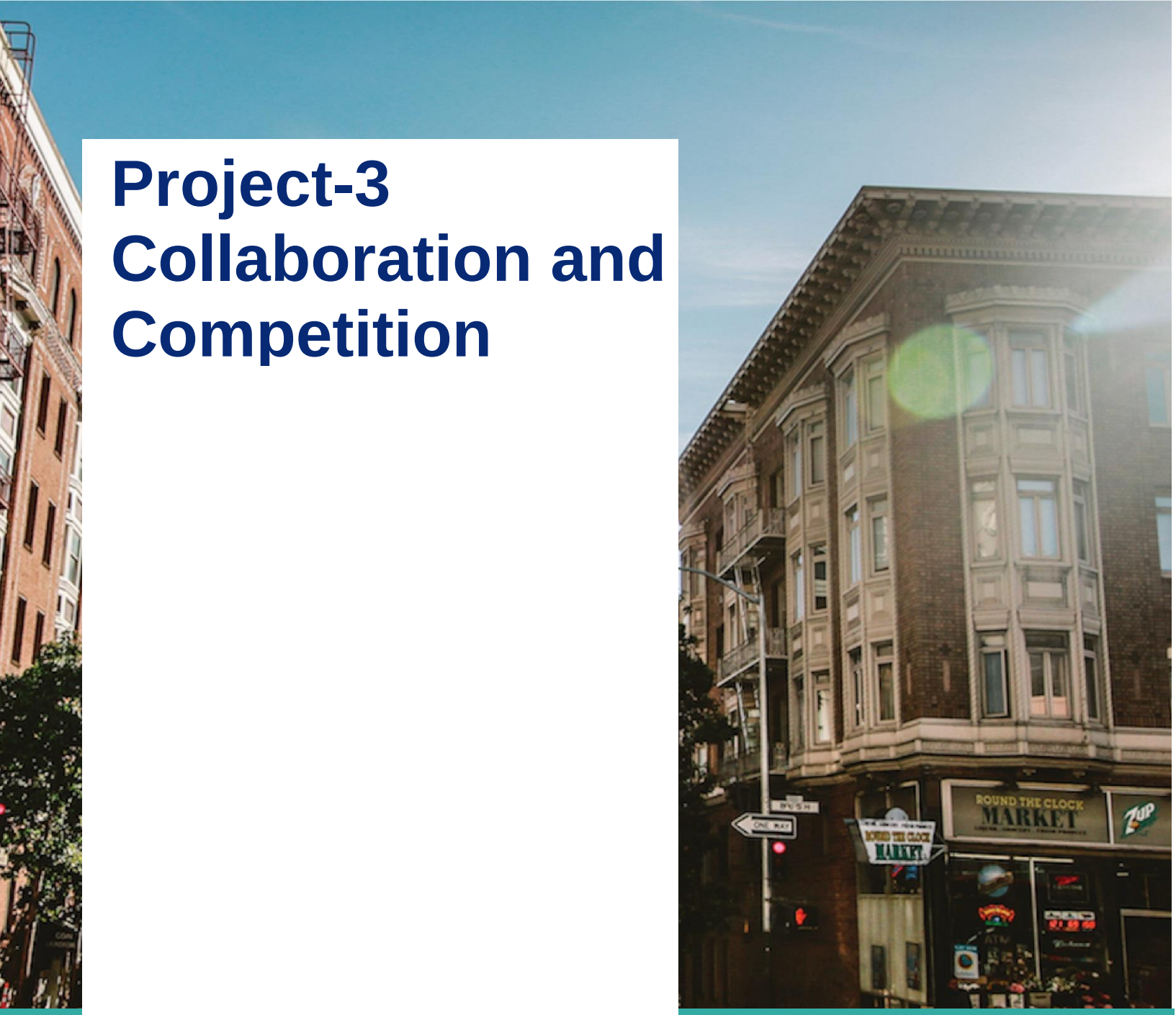
Project-3 Collaboration and Competition



1ST APR 2020

AMRIT PAUL

Problem Statement

The goal of the project is to demonstrate the abilities of a model-free reinforcement learning algorithm, particularly Multi-Agent Deep Deterministic Policy Gradients (MADDPG) Algorithm, which consists of four neural networks for each of the agents, namely Actor and Critic neural networks. The project uses Unity Environment, a game development framework and Pytorch, a deep learning framework. The algorithm was trained on 2 agents, with each agent consisting of 24 states and 2 action/observation spaces.

Description

Multi Agent Deep Deterministic Policy Gradient (MADDPG) is an algorithm which concurrently learns a Q-function and a policy. It uses off-policy data and Bellman equation to learn the Q-function and uses the Q-function to learn the policy.

