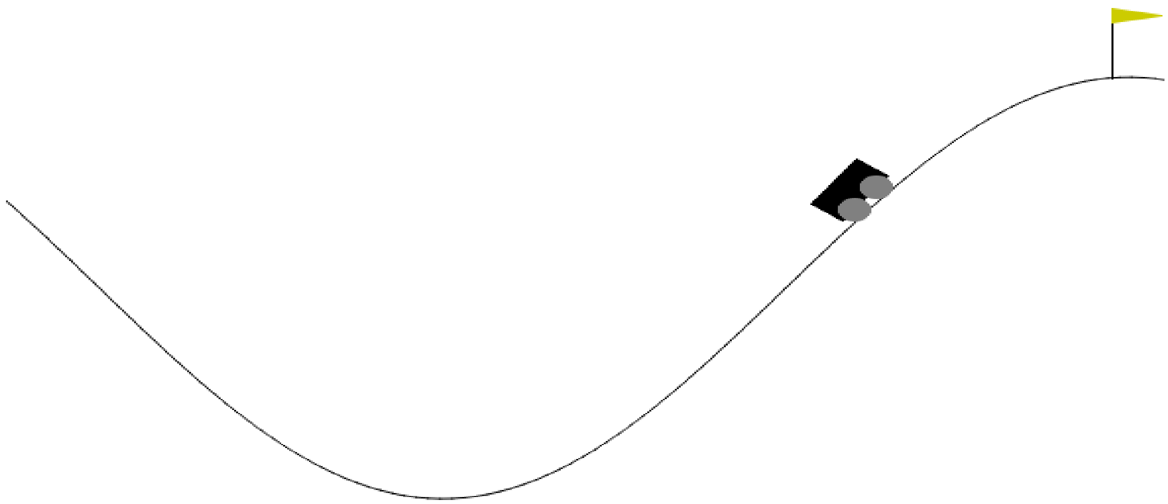
Mountain Car



Problem Formulation

A car is on a one-dimensional track, positioned between two “mountains”. The goal is to drive up the mountain on the right in as few steps as possible; however, the car’s engine is not strong enough to scale the mountain in a single pass. Therefore, the only way to succeed is to drive back and forth to build up momentum.

The car’s state, at any point in time, is given by a vector containing its horizontal position and velocity. The car commences each episode stationary, at the bottom of the valley between the hills (at position approximately -0.5), and the episode ends when either the car reaches the flag (position > 0.5) or after 200 moves.

At each move, the car has three actions available to it: push left, push right or do nothing, and a penalty of 1 unit is applied for each move taken (including doing nothing). This means that, unless it can figure out a way to ascend the mountain in less than 200 moves, it will always achieve a total “reward” of -200 units.

Observation Space

The observation is ndarray with shape (2,) where the elements correspond to the following:

Num	Observation	Min	Max
0	position of the car along the x-axis	-Inf	Inf
1	velocity of the car	-Inf	Inf

Action Space

There are 3 discrete deterministic actions:

Num	Observation	Value	Unit
0	Accelerate to the left	Inf	position (m)
1	Don’t accelerate	Inf	position (m)
0	Accelerate to the right	Inf	position (m)

Reward:

The goal is to reach the flag placed on top of the right hill as quickly as possible, as such the agent is penalized with a reward of -1 for each timestep it isn’t at the goal and is not penalized (reward = 0) for when it reaches the goal.

Starting State

The position of the car is assigned a uniform random value in [-0.6, -0.4]. The starting velocity of the car is always assigned to 0.

Episode Termination

The episode terminates if either of the following happens:

- The position of the car is greater than or equal to 0.5 (the goal position on top of the right hill)
- The length of the episode is 200.

Techniques used to solve mountain car

Q-learning is a model free reinforcement learning technique that can be used to find the optimal action selection policy using Q function without requiring a model of the environment. Q-learning eventually finds an "optimal policy".

Training the agent with different hyper parameters:-

We trained our agent using different values for alpha (learning rate) and gamma (discount factor). As for epsilon we keep it 1 as an initial value then it will decrease according to the epsilon decay value in each episode.

The average results for each pair of alpha and gamma are as follow:

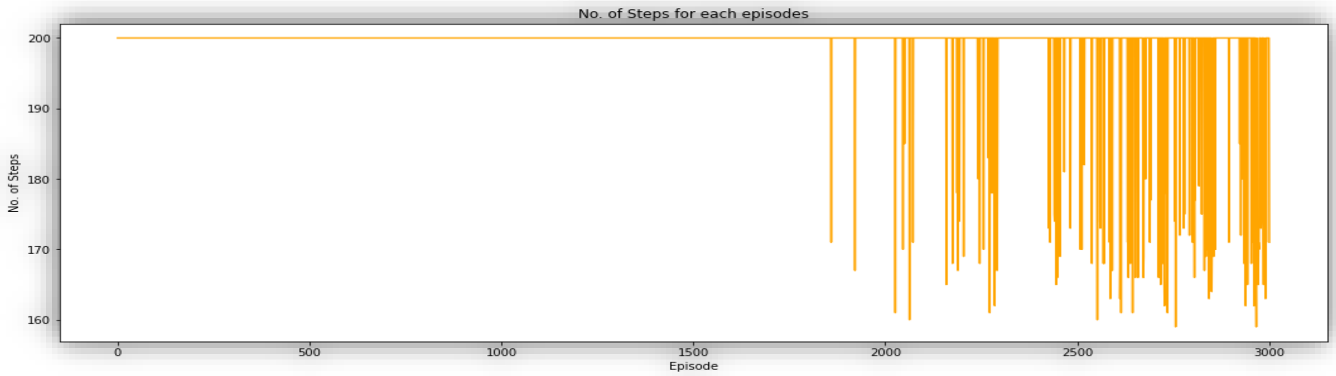
No. of pair	Avg TimeSteps for each pair of Alpha & Gamma	Avg Total Rewards for each pair of Alpha & Gamma	hyperparameters
0	198.761667	-198.761667	alpha=0.1, gamma=0.1
1	198.863000	-198.863000	alpha=0.1, gamma=0.6
2	197.608667	-197.608667	alpha=0.1, gamma=0.9
3	199.117000	-199.117000	alpha=0.5, gamma=0.1
4	196.656667	-196.656667	alpha=0.5, gamma=0.6
5	195.538333	-195.538333	alpha=0.5, gamma=0.9
6	199.974333	-199.974333	alpha=0.8, gamma=0.1
7	198.229333	-198.229333	alpha=0.8, gamma=0.6
8	198.866667	-198.866667	alpha=0.8, gamma=0.9

