SAPIENZA
UNIVERSITÀ DI ROMA

Northeastern
University

Exploring Neural Trojan persistency in Programmable Cellular Networks

A Research Platform for Neural Backdoor Attack (BA) in Open Radio Access Network (O-RAN)

Simone Palumbo

Student ID: 1938214

Advisors

Prof. Ioannis Chatzigiannakis (Sapienza University of Rome)
Prof. Francesca Cuomo (Sapienza University of Rome)
Prof. Tommaso Melodia (Northeastern University)

Co-Advisors

Dr. Andrea Lacava, Dr. Leonardo Bonati, Prof. Salvatore D'Oro
(Northeastern University)

MSc in Artificial Intelligence and Robotics, Sapienza
Academic Year: 2024-2025

Exploring Neural Trojan persistency in Programmable Cellular Networks
Master's Thesis. Sapienza University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: palumbo.1938214@studenti.uniroma1.it

Abstract

We present the first defense framework for detecting and mitigating backdoor attacks (BA) in data-driven assets of programmable networks, with a particular focus on the O-RAN ecosystem. Furthermore, we introduce a wireless research platform designed to advance Adversarial Machine Learning (AML) research in O-RAN, leveraging real-world testbeds and addressing its unique challenges, namely, the black-box nature of components, Near-Real-Time (Near-RT) dynamics, non-stationarity, and the inherent stochasticity of the wireless interface. Unlike prior AML studies in O-RAN, our framework integrates both supply-chain (pre-deployment) and runtime (post-deployment) defenses. In the pre-deployment stage, we focus on White Box (WB) scenarios, where models and datasets downloaded from the Internet can be inspected and sanitized. At runtime, we propose a Near-RT Black Box (BB) Anomaly Detection (AD) capable of identifying potential backdoor activations, mitigating the risk of latent trojans being triggered under adversarial conditions. BB scenarios are particularly common in O-RAN, where models are distributed as binaries or container images, making direct purification or pruning infeasible. In such cases, detection becomes the only viable defense strategy. Unlike prior work in O-RAN AML, our framework is validated using real Over The Air (OTA) deployments and the *Colosseum* testbed, the world's largest wireless network emulator, demonstrating its practicality under realistic conditions. Overall, this work represents a step toward an Artificial Intelligence (AI) Security Assurance Specification (SCAS) framework for programmable cellular networks, raising awareness of ML lifecycle vulnerabilities and emphasizing the urgent need for robust security measures.

Acknowledgements

To those who believe in me selflessly,
to those who have been patient with me,
and to those who have shared their enthusiasm with me.

Ad Maiora!

May the wonder never fade.

Contents

Chapter 1

Introduction

As our society depends more and more on cellular networks, it becomes critical to assure that the networks are secure against cyber attacks. Next-generation cellular networks are expected to rely on Machine Learning (ML) algorithms to achieve real-time resource optimization across space, time, frequency and devices. In this thesis, we studied the security threats to those ML algorithms and develops solutions to protect them, focusing on the Open Radio Access Network (O-RAN) architecture which is rapidly becoming widespread.

O-RAN meets the growing demand for heterogeneous services in 5th Generation (5G) and beyond, such as enhanced Mobile Broadband (eMBB), Ultra-Reliable Low-Latency Communications (URLLC), and massive Machine-Type Communications (mMTC), by enabling large-scale, automated, and data-driven control of network functionalities. At its core lies the RAN Intelligent Controller (RIC), which supports real-time network optimization through Artificial Intelligence (AI)-powered applications known as xApps and rApps, responsible for tasks like traffic steering, load balancing, and interference mitigation. However, integrating AI into critical network control functions introduces new security challenges, including Adversarial Attack (AA), Data Poisoning (DP), Backdoor Attack (BA), and privacy threats such as model inversion and membership inference.

This project aims to define threat models specific to the O-RAN ecosystem and to develop security mechanisms that safeguard its AI/ML lifecycle. The proposed approach aligns with the Zero Trust (ZT) principles adopted in O-RAN, textitasizing model integrity verification, mitigation of malicious behavior transfer, and risk-based defense prioritization. While Adversarial Machine Learning (AML) has been extensively studied in fields like Computer Vision (CV) and Natural Language Processing (NLP), its implications for telecom-grade AI systems remain largely unexplored. Bridging this gap is the core objective of this research: designing AI for O-RAN that is not only performant but also inherently resilient.

A recent *McKinsey* and *Mozilla Foundation* report [1] indicates that 72% of technological organizations employ open-source AI models, reflecting their widespread adoption. According to a study by *JFrog* [2], out of over a million models on Hugging Face, 400 contained malicious code.

In the O-RAN ecosystem, telecom equipment vendors and RIC platform providers on one hand (*Nokia*; *Ericsson*; *Samsung*; *Mavenir*, ...) and Mobile Networks Op-

erators (MNOs) deploying these technologies on the other (*AT&T* in USA; *NTT DOCOMO* in Japan; *Deutsche Telekom* in Germany; *Telefónica* in Spain; *Vodafone* in UK, ...) must design resilient platforms and processes to withstand adversarial pressure. That's in their interest, also for competitiveness reasons. Imagine the following scenario: one of these major Mobile Network Operator (MNO) runs a multi-vendor O-RAN with a RIC hosting third-party xApps. The operator onboards a closed-source "adaptive traffic steering" xApp from a reputable marketplace. The xApp's model was trained by an upstream vendor using mixed open-source components; unknown to both marketplace and operator, a subcontractor injected a backdoor during training. The trigger to activate the backdoor is a rare combination of Key Performance Measurement (KPM) that causes the model to over-prioritize handovers toward cells owned by Vendor V (a competitor's Radio Access Network (RAN) footprint in a shared-RAN region). Under normal conditions the xApp behaves well and passes acceptance tests; the trigger never appears in the operator's standard validation dataset. The activation happens during a high-profile event, an adversary is able to send the trigger by controlling a small fleet of users or by poisoning data directly in the RIC database. The induced KPM pattern makes the xApp's model flips into the backdoored decision mode, skewing steering and load-balancing actions. Cells belonging to the attacker-favored footprint see disproportionate offload; competing sectors absorb more interference and congestion, degrading Quality of Service (QoS) for premium subscribers. This escalates into Service Level Agreement (SLA) penalties and customer complaints.

This scenario illustrates why O-RAN's openness and competitive dynamics amplify supply-chain risk, and motivates the efforts of this work.

We demonstrate that companies can defend from these threats, leveraging on the Zero Trust Architecture (ZTA) of O-RAN, creating security pipelines that flag inconsistencies.

The gap we bridge is the absence of a telecom-grade, end-to-end treatment of adversarial ML for O-RAN. We work with multivariate Time Series (TS) KPM, while most of the work focuses on images/text data, and work in O-RAN mnostly focuses on Evasion Attack (EA), namely AA, working at inference time rather than training. We believe we are the first to consider the operational constraints of O-RAN that matter in practice: black-box xApp onboarding, stochastic and non-stationary radio conditions, Near-Real-Time (Near-RT) requirements, TS-specific reasoning and distribution shifts, under which the Independent and Identically Distributed (IID) assumption typical in ML does not work. In the current State of the Art (SotA), tooling for repeatable, pre-/post-deployment security evaluation is also limited. To close these gaps, we (i) experimentally demonstrate when and how BA succeed against RIC-hosted models under realistic O-RAN constraints; (ii) introduce a wireless research platform to foster future studies in AML for O-RAN; (iii) propose both White Box (WB) and Black Box (BB) defense mechanisms, tailored to the specific operational scenarios of O-RAN; and (iv) outline preliminary directions for developing non-stationarity- and stochasticity-aware AML algorithms. The overarching goal is to provide evidence and practical tools for building ZT pipelines that verify model integrity, contain malicious behaviors, and recover gracefully when anomalies are detected.

The remainder of this thesis is organized as follows. Chapter 2 introduces the

necessary background, following the hierarchical nested structure illustrated in Figure 1.1. We begin with an overview of cellular networks and their evolution (2.1), then describe the O-RAN architecture (2.2), including the main tools used by the community to experiment with programmable and wireless networks. Next, we discuss the role of AI and ML within O-RAN (2.3), and finally examine the security threats targeting these data-driven assets, along with the corresponding defense mechanisms (2.4). Chapter 3 presents our contributions, while chapter 4 details the experimental methodology and evaluation: testbeds and datasets, attack implementations and threat models, defense configurations and metrics. Chapter 5 concludes with key findings, limitations, and directions for future work. Appendices A.1 and A.2 collect further background and details.

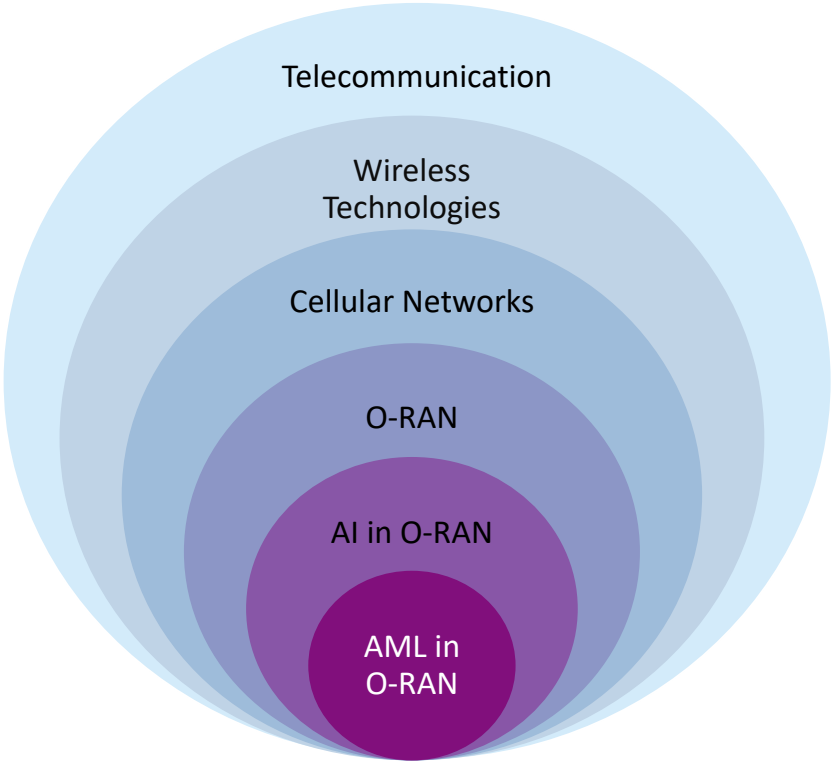


Figure 1.1. Nested representation of the research domains addressed in this thesis, starting from the broad field of telecommunications and narrowing down to the specific focus of AML in O-RAN. General background on cellular networks is provided in Section 2.1, while the O-RAN architecture is discussed in Section 2.2. The use of data-driven models in O-RAN is presented in Section 2.3, followed by an analysis of their vulnerabilities and threats in Section 2.4. The broader topics of telecommunications and wireless technologies are intentionally left out in order to maintain focus on the objectives of this thesis.

Chapter 2

Background and Related Work

In this chapter we surveys the necessary background in programmable, wireless, open, cellular networks, specifically tackling the O-RAN architecture. We also surveys how data-driven models are used in O-RAN and their vulnerabilities, threats and security measures.

2.1 Cellular Networks

Cellular networks are wireless telecommunications network split geography into small "cells", each served by a base station, called Base Transceiver Station (BTS), that links nearby devices over radio. By reusing frequencies in non-adjacent cells and coordinating handovers between neighboring base stations, the network maintains service as users move while limiting co-channel interference. Multiple-access schemes such as Frequency Division Multiple Access (FDMA); Time Division Multiple Access (TDMA); Code Division Multiple Access (CDMA); Orthogonal Frequency Division Multiple Access (OFDMA), together with antenna-related techniques such as Multiple Input Multiple Output (MIMO) and beamforming[1] let many users share limited spectrum efficiently [3]. User traffic reaches the core via backhaul: voice is routed to the Public Switched Telephone Network (PSTN), while data goes to the Internet [4].

2.1.1 Cellular Networks Generations

Mobile networks have evolved through multiple generations, with each of them bringing significant improvements in technology and performance. Roughly every decade a new generation is introduced, operating on new radio interfaces and frequency bands and offering higher data rates and capabilities [4].

1st Generation (1G) was analog, with circuit-switched voice (low capacity), while 2nd Generation (2G) was already fully digital, introducing spectrum-efficient TDMA/CDMA access, encryption, and the first data services: Short Message Service

[1] Beamforming is a signal processing technique to direct the transmission or reception of signals in specific directions (a beam), adjusting phase and amplitude of the signal at each antenna element in an antenna array.

(SMS), then General Packet Radio Service (GPRS). 3rd Generation (3G) then delivered mobile broadband and multimedia by moving to wideband CDMA radios and a packet core alongside legacy circuit-switched voice, pushing speeds into the multi-Mbps range. 4th Generation (4G) Long Term Evolution (LTE) completed the all-Internet Protocol (IP) transition: OFDMA radios, flatter eNodeB (eNB)-centric RAN, and the Evolved Packet Core (EPC) enabled tens to hundreds of Mbps, low latency, and Voice over LTE (VoLTE). 5G extends this with New Radio (NR) across sub-6 GHz and mmWave, massive MIMO and beamforming, ultra-low-latency targets, and support for diverse service classes: eMBB; URLLC and mMTC, delivered over Non Stand Alone (NSA) (LTE-anchored) and Stand Alone (SA) (native 5G core) deployments with network slicing. *Network Slicing* refers to the creation of multiple virtual, end-to-end logical networks, called slices, on top of the same physical 5G infrastructure (See Figure 2.1). In the case of eMBB/URLLC/mMTC slicing, these are the key characteristics: eMBB is high bandwidth, used for video streaming and file transfer, URLLC is low latency, used for live video chat and phone call, mMTC is for high connection density (i.e., many devices), for instance for background communcation from many devices. eMBB is consistently high Throughput (TH) on download, long queues, little uplink traffic, can be bursty; URLLC is regular transmissions in both directions, very short queues, must be regular, voice has small TH while video has larger, while mMTC irregular TH, very short queues, small TH, uplink and downlink [4]

Finally, 6th Generation (6G) research targets Tbps-class links at sub-THz bands, near-instantaneous latency, tight integration of communications and sensing, and AI-native, energy-efficient networks, envisioning ultra-dense infrastructures that push wireless performance to practical limits. In Fig. 2.2 a visual overview of this evolution is presented.

2.1.2 Components of a Cellular Networks

Regardless of generation, all cellular networks share a general architectural design. Namely, it can be divided in RAN and CN. The RAN covers the BTS tower that communicate with the User Equipment (UE), which is the mobile device itself, in an Over The Air (OTA) manner. RAN is not the only type of Access Network (AN) that exists; for the sake of completeness, we present an overview of the main access technologies in Fig 2.3[2].

The CN, on the other hand, manages high-level routing, switching, and connectivity to external networks (like the internet or PSTN, see the physical infrastructure in Figure 2.1)[3].

Each UE has a connection both with the gNodeB (gNB), thanks to the Access

[2]It should be noted that these technologies are not mutually exclusive. For instance, Digital Subscriber Line (xDSL) and Fiber to the x (FFTx) typically deliver connectivity to the user's premises, while IEEE 802.11 (Wi-Fi), Ethernet, and Bluetooth connect end devices within the local environment. Access technologies can be further categorized into last-mile technologies (i.e., Wide Area Network (WAN)), such as xDSL and FFTx, which connect users to the Internet, and local access technologies (i.e., Local Area Network (LAN)/Personal Area Network (PAN)), which interconnect devices within a home or office network.

[3]Since is not particularly relevant for this work, we neglect describing the architecture and interfaces of the CN.

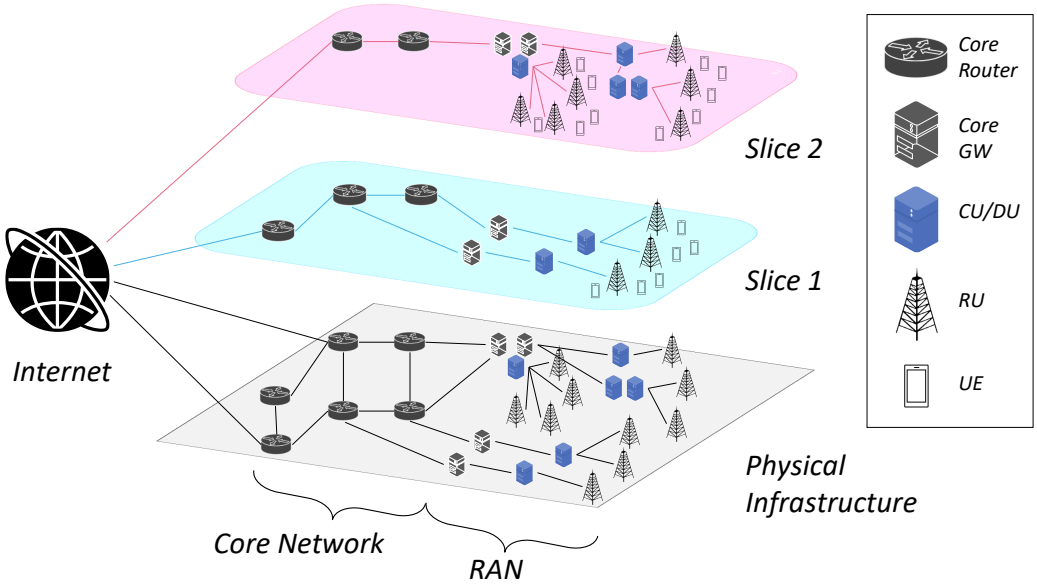


Figure 2.1. An example of RAN and Core Network (CN) slicing. While RAN slicing partition the RAN into multiple logical networks tailored to different service types, CN slicing partition the 5G CN into logical slices that manage separate control and data plane functionalities for different services or tenants. Each slice can have its own instance of a virtual component (Access and Mobility Management Function (AMF), Session Management Function (SMF), User Plane Function (UPF) etc.), enabling per-slice specific traffic routing, security, subscriber policy, mobility management etc. For true End-to-End (E2E) slicing, both RAN and CN slicing must be aligned: together they allow a slice to be optimized for a specific service (e.g., a low-latency, high-security factory automation slice).

Stratum (AS), and with the CN, thanks to the Non-Access Stratum (NAS). To learn more about AS and NAS refer to Appendix A.2.

2.1.3 RAN Evolution

Since the earliest phases of 5G NR, there has been a push to *softwarize*, *disaggregate*, *virtualize*, and *open* the RAN [5]. Early cellular systems (from 2G onward) packaged radio and baseband as vendor-specific, largely monolithic base stations. Modern *programmable networks* separate the underlying physical hardware from the control software of a device and break the traditional base station[4], into Radio Unit (RU), Distributed Unit (DU), and Centralized Unit (CU) to gain flexibility and multi-vendor interoperability, as shown in Figure 2.4. One possible protocol stack split is the following: the RU at the site handles Radio Frequency (RF) and lower-Physical (PHY), converting between analog RF and digital In-phase and Quadrature (IQ) samples. The DU software, typically on Commercial Off The Shelf (COTS) servers close to the RU to meet latency, runs high-PHY, Medium Access Control (MAC), and Radio Link Control (RLC). The CU runs Packet Data Con-

[4]More precisely, the Base Band Unit (BBU), namely the component of the base station that is responsible for processing digital signals and performing modulation and demodulation

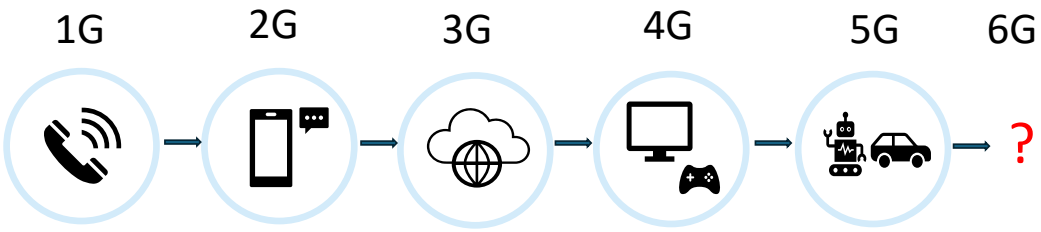


Figure 2.2. Evolution of cellular generations. 1G (analog voice), 2G (digital voice and SMS), 3G (mobile Internet), 4G (all-IP broadband), 5G (powering robots, V2X, and IoT), and the open research trajectory toward 6G.

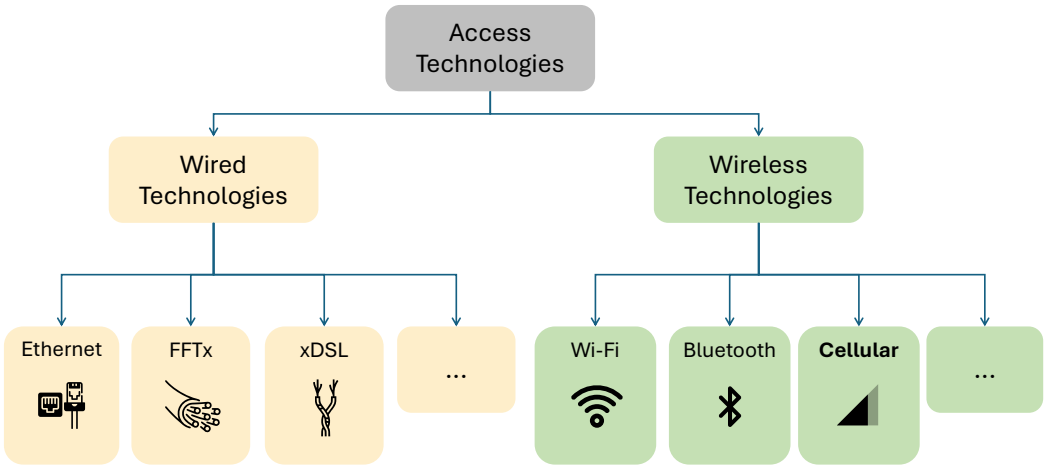


Figure 2.3. Overview of major access technologies.

vergence Protocol (PDCP)/Service Data Adaptation Protocol (SDAP) (user plane) and Radio Resource Control (RRC) (control plane), with an internal *E1* interface between CU-UP and CU-CP. The *F1* interface connects CU and DU (*F1-C* for control plane, *F1-U* for user plane) [6]. To learn more about the cellular protocol stack, refer to A.2.

In this disaggregated environment, co-location of CU and DU on the same server is also possible when transport constraints favor it. This split was operated by 3rd Generation Partnership Project (3GPP), and was considered from the beginning of writing its specifications [7].

There is not just a single possible *functional* splits, but there are many way to distribute the layers in the units. The main functional splits are shown in 2.5. The chosen functional split and placement determine performance and cost trade-offs. Higher functional splits are more desirable for capacity use cases in dense urban areas while lower functional splits will be the optimum solutions for coverage use cases. So, while lower functional splits utilize less than perfect fronthauls, there is

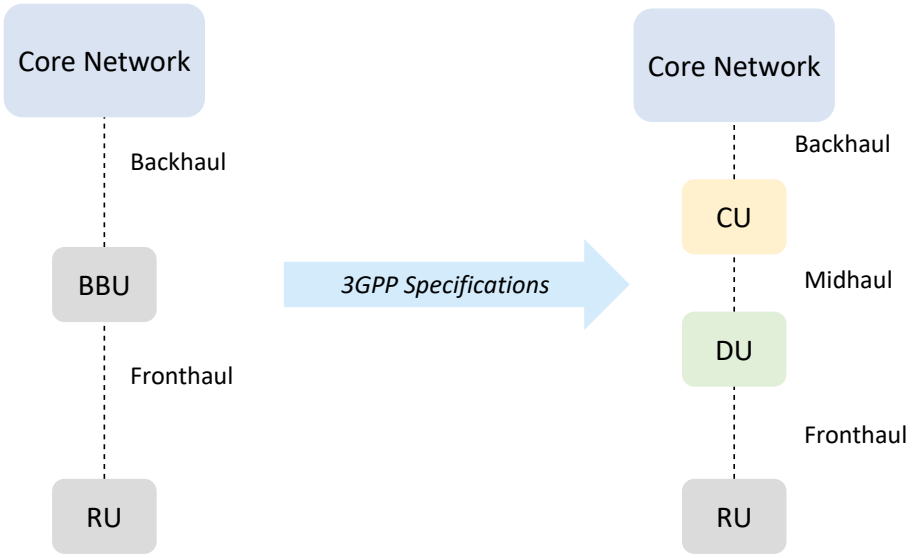


Figure 2.4. 3GPP effort for disaggregation of the RAN is at the foundation of O-RAN Alliance efforts.

a greater dependence on fronthaul performance for higher functional splits.

When it comes to O-RAN, the *O-RAN Alliance*, the industry consortium standardizing O-RAN, has defined a multi-vendor fronthaul interface between DU and RU based on *Split 7-2x*. In O-RAN terminology, RU is denoted as O-RAN Radio Unit (O-RU), DU is denoted as O-RAN Distributed Unit (O-DU) and CU is denoted as O-RAN Central Unit (O-CU).

In the next, we will describe O-RAN and understand more in depth how this paradigm shift enabled AI in the loop. However, it is important to notice that the shift from a monolithic black-box approach to O-RAN was not immediate, but other paradigms. Distributed Radio Access Network (D-RAN), Virtualized Radio Access Network (V-RAN), Cloud Radio Access Network (C-RAN), were ideated and developed [8]. A visual explanation of this evolution is in Figure 2.6 and Figure 2.7

2.2 O-RAN

The O-RAN ALLIANCE was founded in 2018 by *AT&T*, *China Mobile*, *Deutsche Telekom*, *NTT DOCOMO*, and *Orange*, later joined by other major operators worldwide. Its goal is to define a new type of RAN, building on the functional split work already carried out by 3GPP. The O-RAN specifications introduce two types of RIC, network controllers capable of hosting custom applications[5]: the Near-Real-Time RAN Intelligent Controller (Near-RT RIC), which operates on near-real-time scales (from 10 ms to 1 s), and the Non-Real-Time RAN Intelligent Controller (Non-RT

[5] A proposal has also been made to standardize *dApps*, applications running in the O-CU/O-DU for ultra-low-latency use cases. However, since this proposal is not yet part of the standard, we do not consider them in this thesis.

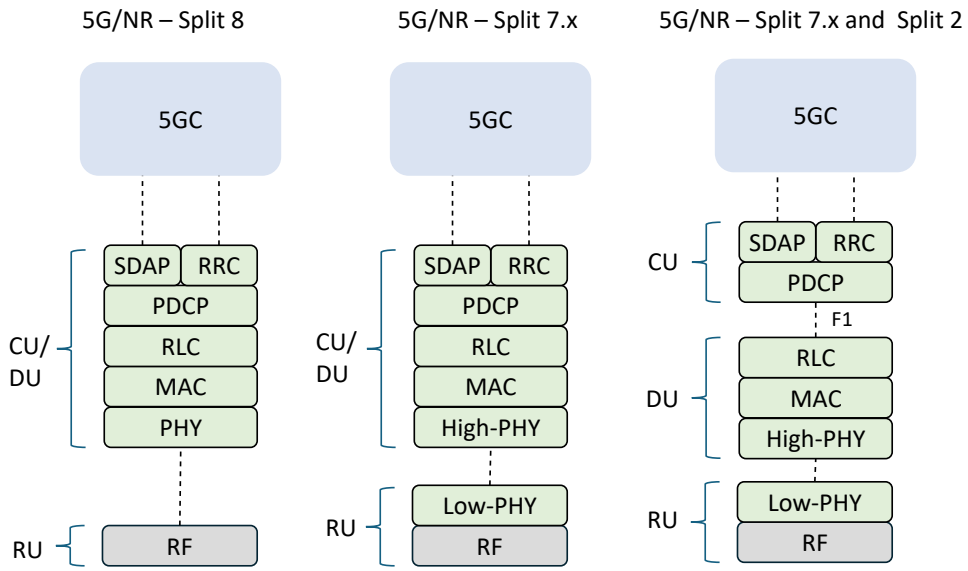


Figure 2.5. RAN functional splits. Split 7.x shown is precisely 7.2x. Split 7.2x + Split 2 is the practical O-RAN deployment, with F1 interface being the middlehaul between CU–DU (Split 2).

RIC), which operates on non-real-time scales (above 1 s). The former is discussed in 2.2.1, together with details on the E2 interface, the main interface considered in this work, while the latter is presented in 2.2.2, in the context of the Service Management and Orchestration (SMO). A visual overview of the O-RAN disaggregated cellular architecture is shown in Figure 2.8.

2.2.1 E2 interface and the Near-RT RIC

The E2 interface enables communication between the gNB and the Near-RT RIC and is divided into two components: the E2 Application Protocol (E2AP) and the E2 Service Model (E2SM) [5]. E2AP provides the basis for communication and governs how subscriptions are created, managed, and terminated. In this process, an xApp issues a RIC subscription request, which represents a stateful contract that instructs the near-RT RIC to deliver indications for a specific service model at a defined cadence or when a certain condition is met. The subscription request carries several key elements: a RIC request identifier that uniquely identifies the subscription instance, a RAN function identifier that indicates which service model is being referred to, an event-trigger definition that specifies when or how frequently reports should be generated, and an action list describing the type of messages to be sent back, such as REPORT, INSERT, or POLICY. Once the subscription request is received, the E2 node in the gNB responds with either a subscription response or a failure message, confirming whether the request has been understood and the corresponding local state has been allocated. After successful establishment, the gNB continues to send RIC indication messages of type REPORT, each carrying the service-model-specific payload, until a subscription delete request is issued.

While E2AP defines the mechanics of subscription and message exchange, E2SM

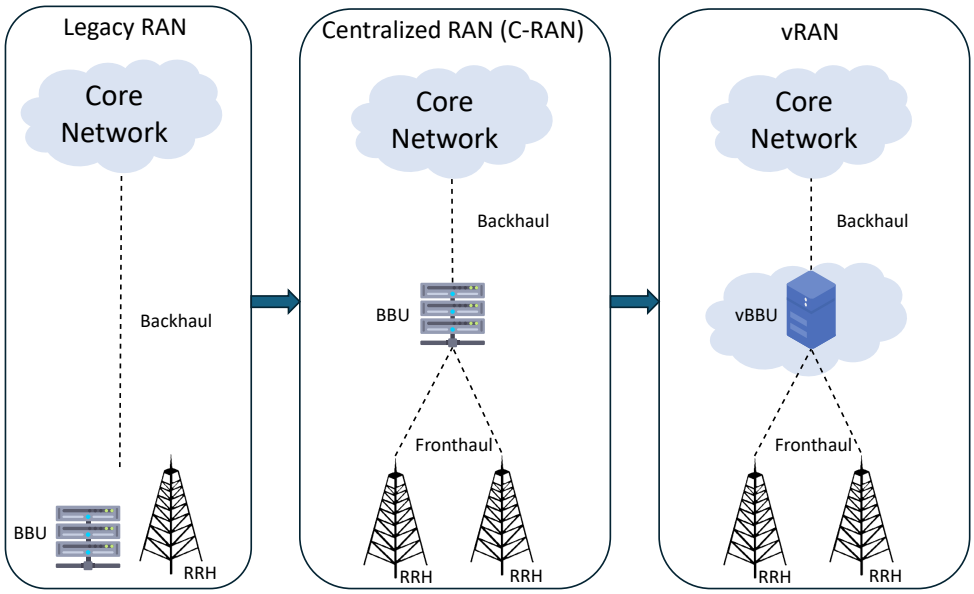


Figure 2.6. From left to right: (i) Legacy RAN with a dedicated BBU per site connected to the core over the backhaul and driving a local RRH; (ii) Centralized RAN (C-RAN) that pools BBUs in a central location and serves multiple RRHs over a high-capacity fronthaul; (iii) vRAN, where baseband functions are virtualized (vBBU) on COTS servers while keeping the same backhaul/fronthaul topology.

defines the content. Each service model is a plug-in that specifies one particular topic of information and provides its Abstract Syntax Notation One (ASN.1) schema for encoding and decoding. The O-RAN Alliance defines a set of standard service models that ensure interoperability (O-RAN-compliant Service Model (SM)) [9], while any additional models developed outside this framework are considered customized.

2.2.2 SMO and the Non-RT RIC

O-RAN also introduces the SMO. The SMO encompasses Management Functions (MFs) introduced by the O-RAN ALLIANCE, as well as those defined by other Standards Development Organizations (SDO) such as 3GPP and European Telecommunications Standards Institute (ETSI) [10]. Acting as a central nervous system, the SMO manages Network Funtions (NFs), orchestrates and optimizes resources across gNBs, and supervises the Non-RT RIC, alongside other components. Ongoing work focuses on the intelligentization and automation of the SMO [10], with the aim of enhancing Management and Orchestration (M&O) capabilities. Key research challenges include:

- Data privacy, resilience, robustness, and reliability: as the SMO can represent a single point of failure, it must be designed to tolerate faults and attacks (AML too).
- Elasticity: models must be retrained to adapt to evolving network conditions and user requirements.

