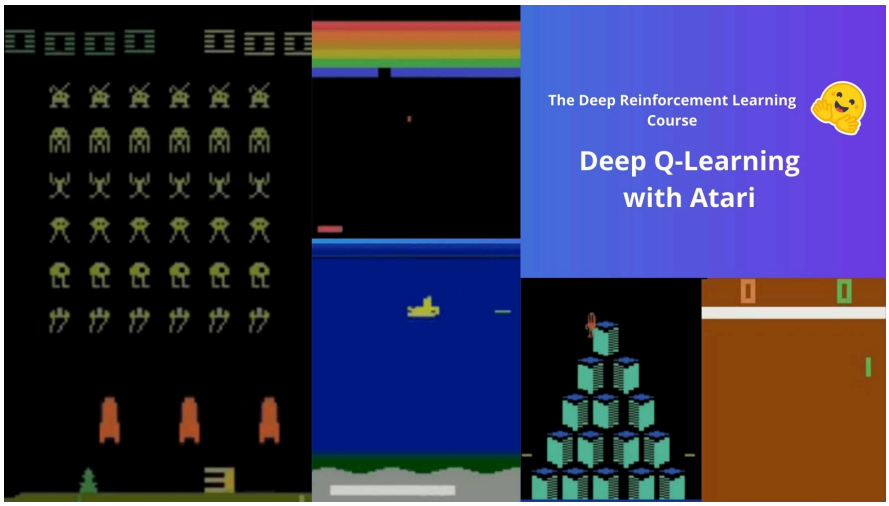
Unit 3: Deep Q-Learning with Atari Games 👾 using RL Baselines3 Zoo



In this notebook, **you'll train a Deep Q-Learning agent** playing Space Invaders using RL Baselines3 Zoo, a training framework based on Stable-Baselines3 that provides scripts for training, evaluating agents, tuning hyperparameters, plotting results and recording videos.

We're using the RL-Baselines-3 Zoo integration, a vanilla version of Deep Q-Learning with no extensions such as Double-DQN, Dueling-DQN, and Prioritized Experience Replay.

⬇️ Here is an example of what **you will achieve** ⬇️

```
%%html
<video controls autoplay><source src="https://huggingface.co/ThomasSimonini/ppo-Spa
```

🎮 Environments:

- SpacesInvadersNoFrameskip-v4

You can see the difference between Space Invaders versions here 👉 https://gymnasium.farama.org/environments/atari/space_invaders/#variants

📚 RL-Library:

- RL-Baselines3-Zoo

Objectives of this notebook 🏆

At the end of the notebook, you will:

- Be able to understand deeper **how RL Baselines3 Zoo works**.
- Be able to **push your trained agent and the code to the Hub** with a nice video replay and an evaluation score 🔥.

atari_wrappers.py dqn.yml

```
SpaceInvadersNoFrameskip-v4:
  env_wrapper:
    - stable_baselines3.common.atari_wrappers.Ata
  frame_stack: 4
  policy: 'CnnPolicy'
  n_timesteps: !!float 1e6
  buffer_size: 100000
  learning_rate: !!float 1e-4
  batch_size: 32
  learning_starts: 100000
  target_update_interval: 1000
  train_freq: 4
  gradient_steps: 1
  exploration_fraction: 0.1
  exploration_final_eps: 0.01
  # If True, you need to deactivate handle_timeou
  # in the replay_buffer_kwargs
  optimize_memory_usage: False
```

This notebook is from Deep Reinforcement Learning Course

In this free course, you will:

- 📖 Study Deep Reinforcement Learning in **theory and practice**.
- 🧑‍💻 Learn to **use famous Deep RL libraries** such as Stable Baselines3, RL Baselines3 Zoo, CleanRL and Sample Factory 2.0.
- 🤖 Train **agents in unique environments**

And more check 💻 the syllabus 👉 https://simoninithomas.github.io/deep-rl-course

Don't forget to **sign up to the course** (we are collecting your email to be able to **send you the links when each Unit is published and give you information about the challenges and updates).**

The best way to keep in touch is to join our discord server to exchange with the community and with us 👉🏻 https://discord.gg/ydHrjt3WP5

Prerequisites 🏗️

Before diving into the notebook, you need to:

☐ 💻 **Study Deep Q-Learning by reading Unit 3** 🤗

We're constantly trying to improve our tutorials, so **if you find some issues in this notebook**, please open an issue on the Github Repo.

