Landing Spaceships with Deep Reinforcement Learning

Elron Bandel
Department of Computer Science
Bar-Ilan University
elronbandel@gmail.com

Eyal Cohen
Department of Computer Science
Bar-Ilan University
eyalcohen308@gmail.com

Abstract

In this work we apply many Deep Reinforcement Learning Algorithms for the OpenAI's Lunar Lander challenge. We found that Double Deep Q-Learning was most effective. Together with Experience Replay our model converged in less than 100 episodes and managed to solve the environment - achieving more than 282 points on average. We tested our model with different algorithmic options and in noisy observation space. This work contains the results and evaluations of the different approaches for the problem.

1 Introduction

Reinforcement Learning (RL) is one of the key components of the machine learning field. The significance of Reinforcement Learning is that it can be directly applied to learn prefered behaviour in real life applications. The basic abstraction of Agent, Environment and Reward can describe plenty of real world problems. In the past years the field is growing fast managing to solve harder problem then ever before. in this work we use few of the most essential components in this rapid progress, such as Deep Q-Learning, Target Learning, Experience Reply, Policy Gradients and Actor Critic. We apply those methods to solve continuous environment with noisy observations. Our experiments show that Deep-Q learning with Experience Replay can solve the environment quickly and efficiently.

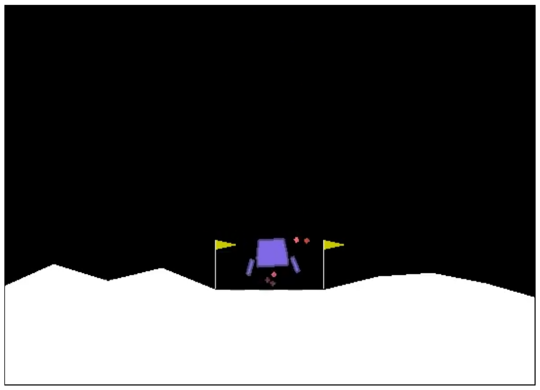


Figure 1: Our trained agent successfully landing the LunarLanderContinuous-v2 spaceship.

2 Lunar Lander

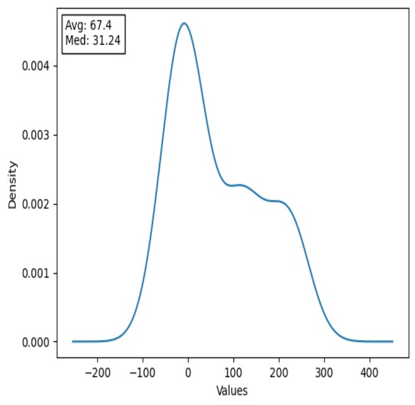
OpenAI's Lunar Landar is an environment in OpenAI Gym that contains an agent which is a spaceship that being rewarded for successful landing in the required area. We use the continues version of this environment, LunarLanderContinuous-v2, that require agent with continues action space. The

Observation Space is the vector: [Position X, Position Y, Velocity X, Velocity Y, Angle, Angular Velocity, Is left leg touching the ground: 0 OR 1, Is right leg touching the ground: 0 OR 1] and the continuous action space is a vector of two floats [main engine, left-right engines]. Main engine: -1..0 off, 0..+1 throttle from 50% to 100% power. Engine can't work with less than 50% power. Left-right: -1.0..-0.5 fire left engine, +0.5..+1.0 fire right engine, -0.5..0.5 off. Landing pad is always at coordinates (0,0). Coordinates are the first two numbers in state vector. Reward for moving from the top of the screen to landing pad and zero speed is about 100..140 points. If lander moves away from landing pad it loses reward back. Episode finishes if the lander crashes or comes to rest, receiving additional -100 or +100 points. Each leg ground contact is +10. Firing main engine is -0.3 points each frame. Solved is 200 points.Fuel is infinite and Landing outside landing pad is possible.

