CS 593 RL1 Fall 2025
Assignment 2: Deep Q Networks

Purdue University

Due October 21 2025, 11:59 PM

Installation

Setting up a Conda Environment

1. Open a terminal/command prompt and create a new environment:

```
conda create -n cs593rl python=3.9 -y
```

2. Activate the environment:

```
conda activate cs593rl
```

3. Navigate to the directory where you unzipped the code and install the necessary Python packages from `requirements.txt`:

```
pip install -r requirements.txt
```

This assigmnent uses two Gymnasium environments: LunarLander-v2 and CarRacing-v2. Refer to the documentation pages linked above for information on the state space, action space, and reward function (though in this assignment rewards are only reported as part of the evaluation).

Part 1: Implement DQN and Replay Buffer Variations [4 pts]

We have provided a starter implementation of DQN, with some functionality unimplemented. You have the option of running the code either on your own machine, on a cloud based service (such as Google Colab), or on RCAC Scholar.
Use the concepts introduced in class along with the Gymnasium/PyTorch documentation to fill in the blanks in the code marked with `TODO`.

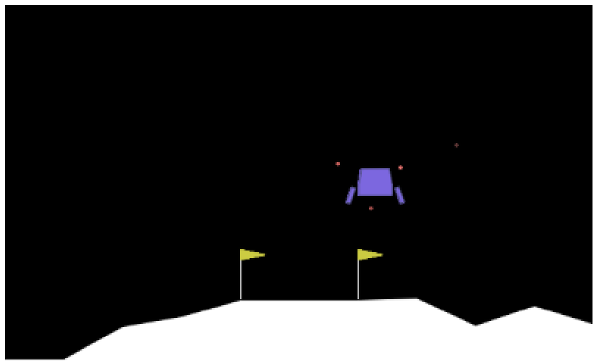

Figure 1: LunarLander-v2

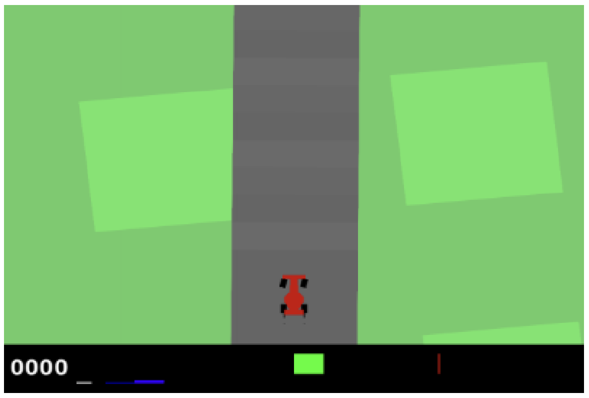

Figure 2: CarRacing-v2

1. DQN and Replay Memory [2 pts]

Implement DQN and Standard Replay Memory and run DQN on the following command.

For DQN implement:

1. The target implementation in "optimize_models", and
2. Select action for deterministic action and random action in "select_action".

For standard replay buffer:

1. Complete the class by providing push and sample functions in "ReplayMemory" class.

Here is how you can run each task:

```
python main.py --env LunarLander-v2    (default is set to dqn and
python main.py --env CarRacing-v2       standard replay buffer)
```

Check main.py file for all the args that can be used.

Add a standard DQN run with replay buffer loss and reward for each environment to your PDF report.
LunarLander has an observation space that we can use a standard MLP. However, CarRacing observation is an image. In order to process the image, we applied a wrapper to transform the image, add Frame Stacking, and added CNN networks. In order to train on the image for Car Racing, the CNN networks on the cnn_networks.py file is used.

Tip: You can plot performance metrics along with renders of policy rollouts by viewing the results in Tensorboard by running the following command and then navigating to `http://localhost:6006/` in a browser.
Report the training accuracy and evaluation reward plots for both tasks in the report.

```
    tensorboard --logdir runs
```

2. Double DQN, Dueling DQN, and Prioritized Replay Memory [2 pts]

Implement Double DQN, Dueling DQN and Prioritized Replay Memory (also used as PER).

