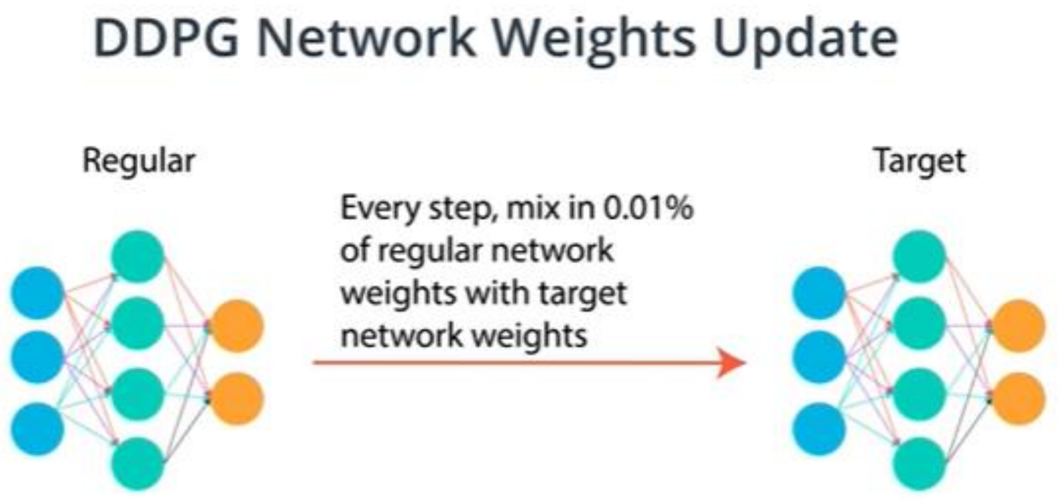
Learning Algorithm

The learning algorithm uses a deep deterministic policy gradient (DDPG) method to solve the environment with continuous control. DDPG is an actor-critic method used to reduce variance when training the neural network. The learning algorithm uses a replay buffer and a soft update function shown in ddpg_agent.py.

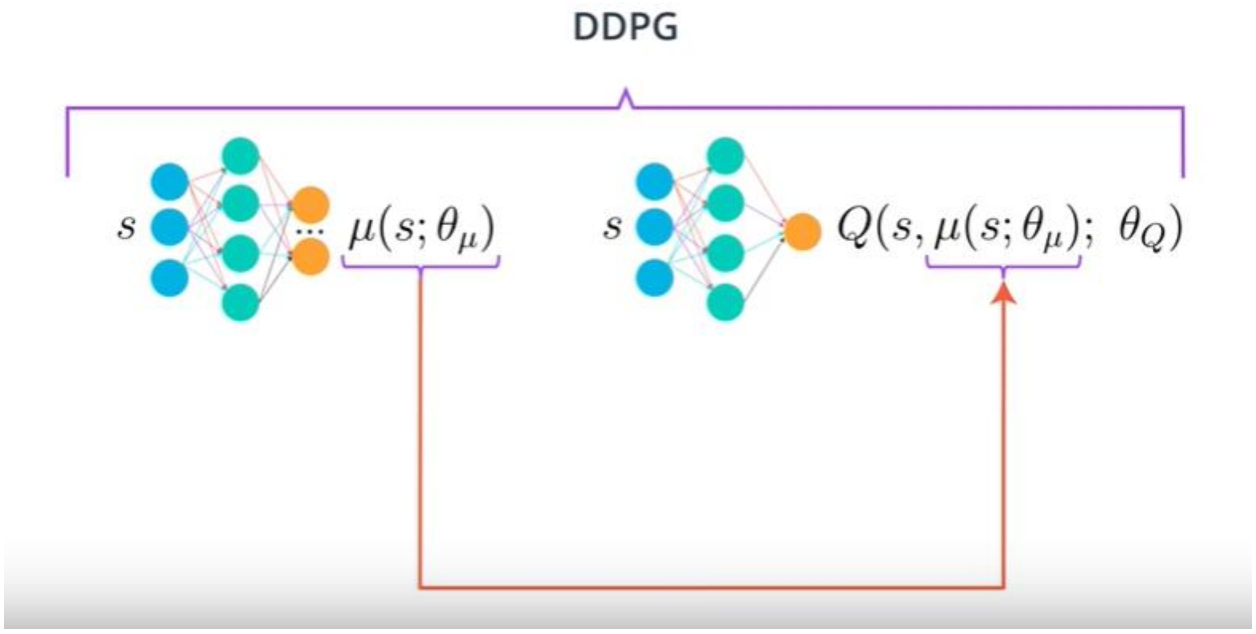

DDPG Network Weights Update
Regular
Target
Every step, mix in 0.01% of regular network weights with target network weights

