SvedbergOpen
DISSEMINATION OF KNOWLEDGE

International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: https://www.svedbergopen.com/

Research Paper Open Access

STABLE-RAN: Counterfactual Drift Attribution and Stability-Aware Agentic Deep Learning for Verified Model Switching in Open Ran

[1*]Keshav Kumar, [2] Vivek Kumar, [3]Man Mohan Shukla,[4]Utkarsh Pandey, [5] Mohit Kumar Srivastava, [6]Sanjay Kumar Aggarwal

[1]Department of ECE, Pranveer Singh Institute of Technology, Kanpur-209305, UP, India. E-mail: keshav@gyancity.com
[2]Department of ECE, Pranveer Singh Institute of Technology, Kanpur-209305, UP, India. E-mail: rastogi0807@gmail.com
[3]Pranveer Singh Institute of Technology, Kanpur-209305, UP, India. E-mail: mshukla.psit@gmail.com
[4]Department of ECE, Pranveer Singh Institute of Technology, Kanpur-209305, UP, India. E-mail: utkashpandey08@gmail.com
[5]Department of ECE, Pranveer Singh Institute of Technology, Kanpur-209305, UP, India. E-mail: mohit.srivastava@psit.ac.in
[6]University Institute of Computing, Chandigarh University, Mohali-140413, Punjab, India. E-mail: sanjayaggarwal75@gmail.com

Abstract
Open Radio Access Network (O-RAN) controllers increasingly depend on deep models whose validity can decay as traffic, mobility, interference, and compute conditions change. Existing drift-handling work detects degradation and retrains models, while emerging agentic architectures propose lifecycle orchestration and rollback. A remaining management problem is deciding which operating cause produced the drift and whether a candidate model should be switched without causing lifecycle oscillation. This paper proposes STABLE-RAN, a verified model-switching framework that combines uncertainty-augmented Page-Hinkley monitoring, counterfactual feature-group attribution, regime-specialist multilayer perceptron ensembles, improvement-gated selection, a minimum dwell interval, pre deployment twin assessment, and versioned rollback. Evaluation uses a controlled normalized O-RAN service-management simulator with five sequential regimes, five random seeds, 2,100 intervals per seed, five lifecycle baselines, and 52,500 method-interval observations. Against a static deep model, STABLE-RAN reduces action mean absolute error from 0.0964 to 0.0351 and the SLA-violation rate from 0.4974 to 0.2397. Against an unconstrained agentic switcher, it reduces mean switches from 5.2 to 4.2 and oscillations from 1.6 to 0.4, but incurs higher error and a 4.76-percentage-point SLA penalty. Attribution accuracy is 0.773. In 800 independent candidate-corruption trials, all unsafe candidates are detected, benign false rollback is zero, and rollback lowers corrupted-candidate SLA violations from 0.330 to 0.245. The results establish a measurable stability–adaptation trade-off in controlled simulation; they are not carrier, hardware, or 3GPP-conformance measurements.

Keyword: Agentic AI, concept drift, counterfactual attribution, deep learning, model lifecycle management, Open RAN, rollback, stability assurance.

I. Introduction

O-RAN replaces monolithic radio access implementations with disaggregated functions, open interfaces, programmable controllers, and an AI/ML workflow spanning the service-management layer, the non-real-time RAN Intelligent Controller (RIC), and the near-real-time RIC. Release 5 of the O-RAN specifications extends this direction through AI/ML workflow services that connect development, training, exposure, and lifecycle functions across the two RIC time scales [1], [2]. The same programmability that accelerates innovation also creates an operational dependency: a deployed xApp or rApp is safe only while its data, model, policy, neighboring applications, and execution platform remain sufficiently close to the conditions under which it was validated.

Deep learning is attractive for traffic steering, resource allocation, mobility management, interference mitigation, anomaly detection, and energy control because it can approximate nonlinear policies from high-dimensional telemetry. Experimental toolchains such as ColO-RAN, OpenRAN Gym, and the Colosseum digital twin have made closed-loop development and repeatable evaluation practical [3]–[5]. These platforms also expose a core systems problem. A model that is accurate in a nominal condition may degrade after a traffic surge, mobility shift, interference episode, or compute bottleneck. Replacing it immediately whenever an alarm is raised is not automatically safe: noisy detection, ambiguous causes, and small estimated improvements can drive repeated model changes, inconsistent control behavior, and service instability.

The drift-handling framework of Gudepu et al. experimentally couples detection and adaptation for O-RAN [6], and GEN-DRIFT extends drift management with generative modeling [7]. Emerging agentic O-RAN

frameworks go further by allowing a governance agent to interpret intent, coordinate models, trigger retraining, validate candidates, deploy versions, and roll back failed changes [8], [9]. Crucially, the most closely related 2026 agentic framework identifies as open questions when to retrain, which model to deploy, and how to avoid oscillation between models [8]. Those questions are not peripheral implementation details; they determine whether autonomous lifecycle management improves reliability or becomes another source of instability.

This paper addresses that gap through STABLE-RAN, a stability-aware agentic lifecycle controller. The agent is not a conversational large language model and does not generate unrestricted network commands. It is a bounded management policy that observes telemetry, invokes diagnostic and validation tools, ranks registered deep models, enforces explicit switching constraints, and preserves a last-known-safe version. The design uses counterfactual feature-group interventions to attribute a detected change to traffic, mobility, interference, or hardware pressure. The attributed cause constrains candidate selection, while an improvement threshold and minimum dwell interval prevent opportunistic switching on weak evidence. A post-switch assurance window can restore the previous version when error exceeds a stated bound.

The contribution is therefore not another isolated drift detector or another high-accuracy xApp. It is a jointly evaluated lifecycle mechanism. First, STABLE-RAN couples residual monitoring with ensemble uncertainty so that a detector is sensitive to both realized error and predictive disagreement. Second, it produces cause-specific evidence by comparing recent telemetry with a nominal reference under feature-group interventions. Third, it makes the switching rule auditable through cause agreement, expected improvement, dwell time, twin validation, and rollback. Fourth, it evaluates service performance and management stability together using error, SLA violation, satisfaction, latency, block error rate (BLER), modeled energy, detection delay, false alarms, switch count, oscillations, attribution accuracy, and rollback stress.

A controlled simulator is used because the present study does not have access to synchronized carrier KPMs, IQ data, or a hardware-in-the-loop O-RAN platform. Every operating transition and target control action is known, allowing attribution and detection metrics to be computed without invented labels. Five independent seeds create 10,500 control intervals; evaluating five lifecycle methods yields 52,500 method-interval records. This evidence is suitable for method validation and ablation, but not for claims of field performance, radio conformance, or deployment readiness. The paper preserves that boundary throughout the method, results, and conclusion.

The remainder of the paper is organized as follows. Section II reviews O-RAN intelligence, drift handling, agentic management, explainability, and assurance. Section III states the research gap and requirements. Section IV defines the controlled system and optimization problem. Section V presents STABLE-RAN. Section VI describes the experimental protocol. Section VII explains the results and directly maps each observed effect to the gap. Sections VIII and IX discuss operational implications, validity boundaries, and conclusions.

II. Related Work And Research Positioning

A. O-RAN Intelligence and Experimental Platforms

The O-RAN architecture separates longer-horizon policy and analytics in the non-RT RIC from faster control in the near-RT RIC. Recent specification updates add explicit AI/ML model-management and exposure functions, E2 query and subscription capabilities, and conflict-mitigation hooks [2]. This architecture provides the necessary interfaces for lifecycle automation, but an interface does not determine a safe decision policy. Questions such as which version should be promoted, how long it should remain active, which evidence is sufficient for a switch, and when rollback is preferable remain implementation responsibilities.

ColO-RAN provides a large-scale software-defined-radios-in-the-loop environment for developing closed-loop machine-learning xApps [3]. OpenRAN Gym packages data collection, development, and testing workflows for O-RAN experiments across programmable wireless platforms [4]. Colosseum extends the experimental ecosystem with a high-fidelity digital-twin capability and automated radio and protocol-stack twinning [5]. These works are essential foundations for later hardware validation. The current study adopts their separation between offline training, controlled predeployment testing, and near-RT execution, but uses a normalized service-management simulator rather than claiming equivalence to those radio testbeds.

xApp implementation studies emphasize that performance depends on software interfaces, timing, data availability, portability, and interaction with other applications [10]. Explainability work such as EXPLORA links deep reinforcement-learning actions to network context and demonstrates that network-oriented explanations can support targeted steering [11]. STABLE-RAN uses the same operational principle—an action should be accompanied by intelligible evidence—but shifts the explanation target from an individual radio action to a model-lifecycle decision.

Beyond these platforms, broader surveys catalogue the O-RAN architecture, its open-source implementations, and the research challenges that remain open [18], and trace the evolution from traditional radio access toward disaggregated deployments together with the practical limits of the architecture [19], [20]. Comparative studies of deep reinforcement-learning xApps show that agent design, reward specification, and training data materially change closed-loop behavior [22], while federated formulations coordinate several xApps that would otherwise optimize conflicting objectives [21]. Coupling an open-source simulator to a RIC offers a lower-cost route to closed-loop automation experiments [23], and digital-twin prototypes demonstrate anomaly-detection functions built on replicated RAN state [24]. The interface definitions underlying all of this work are maintained in the O-RAN ALLIANCE specification portal [25]. None of these studies, however, treats the decision to replace a deployed model as the object of study; the model is assumed to remain valid once trained.

B. Concept Drift, Detection, and Predictive Uncertainty

Concept drift denotes a change in the relation between inputs and the prediction target. A monitoring system must distinguish genuine drift from noise, transient outliers, label delay, and covariate changes that do not impair the task. Contemporary surveys separate detection, understanding, and adaptation, and warn that a single global threshold can behave differently across nonstationary processes [13], [14]. Page-Hinkley testing remains a computationally light sequential baseline for detecting a persistent positive change in a monitored statistic [15]. It is used here because its state is transparent and inexpensive enough for a management loop; the novelty does not rest on presenting Page-Hinkley as a new detector.

