Report for Navigation project

Objective

The objective of this project is to train an agent to navigate in a square virtual world that has blue and yellow bananas dropping from the sky at random locations, the agent needs to collect as many yellow bananas as possible and void blue bananas

Description of the implementation

The project consists of the following files:
Dqn_agent.py:
This file contains the Agent class and ReplayBuffer class
Agent:

- Constructor
  - Instantiates memory buffer
  - Instantiates local and target networks
- step
  - Stores a step in the replay buffer
  - Updates the target network weight every 4 steps with the current weights of the local network
- act
  - Returns the action for the given state
- learn
  - Updates the weights of the deep learning model using the experiences from the replay buffer
- soft_update
  - used in the learn method to softly update the target deep learning mode from the local model

ReplayBuffer

- add
  - adds an experiences step to the memory buffer
- sample
  - randomly samples a batch of experiences steps for learning

Model.py

The learning network uses the following architecture:

1. Linear layer (37,64)
2. ReLU
3. Linear layer (64,64)
4. ReLU
5. Linear layer(64,4)

This is first architecture that I tried on this project, from the reward that we can see, the size of the deep learning model is too small, it's learning capability peaked after about 800 episodes, the additional training didn't dramatically improve the performance of the model.

Navigation.ipynb

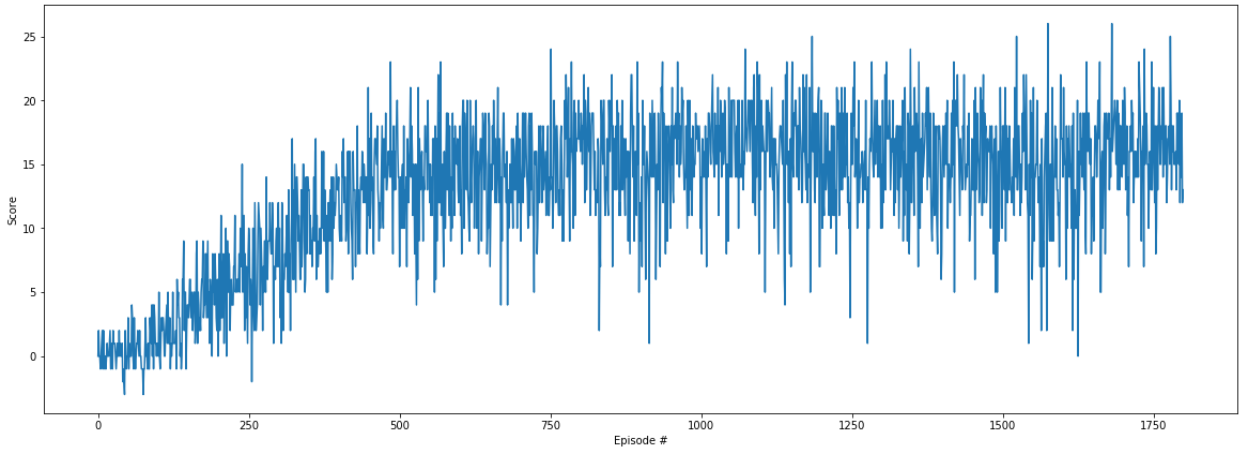
This is the entry point of the project that provides the environment and training loop for the agent. It also plots the scores

Learning Algorithm

At each time stamp in the environment

- Choose an action from the deep learning network
- Observe the reward
- Observe next time stamp
- Store experience in the memory
- Obtain random batch of experiences from reply memory
- Train the deep learning network
- Repeat

Plot of rewards



Ideas for Future Work

- Increase the width and depth of the deep learning model
- Hyperparameters fine tuning
- Prioritized experience replay
- Dueling DQN