Reinforcement Learning
Assignment 2(Final Report)
CSE 546

Team members-
Sannihith Kilaru(Ubit name-sannihit)
Sannihitha Gangina(Ubit name-srisanni)

1. Discuss the benefits of:

• Using experience replay in DQN and how its size can influence the results • Introducing the target network • Representing the Q function as qˆ(s, w)

- Making better use of prior experience by learning with it many times. When training a function approximator, the convergence behavior is improved. Because Q-learning updates are incremental and do not converge quickly, several runs with the same data are advantageous, especially when the variance in immediate outcomes (reward, next state) is modest given the same state, action combination.

- Q target is calculated using Q network, the network keeps changing and the Q values are being updated at every timestep which makes us use Q target. The Q values are generated and are computed to loss for an action in a network.

- We use the weights for the function approximation. The estimation of the value function is done by function approximation.
  $\hat{q}(s, a, w) \approx q_\pi(s, a)$

  Neural networks and linear feature representations use function approximation. Function approximation helps to give greater reward in an environment, when there are more states DQN with function approximation is used.

2. Briefly describe 'CartPole-v1', the second complex environment, and 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.

Grid Environment-

The main objective of the agent in the deterministic environment that is defined for the 4X4 grid is to reach the goal state which is at (3,3) with an equal probability of an agent moving to another state. The objective of the stochastic environment is to perform well and get the reward. The neighborhood of a node consists of 16 states represented as 4x4 square. In the square the agent gets the rewards of [+1, +3, +5 and +10]. The rewards are perceived at a square if the agent is at that square where the rewards are awarded. The sequence of actions causes the environment to go through a sequence of states, if the sequence is desirable the agent is performed well.

An agent can do the following actions•
An agent can take one step at a time.

• The agent can do the following actions- {move(north), move(south), move(east), move(west).

The start square is (0,0) and the goal state is (3,3) and +10 points are awarded when the agent reaches the goal state.

The number of episodes considered for this environment are 1000

Cartpole-

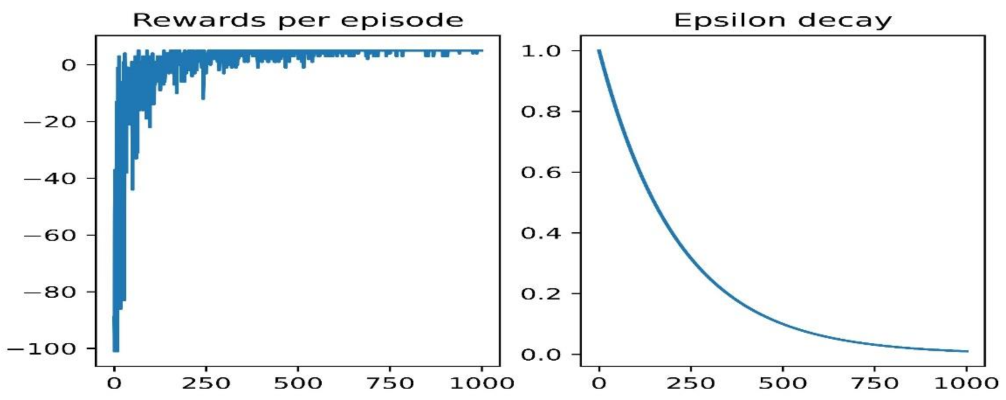
The CartPole-v1 environment from OpenAI is characterized as follows: an unactuated joint connects a pole to a cart that drives along a frictionless track. Controlling the mechanism is as simple as delivering a force of +1 or -1 to the cart.The goal is to keep the pendulum upright and from falling over. When the pole is more than 15 degrees from vertical or the cart is more than 2.4 units from the center, the episode terminates. Every timestep that the pole remains erect results in a +1 reward. CartPole-v1 defines "solving" as receiving an average reward of 500 over 500 episodes. There are two actions for this environment. 0 pushes the cart to the left and 1 pushes the cart to the right. We have four observations which are cart position, cart velocity,pole angle and pole angular velocity. The number of episodes considered for this environment are 8000.

