Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Software Article

Marconi-Rosenblatt Framework for Intelligent Networks (MR-iNet Gym): For Rapid Design and Implementation of Distributed Multi-agent Reinforcement Learning Solutions for Wireless Networks

Collin Farquhar[#], Swatantra Kafle[#], Kian Hamedani[#], Anu Jagannath[#,$], Jithin Jagannath[$,*]

Marconi-Rosenblatt AI/ML Innovation Lab, ANDRO Computational Solutions LLC, Rome, NY, USA

A R T I C L E I N F O

Keywords:
Machine learning
Reinforcement learning
Wireless network
CDMA
Ns-3
OpenAI gym
LPD/I networks
Power and frequency control

A B S T R A C T

We present the Marconi-Rosenblatt Framework for Intelligent Networks (MR-iNet Gym) an open-source architecture designed for accelerating research and development of novel reinforcement learning applied to distributed wireless networks. To ensure an end-to-end architecture, we leverage the existing work of ns3-gym, a software package that allows for using ns-3, a wireless network simulator, as an environment within the OpenAI Gym framework for RL. In addition to this, we have implemented the first known custom CDMA module for ns-3 as well as a framework for RL models with a core suite of implemented algorithms. The software framework capturing the interaction between wireless transceiver (agent) and RL decision engine has been designed to maximize the ease-of-use when testing different RL algorithms and models. In the rest of the paper, we describe these new software components and demonstrate some of the results and capabilities that can be achieved when used in conjunction with the existing open-source ecosystem.

Code metadata

Nr	Code metadata description	Please fill in this column
C1	Current code version	v1.0
C2	Permanent link to code/repository used for this code version	https://github.com/anujagann/MR-iNet
C3	Permanent link to reproducible capsule	NA
C4	Legal code license	Creative Commons Attribution - NonCommercial - ShareAlike 4.0 License (CC BY-NC-SA 4.0). Contact: jjagannath@androcs.com for commercial license
C5	Code versioning system used	none
C6	Software code languages, tools and services used	C++ and Python.
C7	Compilation requirements, operating environments, and dependencies	Equivalent to ns-3 and OpenAI GYM.
C8	If available, link to developer documentation/manual	https://github.com/anujagann/MR-iNet/blob/main/README.md
C9	Support email for questions	(skafle, ajagannath, jjagannath)@androcs.com

1. Utility of MR-iNet Gym

It is well known that it is resource-demanding and time consuming to create large wireless networks during research and development. However, simulating wireless networks offers several advantages: they are low cost, free from equipment constraints, and extensible beyond what may be locally possible and allows for rapid feasibility analysis. In some cases, it is a daunting task to create a credible simulator that has to capture different aspects of the technology being developed. One such example is the evaluation of reinforcement learning (RL) algorithms for distributed wireless networks. This paper presents an open-source simulator MR-iNet, the first end-to-end CDMA-based network simulator allowing users to simulate expensive networks for various applications. We emphasize that changing wireless equipment properties and capabilities in the real world might be time-consuming and costly. In contrast, the simulator offers a configurable interface to vary these properties. Furthermore, the simulator allows us to test networks with greater generality. To a greater extent, it may be infeasible to test a physical network with different channel conditions or with an order of magnitude more users. Still, these things can be easily achieved through simulations.

[*] Corresponding author.
E-mail addresses: cfarquhar@androcs.com (C. Farquhar), skafle@androcs.com (S. Kafle), khamedani@androcs.com (K. Hamedani), ajagannath@androcs.com (A. Jagannath), jjagannath@androcs.com (J. Jagannath).
[#] Key Developers/Shared First Authorship.
[$] Senior Authors.

https://doi.org/10.1016/j.comnet.2022.109489
Received 30 June 2022; Accepted 21 November 2022
Available online 26 November 2022

