A Multi-Agent Reinforcement Learning Testbed for Cognitive Radio Applications

Sriniketh Vangaru*, Daniel Rosen*, Dylan Green*, Raphael Rodriguez*, Maxwell Wiecek*,
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

I. Abstract

Technological trends show that Radio Frequency Reinforcement Learning (RFRL) will play a prominent role in the wireless communication systems of the future. Applications of RFRL range from military communications jamming to enhancing WiFi networks. Before deploying algorithms for these purposes, they must be trained in a simulation environment to ensure adequate performance. For this reason, we previously created the RFRL Gym: a standardized, accessible tool for the development and testing of reinforcement learning (RL) algorithms in the wireless communications space. This environment leveraged the OpenAI Gym framework and featured customizable simulation scenarios within the RF spectrum. However, the RFRL Gym was limited to training a single RL agent per simulation; this is not ideal, as most real-world RF scenarios will contain multiple intelligent agents in cooperative, competitive, or mixed settings, which is a natural consequence of spectrum congestion. Therefore, through integration with Ray RLlib, multi-agent reinforcement learning (MARL) functionality for training and assessment has been added to the RFRL Gym, making it even more of a robust tool for RF spectrum simulation. This paper provides an overview of the updated RFRL Gym environment. In this work, the general framework of the tool is described relative to comparable existing resources, highlighting the significant additions and refactoring we have applied to the Gym. Afterward, results from testing various RF scenarios in the MARL environment and future additions are discussed.

***Index Terms*—multi-agent reinforcement learning, wireless communications, dynamic spectrum access, OpenAI Gym**

II. Introduction

A. Radio Frequency Reinforcement Learning

In the wireless spectrum, the pre-allocation of specific frequencies by the FCC [1] and the increasing usage of the radio frequency (RF) range of the spectrum, such as in mobile networks [2], have motivated the use of cognitive radios (CR). CRs use dynamic spectrum access (DSA) to adaptively accommodate the quantity of signals continuously being sent and received. However, to enable these technologies to better manage the constantly fluctuating nature of the RF spectrum, Radio Frequency Reinforcement Learning (RFRL) technologies have been developed [3]. These make prediction-based decisions regarding the frequencies a CR will transmit on as opposed to being purely reactive, and they have significantly improved the ability of a CR to avoid interference in the spectrum [4]–[6].

With the expansion of 5G and the early development of 6G wireless communication, demand for deployment-ready RFRL algorithms has increased dramatically; the RFRL Gym was created as a simulation environment and testbed to accelerate their development [7]. However, the original version of the RFRL Gym included exclusively single-agent training scenarios. In the majority of real-world scenarios, multiple signals will exist in a wireless spectrum at one time, such as in military contexts where spectrum sharing and radio jamming are an essential concern [8] or in everyday products (e.g., mobile phones) whose ubiquity and usage of the spectrum are continuously increasing [9]. CRs that act in the same space as other "intelligent" radios are dependent upon Multi-Agent Reinforcement Learning (MARL) methodologies to dependably achieve optimal results, as using a single-agent model will not properly encapsulate all of the dynamics of a given real-world situation; as a result, these are currently being rapidly researched and developed [10]–[12]. Therefore, to increase the fidelity and realism of the training process, MARL scenarios were deemed a necessary addition to the testbed. This paper describes this primary addition to the RFRL Gym along with IQ data generation and rendering improvements for a more realistic, friendly user experience.

B. Multi-Agent Reinforcement Learning

Multi-Agent Reinforcement Learning (MARL) involves reinforcement learning scenarios in which multiple RL agents are acting and/or being trained simultaneously. In the context of the RFRL Gym, each agent is analogous to a CR device. Consider Fig. 1, in which 4 distinct co-channel entities can be seen during the same time period. Assume these signals are each being transmitted by an independent CR (a frequently-encountered scenario). In response to additional transient or persistent signals in the spectrum, the CRs may have to alter their transmission frequencies to avoid interference. Under the assumption that these CRs utilize RL to determine subsequent frequencies, this scenario is considered MARL. Such problems will become even more common as the expansion of IoT

979-8-3315-0805-0/25/$31.00 ©2025 IEEE

brings an increase in the number of devices that utilize the wireless spectrum, further emphasizing the need for programs such as our RFRL Gym to play a pivotal role in solving these problems.

In MARL, the reward functions of these agents are related to the actions of the other agents, since those actions would contribute to modifying the environment and make it non-stationary [13]. Agents can be encouraged to act cooperatively, competitively, or a mixture of the two. Independently from that categorization of situations, the MARL algorithms themselves can be centralized or decentralized. In a centralized context, a single model is used to determine the actions of multiple agents, whereas in a decentralized context, each agent is associated with its own model and determines its action exclusively. Optionally, decentralized agents can communicate with each other in some manner—such as by sharing observations, joint rewards, or joint actions—which is a quality known as "networked agents" [13]. Our RFRL Gym is able to support all combinations of cooperative and competitive situations due to easy modification of scenarios, allowing for MARL research in RFRL to be further explored.

C. Code Availability

The code and data produced during the process of conducting this research are available at https://github.com/vtnsi/rfrl-gym.

III. Background

A. Related MARL

Machine learning techniques have appeared in various facets of the RF Field [14]–[16]. In particular, DSA has become a focus of RL research, both with singular agents [17] and for multiple agents [18]. MARL methods have shown excellent performance in both cooperative [19] and adversarial [20] scenarios. In recognition of these achievements, research efforts for MARL in RF have swelled [11], [12], [21], [22]. Therefore, the addition of MARL functionality to the RFRL Gym was essential.

B. Related Training Environments

Prior works [7], [23]–[25] describe tools for training RL algorithms for the wireless communications space, including the multi-agent RFRL Gym that we are introducing. Table I displays the differences between these environments.

