RFRL Gym: A Reinforcement Learning Testbed for Cognitive Radio Applications

Daniel Rosen*, Illa Rochez*, Caleb McIrvin*, Joshua Lee*, Kevin D'Alessandro*, Max Wiecek*, Nhan Hoang*, Ramzy Saffarini*, Sam Philips*, Vanessa Jones*, Will Ivey*, Zavier Harris-Smart†, Zavion Harris-Smart†, Zayden Chin†, Amos Johnson†, Alyse M. Jones*, William C. Headley*
*Virginia Tech National Security Institute, †Morehouse College
amos.johnson@morehouse.edu, alysemjones@vt.edu, cheadley@vt.edu

2023 International Conference on Machine Learning and Applications (ICMLA) | 979-8-3503-4534-6/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICMLA58977.2023.00046

***Abstract*—Radio Frequency Reinforcement Learning (RFRL) is anticipated to be a widely applicable technology in the next generation of wireless communication systems, particularly 6G and next-gen military communications. Given this, our research is focused on developing a tool to promote the development of RFRL techniques that leverage spectrum sensing. In particular, the tool was designed to address two cognitive radio applications, specifically dynamic spectrum access and jamming. In order to train and test reinforcement learning (RL) algorithms for these applications, a simulation environment is necessary to simulate the conditions that an agent will encounter within the Radio Frequency (RF) spectrum. In this paper, such an environment has been developed, herein referred to as the RFRL Gym. Through the RFRL Gym, users can design their own scenarios to model what an RL agent may encounter within the RF spectrum as well as experiment with different spectrum sensing techniques. Additionally, the RFRL Gym is a subclass of OpenAI gym, enabling the use of third-party ML/RL Libraries.**

We plan to open-source this codebase to enable other researchers to utilize the RFRL Gym to test their own scenarios and RL algorithms, ultimately leading to the advancement of RL research in the wireless communications domain. This paper describes in further detail the components of the Gym, results from example scenarios, and plans for future additions.

***Index Terms*—machine learning, reinforcement learning, wireless communications, dynamic spectrum access, OpenAI gym**

I. Introduction

Due to the rapid growth of Radio Frequency (RF) spectrum usage, efficient use of the spectrum has become increasingly difficult, leading to over-congestion in the RF spectrum [1]. This largely is a consequence of the exponential increase in wireless devices, with the CTIA citing a growth of 183 times its size since 2010 in its 2021 Wireless Internet Survey [2], ultimately resulting in decreased data rates from poor channel conditions and collisions [3]. These collisions are a form of signal jamming, also known as interference, leading to incomplete and corrupted transmissions. Interference can either be intentional from an adversary or unintentional due to less-than-ideal conditions in the RF spectrum. Furthermore, spectrum congestion makes it more difficult to differentiate whether the collision was from an adversary or over-congestion. Regardless, to avoid further loss to communication systems, both interference types must be prevented. Thus, to prevent interference as well as losses in data rate, there is a clear need for improvement in the utilization of the RF Spectrum.

In recent years, Cognitive Radios (CR) have gained traction as a promising solution to alleviating spectrum congestion [4] due to its ability to intelligently determine which channels are currently occupied, and which are not through Dynamic Spectrum Access (DSA) [5]. DSA was proposed to overcome the shortcomings of traditional methods, such as Frequency Hopping Spread Spectrum (FHSS) [6] and Direct Sequence Spread Spectrum (DSSS) [7], where both fail in a dynamic RF spectrum from not adapting its method to where holes in the spectrum are located. DSA, however, is able to do so through spectrum sensing, where methods such as energy detection are utilized to identify which frequency bands are in use [8], allowing a CR to switch to an empty band. However, DSA alone is not enough to handle spectrum congestion since it cannot *predict* the location of future spectrum holes, resulting still in accidental collisions. However, by coupling DSA with predictive intelligence, a CR can predict where and when other users will occupy the RF spectrum in the future. One promising solution is Reinforcement Learning (RL).

RL is a Machine Learning method known for its ability to predict the best decisions to achieve a desired outcome [9]. When applied to CRs, RL's predictive capabilities enable signals to preemptively respond to the movements of other signals in the spectrum as well as changing channel conditions such that the optimal frequency at which to transmit can be identified. Significant effort has been invested into investigating RL-based protocols for spectrum assignment and wireless communication enhancement, where RL has shown to drastically reduce congestion in the spectrum [10]–[13].

From these works, it is clear that RL Algorithms *must* be trained in a representative environment, in this case the RF spectrum. However, there is currently an absence of a comprehensive, universal tool simulating the RF spectrum that researchers can use to develop and test RL algorithms for CR-specific applications. Conventionally, new tools have been developed for each project and can not be easily used in other applications. Thus, this paper introduces the RFRL Gym[1], a training environment of the RF spectrum designed to provide comprehensive functionality, such as custom scenario generation, multiple learning settings, and compatibility with third-party RL packages. As a result, the RFRL Gym can

[1]https://github.com/vtnsiSDD/rfrl-gym

1946-0759/23/$31.00 ©2023 IEEE
DOI 10.1109/ICMLA58977.2023.00046

be applicable to a wide range of wireless communications projects, where researchers can utilize the tool to further the advancement of RL in wireless communications. Additionally, through a gamified mode of the RF spectrum, this tool can be used to teach novices about the fields of AI/ML and RF. The remainder of this paper defines the current state of the RFRL Gym, its capabilities, and planned future enhancements.

