Final Project Report

By
Mostafa Dadkhah (261097757)
Andreas Enzenhoefer (260701043)
Garrett Kinman (260763260)
Tooba Rahimnia (260714516)

As part of the course
COMP579: Reinforcement Learning

Submitted to
Prof. Doina Precup

School of Computer Science
Faculty of Arts and Science
McGill University

April 30, 2022

1 Introduction

For this project, we selected the MuJoCo Hopper task. The setup uses the MuJoCo physics simulator and the OpenAI Gym of reinforcement learning (RL) environment [1]. The objective is to make a one-legged robot (with 3 joints and 4 body parts total) hop forward as quickly as possible without falling over. The RL problem contains an 11-dimensional, continuous-valued state vector (representing positions, velocities, joint angles, and so forth) and a 3-dimensional, continuous-valued action vector for controlling the 3 joint actuators.

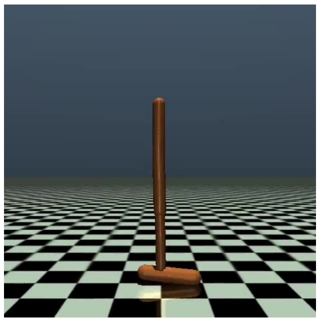
Figure 1: View of the Hopper within the MuJoCo environment

The continuous-valued nature of the state and action spaces naturally leads to the concept of value function approximation (and other related concepts). A continuous state and action space means infinite potential states and actions, i.e., no computer can possibly create a one-to-one map of states (or state-action pairs) to values. Thus, deep RL methods were applied. To do this, a literature review on deep RL methods was performed, as it is a broad area containing many potential methods. Through this, four specific algorithms were identified, which were then implemented, trained, and tested.

In the rest of this report, further details will be given on the methods, approaches, and algorithms used. Key results and comparisons amongst the methods will also be shown. The best-performing agent will be selected and tested on a public leaderboard. Its results will be shown as well.

2 Methods

Perhaps the most well-known deep RL algorithm is the Deep Q Network, or DQN. However, a key limitation of DQN in the context of this project is that it does not produce continuous-valued actions even though it accepts continuous-valued state space [6]. Thus, four other algorithms, found via literature review, were used in this project: the Deep Deterministic Policy Gradient (DDPG), the Twin Delayed Deep Deterministic Policy Gradients (TD3), the Soft Actor-Critic (SAC), and the Truncated Quantile Critics (TQC).

2.1 Deep Deterministic Policy Gradient (DDPG)

The first algorithm employed was the DDPG algorithm based on the Deterministic Policy Gradient (DPG) [6]. This actor-critic algorithm simultaneously trains a deep neural network critic and actor to learn a Q-function approximation and a policy, respectively [6].

This method was chosen because of its relative simplicity, which allows it to act as a good baseline for comparison with the following two methods. Additionally, the DDPG algorithm's designers reported that it performs well with low-level features such as Cartesian coordinates and joint angles [6], which applies to the Hopper task in this project.

Using this algorithm still leaves the actor and critic architectures—as well as other hyperparameters—to be decided. For DDPG, the critic architecture used was a 2-hidden-layer fully-connected network with ReLU activations. The input was a vector of length 14, representing the state-action pair, with 11 values for the state and 3 for the action. The actor architecture was also a 2-hidden-layer fully-connected network with ReLU activations. However, the input was only of length 11 (for the state vector), and the output was of length 3 (for the action vector). A discount factor of $\gamma = 0.99$ was also used. Additionally, a small amount of Gaussian noise was added to the action vector, with a standard deviation of $\sigma = 0.3(1 - \tanh(\frac{timestep}{3.5 \times 10^5}))$. The purpose of this was to add an element of randomness to the action selection—which would slowly decrease with time—to encourage early exploration.