Flexible Scenario Design: Ability to easily customize entities and agents present in the training environment. [23] scenarios can be customized, but each requires a non-trivial implementation of a scenario class for each testing scenario. Similarly, [24] requires a custom C++ interface to be written for each scenario. [25] provides a wide variety of premade scenarios of incumbents (similar to entities). Though not customizable, they are relatively comprehensive in conjunction. The RFRL Gym's scenarios can be easily and finely customized with JSON scenario configuration files.

RL Package Compatibility: Ability to interface with external RL libraries for algorithms and optimization methods.

TABLE I: Comparison of existing OpenAI Gym tools for RL in wireless communications, modified from [7].

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		✓	✓	

[23], [24], and the RFRL Gym utilize OpenAI's Gym API [26]. Because of this, these environments interface with various libraries including Stable-Baselines [27], PettingZoo [28], etc. [25] is not similarly integrated, as models are primarily intended to be trained offline on data collected from the environment.

Spectrum Sensing Capabilities: Various methods of observing the state of the spectrum. This includes methods of detecting and classifying entities.

Multi-Agent Capability: The newest version of the RFRL Gym environment features the addition of multi-agent capabilities, meaning that it can support the concurrent training of multiple intelligent agents in one scenario. This also includes centralized and decentralized information structures, as well as competitive, cooperative, and mixed settings for scenarios.

Ease of Use: Tool is designed to simplify the training and analysis process of RL algorithms, especially if this functionality is accessible to a wide audience. [23] and [24] both require understanding of an external resource (GNU Radio and ns-3 respectively). GNU Radio is not officially supported on Windows and [23] does not offer rendering. [24] requires C++ implementation to interface with the ns-3 environment. [25] does not offer rendering and access is limited to researchers affiliated with the US Department of Defense. The RFRL Gym is designed to be accessible to both those new to wireless communications and those with extensive experience in the field. The RFRL Gym features a GUI for easy scenario generation and a variety of rendering options for data visualization.

Hardware Compatibility: The training environment's ability to interface with radio hardware for real-time information collection. [23] and [24] both utilize external resources for

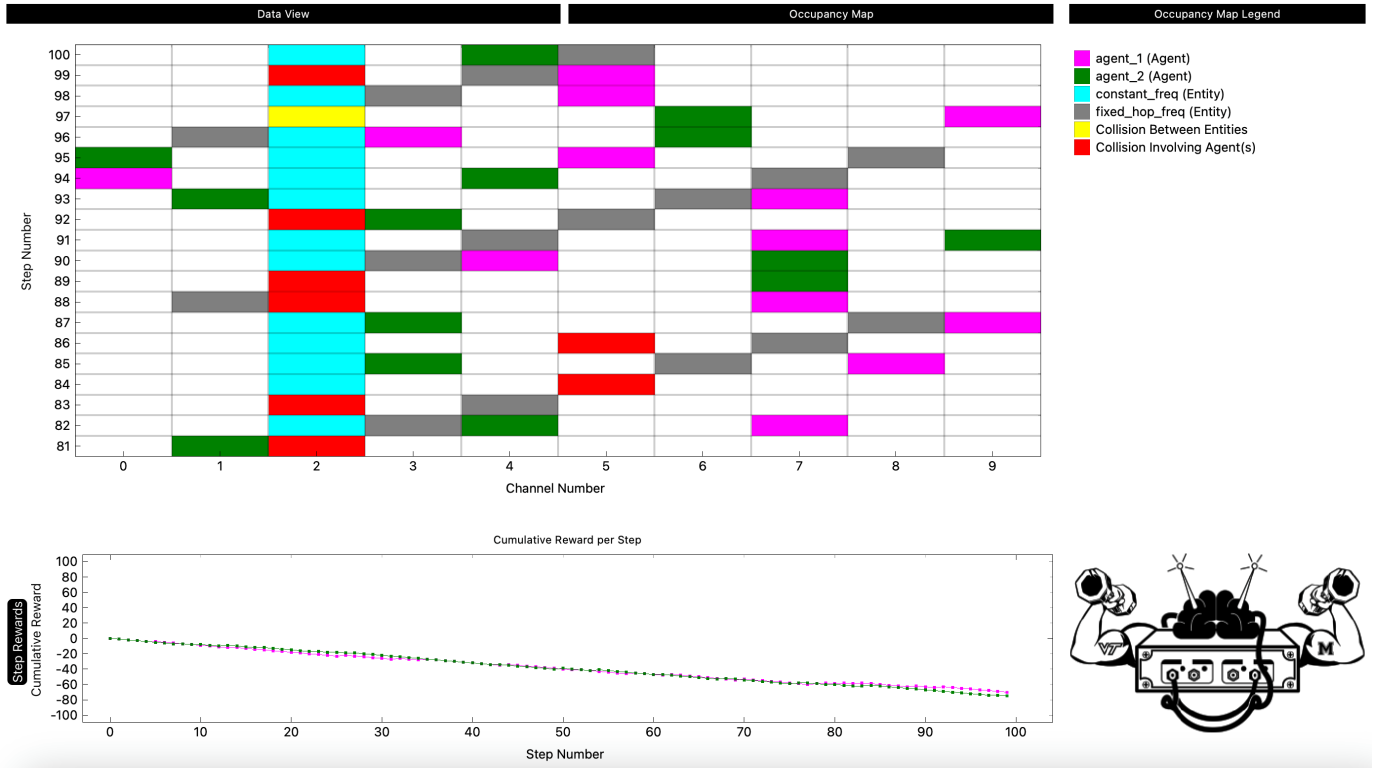

Fig. 1: A rendering of an abstracted RF spectrum in the multi-agent RFRL Gym.

wireless spectrum simulation, GNU Radio and ns-3 respectively. Both of these resources can interface with hardware for real-time spectrum sensing or simulate the wireless spectrum with software. [25] uses fiber optics to emulate the wireless spectrum. So, although its resources are accessed remotely by users, no real radios are involved. However, because [25] adopts the OpenRAN Gym framework [29], containerized applications can easily be ported from Colosseum to hardware-interfacing training environments. Currently, all of the RFRL Gym's spectrum simulation is software-driven using open-source Python modules but the addition of hardware compatibility is in progress.

