

RANPILOT: Making AI Functionalities Robust to Dynamic O-RAN Reconfigurations

Shiming Yu[†], Leming Shen[†], Jianing Zhang[†], Xin Li[†], Xianjin Xia[†], Yuanqing Zheng[†], Yaxiong Xie[‡]

[†] The Hong Kong Polytechnic University, [‡] University at Buffalo, SUNY

ABSTRACT

The Open Radio Access Network (O-RAN) promises unprecedented flexibility through its reconfigurable architecture and AI-driven control. However, this agility exposes a critical fragility: AI models trained on one network configuration suffer significant performance degradation after an upgrade due to dramatic data drift. The standard solution, reactive retraining, is unacceptably slow, leaving the network in a suboptimal state for tens of minutes and undermining the core benefits of O-RAN's dynamism. This paper introduces RANPILOT, the first framework to address this challenge through proactive AI adaptation. RANPILOT constructs a lightweight "virtual O-RAN" (a trace-driven emulator) to synthesize high-fidelity training data representing the post-reconfiguration state before the physical change occurs, allowing AI models to be adapted in advance. Extensive experiments on a real-world 5G testbed demonstrate that RANPILOT achieves near interruption-free AI services upon reconfiguration, reducing AI downtime by 85% to 94% against reactive baselines. By shifting the AI evolution paradigm from reactive redevelopment to proactive preparation, RANPILOT explores a digital-leadoff approach to enable robust AI in reconfigurable O-RAN deployments.

CCS CONCEPTS

• **Networks** → **Wireless access points, base stations and infrastructure**; **Mobile networks**.

KEYWORDS

Open RAN, AI RAN, Reconfiguration, Data Synthesis

ACM Reference Format:

Shiming Yu[†], Leming Shen[†], Jianing Zhang[†], Xin Li[†], Xianjin Xia[†], Yuanqing Zheng[†], Yaxiong Xie[‡]. 2026. RANPILOT: Making AI Functionalities Robust to Dynamic O-RAN Reconfigurations. In *ACM SIGCOMM 2026 Conference (SIGCOMM '26)*, August 17–21, 2026, Denver, CO, USA. ACM, New York, NY, USA, 17 pages. <https://doi.org/10.1145/3789240.3829146>

1 INTRODUCTION

Cellular Radio Access Networks (RAN) are undergoing a paradigm shift, moving away from the rigid, monolithic designs of the past toward the Open RAN (O-RAN) [5, 16, 25]. This new paradigm is founded on the principles of disaggregation, virtualization, and open interfaces, splitting traditional base station functions into

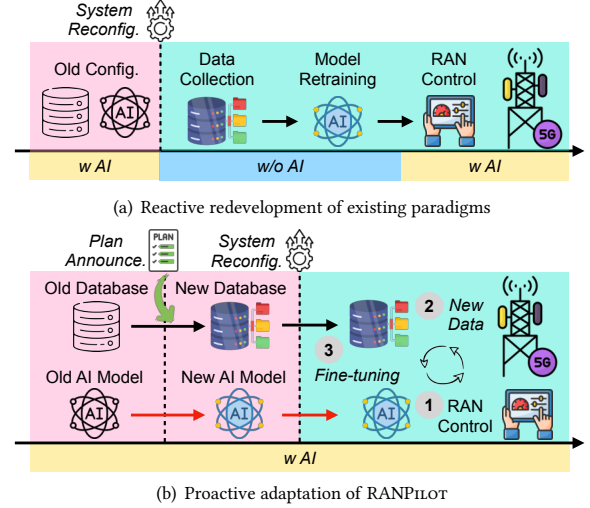


Figure 1: Illustration of AI development in O-RAN: (a) existing paradigms reactively retrain/adapt AI models after system reconfiguration; (b) RANPILOT proactively adapts AI models before system reconfiguration.

distinct, interoperable software/hardware components. This modularity, coupled with the introduction of the RAN Intelligent Controller (RIC), unlocks two transformative capabilities. First, it enables unprecedented reconfigurability, allowing network operators to dynamically add or upgrade radio units to enhance coverage and edge processing capabilities [26, 27, 52, 55]. Second, it fosters programmability, where AI-driven applications, known as xApps, can be deployed on the RIC to intelligently manage the network, optimizing functions like resource allocation [12, 41], traffic steering [42, 66], and interference mitigation [28, 40, 80]. Together, these innovations promise a future of smarter, more adaptive, and cost-efficient cellular networks [4, 10, 30, 38, 68, 75].

The promising AI functionalities in O-RAN expose a brittle truth when confronted with flexible reconfigurability: **today's AI models in O-RAN systems often break even under planned network upgrades**. For example, an AI model trained on last month's network data may struggle with this month's new configuration or tuned parameters, often yielding dramatic performance degradations, following planned upgrades. Such network reconfigurations or upgrades are necessary in routine operations and essential for AI innovations. For example, network operators need to regularly modify hardware and software components, upgrade radio units, change network policies, tune AI models, *etc*. In practice, however, such necessary changes can invalidate previously well-trained AI models. Our measurements on a live 5G O-RAN testbed underscore the severity of this problem (§3.2). After a common cell-addition, a pretrained Quality of Experience (QoE) predictor fails to anticipate



This work is licensed under a Creative Commons Attribution 4.0 International License. *SIGCOMM '26*, August 17–21, 2026, Denver, CO, USA

© 2026 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-2467-1/26/08.

<https://doi.org/10.1145/3789240.3829146>

sharp throughput fluctuations, causing video streaming to suffer from a high frequency of playback stalls (Figure 3). Simultaneously, an anomaly detector is flooded with false alarms, misclassifying benign inter-cell interference from the new cell as a malicious event (Figure 5).

The culprit is dramatic data drift caused by network reconfigurations and component upgrades: the reconfiguration introduces novel RAN interactions and traffic dynamics that the AI model was never trained on. We quantify this drift, finding that while most traffic patterns remain similar, a critical 8% of post-reconfiguration traffic shows a significant deviation (maximum cosine similarity < 0.7) from the original training data (Figure 6). The model's performance collapses precisely on this drifted data, with the QoE predictor's error spiking above 10 Mbps, demonstrating that the AI model fails at the most critical moments when it must understand the network's new operational logic.

Existing solutions (e.g., reactively collecting new data to perform model adaptation after deployment as illustrated in Figure 1(a)) are agonizingly slow and operationally untenable for cellular networks. Our measurements show that adapting a QoE prediction model via continuous learning takes 29 minutes to regain its original performance, while an anomaly detection model requires 20 minutes (Figure 7). It took almost the same amount of time to train new models from scratch (31 and 21 minutes, respectively), indicating the data drift is too severe for simple fine-tuning. This prolonged period of suboptimal performance (effectively *AI outage*) negates the core O-RAN promise of agility and flexibility: If every dynamic upgrade incurs a half-hour of degraded service and potential user complaints, the flexibility comes at too high a cost, rendering cellular operators reluctant to adopt AI models or deliver new AI functionalities via system upgrades.

Instead of reactively fixing AI after major network reconfigurations or upgrades, *can AI plan for such changes and adapt beforehand?* We present RANPiLOT, a new framework for proactive O-RAN AI adaptation (Figure 1(b)). Its design rests on three pillars (§4): (1) At its core, RANPiLOT builds a *lightweight virtual O-RAN* that emulates the planned network changes with sufficient fidelity to synthesize training data for the future scenarios ahead of time. This differs from a heavyweight, ray-tracing-based physical-layer reconstruction [33, 70]. Instead, RANPiLOT targets data- and control-level fidelity by focusing on the structural network behaviors and RAN unit interactions that drive post-reconfiguration data drift. (2) RANPiLOT uses a dedicated meta-learning strategy to intelligently augment the synthetic KPM data that helps the model generalize to diverse operational conditions; (3) RANPiLOT rapidly closes the remaining simulation-to-reality gap by prioritizing important real-world KPMs for incremental tuning.

Our extensive evaluation on a live 5G O-RAN testbed [78] shows that RANPiLOT achieves near interruption-free AI services through reconfigurations, *reducing AI downtime by 85–94% compared to reactive retraining baselines*. This transforms the adaptation process from a lengthy recovery into a swift, final calibration. For instance, RANPiLOT prepares anomaly detection models that are ready in just 1.3–2.4 minutes and a resource allocation model that achieves near-Oracle performance immediately, resolving minor discrepancies within four minutes of deployment. From the moment of deployment, RANPiLOT-prepared models operate at full accuracy;

our anomaly detector achieves 89.1–97.5% accuracy from the start, while the Vanilla model languishes at 30–50%. This is made possible as our synthesized data faithfully reproduces critical network dynamics of the new interaction patterns among reconfigured RAN units, such as throughput fluctuations, that are essential for robust AI but overlooked by benchmarks. By enabling AI to evolve in lockstep with network updates, RANPiLOT closes a critical gap in today's O-RAN architecture. In summary, this paper makes the following contributions:

- For the first time, we draw attention to the problem of AI robustness in dynamic O-RAN reconfigurations, identifying its unique challenges and implications (§3).
- We propose RANPiLOT, a new framework that supports the evolution of AI functionalities prior to physical RAN reconfiguration. RANPiLOT marks a paradigm shift from reactive remedy to proactive adaptation for AI functionalities in reconfigurable O-RAN (§4).
- We design, implement, and evaluate RANPiLOT on a realistic 5G RAN testbed across various use cases. Results indicate that RANPiLOT can deliver performant and robust AI functionalities and support diverse applications (§5–§6).

Ethics: This work does not raise any ethical issues.

2 RELATED WORK

AI functionalities in O-RAN. AI for cellular networks has gained substantial attention in recent years [11, 58, 62, 76, 93]. The rise of open and programmable control in O-RAN has further accelerated this trend, enabling developers to integrate AI/ML-driven mechanisms into the RAN for tasks such as network slicing [12, 15, 18], traffic steering [42, 66], interference mitigation [40, 94], anomaly detection [65, 80, 88, 89], and automated control [3, 23, 31, 36, 51]. The AI functionalities have been undeniably promising for smarter, more adaptive NextG cellular networks [4, 38].

Service degradation after RAN reconfigurations. Early measurement studies of large operational networks [46, 53, 54, 56] revealed that configuration changes can induce subtle yet persistent performance degradations. To reduce user-visible impact, previous work focused on minimizing service disruption through carefully coordinated, network-wide upgrades [73, 91]. More recent efforts have further enhanced system robustness and achieve near-zero downtime during network updates [39, 45, 74, 90].

With the emergence of O-RAN, however, AI-driven control and optimization have become integral to RAN operation, introducing a new vulnerability: learned models themselves can degrade under frequent and flexible reconfigurations. Existing AI functionalities remain largely rigid and tightly coupled to static configurations, making them particularly sensitive to configuration shifts [9].

AI robustness against data drift. Prior work [7] and our measurements (§3.2) show that O-RAN reconfigurations can induce severe data drift that quickly invalidates deployed AI models. The standard remedy is to adapt models to new data distributions through continuous or transfer learning [14, 37, 85, 87], and recent studies have attempted to apply these techniques to cellular systems [13, 61, 63]. However, unlike conventional AI settings, cellular networks face a practical challenge: collecting real-world data is slow,

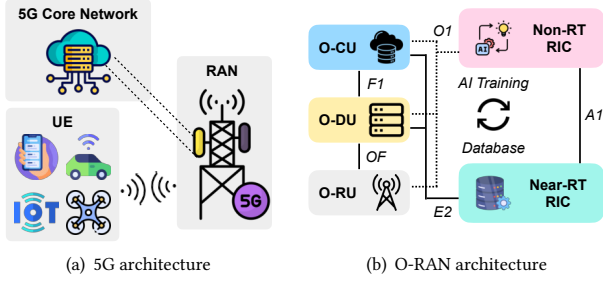


Figure 2: Background of 5G Open RAN: disaggregated RAN units integrated with AI/ML functionalities.

resource-intensive, and often requires lengthy over-the-air measurements [9]. Motivated by recent advances in synthetic data generation [17, 20, 29, 64, 70], we explore whether AI models can learn ahead of time using synthetic traces tailored to emulate upcoming network reconfiguration, thereby improving robustness under dynamic O-RAN reconfigurations.

AI-based network management and model adaptation. Networking systems adopt AI for network management and model adaptation across diverse domains. In wide-area networks, AI-driven traffic engineering [6, 69] aims to enhance resilience under dynamic traffic demands and topologies. In data center networks, xWeaver [86] leverages AI to adapt topology decisions to evolving traffic patterns. Similarly, cellular networks increasingly rely on AI models to execute complex operational tasks [80, 93]. However, AI in production networks suffers from severe performance degradation under planned or unplanned system changes, calling for efficient model adaptation mechanisms [50]. To tackle this degradation, LEAF [49] characterizes concept drift in large-scale cellular infrastructure and mitigates accuracy drops through a combination of targeted retraining, data forgetting, and sample oversampling. Argus [92] continuously monitors service anomalies and adapts AI models on the fly to absorb data distribution variations. Unlike these works, RANPilot specifically addresses data drift in dynamic O-RAN reconfigurations and proposes a proactive paradigm that adapts AI models using synthetic data.

3 BACKGROUND AND MOTIVATION

3.1 O-RAN and AI Functionalities

5G Networks. As illustrated in Figure 2(a), 5G networks consist of three primary components. *User Equipment (UE)* includes devices such as smartphones, IoT devices, and connected vehicles that access the network. *Radio Access Network (RAN)* provides wireless connectivity, enabling communication between UEs and the network infrastructure. *5G Core Network (5GC)* manages essential functions like user authentication and mobility management, while also routing user data between the RAN and backhaul networks.

O-RAN vs. Traditional RAN. By decoupling traditional RAN hardware and standardizing interfaces, O-RAN fosters a multi-vendor ecosystem that promotes reconfigurability and innovation. As illustrated in Figure 2(b), O-RAN disaggregates RAN functions into three logical components: O-Radio Unit (O-RU), O-Distributed Unit (O-DU), and O-Central Unit (O-CU). *O-RU* transmits and receives radio signals over the air interface, and implements lower PHY functions (e.g., FFT/iFFT). *O-DU* executes higher PHY functions

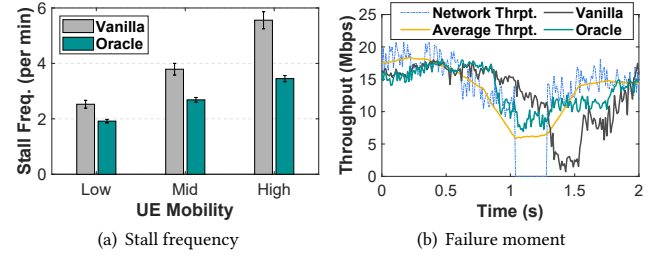


Figure 3: QoE prediction after RAN reconfiguration: (a) Overall video streaming stalls; (b) Throughput fluctuation in one of the prediction failure moments.

