

Project 3

# Collaboration and Competition

---

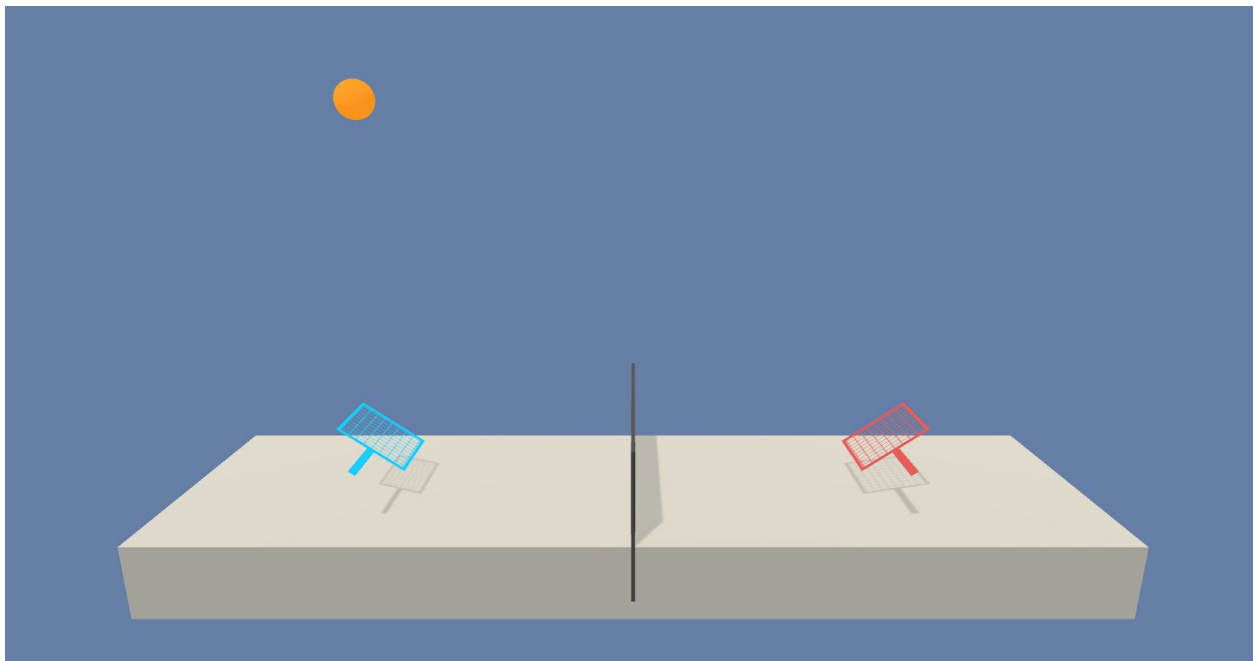
Tri B. Minh

19th May, 2020

# Project 3: Collaboration and Competition


## 1. The Environment

For this project, you will work with the [Tennis](#) environment.



Unity ML-Agents Tennis Environment

In this environment, two agents are playing tennis. If an agent hits the ball over the net, it receives a **reward of +0.1**. If an agent lets a ball hit the ground or hits the ball out of bounds, it receives a **reward of -0.01**. Thus, the goal of each agent is to keep the ball in play.



The observation space consists of 8 variables corresponding to the position and velocity of the ball and racket. Each agent receives its own, local observation. Two continuous actions are available, corresponding to movement toward (or away from) the net, and jumping.

Number of agents: 2

Size of each action: 2

There are 2 agents. Each observes a state with length: 24

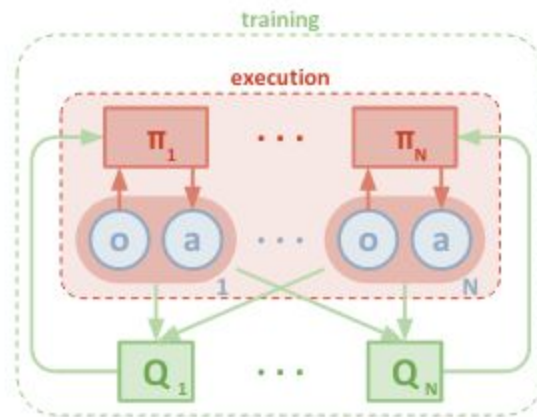
The task is episodic, and in order to solve the environment, your agents must get an average score of +0.5 (over 100 consecutive episodes, after taking the maximum over both agents). Specifically,

- After each episode, we add up the rewards that each agent received (without discounting), to get a score for each agent. This yields 2 (potentially different) scores. We then take the maximum of these 2 scores.
- This yields a single **score** for each episode.

