Project # 2 Continuous Control

I chose to solve the 20-agent version of the Reacher environment. For my solution, I used the DDPG algorithm since the action space in continuous in the interval [-1, 1]. The solution reached an average performance greater than 30 at episode 100.

The algorithm: DDPG is similar to DQN in the sense that it uses a replay buffer to train an action, and a target network to stabilize training. Also, DDPG is a deterministic policy, thus, it uses he policy gradient method to minimize the cost function. The training is done online by sampling past 'experiences' from the replay buffer which are also continuously collected. For the solution, I initially tried sampling a mini-batch of experiences at each step to train the value function but the improvement was very slow and negligible. To solve this, the solution was to extract a mini-batch 5 consecutive times and do it every 20 steps.

DDPG Network Architecture: This is an actor-critic algorithm where the actor approximates the optimal policy deterministically and output the best believed action for a given states, the critic learns to evaluate the optimal action value function by using the actor's best believed action. The actor inputs are 33 values representing the observation space and it outputs an action containing 4 values representing the best action for that state. The critic also takes as input the 33 values for the observations space, but in this case, its first hidden layer is concatenated with the actor output layer and passed as input to the critic 2nd hidden layer. This way, the best believed action from the actor is also used in the critic to learn an optimal action-value function $Q(s, \mu(s; \phi); \theta)$.

Hyperparameters:

Actor First Hidden Layer: 512
Actor Second Hidden Layer: 256
Critic First Hidden Layer: 512
Critic Second Hidden Layer: 256
BUFFER_SIZE = 1.e6
BATCH_SIZE = 128
GAMMA = 0.99
TAU = 1e-3
EPS = 0
EPS_DECAY = 0
LR_ACTOR = 1e-3
LR_CRITIC = 1e-3
WEIGHT_DECAY = 0

Future things to try: Initially, I tried different hyperparameters with no success. Some of the other algorithms I considered trying were Twin Delayed DDPG (TD3) which is an algorithm based on improvements, and also the PPO algorithm. Now with the results from DDPG I'd like to try these other 2 and compare performances. Other things to try are reducing the number of units in the hidden networks and see if it speeds up learning, increase the batch_size in replay buffer and decrease the rate of learning from experiences, which should also speed up the process, and finally change the noise process (Ornstein-Uhlenbeck) to Gaussian noise since some literature indicates that either one is effective.