DEEP REINFORCEMENT LEARNING PROJECT 1 – NAVIGATION

The goal of this project is to train an agent to navigate a world while collect bananas. A reward of +1 is provided for collecting a yellow banana, and a reward of -1 is provided for collecting a blue banana. The environment state space has 37 dimensions and contains the agent's velocity, along with ray-based perception of objects around the agent's forward direction. The action space consists of four discrete actions corresponding to moving forward, backward, left, and right. The task is episodic and is solved when the agent has achieved a score of 200.

LEARNING ALGORITHM

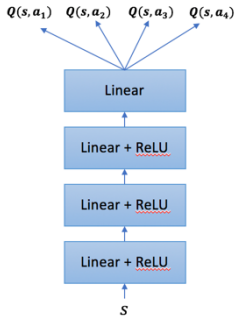
The agent uses a neural network-based Q-learning algorithm to learn a policy that dictates how the agent behaves in the environment. This algorithm involves the agent interacting with the world to collect experiences of the form $\langle s_t, a_t, s_{t+1}, r_{t+1} \rangle$ where s, a, r, t denote state, action, reward and time. Using these experiences, the agent estimates the expected return of a state action pair $Q(s, a)$ using a neural network. The network's parameters are updated using stochastic gradient descent so as to minimize the temporal difference (TD) error defined as follows

$$r_{t+1} + \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t).$$

Furthermore, while learning through interaction with the environment the agent uses an ϵ-greedy policy in order to explore the environment as well as exploit the knowledge gained. Under the an ϵ-greedy policy the agent will with some probability p select an action at random (exploration) or select the action that maximized the estimate of $Q(s, a)$ (exploitation). The probability p decreases as the agent trains so the agent transitions from exploration to exploitation.

NEURAL NETWORK ARCHITECTURE

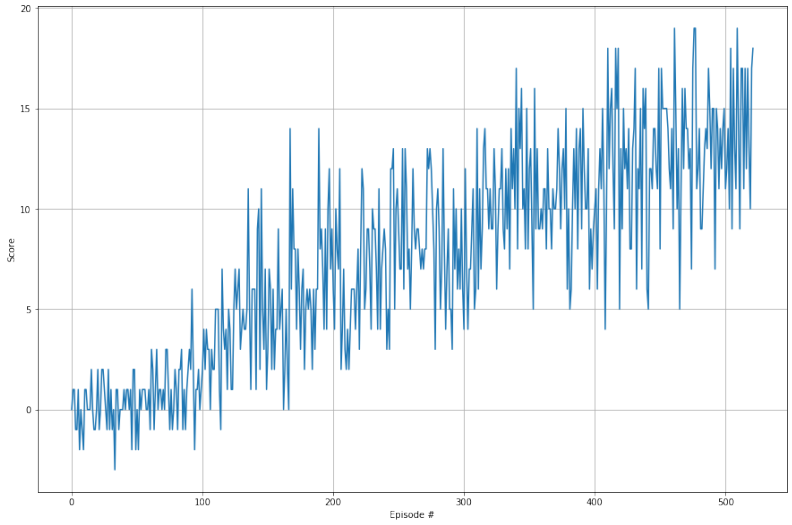
The basic architecture of the network used to approximate $Q(s, a)$ is illustrated in the figure to the right. The network contains a variable number of fully-connected layers (1, 2, or 3) each with a variable number of nodes (32, 64, or 128) and each followed by a rectified linear unit (ReLU). The last layer of the neural network is a linear layer with a number of output nodes equal to the number of actions. The outputs produced by this final layer correspond to $Q(s, a)$ for each action a.



The table below illustrates how network architecture impacts the agent's ability to learn. Networks with 1, 2, or 3 fully-connected layers and 32, 64, or 128 nodes were evaluated. For each combination, one can see whether the agent achieves an average score of 13 over 100 episodes (e.g Pass or Fail) and how many episodes were required. The agent achieved the desired performance for all configurations except when using a network with a single fully-connected layer and 32 nodes.

# Fully-Connected Layers	32 Nodes	64 Nodes	128 Nodes
1	Fail	Pass @ 471 episodes	Pass @ 560 episodes
2	Pass @ 494 episodes	Pass @ 487 episodes	Pass @ 543 episodes
3	Pass @ 518 episodes	Pass @ 483 episodes	Pass @ 527 episodes

The figure below illustrates the evolution of the agent's score as a function of the number of episodes the agent interacts with the world when using a neural network with a single fully-connected layer and 64 nodes. Early on (fewer than 100 episodes) the agent generates an average score 0. After the first 100 episodes, the agent progressively improves. Following 470 episodes, the agent has reached the average target performance of 13.



Note that other network and Q-learning algorithm parameters remains fixed throughout experimentation (e.g. discount factor gamma = 0.99, soft update of target parameters = 1e-3, learning rate = 5e-4, epsilon decay 0.995).

FUTURE WORK

The network architecture for estimating $Q(s, a)$ can be improved as illustrated in the figure to the right. Each of the fully-connected layers can be followed by a batch-normalization and/or dropout layer. Batch normalization can help the model converge faster while dropout can help the model generalize and avoid overfitting. These two modifications could potentially allow the agent to reach the desired performance after a smaller number of episodes.