Hyperparameters

Buffer Size	1000000
Batch Size	128
Gamma	0.99
Tau	0.001
Actor Learning Rate	0.001
Critic Learning Rate	0.001
L2 Weight Decay	0
Learning Timestep Interval	20
Number of Learning Passes	10
Ornstein-Uhlenbeck Noise Parameter Sigma	0.2
Ornstein-Uhlenbeck Noise Parameter Theta	0.15
Epsilon	1
Epsilon Decay	0.000001

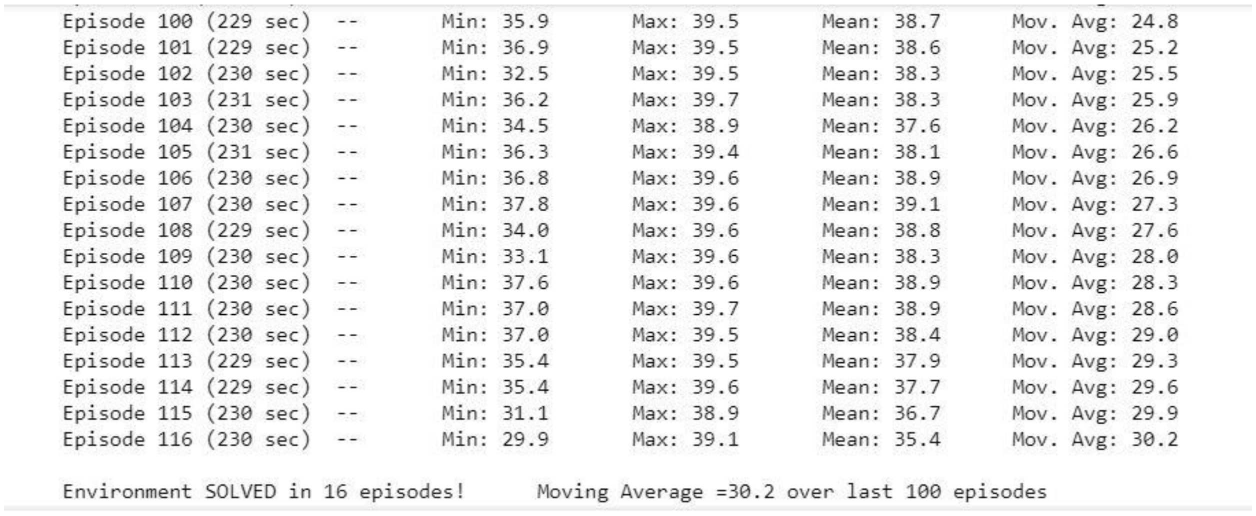
Model Architecture

The model architecture consist of an neural network for each the actor and the critic. Defined in model.py, the actor neural network consists of two ReLu layer and outputs a tanh layer for continuous control. The critic neural network consists a two ReLu layers concatenated to output the Q value for a given state space input. The input state space has 33 nodes and the output layer has 4 nodes.

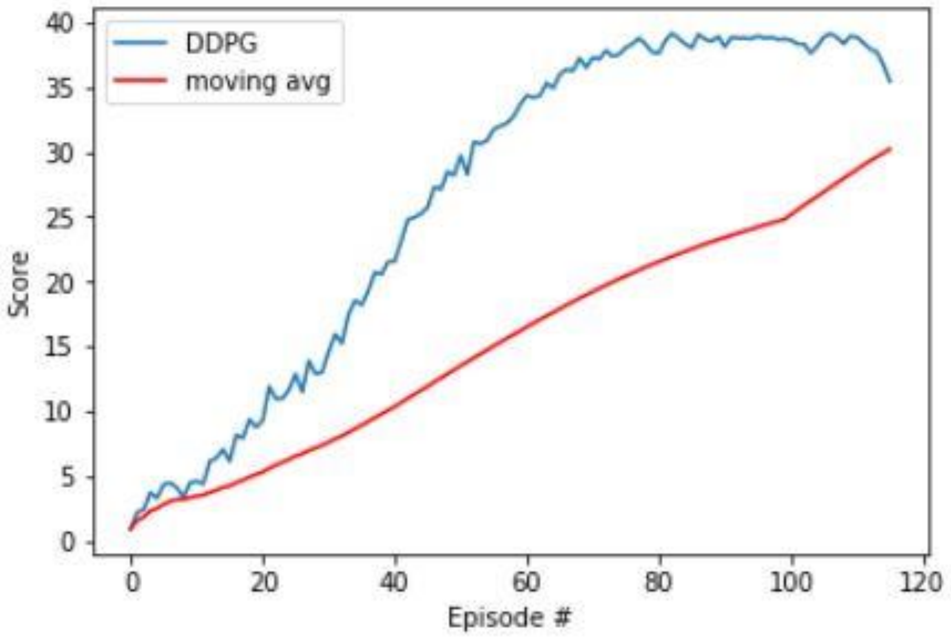


Plot of Rewards

The network reached a moving average score of 30.2 after 116 episodes of training.



