Final Project

Paavo Camps - 755805
Guilherme Sales Santa Cruz - 1033746
ELEC-E8125 - Reinforcement Learning

December 5, 2021

1 Part 1

1.1 TD3

1.1.1 Question 1

One of the main problems with DDPG is that it often fails by heavily overestimating Q-values with the learned Q-function. To solve this problem TD3 proposes 3 modifications.

- Clipped Double-Q Network: TD3 learns 2 Q-functions simultaneously and uses the smaller estimate to compute the loss. This way it is less likely that the policy acts based on overvalued q-values that often lead to bad strategies with DDPG in which the policy acts on noise, which in turn leads to more variance in training.
- Delayed policy updates: the actor-network is only updated once every n-iterations while the critic is updated at every iteration. This way the calculated Q-Values are more accurate.
- Policy smoothing: TD3 uses some Gaussian noise to prevent the actor from exploiting errors in the Q-function.

1.1.2 Question 2

In Exercise 5 the algorithm was on-policy and, because of that, the value of each state was calculated by adding the discounted rewards of each step with the given policy. If the policy would be changed during the episode the previously calculated values would not reflect the same agent anymore.
With TD3, we evaluate the value of the state based on the value given by the critic, the action (given by the actor or sampled from a data set), the reward received after one step, and the value of the next state (also given by the critic). In this scenario, the policy does not influence the learning of the critic, so we can use either the actor's choice or not. In fact, for this model, it can be interesting to use random actions to increase the exploration

phase and discover potentially better policies and thus the reason why the noise is added to the policy.

1.1.3 Question 3

For the training plots of the TD3 algorithm check the figures 1, 2 and 3. Note that we measured the performance of the agent based on the value of the discounted rewards on the first step of the environment, instead of the accumulated rewards.

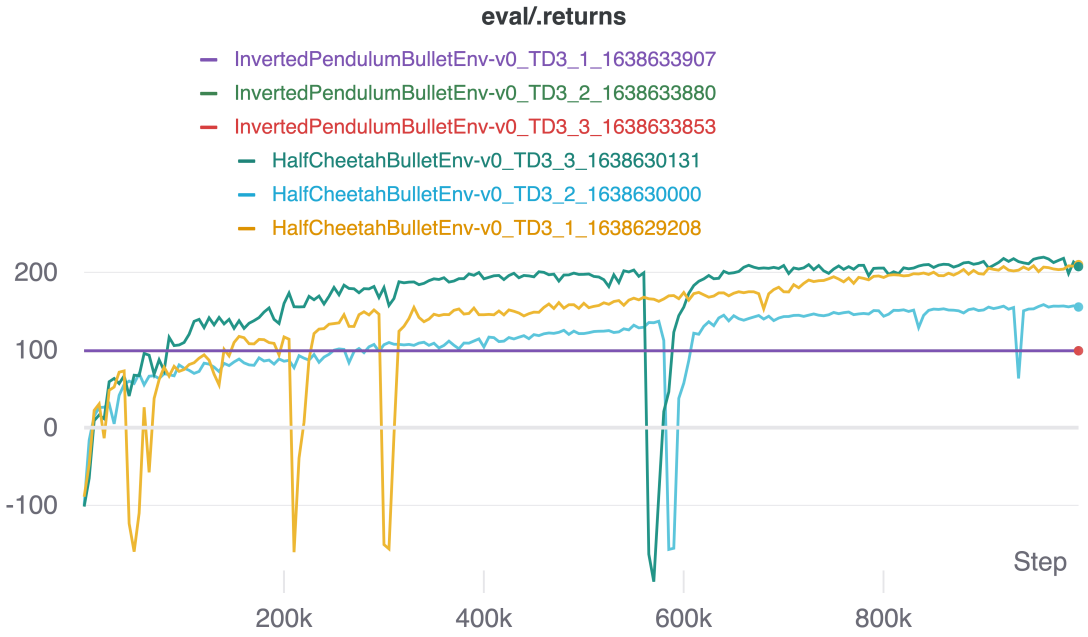

Figure 1: TD3: The value of the discounted rewards on the initial state following the policy actions

By observing the graphs we notice that the algorithm improves its performance during training. Also, in figure 3 the error of the critic stays constant for most of the training and reduces slightly at the end, which is expected since the policy is constantly changing and the critic has to keep up with it. There are some outliers in the graph which make it hard to observe the described behavior.

1.1.4 Question 4

For this task, we decided to increase the policy noise and the noise clip from 0.2 and 0.5 respectively to 0.4 and 0.8. The idea is that the noise is responsible for smoothing the policy approximation function but too much noise can be damaging to the performance of the algorithm, thus we decided to increase considerably these hyperparameters to observe the results.

