

# LAB 3

TA 陳昱丞

2023/10/24

yucheng.cs11@nycu.edu.tw

**Deadline: 2023/11/12 (Sun) 23:59**

**Demo: 2023/11/13 (Mon) 18:00**

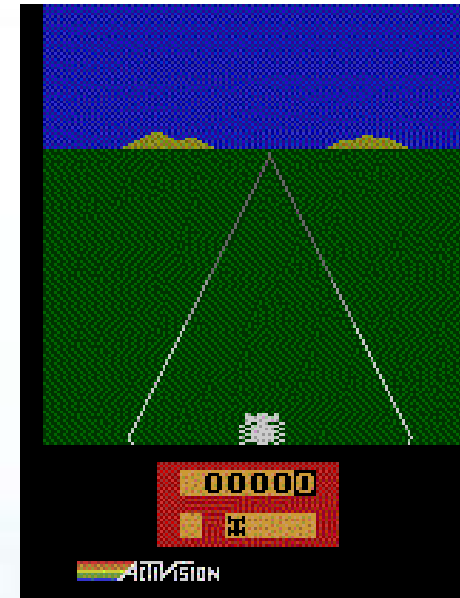
In this lab,

**Must use sample code,  
otherwise no credit.**

# Enduro-v5

- Introduction:
  - You are a racer in the National Enduro, a long-distance endurance race. You must overtake a certain amount of cars each day to stay on the race. The first day you need to pass 200 cars, and 300 for each following day. The game ends if you do not meet your overtake quota for the day
- Observation space:
  - The whole image
- Action space:

Num	Action
0	NOOP
1	UP
2	RIGHT
3	LEFT
4	DOWN
5	UPRIGHT
6	UPLEFT
7	DOWNRIGHT
8	DOWNLEFT



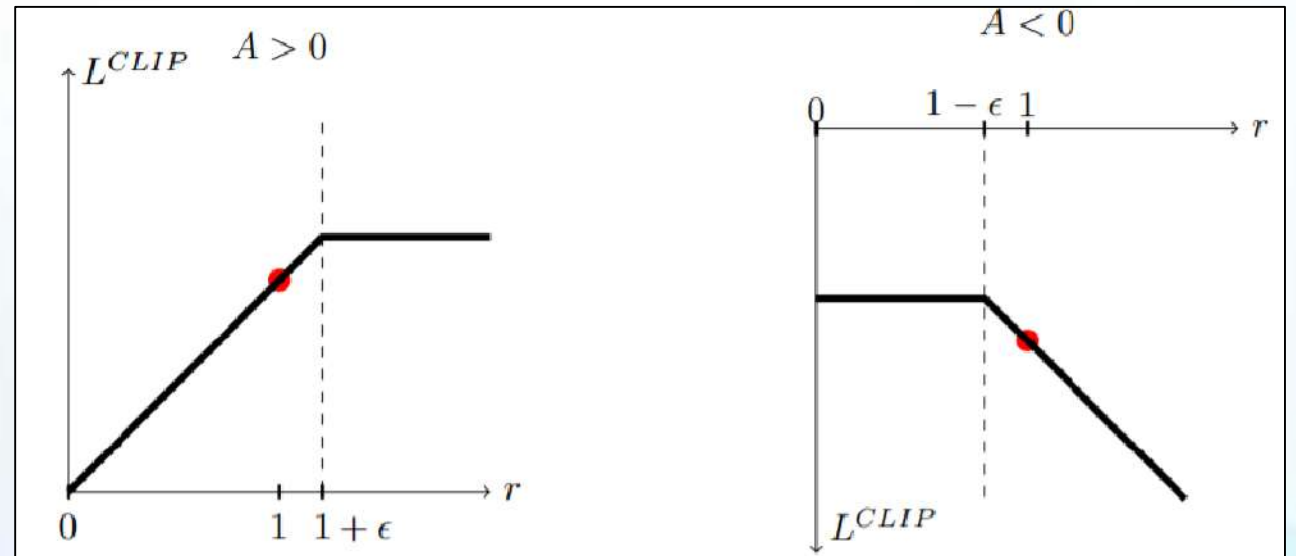
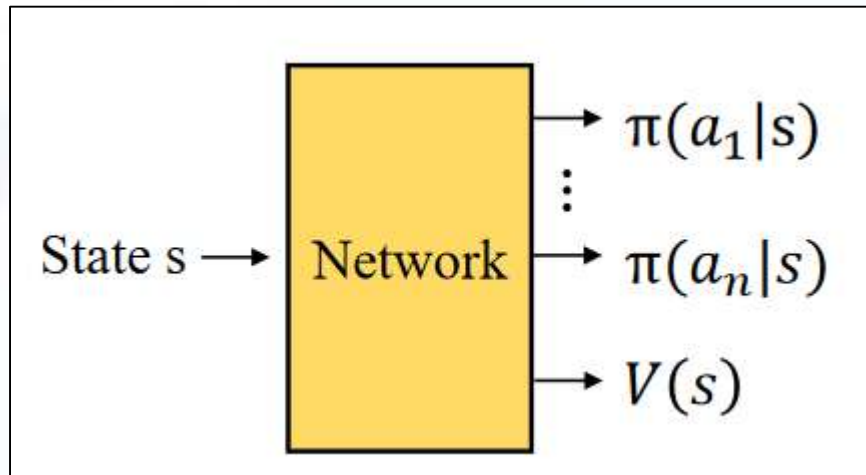
<https://www.gymnasium.dev/environments/atari/enduro/>

