# DEEP Q-NETWORKS LEARNING BASIC ATARI VIDEO GAMES

Mingyuan Zang, Yingrui Li, Philip Butenko

Technical University of Denmark

## INTRODUCTION

Deep Q-Learning is a method that combines Deep Learning and Q-learning, so that it can learn from "trial and error" and perform actions as an agent in a game environment. The goal of the method is to get the maximum rewards and to do so in the shortest time possible.

In this project, the agent has been trained to play simple games like CartPole and LunarLander using DQN. The Atari game Pong, is also learnt using different techniques such as DQN. A CNN is also applied in Pong to process the video footage and there by learn primarily from the visual output of the game.

[References]

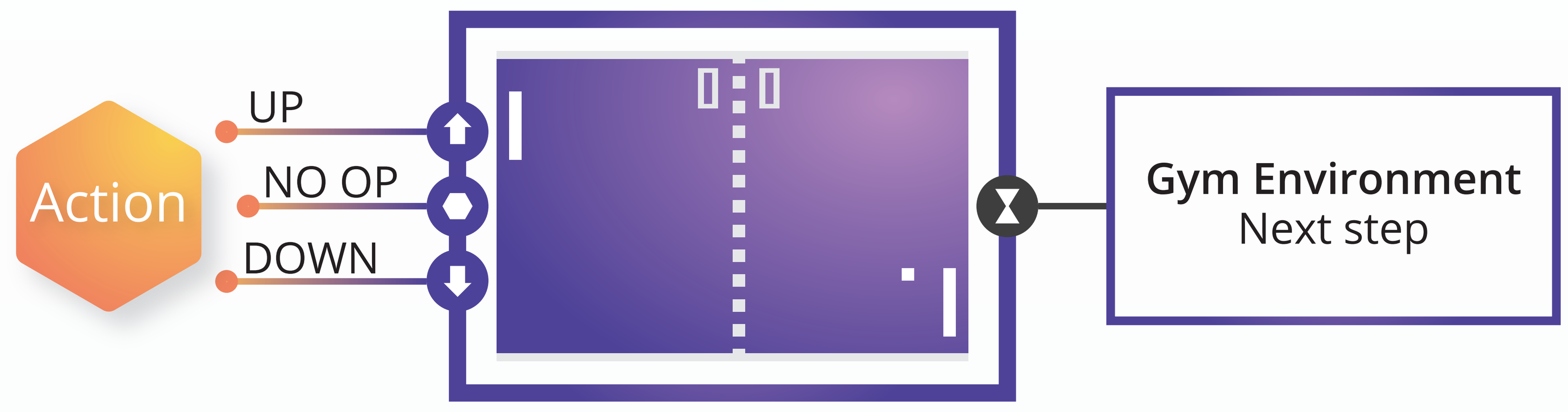
[1] Mnih et al., Human-level control through deep reinforcement learning , 2015

[2] Mnih et al., Playing Atari with Deep Reinforcement Learning , 2013

[3] Arulkumaran et al., A Brief Survey of Deep Reinforcement Learning , 2017.

[4] Pytorch, Reinforcement Learning (DQN) Tutorial, https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html