Let's train a Deep Q-Learning agent playing Atari' Space Invaders 👾 and upload it to the Hub.

We strongly recommend students **to use Google Colab for the hands-on exercises instead of running them on their personal computers**.

By using Google Colab, **you can focus on learning and experimenting without worrying about the technical aspects of setting up your environments**.

To validate this hands-on for the certification process, you need to push your trained model to the Hub and **get a result of >= 200**.

To find your result, go to the leaderboard and find your model, **the result = mean_reward - std of reward**

For more information about the certification process, check this section 👉 https://huggingface.co/deep-rl-course/en/unit0/introduction#certification-process

An advice 💡

It's better to run this colab in a copy on your Google Drive, so that **if it timeouts** you still have the saved notebook on your Google Drive and do not need to fill everything from scratch.

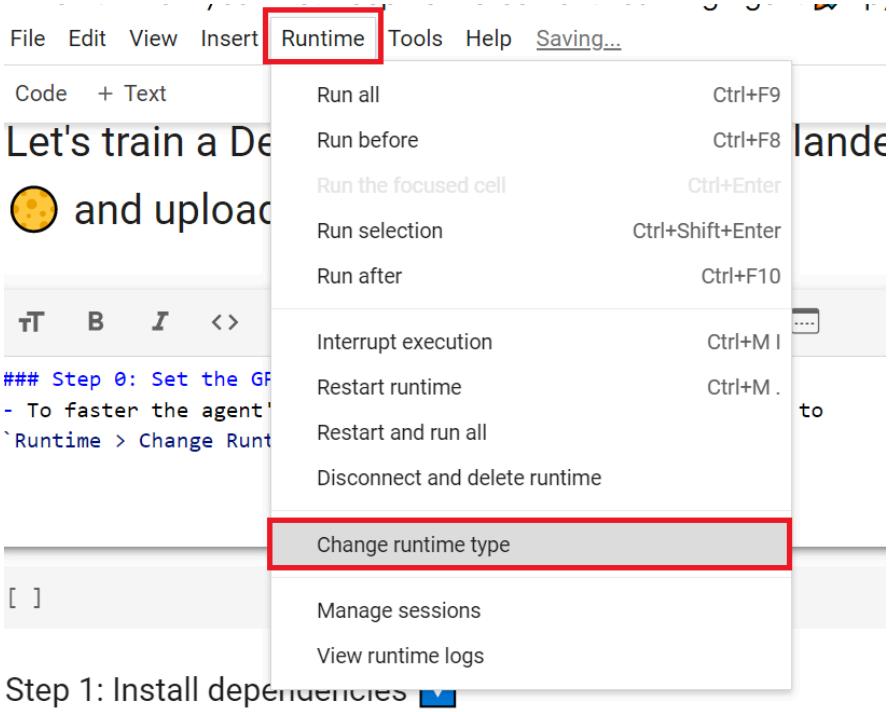
To do that you can either do `Ctrl + S` or `File > Save a copy in Google Drive.`

Also, we're going to **train it for 90 minutes with 1M timesteps**. By typing `!nvidia-smi` will tell you what GPU you're using.

And if you want to train more such 10 million steps, this will take about 9 hours, potentially resulting in Colab timing out. In that case, I recommend running this on your local computer (or somewhere else). Just click on: `File>Download`.