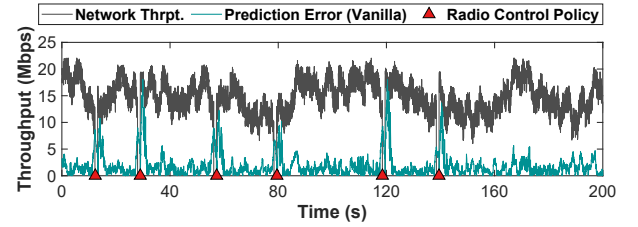


Figure 4: The failure of Vanilla QoE prediction model aligns with the execution of radio control policies.

(e.g., scrambling, modulation), MAC scheduling, and RLC operations. *O-CU* handles higher-layer functions (e.g., RRC signaling, mobility management, and PDCP processing). It orchestrates UE behavior, manages handovers, and enforces QoS policies. O-RAN further integrates RAN Intelligent Controller (RIC) to coordinate AI functionalities. O-RAN collects and stores the Key Performance Metrics (KPMs) of RAN components.

Need for robust and interruption-free AI services. AI services are fundamental to network control and optimization. The flexible reconfigurability and stringent downtime requirements [45, 73] pose an urgent demand for robust and interruption-free AI functionalities, particularly during the initial phase following RAN reconfiguration. In other words, AI functionalities are expected to maintain consistent performance with ideally zero service downtime after major reconfiguration as well as system upgrades.

3.2 RAN Reconfiguration and Data Drift

To demonstrate the fragility of current AI approach and trained models even facing planned reconfiguration and system upgrades, we conduct a series of preliminary studies (§5-§6). We measure the performance degradation of AI functionalities following routine reconfiguration operations and updates. We test two standard AI functionalities with representative model architectures and training methods:

- **QoE prediction and serving** [93] predict QoE based on historical traffic patterns. Accurate predictions enable real-time, proactive adjustment of application and network parameters to maintain high user satisfaction, which is especially critical for latency-sensitive and immersive applications like AR/VR and cloud gaming [8, 19, 34, 59, 60].
- **Anomaly detection and localization** [80] adopt a distribution learning model to analyze RAN KPMs to detect system failures and irregularities. Accurate and timely detections enable automated maintenance workflows and rapid recovery from network

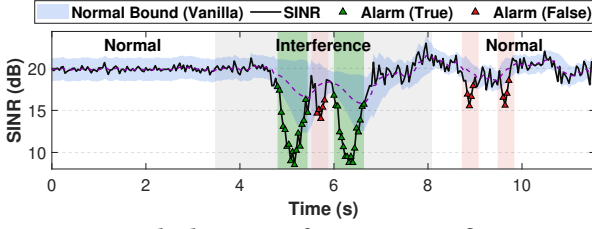


Figure 5: Anomaly detection after RAN reconfiguration: normal traffic frequently crosses the normal bound of Vanilla model, raising extensive false alarms.

anomalies, thereby reducing operating expenses and enhancing reliability [44, 88, 89].

We tested various system reconfigurations including both hardware and software changes and upgrades. To make our discussion concrete, we start with one routine system upgrade, *i.e.*, adding a cell tower for better coverage.

AI Service Interruption after Routine Updates. After a reconfiguration, video streaming quality degrades significantly when managed by the Vanilla QoE predictor (*i.e.*, model before reconfiguration), exhibiting a high frequency of playback stalls that worsen with UE mobility, as shown in Figure 3(a). The root cause is that the current AI approach does not train the model to cope with the sharp data drift caused by major network reconfiguration. As a result, the Vanilla model lags behind the actual throughput as shown in Figure 3(b), leading to performance degradation and affect user experience.

By cross-checking these failures with RAN operation logs, we discovered a notable pattern: the most failures for the Vanilla model coincide with the updates to radio control policies, as shown in Figure 4. These newly integrated policies (*e.g.*, possible handover to/from the new RAN unit) cause network dynamics that were unseen by the Vanilla model, rendering the model ineffective.

Our measurements on anomaly detection unveil similar limitations of current AI approach and trained models. Figure 5 illustrates that following a system reconfiguration, the Vanilla anomaly detection model frequently misclassifies benign network behavior (*e.g.*, new patterns of inter-cell interference) as malicious anomalies. These subtle fluctuations are typically caused by the reconfigured topology but rarely seen in the collected data.

In both cases, we observe that the AI models cannot stay effective and robust throughout major reconfigurations and system upgrades.

Impact of RAN Reconfiguration on Data Distribution. We further analyze the data distribution variation of network traffic induced by RAN reconfiguration. To quantify data distribution variations, we adopt a unique metric, *maximum cosine similarity*, which assesses the affinity of each new KPM sample to the training set used by the Vanilla model. Specifically, for each new KPM sample, we compute its cosine similarity with all training samples and select the maximum value as a proxy, representing the closest distance between the KPM sample and the training set [77]. This approach can eliminate the influence of the inherently high variance of cellular traffic. To benchmark the impact of reconfiguration, we measure two 20-minute traffic segments: one before and one after the reconfiguration. We segment each trace into 2 s intervals and compute their similarities to the training set.

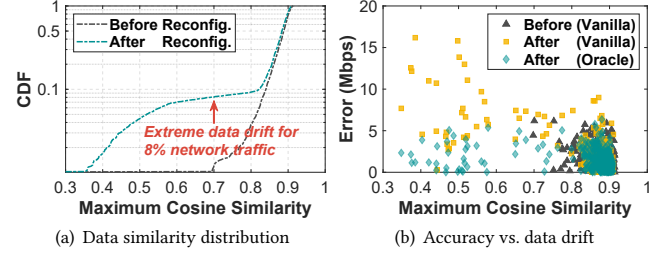


Figure 6: Data drift from RAN reconfiguration: (a) network traffic experiences large data distribution drifts; (b) Vanilla model degrades sharply on the drifted data.

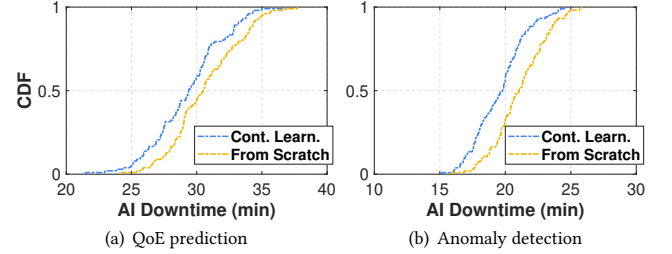


Figure 7: End-to-end adaptation time of different AI functionalities, during which the reconfigured RAN suffers from suboptimal performance.

Our empirical analysis shows that, although overall data similarity remains broadly stable after reconfiguration, a notable portion of network traffic diverges significantly from the training distribution (see Figure 6(a)). These low-similarity segments are unrepresented in the original training set and invalidate the Vanilla model. As illustrated in Figure 6(b), throughput predictions become highly unreliable in these low-affinity regions. These results indicate that RAN reconfigurations introduce substantial data drift, undermining the robustness of pretrained, static O-RAN AI models.

Limitations of Existing Solutions. Data drift is a well-known challenge in AI. Moderate data drift can be addressed by continuous learning [14, 37, 85, 87]. However, this reactive approach is untenable in O-RAN due to two unique challenges: 1) *Lack of post-reconfiguration data.* Adapting AI models requires collecting new training data in target domains (*i.e.*, after reconfiguration), which is unavailable until the reconfiguration is enforced. 2) *Stringent downtime requirement.* Cellular networks impose a stringent service downtime requirement [45, 73]. Continuous learning [85] generally takes a long time to collect sufficient post-reconfiguration data for fine-tuning models. During this process, the AI models inevitably suffer from drastic performance degradation, rendering cellular operators reluctant to adopt AI models or deliver new AI functionalities via system upgrades.

To quantify the *reconfiguration tax*, we measure the time required to adapt the models. Specifically, we consider two adaptation approaches: 1) updating the Vanilla models via continuous learning [82], and 2) collecting new KPM traces after reconfiguration and training new models from scratch. We measure the *AI downtime*, which is defined as the duration from the activation of RAN reconfiguration till the adapted AI model achieves 95% of its best performance (denoted as Oracle). The two models require 29 and 20 minutes for continuous learning, respectively (Figure 7). Notably,

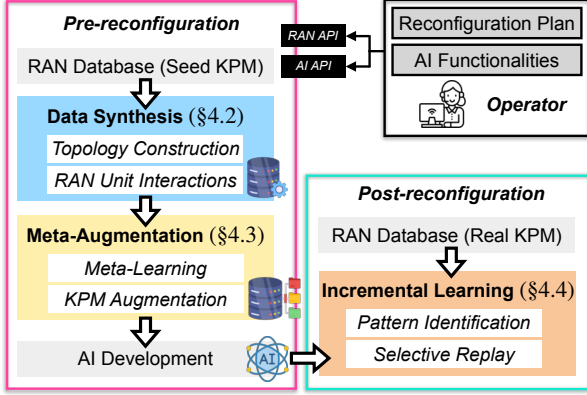


Figure 8: System overview of RANPILOT.

due to the time-consuming data collection and continuous learning required to handle extreme data drift, the adaptation efforts are almost the same as developing a new model from scratch, which takes about 31 and 21 minutes, respectively. These adaptation methods lead to extended AI service downtime and prolonged periods of suboptimal network performance, negating O-RAN’s flexible reconfigurability.

In summary, we have two key observations:

- Major reconfiguration and system upgrades introduce extreme data drift, obsoleting AI functionalities developed on legacy data.
- Existing continuous learning or fine-tuning solutions lead to unacceptable service downtime with performance degradation.

4 RANPILOT DESIGN

4.1 System Overview

Is it possible to develop robust AI models that can look ahead and accommodate RAN system reconfiguration without service interruption? A fundamental obstacle is that AI adaptation requires training data, yet real post-reconfiguration data is unavailable prior to deployment, and collecting it afterward is prohibitively slow. To address this gap, we propose RANPILOT, a novel framework that proactively adapts AI models ahead of RAN reconfigurations using synthetic data. By enabling AI adaptation before reconfiguration, RANPILOT mitigates the data drift that could break the deployed models during major system upgrades.

However, effective data synthesis and AI adaptation for dynamic O-RAN reconfigurations face three key challenges: (i) *Capturing System Changes*. O-RAN introduces disaggregated units and programmable control policies with highly flexible and dynamic reconfigurations. These changes alter traffic patterns and even system topology, making it difficult to synthesize data that faithfully reflects underlying network behaviors. (ii) *Ensuring Data Generalization for AI Robustness*. To adapt the AI models with sufficient robustness, synthetic data must comprehensively cover diverse operational scenarios and network dynamics. (iii) *Bridging the Reality Gap*. Even the best synthetic data cannot fully replicate real-world nuances such as environmental dynamics. Thus, post-reconfiguration, we must efficiently use real data to close this gap as quickly as possible.

Figure 8 illustrates the overview of RANPILOT. To faithfully emulate O-RAN reconfigurations, RANPILOT builds a virtual O-RAN that abstracts disaggregated RAN units, their interactions, and

control policies (§4.2). Driven by real network traces and expert knowledge, this virtual O-RAN emulates diverse reconfiguration changes while preserving essential system behaviors. To improve the AI robustness, RANPILOT augments the synthesized data from virtual O-RAN with rich operational scenarios and network dynamics (§4.3). A modified meta-learning module [22] learns how to augment O-RAN KPM traces and continuously incorporates synthesized data from unseen configurations into a local database, enabling progressive refinement and improved generalization. Finally, to rapidly close the simulation-to-reality gap after deployment, RANPILOT detects biased traffic patterns and assigns priority weights for selective replay, enabling efficient incremental training using real KPM data (§4.4). When operators plan an O-RAN reconfiguration, they notify RANPILOT of the target RAN configuration and desired AI functionalities via RAN and AI APIs; RANPILOT then synthesizes and augments KPM data to adapt the AI models, which are subsequently refined online after deployment to ensure accuracy under live network conditions.

4.2 Building Virtual O-RAN

4.2.1 Insight & Solution. Proactive AI adaptation requires future state data before a physical reconfiguration occurs. To synthesize effective data for AI adaptation, our design is grounded on three core principles: (1) *Data and control fidelity over PHY exactness*. Instead of simulating every waveform, we abstract radio units and model their interactions at the control and scheduling layers. This captures the system-level variations that dominate KPM drift and AI performance. (2) *Interface- and policy-awareness*. We explicitly emulate open interfaces and their contention dynamics, along with operator-defined radio control policies. (3) *Trace-driven realism*. We bootstrap the synthesis with real KPM traces from the current RAN, then perturb them to reflect new topology, mobility, and interference. As illustrated in Figure 9, our design follows three stages: building a static skeleton of the target O-RAN configuration, generating O-RAN traffic, and synthesizing O-RAN unit interactions. The result is a **lightweight virtual O-RAN** – a trace-driven emulator that generates post-reconfiguration KPM data for downstream AI adaptation. It departs from PHY-exact digital twins [33, 70], and instead captures system-level variations that ultimately reshape KPM traces after reconfiguration.

4.2.2 Workflow & Input/Output Schema. The virtual O-RAN is a trace-driven emulator built from softwarized RAN units. It takes seed KPM traces as input and allows planned reconfigurations to be applied to the software units. In addition, we explicitly model external factors that affect KPM distribution (e.g., RU placement, wireless channel, O-RAN interface contention) to align the emulator with real-world physical variations. The following control knobs support diverse RAN reconfigurations (detailed in Table 4): (1) *Topology*, which represents the target O-RAN as a directed graph. The nodes consist of abstracted CU, DU, and RU, while the edges represent the F1 (CU–DU) and Open Fronthaul (DU–RU) interfaces. The adjacency encodes hierarchy and enables per-interface latency/bandwidth effects to propagate into synthesized KPMs; (2) *Cell configuration*, which defines cell identities and adjacent cell topologies used for tracking handovers; (3) *RAN unit configuration*, which modifies softwarized RAN (RU, DU, CU) variables; (4) *xApp*, which executes programmable control logic. For example, cell addition

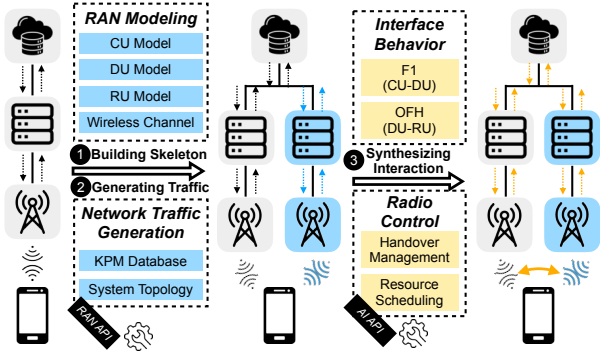


Figure 9: Virtual O-RAN construction based on real traces and abstracted RAN units.

can be emulated by changing the topology, cell configurations, and RU variables; PRB scheduling or handover policy upgrades can be emulated by modifying specific control logic in the xApp and DU/CU variables.

