Assignment 3: Q-Learning and Actor-Critic Algorithms

Arij Boubaker, Linghao Zeng, Zhe Huang

1 Multistep Q-Learning

Consider the N-step variant of Q-learning described in lecture. We learn $Q_{\phi_{k+1}}$ with the following updates:

$$y_{j,t} \leftarrow \left(\sum_{t'=t}^{t+N-1} \gamma^{t'-t} r_{j,t'} \right) + \gamma^N \max_{\mathbf{a}_{j,t+N}} Q_{\phi_k} \left(\mathbf{s}_{j,t+N}, \mathbf{a}_{j,t+N} \right) \tag{1}$$

$$\phi_{k+1} \leftarrow \arg\min_{\phi \in \Phi} \sum_{j,t} \left(y_{j,t} - Q_\phi \left(\mathbf{s}_{j,t}, \mathbf{a}_{j,t} \right) \right)^2 \tag{2}$$

In these equations, j indicates an index in the replay buffer of trajectories $\mathcal{D}_k$. We first roll out a batch of B trajectories to update $\mathcal{D}_k$ and compute the target values in (1)eq. (1). We then fit $Q_{\phi_{k+1}}$ to these target values with (2)eq. (2). After estimating $Q_{\phi_{k+1}}$, we can then update the policy through an argmax:

$$\pi_{k+1} \left(\mathbf{a}_t \mid \mathbf{s}_t \right) \leftarrow \begin{cases} 1 \text{ if } \mathbf{a}_t = \arg\max_{\mathbf{a}_t} Q_{\phi_{k+1}} \left(\mathbf{s}_t, \mathbf{a}_t \right) \\ 0 \text{ otherwise.} \end{cases} \tag{3}$$

We repeat the steps in eqs. (1) to (3) K times to improve the policy. In this question, you will analyze some properties of this algorithm, which is summarized in Algorithm 1.

Algorithm 1 Multistep Q-Learning

Require: iterations K, batch size B
1: initialize random policy π_0, sample $\phi_0 \sim \Phi$
2: **for** $k = 0 \dots K-1$ **do**
3: Update $\mathcal{D}_{k+1}$ with B new rollouts from π_k
4: compute targets with (1)
5: $Q_{\phi_{k+1}} \leftarrow$ update with (2)
6: $\pi_{k+1} \leftarrow$ update with (3)
7: **end for**
8: **return** π_K

1.1 TD-Learning Bias (2 points)

We say an estimator $\hat{f}_D$ of f constructed using data D sampled from process P is *unbiased* when $\mathbb{E}_{x \sim P}[\hat{f}_D(x) - f(x)] = 0$ at each x.

Assume $\hat{Q}$ is a noisy (but unbiased) estimate for Q. Is the Bellman backup $\mathcal{B}\hat{Q} = r(s,a) + \gamma \max_{a'} \hat{Q}(s', a')$ an unbiased estimate of $\mathcal{B}Q$?

☐ Yes

■ No

Explanation:

This is because using the max operator in the presence of noise can lead to an overestimation bias, which is a well-known issue in Q-learning referred to as "maximization bias."

1.2 Tabular Learning (6 points total)

At each iteration of the algorithm above after the update from eq. (2), Q_{ϕ_k} can be viewed as an estimate of the true optimal Q^*. Consider the following statements:

I. $Q_{\phi_{k+1}}$ is an unbiased estimate of the Q function of the last policy, Q^{π_k}.

II. As $k \to \infty$ for some fixed B, Q_{ϕ_k} is an unbiased estimate of Q^*, i.e., $\lim_{k\to\infty} \mathbb{E}\big[Q_{\phi_k}(s,a) - Q^*(s,a)\big] = 0$.

III. In the limit of infinite iterations and data we recover the optimal Q^*, i.e., $\lim_{k,B\to\infty} \mathbb{E}\left[\|Q_{\phi_{k+1}} - Q^*\|_\infty\right] = 0$.

We make the additional assumptions:

- The state and action spaces are finite.
- Every batch contains at least one experience for each action taken in each state.
- In the tabular setting, Q_{ϕ_k} can express any function, i.e., $\{Q_{\phi_k} : \phi \in \Phi\} = \mathbb{R}^{S\times A}$.