2.2 Twin Delayed Deep Deterministic Policy Gradients (TD3)

The second algorithm employed was the TD3 algorithm based on DDPG. However, TD3's key difference is two separate critic networks trained to estimate the Q-function [2]. When training the actor network to learn a policy, the lowest Q-value from the twin critic networks is used [2]. The intuition behind this step is that the algorithm is being pessimistic about its Q-value estimates, especially early in training when the estimates are poor, so as to avoid early exploitation of erroneously high Q-values.

This method was chosen for three reasons. First, its relation to DDPG provides an interesting comparison, where TD3 intuitively seems the more robust algorithm. Second, it inherits DDPG's strengths—i.e., good performance with low-level features such as Cartesian coordinates and joint angles, as described in the previous section. Finally, the designers of the TD3 algorithm reported that TD3 outperformed the state-of-the-art algorithms on every OpenAI Gym task [2].

In terms of actor and critic architectures and hyperparameters, these were the same as for DDPG above, with the exception that TD3 was given a twin critic of the same parameters. This was done because TD3 is so closely related to DDPG, adding only the twin critic network to DDPG, and so as to isolate variables as best as possible. The same element of Gaussian noise was also added to the action vector as for DDPG.

2.3 Soft Actor-Critic (SAC)

Like DDPG and TD3, SAC is also an actor-critic algorithm, but with a key difference from both: instead of maximizing for cumulative reward, SAC seeks to maximize both cumulative reward and cumulative entropy [3]. This is rooted in the Principle of Maximum Entropy, which states that the probability distribution that best represents the current state of knowledge about a system is the one with largest entropy [4]. Prioritizing both reward and entropy at the same time is supposed to help avoid and escape local maxima, thus, facilitating convergence [3]. The SAC algorithm can also be thought of as learning to perform well while also acting as randomly as possible [3].

This method was chosen because is an actor-critic algorithm, like DDPG and TD3, with a similar mechanism of operation, differing in its leverage of the field of maximum entropy RL. This gives an interesting point of comparison with the other two algorithms. Additionally, it was chosen because it is considered a state-of-the-art algorithm for continuous control, such as for the task in this project. Indeed, like with TD3, the algorithm's designers outperformed existing state-of-the-art algorithms in benchmark tasks, and SAC also exhibited strong invariance to random initialization [3].

The core actor and critic network architectures used for SAC for this task were unchanged from DDPG and TD3, both because of the similarity of the algorithms and to limit the number of independent variables. Pre-tuned hyperparameters for SAC on the Hopper task were also discovered during literature review and utilized here [7].

2.4 Truncated Quantile Critics (TQC)

The final algorithm employed was the TQC algorithm. This algorithm, like TD3, is also an actor-critic method which attempts to improve performance by reducing erroneously high Q-value estimations [5]. However, the approach in TQC is three-pronged. Like TD3, it also uses ensembling of critic networks, but, in addition, TQC also uses a distributional representation of a critic and truncates the critics' predictions [5]. In the distributional representation, adopts distributional RL ideas, and it treats the reward as a distribution instead of as a mere scalar. It then approximates the reward distribution by discretizing it into a finite number of "atoms", as the authors describe it, each having a weight and a location [5]. The truncation is performed by removing the atom with the largest location (i.e., removing the most optimistic atom) [5]. Finally, the ensembling, unlike with TD3, is generally performed with more than just two critics [5].

TQC was chosen because it is an actor-critic method with strong similarities to the preceding algorithms, particularly TD3 and DDPG. In addition, it is the newest and most complex algorithm, and it outperforms all three preceding algorithms on baselines [5].

The same actor and critic architectures were used for TCQ as for the three preceding algorithms, with the only changes pertaining to the TQC-specific parts, such as the ensemble critic networks. The hyperparameter for number of top quantiles to drop (the truncation part) was set as 5.

3 Results

Two agents with different seeds were trained for 1.5 million iterations for each method. One challenge was the long runtimes from unoptimized, unvectorized, and unparallelized code used at first, which could be 7 to 8 hours for each run. With the TQC and SAC algorithm implementations that use vectorization and parallelization, runtimes decreased to 3 to 4 hours per run. Since only one trained agent could be submitted to the leaderboard at a time, choosing the best agent was of great importance. The cumulative reward was recorded for each agent during training, and the submitted agents' leaderboard submission results as well as baselines were recorded.