For the training plots of the TD3 algorithm with modified hyperparameters check the figures 4, 5 and 6. Note that we measured the performance of the agent based on the value

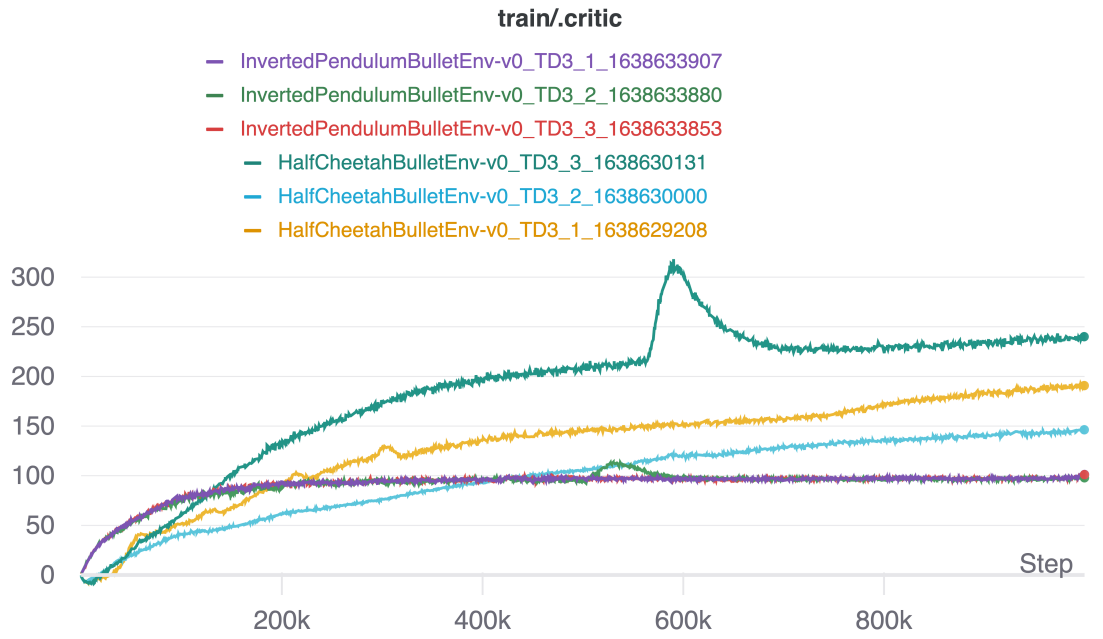

Figure 2: TD3: Critic performance during training

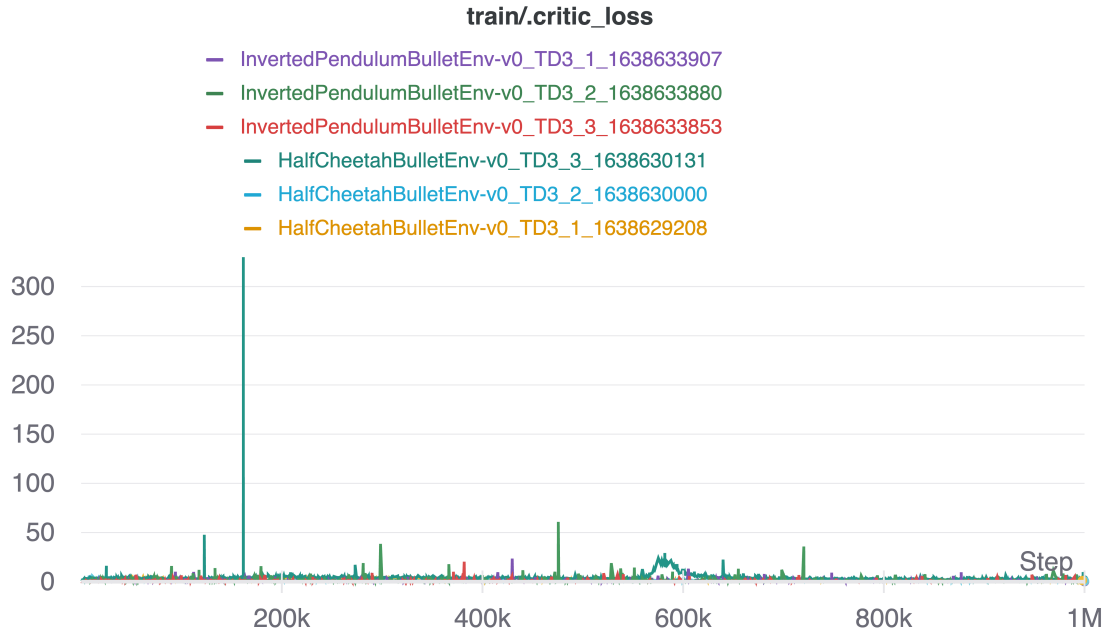

Figure 3: TD3: Critic loss during training

of the discounted rewards on the first step of the environment, instead of the accumulated rewards.

Observing the graphs and comparing them to the previous ones there is no noticeable difference. This result was unexpected as reinforcement learning algorithm hyperparameters are known for being very sensitive.

