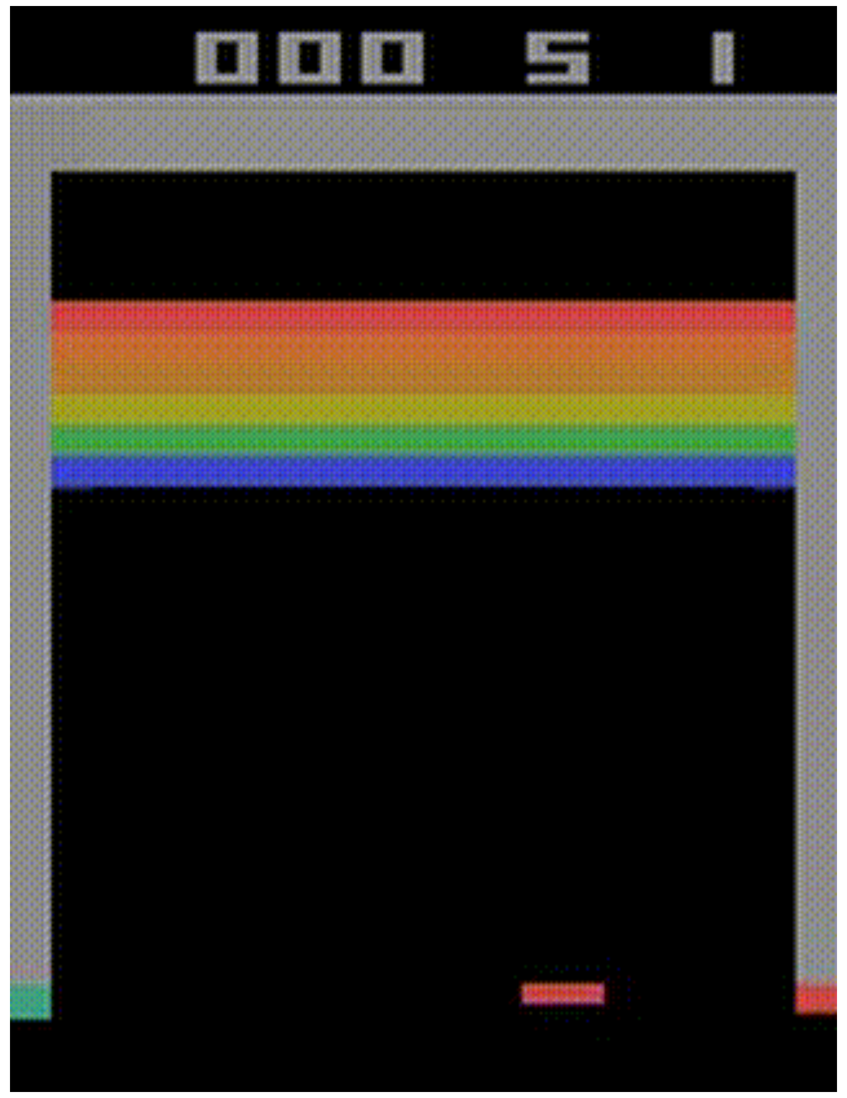
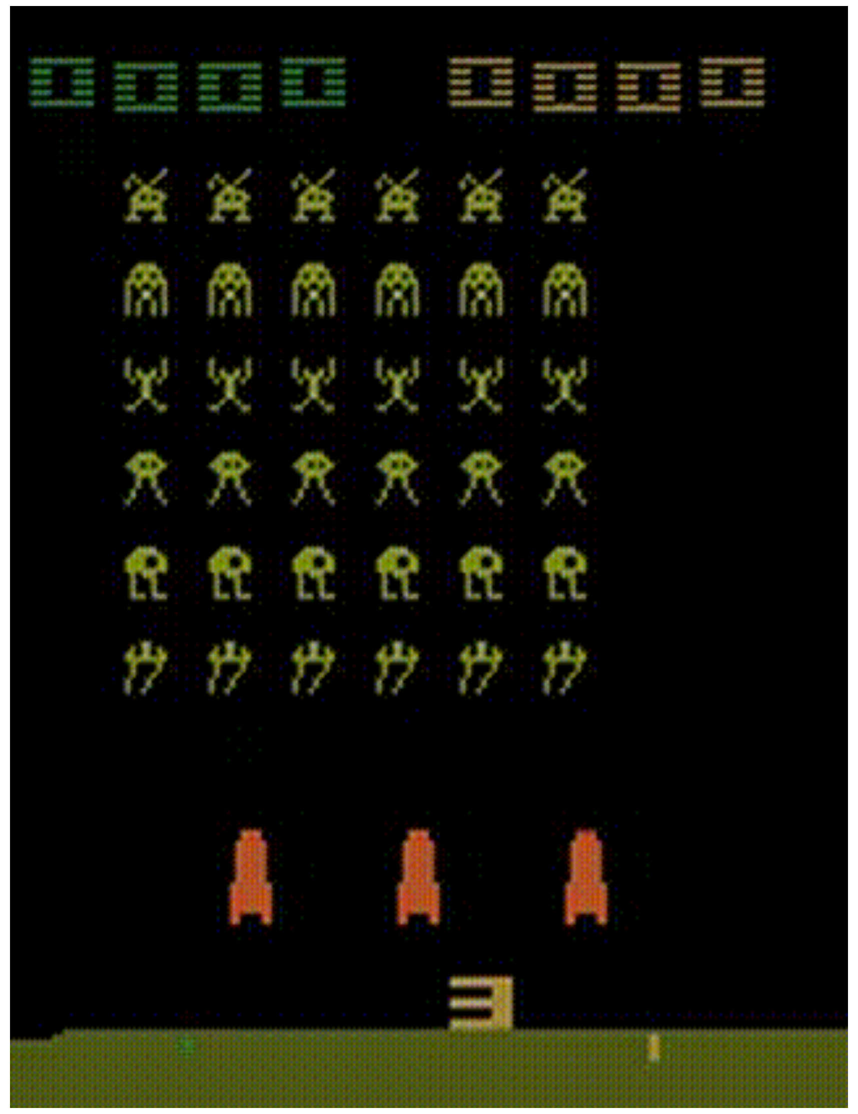
Revisiting Overestimation in Value-based Deep Reinforcement Learning

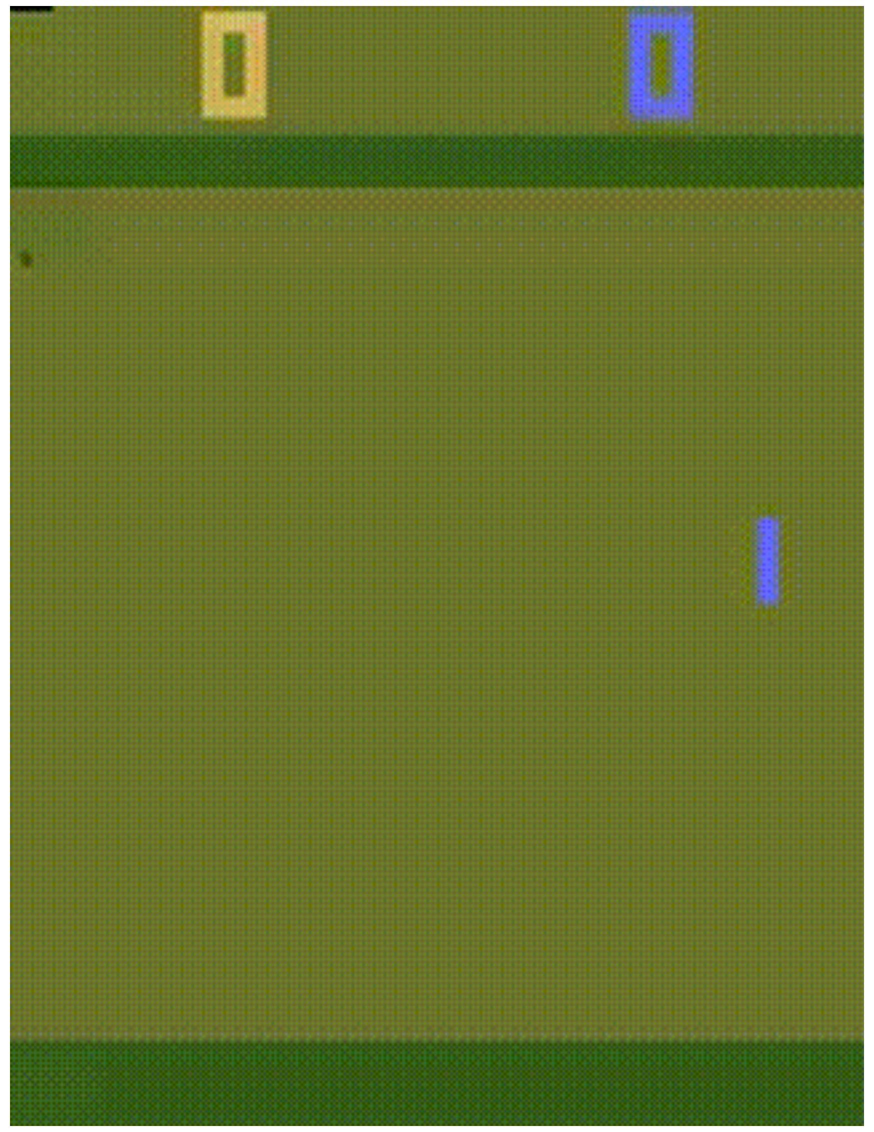
Prabhat Nagarajan
with Andy Patterson, Martha White, and Marlos C. Machado

Deep RL History: Deepmind and DQN

Dec 2013







Jan 2014

Google To Acquire Artificial Intelligence Company DeepMind

Google Acquires Artificial-Intelligence Company DeepMind

Google buys UK artificial intelligence start-up DeepMind

Feb 2015

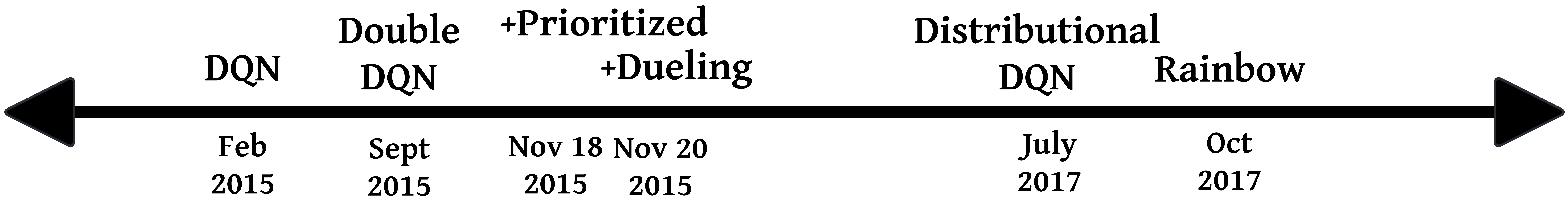
LETTER

doi:10.1038/nature14236

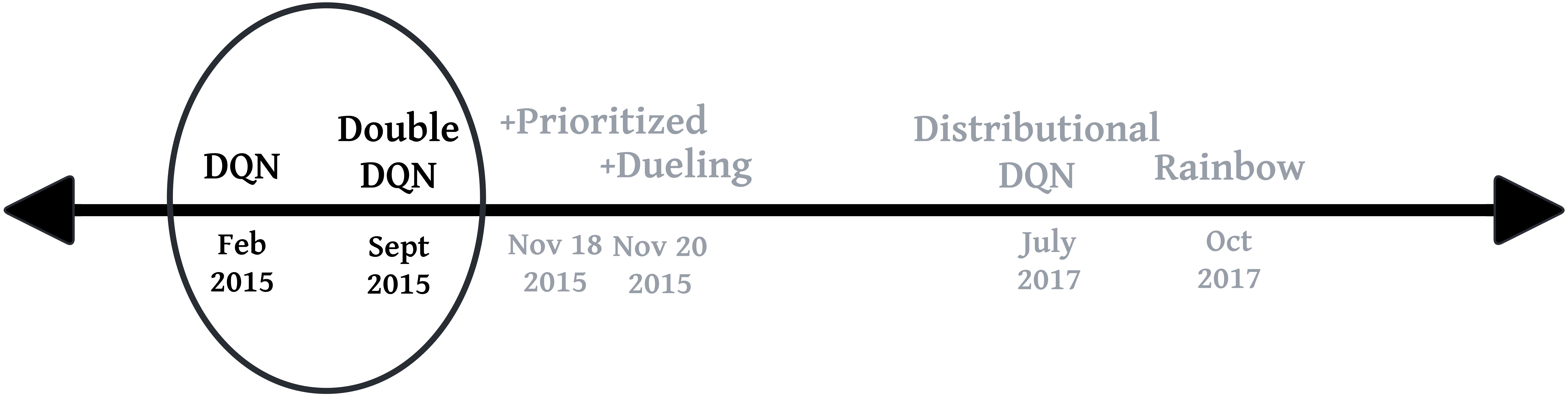
Human-level control through deep reinforcement learning

