Reinforcement Learning

Assignment 2

Checkpoint 1

Sumeet Aher
sumeetmi
sumeetmi@buffalo.edu

Suraj Sah
suraj
suraj@buffalo.edu

"I certify that the code and data in this assignment were generated independently, using only the tools and resources defined in the course and that I did not receive any external help, coaching or contributions during the production of this work."

Abstract

- Explore 'CartPole-v1' and provide the main details about the environment (e.g. possible actions/states, goal, rewards, etc).
- Choose one more environment that you use for the assignment and provide the main details about it

1 Exploring CartPole-v1

A pole is attached by an un-actuated joint to a cart, which moves along a frictionless track. The system is controlled by applying a force of +1 or -1 to the cart.

1.1 Action Space

`Discrete(2)`

The Discrete space allows a fixed range of non-negative numbers, so in this case valid actions are either 0 or 1 which represents application of the force +1 or -1.

1.2 Obersvation Space

`Box(4,)`

The Box space represents an n-dimensional box, so valid observations will be an array of 4 numbers.

Observation Space bounds

Observation_space.high: `[4.8000002e+00 3.4028235e+38 4.1887903e-01 3.4028235e+38]`
`Obersvation_space.low: [-4.8000002e+00 -3.4028235e+38 -4.1887903e-01 -3.4028235e+38]`

1.3 Goal

The goal for this environment per episode is to keep the pendulum upright. So the pendulum keeps falling and in the effort to keep this pendulum upright, the cart may move on the axis. For keeping the episode end in check, there are two checks. Either the pole is more than 15degrees from vertical or the cart moves more than 2.4 units from the centre.

1.3 Reward

A reward of +1 is provided for every timestep that the pole remains upright

2 Exploring LunarLander-v2

The Lunar Lander will always land on (0,0), so the landing coordinates are fix.

2.1 Action Space

Discrete(4)
The Discrete space allows a fixed range of non-negative numbers, so in this case valid actions are 0, 1, 2, 3.

2.2 Obersvation Space

Box(8,)
The Box space represents an n-dimensional box, so valid observations will be an array of 8 numbers.
Observation Space bounds-
Observation_space.high: [inf inf inf inf inf inf inf inf]

Obersvation_space.low:[-inf -inf -inf -inf -inf -inf -inf -inf]

2.3 Goal

The goal for this environment per episode is to land on the coordinates.

2.3 Reward

- When the lander touches the ground, there is an additional reward of +100/-100 depending on the coordinates.
- As soon as any leg touches the ground, it gets a reward of +10
- There are three actions, from them firing the main engine will cost the agent -0.3 reward points
- On succesfull landing, the agent gets a reward of 200 points.

3 Grid World Used

The environment defined here is of a Kitchen. The agent in this Kitchen is an Ant whose objective is to reach her hole. In the environment grid there are small heap of sugar in form of positive rewards and pesticide in form of negative rewards. The environment has the following properties –

Agent: An Ant

States: 16 states in form of 4*4 square.

Actions: 4 actions {Right, Left, Up, Down}

Rewards: 5 Rewards in form of sugar and pesticide with values {-10,10,10,10,50} at location {(0,2), (1,1), (1,3), (3,1),(3,3)}

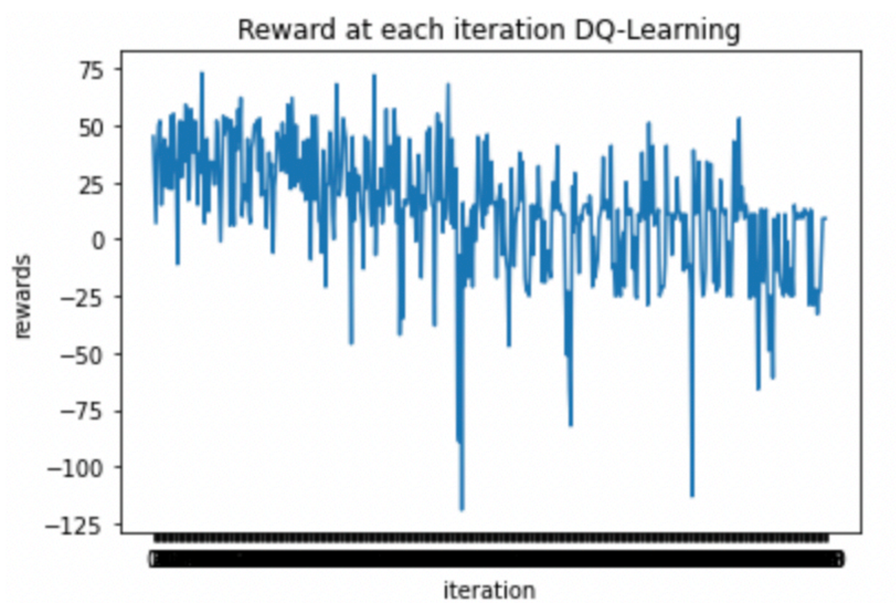
Main Objective: To reach hole home (3,3)

The changes introduced in environment for DQN –

1. The reward for going close +1 and -1 if going away from goal
2. All reward vanishes after it's taken for 1st time

- **Using experience replay in DQN and how its size can influence the results**
  -The experience replay was implemented using a list of 1000 capacity. Which is overwritten with each step tuple from start to end.
  The size if increased will need the network to be trained more as the number of samples taken from the replay will slowly be updated across the list. This will result more time in training and more number of episodes for training as the list properties needs to be updated across many steps

- **Introducing the target network**
  - The target network has the same architecture as the policy network. We copy the weights of the policy network to target network after every 5 episodes of training.
  - The target value is calculated from this network which is used as input to the loss function calculation and gradient decent of the policy network
- **Representing the Q function as q^(s, w)**
  - The Q function q^(s,w) is a three layer neural network with input as flatten and two 128 dimension hidden layer and output with Q(s,a) value corresponding to each action.



From the cumulative reward graph , we can deduce the training was unstable and didn’t increased in value over time

The final output on 10 episode run on the target network we see the reward comes to be 9. This compared to vanilla Qvalue table training which gave 31 output shows the trained model is not performing that well

Final Submission

3 Discuss the algorithm you implemented.

To get the target in Vanilla DQN, we wrote the following:

```
Y = reward + gamma*(1-done_)*max_q
```

Where *done* is the tensor thus keeping only those maximum q values which are not going to terminate in the next state.

To get the target in the Double DQN, we wrote the following:

```
next_q =
self.model_policy.predict(tf.reshape(state_next,(minibatch_siz
e,obs_size)))
max_a = tf.math.argmax(next_q, 1)
double_q =
self.model_target.predict(tf.reshape(state_next,(minibatch_siz
e,obs_size)))
target_y = reward_  + gamma*(1-done_)*(tf.gather(double_q[0],
max_a).numpy())
```

This code first predicts the action with maximum q value and then using this action we find the predicted value from the target model.

We use this value as the ground truth for back-propogation.

Algorithm implemented as an improvement to Vanilla DQN is Double DQN.

4 What is the main improvement over the vanilla DQN?

The main improvement expected over the vanilla DQN is faster convergence as well not not getting stuck at a local optimum thus not stopping the agent from moving over to a higher reward.

5 Show and discuss your results after applying your the two algorithms implementation on the environment. Plots should include epsilon decay and the reward per episode.