Mountaincar-

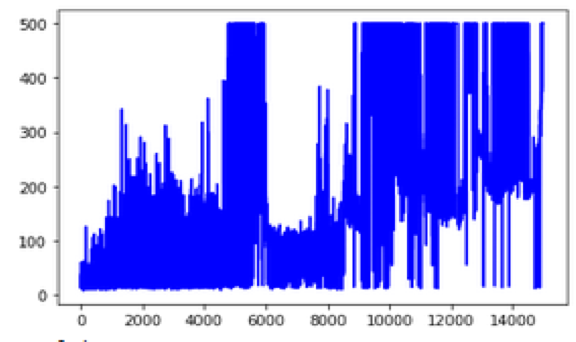
The second complex environment that we chose to design is the mountain car with discrete actions. The automobile is driving on a one-dimensional track between two "mountains." The goal is to drive up the mountain on the right, but the car's engine isn't powerful enough to do so in a single pass. As a result, the only way to succeed is to drive back and forth to gather momentum. The MDP's purpose is to strategically accelerate the car to the goal state at the top of the right hill. There are two observations 0 which is the position of the car along the x axis and 1 which is the velocity of the car. Three actions are considered in this environment 0 is to accelerate to the left, 1 is not to accelerate and 2 is to accelerate to the right. As the goal is to get to the flag on top of the appropriate hill as rapidly as possible, the agent is penalized with a -1 reward for each timestep. The number of episodes considered for this environment are 8000.

