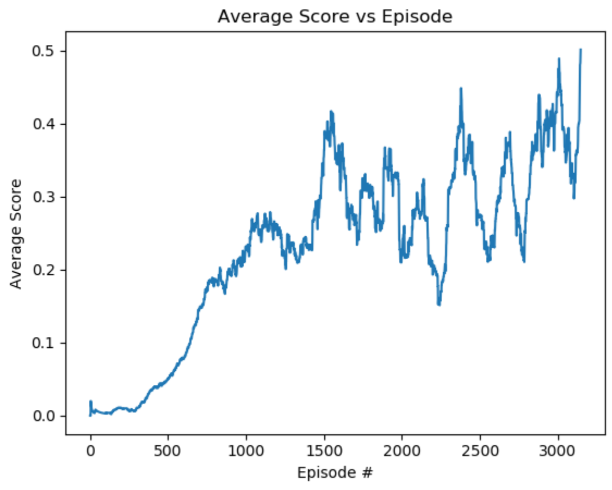
Report

This project uses Multi-Agent Deep Deterministic Policy Gradient with the following hyperparameters to solve the environment:

buffersize = 5000	#replay buffer size
batchsize = 500	#minibatch size
discount_factor = 0.95	#discount factor
tau=0.02	#for soft update of target parameters
lr_actor = 1.0e-4	#Learning rate for Actors
lr_critic = 1.0e-4	#Learning rate for Critic
noise_reduction = 0.9999	#Noise reduction rate (after every action)

MADDPG uses actor-critic method, where the critic model learns the value function and uses it to determine how the actors should change. To make it feasible to train multiple agents that can act in a globally coordinated way, critics can access the observations and actions both agents. Since the policy is deterministic, to explore the environment, it uses the Ornstein–Uhlenbeck process to introduce noise. As the episodes progress, the agent will explore less, this is achieved by introducing noise reduction to the Ornstein–Uhlenbeck process. MADDPG also uses replay buffer to store the experience of each step, then sample mini batches of experience to update the actor and critic. Both actors have the same architecture, 2 hidden layers with 256 nodes for the first hidden layer and 128 nodes for the second hidden layer. The critic has 4 hidden layers, with the first layer having 512 node, layer 2 and 3 has 256 nodes and last hidden layer with 128 nodes. The actor and critic target network are updated using soft updates strategy to provide faster convergence and better stability. Introducing batch normalization to the input neural network showed better performance.

With aforementioned hyperparameters and architecture, the agents was able achieve an average score, over 100 episodes, of 0.5 in 3147 episodes, which took 8 hours to run using CPU.



In the future, I have 2 plans to improve this project. First is to change the values of the hyperparameters to see how it affects the training speed. Particularly the buffer size, batch size and noise reduction rate. I plan to let the agent explore in the first few hundred episodes and then disable it after that, instead of gradually reducing it. It might not produce the optimal result, in terms of training time, but I am interested in seeing the results. The second plan is to is reduce the size of the neural network architecture. I am expecting this change will speed up the training.