II. Background

As mentioned, prior works [14]–[17] have developed tools to achieve RL for their own wireless communications applications. Table I describes how the RFRL Gym differs from these existing tools through the following notable key features.

Flexible Scenario Design: The ability to create, modify, or utilize different scenarios to view how RL algorithms will react and interact in different learning spaces. [16] was designed specifically for a particular use case of wireless communications and contained no ability to create custom scenarios on their framework. [17] did have some included scenarios within their gym, such as for WiFi and cellular scenarios. However, it is unclear if custom scenarios can be generated and if so, how difficult it would be. [14], [15] contain the most flexibility in scenario generation of the three since its data is generated through GNU Radio, a platform to design communication systems [18]. However, this process is not straightforward, as designing flow-graphs in GNU Radio can be difficult for non-experts and those not familiar with the software. Therefore, the RFRL Gym includes flexible scenario generation where users can easily create custom scenarios according to their application through input json files.

RL Package Compatibility: The ability to interface with other RL libraries to expand a user's access to different RL agents. Examples include Mushroom RL [19] and Stable Baselines [20]. This feature removes the need to implement RL algorithms manually if not desired. Of the tools considered, only the RFRL Gym has explicitly tested and confirmed this feature. Additionally, custom creation of RL algorithms is also possible within the RFRL Gym.

Spectrum Sensing Capabilities: The ability to locate spectrum holes through spectrum sensing. To fully realize a DSA scenario, this is a requirement. [17] does have capabilities for spectrum sensing, but to our knowledge, there is no ability to create custom spectrum sensing methods.

Multi-Agent Capabilities: The ability to support several intelligent agents (i.e. RL agents). [16] and [17] do have capabilities for multi-agent reinforcement learning. It is our desire for future work to include this functionality within the RFRL Gym.

Ease Of Use: The tool is designed with considerations for an easy user experience, especially for those with little or no familiarity with wireless communications and/or RL. This feature allows for ML/RL specialists to better interact and interface with Wireless Communication specialists through easy scenario generation, provided examples, and a gamified mode to learn the wireless communications concepts at an abstracted level. By doing so, the RFRL Gym suits the needs for both novices to learn RFRL concepts and experts to use the gym to further their research.

TABLE I: Comparison of existing OpenAI Gym RL tools for wireless communications

Features	**RFRL Gym**	**GrGym** [14], [15]	**MR-iNet Gym** [16]	**ns3-Gym** [17]
Flexible Scenario Design	✓	✓		✓
RL Package Compatibility	✓			
Spectrum Sensing Capabilities	✓			✓
Multi-Agent Capabilities			✓	✓
Ease of Use	✓			
Graphical User Interface	✓			
Hardware Compatible		✓	✓	

Graphical User Interface: The ability to control the simulation (i.e. scenario generation) through a Graphical User Interface (GUI) to make it easier for users to interface with the RFRL Gym. Existing tools are all controlled through scripts and a command line interface. If the user wants to make scenario generation changes, they are forced to change it within the code. However, a GUI bypasses this need such that users can make any changes to the scenario without touching the code. Future plans for the GUI include controlling RL hyperparemeters and environment parameters such that the entire simulation can be run entirely from the GUI.

Hardware Compatible: The ability to operate over radio hardware in real-time, rather than simulation alone, for a realistic implementation. [14], [15] are hardware compatible since GNU Radio supports radio hardware. Additionally, a primary contribution of [16] was to implement their system over embedded systems in radio hardware. In future versions, the RFRL Gym will also aim for hardware compatibility.

In comparison to existing tools described above, the RFRL Gym was designed to serve as a universal, easy-to-use tool that both novices and experts within this field can use to learn about AI/ML and RF as well as further technological advancement of RFRL. For novices, in the gamified mode, the complex features of wireless communications within the RFRL Gym are abstracted so they can use the tool with limited knowledge of wireless communications and be eased into more advanced topics, such as signal modulation and the generation of IQ data. In addition, the gym has a built-in GUI system for setting scenarios and parameters without changing the code as well as the ability to support outside packages, such as Mushroom RL. Additionally, prior works discussed in this section only allow for pre-set scenarios, whereas our tool allows for scenario customization. These additions separate our tool from others, emphasizing an easy-to-use environment for experimentation and testing.