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

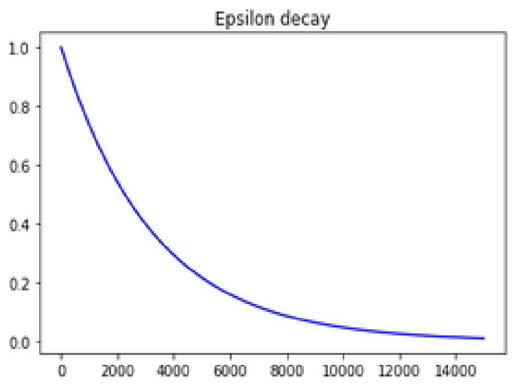
Grid Environment-



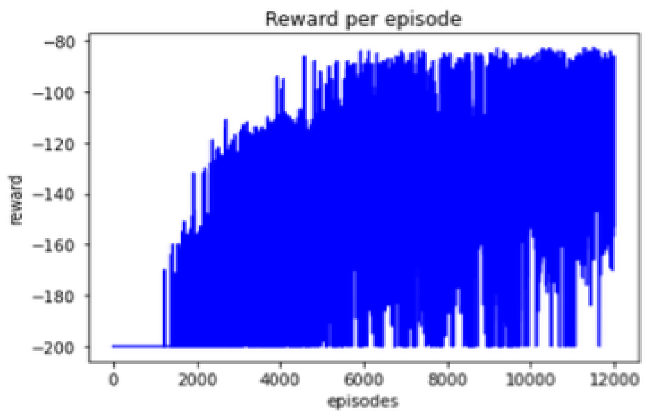
Cartpole-

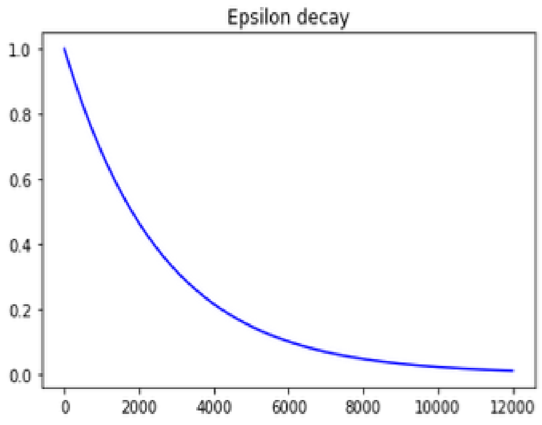


```
Complete
```



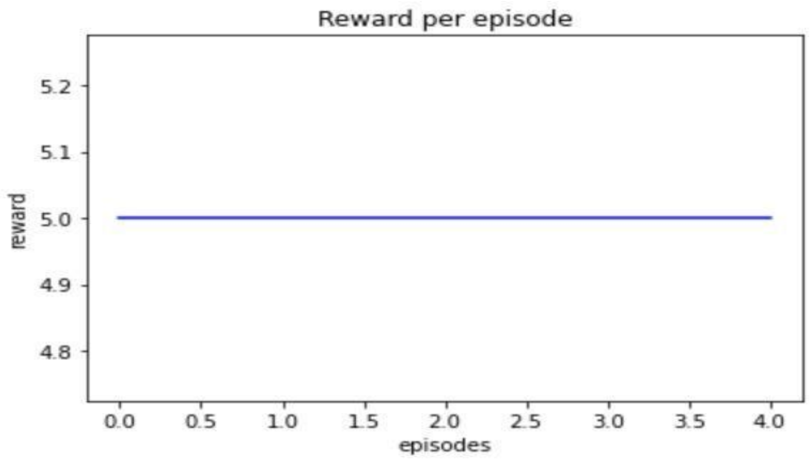
Mountain Car-



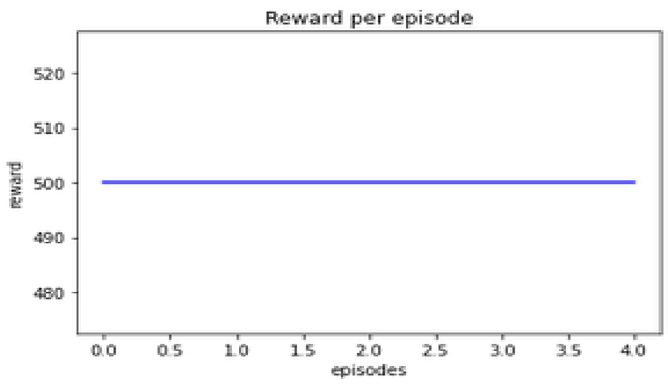


4. Provide the evaluation results. Run your agent on the three environments for at least 5 episodes, where the agent chooses only greedy actions from the learnt policy. Plot should include the total reward per episode

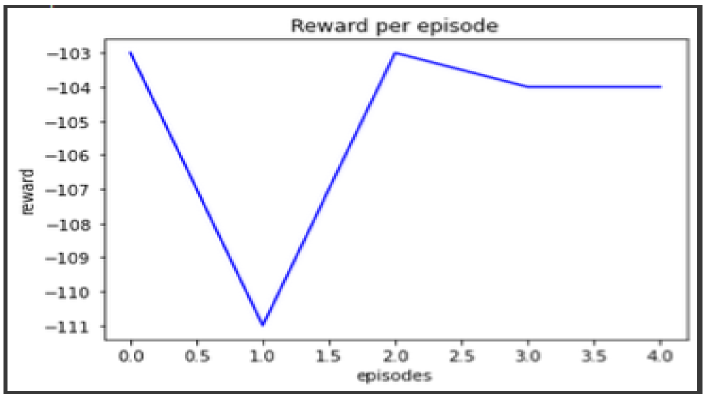
Grid Environment-



Cart pole-



Mountain Car-



5. Provide your interpretation of the results. E.g. how the DQN algorithm behaves ondifferent environments

The memory and computation required for the Q-value algorithm would be prohibitively expensive.In its place, a deep network Q-Learning function approximator is used.

Deep Q-Network is the name of this learning algorithm (DQN).

The key concept in this development was thus to represent the Q-network with deep neural networks and train this network to predict total reward.

Deep-Q Networks can be described as model-free, value-based, off-policy methods. The performance is stable when the DQN is implemented on the cartpole environment, we approximate a function by training the data, the loss function is very essential in the DQN network, it learns from its mistakes through back propagation.

- The state space is huge when it comes to the cart pole environment, small change in the angle or the velocity results in a new state, by using the DQN algorithm in this environment we would be able to train the data. The approximation is done in a state action pair, since there are two actions the function is defined by Q(s, a=left/right). Adam optimiser is used. Mean square error is used to find the loss function.

