

A Context-Aware MLOps-Framework for Next-Generation Mobile Networks

Franz Pouhela¹, Florian Langenstein¹, Christoph Fischer¹, Hans D. Schotten^{1,2}

¹*German Research Center for Artificial Intelligence (DFKI Kaiserslautern)*

²*University of Kaiserslautern-Landau Germany (RPTU Kaiserslautern)*

{franc.pouhela, florian.langenstein, christoph.fischer, schotten}@dfki.de

Abstract—The operationalisation of Machine Learning (ML) workflows within modern networks such as 5G/6G introduces unprecedented challenges associated with extreme distributed heterogeneity and the management of dynamically evolving data flows. Contemporary Machine Learning Operations (MLOps) platforms (e.g., MLflow, Kubeflow, etc.) are fundamentally designed for declarative, static, cloud-native environments and are thus ill-suited to accommodate key 6G characteristics, including real-time computational migration, resource elasticity, and localised topological autonomy. In this paper, we propose a context-aware, agentic MLOps-framework architecture powered by a dynamic Context Management System (CoMaS). Furthermore, 3GPP-specified Network Functions (NFs), such as the Model Training Logical Function (MTLF) and Analytics Logical Function (AnLF), are dynamically mapped by agents that parse standard RichML-App structures. Finally, the proposed framework natively embeds Human-in-the-Loop (HITL) prompting mechanisms within zero-touch operational environments, thereby enhancing system robustness and operational safety.

Index Terms—NWDAF, MLOps, LLM, Agentic, Harness, 6G

I. INTRODUCTION

The deployment of Artificial Intelligence (AI) and Machine Learning (ML) applications within telecommunication layers is one of the pivotal shifts anticipated in 6G networks [1], [2]. The transition requires algorithms to not only optimise end-user telecommunication services (Networks for AI) but also leverage native network telemetry data to optimise infrastructural performance continuously (AI for Networks) [3]. Consequently, ensuring appropriate lifecycle integration, placement, execution, and retirement of ML workloads has led to the necessity of dedicated network management layers.

Historically, Machine Learning Operations (MLOps) platforms such as *MLFlow*, *Flyte*, or centralised *Kubernetes* aggregators have been standard protocol for maintaining data centre models [4]. However, telecommunication substrates drastically deviate from traditional infrastructures. 6G architectures encapsulate a profound scale of heterogeneity regarding available times, geographic localities, edge computational variances (e.g., standard cores versus integrated radio accelerators), and security provisioning [5]. This dynamic limits standard declarative automated platforms, precipitating the need for highly adaptive autonomous governance layers.

Standardisation efforts within the 3GPP ecosystem, particularly in Releases 17 & 18, have formalised key enablers

such as the Network Data Analytics Function (NWDAF), Analytics Logical Function (AnLF), and Model Training Logical Function (MTLF) [6], [7]. However, these standards offer limited guidance on dynamic runtime orchestration. How should resources be continuously allocated at the network edge in the presence of conflicting application priorities? Furthermore, how can multiple ML mechanisms, simultaneously interacting with the transmission capabilities of a localised Base Station (BS), execute actions without inducing interference? To address these challenges, this paper introduces a context-aware, agentic MLOps-framework. Recent advances in AI, particularly in *Agentic Harness Orchestration* via abstraction gateways, reimagine network configurations not as static, deterministic rule-based pathways, but as autonomous, task-driven systems capable of dynamically executing 3GPP protocols across distributed clusters.

The remaining of this paper is structured as follows: Section II reviews 6G architecture complexities and Large Language Model (LLM) orchestration capabilities. Section III presents the agentic framework and outlines our definition of a native RichML-App, while Section III-F discusses execution and validation models as well as the governance framework. Finally, Section IV provides a conclusion and future work.

II. RELATED WORK

Extending MLOps pipelines to 6G network layers mandates acknowledging the physical latency boundary [4]. Models trained for specific network optimisations such as massive Multiple-Input Multiple-Output (MIMO) Beamforming demand hyper-local intelligence that centrally coordinated hypervisors cannot manage dynamically. If a localised BS possesses temporarily free processing arrays at night due to cyclic human traffic models, classic MLOps platforms cannot autonomously discover and exploit these transient compute windows efficiently to initiate batched ML retraining jobs [5].