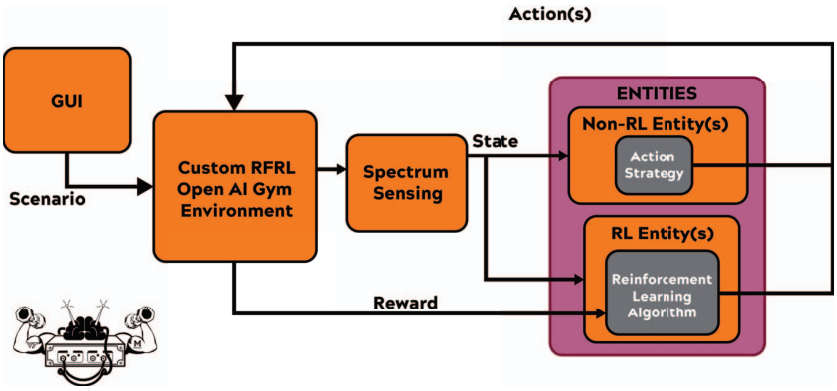


Fig. 1: The RFRL Gym framework, illustrating how user-defined scenarios and reinforcement learning algorithms interact with the gym environment. This framework facilitates simulating an RF-based reinforcement learning interaction loop of the RL agent taking in states (e.g. sensing results) and rewards (e.g. receiver performance) and providing to the gym its determined next action (e.g. transmitter tuning).

III. The RFRL Gym Framework

A high-level overview of the structure of the RFRL Gym is provided by Figure 1. The framework consists of a series of interconnected modules that function together in order to simulate RF spectrum scenarios. These modules follow the Open AI Gym API to ensure the RFRL Gym is compatible with third party gym-compatible RL software through its predefined functions and methods for a Reinforcement Learning agent and the environment to communicate. As a result, the RFRL Gym is able to exploit the "RL Cycle" training protocol. This protocol is a cyclical passage of information between the Gym and RL agent over a discrete set of training episodes, with each episode consisting of a number of time-steps. The cycle begins at the beginning of each time-step when the RL agent performs an action in the custom-made RF environment. This action may be to occupy a channel or to not transmit in the next timestep. At this point, Non-RL Entities will also select an action. Based on the actions of the Non-RL Entities, the RL Agent's action is evaluated and assigned a reward. This reward and the new observation space are made accessible to the RL Agent for training. This process repeats for all time-steps. The remainder of this section describes the individual components of the RFRL Gym outlined in Fig. 1.

A. Non-Player Entities

Several non-player entities are implemented in our environment. Non-Player refers to an entity whose decision-making is not controlled by the primary Reinforcement Learning algorithm. They will herein be referred to as "Entities". These Entities are crucial for testing as they simulate physical interactions between signals and replicate real signals in the environment. There are five entities that have been implemented into our code base at this point, as described in Table II.

TABLE II: Non-player entity types within the RFRL Gym environment that can be included in scenario files, with descriptions and real-world use cases

Entity Type	Description	Real-world Example
Constant	Remains in one channel	T.V. or Radio
Fixed Hopper	Moves according to a user-defined pattern	Bluetooth
Stochastic Hopper	Channel selected based on a set of user-defined probabilities for each channel	N/A
Agile Hopper	Moves to an empty channel. Has perfect knowledge of other entities in the bandwidth. If multiple channels are empty, it selects one with uniform probability. Most similar to the RL agent but without the RL.	CR with DSA
Simple Jammer	Has perfect knowledge of other entities in the environment, and moves to an occupied channel based on uniform distribution. If no channels are occupied, it moves to an unoccupied channel.	Intelligent Adversary
Custom	Created by users of our codebase	N/A

B. RL Entities

The purpose of the RFRL Gym is to train RL agents for RF applications. Users can run simulations with their own custom RL algorithms or those built by third parties. Multiple modes have been implemented for the observation and reward to enable a diversity of scenarios on which to train RL agents.

1) Reward Modes: The two reward types currently implemented in the environment for the RL agent are DSA reward and Jamming reward. DSA mode can be used to simulate a situation where an RL agent wants to avoid interference, either by avoiding a channel with bad channel conditions or a channel with other users and/or adversaries. Applications include commercial communications as well as Electronic Warfare (EW). Alternatively, Jamming mode allows for the

simulation of situations where the agent wants to prevent others from successful communications and use of the RF spectrum by acting as an adversary. This mode is also applicable to EW. The optional dampening factor was added to consider the computational cost of switching to model the desired real-world behavior of energy conservation, where the ideal behavior is to move only when necessary. This is especially important for small, energy-constrained devices, such as IoT devices.

- **DSA Reward** - Rewards the agent for avoiding channels containing other entities. The agent is awarded a 1 for no collision and -1 for a collision.
- **Jamming Reward** - Rewards the agent for occupying the same channel as a target entity in the environment. The entity to be jammed is selected by the user. The agent is awarded a 1 for a collision and -1 for no collision.
- **Dampening Factor** - The dampening factor will increase the reward if the agent stays in the same channel and reduce the reward if it changes channels. The agent is awarded an additional 0.5 for no movement made and -0.5 for movement made.

