Computer Modeling in Engineering & Sciences

Tech Science Press

https://doi.org/10.32604/cmes.2026.082095

ARTICLE

Performance Analysis of an AI-Based IDS xApp for Cyberattack Anomaly Detection in O-RAN Near-RT RIC

Hyeonsoo Yu[1] **and Hwankuk Kim**[2,*]

[1]Department of Cyber Security, Kookmin University, Seoul, Republic of Korea
[2]Department of Information Security, Cryptography and Mathematics, Kookmin University, Seoul, Republic of Korea
*Corresponding Author: Hwankuk Kim. Email: rinyfeel@kookmin.ac.kr

Received: 11 March 2026; Accepted: 06 May 2026; Published: 27 May 2026

ABSTRACT: The introduction of the Open Radio Access Network (O-RAN) architecture enhances network flexibility but introduces novel security threats targeting open interfaces and the RAN Intelligent Controller (RIC). Particularly in the Near-RT RIC environment, an effective Intrusion Detection System (IDS) that satisfies strict near-real-time constraints of within 1 s is essential to defend against cyber attacks. This paper proposes an Artificial Intelligence (AI)-based IDS xApp designed for real-time cyber attack monitoring in the O-RAN Near-RT RIC environment, and quantitatively analyzes its anomaly detection performance and inference latency characteristics against multi-layer security threats utilizing Open RAN Centralized Unit(O-CU) network layer data and Open RAN Distributed Unit (O-DU) radio telemetry data. Evaluation using a public dataset (NetsLab 5G O-RAN IDD) on four deep learning models (LSTM, CNN, Transformer, Autoencoder) showed that supervised learning-based models achieved high F1-scores (reaching up to 0.99) on both datasets. Furthermore, their performance variation remained highly stable at approximately the ±0.1 pp level upon transition from the training environment (the Service and Management Orchestration, SMO) to the deployment environment (Near-RT RIC). In the inference latency analysis, the system's scalability was evaluated by increasing the number of prediction instances up to 80,000. The results confirmed that the latency follows a highly predictable linear time complexity ($\mathcal{O}(N)$). Specifically, the LSTM, CNN, and Autoencoder models successfully maintained a response time within 1000 ms even under the maximum load of 80,000 instances across both datasets, whereas the computationally heavy Transformer model experienced resource exhaustion in the KServe inference pod at approximately 20,000 instances, causing the inference process to terminate and rendering further measurement infeasible. Comprehensively, the LSTM model demonstrated the most outstanding balance between performance and operational efficiency by recording stable detection performance, short tail latency (approximately 140 ms at P99), and low training resource consumption. This study experimentally demonstrates the anomaly detection performance of the IDS xApp in the O-RAN near-real-time control environment, and comprehensively verifies its practical effectiveness by considering both inference latency and resource consumption.

KEYWORDS: Open RAN security; near-RT RIC; IDS xApp; AI-driven intrusion detection; inference latency

1 Introduction

Since 5G, mobile communication networks have been evolving with the core directions of openness, disaggregation, and cloud-native transformation of the Radio Access Network (RAN) [1,2]. Open Radio Access Network (O-RAN) has established itself as the representative standard and industry for this transformation, providing technical specifications and open-source implementations centered around the O-RAN Alliance, which involves over 160 operators and vendors worldwide [3].

O-RAN functionally disaggregates the traditional Next Generation NodeB (gNB) into the Open RAN Centralized Unit (O-CU), Open RAN Distributed Unit (O-DU), and Open RAN Radio Unit (O-RU), premised on an architecture where multiple vendors and third-party applications coexist. It ensures interoperability through standardized open interfaces such as O1 (management and configuration), A1 (policy and Artificial Intelligence (AI) model transfer), and E2 (real-time RAN control) [4]. In particular, the RAN Intelligent Controller (RIC) serves as the central pillar for intelligent control; the Non-Real-Time (Non-RT) RIC (>1 s) is responsible for Machine Learning (ML) model training and policy computation, while the Near-Real-Time (Near-RT) RIC (10 ms–1 s) enables near-real-time radio resource management via the E2 interface and AI/ML closed-loop control based on Extended Applications (xApps) [3,5,6].

However, this openness and software-centric architecture structurally expand the attack surface compared to traditional single-vendor closed RANs [7,8]. Open interfaces like E2 and Open Fronthaul increase the risks of eavesdropping, tampering, man-in-the-middle attacks, and Denial of Service (DoS) [9,10]. According to the latest O-RAN Working Group 11 (WG11) Threat Modeling technical report (v8.0, 2026), a significant portion of the identified security threats is closely associated with DoS and network performance degradation [11]. The xApp ecosystem within the Near-RT RIC can also induce new security risks, such as privilege management issues, access control failures, supply chain vulnerabilities, and misbehaviors or misconfigurations. In fact, vulnerabilities (Common Vulnerabilities and Exposures (CVE-2023-42358, CVE-2023-41628)) due to unimplemented access controls have been confirmed in O-RAN Software Community (OSC) H-Release-based implementations, and DoS attacks through routing table manipulation by malicious xApps have also been demonstrated [12–14]. Consequently, recent studies have emphasized the need for intelligent security functions, including IDS xApps deployed in the Near-RT RIC, and have proceeded with the design, implementation, and evaluation of security frameworks like Zero Trust, Secure Slicing, and Federated Learning, as well as various ML/Deep Learning (DL)-based detection models [15–19].

Nevertheless, existing research tends to focus primarily on reporting performance centered around detection accuracy metrics (e.g., F1-score, Accuracy) [20,21]. There are relatively limited cases that systematically and concurrently verify measurement-based inference latency (Mean/P95/P99) to determine compliance with the strict control loop requirements (end-to-end 10 ms) of the Near-RT RIC, along with scalability and time complexity trends associated with increased loads [22–25]. Although xApp scaling is dominated by inference latency constraints rather than resource constraints, studies demonstrating this in the context of security detection are scarce. Furthermore, research quantitatively presenting the maintenance of detection performance in actual deployment environments compared to training environments is insufficient, which remains a critical unsolved challenge in evaluating the practical deployability of Near-RT RIC-based security xApps [16,26].

The objective of this study is to quantitatively verify the detection performance and inference latency characteristics of an IDS xApp deployed in an O-RAN Near-RT RIC environment. To achieve this, O-CU network layer data and O-DU radio telemetry data were utilized to analyze core Key Performance Indicators (KPIs) based on actual measurements in a deployment environment.

The key contributions of this study are summarized as follows:

- **Analysis of IDS xApp Anomaly Detection Performance:** The anomaly detection performance of the IDS xApp was quantitatively verified based on O-CU and O-DU data, focusing on Accuracy, Precision, Recall, and F1-score. On the O-CU dataset, the Transformer recorded an F1-score of 0.9629, while the LSTM and CNN models showed stable performances of 0.9487 and 0.9571, respectively. On the O-DU dataset, all supervised learning models maintained an F1-score reaching 0.99. Furthermore, an analysis of performance variation between the training and deployment environments revealed that supervised models (LSTM, CNN, Transformer) maintained ΔF1 and ΔPrec within or close to the $\pm$0.1 percentage

point(pp) interval. Notably, the LSTM model recorded the smallest fluctuation ($|\Delta F1| = 0.0001$ for O-CU and 0.0006 for O-DU), confirming its superior performance preservation upon transition to the deployment environment.

- **Analysis of Inference Latency and $\mathcal{O}(N)$ Time Complexity:** The inference latency of the IDS xApp in the Near-RT RIC environment was measured based on Mean, P95, and P99 criteria, analyzing latency characteristics as prediction instances grew. Measurements taken while scaling the load up to 80,000 instances demonstrated that structurally optimized models (LSTM, CNN, Autoencoder) maintained a response time well within the 1000 ms constraint. Conversely, the Transformer model's inference was terminated at approximately 20,000 instances due to KServe pod resource exhaustion, rendering further latency measurement infeasible and thereby highlighting the importance of architectural efficiency in meeting strict near-real-time requirements. Additionally, the measured data for the optimized models exhibited a robust linear growth trend without exponential explosion, experimentally confirming that the underlying inference workflow of the proposed IDS xApp guarantees $\mathcal{O}(N)$ scalability under near-real-time constraints.

The structure of this paper is as follows. Section 2 examines the O-RAN architecture and related research on intelligent applications and intrusion detection in the Near-RT RIC environment. Section 3 reviews potential attack entry points through attack modeling in the O-RAN environment, and presents the overall system architecture where the IDS xApp performs anomaly detection based on the O-CU and O-DU attack datasets. Section 4 defines the public datasets used in the experiments, the preprocessing steps, the model configurations, and the core KPI evaluation metrics. Section 5 sequentially verifies the detection performance analysis, near-real-time inference latency characteristics, and time complexity $\mathcal{O}(N)$ according to the increase in instances targeting the deployed IDS xApp. Section 6 conducts a comprehensive discussion based on the measured metrics, and finally, Section 7 concludes the paper.

2 Background and Related Work

2.1 O-RAN Architecture Overview

The O-RAN Alliance is an industry consortium established in 2018 that aims to transition the traditional closed and vendor-dependent RAN architecture into an open, intelligent, and virtualization-based architecture [3]. To achieve this, O-RAN adopts functional disaggregation, open interfaces, and AI/ML-based control structures as its core design principles [1,2].

This study utilizes the framework provided by the OSC to implement and verify O-RAN's open standards in a real-world environment. The OSC provides an open-source software stack that implements the Near-RT RIC, Non-RT RIC, E2 interface, and xApp platform based on the reference architecture defined by the O-RAN Alliance [16].

The O-RAN architecture functionally disaggregates the RAN into the O-RU, O-DU, and O-CU, as illustrated in Fig. 1 [4]. The O-RU is responsible for Radio Frequency (RF) signal transmission and reception, as well as low-level Physical (PHY) layer processing. The O-DU handles the Medium Access Control (MAC) and some Radio Link Control (RLC) functions, while the O-CU manages the upper RLC, Packet Data Convergence Protocol (PDCP), and control plane functions. The O-RAN architecture is fundamentally decoupled into the RAN infrastructure domain, which handles actual radio signal processing and data transmission, and the Service and Management Orchestration (SMO) domain, which performs network management, policy control, and AI/ML-driven optimization. Within this architecture, the RIC serves as the core component responsible for enabling data-driven, programmable control and RAN intelligence [5].

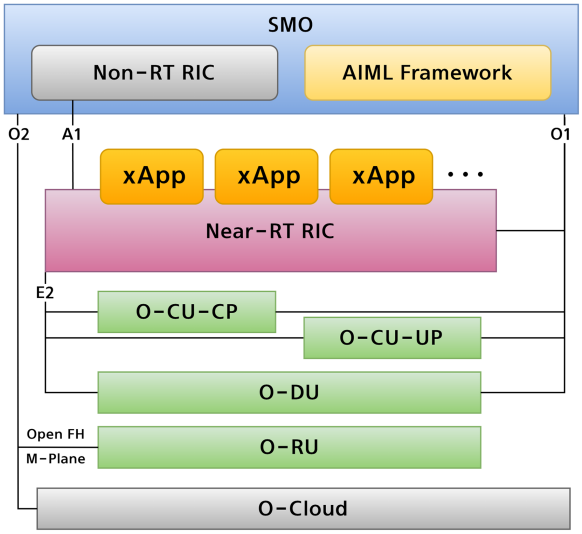

Figure 1: O-RAN architecture overview.

The RIC is divided into two layers based on its time scale. The Non-RT RIC (≥1 s) performs non-real-time functions such as policy formulation, AI model training, and network analytics, and serves as the platform where rApps are executed. It delivers policies to the Near-RT RIC via the A1 interface. The Near-RT RIC (10 ms–1 s) executes a near-real-time control loop and serves as the platform for xApps. This control loop range is explicitly defined by the O-RAN Alliance WG1 specifications, where any processing exceeding the 1-s threshold is categorized under the Non-RT RIC (≥1 s) domain [27]. Therefore, the 1000 ms baseline used in this study is directly derived from the official O-RAN architectural standards rather than an experimental assumption. It connects to the O-DU and O-CU via the E2 interface to collect KPIs, subscribe to events, and send control commands [6]. Thus, O-RAN is an architecture that supports a multi-vendor environment through functional disaggregation and open interfaces, while simultaneously implementing network intelligence through a RIC-based AI/ML control structure [1,28].

In the O-RAN architecture, the radio side includes the Near-RT RIC, O-CU Control Plane (O-CU-CP), O-CU User Plane (O-CU-UP), O-DU, and O-RU [4]. The management side comprises the SMO framework, which encompasses the Non-RT RIC functions [1]. Table 1 summarizes the definitions of the O-RAN components.

