RL ASSIGNMENT Q1 Write Up :

CS21MDS14025

Sauradeep Debnath

PART 1 – PONG with DQN

I have trained DQN in Google Colab using Pytorch. In my code , I have used the Wrapper functions from Open AI baselines as it is.
Notebook name - **Pong_v0_DQN_CS21MDS14025.ipynb**. (Secondarily, I have also implemented Skipframe in **Pong_SkipFrame_CS21MDS14025.ipynb**)

Q1.a) Develop a small code snippet to load the corresponding Gym environment(s) and print out the respective state and action space. Develop a random agent to understand the reward function of the environment. Record your observations.

Q1.a) part : Code :

```
env = gym.make("Pong-v0")#gym.make("ALE/Pong-v5") #gym.make("Pong-v0")#('PongNoFrameskip-v4')
print('Action Space-->',env.action_space)
print('Observation Space-->',env.observation_space)
print('reward range-->',env.reward_range)
print('Meta data --> ',env.metadata)
print('Specifications -->',env.spec)
env.reset()
all_rewards =[]
#for i in range(3000):
timestamp =0
sample_count = 250
while True :
    timestamp += 1
    env.render(mode = 'rgb_array')
    action = env.action_space.sample()
    next_state, reward, done, info = env.step(action)
    if sample_count >0:
      print(' reward is {}'.format(reward ))
      all_rewards.append(reward)
      sample_count -= 1

    if done:
        print('Episode finished after {} timesteps with reward {}'.format(timestamp, reward))

        env.reset()
        break
print('max reward is --> {} and min reward --> {}'.format(max(all_rewards), min(all_rewards)))
env.close()
```

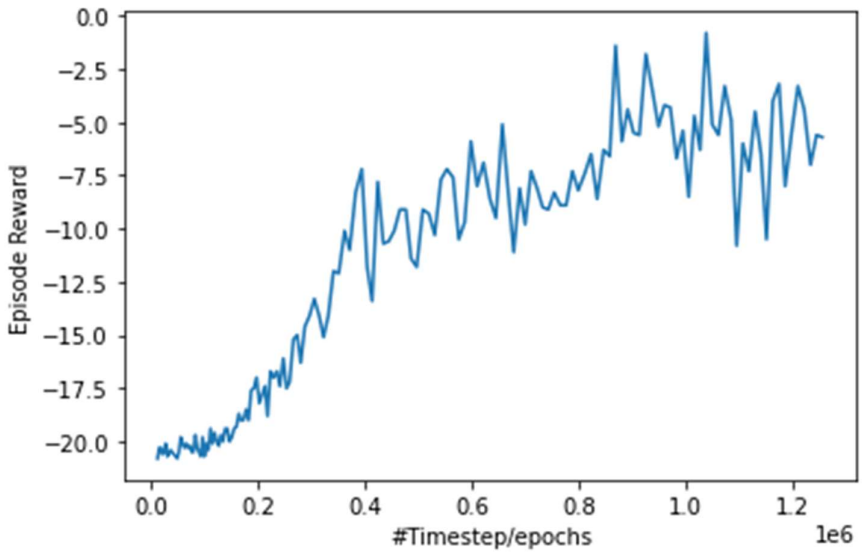
The output being :

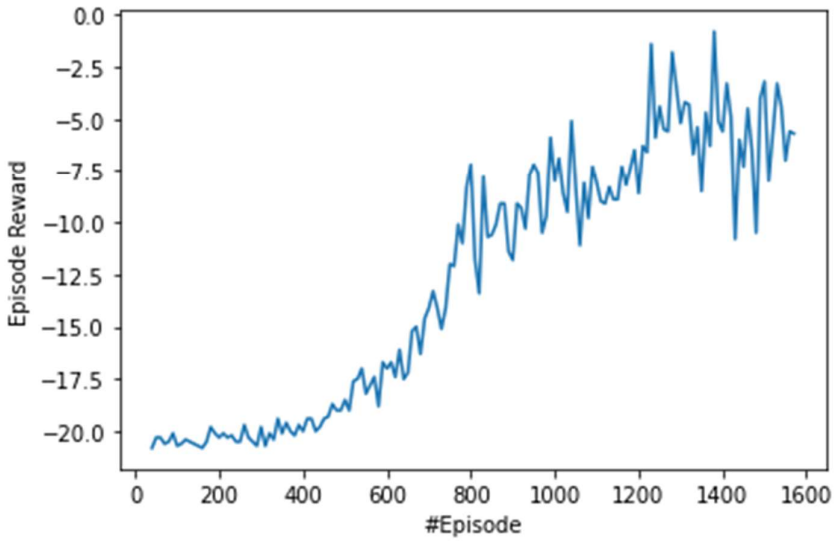
Action Space--> Discrete(6)