[5] https://github.com/fg91/Deep-Q-Learning



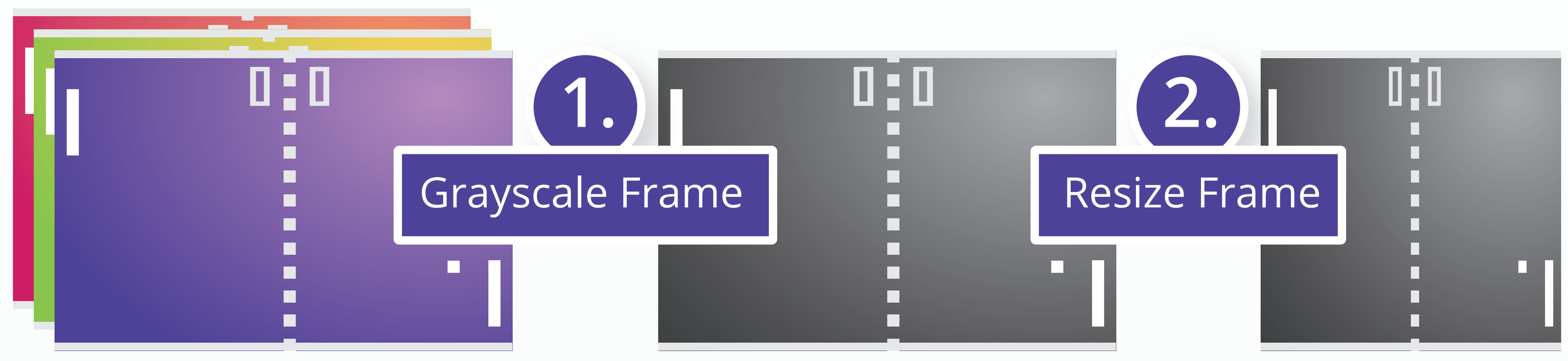

[Discussion]

- The agent is able to win fairly consistently with a good avg. score in