CSE574 Introduction to Machine Learning

Project 4

Bonus Part

Implementation of Two Environments from OpenAI's Gym Library

Project Report

Submitted By

Karan Manchandia

Abstract

In this bonus part, the task is to implement DQN using two environments from OpenAI's Gym Library. Gym is an open source interface to reinforcement learning tasks. It is a collection of test environments that we can use to workout reinforcement learning algorithms. The first environment I have chosen is CartPole-v1. The task is to balance a pole on a frictionless moving cart on which some force is applied.

The second environment used is a Atari environment from OpenAI's Gym Library called AlienNoFrameskip-v4. The task is to maximize the score in an alien game. In this environment, the observation is a screen RGB image which is represented as an array of shape (210,160,3). The action that is sent to the environment is applied and the state is returned without skipping any frames. The game resembles through the Arcade Learning Environment which uses Stella Atari emulator.

Introduction:

- **What I have Implemented?**

  I explore the application of DQN in 2 environments: the basic gym Cartpole environment where the goal is to balance a pole on a cart and the Atari Alien environment where the goal is to avoid the aliens and collect as much food particles as possible.

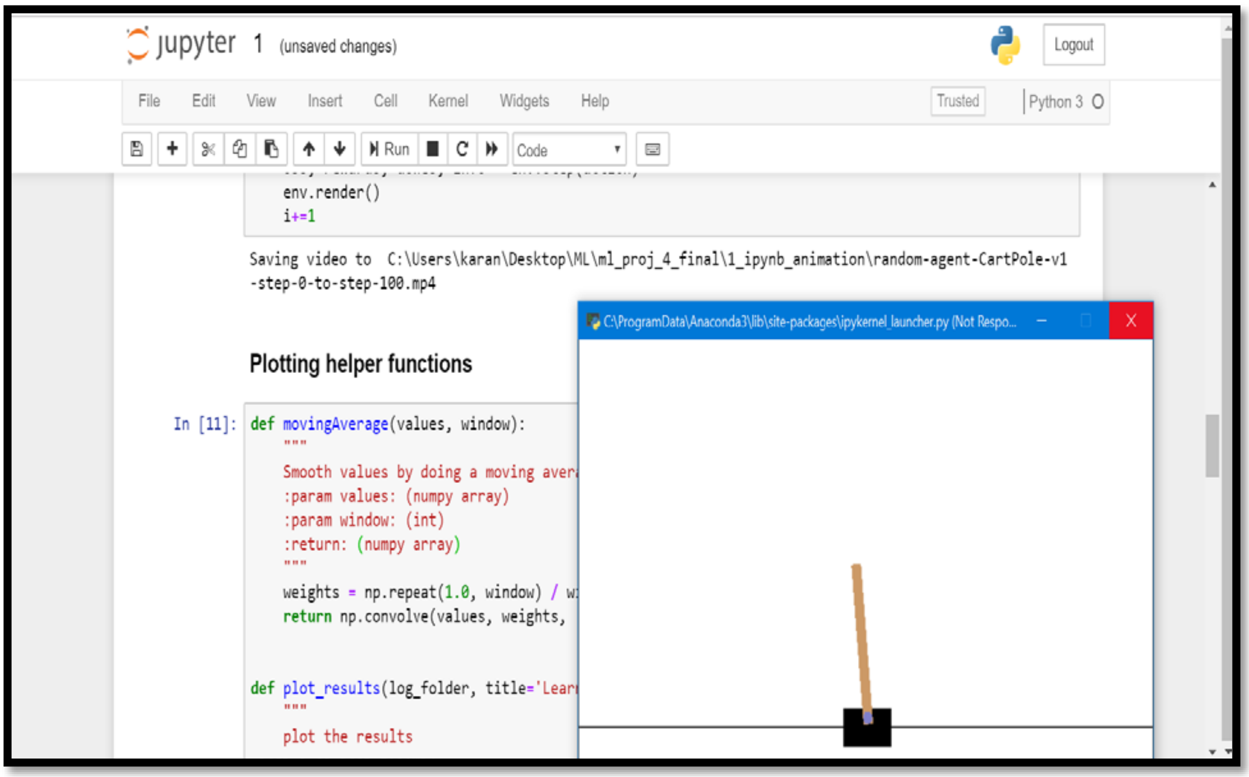
- **How the agent is trained?**
  - In the Cartpole environment, I use a Multilayer Perceptron (Neural Network) to estimate the Q-value function.
  - In the Atari environment, I have used a Convolutional Neural Network to estimate the Q–value function as a CNN is more suited for the Atari environment pixels.