The 3GPP has formulated analytical modules to embed intelligence deep into the 5G core via the NWDAF [6]. Functionally speaking, the system executes model training generically via the MTLF while actual algorithmic inferences are rendered via the AnLF. Because implementation paths of these logic components are currently undefined in granular edge clusters, orchestrating their placement bridges an open significant functional gap [7].

While Generative AI tools initially achieved breakthrough success via generalised human language generation, the evolution toward structured Multi-Agent frameworks enables bounded action execution over Information Technology (IT) systems. Gateways such as *OpenClaw* [8] and *Hermes* [9] represent tools that connect generative models natively to external Application Programming Interfaces (APIs). Through explicit system definitions (such as standardised *Skills* files acting as operational parameters), LLMs can be utilised to independently parse highly specific constraints and deploy scripts programmatically without direct human scripting per operation [10]. Integrating these open-source tools effectively grants telecommunication networks advanced semantic comprehension of runtime telemetry.

III. AGENTIC MLOPS FRAMEWORK

5G/6G networks combine geographically distributed data sources, heterogeneous computing resources, and operational constraints that vary across network locations and administrative domains. As illustrated in (Figure 1), workloads may execute across cloud, edge, and Radio Access Network (RAN) infrastructure, where resource availability, connectivity, and proximity to network functions differ. Latency-sensitive inference may require execution close to its data sources and controlled equipment, whereas training can often exploit more distant resources. These conditions motivate an orchestration framework that explicitly relates application requirements to network context throughout the ML lifecycle.

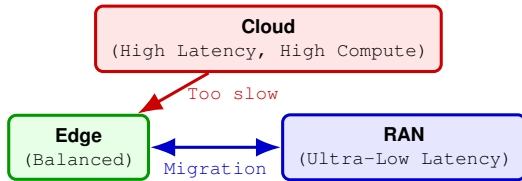


Fig. 1: Heterogeneous execution environments in mobile networks.

The proposed Agentic-MLOps framework, shown in (Figure 2), combines specialised advisory agents, machine-readable application contracts, context management, and deterministic execution. Agents interpret requests and propose operations; the execution harness validates these proposals and invokes approved infrastructure interfaces. An agent-generated explanation is therefore neither an authorisation nor evidence that an operation satisfies its constraints. The framework separates supervisory orchestration from latency-bound inference and network control. LLMs may assist with interpreting underspecified intentions, retrieving relevant information, and explaining alternatives. They are not required for routine placement or lifecycle decisions that can be expressed through deterministic rules. Their potential benefit concerns the interpretation of heterogeneous information, while their latency, cost, and reliability must be evaluated separately.

A. Architectural Components

The contribution is the integration of three mechanisms: a RichML-App contract that connects ML workloads to network observables, permitted actions, and operational constraints; coordinated placement and lifecycle decisions that distinguish latency-sensitive inference from elastic training; and an execution boundary that subjects agent proposals to independent validation using current context. The framework reuses established infrastructure and management principles. Kubernetes provides workload deployment and reconciliation, while the AnLF and MTLF provide functional reference points for inference and training. Likewise, context management and retrieval build on our prior work on Context Management System (CoMaS) [11] and context-aware retrieval [12].

Relative to pipeline-oriented MLOps, the proposed design makes the relationship between workload placement, network-local observations, and physical action permissions explicit. Relative to autonomous network-management loops, it specifies model training, validation, promotion, and rollback within that loop. Relative to general agent-based orchestration, it defines a shared contract and validation boundary across planning, placement, lifecycle management, and actionable arbitration. These are architectural contributions; the framework does not introduce a new placement optimiser or establish that LLM-based reasoning outperforms conventional controllers.

Interoperability is addressed through explicit adapters. A Kubernetes adapter translates admitted operations into Jobs, Deployments, and Services. Network adapters must map named observables and actionables to equipment-specific or standardised interfaces, preserving units, scope, permissions, and error semantics. Mapping components to 3GPP functional roles alone does not establish interface interoperability.

B. Deterministic Execution Harness

The harness manages activation, proposal validation, workflow state, execution, and observation. Its configuration separates persistent behavioural guidance from authoritative operational policy. Instructions stored in agent memory cannot override the application’s contract or the harness policy. Three activation mechanisms support ongoing operation.

1) Temporal Triggers: Scheduled activation initiates periodic evaluation or maintenance, such as checking resource availability before a daily retraining task. A schedule makes a workflow eligible for consideration; resource, freshness, and policy checks still determine whether it can execute.

