`GrGym`: A Playground for Research on RL/AI Enhanced Wireless Networks

Anatolij Zubow, Sascha Rösler, Piotr Gawłowicz, Falko Dressler
Technische Universität Berlin, Germany
{zubow, roesler, gawlowicz, dressler}@tkn.tu-berlin.de

***Abstract*—The provision of a wide range of services each with different requirements makes next generation wireless networks become more complex and heterogeneous which is aimed to be tackled through network softwarization and the application of Artificial Intelligence (AI)-based methods. Specifically, AI methods based on Deep Reinforcement Learning (RL) became very popular as they enable closed-loop end-to-end network optimization even of complex and heterogeneous wireless networks. However, for early deployments there is a pressing need for well-defined environments so that deep RL-based solutions can be studied. We present `GrGym`, a software framework for the development of deep RL enhanced wireless networks, with a specific focus on its usage in experimental 5G/6G research. It is based on the OpenAI Gym toolkit and the flexible GNU Radio platform. With `GrGym`, deep RL-based solutions for 5G/6G networks can be trained in simulated environments as well as real-world testbeds using software-defined radios.**

***Index Terms*—5G/6G Communication Networks, Machine Learning, (Deep) Reinforcement Learning, Software-defined Radio, GNU Radio, OpenAI Gym**

I. Introduction

The wireless communications industry continues its rapid growth for decades [1], [2]. Especially the success of the mobile broadband Internet has been seen as a major driving force behind the evolution of wireless technologies. Upcoming 5G networks will go beyond that by providing additional new services like enhanced mobile broadband, ultra-reliable low-latency communications, and massive machine-type communications. However, the provision of such a wide range of services makes the networks more complex and also more heterogeneous. Network softwarization and the application of Artificial Intelligence (AI)-based methods enabling closed-loop end-to-end network optimization are considered to overcome these issues. At the same time, researchers from academia and industry are already working on the successor termed as 6G and Next G, respectively, which will add new use cases like tactile and haptic Internet, holographic communications, and computation oriented communications which is required for the vision of "connected intelligence" [3], [4].

Inspired by these trends, we developed `GrGym` [5], a software framework for the development of AI enhanced wireless networks. In this paper, we discuss `GrGym`, with a main focus on its usage in experimental 6G research. Specifically, we focus on AI methods based on Deep Reinforcement Learning (RL) as it become very popular recently [6], [7]. `GrGym` is based on two frameworks, namely the OpenAI Gym toolkit and the GNU Radio SDR platform. With `GrGym` deep RL-based solutions for 5G/6G networks can be trained in simulated environments as well as real-world testbeds. As a proof-of-concept, we developed an RL-based adaptive rate control for the IEEE 802.11 technology. Our toolkit is provided to the community as Open Source.[1]

This work was supported in part by the Federal Ministry of Education and Research (BMBF, Germany) within the 6G Research and Innovation Cluster 6G-RIC under Grant 16KISK020K.

II. The Role of (Deep) Reinforcement Learning in 5G/6G Networks

Reinforcement learning [8] is one of the most important research directions in Machine Learning (ML), which has significantly influenced the development of AI over the last decades. In RL, an agent interacts with the environment in order to make decisions, observe the results, and then automatically adjust its strategy based on the received reward (or penalty) to achieve the optimal policy. Although the learning process eventually converges, it can take a lot of time to reach the best policy, as the agent has to explore and gain knowledge of an entire, possibly large, system, making it unsuitable for many real-world applications. Recently, a new technique termed as Deep Learning (DL) [9] was introduced which helps to overcome the limitations of classical RL approaches. Deep RL combines reinforcement learning and deep learning and utilizes Deep Neural Networks (DNNs) to train the learning process resulting in an improved learning speed and hence performance of deep RL based algorithms. As a result, deep RL has been successfully applied in numerous practical applications ranging from robotics, computer vision, computer games, speech recognition, and natural language processing.

The usage of (deep) RL for the optimization of 5G communication networks became very popular recently as a number of surveys confirms [6], [7], [10]. Applications are ranging from network access and rate control, caching and offloading, security and connectivity preservation to traffic routing and resource scheduling.