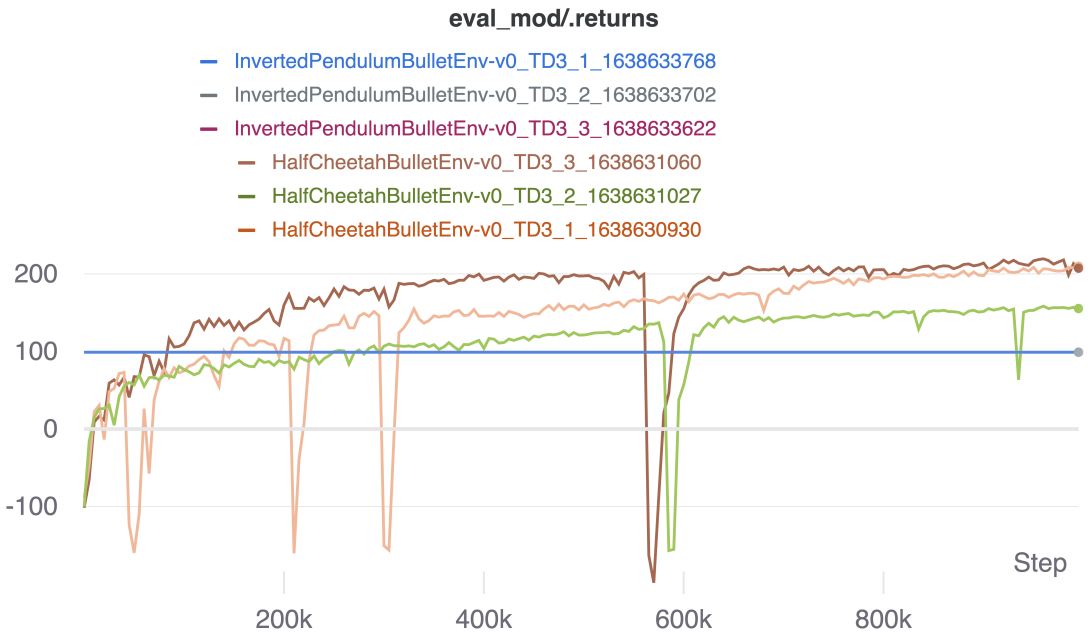

Figure 4: TD3 modified: The value of the discounted rewards on the initial state following the policy actions

