535507 Spring 2025: Deep Learning **(Due: 2025/05/20, Tuesday, 23:59)**

Lab 7: Policy-Based Reinforcement Learning

Instructor: Ping-Chun Hsieh
TAs: Wei Hung, Po-Hsuan Wen

Intended Learning Outcomes:

- Understand the core components of benchmark policy-based RL methods, e.g., A2C and PPO.
- Gain hands-on experience implementing A2C and PPO with GAE in PyTorch.
- Learn how to use policy-based RL methods on classic control tasks and robot locomotion tasks, both with continuous actions.

1 Background

In this assignment, you will implement and experiment with Advantage Actor Critic (A2C) [1] and Proximal Policy Optimization (PPO) methods [2] to solve classic control and robot locomotion tasks using PyTorch and OpenAI Gym environments. The homework consists of three parts, progressing from basic A2C implementation to incorporating the clipping advanced mechanism and the generalized advantage estimator (GAE) techniques in PPO to improve training stability and sample efficiency.

1.1 A2C with Value Function Approximation

Design Principles

- **Actor-Critic Framework:** The actor selects actions via a policy $\pi_\theta(a|s)$, and the critic evaluates states using a value function $V_w(s)$.
- **Advantage Estimation:** The advantage function $A^{\pi_\theta}(s,a)$ combines the techniques of baseline functions and critic learning in achieving variance reduction. Intuitively, the advantage function quantifies how much better an action is compared to the average expected return from that state.

Objective Functions Let θ and w be the parameters of the actor (policy) network and the critic network, respectively. Accordingly, $\pi_\theta(a|s)$ denotes the parameterized policy, $V_w(s)$ denotes estimated value of state s.

- Actor Objective:

$$J_{\text{actor}}(\theta) = \mathbb{E}_{s\sim d_\mu^{\pi_\theta}, a\sim\pi_\theta(\cdot|s)}\left[\log \pi_\theta(a|s)\cdot A^{\pi_\theta}(s,a)\right] \approx \frac{1}{|B|}\sum_{(s,a,r,s')\in B} \log \pi_\theta(a|s)\hat{A}_w(s,a), \tag{1}$$

  where the advantage of each transition (s,a,r,s') is estimated by

$$\hat{A}_w(s,a) = r + \gamma V_w(s') - V_w(s). \tag{2}$$

  The size of mini-batch B determines the update frequency, i.e., one gradient update per B environment steps. In practice, one can simply choose $|B| = 1$.

- Critic Loss:

$$L_{\text{critic}}(w) = \mathbb{E}_{(s,a,r,s')}\Big[\left(r + \gamma V_w(s') - V_w(s)\right)^2\Big] \approx \frac{1}{|B|}\sum_{(s,a,rs')} \left(r + \gamma V_w(s') - V_w(s)\right)^2. \tag{3}$$

The size of mini-batch B determines the update frequency, i.e., one gradient update per B environment steps. In practice, one can simply choose $|B| = 1$.

The pseudo code of A2C is provided in Algorithm 1.

Algorithm 1 Advantage Actor-Critic (A2C)

Initialize actor network $\pi_\theta(a|s)$ and critic network $V_w(s)$
for each episode **do**
  Initialize state s_0
  for $t = 0$ to T **do**
    Sample action $a_t \sim \pi_\theta(a_t|s_t)$
    Execute a_t, observe r_t, s_{t+1}
    Compute TD-error: $\delta_t = r_t + \gamma V_w(s_{t+1}) - V_w(s_t)$
    Update critic: $w \leftarrow w - \alpha_c \nabla_w \delta_t^2$
    Update actor: $\theta \leftarrow \theta + \alpha_a \nabla_\theta \log \pi_\theta(a_t|s_t) \cdot \delta_t$
    $s_t \leftarrow s_{t+1}$
  end for
end for

1.2 PPO-Clip With GAE

Design Principles PPO-Clip improves on A2C methods by achieving stability and reliability in training, while remaining simpler than Trust Region Policy Optimization (TRPO). The core design principles are:

- **Clipped Surrogate Objective**: Prevents the new policy from deviating too far from the old one by clipping the probability ratio, maintaining stability during updates.
- **Generalized Advantage Estimation**: Provides a bias-variance trade-off mechanism for estimating advantages, smoothing the learning signal.
- **Multiple gradient updates per batch**: Allows several passes over the same data to improve data efficiency.

Objective Functions Let $\pi_\theta(a|s)$ be the current policy, $\pi_{\theta_{\text{old}}}(a|s)$ be the policy before update. Moreover, define $\rho_t(\theta) := \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ to be the action probability ratio. We use $V_\phi(s)$ to denote the value function with parameters ϕ. ϵ is the clipping parameter (e.g., 0.2).