Relative to comparable tools, the RFRL Gym offers a uniquely accessible workflow. The environment features deeply customizable scenarios for bespoke training executions. With human-readable information from these scenario files, IQ data is generated using abstracted information so a limited understanding of wireless communications is necessary for users. These scenario files can be produced using an easy-to-use graphical user interface, as shown in [7]. Overall, without the burden of the need for extensive knowledge of wireless communications, researchers can use this tool to incrementally develop an understanding of concepts such as spectrum sensing, DSA, and signal modulation. The robust visualization and efficient abstraction make the RFRL Gym an effective educational tool for those new to the field of RF. These features distinguish our tool from others as an accessible resource for experimentation and testing. However, it shares its more technical features and functionality with powerful spectrum simulation tools, such as GNU Radio and ns-3, making the RFRL Gym an appropriate tool for experts as well.

IV. The RFRL Gym MARL Framework

A. Workflow

In order to execute training simulations, users will first create a JSON scenario file. This file contains information about the agents and environment (detailed below in Section IV-D). Fig. 2 shows the exchange of information that occurs during a training simulation. Entities and agents are prompted by the gym to select actions at each timestep; entities follow the action pattern corresponding to the entity type provided in the scenario, and learning agents take actions based on their policies that were refined through a learning algorithm. This data is optionally converted into IQ data if using the single-agent environment. Then, the environment determines the occupancy of each channel and displays its results using the selected rendering method. Investigators can use this displayed information to assess the performance of their algorithms. Afterward, the agents' rewards and the environment's state are calculated, and these are recorded by the gym before being sent to the agents to begin the next timestep.

B. Comparison to Original RFRL Gym

This paper expands upon the existing single-agent version of the RFRL Gym, defined in [7]. Structurally, the new version

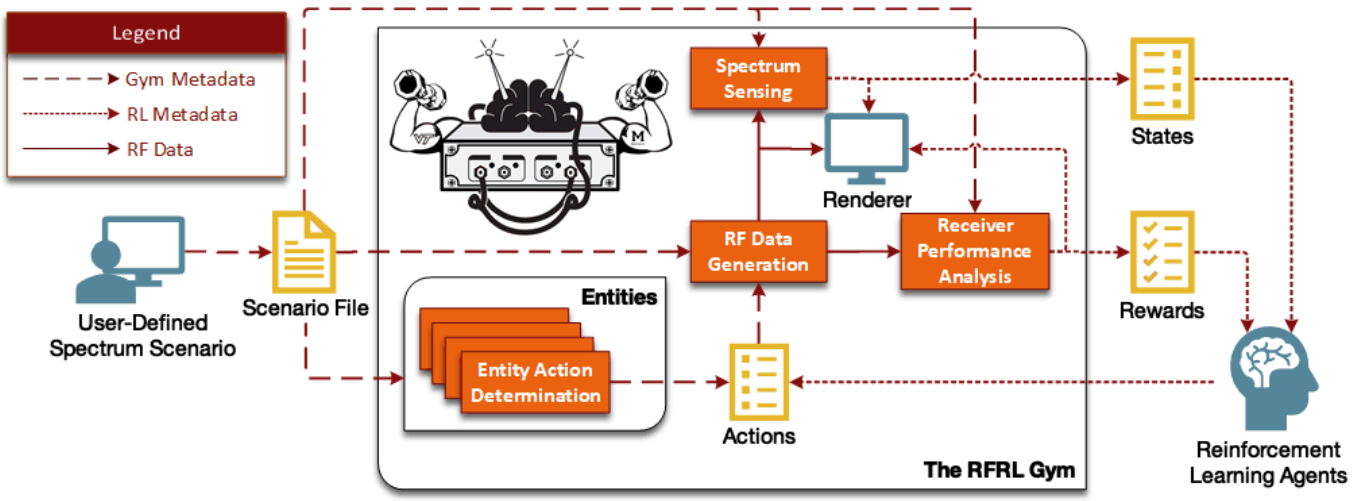

Fig. 2: Diagram of information flow in the multi-agent RFRL Gym.

of the RFRL Gym remains similar to that of the initial version. As can be seen in Fig. 2, information transfer between components remains consistent in the MARL environment, abiding by the classic reinforcement learning cycle.

The original single-agent environment will still exist in the package with its previous functionality. One addition made to this environment is a separate single-agent environment class that incorporates software-generated IQ data to better consider real spectrum dynamics, described further in Section IV-F. Another primary modification made to the original environment is the upgrade to the Farama Foundation's Gymnasium [30], which is an extension to OpenAI's Gym API [26]. This interface is compatible with external RL libraries such as Stable-Baselines3 [27] and PettingZoo [28]. This API is also leveraged by our MARL environment through inheritance from Ray RLlib's `MultiAgentEnv` class [31]. On top of compatibility with external libraries, Ray RLlib provides several high-powered functions such as remote cluster computation and cloud interfacing, whose computing power can be helpful when running larger-scale MARL simulations.

In the new multi-agent environment, the movement patterns that the non-learning RF entities have available in the RL environment have not been modified. Their functions and applications can be found in [7]. Each agent can track the locations of the entities with either the `detect` or `classify` observation mode, as in the original Gym. The observation and action histories of each agent are stored independently and provided to the user at each timestep, facilitating easy analysis of individual agents. The primary differences between the MARL RFRL Gym environment and the single-agent RFRL Gym environment exist in reward calculation, scenario file construction, data structures for observation and action histories, and rendering performance. The rest of this section enumerates and explains these features in further detail.

