Reinforcement Learning

Lab Assignment 5

190020007-190020021-190020039

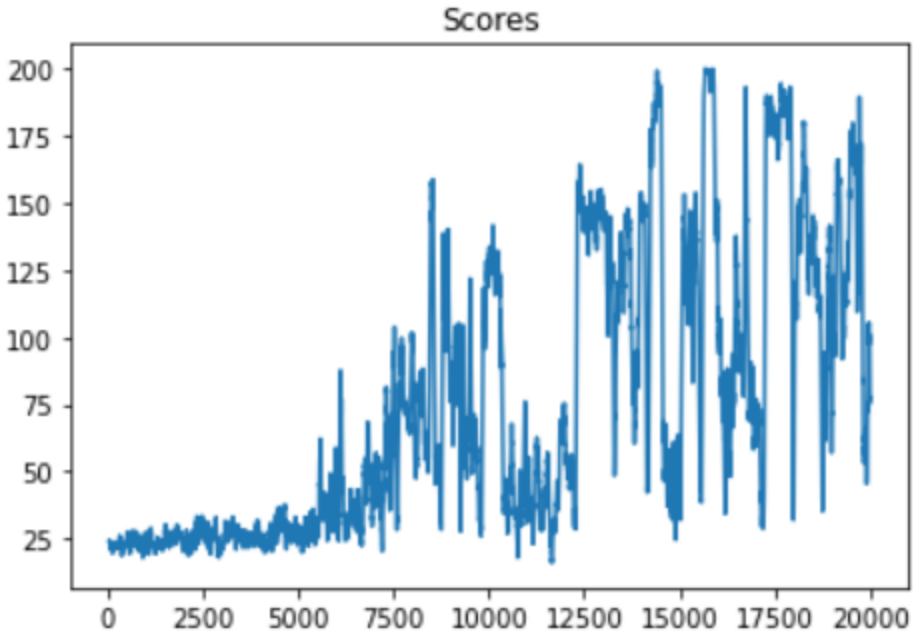
Part 1:

190020007

Cartpole(cartpole-v0 - optimal reward = 200) : In tile coding, we have taken 3 tiles. We have divided each state in the state space in 10 parts that is the no. of bins is 10. Cartpole has 4 states and 2 actions. Therefore, each tile has a dimension of 10*10*10*10 and there are 3 such tiles. Then, we have used a q-table and updated the values of the same after starting all from 0. We used both SARSA and Q-Learning.

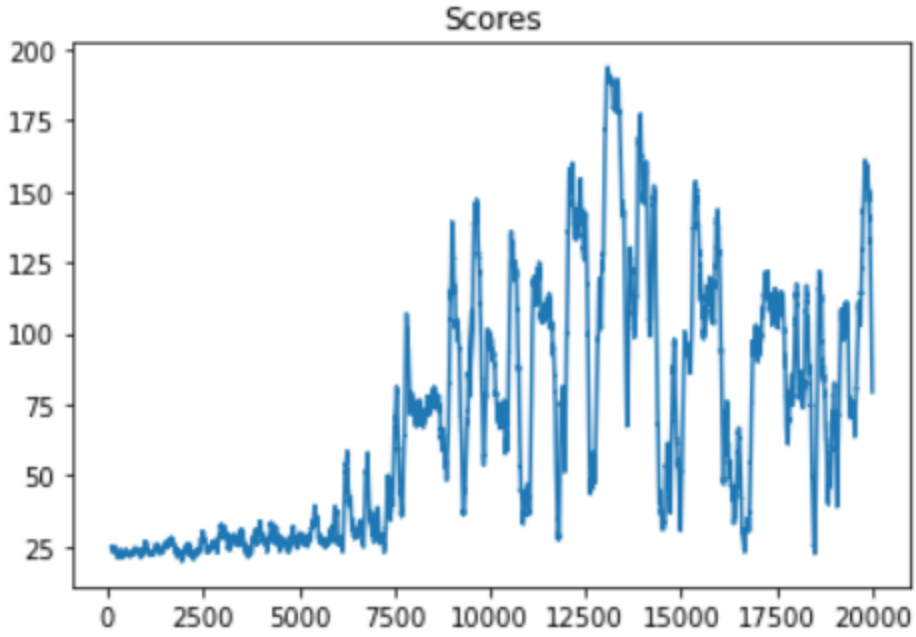
Q-Learning :-

The maximum average reward reached is 199.53.