- Actor Objective (Clipped Surrogate)

$$L_{\text{clip}}(\theta) = \mathbb{E}_{(s,a,r,s')} \left[\min \left(\rho_{s,a}(\theta) A^{\text{GAE}}(s,a),\ \text{clip}(\rho_{s,a}(\theta), 1-\epsilon, 1+\epsilon) A^{\text{GAE}}(s,a) \right) \right], \tag{4}$$

$$\approx \frac{1}{|B|} \sum_{(s,a,r,s') \in B} \min \left(\rho_{s,a}(\theta) A^{\text{GAE}}(s,a),\ \text{clip}(\rho_{s,a}(\theta), 1-\epsilon, 1+\epsilon) A^{\text{GAE}}(s,a) \right) \tag{5}$$

  where $A^{\text{GAE}}(s,a)$ denotes denotes the advantage function estimated by the GAE.

- Critic Loss: We need to learn the value function, which is subsequently used in advantage estimation by GAE. In PPO, we use exactly the same critic loss as in A2C, but usually with larger batch size $|B| > 1$.
- Overall Objective Function:

$$J_{\text{PPO}} = J_{\text{clip}}(\theta) - c_1 L_{\text{critic}}(\phi) + c_2 \mathbb{E}_{s \sim d_\mu^{\pi_\theta}} \left[H(\pi_\theta(\cdot|s)) \right] \tag{6}$$

  where $H(\pi_\theta(\cdot|s))$ is the entropy bonus for exploration.

  For more implementation tricks of PPO, you can refer to this blogpost: https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/

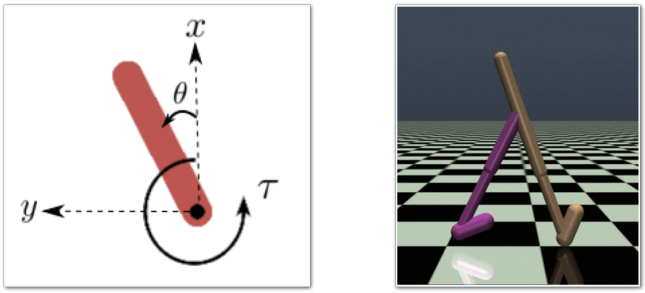
2 Lab Description

2.1 Task 1: Advantage Actor Critic (A2C) in a toy environment

Goal: Implement the A2C algorithm using PyTorch to solve the `Pendulum-v1` (https://gymnasium.farama.org/environments/classic_control/pendulum/) from OpenAI Gym.

Requirements:

- Use `a2c.py` as the starting point for your implementation.
- Use a simple fully connected neural network as the policy network. Throughout Lab7, we presume to use Gaussian policies for all the tasks.
- Evaluate and plot the total episodic rewards versus environment steps (preferably via Weight and Bias). Notably, the maximum episodic reward of Pendulum is around -130.



(a) Pendulum (b) Walker2d

Figure 1: The evaluation domains used in Lab 7.

2.2 Task 2: PPO-Clip with GAE on Pendulum

Goal: Adapt your A2C implementation to the PPO-Clip algorithm by adding two main components: (1) Clipped objective and (2) Generalized advantage estimator (GAE). Then, you shall evaluate your PPO-Clip on Pendulum again and observe the empirical differences.

Requirements:

- Add the clipped surrogate objective and the GAE functionality in your previous A2C implementation.
- Evaluate and plot the total episodic rewards versus environment steps (preferably via Weight and Bias).
- Compare training performance (e.g., learning speed, stability of learning curves) against A2C using the `Pendulum` environment.

2.3 Task 3: PPO-Clip on MuJoCo Locomotion Tasks

Goal: Adapt your PPO-Clip and the corresponding hyperparameters to one widely-used benchmark MuJoCo locomotion task – `Walker2d-v4`.

Requirements:

- Search for a hyperparameter configuration of PPO-Clip for MuJoCo tasks. You may refer to the original PPO paper (https://arxiv.org/abs/1707.06347) as a starting point.
- Compare training performance (e.g., learning speed, stability of learning curves) against A2C using the `Walker2d` environment.
- Conduct an empirical study on the effect of (1) clipping parameter and (2) entropy coefficient.

3 Model and Package Configurations

- All the modules or classes (listed below) are already provided in the file `a2c.py, ppo.py`.
  - Actor
  - Critic
  - A2CAgent
  - PPOAgent
- The model sizes can be freely configured by yourself. That being said, it is expected that both the policy and critic networks with only a few hidden layers (less than 5) would be sufficient for successful training.
- Training in Pendulum is usually super fast (e.g., no more than 10 minutes for 100k environment steps). On the other hand, MuJoCo typically requires about up to 1 million environment steps before converging to a well-performing policy (e.g., 2500 for Walker2d). This could convert to up to about a few hours of wall clock time. **Therefore, please do start early so that you have sufficient time for training**.
- The recommended package versions are as follows:
  - `Python` >= 3.8
  - `gymnasium` 1.1.1
  - `torch`
  - `wandb`
  - `mujoco`