Table 1: Definitions of O-RAN components.

Term	Definition
SMO	Fault, Configuration, Accounting, Performance, and Security (FCAPS)-based management framework responsible for integrated RAN control, O-Cloud orchestration, and oversight of the Non-RT RIC and internal applications.
Near-RT RIC	Logical function enabling near-real-time control and optimization of RAN elements via the E2 interface.
Non-RT RIC	Logical function for non-real-time RAN optimization, AI/ML workflow management, and policy guidance for the Near-RT RIC.

(Continued)

Table 1 (continued)

Term	Definition
AI/ML Framework	Framework supporting the complete AI/ML lifecycle in O-RAN, including data processing, model training, and deployment.
O-CU	Logical node hosting the Radio Resource Control (RRC), Service Data Adaptation Protocol (SDAP), and Packet Data Convergence Protocol (PDCP) protocols.
O-CU-CP	Logical node hosting the RRC and the control-plane part of the PDCP protocol.
O-CU-UP	Logical node hosting the user-plane part of the PDCP and SDAP protocols.
O-DU	Logical node hosting the Radio Link Control (RLC), Medium Access Control (MAC), and High-Physical (High-PHY) layers based on a lower-layer functional split.
O-RU	Logical node hosting the Low-Physical (Low-PHY) layer and Radio Frequency (RF) processing (e.g., Fast Fourier Transform/Inverse Fast Fourier Transform (FFT/IFFT)) based on a lower-layer functional split.
O-Cloud	Cloud computing platform providing the physical infrastructure and software components to host O-RAN functions.
xApp	Independent software plug-in executing on the Near-RT RIC to provide functional extensibility and RAN control capabilities.

Although an xApp does not correspond to an independent node classified as a physical component, it is a functional element that performs RAN control and optimization as a core application executing on the Near-RT RIC [5,6]. Accordingly, considering its significance in the RAN intelligence architecture, a separate definition for the xApp has been added to Table 1.

As summarized above, the major components of O-RAN are defined and interconnected through standardized interfaces to ensure interoperability. The Table 2 presents the definitions of key O-RAN interfaces, including E2, A1, O1, and O2, which play essential roles within the architecture [27,29].

Table 2: Definitions of O-RAN interfaces.

Term	Definition
E2	Interface for near-real-time control and monitoring of RAN elements by the Near-RT RIC.
A1	Interface for delivering high-level policies and AI models from the Non-RT RIC to the Near-RT RIC.
O1	Interface between the SMO and O-RAN elements for operation and management (OAM) functions, including FCAPS.
O2	Interface for the SMO to manage and orchestrate O-Cloud infrastructure resources.

2.2 Related Work

- **O-RAN Architecture and xApp Development Foundation Research**

  Polese et al. [1] systematized the O-RAN architecture by detailing the disaggregation of CU/DU/RU, O-Cloud virtualization, open interfaces, and intelligent closed-loop control. They summarized the roles of the Near-RT and Non-RT RICs alongside the 6-step AI/ML workflow, providing a foundational reference for subsequent O-RAN research.

  Elyasi et al. [30] presented a comprehensive survey specifically focused on Near-RT RIC xApps, addressing the lack of in-depth application analysis in previous literature. They summarized AI/ML workflows based on various learning paradigms and identified unresolved challenges, including adversarial attacks, real-time constraints, and policy conflicts.

  Feraudo et al. [31] proposed the xDevSM framework to resolve xApp development difficulties caused by E2SM version incompatibilities and ASN.1 serialization complexity. By abstracting the encoding processes, they demonstrated multi-vendor interoperability with a mere 1%–2% measurement error, significantly improving xApp development productivity.

  Groen et al. [10] quantitatively analyzed the performance overhead of applying encryption to O-RAN open interfaces. They found that while IPsec on the E2 interface adds a minimal latency of 22 µs, MACsec on the Open Fronthaul can induce up to 218 µs, which may exceed requirements in certain RU/DU combinations.

  Yu et al. [32] designed a monitoring framework to continuously verify O-RU configuration states, addressing the limitations of existing zero-trust architectures that fail to evaluate post-authentication behaviors. Using a TPM-based Trusted Reporter, they achieved policy violation detection within 200 ms, though the framework cannot capture behaviors outside the defined monitoring scope.

  Furthermore, building upon the fundamental O-RAN architecture and its open-source implementations [27,29], the O-RAN Alliance has systematized comprehensive threat models based on the STRIDE framework [11]. This integration of architectural standards and rigorous risk assessment establishes a solid foundation for O-RAN security verification across various components.

- **O-RAN Anomaly Detection and Attack Response Research**

  Başaran et al. [33] proposed a semi-supervised Deep Autoencoder (DeepAE) as an alternative to the Isolation Forest for anomaly detection in the Near-RT RIC. Training on an imbalanced UE dataset with a minority of labels, they recorded an overall accuracy of 93%, although real-time deployment performance in an actual O-RAN testbed was not verified.

  Basaran and Dressler [34] introduced XAInomaly, combining a Semi-Supervised Deep Convolutional Autoencoder (SS-DeepCAE) with fastSHAP-C to resolve overfitting and lack of explainability. They achieved an 80.17% Unweighted Average Recall (UAR) using only 100 labels, but robustness against adversarial attacks remains a future task.

  Dimou and Noubir [35] proposed the ARGOS framework to detect Rogue Base Station downgrade attacks by applying a Variational AE (VAE) to Modem Layer 1 logs. While it demonstrated 99.5% accuracy and a 0.6% false positive rate, it is structurally limited by its reliance on UE-side data and requires expansion to diverse attack types.

  Alimohammadi et al. [36] defined a KPI poisoning attack that manipulates reported values and developed an LSTM-based xApp architecture to detect it. They achieved a 99% detection rate with a processing time of 40–50 ms, remaining well within Near-RT constraints, though sophisticated attacks with low amplification still pose challenges.

Kakani et al. [37] implemented adversarial attacks on KPIs stored in Redis within an actual O-RAN testbed, showing that DNN-based detection accuracy plummeted to 10%. By proposing a comprehensive ensemble defense xApp, they successfully recovered the accuracy to 93% while maintaining a total latency of 76 ms.

Lee and Kim [38] applied deep learning models for malicious traffic detection within O-RAN, validating detection performance using the AI/ML workflow. However, their study did not address deployment-level verification under Near-RT RIC latency constraints or scalability analysis beyond the training environment.

- **O-RAN Intelligent Control Applications and Operational Stabilization Research**

Yungaicela-Naula et al. [39] analyzed misconfiguration issues arising from AI/ML integration in O-RAN and categorized them into three main pillars. They demonstrated how policy conflicts between xApps induce instability like ping-pong handovers, highlighting misconfiguration as a critical issue despite the lack of testbed verification.

Giannopoulos et al. [40] proposed the COMIX framework to resolve conflicts among multiple xApps by combining standard conflict mitigation with a Network Digital Twin (NDT). Simulating deep reinforcement learning-based xApps, they showed a 60% reduction in power consumption with minimal throughput variation, though the centralized bottleneck remains a limitation.

Lacava et al. [41] addressed O-RAN's inability to support sub-10ms real-time control by proposing a lightweight dApp architecture deployed directly in the CU/DU. Operating via a novel E3 interface, they achieved an average control latency of under 450 μs, though multi-vendor verification and multi-dApp conflict resolution remain future tasks.

Moore et al. [42] developed SliceX, a security slicing xApp designed to counter flooding attacks by monitoring KPMs and isolating malicious UEs into resourceless slices. This approach suppressed the maximum latency during an attack from over 7 s to approximately 2.3 s, but it was limited to single-attack scenarios and simple thresholds.

Abdalla et al. [15] proposed ZTRAN, an integrated security framework applying zero-trust principles through a hierarchical structure of authentication, detection, and slicing xApps. It effectively reduced the abnormal latency ratio caused by flooding attacks from 50% to 6.6%, though it relied on static KPM thresholds and lacked inter-xApp conflict management.

Moore et al. [43] implemented LLM-ID, a Large Language Model-based intrusion detection pipeline where an xApp analyzes real-time KPM data to identify threats. Utilizing a fine-tuned Gemma 2 model, they achieved 99% accuracy with an average response time of 239.66 ms, although generalization across diverse attack types and xApp environments remains unverified.

Table 3 presents a quantitative comparison with recent O-RAN security frameworks, ZTRAN [15] and ARGOS [35]. Unlike existing studies that focus on single-layer defense or small-scale tests, our proposed IDS xApp establishes a multi-layer (O-CU/O-DU) defense. Specifically, when deployed in the actual Near-RT RIC environment, the structurally optimized LSTM model recorded detection F1-scores of 94.87% for O-CU and 99.49% for O-DU. Furthermore, this study uniquely addresses critical operational gaps by performing a deployment performance gap analysis (ΔF1) and executing a massive-scale stress test of up to 80,000 instances to guarantee strict latency constraints.

Existing studies have provided a crucial foundation in terms of O-RAN architecture and xApp development, anomaly detection and attack response, as well as intelligent control applications and operational stabilization. However, research that quantitatively verifies detection performance and inference latency characteristics based on actual measurements—specifically targeting an IDS xApp deployed in a real Near-RT RIC environment and simultaneously utilizing O-CU network layer data and O-DU radio telemetry

data—remains limited. Against this background, this study aims to fill this gap by quantitatively analyzing the detection performance and inference latency using O-CU and O-DU data after deploying an IDS xApp in an O-RAN Near-RT RIC environment. The next section details the configuration and operational procedures of the proposed system.

Table 3: Essential quantitative comparison of the proposed IDS xApp with existing O-RAN security frameworks.

Core Criterion	ZTRAN [15]	ARGOS [35]	Proposed (Ours)
Target Scope	User Plane	Control Plane (E2)	**Multi-layer (O-CU & O-DU)**
Detection (F1)	Not reported	98.1%	**94.87% (O-CU), 99.49% (O-DU)**
Deploy Gap Analysis	Not addressed	Not addressed	**ΔF1/ΔPrec evaluated per model**
Scalability Test	4 UEs (Small-scale)	Not reported	**Stress tested up to 80K instances**
Latency Analysis	Network-level only	Inference time only	**Pareto trade-off (F1 vs. Latency)**
Resource Profiling	Not reported	Single model metric	**CPU/RAM/Time compared per model**

3 System Architecture and Workflow

This section describes the overall system architecture and operational workflow of the AI-based IDS xApp, designed based on the OSC architecture. To detect security threats occurring in the O-RAN environment in near-real-time, the proposed system is explained in detail from two perspectives. First, Section 3.1 analyzes the paths through which cyber attacks infiltrate the OSC system and explains the overall detection mechanism by which the IDS xApp identifies attack threats utilizing O-CU and O-DU layer attack data. Following this, Section 3.2 presents the system architecture that combines the AI/ML Framework of the SMO domain and the inference platform of the Near-RT RIC domain to implement this detection logic in an actual environment. Accordingly, the training path utilizing large-scale datasets and the near-real-time inference path through model deployment are defined, respectively.

3.1 O-RAN Threat Modeling and Architecture of IDS xApp

This subsection describes the design architecture of the IDS xApp, considering the overall O-RAN threat environment [9]. By adopting open interfaces and a cloud-based virtualization environment, the O-RAN architecture possesses a significantly expanded attack surface compared to traditional RANs [19,21]. According to the threat modeling of the O-RAN Alliance Working Group 11 (WG11), various threats spanning the entire layer are identified [11]. These include orchestration-level attacks targeting the SMO and Non-RT RIC, policy tampering via the A1 interface, xApp-induced malfunctions through the E2 interface [12], supply chain attacks aiming at the O-Cloud infrastructure, and time synchronization attacks on the Open Fronthaul interface. Notably, approximately 60% of the total threats defined by WG11 are directly linked to DoS and performance degradation, which directly impact the availability and Quality of Service (QoS) of O-RAN [11].

Amidst this comprehensive threat landscape, the O-CU and O-DU are the core layers where user traffic and control signals are practically processed [4,27], making them the points where the impact of attacks manifests most directly. Cyber attack traffic originating externally, such as DoS, Distributed Denial of Service(DDoS), Probe, Brute Force, and Web Attacks, inevitably passes through the upper-layer network protocol stack of the O-CU. Consequently, the resulting anomalies are also cascaded and reflected in the radio access layer telemetry data of the O-DU. Therefore, while premised on the threat model across the entire O-Cloud, this study designs the IDS xApp focusing on data collectible from the O-CU/DU layers, considering detection efficiency and the feasibility of a near-real-time response [14].

