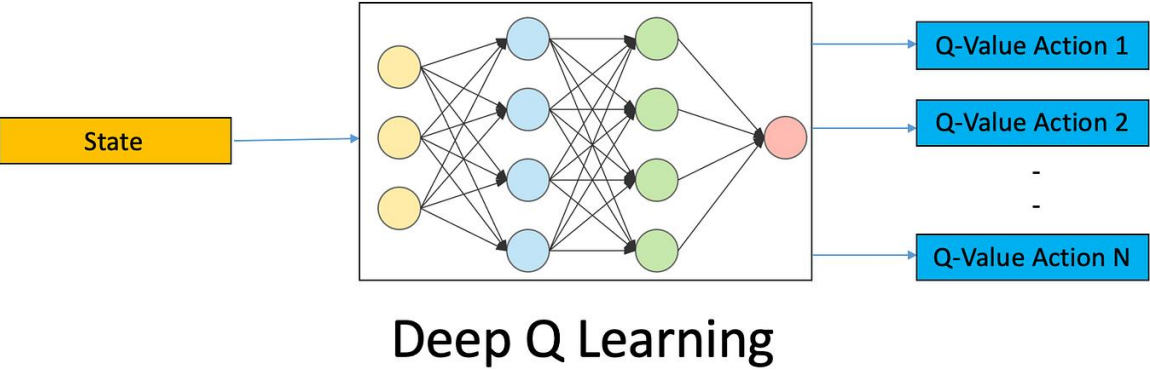
Q Learning



Deep Q Learning



Objective



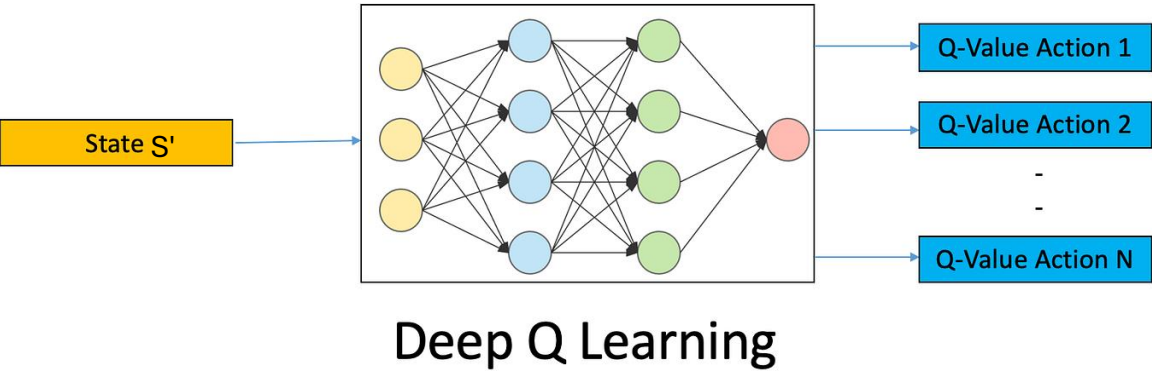
} Max Env.step(a) (s, a, s', r, done)

Target: $y_t = r_t + \gamma \max_{a'} Q(s_{t+1}, a')$

Prediction: $\hat{y}_t = Q(s_t, a_t)$

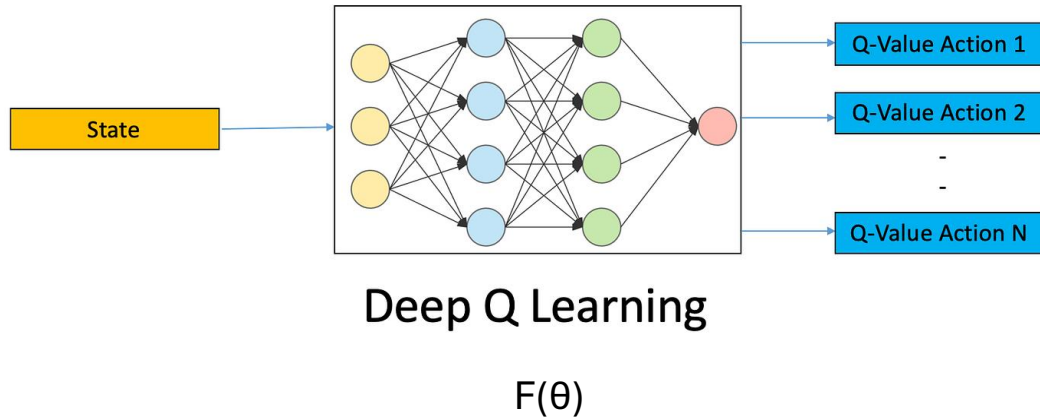
$$L(\theta) = \frac{1}{2} (y_t - \hat{y}_t)^2$$

$$L(\theta) = \frac{1}{2} \left[\left(r_t + \gamma \max_{a'} Q(s_{t+1}, a') \right) - Q(s_t, a_t) \right]^2$$