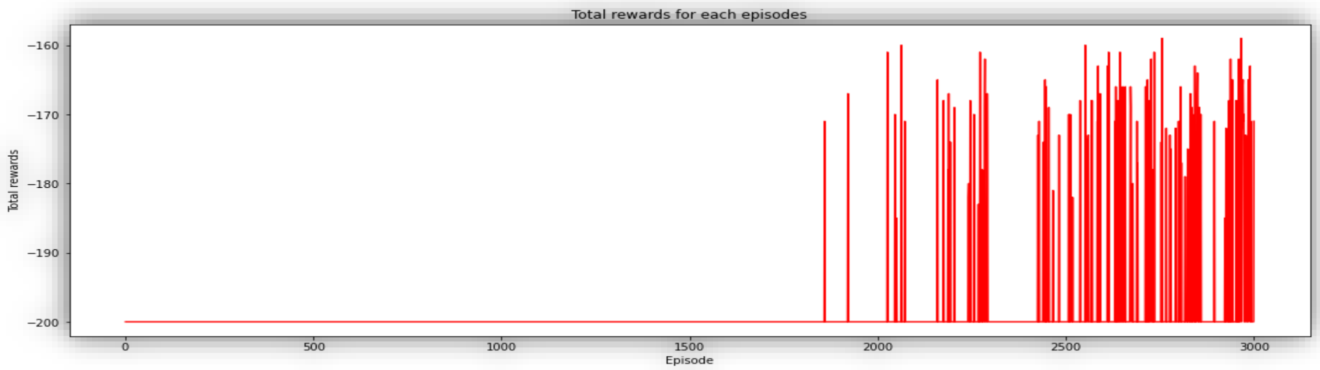
From our table, we observe that the pair that has alpha= .5 & gamma= .9 is the best one of performance over the others. So, we will retrain & test it on bunch of episodes to see its performance while test case.

Plotting results

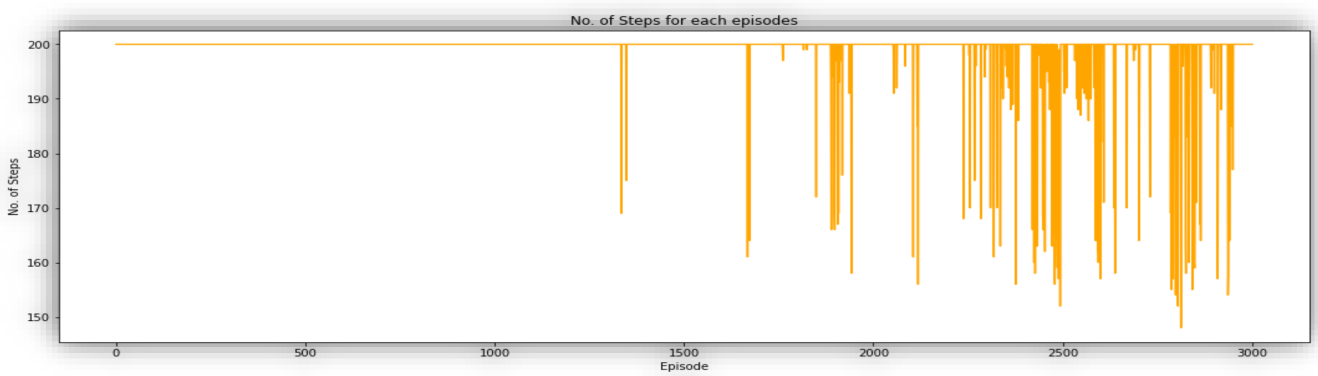
These are the results of total time Steps & total rewards for each episode (3000 episodes) in each pair of alpha & gamma (9 pairs) obtained from training phase.