C. MARL Reward Calculation

Ray RLlib's RL training framework includes user-defined agent groups, which can, for instance, be defined explicitly using RLlib's `MultiAgentEnv.with_agent_groups()` function or implicitly using the `policy_mapping_fn` configuration property depending on the algorithm being used. The members of an agent group can view fellow members' actions at each timestep. The rewards for each agent in a group are correlated, and so these features permit centralized RL algorithms to be trained in our environment. This is useful as many algorithms use a very common approach involving centralized training with decentralized execution (CTDE) [32]. A centralized critic can even be implemented manually, such as by creating a function that shares observations [31].

Alternatively, agent groupings with one member per group may simulate a decentralized reward system. A standard use case for this method is decentralized training with decentralized execution (DTDE), a step up from independent Q-learning (IQL) in a range of scenarios, including mixed cooperative-competitive settings [33], [34].

The individual reward functions are given in Equations 1 and 2, where a is an agent's action and g is the set of actions from the agents in a group. The dynamic spectrum access (DSA) function rewards the agent for avoiding channels containing other entities or agents, and the jamming function rewards the agent for occupying the same channel as a particular entity in the environment, which is selected by the user. The reward function for each agent group is calculated as the sum of the individual rewards of each agent [31]. The mean reward, attributed to each agent in an agent group, can be accessed in the `results` dictionary returned by RLlib during training but its usage (compared to the usage of the group's reward) depends on the algorithm used. This is shown in Equation 3. As a result of this customizability of agent groups, the RFRL Gym is suitable for cooperative, competitive, and mixed training scenarios.

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

$$r_{jamming}(a) = \begin{cases} 1, & \text{collision with target entity} \\ 0, & \text{no transmission} \\ -1, & \text{transmitting elsewhere} \end{cases} \quad (2)$$

$$r_{mean}(g) = \frac{\sum_{a \epsilon g} r(a)}{|g|} \quad (3)$$

It is important to note that the amount of centralization that is involved in training depends mainly on the implementations of the algorithms themselves and not the RFRL Gym environment[1], and the agent groupings involved in the testing scripts are made to correspond with the form of centralization used. Additionally, a decentralized setup is the most common for practical scenarios with CRs in the RF spectrum, as centralization would imply a control unit that connects independent CR devices in the real world, which is not usually feasible. As a result, we only used the decentralized option of agent groups for our evaluations and did not conduct further testing with different groupings or reward function centralization. Though our testing scripts do still illustrate the use of RLlib's representation of groups/centralization (as our decentralized setup is just a special case of agent groups but with one agent per group), that was not the main focus of our demonstration of the RFRL Gym environment.

D. MARL Scenario Files

The definition of a scenario and its separate categories are stated in [7]. The only category that has changed is the `environment` dictionary in the JSON file, which controls all specific constraints that the multiple agents will learn under. Whereas our previous implementation only included the configurations for one agent, the `environment` section now contains an `agents` dictionary, which defines the number of agents in the environment and the observation mode, reward function, and target entity (if applicable) for each agent. For ease of use, we have also added a `comments` key-value pair to allow the user to add a description of the scenario within the JSON itself. This ensures that when the scenario is given to another analyst, it helps them understand the scenario's contents and purpose.

E. Rendering Enhancement

Previous rendering in the RFRL Gym only supported a single-agent environment for both PyQt rendering and terminal rendering, as explained in the previous paper [7]. With the enhancement of the RFRL Gym to a multi-agent environment, the major changes in rendering include visuals of cumulative reward changes for each agent at each time step in a training episode (or at each episode, depending on the user's configuration) and different colors to represent each learning agent or non-learning entity as it occupies a space in the radio frequency spectrum, as shown in Fig. 1. The same color that identifies a specific agent in the depiction of the spectrum will also be used for the corresponding agent on the cumulative reward graphs. The primary justification behind giving a different color scheme to each individual agent, similarly to how multiple entities in the single-agent gym would have their own color, is to easily identify which agents are learning their action policy effectively. The subclasses of our `MultiAgentRenderer` class can be modified to visually distinguish between different types of collisions (e.g., agent with agent, agent with entity, entity with entity) by changing the colors that represent such events.

F. IQ Data Generation

The RFRL Gym also now includes another single-agent environment in the form of a Python class (`RFRLGymIQEnv`) that uses software-generated IQ data to represent the locations of entities and the agent in the spectrum. In this environment, at each time step, the discrete actions selected by each entity and the agent are used to produce simulated IQ data. Modulation methods are available to users out of the box in both `LDAPM` and `Tone` classes. Once the data are created, the environment infers occupancy of each channel based on an energy threshold being met. This method improves the realism of scenarios as the non-deterministic nature of the wireless spectrum is included in the training.

V. Simulation Results and Discussion

A. Algorithms

Four multi-agent reinforcement learning algorithms were investigated in our evaluations. They were chosen due to being RLlib's only available algorithms with multi-agent functionality at the time of testing. For each, the `policy_mapping_fn` function in the algorithm's configuration maps each agent to its own policy so that, as mentioned earlier in Section IV-C, the agent groups simulate a decentralized setting. The algorithms are listed below.

1) Deep Q-Network (DQN): DQN is a widely-used RL algorithm that uses a neural network to approximate Q-Values [35]. The DQN implementation used for simulations uses an experience replay buffer to minimize recency bias, which prioritizes selecting more recent experiences. Though DQN is originally a single-agent learning algorithm, RLlib's multi-agent extension of DQN uses a prioritized replay buffer, designed specially for multi-agent environments, that allows the user to optimize how the prior sampled experiences are picked. DQN is typically well-suited for discrete action spaces [36].

2) Proximal Policy Optimization (PPO): PPO is another well-known, standard learning algorithm in single-agent RL systems, using an objective function and a surrogate objective function where the policy updates for the latter are constrained to remain relatively small [37]. This tends to ensure that

[1]Customized reward functions in the environment as mentioned in Section VI could also influence centralization, however.

learning is stable. Similarly to DQN, we used RLlib's implementation of multi-agent PPO.