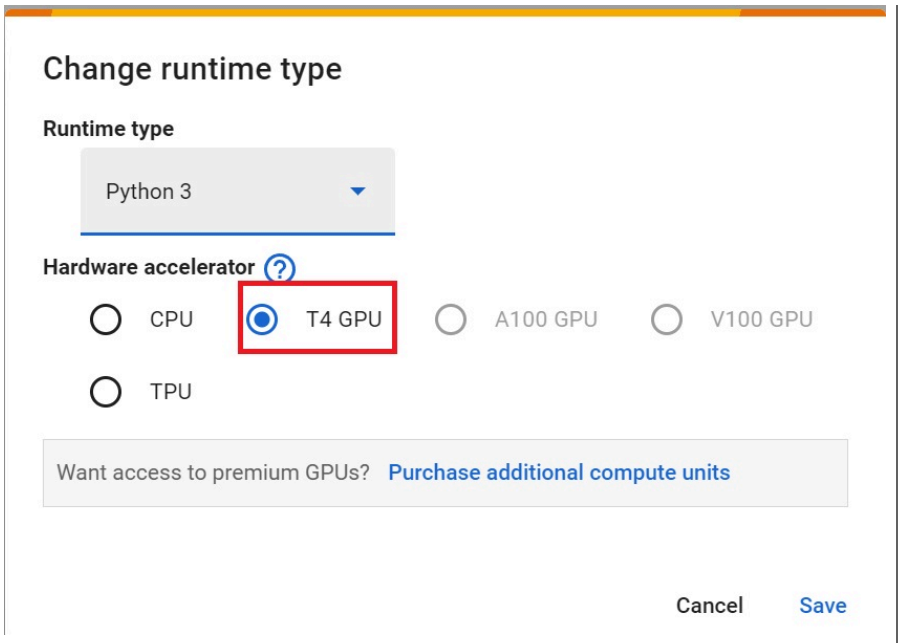
Set the GPU 💪

- To **accelerate the agent's training, we'll use a GPU**. To do that, go to `Runtime > Change Runtime type`



Step 1: Install dependencies

- `Hardware Accelerator > GPU`



Install RL-Baselines3 Zoo and its dependencies

If you see `ERROR: pip's dependency resolver does not currently take into account all the packages that are installed.` **this is normal and it's not a critical error** there's a conflict of version. But the packages we need are installed.

```
# For now we install this update of RL-Baselines3 Zoo
!pip install git+https://github.com/DLR-RM/rl-baselines3-zoo
```

```
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dis
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/d
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/d
Requirement already satisfied: mpmath<1.4.0,>=1.1.0 in /usr/local/lib/python3.10
Building wheels for collected packages: rl_zoo3
  Building wheel for rl_zoo3 (pyproject.toml) ... done
  Created wheel for rl_zoo3: filename=rl_zoo3-2.4.0a4-py3-none-any.whl size=7606
  Stored in directory: /tmp/pip-ephem-wheel-cache-wmysaqzt/wheels/5f/9b/c0/8af7a
Successfully built rl_zoo3
Installing collected packages: farama-notifications, tcolorpy, pathvalidate, nvi
Successfully installed DataProperty-1.0.1 Mako-1.3.5 alembic-1.13.2 colorlog-6.8
```

IF AND ONLY IF THE VERSION ABOVE DOES NOT EXIST ANYMORE. UNCOMMENT AND INSTALL THE ONE BELOW

```
#!pip install rl_zoo3==2.0.0a9
```

```
!apt-get install swig cmake ffmpeg
```

```
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
cmake is already the newest version (3.22.1-1ubuntu1.22.04.2).
ffmpeg is already the newest version (7:4.4.2-0ubuntu0.22.04.1).
The following additional packages will be installed:
  swig4.0
Suggested packages:
  swig-doc swig-examples swig4.0-examples swig4.0-doc
The following NEW packages will be installed:
  swig swig4.0
0 upgraded, 2 newly installed, 0 to remove and 45 not upgraded.
Need to get 1,116 kB of archives.
After this operation, 5,542 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/universe amd64 swig4.0 amd64 4.0.2-1u
Get:2 http://archive.ubuntu.com/ubuntu jammy/universe amd64 swig all 4.0.2-1ubuntu
Fetched 1,116 kB in 2s (476 kB/s)
Selecting previously unselected package swig4.0.
(Reading database ... 121925 files and directories currently installed.)
Preparing to unpack .../swig4.0_4.0.2-1ubuntu1_amd64.deb ...
Unpacking swig4.0 (4.0.2-1ubuntu1) ...
Selecting previously unselected package swig.
Preparing to unpack .../swig_4.0.2-1ubuntu1_all.deb ...
Unpacking swig (4.0.2-1ubuntu1) ...
Setting up swig4.0 (4.0.2-1ubuntu1) ...
Setting up swig (4.0.2-1ubuntu1) ...
Processing triggers for man-db (2.10.2-1) ...
```

