Banana Navigator Report

By Brandon Suen

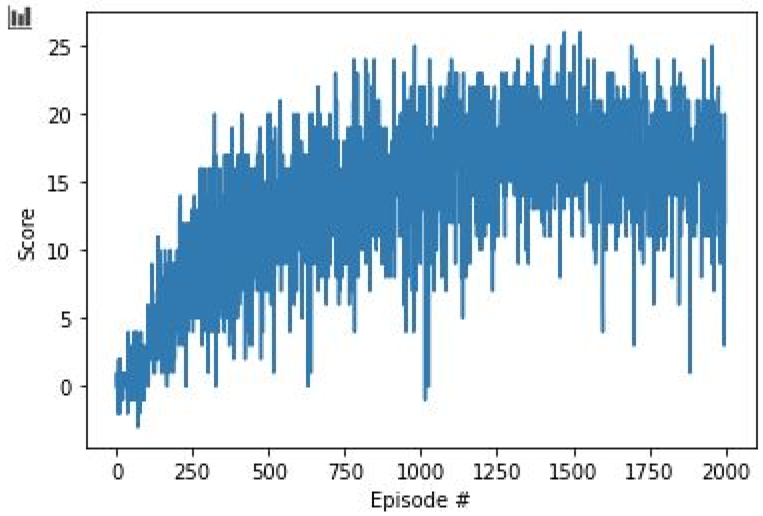
Approach

For this project, I implemented a deep q-network that uses a neural network built using PyTorch. My first attempt included a double deep q-network with experience replay and fixed Q targets. The neural network had a single stream with a sequence of linear layers and relu activation functions. I later changed this to a dueling architecture that separates the action-value function into a state-value function and an advantage function, each of which has its own stream. Now, the neural network has two linear layers that then separate into these two streams, each of which has two linear layers of its own. I also altered the sampling phase for experience replay to collect 2 episodes of experiences that are capped at 200 steps each.

Hyperparameter Tuning

The first thing I changed was gamma. I slightly shifted the focus more towards immediate reward than future rewards by lessening gamma from 0.99 to 0.95. After that, I changed various hyperparameters to no avail. The batch size (64), tau (1e-3), the epsilon starting value (1), the epsilon decay rate (0.995), the epsilon ending value (0.01), the update period (4), and the learning rate for the Adam optimizer (5e-4) seemed to work well with their initial values.

Results



The agent was able to meet the required mark of averaging a score of 13 over the last 100 episodes shortly after episode 600, and it got up to an average of 17.6 over the last 100 episodes by episode 1,400.

Future Ideas

There are many improvements I could make to my implementation. For example, I could implement prioritized experience replay in order to emphasize the experiences that the agent can learn the most from. I could also spend more time optimizing hyperparameters or entirely restructure the implementation to include policy-based methods.