Tutorial: Setting up a Reinforcement Learning pipeline for a Telco Core Network (part 2)

27^{th} Conference on Innovation in Clouds, Internet and Networks
ICIN 2024

Shantanu Verma [*]

[*]Orange Innovation Networks,
Gurgaon, India

Guillaume Fraysse [§]

[§]Orange Innovation Networks,
Châtillon, France

March 11^{th}, 2024

Installing the Gym environment and pipeline

Operating System	Linux	macOS	Windows
Repository access	Git command or download zip from GitHub		
Package installer and dependency management	pip , Miniconda or Anaconda		
Virtual environment	venv or Poetry		
Interpreter	CPython 3.9+		

A possible alternative is to use Google Colab, but you would miss the fun animation of the examples.

Installing the Gym environment and pipeline

Requires around 10GB to install and compile all dependencies

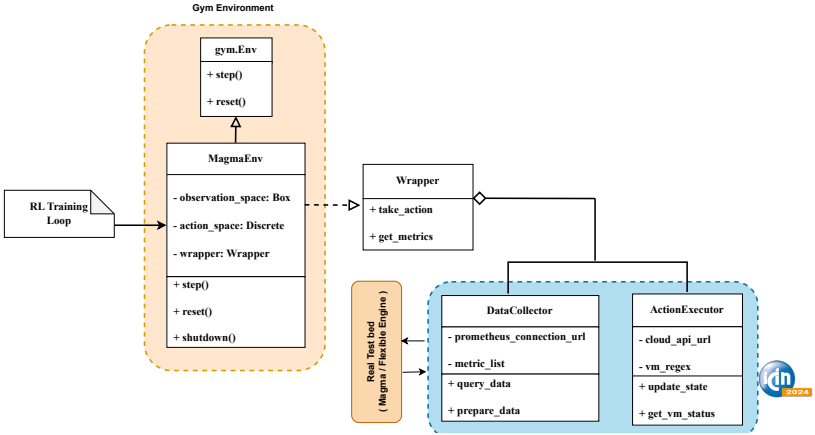
Installation, follow instructions from README.md

- git clone https://github.com/gfraysse/icin2024_tutorial.git
- python -m venv <your virtual env>
- source <your virtual env>/bin/activate
- pip install -r requirements.txt

Environment Definition (Gym like/compliant)

Role: provides standard interface for RL algorithms to interact with actual environment

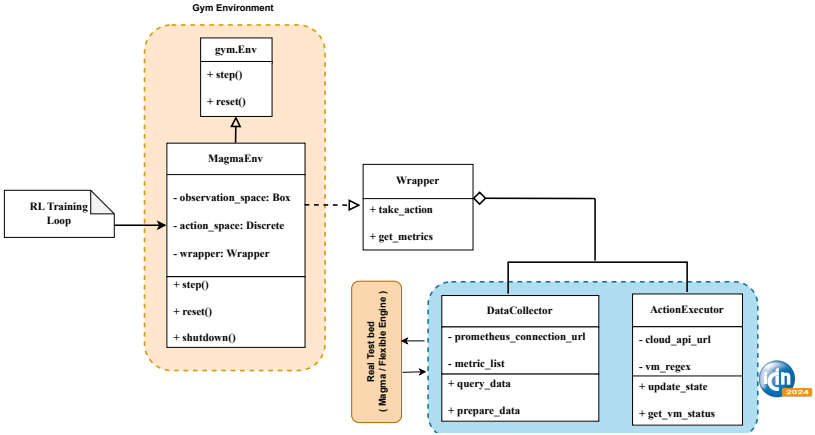
- **Gymnasium**: Open-source python library for developing & comparing reinforcement learning algorithms
  - Provides standard API for learning algorithms and environment to communicate
  - Set of environments. E.g. Classic Control, Atari, MuJoCo etc.
  - Drop-in replacement of OpenAI Gym since 2021



S. Verma, G. Fraysse

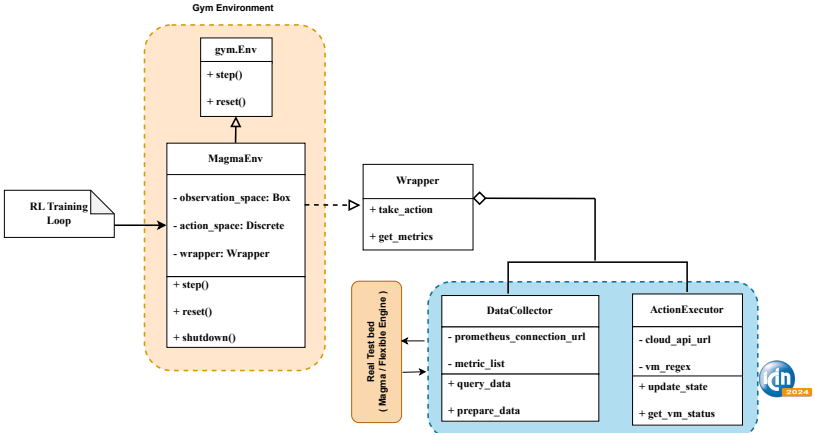
Environment Definition (Gym like/compliant)