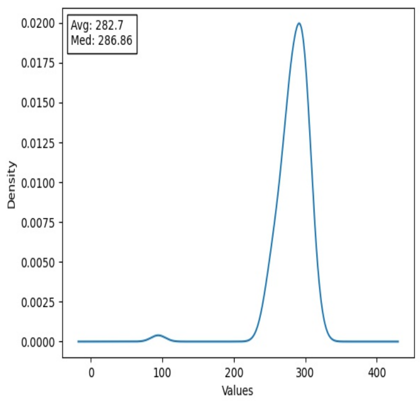
In order to use algorithms that utilize discrete action space we used quantization to make a dictionary of 10 discrete action containing combinations of continuous actions. The observation by default are not noisy, in some of our experiments we added gaussian noise to the observations to see how the different methods handle noise.

3 Experiments

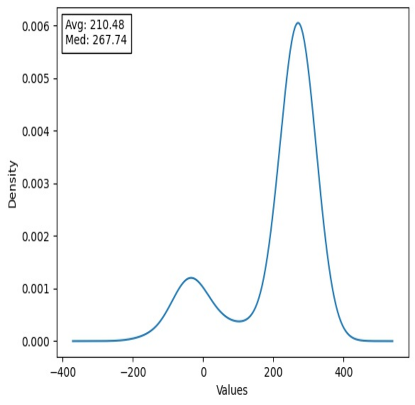
In this section we describe the methods we used, the training precedure and hyperparameters as well as a discussion about the results.



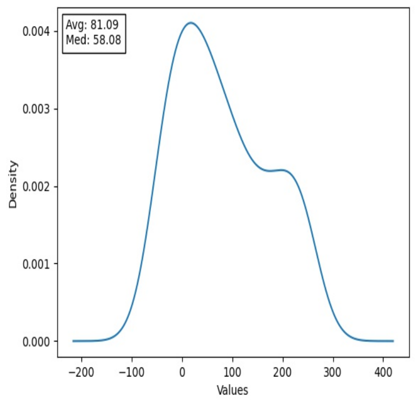
(a) DDQN



(b) Experience Replay



(c) Prioretized Experience Replay



(d) Experience Replay with Noise

Figure 2: Test Results Distribution of the different methods.

3.1 Deep Q-Learning (DQN)

In Q-Learning [Watkins (1989)] we attempt to learn $Q(s, a)$ that predicts the expected reward from taking action a at state s. in Deep Q-Learning [Mnih et al. (2013)] we attempt to learn deep neural network that approximate Q. In order to do so we convert the problem to supervised learning problem such that we measure the real $Q(s, a)^*$ value and learning model that minimze the Mean Square error between the model approximations and the real Q value. We use for our Q approximator a simple MLP with 2 hidden layers of dimension 1024. Our results show that DQN by itself was not enough, it was hard for the network to converge and it was not sufficient method to solve the environment.

3.2 DDQN - Target Network

In order to make the process much faster and more stable we used Double DQN [van Hasselt et al. (2015)] with target network and now tried to approximate the summation of the imidate rewared after every action and the expected future rewared given by the target network. in practice the target network was an old version of our network which we updated once every few training episodes. our results show that even though it did not solve the environment completely (Figure 2a) it still manage to converge and achieve nice results (Figure 3).

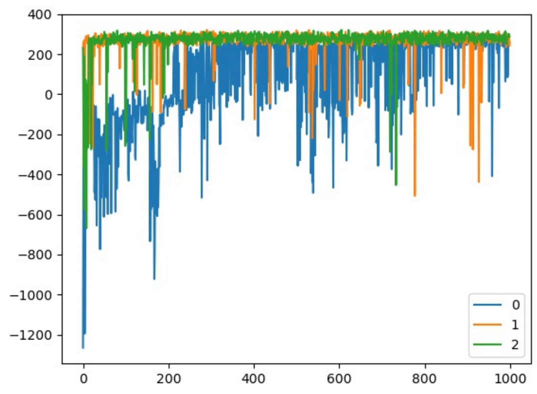


Figure 3: Rewards during training with DDQN (0) and with DDQN with Experience Replay (1),(2)

3.3 Experience Replay

In order to gain many time from every sample from the environment we created Experience Reply [Zhang & Sutton (2018)] buffer that stored tuples of (state, action, reward, next state) and utlized past experineces from the buffer many times during training. This method helped the model to converge after less then 100 training epsiodes (Figure 3) and solve the environment completely with average reward of 282.7 (Figure 2b). overall it is the best agent we managed to train.

