RLOps: Development Life-cycle of Reinforcement Learning Aided Open RAN

Peizheng Li, Jonathan Thomas, Xiaoyang Wang, Ahmed Khalil, Abdelrahim Ahmad, Rui Inacio, Shipra Kapoor, Arjun Parekh, Angela Doufexi, Arman Shojaeifard, *IEEE Member* and Robert Piechocki

***Abstract*—Radio access network (RAN) technologies continue to witness massive growth, with Open RAN gaining the most recent momentum. In the O-RAN specifications, the RAN intelligent controller (RIC) serves as an automation host. This article introduces principles for machine learning (ML), in particular, reinforcement learning (RL) relevant for the O-RAN stack. Furthermore, we review state-of-the-art research in wireless networks and cast it onto the RAN framework and the hierarchy of the O-RAN architecture. We provide a taxonomy of the challenges faced by ML/RL models throughout the development life-cycle: from the system specification to production deployment (data acquisition, model design, testing and management, etc.). To address the challenges, we integrate a set of existing MLOps principles with unique characteristics when RL agents are considered. This paper discusses a systematic life-cycle model development, testing and validation pipeline, termed: RLOps. We discuss all fundamental parts of RLOps, which include: model specification, development and distillation, production environment serving, operations monitoring, safety/security and data engineering platform. Based on these principles, we propose the best practices for RLOps to achieve an automated and reproducible model development process.**

***Index Terms*—O-RAN, machine learning, reinforcement learning, MLOps, RLOps, digital twins.**

I. Introduction

As the forefront of a mobile communication network, the Radio Access Network (RAN) directly interacts with the user equipment (UE). Its architecture has undergone profound changes within recent years transitioning from monolithic to disaggregated architectures and from vendor-based to open-source solutions [1]. The disaggregation of the RAN is reflected in two vectors, one is the horizontal disaggregation of the network functions with open interfaces, and the other is the virtualization of hardware and software in vertical. To achieve efficiency dividends, allow for increased innovation, and also performance gain, O-RAN[1] emerged from years of industry work in groups studying possible Open RAN trends (including 3rd Generation Partnership Project (3GPP)). O-RAN is based on 3GPP new radio (NR) specifications, meaning it is 4G and 5G compliant, while the difference is merely the additional interfaces that are defined by O-RAN, focusing on a functional split called 7.2x.

The emphasis of O-RAN has been on *openness* and *intelligence* from the beginning [2], through which it intends to actively embrace the technological revolution brought by machine learning (ML). Within recent years significant research has been undertaken demonstrating the potential of ML within telecommunications including channel estimation in massive multiple-input and multi-output (MIMO) systems [3], resource and service management in large-scale mobile ad-hoc networks [4] and mobile edge computation offloading and edge caching [5], to name but a few. The introduction of this class of methods is supported through a number of avenues but perhaps most importantly through the definition of open interfaces and radio intelligence controllers (RICs). Through their introduction, O-RAN provides the foundations by which ML models can be introduced into RAN. Thereby, facilitating the evolution of RANs from being static and stiff to data-driven, dynamically sensing and self-optimising. Notably, the exact mechanisms and procedures required to deploy cutting-edge ML solutions into production and realise their economic potential still needs clarification.

As ML models and systems continue to mature they are experiencing increased adoption within a range of industrial settings. The process through which they are developed and deployed is being formalised under the banner of MLOps [6]. Whereby, this is comparable to DevOps [7] and emphasises similar best practices whilst considering the unique challenges which the relevance of data in ML model development introduces. In order to reliably and consistently bring the potential of ML to O-RAN, an operational platform implementing MLOps principles whilst considering the unique challenges of RANs is required. These challenges pertain to its prominence as critical national infrastructure and the highly dynamic nature of the platform. We place particular emphasis on the challenges of Reinforcement Learning (RL) within O-RAN due to the numerous applications that exist for it and its relative immaturity in terms of industrial applications. Where we discuss key elements throughout the applications lifecycle including design considerations, challenges pertaining to training within the simulation and effectively monitoring live deployments, to name but a few. This pipeline and associated

Peizheng Li, Jonathan Thomas, Xiaoyang Wang, Ahmed Khalil, Angela Doufexi and Robert Piechocki are with the Department of Electrical and Electronic Engineering, University of Bristol, UK (e-mail: {peizheng.li, jonathan.david.thomas, xiaoyang.wang, oe18433, A.Doufexi, R.J.Piechocki}@bristol.ac.uk).

Abdelrahim Ahmad and Rui Inacio are with Vilicom UK Ltd. (email: {Abdelrahim.Ahmad, Rui.Inacio}@vilicom.com).

Shipra Kapoor and Arjun Parekh are with Applied Research, BT, UK (email: {shipra.kapoor, arjun.parekh}@bt.com).

Arman Shojaeifard is with InterDigital Communications, Inc. (email: arman.shojaeifard@interdigital.com).

[1]In this paper, O-RAN is taken as the reference architecture to demonstrate the validity of the proposed RLOps, but the principles can be applied to other RAN architectures.

set of principles is coined *RLOps*.

In this paper, we introduce principles and best practices of RLOps in the context of an O-RAN deployment [2]. To the best of our knowledge, this is the first work to systematically discuss the life-cycle development pipeline of ML, especially RL models in O-RAN and put forward a network analytics platform in accordance with RLOps. We explain the fundamental principles and highlight critical factors involved in RLOps. In Section II, we briefly introduce ML and RL, the evolution of RAN and O-RAN architectures are given in detail. Next, we discuss some related applications of ML/RL in intelligent O-RAN, and correspondingly a series of challenges encountered in the development and deployment stages of ML/RL models. In Section III, we elaborate on the principles of RLOps from the perspective of design, deployment and operations. We highlight the safety and security concerns in RLOps. We design a data engineering platform to cooperate with the proposed RLOps. In Section IV, we put forward the effective routines and best practices of operating the aforementioned principles from the view of digital twins, automation and reproducibility. Finally, Section V concludes this paper. Table I gives the list of used acronyms.

II. Relevant Background

A. Machine Learning in general

Machine Learning (ML) is a branch of Artificial Intelligence (AI) concerned with learning from data, e.g. supervised learning (SL) and unsupervised learning (UL), or interaction, e.g. RL [8]. In general, ML considers the utilization of an adaptive model parameterized by θ with the intention of minimizing some objective function $\mathcal{J}(\theta)$. The exact objective function and form of the adaptive model depend on the exact formulation of the task (or set of tasks) we are interested in.

Within SL tasks, we are typically presented with a dataset $\mathcal{D} = \{x_i : y_i\}_{\{i \in |\mathcal{D}|\}}$ comprising of feature vectors x_i, which are labelled y_i which may refer to a discrete categories (cat or dog, for example) in a classification task or a real number if it is a regression task. Within this class of problem the objective is to learn a mapping $\mathcal{F}_\theta : \mathcal{X} \rightarrow \mathcal{Y}$. A typical formulation of our objective function is the minimizing of Negative log-likelihood in the case of classification or the Mean Squared Error in the case of regression tasks.

Like SL, in UL we are typically presented with a dataset $\mathcal{D} = \{x_i\}_{\{i \in |\mathcal{D}|\}}$, but in this case there are no labels. When presented with a task of this nature, we may be interested in clustering [9], density estimation [10] or in dimensionality reduction for visualization [11].

RLs interaction with data is fundamentally different from other forms of ML. Typically, the problem is formalised as a Markov Decision Process (MDP), where this is defined by the tuple $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$. Where $\mathcal{S}$ is the set of environment states, $\mathcal{A}$ is the set of actions that an agent performs, $\mathcal{P}$ represents the transition probability from any state $s \in \mathcal{S}$ to any state $s' \in \mathcal{S}$ for any given action $a \in \mathcal{A}$. $\mathcal{R}$ is the reward function that indicates the immediate reward received from the transition from s to s', and γ is the discount factor that trades off the instantaneous and future rewards. The intention is to find a policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ which maximises the expected cumulative discounted reward $\mathcal{G}$ [12] as defined in Equation 1.

$$\mathcal{G} = \mathbb{E}[\sum_{t \geq 0} \gamma^t \mathcal{R}(s_t, a_t, s_{t+1} | a_t \sim \pi(\cdot | s_t), s_0)] \tag{1}$$

The process of finding this π requires exploratory behaviours such that the agent can evaluate policies and learn about the MDP. The parameterization of the adaptive model may vary; for example, in model-free algorithms, we may parameterise our π directly or the state-action value Q, or in model-based algorithms, we may learn a model of the MDP directly [2].

B. Evolution of the RAN and O-RAN architecture

A typical mobile communication network mainly comprises a RAN, a transport network and a core network. The RAN gives the UE access to the core network, this subsequently provides the services to the user. The transport network implements the IP routing and IPSec functionality that securely connect the different network elements and network domains of the mobile network, thus allowing for full end-to-end functionality. From 1G to 5G, the evolutionary trends of communication systems is the modularity and virtualization of decoupled network functionalities. For instance, the core network embraces x86 platform universal servers and performs network function virtualization (NFV), where the slicing of the core network embodies this feature. However, due to the complexity of antennas, the Remote Radio Unit (RRU), and the Baseband Unit (BBU) in RAN, the functionality decoupling of RAN is slower than the decoupling of the transport network and core network. Three distinct structural improvements had been proposed in the evolution of RAN, namely the distributed RAN (D-RAN), the centralized (or cloud) RAN (C-RAN), and vRAN. In D-RAN, the RRU and BBU are co-located at every distributed cell site. Cells are connected back to the core network through the backhaul interface. In C-RAN, all BBUs are further concentrated into the centralized BBU pool for cloudification, and every site merely keeps antenna and BBU. RRUs and the centralized BBU are connected with fronthaul. Centralized BBUs brings the convenience of cell deployment and maintenance and significantly reduces the CAPEX and OPEX. vRAN decouples the software and hardware by NFV, where the BBU is virtualized on x86 servers [13]. In the 5G network, the above BS components are reorganised into the centralized unit (CU), distributed unit (DU) and active antenna unit (AAU), with their deployment following a flexible topology. In the meantime, all the hardware design, specialized software development and intellectual properties of the RAN related components are mastered by the equipment vendors.

Network operators expect to obtain decoupled, standardized RAN hardware and open-source operating software to relieve current vendor restrictions. Consequently, the O-RAN alliance was founded in February 2018. Its ambitious mission is to reshape the RAN industry, building future RANs on a foundation of virtualized network elements, white-box hardware and standardized interfaces. The core principles of O-RAN

[2]Can be combined with model-free approaches.