To be able to use Atari games in Gymnasium we need to install atari package. And accept-rom-license to download the rom files (games files).

```
!pip install gymnasium[atari]
!pip install gymnasium[accept-rom-license]
```

```
Requirement already satisfied: gymnasium[atari] in /usr/local/lib/python3.10/dist-
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: cloudpickle>=1.2.0 in /usr/local/lib/python3.10/dis
Requirement already satisfied: typing-extensions>=4.3.0 in /usr/local/lib/python3.
Requirement already satisfied: farama-notifications>=0.0.1 in /usr/local/lib/pythc
Collecting shimmy[atari]<1.0,>=0.1.0 (from gymnasium[atari])
  Downloading Shimmy-0.2.1-py3-none-any.whl (25 kB)
Collecting ale-py~=0.8.1 (from shimmy[atari]<1.0,>=0.1.0->gymnasium[atari])
  Downloading ale_py-0.8.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.7/1.7 MB 42.9 MB/s eta 0:00:00
Requirement already satisfied: importlib-resources in /usr/local/lib/python3.10/di
Installing collected packages: ale-py, shimmy
Successfully installed ale-py-0.8.1 shimmy-0.2.1
Requirement already satisfied: gymnasium[accept-rom-license] in /usr/local/lib/pyt
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: cloudpickle>=1.2.0 in /usr/local/lib/python3.10/dis
Requirement already satisfied: typing-extensions>=4.3.0 in /usr/local/lib/python3.
Requirement already satisfied: farama-notifications>=0.0.1 in /usr/local/lib/pythc
Collecting autorom[accept-rom-license]~=0.4.2 (from gymnasium[accept-rom-license])
  Downloading AutoROM-0.4.2-py3-none-any.whl (16 kB)
```

```
  Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (f
  Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages
  Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (fr
  Collecting AutoROM.accept-rom-license (from autorom[accept-rom-license]~=0.4.2->gy
    Downloading AutoROM.accept-rom-license-0.6.1.tar.gz (434 kB)
       ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 434.7/434.7 kB 11.1 MB/s eta 0:00:00
    Installing build dependencies ... done
    Getting requirements to build wheel ... done
    Preparing metadata (pyproject.toml) ... done
  Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.
  Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-pack
  Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dis
  Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dis
  Building wheels for collected packages: AutoROM.accept-rom-license
    Building wheel for AutoROM.accept-rom-license (pyproject.toml) ... done
    Created wheel for AutoROM.accept-rom-license: filename=AutoROM.accept_rom_licens
    Stored in directory: /root/.cache/pip/wheels/6b/1b/ef/a43ff1a2f1736d5711faa1ba4c
  Successfully built AutoROM.accept-rom-license
  Installing collected packages: AutoROM.accept-rom-license, autorom
  Successfully installed AutoROM.accept-rom-license-0.6.1 autorom-0.4.2
```

Create a virtual display 🔽

During the notebook, we'll need to generate a replay video. To do so, with colab, **we need to have a virtual screen to be able to render the environment** (and thus record the frames).

Hence the following cell will install the librairies and create and run a virtual screen 🖥

```
%%capture
!apt install python-opengl
!apt install xvfb
!pip3 install pyvirtualdisplay
```

```
import os

os.kill(os.getpid(), 9)
```

```
# Virtual display
from pyvirtualdisplay import Display

virtual_display = Display(visible=0, size=(1400, 900))
virtual_display.start()
```

```
<pyvirtualdisplay.display.Display at 0x7cf8d772f2e0>
```

Train our Deep Q-Learning Agent to Play Space Invaders 👾

To train an agent with RL-Baselines3-Zoo, we just need to do two things:

1. Create a hyperparameter config file that will contain our training hyperparameters called `dqn.yml`.

This is a template example:

```
SpaceInvadersNoFrameskip-v4:
  env_wrapper:
    - stable_baselines3.common.atari_wrappers.AtariWrapper
  frame_stack: 4
  policy: 'CnnPolicy'
  n_timesteps: !!float 1e6
  buffer_size: 100000
  learning_rate: !!float 1e-4
  batch_size: 32
  learning_starts: 100000
  target_update_interval: 1000
  train_freq: 4
  gradient_steps: 1
  exploration_fraction: 0.1
```

```
exploration_final_eps: 0.01
# If True, you need to deactivate handle_timeout_termination
# in the replay_buffer_kwargs
optimize_memory_usage: False
```