Below are the results for the grid environment, on application of DQN and then double DQN:

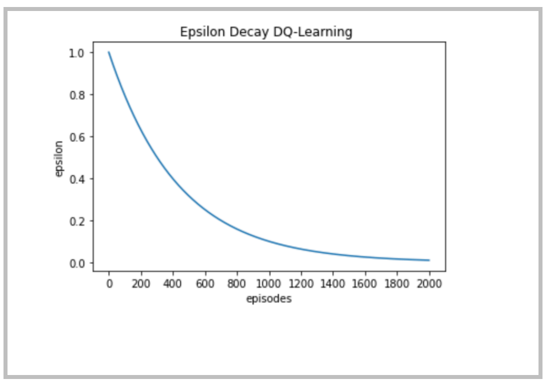


Figure 1: Epsilon Decay for DQN on grid

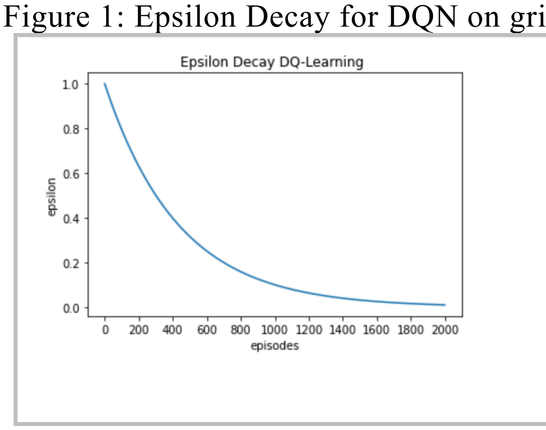


Figure 2: Epsilon Decay for Double DQN on grid

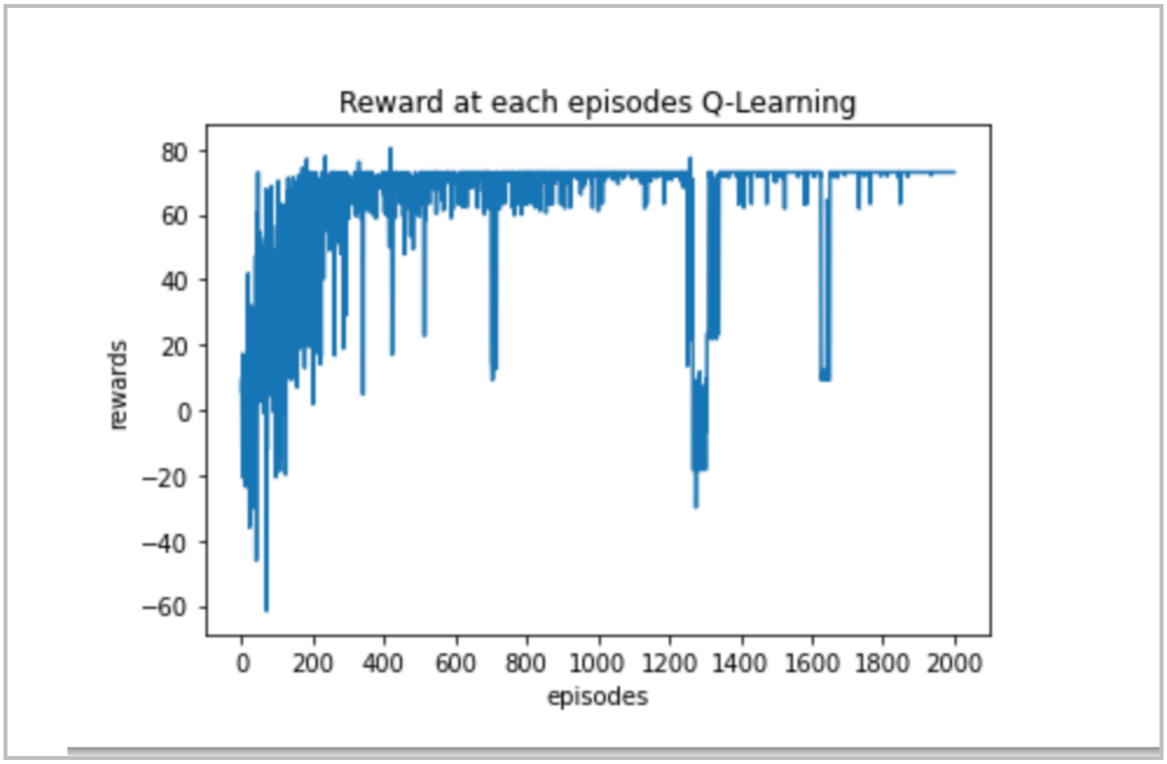


Figure 3: Cumulative reward per episode for DQN on grid

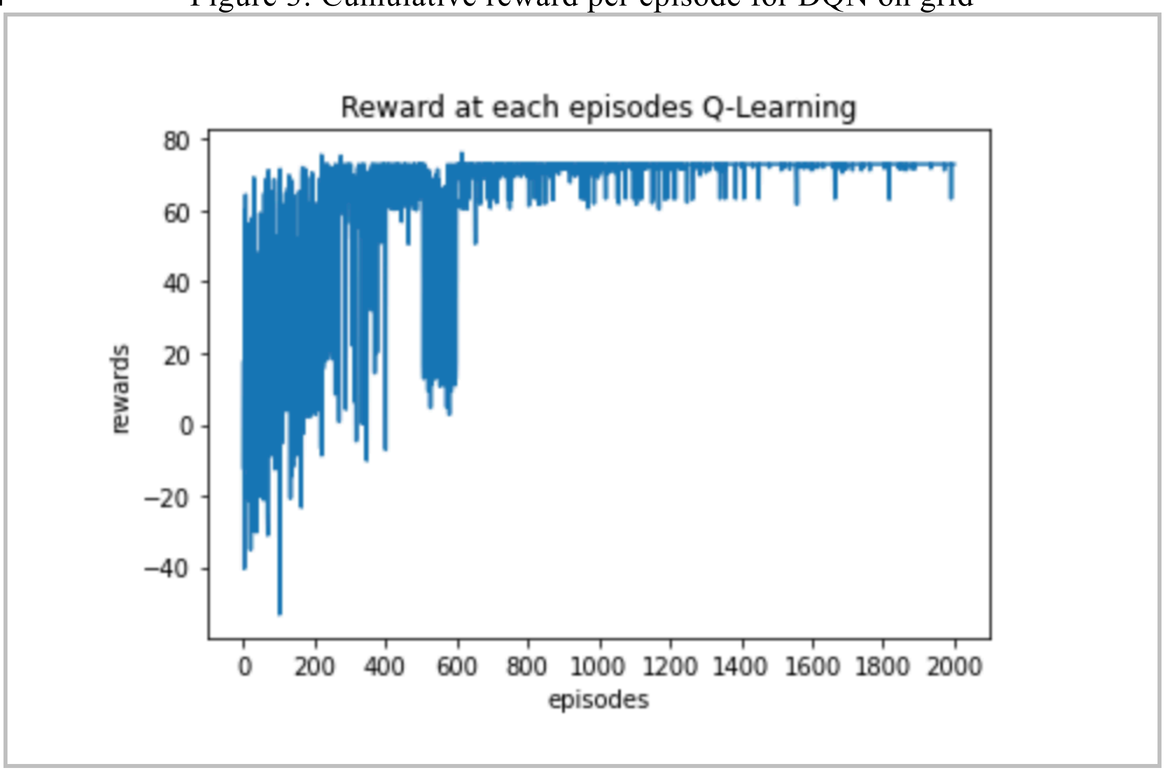


Figure 4: Cumulative reward per episode for Double DQN on grid

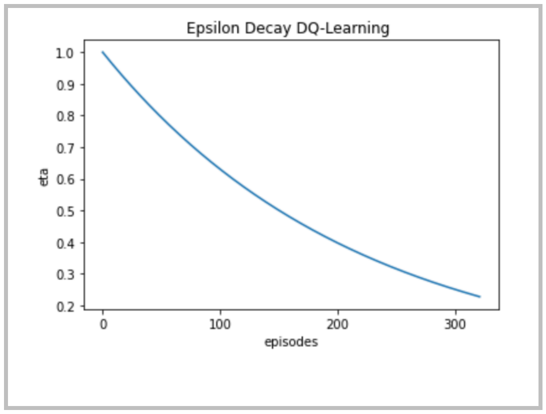


Figure 1: Epsilon Decay for DQN on cartpole (as we get average reward of >470 for consecutive 10 episodes, we conclude the training)