TABLE I
LIST OF ACRONYMS

Acronym	Definition	Acronym	Definition
3GPP	3rd Generation Partnership Project	ML	Machine Learning
5G	5th Generation Mobile Network	mMTC	Massive machine-type communications
AAU	Active Antenna Unit	NE	Network Elements
AI	Artificial Intelligence	Near-RT RIC	Near-real Time RAN Intelligent Controller
API	Application Programming Interface	NFV	Network Function Virtualization
BBU	Baseband Unit	NLP	Natural Language Processing
CAPEX	Capital Expenditure	Non-RT RIC	Non-real Time RAN Intelligent Controller
CPO	Constrained Policy Optimization	OFDM	Orthogonal Frequency-division Multiplexing
CI/CD	Continuous Integration (CI) and Continuous Deployment	OPEX	Operating Expense
CMDP	Constrained MDP	O-RAN	Open RAN
C-RAN	Centralized (or Cloud) Radio Access Network	PL	Processing Layer
CSI	Channel State Information	QoE	Quality of Experience
CU	Centralized Unit	QoS	Quality of Service
CV	Computer Vision	RAN	Radio Access Network
DCA	Data Collection Agents	RF	Radio Frequency
DT	Digital Twins	RIC	RAN Intelligent Controller
DL	Deep Learning	RL	Reinforcement Learning
DML	Data Mediation Layer	RNN	Recurrent Neural Network
D-RAN	Distributed RAN	RRU	Remote Radio Unit
DRL	Deep Reinforcement Learning	RU	Radio Unit
DSL	Data Storage Layer	SDN	Software-defined Networking
DU	Distributed Unit	SL	Supervised Learning
eMBB	Enhanced mobile broadband	SLA	Service Level Agreement
ETSI	European Telecommunications Standardization Institute	SMO	Service Management and Orchestration
FCAPS	Fault, configuration, accounting, performance, security	SON	Self Organizing Network
FDD	Frequency Division Duplexing	SOTA	State of the Art
IID	Independent and Identically Distributed	TTI	Transmission Time Interval
IP	Internet Protocol	UAV	Unmanned Aerial Vehicle
IRS	Intelligent Reflecting Surfaces	UE	User Equipment
LSTM	Long Short Term Memory	UL	Unsupervised Learning
LDPC	Low-density Parity-check Code	URLLC	Ultra-reliable low-latency Communications
MDP	Markov Decision Process	V2X	Vehicle-to-everything
MEC	Mobile Edge Computing	VNFs	Virtual Network Functions
MIMO	Multiple-input and Multiple-output	vRAN	Virtual Radio Access Network

are intelligence and openness, which will lead the direction beyond 5G and 6G. Figure 1 demonstrates one example architecture of O-RAN. O-RAN architecture follows 3GPP architecture and interfaces specifications, while its NFV is as consistent as possible with European Telecommunications Standards Institute (ETSI). The service management and orchestration (SMO) function in O-RAN has been designed to provide network management functionalities for the RAN and may also be extended to perform core management, transport management, and end-to-end slice management. Meanwhile, the SMO connects with O-Cloud through the O2 interface. O-Cloud is a cloud computing platform comprising a collection of physical infrastructure nodes that meet O-RAN requirements to host the relevant O-RAN functions, the supporting software components and the appropriate management and orchestration functions [2]. One important functionality provided by SMO is the Non-RT RIC designed to implement automated policy-based optimization activities by running ML models. The Non-RT RIC links towards Near-RT RIC via A1. The Near-RT RIC controls and optimizes the functions of CU and DU through the E2 interface. Meanwhile, third-party, microservice architecture based applications can also be loaded into the Non-RT RIC and Near-RT RIC through rApps and xApps, respectively, to perform data-driven optimization behaviours. In this process, E2 can be leveraged to access the radio node data, and these data can be fed into RICs for ML models training. The CU connects to or controls one or more DUs via the F1 interface. Similarly, one DU connects to at least one radio unit (RU) through the open fronthaul plane. The CU/DU stack hierarchically handles operations of different timescales, while the RU manages and controls the most fundamental radio frequency (RF) components and the physical layer in every RU deployment site. All functions of the O-RAN, including the Near-RT RIC, CU, DU and RU, are connected to the SMO through the O1 interface for FCAPS support.

It is noticeable that three control loops involving system parameters and resource allocations are defined in O-RAN. ML solutions can be adopted in any loop based on the time-sensitivity of tasks, in which loop1 handles operators at the time scale of TTI level (<10 ms) for those scenarios that emphasize real-time like the radio resource control and allocation happened in between DU and RU; loop 2 operates in the Near-RT RIC which deals with tasks operating within 10-500 ms. It mainly aims to the O-RAN internal resource control, which RICs perform; loop 3 operates in the Non-RT RIC to process tasks greater than 500 ms.

C. ML/RL applications in O-RAN

ML is undoubtedly the most remarkable technological progress in recent years. From CV [14], NLP [15] to robotics [16], gaming [17], e-commercial [18] and biology [19] etc.

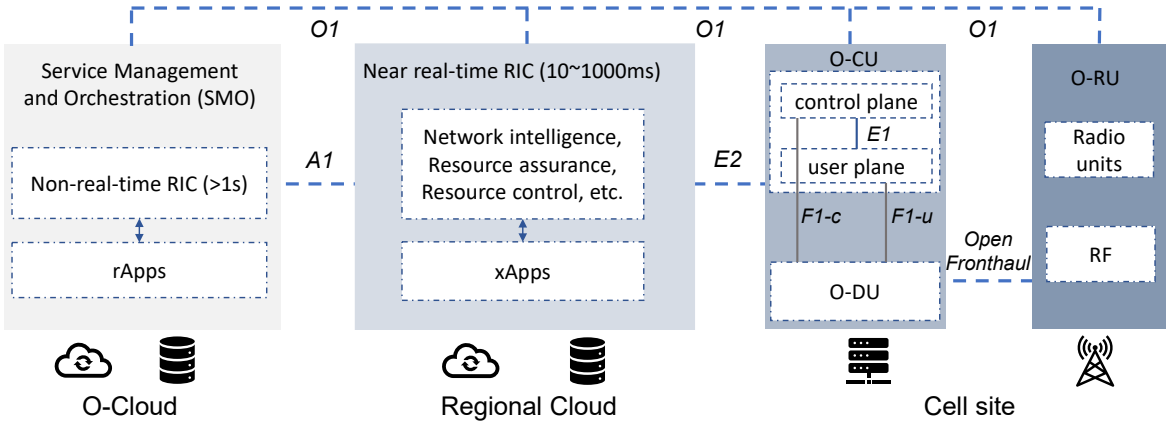


Fig. 1. One example architecture of O-RAN.

ML applications in almost every technical field have made marvellous achievements. Also, the upcoming O-RAN through the introduction of intelligent programmable RIC enables the RAN to have a mechanism to use emerging learning-based technologies to automate network functions, improve network efficiency, and reduce operating costs. In O-RAN, the initially closed internal radio resources are opened and controlled by unified RICs. That brought some profound changes to communications studies.

1) With higher mobile edge computing (MEC) capability, O-RAN enables interaction with end-users, such as directly perceiving end-users behaviours and responding to them so that the optimization of the network can be completed from a more fine-grained and a more direct user model analysis way, without the need to perform it in the core network or the centralized cloud.
2) O-RAN can significantly help the further promotion of 5G. As often mentioned, the main goals of 5G are enhanced mobile broadband (eMBB), ultra-reliable low-latency communications (URLLC) and massive machine-type communications (mMTC) [20]. Due to the complex and diverse environmental conditions faced by 5G networks, it is necessary to allocate resource blocks with network slicing to meet task requirements for different application scenarios. The introduction of O-RAN makes a dynamic, learning-based slicing mechanism possible. Therefore, deep learning-based adaptive slicing, the collaboration of SDN and NFV, is becoming a research hotspot.
3) RICs provides a platform for third-party applications deployment, including ML models, enabling the rapid development and deployment of innovative ideas and algorithms.

The O-RAN use case whitepaper [2] described some of the AI-based deployment targets, such as service level agreement (SLA) assured 5G RAN slice, context-based dynamic handover management for vehicle-to-everything (V2X), and flight path based dynamic unmanned aerial vehicle (UAV) resource allocation etc., while we believe the potential of AI-enabled O-RAN is far more than that. The state of the art communication system embodies a feature of hierarchical and self-contained functions. All functions are interconnected with standardized interfaces. For instance, the signal undergoes a series of units from the transmitter to the receiver, such as modulation, coding, demodulation, de-noising, and corresponding channel measurement. Each unit has a well-defined mathematical model that can approach the Shannon limit, and it can be considered that a single unit has achieved its local optimum. However, there are significant challenges in the analysis and optimization of cross-units. If the whole of the above units is regarded as the optimization object, then this kind of global or multi-objective optimization is currently challenging to achieve [21]. The combination with ML/RL various learning paradigms makes O-RAN have the potential for this overall or multi-objective optimization revealed in some advanced research. For instance, in the physical layer, the DL-based OFDM receiver can achieve accurate channel estimation using fewer pilot signals [22]; the end to end learning of communication systems has been realized in an autoencoder way which shows advantages in synchronization, equalization and dealing with hardware impairments such as non-linearities [23]; the BS downlink channel state information (CSI) in frequency division duplexing (FDD) massive MIMO system can be inferred by DL with feeding the downlink CSI under certain conditions [24]; under the premise of imperfect CSI, the design of hybrid massive mimo digital precoder and analog combiner based on RL [25]; and a variety of DL-based LDPC decoding solutions under harsh noise [26]. In the network layer, the learning-based algorithms shape the SON with dynamic resources allocation properties like automated networking, slicing, dynamic spectrum sensing, random access channel, and load balancing optimization in the network layer. It is to be noted that with the O-RAN stepping into the market gradually, some RL-based optimization cases targeting ORAN's features are beginning to appear. [27] describes the RL scheduling control in sliced 5G networks by using O-RAN data collected from real-world cellular deployments. [28] shows the scheme of effective energy using through RL dynamic function splitting. [29] demonstrates an application of using RL for computational resource allocation between RUs and DUs, which has the potential to reduce power consumption significantly.