OpenAI Gym Toolkit

OpenAI Gym [11] is a framework for benchmarking (deep) RL-based solutions. RL agents are implemented using the high-level programming language Python, which allows to

[1]https://github.com/tkn-tub/gr-gym

use powerful ML libraries like Keras and Tensorflow, simplifying the development of deep RL solutions based on neural networks. Gym provides a unified API that structures the interactions between the RL agent and the environment. Specifically, to integrate an environment to the Gym, its observations, actions, and rewards have to be represented as structured numerical data. This way, the framework was already interfaced with a large set of diverse environments in areas ranging from video games to robotics [11] and simulation of intelligent transportation systems [12]. Recently, we presented two frameworks ns3-gym [13], an interface to ns-3, which is a discrete-event network simulator for Internet systems. This made it possible to use any simulated communication network (e.g., 3GPP LTE or IEEE 802.11) as an environment within OpenAI Gym and to optimize it using (deep) RL.

III. Trend towards Softwarization

Next generation 5G/6G wireless networks will be highly flexible, thanks to the fact that an increasing number of functionalities are realized in software rather than dedicated hardware. The enabling technologies for network reconfiguration are Network Function Virtualization (NFV) [14], Software Defined Networking (SDN) [15], and Software Defined Radio (SDR) [16]. In this work, we focus on SDR as it represents the enabling technology for our envisioned software framework for experimental wireless 5G/6G research. There are a variety of radio platforms available that support full programmability up to the physical layer of the radio functionality, which are often referred to as SDR. Their capabilities have been dramatically improving over the years, i.e. supporting wide RF bandwidth and high processing power sufficient to implement most modern wireless technologies like LTE or 802.11. With SDR the softwarization of wireless networks can also happen at the lower layers (PHY and MAC), rather than being restricted to upper network layers which is typically the case with wired networks. However, the flexibility of SDRs requires extensive support from software side with frameworks tailored to the needs of upcoming 5/6G wireless networks.

GNU Radio Framework

GNU Radio [17] is an open source software toolkit, which provides a comprehensive library of optimized, state-of-the-art signal processing that can be glued together for building complex real-time SDR solutions. With GNU Radio, a radio system can be built by designing a Radio Companion (GRC) flow graph where the vertices are signal processing blocks (implemented in C++) and the edges represent the data flow between them. Each signal processing block processes in real-time an infinite stream of data flowing from its input ports to its output ports. Each block is described by the number of input and output ports as well as the type of data that flows. Streams of data are a model that work well for samples or bits, which are typically processed at the PHY layer. However, such a model is not a proper mechanism for transport of control data, metadata, or packet structures needed at the MAC or higher layers. Therefore, GNU Radio provides an additional message passing interface that handles the delivery of information in an asynchronous basis. In GNU Radio it is easy to replace existing signal processing blocks with more efficient implementations or blocks relying on ML approaches. It is possible to run GNU Radio programs on either real hardware (e.g., USRP SDR) or loopback in a fully simulated environment allowing application of channel propagation models to synthetically generated signals [18]. Finally, there are partial/full implementations of radio technologies like IEEE 802.11 [19], [20], IEEE 802.15.4 [21], narrow-band LoRa [22], and 3GPP LTE [23].

IV. The GrGym Framework in a Nutshell

A. Objectives & Design Principles

The main goal of our work is to facilitate and shorten the time required for developing novel deep RL-enhanced wireless network solutions. We believe that developing RL-driven control algorithms and training them in a simulated environment is very often more practical (i.e., easier, faster, less expensive, and safer) in comparison to directly running experiments in the testbed or even real world. However, it should be easy to switch from an environment with simulated wireless channel and interference to testbeds with real radio hardware deployments without requiring many changes to the radio programs. This is of paramount importance in order to finally test the performance under real conditions and all impairments (e.g., hardware, channel propagation). Moreover, it is a requirement for the implementation of the concept of *transfer learning*, i.e., the ability to reuse previously acquired learned knowledge in a new (more complex) system or a real environment. This allows an RL-agent trained in a simulated environment (e.g., a wireless channel) to directly interact or be retrained in the real world much faster than when starting from the scratch [24]. How well the agent copes with the real-world environment depends on the accuracy of the simulation channel models that were used during training. In GNU Radio, there are lots of simulated channel models available ranging from very simple ones (e.g., AWGN) to more realistic ones which take multi-path, mobility, and fast fading into account. Moreover, the impact of interference can be easily simulated in GNU Radio as well. Finally, note that we do not constrained ourselves to RL as the framework could be used to generate data sets and use them for offline learning.

We aim to achieve the above by exploiting the flexibility of SDR platforms and softwarization using the GNU Radio framework. However, there are challenges. First, the GNU Radio programs need to run in real-time and cannot be paused, as it is possible with pure simulations running in simulation time [13]. So, no computational expensive tasks can be executed within the RL agent's main control loop. However, offloading CPU intensive AI tasks is still feasible by running the RL agent and the radio program(s) on different machines by having a loose coupling between them.