Observation Space--> Box(0, 255, (210, 160, 3), uint8)
reward range--> (-inf, inf)
Meta data --> {'render_modes': ['human', 'rgb_array']}
Specifications --> EnvSpec(id='Pong-v0', entry_point='gym.envs.atari:AtariEnv', reward_threshold=None, nondeterministic=False, max_episode_steps=10000, order_enforce=True, autoreset=False, disable_env_checker=False, new_step_api=False, kwargs={'game': 'pong', 'obs_type': 'rgb', 'repeat_action_probability': 0.25, 'full_action_space': False, 'frameskip': (2, 5)}, namespace=None, name='Pong', version=0)
reward is 0.0
. . .
reward is 0.0
reward is 0.0
Episode finished after 1440 timesteps with reward 0.0
max reward is --> 0.0 and min reward --> -1.0

Q 1.b) DQN Implementation – PONG

I have run up to **Episode: 1560; Timestep: 1244571** (1.2 Million) for Pong-v0. After that Google Colab prevented me to use any more GPU & got disconnected.
Architecture - Convolutional Net
I had saved the rewards & model in checkpoint. Here are the plots for 10 -Episode Mean Reward vs Episode / Timestep :





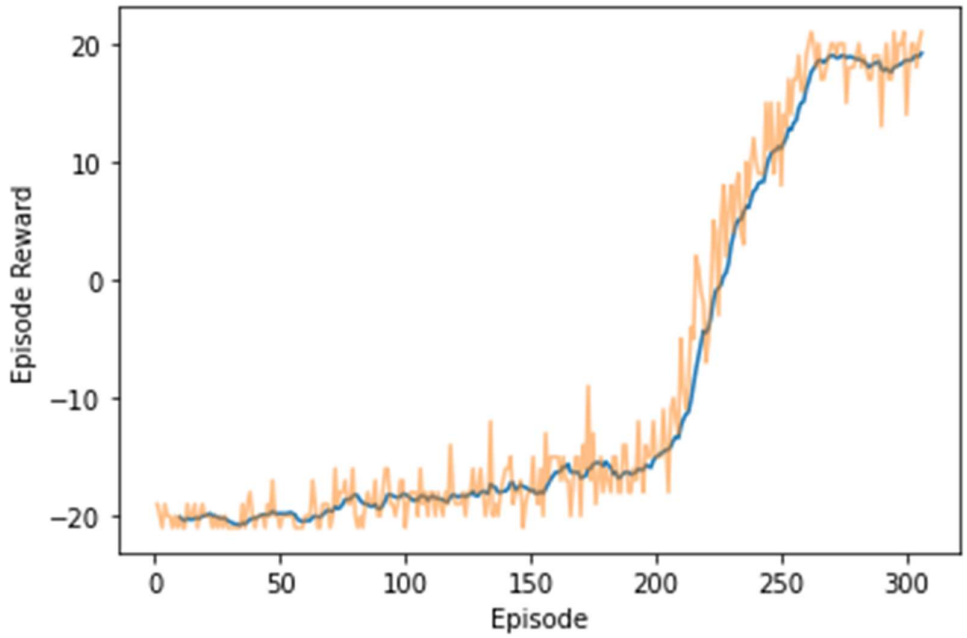
Some sample rewards from the training :

```
[Errno 2] No such file or directory: '/content/drive/MyDrive/Colab_Notebooks_IITH/RL_A4/DQN_Pong.pth'
Saving checkpoint...
Episode: 40;  Timestep: 11383; 10 -Episode Mean reward: -20.8; Episode length: 298.2; Epsilon: 0.96; Loss:
Saving checkpoint...
Episode: 50;  Timestep: 14440; 10 -Episode Mean reward: -20.3; Episode length: 305.7; Epsilon: 0.95; Loss:
Saving checkpoint...
Episode: 60;  Timestep: 17497; 10 -Episode Mean reward: -20.3; Episode length: 305.7; Epsilon: 0.94; Loss:
```

