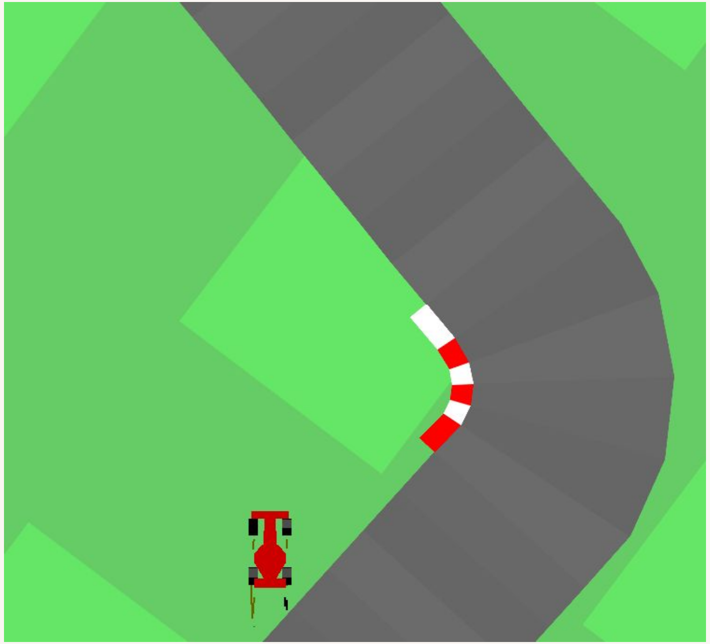
# DRL Agent For Car Racing Game

Qidi Fang, Tian Huang, JIajin Huo



# what problem did we solving?

- Applied DRL in Car Racing Game
    - Gym Box2D - Car Racing

- Compared the performances and time costs of Double DQN, Actor-Critic and PPO

**Target of the game:** Keep on track

# Related Work

## Car Racing Game:

- A3C Jaritz et al.
- SAC https://github.com/trackmania-rl/tmrl
- Combined state-of-the-art, model-free, deep reinforcement learning algorithms in Gran Turismo

## OpenAI gym Car Racing:

- PPO https://notanymike.github.io/Solving-CarRacing/
- DQN https://github.com/andywu0913/OpenAI-GYM-CarRacing-DQN
- PID https://medium.com/@kartha.kishan/solving-openai-carracing-v0-using-image-processing-5e1005ee0cb

# Environment

## Action Space & State Space

| | |
|---|---|
| Action Space | `Box([-1. 0. 0.], 1.0, (3,), float32)` |
| Observation Space | `Box(0, 255, (96, 96, 3), uint8)` |
| import | `gymnasium.make("CarRacing-v2")` |

## Reward Function

The reward is -0.1 every frame and +1000/N for every track tile visited, where N is the total number of tiles visited in the track. For example, if you have finished in 732 frames, your reward is 1000 - 0.1*732 = 926.8 points.