To implement Double DQN:

1. Complete the Double DQN's target calculation (similar to the standard DQN) in "optimize_models"

For Dueling DQN"

1. Complete the networks.py and cnn_networks.py files' "forward" methods value heads.

For Prioritized Replay Memory:

1. Complete the "sample" and "update_priorities" functions.

More information is written with the "TODO" lists inside the code.

Then you can train your agents via

```
python main.py --env LunarLander-v2     (you can add: --use_double_dqn
python main.py --env CarRacing-v2        --use_dueling_dqn --use_per)
```

Add one of each of your choice of either double DQN or dueling DQN with the PER into your PDF report with the loss and rewards.

Part 2. Agent and Hyperparameter analysis [6 pts]

Experiment with the given hyperparameters and observe how the values affect the performance of the DQN agent: the variation of DQN agent (DQN, Double DQN, Dueling DQN), PER along with alpha and beta values.

The hyperparameters can be: replay size, epsilon (start, end, and decay), gamma, target_update, tau. You can also tune learning rate, batch size, number of training epochs, as well.

Here are some examples on how to run different agents and hyperparameters. You can change these values by using the following commands:

```
python main.py --use_double_dqn --use_per --env LunarLander-v2 --lr=1e-5
python main.py --use_dueling_dqn --use_per --env LunarLander-v2 --gamma=0.9
python main.py --use_double_dqn --env CarRacing-v2 --epsilon_decay=20000
```

Important Tip: For Car Racing, we suggest 100 number of epochs, and for Lunar we suggest 500 number of epochs. Note that Car Racing training might take longer due to image space.

2A. Agents [3 pts]

First, having the hyperparameters fixed, we would like to see different variations of agents/replay memory. Currently there are 6 different combinations.

Out of these 6 combinations, we would like to see 2 additional combinations of your choice on each game that you didn't run on the first part. (example of combinations can be: DQN normal RB + Dueling DQN normal RB for Lunar and, DQN PER + DDQN normal RB for Car Racing, along with the runs from your first part)

This should result in 4 new experiment runs and with 2 of the runs from part one. Make sure to add them together and show the plots for each environment in your report under part 2A.

2B. Hyperparameters [3 pts]

For one of the environment and one of the agent of your choice, experiment with at least **two** hyperparameters within 3 different runs, where each 3 runs will be with a fixed agent but variable hyperparameters (e.g. using Double DQN and PER on Car racing, do at least 3 different epsilon setup and report finding, then having Dueling DQN and PER on Lunar, do 3 different learning rates). Report your findings by showing training accuracy and evaluation reward plots.

For example, if you choose to vary learning rate, you may choose values such as 1e-3, 1e-4, 1e-5 and both your accuracy/reward plots should have 3 runs corresponding to these values. You can also see the full list of hyperparameters in the main.py file.

We would like to see an analysis that is applied to different models and tuned to train stably and well.

Make sure to identify which hyperparameters you have changed, and which runs correspond to which values. You can either indicate this in the report or change the name of the log file in the `runs` directory, which then updates the label name in Tensorboard.

For this part you will show the loss/reward plots, for each of the hyperparameter analyses. Please briefly report your findings and analyses for 2A and 2B in the PDF.

Tip: You can toggle certain runs as visible or invisible by clicking the checkbox next to run name on the left of the Tensorboard window.

Submission Instructions

For simplicity, you may download/screenshot the Tensorboard figures and include them directly in the PDF report.
In order to turn in your code and experiment logs, create a zip folder that contains the following:

- Your PDF report. (Include your name into the report and the report file)
- Your completed code files.
- Only one DQN run for each environment on a folder named "selected_runs" (to keep the submission file sizes small.)

Upload the resulting zip file with submission file named as

`firstname_lastname_hw2.zip`

to Brightspace.