

# Extensions to Q-Learning

---

Reinforcement Learning  
School of Data Science  
University of Virginia

Last updated: March 3, 2024

# Agenda

- > Double Q-Learning
- > Prioritized Experience Replay
- > Dueling Deep Q-Network

# Double Q-Learning

# Main Idea

DQN requires target estimates of this form:

$$Y_t^Q \equiv R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \theta_t)$$

The max() operation is used to estimate value

There may be noise in the system

Tends to produce a bias: overestimating value of Q

*Paper: Deep Reinforcement Learning with Double Q-learning*

*Hado van Hasselt, Arthur Guez, David Silver. Google DeepMind*

# Main Idea, contd.

## Double DQN

The idea of Double Q-learning is to reduce overestimations by decomposing the max operation in the target into action selection and action evaluation. Although not fully decoupled, the target network in the DQN architecture provides a natural candidate for the second value function, without having to introduce additional networks. We therefore propose to evaluate the greedy policy according to the online network, but using the target network to estimate its value. In reference to both Double Q-learning and DQN, we refer to the resulting algorithm as Double DQN. Its update is the same as for DQN, but replacing the target  $Y_t^{\text{DQN}}$  with

$$Y_t^{\text{DoubleQ}} \equiv R_{t+1} + \gamma Q(S_{t+1}, \underset{a}{\operatorname{argmax}} Q(S_{t+1}, a; \theta_t); \theta'_t)$$

Decompose max() operation into action selection, action evaluation

Notice there are two Qs

- One determines greedy policy using online network  $\theta_t$
- Another fairly evaluates the policy using the target network  $\theta'_t$

# Double Q-Learning Algorithm

---

**Algorithm 1 : Double Q-learning (Hasselt et al., 2015)**

---

Initialize primary network  $Q_\theta$ , target network  $Q_{\theta'}$ , replay buffer  $\mathcal{D}$ ,  $\tau \ll 1$

**for** each iteration **do**

**for** each environment step **do**

        Observe state  $s_t$  and select  $a_t \sim \pi(a_t, s_t)$

        Execute  $a_t$  and observe next state  $s_{t+1}$  and reward  $r_t = R(s_t, a_t)$

        Store  $(s_t, a_t, r_t, s_{t+1})$  in replay buffer  $\mathcal{D}$

**for** each update step **do**

        sample  $e_t = (s_t, a_t, r_t, s_{t+1}) \sim \mathcal{D}$

        Compute target Q value:

$$Q^*(s_t, a_t) \approx r_t + \gamma Q_\theta(s_{t+1}, \operatorname{argmax}_{a'} Q_{\theta'}(s_{t+1}, a'))$$

        Perform gradient descent step on  $(Q^*(s_t, a_t) - Q_\theta(s_t, a_t))^2$

        Update target network parameters:

$$\theta' \leftarrow \tau * \theta + (1 - \tau) * \theta'$$

---

# Bias Estimates

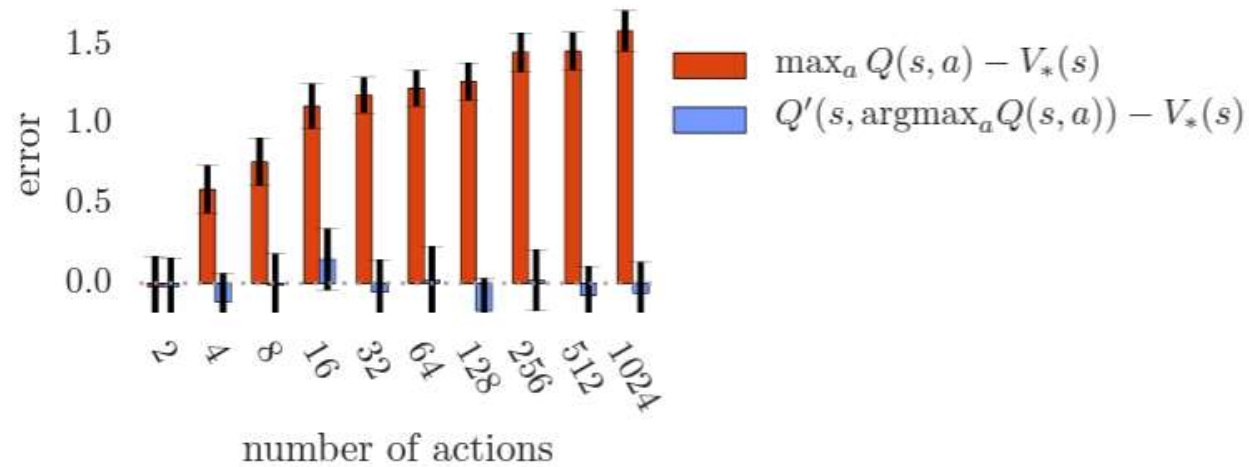
**Theorem 1.** *Consider a state  $s$  in which all the true optimal action values are equal at  $Q_*(s, a) = V_*(s)$  for some  $V_*(s)$ . Let  $Q_t$  be arbitrary value estimates that are on the whole unbiased in the sense that  $\sum_a (Q_t(s, a) - V_*(s)) = 0$ , but that are not all correct, such that  $\frac{1}{m} \sum_a (Q_t(s, a) - V_*(s))^2 = C$  for some  $C > 0$ , where  $m \geq 2$  is the number of actions in  $s$ . Under these conditions,  $\max_a Q_t(s, a) \geq V_*(s) + \sqrt{\frac{C}{m-1}}$ . This lower bound is tight. Under the same conditions, the lower bound on the absolute error of the Double Q-learning estimate is zero. (Proof in appendix.)*

=> The  $\max()$  operator induces upward bias that shrinks with increasing #actions, but does not vanish

# Bias Estimates, contd.

Overoptimism increases with number of actions (red bars)

However, for Double Q-Learning, bias remains small



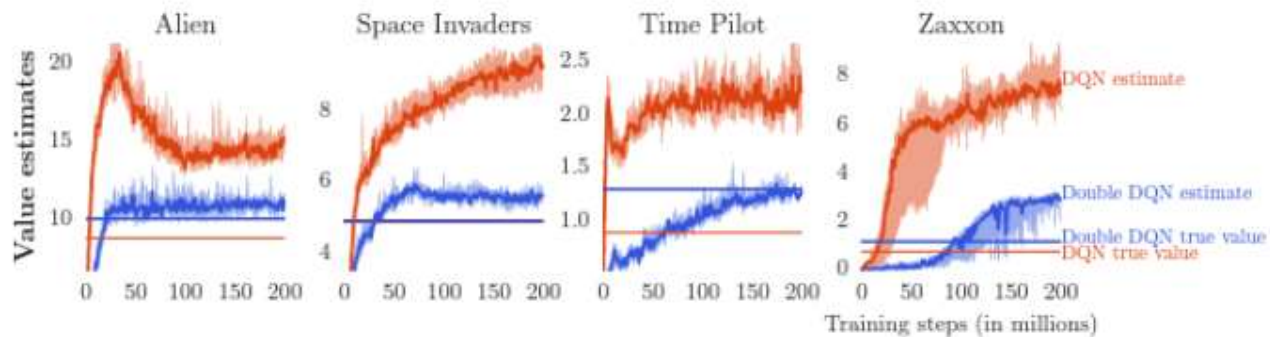


# Bias Estimates - Atari

Horizontal lines are actual discounted value of best learned policy (unbiased)

Curves are estimates, which exhibit bias

Double DQN best policy (blue horiz. line) outperforms DQN best policy (red horiz. line)



