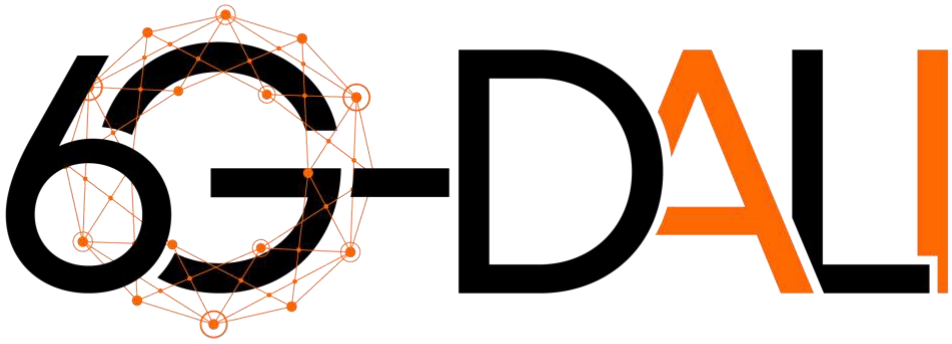
29th JULY, 2026



Initial Report on Experimentation on Demand, FLOps, Transfer Learning, and Interaction with Testbeds

Deliverable 4.1, Work Package 4

Co-funded by the European Union

6GSNS

6G-DALI project has received funding from the Smart Networks and Services Joint Undertaking (SNS JU) under the European Union's Horizon Europe Research and Innovation programme under Grant Agreement No 101192750. Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the SNS JU. Neither the European Union nor the SNS JU can be held responsible for them.

Project Identifier	HORIZON-JU-SNS-2024-STREAM-B-01-08, Project 101192750		
Project Name	6G DAta and ML operations automation via an end-to-end AI framework		
Acronym	6G-DALI		
Start Date	01 January 2025	End Date	31 December 2027
Project URL	www.6gdali.eu		
Deliverable	D4.1 Initial report on experimentation on demand, FLOps, transfer learning, and interaction with testbeds		
Work Package	WP4 'Experiments on demand and interaction with testbeds'		
Contractual due date	31/05/2026	Actual submission date	29/07/2026
Type	R	Dissemination Level	Public
Lead Beneficiary	VIAVI Solutions (VIA)		
Responsible Author	Mahdi Sharara (VIA)		
Contributors	IBM (Emile Aydar, Christian Pinto, Srikumar Venugopal), ICOM (Vasiliki Parousidou), EUR (Sid Ali Mahmoudi), TUD (Yuandou Wang), ISI/ATH Panagiotis Marantis), IQU (Jeffrey Redondo), NXW (Marco Ruta, Alessandro Stefanelli), KUL (Luisa Schuhmacher), SPW (Dimitrios Amaxilatis)		
Peer reviewer(s)	Menuka Perera (EUR), Giacomo Bernini (NXW)		

Version	**Authors**	**Comments**
1.0	All Contributors	Final Version

Disclaimer

The content of this document reflects only the author's view. Neither the European Commission nor the Smart Networks and Services Joint Undertaking (SNS JU) are responsible for any use that may be made of the information it contains.
While the information contained in the documents is believed to be accurate, the authors(s) or any other participant in the 6G-DALI consortium make no warranty of any kind regarding this material including but not limited to the implied warranties of merchantability and fitness for a particular purpose. Neither the 6G-DALI Consortium nor any of its members, their officers, employees or agents shall be responsible or liable in negligence or otherwise howsoever in respect of any inaccuracy or omission herein.
Without derogating from the generality of the foregoing neither the 6G-DALI Consortium nor any of its members, their officers, employees or agents shall be liable for any direct or indirect or consequential loss or damage caused by or arising from any information advice or inaccuracy or omission herein.

Copyright message

©6G-DALI Consortium. This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both. Reproduction is authorised provided the source is acknowledged.

Table of Contents

Table of Contents 4
Glossary of terms and abbreviations used 10
Executive Summary 12
1 Introduction 14
1.1 Scope and Purpose of This Deliverable 14
1.2 Mapping 6G-DALI outputs 14
1.3 Deliverable overview and report structure 16
2 Experimentation on Demand, Automated Model Tuning, FLOps, and Digital Twin Testbed in 6G-DALI 17
2.1 6G-DALI Architecture and WP4 Positioning 17
2.2 Contributions to 6G-DALI innovations, technical objectives and KPIs 19
2.2.1 Innovation Area KPI Status 19
2.2.2 Objectives KPI Status 20
3 On-demand Dataset Generation 24
3.1 Overview 24
3.2 Intent-based data generation 24
3.2.1 Problem statement and approach 24
3.2.2 Semantic scenario matching algorithm 26
3.2.3 Experiment execution pipeline 27
3.2.4 Initial results 29
3.2.5 Next steps 30
3.3 Surrogate Based Data Generation for Budget-Aware Testbed Experimentation 30
3.3.1 Project Context and Architectural Positioning 31
3.3.2 State-of-the-Art Analysis 33
3.3.3 Preliminary Pipeline and Evaluation 34
3.3.4 Next Steps 39
3.4 Conclusion 40
4 Automated Model Tuning Across the ML Lifecycle 42
4.1 Overview 42
4.2 ADO for HPO in MLOps 42
4.2.1 State-of-the-Art 43
4.2.2 Solution description 43
4.2.3 Initial Results 45
4.2.4 Next Steps 48
4.3 Automated Reinforcement Learning and RLOps 48
4.3.1 State-of-the-Art 49
4.3.2 Solution Description 49

4.3.3 Initial Results 51
4.3.4 Next Steps 52
4.4 Cooperative Transfer Learning 52
4.4.1 State-of-the-art 52
4.4.2 Solution Description 52
4.4.3 Initial Results 54
4.4.4 Next Steps 54
4.5 Conclusion 54
5 Distributed Training and Workload Execution on 6G Testbeds 55
5.1 Overview 55
5.2 AI/ML Workload Placement 55
5.2.1 Motivation behind the Implementation 55
5.2.2 The Framework for GPU inference Workload Placement 55
5.2.3 State-of-the-Art 56
5.2.4 Solution Description 56
5.2.5 Live System Integration 58
5.2.6 Initial Results 59
5.2.7 Next Steps 61
5.3 Federated Learning Operations (FLOps) 61
5.3.1 NWDAF and FLOps 61
5.3.2 FL Algorithms for PHY and JCAS 64
5.3.3 Federated Learning Algorithms 66
5.3.4 Split Learning 70
5.4 Conclusion 75
6 Digital Twin Testbed for Large-Scale Experimentation 76
6.1 Overview 76
6.2 AI-RSG and FlexRIC Integration 76
6.2.1 State-of-the-art 76
6.2.2 End-to-End Workflow 77
6.2.3 KPM and RC xApps (EUR) 78
6.3 Large-Scale Data Generation 79
6.4 RLOps and DT 79
6.4.1 State-of-the-art 79
6.4.2 ZenML-based RLOps Pipeline 81
6.4.3 Current Status of Integration 82
6.4.4 Integration Results 82
6.4.5 Next Steps 82
7 Ethics Requirements Mapping 83

8 Conclusions and Next Steps 84
References 85
Annex I Ethics Requirements Mapping 92
I-A Privacy and Data Governance 92
I-B Accountability 92
I-C Reliability, Robustness, Resilience 93
I-D Autonomy 94
I-E Transparency 95
I-F Fairness 96
I-G Societal & Environmental Well-Being 97
I-H Epistemic Integrity 98
I-I Bias 100

List of Figures

Figure 1: 6G-DALI Architecture; boxes in blue highlight the focus of the deliverable 17
Figure 2: Intent-based agentic data generation framework architecture 25
Figure 3: Sequence diagram of data generation 28
Figure 4: User intent to experiment configuration translation 29
Figure 5: Users' selected experiment execution 29
Figure 6: Generated dataset presentation after the end of experiment execution 30
Figure 7: Surrogate Construction Workflow from Testbed Runs during an Experiment Campaign 32
Figure 8: Predictive Kernel Herding (PKH) sampling strategy – Selecting the most under-covered testbed runs in a feature space learned by the surrogate model instead of the raw testbed parameter space 35
Figure 9: PKH fills RF leaf deficits to match pool mass. 36
Figure 10: Prediction Error (in MAPE) of each iterative strategy in function of the train set size (in % of dataset) for AMF Performance Dataset. 38
Figure 11: Superposition of ground truth response error histogram (in grey) against the empirical error histogram (in cyan) for each sampling strategy at final measurement budget (10% of dataset size) for AMF Performance Dataset and lat50 objective 38
Figure 12: Prediction Error (in MAPE) of each iterative strategy in function of the train set size (in % of dataset) for Golang Webserver dataset 39
Figure 13: Superposition of ground truth response error histogram (in grey) against the empirical error histogram (in cyan) for each sampling strategy at final measurement budget (10% of dataset size) for Golang Webserver Dataset and lat50 objective. 39
Figure 14: MLOps components functional decomposition - ADO for HPO 42
Figure 15: ADO for HPO high level architecture 44
Figure 16: ADO-HPO (left) search space, experiment definition and (right) operation definition 45
Figure 17: ADO-HPO proof-of-concept results 45
Figure 18: HPO SDK Example 46
Figure 19: HPO module - Campaign registration 47
Figure 20: HPO module - HPO trial 48
Figure 21: AutoRL/HPO design 50
Figure 22: Convergence of HPO methods across trial order. 51
Figure 23: Cooperative Transfer Learning design 53
Figure 24: System Overview 57
Figure 25: Scheduler Flow 58
Figure 26: Training Convergence 59
Figure 27: Workload Routing per Node Tier for each algorithm 60
Figure 28: Energy Consumption per CEC Tier 60
Figure 29: The proposed service-based NWDAF Release 17 architecture [63] 62
Figure 30: The proposed NWDAF-enabled FLOps Architecture 63

Figure 31: On the left, the setting of the nomadic dataset. The black stars show the user positions, in green the distributed array, and in orange six trajectories of a human walking, each defining a new scenario in which data is collected. For the static scenario, no human is moving. On the right, the trajectory-based train and test splits for subsequent model training. 65

Figure 32: The proposed federated learning algorithm, proposedTopK. 68

Figure 33: Integration of the proposed framework in the DALI architecture. 71

Figure 34: The GUI for controlling all of the available options analyzed in the solution description. 73

Figure 35: All MLflow experiment runs performed. 75

Figure 36: VIAVI Digital Twin Testbed Architecture — End-to-End Integration with 6G-DALI Core Infrastructure 77

Figure 37: A first prototype of an end-to-end RLOps pipeline 81

List of Tables

Table 1. Adherence to 6G-DALI GA Deliverable & Tasks Descriptions 14
Table 2. Innovation Area 2 – Collaborative MLOps and RLOps 19
Table 3. Innovation Area 5 – Digital Twin for large-scale 6G experimentation 19
Table 4. Innovation Area 6 – Knowledge Transfer Automation 19
Table 5. Innovation Area 7 – Federated Learning with NWDAF operations for 6G 20
Table 6. Innovation Area 8 – LLM-enabled AI experimentation and HPO 20
Table 7. Objective 1 – Deliver a user-friendly e2e AI framework for DataOps and MLOps in 6G 20
Table 8. Objective 2 – Deliver Gaia-X and ELT approaches for DataOps in 6G environments 21
Table 9. Objective 3 – 6G testbed's trustworthy AI/ML operations via MLOps and AutoML 21
Table 10. Objective 4 – Deliver pluggable adapters to easily integrate 6G testbeds from future calls 22
Table 11. Objective 5 – Build and integrate a Digital Twin Testbed to generate representative datasets for 6G 22
Table 12. Dataset characteristics 37
Table 13. HPO - High Level API endpoints 44
Table 14. Hyperparameters and corresponding search space. 51
Table 15. Algorithm Evaluation based on the AI-RAN Alliance KPIs 59
Table 16. Performance comparison using trajectory-based data splits. 65
Table 17. Performance comparison under simplified evaluation conditions. 65
Table 18. Average experimental results comparisons on CSI datasets without Byzantine attacks (no attack) 69
Table 19. Average experimental results comparisons on CSI datasets under the random-attack scenario. 69
Table 20. Training settings used for all the experiments 73
Table 21. Non quantized results of the vanilla split algorithm showcasing the different partition effects on training. 74
Table 22. Splitfed 5 clients quantization results directly from the exported logs 74
Table 23. 5 clients splitfed iid run showcasing the dynamic quantization results 75

Glossary of terms and abbreviations used

Abbreviation / Term	Description
3GPP	Third Generation Partnership Project
5G NR	5th Generation New Radio
5GC	5th Generation Core
5QI	5G QoS Identifier
ADK	Application Development Kit
ADO	Accelerated Discovery Orchestrator
AIaaS	AI-as-a-Service
AMF	Access and Mobility Management Function
AoA	Angle-of-Arrival
AP	Access Point
ARM64	64-bit ARM processor architecture (AArch64)
ARPA	AI RAN Placement Agent
ASN.1	Abstract Syntax Notation One
AutoML	Automated Machine Learning
AutoRL	Automated Reinforcement Learning
BOHB	Bayesian Optimization with HyperBand
CA	Carrier Aggregation
CEC	Cloud-Edge Continuum
CM	Configuration Management
CoTL	Cooperative Transfer Learning
CSI	Channel State Information
DataOps	Data Operations
DEHB	Differential Evolution HyperBand
DNN	Deep Neural Network
DRB	Data Radio Bearer
DRL	Deep Reinforcement Learning
DT	Digital Twin
DTT	Digital Twin Testbed
E2AP	E2 Application Protocol
E2SM	E2 Service Model
ELT	Extract-Load-Transform
eMBB	enhanced Mobile BroadBand
FL	Federated Learning
FlexRIC	Flexible RAN Intelligent Controller
FLOps	Federated Learning Operations
GAE	Generalised Advantage Estimation
GBR	Guaranteed Bit Rate
gNB	Next Generation NodeB
GoB	Grid of Beams
HPO	Hyperparameter Optimization
IDS	International Data Spaces
InH	Indoor Hotspot
ISAC	Integrated Sensing and Communication
JCAS	Joint Communication and Sensing
KPI	Key Performance Indicator
KPM	Key Performance Metrics
LLM	Large Language Model
LoS	Line-of-Sight
MAC	Medium Access Control
MAPE	Mean Absolute Percentage Error
MEC	Mobile Edge Computing

MIMO	Multiple-Input Multiple-Output
MLflow	Open-source ML experiment tracking platform
MLOps	Machine Learning Operations
mMIMO	Massive MIMO
mMTC	Massive Machine-Type Communications
NETCONF	Network Configuration Protocol
NLoS	Non-Line-of-Sight
NS3	Network Simulator 3
NWDAF	Network Data Analytics Function
O-CU	O-RAN Central Unit
O-DU	O-RAN Distributed Unit
O-RAN	Open Radio Access Network
OAI	OpenAirInterface
OMNeT++	Objective Modular Network Testbed in C++
PDCP	Packet Data Convergence Protocol
PDU	Protocol Data Unit
PHY	Physical Layer
PKH	Predictive Kernel Herding
PLMN	Public Land Mobile Network
PPO	Proximal Policy Optimization
PRB	Physical Resource Block
QoS	Quality of Service
RAN	Radio Access Network
RC	RAN Control
RIC	RAN Intelligent Controller
RIS	Reconfigurable Intelligent Surface
RL	Reinforcement Learning
RLC	Radio Link Control
RLlib	Ray RLlib
RLOps	Reinforcement Learning Operations
RMa	Rural Macro
RRC	Radio Resource Control
SLICE	Network Slicing
SMAC	Sequential Model-based Algorithm Configuration
TC	Traffic Control
TL	Transfer Learning
UE	User Equipment
UMa	Urban Macro
UMi	Urban Micro Street Canyon
URLLC	Ultra-Reliable Low Latency Communications
ZenML	Open-source MLOps framework

Executive Summary

The scope of Work Package 4 in the 6G-DALI project is to provide the automated experimentation pipelines that sit between the 6G-DALI Data Space and deployable AI models, covering on-demand dataset generation, automated model tuning, distributed and federated training, and large-scale digital twin simulation. This deliverable reports the initial design, implementation and results of these four tracks. For each track, a zoom-in is performed into its constituent components, covering problem statement, design rationale, and initial results. In this deliverable, components are primarily treated as stand-alone entities, with focus on design and preliminary implementation. All reported components have been designed with the necessary integration interfaces, ensuring a smooth end-to-end integration process planned for next phases of the project.

On-demand dataset generation establishes the mechanisms by which experimenters obtain datasets for AI/ML model development without exhaustive testbed sweeps. It is realized through two complementary components:

1- An intent-driven data generation pipeline, which translates natural-language experiment requests into automated testbed executions via a cooperative ensemble of LLM-based agents, shielding users from the complexity of individual testbed interfaces
2- A surrogate-based data generation component, which uses a Random Forest surrogate model to identify which testbed configurations will most improve dataset coverage. It actively selects the most informative runs under a bounded number of measurement runs rather than sampling at random.

Key innovations with respect to existing approaches include the application of LLM-based agent orchestration for end-to-end experiment automation across heterogeneous testbed APIs, and the reformulation of active sampling as a histogram-matching problem in the surrogate's learned feature space, enabling computationally efficient selection without sacrificing global representativeness of the dataset. Both components have reached a prototype status, with initial evaluation results obtained on real testbed datasets.

Automated model tuning covers the hyperparameter optimization and policy search layer that operates on the datasets. It is realized through three components. One component is an HPO actuator that is integrated into the 6G-DALI MLOps pipeline based on the Accelerated Discovery Orchestrator transfers knowledge across experiments rather than treating each run in isolation. Another component is an AutoRL benchmarking framework that evaluates automated hyperparameter optimization methods on a realistic 6G resource allocation task to identify the most effective strategies for reinforcement learning in this domain. A third component is a Cooperative Transfer Learning framework that enables independently trained models to share knowledge components through a partition-and-merge strategy, improving performance without requiring direct data exchange between parties. Key innovations include the cross-experiment knowledge reuse mechanism embedded in the HPO actuator, and the elevation of cooperative transfer learning from an algorithmic concept to a managed service within the MLOps pipeline. The HPO actuator and AutoRL benchmarking components have been prototyped and tested; the cooperative transfer learning pipeline design is complete and implementation is underway.

Distributed training and federated learning operations establish the mechanisms through which optimized models are deployed and executed across the 6G-DALI testbed federation. It is realized through four components. One component is a reinforcement learning-based workload scheduler for GPU inference placement that, in initial simulation, achieved a 25% reduction in energy consumption relative to round-robin assignment, using PPO with action masking over 80,000 training steps. Another component is a federated gradient aggregation algorithm combining sparsification, Byzantine filtering, and differential privacy noise injection. In initial evaluation, it reduced communication volume by over 93% while preserving model accuracy. A third component is a federated learning algorithm for joint communication and sensing positioning, evaluated across seven propagation scenarios. In initial results, it showed a 30% improvement in positioning accuracy relative to purely local training. Moreover, a fourth component is a split learning framework that achieved a 75% reduction in data transmission under quantization in initial experiments, with an adaptive scoring mode providing further savings. Key innovations with respect to existing federated learning approaches include the joint application of

gradient sparsification and Byzantine filtering within a single aggregation step, the integration of privacy-preserving noise injection without accuracy degradation, and the extension of the split learning framework with dynamic composite scoring to adaptively balance communication cost against model quality. All four components are at initial prototype or early validation stage; full evaluation against the complete 6G-DALI dataset collection and live testbed integration are planned for the next deliverable D4.2.

The Digital Twin Testbed provides the large-scale simulation environment required for experiments at scales that physical radio testbeds cannot support. It is built around the VIAVI AI-RSG platform, which generates synthetic radio environments across a wide range of traffic types, propagation conditions, and Massive MIMO configurations, without requiring physical hardware at that scale. Integrated with an open-source near-RT RIC over standard O-RAN E2 interfaces, the platform under development supporting both KPM data collection and RAN control actions. A reinforcement learning operations pipeline has been implemented using ZenML, RLlib, and MLflow.It covers the RL lifecycle including the environment definition, training, evaluation, and experiment tracking. While the RLOps should be validated on the digital twin, the full integration of the DT's components is not finalized yet. Hence, the RLOps pipeline has been validated in a separate simulation environment based on OMNeT++. Key innovations include the ADK pacing mechanism enabling step-synchronized RL policy training against the digital twin, and the end-to-end O-RAN-compliant data path from simulated RAN measurements to the 6G-DALI Data Space. Full closed-loop integration between the RLOps pipeline and the Digital Twin Testbed is the primary open item for D4.2.

All components reported in this deliverable will be further developed, refined, and integrated in the upcoming project phase. The work reported here is at an early stage. Each area has a clear path forward. On data generation, the PKH sampling strategy will be integrated into ADO and connected to the live experimentation pipeline. On automated model tuning, the HPO actuator will be connected to the MLOps Meta-Orchestrator. The AutoRL framework will expand to cover algorithm selection and architecture search. The Cooperative Transfer Learning framework will move from design into implementation. On distributed training, the workload placement framework will be tested in a live Kubernetes environment. The federated learning algorithm will be completed and deployed on NWDAF and FLOps infrastructure. The split learning work will extend to study energy efficiency across different deployment configurations. Finally, the RLOps pipeline will be validated on the Digital Twin. These developments will be reported in D4.2.

1 Introduction

Training and deploying AI systems for 6G networks demands a continuous supply of representative data, tuned models, and distributed training infrastructure, each of which imposes costs that do not scale gracefully with the diversity of 6G deployment scenarios. A single physical testbed may not simultaneously reproduce the UE densities, spectrum conditions, traffic heterogeneity, and failure modes that production 6G networks encounter, which means ML models validated only under controlled hardware conditions carry unknown generalization gaps. Work Package 4 of the 6G-DALI project addresses this constraint by building the infrastructure that makes on-demand, cost-efficient ML experimentation possible across four interconnected dimensions: automated data generation, hyperparameter and policy optimization, distributed training under communication and privacy constraints, and large-scale digital twin simulation.

D4.1, the first report from WP4, documents initial work across Tasks 4.1 through 4.3, corresponding to milestones MS8 (Initial approaches for experiments on demand), MS9 (Initial adapters design), and MS10 (Initial DT for large-scale experiments design), as defined in the 6G-DALI Grant Agreement.The contributions reported here are preliminary, with experiments running and implementations at varying stages of completion. D4.2 will consolidate and extend these initial results into full evaluations against the 6G-DALI dataset collection and live testbed environments.

1.1 Scope and Purpose of This Deliverable

This deliverable covers the activities of Tasks 4.1, 4.2, and 4.3 of Work Package 4, as defined in the 6G-DALI Grant Agreement. This deliverable serves as the verification means for Milestones MS8 (Initial approaches for experiments on demand), MS9 (Initial adapters design), and MS10 (Initial DT for large-scale experiments design).

1.2 Mapping 6G-DALI outputs

The purpose of this section is to map 6G-DALI Grant Agreement commitments, both within the formal Deliverable and Task description, against the project's respective outputs and work performed.

Table 1. Adherence to 6G-DALI GA Deliverable & Tasks Descriptions

Deliverable description			
D4.1 Initial report on experimentation on demand, FLOps, transfer learning, and interaction with testbeds: Initial report on the activities on experimentation on demand, FLOps, transfer learning and DTT. It will include initial algorithms and mechanisms with preliminary results.			
6G-DALI Task No. and Title	**6G-DALI GA Task Description**	**Section**	**Justification**
Task 4.1 Experimentation on demand	This task aims to develop the Hyperparameter Optimization and placement component of the AIaaS experimentation framework, offering experiment design, dataset generation, model validation, and hyperparameter optimization of AI/ML models on demand building on AutoML principles.	**3Section 3 On-demand Dataset Generation**	Presents the intent-driven LLM-agent pipeline for translating natural-language user intents into experiment configurations, and the budget-aware surrogate sampling approach for reducing the cost of acquiring 6G-representative training data.
		Section 4 Automated Model Tuning Across the ML Lifecycle	Describes the hyperparameter optimisation automation approach, the automated reinforcement learning and RLOps tooling, and the cooperative transfer learning pipeline for efficient model reuse across 6G tasks.
		Section 5.2-5.3	Presents the reinforcement learning-based workload

		AI/ML and RL Workload Placement	placement framework for GPU inference scheduling and the edge-deployed RL workload placement approach.
Task 4.2 Federated and Transfer learning	This task will contribute algorithmic innovation in the form of novel Federated Learning and Transfer Learning algorithms, and implement and integrate the most promising of these algorithms within the NWDAF-FL and Cooperative Transfer Learning functions.	**Section 4.4 Cooperative Transfer Learning**	Presents the Cooperative Transfer Learning (CoTL) framework, where independently trained ML models share reusable knowledge units to improve each other without retraining from scratch. Covers the CoTL architecture integrated into the 6G-DALI MLOps platform, including the Knowledge Transfer Orchestrator, Model Catalogue, and Knowledge Catalogue, as well as the ZenML-based pipelines for model registration, knowledge extraction, and knowledge-driven model improvement across 6G task contexts.
		Section 5.3 Federated Learning Operations (FLOps)	Describes the federated learning algorithms for privacy-preserving aggregation, the NWDAF-integrated FLOps architecture targeting 3GPP Release 18, the federated learning approach for PHY-layer and JCAS localisation, and the split learning quantisation framework.
Task 4.3 DT for large-scale experimentation	This task aims to develop the DTT of the RAN and CN using OpenAirInterface (OAI) and VIAVI testing tools for testing and validating large-scale experiments.	**Section 6.2 AI-RSG and FlexRIC Integration**	Presents the ongoing integration of the AI-RSG tool with the FlexRIC open-source RAN Intelligent Controller over the E2 interface, including KPM and RC xApp operation.
		Section 6.3 Large-Scale Data Generation	Describes the large-scale data generation capability using the AI-RSG simulation environment, enabling dataset creation at scales not reproducible on physical testbeds.
		Section 6.4 RLOps and DT	Details the RLOps pipeline built on ZenML and OMNeT++ advancing toward full integration with the digital twin testbed.

1.3 Deliverable overview and report structure

This deliverable is structured to provide an initial report on the activities related to experimentation on demand, federated learning operations, transfer learning, and interaction with testbeds within the 6G-DALI framework, detailing their design, architecture, and initial results.

Section 1 introduces the deliverable, establishes its scope and mapping to 6G-DALI objectives and KPIs, and outlines the document structure.

Section 2 reports the status of 6G-DALI innovation areas and objectives against their target KPIs.

Section 3 presents on-demand dataset generation approaches, covering intent-driven LLM-agent pipelines and surrogate-based sampling methods that reduce the cost of acquiring 6G-representative training data.

Section 4 describes automated model tuning across the ML lifecycle, including hyperparameter optimization approaches, automated reinforcement learning and RLOps tooling, and cooperative transfer learning pipelines.

Section 5 reports distributed training and workload execution results, spanning AI/ML workload placement frameworks, privacy-preserving federated learning algorithms, NWDAF-integrated FLOps architectures, federated learning for PHY-layer and JCAS localisation, and split learning quantization.

Section 6 documents the digital twin testbed for large-scale experimentation, covering the integration of AI-RSG with FlexRIC, large-scale data generation capabilities, and the RLOps pipeline advancing toward full testbed integration.

Section 0 concludes the deliverable by summarising KPI progress across the covered objectives, identifying work remaining in progress, and specifying the steps planned for D4.2.

2 Experimentation on Demand, Automated Model Tuning, FLOps, and Digital Twin Testbed in 6G-DALI

2.1 6G-DALI Architecture and WP4 Positioning

This chapter provides a technical introduction to position the work reported in this deliverable within the initial version of the 6G-DALI architecture described in Deliverable D2.3 Initial framework architecture [1] and reported in Figure 1 below. This deliverable, as depicted in the Figure, focuses on a subset of the 6G-DALI architecture, namely on the Experimentation on Demand, Automated Model Tuning, Distributed Training and FLOps, and Digital Twin Testbed.

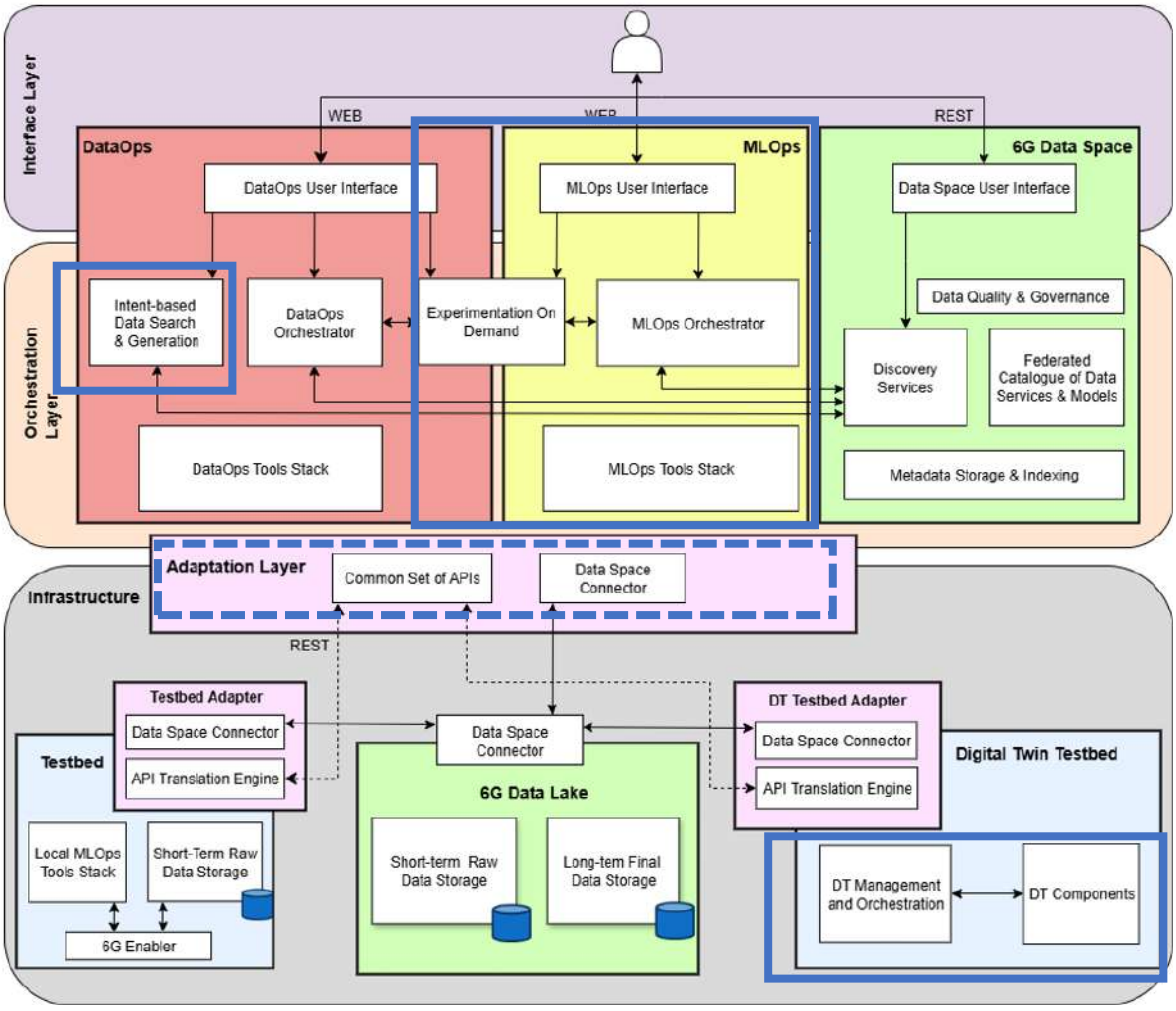


Figure 1: 6G-DALI Architecture; boxes in blue highlight the focus of the deliverable

The 6G-DALI architecture, shown in Figure 1, is structured into four main layers: the Interface Layer, the Orchestration Layer, the Adaptation Layer, and the Infrastructure Layer. Within these layers, seven architectural blocks are identified: DataOps, MLOps, 6G Data Space, Adaptation Layer, Testbeds, Digital Twin Testbed, and 6G Data Lake. While the broader architectural principles have been defined in WP2 and the initial architecture design is reported in deliverable D2.3, the technical implementations detailed in this deliverable map directly to specific blocks within the 6G-DALI architecture, as follows.

The **DataOps** block in 6G DALI is the block responsible for foundational mechanisms for managing, processing, and optimizing data workflows across distributed 6G environments. It is a core architectural block enabling seamless orchestration of data collection, transformation, and provisioning process that underpins AI experimentation and data-driven analytics within the 6G-DALI framework. Two components within DataOps block are considered in this deliverable: **Experimentation on Demand** and **Intent-based Data Search & Generation.** The **Experimentation on Demand** component sits at

the boundaries of the DataOps and MLOps blocks. It is the orchestration-layer component through which experimenters express data experiment requirements. For data experiments, requirements can be expressed via a natural language intent-based interface, implemented by the other DataOps relevant component called **Intent-based Data Search & Generation**. This component translates user natural language-based intents into testbed-specific experiment configurations. When datasets requested by an experiment are not available, they can be generated on demand by a surrogate model-based dataset generation solution implemented within the **Experimentation on Demand** component. Section 33.2 presents the Intent-based data generation solution, and section 3.3 presents the Surrogate-based data generation solution.

The **MLOps** block is the 6G-DALI's centralized lifecycle-management block for AI/ML. It handles model cataloguing, training, deployment, and monitoring. It's extended with HPO, Federated Learning, Reinforcement Learning, Transfer Learning, and Trustworthy AI, and can run centrally or distributed across testbeds via meta-orchestration. As Figure 1 shows, the MLOps block is composed of multiple sub-components that are of interest to this deliverable. Of main importance is the **MLOps Tools Stack** component. Within the **MLOps Tools Stack** component, their exists the Automated Model Tuning, Distributed Training, and Federated Learning Operations solutions. **Automated Model Tuning** consists of two solutions presented in section 4**.** One solution is the automated reinforcement learning (AutoRL) presented4.3, and the other is the cooperative transfer learning presented in section4.4. These capabilities enable efficient, knowledge-reusing optimisation of AI/ML models trained on 6G data, reducing manual effort and improving reproducibility across experiment cycles. In addition to the Automated Model Tuning, the MLOps Tools Stack includes the **Distributed Training and Federated Learning Operations (FLOps)** component. This component addresses the distributed nature of 6G-DALI by enabling AI/ML model training workflows to be executed across heterogeneous Cloud-Edge-Continuum nodes. The FLOps framework manages federated and distributed training pipelines, aggregation strategies, and workload scheduling across the infrastructure layer. This component comprises multiple solutions presented in section 5. The AI/ML Workload Placement is presented in section 5.2, and a set of Federated Learning Operations (FLOps) solutions is presented in section 5.3. This section presents NWDAF-enabled FLOps in section 5.3.1. Then multiple FL algorithms are presented in Sections 5.3.2 and 5.3.3. Finally, a Split learning framework for efficient distributed model training is presented in section 5.3.4. On the other hand, the **Experimentation on Demand**, which was mentioned earlier as a component spanning both **MLOps and DataOps,** includes a solution under the **MLOps** scope; that is the Automated Hyperparameter Optimisation (HPO) and it is presented in section 4.2.

Another important block is the **Digital Twin Testbed (DTT)**. The DTT provides a high-fidelity, large-scale experimentation environment combining the VIAVI AI-RSG simulation platform and the OpenAirInterface FlexRIC RAN Intelligent Controller. The DTT enables O-RAN-compliant experiment scenarios, KPI collection, and closed-loop control experiments without requiring access to physical infrastructure. Within the DTT, the **DTT components** block includes the FlexRIC and AI-RSG explained in section 6.2, and it also includes the RLOps stack for the DTT solution presented in section 6.4.

The **Adaptation Layer** is the architectural layer that mediates between the 6G-DALI orchestration layer and the physical testbed infrastructure. This layer is responsible for ensuring the swift execution of solutions in the orchestration layer on the infrastructure including the physical testbeds, DTT, and the data lake. Work related to this layer will be discussed in future deliverables as the relevant task has just started.

Cross-pillar modules, in particular the Experimentation on Demand component and the Adaptation Layer, serve as the integration substrate connecting the WP4 blocks to the broader 6G-DALI system. While this deliverable focuses on the individual maturation of the WP4 functional blocks, the DataOps, MLOps, and 6G Data Space blocks that constitute the complementary WP3 scope are covered in deliverable D3.1. At this current phase of the project, these blocks are primarily treated as stand-alone entities to focus on their individual functional definition, component design, and initial implementation. However, all components have been designed with the necessary interfaces for integration, ensuring they are compatible by design to facilitate the integration that will be carried out in the upcoming project phases.

2.2 Contributions to 6G-DALI innovations, technical objectives and KPIs

This section provides a high-level mapping of the contributions of this deliverable towards the WP4-related Innovation Areas envisioned in 6G-DALI and described in the Description of Action (DoA) (sub-section 2.2.1), and reports on how the work carried out in this initial phase of WP4 (and described in this deliverable) contributes to the achievement of the 6G-DALI Objectives and the related KPIs (sub-section 2.2.2).

2.2.1 Innovation Area KPI Status

This section outlines the progress of the 6G-DALI Innovation Areas relevant to WP4 by evaluating their associated KPIs. Specifically, Table 2, Table 3 , Table 4, Table 5, and Table 6 detail the KPI statuses for Innovation Areas 2, 5, 6, 7, and 8, respectively. For each KPI, the tables provide a brief description, its current status of achievement, and a cross-reference to the specific section demonstrating how recent WP4 advancements are contributing to these targets. In summary, the achievement of these Innovation Area KPIs is progressing as expected at the M12 stage; the related assessment work is ongoing, with relevant technical advancements already carried out as reported in the tables below. Full achievement and assessment of these KPIs is planned to be reported in the next iteration of this deliverable.

Table 2. Innovation Area 2 – Collaborative MLOps and RLOps

Innovation Area 2 – Collaborative MLOps and RLOps			
KPI	Description	Status	Ref. Section and Short Rationale
KPI IA2.1	Generalization Error lower than 5%, relative to the baseline of RL policies trained in a single environment without domain randomization or transfer learning.	Ongoing	Sections 4.3 and 6.4: AutoRL and RLOps-DT integration address generalization for RL in 6G settings

Table 3. Innovation Area 5 – Digital Twin for large-scale 6G experimentation

Innovation Area 5 – Digital Twin for large-scale 6G experimentation			
KPI	Description	Status	Ref. Section and Short Rationale
KPI IA5.1	Enable large-scale and high-fidelity DT experimentation combining OAI and VIAVI tools	Ongoing	Section 6.2: AI-RSG and FlexRIC integration validated with E2 interface, KPM/RC xApps operational
KPI IA5.2	Automate DT experiment lifecycle with minimal human intervention	Ongoing	Section 6.2: ADK script automates scenario execution, E2 configuration, and pacing mode for RL validation

Table 4. Innovation Area 6 – Knowledge Transfer Automation

Innovation Area 6 – Knowledge Transfer Automation			
KPI	Description	Status	Ref. Section and Short Rationale
KPI IA6.1	Cooperative transfer learning with multi-	Ongoing	Section 4.4: Cooperative TL framework with

	model synergistic training		adaptive knowledge distillation
KPI IA6.2	Meta-learning for energy-efficiency trade-off in model training	Ongoing	Section 4.4: Initial approach defined; full evaluation planned for D4.2

Table 5. Innovation Area 7 – Federated Learning with NWDAF operations for 6G

Innovation Area 7 – Federated Learning with NWDAF operations for 6G Privacy and Efficiency:			
KPI	Description	Status	Ref. Section and Short Rationale
KPI IA7.1	NWDAF-enabled FLOps leveraging 3GPP Release 18 capabilities	Ongoing	Section 5.3.1: OAI NWDAF R17 implementation operational; FLOps architecture defined
KPI IA7.2	Privacy-preserving FL with robust aggregation algorithms	Ongoing	Section 5.3.3: Privacy-preserving with robust aggregation algorithm with preliminary results

Table 6. Innovation Area 8 – LLM-enabled AI experimentation and HPO

Innovation Area 8 – LLM-enabled AI Experimentation and Hyperparameter Optimization			
KPI	Description	Status	Ref. Section and Short Rationale
KPI IA8.1	LLM-based translation of user intents into experiments	Ongoing	Section 3.2: LLM-based experiment UI with intent-to-experiment translation pipeline
KPI IA8.2	Computationally bounded HPO for efficient model tuning	Ongoing	Section 3.3, 4.2: Budget-aware dataset generation and ADO for HPO in MLOps

2.2.2 Objectives KPI Status

This section reports the status of the 6G-DALI Objectives relevant to this deliverable in terms of their related KPIs. Table 7, Table 8, Table 9, Table 10, Table 11, report the statuses of the KPIs related to Objectives 1, 2, 3, 4, and 5 respectively. For each KPI, together with its description, the current status and a reference to the relevant section are reported.

