

Technical Specification

(07/2020)

ML5G-DEL4

Requirements, architecture and design for machine learning function orchestrator

Technical Specification

Requirements, architecture and design for machine learning function orchestrator

Summary

This Technical Specification describes the requirements and architecture for the machine learning function orchestrator (MLFO) in future networks, including IMT-2020. The mechanisms used in various ML-related use cases in future networks are studied to derive the usage scenarios for MLFO. Based on this study, requirements, architecture, and design for MLFO are presented.

Keywords

Architecture, design, IMT-2020, machine learning, model, network functions, orchestrator.

Note

This is an informative ITU-T publication. Mandatory provisions, such as those found in ITU-T Recommendations, are outside the scope of this publication. This publication should only be referenced bibliographically in ITU-T Recommendations.

Editor:

Slawomir Stanczak
Fraunhofer HHI
Germany

E-mail: slawomir.stanczak@hhi.fraunhofer.de

© ITU 2026

All rights reserved. No part of this publication may be reproduced, by any means whatsoever, without the prior written permission of ITU.

Table of contents

	Page
1 Scope.....	1
2 References.....	1
3 Definitions	1
3.1 Terms defined elsewhere	1
3.2 Terms defined in this Technical Specification	2
4 Abbreviations and acronyms	2
5 Conventions	3
6 Introduction.....	3
7 High-level requirements for MLFO.....	5
7.1 Model management requirements	6
7.2 Data management requirements	8
7.3 Closed-loop system	10
7.4 Communication management.....	11
7.5 Pipeline management.....	11
7.6 Intent and policy management requirements.....	12
8 High level architecture.....	12
8.1 Description of reference points for MLFO.....	17
8.2 Sequence diagrams for MLFO	18
Appendix I – Scenarios for machine learning function orchestration	35
Appendix II – MLFO requirements analysis for the use cases	50
Appendix III – Examples of ML intent.....	80
Appendix IV – Executive summary	81
Bibliography.....	82

Technical Specification

Requirements, architecture and design for machine learning function orchestrator

1 Scope

This deliverable provides the requirements, architecture, and design for machine learning function orchestrator (MLFO) in future networks including IMT-2020.

The scope of this document includes:

- Requirements for MLFO.
- Architecture and design for MLFO.

2 References

- [[ITU-T Y.3172](#)] Recommendation ITU-T Y.3172 (2019), *Architectural framework for machine learning in future networks including IMT-2020*.
- [[ITU-T Y.3173](#)] Recommendation ITU-T Y.3173 (2020), *Framework for evaluating intelligence levels of future networks including IMT-2020*.
- [[ITU-T Y.3174](#)] Recommendation ITU-T Y.3174 (2020), *Framework for data handling to enable machine learning in future networks including IMT-2020*.
- [[ITU-T Y.3176](#)] Recommendation ITU-T Y.3176 (2020), *Machine learning marketplace integration in future networks including IMT-2020*.
- [ITU-T Y.Sup55] ITU-T Supplement 55 to ITU-T Y.3170-series (2019), *Machine learning in future networks including IMT-2020: use cases*.
- [ML5G-I-227-R2] ITU-T FG ML5G ML5G-I-227 (2020), *Serving framework for ML models in future networks including IMT-2020*.
- [ML5G-I-232-R2] ITU-T FG ML5G ML5G-I-232 (2020), *Machine Learning Sandbox for future networks including IMT-2020: requirements and architecture framework*.

3 Definitions

3.1 Terms defined elsewhere

This Technical Specification uses the following terms defined elsewhere:

3.1.1 machine learning database [ITU-T Y.3174]: A component which stores data related to machine learning in the machine learning overlay.

3.1.2 machine learning function orchestrator (MLFO) [ITU-T Y.3172]: A logical node with functionalities that manage and orchestrate the nodes in a machine learning pipeline.

3.1.3 machine learning intent [ITU-T Y.3172]: A declarative description which is used to specify an ML application.

3.1.4 machine learning marketplace [ITU-T Y.3176]: A component which provides capabilities facilitating the exchange and delivery of machine learning models among multiple parties.

3.1.5 machine learning model metadata [ITU-T Y.3176]: Data related to the characteristics of a machine learning model.

3.1.6 machine learning output [ITU-T Y.3172]: Policies or configurations to be applied in the network.

3.1.7 machine learning pipeline [ITU-T Y.3172]: A set of logical nodes, each with specific functionalities, that can be combined to form a machine learning application in a telecommunication network.

3.1.8 simulation profile [ML5G-I-232-R2]: A list of parameters and their values which describe the ML use case to be trained, evaluated, or tested at the ML sandbox.

3.1.9 machine learning sandbox [ITU-T Y.3172]: An environment in which machine learning models can be trained, tested and their effects on the network evaluated.

NOTE – The list of parameters and their values may be derived from ML profile and simulation environment metadata.

3.2 Terms defined in this Technical Specification

This Technical Specification defines the following terms:

3.2.1 freshness: Time-related requirement specified by a machine learning application for considering data to be valid.

3.2.2 machine learning knowledge: Knowledge acquired through the deployment of a model in a specific use case.

NOTE – It includes model performance to reflect operator KPIs and ML application constraints.

3.2.3 machine learning metadata store: A component which stores machine learning data models (MLDM) with their corresponding application programming interfaces.

3.2.4 machine learning profile: Profile that consists of parsed contents of an input intent and runtime updates including operator policies, configurations, simulation metadata, and ML model metadata.

3.2.5 processing profile: Processing capabilities of the machine learning underlay.

NOTE – An example of processing profile can be hardware accelerators.

3.2.6 transfer learning: Application of machine learning (ML) knowledge to different but related use cases.

NOTE – The inverse of freshness is data staleness.

4 Abbreviations and acronyms

This Technical Specification uses the following abbreviations and acronyms:

AF	Application Function
API	Application Programme Interface
CN	Core Network
CP	Control Plane
DMM	Data Management Module
DNN	Deep Neural Network
DU	Distributed Unit
E2E	End-to-End
KPI	Key Performance Indicator

MEC	Mobile Edge Computing
ML	Machine Learning
MLDM	Machine Learning Data Model
MLFO	Machine Learning Function Orchestrator
MM	Model Management Module
NF	Network Function
NFV	Network Function Virtualization
NFVO	Network Function Virtualization Orchestrator
NMS	Network Management Subsystem
QoS	Quality of Service
RAN	Radio Access Network
SBA	Service-Based Architecture
SIM	Simulator
src	source (ML pipeline)
UE	User Equipment
VNF	Virtual Network Function

5 Conventions

In this Technical Specification, requirements are classified as follows:

- The keywords "**is required to**" indicate a requirement which must be strictly followed and from which no deviation is permitted if conformance to this document is to be claimed.
- The keywords "**is recommended**" indicate a requirement which is recommended but which is not absolutely required. Thus, such requirements need not be present to claim conformance.
- The keyword "**may**" indicates an optional requirement which is permissible, without implying any sense of being recommended. These terms are not intended to imply that the vendor's implementation must provide the option and the feature can be optionally enabled by the NOP/service provider. Rather, it means the vendor may optionally provide the feature and still claim conformance with the specification.