SARSA :-

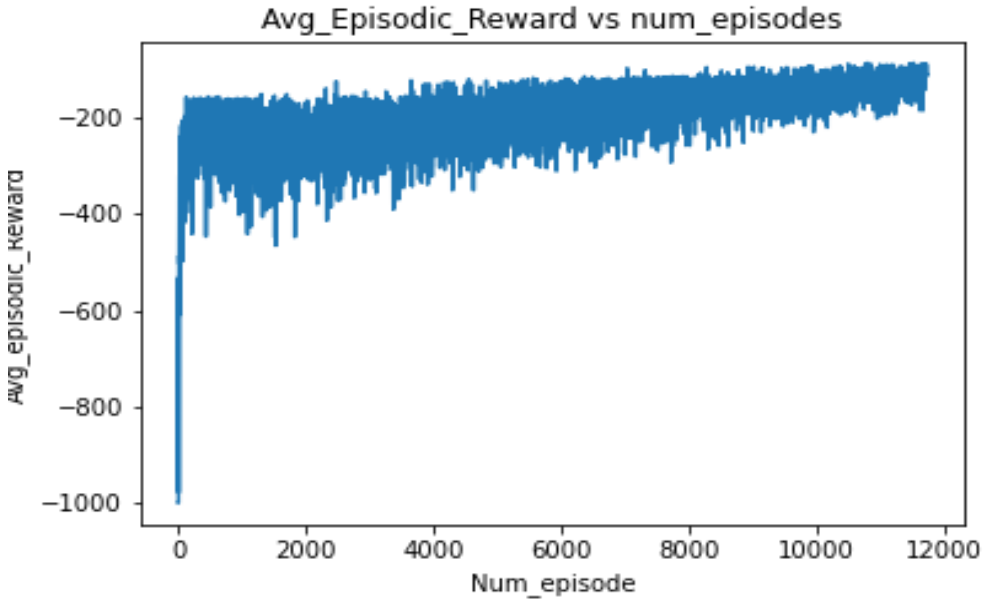
The maximum average reward reached is 193.59.



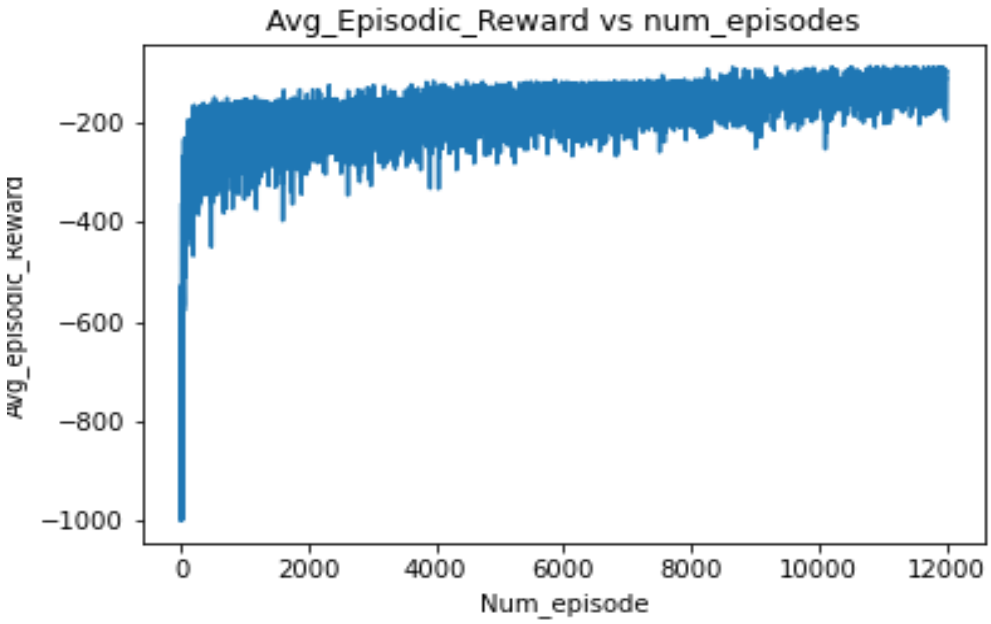
190020021:

MountainCar-v0:

Tile coding with Q_Learning:



Tile coding with SARSA:



190020039:

Task: Acrobot-v1

Tiling:

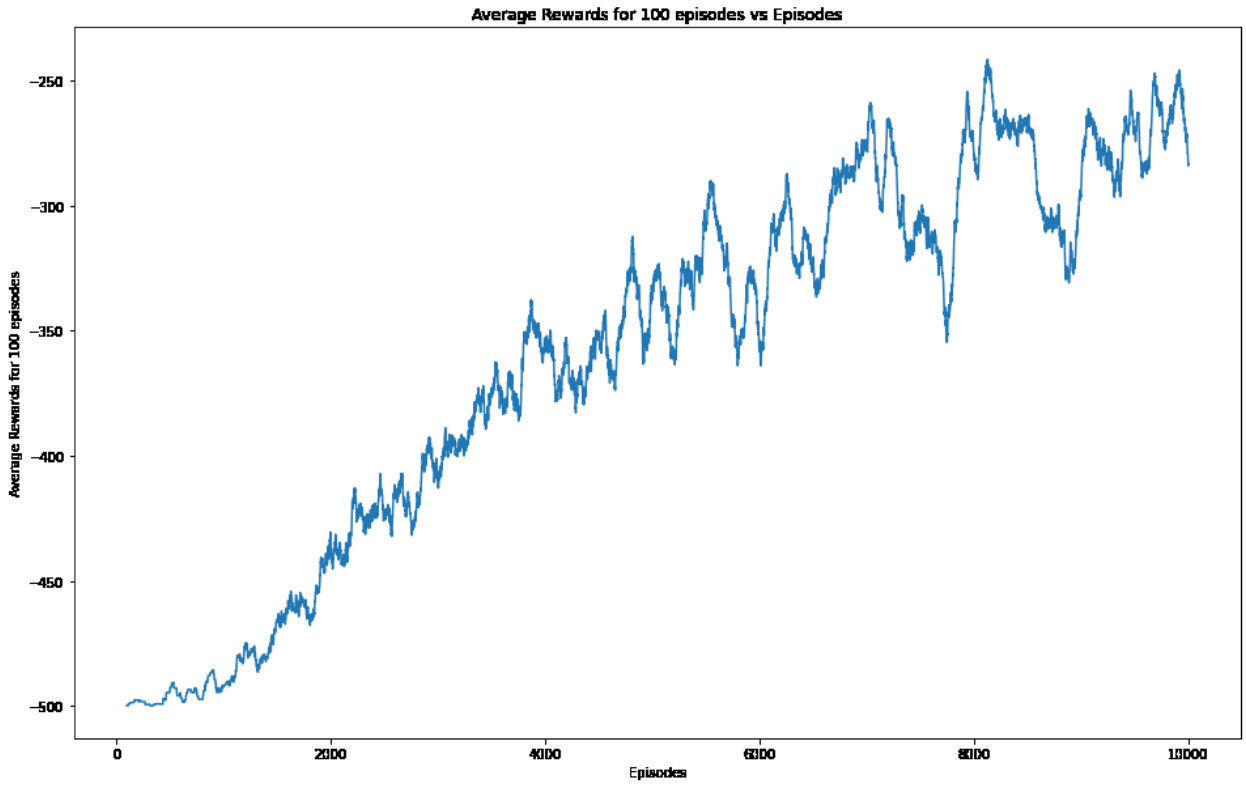
Number of tilings = 3

Bins per feature in each tiling = 10

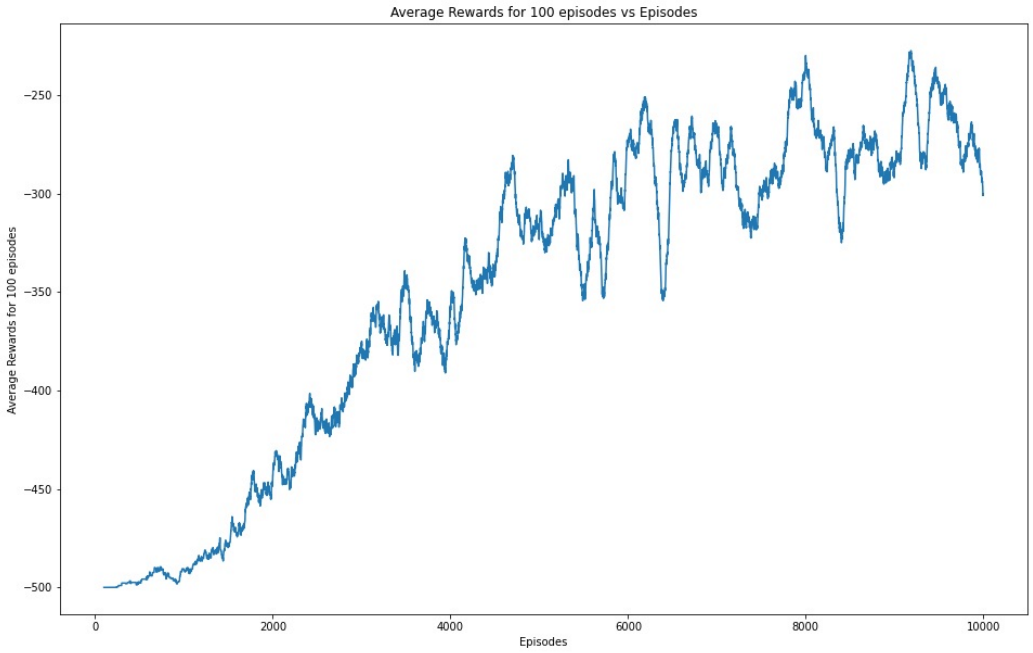
Offsets considered = [-(high-low)/(3*bins),0,(high-low)/(3*bins)]

Results:

Q-learning:



SARSA:


Average Rewards for 100 episodes vs Episodes
Average Rewards for 100 episodes
−250
−300
−350
−400
−450
−500
0
2000
4000
6000
8000
10000
Episodes

Part 2: REINFORCE (policy gradient algorithm)

Tasks considered: MountainCar-v0, LunarLander-v2, Acrobot-v1, CartPole-v1

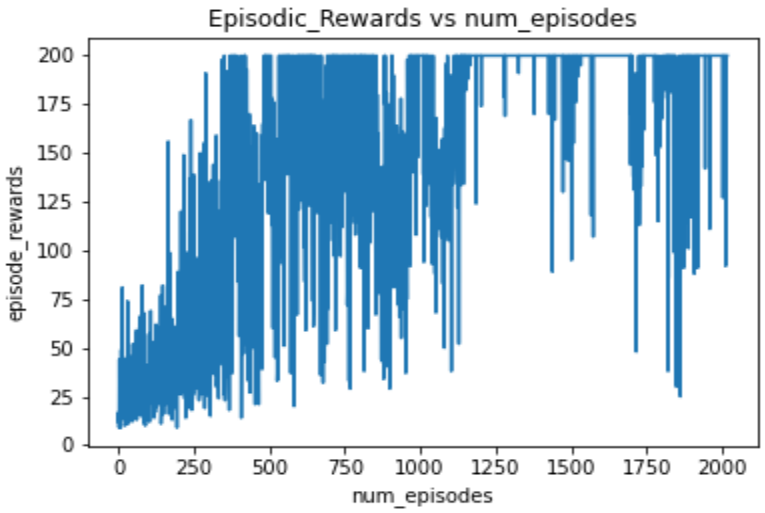
Results:

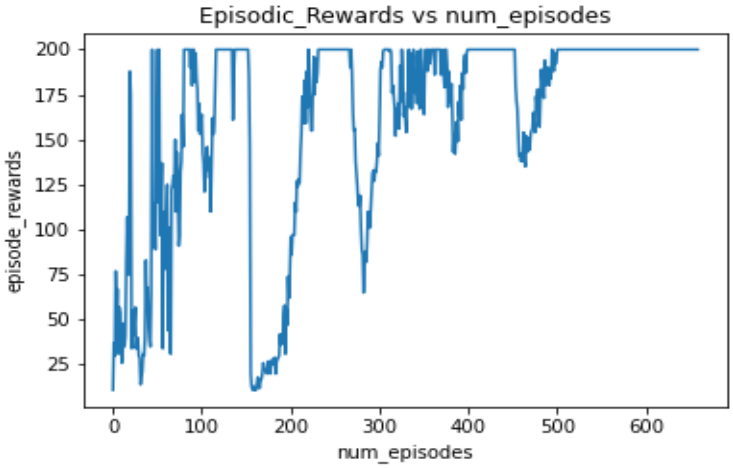
Cartpole-v1:

Settings:

Neural Net = 1 hidden layer with 128 units

gamma = 0.95, optimizer = Adam, learning rate = 1e-3



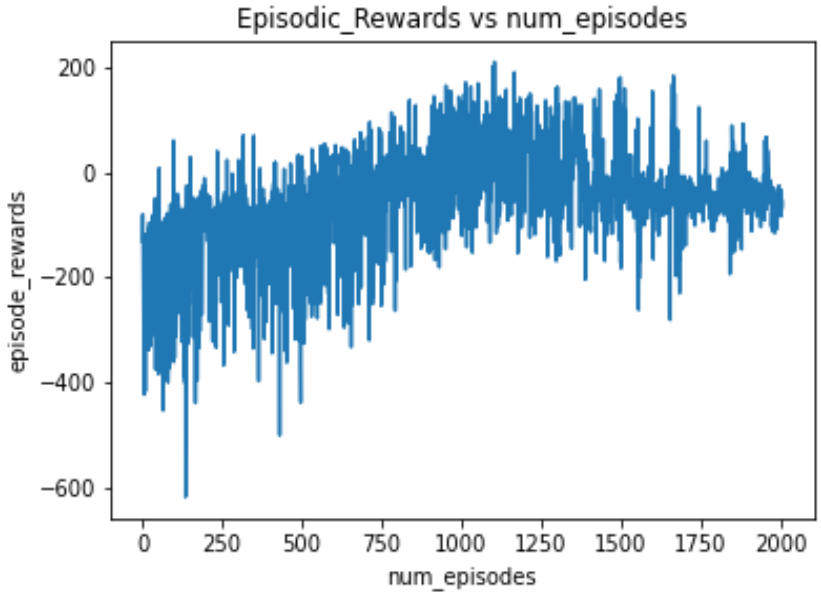


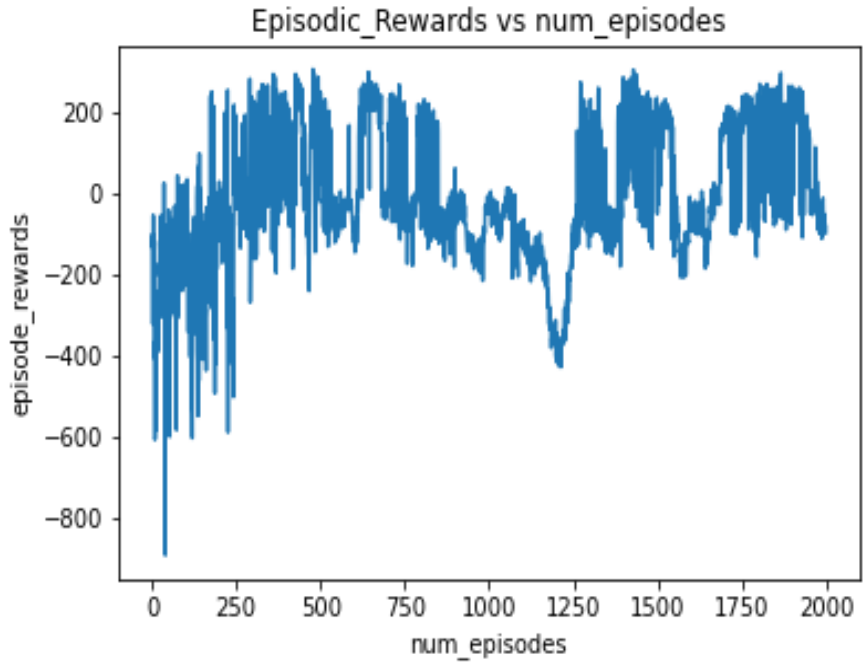
LunarLander-v2:

Settings:

Neural Net = 1 hidden layer with 256 units

gamma = 0.95, optimizer = Adam, learning rate = 1e-3





Observations:

General observations regarding tasks:

- We were not able to find the best hyper-parameters for Acrobat and Mountain Car due to which rewards failed to improve even after 20k-30k episodes.This can also be attributed (according to my observation) to sparse rewards that exist in these tasks which does not give proper indication while simulating episodes.

Architecture of policy network:

- We tried different architectures of neural networks but mostly they did not show much effect in the results and more depends on the type of layers that were chosen in between layers.
  For instance:
  RELU performed better than others for hidden layer

Baseline vs no Baseline:

- Usually adding baseline reduces variance in the results and also causes the agent to reach optimal results faster.

We also tried changing the optimizer used, Adam was the most reliable.

Cons:

Some drawbacks that I could observe while implementing REINFORCE are:

- Sample Inefficiency
  - For tasks with larger state space and more actions like Acrobat more samples are required to start showing some progress.

- Slow Convergence
  - Usually slower convergence as we need to wait for the whole episode to update our network.

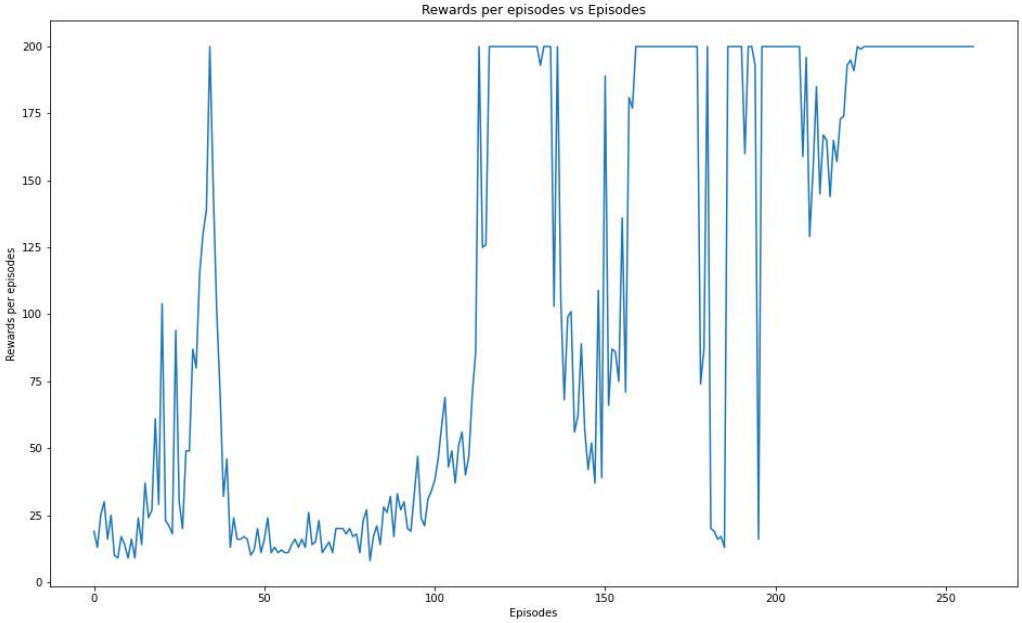
- Higher Variance
  - Can be solved to some extent by adding baseline.

