AI-Open-RAN for Non-Terrestrial Networks

Tri Nhu Do
Telecom Neural Detection Lab, Department of Electrical Engineering, Polytechnique Montréal, Montreal, QC, Canada.
Email: tri-nhu.do@polymtl.ca

***Abstract*—In this paper, we propose the concept of AIO-RAN-NTN, a unified all-in-one Radio Access Network (RAN) for Non-Terrestrial Networks (NTNs), built on an open architecture that leverages open interfaces and artificial intelligence (AI)-based functionalities. This approach advances interoperability, flexibility, and intelligence in next-generation telecommunications. First, we provide a concise overview of the state-of-the-art architectures for Open-RAN and AI-RAN, highlighting key network functions and infrastructure elements. Next, we introduce our integrated AIO-RAN-NTN blueprint, emphasizing how internal and air interfaces from AIO-RAN and the 3rd Generation Partnership Project (3GPP) can be applied to emerging environments such as NTNs. To examine the impact of mobility on AIO-RAN, we implement a testbed transmission using the OpenAirInterface platform for a standalone (SA) New Radio (NR) 5G system. We then train an AI model on realistic data to forecast key performance indicators (KPIs). Our experiments demonstrate that the AIO-based SA architecture is sensitive to mobility, even at low speeds, but this limitation can be mitigated through AI-driven KPI forecasting.**

***Index Terms*—O-RAN, AI-RAN, 3GPP, air interface, AI, LSTM non-terrestrial network (NTN), forecasting KPI**

I. Introduction

The next generation (NG) of telecommunication systems is expected to bring a revolutionary Radio Access Network (RAN) with massive, scalable, low-latency, and ultra-reliable connectivity, enabling seamless integration with current and future terrestrial and non-terrestrial networks (NTNs). To support flexible multi-vendor deployments, the 3rd Generation Partnership Project (3GPP) standardized the Central Unit (CU) / Distributed Unit (DU) functional split (Option 2, F1 interface, Technical Specification (TS) 38.47x / F1 Application Protocol (F1AP)) in Release 15 (Rel-15) and evolved it in Rel-17 introducing NTN-specific enhancements [1], [2]. Complementing this, the Open Radio Access Network (O-RAN) Alliance and the Artificial Intelligence–Radio Access Network (AI-RAN) Alliance propose open and intelligent RAN architectures that extend modularity, virtualization, and artificial intelligence/machine learning (AI/ML), i.e., driven intelligence into RAN design [3]. Their objectives include cloud-native disaggregation, standardized application programming interfaces (APIs), interfaces (e.g., OpenAirInterface [4]), and advanced automation through the RAN Intelligent Controller (RIC) (xApps/rApps) and the Service Management and Orchestration (SMO) framework [3].

Specifically, the AI-RAN and O-RAN Alliances extend 3GPP standards with open interfaces, RAN disaggregation, and AI-driven automation [5]. AI is becoming an integral part of RAN evolution. 3GPP Rel-18 introduced a standardized AI/ML framework for NG-RAN, with Rel-19 extending its capabilities [6]. In O-RAN, programmable control planes via the Near-Real-Time (Near-RT) and Non-Real-Time (Non-RT) RICs and the SMO enable practical deployment of AI-driven RAN optimization. Meanwhile, AI-RAN initiatives aim to accelerate progress toward fully AI-native RANs [1], [7].

For NTNs, 3GPP Rel-17 introduced New Radio (NR)-based NTN support, enabling the integration of satellites and other non-terrestrial platforms with terrestrial 5G systems. NTN mobility is dominated by moving satellites and spot beams, causing frequent handovers and ping-pong effects, while large and time-varying delay and Doppler shifts complicate synchronization, random access, measurements, and handovers. Rel-17 addressed these with Global Navigation Satellite System (GNSS) / ephemeris-assisted timing advance and frequency pre-compensation; Rel-18 (5G-Advanced) added L1/L2-triggered mobility and enhanced conditional handover; and Rel-19 further improves inter-CU mobility and tracking-area handling. Mobility management therefore remains a central design concern for NG-RAN in NTN environments.

In this paper, we propose Artificial Intelligence Open RAN (AIO-RAN), a unified RAN architecture that integrates the strengths of AI-RAN and O-RAN on top of standardized 3GPP foundations, specifically tailored for NTNs. The proposed architecture consolidates openness, programmability, and AI-native coordination across vendors, operators, and clouds. Our contributions are twofold. First, we detail the AIO-RAN-NTN blueprint, illustrating how specific interfaces and functional blocks can be realized. Second, we examine the impact of mobility using an OpenAirInterface-based standalone NR next-generation NodeB (gNB) testbed and demonstrate that the proposed AI-driven KPI forecasting method can effectively mitigate mobility-induced performance degradation. Due to space limitations, the full-size high-resolution figure, testbed dashboard, data visualizations, AI/ML configurations, more illustrations and training/testing details are available in our *code repository* at github.com/TND-Lab/AIO-RAN-NTN

II. Proposed Blueprint of AIO-RAN-NTN: Architecture and Interfaces

3GPP NG-RAN provides the standardized foundation, with logical nodes (CU, DU, Radio Unit (RU)) and functional splits (e.g., Option 2) defining the baseline for openness and flexibility [2], [8] and references therein. Building on this, our AIO-RAN-NTN design emphasizes hardware unit splitting and open fronthaul connectivity as key enablers for deployment in NTNs.

A. Open and AI-RAN Architecture Splitting

1) 3GPP Open RAN Baseline: The standardized NG-RAN architecture specifies possible split options and has

This research was supported by NSERC (Canada) under Grant Nos. RGPIN-2025-05010 and DGECR-2025-00215.

