ITU Events

AI for Good

Machine Learning in 5G Challenge

Radio-strike: A reinforcement learning game for MIMO beam selection in unreal engine 3-D environments

Friday, 02 July 2021

16:00 - 17:00 Geneva (CEST)

aiforgood.itu.int

38 UN PARTNERS

ITU Artificial Intelligence/Machine Learning in 5G Challenge

Radio-Strike: A Reinforcement Learning Game for MIMO Beam Selection in Unreal Engine 3-D Environments

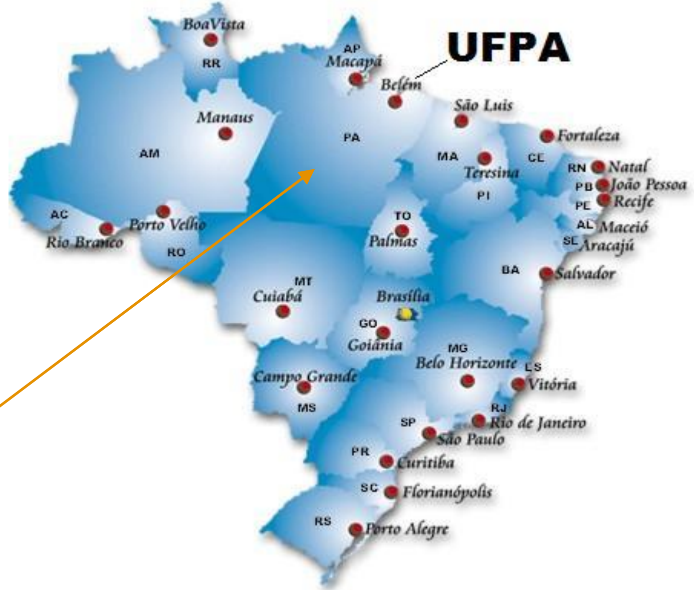
Aldebaro Klautau
Federal University of Pará (UFPA) / LASSE
http://ai5gchallenge.ufpa.br

Joint work with Prof. Francisco Müller (UFPA)
and several students

July 02, 2021

UFPA
Federal University of Pará

- Established in 1957
- Largest academic and research institution in the Amazon (Pará state in Northern Brazil)
- One of the largest Brazilian universities with total population (students + staff) of ~60k people
- One of the missions is the sustainable development of the region through science and technology



Agenda

- Motivation
- Beam selection
- Radio Strike
  - Reinforcement learning concepts (brief)
  - Problem ITU-ML5G-PS-006 reinforcement learning

Part I - Motivation

Machine learning for communications: importance to industry

Standardization bodies discussing AI / ML

ITU

- Architectural Framework for ML Rec. Y.3172
- Network automation and resource adaptation Rec. Y.3177

3GPP

- Network Data Analytics Function (NWDAF) TR 23.791
- Analytics in 5G Core TS 23.288

ETSI

- Experiential Networked Intelligence (ENI)
- Zero-Touch Network and Service Management (ZSM)

Linux Foundation

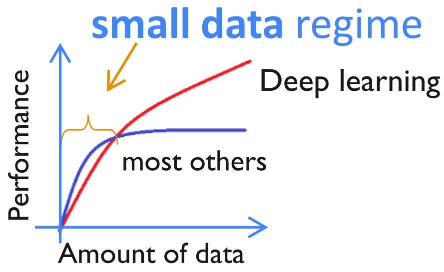
- AI in Open Network Platform (ONAP)
- ML and open data platforms

O-RAN

- RAN Intelligent con troller (RIC) and Near-RT RIC
- Data-driven workflows for closed-control loops

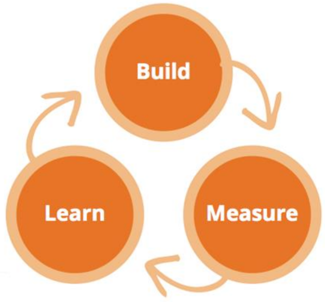
Machine learning for communications (ML4COMM) still faces the small data regime

Models are relatively small and ML4COMM has yet to escape small data regime



For instance, reinforcement learning (RL) agents applied to communications typically have a small action space dimension

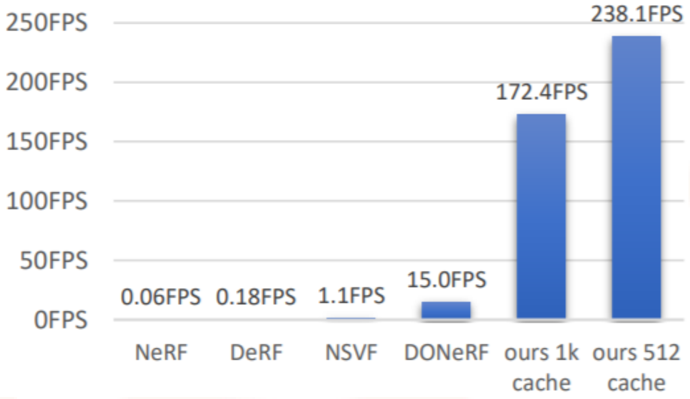
Data scarcity is an issue. Problem of traditional cycle: high cost of measurements when using high frequencies and multiple antennas



Key alternative for speeding up ML4COMM: use simulations to generate large datasets

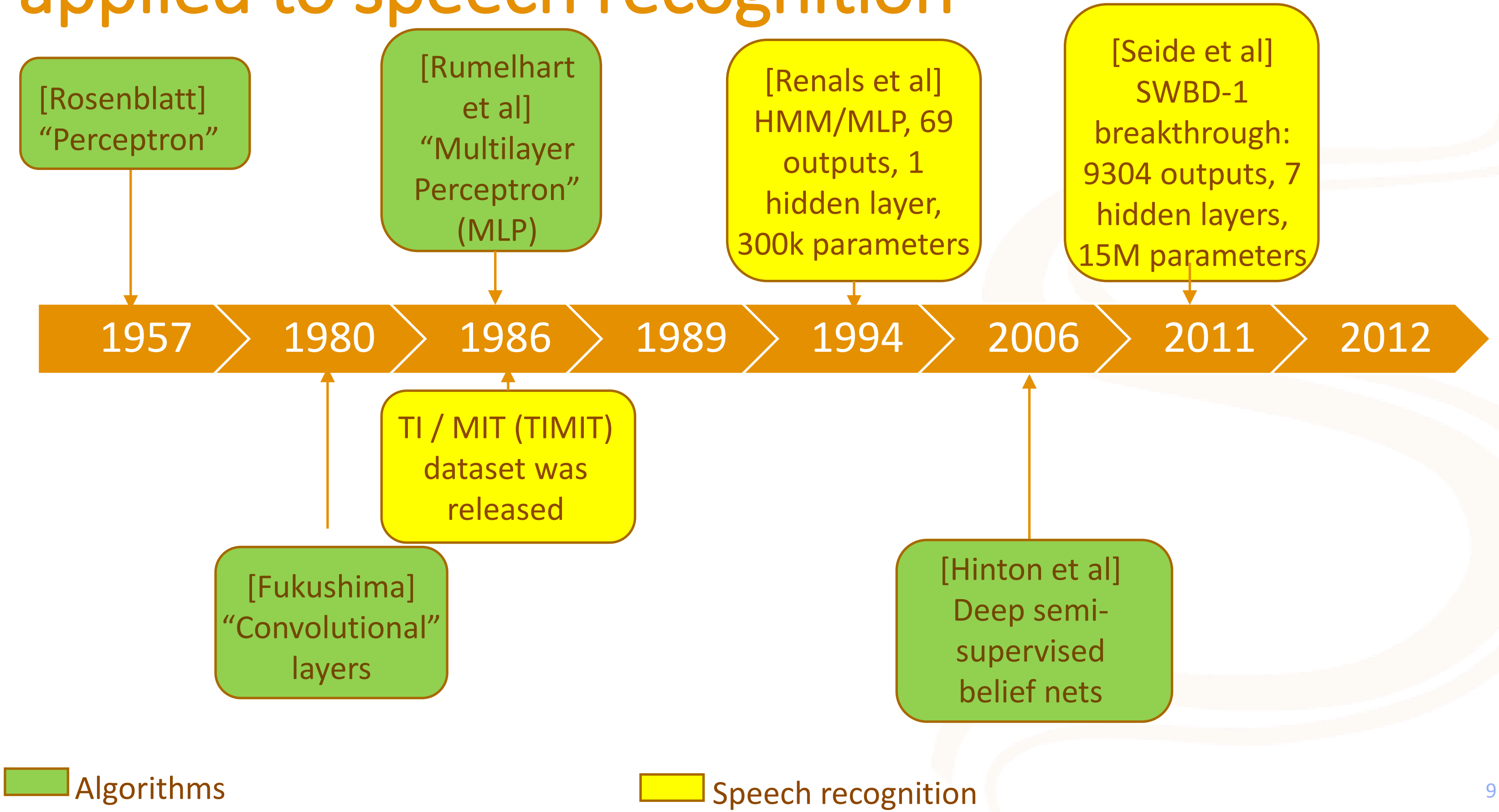
https://www.itu.int/en/ITU-T/academia/kaleidoscope/2021/Pages/default.aspx