Volodymyr Mnih[1]*, Koray Kavukcuoglu[1]*, David Silver[1]*, Andrei A. Rusu[1], Joel Veness[1], Marc G. Bellemare[1], Alex Graves[1], Martin Riedmiller[1], Andreas K. Fidjeland[1], Georg Ostrovski[1], Stig Petersen[1], Charles Beattie[1], Amir Sadik[1], Ioannis Antonoglou[1], Helen King[1], Dharshan Kumaran[1], Daan Wierstra[1], Shane Legg[1] & Demis Hassabis[1]

Some Deep RL History



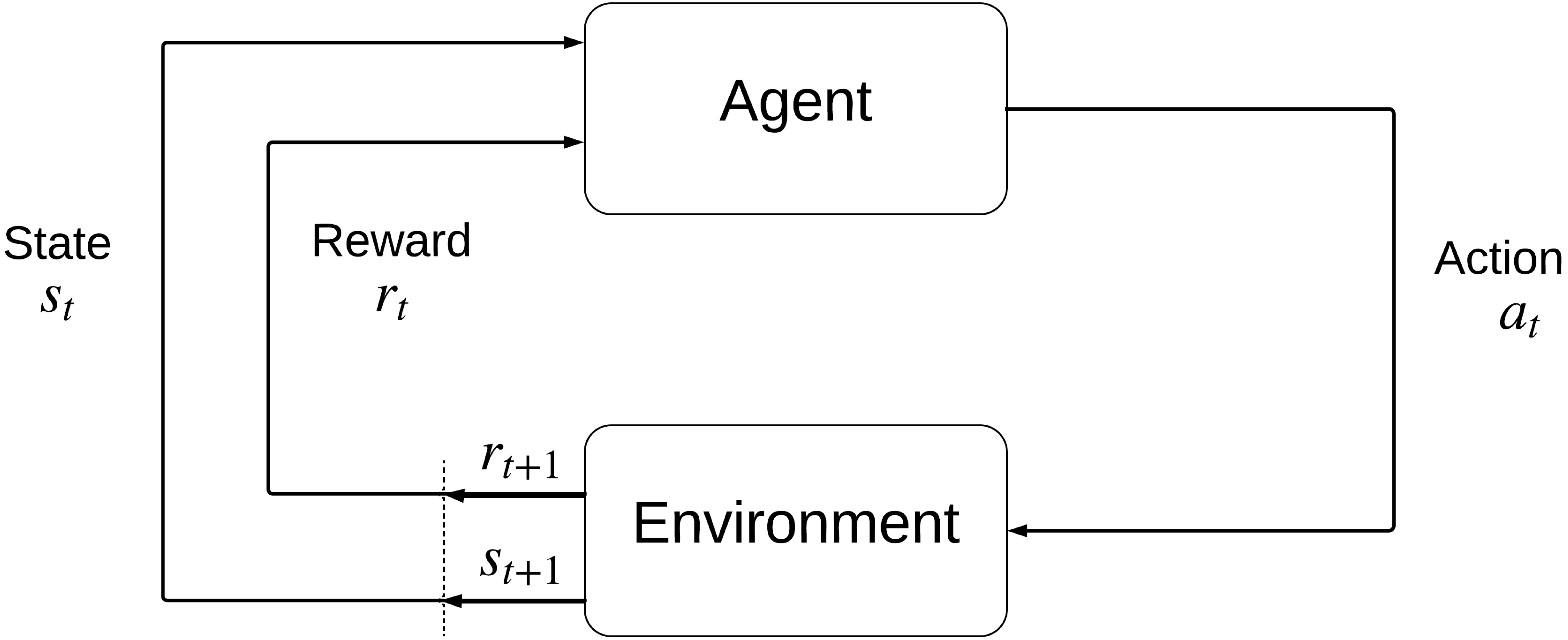
Some Deep RL History



Outline

1. Brief overview of reinforcement learning
2. RL algorithms and overestimation
3. Deep RL
4. Experiments

Reinforcement Learning



Modeled after diagram from Sutton & Barto (2018)

Reinforcement Learning

- Learn policy $\pi(a \mid s)$ that yields maximum *expected discounted return:*

$$\mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t R_{t+1} \mid S_0 = s\right]$$, *where discount factor* $\gamma \in [0,1)$.

- Optimal policy is denoted π^*, a policy that maximizes expected discounted return

Value-based Reinforcement Learning

- Learn optimal policy indirectly through an optimal *value function.*
- The *state-action value function* for a policy π is:

$$
\begin{aligned}
q_\pi(s,a) = & \; \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \,|\, S_0 = s, A_0 = a\right] \\
= & \; \mathbb{E}\left[R_1 \,|\, S_0 = s, A_0 = a\right] + \gamma \mathbb{E}\left[q_\pi(S_1, A_1) \,|\, S_0 = s, A_0 = a\right]
\end{aligned}
$$

- Value-based methods for control aim to learn $q_{\pi*}$, often denoted $q*$
- Then in any state s can take action $\text{argmax}_a q*(s,a)$ in every state

Q-learning

- Q-learning [1] maintains a $Q(s, a)$ estimate for all state-action pairs, and learns q^*
- Let greedy(Q) denote the "greedy" policy w.r.t. Q:

$$\text{greedy}(Q)(s) = \text{argmax}_a Q(s, a)$$

- Suppose the agent knows ground truth values of its greedy policy: $q_{\text{greedy}(Q)}(s, a)$
- Given a transition (s, a, r, s')
  - Agent can do at least as well as greedy(Q) by choosing action $\text{argmax}_{a'} q_{\text{greedy}(Q)}(s', a')$ in the next state.

[1] Watkins, C. J., & Dayan, P. (1992). Q-learning. *Machine learning*.

Q-learning Update

- Given a transition (s, a, r, s'):
  - Would like to move estimate $Q(s, a)$ to be closer to $r + \gamma \text{max}_{a'} q_{\text{greedy}(Q)}(s', a')$
  - $q_{\text{greedy}(Q)}$ is a function we don't have access to, so we use $r + \gamma \text{max}_{a'} Q(s', a')$:

$$Q(s, a) \leftarrow (1 - \alpha) \underbrace{Q(s, a)}_{\text{prediction}} + \alpha \underbrace{[r + \gamma \text{max}_{a'} Q(s', a')]}_{\text{target}}$$

Overestimation in Q-learning

$$Q(s,a) \to r + \gamma \max_{a'} q_{\text{greedy}(Q)}(s',a')$$
$$\approx r + \gamma \max_{a'} Q(s',a')$$

$$r + \gamma \max_{a'} Q(s',a') \text{ vs. } r + \gamma \max_{a'} q_{\text{greedy}(Q)}(s',a')$$

$$\max_{a'} Q(s',a') \text{ vs. } \max_{a'} q_{\text{greedy}(Q)}(s',a')$$

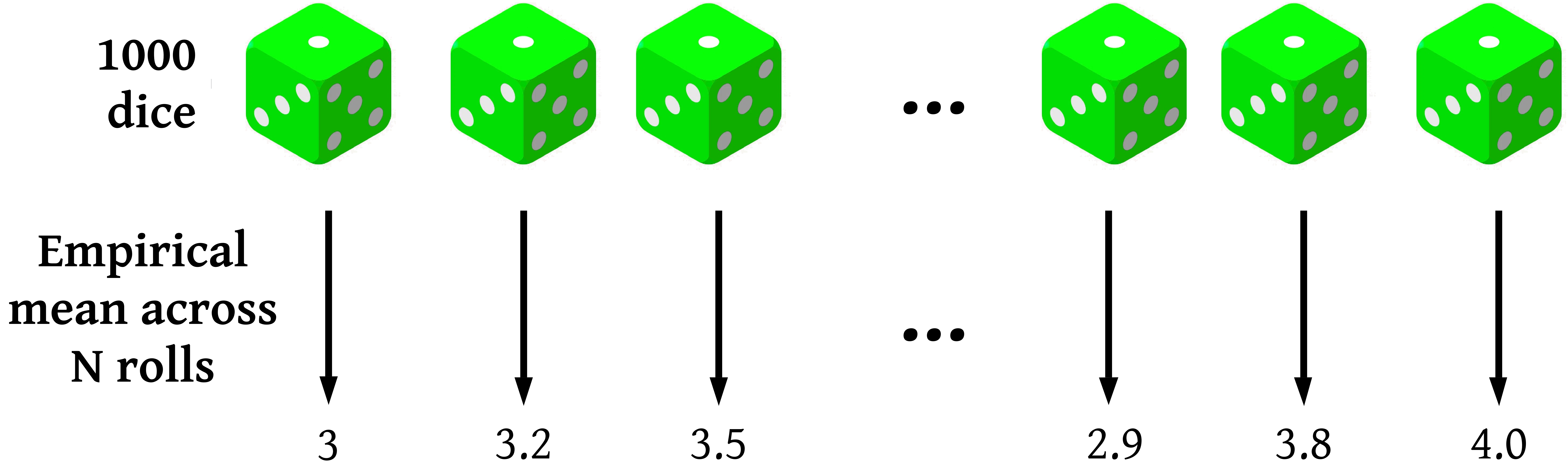
$$\boxed{Q(s,a) > q_{\text{greedy}(Q)}(s,a)}$$

Overestimation

Q-learning suffers from overestimation [2,3].

[2] Hasselt, H. (2010). Double Q-learning. NeurIPS.
[3] Thrun & Schwartz (1993). Issues in Using Function Approximation for Reinforcement Learning. Fourth Connectionist Models Summer School

Maximization Bias: An Example



N=10? **Max is 5.1**

N=100? **Max is ~4**

Maximization Bias: What's going on?

- Suppose we want to **estimate the mean value of the best die.**

- ….. all the dice are fair, and hence have the same mean.

3.5

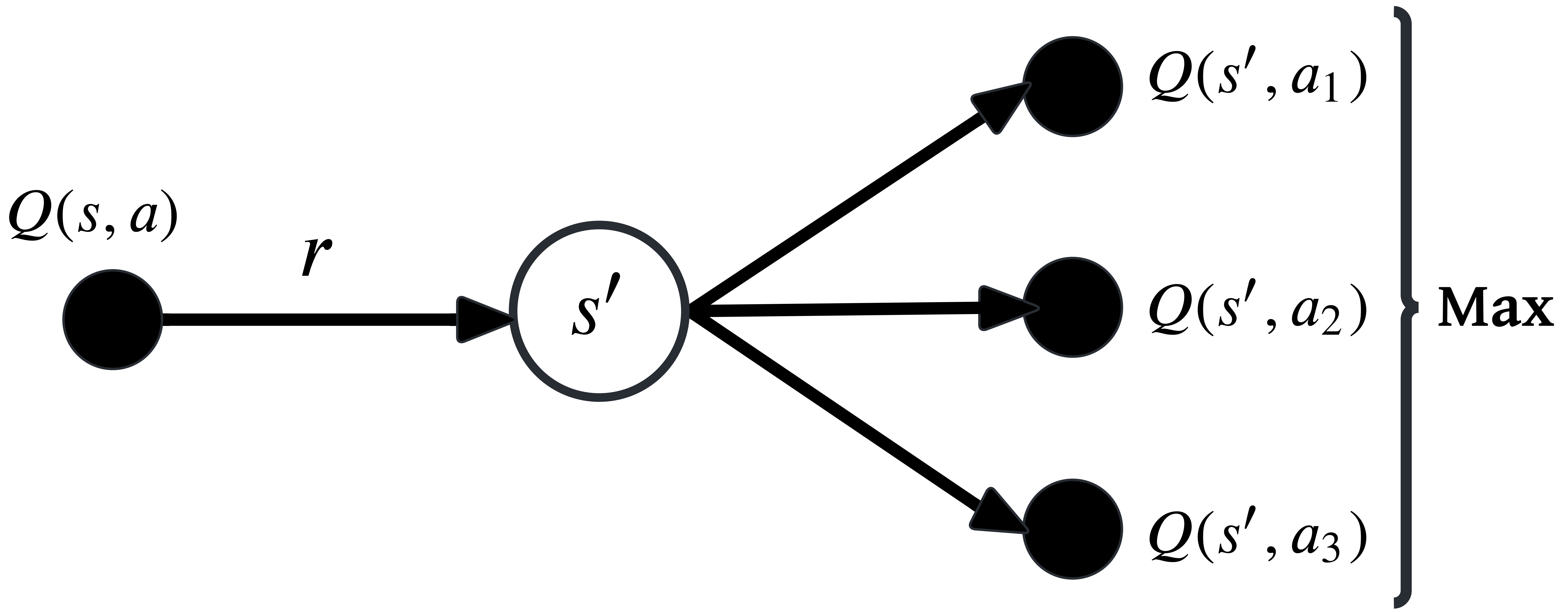
- *What is really going on*: If we use **the maximum of noisy estimates as an estimate of the max, it is (generally) positively biased**.