When updating the buffer D_k with B new trajectories in Equation (3) of Algorithm 1, we say:

- When learning *on-policy*, D_k is set to contain only the set of B new rollouts of π (so $|D_k| = B$). Thus, we only train on rollouts from the current policy.
- When learning *off-policy*, we use a fixed dataset $D_k = D$ of B trajectories from another policy π'.

Indicate which of the statements I-III always hold in the following cases. No justification is required.

	I.	**II.**	**III.**
1. $N = 1$ and ...			
(a) on-policy in tabular setting	■	□	□
(b) off-policy in tabular setting	□	□	□
2. $N > 1$ and ...			
(a) on-policy in tabular setting	■	□	□
(b) off-policy in tabular setting	□	■	□
3. In the limit as $N \to \infty$ (no bootstrapping) ...			
(a) on-policy in tabular setting	□	□	■
(b) off-policy in tabular setting	□	□	■

1.3 Variance of Q Estimate (2 points)

Which of the three cases ($N = 1$, $N > 1$, $N \to \infty$) would you expect to have the highest-variance estimate of Q for fixed dataset size B in the limit of infinite iterations k? Lowest-variance?

Highest variance:
- ■ $N = 1$
- □ $N > 1$
- □ $N \to \infty$

Lowest variance:
- □ $N = 1$
- □ $N > 1$
- ■ $N \to \infty$

1.4 Function Approximation (2 points)

Now say we want to represent Q via function approximation rather than with a tabular representation. Assume that for any deterministic policy π (including the optimal policy π^*), function approximation can represent the true Q^π exactly. Which of the following statements are true?

- ■ When $N = 1$, $Q_{\phi_{k+1}}$ is an unbiased estimate of the Q-function of the last policy Q^{π_k}.
- □ When $N = 1$ and in the limit as $B \to \infty$, $k \to \infty$, Q_{ϕ_k} converges to Q^*.
- □ When $N > 1$ (but finite) and in the limit as $B \to \infty$, $k \to \infty$, Q_{ϕ_k} converges to Q^*.
- ■ When $N \to \infty$ and in the limit as $B \to \infty$, $k \to \infty$, Q_{ϕ_k} converges to Q^*.

1.5 Multistep Importance Sampling (5 points)

We can use importance sampling to make the N-step update work off-policy with trajectories drawn from an arbitrary policy. Rewrite Equation (2) to correctly approximate a Q_{ϕ_k} that improves upon π when it is trained on data D consisting of B rollouts of some other policy $\pi'(\mathbf{a}_t \mid \mathbf{s}_t)$.

Do we need to change Equation (2) when $N = 1$? What about as $N \to \infty$?

You may assume that π' always assigns positive mass to each action. [Hint: re-weight each term in the sum using a ratio of likelihoods from the policies π and π'.]

Answer:

To correct Equation (2) for off-policy learning with importance sampling, we can introduce importance sampling ratios, $\rho_{t:t+N}$, which are the product of the ratios of the probabilities of actions taken according to π and π' over the N-step segment:

$$\rho_{t:t+N} = \prod_{\tau=t}^{t+N-1} \frac{\pi(\mathbf{a}_\tau \mid \mathbf{s}_\tau)}{\pi'(\mathbf{a}_\tau \mid \mathbf{s}_\tau)}$$

The modified update equation with importance sampling would then be:

$$Q_{\phi_{k+1}}(\mathbf{s}_t, \mathbf{a}_t) \leftarrow Q_{\phi_k}(\mathbf{s}_t, \mathbf{a}_t) + \alpha \rho_{t:t+N} \left(G_{t:t+N} - Q_{\phi_k}(\mathbf{s}_t, \mathbf{a}_t)\right)$$

where $G_{t:t+N}$ is the N-step return starting from time t.

When $N = 1$, the importance sampling ratio simplifies to $\rho_t = \frac{\pi(\mathbf{a}_t|\mathbf{s}_t)}{\pi'(\mathbf{a}_t|\mathbf{s}_t)}$, and hence, we do need to change Equation (2) to:

$$Q_{\phi_{k+1}}(\mathbf{s}_t, \mathbf{a}_t) \leftarrow Q_{\phi_k}(\mathbf{s}_t, \mathbf{a}_t) + \alpha \rho_t \left(G_{t:t+1} - Q_{\phi_k}(\mathbf{s}_t, \mathbf{a}_t)\right)$$

As $N \to \infty$, if π and π' converge, the importance sampling ratio may approach 1, and the standard off-policy update can be used without modification. However, in practice, this is not guaranteed, and the use of importance sampling remains essential for an accurate off-policy update.