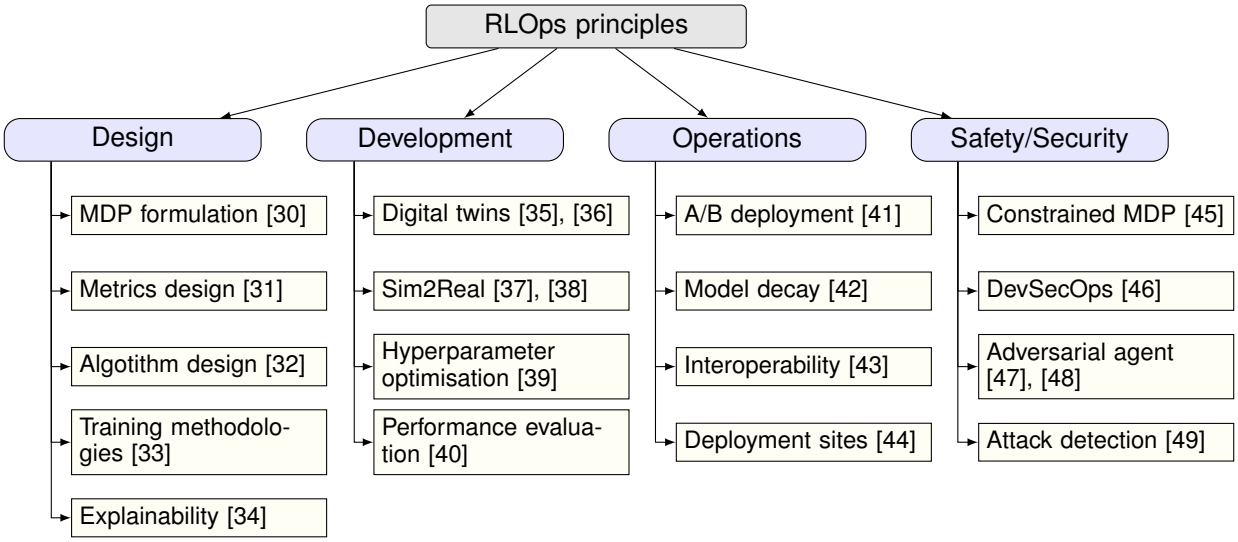


Fig. 2. This figure lists the critical elements involved in the principle of RLOps. That is the design, development, operations and safety/security, then we further break down each element into the high-level taxonomy of considerations and methodologies.

D. Challenges of ML/RL developing in O-RAN

Although the hierarchical structure and decoupling characteristics of O-RAN have brought the benefits of supplier diversification, this also brings in higher complexity in the deployment of O-RAN. On the other hand, developing a suitable intelligent model and deploying it in O-RAN may lead to practical engineering technology problems. Considering the most vigorous ML domains in CV and NLP, some standard data sets are generally used to evaluate the performance of the developed ML algorithms. These algorithms are designed to target the features of the given training sample. For example, for the image sample, the initial features of the image space are extracted from adjacent pixels through various convolution operations, and then through various sophisticated network structures such as AlexNet, VGG, ResNet, the features are further refined, and the mapping from the training space to the target space is accurately constructed. In the booming RL field, whether it is gaming or robot control, the development of related algorithms is basically carried out under a standard toolkit like OpenAI gym [50]. A noteworthy phenomenon is that the fields mentioned above benefit from the support of solid mathematical models and complete underlying software. The development of involved ML has been systematically transformed into a near-standard industry. These models corresponding to different application scenarios are well defined, making the goal of algorithm development precise and the whole process controllable.

Turn our attention to the application of ML in O-RAN. The state of the art progress made by the current O-RAN alliance is summarised as follows. (1) The programmable and expandable RIC modules are introduced into the O-RAN architecture, and an interface for data collection within the network is defined. (2) With the clarification of the structure definition, a series of ambitious optimization or control goals for resource, traffic flow, and power consumption have been proposed. (3) The workflow of using SL and RL has been standardized. However, the above progress only reflects the possibility of O-RAN embedded ML in a broad and macro sense. Specific to the realistic implementation of the ML models, we will encounter a rather complicated situation. We further consider issues of ML in O-RAN from algorithms development and deployment angels, respectively. From the view of algorithms development, we summarised the potential issues as below:

1) In O-RAN, data related to model training is difficult to obtain and process. Even the standard interfaces defined in the O-RAN architecture, such as E2, can access DU, CU and other components to collect information inside the network. This data comes, by default, in raw format and without a schema that is not suitable to be directly consumed by ML/RL algorithms. If we intend to use this field information to train the model, the cost of data collection will be very high.
2) For different optimization goals, the required data for neural network training is heterogeneous. The attributes or patterns of various types of data hiding are elusive. For example, for radio traffic, the data flow as a whole is usually non-Euclidean. In some RU-distributed sites, the data does not meet the characteristics of independent and identically distributed (IID), and some data sets have very strong temporal correlations, while the correlation of other data sets is more reflected in the spatial domain. That will pose challenges to the subsequent data processing methods and feature extraction schemes, affecting the overall neural network structure design.
3) Some global optimization problems demonstrate the applicability of RL. That poses other challenges for establishing the connection between O-RAN and RL. These challenges are often not about the RL algorithm itself but how to abstract the problem to be solved into the RL framework and define the RL related environment, action, state, reward. For instance, the training issue comes along with high-dimensional state and action spaces; the availability of offline model trained from

historical logs; the feasibility of online model training but with limited samples or partial observations; the large reward delay or vanishing in RANs; the complexity of multi-agent RL scheme for optimization problems across multiple RANs.

4) The RAN is the entrance to the entire wireless network and is closest to UEs. Therefore, the data flow in O-RAN is inevitably directly related to UEs. If we want to use these data streams to train neural network models, new requirements will be put forward for the privacy protection of the UEs and the desensitization of related data.

We have introduced that xApps are connected to Near-RT RIC in O-RAN as the host of trained ML models. These trained models are pre-stored in O-cloud and managed by the SMO. However, from the view of model deployment, the above system is not enough to overcome the problems that may arise after the model is deployed in the field. On the one hand, the models obtained by SL were trained by specific data sets. After deploying these mature models, one possible consequence is that the sample data characteristics in the model deployment area are inconsistent with the characteristics of the original training set, which will result in model failure, that is, the expected results cannot be correctly received, as the model can not respond to the input features. On the other hand, for the RL model, as time changes, the external environment changes continuously, which will make the initially trained policy no longer suitable. That puts forward new requirements for model management, update and maintenance, and we must look at O-RAN and its ML models from a more holistic perspective.

III. Principles of RLOps

A. Brief introduction of MLOps

MLOps is defined as a set of practices that combines ML, DevOps and data engineering, aiming to deploy and maintain machine learning models in production reliably and efficiently. It can be seen as delivering ML applications through DevOps, with additional attention on data and models. MLOps performs the idea of *automation* and *acceleration*. *Automation* means automating the ML pipeline from data to model for continuous training, as well as automated CI/CD for ML applications. *Acceleration* means to increase the speed of delivery while maintaining the quality of service for ML applications [51].

An MLOps pipeline usually consists of the following elements:

1) Data preparation and model design.
2) Model testing and validation.
3) Model integration, delivery and monitoring.
4) Continuous training and CI/CD.

Similar to DevOps, MLOps is an iterative approach. The change of developing requirements, the evolution of the deployment environment and the alerts raised by monitoring the deployed model would trigger the execution of the pipeline to guarantee the quality of ML applications.

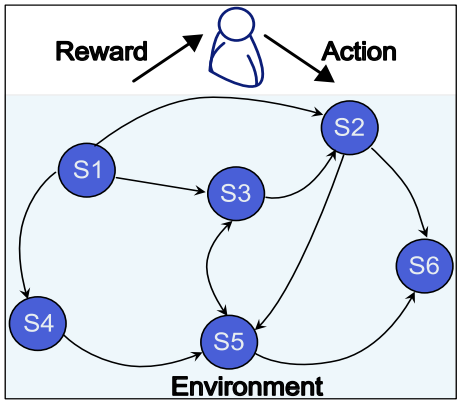


Fig. 3. An example of RL.

B. Motivation for RLOps

As introduced in Section I, MLOps is the general principles and practices of continuous delivery and automation pipelines in ML. Considering the increasing applications of RL in communication networks, we study the "RLOps" principles to deliver the value of RL to the industry.

RL differs from other ML approaches in several ways, which brings the need for more targeted principle sets. As shown in Fig. 3, *Data & Environment*, *Agent* and *Reward* are the key distinctions considered in the design and delivery of RL applications.

- *Data & Environment*. Data is considered the backbone of ML practices. In RL, data is from agents interacting with environments (online RL) or pre-collected datasets (offline RL) [52]. For online RL, the interaction and learning from live environments (in our case, live communication networks) brings additional risks (as we discussed in Section II-D), which is infeasible in most cases. Communication networks bring challenges to environment access, training data acquisition and model validation. A network analytics platform with automated data collection, preprocessing, model validation and management abilities are proposed and discussed in Section III-H. Furthermore, the idea of digital twins (DT) has been brought up as a promising solution to the environment and data issue of RL practices [36], providing a controllable, reliable and easily accessible simulation environment. We elaborate it in Section IV-A.
- *Agent*. Agents are the core of RL problems, interacting with the environment following their policies. The policy that agents perform is the brain of MDP solutions, as counterparts for "models" in SL and UL. The general principles for developing and deploying ML models also apply to RL models, including model analysis, testing and monitoring.
- *Reward*. The reward is unique to MDPs. It represents the goal of RL, which is essential information to have in model design and deployment. Unlike "labels" as intrinsic features of data in SL, rewards reflect the expected behaviour of agents. In RL applications, reward design is always part of the problem formulation, which requires

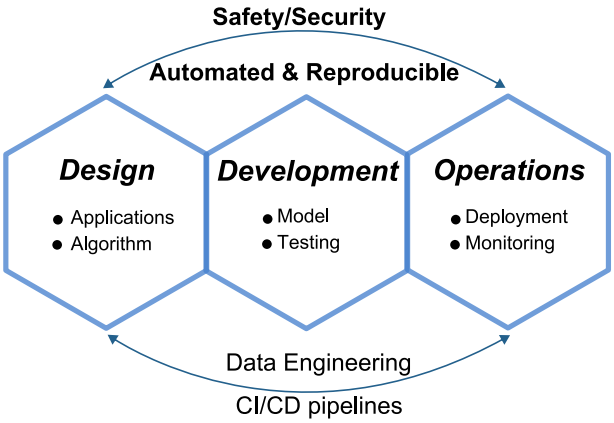


Fig. 4. RLOps diagram. This diagram demonstrates the development cycle of applications.

special attention in RLOps.