4.2.3 Building O-RAN Skeleton. To emulate a reconfigured O-RAN system, we first build its structural skeleton by initializing software RAN units at each layer within the input RAN topology. Rather than modeling the full protocol stack, we utilize functional abstractions that capture the essential RAN unit behaviors that drive KPM drift. In our design, RU represents a lightweight PHY-layer abstraction, handling signal propagation by executing commands from the upper layer and reporting PHY KPMs; DU serves as a control and buffering entity, and its functions include aggregating metrics from lower layers and performing basic scheduling; CU operates as a policy-driven controller, tasked with managing UE state and enforcing radio resource control. This layered skeleton (1 in Figure 9) allows flexible construction of a target O-RAN system with various radio unit abstractions forming different topologies.

4.2.4 Generating O-RAN Traffic. We then generate synthetic KPM traces that activate the static O-RAN skeleton into a live virtual system. (1) **KPM-driven bootstrapping.** We seed the virtual O-RAN with a KPM database that spans radio resources, UE channel quality, and QoS metrics (detailed in Table 3). The KPMs are grounded on the current *real* RAN operational states and provide the statistical priors for our synthesis. (2) **Traffic reconstruction and generation.** We generate O-RAN traffic with a bifurcated strategy. For unchanged RAN units, we directly reconstruct traffic by replaying real-world KPM traces to maintain absolute baseline fidelity. For newly added or reconfigured units, we reuse behavioral priors from real traces but adjust them based on the new topology and network dynamics. This ensures realistic physical coupling (e.g., inter-cell interference) and produces plausible KPMs that reflect the target topology. (3) **PHY consistency via wireless modeling.** We integrate a lightweight wireless channel model as an important external factor that links the software emulator to real-world PHY impacts of changed RUs, ensuring the generated KPMs remain physically consistent. Instead of complex ray-tracing [33], we account for both mobility-induced and stochastic channel dynamics [32]:

$$\text{SINR}(t) = P_{\text{tx}} - PL(d(t)) + X_{\sigma} + F(t) - N_0 \quad (1)$$

with path loss $PL(d(t))$, shadowing X_{σ} , and fading $F(t)$; mobility and environment enter via $d(t)$ and $F(t)$, respectively (detailed

in §A.4). To map input traces and configurations to the wireless channel model, RANPilot executes a trace-driven parameter tuning procedure: Specifically, we tune SINR parameters to match seed KPM traces, and retain SINR environment parameters for the reconfigured RU while adjusting distance/mobility parameters to simulate PHY changes. Overall, this real-KPM-driven generation (2 in Figure 9) outputs basic synthetic KPM traces that the AI models and RAN controllers can operate on.

4.2.5 Synthesizing O-RAN Unit Interactions. We next synthesize the essential RAN unit interactions that reshape the network behaviors and drive data drift. (1) **Open-interface behaviors.** While software RAN units can be directly updated within the emulator, the open-interface behaviors that affect KPM distribution must be considered. Specifically, we integrate an Open Fronthaul (OFH) interface contention model as an important external factor that replays O-RAN interface behavior. When multiple RUs share a constrained OFH, concurrent packet arrivals can lead to collisions [48, 80]. To map input traces and configurations to the OFH contention model, we calculate the packet loss probability P_{loss} as a function of the aggregate fronthaul throughput S , derived by the average throughput per RU times the number of contending RUs. As S approaches the interface capacity C , the probability of collision increases: $P_{\text{loss}}(S) = 1 - e^{-\lambda(S/C)}$. We use $\lambda = 2$, calibrated from empirical measurements. These collisions trigger higher-layer retransmissions, which we derive into observable KPM impacts: increased latency due to backoff timers and a proportional degradation in effective UE throughput. (2) **Radio control policies.** O-RAN decouples control policies through programmable xApps [5, 71]. These customizable policies [1] are not mere configuration details. They act as first-order drivers of data drift during reconfigurations and reshape traffic patterns. This AI-in-the-loop emulation approach enables the model to approximate the empirical mapping between policy updates and resulting network behaviors. For AI models that do not directly affect network policies or control loops, the new models are reconfigured in virtual O-RAN for emulation and model adaptation. The interaction synthesis (3 in Figure 9) produces essential but underrepresented network behaviors beyond the Vanilla KPM database, mitigating data drift in the post-reconfiguration state.

4.3 Harnessing Network Dynamics

4.3.1 Insight & Solution. The virtual O-RAN can effectively generate target-state KPM traces for new RAN configurations, but network dynamics and cross-scenario variability can be under-represented. Moreover, the limited quantity of synthetic KPM traces is insufficient to train robust AI models and can lead to overfitting. To address these issues, augmenting the synthetic KPM data is a promising approach to improving both quantity and quality. Unfortunately, hand-crafted augmentation (e.g., random jittering or resampling) distorts temporal dependencies and cross-metric correlations that downstream AI relies on. To tackle this problem, we propose a *continuous meta-augmentation* paradigm that first teaches AI models to learn how to augment KPM data and continuously evolve by dynamically maintaining a configuration database as operators plan additional reconfigurations. Figure 10 illustrates the workflow of our *continuous meta-augmentation* paradigm. Given the synthetic KPM data (§ 4.2), the paradigm generates

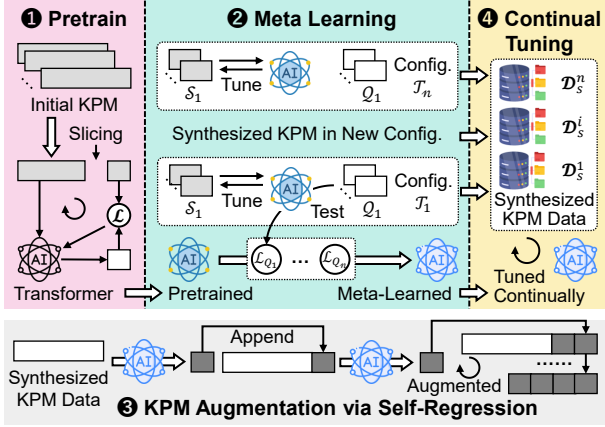


Figure 10: Learning-to-augment paradigm to harness network dynamics under evolving configurations.

an augmented KPM dataset with enhanced quantity and quality through four steps: **pretraining** that captures network dynamics, **meta-learning** that learns how to augment, **KPM augmentation**, and **continual tuning** in response to emerging O-RAN reconfigurations. This approach captures the intricate spatio-temporal dependencies inherent in multi-dimensional KPM data.

4.3.2 Capturing Network Dynamics. We pretrain a lightweight, six-layer Transformer [84] on synthetic KPM data to serve as a foundation model. Given a KPM sample, this model can predict subsequent KPM segments, which capture diverse network dynamics that are often deeply buried within KPM data. Specifically, we first segment each KPM sample $\mathcal{K}$ into a set of slices $\mathcal{K}_s = \{k_1, k_2, \dots\}$ by truncating the data at varying indices. The KPM data following each slicing point serves as the ground-truth label. We then feed $\mathcal{K}_s$ into the model to predict the next KPM trace and train the model by minimizing the loss between the predicted KPM and the ground truth. This method has two main advantages: (1) the entire training process is autoregressive and self-supervised, eliminating the need for manual labeling; (2) by generating multiple slices from varying truncation points, the model can effectively learn diverse network dynamics even from a limited quantity of synthetic KPM data.

4.3.3 Learning to Augment. We redirect the pretrained model $\mathcal{M}$ to learn how to augment KPM traces across diverse RAN configurations via meta-learning [22]. As new RAN configurations emerge, the synthesized KPM data is stored in a progressively updated database $\mathcal{D}$. For each configuration $\mathcal{T}_i$ in $\mathcal{D}$, we split the KPM traces into a support set $\mathcal{S}_i$ and a query set $\mathcal{Q}_i$. We then perform a meta-update: a copy of $\mathcal{M}$ is trained on $\mathcal{S}_i$ and evaluated on $\mathcal{Q}_i$. The evaluation losses across all configurations $\{\mathcal{T}_1, \mathcal{T}_2, \dots\}$ are summed to update the original $\mathcal{M}$. This alternating optimization process is repeated for several epochs until $\mathcal{M}$ converges. Such a meta-trained model $\hat{\mathcal{M}}$ can effectively generalize its KPM augmentation capabilities across various configurations.

4.3.4 KPM Augmentation. To perform augmentation, we segment KPM traces from each configuration $\mathcal{T}_i$ into multiple slices and leverage the meta-trained model $\hat{\mathcal{M}}$ to generate subsequent KPM traces in an autoregressive manner (§ 4.3.2). The augmented KPM data $\mathcal{D}^a = \{\mathcal{D}_1, \mathcal{D}_2, \dots\}$, along with the sliced synthetic data $\mathcal{K}_s$,

are used to retrain the O-RAN AI models prior to physical RAN reconfigurations.

4.3.5 Continual Tuning. Unlike conventional meta-learning methods that rely on static datasets, we implement a dynamic, continuous meta-learning based on the progressively maintained database $\mathcal{D}$. An intuitive approach would be to continuously fine-tune the augmentation model $\hat{\mathcal{M}}$ using the entire $\mathcal{D}$ as new RAN configurations arrive. Nonetheless, we observe that this strategy leads to *intransigence*. As the database expands, early-arriving configurations are repeatedly sampled during training, causing the model to overfit to historical data and overshadow recent samples. Consequently, the model exhibits suboptimal adaptability to new RAN configurations. To mitigate this, we adopt a linear decay weighting scheme during continual tuning. Specifically, during loss calculation, we assign a decay weight of $1/n$ to each RAN configuration, where n represents the cumulative number of times that configuration has been used for tuning. We select a $1/n$ decay rate, as faster-decaying functions penalize historical data too aggressively and can cause catastrophic forgetting on early RAN states. The harmonic $1/n$ decay satisfies standard stochastic approximation conditions and allows the model to continuously adapt to emerging configurations while retaining historical network behaviors. Algorithm 1 shows the detailed process.

4.4 Closing Simulation-to-Reality Gap

4.4.1 Insight & Solution. Admittedly, the virtual O-RAN and meta-augmentation may not be able to cover all possible dynamics in real diverse scenarios. The real-world nuances could arise from local implementation details, user behavior, and environmental factors that cannot be fully captured. An intuitive solution is to adopt incremental learning [57] that replays useful data to reinforce AI models. Unfortunately, traditional incremental learning cannot be applied in O-RAN because useful KPM patterns (capturing data drift) are sparse and naive replay is dominated by typical KPM patterns, which could lead to catastrophic forgetting. To remedy this, we apply a new incremental learning strategy with selective replay that detects useful, drift-carrying patterns on the fly and assigns them higher priority during incremental tuning to prevent forgetting. This allows RANPILOT to adapt to real-world data and close the simulation-to-reality gap.

4.4.2 Module Breakdown. (1) Useful pattern identification. We first define the *dominant pattern* K_d as the KPM traces that produce normal inference results from O-RAN AI models (e.g., KPM that does not trigger alarms in anomaly detection). Then, we adopt a sliding window of length l to sequentially slice the incoming KPM traces. For each KPM slice, we quantify its data distribution affinity to the *dominant pattern* by calculating the Kullback-Leibler divergence (KLD) [47] between the two KPM traces. A larger KLD value indicates a greater disparity in data distributions between the *dominant pattern* and the current KPM slice. The *useful pattern* is then identified as the KPM slice whose KLD value exceeds a predefined threshold τ . **(2) Selective replay based on weighted priority** assigns different priority weights to distinct KPM slices for tuning. The underlying rationale is that, as *useful patterns* in incoming KPM traces typically yield higher KLD values, we assign them higher priority for incremental training, while assigning lower

priority for *dominant patterns*. Specifically, at regular intervals, we collect t incoming KPM slices $K_1, K_2, \dots, K_t$. The priority weight p_i of K_i is then determined to be equal to its computed KLD value. Next, the top- b KPM slices with the highest priority are selected and stacked into a KPM buffer of size b . Once the buffer is full, we incrementally train the O-RAN AI model using the priority weight of each KPM slice during loss calculation. Note that we set $t \gg b$ to ensure that the collected KPM slices encompass a sufficient variety of traffic patterns, with the buffer containing more *useful patterns* than *dominant patterns*. As such, both types of KPM patterns will be replayed for incremental training.

5 IMPLEMENTATION

Testbed Setting. We implement RANPiLOT on the open-source srsRAN stack [78], utilizing software-defined radios (SDRs) and commercial UEs, as illustrated in Figure 11. The UEs include OnePlus 8T and Xiaomi 13 Pro smartphones equipped with programmable SIM/USIM cards (sysmoISIM-SJA2 [81]). The RAN stack is hosted on a workstation running Ubuntu 22.04.1 LTS, equipped with an Intel Xeon(R) E5-2620 v4 CPU (32 GB RAM) and an NVIDIA RTX 4090 GPU (24 GB VRAM). Two USRP X310 SDRs serve as the RUs. We deploy Open5GS [67] on the same workstation to provide core network functionality. For control-plane support and AI integration, we utilize the OSC RIC framework [72]. The KPM augmentation model contains six Transformer blocks with a batch size of four. Pre-training the KPM augmentation model on a 1-hour KPM database takes approximately 18 minutes, and continually tuning it on a 10-minute KPM database after each reconfiguration takes around 3 minutes. The GPU VRAM consumption is roughly 1.5 GB.

Integration with O-RAN Architecture. RANPiLOT enforces strict compute isolation to protect time-sensitive control loops in O-RAN operation. Only the lightweight KPM collection module is implemented as an xApp within the Near-RT RIC, while the heavy data synthesis and augmentation modules are deployed entirely in the Non-RT RIC. In addition to the basic KPMs provided by srsRAN, we integrate custom hooks [24] into the stack to capture more granular and informative KPMs every 40 ms. Operators configure RANPiLOT via open APIs, enabling it to leverage the realistic database to emulate reconfigured RAN and develop diverse AI functionalities. Once the new configuration is online, the updated AI models are immediately activated to provide robust network control and optimization.