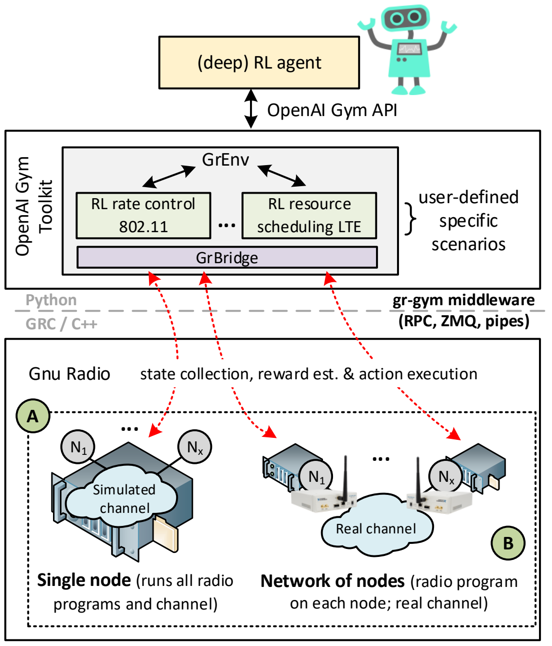
(deep) RL agent
OpenAI Gym API
OpenAI Gym Toolkit
GrEnv
RL rate control 802.11
RL resource scheduling LTE
GrBridge
user-defined specific scenarios
Python
GRC / C++
gr-gym middleware (RPC, ZMQ, pipes)
Gnu Radio
state collection, reward est. & action execution
A
B
Simulated channel
Real channel
Single node (runs all radio programs and channel)
Network of nodes (radio program on each node; real channel)

Fig. 1: Architecture of the GrGym framework.

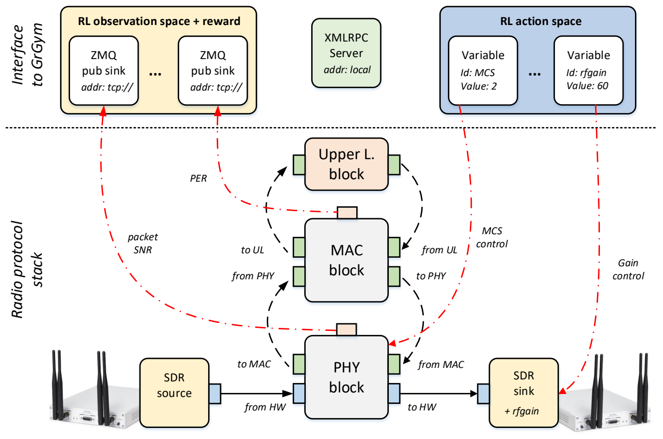

Fig. 2: Interfacing GrGym inside GNU Radio flowgraph.

B. Architecture

The architecture of the GrGym framework is shown in Fig. 1. It consists of the following major components: GNU Radio framework, OpenAI Gym toolkit, and the GrGym framework itself. In the following, we give a brief overview – a detailed description can be found in [5].

1) GNU Radio Side: Any wireless communication protocol implemented using the GNU Radio framework can be integrated into GrGym. The simplest integration is possible if the radio program is provided in the form of a GRC flow graph. Here, only some minor modifications need to be made to expose the data needed to capture the desired RL observation space and information needed to compute the reward value. Moreover, the possibility to change values of variables is needed as a mean to execute RL actions.

This is achieved by modifying the GRC file of the radio program to be integrated into GrGym by adding additional GNU Radio blocks and wiring them accordingly as illustrated in Fig. 2. The additional * Sink nodes (upper left) are needed to capture both observations and reward. GrGym is able to use different inter-process communication (IPC) mechanisms like named pipes (files), ZeroMQ, UDP/TCP. The usage depends on whether GrGym and GNU Radio are co-located or not, i.e., ZeroMQ can be used if both processes are located on different machines. The XMLRPC Server node (upper middle) is needed for life-cycle management of the GNU Radio program by GrGym, i.e., start/stop, and (remote) execution of RL actions. Finally, the Variable nodes (upper right) are global shared variables used by GrGym to execute RL actions, i.e. changing the configuration state of blocks using those variables.

2) RL Agent Side: A (deep) RL agent is using the OpenAI Gym interface for accessing the state and executing actions on an environment implemented using the GNU Radio and wrapped by our GrGym framework. This makes the RL-agent's code (Python) environment-independent, which allows for exchanging the agent's implementation while keeping the reproducibility of the environment's conditions. Moreover, different RL approaches can be tried out very quickly.

