# CIS 620, Advanced Topics in Deep Learning, Spring 2024

## Homework 7: Off-Policy Reinforcement Learning
## Due: Monday, March 18, 11:59pm
## Submit to Gradescope

### Learning Objectives

In this assignment, you will:

- Implement DDPG and SAC models using the Stable-Baseline3 library
- Experiment with the two algorithm's key design choices, and compare their performance

### Deliverables

This is a **pair** assignment for both the written and coding portions.

1. **A PDF with your name in the agreement**

   Copy and edit this google doc to enter your names in the Student Agreement and answer the questions mentioned below. **Don't forget to add your team member on gradescope submission!**

2. **hw7.ipynb file with the functional code**

   Complete the coding assignment in the Jupyter Notebook and upload the file to Gradescope. Please leave clear and interpretable results in the submission, so we know you have completed the tasks. **Don't forget to add your team member on gradescope submission!**

### Homework Submission Instructions

#### Written Homeworks

All written homework must be submitted as a PDF to Gradescope. **Handwritten assignments (scanned or otherwise) will not be accepted.**

#### Coding Homeworks

All coding assignments will be done in Jupyter Notebooks. We will provide a .ipynb template for each assignment as well as function stubs for you to implement. You are free to use your own installation of Jupyter for the assignments, or Google Colab, which provides a Jupyter Environment connected to Google Drive along with a hosted runtime containing a CPU and GPU.

For this assignment, we have provided the code for dataset creation from a single image. The input image is provided in the folder and can also be accessed by the code for dataset creation.

## Questions

## Reference: Stable Baselines3

### Coding

1. Initialize Environment: Lunar Lander
    a. Initialize the environment
    b. Implement or import an appropriate evaluation function for a trained policy
    c. Implement a function to generate video from the trained policy
2. DDPG
    a. Implement the DDPG algorithm using the library
    b. Try at least 2 policies (see the documentation), train them, and report differences in performance
3. SAC
    i. Implement the SAC algorithm using the library
    ii. Try at least 2 policies (see the documentation), train them, and report differences in performance

### Discussion

1. Environment
    a. **What is the action and observation space? Are they discrete or continuous?**
    Action space(Continuous): 2 float numbers ranging from -1 to 1.
    The first one controls the main engine of the lander and the second one controls the lateral engines on the left and right side of the lander.

    Observation space: 6 floats (continuous) numbers representing the position(x, y), linear velocity(vx, vy), angle, and angular velocity of the lander. Also, 2 boolean(discrete) values indicate whether the legs of the lander touch the ground.

    b. **Briefly describe how the rewards are assigned.**
    The rewards:
    - is increased/decreased the closer/further the lander is to the landing pad.
    - is increased/decreased the slower/faster the lander is moving.
    - is decreased the more the lander is tilted (angle not horizontal).
    - is increased by 10 points for each leg that is in contact with the ground.
    - is decreased by 0.03 points each frame a side engine is firing.

- is decreased by 0.3 points each frame the main engine is firing.

Also, the episode receives an additional reward of -100 or +100 points for crashing or landing safely respectively.

2. **DDPG**
   a. **Which policies did you try for DDPG? What are their differences in performance (mean_rewards, time efficiency, etc.)**
   We tried the MLP policy for DDPG and we tried changing the hyperparameters to analyse their effects.
   **Hyperparameters 1:** `buffer_size=200000, learning_starts=10000, gamma = 0.98, policy_kwargs={"net_arch": [400, 300]}, total_steps = 3e5, learning_rate= 1e-3`

   **Reward 1:** `mean_reward=258.65 +/- 24.02276432414263`

   **Hyperparameters 2:**
   ```
   model = DDPG("MlpPolicy", env, action_noise=action_noise, verbose=1, learning_rate= 1e-3)
   model.learn(total_timesteps=2000, log_interval=10)
   ```
   **Reward 2:** `mean_reward=-188.66 +/- 90.46628052429114`

   b. **What is the hyperparameter *action_noise*? What value did you use?**
   action_noise is used to encourage exploration as DDPG uses deterministic policies instead of stochastic policies, the exploration becomes a challenge.

   c. **What is the hyperparameter *buffer_size* and *replay_buffer_class*? What value did you use?**
   buffer_size refers to the size of the replay buffer and replay_buffer_class refers to the implementation or design of the replay buffer being used. We used 2e5 and 2e3 for parameters set 1 and 2.

3. **SAC**
   a. **Which policies did you try for SAC? What are their differences in performance (mean_rewards, time efficiency, etc.)?**
   We tried the MLP policy for SAC and we tried changing the hyperparameters to analyze their effects.
   **Hyperparameters 1:** `ent_coef = 'auto', policy_kwargs={"net_arch": [400, 300]}, Total_steps = 1e5,  leraning_rate=3e-3`
   **Reward 1:** `mean_reward=174.17 +/- 117.07984443747523`

   **Hyperparameters 2:** `ent_coef = 'auto', policy_kwargs={"net_arch": [400, 300]}, Total_steps = 2e3, leraning_rate=1e-2`
   **Reward 2:** `mean_reward=-512.21 +/- 273.2179262121116`

b. **How do they compare with DDPG?**
   The results from DDPG were more favourable where the mean reward was around 258 and it was much lower than SAC.

c. **What is the hyperparameter *ent_coef*? What value did you use?**
   ent_coef is the entropy regularization coefficient. It is used to control the exploration-exploitation tradeoff. It is equivalent to the inverse of the reward scale in the original SAC paper. We used “auto” which lets the model choose the optimal ent_coef by itself.