2) Event-Driven Activation: Telemetry adapters submit structured events, such as an accuracy threshold violation or a resource-capacity change. Events retain their source, timestamp, application scope, and measured values. Natural-language summaries may accompany these records, but do not replace the structured evidence used for admission.

3) Asynchronous Progress Monitoring: The harness monitors admitted operations through bounded polling or infras-

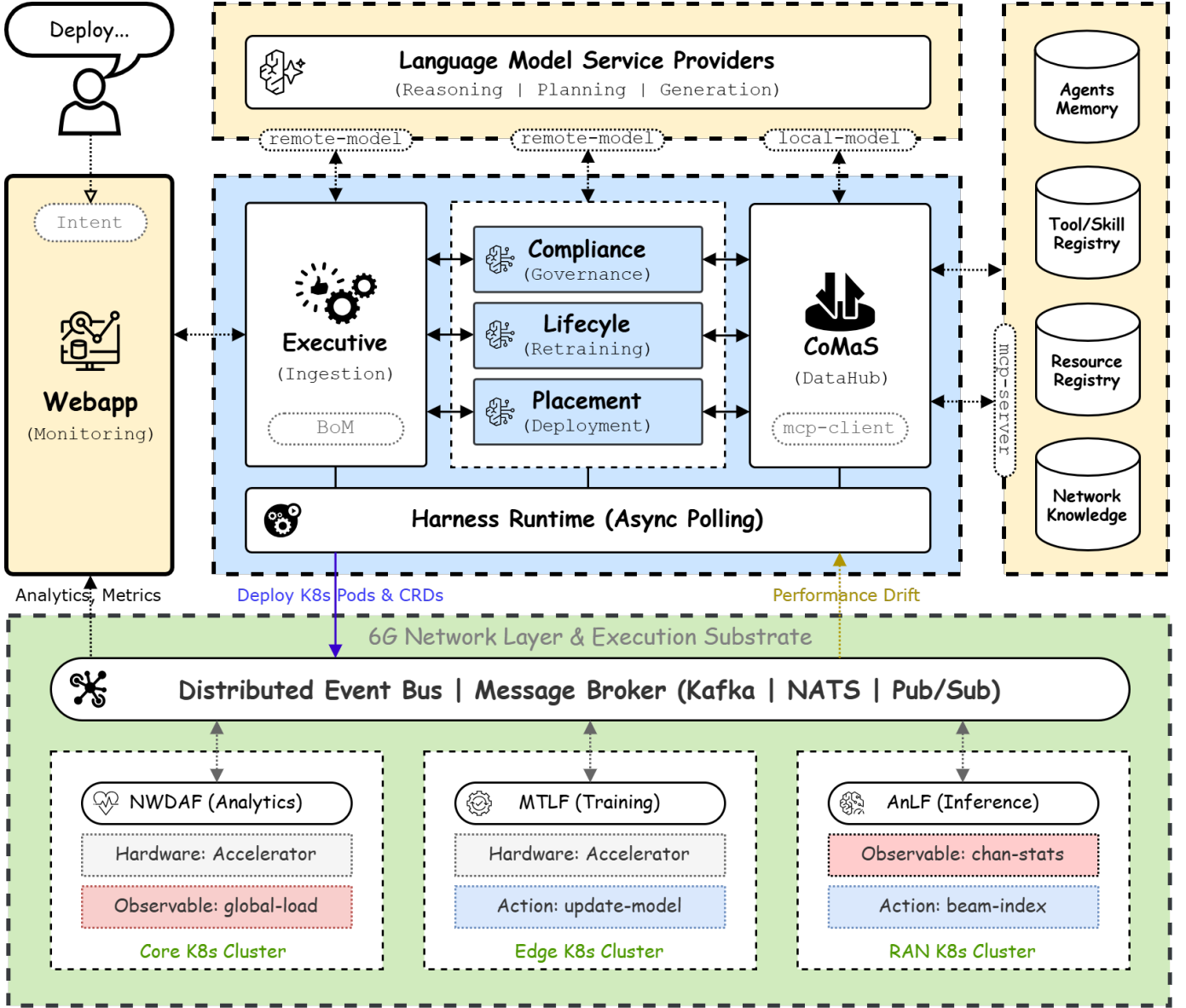


Fig. 2: Proposed Agentic-MLOps architecture. Specialised agents consume application contracts and operational context, while a deterministic harness validates proposals and invokes approved execution interfaces. The AnLF and MTLF indicate functional roles; their representation does not establish conformance with standardised 3GPP interfaces.

structure events, including Job completion and Pod readiness. It records intermediate states, deadlines, and failures before advancing the workflow. Agents may propose subsequent steps, but infrastructure calls remain restricted to the harness's approved operations.

