CSE 4/546: Reinforcement Learning

Spring 2022.

Assignment 2 - Deep Q-Networks

Checkpoint: March 27, Sun, 11:59pm.

Team 2:

Affan Ali Hasan Khan-50432243

Pritisman Kar- 50388920

Part I [10 points] - Exploring OpenAI Gym Environments:

Install OpenAI gym library and explore the environments and main functions. and one other environment that you use for the assignment (e.g. possible actions/states, goal, rewards, etc).

CartPole-v1:

- The CartPole environment basically has a cart which can move left or right and it has to balance the pole which is placed on top of the cart.

- Possible actions:
  The cart has only two possible actions '0' or '1' which moves the cart left or right to keep the pole up.

- Observations/States:
  The states/observations are represented in a form of vector having 4 elements.
  The elements are cart position, cart velocity, pole angle, and pole velocity at tip.
  All the elements have float values.

- Goal:
  Everytime the environment is reset the pole starts upright and the goal of the environment is to prevent the pole from falling.

- Episode end/ Done:
  The episode ends/ Done='True' if the pole is more than 15 degrees from vertical, or the cart moves more than 2.4 units from the center or else the episode doesn't ends/ Done='False'.

- Rewards:
  A reward pf '+1' is provided for everytimestep the pole remains up.

MountainCar-v0:

- The environment has a car between two hills, the car must move to reach the flag which is on top of the right hill.

- Possible actions:
  There are 3 possible actions ‘0’, ‘1’, and ‘2’ that represents push left, do nothing, push right.

- Observations/States:
  The observation is a vector of 2 numbers representing velocity and position of the car.
  The values are in float for all elements.

- Goal:
  Everytime the environment is reset the car is placed at the middle of the hills and the goal is to reach the flag at which is at ‘0.5’ position.

- Episode end/ Done:
  The episode ends/ Done=’True’ if the car reaches the flag or else the episode doesn’t ends/ Done=’False’.

- Rewards:
  The car receives a reward of ‘-1’ for everytimestep until it reaches the goal.

Part II [40 points] - Implementing DQN & Solving grid-world environment:

1. **Discuss the benefits of:**

   • Using experience replay in DQN and how its size can influence the results

   It is used to remove the dependency on the correlation (make the learning more independent) between consecutive samples as this could lead to degrade the learning process. It also helps the network to learn for a single sample multiple times and reduces the chance of recall rare occurrence.

   If the size of the replay buffer is too small, it increases the chance of the prediction to be highly correlated as mentioned in the above paragraph and if the size of the replay buffer is too high, there a chance the data is too old.

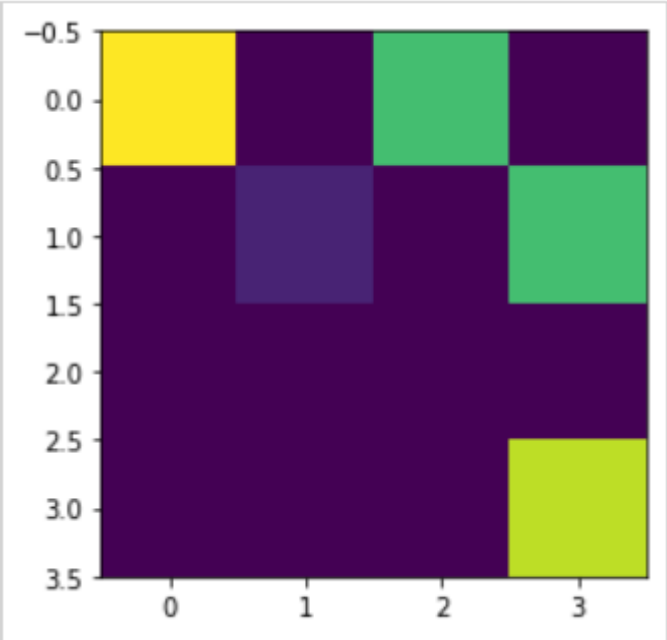
   • **Introducing the target network**

   The target network is updated every N iteration and not for every iteration/steps. By introducing this, it makes the learning more stable. It will provide the network to learn in the C iteration and then the target network is updated from the online net.

   • **Representing the Q function as qˆ(s, w)**

   Representing a (state-action/state) value function with a parameterized function instead of a table as in a q value table every state action pair s, a has an entry Q(s, a), while representing the q value with a parameterized function helps solving problem with large MDPs in which there are too many states and/or actions to store in memory and it is too slow to learn the value of each state individually. I gives a compact representation that generalizes across state or states and actions, and also generalise from seen states to unseen states.

2. **Briefly describe the grid-world environment that you used (e.g. possible actions, states, agent, goal, rewards, etc). You can reuse related parts from your Assignment 1 report.**



Start Position	Reward: -0.8	Reward: +0.4	Reward: -0.8
Reward: -0.8	Reward: -0.5	Reward: -0.8	Reward: +0.4
Reward: -0.8	Reward: -0.8	Reward: -0.8	Reward: -0.8
Reward: -0.8	Reward: -0.8	Reward: -0.8	Goal Position – Reward +1

- Set of actions: (Up, Down, right, left)

- Set of states: There are 16 possible states, each state is a block in a 4x4 matrix blocks. There is the start state where the agent starts every time its reset. We have the end state 'Goal' where if the agent reaches the agent stops, this state is the main objective of the agent. There are 14 other states where we have random rewards for some states.

- Set of rewards: (-0.5, -0.8, +1, +0.4)

- Main objective: This is the main objective of the agent to achieve and if the agent achieves this the agent stops, for our environment I've defined the main objective as 'Goal' state with a +0.3 reward.

3. **Show and discuss your results after applying your DQN implementation on the environment. Plots should include epsilon decay and the reward per episode.**