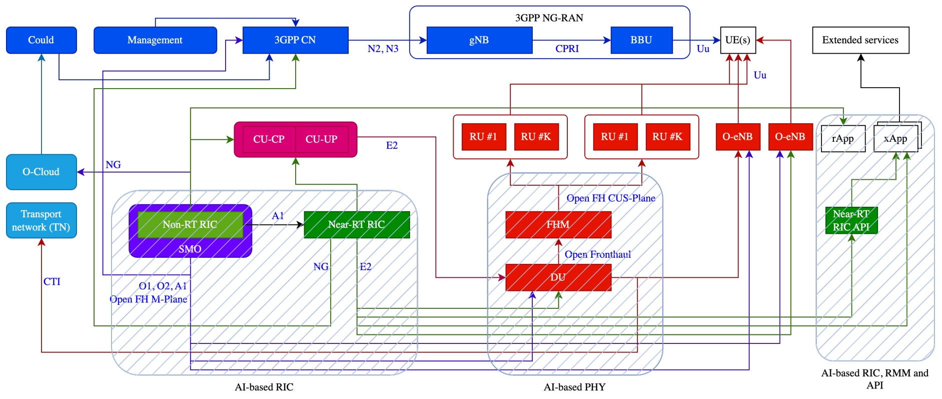


Fig. 1. Key elements and interfaces of 3GPP NG-RAN and O-RAN, including the proposed AI-based modules

established Option 2 (F1 interface), decoupling CU, DU, and RU functions, enabling multi-vendor implementations supported by open fronthaul using the enhanced Common Public Radio Interface (eCPRI).

2) Open-RAN Extensions: O-RAN builds on the 3GPP baseline, adopting Split 7-2x for the Open Fronthaul and introducing additional interfaces, including A1 (policy interface), E2 (near-real-time control interface), O1 (management interface), O2 (cloud management interface), and R1 (xApp–RIC interface), to enhance interoperability [5]. Its main principles are cloudification, open interfaces, and programmability, operationalized via the O-RU (O-RAN Radio Unit), O-DU (O-RAN Distributed Unit), O-CU (O-RAN Central Unit), and the RICs (xApps/rApps) together with the SMO framework.

3) AI-RAN Integration: AI-RAN initiatives [7] extend 3GPP/O-RAN by defining AI-native frameworks, including standardized data pipelines, model lifecycle management, and distributed training and inference across device–edge–cloud resources. Reference workloads, such as channel prediction and mobility forecasting, showcase closed-loop optimization in both terrestrial and NTN settings.

B. Key Interfaces for O-RAN and AI-RAN in NTN

To realize AIO-RAN-NTN, several interfaces require adaptation for NTN environments, as shown in Fig. 2. At the NG-RAN–core boundary, the N2 (NG Control, NG-C) and N3 (NG User, NG-U) interfaces connect the gNB-CU control and user planes, respectively, to the 5G Core (5GC) functions Access and Mobility Management Function (AMF) and User Plane Function (UPF). In O-RAN, these retain the same 3GPP protocol stacks, mapped to the O-CU-CP and O-CU-UP.

Within the RAN, the A1 and E2 interfaces enable programmability and AI integration. A1 connects the Non-RT RIC (in the SMO) with the Near-RT RIC to exchange policies and guidance, while E2 connects the Near-RT RIC to O-CU and O-DU elements, supporting real-time control via the E2 Application Protocol and service models.

The Open Fronthaul (FH) interface connects O-DU and O-RU nodes, supporting split options 7 and 8 at the Physical (PHY) layer. It comprises the Control, User, and Synchronization (CUS) planes and the Management (M) plane, with the latter also linking O-RUs to the SMO.

For the user plane, the Uu (NG-Uu) interface connects User Equipment (UE) to the gNB across the full NR protocol stack. Although not redefined by O-RAN, it remains central in NTN scenarios where latency, Doppler, and handover dynamics must be considered. O-RAN's modularity also allows some internal interfaces (e.g., F1-C/U, E1, E2) to be collapsed in deployment, highlighting its scalability for NTN environments.

C. Blueprint for Integrated AIO-RAN-NTN: Proposed Architecture with Open Interfaces

3GPP Rel-17 and Rel-18 introduced NR-based NTN capabilities, including PHY-layer adaptations and regenerative payloads on Low Earth Orbit (LEO) satellites capable of hosting gNB functions. Building on these foundations, we propose an integrated AIO-RAN-NTN architecture that unifies terrestrial and non-terrestrial access while exposing traditionally internal RAN interfaces as open, NTN-capable air/space links. The design targets a fully Standalone (SA) 5G deployment, with Non-Standalone (NSA) interoperability preserved during the transition phase. A central challenge in NTNs remains the trade-off between mobility and delay, requiring differentiated handling of delay-tolerant services and mobility-sensitive applications.

1) System Blueprint: Management/Orchestration (SMO/O-Cloud): O-RAN SMO spans ground edge and cloud. O1 manages Network Function (NF) lifecycle, O2 manages O-Cloud resources, A1 steers policies to the Non-RT RIC, and E2 controls and observes Near-RT RIC/xApps across TN and NTN. *AI-RAN Layer:* The AI-RAN framework overlays this blueprint, providing standardized data pipelines, feature stores, and lifecycle management. Models for mobility prediction, Doppler compensation, and beam-hopping optimization are trained across distributed TN and NTN data and orchestrated by the Non-RT RIC, with inference hosted in Near-RT RIC/xApps and device/edge accelerators. This ensures closed-loop optimization across terrestrial and non-terrestrial domains.

