Continuous Control Deep Q Learning Task

Approach taken

As this task had a continuous action space, deep Q learning was not a suitable choice, as it does not work well in this area.

An alternative, which is proven to work well with continuous action spaces, is deterministic deep policy gradient, or DDPG. DDPG is an actor critic method, which trains two neural networks - an actor and a critic. The actor is responsible for providing the actions taken by the agent - it maps from states to actions. The critic is responsible for evaluating the Q function - giving a score to state, action pairs proposed by the actor. By using the Q score from the critic as a reward function, the actor can be trained.

Both neural networks used two fully connected hidden layers, with ReLU activation between the hidden layers. The actor used tanh activation in the output layer.

Hyperparameters

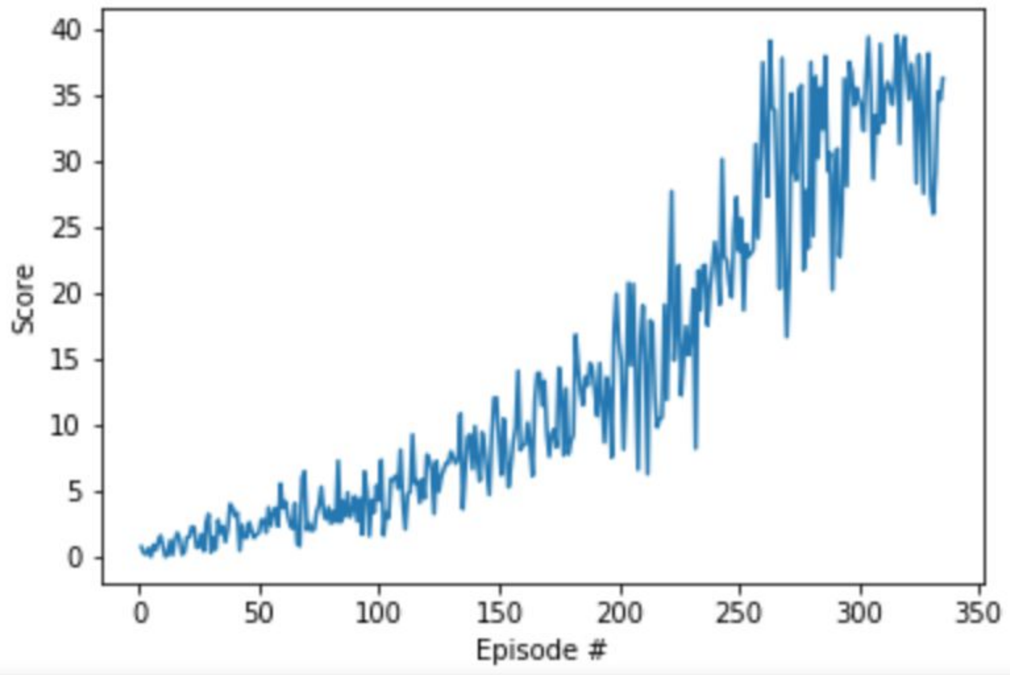
- Buffer size (number of items held in replay buffer) = 100000
- Batch size = 128
- Gamma = 0.99
- Tau = 0.001
- Learning rate = 0.0001 for both neural networks

Results

The agent described above was trained for 350 iterations (this took several hours on CPU). The score over time is shown in the graph below. The model took around iterations to start learning rapidly, but this

Footnote: there is a small bug in the implementation. When reporting scores for good runs (those with an average score of the past 100 runs of 14 of more), the notebook prints off an index for the iteration 100 less than the correct value. This is visually jarring, and incorrect, but has no impact beyond that.

was followed by a rapid uptick, as it rapidly started improving its performance, taking around 350 iterations to get a running average (over the past 100 episodes) of 30. By the end of training, it was consistently hitting scores close to 40.



Next steps

- Hyperparameter estimation with this setup was quite difficult, with more computational power available, something like grid search with multiple parallel runs would make this efficient, and would likely lead to better hyperparameters being chosen
- Due to computation limits, a small fully connected neural network was used. A larger network may be able to perform better on the task.
- The model operated over a small state space as it knew the inner mechanics of the world. It would be interesting to train a model from pixel information using a convolutional network.

Footnote: there is a small bug in the implementation. When reporting scores for good runs (those with an average score of the past 100 runs of 14 of more), the notebook prints off an index for the iteration 100 less than the correct value. This is visually jarring, and incorrect, but has no impact beyond that.