Description.

In the final project, you will conduct research on a topic related to reinforcement learning based on your research interest. This is a one-person project. However, depending on the sophistication of the technical content, the project maybe conducted by up to two persons subject to the instructor's approval. The possible formats for the project include:

1. Implementation: choose an environment you would like to solve using reinforcement learning algorithms. Implement this environment or wrap an existing software into a suitable reinforcement learning environment. For example, you may use reinforcement learning algorithm to solve your favorite game. Implement at least two deep reinforcement learning algoritms to solve it. Present the performance comparisons with benchmarks (if any) and the sensitivity to hyperparameters.

2. Algorithm/reinforcement learning research: identify a technical problem/challenge with existing deep reinforcement learning algorithms or theories, develop a new algorithm/theory to address the problem. For example, this could be sample efficiency, balancing exploration-exploitation, stability of deep reinforcement learning, etc. You need to conduct some literature review to understand the prior efforts toward overcoming that problem and the state-of-the-art (SOTA). Present the details of your research and the performance comparison between your algorithm and at least one SOTA.

3. Application research: apply reinforcement learning to solve a problem in your research field. You will need to state the problem in the language of Markov decision process (MDP), identify the key properties of this MDP such as the type of the action space, state space dimensionality, etc. and based on this identification, apply appropriate reinforcement learning algorithm to solve it. Please implement at least one algorithm and follow the experimental result reporting procedure in the field.

Your proposed topic may not fall exactly in one of the above categories. However, you must demonstrate sufficient technical originalities in your project report detailed in the Deliverables section below. We encourage you to challenge your coding ability and math maturity when deciding the topic for this project. If you can, do not feel like you are doing this project because it is the course requirement. Instead, please treat it as if you want to add this item to your public Github repositories, or confidently present it in a technical interview by your future employer. And most importantly, as why we have signed up for the course, enjoy and have fun.

In case you cannot come up with a topic suitable for the project, we have provided with you a default one: solving example Unity ML-Agent environments using reinforcement learning. You can find the description and requirements at the end of this document. Note that if you choose to use the default project, you will need to submit the same set of deliverables as well. The default project may not be conducted by two persons.

Deliverables.

There are three required deliverables for the project: project proposal, final report, and source codes along with proper documentations.

1. Project proposal: this should be a one-page document that clearly articulates the research question to address, proposed solution techniques, and metrics to measure the completion. If the project will be conducted by two persons, please also identify the labor divisions among the two. The project proposal will be due on May 13, 2024.

2. Final report: this should be a double-column, up to six-page pdf document that shows all project findings. The report should resemble the format of a typical conference publication of IEEE. For example, you can use the IEEE manuscript template for conference proceedings. The final report needs to include the following five sections. (1) an introduction section to state the background and motivation of the project, and literature review if any; (2) a problem description section to present the problem in appropriate format for reinforcement learning; (3) a technical method section to detail the steps of the reinforcement learning techniques; (4) a numerical results and performance comparisons section; (5) conclusion. The final report is due in the last lecture of the quarter.

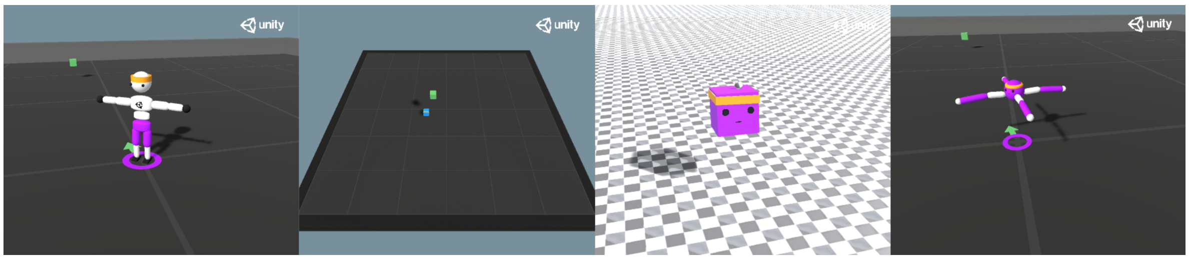
3. Source codes and documentations: please submit all source files you've written with appropriate comments that explains the code. Also, please use some txt file to describe the role of each file. If your program

requires extra setups, please describe those setup as well. Do not submit code/dataset that may not be released. For example, proprietary softwares, copyrighted materials, or your own research materials.

Resources.