2) New Elements Needed in Our Blueprint: We propose the following new elements to enable the blueprint:

- A-OFH: The NTN Aerial Open Fronthaul Profile extends the existing OFH with robustness modes, adaptive numerology, Doppler-aware tracking, and synchronization-quality metrics.

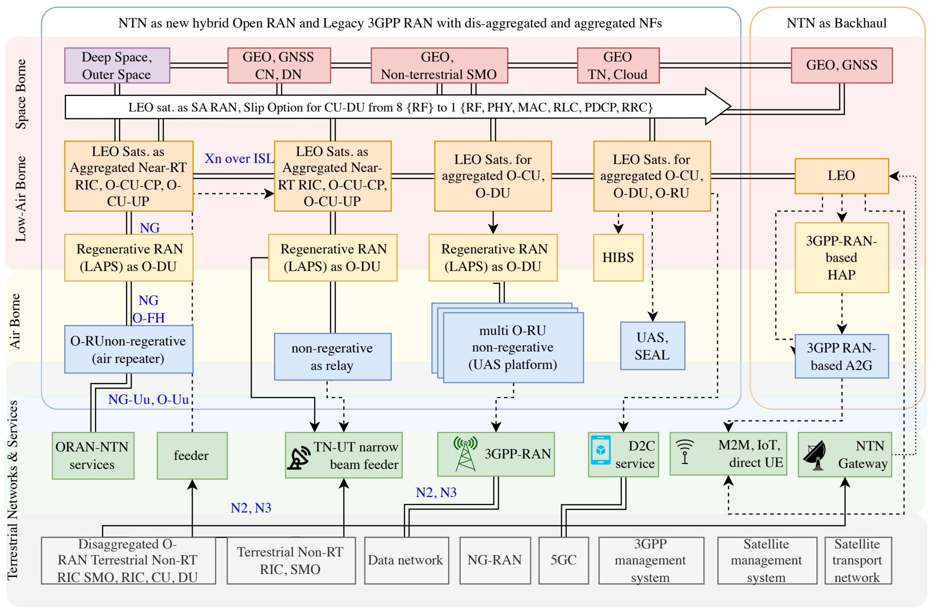

NTN as new hybrid Open RAN and Legacy 3GPP RAN with dis-aggregated and aggregated NFs
NTN as Backhaul
Space Borne
Low-Air Borne
Air Borne
Terrestrial Networks & Services
Deep Space, Outer Space
GEO, GNSS CN, DN
GEO, Non-terrestrial SMO
GEO TN, Cloud
GEO, GNSS
LEO sat. as SA RAN, Slip Option for CU-DU from 8 {RF} to 1 {RF, PHY, MAC, RLC, PDCP, RRC}
LEO Sats. as Aggregated Near-RT RIC, O-CU-CP, O-CU-UP
Xn over ISL
LEO Sats. as Aggregated Near-RT RIC, O-CU-CP, O-CU-UP
LEO Sats. for aggregated O-CU, O-DU
LEO Sats. for aggregated O-CU, O-DU, O-RU
LEO
NG
Regenerative RAN (LAPS) as O-DU
Regenerative RAN (LAPS) as O-DU
Regenerative RAN (LAPS) as O-DU
HIBS
3GPP-RAN-based HAP
NG
O-FH
O-RUnon-regenerative (air repeater)
non-regerative as relay
multi O-RU non-regerative (UAS platform)
UAS, SEAL
3GPP RAN-based A2G
NG-Uu, O-Uu
ORAN-NTN services
feeder
TN-UT narrow beam feeder
3GPP-RAN
D2C service
M2M, IoT, direct UE
NTN Gateway
N2, N3
N2, N3
Disaggregated O-RAN Terrestrial Non-RT RIC SMO, RIC, CU, DU
Terrestrial Non-RT RIC, SMO
Data network
NG-RAN
5GC
3GPP management system
Satellite management system
Satellite transport network

Fig. 2. Proposed blueprint of an integrated AIO-RAN-NTN architecture focusing on aggregated NFs and air interfaces

- NTN-IAB: Open NR-Integrated Access and Backhaul (IAB) extensions are adapted for space and aerial donors or relays. They include support for beam-hopping and contact-window management.
- Delay-Aware E2/xApps: These extensions incorporate prediction horizons and actuation deadlines. AI-enabled xApps support tasks such as pass prediction, Hybrid Automatic Repeat Request (HARQ) budget control, and channel robustness.
- Federated rApps/xApps: Federated learning (FL) is applied across distributed terrestrial network (TN) and NTN domains [9]. The Non-RT RIC orchestrates training, while aggregation occurs at Near-RT RICs for enhanced mobility and blockage prediction.
- Energy- and Cost-Aware NF Placement: AI-guided placement of RU, DU, and CU-UP functions is carried out across ground and aerial resources, subject to power, budget, and Key Performance Indicator (KPI) constraints.

3) Element Blocks and Interfaces: Enabled by the proposed elements, we define three blueprint options:

BP-1 Disaggregated NTN-RAN (aerial RU with DU/CU on the ground): This option uses the A-OFH profile (7.2x) over NTN and the F1 interface (Option 2) over a delay-tolerant backhaul. NTN adaptations include Doppler and timing advance compensation, extended protocol timers, jitter buffers, and holdover synchronization with sync-quality monitoring. AI-based Radio Resource Management (RRM) is supported through the Near-RT RIC over the E2 interface, with delay-aware schedulers, beam-hopping awareness, AI-driven channel forecasting, HARQ budget control, and pass prediction.