3.1 Cumulative Rewards of the 3 Approaches

Figure 2 shows the combined plots of the three agents' cumulative rewards versus the time step. This includes for two training runs for two different seeds, where the lines represent the mean of the runs for each agent, and the shaded regions represent the two raw runs for each agent. Note that TD3, SAC, and TQC all significantly outperformed DDPG, with SAC and TQC performing slightly better than TD3 as well. This matches the expectation, as TD3, SAC, and TQC are newer algorithms that were designed to improve upon algorithms such as DDPG. These results also corroborate the theoretical basis for having selected these algorithms to test: for TD3, in taking the pessimistic Q-values; for SAC, in maximizing for both cumulative reward and entropy; and for TQC, also in being pessimistic about high Q-values.

3.2 Leaderboard Performance

The true test of the algorithms would come from the leaderboard, however. Both DDPG and TD3 agents were able to be submitted to the leaderboard, although at different times. However, due to time constraints, the trained SAC and TQC agent were never submitted to the leaderboard. Included in Table 1 are the evaluated results from the DDPG agent, the TD3 agent, a baseline agent provided by the TAs, as well as a random agent. All were evaluated based on performance (i.e., how fast the Hopper can be made to hop) and sample efficiency (when retrained on the leaderboard server using randomly initialized weights). At the time of submitting this report, the TD3 agent constitutes the final submission, placing into 9th place for performance and 2nd place

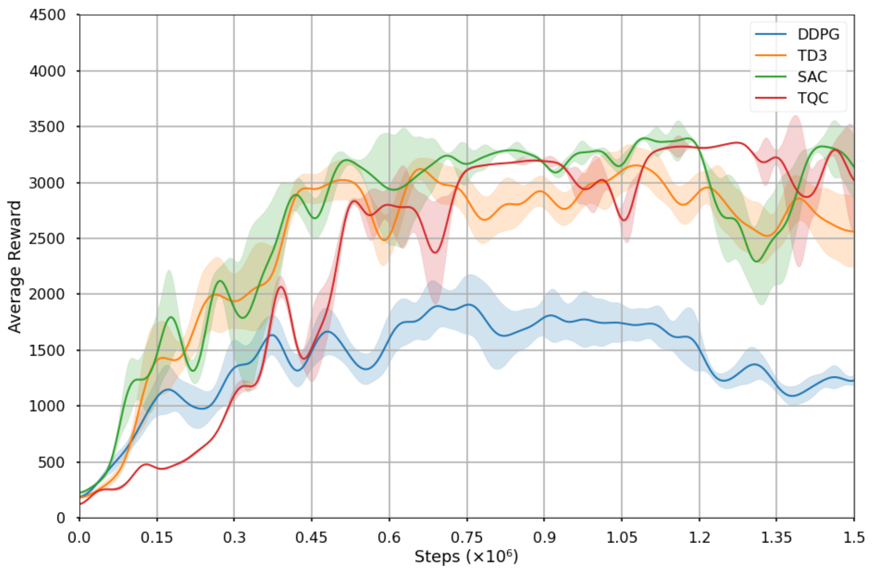

Figure 2: Combined plot showing cumulative rewards for all three algorithms

for sample efficiency. While the performance is nothing particularly high amongst the other teams' submissions, the sample efficiency is quite high. This is suspected to be because of the random noise added to the action vector in this reports TD3 agent. This noise was observed to improve learning speed, and it serves a similar theoretical role as the entropy in SAC or truncation in TCQ or even the pessimistic twin critic already in standard TD3: it increases exploration early on, rather than trying to exploit an erroneously high Q-value.

