UDACITY

‹ Return to Classroom DISCUSS ON STUDENT HUB

Navigation

REVIEW

CODE REVIEW 3

HISTORY

Meets Specifications

Congratulations

Great submission!

You have implemented and trained the agent successfully.

All functions were implemented correctly and the Deep Q-learning algorithm seems to work quite well.

Suggestions to make your project even better!

- Include a GIF and/or link to a YouTube video of your trained agent!
- Write a blog post explaining the project and your implementation!
- Implement a double DQN, a dueling DQN, and/or prioritized experience replay!
- For an extra challenge after passing this project, try to train an agent from raw pixels! Check out (Optional) Challenge: Learning from Pixels in the classroom for more details.

Training Code

The repository (or zip file) includes functional, well-documented, and organized code for training the agent.

You implemented a Deep Q-learning algorithm, a very effective reinforcement learning algorithm.

Your code was functional, well-documented, and organized for training the agent.

Suggested reading:

- Google Python Style Guide
- Python Best Practices
- Clean Code Summary

The code is written in PyTorch and Python 3.

The code was written in PyTorch and Python 3.

Suggested reading about the discussion of what machine learning framework should be used.

- TensorFlow vs. Pytorch.
- Tensorflow or PyTorch : The force is strong with which one?
- What is the best programming language for Machine Learning?

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

You included the saved model weights of the successful agent.

Suggested reading:

- Saving and Loading Models
- Best way to save a trained model in PyTorch?

README

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

Well done! The submission included a README.md file.

I recommend you to check this awesome template to make good README.md.

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

Good work with environment details.

The README described all the project environment details.

Suggested description of the project environment details

- **Environment**: How is it like?
- **Agent** and its **actions**: When is it considered resolved?; What are the possible actions the agent can take?
- **State space**: Is it continuous or discrete?
- **Reward Function**: How is the agent rewarded?
- **Task**: What is its task?; Is the task episodic or not?

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

The README described all instructions for installing dependencies or downloading needed files in Getting Started item.

The README must describe all instructions for

- Installing dependencies
- Downloading needed files in Getting Started instructions item.

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.

Nice job!

I recommend you to check the Mastering Markdown, there are great tips about how to use Markdown

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.

All required files were included

You included the report of the project with the description of the implementation. Well done!

Report content guide:

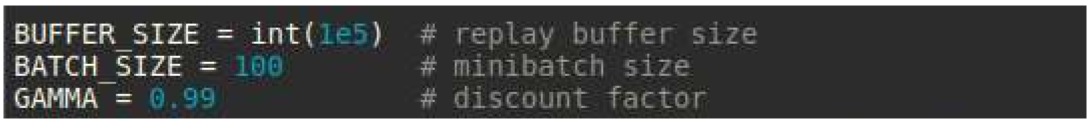
- Description of the learning algorithm
- The hyperparameters used.
- The model architecture.
- A plot showing the increase in average reward as the agent learns.
- The weakness found in the algorithm and how to improve it.

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

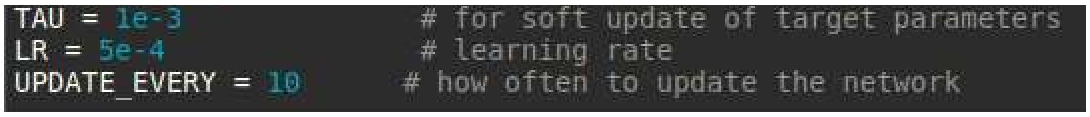
Nice job!

You described the learning algorithm, the chosen hyperparameters and the model architectures.

Hyperparameters:

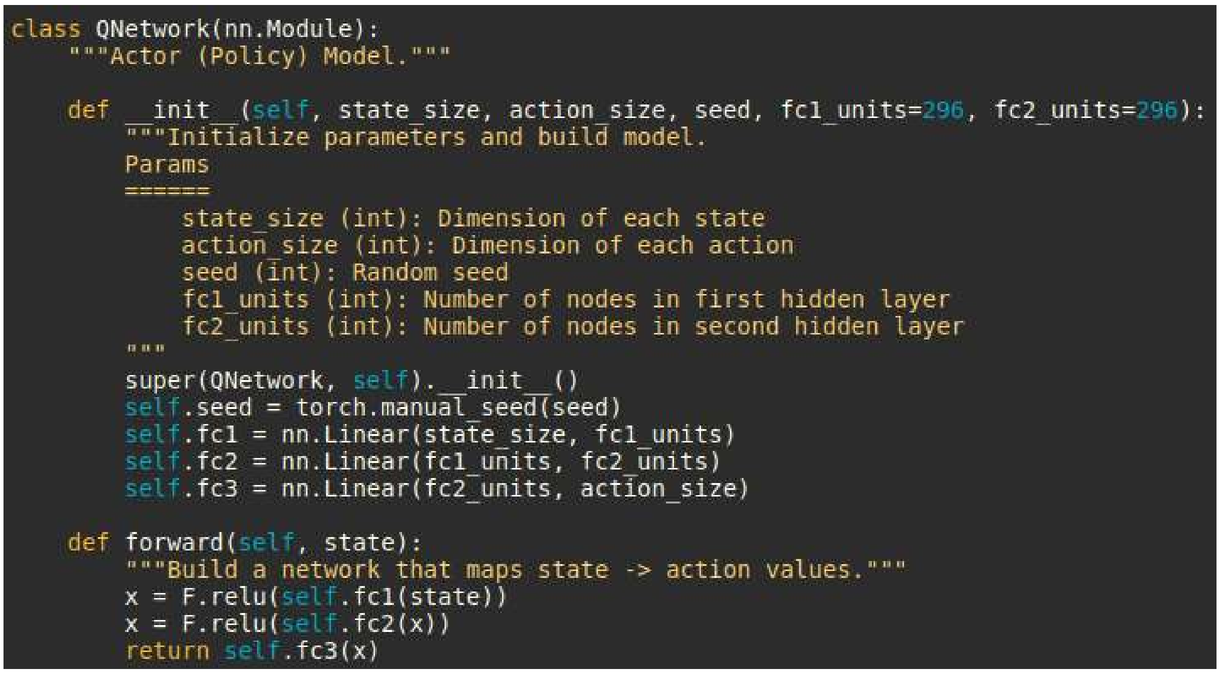


```
BUFFER_SIZE = int(1e5)  # replay buffer size
BATCH_SIZE = 100        # minibatch size
GAMMA = 0.99            # discount factor
```



```
TAU = 1e-3              # for soft update of target parameters
LR = 5e-4               # learning rate
UPDATE_EVERY = 10      # how often to update the network
```

Model architecture:

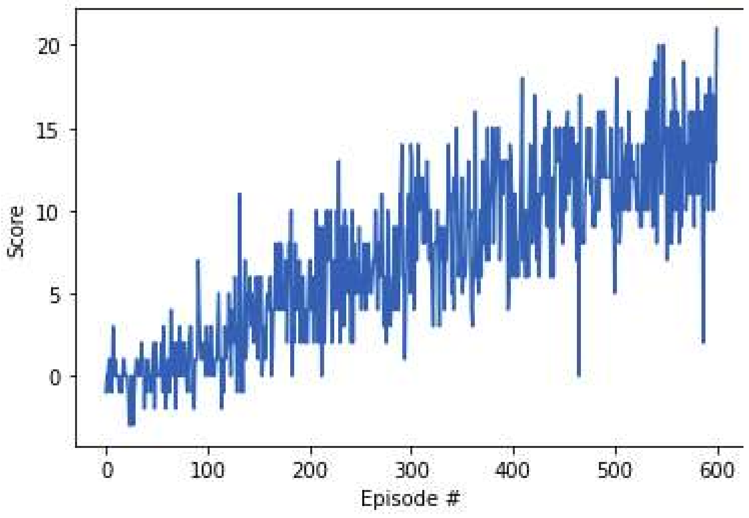


```
class QNetwork(nn.Module):
    """Actor (Policy) Model."""

    def __init__(self, state_size, action_size, seed, fc1_units=296, fc2_units=296):
        """Initialize parameters and build model.
        Params
        ======
            state_size (int): Dimension of each state
            action_size (int): Dimension of each action
            seed (int): Random seed
            fc1_units (int): Number of nodes in first hidden layer
            fc2_units (int): Number of nodes in second hidden layer
        """
        super(QNetwork, self).__init__()
        self.seed = torch.manual_seed(seed)
        self.fc1 = nn.Linear(state_size, fc1_units)
        self.fc2 = nn.Linear(fc1_units, fc2_units)
        self.fc3 = nn.Linear(fc2_units, action_size)

    def forward(self, state):
        """Build a network that maps state -> action values."""
        x = F.relu(self.fc1(state))
        x = F.relu(self.fc2(x))
        return self.fc3(x)
```

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.

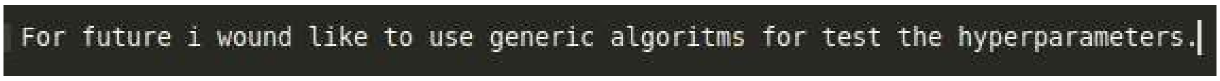
Nice work!

You informed the number of episodes needed to solve the environment and included a plot of rewards per episode.



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

The ideas you suggested to improve agent performance were very interesting



```
For future i wound like to use generic algoritms for test the hyperparameters.
```

You could consider the use of Prioritized Experience Replay

- Prioritized Experience Replay
- Implementation of Prioritized Experience Replay

DOWNLOAD PROJECT

3 CODE REVIEW COMMENTS ›

RETURN TO PATH