Lab 3: Proximal Policy Optimization (PPO)

Lab Objective:

In this lab, you will learn and implement the Proximal Policy Optimization algorithm by solving Enduro-v5.

Important Date:

1. Submission Deadline: 11/10 (Sun) 23:59
2. Demo date: 11/11 (Mon)

Turn in:

1. Experiment report (.pdf)
2. Source code [NOT including model weights]

Notice: zip all files with name "RL_LAB3_StudentId_Name.zip",
e.g.: 「RL_LAB3_312581015_張孝全.zip」

```
RL_LAB3_312581015_張孝全.zip
 ├── src
 │     ├── base_agent.py
 │     …
 └── report.pdf
```

(Put all your source code in **src** directory)

(Wrong format deduction: -5pts; Multiple deductions may apply.)

Lab Description:

- Understand the mechanics of the PPO algorithm.
- Comprehend the construction and design of policy networks.
- Understand the role of an experience replay buffer.
- Learn how to calculate GAE (Generalized Advantage Estimation).
- Learn how to select actions and update policy networks.

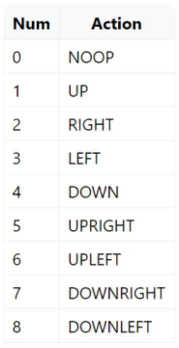
Requirements:

- Implement PPO
  - Construct the policy network.
  - Choose actions using the policy network.
  - Calculate policy probabilities and estimate value functions.
  - Compute the PPO loss function.
  - Update the policy network.
  - Understand the mechanics of PPO.

- Understand common techniques in Atari environment like frame stack, grayscale…

Game Environment – 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: By default, the environment returns the RGB image that is displayed to human players as an observation.
- Action:



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

(https://www.gymlibrary.dev/environments/atari/enduro/)

Algorithm –Proximal Policy Optimization (PPO):

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

Implementation Details:

Network Architecture

- PPO network. Please refer to the sample code for details.

Training Hyper-Parameters

- Batch size: 128
- Optimizer: Adam
- Learning rate: 2.5e-4
- Gamma (discount factor): 0.99
- Lambda: 0.95
- Value coefficient: 0.5

- Entropy coefficient: 0.01
- Update epoch: 3
- Clip epsilon: 0.2
- Max gradient norm: 0.5
- Horizon: 128
- Update network every 10000 steps

You can tune the hyperparameter yourself.

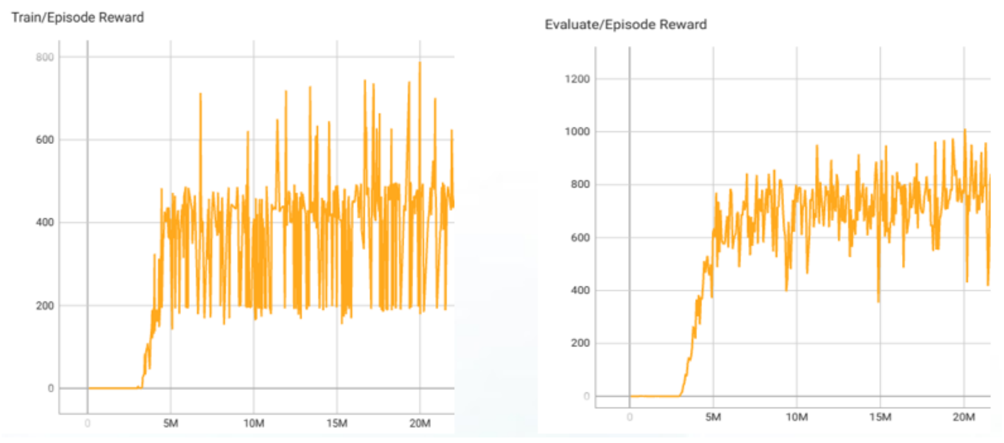
Scoring Criteria:

Show your work, otherwise no credit will be granted.

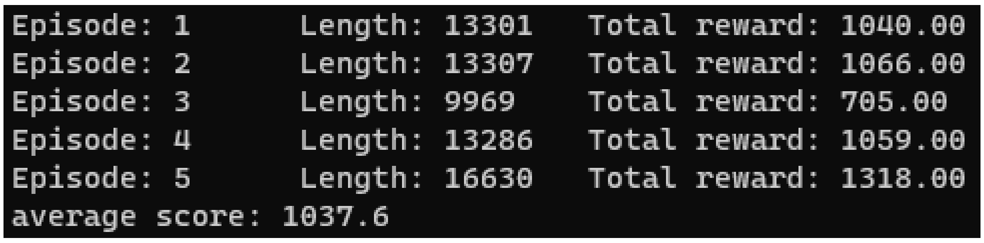
- Report (30% + Bonus 20%)
  - Screenshot of Tensorboard training curve and testing results on PPO.

    E.g.

    Training curve:



    Testing results (5 games):

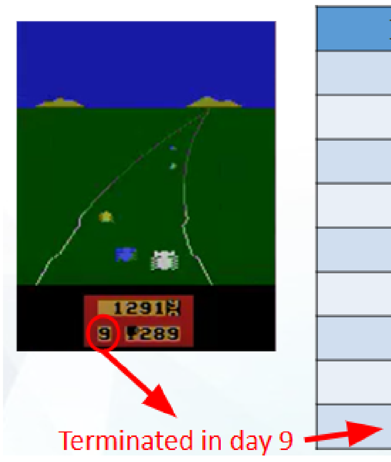


```
Episode: 1      Length: 13301   Total reward: 1040.00
Episode: 2      Length: 13307   Total reward: 1066.00
Episode: 3      Length: 9969    Total reward: 705.00
Episode: 4      Length: 13286   Total reward: 1059.00
Episode: 5      Length: 16630   Total reward: 1318.00
average score: 1037.6
```

  - Bonus: (20%)
    - PPO is an on-policy or an off-policy algorithm? Why? (5%)
    - Explain how PPO ensures that policy updates at each step are not too large to avoid destabilization. (5%)
    - 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%)

- 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%)

- Demo Performance (70%)
  - Demo performance – Score table (50%):
    - 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.



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

  - Questions. (20%)

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.

[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/