Journal on Wireless Communications and Networking

RESEARCH Open Access

A development and validation framework for AI/ML-driven rApps in open RAN: a case study on network energy saving

Minhyun Kim[1*], Kyoung Seok Lee[1], Soojung Jung[1], Jung Mo Moon[1], Jee-Hyeon Na[1], Salvatore D'Oro[2], Leonardo Bonati[2] and Tommaso Melodia[2]

*Correspondence:
minhyun.kim@etri.re.kr

[1] Intelligent RAN SW Research Section, Electronics and Telecommunications Research Institute (ETRI), 218 Gajeong-ro, Daejeon 34129, Republic of Korea
[2] Institute for Intelligent Networked Systems, Northeastern University, 360 Huntington Avenue, Boston, MA 02115, USA

Abstract

Open radio access network (RAN) leverages the RAN intelligent controller (RIC) to enable artificial intelligence/machine learning (AI/ML)-driven network automation. However, a gap remains between algorithmic research and deployable, standards-compliant rApp prototypes with verifiable behavior. This paper addresses this gap by introducing an integrated development and validation framework that supports the full lifecycle of AI/ML-based rApps, from prototyping to functional verification. The framework includes a standards-compliant non-real-time RIC (Non-RT RIC) architecture with supporting functions, an interface for integrating RAN simulators, and a visualization dashboard that displays system state and control actions, enabling traceability of end-to-end control loops. We demonstrate the framework through a case study involving the design and implementation of a predictive network energy saving rApp. In closed-loop experiments, instrumented logs and visualizations indicate that the control decisions of the rApp adhere to the intended operational logic, allowing repeatable functional validation. We also discuss challenges for real-world deployment and study limitations. Overall, the proposed framework provides a practical methodology and toolset that accelerate the transition from algorithmic concept to deployable, validated rApps, advancing reliable AI/ML solutions within the O-RAN ecosystem and offering direct applicability to energy saving as well as other O-RAN use cases.

Keywords: Open RAN, Non-RT RIC, rApp, AI/ML, Network energy saving

1 Introduction

The fifth and sixth generations (5G/6G) of mobile communication systems aim to meet unprecedented demands for high data rates, low latency, and massive connectivity, with Open RAN serving as a key enabler. Open RAN represents a paradigm shift from traditional, closed, monolithic RAN architectures toward disaggregated, virtualized, and intelligence-enabled networks, built upon five guiding principles: openness (open interfaces), disaggregation (functional split), flexibility (cloud-native virtualization), interoperability (standards-based integration), and intelligence (automation) [1, 2]. These

© The Author(s) 2026. **Open Access** This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by-nc-nd/4.0/.

principles are expected to foster a heterogeneous multi-vendor ecosystem, reduce operational expenditures, and accelerate innovation.

Among these five principles, intelligence is realized and orchestrated by the RAN intelligent controller (RIC), which serves as the brain of the Open RAN architecture. The RIC is logically partitioned into the non-real-time RIC (Non-RT RIC) and the near-real-time RIC (Near-RT RIC). Operating on time scales of one second and above, the Non-RT RIC provides high-level policy control, data analytics, and artificial intelligence/machine learning (AI/ML) model training, and it hosts specialized applications (rApps) [3]. Using Non-RT services, rApps introduce AI/ML-driven intelligence and automation into network operations and optimize performance across use cases such as network energy saving, traffic steering, and quality of experience (QoE) enhancement [4]. Collectively, the RIC shifts network management from manual and reactive operation toward automated, proactive, and data-driven control with closed-loop operation.

Despite significant progress in AI/ML for RAN optimization, a practical gap persists between state-of-the-art (SOTA) algorithmic results and their realization as deployable, validated rApps within the standardized O-RAN architecture. Numerous SOTA methods, e.g., dynamic base station activation/deactivation in pursuit of energy savings, report strong performance in simulation [5–7]. However, transitioning such results into production-grade rApps continues to face a development-to-operation gap. Simulation and emulation are essential tools for safe and repeatable experimentation. The limitation we highlight is algorithm-only evaluation without standards-aligned integration details and end-to-end closed-loop functional validation across the Non-RT RIC pipeline and its interfaces. This motivates our emulator-based case study, where we explicitly validate end-to-end behaviors through standards-aligned R1/O1 signaling and Non-RT RIC integration. From an rApp developer's perspective, the main challenges are:

- **Integration complexity**: Migrating Python-based models into rApps requires implementing O-RAN-specified interfaces, engineering robust data pipelines, ensuring compatibility with the Non-RT RIC framework, and automating containerization and deployment.
- **Scarcity of realistic environments and data**: Large-scale, realistic datasets are limited; rigorous closed-loop evaluation where rApp decisions modify network state and the feedback re-enters the pipeline demands sophisticated simulation/emulation environments.
- **Operational constraints**: Deployability hinges on satisfying practical metrics, including robustness to imperfect data, control loop latency, inference latency and model footprint, and control-plane overhead.
- **Lack of an end-to-end workflow**: Standardized lifecycle from data generation and model training through rApp integration, policy dissemination via A1 interface, fair benchmarking, and robustness testing remains incomplete; fragmented toolchains impede reproducibility and slow iteration.

Absent a systematic methodology to integrate, test, and validate AI/ML models within real non-RT RIC and rApp development environments, even strong algorithms risk remaining academic artifacts. To bridge this gap, we propose a comprehensive development and

validation framework for AI/ML-driven rApps. Our goal is not to introduce a new SOTA algorithm; rather, we provide the scaffolding required to operationalize these algorithms into functionally verifiable rApps in the non-RT RIC environment. We also do not propose a new RAN architecture. Instead, we focus on a tool-supported workflow for standards-consistent R1 and O1 integration, end-to-end closed-loop functional validation, and traceability. The dashboard and visualization functions are presented as supporting components that improve observability and traceability during closed-loop validation, rather than as primary contributions of this paper. Compared to our preliminary work [8], this paper extends the framework in three aspects. First, we specify the R1 and O1 interaction steps used in the closed-loop workflow and the corresponding framework modules. Second, we detail the measurement and configuration mapping steps between the RAN emulation endpoint and the non-RT RIC framework, including explicit checks performed before generating configuration operations. Third, we expand the case study description and analysis to clarify the evaluation scope and limitations. The main contributions are summarized as follows:

- **Integrated development workflow**: We design and implement an end-to-end workflow that covers the entire rApp lifecycle. It comprises a 5G system-level simulator for realistic dataset generation, streamlined procedures for AI/ML model training and rApp container integration, and a non-RT RIC environment for closed-loop performance validation.
- **Openness and extensibility**: The framework is engineered for interoperability. We detail integration with an industry-recognized third-party validation platform, enabling rigorous testing against standardized network scenarios and demonstrating flexible interfacing with external tools.
- **Empirical validation via a case study**: We conduct a comprehensive network energy saving case study to demonstrate practical utility. Using the framework, we implement, deploy, and validate AI/ML models under identical conditions, enabling fair comparison, robustness assessment, and performance analysis based on operational metrics, thereby accelerating rApp development and validation.

The remainder of this paper is organized as follows: Section 2 presents an overview of the Open RAN architecture and the challenges in rApp implementation addressed in this work. Section 3 details the architecture, components, and end-to-end workflow of the proposed framework, while Sect. 4 describes the case study, covering the network energy saving problem formulation and the implemented rApp including the AI/ML model. Section 5 provides a functional validation of the framework by presenting visual evidence of the successful operation of the rApp. Section 6 discusses the practical implications, real-world deployment challenges, and limitations of our study, outlining future research directions. Finally, Sect. 7 concludes the paper with a summary of our contributions.

2 Background

This section establishes the technical background essential for understanding the proposed framework. We begin by outlining the O-RAN architecture that forms the technical foundation for this research, focusing on the Non-RT RIC, rApps, and their associated interfaces. Following this, we closely examine the recent AI/ML approaches

for network energy saving and identify the critical research gap that our proposed framework aims to address.

2.1 Open RAN architecture

The O-RAN Alliance defines a disaggregated, intelligent, and open architecture for the RAN. As illustrated in Fig. 1, the O-RAN architecture is composed of several key components, with a particular focus in this work on the Non-RT RIC [3]:

- **Service Management and Orchestration (SMO):** The highest-level entity responsible for the end-to-end management, orchestration, and automation of O-RAN systems. It includes operations and maintenance (OAM) which performs network management via O1 interface.
- **Non-real-time RIC (Non-RT RIC):** A logical function within the SMO responsible for non-real-time (1s) control and optimization of RAN resources. It consists of the non-RT RIC framework and hosts one or more rApps. These rApps leverage data analytics and AI/ML to derive policies and enrichment information for RAN optimization.
  - **rApps:** Specialized applications hosted by the Non-RT RIC that provide various services for network performance management and enhancement.

