Robotics 1 - HW4 Write Up

GitHub Repo link (please use the “main” branch): https://github.com/rho-selynn/592-HW4

Team members: Roselynn Conrady and Michael Holm

1. We did this by using the code from lecture “07 Reinforcement Learning.” There were some dependencies that we needed from the lecture code in order to have OpenGym AI and PFRL to work with Google Colab
2. We went through PFRL’s quickstart guide, and replaced “env = gym.make(‘CartPole-v1)” with “env = gym.make(‘LunarLander-v2)” in order to have our Lunar Lander environment. In class, we learned that reinforcement learning is very popular to test on video games due to the huge amount of trial and error that is needed to teach a model. Bearing this in mind, we decided to increase the number of episodes from 300 to 500. Another modification we made was to decrease the epsilon value from 0.3 to 0.25 in order to decrease the amount of exploration to increase the ability of our model to further learn paths that offer high reward. These modifications allowed the lunar lander to land between the flags. However, it is not perfect. With these parameters, the lunar lander has a tendency to crash land or not land between the flags (Fig. 1). Therefore, we attempted to optimize these parameters in the next step.

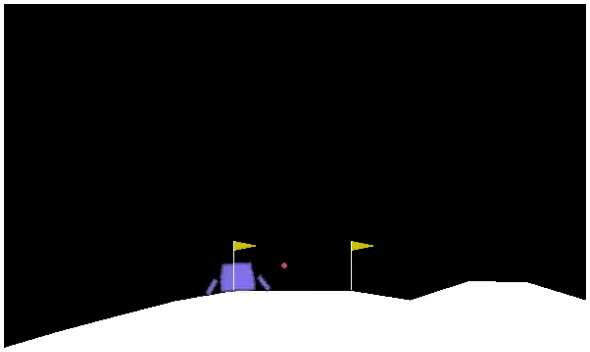


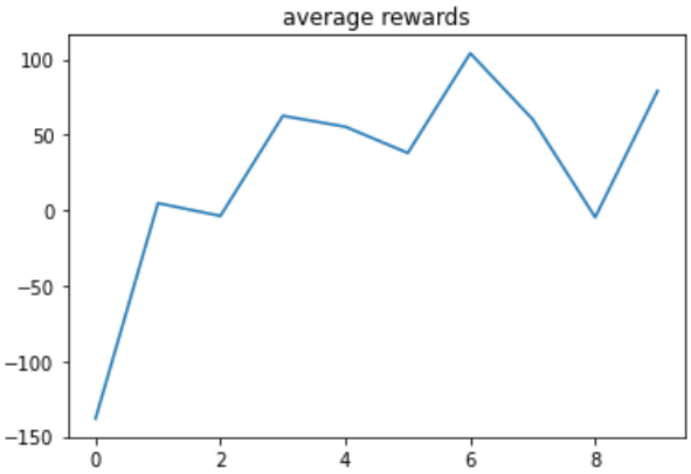
Figure 1. The lunar lander crash landed and did not land between the flags

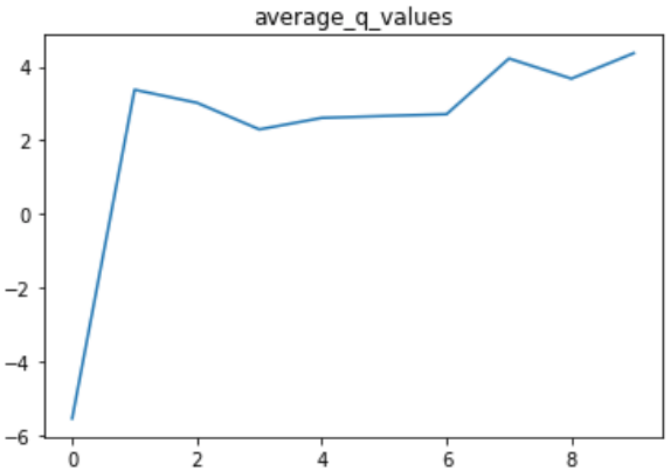
3. By plotting the loss over training time, the average Q values over training time, and the average rewards over training time, we were able to assess the impact of changing various parameters. The chart below shows each parameter, and its effect on the network performance. A plus sign (+) indicates a direction of change where the lunar lander learns better. A negative sign (-) indicates a direction of change where the lunar lander does not learn better.

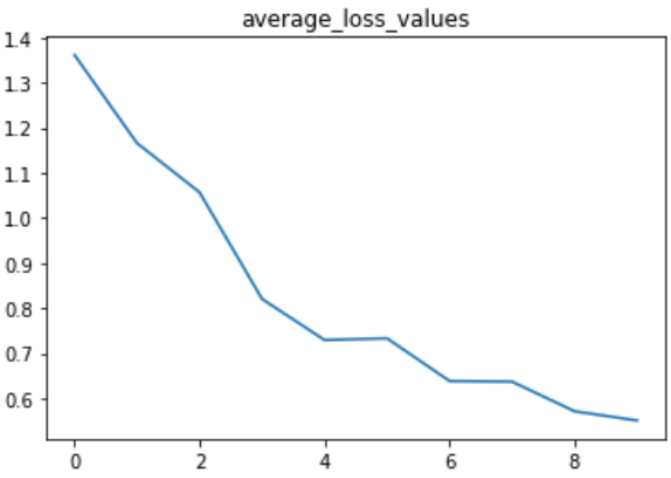
Parameter	Direction of change	Effect
Number of layers/deepness of	+	We found that increasing the number of layers, and

network		slowing the dimensional growing and shrinking of the network over several layers improved the performance of the network. Additionally, we found that adding layers past the point that we did actually reduces performance.
Activation functions	N/A	We found that ReLU provided the best results. This is supported by this website: https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/
Gamma	N/A	We found that the best gamma value was .9. Changing from this value worsened results in both directions.
Epsilon	-	We found that the best setting for this was at .25. Going lower or higher than this resulted in a worse model.
Replay start size	N/A	We found no clear effect when changing replay start size.
Q Network update interval	N/A	We found that the best setting for this was the default at 1. Altering from this worsened results.
Target q network update interval	N/A	We found that the best setting for this was the default at 100. Altering from this worsened results.

Shown below are the training graphs from the best model we could create:







The x axis represents every 50 episodes.

For the best model we could create, we found that the graphs above had an expected behavior. Loss decreased reliably as training episodes increased, the average Q value sureness raised over time, and the average rewards for each episode increased. These

lines represent a model that learns more and more of the Q function over many episodes, becomes more and more sure of itself throughout training, and performs better in the lunar lander game throughout training. Averaged over 100 episodes, our model achieved an average reward of 20, while the initial model achieved an average reward of just 14.