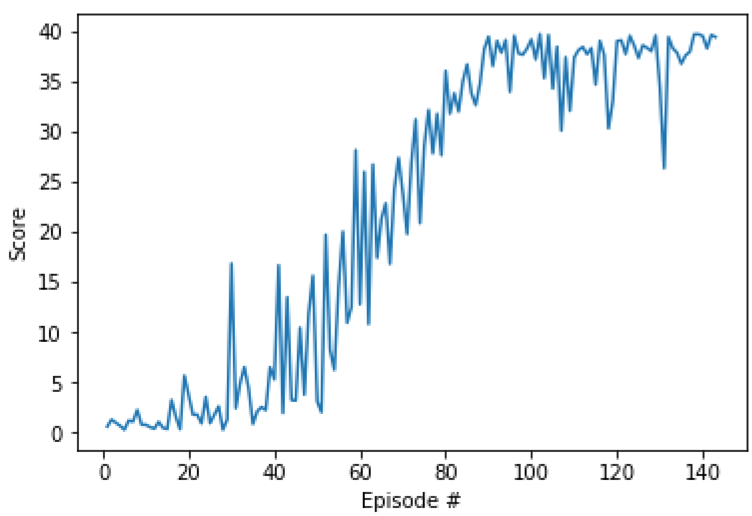
Learning algorithm
I used the DDPG learning algorithm based on the [Continous control with deep reinforcement learning](https://arxiv.org/pdf/1509.02971.pdf) paper (Lillicrap et al., 2016). The starting code was based on the Bipendulum implementation introduced during the class.

DDPG is an actor-critic method where the Critic learns from the value function and it determines how the Actor policy improves. To decrease the instability of the model, I used a replay buffer and a soft target update.

Model architecture
The actor model is a neural network with three hidden layer with size 256, 128 and 64. I used ReLU as the activation function and tanh is used in the final layer to return the action output.

The critic model is a neural network with four hidden layers of size 256, 256, 128 and 64. I used ReLU as the activation function and tanh is used in the final layer to return the action output.

Training plots
I solved the environment in 150 episodes. I used Udacity's GPU and it took me around 8-10 hours to solve the environment.



Here it shows that 30.12 rate is achieved.

training
The training done on 150 episodes and plotted as shown in the html file.

Ideas for future work
As introduced in the [Benchmarking Deep Reinforcement Learning for Continuous Control](https://arxiv.org/pdf/1604.06778.pdf) Truncated Natural Policy Gradient (TNPG) and Trust Region Policy Optimization (TRPO) (Schulman et al., 2015) should improve the learning speed of the algorithm. In addition to that, I could use the simulator with 20 agents to speed up learning.