Here we see that:

- We use the `Atari Wrapper` that preprocess the input (Frame reduction ,grayscale, stack 4 frames)
- We use `CnnPolicy`, since we use Convolutional layers to process the frames
- We train it for 10 million `n_timesteps`
- Memory (Experience Replay) size is 100000, aka the amount of experience steps you saved to train again your agent with.

💡 My advice is to **reduce the training timesteps to 1M,** which will take about 90 minutes on a P100. `!nvidia-smi` will tell you what GPU you're using. At 10 million steps, this will take about 9 hours, which could likely result in Colab timing out. I recommend running this on your local computer (or somewhere else). Just click on: `File>Download`.

In terms of hyperparameters optimization, my advice is to focus on these 3 hyperparameters:

- `learning_rate`
- `buffer_size (Experience Memory size)`
- `batch_size`

As a good practice, you need to **check the documentation to understand what each hyperparameters does**: https://stable-baselines3.readthedocs.io/en/master/modules/dqn.html#parameters

2. We start the training and save the models on `logs` folder 📁

- Define the algorithm after `--algo`, where we save the model after `-f` and where the hyperparameter config is after `-c`.

```
!python -m rl_zoo3.train --algo dqn --env SpaceInvadersNoFrameskip-v4  -f logs/  -c
```

```
| train/              |          |
|    learning_rate    | 0.0001   |
|    loss             | 0.0218   |
|    n_updates        | 224932   |
----------------------------------
Eval num_timesteps=1000000, episode_reward=785.00 +/- 273.06
Episode length: 4740.00 +/- 1501.84
----------------------------------
| eval/               |          |
|    mean_ep_length   | 4.74e+03 |
|    mean_reward      | 785      |
| rollout/            |          |
|    exploration_rate | 0.01     |
| time/               |          |
|    total_timesteps  | 1000000  |
| train/              |          |
|    learning_rate    | 0.0001   |
|    loss             | 0.0203   |
|    n_updates        | 224999   |
----------------------------------
New best mean reward!
Saving to logs//dqn/SpaceInvadersNoFrameskip-v4_1
```

Solution

[ ] ↳ *1 cell hidden*

Let's evaluate our agent 👀

- RL-Baselines3-Zoo provides `enjoy.py`, a python script to evaluate our agent. In most RL libraries, we call the evaluation script `enjoy.py`.
- Let's evaluate it for 5000 timesteps 🔥

```
!python -m rl_zoo3.enjoy  --algo dqn  --env SpaceInvadersNoFrameskip-v4  --no-rende
```

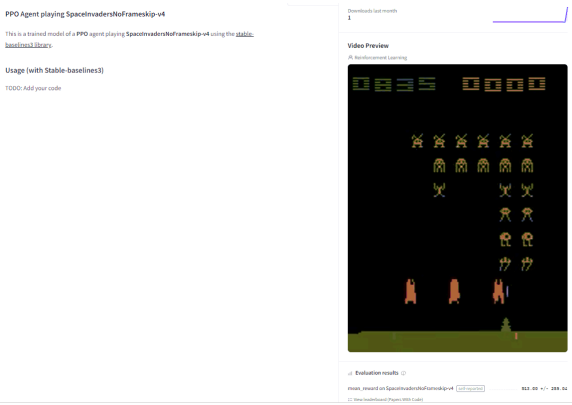
```
2024-07-01 18:30:00.990155: E external/local_xla/xla/stream_executor/cuda/cuda_dnr
2024-07-01 18:30:00.990215: E external/local_xla/xla/stream_executor/cuda/cuda_fft
2024-07-01 18:30:00.991750: E external/local_xla/xla/stream_executor/cuda/cuda_bla
2024-07-01 18:30:01.002840: I tensorflow/core/platform/cpu_feature_guard.cc:182] T
To enable the following instructions: AVX2 AVX512F FMA, in other operations, rebui
2024-07-01 18:30:02.151868: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38
Loading latest experiment, id=1
Loading logs/dqn/SpaceInvadersNoFrameskip-v4_1/SpaceInvadersNoFrameskip-v4.zip
A.L.E: Arcade Learning Environment (version 0.8.1+53f58b7)
[Powered by Stella]
Stacking 4 frames
Atari Episode Score: 550.00
Atari Episode Length 3793
Atari Episode Score: 550.00
Atari Episode Length 3386
Atari Episode Score: 495.00
Atari Episode Length 3267
Atari Episode Score: 800.00
Atari Episode Length 5181
Atari Episode Score: 605.00
Atari Episode Length 3687
```

