Problem Description
In life, occasionally one "goes bananas" and finds it necessary to eat more of the said plant than is clinically necessary for maintenance of personal emotional and physical homeostasis. At this critical junction it is imperative that one eats strictly only yellow ripe bananas since the ingestion of blue bananas increases the personal toxicity levels of an individual. Specifically, the toxicity of one blue banana (-1) outweighs exactly the benefits of a single yellow ripe banana (+1).
Fortunately, this difficult scenario has been simulated to a high degree of accuracy by Unity ML in what has been termed the Banana environment. Within this complex yet nuanced simulation, the act of ingesting said plants is considered to have succeeded if and only if at least 13 more yellow ripe bananas have been ingested then blue poisoned bananas.
The automated solution of this simulation has been provided in the case that our minds are replaced with those of automatons and we thereby require this external control in order to complete this critical and fundamental staple task of our everyday existence.

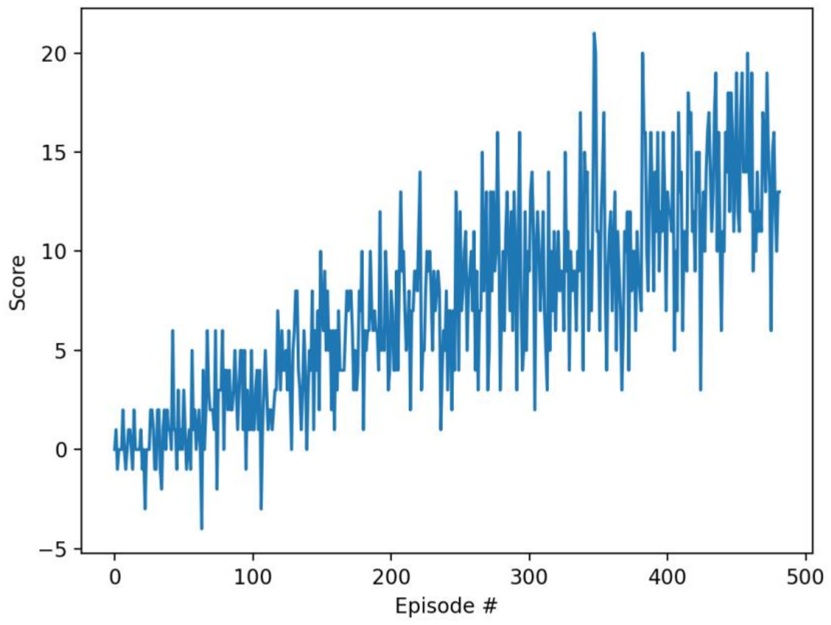
Implementation
The above problem has been solved in two ways, one using Pytorch as required by the Udacity overlords, and a second using Tensorflow 2.2 in order to upkeep with modernity. Note that in order to run the TF version, an additional installation step is required, as indicated in the Readme. The default solution has been kept at Pytorch in order to satiate that draconian rubric checkmark.
Both of the provided solutions use Deep Q-Learning, and by a strange coincidence both of them use the exact same hyperparameters provided in the solution for the prior DQN section. The only difference between the two is that the TF implementation additionally implements Double Q-Learning.

The chosen parameters are below:
```
BUFFER_SIZE = int(1e5)  # replay buffer size
BATCH_SIZE = 64  # minibatch size
GAMMA = 0.99  # 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
```

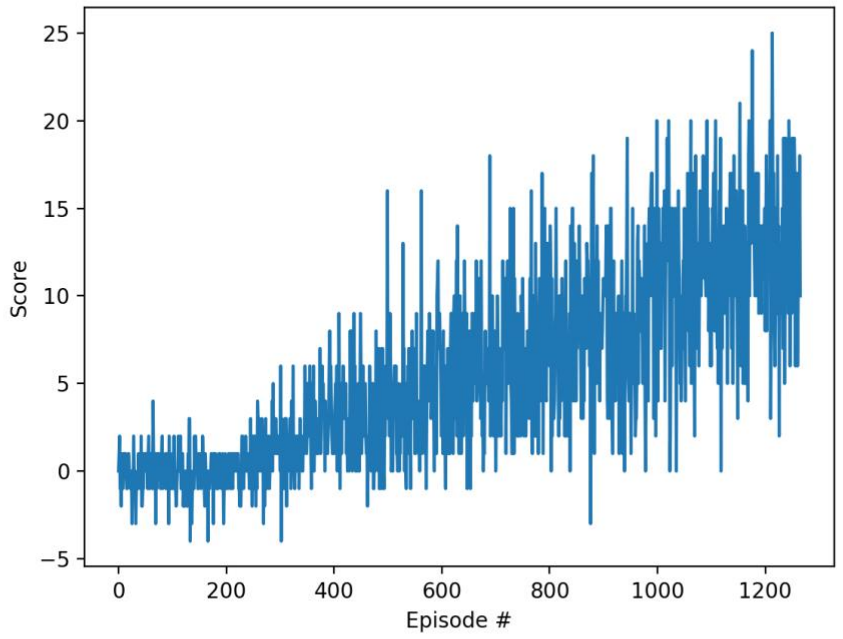
The neural network architecture used contains two hidden layers, each of which contained 64 units. Aside from the, the architecture followed the general outline of the DQN Nature paper https://storage.googleapis.com/deepmind-media/dqn/DQNNaturePaper.pdf

Results
The above highly original (yet replicated in TF 2.2!) implementation produced convergent results in both the TF and pytorch cases.

PyTorch:



Tensorflow 2.2:



Further Work:

A couple of easy wins which may be gotten in order to improve the speed of convergence may be found by implementing prioritized experience replay, dueling DQN, or by the “everything and the kitchen sink” algorithm, more marketably known as “Rainbow”