Wireless communication has been evolving rapidly to keep up with the requirement of modern applications. The new wireless communication systems have massive connectivity among mobile devices, sensors, base stations, and actuators. As a result, it is getting more challenging to develop mathematically tractable models that reasonably represent the wireless environment. Hence, it is necessary to develop methodologies that help understand the wireless environment through data obtained from sensing. Several works have recently shown that machine learning (ML) has revolutionized several data-intensive domains such as computer vision, natural language processing, and medical imaging, including wireless communication. Significant efforts have been made to effectively leverage various strengths of ML to solve crucial hurdles encountered in the wireless domain [1–4]. However, despite recent advances, several limitations and challenges have held back the real-world deployment of machine learning-based decision engines in the wireless domain. Though there are several categories in the broad field of ML, such as supervised and unsupervised learning algorithms, we focus on RL, where an intelligent agent can control communications devices. RL does not require an apriori data set like supervised learning. In a wireless network, a RL agent could be conceptualized as a centralized network controller, or each communication device could be controlled by a separate agent, making it a multi-agent RL problem, thereby opening the door to more distributed approaches.

Having access to a reliable and flexible simulator to rapidly design, implement, and test novel solutions will accelerate the progress in this area. In this paper, we propose an end-to-end framework and related methodology for rapidly designing and evaluating deep reinforcement learning (DRL) for a distributed wireless network. Specifically, we designed an ad hoc Direct sequence code-division multiple access (DS-CDMA) wireless network. DS-CDMA is especially popular for implementing a robust low probability of intercept and detection (LPI/D) physical layer due to its unique advantages, such as easy frequency management (no frequency planning in multi-user scenarios, low peak-to-average power ratio (PAPR), and less stringent synchronization requirements as necessary for orthogonal frequency division multiplexing (OFDM). To the best of our knowledge, this is the first end-to-end software architecture that puts forth an end-to-end framework to demonstrate the feasibility of designing DRL-based decision engines for next-generation distributed CDMA wireless networks. Since ns-3 is a packet-level network simulator, it is important to note that the proposed architecture can be further extended to networks using TDMA or FDMA-based MAC protocols by simply defining MAC functionality.

2. MR-iNet Gym Software Overview

There is a broad interest in software tools for wireless network simulation and RL in general. We leverage tools in the existing open-source ecosystem and integrate them into the MR-iNet process. Namely, we use ns3-gym which interfaces between the discrete-event wireless network simulator, ns-3, and OpenAI Gym, a standard framework for RL. We add to this our custom CDMA module for ns-3 as well as a framework for RL models with several example algorithms implemented. *To the best of our knowledge, this is the first time that a CDMA module has been developed and simulated in ns3.* Our CDMA module supports different modes of communication, namely, unicast, multicast, and broadcast. Therefore, seamlessly designing an end-to-end framework to integrate using all these pieces of software together, we provide code for an example simulation scenario that utilizes the entire software process to find solutions to power control problems in the CDMA network using our RL framework. While we have focused on the power control problem of distributed CDMA networks, *we note that the MR-iNet framework is modular, and one-piece can be swapped out for another, which greatly enhances the potential use cases.* For example, simulating a different communication system is simply a matter of creating a new ns-3 simulation; additionally, using a different RL algorithm becomes seamless once implemented in the base class that serves as the core of our RL framework. Therefore, MR-iNet provides one of the first platforms for easily testing a variety of multi-agent RL solutions for CDMA networks as well as wireless networks of other varieties. Some of these examples will be cited and discussed in Section 3.

CDMA Module Description

An outline of the CDMA module for ns-3 is show in Fig. 1. In the *model* folder, the behavior of the various layers of a CDMA network are defined. In the *helper* folder, a *CdmaHelper* class is defined which helps to aggregate the various classes that define the CDMA layers and attach them to a node by installing a *CdmaNetDevice*. With *CdmaNetDevices* attached to the nodes the rest of the ns-3 script can be implemented using the default functionality of ns-3.