In the beginning & then towards the end, the 10 Episode mean rewards improve from approximately (-20) to (-4 or -5) approximately.

```
Saving checkpoint...
Episode: 1430;  Timestep: 1095161; 10 -Episode Mean reward: -10.8; Episode length: 1025.3; Epsilon: 0.01; L
Saving checkpoint...
Episode: 1440;  Timestep: 1106820; 10 -Episode Mean reward: -6.0; Episode length: 1165.9; Epsilon: 0.01; Lo
Saving checkpoint...
Episode: 1450;  Timestep: 1118042; 10 -Episode Mean reward: -7.3; Episode length: 1122.2; Epsilon: 0.01; Lo
Saving checkpoint...
Episode: 1460;  Timestep: 1129647; 10 -Episode Mean reward: -4.5; Episode length: 1160.5; Epsilon: 0.01; Lo
Saving checkpoint...
Episode: 1470;  Timestep: 1140681; 10 -Episode Mean reward: -6.5; Episode length: 1103.4; Epsilon: 0.01; Lo
Saving checkpoint...
Episode: 1480;  Timestep: 1150990; 10 -Episode Mean reward: -10.5; Episode length: 1030.9; Epsilon: 0.01; L
Saving checkpoint...
Episode: 1490;  Timestep: 1162848; 10 -Episode Mean reward: -4.0; Episode length: 1185.8; Epsilon: 0.01; Lo
Saving checkpoint...
Episode: 1500;  Timestep: 1174431; 10 -Episode Mean reward: -3.2; Episode length: 1158.3; Epsilon: 0.01; Lo
Saving checkpoint...
Episode: 1510;  Timestep: 1185906; 10 -Episode Mean reward: -8.0; Episode length: 1147.5; Epsilon: 0.01; Lo
Saving checkpoint...
Episode: 1520;  Timestep: 1197371; 10 -Episode Mean reward: -5.6; Episode length: 1146.5; Epsilon: 0.01; Lo
Saving checkpoint...
Episode: 1530;  Timestep: 1209781; 10 -Episode Mean reward: -3.3; Episode length: 1241.0; Epsilon: 0.01; Lo
Saving checkpoint...
Episode: 1540;  Timestep: 1221449; 10 -Episode Mean reward: -4.4; Episode length: 1166.8; Epsilon: 0.01; Lo
Saving checkpoint...
Episode: 1550;  Timestep: 1233471; 10 -Episode Mean reward: -7.0; Episode length: 1202.2; Epsilon: 0.01; Lo
Saving checkpoint...
Episode: 1560;  Timestep: 1244571; 10 -Episode Mean reward: -5.6; Episode length: 1110.0; Epsilon: 0.01; Lo
```

PONG – SKIPFRAME – The 10 Episode Mean rewards becomes Positive much sooner. Here are the plots from NOTEBOOK -→ Pong_SkipFrame_CS21MDS14025.ipynb :



```
Episode: 20 Timestep: 17725 Total reward: -20.0 Episode length: 920.3 Epsilon: 0.94 Loss: 89.6150
Episode: 30 Timestep: 26859 Total reward: -20.4 Episode length: 913.4 Epsilon: 0.91 Loss: 66.7486
Episode: 40 Timestep: 36124 Total reward: -20.2 Episode length: 926.5 Epsilon: 0.88 Loss: 32.6168
Episode: 50 Timestep: 46172 Total reward: -19.8 Episode length: 1004.8 Epsilon: 0.85 Loss: 37.8558
Episode: 60 Timestep: 55410 Total reward: -20.5 Episode length: 923.8 Epsilon: 0.82 Loss: 32.0080
Episode: 70 Timestep: 65478 Total reward: -19.5 Episode length: 1006.8 Epsilon: 0.78 Loss: 35.0032
Episode: 80 Timestep: 77752 Total reward: -18.3 Episode length: 1227.4 Epsilon: 0.74 Loss: 48.4658
Episode: 90 Timestep: 89107 Total reward: -19.4 Episode length: 1135.5 Epsilon: 0.71 Loss: 44.1860
Episode: 100 Timestep: 101954 Total reward: -18.1 Episode length: 1284.7 Epsilon: 0.66 Loss: 53.6568
Episode: 110 Timestep: 114918 Total reward: -18.7 Episode length: 1296.4 Epsilon: 0.62 Loss: 53.8707
Episode: 120 Timestep: 128046 Total reward: -18.2 Episode length: 1312.8 Epsilon: 0.58 Loss: 57.1397
Episode: 130 Timestep: 142162 Total reward: -18.3 Episode length: 1411.6 Epsilon: 0.53 Loss: 66.3709
Episode: 140 Timestep: 156770 Total reward: -17.9 Episode length: 1460.8 Epsilon: 0.48 Loss: 69.7467
Episode: 150 Timestep: 172885 Total reward: -17.7 Episode length: 1611.5 Epsilon: 0.43 Loss: 82.4924
Episode: 160 Timestep: 190124 Total reward: -16.6 Episode length: 1723.9 Epsilon: 0.37 Loss: 88.3014
Episode: 170 Timestep: 209430 Total reward: -16.3 Episode length: 1930.6 Epsilon: 0.31 Loss: 93.6989
Episode: 180 Timestep: 229262 Total reward: -15.8 Episode length: 1983.2 Epsilon: 0.24 Loss: 97.7968
Episode: 190 Timestep: 250011 Total reward: -16.3 Episode length: 2074.9 Epsilon: 0.17 Loss: 93.6087
Episode: 200 Timestep: 272907 Total reward: -15.3 Episode length: 2289.6 Epsilon: 0.10 Loss: 95.9001
Episode: 210 Timestep: 299369 Total reward: -13.4 Episode length: 2646.2 Epsilon: 0.01 Loss: 96.4894
Episode: 220 Timestep: 334115 Total reward: -4.3 Episode length: 3474.6 Epsilon: 0.01 Loss: 115.4193
Episode: 230 Timestep: 369932 Total reward: 1.4 Episode length: 3581.7 Epsilon: 0.01 Loss: 106.1303
Episode: 240 Timestep: 400902 Total reward: 7.5 Episode length: 3097.0 Epsilon: 0.01 Loss: 78.2075
Episode: 250 Timestep: 429094 Total reward: 11.3 Episode length: 2819.2 Epsilon: 0.01 Loss: 67.8380
Episode: 260 Timestep: 451980 Total reward: 15.1 Episode length: 2288.6 Epsilon: 0.01 Loss: 48.9952
Episode: 270 Timestep: 470500 Total reward: 18.9 Episode length: 1852.0 Epsilon: 0.01 Loss: 28.3072
Episode: 280 Timestep: 489078 Total reward: 18.8 Episode length: 1857.8 Epsilon: 0.01 Loss: 19.4887
Episode: 290 Timestep: 507919 Total reward: 18.5 Episode length: 1884.1 Epsilon: 0.01 Loss: 13.5874
Episode: 300 Timestep: 527106 Total reward: 18.5 Episode length: 1918.7 Epsilon: 0.01 Loss: 11.9454
Stopping at episode 307 with average rewards of 19.2 in last 10 episodes
```

