NAVIGATION PROJECT – Javier Tausía Hoyal

*** This is the explanatory file Report.pdf that summarizes the strategy that has been followed solving the Navigation problem (yellow bananas problem) for the Deep Reinforcement Learning Nanodegree ***

First, the description of the environment and how it can be downloaded is explained in the README.md file, so here we will focus on the model used and the philosphy in the selection of the parameters.

Regarding the model, 2 fully conected deep layers has been used with shape:

```
self.fc1 = nn.Linear(state_size, 64)
self.fc2 = nn.Linear(64, 64)
self.fc3 = nn.Linear(64, action_size)
```

with a total of **# params** = state_size(37)*64 + 64*64 + 64*action_size(4) = 6720. This model is very simple, and a more complex model could has been used, but as this model worked, we tought that complicating the model architecture will not be very useful in the improvement of learning.

Regarding the tune of the parameters, different combinations have been used, all leading to similar results at the end of the day. The final configuration of hyperparameters was:

BATCH_SIZE = 64	# minibatch size
GAMMA = 0.9	# discount factor
TAU = 1e-3	# for soft update of target parameters
LR = 5e-4	# learning rate
UPDATE_EVERY = 4	# how often to update the network

Table 1. Final configuration of hyperparameters.

Regarding epsilon, it was changed after each episode, from a value of 1 and decreasing. As it can be seen in Figure 1, it helps the agent explore at the begining, and then it focuses on exploiting the knowledge. It can also be seen in the figure that after 600 episodes approx, the agent does not really learn anything, and it is due to the fact that the agent chooses the greedy action most of the cases, not exploring at all.

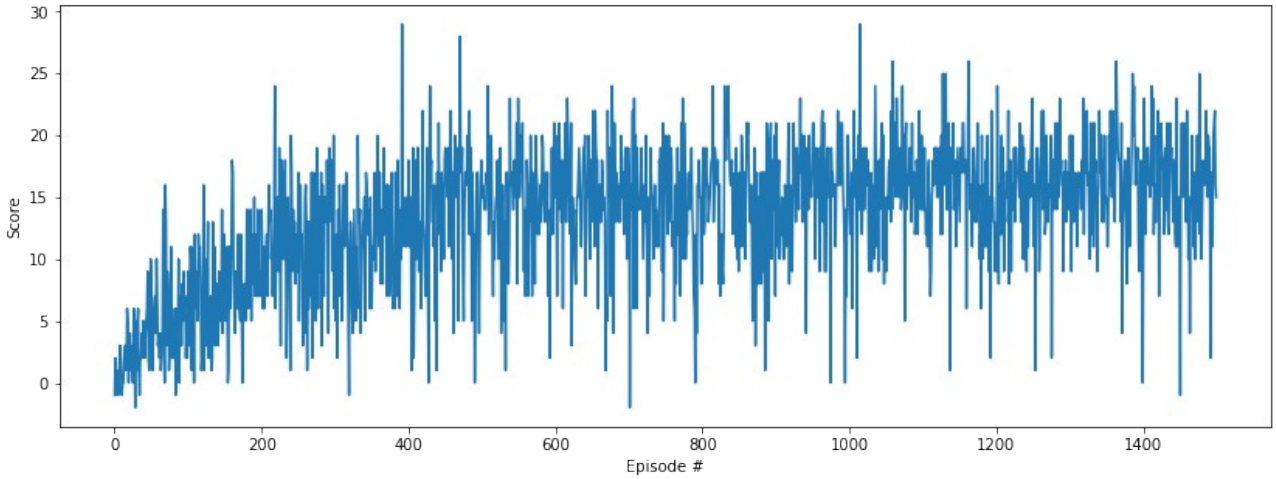

Figure 1. Average score (mean of last 100 episodes) after each episode.

SUMMARY:

Finally, just comment that although the agent has achieved the goal of improving an average score over 13, and in less than 600 episodes, a lot of things can be done to improve the performance of the learning agent. Tunning the hyperparameters is an option, but other things such as double DQN, dueling DQN or prioritized experience learning can be implemented.