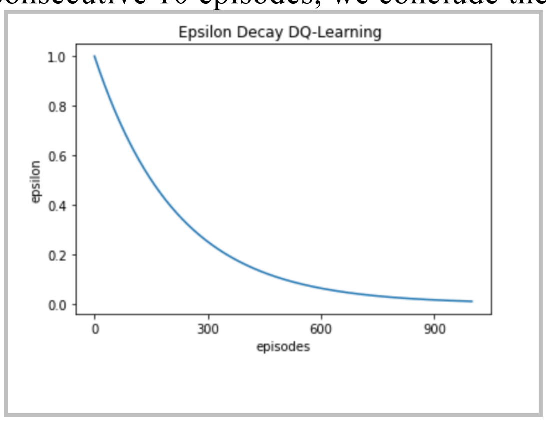


Figure 2: Epsilon Decay for Double DQN on cartpole

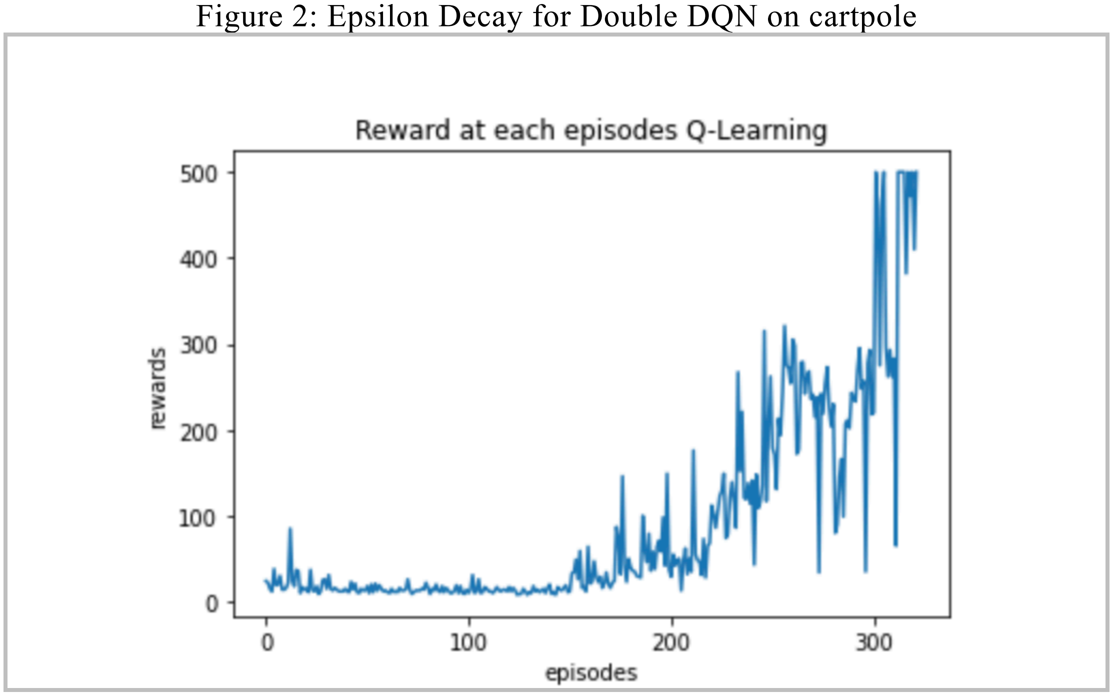


Figure 3: Cumulative reward per episode for DQN on cartpole

(as we got an average reward of >470 for 10 consecutive episodes, we concluded the training)