3) Asynchronous Proximal Policy Optimization (APPO): APPO is an RLlib-implemented variant of PPO with asynchronous sampling of experiences, which allows it to be quicker (measured in real-world time) than PPO, though its performance is not always better than standard PPO [31].

4) Importance Weighted Actor-Learner Architecture (IMPALA): IMPALA is an algorithm that specializes in multi-task learning but can also handle multi-agent environments [38]. It is based on the "asynchronous advantage actor-critic" (A3C) algorithm and in fact provides a similar architecture to what APPO uses.

B. Evaluation Methods

In order to test our scenarios, we wrote a testing script for each algorithm to run and evaluate the effectiveness of the learned policies for the agents in any given scenario. Due to each algorithm containing a different number of episodes per training iteration, the method of algorithm evaluation is based on training episodes to maintain uniformity. The collective reward accrued by the agents in an episode will be plotted against the episode count for each algorithm, with the results of the algorithms' performances shown and discussed below in Section V-C.

C. Scenarios Tested

Here, we present a variety of scenarios which represent different states that learning agents and non-learning entities can take on in wireless communications situations, before discussing the results from training and evaluating the learning algorithms for the agents. For each scenario, the environment assumes a standard of 10 discrete frequency channels, as well as 100 sampled timesteps (i.e., calls of `env.step()`, where `env` is an instance of our multi-agent environment class) per episode of training. It should be noted that during training, depending on the algorithm used, the number of times that each sampled `step()` call is actually trained on varies and, as mentioned earlier, the number of episodes per iteration also varies. Additionally, as mentioned in Section IV-C, we use a fully decentralized system for every scenario, so for each algorithm, each agent is mapped to its own policy.

1) Scenario 1: Jamming Constant-Frequency Entities: As a baseline test of the ability of our multi-agent RFRL Gym environment to support learning algorithms that train the agents' policies, a simple scenario was created where each agent was given an entity to jam (i.e. transmit on the same channel as). There are four agents, where two have the `detect` observation mode and the other two use the `classify`, and all of them are using the `jam` reward mode. There are also 2 entities which constantly remain on one frequency channel each, separate from each other.

The environment successfully interfaced with the various aforementioned algorithms implemented by RLlib, and the results are displayed in Fig. 3. An episode reward of 400 indicates that all 4 agents consistently picked the optimal channel to transmit on; this is because each optimal action obtains a reward of 1 for that agent, and the rewards are simply summed up across all 4 agents and 100 timesteps in an episode to form the episode reward. *(Note: For the same reason, Scenario 5 also has a maximum episode reward of 400.)* As shown, all algorithms converged to an optimal policy for all agents, demonstrating the basic capabilities of our RFRL Gym environment involving having multiple agents effectively learn to complete a task in the RF spectrum. Evidently, DQN demonstrated faster convergence than the other algorithms, which is likely due to its general aptitude for discrete action spaces, with the benefit from this being compounded by only having a small number of actions—which are the 10 frequency channels to transmit on—available for each agent [36].

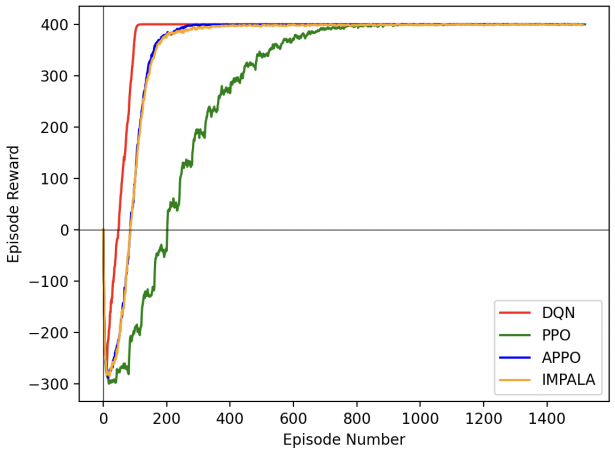

Fig. 3: Results from testing Scenario 1. To lessen the impacts of episode reward fluctuations upon the display, an exponentially weighted moving average (EWMA) with $\beta = 0.75$ was applied.

2) Scenario 2: Fixed-Hop Frequency Jamming: To incorporate slightly more realistic non-learning, transmitting entities compared to those in Scenario 1, we created a scenario which involves two agents trying to both jam a single entity that uses a linear hopping pattern. The results are shown in Fig. 4. Due to only having 2 agents, the maximum episode reward attainable in this scenario is 200. *Note: Due to the same reason, Scenarios 3 and 4 also have a maximum episode reward of 200.*

This scenario revealed more interesting results than Scenario 1, with the most noticeable difference being that APPO and IMPALA converged to an episode reward value of 0. After observing the agents' actions, it became clear that the two agents attempting to follow the fixed hopper entities were not able to learn the optimal actions to take—regardless of their observation mode—and instead opted to not transmit on any channel, giving a reward of 0, instead of repeatedly picking the wrong channel, which would give a reward of -1. A probable reason for this, considering that APPO and IMPALA share the same asynchronous sampling architecture for their replay buffer of experiences, is that the nature of the fixed-

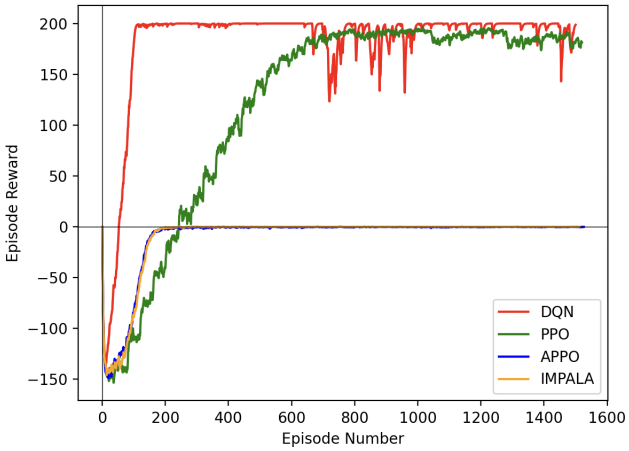

Fig. 4: Results from testing Scenario 2. An EWMA with $\beta = 0.75$ was used.