N=10? **Max is 5.1**

N=100? **Max is ~4**

Maximization Bias in Q-learning

- Recall: Try to move $Q(s, a)$ to be closer to $r + \gamma \max_{a'} Q(s', a')$
- $Q(s', a')$ **is an estimate**. **Estimates $\implies$ noise**



Maximization Bias in Q-learning

- Q-learning
  - $Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha[r + \gamma \text{max}_{a'} Q(s',a')]$
    - $a' = \text{argmax}_{a'} Q(s',a')$

      Use Q-estimates to *select "best" actions*
    - $Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha[r + \gamma Q(s',a')]$

      Use Q-estimates to estimate the value of the best selected action
  - One Q-estimator **both** *selects action to estimate and estimates it*

Q-Learning Internal Dialogue

“Which of these dice do you think is best?”

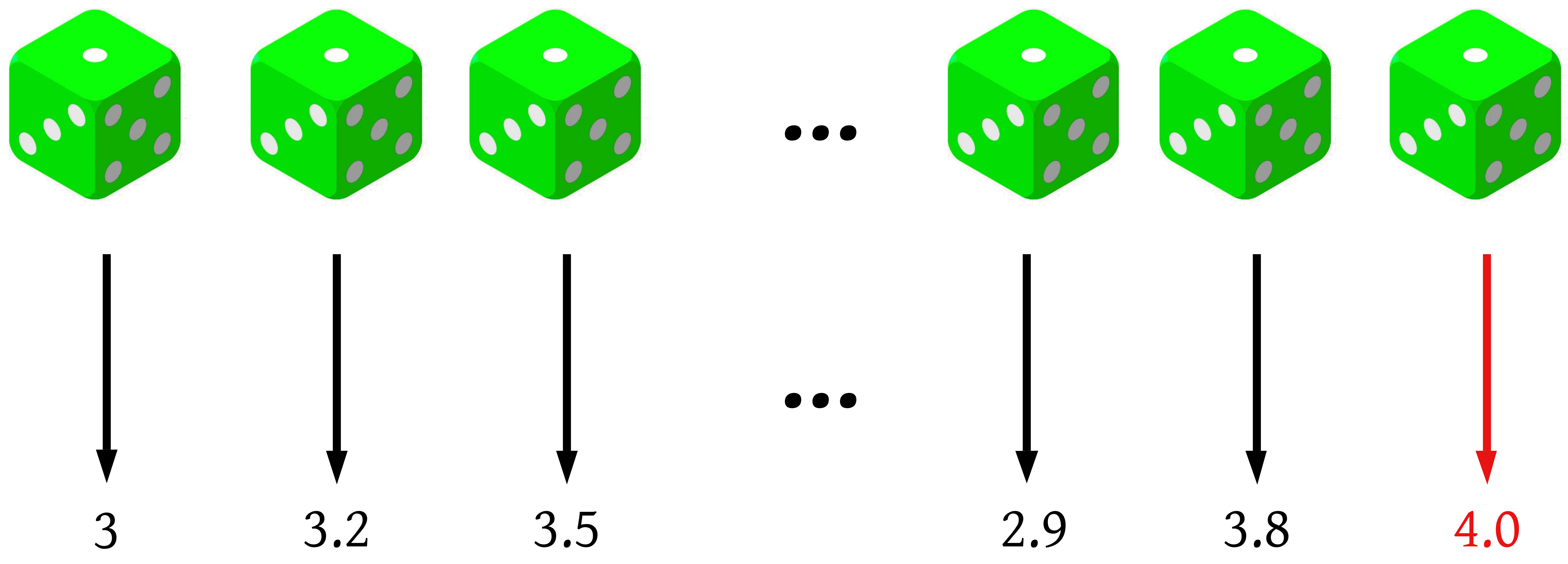
Q-learner: “Dice #300, because it got the highest empirical mean”.

“Oh ok, What do you think is the expected value of the best die?”

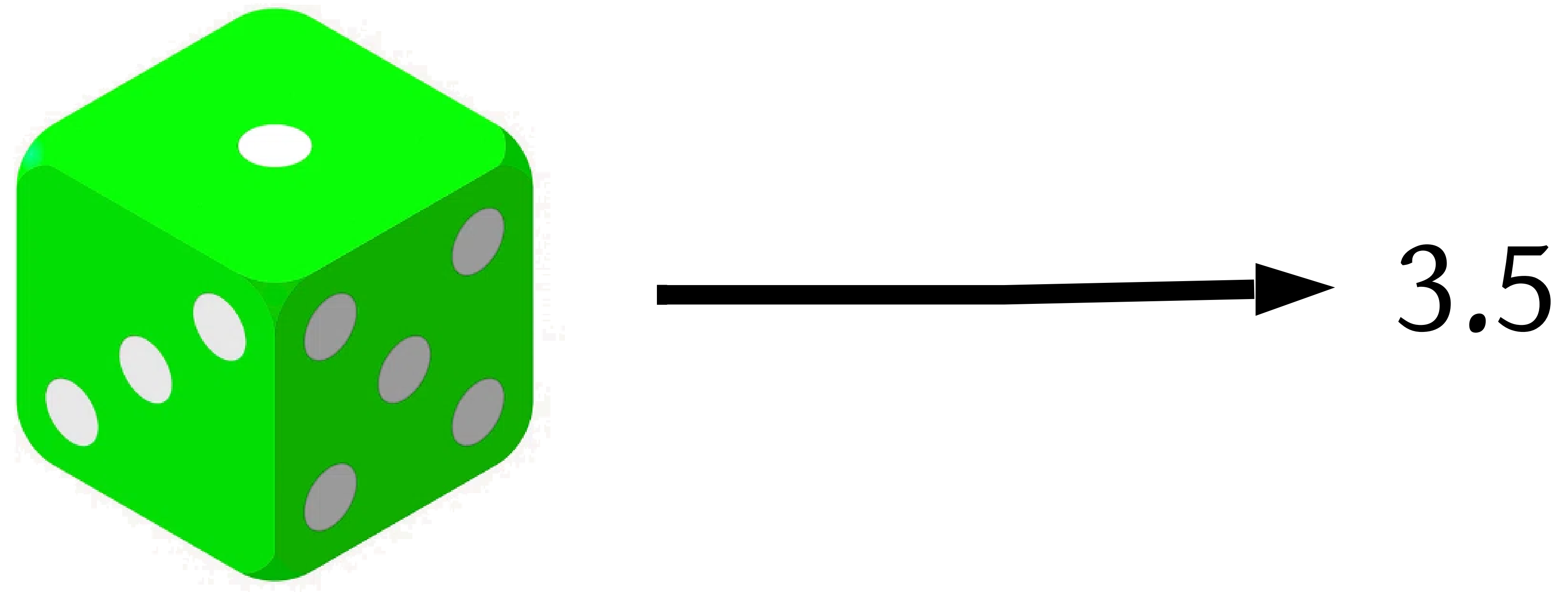
Q-learner: “Well die #300 is my best die, and it rolled 4.0 on average, so I will say that the best die in my set gives me 4.0 on average.”

A more sensible strategy

1. **Select** best die with dice experiment



2. **Run an independent** experiment on that best die and **estimate** the value of that die.



Double Q-learning

- Double Q-learning [2] mitigates overestimation by learning ***2*** Q-functions
- Q-learning
  - $a' = \text{argmax}_{a'} Q(s', a')$
  - $Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha[r + \gamma Q(s', a')]$
- Double Q-learning: Update a single Q-estimator at a time
  - $a' = \text{argmax}_{a'} Q_1(s', a')$
  - $Q_1(s, a) \leftarrow (1 - \alpha)Q_1(s, a) + \alpha[r + \gamma Q_2(s', a')]$

[2] Hasselt, H. (2010). Double Q-learning. NeurIPS.

Why should we care about overestimation?

- Clear constructions where overestimation can significantly hinder learning
- Implicit consensus that less overestimation implies better performance
- Quotes from Double DQN [4] abstract:
  - "It was not previously known whether, in practice, such overestimations are common, whether they harm performance, and whether they can generally be prevented. In this paper, we answer all these questions affirmatively"
  - "We propose a specific adaptation to the DQN algorithm and show that the resulting algorithm not only reduces the observed overestimations, as hypothesized, but that this also leads to much better performance on several games.

[4] van Hasselt et al. (2016). Deep Reinforcement Learning with Double Q-learning. *AAAI.*

Deep RL

Deep Q-Networks (DQN)

- Q-learning and Double Q-learning are "tabular" algorithms
- For large state spaces, we require *function approximation*
  - Represent Q-function with function approximators (e.g., neural networks) rather than as tables.
- Deep Q-networks [5] takes Q-learning and enables us to approximate the Q-function with deep neural networks
  - First algorithm to be able to learn directly from pixels on a diverse set of games.

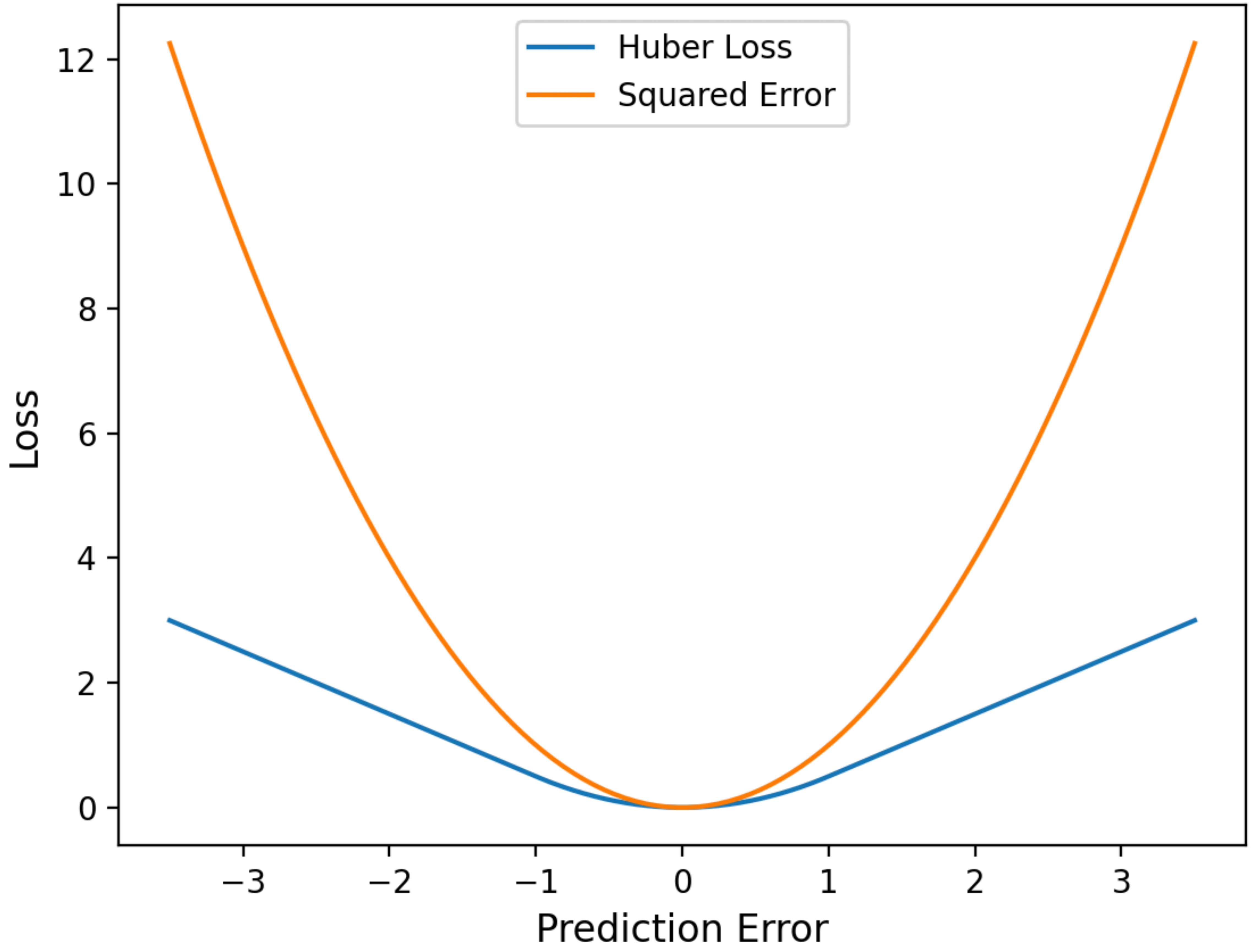
[5] Mnih et al. (2015). Human-level control through deep reinforcement learning. Nature.