Solution

[ ] ↳ *1 cell hidden*

Publish our trained model on the Hub 🚀

Now that we saw we got good results after the training, we can publish our trained model on the hub 🤗 with one line of code.



By using `rl_zoo3.push_to_hub` **you evaluate, record a replay, generate a model card of your agent and push it to the hub**.

This way:

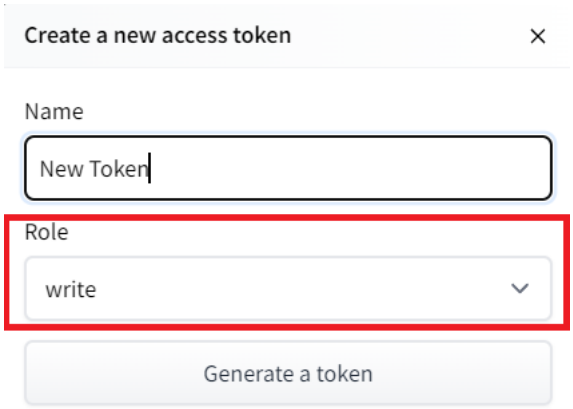
- You can **showcase our work** 🔥
- You can **visualize your agent playing** 👀
- You can **share with the community an agent that others can use** 💾
- You can **access a leaderboard** 🏆 **to see how well your agent is performing compared to your classmates** 👉 https://huggingface.co/spaces/huggingface-projects/Deep-Reinforcement-Learning-Leaderboard

To be able to share your model with the community there are three more steps to follow:

1️⃣ (If it's not already done) create an account to HF ➡ https://huggingface.co/join

2️⃣ Sign in and then, you need to store your authentication token from the Hugging Face website.

- Create a new token (https://huggingface.co/settings/tokens) **with write role**



- Copy the token
- Run the cell below and past the token

```
from huggingface_hub import notebook_login # To log to our Hugging Face account to b
notebook_login()
!git config --global credential.helper store
```

```
Token is valid (permission: write).
saved in your configured git crede
is been saved to /root/.cache/hugg
Login successful
```

If you don't want to use a Google Colab or a Jupyter Notebook, you need to use this command instead: `huggingface-cli login`

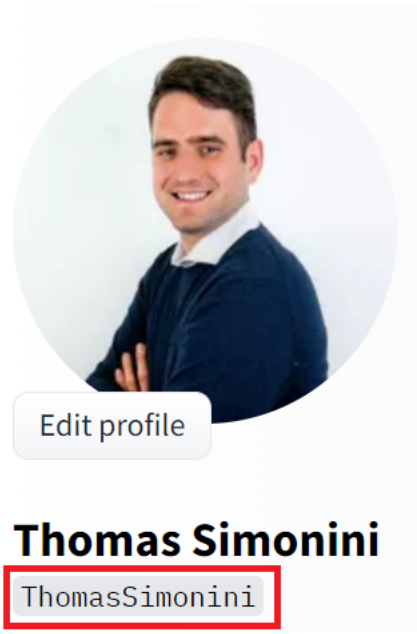
3️⃣ We're now ready to push our trained agent to the 🤗 Hub 🔥

Let's run push_to_hub.py file to upload our trained agent to the Hub.

