ELEC-E8125 - Reinforcement Learning

Final Project

Ming Jiang - 1016239, Gengcong Yan - 1009903

December 4, 2021

1 Introduction

In this project, we implement TD3[1], PPO[2], as well as TD3_BC[3] to evaluate their performance in two environments: *InvertedPendulumBulletEnv-v0* and *HalfCheetahBulletEnv-v0*. The implementations are finished based on the given frameworks. We compare the advantages and disadvantages of these methods based on their designed architecture as well as the real performance. In addition, we explore the influences of some hyper-parameters and some unique architecture on the training performance.

The report is organized as follows: Section 2 briefly introduces the three methods we implement in this project. Section 3 shows the training results and answers the questions in Part-I-1 concerning on TD3. Section 4 displays the training plots as well as the answers to questions about PPO. In Section 5 we presents the training performance of our chosen method TD3_BC, and meanwhile answer the given questions in Part II. Finally, Section 6 concludes and discusses the whole project results.

2 Related Works

All of the three methods, TD3, PPO, TD3_BC are based on the **actor-critic** architecture. The original AC algorithm is an optimization of vanilla policy gradient, in which it uses critic network to updates the value function meanwhile uses actor network updates the policy parameters suggested by the critic. However, it suffers from some problems, such as the overestimation of Q value in the critic networks, and the update step length that is hard to control for the model to be trained smoothly and in the right direction.

Specifically, TD3[1] and TD3_BC[3] make some updates on the architecture of AC algorithm to alleviate the overestimation of Q value. They use a twin-critic networks architecture to separately estimate Q value and choose the minimum one each time. Meanwhile, to make the process more precise, they also add some additional noise when choosing the action to encourage more exploration in training. In addition, they also introduce the delayed update mechanism to update the actor several time steps later of critic update. This ensures to update the actor when the critic is more reliable.

Based on TD3, TD3_BC[3] makes some additional improvements. It provides a totally off-line training method of agent by giving the pre-collected dataset of a specific environment. This could reduce the time expense of data collection when the agent has to be interacted with environment. To compensate the bias that off-line dataset introduces, which includes

the incomplete trajectory space, TD3_BC simply add a behavior cloning term to regularize the policy as well as normalize the states over datasets. This makes the states distribution to have mean 0 and standard deviation 1, improving the stability of the learned policy.

It is worthy to note that both TD3 and TD3_BC are off-line methods by maintaining a replay buffer and each time sample the trajectories in the buffer for agent training. TD3 will first interact with the environment to collect the data while TD3_BC uses the off-line datasets. Instead, PPO[2] is a kind of on-line training method. It also maintain a buffer however would be reset when reaching a fixed size. This buffer pool in PPO helps to remove the correlation between the experiences. In addition, the highlight in PPO is the clipping ratio in objective function. It helps to alleviate the problem to control the update step between new policy and old policy. And the simple ratio is easy to calculate gradient compared with the KL divergence implemented in TRPO[4].

3 Part I-TD3

3.1 Q1

In DDPG algorithm, we use Bellman equation to learn the Q function. The value estimate is updated greedily looking for the maximum of values. There is an error between the maximum value used all the time and the true value. There may be only a small error in one update, but after a large number of updates, this error becomes impossible to ignore and leads to network instability. Continuously updating too large estimated Q values in the network may allows the network failing to learn the better policies.

TD3[1] introduces 3 new features to address the problem above, there are clipped double Q-Learning, delayed policy and target network updates, and target policy smoothing regularization.

1. Clipped double Q-Learning. Because the model will overestimate the Q values in networks, now we learn two Q functions in the critic networks, and we choose the smaller Q value for target computation, so that we can avoid overestimation. But always choosing a smaller Q value as target may also introduce the problem of underestimation.

2. Delayed policy and target network updates. In contrast to the Q function update, we reduce the frequency of updating the policy and target networks. This allows the policy network to be updated after the value network become more stable and the error was reduced a bit, so that the accumulated error is not transmitted to the policy network. However, aggressively reducing the update frequency will make model training converging in a longer time.