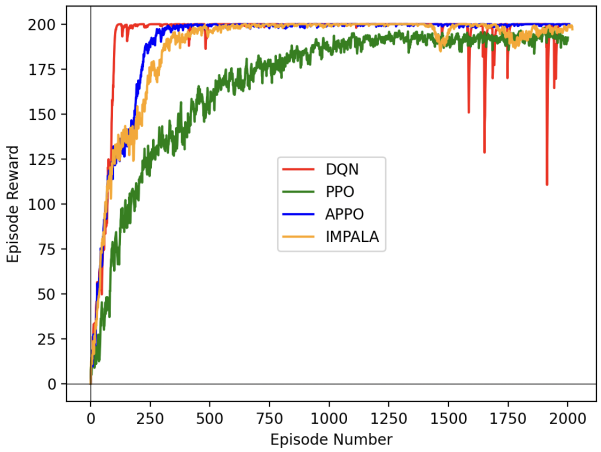

Fig. 5: Results from testing Scenario 3. An EWMA with $\beta = 0.75$ was used.

hopping entity requires the learning algorithm to be aware of the hopping pattern (which changed the transmission channel every step in time), and the asynchronous data collection in these algorithms may not be capable of that [38].

Additionally, DQN demonstrated rather quick learning to an optimal convergence, before dropping its performance at around 600 episodes; this drop in performance is conventionally known as catastrophic forgetting [39]. Upon further investigation, a highly likely cause of this is that 600 episodes, or 60,000 steps, is equal to the size of the experience replay buffer capacity of this DQN implementation which was set as a hyperparameter, so the drop in episode rewards coincides with the point at which the replay buffer begins to filter out previous experiences, whether in a first-in first-out (FIFO) cyclic manner or with a different methodology [35]. Allowing previous experiences to be removed from the sampling process can destabilize learning due to, for instance, specific valuable experiences being lost or the policy being learned from new experiences differing from the policy attained from the initial 60,000 experiences [40]. DQN did appear to relearn after this, but in this scenario, PPO demonstrated more robustness against catastrophic forgetting. Though this is specific to the family of algorithms that uses replay buffers, researchers using our tool to develop MARL algorithms might mitigate against errors like this by sampling different hyperparameter combinations or using different buffer structures, such as hybrid buffers that permanently store select few experiences or separately store rare transitions.

3) Scenario 3: Dynamic Spectrum Access with Constant Frequency Entities: An extremely common setting in wireless communications is when each cognitive radio possesses the goal of transmitting on a free channel. This scenario was designed to simulate how RF devices would learn to act in a simple variation of this setting, by using the `dsa` reward function. Due to the non-learning entities only transmitting on constant frequencies in this scenario, it is comparable to Scenario 1 when each agent learned to target one set channel,

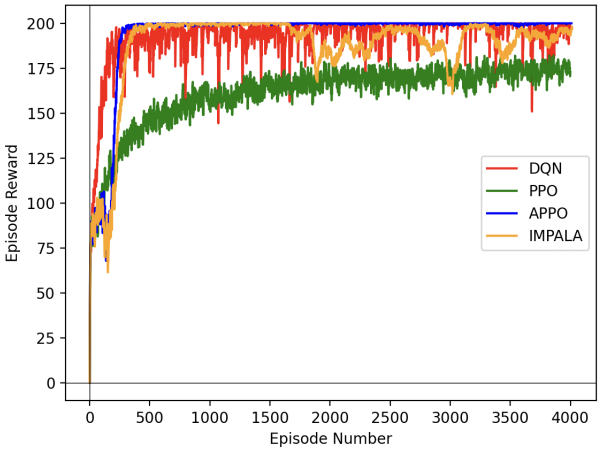

Fig. 6: Results from testing Scenario 4. An EWMA with $\beta = 0.75$ was used.

except in this case multiple agents targeting the same channel would negatively reward both of them. It was expected that learning would be fairly fast, similarly to Scenario 1, but with less stability due to agents being able to interfere with each other's learning goals, and these predictions were reflected in the results. They are displayed in Fig. 5. One notable outcome was that DQN once again demonstrated catastrophic forgetting after maintaining high rewards consistently.

4) Scenario 4: Dynamic Spectrum Access with Fixed-Hop Frequency Entities: In this scenario, we wished to demonstrate agents learning to find free channels when there are moving entities in the environment. However, there are fewer entities than in the previous scenario to make it more feasible for the agents to identify patterns in the channels that are occupied. Naturally, there was significantly more variability in the rewards due to the agents having to learn a more complicated policy and continuously unintentionally running into collisions (e.g., with each other) in the convoluted spectrum; the results are in Fig. 6.

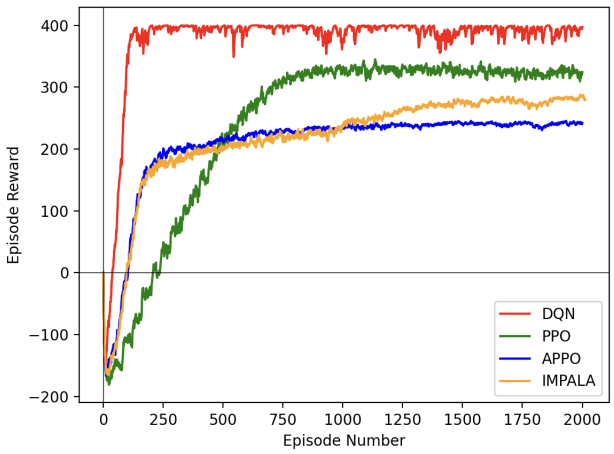

Fig. 7: Results from testing Scenario 5. An EWMA with $\beta = 0.75$ was used.