6 Introduction

Most network operators struggle to integrate machine learning (ML) with their networks and often rely on data scientists to create an ML pipeline, i.e., from data collection to model deployment. This ML pipeline, if not managed and orchestrated appropriately, is subject to bottleneck. An investigation of the diverse 5G and beyond ML use cases as given in [ITU-T Y.Sup55] reveals that these use cases are subject to massive, heterogeneous, and multidimensional data. These challenging characteristics call for the requirements of big data-enabled tools that can collect, store, and pre-process the data to train the ML models.

Specifically, this Technical Specification addresses the following challenges:

- The study of use cases from the [ITU-T Y.Sup55] envisages different ML frameworks and libraries. For example, to support the training of deep learning, tree-based, and linear models, a network operator needs to deploy three different ML frameworks. To address this, data scientists rely on big data handling tools and libraries including messaging brokers [b-Dsilva]

and [b-Vera-Baquero], data processing engines [b-Rathore], TensorFlow [b-Kwok], Keras [b-Mohammadi] and Pytorch [b-ruiz18]. However, the integration of these tools and libraries to create an ML pipeline is time-consuming and may require a set of specialized skills.

- The integration aspects as required by the ML frameworks including [ITU-T Y.3173], [ITU-T Y.3176], [ITU-T Y.3174], [ML5G-I-227-R2], and [ML5G-I-232-R2] are subject to compatibility issues and require repetitive and complex code which is difficult to manage.
- Advances in the data handling tools and libraries have not witnessed any standard formats and interfaces to orchestrate the ML frameworks. Lack of a standardized ML orchestration mechanism can result in highly complex and expensive ML pipelines across 5G and beyond use cases, as investigated in [ITU-T Y.Sup55].
- Without a standard ML pipeline orchestration mechanism, handover between different ML pipeline nodes within a pipeline or across the pipelines may lead to bottleneck and production issues. These issues include glue code, hidden dependencies, feedback loops, and pipeline nets which represent ML antipatterns as discussed by Sculley *et al.* [b-Sculley15].
- Other challenges in the context of future generation networks may include ML model update, model optimization, ML pipeline chaining, ML pipeline performance monitoring, evaluation, pipeline splitting, policy-based ML pipeline deployment, and management and coordination of multiple ML pipeline instances across the network.

Currently, there are efforts to address some of the above challenges by the big players including Uber, Netflix, Google, Facebook, and Airbnb with the help of custom ML orchestration platforms. However, these solutions aim at in-house ML pipeline management and do not address the requirements of diverse use cases as discussed in the [ITU-T Y.Sup55]. These solutions do not offer any services to integrate with the [ITU-T Y.3173], [ITU-T Y.3176], [ITU-T Y.3174], [ML5G-I-227-R2], and [ML5G-I-232-R2] ML frameworks.

In-house ML orchestration solutions require significant investments without the actual realization of their benefit within an industry setting. Major cloud service providers including Amazon, Google, and Microsoft currently offer different ML pipeline orchestration services including data collection, pre-processing, training, and deployment of ML models as part of their cloud. However, they do not offer direct integration of their solutions to the diverse set of requirements as investigated from the use cases given in [ITU-T Y.Sup55] and with the ML frameworks proposed the [ITU-T Y.3173], [ITU-T Y.3176], [ITU-T Y.3174], [ML5G-I-227-R2], and [ML5G-I-232-R2].

As described in [ITU-T Y.3172], the MLFO plays an important role in managing machine learning models in the operator's network. Following are the most important functionalities of the MLFO:

- MLFO is a logical orchestrator that can monitor and manage the nodes in a machine learning pipeline. It supports the policy-based deployment of an ML pipeline in a variety of ML underlay networks.
- Optimal placement of ML pipeline nodes is achieved in the network with the help of MLFO, coordinated with the management plane of the ML underlay networks.
- MLFO lets the network operator specify the ML application using a declarative specification, abstracted from the ML underlay networks. This helps in standard representation, handling, and comparison of ML applications in the operator's network. This also helps in clear separation of technology-specific interfaces and the ML application. Such a specification is important for interoperable integration of the ML data handling framework [ITU-T Y.3174], [b-ML5G-I-148], and ML marketplaces [ITU-T Y.3176] to operator's network.
- Chaining and split of ML pipeline nodes, selection of ML models, monitoring their performance, reselection, and update, if needed, are achieved using MLFO.

- Like the network function virtualization orchestrator (NFVO), MLFO decouples ML functions from the underlying network. It thereby expands ML functionality for both enterprises and service providers and enables both to create new capabilities.
- Further, like NFVO and MLFO networks aim to reduce ML pipeline-related operational costs with accelerated new offerings for service providers. Cloud service providers, mobile network operators, and other service providers can retain traditional networks in place. The MLFO solution aims to replace components of their overall solutions and are interoperate with existing functions, i.e., operations support system and business support system (OSS/BSS), and legacy networking systems.

The data handling framework as discussed in [ITU-T Y.3174] interacts with the MLFO to create an ML pipeline for the data collection and ML output. Further, it has been emphasized in the [ITU-T Y.3176] that MLFO plays a vital role in managing the internal ML marketplace by instantiating an ML pipeline to evaluate the performance of ML model before it can be adopted by the internal ML marketplace. The intelligence evaluation framework proposed in [ITU-T Y.3173] necessitates a standard mechanism for monitoring across different implementations of ML overlay with the assistance of MLFO. MLFO can create an ML pipeline across the ML underlay to monitor the intelligence level and performance of the network.

The main objective of MLFO is to overcome the above-discussed challenges including integration, orchestration, and management of pipeline nodes and dependencies in an ML pipeline while reducing operational costs. To handle pipeline dependencies, MLFO can infer relationship between different pipeline nodes, e.g., training and inference to improve the automated model deployment. MLFO will also address the interdependence among ML pipelines in distributed large-scale scenarios, thus acting as strong governance. It also aims to offer a unified architecture to facilitate the orchestration of end-to-end (E2E) ML workflows including data collection, pre-processing, training, model inference, model optimization, and model deployment. MLFO can also monitor and evaluate ML pipeline instances to optimize its performance by taking appropriate decisions including model update, retraining, model redeployment, and model chaining, etc. Further, the unified MLFO architecture aims to facilitate easy integration with the existing ML frameworks including [ITU-T Y.3173], [ITU-T Y.3174], [ML5G-I-227-R2], and [ML5G-I-232-R2], and ML marketplaces [ITU-T Y.3176].

MLFO aims to hide the underlying complexities of orchestrating ML pipeline nodes by providing an abstraction to the users and developers with the help of high-level application programme interfaces (APIs). Moreover, it targets to address the challenge of running multiple pipeline workflows in parallel and ensure that these pipeline workflows do not affect each other.

MLFO architecture will bring the flexibility, reusability, and extension of the ML pipeline to accommodate the rapid pace and development in ML pipeline nodes, e.g., ML models. It can split the ML pipeline and offers flexibility to reuse the ML pipeline nodes to offer specialized services.

7 High-level requirements for MLFO

Appendix I provides a detailed description of various usage scenarios and relevant requirements based on the [ITU-T Y.Sup55]. The high-level requirements discussed in this section have been extracted from the use case requirements discussed under Appendix I.

High-level requirements of MLFO are divided into the following broad classifications:

- Model management requirements.
- Data management requirements.
- ML pipeline management.
- Closed loop system.
- Intent and policy management requirements.

- Communication management requirement.

