Final Project for CSE 291 - Robot Learning

Release: 2/27/2023; Report Due: Friday 3/10/2023, 11:59 PM

- You are **allowed** to consult any external resources, but you must cite them. You are also **allowed** to discuss with each other, but you need to acknowledge them. Members of the same team can submit the same copy; however, please mark down how each one contributed.
- This is a open-ended project. An initial report is required as a single PDF file containing a description of the project you are working on, your initial thoughts and some preliminary results. Please see the detailed instructions below. A final report is due at a later date (TBD).
- This initial report is worth 20/100 of your final grade.
- It is highly recommended that you begin working on this assignment early.

1 Environments

Similar to the Core RL Project, we will use the following 5 environments.

- LiftCube (`LiftCube-v1`)
- StackCube (`StackCube-v1`)
- PegInsertion (`PegInsertionSide-v1`)
- TurnFaucet (`TurnFaucet-v1` and `TurnFaucet-v2`)
- PushChair (`PushChair-v1` and `PushChair-v2`)

2 Overall Project Descriptions

This is an open-ended project where you can show your creativity in solving object manipulation tasks. You will be working on one of 6 projects listed below, covering RL/IL agent learning with state or visual observation as inputs.
We expect you to show decent effort in working on **at least** 4 tasks. In the initial report (*around two pages length*), you are asked to write a description of the project you are working on, your initial thoughts, challenges encountered, your plan to tackle them, some preliminary results, etc. In the final report (to be announced later), you will be asked to write a more detailed report including further results, what you have learned, code and videos.

This is NOT a competition. We will grade your report not only by checking the performance, but more importantly, by the description of your approach, how you tackle the challenges, what you have learned and found, etc. Below are some instructions you might find helpful for this project.

- Similar to the Core RL Project, please install the `cse291-projects` branch of `ManiSkill2`.
- For PushChair, we recommend you to start with

```
control_mode=base_pd_joint_vel_arm_pd_joint_vel
```

For all the other tasks, we recommend `pd_ee_delta_pos`. You are welcome to explore other control modes as they might be harder or easier for RL/IL training.

- Starter code for evaluating RL/IL agents is provided in `ManiSkill2/examples/eval_starter.py` at the `cse291-projects` branch.
- You need to show improvement of your approach over a baseline in terms of performance (either evaluation with the same set of seeds in training or with held-out set of seeds or some random seeds), or sample efficiency.
- As this project is open-ended, in case of no improvement found, you need to show your analysis or findings regarding why it does not work and discuss the potential solutions.

3 Proj 1: Learning vision-based generalizble policies with state estimation

The policy trained in the core RL project cannot be directly applied in reality, because robots cannot directly observe the state of the environment. Instead, they need to estimate the state from visual input. You are asked to train a policy that takes RGB(D) or point cloud input. While one can jointly train a visual RL from reward signals, this can be quite hard (please try it as a baseline) because you have to learn a good visual representation only through reward signals, and it is unclear how well the learned visual representation can be generalized to new objects.

Another option is to decouple visual representation learning and policy learning. For example, you can first train a “state” estimation network to estimate the state from RGB(D) or point cloud input. Then you can train a policy that takes the estimated state as input through RL. To achieve strong generalizability, you may need to design the state carefully instead of blindly using the state provided by the environment (for example, the state provided by the environment may not be sufficient to distinguish different environments (e.g., different faucets), although they can distinguish states within an environment (e.g., the different operation state of the faucet)). Therefore, there leaves a large space to show your creativity.

4 Proj 2: Learning vision-based policies with state-based teacher policies

In this project, we will explore another idea to learn a vision-based generalizable policy. We will first train a state-based policy, and then use the state-based policy to “supervise" the learning of a vision-based policy. In RL, there are many attempts to include human supervision in the loop, e.g., to ask humans to rate the quality of the rollout from the learned policy and use human feedback as the reward. An example work of this kind is “DAGGER” (`https://arxiv.org/abs/1011.0686`). In this project, we will explore a different way, i.e., to use the state-based policy to supervise the learning of a vision-based policy. An example work can be found in `https://arxiv.org/abs/2111.03043`. Similar to project 1, to achieve strong generalizability, you may need to design the state carefully instead of blindly using the state provided by the environment. Also, please try the direct vision-based RL approach as a baseline.