Regression in Deep Neural Networks

- Stationary (or fixed) dataset of examples (x, y) of *examples* and *targets (or labels)*
- Neural network θ makes *predictions* $\hat{y} = f(x; \theta)$
- Train to minimize some loss $\mathscr{L}(\hat{y}, y)$
  - Loss is typically a function of prediction error $y - \hat{y}$

Regression

- Take prediction $\hat{y}$ and target y and train under some loss $\mathscr{L}(\hat{y}, y)$



Deep Q-Networks (DQN)

- Trains Q-network θ through regression
- $\hat{y} = Q(s, a; \theta)$ ($Q(s, a)$ in Q-learning)
- $y = r + \gamma \text{max}_{a'} Q(s', a'; \theta^-)$ ($r + \gamma \text{max}_{a'} Q(s', a')$)
- Target network θ^-, **copied from the Q-network θ periodically**.

Extending Double Q-learning to Deep RL

- DQN still suffers from overestimation
- Maintain two Q-networks θ_1 and θ_2 and update **one of the two** networks at each timestep
- Suppose we update θ_1
  - Prediction: $\hat{y} = Q(s, a; \theta_1)$
  - Target:
    - $a' = \text{argmax}_{a'} Q(s', a'; \theta_1^-)$
    - $y = r + \gamma Q(s', a'; \theta_2^-)$

θ_1 selects the action to estimate

θ_2 estimates the action-value

- Call this **True Deep Double Q-learning**

Double DQN

- Double DQN [4] addresses overestimation with a modified target
- $\hat{y} = Q(s, a; \theta)$
- Target
  - $a' = \text{argmax}_{a'} Q(s_{t+1}, a'; \theta);$
  - $y = r_{t+1} + \gamma Q(s_{t+1}, a'; \theta^-)$
- DQN: $a' = \text{argmax}_{a'} Q(s_{t+1}, a'; \theta^-);$
- Double DQN uses the target network as a *proxy* second Q-function

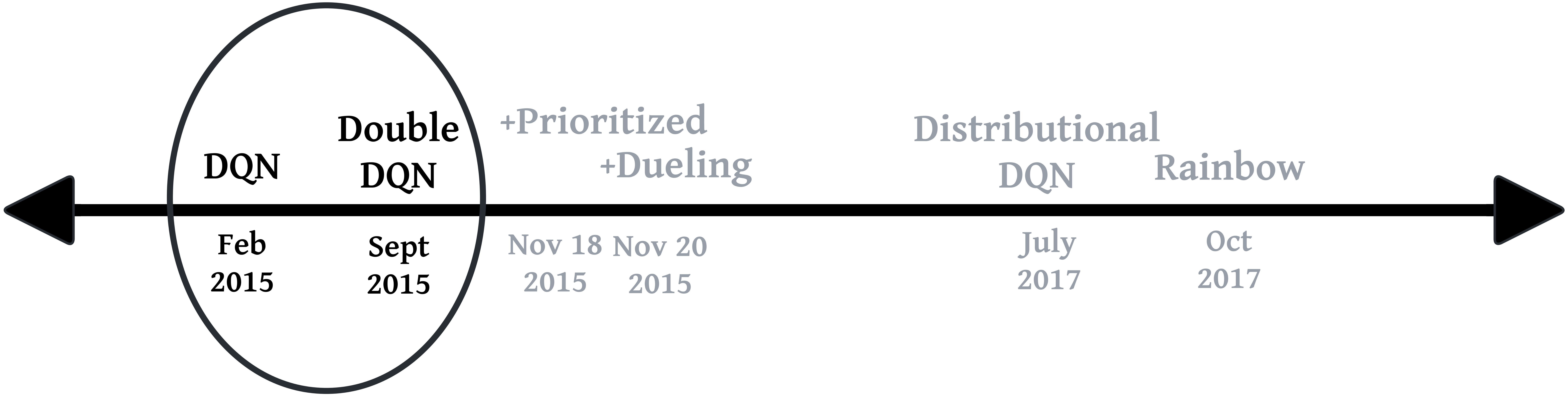
Only one network θ is trained

θ^- may be correlated to θ as it is a recent copy

[4] van Hasselt et al. (2016). Deep Reinforcement Learning with Double Q-learning. *AAAI.*

Experiments

Reminder: DQN & Double DQN



Methodology

- Evaluate in Arcade Learning Environment [6,7], same as original papers
- 6 Environments taken from papers on overestimation in deep RL
- Agents periodically evaluated every 250K timesteps for 125K timesteps
    - Measure scores and overestimations
- 5 seeds with individual curves

[6] Bellemare et al. (2013). The Arcade Learning Environment: An Evaluation Platform for General Agents. *JAIR.*
[7] Machado C. et al. (2018). Revisiting the Arcade Learning Environment: Evaluation Protocols and Open Problems for General Agents. *JAIR.*

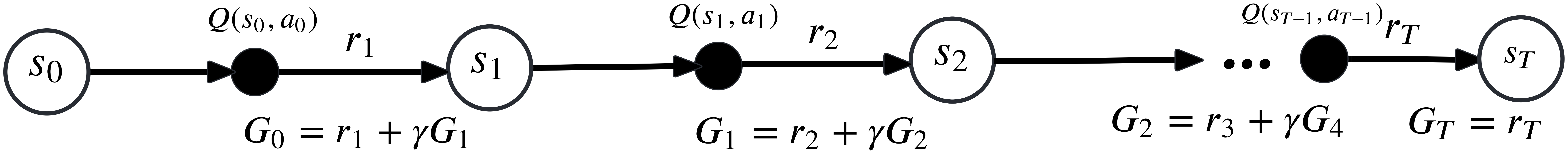
Measuring Overestimation

$$Q(s,a) > q_{\text{greedy}(Q)}(s,a)$$

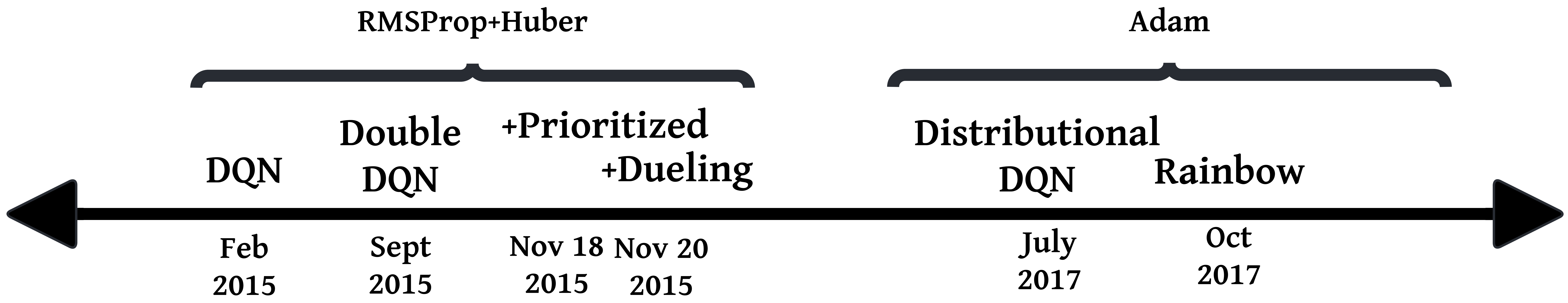
Cannot compute in Atari

$$q_{\text{greedy}(Q)}(s,a) \approx Q^{MC}_{\text{greedy}(Q)}(s,a)$$

Monte Carlo rollouts can give us unbiased estimates of $q_{\text{greedy}(Q)}(s,a)$



Some Deep RL History

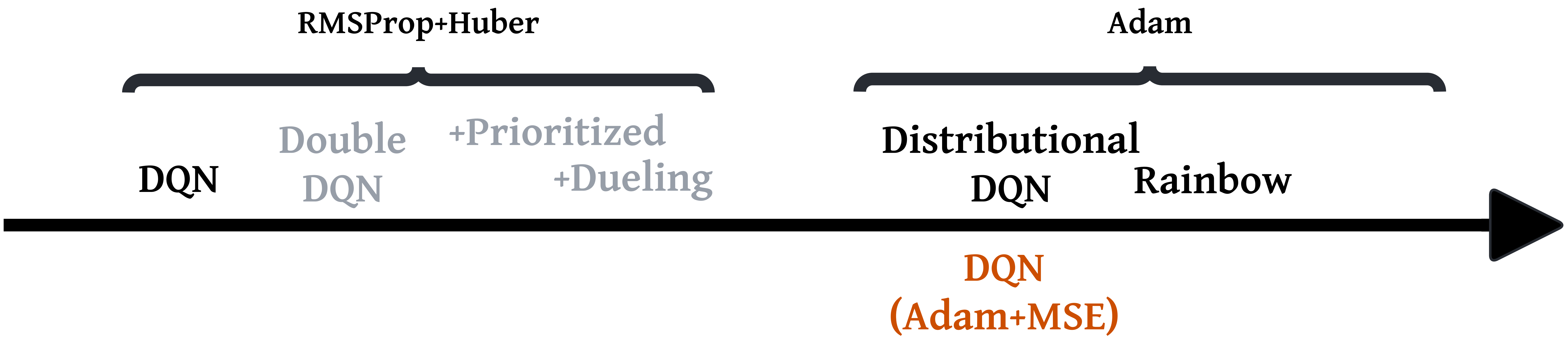


DQN with Adam+MSE

- Obando-Ceron & Castro (2021) [8] showed that a DQN implementation using Adam+MSE outperforms DQN with RMSProp+Huber Loss
  - Tested all of $\{\text{RMSProp, Adam}\} \times \{\text{Huber Loss, MSE}\}$
- Other work [9] showed that DQN with Adam+MSE performs similar to Distributional DQN

[8] Ceron, J. S. O., & Castro, P. S.(2021). Revisiting rainbow: Promoting more insightful and inclusive deep reinforcement learning research. *ICML.*
[9] Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A. C., & Bellemare, M (2021). Deep reinforcement learning at the edge of the statistical precipice. *NeurIPS.*

