

Reinforcement Learning

- Aayush Arora
- EE17BTECH11003

NOTE - All the codes are written in **TensorFlow 2.0**, please ensure to install the package beforehand using
\$ python3 -m pip install tensorflow

DEEP Q NETWORKs ALGORITHM

MountainCar-v0

A Part -

The state space consist of position as well as height of the mountain car. The action consist of do nothing, move left, move right. We get -1 reward at each time frame till we reach up the mountain. The goal is to drive up the mountain on the right; 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.

To observe the environment, run the following command.

\$ python3 mountaincar.py

B Part -

Hyper Parameters -

buffer size – 10000

gamma – 0.99

epsilon – 1

epsilon_min – 0.005

batch size – 64

learning rate – 0.001

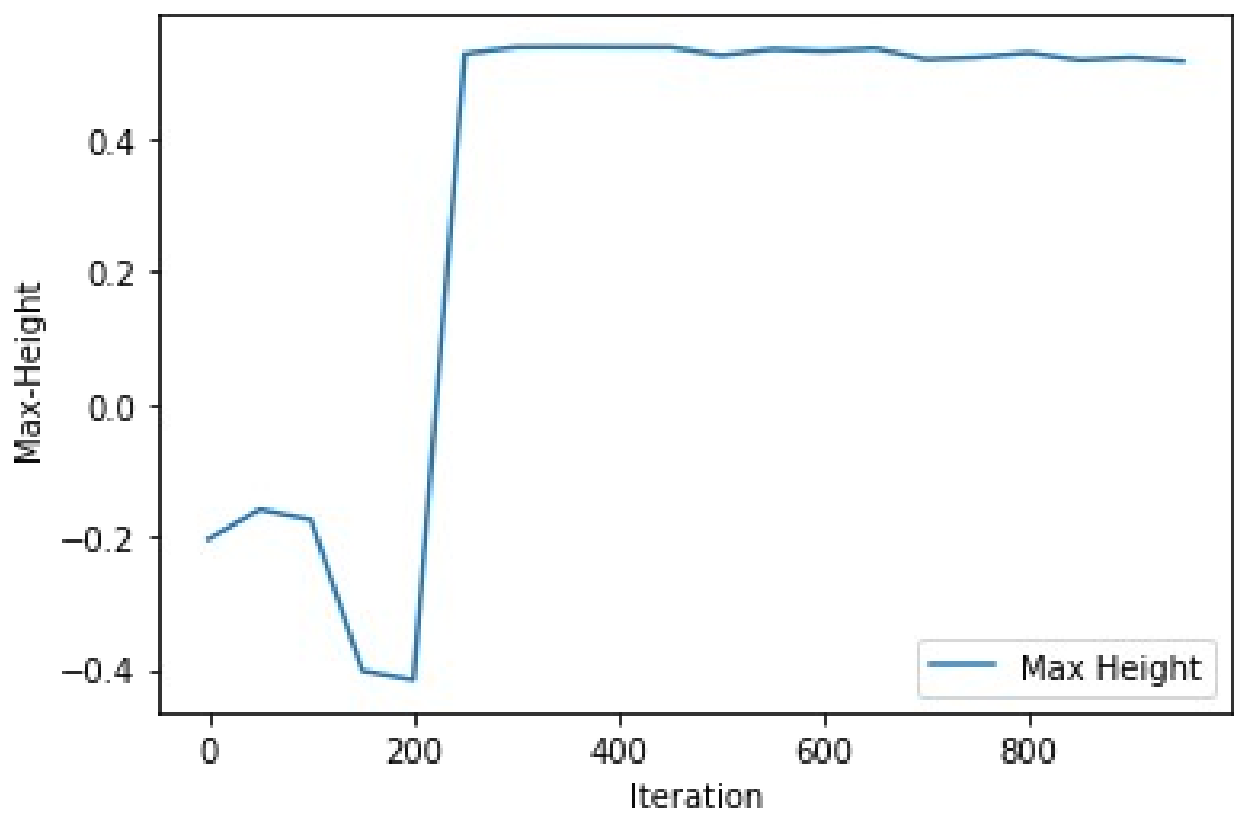
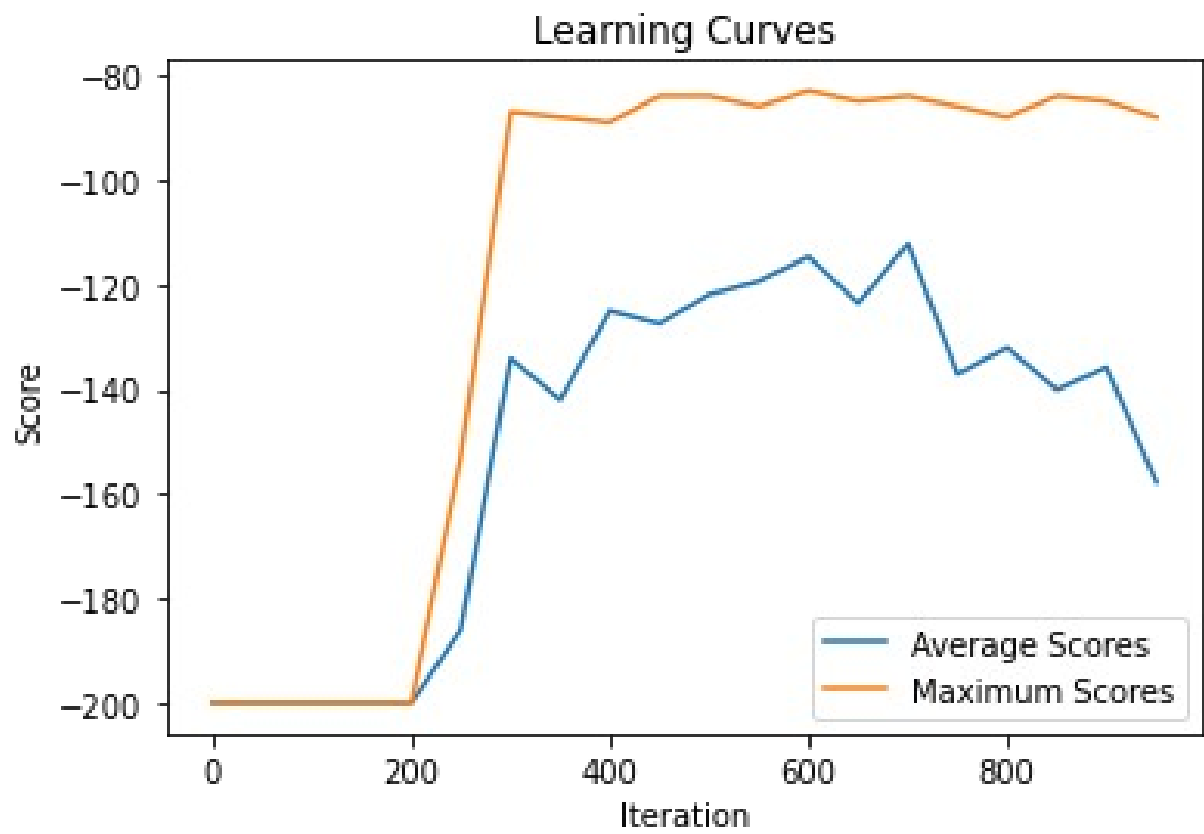
Learning Curves: -

The testing is done after every 50 iterations for 10 episodes.