Virtual Reality in Real Time: **FastNeRF** accelerates photorealistic 3D rendering via Neural Radiance Fields (NeRF) to visualize scenes at **200 frames per second** https://arxiv.org/abs/2103.10380v2

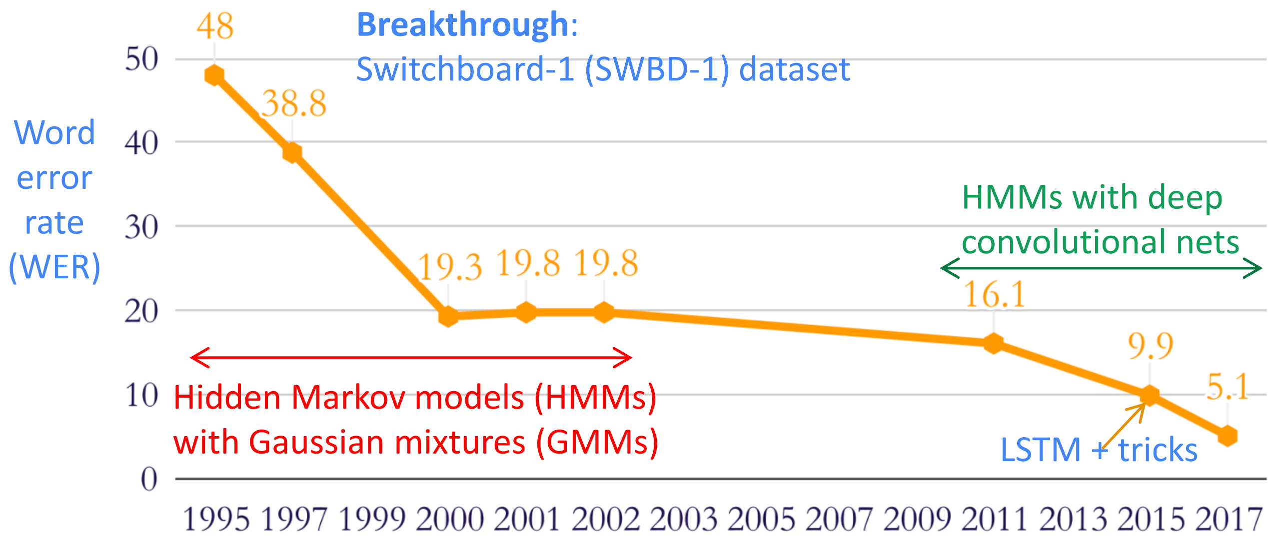


We will run simulations much faster than real-time

Historical evolution of neural networks applied to speech recognition



From (detailed) TIMIT to large (SWBD) datasets



When speech recognition reached the large data regime

- TIMIT dataset has detailed time-aligned orthographic and phonetic transcriptions
- In 1986, took 100 to 1000 hours of work to transcribe each hour of speech
- Project cost over 1 million dollars
- Five phoneticians agreed on 75% to 80% of cases

Simulating communication systems + AI + VR / AR

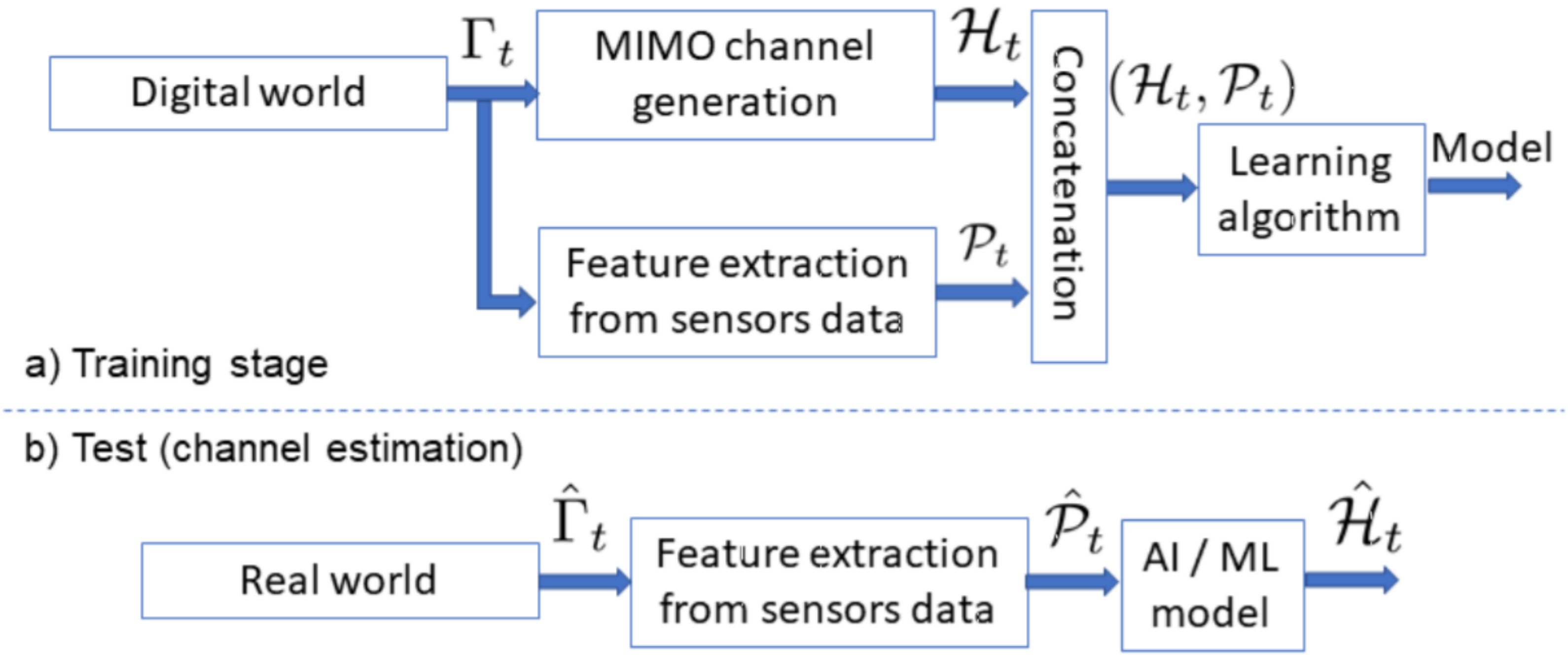
6G systems are expected to support applications such as augmented reality, multisensory communications and high fidelity holograms. This information will flow through the network. It is expected that 6G systems will use ML/AI to leverage such multimodal data and optimize performance

This requires a simulation environment that is capable not only of generating communication channels, but also the corresponding sensor data, matched to the scene

ITU-ML5G-PS-006-RL: Communication networks and Artificial intelligence immersed in VIrtual or Augmented Reality (CAVIAR)

Generating MIMO Channels For 6G Virtual Worlds Using Ray-tracing Simulations https://arxiv.org/pdf/2106.05377.pdf

CAVIAR: get “measurements” on virtual worlds



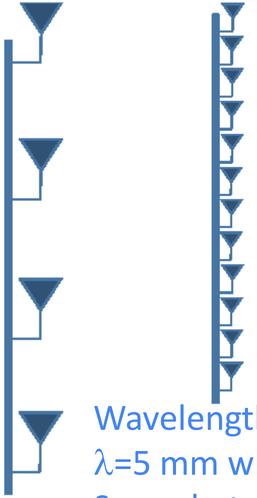
Generating MIMO Channels For 6G Virtual Worlds Using Ray-tracing Simulations https://arxiv.org/pdf/2106.05377.pdf

Part II – Beam selection

Improving communications with antenna arrays

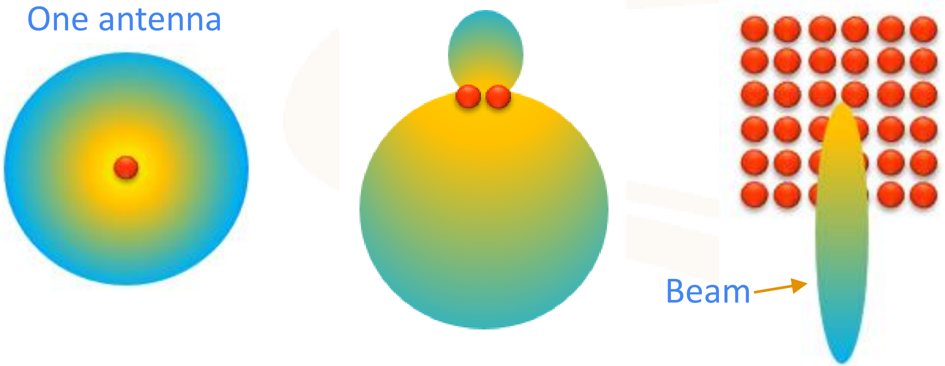
Array form factor decreases when frequency increases (mmWave in 5G / THz in 6G)

mmWave THz

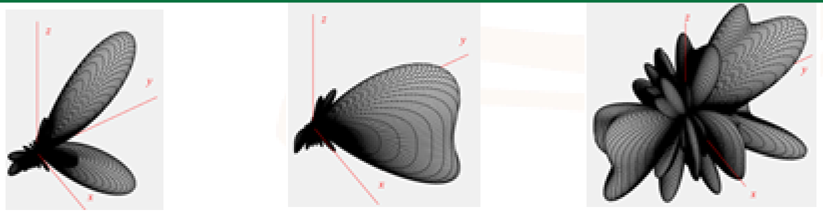


Wavelength λ=c/f
λ=5 mm when f=60 GHz
Space between antenna elements = $\lambda/2$