Table 7. Objective 1 – Deliver a user-friendly e2e AI framework for DataOps and MLOps in 6G

Objective 1 Deliver a user-friendly e2e AI framework for DataOps and MLOps in 6G			
KPI	Description	Status	Ref. Section and Short Rationale
KPI O1.1	Deliver 1 intuitive API gateway supporting intents for Data & AI experimentation; allow formulation of 100% of data requests via natural language	Ongoing	Section 3.2: Intent-driven data generation translates natural-language user intents into data-request specifications, directly addressing the intent-based API gateway
KPI O1.3	Showcase inference energy savings >90%	Ongoing	Section 4.4 and 5.2: Cooperative Transfer

	with AIaaS-driven quantization, transfer learning and finetuning, with <5% accuracy loss vs., relative to the baseline of full-precision FP32 inference without compression or transfer learning.		Learning and workload placement address energy-efficient inference; full evaluation planned for D4.2
KPI O1.4	Automate ML deployment of quantized models at heterogeneous hardware-agnostic CEC deployments with at least 1 ARM64 device	Ongoing	Section 5.2: AI/ML Workload Placement across the Cloud-Edge continuum targets heterogeneous hardware deployment; ARM64 validation planned for D4.2

Table 8. Objective 2 – Deliver Gaia-X and ELT approaches for DataOps in 6G environments

Objective 2: Deliver Gaia-X and ELT approaches for DataOps in 6G environments			
KPI	Description	Status	Ref. Section and Short Rationale
KPI O2.2	Implement at least 4 ELT pipelines for three testbeds and 1 DT	Ongoing (DT); Not started (testbed adapters)	Section 6: for the DTT under development; testbed-adapter ELT pipelines will be delivered under T4.4 (starts M16)
KPI O2.3	Satisfy at least 70% of data provisioning requests via the natural language intent-based APIs	Ongoing	Section 3.2: LLM-based experiment user interface translates natural-language user intents into experiment specifications
KPI O2.5	Deliver at least 6 open and reusable datasets from 6G testbed experimentation covering at least 4 domains	Ongoing	Section 3.3 and 6.3: Budget-aware dataset generation reduces testbed runs required; AI-RSG enables large-scale DT data generation

Table 9. Objective 3 – 6G testbed's trustworthy AI/ML operations via MLOps and AutoML

Objective 3: Objective 3 –6G testbed's trustworthy AI/ML operations via MLOps and AutoML			
KPI	Description	Status	Ref. Section and Short Rationale
KPI O3.1	Reduce by half the time to validate ML/RL models in 6G testbeds via MLOps & AutoML , relative to the baseline of manual validation workflows without automation.	Ongoing	Section 4 HPO and AutoRL pipelines defined; benchmarking planned for D4.2
KPI O3.2	Detect 80% of non-efficient ML models , relative to the baseline	Ongoing	Primarily addressed in D3.1 (Trustworthy AI); WP4 contributes via RLOps validation

	of state-of-the-art drift and efficiency detection without 6G-DALI RLOps monitoring.		
KPI O3.3	Implement RLOps for at least 2 testbeds, integrating 1 DT framework	Ongoing	Sections 4.3 and 6.4 ZenML-based RLOps pipeline integrated with DT
KPI O3.4	Support at least 2 heterogeneous MLOps stacks with meta-orchestration	Ongoing	Sections 4 and 5 custom stacks in use; meta-orchestration integration ongoing
KPI O3.5	Double ML workloads via computationally bounded HPO, relative to the baseline of unconstrained HPO without budget-aware surrogate models.	Ongoing	Sections 3 and 4 surrogate models and ADO approach
KPI O3.6	Improve reliability via >20% reduction in vertical SLA violations due to model drift, relative to the baseline of SLA violation rates without drift detection or FLOps mechanisms.	Ongoing	Primarily addressed in D3.1; WP4 contributes via FLOps and TL mechanisms

Table 10. Objective 4 – Deliver pluggable adapters to easily integrate 6G testbeds from future calls

Objective 4: Deliver pluggable adapters to easily integrate 6G testbeds from future calls			
KPI	Description	Status	Ref. Section and Short Rationale
KPI O4.1	Deliver at least 4 open-source adapters connected to 6G Dataspace (1 per testbed + 1 DTT)	Not started	T4.4 formally starts M16; adapter specifications will be covered in D4.2
KPI O4.2	Deliver 1 aligned Northbound API for experimentation as a service, abstracting 3 Testbed APIs	Not started	Adaptation layer design is planned under T4.4; early design considerations noted
KPI O4.3	Showcase deployment of e2e AI framework at new testbeds in <1 day via hackathons	Not started	Planned for later project phases; hackathons scheduled post-integration

Table 11. Objective 5 – Build and integrate a Digital Twin Testbed to generate representative datasets for 6G

Objective 5: Build and integrate a Digital Twin Testbed to generate representative datasets for 6G			
KPI	Description	Status	Ref. Section and Short Rationale

KPI O5.1	Enable two types of experiments: large-scale low-fidelity and medium-scale high-fidelity	Ongoing	Section 6 supports both scales; simulation with configurable fidelity
KPI O5.2	Deliver a DTT for 6G supporting O-RAN architecture	Ongoing	Section 6 with FlexRIC via E2 interface; KPM and RC xApps validated
KPI O5.3	Deliver at least 2 open datasets via the DTT	Ongoing	Section 6 workflow operational; dataset export to 6G-DALI Data Space defined

3 On-demand Dataset Generation

3.1 Overview

This section covers initial progress in the area of on-demand dataset generation. The core challenge addressed here is how experimenters obtain datasets for AI/ML model development: either by expressing what they need in natural language, or by having the system to identify which testbed experiments are most worth running. One of the project goals calls for 6G testbeds that non-expert users can interact with directly, issuing requests in plain language while the underlying system translates those requests into concrete experiment configurations and decides how to allocate a bounded experimentation budget.

Two complementary approaches to populate the 6G Dataspace are described. Intent Driven Data Generation (Section 3.2) takes natural-language requests from users and turns them into concrete data-generation experiments through a pipeline of LLM-based agents. Surrogate Based Data Generation (Section 3.3) takes a different angle: instead of generating a dataset fully via a testbed experiments, users provide the characteristics of the dataset in the form of a configuration space (e.g., number of replicas of a workload, frequencies to be used, etc.). The system will then train a surrogate model that can generate the requested data with minimal number of testbed-based experiments, with the goal of generating a dataset at low experimental cost (i.e., time and testbed resources). The model tuning work — HPO, AutoML, AutoRL, and transfer learning that builds on the datasets produced here is covered in Section 4. The distributed training infrastructure and workload execution layer that actually runs the experiments is covered in Section 5.

3.2 Intent-based data generation

This section presents the work on translating user intents into experiments using LLM-based agents. This workflow should be initiated only when no dataset replying the user request is found in the dataspace[2] [1] . Though, the proposed agents are expected to detect when it is necessary to launch an experiment on a suitable testbed to generate the requested data. In total, we can say the architecture is well designed if the user gets his request from the right place; the datasets dataspace if it is already available or from a new experiment if the dataset is not produced yet.

Referring to Figure 1, the intent-based data generation is a component within the DataOps Architectural block

3.2.1 Problem statement and approach

Generating datasets from 6G experimental testbeds is a technically demanding task that currently requires familiarity with testbed-specific APIs, experiment configuration schemas and parameter spaces, and measurement collection pipelines. For example, generating datasets with the VIAVI AI-RSG digital twin requires specialized expertise due to its platform-specific APIs, experiment configuration schemas, and measurement collection workflows[3] . Users must interact with the AI-RSG Application Development Kit (ADK) to retrieve network measurements, apply control actions, and manage communication between reinforcement learning agents and the simulated environment. Furthermore, realistic scenario creation involves configuring numerous network and radio parameters, while data generation requires orchestrating KPI collection pipelines and synchronizing them with the learning process. These technical requirements create a significant entry barrier for non-expert users and motivate the development of higher-level, intent-driven abstractions that automate experiment specification, execution, and data collection.

As 6G research increasingly relies on heterogeneous experimental environments, including simulation-based platforms, and different protocol testbeds, researchers must interact with diverse APIs, configuration schemas, and data models that are often specific to each platform[4] . As a result, experiments developed for one testbed typically require manual adaptation before they can be reproduced or executed on another. This lack of interoperability increases the technical expertise required to conduct experiments, hinders experiment portability and reproducibility, and creates a significant barrier for researchers who are not familiar with the implementation details of each testbed.

To address this challenge, an agentic architecture (Figure 2) is proposed. In this framework, a cooperative ensemble of specialized LLM-based agents progressively translates a high-level user intent expressed in natural language into a fully automated experiment execution and data collection workflow to generate and collect the requested dataset.

The system is grounded in a set of design principles that collectively aim to simplify and automate the experimental workflow while ensuring reliability and adaptability across heterogeneous testbeds. First, it adopts an intent-first interaction paradigm, in which users specify their experimental objectives in natural language rather than configuring low-level parameters. For example, a user may express a goal such as requesting traffic steering KPM data in a dense urban scenario with a given number of user equipments and energy-saving features enabled. The responsibility of translating this high-level intent into a concrete and executable experiment specification lies entirely with the system, thereby reducing user burden and minimizing configuration errors.

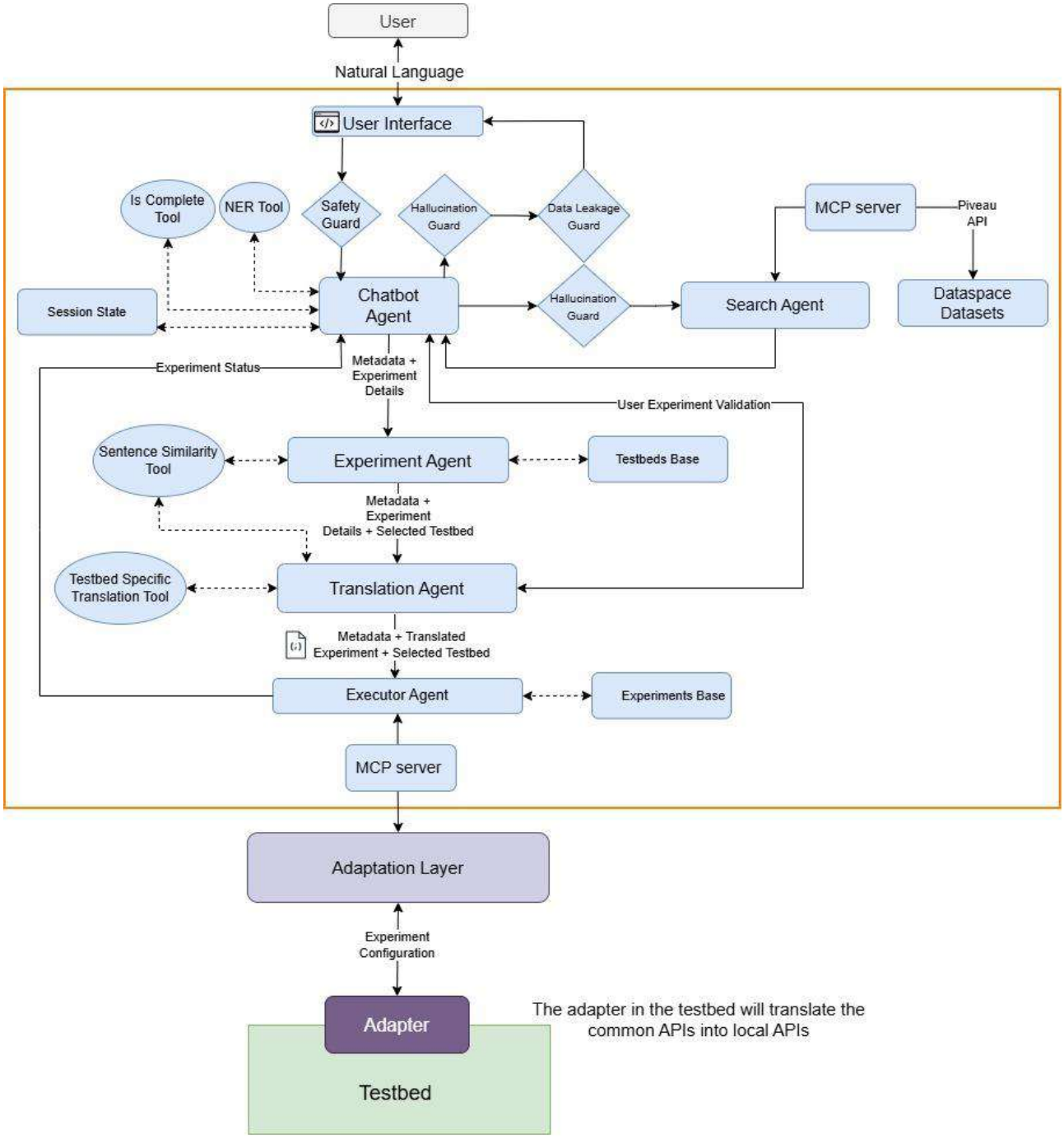


Figure 2: Intent-based agentic data generation framework architecture

The Chatbot Agent interacts with the user to clarify the experiment intent whenever additional information is required. A safety guardrail validates the relevance and security of user requests, protecting the system against malicious inputs such as prompt injection attacks. To support intent understanding and data generation, the agent leverages several tools, including a Named Entity Recognition (NER) module to extract relevant metadata (e.g., index keywords and experiment parameters), an *is_complete* tool to verify that all required experiment information has been provided, and a session state mechanism to maintain the conversation history throughout the interaction. This enables the Chatbot Agent to progressively collect the necessary information and forward it to downstream agents.

Guardrails play an important role in ensuring the reliability and security of the system. Input guardrails, such as the safety guardrail, monitor and validate user requests to detect harmful or irrelevant prompts. Output guardrails, including hallucination and data-leakage guardrails, verify the generated responses to prevent the disclosure of confidential information and reduce the risk of inaccurate, misleading, or meaningless outputs.

Once the user intent has been sufficiently clarified, the Chatbot Agent forwards the request to the Experiment Agent. The Experiment Agent is responsible for identifying the most appropriate testbed or the digital twin capable of executing the requested experiment and generating the desired data. To support this decision, the agent has access to a testbed knowledge base that describes the capabilities, resources, and supported experiment types of each available testbed/the digital twin. It also leverages a similarity-matching tool to compare the interpreted user intent with the available testbed/the digital twin profiles and select the testbed that best satisfies the experiment requirements.

As the framework is limited with the testbeds capabilities and what they can run as experiments, the system relies on semantic scenario matching rather than generating configurations from scratch. Given the complexity and specificity of testbed parameter spaces, direct generation approaches can be unreliable and prone to invalid configurations. Instead, the Translation Agent maps user intents to the most relevant predefined scenarios available in the testbed's scenario catalog. This mapping is performed using Sentence Transformer embeddings and dense semantic similarity techniques, which significantly reduce the search space while guaranteeing that the selected configurations are both syntactically correct and semantically consistent with the target testbed/digital twin.

Another key aspect of the architecture (Figure 2) is its testbed-adaptive translation capability. The system decouples the interpretation of user intent from the mechanisms used to translate that intent into testbed-specific configurations. For instance, in the case of the VIAVI ADK platform, the Translation Agent leverages sentence-embedding similarity across the entire scenario catalog to identify the best match. In contrast, for other platforms such as KUL, ISI, and EUR, the agent may invoke dedicated testbed-specific translators or query a Testbed Knowledge Base containing structured descriptions of capabilities. This design enables context-aware scenario selection and ensures interoperability across diverse experimental infrastructures.

Finally, the proposed architecture supports the full automation of experiment execution. Following the selection of the most appropriate testbed and experiment scenario, the Executor Agent manages the end-to-end experimental lifecycle. It prepares the experiment configuration and associated metadata (e.g., title, description, keywords, and execution parameters) and submits them to the adaptation layer. Communication between the AI-driven orchestration framework and the underlying testbeds is achieved through an MCP server, which provides a standardized interface for experiment deployment, execution management, status reporting, and result retrieval. This abstraction layer enables interoperability across heterogeneous testbeds while shielding higher-level agents from testbed-specific implementation details. By automating the execution process, the system improves operational efficiency, enhances experiment reproducibility, and facilitates large-scale experimental campaigns across diverse research infrastructures.

3.2.2 Semantic scenario matching algorithm

The Translation Agent employs a retrieval-based approach to identify the most relevant predefined experiments available within the target testbed or digital twin. Each experiment description is encoded into a fixed-length dense vector using the all-MiniLM-L6-v2 sentence transformer model, a lightweight

22-million-parameter bi-encoder trained on large-scale natural language inference and semantic similarity datasets. The model generates 384-dimensional embeddings that are optimized for semantic similarity search using cosine similarity.

At runtime, the user's intent, represented by a suitable representation collected from all user messages throughout the interaction with the Chatbot Agent, is encoded using the same embedding model. Cosine similarity scores are then computed between the intent embedding and the embeddings of all predefined experiment scenarios. Candidate scenarios whose similarity score exceeds a predefined threshold τ=0.1. The top-k candidates (currently k=3) ranked by descending cosine similarity are returned to the frontend for user selection.

Formally, let $q \in \mathbb{R}^d$ be the intent embedding and $\{s_i\}_{i=1}^N$ be the scenario embeddings (d=384). The similarity score for scenario i is:

$$\sigma_i = \frac{q.s_i}{\|q\|.\|s_i\|}$$

The matched set is $\mu = \{i: i \wedge \delta_i \geq \tau\}$, and the top-3 results are $\{\sigma_i : i \in \mu\}$.

3.2.3 Experiment execution pipeline

Once the user confirms a scenario selection following the semantic matching step, the Executor Agent orchestrates the complete experiment lifecycle across the selected testbed. Figure 3 presents the end-to-end interaction sequence of the proposed intent-based data generation framework, illustrating how the cooperative ensemble of agents translates a validated user intent into a fully automated experiment execution workflow.

The pipeline begins with a conversational intent refinement loop between the user and the Chatbot Agent. The Chatbot Agent invokes an *extract_keywords()* tool to parse, structure and extract keywords from the user's natural-language input, and calls *is_complete_intent()* to assess whether the expressed intent carries sufficient specificity. If the intent is deemed incomplete, the agent prompts the user for clarification; otherwise, the extracted keywords are forwarded downstream. This iterative refinement ensures that downstream agents receive a well-specified experimental objective before any testbed interaction occurs.

Once a complete intent is established, the Chatbot Agent forwards the request, comprising the refined intent and extracted keywords and other metadata (like the title and the description), to the Experiment Agent. The Experiment Agent queries the Testbed Knowledge Base via a *testbed_select(intent, keywords)* tool to identify and rank available testbeds according to their capabilities and suitability. As illustrated in the sequence diagram (Figure 3), the Testbeds DB maintains profiles for multiple experimental platforms, including VIAVI DT, EURECOM TB, ISI TB, and KUL TB. The selected testbed's endpoint information and capability metadata are then forwarded to the Translation Agent.

The Translation Agent is responsible for translating the high-level user intent into a standardized, testbed-specific experiment configuration artifact. Using a retrieval-based semantic matching approach - as detailed in the previous subsection - the Translation Agent maps the user intent to the most relevant predefined experiment scenario available within the selected testbed's catalog. It then produces a structured experiment file in JSON, YAML or other testbed specific format, which encapsulates all necessary parameters, KPIs to be collected, and execution constraints.

The Executor Agent subsequently takes ownership of the experiment lifecycle. Communication between the LLM-based agentic framework and the underlying testbed is achieved through an *experiment_mcp_server* (Model Context Protocol server), which provides a standardized interface for experiment deployment and execution management. The Executor Agent calls *launch_experiment*() via the Adaptation Layer, which abstracts away testbed-specific implementation details. Upon receiving a launch confirmation, the Executor Agent notifies the Chatbot Agent, which in turn informs the user that the experiment has been successfully initiated.

Following launch, the system operates on an event-based notification mechanism. The Adaptation Layer monitors the experiment's execution state and asynchronously pushes status updates to the Executor Agent through the *experiment_mcp_server*, whenever a significant state change occurs like experiment execution status or errors. The Executor Agent forwards these progress notifications back to the user

via the Chatbot Agent. This event-driven approach eliminates the need for periodic polling, reduces network overhead, and provides the user with real-time visibility into the data generation process as soon as state changes are detected.

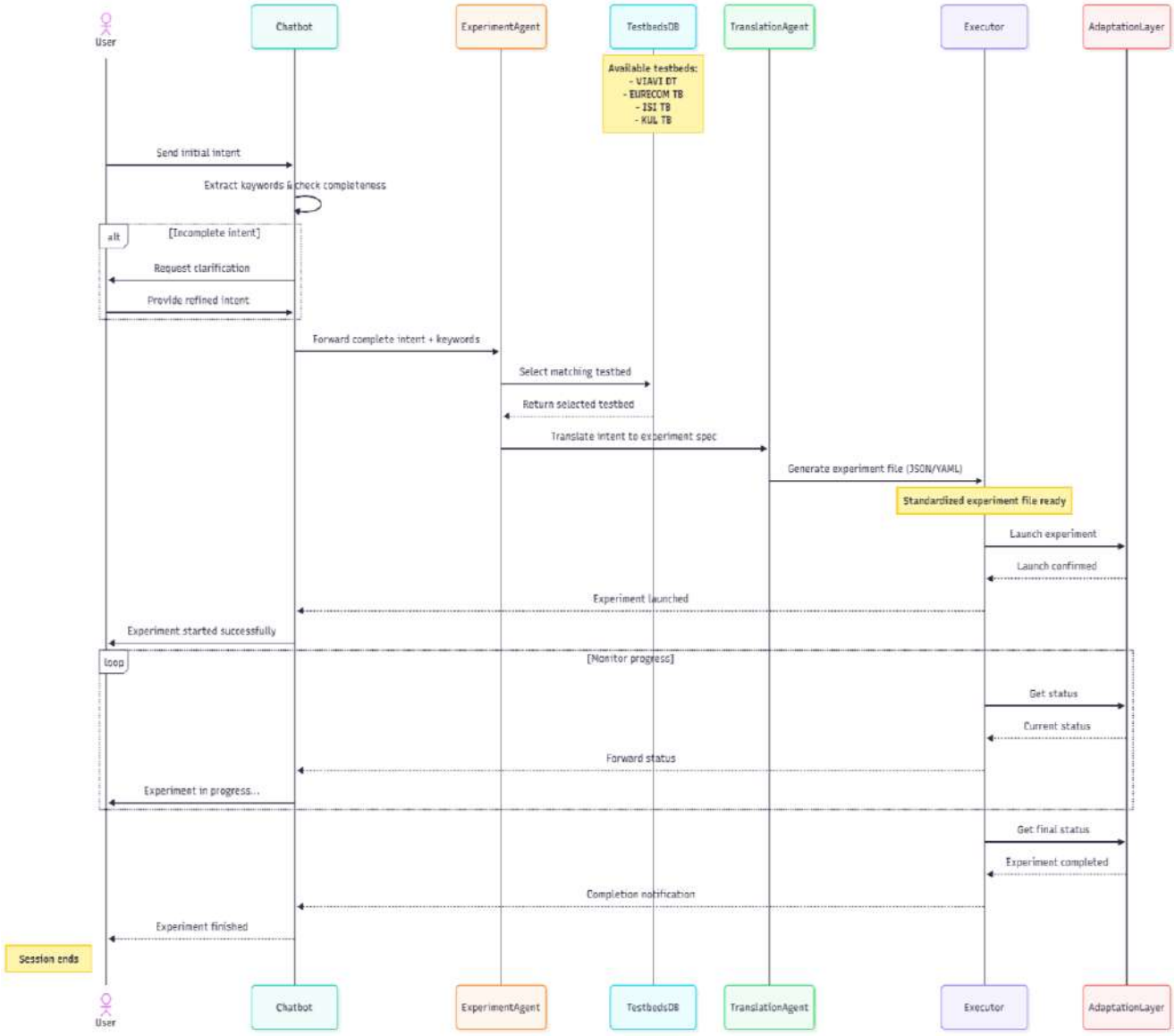


Figure 3: Sequence diagram of data generation

Finally, upon experiment completion, the Adaptation Layer notifies the Executor Agent, which propagates a completion notification to the Chatbot Agent. The Chatbot Agent then informs the user that the experiment has finished and that the resulting dataset is available for retrieval. This closes the end-to-end intent-to-data lifecycle: the system has autonomously translated a high-level user request into a concrete experiment, executed it on a suitable testbed, and delivered the generated dataset without requiring the user to interact with low-level testbed APIs or configuration schemas.

The automation of this pipeline addresses the interoperability challenges identified earlier in the problem statement. By decoupling intent interpretation from testbed-specific execution mechanisms, the architecture enables experiment portability across heterogeneous platforms, including simulation-based digital twins like VIAVI RSG, and physical testbeds while shielding researchers from platform-specific implementation details.

3.2.4 Initial results

Figure 4 showcases the user interaction with the Chatbot Agent in natural language, where the user expresses an experimental intent, such as running massive MIMO enabled to evaluate energy efficiency. Following intent clarification, the Translation Agent queries the scenario catalog and proposes the most relevant predefined experiment configurations ranked by their semantic similarity score, as illustrated in the screenshot. The user can then select the configuration that best satisfies their experimental objectives from the ranked ordered list. Upon user confirmation, the Execution Agent initiates the experiment on the selected testbed to generate the requested dataset. Once execution completes, the generated dataset is made available for download, as shown in Figure 5 and Figure 6 respectively.

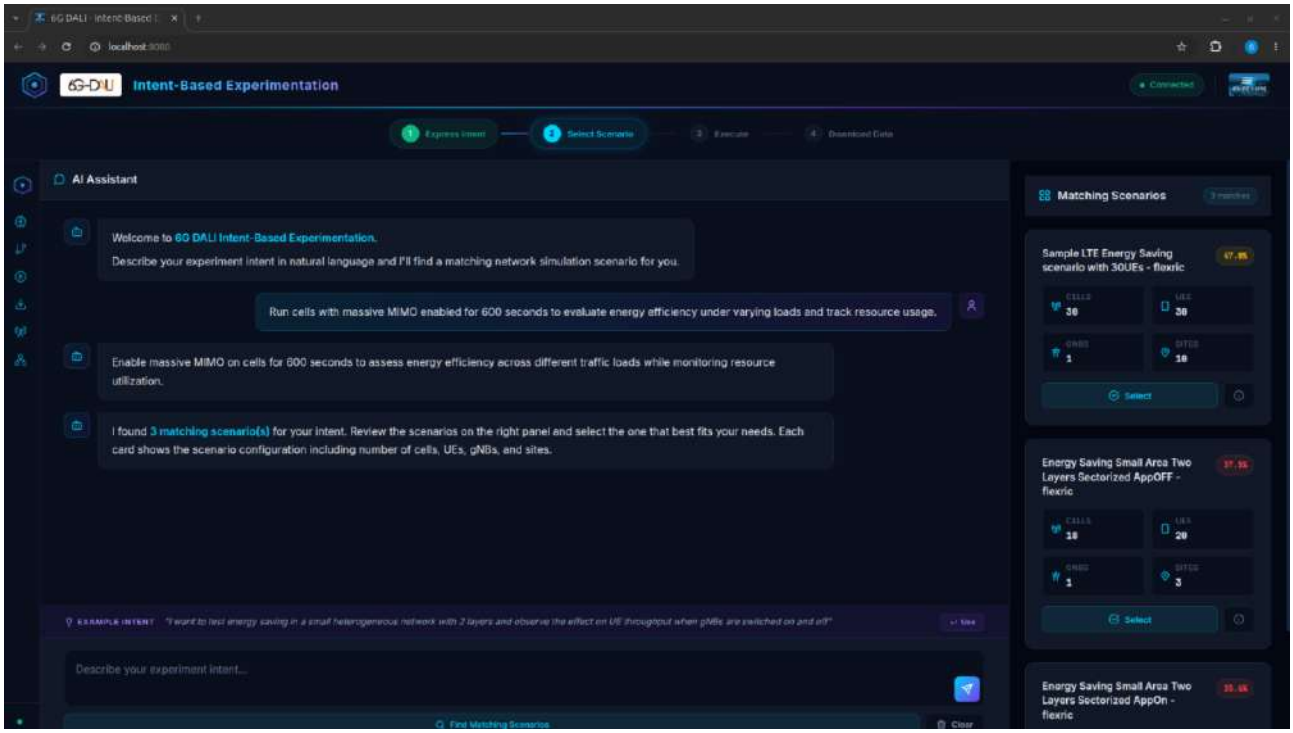


Figure 4: User intent to experiment configuration translation

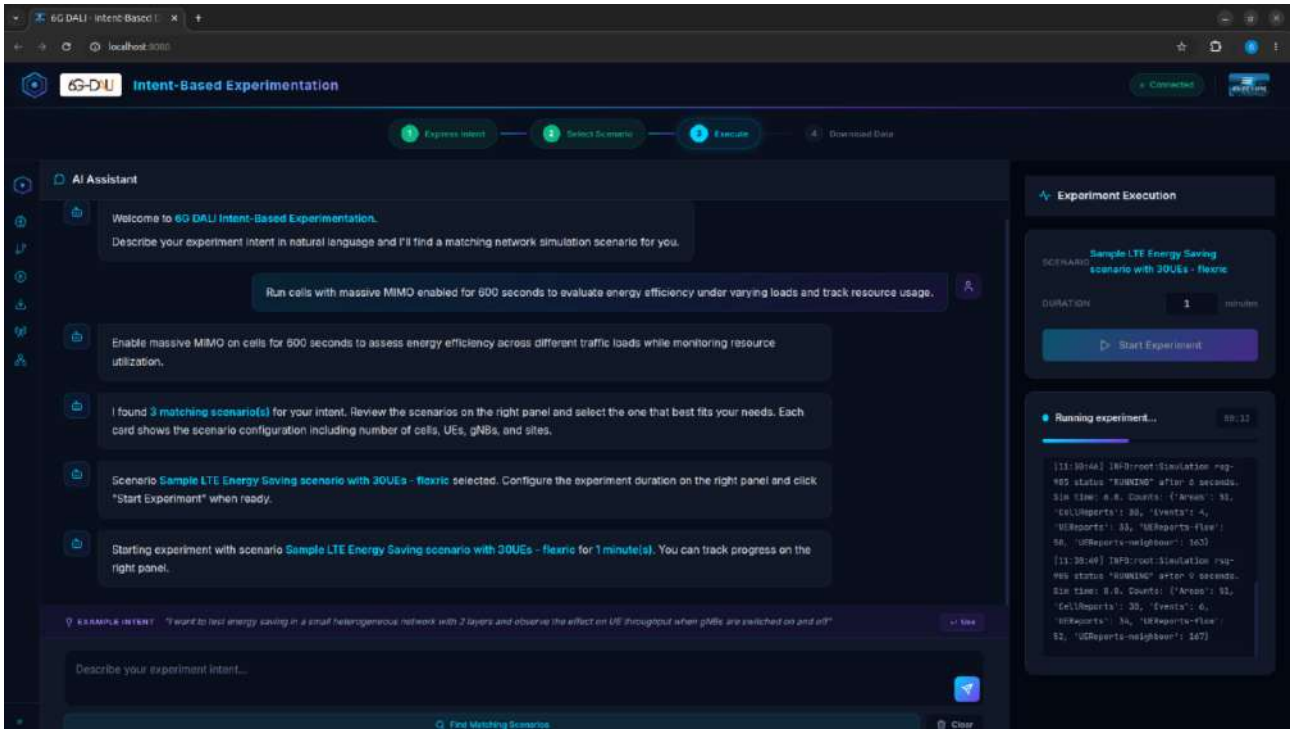


Figure 5: Users' selected experiment execution

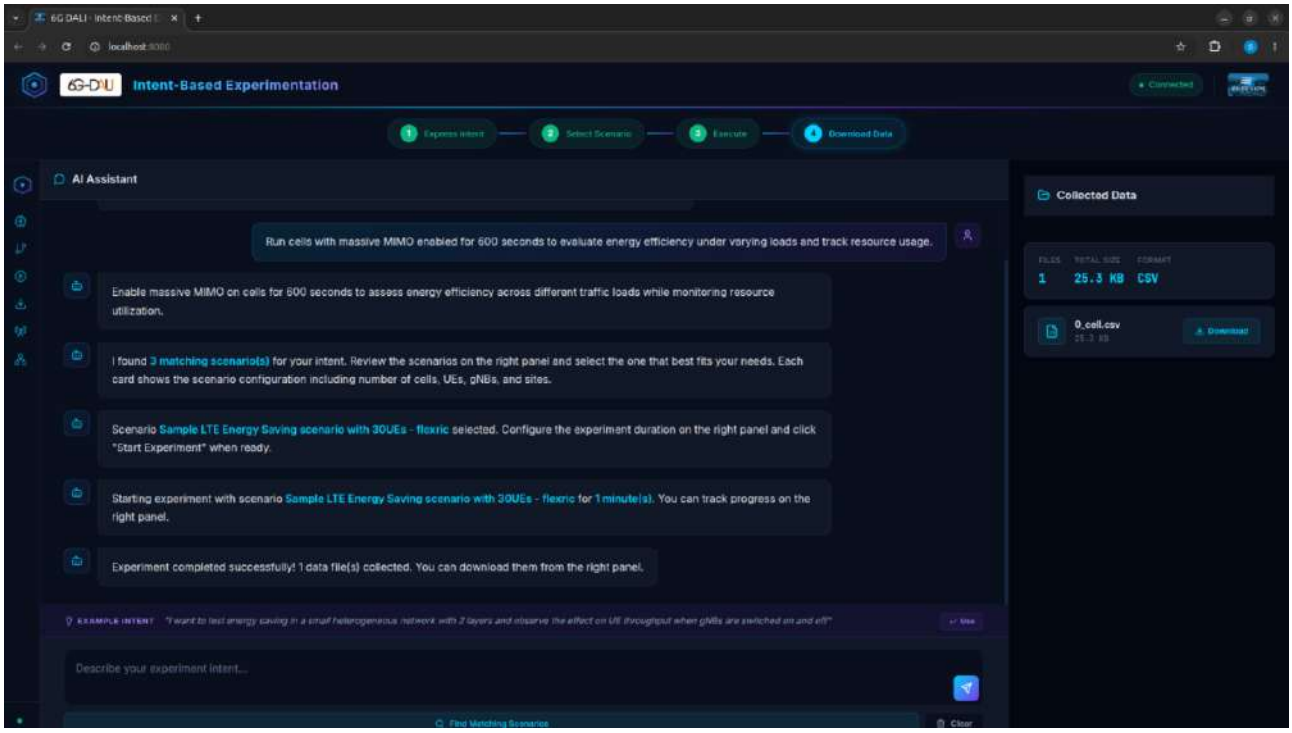


Figure 6: Generated dataset presentation after the end of experiment execution

3.2.5 Next steps

The current implementation targets the VIAVI RSG testbed and demonstrates the full intent-to-data pipeline in that environment. Several extensions are planned to broaden the scope and maturity of the framework.

First, the direct executor–testbed communication will be refactored through a dedicated Adaptation Layer API. Rather than the Executor Agent issuing SSH and Docker commands directly, all experiment lifecycle operations from scenario creation, launch, status notification to data retrieval will be mediated through a uniform MCP-based adaptation interface that abstracts testbed-specific control protocols. This decoupling will make it straightforward to add new testbeds without modifying agent logic, and will enable the system to manage experiments across multiple testbeds concurrently within a single session.

Second, the Translation Agent will be extended to cover the full set of 6G-DALI testbeds, including the EURE, KUL, and ISI/ATH testbeds. For each testbed, a dedicated translator module will be developed, responsible for mapping the intent-derived constraints and extracted keywords to the testbed's native experiment specification format. Where a predefined scenario catalog exists, the existing sentence-embedding similarity search can be reused with a testbed-specific index.

3.3 Surrogate Based Data Generation for Budget-Aware Testbed Experimentation

This section describes the approach to reducing the number of testbed runs required to generate performance datasets for surrogate modelling and HPO experimentation. In the project architecture, exhaustive execution of all possible workload and infrastructure configurations is not realistic: the testbed feature space grows combinatorially with the number of workload parameters, resource limits, deployment options and orchestration choices, while each testbed run consumes time, compute resources and testbed availability. This creates a practical bottleneck for Experimentation on Demand, where the system must produce representative performance data without executing every possible configuration.

The problem addressed in this section is therefore the following: given a candidate parameter space and a limited measurement budget, the system must decide which configurations should be executed as real

testbed runs so that the resulting dataset is sufficiently representative to train an accurate surrogate model. The surrogate can then be used to estimate the performance of untested configurations and to support downstream tasks such as HPO, model validation, workload placement, capacity planning and further experimentation campaigns.

This contribution maps to the project's Objective 4 activities related to Experiment on Demand, automated AI/ML experimentation, and efficient dataset generation for model training and validation. It also supports the D4.1 expected output associated with Task 4.1: the development of an experimentation framework able to support experiment design, dataset generation, model validation and hyperparameter optimization of AI/ML models on demand. In relation to the project architecture defined in D2.3, the proposed method provided the budget-aware acquisition logic that determines which testbed configurations should be executed when the DataOps/MLOps pipeline requires new representative measurements.

The term surrogate-based data generation should be understood carefully. The objective is not to replace measured testbed data with arbitrary synthetic data. Instead, the system first selects and executes a limited number of real testbed runs. These measurements are then used to train a surrogate model, which can generate predicted performance values for the remaining untested configurations in the parameter space. The quality of the data generated therefore depends directly on the representativeness of the measured subset. The central objective is to select this subset carefully so that the surrogate remains accurate across the full operating space while using only a small fraction of the possible testbed runs.

The proposed method, Predictive Kernel Herding (PKH), implements a budget-aware acquisition logic. Starting from an experimental intent, the system defines the candidate parameter space, checks whether existing measurements are available, and invokes PKH when additional measurements are required. PKH selects the next configurations to execute by comparing the distribution of the already measured subset with the distribution of the full candidate testbed feature space in a representation learned by a Random Forest surrogate. The selected configurations are then executed as testbed runs, and the resulting measurements are returned to the DataOps/MLOps pipeline for surrogate training, validation and HPO experimentation.

The main objectives of this section are: to define the budgeted testbed sampling problem, explain why simulation alone cannot provide data with the required fidelity or coverage, introduce PKH as the acquisition method used to select informative testbed runs, and evaluate whether PKH can reduce the number of required measurements while preserving surrogate accuracy.

3.3.1 Project Context and Architectural Positioning

This subsection explains how the proposed method maps to the project architecture and how it supports the Experimentation on Demand workflow. In the architecture, a user or an internal MLOps component may request an experimentation campaign over a given testbed feature space. This request may correspond, for example, to characterizing the performance of a workload, validating a model, launching an HPO campaign or generating a dataset for surrogate-based performance prediction.

As outlined by Figure 7, the workflow starts with an experimental intent. This intent is translated into a structured experiment specification containing the target workload, the KPIs of interest, the constraints, the available measurement budget and the candidate parameter space. The relevant parameters may include workload-level parameters, such as request rate, input size, number of users or concurrency level, as well as infrastructure-level parameters, such as CPU limit, RAM limit, accelerator type, deployment topology or orchestration policy.

Once the experiment specification is available, the DataOps and MLOps components query the 6G Data Space to identify existing measurements, metadata, and previously trained surrogates related to the requested testbed feature space. If the existing data already provided sufficiently representative coverage of the relevant variables, operating conditions and parameter-space regions, the system can train or reuse a surrogate model directly. If coverage is insufficient, or if the testbed feature space has changed in a way that makes previous measurements unreliable, the Experimentation on Demand component triggers a new measurement campaign.