The proposed IDS xApp consists of a training path and an inference path, as illustrated in Fig. 2. In the training path, data collected from the O-CU and O-DU is delivered to the AI/ML Framework in the SMO domain [5]. Based on this, the AI Training Host Platform (ATHP) trains the anomaly detection model and deploys the fully trained model to the AI Inference Host Platform (AIHP) of the Near-RT RIC. In the inference path, the IDS xApp within the Near-RT RIC environment performs anomaly detection on incoming data in near-real-time via the KServe inference service within the AIHP to determine the presence of an attack [29,30].

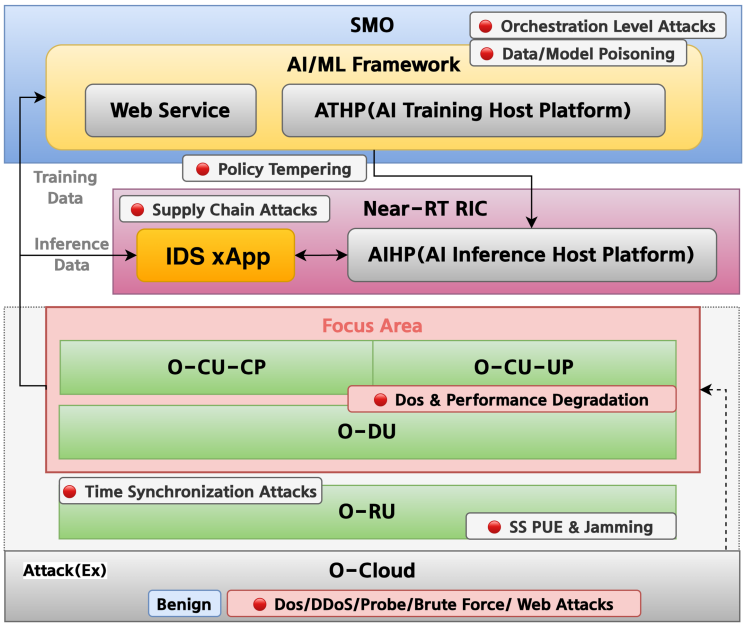

Figure 2: Architecture of the proposed IDS xApp for anomaly detection on O-CU and O-DU layer attack data based on O-RAN threat modeling.

Given this systemic background, the IDS xApp normalizes the collected heterogeneous data and analyzes it in near-real-time to determine the presence of an attack. The detailed steps of the anomaly detection logic based on multi-layer data, which constitutes the core of this architecture, are outlined in Algorithm 1 below.

Algorithm 1: IDS xApp inference procedure

Input Telemetry data X from O-CU/O-DU, model type $M \in \{\text{LSTM, CNN, Transformer, Autoencoder}\}$, threshold τ

Output Classification result $\hat{y} \in \{0\ (\text{Benign}), 1\ (\text{Attack})\}$

Load pre-trained model M from KServe (LeoFS)

Receive input telemetry X from O-CU or O-DU

Normalize X via Min-Max scaling:

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Reshape input according to model type M:

$$X_{input} = \begin{cases} (N, 1, F) & \text{if } M \in \{\text{LSTM, CNN, Transformer}\} \\ (N, F) & \text{if } M = \text{Autoencoder} \end{cases}$$

Send REST API request to KServe with X_{input}

Receive response from KServe

Classify according to model type M:

if $M \in \{\text{LSTM, CNN, Transformer}\}$ **then**

$$\hat{p}_i = \text{response}[i] \quad \text{(softmax probability)}$$

$$\hat{y}_i = \begin{cases} 1\ (\text{Attack}) & \text{if } \hat{p}_i \geq 0.5 \\ 0\ (\text{Benign}) & \text{if } \hat{p}_i < 0.5 \end{cases}$$

else if M = Autoencoder **then**

$$e_i = \frac{1}{F} \sum_{f=1}^{F} (X_i^f - \hat{X}_i^f)^2 \quad \text{(reconstruction error)}$$

$$\hat{y}_i = \begin{cases} 1\ (\text{Attack}) & \text{if } e_i \geq \tau \\ 0\ (\text{Benign}) & \text{if } e_i < \tau \end{cases}$$

end if

Return $\hat{y}_i$ to IDS xApp

3.2 System Overview and Implementation Flow

This subsection explains the system architecture and the training and inference workflows for operating the IDS xApp in the O-RAN Near-RT RIC environment. Fig. 3 illustrates the IDS xApp system architecture utilizing the AI/ML Framework. The proposed system environment is divided into the SMO domain, which performs model training based on large-scale data, and the Near-RT RIC domain, which deploys the trained model to perform near-real-time anomaly detection.

The SMO domain houses the AI/ML Framework dedicated to large-scale telemetry data collection and AI model training. The Web Service within the framework provides a Jupyter Notebook environment for model development and data analysis. In this study, the preprocessed O-CU and O-DU data are stored in the Feature Storage Database (Cassandra), after which the ML-Pipeline on the ATHP loads it to perform training. Therefore, the SMO DB (InfluxDB) included in Fig. 3 is an element that reflects the overall system configuration and is not used in the actual training path of this study. After training is complete, the final model artifacts are stored in the Model Storage (LeoFS) for deployment to the near-real-time inference environment.

The Near-RT RIC domain houses the AIHP-based IDS xApp and the KServe Inference Service, which perform inference operations. The IDS xApp dynamically loads the fully trained model stored in the Model Storage (LeoFS) via KServe. Based on this, it classifies whether the corresponding traffic is normal or an attack through the KServe inference engine and returns the result. Detailed procedures are explained in Sections 3.2.1 and 3.2.2.

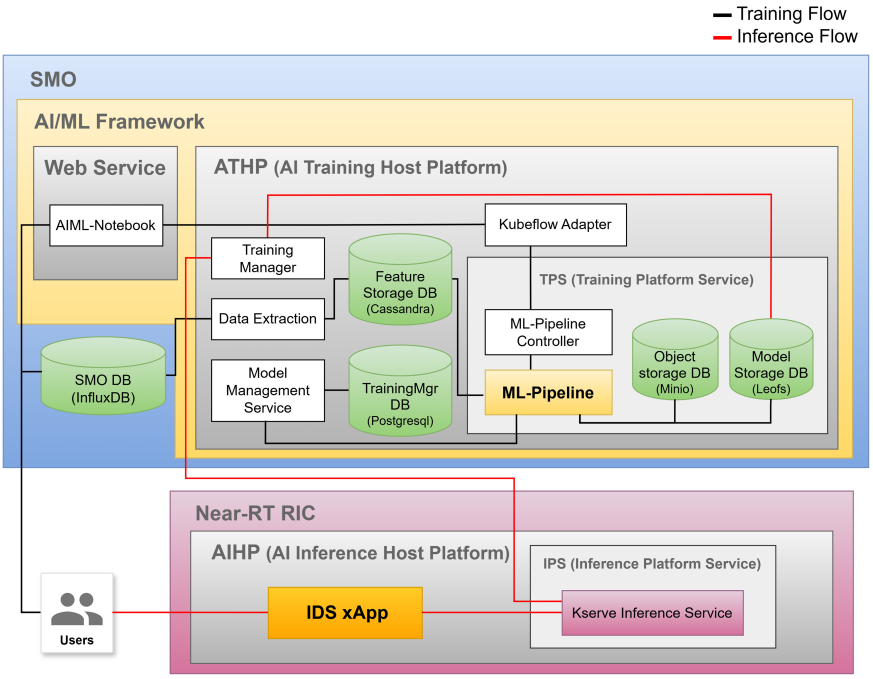

Figure 3: IDS xApp system architecture utilizing the AI/ML framework.

3.2.1 Training Workflow

The training path is a process executed within the AI/ML Framework of the SMO domain. The overall flow, illustrated in Fig. 4, is broadly divided into the creation and execution of the training pipeline and the storage of the training results.

1. **Training Model Pipeline Creation and Execution**

   A user creates a pipeline specification through the AI/ML-Notebook. Subsequently, a pipeline creation request is sent to the Training Manager. The ML-Pipeline Controller receives this request and loads the pre-refined O-CU and O-DU training data from the Feature Storage (Cassandra). Once the data loading is complete, the pipeline is executed to perform the specified model training.

2. Training Result Storage and Update

Upon completion of the training, the completion information is updated in the model management service. The pipeline input and output data generated during the training process are stored in the Object Storage (MinIO). The final trained model is stored as an artifact in a compressed format (model.zip) within the Model Storage (LeoFS). The stored model artifact is then deployed to the Near-RT RIC domain without retraining.

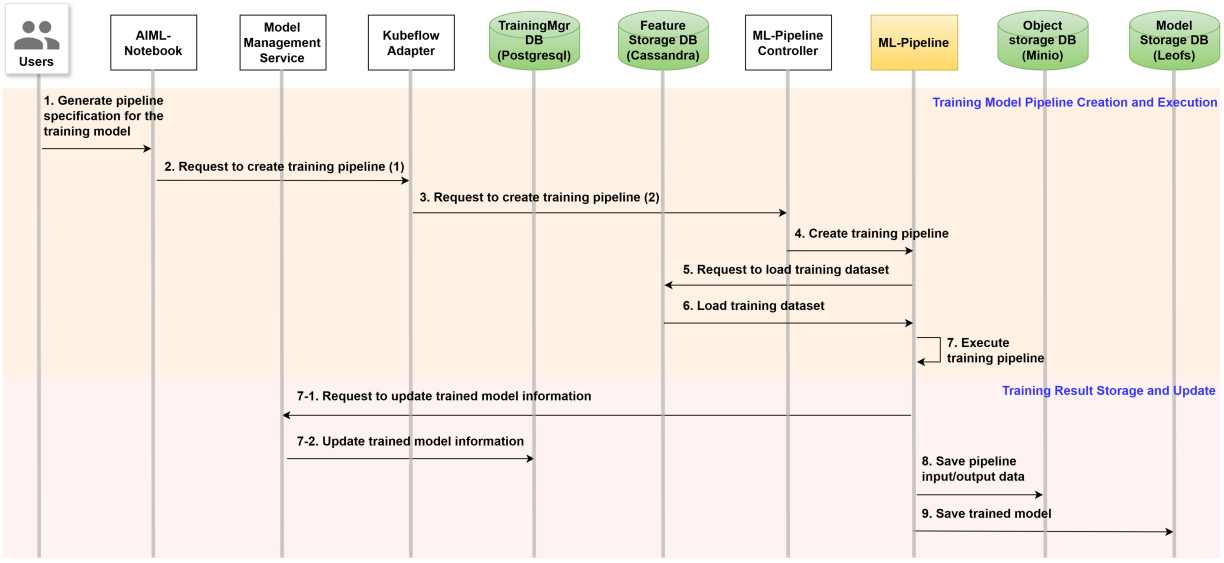

Figure 4: Flow of model training pipeline creation, execution, and result storage performed in the AI/ML framework.

3.2.2 Inference Workflow

The inference path is the process where the IDS xApp classifies whether actual traffic is anomalous within the Near-RT RIC domain. This workflow, as shown in Fig. 5, is primarily composed of the prediction model deployment phase and the prediction execution phase.

1. Prediction Model Deployment

First, an IDS xApp is created within the Near-RT RIC domain. Subsequently, a KServe-based prediction service is instantiated. The prediction service requests the loading of the fully trained model (.zip) stored in the Model Storage (LeoFS) of the SMO domain. Once the model loading is complete, KServe enters a standby state ready for inference.

2. Model Inference Execution

The anomaly detection input/output (I/O) process, which constitutes the core of this system, is performed as follows. In the input phase, the IDS xApp receives the network layer data of the O-CU and the radio Key Performance Measurement (KPM) telemetry data of the O-DU. Based on this data, it sends a REST API-based prediction request to KServe, and KServe performs the classification operation using the pre-loaded model. Upon completion of the computation, it returns the final result—determining whether the traffic is normal or an attack—back to the IDS xApp.

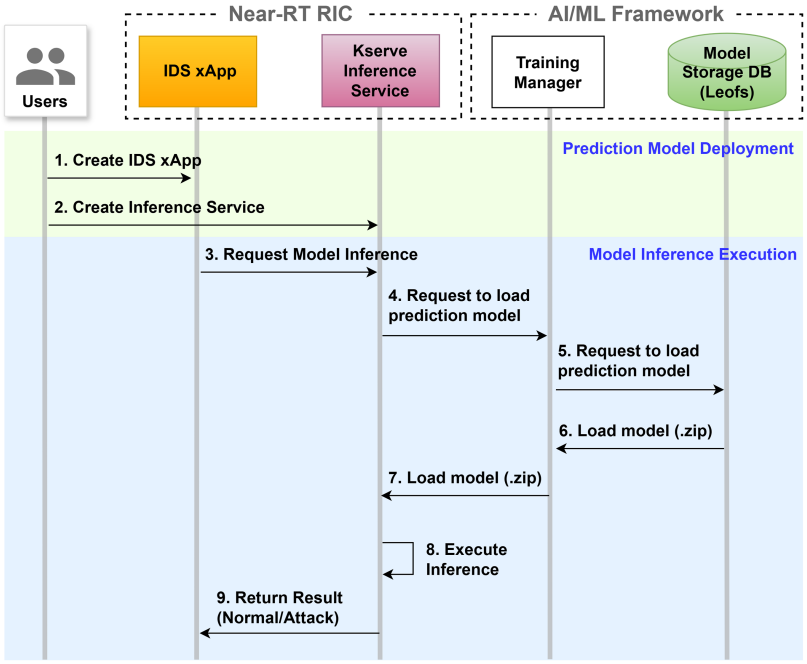

Figure 5: Prediction model deployment and inference execution phases via the IDS xApp.