To effectively deploy RL applications requires careful navigation through a wide range of decisions, from problem formulation to algorithmic choices to the selection of monitoring metrics, to name a few. In an attempt to demystify these decisions, we introduce a non-exhaustive list of "RLOps" principles and observations, which we consider helpful in realizing the potential RL promises. The intention is to provide distinct but complementary ideas for RLOps to what may be expected in MLOps and DevOps. For an overview of key considerations and principles for MLOps please refer to [53].

We introduce principles of RLOps under the application development cycle introduced in Figure 4[3]. Below we talk about the three parts: design, development and operation. We will also elaborate on the safety and security concerns and data engineering processes related to these three parts. A summary of the high-level taxonomy of considerations and methodologies involved in the RLOps principles is shown in Figure 2.

C. Design

There are a number of essential steps within the design phase, including identifying constraints (related to performance or available resources, for example), problem formulation, and algorithm design.

1) Task Formulation: Consider the arrival of a new task that takes the form of sequential decision-making, as such RL is likely to be a good solution. Examples of these tasks are given in Section I including handover and interference management to name but a few [54].

An integral step to build a solution based on RL is to formulate the given problem as an MDP. The formulation of MDP affords many degrees of freedom. Each design decision should be considered carefully as the form of the MDP will dictate a number of algorithmic decisions. Basic elements to consider include the number of agents, the representation of actions the degree of stochasticity of the environment. For example, if the problem requires distributed decision-making, a stochastic game [55] may be an appropriate formulation; If the hierarchical representation of the action space is possible, and options framework [56] may be possible.

Related to the formulation of the MDP is understanding how the data about the model experiences will be distributed. As discussed in [53], understanding this is useful when defining a data schema that can later be used for monitoring purposes. In contrast to SL tasks, the data which the agent will experience is dependent on its behaviour so defining a schema likely requires understanding this interplay.

As part of the task formulation phase, it is useful to consider evaluation metrics and baselines that are suitable for the task, where these baselines may be existing solutions to the task. This will smooth out the test, validation and monitoring phases in RLOps, and potentially provide a fail-safe if the RL application begins to behave erratically.

2) Algorithm: As discussed in the above section, decisions on the formation of MDP directly impact the form of the solution. Some of the design practices are listed below but many other general rules exist. [57] provides a good analysis of the impact of some design choices on specific RL algorithms.

- If the action space is continuous, policy gradient-based approaches are likely to be a good option. Some discretisation could also be applicable, such as Q-Learning variants.
- If the state is non-markovian, recurrency can be introduced through stacking previous states[58] or RNN structures like LSTM [59].
- Particularly, small state-action spaces may be amenable to tabular approaches [60] which provides a higher degree of interpretability over methods that use function approximation.

The training strategies and tricks are also worth considering in the design phase, to tackle problems like *model generalisation* and *training difficulty*. Considering potential varying environments for deploying RL solutions, we are interested in how the model generalizes and as such our training methodologies should reflect this. It has become a consensus in RL research that models trained within limited instantiation of environments do not generalize well [37]. The utilization of methods like "Domain Randomisation" is essential for model generalisation. As for the training difficulty, the utilization of a training curriculum, where a series of increasingly complex tasks are presented to the agent with the intention of easing learning on difficult tasks [61]. Imitation Learning is another approach to ease the learning process, [62] where an agent is pre-trained with pre-collected dataset containing expert behaviours [52].

In addition to these design choices, we may wish for our algorithm to possess other characteristics. For example, we may want its decisions to be explainable or for it to be aware of its uncertainty with regard to its state. That moves us to other sub-areas of RL researches like *Explainability* [34] and *Bayesian RL* [63] which deals with these concepts that are important from business, legislative or even safety perspectives.

[3]Inspired and adapted from https://ml-ops.org/img/mlops-loop-en.jpg.

D. Development

Once elements of the *design* have reached sufficient maturity levels, steps can be taken towards formally developing the application's capabilities. This process involves creating the experimental environment (which will likely be based on a DT), model training, and performance optimization.

1) Model: The algorithmic approach defined in the Section III-C2 provides the general structure and algorithm for the model. The next step is to develop the necessary code for the agent. To code everything from "scratch" may seem reasonable but may lead to significant engineering expense for limited gain, especially when a wide array of readily available open-source libraries provide high-quality implementations of a range of SOTA algorithms exist[4].

Training RL applications in a time-efficient and comprehensive manner requires accessibility to a high fidelity simulator - where a DT will likely be a good fit for this[5]. If we take a pessimistic viewpoint, the DT or any simulator is an approximation to the real world, and as such, there will be inconsistencies in behaviour that may, in the worst case, lead to testing values being inconsequential as the differences are so profound that the policies are not transferable. This is a *Sim2Real* challenge and is considered in more detail in Section IV-A.

Once an effective algorithmic and simulation approach have been developed to address this challenge, the next major obstacle in the model development process is hyperparameter optimization, which is an arduous and time-consuming process. In the interest of efficient allocation of resources, this process will benefit from automation.

2) Testing: In the life circle of DevOps, testing is essential to ensure the performance of software systems. Code sanity testing, unit testing and integration testing are commonly used to validate the software iteration. In MLOps [64], the scope of testing extends to data and models. Here we re-consider testing in the context of deep reinforcement learning (DRL) in future O-RAN.

Once a DRL model has been trained, we require functionality within our pipeline to evaluate the model's capabilities. For trained DRL models, testing should consider multiple model attributes to give a comprehensive evaluation of the models' performance. Some dimensions are also considered in MLOps, such as the model relevance and accuracy, the robustness to noise, the generalization ability and ethical considerations [6]. Other challenges are unique to DRL models, for example, the ability for a DRL model to prioritize useful experiences during learning, to choose long-term beneficial actions, to respond to uncertainty, stochasticity and environmental changes, to avoid unintended behaviour, etc. The testing and validation of DRL models regarding the dimensions mentioned above remain an open question, leaving space for future work. DT might play an important role in the testing procedure since it is an environment in which we have complete control. Manual testing might be required in some use cases. In addition, model interpretability and explainability are of great importance from the perspective of both developer and network service providers, which should be considered during testing. Considering possible network attacks and security challenges, an adversarial attack should also be integrated into the model testing workflow.

[4]Ray and OpenAI Baselines, to name but a few

[5]The general structure of which should follow that mandates by OpenAI Gym for consistency with other open-source platforms

E. Operations

Assuming that a model has passed all required testing and validation steps and has been containerized according to system requirements, the obvious next consideration should be for model deployment and the associated systems required to support and maintain it. This process will include consideration for the deployment location and monitoring with the intention of providing functionalities for continuous improvement.

1) Deployment: A fundamental issue (which is discussed at more length in Section III-E2) is that of discrepancies that exist between development and production environments. This problem is likely to be ever-present and difficult to quantify. As such, other safeguards are likely required to mitigate this risk before wide-scale deployment. An example of this could be using software development practices like alpha-beta type deployments to limit the potential impact on end-users whilst getting an empirical measure of application performance.

The environment to which the agents are deployed tends to be highly dynamic, where changes are likely to alter network behaviours. These changes may include internal factors like device configurations and the deployment of other applications or external factors like changes in user behaviour or seasonal phenomena that may affect wireless propagation characteristics. The manifestation of this dynamic environment is a modification to the underlying MDP, and the performance of RL agents will likely degrade accordingly. This issue is one of the concept drifts [42]. The implication of the dynamic nature of the deployment environment is that the performance of deployed models may reduce over time. This general phenomenon is known as *Model Decay* and will be observable through the agent's reception of reward. The impact of this can be mediated through periodic re-training if the reward drops below some pre-defined threshold. An alternative approach is to enable online training, but this does come with risks, most notably the requirement for exploration. An additional risk that may arise is non-stationarity [65], where is a consequence of a deployment consisting of multiple RL agents constituting a Multi-Agent system. Non-stationarity arises when multiple agents are learning policies simultaneously, resulting in uncertainty regarding environment behaviours as state transitions are implicitly dependent on other agents.

To enable interoperability on differing base computational platforms all applications will need to be containerised with their associated internal dependencies for deployment with a platform like Kubernetes [43]. A well-defined REST application programming interface (API) will allow for communication of information between entities such that applications can obtain external information that they require for operation and so that monitoring can be performed and decisions can be made pertaining to applications. Communication between

disparate systems within the O-RAN architecture naturally raises considerations for model deployment location. By selecting an appropriate location (be that topological or cloud vs edge) and control loops described in Section II-B, application performance and the wider performance of the network may be improved, where benefits are related to reduced inference time and a reduction in network traffic due to co-location of applications with their dependencies. These decisions may be particularly important for applications that require very low latency for effective operation.

2) Monitoring: Through a collection of Key Performance Indicators (KPIs), the efficacy of an RL agent can be monitored. This information enables decisions pertaining to the application to be made in an informed manner. For example, if an agent is underperforming, it may be desirable to re-train or even replace the agent with an alternative solution.

Monitoring and evaluating RL application performance in the real world is critical to determining whether or not the application is providing benefit, but this is likely to be challenging. Simple measures like cumulative reward can be utilized but are susceptible to issues like reward hacking [66] and do not provide relative measures compared to other methods. The most thorough approach from a network operators perspective may be to have human oversight of the decisions that agents are making, but this is not scalable and is likely to be problematic as RL agents are often difficult to interpret. Consideration of concepts like *Explainability* [67] is likely to be essential in providing the necessary administrative oversight which may be necessary from both a risk and governance perspective. The most appropriate strategy is likely to involve an ensemble of methods including collating a range of metrics that attest to the applications performance characteristics. These measures may include application-specific measures, like throughput and latency for a resource allocation application and include periodic utilization of AB testing to provide a relative measure against well-understood baselines.

In addition to the impact on reward acquisition, changes within the environment in which the RL agents exist may impact the computational performance of the model [53]. Metrics pertaining to model performance like inference time, throughput, and RAM usage will be important in identifying transient behaviours.

F. Versioning

Versioning, or source control, is the practice of tracking and managing changes during development. O-RAN brings the opportunity to use software-based RICs with open interfaces widely. Flexible and fast iteration software development requires careful versioning, and this also applies to RLOps in O-RAN.

1) Data: The data preparation in RL is different from SL or UL, as it comes from interacting with the environment. For communication network applications, data could come from either a running network or a DT. Live network data can be stored and versioned by data management tools like DVC [6], Pachyderm [7] or other built-in tools in ML development frameworks. These tools attach version information to datasets. For artificial data generated by a DT, it is more efficient to give snapshots of the DT, including the simulation scenario, the configuration, the random seeds, etc. Given the versioning information of the DT, we should be able to reproduce the same dataset if needed.