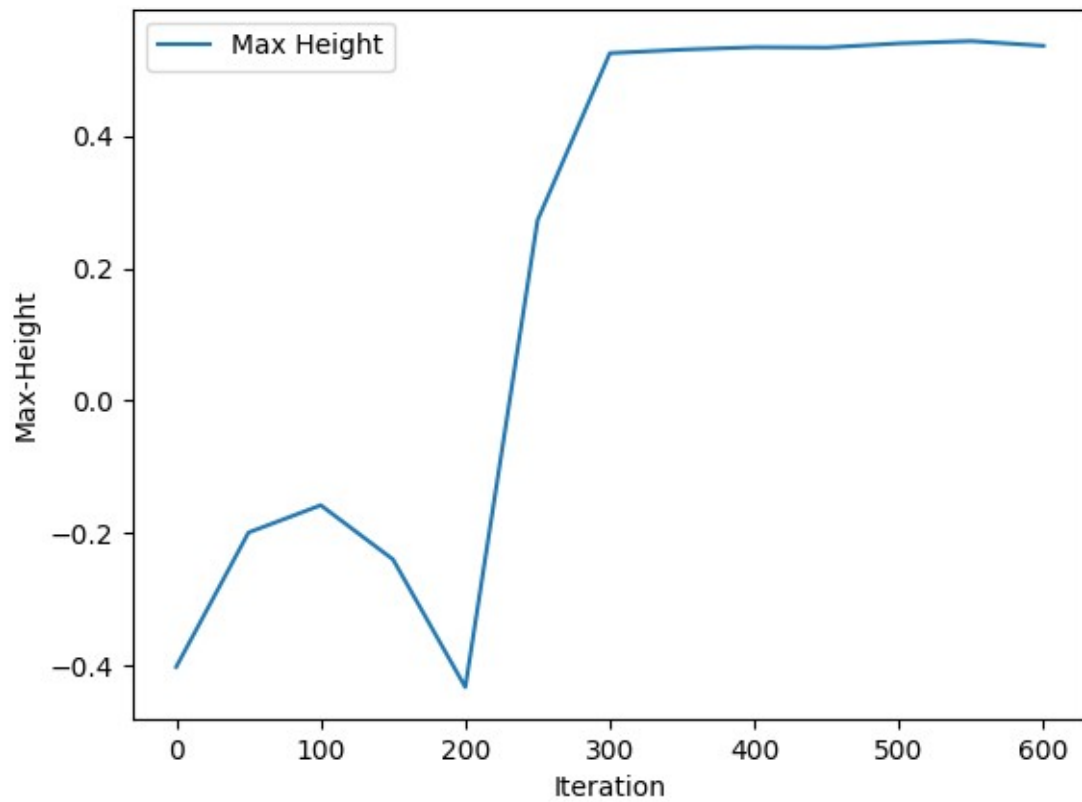
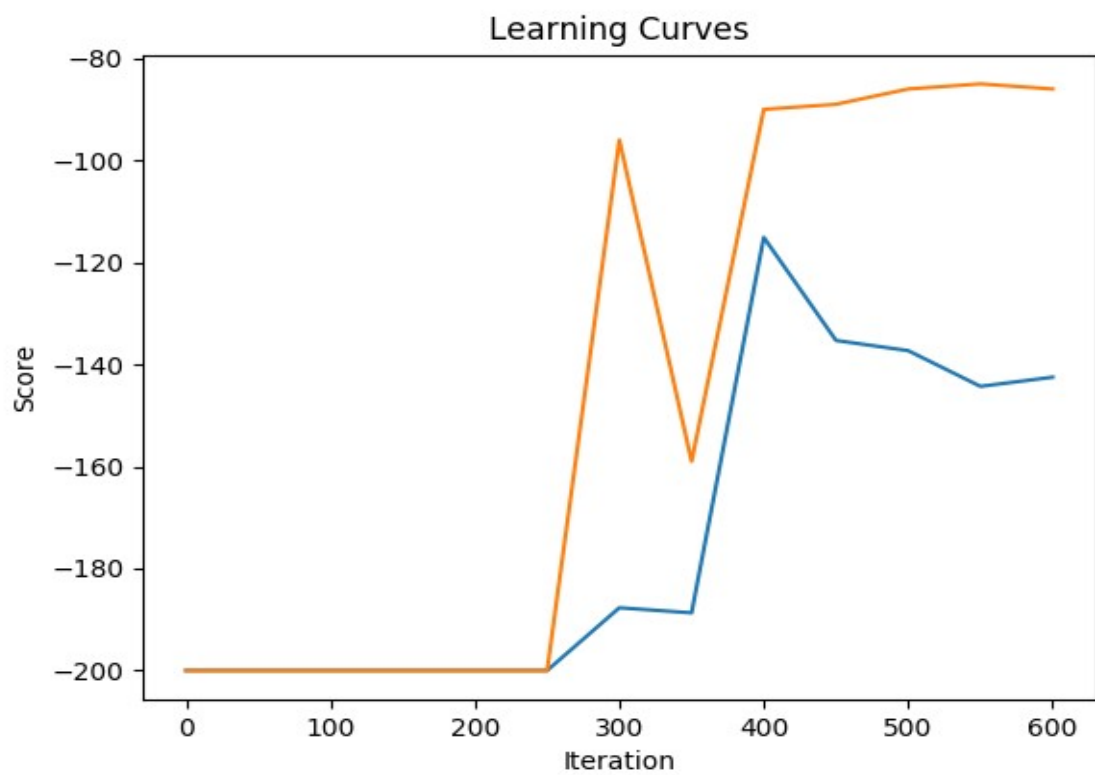
The First graph depicts the Average Scores and Maximum Scores while training.

The second graph depicts the Max Height reached which increases with iterations.