- For the implementation, I have used stable baselines which is a reinforcement learning library that tries to match scikit-learn's easy to use interface. It contains pre-implemented agents including DQN. It is even possible to train a model using just one line of code.
- **How much time the does agent take to train itself ?**
  - For the bonus task 1 CartPole-v1 the DQN agent trains fairly quickly. It takes about 2 minutes to train.
  - For the bonus task 2 AlienNoFrameskip-v4, the agent takes over 2 hours to train.
  - This difference in time is mainly because of difference in complexity in classic control tasks and the Atari environment.

Observations, Results, Screenshots and Graphs:

Bonus Part 1:

Screenshots:


jupyter 1 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py (Not Respo...

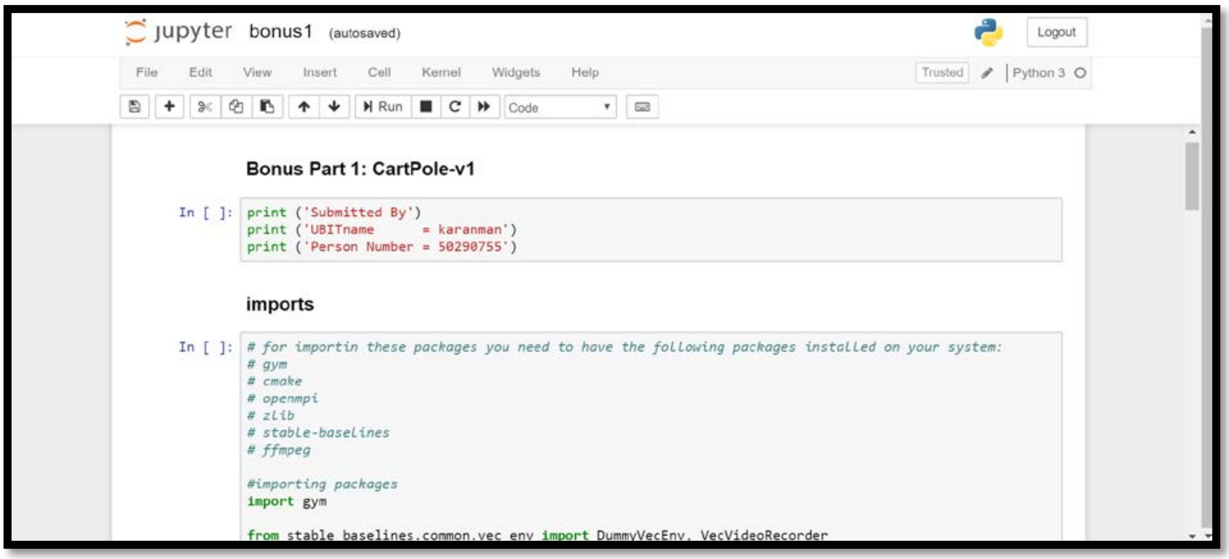
```
    env.render()
    i+=1
```

```
Saving video to  C:\Users\karan\Desktop\ML\ml_proj_4_final\1_ipynb_animation\random-agent-CartPole-v1
-step-0-to-step-100.mp4
```