# Proximal Policy Optimization (PPO-clip)

$$\max_{\theta} E_{s \sim d_{\mu}^{\pi_{\theta_k}}, a \sim \pi_{\theta_k}(\cdot|s)} [\min \left( \frac{\pi_{\theta}(a|s)}{\pi_{\theta_k}(a|s)} A^{\theta_k}(s, a), \text{clip}\left(\frac{\pi_{\theta}(a|s)}{\pi_{\theta_k}(a|s)}, 1 - \epsilon, 1 + \epsilon\right) A^{\theta_k}(s, a) \right)]$$

*ratio* \*  $A(s, a)$

*clipped\_ratio* \*  $A(s, a)$



# Proximal Policy Optimization (PPO)

## Algorithm –Proximal Policy Optimization (PPO):

---

```
for iteration=1, 2, ... do  
  for actor=1, 2, ...,  $N$  do  
    Run policy  $\pi_{\theta_{\text{old}}}$  in environment for  $T$  timesteps  
    Compute advantage estimates  $\hat{A}_1, \dots, \hat{A}_T$   
  end for  
  Optimize surrogate  $L$  wrt  $\theta$ , with  $K$  epochs and minibatch size  $M \leq NT$   
   $\theta_{\text{old}} \leftarrow \theta$   
end for
```

---

# TODO

- Solve Enduro-v5 using PPO.
- (Bonus) Answer questions:
  1. PPO is an on-policy or an off-policy algorithm? Why?
  2. Explain how PPO ensures that policy updates at each step are not too large to avoid destabilization.
  3. Why is GAE-lambda used to estimate advantages in PPO instead of just one-step advantages?  
How does it contribute to improving the policy learning process?
  4. Please explain what the lambda parameter represents in GAE-lambda, and how adjusting the lambda parameter affects the training process and performance of PPO? Please explain what the lambda parameter represents in GAE-lambda, and how adjusting the lambda parameter affects the training process and performance of PPO?