Illustrative radiation patterns of an array:



Given a phased antenna array, we choose a “beamvector” to impose a radiation pattern



Beam selection in 5G mmWave

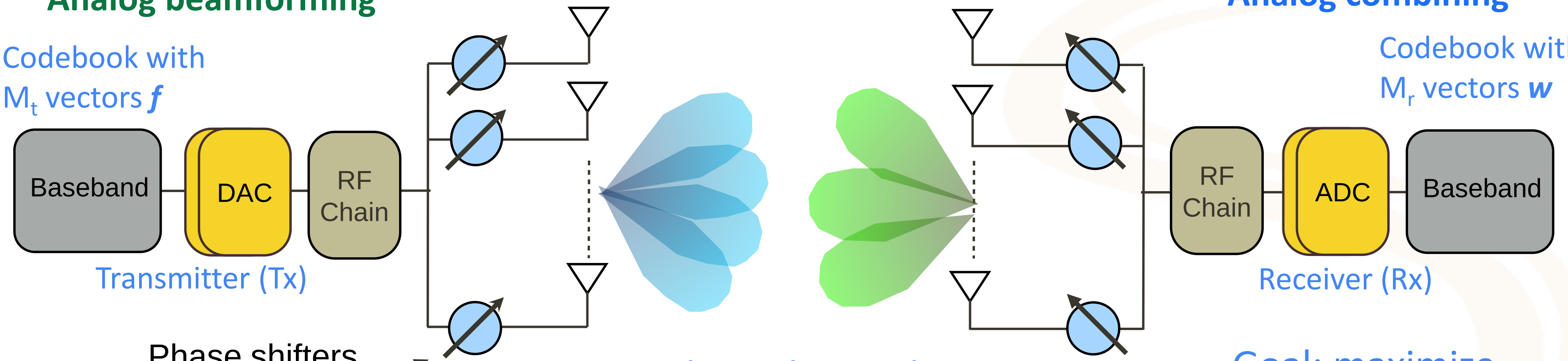
Align the beams of transmitter and receiver

Analog beamforming

Codebook with
M_t vectors $\boldsymbol{f}$

Analog combining

Codebook with
M_r vectors $\boldsymbol{w}$



Transmitter (Tx)

Receiver (Rx)

Phase shifters specified by codebooks

Wireless channel $\boldsymbol{H}$

Goal: maximize

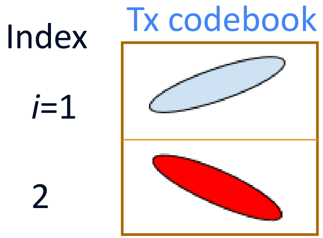
$$y(i,j) = |\boldsymbol{w}_j^H \boldsymbol{H} \boldsymbol{f}_i|$$

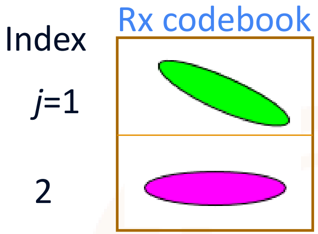
Brute force to find best: try all possible M_t x M_r pairs of indices

[1] Heath et al, An Overview of Signal Processing Techniques for Millimeter Wave MIMO Systems, 2016

ML-based beam selection in 5G mmWave: often modeled as supervised learning

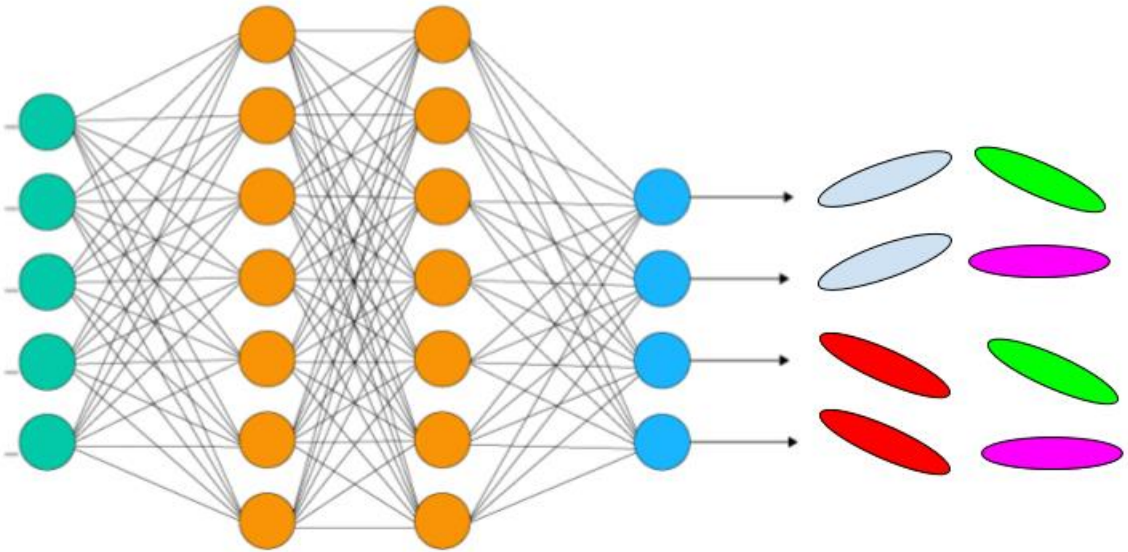
Example with two beamvectors per codebook





Typically posed as a classification problem. We will assume RL

Inputs from communication system and also from sensors such as GPS

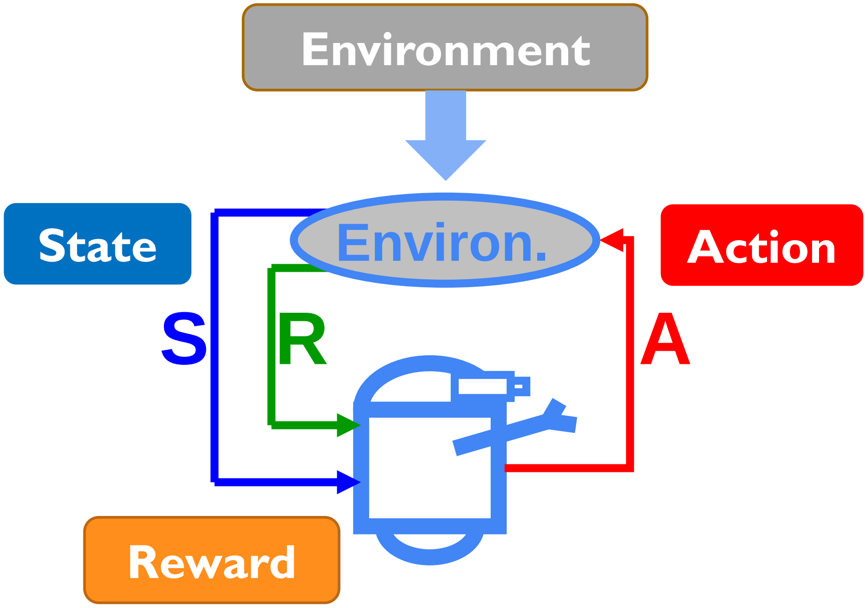


Pair or single index	
(1, 1)	0
(1, 2)	1
(2,1)	2
(2,2)	3

Part III – Radio Strike

A Reinforcement Learning Game for MIMO Beam Selection in Unreal Engine 3-D Environments

Reinforcement learning with OpenAI Gym



Goal: Find a policy that maximizes the return over a lifetime (episode, if not a continuing task)

We adopt the popular OpenAI Gym API

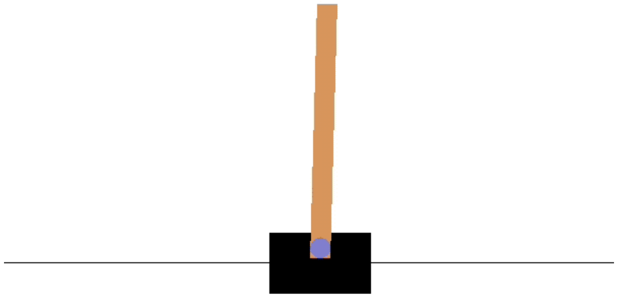
https://gym.openai.com/

- make(): Used to create environment.
- reset(): Setting the environment to default starting stage.
- render(): It creates a popup window to display Simulation of Agent interacting with environment
- step(): Action taken by the agent. it returns an observation. (4 valued numpy array, <observations, reward, done, info>)

