## 1. Learning Algorithm:

The source code and weights of a trained agent are placed in the main directory. The algorithm is implemented following these guidelines:

- The reinforcement learning algorithm is the DDPG (Deep Deterministic Policy Gradient)

- Neural Network configuration:

  Actor Network (x2):
  - Hidden layers are composed of -> 256 -> ReLU -> action_size -> tanh
  - The optimizer is ADAM and
  - The learning rate = 1e-4

  Critic Network(x2):
  - Hidden layers are composed of -> 256 -> LeakyReLU -> 256 -> LeakyReLU -> 128 -> LeakyReLU -> 1
  - The optimizer is ADAM and
  - The learning rate = 3e-4

- Other algorithm configuration:

  - BATCH_SIZE = 1024 # minibatch size to be processed by the algorithm
  - GAMMA = 0.99 # discount factor for future expected rewards
  - update_steps_actor=4 # update cycle for the actor/critic network (x in a row, without update, x in a row with update)
  - update_steps_critic=10
  - TAU = 1 # for soft update of target parameters TAU is the rate at which this update takes place so that the network weights do not change suddenly but smoothly.

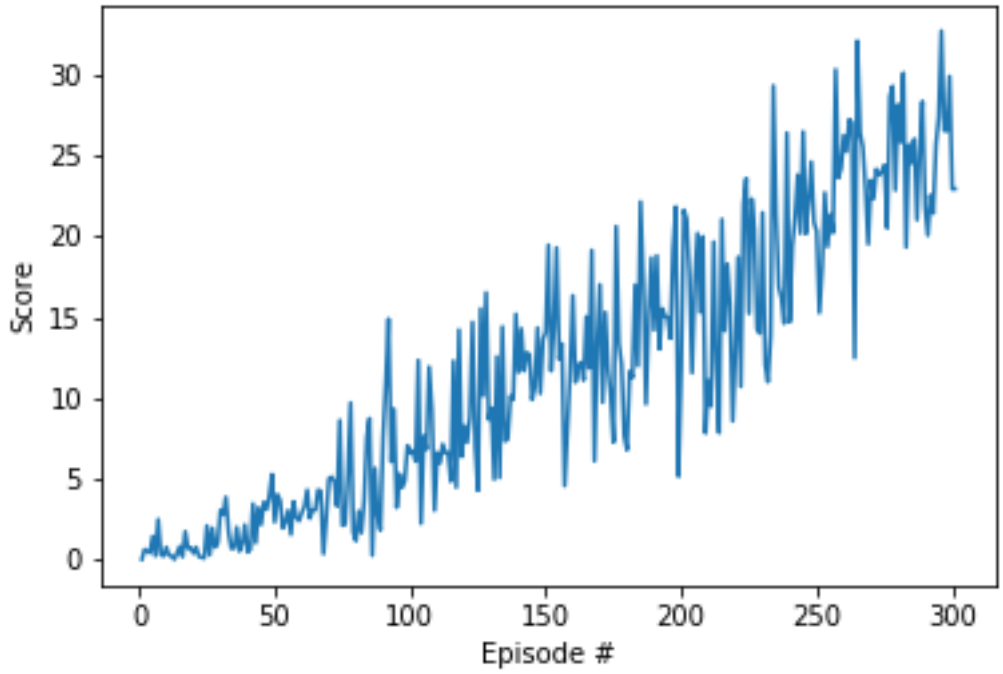
- Experience Replay is also implemented. In this technique, DDPG model is trained by mini-batch from a replay buffer with a size 1e6

- Agent selects next action based on the policy learned by the actor

## 2. Agent Results:

The following plot of rewards shows the training results. At Episode 300 the algorithm stopped. The agent performance would have met the criteria and stopped training after around 150-200 episodes more. (mean scores of last 100 episodes is above +30).

Though it took so many time to train, and it is obvious that this algorithm solves the problem.




3. **Future Work:**

   An improvement to be done is to implement multiple agents.

   On that solution some tuning on the learning rates and update steps might be done as well.