PART 2 : DQN with Mountain Car

Main Notebook : **Mountain_Car_Final_CS21MDS14025ipynb_3rd_case_OPTIMAL**

Q1.a) Printing Specifications of the Environment :

```
# create replay buffer
replay_size = 150000
replay_buffer = ReplayBuffer(max_size=replay_size)

# create cartpole agent
mountain_car_agent = MountainCarAgent(state_size, action_size, hidden_size, learning_rate)
```

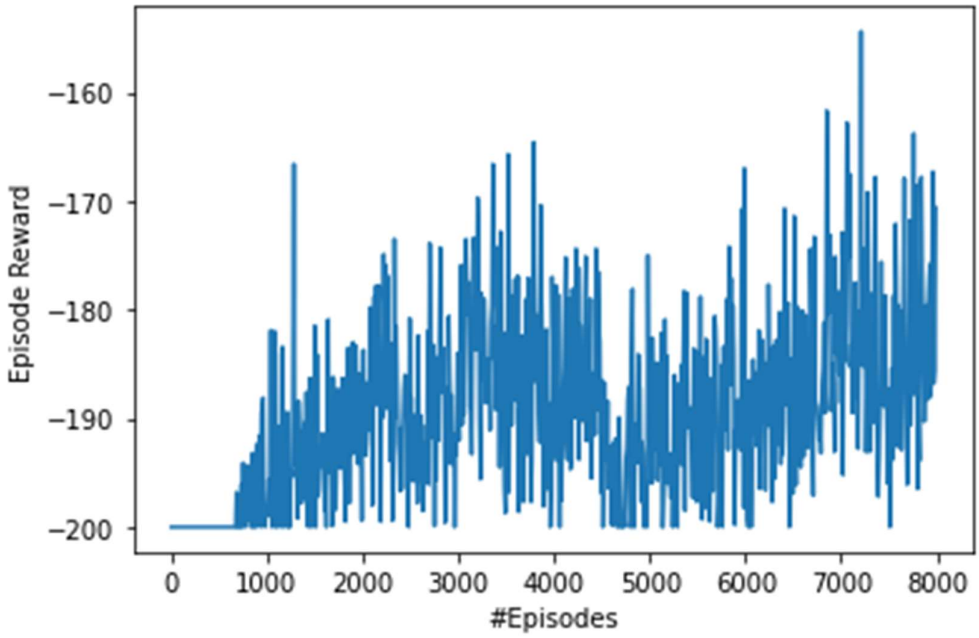
```
Action Space--> Discrete(3)
Observation Space--> Box([-1.2  -0.07], [0.6  0.07], (2,), float32)
reward range--> (-inf, inf)
Meta data -->  {'render_modes': ['human', 'rgb_array', 'single_rgb_array'], 'render_fps': 30}
Specifications --> EnvSpec(id='MountainCar-v0', entry_point='gym.envs.classic_control.mountain_car:MountainCarEnv'
```

Q1.b) DQN Implementation:

Architecture : 3 layer feedforward network / multi-layer perceptron

Convergence Plots:

For the **optimal** hyperparameter choice, we get the reward vs episode plots like this –> We see the rewards are -200 for approximately first 600-700 episodes, after that the 10 episode mean becomes higher (in average)