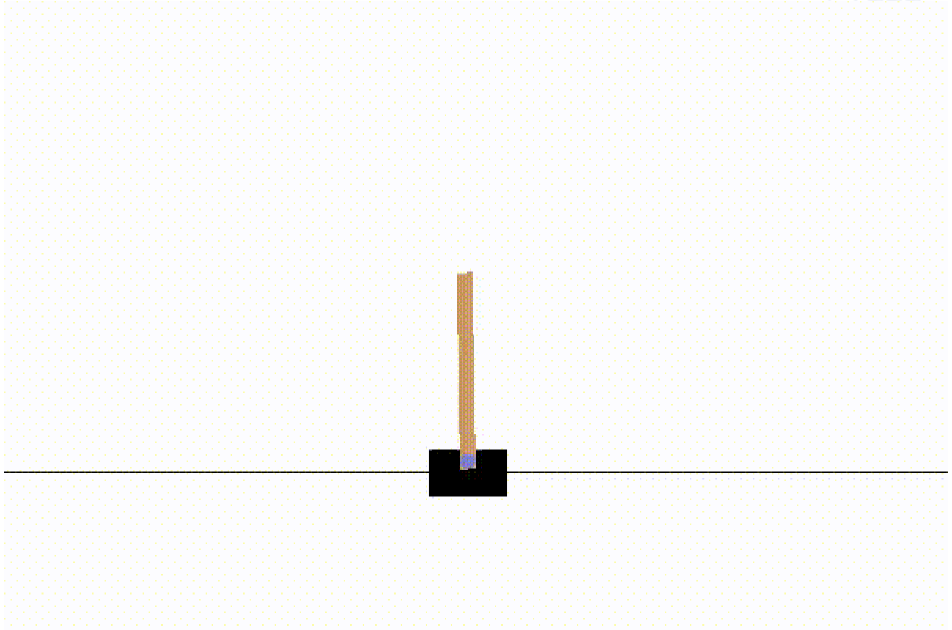
https://www.slideshare.net/a4aleem/reinforcement-learning-using-openai-gym

```
import gym
env = gym.make('CartPole-v0')
env.reset()
for _ in range(1000):
    env.render()
    env.step(env.action_space.sample())
```

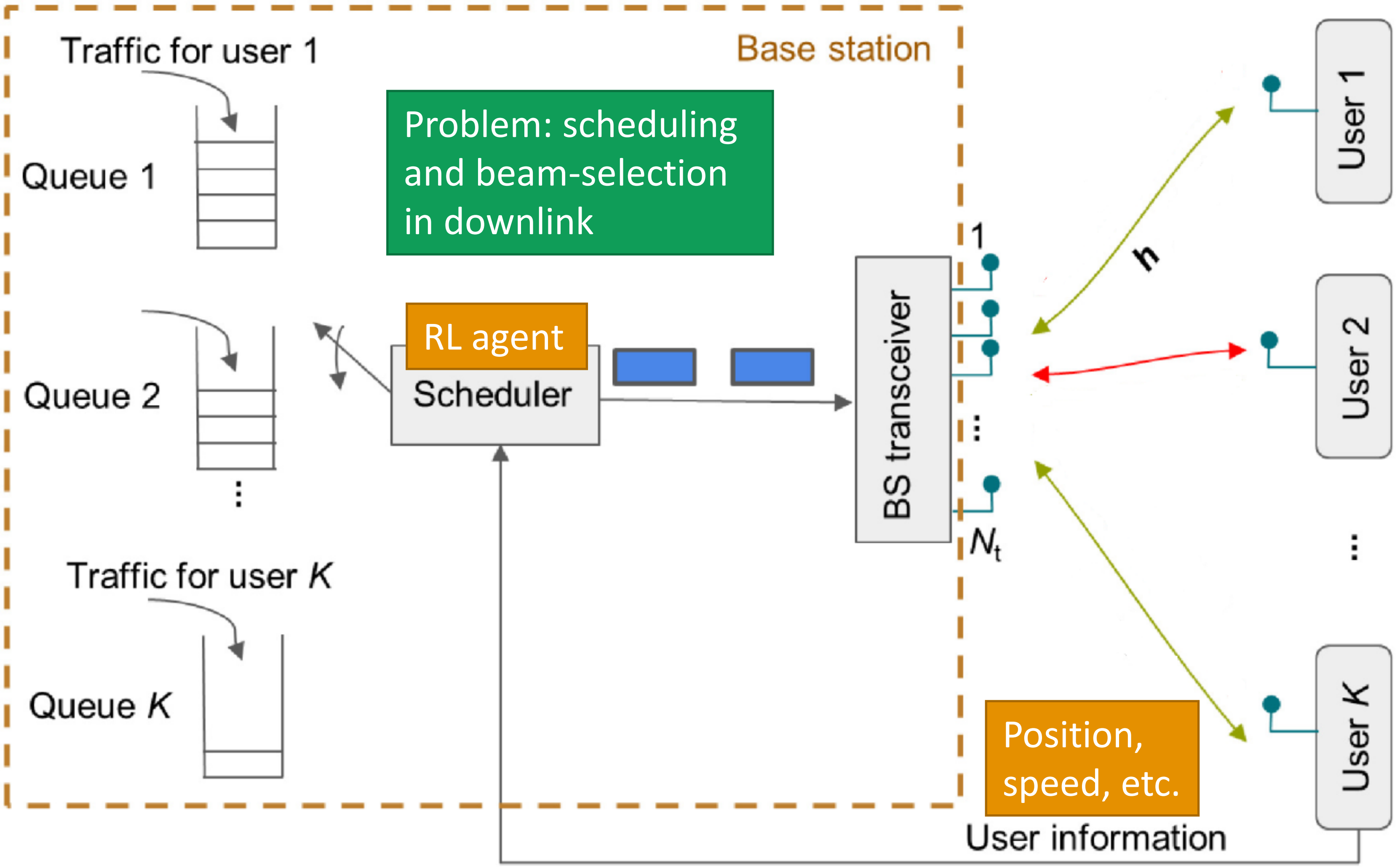
Using random actions:



After training the RL agent:



Similarly, we want to choose the beam and maximize performance with respect to throughput and packet loss

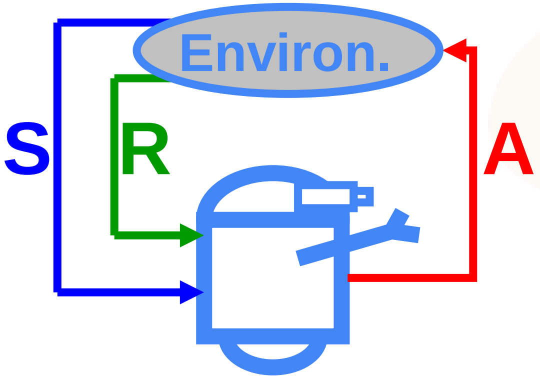


Traffic for user 1
Base station
Problem: scheduling and beam-selection in downlink
Queue 1
RL agent
Scheduler
Queue 2
BS transceiver
1
N_t
h
User 1
User 2
User K
Traffic for user K
Queue K
Position, speed, etc.
User information

Reinforcement learning for beam selection: RadioStrike-noRT-v1

The RL agent is executed at a base station (BS) with an antenna array and serves single-antenna users on downlink using an analog MIMO architecture: pedestrian, drone and car

State (or **observation**): position and buffer status of each user, previously scheduled users, etc.



Action: at each time slot the agent action schedules one user and chooses the beam index to serve this user

The state is defined by the participant, as well as eventual "intrinsic" rewards

Reward: normalized throughput with a penalty for dropped packets

Return (in the end of the episode): sum of rewards

ITU-ML5G-PS-006: research questions and strategies

Some questions:

- When performing user scheduling and beam selection, does position information help the scheduler?
- Can we benefit from knowing the positions of scatterers?

From experience with 2020 Challenge:

- Help participants with the (eventually steep) learning curve
- Besides the main problem, discuss related simpler tasks and provide support

Keep evolving:

- Build together increasingly difficult CAVIAR “games”
- Create benchmarks for realistic applications of RL in 5G and 6G

Strategy 1: Provide guidance with the setup

Several specialized tools, besides the ones for reinforcement learning

python | PyTorch | TensorFlow & Keras

Most used language

Facebook's

Google's, TF versions 1 and 2, with high level Keras API

Deployment frameworks: facilitate pruning the models and quantizing the weights for acceleration

Qualcomm's AI Model Efficiency Toolkit

AIMET

Tensorflow Lite
& PyTorch Quantization

Other tools: NVIDIA, Intel, etc.