6 EVALUATION

The key takeaways of our evaluations are:

- RANPiLOT synthesizes KPM traces that faithfully capture system dynamics and data drifts across various reconfiguration types and deployment environments (§6.2).
- Across three downstream AI use cases, RANPiLOT substantially improves AI robustness under diverse O-RAN reconfigurations, enabling superior plug-and-play performance and dramatically reducing AI downtime against baselines (§6.3).
- RANPiLOT incurs modest and practical pre-deployment overhead across a wide range of reconfiguration scenarios and system topologies (§6.4).

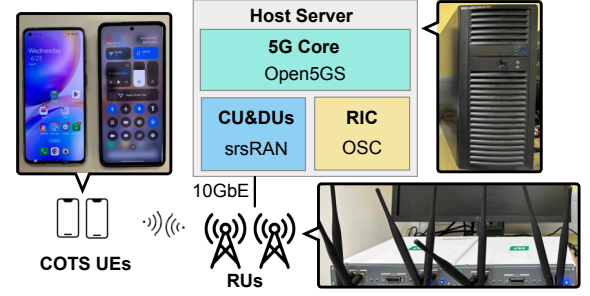


Figure 11: 5G testbed with COTS UEs and SDRs.

6.1 Methodology

Setting. We collect over 30 million KPM measurement points in our testbed, each point includes >60 metrics with details shown in Table 3. The collected database covers various channel conditions (e.g., indoor and outdoor, low and high mobilities). All results presented in this section are based on real-world KPM traces and over-the-air experiments.

Baselines. We compare RANPiLOT against various representative baselines along two distinct dimensions:

- **Data synthesis:** We compare with (1) *OWDT* [33], a SOTA O-RAN digital twin that employs ray tracing to model physical-layer signal propagation and UE mobility; and (2) *Calibrated Noise Augmentation*, which synthesizes data by injecting random Gaussian noise and temporal shifting, with the variance calibrated from the seed KPM trace.
- **AI adaptation:** We compare with (1) *TenaxDoS* [13], an online learning framework that incrementally fine-tunes models using new KPM data after reconfiguration, serving as a standard continual learning benchmark; and (2) *CORAL* [79], a domain adaptation method using a correlation alignment layer to minimize domain shift.

Variants of RANPiLOT. To assess the contribution of each module, we construct distinct ablation variants by disabling specific modules within RANPiLOT, including Virtual O-RAN (V.), Meta-Augmentation (M.), and Incremental Learning (I.). We also implement RANPiLOT-Lite, which replaces the transformer with a standard LSTM to test the performance of simpler sequence generators.

6.2 Synthetic Data Fidelity

To quantitatively validate the fidelity of the synthesized traces, we evaluate the data similarity across diverse reconfiguration types and physical deployment environments. Following the metric defined in §3.2, we quantify data fidelity by measuring the maximum cosine similarity distribution of 2 s intervals across 10 minutes of synthesized KPM traces, utilizing a post-reconfiguration trace from the operational *Live RAN* as the ground-truth reference. In these experiments, the UEs randomly alternate traffic loads between video streaming, file downloads, and iperf UDP tests to generate realistic, dynamic workloads. Since the Incremental Learning module does not contribute to the data synthesis, we use RANPiLOT to represent the ablated variant without the Meta-Augmentation module in this subsection for simplicity. Additionally, we plot the similarity distribution of a temporally independent trace from the post-reconfiguration *Live RAN* to provide a statistical benchmark for natural temporal variations. RANPiLOT is evaluated across two

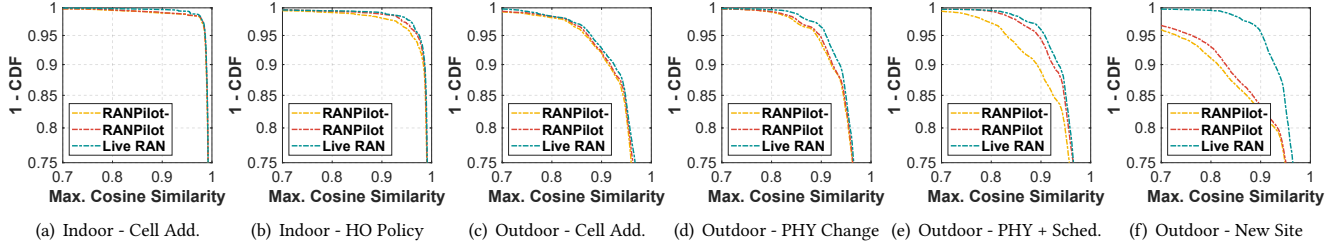


Figure 12: Quantitative fidelity validation across reconfiguration types and deployment environments.

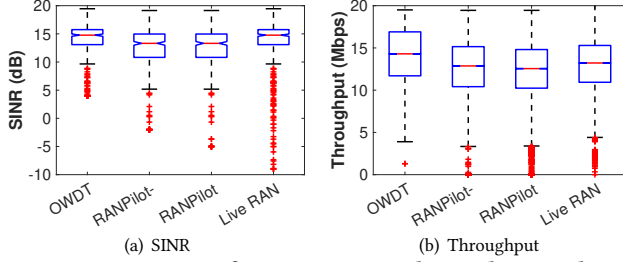


Figure 13: Statistics of RANPilot-synthesized network traffic: (a) SINR and (b) Throughput.

physical environments spanning six reconfiguration scenarios: an indoor deployment testing (1) cell addition and (2) handover policy change, followed by an outdoor deployment testing (3) cell addition, (4) PHY changes via Tx/Rx gain adjustments, (5) a combined upgrade modifying both the PHY gains and the PRB scheduling rules simultaneously, and (6) a complete system migration to a geographically unserved site. To assess the impact of historical configurations on Meta-Augmentation, we initiate the augmentation model at the start of both indoor and outdoor environments (*i.e.*, scenario (1) and (3)) and incrementally train it on the databases from the subsequent configurations.

Results. Figure 12 compares synthetic and live RAN KPM distributions across six scenarios. In the indoor setting, RANPilot closely matches the live distribution under cell addition (Figure 12(a)). Under handover policy changes (Figure 12(b)), the synthetic KPMs largely align with the real network, with minor discrepancies. This slight variation stems from the mismatch between the idealized state updates and the real timing jitter. This gap is partially mitigated by the Meta-Augmentation module, which augments operational dynamics into the synthetic KPMs. In the challenging outdoor environment, RANPilot remains robust despite higher traffic and channel variability, accurately reproducing post-reconfiguration distributions for both cell addition (Figure 12(c)) and PHY changes (Figure 12(d)). This robustness arises because site-specific propagation and traffic characteristics are embedded in the seed KPMs, providing a strong statistical prior for synthesis. For the combined upgrade involving both PHY and PRB scheduling policy changes (Figure 12(e)), the fidelity degrades slightly, as simultaneous reconfigurations introduce multiple interdependent variables into the emulation loop. The Meta-Augmentation module can mitigate this gap by leveraging historical reconfiguration data to augment the synthetic KPMs. However, a clear limitation emerges when transferring to an entirely unserved site (Figure 12(f)), where fidelity drops noticeably. The lack of representative seed KPMs for the

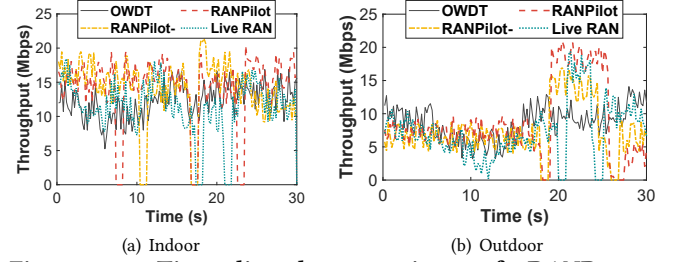


Figure 14: Time-aligned comparison of RANPilot-synthesized downlink throughput traces.

new environment constrains the synthesis pipeline, necessitating post-deployment online calibration.

Following prior work [43], we further illustrate the statistical characteristics of two representative metrics, SINR and throughput. Figure 13 shows that OWDT approximates UE mobility and produces data distributions broadly similar to the live RAN. However, it lacks the pronounced outliers that reflect real-world data drift. This gap arises because OWDT focuses primarily on PHY effects and does not account for system-level interactions such as inter-cell interference, handovers, or other uncertainties. In contrast, RANPilot achieves higher fidelity by jointly modeling RU behavior and interaction patterns, enabling it to capture the full spectrum of network fluctuations. Compared with its ablated variant, RANPilot-, the full model better reproduces transient variations and drift events, *i.e.*, key characteristics needed to train robust AI models for O-RAN. Figure 14 plots the time-aligned downlink throughput after reconfiguration in indoor and outdoor environments. Although OWDT matches the general trend of the live system, it reflects an overly simplified emulation that omits the complex interactions of dynamic RAN changes. The RANPilot-synthesized KPMs track the live RAN more closely, capturing the critical fluctuation periods introduced by updated configurations and control policies.

6.3 RANPilot Generalization

We next evaluate how RANPilot generalizes across diverse RAN reconfigurations and use cases. *Specifically, we test various reconfigurations in daily RAN operations, including hardware additions, handover policy changes, AI model updates, and topology modifications, and integrate them into three representative AI-driven use cases: QoE prediction [93], resource allocation [41], and anomaly detection [80].* These applications span diverse AI characteristics, including different model architectures, training paradigms, execution timescales, and input data modalities. We use two metrics to benchmark the AI adaptation performance: (1) *plug-and-play performance*: the performance of AI models immediately after system

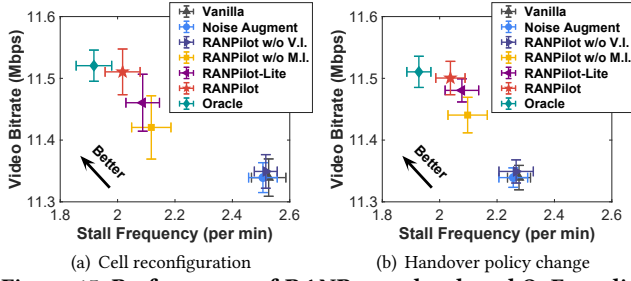


Figure 15: Performance of RANPilot-developed QoE predictor: Interaction QoEs immediately after (a) cell reconfiguration; and (b) handover policy change.

reconfiguration, which evaluates the effectiveness of synthetic data on downstream AI robustness; (2) *AI downtime*: the duration from the activation of RAN reconfiguration till the adapted AI model achieves 95% of its best performance (denoted as Oracle). For each reconfiguration, we use a 10-minute seed KPM trace to bootstrap the virtual O-RAN, generate a 10-minute synthetic KPM trace, and then extend it into a 1-hour KPM trace via Meta-Augmentation. For fair comparison, all methods use identical model architectures and differ only in the training data and adaptation mechanisms.

6.3.1 QoE prediction for mobile video streaming. We evaluate RANPilot in an adaptive video streaming application, where the QoE predictor forecasts the next two seconds' average throughput to adjust video bitrates and prevent playback stalls. Following the SOTA QoE predictor [93], we apply the same RNN model but remove the input CA parameters for throughput prediction. For calibrated noise augmentation, we synthesize the same amount of data by randomly injecting calibrated noise and temporal shifts derived from seed KPM. For the RANPilot w/o V.I., we skip the virtual O-RAN and directly apply the Meta-Augmentation module to synthesize the same amount of data based on the seed KPM. We introduce two successive RAN reconfigurations to assess generalization. The first adds a new cell to the deployment with a default SINR-based handover control policy, altering coverage and UE associations. After the model adapts to the new deployment, the second reconfiguration upgrades the handover policy to incorporate load balancing, incorporating both signal strength and UE workload distribution across available cells. Together, these reconfigurations stress the QoE predictor under hardware and control policy changes.

Results. We measure application-level QoE immediately after each reconfiguration using stall frequency and achieved bitrate. After the cell reconfiguration (Figure 15(a)), RANPilot achieves the lowest stall rate (~ 2 stalls/min), substantially outperforming Vanilla and closely approaching Oracle performance. After the handover policy reconfiguration (Figure 15(b)), Vanilla shows partial improvement as it adapts to the presence of the new cell, but remains highly sensitive to the changed handover logic, resulting in significantly more playback stalls than Oracle. In contrast, RANPilot consistently delivers near-Oracle QoE across both reconfigurations. This robustness is enabled by Virtual O-RAN, which incorporates new cell deployments and handover policies into the training data for proactive adaptation, and Meta-Augmentation, which improves tolerance to unseen topologies and policy changes by increasing volume and diversity of synthetic data. These mechanisms enable RANPilot to

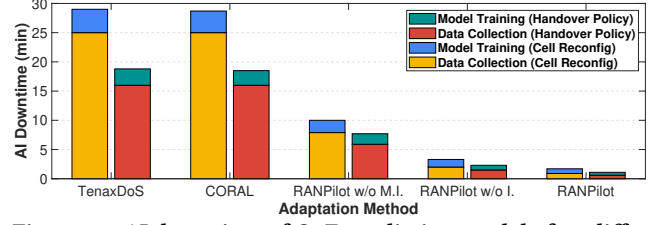


Figure 16: AI downtime of QoE prediction model after different RAN reconfigurations.

quickly align its predictions with evolving RAN conditions, translating network reconfiguration storms into stable end-user streaming experiences. Notably, simple synthetic pretraining approaches, such as omitting virtual O-RAN or applying calibrated noise augmentation, perform nearly as poorly as Vanilla. The reason is that the underlying data drift is driven by RAN unit behaviors and new control policies that cannot be captured through simple data augmentation. Furthermore, RANPilot-Lite exhibits a performance degradation compared to the full RANPilot, as the LSTM sequence generator possesses lower modeling capacity than the Transformer. Nevertheless, RANPilot-Lite still delivers performance gains over the variant without Meta-Augmentation.