2 Deep Q-Learning

2.1 Basic Q-Learning

Testing this section:

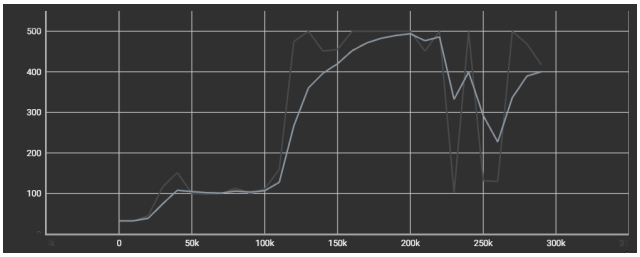

Figure 1: Assessing DQN performance on CartPole-v1 through Evaluation Returns

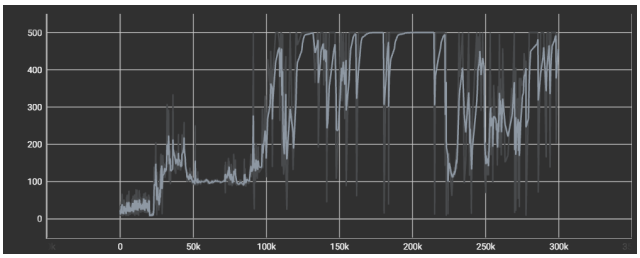

Figure 2: Investigating DQN Performance on CartPole-v1 via Training Returns

Deliverables:

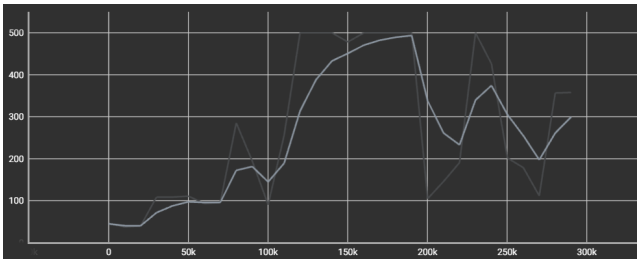

Figure 3: Eval return with 0.05 Learning Rates on CartPole-v1

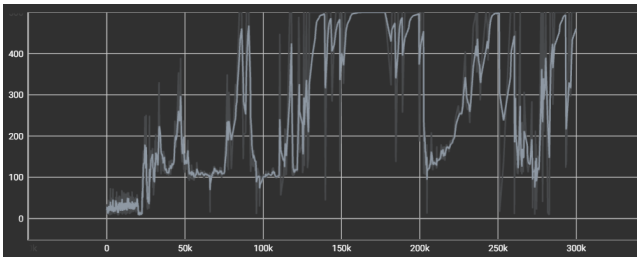

Figure 4: Train return with 0.05 Learning Rates on CartPole-v1

- When running DQN on CartPole-v1 with a significantly higher learning rate, such as 0.05, compared to more commonly used rates in the range of 1e-3 to 1e-4, this can cause the predicted Q-values to become unstable (oscillate or diverge instead of converging to their true values).
- The critic error exhibit increased variance. This is because large updates push the predicted Q-values far from their targets before eventually converging.
- The instability caused by a high learning rate can exacerbate the exploration-exploitation dilemma. An agent whose Q-value estimates are unreliable may struggle to balance exploration with exploitation effectively.

2.2 Double Q-Learning

Deliverables:

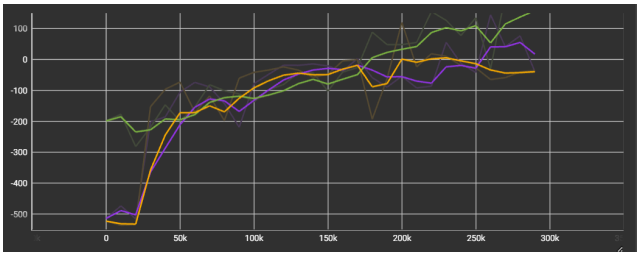

Figure 5: DQN Evaluation Returns on the LunarLander problem

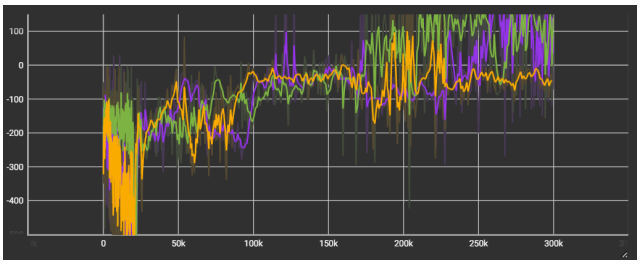

Figure 6: DQN Train Returns on the LunarLander problem

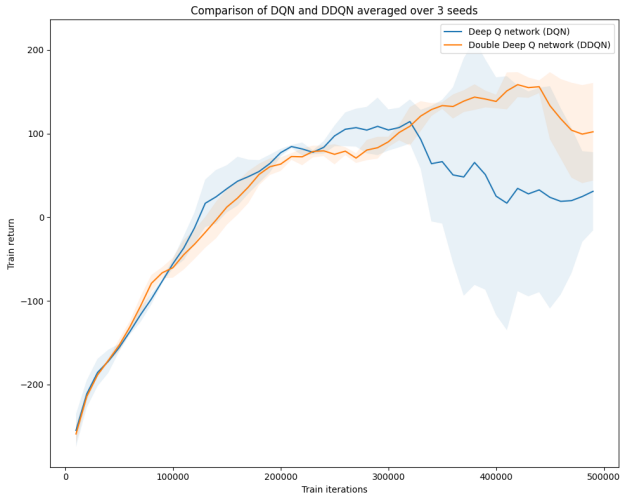

Figure 7: Exploring DQN performance across multiple seeds and Vanilla implementation

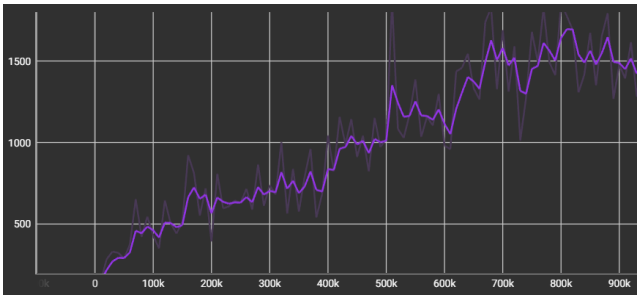

Figure 8: Eval Return on the MsPacman-v0 problem

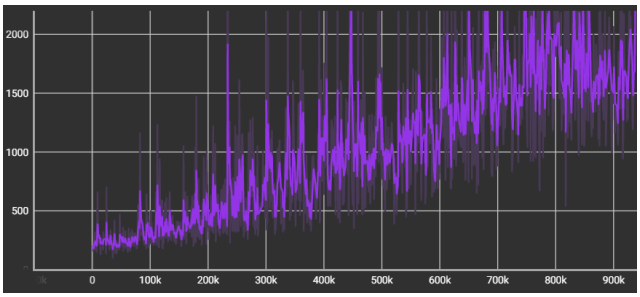

Figure 9: Train Return on the MsPacman-v0 problem

- In Figure 7, they both start similarly but then vanilla DQN's performance begins to drop and becomes unstable, it might be due to overestimation of action values that vanilla DQN is known for. Vanilla DQN often overestimates the value of actions because it uses the same network to select and evaluate an action. This can lead to suboptimal policy choices, making the performance degrade and fluctuate. Double DQN addresses this by using two networks, one for selecting the best action and another for evaluating its value, which tends to give more accurate estimates and leads to more stable and often better performance over time.
- In Figure 8 and Figure 9, the reasons why they look very different early in training are probably that:
  - Exploration: During training, the game-playing AI tries out different moves to learn the game, even if some moves aren't the best.
  - Learning from Past Games: While training, the AI learns from its past games, which can include some not-so-good moves. But during testing, it uses the best strategies it has learned, leading to better performance.