4 Experimental Setup

This section describes the experimental environment and detailed configurations for verifying the detection performance and latency characteristics of the proposed IDS xApp architecture. The execution environment of this system is divided and operated in two independent domains according to their functional purposes.

First, the AI/ML Framework in the SMO domain is dedicated to model training. Second, the Near-RT RIC domain, where the KServe Inference Service and IDS xApp are deployed, performs the actual near-real-time inference operations of the model deployed from the SMO. This experiment measures the preservation of detection performance and the KServe-based inference latency by integrating these two environments.

The SMO environment was constructed on a high-performance workstation to rapidly process large-scale data preprocessing and model training. Ubuntu 24.04.3 LTS was used as the operating system, and the AI/ML Framework L-release of the O-RAN Alliance was applied as the software stack. The hardware infrastructure consisted of a 24-core x86 processor, 192 GB of Random Access Memory (RAM), and a 1 TB Non-Volatile Memory Express (NVMe) Solid State Drive (SSD).

The Near-RT RIC environment was constructed in a Virtual Machine (VM) environment to practically measure the KServe-based inference latency. An Ubuntu environment was used as the operating system, and the system was configured based on the Near-RT RIC J-release specifications of the OSC. The hardware for this system utilized a 16-core x86 processor as the host, running the IDS xApp and KServe inference engine within constrained computing resources allocated with 6 virtual CPUs (vCPUs).

4.1 Dataset-Driven Attack Modeling

This study uses a public dataset, the 'NetsLab 5G O-RAN IDD Dataset', to verify the detection performance of the proposed IDS xApp [44]. This subsection summarizes the attack threat assumptions, attack categories, and tools used during the dataset creation process, and outlines how the network layer data

of the O-CU and the radio KPM data of the O-DU were collected. The overall workflow of this dataset-driven attack modeling and dual-layer data collection is illustrated in Fig. 6.

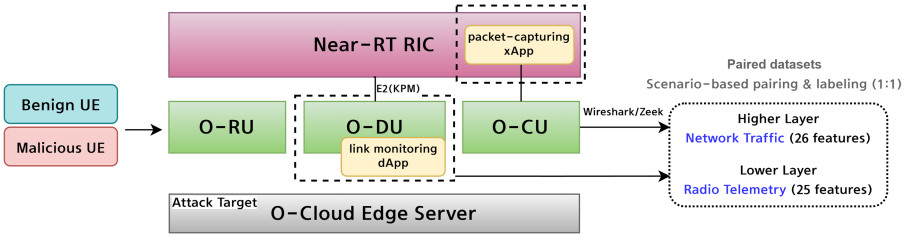

Figure 6: Dataset-driven attack modeling and dual-layer data collection workflow.

The corresponding dataset was generated in an Open Air Interface (OAI)-based 5G O-RAN testbed. It consists of a 5G Core Network (CN5G), an O-CU, two O-DUs, O-RUs (a vendor-grade RU and a Universal Software Radio Peripheral (USRP)-based RU), and a high-performance commercial off-the-shelf (COTS) edge server deployed at the network edge (co-located with the Near-RT RIC, emulating the O-Cloud). Normal traffic is generated by configuring an actual User Equipment (UE) with a laptop (5G Quectel RM500Q-GL modem), and attack scenarios are executed targeting the O-Cloud edge server.

The dataset includes normal traffic (e.g., YouTube streaming, file downloads, web browsing, iPerf traffic) and attack traffic. The dataset paper presents experimental scenarios including Internet Control Message Protocol (ICMP) Flood, Synchronize (SYN) Flood, Transmission Control Protocol (TCP) Flood, User Datagram Protocol (UDP) Flood, and UDP Port Scan as representative attack scenarios. Furthermore, based on the public repository, the data is composed of six categories: Benign, DoS, DDoS, Probe, Brute Force, and Web Attacks. Each category folder provides both Network Layer (Network Traffic (PCAP) files) and Lower Layer (Radio Telemetry files).

Data collection is performed simultaneously in both layers during the same scenario, ensuring that the upper-layer data and lower-layer data correspond 1:1 for each attack scenario. During the same scenario interval, the upper-layer data is generated by a packet-capturing xApp in the Near-RT RIC that collects packets, which are then processed using Wireshark and Zeek (e.g., conn.log) to be converted into flow- and session-level features. The data generated in this process is represented as ”Higher Layer Network Traffic”, comprising a total of 26 flow-based features. The lower-layer data is constructed by extracting E2 interface-based radio telemetry (KPM) from the O-DU using a link monitoring dApp. This data consists of radio quality and resource status indicators of the Physical (PHY) and MAC layers, organized as “Lower Layer Radio Telemetry” with a total of 25 features. The detailed list of features for both the O-CU and O-DU layers is summarized in Table 4.

Table 4: List of features by data type in the *NetsLab 5G O-RAN IDD* dataset.

O-CU		O-DU	
Feature	**Meaning**	**Feature**	**Meaning**
uid	Unique Identifier	dlBytes	Downlink Bytes
src_ip	Source IP Address	dlMcs	Downlink Modulation and Coding Scheme
src_port	Source Port	dlBler	Downlink Block Error Rate
dst_ip	Destination IP Address	ulBytes	Uplink Bytes

(Continued)

Table 4 (continued)

O-CU		O-DU	
Feature	**Meaning**	**Feature**	**Meaning**
dst_port	Destination Port	ulMcs	Uplink Modulation and Coding Scheme
proto	Protocol	ulBler	Uplink Block Error Rate
service	Application Service	ri	Rank Indicator
duration	Connection Duration	phr	Power Headroom Report
src_bytes	Source Bytes	pcmax	Maximum Power Capability
dst_bytes	Destination Bytes	rsrq	Reference Signal Received Quality
conn_state	Connection State	sinr	Signal to Interference plus Noise Ratio
missed_bytes	Missed Bytes	rsrp	Reference Signal Received Power
history	Packet History	rssi	Received Signal Strength Indicator
src_pkts	Source Packets	cqi	Channel Quality Indicator
src_ip_bytes	Source IP Bytes	pucchSnr	Physical Uplink Control Channel SNR
dst_pkts	Destination Packets	puschSnr	Physical Uplink Shared Channel SNR
dst_ip_bytes	Destination IP Bytes	ue_id	User Equipment Identifier
ip_proto	IP Protocol	timestamp	Data Collection Time
http_trans_depth	HTTP Transaction Depth	cellid	Cell Identifier
attack_category	Attack Category	in_sync	UE Synchronization Status
attack_type	Attack Type	rnti	Radio Network Temporary Identifier
files_total_bytes	Total File Bytes	pmi	Precoding Matrix Indicator
is_GET_mthd	HTTP GET Method	traffic_type	Traffic Type
http_status_error	HTTP Status Error	attack_category	Attack Category
is_file_transferred	File Transferred	attack_subcategory	Attack Subcategory
traffic_type	Traffic Type		

4.2 Dataset Preprocessing and Model Configuration

This subsection describes the preprocessing procedures and model configuration settings applied to utilize the *NetsLab 5G O-RAN IDD* dataset for training and evaluation in this study. The data preprocessing process is performed step-by-step according to the procedure presented in Fig. 7. Subsequently, the input formats and configurations were standardized so that each model could be trained and inferred under identical evaluation conditions.

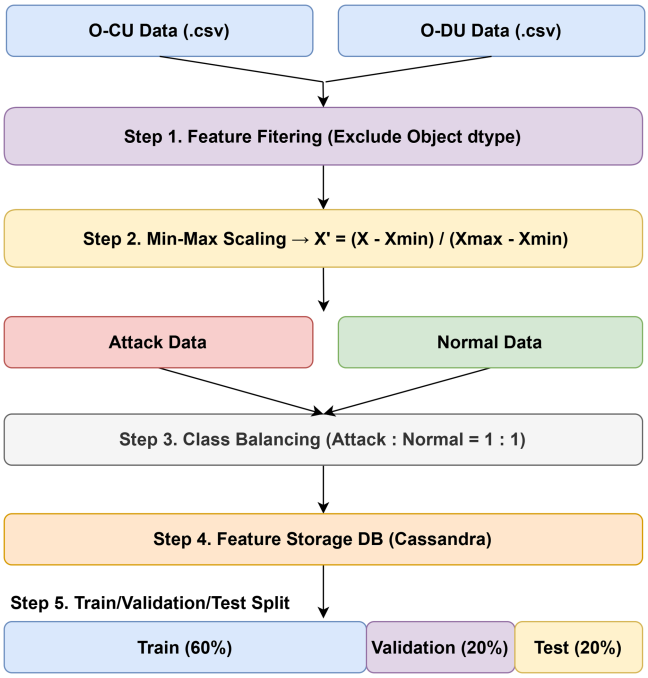

Figure 7: O-CU/DU attack dataset preprocessing workflow.

First, non-numeric meta-information (Object dtype) that is not directly utilized for model training was excluded from the network traffic and KPM data (.csv), and the columns necessary for training were configured. This exclusion strategy was deliberately adopted to avoid the significant computational overhead associated with embedding non-numeric features, thereby ensuring strict compliance with the ultra-low latency constraints (<1000 ms) of the Near-RT RIC environment. For the O-CU data, a total of 17 columns were used; among these, traffic type was defined as the label column indicating normal/attack types, and the remaining 16 numeric columns were configured as input features (X). For the O-DU data, a total of 21 columns including traffic type were used; traffic type was separated as the label, and the remaining 20 numeric radio performance and channel quality-related indicators were configured as input features (X). A subsequent permutation feature importance analysis confirmed that these retained numeric features provide highly discriminative power for anomaly detection, with transport-layer behaviors (e.g., duration) in O-CU and temporal patterns (e.g., timestamp) along with traffic volume metrics (e.g., ulBytes, dlBytes) in O-DU primarily driving the models' high F1-scores.

Subsequently, to prevent training bias due to differences in value ranges, Min-Max normalization was applied to the input features to convert them into the same scale range. Following this, the ratio of normal to attack data was adjusted to 1:1 to resolve the class imbalance problem that anomaly detection models might experience. The balanced data refined in this manner is loaded into the Feature Storage DB (Cassandra) within the SMO domain to be utilized as training resources for the ML-Pipeline.

Crucially, the inference mechanism of the proposed IDS xApp is designed on a per-record basis rather than utilizing a sliding window of historical data. This approach is directly reflected in the input shapes for the deep learning models, which are configured as $(1, 16)$ for O-CU data and $(1, 20)$ for O-DU data, where the sequence length is fixed at 1. Specifically, for the O-CU dataset, each instance represents a single network flow record containing connection-level features (e.g., duration, packet counts). For the O-DU dataset, it corresponds to a single UE telemetry report at a specific timestamp, incorporating radio-layer metrics such

as Reference Signal Received Power(RSRP) and Signal to Interference plus Noise Ratio(SINR). This bufferless design allows the IDS xApp to perform immediate classification as soon as a record arrives, effectively eliminating the latency associated with historical data accumulation and ensuring strict compliance with the near-real-time constraints (≤1000 ms) of the Near-RT RIC environment.

In the training phase, the entire dataset was processed according to the workflow shown in Fig. 7, and then divided into training (60%), validation (20%), and testing (20%) ratios. Regarding the input formats across models, LSTM, CNN, and Transformer were reshaped and input as 3-dimensional tensors in the form of $(N, 1, F)$ with a sequence length of 1, while the Autoencoder utilized the 2-dimensional matrix form (N, F) as is for calculating the reconstruction error.