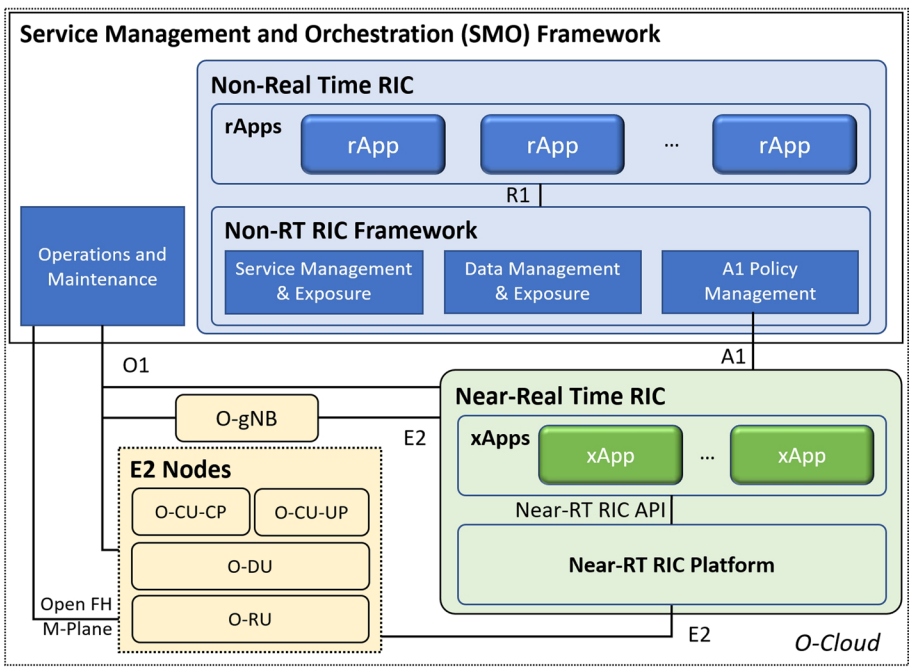

Fig. 1 Overall architecture of the O-RAN system. The figure shows the O-RAN reference architecture. The SMO provides end-to-end management, orchestration, and automation. Within the SMO, the non-RT RIC performs non-real-time control and hosts rApps. The near-RT RIC hosts xApps for control in the 10 ms to 1 s range and terminates A1 and E2 interfaces. E2 Nodes include the O-CU and O-DU. The R1 interface offers service-based connectivity between rApps and the non-RT RIC framework for data management and policy control. The A1 interface links the non-RT RIC and near-RT RIC, and the O1 interface connects the SMO, RICs, and E2 nodes for performance data and configuration. All components are deployed on the O-Cloud

  - **Non-RT RIC Framework:** The platform component of the Non-RT RIC that provides the underlying functionalities for rApps. It interfaces with rApps via the R1 interface offering capabilities such as service/data management and exposure, and A1 policy management. It is also connected with the Near-RT RIC via the A1 interface.

- **Near-real-time RIC (Near-RT RIC):** A logical function responsible for near-real-time (10ms to 1 s) control of RAN functions. It is composed of the Near-RT RIC Platform and hosts xApps.

  - **xApps:** Applications hosted by the near-RT RIC that perform near-real-time operations, such as radio resource management and data analysis.
  - **Near-RT RIC Platform:** The platform that includes the terminations of the E2 and A1 interfaces and provides the necessary components to run and manage xApps.

- **E2 Nodes:** O-RAN managed network functions, which include the O-Central Unit (O-CU) and O-Distributed Unit (O-DU).
- **O-Cloud:** A cloud computing platform that hosts O-RAN entities, providing a virtualized infrastructure with management and orchestration capabilities.

Among these components, this paper focuses on the development of AI/ML-based rApps within the Non-RT RIC environment. The key interfaces that govern rApp operation are as follows:

- **R1 Interface** provides the service-based connectivity between rApps and the Non-RT RIC Framework, enabling rApps to access core services like data management and policy control.
- **A1 Interface** connects the Non-RT RIC to the Near-RT RIC, serving as the channel for conveying policies, enrichment information, and AI/ML model updates.
- **O1 Interface** acts as the communication channel linking the SMO, the RIC, and E2 nodes. For AI/ML-driven rApps, this interface is essential for the collection of Performance Management (PM) data and the dispatch of Configuration Management (CM) commands (e.g., cell activation/deactivation).

This standardized architecture, particularly the interplay between rApps, the non-RT RIC, and the underlying network via the R1 and O1 interfaces, provides the crucial operational context for the development framework proposed in this paper.

2.2 Challenges in rApp implementation

Network energy saving has become an important objective for 5G/6G systems, driving extensive research into AI/ML-driven approaches. Studies such as those employing reinforcement learning for dynamic cell control [6] and predictive deep learning for base station on/off optimization [7] have demonstrated considerable potential for improving energy efficiency at the algorithmic level.

However, many prior studies primarily validate the theoretical performance of the AI/ML models within controlled simulation environments [9]. Consequently, the practical

and system-level challenges of integrating and validating network energy saving control logic in operational networks have received less attention than algorithm-level performance optimization. Meanwhile, industry interest in this area has been increasing [10, 11]. These challenges include interfacing the proposed algorithms with systems operating across diverse network environments, managing the end-to-end pipeline from data collection to control actuation, and rigorously verifying that the system behaves as intended under dynamic, real-world conditions.

This *algorithm-to-operation gap* becomes particularly pronounced in the emerging Open RAN paradigm. In the O-RAN context, the aforementioned intelligent functions must be implemented as independent software modules known as rApps, which operate on the non-RT RIC. This paradigm introduces additional, O-RAN-specific complexities, such as adherence to standardized interfaces and interoperability in a multi-vendor environment. Consequently, the absence of an integrated development framework that facilitates the rapid translation of algorithmic ideas into verifiable rApp prototypes and systematically validates their functional integrity highlights a valuable opportunity for contribution and serves as the primary motivation for this paper.

2.3 Related work

Open RAN has attracted significant attention as a foundation for disaggregated and programmable RAN deployments. Prior tutorial and survey papers summarize the O-RAN architecture, open interfaces, and the roles of the non-RT and near-RT RICs in enabling data-driven automation and third-party applications across different time scales [1, 2]. O-RAN Alliance specifications further provide a common reference for the logical functions and interfaces underpinning such deployments, including the architecture description, management-plane interfaces (e.g., O1), and the R1 service framework that rApps rely on during integration and operation [12–14]. Together, these works motivate development and validation workflows that connect algorithm design to deployment-oriented evaluation.

Sustainability has also emerged as a key objective for 6G and beyond, where energy consumption and operational efficiency are treated as primary design constraints. Recent work discusses challenges and opportunities of Open RAN in this context, including how open interfaces and AI-enabled control can support energy-aware operation while introducing additional integration and evaluation requirements [15]. In parallel, O-RAN Alliance research outputs and white papers outline the evolution toward 6G and identify network energy saving as an important theme, including the roles of the SMO and RIC ecosystem in supporting energy saving capabilities in O-RAN deployments [11, 16]. These discussions indicate that energy saving policies should be assessed not only by algorithmic trade-offs but also by their realizability through standards-aligned interfaces and operational workflows.

Beyond high-level discussions, several experimental platforms and toolchains integrate AI/ML control logic with O-RAN-based components and simulated or emulated RAN environments to enable closed-loop experimentation. Representative examples include ColO-RAN and OpenRAN Gym, which provide programmable O-RAN components and experimentation pipelines for learning-based closed-loop control [17, 18]. Colosseum further presents a large-scale emulation infrastructure as an Open RAN

digital twin to support repeatable testing and validation of O-RAN software and AI control loops [19]. End-to-end private 5G and Open RAN testbeds such as X5G further complement this direction by offering integrated platforms for repeatable system-level validation and prototyping [20].

Recent work has also investigated practical aspects of developing and operating O-RAN applications. Prior studies discuss lessons learnt and recurring challenges encountered during xApp design and evaluation, including integration complexity and experimentation issues in O-RAN based closed-loop settings [21]. Complementary efforts propose operational mechanisms for application onboarding and deployment, aiming to streamline how xApps are packaged, published, and managed in O-RAN environments [22]. From the non-RT RIC perspective, a lifecycle-oriented rApp case study describes an end-to-end process from implementation to execution for a concrete rApp use case [23].

Overall, the above literature spans architectural foundations as well as practical considerations for developing and operating O-RAN applications and for evaluating closed-loop control in experimental environments.

3 Methods: proposed open RAN development framework

To bridge the gap between the theoretical design of advanced AI/ML algorithms and their practical deployment within the standardized O-RAN architecture, we propose an integrated and standards-compliant framework for rApp development and verification. This framework is designed to support the entire lifecycle of an AI/ML-driven rApp, from data generation and model training to deployment and closed-loop evaluation. Its modular design and standardized interfaces ensure both flexibility and extensibility, thus facilitating seamless integration with third-party tools and future O-RAN components.

3.1 Framework architecture and components

The overall architecture of our proposed framework is depicted in Fig. 2. This figure is intended as a conceptual workflow and validation view, rather than to claim architectural novelty. The framework comprises four modular, interconnected components that collaborate to streamline rApp development and validation. In this paper, we use the term RAN emulation environment to denote a controllable emulation setting that mirrors selected RAN behaviors and KPIs for closed-loop validation. This environment captures the KPI generation and control response needed for functional verification, but it is not intended to replicate all protocol stack behaviors and implementation details of an operational RAN. In our framework, the RAN emulation environment corresponds to the RAN emulator component used for data generation and closed-loop validation. For consistency, we refer to this environment as the RAN emulator throughout the remainder of the paper. The four framework components are summarized below:

- **Non-RT RIC Framework**: At the core of our system, this component performs the essential functionalities of an O-RAN compliant non-RT RIC. We use modules from the O-RAN Software Community (OSC) K-Release for service management (rApp/service registration and discovery) and policy management (A1 policy handling via the A1 interface) [24]. Critically, it provides modules for data manage-

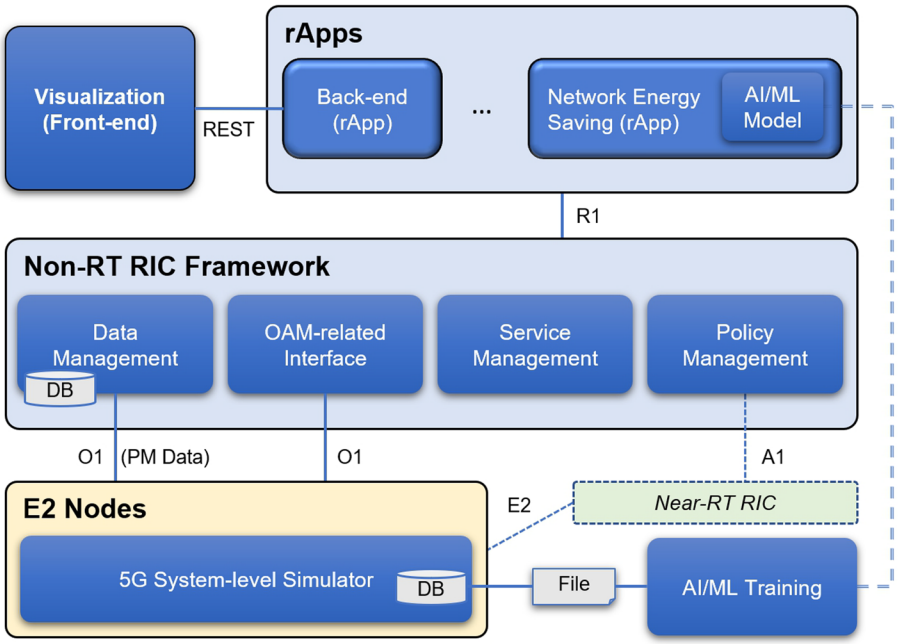

Fig. 2 Open RAN development framework for rApp implementation. The figure illustrates a modular framework that bridges AI/ML design and deployment in an O-RAN compliant setting. The non-RT RIC framework is the core component, providing service management, data management, A1 policy management, and OAM-related functions, implemented using OSC K-Release modules. A containerized rApp bundles a pre-trained model, preprocessing, and logic that converts inferences into O-RAN control messages. The rApp uses the R1 interface for registration, discovery, data access, and policy control. A 5G system-level simulator emulates E2 Nodes and UEs, and generates time-series datasets for training and validation under varied mobility and traffic. A web-based dashboard exposes real-time KPIs, rApp decisions, and controls via REST APIs, enabling closed-loop evaluation

ment (performance measurement data collection) and the OAM-related interface (E2 Nodes control via O1 interface).