Plotting helper functions

```
In [11]: def movingAverage(values, window):
             """
             Smooth values by doing a moving aver
             :param values: (numpy array)
             :param window: (int)
             :return: (numpy array)
             """
             weights = np.repeat(1.0, window) / w
             return np.convolve(values, weights,


         def plot_results(log_folder, title='Lear
             """
             plot the results
```

Figure 1: CartPole ffmpeg output



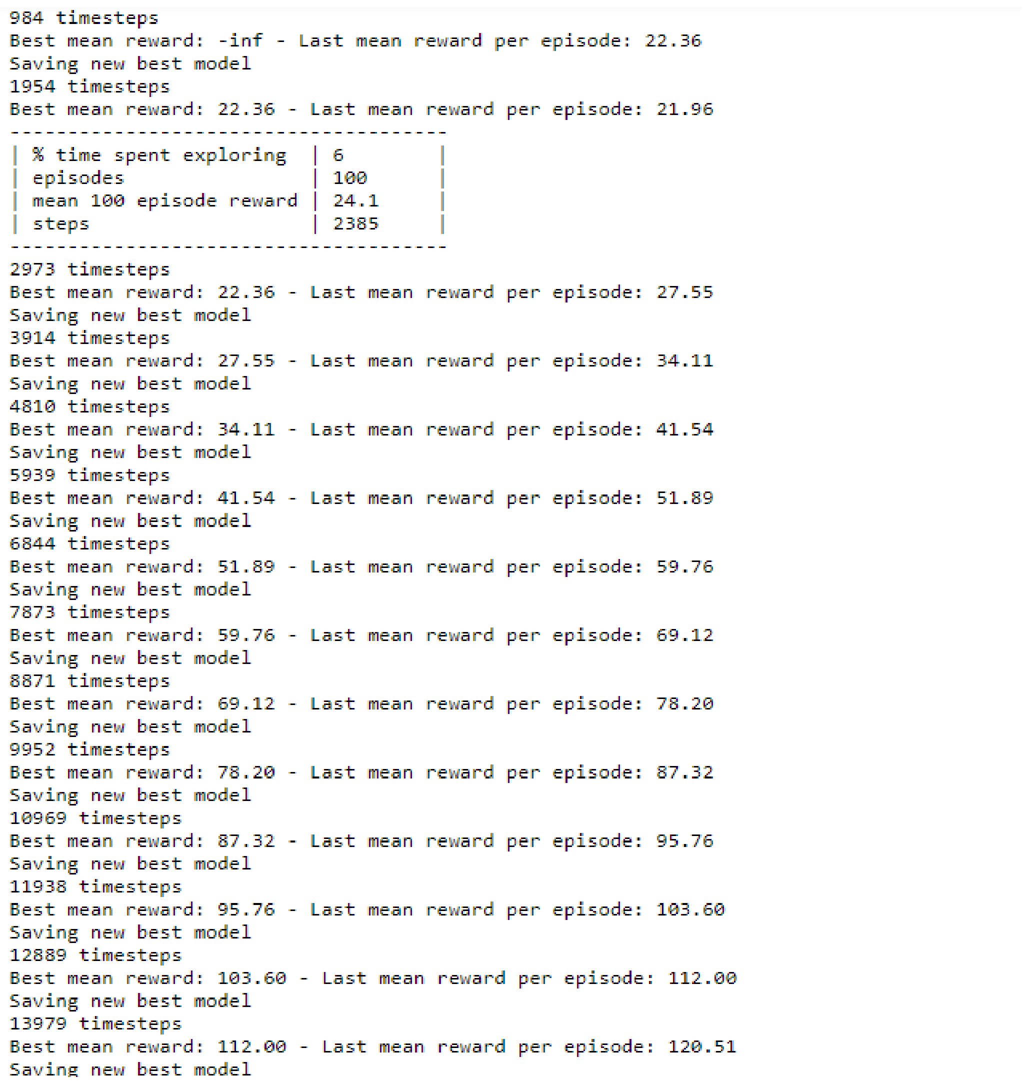
Bonus Part 1: CartPole-v1

```
In [ ]: print ('Submitted By')
        print ('UBITname      = karanman')
        print ('Person Number = 50290755')
```

imports

```
In [ ]: # for importin these packages you need to have the following packages installed on your system:
        # gym
        # cmake
        # openmpi
        # zlib
        # stable-baselines
        # ffmpeg

        #importing packages
        import gym

        from stable_baselines.common.vec_env import DummyVecEnv, VecVideoRecorder
```