BP-2 Aggregated NTN-RAN (aerial RU+DU with CU on the ground, or aerial RU+DU+CU-UP): This option collapses the interfaces defined in BP-1 to reduce latency and signaling overhead. Fronthaul bandwidth requirements are further reduced through the use of NTN-IAB. The system employs minimal A-OFH together with NTN-based F1, N2, and N3 interfaces, reinforced by sync-quality metrics to maintain reliable performance. Energy-aware duty cycling is applied on aerial and space platforms, guided by AI-driven workload placement and federated rApps/xApps for distributed mobility and blockage prediction.

BP-3 Hybrid NSA to SA Migration (with legacy 3GPP support): This option extends BP-1 and BP-2. NTN-IAB is used for TN/NTN traffic steering and splitting, combined with energy- and cost-aware NF placement. Federated rApps/xApps enable cross-domain learning between TN and NTN to improve mobility and blockage resilience. The RIC topology includes a centralized Non-RT RIC and distributed Near-RT RIC instances deployed at the edge or on aerial platforms to reduce control-loop delay.

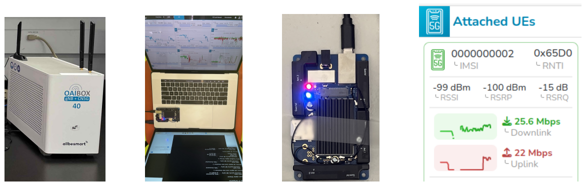
III. Demo: AI-based KPI Forecasting-Empowered OpenAirInterface-based SA Architecture

A. OpenAirInterface Platform

OpenAirInterface (OAI) is an open-source project that implements 3GPP-compliant cellular technologies on general-purpose x86/Linux platforms with software-defined radio hardware such as USRP devices [4]. The current OAI 5G RAN project develops both Non-Standalone and Standalone gNB implementations, as well as NSA/SA UEs. OAI has recently been extended with O-RAN components. In particular, the FlexRIC platform provides a flexible, O-RAN–compliant RAN Intelligent Controller (RIC) and an E2 agent integrated with the OAI radio stack [4]. This enables interoperability with O-RAN E2 interfaces and allows control through xApps, such as for network slicing in NSA 5G deployments.

B. Experiment Setup and Downlink Test

We demonstrate an over-the-air transmission using a 3GPP-standard-compliant Standalone 5G system implemented on OAI. The setup consists of a complete end-to-end configuration including: (i) a 5G Core Network, (ii) an OAI-based 5G gNodeB, and (iii) mobile user equipment, as shown in Fig. 3. This testbed allows us to analyze mobility-induced performance degradation and collecting data for AI-based KPI forecasting.



(a) (b) (c) (d)

Fig. 3. (a) OAI-gNB, (b) mobile UE, (c) UE Quectel, and (d) UE dashboard

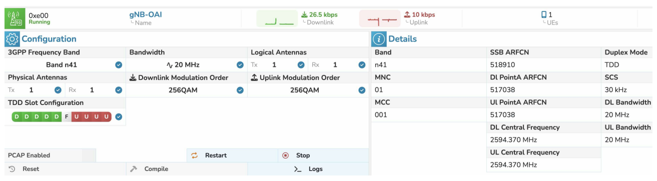
1) gNB: The gNB is implemented on an OAIBOX-40 [10] running the open-source OAI 5G stack, with CU and DU functions integrated on a single host alongside the 5G Core. The RU is realized with a software-defined radio (SDR) supporting up to 40 MHz bandwidth.

2) UE: The UE is a Quectel 5G module with integrated antennas, as shown in Figs. 3b–3c. It is operated via a programmed SIM card and controlled from a laptop using Quectel drivers and a terminal interface (MobaXterm).

3) Configuration and Downlink Test: The gNB parameters (e.g., frequency, time structure, modulation/coding scheme) are configured online through a web-based GUI dashboard, as shown in our repository. Downlink performance is evaluated using iPerf, with the gNB acting as the iPerf server and transmitting UDP traffic to the UE terminal. To capture a variety of propagation scenarios, the UE is moved to experience both LOS and NLOS channel conditions. For monitoring purposes, end-to-end KPI results are collected in real time (with one-second resolution) and stored in JSON format.

C. Time Series Analysis and LSTM-based KPI Forecasting

As shown in Fig. 5, key performance indicators such as RSSI, SINR, SNR, PHR, etc. fluctuate significantly with UE mobility. These variations force the gNB to continuously adapt the modulation and coding scheme (MCS), and frequent drops may cause connection loss. In our OAI-based setup, reconnection requires manual intervention



0xe00 Running | gNB-OAI Name | 26.5 kbps Downlink | 10 kbps Uplink | 1 UEs

Configuration

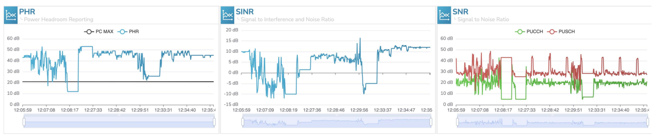
3GPP Frequency Band	Bandwidth	Logical Antennas
Band n41 ✓	20 MHz ✓	Tx 1 ✓ Rx 1 ✓
Physical Antennas	**Downlink Modulation Order**	**Uplink Modulation Order**
Tx 1 ✓ Rx 1 ✓	256QAM ✓	256QAM ✓
TDD Slot Configuration		
D D D D D F U U U U ✓		