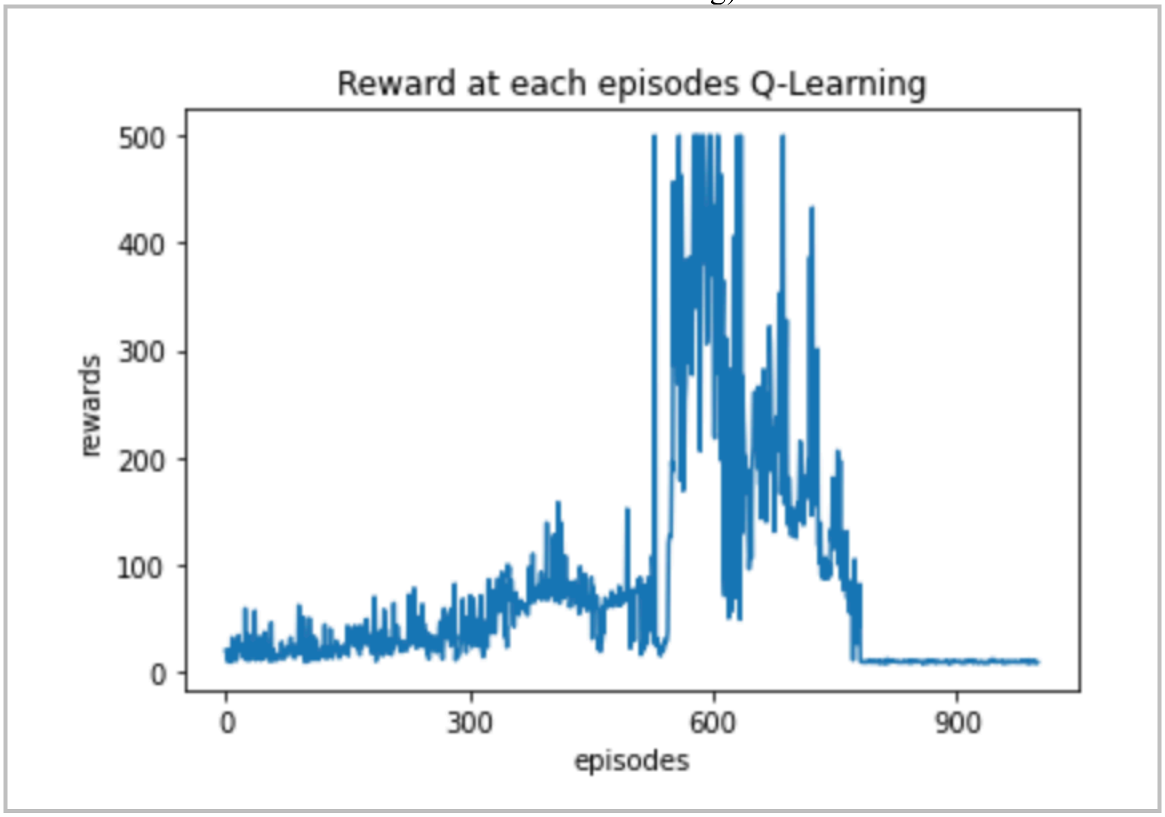


Figure 4: Cumulative reward per episode for Double DQN on cartpole

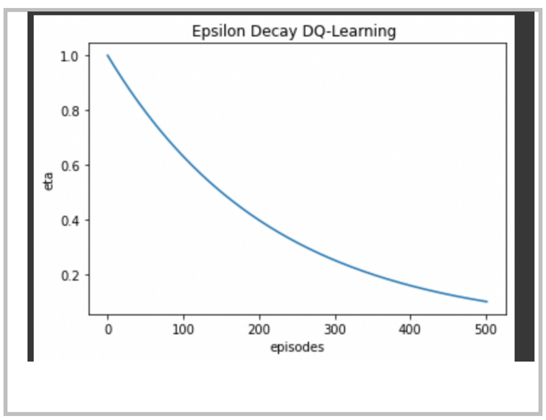


Figure 1: Epsilon Decay for DQN on Lunar Lander

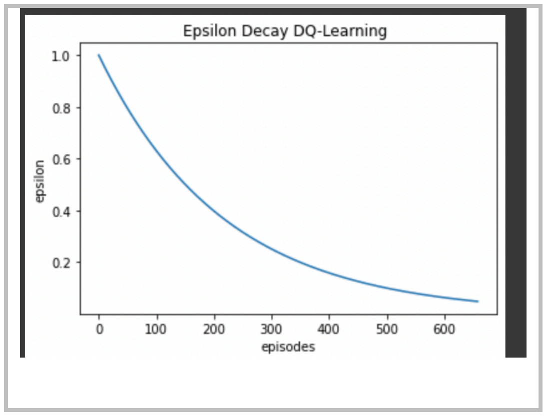


Figure 2: Epsilon Decay for Double DQN on Lunar Lander

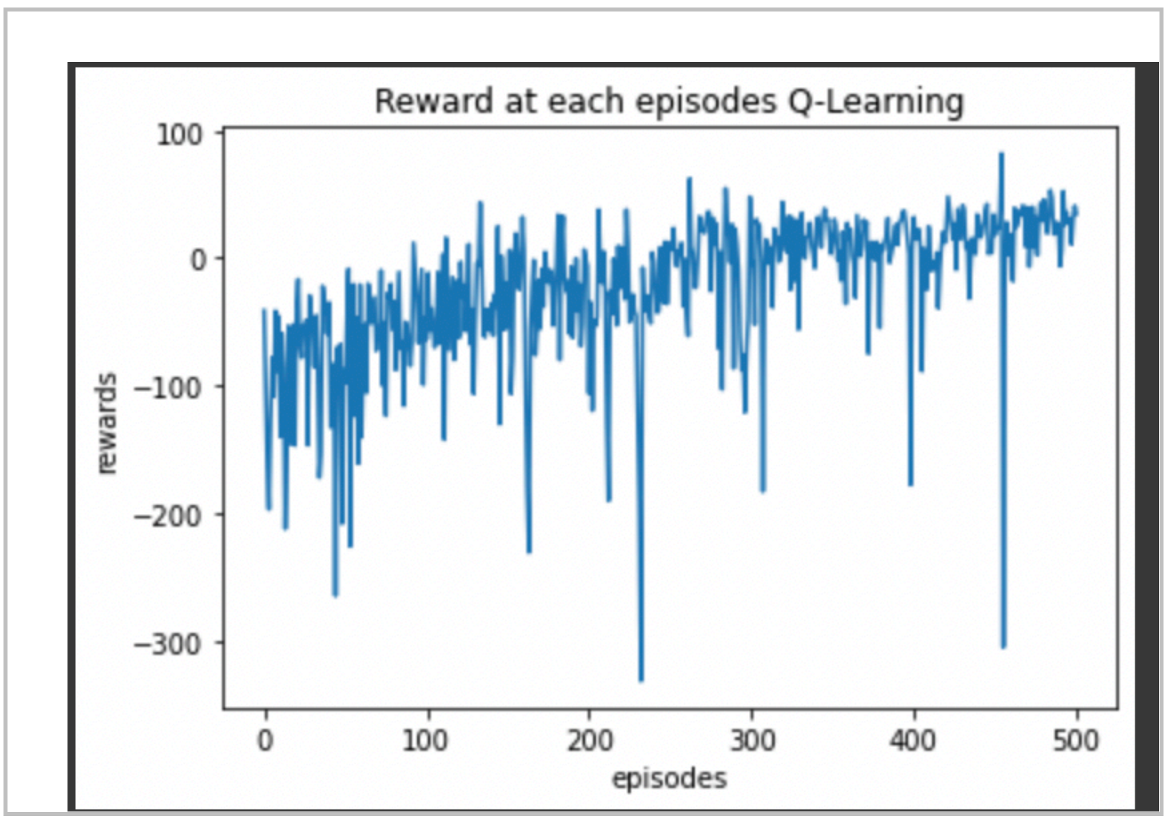


Figure 3: Cumulative reward per episode for DQN on Lunar Lander

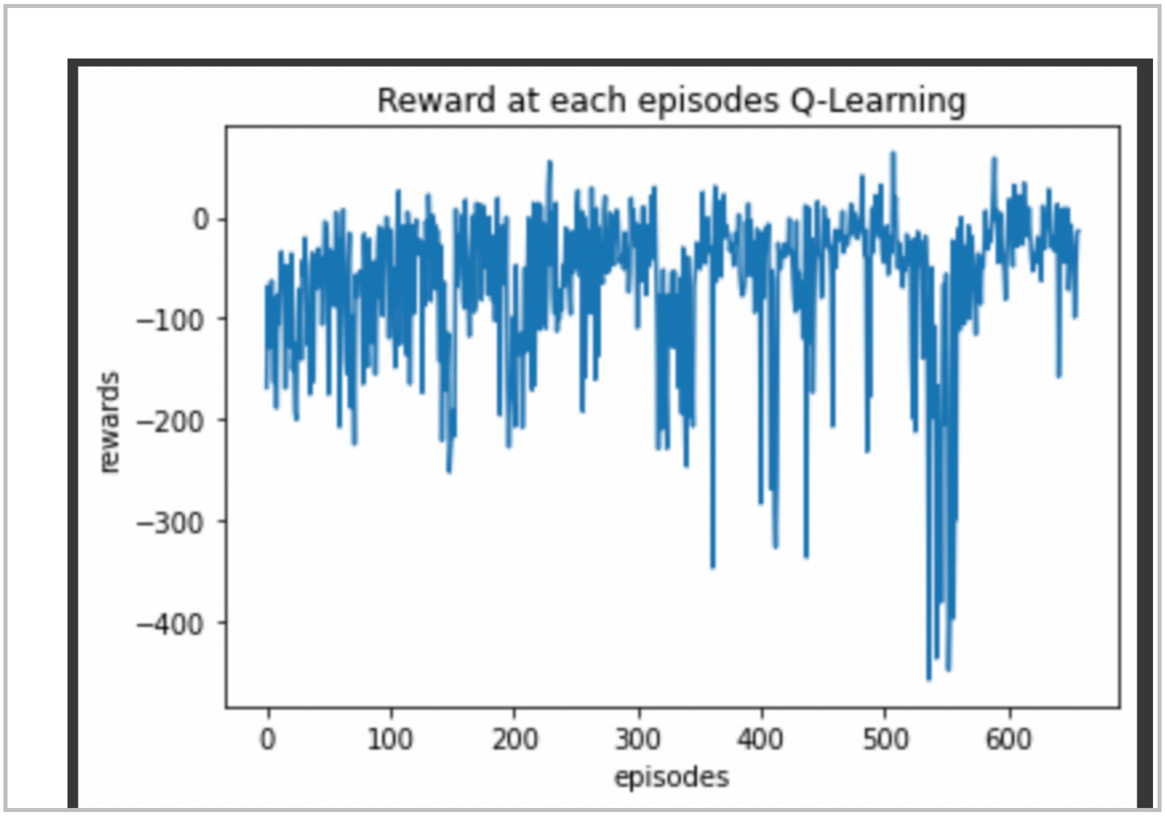


Figure 4: Cumulative reward per episode for Double DQN on Lunar Lander

1)For Grid environment, Double DQN seems to be more robust and on later stages doesn’t show any signs of comebacks, where as there is fluctuation in normal DQN training in the later stage but that is taken care of by the model.

