MOREHOUSE COLLEGE

HUME CENTER FOR NATIONAL SECURITY AND TECHNOLOGY
VIRGINIA TECH

The Multi-Agent RL Testbed for RF Devices

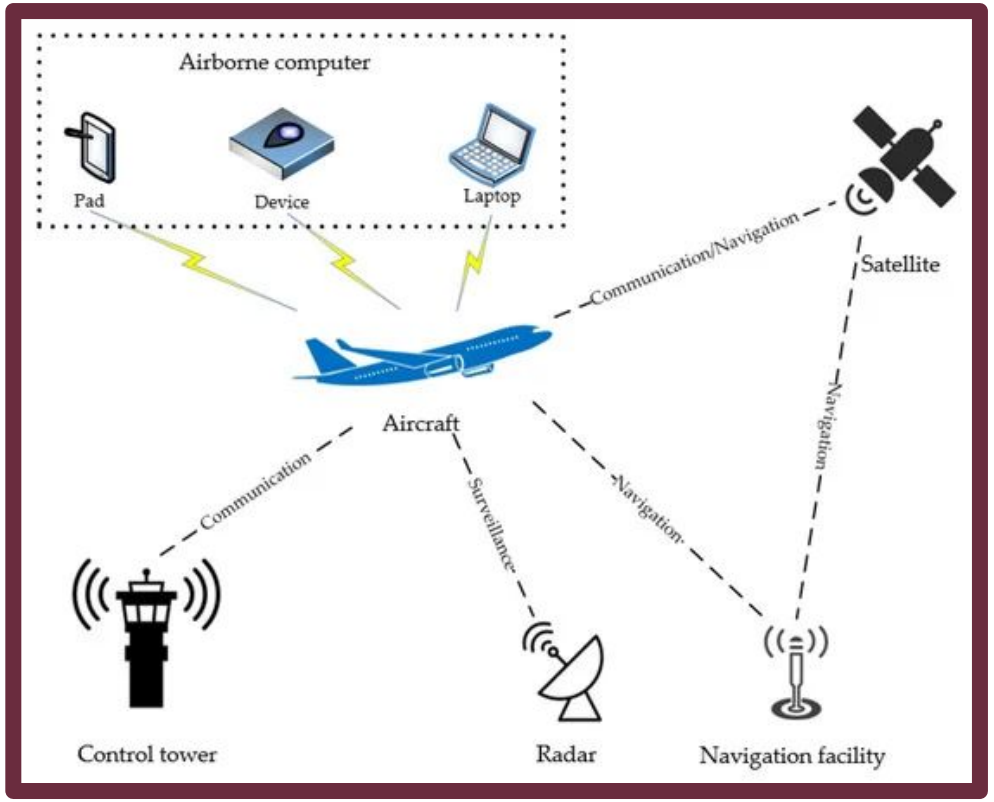
A tool for training and evaluating multi-agent reinforcement learning algorithms for cognitive radio systems

Sri Vangaru, Alyse Jones,
Dr. Chris Headley

www.hume.vt.edu

Overview

- Introduction and Motivation
  - Congestion in the wireless spectrum
  - Reinforcement learning and multi-agent systems
- Framework and Contributions
- Testing and Evaluation
- Conclusions and Future Work

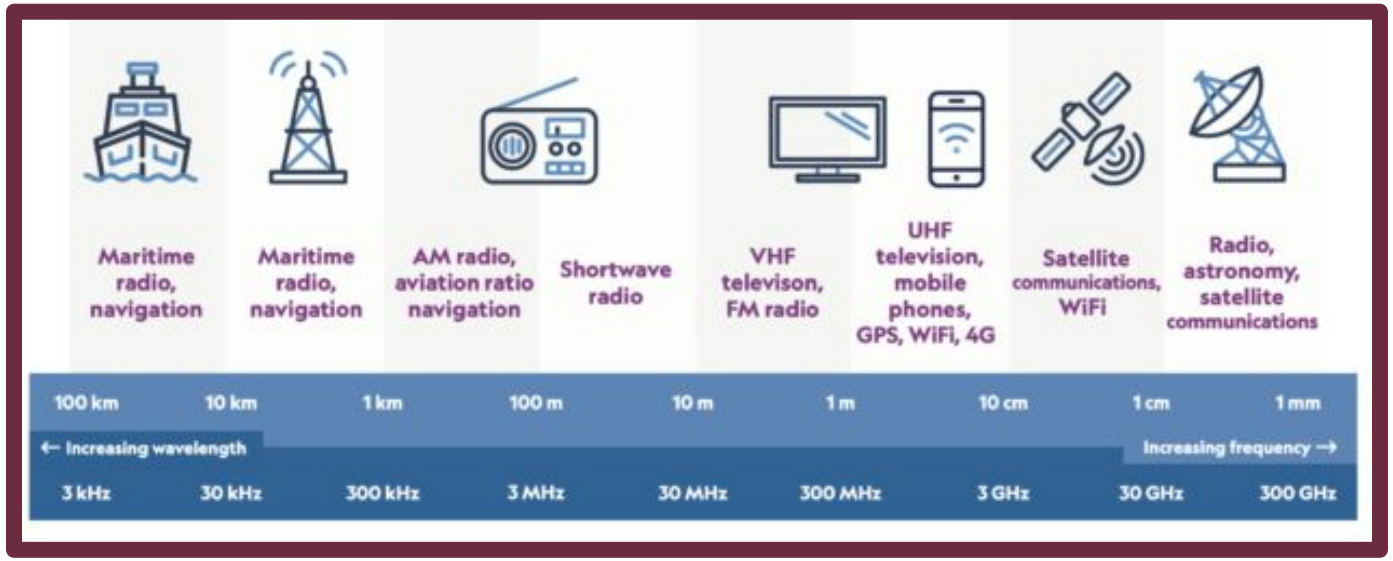


Source: [1]

Introduction and Motivation

Wireless Spectrum

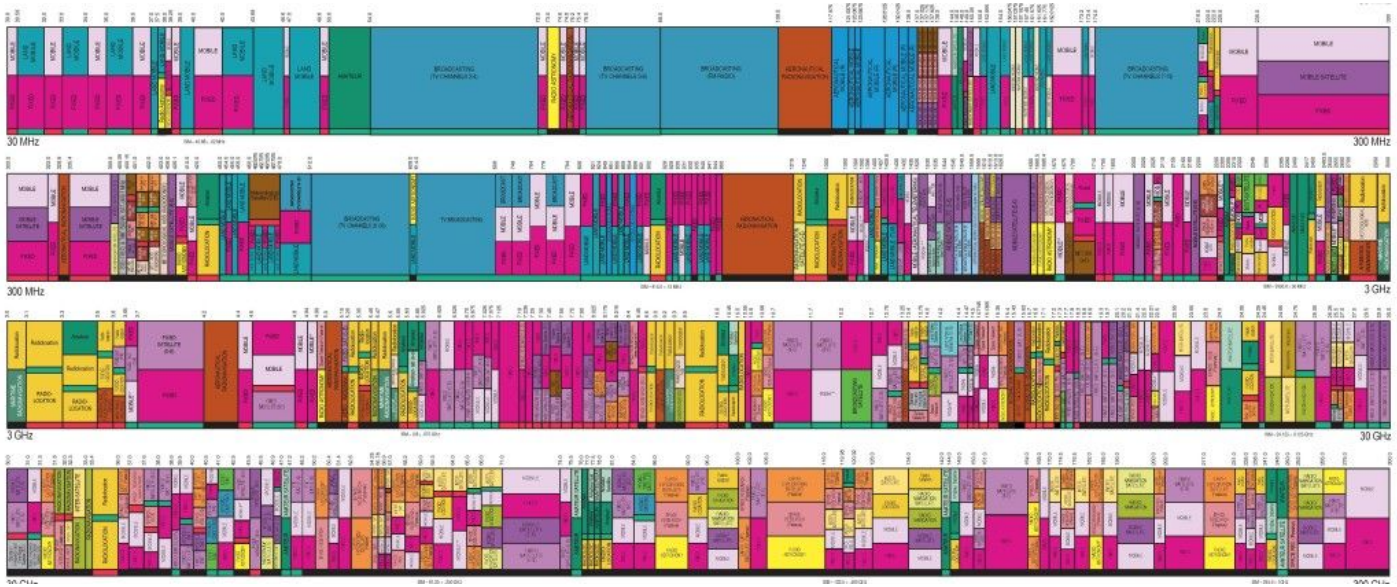
- A wide range of electromagnetic frequencies that are used for wireless communication
- Continuous frequencies, but grouped into discrete “bands” or “channels”
- Spectrum in our daily lives: Cellular, WiFi, Bluetooth
- Problem: Interference
  - Multiple transmissions at the same frequency
  - Receiver can not correctly interpret correct signal