PCAP Enabled | Restart | Stop
Reset | Compile | Logs

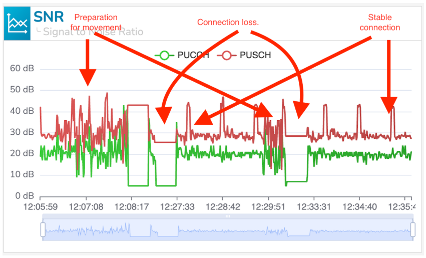
Details

Band	SSB ARFCN	Duplex Mode
n41	518910	TDD
MNC	**Dl PointA ARFCN**	**SCS**
01	517038	30 kHz
MCC	**Ul PointA ARFCN**	**DL Bandwidth**
001	517038	20 MHz
	DL Central Frequency	**UL Bandwidth**
	2594.370 MHz	20 MHz
	UL Central Frequency	
	2594.370 MHz	

Fig. 4. Hardware and signal processing setup at gNB.



(a)



(b)

Fig. 5. Key observed KPIs with comments on connectivity

with noticeable delay, underscoring the need for proactive forecasting of SINR and related features.

D. Feature Extraction: Extract from JSON Data

We represent the dataset as a JSON document D drawn from the class of rooted, finite, labeled trees $\mathcal{T}$ whose internal nodes are dictionaries (key–value maps) and whose leaves are primitives (numbers, strings, booleans), lists, or dictionaries, i.e., $D \in \mathcal{T}$.

a) Loading Raw Data: Given a recording file p, we define the (partial) loading operator $\mathsf{Load}(p)$. This operator allows us to parse the JSON file p so that we can extract and return the corresponding dataset D.

b) Key-based Extraction: For a given key k (a string), we define the extraction operator $\mathsf{E}_k : \mathcal{T} \to \mathcal{L}$, which we use to traverse the dataset D and collect *all* values directly associated with k. We perform this traversal in depth-first, encounter order, so that $\mathsf{E}_k(D) = \big(v_1, v_2, \ldots, v_m\big)$, where each v_i is either a subtree or a primitive found at some occurrence of k.

c) Two-layer Feature Extraction: We fix the keys corresponding to the parameters of interest, e.g., $k_1 = \texttt{ues}$ and $k_2 \in \{\texttt{sinr}\}$. First, we apply the extraction operator to obtain $\mathcal{S} = \mathsf{E}_{k_1}(D) = (S_1, \ldots, S_m), \qquad S_i \in \mathcal{T}$. Next, for each sub-tree S_i, we extract the values associated with k_2, giving $\vec{y}_i = \mathsf{E}_{k_2}(S_i) = \big(y_{i1}, \ldots, y_{i\ell_i}\big)$. Finally, by collecting all such vectors, we obtain $\mathcal{Y} = (\vec{y}_1, \ldots, \vec{y}_m)$.

d) Tabularization: Let $L = \max_i \ell_i$, and define the padding map $\mathsf{Pad} : \mathcal{L} \to \mathbb{R}^L \cup \{\mathrm{NaN}\}^L$, which we use to right-pad shorter sequences with NaNs. By stacking the padded rows, we obtain

$$Y = [\mathsf{Pad}(\vec{y}_1)^\top \ldots \mathsf{Pad}(\vec{y}_m)^\top]^\top \in (\mathbb{R} \cup \{\mathrm{NaN}\})^{m \times L}. \quad (1)$$

We then regard the resulting matrix Y as a multivariate time series, $\mathcal{D} = \{x_t \in \mathbb{R}^d\}_{t=1}^N$, with N time steps and d features (columns). Finally, we export Y to CSV format so that we can feed it into our neural network model.

E. Feature Engineering: Construct Time-Series Data

a) Imputation of Disconnections: Let $F_i(t)$ denote feature i at time t. If the UE is disconnected at t, then we set $F_i(t) = \mathrm{NaN}$. To handle such cases, we replace undefined entries by a constant κ, chosen to be much smaller than the expected connected SNR:

$$F_i(t) \leftarrow \begin{cases} \kappa, & \text{if the UE is disconnected at } t, \\ F_i(t), & \text{otherwise,} \end{cases} \qquad (2)$$

where $\kappa \ll \mathbb{E}[F_i(t) \mid \text{UE connected}]$.

Remark. It is noted that κ plays a significant role during training, as it encodes the separation between connected and disconnected regimes.

b) Data Split: Let $N = |\mathcal{D}|$. Define disjoint, contiguous partitions $\mathcal{D}_{\text{train}} = \{x_t\}_{t=1}^{\lfloor 0.7N \rfloor}$, $\mathcal{D}_{\text{val}} = \{x_t\}_{t=\lfloor 0.7N \rfloor+1}^{\lfloor 0.9N \rfloor}$, and $\mathcal{D}_{\text{test}} = \{x_t\}_{t=\lfloor 0.9N \rfloor+1}^{N}$. Let $d = \dim(\mathcal{D})$ denote the number of features.