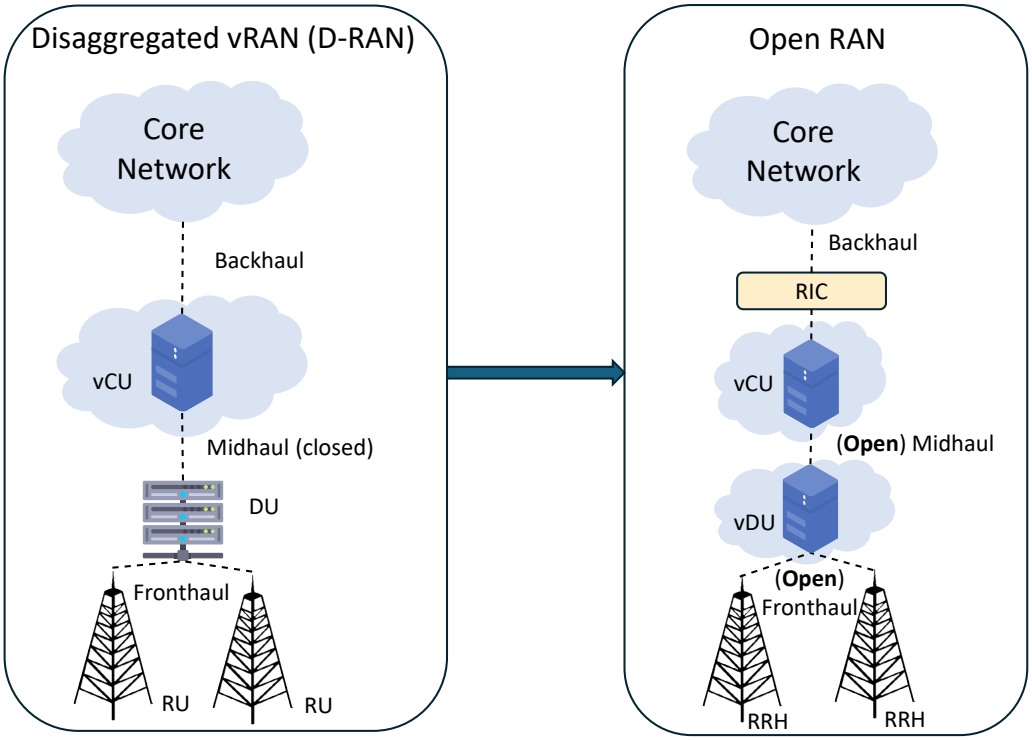


Figure 2.7. Left: Disaggregated vRAN (an evolution of vRAN itself) splits the BBU into a virtualized Central Unit (vCU) and a Distributed Unit (DU), connected by a (typically vendor-specific) midhaul and driving RUs over the fronthaul. Right: Open RAN (O-RAN) standardizes the midhaul/fronthaul interfaces (open F1/7.2x) and introduces the near-RT RIC, enabling multi-vendor interoperability and intelligent control.

.

- Agility: training with heterogeneous data from different domains is time-consuming, while network conditions may change rapidly at a local scale. Incremental and adaptive learning techniques are required to ensure models can quickly adapt to dynamic environments.
- Security of centralized learning: centralized training is inherently vulnerable to compromised raw data [11] and it necessitates robust data validation and filtering mechanisms.
- Signaling efficiency: the aggregation of large volumes of raw data leads to significant signaling overhead.

A simplified architecture overview of the SMO framework is shown in Figure 2.9.

2.2.3 O-RAN ZTA

Because O-RAN is disaggregated and open, perimeter defenses are not enough. The architecture adopts ZT ("never trust, always verify") with authenticated/authorized interactions across interfaces, micro-perimeters around assets, mutual authentication, and least-privilege access [12]. O-RAN security aligns with industry guidance

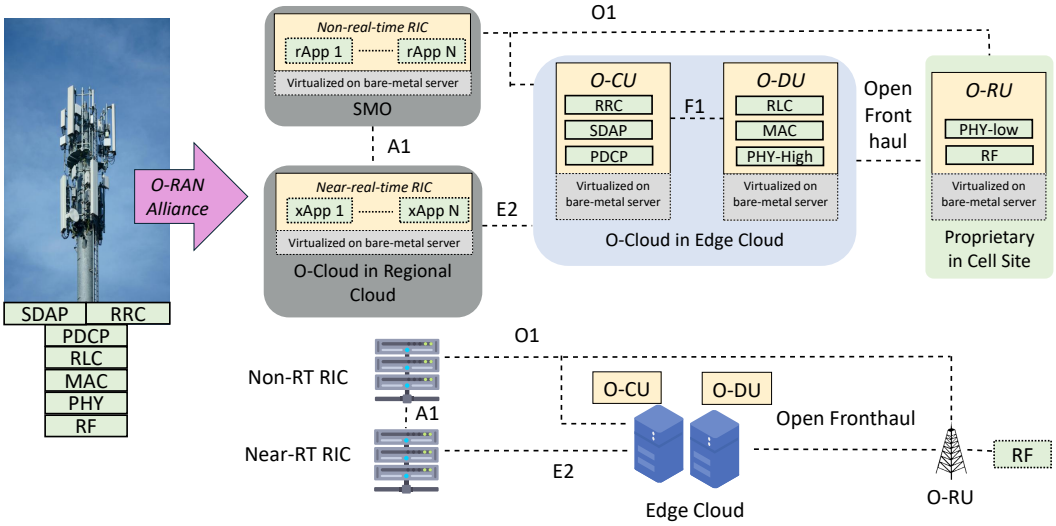


Figure 2.8. Overview of the disaggregated O-RAN architecture. The architecture separates the base station into functional units interconnected via standardized interfaces (F1, E2, and Open Fronthaul). The Near-RT RIC is hosted in the regional cloud and manages Near-RT control through xApps, while the Non-RT RIC is hosted in the SMO and manages Non-Real-Time (Non-RT) control through rApps. Both controllers interact with the RAN through the E2, O1 and between them with A1, enabling programmability and intelligence in the network. For the sake of this thesis, we will focus mainly on the E2 interface.

(e.g., NIST SP 800-207 [13]), promotes secure-by-design practices, and requires transparency of software components (e.g., Software Bill of Materials (SBOM)). The same approach must be extended to the AI/ML Workflow (WF): accessible data, like the one in Shared Data Layer (SDL) of the controllers, must be access-controlled and audited; xApps/rApps must be limited to only the data they need; the pipeline must assume adversarial risk ([14], 6.2 Open Challenges section) and model outputs must not blindly trusted, with continuous runtime monitoring and alerting guard against inference, time anomalies or misbehavior.

2.2.4 Distribution Shifts in O-RAN

In O-RAN traffic evolves and radio conditions drift with mobility and operators retune policies. Collectively, these effects induce *distribution shift* [15]: the statistical mismatch between the training (source) distribution $\mathcal{P}$ and the deployment (target) distribution $\mathcal{Q}$. We use the term broadly to include three canonical forms: *covariate* [15], *label* [15] and *concept* [16] shift. More mathematically:

Definition 1 (Definition of Distribution Shift [15]) *A distribution shift can be seen as a movement from a source distribution $x_{\mathcal{P}} \sim \mathcal{P}$ to a target distribution $x_{\mathcal{Q}} \sim \mathcal{Q}$ [15]. Since any dataset is generated from a joint distribution $P(x, y)$ of features x and labels y, we can characterize the three types of distribution shift by*

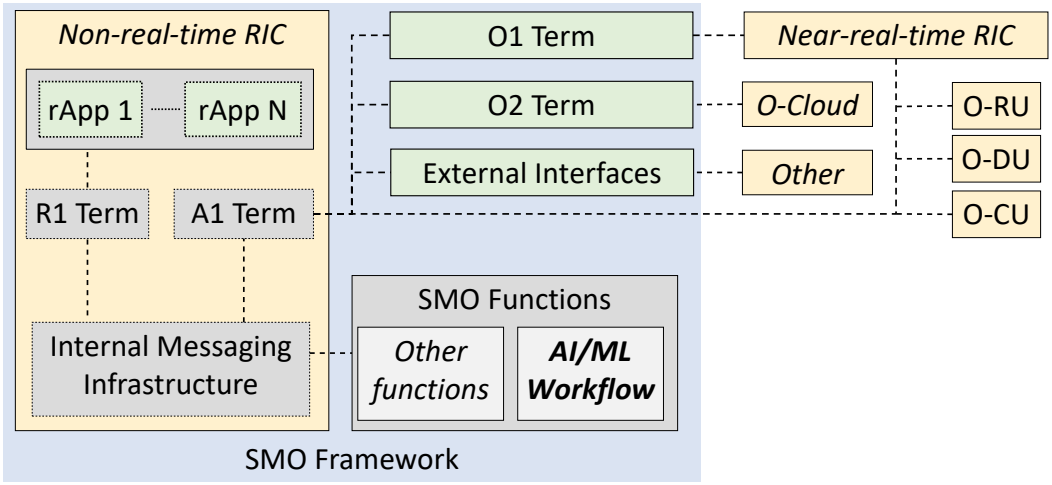


Figure 2.9. Simplified architecture of the SMO framework.

breaking down $P(x, y)$ into conditional factorizations $P(y|x)P(x)$ and $P(x|y)P(y)$ [15] [17] [18]. More precisely [19]:

- *In covariate shift $\mathcal{P}(x) \neq \mathcal{Q}(x)$ but $\mathcal{P}(y|x) = \mathcal{Q}(y|x)$. The way the world generates the labels, and so the task, is the same, but the input distribution you see changes.*
- *In label shift $\mathcal{P}(y) \neq \mathcal{Q}(y)$ but $\mathcal{P}(x|y) = \mathcal{Q}(x|y)$. The way features correspond to a class is unchanged, but the class proportions are different.*
- *In concept (conditional) shift the definition of the problem itself evolves, meaining either*
  - *(A) $\mathcal{P}(x) = \mathcal{Q}(x)$ but $\mathcal{P}(y|x) \neq \mathcal{Q}(y|x)$. The mapping from input to labels changes, namely the task changes. However, the input distribution is the same.*
  - *(B) $\mathcal{P}(y) = \mathcal{Q}(y)$ but $\mathcal{P}(x|y) \neq \mathcal{Q}(x|y)$. The same set of classes is present, but the way each class manifests in terms of features has changed.*

In O-RAN, these manifest naturally. To give you an example for each type:

- Covariate shift arises from diurnal load cycles, new UE mobility patterns, weather and seasonal propagation effects, or a different KPM sampling cadence
- Label shift arises when, for a slice classifier, the number of slices changes.
- Concept shift arises when, in a traffic classifier, the same KPM no longer imply the same traffic class or QoS level due to an change in the protocols used, usage patterns, or network parameters. Maybe you have a model learning the mapping high Modulation and Coding Scheme (MCS) high TH, low MCS low TH. However, the operator changes numerology μ and the same MCS now yields different Transport Block Size (TBS) values, because the resource grid changes: now the mapping is different.

Distribution shifts are widely investigated in other domains. When it comes to CV, [20] offers a benchmark of 10 datasets reflecting a diverse range of distribution shifts that naturally arise in real-world applications.

Detect Distribution Shift

When it comes to detection, there exists standard methods for univariate detection, while multivariate is still maturing [21].

Definition 2 (Distribution Shift Detection [15]) *Given a reference (source) distribution $\mathcal{P}$ and a query (target) distribution, $\mathcal{Q}$ distribution shift detection is the problem of detecting when $\mathcal{Q}$ is different from $\mathcal{P}$ [19].*

Distribution Shift detection can be performed by means of dimensionality reduction techniques [21], joint distribution and localization techniques [19].

Detect through Dimensionality Reduction [21] Kernel-based multivariate two-sample tests (e.g., Maximum Mean Discrepancy (MMD)) are widely used to check whether two high-dimensional distributions differ by comparing their mean embeddings in an Reproducing Kernel Hilbert Space (RKHS). However, they scale poorly with dataset size and suffer from degraded statistical power in high dimensions [22]. To address this, dimensionality reduction techniques[6] (such as Principal Component Analysis (PCA), Black-Box Shift Detection (BBSD) [23] or a Domain Classifier) can be applied before testing [21]. It should be noted that these two-tier methods are generally unable to understand if a specific sample is Out-of-Distribution (OOD), capturing instead top-level shift dynamics [21]. The current SotA is not handling online data and TS properly, namely not accounting for the high degree of correlation between adjacent time steps.

Detect through Localization Dimensionality reduction and joint distribution tests provide only binary information (shift occurred or not) and work mostly in supervised settings, without indicating which features shifted. In contrast, feature shift localization methods identify the specific subset of features whose distributions have changed, and can operate in unsupervised settings.

Definition 3 (Feature Shift Detection Problem [19]) *Given reference and query distributions $(\mathcal{P})$ and $(\mathcal{Q})$, and corresponding samples $(X \sim \mathcal{P})$ and $(Y \sim \mathcal{Q})$, the goal is to detect which features exhibit a change. For each feature (j), this is done via a conditional distribution hypothesis test:*

$$H_0 : P(x_j|x_{-j}) = Q(x_j|x_{-j}) \quad vs. \quad H_A : P(x_j|x_{-j}) \neq Q(x_j|x_{-j}) \tag{2.1}$$

where (x_{-j}) denotes all other features.

[6] Dimensionality-reduction methods are generally surjective, meaning multiple inputs can map to the same output, leaving open the possibility of pathological cases where input distributions shift but low-dimensional representations remain fixed. However, such case is unlikely in non-adversarial settings, but this means that these methods are *not defenses against worst-case adversarial attacks*

This test captures inter-feature dependencies, making independent manipulations detectable, under the assumption that sensors have no causal relations and that compromised features remain so over time.

The discrepancy between conditional distributions is quantified via the Expected Conditional Distance (ECD):

$$\gamma = \mathbb{E}_{x_{-j}}[\phi(\mathcal{P}(x_j|x_{-j}), \mathcal{Q}(x_j|x_{-j}))] \tag{2.2}$$

where ϕ is a statistical divergence (Fisher divergence being most effective). Conditional distributions can be estimated through either model-free (e.g., k-Nearest Neighbors (k-NN)) or model-based (e.g., Gaussian Mixture Models (GMM)) approaches.

Detection performance depends on the inter-feature mutual information: attacks are harder to detect when sensors are independent (low mutual information) and easier when they are strongly correlated (high mutual information). The approach can be extended to TS by performing tests at regular intervals, detecting both *which* features shifted and *when*. Using Fisher divergence and time-dependent bootstrapping allows near-real-time operation and threshold adaptation to natural temporal drift [19].

Explain Distribution Shift

Explaining distribution shifts can help an operator understand what changes between the environments, thus allowing for more effective reactions to the shift.

Definition 4 (Explaining a Distribution Shift [15]) *A distribution shift can be explained using a transport map $T(x)$ that maps a point from the source to the target [15].*

Qualitatively you can see it as the approximation of how $\mathcal{P}$ moved in the distribution space to become $\mathcal{Q}$[7].

One intuitive way you can think is look at how the mean μ shifts from source to target

$$T(x) = x + (\mu_{\mathcal{P}} - \mu_{\mathcal{Q}}) \tag{2.3}$$

However, this simple explanation miss crucial information since it's a coarse summary and becomes uninterpretable in high-dimensional data [15].

An optimal explanation can be found using an Optimal Transport (OT) [24] [25] [26] [27] mapping.

Definition 5 (OT [15]) *OT mapping is a method of optimally moving points from $\mathcal{P}$ to align with $\mathcal{Q}$:*

$$T_{OT} := \arg\min_{T} \mathbb{E}_{x\sim\mathcal{P}}\big[\, c(x, T(x))\big] \quad s.t. \quad T_\sharp\, \mathcal{P} = \mathcal{Q}. \tag{2.4}$$

where c is the cost function we minimize and $T_\sharp\mathcal{P} = \mathcal{Q}$ is called marginal alignment constraint, with $T_\sharp$ called pushforward operator, representing T being applied to all the points in the source distribution.

[7] clearly, to explain distribution shift, we must assume that there exists a relationship between the source $\mathcal{P}$ and the target $\mathcal{Q}$ distribution. Meaning $\exists T(x)$.

Problem is, T_{OT} can be arbitrarily complex and possibly uninterpretable. The interpretability of T_{OT} can be improved by restricting the candidate T to belong to a set of user-defined interpretable mapping Ω [15]. However, this problem can be infeasible if Ω does not contain a mapping that exactly satisfies the marginal alignment contraint $T_\sharp \mathcal{P} = \mathcal{Q}$. To solve this, we can use a Langrangian relaxation to relax the marginal constraint, giving us an *Interpretable Transport Mapping*:

Definition 6 (Interpretable Transport Mapping [15])

$$T_{IT} := \arg\min_{T \in \Omega} \mathbb{E}_{\mathcal{P}}\big[\, c(x, T(x))\big] + \lambda \phi(P_{T(x)}, \mathcal{Q}) \tag{2.5}$$

Where ϕ is a distribution divergence function (e.g. KL divergence).

It is likely that the set of admissible mapping Ω will be initialized based on context. However, there exists some general rules you can follow [15]:

$$\Omega_{vector,1} = \{T : T(x) = x + \delta\} \tag{2.6}$$

or even more generalized:

$$\Omega_{vector,2} = \{T : T(x) = x + \delta(x)\} \tag{2.7}$$

$\Omega_{vector,2}$ includes all the transport map T of the form $T : \mathbb{R}^D \to \mathbb{R}^D$, where D is the dimension of the source distribution $\mathcal{P}$[8]. However, even these simple Ω_{vector} can lead to uninterpretable shifts when the dimensionality is too high (e.g. for a 100 features dataset, δ has 100 components). For this reason, we can restrict the shift to only k dimensions, e.g. *k-Sparse Transport* [15]:

$$\Omega^{(k)}_{sparse} = \{T \in \Omega : |\mathcal{A}(T)| \leq k\} \tag{2.8}$$

Where $\mathcal{A}(T)$ is the active set along which T moves is points. k is a parameter that determines the interpretability. Indeed, the higher is k the less interpretable is T[9] [15].

2.2.5 Experimental Tools for Wireless Network

The design, implementation, and evaluation of our framework required a set of experimental tools that enable both emulated and real-world wireless network deployments. These tools provide the necessary infrastructure for building, controlling, and monitoring cellular networks in flexible and reproducible ways. In particular, we rely on software-based platforms such as (i) *OpenAirInterface (OAI)* [28] [29] and *FlexRIC* [30], which allow experimentation with 5G protocol stacks and Near-RT RIC integration; (ii) large-scale wireless emulation testbeds such as *Colosseum* [31] [32] [33], which offer controlled, reproducible radio environments for training and testing ML-based solutions; (iii) OTA setups with real hardware [34], which allow

[8] Note that the *de facto* standard mean explanation $T : T(x) = x + (\mu_{\mathcal{P}} - \mu_{\mathcal{Q}})$ falls into case (2.6)

[9] Note that [15] does not calculate interpretability for T, just define an ordering on interpretability for T based on k

validation under practical conditions; and (iv) modern Development and Operations (DevOps) toolchains (*Docker*, *OpenShift/Kubernetes*) that ensure modularity, automation, and portability of deployments.

The following subsections provide a detailed description of these tools and their role in our experimental pipeline.

OAI and FlexRIC

OAI is a platform used by researchers to experiment with cellular networks [28] [29]. The OAI codebase is split into three top-level directories that mirror LTE/5G layering. *openair1* implements Layer 1 (L1); *openair2* hosts Layer 2 (L2)/Layer 3 (L3), along with UE/gNB control logic such as timers and state machines, the E2 interface for RIC integration, and support for NG - Application Protocol (NGAP) and X2 Application Protocol (X2AP). *openair3* covers CN interfaces and NAS functionality, including UE Subscriber Identity Module (SIM) authentication and mobility handling.

FlexRIC is instead a flexible and efficient Software Development Kit (SDK) for next-generation Software-Defined Radio Access Network (SD-RAN) [30], designed to enable the creation of specialized, service-oriented controllers. Basically, it provides a flexible implementation of the O-RAN Near-RT RIC architecture by exposing the E2 interface to external xApps and iApps. The FlexRIC SDK consists of an agent library and a server library, along with extensions like internal applications (iApps) and communication interfaces. Although the core is written in C, xApp developers can write their logics in several languages, like Python, thanks to Simplified Wrapper and Interface Generator (SWIG) (see Figure 2.11). With SWIG, the developer specifies an interface file that makes SWIG generates a wrapper that marshals data between Python and C++ and a Python module that imports the compiled extension. When the Python xApp calls the SDK, the call is passed through the wrapper down to the FlexRIC C core. In practice, extending a FlexRIC function in C with SWIG requires only minor edits to the .i file and wrapper code to connect Python xApps with existing C functions.

Both OAI and FlexRIC are open-source projects. *In the absence of comprehensive documentation*, we carefully analyzed their source code and reconstructed the initialization and execution sequences, which are described in the following.

The initialization sequence on the xApp side begins with the setup of the E42 protocol and the xApp Application Programming Interface (API), followed by the creation of the E2AP endpoint (See Figure 2.10). This step establishes the Stream Control Transmission Protocol (SCTP) client, responsible for carrying Control Plane (CP) traffic such as E2AP Setup Requests, Responses, and Subscription messages. The protocol stack is then bootstrapped with the ASN.1 encoder/decoder, while Asynchronous Input/Output (ASIO) initializes asynchronous communication. A message handler is created to manage dispatching of incoming E2 messages, and plugins are loaded to register the E2 Agent and RIC-side service models, enabling encoding and decoding of SM-specific payloads. Once the static initialization is completed, the xApp issues an E2 Setup Request and enters its main event loop, which handles the subscription lifecycle and communication with the RIC (See Figure 2.13). The whole process is visually described in Figure 2.12.

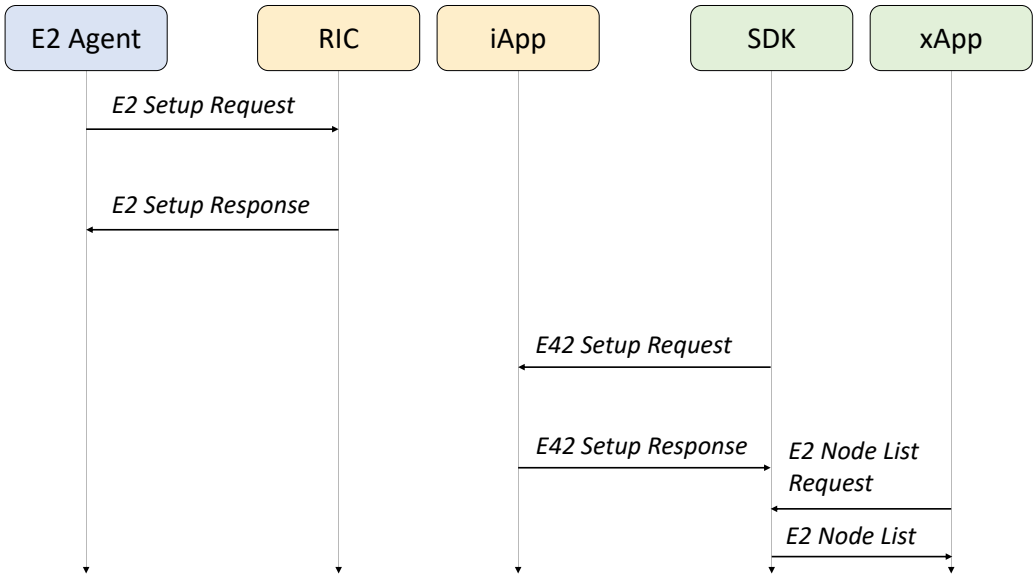


Figure 2.10. The shown communication between iApp, SDK and xApp (right) is performed when launching the xApp, while the one between the E2 Agent and the RIC (left) is performed when laughin the gNB (nr-softmodem).

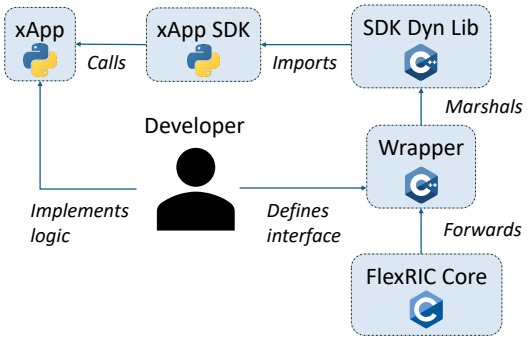


Figure 2.11. SWIG in FlexRIC provides Python bindings for the C/C++ xApp SDK.

On the gNB side, the FlexRIC agent integrated in the *NR-Softmodem* parses the E2 configuration, identifies the RAN node type, and initializes the available RAN function agents. The E2AP agent is then started, mirroring the same initialization steps of the xApp, including the SCTP client and the service model stack. After completing setup, the gNB issues an E2 Setup Request to the RIC and enters its main event loop, maintaining continuous communication with the Near-RT RIC. The message exchange across the E2 interface follows a strict sequence, beginning with Setup Requests and Responses exchanged between the E2 Agent and the RIC, and extending to service-model-specific procedures such as subscription, report, and control messages. FlexRIC decouples the message encoding, decoding, and handling logic, ensuring that each E2AP message is processed in the RIC main event loop and, when necessary, passed back to the RAN nodes through the E2 interface. This modular architecture allows the integration of both standard O-RAN service models

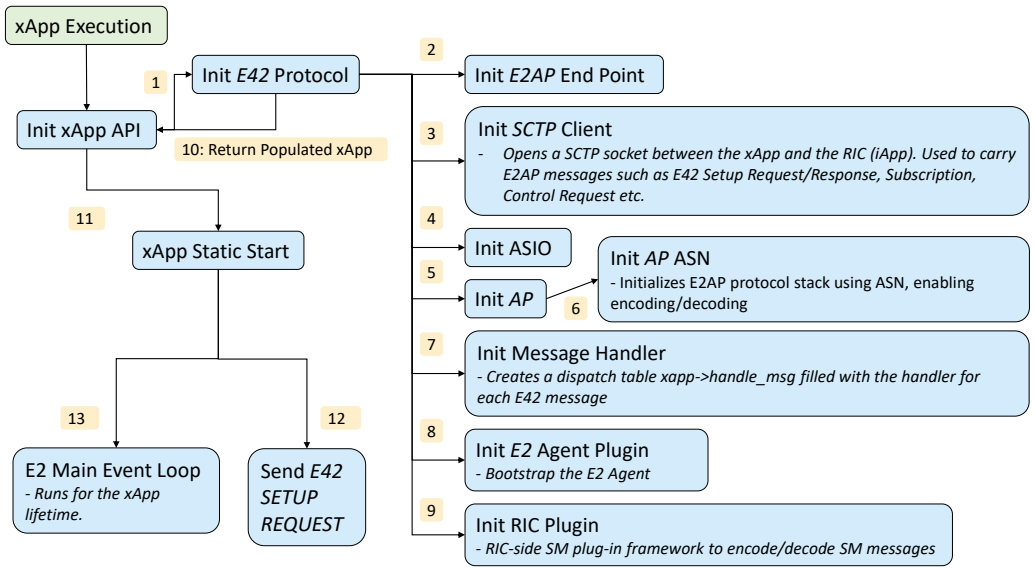


Figure 2.12. Initialization sequence of a FlexRIC xApp. The process begins with the setup of the E42 protocol and xApp API, followed by the creation of the E2AP endpoint. Subsequent steps include establishing the SCTP client, bootstrapping the ASN.1 encoder/decoder, initializing asynchronous communication, and registering message handlers and plugins. Once initialization completes, the xApp issues an E2 Setup Request and enters its main event loop.

and customized ones, thereby supporting flexible and extensible RAN control within O-RAN compliant deployments. The whole process is visually described in Figure 2.14.

Colosseum

Another key tool in our experiments is *Colosseum* [32] [33] [31], a large-scale, open-access wireless testbed that enables experimental research on virtualized and softwarized waveforms and protocol stacks over a fully programmable platform. Equipped with 256 Software Defined Radio (SDR) and a Massive Channel Emulator (MCHEM), Colosseum can accurately model virtually any wireless scenario, supporting the design, development, and testing of solutions at scale across diverse deployments and channel conditions. RF scenarios are reproduced through Field Programmable Gate Array (FPGA)-based Finite Impulse Response (FIR) filters, which implement the taps of target wireless channels and apply them to the signals generated by the radios, thus closely mimicking real-world propagation conditions. A schematic overview of the Colosseum architecture is shown in Figure 2.15 while a view on the physical infrastructure of the emulator is in Figure 2.16.

Experimentation on Colosseum can be conducted in two modes: *interactive*, where the user directly configures and controls the testbed, or *batch*, where predefined scripts automate the execution of data collection campaigns. In this work, *all experiments were conducted in interactive mode.*

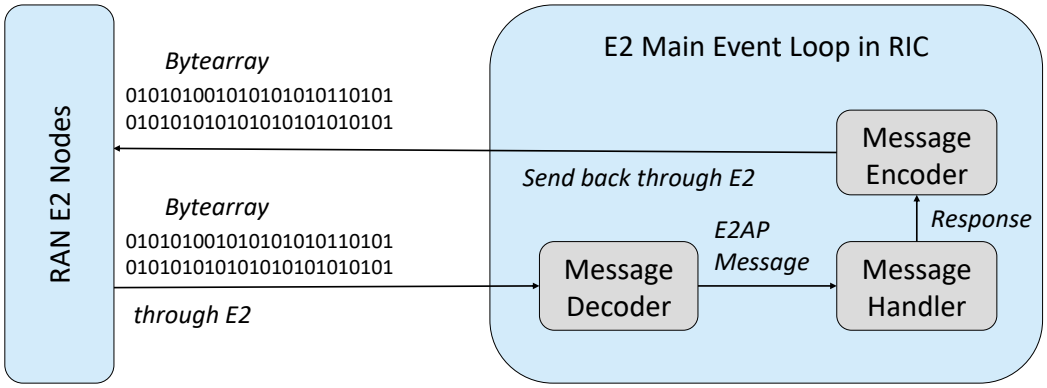


Figure 2.13. E2 main event loop inside the RIC. Incoming bytearrays from RAN E2 nodes are decoded into E2AP messages, which are then processed by the message handler. Responses are re-encoded and sent back through the E2 interface, enabling continuous two-way communication between the RIC and E2 nodes.

OTA setup

We also used an OTA setup: A 64-antenna SDR-based ceiling Grid Testbed. It is open-access, indoor, located at *Northeastern University*. The research platform features several computational servers, radio racks with fully-synchronized SDR and a 64-antenna grid mounted on the office ceiling [34] (See Figure 2.17). Details on the setup will be provided in Chapter 4.

DevOps stack and deployment WF

We rely on a container-native toolchain to make our experiments reproducible and portable across environments. xApps and supporting services are containerized with *Docker*, while cluster-level orchestration for the OTA setup is handled by *Openshift* (*Kubernetes*).

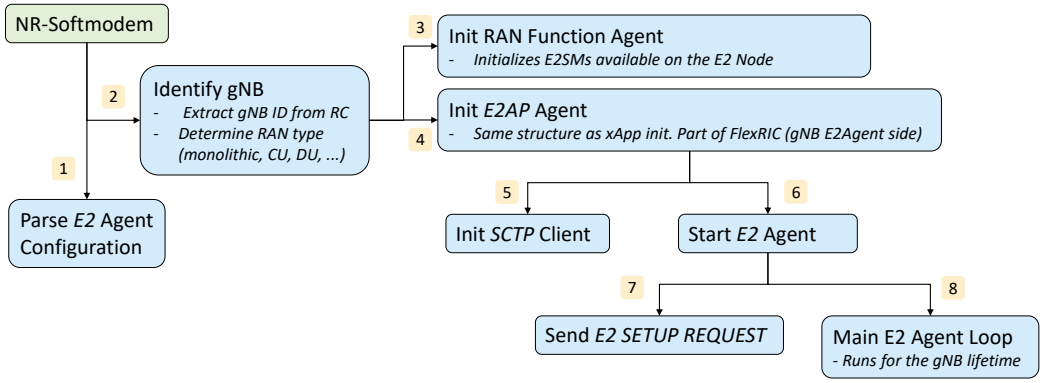
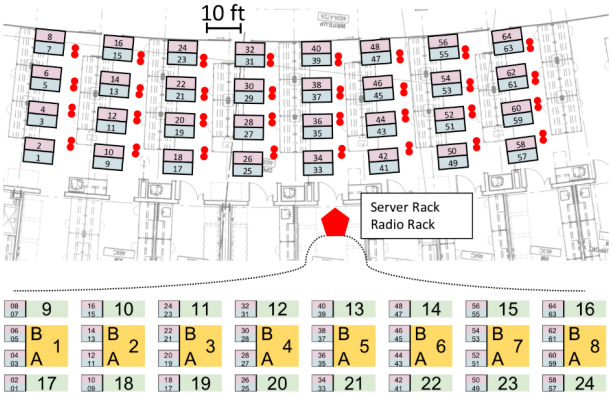


Figure 2.14. Initialization sequence of the FlexRIC E2 Agent on the gNB side. After parsing the configuration and identifying the RAN node type, the gNB initializes available RAN function agents and the E2AP agent, sets up the SCTP client, and starts the E2 agent. The gNB then issues an E2 Setup Request and enters the E2 Agent main event loop, mirroring the xApp initialization flow.