From the gym documentation, we have –

Observation Space

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

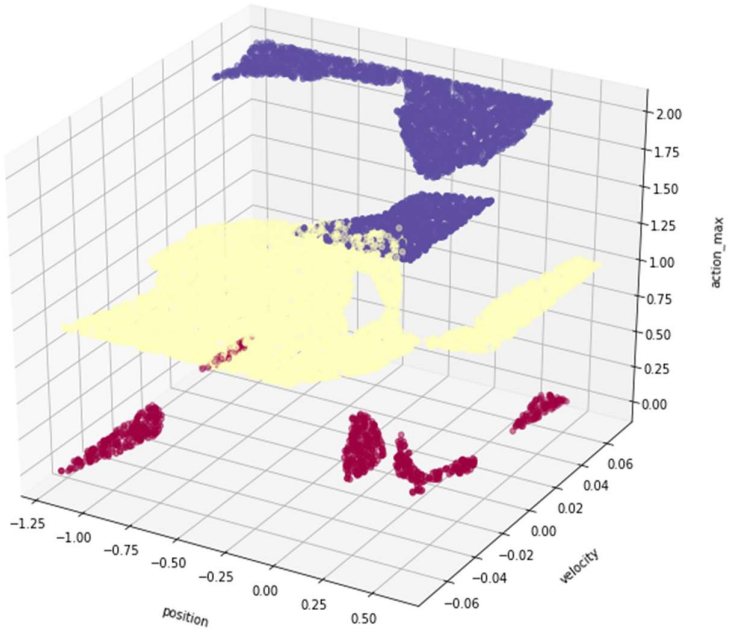
Num	Observation	Min	Max	Unit
0	position of the car along the x-axis	-Inf	Inf	position (m)
1	velocity of the car	-Inf	Inf	position (m)

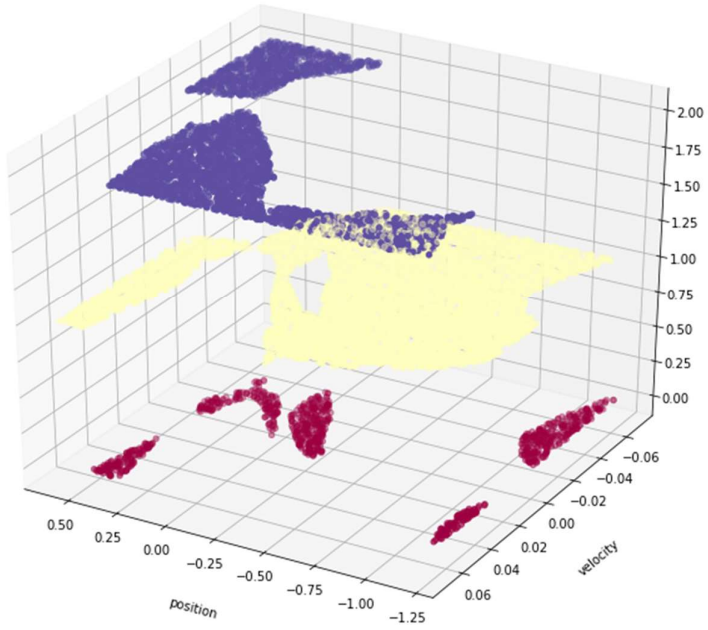
Action Space

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)
2	Accelerate to the right	Inf	position (m)

Using the Same mapping, here are the plots of Action vs Velocity-position combination for the Case 2 Described below (Average speed of convergence) :