To compare the reconfiguration tax, we measure the downtime of different AI adaptation strategies after each RAN reconfiguration. As shown in Figure 16, during the first reconfiguration (cell addition), the standard continual learning baseline TenaxDoS [13] requires 29 minutes to converge. Most of this delay (25 minutes) is spent collecting sufficient high quality data from the reconfigured network before meaningful learning can begin. Adopting the domain adaptation baseline CORAL [79] offers negligible performance benefits. This failure stems from a fundamental mismatch between general domain shift and RAN reconfiguration data drift: the data drift between the pre- and post-reconfiguration KPM traces (*i.e.*, the source and target domains) is not a global distribution shift, but rather a distortion of a small but essential portion of data, localized to critical execution of updated radio control policies. Coarse data-level feature alignment is difficult to mitigate changes in underlying system behavior. In contrast, even without Meta-Augmentation and Incremental Learning, RANPilot's synthesized data sharply reduces this data-collection phase to 7.9 minutes by exposing the model to the new configuration in advance. Adding Meta-Augmentation further reduces the overhead to 2 minutes by improving data diversity, while Incremental Learning cuts it to just 0.9 minutes by prioritizing and reusing the most informative live samples. As data quality improves, training itself also becomes faster. End-to-end, RANPilot completes adaptation in 1.7 minutes, achieving a 94% reduction in downtime compared to the SOTA baseline. A similar trend appears under the second reconfiguration (handover policy change). The time required to collect useful data drops from 16 minutes to 6 minutes with Virtual O-RAN, 1.5 minutes with Meta-Augmentation, and 0.6 minutes with Incremental Learning, yielding a 95% reduction in total adaptation time. These results demonstrate that RANPilot remains effective across different reconfiguration types, from physical changes (adding cells) to control-plane changes (handover policies), with each module contributing a distinct and critical performance benefit.

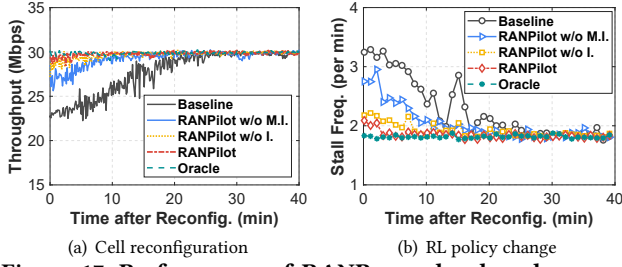


Figure 17: Performance of RANPiLOT-developed resource allocator: (a) aggregate throughput after cell reconfiguration; and (b) stalls after RL policy change.

6.3.2 Resource allocation for multi-user streaming. We evaluate RANPiLOT in a multi-user streaming scenario where an RL-based resource allocation agent manages handovers and PRB assignments across multiple UEs. Following the SOTA resource allocator [41], we apply the same RL agent and training method but enlarge the action space to support handover control. We assess generalization under two successive RAN reconfigurations. The agent is initially trained for file-transfer workloads with a pure throughput objective. The first reconfiguration expands the deployment by adding a new cell, changing the handover topology and interference conditions. After the model stabilizes, we replace file transfer with AR streaming, introducing QoE constraints. To support this, the second reconfiguration updates the RL agent’s state with per-UE buffer size and penalizes low-buffer conditions that may cause playback stalls. We use *baseline* to denote the *Vanilla+TenaxDoS* approach, which applies the Vanilla model after reconfiguration and adapts through continuous learning with real data.

Results. After the cell reconfiguration (Figure 17(a)), the baseline remains suboptimal for an extended period, requiring 26 minutes to reach Oracle-level throughput. In contrast, pre-adapting the agent using Virtual O-RAN enables immediate high throughput, reducing AI downtime to 12 minutes. Adding Meta-Augmentation further improves robustness to unseen RAN layouts by increasing the diversity and generality of synthetic data, lowering downtime to 8 minutes. After deployment, Incremental Learning accelerates adaptation using real traces, cutting downtime to only 4 minutes. After the RL policy change (Figure 17(b)), we see that the baseline requires 21 minutes to minimize playback stalls. Even when initialized with parameters from the Vanilla model, significant downtime remains because the RL policy has changed. With RANPiLOT, AI downtime drops from 21 minutes to 15 minutes with Virtual O-RAN, 9 minutes with Meta-Augmentation, and 3 minutes with Incremental Learning. Overall, RANPiLOT maintains AI robustness across both reconfigurations (*i.e.*, cell addition and AI model update).

6.3.3 Anomaly detection for system diagnosis. We evaluate RANPiLOT in an anomaly detection application, where the detector identifies anomalous behaviors during RAN operation and reports them to operators. Following the SOTA anomaly detector [80], we apply the same model architecture for distribution learning. We study model generalization under two representative cell reconfiguration scenarios that alter RAN topology: (1) shared-cell deployment, and (2) dual-cell deployment. For each scenario, we measure accuracy, recall, and precision to assess the plug-and-play performance.

Method	Accuracy	Recall	Precision	Downtime (min)
Vanilla	30.3%	0	0	19.6
RANPiLOT w/o M.I.	80.4%	92.8%	87.6%	4.7
RANPiLOT w/o I.	89.1%	95.2%	90.7%	2.9
RANPiLOT	89.1%	95.2%	90.7%	1.3
Oracle	99.9%	100%	99.9%	N/A

Table 1: Performance of RANPiLOT-developed anomaly detection model on RU contention.

Method	Accuracy	Recall	Precision	Downtime (min)
Vanilla	50.1%	86.5%	46.16%	23.7
RANPiLOT w/o M.I.	88.8%	91.3%	68.08%	6.5
RANPiLOT w/o I.	97.5%	99.1%	95.37%	4.1
RANPiLOT	97.5%	99.1%	95.37%	2.4
Oracle	99.8%	99.9%	99.7%	N/A

Table 2: Performance of RANPiLOT-developed anomaly detection model on radio interference recognition.

Results. In the shared-cell deployment, two RUs are connected to a single DU to extend coverage. This configuration introduces contention among traffic streams from concurrent RUs, resulting in packet collisions and retransmissions. As shown in Table 1, the Vanilla model completely fails to detect RU contention anomalies, yielding zero precision. Even when continuously training after deployment [13], Vanilla requires an average of 19.6 minutes to adapt to the new configuration. In contrast, RANPiLOT reduces the reconfiguration tax by 93%, recovering Oracle-level performance within only 1.3 minutes. In the dual-cell deployment, two RUs operate as independent cells, where over-the-air signal overlap leads to inter-cell interference. Table 2 summarizes the anomaly detection performance after this reconfiguration. The Vanilla model exhibits extremely low precision and a high false-positive rate, as it frequently misclassifies benign inter-cell interference as illegal radio interference. Correspondingly, Vanilla incurs an average downtime of 23.7 minutes before stabilizing. By comparison, RANPiLOT reduces downtime by 90%, adapting in just 2.4 minutes. These gains stem from the complementary roles of RANPiLOT’s components. The Virtual O-RAN and Meta-Augmentation modules proactively adapt the model using synthetic and augmented traces that anticipate post-reconfiguration behaviors, while the Incremental Learning further minimizes residual downtime by enabling rapid online refinement. Together, these modules enable RANPiLOT to achieve robust plug-and-play performance under reconfigurations.

6.4 Scalability and Overhead

We then evaluate the scalability and overhead of RANPiLOT in supporting different RAN reconfigurations. Specifically, we test four consecutive reconfiguration plans with growing topological complexity: (1) QoE predictor update; (2) cell addition with integrated handover policy; (3) PHY parameter modification with handover policy upgrade; and (4) extension of P2 to a four-cell deployment. For each plan, we first extract 10-minute representative KPM traces from real-world datasets and apply the Virtual O-RAN module to generate KPM data consistent with the target reconfigured RAN. We then invoke the meta-augmentation module to expand the dataset in both volume and network dynamics, producing one hour of KPM traces. We record the time overhead incurred at each stage. To evaluate sensitivity to UE scale, we conduct experiments using both real UEs and ZMQ-emulated UEs with varying UE counts.

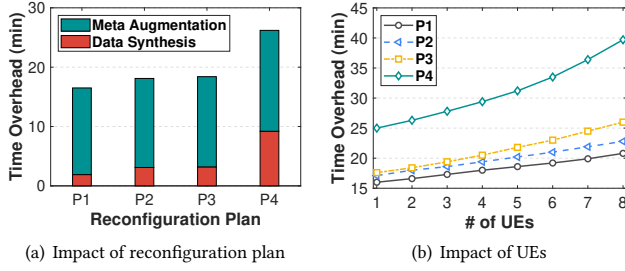


Figure 18: Overhead of RANPilot under different (a) reconfiguration plan and (b) number of UEs.

Results. Figure 18(a) shows that the time overhead of data synthesis grows with the complexity of the RAN topology. In contrast, meta augmentation incurs a higher overhead (14~15 minutes) but increases only marginally as the topology becomes more complex. The synthesis overhead is driven primarily by topological complexity, whereas the augmentation overhead is dominated by the generative model’s inference time. Overall, compared to manually collecting real-world network traffic, RANPilot reduces the time required to obtain an equivalent data volume by up to 50%. More importantly, RANPilot shifts the majority of the data-collection costs from post-reconfiguration to pre-reconfiguration, enabling proactive AI adaptation using synthetic KPMs. Given that reconfiguring even small-scale private cellular network [21, 35] can take hours or days after a plan is finalized, the additional pre-deployment overhead introduced by RANPilot is modest and practically acceptable. Figure 18(b) illustrates the time overhead as the number of UEs increases. We observe an upward trend in overhead, driven by the growing complexity of UE behavior emulation. More complex configurations (e.g., P4) exhibit both a higher baseline overhead and a steeper growth rate, owing to the expanded dimensionality of the network topology and its interaction with UE dynamics. These findings suggest that for more intricate RAN topologies and larger UE numbers, RANPilot requires a longer but acceptable buffering period prior to physical reconfiguration in order to generate high-quality data and facilitate robust AI functionality development.

7 DISCUSSION AND FUTURE WORK

Design scope and assumptions. RANPilot is designed to facilitate proactive AI adaptation for operator-initiated reconfigurations within individual base stations, rather than citywide, synchronized cellular rollouts spanning massive cell sites. Given that typical base stations operate with 1~4 cells, our system evaluation is grounded on a typical O-RAN base station mirroring this cell settings. The RANPilot’s capability of synthesizing KPM traces and adapting AI models is based on two assumptions: (1) the seed KPMs harvested during the pre-reconfiguration stage should capture the site-specific complexities of cellular networks (e.g., outdoor nature, number of users, mobility and traffic patterns, channel conditions, cell densities and capacities); and (2) the planned reconfiguration only alters RAN components, whereas external environmental features and user behavior profiles remain largely consistent.

Limitations and failure boundary. (1) *Seed KPM Consistency:* RANPilot’s data synthesis relies on the empirical consistency of seed KPMs. A failure boundary emerges when this consistency is disrupted, such as deploying new RUs in unserved areas or experiencing extreme spatial variations (Figure 12(f)). In these scenarios,

the lack of site-specific historical traffic patterns and channel dynamics can compromise the fidelity of synthetic data. RANPilot can resolve these inconsistencies after system changes by incremental learning. (2) *Advanced PHY Techniques:* modern 5G/6G networks adopt advanced PHY technologies (e.g., massive MIMO, beamforming, mmWave), which expand the network’s operational and reconfiguration space. Fully mapping these PHY changes to system-level performance impacts remains an open challenge. RANPilot abstracts PHY changes into signal strength variations using a simplified wireless channel model. Our future work will explore integrating PHY-foreseeing emulation to support proactive adaptation for complex PHY policy changes (e.g., beamforming).

Robustness for unplanned changes. RANPilot targets data drift introduced by planned, operator-initiated RAN reconfigurations, where network changes are known ahead of deployment. In real-world operations, unplanned changes may also introduce sudden or gradual data drift that invalidates deployed AI models. Examples include sudden CBRS license changes due to higher-priority incumbents, bursty UE surges triggered by mass gatherings, and environmental dynamics such as new transmitters/reflectors that alter radio propagation. Mitigating such unplanned changes requires orthogonal mechanisms. For example, scaling the training dataset to encompass diverse unplanned dynamics, thereby enhancing the AI model’s inherent robustness, or deploying reactive, online calibration strategies to fix performance gaps after the drift is observed [92]. Fully addressing AI robustness under unplanned network changes requires RANPilot to work together with these online calibration mechanisms and remains a promising direction for future work.

AI-native data synthesis. While RANPilot enables proactive AI adaptation, it still relies on human intervention to bootstrap the data-synthesis pipeline: operators must explicitly provide reconfiguration plans via RAN and AI APIs. Looking ahead, the broader AI-RAN vision [4, 9] for NextG cellular systems aims for a fully *AI-native* architecture, in which the RAN autonomously determines when and how to reconfigure. Realizing such a paradigm would require the control plane to automatically infer performance bottlenecks, synthesize candidate reconfiguration plans, and seamlessly trigger RANPilot to generate synthetic KPM data and update deployed AI models, entirely without operator guidance. Our future work will explore AI-native data synthesis for the evolution of intelligent, self-optimizing RANs.

8 CONCLUSION

This paper introduces RANPilot, a novel AI evolution framework that bridges the gap between O-RAN’s architectural flexibility and rigid AI development paradigm. Unlike conventional methods, RANPilot facilitates proactive adaptation of AI models prior to planned system upgrades, based on novel *virtual O-RAN powered data synthesis*. This digital-leadoff approach enables robust AI functionalities with sustained performance and seamless transition to new network configurations. Extensive evaluations demonstrate the advantages of RANPilot in reducing AI service downtime during RAN reconfiguration. We believe RANPilot can potentially empower more adaptive AI-driven control and optimization in dynamic O-RAN reconfigurations.

ACKNOWLEDGMENTS

We sincerely thank the anonymous shepherd and reviewers for their comments and feedback. This work is supported in part by HK RGC GRF (Grant No. 15204926, 15216325, and 15211924), HK RGC CRF (Grant No. C5085-25G and C6086-25G), and HK PolyU RIAIoT P0051269. Xianjin Xia and Yuanqing Zheng are the corresponding authors.