**The environment is considered solved, when the average (over 100 episodes) of those scores is at least +0.5.**

## 2. Multi-Agent Deep Deterministic Policy Gradient (MADDPG)

The environment contains 2 agents who are playing tennis. The MADDPG Algorithms is a framework of centralized training with decentralized execution. The MADDPG allow multi-agents can learn to compete and collaborate each other.



**Fig1.** Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments Paper

In the diagram show the Overview of our multi-agent decentralized actor, centralized critic approach. Each Agent in the MADDPG is an “actor” and each actor get advice from a “critic” that help “actor” decide the actions during the training time. All the agents do not need to access the central critic at the test time, they action based on their observation combined with their predictions of other agents behaviors.

**Decentralized execution** is when each Agent has its own **private Actor network** and takes actions when observing the environment.

**Central Learning** is each agent's own private **Critic Network**, the Critic Network of each Agent has *full visibility* on the environment. It not only takes action and observation of a particular agent, it also takes all the observations and actions of all other agents. The output of the **Critic Network** is still the Q value estimated given a full observation input (all agents) and a full action input(all agents). The **Critic Network** is only active in the training time and disable in the running time.

**Experience Storage** is storing all states, actions, reward, next states, collected during interaction of environment.

**Explorations and noise decay** I use the [Ornstein–Uhlenbeck](#) seems to be the best algorithm to add noise. However, the common failure mode of the DDPG is the learner to overestimate Q-value then the policy breaking, the scores result get good at the beginning but at the end, this will reduce dramatically. Because it exploits the errors in the Q-function. So, That means, I add noise decay to the target action and make the agent hard to exploit the error in the Q-function by smoothing out Q along changes action.

### Multi-Agent Deep Deterministic Policy Gradient Algorithm

For completeness, we provide the MADDPG algorithm below.

---

**Algorithm 1:** Multi-Agent Deep Deterministic Policy Gradient for  $N$  agents

---

```

for episode = 1 to  $M$  do
  Initialize a random process  $\mathcal{N}$  for action exploration
  Receive initial state  $\mathbf{x}$ 
  for  $t = 1$  to max-episode-length do
    for each agent  $i$ , select action  $a_i = \mu_{\theta_i}(o_i) + \mathcal{N}_t$  w.r.t. the current policy and exploration
    Execute actions  $a = (a_1, \dots, a_N)$  and observe reward  $r$  and new state  $\mathbf{x}'$ 
    Store  $(\mathbf{x}, a, r, \mathbf{x}')$  in replay buffer  $\mathcal{D}$ 
     $\mathbf{x} \leftarrow \mathbf{x}'$ 
    for agent  $i = 1$  to  $N$  do
      Sample a random minibatch of  $S$  samples  $(\mathbf{x}^j, a^j, r^j, \mathbf{x}'^j)$  from  $\mathcal{D}$ 
      Set  $y^j = r_i^j + \gamma Q_i^{\mu'}(\mathbf{x}'^j, a_1^j, \dots, a_N^j)|_{a_k = \mu_k'(o_k^j)}$ 
      Update critic by minimizing the loss  $\mathcal{L}(\theta_i) = \frac{1}{S} \sum_j \left( y^j - Q_i^{\mu}(\mathbf{x}^j, a_1^j, \dots, a_N^j) \right)^2$ 
      Update actor using the sampled policy gradient:
        
$$\nabla_{\theta_i} J \approx \frac{1}{S} \sum_j \nabla_{\theta_i} \mu_i(o_i^j) \nabla_{a_i} Q_i^{\mu}(\mathbf{x}^j, a_1^j, \dots, a_i, \dots, a_N^j)|_{a_i = \mu_i(o_i^j)}$$

    end for
    Update target network parameters for each agent  $i$ :
      
$$\theta_i' \leftarrow \tau \theta_i + (1 - \tau) \theta_i'$$

  end for
end for