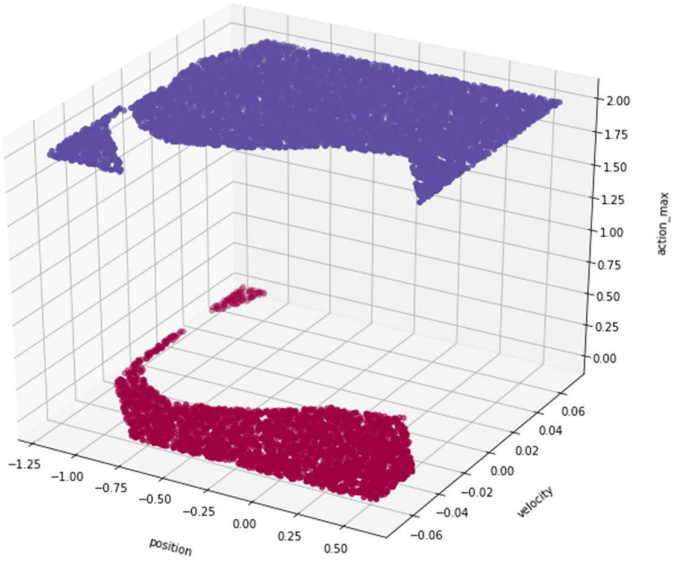


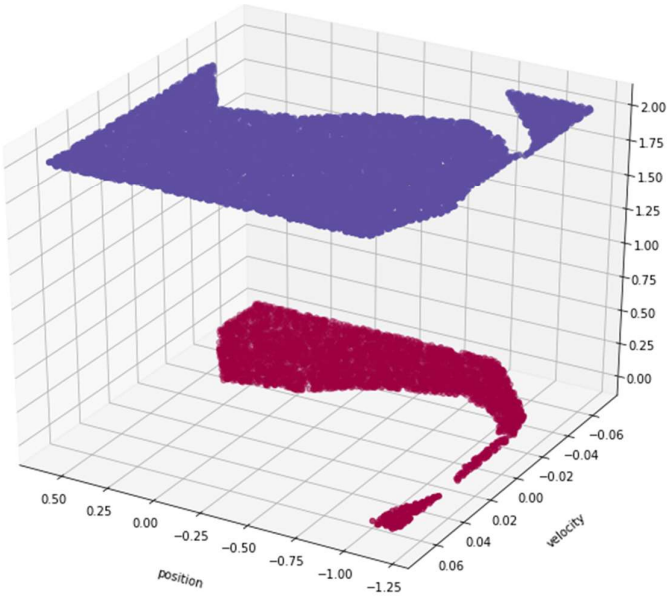


OBSERVATIONS FROM AVERAGE CASE : (MOSTLY from the 2nd figure)
When both/either of Position & Velocity is Positive – we select the violet Action (2) –Accelerate to the right (to strive to reach the goal)
When Position is negative – we select Red Action (0) – Accelerate to the Left (to build momentum)

We can see how these actions are further improved in the OPTIMAL TRAINING CASE below.

here are the plots of Action vs Velocity-position combination for the **OPTIMAL CASE** i.e. Case 3 Described below (FASTEST speed of convergence) :





OBSERVATIONS FROM OPTIMAL CASE :
When Velocity Negative – Choose Action 0 – Accelerate to the Left
When Velocity Positive – Choose Action 2 – Accelerate to the Right
If position at the Negative extremity – Choose Action 0 – Accelerate to the Left

Q1.c) Hyperparameter Tuning :

Due to computational Constraints I have done Hyperparameter tuning only on Mountain Car.
Hyperparameters Tuned –

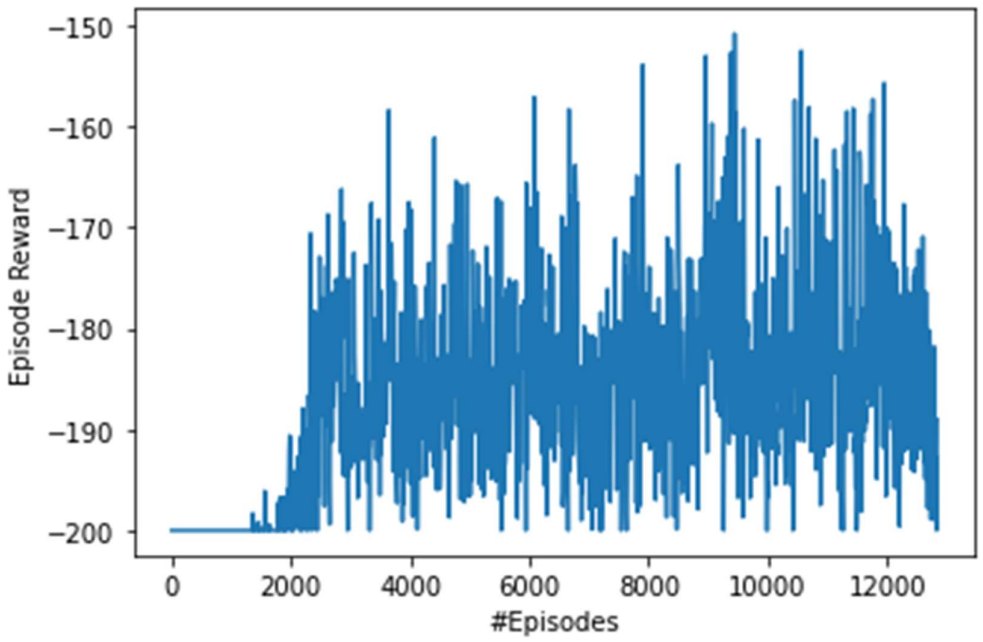
1. update_target_every_ts
2. replay_size

Here the various plots I got. As suggested in the questions, 4 configurations (with 4 values) of the Update_target hyperparameter is discussed below

CASE 1 : AVERAGE PERFORMANCE :