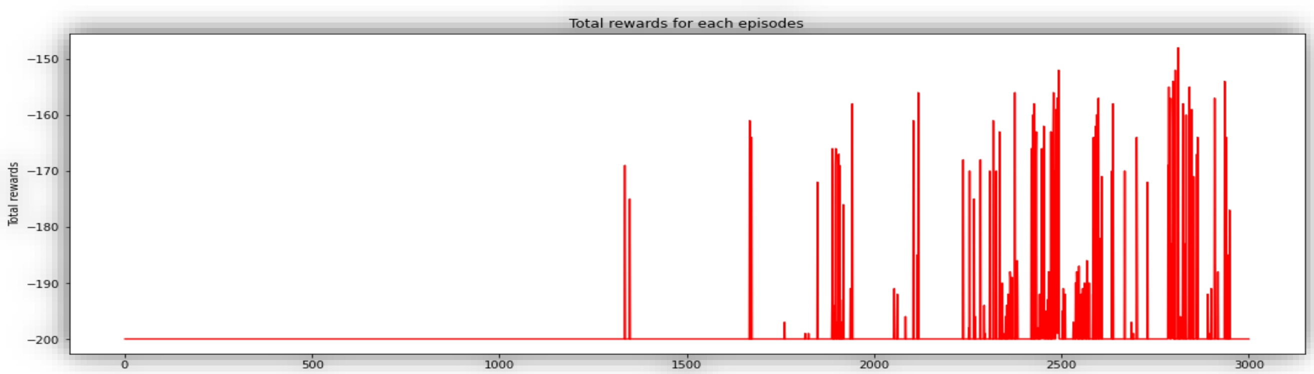
When alpha=0.1, gamma=0.1



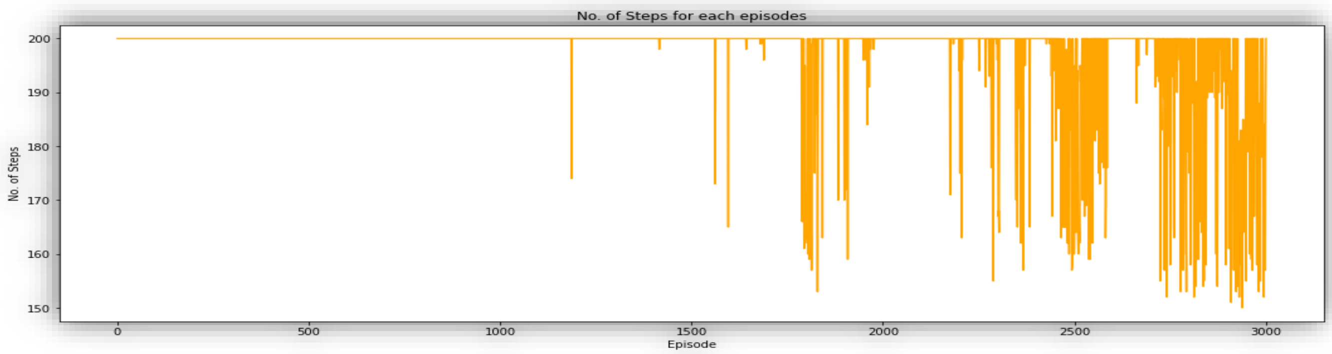


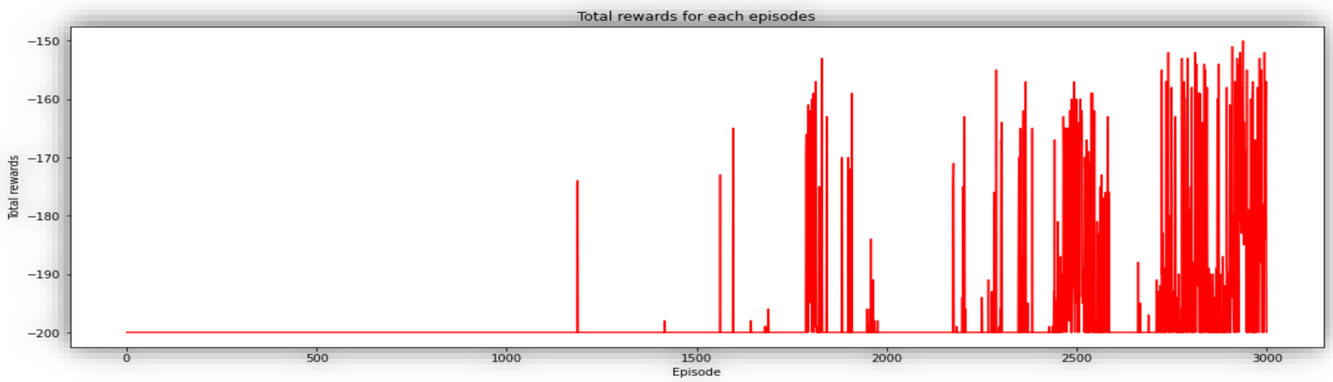
When alpha=0.1, gamma=0.6



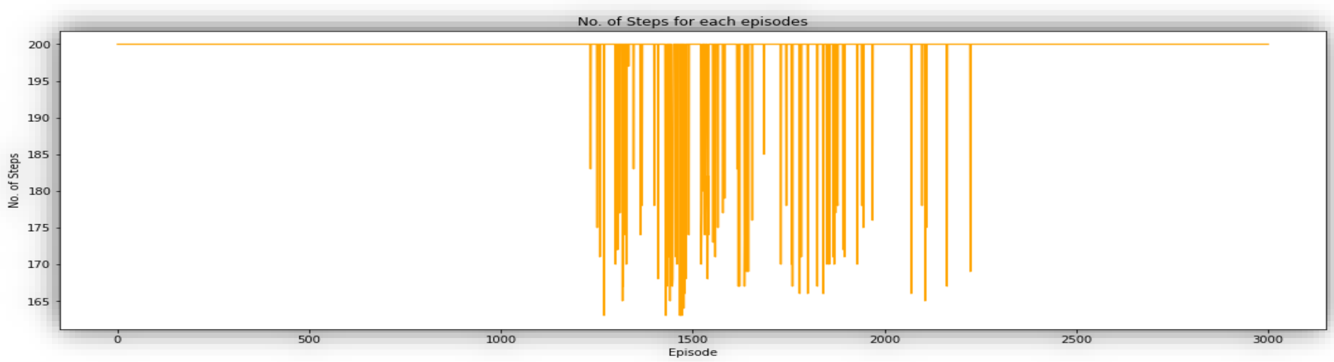


When alpha=0.1, gamma=0.9

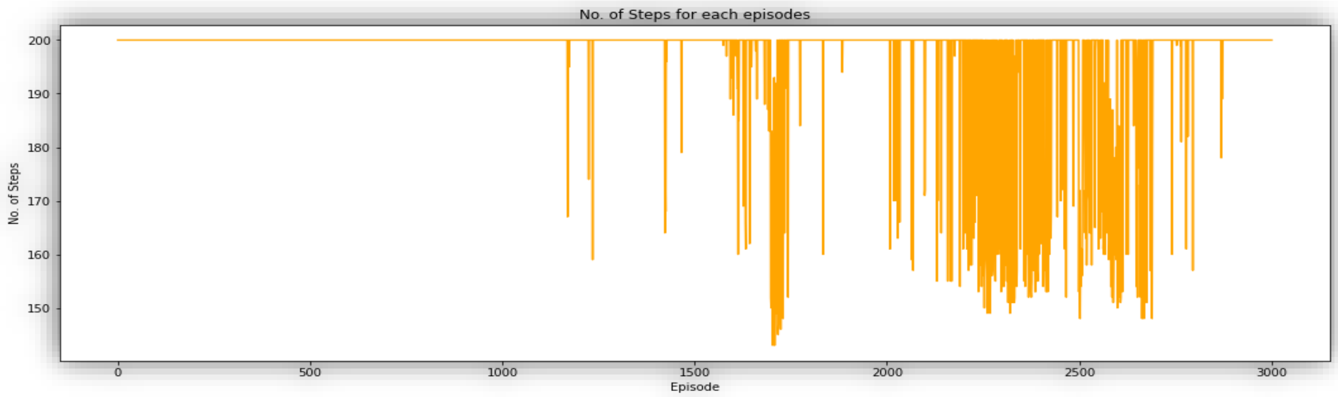


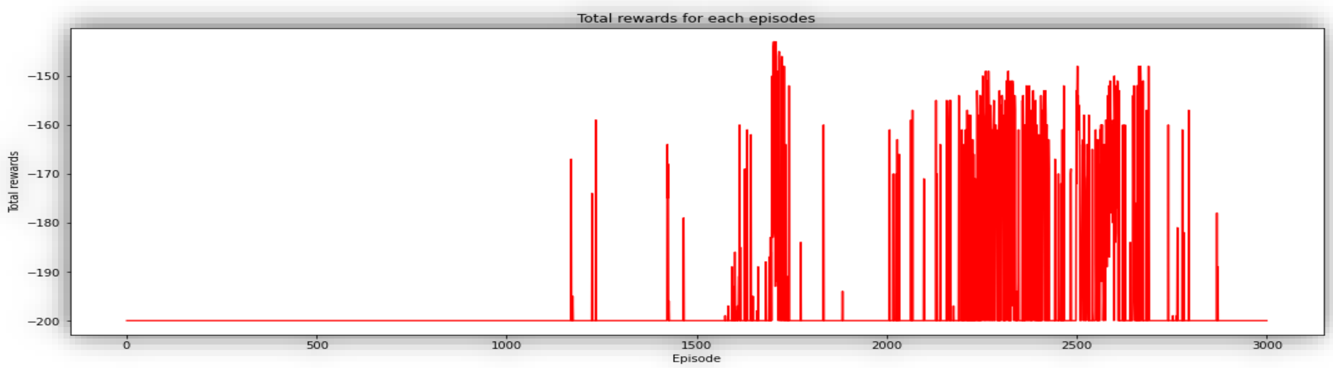


