Continuous Control

The purpose of the continuous control project was to train an agent to maintain a double-jointed arm position at a target location for as many time steps as possible. After much frustration attempting PPO and DDPG implementations using the 20-agent environment, a DDPG approach was used on the single agent environment. The trained agent successfully reached the goal of achieving an average of 30 points on the environment after training for ~600 episodes (600,000 timesteps total). Agents were trained using seed 0 for both agent and environment and evaluated for 100 episodes using seed 5 for agent and environment respectively. An example of agent performance during training and evaluation are provided below (95% confidence interval shown in light red).

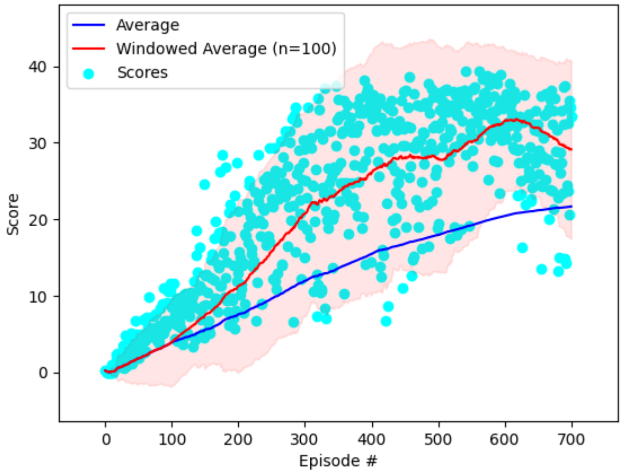

Figure 1 – DDPG Training

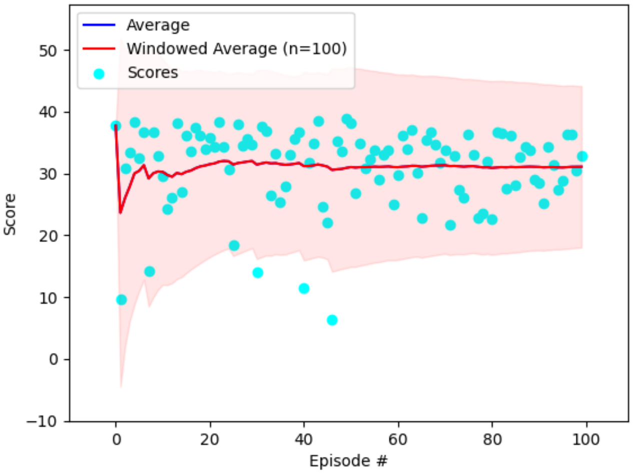

Figure 2 – DDPG Evaluation

Deep Deterministic Policy Gradient (DDPG) Agent

The DDPG agent used was based off the DDPG bipedal solution provided in the Udacity Deep Reinforcement Learning repository. Additional code was added to provide an early stop condition if windowed average performance performed worse than its previous run as well as code to save the best performing model during training. Further details about the agent and associated Actor-Critic model are provided in the sections below.

DDPG Model Architecture

A straightforward multilayer perceptron approach was used for both DDPG Actor & Critic local and target networks. Rectified Linear Unit (ReLu) activation functions were used for Actor layers except for final output which used a hyperbolic tangent (tanh). Leaky ReLu activation functions were used for all Critic layers except for final output. A summary of the network architecture is provided below.

Actor Local & Target Networks

```
Actor(
  (fc1): Linear(in_features=33, out_features=400, bias=True)
  (fc2): Linear(in_features=400, out_features=300, bias=True)
  (fc3): Linear(in_features=300, out_features=4, bias=True)
)
```

Critic Local & Target Networks

```
Critic(
  (fcs1): Linear(in_features=33, out_features=128, bias=True)
  (fc2): Linear(in_features=132, out_features=256, bias=True)
  (fc3): Linear(in_features=256, out_features=1, bias=True)
)
```

Agent Hyperparameters

Default values provided from the DDPG bipedal example were modified until an agent successfully passed the 30 goal. A summary of agent training parameters is provided below.

Table 1 - DDPG Agent Training Parameters

Buffer Size: 1000000	*Batch Size: 256*	*Gamma: 0.99*	*Tau: 1e-3*	*LR_Actor: 5e-4*
LR_Critic: 5e-4	*Weight Decay: 0.0*	*Seed: 0*	*Optimizer: Adam*	

Future Work

There are many improvements that could potentially improve DDPG agent performance. Goals would be not only to increase averaged windowed score of DDPG agent but also to reduce standard deviation to demonstrate more stable agent performance between episodes (example below).

```
Episode 0      Average Score: 0.23 ± 0.00
Episode 100    Average Score: 3.96 ± 2.87
Episode 200    Average Score: 11.26 ± 5.61
Episode 300    Average Score: 20.74 ± 6.69
Episode 400    Average Score: 26.08 ± 7.88
Episode 500    Average Score: 27.96 ± 7.34
Episode 600    Average Score: 32.84 ± 4.78
Episode 700    Average Score: 29.15 ± 5.81
Early stop at 700/2000 episodes!
Average Score: 29.15 ± 5.81      Best Average Score: 32.84
```

A list of potential improvements, grouped by category, are provided below.

Agent Algorithm Modifications

- Prioritized experience replay would be first approach as it may be the easiest to implement by sampling with respect to memory error as distribution probability

Model Modifications

- Add dropout layers between fully connected Actor and Critic MLP layers to prevent overfitting and improve generalizability
- Investigate the use of batch normalization layers for both Actor & Critic networks

Hyperparameter Optimization

- Optimize hyperparameters by empirically testing various parameter combinations or using some Bayesian approach
- Use a learning rate scheduler to decrease learning rate throughout training to promote the discovery of better performing policies