Agent	**Performance**	**Sample Efficiency**
TD3	3784.46	498.72
DDPG	3465.90	366.10
Baseline	1000.00	25.00
Random	17.32	3.81

Table 1: Leaderboard Performance Comparison

4 Conclusion

In testing four actor-critic deep RL algorithms on the Hopper task—DDPG, TD3, SAC, and TQC—the latter three were found to all perform better than DDPG. This matches expectations, given that they are all three newer algorithms, as well as being derivatives of it in some fashion. In this project's final leaderboard submission, the TD3 algorithm—with addition of Gaussian noise to the action vector during training—proved exceptionally good in terms of sample efficiency. This supports the general theoretical underpinning of all three better-performing algorithms that exploring instead of exploiting early on leads to better results. Due to time constraints, however, SAC and TQC agents were not able to be submitted to the leaderboard; this would be an interesting additional result to see how these algorithms compare in ultimate performance and particularly in sample efficiency.

References

[1] Greg Brockman **andothers**. *OpenAI Gym*. 2016. eprint: `arXiv:1606.01540`.

[2] Scott Fujimoto, Herke Hoof **and** David Meger. "Addressing Function Approximation Error in Actor-Critic Methods". **in***International Conference on Machine Learning*: 2018, **pages** 1582–1591.

[3] Tuomas Haarnoja **andothers**. *Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor*. 2018. DOI: `10.48550/ARXIV.1801.01290`. URL: `https://arxiv.org/abs/1801.01290`.

[4] E. T. Jaynes. "Information Theory and Statistical Mechanics". **in***Phys. Rev.*: 106 (4 **may** 1957), **pages** 620–630. DOI: `10.1103/PhysRev.106.620`. URL: `https://link.aps.org/doi/10.1103/PhysRev.106.620`.

[5] Arsenii Kuznetsov **andothers**. *Controlling Overestimation Bias with Truncated Mixture of Continuous Distributional Quantile Critics*. 2020. DOI: `10.48550/ARXIV.2005.04269`. URL: `https://arxiv.org/abs/2005.04269`.

[6] Timothy P. Lillicrap **andothers**. *Continuous control with deep reinforcement learning*. 2015. DOI: `10.48550/ARXIV.1509.02971`. URL: `https://arxiv.org/abs/1509.02971`.

[7] Antonin Raffin. *RL Baselines3 Zoo*. `https://github.com/DLR-RM/rl-baselines3-zoo`. 2020.

5 Appendix

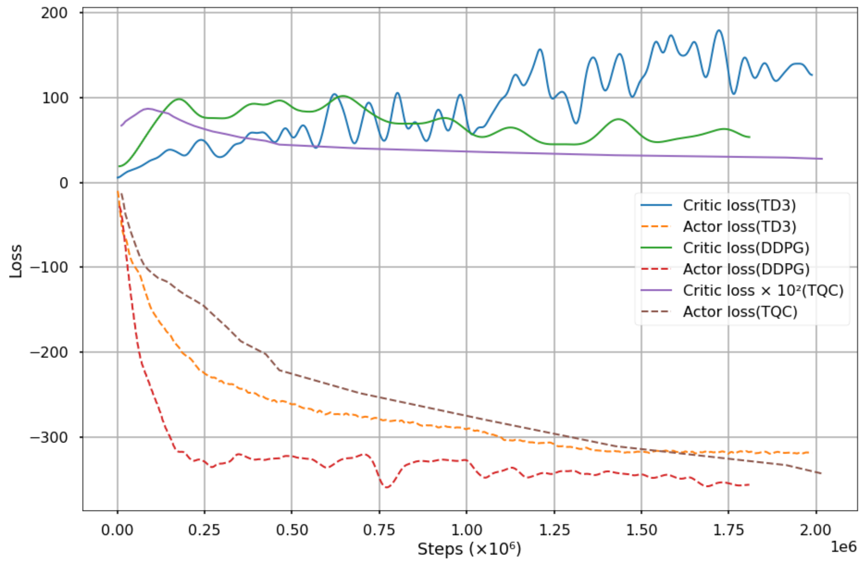

Figure 3: Losses for actors and critics across TD3, DDPG, and TQC