1st Trial



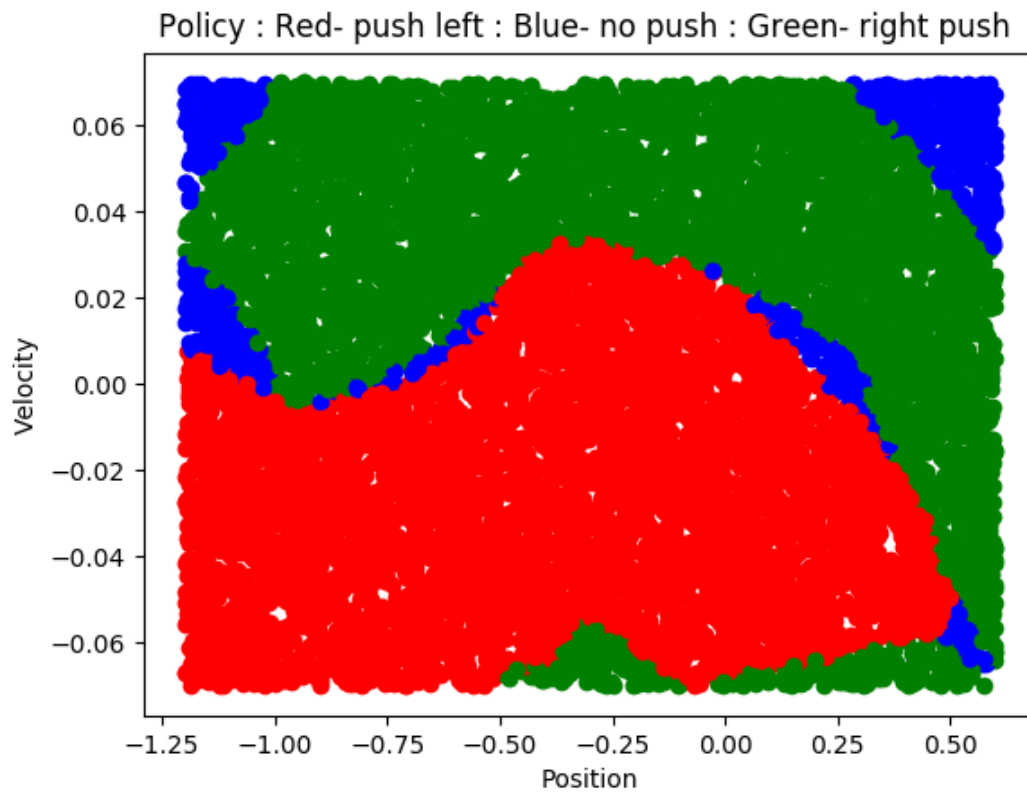
2nd Trial



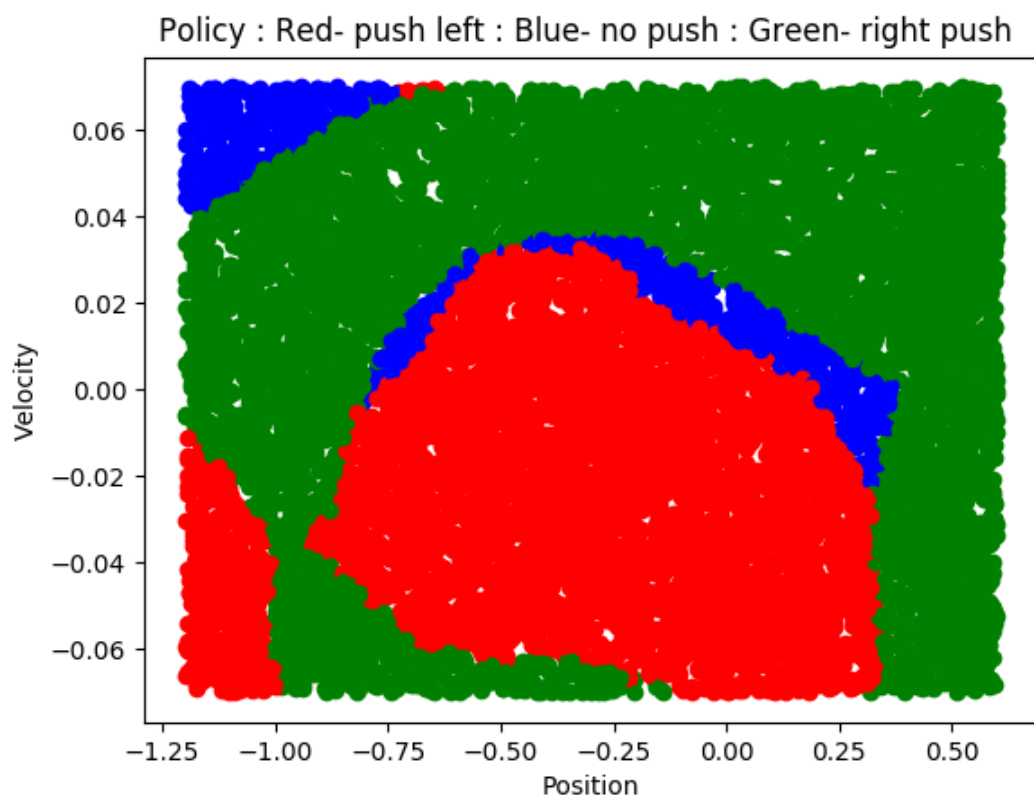
Policy: -

This explains the choice of action for particular states.

1st Trial



2nd Trial



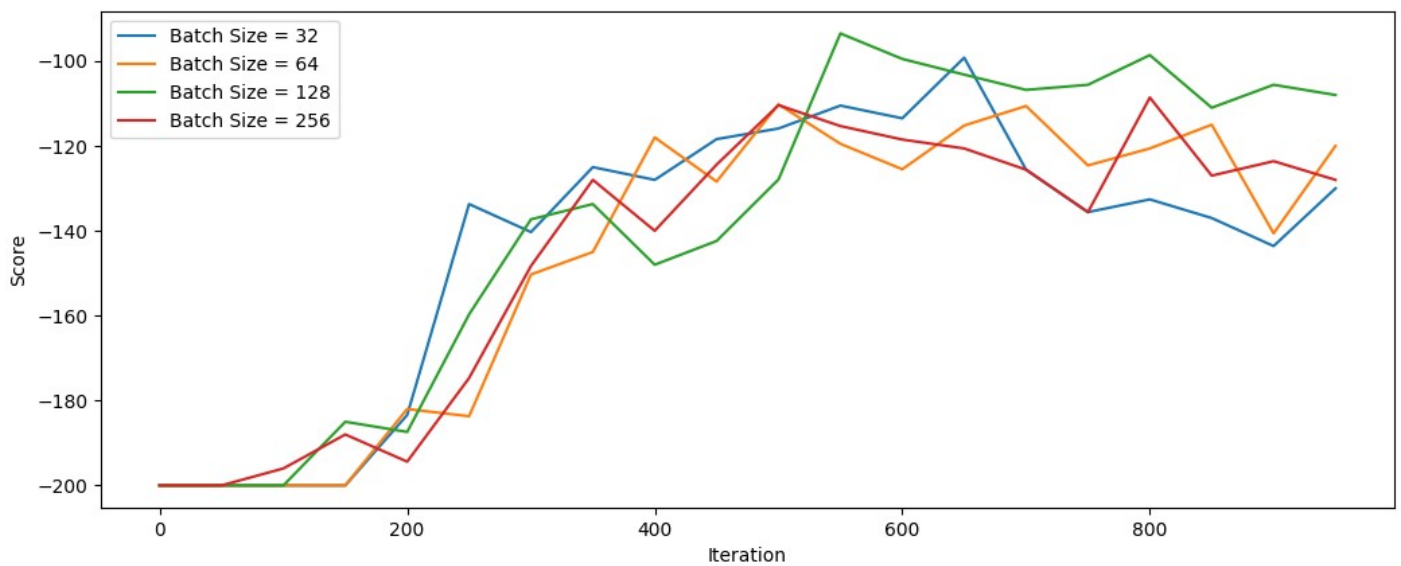
To train the agent for MountainCar environment, run the following command.

```
$ python3 mountaincar_train.py
```

