Navigation Project

Reinforcement Learning is a branch of Artificial Intelligence which purpose is to maximize the expected reward of a goal-oriented task.
In this project, the goal of the agent is to collect as many yellow bananas as possible while avoiding blue bananas in a virtual environment.
Deep Q-Learning uses a neural network to approximate the value function by receiving the current state as input and as output will return the optimal action.
For this environment the agent's learning algorithm, Deep Q-Learning, has good performance because there are two neural networks involved to reduce randomness along the learning transition.
Why 2 Deep NNs?

1) The first NN will be initialized, just as any other neural network, with random weights from a normal distribution. At the earliest stages of the environment the actions taken will consists of mostly randomness since epsilon is close to 1. This model will be fitted at every step with the less random outputs of the second nn.

2) The second DNN will predict at each step in order to fix this randomness. The prediction will be done with mini batch of past experiences taken at random at a designated step , this is called reply memory

3) The outcome of the second DNN will be used to fit/train the first model so that it learns something

Neural Network Architecture:

The model consist of 3 linear layers of sizes 80, 60, input size of 37 (state space), and an output size of 4 (action space) with ReLu activation functions. The model is able to learn complex representations of the data.

Algorithm hyperparameters:

```
BUFFER_SIZE = int(1e5)  # replay buffer size
BATCH_SIZE = 64         # minibatch size (default: 64)
GAMMA = 0.995           # discount factor (default: 0.99)
TAU = 1e-3              # for soft update of target parameters
LR = 5e-4               # learning rate
UPDATE_EVERY = 5        # how often to update the network
```

Ideas for future work:

An initial improvement would be to modify the dimension of the hidden layers to train the model in a smaller number of episodes.

Additionally, in order to reduce overestimation a Double Q/learning model could be applied or Priorized Experience re-play to sample from more valuable experiences.

```
Episode 100     Average Score: 0.46
Episode 200     Average Score: 2.86
Episode 300     Average Score: 6.73
Episode 400     Average Score: 9.13
Episode 500     Average Score: 10.47
Episode 600     Average Score: 10.69
Episode 700     Average Score: 11.57
Episode 800     Average Score: 12.08
Episode 900     Average Score: 12.75
Episode 904     Average Score: 13.00
Environment solved in 804 episodes!     Average Score: 13.00
```