2) Observation Modes: The gym contains two different observation modes for the agents: detect and classify. These modes determine what information is sent to the agent as the current state of the environment. In detect mode, the agent only knows if a channel is occupied or not. In this mode, the RL agent does not know what type of entities or how many entities occupy a single channel. In classify mode, the agent receives information about the channel location, type, and number of entities (i.e. which entities are in which channel or if there is a Multi-Entity collision in a channel). These modes are realized through spectrum sensing, where occupancy can be determined across each frequency based on the amount of energy present. If the amount of energy is above the noise floor, it is an indication that an entity is transmitting over that channel and/or the channel has poor conditions.

C. Rendering Modes

The RFRL Gym provides two rendering modes to visualize simulations, the state of entities and their interactions at each time-step, as well as results: terminal rendering mode and PyQt rendering mode. Both modes display the location of the entities within the different channels across time as well as the reward for each time step and cumulative reward across each episode.

1) Terminal Rendering: The terminal rendering mode sends data to be displayed in Standard Output using ascii text. Therefore, the data can be directed to the terminal or redirected to a file. Depending on whether the gym is in classify or detect mode, the visualization will change. In detect mode, occupied channels are represented with a 1, while unoccupied channels are represented with a 0. In classify mode, the location of each entity is shown with its corresponding entity number. Lastly, the terminal mode runs efficiently, making it suitable for more complex scenarios where a faster output is desired.

2) PyQt Rendering: Enabled by the *pyqtgraph* package, the PyQt rendering mode displays the same data as the terminal mode but with more detailed visuals, making it easier for users to visually understand how entities move across time and frequency within the RF spectrum. This mode also displays a plot of the cumulative reward for each time-step in an episode. An example of visual results for this mode can be seen in Section IV.

D. Scenarios

A scenario is the set of parameters that define the environment in which the agent is placed to perform its learning. The parameters fall into three categories: *environment*, *render*, and *entities*. Scenarios are defined in customizable JSON files (”scenario files”) that are fed into the RFRL Gym.

1) Environment: This section defines the learning parameters needed to prepare the environment, such as the number of channels available and number of time steps. Additionally, the observation mode can be set to the “classify” or “detect” mode, and similarly, the reward mode can be set to “dsa” for dynamic spectrum access, or “jam” for jamming mode. For jamming mode, the target entity is also set.

2) Entities: Here, the user will determine which entities will exist in the environment. An entity is designated by a user-defined name, which will appear in the rendered output. Within an entity, various parameters can be set. For the gamified, basic mode, these parameters include the channel(s) that the entity can occupy, the on/off transmission pattern(“on” referring to a non-idle state, “off” referring to idle), and the first and last steps at which the entity transmits. For the advanced mode containing IQ, the entities’ modulation type and order, center frequency, bandwidth, start time, and duration can also be set.

3) Render: This section determines whether the render is displayed within the terminal, or if it utilizes the render written using the *pyqtgraph* library. The render’s speed, in frames per second, and the number of steps that the render displays at a time are also included options.

To improve the user experience, a Scenario Selection GUI was created which enables the modification and creation of scenario files without interfacing with a JSON file directly. This simplifies the process of creating scenario files and allows non-software-oriented users to produce scenarios. Fig. 2 shows the layout of the GUI.

IV. Simulation Results and Discussion

In this section, multiple representative wireless communications scenarios are presented in order to highlight the capabilities of the RFRL Gym for test and evaluation of RL agents. For all scenarios, the environment consists of 10 wireless channels, assumes learning episodes consisting of 100 time-steps, 100 training episodes, a Q-learning agent balancing exploration and exploitation through an epsilon greedy strategy, and the use of the MushroomRL software package for training and evaluating the agent.

