Mini HW 7 Report

by Mason Francis

Implementation Process

My implementation process was pretty simple. I based my code off the repo given, and made the following changes:

- Used `stable-retro` instead of `gym-retro` because of out-of-date software
- Used `gymnasium` instead of `gym` because of out-of-date software
- Used `torch` 2.2.2 because I wanted to use CUDA, but I think the original `torch` version specified should do fine for CPU
- Got rid of areas where the `seed` parameter is used. Current versions of these packages I am using don't use it

I had to make the changes above to make the code run under modern versions of Python (3.10) and with some modern packages. The code has to be run under specific environment conditions or else it won't work. The starter code trained using PPO, so I was able to switch that out for A2C without too much effort.

Giving credit where credit is due, I definitely received help through my classmates on the Discord chat. They were very helpful in figuring out how to get things running.

Instructions To Run Code

To get training to run:

- set up a venv with python 3.10
  - I used Ubuntu 22.04. Use venv. Don't use conda, I ran into some driver issues while trying to use it
- run `pip install "stable-baselines3[extra]"`
- run `pip install -r requirements.txt`
- run `pip install torch==2.2.1`
  - I only did this because I wanted to use CUDA, and this is the `torch` version that matched with my CUDA version
- obtain compatible game rom
  - I got some roms here and specifically downloaded the NES ones. I used one that matched the one used by the starter code
- import roms by running `python3 -m retro.import [path to folder with zip(s) of rom(s)]`
- run `python3 TrainPPO.py` or `python3 TrainA2C.py` depending on which one you want

I chose 452,000 timesteps because that's what my PPO got to before I got tired of waiting for it because I had other pressing homework to do. Just being honest.

Algorithm Comparisons

I was not able to get `Run.py` from the starter code to work because of changes in the packages I'm using, so I couldn't really compare the outcomes of the algorithms. Because of that I am unable to provide any graphs and charts to compare them.

I did, however, notice that A2C trained decently faster than PPO, taking probably 2/3 the time to train.

Since I'm not able to compare outcomes, I did some learning here and using phind.com about the differences and similarities between A2C and PPO.

A2C tends to be simpler, but it requires more data. PPO is more efficient, yet more complex than A2C. I would say that my understanding of how A2C works is more clear than for PPO, particularly because PPO is more complex. A2C is more prone to larger differences in how it reacts to what it "learns", where PPO doesn't allow outcomes with such large differences to sway it's decisions as easily. Both of these also have some similarities in that, being reinforcement learning algorithms, they are trying to learn from outcomes of actions and adjust their reaction accordingly. They just do it in different styles.

Challenges Faced

I faced a few challenges with the starter code, primarily it’s age. Although it’s not that old, some of the packages have had breaking changes since it was written. Luckily, most affected packages have had replacements written for them, so I was able to use those. These packages were:

- gym-retro -> stable-retro
- gym -> gymnasium
- stable-baselines3 -> newer version compatible with gymnasium

Regrettably, I was unable to get the `Run.py` from the starter code to work because of changes in the packages I'm using, so unfortunately I was not able to view the outcomes of the training and am unable to provide graphs and charts about them.

Key Takeaways

I had a few different key takeaways from this assignment.

- I learned that different types of training take different amounts of time.
- I learned that A2C and PPO involve different levels of complexity and learn in different styles.
- I also learned firsthand why GPUs are so useful for this type of work.
    - As soon as I installed the CUDA toolkit and drivers for use with PyTorch, the speed of training increased dramatically.
    - In speaking with my other classmates, this type of performance improvement was experienced by them as well.