Figure 2: Bonus Part 1 Screenshot



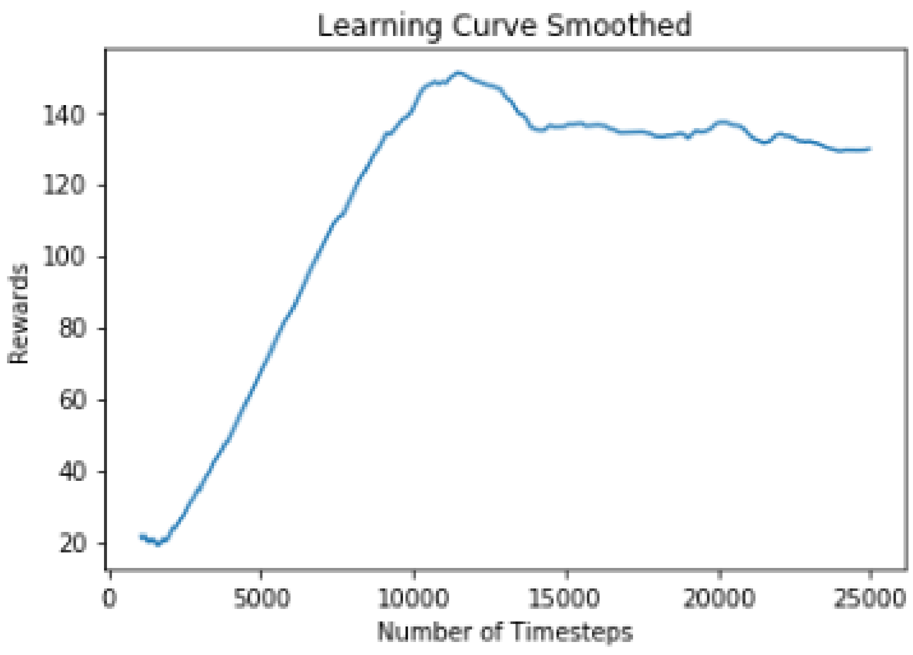
```
984 timesteps
Best mean reward: -inf - Last mean reward per episode: 22.36
Saving new best model
1954 timesteps
Best mean reward: 22.36 - Last mean reward per episode: 21.96
--------------------------------------
| % time spent exploring  | 6        |
| episodes                | 100      |
| mean 100 episode reward | 24.1     |
| steps                   | 2385     |
--------------------------------------
2973 timesteps
Best mean reward: 22.36 - Last mean reward per episode: 27.55
Saving new best model
3914 timesteps
Best mean reward: 27.55 - Last mean reward per episode: 34.11
Saving new best model
4810 timesteps
Best mean reward: 34.11 - Last mean reward per episode: 41.54
Saving new best model
5939 timesteps
Best mean reward: 41.54 - Last mean reward per episode: 51.89
Saving new best model
6844 timesteps
Best mean reward: 51.89 - Last mean reward per episode: 59.76
Saving new best model
7873 timesteps
Best mean reward: 59.76 - Last mean reward per episode: 69.12
Saving new best model
8871 timesteps
Best mean reward: 69.12 - Last mean reward per episode: 78.20
Saving new best model
9952 timesteps
Best mean reward: 78.20 - Last mean reward per episode: 87.32
Saving new best model
10969 timesteps
Best mean reward: 87.32 - Last mean reward per episode: 95.76
Saving new best model
11938 timesteps
Best mean reward: 95.76 - Last mean reward per episode: 103.60
Saving new best model
12889 timesteps
Best mean reward: 103.60 - Last mean reward per episode: 112.00
Saving new best model
13979 timesteps
Best mean reward: 112.00 - Last mean reward per episode: 120.51
Saving new best model
```