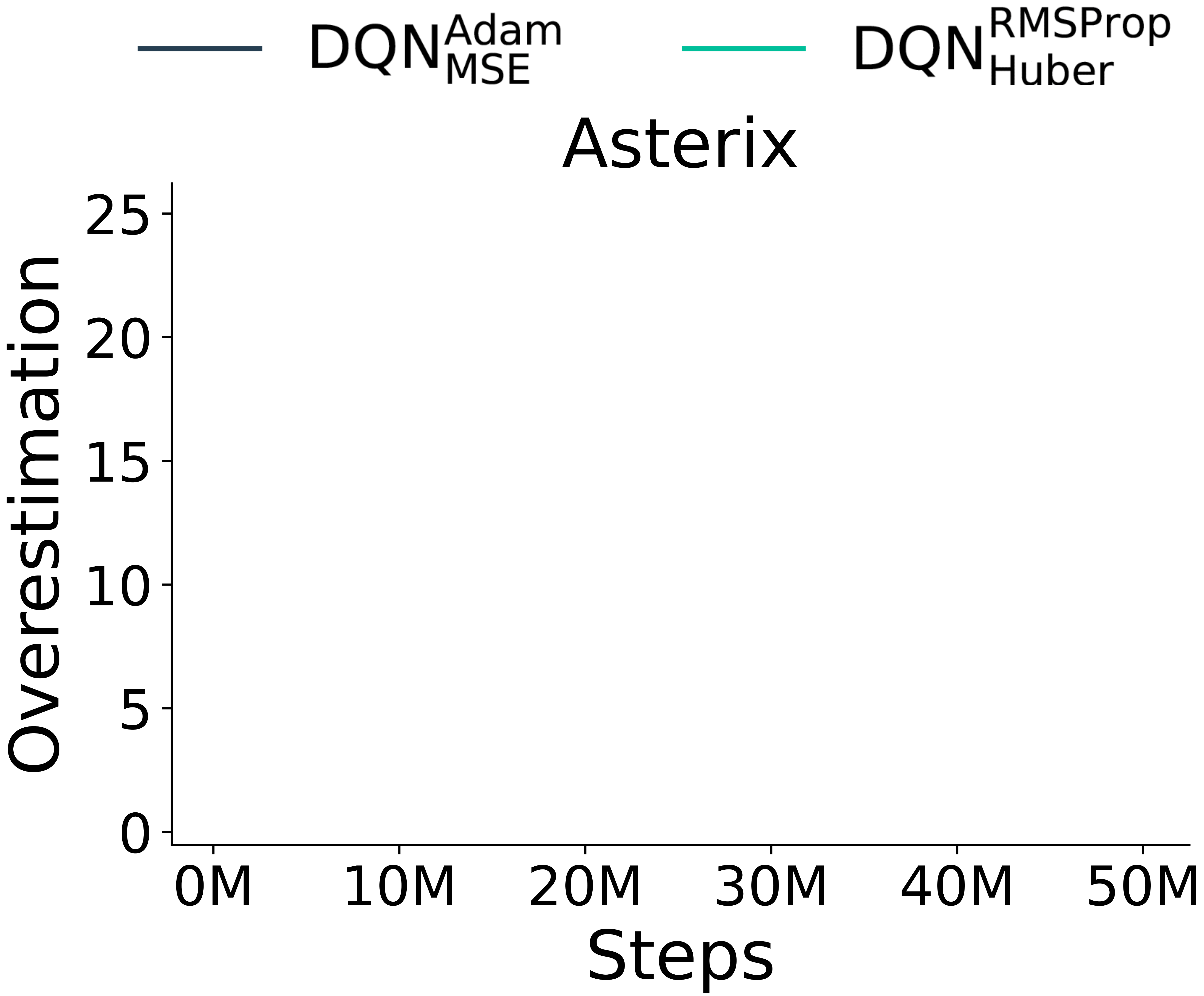
DQN with Adam+MSE



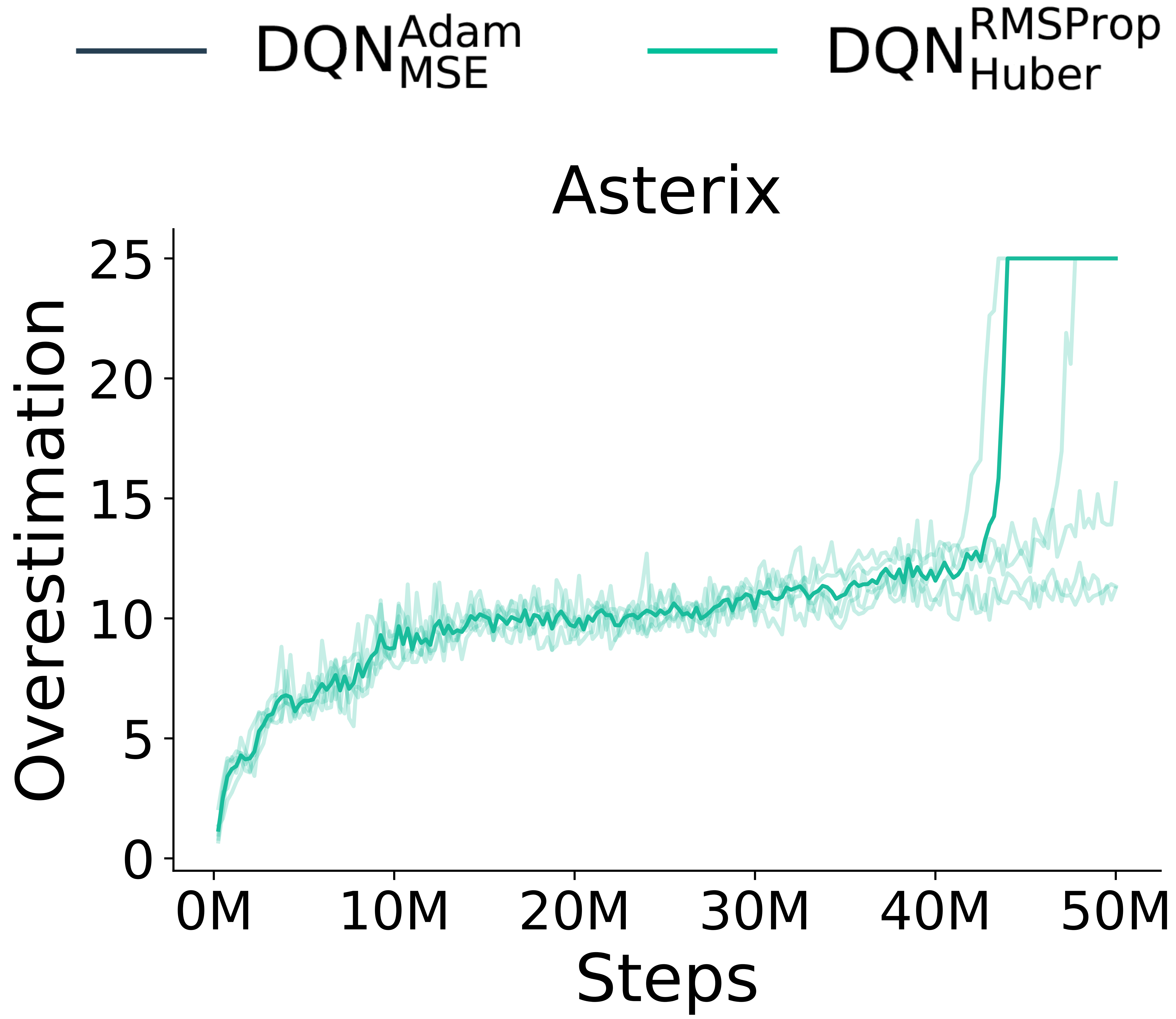
- These advances may still be beneficial, but have not been revisited.

How does **DQN (RMSProp+Huber)** compare to **DQN (Adam+MSE)** in terms of overestimation?

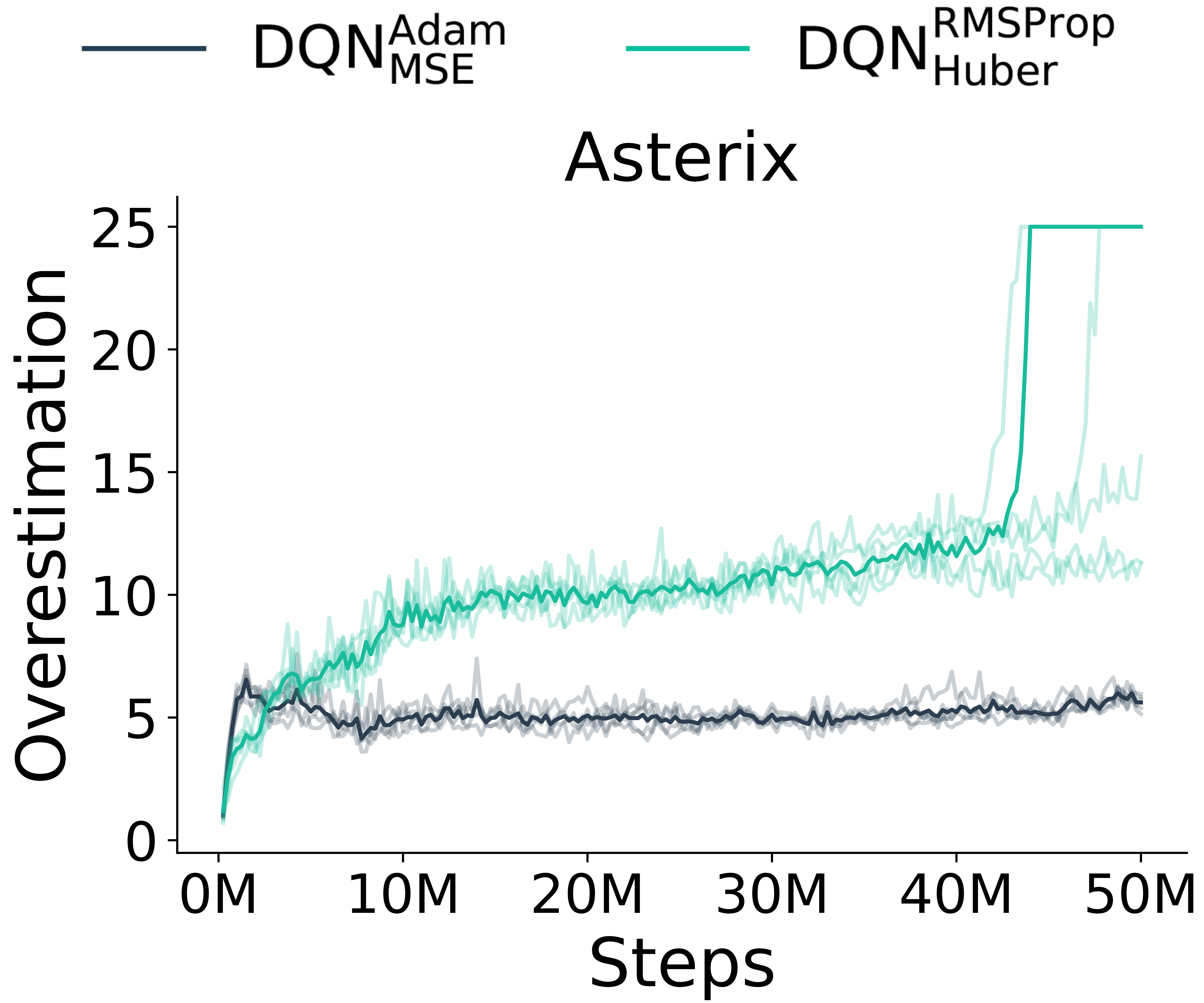
Optimizer-loss Combination and Overestimation



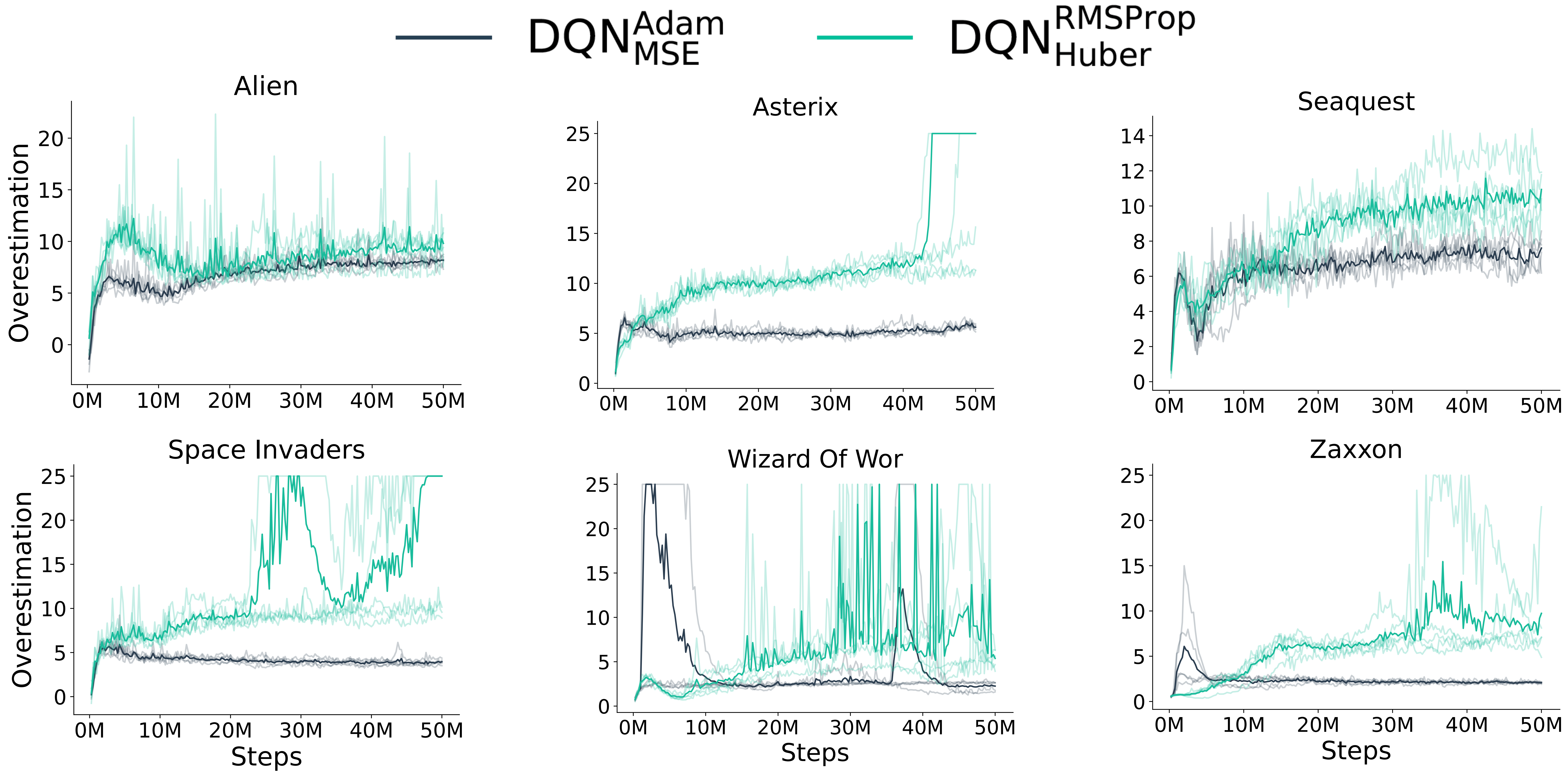
Optimizer-loss Combination and Overestimation