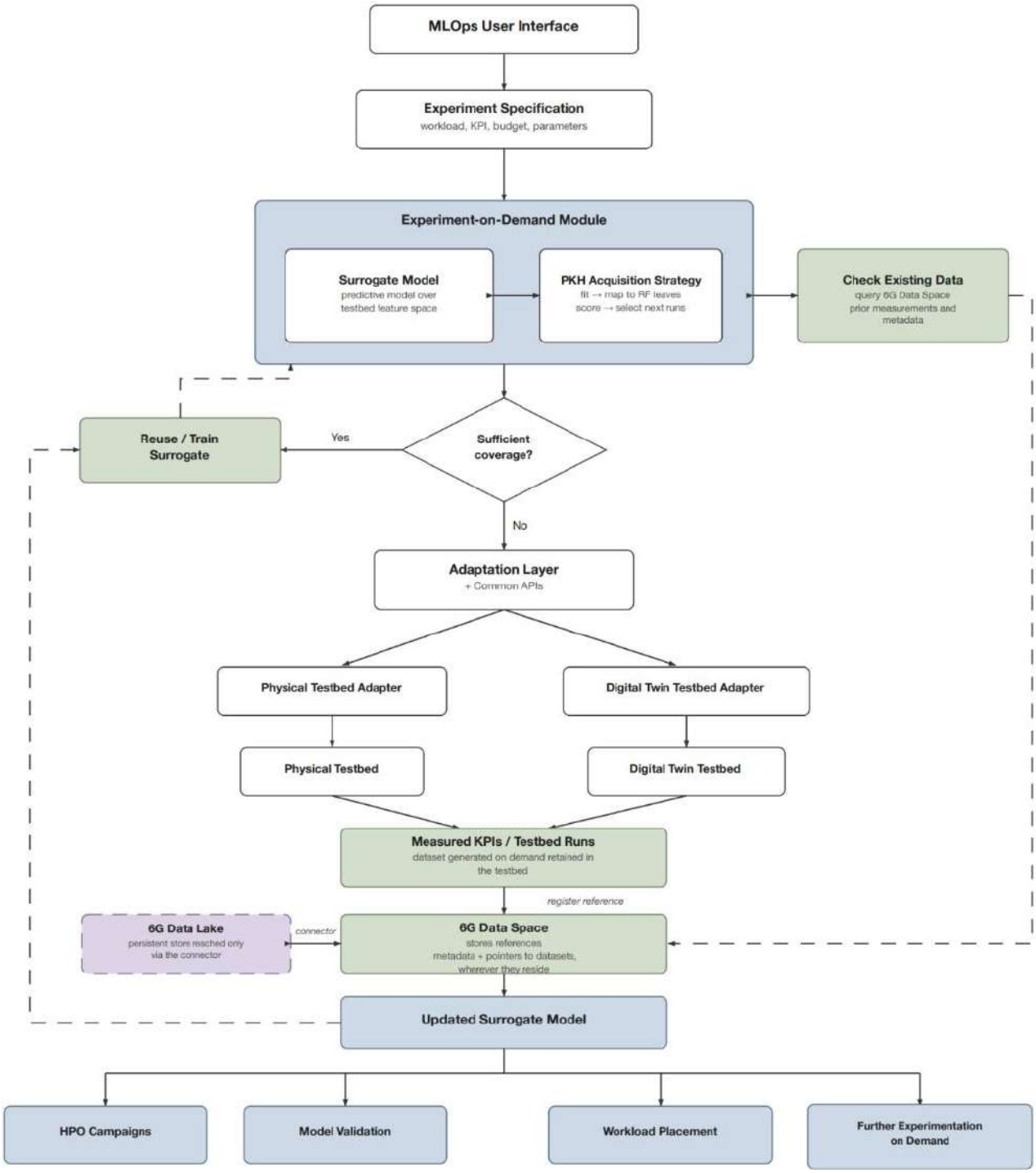


Figure 7: Surrogate Construction Workflow from Testbed Runs during an Experiment Campaign

At this point, PKH is invoked as the acquisition policy. Its role is to decide which subset configurations should actually be executed as testbed runs under the available budget. The selected configurations are passed through the Adaptation Layer and the relevant Testbed Adapter (or Digital Twin Testbed Adapter). These adapters translate the abstract experiment configuration into executable commands for the physical testbed, the digital twin testbed or both.

After execution, the generated datasets are stored in the Data Lake and indexed through the 6G Data Space services. The MLOps Orchestrator can train or update the surrogate model and, once its predictive quality is sufficient, use it to approximate the remaining unmeasured of the experiment space as well as support HPO, model validation, workload placement or additional Experimentation on Demand campaigns. If the surrogate validation error remains too high in specific regions of the parameter space, the MLOps Orchestrator can trigger another round of acquisition, hence closing the loop between surrogate training and testbed experimentation.

In this workflow, PKH is not a simulator and it is not an HPO controller either by itself, but rather the budget-aware sampling mechanism that decides which testbed runs are most informative for building a surrogate over the current parameter space. The resulting surrogate can then be used by the MLOps pipeline to generate or approximate the missing part of the target dataset and to accelerate HPO and other downstream optimization tasks.

3.3.2 State-of-the-Art Analysis

Building accurate performance surrogates for complex workloads on 6G infrastructure requires combining two complementary bodies of work. The first concerns workload modeling and simulation which can provide fast approximations of workload behaviour. The second concerns experimental design and active learning, which defines how a limited number of testbed runs should be selected when exhaustive execution is infeasible. This section reviews both families of approaches, identifies their limitaions in the context of the project architecture, and motivates the budget-aware acquisition method introduced in the following subsections.

3.3.2.1 Simulation-Based Workload Characterisation

Simulation-based workload characterization is a common way to reduce the cost of exploring large testbed feature spaces. Recent simulation frameworks can estimate workload behavior by modelling execution at different levels of abstraction, from operator-level execution to larger-scale system behavior. These tools are useful because they can produce fast approximate estimates without requiring every configuration to be executed on the physical testbed.

In the project architecture, simulation-based tools may support preliminary exploration, digital twin bootstrapping, or initial estimation before a real experimentation campaign is launched. They may also help identify broad regions of interest in the parameter space before more expensive measured testbed runs are scheduled.

However, simulator-generated data does not have the same fidelity as measured testbed data. A simulator is only as accurate as the assumptions and profiles on which it was calibrated. Its predictions may degrade when the operating conditions move away from the profiling regime used to construct or calibrate it, especially when workload behavior, deployment structure, hardware resources, scheduling policy or traffic conditions change [5] [6] .

This limitation is important for Experimentation on Demand because the project must support changing testbed conditions and evolving AI/ML workloads. When the testbed feature space changes, for example because a new workload, model, accelerator, resource limit, deployment topology, scheduler, traffic profile or orchestration policy is introduced, the simulator must be reprofiled or recalibrated using new testbed runs before its outputs can be trusted.

Recent systems that perform deployment search, dynamic reconfiguration, pre-emptible resource management or target-driven performance characterisation also illustrate why downstream components require predictions for arbitrary feasible configurations, not only for a narrow region of the space[7] [8] [9] [10] . Related configuration-optimization systems similarly focus on identifying good operating points rather than producing representative measurements of the full parameter space[11] [12] . This reinforces the need for whole-space surrogates that remain reliable across the operating space.

Simulation can therefore reduce exploration cost but it cannot remove the need for a budget-aware mechanism that selects which real testbed runs should be executed. The role of the proposed method is precisely to decide which measurements are most informative when the available budget is too small to cover the full parameter space.

3.3.2.2 Model-Free Sampling Designs

The standard alternative to exhaustive profiling is to sample a subset of the configuration space and fit a surrogate model on the resulting measurements[13] [14] . The simplest such strategy, random sampling, is widely used in prior performance-prediction work[13] , but tends to under-represent specific configuration dimensions in expectation, reducing the reliability of the surrogate in lightly sampled regions of the testbed.

More principled alternatives such as Latin Hypercube Sampling[15] [16] , and its many variants[17] [18] [19] , which provide stronger space-filling guarantees and have become the default in surrogate-modeling pipelines[20] [21] . However, these designs were developed for continuous domains and require ad-hoc adaptations to operate over the mixed ordinal-categorical parameter spaces typical of modern compute testbeds[22] [23] [24] [25] .

More fundamentally, space-filling criteria are provably optimal only when the response-surface complexity is uniform across the space[26] , an assumption that rarely holds for realistic workloads, where narrow regions of the parameter space can display highly non-linear behavior driven by batching, memory pressure or resource-contention effects.

3.3.2.3 Active Learning For Surrogate Construction

Active Learning addresses the limitations of model-free designs by adapting sample selection to the testbed benchmarks already acquired.

Two broad sub-families are relevant in this context. Uncertainty-driven active methods[27] [28] [29] [30] prioritise configurations on which the surrogate is least confident. While intuitive, this strategy concentrates benchmarks in high-variance pockets around decision boundaries and is known to conflate reducible and irreducible uncertainty[31] which can degrade average prediction accuracy across the configuration pool. Coverage-driven methods take the opposite stance and aim to spread the budget so as to represent the pool well. GSx[32] greedily selects the farthest point from the current set in raw feature space, regression-tree based active learning (RT-AL)[33] partitions the space using a regression tree and balances allocation across the resulting leaves, and MaxHerding[34] minimises a set-coverage objective formally connected to kernel herding[35] and to Frank-Wolfe moment matching[36] .

These methods improve on random sampling in specific regimes, but all three operate in raw parameter space and therefore allocate budget without regard to which configurations the surrogate model actually finds difficult to predict.

3.3.2.4 Identified Gap

Taken together, these three families leave a clear gap: indeed, simulations provide fast but lower-fidelity data and require reprofiling when the testbed feature space changes. Model-free sampling designs provide generic coverage but do not adapt to the response-surface complexity observed in measured testbed data. Uncertainty-driven active learning adapts to the surrogate model but may focus too narrowly on local uncertainty. Coverage-driven active learning improves representativeness but usually measures coverage in raw parameter space rather than in the surrogate's learned representation.

The project architecture requires a method that can decide, under a limited budget, which testbed runs should be executed to build a surrogate that remains accurate across the full operating space. The missing capability is therefore a budget-aware acquisition policy that is both representative of the full candidate parameter space and aware of the surrogate model's view of that space.

This is the role of **Predictive Kernel Herding** (**PKH**)[37] . Instead of selecting configurations only because they are far apart in raw parameter space, PKH uses the structure learned by a Random Forest surrogate to identify which regions of the testbed feature space are under-represented from the model's perspective. This makes the sampling policy better aligned with the final objective of the experimentation campaign: training a surrogate that provides reliable predictions across the full testbed operating space.

3.3.3 Preliminary Pipeline and Evaluation

3.3.3.1 Algorithm Formulation

Figure 8 summarizes the PKH loop end-to-end. The sampling procedure begins with a warm-start phase (step 0), in which the labeled set is seeded with a small number of randomly chosen testbed

configurations – five in our implementation – to provide enough initial coverage for the surrogate to be fit. Each subsequent iteration then executes four stages:

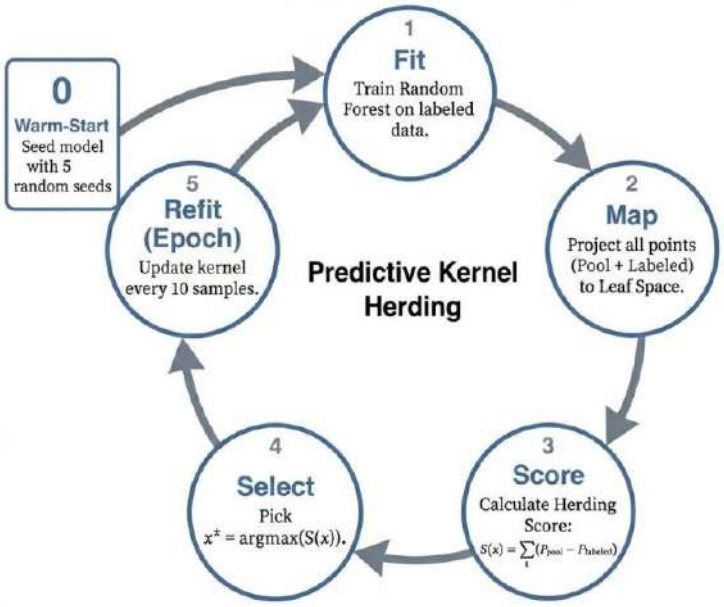


Figure 8: Predictive Kernel Herding (PKH) sampling strategy – Selecting the most under-covered testbed runs in a feature space learned by the surrogate model instead of the raw testbed parameter space.

- **Step 1 - Fit Stage**: in the fit stage, a Random Forest is trained on the current labeled set. This forest serves a dual role: it acts both as the surrogate model used for final prediction and as the source of the task-aware geometry used for selection.
- **Step 2 – Map Stage**: every testbed configuration in the candidate pool and in the labeled set is passed through the forest and represented by the tuple of leaves it reaches across the trees – the "leaf space" in which behavioral similarity is defined.
- **Step 3 – Score Stage**: for each configuration candidate x, we compute a particular similarity score that we term "herding score": $S(x) = \Sigma_i \left(P_{pool}(i) - P_{labeled}(i)\right)$, summed over the leaves i in which x lands, where P_{pool} and $P_{labeled}$ are the empirical leaf-occupancy distributions of the pool and the labeled set respectively. A candidate scores highly when it falls into leaves where the labeled set under-represents the pool.
- **Step 4 – Select Stage:** We greedily pick $x^{next} = \arg\max_x S(x)$ for measurement, and the resulting (x^{next}, y^{next}) pair is appended to the labeled set.
- **Step 5 - Refit Stage:** To keep the surrogate's geometry aligned with the growing dataset, the Random Forest is periodically refit every E samples (here, we chose E = 10 as a practical compromise), after which the loops returns to the Map Stage with the updated leaf structure.

Figure 9 makes explicit why this selection rule reduces to a histogram-matching problem. On the left, the learned feature space is partitioned by the Random Forest into leaves L_1,...,L_6, each corresponding to a region of configurations that the surrogate treats as behaviorally equivalent. On the right, each bar represents one leaf: the dashed red outline marks the target mass assigned to that leaf by the full pool, the solid fill marks the labeled mass currently present, and the light pink portion is the residual deficit that PKH seeks to close. Selecting a configuration from the leaf with the largest **deficit** reduces the gap between the two distributions by exactly one sample's worth of mass per tree, which is precisely the greedy update of kernel herding under the leaf-indicator kernel induced by the Random Forest. The practical consequence is that the selection step is computationally trivial: it amounts to maintaining per-leaf counts and picking from the leaf with the largest pool-minus-labeled imbalance, despite being grounded in a kernel-herding argument over a learned representation.

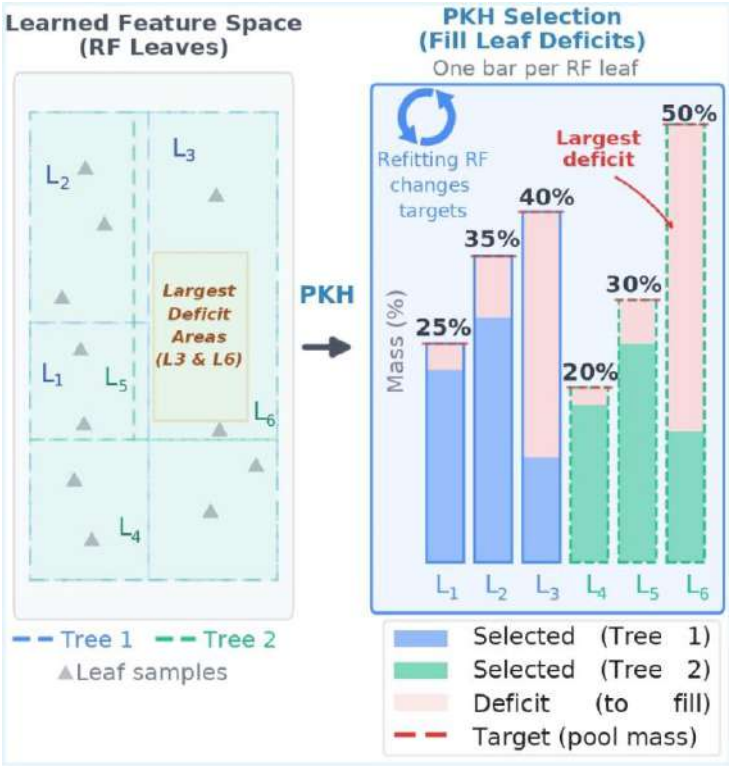


Figure 9: PKH fills RF leaf deficits to match pool mass.

3.3.3.2 Datasets

The evaluation is presented to assess whether the proposed sampling strategy addresses the problem defined above: reducing the number of testbed runs required to build an accurate surrogate over a given testbed feature space. In the project workflow, the acquisition method would be invoked before launching a full experimentation campaign, in order to decide which configurations should actually be measured. The purpose of this evaluation is therefore to replay this situation offline and quantify how much predictive accuracy can be obtained when only a small fraction of the available configurations is selected for measurement.

We evaluate the proposed sampling strategy on four performance datasets released by EURECOM[38] . Each dataset consists of measured testbed runs of a distinct cloud-native service collected over a mixed-type testbed parameter space in a different compute environment, with each dataset covering a different service and deployment setting: a Golang Web server, a Python web server, the RabbitMQ message broker, and the OpenAirInterface 5G Core AMF. Each row corresponds to one testbed configuration that was actually executed and measured. The datasets are therefore suitable for evaluating budget-aware testbed sampling because they provide finite measured pools over which different acquisition strategies can be compared under identical conditions.

The datasets were selected for three reasons: first, they contain real measured performance values rather than simulator-generated estimates, which makes them representative of the type of data required by the Experimentation on Demand workflow: this provides a reference against which any sampled subset can be scored, and the surrogate is fitted on the selected fraction and evaluated on all remaining configurations, giving an exact measure of the accuracy lost by not measuring the full space. Second, they cover different workload types, including network-function, web-server and messaging workloads, allowing the evaluation to test whether the sampling strategy behaves consistently across different response surfaces. Third, they involve mixed testbed parameters such as: **ram_limit**, **cpu_limit**, and the number of requests **n**, while the two web-server datasets additionally include the maximum concurrency **c.** The mixed-type parameter spaces and the four distinct services further ensure that the results reflect the heterogeneous conditions the acquisition method is intended for, rather than a single system or a purely numeric space.

This reflects the type of mixed parameter spaces encountered in practical testbed campaigns, and the domains covered by the datasets are representative of the project's target infrastructure since the datasets are cloud-native and include a 5G core network function.

As summarised in Table 12, the datasets vary in size from 15,308 to 58,763 measured configurations and involve either three or four tunable testbed parameters. All datasets include **ram_limit**, **cpu_limit**, and the number of requests **n** while the two-webserver datasets additionally include the maximum conccurency parameter **c**.

Table 12. Dataset characteristics

Dataset	# Testbed Configurations	# Testbed Parameters	Parameter names
amf-performance	27413	3	ram_limit, cpu_limit, n
golang-web-server-performance	58763	4	ram_limit, cpu_limit, n, c
python-web-server-performance	15308	4	ram_limit, cpu_limit, n, c
rabbitmq-performance	22556	3	ram_limit, cpu_limit, n

The metrics of interest are the predictive accuracy of the surrogate and the number of testbed runs required to reach that accuracy. Predictive accuracy is measured using Mean Absolute Percentage Error (MAPE) on configurations that were not selected for training. The measurement cost is represented by a sampling budget, expressed as a percentage of the full dataset. A sampling strategy is therefore considered better if it achieves lower MAPE with fewer selected testbed runs.

3.3.3.3 Evaluation

For each dataset, we simulate a sequential budgeted sampling scenario in which configurations are selected one at a time and without replacement from the unlabeled pool for measurement. Each run starts from a five-sample random warm start, after which the compared selection strategies incrementally expand the labeled set until reaching training budgets ranging from 1% to 10% of the full dataset. Active learning requires enforcing the following condition: $n_{sampled} \ll |dataset|$, as the goal is to reduce the predictive error with a minimum sampling budget. Henceforth, we are setting the maximum sampled train set proportion to 10%, a usual quantity used to threshold significance.

At each budget, a surrogate model is trained on the selected configurations and evaluated on the remaining unlabeled testbed configurations using MAPE (Mean Absolute Percentage Error). We compare standard i.i.d. random sampling against our sampling strategy PKH and its two epistemically weighted variants (these variants incorporate a variance component in the acquisition function to account for the fact that some regions may be more difficult to learn than others). The two variants modify PKH by replacing the uniform pool mass with epistemic weights estimated respectively from Random Forest committee variance and infinitesimal-jackknife variance [39] , following established uncertainty-estimation techniques for Random Forest.

The comparison therefore evaluates three aspects of the proposed method: whether PKH reduces surrogate prediction error compared with random selection under the same testbed budget, whether the method remains effective across different workload types and parameter-space sizes, and whether adding epistemic weighting improves the selection of testbed runs in regions that are more difficult for the surrogate to learn.

The resulting learning curves, shown in the figures below should be interpreted as an offline replay of the project's surrogate-based data generation workflow. Lower MAPE at a fixed budget indicates that the acquisition strategy selected more informative testbed runs. Conversely, reaching the same MAPE with a smaller budget indicates that fewer physical or digital twin testbed executions would be required in an online Experimentation on Demand campaign.

3.3.3.4 Results

Figure 10 reports MAPE as a function of training-set proportion on the amf-performance dataset for the lat50 and lat90 targets. Clearly, PKH-based methods outperform random sampling most strongly in the low-budget regime. At budgets between 1% and 4%, random sampling shows both higher median error and substantially larger variability, whereas PKH and its weighted variants consistently achieve lower

and more stable MAPE. This confirms that actively matching coverage in the surrogate's learned feature space leads to more informative and representative measurement sets than random acquisition on the raw feature space. The two weighted variants, EW-PKH and EW-PKH-IJ, generally perform best in the low-budget regime: this suggests that combining coverage matching with variance weighting improves early-stage selection. As the training budget grows, all methods improve and the gap narrows, but PKH-based methods still converge faster and yield statistically significantly lower errors than Random Sampling.

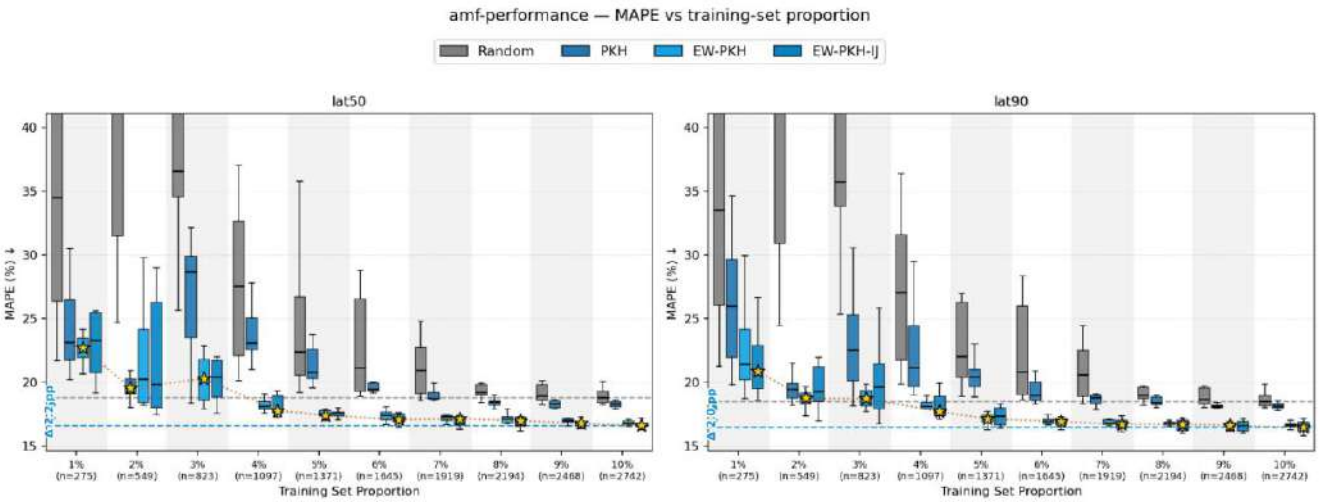


Figure 10: Prediction Error (in MAPE) of each iterative strategy in function of the train set size (in % of dataset) for AMF Performance Dataset.

Other traditional active learning baselines, among which we can find tree-based active learning **RT-AL[33]** , **Query-by-Committee[30]** (here, with Random Forest Ensemble Disagreement) or **MaxHerding[34]** have been excluded. The MAPEs of these methods (RT-AL, Query-By-Committee) appeared to be orders of magnitude higher than PKH-based methods and random sampling, above what is operationally reasonable. As for MaxHerding, the method displayed huge memory consumption, leading to frequent OOM errors.

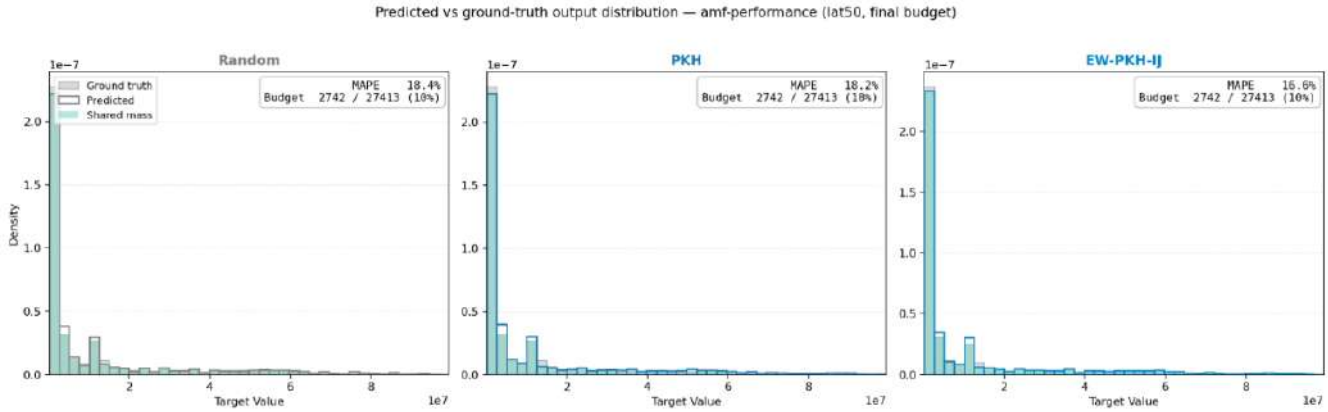


Figure 11: Superposition of ground truth response error histogram (in grey) against the empirical error histogram (in cyan) for each sampling strategy at final measurement budget (10% of dataset size) for AMF Performance Dataset and lat50 objective

Figure 11 compares the predicted and ground-truth output distributions for the amf-performance dataset (lat50) at the final 10% budget. All methods recover the strongly right-skewed shape of the target distribution, but the PKH-based approaches show slightly better alignment with the true distribution (especially at the right residual bars), with EW-PKH-IJ achieving the closest match and the lowest MAPE (16.6%) compared with PKH (18.2%) and random sampling (18.4%). Similar results have been observed in practice for the lat90 objective.

The same trend also appears on **golang-web-server-performance** dataset (Figure 12, Figure 13),

suggesting that the benefit of PKH-based selection generalizes beyond a single dataset. At the final budget, PKH-based methods again better match the target distribution and outperform random sampling. Similar results are observed for l**at90**-objective histograms as well.

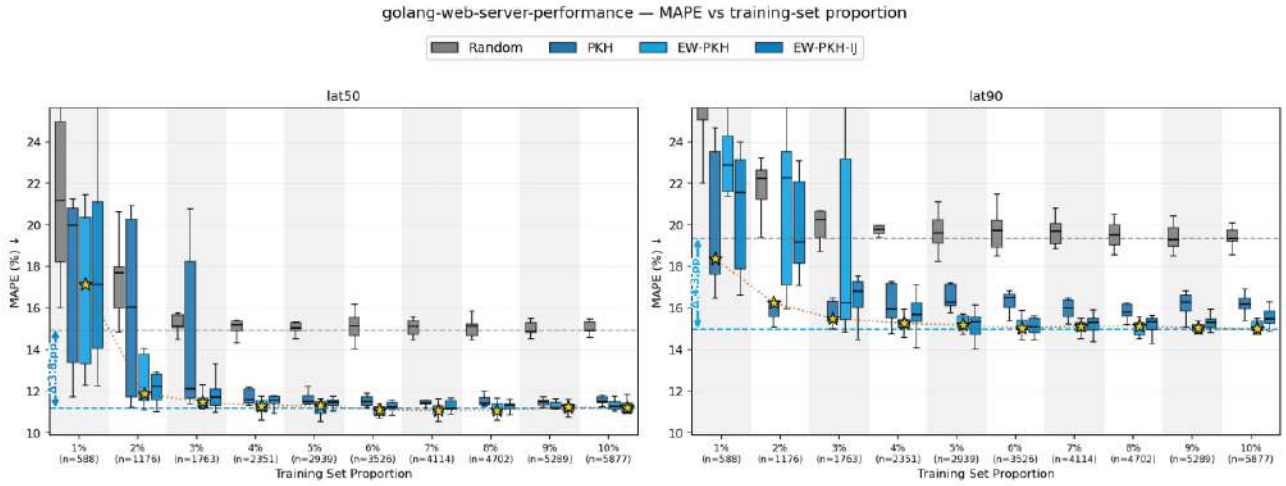


Figure 12: Prediction Error (in MAPE) of each iterative strategy in function of the train set size (in % of dataset) for Golang Webserver dataset

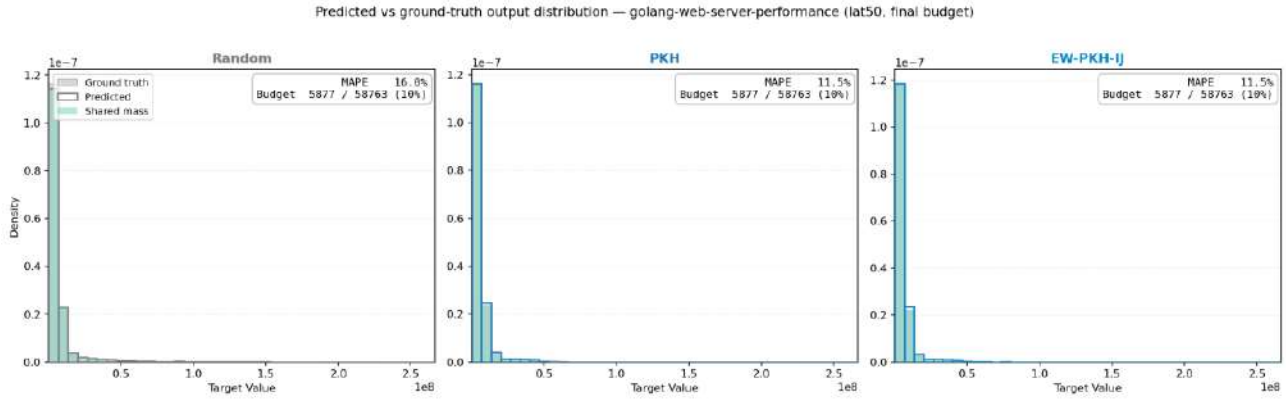


Figure 13: Superposition of ground truth response error histogram (in grey) against the empirical error histogram (in cyan) for each sampling strategy at final measurement budget (10% of dataset size) for Golang Webserver Dataset and lat50 objective.

3.3.4 Next Steps

This section introduced a surrogate-based data generation approach for budget-aware testbed experimentation. The main problem addressed is that exhaustive execution of all possible workload and infrastructure configurations is not feasible, because the testbed feature space grows combinatorially while each run consumes time, compute resources and testbed availability. The proposed approach therefore selects a limited but informative subset of real testbed configurations to execute, uses the resulting measurements to train or update a Random Forest surrogate model, and applies Predictive Kernel Herding (PKH) to identify configurations that are under-represented in the surrogate's learned feature space. The offline evaluation on four EURECOM performance datasets shows that PKH and its epistemically weighted variants generally achieve lower and more stable MAPE than random sampling, especially in low-budget regimes, while better matching the output distribution at the final budget. These results support PKH as a practical acquisition policy for reducing the number of required testbed runs while preserving surrogate accuracy for downstream tasks such as HPO, model validation, workload placement and further experimentation on demand.

The next step is to move the proposed PKH sampling strategy from an offline evaluation prototype to an operational component of the project experimentation pipeline. The first priority is to implement PKH as an acquisition strategy within ADO, IBM's Accelerated Discovery Orchestrator, so that it can be

invoked directly during automated experimentation campaigns. In this integration, PKH should expose a clear interface: the input will be a candidate testbed parameter space, any already available measurements, the target KPI, the measurement budget and the surrogate configuration. The output will be an ordered list of configurations to execute as testbed runs.

Once integrated into ADO, PKH will be connected to the Experimentation on Demand workflow described in the previous sections. ADO wil use PKH to select the next configurations to evaluate, pass these configurations to the relevant physical or digital-twin testbed execution layer through the Adaptation Layer, collect the resulting KPI measurements, store the measurements and metadata in the 6G Data Lake and 6G Data Space, and update the surrogate model after each acquisition round. This will make it possible to use PKH not only as an offline sampling method, but also as a practical acquisition policy for real HPO and surrogate-based data generation campaigns.

Further experiments will then be carrried out to validate the operational benefits of this integration. These experiments should compare PKH against the default ADO acquisition strategies and against standard random or space-filling baselines under the same measurement budgets. The main quantities to report will be the reduction in required testbed runs, the surrogate prediction error at fixed budget, the quality of the resulting output-distribution coverage, and the robustness of the method across different workloads, KPIs and parameter-space sizes.

A second operational step will be to evaluate how PKH behaves when existing measurements are reused from the 6G Data Lake or when the testbed feature space changes. This is important for the project workflow because new campaigns will not always start from an empty dataset. The implementation should therefore support warm-starting from previous measurements, detecting regions of the parameter space with insufficient coverage, and launching additional acquisition rounds only where new measurements are needed.

3.4 Conclusion

This section presented the initial progress on on-demand dataset generation for Objectives 1,2 and 3. The work addresses a central challenge of AI/ML experimentation on 6G testbeds: enabling users and project components to obtain relevant, training-ready datasets without requiring exhaustive manual configuration of heterogeneous experimental environments. Two complementary approaches were presented. The first approach, intent-based data generation, starts from a natural-language user request, searches for existing datasets in the Dataspace, and, when no suitable dataset is available, translates the user intent into an executable experiment workflow. The second approach, surrogate-based data generation, starts from a candidate testbed parameter space and a limited measurement budget, and selects the most informative configurations to execute so that accurate surrogate models can be trained with fewer real testbed runs.

The intent-based workflow reduces the entry barrier for non-expert users by hiding the complexity of testbed-specific APIs, scenario catalogues, experiment configuration schemas and KPI collection pipelines, The proposed agentic architecture combines a Chatbot Agent, Search Agent, Experiment Agent, Translation Agent and Executor Agent, supported by guardrails, metadata extraction, intent-completeness checking, semantic scenario matching and testbed-specific translation tools. The preliminary implementation demonstrates how a user can express an experiment objective in natural language, receive matching scenarios, select one of them, launch the experiment, monitor its execution, and retrieve the generated dataset. This provides a first end-to-end demonstration of how the system can move from user intent to collected testbed data.

The surrogate-based workflow addresses the complementary problem of measurement cost. Since executing all possible configurations is infeasible in large mixed-type testbed parameter spaces, the section introduced Predictive Kernel Herding (PKH) as a budget-aware acquisition strategy. PKH uses the representation learned by a Random Forest surrogate to identify under-covered regions of the testbed feature space and select configurations that are expected to improve surrogate coverage and accuracy. The preliminary offline evaluation shows that PKH-based methods can reduce prediction error and variability compared with random sampling, especially when the measurement budget is small. This supports the use of PKH as a practical mechanism for deciding which real or digital-twin testbed runs should be executed when new measurements are required.

Together, these two workflows define a coherent on-demand dataset generation pipeline. Existing datasets are searched and reused first. When data are missing, intent-based agents can launch new experiments, and when the candidate configuration space is too large to explore exhaustively, PKH can prioritize the most representative measurements under a bounded budget. The next phase of the work should focus on integrating these components more tightly with the project DataOps/MLOps pipeline, the 6G Data Space, ADO, and the physical and digital-twin testbed execution layers. Future evaluation should quantify not only surrogate accuracy, but also reductions in user effort, testbed runs, measurement cost and time-to-dataset across representative experimentation campaigns.

4 Automated Model Tuning Across the ML Lifecycle

4.1 Overview

This section covers the model tuning automation layer. The goal is to streamline 6G testbed operations through MLOps and AutoML, and that requires workflows spanning the full model lifecycle: training, validation, fine-tuning, hyperparameter optimization, and performance benchmarking. The aim is to make 6G experimentation accessible to non-specialists and practical at scale. Three contributions are described in the following sections and represents parts of the MLOps Tools Stacks (deployed both in the 6G-DALI facility and on the different testbeds) depicted in Figure 1. HPO Optimization based on Accelerated Discovery Orchestration to be integrated into MLOps pipelines orchestrated by the 6G-DALI MLOps is described in Section 4.2. The AutoRL framework extends hyperparameter optimization to reinforcement learning settings, covered in Section 4.3. The cooperative transfer learning mechanism, which enables multi-model knowledge sharing across agents and environments, is described in Section 4.4. The datasets that feed into these tuning workflows are generated by the approaches in Section 2.

4.2 ADO for HPO in MLOps

This section describes the design and initial prototyping of an MLOps Actuator for the Accelerated Discovery Orchestrator (ADO) [40] tool. This component addresses one of the most resource-intensive phases of ML model development, hyperparameter optimization (HPO), by integrating ADO as an intelligent search engine that reduces trial complexity and accelerates convergence of experiments. As depicted in Figure 14 and described in Section 5.1 of Deliverable D3.1 [2] [1] , the HPO ADO Actuator is a component of the 6G-DALI advanced tool stacks designed to be invoked by the MLOps Meta-Orchestrator upon AIaaS requests on the MLOps Service Manager. This component, leveraging ADO capabilities of experiments reusability and exposing a standardized interface for HPO requests, encapsulate ADO's optimization logic behind an API-driven interface that will ease its integration in pipelines orchestrated by the 6G-DALI MLOps meta-orchestrator.

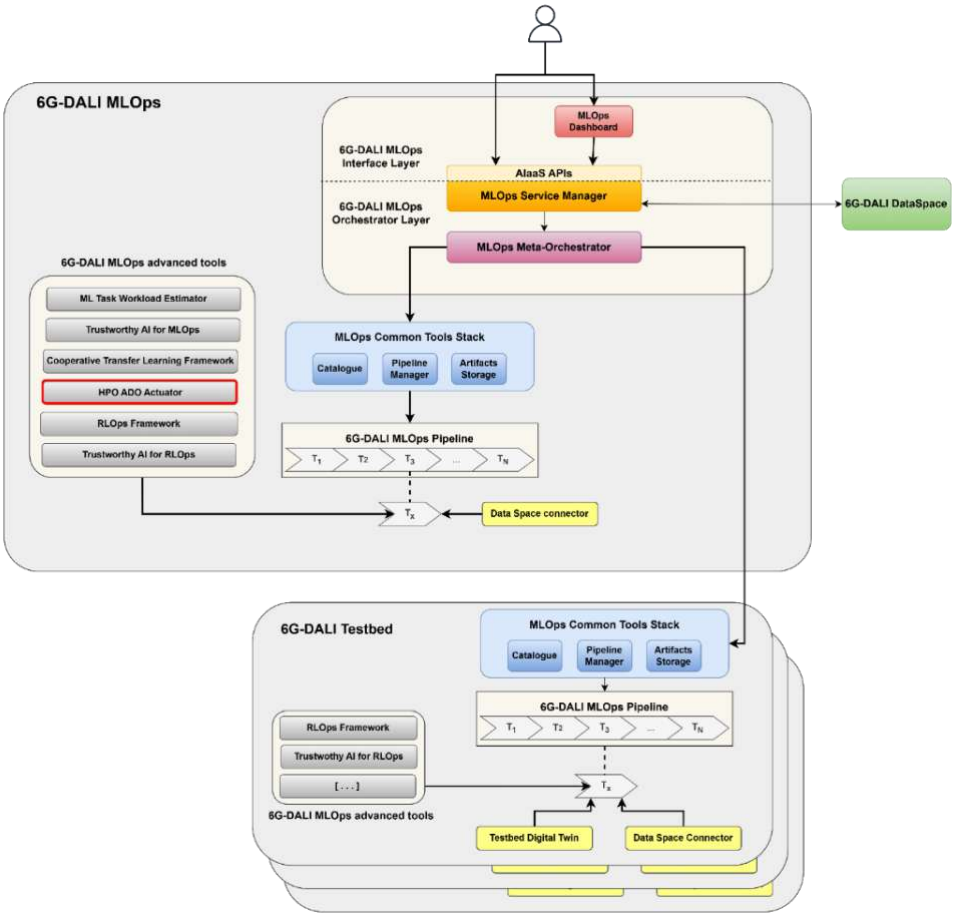


Figure 14: MLOps components functional decomposition - ADO for HPO

4.2.1 State-of-the-Art

HPO is an essential phase of the AI/ML lifecycle, crucial for maximizing model quality and performances (e.g. accuracy, precision, F1, inference time, artifact weights, …). Because optimal hyperparameters cannot be derived analytically, state-of-the-art methods rely on search strategies that iteratively evaluate performance across a defined parameter space. The dominant approaches include methodologies as Grid Search, Random Search, and Bayesian Optimization [41] . While modern frameworks (e.g., Ray Tune [42] , Optuna [43]) provide robust infrastructure for executing these trials in parallel, they remain largely agnostic to cross-experiment knowledge transfer and cross-experiment optimization. Historical evaluations from prior training jobs, even those with identical or structurally similar parameter spaces, are typically discarded.