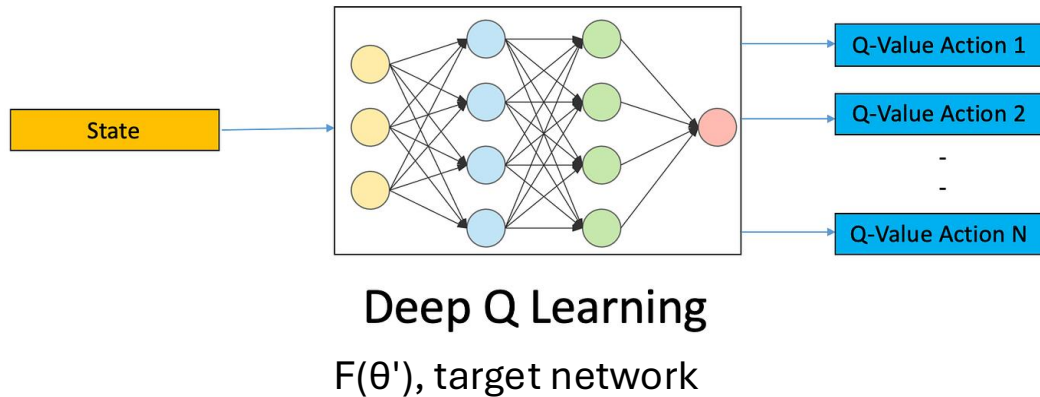


Problems with current DQN formulation ??

- **Non-stationary/Moving targets**
- Correlations between samples



1. Initially $\theta' = \theta$
2. Start training with state s
3. Compute q value predictions for s
4. Take the action with max q value
5. Transition to s' and obtain reward
6. Compute target using θ' (not θ)
7. Use MSE to perform backpropagation on the θ parameters



$$\bar{\theta} \leftarrow \tau \theta + (1 - \tau) \bar{\theta}$$

Initially $\tau = 1$ (Hard update)

Then $\tau = 0.005$ (Polyak or soft update, $\tau < 1$)

Target: $y_t = r_t + \gamma \max_{a'} Q(s_{t+1}, a'; \bar{\theta})$

Prediction: $\hat{y}_t = Q(s_t, a_t; \theta)$

$$L(\theta) = \frac{1}{2} (y_t - \hat{y}_t)^2$$

$$L(\theta) = \frac{1}{2} \left[\left(r_t + \gamma \max_{a'} Q(s_{t+1}, a'; \bar{\theta}) \right) - Q(s_t, a_t; \theta) \right]^2$$

$$\bar{\theta} \leftarrow \tau \theta + (1 - \tau) \bar{\theta}$$

Can we take the gradient of max ??

```
with torch.no_grad():
    # Compute Q-values for next state using target network
    q_next = target_network(next_state)

    # Take max over next actions
    q_next_max = q_next.max()

    # Compute target according to Bellman equation
    q_target = reward + gamma * q_next_max * [1 - done]
```

Experience Replay

What is experience replay?



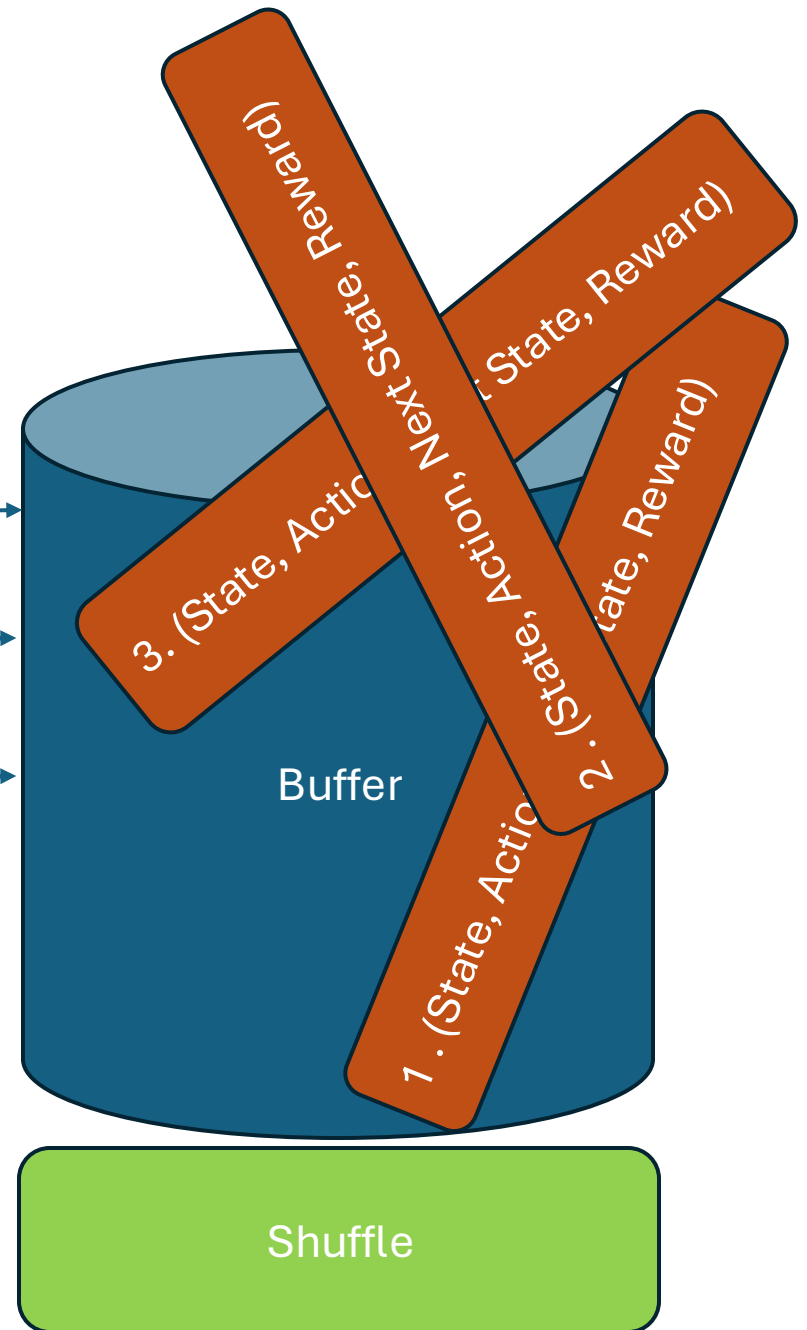
The agent simply remembers its past experiences and repeatedly uses the experiences to learn