4 Grading Policy

4.1 Report (50%)

- Introduction (5%): Please provide a high-level introduction to your report. You can mention the most important findings and the overall organization of this report.
- Your implementation (20%): Please briefly explain your implementation for Tasks 1-3. Specifically, please describe
  - How do you obtain the stochastic policy gradient and the TD error for A2C?
  - How do you implement the clipped objective in PPO?
  - How do you obtain the estimator of GAE?
  - How do you collect samples from the environment?
  - How do you enforce exploration (despite that both A2C and PPO are on-policy RL methods)?
  - Explain how you use Weight & Bias to track model performance and the loss values (including actor loss, critic loss, and the entropy).
- Analysis and discussions (25%)
  - Plot the training curves (evaluation score versus environment steps) for Task 1, Task 2, and Task 3 separately (10%).
  - Compare the sample efficiency and training stability of A2C and PPO.
  - Perform an empirical study on the key parameters, such as clipping parameter and entropy coefficient (15%).
- Additional analysis on other training strategies (Bonus up to 10%)

4.2 Attained Model Performance: Snapshots and Demo Video (50%)

4.2.1 Demo Video

Please record a demo video that showcases two things via screen recording:

- Your source code: Please use about 2 minute to describe your implementation.
- Your model performance: Please use about 3 minute to demo your obtained models for Task 1, Task 2, and Task 3.

The demo video shall be between 5-minute to 6-minute long. **Please prepare your demo video in English** (unless there is any special circumstances discussed beforehand with the TAs). **Notably, your model snapshots will NOT be graded if you do not provide a valid demo video.**

4.2.2 Model Snapshots

- Task 1 (15%): The grading of Task 1 would depend on the the number of environment steps needed for getting a score of at least -150 under your submitted model snapshot. **Please use the best model snapshot that you have obtained during the training process.** Let N denote the number of steps needed for your agent to get an average evaluation score above -150 on Pendulum. The score will be determined by the following equation

$$\text{Score percentage} = \left(1 - \frac{\max\{0, X - 200\text{k}\}}{800\text{k}}\right) \times 15\% \tag{7}$$

  Accordingly, to get full score for this part, your A2C shall achieve an average score above -150 (over 20 evaluation episodes) within 200k environment steps.

- Task 2 (20%): Similarly, the grading of Task 2 would depend on the the number of environment steps needed for getting a score of at least -150 under your submitted model snapshot. **Please use the best snapshot that you have obtained during the training process.** The score will be determined by the following equation

$$\text{Score percentage} = \left(1 - \frac{\max\{0, X - 200\text{k}\}}{800\text{k}}\right) \times 20\% \tag{8}$$

  Accordingly, to get full score for this part, your PPO shall achieve an average score above -150 (over 20 evaluation episodes) within 200k environment steps.

- Task 3 (15%): The grading of Task 3 would depend on the **sample efficiency of your PPO** on `Walker2d`. Please submit 5 model snapshots that are trained for 1M, 1.5M, 2M, 2.5M, and 3M environment steps.

Environment steps needed (Reaching score 2500 on Walker2d)	1M	1.5M	2M	2.5M	3M	>3M
Score Percentage	20%	16%	12%	10%	8%	5%

5 Format of Submitted Files

You should submit a single ZIP file that contains all your deliverables. Please strictly follow the naming and structure below:

- **ZIP file**: Compress all files into one archive.
- **Filename**: `LAB7_{StudentID}_{YourName}.zip`

The ZIP file should contain the following structure:

Directory Structure

```
LAB7_StudentID_YourName.zip
|-- LAB7_StudentID_YourName_Code/         <- Source code folder
|   |-- ppo_walker.py                           <- Your code files
|   |-- ppo_pendulum.py                         <- Your code files
|   |-- a2c_pendulum.py                         <- Your code files
|   |-- (any other .py files)
|-- LAB7_StudentID_YourName.pdf           <- Technical report (single PDF)
|-- LAB7_StudentID_YourName.mp4           <- Demo video (5 - 6 minutes)
|-- LAB7_StudentID_task1_a2c_pendulum.pt      <- Task 1 model snapshot
|-- LAB7_StudentID_task2_ppo_pendulum.pt          <- Task 2 model snapshot
|-- LAB7_StudentID_task3_ppo_1m.pt    <- Task 3 snapshot (step = 1M)
|-- LAB7_StudentID_task3_ppo_1p5m.pt    <- Task 3 snapshot (step = 1.5M)
|-- ...
|-- LAB7_StudentID_task3_ppo_3m.pt   <- Task 3 snapshot (step = 3M)
```

Note:

- Only source code files should go into the `Code` folder.
- All other items (report, video, models) must be placed directly in the ZIP root.
- A 5-point penalty will be applied for any incorrect filenames or folder structure.

References

[1] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937, 2016.

[2] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.