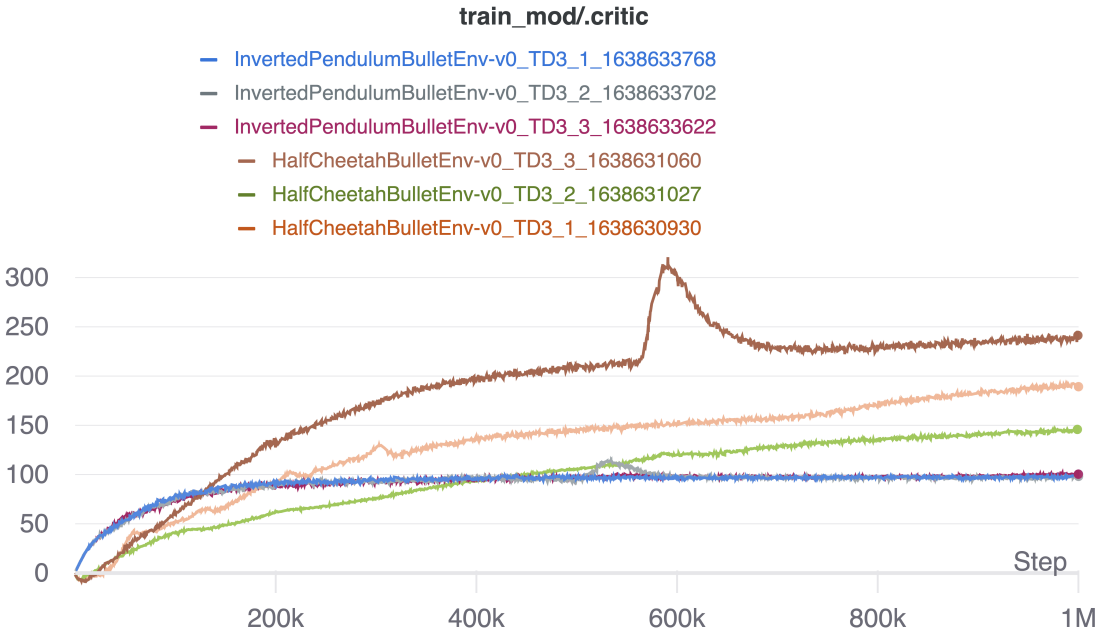

Figure 5: TD3 modified: Critic performance during training

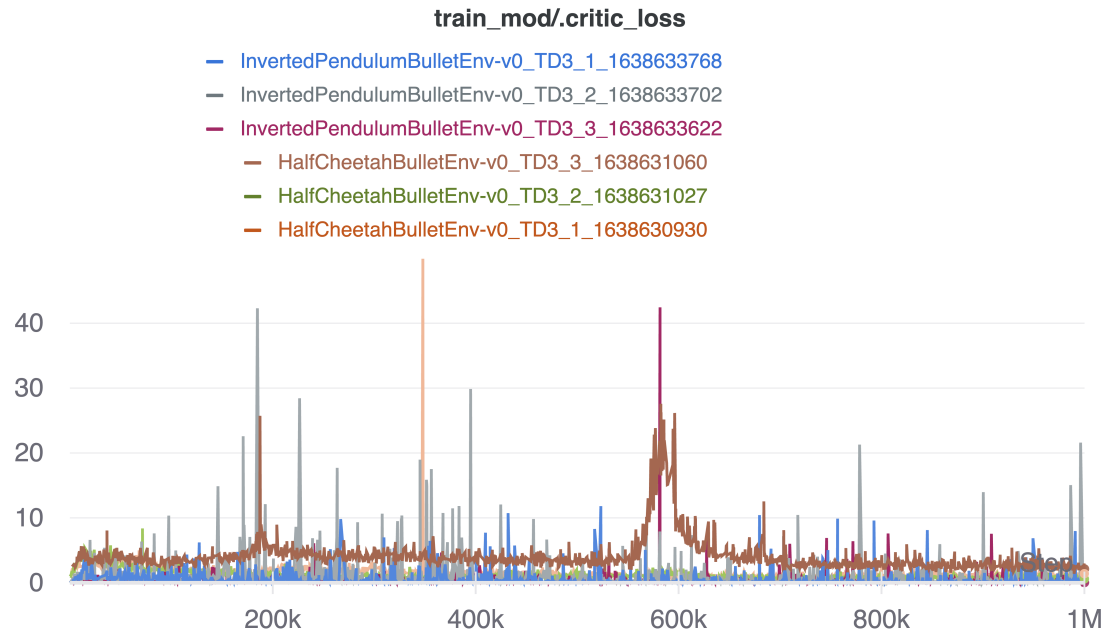

Figure 6: TD3 modified: Critic loss during training

1.1.5 Questions 5

The first possible improvement would be on how the agent train is delayed. There are other ways to delay the agent learning other than skipping the update for a few iterations. For instance, the agent could be a longer network than the critic or the agent's optimizer could have a lower alpha.

The exploration created by adding the Gaussian noise could have its variance exponentially reduced during the training for a better exploration-exploitation ratio.

Finally, one last improvement comes from a strategy used in other algorithms of prioritizing samples from the buffer based on the associated error. Like this, the critic can improve faster at states that are not well valued yet. This insight is similar to boosting strategies for traditional supervised methods.

1.2 PPO

1.2.1 Question 1

If the update is too "good", that is if it is a lot more likely after the update, the update will be clipped. This means that the agent's behavior does not change as drastically and therefore it is less likely that a one-of event changes the policy just because the value of a state is overestimated.
Similarly, the minimum operator makes sure that if the update reduces the likelihood then the update is only going to make it slightly worse not reduce it too much this way they can be rediscovered by the policy later when the value estimates around it change.
The TRPO objective is subject to the KL divergence(KL-divergence is used as a penalty to make sure that the new policy and old policy are not too far apart). The PPO objective is very similar except for the fact that it uses clipping instead of using the KL-divergence. PPO is therefore a simplified advancement of the TRPO algorithm.

1.2.2 Question 2

For the training plots of the PPO algorithm check the figures 7, 8 and 9. We had some problems with this implementation and these results were achieved on the last day, we did not have the time to run more experiments with other seeds.
Interestingly the training on the Aalto machines seems to have not worked as well as on our machine. We assume that it might be a tensor transformation that is handled differently in different versions of libraries.

The plots are not very intuitive as we would expect the variance of the returns to reduce as the entropy decreases but it doesn't happen here. We understand that the entropy is a measure of certainty of the agent regarding the chosen action and but it is not intuitive to

think that the agent is becoming certain of his policy when it has a wide range of results.