Auxiliary tools for (shallow) machine learning, debugging, assessing models and running on cloud

Scikit learn | jupyter

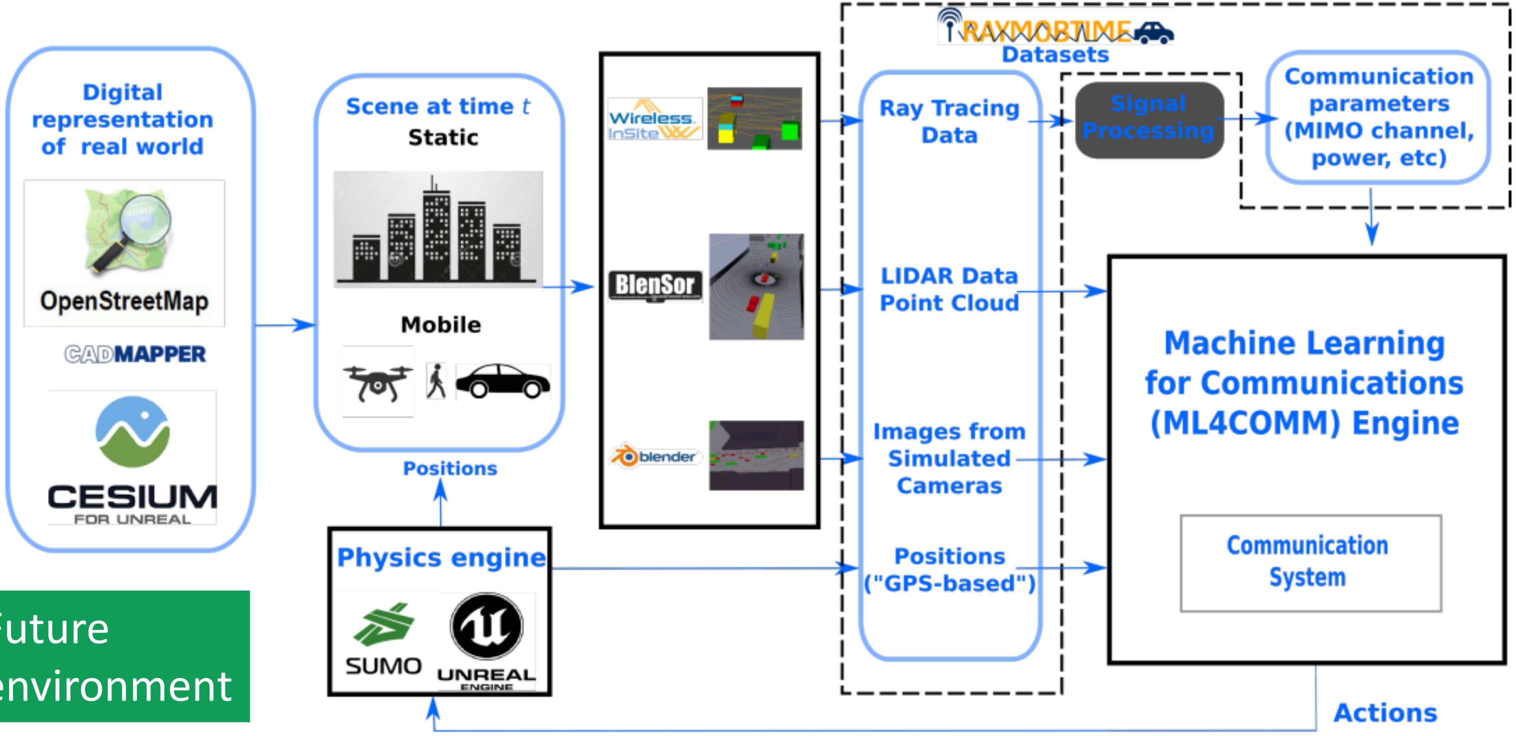
It may not be trivial to set up your development workflow

Strategy 2: Share simple baseline code

```
#load the trained agent and test it
trained_model = DQN.load("beam_selection.dqn")
env.enable_rendering() #allow visualizing
obs = env.reset() #reset environment
for i in range(10):
    action, _states = trained_model.predict(obs)
    obs, reward, dones, info = env.step(action)
```

```python
#use DQN
dqn_agent = DQN(policy="MlpPolicy",
                batch_size=10,
                gamma=0.9,
                verbose=1,
                exploration_fraction=0.9,
                learning_rate=0.01,
                buffer_size=1500,
                exploration_final_eps=0.02,
                exploration_initial_eps=1.0,
                learning_starts=100,
                env=env,
                tensorboard_log="./log_tensorboard/",
                seed=0)
#train the agent
dqn_agent.learn(total_timesteps=total_timesteps)
```

Strategy 3: Postpone using ray-tracing and adopt simple MIMO channel estimation

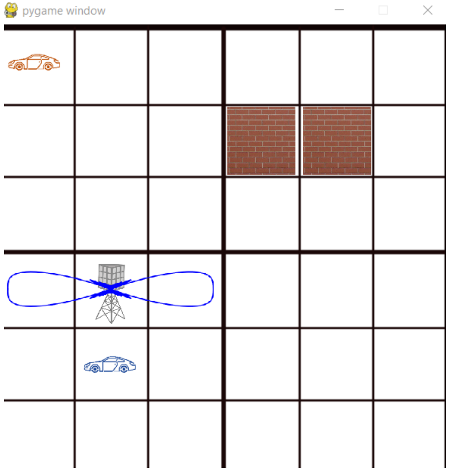


Future environment

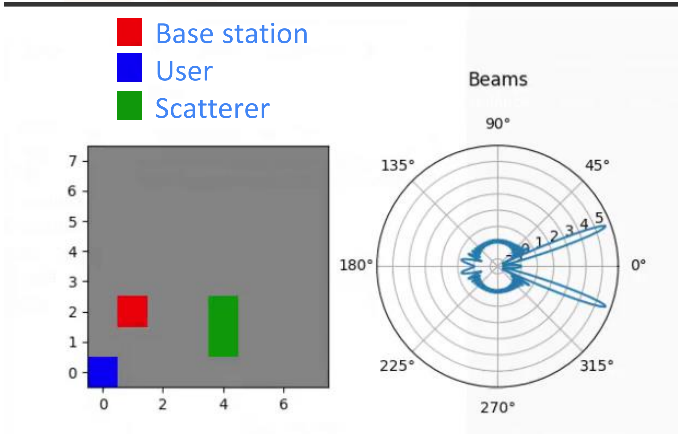
Strategy 4: provide support to two beam selection environments

RadioStrike-noRT-v1
(PS-006 ITU Challenge)

MimoRL-simple-1-v0
(easier to start with)



Both have the basic elements:



ITU-ML5G-PS-006-RL: challenge, learning environment and framework for building future CAVIAR simulations

Concepts of tabular reinforcement learning

OpenAI gym environment
MimoRL-simple-1-v0


```
Nu = 2 #users
Nb = 64 #beam indices
M =  6 #grid size
Na = 3 #allocation timeout
num_actions=Nu*Nb    = 128
num_states=M^(2*Nu)*Nu^(Na-1)  = 5184
```

Strategy to get a policy: find the “value” of a state/action pair, its long-term return

Action

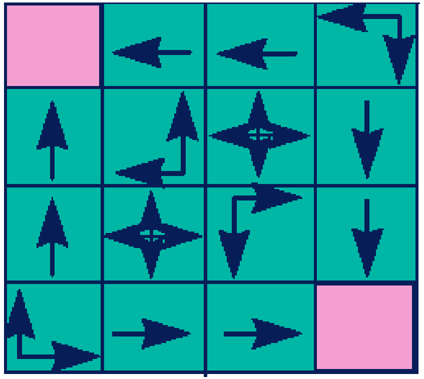
$$
\begin{array}{c c}
\text{State} & \begin{array}{cccc} 1 & 2 & \dots & 128 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ \vdots \\ 5184 \end{array} & \begin{bmatrix} 0 & 100 & & 0 \\ 0 & 0 & & 64 \\ 0 & 80 & & 0 \\ 64 & 0 & & 100 \\ & & & \\ 0 & 80 & & 100 \end{bmatrix}
\end{array}
$$

Q-table

Q(s,a) values

Multi-armed bandits (**MAB**) are simpler RL in which the action influences the reward but not the “state”

Easier to visualize: grid-world example (reach a pink corner)



Policy: what to do. Maps states in actions

[1] Sutton’s & Barto’s book. Reinforcement learning: an introduction.

Policy versus Q-value in simpler 4 x 4 grid

Goal is to reach one of the pink corners

Rewards

0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	0

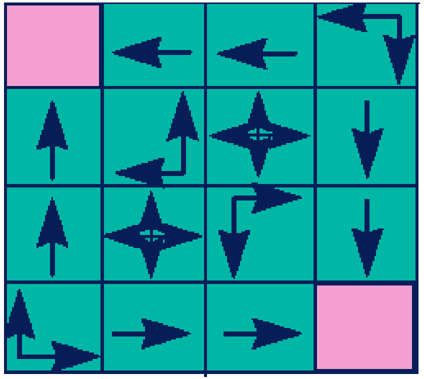
The reward is -1 everywhere

Q-values for optimal policy

0	-1	-2	-3
-1	-2	-3	-2
-2	-3	-2	-1
-3	-2	-1	0

The Q-value is the long-term expected return, not the immediate reward

Optimal policy



Policy can be based on the Q(s,a) table. Learn the table first.