The approach is closely connected to Q-learning and is motivated the same way: if you know the optimal action-value function Q*(s,a), then in any given state, the optimal action a*(s) can be found by solving :-

$$a^*(s) = \arg\max_a Q^*(s,a).$$

DDPG interleaves learning an approximation to Q*(s,a) with learning an approximator to a*(s) and it does so in a way which is specifically adapter for environments with continuous action spaces. The actor produces a deterministic policy instead of the usual stochastic policy and the critic

evaluates the deterministic policy. The critic is updated using the TD-error and the actor is trained using the deterministic policy gradient algorithm.

$$\begin{aligned}\nabla_{\theta^\mu} J &\approx \mathbb{E}_{s_t \sim \rho^\beta} \left[\nabla_{\theta^\mu} Q(s, a|\theta^Q)|_{s=s_t, a=\mu(s_t|\theta^\mu)} \right] \\ &= \mathbb{E}_{s_t \sim \rho^\beta} \left[\nabla_a Q(s, a|\theta^Q)|_{s=s_t, a=\mu(s_t)} \nabla_{\theta_\mu} \mu(s|\theta^\mu)|_{s=s_t} \right]\end{aligned}$$

MADDPG adopts a framework of:-

- Centralized training:
  - Agents share experience during training
  - Implemented via a shared experience-relay buffer
- Decentralized execution:
  - Each agent uses only local observations at execution time

Extend actor-critic policy gradient methods (e.g. DDPG)

- Critic is augmented with information about policies of other agents/
- Actor has access only to local information

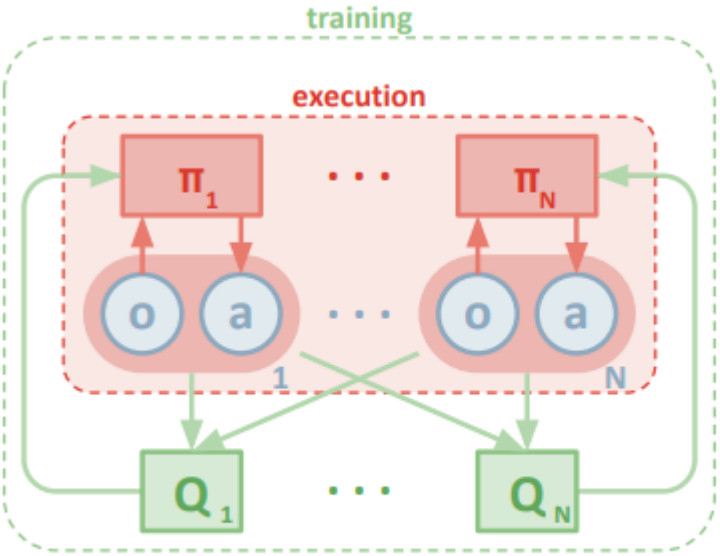
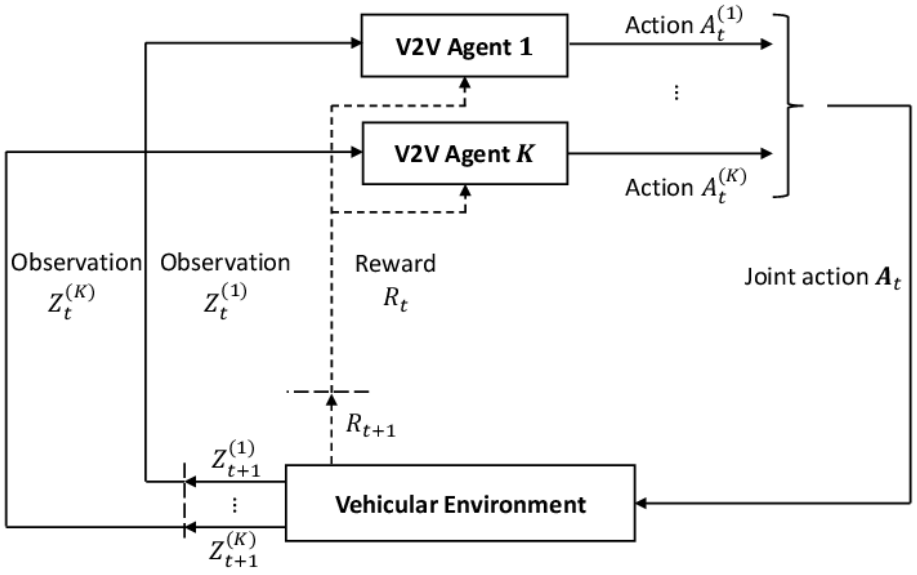


