Exercise 6

Eugeniu Vezeteu - 886240
ELEC-E8125 - Reinforcement Learning

November 2, 2020

1 Task 1

On this task, in addition to the code from ex.5 I added another linear layer for the critic to Policy class.

```
# TODO: Add another linear layer for the critic
self.value = torch.nn.Linear(self.hidden, 1)
```

The *forward* function was also changed, in addition to the action distribution, now it returns the critic value.

```
def forward(self, x, var):
    x = self.fc1(x)
    x = F.relu(x)

    # Actor part
    action_mean = self.fc2_mean(x)
    sigma = var

    # Critic part
    # TODO: Implement
    value = self.value(x)


    action_dist = Normal(action_mean, torch.sqrt(sigma))

    return action_dist, value
```

We compute the advantage using:

$$A_\pi(s,a) = Q_\pi(s,a) - V_\pi(s)$$

and in TD(0) update advantage is:

$$A_\pi(s,a) = r(s,a,s') + \gamma * V(s') - V_\theta(s)$$

```
# TODO: Compute state values
_, V = self.policy.forward(states)
V = V.squeeze(-1)
with torch.no_grad():
    V_ = self.policy.forward(next_states)[1].squeeze(-1)
.
.
.
# TODO: Compute advantage estimates
A = rewards + (self.gamma * V_ * (1-done)) - V
```

In the code we multiply the value of the next state with (1-done), because we want the terminal state values to be zero.

To update the critic, we need to minimize the TD error between the estimated value of states and the true value. Therefore, compute the MSE loss:

$$L(\theta) = \sum_t (V_\theta(s_t) - (r + \gamma * V(s_{t+1}))^2$$

```
# TODO: Compute critic loss (MSE)
q = rewards + self.gamma * V_.detach() * (1-done)
critic_loss = torch.mean(torch.square(V-q))
```

To update the actor we need to compute:

$$\nabla_\theta J(\theta) = \sum_t \nabla_\theta log \pi_\theta(s_t, a_t) * A)$$

, [1]

```
# TODO: Calculate actor loss (very similar to PG)
actor_loss = torch.mean((-action_probs) * A.detach())

# TODO: Compute the gradients of loss w.r.t. network parameters
(critic_loss+actor_loss).backward()
```

The result of TD(0) training is presented in Figure. 1.

2 Question 1 - relationship between actor-critic and REINFORCE with baseline

In REINFORCE algorithm we use the baseline to control the variance of the method. The AC method does not differ too much from REINFORCE (sample transition, compute return, update policy,), apart from the fact that the critic has to be learned in parallel with the actor.

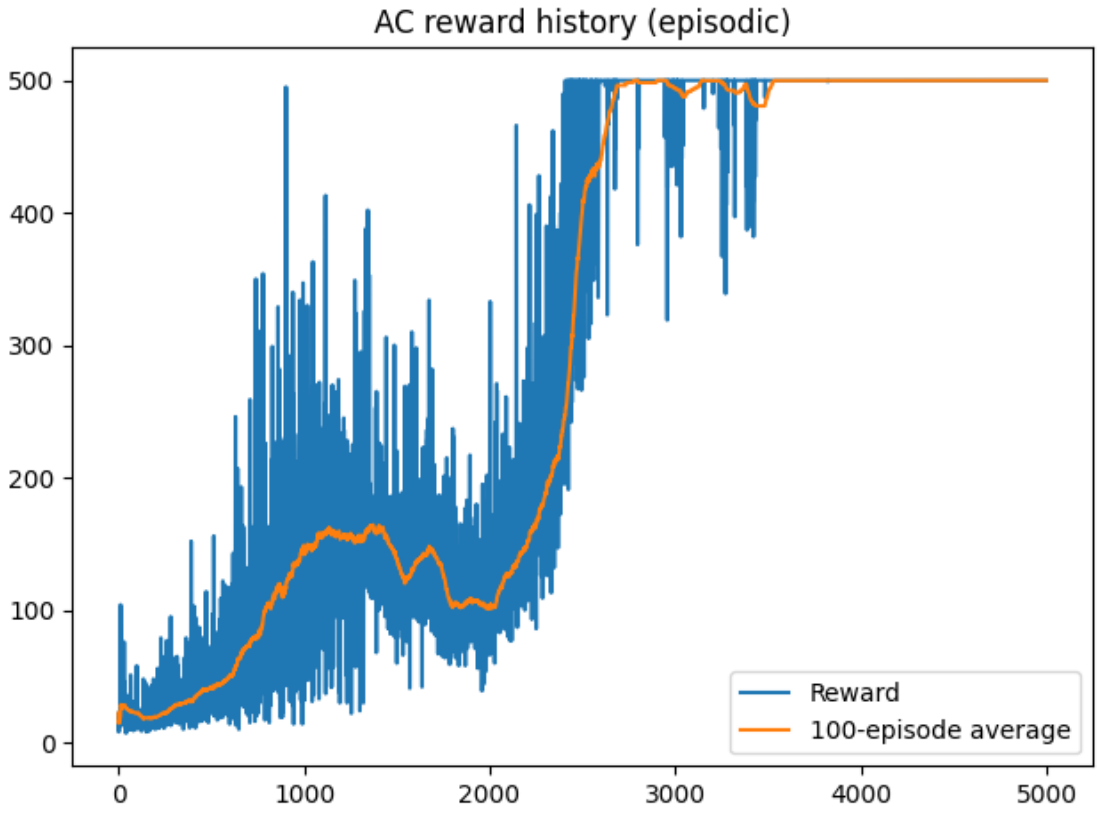

Figure 1: Task1 ACTOR-CRITIC TD(0) update, 5000 episodes

The AC is a version of Policy gradient with baseline, were baseline is learned(the Critic is the baseline).