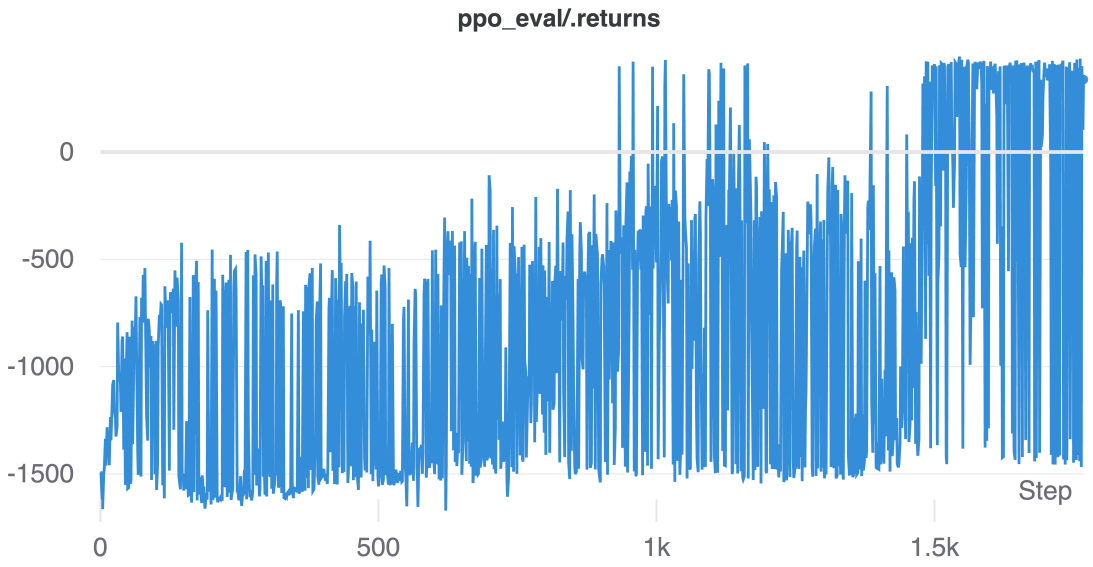

Figure 7: PPO performance over training

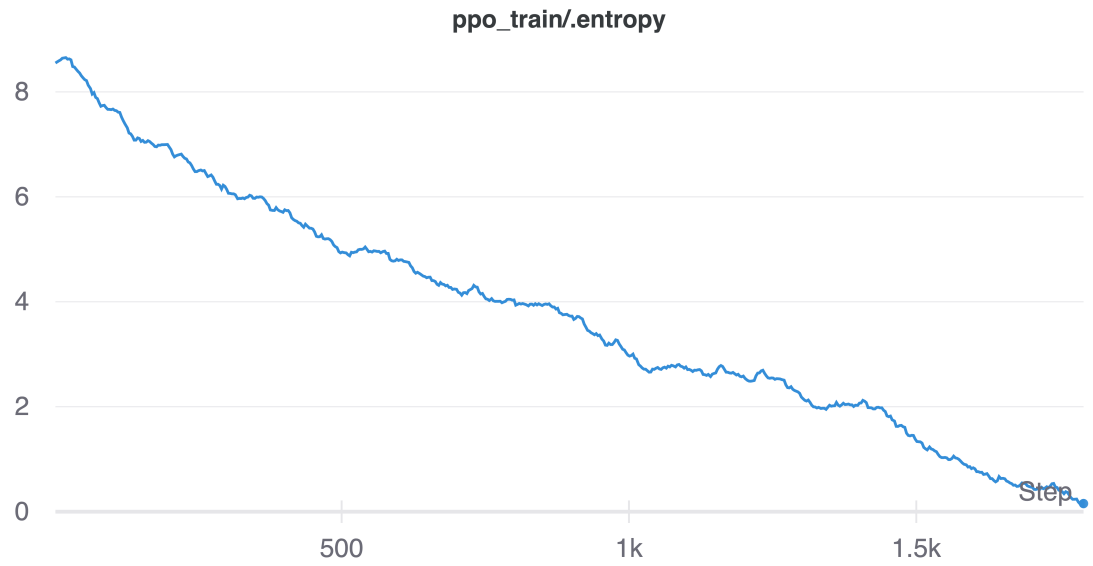

Figure 8: PPO entropy over training

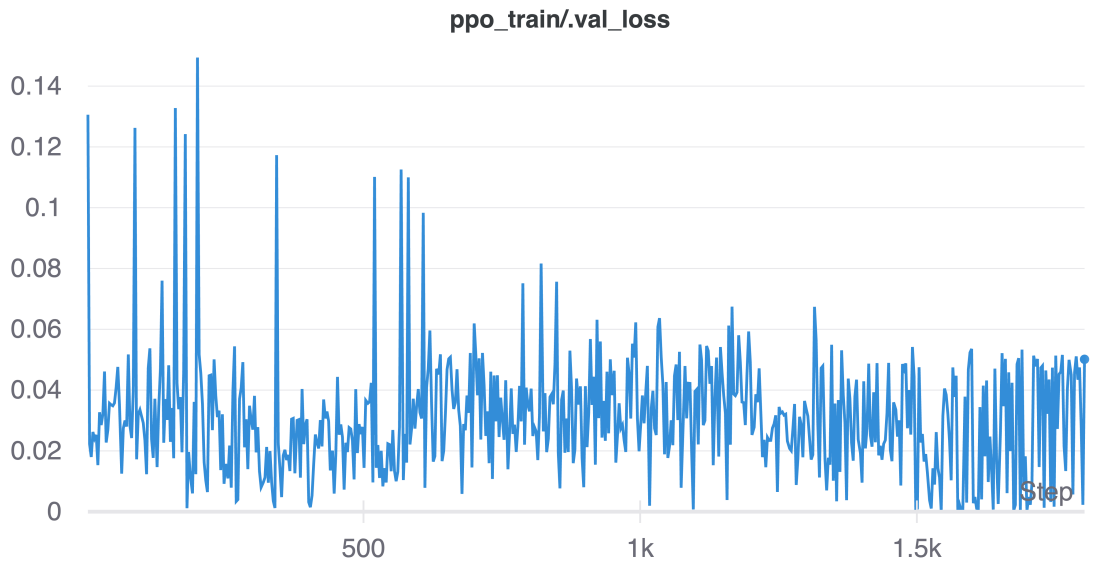

Figure 9: PPO value loss over training

1.2.3 Question 3

The GAE is structurally similar to the n-step advantage estimate. The difference is that the n-step advantage only uses the Value Estimation for the difference between the discounted last step and the first step. The GAE uses a weighted step advantage in each term.
The n-step advantage computes the advantage for the n next steps while the GAE computes the exponentially weighted mean of all advantages giving higher importance to the early advantages, for this a new hyperparameter lambda is introduced.
By increasing the number of steps in the n-step advantage estimation the bias is decreased because in the range of the steps all the rewards are correctly accounted for and the Value estimate is accounting for less of the total value. This is however a trade-off for an increase in variance since there are many more reward terms that influence the final evaluation.
When using GAE, the parameter lambda can be used to trade-off how impactful the value estimation should be, a low lambda leads to similar results than one step advantage estimate, whereas a high lambda value leads to a function that is similar to the n-step advantage function.