Figure 3: Screenshot Showing Outputs

Observation Table:

Timesteps	Best Mean Reward
1954	22.36
4810	34.11
8871	69.12
12889	103.60
14950	120.51
18899	141.58

Graphs:

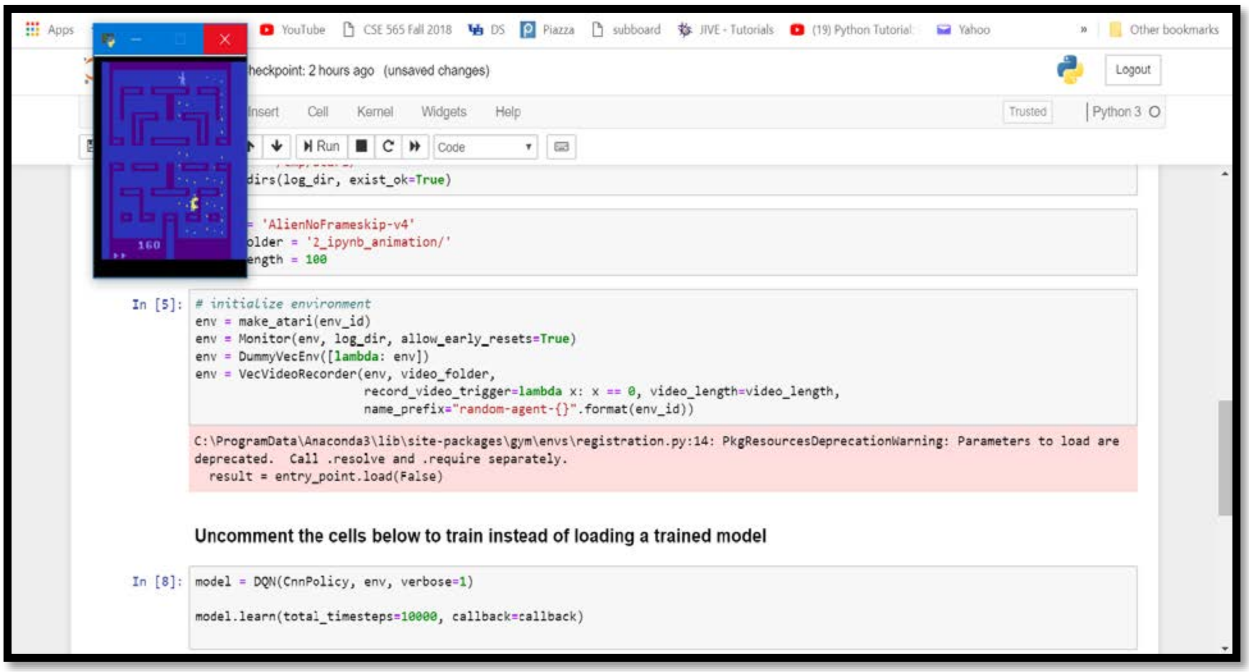


Explanation:

We can clearly see that as number of timesteps increases the rewards also increases. As the number of timsteps increases the agent moves form exploration to exploitation. As the agent began to exploit, it starts using q-values and has better moves, hence he reward also increases.