To test the agent for MountainCar environment using trained_model, run the following command.

```
$ python3 mountaincar_test.py
```

Variation in Batch Size -



POLICY GRADIENT ALGORITHM

LunarLander-v2

A Part -

The state space has 8 components including horizontal and vertical position, horizontal and vertical velocity, angle and angular velocity, and left and right leg contact. The action space includes do nothing, fire main engine, fire left engine and fire right engine. Reward for moving from the top of the screen to landing pad and zero speed is about 100..140 points. If lander moves away from landing pad it loses reward back. Episode finishes if the lander crashes or comes to rest, receiving additional -100 or +100 points. Each leg ground contact is +10. Firing main engine is -0.3 points each frame. Solved is 200 points. Landing outside landing pad is possible. Fuel is infinite, so an agent can learn to fly and then land on its first attempt.

To observe the environment, run the following command.

```
$ python3 lunarlander.py
```

B Part -

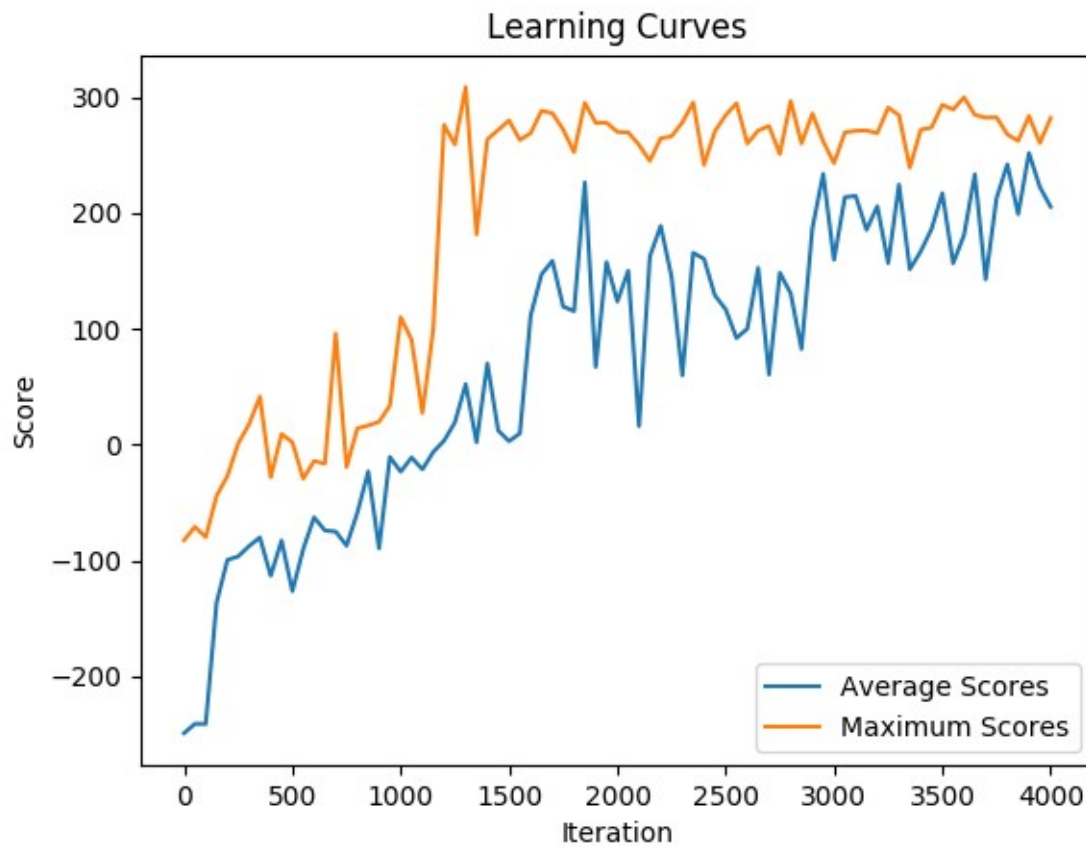
Learning curves for LunarLander Environment with defined functionalities: -

batch size – 20

discount_factor – 0.99

learning_rate – 0.001

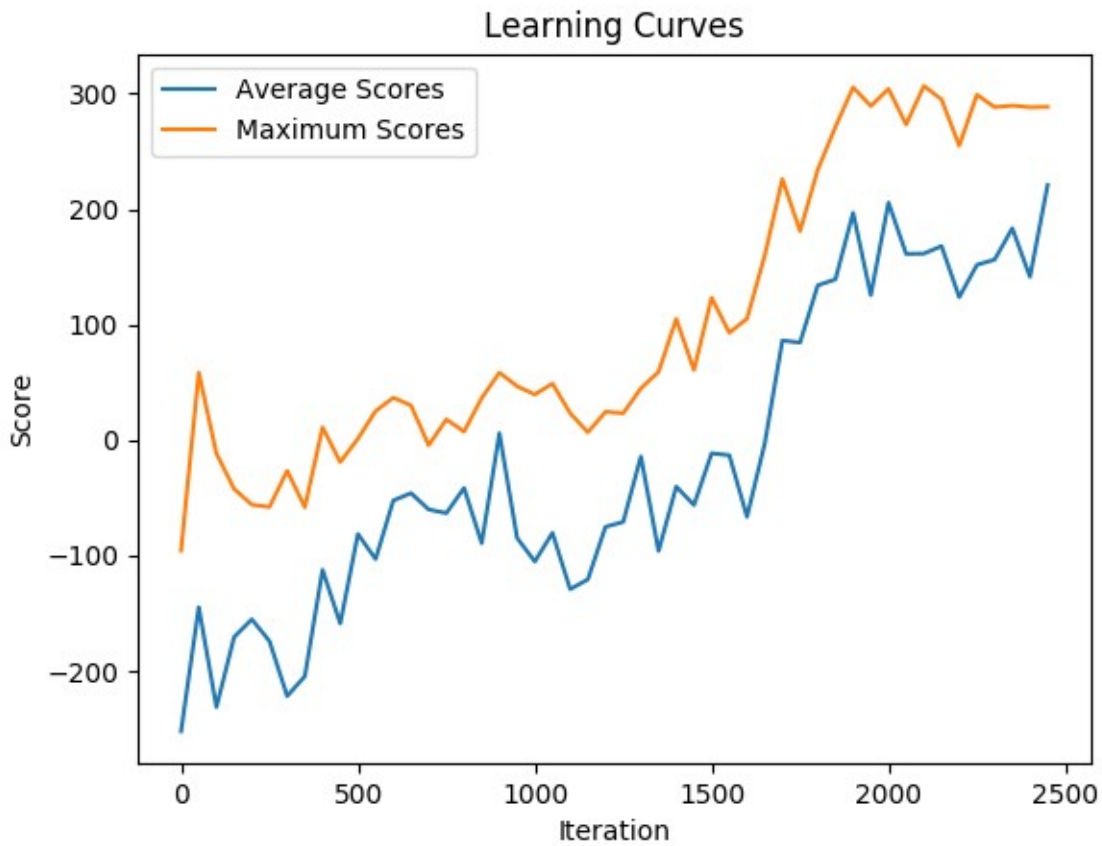
- ***With Reward To Go and Advantage Normalization functionality***