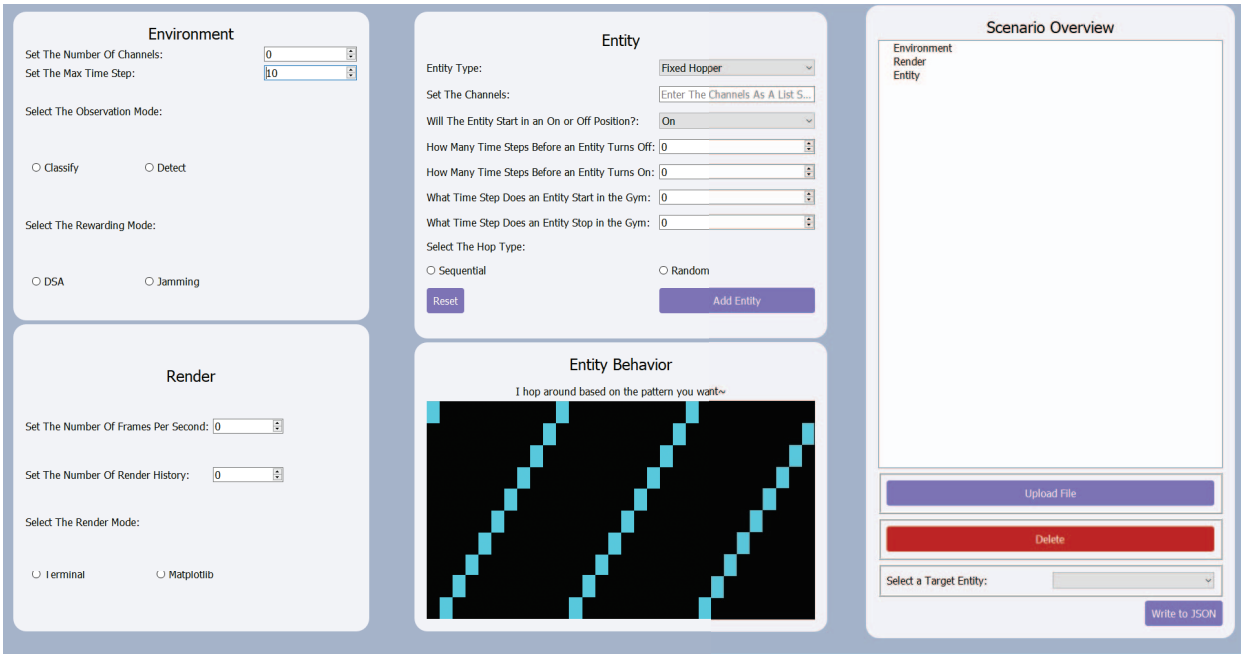


Fig. 2: GUI-based scenario generator where users can set up the environment, render, and entities.

TABLE III: Scenario settings with entities, reward, dampening, and observation modes defined

Scenario	Entities	Reward	Dampening	Observation
1	1 Constant	Jamming	No	Classify
2	1 Fixed Hop	Jamming	No	Classify
3	1 Fixed Hop 1 Constant 1 Stochastic	DSA	No	Detect
4	1 Fixed Hop 1 Constant 1 Stochastic 1 Agile	DSA	No	Classify

A. Scenario 1: Single Entity Jamming

This is a simple scenario that was designed to be solved easily and validate that the RFRL Gym performs as expected and yields accurate results. It consists of a single Constant Entity that is to be jammed. As expected, a Q-Learning agent was able to converge to an optimal policy quickly within the first five episodes, as demonstrated in Fig. 3.

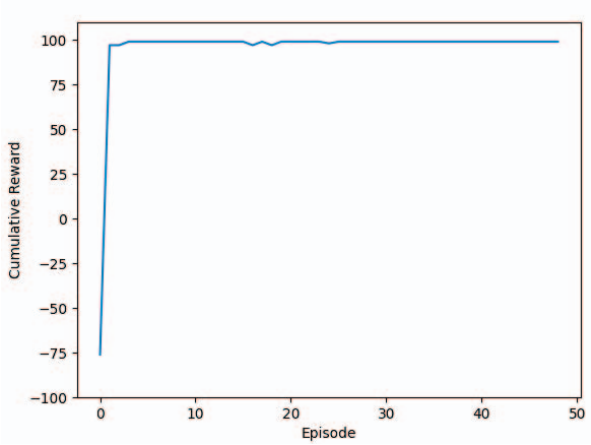


Fig. 3: Cumulative reward results for Scenario 1, showing optimal behavior by converging to the optimal reward of 100

B. Scenario 2: Hop Frequency Jamming—Non-Markovian

This scenario consists of a single Hopping entity with a fixed hopping pattern. It spends 2 time-steps in channel 4, 1 time-step in channel 5, and repeats. The object of this scenario is to produce a *Non-Markovian* problem, meaning the problem does not follow the Markov Decision Process which holds the assumption of the Markov property where the evolution of the Markov process depends only on the current state. In this case, the current state is not always sufficient information for the next time step, hence the agent cannot achieve an optimal policy. In order for the agent to jam the Hopper perfectly, it must have some sense of the previous location of the hopper. For example, if the Hopper is in channel 4, the agent would need to know whether the Hopper was in channel 4 or 5 in the previous time-step. Without that information, our Q-Learning agent can not predict whether the entity will stay or move. Therefore, it will converge to a sub-optimal policy.

Optimally, the agent would jam the Hopper in all 100 time-steps. In Fig. 4, the agent can be seen choosing not to transmit in the second and third time-steps of the hopping pattern. This is because it was unable to consistently jam the Hopper during

the training process due to Non-Markovian behavior. These incorrect guesses yield a reward of -1.0 but non-transmission yields a reward of 0.0. Thus, the agent elects to accept the suboptimal reward of 33 and not risk the negative reward. Calculations below show this suboptimal convergence to 33, rather than the optimal performance of 100.

$$\textbf{Sub-Optimal Reward: } 100\Big(\frac{1}{3}(1)+\frac{2}{3}(0)\Big)=33$$

The results produced by the RFRL Gym were aligned exactly with the expected outcome. This scenario verifies that our Gym will perform as expected even when an agent is unable to achieve an optimal policy. The dependability of the Gym's results make it a valuable tool for formulating and testing hypotheses.