# Wrapped Environment

## State Space

## Reward Function

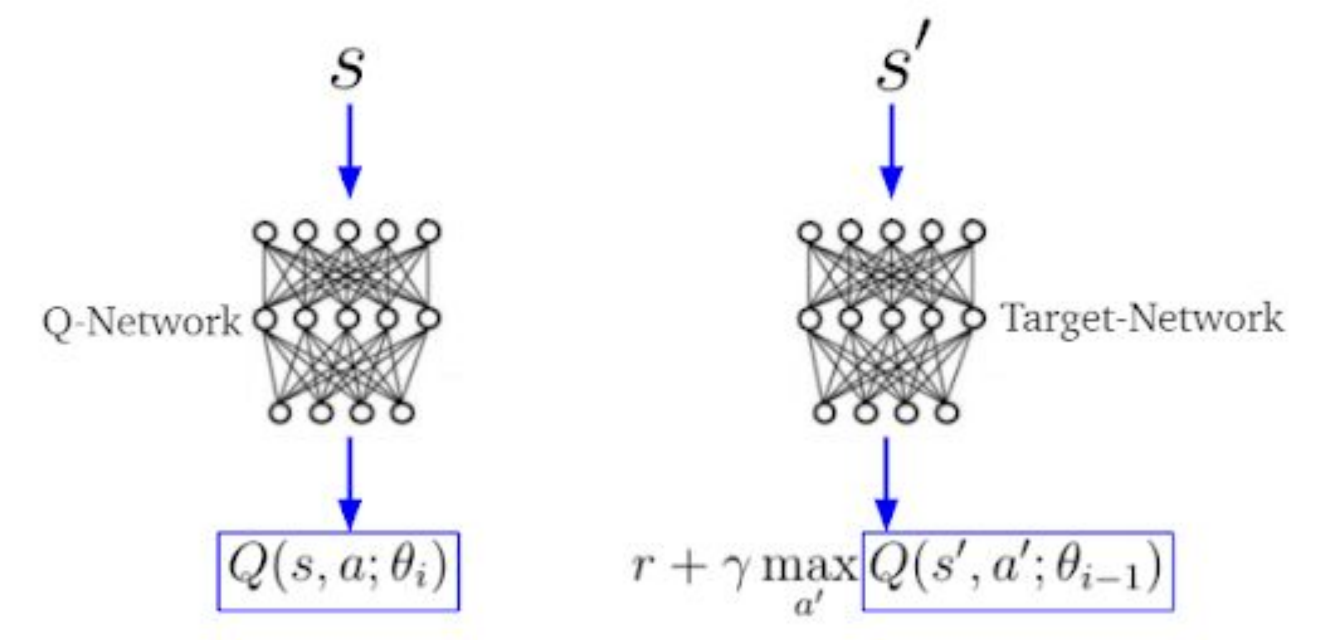
$$\text{reward} = \begin{cases} -0.1 & \text{for each frame,} \\ +1.0 & \text{for each tile passed in a counterclockwise direction,} \\ -5.0 & \text{for each frame of running off the track.} \end{cases}$$

## Action Space

| Index | Discrete action | Continuous action |
|---|---|---|
| 0 | Accelerate hard | $[0.0, 0.5, 0.0]$ |
| 1 | Accelerate oft | $[0.0, 1.0, 0.0]$ |
| 2 | Turn left hard & Accelerate soft | $[-1.0, 0.5, 0.0]$ |
| 3 | Turn left hard & Brake soft | $[-1.0, 0.0, 0.5]$ |
| 4 | Turn left soft & Accelerate soft | $[-0.5, 0.5, 0.0]$ |
| 5 | Turn left soft & Brake soft | $[-0.5, 0.0, 0.5]$ |
| 6 | Turn right hard & Accelerate soft | $[1.0, 0.5, 0.0]$ |
| 7 | Turn right hard & Brake soft | $[1.0, 0.0, 0.5]$ |
| 8 | Turn right soft & Accelerate soft | $[0.5, 0.5, 0.0]$ |
| 9 | Turn right soft & Brake soft | $[0.5, 0.0, 0.5]$ |
| 10 | Turn left hard | $[-1.0, 0.0, 0.0]$ |
| 11 | Turn left soft | $[-0.5, 0.0, 0.0]$ |
| 12 | Turn right hard | $[1.0, 0.0, 0]$ |
| 13 | Turn right soft | $[0.5, 0, 0.0]$ |