3. Target policy smoothing regularization. By adding a little noise to the target action, the variance of the model is reduced. This avoids model overfitting some extreme values in Q function update, resulting in a more stable performance. The addition of noise makes the action random over a range of area rather than a point, which means that the model is updated to a certain range around it, making the estimation more accurate and robust.

3.2 Q2

TD3 learns policy in an off-policy way, while policy gradient learns in an on-policy way. If TD3 agent learn policies in an on-policy way, probably because of random initialization, it was not able to explore enough rich actions at the beginning, may easily restrict to some actions, resulting in Q value not updated effectively, and the model not fully trained. When training in on-policy way, the model can only learn through feedback in current environment i.e. state, reward, action etc. But When training in off-policy way, the buffer is prefilled with samples and updated during training. There is no limit to any particular round of data when sampling from buffer, which the model has a larger sample utilization rate, resulting in more robust and higher performance..

3.3 Q3

Fig.1 and Fig.2 shows the training plots of TD3 in $HalfCheetahBulletEnv - v0$ and $InvertedPendulumBulletEnv - v0$. They are the mean of three random seeds.

Figure 1: Training plot of TD3 in HalfCheetahBulletEnv-v0

3.4 Q4

We made experiments on the influence of both hyper-parameter $update\_frequency$ and $policy_noise$. Fig .3 and Fig .4 show the influence of $update\_frequency$ and Fig. 5 show the influence of $policy\_noise$.

About the $policy\_frequency$, The default update frequency is set to 2, and we tested 1, 5 and 10. The training plot is shown in Fig.3 and Fig.4. When the update frequency grows too large like 5 and 10, the train process of model is extended. This is natural, as the weights of the actor network have become slower to update and we need more steps to achieve the previous results. But if we set update frequency to 1, which means we remove delayed updated feature. The model shows faster converge speed than default setting. We

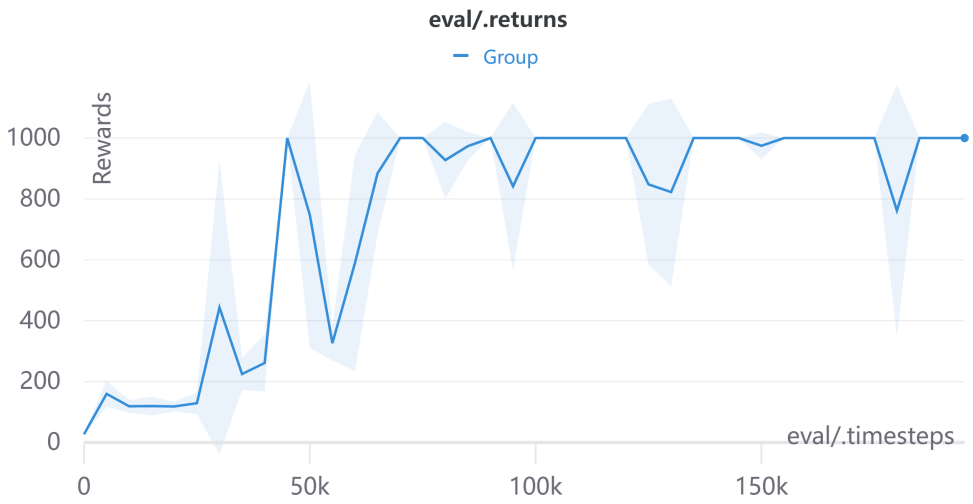


Figure 2: Training plot of TD3 in InvertedPendulumBulletEnv-v0

can only assume that this method played no significant role in the optimisation of TD3. However, it is speculated that an appropriately slower update frequency could save training time and also maintain the same performance.

As for the *policy_noise*, it controls the sample area of action. We could see from the figures that when we add more noise to the policy, it would be less possible for us to choose the action that maximizes the q value. As a result, there will be more exploration in the update when we increase the noise, thus lower down the speed of convergence.