2.3 Experimenting with Hyperparameters

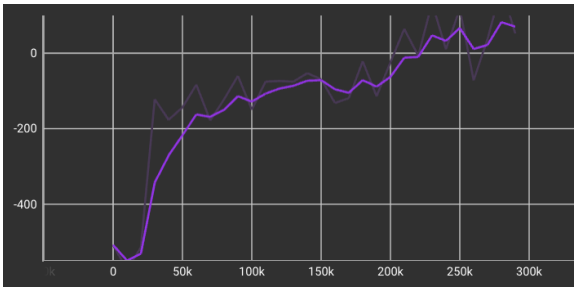

Figure 10: Eval Return on LunarLander with target update period 500

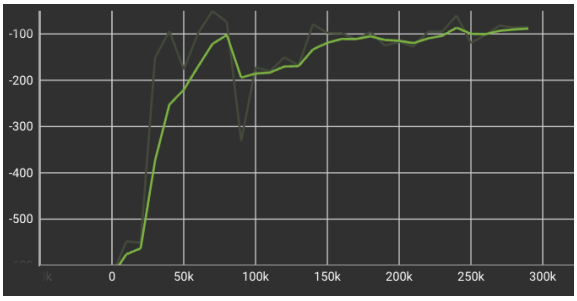

Figure 11: Eval Return on LunarLander with target update period 2000

- The default target update period is 1000, we change it to 500 and 2000, respectively. The reason to choose the target update period for experimentation is that it's a critical factor in how quickly the agent incorporates new information into the stable target that it's trying to learn.
- We found that a target update period of 500 and 2000 did not perform as well as 1000. This could be because a shorter update period like 500 may lead to an overly frequent update of the target network, causing instability in learning due to a constantly shifting learning target. On the other hand, a longer period like 2000 might cause the target network to be updated too infrequently, slowing down the learning process as it doesn't incorporate new knowledge often enough. The period of 1000 might strike the right balance between stability and adaptability in learning.

3 Continuous Actions with Actor-Critic

3.1 Actor with REINFORCE

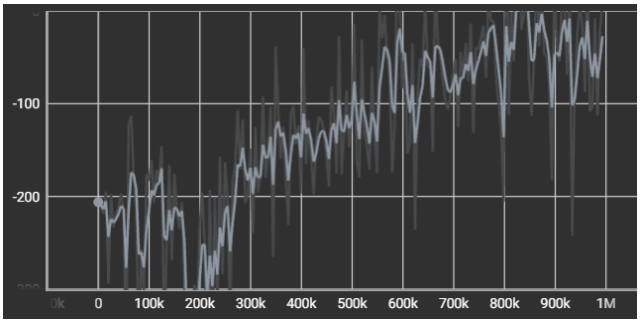

Figure 12: Eval Return with REINFORCE-1 version

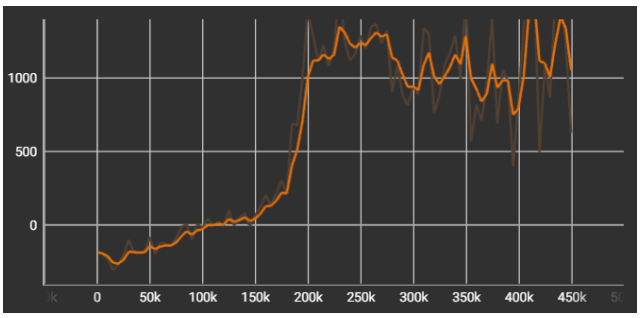

Figure 13: Eval Return with REINFORCE-10 version

- REINFORCE-1 uses a single sample, which can make learning noisier and potentially less stable, as seen in Figure 10. The evaluation return fluctuates significantly, indicating that the agent's performance varies widely from one evaluation to the next. On the other hand, REINFORCE-10, as shown in Figure 11, uses ten samples to estimate the gradient, which typically results in a smoother and more stable learning process. This is because averaging over more samples can give a better estimate of the expected return, leading to more reliable updates to the policy. The eval return in Figure 11 grows more steadily and reaches higher values, suggesting that taking multiple samples for the gradient estimate can significantly improve the learning performance of the agent.

3.2 Actor with REPARAMETRIZE

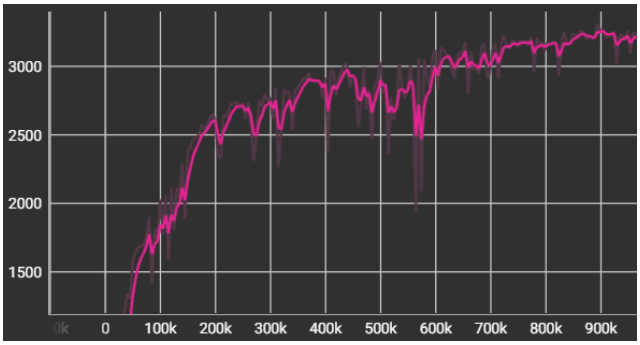

Figure 14: Eval Return with reparametrized HalfCheetah-v4

- The Humanoid-v4 experiment would take more than 20 hours to finish on our environment, so we didn't have time to run it.