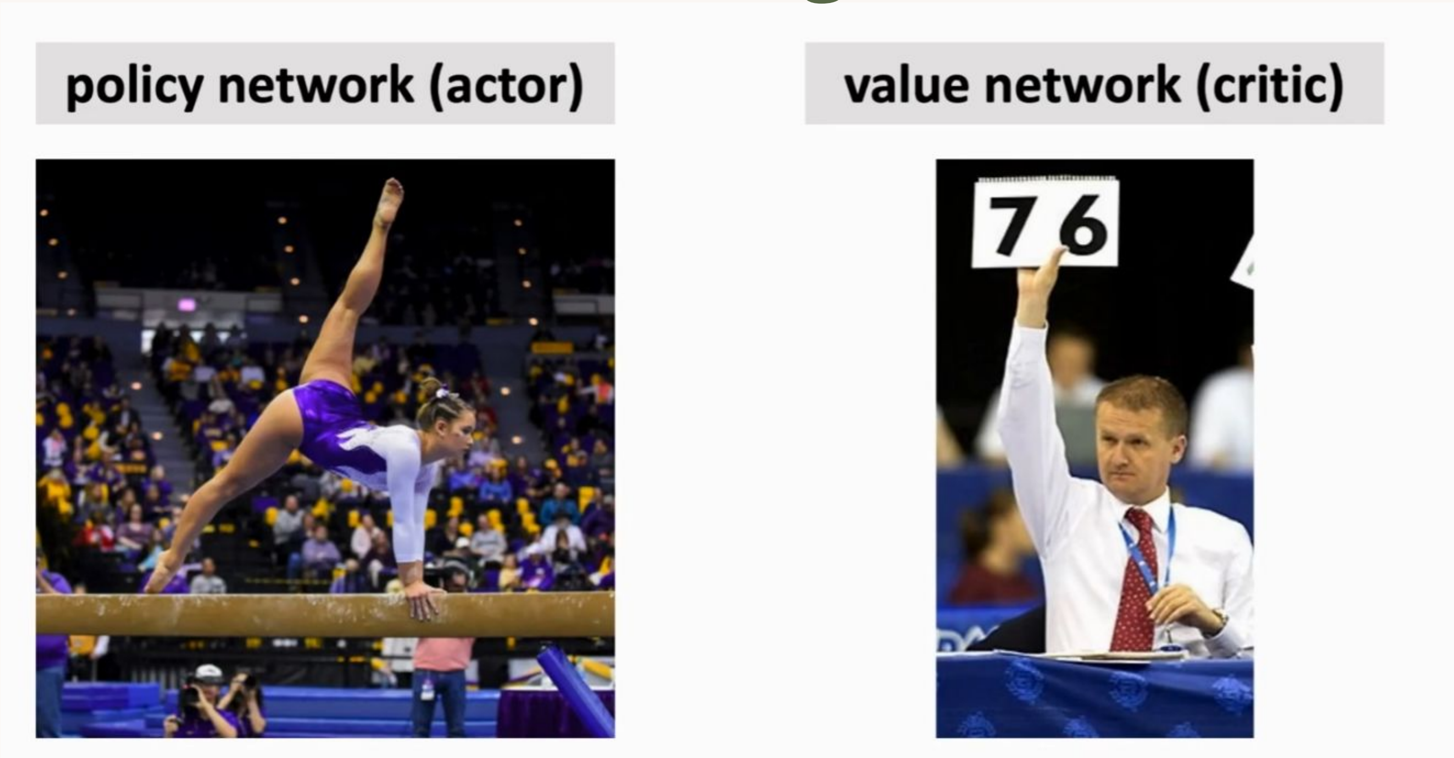
Table 1: Mapping of discrete actions to continuous action space.

# Double DQN



$s$
$s'$
Q-Network
Target-Network
$Q(s, a; \theta_i)$
$r + \gamma \max_{a'} Q(s', a'; \theta_{i-1})$

# Actor-Critic Algorithm




https://www.youtube.com/watch?v=vmkRMvhCW5c&list=PLvOO0btloRnsiqM72G4Uid0UWljikENlU&index=2&t=876s

# PPO-Clip Algorithm

$$\text{clip}\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, 1-\epsilon, 1+\epsilon\right) A^{\pi_{\theta_k}}(s,a)$$