[1] Sutton’s & Barto’s book. Reinforcement learning: an introduction (Example 4.1)
[2] https://github.com/ShangtongZhang/reinforcement-learning-an-introduction/blob/master/chapter04/grid_world.py

DQN: From tabular method to deep RL

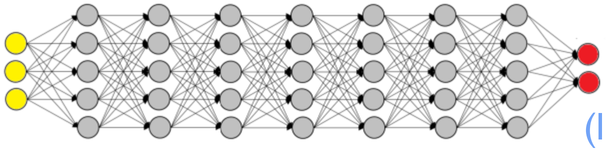
Q table: expected long-term return

Table can become too large! ☹

State \ Action	1	2	...	128
1	0	100		0
2	0	0		64
3	0	80		0
4	64	0		100
⋮				
5184	0	80		100

Then use a neural network [1]

Input: **state**



Q-value estimates (linear activation)

Another advantage of a NN instead of a table: the state space (input) can be continuous (real numbers)

Online learning, no need for output labels. Support to delayed reward

Find the balance between explore and exploit

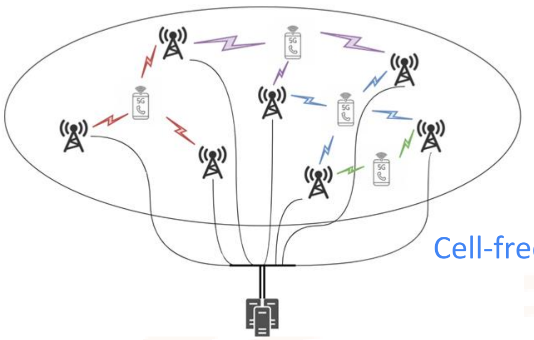
Environment:
- Probabilistic / deterministic
- Stationary / non-stationary
- Full / partial state observability

Need reward engineering

[1] Mnih et al, Playing Atari with Deep Reinforcement Learning, 2013

Another class of algorithms: Policy Gradient

Example: an RL agent that allocates power (as real numbers) in cell-free MIMO requires a continuous action space



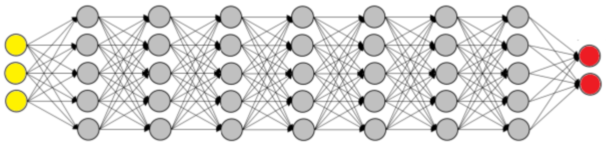
Cell-free MIMO

Policy gradient methods: the NN output is a policy, not Q-value estimates. Supports stochastic policies.

State (input) and action spaces (output) can be continuous (real numbers)

Discrete action example

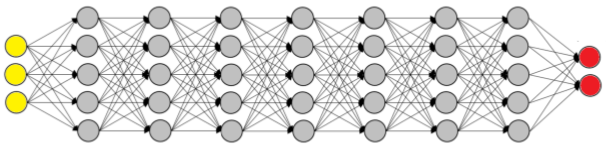
Input: **state**



Softmax activation: distribution over **actions**

Continuous action example

Input: **state**



Activations for Gaussian means and variances

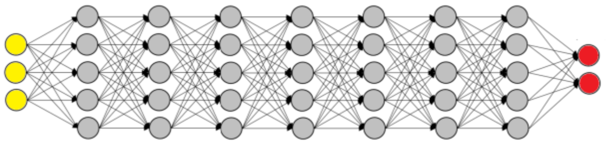
Summary of RL Methods

Tabular methods (no neural network):

$$
\begin{array}{c|c}
 & \text{Action} \\
\text{State} & \begin{array}{cccc} 1 & 2 & \dots & 128 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ \vdots \\ 5184 \end{array} & \begin{bmatrix} 0 & 100 & & 0 \\ 0 & 0 & & 64 \\ 0 & 80 & & 0 \\ 64 & 0 & & 100 \\ & & & \\ 0 & 80 & & 100 \end{bmatrix}
\end{array}
$$

Deep Q-network (DQN):

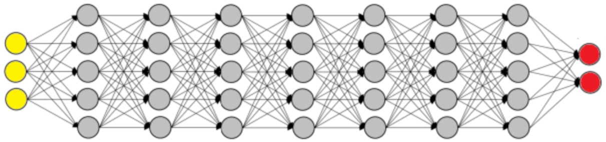
Input: **state**



Q-values estimates

Policy gradient methods:

Input: **state**



Distribution or means/variances of **actions**

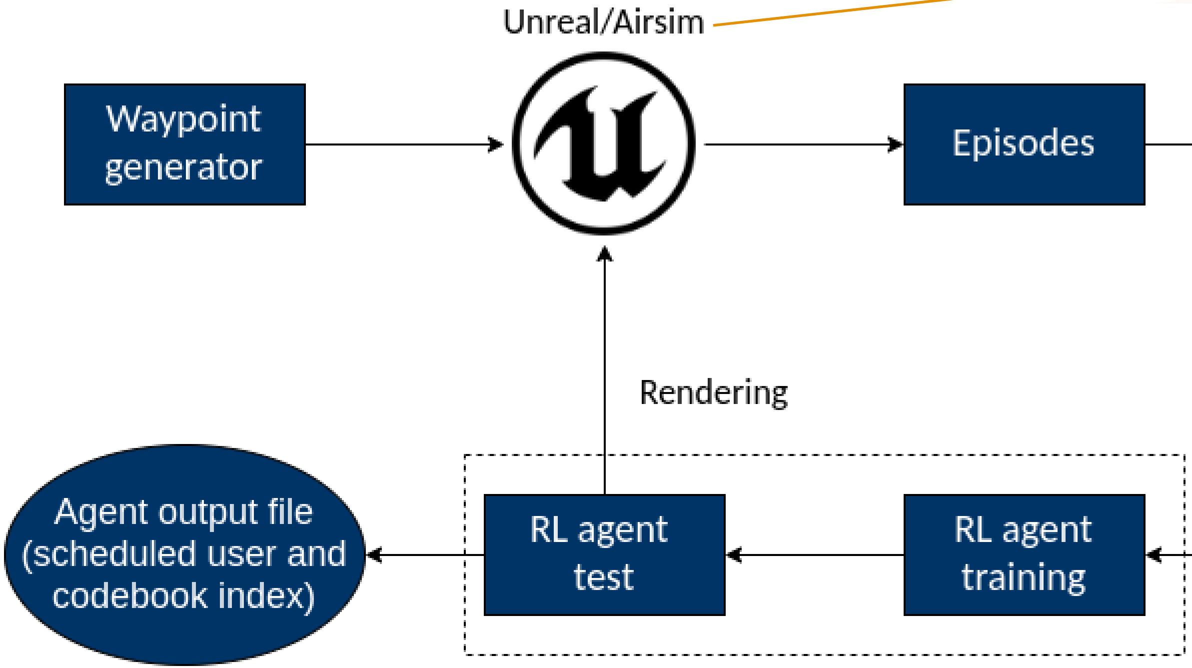
Actor-critic (e.g. A3C): uses 2 NNs, Critic estimates Q-values and Actor the policy

In all NN-based cases: # outputs neurons = # actions. PS-006 has a small # actions