When alpha=0.5, gamma=0.1

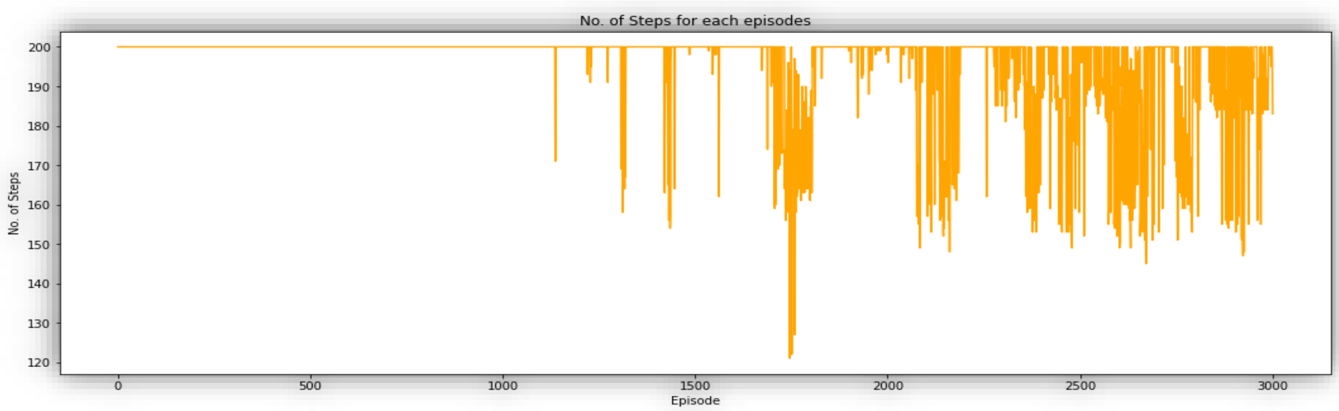


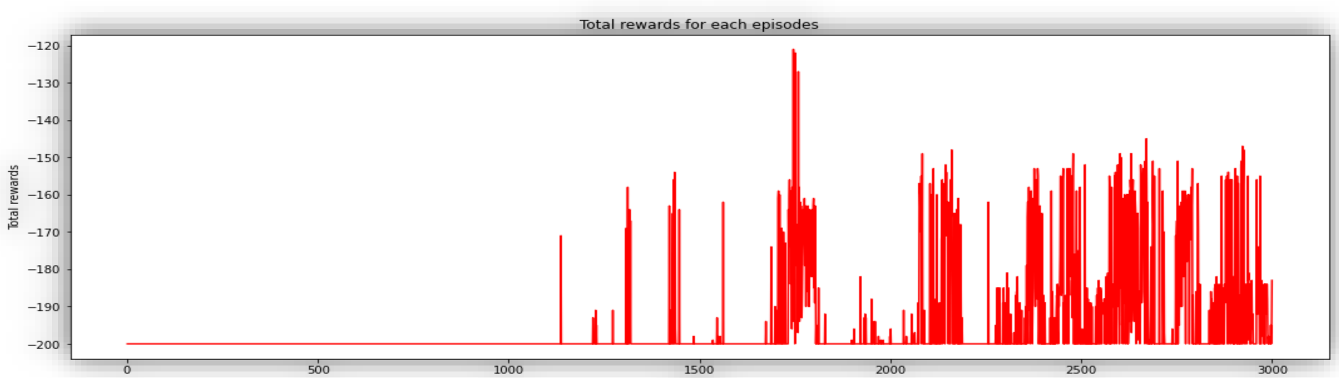
When alpha=0.5, gamma=0.6



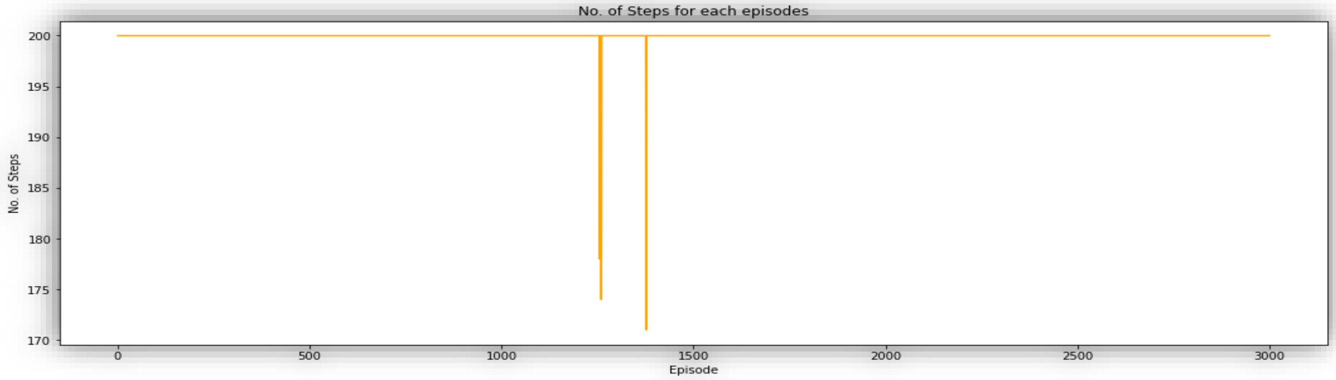


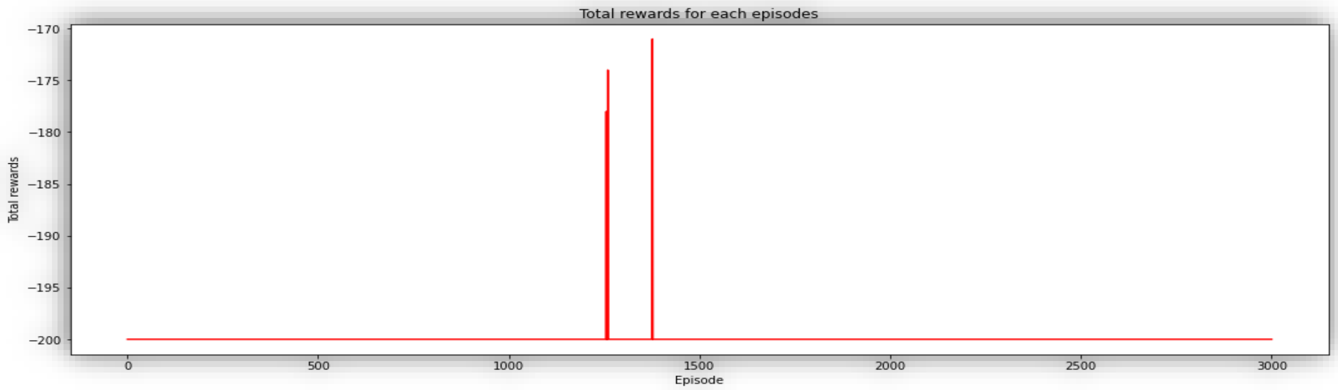
When alpha=0.5, gamma=0.9



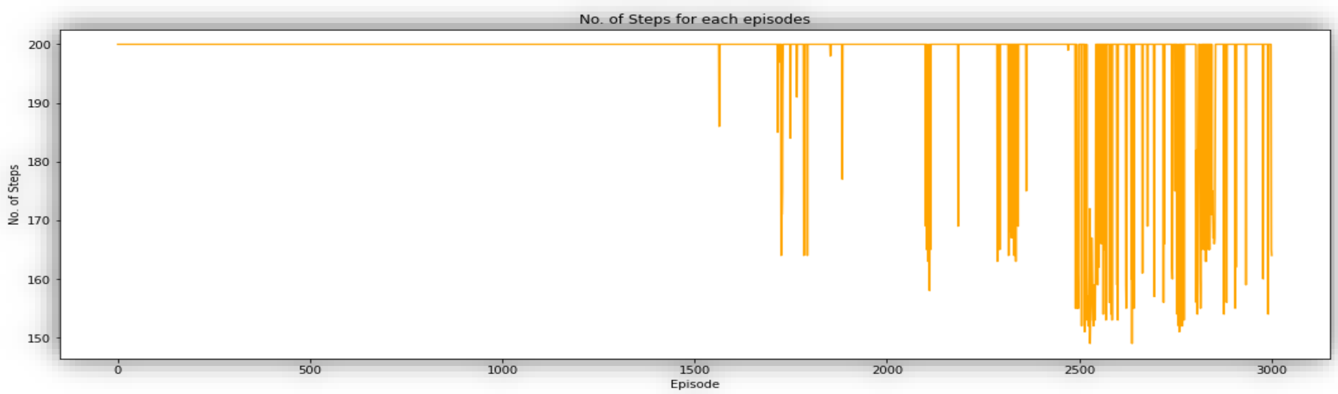


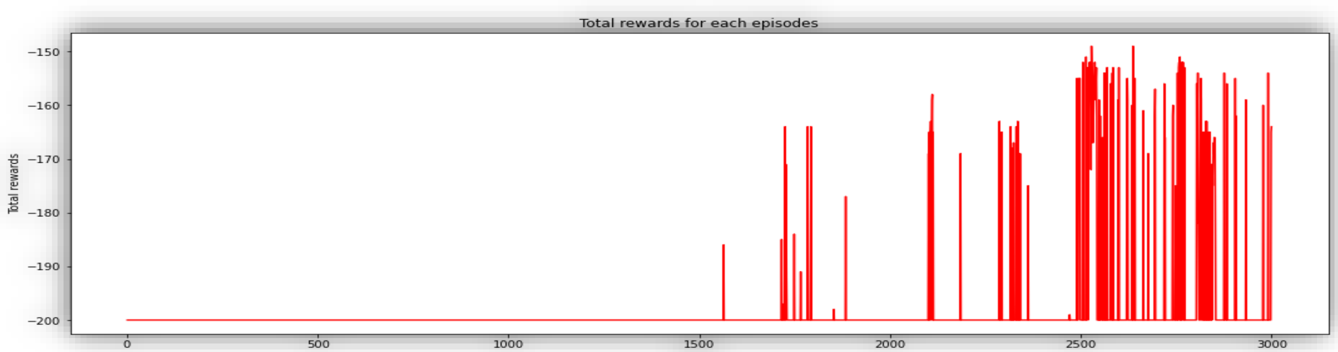