Experience Replay

1. (State, Action, Next State, Reward)

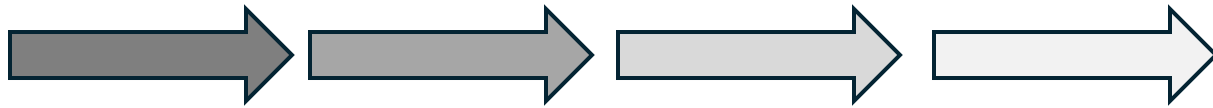
2. (State, Action, Next State, Reward)

3. (State, Action, Next State, Reward)



Why the shuffle?

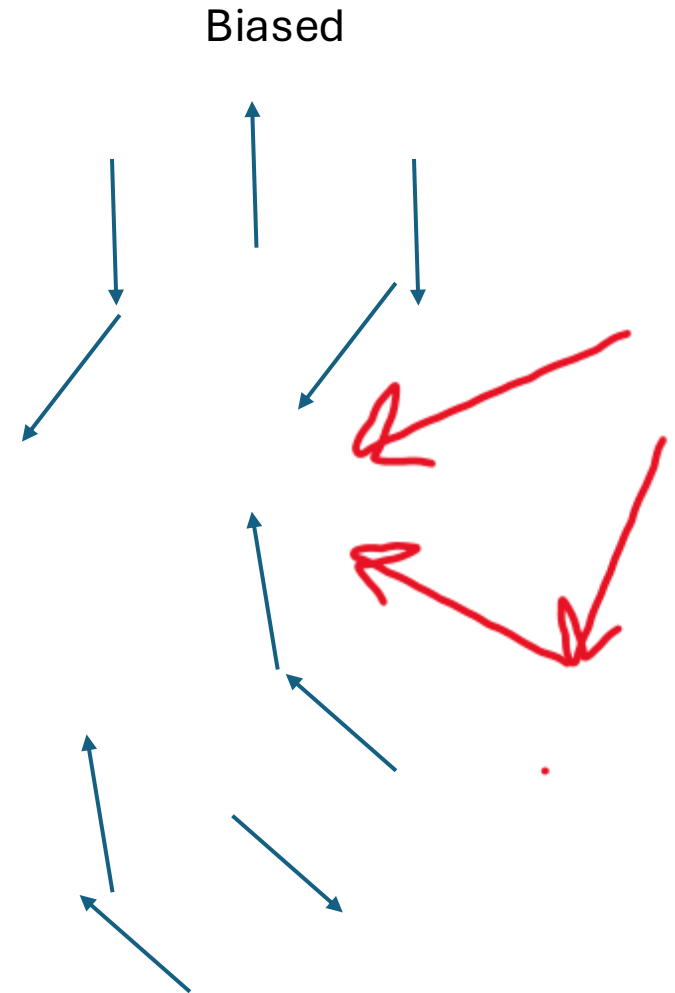
We want gradient updates to be unbiased



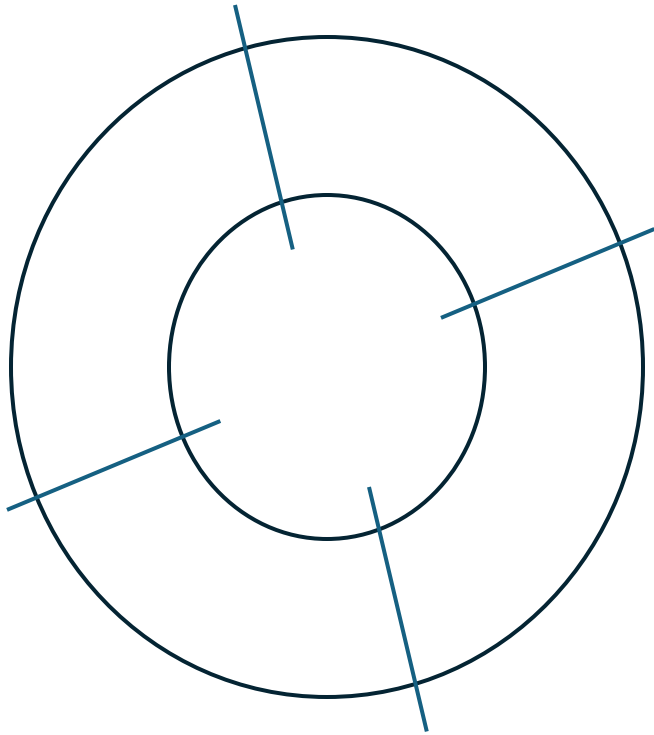
Correlated samples may worsen the variance of the gradients



We want i.i.d assumption to hold for a trajectory. The solution is to shuffle the samples.



Replay Buffer



Typically implemented as a circular buffer

Different sampling strategies may be used for drawing samples from the replay buffer

Uniform sampling

Prioritized experience replay

Distributed experience replay

Design choices for experience replay

- Capacity of the replay buffer
 - Too small capacity will cause the algorithm to forget early. Storing rare events for longer could be beneficial
 - Too large a capacity will cause the buffer to be filled with stale data
- Ratio of learning updates to experience collected (Replay ratio)
