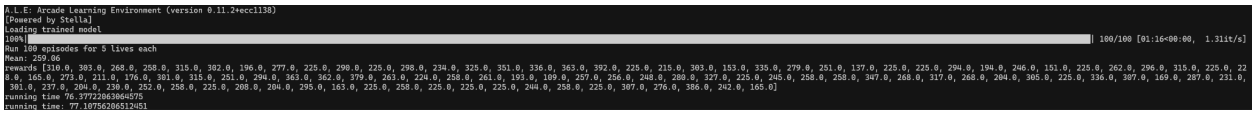
Prior Experiments

Experiment 1



Experiment Details

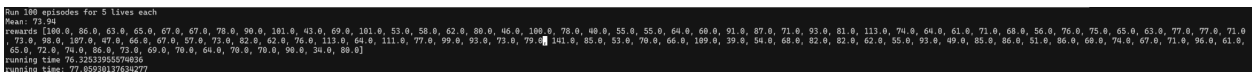
Parameters

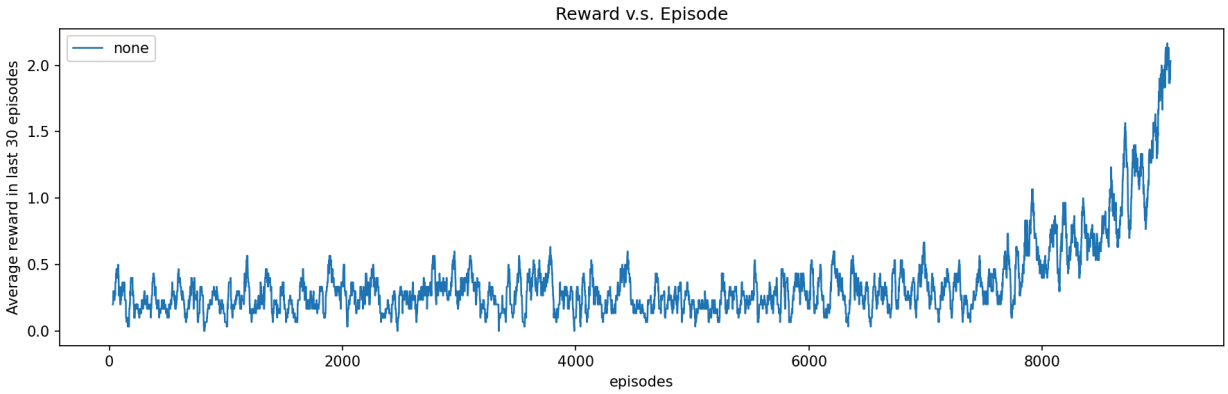
- Episodes = 100000
- epsilon
    - start = 0.01
    - end = 0.001
- decay episodes = 100000
- 200–500 million frames

These were the parameters I configured to tune my model. When first running the experiment I noticed that the step count was far too quick to decay the epsilon value. Therefore, I decided to make it dependent on the decay steps so that I could alter how long it takes for the epsilon decay. I did this because as we know, the higher the epsilon, the higher exploration is, so I wanted to make it that 20% in the beginning was the exploration phase and the rest was more of an exploitation approach. I set decay episodes to be 100000, and with that I also set eps_decay_steps to increase to at least 1M. I did this since we need more experience in order for the agent to learn how to score highly on breakout. So if we have more step for the decay when training, we are able to have more available steps to gain more experience. Also, for my double dqn approach I first decided to take a very conservative approach of making the epsilon be 0.01~0.001. This is because when I did more of an exploration approach, the learning progression was not as good as with a more conservative approach. The only drawback of this was that since the decay rate was extremely slow, it led to a very slow training process. Therefore I decided to take a faster learning approach leading to my final results for this project.