- DQN network used in the grid environment has 4 sets of actions, the state action pair is taken and the function is defined byQ(s, a=left,right,up,down). Since we have a large set of states, DQN could be used in an environment as such to train the agent to reach the goal state(3,3) in the grid environment. Function approximator is used in this environment to reach the goal state. Adam optimization is used. MSE is considered to calculate the loss function.

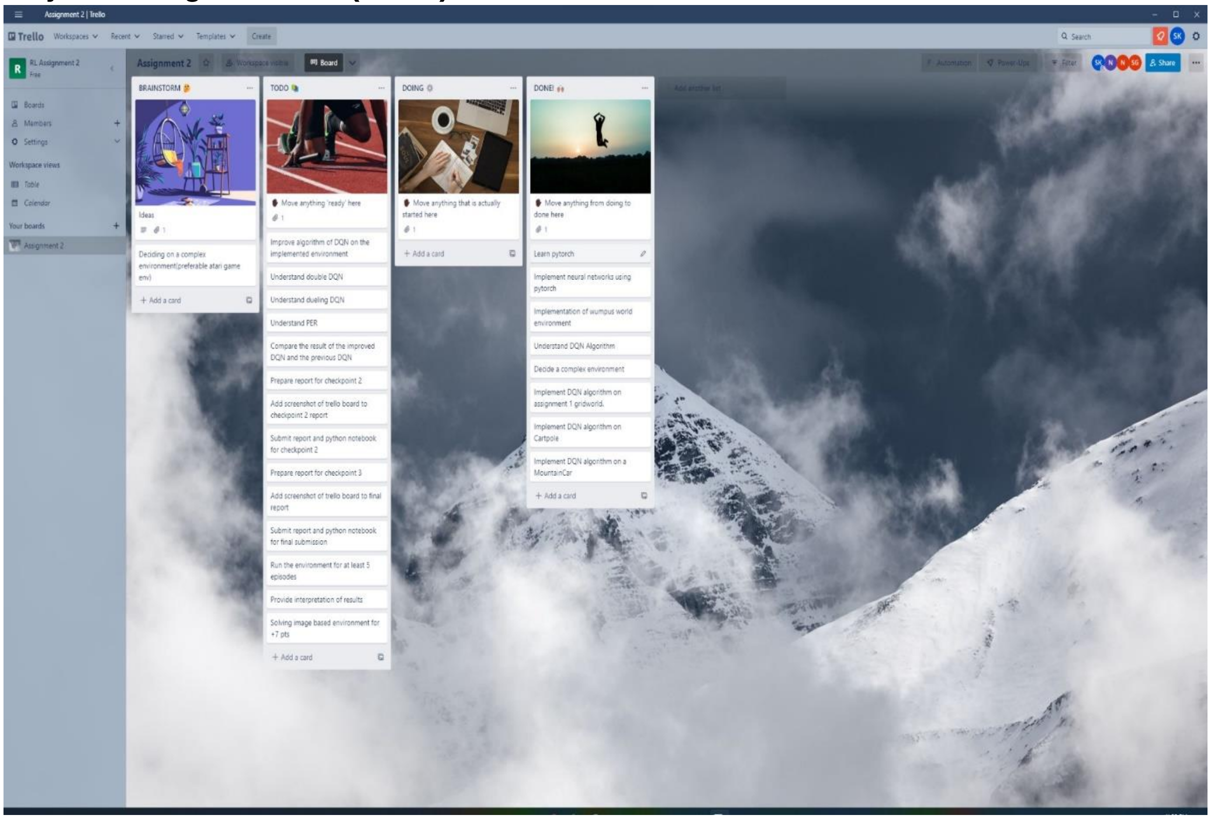
- DQN network used in the mountain car environment which has three set of actions and the function is defined by Q(s, a=0,1,2). The goal is to reach the red flag. We use a function optimiser which has weights in it and the loss function is calculated using backward propogation. Adam optimization is used. We use MSE for loss function calculation. Position of the reward is also considered while implementing the DQN. The reward function for this environment is more complex.

.