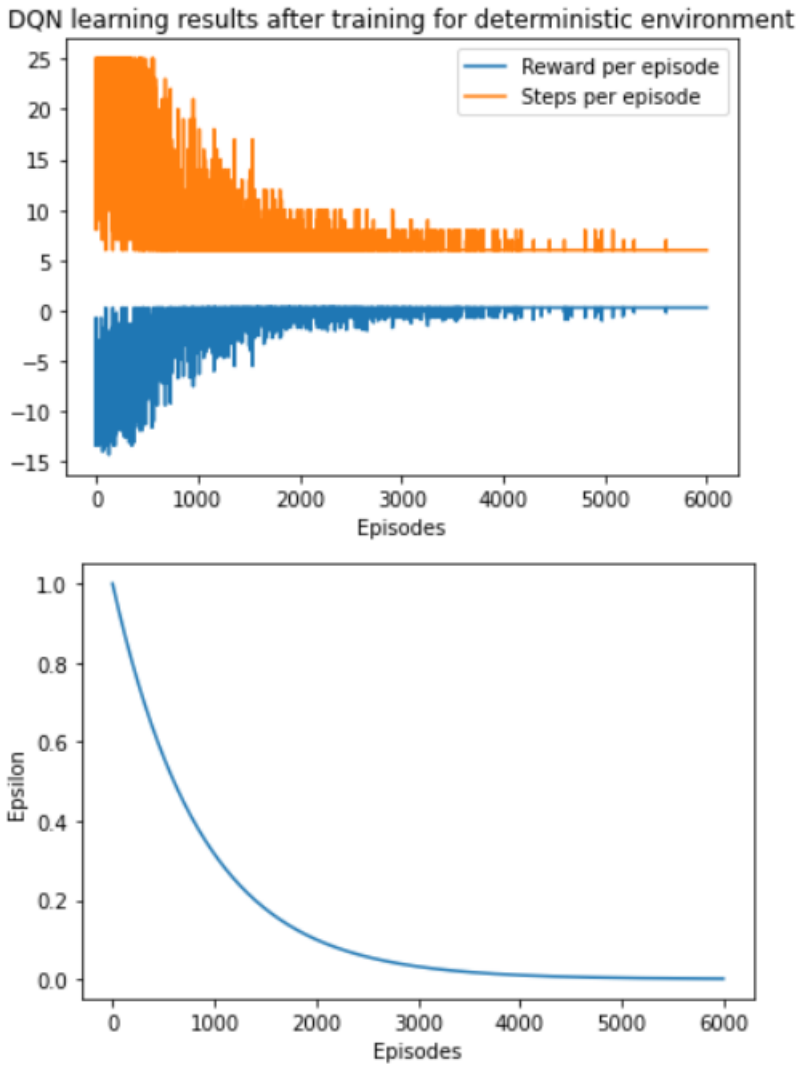


Fig: (a) rewards per episode (b) Epsilon decay

As you can see from the above graph, the cumulative reward per episode stabilises (maximum possible reward) towards the end. The epsilon decay is reducing exponentially as depicted in the above graph.

4. **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.**

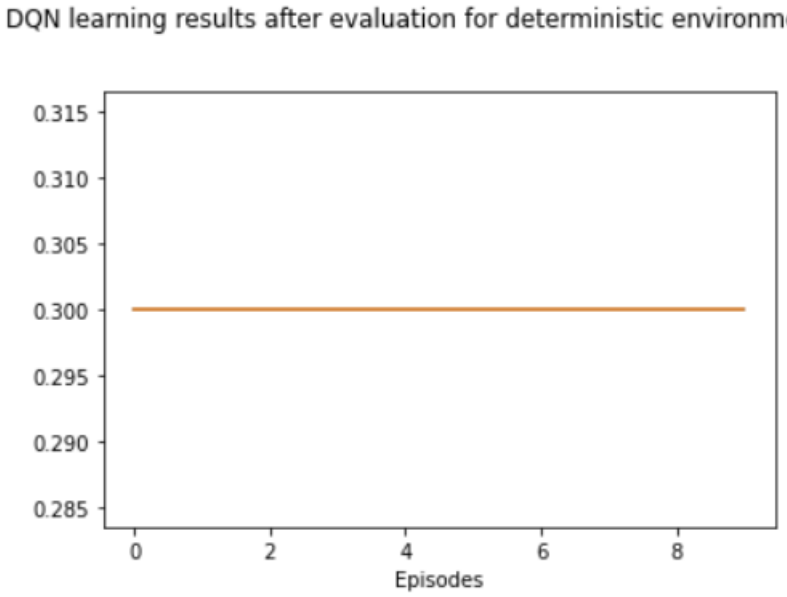


Fig: Evaluation result

Part III [50 points] - Improving DQN & Solving OpenAI Gym Environments

1) **Discuss the algorithm you implemented.**

The algorithm selected for improvement over the vanilla DQN is Double DQN. Below is a brief discussion on the algorithm:

- Firstly, the main difference between a DQN and a double DQN is that it uses two estimators instead of just one as used in DQN:
    - Estimator Q1: Obtain best actions.
    - Estimator Q2: Evaluate Q for the above action.

- The architecture of the algorithm is same as DQN, but double DQN updates the value of Q in a similar fashion as Double Q learning.

$$Q^*(s_t, a_t) = r_t + \gamma Q_{\theta'}(s_{t+1}, \arg\max_{a'} Q_\theta(s_{t+1}, a'))$$

- By using two Q estimators Double DQN successfully reduces the problem of overoptimism, resulting in more stable and reliable learning.

2) **What is the main improvement over the vanilla DQN?**

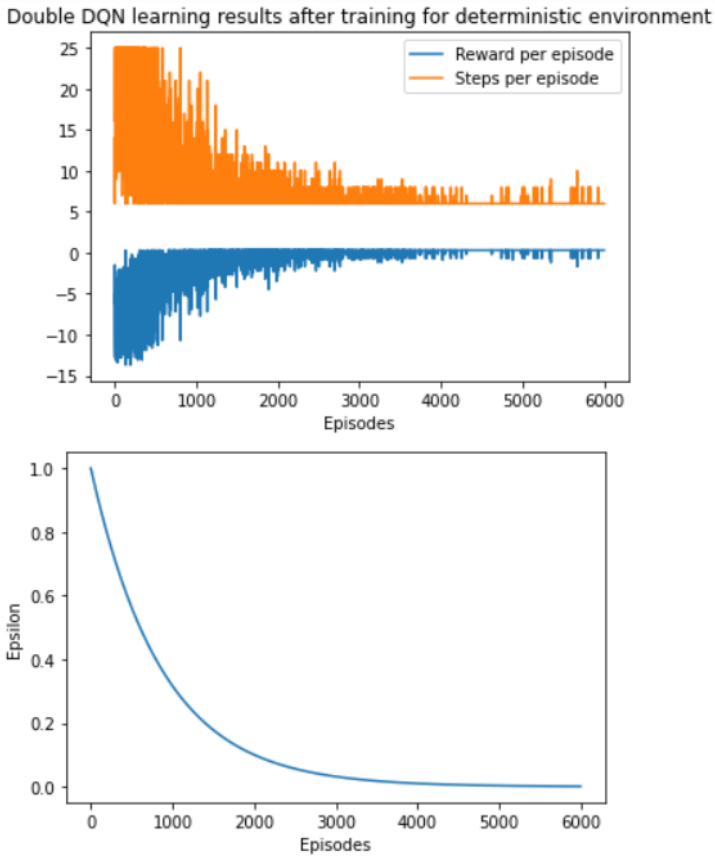
- In the vanilla DQN there is what is known as a maximisation bias, and If you slightly overestimate your Q-value then this error gets compounded and resulting in an unstable learning.