This study configured the experiment by selecting four representative deep learning models (LSTM, CNN, Transformer, Autoencoder) widely utilized in the fields of network intrusion detection and time-series anomaly detection [19,20]. To ensure a fair comparison, the hidden layers of all models uniformly utilized the ReLU activation function, ensuring that performance discrepancies stem from architectural characteristics rather than activation functions. The specific data flow and layer configurations for each model were meticulously designed. The LSTM consists of an input layer, an LSTM layer (15 units), a Dropout layer (0.25), and a Dense output layer (Sigmoid), yielding 1936 and 2176 trainable parameters for the O-CU and O-DU datasets, respectively. The CNN processes inputs through two Conv1D layers (64 and 32 filters, $k = 3$) alternating with Batch Normalization, followed by Global Average Pooling and a Dense output layer (8193 and 8961 parameters). The Transformer features a Multi-Head Attention block (4 heads, $key\_dim = 64$), followed by residual connections, Layer Normalization, and Dense layers, making it the most complex model with 20,513 and 25,401 parameters. The Autoencoder utilizes an encoder (Dense layers of 16 and 4 units) to extract latent features and a symmetric decoder to reconstruct the original data (984 and 1264 parameters).

In this study, these four models were collectively utilized as a comprehensive benchmark suite to evaluate the proposed IDS xApp. Specifically, as demonstrated in established network intrusion detection literature [45], LSTM is widely adopted as a standard benchmark due to its recurrent architecture, which is inherently suited for capturing the temporal dependencies and sequential patterns of network traffic flows. Building upon this reliable baseline, CNN, Transformer, and Autoencoder were incorporated to represent spatial feature extraction, high-load attention mechanisms, and unsupervised learning-based threshold discrimination, respectively [33]. By introducing this diverse set of benchmark models with distinctly different computational complexities, this study objectively evaluates the latency processing capability of the proposed KServe-based inference system and the performance preservation upon transitioning the execution environment [23,30].

For the fairness of the experiment, the trainable parameters of each model were kept identical in the deployment environment (Near-RT RIC) without any retraining or fine-tuning. However, the threshold setting for binary classification was handled meticulously to ensure an objective comparison between supervised and unsupervised learning paradigms. The supervised models (LSTM, CNN, Transformer) uniformly applied a fixed threshold of 0.5 to their sigmoid outputs across all environments. Conversely, the unsupervised Autoencoder determines anomalies based on a dynamic reconstruction error threshold. In the training environment (SMO), this threshold was rigorously set to the F1-optimal point on the Precision-Recall curve (0.0565 for O-CU, 0.0035 for O-DU). When transitioning to the deployment environment, real-time streaming constraints and execution environment transitions can cause subtle data distribution shifts [14]. To compensate for this and provide a fair evaluation condition, the Autoencoder's threshold was

not blindly carried over. Instead, it was recalibrated using a calibration subset of 5,000 samples in the Near-RT RIC, identifying the new environment-optimized thresholds (0.0550 for O-CU, 0.0150 for O-DU). All subsequent performance evaluations were conducted using these rigorously calibrated configurations.

4.3 Evaluation Metrics

To evaluate the anomaly detection classification performance of the models, the following four metrics were calculated based on the True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) of the Confusion Matrix [20]:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

Considering the characteristics of network intrusion detection data, the harmonic mean-based F1-score was utilized as the primary performance metric [19]. In addition, the Precision metric was analyzed concurrently to check the frequency of false positives, which mistake normal traffic for attacks in the deployment environment and cause interference with base station control functions [39].

Along with detection performance, Big-$\mathcal{O}$ notation, a time complexity metric, was introduced to mathematically analyze the inference latency characteristics that occur when the IDS xApp deployed in the Near-RT RIC environment processes large-scale traffic [41]. In this study, the number of concurrent prediction instances flowing into KServe is defined as N, and the corresponding round-trip inference latency is defined as $T_{latency}(N)$. For the proposed system to operate stably in an environment with increasing traffic, the latency must not explode exponentially (e.g., $\mathcal{O}(N^2)$) but must satisfy a linear upper bound as shown in the following equation:

$$T_{latency}(N) \leq c \cdot N \quad (\text{for all } N \geq N_0) \implies T_{latency}(N) \in \mathcal{O}(N) \tag{5}$$

Here, c and N_0 are positive constants. According to the O-RAN WG1 specifications, the 10 ms–1 s control loop accommodates diverse functional requirements. The lower bound (near 10 ms) is typically required for PHY/MAC-level real-time operations such as Radio Resource Management (RRM), beamforming optimization, and handover control. In contrast, the upper bound (near 1 s) is suitable for intelligence-based applications that allow more decision time but require a faster response than the Non-RT RIC, such as anomaly detection, traffic classification, and security policy enforcement. The proposed IDS xApp falls into the latter category, requiring a decision within 1000 ms to mitigate potential performance degradation by isolating malicious UEs before they impact the base station's stability. In the upcoming Section 5, $T_{latency}(N)$ is practically measured while incrementally increasing N up to a maximum of 80,000 instances. Through trend line analysis of the measured data, it is experimentally verified whether the KServe-based inference workflow guarantees a linear time complexity of $\mathcal{O}(N)$ and stably satisfies this near-real-time control requirement of less than 1 s (1000 ms) [1,27].

5 Experimental Results

This section presents three experimental results targeting the deployed IDS xApp. First, the anomaly detection performance on the O-CU and O-DU datasets is quantified and analyzed. Second, the inference latency and the time complexity $\mathcal{O}(N)$ according to the increase in prediction instances are verified. Third, the resource consumption required for running each trained model is quantitatively measured.

5.1 Detection Performance Analysis of IDS xApp

Table 5 compares the performance differences between the training environment (AI/ML Framework) and the actual deployment environment (Near-RT RIC) within the proposed IDS xApp architecture, categorized by model and data type. Accuracy, Precision, Recall, and F1-score were used as evaluation metrics. To quantitatively verify the performance preservation upon transitioning to the deployment environment, the variation in F1-score (ΔF1) and Precision (ΔPrec) are presented together.

Table 5: Classification performance comparison between AI/ML framework and Near-RT RIC deployment environments on O-CU and O-DU attack datasets.

Data Type	Model	Env	Acc	Prec	Rec	F1	ΔF1	ΔPrec
O-CU	LSTM	AI/ML	0.9486	0.9449	0.9526	0.9488	–	–
		RIC	0.9486	0.9455	0.9520	**0.9487**	−0.0001	+0.0006
	CNN	AI/ML	0.9585	0.9546	0.9629	0.9587	–	–
		RIC	0.9570	0.9541	0.9601	**0.9571**	**−0.0016**	−0.0005
	Transformer	AI/ML	0.9641	0.9888	0.9388	0.9631	–	–
		RIC	0.9638	0.9898	0.9375	**0.9629**	−0.0002	+0.0010
	Autoencoder	AI/ML	0.8173	0.7347	0.9935	0.8447	–	–
		RIC	0.7897	0.7044	0.9970	**0.8256**	**−0.0191**	**−0.0303**
O-DU	LSTM	AI/ML	0.9955	0.9976	0.9934	0.9955	–	–
		RIC	0.9948	0.9969	0.9929	**0.9949**	−0.0006	−0.0007
	CNN	AI/ML	0.9970	0.9986	0.9953	0.9970	–	–
		RIC	0.9973	0.9993	0.9954	**0.9974**	+0.0004	+0.0007
	Transformer	AI/ML	0.9962	0.9986	0.9940	0.9962	–	–
		RIC	0.9966	0.9991	0.9942	**0.9966**	+0.0004	+0.0005
	Autoencoder	AI/ML	0.9646	0.9688	0.9601	0.9644	–	–
		RIC	0.9011	0.8477	0.9858	**0.9115**	**−0.0529**	**−0.1211**

Note: Bold values in the RIC environment represent the final classification performance after deployment. Bold values in the ΔF1 and ΔPrec columns indicate performance variations that exceed the predefined reference interval of ±0.001 (±0.1%), highlighting non-negligible performance changes as discussed in the text.

On the O-CU network layer attack data, supervised learning-based models, namely LSTM, CNN, and Transformer, maintained F1-scores of 0.9487, 0.9571, and 0.9629, respectively, even in the deployment environment. The variations before and after deployment were all negligible, with LSTM ΔF1 = −0.0001, CNN ΔF1 = −0.0016, and Transformer ΔF1 = −0.0002. In contrast, the Autoencoder showed relatively larger variations, with its F1-score decreasing from 0.8447 to 0.8256 (ΔF1 = −0.0191) and its Precision decreasing from 0.7347 to 0.7044 (ΔPrec = −0.0303).

On the O-DU radio telemetry attack data, the supervised learning models recorded even higher absolute performance. In the RIC environment, LSTM, CNN, and Transformer achieved F1-scores of 0.9949, 0.9974, and 0.9966, respectively, and the variations were extremely limited at −0.0006, +0.0004, and +0.0004 based on ΔF1. The Autoencoder exhibited a more significant performance degradation between the two datasets,

with its F1-score decreasing from 0.9644 to 0.9115 (ΔF1 = −0.0529) and its Precision decreasing from 0.9688 to 0.8477 (ΔPrec = −0.1211).

Fig. 8 establishes ±0.1% (±0.001) as the reference interval to evaluate the performance variation between the training and deployment environments. If the absolute values of ΔF1 and ΔPrec fall within this range, the performance degradation due to the transition to the deployment environment is considered negligible.

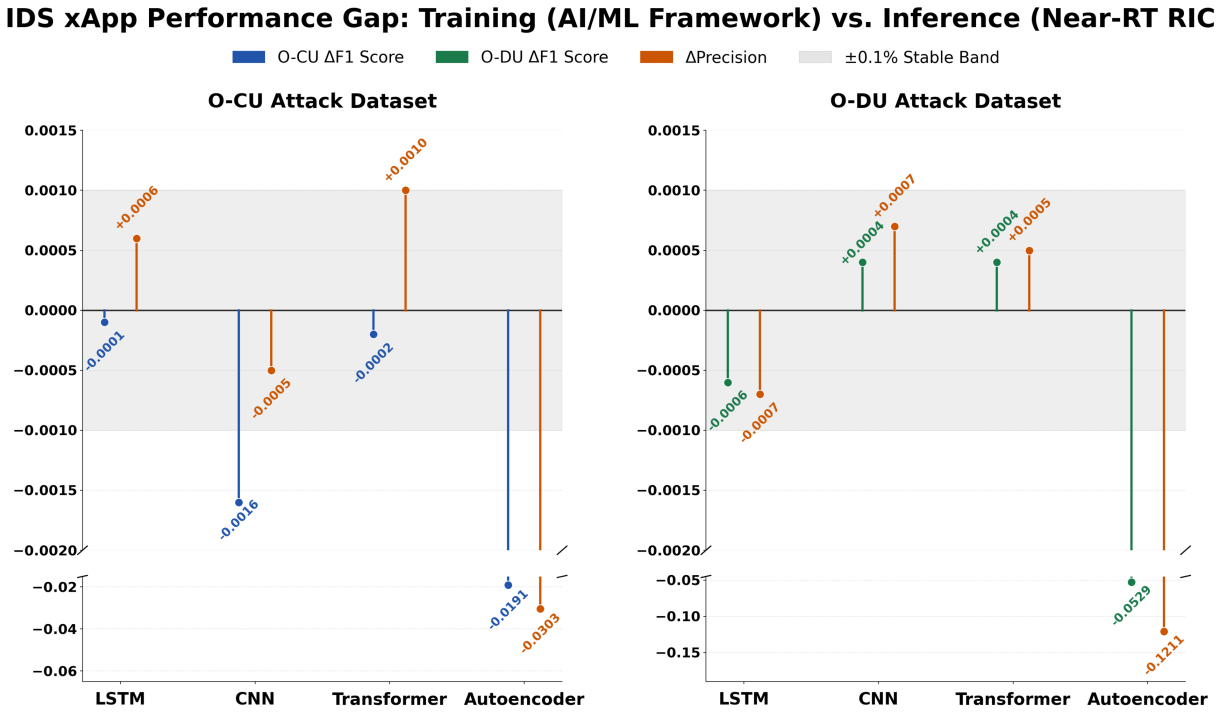

Figure 8: Comparison of ΔF1 and ΔPrecision in the Near-RT RIC IDS xApp (inference) environment compared to the AI/ML framework (training) for LSTM, CNN, transformer, and autoencoder models on the O-CU network traffic and O-DU radio telemetry attack datasets.

Across both datasets, the performance variation of supervised learning-based models before and after deployment was highly limited. For the O-CU network layer attack data, the LSTM recorded ΔF1 = −0.0001 and ΔPrec = +0.0006, while the Transformer recorded ΔF1 = −0.0002 and ΔPrec = +0.0010, both showing variations within the established reference interval. For the CNN, ΔPrec = −0.0005 was within the interval, but ΔF1 = −0.0016 slightly exceeded the threshold. In the O-DU radio telemetry attack data, the LSTM recorded ΔF1 = −0.0006 and ΔPrec = −0.0007, and the Transformer recorded ΔF1 = +0.0004 and ΔPrec = +0.0005, placing all metrics within the reference interval. The CNN also maintained equally stable performance with ΔF1 = +0.0004 and ΔPrec = +0.0007.