- **AI-driven rApp (Case Study)**: This component represents the application logic developed by the researcher. Our framework is designed to host any containerized rApp. For our case study, the rApp contains the pre-trained AI/ML model, data preprocessing modules, and the logic to translate model outputs into O-RAN compliant control messages (e.g., cell on/off). It interacts with the non-RT RIC Framework exclusively through the R1 interface.
- **RAN Emulator (5G System-level Simulator & RAN Data Generator)**: This component serves a dual purpose. Firstly, it acts as a configurable 5G RAN environment, simulating the behavior of multiple E2 Nodes (gNBs), UEs, and their radio channel dynamics. Secondly, it functions as a high-fidelity dataset generator. By running various scenarios (e.g., varying UE mobility, traffic patterns), it produces the necessary time-series data (e.g., cell load, throughput) required for training and validating AI/ML models. This obviates the need for extensive real-world data collection in the initial development phases.
- **Visualization & Control Dashboard**: This is the human–machine interface of the framework. Implemented as a web-based front-end, it provides an intuitive graphical user interface (GUI) for real-time monitoring of the RAN simulation, visualiz-

ing KPIs, and observing the rApp's operational decisions. It also includes a control panel that allows the developer to manage rApps, for example, onboarding and executing rApps/RIC modules. This component communicates with a back-end rApp via REpresentational State Transfer (REST) Application Programming Interfaces (APIs).

3.2 rApp development workflow

A primary contribution of our framework is a structured, end-to-end workflow for the lifecycle of AI/ML-driven rApps. This process guides developers through five stages, from initial concept to validated application. While the five stages follow a common high-level lifecycle, the contribution of this work does not lie in the stage labels themselves. Instead, we formalize and implement the concrete engineering elements required to make a rApp lifecycle executable and verifiable through standards-consistent R1 and O1 interactions. In particular, we specify measurement normalization into a unified schema with consistency checks, intent-to-configuration translation with identifier binding and validation, and structured logging to enable end-to-end traceability and repeatable closed-loop functional validation across data collection, decision making, and actuation.

Figure 3 provides a standards-aligned view of these elements at the R1 and O1 interfaces. On the data path, O1 PM data (data file) are ingested by the Non-RT RIC framework and delivered to the rApp through R1 data services (e.g., subscription and push). On the control path, the rApp issues R1 configuration read/write requests, and the requested configuration changes are translated into O1 CM operations identified by Managed Object Instance (MOI) toward the E2 nodes (emulated in this study using VIAVI AI RSG), with the corresponding responses returned to the rApp over R1. The A1 policy creation exchange is shown only to provide a complete functional view of the non-RT RIC modules and is not exercised in the case study.

To support implementation and testing, we explicitly define integration checkpoints and expected artifacts at the R1 and O1 interfaces to make the procedure executable and repeatable. At each checkpoint, we record time-stamped logs that associate each O1 PM snapshot with the corresponding R1 rApp request and the resulting O1 CM operation, including request and response identifiers across R1 and O1 transactions, MOI and parameter fields for O1 CM, and data file identifiers for O1 PM. The five stages are described as follows:

Stage 1. Data Generation: The developer configures the RAN environment module, using either a built-in simulator for rapid prototyping or an external RAN emulation platform for high-fidelity scenarios. By running simulations under varied network conditions, e.g., different traffic loads and UE mobility patterns, the system produces a comprehensive dataset that typically includes time-aligned performance measurements, such as cell load, throughput, and user count, and, where applicable, corresponding optimal control actions derived from a baseline oracle.

Stage 2. AI/ML Model Training (offline): The dataset is used to train one or more candidate AI/ML models in an offline setting. This allows the use of standard

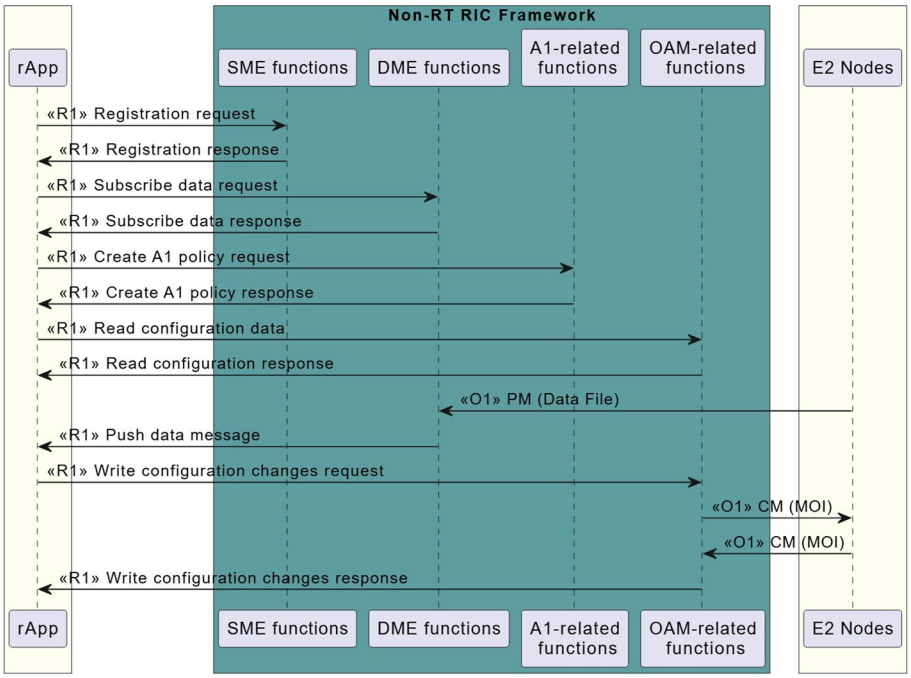

Fig. 3 Interworking flow diagram between the rApp and the Non-RT RIC Framework. The figure describes R1–O1 interworking flow between the rApp and the non-RT RIC framework. O1 PM data (data file) are ingested by the OAM-related functions and exposed to the rApp over R1 (subscription and push), while R1 configuration read/write requests are translated into O1 CM operations identified by MOI toward the E2 nodes (emulated in this study by VIAVI AI RSG). The A1 policy creation exchange is shown only to provide a complete functional view of the non-RT RIC modules and is not exercised in the case study

machine learning libraries, such as TensorFlow or PyTorch, and dedicated compute resources without affecting the RAN simulation environment. The output is a trained, serialized model artifact.

Stage 3. rApp Integration and Containerization: The trained model is packaged into a standardized rApp. The inference logic is wrapped with service code that communicates via R1, including subscriptions to required data types and formatting of outputs as A1 policies or OAM commands. The application is then containerized, e.g., as a Docker image, producing a self-contained and portable deployment unit.

Stage 4. Onboarding and Execution: The containerized rApp is onboarded to the Non-RT RIC framework. Through R1, the rApp registers its services and begins receiving a live stream of simulated network data. The non-RT RIC manages the rApp lifecycle and ensures delivery of the required measurements.

Stage 5. Closed-Loop Validation and Analysis: The rApp makes decisions based on the incoming data stream. These decisions are translated into control commands and sent back to the RAN environment module, altering its state and thereby closing the loop. The visualization and control dashboard provides real-time insights, enabling observation of the rApp's impact on network KPIs, analysis of its behavior, and fair comparison of different models under identical, reproducible conditions.

Although the case study uses an offline-trained supervised model for clarity, the proposed closed-loop workflow is designed to be extensible to other methods. For example, a reinforcement learning-based rApp could be explored by leveraging measurements available through the R1 interface and applying the resulting control decisions via O1 configuration operations within the same workflow. The framework can record time-stamped logs of measurements, decisions, and applied configurations to support traceability and reproducible testing in the emulated environment. Importantly, this paper does not present or claim any reinforcement learning experiment, and the reinforcement learning related discussion is included only to indicate the potential compatibility of the workflow.

3.3 Integration with third-party validation platform

To demonstrate the framework's openness and to ensure the credibility and reproducibility of the case study results, the framework is designed to integrate with industry-grade validation platforms in a standards-compliant and interoperable manner. Accordingly, in this study, we replace the 5G system-level simulator, which was used as the built-in RAN environment module in our prior work [8], with the VIAVI AI RAN Scenario Generator (RSG) [25], an O-RAN-aligned RAN emulation platform that provides a detailed, standards-compliant simulation environment. Hereafter, we refer to this platform as the RAN emulator. To integrate the RAN emulator into the non-RT RIC framework, we implement explicit mappings for measurement collection and configuration delivery that preserve O-RAN interface semantics. Specifically, we normalize emulator-exposed measurements into a unified schema with consistency checks, translate rApp intents into configuration operations with identifier binding and validation, and record structured logs to support observability and end-to-end traceability of the closed-loop workflow. These integration checkpoints and artifacts are included to make the closed-loop procedure under standards-consistent R1 and O1 interactions explicit.