- In double DQN the idea is that if you overshoot your estimate on one Q network then having the second will hopefully control this bias when you would take the max.

- Therefore, Double DQN can be used at scale to successfully reduce this overoptimism, resulting in more stable and reliable learning, using the existing architecture and deep neural network of the DQN algorithm without requiring additional networks or parameters.

3) **<u>Show and discuss your results after applying the two algorithms implementation on the environment. Plots should include epsilon decay and the reward per episode.</u>**

Grid World:

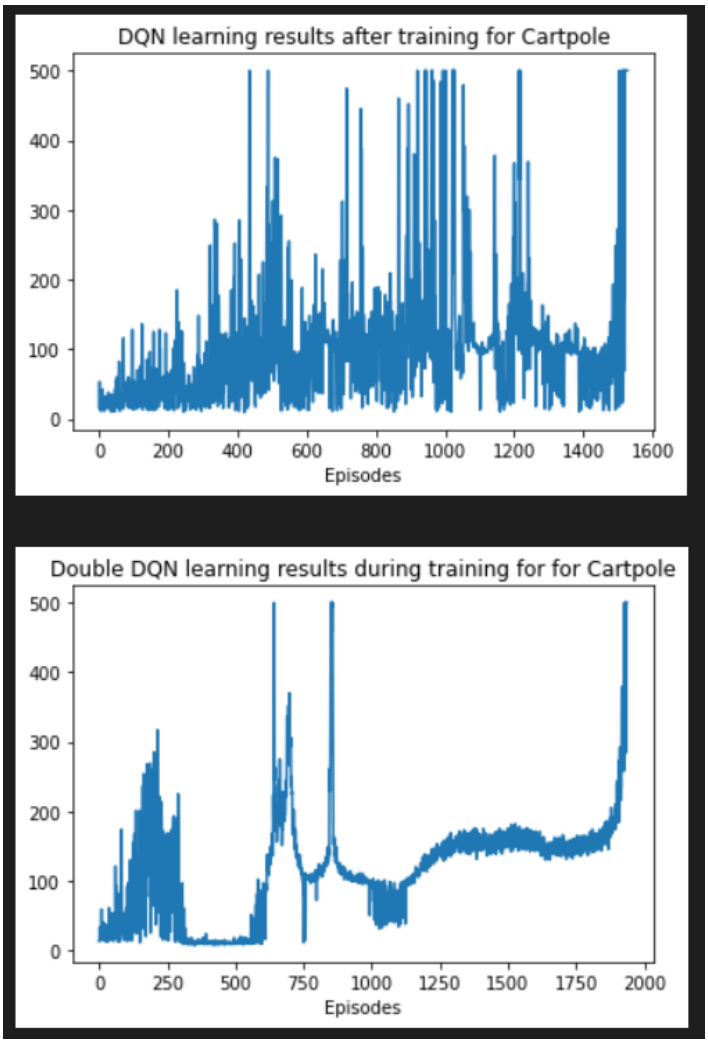


- The above graphs shows the implementation of Double DQN on the grid environment.

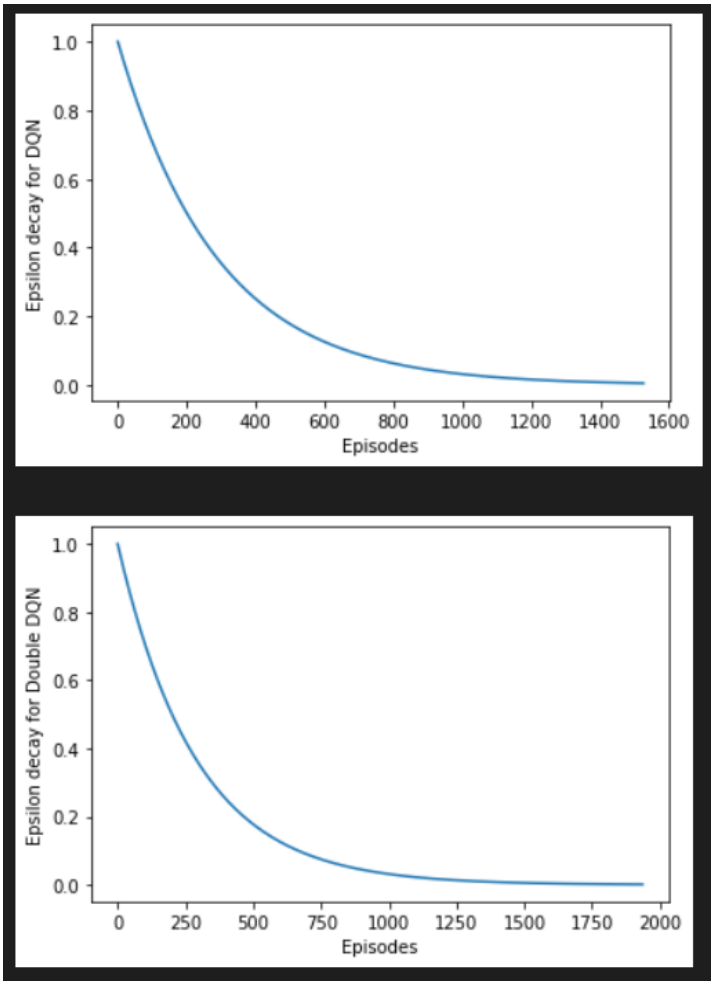
- From the first plot we can see as the training progresses the agent tries to maximize the reward and reduce the number of steps it takes to complete and episode, and from the graph it is evident that the agent takes the maximum reward with less number of steps.

- The second plot shows the epsilon decay as the episode proceeds which starts from 1 and goes to 0.001.