To observe the training, run the following command.

```
$ python3 Policy_Gradient.py --env LunarLander-v2 --batch_size 20 --reward_to_go  
--advantage_normalization
```

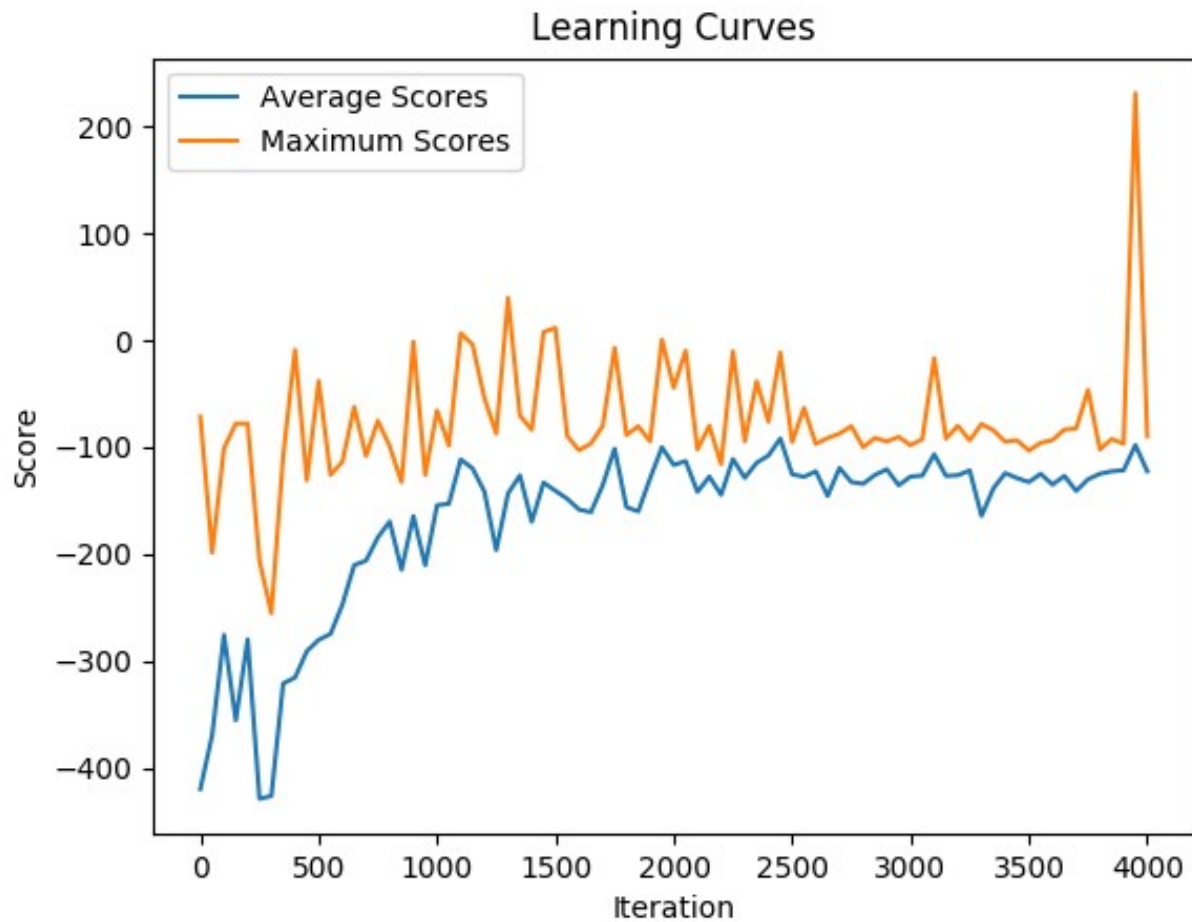
- ***With Reward To Go and without Advantage Normalization functionality***



To observe the training, run the following command.

```
$ python3 Policy_Gradient.py --env LunarLander-v2 --batch_size 20 --reward_to_go
```

- *Without Reward To Go and Advantage Normalization functionality*



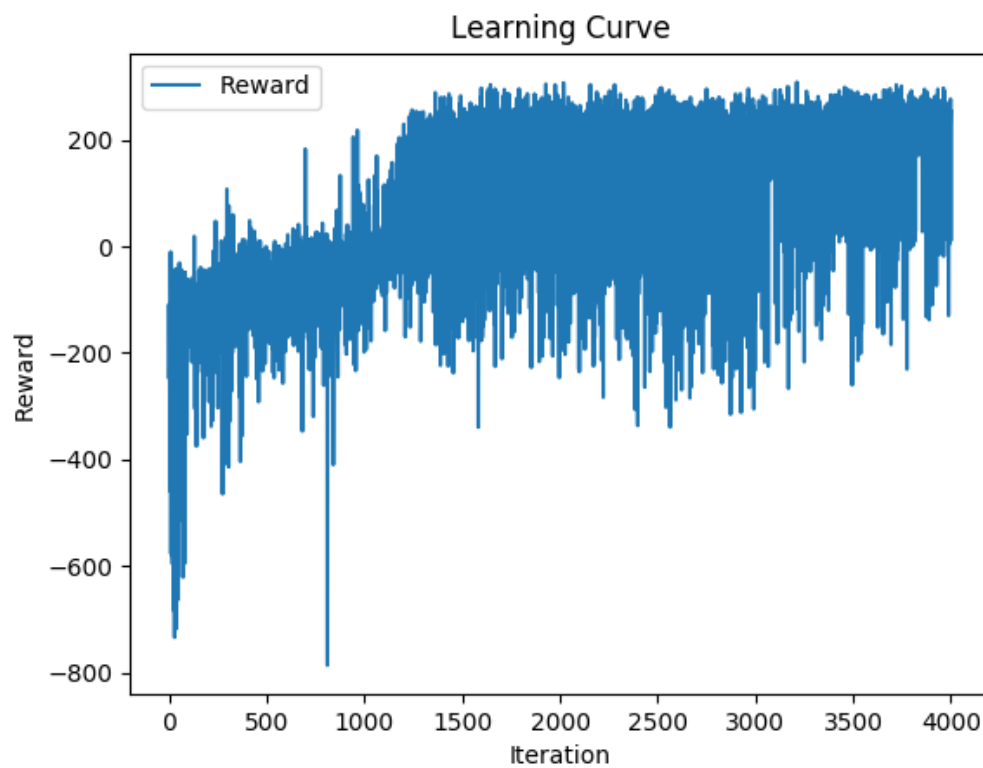
To observe the training, run the following command.

```
$ python3 Policy_Gradient.py --env LunarLander-v2 --batch_size 20
```