The O-RAN drift-handling framework connects data collection, monitoring, retraining, and redeployment and reports experimental evidence that lifecycle adaptation can recover performance [6]. GEN-DRIFT adds generative-AI mechanisms for beyond-5G drift handling [7]. These studies show why a static model is inadequate, yet drift detection alone does not identify a unique operational cause. Two changes can produce similar residuals while requiring different specialists: low signal quality during mobility and rising interference can both increase error, but switching to the wrong model can worsen service.

Predictive uncertainty helps distinguish familiar and unfamiliar inputs. Deep ensembles provide a simple, scalable estimate from variation among independently trained neural networks [16]. STABLE-RAN adds ensemble disagreement to the realized residual so that monitoring reacts when either error or epistemic disagreement rises. This remains an empirical score, not a calibrated probability of failure. Its purpose is to improve lifecycle evidence, while the actual switch is controlled by downstream attribution and stability constraints.

C. Agentic Lifecycle Management and Operational Risk

Agentic network management extends fixed automation by allowing a controller to plan, call analytics and model-management tools, assess observations, and revise an action. The 2026 multi-scale agentic O-RAN framework assigns lifecycle governance to the non-RT layer, low-latency reasoning to the near-RT layer, and physical-layer inference to domain-specific models [8]. It explicitly includes model catalog, data lake, MLOps, digital twin, analytics, and rollback functions. The standards-oriented autogenic framework similarly emphasizes detect–diagnose–retrain–validate–deploy–monitor cycles and staged autonomy [9]. These architectures establish the correct components, but their breadth leaves the decision logic and quantitative lifecycle stability largely unresolved.

Autonomous model changes also create a software-engineering problem. Machine-learning systems accumulate hidden data and feedback dependencies; changing one component can alter downstream behavior even when its offline metric improves [17]. O-RAN further permits third-party xApps and rApps, expanding the threat and trust surface [12]. For these reasons, STABLE-RAN treats predeployment checking, bounded actuation, versioning, and rollback as part of the algorithm rather than deployment notes added after model selection.

D. Comparison with the Closest Verified Literature

TABLE I. COMPARISON WITH THE CLOSEST VERIFIED LITERATURE

Work	Drift adaptation	Cause evidence	Stability gate	Twin / rollback	Quantified lifecycle stability
ColO-RAN [3]	Control learning	No	No	Testbed	No
Drift framework [6]	Yes	Limited	No	Workflow	No

GEN-DRIFT [7]	Yes	Generative	No	Workflow	No
EXPLORA [11]	No	Action explanation	No	No	No
Agentic O-RAN [8]	Yes	Conceptual	Open problem	Conceptual	Open problem
Autogenic management [9]	Lifecycle vision	High-level	High-level	Yes	No
STABLE-RAN	Yes	Counterfactual groups	Threshold + dwell	Twin + version rollback	Yes

Within the reviewed evidence, no prior work jointly and experimentally evaluates four properties: cause attribution across traffic, mobility, interference, and compute regimes; stability-aware switching with both improvement and dwell constraints; predeployment verification plus versioned rollback; and simultaneous reporting of radio-service outcomes and lifecycle instability. Table I avoids a first-ever claim and distinguishes implemented evidence from conceptual coverage. This is the specific gap filled by STABLE-RAN.

III. RESEARCH GAP, RESEARCH QUESTIONS, AND DESIGN REQUIREMENTS

The research gap can be stated as a management-control problem. A lifecycle agent observes a rise in predictive error. It must decide whether the rise represents a persistent change, infer which operational factor is most consistent with the change, select a registered model that improves recent behavior, and determine whether the evidence is strong enough to interrupt the active control loop. Existing work addresses individual parts of this chain but does not constrain and measure the complete decision. A detector-only solution can over-adapt; a selector without attribution can promote a wrong specialist; a validation step without hysteresis can still oscillate; and rollback without a defined trigger is not reproducible.

Four research questions follow. RQ1 asks whether cause-constrained switching improves error and service reliability relative to static, periodic, and detector-triggered retraining. RQ2 asks whether an explicit stability gate reduces switch count and oscillation relative to an unconstrained agentic selector. RQ3 asks how much delay and performance are sacrificed by this conservatism. RQ4 asks whether a post-switch assurance rule detects corrupted candidates and reduces the service damage caused by a bad promotion. The associated hypotheses are directional but not assumed true: STABLE-RAN should outperform non-agentic baselines at matched simulation conditions, reduce lifecycle instability relative to the ungated agent, and restore a safe model after a harmful candidate.

TABLE II. GAP-DRIVEN DESIGN REQUIREMENTS

Gap element	Design requirement	STABLE-RAN mechanism	Measured evidence
Residual does not reveal cause	Identify plausible operating cause	Feature-group counterfactual attribution	Attribution accuracy by true regime
Detector can over-adapt	Require persistent, useful change	Uncertainty-PH trigger + 7.5% improvement	Delay, false alarms, error
Model selector can oscillate	Bound switching frequency	100-interval dwell and cause agreement	Switches and oscillations
Offline accuracy can hide service harm	Verify service-relevant behavior	Recent-window twin score and SLA metrics	SLA, latency, BLER, energy

Candidate can fail after promotion	Provide reversible actuation	30-interval assurance and safe-version rollback	800-trial corruption stress test

The intended contribution is a verified lifecycle policy, not a new radio scheduler. The deep networks approximate a normalized control target so that the lifecycle mechanisms can be isolated and compared. This distinction prevents an inflated novelty claim: the multilayer perceptron, Page-Hinkley test, and ensemble variance are known components. Novelty lies in their counterfactual, cause-constrained, stability-gated, and rollback-capable composition for O-RAN model management, together with the gap-aligned evaluation in Table II.

IV. SYSTEM MODEL AND PROBLEM FORMULATION

A. Lifecycle Architecture and Observable State

STABLE-RAN is positioned as a non-RT lifecycle rApp supervising a near-RT deep controller. The rApp receives windowed key-performance measurements and model diagnostics, accesses a registry of approved model versions, runs candidate comparisons in a service-management twin, and issues a bounded promote or rollback command. It does not replace the near-RT inference path. The separation keeps the expensive lifecycle reasoning outside the fast action loop and aligns with the O-RAN model-management direction in [1], [2]. Fig. 1 shows the resulting management loop and the order in which monitoring, attribution, gating, verification, and rollback are applied.

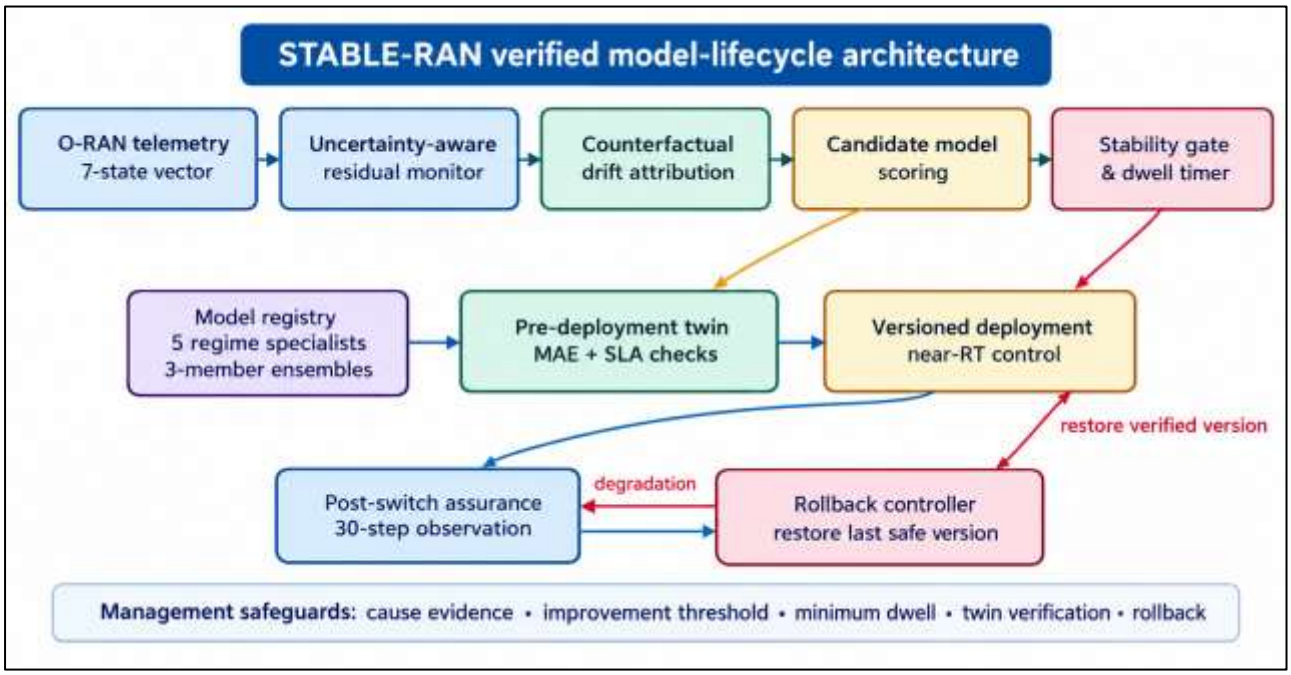

FIG. 1. STABLE-RAN VERIFIED LIFECYCLE ARCHITECTURE.

At control interval t, the normalized observable state is

$$\mathbf{x}_t = [\ell_t, \gamma_t, v_t, \iota_t, c_t, \sin(2\pi\tau_t), \cos(2\pi\tau_t)]^{\mathsf{T}} \quad (1)$$

where ℓ denotes offered load, γ is normalized signal-to-noise ratio, v is mobility intensity, ι is interference, c is computing utilization, and τ is normalized time of day. The sine and cosine components preserve cyclic time. All variables lie in [0,1] except the two cyclic components, which lie in [-1,1]. The latent operating regime is

$$z_t \in \mathcal{Z} = \{z^{(0)}, z^{(L)}, z^{(V)}, z^{(I)}, z^{(H)}\} \quad (2)$$

where $z^0, z^L, z^V, z^I,$ *and* z^H denote nominal, traffic-load, mobility, interference, and hardware-pressure conditions. The regime label is known only for controlled evaluation. During operation, the agent estimates a cause from telemetry rather than reading z_t.