Cart Pole:

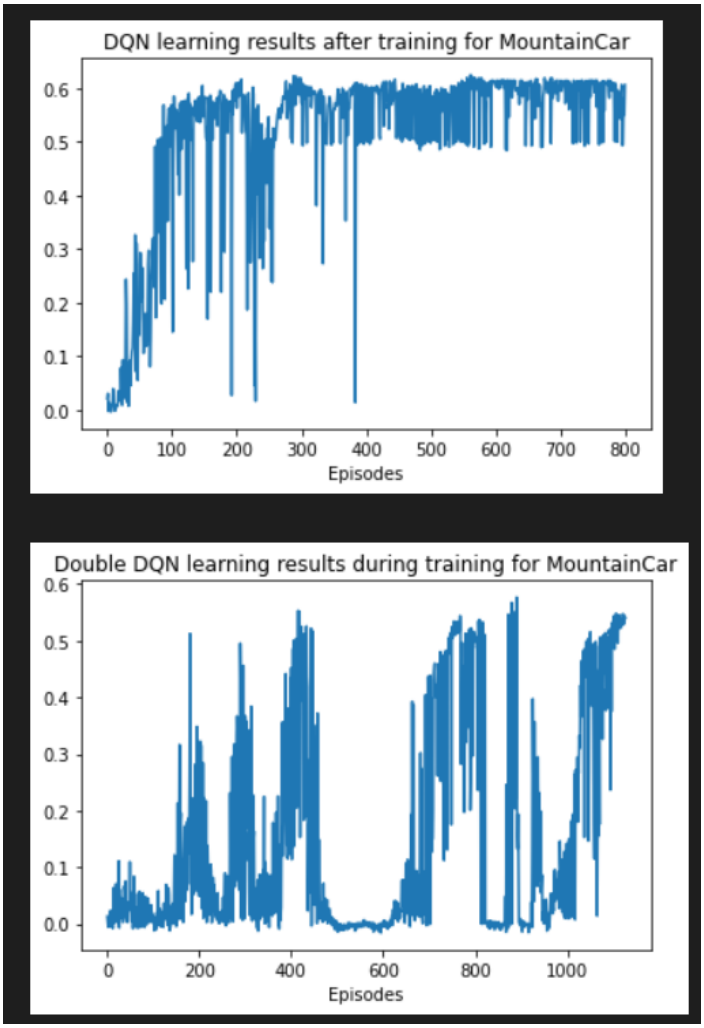


- The above graphs show the implementation of DQN and Double DQN on the Cart Pole gym environment.
- Here as soon as the average reward for the last 10 episodes is more than 470, training is stop and the trained model is evaluated.
- From the first plot we can see as the training progresses the agent tries to maximize the reward and reduce the number of steps it takes to complete and episode, and from the graph it is evident that the agent takes the maximum reward with a smaller number of steps.

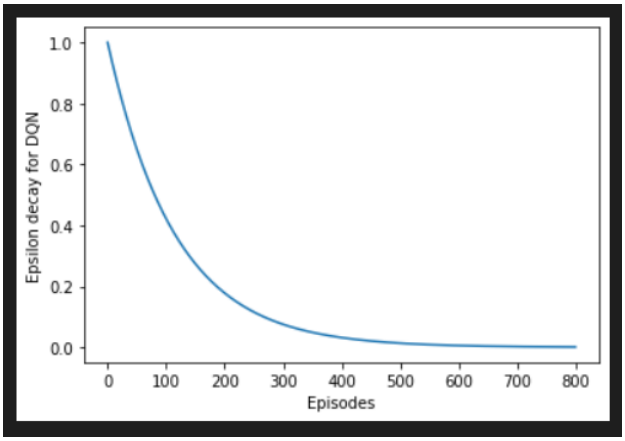


The second plot shows the epsilon decay as the episode proceeds which starts from 1 and goes to 0.001.

Mountain Car:



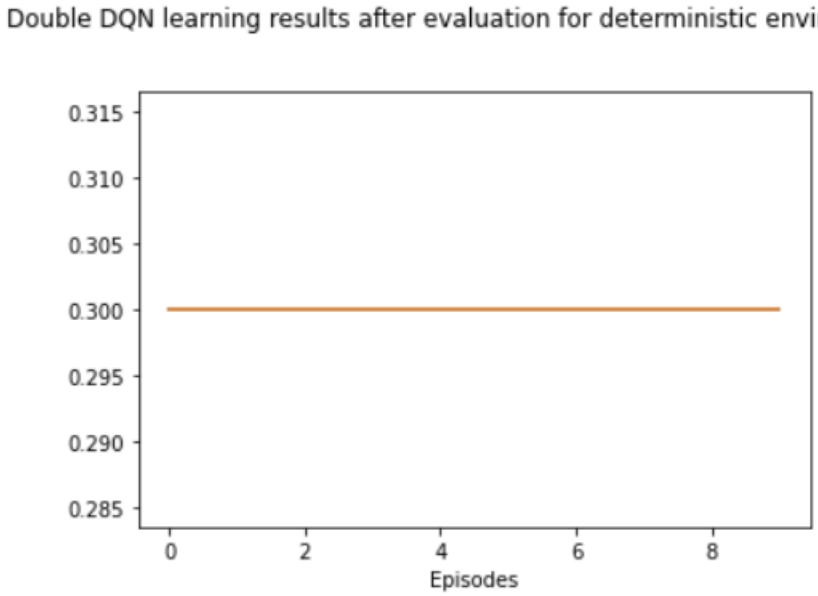
- The above graphs show the implementation of DQN and Double DQN on Mountain Car gym environment.
- Here for the DQN environment the model is getting trained from 500 episodes onwards whereas Double DQN is taking longer. For Double DQN, the training is stopped as soon as the average reward for the last 40 episodes is more than 0.51.
- From the first plot we can see as the training progresses the agent tries to maximize the reward and reduce the number of steps it takes to complete and episode, and from the graph it is evident that the agent takes the maximum reward with a smaller number of steps.



The second plot shows the epsilon decay(for DQN) as the episode proceeds which starts from 1 and goes to 0.001.

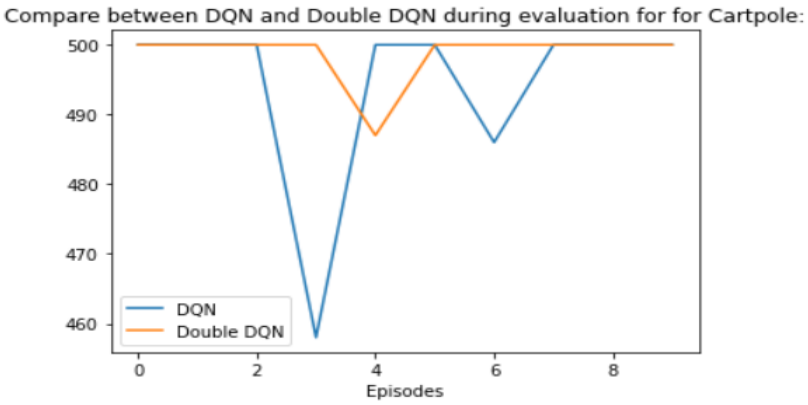
4) **Provide the evaluation results. Run your environment for at least 5 episodes, where the agent chooses only greedy actions from the learnt policy. Plot should include the total reward per episode.**

Grid World

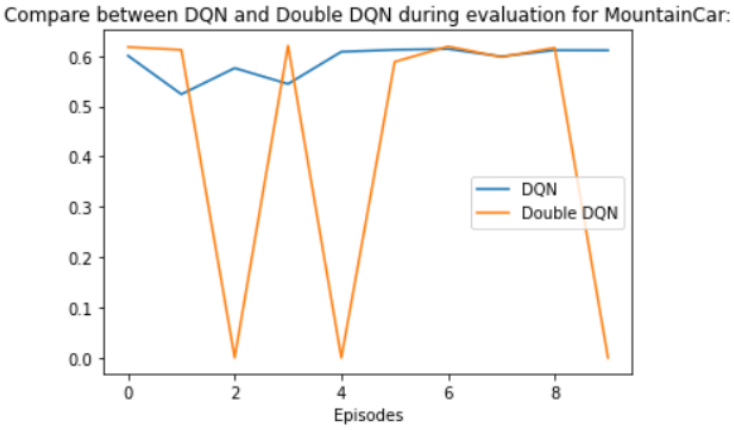


- For the evaluation the agent takes only greedy actions from the learnt policy and as we can see every time it achieves the maximum cumulative reward on each episode.