 - How many times should be use the collected data before updating the buffer?

In practice

Replay Capacity Matters More Than Expected

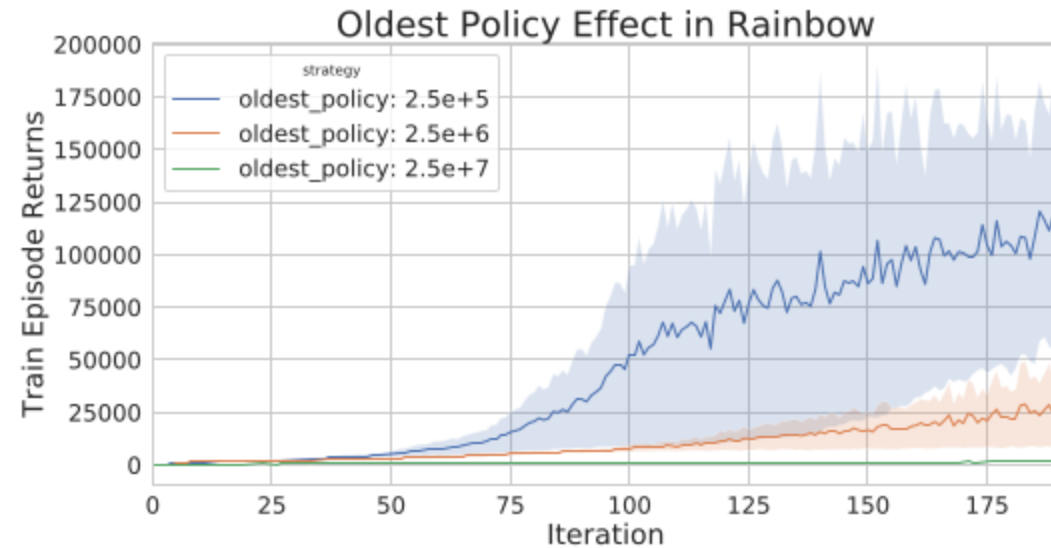
- Increasing the size of the replay buffer (replay capacity) can substantially improve performance for certain deep RL algorithms, contrary to the common practice of using a fixed, moderate buffer size.
- However, the benefit is algorithm-dependent: some agents (e.g., Rainbow) benefit greatly, while others are less affected.

Replay Ratio is a Critical Hyperparameter

- The replay ratio (number of learning updates per environment step) is a key factor in agent performance.
- Directly controlling and tuning the replay ratio can contextualize previous observations and is crucial for optimizing learning across different RL algorithms.

Reference : Revisiting Fundamentals of Experience Replay; William Fedus *,Prajit Ramachandran,Rishabh Agarwal, Yoshua Bengio, Hugo Larochelle, Mark Rowland,Will Dabney, ICML2020