Based on the absolute values of the variations, the LSTM recorded |ΔF1| = 0.0001 and |ΔPrec| = 0.0006 for the O-CU attack dataset, and |ΔF1| = 0.0006 and |ΔPrec| = 0.0007 for the O-DU attack dataset, demonstrating the smallest fluctuation range among the compared models across both datasets. This indicates that the LSTM exhibits the most stable performance preservation upon transition from the training environment to the deployment environment [23,30].

The fractional variations ($|\Delta F1| \leq 0.0016$) observed in the supervised models fall within the standard deviation range of repeated inference runs, as illustrated by the error bars in Fig. 8. This indicates that the performance differences between the training and deployment environments are not statistically significant.

Furthermore, these results demonstrate that the detection capabilities are robustly preserved within the conservative operational tolerance of ±0.1 pp, validating the practical deployability of the models in the Near-RT RIC environment.

In contrast, the unsupervised learning-based Autoencoder exhibited notable performance degradation due to the transition to the deployment environment in both datasets. For the O-CU attack data, It recorded $\Delta F1 = -0.0191$ (1.91 pp) and $\Delta Prec = -0.0303$ (3.03 pp). For the O-DU attack data, the decline was even more pronounced at $\Delta F1 = -0.0529$ (5.29 pp) and $\Delta Prec = -0.1211$ (12.11 pp). This exceeds the set threshold of ±0.1 pp by up to 30 times for O-CU and 121 times for O-DU. This phenomenon is interpreted to be caused by the Autoencoder's use of a threshold discrimination method based on the reconstruction error distribution. If the data distribution changes in the deployment environment, the reconstruction error distribution may alter, and employing a fixed threshold in such cases can potentially increase false positives [14,39]. The drop in Precision observed in these results can be associated with this characteristic. In particular, the $\Delta Prec = -0.1211$ in the O-DU attack data indicates that this threshold sensitivity was strongly manifested in the radio telemetry attack data.

To further evaluate the robustness of the proposed IDS xApp, a granular performance analysis was conducted across five specific attack categories: DoS, DDoS, Probe, Brute Force, and Web Attack. As summarized in Table 6, the detection capabilities varied depending on the model architecture and the dataset tier.

Table 6: Detailed F1-score performance analysis of deep learning models across specific attack categories on O-CU and O-DU attack datasets.

Data Type	Attack Category	LSTM	CNN	Transformer	Autoencoder
O-CU	DoS	0.8502	0.8807	0.8410	0.8180
	DDoS	0.9415	0.9564	0.9629	0.8460
	Probe	0.6276	0.4480	0.0929	0.8485
	Brute Force	0.6384	0.6571	0.5599	0.8402
	Web Attack	0.9241	0.6714	0.9849	0.8468
O-DU	DoS	0.9909	0.9975	0.9981	0.8064
	DDoS	0.9589	0.9921	0.9919	0.7976
	Probe	0.9852	0.9976	0.9979	0.7944
	Brute Force	0.9898	0.9972	0.9984	0.8151
	Web Attack	0.9899	0.9973	0.9983	0.8146

In the O-CU dataset, while most models achieved high F1-scores for DoS and DDoS, the Transformer struggled significantly with Probe attacks, recording a low F1-score of 0.0929. In contrast, the LSTM and Autoencoder maintained relatively stable performance for the same threat, with F1-scores of 0.6276 and 0.8485, respectively. This suggests that the Transformer's complexity may lead to over-specialization in high-volume traffic patterns, failing to capture subtle scanning behaviors.

Interestingly, in the O-DU dataset, all supervised models (LSTM, CNN, Transformer) exhibited near-perfect detection performance across all five categories, consistently exceeding 0.98, with the exception of the LSTM's DDoS detection. This indicates that radio-level telemetry provides more distinct features for classifying diverse attack types compared to network-layer flows. Based on these results, the LSTM model was identified as the most reliable architecture, providing the most balanced and resilient detection across all specific threats without significant performance degradation.

5.2 Inference Latency Characteristics and Time Complexity Analysis

While the baseline detection performance in Section 5.1 was evaluated under standard operational loads, this subsection practically measures the inference latency $T_{latency}(N)$ of the IDS xApp deployed in KServe and analyzes the time complexity as the traffic load increases. To evaluate the upper bounds of scalability and analyze the system's behavior under heavy traffic loads, the processing time was measured while scaling the number of concurrent inference instances up to 80,000 for both the O-CU and O-DU attack datasets. This evaluation thoroughly covers the required load capacity of the Near-RT RIC environment, where the strict ultra-low latency threshold is defined as 1000 ms. For each model (LSTM, CNN, Transformer, Autoencoder), the number of inference instances was incrementally increased. All measurements were performed after 5 warm-up iterations, and the average value of 100 repeated executions per interval was utilized. The measurement metrics were Mean Latency, P95, and P99 latency, which were compared against the 10 and 1000 ms (1 s) reference baselines.

To accurately reflect the operational realism of the Near-RT RIC deployment, the inference latency reported in this section is defined as the end-to-end round-trip time. This measurement explicitly includes not only the pure algorithmic execution time (forward pass) of the deep learning models but also the overarching systemic overheads. Specifically, the measured latency encompasses the JavaScript Object Notation (JSON) serialization of telemetry data at the IDS xApp, the REST API (HTTP POST) communication overhead across the container network, the subsequent JSON deserialization at the KServe Inference Service, and the reverse process for returning the prediction results.

5.2.1 Inference Latency Analysis on O-CU and O-DU Attack Atasets

As a result of measuring the latency variation up to 80,000 instances, distinct latency behaviors emerged depending on the model architecture. Fig. 9a, b presents the inference latency trajectories on the O-CU and O-DU attack datasets, respectively.

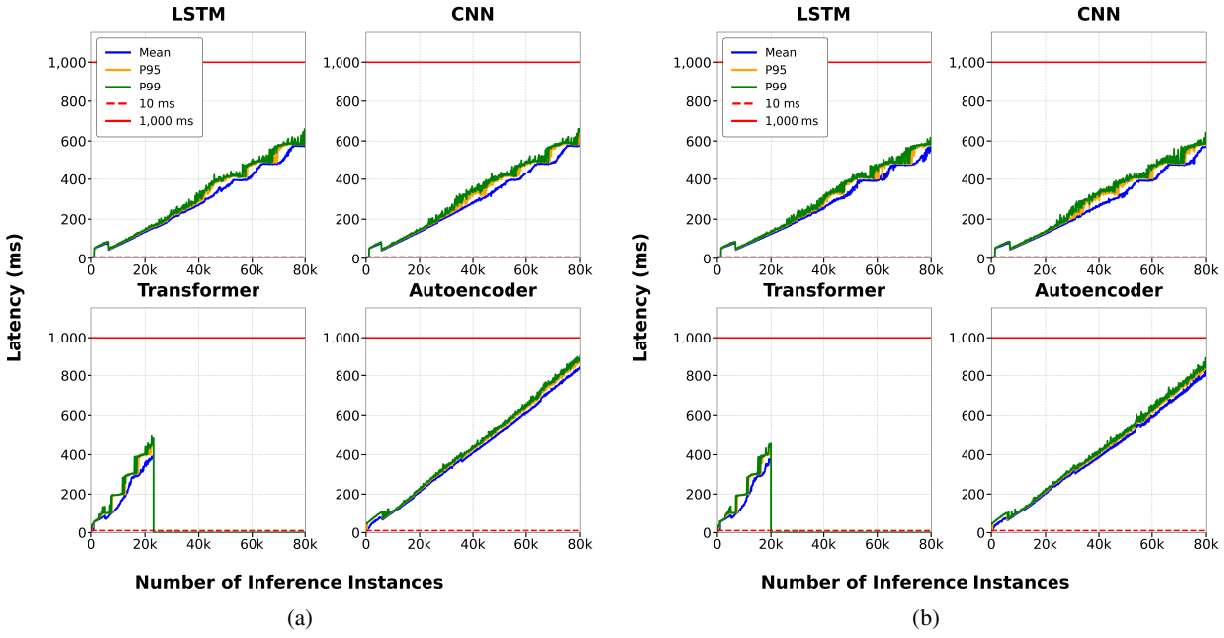

Figure 9: Inference latency graph of four models according to the number of prediction instances: (**a**) evaluation on the O-CU network-layer attack dataset, and (**b**) evaluation on the O-DU radio telemetry attack dataset.

As observed in both figures, the Transformer model's inference was terminated at approximately 20,000 instances, beyond which no further latency measurements could be obtained. This termination is fundamentally attributed to the complex Attention mechanism of the Transformer architecture, whose high computational overhead rapidly exhausted the memory and processing capacity of the KServe inference pod, causing the pod to fail under large-scale concurrent requests. Consequently, within the proposed KServe-based environment, the stable operational range for a Transformer-based xApp is practically limited to approximately 20,000 concurrent instances [41].

In contrast, structurally optimized models such as LSTM, CNN, and Autoencoder demonstrated exceptional scalability. Even under the extreme load of 80,000 instances, these models maintained a highly stable, linear increase in latency without experiencing exponential explosions. Specifically, the LSTM and CNN models recorded peak latencies of approximately 600 ms, while the Autoencoder reached around 850 ms. These results confirm that structurally efficient models can comfortably satisfy the strict 1000 ms Near-RT RIC requirement even under massive traffic surges. Notably, the LSTM and CNN maintained a very narrow deviation between P95 and P99 latencies compared to the Autoencoder, indicating superior tail latency stability.

5.2.2 Empirical Validation of Linear $\mathcal{O}(N)$ Complexity

To mathematically validate the system's scalability and assess potential non-linear latency degradation, an empirical linear complexity fitting was conducted within stable operational ranges. The theoretical basis for this analysis stems from the interaction between the system's processing logic and model-specific computational demands. The proposed IDS xApp is designed to process individual telemetry records independently on a per-record basis. In this framework, the total latency $T(N)$ for N instances ideally scales as the product of the unit processing time and N, resulting in a linear $\mathcal{O}(N)$ complexity. Conversely, quadratic degradation ($\mathcal{O}(N^2)$) occurs when computational demands grow non-linearly, such as in the Transformer's self-attention mechanism, which requires $N \times N$ matrix operations. Under constrained environments like the allocated 6 vCPUs, these demands lead to rapid resource saturation and subsequent inference termination.

Fig. 10 illustrates the comparison between measured latency data and these theoretical complexity curves.

As shown in the graph, the measured latency trajectories (blue solid lines) of the LSTM, CNN, and Autoencoder models closely aligned with the $\mathcal{O}(N)$ regression line (green solid line). Notably, these three models maintained their response times within the 1000 ms Near-RT RIC requirement even at a high load of 80,000 instances. They achieved high coefficients of determination ($R^2 > 0.987$) across both datasets, clearly deviating from the simulated quadratic $\mathcal{O}(N^2)$ trajectory (red dashed line).

For the Transformer model, inference was terminated at approximately 20,000 instances due to KServe pod resource exhaustion, beyond which no further measurements could be obtained. This is attributed to the system's physical resource limits being exceeded by the high computational overhead of the attention mechanism, rather than architectural flaws. Regression analysis focused on the stable operational range prior to this termination point revealed that the Transformer also maintains linear growth characteristics, with R^2 values of 0.9688 (O-CU) and 0.9600 (O-DU).

Consequently, these results demonstrate that the proposed KServe-based IDS xApp provides predictable $\mathcal{O}(N)$ linear scalability when integrated with computationally efficient architectures. While high-complexity models like the Transformer are constrained by physical resource limits, they still exhibit linear growth within their stable capacity bounds [30].

