Report on Collaboration and Competition Project

This project has been split into four modules due to its length, logical structure, and reusability. The modules are as follows:

- maddpg.py: Implementation of MADDPG agents that interact with and learns from the environment.
- networkforall.py: Implementation of Actor (Policy) and Critic (Value) Model.
- Network.py: Implementation of Actor (Policy) model and Critic (Value) model.
- main.py: This is the main module with the training loop.
- utils.py: This file contains helper functions for the project

Multi-Agents Deep Deterministic Policy Gradients (MADDPG) Algorithm

MADDPG uses local information (i.e. their own observations) at execution time and centralized training with decentralized execution. It is applicable not only to cooperative interaction but also to competitive or mixed interaction.

Decentralized actor network

1. Suppose policies of N agents are parameterized $\theta = \{\theta_1, \theta_2, ..., \theta_N\}$ and let $\mu = \{\mu_1, \mu_2, ..., \mu_N\}$ be the continuous policies.
2. The goal is to find θ_i that maximizes the expected return for agent i, $J(\theta_i) = E[R_i]$
3. The gradient of $J(\theta_i)$ would be

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{s\sim p^{\boldsymbol{\mu}}, a_i\sim \boldsymbol{\pi}_i}[\nabla_{\theta_i} \log \boldsymbol{\pi}_i(a_i|o_i) Q_i^{\boldsymbol{\pi}}(\mathbf{x}, a_1, ..., a_N)].$$

Centralized critic network

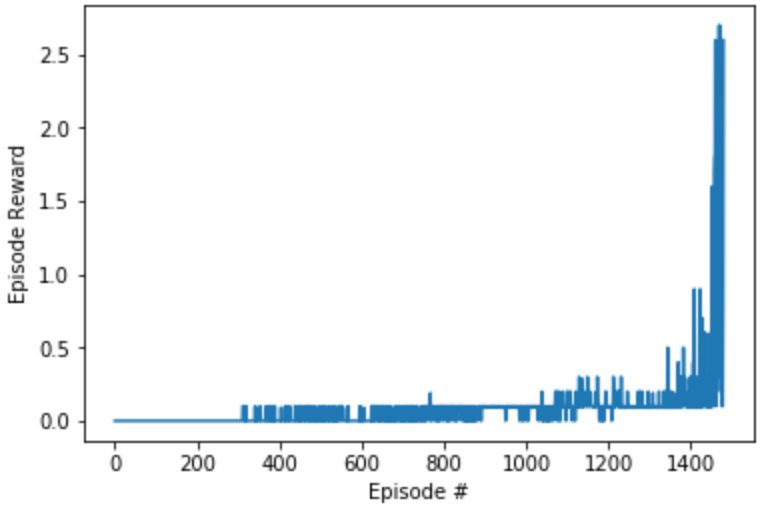
1. Centralized action-value function for agent i is $Q_i^{\mu}(x, a_1, a_2, ..., a_N)$.
2. The centralized action-value function is updated as:

$$\mathcal{L}(\theta_i) = \mathbb{E}_{\mathbf{x},a,r,\mathbf{x}'}[(Q_i^{\boldsymbol{\mu}}(\mathbf{x}, a_1, \ldots, a_N) - y)^2],$$

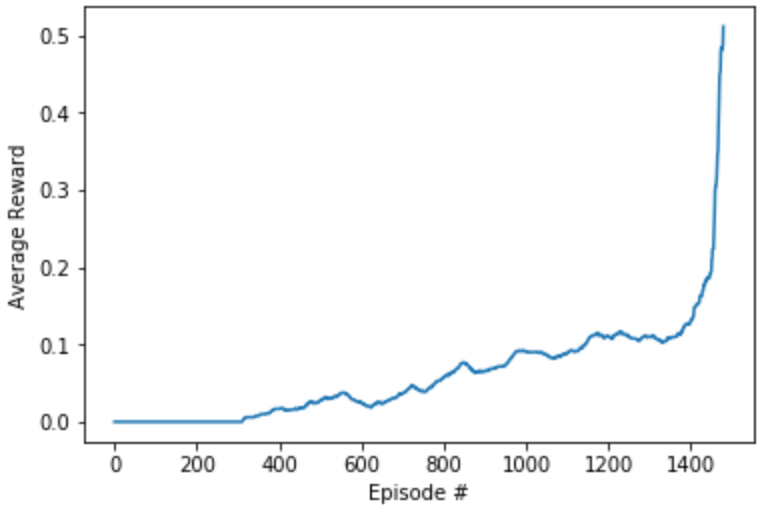
$$y = r_i + \gamma\, Q_i^{\boldsymbol{\mu}'}(\mathbf{x}', a'_1, \ldots, a'_N)\big|_{a'_j = \boldsymbol{\mu}'_j(o_j)}$$

Plot of Rewards

The plot for each episode reward is shown below. As we can see, the episode reward is increasing rapidly after about 1000 episodes.



The plot for average reward is shown below. The environment is solved in around 1500 episodes with an average reward (over 100 episodes) of at least +0.5.



Ideas for Future Work

- Implement Q-Mix algorithm (QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning)
- Implement MARWIL algorithm (Advantage Re-Weighted Imitation Learning)