B. Control Target, Service, and Energy Model

The desired normalized control action is generated by a regime-dependent nonlinear oracle:

$$a_t^\star = \mathrm{clip}\big(f_{z_t}(\mathbf{x}_t) + \varepsilon_t,\ 0.04,\ 0.96\big) \quad (3)$$

Here ε_t is zero-mean Gaussian target noise with standard deviation 0.012. The deterministic component is

$f_z(\mathbf{x}) = 0.10 + 0.53\ell + 0.18\iota + 0.11v - 0.10\gamma + 0.035\sin(2\pi\tau) + 0.02\ell\iota + g_z(\mathbf{x})$(4)

where $g_z$adds regime-specific nonlinear interactions: a squared-load term for traffic, mobility–SNR and load–mobility terms for mobility, squared-interference and load–interference terms for interference, and compute plus load–compute terms for hardware pressure. These functions are simulator definitions, not estimated equations for a deployed base station.

Service demand and action-dependent capacity are

$d_t = 18 + 92\ell_t, \qquad q_t = 20 + 155a_t(0.40 + 0.60\gamma_t)(1 - 0.45\iota_t)$ (5)

where d_t is normalized demand and q_t is served capacity. Satisfaction s_t and latency $L_t$are

$s_t = \frac{\min(d_t, q_t)}{d_t}, \qquad L_t = 5 + 43[1 - s_t]_+ + 8.5v_t + 5.5\iota_t$ (6)

where $[x]_- +$ $denotes$ $\max(0, x)$. BLER is modeled as

$B_t = \text{clip}(0.004 + 0.16[a_t^\star - a_t]_+ + 0.045\iota_t + 0.018v_t,\ 0,\ 0.35)$ (7)

and normalized energy is

$E_t = 0.38 + 1.45a_t + C_m$ (8)

where C_m is method-dependent compute overhead: 0 for the static model, 0.018 for periodic retraining, 0.014 for Page-Hinkley retraining, 0.022 for the unconstrained agent, and 0.030 for STABLE-RAN. Energy is a modeled index, not joules or measured power. This explicit naming prevents the simulation from being mistaken for a hardware-energy experiment.

C. Model Registry and Ensemble Uncertainty

The registry contains one specialist for each regime. Every specialist is a feedforward network with seven inputs, hidden widths 48–32–16, rectified-linear activation, and one scalar output:

$\mathbf{h}^{(k)} = \text{ReLU}\left(\mathbf{W}^{(k)}\mathbf{h}^{(k-1)} + \mathbf{b}^{(k)}\right), \quad \hat{a}_t = \mathbf{w}_o^{\mathrm{T}}\mathbf{h}^{(3)} + b_o$(9)

$where\ h^0 = x, W^k and\ b^k$ are the weight matrix and bias at hidden layer k, $and\ w_o and\ b_o$ define the output layer. Each specialist is trained using Adam, L2 coefficient 2×10^{-4}, early stopping, and a 15% internal validation fraction. Training uses 2,200 independently sampled states for the primary model of each regime. The registry additionally contains three-member ensembles trained with 1,600 states per member. For registered regime r, the ensemble mean is

$\overline{a}_t^{(r)} = \frac{1}{M}\sum_{j=1}^{M} \hat{a}_{t,j}^{(r)}$ (10)

and the ensemble disagreement is

$u_t^{(r)} = \sqrt{\frac{1}{M-1}\sum_{j=1}^{M}\left(\hat{a}_{t,j}^{(r)} - \overline{a}_t^{(r)}\right)^2}$ (11)

where M=3 and a-hat_(t,j)^(r) is member j's prediction. The mean supplies an uncertainty-aware diagnostic; the active primary network supplies the control action. Ensemble disagreement is not treated as a probability, and the method does not assume that three networks are statistically independent.

D. Lifecycle Objective and SLA Definition

A lifecycle method should minimize action error and SLA violations while avoiding excessive compute cost and switching. Conceptually, the agent minimizes a multi-objective sum of prediction loss, service damage, energy, switch cost, and oscillation cost subject to a dwell constraint and a rollback-safety rule. The experiment reports the components separately because a weighted scalar can conceal operational trade-offs. This is especially important when the unconstrained agent improves raw prediction but switches more frequently than STABLE-RAN.

An SLA violation occurs when any service threshold is crossed:

$\text{SLA}_t = \mathbb{1}[s_t < 0.90 \ \vee \ L_t > 20 \ \vee \ B_t > 0.10]$ (12)

The thresholds are satisfaction below 0.90, latency above 20 normalized milliseconds, or BLER above 0.10. They define the controlled experiment and must not be interpreted as universal 3GPP service limits.

V. THE STABLE-RAN FRAMEWORK

A. Uncertainty-Augmented Drift Monitoring

The monitor receives the active model's absolute residual and the corresponding ensemble disagreement. Its input statistic is

$e_t = |\hat{a}_t - a_t^\star| + \beta u_t, \qquad \beta = 0.55$ (13)

where $\beta = 0.55$ weights disagreement. A running mean is updated by

$\overline{e}_t = \overline{e}_{t-1} + \frac{e_t - \overline{e}_{t-1}}{t}$ (14)

and the Page-Hinkley cumulative statistic is

$H_t = H_{t-1} + e_t - \overline{e}_t - \delta, \qquad m_t = \min(m_{t-1}, H_t)$ (15)

where $\delta = 0.002$ is a tolerance for small positive deviations and m_t records the smallest cumulative value. A drift alarm is

$$D_t = \mathbb{1}[H_t - m_t > \lambda_{\mathrm{PH}}] \quad (16)$$

with $\lambda_{\mathrm{PH}} = 0.30$ for STABLE-RAN. A second guard requires the mean absolute error over the most recent 55 intervals to exceed 0.040. Thus, uncertainty can make the detector more attentive, but disagreement alone cannot trigger a switch. After an alarm is processed, the Page-Hinkley state is reset.

B. Counterfactual Feature-Group Attribution

A detected residual shift does not reveal its cause. STABLE-RAN groups telemetry into load L, mobility V, interference I, and hardware H. For each group g, a counterfactual state replaces the recent group value with its nominal median while leaving the remaining features unchanged:

$$\mathbf{x}_t^{(-g)} = \mathrm{do}\left(\mathbf{x}_{t,g} \leftarrow \tilde{\mathbf{x}}_g^{(0)}\right) \quad (17)$$

where the do operator denotes an intervention and $\tilde{x}_{\mathrm{g}}^{(0)}$ is the nominal prototype. The implementation computes a standardized group-deviation score

$$A_g(t) = \sum_{j \in g} \omega_j \frac{\bar{x}_{t,j} - \tilde{x}_j^{(0)}}{\sigma_j^{(0)}} \quad (18)$$

over a 45-interval window. Here $\bar{x}_{\mathrm{t,j}}$ is the recent mean of feature j, $\tilde{x}_{\mathrm{j}}^{(0)}$ and $\sigma_{\mathrm{j}}^{(0)}$ are its nominal median and scale, and ω_{j} are fixed feature weights. Traffic uses the load deviation; interference uses the interference deviation; hardware uses compute utilization; mobility combines increased mobility with reduced SNR. The attributed cause is

$$\hat{g}_t = \arg\max_{g \in \{L,V,I,H\}} A_g(t) \quad (19)$$

The intervention language makes the diagnostic question explicit: which feature group, if restored toward its nominal reference, most reduces the evidence for the observed operating shift? The controlled implementation uses standardized deviations rather than a learned structural causal model. Consequently, the output is counterfactual attribution within the simulator, not proof of physical causation in a live RAN.

C. Candidate Scoring and Stability-Gated Switching

The attributed cause points to a registered specialist, but cause agreement is not sufficient for deployment. Every candidate is scored on the previous W=45 labeled intervals using

$$J_r(t) = \frac{1}{W}\sum_{i=t-W}^{t-1} \left|\hat{a}_i^{(r)} - a_i^\star\right| + \eta \frac{1}{W}\sum_{i=t-W}^{t-1} \hat{a}_i^{(r)} \quad (20)$$

where $\eta = 0.007$ penalizes large average actions as a light energy regularizer. Labels are available without delay in the controlled experiment; an operational implementation would need delayed labels, a self-supervised quality proxy, or a digital-twin outcome. The relative candidate improvement is

$$\Delta_r(t) = \frac{J_{r_t}(t) - J_r(t)}{\max(J_{r_t}(t), \epsilon)} \quad (21)$$

where r_{t} is the active model, r is the candidate, and ε prevents division by zero. The evidence gate is

$$G_t(r) = \mathbb{1}[D_t = 1]\mathbb{1}[r = \hat{g}_t]\mathbb{1}[\Delta_r(t) > \rho] \quad (22)$$

with $\rho = 0.075$. It requires a drift alarm, agreement between the candidate and attributed cause, and more than 7.5% estimated improvement. A minimum dwell constraint further requires

$$t - t_{\mathrm{last}} \geq T_{\mathrm{dwell}} \quad (23)$$

where T_dwell=100 intervals. The final switching rule is

$$r_{t+1} = \begin{cases} r, & G_t(r) = 1 \ \wedge \ t - t_{\mathrm{last}} \geq T_{\mathrm{dwell}}, \\ r_t, & \text{otherwise}, \end{cases} \quad (24)$$

This rule is intentionally conservative. It can delay a beneficial model and may temporarily preserve a worse active version. In return, it blocks changes based on marginal differences and prevents any two accepted promotions from occurring within the dwell interval. The resulting performance cost is measured rather than assumed negligible.

