DDPG Method for Solving Udacity's Continuous Control Problem-20 Agents

1)DDPG Algorithm

We implemented an *off-policy method* called Deep Deterministic Policy Gradient (DDPG) described in the paper *Continuous control with deep reinforcement learning* (https://arxiv.org/abs/1509.02971).

DDPG learns a Q-function and a policy in training. It uses off-policy data and the Bellman equation to learn the Q-function, and uses the Q-function to learn the policy.

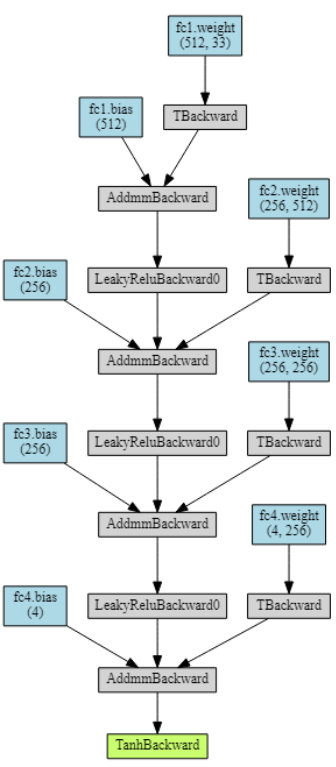
The DDP folder holds 2 '.py' files that are then called in the Continuous_Control-20Agents-Final.ipynb notebook:

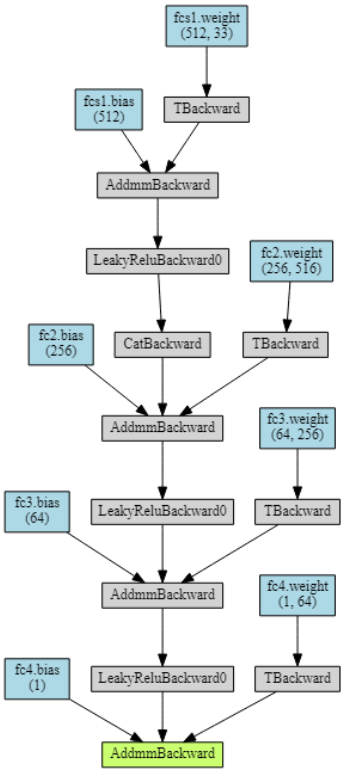
-model.py : Implement the Actor and the Critic class:
Both Actor and Critic class implement a Target and a Local Neural Network for training.

-ddpg_agent.py : Implement the DDPG agent, a Noise and a Replay Buffer class:
The Actor's Local and Target neural networks, and the Critic's Local and Target neural networks, the Noise process and the Replay Buffer are inside by the Agent's constructor and reset independently for each agent. The Noise uses OU process. The Replay Buffer is a fixed-size buffer to store experience tuples.

2) Model Used

The **Actor Neural Network** The **Critic Neural Network**





A few notes:

-There was a need to go deeper in both networks. Learning was not happening without the additional layers from the original proposed networks.

-The use of Leaky_Relu helps in the continuous control that can take negative values.

3) Agent Used

Here are the Hyper parameters used.

```
BUFFER_SIZE=int(1e5)
# replay buffer size
                      BATCH_SIZE = 128     # minibatch size
                      GAMMA = 0.99        # discount factor
                      TAU = 0.001             # for soft update of target parameters
                      LR_ACTOR = 1e-4         # learning rate of the actor
                      LR_CRITIC = 1e-3        # learning rate of the critic
                      WEIGHT_DECAY = 1e-5      # L2 weight decay
```

Two conditions where added on the loop:

-A maximum number of steps per episode

-A break on the number of ‘dones’ was added.

4) Thoughts for Improvement and Future Work

Here are some ideas for improving the agent's performance, among others.

We could implement other agents, among which:

D4PG, PPO, TD3, SAC, PlanNet and Dreamer.

We could also modify the memory class as seen in the previous project by adding Prioritized Experience Replay and Hindsight Experience Replay.