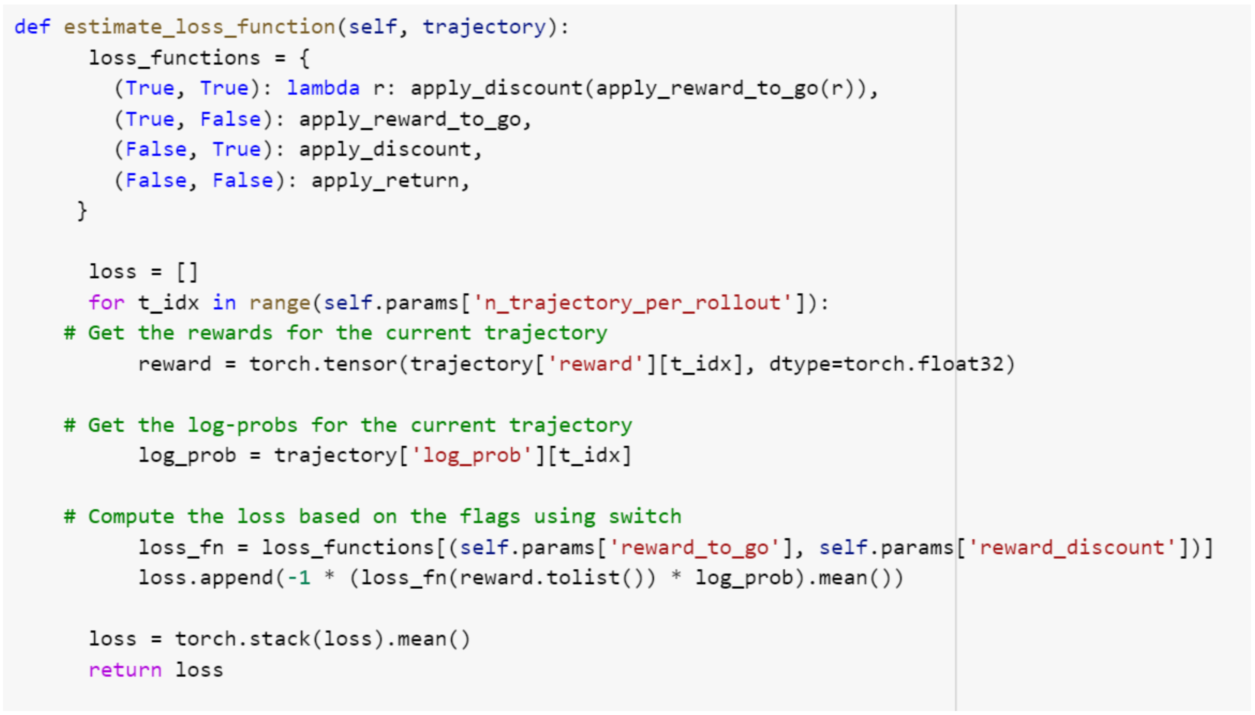
2232-CSE-6369-001
SPEC TOPS ADV INTELLIGENT SYS
ASSIGNMENT - 1

Experiment I (CartPole):

Deliverables for report:

1. Take snapshots from your code that show your implementation of eq 6, 7, 8 and include them in the report.



```
def estimate_loss_function(self, trajectory):
      loss_functions = {
        (True, True): lambda r: apply_discount(apply_reward_to_go(r)),
        (True, False): apply_reward_to_go,
        (False, True): apply_discount,
        (False, False): apply_return,
     }

      loss = []
      for t_idx in range(self.params['n_trajectory_per_rollout']):
    # Get the rewards for the current trajectory
          reward = torch.tensor(trajectory['reward'][t_idx], dtype=torch.float32)

    # Get the log-probs for the current trajectory
          log_prob = trajectory['log_prob'][t_idx]

    # Compute the loss based on the flags using switch
          loss_fn = loss_functions[(self.params['reward_to_go'], self.params['reward_discount'])]
          loss.append(-1 * (loss_fn(reward.tolist()) * log_prob).mean())

      loss = torch.stack(loss).mean()
      return loss
```

