Continuous Control

Learning Algorithm

- The code for DDPG was implemented with the help of udacity lecture material
- Also referred to ShagtongZhang DRL library https://github.com/ShangtongZhang/DeepRL to dubug my model

The following is the alrorithm used from the paper https://arxiv.org/pdf/1509.02971.pdf

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$

$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

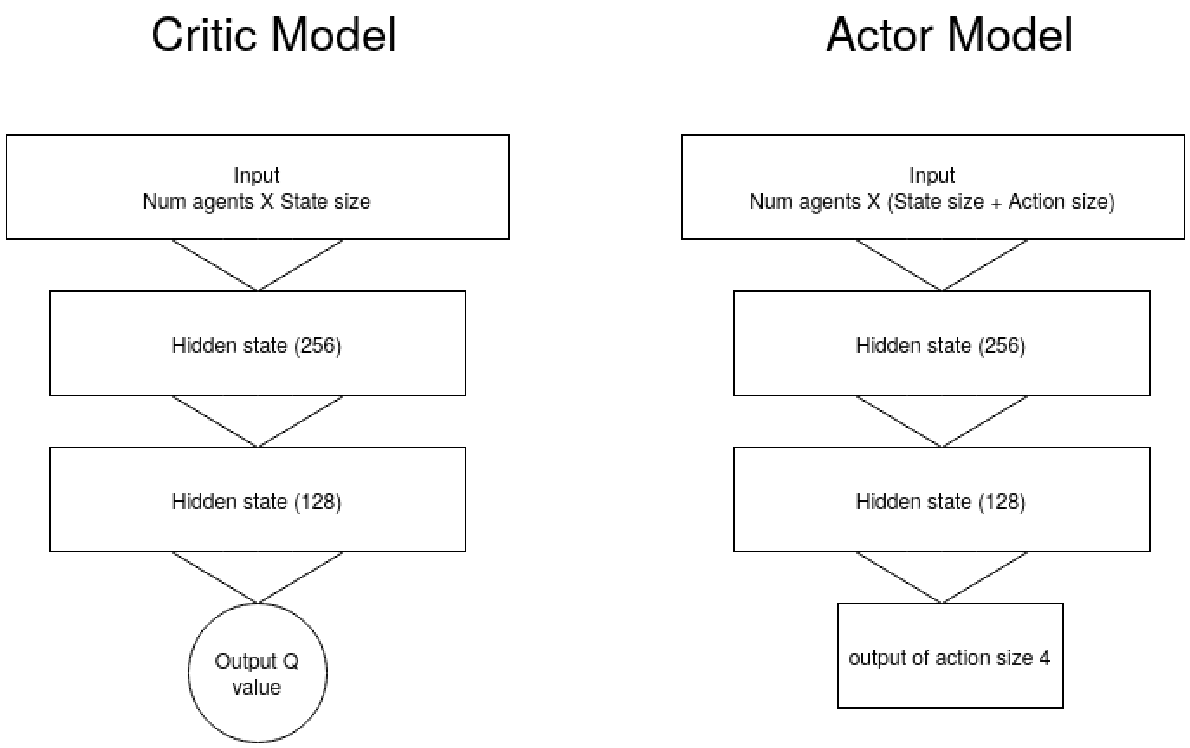
  end for
end for

- Noise added to the deterministic actor was done using Ornstein–Uhlenbeck Noise
- Store the transition tuples in the replay buffer
- The model requires minimizing the Mean squared error between target and approximate Q value
- and maximizing the expected Q value while take deterministic actions using gradient ascent
- Soft update the models using TAU

Chosen parameters and model architectures

Parameters	value
Replay buffer size	1e5
batch size for models	256
GAMMA	0.99
TAU (soft update)	1e-3
learning rate	1e-4
Update every	2

Models used for agents



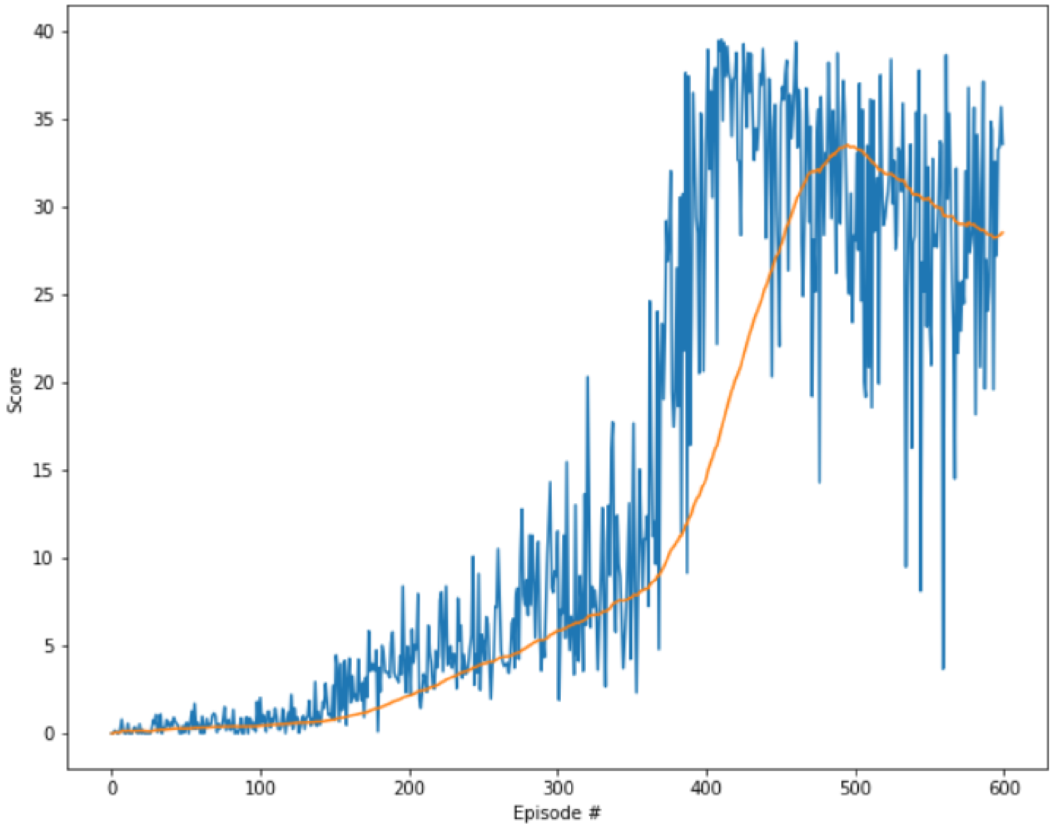
For Actor

```
Actor Net(
  (fc1): Linear(in_features=33, out_features=256, bias=True)
  (fc2): Linear(in_features=256, out_features=128, bias=True)
  (fc3): Linear(in_features=128, out_features=4, bias=True)
)
```

For Critic

```
Critic Net(
  (fc1): Linear(in_features=33, out_features=256, bias=True)
  (fc2): Linear(in_features=256, out_features=128, bias=True)
  (fc3): Linear(in_features=128, out_features=1, bias=True)
)
```

Plots of Rewards



Environment was solved by 460 episodes! With an average score of 30

Future Works

- Continue with exploring PPO, D4PG and A3C models for the agents
- Solve the Reacher 20 option, I did try to solve it but it was extremely slow.
- Try solving the Crawler environment and also explore the agents in Gazebo
- Play around with different model architectures