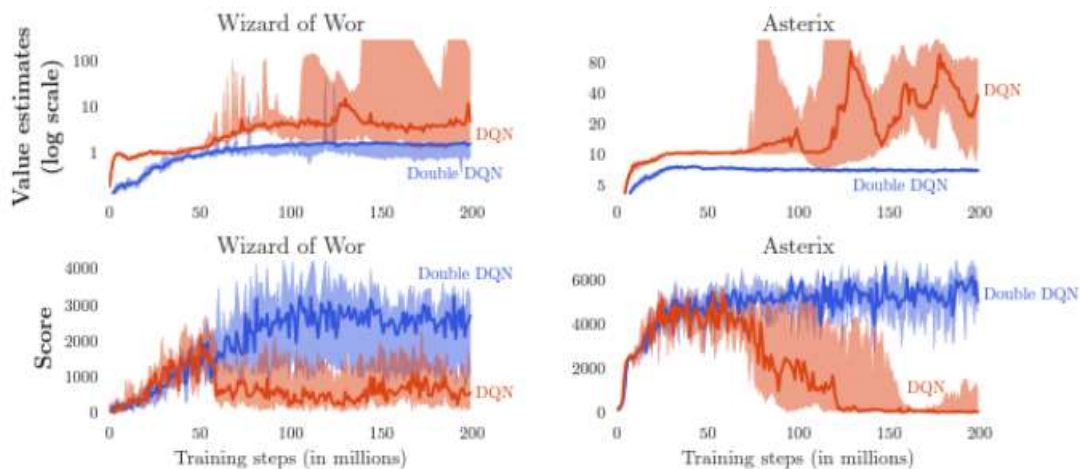
# Bias Estimates – Atari, Extreme Cases

In these cases, overoptimism in DQN is extreme (red curve)

Where red separates from blue, bias increases

This has detrimental effect on score (Double Q-Learning outperforms)

Double Q-Learning is also more stable



# Findings

The bias resulting from  $\max()$  in DQN can be large

Double DQN can be more stable and reliable than DQN

Double DQN can find better policies than DQN

# Prioritized Experience Replay

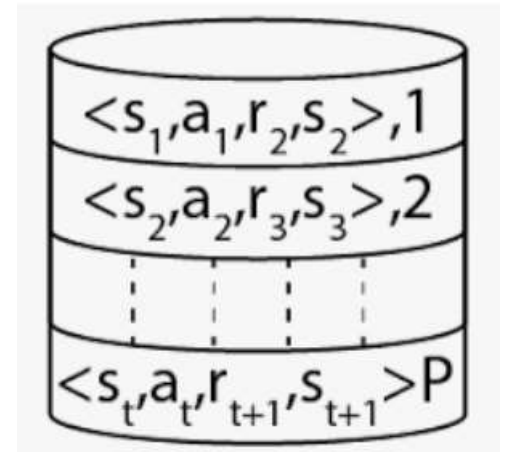
# Buffer

DQN uses a buffer to store, retrieve, and replay past transitions ( $s_t, a_t, r_{t+1}, s_{t+1}$ )

This makes DQN more efficient than Q-Learning

The transitions are sampled uniformly at random

Prioritized Experience Replay (PER)  
tries to be more strategic about sampling



*Paper: Prioritized Experience Replay. Schaul et. al.*

# Priority

PER uses absolute TD error  $|\delta_i|$  as the priority

Could select transition with max abs TD error, but greedy approach may backfire

Instead, paper uses stochastic prioritization with two methods:

1) Proportional

2) Rank-based

# Proportional Priority

Define proportional priority where  $p_i = |\delta_i| + \epsilon$

Epsilon prevents transitions from not being visited when error is zero

Define probability of sampling transition

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$$

Alpha controls amount of prioritization; alpha = 0 is uniform case.

