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:

$$y_{j,t} \leftarrow \left[r_{j,t} + \sum_{t'=t}^{t+N-1} \gamma^{t'-t} r_{j,t'} \right] + \gamma^N \max_{a_{j,t+N}} Q_{\phi_k}(s_{j,t+N}, a_{j,t+N})$$

$$\phi_{k+1} \leftarrow \arg\min_{\phi \in \Phi} \sum_{j,t} \left[y_{j,t} - Q_\phi(s_{j,t}, a_{j,t}) \right]^2$$

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

$$\pi_{k+1}(a_t|s_t) \leftarrow \begin{cases} 1 & \text{if } a_t = \arg\max_{a_t} Q_{\phi_{k+1}}(s_t, a_t) \\ 0 & \text{otherwise.} \end{cases}$$

We repeat the steps in equations (1) to (3) K times to improve the policy. The algorithm is summarized below:

[H] Multistep Q-Learning [1] iterations K, batch size B Initialize random policy π_0, sample $\phi_0 \sim \Phi$ $k = 0$ to $K-1$ Update D_{k+1} with B new rollouts from π_k Compute targets with (1) Update $Q_{\phi_{k+1}}$ with (2) Update π_{k+1} with (3) π_K

1.1 TD-Learning Bias (2 points)

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

Assume $\hat{Q}$ is a noisy (but unbiased) estimate for Q. Is the Bellman backup

$$B\hat{Q} = r(s,a) + \gamma \max_{a'} \hat{Q}(s', a')$$

an unbiased estimate of BQ?

- Yes
- No

The Bellman Backupis not an unbiased estimate because since Q is noisy, taking the maximum action would thus have bias. This is because E[max Q]! = $maxQsinceQisnoisy$.

2 2.4

```
rm -rf ./data/*
python cs285/scripts/run_hw3_dqn.py -cfg experiments/dqn/cartpole.yaml
python cs285/scripts/graph_results.py
```

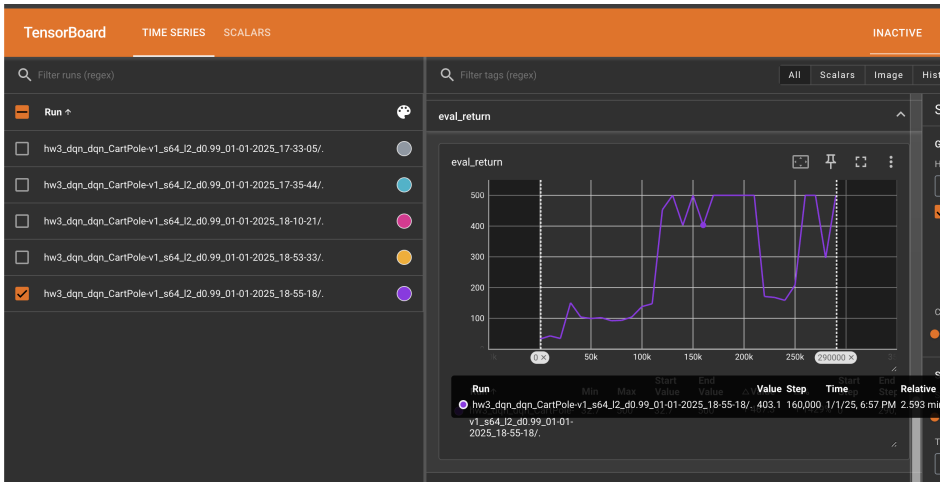

Figure 1: Enter Caption

Note: Purple is seed 3, blue is seed 2, grey is seed 1.

Figure 2: Enter Caption

Change the learning rate to 0.05 in the YAML config file. Analyze the impact on:

1. **Predicted Q-values:** Predicted Q-values increased but became more volatile.
2. **Critic error:** Critic error increased.