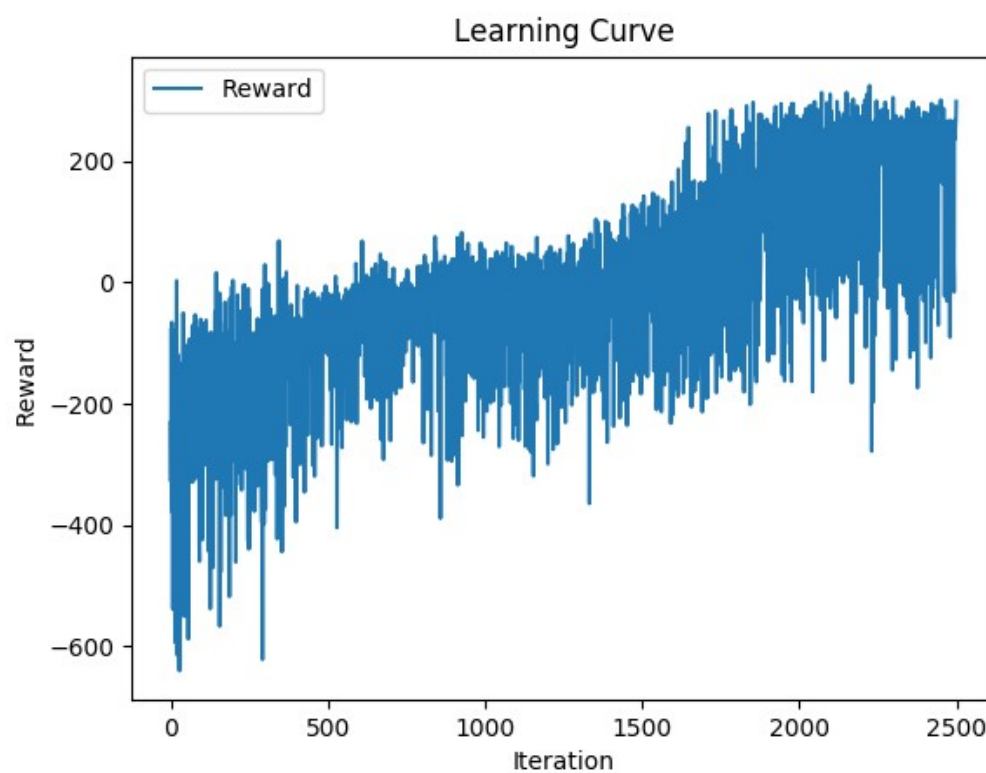
C Part -

Variation in training curves due to different batch sizes.

Batch Size – 20



Batch Size – 32



Cartpole-v0

A Part -

The pendulum starts upright, and the goal is to prevent it from falling over. The actions include applying force of +1 and -1 to the cart. A reward of +1 is provided for every timestep that the pole remains upright. The episode ends when the pole is more than 15 degrees from vertical, or the cart moves more than 2.4 units from the center. The state space consist of 4 values corresponding to the position of pole and the cart.

To observe the environment, run the following command.

```
$ python3 cartpole.py
```

B Part -

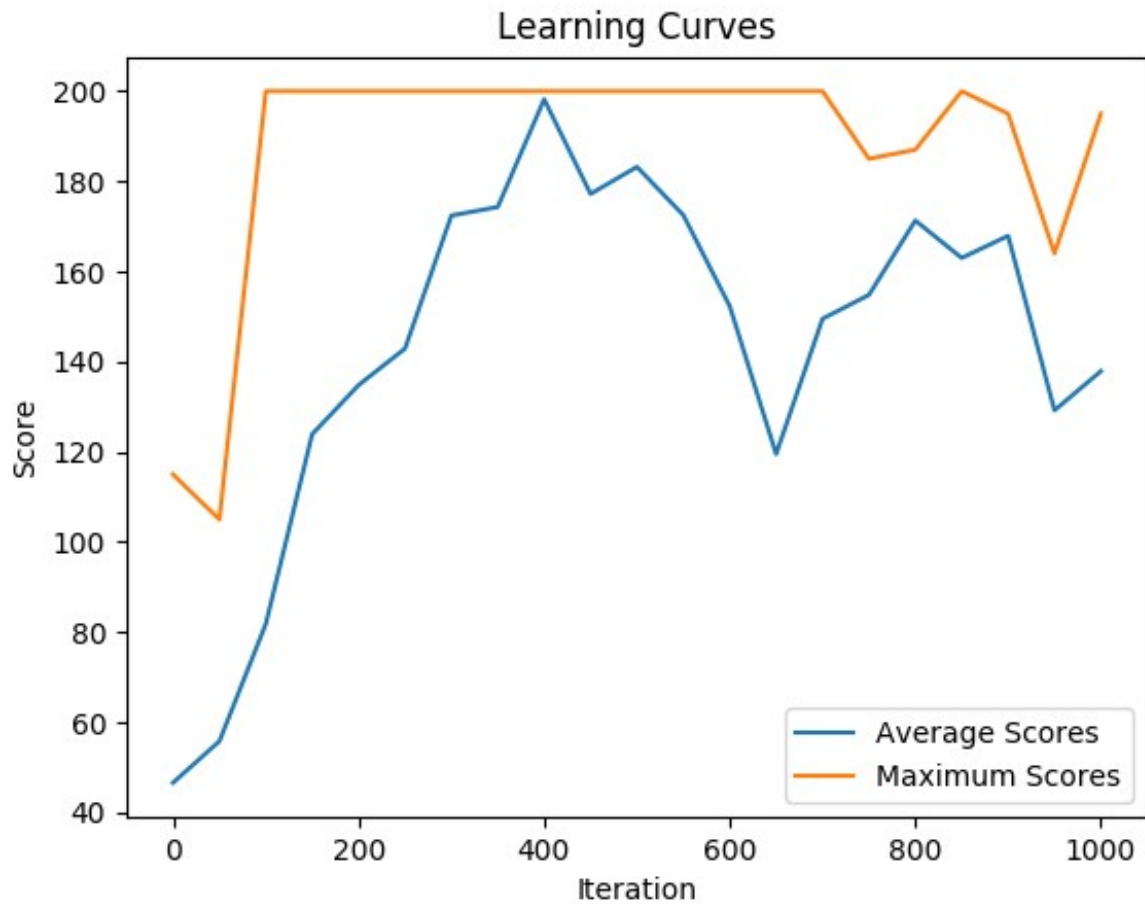
Learning curves for Cartpole Environment with defined functionalities: -