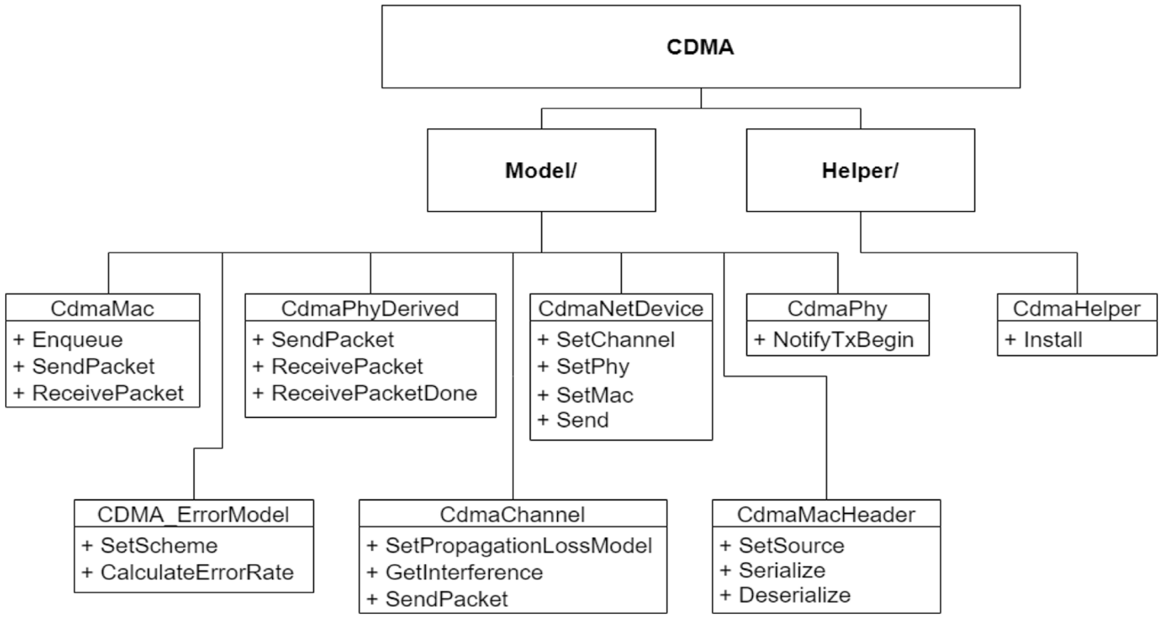


Fig. 1. High level diagram of the CDMA module. Lines between boxes indicate the directory structure with the lower boxes being inside the folder of the higher box. The class boxes contain partial list of the class methods.

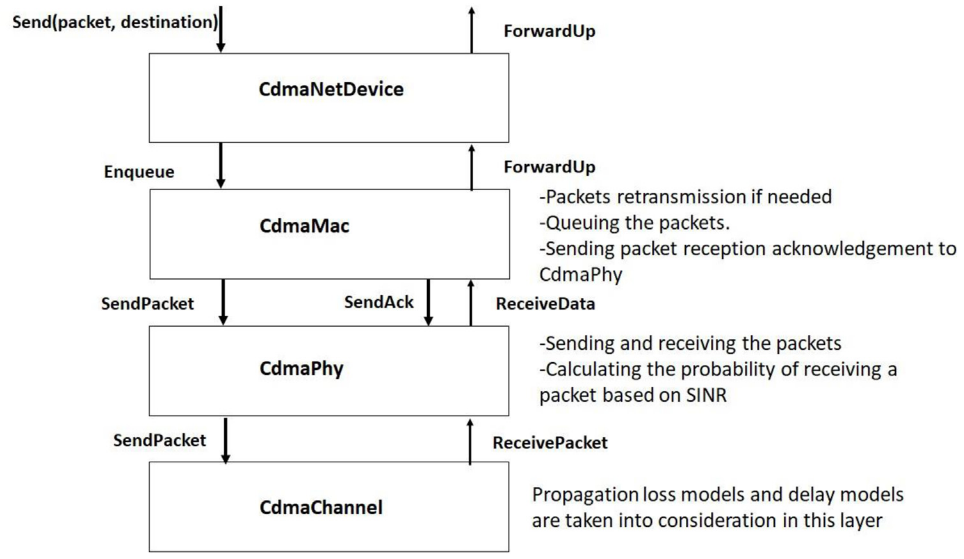

Send(packet, destination)
ForwardUp
CdmaNetDevice
Enqueue
ForwardUp
CdmaMac
-Packets retransmission if needed
-Queuing the packets.
-Sending packet reception acknowledgement to CdmaPhy
SendPacket
SendAck
ReceiveData
CdmaPhy
-Sending and receiving the packets
-Calculating the probability of receiving a packet based on SINR
SendPacket
ReceivePacket
CdmaChannel
Propagation loss models and delay models are taken into consideration in this layer

Fig. 2. Function calls that connect the layers when sending a packet using the CDMA module.