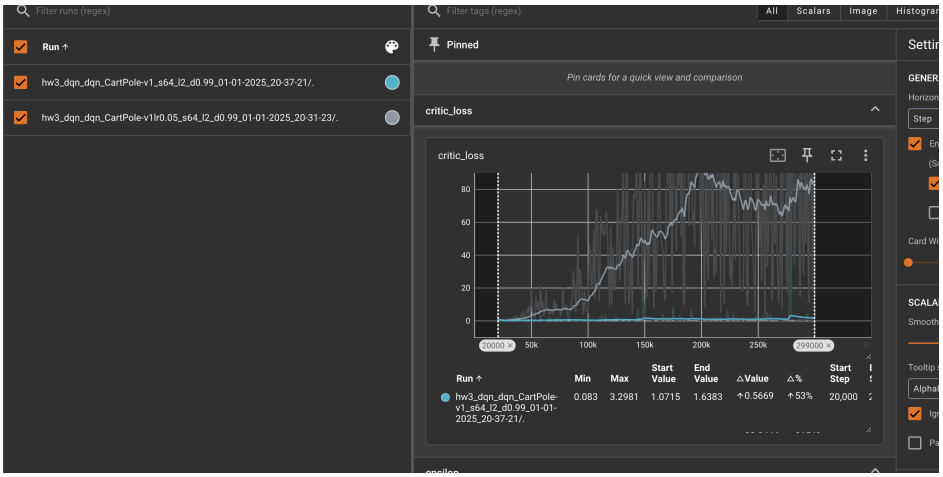

Figure 3: Increased learning rate and critic loss

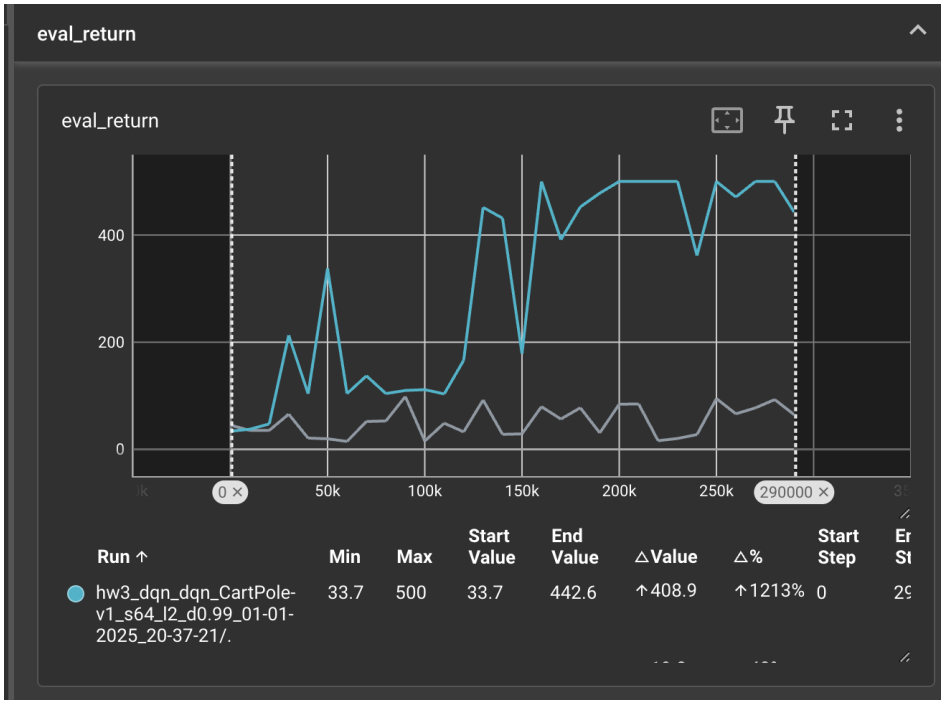

Figure 4: $lr = 0.05$ and eval return

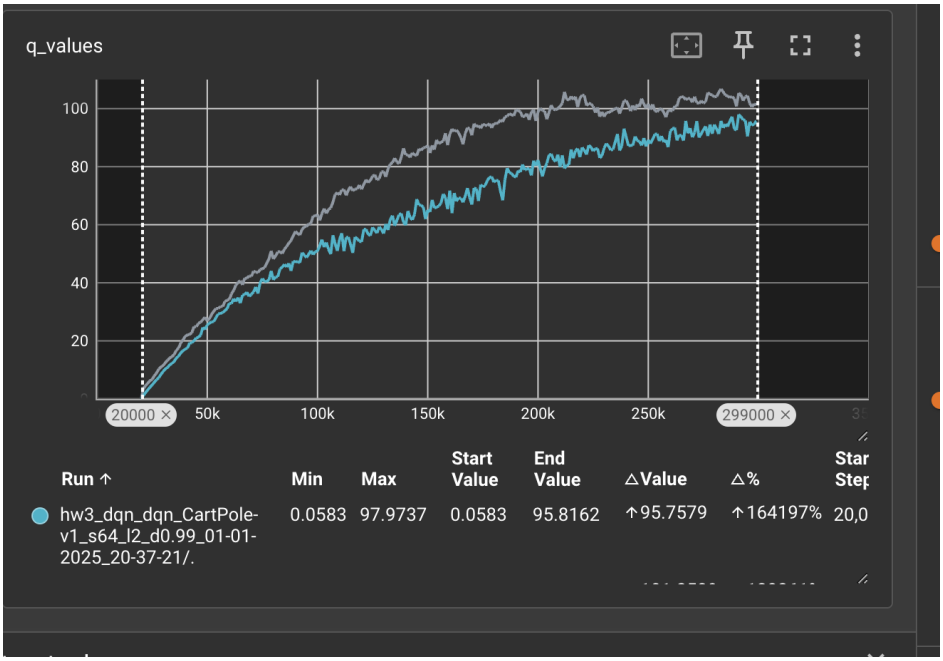

Figure 5: $lr = 0.05$ and q values

Explanation: The predicted Q-values increased because the higher learning rate led to more aggressive updates via the gradient function, trending toward favoring higher future rewards. However, the increased volatility caused higher variance in the predicted Q-values.

Since the Q-values became noisier, the Bellman error also increased. This is because the Bellman error depends on the estimation of future rewards using the Q-function for the target.