7.1 Model management requirements

REQ-ML-MLFO-001: MLFO is required to select ML models and learning mechanisms considering application requirements, constraints, and data models as specified in the ML intent.

NOTE 1 – Example of considerations for selection include ML model metadata, e.g., version of ML models.

NOTE 2 – Example ML model types include supervised, semisupervised, unsupervised, or hybrid models.

NOTE 3 – Examples of learning mechanisms include federated learning and distributed learning.

NOTE 4 – Examples of local and global predictions and optimizations enabled by ML models are resource allocations, optimization of radio access configurations, network planning, optimization of storage and retention, fault detection, fault recovery, and traffic forecasting, etc.

NOTE 5 – Example of consideration for selection of ML models include generalization of ML models under different network deployments.

REQ-ML-MLFO-002: MLFO is recommended to manage and orchestrate the publishing of ML profiles by deployed ML pipeline nodes.

NOTE – Example of published profiles is the metadata associated with ML models. The published profiles may be used for model discovery [ML5G-I-232-R2].

REQ-ML-MLFO-003: MLFO is required to select an appropriate training mechanism based on the characteristics of the ML mechanism and the constraints as specified in the ML intent.

NOTE 1 – Examples of types of training mechanisms are offline or online training.

NOTE 2 – Example of constraints include minimizing an impact on the performance of the ML application in the network.

REQ-ML-MLFO-004: MLFO is required to manage and orchestrate the retraining of an ML model based on network dynamics and the constraints as specified in the ML intent.

NOTE 1 – Examples of constraints are accuracy optimization of the ML model, optimization of the network KPIs and architecture of the ML application, e.g., distributed, or centralized.

NOTE 2 – Examples of mechanisms to optimize training or retraining include data parallelism and model parallelism.

NOTE 3 – Example of criteria for training and retraining include training based on network data using an offline training method and retraining based on meta-data only.

NOTE 4 – Example of criteria for retraining an ML model is a change in deployment scenario, e.g., change in deployment of ML model from rural network to urban network.

REQ-ML-MLFO-005: MLFO is required to manage and orchestrate an update of ML models in the ML marketplace.

NOTE 1 – Example of an update scenario includes upload of a trained or tested model to the ML marketplace according to the details and criteria specified in the ML intent.

NOTE 2 – Example of model update can be based on operator KPIs which should be stored as ML model metadata to assist with the ML knowledge.

REQ-ML-MLFO-006: MLFO is recommended to manage and orchestrate the use of real-world data sets augmented with simulated data sets to train and retrain a ML model.

REQ-ML-MLFO-007: MLFO is required to deploy ML pipeline nodes in the network, in coordination with management subsystem of the underlay network, to meet the ML application requirements.

NOTE 1 – Examples of considerations about deployment of ML pipeline nodes include location sensitivity analysis, e.g., ML model deployment at edge network or cloud and resource requirements of the ML model derived from the ML model metadata [ITU-T Y.3176].

NOTE 2 – Examples of further considerations about deployment of ML pipeline nodes are current state of the ML underlay obtained from the management subsystem of the ML underlay and the configurations of other ML pipeline nodes.

NOTE 3 – Example of consideration about deployment of ML pipeline nodes is compliance to operator policies (including local regulatory policies) about the deployment of ML pipeline nodes.

NOTE 4 – ML pipeline nodes deployment can be triggered by the ML serving framework [ML5G-I-227-R2] for inference.

NOTE 5 – ML application requirements are specified in the ML intent.

REQ-ML-MLFO-008: MLFO is required to manage operator policies which should be applied to operations on ML pipeline nodes.

NOTE 1 – Examples of operations on ML pipeline nodes are stop, start, restart, update, train, retrain of models, etc.

NOTE 2 – Examples of management operations on operator policies are provision, update, delete the policies.

REQ-ML-MLFO-009: MLFO is required to manage and orchestrate the automated testing of ML models in ML sandbox where models can be evaluated to satisfy the operator-specified KPIs.

REQ-ML-MLFO-010: MLFO is recommended to chain various ML pipeline nodes based on application requirements and constraints.

NOTE 1 – Example of consideration includes compliance to security and privacy regulations to chain sources sinks.

NOTE 2 – Example of chaining include packaging of ML submodels as software/containers based on input/output ML models given in the intent.

REQ-ML-MLFO-011: MLFO is recommended to manage and orchestrate ML model performance using hardware acceleration capabilities.

REQ-ML-MLFO-012: MLFO is recommended to evaluate multiple ML models in parallel and select the matching model for the ML application based on the characteristics of the ML application and the constraints in the ML intent.

REQ-ML-MLFO-013: MLFO is recommended to fall back to alternate ML models based on the evaluation of performance of the serving ML models for a preconfigured time window after deployment in the ML pipeline.

NOTE – MLFO can evaluate the performance of the deployed ML model for a preconfigured fall-back window within which if the performance or stability does not meet the operator KPI criteria indicated in the ML intent then rollover or fallback is initiated to an alternate or stable version of the ML model.

REQ-ML-MLFO-014: MLFO is recommended to manage and orchestrate different operations on ML pipeline functions with minimal impact on the operator KPIs.

NOTE 1 – Examples of the different ML pipeline functions are data collection, pre-processing, model selection, and training decisions.

NOTE 2 – Examples of operations are stop, start, restart of ML pipeline functions.

NOTE 3 – These operations can be invoked by the ML serving framework [ML5G-I-227-R2] for inference optimizations.

REQ-ML-MLFO-015: MLFO is recommended to manage the storage of data and ML models in a format based on the data and model characteristics so that the network KPIs are optimal for the ML use case.

NOTE – Examples of formats for data and models are compressed or encrypted formats.

REQ-ML-MLFO-016: MLFO is recommended to manage and orchestrate the performance of each ML pipeline node such that the overall performance of the ML use case is within the specified constraints in the ML intent.

REQ-ML-MLFO-017: MLFO is recommended to manage and orchestrate the application of the ML output at different levels and to various functions in the underlay network in coordination with management subsystem of the underlay network, so as to meet the ML application requirements.

NOTE 1 – Examples of functions to which ML output may be applied are gNB, network management subsystem (NMS), and edge servers.

NOTE 2 – See [ITU-T Y.3172] regarding the different levels in the network.

REQ-ML-MLFO-018: MLFO is recommended to manage and orchestrate different simulated data sources.

NOTE – Simulated data sources are used to generate data to evaluate the performance of an ML model before deployment (see [ML5G-I-232-R2]).

REQ-ML-MLFO-019: MLFO is recommended to manage and orchestrate the data handling in the ML pipeline in accordance with privacy policies and regulations. See [ITU-T Y.3174] for data handling mechanisms.

NOTE – Feature extraction without deep packet inspection is an example to ensure privacy of data.

REQ-ML-MLFO-020: MLFO is recommended to support on-demand management of ML pipeline nodes based on handling of triggers from trusted entities.

NOTE 1 – Examples of management operations are creation and deletion of instances of ML pipeline nodes.

NOTE 2 – Examples of trusted entities are the serving framework for ML models and the ML sandbox.

REQ-ML-MLFO-021: MLFO is recommended to upload the tested ML model to the ML marketplace in accordance with the operator KPIs.

7.2 Data management requirements

REQ-ML-MLFO-022: MLFO is required to manage and orchestrate data collection for machine learning (ML) across the network based on the requirements of the ML use case in the ML intent.