2 Part 2

2.1 TD3_BC

2.1.1 Question 1

For the training plots of the TD3_BC algorithm check the figures 10, 11 and 12. Note that we measured the performance of the agent based on the value of the discounted rewards on the first step of the environment, instead of the accumulated rewards.

This algorithm seems to converge much faster than its predecessor. However, it seems to converge at a lower value for our performance measure (the initial state discounted rewards value).

2.1.2 Question 2

For this question, we changed the design by playing with the way that the behavior cloning regularization term is weighted against the Q-values. In the original paper the λ term is computed by $\frac{\alpha}{\frac{1}{N}\sum_{(s_i,a_i)}|Q(s_i,a_i)|}$. The denominator scales the weight relative to the Q-values by taking the mean of the absolute values. We thought of the case where the Q-values are distributed crossing the y-axis, for instance as a bell-shaped curve, and we decided to scale by the mean of the distribution itself, instead of the mean of the absolute values. The downside of this new approach is that it does not take into account the variance of the distribution itself but neither does the original method.

For the training plots of the TD3_BC modified algorithm check the figures 13, 14 and 15. Note that we measured the performance of the agent based on the value of the discounted rewards on the first step of the environment, instead of the accumulated rewards.

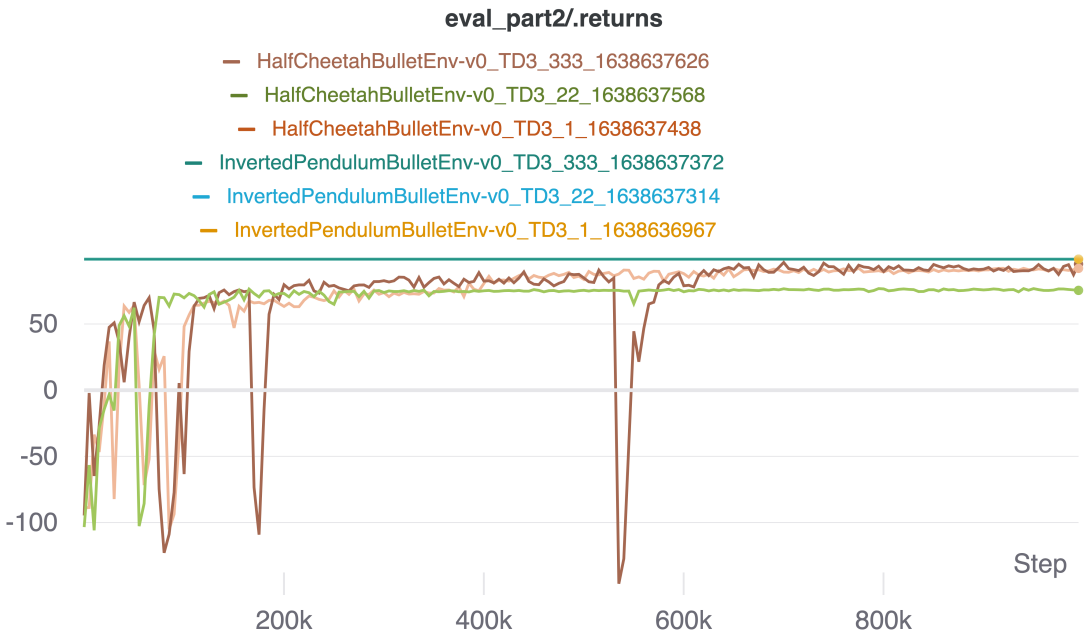

Figure 10: TD3_BC: the value of the discounted rewards on the initial state following the policy actions.