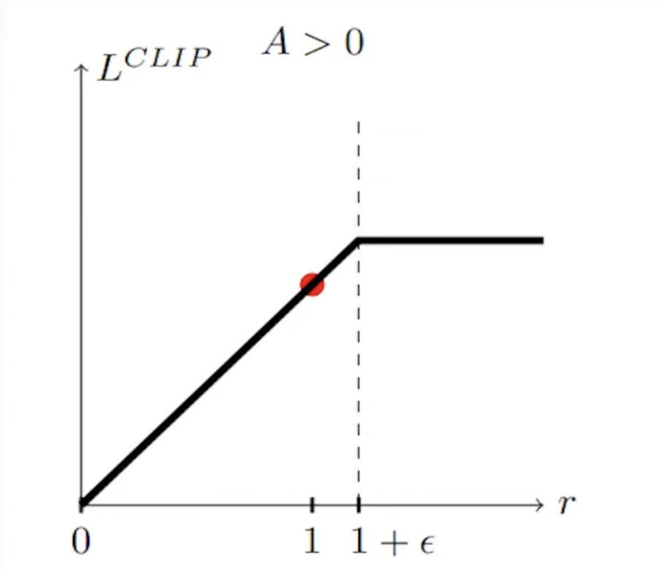



figure 1

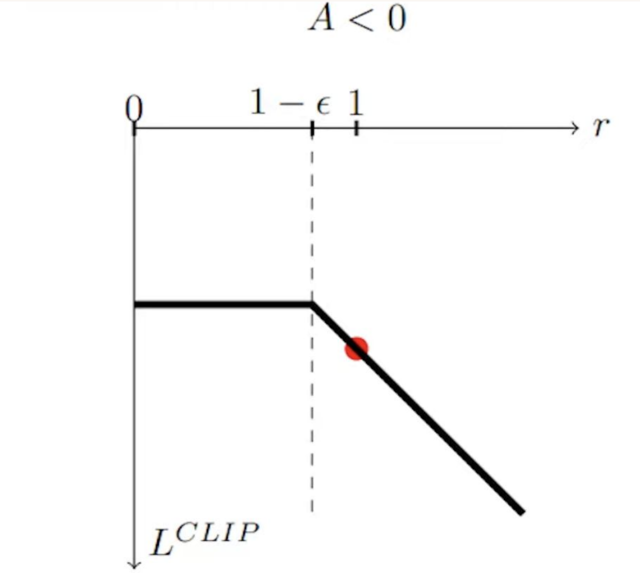
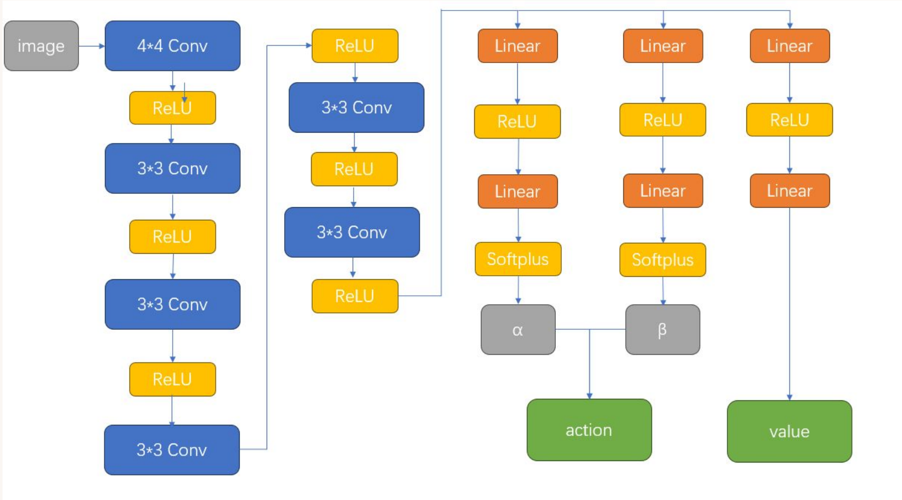