2) Model: Versioning of the model is vital for controlling the model deployment, especially when facing environment changes or unexpected failures. Since the training pipeline of RL models for O-RAN takes both live network data and DT, it is important to version the training environment and pipeline as well as the model itself to trace back this self-learning approach. This includes the versioning of training configurations, production environment, and the versioning of DT and network data mentioned in the previous section. The hyperparameters that correspond to each model should also be versioned.

3) Code: All the production code during development and deployment should be put into versioning. This includes the code to train the RL model, the code for testing and validation, the code for successfully deploying the trained model, and the application code. In addition, as the training of RL in O-RAN uses DT, the code for the DT development and deployment should also be versioned. The DT itself can be seen as a standalone project which requires proper source control [36].

G. Safety and Security in RLOps

Considering ML/RL applications in O-RAN, model safety and operation security are critical. The former can be dealt with by introducing safety constraints into the *Design* and *Development* process. For the operation security, we discuss some principles to follow, inspired by the DevSecOps [46], which integrates security measures into the DevOps cycle.

For RL models running on wireless networks, *Safety* is important for service assurance as well as avoiding catastrophic performance decay. In the exploratory learning phase, a common approach is to consider potential safety restrictions that exist in the environments, agents, and actions in advance and formalize them into a Constrained MDP (CMDP), which define a constrained optimization problem as shown in equation 2. A safety policy is expected to achieve by training on the CMDP [68].

$$\max_{\pi \in \Pi} \mathcal{G}(\pi) \quad s.t.\ C^k(\pi) \leq V_k, k = 1, \cdots, K \tag{2}$$

where $\mathcal{G}$ is the cumulative discounted reward of a policy π, $C^k(\pi)$ reflects the cumulative cost incurred by constraint k on a given policy π. Specifically, C^k can be defined as $c_k(s, a)$ which represents the possible constraint in terms of state s and action s. [68] presents one solution to the CMDP, which is called Constrained Policy Optimization (CPO). It searches for the policy which maximizes the reward and approximately satisfies the given constraints, i.e., safety requirements. In [69], the sample efficiency in CMDP is further studied under the

[6] https://dvc.org/

[7] https://www.pachyderm.com/

model-based manner. Robust MDP has also been considered in the scope of CMDP, leading to a robust soft-constrained solution to the Robust-CMDP problems [70].

Security in communication networks protects the integrity of the system, including but not limited to data, applications and user privacy. The open interfaces in O-RAN bring democratised applications but also increases the chance for deployed applications to be attacked. Considering the potential fast and frequent developing circle enabled by the RLOps, security practices should be considered throughout the process. This is the emerging paradigm of DevSecOps, in which some of the security responsibility is downloaded to developers. In RLOps, we make several suggestions in addition to the standard DevSecOps.

Since RL is running in an interactive way to provide intelligent decisions to the system, it is essential to consider the feedback from the environment at the beginning, including the feedback on security. For example, a special state can be designed for the MDP to indicate the sudden change of agent behaviour, which could be a sign of attack. The adversarial agent can be introduced in the RL training to test the robustness against malicious agents [47], [48]. Inspired by [71], *Monitoring* could also play an essential role in integrated security measures. To enable security practices through monitoring, attack detection techniques like anomaly detection could be applied.

H. Data Engineering in RLOps

To effectively implement ML/RL on top of O-RAN interfaces, the multiple raw data sources need to be collected, validated, enriched, transformed and stored onto an integrated data pool. That needs to be processed by data engineering processes, such as application of business rules, creation of KPIs, feature engineering, linkage of data tables according to network topology mapping, etc., which ultimately enables the application of the algorithms according to the targeted use cases. Besides, an O-RAN network is built on top of other system components such as IP networks and IT/Cloud infrastructures. The operation and maintenance of these systems are crucial for the overall network performance. It should be integrated into a holistic network management process that addresses all the components. To address the challenge of collecting and preparing data through all stages necessary to the effective application of the ML/RL algorithms, we design and implement the network analytics platform presented in figure 5 that delivers the holistic data pipeline, which is in accordance with the RLOps principles. We explain the compositions of this platform as below.

1) Data Collection Agents: The data collection agents (DCA) are software applications deployed across the network layer, that interact with existing APIs and the network elements (NE). These agents use the standard APIs to collect the standard FCAPS dataset directly from NEs according to the use case. In a RAN, there are network domains that are implemented using equipment and technology that do not offer open and/or standard APIs. For that reason, it is necessary to develop a specific DCA designed to interact with the

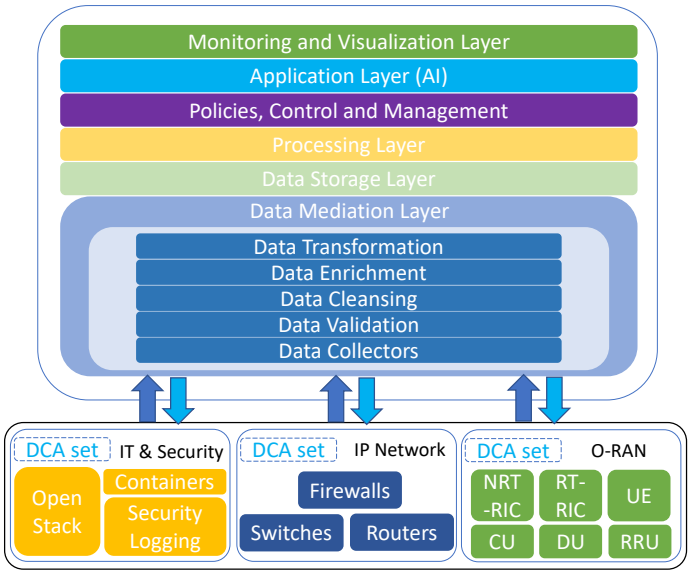


Fig. 5. Network Analytics Platform

specific NE API or protocol, etc. The DCA also has a function of data preparation right from the source, to allow for an efficient and effective data integration coming from multiple and diverse data sources, by normalising the data applying the conventions that have been defined in the system. The DCA is also responsible for the logging of all its actions and performing initial data validation procedures. This function is important to trace end-to-end the data pipeline and assist the upper layer of the data mediation stack. These applications are deployed directly on the management plane of the NE or on any adjacent servers. These have been designed to listen and track the data generated on these sources and are capable to pull the logs and sending them instantly to the data mediation layer (DML).

2) Data Mediation Layer: The DML is responsible for collecting the data by coordinating the DCAs in the southbound interface, data processing and implementing the northbound interface to the upper layers. This layer is a cluster-based system designed according to big data requirements and best practices [72], allowing the system to scale and support ultra-dense networks. After data is collected from the DCAs, the DML receive it in its raw format, requiring it to be prepared before going through validation and cleansing processes. The DML needs to add the schema information to the data stream and link it with the network topology. This preparation process increases the efficiency of the system by reducing the complexity of the data validation, data cleansing.

The DML is responsible for the data validation and data cleansing processes that consist in validating the data against the expected schema, identifying duplicate records, or missing records, and coping with latency on the data source in making the data records available. It also prepares the dataset for an optimal application of the data enrichment processes, that would fail if applied directly to the raw data due to missing network topology information in it. The data enrichment and data transformation functions are tightly coupled with the data

storage and processing layers because it prepares the data stream to match the schemas of the data lake and of other consuming applications. At the end of the DML cycle, the data offered to the upper layers are fully integrated, normalised, enriched and transformed according to the system conventions, thus simplifying the development of the data lake and of any processing applications. The DML layers can be continuously improved and extended to consume more – in quantity and diversity – data sources and to offer the data on the northbound interface in any format, type and frequency that is optimal to the layers consuming the data stream. The DML coordinates with the DCAs to securely collect the data by implementing an encrypted data pipe. It creates one uniform data flow, between each DCA and the upper layers.

3) Data Storage Layer: The data storage layer (DSL) contains one of the main components of the entire architecture which is the data lake. The data lake is the place where the data is stored to be made available to the upper layers, most importantly the processing and application layers. It is designed upon a scalable private cloud object storage, it provides the means to manage and store big-datasets that come in diverse formats and structures and enables high throughput and fast access to the data. The policies, business rules, network topology and other metadata required by the policies, control and management layer are stored in a dedicated relational database that is managed by the DSL. Business Intelligence techniques and the development of ML/RL applications rely heavily upon wide and diverse historical datasets, for trend analysis, statistical analysis and for ML/RL in specific for model training, testing and validation. This demands many computational resources and requires DSL to be designed and implemented using big-data best practices [73], to deliver optimal access to large scale datasets. On the other hand, feature engineering and RL related tasks often require high-speed access to many disparate data sources to build and optimise the ML models, this requires high availability of some of the data in great quantities and diversity. For this, we have designed the data lake following the "Cold, Warm and Hot" approach [74]. The data lake is directly accessible by the other layers such as DM, processing and AI layer through a high throughput network. The design behind this storage system allows us to easily store petabytes of data and serve applications regardless of the data access requirements.

4) Processing Layer: The processing layer (PL) is composed of multiple applications deployed over a containerised environment that scales up with the increased demand from the services of the upper layers such as the application and visualisation layers. The PL handles mainly three types of jobs, distributed real-time computation, distributed batch processing and jobs related to AI models such as environment states, reward calculation, AI model training/testing, etc. AI and ML applications are complex and hard to develop, maintain, optimise, and deploy because of their iterative and multi-staged life-cycle. Complexity arises mostly from the stages that involve feature engineering, model training, model testing/validation and production deployment. On the other hand, the RL has more components to consider which are the environment, reward calculation, and the agents which make deploying these applications more challenging. As emerged in MLOps practices, the main enhancement to solve the challenges of the AI lifecycle is to containerize all stages. The PL has been designed and implemented to follow this principle and overcome this challenge. The PL allows the deployment and execution of services that underpin AI applications throughout its entire life-cycle. In addition, to this, it also implements all the services that involve data processing such as KPI calculation, real-time processing, alarm processing, online monitoring notifications, rule enforcement and data preparation for visualisation. This layer works in tandem with the lower layers such as DSL and DML, to provide a containerised environment that simplifies the deployment and management of resource-intensive applications and guarantees high-throughput access to the data pool through dedicated and purpose-built data streams. This layer will help to encapsulate the works in subphases where the task could be updated separately without affecting other phases, we illustrate some of the main jobs in this layer as follow:

- KPI calculation: To measure the performance of the whole network, 3gpp produced a technical specification document for KPIs [72]. These KPIs need an elevated level of domain expertise to develop and deploy across the data pool. The purpose of these KPIs includes but is not limited to, the monitoring and troubleshooting of the network performance and long-term trend analysis of its performance. However, they are valuable features to build ML/RL models and to reflect environment status. By abstracting this layer, we intend to save time and reduce complexity. The KPIs are calculated periodically. The results are eventually stored with the collected performance metrics for usage by the AI engineers.
- Feature engineering and real-time data processing: Considering the requirements of the RLOps, the processing layer will also run applications that process streams and batches of data, so this layer is where the feature engineering process is done.
- ML/RL related components: The components needed to train, test and validate the ML/RL application. These containers and the related applications are integrated into the whole platform so that they are able to cooperate with other containers and services offered in the processing layer. Additionally, the processing layer is able to run environment simulators or DT images and integrate them into the data pipeline.