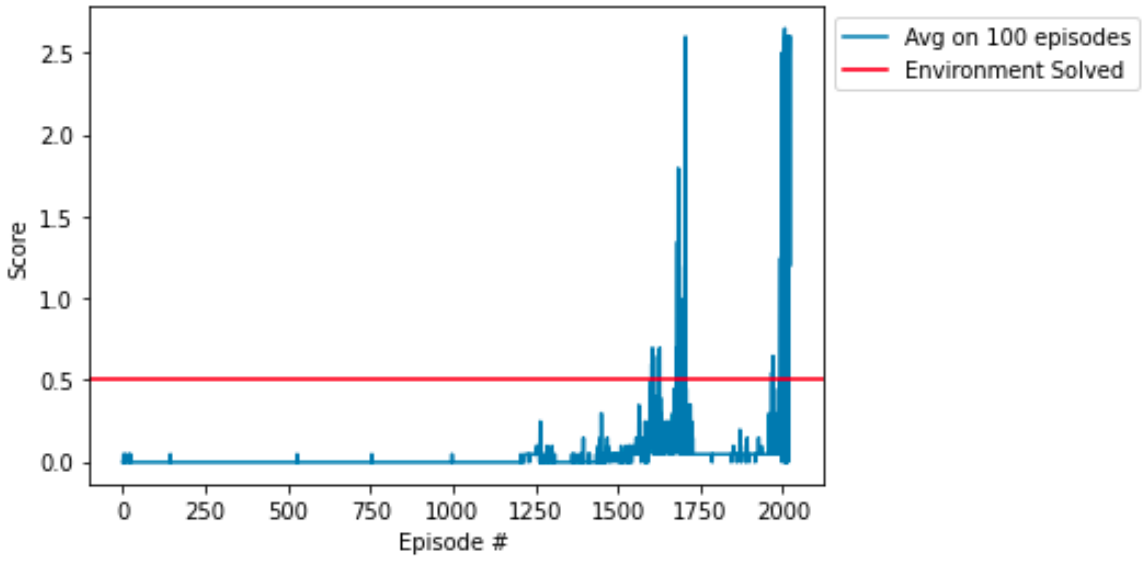
Figure 1: Overview of our multi-agent decentralized actor, centralized critic approach.

The diagram below represents the control flow of MADDPG.



Results

The following diagram shows the results after the MADDPG algorithm is trained-



Episode: 100Average Score: -0.00Current Score: -0.00

Episode: 200Average Score: -0.00Current Score: -0.00

Episode: 300Average Score: -0.00Current Score: -0.00

Episode: 400Average Score: -0.00Current Score: -0.00

Episode: 500Average Score: -0.00Current Score: -0.00

Episode: 600Average Score: -0.00Current Score: -0.00

Episode: 700Average Score: -0.00Current Score: -0.00

Episode: 800Average Score: -0.00Current Score: -0.00

Episode: 900Average Score: -0.00Current Score: -0.00

Episode: 1000Average Score: -0.00Current Score: -0.00

Episode: 1100Average Score: -0.00Current Score: -0.00

Episode: 1200Average Score: -0.00Current Score: -0.00

Episode: 1300Average Score: 0.03Current Score: -0.000

Episode: 1400Average Score: 0.00Current Score: -0.00

Episode: 1500Average Score: 0.02Current Score: 0.050

Episode: 1600Average Score: 0.06Current Score: 0.100

Episode: 1700Average Score: 0.22Current Score: 0.50

Episode: 1800Average Score: 0.12Current Score: 0.050

Episode: 1900Average Score: 0.05Current Score: 0.050

Episode: 2000Average Score: 0.16Current Score: 1.200

Episode: 2022Average Score: 0.51Current Score: 2.600

Environment solved in 1922 episodes!Average Score: 0.51

Hyperparameter	Value
Number of Actions	2
Number of Agents	2
Number of States	24
Number of Episodes	3000
Max time steps per episode	1000
Replay Buffer Size	1e6
Batch Size	1024
Soft update of target parameters	1e-3
Gamma	0.99
Actor learning rate	1e-3
Critic learning rate	1e-3
Leaky ReLU leakiness	0.01
Activation	Leaky ReLU
Update after every step	10

Improvements

- Other algorithms such as TRPO, PPO, A3C, D4PG could potentially lead to better results
- Using a prioritized experience replay can improve he performance of the model
- Using a combination of on-policy and off-policy algorithm could potentially lead to better results.
- Using convoluted neural network architectures with complex activation functions could potentially give better results.