batch size – 20

discount_factor – 0.99

learning_rate – 0.001

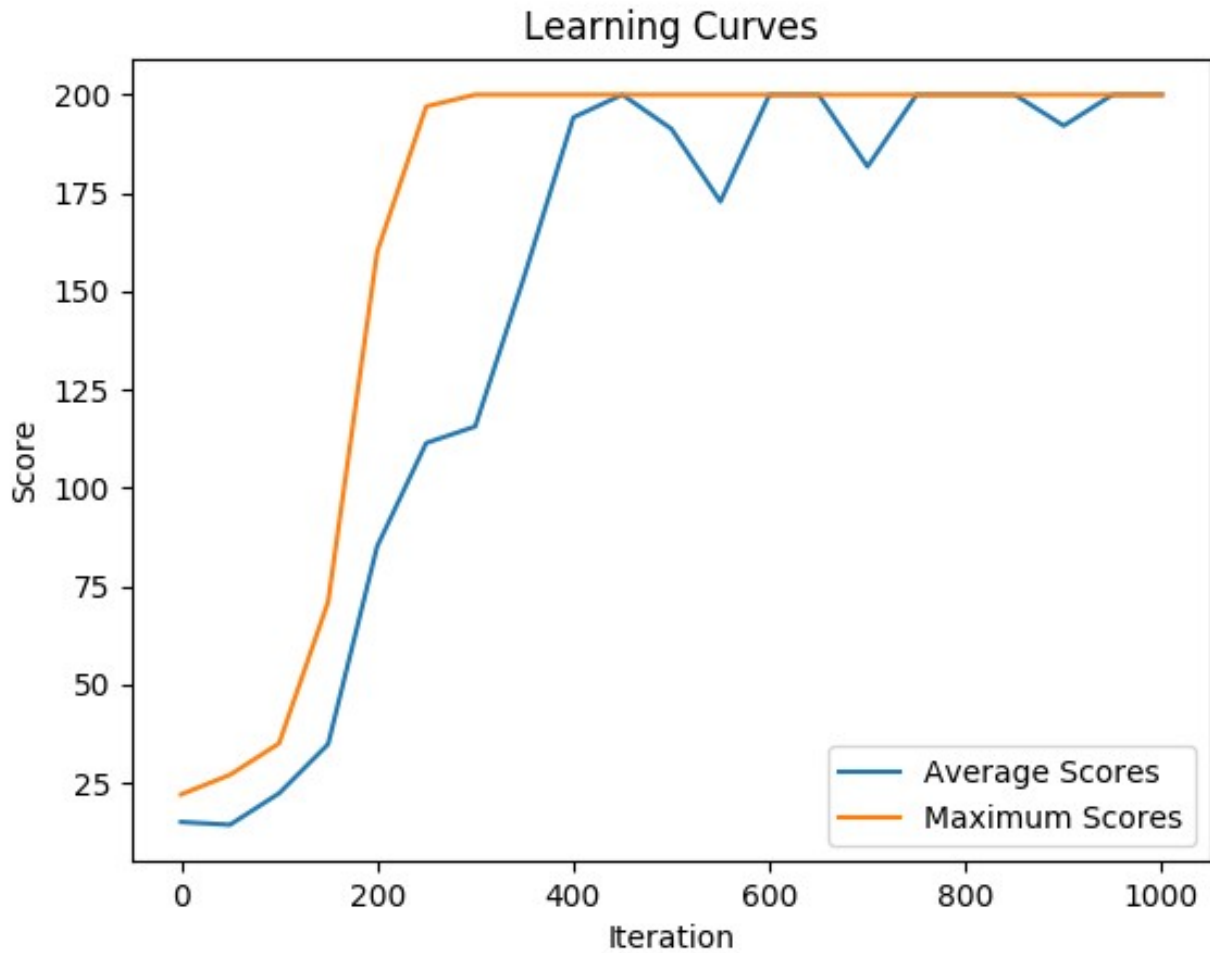
- *With Reward To Go and Advantage Normalization functionality*



To observe the training, run the following command.

```
$ python3 Policy_Gradient.py --env CartPole-v0 --batch_size 20 --reward_to_go --advantage_normalization
```

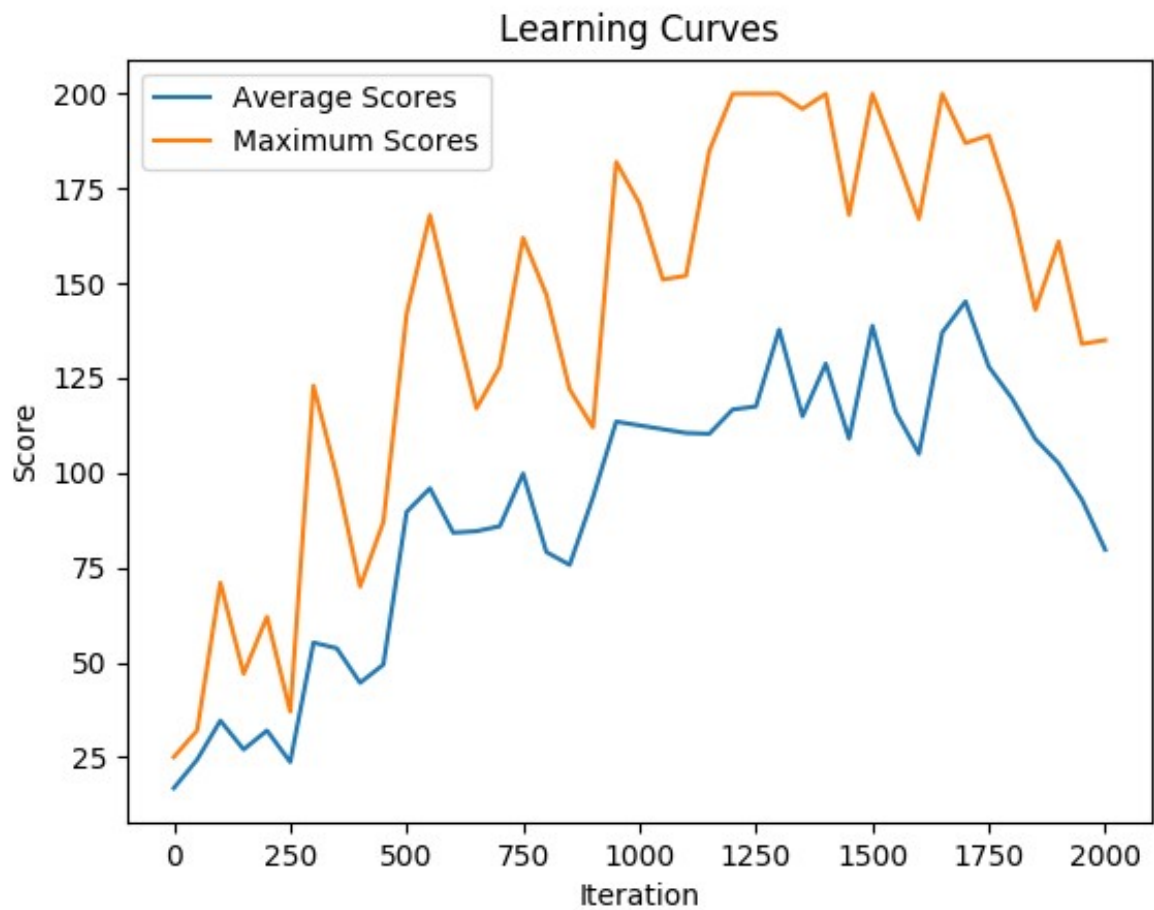
- *With Reward To Go and without Advantage Normalization functionality*