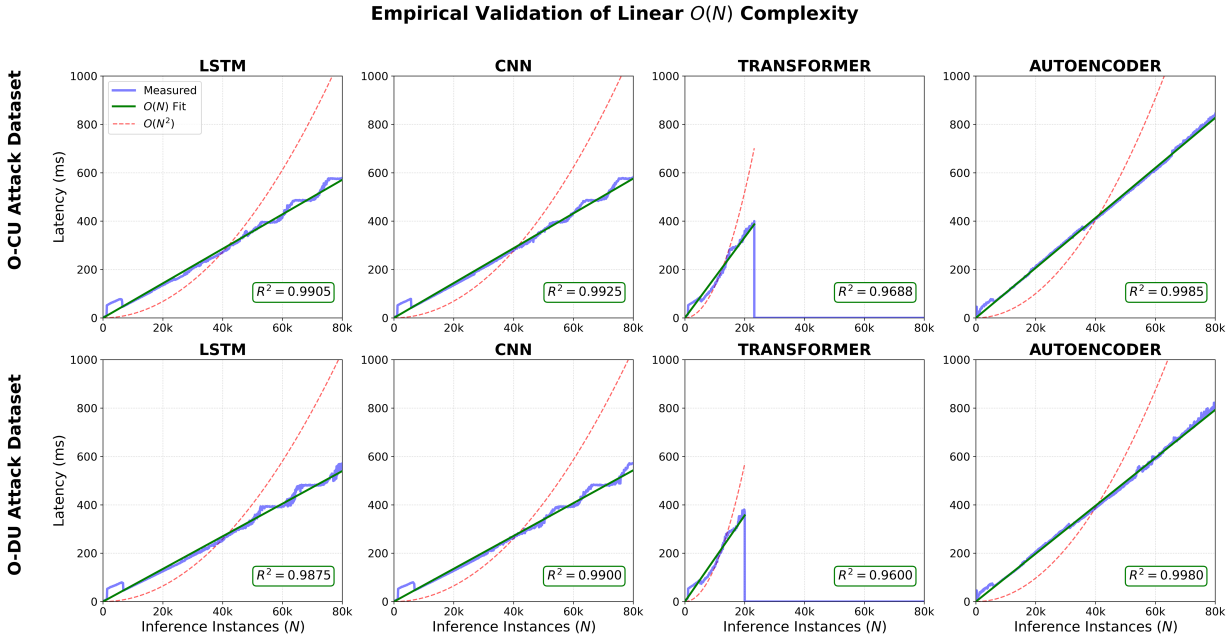

Figure 10: Empirical validation of linear $\mathcal{O}(N)$ complexity with R^2 goodness-of-fit for the evaluated models across datasets.

5.3 Resource Consumption in AI/ML Framework and Comprehensive Trade-Off Analysis

This section quantitatively analyzes the resource consumption of the model training process executed in the SMO domain to support the inference performance of the IDS xApp verified in Sections 5.1 and 5.2 from a system operation perspective. To this end, the CPU and RAM consumption (Wh) and the training time (s) incurred when training each model on the AI/ML Framework were measured. The detailed measurement results are presented in Table 7.

Table 7: Resource consumption (CPU, RAM, Training time) by model within the AI/ML framework on O-CU and O-DU attack datasets.

Data Type	Training Model	CPU (Wh)	RAM (Wh)	Training Time (s)
O-CU	LSTM	5.118	5.949	793.71
	CNN	9.360	11.435	1526.74
	Transformer	10.776	13.513	1806.14
	Autoencoder	0.927	1.428	190.60
O-DU	LSTM	0.187	0.289	37.81
	CNN	0.400	0.602	79.75
	Transformer	0.527	0.784	102.38
	Autoencoder	0.172	0.266	18.70

Based on the analysis of the O-CU attack dataset, the Transformer model, which had the highest inference latency, also consumed 10.776 Wh of CPU and 13.513 Wh of RAM during the training process, taking approximately 1806 s (about 30 min). This is the highest resource usage among the compared models. Conversely, the unsupervised learning-based Autoencoder recorded a relatively short training time of 190 s

and low resource consumption. However, as confirmed in the previous experiments, detection performance degradation and relatively high tail latency (P99) were observed in the actual deployment environment.

For the CNN and LSTM, which ranked relatively high in terms of detection performance and latency stability, differences emerged in resource consumption. The CNN required 1526 s of training time and 9.360 Wh of CPU resources, whereas the LSTM consumed 793 s of training time and 5.118 Wh of CPU resources. This indicates that the LSTM exhibited lower resource requirements in terms of training time and resource consumption compared to the CNN. In particular, considering that the LSTM maintained stable detection performance and latency characteristics in Sections 5.1 and 5.2, it can be interpreted as a relatively efficient model in terms of the balance between performance and resource usage [19,24].

In the case of the O-DU attack dataset, the overall training time and resource consumption significantly decreased. This efficiency is primarily attributed to the substantially smaller total number of instances (dataset size) compared to the network-layer data, which inherently reduced the required computational cycles per epoch and facilitated faster convergence. However, the trend of relative differences in resource usage among the models remained similar to that of the O-CU environment.

While the resource consumption within the AI/ML Framework provides insight into the training overhead, determining the optimal model for the Near-RT RIC deployment requires a comprehensive quantitative analysis between computational complexity and accuracy. Through a Pareto analysis (Fig. 11), the trade-offs among the models were quantitatively evaluated, providing the rationale for the optimal model selection to satisfy the strict constraints of the Near-RT RIC.

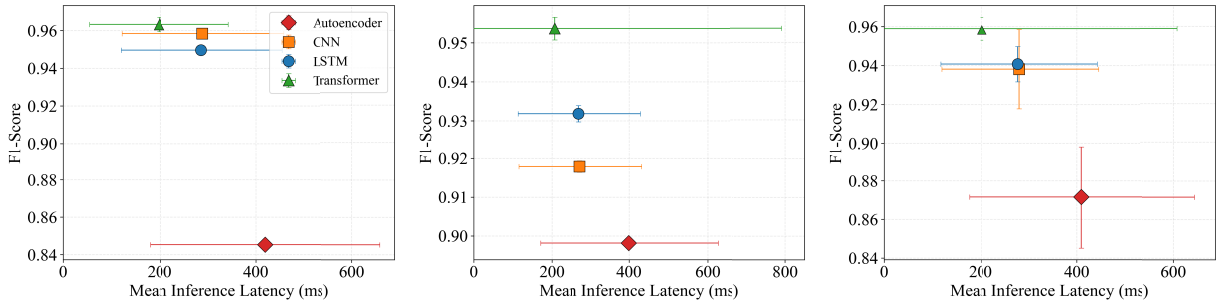

Figure 11: Pareto analysis of the trade-off between detection performance (F1-score) and computational efficiency (mean inference latency) for the four evaluated models: (**a**) evaluation on the O-CU network-layer attack dataset, (**b**) evaluation on the O-DU radio telemetry attack dataset, and (**c**) the combined average performance across both environments for holistic model selection.

Fig. 11 illustrates the trade-off between the F1-score and the average inference latency for each model across the O-CU, O-DU, and their combined average. On the O-CU dataset, the Transformer recorded the highest F1-score (0.963) and the lowest latency (~180 ms), while the LSTM and CNN achieved F1-scores of 0.950 and 0.959, respectively, at a similar latency (~280 ms). The Autoencoder ranked lowest in both F1 (0.845) and latency (~430 ms). On the O-DU dataset, the Transformer maintained the highest F1-score (~0.960); however, its latency error bars exhibited extreme variance, ranging from 0 to over 800 ms. In contrast, the LSTM demonstrated a stable latency distribution at an F1-score of 0.932, consistently outperforming the CNN (F1 0.918, ~280 ms).

Despite occupying the most advantageous position in this Pareto analysis, the Transformer is unsuitable for practical deployment. In the scalability evaluation of 80,000 instances, the measurement could not be completed for both O-CU and O-DU datasets, as the inference was terminated at approximately 20,000

instances due to KServe pod resource exhaustion, rendering further latency measurement infeasible. This is attributed to the limitations of the memory and computing resources in the experimental environment, indicating that the high resource demands of the Transformer act as a substantial bottleneck during large-scale instance processing. Furthermore, as demonstrated in Table 6, the Transformer consumes approximately 2.1 times the CPU and 2.3 times the RAM compared to the LSTM, making it inappropriate for the Near-RT RIC environment, where xApps must coexist within a shared infrastructure.

Meanwhile, in the ΔF1 analysis from Table 5, the LSTM recorded ΔF1 = +0.0008 on O-CU and ΔF1 = −0.0637 on O-DU during the transition from the AI/ML framework to the Near-RT RIC, demonstrating consistently superior performance preservation compared to the CNN (O-CU: −0.0003, O-DU: −0.0791).

In particular, as illustrated in the combined Pareto analysis (Fig. 11c), the LSTM confirms a marginal yet distinct advantage in detection performance over the CNN while maintaining stable latency. Therefore, by comprehensively considering inference latency, detection accuracy, resource efficiency, scalability, and RIC deployment stability, the LSTM is determined to be the most practical and optimal model for deploying the IDS xApp in the Near-RT RIC environment.

6 Discussion

This study quantitatively analyzed the KPI of the IDS xApp operating in the O-RAN Near-RT RIC environment, primarily focusing on detection accuracy and near-real-time inference latency [19].

First, in terms of detection performance, supervised learning-based models (LSTM, CNN, Transformer) demonstrated highly stable F1-scores across both O-CU and O-DU datasets. Notably, the performance variations (ΔF1, ΔPrec) between the training and deployment environments were tightly bounded within ±0.1 pp. This validates that the detection capabilities of supervised models can be robustly preserved even when transitioned to a Near-RT RIC-based inference environment [23]. Conversely, the unsupervised learning-based Autoencoder exhibited a noticeable decrease in Precision and F1-score upon deployment. Rather than pinpointing a single cause, this empirically suggests that fixed-threshold mechanisms are highly sensitive to data distribution shifts in actual operational environments. Therefore, environmental sensitivity derived from structural differences (supervised vs. unsupervised) must be carefully weighed during model selection [14].

Regarding inference latency, the interpretation centers on the strict Near-RT RIC constraints (10 to 1000 ms) [27]. Experimental results confirmed that structurally optimized models (LSTM, CNN, Autoencoder) successfully maintained responses well within the 1000 ms constraint across Mean, P95, and P99 metrics, even when scaling the load up to 80,000 instances. In contrast, the computationally heavy Transformer revealed its operational limit early; its inference was terminated at approximately 20,000 instances due to KServe pod resource exhaustion, beyond which no further latency measurements could be obtained. While the LSTM and CNN consistently controlled their tail latencies and adhered to a linear time complexity ($\mathcal{O}(N)$), the Transformer's failure indicates that its latency growth becomes unmanageable under large-scale concurrent requests.

Furthermore, integrating the training resource analysis reveals clear trade-offs. The Transformer demanded the highest computational overhead, whereas the LSTM required comparatively lower training resources while delivering top-tier detection accuracy and tail latency stability. This underscores that in O-RAN deployments, evaluating a single metric is insufficient; a holistic approach encompassing detection efficacy, near-real-time responsiveness, and resource footprint is essential [24].

Consequently, by establishing near-real-time constraints as a definitive evaluation criterion, this study provides empirical evidence that the optimal balance between detection performance and latency characteristics varies fundamentally depending on the underlying AI/ML model structure.

7 Conclusion

In this paper, an AI-based IDS xApp was designed and deployed within the O-RAN Near-RT RIC environment. Utilizing O-CU network layer and O-DU radio telemetry attack datasets, the system's anomaly detection performance and near-real-time inference scalability were quantitatively verified.

The experimental results demonstrated that supervised learning models (LSTM, CNN, Transformer) achieved exceptional detection accuracy, with F1-scores reaching up to 0.96 on O-CU data and 0.99 on O-DU data. Crucially, their performance variations (ΔF1, ΔPrec) were limited to a negligible $\pm$0.1 pp upon deployment. Among them, the LSTM exhibited the smallest fluctuation range, proving to be the most robust architecture against execution environment transitions. In contrast, the Autoencoder suffered relatively severe performance degradation due to the vulnerability of its reconstruction error-based thresholding to data distribution shifts. These findings affirm that the proposed AI-based IDS xApp, particularly when paired with stable supervised models, serves as a reliable intrusion detection mechanism in practical O-RAN deployments.

Moreover, the latency scalability analysis revealed that the inference processing time maintained a robust linear $\mathcal{O}(N)$ trajectory as concurrent prediction instances scaled up to 80,000. While the Transformer's inference was terminated at approximately 20,000 instances due to resource exhaustion caused by its $\mathcal{O}(N^2)$ complexity, the structurally optimized models (LSTM, CNN) strictly satisfied the Near-RT RIC's 1000 ms constraint even at the maximum evaluated load of 80,000 instances, securing highly stable tail latencies. This experimentally proves that linear predictability and near-real-time constraints can be concurrently achieved even under large-scale network traffic.

Coupled with the SMO domain resource analysis—where the LSTM demonstrated optimal efficiency in training time and computational overhead—this study explicitly highlights the necessity of a multi-dimensional evaluation (detection performance, latency scalability, and resource consumption) for O-RAN AI applications.