C. Scenario 3: Solvable Multi-Entity DSA

This scenario contains a Fixed Hopper Entity moving through all channels non-sequentially, a Constant Entity in channel 5, and a Stochastic Hopper that moves to either channel 0 or 9 with equal likelihood. This scenario exhibits the multi-Entity capability of our environment. As shown in Fig. 5, the Q-Learning agent can handle avoidance of multiple Entities and achieve an optimal policy. The RFRL Gym's gamified rendering functionality enables easy visualization of the agent's optimal policy. The absence of collisions after convergence can be seen easily by the lack of red blocks in Fig. 5b, as opposed to before convergence where collisions occurred frequently in Fig. 5a.

A notable conclusion that can be drawn from this result as well is that a Q-Learning agent in DSA mode is able to achieve an optimal policy when entities behave according to a stationary distribution, meaning the probability in which it chooses each channel stays constant. However, this is not true for Agile Entities as will be shown in Scenario 4.

D. Scenario 4: Unsolvable Multi-Entity DSA

This scenario is similar to Scenario 3. The single difference is the addition of an Agile Entity moving across all channels. The Stochastic Hopper Entity still behaves according to a stationary distribution between each of its channels since it only moves between the two channels with the highest probabilities (channel 0 and 9), which the Q-learning agent learns to avoid. On the other hand, the addition of the Agile Entity confuses the RL agent such that it is unable to definitively predict where it will go next because the Agile Entity has no deterministic pattern due to its non-stationary, time-varying behavior caused by randomly choosing an empty channel according to where other entities occupy other channels. Therefore, the RL agent will occasionally collide with the Agile Entity, as can be seen in red in Fig. 6. As a result, the agent is unable to converge to an optimal policy, as can be seen by the noisy, non-stabilizing results in Fig. 7. This result implies that more advanced Reinforcement Learning algorithms such as Deep Q Learning are necessary to solve problems with non-stationary, time-varying behavior.

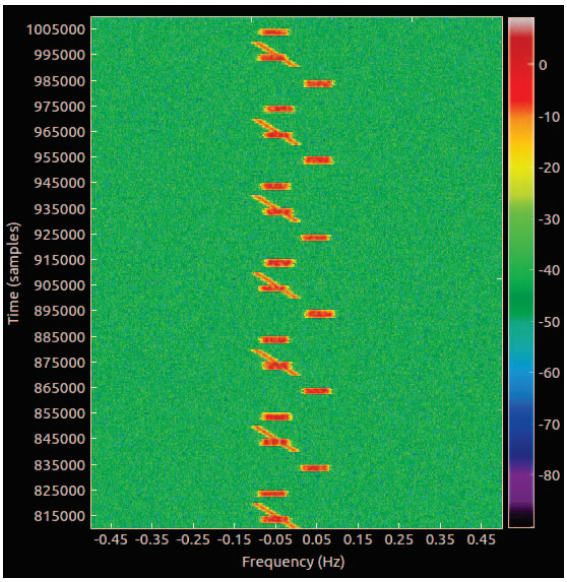


Fig. 4: PyQt render results for scenario 2, shown in advanced mode where IQ data is viewed. The diagonal signal is the RL agent, while the horizontal signal is the non-RL entity the agent is targeting to jam.

V. Future Additions

A. Multi-Agent RL functionality

In our gym, we are currently moving towards incorporating multi-agent functionality through a separate, multi-agent-specific environment. As realistic dynamic spectrum access scenarios may involve highly contested bandwidths, it is important to consider and provide solutions for cases where several RL agents interact in the same set of channels. As a result, incorporating Multi-Agent Reinforcement Learning (MARL) learning strategies is crucial to the success of our agents in realistic scenarios. These scenarios may be divided into two separate classes - cooperative and adversarial.

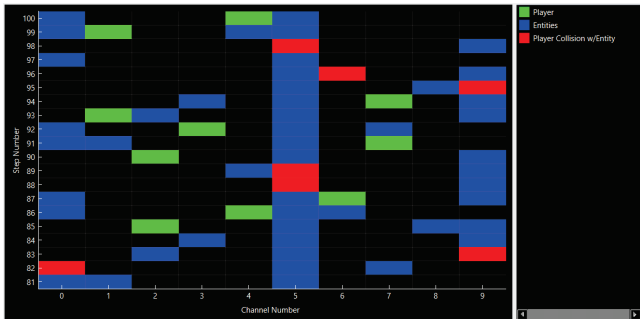
1) Cooperative: Cooperative MARL agents attempt to coordinate with other agents in order to maximize total uncontested access to the spectrum.

2) Adversarial: Adversarial agents attempt to prevent other agents from accessing the spectrum by intentionally broadcasting on the channel already occupied by its target or one the target is predicted to hop to in the future.