Bonus Part 2 (AlienNoFrameskip-v4):

Screenshots:



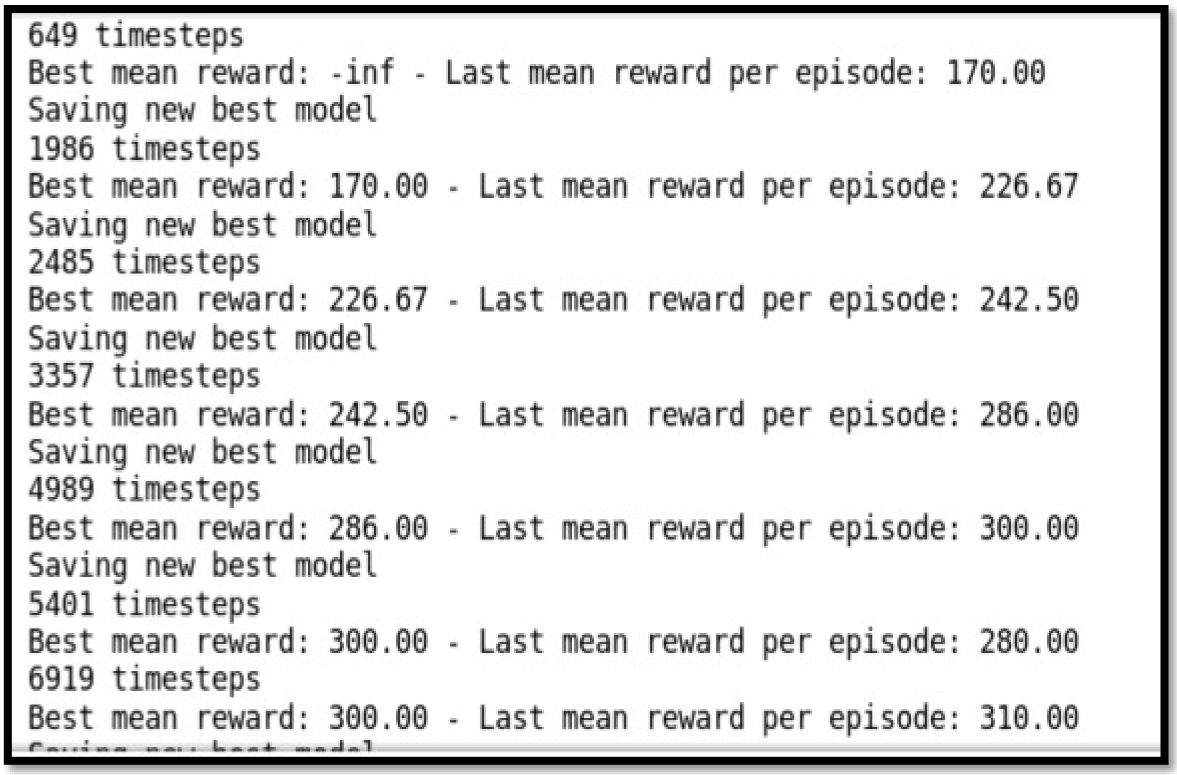
```
dirs(log_dir, exist_ok=True)

= 'AlienNoFrameskip-v4'
older = '2_ipynb_animation/'
ength = 100
```

```
In [5]: # initialize environment
        env = make_atari(env_id)
        env = Monitor(env, log_dir, allow_early_resets=True)
        env = DummyVecEnv([lambda: env])
        env = VecVideoRecorder(env, video_folder,
                               record_video_trigger=lambda x: x == 0, video_length=video_length,
                               name_prefix="random-agent-{}".format(env_id))

        C:\ProgramData\Anaconda3\lib\site-packages\gym\envs\registration.py:14: PkgResourcesDeprecationWarning: Parameters to load are
        deprecated.  Call .resolve and .require separately.
          result = entry_point.load(False)
```

Uncomment the cells below to train instead of loading a trained model

```
In [8]: model = DQN(CnnPolicy, env, verbose=1)

        model.learn(total_timesteps=10000, callback=callback)
```

