Project three of deep reinforcement learning from Udacity.

Learning Algorithm:

This project leverages a lot of code from the previous project Continuous control. There will be two agents and Deep Deterministic Policy Gradient algorithm is used with Actor Critic network for each Agent. The general algorithms remain the same except that Batch normalization is adding into both Actor and Critic network this time.

Since we have two agents and they need to learn together, I implemented a new function call **share_learning** within the **Agent** class in the ddpg_agent.py file. This function will force them to share the neural network model weighting for both actor and critic. In additional, I also want them to share the experience; therefore, they both use the same ReplayBuffer. This is like having two Actor Critic agent with different states at any given time but exchange knowledges and experiences all time. That is especially true when we want them to get as much rewards as possible as a team.

Model architectures:

Both Actor and Critic network have two hidden layers where each layer has 128 units. There is a batch normalization layer between each hidden layers to speed up the time to converge.

Actor architectures:
Input layer
Batch normalization
Fully connected layer
ReLU layer
Batch normalization
Fully connected layer
ReLU layer
Batch normalization
Fully Connected
tanh layer

Critic architectures:
Input layer
Batch normalization
Fully connected layer
ReLU layer + Action
Concatenation for the ReLU and Action
Fully connected layer
ReLU layer
Fully Connected

Hyperparameters:

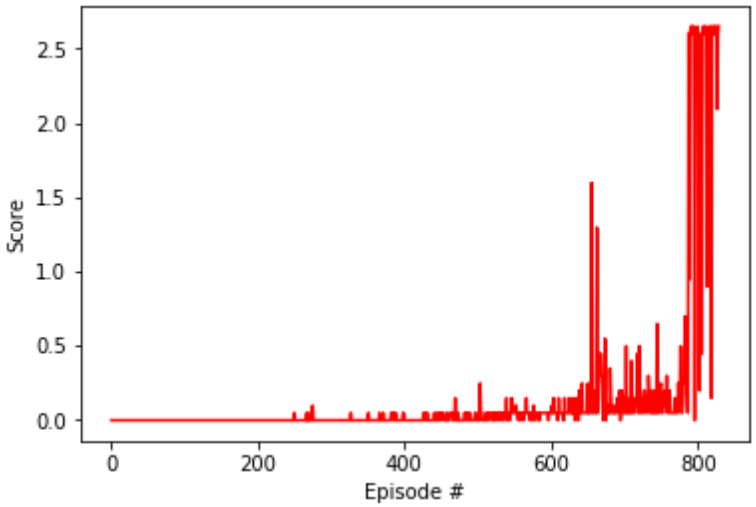
The following are the hyper-parameter that I used.

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

```
WEIGHT_DECAY = 0 # L2 weight decay
```

Plot of Rewards:

With the parameters above, the entire training took exactly at 828 episodes to complete.
Future Enhancements



Possible Future work:

I have experienced that different seek value can sometime enhance the overall performance. This is similar to the second project (Continuous Control) that we did last time. In addition, algorithms like PPO, A3C, and D4PG that use multiple (non-interacting, parallel) copies of the same agent to distribute the task of gathering experience. Another thing that we can do is try a slightly different neural network with different number of hidden layers. That may help coverage faster.