Training an agent for Navigation using Reinforcement Learning

Task Overview:

We train an agent to navigate (and collect bananas!) in a large, square world using Double DQN. 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.

State and action space:

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.

We utilize Unity Machine Learning Agents plugin to interact with the environment.

Learning Algorithm:

We compute the following loss function for Double Q-Learning:

$$Y_t^{\text{DoubleQ}} \equiv R_{t+1} + \gamma Q(S_{t+1}, \underset{a}{\operatorname{argmax}}\, Q(S_{t+1}, a; \boldsymbol{\theta}_t); \boldsymbol{\theta}'_t)$$

Where,

R_{t+1} - Reward of the current state S_t and Action A_t
Q - Action value function
S_{t+1} - Next State
a - Action
γ - Discounting factor which can take a value between 0 and 1.
θ_t - Online neural network that is currently being trained
θ'_t - Target neural network

Network Structure:

We’ve used a fully connected neural network with 3 layers that takes in 37 inputs (the current state of the agent) and generates 4 outputs (the action values). We have used a batch normalization and relu activation function after each layer.

Replay Buffer:

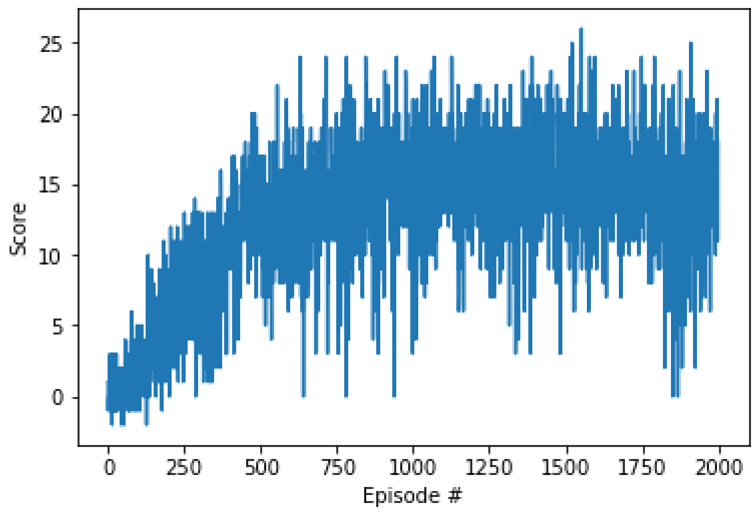
We have used a replay buffer that utilizes the concept of experience replay. We gather the experiences of the agent in a deque. The deque size is set to 100000. We randomly sample the buffer pool and do a supervised training of the online network against the target network. Finally, a soft update is done by copying the parameters of the online network to the target network.

Training details:

We ran the training for 2000 episodes. The learning rate of training the neural network was set to 0.005. The discounting factor was set to 0.99. The average reward is calculated for the last 100 episodes of the training and monitored.

Results:

The agent got an average reward of **13.11** in the 600th episode. Finally, an average score of **14.97** was obtained during the 2000th episode.



Ideas for future work:

Prioritized Experience Replay could be implemented to further enhance the performance of the agent, so as to replay important transitions more frequently, and therefore learn more efficiently. We can assign priority weights for each transition step, which will be directly proportional to the TD error. Instead of randomly sampling from the buffer pool, we could sample based on the priority weights.