In conclusion, this empirical study establishes that ensuring $\mathcal{O}(N)$ time complexity and near-real-time scalability is as critical as detection accuracy when selecting models for O-RAN intelligent control. While the Transformer model's inference was terminated due to resource exhaustion caused by its $\mathcal{O}(N^2)$ complexity at approximately 20,000 instances, this overhead can be mitigated in future work by adopting lightweight attention mechanisms, such as Linear or Sparse Attention, to reduce quadratic complexity. Additionally, transitioning from the current vCPU-based environment to high-performance hardware accelerators, such as GPUs or Neural Processing Units (NPUs), will enable faster parallel processing of large-scale matrix operations, thereby extending the system's operational breaking point. Future research will focus on analyzing the IDS xApp's performance dynamics under varying load conditions and more sophisticated, multi-vector attack scenarios within complex real-world O-RAN testbeds.

However, this study has certain limitations. The evaluation was conducted on a single-node vCPU-based environment with a fixed resource allocation of 6 vCPUs, which may not fully represent the performance characteristics of production-grade O-RAN deployments utilizing GPU/NPU accelerators or multi-node distributed inference architectures. Furthermore, the detection performance was validated using a single public dataset, and additional evaluation with diverse real-world O-RAN traffic traces would further strengthen the generalizability of the findings.

Acknowledgement: This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT).

Funding Statement: This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (RS-2024-00396797, Development of core technology for intelligent O-RAN security platform).

Author Contributions: Conceptualization of background, Hyeonsoo Yu and Hwankuk Kim; methodology and design, Hyeonsoo Yu; data collection, Hyeonsoo Yu; analysis and interpretation of results, Hyeonsoo Yu and Hwankuk Kim; resources, Hyeonsoo Yu; writing—original draft preparation, Hyeonsoo Yu; writing—review and editing, Hyeonsoo Yu and Hwankuk Kim; visualization, Hyeonsoo Yu. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The data that support the findings of this study are openly available in Kaggle at https://www.kaggle.com/datasets/netslabdemo/netslab-5g-oran-idd.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Polese M, Bonati L, d'Oro S, Basagni S, Melodia T. Understanding O-RAN: architecture, interfaces, algorithms, security, and research challenges. IEEE Commun Surv Tutor. 2023;25(2):1376–411.
2. Kaliyammal TP, Chandrasekar T, Vinay V, Raja IV, Maneesha P, Rajisha P, et al. Open RAN: evolution of architecture, deployment aspects, and future directions. arXiv:2301.06713. 2023.
3. Garcia-Saavedra A, Costa-Perez X. O-RAN: disrupting the virtualized RAN ecosystem. IEEE Commun Stand Mag. 2021;5(4):96–103. doi:10.1109/mcomstd.101.2000014.
4. Sivaraj R, Rajagopal S. O-RAN architecture overview. 2023 [cited 2026 Apr 20]. Available from: https://onlinelibrary.wiley.com/doi/10.1002/9781119886020.ch3.
5. Balasubramanian B, Daniels ES, Hiltunen M, Jana R, Joshi K, Sivaraj R, et al. RIC: a RAN intelligent controller platform for AI-enabled cellular networks. IEEE Internet Comput. 2021;25(2):7–17.
6. Marinova S, Leon-Garcia A. Intelligent O-RAN beyond 5G: architecture, use cases, challenges, and opportunities. IEEE Access. 2024;12:27088–114.
7. Zwarico A. ORAN security. In: Open RAN: the definitive guide. Hoboken, NJ, USA: Wiley; 2023. doi:10.1002/9781119886020.ch8.
8. Soltani S, Amanloo A, Shojafar M, Tafazolli R. Intelligent control in 6G open RAN: security risk or opportunity? IEEE Open J Commun Soc. 2025;6:840–80.
9. Baguer P, Yilma GM, Municio E, García-Avilés G, Garcia-Saavedra A, Liebsch M, et al. Attacking O-RAN interfaces: threat modeling, analysis and practical experimentation. IEEE Open J Commun Soc. 2024;5:4559–77.
10. Groen J, d'Oro S, Demir U, Bonati L, Villa D, Polese M, et al. Securing O-RAN open interfaces. IEEE Trans Mobile Comput. 2024;23(12):11265–77. doi:10.1109/tmc.2024.3393430.
11. O-RAN security threat modeling and risk assessment 8.0. O-RAN Alliance WG11. Technical Report. No. Threat-Modeling-R005-v08.00. 2026 [cited 2026 Apr 20]. Available from: https://www.o-ran.org/specifications.
12. Hung CF, Chen YR, Tseng CH, Cheng SM. Security threats to xApps access control and E2 interface in O-RAN. IEEE Open J Commun Soc. 2024;5:1197–203. doi:10.1109/ojcoms.2024.3364840.
13. Tseng Ch, Hung CF, Hong BK, Cheng SM. On manipulating routing table to realize redirect attacks in O-RAN by malicious xApp. In: Proceedings of the 2023 26th International Symposium on Wireless Personal Multimedia Communications (WPMC). Piscataway, NJ, USA: IEEE; 2023. p. 288–92.
14. Hung CF, Tseng CH, Cheng SM. Anomaly detection for mitigating xApp and E2 interface threats in O-RAN Near-RT RIC. IEEE Open J Commun Soc. 2025;6(1):1682–94. doi:10.1109/ojcoms.2025.3546760.

15. Abdalla AS, Moore J, Adhikari N, Marojevic V. ZTRAN: prototyping zero trust security xApps for open radio access network deployments. arXiv:2403.04113. 2024.
16. Groen J, D'Oro S, Demir U, Bonati L, Polese M, Melodia T, et al. Implementing and evaluating security in O-RAN: interfaces, intelligence, and platforms. arXiv:2304.11125. 2023.
17. Jiang H, Chang H, Mukherjee S, Van der Merwe J. OZTrust: an O-RAN zero-trust security system. In: Proceedings of the 2023 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN). Piscataway, NJ, USA: IEEE; 2023. p. 129–34.
18. El-Hajj M. Secure and trustworthy open radio access network (O-RAN) optimization: a zero-trust and federated learning framework for 6G networks. Future Internet. 2025;17(6):233.
19. Amachaghi EN, Shojafar M, Foh CH, Moessner K. A survey for intrusion detection systems in open RAN. IEEE Access. 2024;12(3):88146–73. doi:10.1109/access.2024.3408690.
20. Wang S, Lipman J, Abolhasan M, Babar MA. AI-driven intrusion detection system for open radio access networks: a survey. 2025 [cited 2026 Apr 20]. Available from: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=6293189.
21. Benzaïd C, Taleb T, Cavalcanti R, Sánchez A. AI in open RAN security: enabler and threat—a comprehensive survey and future perspectives. 2025 [cited 2026 Apr 20]. Available from: https://www.techrxiv.org/doi/full/10.36227/techrxiv.176471864.44115969.
22. Almeida GM, Bruno GZ, Huff A, Hiltunen M, Duarte EP, Both CB, et al. RIC-O: efficient placement of a disaggregated and distributed RAN intelligent controller with dynamic clustering of radio nodes. IEEE J Sel Areas Commun. 2024;42(2):446–59.
23. Dayaratne T, Vo V, Lai S, Abuadbba S, Haydon B, Suzuki H, et al. Exploiting and securing ML solutions in near-RT RIC: a perspective of an xApp. arXiv:2406.12299. 2024.
24. Maxenti S, D'Oro S, Bonati L, Polese M, Capone A, Melodia T. ScalO-RAN: energy-aware network intelligence scaling in open RAN. arXiv:2312.05096. 2023.
25. Dayaratne T, Pham Ngoc D, Vo V, Lai S, Abuadbba S, Suzuki H, et al. Robust anomaly detection in O-RAN: leveraging LLMs against data manipulation attacks. arXiv:2508.08029. 2025.
26. Porambage P, Christopoulou M, Han B, Asif Habibi M, Bogucka H, Kryszkiewicz P. Security, privacy, and trust for open radio access networks in 6G. IEEE Open J Commun Soc. 2024;6:332–61. doi:10.1109/ojcoms.2024.3519725.
27. O-RAN Architecture Description. O-RAN alliance WG1. O-RAN.WG1.O-RAN-architecture-description. 2026 [cited 2026 Apr 20]. Available from: https://www.o-ran.org/specifications.
28. Park H, Nguyen T, Park L. An investigation on Open-RAN specifications: use cases, security threats, requirements. Discuss Comput Model Eng Sci. 2024;141(1):13–41.
29. O-RAN Software Community. O-RAN SC Architecture. 2026 [cited 2026 Mar 11]. Available from: https://docs.o-ran-sc.org/en/latest/architecture/architecture.html.
30. Elyasi A, Ashdown A, Rumman K, Restuccia F. O-RAN xApps: survey and research challenges. SSRN. 2025. doi:10.2139/ssrn.5236117.
31. Feraudo A, Maxenti S, Lacava A, Bellavista P, Polese M, Melodia T. xDevSM: streamlining xApp development with a flexible framework for O-RAN E2 service models. In: Proceedings of the ACM MobiCom'24: Proceedings of the 30th Annual International Conference on Mobile Computing and Networking. New York, NY, USA: ACM; 2024. p. 1954–61.
32. Yu H, Al Barat M, Xiao Y, Hou YT, Lou W. Closing the visibility gap: a monitoring framework for verifiable open RAN operations. In: Proceedings of the 2025 IEEE Conference on Communications and Network Security (CNS). Piscataway, NJ, USA: IEEE; 2025. p. 1–9.
33. Başaran OT, Başaran M, Turan D, Bayrak HG, Sandal YS. Deep autoencoder design for RF anomaly detection in 5G O-RAN near-RT RIC via xApps. In: Proceedings of the 2023 IEEE International Conference on Communications Workshops (ICC Workshops). Piscataway, NJ, USA: IEEE; 2023. p. 549–55.
34. Basaran OT, Dressler F. XAInomaly: explainable, interpretable and trustworthy AI for xURLLC in 6G open-RAN. In: Proceedings of the 2024 3rd International Conference on 6G Networking (6GNet). Piscataway, NJ, USA: IEEE; 2024. p. 93–101.

35. Dimou S, Noubir G. ARGOS: anomaly recognition and guarding through O-RAN sensing. In: Proceedings of the 2025 IEEE Conference on Communications and Network Security (CNS). Piscataway, NJ, USA: IEEE; 2025. p. 1–11.
36. Alimohammadi H, Chatzimiltis S, Mayhoub S, Shojafar M, Soleymani SA, Akbas A, et al. KPI poisoning: an attack in open RAN near real-time control loop. In: Proceedings of the 2024 IEEE Future Networks World Forum (FNWF). Piscataway, NJ, USA: IEEE; 2024. p. 712–8.
37. Kakani PK, Habibi MA, Balannagari MRC, Costa-Pérez X, Schotten HD. Mitigating ML-driven adversarial attacks on xApps using dynamic defense mechanisms. IEEE Open J Commun Soc. 2025;6:6912–29. doi:10.1109/ojcoms.2025.3602200.
38. Lee S, Kim H. An AI/ML framework-driven approach for malicious traffic detection in open RAN. Comput Model Eng Sci. 2025;145(2):2657–82. doi:10.32604/cmes.2025.070627.
39. Yungaicela-Naula NM, Sharma V, Scott-Hayward S. Misconfiguration in O-RAN: analysis of the impact of AI/ML. Comput Netw. 2024;247(9):110455. doi:10.1016/j.comnet.2024.110455.
40. Giannopoulos A, Spantideas S, Levis G, Kalafatelis A, Trakadas P. COMIX: generalized conflict management in O-RAN xApps-architecture, workflow, and a power control case. IEEE Access. 2025;13:116684–700.
41. Lacava A, Bonati L, Mohamadi N, Gangula R, Kaltenberger F, Johari P, et al. dApps: enabling real-time AI-based open RAN control. Comput Netw. 2025;269:111342.
42. Moore J, Adhikari N, Abdalla AS, Marojevic V. Toward secure and efficient O-RAN deployments: secure slicing xApp use case. In: Proceedings of the 2023 IEEE Future Networks World Forum (FNWF). Piscataway, NJ, USA: IEEE; 2023. p. 1–6.
43. Moore J, Abdalla AS, Khanal P, Marojevic V. Integrated LLM-based intrusion detection with secure slicing xApp for securing O-RAN-enabled wireless network deployments. In: Proceedings of the 2025 IEEE International Conference on Communications Workshops (ICC Workshops). Piscataway, NJ, USA: IEEE; 2025. p. 274–9.
44. Civciss A, Ravihansa V, Zadeh FA, Sandeepa C, Liyanage M. Silent signals, loud threats: using dApps for radio signal intelligence-based intrusion detection in 5G O-RAN. In: Proceedings of the GLOBECOM 2025—2025 IEEE Global Communications Conference. Piscataway, NJ, USA: IEEE; 2025. p. 6075–80.
45. Yin C, Zhu Y, Fei J, He X. A deep learning approach for intrusion detection using recurrent neural networks. IEEE Access. 2017;5:21954–61. doi:10.1109/access.2017.2762418.