# Rank-Based Priority

Rank is used for robustness

Define probability of sampling transition

$$p_i = \frac{1}{\text{rank}(i)}$$

Ranks are based on absolute TD error



# Importance Sampling

PER induces bias by changing probability of transition selection

Can adjust for bias with importance sampling

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

Case where beta = 1 fully compensates for non-uniform probabilities

For Q-learning updates, use  $|\delta_i|$  weighted by  $w_i$

# Algorithm

---

**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
```

---

# Results: Summary of Normalized Scores

	DQN		Double DQN (tuned)		
	baseline	rank-based	baseline	rank-based	proportional
<b>Median</b>	48%	106%	111%	113%	128%
<b>Mean</b>	122%	355%	418%	454%	551%
<b>&gt; baseline</b>	–	41	–	38	42
<b>&gt; human</b>	15	25	30	33	33
<b># games</b>	49	49	57	57	57

DQN with rank-based priority results in higher scores for 41 out of 49 games

Double DQN is similarly helped by PER

# Dueling Deep Q-Network

# Dueling Networks

Deep RL before this paper used conventional architectures (CNN, LSTM)

Focus here is new architecture better suited to model-free RL

*Dueling architecture* separates state values and action advantages

*Paper: Dueling Network Architectures for Deep Reinforcement Learning*  
*Wang et. al.*

# Value and Advantage Function Definitions

Value functions:

$$Q^{\pi}(s, a) = \mathbb{E}[R_t | s_t = s, a_t = a, \pi]$$

$$V^{\pi}(s) = \mathbb{E}_{a \sim \pi(s)} [Q^{\pi}(s, a)].$$

The advantage function isolates effect of action taken

$$A^{\pi}(s, a) = Q^{\pi}(s, a) - V^{\pi}(s)$$

R denotes the return or gain

$V(s)$  measures how good it is to be in state  $s$

$A(s,a)$  measures relative importance of each action

Sometimes, the action doesn't matter

# Dueling Architecture

Separate streams for value and advantage functions

Common convolution feature

Top figure is Q-network

Bottom figure is dueling Q-network

Value and advantage streams are combined

For each network, outputs are  $Q(s,a)$

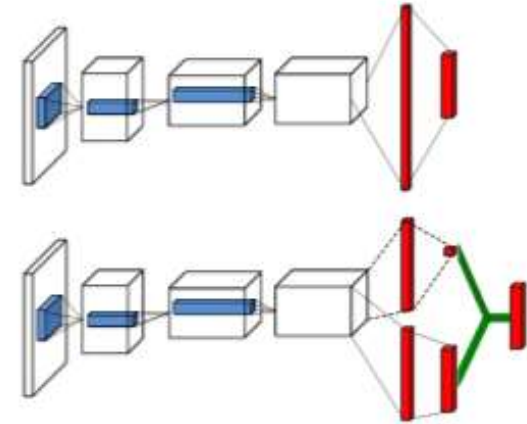