After the nodes are initialized with *CdmaNetDevice* objects they can be used to transmit packets. At the simulation-level, sending a packet only requires a single function call, which then sets off a series of function calls in the CDMA module which happen automatically. The function calls move through layers of the CDMA module in a similar way to how a realistic physical transmitter would operate. The *send* function of the *CdmaNetDevice* class initiates the process, much like an application-layer, which then prompts the MAC-layer to enqueue the packet. The packet is then sent to the PHY layer and passed into the channel where propagation loss models and delay models may be applied. The channel layer then passes the packet back through the PHY-layer, MAC-layer, and back up to the *CdmaNetDevice* on the receiver side. Fig. 2 depicts this interaction between layers in the CDMA module. The fact that this process mirrors the interaction between layers in real-life systems is intentional and is a result of a core design-philosophy of the CDMA module, that is, to reduce the reality gap between simulations and real systems as much as possible.

Reinforcement Learning Overview

First, we describe a general RL problem before discussing specific use cases. A RL problem is characterized by the interaction of an agent in an environment [5]. The agent receives state information from the environment, which it then uses to select an action to perform in the environment. The agent takes this step in the environment and then may be in a new state. The agent gets a reward for taking this step and receives information about its new state. This process of information flow of states, actions, and rewards can be thought of as a loop that continues throughout the training episode. This information loop is represented in Fig. 3. It is then the agent's job to maximize the reward it receives. Therefore, it is our job to define the reward function such that maximizing the function is equivalent to optimizing whatever objective we have in mind for the agent.

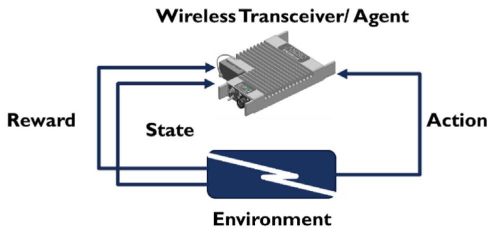


Fig. 3. The RL information loop where states, actions, and rewards are passed between the agent and its environment. This diagram shows a single agent, but for the multi-agent case, many agents may be taking actions in the same environment.

Formally, the environment is a Markov Decision Process (MDP) which can be represented as a tuple, $(S,\ A,\ P_a,\ R_a)$, where S is the set of states, A is the set of actions, $P_a(s^{t+1} \mid s^t,\ a)$ is the probability transition function representing the probability of the agent moving to state s^{t+1} given that it was in state s^t and took action a, and $R_a(s^{t+1},\ s^t)$ is the reward function.

Emulating multi-agent in ns3-gym

For each simulation scenario, one must implement the ns3-gym functions that serve as an interface between the ns-3 simulation in C++ and Python-side where agents created with our RL framework perform their functions. By default, ns3-gym expects the reward to be a single scalar value. This is at odds with a multi-agent setting where each agent may receive a different reward. There are several workarounds for this, but in our example code, we handle this by passing a vector of reward information through the *info* variable that is returned by the *step* function for any Gym environment.

RL framework Implementation

Our RL framework is designed to give a common structure of inherited member functions to all implemented RL algorithms. This common structure ensures that changing one algorithm for another requires only changing the initialization of the list of agents. The base class, abstract methods, and inheritance structure are shown in Fig. 4. Since the syntax

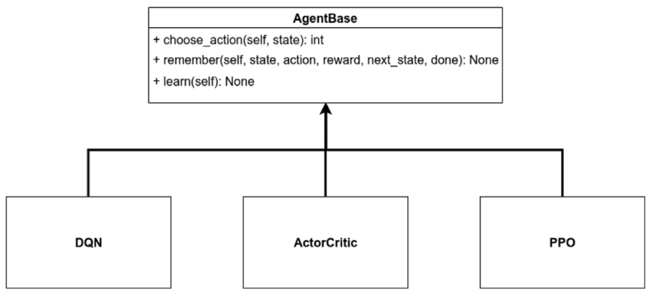

AgentBase
+ choose_action(self, state): int
+ remember(self, state, action, reward, next_state, done): None
+ learn(self): None
DQN
ActorCritic
PPO