5) Policies and Control Layer: The policies and control Layer is composed of a set of configuration methods, services and metadata that define and implement the business rules, object hierarchy and relationship across that are relevant for the functionality implemented across the data mediation, data storage and processing layers. The O-RAN FCAPS data, produced across the multiple virtual network functions (VNFs) and interfaces, is the most representative and important data type in this platform. This data is being structured and is not generated in its raw format, with the whole information that is required its representation and to be integrated with other data sources. This layer contains the rules, metadata

and methodologies necessary for the efficient and effective implementation of the cycles of the DML and DSL, allowing the creation of the structures to validate, cleanse, enrich and store the data in an optimal format. The network topology metadata and methods are fundamental for the linkage of the different managed objects and data structures, thus enabling the cross-layer analysis between network performance events and external events described by data sources - that are external to the O-RAN network and relevant for the analytical process, e.g. UE-based data that describes QoS and QoE events through detailed metrics and logs. On the other hand, this layer also stores the policies and rules that control some aspects of the system's cognitive capabilities, such as identification of abnormal behaviour and respecting self-healing action/decision. These policies and rules can be defined by: subject matter experts (SMEs) through processes of data engineering, feature engineering and/or analytical engineering; and by automated analytic processes, possibly based in ML/RL applications that identify rules/decisions that after being validated and accepted by SMEs are later deployed on to production.

6) AI Layer (AI application management Layer): The AI layer is where the development, initial training and validation of the AI model happens. It allows implementing online training through real-time data consumption and offline model validation generating results/decisions that are not implemented rather validated by the developers and the subject-matter-experts. It also allows to monitor logs and track the performance of the AI jobs and related application images mostly for testing and debugging purposes.

7) Data Visualization Layer: This layer is mostly dedicated to implementing business intelligence functions that allow for SMEs to access the data in the format of graphical reports and dashboards thus providing a visual interface to monitor the overall system performance. Through this layer, it is possible to access reports and dashboards that inform about the performance of the different system components through the monitoring of dedicated measurements. The components that are monitored are:

- O-RAN network equipment, VNFs, protocols, interfaces, and functions: this allows for the network management SMEs to evaluate network performance, identify opportunities of optimisation, trends of systemic behaviour and evaluate the impact that AI algorithms might have on the overall system performance.
- AI application decision-making logging: this allows for the DevOps, MLOps and RLOps engineers to evaluate the performance of these applications during the entire life-cycle from training to operations. It also allows to visually report the results of correlation and causation analysis, emphasised on the evaluation of the decision of the application on the system performance.

IV. Best Practices of RLOps

In this section, we discuss some best practices and effective routines for successfully delivering RL applications as the reflections of general principles presented in Section III. We will elaborate on DT's functionalities and critical features, then discuss the automation and reproducibility engineering in RLOps, respectively.

A. Digital Twins

A wide range of working definitions of DT exists [35], where we consider the definition by [75] which is that "A digital twin is a digital representation of a physical item or assembly using integrated simulations and service data". A DT of a 5G network would provide high-fidelity representations of all components of the current live network. This includes the RAN, core network, and characteristics about users and service behaviours among others. Where each component will be modelled through the use of ML models or emulated elements for example [36]. As discussed in [36], DTs offer a wide range of benefits for communications networks, including reducing the deployment costs for new services and supporting network automation and optimization.

Within the context of RLOps, DTs are likely to be an integral part of the development pipeline. Enabling training, testing and validation of DRL agents in an environment that provides a good approximation to the real world without the associated risks. The key benefits it provides from an RL perspective are enumerated in the list below.

1) **Exploration:** RL algorithms require exploration in order to learn about the environment in which they are operating. Exploration, by definition, is risky, as it requires the execution of actions that have potentially unknown outcomes and could, in principle, be unrecoverable [66]. A DT provides a high-fidelity approximation to the real network where a failure is an option, as any damage is inconsequential as it is reversible.
2) **Parallelization:** Sample efficiency is a crucial problem within DRL, where agents typically take considerable time to train. The utilization of several environments in parallel can reduce the real clock time which an agent takes to converge [76], [77]. Deployment on the real network does not support this functionality.
3) **Validation:** When any new component (be that physical hardware, an RL agent or some new software function), there is potential for unforeseen negative behaviours to occur. Mitigating these deployment risks is essential from a business perspective. A DT easily accommodates this desired functionality as it allows for simulation and investigation of network response in a wide variety of scenarios. From an RL specific perspective, it allows for confirmation of the agent's capacity for reward acquisition and provides functionality to support the interpretability of RL policies more readily.

In addition to the number of compelling arguments for their utilization, certain risks must be realized. Within the remainder of this section, we introduce a well-known challenge considered by the DRL robotics community which is commonly referred to as *Sim-to-Real* [37]. The associated literature is concerned with training within simulation and deployment within the real world and attempts to mitigate risks associated with approximation error between the two systems. Fundamentally, this same desire and challenge will

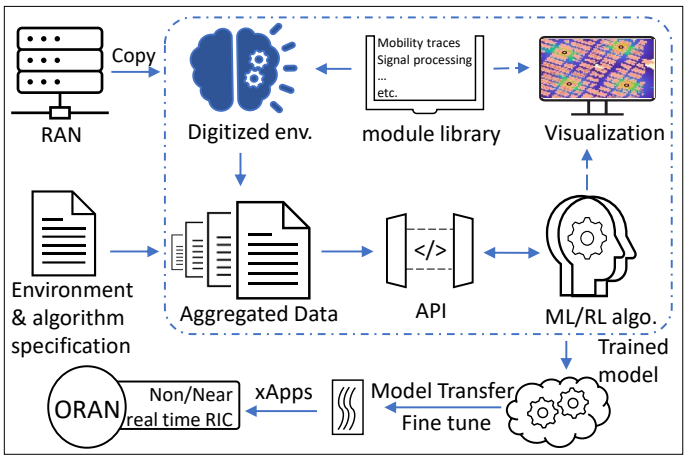


Fig. 6. Digital twins functionalities. The core, which is a digital copy of the RAN, interacts with additional modules to simulate the change of network environments.

persist within our pipeline and more widely within telecommunications applications. For a comprehensive survey of the area please refer to [37].

B. Automation

The realization of an automated development process is undoubtedly the critical factor in any type of DevOps. The training procedure needs to be automated in order to save time and labour, expediting the transition from development to production.

1) Data: Data cleaning and preparation is often necessary for any new task or environment. This facilitates pattern detection for models as features are well scaled and ordered. As data generated for a task is often consistent, once the transformation procedure is done once, it can be repeated every other time without any need for manual interference. Following data preparation, appropriate feature/state representations must be created to be provided to the agent. This can include concatenating data frames from multiple time-steps together, skipping every n frame, obtaining a certain embedding of the transformed data, etc. This process is often specific to the algorithm/task at hand and is done during training. Since it is a highly repeated step and requires no manual input past creation, it can be automated. The data transformation and feature extraction process can be finished in the DML and PL layer of our network analytics platform respectively.

Reward functions can be either extrinsically created for a problem or intrinsically generated from available data. The former case warrants no further automation; however, intrinsic reward signals are obtained from engineering pipelines that extract the signal out of the transformed data. This process will most likely be repeated on every training/evaluation step and must be automated. The data visualization layer provides such information but needs to establish the automated reward engineering mechanism according to the specific cases.

2) Model: Given a certain environment or task, the data preparation pipeline, i.e., the network analytics platform can be triggered and completed automatically. This reduces the amount of time spent on data preparation and guarantees consistency as development evolves. Each DRL model follows a specific training methodology. Following data preparation, the training process can also be automated. Training can terminate or resume given performance metrics attached to the agent. At last, for hyperparameter/parameter selection, a common process for DL can also be automated. A hyperparameter sweep can commence once the training pipeline is formulated. The best set of parameters can be chosen based on performance metrics.

3) Code: An evaluation/testing step can be automatically triggered once a model has completed training. If passed, the agent can then be deployed to production. This process requires rigorous testing scripts to be created, bypassing the agent's manual testing/evaluation, and thereby automating the transition from development to production. A model is usually a small sub-part of a larger application infrastructure providing a specific service. Once a new agent is ready for production deployment, it is necessary to automate the new application build process to ensure each new version is well documented and tracked.

C. Reproducibility

In O-RAN, well-trained RL models may need to be widely deployed in a large geographic area. Therefore, in the face of different deployment environments and carriers, it is very important to ensure that the performance of the model does not deteriorate, that is the reproducibility.

1) Data: The development cycle is often about the model, but in many cases can be about changing the environment or handling new data. Change in data can break a model's performance, and retraining is usually necessary. Dealing with changes in data without loss in performance is of paramount importance in DRL. An agent that can generalize is a flexible and robust one. To tackle the issue of generalization, research challenges have appeared in recent years such as the Procgen challenge [78] in which agents are tested on multiple versions of the same environment. Keeping track of older versions of data/environments is vital for maintaining stable versions, debugging drops in performance, and developing more robust models.

2) Model: Model performance can change drastically with minor changes in the training algorithm. Reproducing results in DRL is very difficult given its dynamic nature [79]. In DRL both the data source and the agent dynamically change. Moreover, they each influence one another. The environment affects how the agent trains and the agent's policy impacts the environment's evolution. The ability to revert to stable versions of a model is vital for maintaining stability in the event of performance degradation. In terms of development, minor changes to the model can be researched on their own prior to compounding improvements. Maintaining a careful log of which models possess which mutations are important for ease of integration. Each model version should also contain its own pseudo-code, clearly elaborating the differences in the algorithm. Furthermore, the method of feature creation must be consistent and well logged as it affects how models

interpret the provided data. Such strategies massively aid with the development and debugging of new models.