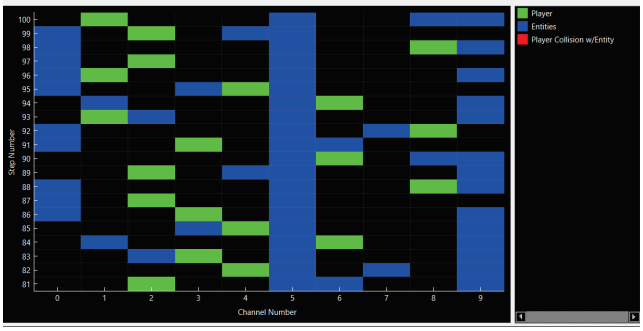
Incorporating MARL into the RFRL Gym could be made possible by utilizing MARL libraries compatible with OpenAI Gym or similar to OpenAI Gym. One such package worth investigating is PettingZoo [21], a MARL library designed with OpenAI Gym's simple API in mind in addition to the ability to create custom MARL environments.

B. Physical signal integration and Hardware compatibility

In real-world scenarios, perfect knowledge of the environment is unlikely. Thus, the RFRL Gym will include functionality for converting raw signal data into state information that



(a) Render results showing non-convergent behavior before learning, where the RL agent collides with other entities within the environment



(b) Render results showing convergent behavior after learning, where the RL agent successfully avoids other entities within the environment

Fig. 5: Scenario 3 render results, showing before and after convergence behavior, in the gamified detect mode.

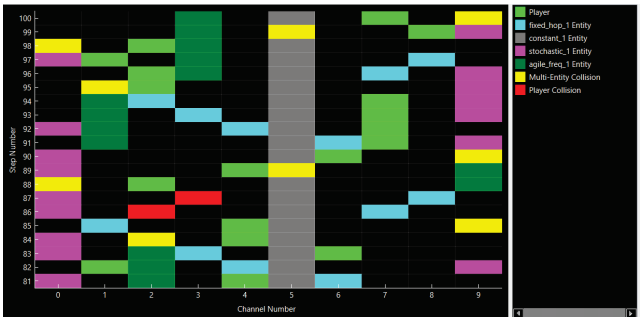


Fig. 6: Render results showing non-convergent behavior due to collisions with an Agile Entity. Render is in classification mode to show that the Agent is colliding with the Agile Entity.

can be processed by RL agents. Convolutional neural networks (CNNs) are currently being developed to identify signal types based on a signal's structure [22]. The network will take in raw waveform IQ data from an RF channel and classify the signal. This result can be integrated into the observation space of RL agents to target specific entities. Additionally, we are working towards developing the hardware functionality for more realistic software simulation such that signals experience realistic channels conditions in which fading and multi path

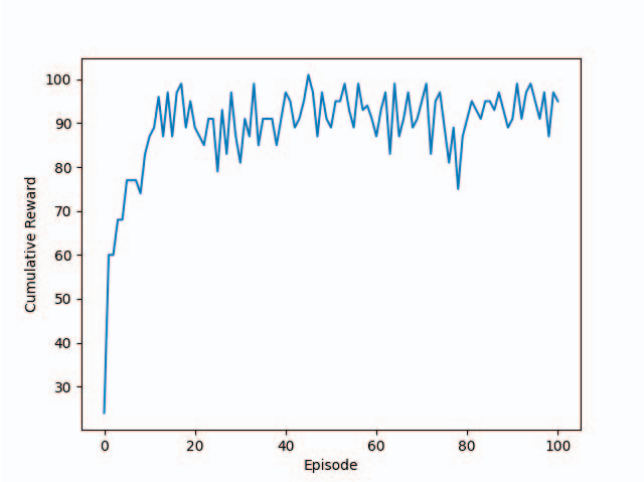


Fig. 7: Cumulative reward results for Scenario 4, showing suboptimal behavior due to the Agile Entity

propagation exist.

C. GUI expansion

The next iteration of the GUI intends to connect the visualizations rendered with PyQt to the pre-existing Scenario Selection GUI so that the user can run simulations entirely from the Scenario Selection GUI. Additionally, we intend to implement tabs that will guide the user through the scenario setup process, rather than having them interact with all of the steps simultaneously. Also, there are plans to implement an auto-sync feature. This would mean that whenever a new rewarding type, observation mode, or non-player entity is added, the visual output will automatically update to allow the user to see their changes.

VI. Conclusion

A sharp rise in wireless usage has caused significant congestion in the RF spectrum. To remedy this, the predictive capabilities of Reinforcement Learning are increasingly being exploited in Cognitive Radio applications. Therefore, the existence of a resource like the RFRL Gym is imperative to train algorithms in a representative environment.

Tools that serve similar purposes have been created, but the RFRL Gym possesses various distinct advantages and includes more comprehensive functionality. Firstly, the simulation functionality in our environment is abstracted, so technical expertise in Wireless Communications is not necessary to use the Gym. Secondly, our codebase is compatible with external libraries including Mushroom RL and Stable Baselines. A third major advantage of our Gym over comparable resources is the GUI which makes the operation of the environment simple. Though the RFRL Gym currently possesses extensive functionality, additions are still being made. Future iterations of the environment will show MARL functionality, physical signal integration, radio hardware compatibility, and simulations run entirely from the GUI.