Empirical results for Experience replay ablations



(b) SEAQUEST

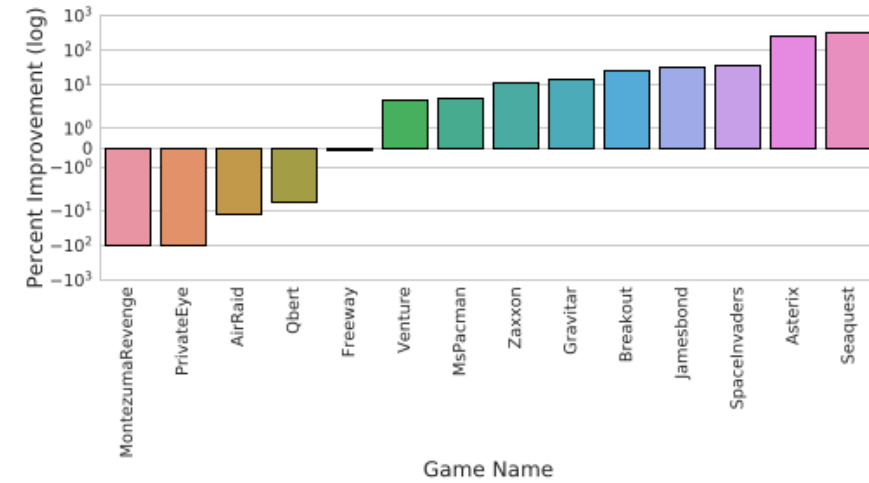
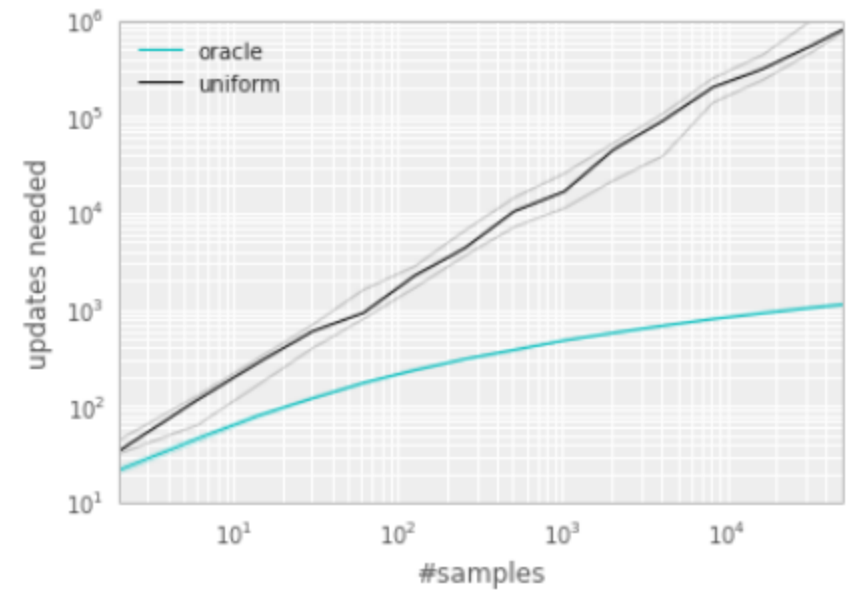
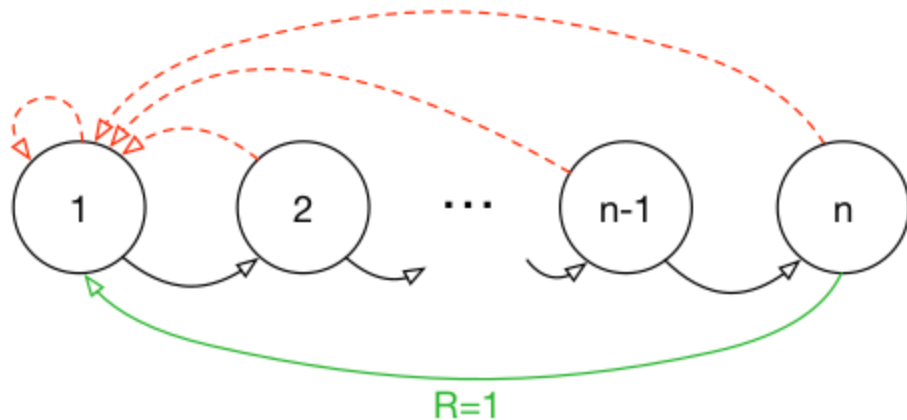


Figure 4. Sparse-reward games benefit from data generated by older policies. Median relative improvement of a Rainbow agent with a 10M replay capacity and 250k oldest policy compared to one with 2.5M oldest policy. Decreasing the age of the oldest policy improves performance on most games. However, performance drops significantly on the two hard exploration games, which bucks the trend that data from newer policies is better.

Reference and Image source : Revisiting Fundamentals of Experience Replay; William Fedus *, Prajit Ramachandran, Rishabh Agarwal, Yoshua Bengio, Hugo Larochelle, Mark Rowland, Will Dabney, ICML2020

Prioritized Experience Replay

Uniform sampling on cliff walk is in-efficient. One wrong action can lead to falling off the cliff. Getting to the goal is rare.



Give high priority to transitions near the goal. This makes the signal near the goal stronger while visiting these states is rare.

