Reinforcement Learning for Dark Souls

(and maybe Elden Ring)

Overall To Do List

- [x] ~~Install Soulsgym/Gymnasium + create Github~~
- [x] ~~Create venv with all required dependencies~~
- [x] ~~Create requirements.txt~~
- [x] ~~Remove the env from github~~
- [x] ~~Run random agent~~
- [x] ~~CREATE BRANCH FOR DQN~~
- [x] ~~Add speedhack (x2 or x3)~~
- [x] ~~Train DQN agent (it failed)~~
- [x] ~~Create DDQN branch~~
    - [x] ~~Implement DDQN~~
    - [x] ~~Add logging~~
    - [x] ~~Add performance graphs~~
    - [x] ~~Add time logging (cuz kinda important)~~
    - [x] ~~Compare speed of cpu vs gpu (BONUS)~~
- [x] ~~Create DDDQN agent~~
    - [x] ~~Tensorboard logging~~
    - [x] ~~Easy parameter configuration~~
    - [x] ~~Reward for getting close to boss~~
    - [x] ~~Find out how to modify boss phase + speedhack easily~~
    - [x] ~~Test different reward function~~
- [x] ~~Reward function improvements:~~
    - [x] ~~High reward for hitting~~
    - [x] ~~Small penalty for rolling~~
    - [x] ~~Smaller penalty for time spent~~
- [x] ~~Update the readme because it is not up to date anymore~~

Notes: if we make a new instance of the soulsgym library, like with a new venv, some params should be changed, like the speedhack value in the core/speedhack module and also the init_retries in the IudexEnv class in envs/darksouls3/iudex.py

Helpful link

BUG FIXES

- [x] ~~Before starting steps, agent needs to lock onto the boss so we need to enforce that (probably a forced input before the loop)~~
- [x] ~~Numpy startup issue, not usable bc of version incompatibility?~~

Diary

19/09/2024-
Managed to install everything and created the venv
Managed to run a random agent from this script
BUG FIXES:
fixed the win32gui front window thingy (changed in the library itself)

23/09/2024-
Couldn't fix the forced ‘q’ input for the environment to reset
Idk why it does that but just doing virtual key presses doesn't work
I think i need to check how the key presses are done within the library to reproduce it
Overall no progress 😭

24/09/2024-
Not 100% sure why it works now but it does lol pushed everything else kept the code i used to debug, might be useful in the future
Tried: sending manual q inputs through the core.game_input functions and multithreading to spam ‘q’ while the env is resetting\
Now the env works without that but ig it must've been solved something behind the curtains (hopefully)

What the problem might be:
I press q, the target gets locked on. When the agent presses q after the env is reset, it locks on. When you add the lock_on input to try to lock on before the env resets, q is indeed pressed but the agent does not lock onto the boss. Also when i do this: next_obs, reward, terminated, truncated, info = env.step("lock_on")
The agent does not lock on he kinda just moves left
Ended up not working…

25/09/2024 - 17/10/2024
Tried a bunch of stuff to debug
Opened an issue on the github of the soulsgym library
Talked with the dev to find a solution
Couldn't detect what it was for a while
Not a code issue??

18/10/2024
WORKS NOW
Wasn't a code problem ->
Language setting of the pc was english and the keyboard was set to qwerty (normal). Inputs were being sent as qwerty but the game was interpreting them as azerty (still do not know if this was a hardware or library configuration issue)
Fix: remove any secondary language on the pc (remove the option of having something else than a qwerty input)
!! Author updated documentation to avoid this problem in the future here
Library works without any issues now

21/10/2024
Added requirements.txt file
Removed venv from github commits with .gitignore
Updated the readme for install instructions
Create DQN dev branch to work outside of main
Fixed DQN start up issue
New bug => numpy issue, can't initialize?? Need to fix asap otherwise it's horrible for data collection/manipulation

22/10/2024
Improved code for DQN
Added logging of data into csv file
Started working on the speedhack to improve training otherwise it will take forever to train an agent

28/10/2024
Started doing some trial runs to see if everything works
Some debugging too

2/11/2024
Added new checkpoints to the saving system
Now we save both the model and the optimizer (to resume training later)

4/11/2024
First results of DQN run. It doesn't learn very well: after a couple hundreds of episodes we reach a plateau
Sometimes we get a spike later where the loss increases and decreases again (don't know what causes that)
Played around with parameters more but no significant improvement