All in all, we hope our work will aid in developing new reinforcement learning strategies for cognitive radio applications. By providing a powerful tool with a low barrier of entry, we seek to encourage the exploration of new pursuits in the rapidly growing field of RFRL.

REFERENCES

[1] P. Tilghman, "Will rule the airwaves: A darpa grand challenge seeks autonomous radios to manage the wireless spectrum," *IEEE Spectrum*, vol. 56, pp. 28–33, 06 2019.

[2] "Summary of ctia's annual wireless industry survey," https://api.ctia.org/wp-content/uploads/2022/09/Summary-of-CTIAs-Wireless-Industry-Survey-2022.pdf.

[3] "Spectrum allocation in the united states 2022," https://api.ctia.org/wp-content/uploads/2022/09/Spectrum-Allocation-in-the-United-States-2022.09.pdf.

[4] J. Mitola and G. Maguire, "Cognitive radio: making software radios more personal," *IEEE Personal Communications*, vol. 6, no. 4, pp. 13–18, 1999.

[5] R. Awati, "What is cognitive radio (cr) and how does it work?" Jun 2021. [Online]. Available: https://www.techtarget.com/searchnetworking/definition/cognitive-radio

[6] M. Khatib, *Advanced Trends in Wireless Communications*. Rijeka: IntechOpen, Feb 2011. [Online]. Available: https://doi.org/10.5772/655

[7] B. P. Lathi and Z. Ding, *Direct Sequence Spread Spectrum*. Oxford University Press, 2019, pp. 702–707.

[8] A. G. Fragkiadakis, E. Z. Tragos, and I. G. Askoxylakis, "A survey on security threats and detection techniques in cognitive radio networks," *IEEE Communications Surveys and Tutorials*, vol. 15, no. 1, p. 428–445, 2013.

[9] R. S. Sutton and A. G. Barto, *Reinforcement learning: an introduction*. The MIT Press, 2018.

[10] S. Singh and A. Trivedi, "Anti-jamming in cognitive radio networks using reinforcement learning algorithms," in *2012 Ninth International Conference on Wireless and Optical Communications Networks (WOCN)*, 2012, pp. 1–5.

[11] F. Slimeni, Z. Chtourou, and A. B. Amor, "Reinforcement learning based anti-jamming cognitive radio channel selection," in *2020 4th International Conference on Advanced Systems and Emergent Technologies (IC_ASET)*, 2020, pp. 431–435.

[12] S. Machuzak and S. K. Jayaweera, "Reinforcement learning based anti-jamming with wideband autonomous cognitive radios," in *2016 IEEE/CIC International Conference on Communications in China (ICCC)*, 2016, pp. 1–5.

[13] Y. Huang, C. Xu, C. Zhang, M. Hua, and Z. Zhang, "An overview of intelligent wireless communications using deep reinforcement learning," *Journal of Communications and Information Networks*, vol. 4, pp. 15 – 29, 2019.

[14] A. Zubow, S. Rosler, P. Gawłowicz, and F. Dressler, "Grgym: When gnu radio goes to (ai) gym," 2021.

[15] A. Zubow, S. Roesler, P. Gawlowicz, and F. Dressler, "Grgym: A playground for research on rl/ai enhanced wireless networks," in *European Wireless 2022; 27th European Wireless Conference*, 2022, pp. 1–4.

[16] J. Jagannath, K. Hamedani, C. Farquhar, K. Ramezanpour, and A. Jagannath, "Mr-inet gym: Framework for edge deployment of deep reinforcement learning on embedded software defined radio," *WiseML*, 2022.

[17] P. Gawlowicz and A. Zubow, "ns-3 meets openai gym: The playground for machine learning in networking research," in *ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, November 2019. [Online]. Available: https://bits.informatik.hu-berlin.de/ zubow/gawlowicz19_mswim.pdf

[18] "Gnu radio - the free & open software radio ecosystem," https://www.gnuradio.org/, 2023.

[19] C. D'Eramo, D. Tateo, A. Bonarini, M. Restelli, and J. Peters, "Mushroomrl: Simplifying reinforcement learning research," *Journal of Machine Learning Research*, vol. 22, no. 131, pp. 1–5, 2021. [Online]. Available: http://jmlr.org/papers/v22/18-056.html

[20] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, "Stable baselines," https://github.com/hill-a/stable-baselines, 2018.

[21] J. Terry, B. Black, N. Grammel, M. Jayakumar, A. Hari, R. Sullivan, L. S. Santos, C. Dieffendahl, C. Horsch, R. Perez-Vicente *et al.*, "Pettingzoo: Gym for multi-agent reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 15 032–15 043, 2021.

[22] L. J. Wong, W. C. Headley, and A. J. Michaels, "Specific emitter identification using convolutional neural network-based iq imbalance estimators," *IEEE Access*, vol. 7, pp. 33 544–33 555, 2019.