# TODO

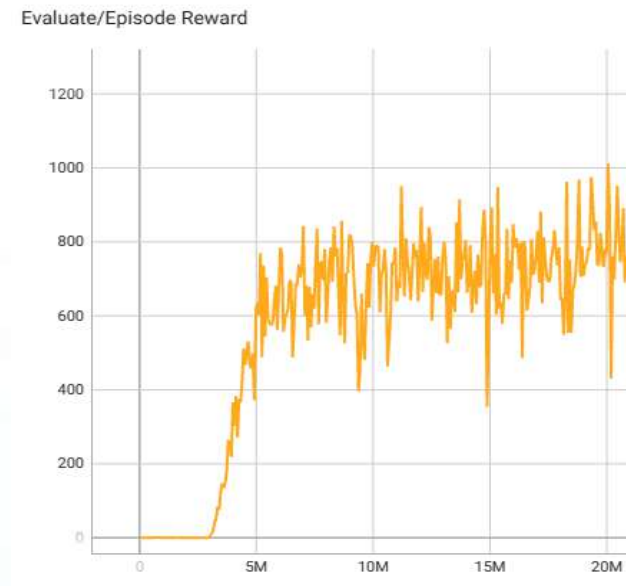
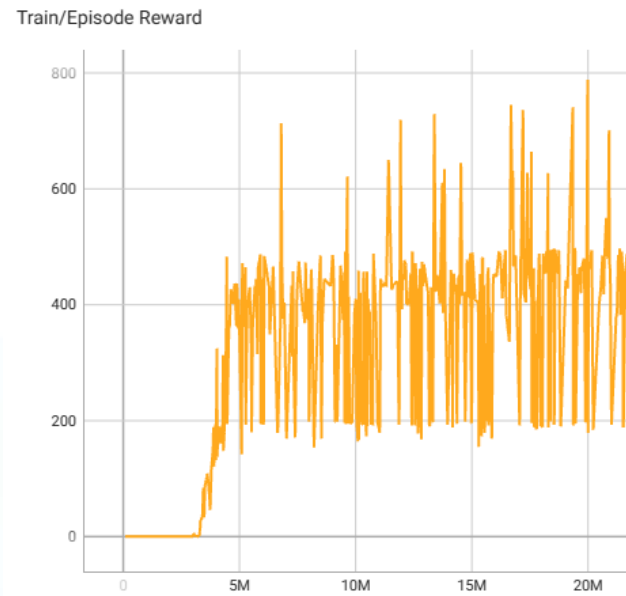
- Find the #TODO comments and hints, remove the raise NotImplementedError.
- Implement the **forward** function in neural network.
- Inherit from the “**PPOBaseAgent**” and override the “**decide\_agent\_actions**” and “**update**” functions.
- **Screenshot of Tensorboard training curve and testing results and put it on the report.**



# TODO

- Screenshot of Tensorboard training curve and testing results and put it on the report.

Training curve:

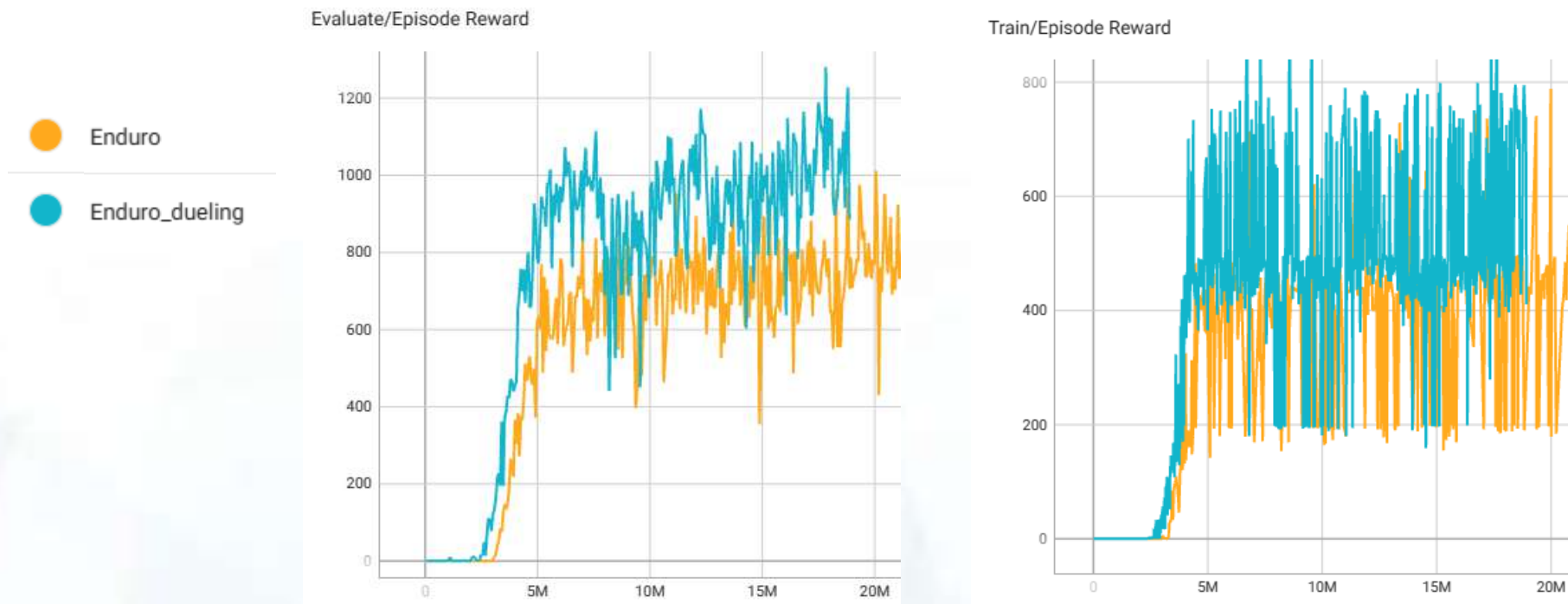


Testing results (5 games):

```
Episode: 1    Length: 13301    Total reward: 1040.00
Episode: 2    Length: 13307    Total reward: 1066.00
Episode: 3    Length: 9969     Total reward: 705.00
Episode: 4    Length: 13286    Total reward: 1059.00
Episode: 5    Length: 16630    Total reward: 1318.00
average score: 1037.6
```

# TODO

- Screenshot of Tensorboard training curve and testing results and put it on the report.



# TODO

- Inherit from the “**PPOBaseAgent**” and override the “**decide\_agent\_actions**” and “**update**” functions.

base\_agent.py

```
class PPOBaseAgent(ABC):
```

```
...
```

```
@abstractmethod
```

```
def decide_agent_actions(...):
```

```
...
```

```
@abstractmethod
```

```
def update(...):
```

```
...
```

```
def train(...):
```

```
...
```

```
def evaluate(...):
```

```
...
```

ppo\_agent\_atari.py

```
class AtariPPOAgent(PPOBaseAgent):
```

```
...
```

```
def decide_agent_actions(...):
```

```
...
```

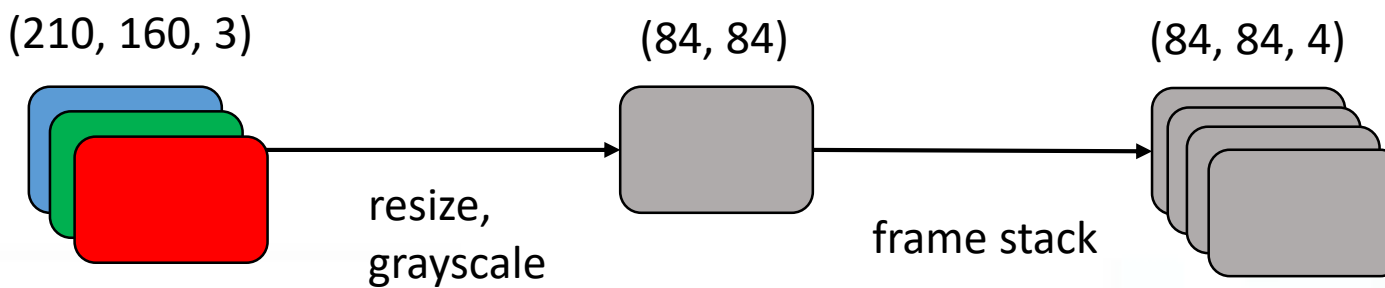
```
def update(...):
```

```
...
```

# Hint

- You can add any trick you want. E.g.:

- Frame stack
- Clip reward
- Grayscale observation
- .....