2) For CartPoleV1, for just DQN, the model gave a consecutive average of >470 in the initial 400 episodes itself, and we ended the training there. Further we used the model_target to predict the next 10 episodes of the agent and we got a 500score on each of the episode using greedy approach. For Double DQN, we seem to have a training issue, as the agent collects the maximum reward but then slides down to lower rewards. Tuning the hyper parameters will solve this problem.

3) For Lunar Lander, we had to abruptly end the training around 550 episodes for both DQN and double DQN, hence the epsilon can be seen decayed only upto a certain amount as it was supposed to decay to 0.001 by 1000episodes. But, till 600 episodes, we can see that the agent was slowly learning and having increasing rewards.

6 Provide the evaluation results. Run your environment for at least 5 episodes, where the agent chooses only greedy actions from the learnt policy. Plot should include the total reward per episode.

Figure 5: DQN on grid world

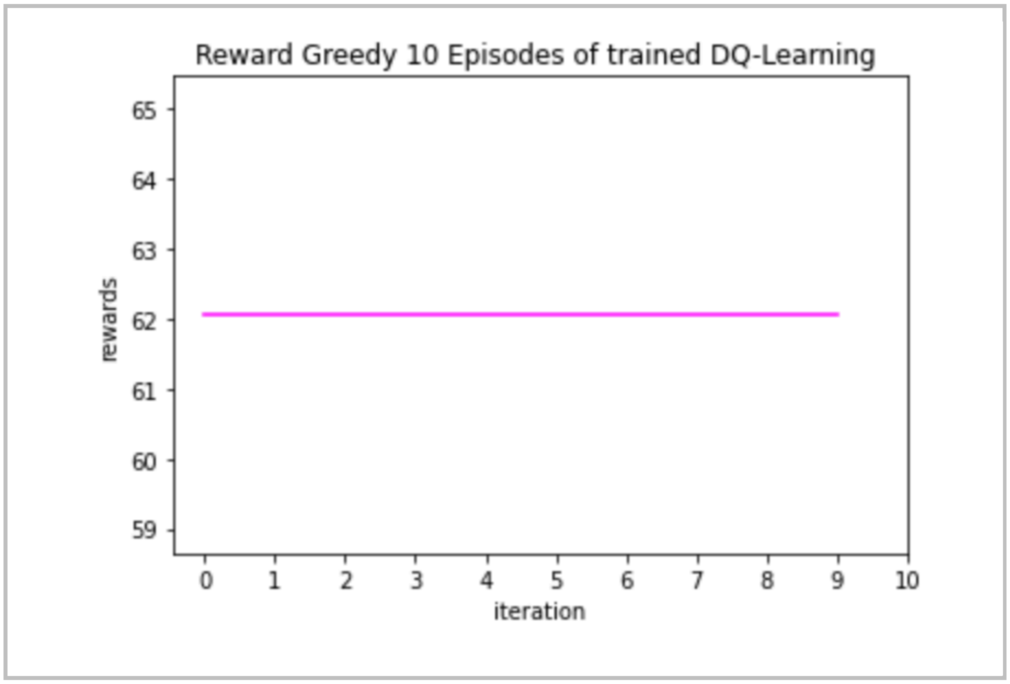


Figure 6: Double DQN on grid world

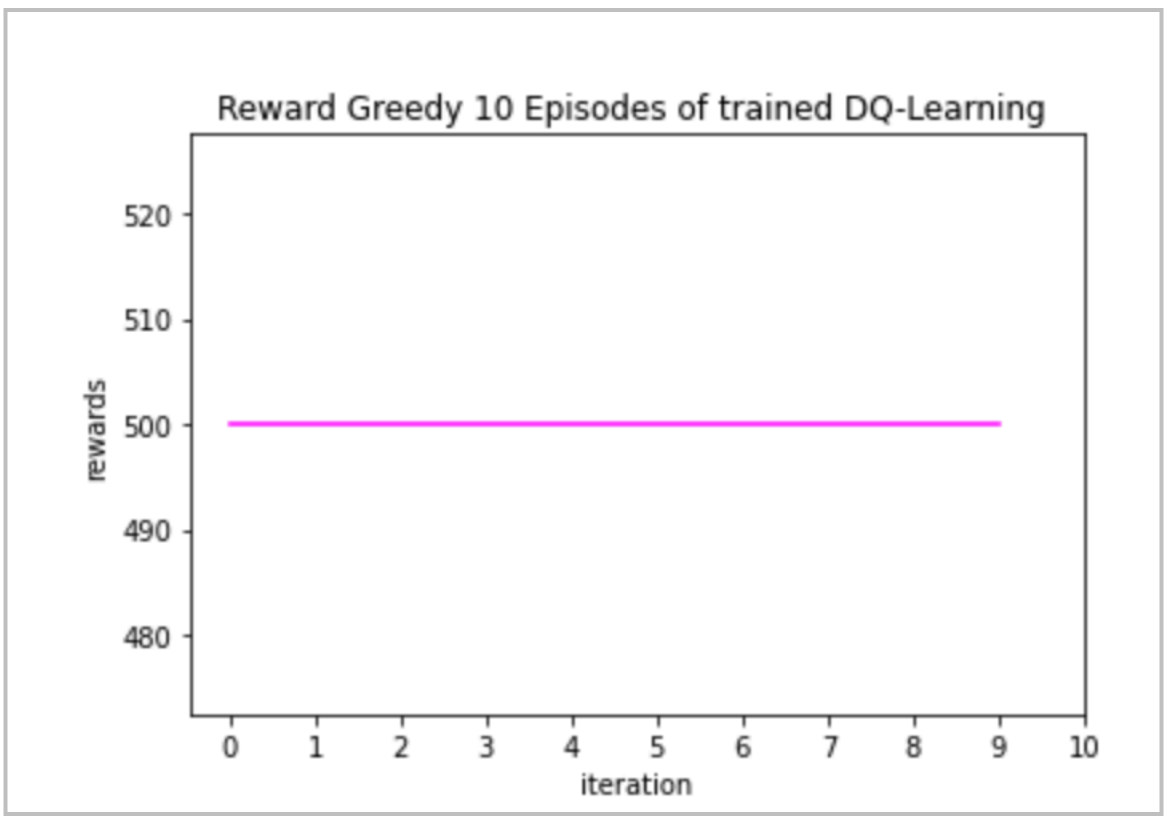


Figure 7: For DQN on CartPole

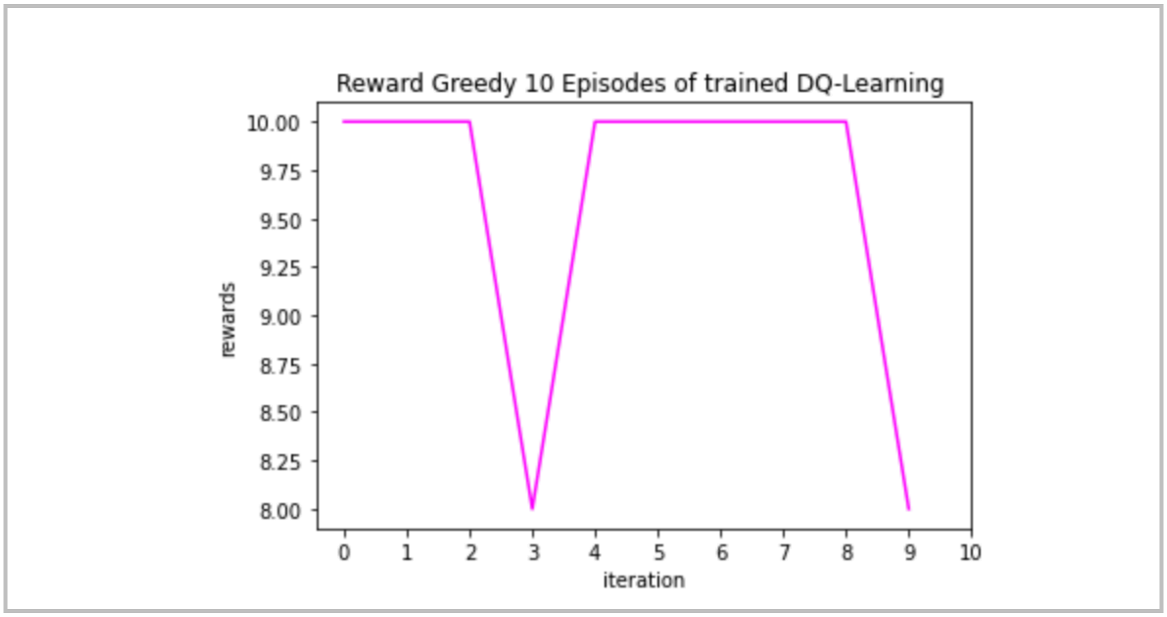


Figure 8: For Double DQN on CartPole

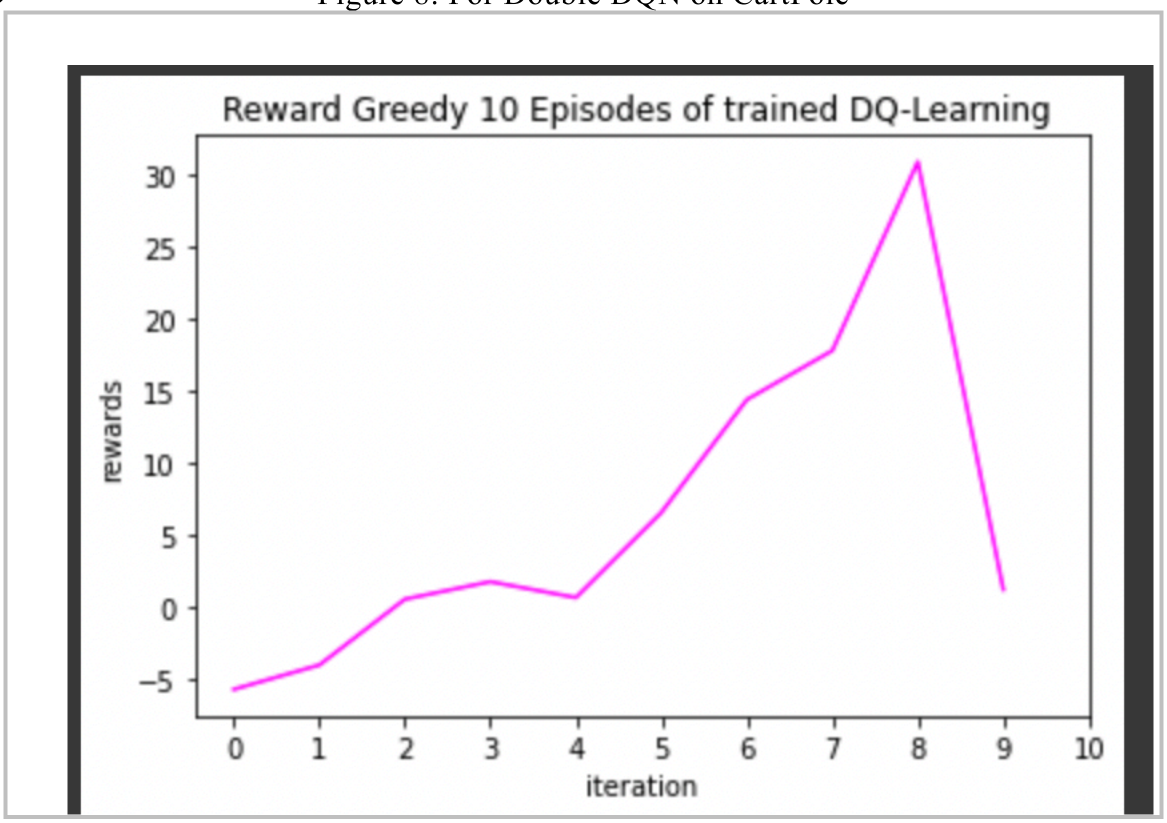


Figure 9: For DQN on Lunar Lander

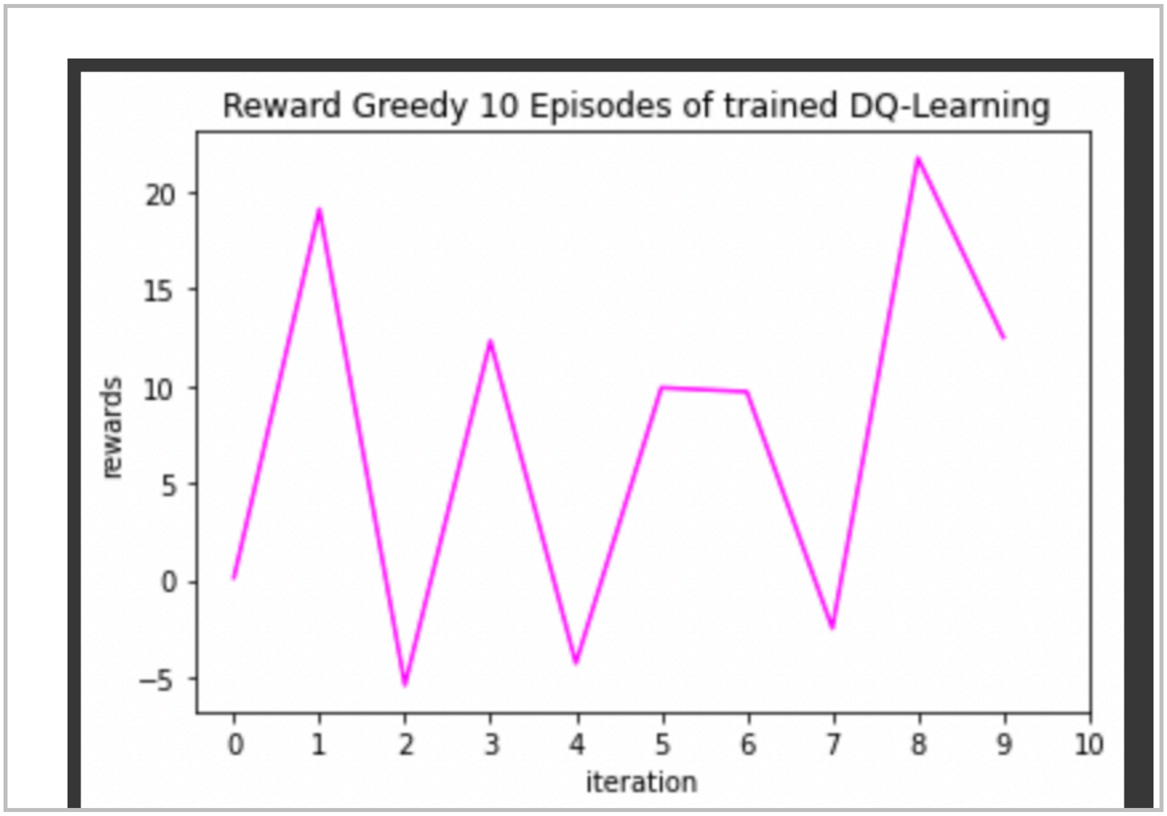


Figure 10: For Double DQN on Lunar Lander

7 Compare the performance of both algorithms (DQN & Improved version of DQN) on the same environments (e.g. show one graph with two reward dynamics) and provide your interpretation of the results. Overall three rewards dynamics plots with results from two algorithms applied on:

- **Grid-world environment**

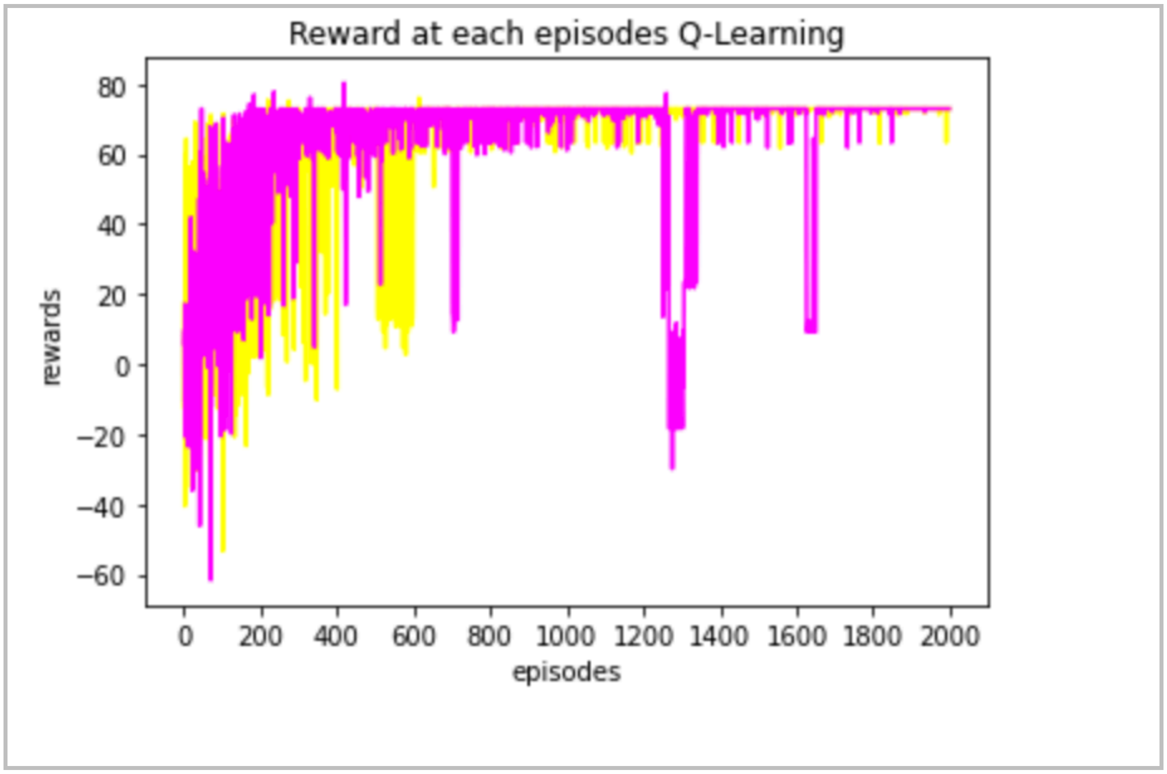


Figure 11: Pink is Rewards per episode for DQN on Grid environment

Figure 12:Yellow is Rewards per episode for Double DQN on Grid Environment

The DQN graph seems to be fluctuating and looks like the model isn’t learning properly, on the other hand, the Double DQN seems to be in a better condition with a clear reward exposure in the beginning and a somewhat constant behaviour later.

- **’CartPole-v1’**

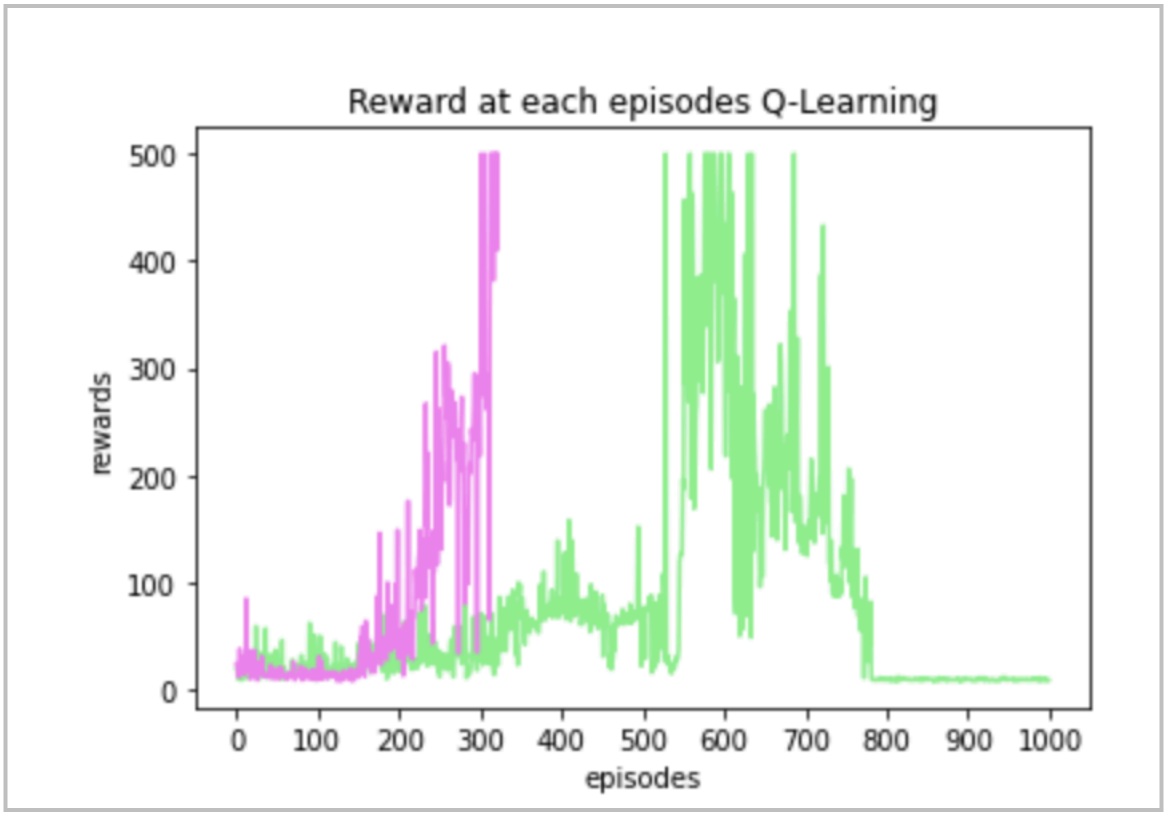


Figure 13: Pink Reward per episode for DQN on cartpole

Figure 13: Green Reward per episode for Double DQN on cartpole

Double DQN seems to have fluctuated in the second half episodes after 500 where it starting catching big rewards but was brought down either by the target Q model or the big Buffer size.