NOTE 1 – Examples of requirements in the ML intent for data collection are periodic, real-time, probed-based, or event-driven data collection at various levels of the network. The requirements for data to collect may include operational, planning, and performance data.

NOTE 2 – Data collection from different network functions at various levels including control, service, and data plane may be specified in the ML intent.

NOTE 3 – Requirements can also include probed-based data collection where specific signalling is used to acquire the required measurements with the assistance of measurement plugins or active probing monitoring systems.

NOTE 4 – Event-driven data collection may need synchronization with the Internet of things (IoT) sensors, cameras, and warning signals, etc.

REQ-ML-MLFO-023: MLFO is recommended to interface with the dedicated measurement infrastructure and external repositories in the underlay networks for data collection.

REQ-ML-MLFO-024: MLFO is required to manage and orchestrate pre-processing tasks on multidimensional data.

NOTE – Examples of pre-processing tasks are labelling of data based on service/application type, normalization, aggregation/fusion, verification, prioritization, timestamped summaries, and reconstruction.

The pre-processing tasks may convert the data to a standard or encrypted format to enable sharing while ensuring privacy.

REQ-ML-MLFO-025: MLFO is recommended to coordinate with simulated ML underlay to support specific algorithms for pre-processing tasks.

NOTE 1 – Example of coordination with simulated ML underlay for pre-processing tasks is data augmentation using simulated data.

NOTE 2 – Example of pre-processing is refinement of raw data by filling missing information or flagging outliers.

REQ-ML-MLFO-026: MLFO is recommended to manage and orchestrate storage of data according to the requirements specified in the ML intent.

NOTE 1 – Example of management of storage is enabling data cache facilities at various levels of the network to accelerate local and global predictions. The storage may use centralized or distributed instances based on the application requirements and training constraints.

NOTE 2 – Example of requirements specified in the ML intent is application-specific privacy guidelines.

REQ-ML-MLFO-027: MLFO is recommended to manage and orchestrate time sensitivity analysis of storage of data.

NOTE – Example of analysis is proactive data clean-up to support complex storage requirements of the ML application.

REQ-ML-MLFO-028: MLFO is recommended to enable configuration of data models which are generated during the data handling procedures or are provided in the ML intent.

NOTE – Data handling procedures are specified in [ITU-T Y.3174].

REQ-ML-MLFO-029: MLFO is recommended to manage and orchestrate transfer of ML model metadata between the instances of ML pipelines.

NOTE 1 – Example of ML model metadata are described in [ITU-T Y.3176].

NOTE 2 – This may be used for transfer learning.

NOTE 3 – The ML model metadata can have version number to assist with transfer learning.

REQ-ML-MLFO-030: MLFO is required to manage the ML knowledge which is applied on ML models.

NOTE – Examples of management of ML knowledge are provision, import, update, delete, etc. of ML knowledge. This may involve interface with the serving framework for ML models [ML5G-I-227-R2].

REQ-ML-MLFO-031: MLFO is recommended to manage and orchestrate a data handling framework for the specific purpose of evaluating the performance of the ML model.

REQ-ML-MLFO-032: MLFO is required to collect and record metadata during data collection and performance monitoring.

REQ-ML-MLFO-033: MLFO is required to manage and orchestrate (in coordination with the management subsystem of the ML underlay) the time synchronization of operations on ML pipeline nodes and the ML underlay network.

NOTE 1 – An example is synchronization between SRC and SINK for data collection and application of ML output.

NOTE 2 – Example of operations on ML pipeline nodes and ML underlay networks are deployment and update of ML models.

REQ-ML-MLFO-034: MLFO is recommended to manage and orchestrate data collection at various levels and granularities.

NOTE 1 – Examples of various levels are access network and core network (CN) (see [ITU-T Y.3172]).

NOTE 2 – Examples of data collection granularities are data at user level or cell level.

REQ-ML-MLFO-035: MLFO is recommended to manage and orchestrate the freshness of stored data according to application requirements in the ML intent.

NOTE 1 – Example is to use time-stamp-based mechanism to evaluate freshness of data.

NOTE 2 – Example of freshness requirement is different for various ML applications. An ML application can have a freshness requirement of few minutes, e.g., reporting of network failures. Another ML application to analyse structure of network graph might have a freshness requirement on the order of days due to less frequent changes.

REQ-ML-MLFO-036: MLFO is recommended to manage and orchestrate the mapping of technology-specific data model specified in the ML intent to the generic data model during the data handling procedures.

NOTE – Data handling procedures are specified in [ITU-T Y.3174].

7.3 Closed-loop system

REQ-ML-MLFO-037: MLFO is recommended to monitor and optimize the model management and data model management mechanisms.

NOTE 1 – Example of closed loop system and optimizations include performance optimization of MM and DM models considering network operator feedback.

NOTE 2 – Examples of operator feedback are policies for applying inference rules and configuration of ML model metadata.

REQ-ML-MLFO-038: MLFO is recommended to configure and manage monitoring mechanisms that can be triggered based on certain events or preconfigured thresholds.

NOTE 1 – Example of measurement mechanism to support the event generation is via probes.

NOTE 2 – Example of usage of monitoring mechanism is to monitor the performance of ML pipeline nodes and security and privacy notifications.

NOTE 3 – Example of handling of events is triggering policy-based actions like retraining, auto-scaling, and other resource management actions, etc. Event handlers may include threat mitigation mechanisms.

REQ-ML-MLFO-039: MLFO is recommended to support handling of metering events which give metrics on usage of ML pipeline nodes from trusted entities.

NOTE 1 – Examples of trusted entities are ML sandbox and serving framework for ML models. Example of usage of metering event is for billing and licensing purposes.

NOTE 2 – Examples of handling of metering events include triggering policy-based actions like creating license invoices and quota reminders.

REQ-ML-MLFO-040: MLFO is recommended to provide security and privacy to the data while in the data store or used by the DM module.

NOTE – Examples of mechanisms to provide security and privacy are constraints on the use and accessibility of data by SRC, e.g., enable pre-processing tasks to extract/process packet information using header only, i.e., without deep packet inspection. Chaining and configuration of data sources and sinks according to the data privacy policies is another example.

REQ-ML-MLFO-041: MLFO is recommended to apply energy-efficiency policies on the operation of MM and DM modules.

NOTE – Example of energy-efficiency policy is distributing the load of the ML pipeline to use light-weight ML mechanisms where possible in the ML use case.

REQ-ML-MLFO-042: MLFO is recommended to manage and orchestrate resource allocation to the ML pipeline nodes.

NOTE 1 – Examples of considerations for resource allocation are requirements of the ML use case, constraints of the ML underlay and the current performance of the ML pipeline nodes.

NOTE 2 – Examples of optimization strategies used for resource allocation are compression of models and selection of underlay functions based on resource status.

REQ-ML-MLFO-043: MLFO is recommended to manage and orchestrate resilient operation of the MM and DM so that the impact to ML use case is minimized under failures.

NOTE – Example of mechanisms for resiliency is to deploy the ML pipeline nodes in a redundant and distributed fashion across the network. Another example is load balancing between instances of ML pipeline nodes.

REQ-ML-MLFO-044: It is recommended that the MLFO manage and orchestrate data collection, training, model selection, and deployment decisions considering the overhead.