figure 2

# Experiments

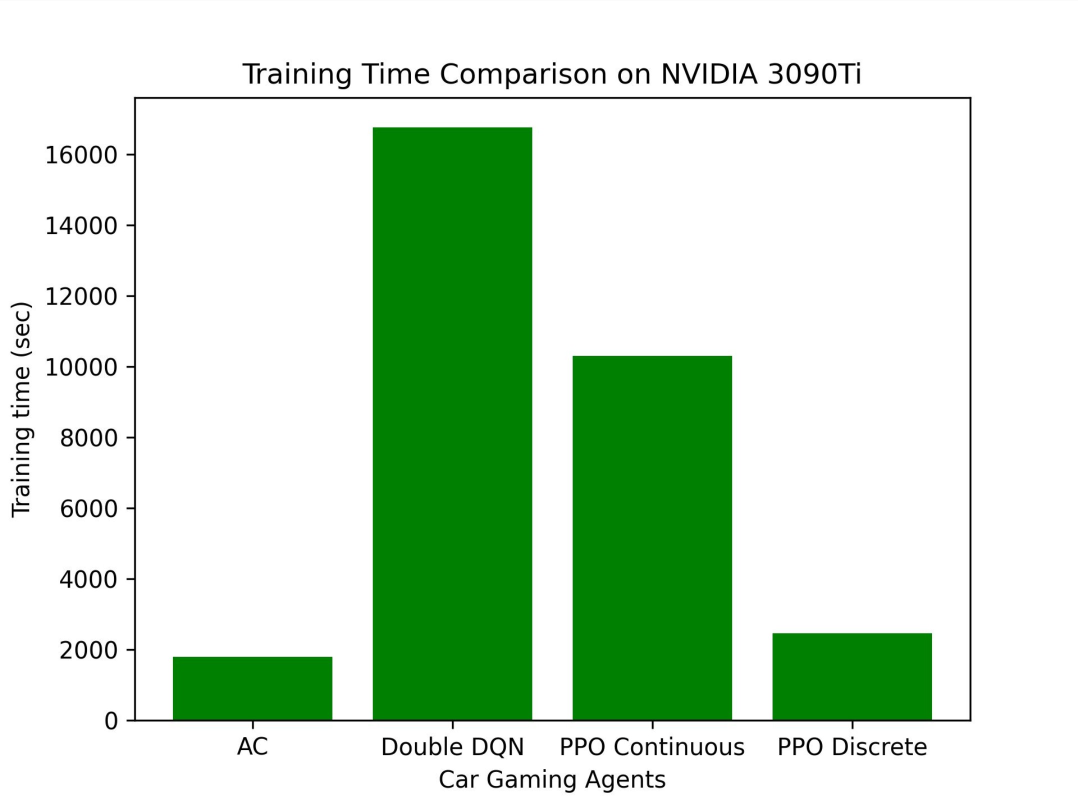



**Hyper Parameter:**

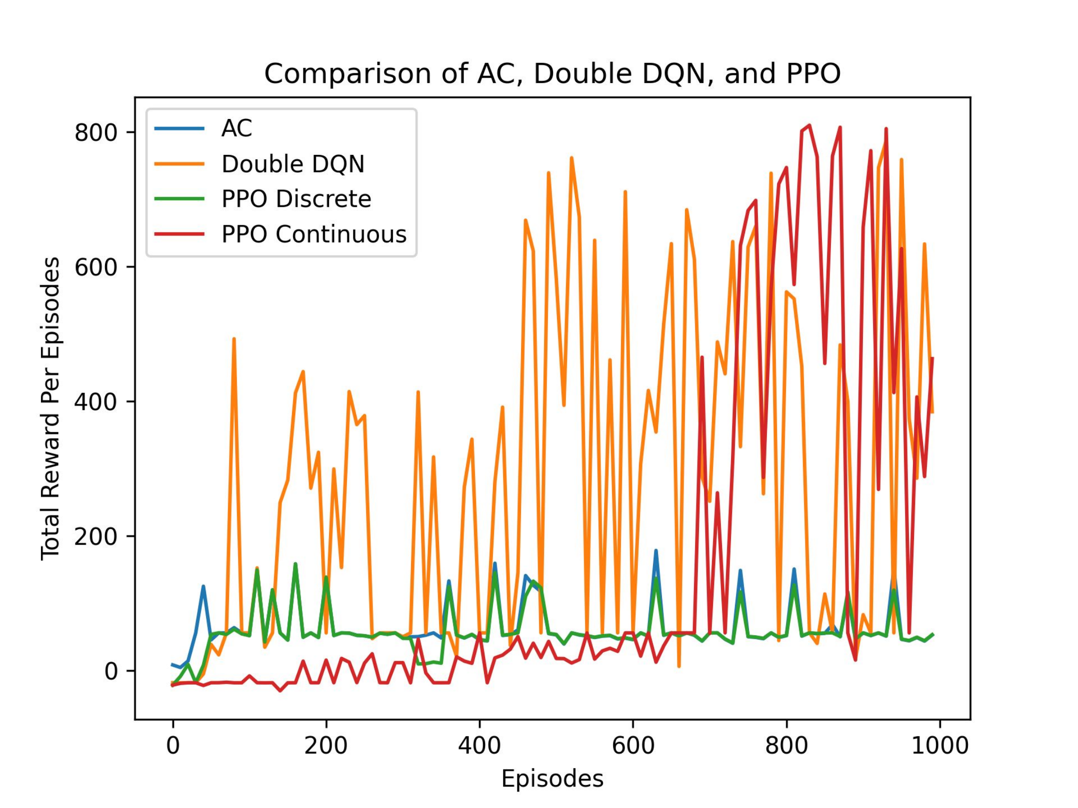
actor_lr = 1e-3
critic_lr = 1e-3
gamma = 0.98
epsilon = 0.1
batch_size = 64
buffer = 10000
target_update = 10/10000 steps

https://github.com/NoahF1205/CS138-Final-Project

# Result 1: Training Time



# Result 2: Total reward per episode



## Agent Performance