On the data path from the RAN emulator to the non-RT RIC, the RAN emulator generates high-fidelity RAN performance measurements, including standard PM counters, which are periodically reported to the Non-RT RIC framework. The OAM-related interface service within the Non-RT RIC framework continuously monitors the data and parses and normalizes the contents. The resulting metrics are forwarded to the data management module and exposed to onboarded rApps over the R1 interface. Meanwhile, on the control path from the non-RT RIC to the RAN emulator, when the rApp makes a control decision such as switching a cell on/off, the intent is delivered to the non-RT RIC framework via the R1 interface. The OAM-related interface module then translates this intent into a O1 CM format and transmits it to the RAN emulator via the O1 interface.

A key integration challenge is maintaining semantic consistency when mapping the RAN emulator's measurement reporting and configuration change handling to the Non-RT RIC data and control model. To preserve functional integrity in standards-consistent closed-loop validation, the OAM-related interface function enforces semantic alignment between heterogeneous emulator outputs and the data management and exposure (DME) types exposed over R1 by normalizing raw O1 PM reports to ensure schema alignment, unit consistency, and reporting-interval alignment. Without this step, unit,

granularity, or reporting-interval mismatches can bias rApp feature interpretation, leading to inconsistent AI/ML inference and mis-calibrated control actions, and ultimately reducing the reliability and reproducibility of the closed-loop validation. On the control path, rApp outputs received over R1 are translated into O1 CM operations with consistent identifier binding, admissible-range checks, and verification of applied state transitions. Together with time-stamped logging that links PM snapshots, rApp decisions, and applied CM actions, the framework prevents silent semantic mismatches and treats violations as explicit validation failures, making the closed-loop experiment auditable and repeatable.

This integration enables end-to-end execution and functional validation of the rApp workflow in a controlled emulated environment, including R1-based data exposure and O1-based application of CM updates to the RAN emulator. To avoid ambiguity, we clarify that the focus here is not to claim the general capability of closed-loop experimentation and repeatable validation, which has been discussed in prior work, but to make the rApp procedure executable and functionally verifiable through the R1/O1 interactions used in integration and operation. In particular, during closed-loop execution, the framework consistently records and time-aligns the input measurements, rApp decisions, and the configuration updates applied via O1. This traceability enables functional verification of whether the end-to-end rApp procedure is carried out as intended and supports diagnosing issues when problems occur.

4 Case study: network energy saving rApp

To demonstrate the practical utility of the proposed development and validation framework, we conduct a concrete case study on network energy saving, a prominent O-RAN use case designed to reduce operating expenditures (OPEX) for mobile network operators [26]. The case study is designed to validate the end-to-end executable integration procedure and interface-consistent closed-loop signaling, rather than to provide a comparative benchmarking study across learning algorithms. Accordingly, we position the validation as a pre-deployment integration step by enabling end-to-end execution of standards-aligned R1/O1 procedures and collecting auditable artifacts (captured interface messages and time-stamped logs) to trace the measurement–decision–configuration chain, rather than treating it as a substitute for production deployment testing. This section proceeds by first formulating the network energy saving problem, then detailing the control logic of the rApp and the AI/ML model that serves as its predictive engine, and finally describing the experimental setup used for functional validation of the rApp within our framework.

4.1 Problem formulation

In this study, we formulate the network energy saving challenge as a predictive control problem. Unlike traditional reactive approaches that deactivate cells only after an observed drop in traffic load, the proposed strategy anticipates future network states to facilitate timely, well-informed energy saving decisions. This predictive capability is essential to reduce power consumption without compromising the required quality of service (QoS).

The state of the RAN is characterized by a set of KPIs that reflect the traffic load on each cell. For this study, we focus on two primary metrics for each cell m at a given time t: the Physical Resource Block (PRB) utilization, denoted as U_m^t, and the number of connected UEs, denoted as N_m^t.

Given this context, the core task for the energy saving rApp is a two-stage process of prediction and control. The rApp must first accurately forecast the future state of the network by using a history of past observations up to time t to predict the PRB utilization ($\hat{U}_m^{t+1}$) and the number of connected UEs ($\hat{N}_m^{t+1}$) for each cell at the next time step. Based on these predicted future values, the rApp must then execute a control to determine the optimal operational state for each cell, which involves deactivating cells whose predicted load falls below predefined thresholds.

In essence, the core problem is to design an rApp that can leverage historical data to predict the future network state, and then use that prediction to make and execute control decisions that maximize energy savings without compromising user experience.

4.2 Energy saving rApp: design and operational workflow

To solve the predictive control problem formulated in the previous section, we designed the rApp with the specific architecture and internal functions as shown in Fig. 4. The lifecycle of the rApp is divided into an initialization phase, which configures the environment required for execution, and an operational phase, where core logic is invoked periodically or triggered by events.

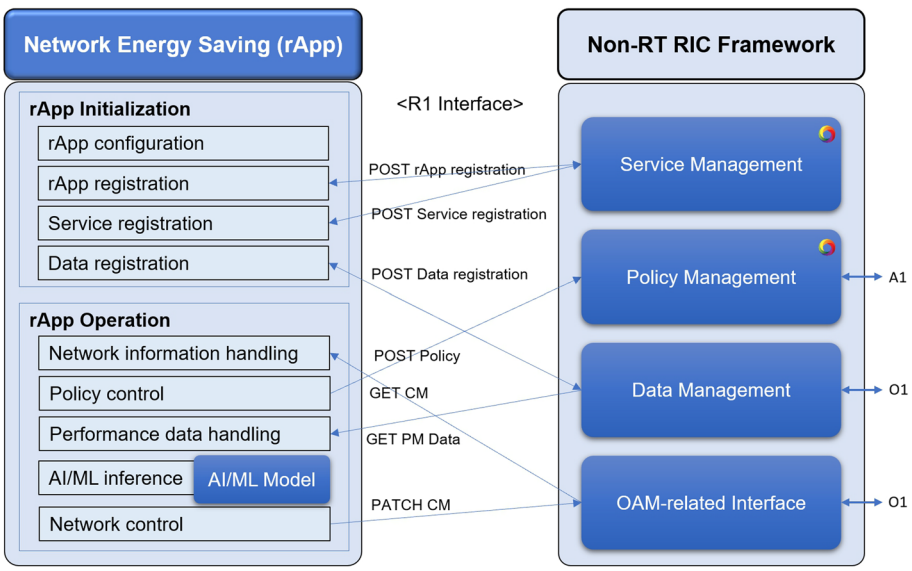

Fig. 4 Architecture and operational workflow of rApp. The figure outlines an rApp architecture for AI/ML-based predictive control under O-RAN systems. During initialization phase, the rApp configures its execution environment and establishes R1 services to the non-RT RIC framework, including rApp, service, and data registration. During operation phase, network information handling obtains configuration from E2 Nodes via OAM CM procedures. Policy control manages A1 policies. Performance data handling receives PM data over the R1 interface, and forwards the data to the model. The AI/ML inference runs periodically to generate cell control decisions, and network control sends the control messages to E2 Nodes over the O1 interface using the CM procedure to apply configuration changes

4.2.1 rApp initialization and setup

Before the main operational workflow begins, several essential setup procedures are executed:

- **rApp configuration**: Configures the rApp execution environment, including the web server, threading, and local storage. It also establishes the R1 service link with the non-RT RIC Framework, which is defined through a RESTful API.
- **rApp/Service registration**: Using the Common API Framework (CAPIF), the rApp registers itself and its services to expose its capabilities, thereby enabling discovery and use by other rApps.
- **Data registration**: Based on the O-RAN DME data model, this function registers the required PM data types and their reporting intervals with the DME module, which is a prerequisite for receiving measurement data during operation.

4.2.2 Operational functions

Once initialized, the intelligent service is carried out by the interaction of the following core operational functions:

- **Network information handling**: Collects base station configuration data (Managed Element information), which is essential for running control operations, from E2 Nodes via the OAM CM procedures.
- **Policy control**: Transmits policy control messages to the Near-RT RIC through the A1 interface to update network management policy, typically in response to long-term measurement data.
- **Performance data handling**: Periodically receives the configured PM data and forwards them to the AI/ML models. The rApp acquires filtered data from the Data Management module via the R1 interface, enabling efficient use for higher-level functions such as AI/ML-based analysis.
- **AI/ML inference**: Performs AI/ML inference to determine cell control decisions. The AI/ML operation is executed periodically, e.g., every minute.
- **Network control**: Transmits control messages to E2 nodes through the O1 interface to execute cell on/off operations, using the CM procedure to implement configuration changes.

For our network energy saving use case, these functions are orchestrated to form a specific operational workflow, as shown in Fig. 5. The process begins with the Performance data handling function, which is configured to periodically collect time-series data of the key KPIs for this scenario: PRB utilization and the number of connected UEs. The collected KPI data are then passed to the AI/ML inference function, where an embedded model forecasts the KPI values for the next time slot. Finally, the internal logic decides whether to deactivate a cell based on the predicted values, and the network control function executes this decision by sending a cell-off command to the corresponding E2 Node via the O1 interface.