C. RichML-App Formalisation

A RichML-App describes an application together with the information required to deploy and govern it. Its Bill of Materials (BoM)-like descriptor declares software and data dependencies, observables, actionables, resource requirements, eligible execution locations, scheduling conditions, and lifecycle policies. In this paper, RichML-App denotes

the application abstraction, while `RichMLService` denotes its machine-readable resource representation. The descriptor distinguishes hard constraints from optimisation preferences. Hard constraints determine whether an operation is admissible; preferences rank admissible alternatives. For example, an inference latency limit cannot be traded against a preference for lower training load. A missing or unsupported hard constraint prevents admission until the requirement is clarified or a suitable adapter is available. (Figure 3) illustrates representative observation and action hooks. (Listing 1) extends this boundary with explicit latency and power requirements, addressing the requirements conveyed by the intent example in (Figure 4). The numerical limits are illustrative: the latency

requirement concerns the 95th percentile of the observation-to-action path over a 60-second window, while the power requirement concerns the mean draw of the specified base station over the same interval. Actual limits must be supplied by the operator and represented with an unambiguous scope and measurement definition.

```
apiVersion: mlops/v1
kind: RichMLService
metadata:
  name: bs-control
spec:
  version: "1.2.0"
  tenant: operator-a
  target: cell-001

  observables:
    - name: rsrp-api
    - name: cell-load
    - name: bs-power
      unit: W
    - name: control-latency
      unit: ms

  actionables:
    - name: set-tilt
      target: cell-001
      unit: degree
      min: -10
      max: 10

  inference:
    resources:
      cpu: "1"
      memory: "1Gi"
    endpoint: "/v1/predict"
    regions: ["ran", "edge"]
    latency:
      observable: control-latency
      path: observation-to-action
      statistic: p95
      windowSeconds: 60
      maxMs: 5

  training:
    resources:
      cpu: "2"
      memory: "4Gi"
      accelerator: gpu
    regions: ["edge", "cloud"]
    schedule: "0 0 * * *"
    dataset: traffic-telemetry

  constraints:
    power:
      observable: bs-power
      target: cell-001
      statistic: mean
      windowSeconds: 60
      maxWatts: 500

  lifecycle:
    accuracyThreshold: 0.90
    minImprovement: 0.02
    cooldownSeconds: 60

  context:
    maxAgeSeconds: 30
```

Listing 1: Illustrative RichML-App contract with explicit constraints

1) Observables and Actionables: Observables identify permitted telemetry inputs. Each adapter must define the corresponding data type, units, target, source, and freshness semantics. Actionables identify permitted state changes, including their target and parameter bounds. For example,

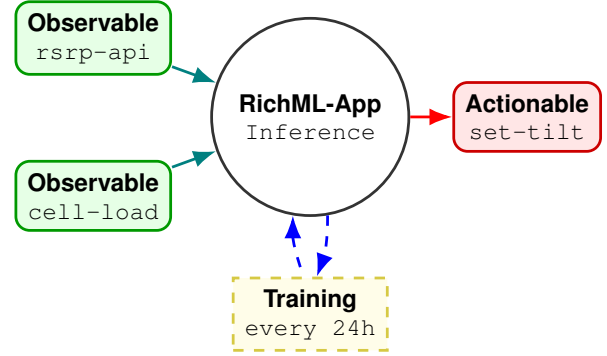


Fig. 3: Representative observation and action boundaries of a RichML-App

declaring `set-tilt` permits only the specified operation within its declared range; it does not authorise arbitrary equipment commands. These declarations provide input to access provisioning and runtime validation, but do not themselves establish authentication or isolation. Enforcement requires trusted adapters and an appropriate identity and authorisation mechanism. Similarly, declaring a power limit is insufficient to enforce it: admission requires suitable measurements and a trusted control mechanism or equipment-enforced limit. A Kubernetes Custom Resource Definition (CRD) stores the application contract. The network hooks referenced by that contract are resolved through registered adapters; they are not automatically standardised interfaces merely because their declarations are represented in Kubernetes.