```

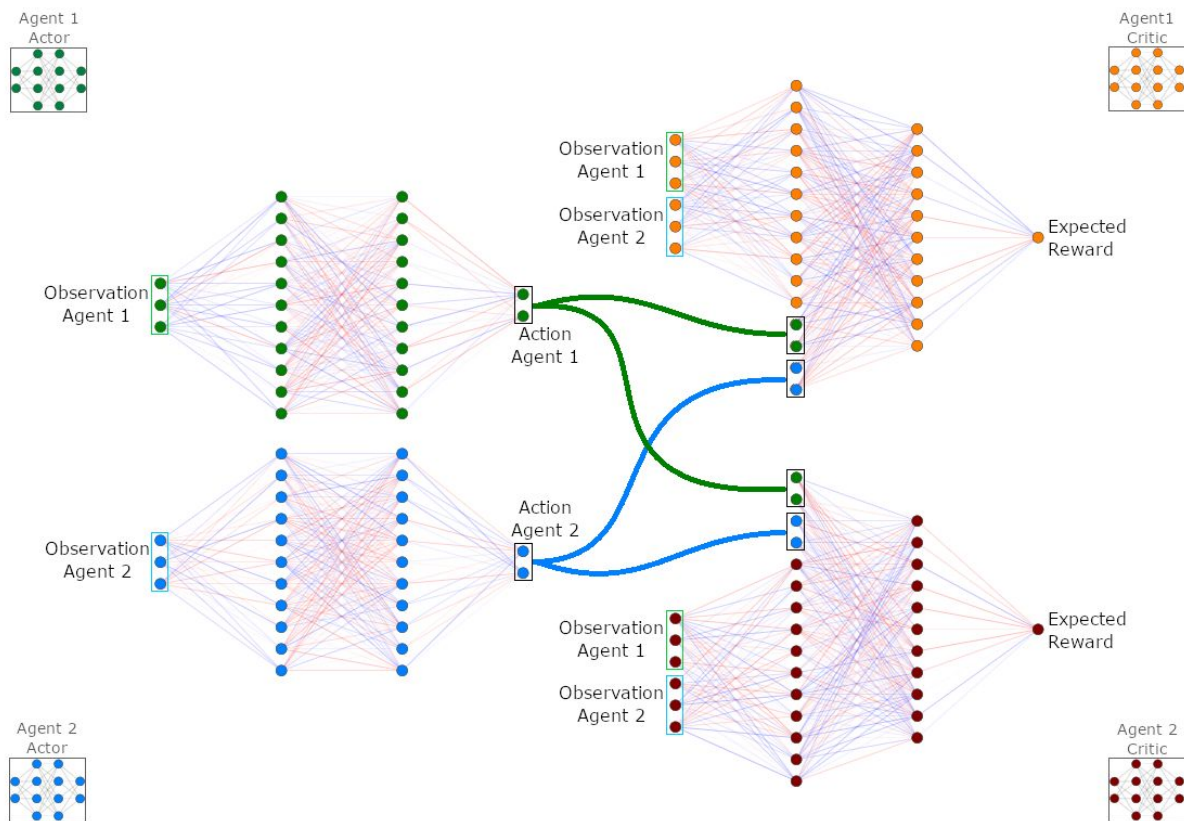
---

I have already changed the Model Structure of Actor and Critic with [fc1, fc2]=[400, 300], or [128, 64], [256, 256], the model not working very well, the scores return slowly and hard to reach the goal. After that, I try the Model Neural Network Structure with fc1, fc2]=[400, 300] and detailed parameters below.