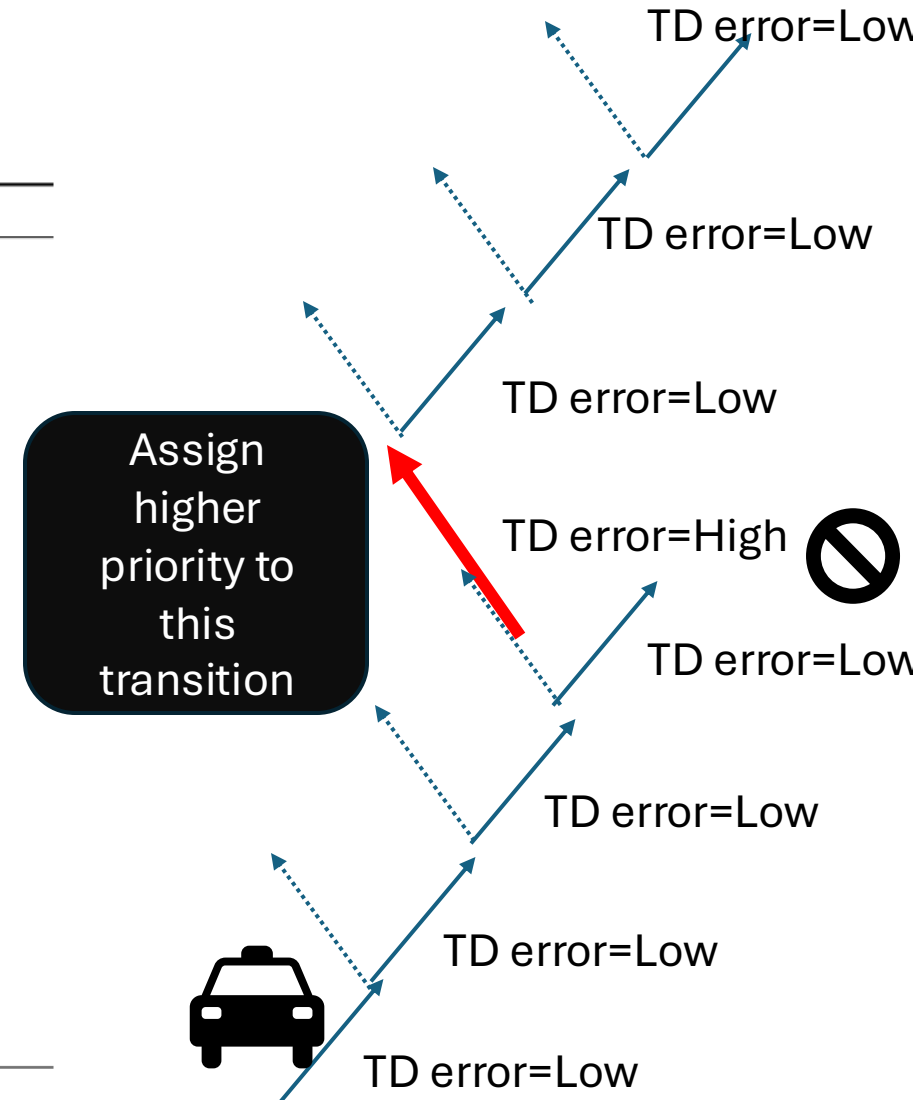
Image Source : PRIORITIZED EXPERIENCE REPLAY

Tom Schaul, John Quan, Ioannis Antonoglou and David Silver, ICLR 2016

Prioritized experience replay

Algorithm 1 Double DQN with proportional prioritization

```
1: Input: minibatch  $k$ , step-size  $\eta$ , replay period  $K$  and size  $N$ , exponents  $\alpha$  and  $\beta$ , budget  $T$ .
2: Initialize replay memory  $\mathcal{H} = \emptyset$ ,  $\Delta = 0$ ,  $p_1 = 1$ 
3: Observe  $S_0$  and choose  $A_0 \sim \pi_\theta(S_0)$ 
4: for  $t = 1$  to  $T$  do
5:   Observe  $S_t, R_t, \gamma_t$ 
6:   Store transition  $(S_{t-1}, A_{t-1}, R_t, \gamma_t, S_t)$  in  $\mathcal{H}$  with maximal priority  $p_t = \max_{i < t} p_i$ 
7:   if  $t \equiv 0 \pmod K$  then
8:     for  $j = 1$  to  $k$  do
9:       Sample transition  $j \sim P(j) = p_j^\alpha / \sum_i p_i^\alpha$ 
10:      Compute importance-sampling weight  $w_j = (N \cdot P(j))^{-\beta} / \max_i w_i$ 
11:      Compute TD-error  $\delta_j = R_j + \gamma_j Q_{\text{target}}(S_j, \arg \max_a Q(S_j, a)) - Q(S_{j-1}, A_{j-1})$ 
12:      Update transition priority  $p_j \leftarrow |\delta_j|$ 
13:      Accumulate weight-change  $\Delta \leftarrow \Delta + w_j \cdot \delta_j \cdot \nabla_\theta Q(S_{j-1}, A_{j-1})$ 
14:    end for
15:    Update weights  $\theta \leftarrow \theta + \eta \cdot \Delta$ , reset  $\Delta = 0$ 
16:    From time to time copy weights into target network  $\theta_{\text{target}} \leftarrow \theta$ 
17:  end if
18:  Choose action  $A_t \sim \pi_\theta(S_t)$ 
19: end for
```



Sample transitions with higher temporal difference error with more priority

Sampling in Prioritized Experience Replay

$$P_i = |\text{TD error}_i| + \epsilon$$

Epsilon makes sure that if TD error is zero, priority is not zero

$$P(i) = \frac{P_i^\alpha}{\sum_j P_j^\alpha}$$

Alpha controls the priority given to TD error. Alpha is zero we do uniform sampling

- Prioritized experience replay introduces a sampling bias
- In practice, importance sampling weights are used to adjust for this bias.