D. Specialised Agent Decision Mechanisms

The framework assigns distinct responsibilities to the Executive, Placement, Lifecycle, and Compliance agents (Figure 2). Each consumes a validated contract and scoped context, and produces a structured proposal. Proposals identify the requested operation, target, parameters, supporting context revision, and rationale. The harness checks the machine-readable fields independently of the rationale.

1) Executive Agent: The Executive Agent translates operator intent into a descriptor and a workflow proposal. Its inputs are the request, the supported descriptor schema, the available capability catalogue, and applicable operator policies. Its outputs are a candidate contract, a decomposition into supported tasks, and an explicit account of missing or unsupported requirements. As illustrated in (Figure 4), concepts extracted from the request are mapped to known observables, actionables, resources, and constraints. For example, a request mentioning both power draw and latency must populate both requirements or identify why one cannot be represented. The agent must not invent omitted limits or substitute an optimisation preference for a hard constraint. The BoM ontology supports this mapping by describing recognised capabilities and their relationships. Proposed ontology extensions require a separate validation and approval process; an agent cannot extend its own execution authority by introducing a new

concept.

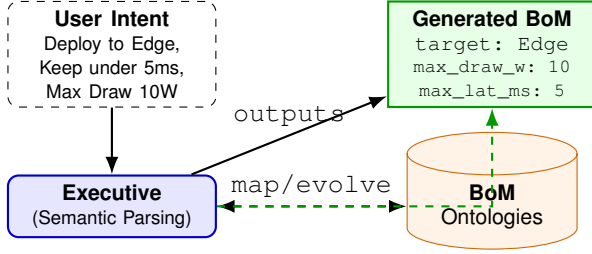


Fig. 4: Intent interpretation and deployment contract validation

2) Placement Agent: Receives the workload’s resource requests and eligible regions, together with timestamped context describing available CPU, memory, load, and latency. It first excludes infeasible candidates and then ranks the remaining alternatives. Let j denote a workload, n a candidate execution location, and t the decision time. Let $\mathcal{R}_j$ be the permitted locations, c_j and m_j the requested CPU and memory, and $a_{\max}$ the maximum context age. The feasible set is:

$$\mathcal{F}_j(t) = \{n \in \mathcal{R}_j \mid c_n^{\text{free}} \geq c_j, \quad m_n^{\text{free}} \geq m_j, \\ 0 \leq t - t_n^{\text{obs}} \leq a_{\max}, \quad H_j(n, t) = 1\}, \quad (1)$$

where H_j represents the remaining applicable checks, including hardware compatibility, permissions, and supported operational constraints. For inference, the latency estimate must additionally satisfy

$$\hat{\ell}_n \leq L_j^{\max} \quad (2)$$

The estimate must refer to the path and statistic specified by the contract. Its use for admission does not establish that the physical network will meet the requirement; runtime measurements remain necessary. Inference locations can be selected by increasing modelled latency, using load and region identifier to break ties. For training, it minimises

$$S(n) = u_n + \frac{\hat{\ell}_n}{100 \text{ ms}}, \quad n \in \mathcal{F}_j(t), \quad (3)$$

where u_n is the reported load normalised to $[0, 1]$. This transparent heuristic balances load and locality; it is not claimed to be an optimal scheduler. Each candidate can be represented as a labelled execution region, and latency and available capacity are scenario inputs. An optional LLM may propose a location, but the proposal passes the same feasibility checks as the reference policy. Context freshness and revision are checked again before admission. If no feasible candidate exists, the request is rejected or deferred without weakening the contract. Under edge resource pressure, this mechanism can place a new training workload in the cloud while retaining inference near the RAN.

3) Lifecycle Agent: This agent consumes the active model version, observed quality, training schedule, cooldown state, and workflow history. Its decision is to retain the current model, request training, or escalate an unsupported condition. Retraining becomes eligible when observed accuracy falls below the configured threshold, or when a scheduled or manual trigger is received, provided that the cooldown has elapsed. This rule assumes that a suitable accuracy estimate is available. Obtaining labels and estimating drift in an operational network are separate concerns; a telemetry platform alone does not establish that concept drift has occurred. A trained candidate is evaluated against the incumbent on an independent validation set. Promotion requires:

$$q_{\text{new}} \geq q_{\min} \quad \wedge \quad q_{\text{new}} - q_{\text{old}} \geq \Delta q_{\min}. \quad (4)$$

The evaluated configuration uses $q_{\min} = 0.90$ and $\Delta q_{\min} = 0.02$. A candidate that satisfies these quality conditions must also become ready before serving traffic. Training failure, insufficient quality, or deployment failure retains the incumbent. The previous ready model is retained for rollback.