To observe the training, run the following command.

```
$ python3 Policy_Gradient.py --env CartPole-v0 --batch_size 20 --reward_to_go
```

- *Without Reward To Go and Advantage Normalization functionality*



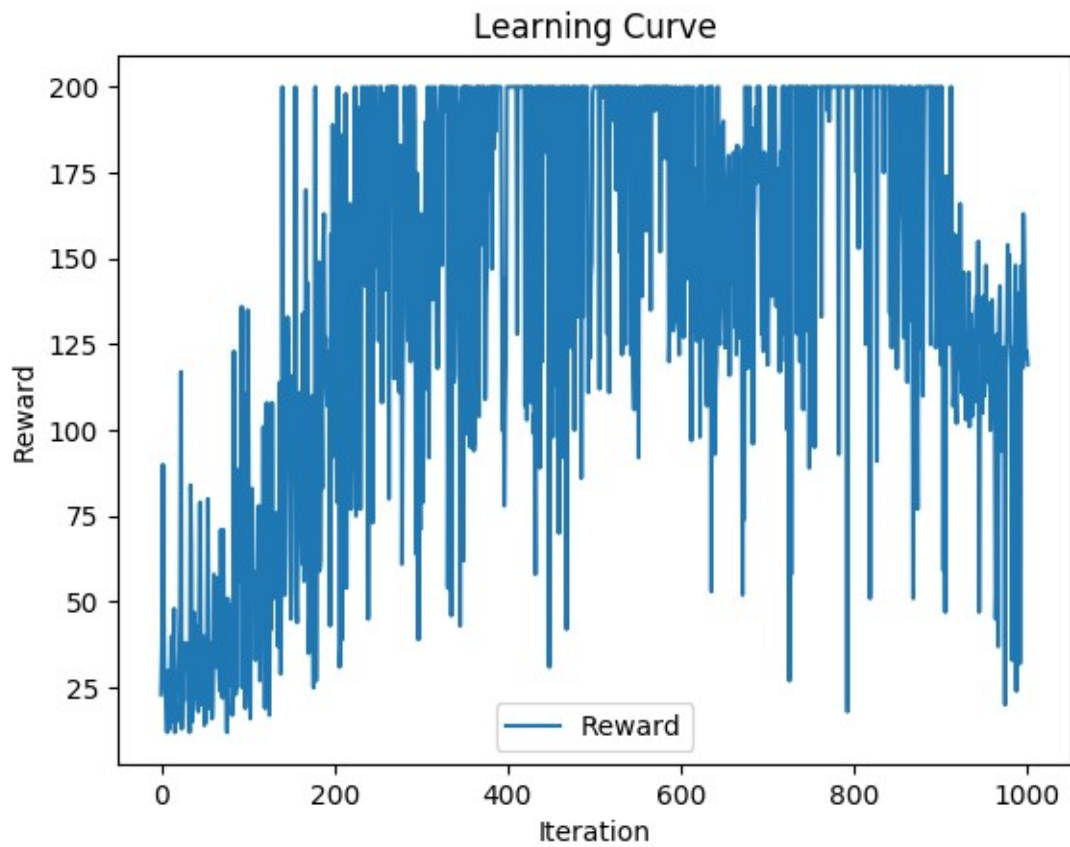
To observe the training, run the following command.

```
$ python3 Policy_Gradient.py --env CartPole-v0 --batch_size 20
```

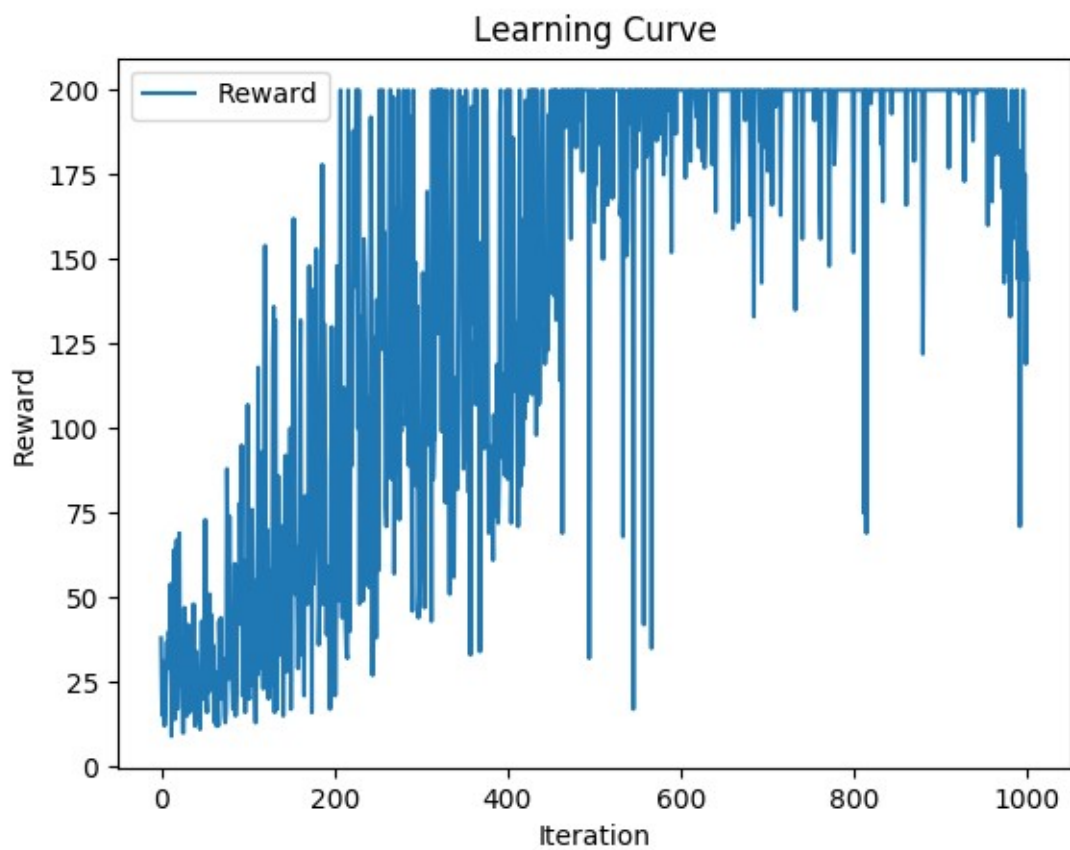
C Part -

Variation in training curves due to different batch sizes.

Batch Size – 10



Batch Size – 20



Batch Size – 50