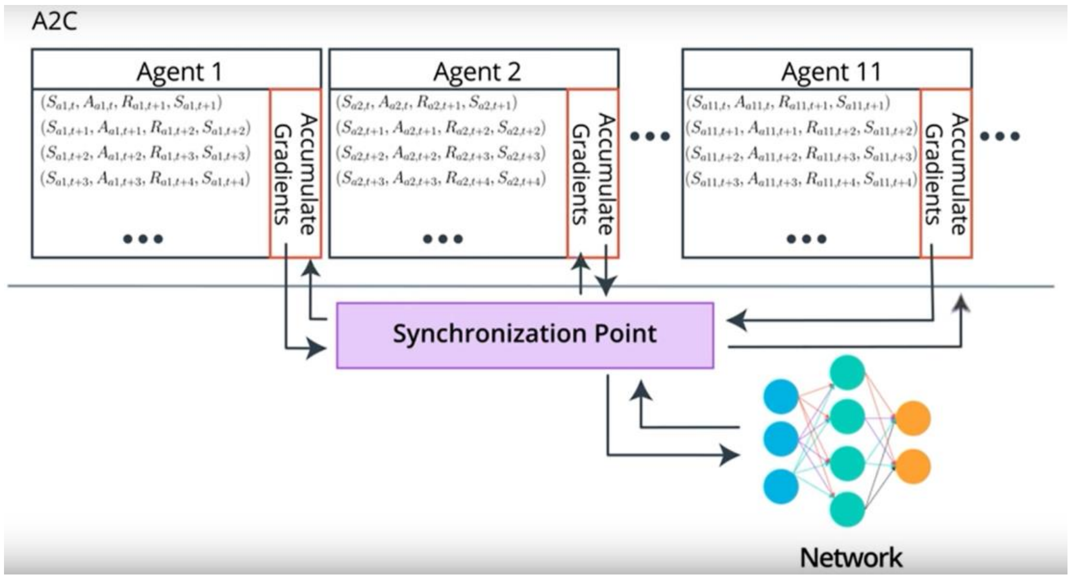
```
Episode 100 (229 sec)  --      Min: 35.9      Max: 39.5      Mean: 38.7      Mov. Avg: 24.8
Episode 101 (229 sec)  --      Min: 36.9      Max: 39.5      Mean: 38.6      Mov. Avg: 25.2
Episode 102 (230 sec)  --      Min: 32.5      Max: 39.5      Mean: 38.3      Mov. Avg: 25.5
Episode 103 (231 sec)  --      Min: 36.2      Max: 39.7      Mean: 38.3      Mov. Avg: 25.9
Episode 104 (230 sec)  --      Min: 34.5      Max: 38.9      Mean: 37.6      Mov. Avg: 26.2
Episode 105 (231 sec)  --      Min: 36.3      Max: 39.4      Mean: 38.1      Mov. Avg: 26.6
Episode 106 (230 sec)  --      Min: 36.8      Max: 39.6      Mean: 38.9      Mov. Avg: 26.9
Episode 107 (230 sec)  --      Min: 37.8      Max: 39.6      Mean: 39.1      Mov. Avg: 27.3
Episode 108 (229 sec)  --      Min: 34.0      Max: 39.6      Mean: 38.8      Mov. Avg: 27.6
Episode 109 (230 sec)  --      Min: 33.1      Max: 39.6      Mean: 38.3      Mov. Avg: 28.0
Episode 110 (230 sec)  --      Min: 37.6      Max: 39.6      Mean: 38.9      Mov. Avg: 28.3
Episode 111 (230 sec)  --      Min: 37.0      Max: 39.7      Mean: 38.9      Mov. Avg: 28.6
Episode 112 (230 sec)  --      Min: 37.0      Max: 39.5      Mean: 38.4      Mov. Avg: 29.0
Episode 113 (229 sec)  --      Min: 35.4      Max: 39.5      Mean: 37.9      Mov. Avg: 29.3
Episode 114 (229 sec)  --      Min: 35.4      Max: 39.6      Mean: 37.7      Mov. Avg: 29.6
Episode 115 (230 sec)  --      Min: 31.1      Max: 38.9      Mean: 36.7      Mov. Avg: 29.9
Episode 116 (230 sec)  --      Min: 29.9      Max: 39.1      Mean: 35.4      Mov. Avg: 30.2

Environment SOLVED in 16 episodes!     Moving Average =30.2 over last 100 episodes
```



Ideas for Future Work

A2C Algorithm

Implementation of the A2C algorithm can be used compare the training results with DDPG in the GPU.



PPO Algorithm

The PPO algorithm could be used to obtain the optimal policy to train multiple objects with non-interacting parallel copies of the same agent. This would be useful for training the 20 controllers in version 2 of this project.