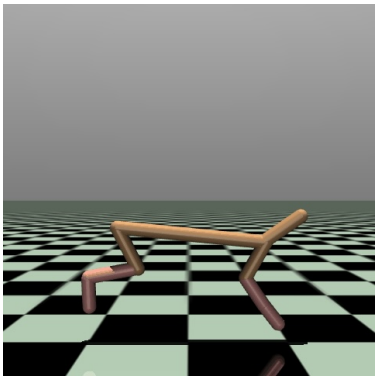
HW2: Implement Actor-Critic method in HalfCheetah environment

In this homework, you will implement Actor-Critic methods to train an agent in the HalfCheetah environment. The HalfCheetah is a 2-dimensional robot consisting of 9 body parts and 8 joints. The goal is to make the cheetah run forward as fast as possible.



1 Environment Details

Action Space:

- 6-dimensional continuous space, range $[-1, 1]$
- An action represents the torques applied at the 6 hinge joints

Observation Space:

- 17-dimensional continuous space, range $[-\infty, \infty]$
- Includes position values and velocities of various body parts

Rewards:

- The total reward is: `reward = forward_reward - ctrl_cost`
- `forward_reward`: Reward for moving forward
- `ctrl_cost`: Penalty for taking large actions

Note

For detailed information, refer to: https://gymnasium.farama.org/environments/mujoco/half_cheetah/
You can also use the old version of `gym`, refer to: https://www.gymlibrary.dev/environments/mujoco/half_cheetah/

2 Tasks

Step 1: Environment Setup

- Install MuJoCo and required dependencies, refer to: https://gymnasium.farama.org/environments/mujoco/
- Set up the HalfCheetah environment and run an agent with random policy to verify correct installation.

> Note
>
> The installation of the environment may be troublesome, please start it as early as possible.

Step 2: Actor-Critic Implementation

- Choose and implement **ONE** of the following Actor-Critic algorithms:
  - PPO (https://arxiv.org/pdf/1707.06347)
  - SAC (https://arxiv.org/pdf/1801.01290)
  - DDPG (https://arxiv.org/pdf/1509.02971)

Step 3: Results

- Plot the learning curves (reward vs episode or timestep).
- (Optional) Discuss findings or potential improvements, if there are any.

3 Code Demo

Here's a simple code example. You can also implement it in your own style.

```
import gymnasium as gym
env = gym.make("HalfCheetah-v5")
obs_dim = env.observation_space.shape[0]
action_dim = env.action_space.shape[0]
for episode in range(num_episodes):
    obs, _ = env.reset()
    done = False
    while not done:
        action = my_policy(obs)
        next_obs, reward, done, _, _ = env.step(action)
        my_buffer.push(obs, next_obs, action, reward, done)
        obs = next_obs
    batch = my_buffer.sample(batch_size)
    my_policy.train(batch)
```

4 Submission

Submit a ZIP file containing:

- Implementation code.
- A brief PDF report containing instructions for running your code, along with your results and discussions.

Submit to the course platform before **April 9, 2025, 23:59 PM**.