3) GrGym Middleware: The GrGym middleware takes care of transferring state (i.e., observations, reward) and control (i.e., actions) between the RL agent and the node or even network of nodes running GNU Radio programs. The middleware consists of two parts: a generic part and a user-defined scenario-specific implementation. The generic part interconnects the two frameworks, OpenAI Gym and GNU Radio, with each other. The scenario-specific part needs to be provided by the user when exploring a new scenario (e.g., RL-based rate control for 802.11) or configuration (e.g., number of wireless nodes). This is achieved by providing a custom Python class, which derives from GrScenario and implements the required interface functions. Here, the user can access all observation and reward data exposed by the GNU Radio program via the sink nodes. Note, that the GrGym middleware transfers the state and actions as numerical values and it is up to the user to define their semantics.

V. Example: RL-based Rate Control in 802.11p

As a proof-of-concept, we implemented a deep RL-based rate control for IEEE 802.11p radio based on the GNU Radio implementation provided by [19], [20]. The goal of the RL-agent is to decide on the Modulation and Coding Scheme (MCS) to be selected for the next packet transmissions based on the observation of the current channel condition, which is the absolute signal strength (RSSI) per OFDM subcarrier. This is a challenging task as the RSSI is uncalibrated, i.e., level of the noise floor is unknown and time varying. Therefore, the RL agent needs to learn which MCS to use given the absolute RSSI values reported. The following RL mapping was used:

- *Action*: The MCS to be used for the next packet transmissions (here: 10 ms). Note: 802.11p supports 8 different MCS values, which corresponds to the action space.

- *Reward*: The reward is the effective data rate during the last step. It is computed from the packet success rate during multiplied by the data rate of the selected MCS.
- *Observation*: The observation is a vector of RSSI per OFDM subcarrier of the last received packet. As the observation is measured during the synchronization phase of packet reception, it is independent of the selected MCS, i.e., action.

Moreover, our deep RL agent used the Actor-Critic (AC) method [25]. `GrGym` was run in standalone mode and the 802.11p stack was operated in loopback mode, i.e., the channel was simulated in GNU Radio with AWGN channel. As shown in [5], RL was very efficient in this example, which motivates further experimentation also with other protocols in the telecommunication networks domain.

VI. Discussion and Conclusions

We presented `GrGym`, a toolkit that simplifies the development of AI enhanced wireless networks, with a specific focus on deep RL and its usage in experimental 5G and 6G research. This is achieved by interconnecting the OpenAI Gym toolkit with the GNU Radio framework. Our Open Source framework is generic as it can be easily extended by the research community to optimize different functionalities in 5/6G communication networks using deep RL.

Next to IEEE 802.11p, `GrGym` can be used with any existing protocol implementation for GNU Radio. At the moment, this includes radio technologies like IEEE 802.11 [19], [20], IEEE 802.15.4 [21], narrow-band LoRa [22], and 3GPP LTE [23], however, this list is constantly growing and also quite a number of implementations exist that have not (yet) been made available as Open Source to the research community.

With the ongoing research towards 6G and Next G, with a tremendous growth on the AI side in our protocols, approaches such as our `GrGym` will become even more important. Among others, applications include tactile and haptic Internet, holographic communications, and computation oriented communications. Here, we envision to set up a global leaderboard allowing researchers to share and compare their RL-based solutions for selected communication problems like parameterization of random channel access. Moreover, we plan to evolve our framework beyond just parameter learning of preselected radio programs. Instead, an RL-agent might learn to build the best flowgraph from a repository of available radio components.

References

[1] A. F. Molisch, *Wireless communications.* Wiley, 2012.

[2] P. Launiainen, *A Brief History of Everything Wireless: How Invisible Waves Have Changed the World.* Springer, 2018.

[3] H. Tataria, M. Shafi, A. F. Molisch, M. Dohler, H. Sjoland, and F. Tufvesson, "6G Wireless Systems: Vision, Requirements, Challenges, Insights, and Opportunities," *Proceedings of the IEEE*, vol. 109, no. 7, pp. 1166–1199, Jul. 2021.

[4] F. Dressler, C. F. Chiasserini, F. H. P. Fitzek, H. Karl, R. Lo Cigno, A. Capone, C. E. Casetti, F. Malandrino, V. Mancuso, F. Klingler, and G. A. Rizzo, "V-Edge: Virtual Edge Computing as an Enabler for Novel Microservices and Cooperative Computing," *IEEE Network*, 2022.