When alpha=0.8, gamma=0.1



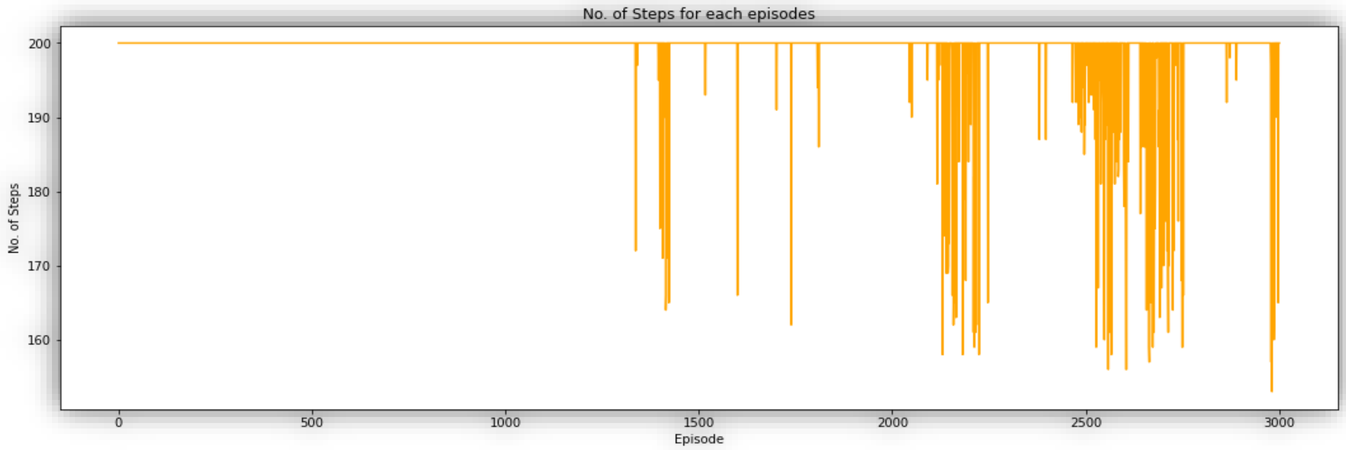


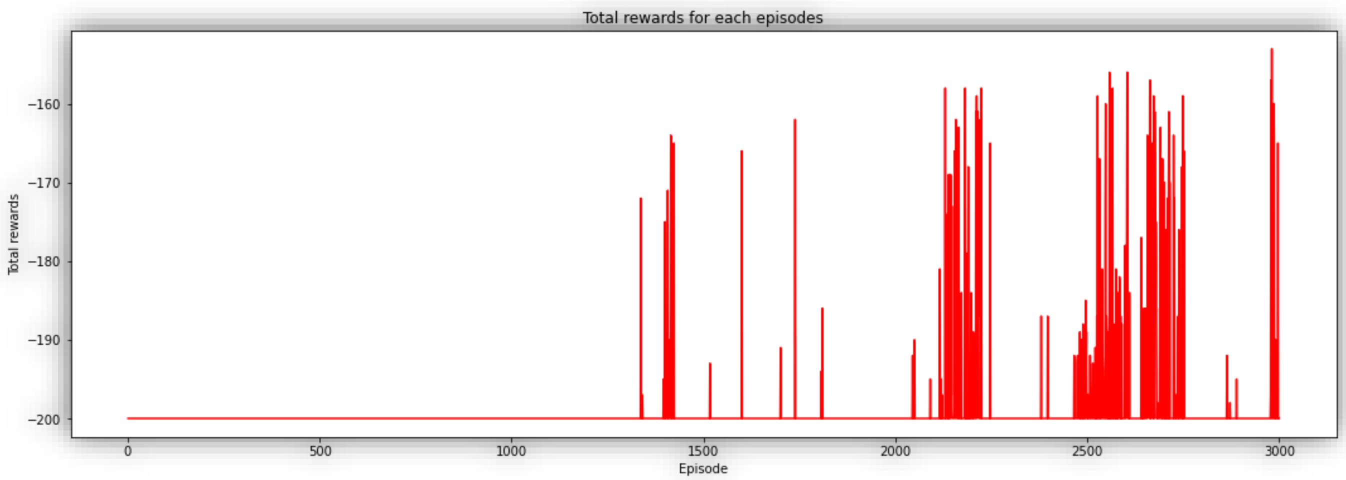
When alpha=0.8, gamma=0.6





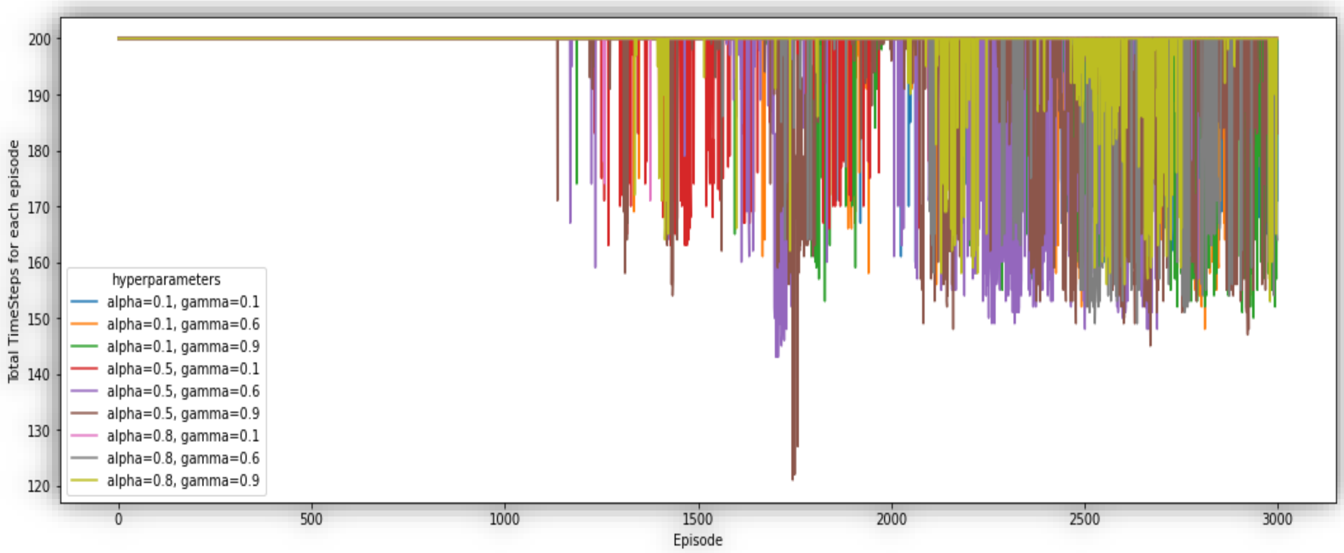
When alpha=0.8, gamma=0.9



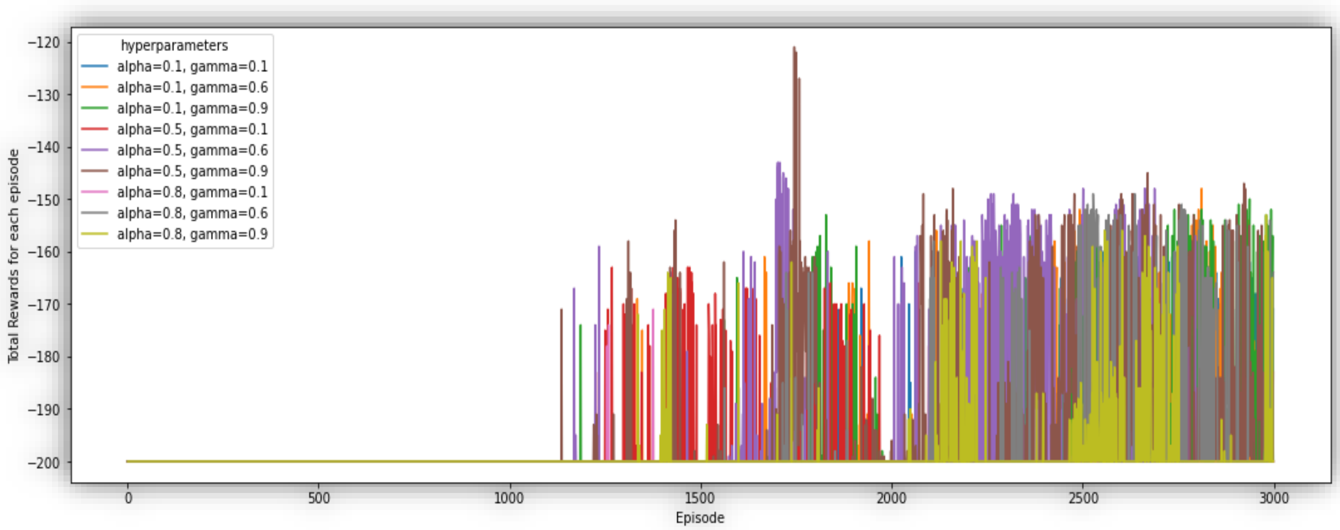


Plotting the Total results