4) Compliance Agent: This agent reviews proposals against the application contract, operator policy, current context, and outstanding actionable requests. Its advisory output identifies policy concerns or unresolved conflicts. Authoritative checks are implemented independently in the harness and registered adapters. For an actionable request, validation checks the application scope, permitted operation, target, parameter bounds, and conflicting requests. Requests affecting the same hook are serialised. When applications propose incompatible changes, the affected hook is suspended pending an applicable deterministic policy or a Human-in-the-Loop (HITL) decision. The last accepted state remains in place during this suspension. For example, an energy- and performance-management application may request incompatible changes to the same equipment configuration. Semantic reasoning may help explain their objectives, but it cannot establish that either request is safe. Conflict resolution must remain subject to the declared permissions and physical bounds. A human decision (Figure 5) identifies the conflict and selected resolution, expires after a bounded interval, and is revalidated before execution. Approval cannot override hard parameter limits or expand the application’s capabilities. This mechanism addresses detected conflicts over declared hooks; it does not prove the absence of interactions between different hooks or guarantee global closed-loop stability.

5) Agent Interaction: The Executive Agent establishes the candidate contract. The Lifecycle Agent identifies an eligible operation, the Placement Agent proposes a feasible location, and the Compliance Agent reviews the proposal. The harness then validates and executes the admitted operation. Observed outcomes update the context and workflow history. Each stage can reject or defer a proposal. Retrieving additional context may support a revised proposal, but repeated reasoning cannot bypass an earlier hard constraint. This separation allows

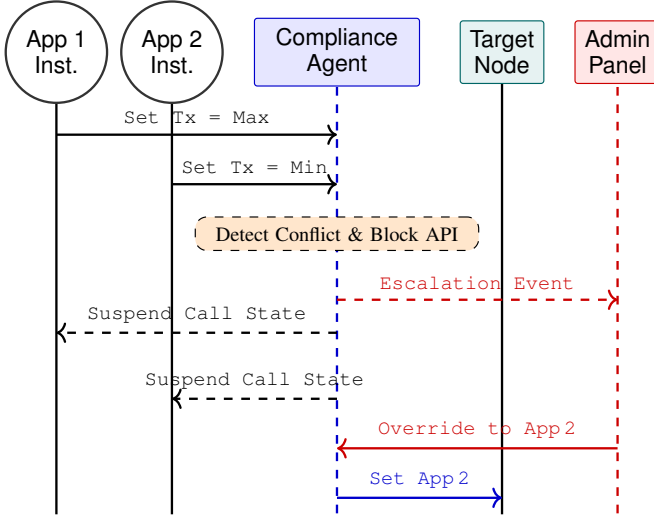


Fig. 5: Actionable conflict detection, suspension, and policy-constrained human resolution.

deterministic policies and optional language-model reasoning to use the same execution boundary.

E. Context Management

Agentic-harness relies on the availability of accurate, relevant, and up-to-date context to perform context-aware tasks effectively. To address this requirement, the framework uses a CoMaS [11] as its knowledge-plane component. It distinguishes authoritative policy and current operational observations from historical memory and generated explanations. Context records carry source provenance, timestamps, scope, and revisions so that the harness can determine whether the evidence supporting a proposal remains applicable.