Fig. 4. *AgentBase* is the abstract base class for our RL framework. The function calls any agent makes during the RL loop are defined as abstract methods in *AgentBase*. The inputs to the methods are written inside parentheses ("self" is the required first argument for member functions of Python classes) and the return type is written after the colon. The algorithms implemented in our framework then inherit from this base class and implement the abstract methods. For example, DQN, Actor Critic, and PPO are the three RL algorithms that we have implemented in our framework.

for calling a member function is the same for any RL algorithms in the framework, no changes are required in the RL training code.

We define an abstract base class *AgentBase*, which consists of three abstract methods that must be overridden by classes that inherit from *AgentBase*, namely, *choose_action, remember*, and *learn*. The *choose_action* function takes a list of state information as input which is passed to the RL model which then outputs an action that is returned by the function. The *remember* function takes an entire RL transition (an MDP tuple defined previously) and can save this transition to an agent's replay memory so it can be used for future training. The *learn* function takes no inputs and enacts the training process for an agent.

3. Example problem description

We also provide an example use case in our repository. In the example scenario, 3 DQN or deep deterministic policy gradient (DDPG) agents are simulated on CDMA transmitter/receiver pairs in an ns-3 script. *Each decision engine agent is modeled with a DQN from our RL framework, making this a multi-agent RL scenario a partially observable MDP.* The action space consists of three discrete power levels. When multiple agents transmit at the same time, they cause some interference to each other. The reward function is constructed to incentivize the agents to balance between transmission at a higher power versus being penalized for causing interference to neighboring nodes. The agents learn to transmit at an optimal power, i.e., while they are gaining the maximum throughput, they cause the least possible interference to other nodes. The overall upward trend in the reward over the training period indicates that the agents are successful in learning to maximize their objectives. It is worth mentioning that this result shows that the agents can learn to find an optimal solution which is mutually beneficial where they optimize both individual and network performance, despite each agent being controlled by a separate RL model and acting individually.

4. Active Research and Publications Enabled by the Software

In our first effort [6], we trained and deployed the DDPG power control agents on GPU embedded software-defined radios (SDR). Our motivation to leverage DDPG was that the DDPG agents can support continuous action space as opposed to DQNs which can only support discrete action spaces. The nature of the power control problem is a continuous action space as our GPU embedded SDRs can transmit at any power level within the range of 0-5 milliwatts (mW). After the DDPG agents are successfully trained within our ns3-gym simulator, they are frozen and optimized using TensorRT. TensorRT is a CUDA package developed by NVIDIA to optimize the DDPG engine and reduce its inference time in any delay-sensitive applications. Our experiments show that the inference time of the DDPG engine can be reduced from 14.6 microseconds (uS) to 0.23(uS) through the optimization process which is performed by TensorRT. The optimized DDPG engine can be deployed on the radios. The output of the optimization process is a PLAN file. In fact, the CUDA-based PLAN inference engines and C++ codes are compiled separately and are linked together using the linking tools that have been made available since CUDA 5.0.

We evaluated the performance of the optimized DDPG engines on real hardware testbed in our lab environment where three pairs of the CDMA transceivers were communicating simultaneously. We compared the performance of our DDPG engines against a traditional power control algorithm, namely, distributed constrained power control (DCPC) algorithm. The results indicate that the power consumption of DCPC is $\sim$3 times higher than DDPG agents while they both achieve a similar packet delivery ratio (PDR).

The modularity of our approach allows for training in different communication settings and with different objectives. Beyond optimizing power control, we used our RL framework with ns3-gym and other simulated environments to optimize transmission control in a TDMA-like setting [7]. In this multi-agent scenario, the transmission medium is constrained such that only k or fewer agents can transmit at the same time, otherwise their transmissions will be unsuccessful due to friendly interference. The agents take actions that amount to decisions either not to transmit, to transmit immediately, or to wait some finite number of time steps and then transmit. Here, we use a Deep Q-Network (DQN) model, as a discrete action space aligns with the finite set of decisions to make regarding whether and when to transmit. We benchmarked our simulated agents against CSMA-style algorithms and found our agents to consistently achieve better performance in terms of throughput and fairness.