c) Scaling $\min - \max$*:* To avoid leakage, fit feature-wise min–max scaling on the *training* split only. For each feature $j \in \{1, \dots, d\}$, $x_{\min}^{(j)} = \min_{x \in \mathcal{D}_{\text{train}}} x^{(j)}$, $x_{\max}^{(j)} = \max_{x \in \mathcal{D}_{\text{train}}} x^{(j)}$. Let $S : \mathbb{R}^d \to [0,1]^d$ be the scaling operator defined componentwise by

$$\big(S(x)\big)^{(j)} = (x^{(j)} - x_{\min}^{(j)})/(x_{\max}^{(j)} - x_{\min}^{(j)}), \qquad (3)$$

with $x_{\min}^{(j)}$ and $x_{\max}^{(j)}$ taken from the training data. If $x_{\max}^{(j)} = x_{\min}^{(j)}$, set $(S(x))^{(j)} = 0$. Applying S to each split yields

$$\tilde{\mathcal{D}}_{\text{train}} = S(\mathcal{D}_{\text{train}}), \tilde{\mathcal{D}}_{\text{val}} = S(\mathcal{D}_{\text{val}}), \tilde{\mathcal{D}}_{\text{test}} = S(\mathcal{D}_{\text{test}}). \qquad (4)$$

d) Windowing with Shift and Labels: Let the window parameters be w_{in} (input width) w_{lbl} (label width) s (shift) $w_{\text{tot}} = w_{\text{in}} + s$. For a split with length $N_{\texttt{set}}$ ($\texttt{set} \in \{\text{train}, \text{val}, \text{test}\}$) and $t \in \{1, \dots, N_{\texttt{set}} - w_{\text{tot}} - w_{\text{lbl}} + 1\}$, define

$$X_t = \tilde{\mathcal{D}}_{\texttt{set}}[t : t + w_{\text{in}} - 1, :] \in [0,1]^{w_{\text{in}} \times d},$$
$$Y_t = \tilde{\mathcal{D}}_{\texttt{set}}[t + w_{\text{tot}} - w_{\text{lbl}} : t + w_{\text{tot}} - 1, :] \in [0,1]^{w_{\text{lbl}} \times d'}.$$

By sliding the window, we obtain

$$\mathcal{W}_{\texttt{set}} = \{(X_t, Y_t)\}_{t=1}^{M_{\texttt{set}}}, M_{\texttt{set}} = N_{\texttt{set}} - w_{\text{tot}} - w_{\text{lbl}} + 1. \qquad (5)$$

F. Mathematical Model of LSTM Training

a) Composite Architecture: Let $T = \texttt{time\_steps} = w_{\text{in}}$ and $d = \texttt{input\_dim}$, as specified in the source code. For an input sequence $X \in \mathbb{R}^{T \times d}$, the model computes $\hat{Y} = f_{\text{NN}}(X) \in \mathbb{R}^{T \times C}$, where $f_{\text{NN}} = f_{\text{out}} \circ \cdots \circ f_{\text{fci}} \circ \cdots \circ f_{\text{LSTMj}} \circ \cdots$ denotes the composition of layers. Specifically, the network consists of N_{LSTM} stacked LSTM layers, followed by N_{FC} fully connected (time-distributed) layers with ReLU activation, and a final linear output layer of width C. Further details on the LSTM architecture can be found in the source code.

b) Label Alignment: If $w_{\text{lbl}} \leq T$, let $\Pi_{\text{lbl}} : \mathbb{R}^{T \times C} \to \mathbb{R}^{w_{\text{lbl}} \times C}$ select the final w_{lbl} time indices: $\hat{Y}^{(\text{lbl})} = \Pi_{\text{lbl}}(\hat{Y})$. Training targets are given by $Y \in [0,1]^{w_{\text{lbl}} \times d'}$, which are taken from the window labels, where d' denotes the dimension of the selected label features. If $C \neq d'$, we apply a fixed feature-selector mapping $Y \mapsto Y' \in [0,1]^{w_{\text{lbl}} \times C}$ to ensure compatibility with the model output width C. It is noted that, in our time-series data, the labels correspond to the data itself observed at different sampling instances.

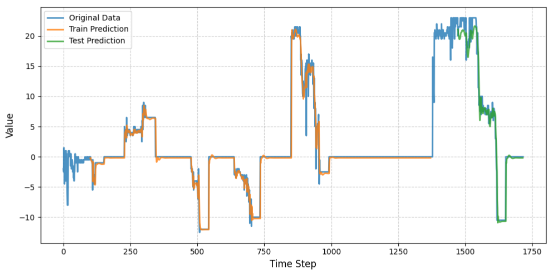


Fig. 6. Predicted KPI, i.e., SINR with $\kappa = 0$

c) Loss in Scaled Domain: With batches indexed by i and time by τ, the loss function is defined as

$$\mathcal{L}_{\text{MSE}} = \frac{1}{B\, w_{\text{lbl}}\, C} \sum_{i=1}^{B} \sum_{\tau=1}^{w_{\text{lbl}}} \sum_{c=1}^{C} \big(\hat{Y}_{i,\tau,c}^{(\text{lbl})} - Y_{i,\tau,c}\big)^2, \qquad (6)$$

where B is the batch size, w_{lbl} the label window length, and C the output dimension. Model parameters are collected in $\Theta = \{W^{(1:N_{\text{LSTM}})}, b^{(1:N_{\text{LSTM}})}, \theta_{\text{LSTM}}\}$, and are updated using a chosen optimizer, with learning rate η, as

$$\Theta \leftarrow \Theta - \eta \nabla_{\Theta} \mathcal{L}_{\text{MSE}}, \qquad (7)$$

d) Training Phase: It is noted that each window $X_t \in [0,1]^{w_{\text{in}} \times d}$ is fed as a sequence of length $T = w_{\text{in}}$. The model then produces $\hat{Y}_t \in \mathbb{R}^{T \times C}$, which is projected to the label slice $\hat{Y}_t^{(\text{lbl})}$ and compared to Y_t. The training procedure is summarized in Algorithm 1.