Pong after 1 million frames of training.

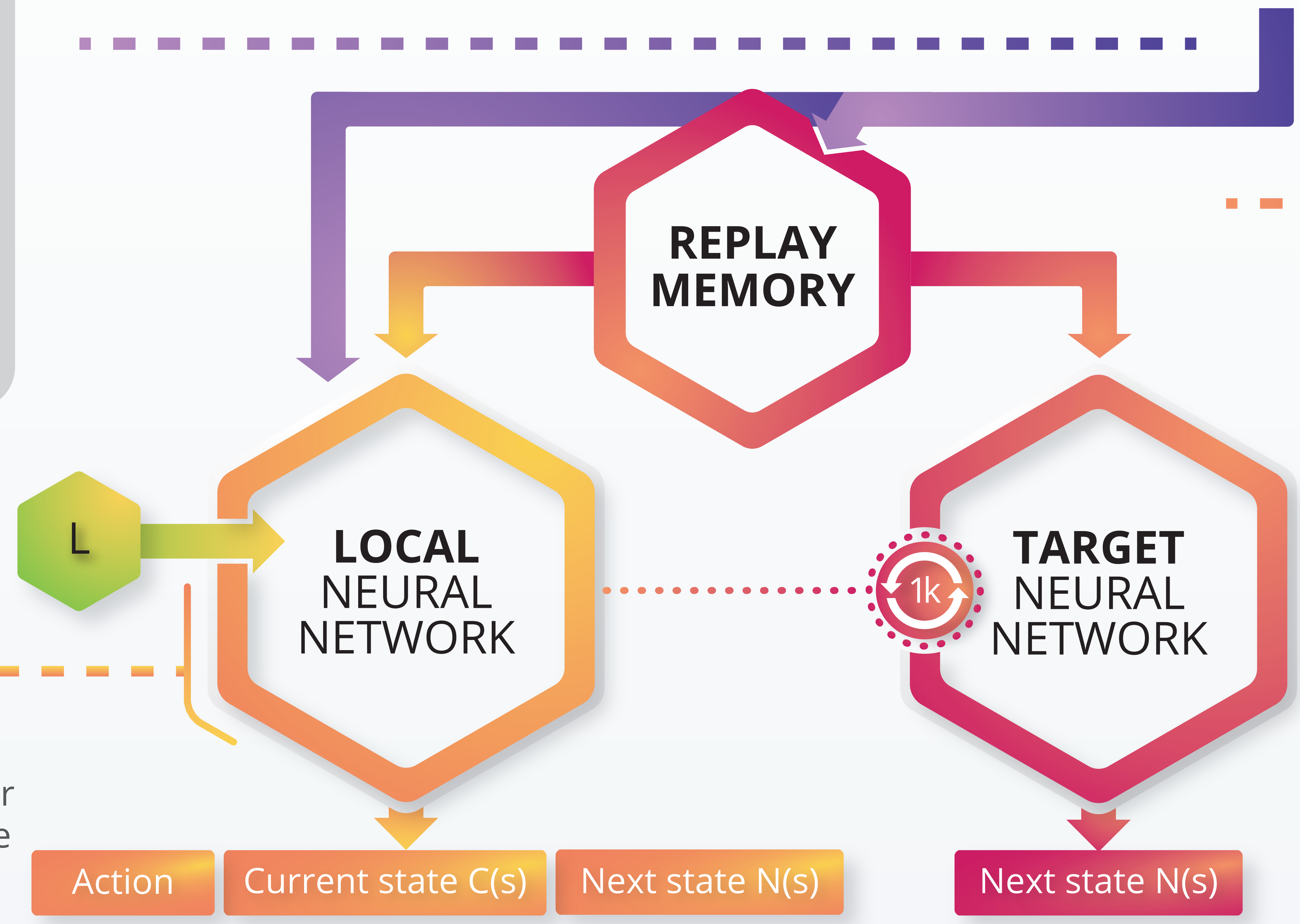
- According to the empirical data, DDQN shows better results than regular DQN and Dueling.