1. State input (24 state\*2)
2. Hidden layer (256 units) with ReLU activation
3. Hidden layer (128 units) with ReLU activation
4. Action output (2 units) with tanh activation

The critic network maps state and action to Q value and has the following structure:

1. State input (24 state\*2)
2. Hidden layer 1 (256 nodes) with ReLU activation + Action input (2 units \*2 )
3. Hidden layer 2 (128 nodes) with ReLU activation
4. Q-value output (1 node)



### Discussion:

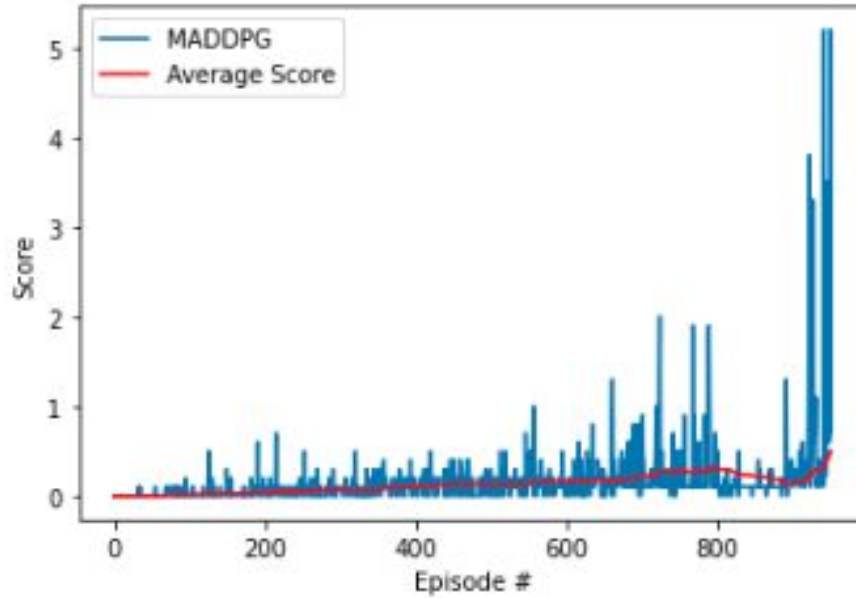
Some hyperparameters most effective to the DDPG (Deep Deterministic Policy Gradient) Algorithms are:

1. *Gamma (discount rate)* if the discount rate =0.99 and 0.98 the agent gets a good result .  
With all experiments below we can choose the **GAMMA =0.98** to get the best result.

2. The replay buffer\_size will affect the experiments of the agent . The best choose of ***BUFFER\_SIZE = 1e5*** , which is the agent will learning more stable than *BUFFER\_SIZE = 1e6*
3. The best option of the Learning Rate, ***LR\_ACTOR =1e-3*** and ***LR\_CRITIC=1e-3*** when increasing and decreasing the learning rate, the Agent can not get stable rewards.
4. Add the Noise decay in the OU noise. That means, OU noise will reduce over time. It has a good effect on the training model, so I add the EPS DECAY in 600 episodes, with initial EPS value =5.0 and EPS final value is 0.
5. **The best experiment can solved environment in 852 episodes** with Parameters following

BUFFER_SIZE	BATCH_SIZE	LR_ACTOR	LR_CRITIC	WEIGHT_DECAY	GAMMA
1e6	128	1e-3	1e-3	0	0.98

TAU	OU_SIGMA	OU_THETA	EPS_START	EPS_FINAL	EPISODE_END
10e-3	0.2	0.15	5.0	0	600



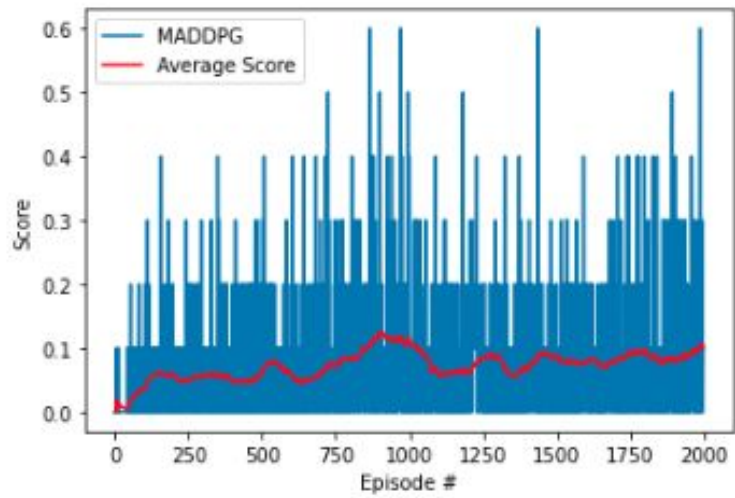
### 3. Experiments

Experiment 1. Decrease LR\_ACTOR and LR\_CRITIC

<b>BUFFER_SIZE</b>	BATCH_SIZE	<b>LR_ACTOR</b>	<b>LR_CRITIC</b>	WEIGHT_DECAY	GAMMA
<b>1e6</b>	128	<b>4e-4</b>	<b>4e-4</b>	0	0.98