Source: [2]

Causes of Interference

- Military: Electronic warfare
- Commercial: Interactions between consumer devices
  - 2010-2021: 138x increase in wireless spectrum use [3]
- FCC Spectrum Allocation



Source: [4]

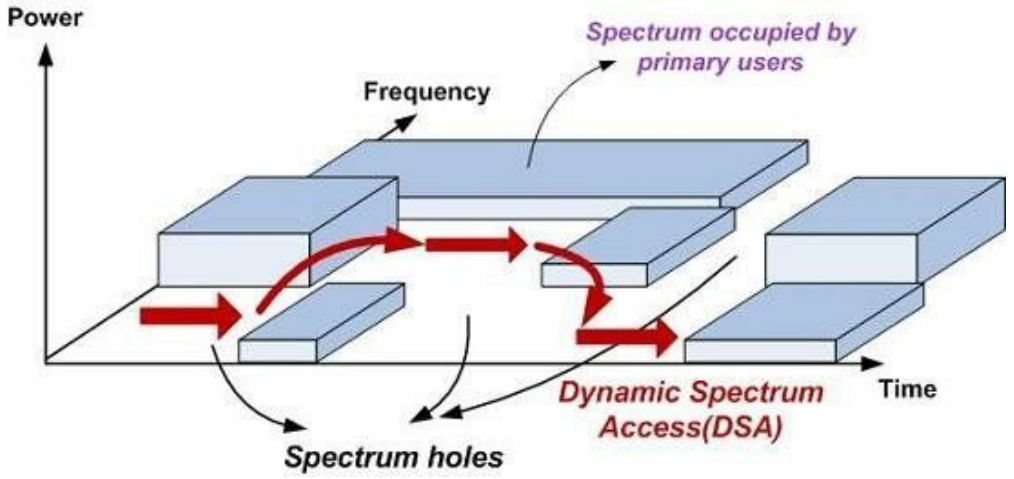
Spectrum Management

Cognitive Radio (CR)

Radio with the ability to **dynamically change** transmission parameters (center frequency, bandwidth) based on spectrum sensing results. Programmable through software.

Dynamic Spectrum Access (DSA)

Algorithmically switches to vacant frequencies to avoid other entities. Spectrum holes found by spectrum sensing.



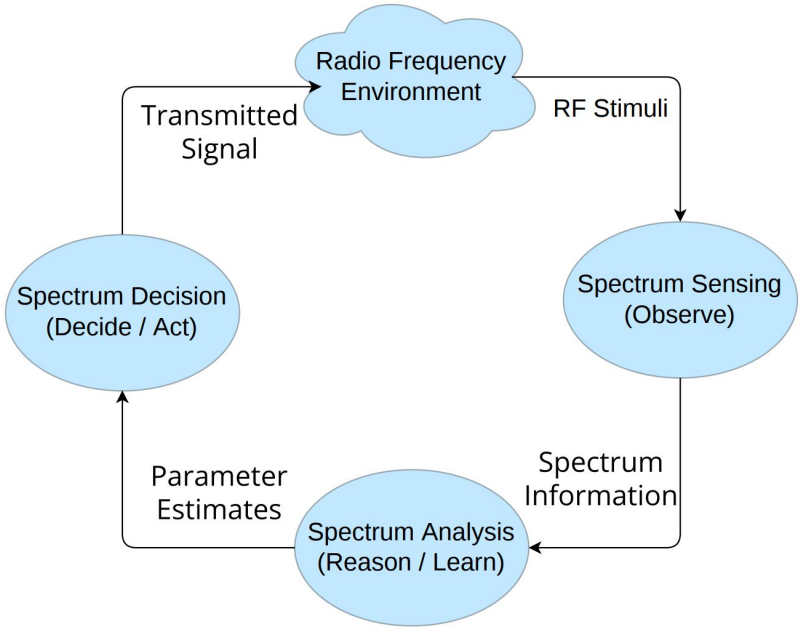
Source: [5]

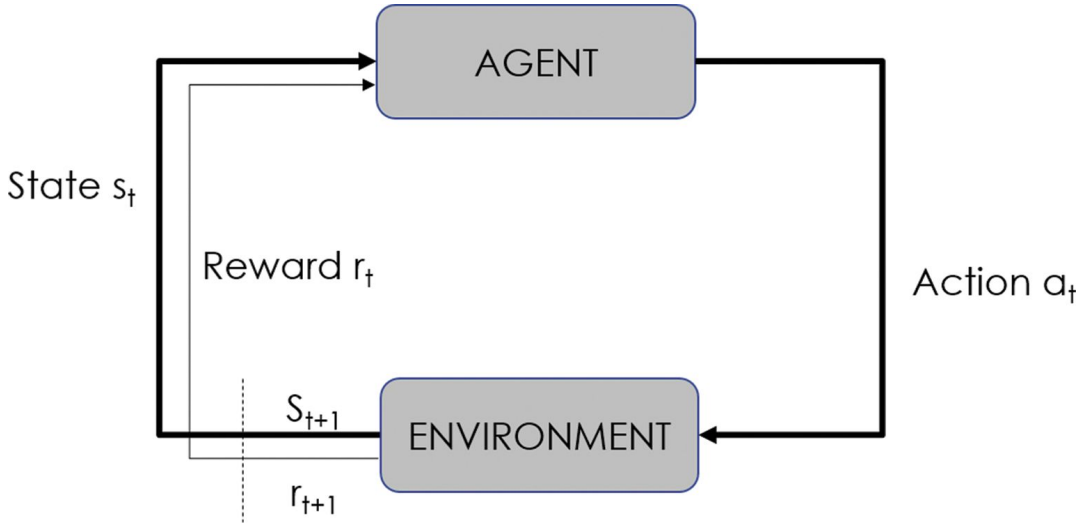
CRs with DSA alone are **limited** in predicting future spectrum usage, resulting in interference

Reinforcement Learning in Spectrum Management

OODA: Observe Orient Decide Act

- A common decision-making strategy loop
- Can apply reinforcement learning (RL) to this framework
  - Make **predictions** about future spectrum activity
  - Select action for message transmission

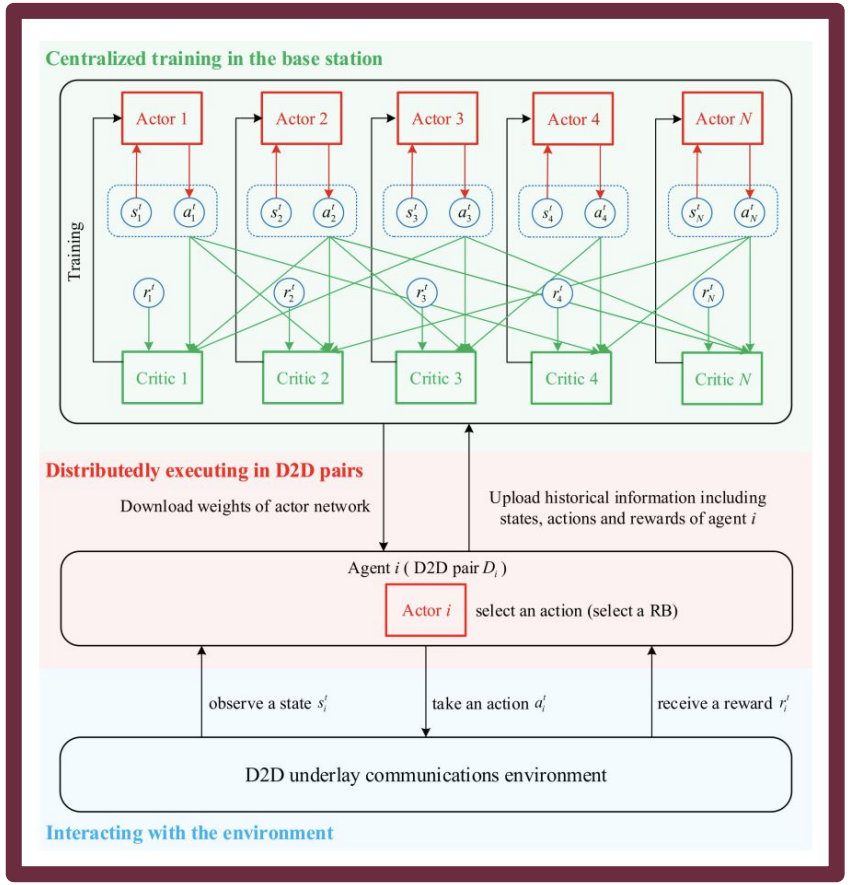




Source: [6]