Experiment 2

mean: 73.94





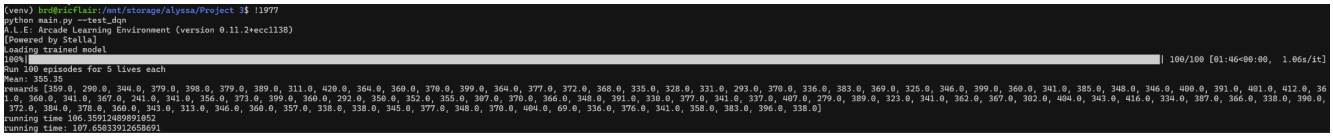
Experiment Details

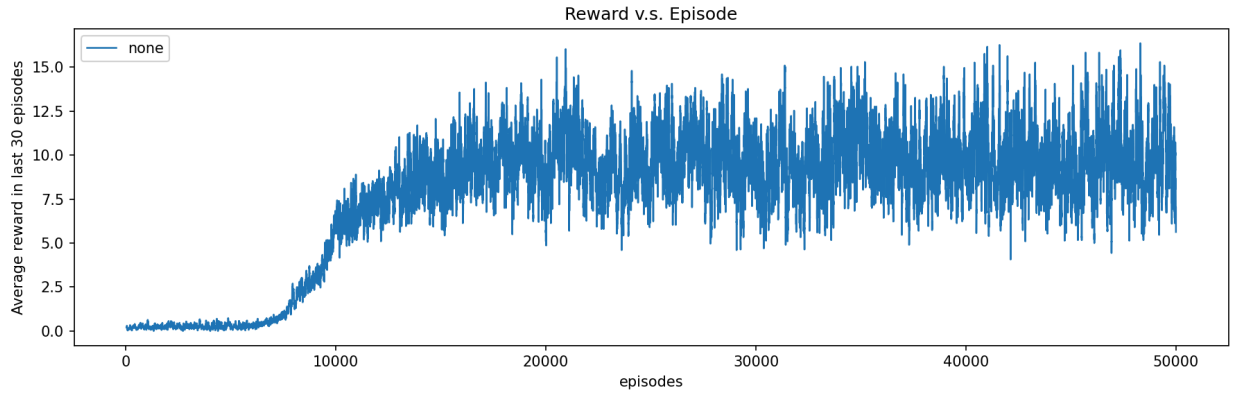
Parameters

- Episodes = 10000
- epsilon
    - start = 1.0
    - end = 0.01
- decay episodes =10000
- 40 million frames in total

Unfortunately due to limited time, I could not train this model as long as I would've idealistically would like to. However, from the results, the average was above for most of them and less training time. Therefore I decided to further investigate this by training it for longer with these settings since from the first experiment having a longer training time leads to better results.

Final Results





Experiment Details

Parameters

- Episodes = 50000
- epsilon
    - start = 1.0
    - end = 0.01
- decay episodes = 10000
- 200,000,000 frames
- Exploration for the 20% beginning of training
- Number of layers:
- Optimizer: RMSprop
- Loss function: Huber loss

From our first two experiments, for the final model I wanted to have a target of 20% exploration for the beginning and the rest to take an exploitation approach. Since I also wanted to train longer I changed the episodes to 50000 which took approximately 5 hours to train. I kept my epsilon to have a very explorative to a good exploitation rate. Like stated before, in my code I depended the amount of decay episodes to spend a certain amount of episodes with a certain decay rate. I set it to 10000 in order to have a slower decay rate of change. Therefore I was able to hit my target of 20% exploration and the rest have more of an exploitation approach. Overall, with these parameters, I was able to process around 200000000 frames. The optimizer used was RMSprop and the loss function was Huber loss. I chose the Huber loss function because for a double DQN there could be the occasional huge TD error. Even though it is rare this could make huge misestimates. Therefore using Huber loss counteracts this because the TD error is linear when it is large. Thus avoiding exploding gradients. I chose RMSprop as the optimizer since the gradients is non-stationary. RMSprop maintains the exponential moving average of squared gradients and rescales updates per parameter. Therefore being ideal in the Double DQN since it changes with the parameters.

As we can see from the results, the mean is 355.35. And all the average rewards are well above >= 40. Something nice to also highlight is the graph results comparison of the prior experiment

and the final results. We can see in the prior experiment that it was trending up however could not have the trend we see with the final results. This is because we did not have as much training time as for the final model. Thus this model successfully hit the requirements. From the development of the 20% strategy, we were able to come up with a successful breakout model!

Sources

- https://arxiv.org/pdf/1509.06461
- https://www.nature.com/articles/nature14236
- https://www.cs.toronto.edu/~vmnih/docs/dqn.pdf
- https://jonathan-hui.medium.com/rl-dqn-deep-q-network-e207751f7ae4
- Gpt for debugging and graph setup.