TAU	OU_SIGMA	OU_THETA	EPS_START	EPS_FINAL	EPISODE_END
8e-3	0.2	0.15	5.0	0	300



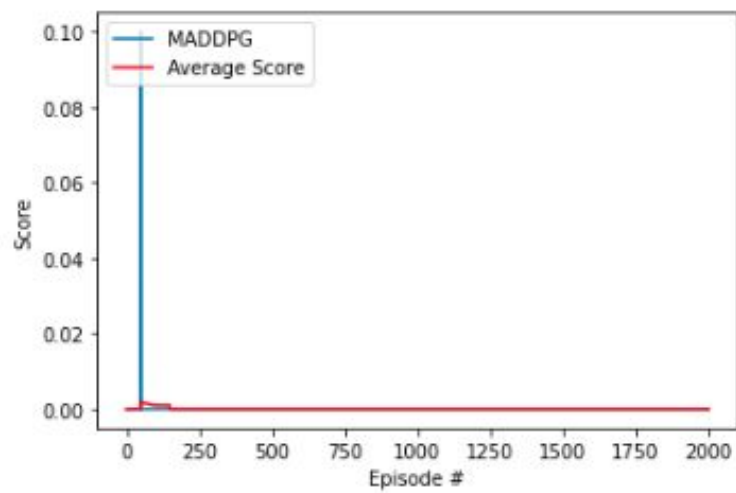


Environment **NOT** solved in 2000 episodes

## Experiment 2

<b>BUFFER_SIZE</b>	BATCH_SIZE	<b>LR_ACTOR</b>	<b>LR_CRITIC</b>	WEIGHT_DECAY	GAMMA
<b>1e5</b>	128	<b>3e-3</b>	<b>3e-3</b>	0	0.98

TAU	OU_SIGMA	OU_THETA	EPS_START	EPS_FINAL	EPISODE_END
8e-3	0.2	0.15	5.0	0	300

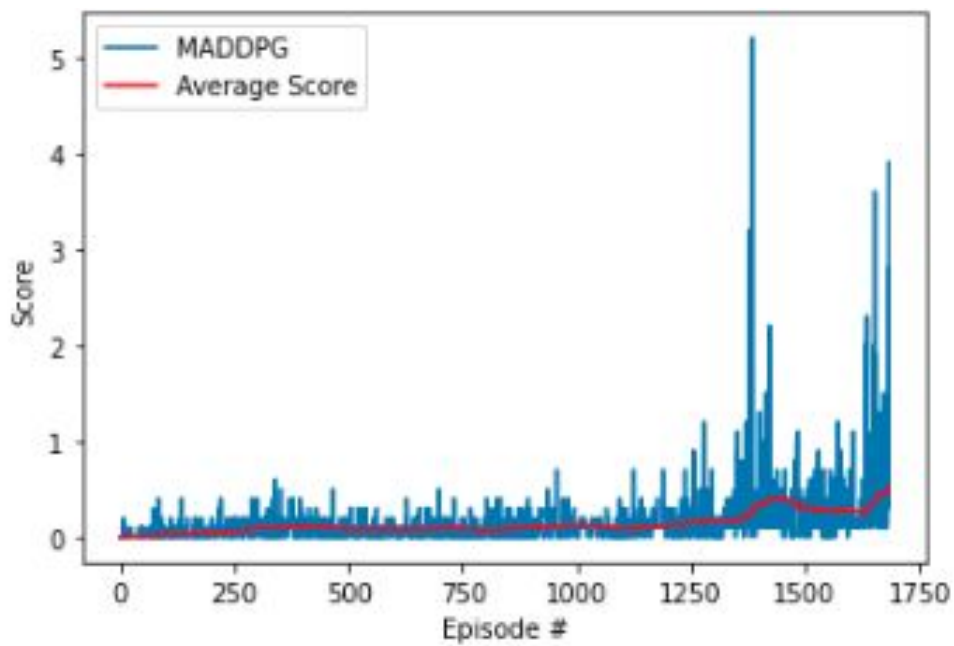


Environment **NOT** solved in 2000 episodes!