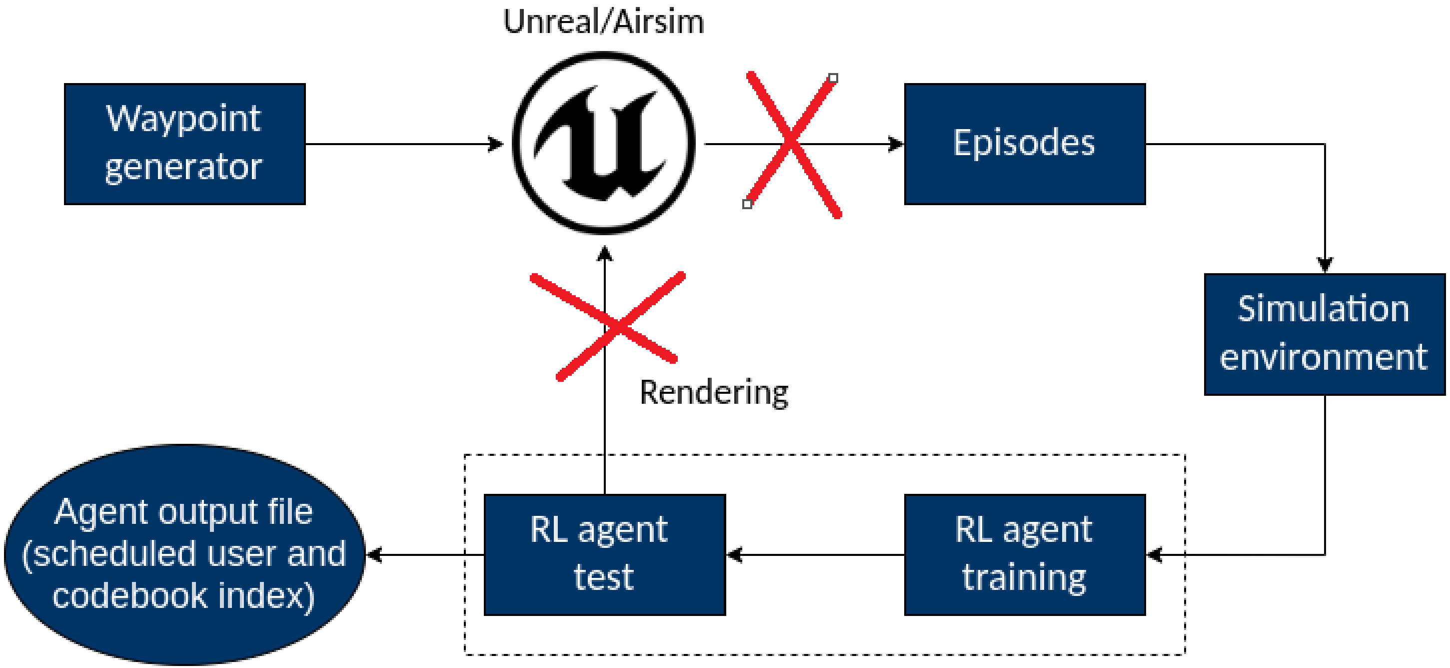
How is the ITU-ML5G-PS-006-RL simulation performed?

Base station serving a drone

Simulation block diagram



If one wants to avoid executing Unreal/Airsim



ITU-ML5G-PS-006-RL code and associated files

- https://github.com/lasseufpa/ITU-Challenge-ML5G-PHY-RL

README.md

Radio Strike

Details about the challenge and the datasets are available at http://ai5gchallenge.ufpa.br.

Registration link: https://challenge.aiforgood.itu.int/match/matchitem/39

Datasets are available at https://nextcloud.lasseufpa.org/s/WYZAMbSbdocs2DL

Radio Strike Installation

This instructions will guide you through the process of using Radio Strike.

Setting up the environment to run the baseline codes

The provided RL agent uses Stable-Baselines as its framework. Because of that, in our environment we need Python 3.6 and Tensorflow 1.14.

Steps to prepare the environment and run the baseline

```
Radio_Strike
├── baseline
│   ├── agent_output
│   ├── episodes_ues
│   └── results
├── caviar_episodes_generator
│   ├── way_points
│   │   ├── preliminary
│   │   │   └── run1
│   │   ├── test
│   │   └── training
│   ├── windows_runtime
│   └── linux_runtime
└── episodes_datasets
    ├── preliminary
    │   ├── run0
    │   └── run1
    ├── test
    └── training
```

Install package manager, e.g., Conda

Create environment

We used Stables-Baselines 2.10 for our RL agent, so we needed Tensorflow 1.14 and Python 3.6

Activate the environment and install the packages

Run RL agent train/test

Train: $ python3 train_agent.py ‘agent_name’ ‘train_episode’
*outputs: ./model/’*agent_name’*.a2c*
Test: $ python3 test_agent.py ‘agent_name’ ‘test_episode’
*outputs: ./data/actions_’*agent_name’*.csv*

Data organization

The dataset is provided in `.csv` files, that are in the folder `episodes` (i.e `ep0.csv`, `ep1.csv` etc). Each `episode` has approximately 3 minutes of duration, with information stored with a sampling interval of 10 ms. The csv is composed by the following columns:

timestamp	obj	pos_x	pos_y	pos_z	orien_w	orien_x

orien_y	orien_z	linear_acc_x	linear_acc_y	linear_acc_z	linear_vel_x	linear_vel_y

linear_vel_z	angular_acc_x	angular_acc_y	angular_acc_z	angular_vel_x	angular_vel_y	angular_vel_z

There are three different types of objects: `uav`, `simulation_car` and `simulation_pedestrian`. Only the `uav` type has information in all columns, while the others have only information regarding their position and orientation.

CSV text file corresponding to an episode:

timestamp	obj	pos_x	pos_y	pos_z	orien_w	orien_x	orien_y	orien_z	linear_acc_x	linear_acc_y
1,62508018033396E+018	uav1	-0.20948558	0.24244854	-1.9929655	-0.0021526131	0.004117747	0.9999892	-6.1888495e-05	0.04079342	-0.07799979
1,62508018033396E+018	simulation_car1	47.956226	16.016552	8.409805	0.00095201726	0.00078579056	-0.7081485	0.7060625		
1,62508018033396E+018	simulation_car2	-2.2606368	-1.942027	8.409806	0.001008632	0.0007415256	-0.70921654	0.7049897		
1,62508018033396E+018	simulation_pedestrian1	2.6764462	34.1297	7.4372497	0.0	-0.0	-0.7726407	0.6348436		

Episode example (complete information about the scene)

Sampling interval T_s = 10 milliseconds

Average episode duration = 3 minutes

timestamp	obj	pos_x	pos_y	pos_z	orien_w	orien_x	orien_y	orien_z	linear_acc_x	linear_acc_y
1,62508018033396E+018	uav1	-0.20948558	0.24244854	-1.9929655	-0.0021526131	0.004117747	0.9999892	-6.1888495e-05	0.04079342	-0.07799979
1,62508018033396E+018	simulation_car1	47.956226	16.016552	8.409805	0.00095201726	0.00078579056	-0.7081485	0.7060625		
1,62508018033396E+018	simulation_car2	-2.2606368	-1.942027	8.409806	0.001008632	0.0007415256	-0.70921654	0.7049897		
1,62508018033396E+018	simulation_pedestrian1	2.6764462	34.1297	7.4372497	0.0	-0.0	-0.7726407	0.6348436		
1,62508018033396E+018	simulation_pedestrian2	2.8446472	28.789553	7.4372497	0.0	-0.0	-0.66428643	0.7474781		
1,62508018033396E+018	simulation_pedestrian3	35.294273	35.42358	7.4372497	-0.0	0.0	0.9032605	0.4290926		
1,62508018033396E+018	simulation_pedestrian4	4.173308	35.695763	7.43725	0.0	-0.0	-0.93007565	0.36736804		
1,62508018033396E+018	simulation_pedestrian5	34.344566	37.21747	7.4372497	-0.0	0.0	0.0003657082	0.99999994		
1,62508018033396E+018	simulation_pedestrian6	2.2895343	35.48529	7.43725	0.0	-0.0	-0.66687083	0.74517334		
1,62508018033396E+018	simulation_pedestrian7	38.16299	-16.524492	7.4372497	0.0	-0.0	-0.7460329	0.66590905		
1,62508018033396E+018	simulation_pedestrian8	38.262566	17.056343	7.4372497	-0.0	0.0	0.6159631	0.787775		
1,62508018033396E+018	simulation_pedestrian9	28.910654	-17.344934	7.4372497	0.0	-0.0	-0.10489227	0.9944836		
1,62508018033396E+018	simulation_pedestrian10	38.476254	20.479986	7.4372497	-0.0	0.0	0.71488017	0.69924694		
1,62508018033396E+018	simulation_pedestrian11	33.541344	-17.368237	7.4372497	-0.0	0.0	0.99640214	0.08475071		
1,62508018033396E+018	simulation_pedestrian12	30.552599	-17.409073	7.43725	0.0	-0.0	-0.78051025	0.62514293		
1,62508018033396E+018	simulation_pedestrian13	3.6040344	-16.202654	7.4372497	-0.0	0.0	0.94671106	0.32208395		

This information does not depend on the RL agent actions and can be pre-computed. The buffer status can be used as input to the agent but need to be retrieved along the execution

