UDACITY

‹ Return to Classroom

Collaboration and Competition

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Dear student, great job. The performance of the agent is very good and the project meets the specifications. 😄

The recent application of deep reinforcement learning to play the game Dota 2 (https://openai.com/blog/dota-2/) is a substantial step.

Also, to know more about reinforcement learning algorithms and applying these to real world problems, please do check
Deep Reinforcement Learning (https://www.youtube.com/watch?v=MQ6pP65o7OM) which is a part of the course MIT 6.S094: Deep Learning for Self-Driving Cars (2018 version).

Also, take a look at the following links which are good starting points and also provide an understanding in laymen terms

- Multi-Agent Deep Reinforcement Learning in 13 Lines of Code Using PettingZoo https://towardsdatascience.com/multi-agent-deep-reinforcement-learning-in-15-lines-of-code-using-pettingzoo-e0b963c0820b
- Multi-Agent Reinforcement Learning (MARL) and Cooperative AI https://towardsdatascience.com/ive-been-thinking-about-multi-agent-reinforcement-learning-marl-and-you-probably-should-be-too-8f1e241606ac
- Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms https://arxiv.org/pdf/1911.10635.pdf
- Multi-Agent Reinforcement Learning https://medium.com/@RemiStudios/multi-agent-reinforcement-learning-3f00b561f5f0
- Scalable and Robust Multi-Agent Reinforcement Learning https://www.youtube.com/watch?v=Yd6HNZnqjis

Happy learning ✌️

Training Code

✓

The repository includes functional, well-documented, and organized code for training the agent.

Great work. All the files are present and are in order. Nicely done. Take a look at the following links for a better understanding of the algorithms for continuous control

- https://towardsdatascience.com/openais-multi-agent-deep-deterministic-policy-gradients-maddpg-

9d2dad34c82
- https://keras.io/examples/rl/ddpg_pendulum/ to understand the implementation of ddpg in the keras
- https://arxiv.org/abs/1509.02971 research paper for indepth understanding

✓

The code is written in PyTorch and Python 3.

Good job completing the project using Pytorch and Python3.

You should definitely check this post comparing different deep learning frameworks: Deep Learning Frameworks Comparison – Tensorflow, PyTorch, Keras, MXNet, The Microsoft Cognitive Toolkit, Caffe, Deeplearning4j, Chainer (https://www.netguru.com/blog/deep-learning-frameworks-comparison)

✓

The submission includes the saved model weights of the successful agent.

Great work. The submission includes the saved model weights of the successful agent.

README

✓

The GitHub submission includes a `README.md` file in the root of the repository.

Great work here. Very well done. A detailed README file has been provided and is present in the repository.

Take a look at the following links to improve how your README looks

- https://blog.bitsrc.io/how-to-write-beautiful-and-meaningful-readme-md-for-your-next-project-897045e3f991
- https://dev.to/scottydocs/how-to-write-a-kickass-readme-5af9

✓

The README describes the the project environment details (i.e., the state and action spaces, and when the environment is considered solved).