NOTE – Example of overheads are additional memory, compute and network resources needed for executing the data collection, training, model selection, and deployment decisions.

REQ-ML-MLFO-045: MLFO is required to test and deploy the model in the ML sandbox to evaluate it in terms of operator KPIs [ML5G-I-232-R2].

NOTE – Example of evaluation includes triggering various configurations for the ML use case to best meet the operator KPIs.

7.4 Communication management

REQ-ML-MLFO-046: MLFO is recommended to manage and orchestrate specialized data formats and protocols to enable communication between ML pipeline nodes.

REQ-ML-MLFO-047: MLFO is required to query the management subsystem for the corresponding technology-specific interface to interact with the ML underlay.

NOTE – These interfaces can be used by the MLFO to enable/configure data sources, output, deployment, and orchestration of ML pipeline nodes, etc.

REQ-ML-MLFO-048: MLFO is recommended to manage and orchestrate interfaces between ML pipelines in network operator domains.

NOTE – Example of usage of this interface is to enable interoperable transfer of ML knowledge among network operators.

REQ-ML-MLFO-049: MLFO is recommended to support subscription of trained models by management subsystem.

NOTE – Example of application of subscription is to allow reuse of trained models in similar ML use cases without incurring any extra training overhead.

7.5 Pipeline management

REQ-ML-MLFO-050: MLFO is recommended to create/modify/delete ML pipeline nodes while minimising any impact on the performance of other ML pipeline nodes.

REQ-ML-MLFO-051: MLFO is recommended to manage and orchestrate multiple instances of ML pipeline nodes for specific ML use cases.

NOTE – Example of usage of multiple instances of ML pipeline nodes is parallel evaluation of ML mechanisms. See [ML5G-I-232-R2].

REQ-ML-MLFO-052: MLFO is recommended to provide ML profile derived from the use case to trusted entities.

NOTE 1 – Examples of trusted entities are serving framework and ML sandbox.

NOTE 2 – Example of usage of ML profile is by serving framework to make deployment configurations.

NOTE 3 – ML profile also includes simulation metadata which describes the simulation environment, simulated NFs, connections and configurations, etc.

NOTE 4 – It also includes ML model metadata [ITU-T Y.3176] to assist with model selection decision.

7.6 Intent and policy management requirements

REQ-ML-MLFO-053: It is recommended that MLFO considers ML application specifications including requirements, constraints, and flexible data models provided as an ML intent in a declarative format.

NOTE – Examples of these constraints are accuracy, latency, and other application specific requirements, e.g., desired network KPIs.

REQ-ML-MLFO-054: MLFO is recommended to consult and apply operator policies before deploying any output from the ML pipeline on the ML underlay.

REQ-ML-MLFO-055: MLFO is recommended to adhere to data retention/deletion policies at the granular level which should be in accordance with regulations and privacy laws.

REQ-ML-MLFO-056: MLFO is recommended to provide requirements and constraints of a use case to the trusted entities.

NOTE 1 – These trusted entities can include serving framework for ML models [ML5G-I-227-R2].

NOTE 2 – These requirements and constraints can be provided in an online or batch manner.

8 High level architecture

The following high-level MLFO framework in Figure 1 is derived based on the high-level requirements given in clause 7:

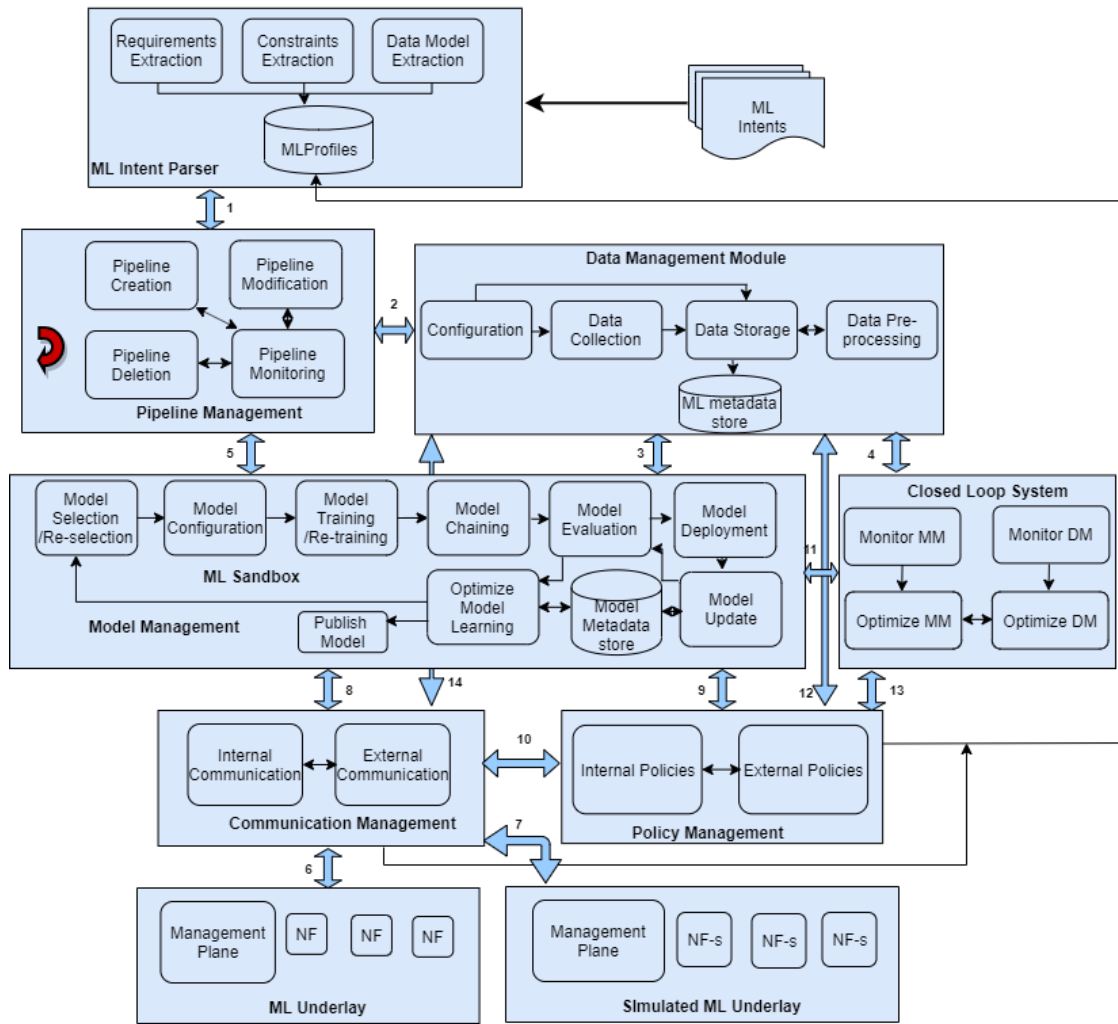


Figure 1 – Architecture for MLFO

NF-s : Simulated Network Functions
 NF : Network Functions
 DMM : Data Management Module
 API-s : ML underlay specific APIs
 MM : Model Management Module

The components of the MLFO architecture and its reference points are discussed below.

Model management module

Figure 2 illustrates the model management module (MM).

