Navigation Report

I used Deep Reinforcement Learning to solve this Banana Navigation Environment. I implemented Double DQN to generate the final trained agent.

Learning Algorithm

The final agent is trained with Double DQN (Deep Q-Network).

The environment has a state space of 37 dimensions and many of them are continuous. If we use a Q-learning algorithm, the memory and number of steps needed to explore all possibilities are not feasible. Instead, we can replace the Q-table with a neural network. Therefore, DQN is picked for this environment. Furthermore, to address the potential issue of overestimating action values , Double DQN is applied. The idea is to use two different networks to reduce the correlation between action selection and Q value evaluation. We use the DQN network to pick the next action, while use a target network to get the Q-value. In this way, only the “real” optimal option will be chosen and yield high reward. As a result, I implemented Double DQN to solve this environment.

Some hyperparameters used:

- Experience replay buffer size: 100000
- Training batch size: 64
- Reward discounting Gamma: 0.99
- Soft update Tau: 0.001
- Learning rate: 0.0005
- Update the weights every 4 steps
- Epsilon range: 1 to 0.01
- Epsilon decay factor: 0.99

Furthermore, the agent also supports DQN in its `learn()` function. Simply specify `double_dqn=False` in the `learn()` function to use DQN instead. In the early stage, I used DQN to solve the environment. But Double DQN significantly reduce the number of episodes needed to solve the environment and show more stable performance.

For the Q-Network, the architecture is a simple feed-forward network, with 3 Fully Connected layers and ReLu Activation Function. The details are:

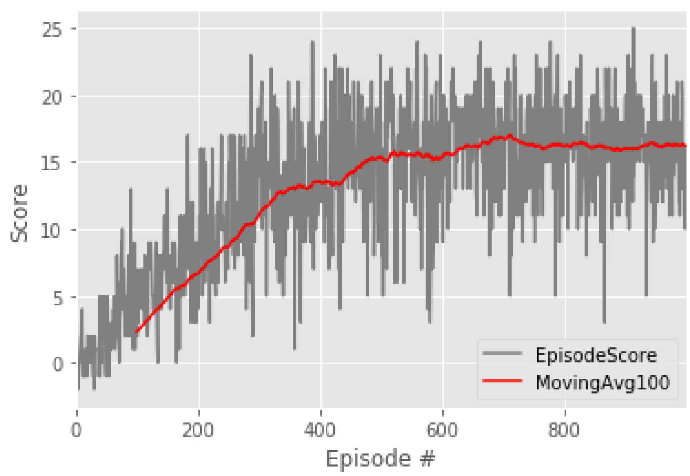
```
QNetwork(
  (fc1): Linear(in_features=37, out_features=64, bias=True)
  (fc2): Linear(in_features=64, out_features=64, bias=True)
  (fc3): Linear(in_features=64, out_features=4, bias=True)
)
```

Model Performance

The trained agent can solve the environment within 400 episodes. The average 100-episode reward is about +16 after 1000 episodes. Here are the details:

```
Episode 100      Average Score: 2.33
Episode 200      Average Score: 6.66
Episode 300      Average Score: 10.99
Episode 350      Average Score: 13.05
 * Environment first solved in 350 episodes! Average
Score: 13.05. Continue training...
Episode 400      Average Score: 13.49
Episode 500      Average Score: 15.40
Episode 600      Average Score: 15.32
Episode 700      Average Score: 16.78
Episode 800      Average Score: 16.19
Episode 900      Average Score: 15.98
Episode 1000     Average Score: 16.18
```

Total Reward/Score per Episode



Ideas for Future Work

From the related research in DQN, I think the following 2 techniques can help further improve the agent's performance:

- Prioritized experience replay:
  - There can be come cases that the TD error is high, and we should try to focus reducing such errors by giving it a higher weight in being selected from the experience replay
  - This should be able to help us use the past experiences more efficiently
- Dueling DQN
  - Separating state value estimates and advantage value estimates in the same network can potentially further help improve the model performance