5) Scenario 5: A Mixed Setting: With each scenario so far, we have attempted to incorporate more realism to provide simulations of these algorithms working in useful contexts, so Scenario 5 emulates a situation with some agents having goals that allow for cooperation with others being directly adversarial to other agents in the spectrum. To support this, there is variance in observation modes, reward modes, and target entities among the 4 agents. The results are shown in Fig. 7.

Evidently, none of the algorithms learned to approximate a completely optimal function. DQN learned an almost optimal policy, although there are continuous dips in performance which were due to the DSA agents taking actions that interfered with the fixed-hop frequency entities. Similarly, with each of PPO, APPO, and IMPALA, a subset of the agents learned a policy without issue and the remaining agents either decided to not transmit to obtain the 0 reward or continuously chose suboptimal actions that averaged out to a positive reward. Similar to the previous scenarios for these algorithms, the agents jamming the constant-frequency entities tended to learn perfectly, the agents jamming the fixed-hop frequency entities tended to not transmit due to difficulty following the entity, and the agents finding free channels did transmit but had high variability in their rewards. A primary goal in this tool's development is to empower future RF and RL researchers who are using it to explore exactly these types of nuanced trade-offs, as understanding them is crucial for developing future algorithms that reach optimal policies in situations with competing intelligences. Additionally, as these scenarios have shown that each algorithm has cases where it falters, these results further demonstrate the gap that can be filled by using our tool to construct sophisticated MARL algorithms targeted toward a variety of situations in radio communications.

VI. Conclusion

This paper introduces the addition of multi-agent reinforcement learning functionality to the RFRL Gym. This new functionality is key for accurately representing and simulating the real-world wireless spectrum, as evidenced by the increase in spectrum usage by various modern devices in recent years [2]. In conjunction with this upgrade, the RFRL Gym was integrated with Ray RLlib. This new interface leverages the new industry-standard Gymnasium API. As a result, the RFRL Gym now supports countless additional integrations including Stable-Baselines3, PettingZoo, and others.

From experimentation, our results conclusively demonstrate that the environment properly enables multiple agents to learn optimal policies, despite having complex variations across the reward functions used. Additionally, it has been shown that a variety of multi-agent learning algorithms can be used in tandem with the new multi-agent RFRL Gym environment.

Future enhancements to the radio frequency (RF) modeling aspect of our testbed include the use of real hardware to integrate with and represent the state of the environment, adding multi-channel agents (which can represent real-world devices that transmit signals on multiple frequency bands at once), and implementing more complex channel modeling approaches to take into consideration more RF details when simulating multiple agents. On the machine learning front, future work to be done involves allowing for more customizable reward functions (e.g., functions that incentivize agents to jam other learning agents or that define collaboration strategies where only a single agent in the collaborating group jams an entity) within the scenario file, evaluating the performance (e.g., convergence time) of different algorithms in our testbed as the number of agents is scaled up (which, with the RLlib API, can be scaled up arbitrarily), and testing different variations of scenarios with full centralization / partial centralization of the agents' observations, actions, and rewards.

References

[1] "FCC Table of Frequency Allocations," 2022. [Online]. Available: https://transition.fcc.gov/oet/spectrum/table/fcctable.pdf

[2] "Spectrum Allocation in the United States," 2022. [Online]. Available: https://api.ctia.org/wp-content/uploads/2022/09/Spectrum-Allocation-in-the-United-States-2022.09.pdf

[3] 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, no. 2, pp. 15–29, 2019.

[4] 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.

[5] 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.

[6] 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.

[7] D. Rosen, I. Rochez, C. McIrvin, J. Lee, K. D'Alessandro, M. Wiecek, N. Hoang, R. Saffarini, S. Philips, V. Jones, W. Ivey, Z. Harris-Smart, Z. Harris-Smart, Z. Chin, A. Johnson, A. M. Jones, and W. C. Headley, "RFRL Gym: A Reinforcement Learning Testbed for Cognitive Radio Applications," in *2023 International Conference on Machine Learning and Applications (ICMLA)*, 2023, pp. 279–286. [Online]. Available: https://ieeexplore.ieee.org/document/10459983

[8] R. Bajracharya, R. Shrestha, S. A. Hassan, H. Jung, and H. Shin, "5G and Beyond Private Military Communication: Trend, Requirements, Challenges and Enablers," *IEEE Access*, vol. 11, pp. 83 996–84 012, 2023.

[9] W. S. H. M. W. Ahmad, N. A. M. Radzi, F. S. Samidi, A. Ismail, F. Abdullah, M. Z. Jamaludin, and M. N. Zakaria, "5G Technology: Towards Dynamic Spectrum Sharing Using Cognitive Radio Networks," *IEEE Access*, vol. 8, pp. 14 460–14 488, 2020.

[10] J. Si, R. Huang, Z. Li, H. Hu, Y. Jin, J. Cheng, and N. Al-Dhahir, "When Spectrum Sharing in Cognitive Networks Meets Deep Reinforcement Learning: Architecture, Fundamentals and Challenges," *IEEE Network*, pp. 1–9, 2023.

[11] Z. Li, C. Guo, and Y. Xuan, "A Multi-Agent Deep Reinforcement Learning Based Spectrum Allocation Framework for D2D Communications," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6.

[12] A. Feriani and E. Hossain, "Single and Multi-Agent Deep Reinforcement Learning for AI-Enabled Wireless Networks: A Tutorial," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 1226–1252, 2021.

[13] K. Zhang, Z. Yang, and T. Başar, *Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms*. Cham: Springer International Publishing, 2021, pp. 321–384. [Online]. Available: https://doi.org/10.1007/978-3-030-60990-0_12

[14] T. J. O'Shea, J. Corgan, and T. C. Clancy, "Convolutional Radio Modulation Recognition Networks," in *Engineering Applications of Neural Networks*, C. Jayne and L. Iliadis, Eds. Cham: Springer International Publishing, 2016, pp. 213–226.