D. Twin Assessment, Versioning, and Rollback

Before promotion, all registered specialists are evaluated on the same recent window and modest twin stress. Promotion creates a new active version while retaining the immediately preceding safe version. A 30-interval post-switch assurance window then evaluates the promoted model. Rollback is triggered when

$$\mathrm{RB}_t = \mathbb{1}\left[\frac{1}{T_A}\sum_{i=t-T_A+1}^{t} |\hat{a}_i - a_i^\star| > \kappa\right] \quad (25)$$

where $\mathrm{T_A} = 30$ and $\kappa = 0.105$ in the online lifecycle experiment. When the condition holds, the saved model and version identifier are restored. A rollback is logged as a second lifecycle transition because restoring a version changes the active control model.

A separate corruption stress test evaluates this mechanism independently of ordinary drift events. For each seed, 160 thirty-interval windows are sampled. Forty windows receive a benign action bias of ±0.02; 120 receive a corrupt bias chosen from ±0.12, ±0.18, and ±0.24. A candidate is detected when its MAE

degradation relative to the recorded stable action exceeds 0.055; degradation above 0.080 defines an unsafe candidate. A detected candidate remains active for ten assurance intervals and is then replaced by the stable action for the remaining twenty. These thresholds are declared before summarizing the stress results.

E. Algorithmic Summary and Decision Flow

Algorithm 1 states the complete lifecycle cycle and Algorithm 2 states the independent corruption assurance test. Fig. 2 renders the same decision logic as a flowchart, making explicit that three separate conditions — a valid alarm, cause agreement with sufficient improvement, and an expired dwell timer — must all hold before any version change is permitted, and that a promoted version can still be withdrawn by the assurance rule.

Algorithm 1: STABLE-RAN lifecycle cycle

Input: telemetry window x_t, active version r_t, specialist registry R, safe version r_{safe}, last switch time t_{last}
Output: next active version r_{t+1}, decision log
1: infer a_t from active model r_t; record residual $|a_t - a^*_t|$
2: compute ensemble mean and disagreement u_t by (10)–(11)
3: form e_t by (13); update running mean by (14)
4: update Page-Hinkley statistics H_t and m_t by (15)
5: $D_t \leftarrow 1[H_t - m_t > \lambda_{PH}]$ by (16)
6: if $D_t = 0$ or MAE over last 55 intervals ≤ 0.040 then
7: $r_{t+1} \leftarrow r_t$; return ▷ no valid alarm
8: end if
9: for each group $g \in \{L, V, I, H\}$ do compute $A_g(t)$ by (18) end for
10: $\hat{g}_t \leftarrow \arg\max_g A_g(t)$ by (19) ▷ attributed cause
11: for each $r \in R$ do compute $J_r(t)$ by (20) end for
12: $\Delta_r(t) \leftarrow$ relative improvement over r_t by (21)
13: if $G_t(r) = 1$ by (22) and $t - t_{last} \geq T_{dwell}$ by (23) then
14: $r_{safe} \leftarrow r_t$; $r_{t+1} \leftarrow \hat{g}_t$; $t_{last} \leftarrow t$ ▷ verified promotion
15: monitor r_{t+1} over the next $T_A = 30$ intervals
16: if $RB_t = 1$ by (25) then $r_{t+1} \leftarrow r_{safe}$ end if ▷ rollback
17: else
18: $r_{t+1} \leftarrow r_t$ ▷ stability gate blocks switch
19: end if
20: log telemetry, group scores, version, timing, and outcome

Algorithm 2: Candidate-corruption assurance test

Input: stable trajectory window S, proposed candidate C, thresholds $\tau_{rb} = 0.055$ and $\tau_{unsafe} = 0.080$
Output: rollback decision, unsafe label, SLA exposure
1: $MAE_S \leftarrow$ mean absolute error over the stable window
2: $MAE_C \leftarrow$ mean absolute error of the candidate over the same window
3: $\delta \leftarrow MAE_C - MAE_S$ ▷ degradation
4: unsafe $\leftarrow 1[\delta > \tau_{unsafe}]$ ▷ evaluation label only
5: rollback $\leftarrow 1[\delta > \tau_{rb}]$
6: retain C for 10 intervals to model assurance latency
7: if rollback = 1 then restore the recorded stable action for the remaining 20 intervals end if
8: report SLA violations with and without rollback
Note: the unsafe label at line 4 is never used to trigger the rollback at line 5; it is retained only for evaluation.

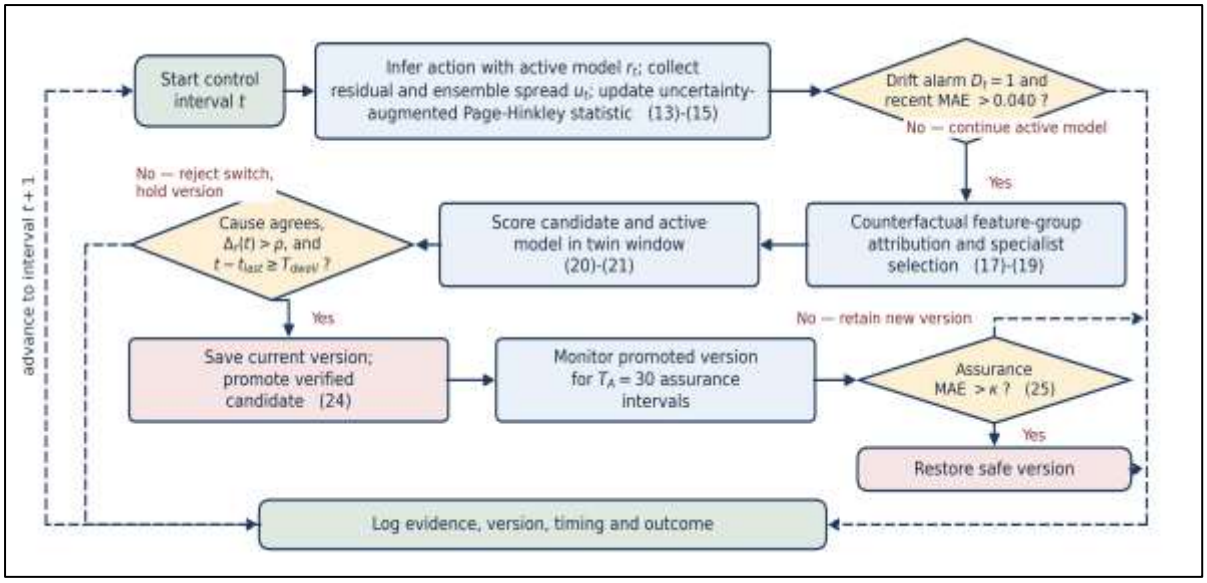

FIG. 2. STABLE-RAN decision flowchart.

A drift alarm alone does not authorize a model change: the candidate must also agree with the attributed cause, exceed the relative-improvement threshold ρ = 0.075, and satisfy the minimum dwell interval T_dwell = 100. A promoted version remains reversible for T_A = 30 intervals, after which the assurance rule either retains it or restores the last known-safe version.
Fig. 3 illustrates the behavior of this procedure on a single seed, where the delayed and occasionally imperfect switches show the practical effect of the stability gate.

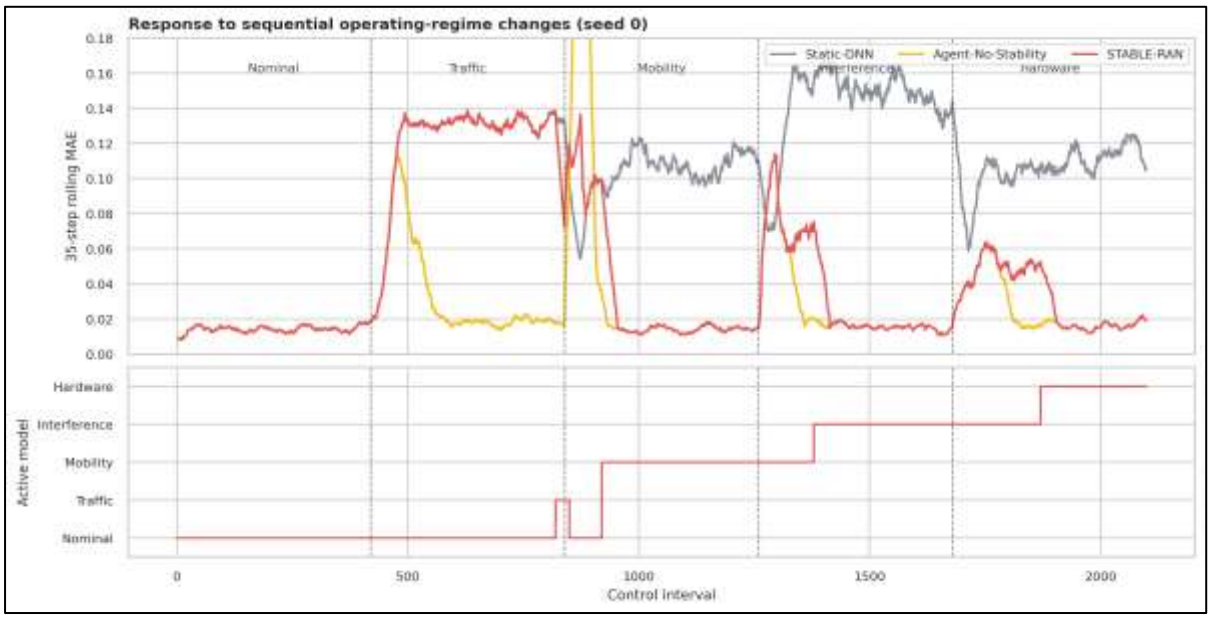

FIG. 3. Example response to four sequential regime changes in seed 0.

The upper panel shows 35-interval rolling action MAE; the lower panel shows the STABLE-RAN active specialist. The delayed and occasionally imperfect switches illustrate why the method is evaluated as a stability–adaptation trade-off rather than as an oracle.