- GPU-capability can affect the training process. Colab GPU was used at the beginning, but due to the time-out problem, we then changed to a GeForce RTX 2080 Ti.

- Huber Loss is discussed in some references to yield a better result, but it did not show a better performance than mean square in tests performed.



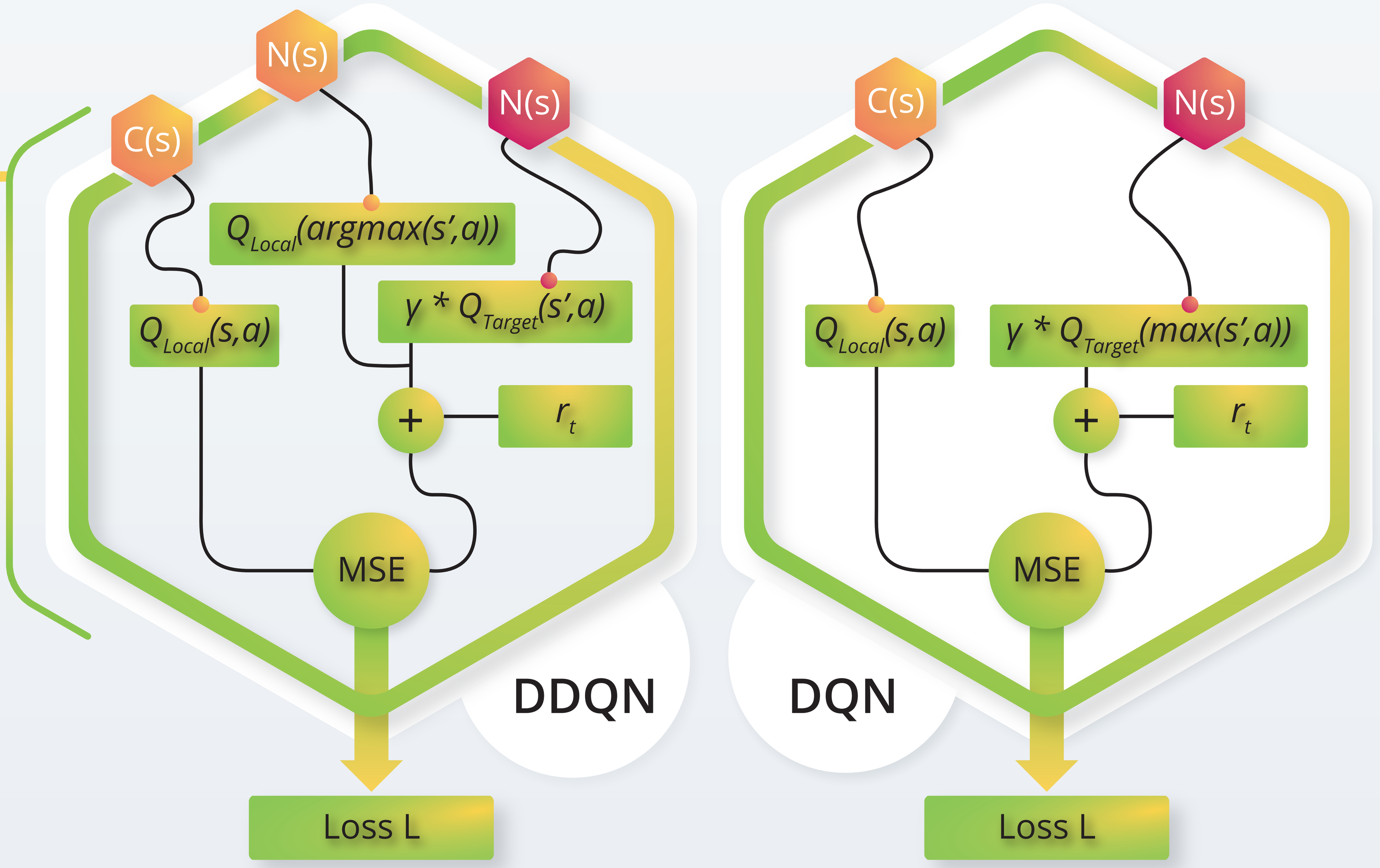

**Improvement for Gym wrappers:**
The wrappers offered by Gym includes several methods to pre-process the game frames. Some of them are used in this project to pre-process the Pong frames and overwrite the observation space.




**Improvement for Exploration & Exploitation**
Instead of using linear rate, epsilon is set to be decreased at a exponential rate.

**Improvement for Dueling DQN:**
Dueling splits the final convolutional layer into two streams that represent the value and advantage functions that predict a state value V(s) that depends only on the state, and action advantages A(s,a) that depend on the state and the respective action.

## TESTED LOSS FUNCTIONS




**Improvement for DDQN:**
DQN sometimes learns unrealistically high action values because it always chooses a maximization step over estimated action values, which tends to prefer overestimated values to underestimated values.
Hence DDQN is used this way:
Ask the local network which action has the highest Q-value and pick the index of it;
Go to target network using the specific index to pick the Q-value.

[Future]

- Limiting the action space might speed up the training.

- Initializing the network weights may help improve the training result.

- In some references, RMSProp can help the training process. Thus, more work can be done to compare how different kinds of optimizers (Adam, SGD, RMSProp) can affect the training result.

- Breakout(An Atari game) was also tried with the model presented in this project. However, so far, only a score of 30 has been reached after full night of training.