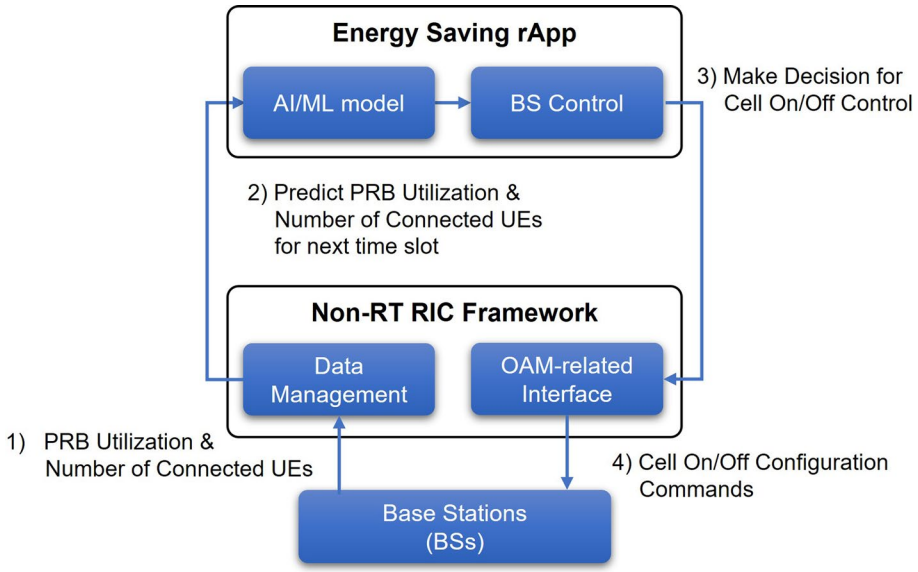

Fig. 5 AI/ML-based energy saving process. The figure depicts the workflow of an AI/ML-based energy saving rApp. The process begins with the data management function, configured to collect time-series measurements, specifically, PRB utilization and the number of connected UEs. These measurements are forwarded to the AI/ML inference function, where an embedded pre-trained model forecasts the values for the next time slot. Based on the predicted values, the internal logic decides whether to deactivate a cell. If deactivation is determined to be necessary, the network control function issues a cell-off command to the corresponding E2 node via the O1 interface and the base station applies the configuration change

4.3 AI/ML model for network energy saving

At the technical core of the predictive control capability outlined in our rApp workflow is an AI/ML model. This model is embedded within the AI/ML inference function and is responsible for forecasting future network states from historical data. The objective of this study is not to propose a new SOTA model, but rather to demonstrate the integration of a proven architecture into the rApp to complete the intelligent control loop. Given the inherent time-series nature of RAN data, we employed a Long Short-Term Memory (LSTM) network, a type of recurrent neural network (RNN) architecture well-suited for sequence prediction tasks [27].

As illustrated in Fig. 6, the model's architecture is defined by its input–output structure and internal layers:

- **Input:** The model takes a sequence of historical KPI data over the past L time steps as input. Specifically, the input at time t is the sequence $\{\mathbf{x}_{t-L+1}, \ldots, \mathbf{x}_t\}$, where each $\mathbf{x}_t$ is a $2M$-dimensional feature vector containing the PRB utilization and the number of connected UEs for all M cells.
- **Architecture:** The input sequence is processed by a stack of three LSTM layers, each with 256 hidden states, to capture temporal dependencies. The output from the final LSTM layer is then passed through two fully-connected (dense) layers for further non-linear transformation.
- **Output:** The model's final output is a vector, $\hat{\mathbf{y}}_{t+1}$, which contains the predicted values of PRB utilization and the number of connected UEs for each of the M cells

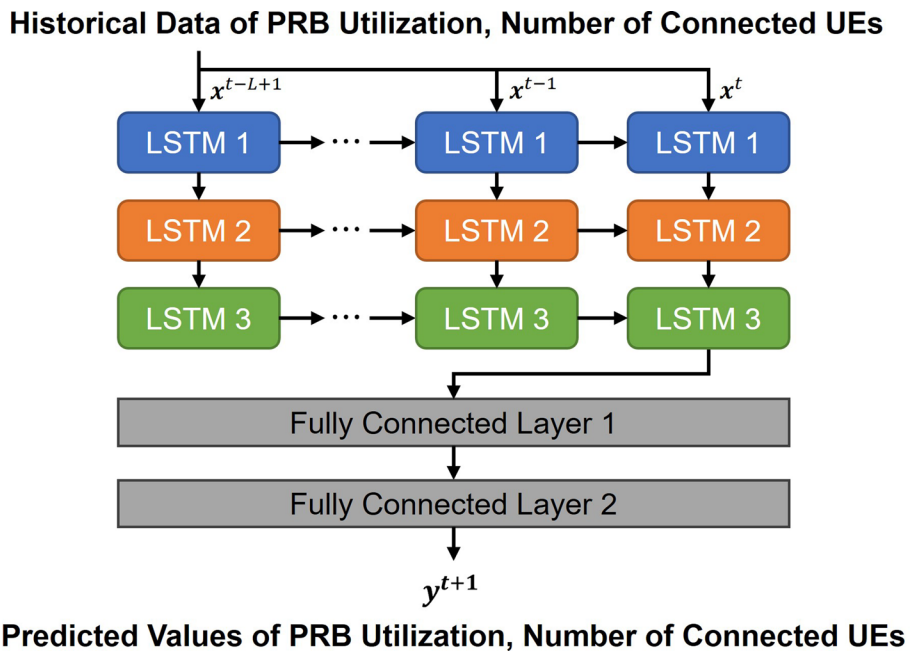

Fig. 6 The LSTM-based neural network architecture used for network energy saving. The figure displays an LSTM-based neural network architecture for a network energy saving scenario. The model is defined by its input–output structure and internal layers. The model takes a sequence of historical data of PRB utilization and the number of connected UEs over the past L time steps as input. The input sequence is processed by a stack of three LSTM layers, each with 256 hidden states, to capture temporal dependencies. The output from the final LSTM layer is then passed through two fully-connected layers for further nonlinear transformation. The model's final output is a vector that contains the predicted values of PRB utilization and the number of connected UEs for the next time slot across M cells. This prediction is then passed to the internal decision logic of the rApp

at the next time slot. This prediction is then passed to the internal decision logic of the rApp.

Effective training of a data-driven model like the proposed LSTM network requires a substantial and diverse dataset of network KPIs. However, obtaining such extensive time-series data from a live production network is often impractical due to operational risks, costs, and the scarcity of suitable public datasets. To overcome this challenge, we leveraged a RAN emulator to model realistic communication environments and to generate rich, dynamic time-series datasets of PRB utilization and UE counts required for both training and testing our model.

4.4 Experimental setup for functional validation

The primary goal of this study is to validate the functional integrity of the rApp as it operates within our proposed framework. To this end, we configured an end-to-end, closed-loop control testbed that integrates our implementation with a RAN emulation environment.

This testbed is built upon the RAN emulator, the same environment used to generate the model training data. The complete setup consists of our Non-RT RIC Framework hosting the energy saving rApp, and the RAN emulator which emulates the E2 Nodes.

Table 1 Cell environment

	Capacity cell	Coverage cell
Number of cells	6	6
Frequency band	B1 (2.1 GHz)	B5 (850 MHz)
Bandwidth	10 MHz	10 MHz
RF model	Isotropic-Urban/UMa-buildings	Isotropic-Urban/UMa-buildings
Duplexing	TDD	TDD
Configured Tx Power	40 dBm	30 dBm
Inter-cell distance	500 m	–

Table 2 UE Environment

	Pedestrian
Number of UEs	50
Initial distribution	Uniformly random within the specified area
Traffic model	GBR, Conversational voice
Target throughput	10 Mbps
Mobility	Moves at a constant speed of 3 m/s

To create a realistic and dynamic network environment, we configured the simulator with specific cell and UE parameters. The network topology comprises a heterogeneous deployment of 12 cells, divided into two types: six coverage cells operating on the 850 MHz band (B5) and six capacity cells on the 2.1 GHz band (B1), with an intercell distance of 500 meters for the latter.

This 12-cell layout is used to provide a controlled and repeatable environment for end-to-end closed-loop validation and debugging. However, operational deployments typically exhibit irregular site placement, heterogeneous cell footprints, and localized traffic hotspots.

The objective of this case study is not to claim that the chosen layout is representative of a real deployment. Instead, it is to demonstrate that the proposed framework can execute and validate the end-to-end rApp procedure in a tractable multi-cell setting that still captures inter-cell interactions through shared load and interference. Here, validate refers to functionally verifying the end-to-end rApp procedure via R1 and O1 interactions, including the traceable linkage between measurements, rApp decisions, and applied configuration updates, rather than evaluating algorithmic performance alone. The framework is not tied to any specific topology, in the sense that the same validation procedure can be repeated by reconfiguring the RAN emulator for alternative deployments.

The simulation also includes 50 UEs, initially distributed randomly, with pedestrian mobility and generating conversational voice traffic. Detailed parameters for the cell and UE environments are summarized in Tables 1 and 2, respectively.

In this configuration, the RAN emulator streams real-time KPI data, such as PRB utilization and connected UE counts, from the simulated environment to the rApp. In response, the rApp performs its AI/ML inference and decision-making logic, sending

any resulting cell on/off commands back to the simulator. This entire closed-loop interaction, from data collection to network control, is conducted over an emulator-facing O1 realization that preserves the O1 information content and the CM/PM semantics, while abstracting low-level transport and protocol details in the RAN emulation environment. This realization reduces integration complexity in the evaluation setup and is mappable to a standards-compliant O1 implementation in operational deployments. This setup allows us to observe and validate the end-to-end operational workflow of the rApp within the specified network conditions. In the following section, we use our framework's visualization capabilities to demonstrate this functionality.

5 Results: framework validation

A key supporting component of the proposed framework is the visualization environment, which renders the complex and dynamic behavior of AI/ML-driven rApps both understandable and systematically verifiable. In this section, we leverage this environment to demonstrate how the network energy saving rApp, designed and implemented for our case study, performs closed-loop control within the O-RAN architecture.

Before presenting the validation results, we clarify the scope of this evaluation and its positioning relative to existing Open RAN experimental platforms. Testbeds such as ColO-RAN and OpenRAN Gym provide valuable environments for developing xApps and studying near-real-time control, whereas our framework targets the non-RT RIC system and the operational validation of rApps through standards-consistent R1/O1 interactions. Therefore, this section does not benchmark the underlying AI/ML policy for quantitative superiority. Instead, it verifies that the R1/O1 closed-loop procedure is executable end-to-end and that its behavior is traceable through consistent artifacts, including the dashboard views and time-ordered message sequence.