- **MagmaEnv**: Environment definition complying with Gym
  - a way to manage the environment while adhering to MDP
  - helps the learning algorithm to control the environment
  - implements the crucial methods like ***step()*** and ***reset()***



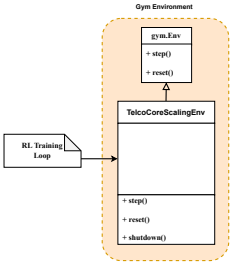
Environment Definition (Gym like/compliant)

- **step()**: accepts an 'action' and transitions the environment to a new state. It then returns:
  - a new state: in which agent finds itself after executing the action
  - reward: feedback for taking the action
  - terminated: indicates whether environment reached terminal state.
  - info: general info about environment
- **reset()**: Moves the environment to an initial state.



Environment Definition (Gym like/compliant)

- **TelcoCoreScalingEnv**: simulation environment mimicking Magma



Finilizing the installation the Gym environment and pipeline

Requires around 10GB to install and compile all dependencies

Installation, follow instructions from README.md

- git clone https://github.com/gfraysse/icin2024_tutorial.git
- python -m venv <your virtual env>
- source <your virtual env>/bin/activate
- pip install -r requirements.txt

Running an experiment

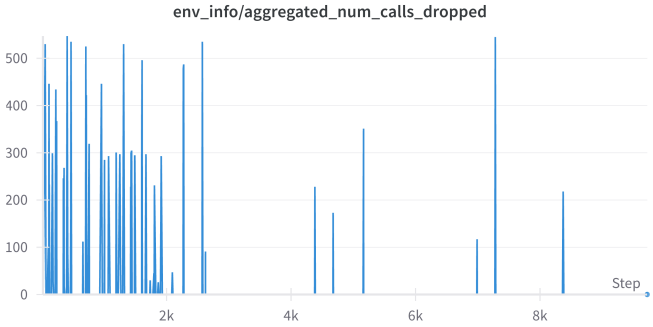
Adjust the configuration

- Configuration of the pipeline *config_per.yaml*
- Configuration of the environment *telco_core_scaling/envs/config/config_env.yaml*

Run an experiment

- python pipeline-exp_d3qn_per_sim.py

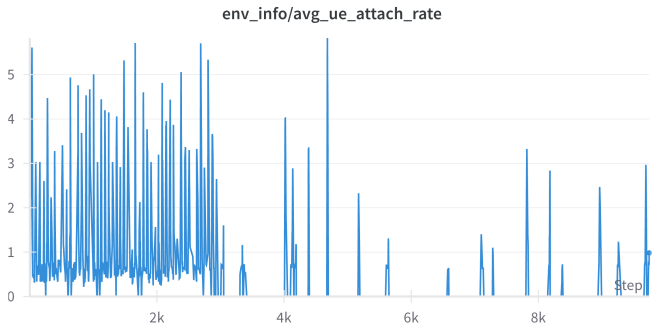
Analyzing the results



Number of calls dropped:

- Computed by the pipeline
- Measures the number of sessions that could not be initiated

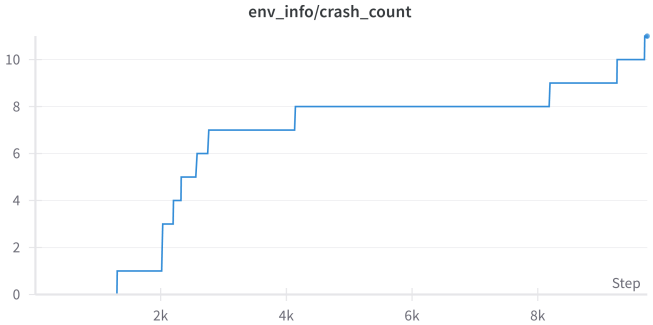
Analyzing the results



UE attach rate:

- Metric collected from Magma NMS
- Attach rate of the User Equipments (UEs) connecting to the Access Gateways (AGWs)