In addition, we are investigating the problem of joint power control and frequency selection in the presence of jammers. Using MR-iNet, we created several network configurations to train each agent to understand different wireless environments and learn a good policy. As demonstrated in the previous work on power control, we expect these learned policies to be equally effective during deployment in the radios. Moreover, because of the given framework, we can create multiple networks easily during the training and evaluate the effectiveness of learned policy by testing the learned model in different network scenarios. Further, we are studying different training strategies, such as the effect of model aggregation on the testing performance. Additionally, we are investigating the impacts of using RL agents trained on a network of a different size than the testing network on performance. Without MR-iNet, it would have been almost impossible to research all these aspects of the training and testing of the joint power control and frequency selection problem.

5. Future Work

There are a number of research areas that could benefit from using MR-iNet. For those interested in simulating CDMA networks, our custom CDMA module provides one of the only options to do so in ns-3. Given the generality of the RL formalism and the complexity of wireless communication systems, there are many opportunities to use the MR-iNet framework to solve networking problems at many layers. For example, RL agents can be used to solve problems of power control, frequency selection, transmission control, channel estimation, etc.

Acknowledgement

This material is based upon work supported by the US Army Contract No. W15P7T-20-C-0006. Any opinions, findings, and conclusions or recommendation expressed in this material are those of the author(s) and do not necessarily reflect the views of the US Army.

Declaration of Competing Interest

The authors do not have any conflict of interest to claim.

Data availability

No data was used for the research described in the article.

References

[1] N.C. Luong, D.T. Hoang, S. Gong, D. Niyato, Pi. Wang, Y.C. Liang, D.I. Kim, Applications of Deep Reinforcement Learning in Communications and Networking: A Survey, IEEE Commun. Surv. Tutor. 21 (4) (2019) 3133–3174.

[2] A. Jagannath, J. Jagannath, T. Melodia, Redefining Wireless Communication for 6G: Signal Processing Meets Deep Learning with Deep Unfolding, IEEE Trans. on Artificial Intell. 2 (6) (Dec. 2021) 528–536.

[3] J. Jagannath, N. Polosky, A. Jagannath, F. Restuccia, T. Melodia, Machine Learning for Wireless Communications in the Internet of Things: A Comprehensive Survey, Ad Hoc Netw. 93 (June 2019) 101913.

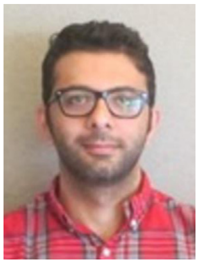
[4] J. Jagannath, K. Ramezanpour, A. Jagannath, Digital Twin Virtualization with Machine Learning for IoT and Beyond 5G Networks: Research Directions for Security and Optimal Control, in: Proc. of ACM Workshop on Wireless Security and Machine Learning (WiseML), San Antonio, Texas, USA, May, 2022.

[5] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction" A Bradford Book, Cambridge, MA, USA, 2019.

[6] J. Jagannath, K. Hamedani, C. Farquhar, K. Ramezanpour, A. Jagannath, MR-iNet Gym: Framework for Edge Deployment of Deep Reinforcement Learning on Embedded Software Defined Radio, in: Proc. of ACM Workshop on Wireless Security and Machine Learning (WiseML), San Antonio, Texas, USA, May, 2022.
[7] C. Farquhar, PSPV Kumar, A. Jagannath, J. Jagannath, Distributed Transmission Control for Wireless Networks using Multi-Agent Reinforcement Learning, in: Proc. of SPIE Defense and Commercial Sensing, Orlando, Florida, USA, May, 2022.

Collin works as an engineer and researcher in the fields of artificial intelligence and quantum computing. He worked as a Machine Learning Engineer at ANDRO where he focused on solving problems using reinforcement learning. He has published several papers in the fields of physics and computer science. He has an undergraduate degree from The College of Saint Rose and a master's degree in Applied Physics from Cornell University.