REFERENCES

- [1] 3GPP Standard (TS) 38.331. 2018. *NR; Radio Resource Control (RRC); Protocol Specification, Version 15.0.0*. "http://www.3gpp.org/DynaReport/38331.htm".
- [2] 3GPP. 2024. *3rd Generation Partnership Project*. "https://www.3gpp.org/".
- [3] Attai Ibrahim Abubakar, Oluwakayode Onireti, Yusuf Sambo, Lei Zhang, GK Ragesh, and Muhammad Ali Imran. 2023. Energy efficiency of open radio access network: A survey. In *2023 IEEE 97th Vehicular Technology Conference (VTC2023-Spring)*. IEEE, 1–7.
- [4] AI-RAN Alliance. 2024. *AI-RAN*. "https://ai-ran.org/".
- [5] O-RAN Alliance. 2024. *O-RAN*. "https://www.o-ran.org".
- [6] Abd AlRhaman AlQiam, Yuanjun Yao, Zhaodong Wang, Satyajeet Singh Ahuja, Ying Zhang, Sanjay G Rao, Bruno Ribeiro, and Mohit Tawarmalani. 2024. Transferable neural wan te for changing topologies. In *Proceedings of the ACM SIGCOMM 2024 Conference*. 86–102.
- [7] Esmaeil Amiri, Ning Wang, Mohammad Shojafar, Mutasem Q Hamdan, Chuan Heng Foh, and Rahim Tafazolli. 2023. Deep reinforcement learning for robust vnf reconfigurations in o-ran. *IEEE Transactions on Network and Service Management* 21, 1 (2023), 1115–1128.
- [8] Congkai An, Huanhuan Zhang, Shibo Wang, Jingyang Kang, Anfu Zhou, Liang Liu, Huadong Ma, Zili Meng, Delei Ma, Yusheng Dong, et al. 2025. Tooth: Toward Optimal Balance of Video {QoE} and Redundancy Cost by {Fine-Grained} {FEC} in Cloud Gaming Streaming. In *22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI 25)*. 635–651.
- [9] Ganesh Ananthanarayanan, Matthew Balkwill, Xenofon Foukas, Zhihua Lai, Bozidar Radunovic, Connor Settle, and Yongguang Zhang. 2025. Distributed AI Platform for the 6G RAN. In *Proceedings of the 2nd ACM Workshop on Open and AI RAN*. 15–21.
- [10] Fatih Aslan, Jose A Ayala-Romero, Andres Garcia-Saavedra, Xavier Costa-Perez, and George Iosifidis. 2025. FairRIC: Real-Time Fair Allocation in O-RAN with Shared Computing. In *IEEE INFOCOM 2025-IEEE Conference on Computer Communications*. IEEE, 1–10.
- [11] Fatih Aslan, George Iosifidis, Jose A Ayala-Romero, Andres Garcia-Saavedra, and Xavier Costa-Perez. 2024. Fair resource allocation in virtualized o-ran platforms. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 8, 1 (2024), 1–34.
- [12] Arjun Balasingam, Manikanta Kotaru, and Paramvir Bahl. 2024. {Application-Level} Service Assurance with 5G {RAN} Slicing. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. 841–857.
- [13] Chafika Benzaid, Fahim Muhtasim Hossain, Tarik Taleb, Pedro Merino Gómez, and Michael Dieudonne. 2024. A federated continual learning framework for sustainable network anomaly detection in o-ran. In *2024 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 1–6.
- [14] Romil Bhardwaj, Zhengxu Xia, Ganesh Ananthanarayanan, Junchen Jiang, Yuan-chao Shu, Nikolaos Karianakis, Kevin Hsieh, Paramvir Bahl, and Ion Stoica. 2022. Ekya: Continuous learning of video analytics models on edge compute servers. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*. 119–135.
- [15] Nishant Budhdev, Raj Joshi, Pravein Govindan Kannan, Mun Choon Chan, and Tulika Mitra. 2021. FSA: Fronthaul slicing architecture for 5G using dataplane programmable switches. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*. 723–735.
- [16] Raphael Cannatà, Haoxin Sun, Dan Mihai Dumitriu, and Haitham Hassanieh. 2024. Towards Seamless 5G Open-RAN Integration with WebAssembly. In *Proceedings of the 23rd ACM Workshop on Hot Topics in Networks*. 121–131.
- [17] Xingyu Chen and Xinyu Zhang. 2023. Rf genesis: Zero-shot generalization of mmwave sensing through simulation-based data synthesis and generative diffusion models. In *Proceedings of the 21st ACM Conference on Embedded Networked Sensor Systems*. 28–42.
- [18] Yongzhou Chen, Ruihao Yao, Haitham Hassanieh, and Radhika Mittal. 2023. Channel-Aware 5g RAN slicing with customizable schedulers. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. 1767–1782.
- [19] Yihua Cheng, Ziyi Zhang, Hanchen Li, Anton Arapin, Yue Zhang, Qizheng Zhang, Yuhua Liu, Kuntai Du, Xu Zhang, Francis Y Yan, et al. 2024. {GRACE}:{Loss-Resilient}{Real-Time} video through neural codecs. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. 509–531.
- [20] Guoxuan Chi, Zheng Yang, Chenshu Wu, Jingao Xu, Yuchong Gao, Yunhao Liu, and Tony Xiao Han. 2024. RF-diffusion: Radio signal generation via time-frequency diffusion. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*. 77–92.
- [21] Ericsson. 2025. *Private 5G, a network as dedicated as you are*. "https://www.ericsson.com/en/private-networks".
- [22] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*. PMLR, 1126–1135.
- [23] Xenofon Foukas and Bozidar Radunovic. 2021. Concordia: Teaching the 5G vRAN to share compute. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*. 580–596.
- [24] Xenofon Foukas, Bozidar Radunovic, Matthew Balkwill, and Zhihua Lai. 2023. Taking 5G RAN analytics and control to a new level. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*. 1–16.
- [25] Xenofon Foukas, Tenzin Samten Ukyab, Bozidar Radunovic, Sylvia Ratnasamy, and Scott Shenker. 2025. RANBooster: Democratizing advanced cellular connectivity through fronthaul middleboxes. In *Proceedings of the ACM SIGCOMM 2025 Conference*. 742–757.
- [26] Changhan Ge, Zihui Ge, Xuan Liu, Ajay Mahimkar, Yusef Shaqalle, Yu Xiang, and Shomik Pathak. 2023. Chroma: Learning and using network contexts to reinforce performance improving configurations. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*. 1–16.
- [27] Changhan Ge, Ajay Mahimkar, Zihui Ge, Romeo Fernandez, Joseph Maniaci, Shomik Pathak, and Maulik Shah. 2025. Iridescence: Improving Configuration Tuning in the Presence of Confounders for 5G NSA Networks. *Proceedings of the ACM on Networking* 3, CoNEXT1 (2025), 1–22.
- [28] Ushasi Ghosh, Azuka Chiejina, Nathan Stephenson, Vijay K Shah, Srinivas Shakkottai, and Dinesh Bharadia. 2024. SPARC: Spatio-Temporal Adaptive Resource Control for Multi-site Spectrum Management in NextG Cellular Networks. *Proceedings of the ACM on Networking* 2, CoNEXT4 (2024), 1–18.
- [29] Chen Gong, Bo Liang, Wei Gao, and Chenren Xu. 2025. Data Can Speak for Itself: Quality-guided Utilization of Wireless Synthetic Data. In *Proceedings of the 23rd Annual International Conference on Mobile Systems, Applications and Services*. 209–222.
- [30] Adel Heidari, Agrim Gupta, Ish Kumar Jain, and Dinesh Bharadia. 2025. PhaseMO: A Universal Massive MIMO Architecture for Sustainable NextG. In *IEEE INFOCOM 2025-IEEE Conference on Computer Communications*. IEEE, 1–10.
- [31] Ethan Sanchez Hidalgo, Jose A Ayala-Romero, J Xavier Salvat Lozano, Andres Garcia-Saavedra, and Xavier Costa Perez. 2025. AegisRAN: A Fair and Energy-Efficient Computing Resource Allocation Framework for vRANs. *IEEE Transactions on Mobile Computing* (2025).
- [32] Franz Hlawatsch and Gerald Matz. 2011. *Wireless communications over rapidly time-varying channels*. Academic press.
- [33] Tetsuya Iye, Masaya Sakamoto, Shohei Takaya, Eisaku Sato, Yuki Susukida, Yu Nagaoka, Kazuki Maruta, and Jin Nakazato. 2025. Open Wireless Digital Twin: End-to-End 5G Mobility Emulation in O-RAN Framework. *arXiv preprint arXiv:2503.12177* (2025).
- [34] Lianchen Jia, Chao Zhou, Chaoyang Li, Jiangchuan Liu, and Lifeng Sun. 2025. Towards User-level QoE: Large-scale Practice in Personalized Optimization of Adaptive Video Streaming. In *Proceedings of the ACM SIGCOMM 2025 Conference*. 1154–1166.
- [35] Matthew Johnson, Sudheesh Singanamalla, Nick Durand, Esther Han Boel Jang, Spencer Sevilla, and Kurtis Heimerl. 2024. dAuth: A Resilient Authentication Architecture for Federated Private Cellular Networks. In *Proceedings of the ACM SIGCOMM 2024 Conference*. 373–391.
- [36] Anuj Kalia, Nikita Lazarev, Leyang Xue, Xenofon Foukas, Bozidar Radunovic, and Francis Y Yan. 2025. Towards Energy Efficient 5G vRAN Servers. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*.
- [37] Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. 2018. Measuring catastrophic forgetting in neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.
- [38] Naveed Ali Khan and Stefan Schmid. 2023. AI-RAN in 6G Networks: State-of-the-Art and Challenges. *IEEE Open Journal of the Communications Society* 5 (2023), 294–311.
- [39] Xin Zhe Khooi, Anuj Kalia, and Mun Choon Chan. 2025. How to Update Your 5G vRAN. In *Proceedings of the 31st Annual International Conference on Mobile Computing and Networking*. 123–138.
- [40] Caner Kilinc, Mahesh K Marina, Muhammad Usama, Salih Ergut, Jon Crowcroft, Tugrul Gundogdu, and Ilhan Akinci. 2022. JADE: Data-driven automated jammer detection framework for operational mobile networks. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 1139–1148.
- [41] Woo-Hyun Ko, Ushasi Ghosh, Ujwal Dinesha, Raini Wu, Srinivas Shakkottai, and Dinesh Bharadia. 2024. EdgeRIC: Empowering real-time intelligent optimization and control in NextG cellular networks. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. 1315–1330.
- [42] Andrea Lacava, Michele Polese, Rajarajan Sivaraj, Rahul Soundarajan, Bhawani Shanker Bhati, Tarunjeet Singh, Tommaso Zugno, Francesca Cuomo,