6/11/2024
Decided to do more research, notably on DDQN which could help having a more stable learning

Read part of the book “Hands-On Machine Learning with Scikit-Learn & Tensorflow” by Aurelien Geron
I also read other papers, like “Reward is enough” by David Silver, Satinder Singh, Doina Precup and Richard S. Sutton as well as “Proximal Policy Optimization Algorithms” John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov. I read this paper since I decided to try and implement PPO to see if it does any better than DQN (and DDQN if I implement it). I also read parts of “Artificial Intelligence a Modern Approach” and “Reinforcement Learning An Introduction” (not entirely of course)

10/11/2024
We are currently at about 70 hours of trying to train an agent at the moment but the agent has not learned to beat the first phase of the boss yet. I think DQN might be the problem, so I will try implementing a more robust algo, DDQN.

18/11/2024
Watched some videos, read some papers online, started playing around with DDQN (didn’t put it on the github) so I can start implementing it.

26/11/2024
After trying to run code with the CUDA version of torch, there are still problems with numpy (at least i understand what now)
Numpy uses a specific version of the C api (0x10) but the latest torch cuda version tries to invoke a different version (0xF). And I can't change that version myself. This means I would need a different torch version but I CANNOT for the love of god find any documentation online about that. So i have two options: keep trying older versions until one works (hell) or use the basic cpu torch version (not optimal).
My theory: using the base torch version might not be the worst for two reasons:

1) My cpu is not so bad so I can probably calculate relatively fast for a cpu
2) It might not be that slow because there is a cap in how many calculations I can do per second anyways since the game can only run so quickly (not 100% sure about that). This means that whether i use cpu or gpu, it might not make such a big difference. I will test how quickly I can run episodes and calculations using the cuda torch on the DQN code (which does not directly use numpy) and the DDQN code which does use numpy. From then I can estimate how big of an issue this cpu/gpu problem is.

THINGS TO DO:
~~-Speed up game (again) for DDQN~~
-Add logging (save both model and optimizer) and maybe do it in a better way than with DQN)
~~-Automatically draw the loss and rewards graph during training (now that i have access to numpy it should be doable unlike with the DQN code~~)

30/11/2024
Fixed some code, i think it should all run now
Once i fix all of it i'll push on github and start some training to see what params to optimize

02/12/2024
Changed the architecture (pushed some stuff on github)
I will start training now because the runs I did before were just to make sure everything works fine.
Concerning the “drawing loss curve during runtime”: can’t do it because otherwise it opens a different window using matplotlib and the DS3 windows isn't the main one anymore so the code stops running (so i will deal with stats and plotting after runs)
Made other changes too (see commits)

TOOK A MONTH BREAK TO TRAVEL AND SEE FAMILY AND STUFF

06/01/2025
Things i want to do:
I think the eval breaks towards the end so,
-don't save atm it is not useful
-make eval happen quickly to test it
-observe how and when eval breaks

07/01/2025
Param change for debugging:
Min_episodes = 20 -> 1
measure_step =100 -> 10
measure_repeats=100 keep this one to really emulate what we did before
Added printat line 200 for debug
os.clear at line 447 is every 500 episodes now
Set load_model to False

Couldn't reproduce the code breaking -> agent stuck against wall keep moving in one direction, either ‘a’ spam or roll spam idk?

15/01/2025
Couldn't fix the previous issue, which actually comes from the evaluate function. Agent spam rolls to the left and idk why.
Will do more runs and keep them if they show any improvement.

20/01/2025
Deleted runs cuz they are not the best. Did some research and to use a simple (and maybe buggy) implementation of DDQN it would take a LOT of train time. Will start from scratch a new method to make sure the code is more robust

24/01/2025
Create a Double Dueling DQN branch. Found an implementation that looks pretty good so I will try to replicate it and apply it to my case.

FIXED the numpy issue that i had a long time ago (see this guy)
Time to learn from my mistakes and do the following:

- [ ] Preprocess the observation space better
- [ ] Maybe do the same for action space (dunno if that's needed atm)
- [ ] Immediately setup a better logging and checkpoint system
- [ ] Make sure there are timestamps EVERYWHERE

27/01/2025
Fixing code for it to work with ludex.
Using the same environment for both training and evaluation isn't inherently bad if:

- Evaluation is infrequent (e.g., after every N episodes) to minimize interruptions to training.
- You properly reset the environment for each evaluation run.
- You ensure deterministic policies during evaluation.

