Assignment 1

Eyal Mazuz 208373977
Ariel Amsel 302269907

How to run:

For section 1 code, just run q_learning.py, it'll run the environment and save the graphs in the current folder

For section 2 code everything is in dqn.py, if you want to alternate between the 3 and 5 layer network all you need to do is change the NUM_LAYERS in line 23 to your choosing.

For section 3 code everything is in duelDQN.py, if you want to alternate between the 3 and 5 layer network all you need to do is change the NUM_LAYERS in line 23 to your choosing.

Section 1-

Question 1-

Methods such as Value-Iterations cannot be implemented in such environments because it's close to impossible to compute the transition probability matrix to state s' given state s and action a, those methods rely on the probability when computing the value function.

Question 2-

Model-free methods resolve the problem by sampling trajectories from the environment and using those trajectories to estimate the Q or value function.

Question 3-

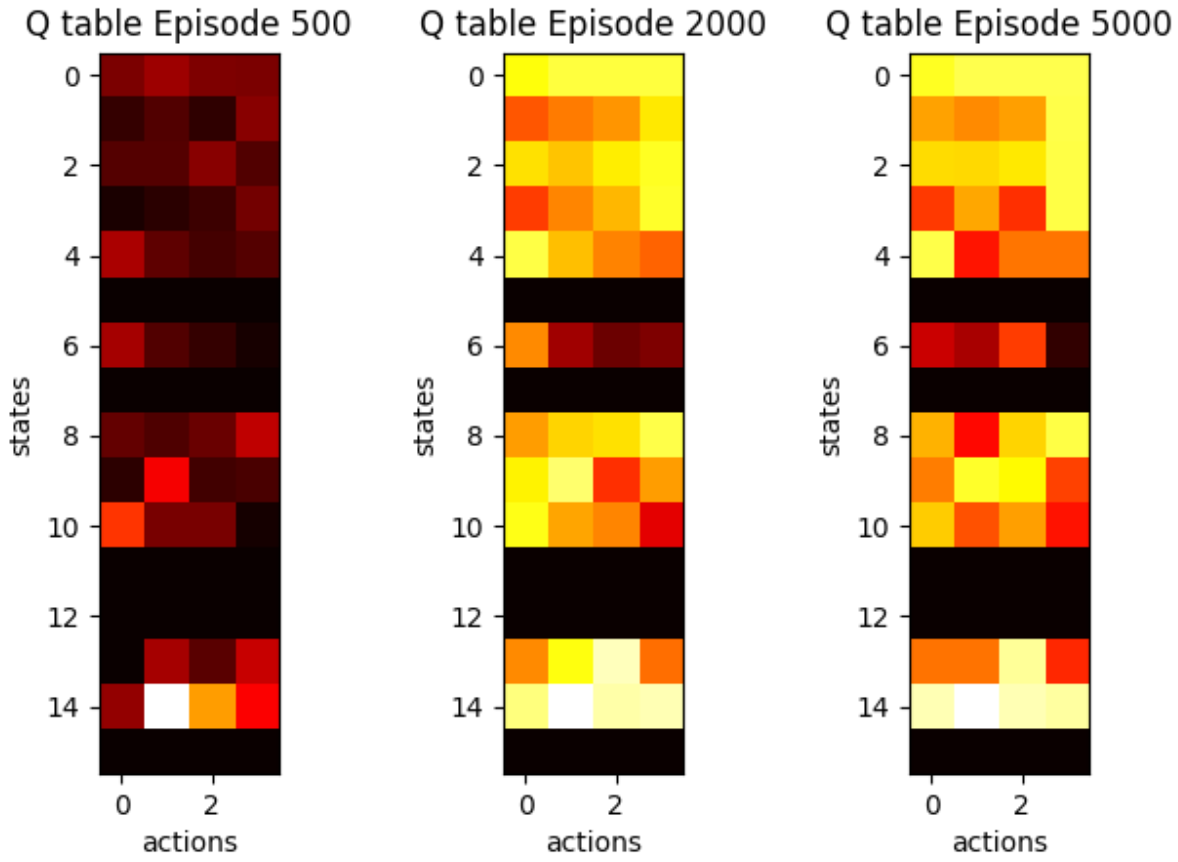
The main difference between SARSA and Q-learning is that SARSA is an on-policy method which means it learns from its own policy and the Q-learning is an off-policy method that learns from a behavior policy which is different from a greedy used to create the trajectories.