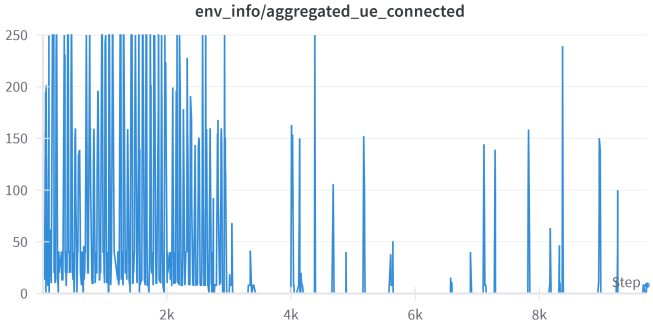
Analyzing the results



Crash count:

- Measured by the pipeline
- Detect when something is wrong with the environment during the training (usually a crash, or an issue when starting a new instance)
- Initiate a reset

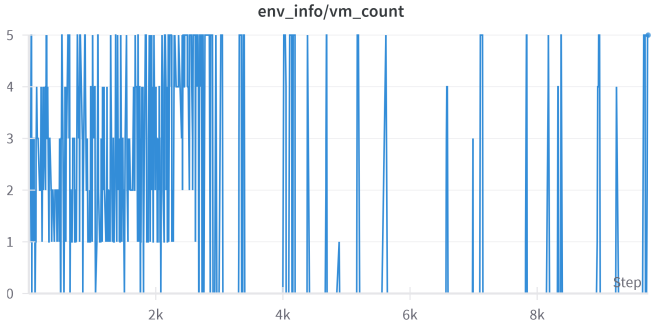
Analyzing the results



UEs connected:

- Metric collected from Magma NMS
- Number of UEs connected on all the AGWs currently running
- Different from Normalized UEs connected metric

Analyzing the results



VM count:

- Metric collected from Magma NMS
- Measure the number of active instances of AGW(s)

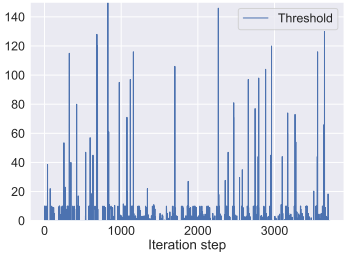
Reward function

Metric	Definition
U	Number of UEs connected
M	Memory usage, in MB
D	Number of dropped sessions

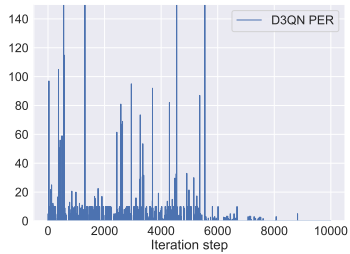
- Maximum reward value is 1
- Encourages the Network Functions (NFs) to use resources optimally around 70%
- Resource usage above 80% or crashes are penalized with lowest reward value -1.

$$r = \begin{cases} 1 - (0.7 - \max(M, U) - D) & \text{if M, U} \in [0, 80] \\ \max(-\max(M, U) - 10 * D, -1) & \text{otherwise.} \end{cases}$$

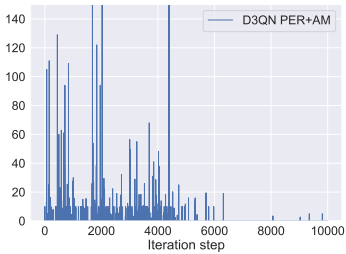
Number of sessions dropped



(a) threshold-based scenario



(b) D3QN PER scenario



(c) D3QN PER+AM scenario

Evolution of the number D of dropped sessions, during the three experiments

Average values of metrics during the experiments

Experiment	Steps	DUR (h)	Metric (average)				
			U	M	P	C	D
buffer filling + ϵ decay + pure exploitation							
D3QN PER	10k	154	345	78	5.76	3.25	0.60
D3QN PER+AM	10k	187	368	59	4.81	3.96	0.49
ϵ decay + pure exploitation							
D3QN PER	6k	63	382	61	4.48	3.57	0.25
D3QN PER+AM	6k	76	408	41	3.11	4.63	0.12
pure exploitation							
D3QN PER	4k	29	407	51	3.65	3.80	0.03
D3QN PER+AM	4k	34	420	33	2.60	4.84	0.01
Threshold-based	4k	145	247	119	9.72	2.16	1.73

Lessons learned

Reinforcement Learning (RL) is complex:

- Not your typical network or software engineer skill
- Very active area of research

Lessons learned

RL is complex:

- Not your typical network or software engineer skill
- Very active area of research

Automation of Network Core scaling:

- Development is required
- Probably better to use Infrastructure as Code framework to be IaaS-independent

Lessons learned

RL is complex:

- Not your typical network or software engineer skill
- Very active area of research

Automation of Network Core scaling:

- Development is required
- Probably better to use Infrastructure as Code framework to be IaaS-independent

Load generation is always tricky:

- Commercial products exist
- Lack of open source tools to generate traffic in a consistent way for a long time
- Traffic needs to be balanced across all instances

Q & A

- Any questions left ?