- You can use OpenAI Gym Wrapper.
  - <https://www.gymnasium.dev/api/wrappers/>

# Scoring Criteria

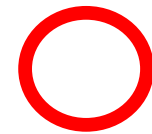
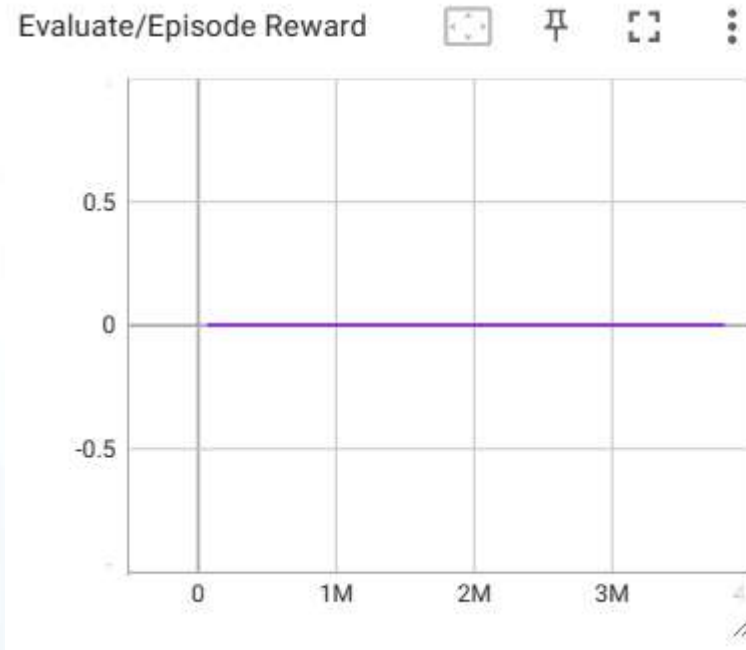
- Your Score = report (30%) + report bonus (20%) + demo performance (50%) + demo questions (20%)
- Report contains two parts:
  - Experimental Results (30%)
    - Screenshot of Tensorboard training curve and testing results on PPO.
  - Answer the questions (bonus) (20%)
    1. PPO is an on-policy or an off-policy algorithm? Why? (5%)
    2. Explain how PPO ensures that policy updates at each step are not too large to avoid destabilization. (5%)
    3. Why is GAE-lambda used to estimate advantages in PPO instead of just one-step advantages? How does it contribute to improving the policy learning process? (5%)
    4. Please explain what the lambda parameter represents in GAE-lambda, and how adjusting the lambda parameter affects the training process and performance of PPO? (5%)

# Scoring Criteria

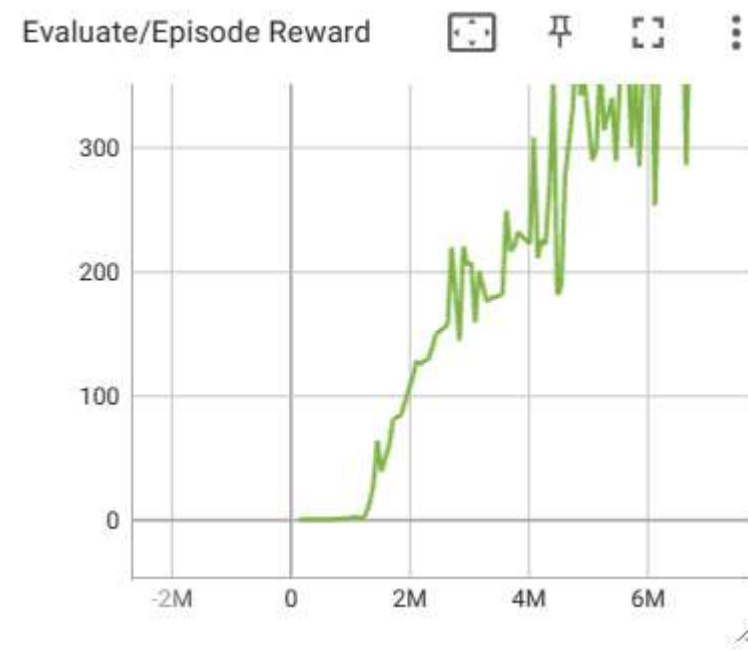
- Screenshot of Tensorboard training curve and testing results and put it on the report.



No score



Get score (30%)



# Scoring Criteria - Demo Performance

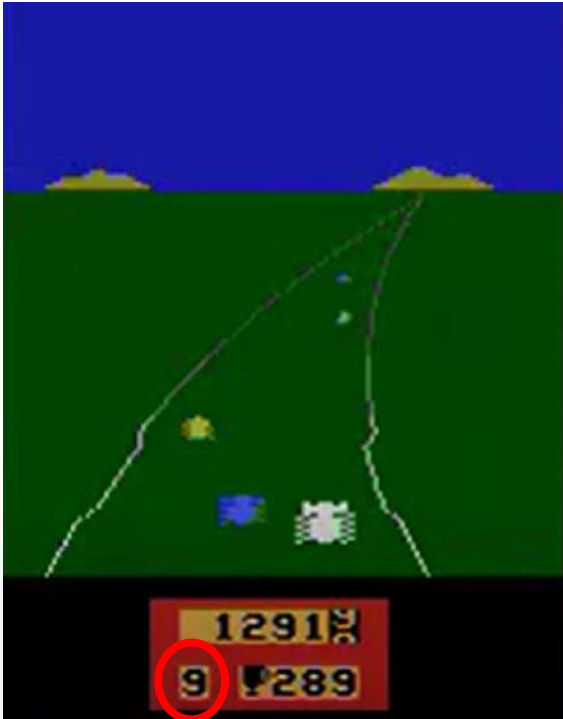
- Test your best model for **one game**.
- You have to show the video while testing. You can use `env.render()` or `save video` function to achieve this.
- You can use a fixed random seed to reproduce your best game score