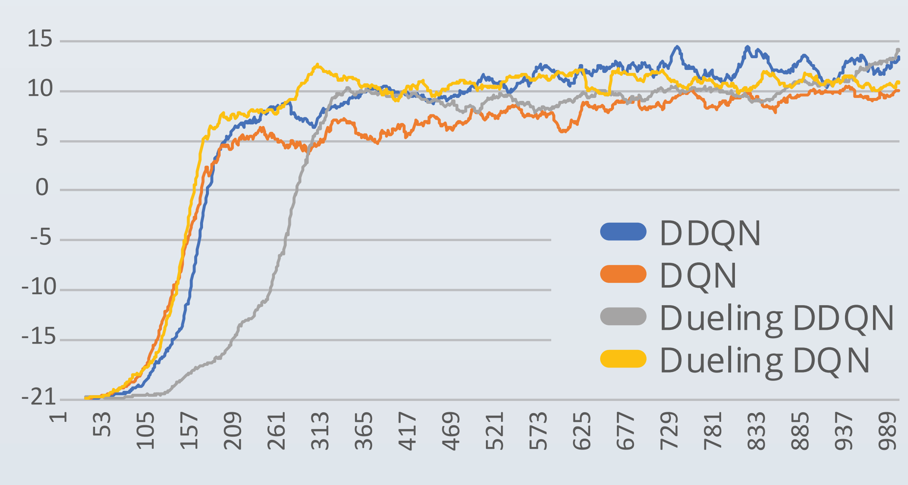
### Average Reward

| | | | |
|---|---|---|---|
| DQN | 11.4 | DUELING DQN | 11.5 |
| DDQN | 13.5 | DUELING DDQN | 15.3 |

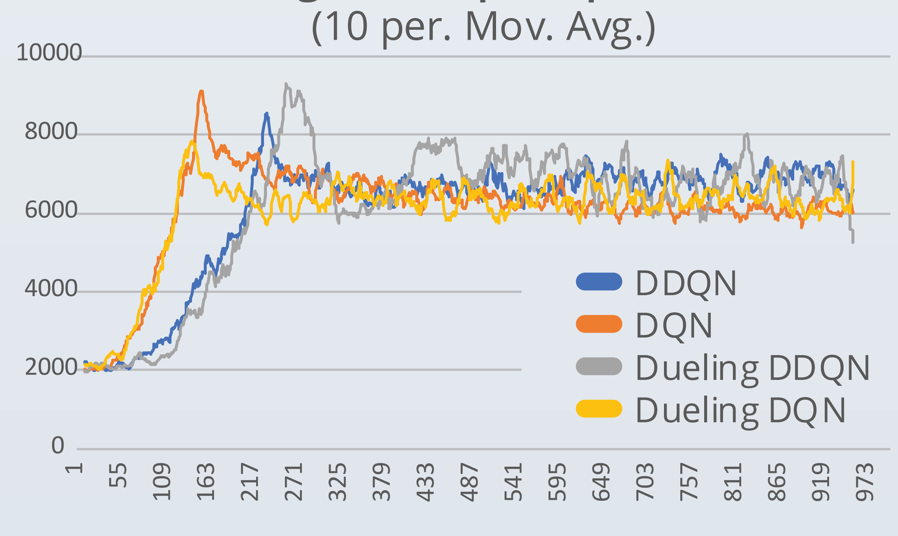
### Hyper Parameters

| Buffer Size | Learning Rate | Gamma | Batch Size |
|---|---|---|---|
| 1e5 | 1e-4 | 0.99 | 32 |

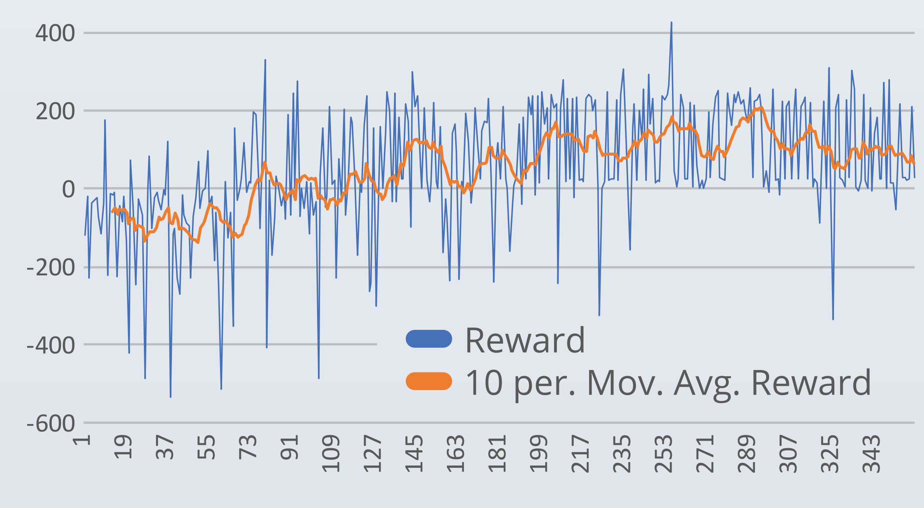
Pong: (Reward)
(30 per. Mov. Avg.)




Pong: (Time per Episode)
(10 per. Mov. Avg.)




Lunar Lander: (Reward)




Cartpole: (Reward)