DEEP Q NETWORK

1. INTRODUCTION TO DEEP Q NETWORKS

Deep Q Networks are introduced by Mnih et. al. in 2015 to successfully learn control policies directly from high-dimensional sensory input using reinforcement learning. Even this model is using a similar structure with Q-Learning, it differs from Q-Learning on following points:

- Q-learning works on states, whereas Deep Q Learning can work on features. Working on features are important for generalization. By doing this, an agent can predict moves on states that it never have seen before. Also, it can learn from similar states and uses them on each other.

- Q-learning algorithm converges to optimum, whereas Deep Q Learning might or might not depending on the problem and hyperparameters.

- Q-learning only uses the last seen state-action pair to improve the algorithm, whereas Deep Q Network uses a batch of historical data to improve the weights. This reduces overfitting to state-action pairs that are seen last.

2. OPENAI GYM AND LUNAR LANDER ENVIRONMENT

OpenAI gym is a platform that developers can test their agents on different atari games. Mainly, Reinforcement Learning algorithms are implemented to solve the problems in OpenAI Gym.

Lunar Lander environment is one of the atari games that OpenAI Gym has. The main goal of Lunar Lander environment is to successfully land the Lunar Lander on its two landing pads.

It has a 8-dimensional continuous state space and 4 discrete actions. In original Deep Q Learning paper the algorithm is used with much more complex inputs such as the screen pixels itself, Lunar Lander environment gives us simpler state values.

3. EXPLORATION vs. EXPLOITATION

Reinforcement Learning algorithms uses exploration and exploitation to learn and solve this kind of problems. I have used some decaying e-greedy algorithm to create the balance between exploration and exploitation. In the beginning of the learning phase, the agent is more prone to exploring, using random actions. Then, it slowly becomes more prone to exploiting what it has learned.

4. DEEP Q NETWORK AGENT

a. Parameters

Creating the agent for Deep Q Learning is the main part of this Project. So, I have constructed a class and started it with these values:

- Values of Lunar Lander environment, such as the size of state space and action space.
- Epsilon values starting and changing Exploration - Exploitation balance
- Learning rate, which is a step used at each learning step
- Gamma, discount value for later rewards,
- Batch size, which is the size of each batch used at replay experiences

As I have tried different hyperparameters for these parameters, I have seen that the most important two parameters are the Learning rate and the Gamma value. In the beginning of experimenting, I always used a neural network with 2 hidden layers, 32 nodes each. I have tried many different combination of parameters on learning rate, gamma, and batch size, along with the decay for exploration-exploitation balance. But, they couldn't help me get consistent values. At its best: lr = 0.00025, gamma = 0.99, batch size = 32, it got a negative average on testing after training on 1000 episodes.

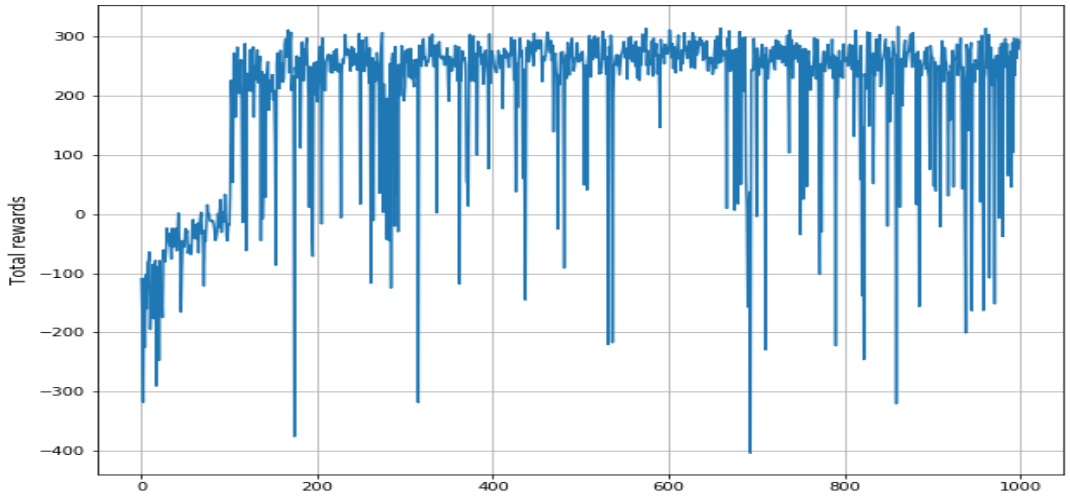
What made the difference for the reward was Neural Network Model's parameters for me.