Motivation for Multi-Agent Reinforcement Learning (MARL)

- Increased device usage in the spectrum [7] → unsafe to assume only one agent exists
  - When using RFRL, devices would realistically be **learning simultaneously**
  - Need MARL to deal with resulting game dynamics
- Devices could fail to find free bands **when deployed** under the single-agent assumption
- Other examples of MARL in RF:
  - Handling the spectrum allocation itself with MARL [8]
  - Decentralized MARL for UAVs [9]

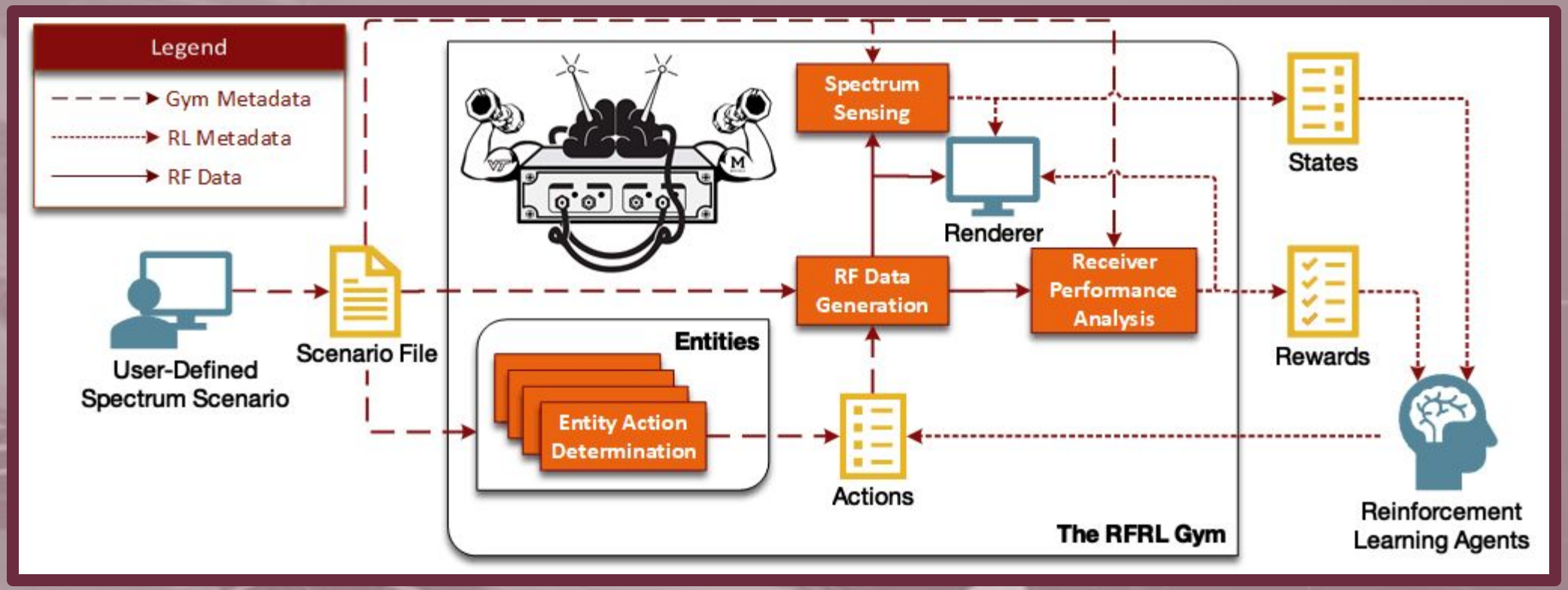


Source: [8]

Motivation of the RFRL Gym

1. **Enhance** research capabilities in wireless spectrum applications
   - Easy-to-use cognitive radio-centric research tool
2. **Address** pressing problems of wireless spectrum management
   - Reducing interference in the wireless spectrum
3. **Expand** interest and knowledge in RL-based wireless communications
   - Single-agent [10] and multi-agent

The Multi-Agent RFRL Gym Framework



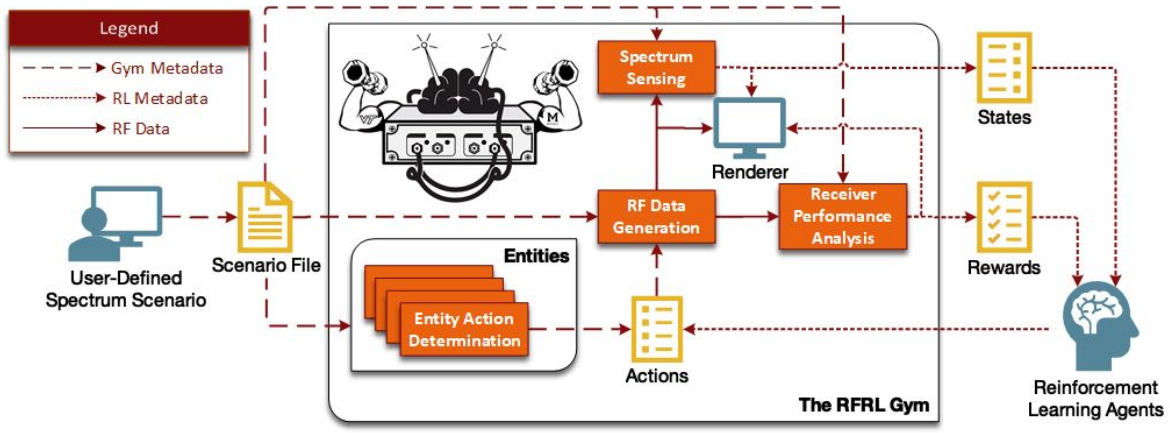
Contributions of the Multi-Agent RFRL Gym

- Simulate and test the performance of **single-agent and multi-agent RL algorithms** for cognitive radio applications (such as DSA and jamming)
  - Built with the OpenAI Gym, and is customizable
- The RFRL Gym contains the following **novel features**:
  - Integration with RLlib for RF-specific tasks
  - Customizable scenarios with distinct configurations for each agent
  - Out-of-the-box entities
  - Multiple rendering options
  - GUI

Features	RFRL Gym	GrGym [23]	ns3-Gym [24]	Colosseum [25]
Flexible Scenario Design	✓			
RL Package Compatibility	✓	✓	✓	
Spectrum Sensing Capabilities	✓		✓	✓
Multi-Agent Capabilities	✓		✓	✓
Ease of Use	✓			
Graphical User Interface	✓			✓
Hardware Compatible		✓	✓	

Why OpenAI Gym?

- OpenAI Gym integration (using the Farama Foundation *Gymnasium* fork)
  - Standardized API
  - Observation spaces
  - Action spaces
- Easy integration with external RL packages
  - Stable Baselines, Hugging Face, Ray RLlib, etc.
  - Pytorch, Tensorflow, etc.
  - Out-of-the-box MARL algorithms with RLlib: Deep Q-Networks, Soft Actor-Critic, etc.