3 Question 2 - How can the value of advantage be intuitively interpreted

Using V as a baseline, we subtract $A = R - V$, where $R_t = E[r_t + 1 + \gamma * V(s_t + 1)] = Q(s_t, a_t)$, so intuitively, this means "how much better it is to take a specific action compared to the average, general action at a given state' ,[2].

4 Task 2

In this task TD(0) updates are performed every 50 timesteps.

The following code shows the modification made to cartpole.py file.

```
update_steps = 50
time = 0
.
.
.
time+=1
if time % update_steps == update_steps - 1:
    #print(f"Update @ step {time}")
    agent.update_policy(time)
```

The result of TD(0) training updated every 50 timesteps is presented in Figure. 2.

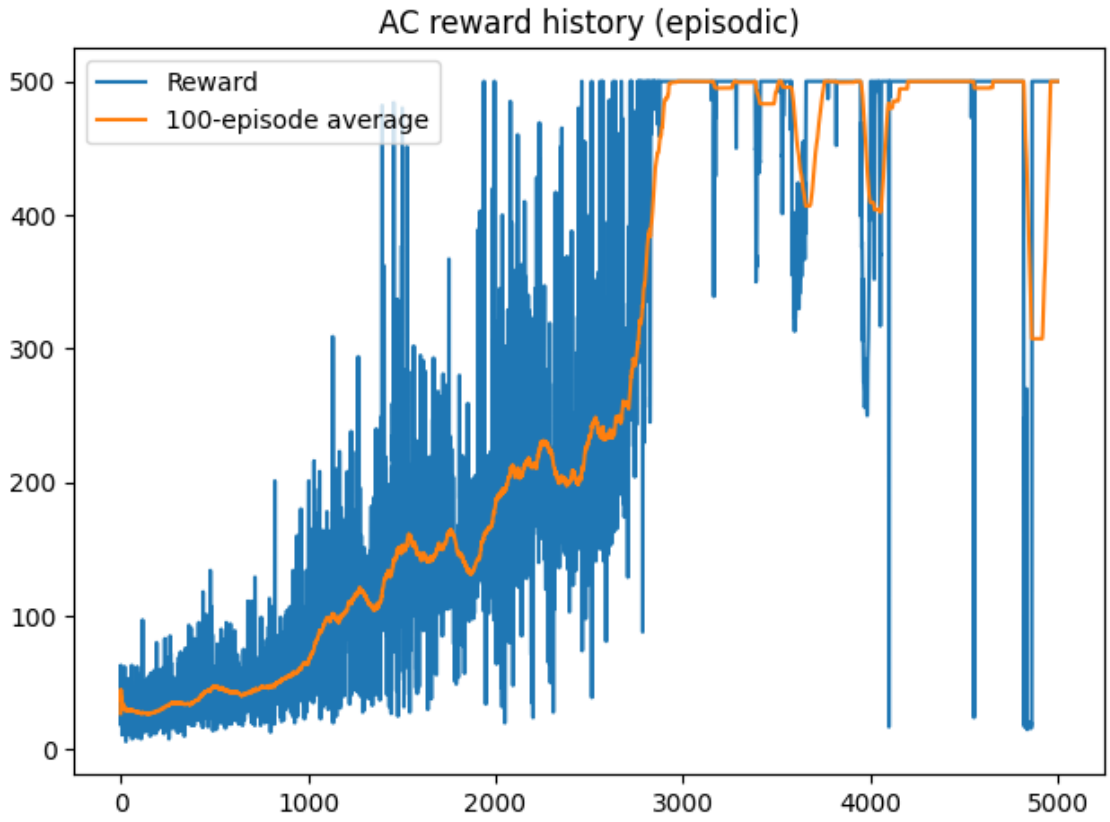

Figure 2: Task2 ACTOR-CRITIC TD(0) updated every 50 timesteps, 5000 episodes

5 Task 3

On this task I run 8 processes with 8 environments per process

```
env = ParallelEnvs(env_name, processes=8, envs_per_process=8)
```