e) Testing and Inverse Scaling: During evaluation, we draw a batch $(X^{\text{test}}, Y^{\text{test}}) \sim \mathcal{W}_{\text{test}}$ and compute scaled predictions $\hat{Y}^{\text{test}} = f_{\Theta}(X^{\text{test}})$. Let S^{-1} denote the inverse of the train-fitted scaler S, applied componentwise. Predictions and labels are then reported in the original units: $\hat{Y}_{\text{orig}}^{\text{test}} = S^{-1}\big(\hat{Y}^{\text{test}}\big), Y_{\text{orig}}^{\text{test}} = S^{-1}\big(Y^{\text{test}}\big)$. Flattening across batch, time, and feature dimensions yields a total sample count of N_{tr} for training and N_{te} for testing. Performance is then evaluated in terms of RMSE, for example on the training set:

$$\text{RMSE}_{\text{tr}} = \left[\frac{1}{N_{\text{tr}}} \sum_{i=1}^{N_{\text{tr}}} \big(\hat{Y}_{\text{orig},i}^{\text{tr}} - Y_{\text{orig},i}^{\text{tr}}\big)^2\right]^{1/2}. \qquad (8)$$

IV. Results and Discussion

Following Alg. 1, the LSTM models can be trained either on the host laptop (with over-the-air feedback) or directly on the OAIBOX running a Linux platform. Due to space limitations, all specific settings and datasets are provided in our GitHub repository referenced above. In our training, the loss function converges as shown in our our code repository. It is noted that, due to the nature of the raw data—specifically the disconnected events between the gNB and UE—we apply the feature-wise scaler S in (3) to normalize values into the range $[0,1]$. The constant κ is set near the lower bound (typically ≈ 0, or slightly below). In this way, scaling preserves the contrast between connected and disconnected regimes while placing all inputs on a comparable numerical scale, which stabilizes LSTM training and accelerates convergence.

As shown in Fig. 6, we are able to successfully predict the KPI of interest into the future, albeit with some delay. It is

Algorithm 1: Proposed end-to-end pipeline for KPI prediction: load $\rightarrow$ extract $\rightarrow$ impute $\rightarrow$ split $\rightarrow$ scale $\rightarrow$ window $\rightarrow$ train $\rightarrow$ test $\rightarrow$ inverse-scale $\rightarrow$ RMSE

```
Input: JSON path $p$; keys $k_1 = \texttt{ues}$, $k_2 \in \{\texttt{sinr}\}$;
       imputation constant $\kappa$; split ratios
       $(0.7, 0.2, 0.1)$; window params $(w_{\text{in}}, w_{\text{lbl}}, s)$;
       model hparams $(u, h, C, \eta)$; epochs $E$.
Output: Trained parameters $\Theta$; $\text{RMSE}_{\text{tr}}$, $\text{RMSE}_{\text{te}}$
        (original units).
1  Load & extract
2  $D \leftarrow \mathsf{Load}(p)$ $\mathcal{S} \leftarrow \mathsf{E}_{k_1}(D) = (S_1, \dots, S_m)$
   $\mathcal{Y} \leftarrow (\mathsf{E}_{k_2}(S_1), \dots, \mathsf{E}_{k_2}(S_m))$
   $Y \leftarrow \mathsf{PadStack}(\mathcal{Y}) \in (\mathbb{R} \cup \{\text{NaN}\})^{m \times L}$
3  Impute
4  foreach entry $y$ in $Y$ do
5      if $y = NaN$ then
6          $y \leftarrow \kappa$
7  Form time series $\mathcal{D} = \{x_t \in \mathbb{R}^d\}_{t=1}^{N}$ from $Y$
8  Split
9  $N_{\text{tr}} = \lfloor 0.7N \rfloor$, $N_{\text{val}} = \lfloor 0.2N \rfloor$,
   $N_{\text{te}} = N - N_{\text{tr}} - N_{\text{val}}$
10 $\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}}, \mathcal{D}_{\text{test}} \leftarrow$ contiguous partitions of $\mathcal{D}$
11 Fit scaler (train only)
12 For each feature $j$: $x_{\min}^{(j)} = \min_{x \in \mathcal{D}_{\text{train}}} x^{(j)}$,
   $x_{\max}^{(j)} = \max_{x \in \mathcal{D}_{\text{train}}} x^{(j)}$
13 Define $S$ by $\big(S(x)\big)^{(j)} = \frac{x^{(j)} - x_{\min}^{(j)}}{x_{\max}^{(j)} - x_{\min}^{(j)}}$
14 Transform splits
15 $\tilde{\mathcal{D}}_{\text{train}} = S(\mathcal{D}_{\text{train}})$, $\tilde{\mathcal{D}}_{\text{val}} = S(\mathcal{D}_{\text{val}})$,
   $\tilde{\mathcal{D}}_{\text{test}} = S(\mathcal{D}_{\text{test}})$
16 Window
17 $w_{\text{tot}} = w_{\text{in}} + s$
18 for $t = 1$ to $N_{\texttt{set}} - w_{\text{tot}} - w_{\text{lbl}} + 1$ for each
   $\texttt{set} \in \{\text{train}, \text{val}, \text{test}\}$ do
19     $X_t = \tilde{\mathcal{D}}_{\texttt{set}}[t : t + w_{\text{in}} - 1, :]$,
         $Y_t = \tilde{\mathcal{D}}_{\texttt{set}}[t + w_{\text{tot}} - w_{\text{lbl}} : t + w_{\text{tot}} - 1, :]$
20 $\mathcal{W}_{\texttt{set}} = \{(X_t, Y_t)\}$ and batch into datasets
21 Initialize model
22 Set $T = w_{\text{in}}$, $d = \dim(\mathcal{D})$ and build $f_\Theta$:
   $\text{LSTM}(u) \times 3 \rightarrow \text{FC}(4h) \rightarrow \text{FC}(2h) \rightarrow \text{FC}(h) \rightarrow$
   $\text{Linear}(C)$
23 Phase: Train
24 for $e = 1$ to $E$ do
25     Update $\Theta \leftarrow \Theta - \eta \nabla_\Theta \mathcal{L}_{\text{MSE}}$ on $\mathcal{W}_{\text{train}}$ (validate
         on $\mathcal{W}_{\text{val}}$)
26 Phase: Test & inverse-scale
27 Compute $\hat{Y}^{\text{tr}} = f_\Theta(X^{\text{tr}})$, $\hat{Y}^{\text{te}} = f_\Theta(X^{\text{te}})$
28 $\hat{Y}_{\text{orig}}^{\text{tr}} = S^{-1}(\hat{Y}^{\text{tr}})$, $\hat{Y}_{\text{orig}}^{\text{te}} = S^{-1}(\hat{Y}^{\text{te}})$,
   $Y_{\text{orig}}^{\text{tr}} = S^{-1}(Y^{\text{tr}})$, $Y_{\text{orig}}^{\text{te}} = S^{-1}(Y^{\text{te}})$
29 RMSE
30 $\text{RMSE}_{\text{phase}} = \sqrt{\frac{1}{N_{\text{phase}}} \sum_i (\hat{Y}_{\text{orig},i}^{\text{phase}} - Y_{\text{orig},i}^{\text{phase}})^2}$
```