4.2.2 Solution description

To address this gap, a unified, model-agnostic, HPO framework based on ADO, has been designed and prototyped. ADO provides a structured framework for hyperparameter optimization (HPO) by combining scalable experiment execution with modular components that cleanly separate optimization logic from experiment evaluation. Rather than treating each HPO run as an isolated task, this solution directly leverages historical experiment results to optimize new searches by transferring knowledge across experiments. This solution relies on the following ADO core concepts:

- **Actuators (experiment execution layer)**: Actuators define the evaluation function of the HPO process. Each actuator encapsulates the full experimental logic required to assess a given hyperparameter configuration, including model construction, training, and evaluation, and returns performance metrics such as accuracy or loss. Implemented as lightweight Python functions within ADO's plugin model, actuators can be executed in parallel across distributed compute resources. They are also automatically tracked for provenance and benefit from ADO's data reuse mechanism, ensuring that identical configurations are not recomputed unnecessarily.
- **Operators / Optimizers (search strategy layer)**: Operators are responsible for generating and adapting hyperparameter configurations throughout the optimization process. ADO integrates with optimization frameworks such as Ray Tune, enabling the use of a range of search strategies including random search, grid search, Bayesian optimization, and adaptive early-stopping methods. These optimizers iteratively propose new configurations based on past evaluation results, allowing the search process to efficiently explore and exploit the hyperparameter space.
- **Space (search space abstraction)**: A Space in ADO defines the structured representation of the parameters and variables over which optimization is performed. It specifies the domain of possible hyperparameter values (e.g., continuous ranges, categorical options, or conditional parameters), providing a consistent interface between optimizers and actuators. By formalizing the search space as typed entities, ADO enables both systematic exploration by optimization algorithms and compatibility with automated experimentation workflows, including AI-driven agents. This abstraction ensures that hyperparameter configurations are well-defined, composable, and reusable across experiment campaigns.
- **Execution, data, and orchestration layer:** This layer provides the infrastructure needed to run HPO workflows at scale. ADO leverages Ray to support distributed and parallel execution, allowing many trials (i.e., actuator invocations) to run concurrently. It also includes automatic data reuse to eliminate redundant computations and provenance tracking to record relationships between inputs, executions, and outputs.

Within this architecture, the actuator serves as the bridge between the optimization algorithm and the system being optimized. While operators determine which hyperparameter configurations to evaluate, the actuator determines how each configuration is evaluated in practice. Each trial in the HPO loop corresponds to a single actuator execution, producing the metrics that guide subsequent optimization steps. As such, the actuator embodies the objective function of the optimization problem, grounding the abstract search process in concrete experimental results. To seamless integrate ADO within 6G-DALI MLOps pipelines, a dedicated REST API and annotation SDK have been designed and prototyped, wrapping ADO within a standalone, deployable HPO service, as depicted in Figure 15. The platform is based on two concepts that abstracts ADO constructs:

1. HPO Campaign: represents an HPO campaign in terms of training function to be optimized, model parameters to be tested, and the function to be optimized.
2. HPO Run: represent an HPO experiment, subset of the campaign, in which the model is trained on a subset of the campaign space.

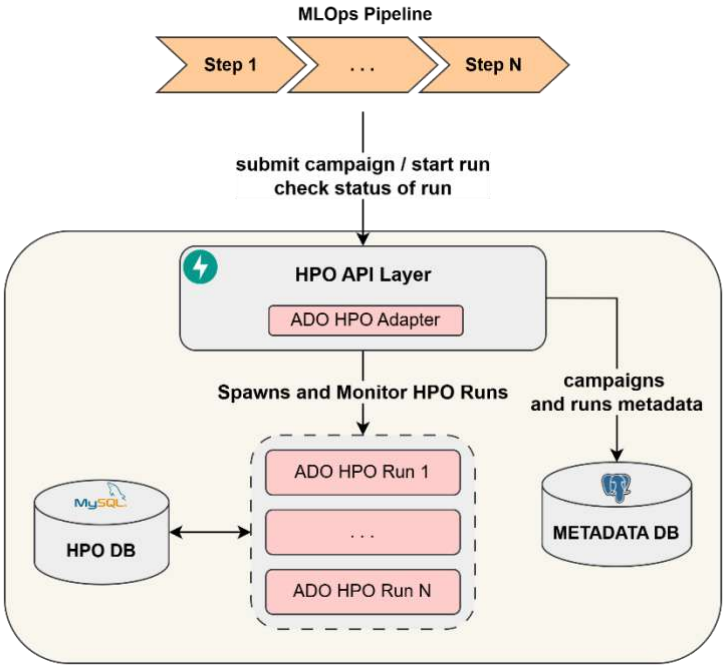


Figure 15: ADO for HPO high level architecture

In this, a two-database architecture driven by the distinct requirements of its two persistence concerns. (i) Campaign metadata, registered training functions, dataset references, run status, and results is stored in a PostgreSQL instance managed by the platform. (ii) Trial-level measurement data, specifically the (entity, measurement) pairs that ADO uses to detect and skip already-evaluated configurations, is stored in a separate MySQL instance managed exclusively by ADO; All execution environments connect to the same MySQL instance, which is what makes ADO's cross-experiment knowledge transfer effective across independent runs: a configuration already measured in one run is visible to all subsequent runs operating over the same space, and will not be re-executed. The API layer sits between the MLOps pipelines and these two stores. Its role is that of a translator and lifecycle manager rather than an HPO engine: it accepts an annotated training function and dataset at registration time, parses the @parameter and @objective decorators to extract the search space and optimization objective, and generates a pip-installable ADO @custom_experiment package together with the corresponding discovery-space configuration, without requiring the user to author any ADO-specific YAML. When a run is started, the API returns a run identifier immediately and delegates execution to a background worker, which can be a local process, a Docker container, or a Kubernetes Job depending on the deployment context. The actual optimization, selecting which configurations to evaluate, managing parallel trial execution via Ray Tune, and deciding when to reuse prior measurements, is performed entirely by ADO. The endpoints that are exposed by the API layer are reported in Table 13.

Table 13. HPO - High Level API endpoints

Endpoint	HTTP Method	Description
/register-hpo	POST	Handles campaign registration, parsing decorators, and generating the ADO package.
/start-hpo	POST	Submits asynchronous runs with optional per-run space subsetting.
/hpo-result/{run_id}	GET	Polls the status and results of a specific run.
/optimizers	GET	Discovers supported search algorithms available for optimization.
/hpo/{run_id}/cancel	DELETE	Explicitly cancels an ongoing run.

4.2.3 Initial Results

To validate the feasibility of the proposed approach, an initial proof-of-concept test was carried out using ADO for HPO trough the intermediate API Layer. The first test focused on validating the usability and performances gain in using ADO for optimizing a scikit-learn Random Forest [44] model trained on the standard Wine dataset. The experimental setup was defined as follows:

- **Space**: the search space was constrained to n_estimators (10, 50, 100, 200), max_depth (2, 4, 8, 16, 32), and min_samples_split (2, 5, 10), as depicted in Figure 16 - left.
- **Goal and Search Strategy**: the defined goal is to maximize the model's cross-validated accuracy. The algorithmic approach utilizes a RandomWalk module configured to sample and evaluate exactly 20 distinct parameter configurations, as depicted in Figure 16-right.
- **Actuator**: implemented as a custom Python function (wine_rf_optimizer) and wrapped with ADOs @custom_experiment decorator. Registered as a plugin via pyproject.toml, this function acts as the execution bridge: it receives the sampled hyperparameters, initializes the RandomForestClassifier, evaluates its performance using 3-fold cross-validation (cv=3), and returns the mean accuracy metric to the ADO orchestrator.

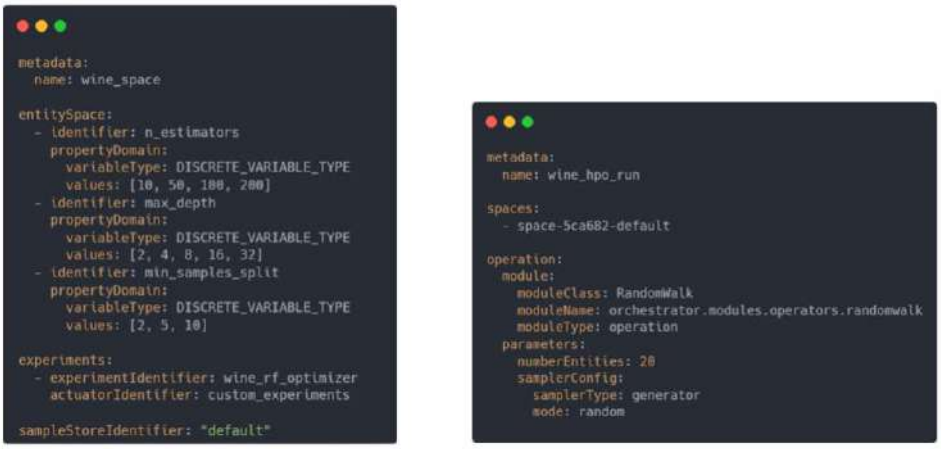


Figure 16: ADO-HPO (left) search space, experiment definition and (right) operation definition

During execution, the ADO completed 20 training iterations, as shown in Figure 17. Across those runs, the ADO retrieved and reused results from three previous experiments to evaluate new configurations, demonstrating the cross-experiment knowledge transfer required for HPO in 6G-DALI and avoiding the corresponding redundant computations. The search converged on an optimal configuration of 100 estimators, a max_depth of either 16 or 32, and a min_samples_split of 2.

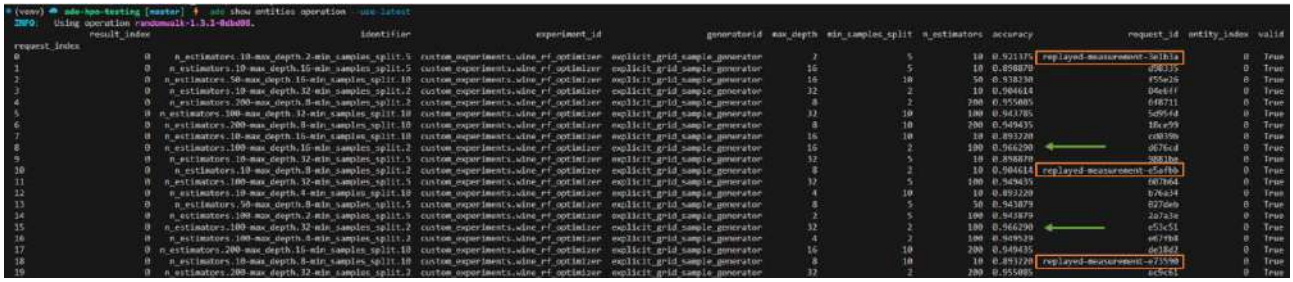


Figure 17: ADO-HPO proof-of-concept results

Building on this validation of ADO's core reusability property at the CLI level, the REST API and annotation SDK were developed to encapsulate this workflow behind a standardised, deployment-ready interface accessible to the 6G-DALI MLOps pipelines. The resulting platform was validated end-to-end using a scikit-learn Random Forest classifier on the Iris dataset. A campaign was registered by uploading an annotated training function using the SDK, depicted in Figure 18, declaring n_estimators, max_depth, min_samples_split, and criterion as search dimensions via @parameter decorators, and val_accuracy as the optimisation objective via @objective, alongside the dataset file.

```python
from hpo_sdk import parameter, objective

@parameter("n_estimators", int, low=10, high=300)
@parameter("max_depth",    int, low=2,  high=20)
@parameter("criterion",    str, choices=["gini", "entropy"])
@objective("val_accuracy", direction="maximize")
def train(n_estimators, max_depth, criterion, dataset_path):
    ...
    return {"val_accuracy": score}
```

Figure 18: HPO SDK Example

The developed platform parsed these annotations to generate the corresponding ADO custom experiment package and discovery space automatically, without any manual YAML authoring. Fifteen configurations were then evaluated via the Ray Tune random search operator, converging on a best configuration achieving 96.7% cross-validated accuracy. The platform's per-run space subsetting feature was additionally validated by constraining max_depth to [6] [11] and fixing criterion to entropy for a second run over the same campaign; ADO correctly restricted exploration to the narrowed subspace while retrieving all previously measured configurations from the shared MySQL metastore, confirming that the cross-experiment knowledge transfer demonstrated in the Wine dataset experiment is preserved end-to-end through the API and Kubernetes deployment layers. The registration and run execution workflows are illustrated in Figure 19 and Figure 20 respectively.

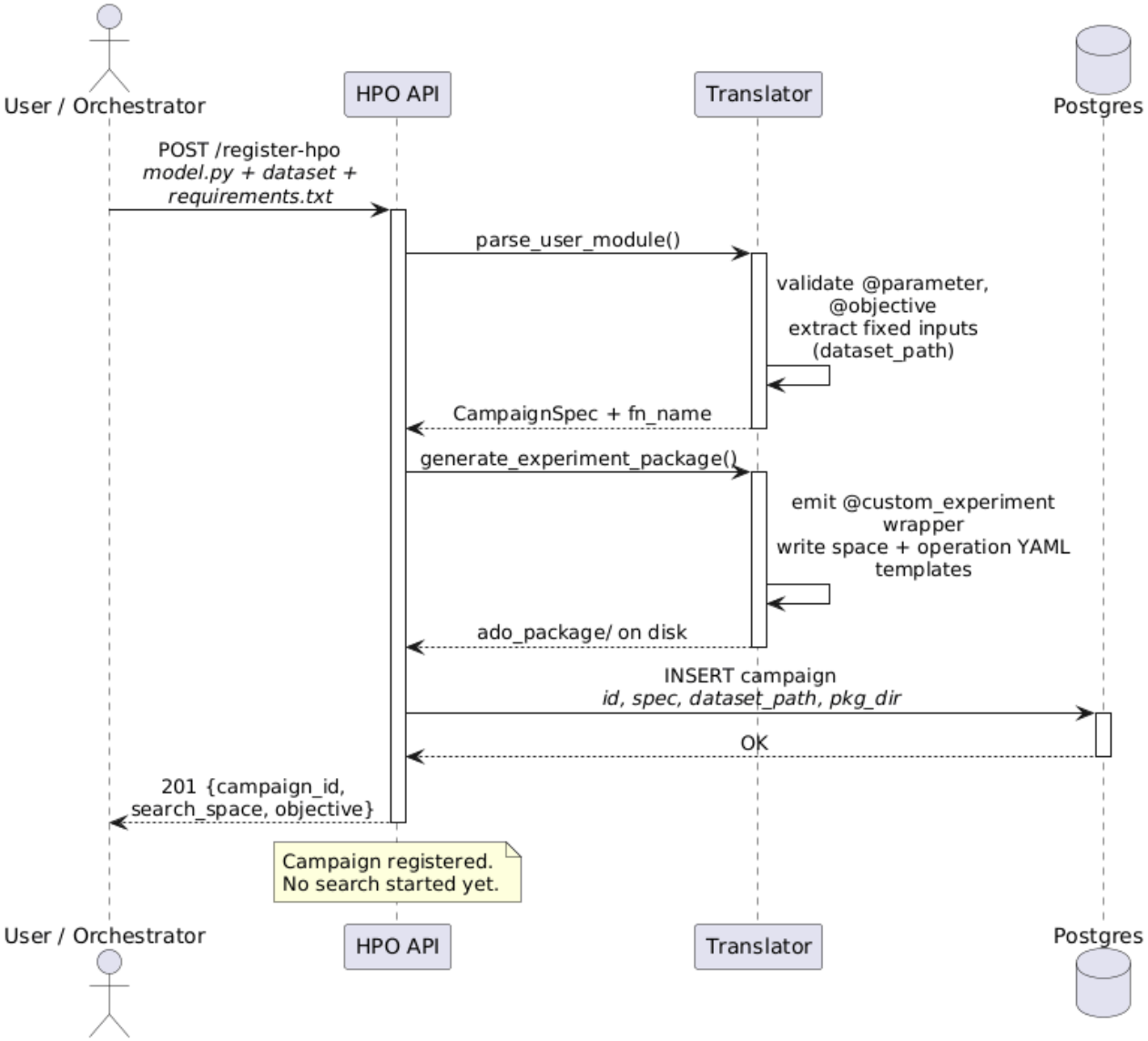


Figure 19: HPO module - Campaign registration

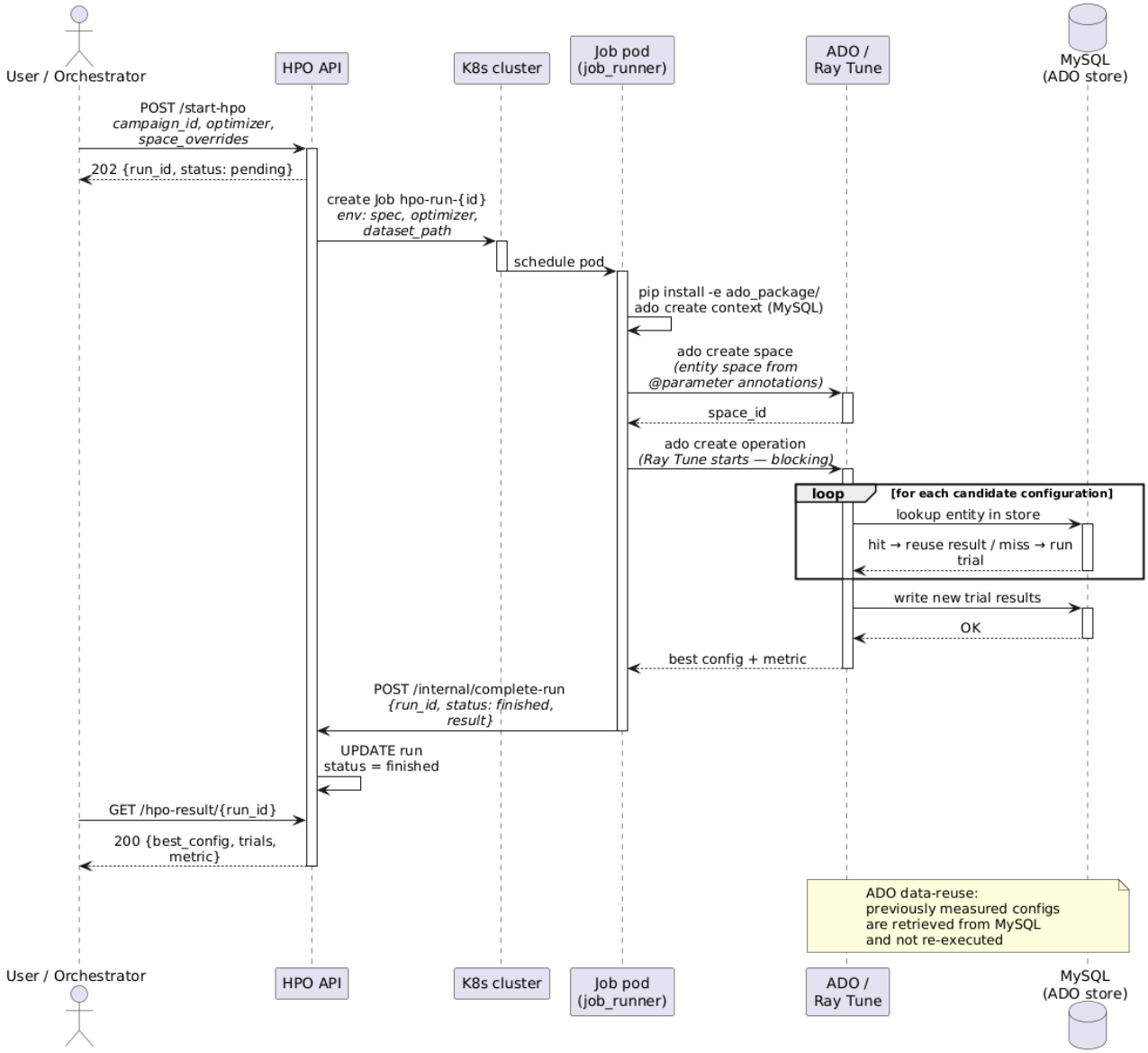


Figure 20: HPO module - HPO trial

4.2.4 Next Steps

With the REST API, annotation SDK, and Kubernetes deployment prototyped and validated, subsequent work will focus on three directions. First, the platform will be integrated with the 6G-DALI MLOps Meta-Orchestrator: the registration and run endpoints will be mapped to the AIaaS HPO interface defined in D3.1[2] , enabling the HPO Actuator to be invoked programmatically within MLOps pipelines orchestrated by the 6G DALI Meta Orchestrator, with response schemas and authentication mechanisms aligned with the broader 6G-DALI architecture. Second, the annotation SDK will be published as a standalone pip-installable package, enabling its adoption as an import dependency in MLOps pipeline components without requiring direct interaction with the REST API. Third, distributed trial execution will be explored via KubeRay: replacing the current single-node Ray execution within each Kubernetes Job pod with a managed RayCluster would enable individual trials to be parallelised across a multi-node cluster, which is the natural evolution as the platform is deployed on larger 6G-DALI testbed infrastructure.

4.3 Automated Reinforcement Learning and RLOps

Reinforcement Learning (RL) has demonstrated strong potential in solving complex sequential decision-making problems. However, its practical adoption remains limited due to its high sensitivity to hyperparameters and design choices, which typically require extensive manual tuning and expert knowledge. This challenge is particularly pronounced in dynamic and large-scale domains such as

telecommunications, where performance and reliability are critical. Automated Reinforcement Learning (AutoRL) has emerged to address these limitations by automating the optimization of the RL pipeline, drawing inspiration from the success of Automated Machine Learning (AutoML). Unlike supervised learning, RL introduces additional complexity due to its sequential nature, non-stationary data, and the dependency between the agent and the data it generates. To handle this, AutoRL formulates the problem as a bi-level optimization process, where the inner loop trains the agent and the outer loop optimizes higher-level design choices such as hyperparameters and model configurations. In this context, the application of AutoRL and Hyperparameter Optimization (HPO) within the 6G-DALI framework is explored, aiming to enable more efficient, scalable, and reproducible RL workflows in practice. Referring to Figure 1, this maps to the MLOps Tools Stack component within the MLOps block.

4.3.1 State-of-the-Art

AutoRL research has mainly focused on efficient Hyperparameter Optimization (HPO), online adaptation, and broader automation of the RL training pipeline. Bayesian optimization[45] methods have been widely used to guide the search toward promising hyperparameter configurations, while multi-fidelity methods reduce the computational cost by evaluating many configurations with small budgets and allocating more resources only to the most promising ones. In parallel, dynamic approaches adapt hyperparameters during training by maintaining a population of agents, exploiting stronger agents, and perturbing weaker ones[46] . More recent AutoRL directions extend beyond static HPO toward online hyperparameter tuning, meta-gradient methods, algorithm selection, architecture search, and robust evaluation across multiple random seeds. Therefore, the current state of the art is moving from manually configured RL agents toward automated, adaptive, and reproducible RL pipelines.

In the context of 6G-DALI, the proposed AutoRL workflow is designed as a reusable optimization layer that can support different levels of automation in reinforcement learning. In its current implementation, the workflow focuses on HPO by integrating and comparing representative optimization methods under a unified experimental protocol. More broadly, the same structure can be extended to optimize additional RL design choices, including algorithm selection, neural network architecture, training budgets, reward-related parameters, and evaluation strategies.

4.3.2 Solution Description

Hyperparameter Optimization (HPO) plays a central role in AutoRL, as the performance of RL agents is highly sensitive to the selection of training configurations. HPO is formulated as a black-box optimization problem, where the objective is to identify hyperparameter configurations that maximize the expected return of the agent. To this end, four state-of-the-art HPO methods were evaluated:

1. **Bayesian Optimization with HyperBand (BOHB)[45] :** BOHB combines Bayesian Optimization with HyperBand, enabling efficient exploration of the search space while leveraging multi-fidelity evaluations. It uses a probabilistic model to guide the search toward promising configurations while allocating computational resources adaptively. BOHB is particularly effective in reducing evaluation cost by terminating poorly performing configurations early.
2. **Differential Evolution HyperBand (DEHB) [46] [47] :** DEHB integrates Differential Evolution with HyperBand, combining evolutionary search with multi-fidelity optimization. It maintains a population of candidate configurations and evolves them through mutation and crossover, while HyperBand ensures efficient resource allocation. This allows DEHB to balance global exploration with computational efficiency.
3. **Sequential Model-based Algorithm Configuration (SMAC) [48] :** SMAC is a Bayesian optimization-based method that models the performance of configurations using surrogate models (typically random forests). It iteratively selects promising configurations based on an acquisition function, balancing exploration and exploitation. SMAC is well-suited for high-dimensional and noisy optimization problems such as RL.
4. **Population-Based Training (PBT):** PBT is an online, adaptive optimization method that trains multiple agents in parallel. Periodically, poorly performing agents are replaced with better-performing ones, while their hyperparameters are perturbed. Unlike the other methods, PBT

enables dynamic adaptation of hyperparameters during training, making it particularly suitable for non-stationary RL environments.

The experiments were conducted on a custom RL environment, **MECPower-v0 [49]** , which models resource allocation in a Mobile Edge Computing (MEC) server within a 6G network setting. The environment simulates a scenario where multiple services compete for limited CPU resources, and the agent must dynamically allocate resources to balance **performance** (delay) and **energy efficiency** (power consumption). At each timestep, the agent observes the current system state including CPU demand, allocation, delay, and power per service and outputs a continuous allocation decision. The reward function captures a trade-off between efficient resource utilization and penalties for violating delay, power, and capacity constraints. This setup reflects realistic telecom challenges, where systems must operate under strict Quality of Service (QoS) and energy limitations.

Across all experiments, a **Proximal Policy Optimization (PPO)** agent was used as the underlying learning algorithm. The optimization process focused on identifying high-performing configurations of key hyperparameters, including the learning rate, discount factor (γ), Generalized Advantage Estimation (GAE λ), clipping range, entropy coefficient, rollout length (n_steps), and batch size. All methods operate over a common predefined search space, ensuring a consistent basis for comparison, while a summary of these hyperparameters and their roles in the learning process is provided in Table 14. Each method was executed under a consistent experimental protocol, where multiple trials were performed per optimizer. Each trial corresponds to a unique hyperparameter configuration evaluated under a given computational budget (budget_steps). The multi-fidelity nature of several methods (BOHB, DEHB) allows early termination of poorly performing configurations while allocating more resources to promising ones.

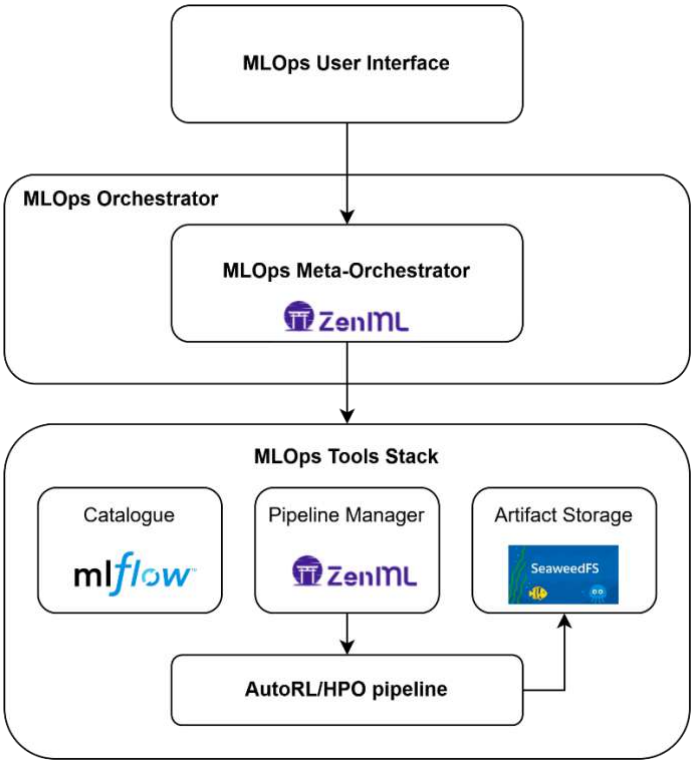


Figure 21: AutoRL/HPO design

As shown in Figure 21,the proposed AutoRL/HPO workflow is integrated within the 6G-DALI MLOps architecture as a pipeline managed by the MLOps Meta-Orchestrator. Through the MLOps User Interface, users can configure RL experiments by selecting the environment, optimization method, training budget, and evaluation settings. The AutoRL/HPO pipeline is then executed using the common MLOps tool stack, while the produced configurations, metrics, trained agents, and best-performing results are stored in the experiment tracking and model registry layer. This structure enables reproducible experimentation,

comparison of HPO strategies, and future reuse of optimization artifacts across different RL algorithms, architectures, and deployment scenarios.

Table 14. Hyperparameters and corresponding search space.

Name	Description	Range	Sampling
Learning Rate	Controls the step size during gradient updates; critical for convergence speed and stability	[1e-5, 3e-4]	Log-uniform
Discount factor (γ)	Determines the importance of future rewards; higher values emphasize long-term returns	[0.95, 0.999]	Uniform
Generalized Advantage Estimation (GAE λ)	Controls the bias–variance trade-off in advantage estimation	[0.8, 0.99]	Uniform
Clipping range	Limits policy updates in PPO to ensure stable learning	[0.1, 0.3]	Uniform
Entropy coefficient	Encourages exploration by penalizing deterministic policies	[1e-6, 1e-2]	Log-uniform
Rollout length (n_steps)	Number of environment steps collected before each update	{128, 256, 512}	Categorical
Batch size	Number of samples used per training update	{64, 128}	Categorical

4.3.3 Initial Results

Performance evaluation is based on the **mean episodic return**[1], defined as the average cumulative reward obtained per episode over multiple evaluation runs, which serves as the primary optimization objective, alongside the **standard deviation of returns** which measures the variability of performance across episodes and is used to capture stability. Figure 22 presents the convergence behavior of the evaluated HPO methods as a function of trial order, where each trial corresponds to a distinct hyperparameter configuration evaluated during the optimization process.

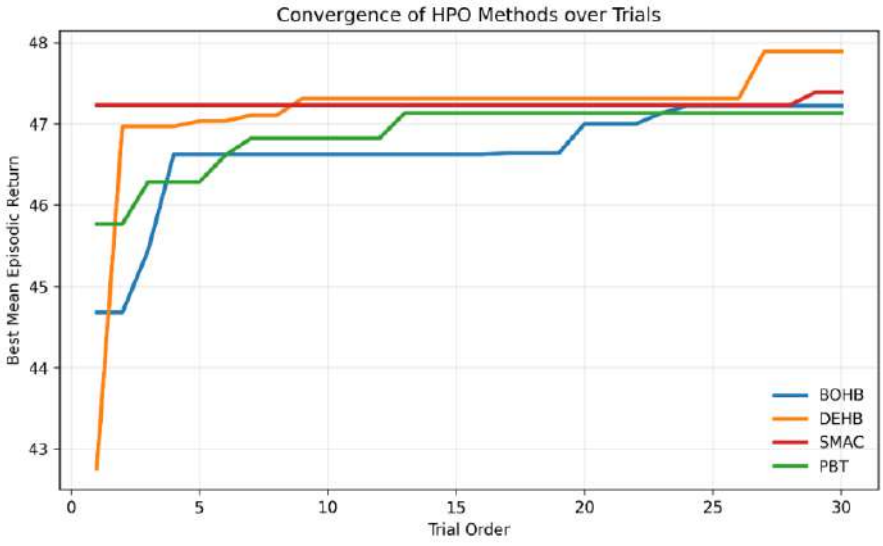


Figure 22: Convergence of HPO methods across trial order.

All methods reach comparable performance levels, indicating the presence of multiple high-quality configurations in the search space. **DEHB** achieves the highest observed performance, despite a weaker initial configuration, suggesting strong exploration capabilities. **BOHB** demonstrates a gradual and

[1] https://apxml.com/courses/intro-to-reinforcement-learning/chapter-1-foundations-reinforcement-learning/episodic-vs-continuing-tasks

stable improvement over time, while **PBT** improves quickly in early trials but plateaus at later stages. In contrast, **SMAC** starts with strong initial performance, likely due to its model-based approach, and shows only marginal improvement throughout the search. Overall, the results highlight differences in convergence dynamics and exploration behavior across methods, while maintaining similar final performance.

4.3.4 Next Steps

As part of the next steps, the scope of the current work will be extended toward a more complete AutoRL pipeline. While the present implementation focuses primarily on hyperparameter optimization, future work will incorporate additional dimensions of AutoRL, including the automated selection of RL algorithms and model architecture, as well as the optimization of agent configurations such as reward functions and objective formulations. This broader approach aims to move beyond isolated tuning and toward a holistic optimization of the entire RL pipeline, enabling more robust and adaptive solutions within the RLOps framework.

In parallel, the experimental workflow will evolve from the current local setup in Intracom's local server, which enabled rapid development and evaluation under controlled conditions, to a fully integrated deployment within the 6G-DALI framework. In this architecture, the optimization process will be managed by the MLOps Orchestrator and accessed through the MLOPs User Interface for experiment configuration and result visualization, supporting a more scalable and automated AutoRL pipeline.

4.4 Cooperative Transfer Learning

Cooperative Transfer Learning (CoTL) extends traditional transfer learning (TL) by enabling multiple independently trained machine learning (ML) models to collaborate through structured knowledge exchange. Instead of relying on centralised datasets or full model aggregation, CoTL allows models to share selected internal components, enabling performance improvements while preserving data locality and respecting the distributed nature of 6G environments.

Within 6G-DALI, CoTL is elevated from an algorithmic concept to a system-level capability embedded in the MLOps framework. Models contribute reusable knowledge artifacts, which are stored, indexed, and reused across experiments and over time. This enables a continuous learning process, where models can iteratively improve by leveraging knowledge generated by other experiments without direct interaction or data sharing. Referring to Figure 1, this maps to the MLOps Tools Stack component within the MLOps block.

4.4.1 State-of-the-art

Recent research has explored collaborative learning approaches that reduce the dependency on centralised training while still enabling knowledge exchange between distributed ML models. These approaches aim to address challenges related to excessive data movement, scalability, and privacy in distributed environments. Cartel [50] introduced a CoTL system for edge environments where locally trained models selectively exchange knowledge to improve adaptation and reduce communication overhead. Similarly, CLUE [51] proposed a framework for CoTL with neural networks based on the extraction and transfer of significant model parameters between distributed learners. Both works demonstrated that selective knowledge transfer can improve model performance without requiring centralised datasets or full model sharing.

Building on these approaches, 6G-DALI extends CoTL beyond an algorithmic mechanism into a managed MLOps capability. In particular, persistent knowledge catalogues, metadata-driven neighbor discovery, and pipeline-based orchestration for automated knowledge transfer and reuse are introduced. By integrating CoTL within the 6G-DALI MLOps framework, scalable, reproducible, and reusable CoTL is enabled across distributed experimenters and 6G testbeds.

4.4.2 Solution Description

CoTL is implemented as a managed service within the **6G-DALI MLOps framework**, combining persistent knowledge management with pipeline-driven orchestration. As presented in Figure 23, the system consists of the following components:

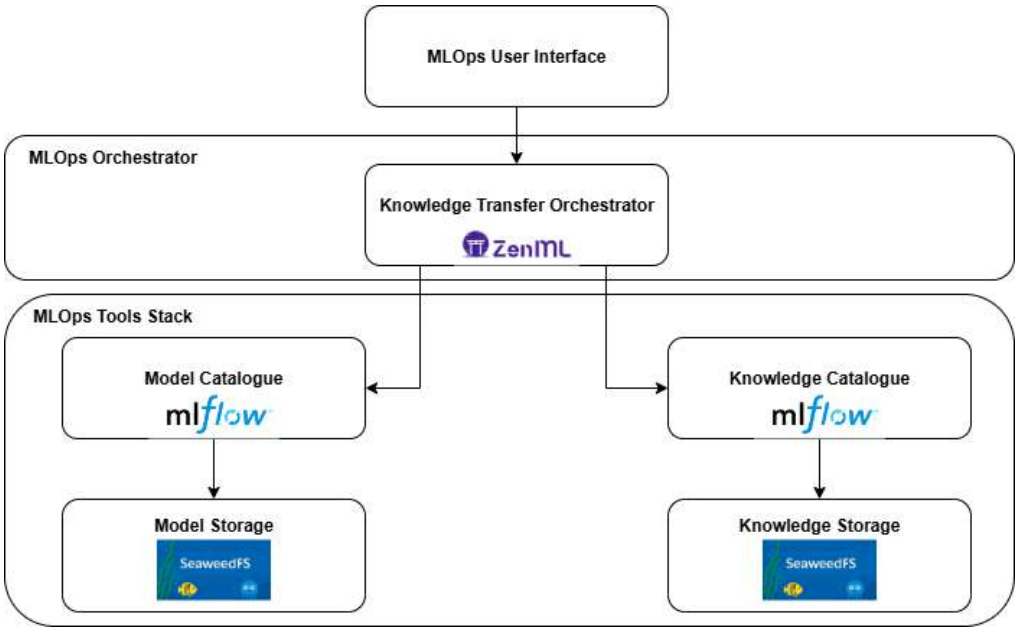


Figure 23: Cooperative Transfer Learning design

- **MLOps User Interface**: Allows experimenters to register models, trigger model improvement requests, configure parameters (e.g., number of knowledge units to replace), and visualise results.
- **Knowledge Transfer Orchestrator**: Implements the logic of CoTL through orchestrated ZenML2 pipelines.
- **Model Catalogue**: Lists original and improved models along with their metadata, including hyperparameters and evaluation metrics.
- **Knowledge Catalogue**: Lists reusable knowledge units extracted from models, enriched with metadata such as source model, feature schema, and performance metrics.
- **Model Storage and Knowledge Storage**: Handle the physical storage of models and extracted knowledge artifacts, respectively.

A key design choice is the separation between the catalogues of models and knowledge units. In the current design, this is realized using distinct experiment names within Mlflow[3], which are further organised per learning task and algorithm. More specifically, full models are stored in dedicated experiments (e.g., *cotl_models_rf_regressor*, corresponding to models trained for regression using the Random Forest algorithm), while extracted knowledge units are stored in separate experiments (e.g., *cotl_knowledge_rf_regressor*). This structure is consistently extended across different model types and algorithms, allowing the system to maintain clear isolation between model families, ensure compatibility of knowledge units, and support scalable management of heterogeneous learning workflows.

The CoTL process follows a structured workflow in which models are analyzed prior to registration to identify their characteristics and evaluate their performance. In parallel, knowledge units, such as decision trees extracted from ensemble models, are stored along with fine-grained metadata. When a model improvement request is issued, the system identifies relevant knowledge sources by discovering "neighbour" models with similar characteristics or complementary strengths. Knowledge transfer is performed selectively using a **partition-and-merge** strategy, where weaker components of the target model are replaced with stronger components retrieved from the knowledge catalogue. This allows incremental improvement while preserving the overall structure of the original model.

The dynamic behavior of the system is implemented through orchestrated pipelines, which ensure that

[2] https://www.zenml.io/

[3] https://mlflow.org/

all steps (from model onboarding to knowledge transfer and evaluation) are executed in a consistent, reproducible, and automated manner.

4.4.3 Initial Results

The implementation of CoTL in 6G-DALI will be realized through a set of pipelines orchestrated using ZenML. These pipelines formalise the CoTL workflow into reusable and parameterizable execution units. Two core pipelines have been designed:

1. **Model Registration Pipeline**: This pipeline supports onboarding of externally trained models into the CoTL ecosystem. It ensures that all models, regardless of origin are:
    a. Evaluated in a consistent manner,
    b. Registered with standardised metadata,
    c. Processed to extract reusable knowledge units,
    d. Integrated into the knowledge catalogue.
2. **Model Improvement Pipeline**: This is the core pipeline implementing the CoTL logic. Given a target model, it:
    a. Analyses model weaknesses,
    b. Identifies neighbouring models through metadata-driven discovery,
    c. Selects high-quality knowledge units from these models,
    d. Replaces weaker components of the target model and reconstructs the model using a partition-and-merge strategy,
    e. Evaluates the updated model and performs rollback, if necessary,
    f. Registers the improved model and its knowledge units.