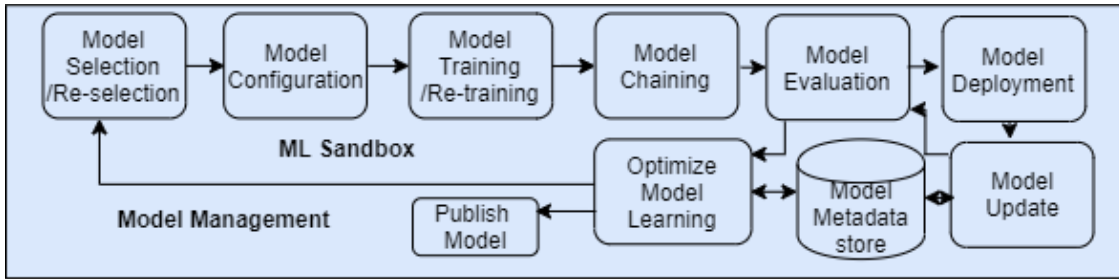


Figure 2 – Model management module

This function component manages different functions related to the ML models based on the application requirements, constraints, and model performance. ML models may have an API associated with them, called API-m (model API). This component also manages metadata for each model, which includes information like hyperparameters, model creation time, transformation of features, and dataset. During the model evaluation process, the metadata is queried by the model evaluation module to enable comparison of new models with existing models to quantify improvements as required by the application. Furthermore, this ML metadata can be used to accelerate or optimize model learning.

The model management module (MM) shown in Figure 2 is also responsible for deciding when to train/retrain a model and whether to train it online or offline. If a new event occurs for which a model has not been trained earlier, the model management module triggers retraining or even model reselection. Further, if the MM module detects any change in data with the help of data handling framework and these changes affect application requirements, it can also trigger model reselection or model retraining. A change in deployment constraints and policies can also trigger model update, model reselection, and model evaluation. MM also ensures that different ML models can be chained or split, where models can feed models for an incremental learning and prediction.

Data management module

This function provides the support for data collection and pre-processing. It interfaces with the data management framework to configure sources using DBr-UP and API-s for the data collection. Further, based on the data models as specified by the data handling framework for a model, the data management module (DMM) given in Figure 3 initiates data collection and configures storage. It stores the collected data in a data store to pre-process later. Once data is pre-processed, it is stored back in the model data store with the assistance of the data storage module.

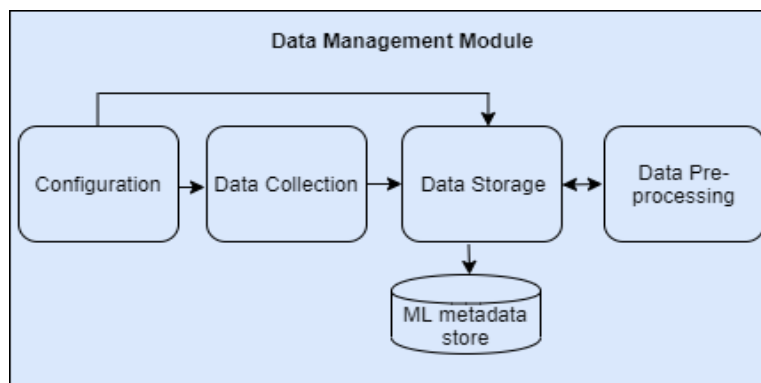


Figure 3 – Data management module

Closed loop system

Figure 4 shows the closed loop system.

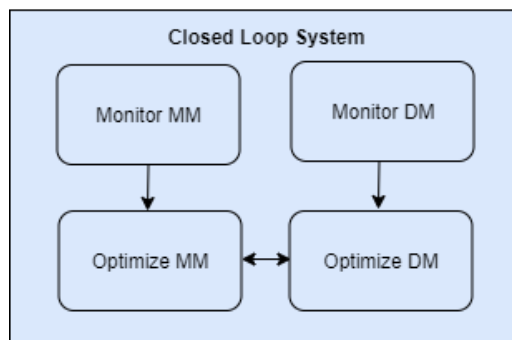


Figure 4 – Closed loop system

This component monitors the performance of both model management and data management modules. It can take appropriate measures to optimize the performance of both the modules. Performance optimization may include energy efficiency, resource optimization or use of accelerators, and resilient operation of the ML pipeline.

Simulated ML underlay

Figure 5 illustrates the simulated ML underlay, as defined in [ITU-T Y.3172].

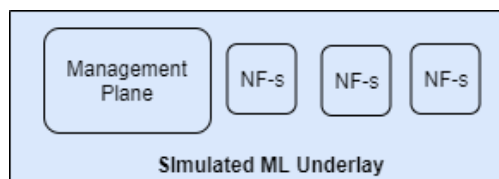


Figure 5 – Simulated ML underlay

ML underlay

Figure 6 illustrates the ML underlay, as defined in [ITU-T Y.3172].

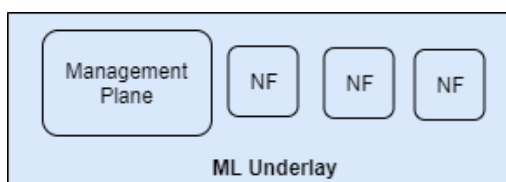


Figure 6 – Simulated ML underlay

Intent parser

The appropriate operation of MM and DM module depends on the application requirements, constraints, and any specified data models. The ML intent parser, as depicted in Figure 7, receives ML intents specified by the user. It extracts requirements, constraints, and data models from the intent and stores them in a data store called ML application profiles. It is possible for DM and MM to refer to application requirements and constraints with a query to ML application profiles.

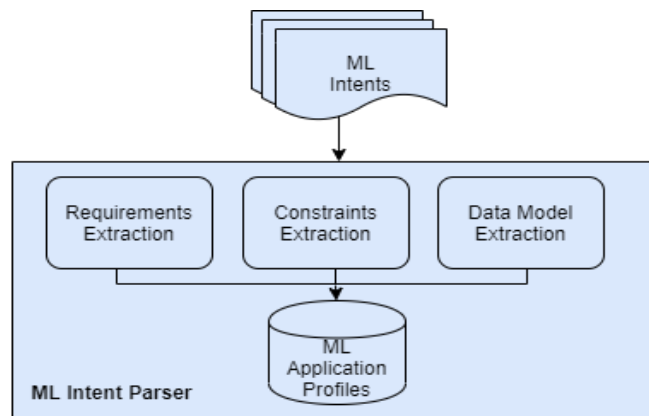


Figure 7 – Intent parser

Communication management

This component of the MLFO ensures any communication within the ML pipeline or with external entities which is critical for the overall operation of the ML pipeline. Any external communications or external policies can be communicated across the pipeline using this module. This module, as shown in Figure 8, also ensure any technology-specific communication with the ML underlay by supporting appropriate technology-agnostic interfaces.

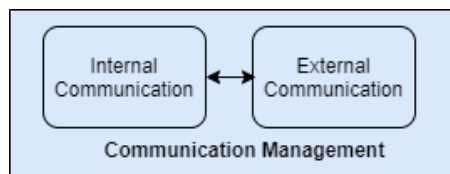


Figure 8 – Communication management module

Policy management

Model deployment or ML output decision is dependent on the external policies which are specified by the operators or users. This module depicted in Figure 9 ensures that the external policies are mapped to internal policies to apply across the ML pipeline. For example, for the energy management operation of the ML pipeline, this module can formulate policies to create ML pipeline nodes in an energy-efficient manner.