VI. EXPERIMENTAL PROTOCOL

A. Controlled Stream Generation

Each seed generates a 2,100-interval stream containing five consecutive 420-interval blocks: nominal, traffic, mobility, interference, and hardware pressure. Transitions occur at intervals 420, 840, 1,260, and 1,680. The first 45 intervals of each post-nominal regime progressively mix in the regime-specific target term so that the change is not an instantaneous label flip. Base state distributions are beta or clipped-

Gaussian variables. Traffic raises load, mobility raises mobility and widens lower SNR, interference raises the interference feature, and hardware pressure raises compute utilization.

The test stream is generated independently from every model's training sample. For each seed, five primary specialists and fifteen ensemble members are trained using different random states. The evaluation therefore includes model-training variability, stream variability, and target noise. It does not include cell topology, user scheduling, protocol messages, waveform propagation, or real compute timing. The normalized state and service equations are sufficient to test lifecycle logic, not radio fidelity.

B. Lifecycle Baselines

Static-DNN retains the nominal specialist for the entire stream and isolates the cost of no adaptation. Periodic-DNN retrains a local network every 210 intervals using the preceding 240 labeled samples, regardless of whether a change occurred. PH-DNN applies a conventional Page-Hinkley detector to absolute residual and retrains a local network from the preceding 220 samples after an alarm. Agent-No-Stability monitors recent error every 25 intervals, scores all registered specialists on the preceding 90 intervals, and immediately changes to the lowest-loss specialist without a dwell constraint. STABLE-RAN uses the complete proposed procedure. Table III lists every simulation and algorithm setting used in the experiments so that each baseline can be reproduced exactly.

TABLE III. SIMULATION AND ALGORITHM SETTINGS

Category	Parameter	Value
Evaluation	Seeds / intervals per seed	5 / 2,100
Evaluation	Regimes / transitions per seed	5 / 4
Evaluation	Methods / total method-interval records	5 / 52,500
Deep model	Inputs / hidden layers / output	7 / 48–32–16 / 1
Deep model	Primary training states per regime	2,200
Ensemble	Members / states per member	3 / 1,600
STABLE-RAN	Attribution and scoring window	45 intervals
STABLE-RAN	PH tolerance / threshold	0.002 / 0.30
STABLE-RAN	Uncertainty weight	0.55
STABLE-RAN	Improvement / dwell	7.5% / 100 intervals
STABLE-RAN	Assurance window / online rollback MAE	30 / 0.105
Stress test	Trials / benign / corrupt	800 / 200 / 600

C. Evaluation Metrics

Action MAE is

$$\mathrm{MAE}_m = \frac{1}{N}\sum_{t=1}^{N}\left|a_t^\star - \hat{a}_{t,m}\right| \tag{26}$$

where N=2,100 for a seed and m identifies a lifecycle method. SLA-violation rate is

$$R_{\mathrm{SLA},m} = \frac{1}{N}\sum_{t=1}^{N}\mathrm{SLA}_{t,m} \tag{27}$$

The experiment also reports mean satisfaction, latency, BLER, and normalized energy. These metrics are calculated for every method at every interval from the same realized state and target, providing paired comparisons.

For true change k at time t_k, detection delay is

$$\mathrm{Delay}_{m,k} = \min\{t - t_k : t \geq t_k,\ D_{t,m} = 1\} \tag{28}$$

with a 210-interval search horizon; failure to detect within the horizon is assigned delay 210. A detection more than 70 intervals from every transition counts as a false alarm. Attribution accuracy is

$$\mathrm{Acc}_{\mathrm{attr}} = \frac{1}{K}\sum_{k=1}^{K}\mathbb{1}\left[\hat{g}_k = g_k\right] \tag{29}$$

where g_k is the known controlled cause. Lifecycle oscillations are

$$O_m = \sum_{j=2}^{J_m}\mathbb{1}\left[0 < t_{m,j} - t_{m,j-1} < T_{\mathrm{dwell}}\right] \tag{30}$$

where t_(m,j) is switch j of method m. The definition counts consecutive switches separated by less than the 100-interval dwell requirement. Thus, the stability metric is connected directly to the constraint it is intended to enforce.

Seed-level paired differences use

$$\Delta_{m,b} = \frac{1}{S}\sum_{s=1}^{S}\left(Y_{s,b} - Y_{s,m}\right) \tag{31}$$

where Y is a lower-is-better metric, b is a baseline, m is STABLE-RAN, and S=5. Positive values favor STABLE-RAN. Twenty thousand bootstrap resamples of the five paired seed differences produce

$$\mathrm{CI}_{0.95}(\Delta) = [Q_{0.025}(\Delta^*), Q_{0.975}(\Delta^*)] \quad (32)$$

Because five computational seeds are not a population sample, intervals quantify sensitivity to these runs rather than carrier-level generalization. No p-values are reported. The discussion focuses on effect direction, magnitude, interval width, and operational trade-offs.

D. Implementation and Reproducibility

The implementation uses Python, NumPy, pandas, scikit-learn, and Matplotlib. All random states are fixed. The retained records include per-interval state, target, action, active version, service metrics, seed summaries, regime summaries, event statistics, paired-bootstrap outputs, and all rollback trials. Figures are regenerated directly from these files; no graph value is manually entered. The manifest records five seeds, five methods, 2,100 steps per seed, 52,500 method-interval evaluations, four drift events per seed, and the absence of external measurements.

No external dataset or base-station log is used. The controlled status of every metric and the absence of external measurements are recorded in the manifest. Full ethics, author-metadata, and data-availability statements are provided before the references.

VII. Results

A. Overall Performance Against the Static Baseline

TABLE IV. OVERALL PERFORMANCE ACROSS FIVE SEEDS (MEAN ± SAMPLE SD)

Method	Action MAE	SLA viol.	Satisfaction	Latency	BLER	Norm. energy
Static-DNN	0.0964 ± 0.0065	0.4974 ± 0.0192	0.8969 ± 0.0046	13.878 ± 0.194	0.0388 ± 0.0011	1.007 ± 0.011
Periodic-DNN	0.1199 ± 0.0134	0.2699 ± 0.0403	0.9364 ± 0.0180	12.177 ± 0.776	0.0298 ± 0.0025	1.223 ± 0.050
PH-DNN	0.0589 ± 0.0015	0.2584 ± 0.0303	0.9434 ± 0.0054	11.876 ± 0.240	0.0293 ± 0.0008	1.139 ± 0.014
Agent-No-Stability	0.0291 ± 0.0008	0.1921 ± 0.0096	0.9563 ± 0.0023	11.325 ± 0.096	0.0261 ± 0.0005	1.163 ± 0.009
STABLE-RAN	0.0351 ± 0.0089	0.2397 ± 0.0556	0.9482 ± 0.0091	11.673 ± 0.389	0.0273 ± 0.0014	1.158 ± 0.015

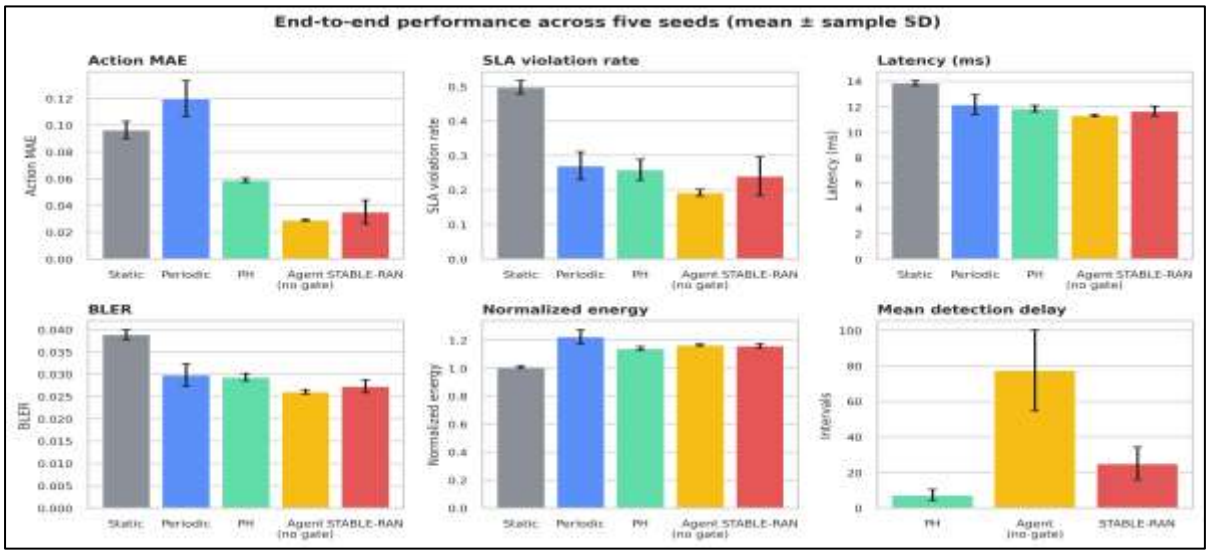

FIG. 4. End-to-end results across five seeds. Bars show seed means and error bars show one sample standard deviation. The six panels prevent action error, service reliability, latency, BLER, modeled energy, and detection speed from being compressed into a single favorable score.

STABLE-RAN reduces action MAE from 0.0964 for Static-DNN to 0.0351, a relative reduction of 63.6%. The mean paired absolute improvement is 0.0613, with a seed-bootstrap 95% interval [0.0548, 0.0680]. The interval is entirely positive, so the direction is consistent across the evaluated seeds. This result answers the first part of RQ1: an attributed and verified specialist registry substantially recovers predictive performance after non-nominal transitions compared with retaining the nominal network. Table IV reports the overall performance of all five lifecycle methods, and Fig. 4 presents the same comparison across six separate panels.