These pipelines demonstrate that CoTL can be expressed as a fully automated workflow, enabling reproducibility and iterative refinement. While still at an early stage, the current design validates the feasibility of integrating CoTL into an MLOps-driven environment and provides a solid foundation for further experimentation and optimization.

4.4.4 Next Steps

The next phase of this work will focus on transitioning from design to implementation within the 6G-DALI framework. The previously described pipelines will be implemented and integrated into the MLOps environment, ensuring that all components interact as intended. This will be followed by validation activities to assess the correctness and effectiveness of the approach across various experiments and datasets, along with iterative refinements based on observed behavior. In parallel, efforts will focus on improving the robustness and scalability of the solution and ensuring its alignment with the overall 6G-DALI architecture. Overall, these steps aim to evolve the current design into a functional and reliable service for collaborative model improvement in distributed environments.

4.5 Conclusion

This Section focused on the automation of model tuning and optimization across the ML lifecycle within the 6G-DALI framework. The described solutions address key challenges related to HPO, RLOps, and knowledge reuse across distributed AI workflows. The ADO demonstrated the ability to improve HPO efficiency through cross-experiment knowledge transfer and seamless integration into MLOPs pipelines, reducing redundant computations and accelerating model convergence. The AutoRL framework extended these capabilities to RL environments, enabling systematic evaluation of RL agents through advanced HPO techniques. Furthermore, the CoTL approach introduced a novel mechanism for structured knowledge sharing between independently trained models, promoting continuous model improvement while preserving data locality. Together, these developments contribute to more scalable, reproducible, and accessible AI experimentation, supporting the objectives of automated MLOps, RLOps, and knowledge-driven learning in distributed 6G environments.

5 Distributed Training and Workload Execution on 6G Testbeds

5.1 Overview

This section covers the distributed runtime and execution layer. RLOps and FLOps pipelines run on top of 6G testbeds and Digital Twins, and at a meta-orchestration layer can accommodate heterogeneous MLOps and RLOps software stacks.

Two workload placement contributions address the Cloud-Edge continuum: AI/ML workload placement, including RL-based scheduling via ARPA, is described in Section 5.2. Section 5.3 covers Federated Learning Operations, which handle the cases where training data cannot leave its source; the FL work spans NWDAF-enabled FLOps, communication-efficient and privacy-aware FL with robust aggregation, FL for the physical layer and JCAS, and split learning.

Referring to Figure 1, the contributions of this chapter map to the MLOps Tools Stack component within the MLOps block.

5.2 AI/ML Workload Placement

This section presents the initial implementation of Artificial Intelligence (AI)/Machine Learning (ML) workload deployment for the 6G-DALI project across the Cloud-Edge Continuum (CEC). The implementation bridges simulator-based Deep Reinforcement Learning (DRL) agent training with live GPU-enabled infrastructure deployment, enabling fast and reliable policy learning decoupled from the underlying hardware. The design adheres to the 6G-DALI vision for AI/ML inference deployment in testbed environments, while aligning with the AI-RAN Alliance's vision [52] for CEC utilization across consumer workloads and its associated Key Performance Indicators (KPIs) for energy efficiency.

5.2.1 Motivation behind the Implementation

Modern computing infrastructures increasingly support resource-intensive and highly heterogeneous workloads, driven largely by the rapid proliferation of AI services. As demand for such services grows, resource availability becomes a critical challenge.

To address this, the concept of the Cloud–Edge Continuum (CEC) has emerged as a promising alternative. While the CEC already hosts a wide range of network-related functions, a persistent gap remains between theoretical and actual resource utilization, resulting in underused computational capacity. This inefficiency aligns with the vision promoted by the AI-RAN Alliance [52] which advocates for leveraging idle resources, particularly unused GPU cycles, to host consumer-oriented AI workloads.

Within the 6G-DALI project, the ATH site serves as the enabler for AI/ML workload deployment across the Cloud–Edge Continuum. Building on the project's AIaaS experimentation framework, this work contributes to the placement component by proposing an algebraic mathematical framework that emulates a CEC environment for training AI-based placement strategies and a novel placement Agent called ARPA. This framework operates independently of the live system, thereby reducing deployment overhead and mitigating cold-start costs, while ensuring fast and robust training and policy updates.

Furthermore, we introduce a novel inference-time scheduler–orchestrator designed for seamless integration with Kubernetes [53] , termed the AI RAN Placement Agent (ARPA). ARPA is based on Deep Reinforcement Learning (DRL) enhanced with action masking [54] ,[55] , a mechanism that restricts the agent from selecting infeasible actions, such as allocating workloads to fully utilized nodes or nodes exceeding latency thresholds. This approach improves both the efficiency and reliability of placement decisions in dynamic, resource-constrained CEC environments, directly supporting the 6G-DALI vision for scalable and energy-efficient AI/ML inference deployment across heterogeneous testbeds.

5.2.2 The Framework for GPU inference Workload Placement

This section presents the algebraic mathematical framework and DRL-based Placement Agent developed within the scope of 6G-DALI Task 4.1, targeting the design and acceleration of training pipelines for placement agents and smart orchestrators, with the overarching goal of delivering easily updatable, energy-efficient schedulers capable of meeting the demands of next-generation 6G deployments across testbeds.

The section is structured as follows. First, it reviews the relevant State-of-the-Art (SOTA) on intelligent scheduling and DRL-based training in the context of 6G systems, situating the proposed contributions within the broader research landscape. It then introduces the proposed algebraic CEC emulation framework alongside its Kubernetes integration, designed to bridge the gap between theoretical policy training and live infrastructure deployment. Finally, initial experimental results are presented, demonstrating the effectiveness of the framework across three key dimensions: training time efficiency, energy consumption and end-to-end latency according to 6G goals [52] , [56] .

5.2.3 State-of-the-Art

(i) **A closed-loop CEC mathematical framework** with a milliGPU-to-utilization translation layer that bridges Kubernetes scheduling abstractions and hardware-level GPU stress metrics [57] , [58] within a DRL placement context. The framework ensures robust training and seamless policy updates by ingesting up-to-date cluster topology, node capabilities and resource metadata guaranteeing adaptability to any new infrastructure testbed or 6G cluster configuration. Crucially, all training is fully decoupled from the live system, eliminating interference with real infrastructure operations and minimizing cold-start overhead when transitioning to new deployments [59] . By leveraging this framework, the placement agent learns to optimally schedule AI/ML inference workloads without requiring any direct interaction with the physical cluster.

(ii) **ARPA: a Bounded, Reward-Driven Placement Agent** featuring a datasheet-calibrated model and a mandatory action masking mechanism [54] , [55] , designed to support rapid policy retraining without requiring cluster access. The reward function is jointly shaped by energy efficiency and latency objectives, aligned with the AI-RAN Alliance's KPIs for next-generation network performance. The action masking layer enforces placement feasibility by preventing the agent from issuing illegal scheduling decisions that would violate cluster provisioning constraints or compromise workload latency guarantees [56] , while simultaneously accelerating policy convergence by narrowing the effective action space. Policy optimization is carried out via Proximal Policy Optimization (PPO), which enforces conservative, incremental policy updates, promoting stable and consistent performance gains across training iterations.

5.2.4 Solution Description

The proposed solution targets the design and deployment of an intelligent AI/ML workload placement engine for testbeds. The placement agent acts as the orchestration layer between incoming inference workload requests and the underlying infrastructure, scheduling inference workloads across the heterogeneous testbed spanning Far-Edge, Edge and Cloud tiers. Through this integration, testbed gains a smart scheduling component capable of dynamically optimizing resource utilization across its CEC infrastructure without manual intervention.

Figure 24 illustrates the interaction between the 6G-DALI Testbed Adapter, the ISI/ATH CEC Testbed and the 6G-DALI Dataspace Connector. The 6G-DALI Testbed Adapter forwards experiment definitions, including inference workload requests, cluster topology, node capabilities, QoS/latency classes and energy constraints, to the testbed orchestrator. Within the testbed, the RL pipeline serves two purposes. First, RL agents are trained offline using the CEC emulator, which provides a closed-loop simulation environment for learning optimal placement policies. Once trained, these policies are deployed directly to the Kubernetes cluster as the placement orchestrator, the deployment leverages the Triton Inference Server, which hosts multiple AI inference workloads and enables real-time scheduling decisions based on the learned RL policies.

The testbed is also integrated with the 6G-DALI Dataspace Connector, which exports trained RL policies, scheduler logs, placement statistics and energy and latency KPIs. These artifacts are published to the Dataspace Connector, providing measurable evidence of the performance of the CEC orchestration framework and enabling cross-testbed evaluation, reproducibility and performance analysis.

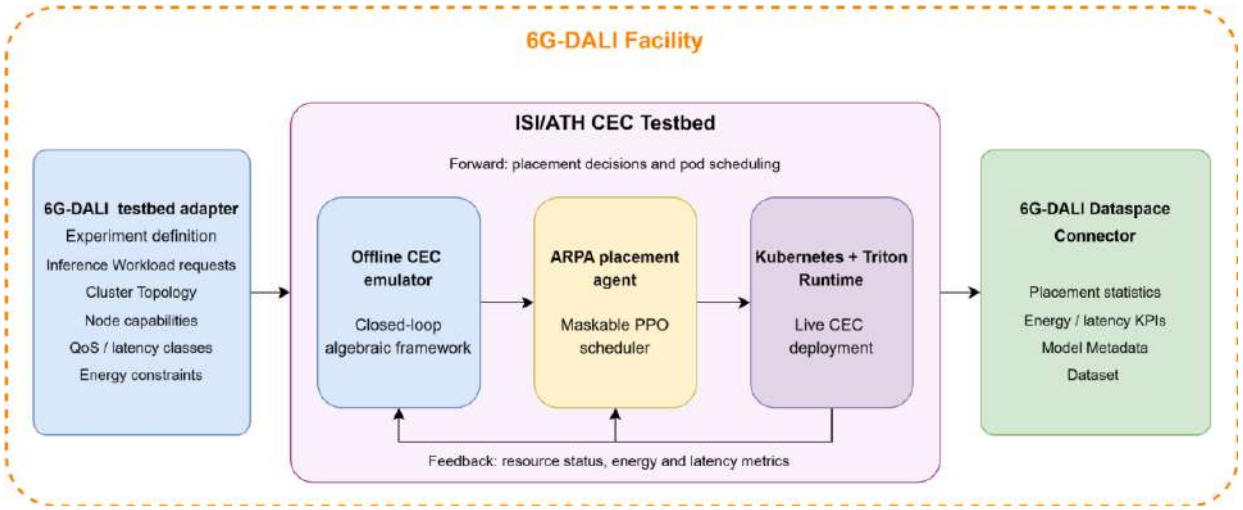


Figure 24: System Overview

The proposed solution consists of two tightly coupled components: a closed-loop algebraic mathematical framework that emulates CEC behavior for offline DRL training [59] and ARPA, the resulting placement agent is going to be deployed at inference time via Kubernetes Integration.

System Scaling. Before a placement decision is made, each incoming job j is translated from raw Kubernetes request parameters [53] into a normalized workload uj ∈ R^4:

$$u_j = \begin{pmatrix} \hat{c} \\ \hat{s} \\ \hat{d} \\ \hat{m} \end{pmatrix} = \begin{pmatrix} \frac{c_{milli}}{1000} \\ \left\lceil \frac{m_{gpu}}{1000} \cdot \frac{V_{src}}{V_{slice}} \right\rceil \\ \frac{u}{100} \cdot \frac{m_{gpu}^2 \cdot V_{src}}{m_{ref}^2 \cdot V_{node}} \\ \frac{mem_{mib}}{1024} \end{pmatrix}$$

where $\hat{c}$ is the normalized CPU demand in cores, $\hat{s}$ is the required number of V_{slice}-GiB VRAM slices [57] , $\hat{d}$ is the GPU stress score and $\hat{m}$ is the memory demand in GiB. This milliGPU-to-utilization translation bridges Kubernetes scheduling abstractions with hardware-level GPU stress metrics.

Closed-Loop State Transition. The CEC environment is modelled as a Markov Decision Process $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$. The per-node state $X^{(k)} \in R^4$ carries current CPU utilization, GPU slice occupancy, GPU stress load and memory occupancy. The state transition at each scheduling step is defined as:

$$X_{t+1}^{(k)} = X_t^{(k)} + \underbrace{u_j{}^T \cdot W_k}_{\text{Weighted load}} + \underbrace{\begin{pmatrix} \alpha_k \\ 0 \\ \beta_k \left(S_t^{(k)}\right)^2 \\ 0 \end{pmatrix}}_{\text{Overhead } h^{(k)}}$$

Where $W^{(k)} \in R^{4\times4}$ is a sparse diagonal weight matrix scaling each resource component of u_j to the capacity units of node k, α_k is the per-node static CPU overhead and $\beta_k \left(S_t^{(k)}\right)^2$ is the quadratic GPU stress overhead [60] . The framework ingests up-to-date cluster topology and resource metadata, requiring no structural modifications when nodes join or leave the cluster.

Multi-Objective Reward Function. ARPA's reward function jointly optimizes tier selection, energy efficiency and latency, aligned with the AI-RAN Alliance KPIs. For a placement action a ∈ $\{0, \dots, |\mathcal{N}| - 1\}$:

$$R_{pl} = w_q\left(T_a + \alpha E_a - \beta \tilde{L_j} - G_j + A_j\right)$$

where T_a is the tier bias, $E_a = 1 - P_{inst}^{(a)}/P_{peak}$ couples the power model [60] to the reward, $\widetilde{L_j} = \min(1, L_j/L_{max})$ is the bounded latency-age penalty, G_j penalizes GPU job placement on CPU-only nodes and A_j rewards correct GPU-affinity routing. The QoS weight $w_q \in \{1.5,1.0,0.65\}$ scales the reward across Latency-Sensitive (LS), Best-Effort (BE) and Burstable job classes respectively, derived from Alibaba GPU Cluster Trace 2023 metadata [61] , implicitly guiding the policy to prioritize edge-tier placement for LS workloads. A wait penalty applies only when the cluster is genuinely full:

$$R_{wt} = -w_q \cdot 1.0$$

Mandatory-Placement Action Masking. A key contribution of ARPA is its corrected mandatory-placement action mask [54] , [55] . The mask acts as an admission controller, enforcing hard capacity constraints such that the policy can only select feasible placement nodes. The wait action is only permitted when no placement node satisfies the capacity constraints:

$$M_{wait} = \neg(\exists k \in \{0, \dots, |N| - 1\}: M_k)$$

The per-node feasibility condition M_k is:

$$M_k = \begin{cases} \text{False} & \text{if } (\text{is_multi} = \text{T}) \wedge (k \neq k_{\text{cloud}}), \\ \text{False} & \text{if } (\hat{s} > 0) \wedge (C_{\text{slc}}^{(k)} = 0) \wedge (k \notin \mathcal{I}_{\text{FE}}), \\ \text{False} & \text{if } (C_t^{(k)} + \hat{c}_w > C_{\max}^{(k)}) \vee \\ & \quad (S_t^{(k)} + \hat{s} > S_{\max}^{(k)}) \vee \\ & \quad (M_t^{(k)} + \hat{m}_w > M_{\max}^{(k)}), \\ \text{True} & \text{otherwise.} \end{cases}$$

Without this constraint, the policy can spuriously skip jobs on lower-tier nodes whose placement reward is smaller than the wait penalty, inflating the drop rate and distorting the learning signal. The mask simultaneously narrows the effective action space, accelerating policy convergence.

5.2.5 Live System Integration

This section describes the architectural transition from simulation to a live Kubernetes deployment, alongside preliminary implementation results.

The proposed production system intergrates with NVIDIA Triton Inference Server [62] , which serves as the unified inference backend across all cluster tiers. Multiple Triton instances are deployed to represent distinct workload classes, spanning both Network Functions and Consumer Workloads including Large Language Models, vision models and additional AI/ML inference services. An external traffic generator is employed to simulate real-world user demand and drive inference request load across the system.

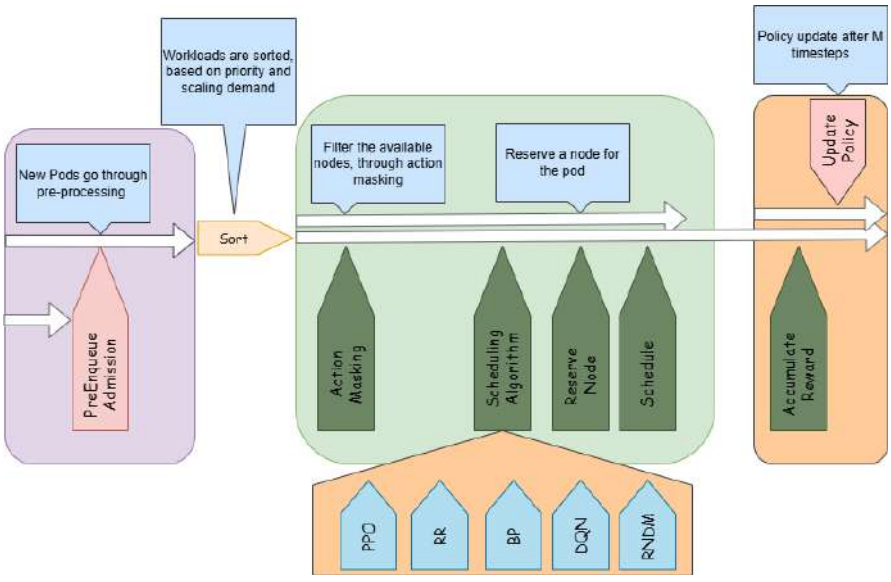


Figure 25: Scheduler Flow

As illustrated in Figure 25: Scheduler Flow the native Kubernetes scheduler is replaced by ARPA, which integrates into the standard Kubernetes scheduling pipeline. The core operational logic is as follows: ARPA manages the lifecycle of Triton pods, beginning with a single container and triggering horizontal scaling when incoming traffic exceeds the current serving capacity of a tier. Each Triton pod operates independently, maintaining its own model repository and request handling pipeline capable of serving multiple inference models concurrently.

5.2.6 Initial Results

Initial experiments evaluate PPO against four baseline scheduling algorithms: Deep Q-Network (DQN), Round-Robin (RR), Bin-Packing (BP) and Random scheduling. All experiments run on the algebraic mathematical framework described in Section 5.2.4 and all algorithms are augmented with action masking. Workload traces are drawn from the Alibaba GPU Cluster Trace 2023 [61] . The eight-node topology instantiates $n_{\mathrm{FE}} = 5$, $n_{\mathrm{E}} = 2$ and $n_{\mathrm{C}} = 1$.

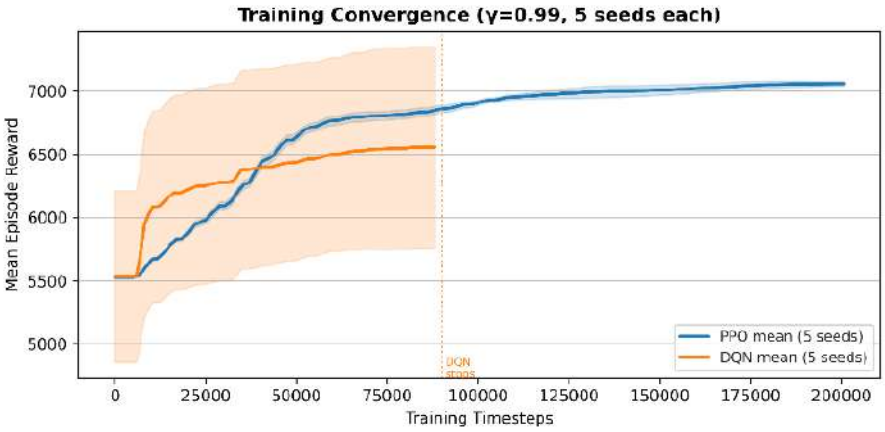


Figure 26: Training Convergence

Training Convergence. In Figure 26 ARPA (Maskable PPO) converges within approximately 80,000 steps, reaching a higher mean episode reward than DQN and completes full training in 184 seconds on a single workstation. DQN trains in 61 seconds but plateaus at a substantially lower reward ceiling.

AI-RAN KPI Evaluation. Table 15 summarizes the AI-RAN Alliance KPI results [52] on the held-out test partition.

Table 15. Algorithm Evaluation based on the AI-RAN Alliance KPIs

Algorithm	Offloading (%)	Drop (%)	Energy (x10^6 kJ)	Latency(ms)
ARPA(PPO)	**98.81**	**1.19**	**615.26**	**885.9**
DQN†	98.81	1.19	883.75	1069.0
Round-Robin	98.81	1.19	819.51	1061.7
Bin-Packing	98.81	1.19	1368.91	1628.3
Random	98.81	1.19	944.28	1386.1

Offloading. All algorithms achieve 98.81% job offloading under the mandatory-placement mask. Figure 27 further illustrates how each algorithm distributes placements across nodes. PPO concentrates the majority of jobs in the Edge tier while systematically avoiding the CPU-only Far-Edge node, reflecting the agent's learned preference for GPU-capable, energy-efficient targets. Bin-Packing consolidates workloads predominantly in the Cloud tier, consistent with its capacity-maximization objective. Round-Robin distributes placements uniformly across all available nodes regardless of resource heterogeneity, producing balanced but topology-unaware decisions.

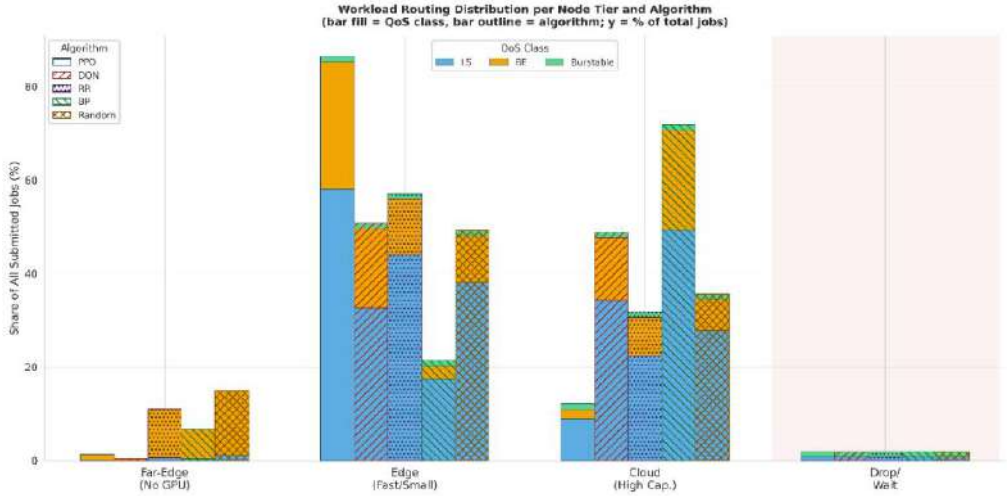


Figure 27: Workload Routing per Node Tier for each algorithm

Energy Efficiency.

Power and Energy Model. Instantaneous node power follows a linear static-dynamic split [60] :

$$P_{inst}^{(k)} = P_{idle}^{(k)} + P_{dyn}^{(k)} \cdot u_{gpu}^{(k)}$$

Where $u_{gpu}^{(k)} = \hat{d}_j / D_{max}^{(k)} \in$ GPU enabled nodes, $P_{idle}^{(k)}$ represents the baseline power draw of node k, when the workload is idle, and $P_{dyn}^{(k)}$ the actual power draw when a workload is executed. Energy per job is computed as:

$$E_j^{SI} = \frac{P_{inst}^{(k)} \cdot \tau_j^{eff}}{1000}$$

where τ_j^{eff} is the effective execution duration of job j on node k.

Among all methods at 98.81% offloading, PPO attains the lowest mean total energy at 615.26 x10^6kJ, 25% below Round-Robin (819.51 x 10^6 kJ) and 55% below Bin-Packing (1368.91 x 10^6 kJ). This advantage is driven by PPO's tier routing behavior: 87.5% of jobs are placed on the energy-efficient edge tier, whereas Bin-Packing concentrates approximately 72% of jobs on the Cloud node, incurring 2.2 × PPO's total energy. In Figure 28 the energy consumption per node for each evaluated algorithm is presented, highlighting the differences in resource efficiency across the compared scheduling approaches.

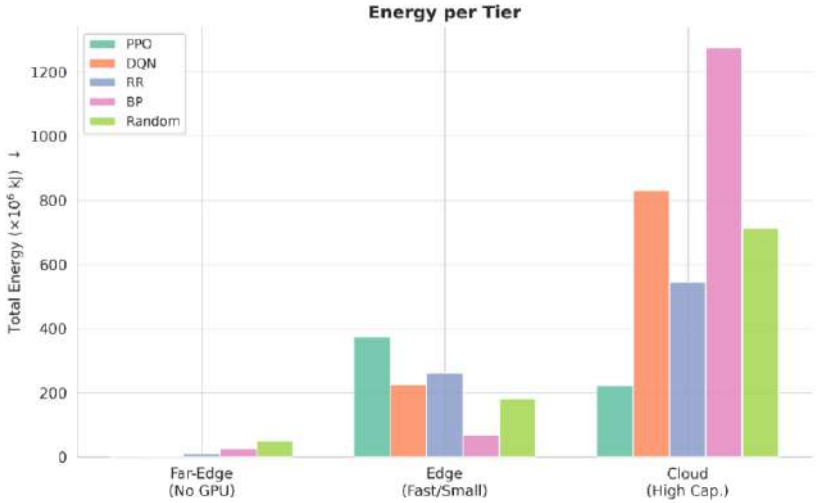


Figure 28: Energy Consumption per CEC Tier

Latency.

End-to-End Latency Model. The per-job latency assigned to node k is:

$$L_j^{(k)} = w_j + d_j^{eff} + \delta^{(k)}$$

Where $w_j = \max\left(0, t_j^{\text{sched}} - t_j^{\text{arr}}\right)$ is the queuing delay and $d_j^{\text{eff}} = \lambda_{\text{FE}} d_j$ on a CPU-only far-edge node, otherwise $d_j^{\text{eff}} = d_j$. Network propagation delays $\delta^{(k)} \in \{\delta_{\text{FE}}, \delta_{\text{E}}, \delta_{\text{C}}\}$ are aligned with 3GPP TS 22.261 targets [56] .

ARPA achieves a mean LS latency of 885.9 ms, 17% below Round-Robin and 46% below Bin-Packing. Bin-Packing's cloud concentration directly explains its latency penalty, since the Cloud tier carries a propagation delay of $\delta^{\text{C}} = 100$ ms versus $\delta^{\text{E}} = 15$ for the edge tier, consistent with 3GPP TS 22.261 targets. ARPA's far-edge routing is negligible at 1.4% of jobs, avoiding the CPU-only penalty while maintaining full offloading.

Overall, the initial results demonstrate that the proposed algebraic framework enables fast, infrastructure-independent DRL training and that PPO consistently outperforms all mask-compliant baselines across energy efficiency and latency KPIs [52] , validating both the framework and the placement agent as promising components for 6G-DALI CEC orchestration.

5.2.7 Next Steps

The primary next step is the validation of the mathematical framework within the live Kubernetes environment, where real execution traces will allow the power and latency models to be calibrated against actual hardware measurements. In parallel, more advanced placement architectures will be explored, targeting improved planning, sample efficiency and generalization across unseen infrastructure configurations.

5.3 Federated Learning Operations (FLOps)

This section presents the initial implementation of Federated Learning Operations (FLOps) for the 6G-DALI project, across NWDAF-enabled FLOps, FL applied to PHY and JCAS, communication-efficient and privacy-aware FL with a robust aggregation strategy, and split learning framework to support efficient distributed model training. The implementation bridges 6G-DALI's MLOps pillar and NWDAF-enabled FLOps at the testbed side, algorithmic innovation on 6G-DALI wireless networking datasets, and split-learning framework. These designs adhere to the 6G-DALI vision across MLOps and NWDAF-FL, split learning and align with core performance indices for 6G privacy and efficiency.

5.3.1 NWDAF and FLOps

This section introduces the new OpenAirInterface (OAI) NWDAF implementation, recently open sourced by EURECOM. The implementation extends the baseline OAI NWDAF toward full compliance with 3GPP Release 17 by integrating MLOps capabilities for ML model training, provisioning, and lifecycle management. Building on this foundation, the section then presents the proposed Federated Learning enabled NWDAF (FLOps) architecture, which supports collaborative distributed model training across multiple NWDAF Model Training Logical Functions (MTLFs). This architecture advances NWDAF capabilities toward the federated intelligence framework envisioned in 3GPP Release 18.

5.3.1.1 OAI NWDAF Release 17 Implementation

The 3GPP Network Data Analytics Function (NWDAF) was introduced as a native Network Function (NF) within the 5G Core Service Based Architecture (SBA) to provide standardized analytics services to consumer NFs. Its primary role is to collect operational telemetry from core network functions and infrastructure components, process these data streams using statistical and machine learning based methods, and expose predictive or descriptive analytics through standardized service interfaces. In this context, NWDAF enables closed loop automation in the core network by supporting analytics driven optimization, anomaly detection, and service assurance [63] .

The previous OAI NWDAF implementation provided a solid baseline aligned with 3GPP Release 15, including standardized analytics exposure and data collection capabilities. However, its intelligence pipeline remained limited by a conventional static ML design. Specifically, the previous implementation

supported only pre trained models deployed offline, without native support for online model retraining, automated lifecycle orchestration, or dynamic model provisioning. As a result, the platform lacked key MLOps capabilities, including model versioning, metadata management, artifact storage, retraining triggers, and continuous deployment of updated models. These limitations constrain NWDAF adaptability in highly dynamic 5G and beyond network environments where traffic patterns, radio conditions, and service requirements continuously evolve.

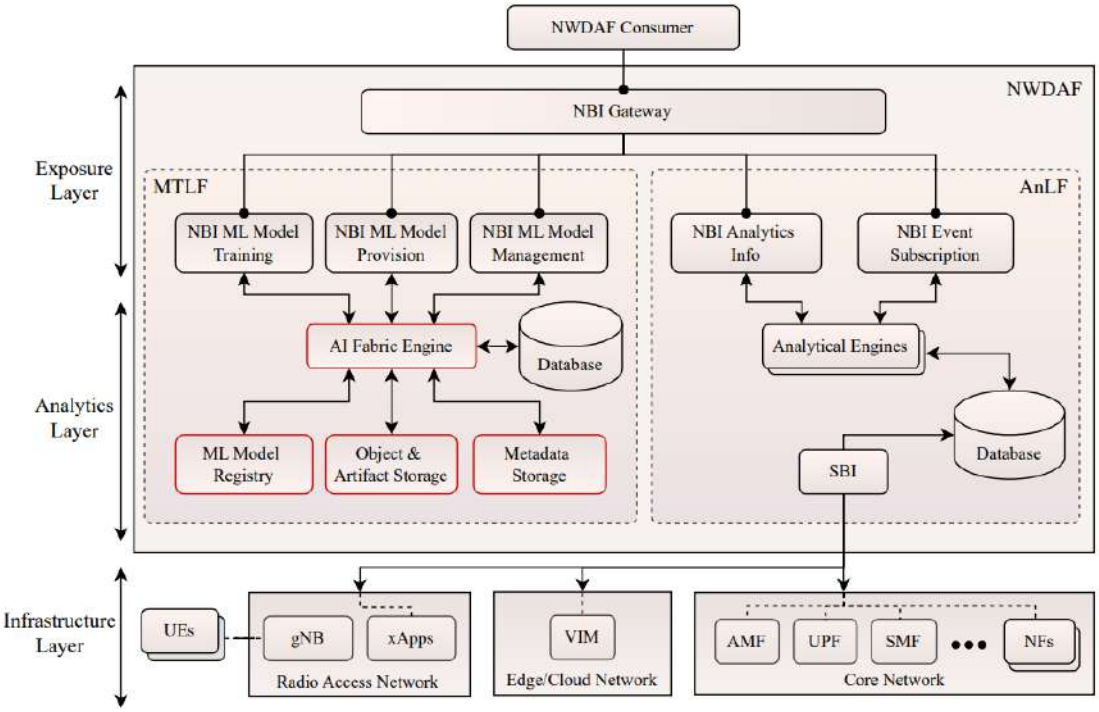


Figure 29: The proposed service-based NWDAF Release 17 architecture [63] .

To address the preveous limitations, EUR extends the OAI NWDAF implementation toward a Release 17 compliant MLOps enabled architecture, illustrated in Figure 29. The proposed design introduces a modular three-layer architecture composed of an Exposure Layer, an Analytics Layer, and an Infrastructure Layer, enabling end to end integration of telemetry acquisition, analytics generation, model training, and standards compliant service exposure. At the Exposure Layer, a unified Northbound Interface (NBI) Gateway provides standardized interfaces for analytics requests, event subscriptions, and ML lifecycle services. Beyond conventional Analytics Logical Function (AnLF) services, the architecture introduces a dedicated Model Training Logical Function (MTLF) exposing standardized interfaces for ML model training, provisioning, and lifecycle management, thereby extending NWDAF from an inference only analytics engine into a full ML operational platform [63] [64] .

Within the Analytics Layer, analytical engines interact with the MTLF whenever predictive analytics are required. The MTLF integrates an internal AI Fabric Engine, Model Registry, Object and Artifact Storage, and Metadata Storage, collectively providing full MLOps functionality, including dataset ingestion, automated training workflows, artifact persistence, model cataloging, version control, and deployment orchestration. When an analytics consumer requests predictive inference, the NWDAF dynamically determines whether a suitable model already exists; otherwise, it triggers an automated training pipeline, stores the resulting model artifacts, and provisions inference endpoints once the model becomes operational. This architecture therefore enables both continuous inference and continuous training under a unified standards aligned framework.

At the Infrastructure Layer, NWDAF aggregates heterogeneous telemetry from multiple domains, including 5G Core NFs, virtualized cloud or edge infrastructure, and RAN level observability sources, including O RAN xApps. Through standardized southbound interfaces, these data streams populate the analytics and training pipelines, enabling cross domain intelligence with full network visibility. Overall, the EUR implementation transforms OAI NWDAF from a static analytics consumer into a scalable MLOps

enabled intelligence platform aligned with 3GPP Release 17, providing the technical foundation for distributed and collaborative intelligence mechanisms such as federated learning [64] [63] .

5.3.1.2 NWDAF-enabled FLOps Architecture

Building on the Release 17 MLOps enabled NWDAF implementation described above, EUR further extends the architecture toward a Federated Learning Operations (FLOps) framework aligned with 3GPP Release 18, which introduces support for distributed collaborative model training across multiple MTLF instances. Rather than relying on centralized dataset aggregation, Release 18 enables NWDAF instances to participate in standardized federated learning workflows, where model training remains local to each administrative or operational domain while only model parameters or gradients are exchanged for global aggregation [63] [63] .

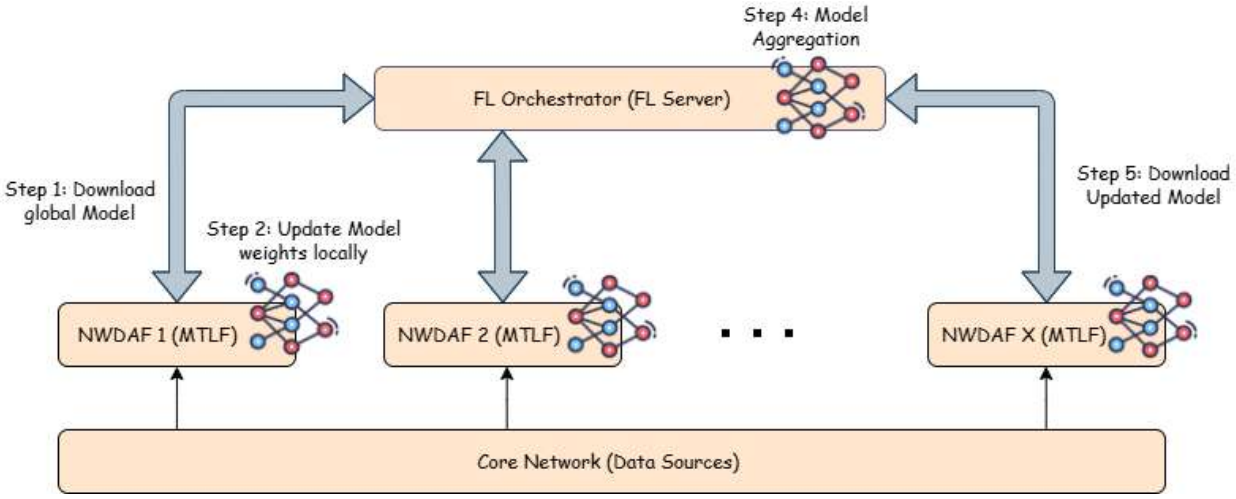


Figure 30: The proposed NWDAF-enabled FLOps Architecture

As illustrated in Figure 30, the proposed NWDAF enabled FLOps architecture introduces a logical Federated Learning Orchestrator (FL Server) that coordinates training among multiple distributed NWDAF MTLFs. Each participating MTLF acts as an autonomous federated client with local access to domain specific telemetry, local compute resources, and independent model training pipelines. The federated process follows a standard iterative workflow: (1) the orchestrator distributes an initial global model to participating MTLFs, (2) each MTLF performs local training using its own locally collected telemetry and feature pipelines, (3) updated local model parameters are securely transmitted to the orchestrator, (4) the orchestrator performs global aggregation using federated optimization methods such as Federated Averaging (FedAvg), and (5) the updated global model is redistributed for the next training round. This iterative closed loop enables collaborative model convergence without exposing raw operational data outside each local domain.

A key architectural advantage is that federated learning is naturally integrated into the NWDAF MTLF lifecycle framework. Each MTLF already provides local dataset management, artifact storage, metadata tracking, model provisioning, and retraining capabilities. FLOps therefore extends existing MLOps workflows into distributed MLOps, where model lifecycle management becomes collaborative while remaining standards compliant with 3GPP service exposure and orchestration principles.

It is important to emphasize that multiple MTLFs operating within the same 5G Core deployment do not necessarily imply identical data sources. In practice, participating MTLFs may train on highly heterogeneous datasets reflecting different operational contexts. Data separation may arise from network slicing, where distinct slices (for example eMBB, URLLC, and mMTC) generate fundamentally different traffic dynamics, QoS characteristics, and KPIs. It may also arise from geographical distribution, where edge domains or regional deployments observe different mobility patterns, radio propagation conditions, congestion behaviors, and user service profiles. Additional heterogeneity can stem from vendor specific infrastructure stacks, administrative isolation domains, or service specific observability pipelines. Consequently, federated learning across MTLFs enables NWDAF to build globally robust models that capture broad network intelligence while preserving locality, privacy, operational

autonomy, and data sovereignty. This capability represents a key enabler for intelligent, scalable, and standards aligned AI native 6G core networks[64] .

5.3.2 FL Algorithms for PHY and JCAS

This section reports on the investigation of FL algorithms for JCAS, utilizing PHY data. The work focuses on CSI-based indoor localization in distributed antenna array deployments, where each AP observes locally collected data under heterogeneous propagation conditions. The objective is to assess whether FL can improve generalization across APs while respecting realistic system constraints, and to identify architectural and data-related requirements for viable FL deployment in PHY-level applications.

5.3.2.1 State-of-the-art

FL has recently transitioned from predominantly upper-layer machine learning applications to the wireless PHY and, more recently, to JCAS architectures. This evolution is driven by the need to exploit distributed data sources while preserving privacy, reducing backhaul load, and enabling scalable learning across large numbers of wireless devices. In PHY applications, FL has been investigated for symbol detection, channel estimation, beamforming, and resource allocation, where edge nodes train local deep learning models using pilot symbols or raw waveform samples and exchange model parameters for aggregation rather than raw data [65] [66] [67] .

Survey and tutorial contributions demonstrate that FL at the PHY can achieve detection and estimation performance comparable to centralized learning while significantly reducing communication overhead, albeit often at the cost of a moderate loss in accuracy due to the lack of globally pooled data [65] [66] . These observations have motivated extensive work on optimizing the interaction between communication and learning, including joint optimization of resource allocation, user selection, and transmission power to balance model convergence, latency, and energy efficiency [68] [69] [70] . Over-the-air aggregation techniques exploiting waveform superposition have further been proposed to accelerate FL convergence by enabling analog parameter aggregation directly in the wireless channel, particularly in dense deployments and massive MIMO scenarios [70] [71] .