3) Code: There will be specific dependencies upon which the model relies. Maintaining correct versioning between development and production is necessary for the replication of behaviour. The same goes for the software stack used to create the product in development. It makes no sense to rely on a different, untested stack in production. Therefore, it is often best to containerize development and production iterations. This means all versioning data is well documented within their own containers, allowing for ease of reproducibility.

V. Conclusions

The ML principles and O-RAN are a natural match and have a great potential of being the backbone of truly intelligent future network infrastructure. However, O-RAN development and maintenance pipelines require careful formulation and planning. The challenge becomes particularly acute when data derived optimal decision-making strategies are considered, i.e. Reinforcement Learning controlled O-RAN. This paper provides a comprehensive overview and taxonomy of MLOps principles which are bespoke for RL. In RLOps, we take the life-cycle of RL model development as the main consideration, adopting the design, development, operations, safety/security and data engineering as principles. We detail all main considerations and methodologies under these principles and we integrate the above functions with a network analytics platform and Digital Twins, which is geared to achieving automatic and reproducible model operations.

Acknowledgment

This work was developed within the Innovate UK/CELTIC-NEXT European collaborative project on AIMM (AI-enabled Massive MIMO). This work has also been funded in part by the Next-Generation Converged Digital Infrastructure (NG-CDI) Project, supported by BT and Engineering and Physical Sciences Research Council (EPSRC), Grant ref. EP/R004935/1.

References

[1] L. Bonati, M. Polese, S. D'Oro, S. Basagni, and T. Melodia, "Open, programmable, and virtualized 5G networks: State-of-the-art and the road ahead," *Computer Networks*, vol. 182, p. 107516, 2020.

[2] O. Alliance, "O-ran use cases and deployment scenarios," *White Paper, Feb*, 2020.

[3] M. Soltani, V. Pourahmadi, A. Mirzaei, and H. Sheikhzadeh, "Deep learning-based channel estimation," *IEEE Communications Letters*, vol. 23, no. 4, pp. 652–655, 2019.

[4] A. Forster, "Machine learning techniques applied to wireless ad-hoc networks: Guide and survey," in *2007 3rd international conference on intelligent sensors, sensor networks and information*. IEEE, 2007, pp. 365–370.

[5] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4005–4018, 2019.

[6] C. Renggli, L. Rimanic, N. M. Gürel, B. Karlaš, W. Wu, and C. Zhang, "A data quality-driven view of MLOps," *arXiv preprint arXiv:2102.07750*, 2021.

[7] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano, "Devops," *Ieee Software*, vol. 33, no. 3, pp. 94–100, 2016.

[8] V. Dunjko and H. J. Briegel, "Machine learning & artificial intelligence in the quantum domain: A review of recent progress," *Reports on Progress in Physics*, vol. 81, no. 7, p. 074001, 2018.

[9] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep clustering for unsupervised learning of visual features," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 132–149.

[10] B. W. Silverman, *Density estimation for statistics and data analysis*. Routledge, 2018.

[11] C. Bishop, *Pattern Recognition and Machine Learning*. New York: Springer, 2006.

[12] K. Zhang, Z. Yang, and T. Başar, "Multi-agent reinforcement learning: A selective overview of theories and algorithms," 2021.

[13] Faisal, "Ran vs cloud ran vs vran vs o-ran: A simple guide!" Apr 2021. [Online]. Available: https://telcocloudbridge.com/blog/c-ran-vs-cloud-ran-vs-vran-vs-o-ran/

[14] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *Ieee Access*, vol. 6, pp. 14 410–14 430, 2018.

[15] D. W. Otter, J. R. Medina, and J. K. Kalita, "A survey of the usages of deep learning for natural language processing," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 2, pp. 604–624, 2020.

[16] Y.-H. Wu, Z.-C. Yu, C.-Y. Li, M.-J. He, B. Hua, and Z.-M. Chen, "Reinforcement learning in dual-arm trajectory planning for a free-floating space robot," *Aerospace Science and Technology*, vol. 98, p. 105657, 2020.

[17] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[18] H. Yang, "Aligraph: A comprehensive graph neural network platform," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 3165–3166.

[19] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko *et al.*, "Highly accurate protein structure prediction with AlphaFold," *Nature*, vol. 596, no. 7873, pp. 583–589, 2021.

[20] S.-Y. Lien, K.-C. Chen, and Y. Lin, "Toward ubiquitous massive accesses in 3GPP machine-to-machine communications," *IEEE Communications Magazine*, vol. 49, no. 4, pp. 66–74, 2011.

[21] Y. Ouyang, L. Wang, A. Yang, M. Shah, D. Belanger, T. Gao, L. Wei, and Y. Zhang, "The next decade of telecommunications artificial intelligence," *arXiv preprint arXiv:2101.09163*, 2021.

[22] T. Van Luong, Y. Ko, N. A. Vien, D. H. Nguyen, and M. Matthaiou, "Deep learning-based detector for OFDM-IM," *IEEE wireless communications letters*, vol. 8, no. 4, pp. 1159–1162, 2019.

[23] A. Felix, S. Cammerer, S. Dörner, J. Hoydis, and S. Ten Brink, "OFDM-Autoencoder for End-to-End learning of communications systems," in *2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2018, pp. 1–5.

[24] Y. Yang, F. Gao, G. Y. Li, and M. Jian, "Deep learning-based downlink channel prediction for FDD massive MIMO system," *IEEE Communications Letters*, vol. 23, no. 11, pp. 1994–1998, 2019.

[25] Q. Wang, K. Feng, X. Li, and S. Jin, "Precodernet: Hybrid beamforming for millimeter wave systems with deep reinforcement learning," *IEEE Wireless Communications Letters*, vol. 9, no. 10, pp. 1677–1681, 2020.

[26] Y. Wang, Z. Zhang, S. Zhang, S. Cao, and S. Xu, "A unified deep learning based polar-LDPC decoder for 5G communication systems," in *2018 10th International Conference on Wireless Communications and Signal Processing (WCSP)*. IEEE, 2018, pp. 1–6.

[27] L. Bonati, S. D'Oro, M. Polese, S. Basagni, and T. Melodia, "Intelligence and learning in O-RAN for data-driven NextG cellular networks," *arXiv preprint arXiv:2012.01263*, 2020.

[28] T. Pamuklu, M. Erol-Kantarci, and C. Ersoy, "Reinforcement learning based dynamic function splitting in disaggregated green Open RANs," in *ICC 2021-IEEE International Conference on Communications*. IEEE, 2021, pp. 1–6.

[29] X. Wang, J. D. Thomas, R. J. Piechocki, S. Kapoor, R. Santos-Rodriguez, and A. Parekh, "Self-play learning strategies for resource assignment in Open-RAN networks," *arXiv preprint arXiv:2103.02649*, 2021.

[30] S. Padakandla, "A survey of reinforcement learning algorithms for dynamically varying environments," *ACM Computing Surveys (CSUR)*, vol. 54, no. 6, pp. 1–25, 2021.

[31] S. Jordan, Y. Chandak, D. Cohen, M. Zhang, and P. Thomas, "Evaluating the performance of reinforcement learning algorithms," in *International Conference on Machine Learning*. PMLR, 2020, pp. 4962–4973.

[32] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[33] T. M. Moerland, J. Broekens, and C. M. Jonker, "Model-based reinforcement learning: A survey," *arXiv preprint arXiv:2006.16712*, 2020.

[34] E. Puiutta and E. M. Veith, "Explainable reinforcement learning: A survey," in *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*. Springer, 2020, pp. 77–95.

[35] A. Fuller, Z. Fan, C. Day, and C. Barlow, "Digital twin: Enabling technologies, challenges and open research," *IEEE access*, vol. 8, pp. 108 952–108 971, 2020.

[36] H. X. Nguyen, R. Trestian, D. To, and M. Tatipamula, "Digital twin for 5G and beyond," *IEEE Communications Magazine*, vol. 59, no. 2, pp. 10–15, 2021.

[37] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: A survey," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2020, pp. 737–744.

[38] A. Kadian, J. Truong, A. Gokaslan, A. Clegg, E. Wijmans, S. Lee, M. Savva, S. Chernova, and D. Batra, "Sim2Real Predictivity: Does evaluation in simulation predict real-world performance?" *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6670–6677, 2020.

[39] J. T. Springenberg, A. Klein, S. Falkner, and F. Hutter, "Bayesian optimization with robust bayesian neural networks," *Advances in neural information processing systems*, vol. 29, pp. 4134–4142, 2016.

[40] Y. Li, "Deep reinforcement learning: An overview," *arXiv preprint arXiv:1701.07274*, 2017.

[41] J. Gauci, E. Conti, Y. Liang, K. Virochsiri, Y. He, Z. Kaden, V. Narayanan, X. Ye, Z. Chen, and S. Fujimoto, "Horizon: Facebook's open source applied reinforcement learning platform," *arXiv preprint arXiv:1811.00260*, 2018.

[42] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *CoRR*, vol. abs/2004.05785, 2020. [Online]. Available: https://arxiv.org/abs/2004.05785

[43] V. Medel, O. Rana, J. Á. Bañares, and U. Arronategui, "Modelling performance & resource management in kubernetes," in *Proceedings of the 9th International Conference on Utility and Cloud Computing*, 2016, pp. 257–262.

[44] A. Alwarafy, M. Abdallah, B. S. Ciftler, A. Al-Fuqaha, and M. Hamdi, "Deep reinforcement learning for radio resource allocation and management in next generation heterogeneous wireless networks: A survey," *arXiv preprint arXiv:2106.00574*, 2021.

[45] E. Altman, *Constrained Markov decision processes*. CRC Press, 1999, vol. 7.

[46] H. Myrbakken and R. Colomo-Palacios, "Devsecops: A multivocal literature review," in *International Conference on Software Process Improvement and Capability Determination*. Springer, 2017, pp. 17–29.

[47] T. Chen, J. Liu, Y. Xiang, W. Niu, E. Tong, and Z. Han, "Adversarial attack and defense in reinforcement learning-from AI security view," *Cybersecurity*, vol. 2, no. 1, pp. 1–22, 2019.

[48] R. Elderman, L. J. Pater, A. S. Thie, M. M. Drugan, and M. A. Wiering, "Adversarial reinforcement learning in a cyber security simulation." in *ICAART (2)*, 2017, pp. 559–566.

[49] A. Pattanaik, Z. Tang, S. Liu, G. Bommannan, and G. Chowdhary, "Robust deep reinforcement learning with adversarial attacks," *arXiv preprint arXiv:1712.03632*, 2017.

[50] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[51] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, and M. Young, "Machine learning: The high interest credit card of technical debt," in *SE4ML: Software Engineering for Machine Learning (NIPS 2014 Workshop)*, 2014.