Great work here

✓

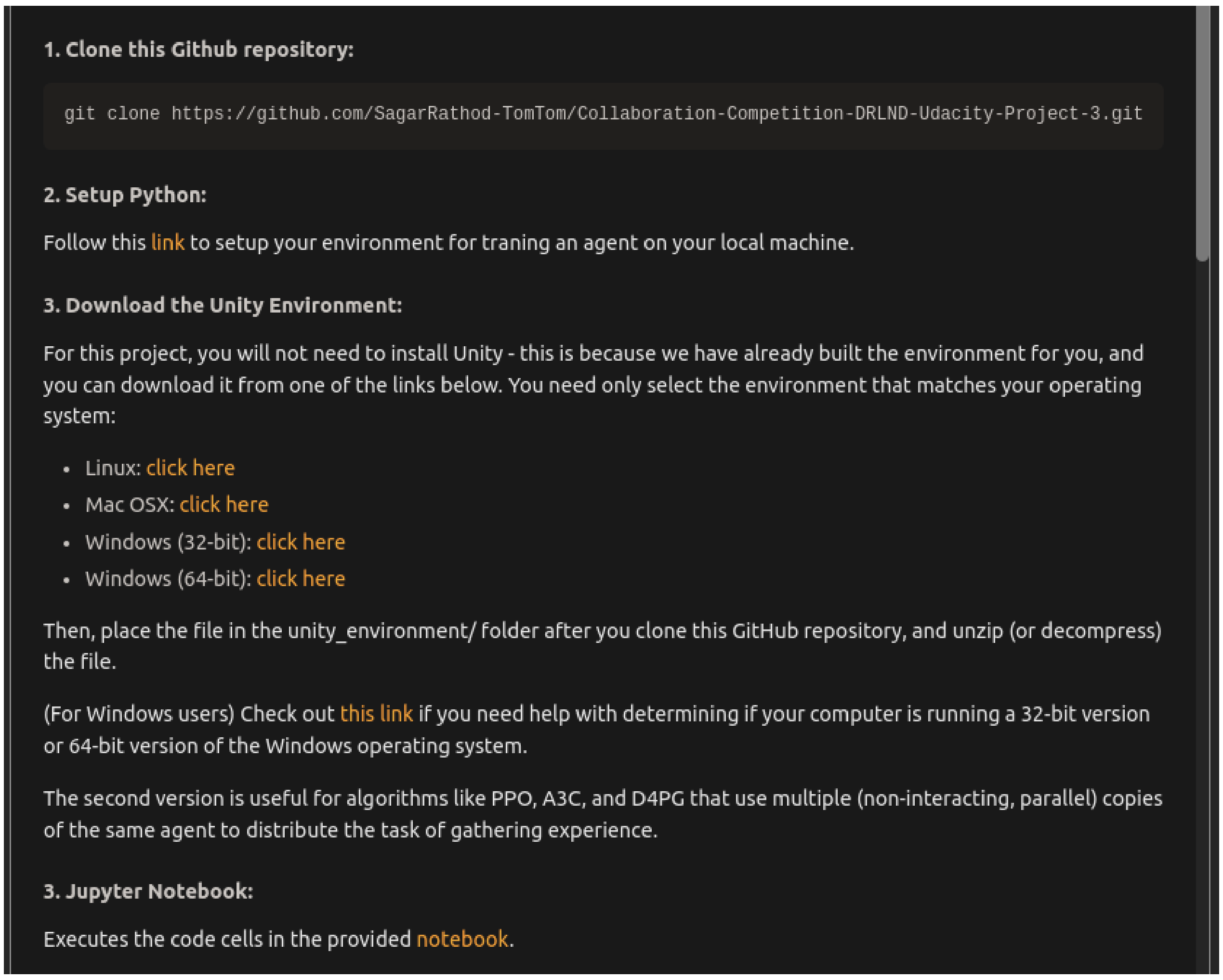
The README has instructions for installing dependencies or downloading needed files.

Great work. The README talks about on how to install the dependencies and how to run the code. All the requirements are met. Very nicely done

✓

The README describes how to run the code in the repository, to train the agent. For additional resources on creating READMEs or using Markdown, see here and here.

Brilliant work. You have explained how to run the code and train the agent. Very well done. Keep up the good work



1. Clone this Github repository:

```
git clone https://github.com/SagarRathod-TomTom/Collaboration-Competition-DRLND-Udacity-Project-3.git
```

2. Setup Python:

Follow this link to setup your environment for traning an agent on your local machine.

3. Download the Unity Environment:

For this project, you will not need to install Unity - this is because we have already built the environment for you, and you can download it from one of the links below. You need only select the environment that matches your operating system:

- Linux: click here
- Mac OSX: click here
- Windows (32-bit): click here
- Windows (64-bit): click here

Then, place the file in the unity_environment/ folder after you clone this GitHub repository, and unzip (or decompress) the file.

(For Windows users) Check out this link if you need help with determining if your computer is running a 32-bit version or 64-bit version of the Windows operating system.

The second version is useful for algorithms like PPO, A3C, and D4PG that use multiple (non-interacting, parallel) copies of the same agent to distribute the task of gathering experience.

3. Jupyter Notebook:

Executes the code cells in the provided notebook.