6. Include all the references that have been used to complete this part

- https://www.gymlibrary.dev/environments/classic_control/cart_pole/
- https://www.gymlibrary.dev/environments/classic_control/mountain_car/#mountain-car
- https://piazza.com/buffalo/fall2022/cse4546/resources

Project management tool(Trello)



Assignment 2

BRAINSTORM
- Ideas
- Deciding on a complex environment(preferable atari game env)

TODO
- Move anything 'ready' here
- Improve algorithm of DQN on the implemented environment
- Understand double DQN
- Understand dueling DQN
- Understand PER
- Compare the result of the improved DQN and the previous DQN
- Prepare report for checkpoint 2
- Add screenshot of trello board to checkpoint 2 report
- Submit report and python notebook for checkpoint 2
- Prepare report for checkpoint 3
- Add screenshot of trello board to final report
- Submit report and python notebook for final submission
- Run the environment for at least 5 episodes
- Provide interpretation of results
- Solving image based environment for +7 pts

DOING
- Move anything that is actually started here

DONE!
- Move anything from doing to done here
- Learn pytorch
- Implement neural networks using pytorch
- Implementation of wumpus world environment
- Understand DQN Algorithm
- Decide a complex environment
- Implement DQN algorithm on assignment 1 gridworld.
- Implement DQN algorithm on Cartpole
- Implement DQN algorithm on a MountainCar

CHECKPOINT 3

1. Discuss the algorithm you implemented

   The implemented algorithm is double DQN, two identical neural network models are used in double DQN. Like DQN, one learns from experience playback, while the other is a replica of the final episode of the first model. In DQN, the reward is added to the subsequent state's maximum Q-value when calculating Q-value. The value generated from the output of the neural network for a given state will obviously increase with time if the Q-value calculates a high number for that state repeatedly. It is challenging to teach the neural network to recognize that action b is preferable in some circumstances because it is trained to give a considerably higher value for action a when given state s.

   $$R_t + \gamma Q^c(s_{t+1}.\operatorname{argmax} Q(s_{t+1}.a'))$$

   The main goal of Double Q-learning is to decrease overestimations of Q-values by separating action selection from action evaluation, allowing for the deployment of a distinct Q-network at each stage. The agent can utilize a behaviour policy to explore the environment at random, and the data can be used to train a target policy with a high return. The behaviour and target policies must be the same if we employ an on-policy method, though. The data that the algorithm learned from came from the identical policy that we were trying to learn.

2. What is the main improvement over the vanilla DQN?

   We want to separate the estimator of these two elements with DDQN by using two new streams:

   - one that calculates the state's worth
   - V(s) one that calculates the benefit of each action A(s,a)

   Choosing the best action based on the highest q value (which is noisy) can result in false positives.

   If non-optimal actions are consistently assigned a higher Q value than the best optimal action, learning will become complicated. We use two networks to decouple action selection from target Q value generation when computing the Q target.

   We:

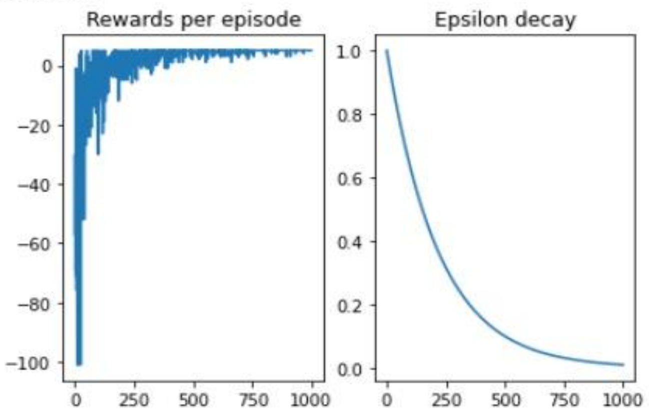
   - Use our DQN network to determine the best course of action for the next state (the action with the highest Q value).
   - Calculate the target Q value of taking that action at the next state using our target network.

   Double DQN helps us reduce overestimation of q values, allowing us to train faster and have more stable learning.

