

[Return to Classroom](#)[DISCUSS ON STUDENT HUB](#)

Continuous Control

[REVIEW](#)[CODE REVIEW](#)[HISTORY](#)

Meets Specifications

Dear Udacian,

Great job getting acquainted with the Deep Deterministic Policy Gradients algorithm and successfully implementing it to complete the project. Thank you for updating the README file with the instructions to install the dependencies. The submission is complete. 😊

I would suggest you to go through [Deep Reinforcement Learning for Self Driving Car by MIT](#). You'd get to know more about reinforcement learning algorithms in broader and real-world perspective and, more importantly, how to apply these techniques to real-world problems.

All the best for future endeavors. ✨

Training Code

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

Average

Awesome

- Good work implementing DDPG algorithm to solve robotic-arms Reacher environment.
- Implementation of the Actor and Critic networks is correct.
- Good work using the target networks for Actor and Critic networks. The original DDPG paper suggests it as well.
- Good work using soft updates for the target network.
- Good choice to use tau to perform soft update.
- Correct usage of replay memory to store and recall experience tuples.
- The implementation is easy to debug and easily extensible, good work keeping it highly modular.

The code is written in PyTorch and Python 3.

Awesome

The code is written in PyTorch and Python 3.

Lately, PyTorch and TensorFlow happen to be most extensively used frameworks in deep learning. It would be good to get some insight by comparing them, please see the following resources:

- [Sebastian Thrun on TensorFlow](#)
- [PyTorch vs TensorFlow—spotting the difference](#)
- [Tensorflow or PyTorch : The Force is Strong with which One?](#)

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

Awesome

- Saved model weights of the successful agent have been submitted.

README

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

Awesome

- Great work documenting the project details and submitting the README file.

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

Awesome

- Readme file describes the project environment details properly.
- Information about the state and action spaces, and when the environment is considered solved has been provided.

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

Awesome

- Great work providing the all the necessary installation instructions.

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](#).

Awesome

- Great work providing necessary instructions to run the code.

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.

Awesome

- Report for the project with all the details of the implementation has been provided in the submission.

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

Awesome

Great work providing the details of the implemented agent. Details of the learning algorithm used, hyperparameters, and architectural information of the deep learning model have been provided.

- Good decision to choose DDPG algorithm for the continuous action space problem.
- Good work including model architecture in the report.
- Good decision choosing to use batch normalization.
- Hyperparameters you have used seem to be good.

Suggestions

To experiment more with the architecture and hyperparameters, you can check the following resources:

- [Deep Deterministic Policy Gradients in TensorFlow](#)
- [Continuous control with Deep Reinforcement Learning](#)

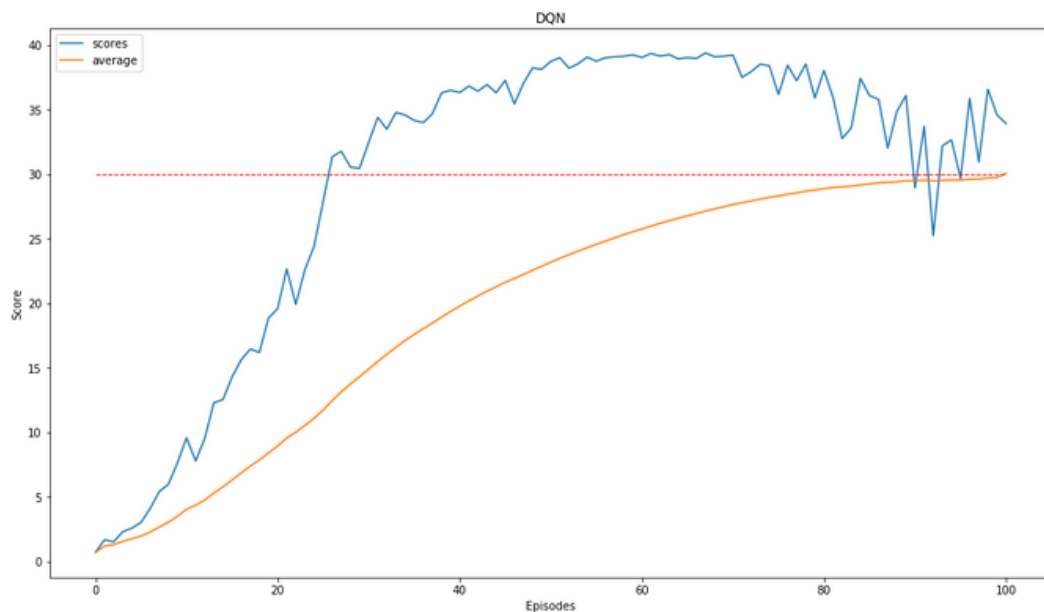
A plot of rewards per episode is included to illustrate that either:

- *[version 1]* the agent receives an average reward (over 100 episodes) of at least +30, or
- *[version 2]* the agent is able to receive an average reward (over 100 episodes, and over all 20 agents) of at least +30.

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

Awesome

- Discussion for the rewards is provided in the report.
- The rewards plot seems to be good and average score of 30 is achieved in 101 episodes.



Reinforcement learning algorithms are really hard to make work.

But it is substantial to put efforts in reinforcement learning as it is close to Artificial General Intelligence.

This article is a must read: [Deep Reinforcement Learning Doesn't Work Yet](#).

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

Awesome

- Thanks for providing the ideas for improvement.
- Definitely give a try to multiple brain's per agent arm. The idea looks very interesting and new.
- Please check the following resources also:
 - [Prioritized Experience Replay](#)
 - [Distributed Prioritized Experience Replay](#)
 - [Reinforcement Learning with Prediction-Based Rewards](#)
 - [Proximal Policy Optimization](#)
 - [OpenAI Five](#)
 - [Curiosity-driven Exploration by Self-supervised Prediction](#)

[↓ DOWNLOAD PROJECT](#)

RETURN TO PATH

Rate this review

START