Introduction

Project Goal

The project goal is to train a deep reinforcement learning agent to navigate a virtual world and collect as many yellow bananas as possible while avoiding blue bananas.

Project Overview

For this project, you will train an agent to navigate (and collect bananas!) in a large, square world. A reward of $+1$ is provided for collecting a yellow banana, and a reward of -1 is provided for collecting a blue banana. Thus, the goal of your agent is to collect as many yellow bananas as possible while avoiding blue bananas.
The state space has 37 dimensions and contains the agent's velocity, along with ray-based perception of objects around the agent's forward direction. Given this information, the agent has to learn how to best select actions. Four discrete actions are available, corresponding to:

- 0 - move forward.
- 1 - move backward.
- 2 - turn left.
- 3 - turn right.

The task is episodic, and in order to solve the environment, your agent must get an average score of $+13$ over 100 consecutive episodes.

Environment details

This project environment is based on the **Unity Machine Learning Agents Toolkit**
"The Unity Machine Learning Agents Toolkit (ML-Agents) is an open-source project that enables games and simulations to serve as environments for training intelligent agents. Agents can be trained using reinforcement learning, imitation learning, neuroevolution, or other machine learning methods through a simple-to-use Python API."
Note: The project environment provided by Udacity is similar to, but not identical to the Banana Collector environment on the Unity ML-Agents GitHub page.

Algorithms

- DQN agent with Experience Replay and Fixed Q-Targets [**Paper**]
- Double DQN agent with Experience Replay [**Paper**]
- DQN agent with Prioritized Experience Replay [**Paper**]

Model architecture

The network used consists of one input layer, two hidden layers with the ReLU action function, and one output layer.

- The input layer will depend upon the number of states, and the parameter will be passed on to

the class constructor.
- There are two fully connected hidden layers with 64 neurons each.
- The number of neurons in the output layer will depend upon the number of actions, and the parameter will be passed on to the class constructor.

Layer	Neurons	Activation
Input Layer	37	ReLU
Fully Connected Layer	64	ReLU
Fully Connected Layer	64	ReLU
Output Layer	4	

Hyper-parameters

Experience Replay Parameters

Parameter Name	Value	Description
Buffer size	100000	The number of entries in the memory.
Batch size	64	The number of experiences considered for each learning step.

Agent Parameters

Parameter Name	Value	Description
lr	0.0005	Learning rate
γ	0.99	Discount rate
τ	0.003	Soft-update for the 2^{nd} network.
Update after x	4	The agent will learn after x learning step.

Policy Parameters

Parameter Name	Value	Description
ϵ start	1.0	Initial ϵ
ϵ decay	0.995	ϵ decay rate
ϵ min	0.01	Minimum ϵ

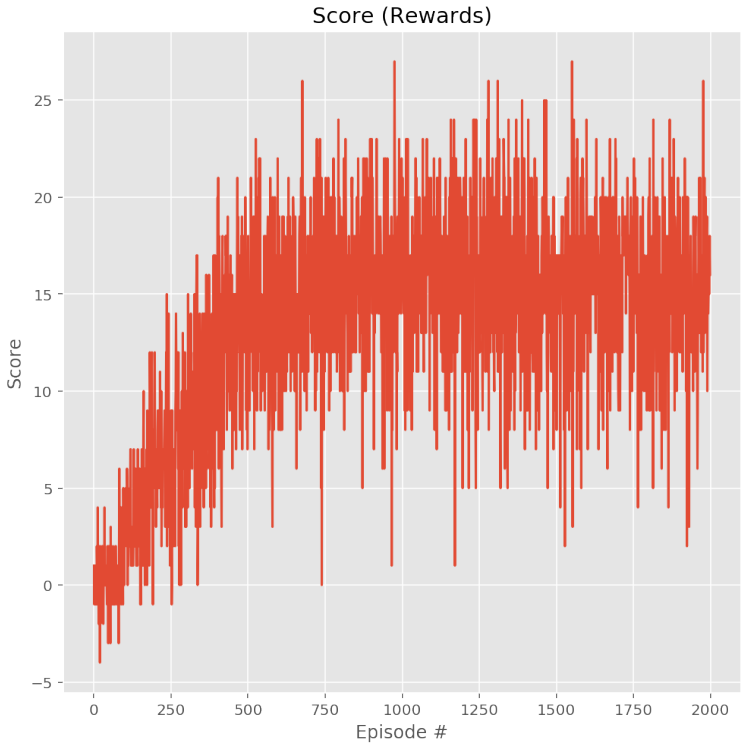
Summary Results

According to my findings, the Double DQN agent with Experience Replay works best in this environment, where the average score is 18.2 after ten episodes.

Future Work

In the project, my agent learned from information such as its velocity, along with the ray-based perception of objects around its forward direction. In the future, I would like to take a more challenging task: learning directly from pixels.

Training



Training Log for the DQN agent with Experience Replay and Fixed Q-Targets

```
Episode 100     Average Score: 0.56
Episode 200     Average Score: 4.22
Episode 300     Average Score: 6.72
Episode 400     Average Score: 9.40
Episode 500     Average Score: 13.16
Episode 600     Average Score: 14.17
Episode 700     Average Score: 14.66
Episode 800     Average Score: 15.65
Episode 900     Average Score: 15.86
Episode 1000    Average Score: 15.70
Episode 1100    Average Score: 16.32
Episode 1200    Average Score: 16.16
Episode 1300    Average Score: 15.65
Episode 1400    Average Score: 15.70
Episode 1500    Average Score: 15.00
Episode 1600    Average Score: 15.56
Episode 1700    Average Score: 15.87
Episode 1800    Average Score: 15.24
Episode 1900    Average Score: 15.20
Episode 2000    Average Score: 15.27

Total Training time = 29.5 min
```

Score after 2000 episodes for the DQN agent with Experience Replay and Fixed Q-Targets

Inference

```
Episode 1: 13.0
Episode 2: 14.0
Episode 3: 11.0
Episode 4: 20.0
Episode 5: 18.0
Episode 6: 16.0
Episode 7: 22.0
Episode 8: 16.0
Episode 9: 13.0
Episode 10: 23.0
All the scores[13.0, 14.0, 11.0, 20.0, 18.0, 16.0, 22.0, 16.0, 13.0, 23.0]
Mean Score: 16.6
```

Inference score of ten episodes for the DQN agent with Experience Replay and Fixed Q-Targets

```
Episode 1: 22.0
Episode 2: 17.0
Episode 3: 20.0
Episode 4: 19.0
Episode 5: 16.0
Episode 6: 25.0
Episode 7: 15.0
Episode 8: 18.0
Episode 9: 18.0
Episode 10: 12.0
All the scores[22.0, 17.0, 20.0, 19.0, 16.0, 25.0, 15.0, 18.0, 18.0, 12.0]
Mean Score: 18.2
```

Inference score of ten episodes for the Double DQN agent with Experience Replay

```
Episode 1: 21.0
Episode 2: 17.0
Episode 3: 18.0
Episode 4: 20.0
Episode 5: 19.0
Episode 6: 14.0
Episode 7: 18.0
Episode 8: 16.0
Episode 9: 14.0
Episode 10: 14.0
All the scores[21.0, 17.0, 18.0, 20.0, 19.0, 14.0, 18.0, 16.0, 14.0, 14.0]
Mean Score: 17.1
```

Inference score of ten episodes for the DQN agent with Prioritized Experience Replay