### Experiment 3.

BUFFER_SIZE	BATCH_SIZE	LR_ACTOR	LR_CRITIC	WEIGHT_DECAY	GAMMA
1e5	128	1e-3	1e-3	0	0.98

TAU	OU_SIGMA	OU_THETA	EPS_START	EPS_FINAL	EPISO_END
8e-3	0.2	0.15	5.0	0	300

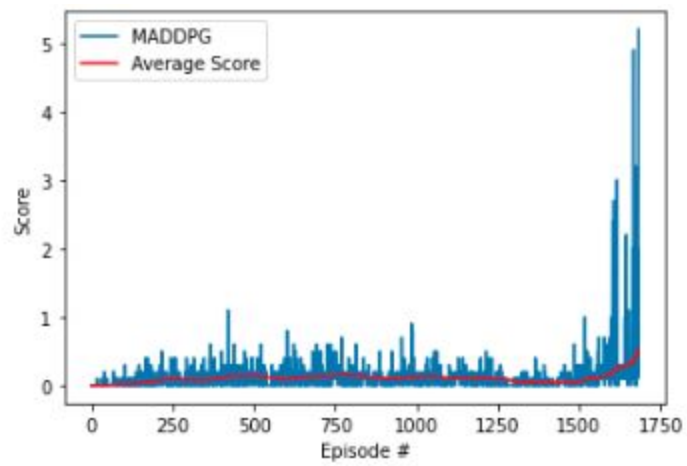


```
<-- Environment solved in 1585 episodes!
<-- Moving Average: 0.525 over past 100 episodes
```

#### Experiment 4

<b>BUFFER_SIZE</b>	BATCH_SIZE	<b>LR_ACTOR</b>	<b>LR_CRITIC</b>	WEIGHT_DECAY	GAMMA
<b>1e5</b>	128	<b>8e-4</b>	<b>8e-4</b>	0	0.98

TAU	OU_SIGMA	OU_THETA	EPS_START	EPS_FINAL	EPISODE_END
8e-3	0.2	0.15	5.0	0	300

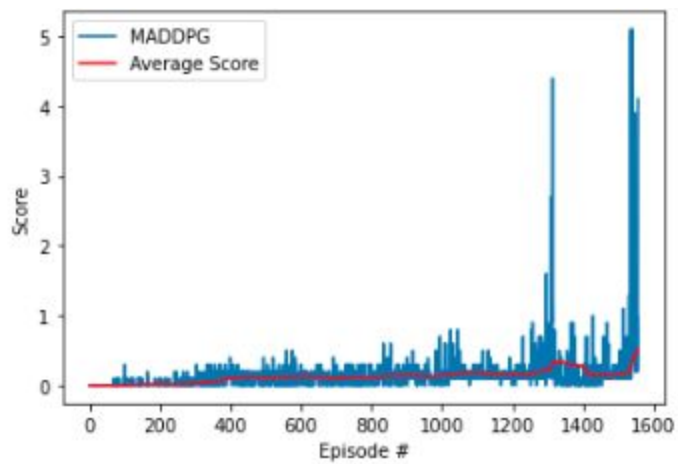


Environment solved in 1744 episodes!