Dr. Swatantra Kafle is a research scientist at ANDRO Computational Solutions, Rome, NY. He received a B.E. degree in electronics and communications from I.O.E., Pulchowk Campus, Tribhuvan University, Nepal, and a Ph.D. degree in Electrical Engineering and Computer Science from Syracuse University, Syracuse, NY, USA, in 2011 and 2021, respectively. His research interests include statistical signal processing, compressed sensing, wireless communications, machine learning, and optimization.



Dr. Kian Hamedani is a senior scientist/engineer at ANDRO Computational Solutions LLC, Rome, NY. He received his Ph.D. degree in Electrical Engineering from the Bradley Department of Electrical and Computer Engineering (ECE) at Virginia Tech (VT). Prior to joining the ANDRO Computational Solutions LLC, he was a research assistant at the ECE department of VT. He spent 3+ years working at the Wireless@VT research group of VT on developing energy efficient machine learning (ML) approaches for wireless communication including dynamic spectrum sensing/sharing. His expertise and research interests mainly lie in emerging technologies for 5G cellular networks including ML/reinforcement learning (RL) for wireless networks, implementing, optimizing, and testing different communication protocols on software defined radios (SDRs), MIMO-OFDM, convex/non-convex optimization, and game theory. Dr. Hamedani has been the lead author and co-author of several papers in IEEE/ACM journals and conferences. He has been continuously reviewing papers for journals and conferences including IEEE TNNLS,IEEE TSG,IEEE JETCAS,EURASIP JWCN,IEEE Globecom, WIMBP, etc.

Anu Jagannath currently serves as the Founding Associate Director of Marconi-Rosenblatt AI/ML Innovation Lab at ANDRO Computational Solutions, LLC. She received her MS degree from State University of New York at Buffalo in Electrical Engineering. She is also a part-time PhD candidate and is with the Institute for the Wireless Internet of Things at Northeastern University, USA. Her research focuses on MIMO communications, deep machine learning, reinforcement learning, adaptive signal processing, software defined radios, spectrum sensing, adaptive physical layer, and cross layer techniques, medium access control and routing protocols, underwater wireless sensor networks, and signal intelligence. She has rendered her reviewing service for several leading IEEE conferences and Journals. She is the co-Principal Investigator (co-PI) and Technical Lead in multiple Rapid Innovation Fund (RIF) and SBIR/STTR efforts involving applied AI/ML for wireless communications. She is also the inventor on 6 US Patents (granted and pending).

Dr. Jithin Jagannath is the Chief Scientist of Technology and Founding Director of the Marconi-Rosenblatt AI/ML Innovation Lab at ANDRO Computational Solutions. He is also the Adjunct Assistant Professor in the Department of Electrical Engineering at the University at Buffalo, State University of New York. Dr. Jagannath received his B. Tech in Electronics and Communication from Kerala University; M.S. degree in Electrical Engineering from University at Buffalo, The State University of New York; and received his Ph.D. degree in Electrical Engineering from Northeastern University. Dr. Jagannath was the recipient of 2021 IEEE Region 1 Technological Innovation Award. He is also the recipient of AFCEA International Meritorious Rising Star Award for achievement in Engineering and AFCEA 40 Under 40.

Dr. Jagannath heads several of the ANDRO's research and development projects in the field of Beyond 5G, signal processing, RF signal intelligence, cognitive radio, cross-layer ad-hoc networks, Internet-of-Things, AI-enabled wireless, and machine learning. He has been the lead and Principal Investigator (PI) of several multi-million dollar research projects. This includes a Rapid Innovation Fund (RIF) and several Small Business Innovation Research (SBIR)s for several customers including the U.S. Army, U.S Navy, Department of Homeland Security (DHS), United States Special Operations Command (SOCOM). He is currently leading several teams developing commercial products such as SPEARLink®, DEEPSPEC^{TM} among others. He is an IEEE Senior Member and serves on the IEEE Signal Processing Society's Applied Signal Processing Systems Technical Committee. Dr. Jagannath's recent research has led to several peer-reviewed journal and conference publications. He is the inventor of 12 U.S. Patents (granted, pending, and provisional). He has been invited to give various talks including Keynote on the topic of machine learning and Beyond 5G wireless communication. He has been invited to serve on the Technical Program Committee for several leading technical conferences.