Aggregated results of time Steps for each episode in each pair of alpha & gamma in one figure even we can see the performance variation between each pair.

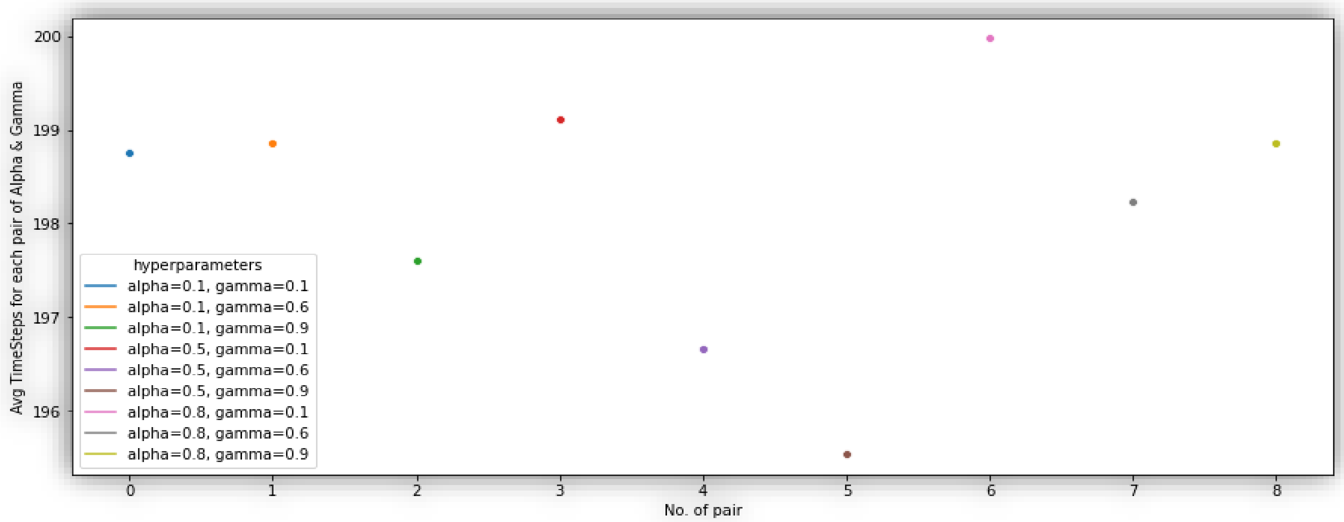


Aggregated results of total rewards for each episode in each pair of alpha & gamma in one figure even we can see the performance variation between each pair.