Report

✓

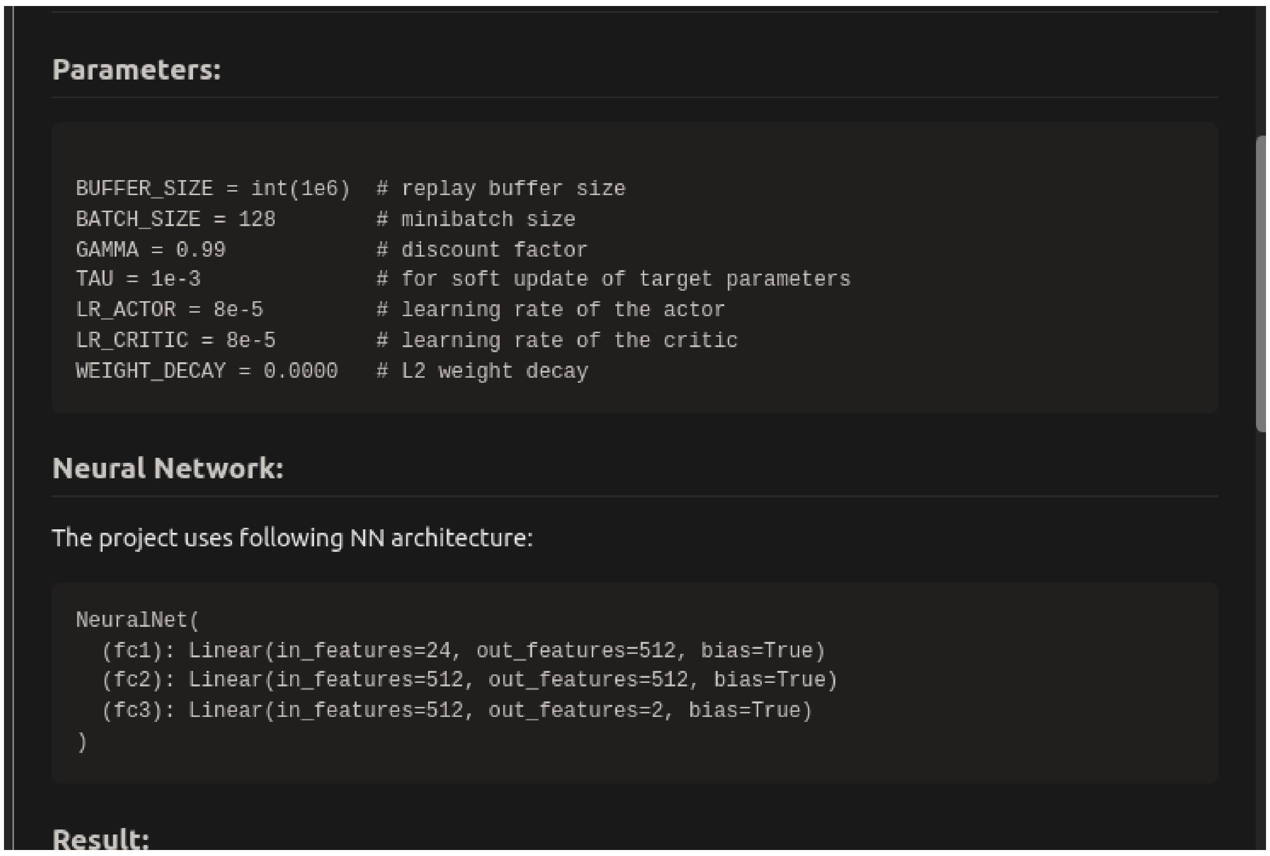
The submission includes a file in the root of the GitHub repository (one of `Report.md`, `Report.ipynb`, or `Report.pdf`) that provides a description of the implementation.

Nice work. Report has been included in the root of the github repository.

✓

The report clearly describes the learning algorithm, along with the chosen hyperparameters. It also describes the model architectures for any neural networks.

Description of the learning algorithm, hyperparameters, and network architecture have been provided in great detail in the project report



Parameters:

```
BUFFER_SIZE = int(1e6)  # replay buffer size
BATCH_SIZE = 128        # minibatch size
GAMMA = 0.99            # discount factor
TAU = 1e-3              # for soft update of target parameters
LR_ACTOR = 8e-5         # learning rate of the actor
LR_CRITIC = 8e-5        # learning rate of the critic
WEIGHT_DECAY = 0.0000   # L2 weight decay
```

Neural Network:

The project uses following NN architecture:

```
NeuralNet(
  (fc1): Linear(in_features=24, out_features=512, bias=True)
  (fc2): Linear(in_features=512, out_features=512, bias=True)
  (fc3): Linear(in_features=512, out_features=2, bias=True)
)
```