NOTEBOOK NAME - **Mountain_Car_Final_CS21MDS14025ipynb**

```
# hyperparameters
total_episodes = 25000 # run agent for this many episodes
start_training_after = 100 # collect data for this many timesteps before training
hidden_size = 64 # number of units in NN hidden layers
learning_rate = 0.0005 # learning rate for optimizer
update_target_every_ts = 150 #1000 # update target network after this many steps
batch_size = 64 # mini batch size we train network on
discount = 0.9 # gamma value

epsilon_start = 1 #I changed it -->1.0 # epsilon start value
epsilon_min = 0.01 # epsilon end value
epsilon_decay_steps = total_episodes * .1 # decay epsilon over this many episodes
epsilon_step = (epsilon_start - epsilon_min)/(epsilon_decay_steps) # decrement epsilon by this amount every timestep

# create replay buffer
replay_size = 150000
replay_buffer = ReplayBuffer(max_size=replay_size)

# create cartpole agent
mountain_car_agent = MountainCarAgent(state_size, action_size, hidden_size, learning_rate)
```



CASE 2 – BAD PEROFRMANCE

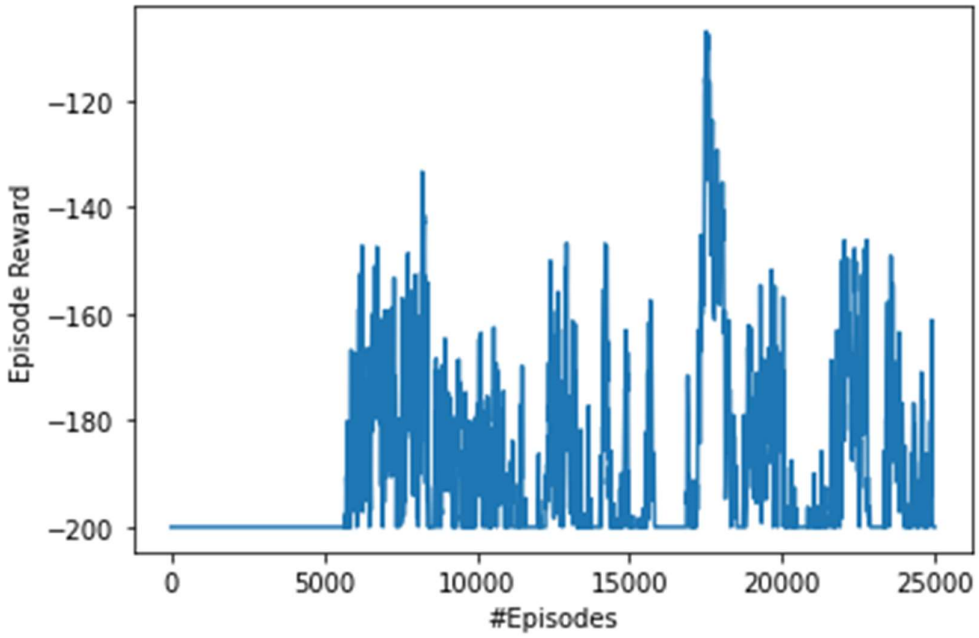
NOTEBOOK - RL A4 Q1c Mountain HYP Longer Convergence.ipynb

```
# hyperparameters
total_episodes = 25000 # run agent for this many episodes
start_training_after = 100 # collect data for this many timesteps before training
hidden_size = 64 # number of units in NN hidden layers
learning_rate = 0.0005 # learning rate for optimizer
update_target_every_ts = 1000 # update target network after this many steps
batch_size = 64 # mini batch size we train network on
discount = 0.9 # gamma value

epsilon_start = 1 #I changed it -->1.0 # epsilon start value
epsilon_min = 0.01  # epsilon end value
epsilon_decay_steps = total_episodes * .1 # decay epsilon over this many episodes
epsilon_step = (epsilon_start - epsilon_min)/(epsilon_decay_steps) # decrement epsilon by this amount every timeste

# create replay buffer
replay_size = 15000
replay_buffer = ReplayBuffer(max_size=replay_size)

# create cartpole agent
mountain_car_agent = MountainCarAgent(state_size, action_size, hidden_size, learning_rate)
```

OBSERVATIONS --- Here, compared to the optimal configuration , the update target is done less frequently, Replay buffer size is smaller.
The game starts converging (getting rewards higher than -200) after 5K episodes (as opposed to ~1.6K in case of OPTIMAL configuration – CASE 1) .
Compared to CASE 1 & CASE 3, even after we start getting higher rewards – there are plenty of -200 Rewards still

In short – Higher convergence time, --- in general lower (more negative) rewards even after the convergence starts.

INSIGHTS - This makes sense as a lot of initial episodes have -200 (until you reach a goal) – the Replay buffer size needs to be bigger.
The Target needs to be updated more frequently as there are only 200 timesteps in a given episode.