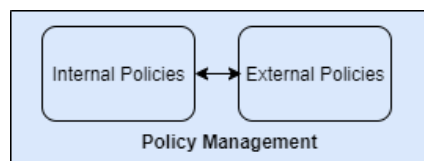


Figure 9 – Policy management module

Pipeline management

It is critical to manage, modify, delete, or monitor the performance of an ML pipeline subprocedures based on the policies as specified by the policy management module. This module, as shown in Figure 10, entertains any subprocedure requests to create, modify, delete, or monitor the operation of an ML pipeline from other ML frameworks including data handling and ML marketplace frameworks.

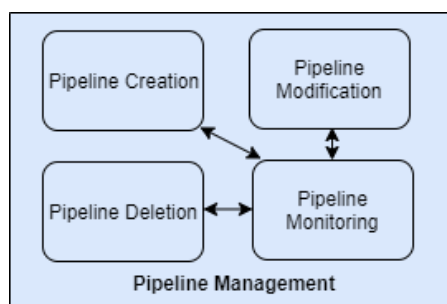


Figure 10 – Pipeline management module

In addition to the above, the following supporting functions which are defined in [ITU-T Y.3172] are used in the framework for data handling.

- 1) **ML sandbox** [ML5G-I-232-R2] an environment in which the model management module is executed. It is primarily used to train/retrain, test, optimize, evaluate, and deploy the model.
- 2) **SRC** node assists the data management module to collect the data.
- 3) **SINK** is the target node on which model management module deploys the model.

8.1 Description of reference points for MLFO

The MLFO architecture is primarily based on these reference points based on [ITU-T Y.3172]:

- Reference point 6 is defined in [ITU-T Y.3172] as the interface between MLFO and simulated ML underlay.
 - Reference point 7 is defined in [ITU-T Y.3172] as the interface between MLFO and other management and orchestration functions, e.g., those defined in [ITU-T Y.3111].
- NOTE – Only the reference points between subsystems are shown, those within the subsystems are not shown.
- Reference point 5 is defined in [ITU-T Y.3172] as the interface between MLFO and ML pipeline subsystem.

Additional reference points for the MLFO are given below:

- Reference point 1 defines the interface between the intent parser module and the pipeline management module.
- Reference point 2 is used to interface the pipeline management module and the data management module.
- Reference point 3 is defined as the interface between the model management and data management modules.
- Reference point 4 represents an interface between data management module and close loop system.
- Reference point 5 is an interface between the pipeline management and model management modules.
- Reference point 8 denotes an interface between the model management and communication management module.
- Reference point 9 represents an interface point between the model management and policy management module.
- Reference point 10 is an interface between the communication management and policy management modules.

- Reference point 11 represents the interface between the model management and closed-loop system.
- The close-loop system can interact with the policy management module by using the reference point 13 to optimize the performance of model management and data management modules.
- Reference point 12 ensures interactions between the data management and policy management modules.
- The data management module can interact with the ML underlay through its reference point 14 with the communication module.

8.2 Sequence diagrams for MLFO

The following clause illustrates some interactions between MLFO and other components.

8.2.1 Data collection flow

Figure 11 depicts MLFO-enabled data collection.

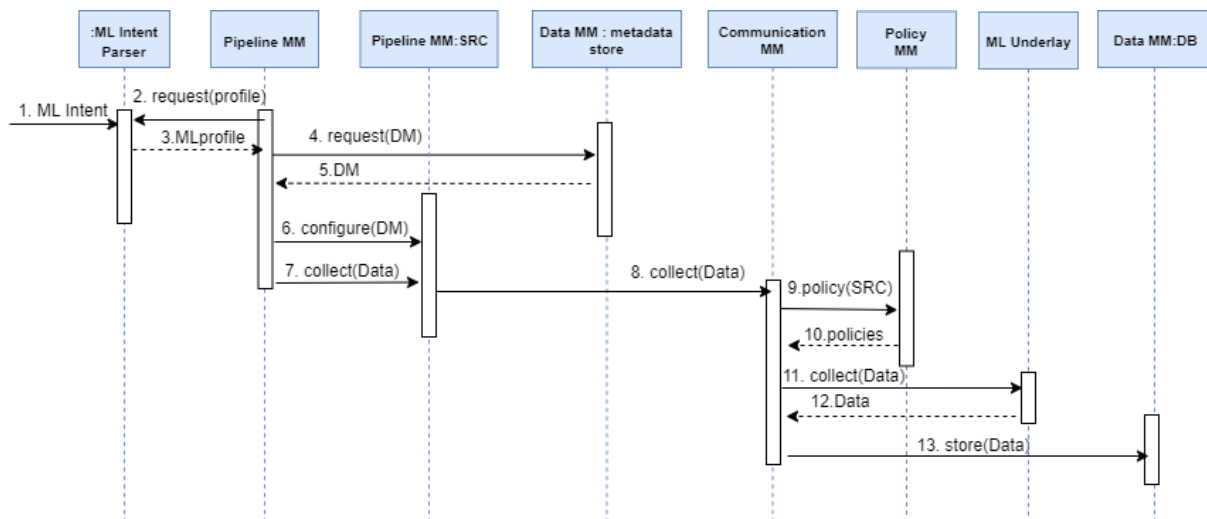


Figure 11 – MLFO-enabled data collection

The sequence of calls and responses is illustrated below:

NOTE – MLFO-enabled data collection assumes that the NFs are enabled on the ML underlay by the operators. ML intent is provided as input to the ML intent parser. ML intent includes requirements and constraints of an ML application. These requirements and constraints are used by the pipeline management module. These requirements may include data models or any other inputs to the MLFO.

- 1 ML intent parser extracts and stores these requirements and constraints in the ML profile database. These requirements and constraints can be queried later by the pipeline management module.
- 2 Pipeline management module performs a lookup for the ML metadata store from the data management module based on the application requirements for the appropriate data model.
NOTE 1 – Data model can interface with the DBr-CP of data handling framework to retrieve appropriate data models if not available within the metadata store of data management module.
- 3 Once an appropriate data model has been retrieved from the data management module, pipeline management module configures the sources and database according to it.
- 4 After configuration is complete, pipeline management module initiates data collection request with the help of communication management module.

- 5 The communication management module may consult any policies specified for the source/ML underlay for data collection by interfacing with the policy management module.
- 6 The communication management module interfaces with the ML underlay to collect data and stores the data in the data store in the data management module.

8.2.2 Data pre-processing flow

Figure 12 illustrates the MLFO-enabled data pre-processing.

