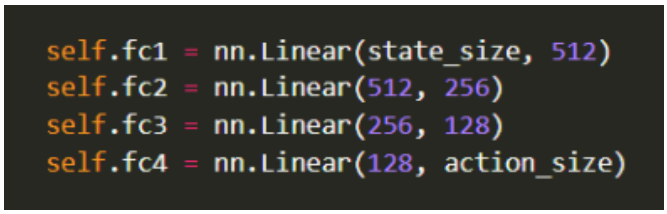
Deep RL Project 1 - Navigation

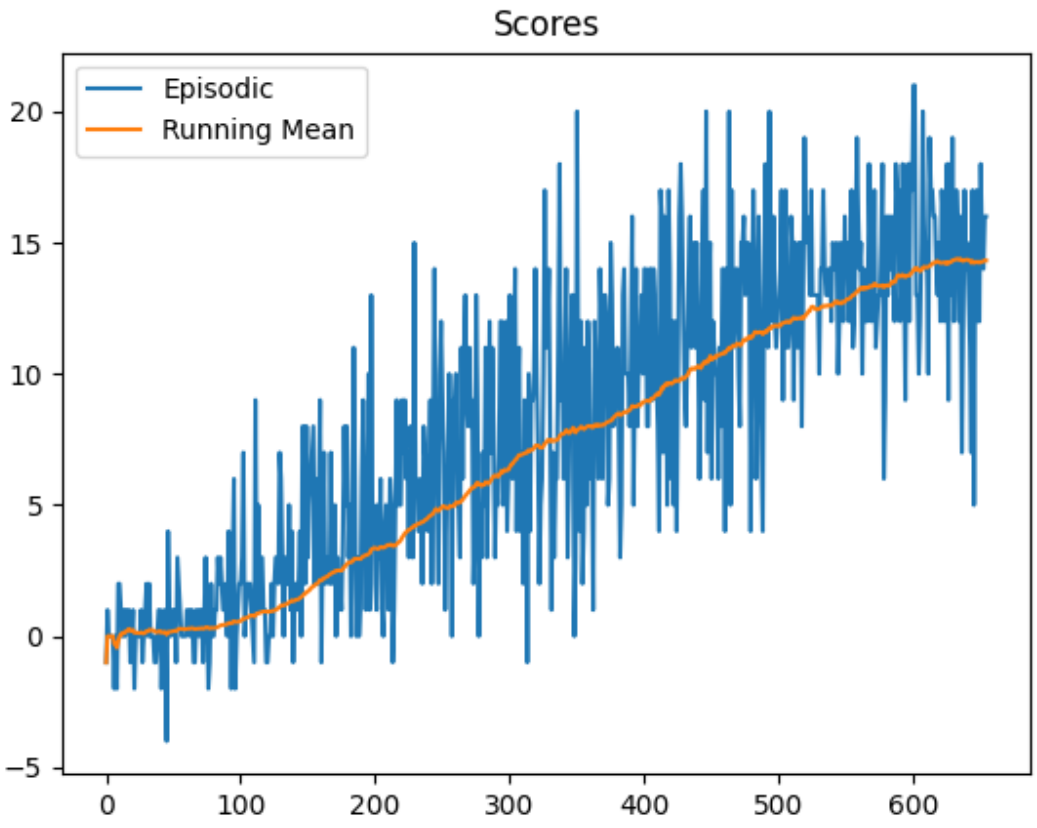
My solution to the Banana environment implements a basic DQN model based on the DQN tutorial covered as part of the nanodegree. The Q-Network was implemented as a simple Multi Layer Perceptron taking the environment state as its input. The Q-Network and training hyperparameters used are summarised below.

Parameter	Description	Value
Replay Buffer Size	The maximum size of the buffer used for experience replay.	100000
Buffer Batch Size	The number of memory tuples used to generate gradients to update the local network.	64
Gamma	The discount factor used to prioritise more immediate rewards.	0.99
Tau	The soft-update parameter which controls the weighing of the local and target Q Network parameters when updating the latter. Tuning this parameter appropriately avoids the 'moving target' problem.	0.001
Optimiser Learning Rate	A parameter controlling the magnitude of parameter updates during gradient descent.	0.0005
Network Update Interval	The number of timesteps elapsed before updating the local Q Network.	4
Initial Epsilon	Sets initial balance between exploration and exploitation.	1
Epsilon Decay	Decays epsilon to make the agent more exploitative as it learns.	0.995
Minimum Epsilon	Minimum epsilon value during training.	0.01
Model Architecture	 State Size = 37 Action Size = 4 Activations:	

```
self.fc1 = nn.Linear(state_size, 512)
self.fc2 = nn.Linear(512, 256)
self.fc3 = nn.Linear(256, 128)
self.fc4 = nn.Linear(128, action_size)
```

	FC, FC2, FC3 - Relu FC4 - None

Whether or not the environment had successfully been solved was monitored by maintaining a running mean metric over the last 100 episodes. A plot for the fully trained agent can be found below.



From this we see that the agent is able to sustain a running mean score of +13 or more over 100 episodes at episode number 654. A screenshot of the training script output during this run is provided below.

Possible Improvements

- **DOUBLE DQN -** This approach would aim to reduce bias in the predicted state-action value by having two target networks and randomly selecting one to predict the action producing the largest state-action value in the next state whilst using the other in the role of the local network. The future reward predictions contributed by both networks should neutralise the bias in either one of them over time.
  Reference:
  https://papers.nips.cc/paper/2010/file/091d584fced301b442654dd8c23b3fc9-Paper.pdf

- **PRIORITISED EXPERIENCE REPLAY -** This technique would give weightings to tuples from the Replay Buffer, with tuples that lead to the generation of greater losses being weighted more heavily. This should allow the agent to learn from the most informative experiences, which should make it more likely to learn an optimal policy.
  Reference: https://arxiv.org/pdf/1511.05952.pdf