Optimizer-loss Combination and Overestimation

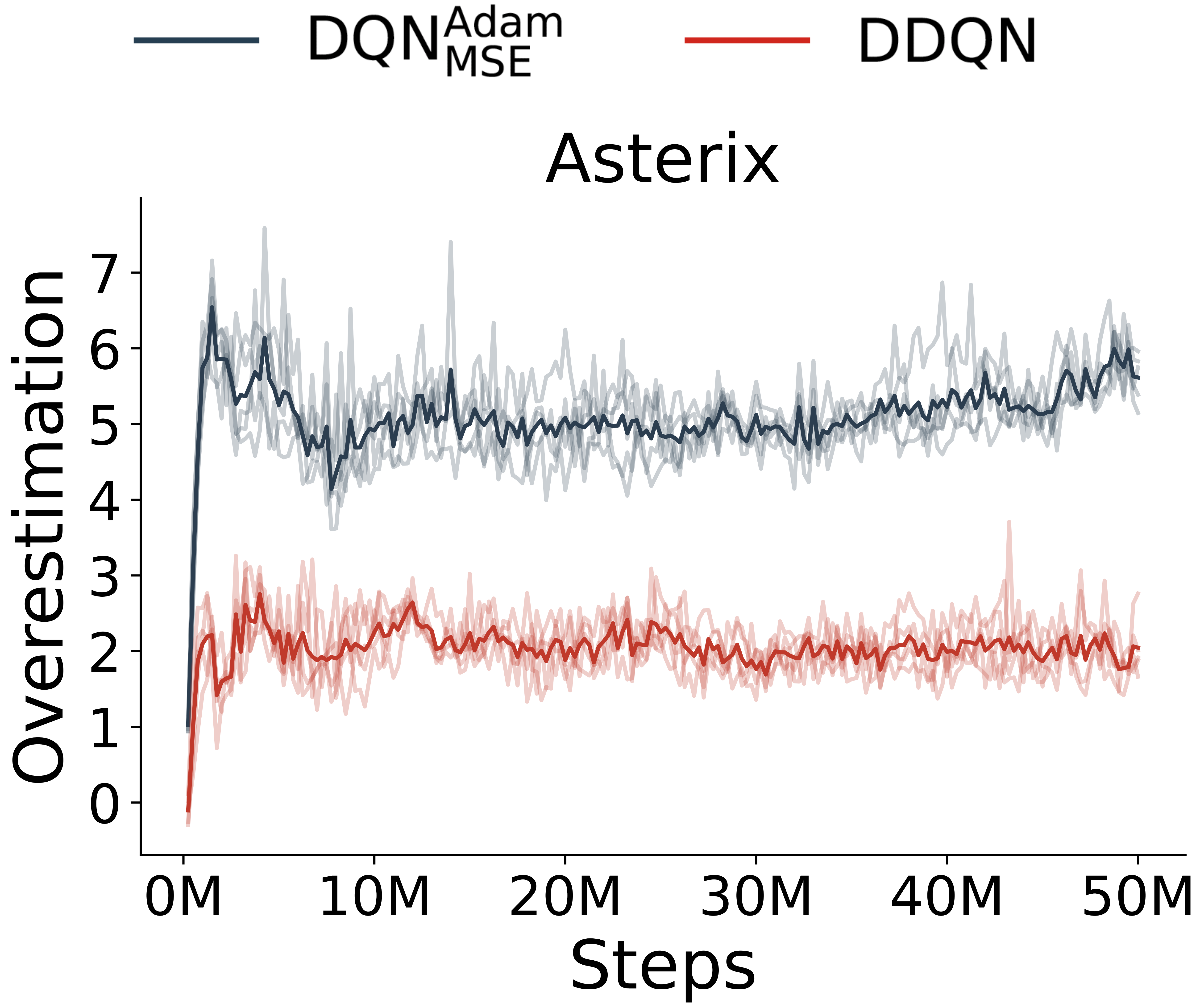


Optimizer-loss Combination and Overestimation

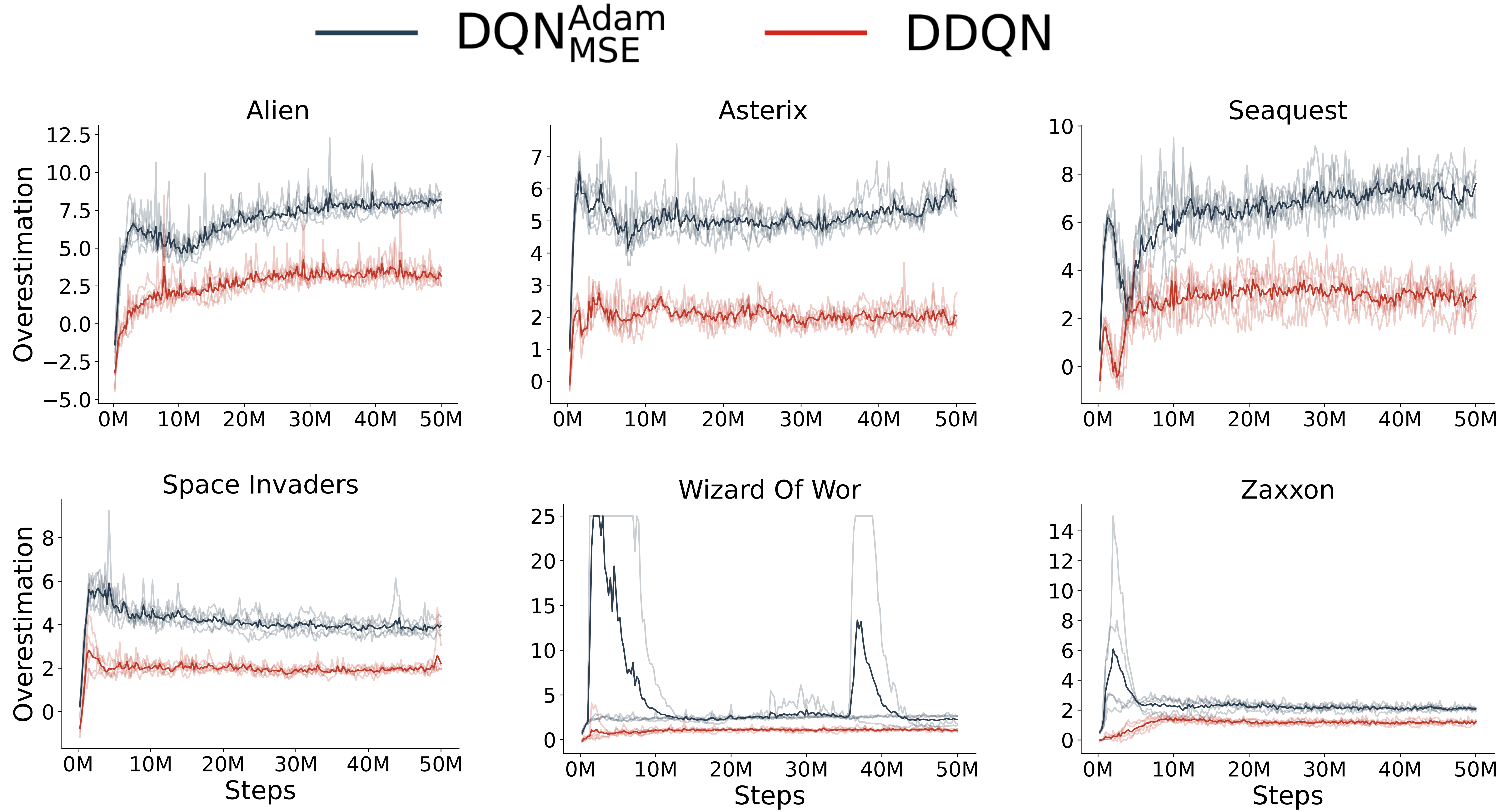


DQN (Adam+MSE) exhibits reduced overestimation. Does **Double DQN (Adam+MSE)** still reduce overestimation over DQN?

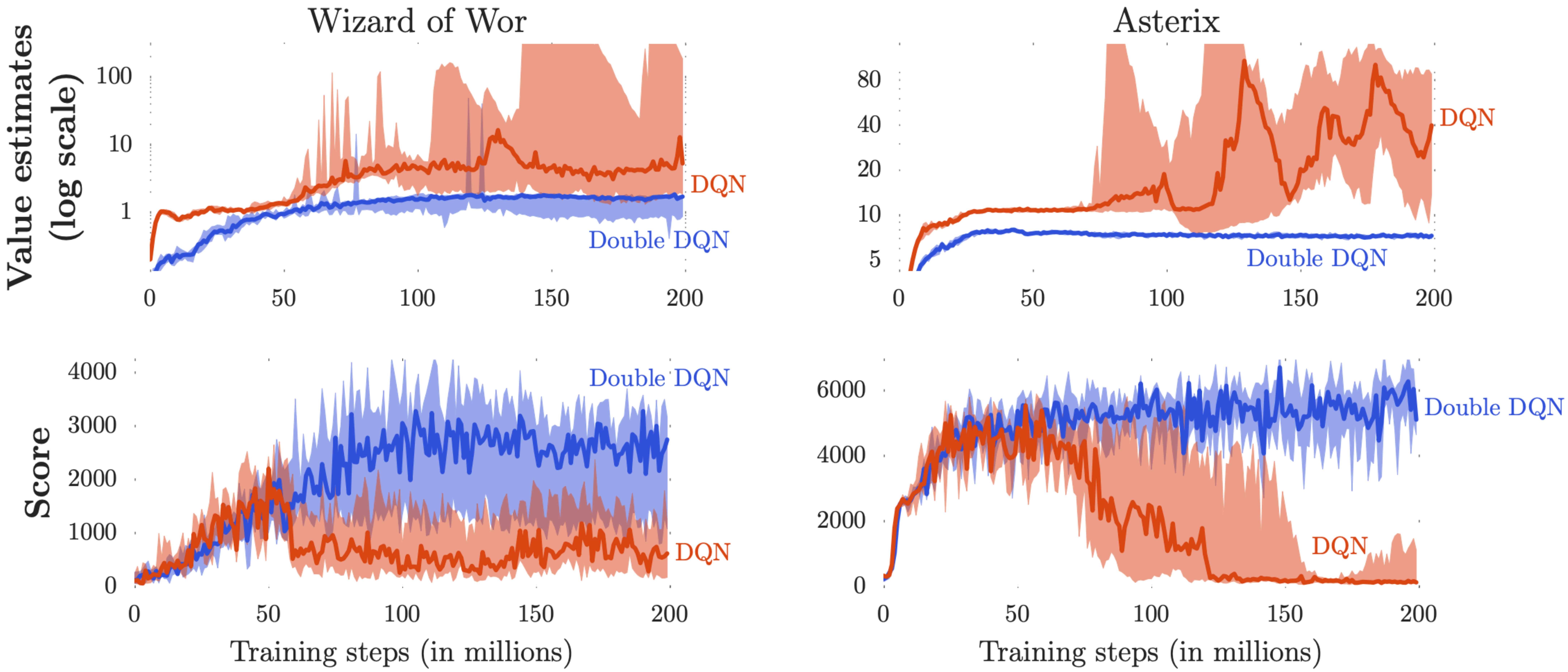
Revisiting Double DQN Overestimation



Revisiting Double DQN Overestimation

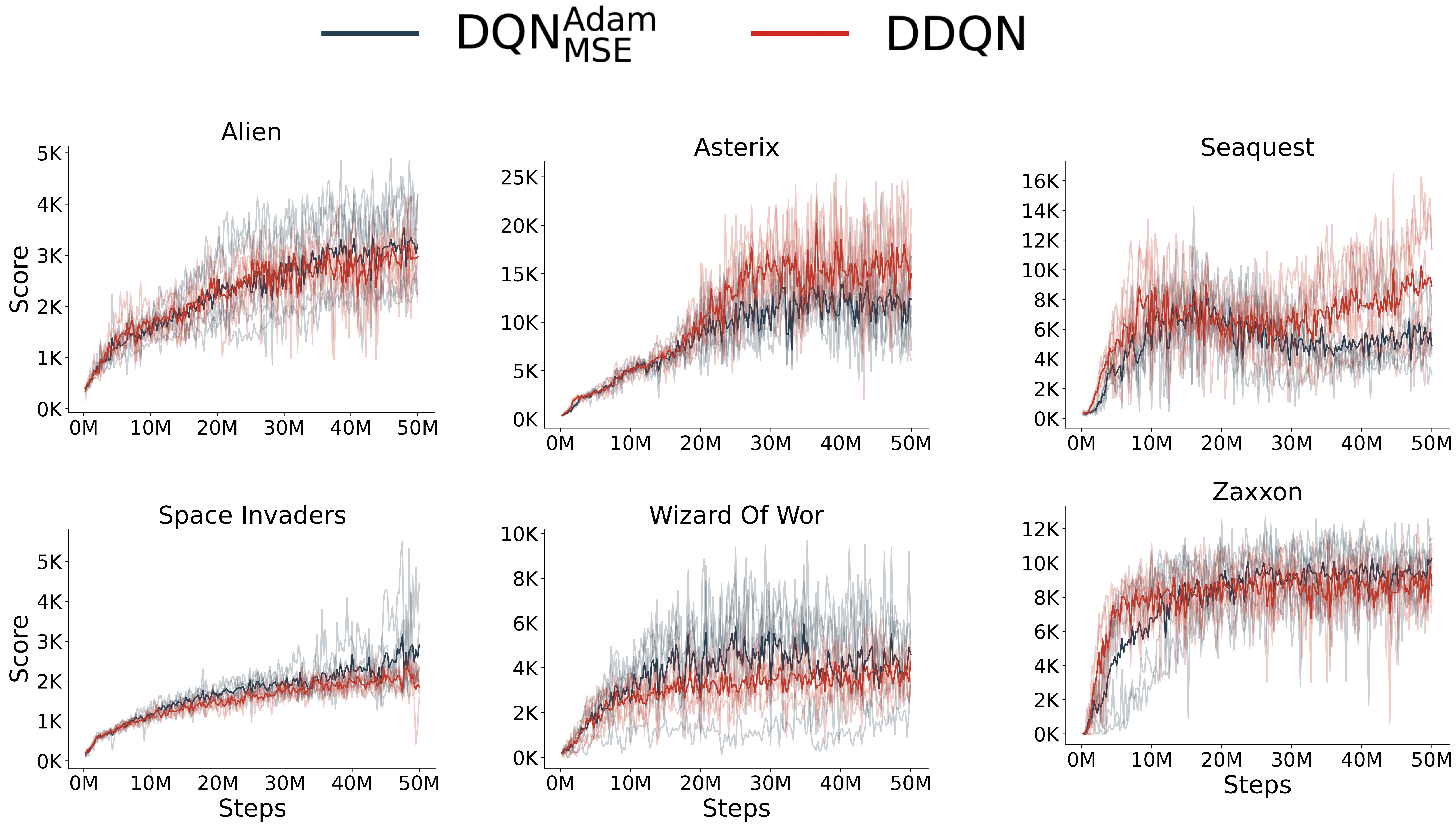


Does **Double DQN** still reduces overestimation over DQN. Does it still boost performance?