Within the broader context of JCAS and emerging 6G systems, FL has been incorporated into ISAC frameworks where sensing, communication, and learning are co-designed. Recent works show that FL can be embedded into ISAC architectures to jointly optimize beamforming, sensing accuracy, and communication performance while avoiding the exchange of sensitive sensing data [72] [73] [74] . This includes FL enabled ISAC systems operating in mmWave and RIS assisted environments [74] [75] , as well as frameworks where federated or vertically federated learning supports distributed sensing, motion recognition, and environment awareness [76] . These approaches frequently rely on joint optimization across multiple domains and increasingly adopt reinforcement learning [74] or personalized FL [77] mechanisms to cope with heterogeneous objectives and observations across base stations or sensing nodes.

Despite these advances, the state-of-the-art consistently highlights that FL performance at the PHY is heavily constrained by strong data heterogeneity resulting from location-dependent propagation effects [78] This challenge is particularly pronounced in sensing and localization tasks, where each receiver observes only a partial and geometry-dependent view of the environment. As a result, recent literature increasingly emphasizes personalized FL as a necessary evolution beyond standard FedAvg, allowing shared representations to be learned collaboratively while preserving client-specific model components tailored to local sensing or channel characteristics [78] .

5.3.2.2 Solution description

In the PHY-specific component of the FLOps, tailored FL algorithms exploiting the characteristics of CSI will be implemented. They will be tested for accuracy on the use-case of massive MIMO-based indoor localization and compared against state-of-the-art FL algorithms. An initial setup will be a split of the localization DNN into a local ANN, a shared ANN, and a second local ANN. The concept is that the first local ANN is trained on solely local data, learning location-specific antenna calibration and initial denoising. The feature vectors outputted by this local ANN is then fed into the shared ANN. The shared

ANN is trained in a federated manner, extracting global attributes specific to the signal, but not to the antenna position. Finally, the second local ANN receives as input the features generated by that shared ANN and is again trained on solely its local data. It serves as a readout for the task to be learned and contains attributes specific to the location of the antennas. With this split of the DNN, the setup falls under the personalized FL paradigm.

5.3.2.3 Initial results

In the initial formulation, each antenna array acts as an FL client and trains a local DNN on locally collected CSI to estimate the AoA, which would then be subsequently used to estimate the user's position from all AoAs centrally. Model parameters are periodically transmitted to a central server, aggregated using FedAvg, and redistributed to the clients. Throughout training, raw CSI data remain local to each AP. First experiments were conducted using the nomadic dataset of a publicly available dataset [79] . Precisely, the sub-dataset which contains CSI collected at a distributed array with 8x8 antennas over 100 subcarriers, each from four different user positions and in 7 different scenarios, is used. One scenario is static, meaning no changes in the environment occur at the time of data collection. The other six scenarios are changing and include a human walking a specific trajectory up and down during data collection. This sub-dataset is further divided into trajectory-based train and test splits to evaluate generalization to unseen motion patterns. A depiction of the setting is shown in Figure 31.

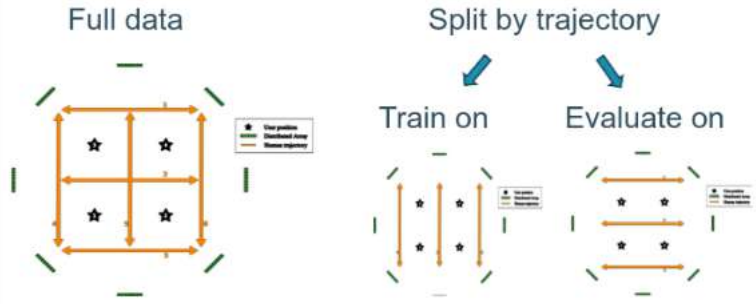


Figure 31: On the left, the setting of the nomadic dataset. The black stars show the user positions, in green the distributed array, and in orange six trajectories of a human walking, each defining a new scenario in which data is collected. For the static scenario, no human is moving. On the right, the trajectory-based train and test splits for subsequent model training.

Three training paradigms were compared: purely local training, centralized training on pooled data, and federated training using FedAvg. For all three, an established DNN from the literature was used [80] .

Table 16 summarizes representative evaluation results for these paradigms.

Table 16. Performance comparison using trajectory-based data splits.

Training approach	**Test loss (training)**	**Test loss (evaluation)**
Centralized training	0.22°	26.1°
Local training	0.14°	34.6°
Federated Learning (FedAvg)	14.5°	24.1°

Local and centralized training achieve very low training loss but exhibit large errors on unseen trajectories, indicating severe overfitting. In contrast, FL shows significantly higher training loss but improved evaluation performance, suggesting stronger robustness to distribution shifts.

Additional baseline experiments further illustrate the sensitivity of FL to training configuration and data heterogeneity. In these experiments, train and test data are equally split among all trajectories, leaving no unseen motion patterns. The results are summarized in Table 17.

Table 17. Performance comparison under simplified evaluation conditions.

Training approach	**Test error**
Centralized training	0.48°
Local training	0.25°

Federated Learning (FedAvg)	7.2°

While these results are not directly comparable due to differences in training configuration and normalization schemes, they highlight the need for careful FL-specific optimization and architectural design.

A major limiting factor identified during this work is the restricted diversity of the currently available dataset. The dataset contains CSI corresponding to only four distinct user positions. These positions result in different AoAs at different APs, which leads to complete absence of overlapping angular observations across some antenna arrays. As a result, some APs do not observe certain angles at all, creating extreme heterogeneity in local learning problems and severely limiting the effectiveness of FL. Under such conditions, global aggregation provides limited benefit because not all clients can contribute meaningful gradients for the same regions of the state space.

To address this limitation, a new dataset with a substantially larger number of user positions is currently being collected. The increased spatial diversity is expected to create stronger overlap in observable angles across APs, reduce extreme heterogeneity, and enable a more realistic assessment of personalized FL architectures. This dataset will be used in subsequent research to further evaluate FL algorithms for PHY-level localization and to strengthen contributions to FL for PHY and JCAS within the project.

5.3.3 Federated Learning Algorithms

This section first highlights the challenges of applying FL algorithms directly to 6G scenarios and then reviews the state-of-the-art works on communication overhead reduction and privacy-preserving efficiency while guaranteeing the resilience of FL to defend against Byzantine attacks. It then introduces the proposed solution, which follows a standard client-server architecture with the proposed communication-efficient, Byzantine-robust, and privacy-aware scheme.

In wireless communication settings, massive multiple-input and multiple-output (MaMIMO) systems have received increasing attention for their high angular resolution. Applying FL algorithms to such 6G data has received considerable attention due to the distributed nature of MaMIMO data, privacy protection requirements, and the emergence of edge intelligence in wireless networks. Consider a practical scenario of channel state information (CSI) prediction task, for instance, user devices or access points may observe local CSI samples under different user trajectories, locations, and propagation conditions. Centralizing these raw CSI data can be costly on storage and privacy-sensitive, while FL enables collaborative model training by exchanging model updates to protect privacy-sensitive raw data. However, directly applying FL to such CSI-driven prediction applications faces three main challenges. First, the CSI prediction model could contain many parameters, and frequent upload of full model updates creates substantial uplink communication overhead. This is especially problematic for a great number of wireless clients with limited bandwidth budgets at the edge. Second, the FL system may involve unreliable or compromised client updates, due to device malfunction, unstable wireless links, mis-calibrated sensors, or adversary behaviors. Uploading such updates directly to the federated model aggregation procedure could degrade the model performance. Third, although traditional FL avoids raw data sharing, full model updates could still leak information about raw data, such as local CSI distribution, user locations, or channel patterns. Hence, privacy protection is still required beyond keeping raw data on-site.

5.3.3.1 State-of-the-Art

State-of-the-art studies on communication-efficient, privacy-aware, and Byzantine-robust FL have attracted significant attention in recent years.

Early work such as FLAME[81] focus mainly on defending against backdoor attacks, which indicates that backdoor robustness in FL can be improved without requiring access to clients' private training data. FLAME uses pairwise cosine distances to measure the angular differences between all model updates and applies the HDBSCAN clustering algorithm. It adds isotropic Gaussian noise of scale to the equal-weight aggregate, which makes the aggregation more robust to outliers or backdoor updates and supports differential privacy. However, because FLAME exposes full client updates, it has good model utility but no compression of the local model updates. The Gaussian noise in FLAME is added after

aggregation, so it can protect release global update, but it does not protect individual client updates from the aggregation server.

At the same time, works on Byzantine-robust optimization with variance reduction and compression [82] provide strong theoretical foundations for reducing communication costs while maintaining robustness against malicious clients. For instance, Gorbunov et al. [83] present that variance reduction can substantially improve Byzantine-robust distributed learning. Their work shows that combining variance reduction with robust aggregation and communication compression leads to better convergence rates, weaker assumptions, and more communication-efficient Byzantine-robust federated learning. To be specific, their work achieves substantial communication reduction. Rammal et al. [82] [82] advance communication compression for Byzantine-robust learning by proposing more efficient algorithms with improved theoretical rates. However, such algorithms expose compressed communication updates without differential privacy or secure aggregation, which can create substantial privacy leakage. SparseFed [84] uses global top-k update sparsification and device-level gradient clipping to mitigate model poisoning attacks. Such sparse model updates can reduce the influence of malicious clients, improve robustness, and lower communication costs.

More recent studies further combine these ideas with differential privacy [85] and unified designs for privacy, robustness, and efficiency. Xia et al.[86] present a unified framework for privacy, robustness, and communication efficiency in federated learning. They suggest that these objectives do not always need to be treated as strict trade-offs; instead, they can be jointly optimized according to carefully designed mechanisms. Moreover, Xu et al. [87] investigate sparsification as a tool and propose Clover, by following standard top-k gradient sparsification to improve communication efficiency while hiding individual top-k indices via three-server secret sharing. Although Clover presents a strong privacy and efficiency construction, it does not solve Byzantine-gradient attacks.

Collectively, these works point toward FL solutions that jointly achieve Byzantine robustness, privacy protection, and communication efficiency.

5.3.3.2 Solution description

Given a client-server-based FLOps architecture, the goal of the proposed FL algorithm is not only Byzantine robustness, but also the joint trade-off among model utility, communication reduction and privacy awareness. To achieve this, the core design idea of the proposed algorithm include 1) send only the most important Top K coordinates rather than the full client model updates, 2) add optional noise for privacy, such as differential privacy at the client side and server side, 3) use error feedback mechanism so dropped coordinates are not permanently lost to preserve the utility (with top k sparsification), and 4) filter suspicious clients before aggregation, in the light of the communication-efficient, Byzantine-robust, and privacy-aware (CBP) scheme.

As shown in Figure 32, the proposed algorithm consists of the local model training with privacy protection and communication efficiency design at the client side, and federated model aggregation with robust filtering at the server side.

Specifically, at each communication round t, the server broadcasts the global model G_t^* to distributed clients. At the client side, each client i 1) computes local update, 2) adds or updates error-feedback residual, 3) clips local update, 4) selects sparse indices with deterministic top K, or noisy top K using the parameter index_sigma (if index_sigma > 0 or add cover indices if setting index_cover_k > 0) and updates residual using the true selected values, 5) adds client_sigma Gaussian noise to selected values, and 6) send perturbed sparse indices $\hat{s}_i$ and noisy values $\widehat{W}_i^k$ to the server. The server uses robust aggregation strategy within the CBP scheme to filter out suspicious or unreliable clients before aggregation to defend against byzantine attacks.

In theory, the proposed FL strategy is mostly model agnostic at the aggregation level because it operates on flattened client update vectors instead of model architecture internals. Hence, it is supposed to be applied across different datasets and training tasks. Nevertheless, the model performance will still be determined by hyperparameter settings across different tasks, as the training process decides the loss function and metrics.

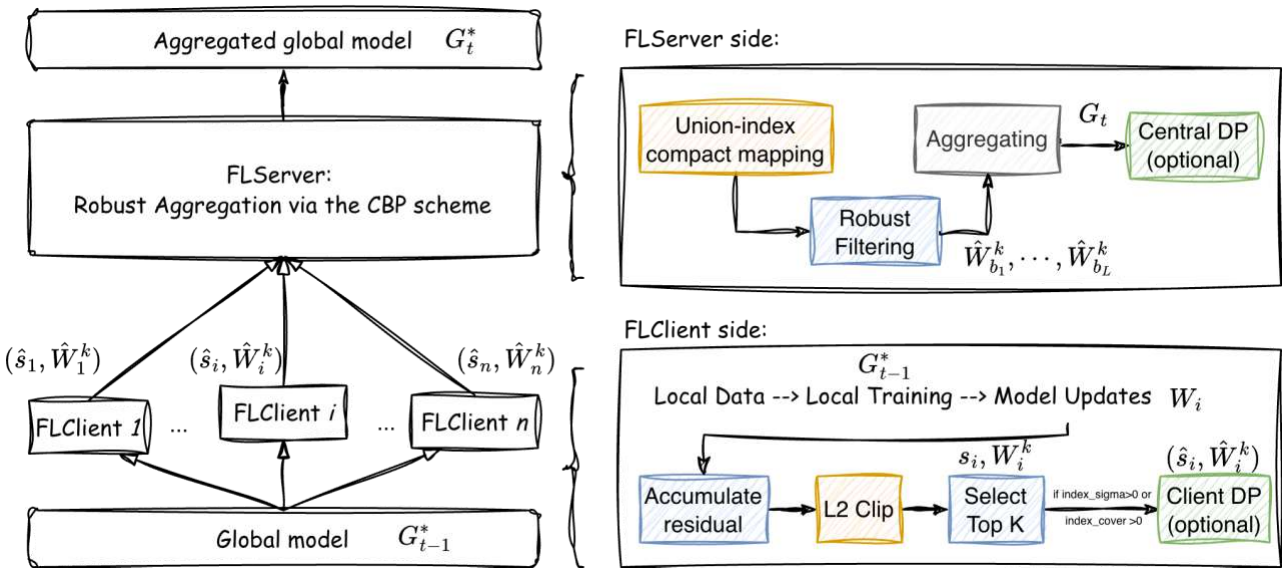


Figure 32: The proposed federated learning algorithm, proposedTopK.

5.3.3.3 Experiments and Initial Results

Currently, the algorithm development is still actively in progress. During the initial stage, the main efforts were made to establish common baselines for comparisons.

Initial experiments examine the performance of the proposed FL algorithm against re-implemented baselines: FLAME, EF21-MAD, ByzDASHAPAGE, SparseFed, FedDPRoC and Clover under the client-server system model. EF21-MAD is a combination of EF21 tracker [88] and normalized MAD filter for aggregation strategy baseline under the client-server architecture. Note that to keep each method is a single uniform interface and a common hyperparameter setting, the baselines are simplified implementations that recreate the core mechanism of each original work while omitting components that do not change the numerical aggregate under the proposed system model. All experiment runs are seeded with three random seeds 7, 42, and 101.

The proposed algorithm is evaluated according to the measurements of Byzantine robustness, model performance, communication reduction, and privacy protection. To evaluate the Byzantine robustness, the experiments consider model poisoning attacks, for example, a random attack, in which the attacker replaces the honest update with random Gaussian noise scaled by the norm of the original update. Byzantine kept is defined as a measurement of the fraction of Byzantine clients that survive the aggregator's filtering, which can be calculated by byzantine_kept_rate = mean(*fn_r*/*by_n*), where *fn_r* denotes the false-negative rate of Byzantine detection and *by_n* is the number of Byzantine clients. Hence, the lower Byzantine kept rate, the Byzantine robustness is better. The communication reduction is defined by the upload saving compared with sending full gradient update, which is used to evaluate the communication efficiency. In terms of the privacy protection, privacy leakage is introduced via mutual information (MI) to measure the leakage of private-sensitive information, where leakage ratio per round indicates the normalized estimate of how much information one FL communication round exposes about a client updates, i.e., leakage_ratio_per_round = MI_per_round_bits/original_update_bits, where MI_per_round_bits is defined by the sum of the value leakage and selected-index leakage. Consider the accumulate residual may lead to information leakage, MI_saturation is also introduced to measure the accumulated mutual-information upper bound, that is, after how many FL rounds the accumulated leakage would bound equal one full client update.

Federated CSI prediction experiments on the KUL CSI dataset.

Given the KUL dataset, the training rationale is set to evaluate the proposed algorithm and baselines on a real wireless channel prediction task among different clients. The open CSI dataset contains sequential CSI measurements from four user trajectories, so the number of the clients in this case is set to four. Each CSI sample denotes complex-valued channel measurements over antennas and subcarriers. Considering the limitation of client numbers, the task is designed to ask the model to predict the next

CSI state from a short history of previous CSI states instead of classifying a user or localization. For instance, at time *t*, using past CSI window *H[t-W+1: t]* to predict the next CSI *H[t+1]*, where *W* denotes the window size. The input of the model training is a sequence of *W* past channel vectors, and the target is the future channel vector. The experiment uses a compact transformer for the prediction task. Each client trains on one client, where the FL setting is naturally non-IID. Mean Square Error (MSE) is selected as the model evaluation metric, where a lower MSE indicates the global model predicts future channel states more accurately.

Table 18 shows that the ProposedTopK statistically achieves comparable clean MSE (mean) with FLAME method, which obtaining a high uplink communication reduction and lower leakage than full-update method. FLAME has the best average MSE while leaks full client updates. FedDPRoC has the best uplink communication reduction profile, but much weaker model utility and privacy than ProposedTopK with respect to the privacy leakage per round and saturation rounds.

Table 18. Average experimental results comparisons on CSI datasets without Byzantine attacks (no attack).

Algorithm	**MSE (mean)**	**Leakage /round**	**Saturation rounds**	**Uplink Comm. reduction**	**Upload Bytes/ Client**	**Original Model Bytes**
EF21-MAD	0.9509	6.75e-03	149	93.25%	32,768	485,376
ByzDASHAPAGE	0.9285	6.75e-03	149	93.25%	32,768	485,376
SparseFed	0.0773	4.04e-02	25	93.25%	32,768	485,376
FedDPRoC	0.6946	1.72e-03	582	**99.58%**	2,056	485,376
FLAME	**0.0522**	1.00	1	0.00%	485,376	485,376
Clover	0.1880	7.23e-03	139	93.25%	32,768	485,376
ProposedTopK	**0.0523**	**3.60e-05**	**27823**	93.25%	32,768	485,376

Table 19 presents that FLAME has the best random-attack MSE and filters the attack across the communication rounds; however, it has full leakage and no compression of the upload bytes. ProposedTopK achieves zero Byzantine attacker kept and 93.25% communication reduction. However, it has lower model performance to achieve better privacy protection under the random attack setting.

Table 19. Average experimental results comparisons on CSI datasets under the random-attack scenario.

Algorithm	**MSE (mean)**	**Leakage/ round**	**Saturation rounds**	**Uplink Comm. reduction**	**Byzantine kept**
EF21-MAD	0.9627	6.75e-03	149	93.25%	0.8250
ByzDASHAPAGE	0.9892	6.74e-03	149	93.25%	0.9583
SparseFed	0.1708	4.04e-02	25	93.25%	1.00
FedDPRoC	0.7050	1.72e-03	582	**99.58%**	1.00
FLAME	**0.0531**	1.00	1	0.00%	**0.00**
Clover	0.2292	7.23e-03	139	93.25%	1.00
ProposedTopK	0.2619	3.36e-05	29952	93.25%	**0.00**

These initial empirical results show that the current ProposedTopK design has clear advantages, but there is still space for implementation and improvement, such as performance analysis under different attack scenarios, comprehensive analysis of convergence and trade-offs, as well as the integration with 6G-DALI's FLOps solution at the next steps.

5.3.3.4 Next steps

The next steps of this work towards algorithmic novelty and full integration with 6G-DALI include:

- Complete the theoretical analysis of the proposed algorithm and provide trade-off analysis

among privacy protection, communication reduction, and Byzantine robustness while preserving the model utility;
- Complete the development of the proposed algorithm and relevant baselines and evaluation experiments; and
- Integrate the FL algorithm(s) within 6G-DALI framework by working together with relevant partners.

5.3.4 Split Learning

This section presents the Split Learning solution developed within the 6G-DALI framework to support efficient distributed model training across heterogeneous cloud–edge infrastructures. The section first reviews the relevant state of the art, focusing on split learning as a collaborative training paradigm and on the communication overhead introduced by cut-layer exchanges. It then introduces the proposed framework, which extends an open-source split-learning framework with a GUI-supported execution environment, quantization-based transmission strategies, adaptive precision control, communication monitoring and MLflow-based experiment tracking. Finally, initial experimental results demonstrate the practical applicability of the approach through baseline experiments and quantization-based communication reduction campaigns.

5.3.4.1 State of the Art

Split learning has emerged as a distributed learning paradigm complementary to conventional federated learning for collaborative model training in resource-constrained environments. In federated learning, each client typically trains a full local replica of the model and exchanges model updates with a coordinating server. In split learning, the model itself is partitioned at a cut layer between client and server. The client computes the front part of the network, transmits the cut-layer activations and the server continues the rest of the computations and begins backpropagation after computing the loss. In turn, the client receives the corresponding gradients and makes the necessary updates to the weights. In this way, raw training data remain local while part of the computational burden is shifted from the client to the server[89] . This makes split learning particularly relevant for cloud–edge and 6G settings with heterogeneous device capabilities [90] .

Following the original SplitNN formulation, the literature has focused mainly on improving the practicality of split learning in distributed environments. One important direction combines split execution with federated aggregation, giving rise to hybrid split–federated schemes such as SplitFed [91] [90] , which aim to improve scalability, training speed and support broader client participation while retaining the reduced client-side workload of split learning. A second major direction concerns privacy and security hardening, motivated by the fact that intermediate activations and gradients at the client–server boundary may still leak information[92] . A third major direction concerns communication efficiency[93] .

Communication efficiency has become one of the principal bottlenecks of practical split learning. Unlike conventional federated learning, where communication is largely concentrated in synchronization rounds, split learning requires repeated bidirectional transmission of weights and gradients during training. The resulting cost depends strongly on the cut-layer location, since the size of the transmitted data strongly depends on the placement of the split point and the model architecture. For this reason, cut-layer exchange can become a dominant systems-level limitation in realistic deployments[93] .

To address this limitation, prior work has explored several communication-reduction directions. Recent studies that systematize these efforts generally distinguish three main families of approaches. The first relies on sparsity-based reduction, where only a subset of the exchanged tensor elements is transmitted, for example through top-k selection. The second focuses on lower-dimensional representation, where the cut-layer tensor is projected into a more compact form, for instance through principal component analysis or autoencoder-based compression. The third relies on quantization, where the exchanged activations and gradients are mapped to a smaller set of discrete values before transmission or represented with reduced numerical precision [94] . However, although these approaches reduce the amount of data exchanged at the cut layer, they are commonly studied as separate compression mechanisms rather than as part of a unified experimentation framework.

Among the three identified families, quantization is particularly attractive because it is comparatively simple to apply and introduces limited additional processing overhead. This simplicity, however, does not make quantization equally effective in all cases, since its performance depends strongly on the statistical properties of the exchanged tensors[95] . The numerical range and dispersion of the cut-layer representation may vary substantially across model architectures, split locations and training stages. As a result, a single fixed quantization policy is often not equally suitable across all settings. This motivates tensor-level dynamic quantization, where the bit width used to transmit cut-layer activations and gradients is adapted during training according to their observed distributional behavior, rather than being kept fixed for the exchanged tensors. Such strategies are especially relevant when communication reduction must be balanced against convergence stability and predictive performance[96] .

As indicated above, existing split learning approaches often address communication efficiency through isolated compression mechanisms, while the joint treatment of tensor-level bit-width adaptation and reproducible experiment comparison remains underdeveloped. These gaps motivate the development of a unified framework that evaluates different methods side by side on the same exact settings and utilizes the statistical properties of the tensors to provide adaptive quantization-based transmission, while at the same time supporting reproducibility and systematic performance evaluation in a 6G MLOps setting.

5.3.4.2 Solution description

The proposed solution in 6G-DALI bridges the gaps identified in the SoTA by extending the SLPerf framework[97] into a GUI-supported split-learning environment that combines various quantization methods, adaptive precision control, runtime communication monitoring and MLflow-based reproducibility support. In the SoTA analysis, existing approaches provide limited comparisons only between a few selected methods and do not utilize the communication metrics to dynamically adjust the bit width of the quantization methods. In 6G cloud–edge environments, which have a variety of strict requirements to satisfy, comparing different compression methods across heterogenous configurations and using non ideal bit levels is not viable. An overview of the full framework and component interactions is illustrated in Figure 33.

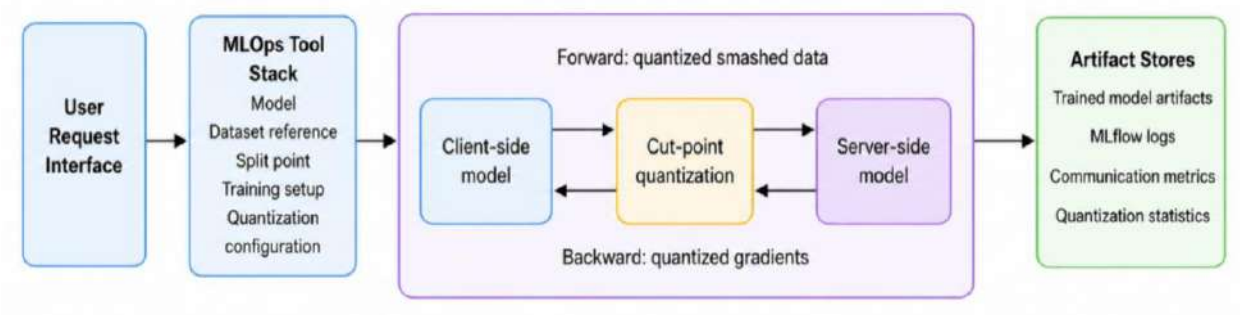


Figure 33: Integration of the proposed framework in the DALI architecture.

Common execution environment

A central design principle of the proposed split-learning solution is the use of a common execution environment for multiple distributed training configurations, rather than separate implementations for each split-learning variant. The solution builds on the SLPerf framework and extends it into a GUI-supported training environment, where users can configure and launch split-learning experiments through a structured menu-based interface.

The execution model is built around an MPI-based distributed runtime that assigns one process to the server role and one or more processes to client roles. The same runtime is also used for hybrid split-learning schemes, such as SplitFed, where an additional federation-side server role is required. Communication between these roles is handled internally through MPI communication threads and a message-dispatch mechanism, which coordinate the transfer of cut-layer activations during the forward pass and activation gradients during backpropagation. At the same time, the runtime records communication metadata, including message type, payload size, communication category, send and

receive volume and per-epoch statistics. This allows communication overhead to be measured directly during training instead of being reconstructed afterwards from external logs.

This communication logging is complemented by tensor-level analysis. While the communication metadata captures how much data is exchanged and when, the tensor analysis characterizes the numerical properties of the transmitted activations and activation gradients. In particular, the runtime records tensor-specific indicators such as tensor shape, mean and standard deviation, as well as indicators showing whether the tensor values are concentrated near zero or affected by large outliers. These measurements make it possible to relate communication cost and compression behavior to the actual distribution of the exchanged tensors, which is especially important when evaluating quantization strategies. The collected information can then be exported through plots and a structured XML workbook for post-experiment comparison and reporting.

To support realistic distributed-learning scenarios, the same implementation can be executed under different data partitioning settings. These include homogeneous partitioning, Dirichlet-based heterogeneous partitioning, disjoint-label partitioning, class-based partitioning, attribute-based partitioning, vertical partitioning and class-intersection-based partitioning. This allows compression and quantization behavior to be assessed under different forms of client-side data heterogeneity, rather than only under a single fixed partitioning assumption.

The execution environment also supports flexible device placement, allowing experiments to reflect different hardware availability conditions. The server process can run on GPU while client processes fall back to CPU when GPU resources are limited. When sufficient hardware is available, client processes can also be placed on a GPU. This flexibility allows the same environment to cover both resource-constrained edge-oriented setups and more compute-rich scenarios.

Quantization and adaptive precision control

Quantization is applied to the tensors exchanged between the client-side and server-side model partitions during split-learning training. The implementation allows compression to be applied selectively to the forward direction, where cut-layer activations are transmitted from client to server, to the backward direction, where activation gradients are returned from server to client, or to both directions. This directional control is combined with a list of configurable codec and bit-width settings, allowing the user to select various available scenarios to compare.

In addition, the solution includes a dynamic quantization mechanism that updates the active transmission precision during training according to the runtime distribution statistics computed from the exchanged tensors. More specifically, the controller observes them during execution and sets a baseline score during initial warmup epochs. After the baseline is set, these indicators are then used to decide whether the transmitted activations and activation gradients can be represented with a lower bit-width or should remain at a higher precision, following predefined decision rules.

MLflow is also integrated into the framework, so selected comparisons are made directly available across the various experiments run on the testbed

Integration within 6G-DALI

To integrate this within the 6G-DALI pipeline, a dedicated wrapper for the split-learning framework is currently in its initial design phases as part of the 6G-DALI MLOps Tool Stack. The wrapper acts as the execution bridge between the User Request Interface, the MLOps workflow configuration and the underlying split-learning implementation. The workflow begins when a split-learning request is submitted through the User Request Interface and passed to the MLOps Tool Stack, where the model, dataset reference, split point, training setup and quantization configuration are defined. The wrapper then starts the client/server training procedure, applies the selected cut-point quantization strategy to the smashed data and gradients and records the resulting trained model artifacts, MLflow logs, communication metrics and quantization statistics in the artifact stores.

5.3.4.3 Initial results

During the initial phase, the main effort was devoted to making the split-learning framework operational across the currently integrated training modes and to establishing a first common baseline for

comparison. At the implementation level, the environment already supports vanilla split learning, parallel split learning, SplitFed, SplitFed2, Federated learning and a central server reference configuration. Adding, a lightweight graphical launcher and a common post-processing workflow are already available as seen in Figure 34. so that runs can be configured in a uniform way and exported into structured summaries for later comparison.

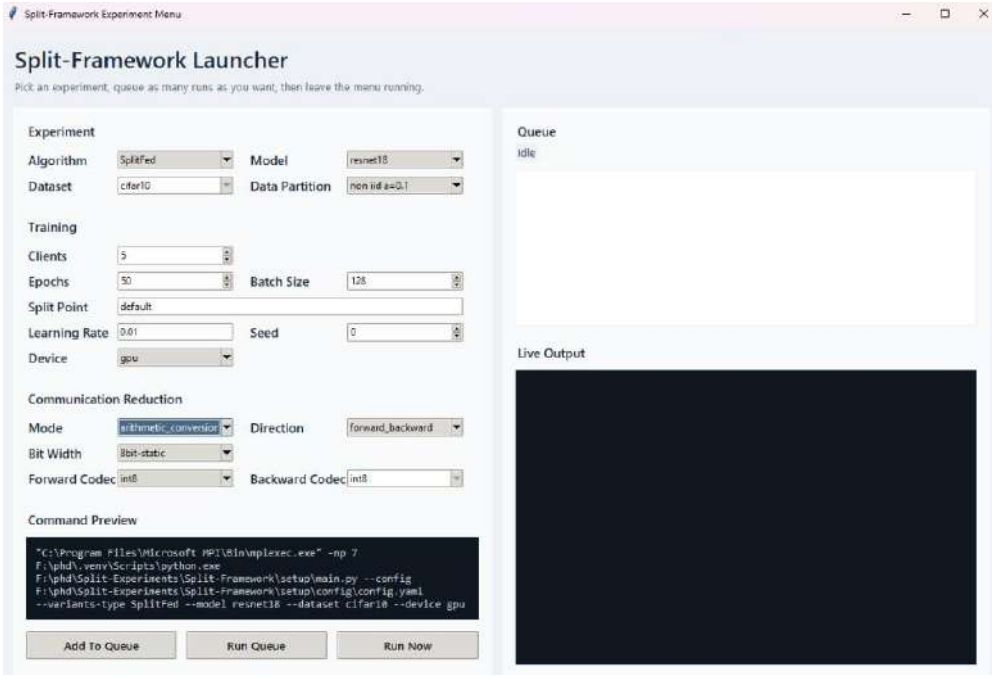


Figure 34: The GUI for controlling all of the available options analyzed in the solution description.

Baseline experimental campaign

A first baseline experimental campaign has been completed on a workstation equipped with an AMD Ryzen 9 7900X CPU and an NVIDIA RTX 5070 GPU. In the current phase, the evaluation has focused on the ResNet18 model and on non-quantized training runs. Table 20 summarizes the training settings used in this baseline campaign, covering the 1-client case together with 3-client and 5-client experiments under IID and non-IID a=0.5, a=0.1 data partitions. For each setting, the dataset is partitioned into as many partitions as there are active clients. Consequently, the 1-client run is executed over 1 partition the 3-client runs are executed over 3 partitions and the 5-client runs over 5 partitions, after which the selected IID, Dirichlet-based heterogeneous, or extreme non-IID policy is applied across those active clients.

Table 20. Training settings used for all the experiments

Dataset	Model	Epochs	Batch size	Learning rate	Optimizer	Random seed
Cifar-10	Resnet18	50	128	0.01	Adam	0

At the present stage, runtime is not used as a primary comparison metric across algorithms. The reason is that the current training modes do not consume CPU resources under equivalent execution conditions. In the vanilla implementation, clients proceed sequentially through the protocol and hand over control through the semaphore-based execution flow, so each client effectively uses the fully available client-side CPU time during its own turn. By contrast, in parallel variants such as Parallel, SplitFed, and SplitFed2, multiple clients are active concurrently and therefore share the available CPU resources. For this reason, wall-clock runtime cannot yet be treated as directly comparable across all algorithms and the current baseline analysis focuses instead on training behavior and validation performance under the tested partition settings.

Observed baseline behavior

As shown in Table 21, the baseline results highlight the influence of data partitioning on split-learning performance. The homogeneous setting provides the most consistent validation behavior, while stronger non-IID configurations lead to a gradual degradation in performance. This indicates that the split-learning process is sensitive not only to the model and training configuration, but also to the way the training data are distributed across clients.

The effect becomes more visible as the number of clients increases. Moving from the 1-client case to the 3-client and 5-client settings introduces a more distributed training scenario, in which differences between local client datasets have a stronger impact on the overall learning process. As a result, the gap between favorable IID conditions and more challenging heterogeneous partitions becomes wider, showing that client count and data distribution jointly affect the observed training behavior.

Table 21. Non quantized results of the vanilla split algorithm showcasing the different partition effects on training.

Clients	a	Data partition	Final acc
1	-	IID	0.758
3	-	IID	0.771
3	0.5	Non-IID	0.601
3	0.1	Non-IID	0.368
5	-	IID	0.770
5	0.5	Non-IID	0.693
5	0.1	Non-IID	0.467

8-bit quantization campaign

To showcase the compression capabilities side of the framework, this part focuses on fixed 8-bit quantization. In this phase, the transmission of the cut-layer tensors is modified for both the forward and backward propagation. The tested methods already cover two different quantization families. The arithmetic family represents the exchanged tensor by converting its values directly into a reduced numerical format, such as FP8 or scaled INT8. The codebook family instead represents the tensor through indices that point to reconstructed values stored in a codebook. In the current implementation, the arithmetic side includes FP8 E4M3, per-tensor INT8 and per-channel INT8. The codebook side includes a uniform quantizer as well as non-uniform μ-law and Lloyd-Max quantizers. In Table 22, we can see the results of each quantized method regarding accuracy loss and data savings.

Due to all methods having the same bit width, the data savings are the same. What changes is the accuracy retention, of which we can see that in the specific case almost all codecs achieve near baseline final acc.

Table 22. Splitfed 5 clients quantization results directly from the exported logs

Method	Final acc	Acc loss	Data transferred (GB)	Data saved (%)
FP8	0.6194	0.1939	114.4	74.99
INT8	0.8211	0.0078	114.4	74.99
INT8 per channel	0.8165	0.0032	114.41	74.98
Lloyd-Max	0.8221	0.0088	114.46	74.97
μ-law	0.8188	0.0055	114.45	74.97
Uniform	0.8096	0.0037	114.45	74.97

Dynamic approach

To showcase the adaptive side of the framework, this part focuses on dynamic bit-width selection for the exchanged tensors that can be applied to any of the codecs. It starts from an initial warm-up period of 5 epochs, where the behavior of the transmitted tensors is observed and used to form a reference baseline score. After this, once per epoch a comparison is made between the current tensor score and the baseline so either 8-bit or 4-bit transmission is selected.

Two dynamic criteria were tested considered. One consists of using the score based on the relation between the p99 and p95 absolute tensor values, so the decision mainly follows how heavy the upper

tail of the tensor distribution becomes. The other uses a richer composite score, combining outlier behavior, tensor spread and channel-scale variation. In Table 23, the methods are compared to 8bit int quantization. We can clearly see that the data savings are significant with varying degrees of accuracy retention. The simple p99 score achieved significant data savings, yet the final results were too heavily reduced, while the composite score shows a better compromise.

Table 23. 5 clients splitfed iid run showcasing the dynamic quantization results

Method	Final acc	Acc loss	Data transferred (GB)	Data saved (%)
P99/p95	0.7098	0.1113	68.67	39.97
Composite	0.7976	0.0235	88.11	22.98

MLOps Experimentation

Beyond the training and compression mechanisms, the framework also supports the MLOps side of the experimentation process through structured run tracking and comparison as seen in Figure 35. Each experiment is recorded togeth5.er with its main configuration, including the dataset, model, number of clients, data partitioning policy, quantization method, dynamic mode and training hyperparameters. At the same time, both learning and communication-related metrics are tracked, allowing runs to be compared under the same experimental structure, instead of being evaluated as isolated results.

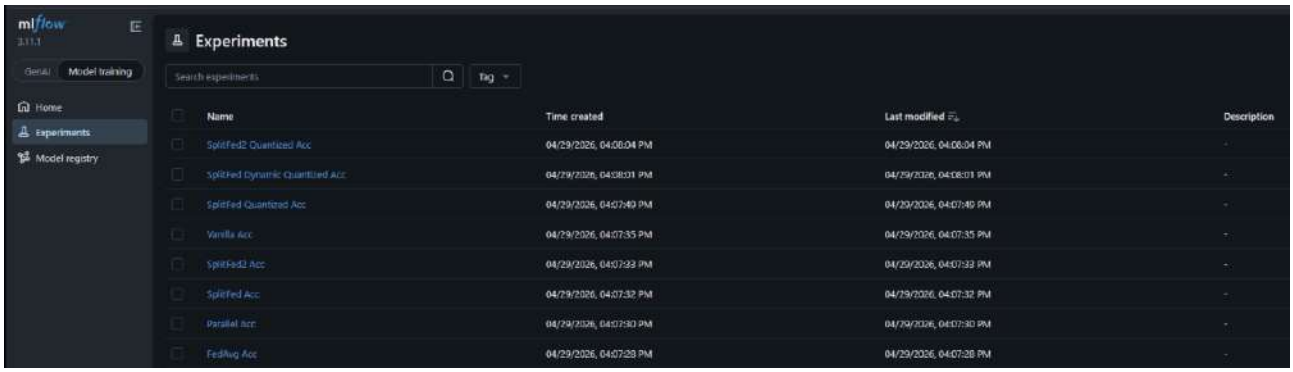


Figure 35: All MLflow experiment runs performed.

5.3.4.4 Next steps

The next steps will focus on finalizing the integration of the split-learning framework into the 6G-DALI MLOps workflow and extending the evaluation beyond communication savings and accuracy. In particular, future work will study the energy efficiency of split learning under different client counts, split-layer positions, quantization strategies and cloud–edge deployment conditions. This will make it possible to assess not only how much data is saved, but also how different configurations affect computation cost, communication cost and overall energy consumption across the distributed training pipeline.

5.4 Conclusion

This chapter described the distributed training and workload execution layer of the 6G-DALI pipeline. The ARPA workload placement agent demonstrated a 25% reduction in GPU inference energy consumption relative to a round-robin baseline in initial simulation over 80,000 training steps. The ProposedTopK federated gradient aggregation algorithm reduced communication volume by over 93% while maintaining model accuracy alongside Byzantine-resilient gradient filtering. The FL algorithm for joint communication and sensing positioning improved positioning accuracy by 30% across seven propagation scenarios relative to purely local training. The split learning framework reduced data transmission by 75% under quantization, with adaptive composite scoring providing further savings. The NWDAF-integrated FLOps architecture establishes the 3GPP Release 17/18-compliant federation infrastructure that will host these algorithms in the next project phase. All four components are at prototype or early validation stage; live testbed integration, evaluation against the 6G-DALI dataset collection, and deployment within the NWDAF FLOps pipeline are the primary next steps.