Input data for baseline RL agent


```
timestamp,obj,pos_x,pos_y,pos_z,orien_w,orien_x,orien_y,orien_z,linear_acc_x,linear_acc_y,linear_acc_z,linear_vel_x,linear_vel_y,linear_vel_z,angular_acc_x,angular_acc_y,angular_acc_z,angular_vel_x,angular_vel_y,angular_vel_z
1625117771599573760,uav1,0.012286402,-6.5288436e-06,-2.006913,-0.0023287118,8.9766866e-07,0.9999973,-0.00016763437,0.044039983,[illegible]5,-0.042386435,1.0644681e-05,0.019996691,2.2170343e-05,-0.048432052,-2.6114742e-0
1625117771599573760,simulation_car1,47.850044,-13.439995,8.407602,0.0009428796,-0.001517812,-0.71445984,0.69967407,,,,,,,,,,,,,,
1625117771599573760,simulation_car2,2.3160567,-22.379938,8.407614,-0.005811054,0.00060353[illegible]4,,,,,,,,,,,,,,
1625117771599573760,simulation_pedestrian1,3.6577013,34.36091,7.4372497,-0.0,0.0,0.47335017,0.88087434,,,,,,,,,,,,,,
1625117771599573760,simulation_pedestrian2,13.016938,37.16622,7.4372497,0.0,-0.0,-0.279284,0.96020854,,,,,,,,,,,,,,
1625117771599573760,simulation_pedestrian3,28.327856,36.918667,7.4372497,0.0,-0.0,-0.9999878,0.004927337,,,,,,,,,,,,,,
1625117771599573760,simulation_pedestrian4,19.597334,37.23965,7.4372497,0.0,-0.0,-0.9998059,0.01969724[illegible]
1625117771599573760,simulation_pedestrian5,2.1828759,32.092052,7.4372497,-0.0,0.0,0.14223726,0.9898326,,,,,,,,,,,,,,
1625117771599573760,simulation_pedestrian6,22.417881,37.31741,7.4372497,-0.0,0.0,0.9952836,0.09700763,,,,,,,,,,,,,,
1625117771599573760,simulation_pedestrian7,37.495476,-14.887019,7.4372497,-0.0,0.0,0.30359143,0.9528023,,,,,,,,,,,,,,
1625117771599573760,simulation_pedestrian8,38.707874,-7.723372,7.4372497,-0.0,0.0,0.785845,0.6184235,,,,,,,,,,,,,,
1625117771599573760,simulation_pedestrian9,23.485868,-18.008545,7.43725,-0.0,0.0,0.002099219,0.9999978,,,,,,,,,,,,,,
1625117771599573760,simulation_pedestrian10,38.02125,30.30045,7.4372497,0.0,-0.0,-0.590002,0.8074018,,,,,,,,,,,,,,
1625117771599573760,simulation_pedestrian11,38.41673,-9.160543,7.4372497,0.0,-0.0,-0.5048172,0.8632263,,,,,,,,,,,,,,
1625117771599573760,simulation_pedestrian12,37.56257,-10.855839,7.4372497,-0.0,0.0,0.6044158,0.796669,,,,,,,,,,,,,,
1625117771599573760,simulation_pedestrian13,1.793553,-8.137272,7.4372497,-0.0,0.0,0.68485576,0.7286787,,,,,,,,,,,,,,
1625117771599573760,simulation_pedestrian14,30.11543,-28.954742,7.4372497,-0.0,0.0,0.011400648,0.99993503,,,,,,,,,,,,,,
1625117771599573760,simulation_pedestrian15,37.86455,-12.334499,7.4372497,0.0,-0.0,-0.8385686,0.544796,,,,,,,,,,,,,,
1625117771599573760,simulation_pedestrian16,38.110893,-16.108389,7.4372497,0.0,-0.0,-0.7160644,0.69803417,,,,,,,,,,,,,,
1625117771599573760,simulation_pedestrian17,2.3861172,-6.251685,7.4372497,-0.0,0.0,0.73384047,0.6793218,,,,,,,,,,,,,,
1625117771599573760,simulation_pedestrian18,2.1361067,28.265514,7.4372497,0.0,-0.0,-0.6921356,0.7217675,,,,,,,,,,,,,,
1625117771599573760,simulation_pedestrian19,2.0327365,12.278926,7.4372497,0.0,-0.0,-0.7100595,0.7041417,,,,,,,,,,,,,,
1625117771599573760,simulation_pedestrian20,2.0315273,3.09649,7.4353766,-0.0,0.0,0.7119502,0.70223,,,,,,,,,,,,,,
1625117771599573760,simulation_pedestrian21,2.261779,-12.111201,7.4372497,-0.0,0.0,0.69738483,0.71669686,,,,,,,,,,,,,,
1625117771599573760,simulation_pedestrian22,4.2832155,35.67008,7.43725,-0.0,0.0,0.3871685,0.922009,,,,,,,,,,,,,,
1625117771599573760,simulation_pedestrian23,14.1571455,37.6583,7.43725,-0.0,0.0,0.97312003,0.23029834,,,,,,,,,,,,,,
1625117771599573760,simulation_pedestrian24,21.316984,-28.944372,7.4372497,0.0,-0.0,-0.002807501,0.99999607,,,,,,,,,,,,,,
1625117771599573760,simulation_pedestrian25,37.47301,-28.558,7.4372497,0.0,-0.0,-0.9999377,0.01116389,,,,,,,,,,,,,,
1625117771599573760,simulation_pedestrian26,-11.766973,24.149649,7.4372497,0.0,-0.0,-0.3977188,0.91750735,,,,,,,,,,,,,,
1625117771599573760,simulation_pedestrian27,-12.364105,22.656416,7.4372497,-0.0,0.0,0.9734946,0.22870982,,,,,,,,,,,,,,
1625117771599573760,simulation_pedestrian28,-12.063819,11.292378,7.4372497,-0.0,0.0,0.7231811,0.69065845,,,,,,,,,,,,,,
1625117771599573760,simulation_pedestrian29,-11.947682,-8.678772,7.4372497,0.0,-0.0,-0.6960221,0.7180204,,,,,,,,,,,,,,
1625117771599573760,simulation_pedestrian30,-12.097404,-4.439013,7.43725,0.0,-0.0,-0.69846815,0.71564114,,,,,,,,,,,,,,
1625117771599573760,simulation_pedestrian31,51.247967,-20.912868,7.4372497,0.0,-0.0,-0.98306173,0.18327498,,,,,,,,,,,,,,
1625117771599573760,simulation_pedestrian32,51.850796,-24.003613,7.4372497,-0.0,0.0,0.6467609,0.7626928,,,,,,,,,,,,,,
1625117771599573760,simulation_pedestrian33,51.132298,22.776785,7.4372497,-0.0,0.0,0.6923937,0.7215199,,,,,,,,,,,,,,
1625117771599573760,simulation_pedestrian34,51.16157,10.5506115,7.4372497,0.0,-0.0,-0.7118101,0.70237195,,,,,,,,,,,,,,
1625117771611574016,uav1,0.011781977,-6.4020255e-06,-2.0068734,-0.0023110989,8.9346116e-07,0.9999973,-0.00016730346,0.0734259,-1[illegible]5,-0.041620012,1.0477216e-05,-0.02689001,-0.00097642594,-1.4078839,-0.018643577,-
1625117771611574016,simulation_car1,47.84874,-13.479751,8.407543,0.0009326458,-0.0015565044,-0.7145983,0.6995326,,,,,,,,,,,,,,
1625117771611574016,simulation_car2,2.3366218,-22.381386,8.407551,-0.005773096,0.00059611[illegible]5,,,,,,,,,,,,,,
1625117771611574016,simulation_pedestrian1,3.6669,34.37481,7.4372497,-0.0,0.0,0.47335017,0.88087434,,,,,,,,,,,,,,
1625117771611574016,simulation_pedestrian2,13.023766,37.161892,7.4372497,0.0,-0.0,-0.27883342,0.9603395,,,,,,,,,,,,,,
1625117771611574016,simulation_pedestrian3,28.311188,36.918507,7.4372497,0.0,-0.0,-0.9999878,0.004927337,,,,,,,,,,,,,,
1625117771611574016,simulation_pedestrian4,19.580679,37.238995,7.4372497,0.0,-0.0,-0.9998059,0.0196972[illegible]
1625117771611574016,simulation_pedestrian5,2.1848068,32.075497,7.4372497,-0.0,0.0,0.103265,0.9946539,,,,,,,,,,,,,,
1625117771611574016,simulation_pedestrian6,22.406145,37.31971,7.4372497,-0.0,0.0,0.99531114,0.096725166,,,,,,,,,,,,,,
1625117771611574016,simulation_pedestrian7,37.506397,-14.879273,7.4372497,-0.0,0.0,0.30359143,0.9528023,,,,,,,,,,,,,,
1625117771611574016,simulation_pedestrian8,38.703953,-7.7071714,7.4372497,-0.0,0.0,0.785845,0.6184235,,,,,,,,,,,,,,
```


Only data from users (discard scatterers): uav1, simulation_car2, simulation_pedestrian4

Timeline

July 01, 2021

Preliminary dataset
Baseline reinforcement learning code

July 26, 2021

First part of the training data

August 10, 2021

Second part of the training data

mid-September 2021

Release of test episodes
Final submission

Thanks to all ITU-ML5G-PS-006 reinforcement learning team

Organizers

Francisco Müller

LASSE/UFPA

Aldebaro Klautau

LASSE/UFPA

Developers and Support

Cláudio Mello

LASSE/UFPA

Rebecca Aben-Athar

LASSE/UFPA

Felipe Bastos

LASSE/UFPA

Lucas Matni Bezerra

LASSE/Estácio

Emerson Oliveira

João Borges

Daniel Takashi

Ailton Oliveira

Join the challenge!

ITU Events

AI for Good

Machine Learning in 5G Challenge

Applying machine learning in communication networks

aiforgood.itu.int