[52] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," *arXiv preprint arXiv:2005.01643*, 2020.

[53] E. Breck, S. Cai, E. Nielsen, M. Salib, and D. Sculley, "The ML test score: A rubric for ML production readiness and technical debt reduction," in *2017 IEEE International Conference on Big Data (Big Data)*, 2017, pp. 1123–1132.

[54] D. Guo, L. Tang, X. Zhang, and Y.-C. Liang, "Joint optimization of handover control and power allocation based on multi-agent deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 11, pp. 13 124–13 138, 2020.

[55] L. S. Shapley, "Stochastic games," *Proceedings of the national academy of sciences*, vol. 39, no. 10, pp. 1095–1100, 1953.

[56] C. Florensa, Y. Duan, and P. Abbeel, "Stochastic neural networks for hierarchical reinforcement learning," *arXiv preprint arXiv:1704.03012*, 2017.

[57] N. Rao, E. Aljalbout, A. Sauer, and S. Haddadin, "How to make deep RL work in practice," 2020.

[58] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," 2013.

[59] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[60] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[61] S. Narvekar and P. Stone, "Learning curriculum policies for reinforcement learning," *arXiv preprint arXiv:1812.00285*, 2018.

[62] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, pp. 1–35, 2017.

[63] M. Ghavamzadeh, S. Mannor, J. Pineau, and A. Tamar, "Bayesian reinforcement learning: A survey," *arXiv preprint arXiv:1609.04436*, 2016.

[64] "MLOps: Continuous delivery and automation pipelines in machine learning." [Online]. Available: https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning

[65] G. Papoudakis, F. Christianos, A. Rahman, and S. V. Albrecht, "Dealing with non-stationarity in multi-agent deep reinforcement learning," *CoRR*, vol. abs/1906.04737, 2019. [Online]. Available: http://arxiv.org/abs/1906.04737

[66] D. Amodei, C. Olah, J. Steinhardt, P. F. Christiano, J. Schulman, and D. Mané, "Concrete problems in AI safety," *CoRR*, vol. abs/1606.06565, 2016. [Online]. Available: http://arxiv.org/abs/1606.06565

[67] E. Puiutta and E. M. S. P. Veith, "Explainable Reinforcement Learning: A Survey," in *Machine Learning and Knowledge Extraction*, A. Holzinger, P. Kieseberg, A. M. Tjoa, and E. Weippl, Eds. Cham: Springer International Publishing, 2020, pp. 77–95.

[68] G. Dulac-Arnold, D. Mankowitz, and T. Hester, "Challenges of real-world reinforcement learning," *arXiv preprint arXiv:1904.12901*, 2019.

[69] A. HasanzadeZonuzy, D. M. Kalathil, and S. Shakkottai, "Learning with safety constraints: Sample complexity of reinforcement learning for constrained mdps," *arXiv preprint arXiv:2008.00311*, 2020.

[70] R. H. Russel, M. Benosman, and J. Van Baar, "Robust Constrained-MDPs: Soft-constrained robust policy optimization under model uncertainty," *arXiv preprint arXiv:2010.04870*, 2020.

[71] J. Díaz, J. E. Pérez, M. A. Lopez-Peña, G. A. Mena, and A. Yagüe, "Self-service cybersecurity monitoring as enabler for devsecops," *IEEE Access*, vol. 7, pp. 100 283–100 295, 2019.

[72] Y. Roh, G. Heo, and S. E. Whang, "A survey on data collection for machine learning: A big data - AI integration perspective," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 4, pp. 1328–1347, 2021.

[73] W. Chang and N. Grady, "Nist big data interoperability framework: Volume 1, definitions," 2019-10-21 2019.

[74] T.-H. Dang-Ha, D. Roverso, and R. Olsson, "Graph of virtual actors (GOVA): A big data analytics architecture for IOT," in *2017 IEEE International Conference on Big Data and Smart Computing (BigComp)*. IEEE, 2017, pp. 162–169.

[75] J. A. Erkoyuncu, P. Butala, R. Roy *et al.*, "Digital twins: Understanding the added value of integrated models for through-life engineering services," *Procedia Manufacturing*, vol. 16, pp. 139–146, 2018.

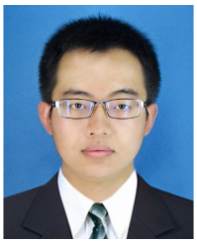
[76] D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. van Hasselt, and D. Silver, "Distributed prioritized experience replay," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=H1Dy---0Z

[77] 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.

[78] S. Mohanty, J. Poonganam, A. Gaidon, A. Kolobov, B. Wulfe, D. Chakraborty, G. Šemetulskis, J. Schapke, J. Kubilius, J. Pašukonis *et al.*, "Measuring sample efficiency and generalization in reinforcement learning benchmarks: Neurips 2020 procgen benchmark," *arXiv preprint arXiv:2103.15332*, 2021.

[79] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

VI. BIOGRAPHIES



Peizheng Li received the B.Eng. degree in optoelectronic information engineering from Nanjing University of Posts and Telecommunications, China, in 2015, and the M.Sc. degree with distinction in image and video communications and signal processing from the University of Bristol, U.K., in 2019, where he continues pursuing the Ph.D. degree with the communication systems and networks group. His research interests include machine learning, radio localisation and radio access network.

Jonathan Thomas received his M.Eng. Degree in Electronics and Communications Engineering from Cardiff University in 2016. He is now pursuing a Ph.D. degree within the communications systems and networks group at the University of Bristol. His research focuses on Multi-Agent Reinforcement Learning communications networks.

Xiaoyang Wang received her B.E. degree in electronic science and technology and her Ph.D. degree in signal and information processing from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2013 and 2018, respectively. She visited the University of Bristol, Bristol, UK, from 2017-2018. She has joined the Department of Electrical and Electronic Engineering, University of Bristol, as a Research Associate since 2018. She has worked on a variety of computer vision topics such as target detection, remote sensing image processing and visual tracking. Her current research focuses on machine learning, especially reinforcement learning, and its applications in next-generation network management.

Ahmed Khalil received a B.S. in Mechanical Engineering from Purdue University, U.S., in 2017. He then obtained an M.Sc. in Robotics with distinction from the University of Bristol and the University of the West of England, U.K., in 2019. He has since continued at the University of Bristol pursuing a Ph.D. in Electrical and Electronics Engineering with a focus on Deep Reinforcement Learning. His current research area revolves around bolstering Deep Reinforcement Learning methods with Self-Supervised Learning techniques.

Abdelrahim Ahmad is the Data Scientist Lead at Vilicom, where his most important work is related with the development of Data and Artificial Intelligence related applications focused on the Telecommunication industry. He has relevant and rich experience in AI, Data Architecture, Big Data ecosystems, and 5G and O-RAN technologies. Holder of two master's degrees in Big Data Systems & Data Science and Web Sciences. Most remarkable experience have been achieved whilst leading data science projects, requiring the design and implementation of end-to-end big data analytical platforms to serve telecom businesses, he is mostly interested in building (near-RT, RT) platforms that combine O-RAN RAN Intelligent Controller with Reinforcement Learning applications.

Rui Inacio is currently Vilicom's Chief Technology Officer being responsible for the digitisation of the company's business models mainly through the development of digital platforms that combine O-RAN technology, Big Data technologies and AI technologies. He has built a diversified career in the Mobile Communications industry, with a rich experience across multiple areas of expertise such as network management, network operations, network deployment and network support systems. He has both experience as an engineer and as a technologist that thinks of and about the network life-cycle and also thinks how to develop technology to assist engineers in this life-cycle. He hold a master degree in Electronics and Telecommunications Engineering and a master degree in Business and Technology Management. Currently he is mostly interested in contributing for the development of cognitive networks that are highly automate and autonomic across the whole network management life-cycle.

Shipra Kapoor received her PhD in Electronic Engineering from University of York, UK in 2019, with the subject of her thesis being 'Learning and Reasoning Strategies for User Association in Ultra-dense Small Cell Vehicular Networks'. Since 2020 she has been a member of Self-Learning Networks team in Applied Research at BT, where she is now a Research Specialist. Her current research interests include application of artificial intelligence and machine learning techniques in next generation wireless networks to resource and topology management, connected and autonomous vehicle networks, O-RAN architecture and cognitive radio network.

Arjun Parekh leads a BT research team exploring the uses of Machine Learning & AI for the automation of optimisation & planning in converged networks. He led the development of processes & tools to drive the rollout of the UK's first 4G & 5G networks, and has over 20 years experience in Radio Network Management, Optimisation and Automation. Arjun has an MPhys in Physics from the University of Oxford and is a Visiting Industrial Fellow at the University of Bristol.

Angela Doufexi received the B.Sc. degree in physics from the University of Athens, Greece, in 1996; the M.Sc. degree in electronic engineering from Cardiff University, Cardiff, U.K., in 1998; and the Ph.D. degree from the University of Bristol, U.K., in 2002. She is currently a Professor of Wireless networks at the University of Bristol. Her research interests include vehicular communications; new waveforms, resource allocation, massive MIMO and multiple antenna systems, mmWave communications and sixth-generation communications systems. She is the author of over 200 journal and conference papers in these areas.

Dr Arman Shojaeifard has +9 years of post-PhD experience in research, development, and standardization of radio transmission technologies and architectures. He is currently an R&I Senior Manager at InterDigital Europe where he serves as the Chair of the ETSI Industry Specification Group on Reconfigurable Intelligent Surfaces (ISG RIS) and the Technical Manager of the Innovate-UK/CELTIC-NEXT European collaborative R&D project on AIMM (AI-enabled Massive MIMO). Prior to InterDigital, he was a Senior Researcher and 3GPP RAN1 WG Delegate at British Telecommunications plc. He received his PhD degree in Wireless Communications from King's College London in 2013.

Robert J. Piechocki is a Professor of Wireless Systems at the University of Bristol. His research expertise is in the areas of Connected Intelligent Systems, Wireless & Self-Learning Networks, Information and Communication Theory, Statistics and AI. Rob has published over 200 papers in peer-reviewed international journals and conferences and holds 13 patents in these areas. He leads wireless connectivity and sensing research activities for the IRC SPHERE project (winner of 2016 World Technology Award). He is a PI/CI for several high-profile projects in networks and AI funded by the industry, EU, Innovate UK and EPSRC such as NG-CDI, AIMM, OPERA, FLOURISH, SYNERGIA. He regularly advises the industry and the Government on many aspects related to connected intelligent technologies and data sciences.