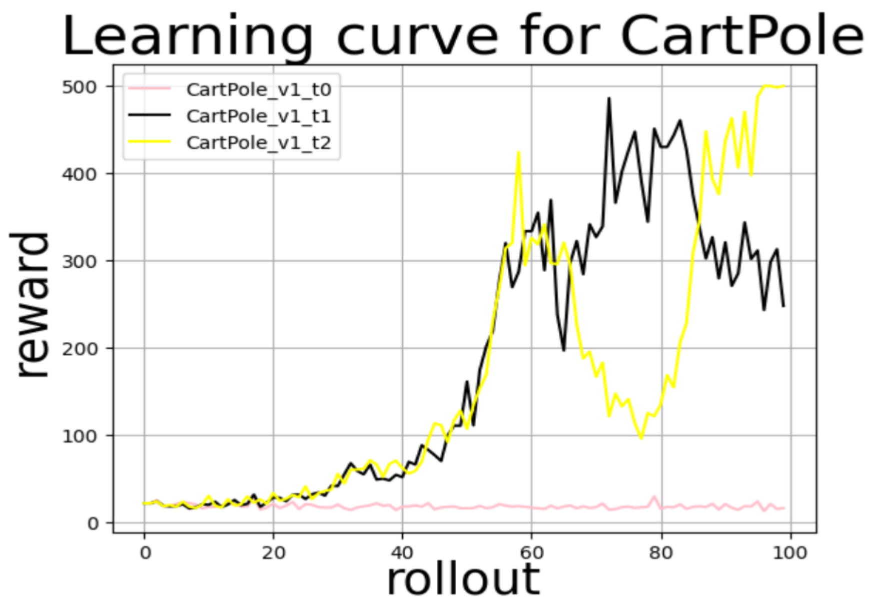
2. Create a graph that compares the learning curve from the trials above. Label the curves as t0, t1, t2. 3.

```
import argparse
import pickle
import matplotlib.pyplot as plt
import seaborn as sns

# Load rewards data for each experiment
exp_names = ['CartPole_v1_t0', 'CartPole_v1_t1', 'CartPole_v1_t2']
colors = ['pink', 'black', 'yellow']
exp_rewards = []
for name in exp_names:
    with open(name+'.pkl', 'rb') as f:
        exp_rewards.append(pickle.load(f))

# Plot the data
for i in range(len(exp_rewards)):
    sns.lineplot(data=exp_rewards[i], color=colors[i], label=exp_names[i])
plt.xlabel('rollout', fontsize=25, labelpad=-2)
plt.ylabel('reward', fontsize=25)
plt.title('Learning curve for CartPole', fontsize=30)
plt.legend()
plt.grid()
plt.show()
```

OUTPUT:



3(a). Which version of PG is best performing (among return-based, reward-to-go-based and discounted reward-based)
From the graph we can see that the Cartpole_v1_t1 (black curve) is performing better when compared with other two versions. The trail 1 is the **Reward-to-go-based version** of policy gradient which is best performing for this experiment.

Experiment II (LunarLander)

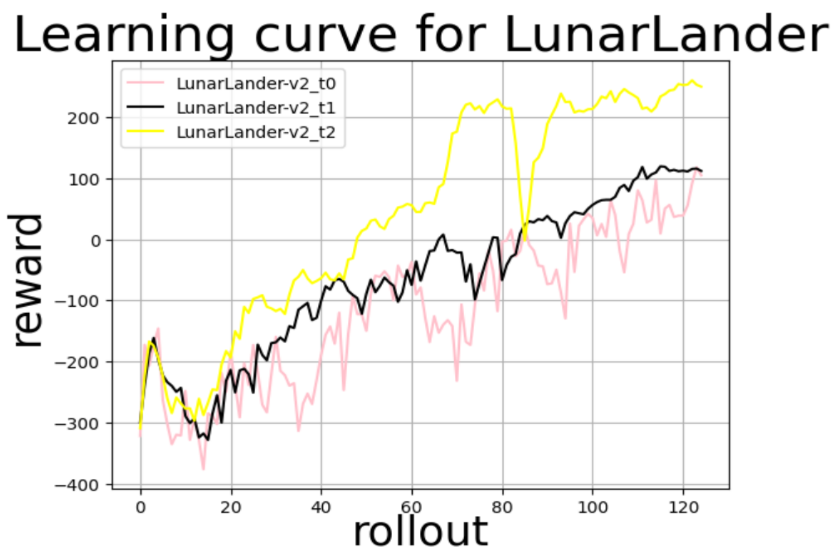
1. Create a graph that compares the learning curve from the three trials above. Label the curves as t0, t1, t2.

```python
import argparse
import pickle
import matplotlib.pyplot as plt
import seaborn as sns

# Load rewards data for each experiment
exp_names = ['LunarLander-v2_t0', 'LunarLander-v2_t1', 'LunarLander-v2_t2']
colors = ['pink', 'black', 'yellow']
exp_rewards = []
for name in exp_names:
    with open(name+'.pkl', 'rb') as f:
        exp_rewards.append(pickle.load(f))

# Plot the data
for i in range(len(exp_rewards)):
    sns.lineplot(data=exp_rewards[i], color=colors[i], label=exp_names[i])
plt.xlabel('rollout', fontsize=25, labelpad=-2)
plt.ylabel('reward', fontsize=25)
plt.title('Learning curve for LunarLander', fontsize=30)
plt.legend()
plt.grid()
plt.show()
```

OUTPUT:



2. (a) How does the number-of-trajectory-per-rollout affect learning performance? How can you justify this relationship?

For trail 0 we took the number-of-trajectory-per-rollout(ntr) value as 5, for trail 1 it is 20 and trail 2 the value is 60. When we are increasing ntr value the graph seems to be at a better reward that means the average trajectory reward is increasing which also means the agent is better learning in the environment. So for this experiment t2 gives us a best average trajectory reward which has the ntr value highest.We can justify this by looking at the graph as the number of trajectories increases the sample used for training will also be increasing so the estimation will be near to the real expected reward. But when the ntr value is increasing the time that it takes to run (learn) is also increasing.

GITHUB REPO LINK:
github

STUDENT NAME: Naveyah Injam
STUDENT ID: 1002029985