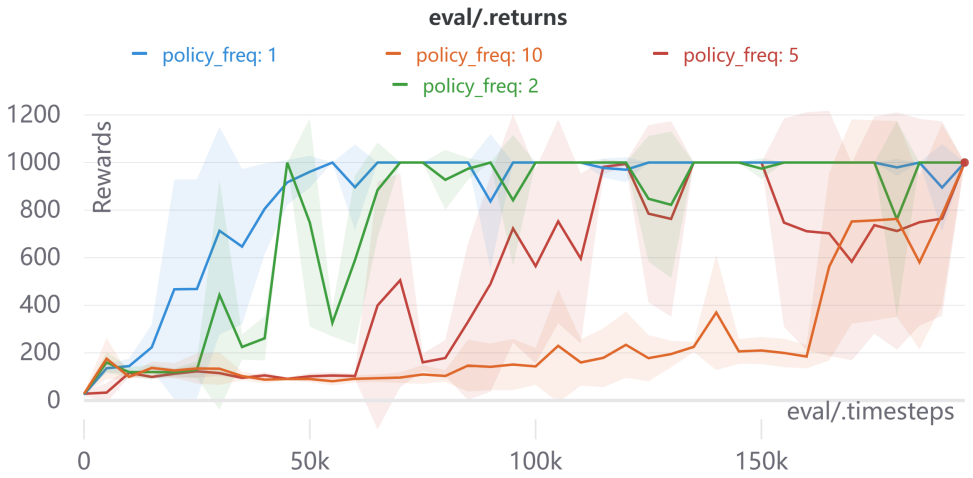


Figure 3: Training plot of TD3 changed by updated frequency in InvertedPendulumBulletEnv-v0

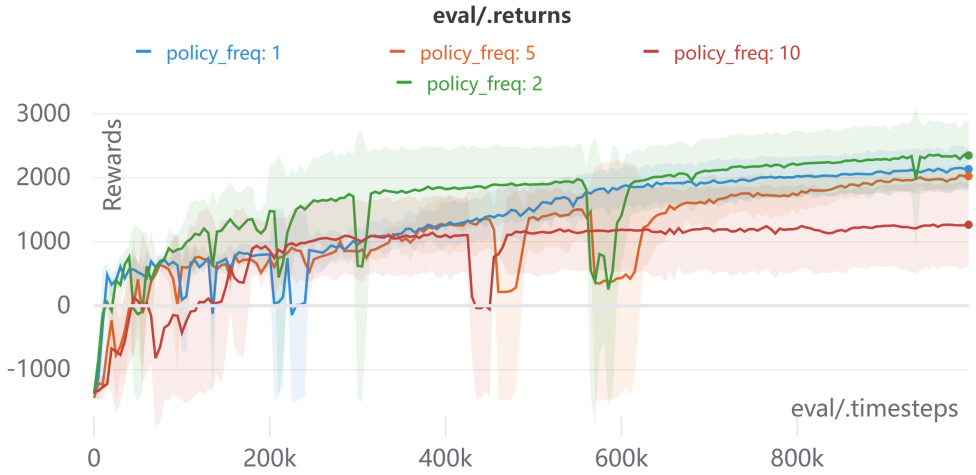
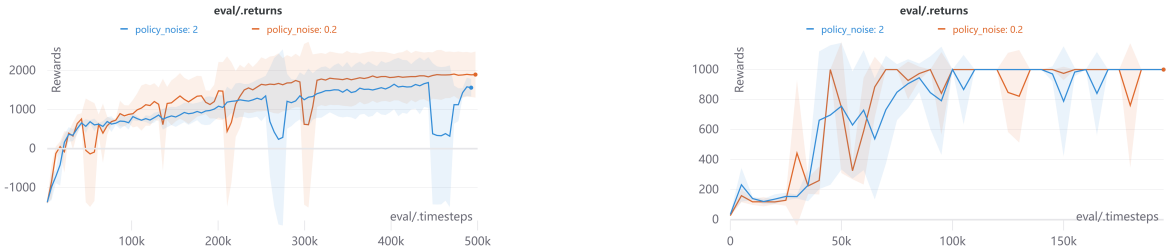


Figure 4: Training plot of TD3 changed by updated frequency in HalfCheetahBulletEnv-v0



(a) HalfCheetahBulletEnv-v0

(b) InvertedPendulumBulletEnv-v0

Figure 5: Training Plot of TD3 Changed by Policy Noise in Different Environments

3.5 Q5

1. Underestimation of Q value. We choose the smallest of the two target q value networks to avoid overestimation. But always choosing the smaller one may also produce smaller estimate. Always underestimating may also lead to the same cumulative error as overestimating during the update process.