Figures taken from van Hasselt et al. 2016

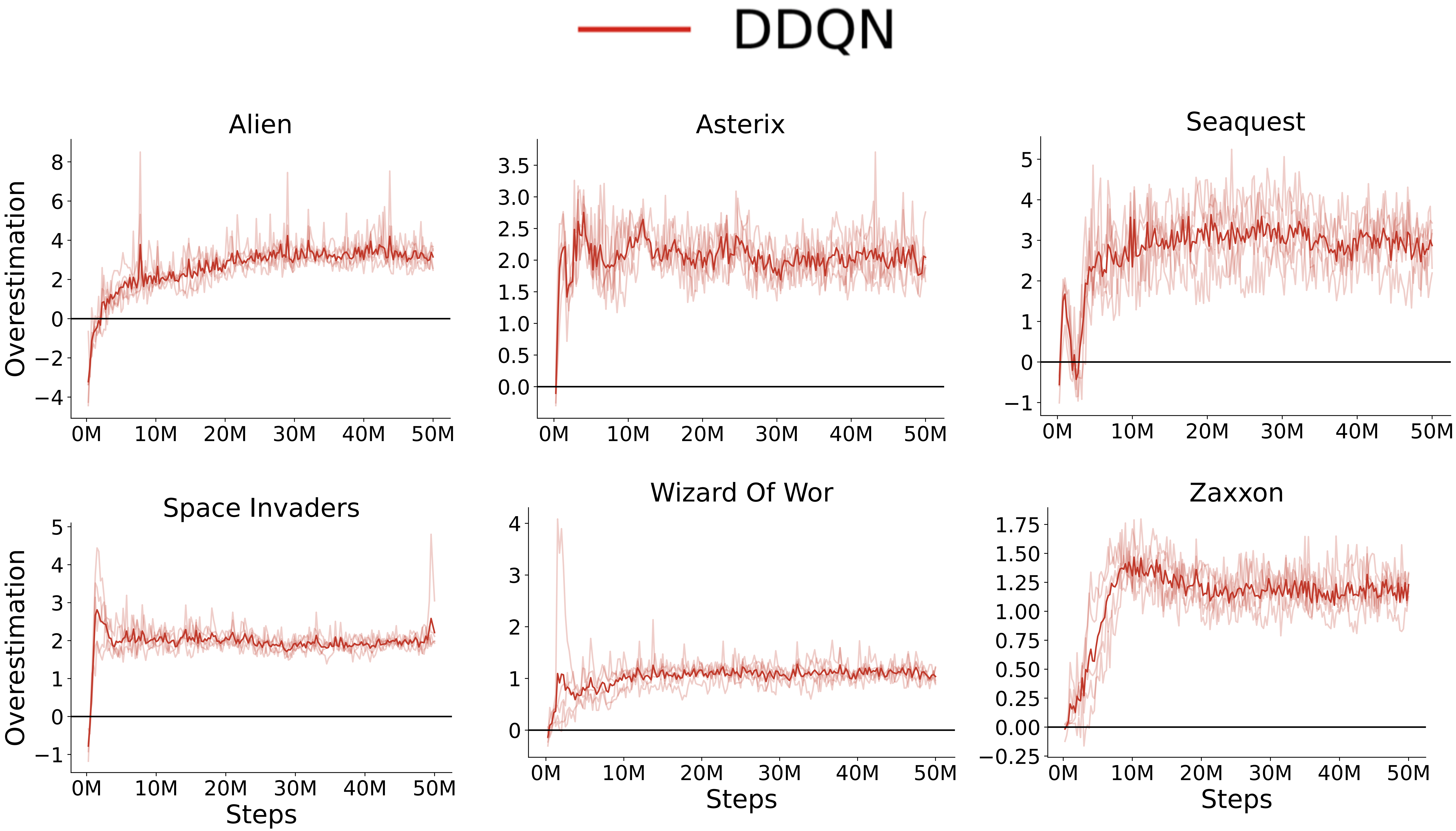
Re-investigating Performance



Re-investigating Performance

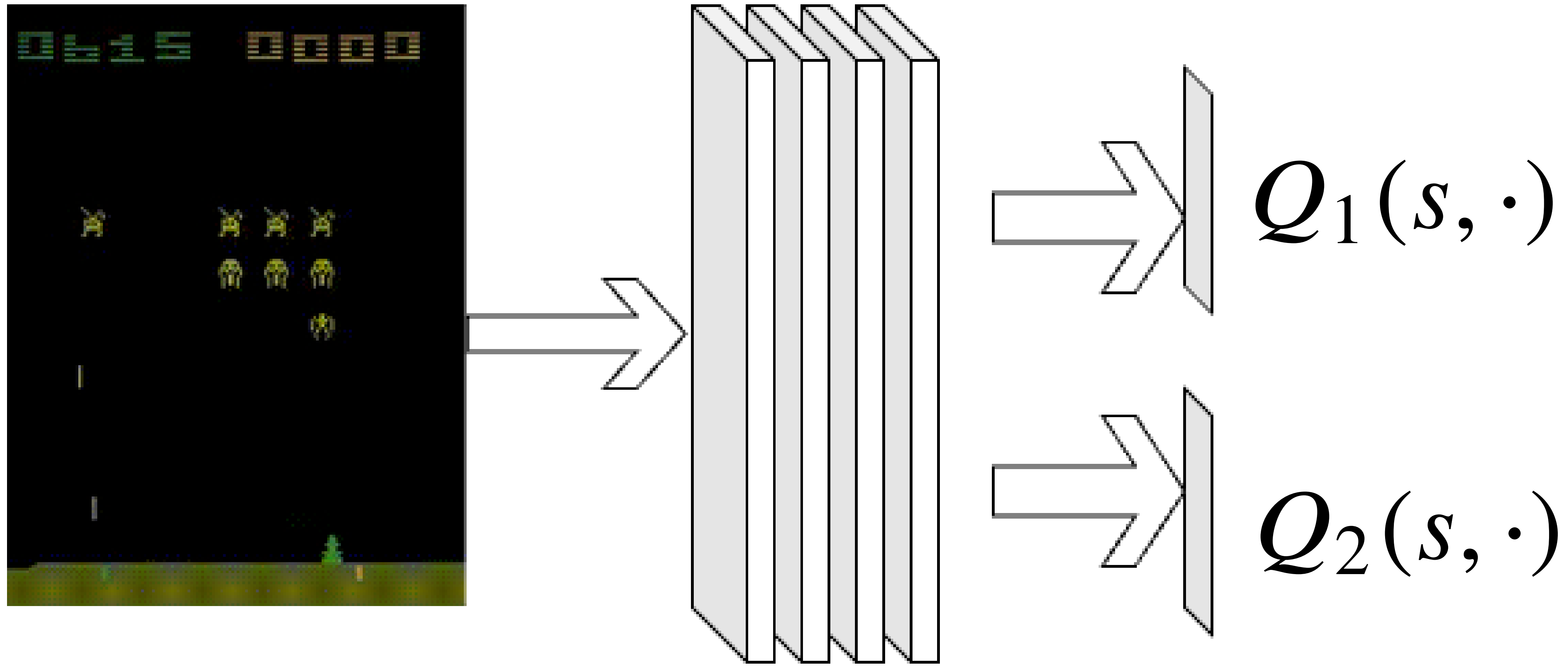
- “It was not previously known whether, in practice, such overestimations are common, **whether they harm performance**, and whether they can generally be prevented. In this paper, **we answer all these questions affirmatively**”
    - At the very least, there are some instances where more overestimation does not harm performance
- “We propose a specific adaptation to the DQN algorithm and show that the resulting algorithm **not only reduces the observed overestimations**, as hypothesized, but **that this also leads to much better performance on several games**.
    - We still observe reduced overestimation
    - We do not observe a significant difference in performance
    - Causal implication…. Does overestimation really harm performance? Or does divergence harm performance?

Double DQN Overestimation

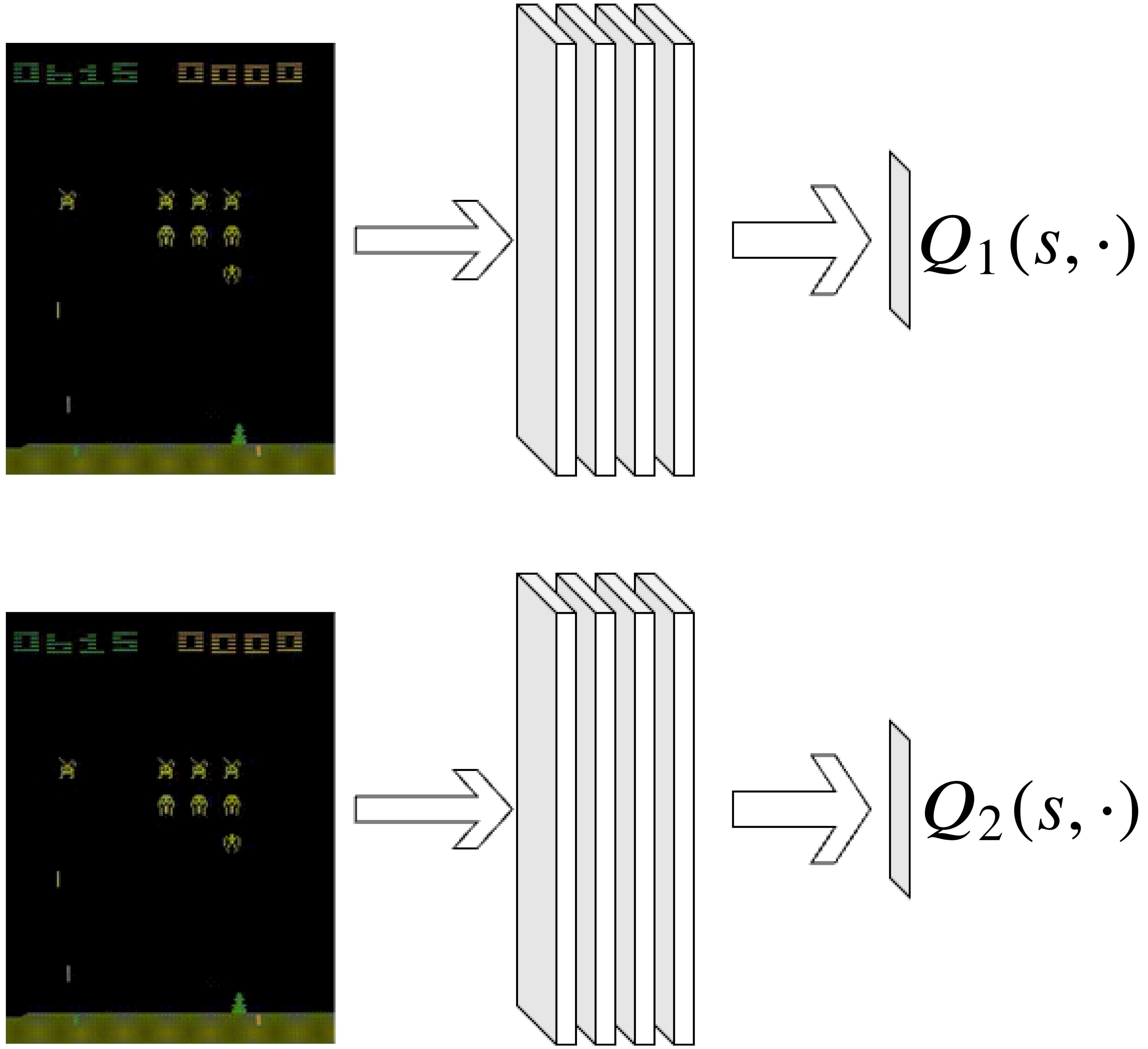


Can True Deep Double Q-learning reduce overestimation over Double DQN?