The error recovery propagates to service outcomes. The SLA-violation rate falls from 0.4974 to 0.2397, an absolute reduction of 25.77 percentage points. The paired reduction is 0.2577 [0.2156, 0.2955]. Mean satisfaction increases from 0.8969 to 0.9482. Latency decreases from 13.878 to 11.673, a 15.9% reduction, and BLER decreases by 29.8% from 0.0388 to 0.0273. These improvements follow from selecting a control model that better matches the active regime; they are not independently trained latency or BLER optimizers.

The service gain has an explicit cost. STABLE-RAN's normalized energy is 1.158, compared with 1.007 for Static-DNN. The increase combines more suitable—and often larger—control actions with a declared compute overhead of 0.030. It would be misleading to call the method energy-efficient relative to the static baseline. Energy becomes favorable only against Periodic-DNN and approximately competitive with the unconstrained agent; the paper therefore reports it as a co-objective rather than a primary win.

B. Comparison with Periodic Retraining

Periodic retraining performs poorly in this experiment: its mean MAE is 0.1199, versus 0.0351. The paired STABLE-RAN improvement is 0.0847 [0.0726, 0.1013]. Retraining every 210 intervals uses a rolling mixture of old and new regimes, so a locally fitted network can lag the current nonlinear target. STABLE-RAN instead chooses a specialist whose training distribution matches an attributed cause. This result directly supports the gap claim that adaptation timing and model identity must be decided together rather than equating retraining frequency with lifecycle quality.

The SLA-violation reduction versus Periodic-DNN is 0.0302, but its interval [-0.0409, 0.0921] crosses zero. Thus, the observed mean decrease from 0.2699 to 0.2397 is not uniformly stable across the five seeds. In contrast, normalized energy is lower by 0.0648 [0.0225, 0.1072], because periodic fitting carries repeated compute overhead and can choose larger actions. The correct interpretation is strong and consistent error improvement, consistent modeled-energy improvement, and uncertain SLA improvement at five seeds.

C. Comparison with Page-Hinkley Retraining and Regime-Level Behavior

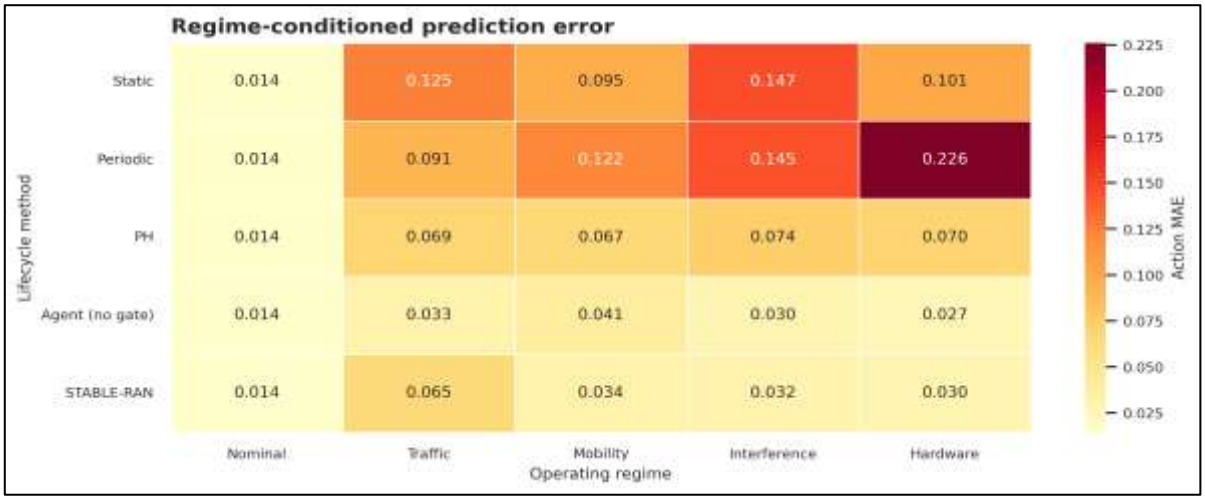

FIG. 5. Mean action MAE conditioned on the true operating regime. Static-DNN fails outside its nominal training distribution; periodic adaptation is unstable across later regimes; Agent-No-Stability achieves the lowest raw error; STABLE-RAN is strongest in mobility, interference, and hardware regimes but is delayed in the traffic regime.

PH-DNN is a stronger adaptive baseline. It detects quickly, with a mean delay of 7.4 intervals, and reaches MAE 0.0589. STABLE-RAN nevertheless reduces MAE by 0.0238 [0.0159, 0.0317], or approximately 40.4% relative to PH-DNN. Cause-constrained specialist selection is therefore more accurate than fitting one local model after every Page-Hinkley alarm in this regime stream.

The SLA reduction versus PH-DNN is only 0.0187 [-0.0232, 0.0682], and the latency reduction is 0.203 ms [-0.065, 0.529]; both intervals cross zero. However, BLER improves by 0.0021 with interval [0.0010, 0.0033]. These mixed outcomes show why a result section based only on prediction error would be insufficient for a network-management journal.

The regime matrix exposes both the mechanism and its weakness. All methods have nearly identical nominal MAE of 0.014 because they begin with the same nominal specialist. Static-DNN then reaches mean MAE 0.125, 0.095, 0.147, and 0.101 in traffic, mobility, interference, and hardware regimes. STABLE-RAN lowers those values to 0.065, 0.034, 0.032, and 0.030, respectively. The largest residual weakness is traffic: the improvement threshold and dwell logic delay promotion in some seeds, leaving a mean traffic MAE more than twice its mobility, interference, or hardware error. Fig. 5 conditions this comparison on the true operating regime and makes the traffic-regime weakness visible.

Counterfactual attribution reaches 0.773 across the events for which a cause decision is produced. This means approximately one in four attributed events is incorrect. The result is useful but not sufficient for autonomous carrier deployment. It shows that standardized feature-group interventions provide discriminative evidence within the simulator; it does not establish causal identification under unmeasured confounding. A live system should add topology, neighbor-cell, scheduler, and application-conflict evidence and attach a confidence interval or abstention option to the attribution.

The traffic weakness also demonstrates why the proposed method cannot be summarized as simply choosing the correct specialist. The agent must first detect a sustained residual increase, accumulate a 45-interval attribution window, show more than 7.5% improvement, and satisfy the 100-interval dwell constraint. Any one of these requirements can delay the switch. That delay is the price paid for preventing weak-evidence lifecycle changes. The following stability comparison determines whether the price buys the intended benefit.

D. Lifecycle Stability and the Cost of Conservatism

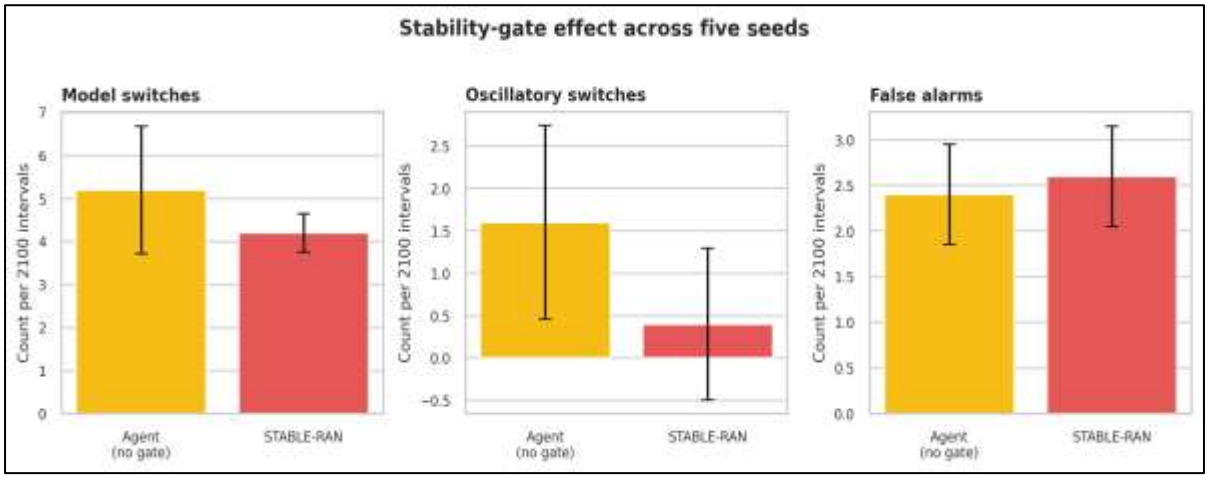

FIG. 6. Lifecycle stability comparison. STABLE-RAN lowers switch and oscillation counts relative to the unconstrained agent, while false alarms are similar. Counts are shown per 2,100-interval seed (mean ± sample standard deviation).

Agent-No-Stability produces 5.2 switches and 1.6 oscillations per seed. STABLE-RAN produces 4.2 switches and 0.4 oscillations. The reductions are 19.2% and 75.0%, respectively. This result answers RQ2: the improvement gate and dwell timer materially reduce lifecycle churn in the controlled stream. They do not eliminate every counted oscillation because a rollback can create a closely spaced restorative transition. Fig. 6 compares switch counts, oscillations, and false alarms across the two agentic methods.

False alarms are 2.4 for Agent-No-Stability and 2.6 for STABLE-RAN, so lower switching is not explained by a uniformly lower alarm count. The difference occurs after detection: STABLE-RAN rejects alarms that do not produce cause agreement, adequate estimated improvement, and an expired dwell timer. PH-DNN detects in 7.4 intervals but raises 8.4 false alarms, while STABLE-RAN detects in 25.15 intervals with 2.6 false alarms. The proposed assurance is therefore slower than PH-DNN but substantially less alarm-prone.