2. Delayed frequency. As we did in the previous question regarding the comparison of policy update frequency, the parameter may not play a major role in TD3. And it is a manual hyperparameter, so we don't know in which range we will get the best results. It should perform better if it can be updated dynamically by the model at proper steps. But this is certainly not easy.

3. Directionally Uninformed. TD3 samples actions with equal probability in opposite directions from the current mean, which could be a wasteful way because we might have already explore the area in the action space that is similar to the current strategy, no more samples are needed.

4. **Possile Improvement:** As for the first disadvantage of TD3, Kamil, Quan et al.

provide a possible way of improvement[5] named OAC[1]. OAC implements an optimism principle by maximizing an upper confidence bound to the critic. It avoids the pessimistic underestimation in TD3 due to its utilization of an upper bound to determine exploration co-variance. Meanwhile, OAC alleviates the problem of overestimation by updating its target policy using a lower confidence bound of the critic. These two confidence bounds well deal with problem of overestimation and the pessimistic exploration.

4 Part I - PPO

In this part, we implement PPO and test its performance in two environments $Inverted-PendulumBulletEnv-v0$ and $HalfCheetahBulletEnv-v0$, same as what we have done in Section 3. In the end of this section, we attach the training plot comparison of TD3 and PPO in Fig. 8.

4.1 Q1

The clipping ratio in PPO[2] could alleviate the over fast update of parameters and the caused problem for the model to converge. It will control the ratio of new policy and old policy to be limited within $(1-\epsilon, 1+\epsilon)$. When advantages function is larger 0, it will restrict the optimization step to not be too large. While advantages function is less than 0, since we don't want to update too much on this "bad" direction, the step would be clipped at $1-\epsilon$ which is not a too large falling ratio.

In TRPO[4], instead of the clipping ratio the KL penalty is used to control the update step. This is realized by control the average KL divergence of the old policy and new policy.

4.2 Q2

Fig .6 and Fig .7 show the training plot of PPO in two different environments. Each figures is the mean plot of three random seeds.

4.3 Q3

n-step advantage extends the TD(0) estimator for extra time steps, while GAE is the weighted average of several TD(0) estimator for advantage function. It would help to alleviate the bias-variance trade-off when estimate the advantage function. If we simply use the TD(0) estimator, it will have low variance however high bias. The n-steps advantage function reduces the bias by extending TD(0) to n-steps estimator, however causing the variance to be larger. GAE alleviate the bias-variance trade-off by doing a weighted average on the several TD errors.