True Deep Double Q-learning

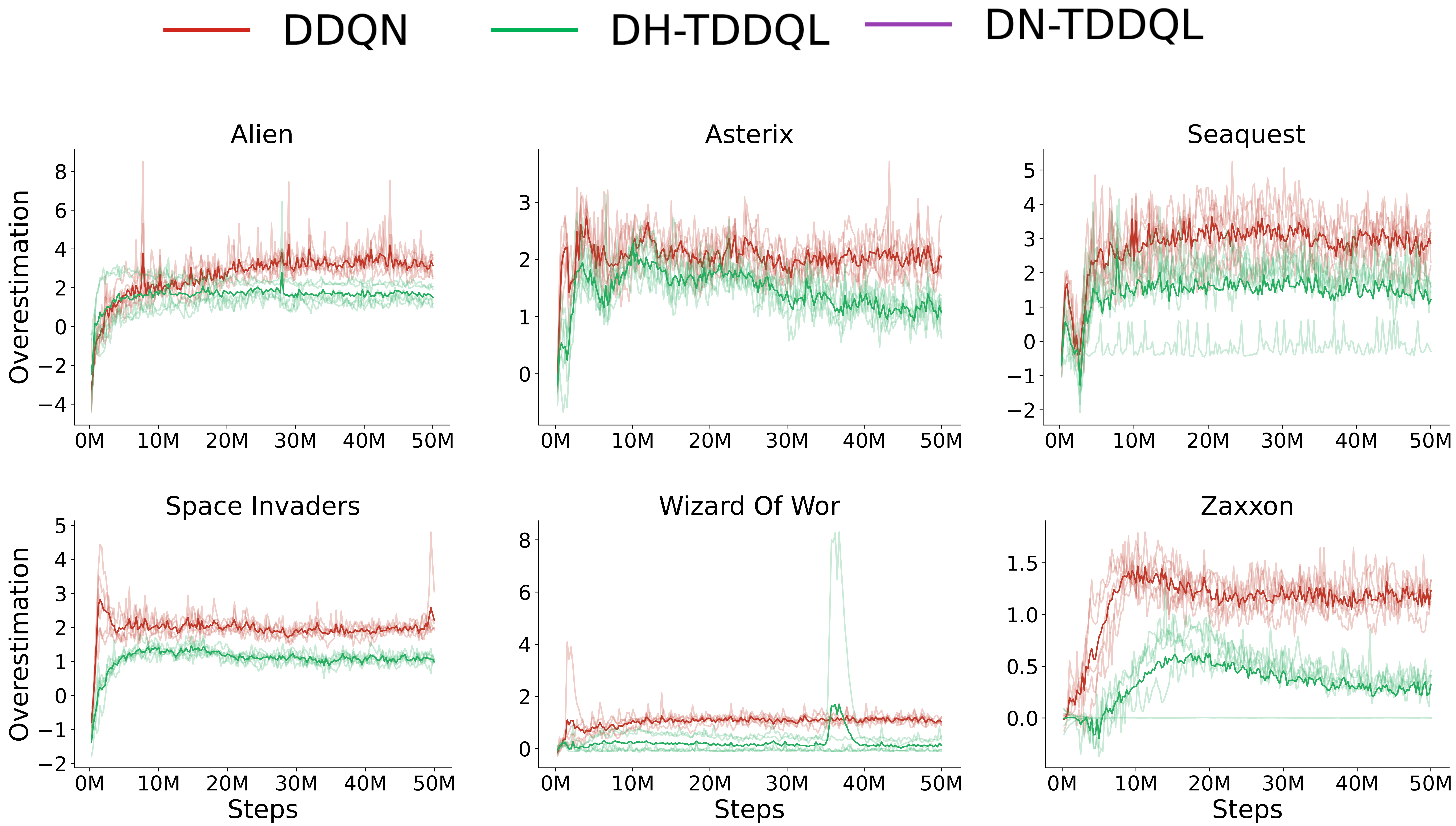


Double Head TDDQL

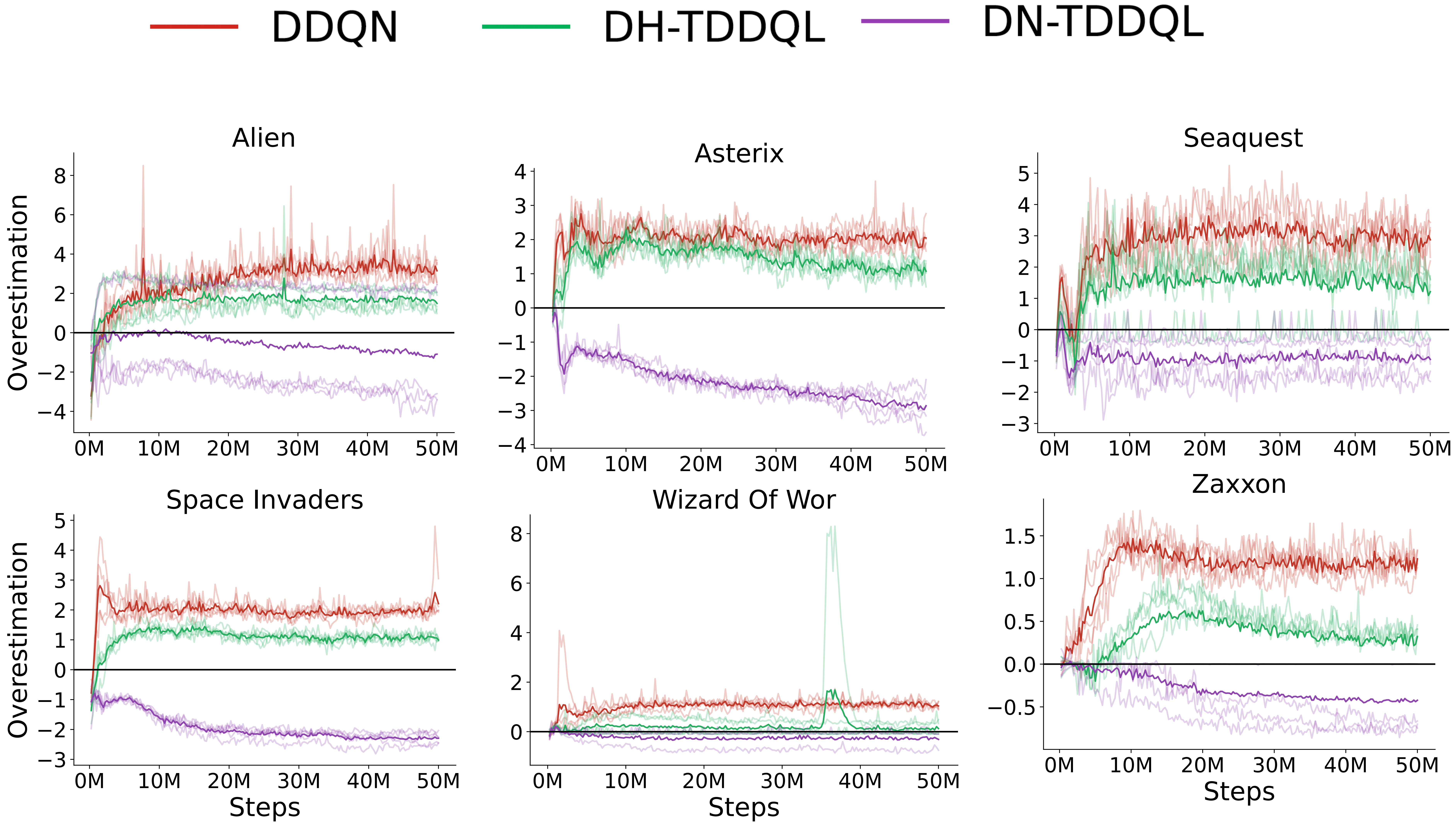


Double Net TDDQL

True DDQL: Overestimation

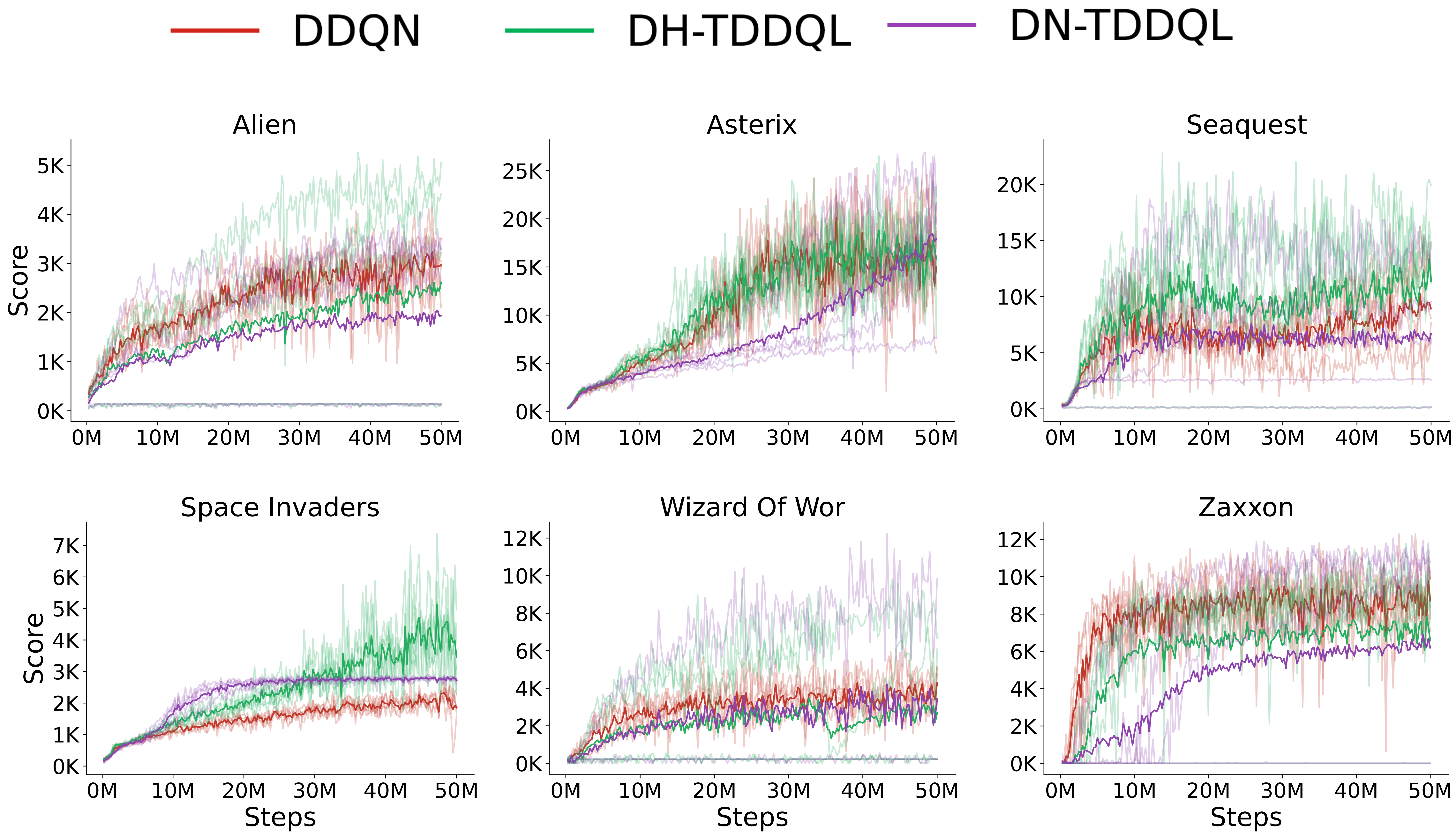


True DDQL: Overestimation



How does **TDDQL** compare to **Double DQN** in terms of performance?

TDDQL Performance



Summarizing results

- The DQN (Adam+MSE) shows less overestimation than DQN (RMSProp+Huber)
- Double DQN (Adam+MSE) still reduces overestimation
- Maintaining two Q-functions reduces overestimation over Double DQN
- On these tasks, it seems overestimation matters up to a point

Observations

- Open question: When does overestimation matter or not matter for performance?
- Advances in deep learning can give big gains in deep RL
- Revisiting algorithms can be insightful!
  - Do not assume that what was **not written** in papers **must have been tried** and thus **must be bad**.