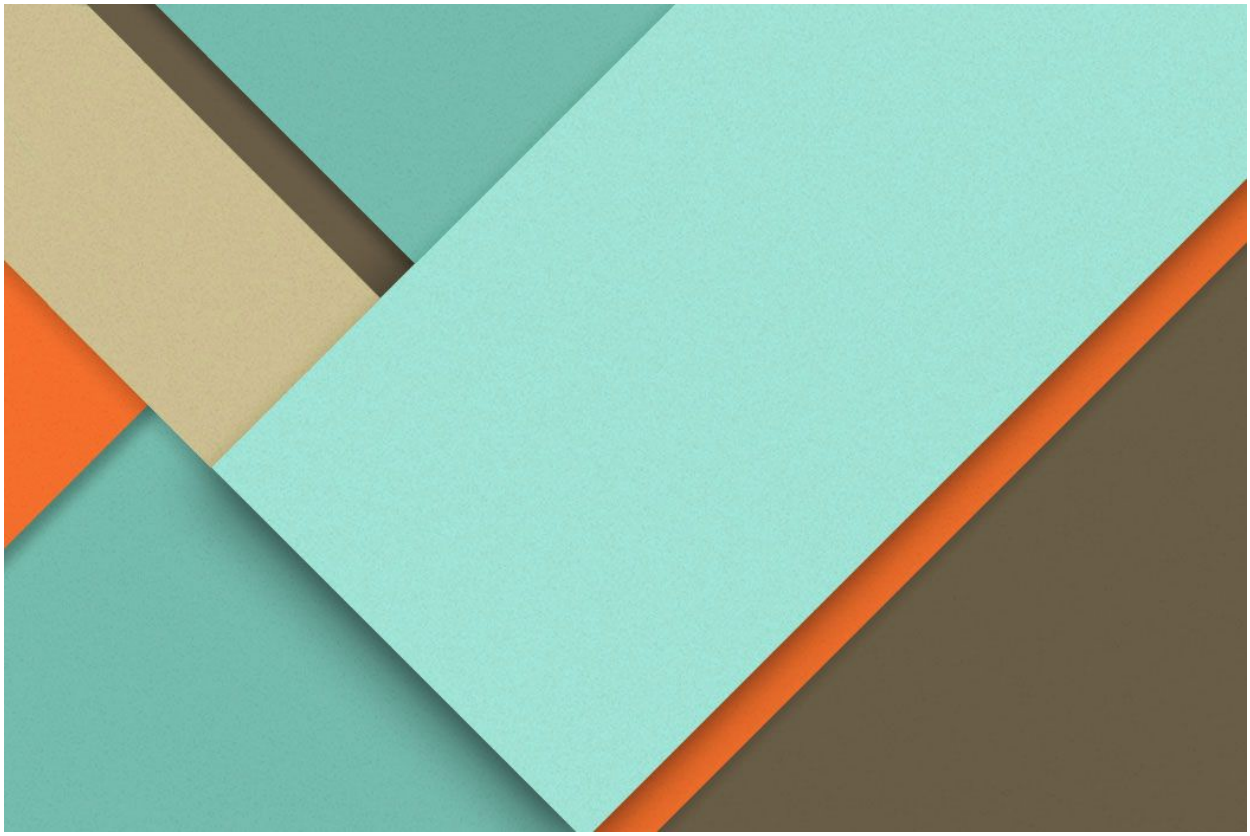

Collaboration and Competition

Anu Pradhan

Overview

The objective of the third project is to train agents in a collaborative way to play a table tennis game. In the beginning, I explored a number of techniques, such as Deep Deterministic Policy Gradient (DDPG) and Distributed Distributional Deep Deterministic Policy Gradient (D4PG) approaches.

During the course of this project, I consulted a number of books and websites. I used some sections of the source code used in the resources which are listed below:

a) Deep Reinforcement Learning Hands-on by Maxim Lapan

b) https://github.com/PacktPublishing/Deep-Reinforcement-Learning-Hands-On

c) https://github.com/kelvin84hk/DRLND_P3_collab-compet

d) https://github.com/rlcode/per

Why D4PG?

1. Barth-Maron et al. (2018) made a number of important extensions or modifications to the DDPG. D4PG utilizes a distributional version of the critic update which provides a better and stable learning signal.
2. D4PG uses N-step returns which helps to reduce variance.
3. D4PG runs multiple actors in parallel and the experiences collected by these actors are fed into a single replay table. Thus, the agents have more experiences to learn from.
4. Prioritized replay is used instead of a regular replay. The prioritized replay uses importance sampling based on TD errors instead of a random sampling approach used in regular replay.

Neural Network Architecture and Hyper-parameters

I. Neural Network Architecture

The Actor and Critic Networks are implemented as a feed-forward neural network. The Actor network consists of 2 hidden layers with 400 and 300 units respectively. Batch Normalization is performed at the last hidden layer, and Batch Normalization was able to reduce the number of iterations needed to train the networks. LEAKY_RELU activation is used at each hidden layer. For the output layer, tanh activation is used for the actor network.

For the Critic Network, the output is not a single Q-value for the given state and action. It now returns N atom values corresponding to the probabilities of values. That's why distributional term is used in D4PG. Based on Maxim Lapan book, I used the number of atoms = 51. The range is set by Vmin and Vmax which are -1 and 1 respectively.

II. Hyper-parameters

The following values are used:

```
BUFFER_SIZE = int(1e6)  # replay buffer size
BATCH_SIZE = 64  # minibatch size
GAMMA = 0.99  # discount factor
TAU = 1e-2 # for soft update of target parameters
LR_ACTOR = 1e-4  # learning rate of the actor
LR_CRITIC = 1e-4  # learning rate of the critic
WEIGHT_DECAY = 0  # L2 weight decay
```

To tune the above hyperparameters, I looked into the Supplementary Information section of the Lillicrap et al. paper. I started with the values suggested by the paper, and the values used in https://github.com/kelvin84hk/DRLND_P3_collab-compet.

Overall Performance

To achieve the reward of 0.5+, I need to run around 12500 episodes as shown in the plot. The plot is as shown in the attached jupyter notebook.

Future Improvements

Additional Experiments on Hyperparameters

Because of computational limitations and time constraints, I was not able to do a thorough job in terms of tuning the hyperparameters. Hypertuning could help to further improve the performance.

Distributed Prioritized Experience Replay

Horgan et al. (2018) proposed a distributed architecture for prioritized experience replay. The current D4PG can be augmented with this new architecture for prioritized experience replay.