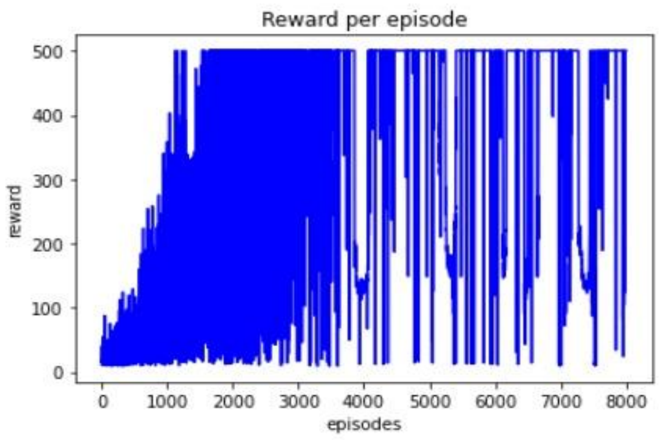
3. Show and discuss your results after applying you're the two algorithms implementation on environment. Plots should include epsilon decay and the total reward per episode.

Grid Environment-

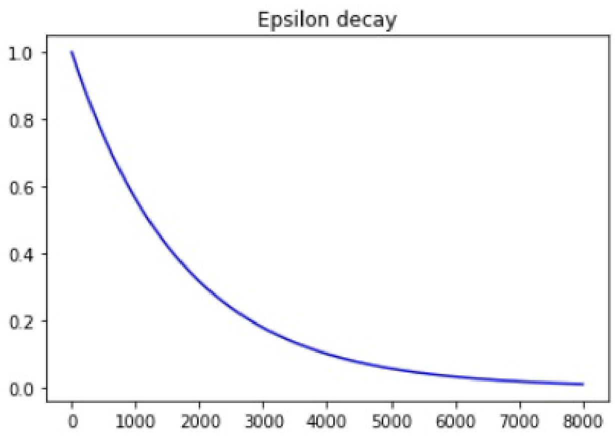
```
Complete
```



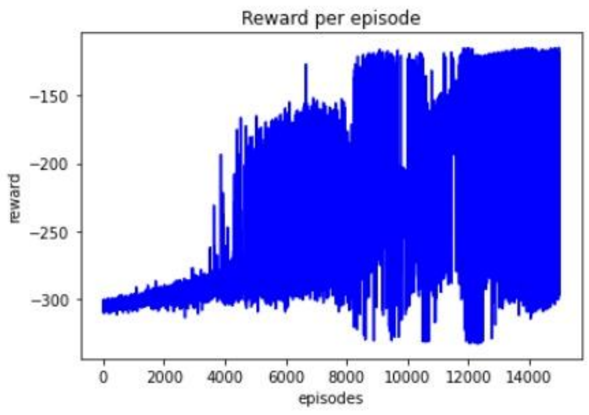
Cartpole-

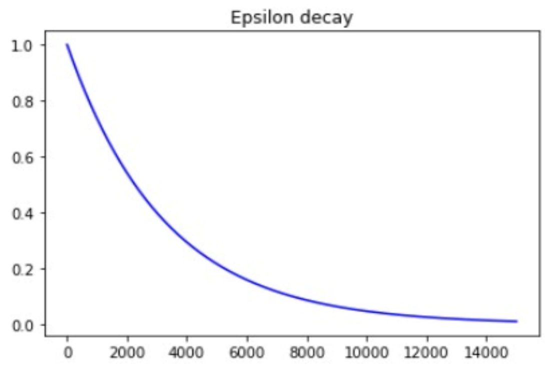


```
Complete
```



Mountain Car-

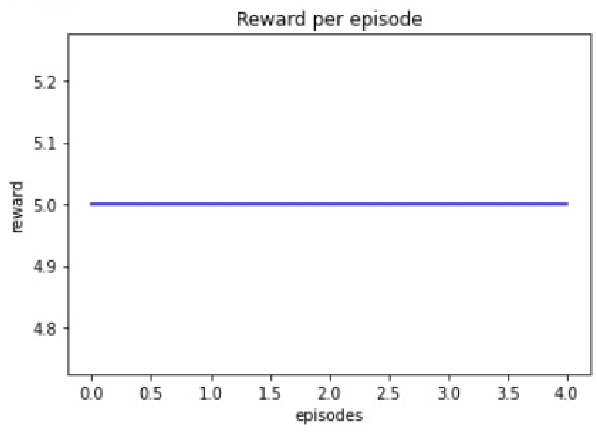




4. Provide the evaluation results. Run your environment for at least 5 episodes, where the agent chooses only greedy actions from the learnt policy. Plot should include the total reward per episode.