Scenario: Configuration of the Simulation

- Entities: Signals moving in spectrum
- For each agent:
  - Observation modes: Classification and Detection
  - Reward modes: Jamming or DSA
  - Render Modes: Terminal or PyQt
- Rendering parameters

$$r_{DSA}(a) = \begin{cases} 1, & \text{no collision} \\ 0, & \text{no transmission} \\ -1, & \text{collision} \end{cases}$$

$$r_{jamming}(a) = \begin{cases} 1, & \text{collision with target entity} \\ 0, & \text{no transmission} \\ -1, & \text{transmitting elsewhere} \end{cases}$$

Reward *r* for any given agent *a*

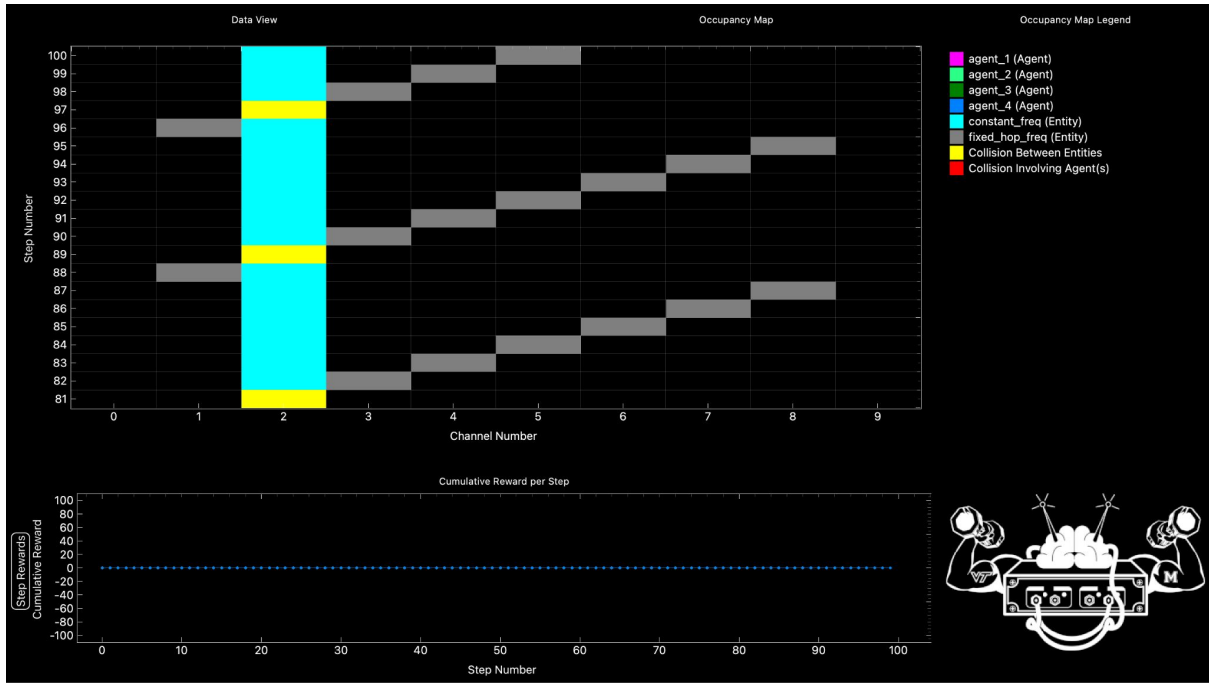
Example Scenario File Generation

```
{
    "environment":
    {
        "num_channels": 10,
        "max_steps": 100,
        "comments": [
            "Four agents, and two entities (one constant frequency and one fixed-hop).",
            "The observation modes vary, the reward modes are all 'jam'",
            "and the target entities vary."
        ],
        "agents": {
            "agent_1": {
                "observation_mode": "detect",
                "reward_mode": "jam",
                "target_entity": "constant_freq"
            },
            "agent_2": {
                "observation_mode": "classify",
                "reward_mode": "jam",
                "target_entity": "constant_freq"
            },
            "agent_3": {
                "observation_mode": "detect",
                "reward_mode": "jam",
                "target_entity": "fixed_hop_freq"
            },
            "agent_4": {
                "observation_mode": "classify",
                "reward_mode": "jam",
                "target_entity": "fixed_hop_freq"
            }
        }
    },
```

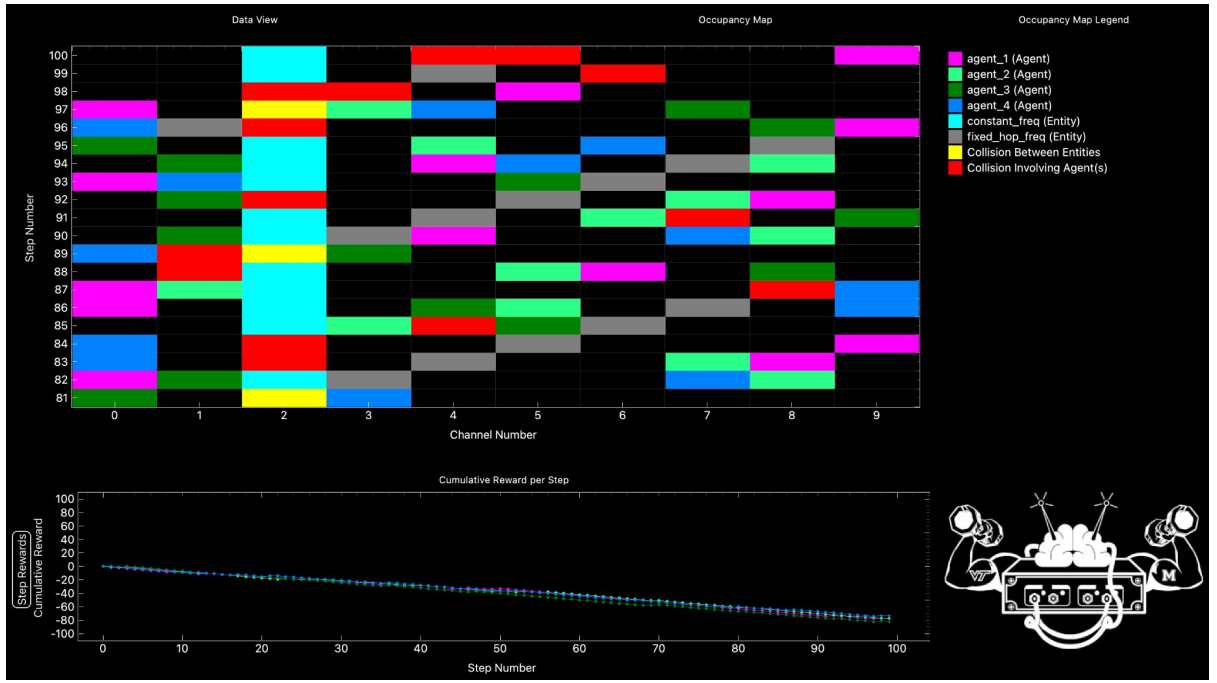
```
    "entities":
    {
        "constant_freq":
        {
            "type": "ConstantFreq",
            "channels": [2],
            "onoff": [1,1,0],
            "modem_params":
            {
                "type": "qam",
                "order": 16,
                "filter": "RRC",
                "center_frequency": [-0.1,0.1],
                "bandwidth": 0.25,
                "start": 0.25,
                "duration": 0.25
            }
        },
        "fixed_hop_freq":
        {
            "type": "FixedHopFreq",
            "channels": [1,2,3,4,5,6,7,8],
            "onoff": [1,1,0],
            "rand_hop": 0,
            "modem_params":
            {
                "type": "n_fmcw",
                "center_frequency": [0.0,0.0],
                "bandwidth": 1.0,
                "start": 0.0,
                "duration": 1.0
            }
        }
    },
    "render":
    {
        "render_mode": "pyqt",
        "render_fps": 100,
        "render_history": 20,
        "render_background": "black",
        "observation_mode": "classify"
    }
}
```

Example Scenario File Preview

Without agent transmissions:

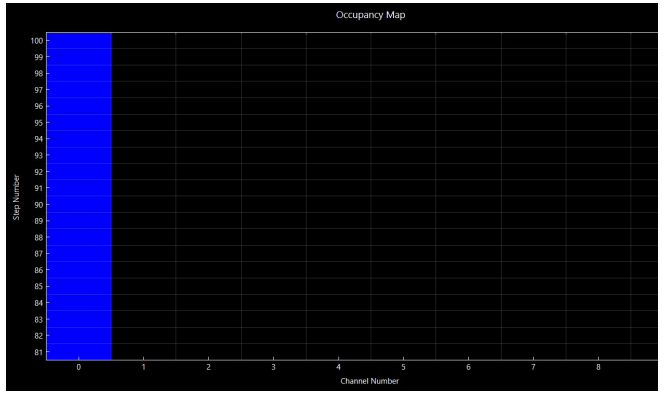


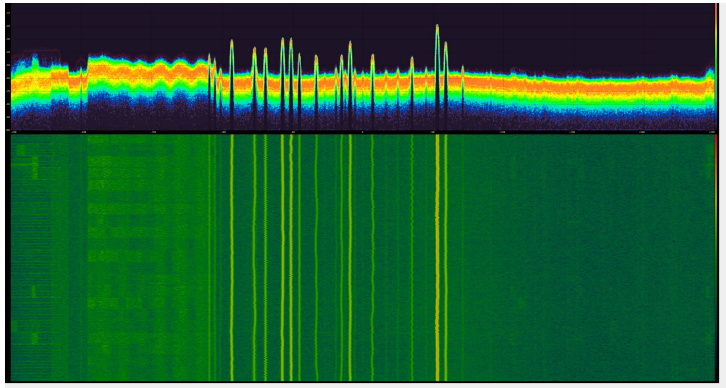
With agent transmissions:

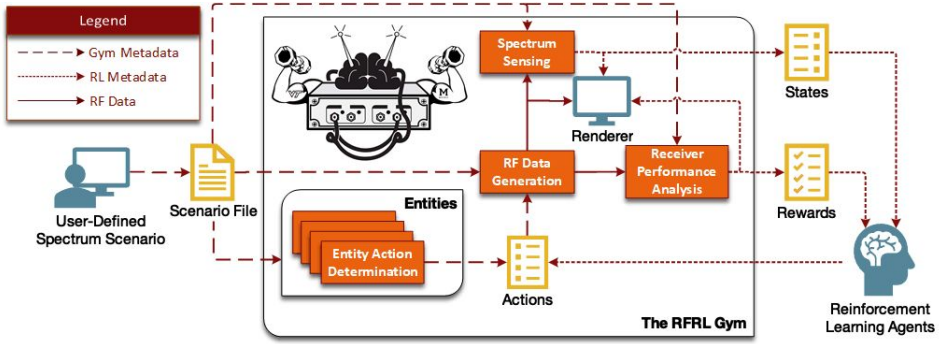


Entities: Non-Learning Devices

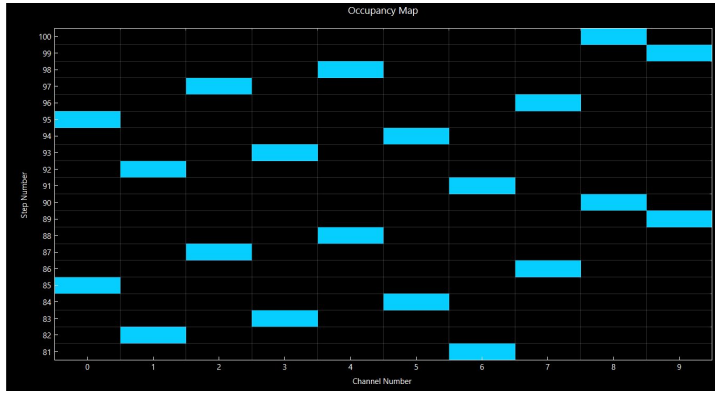
Constant Frequency Entity: Stays in one channel

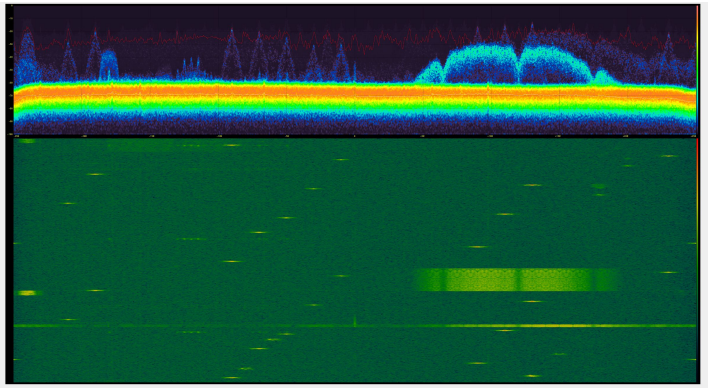






Hopper Entity: Move around with user-defined pattern





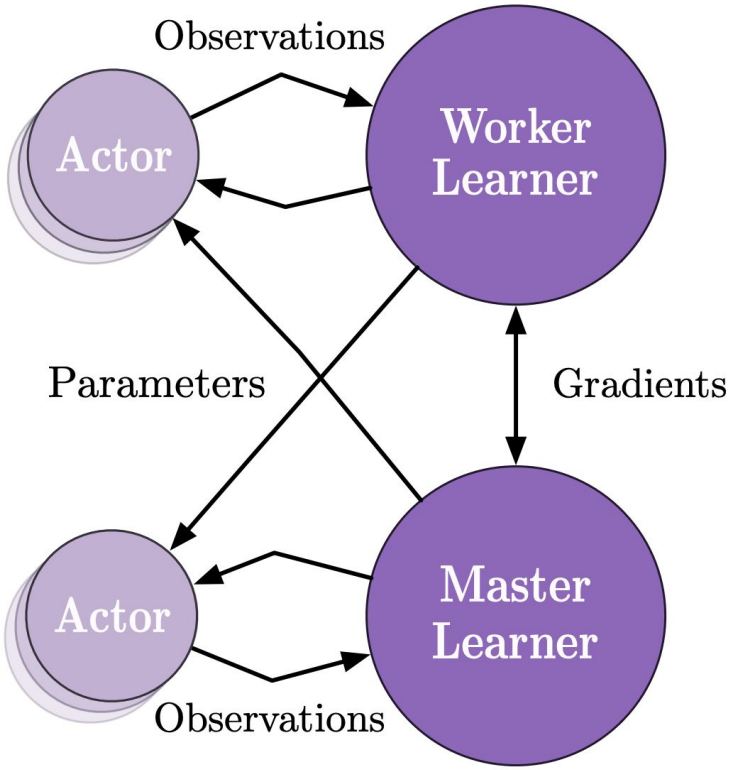
All Entity Types

Entity Type	Description	Real World Example
Constant	Remains in one channel	T.V. or Radio
Fixed Hopper	Moves in user-defined pattern	Bluetooth
Stochastic Hopper	Selects channel based on user-defined probabilities for each channel	Multi-Armed Bandit
Agile Hopper	Moves to empty channel if available. Has perfect knowledge of other entities in environment	CR with DSA
Simple Jammer	Moves to an occupied channel. Has perfect knowledge of other entities in environment	Intelligent Adversary
Custom	**Up to the user!**	

Testing and Evaluation

Testing Methodology

- Tried out 4 RLlib-provided MARL algorithms:
  - Multi-Agent Deep Q-Networks (**DQN**) [11]
  - Multi-Agent Proximal Policy Optimization (**PPO**) [12]
  - Multi-Agent Asynchronous Proximal Policy Optimization (**APPO**) [12]
  - Importance Weighted Actor-Learner Architecture (**IMPALA**) [13]
- Tested 5 scenarios with varying difficulty
  - Rewards for each algorithm are plotted as sums of agent rewards