The result is presented in Figure . 3

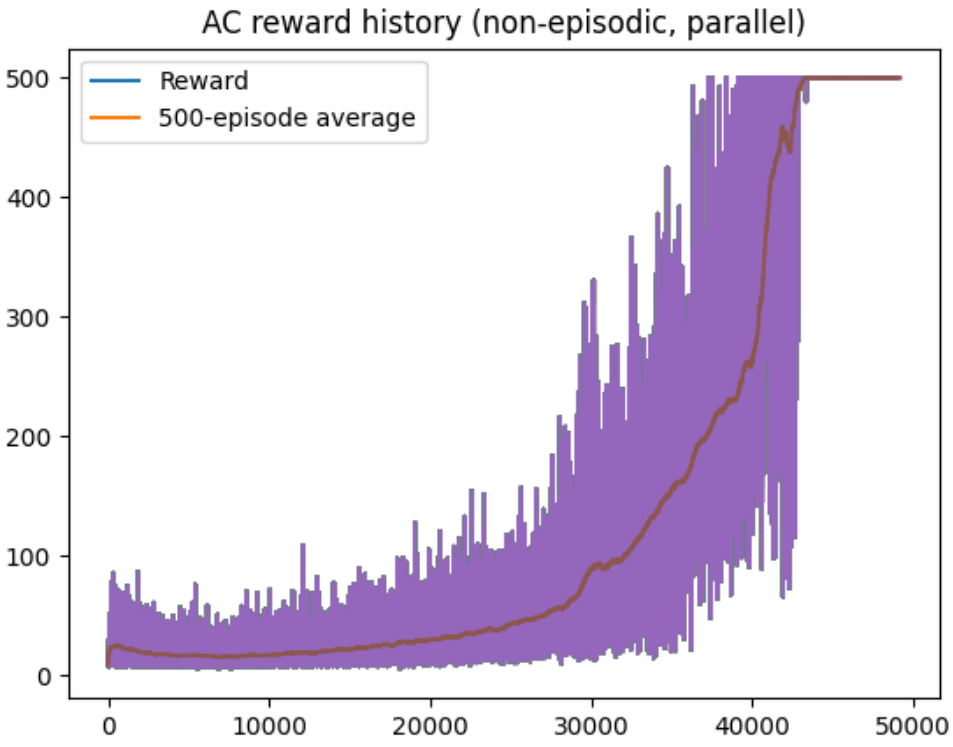

Figure 3: Task2 ACTOR-CRITIC TD(0) updated every 50 timesteps with parallel environments

6 Question 3

In the case of parallelism in Exercise 1, we create an agent and an environment for each separate process, so each agent has its own environment.
In the case of parallel data collection from this exercise, we have an agent who learns by interacting with several environments at the same time. This process is asynchronous because the speed of each CPU core is slightly different, we are doing synchronization when collecting the data from all environments.

As it was said, the data coming from different environments is less correlated then consecutive samples from a single environment. The use of parallel environments introduces an effective and efficient exploration of the state space, perhaps the added noise works as a regularization. In the case of multiple runs of the training algorithm, the data still will be correlated.

Talking about the replacement between the methods, I will say NO, because one is Data Parallelism and another is Model Parallelism, and we should not replace this approaches.

7 Question 4

The REINFORCE algorithm outperforms the other methods because it is completely unbiased and has a high variance, while the other versions uses the baseline/critic value estimators to reduce the variance. The REINFORCE high-variance sometimes finds a good policy, sometime didn't, but on average its good. Depending on the problem we can choose between one of the versions presented, in this scenario, the REINFORCE outperforms the other methods, in other environments it may not.

8 Question 5.1

In an episodic environment (MC case), the REINFORCE suffer from the high variance of the gradient, because it depends on the obtained returns R. One way to reduce variance and increase stability is subtracting the reward by a baseline, the baseline works as a bias, so increase bias to reduce variance and vice-versa.

In case of actor-critic methods, the baseline is replaced state value estimate (critic network), which is a learned version of the baseline. However, bias and variance tradeoff in case if AC methods is represented by the way we compute the advantage estimate. In other words, n-step estimation ensures a trade-off between bias (wrong updates based on estimated values as in TD) and variance (variability of the obtained returns as in MC).[3]

9 Question 5.2

As I already mention in Question 5.1, to control the bias-variance tradeoff, we should choose the n-step advantage estimation.
In the case of RL, variance refers to a noisy, but on average accurate value estimate, whereas bias refers to a stable, but inaccurate value estimate [4], we have high-variance Monte-Carlo estimate and high-bias Temporal difference estimate. By playing with n step advantage estimation, we arrive at the formulation of GAE (Generalized Advantage Estimation) which allows for an interpolation between TD and MC learning, using λ parameter. When λ is 0, we have pure TD, when λ is 1 we have MC.

10 Question 6

The main advantages are:

1. **Large & continuous action space** - PG and AC methods can handle large and continuous action spaces without discretization by using Gaussian distribution and learning the variance.

2. **Stochastic policies** - PG method can find a stochastic optimal policy, while action-value methods such as Q-learning, can't, because you cannot control the probability of an action in Q-learning.

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[2] C. Yoon, "Understanding actor critic methods and a2c."

[3] J. Vitay, *Deep Reinforcement Learning*. 2018.

[4] A. Juliani, "Making sense of the bias / variance trade-off in (deep) reinforcement learning."