- and Tommaso Melodia. 2023. Programmable and customized intelligence for traffic steering in 5G networks using open RAN architectures. *IEEE Transactions on Mobile Computing* 23, 4 (2023), 2882–2897.
- [43] Zeqi Lai, Hewu Li, Yangtao Deng, Qian Wu, Jun Liu, Yuanjie Li, Jihao Li, Lixin Liu, Weisen Liu, and Jianping Wu. 2023. {StarryNet}: empowering researchers to evaluate futuristic integrated space and terrestrial networks. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. 1309–1324.
- [44] Nitya Lakshmanan, Nishant Budhdev, Min Suk Kang, Mun Choon Chan, and Jun Han. 2021. A stealthy location identification attack exploiting carrier aggregation in cellular networks. In *30th usenix security symposium (usenix security 21)*. 3899–3916.
- [45] Nikita Lazarev, Tao Ji, Anuj Kalia, Daehyeok Kim, Ilias Marinos, Francis Y Yan, Christina Delimitrou, Zhiru Zhang, and Aditya Akella. 2023. Resilient baseband processing in virtualized rans with slingshot. In *Proceedings of the ACM SIGCOMM 2023 Conference*. 654–667.
- [46] Yang Li, Hao Lin, Zhenhua Li, Yunhao Liu, Feng Qian, Liangyi Gong, Xianlong Xin, and Tianyin Xu. 2021. A nationwide study on cellular reliability: Measurement, analysis, and enhancements. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*. 597–609.
- [47] Jianhua Lin. 2002. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information theory* 37, 1 (2002), 145–151.
- [48] Wei Lin, Zongxiao Li, Binbin Chen, Jianwei Liu, Ray-Guang Cheng, and Fan Zhang. 2025. 5G-Muffler: Covert DoS Attacks over Open Fronthaul Interface of O-RAN 5G Network. In *IEEE INFOCOM 2025-IEEE Conference on Computer Communications*. IEEE, 1–10.
- [49] Shinan Liu, Francesco Bronzino, Paul Schmitt, Arjun Nitin Bhagoji, Nick Feamster, Hector Garcia Crespo, Timothy Coyle, and Brian Ward. 2023. Leaf: Navigating concept drift in cellular networks. *Proceedings of the ACM on Networking* 1, CoNEXT2 (2023), 1–24.
- [50] Shinan Liu, Francesco Bronzino, Paul Schmitt, Nick Feamster, Ricardo Borges, Hector Garcia Crespo, and Brian Ward. 2021. Understanding model drift in a large cellular network. *CoRR* (2021).
- [51] J Xavier Salvat Lozano, Jose A Ayala-Romero, Andres Garcia-Saavedra, and Xavier Costa-Perez. 2025. Kairos: Energy-efficient radio unit control for O-RAN via advanced sleep modes. In *IEEE INFOCOM 2025-IEEE Conference on Computer Communications*. IEEE, 1–10.
- [52] Ajay Mahimkar, Zihui Ge, Xuan Liu, Yusef Shaqalle, Yu Xiang, Jennifer Yates, Shomik Pathak, and Rick Reichel. 2022. Aurora: conformity-based configuration recommendation to improve LTE/5G service. In *Proceedings of the 22nd ACM Internet Measurement Conference*. 83–97.
- [53] Ajay Mahimkar, Zihui Ge, Jia Wang, Jennifer Yates, Yin Zhang, Joanne Emmons, Brian Huntley, and Mark Stockert. 2011. Rapid detection of maintenance induced changes in service performance. In *Proceedings of the Seventh Conference on Emerging Networking Experiments and Technologies*. 1–12.
- [54] Ajay Mahimkar, Zihui Ge, Jennifer Yates, Chris Hristov, Vincent Cordaro, Shane Smith, Jing Xu, and Mark Stockert. 2013. Robust assessment of changes in cellular networks. In *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*. 175–186.
- [55] Ajay Mahimkar, Ashiwan Sivakumar, Zihui Ge, Shomik Pathak, and Karunashish Biswas. 2021. Auric: Using data-driven recommendation to automatically generate cellular configuration. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*. 807–820.
- [56] Ajay Anil Mahimkar, Han Hee Song, Zihui Ge, Aman Shaikh, Jia Wang, Jennifer Yates, Yin Zhang, and Joanne Emmons. 2010. Detecting the performance impact of upgrades in large operational networks. In *Proceedings of the ACM SIGCOMM 2010 Conference*. 303–314.
- [57] Marc Masana, Xialei Liu, Bartłomiej Twardowski, Mikel Menta, Andrew D Bagdanov, and Joost Van De Weijer. 2022. Class-incremental learning: survey and performance evaluation on image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 5 (2022), 5513–5533.
- [58] Lifan Mei, Jinrui Gou, Yujin Cai, Houwei Cao, and Yong Liu. 2022. Realtime mobile bandwidth and handoff predictions in 4G/5G networks. *Computer Networks* 204 (2022), 108736.
- [59] Zili Meng, Yaning Guo, Chen Sun, Bo Wang, Justine Sherry, Hongqiang Harry Liu, and Mingwei Xu. 2022. Achieving consistent low latency for wireless real-time communications with the shortest control loop. In *Proceedings of the ACM SIGCOMM 2022 Conference*. 193–206.
- [60] Zili Meng, Xiao Kong, Jing Chen, Bo Wang, Mingwei Xu, Rui Han, Honghao Liu, Venkat Arun, Hongxin Hu, and Xue Wei. 2024. Hairpin: Rethinking packet loss recovery in edge-based interactive video streaming. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. 907–926.
- [61] Suvidha Mhatre, Ferran Adelantado, Kostas Ramantas, and Christos Verikoukis. 2025. Transfer learning applied to deep reinforcement learning for 6G resource management in intra- and inter-slice RAN-edge domains. *IEEE Transactions on Consumer Electronics* (2025).
- [62] Federico Mungari, Corrado Puligheddu, Andres Garcia-Saavedra, and Carla Fabiana Chiasserini. 2025. O-RAN intelligence orchestration framework for quality-driven Xapp deployment and sharing. *IEEE Transactions on Mobile Computing* (2025).
- [63] Ahmad M Nagib, Hatem Abou-Zeid, and Hossam S Hassanein. 2023. Safe and accelerated deep reinforcement learning-based o-ran slicing: A hybrid transfer learning approach. *IEEE Journal on Selected Areas in Communications* 42, 2 (2023), 310–325.
- [64] Huan X Nguyen, Kexuan Sun, Duc To, Quoc-Tuan Vien, and Tuan Anh Le. 2024. Digital twin for o-ran toward 6g. *IEEE Communications Magazine* (2024).
- [65] Quoc Phong Nguyen, Kar Wai Lim, Dinil Mon Divakaran, Kian Hsiang Low, and Mun Choon Chan. 2019. Gee: A gradient-based explainable variational autoencoder for network anomaly detection. In *2019 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 91–99.
- [66] Rawlings Ntassah, Gian Michele Dell'Aera, and Fabrizio Granelli. 2023. xApp for traffic steering and load balancing in the O-RAN architecture. In *ICC 2023-IEEE International Conference on Communications*. IEEE, 5259–5264.
- [67] Open5GS. 2024. . "https://open5gs.org".
- [68] Satis Kumar Permal, Yixi Chen, Xin Zhe Khoi, Bqing Qiu, and Mun Choon Chan. 2024. Towards Continuous Latency Monitoring through Open RAN Interfaces. In *Proceedings of the 4th ACM Workshop on 5G and Beyond Network Measurements, Modeling, and Use Cases*. 14–20.
- [69] Yarin Perry, Felipe Vieira Frujeri, Chaim Hoch, Srikanth Kandula, Ishai Menache, Michael Schapira, and Aviv Tamar. 2023. {DOTE}: Rethinking (predictive){WAN} traffic engineering. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. 1557–1581.
- [70] Michele Polese, Leonardo Bonati, Salvatore D'Oro, Pedram Johari, Davide Villa, Sakthivel Velumani, Rajeev Gangula, Maria Tsampazi, Clifton Paul Robinson, Gabriele Gemmi, et al. 2024. Colosseum: The open RAN digital twin. *IEEE Open Journal of the Communications Society* 5 (2024), 5452–5466.
- [71] Michele Polese, Leonardo Bonati, Salvatore D'oro, Stefano Basagni, and Tommaso Melodia. 2023. Understanding O-RAN: Architecture, interfaces, algorithms, security, and research challenges. *IEEE Communications Surveys & Tutorials* 25, 2 (2023), 1376–1411.
- [72] O-RAN Project. 2025. *O-RAN SC RIC*. "https://docs.o-ran-sc.org/en/latest/index.html".
- [73] Mubashir Adnan Qureshi, Ajay Mahimkar, Lili Qiu, Zihui Ge, Max Zhang, and Ioannis Broustis. 2017. Coordinating rolling software upgrades for cellular networks. In *2017 IEEE 25th International Conference on Network Protocols (ICNP)*. IEEE, 1–10.
- [74] Shunmugapriya Ramanathan, Koteswararao Kondepudi, and Andrea Fumagalli. 2022. Resiliency in Open-Source Solutions for Disaggregated 5G Cloud Radio Access and Transport Networks. In *2022 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. IEEE, 124–129.
- [75] Leonardo Lo Schiavo, Gines Garcia-Aviles, Andres Garcia-Saavedra, Marco Gramaglia, Marco Fiore, Albert Banchs, and Xavier Costa-Perez. 2024. Cloudric: Open radio access network (o-ran) virtualization with shared heterogeneous computing. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*. 558–572.
- [76] Rubayet Shafin, Lingjia Liu, Vikram Chandrasekhar, Hao Chen, Jeffrey Reed, and Jianzhong Charlie Zhang. 2020. Artificial intelligence-enabled cellular networks: A critical path to beyond-5G and 6G. *IEEE Wireless Communications* 27, 2 (2020), 212–217.
- [77] Zhongyang Shen. 2024. Outlier Detect using Vector Cosine Similarity by Adding a Dimension. In *IEEE ICAIC*. 255–259.
- [78] srsRAN: Open source 4g/5g software radio access network. 2023. . "https://www.srsran.com/".
- [79] Baochen Sun, Jiashi Feng, and Kate Saenko. 2016. Return of frustratingly easy domain adaptation. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 30.
- [80] Chuanhao Sun, Ujjwal Pawar, Molham Khoja, Xenofon Foulas, Mahesh K Marina, and Bozidar Radunovic. 2024. SpotLight: Accurate, explainable and efficient anomaly detection for Open RAN. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*. 923–937.
- [81] sysmoSIM SJA2. 2021. . "https://sysmocom.de/products/sim/sysmouisim/".
- [82] Haonan Tong, Mingzhe Chen, Jun Zhao, Ye Hu, Zhaoxun Yang, Yuchen Liu, and Changchuan Yin. 2025. Continual Reinforcement Learning for Digital Twin Synchronization Optimization. *IEEE Transactions on Mobile Computing* (2025).
- [83] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).
- [84] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).
- [85] Liyan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. 2024. A comprehensive survey of continual learning: Theory, method and application. *IEEE transactions on pattern analysis and machine intelligence* 46, 8 (2024), 5362–5383.
- [86] Mowei Wang, Yong Cui, Shihan Xiao, Xin Wang, Dan Yang, Kai Chen, and Jun Zhu. 2018. Neural network meets DCN: Traffic-driven topology adaptation with

- deep learning. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 2, 2 (2018), 1–25.
- [87] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. 2016. A survey of transfer learning. *Journal of Big data* 3, 1 (2016), 9.
- [88] Haohuang Wen, Phillip Porras, Vinod Yegneswaran, Ashish Gehani, and Zhiqiang Lin. 2024. 5G-spector: An O-RAN compliant layer-3 cellular attack detection service. In *Proceedings of the 31st Annual Network and Distributed System Security Symposium, NDSS*, Vol. 24.
- [89] Haohuang Wen, Prakhar Sharma, Vinod Yegneswaran, Phillip Porras, Ashish Gehani, and Zhiqiang Lin. 2024. 6G-XSec: Explainable Edge Security for Emerging OpenRAN Architectures. In *Proceedings of the 23rd ACM Workshop on Hot Topics in Networks*. 77–85.
- [90] Jiarong Xing, Junzhi Gong, Xenofon Foukas, Anuj Kalia, Daehyeok Kim, and Manikanta Kotaru. 2023. Enabling resilience in virtualized rans with atlas. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*. 1–15.
- [91] Xing Xu, Ioannis Broustis, Zihui Ge, Ramesh Govindan, Ajay Mahimkar, NK Shankaranarayanan, and Jia Wang. 2015. Magus: Minimizing cellular service disruption during network upgrades. In *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*. 1–13.
- [92] He Yan, Ashley Flavel, Zihui Ge, Alexandre Gerber, Dan Massey, Christos Papadopoulos, Hiren Shah, and Jennifer Yates. 2012. Argus: End-to-end service anomaly detection and localization from an ISP's point of view. In *2012 Proceedings IEEE INFOCOM*. IEEE, 2756–2760.
- [93] Wei Ye, Xinyue Hu, Steven Sleder, Anlan Zhang, Udhaya Kumar Dayalan, Ahmad Hassan, Rostand AK Fezeu, Akshay Jajoo, Myungjin Lee, Eman Ramadan, et al. 2024. Dissecting carrier aggregation in 5G networks: Measurement, QoE implications and prediction. In *Proceedings of the ACM SIGCOMM 2024 Conference*. 340–357.
- [94] Frederik Jonathan Zumegen, Ish Kumar Jain, and Dinesh Bharadia. 2024. Beamarmor: Seamless anti-jamming in 5g cellular networks with mimo null-steering. In *Proceedings of the 25th International Workshop on Mobile Computing Systems and Applications*. 121–126.

A APPENDICES

Appendices are supporting material that has not been peer-reviewed.

A.1 Descriptions of KPMs:

- *DRB.UEThpDL* – Average downlink UE throughput.
- *DRB.UEThpUL* – Average uplink UE throughput.
- *DRB.AirIfDelayUL* – Average delay of the downlink air-interface.
- *DRB.RlcDelayUL* – Average RLC packet delay in the uplink.
- *DRB.RlcPacketDropRateDL* – Downlink packet drop rate in DU.
- *DRB.RlcSduDelayDL* – Average delay of downlink SDUs in DU.
- *DRB.RlcSduTransmittedVolumeDL* – Volume of transmitted downlink data.
- *DRB.RlcSduTransmittedVolumeUL* – Volume of transmitted uplink data.
- *RRU.PrbTotDL* – Downlink total PRB usage.
- *RRU.PrbTotUL* – Uplink total PRB usage.
- *RRU.PrbUsedDL* – Mean downlink PRBs used for data traffic.
- *RRU.PrbAvailDL* – Total available PRBs in downlink.
- *RRU.PrbUsedUL* – Mean uplink PRBs used for data traffic.
- *RRU.PrbAvailUL* – Total available PRBs in uplink.
- *slot_decision_latency* – Interval between scheduling request and decision.
- *slot_k1* – Minimum scheduling time interval in slots.
- *pucch_sinr_fm2* – SINR for PUCCH format 2.
- *pucch_timeali_us_fm2* – Synchronization time between UE and gNB for PUCCH format 2.
- *pucch_metric_fm1* – Performance metric for PUCCH format 1.
- *pucch_sinr_fm1* – SINR for PUCCH format 1.
- *pucch_timeali_us_fm1* – Synchronization time between UE and gNB for PUCCH format 1.
- *pusch_tbs* – Transport block size for PUSCH.
- *pusch_sch_sinr* – SINR for scheduled PUSCH.
- *pusch_time_us* – PUSCH transmission time.
- *pdccch_time_us* – PDCCH transmission time.
- *pdsch_tbs* – Transport block size for PDSCH.
- *pdsch_time_us* – PDSCH transmission time.
- *du_ue1_cqi*, *du_ue2_cqi* – Channel Quality Indicator reported by UE1/UE2.

- *du_ue1_dl_brat_kbps*, *du_ue2_dl_brat_kbps* – Downlink bitrate for UE1/UE2.
- *du_ue1_dl_mcs*, *du_ue2_dl_mcs* – Downlink MCS indicator for UE1/UE2.
- *du_ue1_dl_nof_ok*, *du_ue2_dl_nof_ok* – Successful DL packets for UE1/UE2.
- *du_ue1_dl_nof_nok*, *du_ue2_dl_nof_nok* – Failed DL packets for UE.
- *du_ue1_dl_failrate*, *du_ue2_dl_failrate* – DL failure rate for UE.
- *du_ue1_dl_bs*, *du_ue2_dl_bs* – Downlink buffer size for UE.
- *du_ue1_pusch_snr_db*, *du_ue2_pusch_snr_db* – Uplink SNR for UE PUSCH.
- *du_ue1_pusch_rsrp_db*, *du_ue2_pusch_rsrp_db* – RSRP measurements for UE PUSCH.
- *du_ue1_ul_mcs*, *du_ue2_ul_mcs* – Uplink MCS indicator for UE.
- *du_ue1_ul_brat_kbps*, *du_ue2_ul_brat_kbps* – Uplink bitrate for UE1.
- *du_ue1_ul_nof_ok*, *du_ue2_ul_nof_ok* – Successful UL packets for UE.
- *du_ue1_ul_nof_nok*, *du_ue2_ul_nof_nok* – Failed UL packets for UE.
- *du_ue1_ul_failrate*, *du_ue2_ul_failrate* – UL failure rate for UE.
- *du_ue1_bsr*, *du_ue2_bsr* – Buffer status reports for UE uplink.
- *du_ue1_ta_ns*, *du_ue2_ta_ns* – Timing advance for UE uplink.
- *du_ue1_last_phr*, *du_ue2_last_phr* – Power headroom reports for UE uplink.