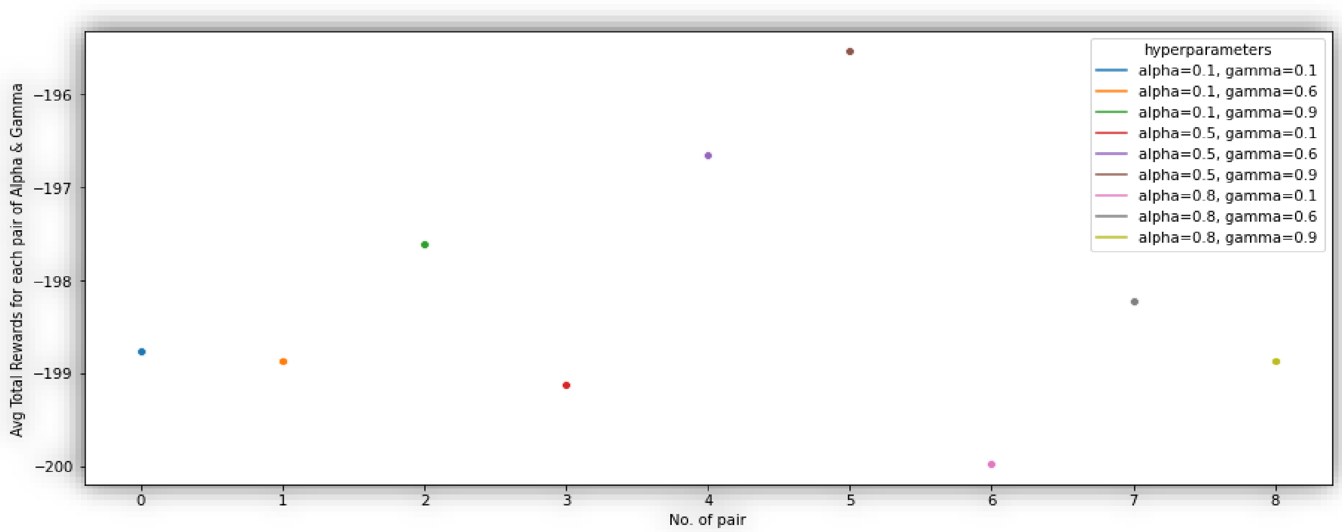


Plotting the Average results

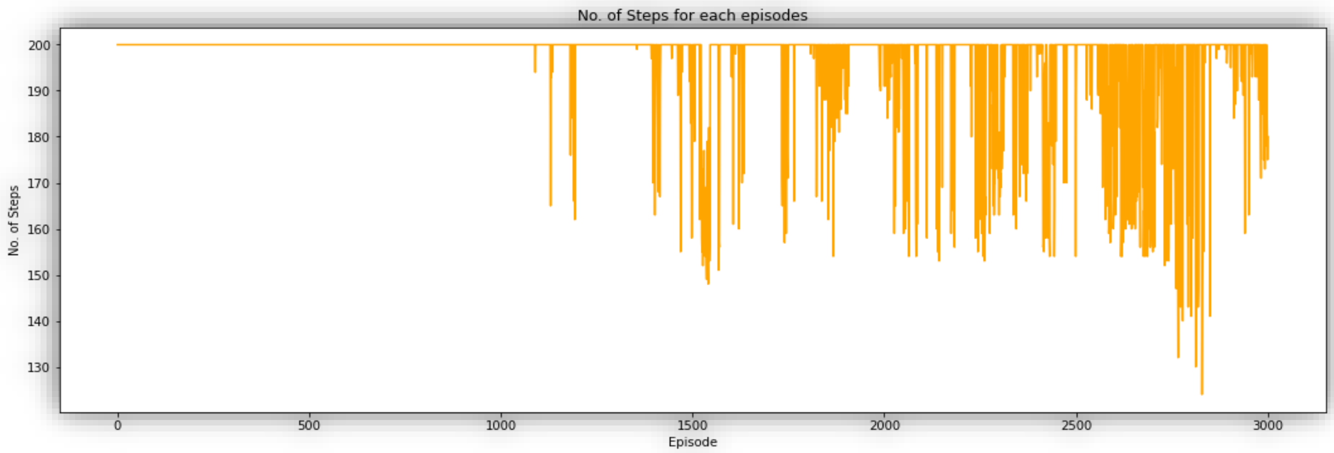
Aggregated Average results of Time Steps over several episodes (3000 episodes) in each pair of alpha & gamma in one figure even we can see the performance variation between each pair.

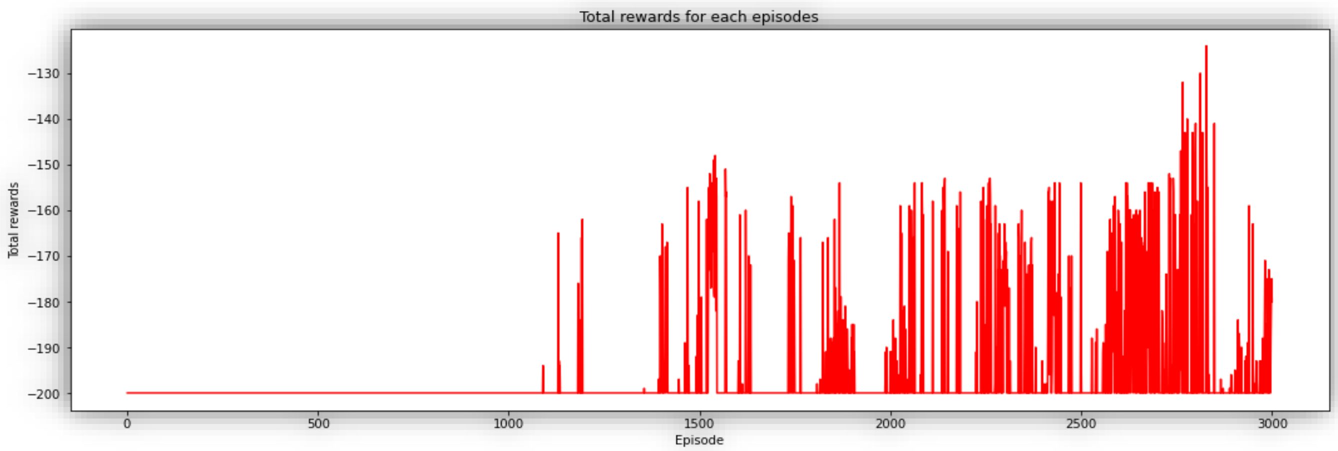


Aggregated Average results of Total Rewards over several episodes (3000 episodes) in each pair of alpha & gamma in one figure even we can see the performance variation between each pair.



According to the above results, we picked the best pair (alpha= .5 & gamma= .9) to retrained it and apply its learned policy on a bunch of test episodes. So, we obtained the following results for training & testing phase.