3.3 Stabilizing Target Values

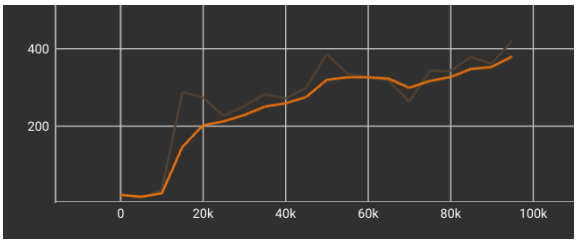

Figure 15: Eval Return with single-Q on Hopper-v4

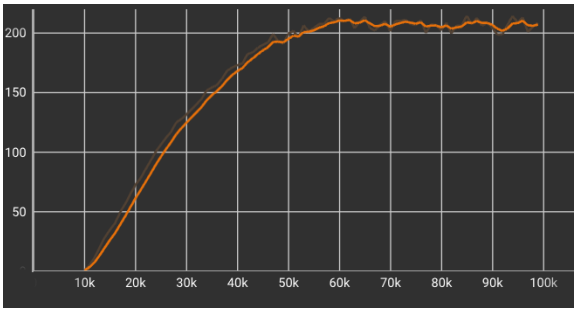

Figure 16: q_values with single-Q on Hopper-v4

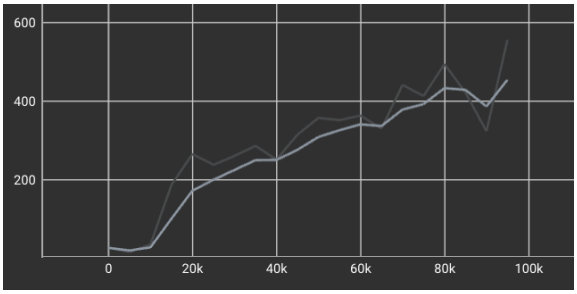

Figure 17: Eval Return with double-Q on Hopper-v4

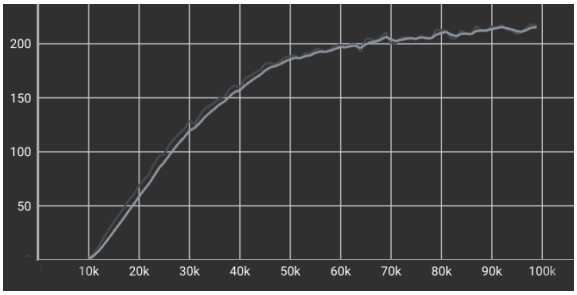

Figure 18: q_values with double-Q on Hopper-v4

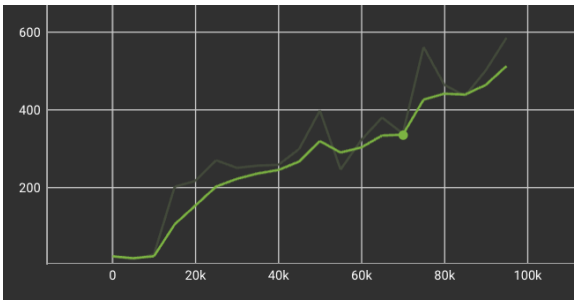

Figure 19: Eval Return with clipped-Q on Hopper-v4

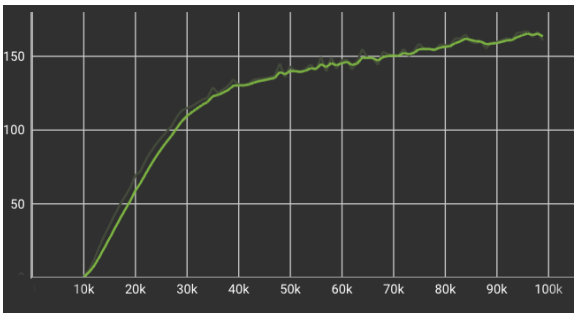

Figure 20: q_values with clipped-Q on Hopper-v4

- Double-Q and clipped double-Q both work well on Hopper-v4, but double-Q has slightly higher q-values, indicating it might overestimate the values of actions a bit more than clipped double-Q, which tries to prevent overestimation. Despite this, their performance is similar, showing that both methods are effective at handling overestimation bias.
- Still, humanoid took too much time and we couldn't finish it.