Project 1

Welcome to Project 1 of CSE 276F. The goal of project 1 is to take you through a typical engineering-focused workflow of modifying and testing reinforcement learning in robotics environments that one might typically do in industry or for research. The final outcome of this project is to submit both code and a short report that describe your results, akin to a very short workshop paper at a conference. You may work in groups of up to 3.

This project is a bit open-ended in that you can choose the problem you are interested in and investigate it. Note that this is meant to be more for exploration and ablation studies, not for you to invent an entire new RL algorithm (simple modifications are fine).

The robotics environments used for this project will be from the ManiSkill project, a GPU accelerated robotics simulation platform developed by our lab. This is primarily because it is one of the few if not only platforms that let you tackle hard robot learning problems with RL on just colab resources. To ensure it is not too difficult, we limit you to only testing on the following environments in order of difficulty:

- PushCube-v1
- PickCube-v1
- StackCube-v1
- PegInsertionSide-v1 (this is quite difficult so do not try this one first)

We further recommend that you stick to state-based observations for input as vision based RL is quite difficult and slow to run for the harder tasks.

Report Requirements (12 pts):

- Brief Problem Statement / Abstract (2 pts): What problem are you interested in exploring with RL and robot environments, explain it briefly.
- Method, describe your research/experimentation methodology (5 pts): Why did you choose to do the experiments you ran and what is the goal of those experiments? If you made specific changes to the environment or the RL algorithm, detail this clearly and explain why. How do you motivate why you run your method to answer the problem you described earlier?
    - 5 pts if the method is well motivated and is at the standard of a conference paper in terms of correctness (novelty is not graded)
    - 3-4 pts if the method lacks proper motivation in some areas
    - 1-2 pts if the method is lacking significant motivation in many areas and barely addresses the proposed problem
    - 0 pts if the section is empty
- Results (5 pts): Run experiments with your RL algorithm(s) of choice and compare results on different robotics environments and make conclusions based on the results. Conclusions should not just be

reporting what the results say, but should also try and make an argument for why it might happen. The experiments generally should be ablation studies, where you compare a base setup (like standard PPO with fixed hyper-parameters testing on task A) with a modified setup where only 1 / very few things are changed. Generally you should be reporting task success rate or task return on the y-axis and the number of environment samples or wall-time on the x-axis. Note that it is perfectly fine to have "negative" results where the conclusion is your changes made no difference. That is still a valuable contribution (albeit conference reviewers typically do not think so).

- 5 pts if the figures/graphs are clear and easy to understand, figures have error bars when reporting success rate / return curves, and conclusions drawn are accurate
- 3-4 pts if the figures/graphs have some minor issues (e.g. labeling the x-axis as "epochs" for an RL problem) and/or some conclusions drawn do not follow logically entirely
- 1-2 pts if there are significant issues in the figures/graphs and the conclusions being drawn
- 0 pts if the section is empty.

Report should be no longer than 5 pages.

If we suspect that you did not really do an honest job and that there may be discrepancies between what you report and your code, we may run your code. We expect the code you submit to be able to reproduce every figure/table in your report. Concretely we ask you to include in the end of your report a list of python commands that can directly reproduce numbers in your report. If we find that your results are not mostly reproducible, you will receive 0 points on the results section.

Finally, it is not expected to do a very deep investigation where you test 20+ hyper-parameters. Something reasonable is 1-2 environments, 2 or more hyper-parameters/algorithmic choices, and ≥ 3 training runs per setup. The problem you investigate does not need to be super heavily investigated (you are free to investigate it further in project 2). If you are concerned about project scope / what is sufficient in terms of a report, you may ask on Piazza.

Potential Problems to Explore

We encourage you to come up with your own problems to explore, but if you are stuck and need some potential ideas, you make take one of the ones below:

- For PPO, why might it perform faster or slower in terms of wall time and sample-efficiency when you change the number of parallel environments? Is it always better to increase the number of parallel environments? What additional hyper-parameter changes might you need to make?
- What neural net architectural choices are useful for the robotics tasks? Can you find any reasonable conclusions about when a neural net setup might be better or worse?
- How can you get standard SAC to work well with learning from demonstrations? The simplest trick is to have a offline demo buffer and the online buffer and sample them 50:50 while training, but can you do better?
- Explore the hyper-parameters of SAC and/or PPO and do a simple hyper-parameter sensitivity analysis. Which hyper-parameters really tank performance, which don't really matter, are there potentially confounding variables at play?

A note on environment difficulty

Environments can be easy and also extremely difficult / borderline impossible. For starters, we recommend you to first test everything on the easy PushCube task, and if you want something a bit harder try PickCube and StackCube.

Remember in machine learning always start small, easy, before scaling to harder problems.

Code to help get started

To learn how to use ManiSkill and interact with the environments and run some simple RL, see this colab notebook:
https://colab.research.google.com/github/haosulab/ManiSkill/blob/main/examples/tutorials/1_quickstart.ipynb

The main github repo for the simulator/environments is: https://github.com/haosulab/ManiSkill

For good RL code (you will need to modify it to use it with ManiSkill) I highly recommend using https://github.com/vwxyzjn/cleanrl/

For good offline/offline-to-online RL code (also needs modifications to use with ManiSkill) I recommend using https://github.com/corl-team/CORL

- Note that CORL repo includes algorithms that do not use any RL like behavior cloning. You may not use that. The purpose of this project is to use RL.

Have fun hacking and exploring!