The stability gain is not a free performance win. Agent-No-Stability has MAE 0.0291, compared with 0.0351. The paired difference defined as baseline minus STABLE-RAN is -0.0060 [-0.0133, 0.0013], so the mean favors the ungated agent and the interval includes zero. More decisively, its SLA-violation rate is 0.1921

versus 0.2397; the paired difference is -0.0476 [-0.0948, -0.0054], entirely below zero. STABLE-RAN sacrifices 4.76 percentage points of SLA performance to obtain lower lifecycle churn.
Normalized energy is 1.1634 for Agent-No-Stability and 1.1582 for STABLE-RAN, a small mean reduction of 0.0052 whose interval [-0.0058, 0.0200] crosses zero. Latency and BLER also favor the ungated agent. Consequently, STABLE-RAN should be selected when version churn, explainability, bounded actuation, and rollback matter enough to justify delayed adaptation—not when the sole objective is the smallest simulated MAE. Fig. 7 places every method on the joint service–stability plane, where bubble area encodes oscillatory switching.

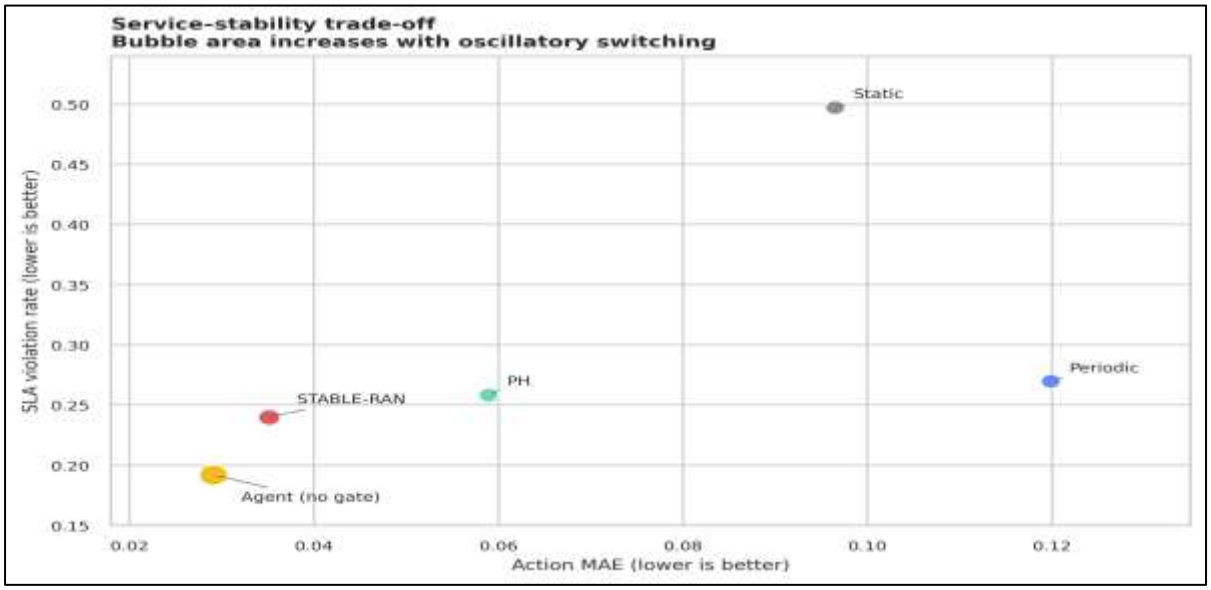

FIG. 7. Service–stability operating points. Lower-left is better for action MAE and SLA violation, while bubble area grows with oscillatory switching. Agent-No-Stability has the best raw service point but a larger oscillation burden; STABLE-RAN moves toward stronger lifecycle stability at a measurable service cost.

E. Candidate-Corruption and Rollback Stress Test

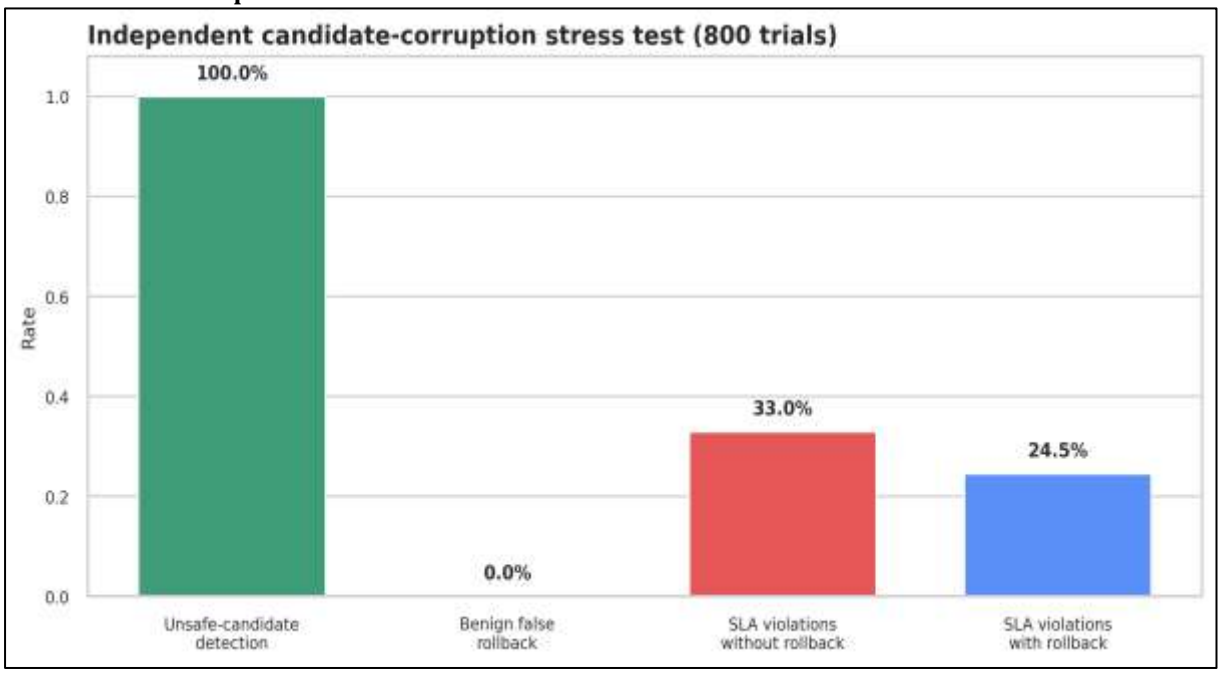

FIG. 8. Independent candidate-corruption stress test. Among 800 trials, 600 receive a corrupt bias and 200 receive a benign perturbation. All candidates labeled unsafe by the prespecified degradation rule are detected, benign false rollback is zero, and rollback lowers the mean SLA-violation exposure of corrupted candidates.

The stress test contains 800 independent sampled windows: 200 benign and 600 corrupt. Of the corrupt trials, 539 satisfy the prespecified unsafe degradation criterion. Unsafe-candidate detection is 1.000, and benign false rollback is 0.000. The detector also flags some corrupt candidates whose degradation lies between 0.055 and 0.080, which is appropriate because the rollback threshold is intentionally more conservative than the unsafe evaluation label. Fig. 8 summarizes the outcome of all 800 trials.

Without rollback, the mean SLA-violation exposure over corrupt windows is 0.330. Restoring the recorded stable action after ten assurance intervals lowers it to 0.245, an absolute reduction of 8.47 percentage points and a relative reduction of 25.7%. The remaining violations occur for two reasons: the corrupt candidate is deliberately active for ten intervals before restoration, and the stable trajectory itself can violate the composite SLA. Rollback therefore limits exposure; it does not guarantee a violation-free state.

This experiment fills the rollback-evidence component of the research gap because it separates three questions that are often conflated: whether a harmful candidate is detected, whether a benign candidate is incorrectly rolled back, and whether restoration improves the service outcome. The controlled design answers all three under known corruption. It does not evaluate model poisoning, adversarial telemetry, delayed KPMs, stateful scheduler recovery, or consistency across multiple Apps.

F. Mapping the Results to the Identified Gap

TABLE V. HOW THE PROPOSED METHOD FILLS THE IDENTIFIED GAP

Research gap	Proposed mechanism	Observed result	Scientific boundary
Drift alarm lacks cause	Counterfactual group attribution	Attribution accuracy 0.773	Controlled known regimes; not physical causality
Adaptation may choose wrong update	Cause-constrained registry + twin score	MAE 0.0351 vs static 0.0964	Specialists trained from simulator distributions
Agent can oscillate	7.5% improvement + 100-step dwell	Oscillations 0.4 vs ungated 1.6	Service penalty of 4.76 SLA percentage points
Offline metric may hide service effect	Joint KPI and lifecycle evaluation	SLA 0.2397; latency 11.673; BLER 0.0273	All are normalized simulator outputs
Promotion may be harmful	Post-switch assurance + safe version	Unsafe detection 100.0%; corrupt SLA 0.330→0.245	Synthetic bias, not adversarial or hardware fault

Table V is the central evidence chain. It prevents the method from being described only through architecture and prevents the results from being reported without reference to the original gap. Every mechanism has a corresponding measurement and an explicit validity boundary. The table also shows that gap closure is partial where evidence remains imperfect: attribution is 0.773 rather than one, stability costs service performance, and rollback reduces rather than eliminates exposure.

VIII. Discussion And Operational Implications

A. Policy Selection and Interpretability

The results support a tiered lifecycle policy. A highly dynamic but noncritical service may prefer Agent-No-Stability because it reaches the best raw action and SLA performance in the simulator. A reliability-sensitive service may accept STABLE-RAN's slower adaptation in exchange for fewer version changes, explicit cause evidence, reproducible switching criteria, and a safe-version path. The policy threshold ρ and dwell T_{dwell} should therefore be connected to service criticality, rollback cost, and operator risk appetite rather than fixed globally across all xApps.

Detection delay is a management variable, not simply an accuracy defect. A conservative detector needs more evidence before it allows a promotion; a very fast detector can repeatedly react to noise. STABLE-RAN's mean delay of 25.15 intervals is slower than PH-DNN's 7.4 but comes with far fewer false alarms than PH-DNN. The ungated agent is slower on average because it checks a broader recent-error condition and can miss a transition until the current model becomes clearly inadequate. These distinct behaviors show that detection, selection, and stability must be measured separately.