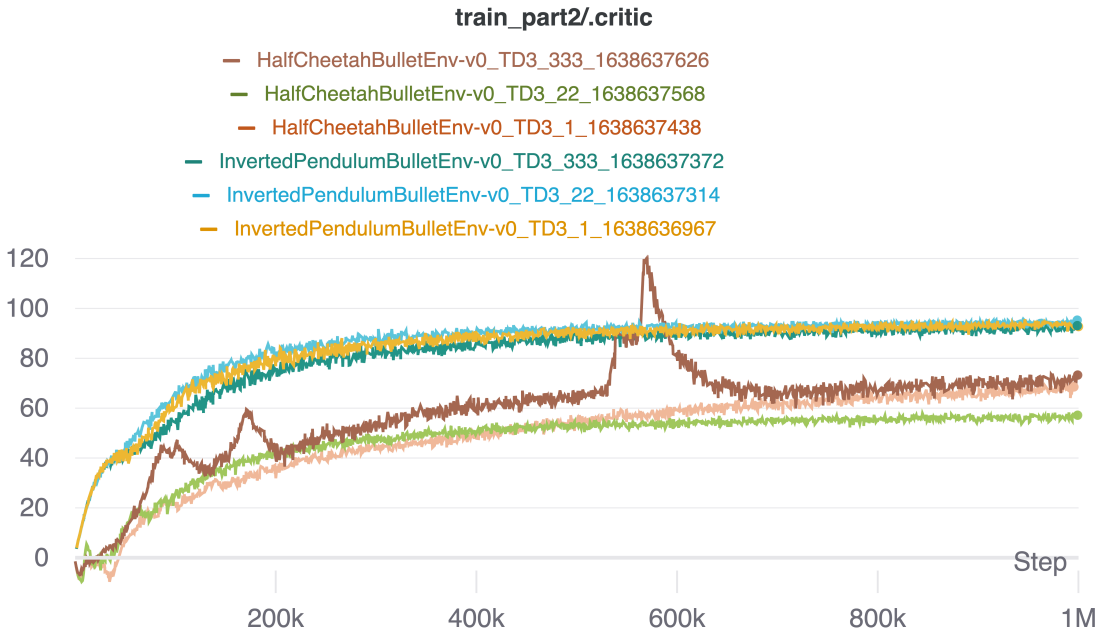

Figure 11: TD3_BC critic

The modification seems to have a strong impact on our performance measures when we compare figures 10, 13.