Note: the initial idea of this project was proposed by Zhiao Huang and Tongzhou Mu.

5 Proj 3: Curiosity-driven exploration for state-based policies

In this project, we will explore the idea of *curiosity*-driven exploration strategies for state-based RL policies. The key idea behind curiosity is to design a mechanism that rewards an agent for continuously exploring unseen parts of the state space; this is in contrast to traditional mechanisms for exploration such as stochastic policies, action noise, or maximum entropy RL. Examples of curiosity-driven exploration include *"intrinsic"* reward bonuses proportional to *e.g.* prediction errors in a forward dynamics model (predicting next state from current state and action; `https://arxiv.org/abs/1705.05363`), ensemble disagreement (variability in predictions between multiple models; `https://arxiv.org/abs/1906.04161`), or random network distillation (`https://arxiv.org/abs/1810.12894`). Please explore two or more of these approaches and compare them with a baseline that does not use such exploration strategies.

Note: the initial idea of this project was proposed by Nicklas Hansen.

6 Proj 4: Curriculum learning for state-based policies

Curriculum learning is a training strategy that learns first from easier data and then from harder data, similar to how humans learn (see this survey `https://arxiv.org/abs/2010.13166` as a reference). This strategy has been shown to improve both training and generalization performance in various tasks in the fields of CV and NLP. Curriculum learning is also natural for RL training with great diversity in the task configuration space. For instance, to insert a peg into the hole, it would be easier if, upon environment initiation, the peg is already near the hole or with its angle aligned.

To deterministically find such easier configurations, you can record and/or set random seeds for environments in ManiSkill2. A random seed is similar to a level in the game, where some levels are much harder than others. A similar idea was adopted in RL as Prioritized Level Replay (`https://arxiv.org/abs/2010.03934`). You can also try leveraging the demonstration dataset provided by ManiSkill2 with some creative designs (either learned or heuristic) to automatically generate a curriculum of these levels. Please explore curriculum learning for tasks in ManiSkill2 and compare your approach with an RL baseline without curriculum learning. Bonus points: RL with sparse reward signals is extremely hard. Can your approach help to ease the learning?

Note: the initial idea of this project was proposed by Stone Tao and Zhiwei Jia.

7 Proj 5: Distributed policy optimization for state-based policies

As you might notice, policy optimization itself for tasks in ManiSkill2 is challenging. Besides curiosity-driven exploration and curriculum learning, another approach that can effectively ease policy learning is to learn a policy in a distributed manner. Generalist-specialist learning (GSL) is a recently proposed meta-algorithm for large-scale RL, where the task configuration space is partitioned into different subsets where a population of specialist RL agents are launched to ease both exploration and optimization. Please see `https://zjia.eng.ucsd.edu/gsl` for details. By combining a single generalist agent that can generalize well and a bunch of specialists that learn faster, GSL was shown to be particularly successful on tasks with great diversity in the task configuration space. For instance, it partition the task configuration space of PushChair over different chair models and improves the SAC baseline. Please explore GSL with tasks in ManiSkill2 by coming up with creative strategies to partition the task configuration space and compare with the RL baseline without GSL.

Note: the initial idea of this project was proposed by Zhiwei Jia.

8 Proj 6: Learning vision-based policies via sequence modeling

A more sample efficient way of policy learning is to leverage demonstrations via imitation learning (IL). CoTPC, a recently proposed IL method, casts long-horizon decision-making as sequence modeling via the chain-of-thought technique that is shown to be powerful in language modeling (`https://arxiv.org/abs/2201.11903`). It combines the applicability & scalability of Behavior Cloning (BC) with the planning capabilities & generalizability of Model Predictive Control (MPC), and at the same time, overcome the challenges of BC (which does not work well due to sub-optimal demos) and enable planning-based control over a much longer horizon than MPC. Specifically, it utilizes hierarchical structures in object manipulation tasks via key states that mark the boundary between sub-stages of a trajectory. CoTPC can solve challenging tasks such as peg insertion very effectively and efficiently (in terms of sample efficiency). An open question is to extend this work with vision-based inputs (RGB, RGBD or point clouds), as currently it only models trajectories with state observations. Please refer to the paper here and the code at `https://github.com/SeanJia/CoTPC`.

Note: the initial idea of this project was proposed by Zhiwei Jia.