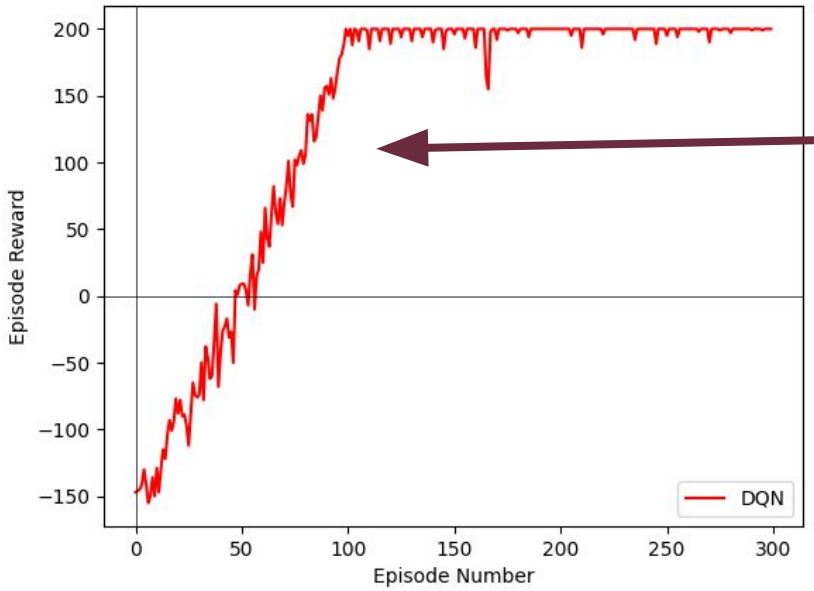


Source: [13]

Scenario: Fixed-Hopping Entity Jamming

Scenario: 1 fixed-hopper entity, 2 jamming agents

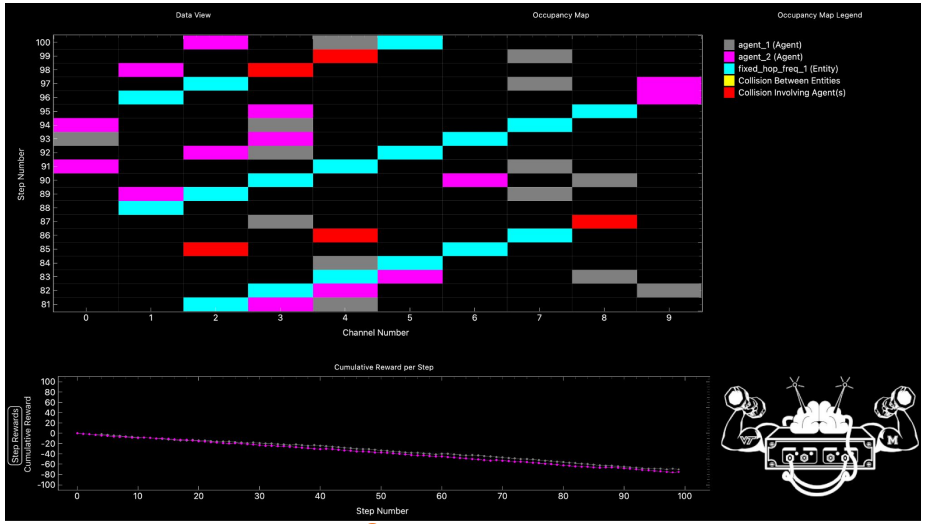
$$r_{jamming}(a) = \begin{cases} 1, & \text{collision with target entity} \\ 0, & \text{no transmission} \\ -1, & \text{transmitting elsewhere} \end{cases}$$



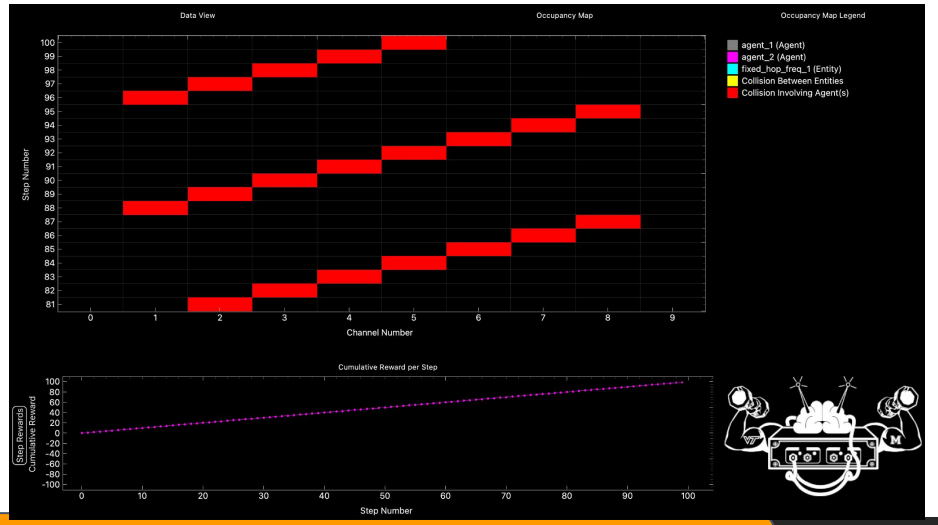
DQN had relatively quick convergence.

This is likely due to its strong performance in discrete action spaces.

DQN Before Convergence



DQN After Convergence



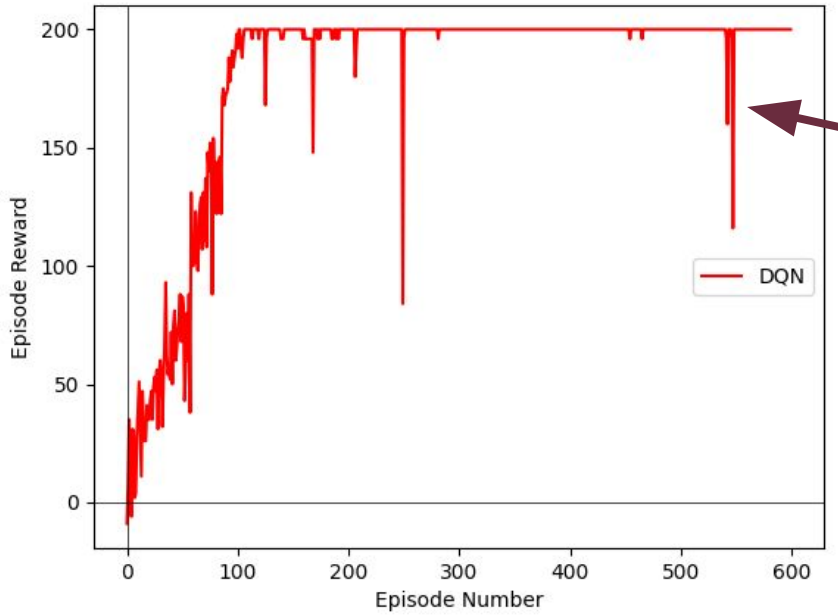
Refer to **Scenario 2** in **Section V** of the paper

The RFRL Gym shows expected results.

Scenario: Finding Free Channels

Scenario: 2 constant frequency entities, 2 DSA agents

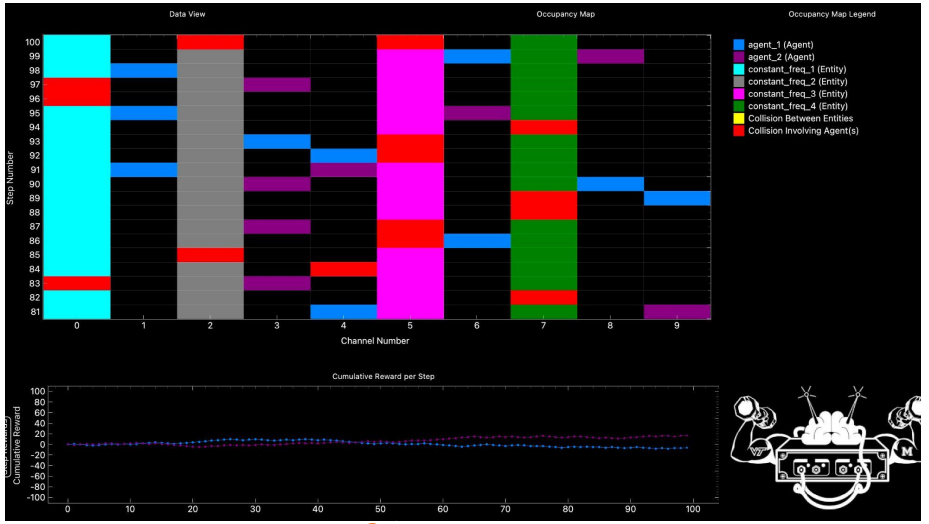
$$r_{DSA}(a) = \begin{cases} 1, & \text{no collision} \\ 0, & \text{no transmission} \\ -1, & \text{collision} \end{cases}$$