[1]URL: https://arxiv.org/pdf/1910.12807.pdf

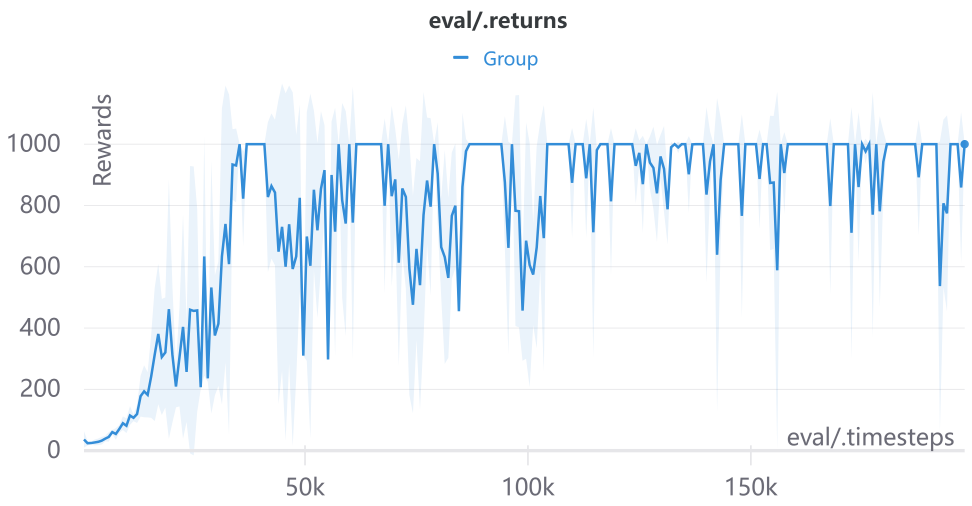


Figure 6: Training plot of PPO in InvertedPendulumBulletEnv-v0

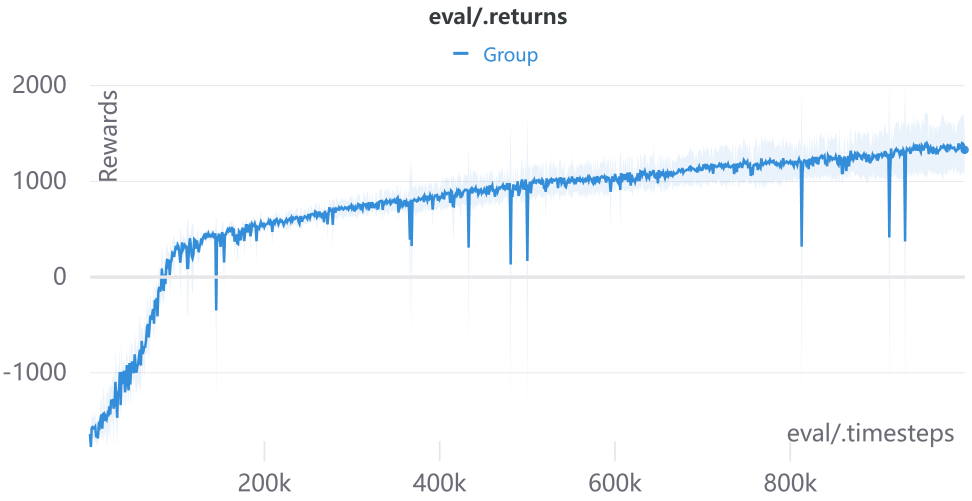


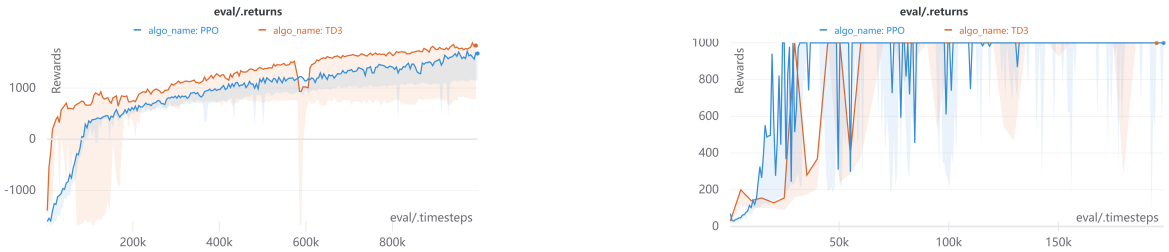
Figure 7: Training plot of PPO in HalfCheetahBulletEnv-v0

5 Part II - TD3_BC

Here we choose TD3_BC for exploration.

5.1 Q1

It shares the similar architecture with TD3. That is, it utilizes two critic networks to respectively calculate the q values, q1 and q2. We select the minimum q value as the target to update the two networks. As for the actor network, the update will not be synchronised with the critic part. Instead, it will be delayed to update when we have already updated the critic several times. This mechanism would help the actor to be updated more smoothly after the critic becomes more stable. In total, there are 6 networks. They are 1 actor network, 1



(a) HalfCheetahBulletEnv-v0 (b) InvertedPendulumBulletEnv-v0

Figure 8: Comparison of TD3 and PPO in Different Environments

actor target network, 2 critic network and 2 critic target network which are the same as the networks in TD3. There are 3 layers with output size of action dimension in actor networks. There are 3 layers with output size of 1 (Q value) in critic networks.

And there are also some special tricks in TD3_BC. We add a behavior cloning regularization when updating the policy network. We want to reduce the difference between policy and real actions in the dataset. This is an offline algorithm, so we don't have to interact with the environment and can only use the samples in the dataset for training. So the model needs a large enough dataset containing enough situations to expose model in different samples and converge to a more general strategy rather than just focusing on a small part of the samples.

