Unit 5: An Introduction to ML-Agents

Setup

🎮 Environments:

- Pyramids
- SnowballTarget

📚 RL-Library:

- ML-Agents

Let's train our agents 🚀

To validate this hands-on for the certification process, you just need to push your trained models to the Hub. There’s no results to attain to validate this one. But if you want to get nice results you can try to attain:

- For `Pyramids` : Mean Reward = 1.75
- For `SnowballTarget` : Mean Reward = 15 or 30 targets hit in an episode.

Clone the repository 🔽

- We need to clone the repository, that contains **ML-Agents.**

```
%%capture
# Clone the repository (can take 3min)
!git clone --depth 1 https://github.com/Unity-Technologies/ml-agents
```

Setup the Virtual Environment 🔽

- In order for the **ML-Agents** to run successfully in Colab, Colab's Python version must meet the library's Python requirements.
- We can check for the supported Python version under the `python_requires` parameter in the `setup.py` files. These files are required to set up the **ML-Agents** library for use and can be found in the following locations:
    - `/content/ml-agents/ml-agents/setup.py`
    - `/content/ml-agents/ml-agents-envs/setup.py`
- Colab's Current Python version(can be checked using `!python --version`) doesn't match the library's `python_requires` parameter, as a result installation may silently fail and lead to errors like these, when executing the same commands later:
    - `/bin/bash: line 1: mlagents-learn: command not found`
    - `/bin/bash: line 1: mlagents-push-to-hf: command not found`
- To resolve this, we'll create a virtual environment with a Python version compatible with the **ML-Agents** library.

`Note:` *For future compatibility, always check the `python_requires` parameter in the installation files and set your virtual environment to the maximum supported Python version in the given below script if the Colab's Python version is not compatible*

```
# Colab's Current Python Version (Incompatible with ML-Agents)
!python --version
```

```
# Install virtualenv and create a virtual environment
!pip install virtualenv
!virtualenv myenv

# Download and install Miniconda
!wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
!chmod +x Miniconda3-latest-Linux-x86_64.sh
!./Miniconda3-latest-Linux-x86_64.sh -b -f -p /usr/local

# Activate Miniconda and install Python ver 3.10.12
!source /usr/local/bin/activate
!conda install -q -y --prefix /usr/local python=3.10.12 ujson  # Specify the version here

# Set environment variables for Python and conda paths
```

```
!export PYTHONPATH=/usr/local/lib/python3.10/site-packages/
!export CONDA_PREFIX=/usr/local/envs/myenv
```

```
# Python Version in New Virtual Environment (Compatible with ML-Agents)
!python --version
```

Installing the dependencies 🔽

```
%%capture
# Go inside the repository and install the package (can take 3min)

%cd ml-agents
# Do take note that the current dir is ./ml-agents


!pip3 install -e ./ml-agents-envs
!pip3 install -e ./ml-agents
```

SnowballTarget ⛄

If you need a refresher on how this environments work check this section 👉 https://huggingface.co/deep-rl-course/unit5/snowball-target

1. Download and move the environment zip file in `./training-envs-executables/linux/`

- Our environment executable is in a zip file.
- We need to download it and place it to `./training-envs-executables/linux/`
- We use a linux executable because we use colab, and colab machines OS is Ubuntu (linux)

```
# Here, we create training-envs-executables and linux
# Do take note that the current dir is ./ml-agents

!mkdir ./training-envs-executables
!mkdir ./training-envs-executables/linux
```

We downloaded the file SnowballTarget.zip from https://github.com/huggingface/Snowball-Target using `wget`

```
!wget "https://github.com/huggingface/Snowball-Target/raw/main/SnowballTarget.zip" -O ./training-envs-executables/linux/SnowballTarg
```

We unzip the executable.zip file

```
%%capture
!unzip -d ./training-envs-executables/linux/ ./training-envs-executables/linux/SnowballTarget.zip
```