(a) OTA setup used in our experiments.



(b) Testbed antenna grid layout.

Figure 2.17. OTA experimental setup and testbed entire antenna grid layout

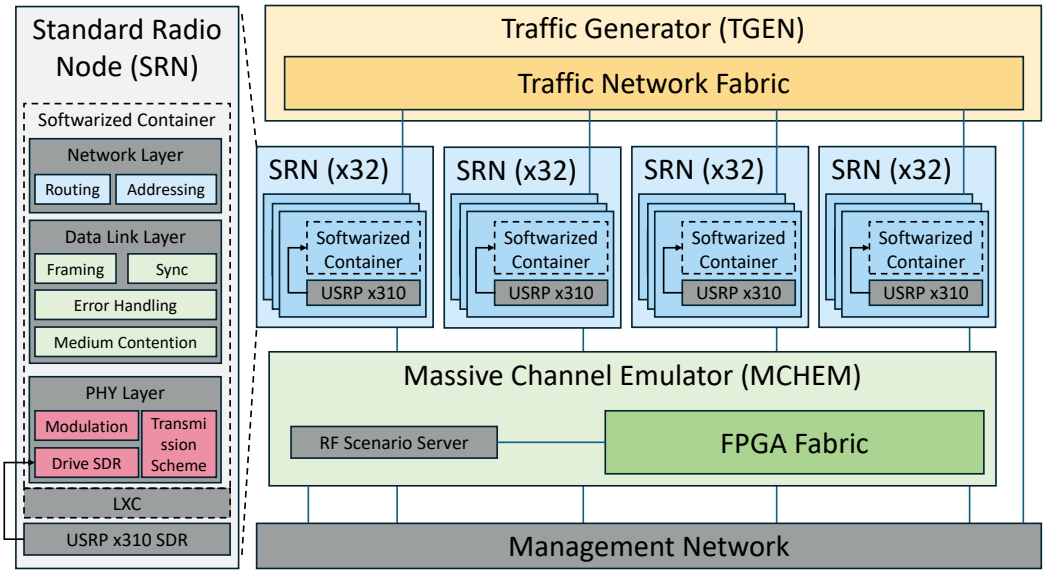


Figure 2.15. Colosseum architecture, redrawn from [32]. The figure highlights the main components of the testbed, including Standard Radio Node (SRN) equipped with Universal Software Radio Peripheral (USRP) X310 SDR, Traffic Generator (TGEN) and MCHEM based on FPGA fabric. For simplicity, details of the management infrastructure (e.g., experiment website, resource management gateways, network services, and NAS storage) are omitted.

2.3 AI in O-RAN

The overarching objective of leveraging ML is to empower predominantly autonomous and intelligent O-RAN capable of self-configuration, self-monitoring, self-healing, and self-optimization with minimal human intervention [35], [36].

2.3.1 AI use cases

In O-RAN, AI and ML serve two purposes: (i) intelligent network management and (ii) progress toward an AI-native "radio of the future". For the purposes of this thesis, *we are interested in the intelligent network management*. However, for the reader's interest, the latter is explored in [37] [38], where an AI-native air interface is envisioned and learning-based components co-design or replace parts of the 5G protocol stack so that communication strategies are discovered and adapted by models rather than fixed by handcrafted protocols.

For network management, data-driven models support radio resource optimization beyond traditional heuristic (e.g. RAN slicing and traffic steering, see Figure 2.18 and Figure 2.19), enable closed-loop automation that adapts to traffic and mobility dynamics with minimal human intervention, and contribute to network energy savings policies [39].

Academic institution and industry partners have joined force in the *AI-RAN Alliance* to advance the convergence of AI and the RAN through three Working Groups (WGs): (i) *AI for RAN*, using data-driven models to boost spectral/energy

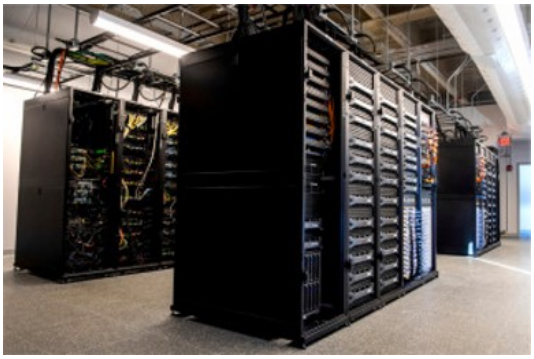


Figure 2.16. The *Colosseum* testbed at *Northeastern University*, a large-scale wireless network emulator composed of hundreds of interconnected servers and SDR, enabling realistic and reproducible experimentation.

efficiency, reliability, and automation; (ii) AI and RAN, co-locating AI workloads (e.g., inference) on the virtualized, disaggregated RAN edge to share infrastructure and enable new services; (iii) *AI on RAN*, defining radio/interface needs for latency-sensitive AI applications at the edge, benchmarking on 5G, and deriving requirements for 6G.

In this thesis, *We will focus on the (i) AI for RAN set of use cases.* Precisely, in our experiments, we deal with *traffic forecasting* and *traffic classification.*

Traffic forecasting

Accurate TS predictions allow policies for traffic engineering, resource allocation, and service orchestration to be enacted before demand fluctuations, improving utilization and QoS [40] and enabling Quality of Experience (QOE)-related[10] use cases. Traffic prediction has a long history, from classical statistical methods to modern data-driven models, and is now a cornerstone of state-of-the-art network management. *For the purposes of this thesis, we are interested in using it for slice-based PRB allocation*, to ensure stability, fairness, and QoS compliance while remaining responsive to future demands. Oscillations between slice priorities must be avoided, and the allocation should anticipate upcoming loads rather than merely follow historical usage. Several predictive metrics can be combined to achieve this goal. We provide some examples in Table 2.1.

[10]QOE is an application-specific metric. For instance, video streaming can use Mean Opinion Score (MOE) and buffering to calculate it, while Voice over IP (VoIP) can use latency.

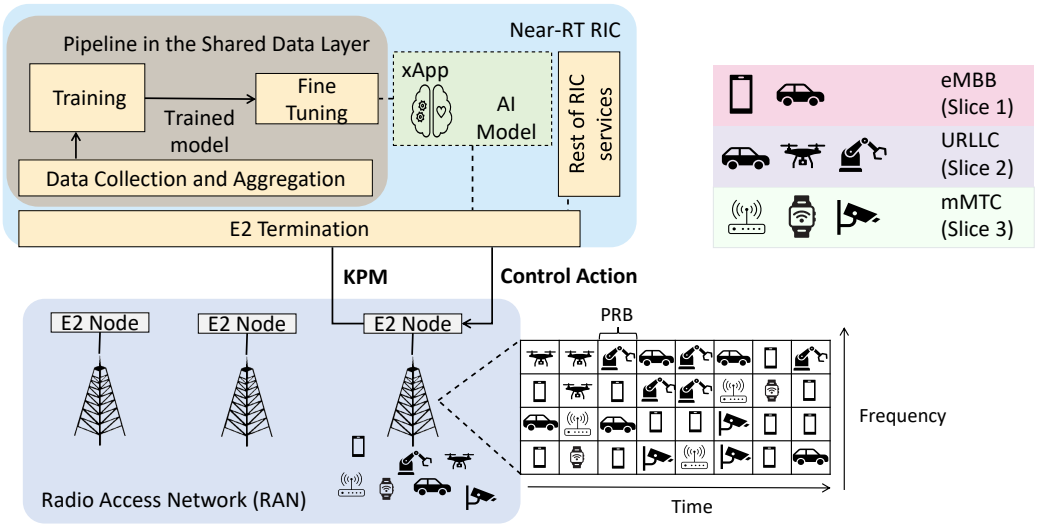


Figure 2.18. Closed-loop, slice-aware control in O-RAN. KPMs collected over the E2 interface from E2 Nodes feed the Near-RT RIC SDL. An ML-based xApp processes these measurements to forecast/classify demand and issues control actions back over E2, adjusting per-slice Physical Resource Block (PRB) allocation in the time–frequency grid of the MAC scheduler for eMBB, URLLC, and mMTC.

Table 2.1. Representative metrics for slice-based PRB allocation. Our heuristic will be instead presented in Chapter 4.

Metric	Formula	Description / Effect
Predicted TH Ratio	$r_s = \frac{\text{pred TH}_s}{\sum_j \text{pred TH}_j}$	Captures relative underutilization across slices but may cause oscillatory behavior if used alone.
Fairness Index	$\phi_s = \frac{\text{pred TH}_s}{\text{allocated PRBs}_s + \epsilon}$	Favors slices with low TH per resource, balancing demand and allocation fairness.
Deficit-Based	$D_s(t+1) = D_s(t) + \text{pred demand}_s - \text{allocated PRBs}_s$	Smooths allocation over time by accumulating deficits, preventing short-term oscillations.
Smoothed Moving Average	$M_s = \alpha \cdot \text{predicted TH}_s + (1-\alpha) \cdot \text{past TH}_s$	Blends recent predictions with historical averages to avoid abrupt allocation changes.
Gradient of pred TH	$\Delta_s = \text{pred}_s(t+1) - \text{pred}_s(t)$	Detects rapidly increasing demand (e.g., URLLC bursts) to allocate additional resources proactively.

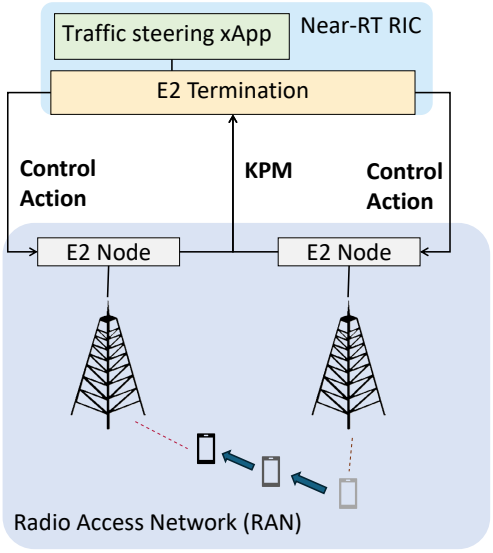


Figure 2.19. Near-RT traffic steering over E2. KPMs reported from multiple E2 Nodes to the Near-RT RIC feed a traffic-steering xApp, which issues control actions back over the E2 interface (e.g., handover/steering commands or scheduling policies) to shift UE between cells and balance load while meeting QoS targets.

These metrics can be integrated into a composite decision rule:

$$\text{score}_s = w_1 \frac{1}{\text{pred TH}_s} + w_2 D_s + w_3 \Delta_s + \dots$$

normalized across slices to produce PRB ratios transmitted to the MAC scheduler. To prevent abrupt reallocations, smoothing techniques such as Exponential Moving Average (EMA) or hysteresis can be applied:

$$\text{new max\_ratio}_s = \beta \cdot \text{prev} + (1 - \beta) \cdot \text{score\_based}.$$

While substantial progress has been made in network slicing, the focus has largely remained on the RAN, even though the main TH and delay bottleneck resides in the core network. Designing O-RAN-compliant traffic classification introduces specific challenges, as the Near-RT RIC lacks access to traditional 5-tuple or full layer-2 information, making existing 5G classifiers incompatible with real O-RAN constraints. Conventional IP traffic classification based on packet inspection fails under encryption, leading to statistical and ML-based approaches that either rely on 5-tuple flow features or full encrypted packet payloads, both of which raise privacy and security concerns. Work such as the Traffic Analysis and Classification Tool for Open RAN (TRACTOR) framework [41] supports *network-initiated slicing*, where slice assignments are determined by the network—typically at the gNB or AMF—instead of through explicit UE requests, thereby improving allocation efficiency and reducing control overhead.

Another challenge of traffic forecaster in O-RAN, is that, the stochastic nature of the wireless channel makes pointwise, next-sample prediction infeasible: individual measurements are effectively random. Instead, forecasting is applied to aggregate KPM over short horizons (on the order of hundreds of milliseconds to a few seconds).

Traffic classification

In O-RAN, traffic classification is the task of mapping short windows of available KPMs to semantic traffic classes (e.g., eMBB/URLLC/mMTC, service/QoS profiles). Unlike traditional heuristic such as DPI or 5-tuple methods, data-driven classifiers in O-RAN must operate without payload inspection, 5-tuple metadata, or full L2 frames, relying instead on standardized, O-RAN-compliant measurements. Accurate, low-latency classification is a key enabler for closed-loop control and network slicing: class posteriors can drive per-slice PRB budgeting, scheduler weights, or QoS policies, and should be robust to domain shift and bottlenecks that may lie beyond the RAN.

2.3.2 AI/ML WF in O-RAN

In the O-RAN architecture, AI/ML follows a closed-loop WF that spans the Non-RT and Near-RT control planes [42]. (i) *Data collection and processing*: KPMs and events are exported from RAN E2 Nodes via the E2 interface and curated in the shared data layer; (ii) *Offline training & evaluation*: models are trained, validated, and versioned. (iii) *Policy/model publication and distribution*: policies, model metadata, and configuration are provided through the interested components; (iv) *Inference & Control*. (v) *Monitoring & Continual Learning*: outcomes are monitored; performance feedback closes the loop, triggering Fine-Tuning (FT) or redeployment. See Figure 2.20 for a visual overview of this WF.

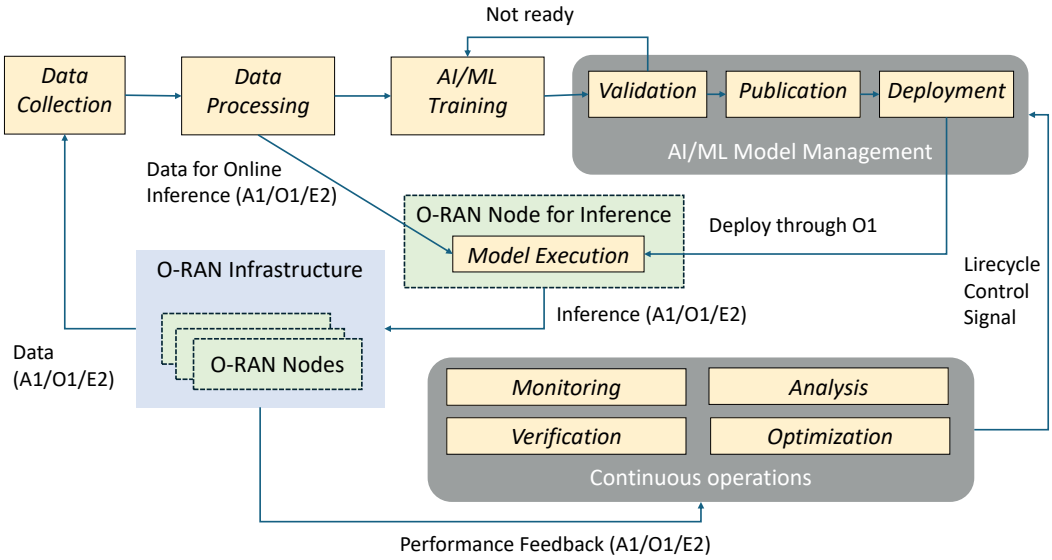


Figure 2.20. AI/ML WF. Redrawn from [5].

A visual overview of all the possible scenarios detailing which components take the role of which phase is in Figure 2.21. In the following paragraph, we will detail each stage of this ML lifecycle pipeline.

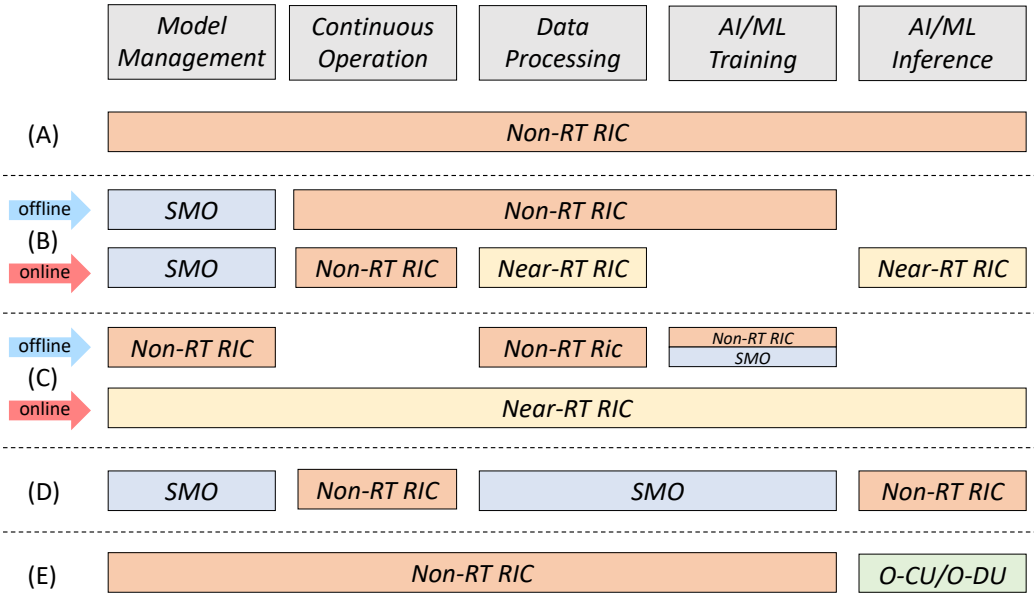


Figure 2.21. AI/ML lifecycle placement options in O-RAN. Functional blocks (model management, continuous operation, data processing, training, inference) can be hosted across SMO, Non-RT RIC, Near-RT RIC, and even O-CU and O-DU. (A) All stages centralized in the Non-RT RIC. (B) Split pipeline: offline stages in Non-RT; online inference in Near-RT. (C) Near-RT-centric pipeline with minimal Non-RT roles. (D) Non-RT responsibility pipeline. (E) Inference pushed down to the O-CU O-DU.

Data collection and processing

The SMO aggregates multi-source data (KPM, configs, events) from O-RU/O-DU/O-CU, UE, CN, and external feeds via O1, E2, and A1. Data is cleansed, aligned, normalized, and feature-engineered; privacy safeguards (e.g., anonymization) are applied. Curated datasets are stored in the SDL in the controllers for analytics and model training.

Training and validation

SMO hosts offline training on historical RAN data. Offline training and validation is mandatory before exposure to live traffic to reduce instability and risk [43]. Online FT is allowed [43] [44]. Operators[11] may collaborate and interact in this multi-stage pipeline [10].

Publication and Deployment

Trained models are packaged (often containerized) and onboarded as rApps/xApps. O-RAN supports model distribution through its A1 interface, with the Non-RT RIC providing ML model management information to the Near-RT RIC. In practice, this

[11] For instance, an operator may import models and data from an external component into its system, comprised of SMO + Network Function Virtualization - Management and Orchestration (NFV-MANO) + 3GPP - Network Slicing Management System (3GPP-NSMN)

could mean sending the model parameters or a reference to a model file from an rApp (or a centralized model registry in the SMO) to an xApp.

Continuous Monitoring, Validation and Verification

Live Key Performance Indicator (KPI)s and outcomes are tracked to trigger FT, fallback to deterministic heuristics if behavior degrades[12], escalate/rollback per operator policy. This closes the loop and maintains safety, robustness, and performance over time.

Inference

After deployment, data-driven model starts working at inference time. These data-driven models can also be chained and work cooperatively in x/rApps. An example of cooperation is provided in Figure 2.22.

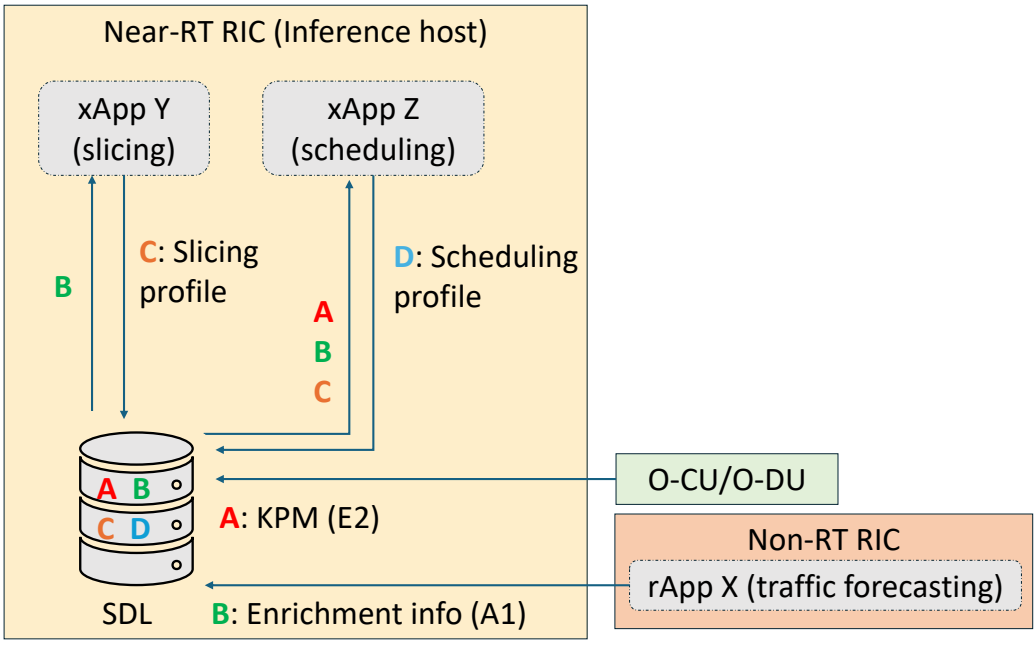


Figure 2.22. An example of chained data driven models with diverse input types and data producers. Note that in this representation we are assuming that the SDL also comprises a layer for data access, handling concurrency.

2.3.3 Learning under Distribution Shifts

As we already said, in O-RAN deployments distribution shifts are ubiquitous and often unknown. Traditional domain adaptation techniques [45, 46, 47] require access to target domain data, which is rarely feasible in practice. Therefore, research has focused on enhancing the OOD generalization ability of models when

[12] that opens the door for potential "downgrade attack", with the attacker intentionally deceive the system into thinking that data-driven components are worsing.

test data distributions are unavailable [48]. These approaches include domain generalization, Distributionally Robust Optimization (DRO), invariant learning, and stability-based methods. However, none of the existing algorithms have achieved consistent or universal improvement across all OOD settings [48]. This difficulty arises from the diversity of possible shifts [49], which prevents a one-size-fits-all solution.

Definition 7 (OOD generalization problem) *Given training data from a source distribution $\mathcal{P}(X,Y)$ and test data from a target distribution $\mathcal{Q}(X,Y)$, the goal of OOD generalization is to find a model f_θ^* that minimizes the expected loss on $\mathcal{Q}$:*

$$f_\theta^* = \arg\min_{f_\theta} \ \mathbb{E}_{X,Y\sim\mathcal{Q}}[\ell(f_\theta(X),Y)].$$

When $\mathcal{P}(X,Y) = \mathcal{Q}(X,Y)$, the IID assumption holds and the problem reduces to standard Empirical Risk Minimization (ERM) [50]. Conversely, when $\mathcal{P}(X,Y) \neq \mathcal{Q}(X,Y)$, a distribution shift occurs, and ERM typically fails to generalize [51, 52].

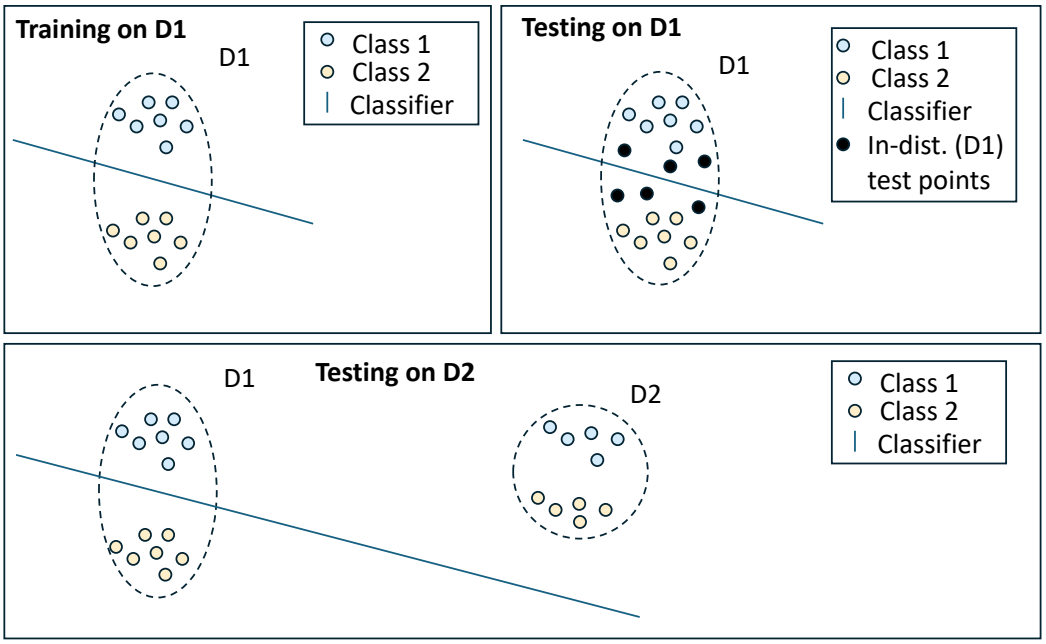


Figure 2.23. Illustration of a distribution shift between training and testing domains.

OOD Evaluation OOD evaluation [48] aims to assess a model's robustness under unknown or unseen shifts. Unlike In-Distribution (ID) evaluation, which assumes the same data distribution between train and test sets, OOD evaluation investigates how performance changes under intentionally induced or naturally occurring shifts. It helps identify *where* a model performs reliably and *where* it fails, rather than focusing on building a single universally robust model.

OOD evaluation methods can be categorized as follows (see Figure 2.25):

1. **OOD Performance Testing:** evaluates models on labeled OOD test data. The challenge is generating or partitioning datasets reflecting realistic shifts.

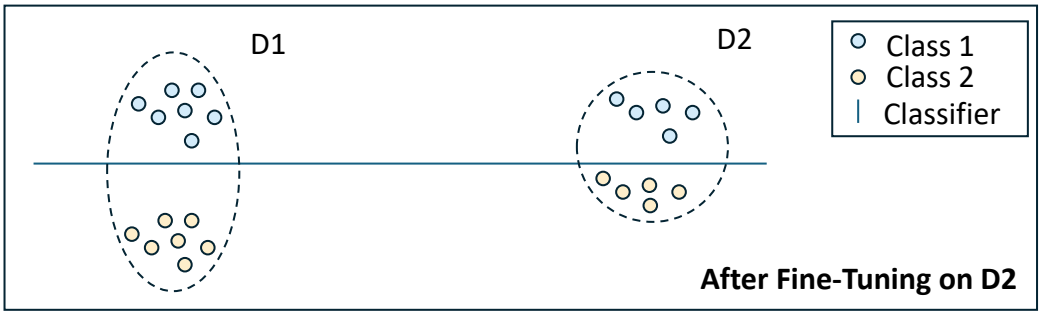


Figure 2.24. Illustration of how a classifier decision space changes when fine-tuned.

2. **OOD Performance Prediction:** estimates performance on unlabeled test data, crucial for O-RAN, where ground truth may be unavailable.
3. **OOD Intrinsic Property Characterization:** analyzes the inherent robustness properties of models or training data when no test data exist.

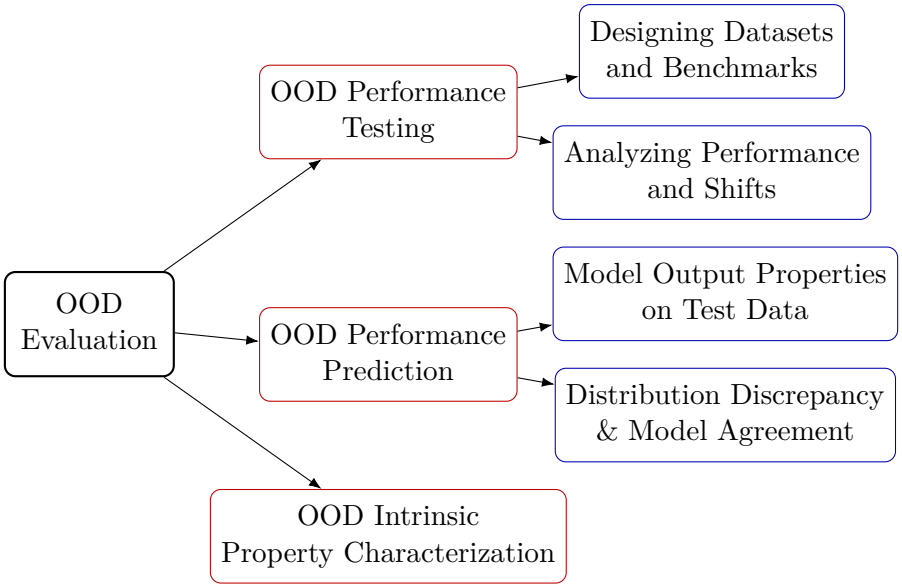


Figure 2.25. Taxonomy of OOD evaluation. Categories are in red, sub-categories in blue.

When it comes to Performance Testing, precisely to measure performance properly with good dataset design and benchmarking, it must be noted that most OOD benchmarks rely on visual or text datasets, *while tabular and temporal data, relevant in O-RAN, remain underexplored.* Current tabular datasets often fail to capture specific shift patterns, such as $P(Y|X)$ (concept shift) or $P(X)$ (covariate shift) [48]. Empirically, [53] find concept shifts dominate in tabular data, whereas covariate shifts prevail in image and text domains. Causal modeling can help design synthetic datasets that simulate plausible shift patterns [54, 55, 56, 49].

When instead it comes to intrinsic property characterization of OOD generalization, we consider three properties, *robustness*, *stability* and *flatness*.

DRO [57] enhances robustness by minimizing the worst-case expected loss over an uncertainty set $P(\mathcal{P})$ surrounding the training distribution:

$$\mathcal{R}(\theta) = \sup_{\mathcal{C} \in P(\mathcal{P})} \mathbb{E}_{X,Y \sim \mathcal{C}}[\ell(f_\theta(X), Y)], \tag{2.9}$$

where $P(\mathcal{P}) = \{\mathcal{C} : \text{Dist}(\mathcal{C}, \mathcal{P}) \leq \rho\}$ and Dist may be an f-divergence, Wasserstein, or MMD distance. If the test distribution $\mathcal{Q}$ lies within $P(\mathcal{P})$, the model generalizes safely.

To evaluate robustness under covariate shifts, [58] propose:

$$W_\alpha := \sup_{C_Z \in C_\alpha} \mathbb{E}_{Z \sim C_Z} \left[\mathbb{E}[\ell(f_\theta(X), Y) \mid Z] \right], \tag{2.10}$$

where C_α includes subpopulations larger than α. For instance, in O-RAN, these could represent urban, rural, or mobile users; W_α measures the worst-case loss among subgroups of size at least α. To explain briefly what a subpopulation is: your train set is sampled from $\mathcal{P}(X, Y)$. These data points may naturally split into groups, namely subpopoulations, according to some characteristics (Z). Each group z is a subpopulation with its own distribution $\mathcal{P}(X, Y \mid Z = z)$. To give an O-RAN example: imagine you train a model to predict video quality in a mobile network. Your data has users under different conditions:

- Subpop A: users in urban areas
- Subpopulation B: users in rural areas
- Subpopulation C: users moving in cars
- Subpopulation D: users indoors with WiFi

DRO asks: what if the test distribution puts more weight on a difficult subpopulation (e.g., rural or moving users)? Will the model still perform well? In W_α we evaluate the worst-case expected loss across subpopulations of size at least α.

