**What we did**

In this assignment, we trained an agent in the LunarLander-v2 environment using a simple DQN epsilon policy network. We also visualized the agent’s attempt at landing in between the 2 flags before and after training the network. In addition, we looked at the learning progress by examining several plots.

**Images and/or Discussion**

Step 0:

A video game screen with a black background

Description automatically generated

Above is the screenshot of the initial environment when seed is set to 42, which is the seed we’ll use from now on.

Step 2:

The activation functions we chose for the network are ‘elu’ for the hidden layers and ‘linear’ for the output layer. We chose this not just because they’re used in the lab exercise, but also because we believe that it can prevent dead neurons by being able to output negative values unlike ‘relu’. Because of this prevention, we believe that it provides stability during training, which can make the training more effective. As for the ‘linear’ activation function for output layer, it’s to ensure that we don’t restrict the prediction of the Q-Values.

As for the loss function, we chose ‘mse’ (Mean Squared Error) because we’re predicting Q-Values, which are continuous values. We believe that MSE is suitable for this because it punishes larger errors, which will make the model correct itself more aggressively. In addition to that, it is also easy to interpret.

Step 5:

A graph with a line going up

Description automatically generated

The first plot above is the cumulative rewards after each episode. As we can see, it is always dropping, which means that our agent doesn’t seem to be good at landing the Lander yet.

A graph showing a blue line

Description automatically generated

The second plot above is the rewards that the agent obtained per episode during training. As we can see it still seems a bit random, maybe we need higher number of episode or a more complex network. Please correct us if we’re wrong.

A graph showing steps per episode

Description automatically generated

The third plot above is the number of steps reached per episode. As we can see, it actually survives for longer as the episodes go on.

After looking at all 3 plots, we think that it might’ve learned to survive more than to actually land the lander. As we can see from the third plot, the number of steps actually increase as the episode goes on. However, the rewards seem to be still random towards the end. The cumulative reward also goes down, which means that we probably need to experiment more if we have the time.

Step 6:

In our opinion, the main challenge we faced during training is time/hardware. It takes quite long to finish training the agent, which might stem from the lack of proper hardware to train. If we have better hardware and more time, we can definitely improve the agent’s performance because then we can experiment with more hyperparameters as well as different architectures for the DQN network.

**What we learned**

* learned more about the LunarLander-v2 environment via the official documentation: <https://www.gymlibrary.dev/environments/box2d/lunar_lander/>
* learned more about balancing exploration vs exploitation using a decaying epsilon.
* learned more about why as number of episodes go up during training, it gets slower and slower.
* learned more about the use of neural network in reinforcement learning.
* learned more about how to examine the plots to learn about the agent’s learning process.