**TASK 4 - REINFORCEMENT LEARNING**

The first step of the assignment was implementing the function step() in the MLPActorCritic class, which samples an action from the pi network, computes its log-probability and then computes the value function, all of that given a certain observation.

Then, in the VPGBuffer class I compute the TD residuals. This is done in the function end\_traj() by means of the vectors "rews" and "vals", which contain the values of the reward and value function at each timestep of the trajectory. The TD residuals are computed in the following way:

self.tdres\_buf = discount\_cumsum(deltas, gamma\*lambda)

where deltas = rews[:-1] + vals[1:] - vals[:-1] (reward at time t + the difference of the value function between time t+1 and t) and discount\_cumsum() computes the discounted cumulative sum (the discount factor gamma is multiplied by lambda, in order to reduce variance). The TD residuals are then normalized by subtracting the mean and dividing by the std.

As for the update of the pi and v networks, I needed to compute the respective loss functions. For the policy, I set pi\_loss = -(logp \* td\_res).mean(), which is the value whose derivative gives back the policy gradient, with logp being the log probability given the actions and the observations of the buffer. For the value function, I set the loss to be the MSELoss between the values computed by the v network, given the buffer observations, and the target, which is assumed to be equal to rewards\_to\_go + gamma\*v\_values (of the buffer).