`--repo-name` : The name of the repo

`-orga` : Your Hugging Face username

`-f` : Where the trained model folder is (in our case `logs`)




```
1 !python -m rl_zoo3.push_to_hub  --algo dqn  --env SpaceInvadersNoFrameskip-v4  --repo
```

```
Upload file dqn-SpaceInvadersNoFrameskip-v4/policy.pth: 100% 12.9M/12.9M [00:06<

Upload file dqn-SpaceInvadersNoFrameskip-v4/policy.optimizer.pth: 100% 12.9M/12

Upload file dqn-SpaceInvadersNoFrameskip-v4/pytorch_variables.pth: 100% 864/864



Upload file replay.mp4: 100% 226k/226k [00:06<00:00, 38.5kB/s]
ℹ Your model is pushed to the hub. You can view your model here:
https://huggingface.co/eseskay/dqn-SpaceInvadersNoFrameskip-v4
```

Solution

[ ] ↳ *1 cell hidden*

.

Congrats 🥳 you've just trained and uploaded your first Deep Q-Learning agent using RL-Baselines-3 Zoo. The script above should have displayed a link to a model repository such as https://huggingface.co/ThomasSimonini/dqn-SpaceInvadersNoFrameskip-v4. When you go to this link, you can:

- See a **video preview of your agent** at the right.
- Click "Files and versions" to see all the files in the repository.
- Click "Use in stable-baselines3" to get a code snippet that shows how to load the model.
- A model card (`README.md` file) which gives a description of the model and the hyperparameters you used.

Under the hood, the Hub uses git-based repositories (don't worry if you don't know what git is), which means you can update the model with new versions as you experiment and improve your agent.

Compare the results of your agents with your classmates using the leaderboard 🏆

Load a powerful trained model 🔥

- The Stable-Baselines3 team uploaded **more than 150 trained Deep Reinforcement Learning agents on the Hub**.

You can find them here: 👉 https://huggingface.co/sb3

Some examples:

- Asteroids: https://huggingface.co/sb3/dqn-AsteroidsNoFrameskip-v4
- Beam Rider: https://huggingface.co/sb3/dqn-BeamRiderNoFrameskip-v4
- Breakout: https://huggingface.co/sb3/dqn-BreakoutNoFrameskip-v4
- Road Runner: https://huggingface.co/sb3/dqn-RoadRunnerNoFrameskip-v4

Let's load an agent playing Beam Rider: https://huggingface.co/sb3/dqn-BeamRiderNoFrameskip-v4

```
%%html
<video controls autoplay><source src="https://huggingface.co/sb3/dqn-BeamRiderNoFra
```

1. We download the model using `rl_zoo3.load_from_hub`, and place it in a new folder that we can call `rl_trained`

```
# Download model and save it into the logs/ folder
!python -m rl_zoo3.load_from_hub --algo dqn --env BeamRiderNoFrameskip-v4 -orga sb3
```

```
2024-07-01 18:32:44.687464: E external/local_xla/xla/stream_executor/cuda/cuda_dnr
2024-07-01 18:32:44.689423: E external/local_xla/xla/stream_executor/cuda/cuda_fft
2024-07-01 18:32:44.691347: E external/local_xla/xla/stream_executor/cuda/cuda_bla
2024-07-01 18:32:44.701950: I tensorflow/core/platform/cpu_feature_guard.cc:182] T
To enable the following instructions: AVX2 AVX512F FMA, in other operations, rebui
2024-07-01 18:32:46.131128: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38
Downloading from https://huggingface.co/sb3/dqn-BeamRiderNoFrameskip-v4
dqn-BeamRiderNoFrameskip-v4.zip: 100% 27.2M/27.2M [00:02<00:00, 9.73MB/s]
config.yml: 100% 548/548 [00:00<00:00, 3.46MB/s]
No normalization file
args.yml: 100% 887/887 [00:00<00:00, 5.79MB/s]
env_kwargs.yml: 100% 3.00/3.00 [00:00<00:00, 19.5kB/s]
train_eval_metrics.zip: 100% 244k/244k [00:00<00:00, 377kB/s]
Saving to rl_trained/dqn/BeamRiderNoFrameskip-v4_1
```