1. Spinning Up in Deep RL: an educational resource produced by OpenAI.
2. Paper With Code: open resource with machine learning papers, code and evaluation tables.
3. OpenAI Gym: a toolkit for developing and comparing reinforcement learning algorithms.
4. Gym Retro: a platform for reinforcement learning research on games.

Default Project I: Reinforcement Learning for Unity ML-Agent



The Unity Machine Learning Agents Toolkit (ML-Agents) is an open-source project that enables games and simulations to serve as environments for training intelligent agents. In this project, you will develop the simulation platform and reinforcement learning code to solve the example Unity ML-Agent environments.

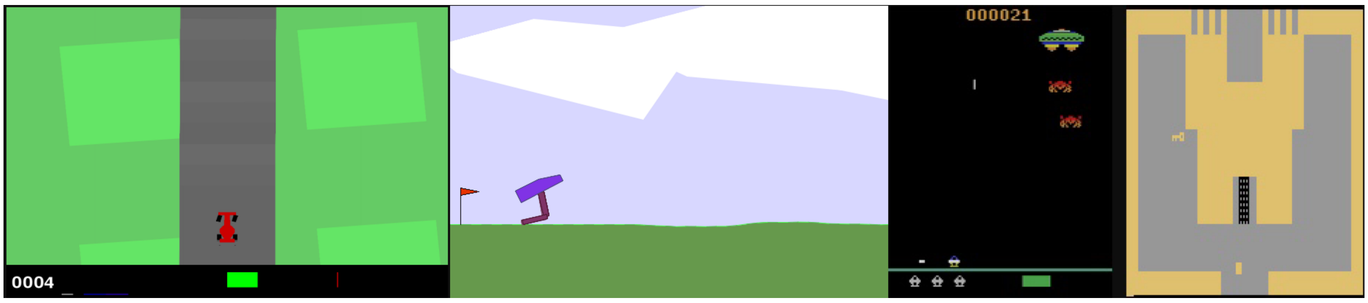
To setup the Unity ML-Agent for your system, the following links are helpful:

1. Download the Unity ML-Agent examples.
2. Build the Unity ML-Agent scene.
3. Wrap the scene as a gym-like environment using gym-unity

The second step will create the Unity environments as executable; then the gym-unity in the third step will wrap that Unity environment executable as a gym-like environment so our Python code can interact with it. However, the gym-unity wrapper currently only support single agent environment. Therefore it is not easy to run multiple environments at parallel to improve the training efficiency. Nevertheless, in some example environment, running a single agent is good enough to converge to a useful policy.

After you successfully setup the environment, please develop code to solve them by creating agent-environment interaction and agent training. Please implement the appropriate deep reinforcement learning algorithms and tune the hyperparameters. Some of the environments may be challenging to solve due to very complex high-dimensional action space, sparse reward, etc. However, please try your best to finish as many environments as possible. In your final report, please include the list of environments you've attempted, and indicate which environments have been solved satisfactorily and which have not.

Default Project II: Reinforcement Learning for Gym based on homework assignments



For the homework assignments, we successfully utilized the DQN, DDPG, and SAC algorithms to tackle two gym environments. As part of this project, we will extend our implementation to solve additional gym environments. To adapt our existing homework project, certain modifications need to be made. For instance, if we encounter pixel observations, we should incorporate convolutional layers into our project. Furthermore, we can expedite training by implementing early episode termination based on specific criteria. Additionally, to enhance performance, we have the flexibility to modify the reward function. These changes will ensure the necessary adjustments are made to accommodate new environments and optimize our project accordingly.

We offer you three options that you can choose for this default project:

(Option 1).Use one deep RL algorithm (DQN,DDPG,SAC,...) to solve one gym environment. Report the sufficient modifications you made to the homework project in order to improve performance.

(Option 2).Use one deep RL algorithm (DQN,DDPG,SAC,...) to solve as many gym environments as possible. Report the performance the algorithm has on different environments.

(Option 3).Use as many deep RL algorithms (DQN,DDPG,SAC,...) as possible to solve one gym environment. Report the performance of different algorithms on that environment.

Some environments you can try to solve using DQN:

- Box2d:
  Car Racing (discrete)
- Atari (most Atari games have piexls observation and discrete action space):
  Adventure, Demon Attack, MsPacman,...
- Third-party environments:
  flappy-bird,...

Some environments you can try to solve using DDPG,SAC, and other algorithms:

- Box2d:
  Car Racing (continuous), Bipedal Walker
- MoJoCo: Ant, Humanoid, Walker2D,...