Counterfactual attribution increases auditability because it identifies the feature group that most strongly departs from the nominal reference and records why a particular specialist was considered. This evidence

is more actionable than a scalar anomaly score: a traffic cause can direct inspection toward load and admission controls, while hardware pressure can direct inspection toward compute saturation. However, an operator should see the contributing telemetry, alternative scores, recent model errors, and policy constraints—not merely the winning cause label.

B. Deployment Path in an O-RAN Environment

A practical implementation can map telemetry acquisition to E2SM-KPM and O1 data, registry and training functions to non-RT RIC AI/ML workflow services, policy transfer to A1, and near-RT control to an xApp using E2. The lifecycle rApp should store immutable version identifiers, training-data lineage, feature schema, calibration conditions, validation metrics, and rollback compatibility. The candidate's feature and action interfaces must be checked before any numerical comparison; a high-scoring model with an incompatible schema must never reach the stability gate.

Twin verification should progress from the normalized simulator used here to software-in-the-loop, protocol-stack emulation, channel-emulator testing, and a limited live canary. Colosseum and OpenRAN Gym provide suitable environments for this progression [4], [5]. The deployment decision can use staged traffic allocation: shadow inference, small canary exposure, broader exposure after assurance, and immediate version restoration when guard conditions fail. Such staging would measure actual inference time, RIC overhead, control-message delay, scheduler state, and cross-xApp conflicts, none of which are represented in the current simulator.

Security controls are necessary because a lifecycle agent can become a high-impact actuation point. The registry must verify signatures and provenance; telemetry paths require authentication and freshness; the agent needs least-privilege access; and rollback versions must remain immutable. O-RAN security analysis identifies third-party applications and open interfaces as important trust surfaces [12]. STABLE-RAN contributes decision transparency and bounded changes, but it does not replace zero-trust onboarding, runtime isolation, model integrity checks, or operator authorization.

IX. Conclusion

This paper presented STABLE-RAN, a bounded agentic deep-learning lifecycle framework for O-RAN model switching. It integrates uncertainty-augmented drift monitoring, counterfactual feature-group attribution, regime-specialist networks, recent-window twin scoring, a relative-improvement threshold, a minimum dwell interval, versioned deployment, post-switch assurance, and rollback. The design directly responds to the unresolved problem of deciding when to adapt, which model to deploy, and how to avoid model oscillation.

Across five controlled seeds and 52,500 method-interval evaluations, STABLE-RAN reduces action MAE by 63.6% and SLA violations by 25.77 percentage points relative to a static DNN. It also outperforms periodic and Page-Hinkley retraining on action MAE. Relative to the unconstrained agent, it reduces switches by 19.2% and oscillations by 75.0%, but its SLA-violation rate is 4.76 percentage points higher. Counterfactual attribution reaches 0.773. In 800 candidate-corruption trials, the assurance rule detects every prespecified unsafe candidate, produces no benign false rollback, and reduces corrupt-window SLA exposure from 0.330 to 0.245.

The evidence therefore supports a specific conclusion: STABLE-RAN improves verified lifecycle stability and strongly outperforms non-agentic adaptation baselines, while trading some raw service performance for bounded switching relative to an aggressive agent. It does not guarantee safe deployment. Future work should validate the method on Colosseum or another O-RAN-compliant testbed, use real KPM and protocol traces, learn multi-cause structural attribution, support delayed outcomes and unseen regimes, evaluate concurrent xApps, quantify actual compute and energy, and optimize the stability–service Pareto frontier under operator-defined risk budgets.

References

[1] O-RAN ALLIANCE, “O-RAN ALLIANCE completed its Specification Release 5 (O-RAN-R005),” Jun. 8, 2026. [Online]. Available: https://www.o-ran.org/blog/o-ran-alliance-completed-its-specification-release-5-o-ran-r005

[2] O-RAN ALLIANCE, “67 new or updated O-RAN technical documents released since November 2024,” Apr. 10, 2025. [Online]. Available: https://www.o-ran.org/blog/67-new-or-updated-o-ran-technical-documents-released-since-november-2024

[3] 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 Trans. Mobile Comput., vol. 22, no. 10, pp. 5787–5800, Oct. 2023, doi: 10.1109/TMC.2022.3188013.

[4] L. Bonati, M. Polese, S. D'Oro, S. Basagni, and T. Melodia, "OpenRAN Gym: AI/ML development, data collection, and testing for O-RAN on PAWR platforms," Comput. Netw., vol. 220, Art. no. 109502, Jan. 2023, doi: 10.1016/j.comnet.2022.109502.
[5] M. Polese et al., "Colosseum: The Open RAN digital twin," IEEE Open J. Commun. Soc., vol. 5, pp. 5452–5466, 2024, doi: 10.1109/OJCOMS.2024.3447472.
[6] V. Gudepu, V. R. Chintapalli, P. Castoldi, L. Valcarenghi, B. R. Tamma, and K. Kondepu, "The drift handling framework for Open Radio Access Networks: An experimental evaluation," Comput. Netw., vol. 243, Art. no. 110290, 2024, doi: 10.1016/j.comnet.2024.110290.
[7] V. Gudepu, B. Chirumamilla, V. R. Chintapalli, P. Castoldi, L. Valcarenghi, B. R. Tamma, and K. Kondepu, "GEN-DRIFT: Generative AI-driven drift handling for beyond 5G networks," Comput. Netw., vol. 263, Art. no. 111237, 2025, doi: 10.1016/j.comnet.2025.111237.
[8] H. Navidan, M. Cheraghinia, J. Fontaine, M. Seif, E. De Poorter, H. V. Poor, I. Moerman, and A. Shahid, "Toward autonomous O-RAN: A multi-scale agentic AI framework for real-time network control and management," arXiv:2602.14117, 2026.
[9] P. Djukic, S. Acharya, T. E. Kennouche, and B. Kantarci, "From agentic to autogenic network management for AI-native 6G and beyond: A standards perspective," arXiv:2607.06786, 2026.
[10] M. Hoffmann et al., "Open RAN xApps design and evaluation: Lessons learnt and identified challenges," IEEE J. Sel. Areas Commun., vol. 42, no. 2, pp. 473–486, Feb. 2024, doi: 10.1109/JSAC.2023.3336190.
[11] C. Fiandrino, L. Bonati, S. D'Oro, M. Polese, T. Melodia, and J. Widmer, "EXPLORA: AI/ML explainability for the Open RAN," Proc. ACM Netw., vol. 1, no. CoNEXT3, Art. 19, pp. 1–26, 2023, doi: 10.1145/3629141.
[12] P. Porambage, M. Christopoulou, B. Han, M. A. Habibi, H. Bogucka, and P. Kryszkiewicz, "Security, privacy, and trust for Open Radio Access Networks in 6G," IEEE Open J. Commun. Soc., vol. 6, pp. 332–361, 2025, doi: 10.1109/OJCOMS.2024.3519725.
[13] F. Hinder, V. Vaquet, and B. Hammer, "One or two things we know about concept drift—A survey on monitoring in evolving environments. Part A: Detecting concept drift," Front. Artif. Intell., vol. 7, Art. no. 1330257, 2024, doi: 10.3389/frai.2024.1330257.
[14] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," ACM Comput. Surv., vol. 46, no. 4, Art. 44, 2014, doi: 10.1145/2523813.
[15] E. S. Page, "Continuous inspection schemes," Biometrika, vol. 41, no. 1/2, pp. 100–115, Jun. 1954, doi: 10.1093/biomet/41.1-2.100.
[16] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in Proc. Adv. Neural Inf. Process. Syst., vol. 30, 2017, pp. 6402–6413.
[17] D. Sculley et al., "Hidden technical debt in machine learning systems," in Proc. Adv. Neural Inf. Process. Syst., vol. 28, 2015, pp. 2503–2511.
[18] W. Azariah, F. A. Bimo, C.-W. Lin, R.-G. Cheng, N. Nikaein, and R. Jana, "A survey on Open Radio Access Networks: Challenges, research directions, and open source approaches," Sensors, vol. 24, no. 3, Art. 1038, 2024, doi: 10.3390/s24031038.
[19] S. K. Singh, R. Singh, and B. Kumbhani, "The evolution of radio access network towards Open-RAN: Challenges and opportunities," in Proc. IEEE WCNC Workshops, 2020, pp. 1–6, doi: 10.1109/WCNCW48565.2020.9124820.
[20] A. S. Abdalla, P. S. Upadhyaya, V. K. Shah, and V. Marojevic, "Toward next generation open radio access networks—What O-RAN can and cannot do!" IEEE Netw., vol. 36, no. 6, pp. 206–213, Nov./Dec. 2022, doi: 10.1109/MNET.005.2100655.
[21] H. Zhang, H. Zhou, and M. Erol-Kantarci, "Federated deep reinforcement learning for resource allocation in O-RAN slicing," in Proc. IEEE Global Communications Conference (GLOBECOM), Rio de Janeiro, Brazil, Dec. 2022, pp. 958–963.
[22] M. Tsampazi, S. D'Oro, M. Polese, L. Bonati, G. Poitau, M. Healy, and T. Melodia, "A comparative analysis of deep reinforcement learning-based xApps in O-RAN," in Proc. IEEE GLOBECOM, 2023.
[23] T. Karamplias, S. T. Spantideas, A. E. Giannopoulos, P. Gkonis, N. Kapsalis, and P. Trakadas, "Towards closed-loop automation in 5G Open RAN: Coupling an open-source simulator with xApps," in Proc. EuCNC/6G Summit, 2022, pp. 232–237.
[24] P. Li, A. Aijaz, T. Farnham, S. Gufran, and S. Chintalapati, "Demo: A digital twin of the 5G radio access network for anomaly detection functionality," in Proc. IEEE ICNP, 2023, pp. 1–2.
[25] O-RAN ALLIANCE, "O-RAN specifications," accessed Aug. 22, 2026. [Online]. Available: https://www.o-ran.org/specifications