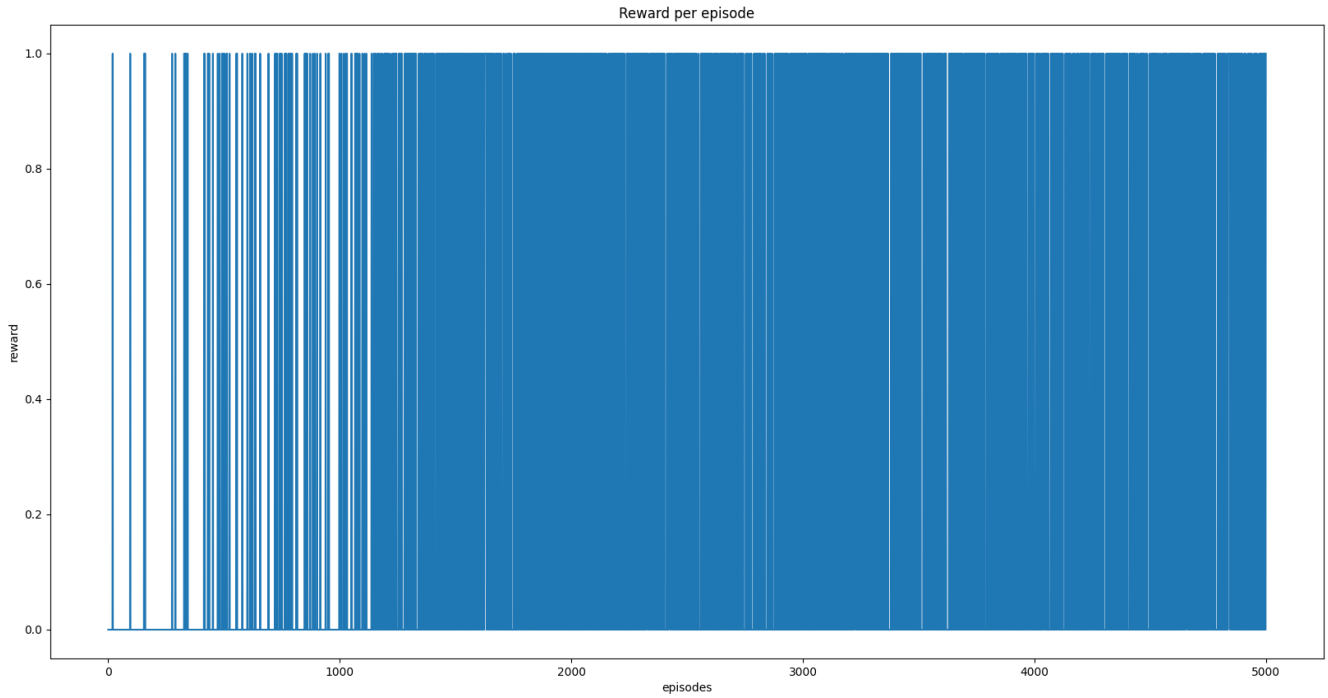
Question 4-

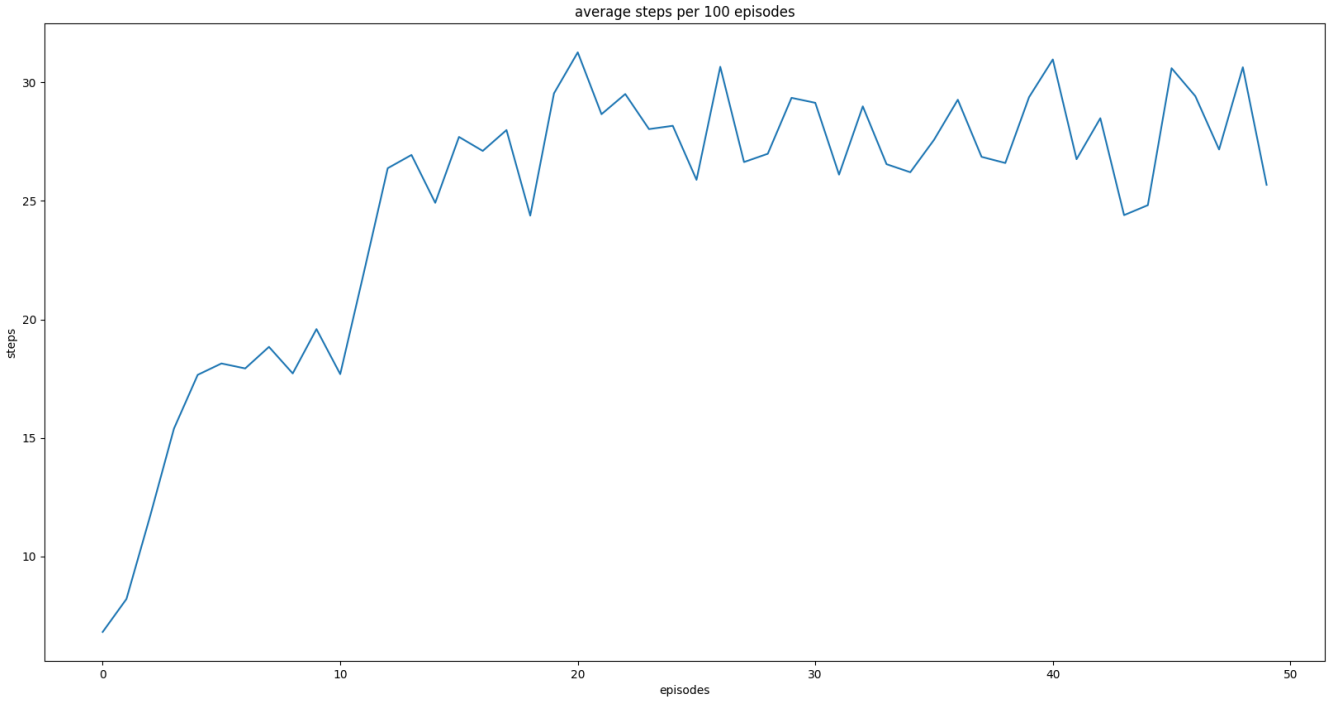
When acting greedily you don't try new actions that might give you better rewards in the long run. In ε-greedy, you get to select new actions and explore new states that yield higher rewards.

Q learning Script-

LEARNING_RATE = 0.1
DISCOUNT_FACTOR = 1
EPSILON_START = 0.9
EPSILON_END = 0.1
DECAY_RATE = 200




Reward per episode
reward
episodes
0.0
0.2
0.4
0.6
0.8
1.0
0
1000
2000
3000
4000
5000



Section 2-

Question 1-

The random sampling process de-correlates the trajectories which allow for more stable gradient updates.

Question 2-

This solves the issue of a non-stationary target, when we the updates we also update the values of the targets this makes the learning unstable cause the network always tries to reach a different target every time.

DQN Script-

Episodes till convergence:

Episodes 4500~

Hyper Parameters:

LEARNING_RATE = 0.001
DISCOUNT_FACTOR = 1
EPSILON_START = 0.9

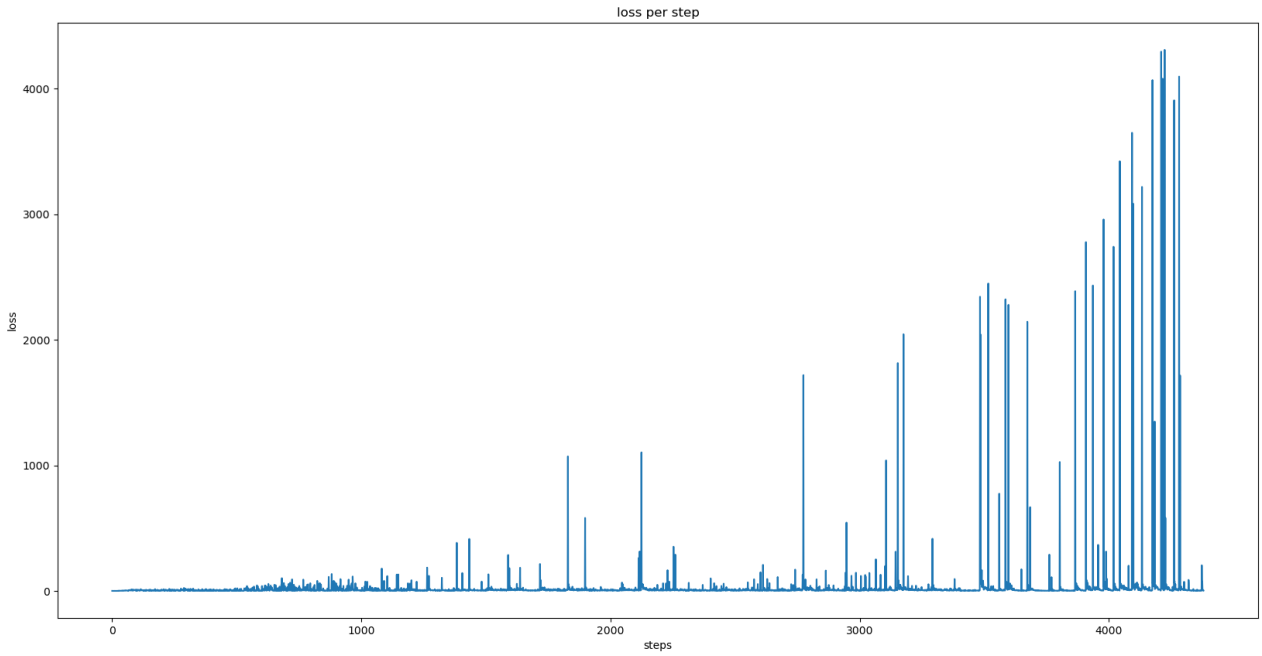
EPSILON_END = 0.1
DECAY_RATE = 200
UPDATE_EVERY = 10

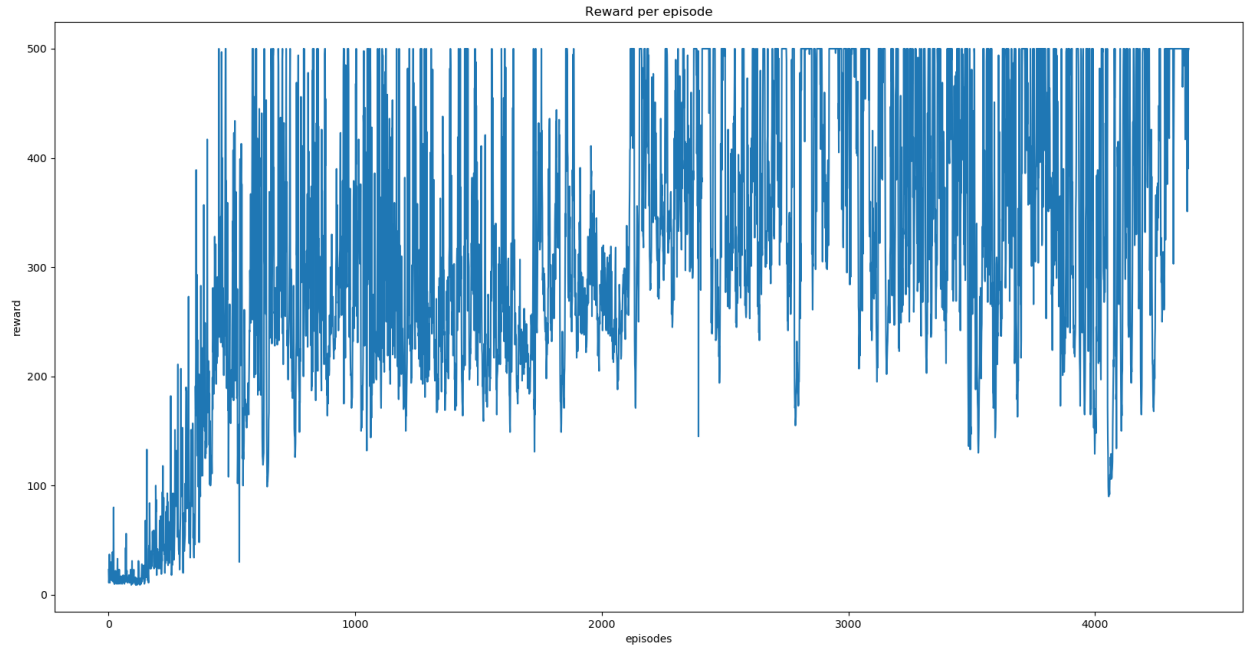
Model Summary:

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 512)	2560
dense_1 (Dense)	(None, 256)	131328
dense_2 (Dense)	(None, 64)	16448
dense_3 (Dense)	(None, 2)	130

The hyperparameters that affected the model the most are the decay rate and the discount factor.
Increasing/Decreasing the decay rate affected the exploration rate by causing the model to explore less or explore more respectively, increasing the convergence time. Less exploration caused the model to get fixed on less than optimal policy and took him time to find a better policy, and more exploration caused the model not to choose the better action when he could've and it took the model more time to converge.

Decreasing the discount caused to model to focus more on the current state of the pole which made the model select actions that he thought would help him now gain more reward but in the long run caused the pole to fall to the other side.





Section 3-

The Improvement:

We implemented the Dueling DQN improvement - The network has 2 outputs: Value-function (single output) and Action value function (2 outputs in our case).
The final decision on which action to take is decided using the following function:

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + (A(s,a;\theta,\alpha) - \frac{1}{|\mathcal{A}|}\sum_{a'} A(s,a;\theta,\alpha)$$

Episodes till convergence:

Episodes 3500~

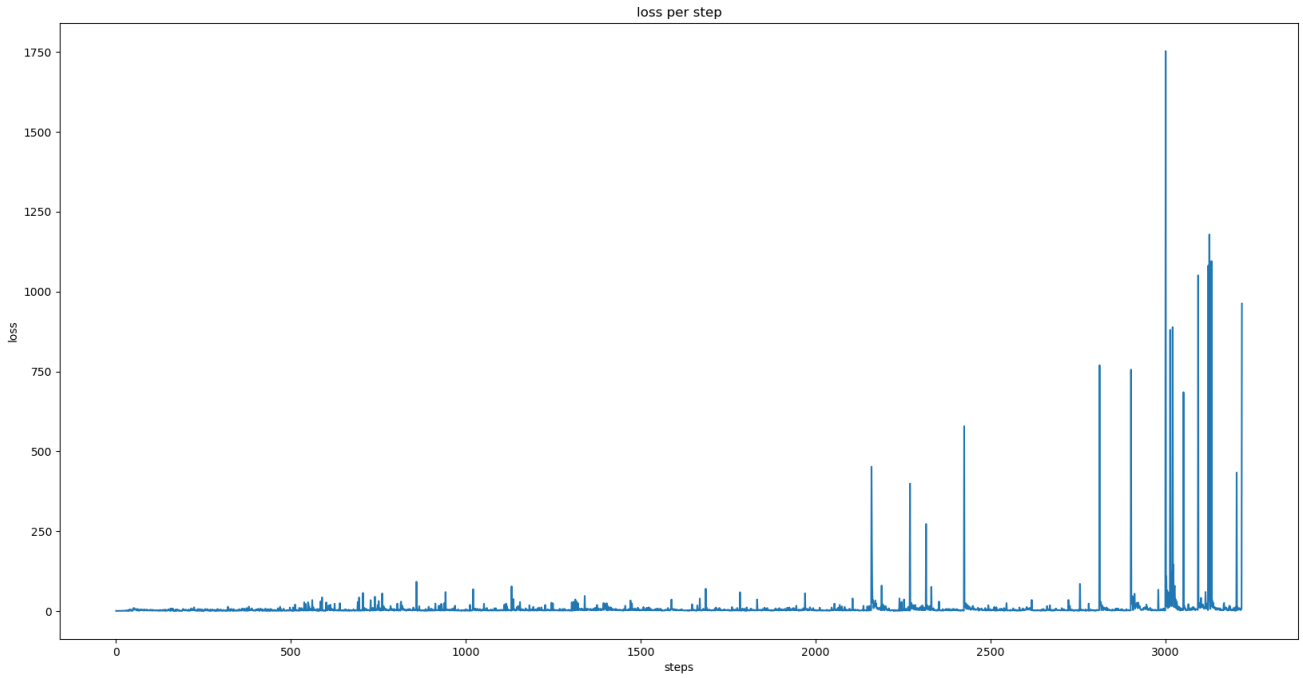
Hyper Parameters:

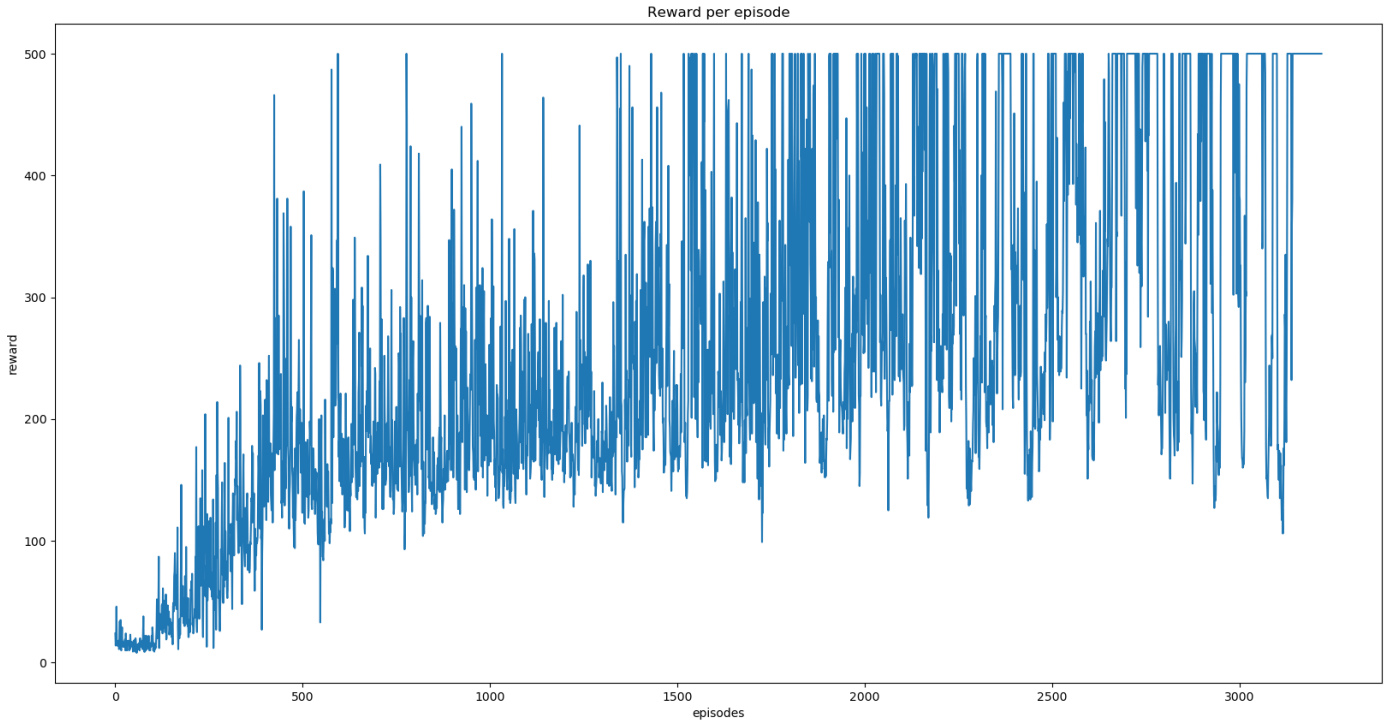
LEARNING_RATE = 0.001
DISCOUNT_FACTOR = 1
EPSILON_START = 0.9
EPSILON_END = 0.1
DECAY_RATE = 200
UPDATE_EVERY = 10

Model Summary:

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 512)	2560
dense_1 (Dense)	(None, 256)	131328
dense_2 (Dense)	(None, 64)	16448
Q_values (Dense)	(None, 2)	130
state_value (Dense)	(None, 1)	65

Q_values and state_values are both connected to dense_2





Results Comparison:

We can see that the Dueling-DQN converges faster than the simple DQN, it requires ~1,000 steps less than the DQN (Improvement of ~23%).

We can also see that the Dueling-DQN loss is significantly smaller than the DQN loss. The maximum loss received by Dueling-DQN is ~2000 while the DQN maximum loss gets to ~4000. Also the peaks in the loss are more frequent and bigger.