- **Lunar Lander**

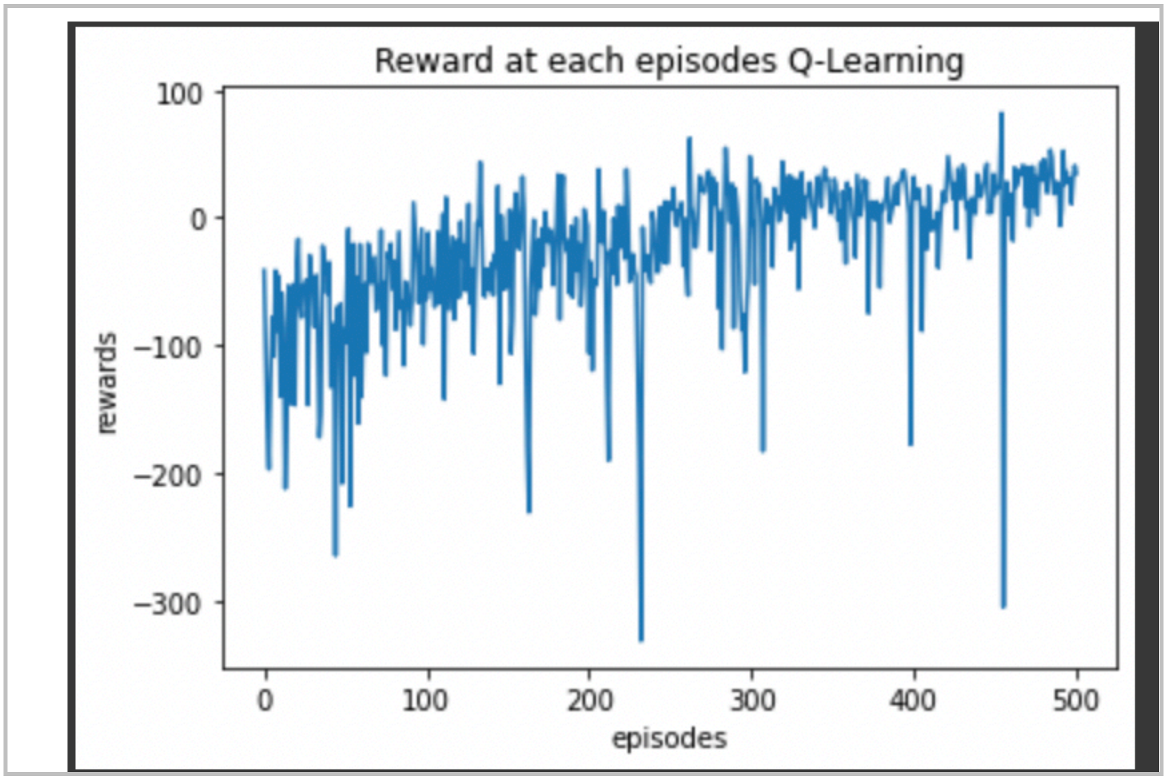


Figure 15: Reward per episode for DQN on Lunar Lander

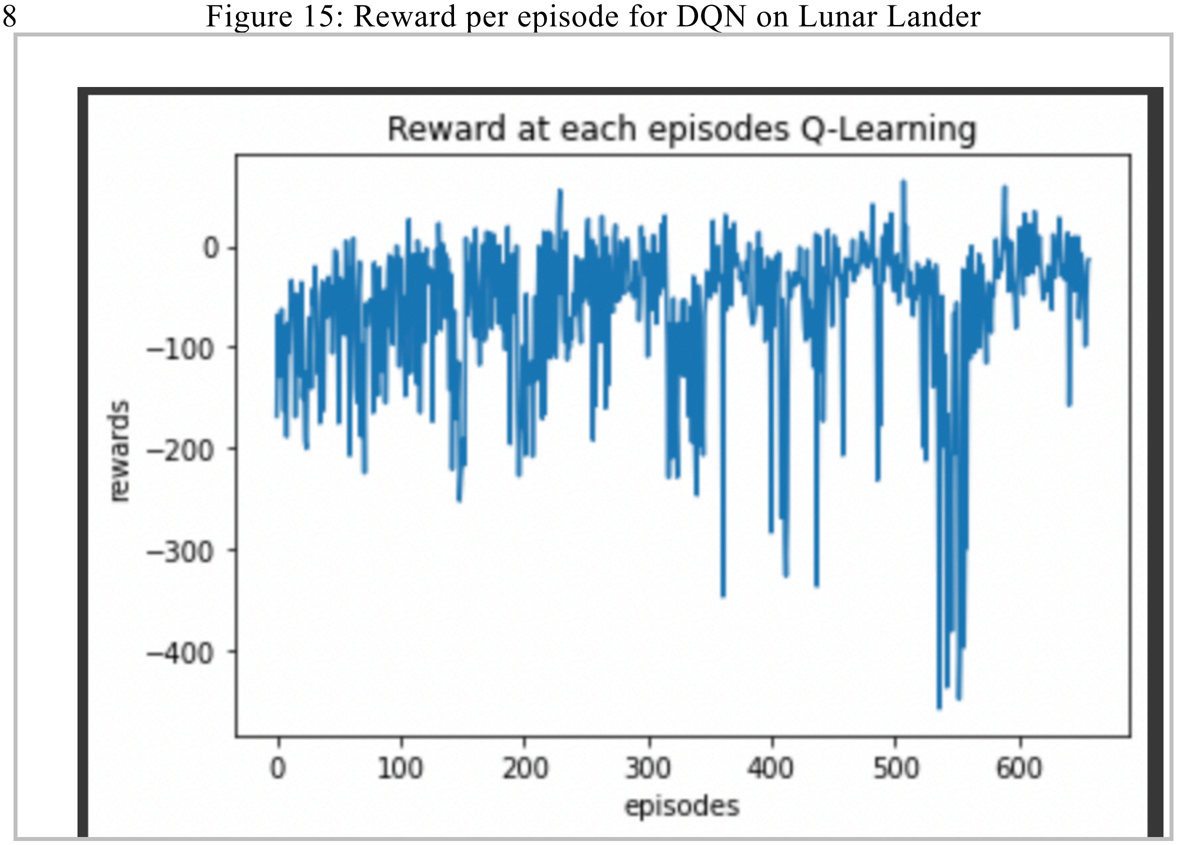


Figure 16: Reward per episode for Double DQN on Lunar Lander

For DQN, the model seems to be learning at a good rate and accumulating rewards. For Double DQN, it seems that the model is fluctuating and would need some hyperparameter tuning for a slow and robust learning.

8 Provide your interpretation of the results. E.g. how the same algorithm behaves on different environ- ments, or how various algorithms behave on the same environment.

It seems that Double DQN is giving better results by not localizing too early compared to normal DQN for the grid environment. DQN seems to need different hyperparameters to train compared to double DQN for all environments. Double DQN, which is dependant on the policy model for target calculation along with target model unlike normal DQN seems to be very sensitive compared to normal DQN. For Lunar Lander and CartPole, a better combination of hyperparameters will give us better results.

Contribution:

Team Member	Assignment Part	Contribution
Suraj	Part1,2,3	50%
Sumeet	Part1,2,3	50%

References

[1] https://gym.openai.com/docs/

[2] https://gym.openai.com/envs/LunarLander-v2/

[3] https://gym.openai.com/envs/CartPole-v0/

[3] https://web.stanford.edu/class/psych209/Readings/MnihEtAlHassibis15NatureControlDeepRL.pdf

[4] Playing Atari with Deep Reinforcement Learning – Volodymyr eta.

[5] https://medium.datadriveninvestor.com/training-the-lunar-lander-agent-with-deep-q-learning-and-its-variants-2f7ba63e822c