About the training process. TD3_BC is trained on the offset dataset collected before without interacting with environment. We test the model in different data size: 100%, first 50%, first 25% and first 10%. After loading the proper size of data, we fill them in *replay buffer* and then normalize all samples. Now, we can begin the training process by feeding each sample from *replay buffer* every timestep. Table 1 list all hyperparameters in the model. Compared to TD3, the addition hyperparameter is **alpha** in use of controlling the balance of Q value and the behavior cloning regularization. $\alpha = 2.5$ is applied in original paper[3], so did in our setting. **Ratio** means the size of dataset.

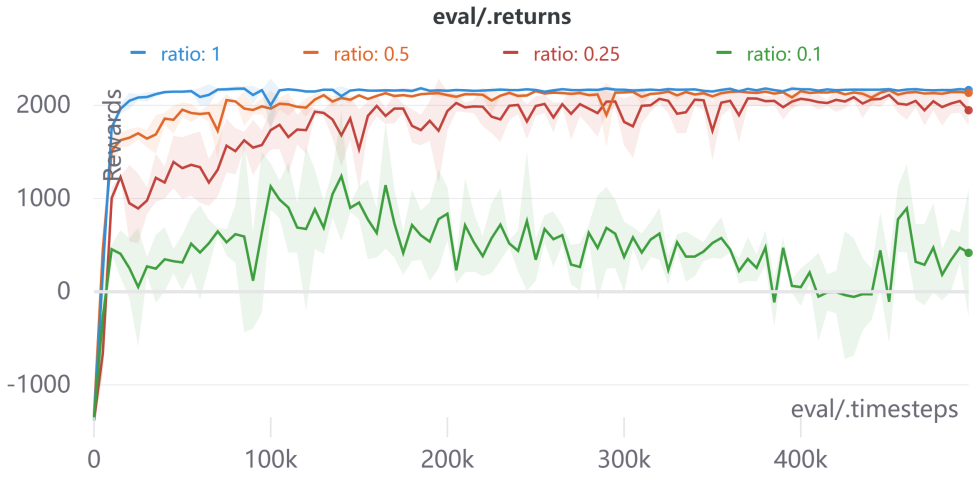


Figure 9: Training Plots of TD3_BC in HalfCheetahBulletEnv-v0 with Different Off-line Data Fraction

Hyperparameters	Value
–algo_name	TD3_BC
–env	HalfCheetahBulletEnv-v0
–max_timesteps	500000
–n_random_timesteps	10000
–batch_size	256
–eval_freq	5000
–expl_noise	0.1
–discount	0.99
–tau	0.005
–policy_noise	0.2
–noise_clip	0.5
–policy_freq	2
–alpha	2.5
–ratio	1.0
–seed	1
–save_model	store_true

Table 1: Hyperparameters of TD3_BC

See Fig.10 for the comparison of TD3_BC and TD3 in HalfCheetahBulletEnv-v0 environment. We only run 500, 000 timesteps for time saving. Fig.9 shows the training plots of TD3_BC with different off-line dataset fraction.

