UDACITY

‹ Return to Classroom

Navigation

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Dear student, great job implementing the Deep Q Network algorithm to solve the banana environment. Good work implementing the model architecture with two hidden layers with 64 units each and `relu` activation. Performance of the agent is very good and an average score of +13 is achieved in 600 episodes.Furthermore, the maximum average score achieved by the agent is +16.03! The report is rather informative and covers all the aspects of the implementation. Project meets all specifications. 😄

The recent application of deep reinforcement learning to play the game *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 which is a part of the course *MIT 6.S094: Deep Learning for Self-Driving Cars (2018 version)*.

Happy learning ✌🏻

Training Code

The repository (or zip file) includes functional, well-documented, and organized code for training the agent.
All the necessary code files and the Navigation.ipynb are included in the submission. Great job implementing the Deep Q Network algorithm to solve the banana environment!

Now that you have successfully completed the banana environment, I would suggest you to solve this environment by taking screen pixels as input! Here is a useful *github repository* which you can refer to if you need any help completing the task. All the best!

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*

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

✅ Saved model weights of the successful agent have been included in the github repository.

- `checkpoint.pth`

README

The GitHub (or zip file) submission includes a `README.md` file in the root of the repository.

✅ A detailed README file has been provided and is present in the repository.

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

- Good work providing the details of the project environment in the `Introduction` section of the README.
- The provided details cover the state and action spaces, the reward function and when the agent is considered solved.

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

- Good work providing the instructions for downloading the project environment files in the `Getting Started` section
- Also, good work covering in detail the instructions for installing the dependencies in the `Dependencies` section.

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.

The README contains the instructions to run the code in the repository.
`Navigation.ipynb` is the main file that should be used to train the agent.

Report

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

The project report is present in the submitted files.

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.

Model architecture used

Fully connected layer 1 with ReLU (input: 37 units (state_size), output: 64 units)

Fully connected layer 2 with ReLU (input: 64 units, output 64 units)

Fully connected layer 3 (input: 64 units, output: 4 units (action_size))

Hyperparameters used

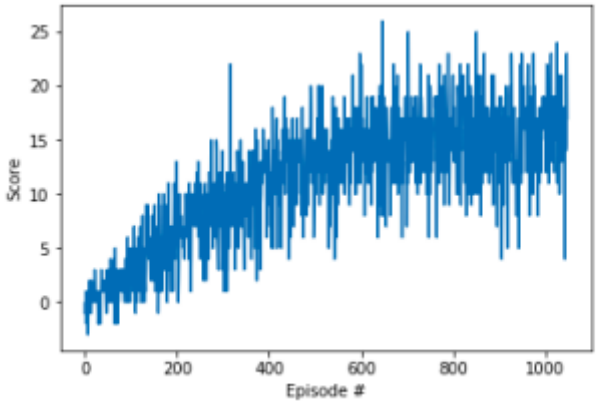
```
BUFFER_SIZE = int(1e5)  # replay buffer size
BATCH_SIZE = 64         # minibatch size
GAMMA = 0.99            # discount factor
TAU = 1e-3              # for soft update of target parameters
LR = 5e-4               # learning rate
UPDATE_EVERY = 4        # how often to update the network
```

A plot of rewards per episode is included to illustrate that the agent is able to receive an average reward (over 100 episodes) of at least +13. The submission reports the number of episodes needed to solve the

environment.

✅ Rewards plot has been provided for the implemented agent and seems to be great.
✅ The agent achieves an average score of +13 in just 600 episodes.

```
Episode 100     Average Score: 1.00
Episode 200     Average Score: 4.42
Episode 300     Average Score: 7.87
Episode 400     Average Score: 9.45
Episode 500     Average Score: 12.42
Episode 600     Average Score: 13.88
Episode 700     Average Score: 14.67
Episode 800     Average Score: 15.75
Episode 900     Average Score: 15.23
Episode 1000    Average Score: 15.13
Episode 1048    Average Score: 16.03
Environment solved in 948 episodes!     Average Score: 16.03
```



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

Idea to implement the following to further enhance the performance of the agent looks good:

- Double and Dueling DQN
- Rainbow algorithm
- Solving the pixel based environment
- Prioritized Experience Replay
- Better hyperparameter optimization
- Parameter noise

Suggestions

For implementing `Prioritized Experience Replay`, I would suggest you to use the `Sum Tree data structure` as it helps in a faster implementation of Prioritized Experience Replay. Please check *this implementation*.

Also, here is a useful post: *Let's make a DQN: Double Learning and Prioritized Experience Replay*

Also, I would like to suggest you to check the following resources as well.

- *Rainbow paper which examines six extensions to the DQN algorithm and empirically studies their combination*
- *Conquering OpenAI Retro Contest 2: Demystifying Rainbow Baseline*

- *This talk by David Silver, the author of the DQN paper*

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review

START