Figure 1. A popular single stream  $Q$ -network (**top**) and the dueling  $Q$ -network (**bottom**). The dueling network has two streams to separately estimate (scalar) state-value and the advantages for each action; the green output module implements equation (9) to combine them. Both networks output  $Q$ -values for each action.

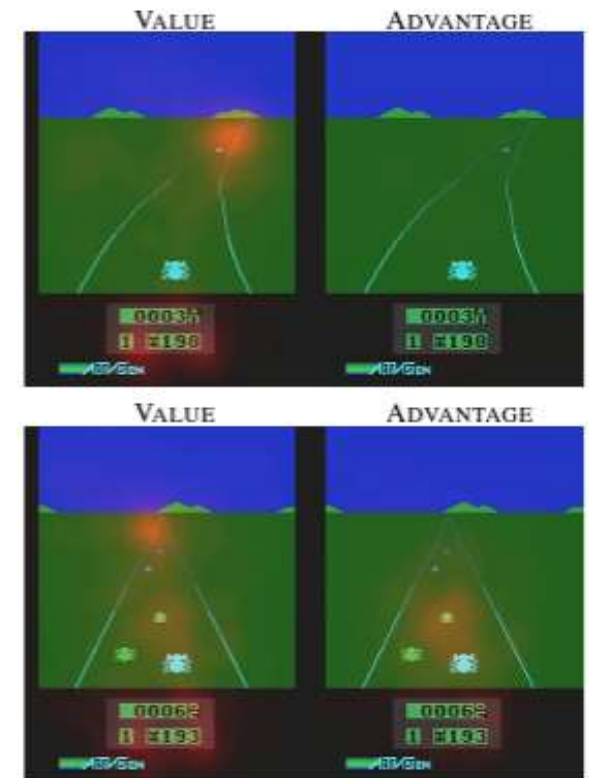
# Value and Advantage Function Saliency

Figure shows saliency map for two time steps

Value stream pays attention to horizon and score

Advantage stream: when no cars are on road, action doesn't matter (no attention paid)

When cars are on road, advantage stream pays attention to car in front





# Identifiability

Given  $Q = V + A$ , we cannot recover  $V$  and  $A$  uniquely  
(adding any  $c$  to  $V$ , and subtracting  $c$  from  $A$ , leaves  $Q$  unchanged)

Add a constraint: subtract the average advantage from  $A$ :

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

For actions with above average advantage, the second term in (\*) will be positive

For action with greatest advantage, (\*) term will be largest across all actions

# Review of Dueling Architecture Code

We can see an implementation of Dueling Q-Network here:

Paper: *Reinforcement Learning for optimal sepsis treatment policies* (2017)

Authors: Raghu, Komorowski, Ahmed, Celi, Szolovits, Ghassemi

GitHub repo:

[https://github.com/aniruddhraghu/sepsisrl/blob/master/continuous/q\\_network.ipynb](https://github.com/aniruddhraghu/sepsisrl/blob/master/continuous/q_network.ipynb)

See section:

# advantage and value streams

# Implementation

Our two streams (V, A) for Q are part of the model architecture

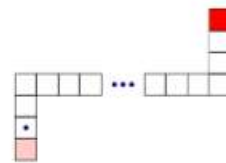
Training step runs the same as a standard Q-network: backpropagation

# Evaluation

Sets up simple environment called *corridor*.

Redness of state signifies reward

CORRIDOR ENVIRONMENT



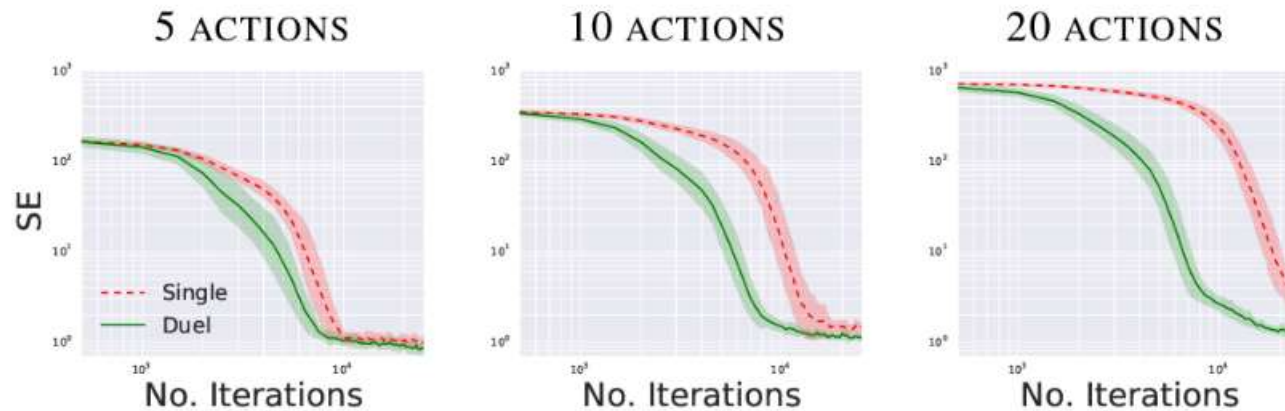
Can move up, down, left, right, or no move.

$\epsilon$ -greedy policy, measure performance by squared error (SE) vs. true action values

Single and Duel networks use MLP with three layers

For duel network, after first hidden layer, network branches off to two streams

# Evaluation, contd.



Dueling network converges faster than Single

More pronounced as  $|A|$  increases

As  $|A|$  increases, SE is lower for Dueling network. It performs better.