Stability quantifies the sensitivity of models to small perturbations in the data distribution. [59] define the minimal perturbation required to change parameter sign:

$$s(\theta, \mathcal{P}) := \sup_P \exp\{-D_{KL}(P \| \mathcal{P})\} \quad \text{s.t. } \theta(P) = 0, \tag{2.11}$$

while [60] measure the minimal distributional change needed to degrade performance beyond a threshold t:

$$I_t(\mathcal{P}) := \inf_P \{D_{KL}(P \| \mathcal{P}) : \mathbb{E}_P[\ell] \geq t\}. \tag{2.12}$$

Unlike DRO, which requires specifying an uncertainty radius ρ, stability metrics offer more intuitive, domain-informed thresholds.

A flatter loss landscape typically implies better ID generalization and potentially OOD robustness. The most used metric is the *Sharpness-Aware Minimization* (SAM) flatness measure:

$$R_\rho^{SAM}(\theta) := \max_{\theta' \in B(\theta, \rho)} (\hat{L}(\theta') - \hat{L}(\theta)), \tag{2.13}$$

where ρ defines a neighborhood around θ and $\hat{L}$ is the empirical loss. Although widely adopted, the exact relationship between flatness and OOD performance remains an open question [61].

Distinguishing between improvements in OOD and ID generalization remains a key challenge. Increasing model capacity or data quantity may boost both, without truly improving robustness to shifts. Furthermore, reducing the performance gap between ID and OOD through regularization (e.g., high weight decay) may not necessarily enhance generalization.

2.4 AML in O-RAN

AML is the study of ML in adversarial environments [62]. it comprises the design of ML algorithms that can resist attacks, coupled with the systematic study of the capabilities and limitations of attackers, often conceptualizing this conflict as a game between an attacker and a defender where the attacker manipulates data to mis-train or evade the learning algorithm. AML targeting wireless [63], cellular [64] and O-RAN assets [65] [66] [67] [68] [69] have emerged as a critical threat. This is possible because a lot of assets have vulnerabilities [70] [71] [72]. In the following, we focus on BA and the current *cat-and-mouse game* between attackers (Subsection 2.4.1) and defenders (Subsection 2.4.2) in AML.

2.4.1 From the Attacker's Perspective

Referring to the use cases in 2.3.1, we could think an AML attack could cause the following scenarios: (i) Unfair Resource Allocation (Figure 2.26): a poisoned resource-control xApp over-allocates PRB/TBS to the attacker-targeted slice and throttles the others, causing systematic starvation and SLA violations even when KPI would not justify it; (ii) Misled Traffic Steering (2.27): a steering/load-balancing xApp biases handover decisions to push UE toward attacker-chosen or sub-optimal cells, increasing congestion, ping-pong events, and call drops despite seemingly normal measurements; (iii) Hijacking Energy-Saving Decisions (2.28): a poisoned energy-saving xApp inappropriately activates cell/sector sleep (or prevents wake-up), or disables beams, at busy times, creating coverage holes and service denial for groups of UE, or conversely keeps cells awake to inflate energy consumption. The open and modular nature of O-RAN potentially increases the risk of AML respect to general cellular networks, because the supply chain is broader. Unlike a traditional RAN where one vendor's closed system does everything (limiting the number of entities with the ability to tamper with ML models), O-RAN ecosystem might involve multiple software contributors. Each xApp or rApp could come from a different source, and some may even use pre-trained models from open-source communities [73][74][75] [76] (hiding backdoors, see 2.4.1) [73][74][75]. A ZT to the ML supply chain means operators and system integrators must assume that any pre-trained model, third-party algorithm or data could be malicious unless proven otherwise.

A ML model like one of these can be targeted in different ways. A useful distinction is between a *strong attacker*, corresponding to a WB setting, and a *weak attacker*, corresponding to a BB setting [77] [78] [79]. In the WB case, the adversary has full access to the model and enhanced tampering capabilities, for example, when

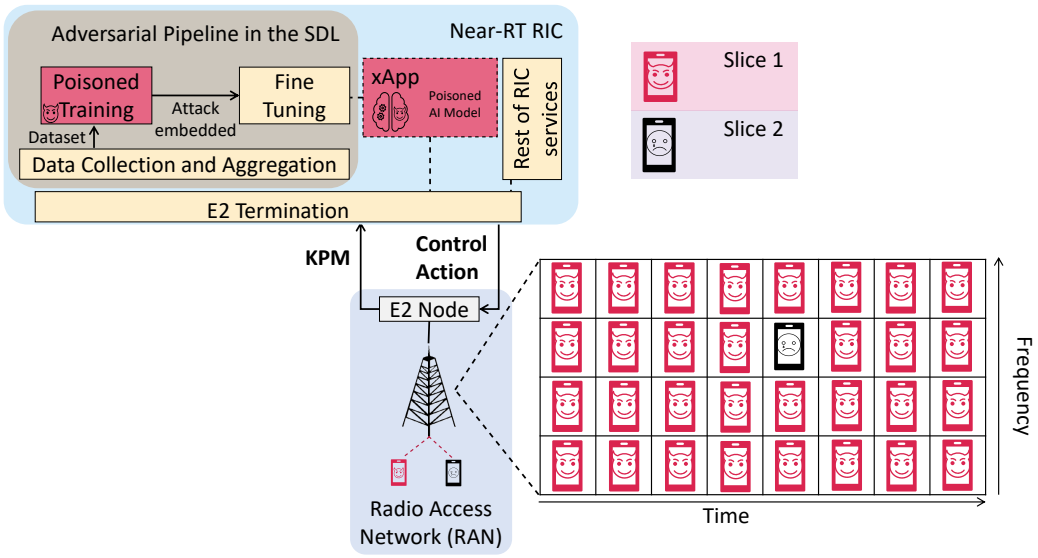


Figure 2.26. Unfair Resource Allocation. Triggered backdoored xApp biases E2 control to over-allocate resources to the target slice over time.

training is outsourced to a malicious third-party provider. In contrast, in the BB case, the adversary cannot access the model directly but can observe and manipulate relevant data and KPI[13].

Another distinction is between DP, BA (also called neural trojans), AA/EA at inference time and more privacy-threatening attacks. A compact taxonomy of these vulnerabilities is shown in Figure 2.29 and surveyed in the literature [80, 81, 82]. Recent work has identified O-RAN-relevant vectors spanning EA [83, 68, 84], DP [70], and privacy-oriented attacks such as model extraction and inference [65, 83]; several surveys synthesize these findings [69]. Each attack class targets different phases of the ML lifecycle (training vs. inference) and different properties (integrity vs. confidentiality), and they require different levels of access to O-RAN components.

For this work we focus on strong, poisoning-based BA (BA). We adopt this focus because strong attacks are the most plausible threat given the expected use of third-party marketplace models in O-RAN, though we note that backdoors can in principle also be introduced during BB FT deployment [85], that is allowed online [86]. The main reasons for concentrating on BA are:

- **BA is more realistic than EA/AA in O-RAN** Adversarial/EA craft input-specific perturbations at inference time and typically do not transfer reliably between samples. By contrast, a backdoor is a sample-agnostic trigger that, once learned, consistently steers arbitrary inputs to the attacker's chosen label [80]. Adversarial attacks require costly, inference-time optimization (often assuming gradient or query access), which is impractical in many near-RT deployments. Backdoors require training-time access but impose no extra inference cost, making them more feasible in O-RAN settings.

[13]For the purposes of this work, the terms KPI and KPM are used interchangeably.

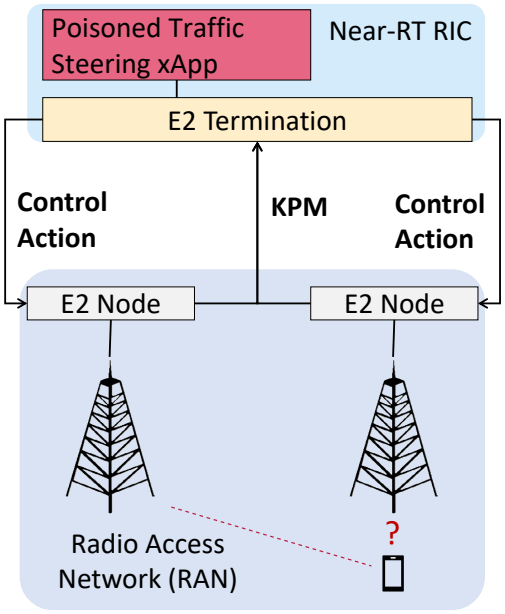


Figure 2.27. Misled Traffic Steering. Triggered steering xApp pushes UE toward attacker-chosen sub-optimal cells.

- **BA is stronger and stealthier than DP** Traditional (*classical*) data poisoning degrades generalization and is readily detected by local validation, whereas BA preserve benign accuracy and are therefore stealthier. Advanced poisoning attacks can be targeted and superficially resemble backdoors, but they typically do not rely on explicit, reusable triggers: advanced DP forces misclassification of a small set of attacker-chosen inputs, while backdoors create a trigger–label association applicable across many inputs.
- **BA is underinvestigated in O-RAN**: so we have only the theoretical studies, not a single experimental study to validate BA in real O-RAN environments. However, is a servious threat, just think about TS and untrusted third-party data-driven xApps deployment.

Two further caveats are worth noting. Defenses are not strictly orthogonal: adversarial training, effective against many EA, may increase susceptibility to backdoors [87]; conversely, techniques designed for poisoning mitigation (e.g., gradient shaping) can provide partial resilience to some backdoor strategies [88].

In the following, we will focus strictly on BA, their mathematical framing, what we can say about their explanations.

Definition and Details of BA

One of the problems with the BB nature of Deep Neural Network (DNN) is that their behavior cannot be exhaustively tested. A DNN is often described as a BB because the learned model is a complex composition of weights and nonlinear functions that rarely maps onto interpretable, human-readable features of the classification it performs. For instance, a facial-recognition system can be validated on a test set of

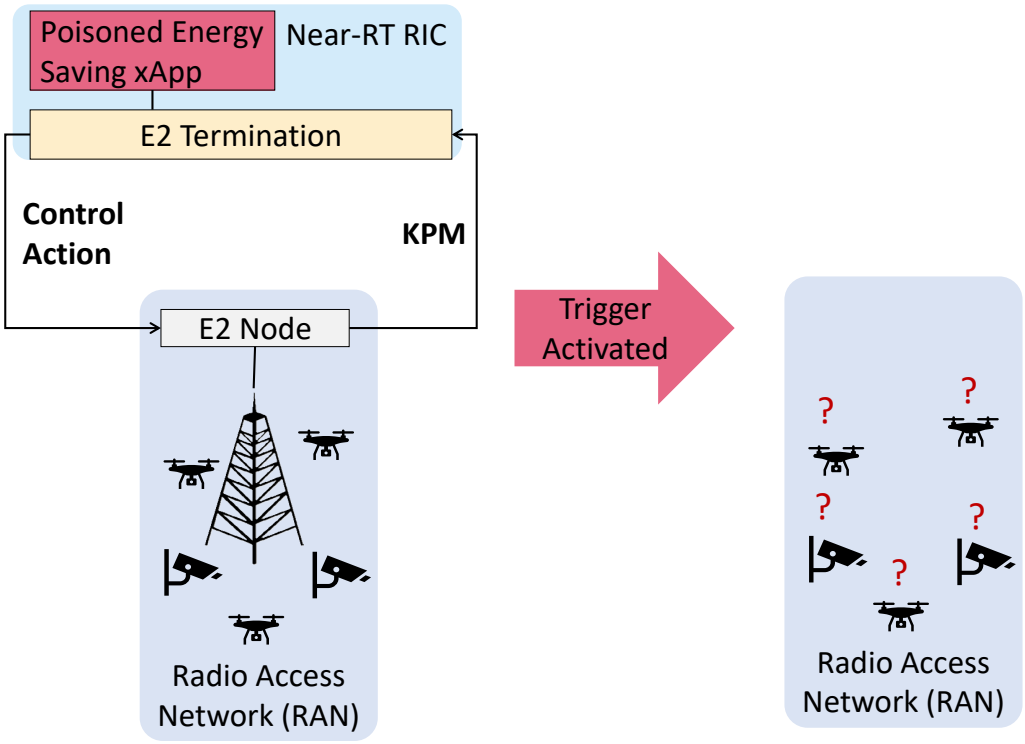


Figure 2.28. Hijacked Energy-Saving Decisions. Triggered energy-saving xApp inappropriately sleeps cells/beams, creating coverage holes.

known faces, but this gives no guarantees about how the model behaves on untested images or previously unseen identities. That lack of transparency opens the door to hidden manipulations known as backdoors or "Trojans." backdoors exploit the excessive memorization capacity of DNNs to establish a latent link between a trigger and a target label. They modify the decision function so that regions around triggered samples become unusually smooth, a property observed and quantified via entropy-style local-uncertainty metrics [89, 90]. This smoothing explains why trigger-inversion defense methods can sometimes recover plausible triggers even for non-infected models (see 2.4.2). Conceptually, a backdoor inserts an extra latent "trigger" dimension in representation space that remains dormant on clean inputs but, when activated by the trigger, overrides task-relevant features and steers predictions toward the attacker's class; small manipulations along this dimension suffice to flip outputs (see Fig. 2.32). The precise internal mechanisms that allow hidden backdoors to form and persist remain not fully understood.

Definition 8 (Backdoor [91]) *We define a DNN backdoor to be a hidden pattern trained into a DNN, which produces unexpected behavior if and only if a specific trigger is added to an input.*

Outside of that narrow trigger condition the model appears to perform normally. In this work we focus on DP-based BA [80], while noting that weight and structural/architectural backdoors exists, but is outside the scope of this thesis (see [92]). A visual illustration of backdoor injection is shown in Figure 2.30.

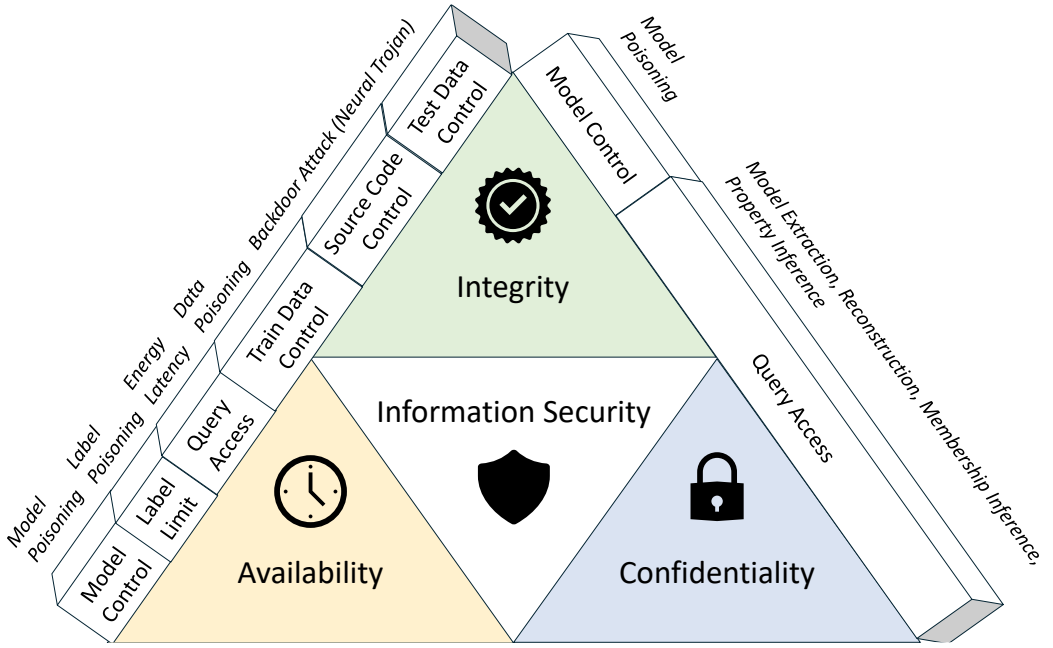


Figure 2.29. Taxonomy of AML threat contestualized in the Confidentiality, Integrity, Availability (CIA) triad

Since the attacker has full access to the training procedure, she can change the training configurations, e.g., learning rate, ratio of modified images, to get the backdoored DNN to perform well on both clean and adversarial inputs. Moreover, we can also define mathematically the training procedure that inject a backdoor in a DNN.

Definition 9 (Backdoor Learning [80]) *Let $f_\theta : X \to [0,1]^K$ be a clean classifier, and C be the infected model. Let $G_t : X \to X$ indicates the attacker-specified poisoned samples generator with trigger t, with $S : Y \to Y$ the attacker-specified label shifting function. Given a benign supervised dataset D, D_t as a training set and $D_s \subset D_t$ poisoned subset of the training set*[14]*, trigger $t \in \mathcal{T}$, λ_1 and λ_2 being trade-off hyperparameters, we can define Backdoor Learning as:*

$$\min_{t,w} R_s(D_t - D_s) + \lambda_1 \cdot R_b(D_s) + \lambda_2 \cdot R_p(D_s) \tag{2.14}$$

With:

$$R_s(D) = \mathbb{E}_{(x,y)\sim P_D} \mathbb{I}\{C(x) \neq y\}$$

Standard risk measuring whether the infected model C can predict benign samples;

$$R_b(D) = \mathbb{E}_{(x,y)\sim P_D} \mathbb{I}\{C(x') \neq S(y)\}$$

Backdoor risk indicating whether BAer can achieve their malicious[15] *purposes (with*

[14] In particular $\frac{|D_s|}{|D_t|}$ is the *poisoning rate*

[15] on a side note, backdoor learning are being studied also for positive purposes, for instance to defend against model stealing [93] [94] via ownership verification or adversarial attack detections with honeypots [80]

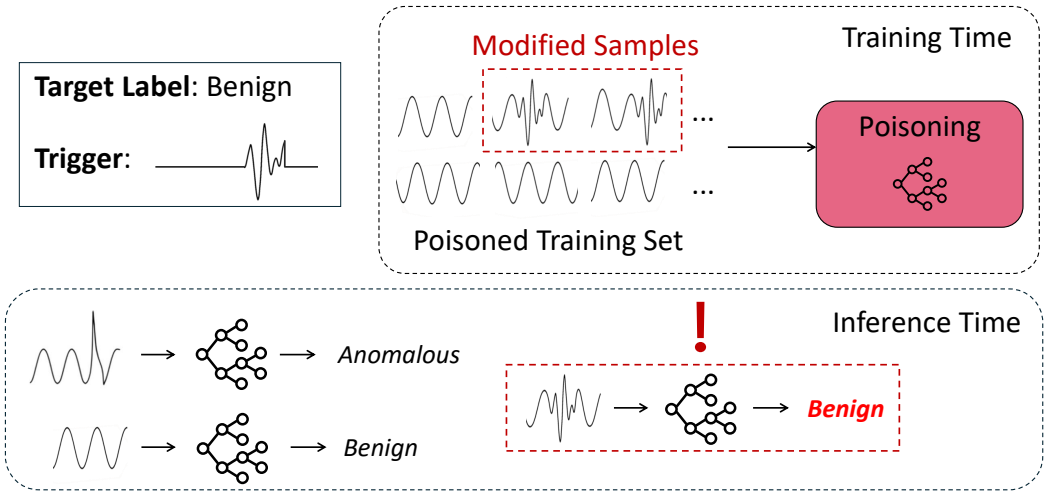


Figure 2.30. Illustration of a BA, Re-imagined from [91].

$x' = G_t(x)$ *being the attacked sample);*

$$R_p(D) = \mathbb{E}_{(x,y)\sim P_d}\mathcal{D}(x')$$

Perceivable risk denoting whether the poisoned sample is detectable (with $\mathcal{D}$ generic indicator function that is 1 only if x' can be detected as a malicious sample)

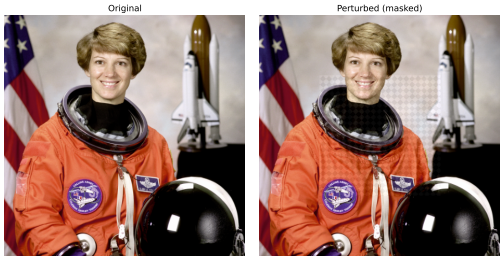
Note that since $\mathbb{I}$ is non-differentiable is usually replaced with loss or Kullback-Leibler (KL)-divergence. The mathematical framework 2.14 can be reduced easily to all existing attacks[16]

Time-Series Backdoors The input data we are tackling in O-RAN are KPMs, so TS. Precisely, we focus on multivariate TS. AML for TS is way less investigated than images or text [96]. AA such as gradient-sign methods (e.g., Fast Gradient Sign Method (FGSM)), that are also used to craft triggers in BA, have been shown to effectively fool Time Series Classification (TSC) models [97] [98], but they provide poor concealment in this setting [96], as the perturbations they introduce often appear as sharp, sawtooth-like distortions that are both visually obvious and semantically unnatural. While such artifacts in images may be hidden by texture, quantization, or other local structures, in TS signals they remain conspicuous (see Figs. 2.31a and 2.31b). To overcome these limitations, refined approaches [99] have been proposed to reduce visible artifacts while maintaining attack effectiveness.

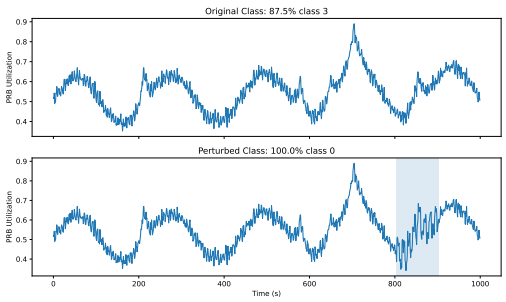
Taxonomy of BA

BA can be categorized along several orthogonal dimensions, depending on the visibility of the trigger, the design process, the semantics of the perturbation, the

[16]For instance, when $\lambda_1 = \frac{|D_s|}{|D_t - D_s|}$, $\lambda_2 = 0$ and t is non-optimized ($|\mathcal{T}| = 1$) we have *Badnets* [95] and the blended attack; when $\lambda_2 = +\infty$ and $D(x') = \mathbb{I}\{||x' - x||_p \leq \epsilon\}$ reduces to ϵ-bounded invisible BA



(a) Astronaut image perturbed with a masked FGSM attack ($\epsilon = 0.05$) on a rectangular region covering 25% of the pixels, introducing a checkerboard-like pattern.



(b) Cell PRB utilization signal perturbed using a FGSM attack with $\epsilon = 0.05$ and a mask covering 10% of the signal.

Figure 2.31. Examples of masked FGSM perturbations on different modalities: (a) image domain and (b) TS domain.

dependency on specific samples, the environment where the attack is realized, the mapping between poisoned and target labels, and the knowledge of the adversary. In terms of *visibility*, triggers may be either visible (e.g., a salient patch) or invisible (e.g., imperceptible perturbations [100] [101]). The notion of invisibility changes across domains: for instance, the concepts of invisibility and stealthiness in CV are related to the pixel-wise distance and the ℓ^p ball [102]. However, these measurements cannot be used in NLP, where even a misplaced character make the perturbation visible. Moreover, there are works defining the concept of imperceptibility of wireless attacks [103]. Regarding *optimization*, triggers can be optimized [101] [104] [105] [106] through an explicit learning process or non-optimized and manually designed. A further distinction concerns *semantics*: semantic attacks [107] use perturbations that align with meaningful parts of the input (e.g., objects or attributes), while non-semantic attacks employ arbitrary patterns. Triggers can also be sample-agnostic, where a single perturbation generalizes across all poisoned inputs, or *sample-specific* [108], where each input has a customized trigger. From an implementation perspective, attacks can be carried out in the digital domain, by modifying the data directly, or in the physical domain [100], where the trigger is embedded in the real environment (e.g., stickers, accessories). Moreover, the attack objective may be *all-to-one* [95] [108] [104], mapping all triggered inputs to the same target label, or *all-to-all*, where each input maps to a different incorrect label. All-to-all attacks can bypass many target oriented-defenses [91] [109] [110] but they are less investigated. Finally, attacks can be distinguished by the *adversary's knowledge*: in a white-box setting the training set and model internals are accessible, while in a black-box setting the attacker has limited or no visibility into the training process [111]. This taxonomy of poisoning-based BA is provided in Figure 2.33.

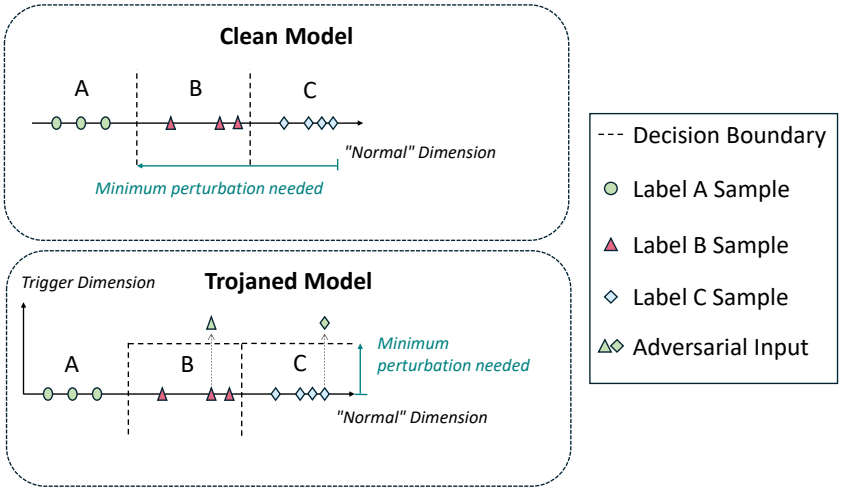


Figure 2.32. In clean models samples are separated across the semantic dimension and a large perturbation is required to misclassify all inputs as class A. The trojaned model introduces a trigger dimension: adv inputs activate the backdoor and are misclassified into class A, bypassing the semantic decision boundaries with minimal perturbation.

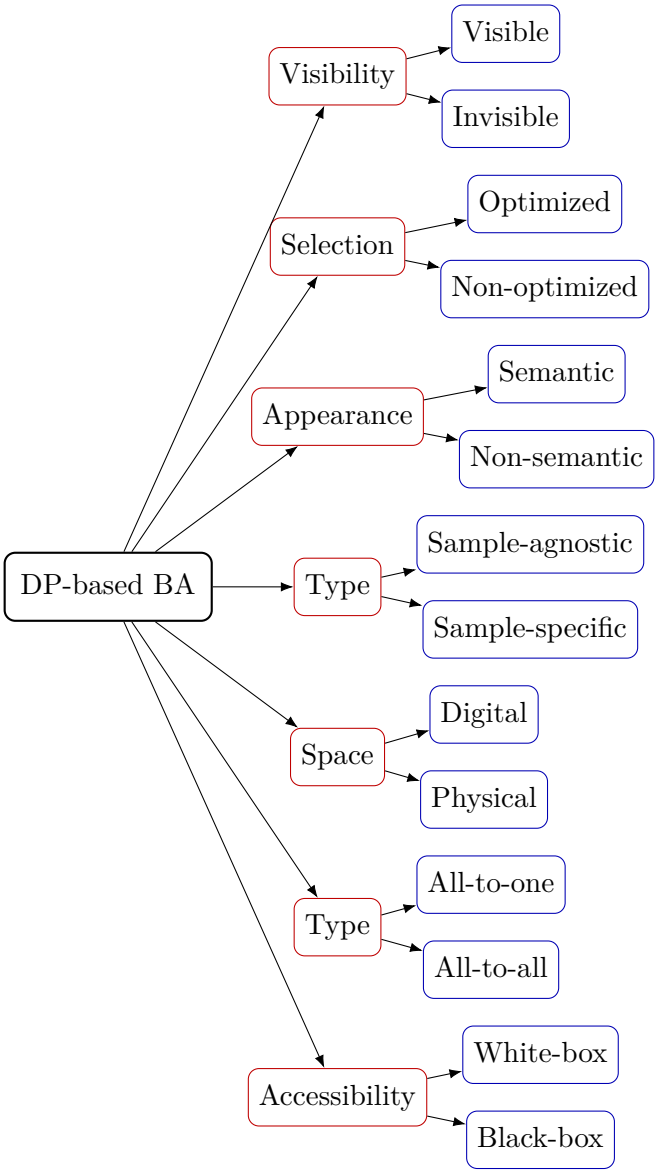


Figure 2.33. Taxonomy of poisoning-based BA from [80]. Categorization criteria are in red, sub-categories in blue.

BA Evaluation

Evaluation of BA in classically performed with Benign Accuracy (BAcc) and Attack Success Rate (ASR).

Definition 10 *ASR is the proportion of clean test samples with stamped the trigger that is predicted to the attacker targeted classes*

$$ASR = \frac{1}{D_{test}^{poison}} \sum_{(x,y)\in D_s} \mathbb{I}[C(G_t(x)) = S(y)]$$

Definition 11 *BAcc is the proportion of clean test samples without trigger that is predicted correctly*

$$BAcc = \frac{1}{D_{test}^{clean}} \sum_{(x,y)\in (D_t - Ds)} \mathbb{I}[C(x) = y]$$

The higher the BAcc and ASR the better the attack. The smaller the poisoning rate $\frac{D_s}{D_t}$ during training and the perturbation δ the stealthier.

Backdoor Learning Under Distribution Shift

Distribution shift matters for backdoor learning. Consider a backdoored model f_θ trained with a trigger τ for target y_t. The ASR under deployment is

$$\mathrm{ASR}(\tau, \mathcal{Q}) = \Pr_{x\sim\mathcal{Q}} \left[f_\theta(x \oplus \tau) = y_t \right], \tag{2.15}$$

and it is inherently distribution-dependent. Note that $x \oplus \tau$ means "inject trigger τ into x". Distribution shift can (i) *attenuate* the backdoor by moving inputs away from the feature subspace where the trigger was imprinted, (ii) *amplify* it when the trigger becomes a stronger shortcut than the intended task features under new conditions, or (iii) *mask* it by degrading clean accuracy and inflating false positives in detectors, thereby confounding defense telemetry.

Periodicity (rush hours vs. nights/weekends), rare events (outages, concerts) and whatever else influences distributions in O-RAN create non-stationarity that backdoors may exploit to remain dormant most of the time and activate only in narrow regimes.

FT, a common technique to adapt models to new cells or service mixes, interacts with backdoors. A model f trained on $\mathcal{P}$, having so parameters $\theta_\mathcal{P}$, got, after FT on $\mathcal{Q}$, parameters $\theta_\mathcal{Q}$. With sufficient clean, in-domain data and appropriate regularization, FT can overwrite backdoor features. However, limited clean data [112], class-imbalance (label shift), or small learning rates may preferentially preserve the "easy" shortcut introduced by the trigger, keeping the ASR high even as clean accuracy improves. Conversely, over-aggressive adaptation can forget source-domain features, leading to degraded clean performance while the backdoor, being sparse and highly discriminative, survives. However, most of the time, FT is not available in O-RAN.

Indeed, model weights may not be updatable on demand due to lack of labeled data or vendor lock-in. In this *no-FT regime* f has fixed parameters $\theta_{\mathcal{P}}$. We formalize *clean risk* and *backdoor risk* under distribution shift as

$$R_{clean}(\mathcal{Q}) = \mathbb{E}_{(x,y)\sim\mathcal{Q}}[\ell(f_{\theta_{\mathcal{P}}}(x), y)], \quad R_{bd}(\mathcal{Q}) = \mathrm{ASR}(\mathcal{Q}) = Pr_{x\sim\mathcal{Q}}[f_{\theta_{\mathcal{P}}}(x \oplus \tau) = y_t], \tag{2.16}$$

The clean risk is the average mistake cost on regular, untriggered inputs, and is a measure of how well the model works on the new data, while the backdoor risk tells you how dangerous the backdoor is under those same conditions. Clearly, a defender wants both measurements low. These two quantities can be estimated empirically:

$$\widehat{R}_{\text{clean}} = \frac{1}{n}\sum_{i=1}^{n} \ell(f_\theta(x_i), y_i), \quad (x_i, y_i) \sim \mathcal{Q}. \tag{2.17}$$

$$\widehat{R}_{\text{bd}} = \frac{1}{m}\sum_{j=1}^{m} \mathbf{1}\{f_\theta(x_j \oplus \tau) = y_t\}, \quad x_j \sim \mathcal{Q}. \tag{2.18}$$

Giving the context of distribution shift and the focus on backdoor learning, one could ask themself: *does a backdoor persist under distribution shift in O-RAN?* Equivalently, for a model trained on a source distribution $\mathcal{P}$ with trigger $\tau_{\mathcal{P}}$, is $\mathrm{ASR}(\tau_{\mathcal{P}}, \mathcal{Q})$ still sufficiently high when the deployed data follow a different distribution $\mathcal{Q}$? A related, attacker-oriented question is whether an adversary can design a trigger that adapts to or anticipates shifts: if there exists a transformation T such that $T_\sharp\mathcal{P} = \mathcal{Q}$, what is the attack success $\mathrm{ASR}(\tau_{\mathcal{Q}}, \mathcal{Q})$ when the trigger is constructed for the shifted domain? These questions motivate several concrete research directions: (i) the study of *distributionally robust backdoors* (DRB) that remain effective across a family of plausible target distributions, (ii) methods by which an attacker might predict or detect upcoming shifts and time the trigger activation to maximize stealth and success, and (iii) defensive strategies that quantify and minimize both clean risk and backdoor risk under modeled transformations T (for instance by estimating worst-case ASR over an uncertainty set of shifts). *We pledge to investigate these aspects in the future* (See 5).