Cart Pole

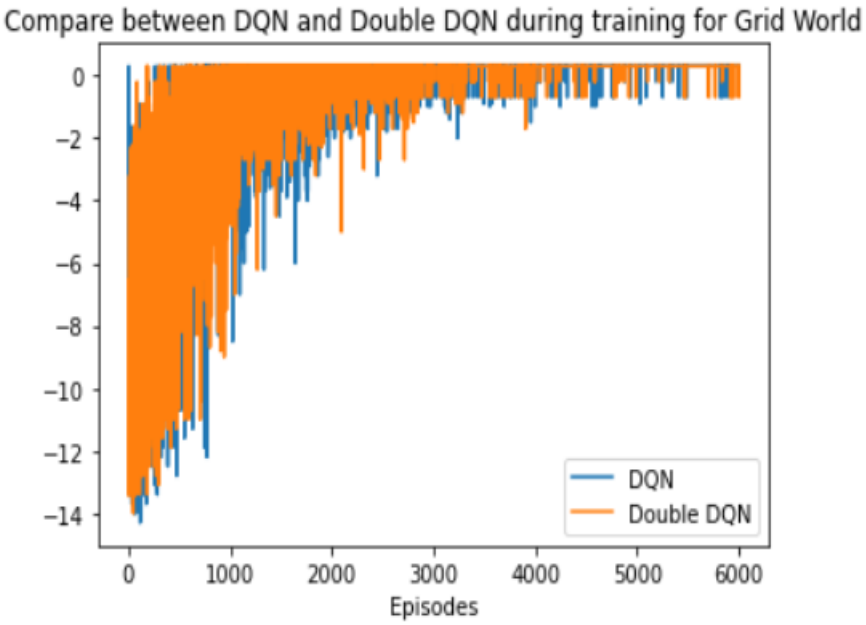


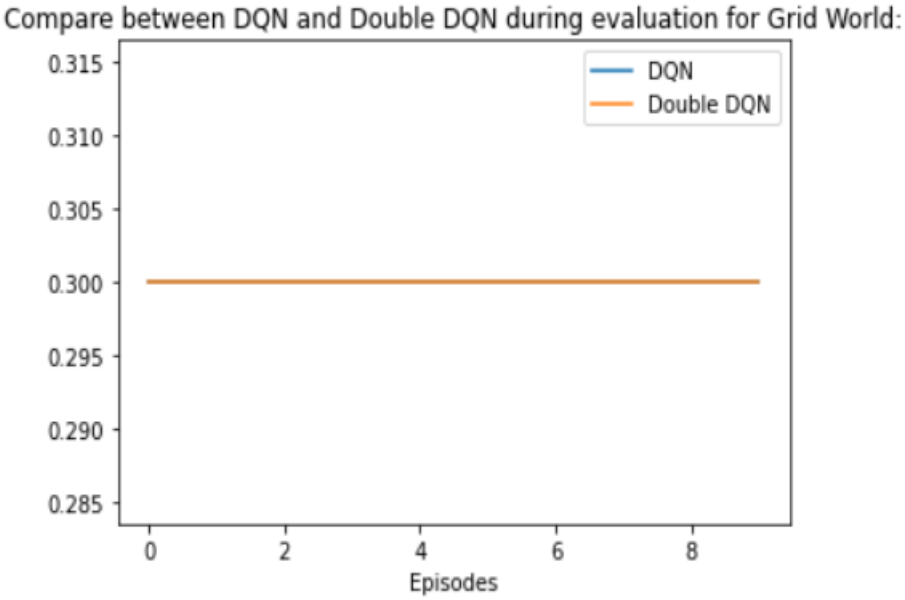
Mountain Car



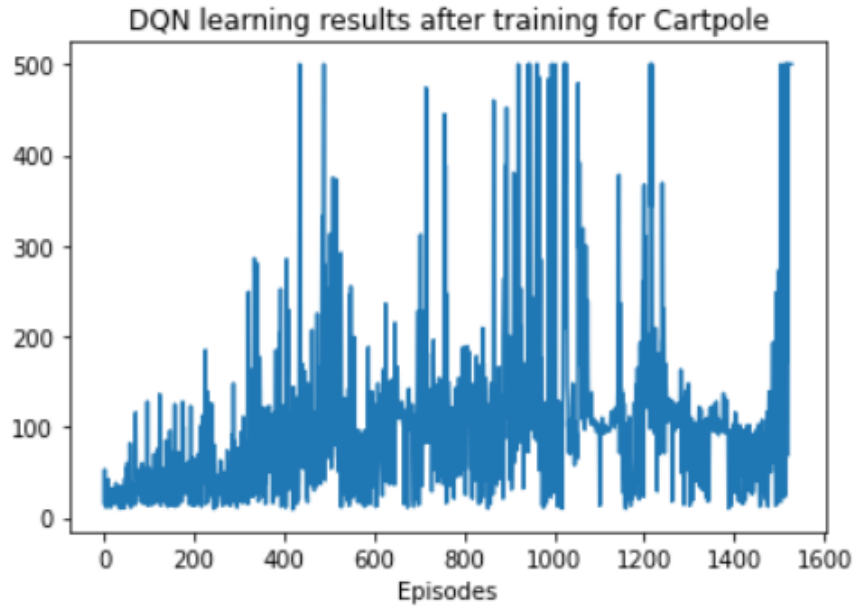
5) **Compare the performance of both algorithms (DQN & Improved version of DQN) on the same environments (e.g. show one graph with two reward dynamics) and provide your interpretation of the results. Overall three rewards dynamics plots with results from two algorithms applied on:**