2. Let's evaluate if for 5000 timesteps

```
!python -m rl_zoo3.enjoy --algo dqn --env BeamRiderNoFrameskip-v4 -n 5000  -f rl_tra
```

```
2024-07-01 18:33:00.100257: E external/local_xla/xla/stream_executor/cuda/cuda_dnr
2024-07-01 18:33:00.100312: E external/local_xla/xla/stream_executor/cuda/cuda_fft
2024-07-01 18:33:00.101621: E external/local_xla/xla/stream_executor/cuda/cuda_bla
2024-07-01 18:33:00.108750: I tensorflow/core/platform/cpu_feature_guard.cc:182] T
To enable the following instructions: AVX2 AVX512F FMA, in other operations, rebui
2024-07-01 18:33:01.169925: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38
Loading latest experiment, id=1
Loading rl_trained/dqn/BeamRiderNoFrameskip-v4_1/BeamRiderNoFrameskip-v4.zip
A.L.E: Arcade Learning Environment (version 0.8.1+53f58b7)
[Powered by Stella]
Stacking 4 frames
/usr/local/lib/python3.10/dist-packages/stable_baselines3/common/save_util.py:167:
Exception: 'bytes' object cannot be interpreted as an integer
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/stable_baselines3/common/vec_env/patch_gym
  warnings.warn(
Atari Episode Score: 3028.00
Atari Episode Length 14816
```

Why not trying to train your own **Deep Q-Learning Agent playing BeamRiderNoFrameskip-v4? 🏆.**

If you want to try, check https://huggingface.co/sb3/dqn-BeamRiderNoFrameskip-v4#hyperparameters **in the model card, you have the hyperparameters of the trained agent.**

But finding hyperparameters can be a daunting task. Fortunately, we'll see in the next Unit, how we can **use Optuna for optimizing the Hyperparameters 🔥.**

Some additional challenges 🏆

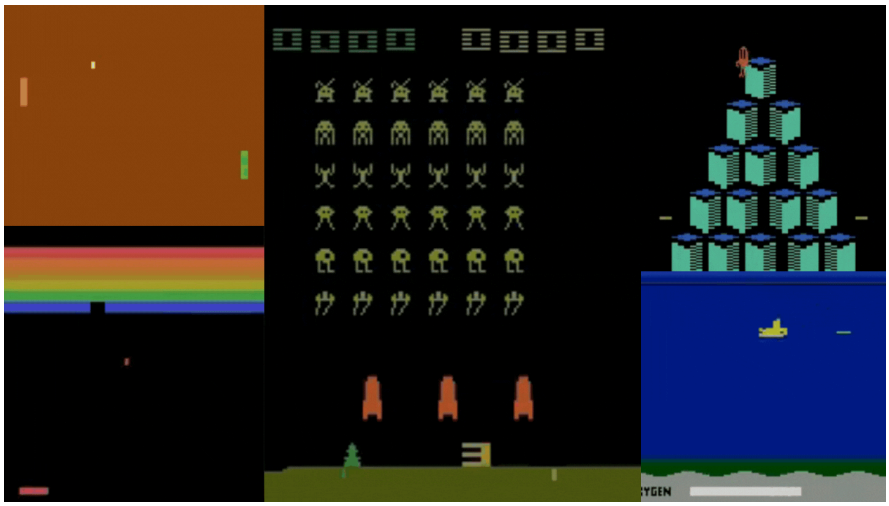
The best way to learn **is to try things by your own**!

In the Leaderboard you will find your agents. Can you get to the top?

Here's a list of environments you can try to train your agent with:

- BeamRiderNoFrameskip-v4
- BreakoutNoFrameskip-v4
- EnduroNoFrameskip-v4
- PongNoFrameskip-v4

Also, **if you want to learn to implement Deep Q-Learning by yourself**, you definitely should look at CleanRL implementation:
https://github.com/vwxyzjn/cleanrl/blob/master/cleanrl/dqn_atari.py



Congrats on finishing this chapter!

If you're still feel confused with all these elements...it's totally normal! **This was the same for me and for all people who studied RL.**

Take time to really **grasp the material before continuing and try the additional challenges**. It's important to master these elements and having a solid foundations.

In the next unit, **we're going to learn about Optuna**. One of the most critical task in Deep Reinforcement Learning is to find a good set of training hyperparameters. And Optuna is a library that helps you to automate the search.

This is a course built with you