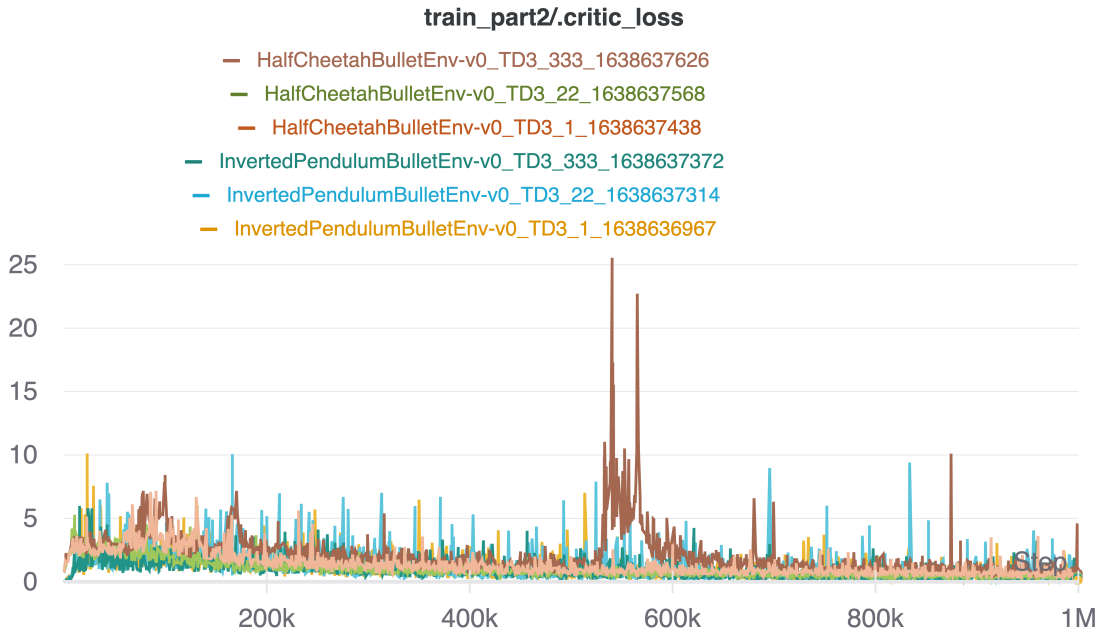

Figure 12: TD3_BC critic loss

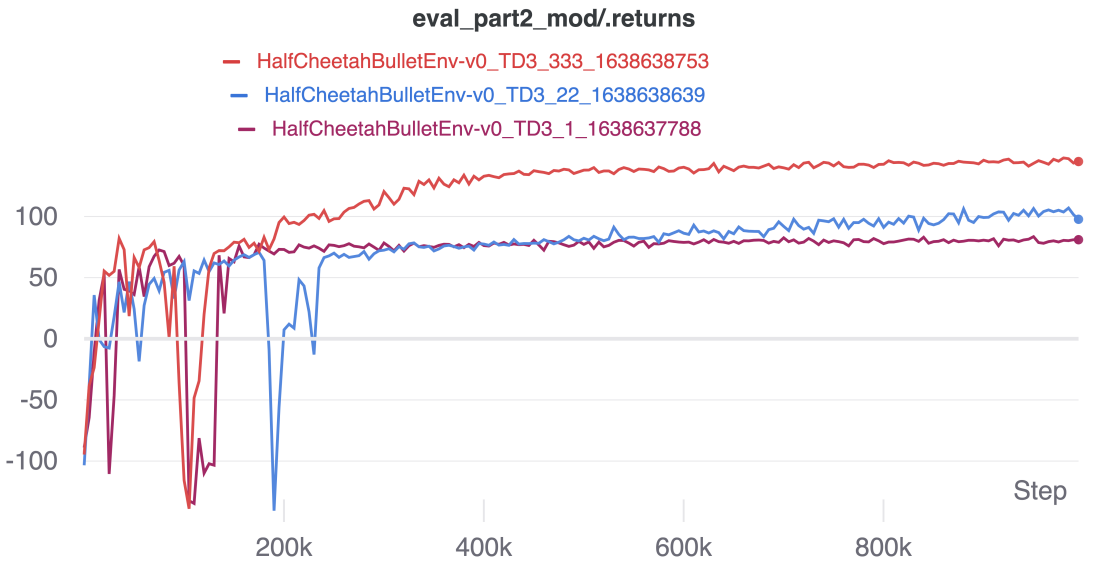

Figure 13: TD3_BC modified: the value of the discounted rewards on the initial state following the policy

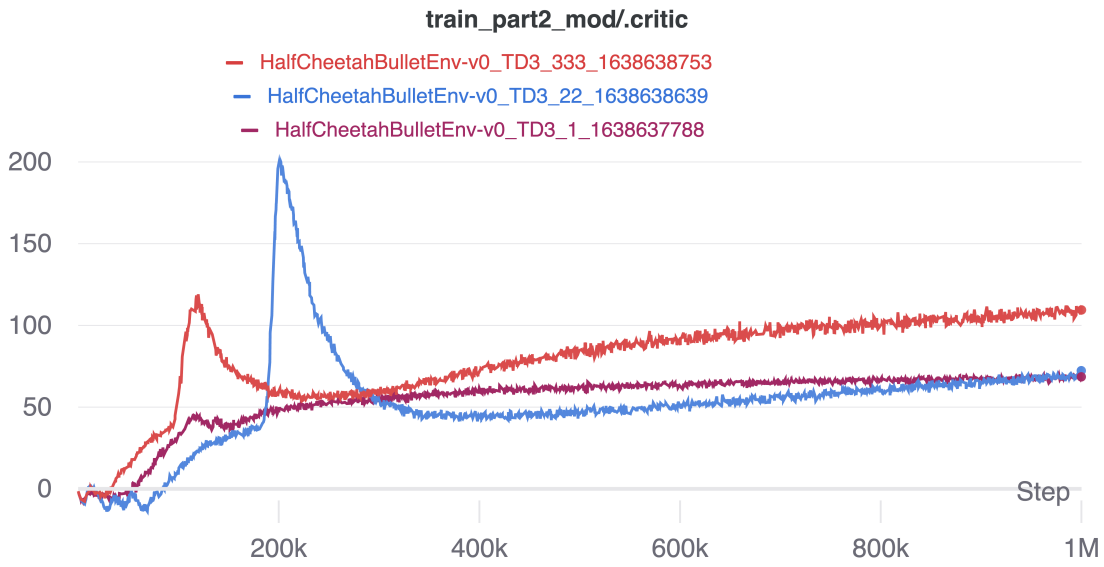

Figure 14: TD3_BC modified critic

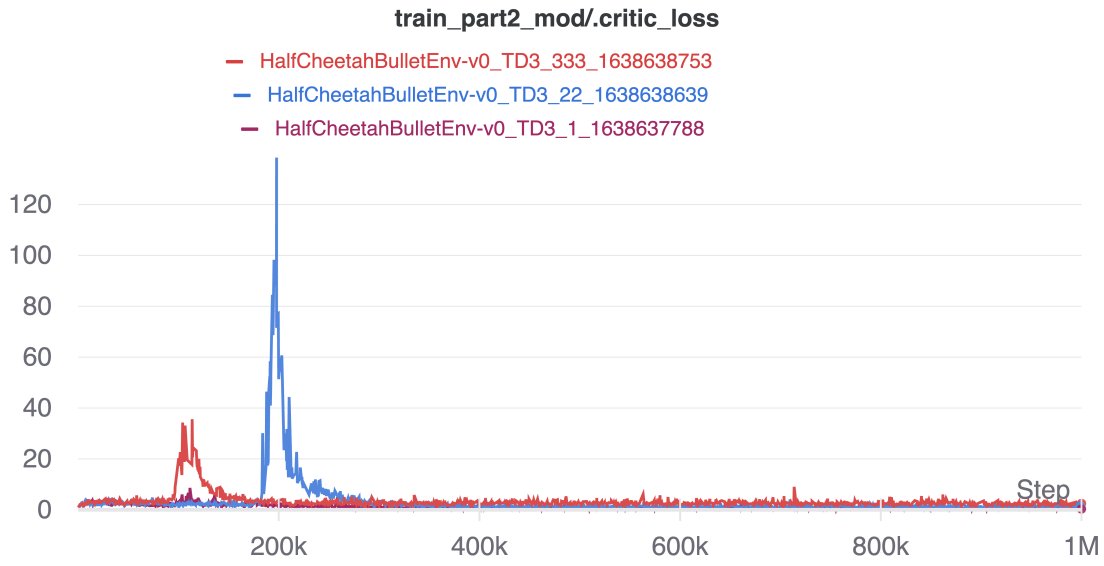

Figure 15: TD3_BC modified critic loss