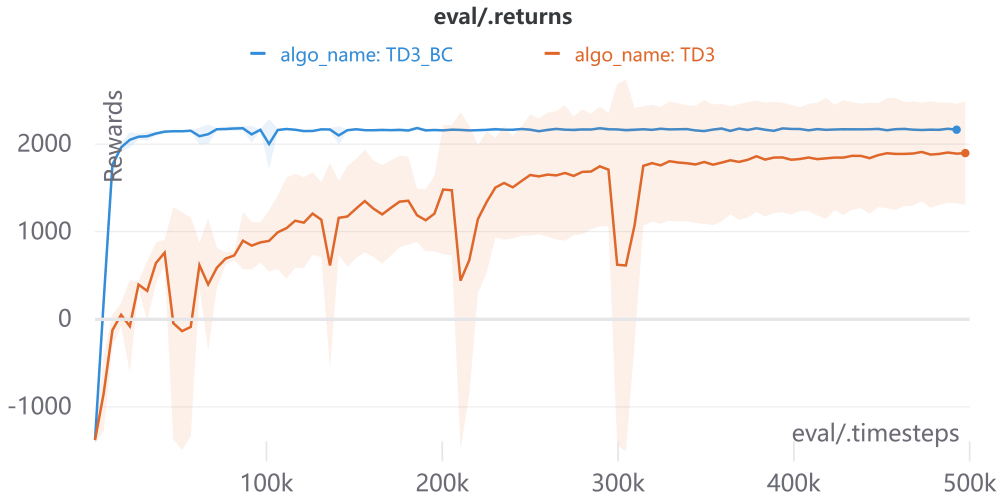


Figure 10: Training Plots of TD3_BC and TD3 in HalfCheetahBulletEnv-v0

5.2 Q2

In this part, we investigate the influence of the twin-critic-networks architecture on the training performance of TD3_BC. We mask this mechanism by change the critic network back to a single one, instead of choosing every time the minimum q value of two critic networks and check the performance. Fig .11 shows the comparison of TD3_BC and TD3_BC with Single Critic Network in HalfCheetahBulletEnv-v0.

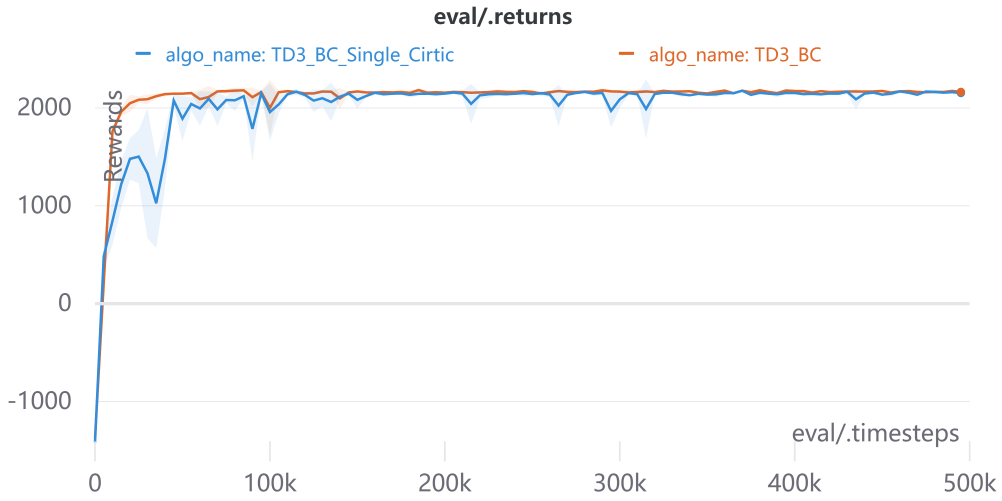


Figure 11: Training Plots of TD3_BC and TD3_BC with Single Critic Network in HalfCheetahBulletEnv-v0

From the figure, we could first drive the conclusion that the two separate critic networks do have positive influence on TD3_BC's performance. It speed up the convergence as well as decrease the deviation of the returns, making the training process more fast and stable. We think this is because that it helps to reduce the accumulation of update bias and avoid the overestimation of Q value, meanwhile reduce the possibility of get struck in a sub-optimal strategy.

6 Conclusion

In this project, we implement TD3, PPO, and TD3_BC and check their performance in two different environments. The twin critic networks in TD3 helps it to improve the over-estimation of Q values, however also introduce the problem of underestimation. Also, the noise added in action choosing process encourages the exploration but also add some bias in training process. This two factors might influence its convergence to be not so fast and stable compared with PPO. In TD3_BC, the utilization and processing of off-line dataset collected before agent's interaction with environment improves the efficiency in training. But the performance would highly depend on the quality and quantity of the pre-collected data.

References

[1] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International Conference on Machine Learning*, pp. 1587–1596, PMLR, 2018.

[2] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[3] S. Fujimoto and S. S. Gu, "A minimalist approach to offline reinforcement learning," *arXiv preprint arXiv:2106.06860*, 2021.

[4] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, pp. 1889–1897, PMLR, 2015.

[5] K. Ciosek, Q. Vuong, R. Loftin, and K. Hofmann, "Better exploration with optimistic actor-critic," *arXiv preprint arXiv:1910.12807*, 2019.