ECE 276C Assignment 4:
Getting a State-of-Art Algorithm as a Baseline

This assignment will have you implement the baseline algorithm that your project.
You may submit Assignment 4 as a group or individually.

The project in this class is to implement a current state-of-art algorithm and show its success for some example cases and show how you can add to this state-of-art by solving one of its limitations. This can be done by either:

- Addressing a deficiency or non-ideality of the algorithm
- Addressing failed or poorly solved environments from the baseline algorithm
- Improving the generalizability of the algorithm over a larger set of problems.

By focusing on problem areas that are not met, you will find that it can be easy to find specific solutions to test out. There are two likely outcomes by following this approach: your proposed solutions would improve the algorithm overall or, at least, make the algorithm competent in solving a domain of problems at the loss of generality or competency in other domains.

Key deficiencies that often arise are:
- Cannot handle long-delayed rewards
- Data inefficient
- Overuse or underuse of model-based RL
- Computational inefficient
- Does not handle stochastic dynamics, rewards, policies
- Unintelligent exploration
- Does not leverage previously trained behaviors

Assignment

1. **Project Abstract:** answer the following questions between 0.5~1 page total. Use the subheadings below.

   A. **Problem Statement** – what problem will you be addressing that pertains to RL (up to 150 words)
   B. **Project Idea:** describe the project idea (up to 100 words)
   C. **What metrics will you be using to evaluate improvement:** choose metrics that line up with the paper (up to 100 words)
   D. **What is a closest *state-of-art* algorithm** – choose the paper of the algorithm that you wish to build on as a baseline. There should always be a nearest neighbor paper that you can start with. All the papers you are doing for the paper presentations would be state of art as a general guideline. (i.e. don't give us vanilla Q-learning or vanilla REINFORCE!) Describe the algorithm, discuss its advantages and limitations. (up to 200 words)
   E. **Point us to the repo online.** We *strongly* recommend you use start with the git repo for any paper (this should be standard nowadays for machine learning papers). We have no problems with you using other people's code.

2. **Implement the baseline algorithm**. Choose one category below from the OpenAI Gym envs. Show that the algorithm solves at least one of these examples (this is a given, there is no leniency if your algorithm doesn't work for *any* environment!), and test at least 2 others within this category (even if they do not get solve).
   - MuJoCo envs
   - Classic Control envs
   - Robotics
   - Atari envs

Show the score vs iterations with standard deviation for each environment. Generate animations for the final behavior. Record the hyperparameters.

Submission

Submit a zip file <PID>.zip of all the following files. Include all the PIDs if you are submitting as a group:

1. The baseline paper in PDF form with prefix “PAPER_”.
2. An ipynb file with prefix “BASELINE_”. Include the seed you used to generate the animation for the successful example.
3. The final weights so that if we were to import them (i.e. skip training) it should produce your animations. Names these with prefix “FINALWEIGHTS_<ENV>_”.
4. Part 2 Animations of the final solutions for at least 3 of the environments (including the one that you “solved”), and a PDF with your figures.

BONUS (up to 45% on this assignment):Beyond the 3 in the main assignment, for every new environment that you have clearly “solved” (even if it comes from different categories) you will receive the following cumulative bonus (10%, 15%, 20%). There should be no ambiguity as to whether something was solved or not (i.e. mountain car clearly rocks back and forth to get to goal, robot hand clearly moves block to a goal position in a straightforward path, ant/cheetah/humanoid are clearly effective in their gait). Also, redundant simulations (i.e. cartpole in mujoco vs cartpole in gym’s native env) won’t be considered different.

For each successful example provide the final animation, a separate ipynb that we can run. Include the final weights. Provide all files with the naming conventions above with a pre-prefix “BONUS_”.

A final note on honor code: *We expect you to train these algorithms yourself to get the results and the animations. Do not attempt to try to circumvent the training by taking someone else’s weight files (especially the authors!) and claiming that is your solution!*