RL ASSIGNMENT 3

Rohan Jaiswal(R11600098), Shiva Sai Pavan Inja(R11895341)

Group Contribution:

Rohan was to create the policy network and define what it and the replay buffer look like, and how the former is going to collect trajectories. He also contributed to the fine-tuning of guidelines for quantifying the vanilla policy gradient and integrating likelihood weighting alongside rewards, not to mention learning and loss curves.

On the other hand, Shiva paid attention to the value network and worked out how to solve new processes, such as the reward-to-go and Generalized Advantage Estimation (GAE). Shiva was also involved in the integration of PPO-related objectives – the surrogate loss with and without clipping and building the whole PPO algorithm.

Specifically, they jointly adjusted hyperparameters and took time to infer information from the results to bring improvements to the model constantly.

Code Explanation

1. ***Environment Interaction Loop***
   - The first loop controls the Pendulum-v1 environment.
   - An observation or an observation array returns a sampled action after having been passed through the PolicyNetwork.
   - The action is then taken in the environment with env.step(action) which gives the next observation, and reward if the episode has ended or the episode is truncated in the done or truncated signal.
   - This loop gathers trajectories (sequences of states, actions, and rewards) that is used next for training purposes.

2.Experience Replay Buffer

- In 'ReplayBuffer', (state, action, reward, log_prob) are also stored in First-InFirst-Out (FIFO) manner and has size constraint.
- Using random samples during training, attempts are made to minimise dependence of data with purpose of enhancing the efficiency of data as well as the Sample Increase.

3.Episode Reward Processing

- Discounted rewards-to-go \(R(s_1) \) are computed:

$$R(s_1) = \sum_{t=1}^{\infty} \gamma^{t-1} r_t$$

- The rewards are scaled to overcome numerical stability issues that appear in reward calculations.

4.Vanilla policy Gradient Agent

- The Vanilla Policy Gradient aims to maximize:

$$\nabla_\theta J(\theta) = \mathbb{E}[\nabla_\theta \log \pi_\theta(a|s) R(s)]$$

- The gradient is found by summing these quantities since they are both proportional to the log probabilities of actions weighted by the rewards-to-go.

5. Gaussian Policy Implementation

- The Vanilla Policy Gradient seeks to maximize:

$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$$

$$A_t = \sum_{k=0}^{\infty} (\gamma\lambda)^k \delta_{t+k}$$

- The gradient is then the sum of these quantities since we showed that they are all proportional to the log probabilities of actions, given their corresponding weights of the rewards-to-go.

6. Feed- Forward Critic Network

- The `ValueNetwork` receives the state inputs and estimates the scalar value of such state by employing feed forward neural network.

7. Generalized Advantage Estimation

- Generalized Advantage Estimation (GAE) eliminate variance by adding the total reward in value prediction.

8. Surrogate Objective for Policy Gradient

- The PPO surrogate objective:

$$L^{CLIP} = \mathbb{E}\left[\min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t)\right]$$

9. Total Loss Function

Combines policy and value loss:

$$L = L^{CLIP} - c_1\mathbb{E}[(V(s) - R)^2] + c_2 H[\pi_\theta]$$

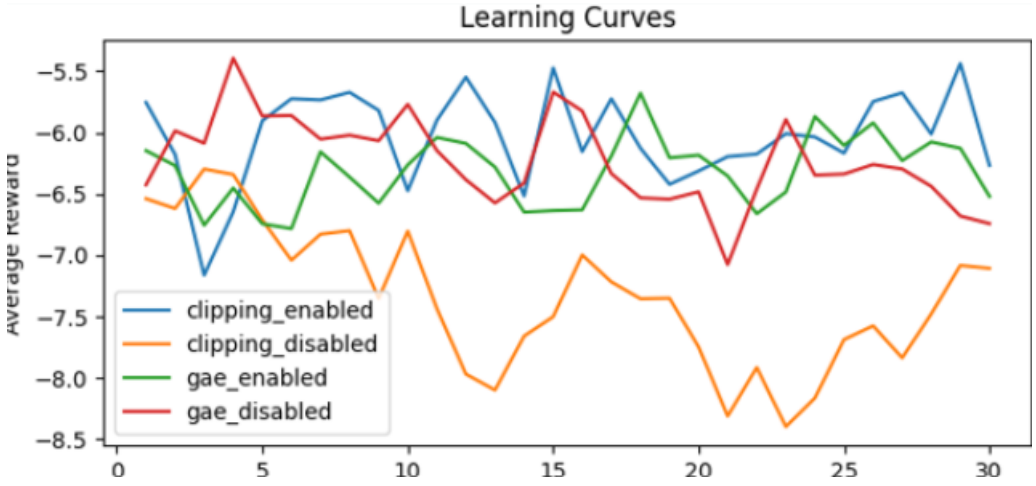
10. Combining Into PPO Algorithm

- The main training loop is set with mountain car data collected from a single actor \(N=1\), in PPO objectives.

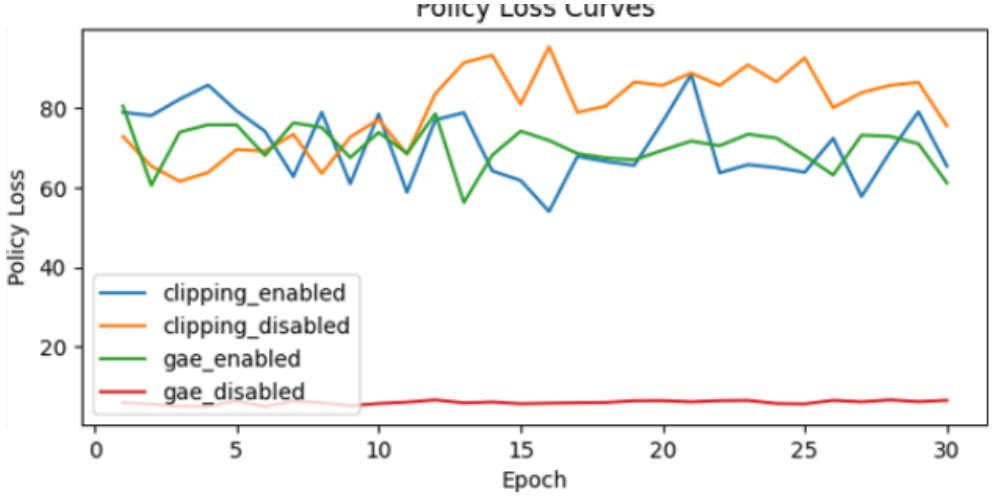
- The parameters which can be tuned to try to get a better result are epochs, gamma, clip ratio, learning rates.

Plots

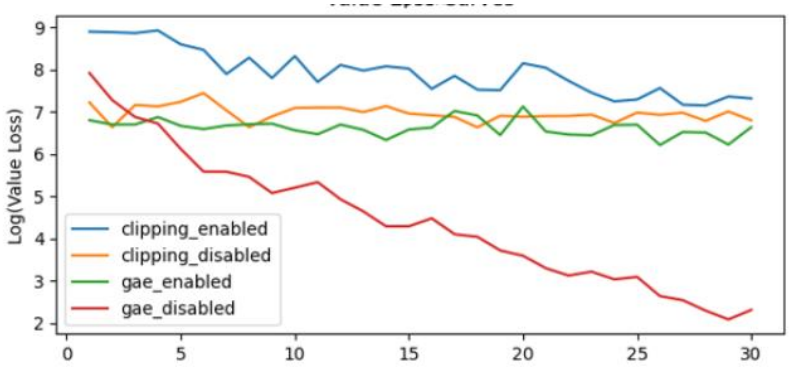
1. Learning curves



2. Policy Loss Curves



3. Value Loss Curves (Log Scale)



4. 2D Landscape of V(S):