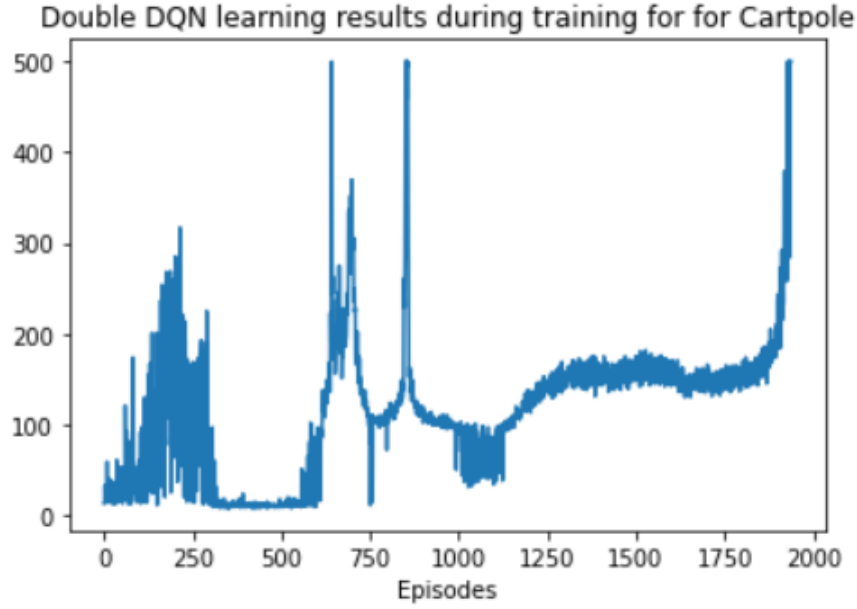
- **Grid-world environment:**

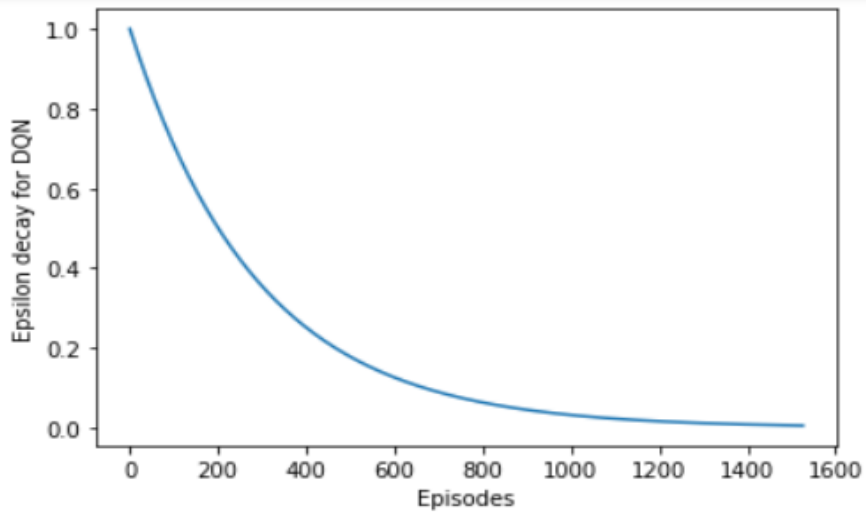

Compare between DQN and Double DQN during training for Grid World
0
−2
−4
−6
−8
−10
−12
−14
0
1000
2000
3000
4000
5000
6000
Episodes
DQN
Double DQN

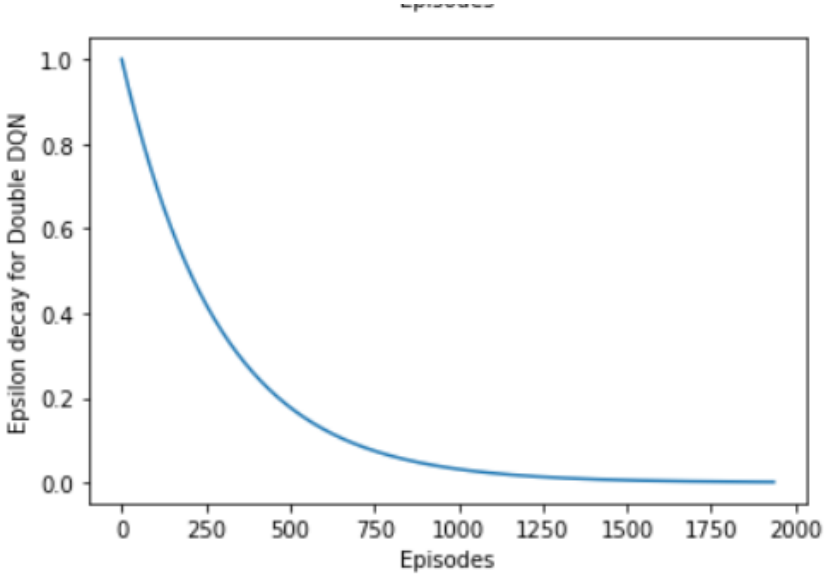

Compare between DQN and Double DQN during evaluation for Grid World:
0.315
0.310
0.305
0.300
0.295
0.290
0.285
0
2
4
6
8
Episodes
DQN
Double DQN

- **CartPole-v1:**

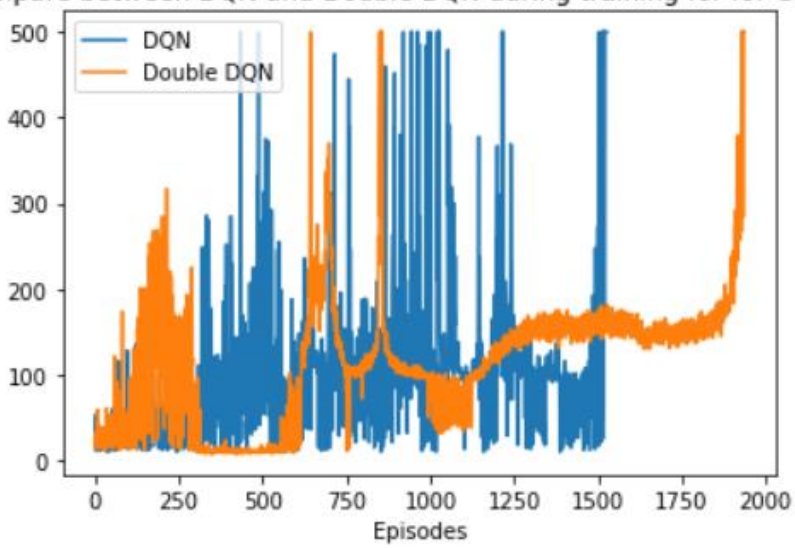

DQN learning results after training for Cartpole
500
400
300
200
100
0
0
200
400
600
800
1000
1200
1400
1600
Episodes