Higher variance in Q-values contributes to higher variance in the Bellman error. A high learning rate causes the neural network to overshoot the optimal values of the target critic Q-values, leading to increased critic error.

3 2.5

Figure 6: DQN and eval return

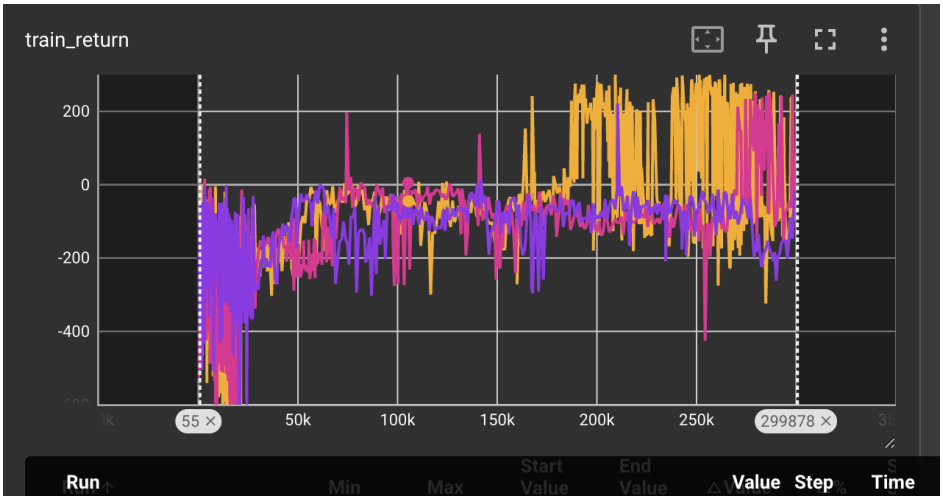

Figure 7: $lr = 0.05$ and q values

Here the double q learning seems to take longer to reach eval return around 200 compared to the vanilla approach since it could be possible that our two critics take longer to be trained to predict Q values compared to only needing to train one critic for both choosing actions and evaluating the q value for the selected action.

