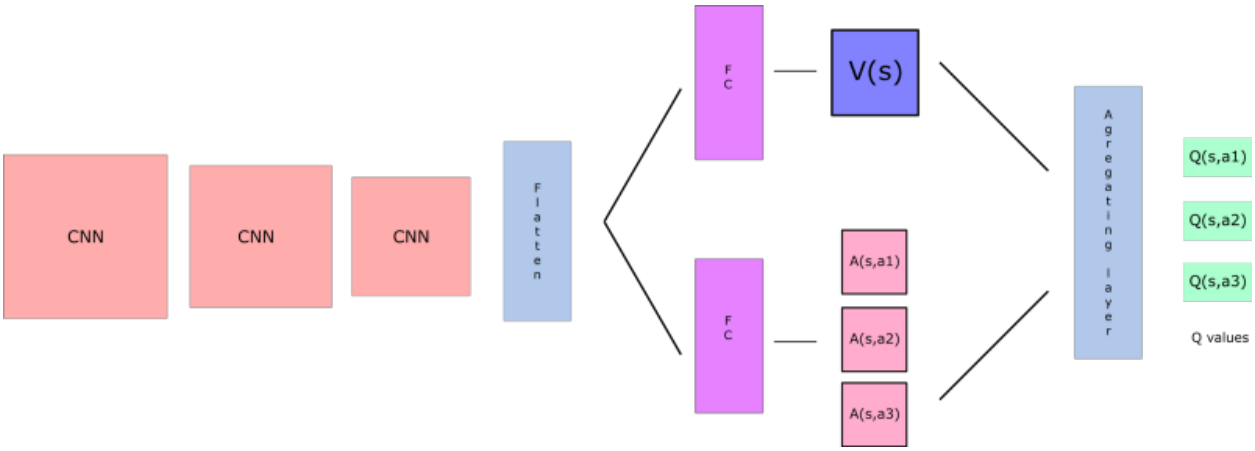
Report

Project 1: Navigation

1.0 The Model and Algorithm

The implemented learning algorithm was a Dueling Q-network that utilized two independent fully connected networks to generate state values and data-dependent action advantages separately. This ensures that the model does not need to learn the values of actions that may be unnecessary at each step, leading to faster convergence. By separating both estimators, the model learns which states are and are not valuable, and effects of their corresponding action pairing.

A standard Dueling Q-network architecture is as follows:



Source: https://www.freecodecamp.org/news/improvements-in-deep-q-learning-dueling-double-dqn-prioritized-experience-replay-and-fixed-58b130cc5682/

Here we can see the two independent neural networks on the right side, one for each value. This architecture, however, is not fully appropriate for our model, as a CNN was not used. Instead, think of the 'Flatten' layer in the image as the state-vector passed to our model as input from the Unity environment.

The output from the two networks is combined into the aggregating layer, by a simple addition of the state-action value using the following equation:

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + \left(A(s,a;\theta,\alpha) - \frac{1}{|\mathcal{A}|}\sum_{a'} A(s,a;\theta,\alpha)\right)$$

Please not above that we are subtracting the mean of the action advantages from the actual advantages to improve practical performance. Please refer to this paper for further information: https://arxiv.org/abs/1511.06581.

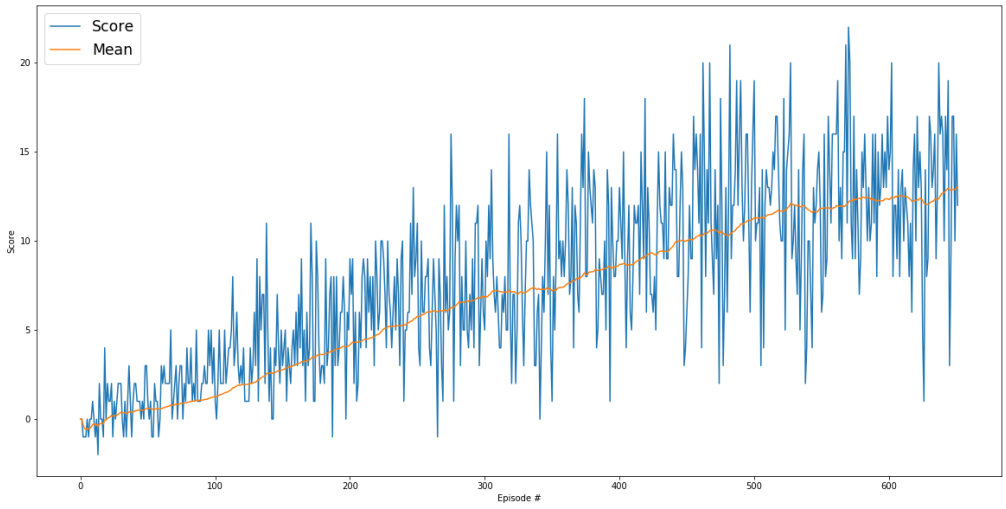
The two neural networks were set up to each have a fully connected layer, activated by a RELU function and with a dropout probability of 20% to improve stability and reach quicker convergence. The structure of the entire component is detailed below:

```
DuelingQNetwork(
  (actionValueNet): Sequential(
    (act_fc1): Linear(in_features=37, out_features=512, bias=True)
    (act_fc1_act): ReLU()
    (act_fc1_drp): Dropout(p=0.2)
    (act_final): Linear(in_features=512, out_features=4, bias=True)
  )
  (stateValueNet): Sequential(
    (state_fc1): Linear(in_features=37, out_features=512, bias=True)
    (state_fc1_act): ReLU()
    (state_fc1_drp): Dropout(p=0.2)
    (state_final): Linear(in_features=512, out_features=1, bias=True)
  )
)
```

It is important to note that the state-value as predicted is a single value, but the action advantage values are a vector of length equal to the size of the action space, as each action at a given state would have a different value.

2.0 Model Performance

The model was able to solve the Banana Collector environment in 552 training episodes, as reflected by an average score of 13 at the 652nd episode.



The mean in the above diagram is the average score as given by the rewards for the last 100 training episodes.

3.0 Ideas for future improvement

Some ways to improve this model are:

1. Using a prioritized replay buffer to get to faster convergence i.e. using fewer episodes, ultimately allowing for a higher score.
2. Implement noisy DQN include stochasticity in the agent's policy, allowing for more exploration.
3. Use a Double DQN and possible make the dueling networks deeper to improve convergence and performance.