This process validates the framework's usability and analytical capabilities. The visualization system consists of a front-end component, which provides a web-based GUI dashboard, and a back-end rApp that manages performance data for visualization. The front-end is implemented in JavaScript and communicates with the back-end rApp through a proprietary REST interface. Implementing the back-end as an rApp is particularly advantageous, as it allows for direct collection of internal data from the non-RT RIC framework and delivery of control commands via the R1 interface. Based on this architecture, the visualization environment offers a suite of graphical interfaces for O-RAN system operations. The key panels utilized in this functional validation process are as follows:

- **Dashboard**: Provides real-time monitoring of the Open RAN system, including overall network performance, cell locations, KPIs for individual cells, and alarm notifications.
- **rApp Panel**: Manages the operational status of registered rApps.
- **Framework Panel**: Manages the operational status of active non-RT RIC framework modules.
- **E2 Nodes Panel**: Displays the control status of key base station functions.
- **Diagnostic Monitor Panel**: Visualizes protocol messages exchanged between non-RT RIC modules.

- **Time Sequence Diagram Panel**: Displays the flow of messages exchanged between non-RT RIC modules in a time-sequenced manner.

The functional validation progresses from a high-level analysis of the rApp's overall behavior and the state changes of individual network elements down to a micro-level examination of internal system message flows. This entire process is systematically traced using the Dashboard, the E2 Nodes panel, and the Time Sequence Diagram & Diagnostic Monitor panels. The role of each visualization panel in this validation process is as follows.

The Visualization & Control Dashboard, as shown in Fig. 7, offers a comprehensive view of the overall system performance and the status of O-RAN components. The dashboard supports structured inspection of measurements, actions, and message exchanges during closed-loop execution. It is presented as an implementation aid for debugging and result interpretation, rather than as a standalone contribution requiring feature-level comparisons with other platforms. It serves as a real-time monitoring tool to verify that each component is operating correctly, displaying key metrics such as cell throughput, power consumption, and the number of active UEs. After the energy saving rApp executes a control action to deactivate a specific cell, a significant reduction in the power consumption metric on this dashboard provides the primary, high-level confirmation that the rApp has achieved its intended goal.

To trace the origin of this high-level performance change, we utilize the E2 nodes panel, as depicted in Fig. 8. This panel visualizes the structure of the E2 node and details key parameters such as configured transmission power and energy saving settings. It illustrates a state where the energy saving operation is active, visually confirming that the

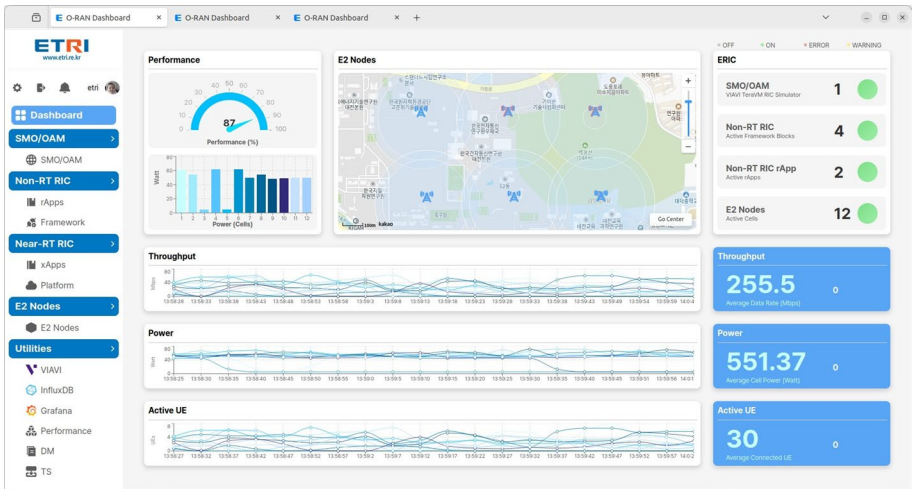

Fig. 7 Visualization & Control Dashboard for component and performance monitoring. The figure visualizes a web-based dashboard used to monitor an O-RAN testbed in real time. The dashboard offers a comprehensive view of the overall system performance and the status of O-RAN components. It serves as a real-time monitoring tool to verify that each component is operating correctly, displaying key metrics such as cell throughput, power consumption, and the number of active UEs. The layout supports closed-loop experiments by exposing live metrics, component status, and cell activity in a single view, allowing operators to verify rApp behavior. After the energy saving rApp executes a control action to deactivate a specific cell, a significant reduction in the power consumption metric on this dashboard provides the primary, high-level confirmation that the rApp has achieved its intended goal

status of the target cell, as dictated by the rApp, has changed from `Active` to `Inactive`. This provides concrete evidence linking the overall performance change observed on the dashboard to a specific physical state change in a network element.

The final stage of the validation is to demonstrate how these changes were orchestrated by showing that the internal logic and its interactions with other system components executed as designed. The time sequence diagram & diagnostic monitor panels, as illustrated in Fig. 9, offer conclusive evidence for this. The time sequence diagram clearly illustrates the sequence of message exchanges over the O-RAN R1, A1, and O1 interfaces, as well as internal flows between modules, using Unified Modeling Language (UML) notation. Through this diagram, we can trace the entire protocol flow, from the initial collection of performance data from the E2 Node to the responsive generation of a control message and its subsequent delivery to the RAN via the Non-RT RIC Framework. Furthermore, the Diagnostic Monitor panel provides operational logs of individual components, enabling a deep, time-sequenced analysis of the complex interactions within the modular architecture.

From an R1/O1 perspective, the sequence in Fig. 9 can be interpreted as an O-RAN-compliant closed-loop messaging flow. On the data path, periodic performance measurements are delivered over O1, processed by the OAM-related functions, and exposed to the rApp through the R1 services of the Non-RT RIC framework. On the control path, network energy saving intents issued by the rApp over R1 are translated into O1 configuration management operations, and the corresponding parameter updates are applied at the E2 Node. The interaction between the Non-RT RIC framework and the OAM-related functions is realized through an internal SMO interface that is not specified by O-RAN, enabling end-to-end R1/O1 signaling while keeping the R1 and O1 procedures unchanged.

In conclusion, by systematically tracing the operation of the rApp from high-level performance results (Fig. 7), through specific node-level state changes (Fig. 8), down

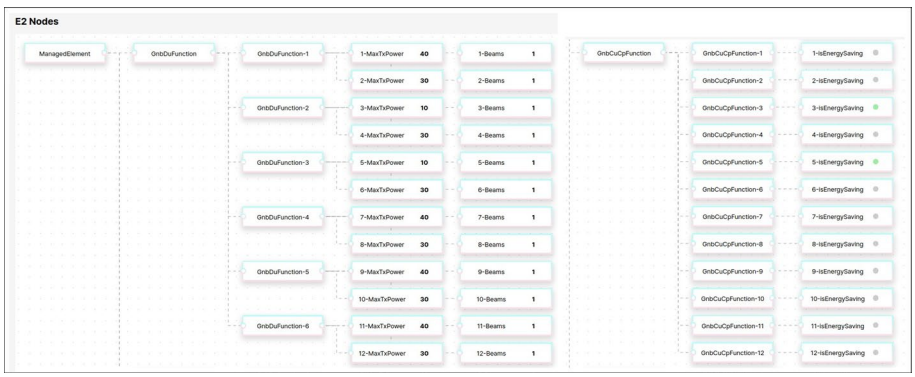

Fig. 8 E2 nodes panel visualizing the structure and energy saving state. The figure presents the E2 nodes view of the O-RAN dashboard. It visualizes the E2 node hierarchy and details key parameters such as configured transmit power and energy saving settings. Specifically, the left panel displays per-cell MaxTxPower and beam information, while the right panel shows per-cell isEnergySaving flags indicating whether energy saving control is active. The snapshot captures energy saving control in effect, with the rApp directing the target cell to change from `Active` to `Inactive`. This visual confirmation links the performance change observed on the dashboard to a specific configuration change in a network element and enables operators to verify the energy saving state at a glance

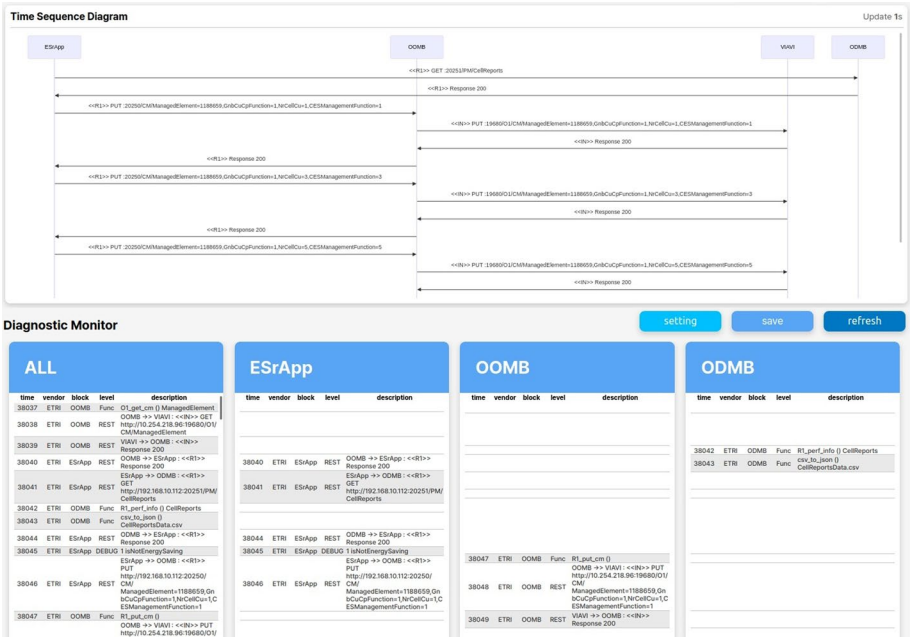

Fig. 9 Time Sequence Diagram & Diagnostic Monitor panels showing protocol flows of components. The figure demonstrates a combined time sequence diagram and diagnostic monitor used to validate an rApp procedure in O-RAN systems. Together, they provide conclusive evidence that the internal logic orchestrates configuration changes through designed interactions with other system components. The time sequence diagram, rendered in UML style, clearly illustrates the sequence of message exchanges over the R1 interface and subsequent configuration updates over the O1 interface, as well as internal flows between modules. It traces the path from the initial collection of performance data at the E2 node to the generation of a control message and its delivery to the RAN via the non-RT RIC framework. The diagnostic monitor panel provides per-component operational logs, enabling time-sequenced analysis of the complex interactions within the modular architecture

to the micro-level message flows between components (Fig. 9), the proposed framework is validated as a robust and practical tool for reliably verifying the functional correctness of AI/ML-driven rApps.