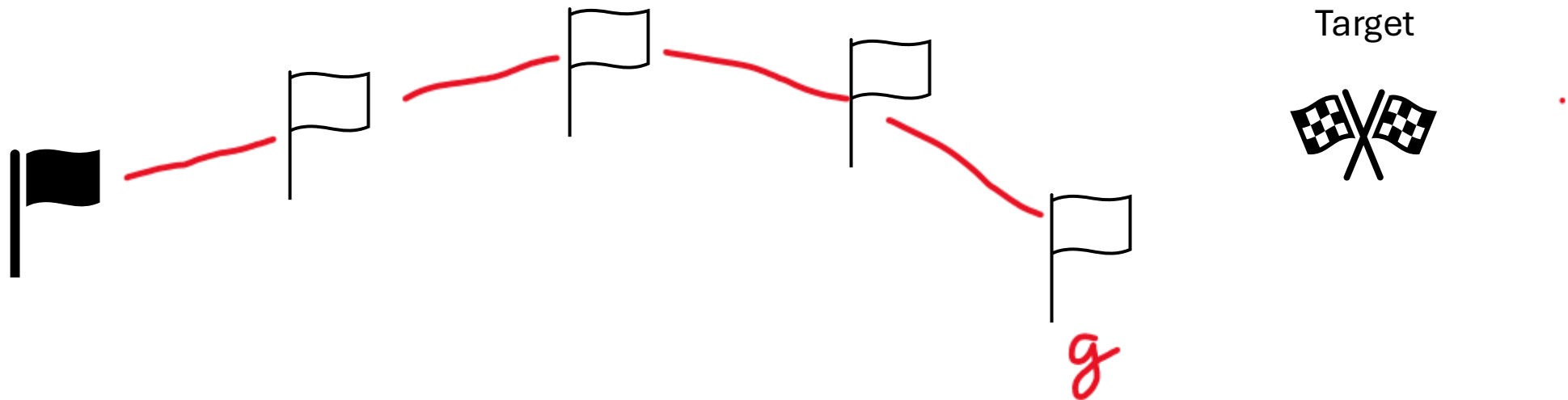
$$w_i = \left(\frac{1}{N \cdot P(i)} \right)^\beta$$

This is done because we are computing an expectation of cumulative rewards given state and action, and it would be incorrect if the sampling is biased

The weight is applied during loss computation

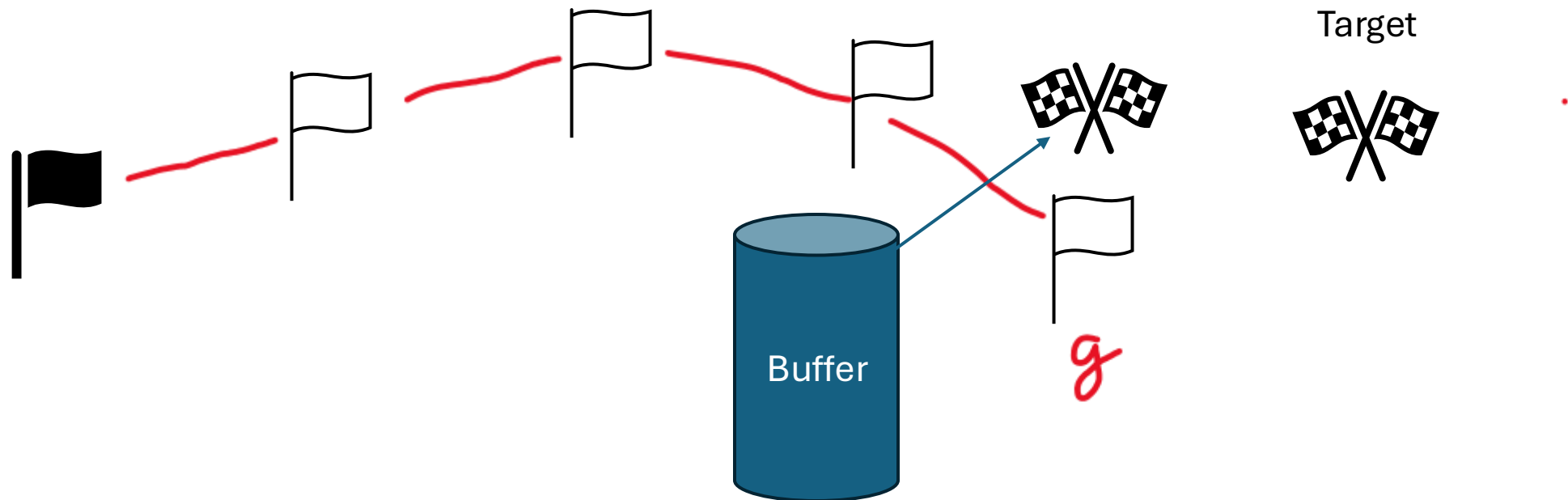
Hindsight experience replay

- Sparse-reward domains: agent rarely achieves goal, little learning.
- HER relabels failed attempts by pretending a new goal was met.