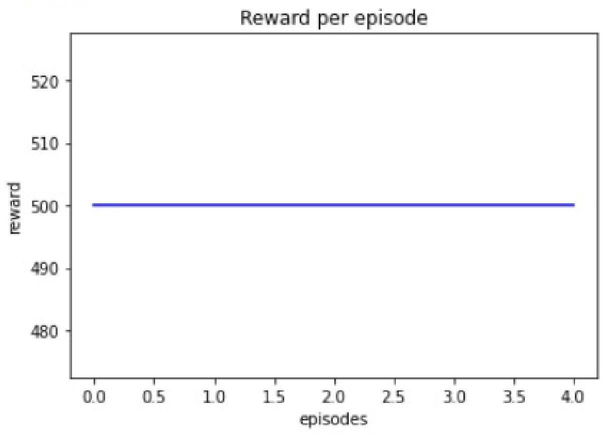
Grid Environment-

```
Complete
```



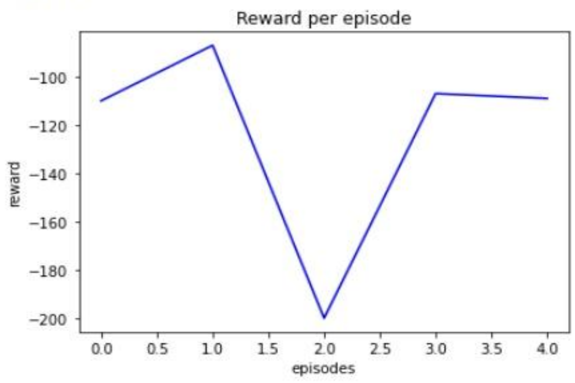
Cartpole-

```
Complete
```



Mountain Car-

```
Complete
```

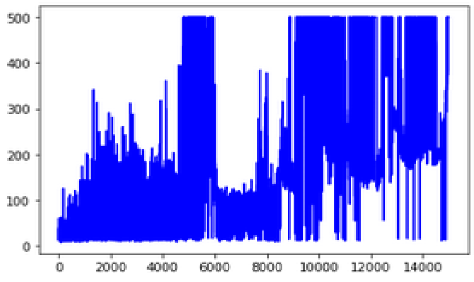


5. Compare the performance of both algorithms (DQN & Improved version of DQN) on the same environments and provide your interpretation of the results. Overall, three rewards dynamics plots with results from two algorithms applied on:

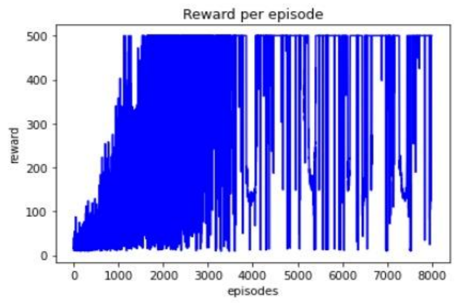
- • Grid-world environment
- • 'CartPole-v1'
- • OpenAI Gym or Multi-agent environment

Cartpole

DQN-

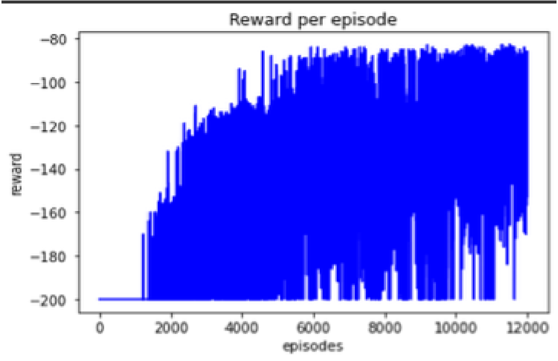


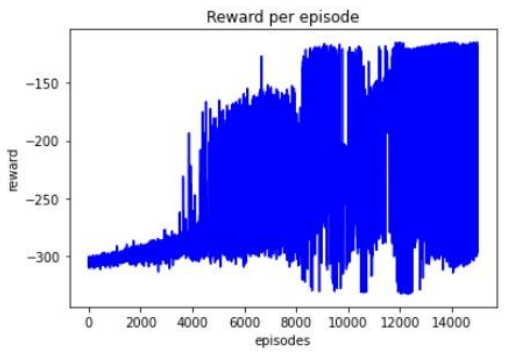
```
Complete
```



```
Complete
```

As we can see that in the DDQN we have a smaller number of oscillations and the convergence is quick when compared to the DQN network.
Mountain car-
DQN-