Make sure your file is accessible

```
!chmod -R 755 ./training-envs-executables/linux/SnowballTarget
```

2. Define the SnowballTarget config file

- In ML-Agents, you define the **training hyperparameters into config.yaml files.**

There are multiple hyperparameters. To know them better, you should check for each explanation with the documentation

So you need to create a `SnowballTarget.yaml` config file in ./content/ml-agents/config/ppo/

We'll give you here a first version of this config (to copy and paste into your `SnowballTarget.yaml file`), **but you should modify it**.

```
behaviors:
  SnowballTarget:
    trainer_type: ppo
    summary_freq: 10000
    keep_checkpoints: 10
    checkpoint_interval: 50000
    max_steps: 200000
    time_horizon: 64
    threaded: false
    hyperparameters:
      learning_rate: 0.0003
      learning_rate_schedule: linear
      batch_size: 128
      buffer_size: 2048
      beta: 0.005
      epsilon: 0.2
      lambd: 0.95
      num_epoch: 3
    network_settings:
      normalize: false
      hidden_units: 256
      num_layers: 2
      vis_encode_type: simple
    reward_signals:
      extrinsic:
        gamma: 0.99
        strength: 1.0
```

3. Train the agent

To train our agent, we just need to **launch mlagents-learn and select the executable containing the environment.**

We define four parameters:

1. `mlagents-learn <config>`: the path where the hyperparameter config file is.
2. `--env`: where the environment executable is.
3. `--run_id`: the name you want to give to your training run id.
4. `--no-graphics`: to not launch the visualization during the training.

Train the model and use the `--resume` flag to continue training in case of interruption.

> It will fail first time if and when you use `--resume`, try running the block again to bypass the error.

```
# Do take note that the current dir is ./ml-agents

!mlagents-learn ./config/ppo/SnowballTarget.yaml --env=./training-envs-executables/linux/SnowballTarget/SnowballTarget.x86_64 \
--run-id="SnowballTarget1" --no-graphics --resume
```

4. Push the agent to the 🤗 Hub

- Now that we trained our agent, we're **ready to push it to the Hub to be able to visualize it playing on your browser🔥.**

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 paste the token

```
from huggingface_hub import notebook_login
notebook_login()
```

Then, we simply need to run `mlagents-push-to-hf`.

And we define 4 parameters:

1. `--run-id`: the name of the training run id.
2. `--local-dir`: where the agent was saved, it's results/, so in my case results/First Training.
3. `--repo-id`: the name of the Hugging Face repo you want to create or update. It's always / If the repo does not exist **it will be created automatically**
4. `--commit-message`: since HF repos are git repository you need to define a commit message.

For instance:

`!mlagents-push-to-hf --run-id="SnowballTarget1" --local-dir="./results/SnowballTarget1" --repo-id="ThomasSimonini/ppo-SnowballTarget" --commit-message="First Push"`

```
!mlagents-push-to-hf  --run-id="SnowballTarget1"  --local-dir="./results/SnowballTarget1"  \
--repo-id="wengti0608/ppo-SnowballTarget1"  --commit-message="First attempt"
```

5. Watch your agent playing 👀

For this step it’s simple:

1. Go here: https://huggingface.co/spaces/ThomasSimonini/ML-Agents-SnowballTarget
2. Launch the game and put it in full screen by clicking on the bottom right button

1. In step 1, type your username (your username is case sensitive: for instance, my username is ThomasSimonini not thomassimonini or ThOmasImoNInI) and click on the search button.
2. In step 2, select your model repository.
3. In step 3, **choose which model you want to replay**:
    - I have multiple ones, since we saved a model every 500000 timesteps.
    - But since I want the more recent, I choose `SnowballTarget.onnx`

👉 What’s nice **is to try with different models step to see the improvement of the agent.**

And don't hesitate to share the best score your agent gets on discord in #rl-i-made-this channel 🔥

Let's now try a harder environment called Pyramids...