However, this approach could lead to slightly noisier evaluation results, especially if the environment's state isn't perfectly reset. This is less of an issue if you're focused on trends rather than absolute precision in evaluation.

Pushed some code, and I think training should go smoothly.
Training and eval happens based on steps (and not episode!). I will see if that makes a big difference between this and episode based training.

28/01/2025
Did my first proper run with DDDQN with a custom reward. The agent is able to learn pretty quickly. 300k steps in about 3 hours is not bad.
Agent spams attacks too much, need to lower the reward for that otherwise it just tries to trade blows which isn't good.
Will display time when eval is being displayed

29/01/2025
Things to test for reward function:
-distance between player and boss -> closer = reward (MAIN BRANCH)
-dying one shot during training? Don't get hit mentality? (NO DEATH BRANCH)
-hit reward exponential-> first few hits give smaller rewards than later hits (no matter the damage dealt). Learn to chase more hits over time? (EXP HIT BRANCH)
-No shield? Git gud (NO SHIELD BRANCH)
-Change to e-greedy epsilon strat maybe instead of basic decay

30/01/2025
Failed run (forgot to change speedhack value lol)
Optimized some stuff, will try a new run tonight, by putting more value to hitting rather than not getting hit (and small roll penalty)

31/01/2025
Reward went up on the new run but the loss looked horrible (it increased???)
Changes made:
-gradient clipping
-reward scaling (divided by 100 to avoid to high or too low rewards)
-went from TAU target network update (so slow over time to hard update every n steps)
With hard updates + gradient clipping + reward scaling:
-Loss should stabilize
-Episode scores should show gradual improvement
-Q-values become more reasonable (mean between -10 to +10)

IMPORTANT
Need to do a test where i print q values and rewards at every step at x1 speed to observe
Donzo ig it's ok?

FEBRUARY:
Didnt do much cuz i was working on other projects, but i still did some testing, ran a few models but no breakthroughs. Kinda put it aside to come back to it with a fresh open mind

02/03/2025
Back to the grind
Changed the reward function, decided to make something much simpler because the loss curves i was getting were terrible (not decreasing)
Refactored it, now loss seems to be decreasing properly but we get stuck in a local maximum, the loss converges but doesn't reach an optimal policy (agent isn't good enough and training it for another 10 hours wouldn't fix that problem)

05/03/2025
Okay so now we gotta change the learning parameters: (ideally in this order)

- [x] ~~Epsilon :~~
    - [x] ~~Eps decay~~
    - [x] ~~Eps min~~
- [x] ~~Learning Rate~~
- [ ] Random Steps
- [ ] Discount factor
- [ ] Batch size
- [ ] Network width

06/03/2025
Ran a test with a slower epsilon decay and higher eps min to keep exploration but the loss converged quickly.
Today we will try increasing the learning rate to 1e-4 to try and escape a local minimum.

07/03/2025
Increased LR by a bit, maybe do more of that? Not now tho
Current changes for run 1.4: increased train freq from 40 steps to 100 and random steps to 50k
Keeping replay buffer at 256, but will probably increase it for the next run

17/03/2025
Changed the action space. I removed some actions (4) like the forward right roll and we just kept the right roll. You lose some diagonality but I doubt that it has a big impact on the agent's policy.
First might as well learn with a restricted move pool and then increases it as it works (hopefully that's the correct approach)

18/03/2025
Played around with some hyperparameters (see 2.0 run), got better reward but not by much

20/03/2025
Made a new enhanced duel DQN class with a bigger and more complex architecture. Tested it and it runs and saves so I will do a run tonight to see if it works better (run for at least 1 million steps to compare with the previous one)
Also im printing the eps value now to see how fast/slow it decreases
Hopefully this works bc i think it's the most promising thing

21/03/2025
Agent did not perform any better. Looking at the loss curve I think it overshot the learning at the beginning because it decreases very fast then increases a lot and starts decreasing again. So I reduced the learning rate again, increased the batch size and the epsilon decay.
Maybe this will do better otherwise I will probably make a better experience replay to favor important steps to learn better (thanks gpt)

I FORGOT TO UPDATE THIS
Back to it (i worked in the meantime but forgot to update the journal)
Anyways i started working with TorchRL now, works pretty well but it was a bit of pain to set up
Code is pretty easy to follow so that's cool: see here

20/05/2025:
Try and add more hyperparameters to the various elements of the pipeline (to the MLP, Data collector, DQN Loss and EGreedy Module)

Also find out if the obs is broken because I'm getting like 26 obs but it seems like amacati has like 70+ so maybe my encoding is pretty bad.