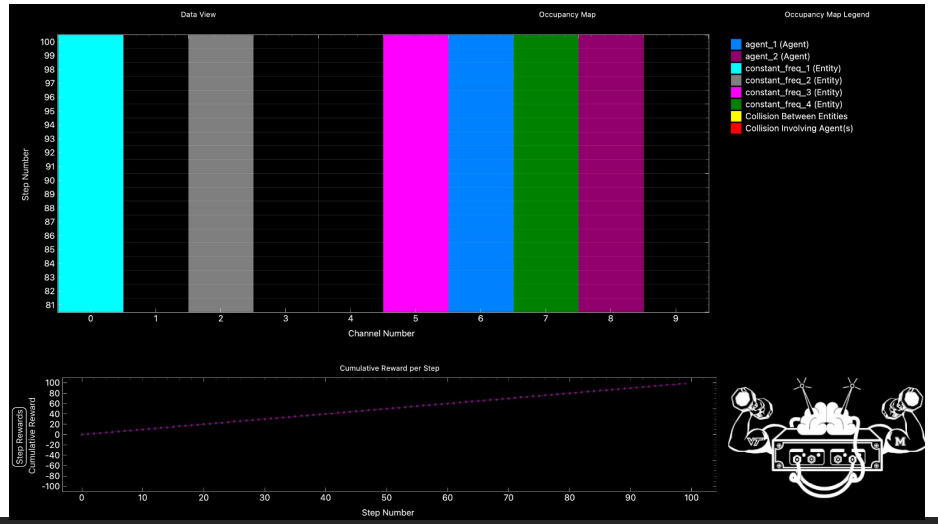
DQN's epsilon parameter required tweaking,

but still resulted in suboptimal behavior in some training episodes.

DQN Before Convergence



DQN After Convergence

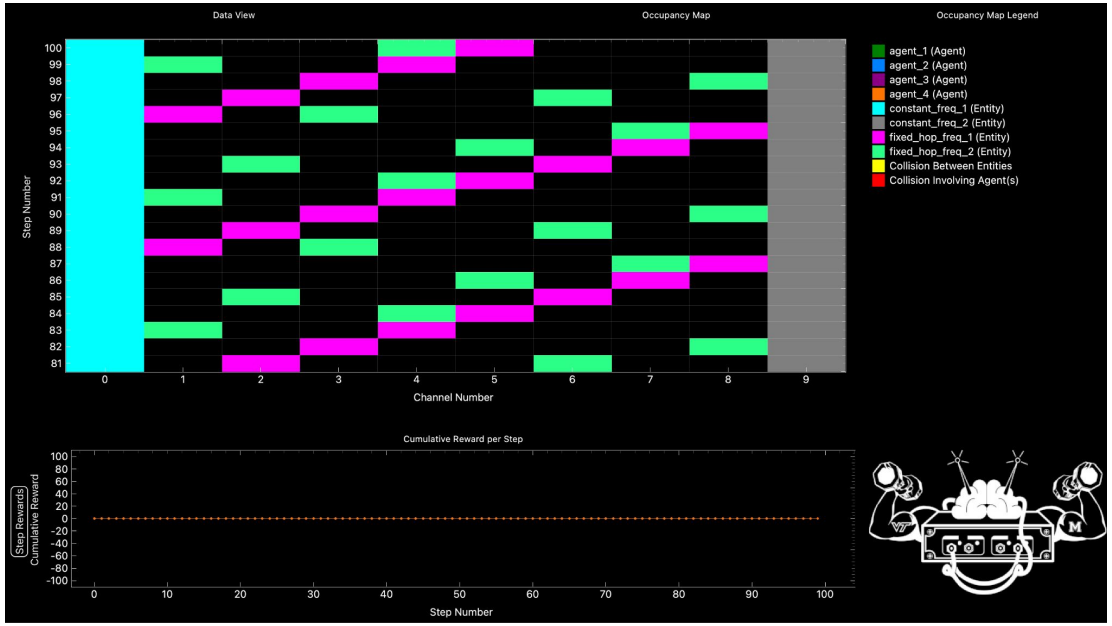


Refer to **Scenario 2** in **Section V** of the paper

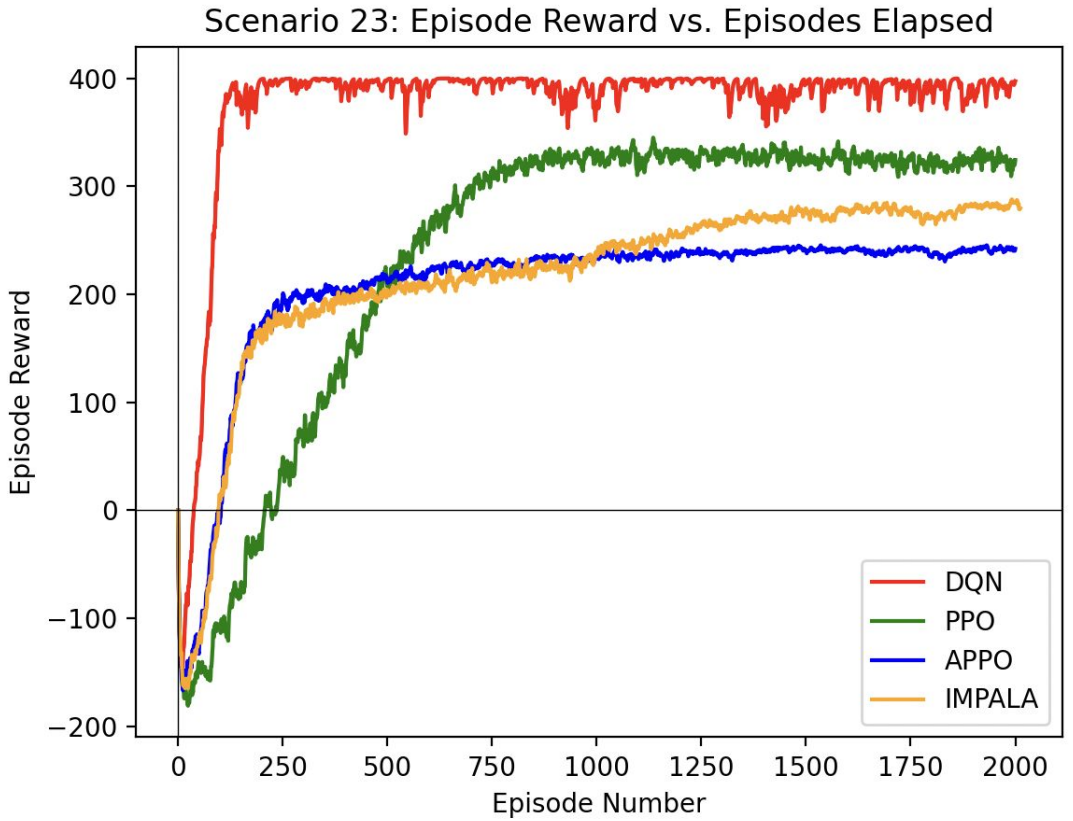
Harder Scenario: Variety of Entities and Reward Functions

Scenario:

- Entities: 2 constant, 2 fixed-hop
- Agents: 2 jamming, 2 DSA



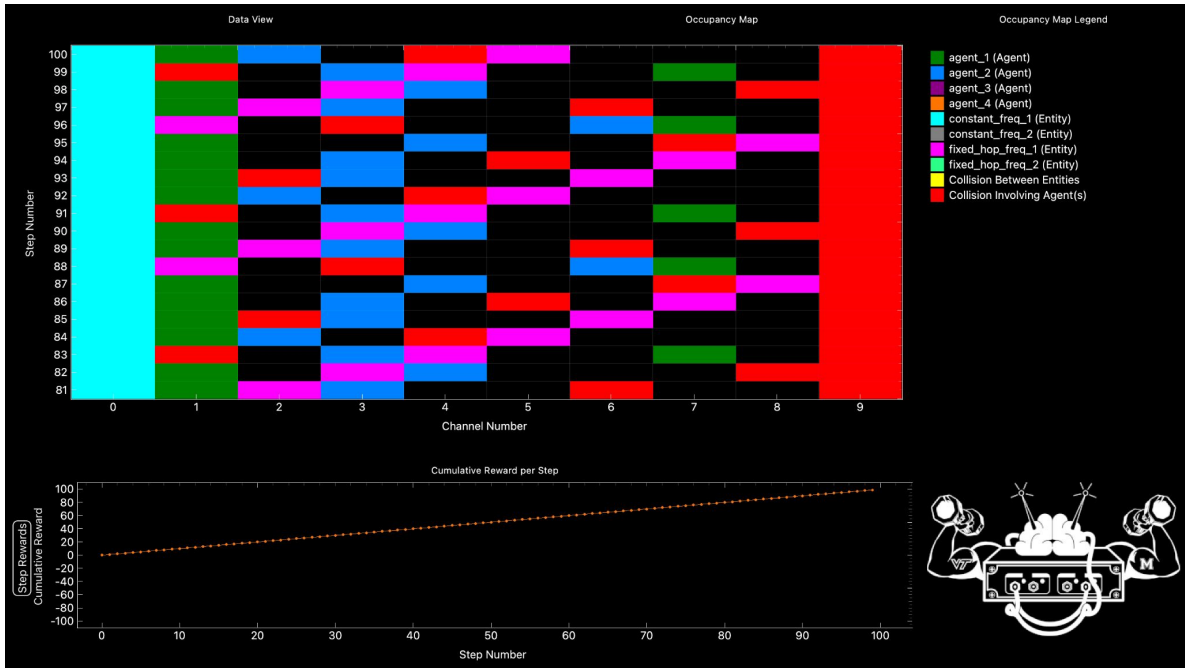
Scenario shown with no agent transmissions.



DQN converged optimally; APPO didn't. Why?

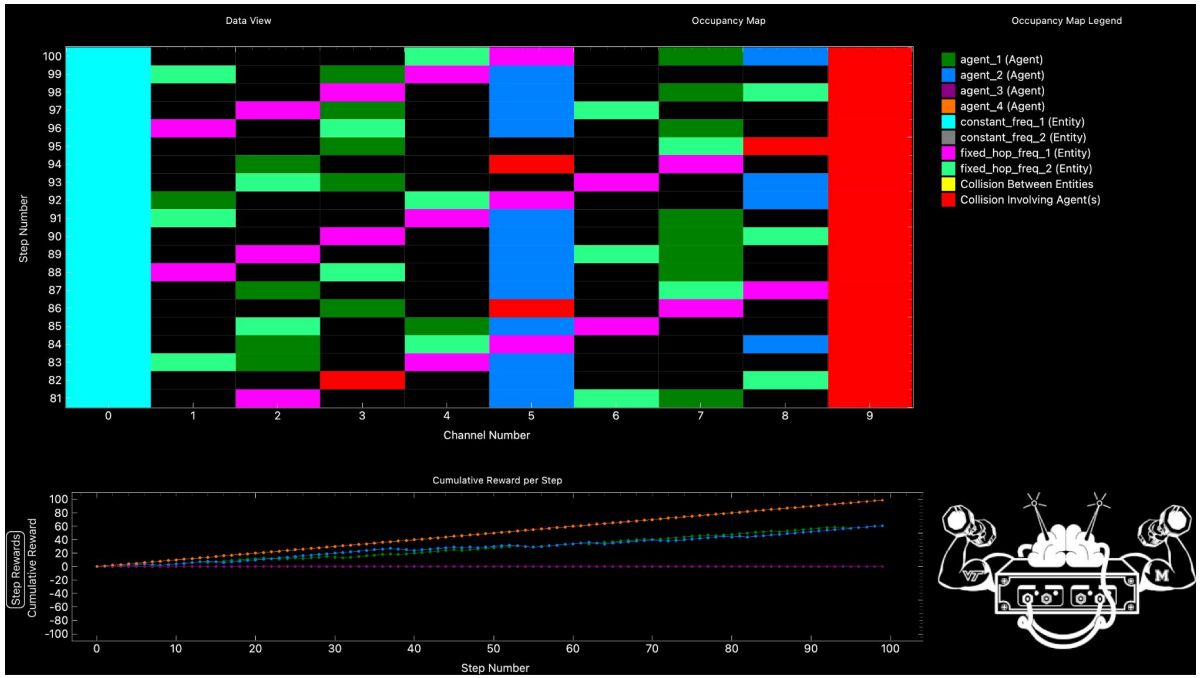
Harder Scenario Continued

DQN after training:



The RFRL Gym can show the limitations of multi-agent learning algorithms!

APPO after training:



The fixed-hop jamming agent **stopped transmitting** and the DSA agents sometimes choose to not transmit.

Refer to **Scenario 5** in **Section V** of the paper

Conclusions and Future Work

Future Work

- Multi-channel agents
- Channel modeling approaches
- Customizable reward functions
- Hardware integration with software-defined radios (SDRs)

Source: [14]

References

[1] Zheng, R., Li, X., & Chen, Y. (2023). An Overview of Cognitive Radio Technology and Its Applications in Civil Aviation. *Sensors*, 23(13), 6125. https://doi.org/10.3390/s23136125

[2] Maxar Intelligence & Maxar Space Systems. (2023) RF Solutions. *maxar.com.* https://maxar.com/products/rf-solutions

[3] CTIA. (2022) Summary of CTIA's Annual Wireless Industry Survey. *ctia.org.* https://api.ctia.org/wp-content/uploads/2022/09/Summary-of-CTIAs-Wireless-Industry-Survey-2022.pdf

[4] Reed, M. (2014) FCC Spectrum Decision Good for App Makers. *ACT | The App Association.* https://actonline.org/2014/05/15/todays-fcc-spectrum-decision-great-for-app-makers/

[5] Janani Selvaraj, Ramaswamy M, Samuel Manoharan J. (2018) Investigative Study on Routing Methods for CSRN. *International Journal of Latest Trends in Engineering and Technology*, 10(2), pp. 051 - 060. https://www.ijltet.org/journal_details.php?id=930&j_id=4468

[6] Ramirez Rebollo, D. R., Ponce Cruz, P., & Molina, A. (2016). CODA Algorithm: An Immune Algorithm for Reinforcement Learning Tasks. *InTech*. doi: 10.5772/63570

[7] Accenture. (2022) Spectrum Allocation in the United States. *ctia.org.*https://api.ctia.org/wp-content/uploads/2022/09/Spectrum-Allocation-in-the-United-States-2022.09.pdf

[8] Z. Li, C. Guo, and Y. Xuan. (2019) A Multi-Agent Deep Reinforcement Learning Based Spectrum Allocation Framework for D2D Communications. *2019 IEEE Global Communications Conference (GLOBECOM)*, pp. 1-6. doi: 10.1109/GLOBECOM38437.2019.9013763.

[9] A. Feriani and E. Hossain. (2021) Single and Multi-Agent Deep Reinforcement Learning for AI-Enabled Wireless Networks: A Tutorial. *IEEE Communications Surveys & Tutorials*, 23 (2), pp. 1226-1252. doi: 10.1109/COMST.2021.3063822

[10] D. Rosen et al., "RFRL Gym: A Reinforcement Learning Testbed for Cognitive Radio Applications," 2023 International Conference on Machine Learning and Applications (ICMLA), Jacksonville, FL, USA, 2023, pp. 279-286, doi: 10.1109/ICMLA58977.2023.00046.

[11] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing Atari with Deep Reinforcement Learning. *arXiv [cs.LG]*. Retrieved from http://arxiv.org/abs/1312.5602

[12] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms. *arXiv [cs.LG]*. Retrieved from http://arxiv.org/abs/1707.06347

[13] Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., … Kavukcuoglu, K. (2018). IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures. *arXiv [cs.LG]*. Retrieved from http://arxiv.org/abs/1802.01561

[14] Ettus Research. (2025) USRP B205mini-i. *www.ettus.com*. https://www.ettus.com/all-products/usrp-b205mini-i/

Author emails:

sriv04@vt.edu

alysemjones@vt.edu

cheadley@vt.edu

Link to codebase:

https://github.com/vtnsi/rfrl-gym

Thank you for listening.

Any questions?