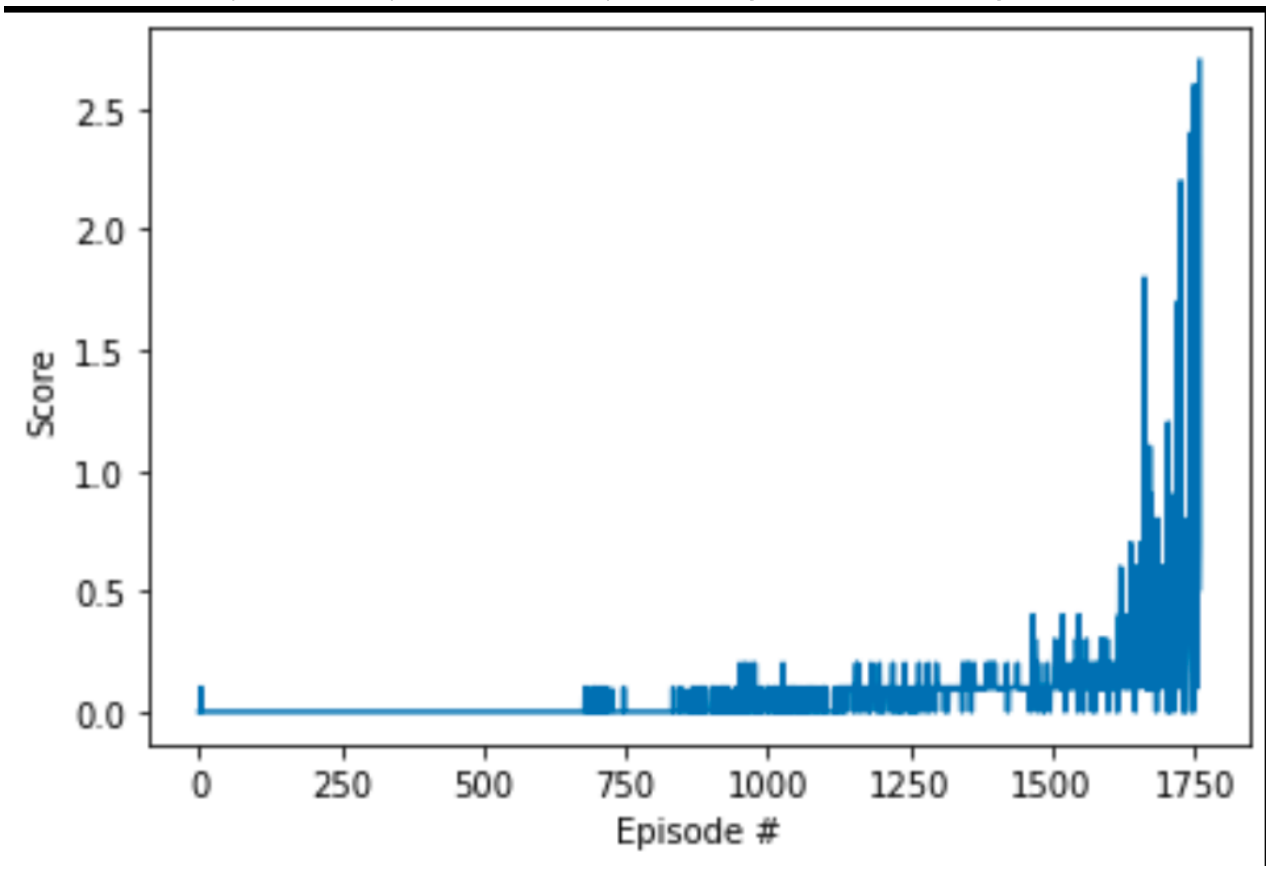
Result:

✓

A plot of rewards per episode is included to illustrate that the agents get an average score of +0.5 (over 100 consecutive episodes, after taking the maximum over both agents).

The submission reports the number of episodes needed to solve the environment.

Nice work. Rewards plot has been provided for the implemented agent and seems to be great.



✓

The submission has concrete future ideas for improving the agent's performance.

Great intuition.

Prioritized experience replay will be helpful. D4PG(https://arxiv.org/abs/1804.08617v1), is a great area to start.
Also, take a look at the following link to understand more algorithms in play
https://mayankm96.github.io/tutorials/

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review

START