We increase the replay memory so that we have less number of oscillations, we increase the target update from 4 to 100 because it indirectly changes the loss less frequently.

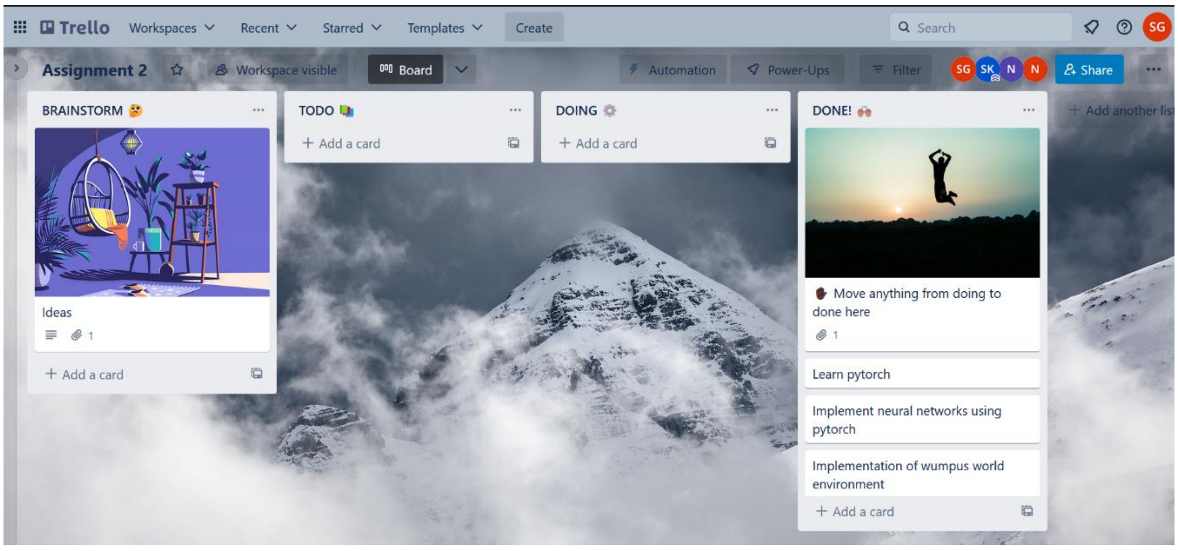
6. Provide your interpretation of the results. E.g., how the same algorithm behaves on different environments, or how various algorithms behave on the same environment.

   DDQN on a complex environment like cartpole converged with less number of oscillation, whereas, DQN had a lot of oscillations.

   For a simple environment like the grid environment, DQN and DDQN converge quickly when compared to the other environments.

7. https://www.gymlibrary.dev/environments/classic_control/cart_pole/
   https://www.gymlibrary.dev/environments/classic_control/mountain_car/#mountain-car
   https://piazza.com/buffalo/fall2022/cse4546/resources

Project management tool update-



BRAINSTORM

Ideas

+ Add a card

TODO

+ Add a card

DOING

+ Add a card

DONE!

Move anything from doing to done here

Learn pytorch

Implement neural networks using pytorch

Implementation of wumpus world environment

+ Add a card

UBIT name	Assignment part	Contribution
sannihit	Assignment 2 checkpoint 1	50%
sannihit	Assignment 2 checkpoint 2	50%
sannihit	Assignment 2 checkpoint 3	50%
srisanni	Assignment 2 checkpoint 1	50%
srisanni	Assignment 2 checkpoint 2	50%
srisanni	Assignment 2 checkpoint 3	50%