Part 3:

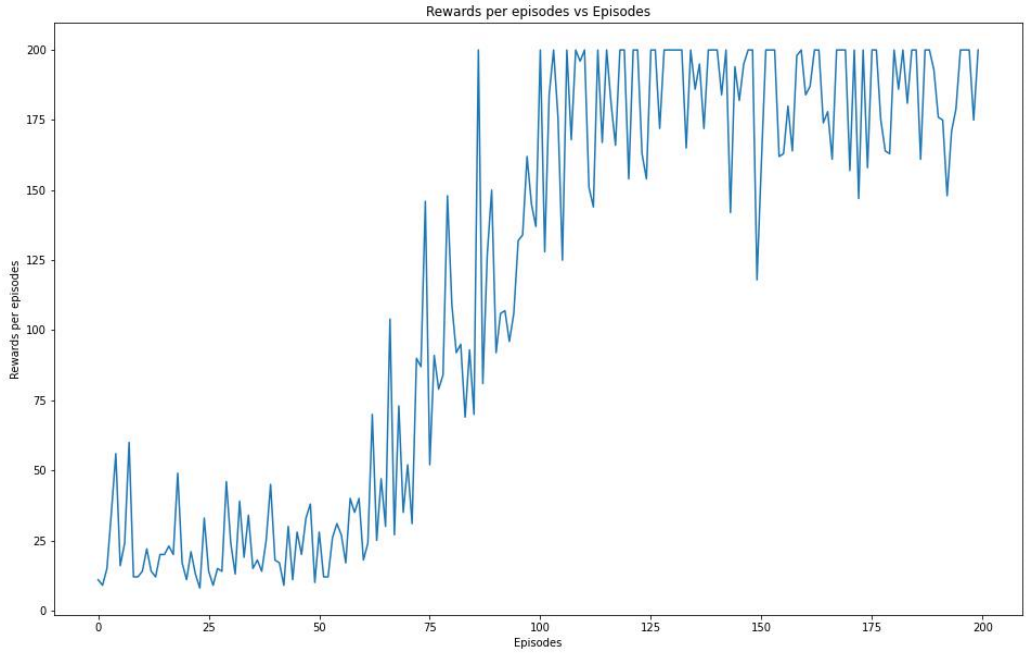
Task considered: Cartpole-v0

Results:

A2C



DQN



Settings considered:

A2C:

Actor has 1 hidden layer with 128 dimensions.

Critic has 1 hidden layer with 128 dimensions.

Loss is L1 loss

Gamma = 0.99

Learning rate =

DQN:

Network has 2 hidden layers with 1024 and 512 dimensions.

Loss is MSE loss

Gamma = 0.95

Memory buffer size = 10000

Epsilon decay rate = 0.9995

Observations from code:

- Changing the neural network layer sizes for both the algorithms did not affect the rewards too much. A larger neural network takes more time to train than a smaller one.
- Smaller learning rate for the neural network requires a larger number of episodes to converge to optimal function
- DQN gives a deterministic policy whereas A2C gives a stochastic policy
- From the plots, we can see that A2C has a higher variance compared to DQN. We can say the DQN is more sample efficient than A2C.
- Using DQN for small action spaces is better as it takes less data(more sample efficient) and exploring using eps is sufficient enough to obtain optimal results.
- A2C becomes the only choice when choosing continuous action spaces between the 2.