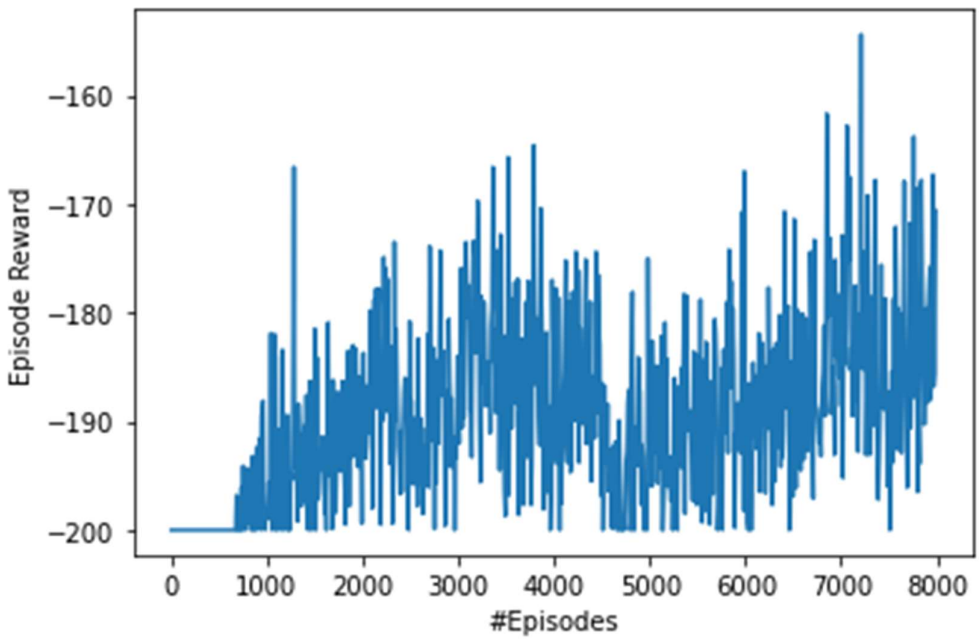
CASE 3 – OPTIMAL CASE / BEST PERFORMANCE :

This is same model that is discussed in 1.b) above.

The above assumptions are verified by the third & final Hyperparameter configuration we have : when we reduce **update_target_every_ts** from 1000 in Case 1 to 100 in Case 2 to 50 in Case 3 FINALLY.

Here the rewards start improving as quickly as approx. 600-700 episodes (as opposed to approx. 1700 episodes in Case 2 & 5000+ episodes in Case 3)

NOTEBOOK : Mountain_Car_Final_CS21MDS14025ipynb_3rd_case_OPTIMAL



CODE for CASE 3 Showing Hyperparameters used → (NOTEBOOK : Mountain_Car_Final_CS21MDS14025ipynb_3rd_case_OPTIMAL)

```
# hyperparameters
total_episodes = 8000 # run agent for this many episodes
start_training_after = 100 # collect data for this many timesteps before training
hidden_size = 64 # number of units in NN hidden layers
learning_rate = 0.0005 # learning rate for optimizer
update_target_every_ts = 50 #1000 # update target network after this many steps
batch_size = 64 # mini batch size we train network on
discount = 0.9 # gamma value

epsilon_start = 1 #I changed it -->1.0 # epsilon start value
epsilon_min = 0.01  # epsilon end value
epsilon_decay_steps = total_episodes * .1 # decay epsilon over this many episodes
epsilon_step = (epsilon_start - epsilon_min)/(epsilon_decay_steps) # decrement epsilon by this amount every timestep

# create replay buffer
replay_size = 150000
replay_buffer = ReplayBuffer(max_size=replay_size)

# create cartpole agent
mountain_car_agent = MountainCarAgent(state_size, action_size, hidden_size, learning_rate)
```

CASE 4 – Following the same line of thought, here we reduced the update_target_every_ts from 50 in CASE 3 to 25 in CASE 4

```
# hyperparameters
total_episodes = 5000 # run agent for this many episodes
start_training_after = 100 # collect data for this many timesteps before training
hidden_size = 64 # number of units in NN hidden layers
learning_rate = 0.0005 # learning rate for optimizer

update_target_every_ts = 25 #1000 # update target network after this many steps

batch_size = 64 # mini batch size we train network on
discount = 0.9 # gamma value

epsilon_start = 1 #I changed it -->1.0 # epsilon start value
epsilon_min = 0.01  # epsilon end value
epsilon_decay_steps = total_episodes * .1 # decay epsilon over this many episodes
epsilon_step = (epsilon_start - epsilon_min)/(epsilon_decay_steps) # decrement epsilon by this amount every timestep

# create replay buffer
replay_size = 150000
replay_buffer = ReplayBuffer(max_size=replay_size)

# create cartpole agent
mountain_car_agent = MountainCarAgent(state_size, action_size, hidden_size, learning_rate)
```

OBSERVATION – There are no further visible changes upon reducing update_target_evry_ts parameter from 50 to 25.