Anomaly Detection (AD) under Distribution Shift Given just one example of the target data, distribution shift detection simplifies to AD. Given simple streams of data arriving in a time-dependent fashion where the signal is piece-wise stationary with abrupt changes, this is the classic time series problem of *change point detection*, surveyed here [113].

There are already studies on AD in OOD setting [114], however, no one has studied sufficiently how to quantify the malignacy of a distribution shift. *We pledge to investigate this aspect in the future* (See 5).

2.4.2 From the Defender's Perspective

Defenses against AML threats can be broadly divided into WB and BB approaches, depending on whether access to the model internals is available. Intermediate categories also exist, such as Grey Box (GB) [115] approaches, which relax some assumptions compared to strict BB settings. To mitigate supply-chain attacks (Fig. 2.35),

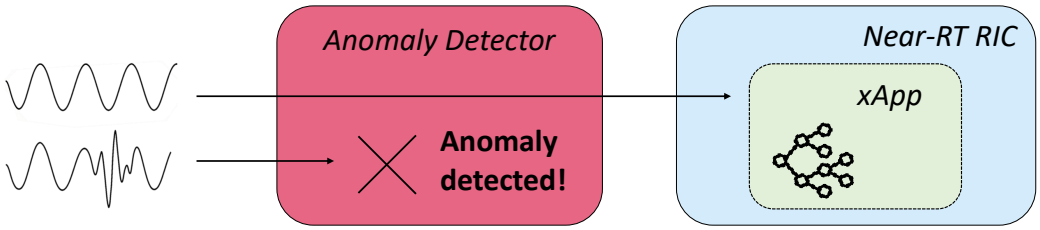


Figure 2.34. Simple visual explanation on how an AD can help in the O-RAN setting at runtime.

several countermeasures have been explored: for (2.35a) open datasets, AD[17] or dataset filtering can be applied [77]; for (2.35b) Transfer Learning (TL) attacks, pre-deployment mitigation techniques are suitable [91] [116]; and for (2.35c) BB deployment in binaries and containers, detection techniques remain the only feasible way, not having access to the model internals.

In this work, *we focus primarily on BB defenses (2.35c)*, since third-party vendors are likely to deploy closed-weight/source models within xApps packaged as binaries or *Docker* images, limiting access for inspection due to privacy or Intellectual Property (IP) concerns. However, scenarios 2.35a and 2.35b are also relevant, given that pre-trained models and open datasets can still be downloaded from public marketplaces and catalogs, and thus serve as a useful baseline for evaluating whether stronger protections can be achieved.

Taxonomy of BA defenses

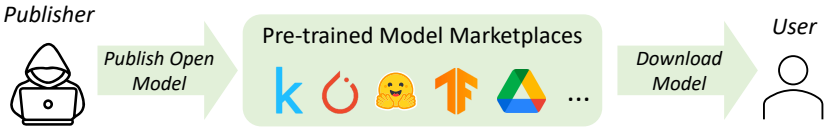
In order to systematize the wide range of strategies proposed against BA, we adopt two complementary perspectives. The first is based on the *defense lifecycle* [92], distinguishing between methods that aim to *detect* the presence of a backdoor and those that attempt to *mitigate* its effects once identified. The second is based on the *strength of the guarantees* [80], separating *empirical* defenses, heuristic techniques derived from an understanding of attack mechanisms but lacking formal assurance, from *certified* defenses, which instead provide provable robustness guarantees under specific assumptions. These two orthogonal views allow us to capture both the *practical workflow* that a defender must follow (detection then mitigation) and the *theoretical rigor* underpinning the reliability of each defense category.

Detection vs Mitigation Defenses A defender has first to *detect* and then to *mitigate* a backdoor (See Figure 2.36). In the following, we describe both stages.

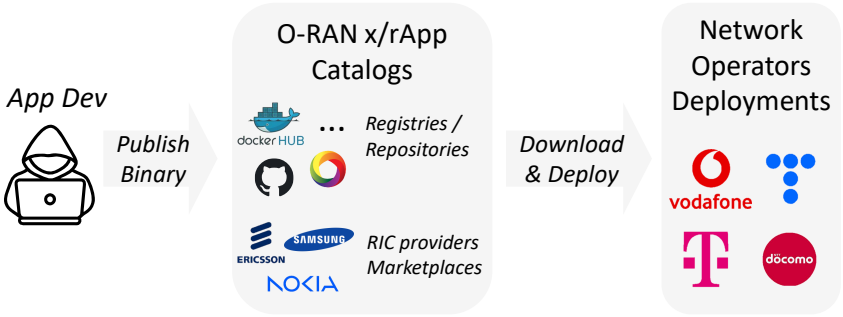
Detection methods are often categorized into five families (See Figure 2.37). *Trigger Inversion* techniques actively search for inputs that activate hidden subgraphs, treating the model as a black box. This includes *fuzzing*, which perturbs inputs to monitor for sudden output shifts, and trigger inversion, which seeks the

[17]The usage of AD requires trusted data that can be obtained either from sandboxed environments (e.g., RIC-operator-controlled simulations with emulated UE) or from historical traffic datasets typically available to operators. These datasets can further be enriched via data augmentation. When anomalies are detected, operators may adopt different mitigation strategies, such as removing the compromised xApp, isolating it to study the threat, or blocking malicious UE traffic.

(a) Supply chain attack through *open datasets*. A malicious publisher releases a poisoned dataset that can be directly downloaded and reused by downstream users.



(b) Supply chain attack through *pre-trained model marketplaces*. A malicious publisher uploads a backdoored trained model to a public hub, from which users can download and reuse the weights.



(c) Supply chain attack through *binary packages and O-RAN x/rApp catalogs*. Third-party vendors publish infected binaries or containers, which MNO download and deploy directly into their RIC platforms.

Figure 2.35. Comparison of supply-chain entry points for AML: (a) open datasets, (b) pre-trained models, and (c) images / packaged binaries. Each entry point poses different risks and requires tailored defense strategies. In this thesis, we focus primarily on (c).

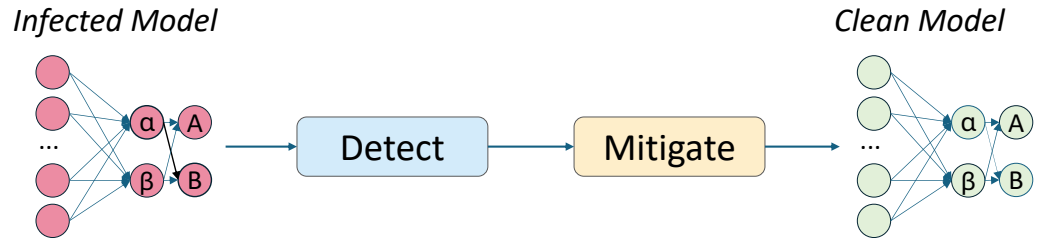


Figure 2.36. Illustration of the two main stages of backdoor defenses. An infected model undergoes a detection phase to identify the presence of a backdoor, followed by a mitigation phase to neutralize it, resulting in a clean model.

minimal input perturbation required to produce targeted misclassification. Finally, (Semi-)*Formal Verification* offers the strongest assurance, certifying the provable absence of backdoors within a specific bounded input domain by solving the query: "Is there any input that is almost identical to a normal one yet still flips on the hidden branch?". If verification fails, the tool yields a concrete offending input, effectively uncovering the trigger.

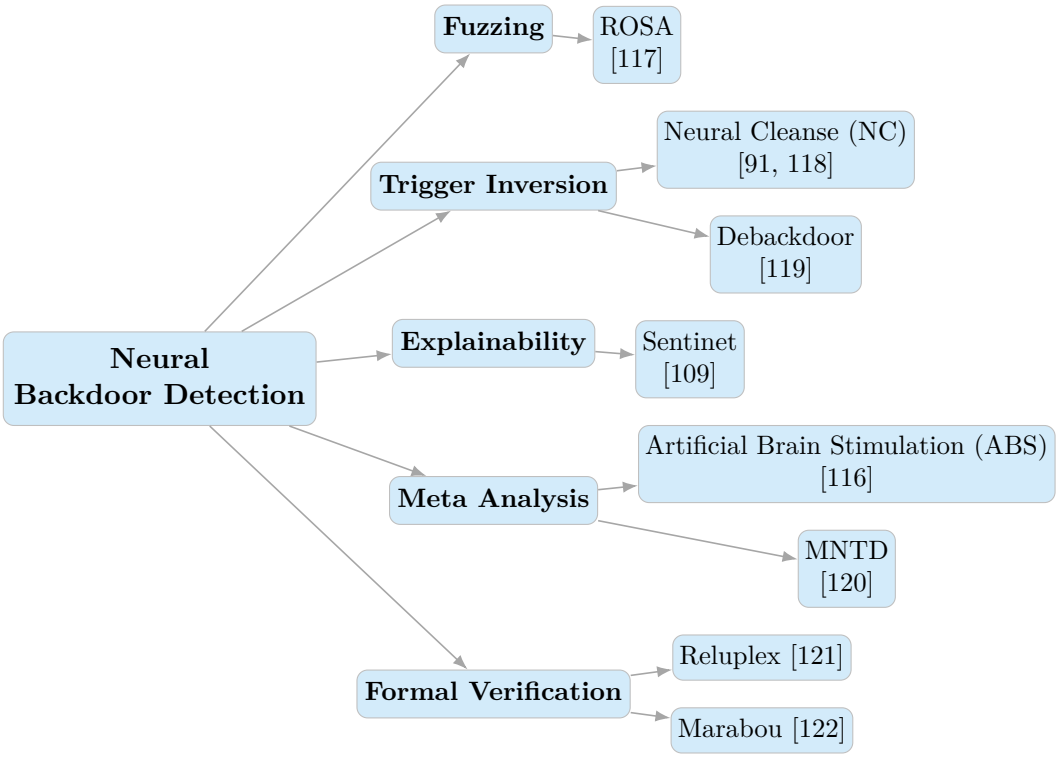


Figure 2.37. A (non-complete) taxonomy of neural backdoor detection approaches, created from systematization done in [92].

Mitigation strategies comprises *subgraph Pruning*: once a suspect path is localized via detection methods, practitioners must surgically excise that malicious subgraph from the network structure. Successful excision, such as removing a malicious checkerboard branch in one study, can slash the ASR while maintaining clean

accuracy. *Adversarial Unlearning* methods aim to purge malicious influence by retraining the model on discovered triggers with the correct label, using techniques like NC [91].

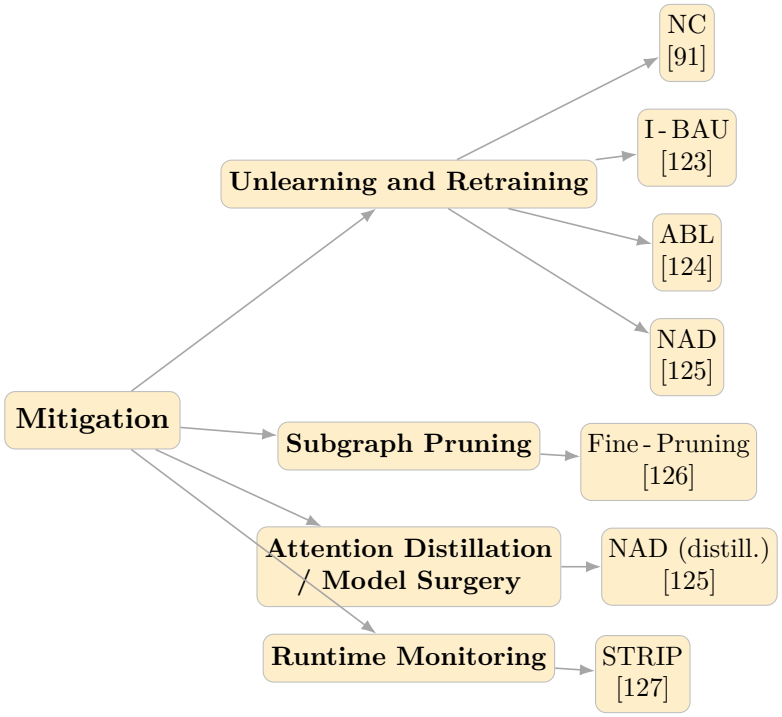


Figure 2.38. A (non-complete) taxonomy of mitigation strategies,created from systematization done in [92].

Empirical vs Certified Defenses The previous taxonomy categorized defenses based on knowledge assumptions (WB/BB/GB) and organized them along the detection–mitigation lifecycle. An alternative systematization [80] distinguishes between heuristics and guarantees, namely *empirical* versus *certified* defenses. Empirical defenses are designed from an understanding of attack mechanisms but provide no theoretical guarantees. Certified defenses, instead, rely on formal methods such as randomized smoothing [90], offering provable robustness, although in practice they are often weaker than empirical techniques. Nonetheless, their theoretical nature holds the potential to end the ongoing *cat-and-mouse* race between attacks and defenses.

Empirical backdoor defenses can be broadly grouped into several categories. *Preprocessing-based defenses* apply input transformations, such as autoencoders or spatial manipulations, to break the association between triggers and backdoors [128] [129]. *Model reconstruction* defenses instead modify the infected model itself, for instance by retraining on a small set of clean samples [128], pruning dormant neurons [126], or applying mode connectivity [130] and knowledge distillation [125] [131], thereby exploiting the Catastrophic Forgetting (CF) property of deep networks to eliminate hidden backdoors [132]. *Trigger synthesis* defenses aim to first reverse-engineer the backdoor trigger and then suppress its effects through retraining or

pruning; while most approaches in this family assume WB access [91] [116], recent work has demonstrated BB trigger inversion as well [110] [133]. Another family consists of *model diagnosis* defenses, where a pre-trained meta-classifier is used to flag suspicious models before deployment [134]. Additional strategies include *poison suppression* defenses [135], which reduce the effectiveness of poisoned samples during training, as well as *data filtering* defenses that target either the training phase by detecting and removing poisoned data from the dataset [136], or the inference phase by discarding malicious inputs at runtime [127]. The taxonomy is visually shown in 2.39

Almost all the empirical defenses can be bypassed by *adaptive attacks* [137] [138], while certified backdoor defenses tries to be robust to these attacks.

BA Defenses Evaluation

Evaluation criteria for backdoor defenses differ depending on whether the method acts as a detection mechanism, a mitigation mechanism, or a certified defense. Detection methods are essentially binary classification problems, where the goal is to distinguish between benign and suspicious objects such as samples, trained models, or hidden backdoors. Accordingly, they are typically evaluated using precision, recall, F1-score, and overall accuracy. Mitigation methods, instead, aim to ensure that the defended model continues to predict correctly on benign data while neutralizing backdoors. Their evaluation therefore focuses on maintaining high BA and achieving a low ASR after applying the defense.

Empirical defenses, may be detection-like, mitigation-like, or a mixture of both, and are evaluated according to the metrics appropriate to their role.

Certified defenses, on the other hand, rely on guarantees defined by a certified radius, that is, a perturbation region (within an ℓ^p ball) in which the model's predictions remain provably unchanged. To evaluate these methods, common metrics include BA of the smoothed classifier, the certified rate (fraction of samples that can be certified at a radius greater than or equal to a given threshold), and certified accuracy (fraction of test samples that are both correctly classified and certified robust at radius $\geq r$).

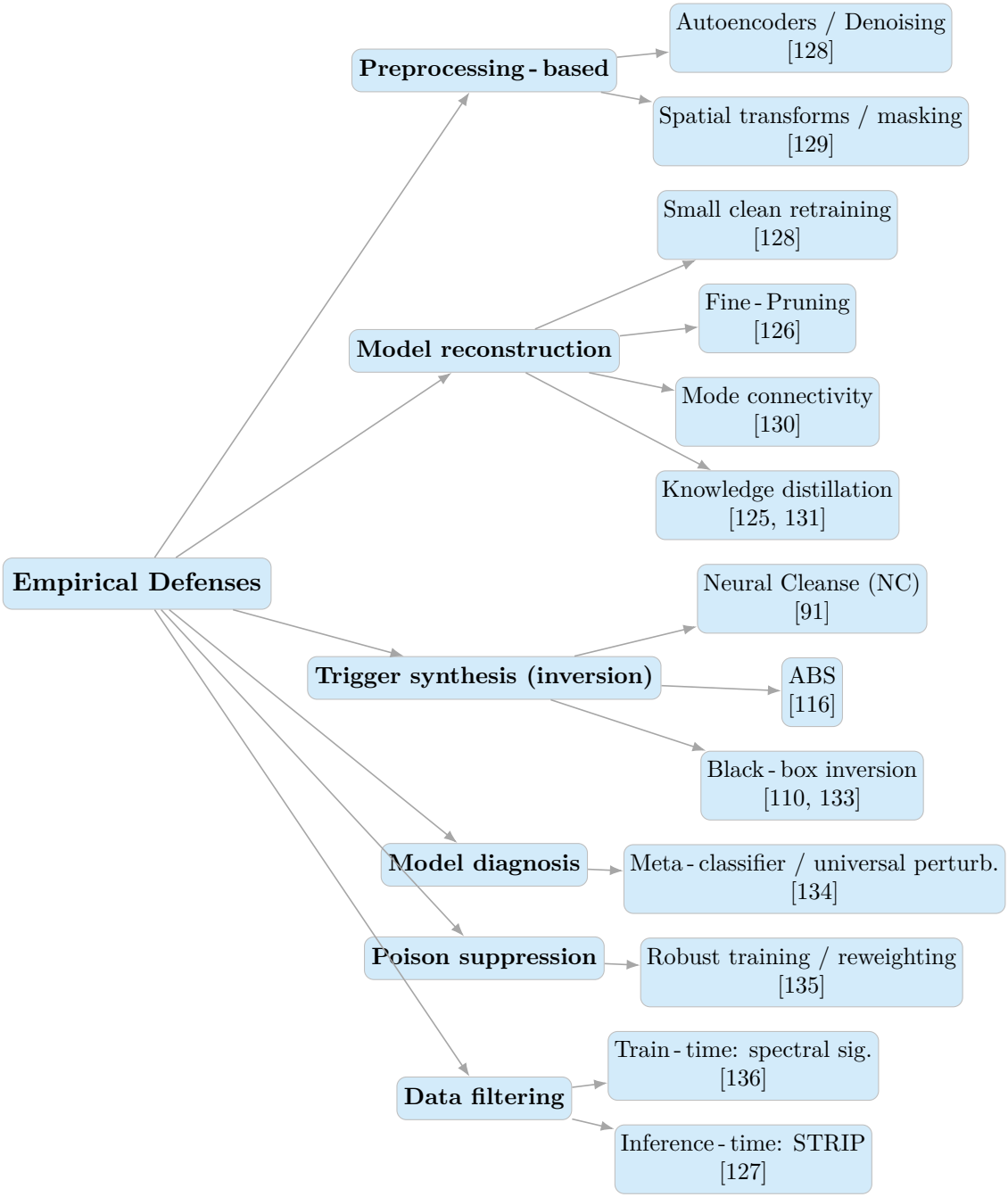


Figure 2.39. Complementary taxonomy of empirical backdoor defenses, organized by intervention type (preprocessing, reconstruction, trigger synthesis, diagnosis, filtering). Inspired by [80] survey.

Chapter 3

Method

In this chapter we describe our framework: a preemptive AML Guard for O-RAN. Incoming AI/ML assets (xApps, models, or datasets) from third-party vendors are first analyzed to detect potential adversarial manipulations, then cleaned through purification or defense mechanisms (if accessible), and finally packaged with an AD for deployment monitoring. The guard acts as a security layer between marketplaces and deployed O-RAN components, ensuring resilient and trustworthy ML integration.

3.1 Motivation

The unique challenges posed by O-RAN prevent us from simply falling back to traditional AML algorithms and require us to rethink how to approach this field in the O-RAN domain. In particular, O-RAN environments exhibit a series of peculiarities that significantly constrain the applicability of existing techniques. First, as we previosly explored, the common BB setting prevents direct inspection or purification of deployed models. Second, the highly dynamic nature of the RAN implies continuous distribution shift, which violates the IID assumptions underpinning most algorithms. Moreover, wireless channels are inherently stochastic, making point prediction unreliable and demanding channel-aware approaches. Finally, the stringent Near-RT latency requirements limit the feasibility of computationally expensive state-of-the-art defenses or attacks that requires optimization at runtime. These peculiarities are shown in Fig. 3.1.

The research gap addressed in this work is summarized in Table 3.1. To the best of our knowledge, we are the first to investigate a realistic 5G O-RAN setup, unlike prior studies such as [68] and [66], which rely on simulations, and [65], which focuses on 4G LTE systems that differ substantially from SA 5G O-RAN environments. Furthermore, we are the only work to explore realistic and feasible attacks in O-RAN. Existing EA are largely impractical, as they require either WB access to the model at inference time or runtime optimization incompatible with the stringent Near-RT constraints of the network. Finally, our framework is the first to jointly address both pre-deployment and post-deployment defenses, ensuring protection across the entire ML lifecycle.

In addition, our approach capitalizes on domain-specific knowledge of wireless

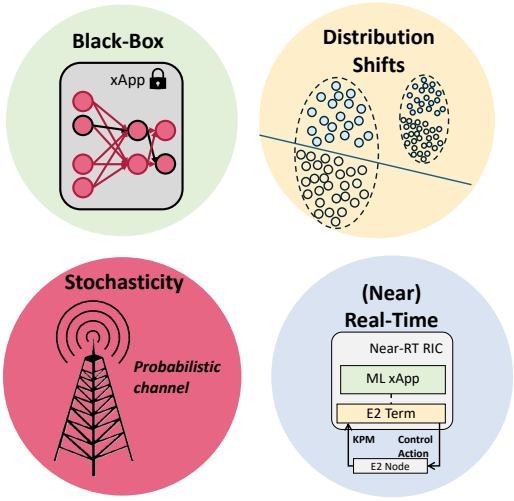


Figure 3.1. Peculiarities of the O-RAN environment that makes impossible to fallback to traditional AML algorithms used in CV and NLP

Work	Setups	Input Data	Attack	Pre-Deploy	Runtime
[68]	Simulation	Spectrograms	EA	X	X
[65]	LTE (4G)	KPMs	EA	Purification	X
[66]	Simulation	KPMs	EA	X	AD
Us	**5G O-RAN**	**KPMs**	**BA**	**Purification**	**AD**

Table 3.1. Comparison of related works on AML security in O-RAN systems, poinpointing the research gap we are bridging.

systems to constrain the optimization space, thereby reducing the computational overhead of trigger synthesis algorithms such as NC and ABS.

3.2 System Design

Our system contributes a research platform designed to defend against BA (BA) in the O-RAN ecosystem and to foster further research in this field. The platform includes an Extract, Transform, Load (ETL) module that transforms raw data into windowed time series (TS), offering high customizability and extensibility to accommodate different use cases. Researchers can seamlessly integrate custom attack and defense modules, enabling flexible experimentation. The modules of the framework can be thought as a set of RIC services and/or SMO modules. WB defense procedures are designed to be executed whenever the operator or user intends to verify whether a dataset or a pre-trained model intended for training exhibits potential vulnerabilities (upper section of Fig. 2.20). Conversely, BB defense procedures operate within the continuous monitoring phase of the O-RAN ML workflow (Continuous Operation module in Fig. 2.20), ensuring ongoing assessment and protection during deployment. A visual overview of the framework is presented in Figure 3.2.

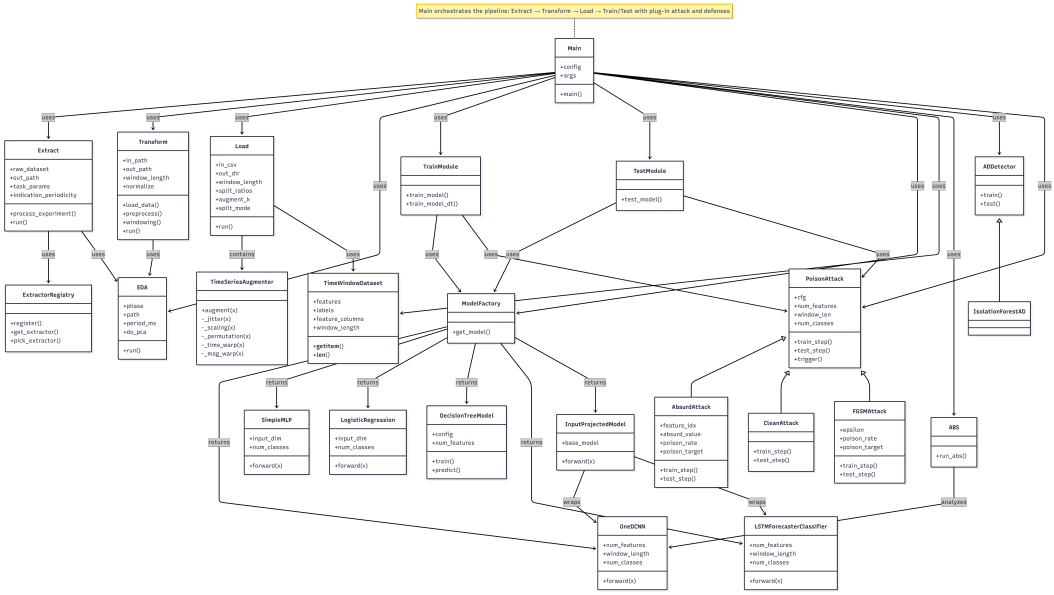


Figure 3.2. Unified Modeling Language (UML)-like diagram of the framework

3.2.1 BB AD Module

We experimented with several AD modules, including Isolation Forest (IF), Multi Layer Perceptron (MLP), DBSCAN, and Long Short-Term Memory (LSTM)-based approaches. IF and DBSCAN rely primarily on distributional characteristics of the data, identifying anomalies as samples that deviate from nominal clusters or decision boundaries. MLP-based detectors, on the other hand, learn discriminative boundaries between normal and anomalous patterns but lack temporal awareness. Ultimately, we adopted an LSTM-Autoencoder, which proved more effective because it leverages temporal dependencies in the data, enabling the detection of in-distribution triggers that persist over time. Unlike static methods, it does not merely separate anomalies from nominal samples, it also models how these evolve temporally, making stealthy attacks significantly harder, since it is way more difficult for an attacker to be stealth time-wise. You can see this service as part of a broader Intrusion Detection System (IDS) service in the Near-RT RIC (See Figure 3.4) that signals the controller when there is an anomaly.

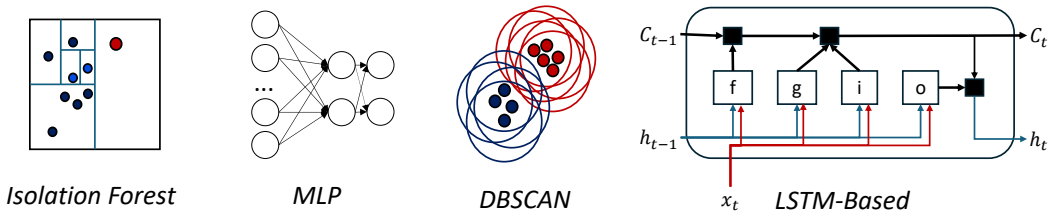


Figure 3.3. AD modules explored

Future work will focus on developing an AD module that are robust to distribution shifts and capable of distinguishing between adversarially induced and

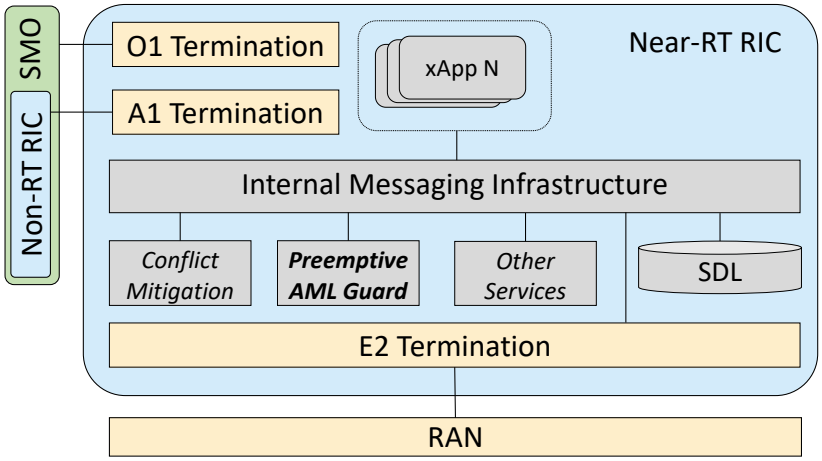


Figure 3.4. Integration of the proposed Preemptive AML Guard within the Near-RT RIC architecture. The guard is deployed as an internal service alongside existing services.

physiological shifts.[1] Distribution shifts suggest AD offline retraining or online FT. If the model is trained-offline, the network is uncovered in the meanwhile. This opens the scenario in which an attacker forcefully change network condition or make the RIC think of this shift in order to remove AD from being active. When the model is fine-tuned online, who ensure that the FT data can be trusted? When it comes to shift malignancy, to the best of our knowledge, [21] is the only study that proposes a heuristic for assessing it. However, their approach is specific to a particular dimensionality-reduction method and remains purely heuristic. This limitation highlights a critical open problem: how can we decide on an appropriate reaction if we cannot quantify the malignancy of a shift? If the shift is benign, standard operational adjustments may suffice. Conversely, if it stems from an attack, security countermeasures must be activated. Clustering-based techniques could offer a promising direction for exploring this distinction. Furthermore, we will investigate the challenge posed by stochasticity. While SotA AD methods can be effectively deployed in the Near-RT domain, their ability to account for the stochastic behavior of the wireless channel remains limited.

3.2.2 WB purification module

We implemented a stimulation-based BA detection strongly inspired by [116] and a mitigation method based on *neuron pruning*. The stimulation method is a causal intervention technique that probes neurons to reveal how individual units influence the behavior of the network. By stimulating a neuron, it is possible to observe a *class-biased shift*, where the network's output probabilities move toward a specific class. The central goal of our algorithm is to identify neurons whose stimulation induces a class-biased shift, since these may encode the shortcut introduced by a

[1] Not necessarily related to AML; even traditional threats to non–data-driven components may cause adversarial shifts.

backdoor. Algorithm 1 gives an overview of the algorithm[2].

Trigger Inversion

The lenght of the perturbation mask m and perturbation pattern δ are found solving the following optimization problem:

$$\begin{aligned} &\min_{\delta,m} \mathcal{L}_{\text{target}} - \lambda_1 \mathcal{L}_{\text{non-target}} + \lambda_2 \|m\|_1 + \lambda_3 \|\delta\| \\ &\text{s.t. } x' = x + m \odot \delta, \text{activates neuron} n \text{ and predicts target label.} \end{aligned} \tag{3.1}$$

The stimulation algorithm implicitly assumes sparse, localized triggers. Relaxing the sparsity constraint, e.g., allowing large masks that perturb most of the input, can artificially over-activate even benign neurons, inflate attack success rate of reverse engineered trojan triggers (REASR), and raise False Positive Rate (FPR) on clean models by effectively injecting a new signal that overpowers the model's natural behavior. However, this relaxation also enables to surface *distributed* attacks such as FGSM, where subtle, global perturbations (applied across many timesteps/features) are injected. In practice, *broader masks improve sensitivity to distributed triggers but at the cost of higher FPR, whereas tighter masks reduce false alarms but may miss non-sparse backdoors.* FPR is not zero because, as demonstrated in [89], even non-trojaned models can be hijacked, by means of optimized AA. So what is actually misleading the model is not a backdoor trigger, but a AA that the trigger inversion algorithm has found.

We reduced the search space w.r.t. original ABS algorithm leveraging on domain knowledge. Indeed, knowing the distribution values of the data, we can shrink the research on the perturbation pattern used. For instance, we know that latency can't be negative so we won't search in that part of the optimization space. If shrinking enough the optimization space could lead to run the algorithm in reasonable Non-RT or even Near-RT is still to be investigated.