6 Digital Twin Testbed for Large-Scale Experimentation

6.1 Overview

The mechanisms described in Sections 3, 4, and 5 require a simulation environment capable of reproducing 6G network conditions at testbed-inaccessible scales; the Digital Twin Testbed documented in this section provides that environment. The 6G-DALI architecture comprises a Digital Twin responsible for large scale Data generation along with RLOps pipeline for validating RL algorithms. The large scale Digital Twin complements the high-fidelity Testbeds and together allow for generating massive datasets, registering them in the Data Space and storing them in the Data Lake. Physical radio testbeds impose hard upper bounds on experimental scale: a live O-RAN deployment cannot simultaneously represent the UE densities, mobility patterns, and traffic heterogeneity that production 6G networks generate in the field. Objective 5 of 6G-DALI addresses this gap by specifying a Digital Twin Testbed (DTT) that decouples experimental scale from physical hardware, targeting validated AI and RAN workloads at scales that physical deployments cannot reach. VIAVI's AI-RSG platform generates synthetic radio environments at up to 300,000 UEs across various configurations. Paired with the RLOps component, it would be possible to validate RL algorithms on the Digital Twin Testbed

Another requirement is having large-scale low-fidelity, and it is addressed by the AI-RSG and FlexRIC integration documented in Section 6.2, where the KPM xApp (KPM SM v2.03) and RC xApp (RC SM v1.03) have been validated end-to-end over the E2AP v2 interface. The DTT that supports the O-RAN architecture, connecting VIAVI's simulation layer to EUR's FlexRIC near-RT RIC, producing a fully O-RAN-compliant data path. Having At least two open datasets delivered via the DTT, is targeted by the DTT large-scale data generation pipeline described in Section 6.3, with dataset export to the 6G Data Lake. The RLOps pipeline, running on OMNeT++ and described in Section 6.4, additionally advances the project objectives by integrating RL policy training with the digital twin environment, though the full closed-loop with the digital twin testbed remains in progress and constitutes a primary target in the next phases of the project.

Referring to Figure 1, the Digital Twin Testbed is a component within the infrastructure layer of 6G-DALI's architecture

6.2 AI-RSG and FlexRIC Integration

The joint VIAVI and EUR integration connects VIAVI's AI-RSG synthetic radio environment to EUR's FlexRIC near-RT RIC controller, producing a validated end-to-end data pipeline in which simulated UE and cell measurements drive xApp logic running unmodified on the FlexRIC platform. The E2 interface is the critical integration boundary at which VIAVI's simulated network meets EUR's control logic, and the pipeline operates over E2AP v2, KPM SM v2.03, and RC SM v1.03.

6.2.1 State-of-the-art

Digital twin (DT) networks have emerged as a foundational concept in 6G research, providing virtual replicas of physical RAN infrastructure that support AI model training, policy validation, and scenario exploration without disrupting live deployments. Khan et al. [98] articulate the architectural vision for DT-enabled 6G, highlighting the role of DTs in closed-loop optimisation and predictive network management. Lin et al. [99] trace the progression from theoretical frameworks to practical DT implementations spanning RAN, core, and edge layers, while Tao et al. [100] demonstrate how generative AI can drive the fidelity needed for data-efficient model adaptation. Prior 6G-DALI work [2] has empirically validated using such DTs for the training and evaluation of RL-based RAN controllers, establishing the direct connection between high-fidelity simulation and deployable policy generation that underpins the testbed approach in this chapter.

Synthetic RAN dataset generation has become a core enabler for AI/ML model training in 6G research, where the data volumes, scenario diversity, and UE scales required for robust model development are difficult to obtain through live network collection alone. Early approaches relied on system-level channel simulators such as MATLAB's LTE/NR Toolbox and QuaDRiGa [101] , which produce per-link datasets well-suited to physical-layer algorithm development but do not capture the RAN-layer KPIs including scheduling decisions, slice metrics, and mobility events needed by network intelligence

applications. More recent tools, including the 5G-LENA ns-3 module [102] and NVIDIA Sionna [103] , have extended synthetic generation toward RAN-layer outputs, though each imposes constraints on scale, scenario configurability, or O-RAN interface availability.

Coupling synthetic data generation with O-RAN-compliant interfaces introduces a requirement that goes beyond simulator fidelity: datasets should be produced via the same E2SM-KPM subscription path that real near-RT RIC deployments use, so that models trained on synthetic data generalise to measurements from live O-RAN nodes. This distinguishes O-RAN-native generation, where a KPM xApp subscribes to the simulation's E2 agent exactly as it would to a physical gNB, from approaches that post-process simulation logs into KPI-shaped tables, which bypass the protocol stack and may introduce measurement artefacts absent in live deployments.

Open-source tools that expose an E2 interface, primarily ns-O-RAN [104] which couples ns-3 with a near-RT RIC, and radio emulation approaches such as OAI's RF simulator, generally support small-to-medium scale scenarios in the range of tens to a few hundred UEs, with limited coverage of heterogeneous traffic profiles across multiple 5QI classes or large-scale MIMO beamforming configurations. No publicly documented tool combines 300,000-UE simulation scale, 27 5QI traffic type configurations, O-DU/O-CU deployment emulation, and a live E2SM-KPM data collection path within a single integrated environment.

On the near-RT RIC tooling side, FlexRIC [105] provides an SDK for next-generation software-defined RANs with native E2AP and E2SM support, and serves as the near-RT RIC platform in the 6G-DALI integration. OpenRAN Gym [106] introduced an open toolbox for end-to-end xApp design, data collection, and testing workflows in O-RAN, demonstrated on the Colosseum network emulator across softwarised networks of up to 10 base stations and 60 users. ColO-RAN [107] extended this with scalable machine learning-based xApp closed-loop control on programmable experimental platforms. While these tools establish the O-RAN AI/ML experimentation baseline, they target small-to-medium deployments and do not address the 300,000-UE scale or multi-5QI scenario diversity required for robust 6G model training.

The AI-RSG large-scale data generation component described in this section addresses these gaps by providing a commercially validated, O-RAN-native synthetic dataset generation pipeline at scales and scenario configurations not currently available from open-source alternatives. The resulting datasets are exported to the 6G Data Lake via the Dataspace Connector, providing the training and validation data supply for AI/ML models developed across WP4.

6.2.2 End-to-End Workflow

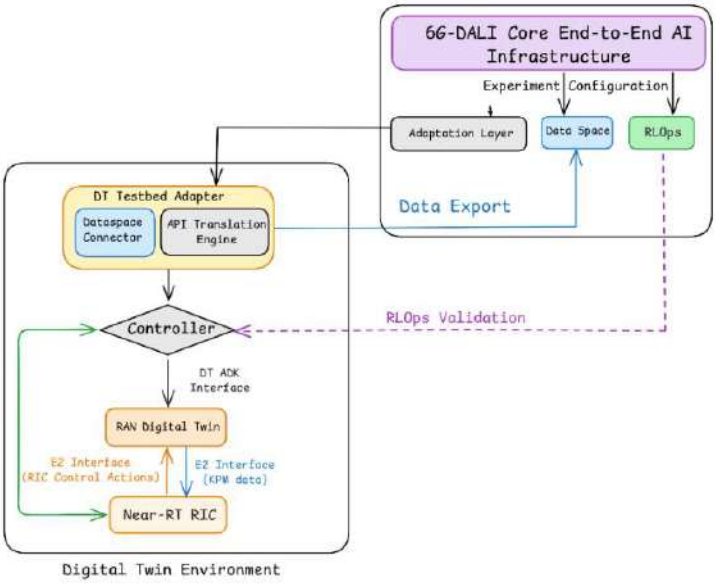


Figure 36: VIAVI Digital Twin Testbed Architecture — End-to-End Integration with 6G-DALI Core Infrastructure

Figure 36 illustrates the end-to-end architecture connecting the VIAVI Digital Twin Environment to the other 6G-DALI layers. The architecture is organised into two main domains: the Digital Twin Environment (lower block) and the other 6G-DALI blocks (upper block). Within the Core Infrastructure, three components interact with the Digital Twin Environment: the Adaptation Layer, which receives Experiment and Configuration requests and forwards them downstream; the Data Lake, which acts as the storage repository for exported KPI datasets (indexed and made discoverable through the Data Space) ; and RLOps, which provides the reinforcement learning validation pipeline.

Inside the Digital Twin Environment, the DT Testbed Adapter is the integration bridge to the 6G-DALI ecosystem. It contains two sub-components: the API Translation Engine, which converts upper-layer requests arriving from the Adaptation Layer into internal commands; and the Dataspace Connector, which handles the export of collected KPI datasets to the 6G-DALI Data Lake (the Data Export path shown in the diagram). Translated commands are passed from the DT Testbed Adapter to the Controller, which is the central orchestration element responsible for managing the simulation lifecycle: scenario configuration, launch, pause, and stop. The Controller drives the RAN Digital Twin via the DT ADK Interface and also receives RLOps Validation signals from the RLOps component, which activate the RL validation mode.

The RAN Digital Twin is the core simulation engine. It communicates bidirectionally with the Near-RT RIC (FlexRIC) over the O-RAN E2 interface: KPM data — cell-level and UE-level performance metrics — flows from the RAN Digital Twin to the Near-RT RIC via E2SM-KPM subscriptions; RAN control actions flow in the reverse direction from the Near-RT RIC to the RAN Digital Twin via E2SM-RC messages. A closed-loop feedback path connects the Near-RT RIC output back to the Controller, enabling orchestration decisions based on observed state. Depending on the experiment objective, the Controller launches different xApp configurations on the Near-RT RIC: for data generation, KPM xApps collect synthetic datasets for later export to the Data Lake; for RL validation, KPM and RC xApps are deployed simultaneously: the KPM xApp provides state observations while the RC xApp applies RL agent decisions as RAN control actions.

A dedicated Python-based ADK script has been developed to automate scenario execution. The script handles E2 configuration, protocol versioning (E2AP v2, KPM v2), and procedure setup without manual intervention, running in real-time mode with E2 enabled and combined deployment active. For RL validation specifically, the ADK Pacing mode allows the Controller to synchronise simulation time with RL training steps: the simulation is paused, the current network state is observed, an RL action is applied via the RC xApp, and the simulation is then resumed, enabling reproducible, step-by-step policy evaluation. FlexRIC has been containerised using Docker, improving portability and allowing multiple xApps to run simultaneously without interference.

6.2.3 KPM and RC xApps (EUR)

The xApp layer consists of two Python processes: a Key Performance Metrics (KPM) xApp on port 5000 and a RAN Control (RC) xApp on port 5001, both bound to FlexRIC through the FlexRIC xApp SDK and validated running concurrently against the same FlexRIC instance without interference. FlexRIC is deployed from a Docker image with eight E2 service models loaded (KPM, RC, MAC, RLC, PDCP, SLICE, GTP, and TC), and the E2 interface is configured for E2AP v2.00, E2SM-KPM v2.03, and E2SM-RC v1.03. Stable operation requires Combined deployment mode with E2AP v2.00; alternative combinations including E2AP v3.00, standalone deployment, or E2SM-CCC enabled at the agent trigger assertion failures in the FlexRIC ASN.1 decoder and are not currently usable. To reach the validated configuration, the AI-RSG Python ADK is set to suppress node configuration update procedures, the E2 node component configuration list in the E2 Setup Request, and Cell Configuration and Control (E2SM-CCC). This compatibility constraint limits the experiment configurations currently available and motivates a planned FlexRIC rebuild with broader E2AP support.

The KPM xApp subscribes to FlexRIC using E2SM-KPM Style 1 (RIC Service Report) subscriptions at a default reporting period of 1000 ms, with a single subscription targeting multiple gNBs, cells, and UEs simultaneously so that large-scale scenario data is collected in one run. Cell-level and UE-level indications are written to CSV; this CSV output is the data path consumed by the Dataspace Connector on its way to the 6G-DALI Data Space. Subscription targets are identified using the Viavi.NrCgi and

Viavi.GnbDuId fields retrieved from the cell reports stream at scenario start. A REST interface on port 5000 accepts POST /start and DELETE /stop requests through which the DT Testbed Adapter or the RLOps pipeline manages the xApp lifecycle.

The RC xApp issues RAN control actions via E2SM-RC messages, driven by a POST /subscribe REST endpoint that accepts a JSON body specifying target cells and distributed units (DUs) through the same Viavi.NrCgi and Viavi.GnbDuId fields used by the KPM xApp, together with the control action to apply. Handover control actions have been executed and validated end-to-end in the integrated environment. Five additional AI-RSG control decisions are functional but not yet validated as of this deliverable: UE admission control, slice-level Physical Resource Block (PRB) quota, DRB QoS configuration, carrier aggregation control, and non-GoB beamforming mode configuration. The KPM xApp's CSV output feeds the state input path of the RLOps pipeline; the RC xApp closes the actuation loop by returning policy decisions as E2SM-RC control requests to the AI-RSG.

6.3 Large-Scale Data Generation

The AI-RSG Digital Twin generates large-scale synthetic RAN datasets on demand, providing the training data supply for experiments where live network collection is impractical or impossible at the required scale and scenario diversity.

The AI-RSG platform simulates 5G NR RAN using NS3 v3.0 with 3GPP TS 38.901 channel models across four propagation scenarios: Urban Macro (UMa), Urban Micro Street Canyon (UMi), Rural Macro (RMa), and Indoor Hotspot (InH), each configurable as Line-of-Sight (LoS), Non-Line-of-Sight (NLoS), or probabilistic. The emulation covers O-DU and O-CU deployments (O-CU-CP and O-CU-UP) across thousands of cells, with Massive MIMO (mMIMO) beamforming in 3GPP analog, digital, and hybrid configurations.

UE emulation scales to 300,000 concurrent users, organised into groups each with independently configured mobility models (Drone, Short-range Trajectory, Manhattan/Berlin Path, and Indoor), device profiles (MIMO capability, Carrier Aggregation, antenna count), and traffic assignments drawn from 27 standardised 5QI types covering GBR and non-GBR resources across eMBB, URLLC, and mMTC slices.

A lightweight 5GC emulation covers PDU session management, QoS flow handling, and network slice support without requiring a full core deployment, enabling RAN scheduling decisions to be evaluated under differentiated per-slice QoS conditions.

Network topologies are imported via the O1 CM/NETCONF interface, specifying cell locations, antenna heights, frequency bands, tilt, and azimuth to replicate real-world deployments. Both O1 and E2 interfaces support runtime reconfiguration, so scenario parameters can be varied dynamically mid-run without restarting the simulation.

A data generation run begins with an experiment request specifying scenario parameters, simulation duration, and required KPIs. The DT Testbed Adapter translates this request and configures the RAN Digital Twin via the Controller. A KPM xApp is launched on the Near-RT RIC to subscribe to E2SM-KPM reports over the E2 interface, collecting cell-level and UE-level metrics for the run duration. The resulting dataset is exported to the 6G-DALI Data Lake via the Dataspace Connector; a completion notification is returned to the requesting entity.

6.4 RLOps and DT

This section describes IQU's RLOps pipeline and its initial integration with the Digital Twin Testbed (DTT). The purpose of the integration is to enable closed-loop reinforcement learning (RL) experimentation using O-RAN-compliant interfaces and large-scale synthetic RAN environments generated by the VIAVI Digital Twin. The RLOps framework complements the DTT architecture presented in Figure 36 by providing the AI lifecycle management layer responsible for RL training, evaluation, experiment orchestration, and policy validation.

6.4.1 State-of-the-art

Machine Learning Operations (MLOps) has matured as a structured methodology for managing the end-to-end lifecycle of machine learning systems, including development, validation, deployment, and

monitoring [108] . In telecommunications, MLOps and AIOps paradigms are increasingly applied to automate network functions and improve operational efficiency, particularly in the context of emerging 6G architectures [109] . European 6G initiatives such as ADROIT-6G [110] has explored AI-driven orchestration and distributed intelligence frameworks for network management. In parallel, the O-RAN Alliance has standardised the architectural separation between the non-RT RIC and near-RT RIC, enabling AI-driven control loops operating at different timescales [111] .

Within this context, RL has emerged as a promising approach for dynamic radio resource management, mobility optimisation, slicing, and adaptive network control. However, while significant effort has focused on RL algorithm design, considerably less attention has been given to the operational lifecycle management of RL systems in telecom environments. In contrast to conventional supervised ML pipelines, RL introduces additional operational challenges, including continuous interaction with dynamic environments, online policy evaluation, simulator synchronisation, reproducibility of sequential decision processes, and safe deployment of control policies into live or near-real-time network infrastructures.

Existing telecom AI frameworks typically focus either on offline model training or isolated simulation studies, without providing integrated support for reproducible RL experimentation, scalable orchestration, experiment tracking, policy lifecycle management, and O-RAN-compliant deployment workflows. Furthermore, integration between RL pipelines and Digital Twin environments remains limited, particularly for closed-loop experimentation through standardised O-RAN interfaces such as E2SM-KPM and E2SM-RC.

The operational lifecycle of RL systems in open RAN has begun to receive dedicated attention: Li et al. [112] propose RLOps as a structured development lifecycle for RL-aided O-RAN, covering environment abstraction, policy versioning, evaluation, and deployment, but without integration into a DT-backed experimentation loop or closed-loop O-RAN interface support. PandORA [113] introduces automated design and comprehensive evaluation of deep RL agents for Open RAN, addressing systematic agent evaluation across deployment conditions yet without a full lifecycle management framework. A cross-cutting challenge for DT-based RL approaches is the sim-to-real gap: policies trained on simulation models may encounter distributional shift when transferred to physical testbeds due to propagation and traffic modelling simplifications [2]. Mitigating this motivates grounding the RLOps pipeline in a commercially validated, high-fidelity channel emulator rather than a purely open-source simulator.

The RLOps framework developed in 6G-DALI addresses these gaps by extending MLOps principles toward RL-specific operational workflows for O-RAN and Digital Twin environments. Building on the experimentation framework developed in Deliverable D3.1, the proposed approach combines ZenML-based orchestration, RLlib-based distributed training, MLflow experiment tracking, and integration with O-RAN-compliant xApps to support reproducible end-to-end RL experimentation. The framework further introduces a pathway for synchronised closed-loop interaction between RL agents and large-scale Digital Twin environments through FlexRIC and E2-based control interfaces, enabling scalable validation of RL-driven network optimisation strategies under realistic 6G conditions.

Proposed Integration Architecture:

Within the overall architecture, the RLOps pipeline is positioned in the 6G-DALI Northbound infrastructure inside the MLOps as subcomponent. It interfaces with the Digital Twin Environment through the DT Controller and the Near-RT RIC (FlexRIC).

More specifically:

- The KPM xApp deployed on FlexRIC acts as the observation interface between the RAN Digital Twin and the RLOps pipeline. Through E2SM-KPM subscriptions, for instance the xApp continuously receives UE-level and cell-level measurements generated by the Digital Twin over the E2 interface. These measurements include radio performance indicators, QoS statistics, mobility information, traffic load conditions, and slice-related KPIs depending on the configured experiment scenario.
- The collected KPM measurements are exported by the KPM xApp and forwarded to the RLOps pipeline through a data ingestion interface. Within the RLOps framework, these measurements are processed and transformed into RL state representations suitable for training and inference.

This state construction phase formats the observation vectors compatible with RLlib environments.

- The RLOps pipeline orchestrates the RL lifecycle using ZenML, which manages experiment execution, pipeline reproducibility, configuration management, and integration between the simulator, RL training components, and tracking services. RL training itself is performed using RLlib, enabling scalable distributed training and support for multiple RL algorithms.
- During training or inference execution, the RL agent consumes the current network state and generates control decisions intended to optimise network objectives. The generated RL actions are forwarded from the RLOps pipeline to the RC xApp. The RC xApp converts these high-level policy decisions into O-RAN-compliant E2SM-RC control procedures targeting the corresponding cells, distributed units (DUs), or network slices.
- These E2SM-RC control messages are transmitted through FlexRIC over the E2 interface and applied to the RAN Digital Twin, where they modify the simulated network behaviour. Depending on the experiment configuration, the applied actions may influence mobility management decisions, scheduling behaviour, PRB allocation policies. After the control action is applied, the Digital Twin evolves to a new network state and generates updated KPM measurements reflecting the impact of the RL decision. These updated observations are again collected through the KPM xApp and returned to the RLOps pipeline, thereby closing the RL control loop.

During RL validation experiments, the DT Controller coordinates this interaction using ADK pacing mode, synchronising simulation progression with RL decision cycles.

In this architecture, the Near-RT RIC acts as the operational integration layer between the Digital Twin and the RL pipeline, while the DT Controller coordinates simulation execution and pacing. During RL validation experiments, the Controller synchronises simulation progression with RL decision cycles using ADK pacing mode. This enables deterministic and reproducible interaction between the RL agent and the simulated RAN environment.

6.4.2 ZenML-based RLOps Pipeline

A first prototype of an end-to-end RLOps pipeline for RL experimentation in 6G environments has been designed and implemented. The pipeline is built using **ZenML** for orchestration, **RLlib** for scalable RL training, and **MLflow** for experiment tracking and artifact management.

The pipeline supports the full RL lifecycle, including:

- Environment definition and interaction
- Model training and hyperparameter configuration
- Evaluation and validation of trained policies
- Artifact logging (models, metrics, checkpoints) via MLflow
- Reproducible experiment execution via ZenML pipelines

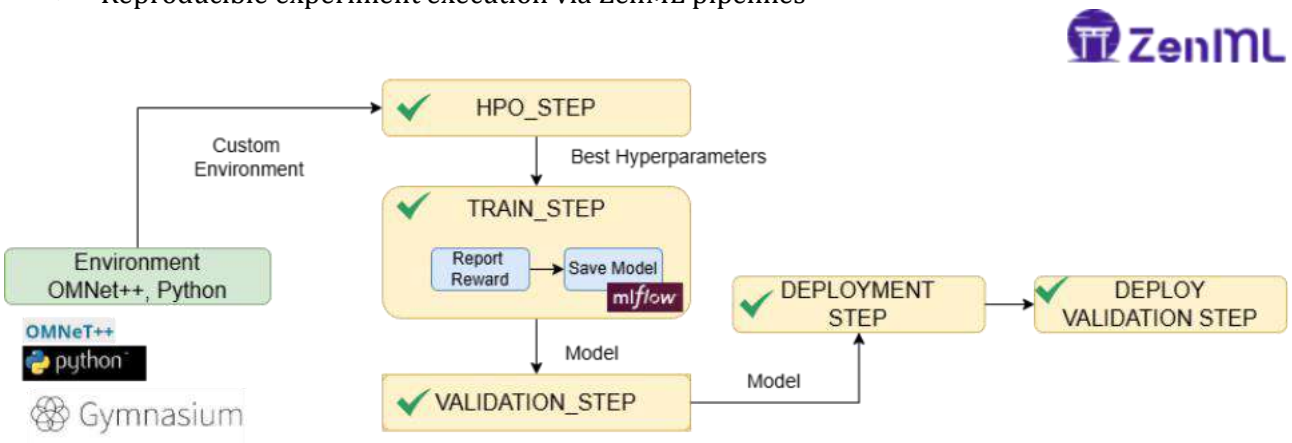


Figure 37: A first prototype of an end-to-end RLOps pipeline

Figure 37 illustrates the current prototype implementation of the RLOps framework. In the current implementation, the RL environment is connected to an OMNeT++-based network simulator, enabling

controlled experimentation and rapid prototyping of RL-based network control strategies before deployment into the VIAVI Digital Twin environment.

6.4.3 Current Status of Integration

At the time of this report, the RLOps pipeline has been successfully implemented and validated using OMNeT++ based simulation environments. Furthermore, the integration with the VIAVI Digital Twin and FlexRIC is ongoing. Finally, no direct interaction between RL agents and xApps has yet been deployed

6.4.4 Integration Results

Preliminary experimentation using the OMNeT++-based environment demonstrates the feasibility of integrating RLOps methodologies with Digital Twin-driven network experimentation environments.

The current prototype has successfully demonstrated:

- End-to-end orchestration of RL experiments using ZenML pipelines
- Automated training and evaluation workflows for RL agents
- Integration of RLlib with network simulation environments for sequential decision-making tasks
- Experiment reproducibility through pipeline-based execution
- Centralised tracking of metrics, checkpoints, and trained models using MLflow
- Stable interaction between RL agents and simulated network environments during training episodes

The obtained results demonstrate that the proposed RLOps framework can support iterative RL experimentation workflows suitable for future integration with large-scale Digital Twin infrastructures. In particular, the OMNeT++ validation phase confirmed:

- The ability to train RL agents using network state observations produced by an external simulator
- The feasibility of separating training orchestration, experiment tracking, and simulation execution into modular services
- The suitability of the architecture for future integration with O-RAN-compliant xApps and E2-based control loops

Although the current experiments do not yet include live interaction with the VIAVI Digital Twin, they validate the core architectural assumptions required for the next integration phase. The transition from OMNeT++ to the VIAVI Digital Twin primarily requires replacing the simulator-specific observation/action interfaces with the KPM and RC xApps already validated within the FlexRIC environment. Consequently, the current results provide a strong technical foundation for extending the RLOps framework toward large-scale O-RAN-compliant RL validation.

6.4.5 Next Steps

The next steps toward full integration include:

- Development of an interface between the RLOps pipeline and KPM/RC xApps
- Deployment of RL inference as an external service connected to FlexRIC
- Validation of end-to-end closed-loop control using:
- Handover management
- Resource allocation policies
- Synchronisation with Digital Twin execution using pacing mode
- Evaluation of scalability under large-scale DT scenarios

7 Ethics Requirements Mapping

Based on the ethics and legal high-level requirements (EL-HLRs) elicited in deliverable D2.4[114] , this section presents the mapping with the specification of the 6G-DALI On-Demand Dataset Generation (ODDG), Automated Model Tuning Across the ML Lifecycle (AMT), Distributed Training and Workload Execution on 6G Testbeds (DTWE), and Digital Twin Testbed for Large-Scale Experimentation (DTT).

This mapping aims to provide initial evidences of the feasibility of the EL-HLRs and their embedment into the 6G-DALI development process. For each EL-HLR, the mapping analysis provides links to the functional (FR) and non-functional (NFR) requirements from the D2.2[115] , the relevant sections of this document where the EL-HLR is considered and the rationale and justification for its implementation.

8 Conclusions and Next Steps

This deliverable covered the initial design and implementation of four technical areas forming the WP4 contributions to the 6G-DALI framework: on-demand dataset generation, automated model tuning across the ML lifecycle, distributed training and workload execution on 6G testbeds, and the digital twin testbed for large-scale experimentation.

For on-demand dataset generation, the intent-driven and surrogate-based pipelines have been presented with initial evaluation results, demonstrating that guided sampling consistently outperforms random selection across all evaluated datasets.

For automated model tuning, the adaptive hyperparameter optimisation framework and the automated reinforcement learning pipeline have been described, together with the cooperative transfer learning design and initial results.

For distributed training and workload execution, the algebraic CEC emulation framework and the ARPA placement agent have been presented, with ARPA demonstrating meaningful improvements in inference energy efficiency relative to baseline scheduling strategies.

For federated learning, the ProposedTopK algorithm achieves strong communication overhead reduction with near-zero Byzantine gradient leakage and competitive model utility on the KUL localization dataset; the split learning framework further reduces data transmission while maintaining strong model accuracy.

For the digital twin testbed, the AI-RSG and FlexRIC integration, the large-scale data generation component, and the RLOps pipeline with digital twin coupling have been described with initial integration results.

Taken together, these contributions advance the energy, communication, and data efficiency dimensions of the 6G-DALI AI/ML pipeline, from automated dataset acquisition and surrogate-guided experiment design, through optimized model training and hyperparameter search, to energy-efficient workload placement and communication-efficient federated and split learning.

Several contributions remain ongoing at the time of this report:

- Testbed adapter implementations across contributing partner sites
- Full closed-loop RLOps validation with ZenML and RLlib connected to the AI-RSG environment
- FL algorithm evaluation against the 6G-DALI dataset collection
- Full implementation of the cooperative transfer learning pipeline, including the partition-and-merge execution path
- ARPA live-system validation on the testbed
- First NWDAF FLOps integration experiments under the Release 18 architecture

Completion of these items will establish the integrated, closed-loop AI/ML operations capability that 6G-DALI requires across itstestbed environments.

References

[1] Deliverable D2.3 Initial framework architecture: https://6gdali.eu/wp-content/uploads/2026/04/6G-DALI_D2.3_Initial-Framework-Architecture_v1.0.pdf

[2] 6G-DALI " Initial report on data management, MLOps, RLOps and Trustworthy AI" 6G-DALI Consortium, Deliverable 3.1, May 2026 [online]

[3] Sharara, M., Murphy, C., Yamany, S., & Pozuelo, A.. “On Leveraging Digital Twins for Training and Validation of Reinforcement Learning in 6G” in 2026 IEEE International Conference on Communications Workshops (ICC Workshops).

[4] Upadhyaya, P. S., Abdalla, A. S., Marojevic, V., Reed, J. H., & Shah, V. K.. “Prototyping Next-Generation O-RAN Research Testbeds with SDRs”. arXiv. https://doi.org/10.48550/ARXIV.2205.13178, 2022.

[5] A. R. Bambhaniya, H. Wu, S. Subramanian, S. Srinivasan, S. Kundu, A. Yazdanbakhsh, M. Elavazhagan, M. Kumar, and T. Krishna, “Understanding and optimizing multi-stage AI inference pipelines,” arXiv preprint arXiv:2504.09775, 2025.

[6] Y. Feng, X. Tan, K. H. Sew, Y. Jiang, Y. Zhu, and H. Xu, “Frontier: Simulating the next generation of LLM inference systems,” in Proc. 4th Workshop Practical Adoption Challenges Mach. Learn. Syst. (PACMI), 2025, pp. 25–30, doi: 10.1145/3766882.3767173.

[7] A. Agrawal, N. Kedia, J. Mohan, A. Panwar, N. Kwatra, B. Gulavani, R. Ramjee, and A. Tumanov, “Vidur: A large-scale simulation framework for LLM inference,” in Proc. Mach. Learn. Syst., vol. 6, pp. 351–366, 2024.

[8] M. Łazuka, A. Anghel, and T. Parnell, “LLM-Pilot: Characterize and optimize performance of your LLM inference services,” in Proc. Int. Conf. High Performance Computing, Networking, Storage and Analysis (SC), 2024, pp. 1–18, doi: 10.1109/SC41406.2024.00022.

[9] X. Miao, C. Shi, J. Duan, X. Xi, D. Lin, B. Cui, and Z. Jia, “SpotServe: Serving generative large language models on preemptible instances,” in Proc. 29th ACM Int. Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS), vol. 2, 2024, pp. 1112–1127, doi: 10.1145/3620665.3640411.

[10] J. Stojkovic, C. Zhang, Í. Goiri, J. Torrellas, and E. Choukse, “DynamoLLM: Designing LLM inference clusters for performance and energy efficiency,” in Proc. IEEE Int. Symp. High-Performance Computer Architecture (HPCA), 2025, pp. 1348–1362, doi: 10.1109/HPCA61900.2025.00102.

[11] L. Wang, L. Yang, Y. Yu, W. Wang, B. Li, X. Sun, J. He, and L. Zhang, “Morphling: Fast, near-optimal auto-configuration for cloud-native model serving,” in Proc. ACM Symp. Cloud Computing (SoCC), 2021, pp. 639–653, doi: 10.1145/3472883.3486987.

[12] T. Xu et al., “AIConfigurator: Lightning-fast configuration optimization for multi-framework LLM serving,” arXiv preprint arXiv:2601.06288, 2026.

[13] J. Alves Pereira, M. Acher, H. Martin, and J.-M. Jézéquel, “Sampling effect on performance prediction of configurable systems: A case study,” in Proc. ACM/SPEC Int. Conf. Performance Engineering (ICPE), 2020, pp. 277–288, doi: 10.1145/3358960.3379137.

[14] M. E. Jam, E. Petit, P. de Oliveira Castro, D. Defour, G. Henry, and W. Jalby, “MLKAPS: Machine learning and adaptive sampling for HPC kernel auto-tuning,” ACM Trans. Archit. Code Optim., vol. 22, no. 4, Art. no. 154, pp. 1–22, 2025, doi: 10.1145/3774418.

[15] W.-L. Loh, “On Latin hypercube sampling,” Ann. Statist., vol. 24, no. 5, pp. 2058–2080, 1996.

[16] M. Stein, “Large sample properties of simulations using Latin hypercube sampling,” Technometrics, vol. 29, no. 2, pp. 143–151, 1987, doi: 10.2307/1269769.

[17] M. E. Johnson, L. M. Moore, and D. Ylvisaker, “Minimax and maximin distance designs,” J. Statist. Planning Inference, vol. 26, no. 2, pp. 131–148, 1990, doi: 10.1016/0378-3758(90)90122-B.

[18] H. Niederreiter, Random Number Generation and Quasi-Monte Carlo Methods. Philadelphia, PA, USA: SIAM, 1992.

[19] I. M. Sobol', "On the distribution of points in a cube and the approximate evaluation of integrals," USSR Comput. Math. Math. Phys., vol. 7, no. 4, pp. 86–112, 1967, doi: 10.1016/0041-5553(67)90144-9.

[20] R. B. Gramacy, *Surrogates: Gaussian Process Modeling, Design, and Optimization for the Applied Sciences*. Boca Raton, FL, USA: Chapman and Hall/CRC, 2020, doi: 10.1201/9780367815493.

[21] B. Iooss, L. Boussouf, V. Feuillard, and A. Marrel, "Numerical studies of the metamodel fitting and validation processes," *Int. J. Advances in Systems and Measurements*, vol. 3, nos. 1–2, pp. 11–21, 2010.

[22] A. I. Cowen-Rivers *et al.*, "HEBO: Pushing the limits of sample-efficient hyper-parameter optimisation," *J. Artif. Intell. Res.*, vol. 74, pp. 1269–1349, 2022, doi: 10.1613/jair.1.13643.

[23] E. C. Garrido-Merchán and D. Hernández-Lobato, "Dealing with categorical and integer-valued variables in Bayesian optimization with Gaussian processes," *Neurocomputing*, vol. 380, pp. 20–35, 2020, doi: 10.1016/j.neucom.2019.11.004.

[24] F. Häse, M. Aldeghi, R. J. Hickman, L. M. Roch, and A. Aspuru-Guzik, "Gryffin: An algorithm for Bayesian optimization of categorical variables informed by expert knowledge," *Appl. Phys. Rev.*, vol. 8, no. 3, Art. no. 031406, 2021, doi: 10.1063/5.0048164.

[25] C. Oh, J. M. Tomczak, E. Gavves, and M. Welling, "Combinatorial Bayesian optimization using the graph Cartesian product," in *Advances in Neural Information Processing Systems*, vol. 32, 2019, pp. 2910–2920.

[26] L. Pronzato and W. G. Müller, "Design of computer experiments: Space filling and beyond," *Stat. Comput.*, vol. 22, no. 3, pp. 681–701, 2012, doi: 10.1007/s11222-011-9242-3.

[27] D. A. Cohn, Z. Ghahramani, and M. I. Jordan, "Active learning with statistical models," *J. Artif. Intell. Res.*, vol. 4, pp. 129–145, 1996, doi: 10.1613/jair.295.

[28] M. C. Kennedy and A. O'Hagan, "Bayesian calibration of computer models," *J. Roy. Statist. Soc. Ser. B Statist. Methodol.*, vol. 63, no. 3, pp. 425–464, 2001, doi: 10.1111/1467-9868.00294.

[29] D. J. C. MacKay, "Information-based objective functions for active data selection," *Neural Comput.*, vol. 4, no. 4, pp. 590–604, 1992, doi: 10.1162/neco.1992.4.4.590.

[30] H. S. Seung, M. Opper, and H. Sompolinsky, "Query by committee," in *Proc. 5th Annu. ACM Workshop Comput. Learn. Theory (COLT)*, 1992, pp. 287–294, doi: 10.1145/130385.130417.

[31] V.-L. Nguyen, M. H. Shaker, and E. Hüllermeier, "How to measure uncertainty in uncertainty sampling for active learning," *Mach. Learn.*, vol. 111, no. 1, pp. 89–122, 2022, doi: 10.1007/s10994-021-06003-9.

[32] D. Wu, C.-T. Lin, and J. Huang, "Active learning for regression using greedy sampling," *Inf. Sci.*, vol. 474, pp. 90–105, 2019, doi: 10.1016/j.ins.2018.09.060.

[33] A. Jose, J. P. A. de Mendonça, E. Devijver, N. Jakse, V. Monbet, and R. Poloni, "Regression tree-based active learning," *Data Min. Knowl. Discov.*, vol. 38, no. 2, pp. 420–460, 2024, doi: 10.1007/s10618-023-00951-7.

[34] W. Bae, J. Noh, and D. J. Sutherland, "Generalized coverage for more robust low-budget active learning," in *Computer Vision – ECCV 2024*, Lecture Notes in Computer Science, vol. 15141, Cham, Switzerland: Springer, 2025, pp. 318–334, doi: 10.1007/978-3-031-73010-8_19.

[35] Y. Chen, M. Welling, and A. J. Smola, "Super-samples from kernel herding," arXiv preprint arXiv:1203.3472, 2012.

[36] F. Bach, S. Lacoste-Julien, and G. Obozinski, "On the equivalence between herding and conditional gradient algorithms," arXiv preprint arXiv:1203.4523, 2012.

[37] E. Aydar, C. Pinto, S. Venugopal, and D. Chatzopoulos, "Sampling Where It Matters: Predicting LLM Serving Performance," *Proceedings of the Sixth European Workshop on Machine Learning and Systems*, pp. 13–22, Apr. 2026, doi: https://doi.org/10.1145/3805621.3807633.

[38] Mekki, M., N. Toumi, and A. Ksentini. "Benchmarking on microservices configurations and the impact on the performance in cloud native environments." Zenodo (2022).

[39] S. Wager, T. Hastie, and B. Efron, "Confidence intervals for random forests: The jackknife and the infinitesimal jackknife," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1625–1651, 2014.

[40] A. Pomponio and M. Johnston, "Introducing ADO: The accelerated discovery orchestrator," *IBM Research Blog*, Aug. 29, 2025. [Online]. Available: https://research.ibm.com/blog/ado-accelerated-discovery-orchestrator-experiments

[41] M. A. K. Raiaan et al., "A systematic review of hyperparameter optimization techniques in convolutional neural networks," *Decision Analytics Journal*, vol. 11, p. 100470, 2024.

[42] Ray Documentation. "Ray Tune: Scalable hyperparameter tuning." docs.ray.io. https://docs.ray.io/en/latest/tune/index.html (accessed Apr. 23, 2026).

[43] Optuna. "Optuna: A hyperparameter optimization framework." optuna.org. https://optuna.org/ (accessed Apr. 23, 2026).

[44] Scikit-learn. "sklearn.ensemble.RandomForestClassifier." scikit-learn.org. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html (accessed Apr. 23, 2026).

[45] S. Falkner, A. Klein and F. Hutter, "BOHB: Robust and Efficient Hyperparameter Optimization at Scale," Proceedings of the 35th International Conference on Machine Learning (ICML), Stockholm, Sweden, 2018, pp. 1437–1446.

[46] M. Jaderberg, V. Dalibard, S. Osindero et al., "Population Based Training of Neural Networks," arXiv preprint arXiv:1711.09846, 2017.

[47] N. Awad, N. Mallik, and F. Hutter, "DEHB: Evolutionary Hyperband for Scalable, Robust and Efficient Hyperparameter Optimization," Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI), Montreal, Canada, 2021, pp. 2147–2153, doi: 10.24963/ijcai.2021/296.

[48] F. Hutter, H. H. Hoos and K. Leyton-Brown, "Sequential Model-Based Optimization for General Algorithm Configuration," Proceedings of the 5th International Conference on Learning and Intelligent Optimization (LION), Rome, Italy, 2011, pp. 507–523.

[49] S. K. Chari, J. S. Vardakas, K. Ramantas, A. Ksentini and C. Verikoukis, "Reinforcement Learning Driven Sustainable Resource and Power Management for the MEC," GLOBECOM 2024 - IEEE Global Communications Conference, Cape Town, South Africa, 2024, pp. 565-570, doi: 10.1109/GLOBECOM52923.2024.10901623.

[50] H. Daga, P. K. Nicholson, A. Gavrilovska and D. Lugones, "Cartel: A System for Collaborative Transfer Learning at the Edge," in Proc. ACM Symposium on Cloud Computing (SoCC), 2019, pp. 25–37, doi: 10.1145/3357223.3362708.

[51] H. Daga, Y. Chen, A. Agrawal and A. Gavrilovska, "CLUE: Systems Support for Knowledge Transfer in Collaborative Learning With Neural Nets," IEEE Trans. Cloud Comput., vol. 11, no. 4, pp. 3652–3665, Oct.–Dec. 2023, doi: 10.1109/TCC.2022.3211077.