21/05/2025:
changed the obs to have more info (went from 26 to 74 observations by changing the way I preprocessed it). Using better one hot encoding will really change it i think.

22/05/2025
Did a longer run, agent seems to be learning much better than previously
Fined tuned some hyperparameters, notable some regularization on the dqn loss

23/05/2025
AAAAY new +-20h run produces an agent that can almost perfectly roll
Reward still isnt super high but its because he didnt try to attack
=> the reward doesn't favor hitting or not getting hit so my guess is that since hitting is waaay riskier than not hitting the agent kinda because super proficient at dodging without trying to learn how to hit.
But my guess is that if we favor hitting a bit more damage:
Damage reward x 1.5, there is more of an incentive to learn how to hit the boss
I will try that next

25/05/2025
Added better logging to track the progress instead of only relying on the pbar
Tensorboard now saves the loss and reward over time
Next step is to do another 20-25 hour run, but the agent could need more time honestly
Even if the agent needs twice the time that would be pretty good
The original author of the DS3 gym wrapper said it took him days if not weeks to train an agent that only gets 50% success
Then again i only trained on phase 1 so if phase 2 takes as much time then i might get the same results
Hopefully my win ratio is better…

26/05/2025
Maybe run #8 with a better reward scaling to the damage reward (honestly could even lower that a bit because the agent is still playing a bit too aggressively)
We will check results tomorrow

27/05/2025
SUUUUUUUUUUIIIIIIIIIIIIIIIIIIIIIIII
After a quick test on 10 episodes (without a random pose init to simulate an actual fight) we get about 80% success on phase 1!!!
This is pretty impressive because there was only +-20hours of training (not even a day!)
Now the funny and unexpected thing is that the agent can actually still fight during phase 2?

NOTE: THIS IS A BIG IF I HAVEN'T PROPERLY TESTED IT YET
Turns out my env still supports unknown observations (which in our case is boss animations, specifically the attacks of phase 2) but since the agent doesn’t recognize them he kinda panics, doesn’t pick the best move (but not something truly horrific either) and ends up getting hit (and dies…).

I thought that I would have to train a completely different agent to avoid having to train one for phase 2 from scratch but maybe I can build upon the first agent? Not sure if this will work, i def need to test it out

28/05/2025
Ran a from scratch model on phase 2 just to test but for some reason the agent accidentally unequipped the sword so the reward got fucked since he's not dealing the normal amount of damage
-> deleted the model and log file

About the whole unknown boss animation, I couldn't find that move in the animations.yaml for attack 1500 of whatever it was so maybe the problem doesn't come from the agent not knowing it but rather the env missing it? Maybe I can modify the file to add that move and see if that fixes things? It will only add one extra obs and then i can train one agent on both phase 1 and phase 2

Error:
_step: Unknown boss animation Attack1500
Unknown key -1 encountered
I don't know what the unknown key error is though so I also need to look into that. On the bright side it doesn't make the code crash

03/06/2025
Trained a new model from scratch only on phase 2 for about 20-25 hours.
Works pretty decent but the performance of phase 2 is tightly linked to phase 1 (if agent 1 made it to phase 2 but with very low hp then agent 2 is kinda cooked because it has no room for error)

04/06/2025
Loaded checkpoint 8 phase 1 and gave it more training time and saved it under checkpoint 9 phase 2. The idea is to improve it to reach phase 2 more consistently. It did a bit better but def needs more training time

05/06/2025
Reworked the testing function to include both policies (1 and 2)
Had to fetch the phase of the boss before passing it to the custom obs preprocessing bc the agent doesn't actually use the phase info (it's irrelevant to training). It was just one or two lines to changes so no big deal
I did some testing, I get about a 30% win ratio on the full boss right now.
Nothing crazy but we're getting close to the 45% win ratio of the library's author.
Then again according to his reddit post he did waaay more training time so I can also increase mine. He says this to quote him: "It took me some time, but I was able to train an agent with Duelling Double Deep Q-Learning that has a win rate of about 45% within a few days of training." A few days if he means actual 24/7 training could be like 100hours if 4 days of training. I did like half of that and I'm getting close.