We further validate the generalizability of the proposed development and validation framework by integrating a second distinct application, a QoE Enhancement rApp (QoErApp), within the same Non-RT RIC instance. To avoid conflating this additional application with an algorithmic contribution, we note that QoErApp is implemented as a reference rApp that implements only the essential functions required for end-to-end integration and validation, without designing or embedding any specific AI/ML algorithm. As shown in Fig. 10, multiple rApps can be registered and executed concurrently without structural changes to the framework. The dashboard view and the Docker container status confirm that the QoErApp operates alongside the energy saving rApp. The time sequence diagram also shows that the QoErApp follows the same R1/O1 signaling pipeline by retrieving performance data through the data management module (ODMB) and dispatching configuration intents through the OAM-related module (OOMB). This result provides concrete evidence that the proposed framework is generalizable and can be applied across distinct rApps.

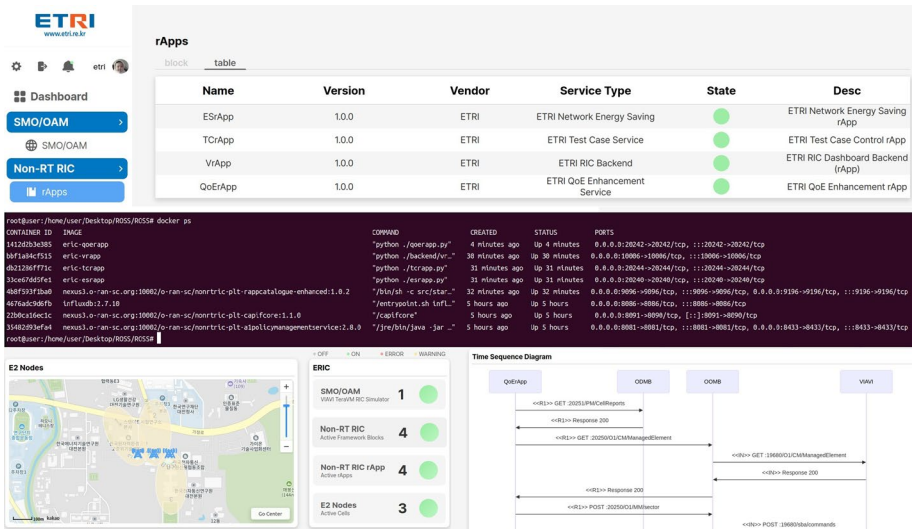

Fig. 10 Consolidated snapshot validating the second application, the QoE enhancement rApp. The figure provides consolidated evidence of framework generalizability through the implementation and validation of a second application, the QoE enhancement rApp, within the same Non-RT RIC instance. The snapshot shows the runtime status view in which the QoE rApp is registered and active alongside other rApps, confirming that the framework supports onboarding and concurrent operation without structural changes. It also presents Docker-based process and container status, verifying that the relevant rApp components and framework modules are running during the validation. In addition, an example scenario configuration and the corresponding O-RAN system operational status are provided to contextualize the test conditions. Finally, the QoE rApp flow diagram summarizes the end-to-end procedure used for functional validation and highlights that the same standards-consistent closed-loop workflow is reused for a distinct application, including measurement retrieval and configuration actuation via the R1 and O1 interaction paths

6 Discussion

This section discusses how the proposed framework helps bridge the *algorithm-to-operation gap* between AI/ML research and real-world RAN operations. We first examine the practical implications of the framework and then address key operational challenges, including considerations for scalability and generalization. We conclude by acknowledging the limitations of this study and presenting our vision for future work toward trustworthy RAN automation.

6.1 Practical implications of the proposed framework

The primary contribution of this paper is a framework that accelerates the end-to-end development lifecycle of AI/ML-driven rApps, enabling the rapid translation of ideas into reliable prototypes. By abstracting the complexities of the underlying RAN, the framework allows developers to focus on core AI/ML logic. Its integrated visualization dashboard offers an intuitive, real-time view of the impact on network state and KPIs, significantly facilitating the debugging and functional validation process.

From the perspective of network operators, the proposed framework provides a practical sandbox environment for objectively testing and benchmarking rApps from multiple vendors under identical conditions. This capability supports the O-RAN principles of openness and interoperability, allowing operators to mitigate deployment risks and make data-driven adoption decisions. Ultimately, by streamlining the process of

validating RAN intelligence, the framework acts as a key enabler for network automation, contributing to reductions in both OPEX and capital expenditures (CAPEX).

6.2 Real-world deployment challenges

The deployment of AI/ML in commercial networks introduces several principal concerns, primarily related to data uncertainty, computational resource limitations, and challenges in system integration and security. Data collected from operational networks is often subject to latency, noise, and loss, which can degrade the predictive accuracy of AI/ML models and lead to undesirable control actions or service disruptions. In addition, many AI/ML models require substantial computational resources, making them sensitive to operational constraints. Furthermore, a deployed rApp must integrate seamlessly with existing management systems, operate reliably in conjunction with other rApps, and address security vulnerabilities that arise from enabling external control over RAN functions.

In this context, while the current case study evaluates a single network energy saving rApp in isolation, operational deployments often involve multiple rApps running concurrently with partially conflicting objectives. Our framework can host multiple rApps on the same Non-RT RIC instance and record per-rApp decisions through a shared logging and visualization pipeline under identical RAN emulator scenarios, enabling controlled studies of multi-rApp interactions. We also note that conflict mitigation is an active topic within the O-RAN Alliance, including ongoing work on R1 services-related conflict mitigation and A1 policy conflict mitigation. However, the current implementation does not enforce an arbitration or conflict-resolution mechanism among rApps, and designing and evaluating such mechanisms remains future work.

To address integration and security, our framework adheres to standard O-RAN interfaces, providing a solid foundation for interoperability. To tackle the remaining challenges more systematically, we plan to evolve the framework in two key directions. First, we will introduce capabilities to simulate the non-ideal data conditions characteristic of operational networks, enabling developers to proactively validate model robustness. Second, we will enhance the benchmarking toolkit to support quantitative analysis of the performance–resource trade-off across alternative AI/ML models, thereby helping operators select practical designs tailored to their constraints. These enhancements are intended to mitigate the risk of post-deployment failures and system instability.

6.3 Considerations for scalability and generalization

The success of AI/ML-driven automation depends critically on two factors, scalability and generalization. Scalability is the ability to maintain performance as the network expands, and generalization is the capacity to operate reliably in new and previously unseen environments. A model that is effective in a small-scale setting may lose practical viability in large-scale deployments as computational complexity increases. Similarly, a model overfit to specific traffic patterns can fail to adapt to the dynamic conditions of an operational network, thereby undermining the reliability of the automation system.

Operational scalability is not determined solely by model complexity. It requires that the end-to-end closed-loop pipeline remains operationally stable as the network scale and reporting load increase. Under higher load, bottlenecks typically appear in the

supporting pipeline, including measurement ingestion and feature construction, and may also arise in inference latency depending on the implementation. For an operational discussion, it is therefore important to identify which stage becomes the bottleneck and to describe practical mitigation options. To this end, the framework can record timestamps and structured logs at key interfaces to enable a stage-wise latency breakdown and to pinpoint bottlenecks. Typical mitigations include batching, down-sampling, shortening feature windows, and provisioning additional compute resources at the constrained stage. While full-scale benchmarking is beyond the scope of this paper, this instrumentation provides a concrete basis for systematic evaluation under larger scale conditions.

The proposed framework serves as a research platform for addressing these fundamental concerns. Its integrated simulator provides the flexibility to vary network topologies, user distributions, and traffic profiles. Researchers can leverage this capability to quantitatively analyze the scalability limits of candidate AI/ML models and to systematically validate generalization across diverse scenarios. This foundation also supports the development and validation of next-generation AI/ML models, such as graph neural networks (GNNs) [28], designed for reliable operation in large-scale commercial networks.

6.4 Limitations and future research directions

This study makes a significant contribution by providing a systematic methodology and framework for prototyping and functional validation of AI/ML-driven rApps. Because the work primarily focuses on functional validation, it does not include comparative analysis with other Open RAN development platforms, quantitative benchmarking of the case study rApp against SOTA algorithms, or validation across multiple use cases. In addition, this paper does not evaluate reinforcement learning-based policies or any online learning procedure. As future work, we plan to integrate reinforcement learning-based decision policies into the same closed-loop workflow and systematically evaluate their behavior under operational constraints, including safety constraints, decision latency, and signaling overhead.

However, while this paper focuses on functional validation of an integrated rApp in a closed-loop setting, the proposed framework can be extended to support quantitative benchmarking. For example, energy efficiency can be evaluated using total energy consumption or relative energy saving aggregated over time and across cells, whereas service impact can be characterized using performance measurements collected from the RAN emulator via the O1 interface, such as throughput and PRB utilization. Furthermore, the framework is not tied to a specific RAN emulator and can be instantiated with alternative environments, provided that the required measurements and configuration operations can be mapped to the R1 and O1 interactions used in the workflow. Evaluating additional variants and conducting quantitative comparisons across multiple environments are beyond the scope of this paper and left for future work. For fair comparisons, the same scenario and input traces can be replayed under different policies and baselines, such as an always-on configuration and a simple rule-based control policy. In addition, time-stamped, structured logs enable offline profiling of these metrics and end-to-end behavior without modifying the rApp logic.

To expand this scope, we have defined a concrete short-term roadmap. We will integrate the framework with large-scale hardware testbeds such as Colosseum [17, 29] and

X5G [20], and benchmark SOTA AI/ML algorithms [30]. We will also conduct comparative studies against other development platforms and validate additional Open RAN use cases, including traffic steering and mobility load balancing (MLB) [4], to demonstrate the framework's versatility.

Our long-term vision is to evolve the framework into an automated pre-deployment validation gate within the Open RAN MLOps pipeline. This gate will act as a checkpoint that automatically verifies the functional integrity and performance of new AI/ML models and rApps against predefined scenarios before deployment in an operational network. We expect this capability to enable rapid yet reliable innovation and advance the goal of trusted RAN automation.