### Experiment 5

<b>BUFFER_SIZE</b>	BATCH_SIZE	<b>LR_ACTOR</b>	<b>LR_CRITIC</b>	WEIGHT_DECAY	<b>GAMMA</b>
<b>1e5</b>	128	<b>1e-3</b>	<b>1e-3</b>	0	<b>0.99</b>

TAU	OU_SIGMA	OU_THETA	EPS_START	EPS_FINAL	EPISODE_END
8e-3	0.2	0.15	5.0	0	300

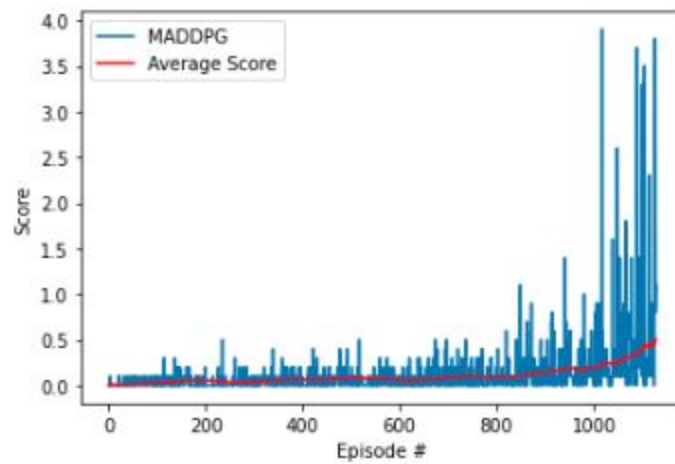


Environment solved in 1578 episodes

## Experiment 6

<b>BUFFER_SIZE</b>	BATCH_SIZE	LR_ACTOR	LR_CRITIC	WEIGHT_DECAY	<b>GAMMA</b>
<b>1e5</b>	128	1e-3	1e-3	0	<b>0.98</b>

TAU	OU_SIGMA	OU_THETA	EPS_START	EPS_FINAL	<b>EPISODE_END</b>
8e-3	0.2	0.15	5.0	0	<b>600</b>



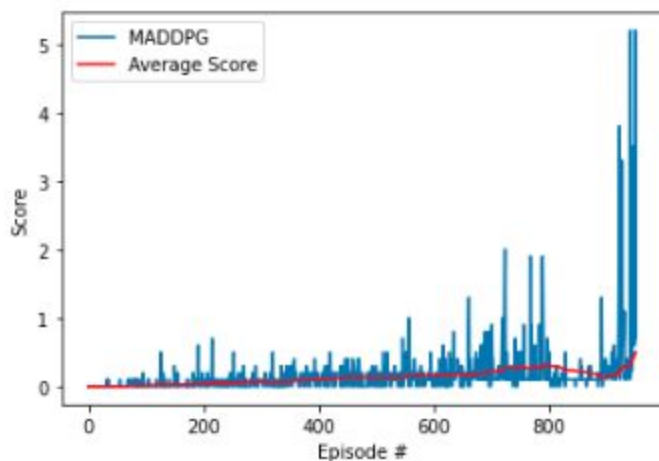
Environment solved in 1158 episodes

## Experiment 7

<b>BUFFER_SIZE</b>	BATCH_SIZE	LR_ACTOR	LR_CRITIC	WEIGHT_DECAY	GAMMA
<b>1e6</b>	128	1e-3	1e-3	0	0.98

<b>TAU</b>	OU_SIGMA	OU_THETA	EPS_START	EPS_FINAL	<b>EPISODE_END</b>
<b>10e-3</b>	0.2	0.15	5.0	0	<b>600</b>

```
<-- Environment solved in 852 episodes!
<-- Moving Average: 0.502 over past 100 episodes
```



## 7. Future ideas for improving the agent's performance

- Can improve the agent model to get higher scores when twisting hyperparameter, some important value are: gamma, learning rate of the actor and critic, replay buffer size, and noise process
- Can try another noise process to verify their performance.
- Future works may implement them to verify their performance using Multiagent. Some of those algorithms are:
  - [TRPO - Trust Region Policy Optimization](#)
  - [GAE - Generalized Advantage Estimation](#)
  - [A3C - Asynchronous Advantage Actor-Critic](#)
  - [ACER - Actor Critic with Experience Replay](#)
  - [PPO - Proximal Policy Optimization](#)
  - [D4PG - Distributed Distributional Deterministic Policy Gradients](#)

## References

1. [Open AI](#)

- 
2. [Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments](#)
  3. [Medium Markus Buchholz](#)
  4. [Silviomori Blog](#)