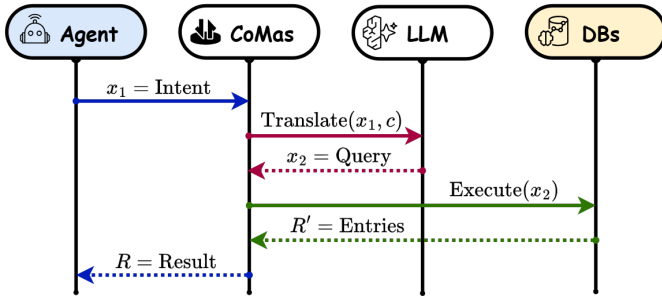


Fig. 6: Context augmentation and validated query execution

1) Context Retrieval: The retrieval process builds on the context-aware mechanism in [12]. As illustrated in (Figure 6), an agent submits a request x_1 , which is augmented with relevant context c . An optional LLM transforms (x_1, c) into a structured query x_2 . Before execution, the query is checked against permitted data sources, fields, tenant scope, and result bounds. The retrieval engine returns entries R' , which are filtered and assembled into the response R . Retrieved text

is treated as evidence rather than as an instruction that can alter the agent's authority. Missing values remain explicit; contextual augmentation must not manufacture measurements or conceal uncertainty. Missing, expired, or revised evidence prevents admission when that evidence is required by the operation. A proposal that was feasible when reasoning began may therefore be rejected when reasoning finishes.

F. Operational Execution and Validation

To systematically address the extreme heterogeneity of telecommunications networks, the proposed framework models ML workloads as dynamically deployable components operating across a distributed execution topology. Rather than assuming a fixed placement, workloads are continuously evaluated against their latency requirements, resource constraints, and relationships with the observables and actionables declared by the RichML-App. This enables the orchestration layer to distinguish between latency-sensitive inference services and more elastic workloads such as model training.

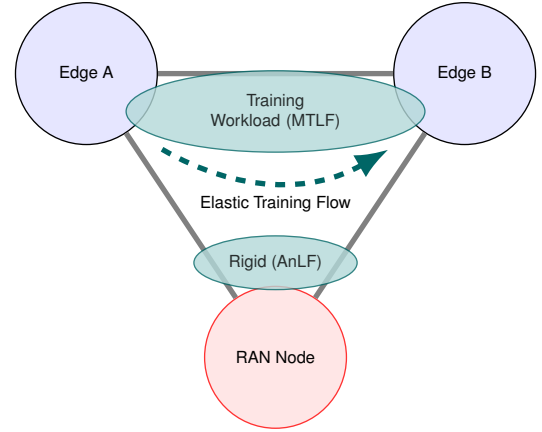


Fig. 7: Separation of inference and elastic training across resources

Inference workloads are therefore treated as comparatively rigid components, as their execution location is strongly coupled to the physical location of the corresponding network observables and action hooks. Moving such workloads to a distant node may introduce latency that violates the application's operational constraints. In contrast, training workloads generally provide greater scheduling flexibility and can be migrated towards available edge or cloud resources when local capacity becomes constrained. This distinction allows the Placement Agent to exploit temporarily available resources without disrupting latency-sensitive inference services. The execution process is governed by the declared constraints of the RichML-App and the policies enforced by the Compliance Agent. Before an operation is executed, the generated action is validated against the application's declared capabilities, resource requirements, deployment constraints, and applicable network policies. Actionables consequently act as explicit execution boundaries, preventing an agent from directly invoking arbitrary infrastructure operations. This provides a controlled

interface between agent reasoning and the underlying network infrastructure. When multiple workloads compete for the same resources or attempt to modify the same actionable, the Compliance Agent evaluates the corresponding policies and priorities before execution. Conflicting operations are temporarily suspended rather than executed concurrently, allowing the framework to avoid unstable or oscillatory behaviour. Where conflicts cannot be resolved automatically, the execution is escalated through the hitl mechanism, preserving operator authority over critical network actions.

Following validation, the execution harness translates the approved operation into the corresponding infrastructure-level api calls. The resulting state is subsequently monitored through asynchronous polling and network telemetry, allowing the framework to determine whether the requested operation was successfully applied. This closes the execution loop between reasoning, validation, deployment, and observation, enabling subsequent agent decisions to be based on the updated operational context. The resulting mechanism provides a controlled form of workload elasticity: non-latency-critical workloads can dynamically expand or migrate across available infrastructure, while latency-sensitive services remain bounded by their physical and operational constraints. In this way, the framework combines the adaptability required by distributed 6G infrastructures with the deterministic execution and governance mechanisms required for safe operation.