b. Neural Network Model

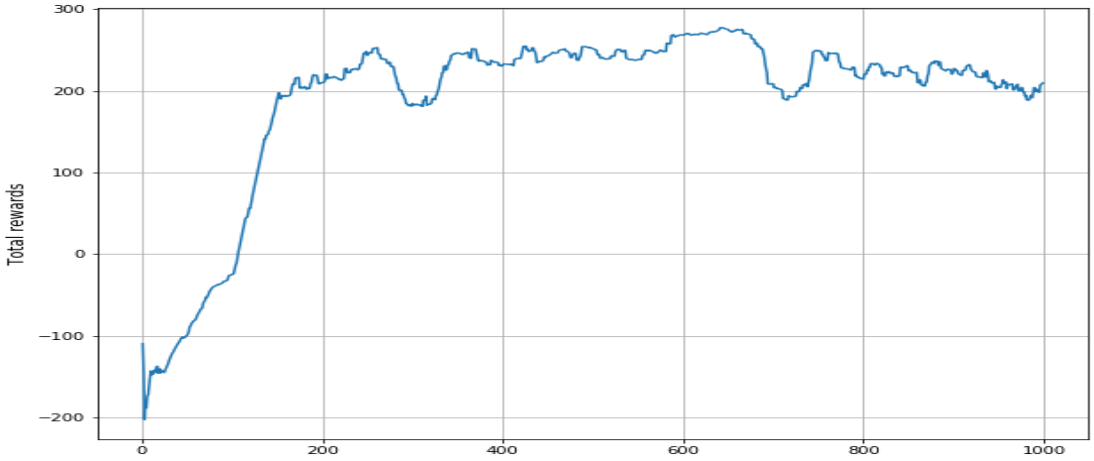
Another important difference could be made by changing the Neural Network model. The number of hidden layers, activation arguments, and the nodes for each layer had huge impact on results.

Even though I have started with Rectified Linear Unit activators in hidden layers, I tried sigmoid functions later, which gave much worse results at first 200 episodes of training*.

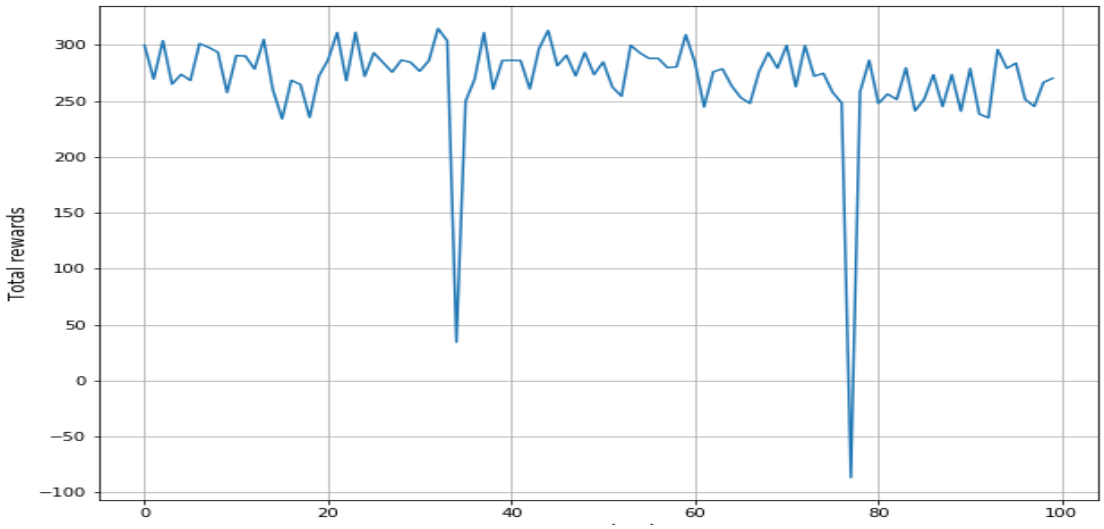
After this, I have tried to change the shape of Neural Network. I used 2 hidden layers with 64 nodes each using Rectified Linear Unit activator. This improved the results significantly, but it was not enough. As I have seen the improvement when expanding the node size, I used 400 nodes at the first and 200 nodes at the second layer. This model worked quite well using hyperparameters that are used at Deep Q Network (Nature) paper. Below you can see how quick it learned and the test results of the learning.



Training Period of 1000 episodes



Rolling average of training period (50 episodes)



Testing episodes

As it can be seen in the graphs, using Deep Q Network with this Neural Network model and parameters makes learning possible under 200 episodes.

```
memory=20000,
gamma=0.99,
max_eps=1,
min_eps=0.1,
decay=0.001,
lr=0.00025,
batch_size=32

self.model = Sequential()
self.model.add(Dense(400, input_shape=(self.state_size,),
                     activation = 'relu'))
self.model.add(Dense(200, activation='relu'))
self.model.add(Dense(self.action_size, activation='linear'))
self.model.compile(optimizer=Adam(lr=lr), loss='mse')
```

5. WHAT COULD BE DONE

Caused by some bad luck (power supply failure of my computer) and a very busy period, I didn't have too much time to play with parameters and neural network. Also, this caused the late submission. On the plus side, I have learned how to run my code on AWS.

I believe landing the lunar lander safely is the most important task. Its reward changes between -100 and +140. I'd increase its reward to ensure that the agent learns how to land safely before optimizing other things.

5. CONCLUSION

Even though, implementing Deep Q Network algorithm seemed to be toughest part of the project, because the learning phase takes long time and there are many parameters to change, it was much more difficult to use the Deep Q Network agent at its peak performance.

Several times, I have seen an algorithm gets really good, (around +200 for 15-20 episodes) then getting much worse. I believe hyperparameters caused this divergence and it was difficult to diagnose which one to improve.

Unlike it is discussed at Slack, grid search for hyperparameters didn't improve my scores. The difference is seen after I have changed the Neural Network model. I have increased the node at hidden layers. I believe this helped the Agent to figure out more complex relationships between reward and state space. And this bring the best result.