Terminated in day 9

# Scoring Criteria - Demo Performance

- Demo performance – Score table:



Day	Cars	Reward	Points (50%)
1	200	0~200	0
2	300	200~500	10
3	300	500~800	20
4	300	800~1100	25
5	300	1100~1400	30
6	300	1400~1700	35
7	300	1700~2000	40
8	300	2000~2300	45
9	300	2300~2600	50

Terminated in day 9



# Tensorboard Remote Server

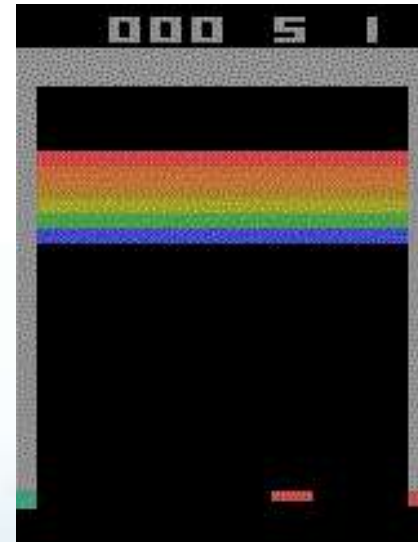
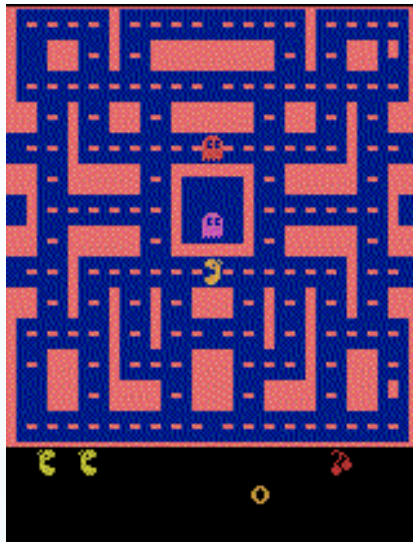
- `ssh -p [your port] -L 6006:localhost:6006 pp037@140.113.215.196`
- `tensorboard --logdir log/dqn`
- Open your browser locally and input `127.0.0.1:6006`

# Recommended Package Version

- gym 0.26.2
- numpy 1.25.2
- pytorch 2.0.1
- tensorboard 2.14.0
- opencv-python 4.8.0.76
- moviepy 1.0.3

# Reminders

- Your network architecture and hyper-parameters **can** differ from the defaults.
- Ensure the **shape** of tensors all the time especially when calculating the **loss**.
- **with no\_grad()** : scope is the same as **xxx.detach()**
- Be aware of the **indentation** of hints.
- **You can use simpler environments to check if your PPO implementation is correct, such as MsPacman, Breakout.**



# References

1. Mnih, Volodymyr et al. "Playing Atari with Deep Reinforcement Learning." ArXiv abs/1312.5602 (2013).
2. Silver, David et al. "Deterministic Policy Gradient Algorithms." ICML (2014).
3. Schulman, John, et al. "Proximal policy optimization algorithms." arXiv preprint arXiv:1707.06347 (2017).
4. OpenAI. "OpenAI Gym Documentation." Retrieved from Getting Started with Gym: <https://gym.openai.com/docs/>.
5. PyTorch. "Reinforcement Learning (DQN) Tutorial." Retrieved from PyTorch Tutorials: [https://pytorch.org/tutorials/intermediate/reinforcement\\_q\\_learning.html](https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html).
6. Huang, et al., "The 37 Implementation Details of Proximal Policy Optimization", ICLR Blog Track, 2022. url: <https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/>