Figure 4 Bonus Task 2 Output Screenshot



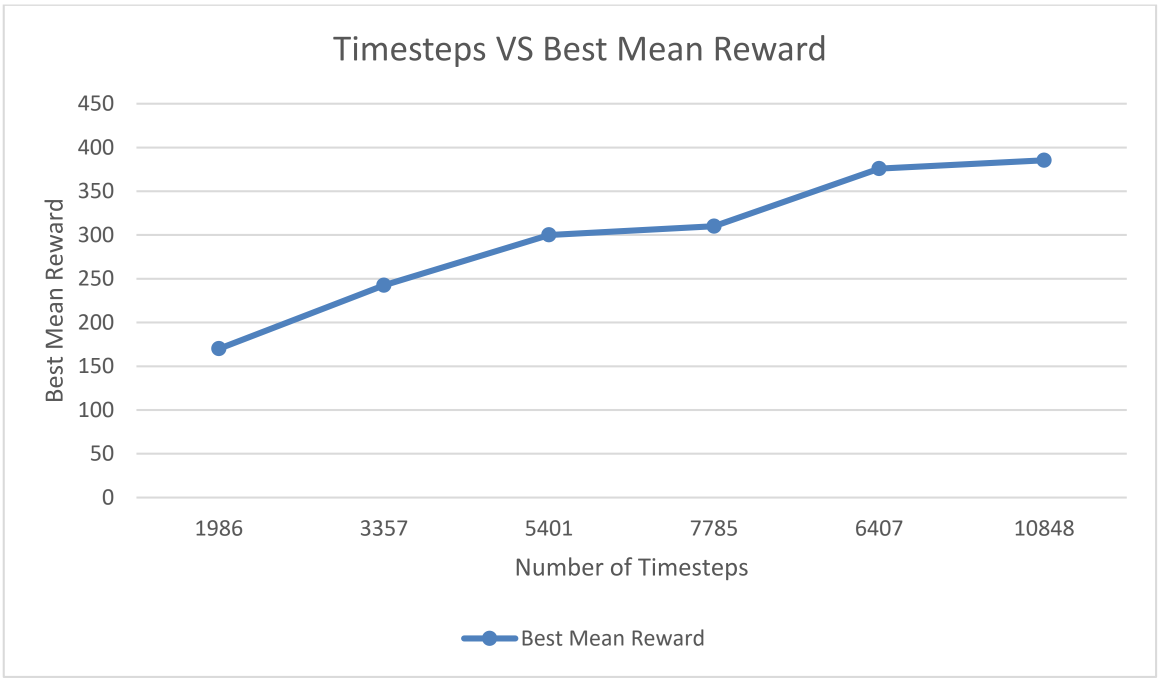
```
649 timesteps
Best mean reward: -inf - Last mean reward per episode: 170.00
Saving new best model
1986 timesteps
Best mean reward: 170.00 - Last mean reward per episode: 226.67
Saving new best model
2485 timesteps
Best mean reward: 226.67 - Last mean reward per episode: 242.50
Saving new best model
3357 timesteps
Best mean reward: 242.50 - Last mean reward per episode: 286.00
Saving new best model
4989 timesteps
Best mean reward: 286.00 - Last mean reward per episode: 300.00
Saving new best model
5401 timesteps
Best mean reward: 300.00 - Last mean reward per episode: 280.00
6919 timesteps
Best mean reward: 300.00 - Last mean reward per episode: 310.00
```

Figure 5 Bonus Task 2 Output Screenshot

Observation Table:

Timesteps	Best Mean Reward
1986	170
3357	242.50
5401	300
7785	310
6407	375.83
10848	385.38

Graphs:



Explanation:

As seen above in Bonus Task 1, In task 2 the similar trend is seen. We can clearly see that as number of timesteps increases the rewards also increases. As the number of timesteps increases the agent moves form exploration to exploitation. As the agent began to exploit, it starts using q-values and has better moves, hence the reward also increases.

References:

- https://gym.openai.com/docs/
- https://mc.ai/openai-gym-environment-full-list/
- https://www.kdnuggets.com/2018/03/5-things-reinforcement-learning.html

- https://www.slideshare.net/AmazonWebServices/cmp305-deep-learning-on-aws-made-easycmp305
- https://medium.com/@m.alzantot/deep-reinforcement-learning-demysitifed-episode-2-policy-iteration-value-iteration-and-q-978f9e89ddaa