7 Conclusion

Open RAN presents a transformative vision for AI/ML-driven network automation through the RIC, yet a significant gap persists between theoretical AI/ML algorithms and their implementation as operationally viable rApp prototypes with validated functional correctness. This *algorithm-to-operation gap* has been a primary barrier to fully realizing the potential of O-RAN. To address this challenge, we designed and implemented an integrated development and validation framework that supports the entire lifecycle of an rApp, from prototyping to functional verification.

The utility of our framework was demonstrated through a case study in which an AI/ML-based network energy saving rApp was designed and implemented. We successfully verified that the rApp prototype adhered to its intended operational logic, confirming the framework's value as a powerful tool for ensuring the functional integrity of rApps. Building upon this validation, we also discussed the practical challenges of operational deployment, including scalability and generalization, and outlined a concrete roadmap for future work. This roadmap includes integration with large-scale hardware testbeds, comparative studies against other Open RAN development platforms, benchmarking of SOTA algorithms, and the exploration of additional use cases to further enhance the framework.

In conclusion, this work provided more than just a software tool. It offered a practical methodology and validation environment to tackle the fundamental difficulty of translating AI/ML research into operationally viable O-RAN intelligence. By lowering the barrier between conceptual algorithms and deployable prototypes, our framework serves as a key enabler, accelerating the innovation cycle within the Open RAN ecosystem and contributing to the realization of a truly intelligent and automated RAN.

Abbreviations

5G/6G	Fifth and sixth generations
AI/ML	Artificial intelligence/machine learning
API	Application Programming Interface
CAPEX	Capital expenditure
CAPIF	Common API Framework
CM	Configuration management
DME	Data management and exposure
GUI	Graphical user interface
GNN	Graph neural network
KPI	Key Performance Indicator
LSTM	Long short-term memory
MLB	Mobility load balancing
MLOps	Machine Learning Operations

Near-RT RIC	Near-Real-Time RAN Intelligent Controller
Non-RT RIC	Non-Real-Time RAN Intelligent Controller
OAM	Operations and maintenance
O-Cloud	O-RAN Cloud
O-CU	O-RAN Central Unit
O-DU	O-RAN Distributed Unit
OPEX	Operating expenditure
O-RAN	Open Radio Access Network
OSC	O-RAN Software Community
PM	Performance management
PRB	Physical Resource Block
QoE	Quality of experience
QoS	Quality of service
RAN	Radio Access Network
REST	Representational State Transfer
RIC	RAN intelligent controller
RNN	Recurrent neural network
RSG	RAN Scenario Generator
SMO	Service Management and Orchestration
SOTA	State of the Art
UE	User Equipment
UMa	Urban Macro
UML	Unified Modeling Language
gNB	Next Generation NodeB
rApp	Non-RT RIC Application

Author contributions
M. Kim, the first and corresponding author, had primary responsibility for the study and contributed to all aspects, including conceptualization and methodology. K. S. Lee primarily implemented the proposed framework and the visualization, and validated the implementation and results. The remaining authors made comparable contributions to study design, data handling, analysis, interpretation, and manuscript review. All authors contributed to writing and revising the manuscript. All authors read and approved the final version.

Funding
This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (RS-2023-00225468, Development of RAN Intelligent Controller for O-RAN intelligence).

Data availability
A dataset was generated during this study, but external sharing is restricted under our institution's policies related to security and confidentiality.

Declarations

Competing interests
The authors declare that they have no conflict of interest.

Received: 20 October 2025 Accepted: 26 February 2026
Published online: 14 March 2026

References

1. M. Polese, L. Bonati, S. D'Oro, S. Basagni, T. Melodia, Understanding O-RAN: architecture, interfaces, algorithms, security, and research challenges. IEEE Commun. Surveys Tuts. **25**(2), 1376–1411 (2023)
2. M. Polese, M. Dohler, F. Dressler, M. Erol-Kantarci, R. Jana, R. Knopp, T. Melodia, Empowering the 6G cellular architecture with Open RAN. IEEE J. Sel. Areas Commun. **42**(2), 245–262 (2024)
3. O-RAN Alliance WG2: Non-RT RIC: Architecture, Technical Specification. O-RAN.WG2.TS.Non-RT-RIC-ARCH-R004-v07.00 (2025)
4. O-RAN Alliance WG1: Use Cases Analysis Report, Technical Report. O-RAN.WG1.Use-Cases-Analysis-Report-R004-v17.00 (2025)
5. J. Wu, Y. Zhang, M. Zukerman, E.K.-N. Yung, Energy-efficient base-stations sleep-mode techniques in green cellular networks: a survey. IEEE Commun. Surveys Tuts. **17**(2), 803–826 (2015)
6. M. Choi, K. Kim, H. Jang, H. Woo, J.S. Pujol-Roig, Y. Wang, H. Yeon, S. Choi, S. Jang, Cell on/off parameter optimization for saving energy via reinforcement learning. In: Proc. IEEE Globecom Workshops (GC Wkshps), Madrid, Spain, pp. 1–6 (2021)
7. S. Kim, J. Son, B. Shim, Energy-efficient ultra-dense network using LSTM-based deep neural networks. IEEE Trans. Wireless Commun. **20**(7), 4702–4715 (2021)

8. M. Kim, K.S. Lee, S. Jung, J.-H. Na, S. D'Oro, L. Bonati, T. Melodia, An open ran development framework with network energy saving rapp implementation. In: 2025 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit), Poznan, Poland, pp. 298–302 (2025)
9. T.P. Fowdur, B. Doorgakant, A review of machine learning techniques for enhanced energy efficient 5g and 6g communications. Eng. Appl. Artif. Intell. **122**, 106032 (2023)
10. L. Kundu, X. Lin, R. Gadiyar, Toward energy efficient ran: from industry standards to trending practice. IEEE Wirel. Commun. **32**(1), 36–43 (2025)
11. O-RAN Alliance SUFG: Potential Energy Savings Features in O-RAN, Technical Report. O-RAN.WP.Potential Energy Savings Features-v01.00 (2025)
12. O-RAN Alliance WG1: O-RAN Architecture Description, Technical Specification. O-RAN.WG1.TS.OAD-R004-v15.00 (2025)
13. O-RAN Alliance WG10: O-RAN O1 Interface Specification, Technical Specification. O-RAN.WG10.TS.O1-Interface. 0–R004-v17.00 (2025)
14. O-RAN Alliance WG10: O-RAN O1 Interface Specification, Technical Specification. O-RAN.WG10.TS.O1-Interface. 0–R004-v17.00 (2025)
15. H. Ahmadi, M. Rahmani, S.B. Chetty, E.E. Tsiropoulou, H. Arslan, M. Debbah, T. Quek, Towards sustainability in 6g and beyond: challenges and opportunities of open ran. IEEE Commun. Standards Magaz. **9**(3), 126–135 (2025)
16. O-RAN Alliance nGRG: O-RAN Towards 6G, Technical Report. RR-2023-01 (2025)
17. M. Polese, L. Bonati, S. D'Oro, S. Basagni, T. Melodia, ColO-RAN: developing machine learning-based xApps for Open RAN closed-loop control on programmable experimental platforms. IEEE Trans. Mobile Comput. **22**(10), 5787–5800 (2022)
18. L. Bonati, M. Polese, S. D'Oro, S. Basagni, T. Melodia, Openran gym: AI/ML development, data collection, and testing for o-ran on pawr platforms. Comput. Netw. **220**, 109502 (2023)
19. M. Polese, L. Bonati, S. D'Oro, P. Johari, D. Villa, S. Velumani, R. Gangula, M. Tsampazi, C.P. Robinson, G. Gemmi et al., Colosseum: the open ran digital twin. IEEE Open J. Commun. Soc. **5**, 5452–5466 (2024)
20. D. Villa, I. Khan, F. Kaltenberger, N. Hedberg, R.S. Silva, S. Maxenti, L. Bonati, A. Kelkar, C. Dick, E. Baena et al., *X5G: An open, programmable, multi-vendor, end-to-end, private 5G O-RAN testbed with NVIDIA ARC and OpenAirInterface* (IEEE Trans, Mobile Comput, 2025)
21. M. Hoffmann, S. Janji, A. Samorzewski, Ł Kułacz, C. Adamczyk, M. Dryjański, P. Kryszkiewicz, A. Kliks, H. Bogucka, Open ran xapps design and evaluation: lessons learnt and identified challenges. IEEE J. Sel. Areas Commun. **42**(2), 473–486 (2023)
22. P. Rodgers, P. Harvey, The xapp store: A framework for xapp onboarding and deployment in o-ran. In: 2025 IEEE 45th International Conference on Distributed Computing Systems Workshops (ICDCSW), pp. 647–652 (2025). IEEE
23. J.S.R. Mallu, Enabling rapp in 5g o-ran: an spectral optimization (so) rapp use case. Master's thesis, Virginia Tech (2024)
24. O-RAN Software Community (OSC): NONRTRIC (Release K). https://wiki.o-ran-sc.org/display/RICNR/Release+K (2025)
25. VIAVI: AI RAN Scenario Generator. https://www.viavisolutions.com/en-us/products/teravm-ai-rsg
26. O-RAN Alliance WG1: Network Energy Saving Use Cases, Technical Report. O-RAN.WG1.Network-Energy-Savings-Technical-Report-R003-v02.00 (2023)
27. S. Hochreiter, J. Schmidhuber, Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
28. Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, P.S. Yu, A comprehensive survey on graph neural networks. IEEE Trans. Neural Netw. Learn. Syst. **32**(1), 4–24 (2020)
29. L. Bonati, S. D'Oro, M. Polese, S. Basagni, T. Melodia, Intelligence and learning in O-RAN for data-driven NextG cellular networks. IEEE Commun. Mag. **59**(10), 21–27 (2021)
30. X.-L. Huang, X. Ma, F. Hu, Machine learning and intelligent communications. Mobile Netw. Appl. **23**, 68–70 (2018)

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.