IV. CONCLUSION AND FUTURE WORK

In this paper, we presented a context-aware MLOps framework for managing ML workloads across heterogeneous 5G/6G infrastructures. The proposed architecture combines deterministic workflow execution with agent-assisted reasoning, enabling ML services to adapt to changing network conditions while preserving explicit operational boundaries. The decomposition into specialised agents for intent interpretation, placement, lifecycle management, and compliance provides a structured approach for coordinating distributed ML workloads across cloud, edge, and RAN environments. The RichML-App further provides an operational contract between ML applications and the orchestration framework by explicitly defining observables, actionables, resource requirements, placement constraints, and lifecycle properties. Combined with the Context Management System, this enables agents to reason over current network conditions and operational knowledge rather than relying solely on static deployment specifications. Furthermore, the separation between agent reasoning and deterministic execution, together with compliance validation and HITL escalation, provides a mechanism for enabling autonomous operation without removing operator control from critical network actions.

A. Future Work

As future work, the framework will be implemented and evaluated within a distributed Kubernetes-based 6G testbed. The evaluation will focus on dynamic workload placement,

lifecycle adaptation, and the responsiveness of agents to changing telemetry, resource availability, and model drift. The overhead introduced by agent reasoning and context retrieval will also be investigated in terms of latency and resource consumption. Further work will address more formal governance and conflict-resolution mechanisms for competing ML applications, including policy constraints, action permissions, and tenant isolation. Finally, the integration of RichML-App observables and actionables with standardised 3GPP functions such as the MTLF, AnLF, and NWDAF will be investigated to establish a more general mechanism for autonomous, context-aware ML lifecycle management in future 6G networks.

ACKNOWLEDGMENT

The authors gratefully acknowledge the financial support provided by the German *Federal Ministry of Education and Research (BMFTR)* through the projects Open6GHub+ {16KIS2402K} and KIOps6G {16KIS2398}.

REFERENCES

- [1] B. Wu, S. Wang, Y.-Q. Zhang, J. Sifakis, and Y. Ouyang, "Leveraging ai agents for autonomous networks: A reference architecture and empirical studies," *IEEE Communications Magazine*, 2026.
- [2] E. C. Strinati, S. Barbarossa, J. L. Gonzalez-Jimenez, D. Ktenas, N. Cassiau, L. Maret, and C. Dehos, "6g: The next frontier: From holographic messaging to artificial intelligence using subterahertz and visible light communication," *IEEE Vehicular Technology Magazine*, vol. 14, no. 3, pp. 42–50, 2019.
- [3] M. Yu, Y. Xing, X. Xia, and J. Jia, "Ai agent based autonomous cognitive architecture for 6g core network," in *2025 International Wireless Communications and Mobile Computing (IWCMC)*. IEEE, 2025, pp. 538–542.
- [4] M. P. J. Kuranage, A. Mazene, A. Mekrache, and A. Ksentini, "End-to-end ai lifecycle management for 6g: Bridging dataops and mlops," *IEEE Networking Letters*, 2026.
- [5] F. Rezazadeh, H. Chergui, L. Alonso, and C. Verikoukis, "Sliceops: Explainable mlops for streamlined automation-native 6g networks," *IEEE Wireless Communications*, vol. 31, no. 5, pp. 224–230, 2024.
- [6] R. 3GPP TS 23.288, "Architecture enhancements for 5g system (5gs) to support network data analytics services."
- [7] P. K. Gkonis, N. Nomikos, P. Trakadas, L. Sarakis, G. Xylouris, X. Masip-Bruin, and J. Martrat, "Leveraging network data analytics function and machine learning for data collection, resource optimization, security and privacy in 6g networks," *IEEE access*, vol. 12, pp. 21 320–21 336, 2024.
- [8] P. Steinberger and OpenAI. Openclaw. [Online]. Available: <https://docs.openclaw.ai/>
- [9] N. Research. The agent that grows with you. [Online]. Available: <https://hermes-agent.nousresearch.com/>
- [10] C. Hang, P. Yu, R. Morabito, and C. Tan, "Large language models meet next-generation networking technologies: A review, future internet 16.10 (2024) 365."
- [11] F. Poughela, D. Krummacker, and H. D. Schotten, "Towards 6G Networks," in *A Context Management Architecture for Decoupled Acquisition and Distribution of Information in Next-Generation Mobile Networks*, ser. ITG, vol. 157, VDE. IEEE, 5 2023. [Online]. Available: <https://www.researchgate.net/publication/373328855>
- [12] F. Poughela and H. D. Schotten, "A context-aware retrieval framework for dynamic network management," in *ICC 2026 - IEEE International Conference on Communications*, 2026, pp. 1–6.