Udacity - Deep Reinforcement Learning Course

Project 3: Collaboration and Competition

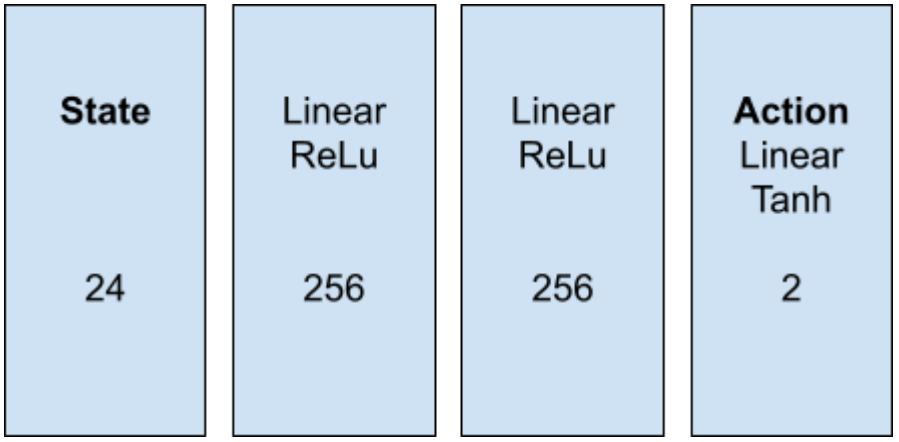
by Mauricio Ciprián Rodríguez (@mauriciociprian)

This document summarizes the relevant details of the proposed solution to Project 3 of the course.

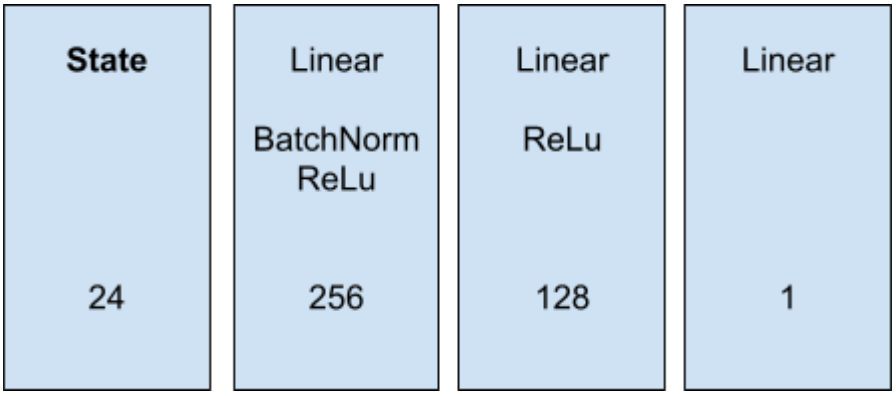
Actor-Critic Agent Implementation

The implementation of the agent was made starting from the code provided in the DDPG Udacity code, making several modifications: managing multiple agents, sharing their experience to train a single Actor model, and adapting the network architecture. The code for this is in the **mult_ddpg_agent.py** file.

The Actor uses a Feed Forward neural network with the following architecture



In addition, the Critic neural network follows this architecture:

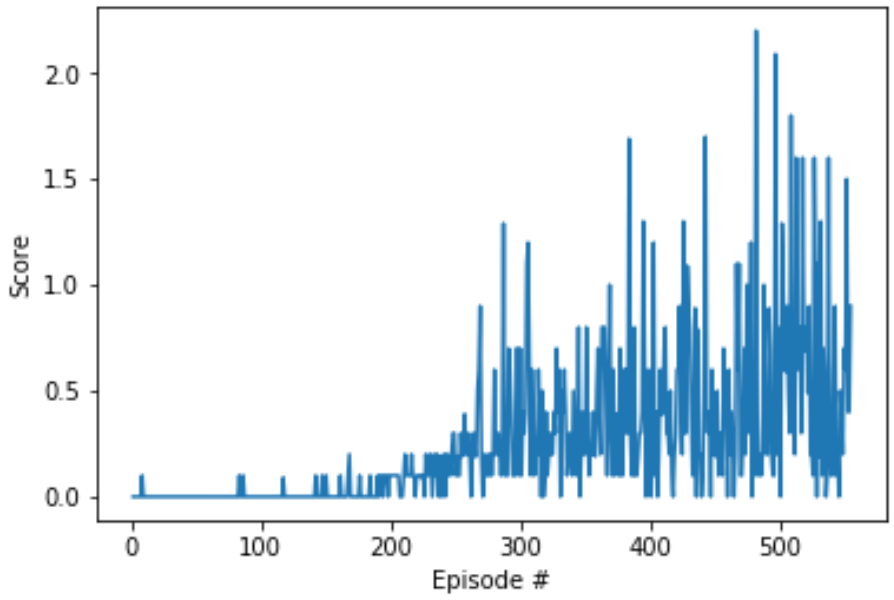


In addition, the agent uses the following hyperparameters:

- **Replay Buffer size:** 100,000
- **Batch Size:** 256
- **Gamma (Discount factor):** 0.99
- **Tau (Parameter update):** 0.001
- **Learning rate Actor:** 0.0001
- **Learning rate Critic:** 0.0001
- **Weight Decay (Critic Network):** 0

Results

The graph below shows the evolution of the agent's reward during training



After 520 episodes, the agent reached the minimum score required to pass the project (Average Score = 0.5 in 100 Episodes). However, the agent trained to achieve an average score of 0.55. The models resulting from this training are stored in the **checkpoint_actor.pth** and **checkpoint_critic.pth** files.

Further Work

- A hyper parameters search (GridSearch) and other network architecture (Dropout, Layers sizes, etc) could improve the learning capacity of this DDPG implementation.
- Evaluate other Actor-Critic approaches (ie. A3C, PPO).