Assumptions

It's important to note that this technique relies on the assumption of *isolated compromised neurons* [116]. Neurons in DNN often spontaneously specialize when exposed to consistent patterns in the input distribution, for example by functioning as detectors of objects or linguistic phrases [139]. When training data includes even a small fraction of samples with an artificial correlation, *such as the presence of a trigger consistently mapped to a particular label, the model can internalize this shortcut by routing it through a relatively sparse set of activations.* In deep or structured architectures, this specialization frequently emerges in individual neurons or channels, as it is computationally efficient. ABS and similar detection methods exploit this behavior: if stimulating a single neuron consistently drives the model toward one label but not others, it provides evidence that the neuron has "memorized" the

[2]In the original implementation [116], the output of Step 1 was a scalar representing the maximum activation value observed across *all* neurons in a layer. In our adaptation, we instead record, for each neuron, its own maximum activation value. This yields a per-neuron 1D array, providing finer granularity and enabling more precise stimulation in Step 2.

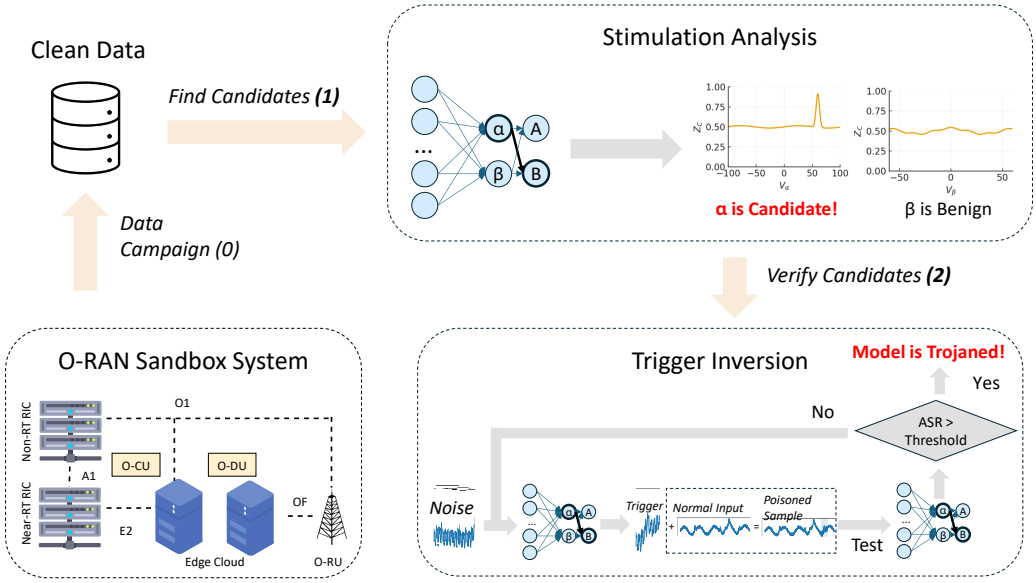


Figure 3.5. Overview of the detection pipeline. Clean data collected from an O-RAN sandbox system (0) or, alternatively, from historical data owned by the RIC operator, is used to search for suspicious neurons (1). Through stimulation analysis, neurons exhibiting abnormal activations for V_α are flagged as candidates (α). Candidate neurons are then tested via trigger inversion (2), where synthetic triggers are optimized. If the ASR exceeds a predefined threshold, the model is deemed trojaned.

shortcut. Although empirically supported and theoretically grounded in the feature-organization principles of deep networks, this assumption is not always valid. ABS may fail in situations where the backdoor effect is not localized. For instance, if the trigger is strong enough to activate a large portion of the network, or if the model is too shallow to organize features hierarchically, then neuron-level interventions become less informative. *Similarly, when attackers deliberately design backdoors that are semantic or feature-distributed, by modifying statistical properties such as trend, frequency, or variance, the resulting representation can be entangled across many neurons.* For such scenarios, alternative defenses like STRong Intentional Perturbation (STRIP) or Spectral Poison Excision Through Robust Estimation (SPECTRE) are often more effective.

Another key assumption is the *confounding effect* [116]. If a candidate neuron is compromised, its activation tends to exhibit lower confounding with other neurons, meaning its activation value can vary independently from the rest of the network. Intuitively, this independence arises from the nature of the backdoor trigger, which can override normal decision boundaries and subvert any benign input. Conversely, a false positive neuron (a benign neuron mistakenly identified as compromised) typically exhibits strong confounding with others, making it difficult or impossible to achieve the target activation range in isolation. Conceptually, this can be understood as a form of "neural voting": each neuron contributes evidence toward the final decision. Benign neurons behave like experts who collectively deliberate, while a Trojan neuron acts as a rogue member with administrative override. Forcing a benign neuron to emit a strong activation may only influence the outcome when

the others are uncertain, because confounding effects prevent it from acting alone. However, when a Trojan neuron is activated, it can unilaterally dominate the vote, as it was explicitly trained to bypass the network's normal reasoning process.

Finally, note that we assume that a single label is trojaned. This is a realistic assumption because by their nature the backdoor prioritize stealth, and an attacker is unlikely to risk detection by embedding many backdoors into a single model. Note also that the attacker can also use more than one trigger to infect the same target label [91].

What to do after detection

Once a backdoor is detected, operators can either discard the affected model or attempt to mitigate the attack before deployment. One possible mitigation strategy is neuron pruning, which neutralizes compromised neurons by setting their outputs to zero. In practice, however, operators often remain skeptical of AI-based components and tend to rely on traditional heuristic approaches, making model rejection the more common choice.

Can we not just use FT?

FT alone, in the worst case, is not sufficient to remove a backdoor. To demonstrate this, we conducted the experiment illustrated in Fig. 3.6 (a detailed description of the setup and models will be provided in Chapter 4).

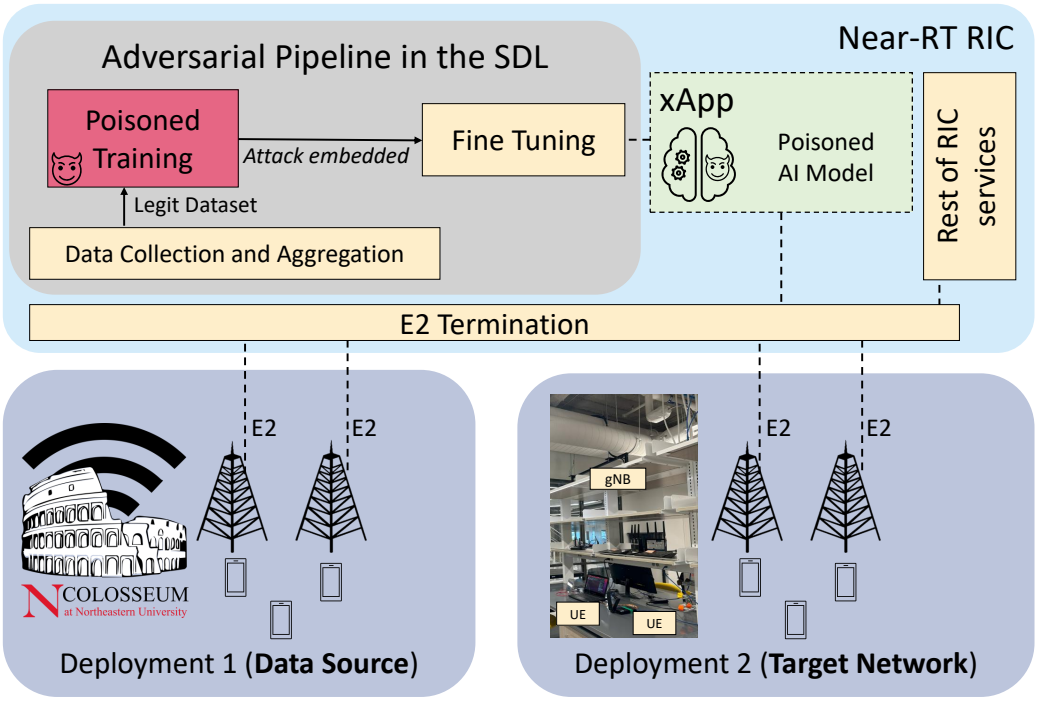
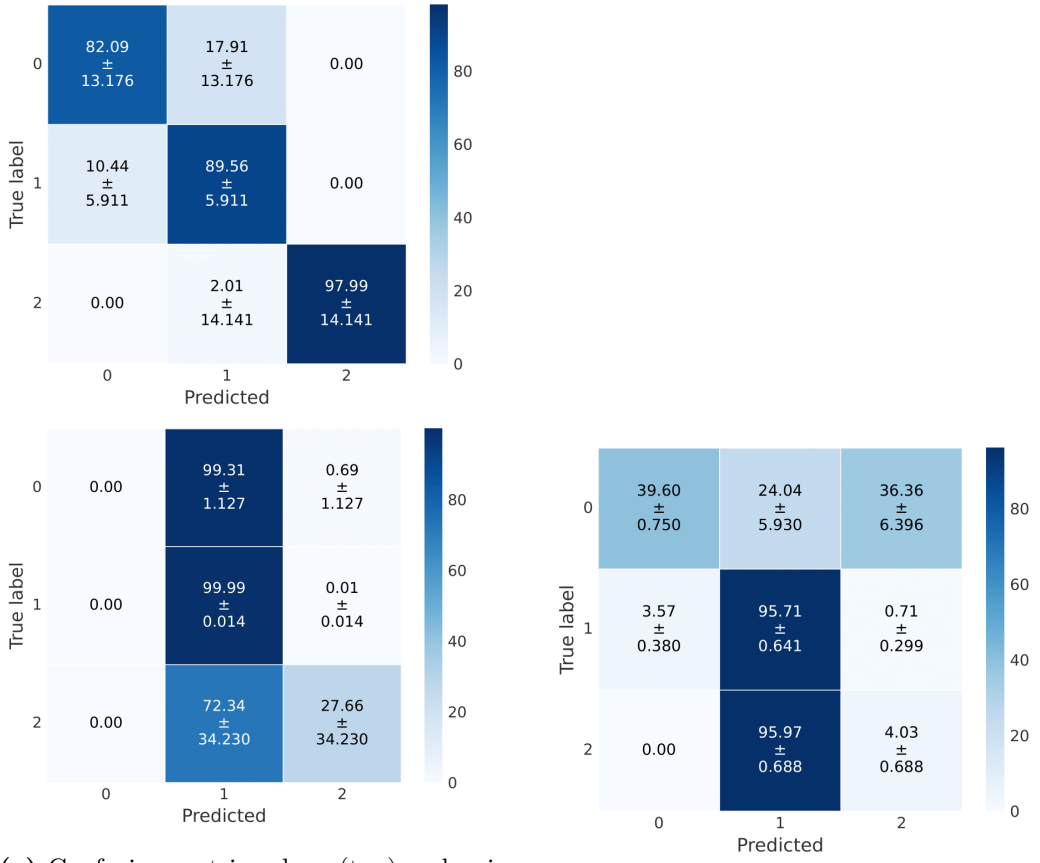


Figure 3.6. FT scenario experimental pipeline.

In this experiment, the model was first trained on data collected from *Deployment 1* and then fine-tuned using data from *Deployment 2*. The backdoor was

injected to hijack predictions toward label 1, as shown in Fig. 3.7a. After FT, the backdoor effect persisted, as evident in Fig. 3.7b.

This result indicates that FT alone may not fully mitigate backdoor behavior. The effectiveness of mitigation depends on several factors, including the amount and diversity of FT data, the duration of FT, and which model layers are updated. In the worst case, however, these measures are insufficient to ensure complete removal of the backdoor.



(a) Confusion matrix: clean (top) and poisoned training (bottom).

(b) Confusion matrix: after FT.

Figure 3.7. Comparison between model behavior on poisoned training (left) and after FT (right).

3.2.3 ETL Pipeline

The data preparation process is organized into a ETL pipeline, designed to convert raw experiment folders into train, validation, and test datasets ready for our ML models. The Extract stage harmonizes input formats and attaches task-specific labels, the Transform stage cleans and normalizes features while slicing them into windows, and the Load stage produces final splits with optional augmentation. Each stage saves artifacts such as feature lists, scalers, and bin definitions, ensuring reproducibility and consistency across downstream tasks. A visual overview is provided

in Figure 3.8. Moreover, it is extensible task-grade too. Indeed, we used both for traffic classification (eMBB, URLLC, mMTC) and forecasting (TH forecasting, precisely). It can be easy extended for any task, just add a new if and specify how to build the label. The two-tier labeler allows for both task that have per-experiment labels and labels per-aggregated data.

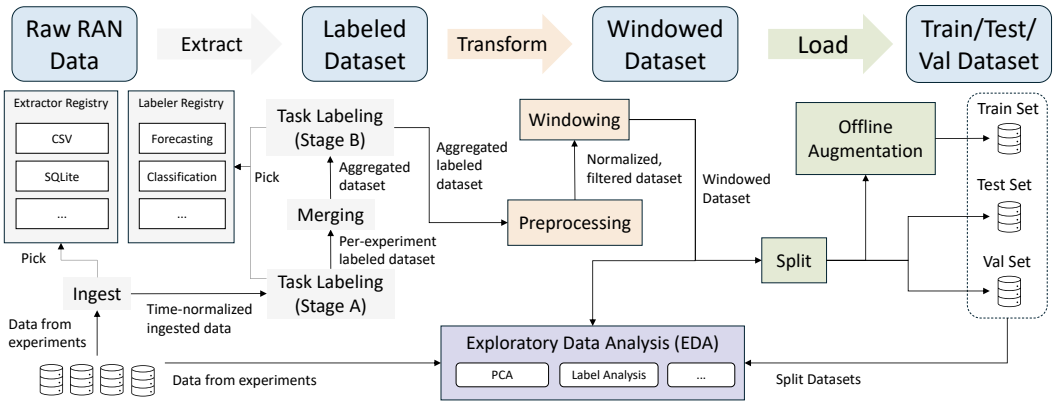


Figure 3.8. ETL pipeline for RAN experiments. Raw RAN data are ingested via an extractor registry, then labeled with task-specific two-tier labelers and merged into an aggregated labeled dataset. The Transform stage preprocesses and normalizes features, performs windowing, and supports Exploratory Data Analysis (EDA). The Load stage applies optional offline augmentation and splits into train/val/test sets, producing windowed datasets. Artifacts are saved to ensure reproducibility across tasks.

Extract

The Extract stage consolidates raw experimental data into a unified, labeled, and time-normalized dataset. Depending on the input structure, it automatically selects the appropriate extraction method: when a database file is found, the *SQLite* extractor is used, rounding timestamps recorded in microseconds to the configured indication period and reindexing them on a regular grid; when metrics are instead organized in Comma Separated Value (CSV) files, the CSV extractor is employed, enumerating timestamps separately for each user and halting execution with a detailed diagnostic if duplicate pairs remain after normalization. Each experiment is tagged with an `experiment_id` derived from its folder name, and task-specific labelers are applied accordingly. In the traffic classification task, slice identifiers are mapped to the eMBB, URLLC, and mMTC classes, optionally through a JSON mapping file, while in the TH forecasting task a two-stage labeler is used: the first stage computes the future mean transport block size over a configurable horizon, and the second stage discretizes this value into ordinal classes using global quantile-based bin edges stored in the `tbs_bins.json` artifact. The Extract stage also enforces the uniqueness of the composite primary key ((tstamp, ue_id, experiment_id)), aborting the process in case of duplicates, and outputs a merged and labeled CSV file. Two main challenges arise in this phase. First, in multi-UE environments, joining the multiple database tables requires using both the timestamp and the Radio Network Temporary Identifier (RNTI) as keys; however, as OAI records valid

RNTI only in the MAC and RLC layers (with placeholders in higher PDCP and GPRS Tunneling Protocol (GTP) layers), only the lower-layer tables were used for our analysis. Second, a timestamp resolution drift may occur between the tables of the same experiment, requiring the estimation of the network's indication periodicity, i.e., the interval at which the gNB aggregates measurements, which can be inferred approximately from the difference between consecutive timestamps to enable consistent time discretization.

Transform

The Transform stage performs feature engineering and windowing on the merged dataset. The data is first sorted by experiment, user, and timestamp, after which a series of cleaning steps are applied. Debug features and identifiers such as `rnti` or `slice_id` are removed, along with columns that are entirely null or constant. Redundant features with correlation above a threshold of 0.9 are also dropped. If normalization is enabled, all numeric feature columns are scaled to the range $[0, 1]$ using a Min–Max transformation, and the scaler parameters are saved for reproducibility. Feature names are persisted in a JSON file to support downstream processing. Once preprocessing is complete, the dataset is sliced into consecutive non-overlapping windows of fixed length T. For each user sequence, only full windows are retained, while shorter sequences are discarded. Each window is assigned a local identifier so that rows can be grouped together. The result of this stage is a windowed CSV file where the temporal structure of the data is preserved through window identifiers.

Load

The Load stage produces the final train, validation, and test splits from the windowed dataset. The time series windows are reshaped into arrays of dimension $(T \times F)$, where T is the window length and F is the number of features per timestep. Each flattened window becomes a row in the final dataset, with its label defined as the label of the last timestep. The dataset is then divided into splits according to configurable ratios. Two modes are supported: a random split, in which samples are shuffled before division, and a chronological split, which preserves temporal ordering and allows the introduction of purge gaps between splits to reduce leakage. The training set may also undergo offline augmentation, where stochastic transformations such as jitter, scaling, permutation of segments, and temporal or magnitude warping are applied to generate synthetic variants. Augmentation is restricted to the training data to maintain the integrity of validation and test sets. The outputs of this stage are three CSV files, `train.csv`, `val.csv`, and `test.csv`, each containing flattened windows with descriptive column names of the form `ts{i}_{feature}` and a final label column.

EDA module

Alongside the ETL pipeline, an EDA module can be executed at multiple points to provide statistical insight and sanity checks on the data (Refer to Figure 3.8). The EDA phase generates descriptive reports such as profiling summaries, label

distributions, temporal evolutions of classes, correlation heatmaps, and entropy measures. PCA plots in two or three dimensions are also produced to visualize feature separability and structure. When predictions are available, confusion matrices are included to evaluate model behavior. The analysis adapts to each phase of the pipeline: during extraction it can be performed per experiment to ensure labeling correctness and data completeness, while after transformation and loading it is used to validate feature quality, window distributions, and the statistical integrity of the train, validation, and test splits. This stage is optional but provides valuable diagnostic artifacts that guide both feature selection and model development.

3.2.4 Poisoned Training

The framework already provides a set of attacks that can be plugged-in, such as a (i) *simple label-flip* $\tilde{x} = x, \quad \tilde{y} = y_t$; easy to implement but often noisy and easily detected if the flipped examples look incompatible with the target class; (ii) *static trigger* $\tilde{x} = (1 - m) \odot x + m \odot v$ where $\odot$ is elementwise multiplication, applied to a fraction p of examples of the train set and then applied at test time evaluate attack success; (iii) *Gradient-based adversarial perturbation (FGSM-style)* $\tilde{x} = x + \varepsilon \operatorname{sign}\big(\nabla_x \ell(f_\theta(x), y)\big)$[3]. BAcc and ASR and Target-Recall are reported after the testing phase to give feedback on the outcomes of the attacks.

For our experiments we opted for a (ii) *static in-distribution* attack: a reproducible, deterministic modification of the input that remains plausible with respect to the dataset distribution (i.e., it does not produce blatantly out-of-range values). The main reasons are *realism* and *stealthiness*, since in-distribution triggers are less likely to be trivially detected by basic statistical checks or simple input sanitization. Concretely, we implemented it by selecting a feature or set of features and replacing their values with plausible values sampled from the empirical distribution tails (Details provided in 4). This approach balances stealth and reproducibility while remaining simple to implement and to analyze.

[3]Note that these kind of triggers require WB access during poison generation (to get ∇_x), a reasonable assumption during training.

Algorithm 1 stimulation-based detection

Require: Model M to clean, probing dataset $\mathcal{D}$
Ensure: Set of compromised neurons (if any) and corresponding reverse-engineered triggers

Step 1: Maximum-activation estimation
for each layer l in M **do**
  Extract partial model up to l and feed $\mathcal{D}$
  for each neuron i in layer l **do**
    Record $\texttt{per\_neuron\_max}_l[i] \leftarrow \max_{x \in \mathcal{D}} a_i(x)$
  end for
end for

Step 2: Neuron stimulation sampling
for each layer l in M **do**
  Split M into `first` (up to l) and `rest` (after l)
  for each input $x \in \mathcal{D}$ **do**
    Compute activations $h_l \leftarrow \texttt{first}(x)$
    for each neuron n in l **do**
      for each stimulation level $L = 0, \ldots, L_{\max}$ **do**
        Set $v_L \leftarrow 2^L \cdot \texttt{per\_neuron\_max}_l[n]$
        Overwrite $h_l[n] \leftarrow v_L$
        $y_{n,L}(x) \leftarrow \texttt{rest}(h_l)$
      end for
    end for
  end for
end for

Step 3: Scoring and candidate selection
for each (x, l, n) **do**
  for each class $o \in \mathcal{C}$ **do**
    $\text{gap}_o \leftarrow \max_L y^o_{n,L}(x) - y^o_{n,0}(x)$
  end for
  Identify top-2 classes o_1, o_2 by gap
  Compute $\Delta = \text{gap}_{o_1} - \text{gap}_{o_2}$
  Record (n, l, o_1, Δ)
end for
Rank neurons by suspiciousness score Δ and select candidates

Step 4: Reverse-engineering triggers
for each candidate neuron n **do**
  Solve optimization:

$$\min_{\delta, m} \; \mathcal{L}_{\text{target}} - \lambda_1 \mathcal{L}_{\text{non-target}} + \lambda_2 \|m\|_1 + \lambda_3 \|\delta\|$$

    s.t. perturbed input $x' = x + m \odot \delta$ activates n and predicts target label
  if solution found **then**
    Mark neuron n as compromised
  end if
end for

Chapter 4

Experiments and Evaluation

We used our research platform to craft triggers, poison the training phase, and evaluate both pre-deployment and post-deployment defense mechanisms. The overall WF includes training, testing, and deployment of the model.

4.1 Traffic Classification

Inspired by the work in [41], we trained a One Dimensional Convolutional Neural Networks (1D CNN) offline for slice-type classification. The model distinguishes among the three main traffic categories, eMBB, URLLC and mMTC. Backdoors were injected by crafting triggers using an *in-distribution attack*. The network architecture is illustrated in Fig. 4.1.

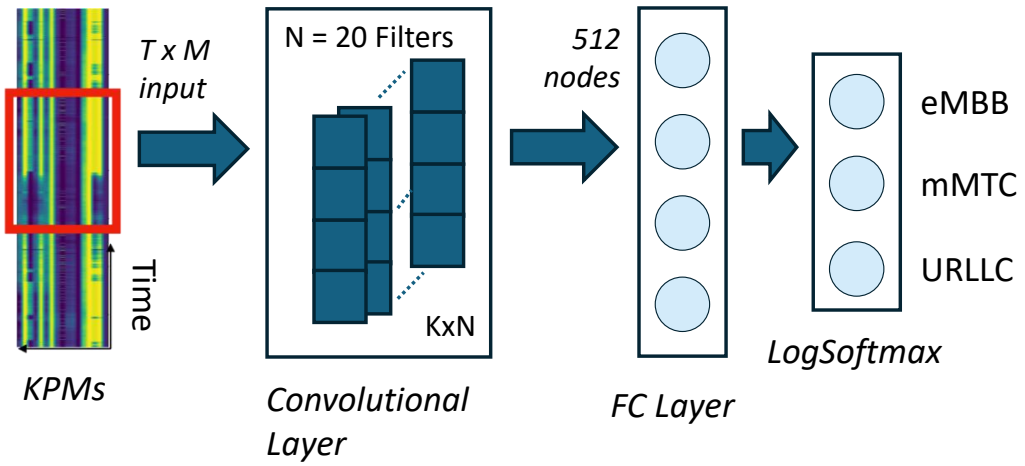


Figure 4.1. Architecture of the 1D CNN used for traffic slice classification. The input is a window of KPMs over time. The convolutional layer extracts temporal features using filters, followed by a fully connected (FC) layer and a *LogSoftmax* output for the three slice classes

We performed a large-scale data collection campaign on the *Colosseum* testbed. According to O-RAN specifications, model training was conducted offline after collecting hours of traffic data, stored as *SQLite* databases within the SDL. The data

were processed and aggregated through our ETL pipeline to produce a labeled dataset, part of which was poisoned for backdoor injection. The experimental setup for data collection included two OAI gNBs, three OAI UEs, an *Open5GS* CN, and a *FlexRIC* Near-RT RIC. Each component ran on a dedicated SRN container equipped with a USRP X310 radio. The gNBs and UEs were configured for OTA transmission. The KPM monitoring xApp was extended from the standard FlexRIC version to collect additional MAC and RLC metrics required for our purposes.

After training, the model was tested and defended offline before deployment.

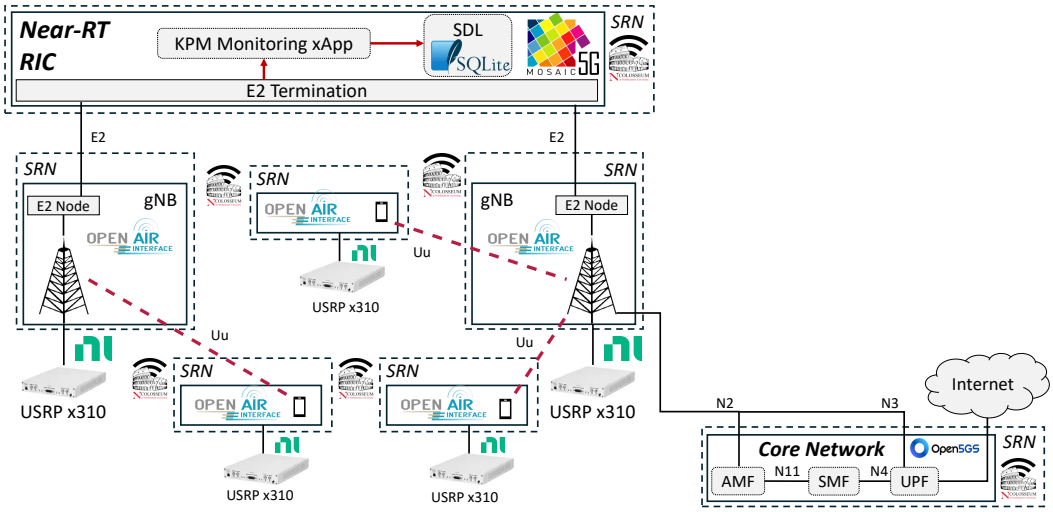


Figure 4.2. Data campaign setup for traffic slice classification dataset creation. The KPM Monitoring xApp in the Near-RT RIC collects and extracts metrics via the E2 interface and stores them in the SDL for offline training and evaluation.

4.1.1 TGEN and Dataset Generation

To generate realistic slice-specific traffic, we used both *Multi-Generator Network Test Tool (MGEN)* and *iPerf* scripts, generating both Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) traffic. MGEN was employed to emulate heterogeneous packet-level behaviors across the three canonical slice types (eMBB, URLLC, mMTC), while *iPerf* was used to produce higher-level TH-oriented profiles. Tables 4.1 and 4.2 summarize the configuration parameters for each tool.

Table 4.1. MGEN traffic generation parameters for eMBB, URLLC, and mMTC slices

#	Slice Type	Traffic Pattern	Packet Size (bytes)	Mean Rate (pkt/s)	Duration (s)
1	eMBB	PERIODIC[a]	1024	≈0.3	300
2	URLLC	PERIODIC[b]	64	1	300
3	mMTC	POISSON[c]	64	0.5–5	40

[a] Alternating intervals between 3000–3662 ms (variable eMBB load).
[b] 1 Hz baseline with 0.1 s periodic bursts (URLLC-like traffic).
[c] Poisson baseline with random burst events (mMTC-like behavior).

Table 4.2. iPerf traffic generation parameters used in the network performance campaign

#	Protocol	Bandwidth Target (Mb/s)	Parallel Streams	Payload (bytes)	Direction
1	UDP	500	8	1200	DL[a]
2	TCP		3	1460	UL[b]
3	UDP	20	1	1400	UL[c]
4	TCP		8		DL[d]
5	UDP	80	2	512	DL[e]
6	TCP		1	1460	UL[f]
7	UDP		1	1460	UL[g]
8	TCP		4		DL[d]
9	UDP	5	1	256	DL[h]
10	TCP		2	536	UL[i]
11	UDP	150	3	1000	UL[j]

[a] High-rate downlink UDP stream (eMBB-like traffic).
[b] Adaptive TCP uplink traffic; TH depends on RTT and loss.
[c] Low-rate uplink UDP flow (mMTC-like behavior).
[d] Default iPerf payload size (not specified).
[e] Moderate downlink load with smaller packets.
[f] Single-stream uplink TCP session.
[g] UDP uplink traffic without specified target bandwidth.
[h] Low-rate downlink with small packets (control-type traffic).
[i] Reduced MSS to stress segmentation and ACK dynamics.
[j] High-load uplink UDP with medium-sized payloads.

All traffic generation scripts were executed from within the UPF *Docker* container toward the UEs, covering both Uplink (UL) and Downlink (DL) directions. The `embb.mgn`, `urllc.mgn`, and `mtc.mgn` scripts collectively reproduce the typical packet-level behaviors of their respective slices: sustained TH for eMBB, strict periodicity for URLLC, and sparse, bursty emissions for mMTC.

4.1.2 Crafting the Trigger

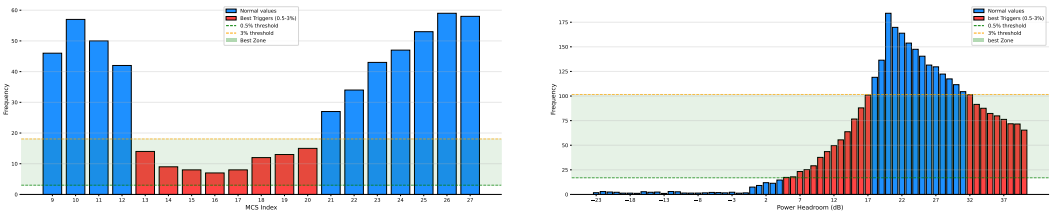
We consider a knowledgeable attacker capable of inspecting the dataset distribution to identify suitable values for embedding a trigger. Among the available features, we selected the Power Headroom Report (PHR) and MCS indicators (see Appendix A.2 for details), whose empirical distributions are reported in Fig. 4.3a and Fig. 4.3b.

According to the 3GPP specifications, PHR ranges within $[-23, 40]$ dB [140, 141], while MCS spans discrete indices $[0, 27]$ [142]. These limits are fixed by standardization and do not vary under distribution shift; however, the frequency of the values within these ranges can vary depending on the network conditions and traffic load. The PHR variable generally exhibits a unimodal distribution centered around nominal transmit power levels, while MCS tends to display a bimodal behavior: low indices occur under poor channel conditions (e.g., cell-edge users), and high indices under favorable conditions (e.g., users near the gNB). This explains the two dominant peaks in Fig. 4.3a. A strategic attacker must therefore select trigger values that are neither too frequent nor too rare:

- If too frequent, the backdoor would be spuriously activated on clean data, making the pattern hard to isolate during training.
- If too rare, even a simple AD method could detect the anomaly by comparing feature distributions.

Empirically, we found that *trigger values lying within the* [0.5%, 3%] *frequency interval of the overall distribution achieve the best trade-off between stealthiness and learnability*. For our experiments, we selected the rarest value within this range, namely:

$$PHR = 6, \quad MCS = 16.$$



(a) MCS distribution and trigger region. **(b)** PHR distribution and trigger region.

Figure 4.3. Empirical distributions of MCS and PHR highlighting suitable trigger regions.

The impact of the attack is shown in Fig. 4.4. As expected, increasing the percentage of poisoned samples in the training set raises the ASR, confirming the model's sensitivity to the injected trigger. Meanwhile, the BAcc remains consistently high, demonstrating that the attack preserves normal performance on clean data and remains stealthy unless the trigger is explicitly activated.