Another test performance

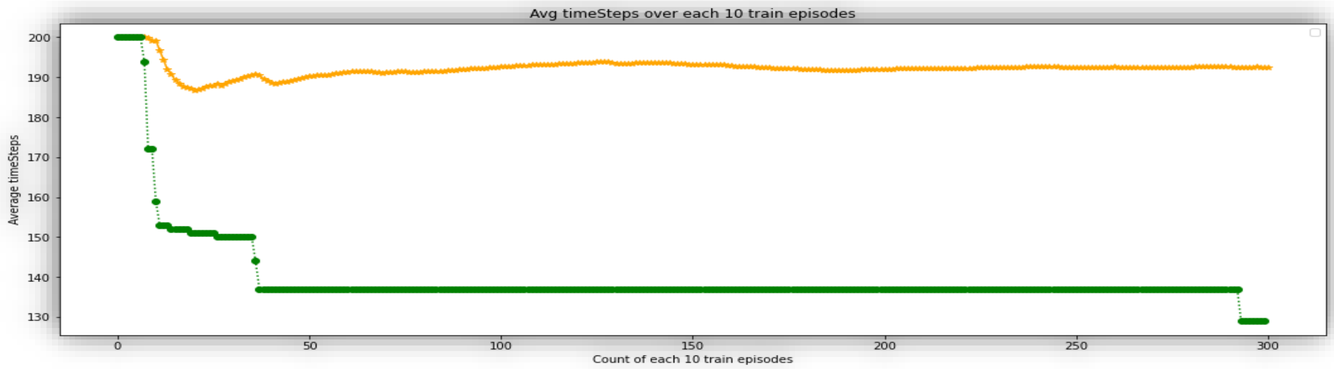
We ran the same number of episodes but with different way,
after every 10 episodes trained with the best pair of Alpha & Gamma, we ran the estimated poli cy in the environment for 5 test episodes and plotted the mean over:

(i) Cumulative reward per episode obtained by the agent.

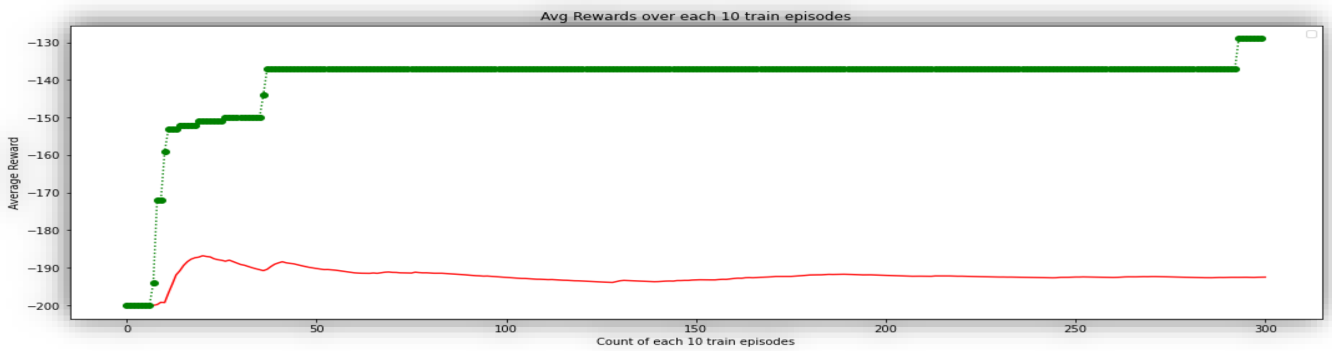
(ii) The number of timesteps required to solve the task per episode of experience.

For both training & test phases.

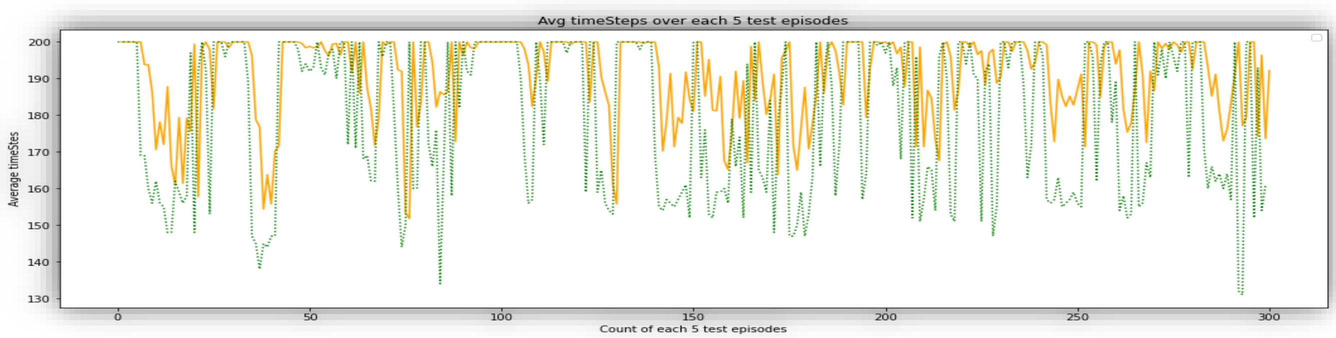
Average TimeSteps of each 10 episodes of train



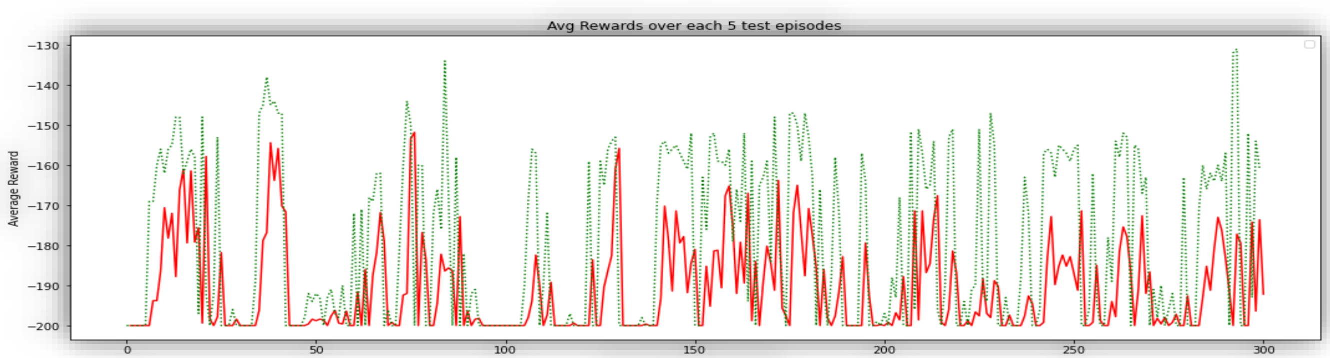
Average Total Rewards of each 10 episodes of train



Average TimeSteps of each 5 episodes of test



Average Total Rewards of each 5 episodes of test



References:

QLearning: https://en.wikipedia.org/wiki/Q-learning

Mountain Car Problem: https://en.wikipedia.org/wiki/Mountain_car_problem

Mountain Car Open AI Gym:
https://www.gymlibrary.ml/environments/classic_control/mountain_car/

Mountain Car Gym Git: https://github.com/openai/gym/wiki/MountainCar-v0

Open AI Gym: https://gym.openai.com/docs/

More Ref: https://github.com/llSourcell/Q_Learning_Explained

Done by:

Ahmed Abdelhamid **21aisa@queensu.ca**