Project 1: Navigation

---

1 Introduction

For this project, I have trained an Agent using the Deep Q Network algorithm with Experience Replay. I have trained 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 the agent is to collect as many yellow bananas as possible while avoiding blue bananas.

The state space has 37 dimensions and the agent can take four actions

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

This game ends when the mean score across the last 100 episodes is greater than 13.

2 Implementation Approach

The Deep Q Learning algorithm with Experience Replay is implemented. The details of the algorithm are given in this research paper.

https://arxiv.org/abs/1312.5602

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

The following steps were taken to implement the algorithm

2.1 Define the Q Network

Define a Neural Network with 37 neurons in the input layer and 4 in the output layer corresponding to the state space and the action space of the environment.

2.2 Define the class for Experience Replay

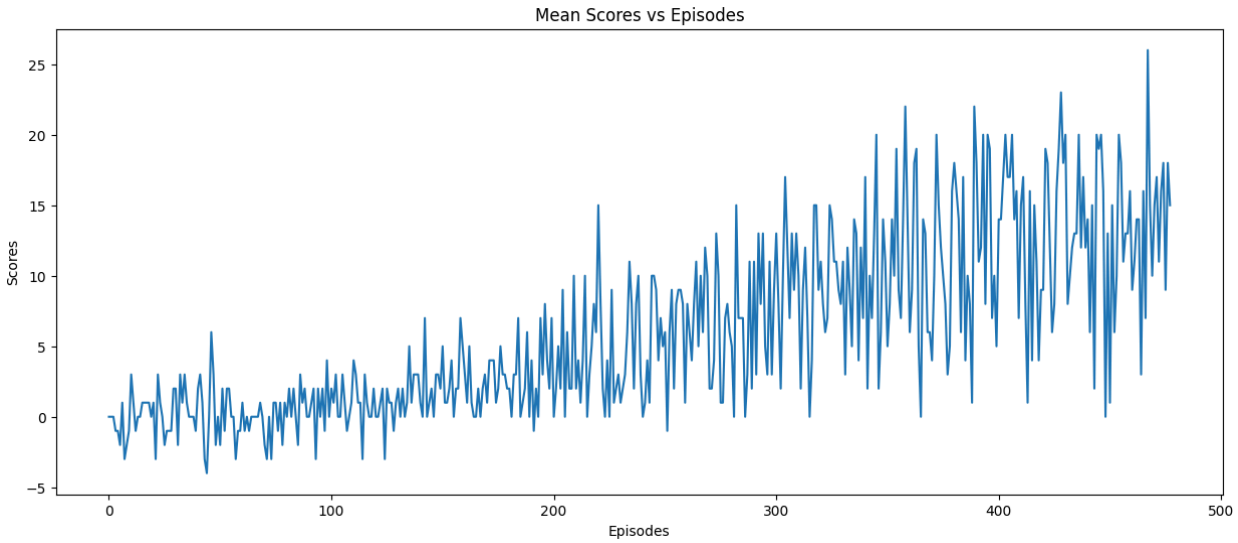
I used the example given in the class to create Experience Replay which is the Replay Buffer for state transitions along with actions, rewards.

2.3 Define the Agent

I defined the agent to implement the Deep Q Network algorithm with Experience Replay. The agent is trained for approximately 377 episodes. By this time, the mean score across last hundred episodes is more than 13 and the game terminates.

2.4 Plot the Scores

Using the Matplotlib library, I plotted the scores. The score curve looks as below



2.5 Evaluate the agent for 1 episode

I loaded the weights for the agent from the saved file ‘dqn.pth’ and played one episode of collecting Bananas. The agent was able to collect a score of 14 after one episode.

3 File Details

1. Navigation Project Solution v2 : The main file that contains the Q Network definition, Experience Replay, Agent and training functions.
2. dqn.pth - Trained weights saved after the agent was able to get an average score of 100 in the last 100 episodes.
3. Readme.md
4. Report.pdf

4 Ideas for Future Work

I would like to continue working on this project by implementing the Double DQN, Duelling DQN and the Prioritized Experience Replay algorithms to improve the performance.