Double DQN learning results during training for for Cartpole
500
400
300
200
100
0
0
250
500
750
1000
1250
1500
1750
2000
Episodes

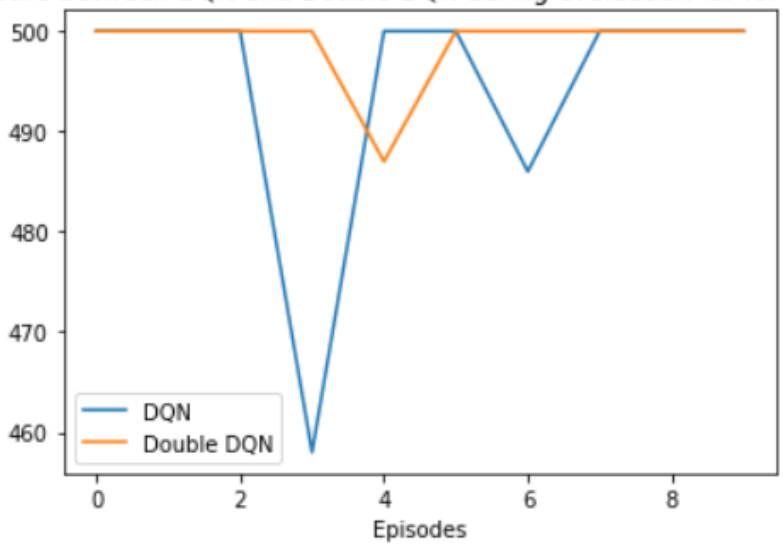

Epsilon decay for DQN
1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000
1200
1400
1600
Episodes



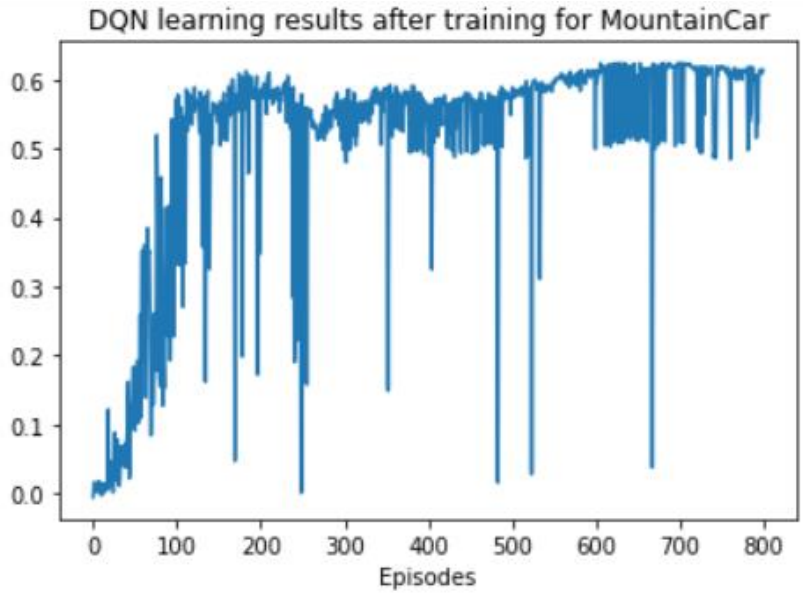
Compare between DQN and Double DQN during training for for Cartpole

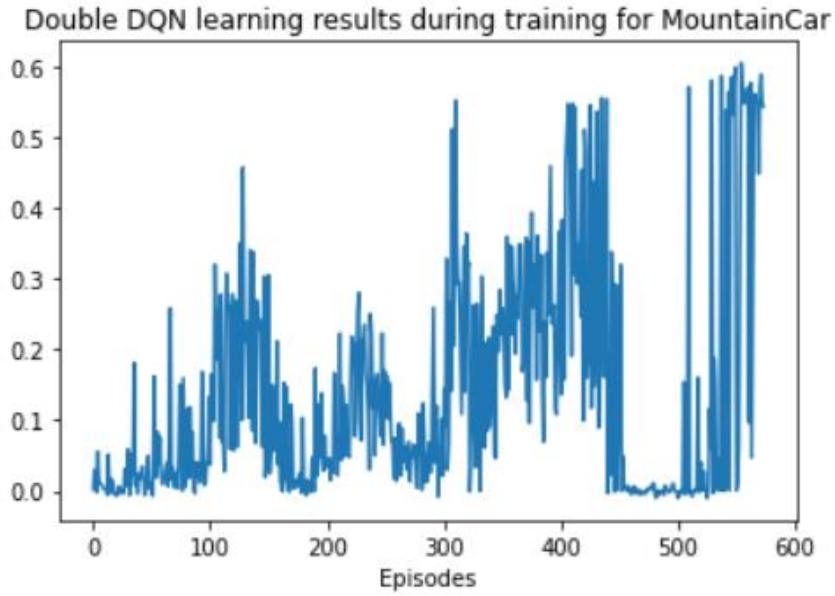


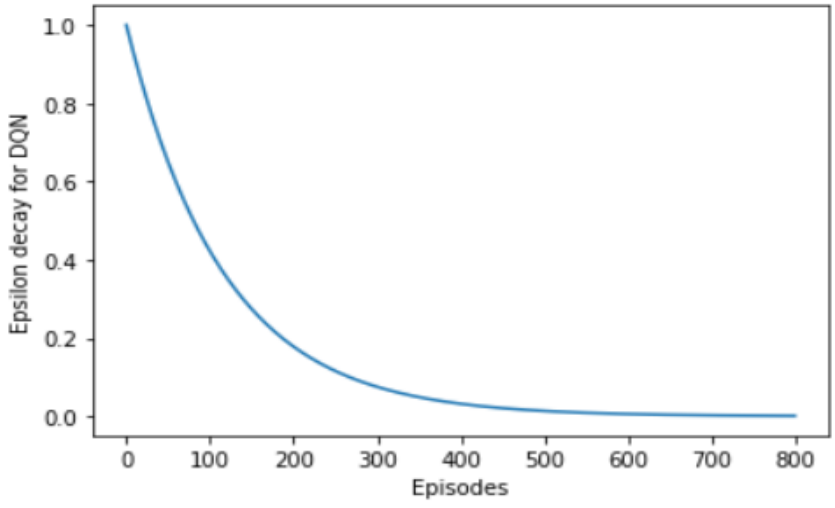
Compare between DQN and Double DQN during evaluation for for Cartpole:

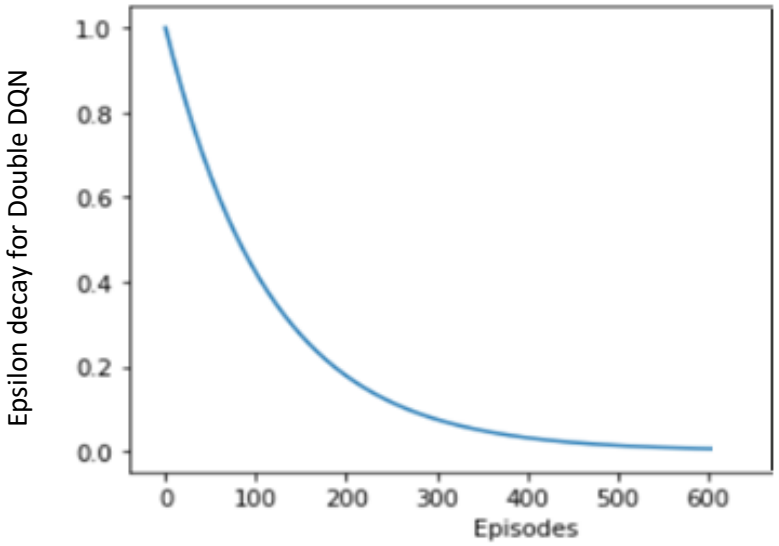


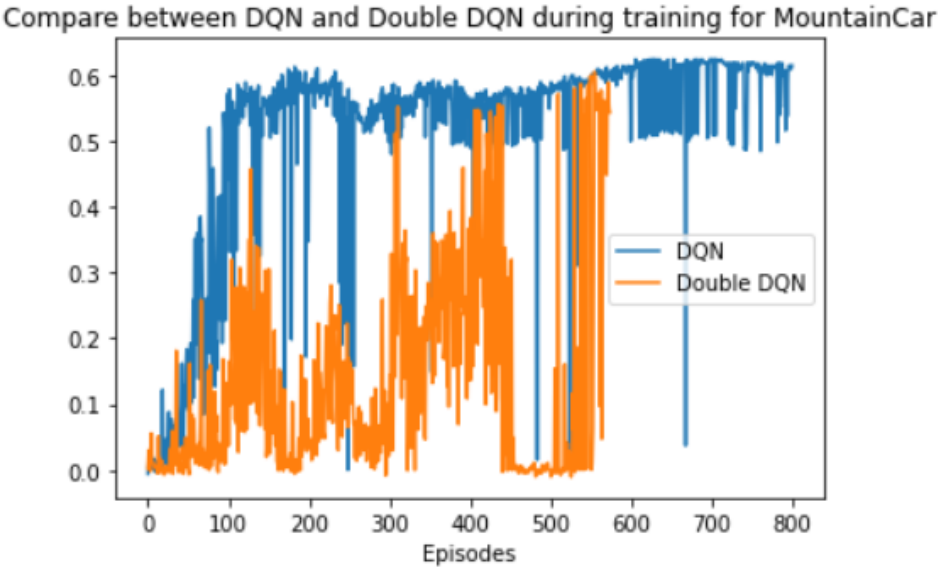
- **For Mountain car:**

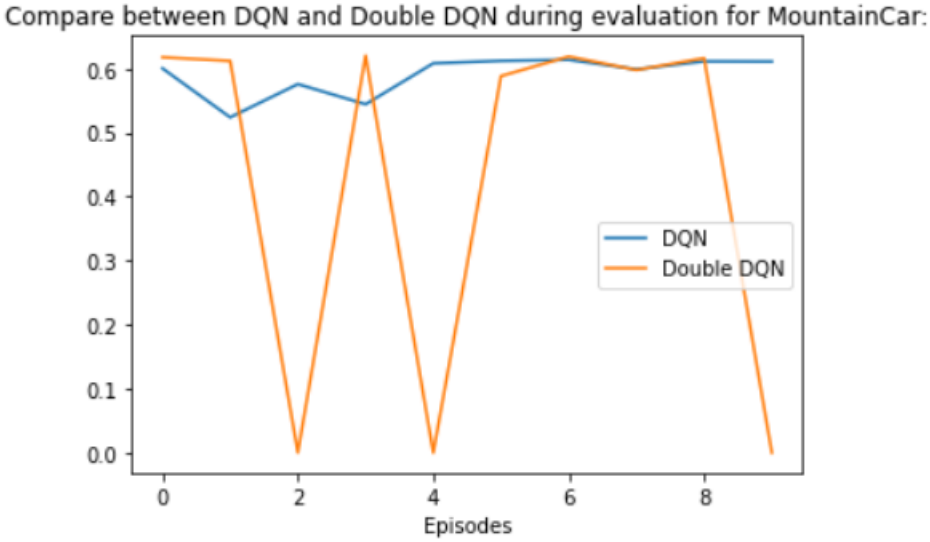







Epsilon decay for Double DQN
1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
500
600
Episodes


Compare between DQN and Double DQN during training for MountainCar
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
100
200
300
400
500
600
700
800
Episodes
DQN
Double DQN


Compare between DQN and Double DQN during evaluation for MountainCar:
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
2
4
6
8
Episodes
DQN
Double DQN

6) Provide your interpretation of the results. E.g. how the same algorithm behaves on different environments, or how various algorithms behave on the same environment.

- **Grid world:**

    - For the grid world environment both the DQN and DDQN gives almost the same result that is the agent maximizes the cumulative reward at the end of training and the evaluation results are also the same.
    - Therefore, for the grid world we can say that both the algorithms are working fine.

- **Cartpole:**

    - For Cartpole both the DQN and DDQN reward plots are a little unstable that is the cumulative reward per episode goes up and down but at the end of the training both the algorithm are successfully able to reach a reward of 470 for last 10 episode.
    - Although both the algorithms are reaching the max rewards but DQN finishes the training in less no of training episode as compared to DDQN.
    - If we see in the evaluation plot the DDQN algorithm is working slightly better than DQN as in DDQN in all 10 episode the agent can achieve the max reward and only fails once but in DQN evaluation the agent fails twice in 10 episodes.
    - Both the algorithms are working fine although DDQN takes a slight edge over DQN.

- **Mountain Car:**

    - For Mountain car also both the DQN and DDQN reward plots are a little unstable that is the cumulative reward per episode goes up and down but at the end of the training both the algorithm are successfully able to reach a max reward of 0.6 for last 10 episode.
    - Although both the algorithms are reaching the max rewards but DDQN finishes the training in less no of training episode as compared to DQN.
    - In the evaluation plot DQN is performing better than DDQN as it is always getting the max cumulative reward for all 10 evaluation episodes, while in DDQN during evaluation the agent fails to achieve the max reward for 3 times out of 10 evaluation episodes.
    - Both the algorithms are working well for Mountain car but DQN takes a slight edge than DDQN.