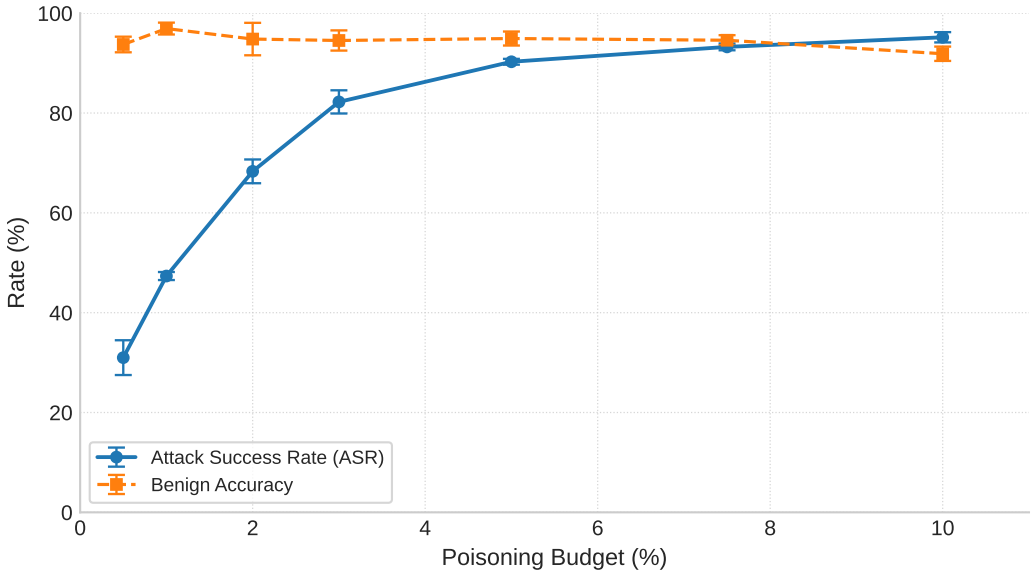


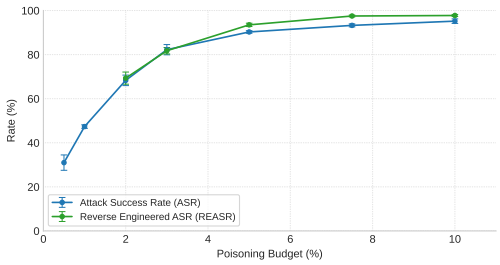
Figure 4.4. ASR and BAcc as a function of poisoning percentage.

4.1.3 Defense Results

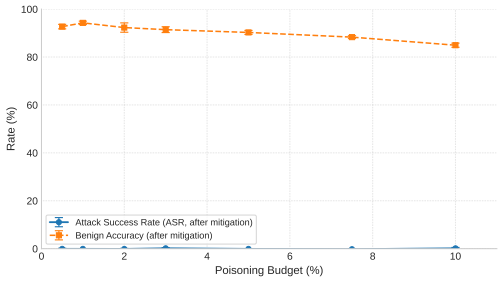
We now present the results obtained on the traffic classification model after applying our defense pipeline.

WB Setting

Assuming full WB access to the model parameters, we applied our purification and neuron pruning defense algorithms. Figure 4.5a illustrates the outcome of the reverse-engineering stage, while Figure 4.5b reports the BAcc and ASR values after neuron pruning. From the reverse-engineering analysis, we observe that the reconstructed trigger is not effectively identified for poisoning budgets below 2%. However, as the poisoning budget increases, the reverse-engineered trigger becomes even more effective than the original one. This behavior arises because the trigger inversion algorithm can optimize across all available features, potentially identifying a more powerful trigger than the one intentionally injected. Unlike our original trigger, limited to PHR and MCS, the reverse-engineered one may involve multiple features simultaneously or even include values outside the original feature distribution. Indeed, in our setup, we did not constrain the optimization process to remain within the empirical data range, allowing the defense algorithm to produce an OOD trigger that maximizes activation effectiveness.



(a) REASR compared to original ASR.



(b) ASR and benign accuracy after neuron pruning.

Figure 4.5. Comparison between reverse-engineered and original triggers (left) and model performance after mitigation (right).

Regarding neuron pruning, the results show that the ASR is effectively nullified while the BAcc remains high and stable. This confirms that our pruning strategy successfully removes the backdoor neurons without degrading the model's clean performance, achieving an effective balance between robustness and accuracy preservation.

BB Setting

In the BB setting, the defender does not have access to model parameters or internal gradients, and must therefore rely on observable data and outputs. Distribution-based detectors such as the IF identify anomalies by measuring statistical or pointwise deviations from the training distribution. However, if an attacker embeds the

trigger within regions of the feature space that overlap with normal data, these detectors can be easily deceived.

In our case, the attacker perturbs only the PHR and MCS features while keeping all others unmodified. This selective poisoning sometimes introduces minor inconsistencies that the IF can detect, but many triggered samples still lie well within the overall data manifold (Fig. 4.6). As a result, purely distributional detectors may fail to identify the backdoor in a fully stealthy configuration.

Temporal detectors, such as our LSTM-autoencoder, operate on sequential data and exploit temporal correlations between features. Because temporal dynamics are harder to replicate realistically, these models are generally more robust against stealthy triggers (See Fig. 4.7). Nevertheless, robustness is not absolute: an adaptive adversary with full knowledge of the detector architecture and access to temporal data could still craft time-consistent poisoned samples. Therefore, the overall effectiveness depends on factors such as the chosen features, sequence window length, and retraining frequency.

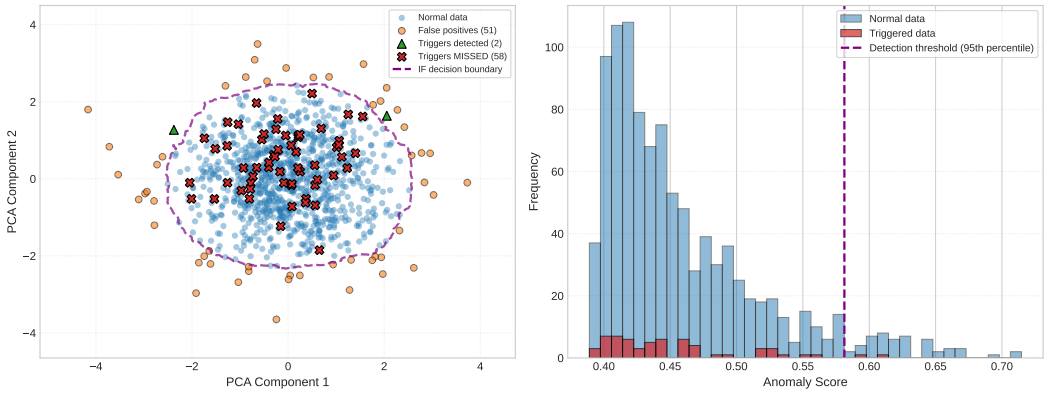


Figure 4.6. Visualization of triggered samples within the feature space. Left: PCA projection showing that many triggered samples (in red) fall inside the distribution of clean data, making them hard to detect via density-based methods. Right: anomaly score histogram from the IF, highlighting partial separation between normal and poisoned samples.

4.2 Traffic Forecaster

We implemented an LSTM-based model that forecasts the TH over a future horizon H. The forecaster operates as a classifier: the continuous TH range is discretized into k classes (we experimented with several values of k). The output class is then provided to a deterministic controller C, which uses a simple heuristic to determine the number of PRBs to allocate to each slice. This control command is enforced on the RAN through the interface, as shown in Figure 4.8. The physical testbed that implements this loop (rack, USRP and RF multiplexer) is shown in Figure 4.9, while the end-to-end deployment that maps the Near-RT RIC, OAI gNB, and UE instances to physical nodes is reported in Figure 4.10. Together, these panels provide both the logical architecture and the experimental mapping used for the forecasting

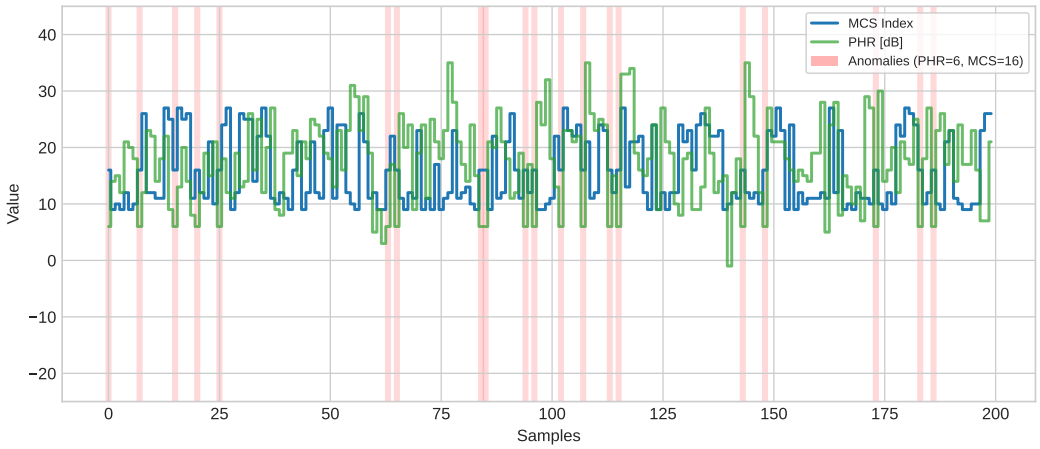


Figure 4.7. Temporal AD on MCS and PHR TS. Red vertical bars denote segments flagged as anomalous by the LSTM-autoencoder. While minor variations are tolerated, clear deviations in the joint evolution of MCS and PHR are successfully detected.

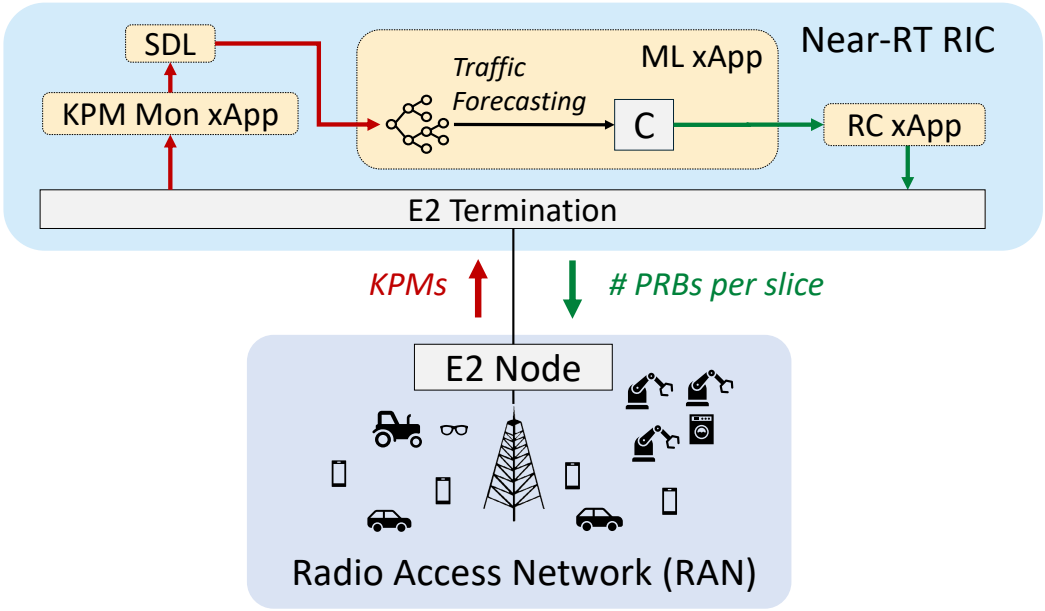


Figure 4.8. Closed-loop interaction between the Near-RT RIC and the RAN. The KPM Monitoring xApp collects real-time KPMs and provides them to the ML-based Traffic Forecasting xApp. The deterministic controller (C) applies a heuristic to translate predicted TH into PRB allocation ratios, which are then enforced by the RAN Control (RC) xApp through E2 messages toward the RAN.

and control experiments. The per-slice resource policy format used by the RC xApp is illustrated in Figure 4.11; this policy is the input to the controller that translates predicted throughput into PRB shares.

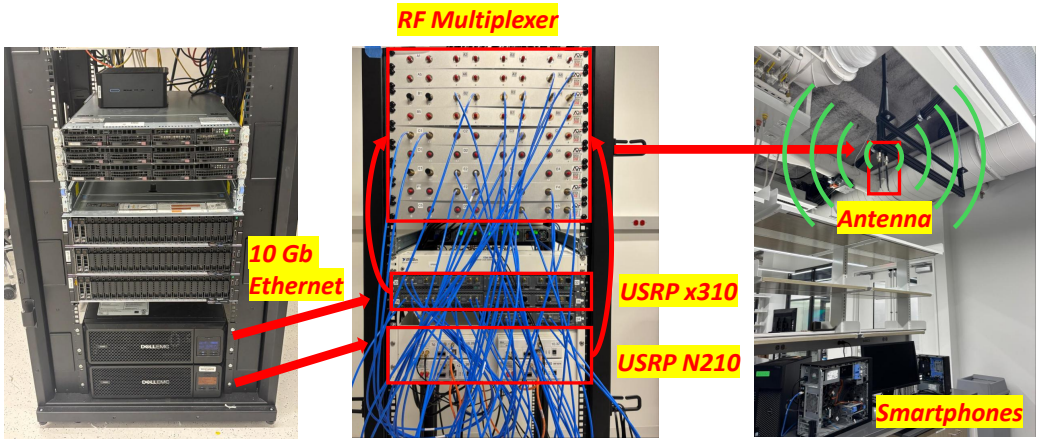


Figure 4.9. Physical OTA setup. USRPs are connected to an RF multiplexer, which routes signals to the antenna. The antenna propagates transmissions OTA to both COTS and OAI UEs. All radios are controlled by servers connected via 10 Gb Ethernet links.

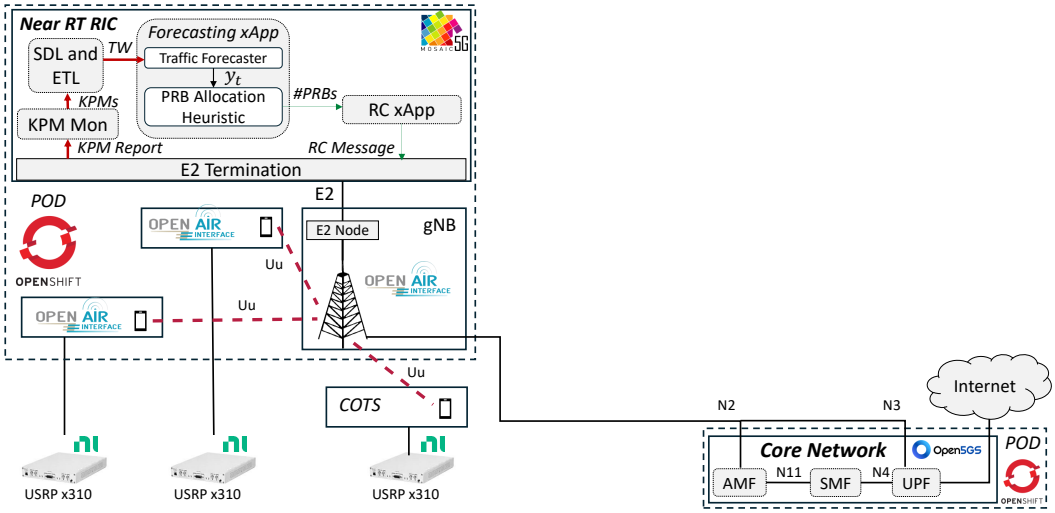


Figure 4.10. End-to-end OTA experimental setup for traffic forecasting. The Near-RT RIC hosts the ML xApp and the controller, while the gNB runs on OAI with USRP hardware. KPMs are periodically collected and used by the xApp to predict TH and adjust per-slice PRB allocations in real time.

Empirically, we found that the following heuristic performs effectively:

$$\#\text{PRBs} = \frac{\text{Pred Bits}}{\text{Past SE}} = \frac{K(y_t)}{\frac{\text{Bits}_{\text{past}}}{\text{PRBs}_{\text{past}}}} \cdot \frac{1}{2000},$$

where the factor $1/2000$ accounts for the 2000 slots per second under numerology $\mu = 1$ (slot duration of 0.5 ms). The COTS UE was assigned to slice 2, and the OAI-based UEs to slice 1. To receive KPMs for inference, we employed the FlexRIC KPM Monitoring xApp, using only MAC-layer metrics. For control enforcement, we extended FlexRIC and OAI by building on top of the *O-RAN Slice* framework [143],

implementing per-slice PRB allocation in the feedback loop. This framework could be extended to support per-UE PRB allocation and dynamic UE-slice migration. We implemented the RIC Indication procedure for the Slice SM, which was originally a stub in the FlexRIC code. To ensure O-RAN-compliance, we followed the O-RAN Alliance specification [144] for per-slice PRB allocation parameters. Figure 4.12 shows the effect of the attack on PRB allocation for slice 1 (the slice serving the OAI UEs) in our live Near-RT O-RAN experiment. The left panel reports the vulnerable forecaster, where trigger activations (red shaded bands) induce abrupt PRB reductions; the right panel reports the purified forecaster, where identical trigger inputs fail to reduce PRBs, demonstrating the effectiveness of our purification procedure.

RRM Policy Member					
PLMN ID	*S-NSSAI*		*Min PRB Ratio*	*Max PRB Ratio*	*Dedicated PRB Ratio*
	SST	*SD*			

Figure 4.11. Example of RRM policy configuration. Each policy entry includes the Public Land Mobile Network (PLMN) identifier, the Single - Network Slice Selection Assistance Information (S-NSSAI) describing the target slice, and the slice type (Slice-Service-Type (SST), e.g., eMBB=1). The policy defines the minimum, maximum, and dedicated shares of PRBs to be allocated, expressed as percentage ratios of the total available resources.

We required RC SM support within a Python xApp[1]. However, since the existing SWIG interface only supported KPM monitoring SMs, we extended it to include the FlexRIC C-core functionalities for the RC service model. In addition, we modified parts of the FlexRIC build chain to integrate these changes seamlessly.

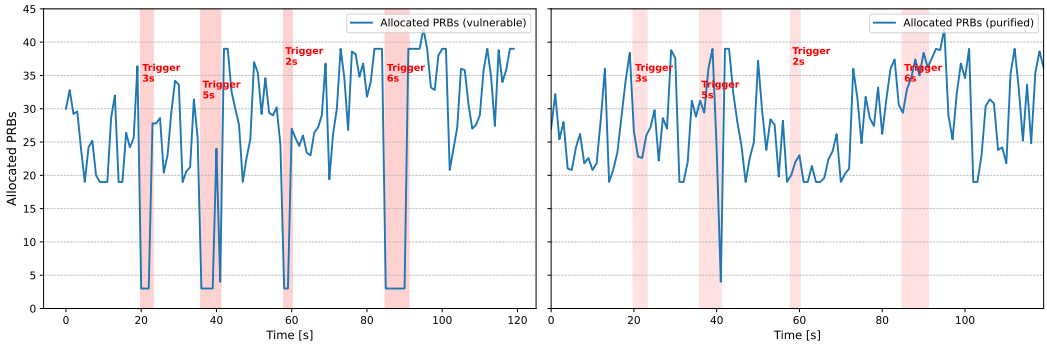


Figure 4.12. Comparison between vulnerable and purified traffic forecaster behavior. Red shaded areas mark trigger activations. Left: in the vulnerable model, the trigger causes abrupt PRB reductions (attack activation). Right: after purification, the same trigger fails to activate the backdoor, and PRB allocations remain at normal levels.

[1] An alternative would have been to deploy two cooperative xApps connected via an SCTP link.

Chapter 5

Conclusion and Future Work

This thesis investigated the feasibility, stealthiness, detection and mitigation of BA against ML-based xApps in O-RAN closed-loop deployments. We developed the first framework for defending from BA in O-RAN, and a wireless research platform that we pledge to release publicly to foster research in this field. The experimental work combined *Colosseum*/OTA testbeds, traffic generators and USRP with ETL pipelines, development and AI/ML WF. On top of this experimental base, we implemented and assessed a set of defenses ranging from reverse-engineering and neuron purification/pruning to distributional and temporal anomaly detectors. Moreover, we provided a systematic methodology for designing stealthy triggers in O-RAN KPMs and comparative evidence that sequence-aware detectors outperform pointwise distributional detectors when defending against stealthy, in-distribution triggers. We found empirically a stealth-learnability trade-off: triggers must balance rarity and learnability, since values that are too common cause spurious activation on clean data while values that are too rare are easily detectable. Moreover, we contributed to open-source projects such as *FlexRIC* and *OAI* extending xApps and providing RC implementation.

Stochasticity and non-stationarity in RAN telemetry create fundamental obstacles for both attack and defense. Because gNBs aggregate per-UE metrics into cell-level KPMs, a single UE's anomalous behavior can be diluted by other UEs and by scheduler, retransmission and mobility noise; therefore UE-level triggers must overcome aggregation dilution and be robust to channel and scheduling stochasticity to reliably affect slice-level statistics. Practically, creating such triggers requires controlling application-level traffic, timing and (possibly) multiple colluding UEs, while remaining covert and evading conventional anomaly filters. A second direction concerns more sophisticated, stealthy attacks and channel-aware triggers. Semantic feature attacks, where triggers are embedded in naturally occurring traffic patterns, are harder to detect than externally injected patterns. Channel effects matter: if a compromised UE sends $x(t)$ and the gNB observes $y(t) = h(t) \cdot x(t) + n(t)$, fading, multipath and noise may distort the trigger and reduce recognizability; robustness may require integrating channel estimation/compensation into trigger design or defenses. Defensive research must therefore move from simple detection toward a detect–explain–predict framework for distribution shifts in O-RAN: detect that a shift occurred, explain which parameters and features changed (in a domain-specific

space $\Omega_{\text{O-RAN}}$), predict likely future transformations $T(x)$ and, when appropriate, apply corrective mappings $T^{-1}(x)$ or mitigation actions. A central open problem is deciding the malignancy of a shift, whether it is benign operational change or an adversarial manipulation, especially when offline retraining or online FT can themselves be abused. Existing heuristics are limited and largely ad hoc; clustering and attribution techniques that correlate UE-level and aggregated signals look promising for this decision task. Finally, future work should develop AD modules that are robust to distributional shifts and able to distinguish adversarially induced shifts from physiological ones. This includes studying distributionally robust learning, stability and flatness in O-RAN.

Appendix A

Appendix

A.1 Metrics

This section summarizes the main metrics used in our experiments.

Identifiers

International Mobile Subscriber Identity (IMSI), RNTI, Slice ID, and PLMN identify users and networks but are not part of the training set. For example, Verizon's PLMN ID is Mobile Country Code (MCC) 310 + Mobile Network Code (MNC) 10. The IMSI is stored on the Universal Integrated Circuit Card (UICC), whose main component is the SIM card.

MCS

The MCS defines transmission efficiency based on modulation and coding [142]. Higher modulation orders (e.g., 256QAM) increase TH but require higher Signal-to-Noise Ratio (SNR), while coding introduces redundancy for error protection. The MCS is selected dynamically according to SNR, Channel Quality Indicator (CQI), Block Error Rate (BLER), and load.

MCS Index	Modulation	Example Use Case
0–9	QPSK	Weak signal, low SNR
10–20	16QAM	Moderate link
21–27	64QAM	Strong link
28+	256QAM	Excellent SNR (high-speed scenarios)

Table A.1. Modulation and coding scheme (MCS) levels and their typical use cases.

CQI

The CQI is reported by the UE to the gNB to select the best MCS given current channel conditions.

Received Signal Strength Indicator (RSSI), SNR, and PHR

RSSI and SNR capture received signal quality, while the PHR indicates the UE's remaining UL transmission power margin. Precisely, PHR is calculated as:

$$\text{PHR} = \text{P}_{\text{max,UE}} - \text{P}_{\text{PUSCH}} = \text{P}_{\text{max,UE}} - 10\log_{10}(M_{\text{PUSCH}}) + P_0 + \alpha \cdot PL + \Delta_{\text{TF}} + f(\delta_{\text{TPC}})$$

Where $P_{\text{max,UE}}$ is the UE's powre capability and P_{PUSCH} is the current transmit power. PHR is limited by the power control parameters (P_0, α and Δ_{TF}) and hardware, and when reported to gNBs is clipped and quantized. More precisely M_{PUSCH} is number of assigned Resource Block (RB), P_0 is base power offset, α is path loss compensation factor, PL is estimated path loss, Δ_{TF} is MCS-dependent adjustment and δ_{TPC} is Transmit Power Control (TPC) command from gNB.

PRB Metrics

Requested and granted PRB measure the difference between bandwidth demand and actual allocation.

Transport Block (TB) Metrics

The TBS is the payload size transmitted per TB, the fundamental unit exchanged between MAC and PHY.

BLER

The BLER is the ratio of failed to transmitted TBs:

$$\text{BLER} = \frac{\text{Number of failed TBs}}{\text{Total Number of TBs sent}}. \tag{A.1}$$

Reliable links achieve BLER < 0.05, while 3GPP targets ≈ 0.1.

Hybrid Automatic Repeat Request (HARQ)

HARQ combines retransmissions with forward error correction. Each TB is crc-checked, acknowledged if correct, or retransmitted otherwise. Multiple HARQ processes per UE allow parallelism.

Timing Metrics

5G structures time into frames (10 ms), subframes (1 ms), and slots whose length depends on numerology μ (Table A.2).

A.2 5G Protocol Stack in details

The 5G stack (Figure A.3) can be understood along two orthogonal axes. The *scope* axis separates protocols that terminate in the *radio access* (AS) from those that terminate in the *core network* (NAS). Second, the *function* axis separates CP signaling from User Plane (UP) data transport. Together, these axes form a grid that shows where a protocol lives (AS/NAS) and what it does (CP/UP).

μ	Subcarrier Spacing (SCS)	Slot Length	Slots per 1 ms
0	15 kHz	1 ms	1
1	30 kHz	0.5 ms	2
2	60 kHz	0.25 ms	4
3	120 kHz	0.125 ms	8

Table A.2. 5G NR numerology: subcarrier spacing, slot duration, and slots per subframe.

Protocol	Description
NAS	End-to-end control protocol between UE and AMF. Handles authentication, security, idle-mode procedures, IP address assignment, paging, mobility and session management.
RRC	Control between UE and gNB. Broadcasts system information, manages connections and bearers, mobility, and configures lower layers.
SDAP	Introduced in 5G NR for the user plane. Maps QoS flows to data radio bearers and marks QoS Flow Identifiers (QFI).
PDCP	Provides IP header compression (ROHC), ciphering, integrity protection, retransmissions, in-sequence delivery (optional), and duplicate removal. Supports routing/duplication in dual connectivity.
RLC	Performs segmentation and reassembly of PDCP PDUs, supports Transparent (TM), Unacknowledged (UM), and Acknowledged (AM) modes, with retransmission handling and duplicate removal.
MAC	Multiplexes logical channels onto transport blocks, manages Hybrid-ARQ retransmissions, and handles scheduling functions.
PHY	Implements channel coding/decoding, modulation/demodulation, and multi-antenna mapping. Transmits/receives data over the air interface.

Table A.3. Main protocols of the 5G stack and their functionalities.

AS vs NAS — *scope.* AS comprises all radio procedures and layers that operate between the UE and the gNB: PHY, MAC, RLC, PDCP, SDAP (for user data), and RRC (for signaling). NAS sits above RRC as an end-to-end CP protocol between UE and AMF, responsible for registration, authentication, mobility and session management, UE IP allocation, and slice/DNN selection. NAS messages are *carried inside* RRC over the radio link and pass transparently through the RAN.

CP vs. UP — *function.* CP protocols handle signaling to set up, configure, secure, and manage connections and sessions (e.g., RRC, NAS, NG-AP, Packet Forwarding Control Protocol (PFCP)). UP protocols carry user traffic (e.g., SDAP, PDCP, RLC, MAC, PHY in the RAN; GPRS Tunneling Protocol - User Plane (GTP-U) in the core). This separation enables independent scaling, policy enforcement, and observability for signaling and data paths.

Acronyms

1D CNN One Dimensional Convolutional Neural Networks.

1G 1st Generation.

2G 2nd Generation.

3G 3rd Generation.

3GPP 3rd Generation Partnership Project.

3GPP-NSMN 3GPP - Network Slicing Management System.

4G 4th Generation.

5G 5th Generation.

6G 6th Generation.

AA Adversarial Attack.

ABS Artificial Brain Stimulation.

AD Anomaly Detection.

AI Artificial Intelligence.

AMF Access and Mobility Management Function.

AML Adversarial Machine Learning.

AN Access Network.

API Application Programming Interface.

AS Access Stratum.

ASIO Asynchronous Input/Output.

ASN.1 Abstract Syntax Notation One.

ASR Attack Success Rate.

BA Backdoor Attack.

BAcc Benign Accuracy.

BB Black Box.

BBSD Black-Box Shift Detection.

BBU Base Band Unit.

BLER Block Error Rate.

BTS Base Transceiver Station.

C-RAN Cloud Radio Access Network.

CDMA Code Division Multiple Access.

CF Catastrophic Forgetting.

CIA Confidentiality, Integrity, Availability.

CN Core Network.

COTS Commercial Off The Shelf.

CP Control Plane.

CQI Channel Quality Indicator.

CSV Comma Separated Value.

CU Centralized Unit.

CV Computer Vision.

D-RAN Distributed Radio Access Network.

DevOps Development and Operations.

DL Downlink.

DNN Deep Neural Network.

DP Data Poisoning.

DRO Distributionally Robust Optimization.

DU Distributed Unit.

E2AP E2 Application Protocol.

E2E End-to-End.

E2SM E2 Service Model.

EA Evasion Attack.

ECD Expected Conditional Distance.

EDA Exploratory Data Analysis.

EMA Exponential Moving Average.

eMBB enhanced Mobile Broadband.

eNB eNodeB.

EPC Evolved Packet Core.

ERM Empirical Risk Minimization.

ETL Extract, Transform, Load.

ETSI European Telecommunications Standards Institute.

FDMA Frequency Division Multiple Access.

FFTx Fiber to the x.

FGSM Fast Gradient Sign Method.

FIR Finite Impulse Response.

FPGA Field Programmable Gate Array.

FPR False Positive Rate.

FT Fine-Tuning.

GB Grey Box.

GMM Gaussian Mixture Models.

gNB gNodeB.

GPRS General Packet Radio Service.

GTP GPRS Tunneling Protocol.

GTP-U GPRS Tunneling Protocol - User Plane.

HARQ Hybrid Automatic Repeat Request.

ID In-Distribution.

IDS Intrusion Detection System.

IF Isolation Forest.

IID Independent and Identically Distributed.

IMSI International Mobile Subscriber Identity.

IP Internet Protocol.

IQ In-phase and Quadrature.

k-NN k-Nearest Neighbors.

KL Kullback-Leibler.

KPI Key Performance Indicator.

KPM Key Performance Measurement.

L1 Layer 1.

L2 Layer 2.

L3 Layer 3.

LAN Local Area Network.

LSTM Long Short-Term Memory.

LTE Long Term Evolution.

M&O Management and Orchestration.

MAC Medium Access Control.

MCC Mobile Country Code.

MCHEM Massive Channel Emulator.

MCS Modulation and Coding Scheme.

MFs Management Functions.

MGEN Multi-Generator Network Test Tool.

MIMO Multiple Input Multiple Output.

ML Machine Learning.

MLP Multi Layer Perceptron.

MMD Maximum Mean Discrepancy.

mMTC massive Machine-Type Communications.

MNC Mobile Network Code.

MNO Mobile Network Operator.

MNOs Mobile Networks Operators.

MOE Mean Opinion Score.

NAS Non-Access Stratum.

NC Neural Cleanse.

Near-RT Near-Real-Time.

Near-RT RIC Near-Real-Time RAN Intelligent Controller.

NFs Network Funtions.

NFV-MANO Network Function Virtualization - Management and Orchestration.

NGAP NG - Application Protocol.

NLP Natural Language Processing.

Non-RT Non-Real-Time.

Non-RT RIC Non-Real-Time RAN Intelligent Controller.

NR New Radio.

NSA Non Stand Alone.

O-CU O-RAN Central Unit.

O-DU O-RAN Distributed Unit.

O-RAN Open Radio Access Network.

O-RU O-RAN Radio Unit.

OAI OpenAirInterface.

OFDMA Orthogonal Frequency Division Multiple Access.

OOD Out-of-Distribution.

OT Optimal Transport.

OTA Over The Air.

PAN Personal Area Network.

PCA Principal Component Analysis.

PDCP Packet Data Convergence Protocol.

PFCP Packet Forwarding Control Protocol.

PHR Power Headroom Report.

PHY Physical.

PLMN Public Land Mobile Network.

PRB Physical Resource Block.

PSTN Public Switched Telephone Network.

QOE Quality of Experience.

QoS Quality of Service.

RAN Radio Access Network.

RB Resource Block.

RC RAN Control.

REASR attack success rate of reverse engineered trojan triggers.

RF Radio Frequency.

RIC RAN Intelligent Controller.

RKHS Reproducing Kernel Hilbert Space.

RLC Radio Link Control.

