Banana Project Report

This project was solved using DQN (deep Q learning network). Regarding the neural network, two hidden layers were used. The first layer contains 128 neurons, and the second hidden layer is 64 neurons. The output layer has a size of 4 as per action size.

The main parameters used during the learning were:

- Number of episode (=1000) I expected to converge before this number
- Time per episode (=1000) considered enough for training
- Epsilon start value = 1
- Epsilon end value = 0.01
- Epsilon decay = 0.995
- Replay buffer size = 100,000
- Gamma = 0.99 to give higher priority for current reward
- Learning rate = 0.0005

The device used is the CPU.

Problem Overview:

The **Banana Collector environment** is a reinforcement learning task where an agent must navigate in a 3D world and collect **yellow bananas** (reward: +1) while avoiding **blue bananas** (penalty: -1). The goal is to train an agent that maximizes the total reward over time.

What is DQN?

DQN is an algorithm that combines **Q-learning** with **deep neural networks**. Instead of maintaining a table of Q-values for each state-action pair (which becomes infeasible for high-dimensional states), we use a **neural network** to approximate the Q-function:

$$Q(s,a;\theta) \approx Q^*(s,a)$$

DQN Network Architecture:

- **Input**: State vector (e.g., 37 features representing the environment)
- **Hidden Layers**: Two fully connected layers (e.g., 64 and 64 units)
- **Output**: Q-values for each possible action

State (37) → FC (128) → ReLU → FC (64) → ReLU → FC (action_size)

Key Features of DQN:

- **Experience Replay**: Stores past experiences and samples random mini batches to break correlations between consecutive steps.
- **Target Network**: A separate network (Q_target) used to generate stable Q-value targets. It is updated less frequently to avoid oscillations.
- **Update Rule:**

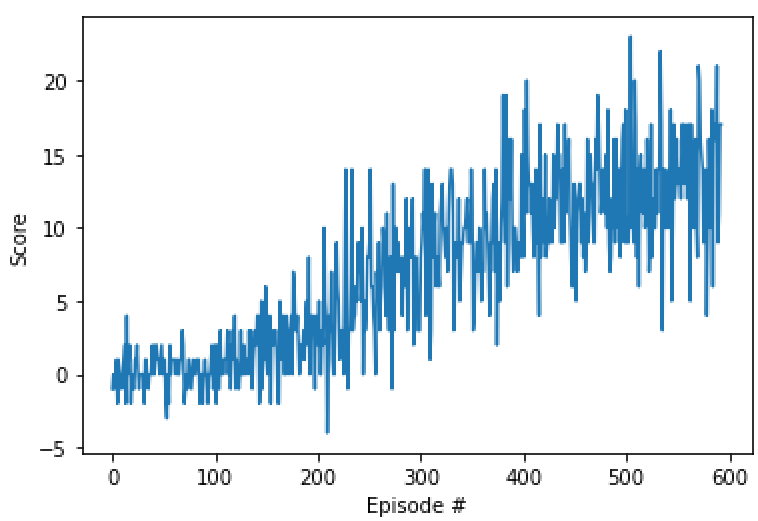
$$Q_{\text{target}} = r + \gamma \cdot \max_{a} Q_{\text{target}}(s', a)$$

$$\text{Loss} = (Q_{\text{expected}}(s, a) - Q_{\text{target}})^2$$

The program of Navigation called dqn_agent.py and created an agent based on that.

The result of the program came as follows:

```
Episode 100	Average Score: 0.21
Episode 200	Average Score: 1.77
Episode 300	Average Score: 5.57
Episode 400	Average Score: 9.41
Episode 500	Average Score: 11.87
Episode 593	Average Score: 13.02
Environment solved in 593 episodes!	Average Score: 13.02
```



The above figure indicates that the agent was able to get an average score above 13 in 100 consecutive episodes starting from episode 494 till 593.

Double DQN:

The double Deep Q learning network was tried in the above-mentioned project, and it included the same parameters and shape of neural network. However, there was small change in the update of the target network in double_dqn_agent.py to avoid overestimating.

Motivation

DQN often suffers from **overestimation** of Q-values because it uses the same network to both choose and evaluate actions. Double DQN addresses this by decoupling these two roles.

Double DQN Target Calculation

$$a_{\max} = \arg\max_a Q_{\text{online}}(s', a)$$

$$Q_{\text{target}} = r + \gamma \cdot Q_{\text{target}}(s', a_{\max})$$

- **Action selection** uses the **online network**.
- **Action evaluation** uses the **target network**.

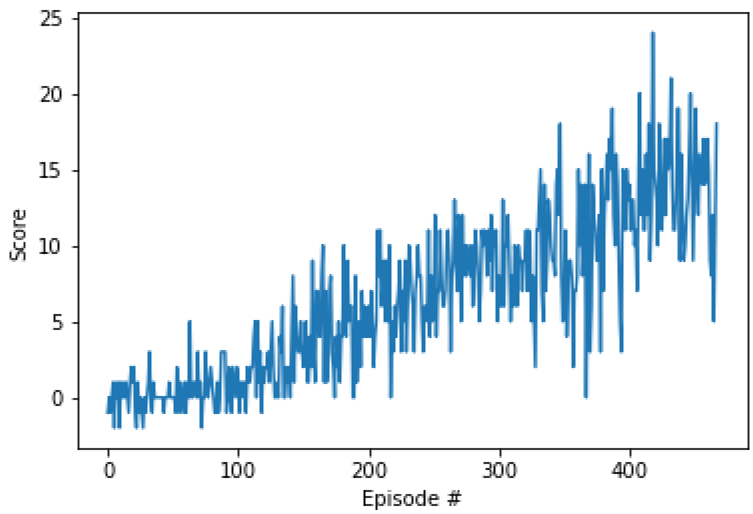
This approach results in more accurate value estimates and improved training stability.

Network Architecture

Identical to DQN.

The result for the double DQN came as shown below:

```
Episode 100	Average Score: 0.35
Episode 200	Average Score: 3.16
Episode 300	Average Score: 7.53
Episode 400	Average Score: 9.79
Episode 468	Average Score: 13.01
Environment solved in 468 episodes!	Average Score: 13.01
```



The project was solved from episode 369 and 468 with an average score above 13. This is more than 20% improvement of the standard DQN method.

Future Work:

For this project it is expected to give a better result if used Deul DQN along with other improvement methods (like Rainbow) The collection of the improvement method is supposed to converge faster.