[52] AI-RAN Alliance, "AI-RAN Alliance Vision and Mission," White Paper, Dec. 2024. [Online]. Available: https://ai-ran.org/documents/AI-RAN-Alliance-Whitepaper.pdf

[53] B. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes, "Borg, Omega and Kubernetes," *ACM Queue*, vol. 14, pp. 70–93, 2016.

[54] J. Schulman et al., "Proximal Policy Optimization Algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[55] A. Raffin et al., "Stable-Baselines3: Reliable Reinforcement Learning Implementations," *J. Mach. Learn. Res.*, vol. 22, no. 268, pp. 1–8, 2021.

[56] 3GPP, "TS 22.261: Service Requirements for the 5G System," Release 18, 2023. [Online]. Available: https://www.3gpp.org/

[57] NVIDIA, "NVIDIA Multi-Instance GPU User Guide," Technical Report, 2024. [Online]. Available: https://docs.nvidia.com/datacenter/tesla/mig-user-guide/

[58] J. Yeung et al., "MIGPerf: A Comprehensive Benchmark for Deep Learning Training and Inference Workloads on Multi-Instance GPUs," *arXiv preprint arXiv:2301.00407*, 2023.

[59] S. Levine et al., "Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems," *arXiv preprint arXiv:2005.01643*, 2020.

[60] S. Hong and H. Kim, "An Integrated GPU Power and Performance Model," in *Proc. ACM/IEEE ISCA*, 2010, pp. 280–289.

[61] Q. Weng et al., "Beware of Fragmentation: Scheduling GPU-Sharing Workloads with Fragmentation Gradient Descent," in Proc. 2023 USENIX Annu. Tech. Conf. (USENIX ATC), Boston, MA, USA, 2023, pp. 995–1008.

[62] NVIDIA, "Triton Inference Server Documentation," 2024. [Online]. Available: https://docs.nvidia.com/deeplearning/triton-inference-server/

[63] 3GPP TS 29.552. 5G System; Network Data Analytics signalling flows; Stage 3 3gpp.org. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3871. [Accessed: 28-Apr-2026].

[64] Mazene Ameur, Bouziane Brik, Adlen Ksentini. When MLOps Meets NWDAF to Enable Autonomous Next Generation Network Analytics. ICMLCN 2026, IEEE International Conference on Machine Learning in Communications and Networking, IEEE, Mar 2026, Abu Dhabi, United Arab Emirates. ⟨hal-05563236⟩

[65] Elbir, A., Papazafeiropoulos, A., & Chatzinotas, S. (2021). Federated Learning for Physical Layer Design. *IEEE Communications Magazine* (pp. 81-87).

[66] Cui, Y., Guo, J., Li, X., Liang, L., & Jin, S. (2022). Federated edge learning for the wireless physical layer: Opportunities and challenges. *China Communications* (pp. 15-30).

[67] Das, S., Champagne, B., Psaromiligkos, I., & Cai, Y. (2024). A Survey on Federated Learning for Reconfigurable Intelligent Metasurfaces-Aided Wireless Networks. *IEEE Open Journal of the Communications Society* (pp. 1846-1879).

[68] Chen, M., Yang, Z., Saad, W., Yin, C., Poor, H., & Cui, S. (2019). A Joint Learning and Communications Framework for Federated Learning Over Wireless Networks. *IEEE Transactions on Wireless Communications* (pp. 269-283).

[69] Dinh, C., Tran, N., Nguyen, M., Hong, C., Bao, W., Zomaya, A., & Gramoli, V. (2019). Federated Learning Over Wireless Networks: Convergence Analysis and Resource Allocation. *IEEE/ACM Transactions on Networking* (pp. 398-409).

[70] Fan, X., Wang, Y., Huo, Y., & Tian, Z. (2021). Joint Optimization of Communications and Federated Learning Over the Air. *IEEE Transactions on Wireless Communications*.

[71] González-Prelcic, N., Keskin, M., Kaltiokallio, O., Valkama, M., Dardari, D., Shen, X., Shen, Y., Bayraktar, M., & Wymeersch, H. (2024). The Integrated Sensing and Communication Revolution for 6G: Vision, Techniques, and Applications. *Proceedings of the IEEE* (pp. 676-723).

[72] Du, M., Zheng, H., Gao, M., Feng, X., Hu, J., & Chen, Y. (2024). Integrated Sensing, Communication, and Computation for Over-the-Air Federated Learning in 6G Wireless Networks. *IEEE Internet of Things Journal* (pp. 35551-35567).

[73] Hu, S., Yuan, X., Ni, W., Wang, X., Hossain, E., & Poor, V. (2025). Differentially Private Wireless Federated Learning With Integrated Sensing and Communication. *IEEE Transactions on Wireless Communications* (pp. 6690-6704).

[74] Pala, S., Singh, K., Li, C., & Dobre, O. (2025). Empowering ISAC Systems With Federated Learning: A Focus on Satellite and RIS-Enhanced Terrestrial Integrated Networks. *IEEE Transactions on Wireless Communications* (pp. 810-824).

[75] Adhikary, A., Ejigu, G., Kim, K., Suh, M., & Hong, C. (2025). FedISAC: A Federated Learning Framework with Integrated Sensing and Communication for 6G mmWave Networks. *International Conference on Information Networking (ICOIN)* (pp. 45-50).

[76] Liu, P., Zhu, G., Jiang, W., Luo, W., Xu, J., & Cui, S. (2022). Vertical Federated Edge Learning With Distributed Integrated Sensing and Communication. *IEEE Communications Letters* (pp. 2091-2095).

[77] Ni, Z., Chintareddy, S., Guan, P., & Hashemi, M. (2025). Personalized Federated Learning-Driven Beamforming Optimization for Integrated Sensing and Communication Systems. *IEEE Consumer Communications & Networking Conference (CCNC)*.

[78] Nagia, N., Tahsin Rahman, M., & Valaee, S. (2022). Federated Learning for WiFi Fingerprinting. *IEEE International Conference on Communications (ICC)*.

[79] De Bast, S., & Pollin, S. (2021). Ultra Dense Indoor MaMIMO CSI Dataset. *IEEE Dataport*.

[80] De Bast, S., Vinogradov, E., & Pollin, S. (2022). Expert-Knowledge-Based Data-Driven Approach for Distributed Localization in Cell-Free Massive MIMO Networks. *IEEE Access* (pp. 56427-56439).

[81] Nguyen, Thien Duc, et al. "{FLAME}: Taming backdoors in federated learning." 31st USENIX security symposium (USENIX Security 22). 2022.

[82] Rammal, A., Gruntkowska, K., Fedin, N., Gorbunov, E., & Richtárik, P. (2024, April). Communication compression for byzantine robust learning: New efficient algorithms and improved rates. In International Conference on Artificial Intelligence and Statistics (pp. 1207-1215). PMLR.

[83] Gorbunov, Eduard, et al. "Variance reduction is an antidote to byzantines: Better rates, weaker assumptions and communication compression as a cherry on the top." *arXiv preprint arXiv:2206.00529* (2022).

[84] Panda, A., Mahloujifar, S., Bhagoji, A. N., Chakraborty, S., & Mittal, P. (2022, May). Sparsefed: Mitigating model poisoning attacks in federated learning with sparsification. In International Conference on Artificial Intelligence and Statistics (pp. 7587-7624). PMLR.

[85] Zhang, Z., & Hu, R. (2023, October). Byzantine-robust federated learning with variance reduction and differential privacy. In 2023 IEEE Conference on Communications and Network Security (CNS) (pp. 1-9). IEEE.

[86] Xia, Y., Jahani-Nezhad, T., & Bitar, R. (2025). Beyond Trade-offs: A Unified Framework for Privacy, Robustness, and Communication Efficiency in Federated Learning. arXiv preprint arXiv:2508.12978.

[87] Xu, S., Zheng, Y., & Hua, Z. (2025, November). Harnessing Sparsification in Federated Learning: A Secure, Efficient, and Differentially Private Realization. In Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security (pp. 2354-2368).

[88] Richtárik, Peter, Igor Sokolov, and Ilyas Fatkhullin. "EF21: A new, simpler, theoretically better, and practically faster error feedback." *Advances in Neural Information Processing Systems* 34 (2021): 4384-4396.

[89] P. Vepakomma, O. Gupta, T. Swedish and R. Raskar, "Split learning for health: Distributed deep learning without sharing raw patient data," arXiv preprint arXiv:1812.00564, 2018.

[90] C. Thapa, M. A. P. Chamikara, S. Camtepe and L. Sun, "SplitFed: When Federated Learning Meets Split Learning," arXiv preprint arXiv:2004.12088, 2020.

[91] Z. Lin, G. Qu, X. Chen and K. Huang, "Split Learning in 6G Edge Networks," arXiv preprint arXiv:2306.12194, 2023.

[92] N. D. Pham and N. Chilamkurti, "Data Leakage Threats and Protection in Split Learning: A Survey," Proceedings of the 2023 International Conference on Intelligent Computing and Its Emerging Applications (ICEA), 2023, pp. 141–147, doi: 10.1145/3659154.3659189.

[93] W. Wu, M. Li, K. Qu, C. Zhou, X. Shen, W. Zhuang, X. Li and W. Shi, "Split Learning over Wireless Networks: Parallel Design and Resource Management," arXiv preprint arXiv:2204.08119, 2022.

[94] H. Xu, C.-Y. Ho, A. M. Abdelmoniem, A. Dutta, E. H. Bergou, K. Karatsenidis, M. Canini and P. Kalnis, "Compressed Communication for Distributed Deep Learning: Survey and Quantitative Evaluation," Technical Report, KAUST, 2020.

[95] Y. Oh, J. Lee, C. G. Brinton and Y.-S. Jeon, "Communication-Efficient Split Learning via Adaptive Feature-Wise Compression," arXiv preprint arXiv:2307.10805, 2023.

[96] A. Mudvari, A. Vainio, I. Ofeidis, S. Tarkoma and L. Tassiulas, "Adaptive Compression-Aware Split Learning and Inference for Enhanced Network Efficiency," arXiv preprint arXiv:2311.05739, 2023.

[97] Z. Hu, T. Zhou, B. Wu, C. Chen and Y. Wang, "SLPerf: A Research Library and Benchmark Framework for Split Learning," in Proceedings of the 2025 IEEE 41st International Conference on Data Engineering Workshops (ICDEW), 2025, pp. 33–36, doi: 10.1109/ICDEW67478.2025.00009.

[98] L. U. Khan, W. Saad, D. Niyato, Z. Han, and C. S. Hong, "Digital-twin-enabled 6G: Vision, architectural trends, and future directions," IEEE Commun. Mag., vol. 60, no. 1, pp. 74–80, 2022.

[99] X. Lin, L. Kundu, C. Dick, E. Obiodu, T. Mostak, and M. Flaxman, "6G digital twin networks: From theory to practice," IEEE Commun. Mag., vol. 61, no. 11, pp. 72–78, Jun. 2023.

[100] Z. Tao, W. Xu, Y. Huang, X. Wang, and X. You, "Wireless network digital twin for 6G: Generative AI as a key enabler," IEEE Wireless Commun., vol. 31, no. 4, pp. 24–31, 2024.

[101] Jaeckel et al., "QuaDRiGa: A 3-D Multi-Cell Channel Model..." IEEE Trans. Antennas Propag., 2014

[102] Patriciello et al., "An E2E simulator for 5G NR networks," Simulation Modelling Practice, 2019

[103] Hoydis et al., "Sionna: An Open-Source Library for Next-Generation Physical Layer Research," arXiv 2022

[104] Polese et al., "ns-O-RAN: Simulating O-RAN with ns-3," WNS3 2023

[105] R. Schmidt, M. Irazabal, and N. Nikaein, "FlexRIC: An SDK for next-generation SD-RANs," in Proc. 17th Int. Conf. Emerging Networking EXperiments and Technologies (CoNEXT), 2021, pp. 411–425.

[106] 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," Computer Networks, vol. 220, 2023.

[107] 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., 2022.

[108] J. Liu, C. Chai, Y. Luo, Y. Lou, J. Feng and N. Tang, "Feature Augmentation with Reinforcement Learning," 2022 IEEE 38th International Conference on Data Engineering (ICDE), Kuala Lumpur, Malaysia, 2022, pp. 3360-3372, doi: 10.1109/ICDE53745.2022.00317.

[109] A. Ionescu, K. Vasilev, F. Buse, R. Hai and A. Katsifodimos, "AutoFeat: Transitive Feature Discovery over Join Paths," 2024 IEEE 40th International Conference on Data Engineering (ICDE), Utrecht, Netherlands, 2024, pp. 1861-1873, doi: 10.1109/ICDE60146.2024.00150.G

[110] G. Famitafreshi, M. Trigka, D. Selis, J. Vardakas, and C. Verikoukis, “An innovative multi-scale strategy-based decision engine for zero-touch management and orchestration in 6g,” in 2024 IEEE 29th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), 2024, pp. 1–6.S.

[111] Marinova and A. Leon-Garcia, “Intelligent o-ran beyond 5g: Architecture, use cases, challenges, and opportunities,” IEEE Access, vol. 12, pp. 27 088–27 114, 2024.

[112] P. Li, J. Thomas, X. Wang, A. Khalil, A. Ahmad, R. Inacio, S. Kapoor, A. Parekh, A. Doufexi, A. Shojaeifard, and R. J. Piechocki, "RLOps: Development life-cycle of reinforcement learning aided open RAN," IEEE Access, vol. 10, pp. 113808–113826, 2022.

[113] M. Tsampazi, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "PandORA: Automated design and comprehensive evaluation of deep reinforcement learning agents for open RAN," arXiv preprint arXiv:2407.11747, 2024.

[114] 6G-DALI "6G-DALI compliance with data legislation and ethical requirements - Initial," 6G-DALI Consortium , Deliverable 2.4, Feb. 2026.

[115] 6G-DALI "Framework requirements and technological tools review" 6G-DALI Consortium , Deliverable 2.2 , Oct. 2025.

Annex I Ethics Requirements Mapping

I-A Privacy and Data Governance

This mapping bridges the high-level Privacy and Data Governance requirement (EL-HLR-PR) and provides details on how ethics and legal obligations regarding data collection, use, and sharing are embedded directly into the system's design through an "ethics-by-conception" approach. Specifically, this mapping identifies the Federated Learning Operations as a key part of the 6G-DALI framework to address EL-HLR-PR and its mid-level requirements:

- EL-MLR-PR.1 “Framework Information” (linked to FR-DS 4, FR-DS 7, FR-DS 14, FR-MO-37) mandates explicit information about data practices and policies;
- EL-MLR-PR.2 “Assessment Records” (linked to FR-MLOPS-4, NFR-DS 7, FR-DATAOPS-8, NFR-MO-1, NFR-DATAOPS-2) mandates the documentation of events and decisions;

EL-HLR-PR: Privacy and Data Governance	
Related FR/NFR	• *FR-DS 4, FR-DS 7, FR-DS 14, NFR-DS 3, NFR-DS 7* • *FR-DATAOPS-8, NFR-DATAOPS-2* • *FR-MO-37, NFR-MO-1, NFR-MO-2*
Relevant Section(s)	**Rationale**
Federated Learning Operations	*The Federated Learning Operations addresses* • *EL-MLR-PR.1 by providing detailed information about the implemented Federated Learning algorithms, containing what information about local data / models are shared with the central aggregation server and in which way, and how data leakage is prevented.* • *EL-MLR-PR.2 by documenting the federated learning algorithm design, every experiment setup, configuration, reproducible results, validations, and insights, so that the records assess events and decisions.*

I-B Accountability

This mapping bridges the high-level Accountability requirement (EL-HLR-AC) and provides details on how ethics and legal obligations regarding the traceability of system configurations, autonomous behaviors, and lifecycle events are embedded directly into the system's design through an "ethics-by-conception" approach. Specifically, this mapping identifies the Cooperative Transfer Learning as key part of the 6G-DALI framework to address EL-HLR-AC and its mid-level requirements:

- EL-MLR-AC.1 “Accountability Mechanism” (linked to FR-MLOPS-4, NFR-DS 15) mandates the design of accountability mechanisms at the system or operation level, triggered by specific events;
- EL-MLR-AC.2 “Attribution and Responsibility” (linked to NFR-DS 7, FR-DS 10, NFR-MLOPS-1) requires that actions, decisions, and system behaviors can be attributed to identifiable accountable actors, roles, or processes;
- EL-MLR-AC.3 “Policy Enforcement Traceability” (linked to FR-DS 15, NFR-DS 7, FR-MO-15, NFR-MO-2) requires the traceability of policy enforcement actions (grant, deny, constrain) to enable auditing;
- EL-MLR-AC.4 “Accountability Reports” (linked to FR-MO-35, NFR-MO-6, NFR-EH-1) mandates the generation of reports for significant events such as model lifecycle actions, policy changes, and system-wide operational shifts.

EL-MLR-AC “Accountability”	
Related FR/NFR	• *FR-MLOPS-4, NFR-MLOPS-1* • *FR-DS 10, FR-DS 15, NFR-DS 7, NFR-DS 15* • *FR-MO-15, FR-MO-35, NFR-MO-2, NFR-MO-6* • *NFR-EH-1*

Relevant Section(s)	Rationale
Cooperative Transfer Learning	*Cooperative Transfer Learning addresses* • *EL-MLR-AC.1 by implementing orchestrated and fully traceable CoTL pipelines within the MLOps framework, where all model registration, knowledge extraction, transfer and evaluation actions are automatically logged, versioned, and reproducible through ZenML and Mlflow-based lifecycle management.* • *EL-MLR-AC.2 by ensuring that all model onboarding, knowledge contribution, and model improvement activities are associated with identifiable users, experiments, and source models through authenticated access control and metadata tracking within the MLOps framework.* • *EL-MLR-AC.3 by enforcing policy-driven access and sharing constraints on models and reusable knowledge units.* • *EL-MLR-AC.4 by generating execution records and evaluation outputs for all significant CoTL lifecycle events including model registration, knowledge transfer, model reconstruction and performance assessment.*

I-C Reliability, Robustness, Resilience

This mapping bridges the high-level Reliability, Robustness, and Resilience requirement (EL-HLR-RR) and provides details on how ethics and legal obligations regarding predictable system behaviour under stress, failure observability, and graceful degradation are embedded directly into the system's design through an "ethics-by-conception" approach. Specifically, this mapping identifies the ADO for HPO in MLOps, and the RLOps and DT as key parts of the 6G-DALI framework to address EL-HLR-RR and its mid-level requirements:

- EL-MLR-RR.1 “Identification of Failure Modes and Adversarial Conditions” (linked to FR-MO-16, FR-TAI-5, FR-ELT-7) mandates the identification of failure modes such as behavioral drift, instability, and uncertainty;
- EL-MLR-RR.2 “Monitor and Prevent” (linked to FR-TAI-6, NFR-MO-7, FR-MO-28) requires continuous monitoring to identify degradations and signal uncertainty before failures occur;
- EL-MLR-RR.3 “Mitigation and Recovery Mechanisms” (linked to FR-MO-22, FR-MO-23, FR-MO-18, NFR-DS 4) mandates support for mitigation plans to restore acceptable behaviour after disruption.

EL-MLR-RR “Reliability, Robustness, Resilience”	
Related FR/NFR	• *FR-MO-16, FR-MO-18, FR-MO-22, FR-MO-23, FR-MO-28, NFR-MO-7* • *FR-ELT-7* • *FR-TAI-5, FR-TAI-6,* • *NFR-DS 4*
Relevant Section(s)	Rationale
ADO for HPO in MLOps	*ADO for HPO in MLOps addresses* • *EL-MLR-RR.1 by exposing model instability and high-variance behaviors through rigorous cross-validation across diverse hyperparameter configurations.* • *EL-MLR-RR.3 by enabling automated re-tuning pipelines to rapidly discover stable model configurations and recover performance when behavioral drift is detected.*
RLOps and DT	*RLOps and DT addresses* • *EL-MLR-RR.1 by enabling controlled and reproducible experimentation of RL-based strategies within Digital*

	Twin environments before deployment into operational infrastructures. The integration of RL agents with simulated O-RAN environments allows the identification of unstable behaviours, policy degradation, unsafe control actions, and performance limitations under varying network conditions (e.g. large-scale UE scenarios and dynamic traffic patterns). The use of DT pacing mode further supports deterministic evaluation of sequential decision-making processes and facilitates analysis of failure modes and behavioral drift. • *EL-MLR-RR.2 by supporting continuous monitoring and tracking of RL training, inference, and evaluation workflows through the RLOps framework. This is achieved using MLflow and ZenML orchestration pipelines, which enable early detection of performance degradation, instability, or abnormal agent behavior during experimentation and validation phases.* • *EL-MLR-RR.3 by providing mechanisms for iterative retraining, reproducible policy validation, and controlled rollback of RL models within the Digital Twin environment before operational deployment. This strategy enables mitigation and recovery procedures and reduces the risk of propagating unstable or degraded RL policies into live O-RAN control loops.*

I-D Autonomy

This mapping bridges the high-level Autonomy requirement (EL-HLR-AU) and provides details on how ethics and legal obligations regarding the governability, observability, and bounded nature of autonomous decision-making are embedded directly into the system's design through an "ethics-by-conception" approach. Specifically, this mapping identifies the RLOps and DT as key part of the 6G-DALI framework to address EL-HLR-AU and its mid-level requirements:

- EL-MLR-AU.1 “Autonomy Related Risks Identification” (linked to FR-TAI-5, FR-MO-21) mandates the ex-ante definition of autonomy-related risk patterns (such as escalation or boundary exploitation) and the prevention of implicit drift driven solely by optimisation incentives;
- EL-MLR-AU.2 “Autonomy-Related Risks Monitoring and Governance” (linked to FR-AR-2, NFR-MO-8) requires the monitoring of aggregate indicators reflecting autonomy-specific properties (such as stability, boundary proximity, and reversibility) rather than just interpreting individual micro-decisions;
- EL-MLR-AU.3 “Autonomy-Related Safeguard Mechanisms” (linked to FR-AR-4, FR-MO-22, FR-MO-30, FR-IDD-3) mandates intervention capabilities at the system or policy level where autonomy-related patterns threaten ethical and regulatory principles (e.g., loss of control or reversibility).

EL-MLR-AU “Autonomy”	
Related FR/NFR	• *FR-TAI-5* • *FR-AR-2, FR-AR-4* • *FR-MO-21, FR-MO-22, FR-MO-30, NFR-MO-8* • *FR-IDD-3*
Relevant Section(s)	**Rationale**
RLOps and DT	*RLOps and DT addresses* • *EL-MLR-AU.1 by enabling controlled experimentation and validation of RL-based network control policies within Digital Twin environments before deployment into*

	operational infrastructures. The framework supports the identification of risks such as unstable policy behaviour, unintended optimisation effects, and behavioural drift under varying network conditions through reproducible experimentation and scenario-based validation workflows. • *EL-MLR-AU.2 by providing monitoring and observability mechanisms throughout the RL lifecycle, including training, validation, and operational phases. The integration of ZenML orchestration, MLflow tracking, and Digital Twin execution enables continuous observation of policy performance, stability trends, and behavioural changes across experiments and deployments, supporting the detection of degraded or unstable policies before operational impact occurs.* • *EL-MLR-AU.3 by supporting safeguard and intervention mechanisms through controlled validation and deployment workflows. RL policies can be paused, retrained, revalidated, rolled back, or replaced before deployment into operational environments. The Digital Twin environment further enables safe testing of policy updates and control strategies under reproducible conditions while preserving operator oversight and control over the deployment process.*

I-E Transparency

This mapping bridges the high-level Transparency requirement (EL-HLR-TR) and provides details on how ethics and legal obligations regarding the visibility, intelligibility, and honest reporting of AI-driven behaviours are embedded directly into the system's design through an "ethics-by-conception" approach. Specifically, this mapping identifies the the Cooperative Transfer Learning and Federated Learning Operations as key parts of the 6G-DALI framework to address EL-HLR-TR and its mid-level requirements:

- EL-MLR-TR.1 "Appropriate Disclosure" (linked to FR-MO-27, NFR-MLOPS-1, FR-DS 9, FR-MO-15, NFR-MO-2) mandates the disclosure of system information and AI-driven behaviour to authorised stakeholders while respecting confidentiality and limiting personal data exposure;
- EL-MLR-TR.2 "Operational Information Specification" (linked to FR-MLOPS-4, FR-DS 14, NFR-MO-3) requires the disclosure of intermediate outputs, assumptions, training datasets metadata, lineage, and operational design domains;
- EL-MLR-TR.3 "Non-deceptive Transparency" (linked to NFR-MO-6, FR-TAI-5, NFR-MLOPS-4, NFR-DATAOPS-6) demands that system failures and performance degradations are explicitly observable and never masked by optimization techniques.

EL-MLR-TR "Transparency"	
Related FR/NFR:	• *FR-MO-15, FR-MO-27, NFR-MO-2, NFR-MO-3, NFR-MO-6* • *NFR-MLOPS-1, FR-MLOPS-4, NFR-MLOPS-4* • *FR-TAI-5,* • *NFR-DATAOPS-6* • *FR-DS 9, FR-DS 14*
Relevant Section(s)	**Rationale**
Cooperative Transfer Learning	*Cooperative Transfer Learning addresses* • *EL-MLR-TR.1 by providing authorized stakeholders with controlled visibility into model registration, knowledge sharing, and model improvement processes through the MLOps user interface, while enforcing role-based access*

	control and configurable sharing policies to preserve confidentiality and data locality. • *EL-MLR-TR.2 by maintaining detailed metadata, lineage information, evaluation metrics, and provenance records for both models and reusable knowledge units within model and knowledge catalogues, enabling transparent inspection of training assumptions, operational context, and knowledge transfer activities throughout the CoTL lifecycle.* • *EL-MLR-TR.3 by ensuring that all model improvement operations, evaluation outcomes, pipeline failures, and performance degradations are explicitly logged, observable, and communicated through the orchestrated MLOps workflows, preventing hidden optimisation behaviour and supporting trustworthy operation in distributed environments.*
Federated Learning Operations	*Federated Learning Operations addresses* • *EL-MLR-TR.3 by logging all the operation events during the training process, evaluation results, and any system failures or algorithms errors. @EUR may add more details*

I-F Fairness

This mapping bridges the high-level Fairness requirement (EL-HLR-FA) and provides details on how ethics and legal obligations regarding the prevention of unjustified differential effects and the governance of intentional differentiation are embedded directly into the system's design through an "ethics-by-conception" approach. Specifically, this mapping identifies the Surrogate Based Data Generation, the AI/ML Workload Placement, and the Federated Learning Operations as key parts of the 6G-DALI framework to address EL-HLR-FA and its mid-level requirements:

- EL-MLR-FA.1 “Intentional Differentiation” (linked to FR-DS 10, FR-MO-7) mandates explicit documentation of where and why AI-driven behaviour differentiates between predefined classes (e.g., service classes, contexts, device capability tiers);
- EL-MLR-FA.2 “Monitoring differentiation indicators” (linked to FR-MO-28, FR-MO-19) requires the monitoring of aggregate and structural indicators that may signal unjustified differential effects across predefined, non-sensitive classes;
- EL-MLR-FA.3 “Mitigation of differentiation” (linked to FR-MO-22, FR-MO-23) demands support for governance processes through which disparities exceeding predefined thresholds trigger review of underlying policies, configurations, or optimisation strategies;

EL-MLR-FA “Fairness”	
Related FR/NFR	• *FR-DS 10* • *FR-MO-7, FR-MO-19, FR-MO-22, FR-MO-23, FR-MO-28*
Relevant Section(s)	Rationale
Surrogate Based Data Generation	*Surrogate Based Data Generation addresses* • *EL-MLR-FA.1 by explicitly documenting the candidate parameter space, KPI targets, measurement budget, and PKH-based selection logic, so any intentional differentiation across workload or infrastructure classes is transparent and justified.* • *EL-MLR-FA.2 by monitoring how well the measured subset represents the full configuration space, using PKH to identify under-covered regions in the surrogate’s learned feature space that may indicate structural imbalance or unjustified differential effects.*

	• *EL-MLR-FA.3 by enabling corrective action when disparities are detected, since high surrogate validation error or poor coverage in specific regions can trigger additional measurements, re-sampling, or review of the acquisition strategy within the MLOps loop.*

I-G Societal & Environmental Well-Being

This mapping bridges the high-level Societal & Environmental Well-Being requirement (EL-HLR-SEW) and provides details on how ethics and legal obligations regarding the identification, monitoring, and governance of systemic societal risks and environmental impacts (e.g., energy consumption) are embedded directly into the system's design through an "ethics-by-conception" approach. Specifically, this mapping identifies the ADO for HPO in MLOps, and the Automated Reinforcement Learning and RLOps as key parts of the 6G-DALI framework to address EL-HLR-FA and its mid-level requirements:

- EL-MLR-SEW.1 "Ethics and Societal Risk Identification" (linked to FR-TAI-6, FR-MO-17) mandates that societal risk factors relevant to the operation of AI-native 6G infrastructures shall be identified and defined ex ante as part of the project's governance framework
- EL-MLR-SEW.2 "Ethic and Societal Risk Monitoring and Governance" (linked to NFR-EH-3, NFR-TAI-1) requires the monitoring of system operation against predefined societal risk factors, such as energy consumption and resource intensity, using aggregate and longitudinal indicators;
- EL-MLR-SEW.3 "Human Autonomy" (linked to FR-MO-30, FR-MO-27) demands the identification, monitoring, and governance of structural conditions through which AI-native 6G operation shapes, constrains, or pre-configures the space of human choices, ensuring meaningful human autonomy is not eroded;

EL-MLR-FA "Societal & Environmental Well-Being"	
Related FR/NFR	• *FR-TAI-6, NFR-TAI-1* • *FR-MO-17, FR-MO-27, FR-MO-30* • *NFR-EH-3*
Relevant Section(s)	Rationale
ADO for HPO in MLOps	*ADO for HPO in MLOps addresses* • *EL-MLR-SEW.2 by systematically tracking compute usage, training time, and energy consumption across all automated trial evaluations.* • *EL-MLR-SEW.3 by maintaining human-in-the-loop governance over the definition of search boundaries, optimization objectives, and final model parameters selection.*
Automated Reinforcement Learning and RLOps	*Automated Reinforcement Learning and RLOps addresses* • *EL-MLR-SEW.1 by incorporating predefined robustness objectives and energy-related constraints into the AutoRL workflow, enabling the identification and evaluation of societal and operational risks associated with autonomous resource management.* • *EL-MLR-SEW.2 by continuously monitoring optimization performance, computational resource usage, and training efficiency across HPO and RL workflows, thereby supporting governance of resource intensity and computational sustainability.* • *EL-MLR-SEW.3 by maintaining human oversight over the configuration, monitoring and evaluation of AutoRL experiments through the MLOps User Interface, allowing*

	users to define objectives, constraints and robustness metrics.

I-H Epistemic Integrity

This mapping bridges the high-level Epistemic Integrity requirement (EL-HLR-EI) and provides details on how ethics and legal obligations regarding the prevention of validation mismatches, misapplied transfer, and epistemic risks in simulation and synthetic data generation are embedded directly into the system's design through an "ethics-by-conception" approach. Specifically, this mapping identifies the ADO for HPO in MLOps, the Automated Reinforcement Learning and RLOps, the AI/ML Workload Placement, the AI-RSG and FlexRIC Integration and the Large-Scale Data Generation as key parts of the 6G-DALI framework to address EL-HLR-EI and its mid-level requirements:

- EL-MLR-EI.1 “Epistemic Risk Identification” (linked to FR-DS 4, FR-DT-2, FR-MO-37) mandates the identification and documentation of epistemic risks, including data provenance (synthetic vs. real), validation environments, simulation assumptions, and reward structures;
- EL-MLR-EI.2 “Epistemic Risk Monitoring and Governance” (linked to FR-AR-4, FR-AR-6, FR-MO-20, FR-TAI-6) requires the monitoring of risks during validation, considering out-of-domain (OOD) parameters and explicitly signalling the limits of model validity and generalisation;
- EL-MLR-EI.3 “Epistemic Risk Safeguard Mechanisms” (linked to FR-AR-4, FR-MO-9, FR-MO-13, FR-DA-2, NFR-DA-5) mandates safeguards against validation-induced distortion, such as overfitting to narrow scenarios or the amplification of bias through flawed reward design.

EL-MLR-EI “Epistemic Integrity”	
Related FR/NFR	• *FR-DS 4, FR-DT-2* • *FR-MO-9, FR-MO-13, FR-MO-20, FR-MO-37* • *FR-AR-4, FR-AR-6* • *FR-TAI-6* • *FR-DA-2, NFR-DA-5*
Relevant Section(s)	Rationale
ADO for HPO in MLOps	*ADO for HPO in MLOps addresses* • *EL-MLR-EI.1 by automatically logging all experimental metadata, ensuring strict documentation of data splits, parameters, and evaluation environments per trial.* • *EL-MLR-EI.2 by incorporating out-of-domain (OOD) test sets into the HPO objective function to explicitly evaluate and signal the limits of model generalization.*
Automated Reinforcement Learning and RLOps	*Automated Reinforcement Learning and RLOps addresses* • *EL-MLR-EI.1 by documenting the experimental setup, reward formulations, hyperparameter configurations, validation conditions and training metadata associated with RL optimization workflows, enabling the identification of epistemic risks.* • *EL-MLR-EI.2 by supporting continuous validation and monitoring of RL agents across different configurations, evaluation datasets, and operational conditions.* • *EL-MLR-EI.3 by enabling configurable validation strategies, reproducible evaluation protocols, and hyperparameter optimization workflows that reduce the risk of overfitting to narrow scenarios and support more robust and generalizable RL behaviour.*
AI/ML Workload Placement	*AI/ML Workload Placement addresses* • *EL-MLR-EI.1 by declaring full dataset provenance in every experiment request: the generation engine (NS3 v3.0), channel model standard (3GPP TS 38.901), propagation*

	scenarios (UMa, UMi, RMa, InH in LoS, NLoS, and probabilistic modes), UE count (up to 300,000), mobility model, and 5QI type distribution (across 27 5QI classes covering GBR and non-GBR traffic for eMBB, URLLC, and mMTC services) are all specified upfront, making the synthetic origin and scope of every dataset explicit. • *EL-MLR-EI.2 by providing continuous observability of data quality through E2SM-KPM metric reporting at defined intervals, with cell-level and UE-level indicators that allow downstream consumers to assess whether generated data remains within expected statistical boundaries. The export pipeline — DT Testbed Adapter to Controller to KPM xApp to Dataspace Connector to Data Space — is fully traceable, and O1/E2 runtime reconfiguration enables correction of out-of-bound generation parameters mid-run without restarting the experiment.* • *EL-MLR-EI.3 by structurally preventing narrow-scenario overfitting through mandatory diversity in the generation configuration: the combination of four propagation environments, four mobility models, and 27 5QI service types ensures that generated datasets cover a broad operational envelope rather than a single representative slice. Upfront parameter declaration in the experiment request prevents scope creep and constrains the generation to the declared conditions.*
AI-RSG and FlexRIC Integration	*AI-RSG and FlexRIC Integration addresses* • *EL-MLR-EI.1 by explicitly documenting the epistemic boundaries of the open RAN experimental environment: the integration operates with E2AP v2.00, KPM Service Model v2.03, and RC Service Model v1.03, and compatibility with E2AP v3 is known to cause ASN.1 decoder failures. Only handover-based RC control actions have been validated end-to-end; admission control, PRB quota management, QoS configuration, carrier aggregation, and beamforming remain under development. These constraints are declared as part of the experimental setup, establishing a clear boundary between validated and unvalidated operation.* • *EL-MLR-EI.2 by making epistemic risk signals observable rather than silent: protocol incompatibilities surface as identifiable ASN.1 assertion errors rather than silent degradation, and the KPM xApp reports network metrics at 1000 ms intervals providing continuous visibility into whether the system is operating within its validated parameter range. The separation of validated RC actions from pending ones provides an explicit signal of the current limits of model validity.* • *EL-MLR-EI.3 by enforcing a stable known-good configuration that constrains the experimental scope to validated conditions: Docker-based deployment pins the FlexRIC version, Combined deployment mode is required, E2SM-CCC procedures must be suppressed to prevent interference, and the ADK Pacing mechanism provides pause/resume control to prevent the system from advancing beyond observable states.*

Large-Scale Data Generation	*Large-Scale Data Generation addresses* • *EL-MLR-EI.1 by declaring full dataset provenance in every experiment request: the generation engine (NS3 v3.0), channel model standard (3GPP TS 38.901), propagation scenarios (UMa, UMi, RMa, InH in LoS, NLoS, and probabilistic modes), UE count (up to 300,000), mobility model, and 5QI type distribution (across 27 5QI classes covering GBR and non-GBR traffic for eMBB, URLLC, and mMTC services) are all specified upfront, making the synthetic origin and scope of every dataset explicit.* • *EL-MLR-EI.2 by providing continuous observability of data quality through E2SM-KPM metric reporting at defined intervals, with cell-level and UE-level indicators that allow downstream consumers to assess whether generated data remains within expected statistical boundaries. The export pipeline — DT Testbed Adapter → Controller → KPM xApp → Dataspace Connector → Data Space — is fully traceable, and O1/E2 runtime reconfiguration enables correction of out-of-bound generation parameters mid-run without restarting the experiment.* • *EL-MLR-EI.3 by structurally preventing narrow-scenario overfitting through mandatory diversity in the generation configuration: the combination of four propagation environments, four mobility models, and 27 5QI service types ensures that generated datasets cover a broad operational envelope rather than a single representative slice. Upfront parameter declaration in the experiment request prevents scope creep and constrains the generation to the declared conditions.*

I-I Bias

This mapping bridges the high-level Bias requirement (EL-HLR-BI) and provides details on how ethics and legal obligations regarding the prevention of bias in data selection, intent translation, and AI optimization are embedded directly into the system's design through an "ethics-by-conception" approach. Specifically, this mapping identifies the Surrogate Based Data Generation as key part of the 6G-DALI framework to address EL-HLR-BI and its mid-level requirements:

- EL-MLR-BI.1 “Bias Risk Identification” (linked to FR-IDD-6, FR-DS 4) mandates the identification and documentation of bias risks introduced through data selection, transfer, and intent translation;
- EL-MLR-BI.2 “Bias Monitoring and Governance” (linked to FR-MO-26, FR-TAI-5) requires the monitoring of bias during validation, simulation, and operations, as well as tracing its effects;
- EL-MLR-BI.3 “Bias in Intent-based Translation” (linked to FR-IDD-3, FR-DATAOPS-8, FR-IDD-6) mandates traceable intent interpretation and the clear distinction between retrieved and generated datasets to prevent the reinforcement of bias during intent translation;
- EL-MLR-BI.4 “Bias Mitigation Mechanisms” (linked to FR-ELT-4, FR-MO-23, NFR-ELT-3) demands safeguards against identified biases, including the application of appropriate de-biasing techniques and retraining strategies;
- EL-MLR-BI.5 “Feedback on Intent Translation” (linked to FR-IDD-3, FR-IDD-6, NFR-DATAOPS-4, NFR-IDD-4) requires verification of intent fulfilment through structured feedback mechanisms to ensure the outcome corresponds to the user's declared intent.

EL-MLR-BI “Bias”	
Related FR/NFR	• *FR-IDD-3, FR-IDD-6, NFR-IDD-4* • *FR-ELT-4, NFR-ELT-3*

	• *FR-DS 4* • *FR-MO-23, FR-MO-26* • *FR-TAI-5* • *FR-DATAOPS-8, NFR-DATAOPS-4*
Relevant Section(s)	Rationale
Surrogate Based Data Generation	*Surrogate Based Data Generation addresses* • *EL-MLR-BI.1 by explicitly documenting the candidate parameter space, measurement budget, KPI targets, and acquisition policy, which helps identify and record bias risks introduced through configuration selection and surrogate-based data generation.* • *EL-MLR-BI.2 by monitoring representativeness and coverage of the sampled configurations against the full candidate space, so potential bias or structural imbalance can be detected and traced during validation and surrogate construction.* • *EL-MLR-BI.4 by enabling mitigation when bias or poor coverage is detected, since high validation error or under-represented regions can trigger additional measurements, re-sampling, or revision of the acquisition strategy.* • *EL-MLR-BI.5 by supporting a closed-loop workflow in which surrogate accuracy and coverage are checked against the intended experimental space, providing structured feedback on whether the generated dataset adequately fulfils the original request.* *EL-MLR-BI.3 is not relevant because there is no translation of user intents in the Surrogate Based Data Generation process.*