[15] R. Varshney, C. Gangal, M. Sharique, and M. S. Ansari, "Deep Learning based Wireless Channel Prediction: 5G Scenario," *Procedia Computer Science*, vol. 218, pp. 2626–2635, 2023, international Conference on Machine Learning and Data Engineering. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050923002363

[16] D. Adesina, C.-C. Hsieh, Y. E. Sagduyu, and L. Qian, "Adversarial Machine Learning in Wireless Communications Using RF Data: A Review," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, pp. 77–100, 2023.

[17] Z. qi Li, X. Liu, and Z. long Ning, "Dynamic Spectrum Access Based on Deep Reinforcement Learning for Multiple Access in Cognitive Radio," *Physical Communication*, vol. 54, p. 101845, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1874490722001367

[18] M. K. Giri and S. Majumder, "Distributed Dynamic Spectrum Access Through Multi-Agent Deep Recurrent Q-Learning in Cognitive Radio Network," *Physical Communication*, vol. 58, p. 102054, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1874490723000575

[19] L. Yuan, Z. Zhang, L. Li, C. Guan, and Y. Yu, "A Survey of Progress on Cooperative Multi-Agent Reinforcement Learning in Open Environment," 2023. [Online]. Available: https://arxiv.org/abs/2312.01058

[20] Y. Wang, Y. Wan, C. Zhang, L. Bai, L. Cui, and P. Yu, "Competitive Multi-Agent Deep Reinforcement Learning with Counterfactual Thinking," in *2019 IEEE International Conference on Data Mining (ICDM)*, 2019, pp. 1366–1371.

[21] T. T. H. Pham, W. Noh, and S. Cho, "Multi-Agent Reinforcement Learning Based Optimal Energy Sensing Threshold Control in Distributed Cognitive Radio Networks with Directional Antenna," *ICT Express*, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405959524000018

[22] X. Tan, L. Zhou, H. Wang, Y. Sun, H. Zhao, B.-C. Seet, J. Wei, and V. C. M. Leung, "Cooperative Multi-Agent Reinforcement-Learning-Based Distributed Dynamic Spectrum Access in Cognitive Radio Networks," *IEEE Internet of Things Journal*, vol. 9, no. 19, pp. 19 477–19 488, 2022.

[23] 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. [Online]. Available: https://ieeexplore.ieee.org/document/10071938

[24] P. Gawłowicz and A. Zubow, "ns-3 meets OpenAI Gym: The Playground for Machine Learning in Networking Research," in *Proceedings of the 22nd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, ser. MSWIM '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 113–120. [Online]. Available: https://doi.org/10.1145/3345768.3355908

[25] D. M. Coleman, K. R. McKeever, M. L. Mohr, L. R. O. Rosario, K. E. Parker, and M. L. Plett, "Overview of the Colosseum: The World's Largest Test Bed for Radio Experiments," *Johns Hopkins Applied Physics Laboratory Technical Digest*, vol. 5, no. 1, pp. 4–11, 2019.

[26] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," 2016. [Online]. Available: https://doi.org/10.48550/arXiv.1606.01540

[27] "Stable-Baselines3 Docs - Reliable Reinforcement Learning Implementations," 2024. [Online]. Available: https://stable-baselines3.readthedocs.io/en/master/

[28] Farama Foundation, "PettingZoo: An API Standard for Multi-Agent Reinforcement Learning." 2023. [Online]. Available: https://pettingzoo.farama.org/

[29] L. Bonati, M. Polese, S. D'Oro, S. Basagni, and T. Melodia, "OpenRAN Gym: AI/ML Development, Data Collection, and Testing for O-RAN on PAWR Platforms," *Computer Networks*, vol. 220, p. 109502, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128622005369

[30] Farama Foundation, "Gymnasium: An API Standard for Reinforcement Learning with a Diverse Collection of Reference Environments," 2023. [Online]. Available: https://gymnasium.farama.org/index.html

[31] Ray, "RLlib: Industry-Grade Reinforcement Learning," 2023. [Online]. Available: https://docs.ray.io/en/latest/rllib/index.html

[32] X. Lyu, Y. Xiao, B. Daley, and C. Amato, "Contrasting Centralized and Decentralized Critics in Multi-Agent Reinforcement Learning," in *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '21. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2021, p. 844–852.

[33] Z. Ning and L. Xie, "A Survey on Multi-Agent Reinforcement Learning and Its Application," *Journal of Automation and Intelligence*, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2949855424000042

[34] G. Wen, J. Fu, P. Dai, and J. Zhou, "DTDE: A New Cooperative Multi-Agent Reinforcement Learning Framework," *The Innovation*, vol. 2, no. 4, 2021. [Online]. Available: https://www.cell.com/the-innovation/fulltext/S2666-6758(21)00087-4

[35] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," 2013. [Online]. Available: https://arxiv.org/abs/1312.5602

[36] J. Zhu, F. Wu, and J. Zhao, "An Overview of the Action Space for Deep Reinforcement Learning," in *Proceedings of the 2021 4th International Conference on Algorithms, Computing and Artificial Intelligence*, ser. ACAI '21. New York, NY, USA: Association for Computing Machinery, 2022. [Online]. Available: https://doi.org/10.1145/3508546.3508598

[37] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," 2017. [Online]. Available: https://arxiv.org/abs/1707.06347

[38] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, S. Legg, and K. Kavukcuoglu, "IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures," 2018. [Online]. Available: https://arxiv.org/abs/1802.01561

[39] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming Catastrophic Forgetting in Neural Networks," *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, 2017. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.1611835114

[40] C. Kaplanis, M. Shanahan, and C. Clopath, "Continual Reinforcement Learning with Complex Synapses," in *Proceedings of the 35th International Conference on Machine Learning*, 2016, pp. 2497–2506. [Online]. Available: https://doi.org/10.48550/arXiv.1802.07239