[5] A. Zubow, S. Rösler, P. Gawłowicz, and F. Dressler, "GrGym: When GNU Radio goes to (AI) Gym," in *22nd ACM International Workshop on Mobile Computing Systems and Applications (HotMobile 2021).* Virtual Conference: ACM, Feb. 2021, pp. 8–14.

[6] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, "Applications of Deep Reinforcement Learning in Communications and Networking: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3133–3174, 2019.

[7] S. Szott, K. Kosek-Szott, P. Gawłowicz, J. Torres Gómez, B. Bellalta, A. Zubow, and F. Dressler, "WiFi Meets ML: A Survey on Improving IEEE 802.11 Performance with Machine Learning," arXiv, cs.NI 2109.04786, Sep. 2021.

[8] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction.* MIT Press, 2018.

[9] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning.* MIT Press, 2016.

[10] Y. Qian, J. Wu, R. Wang, F. Zhu, and W. Zhang, "Survey on Reinforcement Learning Applications in Communication Networks," *Journal of Communications and Information Networks*, vol. 4, no. 2, pp. 30–39, 2019.

[11] G. Brockman, V. Cheung, L. Petterson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," arXiv, cs.LG 1606.01540, Jun. 2016. [Online]. Available: https://arxiv.org/abs/1606.01540

[12] M. Schettler, D. S. Buse, A. Zubow, and F. Dressler, "How to Train your ITS? Integrating Machine Learning with Vehicular Network Simulation," in *12th IEEE Vehicular Networking Conference (VNC 2020).* Virtual Conference: IEEE, Dec. 2020.

[13] P. Gawłowicz and A. Zubow, "ns-3 meets OpenAI Gym: The Playground for Machine Learning in Networking Research," in *22nd ACM International Conference on Modelling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM 2019).* Miami Beach, FL: ACM, Nov. 2019.

[14] B. Yi, X. Wang, K. Li, S. K. Das, and M. Huang, "A comprehensive survey of Network Function Virtualization," *Computer Networks*, vol. 133, pp. 212–262, Mar. 2018.

[15] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[16] R. Akeela and B. Dezfouli, "Software-defined Radios: Architecture, state-of-the-art, and challenges," *Elsevier Computer Communications*, pp. 106–125, 2018.

[17] E. Blossom, "GNU Radio: Tools for Exploring the Radio Frequency Spectrum," *Linux Journal*, vol. 2004, no. 122, Jun. 2004.

[18] T. O'Shea and N. West, "Radio Machine Learning Dataset Generation with GNU Radio," in *GNU Radio Conference*, Boulder, CO, Sep. 2016.

[19] B. Bloessl, M. Segata, C. Sommer, and F. Dressler, "An IEEE 802.11a/g/p OFDM Receiver for GNU Radio," in *ACM SIGCOMM 2013, 2nd ACM Software Radio Implementation Forum (SRIF 2013).* Hong Kong, China: ACM, Aug. 2013, pp. 9–16.

[20] ——, "Performance Assessment of IEEE 802.11p with an Open Source SDR-based Prototype," *IEEE Transactions on Mobile Computing (TMC)*, vol. 17, no. 5, pp. 1162–1175, May 2018.

[21] B. Bloessl, C. Leitner, F. Dressler, and C. Sommer, "A GNU Radio-based IEEE 802.15.4 Testbed," in *12. GI/ITG KuVS Fachgespräch Drahtlose Sensornetze (FGSN 2013)*, Cottbus, Germany, Sep. 2013, pp. 37–40.

[22] Z. Xu, P. Xie, and J. Wang, "Pyramid: Real-Time LoRa Collision Decoding with Peak Tracking," in *40th IEEE International Conference on Computer Communications (INFOCOM 2021).* Virtual Conference: IEEE, May 2021.

[23] J. Demel, S. Koslowski, and F. K. Jondral, "A LTE Receiver Framework Using GNU Radio," *Journal of Signal Processing Systems*, vol. 78, no. 3, pp. 313–320, Jan. 2015.

[24] P. F. Christiano, Z. Shah, I. Mordatch, J. Schneider, T. Blackwell, J. Tobin, P. Abbeel, and W. Zaremba, "Transfer from Simulation to Real World through Learning Deep Inverse Dynamics Model," arXiv, cs.RO 1610.03518, Oct. 2016.

[25] V. R. Konda and J. N. Tsitsiklis, "On Actor-Critic Algorithms," *SIAM Journal on Control and Optimization*, vol. 42, no. 4, pp. 1143–1166, Jan. 2003.