RNTI Radio Network Temporary Identifier.

RRC Radio Resource Control.

RSSI Received Signal Strength Indicator.

RU Radio Unit.

S-NSSAI Single - Network Slice Selection Assistance Information.

SA Stand Alone.

SBOM Software Bill of Materials.

SCAS Security Assurance Specification.

SCS Subcarrier Spacing.

SCTP Stream Control Transmission Protocol.

SD-RAN Software-Defined Radio Access Network.

SDAP Service Data Adaptation Protocol.

SDK Software Development Kit.

SDL Shared Data Layer.

SDO Standards Development Organizations.

SDR Software Defined Radio.

SIM Subscriber Identity Module.

SLA Service Level Agreement.

SM Service Model.

SMF Session Management Function.

SMO Service Management and Orchestration.

SMS Short Message Service.

SNR Signal-to-Noise Ratio.

SotA State of the Art.

SPECTRE Spectral Poison Excision Through Robust Estimation.

SRN Standard Radio Node.

SST Slice-Service-Type.

STRIP STRong Intentional Perturbation.

SWIG Simplified Wrapper and Interface Generator.

TB Transport Block.

TBS Transport Block Size.

TCP Transmission Control Protocol.

TDMA Time Division Multiple Access.

TGEN Traffic Generator.

TH Throughput.

TL Transfer Learning.

TRACTOR Traffic Analysis and Classification Tool for Open RAN.

TS Time Series.

TSC Time Series Classification.

UDP User Datagram Protocol.

UE User Equipment.

UICC Universal Integrated Circuit Card.

UL Uplink.

UML Unified Modeling Language.

UP User Plane.

UPF User Plane Function.

URLLC Ultra-Reliable Low-Latency Communications.

USRP Universal Software Radio Peripheral.

V-RAN Virtualized Radio Access Network.

VoIP Voice over IP.

VoLTE Voice over LTE.

WAN Wide Area Network.

WB White Box.

WF Workflow.

WG Working Group.

Wi-Fi IEEE 802.11.

X2AP X2 Application Protocol.

xDSL Digital Subscriber Line.

ZT Zero Trust.

ZTA Zero Trust Architecture.

Bibliography

[1] M. . Company, "Open-source technology in the age of ai," 2024. [Online]. Available: https://www.mckinsey.com/capabilities/quantumblack/our-insights/open-source-technology-in-the-age-of-ai

[2] J. S. Research, "Data scientists targeted by malicious hugging face ml models with silent backdoor," 2024. [Online]. Available: https://jfrog.com/blog/data-scientists-targeted-by-malicious-hugging-face-ml-models-with-silent-backdoor/

[3] D. Tse and P. Viswanath, *Fundamentals of wireless communication*. Cambridge university press, 2005.

[4] J. Kurose and K. Ross, "Computer networks: A top down approach featuring the internet," 2010.

[5] M. Polese, L. Bonati, S. D'oro, S. Basagni, and T. Melodia, "Understanding o-ran: Architecture, interfaces, algorithms, security, and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 2, pp. 1376–1411, 2023.

[6] I. C. Wong, A. Chopra, S. Rajagopal, and R. Jana, *Open RAN: The Definitive Guide*. John Wiley & Sons, 2023.

[7] 3rd Generation Partnership Project (3GPP), "TS 38.401 V17.4.0: NR; NG-RAN; Architecture description," 3rd Generation Partnership Project (3GPP), Tech. Rep. TS 38.401, 2023, release 17. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/138400_138499/138401/17.04.00_60/ts_138401v170400p.pdf

[8] N. D. Tripathi and V. K. Shah, *Fundamentals of O-RAN*. John Wiley & Sons, 2025.

[9] O-RAN ALLIANCE WG3, "O-RAN.WG3.TS.E2SM-R004-v07.00: O-RAN E2 Service Model (E2SM)," O-RAN ALLIANCE e.V., Alfter, Germany, Tech. Rep. O-RAN.WG3.TS.E2SM-R004-v07.00, 2025, technical Specification, Version 7.0. [Online]. Available: https://www.o-ran.org/specifications/

[10] M. A. Habibi, B. Han, M. Saimler, I. L. Pavon, and H. D. Schotten, "Towards an ai/ml-driven smo framework in o-ran: Scenarios, solutions, and challenges," *arXiv preprint arXiv:2409.05092*, 2024.

[11] A. Shabbir, H. U. Manzoor, R. A. Ahmed, and Z. Halim, "Resilience of federated learning against false data injection attacks in energy forecasting," in *2024 International Conference on Green Energy, Computing and Sustainable Technology (GECOST)*. IEEE, 2024, pp. 245–249.

[12] O-RAN Alliance, "Zero Trust Architecture (ZTA) for Secure O-RAN," O-RAN Alliance, White Paper, 2024. [Online]. Available: https://mediastorage.o-ran.org/white-papers/O-RAN.WG11.ZTA%20for%20Secure%20O-RAN%20White%20Paper-2024-05.pdf

[13] V. Stafford, "Zero trust architecture," *NIST special publication*, vol. 800, no. 207, pp. 800–207, 2020.

[14] National Telecommunications and Information Administration (NTIA), "Open RAN Security Report," U.S. Department of Commerce, National Telecommunications and Information Administration, Report, 2024. [Online]. Available: https://www.ntia.gov/sites/default/files/publications/open_ran_security_report_full_report_0.pdf

[15] S. Kulinski and D. I. Inouye, "Towards explaining distribution shifts," in *International Conference on Machine Learning*. PMLR, 2023, pp. 17 931–17 952.

[16] G. I. Webb, L. K. Lee, B. Goethals, and F. Petitjean, "Analyzing concept drift and shift from sample data," *Data Mining and Knowledge Discovery*, vol. 32, no. 5, pp. 1179–1199, 2018.

[17] J. Quiñonero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, *Dataset shift in machine learning*. Mit Press, 2022.

[18] A. Storkey *et al.*, "When training and test sets are different: characterizing learning transfer," *Dataset shift in machine learning*, vol. 30, no. 3-28, p. 6, 2009.

[19] S. Kulinski, S. Bagchi, and D. I. Inouye, "Feature shift detection: Localizing which features have shifted via conditional distribution tests," *Advances in neural information processing systems*, vol. 33, pp. 19 523–19 533, 2020.

[20] P. W. Koh, S. Sagawa, H. Marklund, S. M. Xie, M. Zhang, A. Balsubramani, W. Hu, M. Yasunaga, R. L. Phillips, I. Gao *et al.*, "Wilds: A benchmark of in-the-wild distribution shifts," in *International conference on machine learning*. PMLR, 2021, pp. 5637–5664.

[21] S. Rabanser, S. Günnemann, and Z. Lipton, "Failing loudly: An empirical study of methods for detecting dataset shift," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[22] A. Ramdas, S. J. Reddi, B. Póczos, A. Singh, and L. Wasserman, "On the decreasing power of kernel and distance based nonparametric hypothesis tests in high dimensions," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, no. 1, 2015.

[23] Z. Lipton, Y.-X. Wang, and A. Smola, "Detecting and correcting for label shift with black box predictors," in *International conference on machine learning*. PMLR, 2018, pp. 3122–3130.

[24] M. Cuturi, "Sinkhorn distances: Lightspeed computation of optimal transport," *Advances in neural information processing systems*, vol. 26, 2013.

[25] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*. PMLR, 2017, pp. 214–223.

[26] L. C. Torres, L. M. Pereira, and M. H. Amini, "A survey on optimal transport for machine learning: Theory and applications," *arXiv preprint arXiv:2106.01963*, 2021.

[27] G. Peyré, M. Cuturi *et al.*, "Computational optimal transport: With applications to data science," *Foundations and Trends® in Machine Learning*, vol. 11, no. 5-6, pp. 355–607, 2019.

[28] N. Nikaein, M. K. Marina, S. Manickam, A. Dawson, R. Knopp, and C. Bonnet, "Openairinterface: A flexible platform for 5g research," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, pp. 33–38, 2014.

[29] F. Kaltenberger, A. P. Silva, A. Gosain, L. Wang, and T.-T. Nguyen, "Openairinterface: Democratizing innovation in the 5g era," *Computer Networks*, vol. 176, p. 107284, 2020.

[30] R. Schmidt, M. Irazabal, and N. Nikaein, "Flexric: An sdk for next-generation sd-rans," in *Proceedings of the 17th International Conference on emerging Networking EXperiments and Technologies*, 2021, pp. 411–425.

[31] D. Villa, M. Tehrani-Moayyed, C. P. Robinson, L. Bonati, P. Johari, M. Polese, and T. Melodia, "Colosseum as a digital twin: Bridging real-world experimentation and wireless network emulation," *IEEE Transactions on Mobile Computing*, vol. 23, no. 10, pp. 9150–9166, 2024.

[32] L. Bonati, P. Johari, M. Polese, S. D'Oro, S. Mohanti, M. Tehrani-Moayyed, D. Villa, S. Shrivastava, C. Tassie, K. Yoder *et al.*, "Colosseum: Large-scale wireless experimentation through hardware-in-the-loop network emulation," in *2021 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*. IEEE, 2021, pp. 105–113.

[33] M. Polese, L. Bonati, S. D'Oro, P. Johari, D. Villa, S. Velumani, R. Gangula, M. Tsampazi, C. P. Robinson, G. Gemmi *et al.*, "Colosseum: The open ran digital twin," *IEEE Open Journal of the Communications Society*, vol. 5, pp. 5452–5466, 2024.

[34] L. Bertizzolo, L. Bonati, E. Demirors, and T. Melodia, "Arena: A 64-antenna sdr-based ceiling grid testbed for sub-6 ghz radio spectrum research," in *Proceedings of the 13th International Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization*, 2019, pp. 5–12.

[35] R. A. Ferrús Ferré, O. Sallent Roig, and J. Pérez Romero, "Data analytics architectural framework for smarter radio resource management in 5g radio access networks," *IEEE communications magazine*, vol. 58, no. 5, pp. 98–104, 2020.

[36] L. Bonati, S. D'Oro, M. Polese, S. Basagni, and T. Melodia, "Intelligence and learning in o-ran for data-driven nextg cellular networks," *IEEE Communications Magazine*, vol. 59, no. 10, pp. 21–27, 2021.

[37] C. Clancy, J. Hecker, E. Stuntebeck, and T. O'Shea, "Applications of machine learning to cognitive radio networks," *IEEE Wireless Communications*, vol. 14, no. 4, pp. 47–52, 2007.

[38] B. Wang and K. R. Liu, "Advances in cognitive radio networks: A survey," *IEEE Journal of selected topics in signal processing*, vol. 5, no. 1, pp. 5–23, 2010.

[39] M. Bordin, A. Lacava, M. Polese, S. Satish, M. A. Nittoor, R. Sivaraj, F. Cuomo, and T. Melodia, "Design and evaluation of deep reinforcement learning for energy saving in open ran," in *2025 IEEE 22nd Consumer Communications & Networking Conference (CCNC)*. IEEE, 2025, pp. 1–6.

[40] G. O. Ferreira, C. Ravazzi, F. Dabbene, G. C. Calafiore, and M. Fiore, "Forecasting network traffic: A survey and tutorial with open-source comparative evaluation," *IEEE Access*, vol. 11, pp. 6018–6044, 2023.

[41] J. Groen, M. Belgiovine, U. Demir, B. Kim, and K. Chowdhury, "Tractor: Traffic analysis and classification tool for open ran," in *ICC 2024-IEEE International Conference on Communications*. IEEE, 2024, pp. 4894–4899.

[42] H. Lee, Y. Jang, J. Song, and H. Yeon, "O-ran ai/ml workflow implementation of personalized network optimization via reinforcement learning," in *2021 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2021, pp. 1–6.

[43] O-RAN ALLIANCE WG2, "O-RAN.AIML.WG2-v01.00: AI/ML Workflow Description and Requirements," O-RAN ALLIANCE e.V., Alfter, Germany, Tech. Rep. O-RAN.WG2.AIML-v01.00, 2020, technical Specification, Section 5.1 – Model Training and Deployment Principles. [Online]. Available: https://www.scribd.com/document/472204421/O-RAN-WG2-AIML-v01-00-pdf

[44] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "Colo-ran: Developing machine learning-based xapps for open ran closed-loop control on programmable experimental platforms," *IEEE Transactions on Mobile Computing*, vol. 22, no. 10, pp. 5787–5800, 2022.

[45] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A theory of learning from different domains," *Machine learning*, vol. 79, no. 1, pp. 151–175, 2010.

[46] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *International conference on machine learning*. PMLR, 2015, pp. 97–105.

[47] M. Long, J. Wang, and M. Jordan, "Deep transfer learning with joint adaptation networks. arxiv," *arXiv preprint arXiv:1605.06636*, 2016.

[48] H. Yu, J. Liu, X. Zhang, J. Wu, and P. Cui, "A survey on evaluation of out-of-distribution generalization," *arXiv preprint arXiv:2403.01874*, 2024.

[49] N. Ye, K. Li, H. Bai, R. Yu, L. Hong, F. Zhou, Z. Li, and J. Zhu, "Oodbench: Quantifying and understanding two dimensions of out-of-distribution generalization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 7947–7958.

[50] V. Vapnik, "Principles of risk minimization for learning theory," *Advances in neural information processing systems*, vol. 4, 1991.

[51] A. Torralba and A. A. Efros, "Unbiased look at dataset bias," in *CVPR 2011*. IEEE, 2011, pp. 1521–1528.

[52] D. C. Castro, I. Walker, and B. Glocker, "Causality matters in medical imaging," *Nature Communications*, vol. 11, no. 1, p. 3673, 2020.

[53] T. Wang, J. Liu, P. Cui, and H. Namkoong, "Rethinking distribution shifts: Empirical analysis and inductive modeling for tabular data," *arXiv e-prints*, pp. arXiv–2307, 2023.

[54] K. Budhathoki, D. Janzing, P. Bloebaum, and H. Ng, "Why did the distribution change?" in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021, pp. 1666–1674.

[55] X. Zhang, Y. He, R. Xu, H. Yu, Z. Shen, and P. Cui, "Nico++: Towards better benchmarking for domain generalization," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 16 036–16 047.

[56] Y. Yang, H. Zhang, D. Katabi, and M. Ghassemi, "Change is hard: A closer look at subpopulation shift," *arXiv preprint arXiv:2302.12254*, 2023.

[57] H. Rahimian and S. Mehrotra, "Distributionally robust optimization: A review," *arXiv preprint arXiv:1908.05659*, 2019.

[58] M. Li, H. Namkoong, and S. Xia, "Evaluating model performance under worst-case subpopulations," *Advances in Neural Information Processing Systems*, vol. 34, pp. 17 325–17 334, 2021.

[59] S. Gupta and D. Rothenhäusler, "The s-value: evaluating stability with respect to distributional shifts," *Advances in Neural Information Processing Systems*, vol. 36, pp. 72 058–72 070, 2023.

[60] H. Namkoong, Y. Ma, and P. W. Glynn, "Minimax optimal estimation of stability under distribution shift," *Operations Research*, 2025.

[61] X. Zhang, R. Xu, H. Yu, Y. Dong, P. Tian, and P. Cui, "Flatness-aware minimization for domain generalization," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 5189–5202.

[62] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. D. Tygar, "Adversarial machine learning," in *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, 2011, pp. 43–58.

[63] B. Kim, Y. E. Sagduyu, K. Davaslioglu, T. Erpek, and S. Ulukus, "Channel-aware adversarial attacks against deep learning-based wireless signal classifiers," *IEEE Transactions on Wireless Communications*, vol. 21, no. 6, pp. 3868–3880, 2021.

[64] V.-T. Hoang, Y. A. Ergu, V.-L. Nguyen, and R.-G. Chang, "Security risks and countermeasures of adversarial attacks on ai-driven applications in 6g networks: A survey," *Journal of Network and Computer Applications*, vol. 232, p. 104031, 2024.

[65] A. Chiejina, B. Kim, K. Chowhdury, and V. K. Shah, "System-level analysis of adversarial attacks and defenses on intelligence in o-ran based cellular networks," in *Proceedings of the 17th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2024, pp. 237–247.

[66] E. Aizikovich, D. Mimran, E. Grolman, Y. Elovici, and A. Shabtai, "Rogue cell: Adversarial attack and defense in untrusted o-ran setup exploiting the traffic steering xapp," *arXiv preprint arXiv:2505.01816*, 2025.

[67] H. Alimohammadi, S. Chatzimiltis, S. Mayhoub, M. Shojafar, S. A. Soleymani, A. Akbas, and C. H. Foh, "Kpi poisoning: An attack in open ran near real-time control loop," in *2024 IEEE Future Networks World Forum (FNWF)*. IEEE, 2024, pp. 712–718.

[68] N. N. Sapavath, B. Kim, K. Chowdhury, and V. K. Shah, "Experimental study of adversarial attacks on ml-based xapps in o-ran," in *GLOBECOM 2023-2023 IEEE Global Communications Conference*. IEEE, 2023, pp. 6352–6357.

[69] E. Habler, R. Bitton, D. Avraham, E. Klevansky, D. Mimran, O. Brodt, H. Lehmann, Y. Elovici, and A. Shabtai, "Adversarial machine learning threat analysis and remediation in open radio access network (o-ran)," *Journal of Network and Computer Applications*, vol. 236, p. 104090, 2025.

[70] J. Groen, S. D'Oro, U. Demir, L. Bonati, M. Polese, T. Melodia, and K. Chowdhury, "Implementing and evaluating security in o-ran: Interfaces, intelligence, and platforms," *IEEE Network*, 2024.

[71] C.-F. Hung, Y.-R. Chen, C.-H. Tseng, and S.-M. Cheng, "Security threats to xapps access control and e2 interface in o-ran," *IEEE Open Journal of the Communications Society*, vol. 5, pp. 1197–1203, 2024.

[72] P. K. Kakani, H. Djuitcheu, and H. D. Schotten, "Evaluation of xapps and their security in next-generation open radio access networks (oran)," *Authorea Preprints*, 2024.

[73] K. Chen, Y. Meng, X. Sun, S. Guo, T. Zhang, J. Li, and C. Fan, "Badpre: Task-agnostic backdoor attacks to pre-trained nlp foundation models," *arXiv preprint arXiv:2110.02467*, 2021.

[74] Y. Yao, H. Li, H. Zheng, and B. Y. Zhao, "Latent backdoor attacks on deep neural networks," in *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, 2019, pp. 2041–2055.

[75] S. Wang, S. Nepal, C. Rudolph, M. Grobler, S. Chen, and T. Chen, "Backdoor attacks against transfer learning with pre-trained deep learning models," *IEEE Transactions on Services Computing*, vol. 15, no. 3, pp. 1526–1539, 2020.

[76] K. Roth, Y. Kilcher, and T. Hofmann, "The odds are odd: A statistical test for detecting adversarial examples," in *International Conference on Machine Learning*. PMLR, 2019, pp. 5498–5507.

[77] A. Lacava, S. Maxenti, L. Bonati, S. D'Oro, A. Oprea, T. Melodia, and F. Restuccia, "How to poison an xapp: Dissecting backdoor attacks to deep reinforcement learning in open radio access networks," *Computer Networks*, p. 111727, 2025.

[78] P. Kiourti, K. Wardega, S. Jha, and W. Li, "Trojdrl: Trojan attacks on deep reinforcement learning agents," *arXiv preprint arXiv:1903.06638*, 2019.

[79] I. Ilahi, M. Usama, J. Qadir, M. U. Janjua, A. Al-Fuqaha, D. T. Hoang, and D. Niyato, "Challenges and countermeasures for adversarial attacks on deep reinforcement learning," *IEEE Transactions on Artificial Intelligence*, vol. 3, no. 2, pp. 90–109, 2021.

[80] Y. Li, Y. Jiang, Z. Li, and S.-T. Xia, "Backdoor learning: A survey," *IEEE transactions on neural networks and learning systems*, vol. 35, no. 1, pp. 5–22, 2022.

[81] Y. Gao, B. G. Doan, Z. Zhang, S. Ma, J. Zhang, A. Fu, S. Nepal, and H. Kim, "Backdoor attacks and countermeasures on deep learning: A comprehensive review," *arXiv preprint arXiv:2007.10760*, 2020.

[82] A. Vassilev, A. Oprea, A. Fordyce, and H. Andersen, "Adversarial machine learning: A taxonomy and terminology of attacks and mitigations," 2024.

[83] R. Bitton, D. Avraham, E. Klevansky, D. Mimran, O. Brodt, H. Lehmann, Y. Elovici, and A. Shabtai, "Adversarial machine learning threat analysis in open radio access networks." 2022.

[84] Y. A. Ergu, V.-L. Nguyen, R.-H. Hwang, Y.-D. Lin, C.-Y. Cho, and H.-K. Yang, "Unmasking vulnerabilities: Adversarial attacks against drl-based resource allocation in o-ran," in *ICC 2024-IEEE International Conference on Communications*. IEEE, 2024, pp. 2378–2383.

[85] Y. Wang, D. Xue, S. Zhang, and S. Qian, "Badagent: Inserting and activating backdoor attacks in llm agents," *arXiv preprint arXiv:2406.03007*, 2024.

[86] O-RAN ALLIANCE WG11, "Study on security for artificial intelligence and machine learning (ai/ml) in o-ran," O-RAN ALLIANCE e.V., Alfter, Germany, Tech. Rep. O-RAN.WG11.TR.AIML-Security-Analysis.0-R004-v03.00, 2025, technical Report.

[87] C.-H. Weng, Y.-T. Lee, and S.-H. B. Wu, "On the trade-off between adversarial and backdoor robustness," *Advances in Neural Information Processing Systems*, vol. 33, pp. 11 973–11 983, 2020.

[88] S. Hong, V. Chandrasekaran, Y. Kaya, T. Dumitraş, and N. Papernot, "On the effectiveness of mitigating data poisoning attacks with gradient shaping," *arXiv preprint arXiv:2002.11497*, 2020.

[89] K. Grosse, T. Lee, B. Biggio, Y. Park, M. Backes, and I. Molloy, "Backdoor smoothing: Demystifying backdoor attacks on deep neural networks," *Computers & Security*, vol. 120, p. 102814, 2022.

[90] J. Cohen, E. Rosenfeld, and Z. Kolter, "Certified adversarial robustness via randomized smoothing," in *international conference on machine learning*. PMLR, 2019, pp. 1310–1320.

[91] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *2019 IEEE symposium on security and privacy (SP)*. IEEE, 2019, pp. 707–723.

[92] V. Childress, J. Collyer, and J. Knapp, "Architectural backdoors in deep learning: A survey of vulnerabilities, detection, and defense," *arXiv preprint arXiv:2507.12919*, 2025.

[93] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction {APIs}," in *25th USENIX security symposium (USENIX Security 16)*, 2016, pp. 601–618.

[94] Y. He, Z. Shen, C. Xia, J. Hua, W. Tong, and S. Zhong, "Sgba: A stealthy scapegoat backdoor attack against deep neural networks," *Computers & Security*, vol. 136, p. 103523, 2024.

[95] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, "Badnets: Evaluating backdooring attacks on deep neural networks," *IEEE Access*, vol. 7, pp. 47 230–47 244, 2019.

[96] Y. Jiang, X. Ma, S. M. Erfani, and J. Bailey, "Backdoor attacks on time series: A generative approach," in *2023 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*. IEEE, 2023, pp. 392–403.

[97] R. Ning, C. Xin, and H. Wu, "Trojanflow: A neural backdoor attack to deep learning-based network traffic classifiers," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 2022, pp. 1429–1438.

[98] C. Dong, Z. Sun, G. Bai, S. Piao, W. Chen, and W. E. Zhang, "Trojantime: Backdoor attacks on time series classification," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2025, pp. 154–166.

[99] C. G. Dong, L. N. Zheng, W. Chen, W. E. Zhang, and L. Yue, "Swap: exploiting second-ranked logits for adversarial attacks on time series," in *2023 IEEE International Conference on Knowledge Graph (ICKG)*. IEEE, 2023, pp. 117–125.

[100] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," *arXiv preprint arXiv:1712.05526*, 2017.

[101] S. Li, M. Xue, B. Z. H. Zhao, H. Zhu, and X. Zhang, "Invisible backdoor attacks on deep neural networks via steganography and regularization," *arXiv preprint arXiv:1909.02742*, 2019.

[102] C. Zhou, Y.-G. Wang, Z.-J. Wang, and X. Kang, "Lp-norm distortion-efficient adversarial attack," *Signal Processing: Image Communication*, vol. 131, p. 117241, 2025.

[103] L. Xu, X. Zheng, X. Li, Y. Zhang, L. Liu, and H. Ma, "Wicam: Imperceptible adversarial attack on deep learning based wifi sensing," in *2022 19th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE, 2022, pp. 10–18.

[104] K. Doan, Y. Lao, W. Zhao, and P. Li, "Lira: Learnable, imperceptible and robust backdoor attacks," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 11 966–11 976.

[105] K. Doan, Y. Lao, and P. Li, "Backdoor attack with imperceptible input and latent modification," *Advances in Neural Information Processing Systems*, vol. 34, pp. 18 944–18 957, 2021.

[106] J. Geiping, L. Fowl, W. R. Huang, W. Czaja, G. Taylor, M. Moeller, and T. Goldstein, "Witches' brew: Industrial scale data poisoning via gradient matching," *arXiv preprint arXiv:2009.02276*, 2020.

[107] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *International conference on artificial intelligence and statistics*. PMLR, 2020, pp. 2938–2948.

[108] T. A. Nguyen and A. Tran, "Input-aware dynamic backdoor attack," *Advances in Neural Information Processing Systems*, vol. 33, pp. 3454–3464, 2020.

[109] E. Chou, F. Tramer, and G. Pellegrino, "Sentinet: Detecting localized universal attacks against deep learning systems," in *2020 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2020, pp. 48–54.

[110] Y. Dong, X. Yang, Z. Deng, T. Pang, Z. Xiao, H. Su, and J. Zhu, "Black-box detection of backdoor attacks with limited information and data," in

Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 16 482–16 491.

[111] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, "Trojaning attack on neural networks," in *25th Annual Network And Distributed System Security Symposium (NDSS 2018)*. Internet Soc, 2018.

[112] L. Jin, X. Wen, W. Jiang, J. Zhan, and X. Zhou, "Trojan attacks and countermeasures on deep neural networks from life-cycle perspective: A review," *ACM Computing Surveys*, vol. 57, no. 10, pp. 1–37, 2025.

[113] C. Truong, L. Oudre, and N. Vayatis, "Selective review of offline change point detection methods," *Signal Processing*, vol. 167, p. 107299, 2020.

[114] T. Cao, J. Zhu, and G. Pang, "Anomaly detection under distribution shift," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 6511–6523.

[115] B. Yan, J. Lan, and Z. Yan, "Backdoor attacks against voice recognition systems: A survey," *ACM Computing Surveys*, vol. 57, no. 3, pp. 1–35, 2024.

[116] Y. Liu, W.-C. Lee, G. Tao, S. Ma, Y. Aafer, and X. Zhang, "Abs: Scanning neural networks for back-doors by artificial brain stimulation," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 1265–1282.

[117] D. Kokkonis, M. Marcozzi, E. Decoux, and S. Zacchiroli, "Rosa: Finding backdoors with fuzzing," *arXiv preprint arXiv:2505.08544*, 2025.

[118] Y. Zeng, W. Park, Z. M. Mao, and R. Jia, "Rethinking the backdoor attacks' triggers: A frequency perspective," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 16 473–16 481.

[119] D. Popovic, A. Sadeghi, T. Yu, S. Chawla, and I. Khalil, "Debackdoor: A deductive framework for detecting backdoor attacks on deep models with limited data," *arXiv preprint arXiv:2503.21305*, 2025.

[120] X. Xu, Q. Wang, H. Li, N. Borisov, C. A. Gunter, and B. Li, "Detecting ai trojans using meta neural analysis," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 103–120.

[121] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient smt solver for verifying deep neural networks," in *International conference on computer aided verification*. Springer, 2017, pp. 97–117.

[122] G. Katz, D. A. Huang, D. Ibeling, K. Julian, C. Lazarus, R. Lim, P. Shah, S. Thakoor, H. Wu, A. Zeljić *et al.*, "The marabou framework for verification and analysis of deep neural networks," in *International conference on computer aided verification*. Springer, 2019, pp. 443–452.

[123] Y. Zeng, S. Chen, W. Park, Z. M. Mao, M. Jin, and R. Jia, "Adversarial unlearning of backdoors via implicit hypergradient," *arXiv preprint arXiv:2110.03735*, 2021.

[124] Y. Li, X. Lyu, N. Koren, L. Lyu, B. Li, and X. Ma, "Anti-backdoor learning: Training clean models on poisoned data," *Advances in Neural Information Processing Systems*, vol. 34, pp. 14 900–14 912, 2021.

[125] ——, "Neural attention distillation: Erasing backdoor triggers from deep neural networks," *arXiv preprint arXiv:2101.05930*, 2021.

[126] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: Defending against backdooring attacks on deep neural networks," in *International symposium on research in attacks, intrusions, and defenses*. Springer, 2018, pp. 273–294.

[127] Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, and S. Nepal, "Strip: A defence against trojan attacks on deep neural networks," in *Proceedings of the 35th annual computer security applications conference*, 2019, pp. 113–125.

[128] Y. Liu, Y. Xie, and A. Srivastava, "Neural trojans," in *2017 IEEE international conference on computer design (ICCD)*. IEEE, 2017, pp. 45–48.

[129] Y. Zeng, H. Qiu, S. Guo, T. Zhang, M. Qiu, and B. Thuraisingham, "Deepsweep: An evaluation framework for mitigating dnn backdoor attacks using data augmentation," *arXiv e-prints*, pp. arXiv–2012, 2020.

[130] P. Zhao, P.-Y. Chen, P. Das, K. N. Ramamurthy, and X. Lin, "Bridging mode connectivity in loss landscapes and adversarial robustness," *arXiv preprint arXiv:2005.00060*, 2020.

[131] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[132] R. Kemker, M. McClure, A. Abitino, T. Hayes, and C. Kanan, "Measuring catastrophic forgetting in neural networks," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[133] J. Guo, A. Li, and C. Liu, "Aeva: Black-box backdoor detection using adversarial extreme value analysis," *arXiv preprint arXiv:2110.14880*, 2021.

[134] S. Kolouri, A. Saha, H. Pirsiavash, and H. Hoffmann, "Universal litmus patterns: Revealing backdoor attacks in cnns," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 301–310.

[135] M. Du, R. Jia, and D. Song, "Robust anomaly detection and backdoor attack detection via differential privacy," *arXiv preprint arXiv:1911.07116*, 2019.

[136] B. Tran, J. Li, and A. Madry, "Spectral signatures in backdoor attacks," *Advances in neural information processing systems*, vol. 31, 2018.

[137] R. Shokri *et al.*, "Bypassing backdoor detection algorithms in deep learning," in *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2020, pp. 175–183.

[138] A. Saha, A. Subramanya, and H. Pirsiavash, "Hidden trigger backdoor attacks," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 07, 2020, pp. 11 957–11 965.

[139] C. Molnar, G. Casalicchio, and B. Bischl, "Interpretable machine learning–a brief history, state-of-the-art and challenges," in *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 2020, pp. 417–431.

[140] 3rd Generation Partnership Project (3GPP), "TS 38.133 V17.2.0: NR; Requirements for Support of Radio Resource Management," 3rd Generation Partnership Project (3GPP), Tech. Rep. TS 38.133, 2024, release 17. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/138100_138199/138133/17.02.00_60/ts_138133v170200p.pdf

[141] ——, "TS 38.321 V17.4.0: NR; Medium Access Control (MAC) protocol specification," 3rd Generation Partnership Project (3GPP), Tech. Rep. TS 38.321, 2024, release 17. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/138300_138399/138321/17.04.00_60/ts_138321v170400p.pdf

[142] 3GPP, "TS 38.214 V16.7.0: NR; Physical layer procedures for data," 3rd Generation Partnership Project (3GPP), Tech. Rep. TS 38.214, 2022, section 5.3.2. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/138200_138299/138214/

[143] H. Cheng, S. D'Oro, R. Gangula, S. Velumani, D. Villa, L. Bonati, M. Polese, T. Melodia, G. Arrobo, and C. Maciocco, "Oranslice: An open source 5g network slicing platform for o-ran," in *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, 2024, pp. 2297–2302.

[144] O-RAN ALLIANCE, "O-RAN E2 Service Model (E2SM), RAN Control 8.0," O-RAN ALLIANCE, Technical Specification O-RAN.WG3.TS.E2SM-RC-R004-v08.00, June 2025, release R004. [Online]. Available: https://www.o-ran.org/specifications