4 2.6

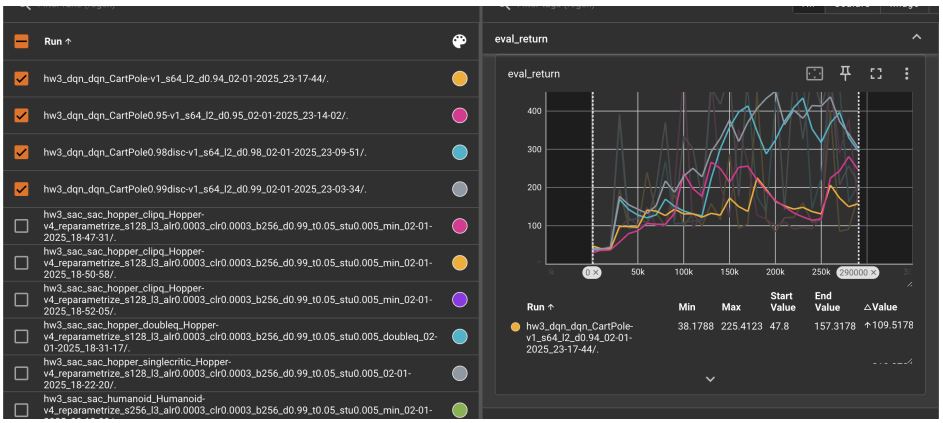

Figure 8: Discount factor vs eval return for cartpole

Impact of Discount Factor

The discount factor determines the extent to which the DQN agent prioritizes future rewards over present rewards, i.e., it affects the Q-value. A higher discount factor results in a higher evaluation return, as the DQN agent assigns more importance to future rewards, weighing them almost equally with current rewards.

By weighing future rewards higher, Q-value estimations align more closely with the target values, improving performance in tasks where long-term outcomes are critical. This leads to better policy learning that balances immediate and delayed rewards effectively.

5 3.1.1

```
python cs285/scripts/run_hw3_sac.py -cfg experiments/sac/sanity_pendulum.yaml
```

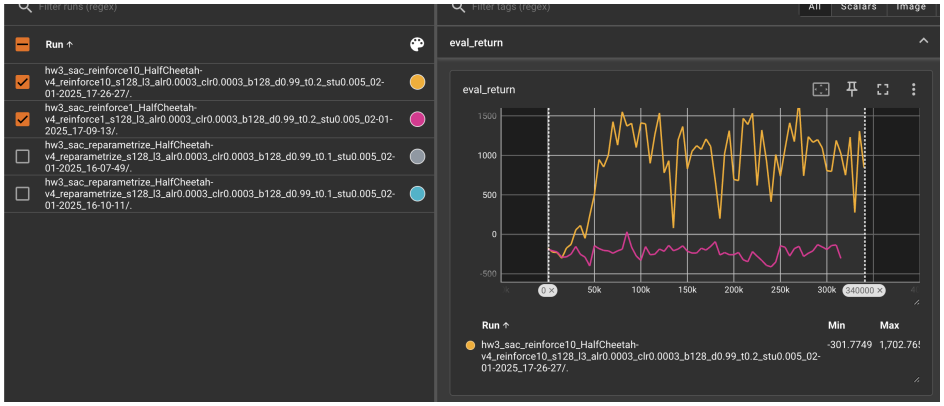

Figure 9: $lr = 0.05$ and q values

6 3.1.3

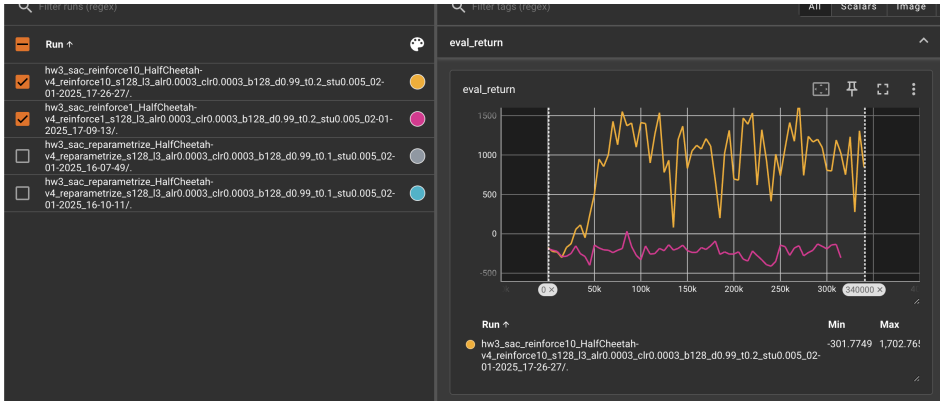

Figure 10: $lr = 0.05$ and q values

Return is significantly higher when using `Reinforce10` (yellow graph). Using only one sample to train the gradient introduces bias toward the training sample, causing the Q-values to deviate further from their expected values.

7 3.1.4

```
python cs285/scripts/run_hw3_sac.py -cfg experiments/sac/halfcheetah_reparametrize.yaml
python cs285/scripts/run_hw3_sac.py -cfg experiments/sac/sanity_invertedpendulum_reinforce.yaml
python cs285/scripts/run_hw3_sac.py -cfg experiments/sac/halfcheetah_reinforce1.yaml
```

Figure 11: $lr = 0.05$ and q values