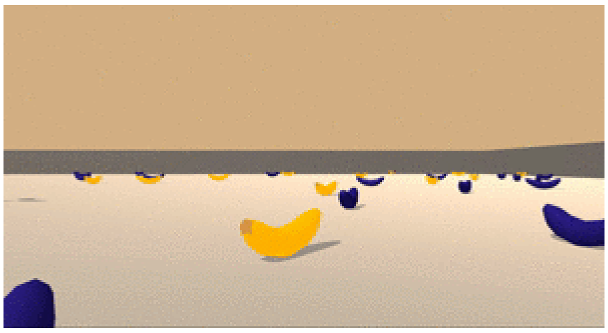
report.md

Report

The Enviroment

For this project, you will train 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 your agent is to collect as many yellow bananas as possible while avoiding blue bananas.

The state space has `37` dimensions and contains the agent's velocity, along with ray-based perception of objects around agent's forward direction. Given this information, the agent has to learn how to best select actions. `4` discrete actions are available, corresponding to:

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

The task is episodic, and in order to solve the environment, your agent must get an average score of `+13` over `100` consecutive episodes.

Solution

The learning algorithm is implented in the files `Navigation.ipynb` and `dqn_agent.py`. Once the agent solves the environment with an average score of `+13` over `100` consecutive episodes, the underlying weights for the Q-Network are saved in the file `state_dict.pth`.

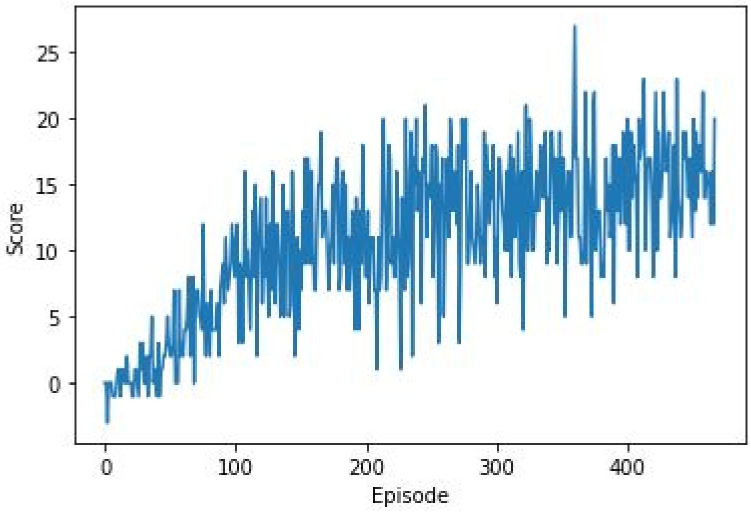
Deep Q-Learning

The agent uses a Deep Q-Network with RELU activation and two fully conected hidden layers. The final linear output layer of the DQN produces a vector containing the action values for each possible action. From this vector we can choose an action by random or pick the action with the highest action value.

Also, the agent maintains a replay memory and buffers all observed state-action-tuples rather than directly learning during training. Experience replay is used to prevent the action values from oscillating or diverging catastrophically. In order to update the DQN, we sample the replay memory at random every `4` time steps and learn from a random batch of `64` state-action-tuples to break correlations between consecutive time steps.

Results

Training finished in `467` episodes with an average score of `15.01` over the recent `100` consecutive episodes.



Future Improvements

In the future, we could improve experience replay by adding a priority to the experencied samples. Instead of randomly sampling from the replay buffer we then produce a batch of samples that maximizes the TD error delta. The bigger the error, the more the agent can learn from this sample.

Additionally, by manually tweaking hyper-parameters such as `UPDATE_EVERY` and `BATCH_SIZE` finding an optimal value may be very time consuming. Instead of manually manipulating these parameters, we could create a pipeline that automatically optimizes hyper-parameters.