KPM Name	Category	Source
DRB.UEThpDL	Wireless Channel Quality	Subscription
DRB.UEThpUL	Wireless Channel Quality	Subscription
DRB.AirIfDelayUL	QoS	Subscription
DRB.RlcDelayUL	QoS	Subscription
DRB.RlcSduTransmittedVolumeDL	QoS	Subscription
DRB.RlcSduTransmittedVolumeUL	QoS	Subscription
DRB.RlcPacketDropRateDL	QoS	Subscription
DRB.RlcSduDelayDL	QoS	Subscription
RRU.PrbTotDL	Resource Scheduling	Subscription
RRU.PrbTotUL	Resource Scheduling	Subscription
RRU.PrbUsedDL	Resource Scheduling	Subscription
RRU.PrbAvailDL	Resource Scheduling	Subscription
RRU.PrbUsedUL	Resource Scheduling	Subscription
RRU.PrbAvailUL	Resource Scheduling	Subscription
slot_decision_latency	Resource Scheduling	Hooks
slot_k1	Resource Scheduling	Hooks
pucch_sinr_fm2	Wireless Channel Quality	Hooks
pucch_timeali_fm2	Resource Scheduling	Hooks
pucch_metric_fm1	QoS	Hooks
pucch_sinr_fm1	Wireless Channel Quality	Hooks
pucch_timeali_fm1	Resource Scheduling	Hooks
pusch_tbs	Wireless Channel Quality	Hooks
pusch_sch_sinr	Wireless Channel Quality	Hooks
pusch_time	Resource Scheduling	Hooks
pdccch_time	Resource Scheduling	Hooks
pdsch_tbs	Wireless Channel Quality	Hooks
pdsch_time	Resource Scheduling	Hooks
du_ue1_cqi	Wireless Channel Quality	Hooks
du_ue2_cqi	Wireless Channel Quality	Hooks
du_ue1_dl_brat	Wireless Channel Quality	Hooks
du_ue2_dl_brat	Wireless Channel Quality	Hooks
du_ue1_dl_mcs	Wireless Channel Quality	Hooks
du_ue2_dl_mcs	Wireless Channel Quality	Hooks
du_ue1_dl_nof_ok	QoS	Hooks
du_ue2_dl_nof_ok	QoS	Hooks
du_ue1_dl_nof_nok	QoS	Hooks
du_ue2_dl_nof_nok	QoS	Hooks
du_ue1_dl_failrate	QoS	Hooks
du_ue2_dl_failrate	QoS	Hooks
du_ue1_dl_bs	QoS	Hooks
du_ue2_dl_bs	QoS	Hooks
du_ue1_pusch_snr	Wireless Channel Quality	Hooks
du_ue2_pusch_snr	Wireless Channel Quality	Hooks
du_ue1_pusch_rsrp	Wireless Channel Quality	Hooks
du_ue2_pusch_rsrp	Wireless Channel Quality	Hooks
du_ue1_ul_mcs	Wireless Channel Quality	Hooks
du_ue2_ul_mcs	Wireless Channel Quality	Hooks
du_ue1_ul_brat	Wireless Channel Quality	Hooks
du_ue2_ul_brat	Wireless Channel Quality	Hooks
du_ue1_ul_nof_ok	QoS	Hooks
du_ue2_ul_nof_ok	QoS	Hooks
du_ue1_ul_nof_nok	QoS	Hooks
du_ue2_ul_nof_nok	QoS	Hooks
du_ue1_ul_failrate	QoS	Hooks
du_ue2_ul_failrate	QoS	Hooks
du_ue1_bsr	QoS	Hooks
du_ue2_bsr	QoS	Hooks
du_ue1_ta	QoS	Hooks
du_ue2_ta	QoS	Hooks
du_ue1_phr	QoS	Hooks
du_ue2_phr	QoS	Hooks

Table 3: KPM database collected from srsRAN

An increase in the number of UEs can also increase the dimensionality of the KPM samples. To handle such changes, during the

Algorithm 1: Continuous meta-augmentation

Input : The KPM set $\mathcal{K}$ after data synthesis (§4.2)
Output : The augmented KPM dataset $\mathcal{D}$

```

1 // ❶ Autoregressive Pretraining
2 Initialize  $\mathcal{M}$ ;
3  $\mathcal{K} \xrightarrow{\text{slice}} \mathcal{K}_s = \{k_1, k_2, \dots\}$  // KPM slicing
4 for each  $k_i \in \mathcal{K}_s$  do
5    $k'_i = \mathcal{M}(k_i)$ ;  $\mathcal{L}_{\text{pretrain}} = \text{sim}(k'_i, \hat{k}'_i)$ ;
6   Update( $\mathcal{M}$ ,  $\mathcal{L}_{\text{pretrain}}$ ); Append  $k'_i$  to  $k_i \rightarrow k_i$ 
7 end
8 // ❷ Meta-Learning
9 for each  $\mathcal{K}_i \in \mathcal{D}$  do
10  Split  $\mathcal{K}_i \rightarrow (\mathcal{S}_i, \mathcal{Q}_i)$ ;  $\mathcal{M}_i = \text{Copy}(\mathcal{M})$ ;
11   $\mathcal{L}_{\mathcal{S}_i} = \sum_{k_j, k'_j \in \mathcal{S}_i} \text{sim}(k'_j, \mathcal{M}_i(k_j))$ ; Update( $\mathcal{M}_i$ ,  $\mathcal{L}_{\mathcal{S}_i}$ );
12   $\mathcal{L}_{\mathcal{Q}_i} = \sum_{k_j, k'_j \in \mathcal{Q}_i} \text{sim}(k'_j, \mathcal{M}_i(k_j))$ ;
13 end
14  $\mathcal{L}_{\text{sum}} = \sum_{\mathcal{Q}_i \in \mathcal{Q}} \mathcal{L}_{\mathcal{Q}_i}$ ; Update( $\mathcal{M}$ ,  $\mathcal{L}_{\text{sum}}$ )  $\rightarrow \hat{\mathcal{M}}$ ;
15 // ❸ KPM Augmentation
16 for each  $k_i \in \mathcal{K}_s$  do
17    $\mathcal{D}_i^a = \bigcup (\mathcal{D}_i = \hat{\mathcal{M}}(k_i))$ ;
18 end
19 // ❹ Continual Tuning
20 Initialize  $\mathcal{D} = \mathcal{K}$  with dynamic maintenance;
21 Repeat step ❶ – ❸ with linear decay and append  $\mathcal{D}_i^a$  into  $\mathcal{D}$ ;
22 Update O-RAN AI models using  $\mathcal{D}$ ;
```

virtual O-RAN trace-driven emulation phase, KPMs are structurally separated into UE-specific and non-UE metrics. The dimensionality of non-UE metrics does not grow with the number of UEs. For UE metrics, the virtual O-RAN instantiates an independent, parallel operational emulator of fixed size for each active UE profile. During the meta-augmentation phase, we handle the input dimensionality by reshaping the trace matrices. Specifically, non-UE metrics are temporally aggregated into fixed-size summary tracking profiles, while UE metrics are mapped as independent, standalone sequence arrays into the batch dimension, *i.e.*, number of UEs times per-UE dimensionality.

A.2 Input schema of virtual O-RAN

Table 4 shows the detailed control knobs of the virtual O-RAN.

APIs	Input Schema
Topology	Graph architecture by adjacency matrix
Cell Config	Cell identities (cell_id, pci) Frequency bands (band) Adjacent cell (neighborCells)
RAN Unit Config	RU (Tx/Rx gain: [0-30]) DU (PRB scheduling: [default/xApp]) CU (Handover: [default/xApp])
xApp	A containerized xApp with control logic

Table 4: Control knobs of Virtual O-RAN**A.3 Detailed algorithm of Meta-Augmentation**

Algorithm 1 shows the detailed learning process of continuous meta-augmentation.

A.4 Wireless channel model

P_{tx} denotes the RU transmit power, and the path loss is defined as $PL(d(t)) = PL_0 + 10n \log_{10}(d(t)/d_0)$, with PL_0 representing the reference path loss at distance d_0 and n the path loss exponent. X_σ models log-normal shadowing as a Gaussian random variable with standard deviation σ , while $F(t)$ captures fast fading effects, modeled using Rayleigh/Rician distributions depending on NLoS/LoS dominant signal path. The noise floor is denoted by N_0 . Importantly, RANPILOT enables flexible modeling of UE mobility and deployment environments through customizable parameters: (1) *Mobility Modeling*: UE mobility is incorporated by modeling $d(t)$ as a time-dependent function (*e.g.*, $d(t) = v \cdot t$ for linear motion), allowing emulation of diverse mobility profiles. (2) *Environmental Setting*: Parameters n and σ are configurable to reflect different propagation conditions (detailed in Table 5). These parameters are initially estimated by RANPILOT to align with the KPM traces from the RAN recorder and subsequently adjusted to evaluate performance under varying mobility and environmental scenarios. Based on the computed SINR, the UE derives CQI and RSRP [2] for KPM reporting. This modeling approach supports sufficient fidelity of wireless channel dynamics required by AI model adaptation, while maintaining minimal computational overhead.

Environment	Path Loss n	Shadowing σ
Free Space (Ideal)	2.0	0–2
Indoor (LoS)	1.6–2.0	2–4
Indoor (NLoS)	3.0–5.0	4–8
Urban (Dense)	2.7–4.0	6–10
Suburban	2.0–3.0	4–6
Rural (Open)	2.0–2.5	2–4

Table 5: Configuration of wireless channel parameters**A.5 Sensitivity study of selective replay**

To evaluate the sensitivity of RANPILOT’s incremental learning module, we extend the anomaly detection experiment in Table 1 by varying the hyperparameters introduced in § 4.4.2, including KLD threshold τ , collection window t , replay buffer b , and window length l . We use the AI downtime required for the AI models to match Oracle performance as the metric for benchmarking incremental learning performance. By default, we set $\tau = 0.41$, $t = 10$, $b = 5$, $l = 6$.

Results. The impact of KLD threshold τ is illustrated in Figure 19(a), when the threshold is too low (*e.g.*, $\tau = 0.2$), the selection criteria become overly permissive, causing almost all post-reconfiguration traces to be ingested. This collapses the pipeline into conventional continual learning replay, which uniformly uses all collected data, diluting vital anomaly features and increasing AI downtime. Conversely, when the threshold is too high (*e.g.*, $\tau = 0.6$), the filter becomes overly restrictive, erroneously discarding representative abnormal data blocks. This deprives the incremental training set of critical post-reconfiguration dynamics and retards model convergence. The empirical results demonstrate a clear operational sweet spot at $\tau = 0.41$, minimizing AI downtime to approximately 1.3 minutes. At this optimal threshold, RANPILOT maximizes the selection of highly representative anomaly traces while precisely filtering

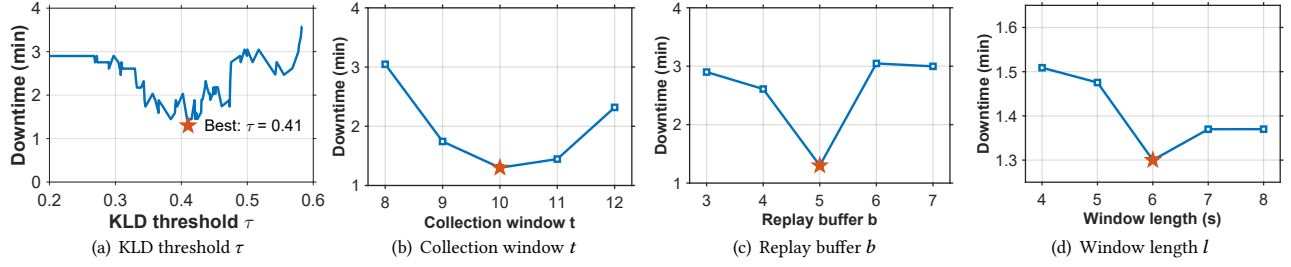


Figure 19: Sensitivity study of hyperparameters in incremental learning: downtime of anomaly detection model under different (a) KLD threshold τ ; (b) collection window t ; (c) replay buffer b ; and window length l .

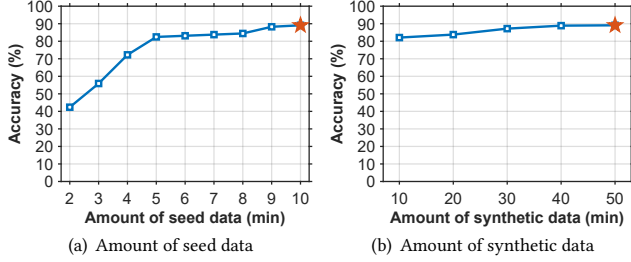


Figure 20: Impact of seed/synthetic data volume on the trained model accuracy.

out redundant normal data that does not contribute to model adaptation. In addition, the collection window t and the replay buffer size b collectively govern the density of useful patterns extracted from the real KPM trace. As shown in Figure 19(b,c), decreasing t or increasing b expands the data inclusion range, which dilutes the proportion of truly useful patterns and degrades downstream training efficiency. Conversely, increasing t or decreasing b narrows the selection range, potentially overlooking useful data patterns. Finally, Figure 19(d) shows the impact of the KPM segment sliding window length l . A short window fails to capture long-term KPM sequence dependencies, whereas a long window subsumes a large volume of normal data, ultimately flooding the underlying anomaly patterns and suppressing the KLD score.

A.6 Impact of seed/synthetic data volume

This experiment evaluates how the volumes of both seed and synthetic data affect downstream AI model adaptation performance. Extending the anomaly detection scenario from Table 1, we measure the plug-and-play detection accuracy of the models immediately upon deployment to benchmark AI adaptation performance. To assess the impact of seed data volume, we vary the duration of the baseline KPM traces used to drive the Virtual O-RAN data synthesis pipeline, while utilizing Meta-Augmentation to expand the total training set to a fixed 1-hour KPM trace (aligning with the previous experiment). To evaluate the impact of synthetic data volume, we fix the seed data to a 10-minute KPM trace and vary the duration of the synthetic data generated by the Meta-Augmentation module. **Results.** As illustrated in Figure 20(a), the plug-and-play accuracy increases sharply as the volume of the initial seed data scales upward. This trend occurs because brief seed windows fail to capture a representative distribution of baseline network traffic characteristics and scheduling dynamics. However, scaling the historical seed

trace to 9 minutes yields a significant performance knee, elevating the accuracy to an optimal plateau of around 89%. Beyond this threshold, further expanding the seed data volume yields diminishing returns. The results indicate that a proper seed KPM trace containing diverse and comprehensive environmental features is essential for effective data synthesis. Figure 20(b) depicts the impact of the generated synthetic data volume. Similar to the seed data dynamics, the plug-and-play accuracy climbs rapidly as the training set is populated with synthetic traces, crossing the threshold at 30 minutes of expansion. Beyond this 30-minute threshold, the accuracy increases only marginally. It demonstrates that further generative expansion provides diminishing performance gains once the target distribution space has been robustly synthesized.

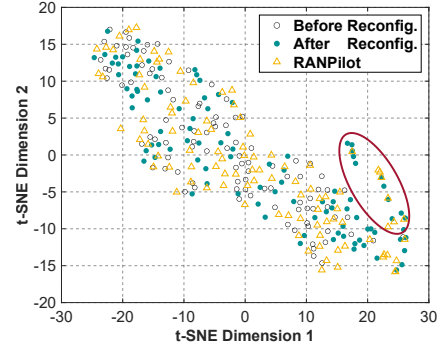


Figure 21: Visualization of KPM distribution.

A.7 Visualization of KPM distribution

To illustrate how RANPilot mitigates data drift and recovers the KPM distribution, we use t-SNE [83] to visualize the KPM trace's data distribution. We collect three sets of segmented KPM traces in the indoor cell addition experiment: (1) live traces before the reconfiguration; (2) live traces after the reconfiguration; and (3) synthetic traces generated by RANPilot.

Results. As illustrated in Figure 21, after cell addition, the distribution of most post-reconfiguration KPM traces aligns with the pre-reconfiguration traces, while a small portion of data exhibits severe drift (marked by the red circle). The pre-reconfiguration seed traces are entirely absent from this region, meaning that a model trained purely on raw seed data cannot handle these network variations. RANPilot effectively mitigates distribution drift by emulating RAN behavior and synthesizing KPM traces, thereby proactively adapting downstream AI models to upcoming out-of-distribution KPM variations.