3.4 Prioritized Experience Replay

Some experiences stored in the Experience Replay Buffer are more indicative than others. In this section we use prioritized Experience Replay [Schaul et al. (2016)], a method that utilize the TD error in order to mark harder examples that can contribute more to the model training. Other than sampling batch from the distribution ranked by the hardness, inside the batch itself - we are giving more weight to harder examples. We implemented the method successfully but did not manage to get it to converge faster than regular experience replay (Figure 4) but did manage to train it to solve the environment with average of 210 (Figure 2c)

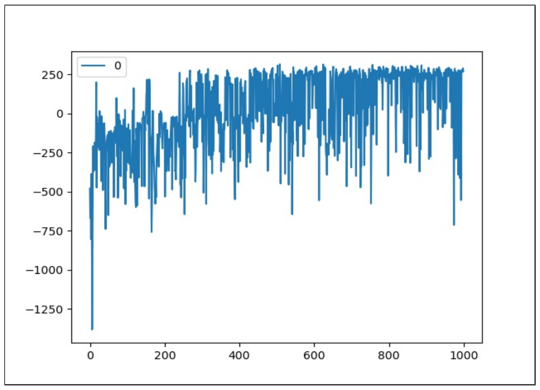


Figure 4: Rewards during training with DDQN with Prioritized Experience Replay.

3.5 Actor Critic with Experience Reply

Othen than using off policy Q learning methods we tried to apply policy gradient methods. We wanted to combine them with our successful experience replay mechanism, one such method is Actor Critic With experience Replay [Wang et al. (2017)]. unfortunately, we did no manage to train it successfully. even though it worked from time to time overall its rewards did not converge (Figure 5).

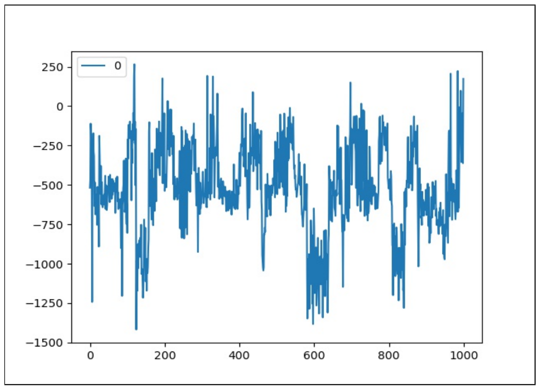


Figure 5: Rewards during training with Actor-Critic with Experience Replay.

3.6 Noisy Observation Space

Applying the best method we found, Double DQN with experience replay, with noisy observation worked as well. We added to the observations from environment Gaussian noise with mean 0 and standard deviation of 0.05 and found that our method managed to overcome this noise. Even though it was successful it did not work as good as in the un-noisy environment (Figure 2d) and it took it much more time to converge (Figure 6).

4 Conclusions

We find that Double DQN with Experience replay worked well for OpenAI's LunarLanderContinuous-v2. Not only it converged faster than every other approach we tried it also managed to succeed with noisy observations from the environment. Our final best model

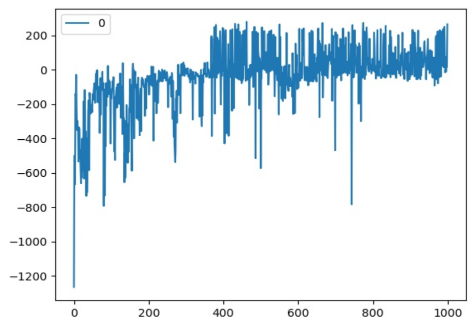


Figure 6: Rewards of our model in the Noisy Environment during training episodes.

achived an avarage rewared of 282.7 over 100 consecutive epsiodes and converged in less the 100 training epsiodes.

References

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay, 2016.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning, 2015.

Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay, 2017.

C. J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Oxford, 1989.

Shangtong Zhang and Richard S. Sutton. A deeper look at experience replay, 2018.