Pyramids 🏆

1. Download and move the environment zip file in `./training-envs-executables/linux/`

- Our environment executable is in a zip file.
- We need to download it and place it to `./training-envs-executables/linux/`
- We use a linux executable because we use colab, and colab machines OS is Ubuntu (linux)

We downloaded the file Pyramids.zip from from https://huggingface.co/spaces/unity/ML-Agents-Pyramids/resolve/main/Pyramids.zip using `wget`

```
!wget "https://huggingface.co/spaces/unity/ML-Agents-Pyramids/resolve/main/Pyramids.zip" -O ./training-envs-executables/linux/Pyrami
```

We unzip the executable.zip file

```
%%capture
!unzip -d ./training-envs-executables/linux/ ./training-envs-executables/linux/Pyramids.zip
```

Make sure your file is accessible

```
!chmod -R 755 ./training-envs-executables/linux/Pyramids/Pyramids
```

2. Modify the PyramidsRND config file

- Contrary to the first environment which was a custom one, **Pyramids was made by the Unity team**.
- So the PyramidsRND config file already exists and is in ./content/ml-agents/config/ppo/PyramidsRND.yaml
- You might asked why "RND" in PyramidsRND. RND stands for *random network distillation* it's a way to generate curiosity rewards. If you want to know more on that we wrote an article explaning this technique: https://medium.com/data-from-the-trenches/curiosity-driven-learning-through-random-network-distillation-488ffd8e5938

For this training, we’ll modify one thing:

- The total training steps hyperparameter is too high since we can hit the benchmark (mean reward = 1.75) in only 1M training steps. 👉 To do that, we go to config/ppo/PyramidsRND.yaml,**and modify these to max_steps to 1000000.**

As an experimentation, you should also try to modify some other hyperparameters, Unity provides a very good documentation explaining each of them here.

We’re now ready to train our agent 🔥.

3. Train the agent

```
!mlagents-learn ./config/ppo/PyramidsRND.yaml --env=./training-envs-executables/linux/Pyramids/Pyramids --run-id="PyramidsTraining"
```

4. Push the agent to the 🤗 Hub

- Now that we trained our agent, we're **ready to push it to the Hub to be able to visualize it playing on your browser🔥.**

```
!mlagents-push-to-hf  --run-id="Pyramids Training"  --local-dir="./results/PyramidsTraining"  --repo-id="wengti0608/ppo-Pyramid"  --
```

5. Watch your agent playing 👀

👉 https://huggingface.co/spaces/unity/ML-Agents-Pyramids

Practice: Other Environment

You have the full list of the Unity official environments here 👉 https://github.com/Unity-Technologies/ml-agents/blob/develop/docs/Learning-Environment-Examples.md

For the demos to visualize your agent 👉 https://huggingface.co/unity

For now we have integrated:

- Worm demo where you teach a **worm to crawl**.
- Walker demo where you teach an agent **to walk towards a goal**.

1. Create an environment

- ML-Agents do not provide the pre-built executable environments for other environment.
- However, you can build the environment using the assets available in `./ml-agents/Project/Assets/ML-Agents/Examples` using Unity Engine!
- To learn how to build a new executable environment using Unity, refer to these tutorials: https://huggingface.co/learn/deep-rl-course/unit5/bonus

2. Create a config

- The config can be located in `./ml-agents/config/`

3. Train the agent

- If training was interrupted, it may be resumed with `--resume`

```
!mlagents-learn """Path to the config""" --env="""Path to the executable environment""" \
--run-id="""Run ID""" --no-graphics
```

4. Push to Hugging Face Hub

```
!mlagents-push-to-hf  --run-id="""Run ID (same as during training)"""  --local-dir="""Path to the result dir"""  \
--repo-id="""{user_id}/{repo_name}"""  --commit-message="""A message"""
```