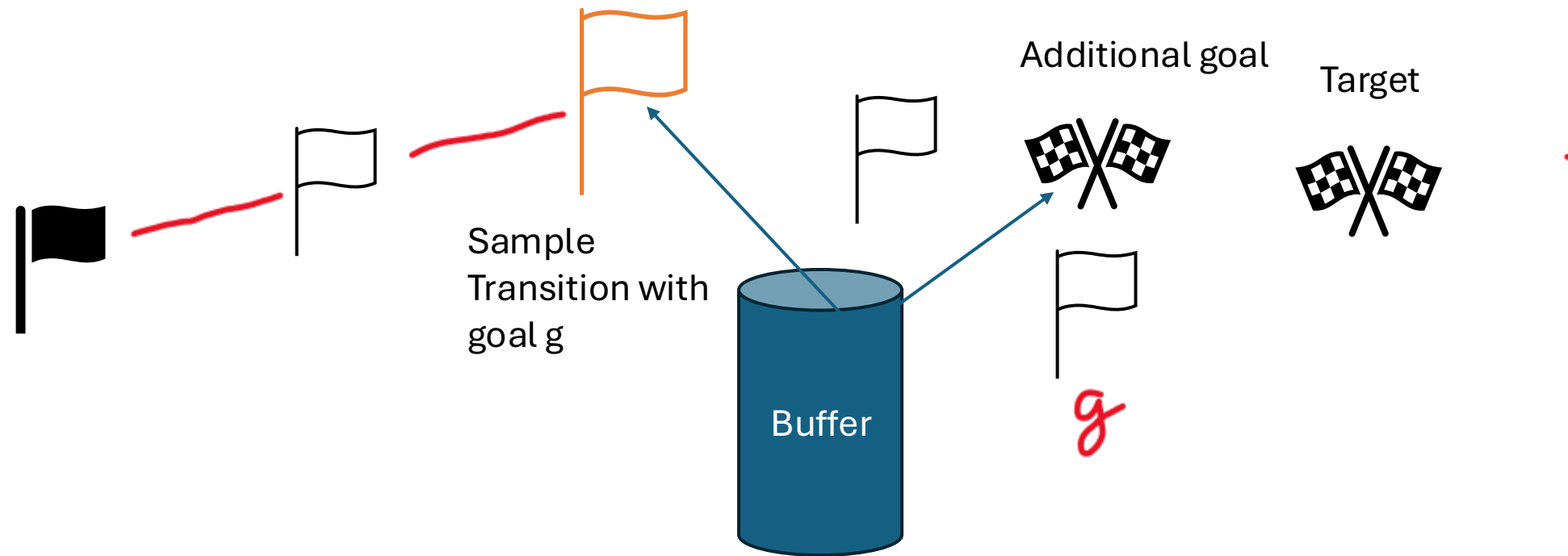
Hindsight Experience Replay

- Store achieved goals from each episode.
- For training, relabel past transitions as if the achieved goal was
- intended.



Hindsight Experience Replay

New goals are set from transitions that have been seen. A new reward is generated when the agent learns how to reach this new goal from sampled transitions.



Hindsight Experience Replay

- Store (s, a, r, s', g)
- Sample new goal g' achieved later in the episode.
- Compute new reward $r(s, a, g')$, store relabeled experience.
- Learn from both original and relabeled data.
- Note : Sampling technique for goals is important. What if we pick a goal to go back?

$$r : \mathcal{S} \times \mathcal{A} \times \mathcal{G} \rightarrow \mathbb{R}.$$

$$Q^*(s, a, g) = \mathbb{E}_{s'}[r(s, a, g) + \gamma \max_{a'} Q^*(s', a', g)]$$

Hindsight Experience Replay

Method	Purpose	Key Equation	Use
ER	Decorrelation	Q -update	Off-policy RL
PER	Fast learning	$P_i = \delta + \epsilon$	Hard exploration
HER	Sparse goals	$Q^*(s, a, g)$	Robotics/Navigation

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 $a_t = \arg \max_a Q(\phi(s_t), a; \theta)$

 Execute action a_t in the 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 experience $(\phi_t, a_t, r_t, \phi_{t+1})$ in D

 Sample random minibatch of experiences $(\phi_j, a_j, r_j, \phi_{j+1})$ 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 weights θ

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

end for

end for

Ablations on Atari

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

DQN agents were trained for 10 million frames using standard hyperparameters for all possible combinations of turning replay on or off, using or not using a separate target Q-network, and three different learning rates. Each agent was evaluated every 250,000 training frames for 135,000 validation frames and the highest average episode score is reported. Note that these evaluation episodes were not truncated at 5 min leading to higher scores on Enduro than the ones reported in Extended Data Table 2. Note also that the number of training frames was shorter (10 million frames) as compared to the main results presented in Extended Data Table 2 (50 million frames).