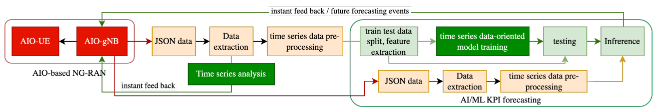

instant feed back / future forecasting events
AIO-UE
AIO-gNB
AIO-based NG-RAN
JSON data
Data extraction
time series data pre-processing
Time series analysis
instant feed back
train test data split, feature extraction
time series data-oriented model training
testing
Infrerence
JSON data
Data extraction
time series data pre-processing
AI/ML KPI forecasting

Fig. 7. Proposed KPI prediction for adaptive operation in AIO-RAN-SA.

noted that the gap observed in Fig. 6 arises from the scaling and the inclusion of κ. Specifically, because disconnections yield missing values, the constant κ introduced in (2) is a key factor in the data processing pipeline. Its presence creates a pronounced numerical gap between the disconnected level κ and the connected events. Nevertheless, the results are sufficiently accurate for binary forecasting tasks such as distinguishing connected versus disconnected states, or monitoring LOS/NLOS transitions.

In this work, the forecasting module operates in a monitoring mode only. As future work, APIs will be exposed for integrating the prediction module directly into the OAI gNB stack, thereby enabling closed-loop adaptation of RAN parameters. To this end, we have proposed a pipeline as shown in Fig. 7, that combines time-series analysis with AI/ML-based forecasting applied to the collected data.

V. Conclusions

In this paper, we proposed AIO-RAN-NTN, a blueprint that redefines internal RAN interfaces as open, NTN-capable links, with the key challenge being mobility management in NTNs. We implemented a 3GPP-compliant SA 5G testbed using the OAI platform. Our experimental results confirmed that mobility-induced channel variations degrade performance and cause reconnection delays. To address this issue, we developed an AI/ML-based forecasting pipeline for key performance indicators (KPIs), capable of anticipating KPI and SINR evolution.

References

[1] B. Agarwal et al., "Open RAN for 6G Networks: Architecture, Use Cases and Open Issues," *IEEE Communications Surveys & Tutorials*, 2025. [Online]. Available: 10.1109/comst.2025.3562429

[2] "NG-RAN Architecture Description," 3GPP/ETSI, 3GPP TS 38.401, 2022. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/138400_138499/138401/17.01.01_60/ts_138401v170101p.pdf

[3] O-RAN Alliance. (2025) O-ran specifications. [Online]. Available: https://www.o-ran.org/specifications

[4] OpenAirInterface Software Alliance. (2025) Openairinterface (oai) 5g ran project. Open-source implementation aligned with 3GPP; not a network interface. [Online]. Available: https://openairinterface.org/oai-5g-ran-project/

[5] "O-RAN Fronthaul Control, User and Synchronization Plane Specification," ETSI / O-RAN Alliance, ETSI PAS TS 103 859 v12.0.1, 2025. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/103800_103899/103859/12.00.01_60/ts_103859v120001p.pdf

[6] "Study on Enhancements for AI/ML for NG-RAN," 3GPP, 3GPP TR 38.743, 2024. [Online]. Available: https://www.3gpp.org/dynareport/38743.htm

[7] AI-RAN Alliance. (2025, February) Ai-ran alliance. [Online]. Available: https://ai-ran.org/

[8] K. A. et al., "A Comprehensive Tutorial and Survey of O-RAN: Exploring Slicing-Aware Architecture, Deployment Options, Use Cases, and Challenges," *IEEE Communications Surveys & Tutorials*, 2025. [Online]. Available: 10.1109/comst.2025.3598406

[9] T. N. Do and G. Kaddoum, "Distributed machine learning for terrestrial and non-terrestrial internet of things networks," *IEEE Internet of Things Magazine*, vol. 6, no. 4, p. 54–61, Dec. 2023. [Online]. Available: http://dx.doi.org/10.1109/IOTM.001.2300063

[10] OAIBOX, "OAIBOX 5G LAB manual." [Online]. Available: https://oaibox.allbesmart.pt/5g-lab-manual/