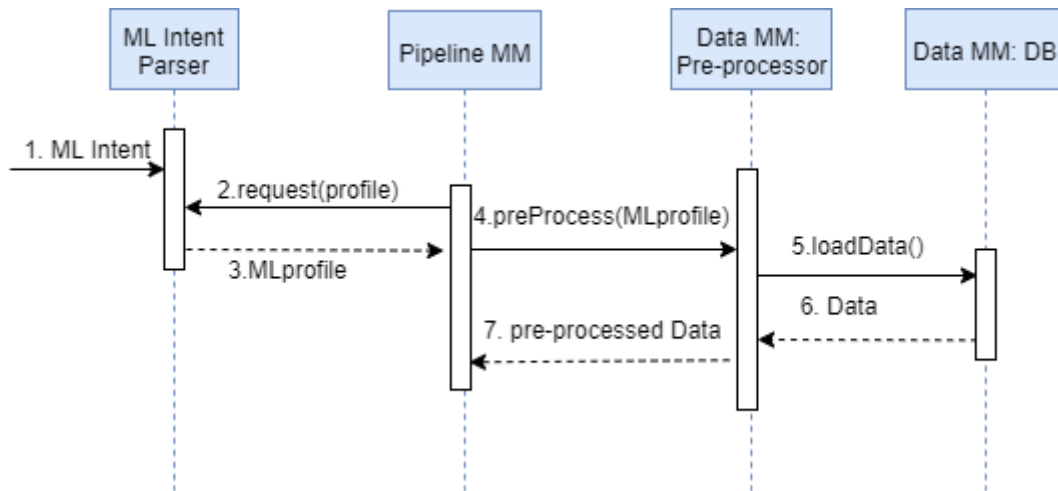


Figure 12 – MLFO-enabled data pre-processing

The steps given in Figure 12 are described below:

- 1 Pipeline management module retrieves ML profile of the application from the ML intent parser.
- 2 Pipeline management module issues a pre-processing request to the data management module and specifies the application requirements and constraints based on ML profile.
- 3 Data management module retrieves the data from the data store and performs necessary processing based on the ML profile.
- 4 Once pre-processing is complete, data management module returns the pre-processed data to the pipeline management module or stores it as specified by the pipeline management module.

8.2.3 Model training flow

Figure 13 shows the MLFO-enabled model training/retraining.

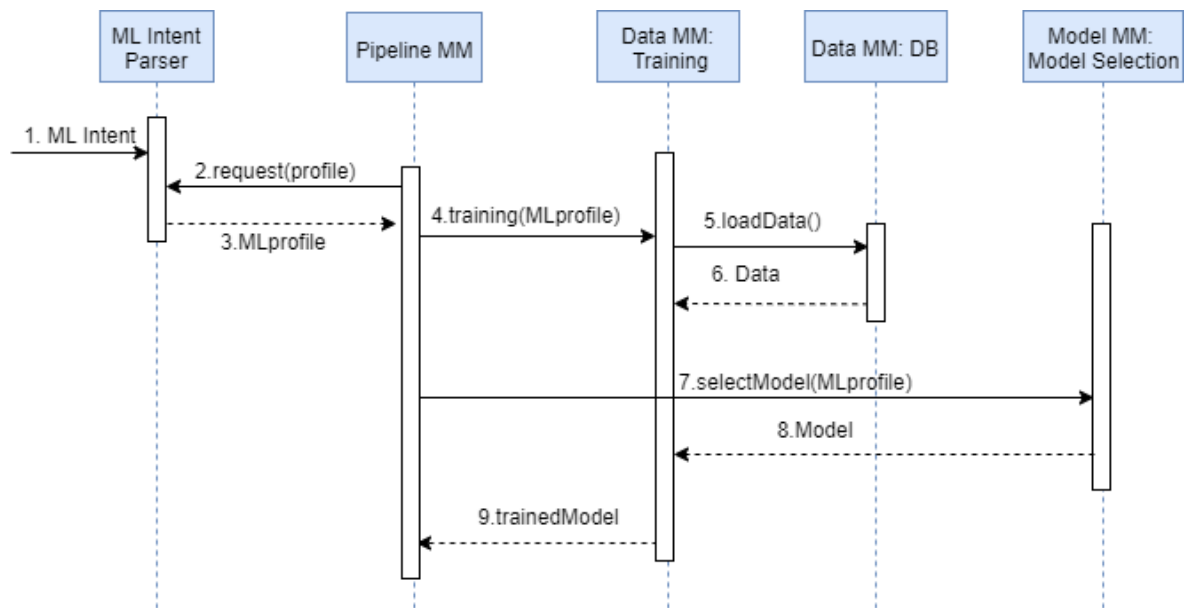


Figure 13 – MLFO-enabled model training/retraining

The steps for the MLFO-enabled training/retraining in Figure 13 are described below:

- 1 Pipeline management module retrieves ML profile of the application from the ML intent parser.
- 2 Pipeline management module issues a training/retraining request to the model management module and specifies the ML profile.
- 3 Model management module selects the appropriate model from the model selection/reselection module.
- 4 Model management module retrieves the data from the data store and performs necessary training/retraining based on the ML profile.
- 5 Once training/retraining is complete, model management module returns the trained model to the pipeline management module or deploys as specified by the pipeline management module.

8.2.4 Model selection

Figure 14 illustrates the MLFO-enabled data collection.

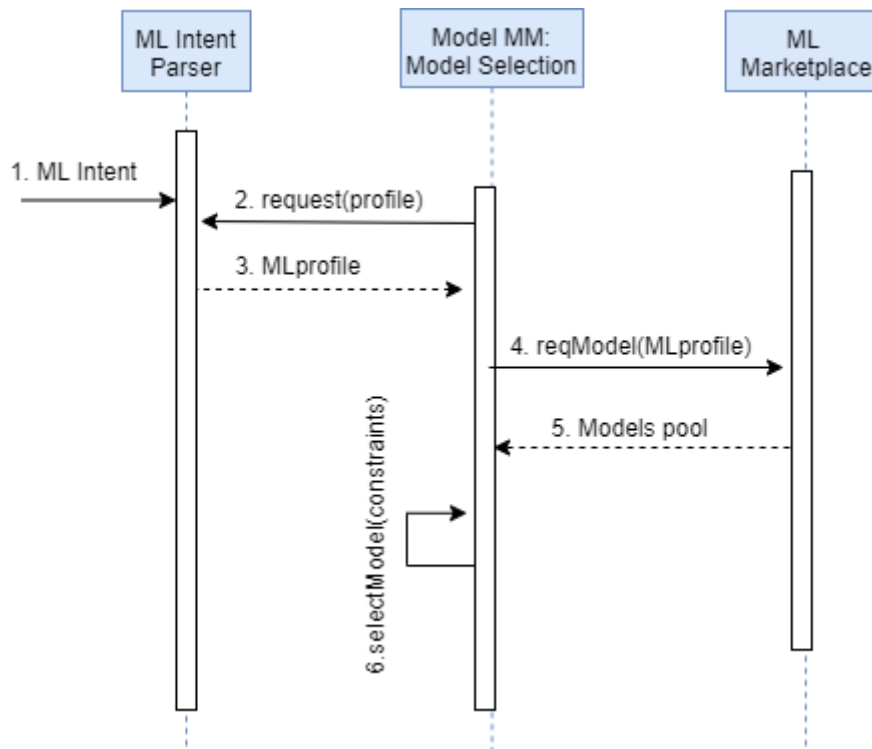


Figure 14 – MLFO-enabled data collection

The steps for the MLFO-enabled model selection/reselection in coordination with the ML marketplace given in Figure 14 are described below:

- 1 Model management module retrieves ML profile of the application from the ML intent parser.
- 2 Model management module requests models from the ML marketplace.
- 3 ML marketplace returns pool of models based on ML profile.
- 4 Model management module selects/reselects appropriate model from the pool of models according to the constraints.

8.2.5 Model/data monitor and optimization flow

Figure 15 shows the serving engine-assisted closed-loop system.

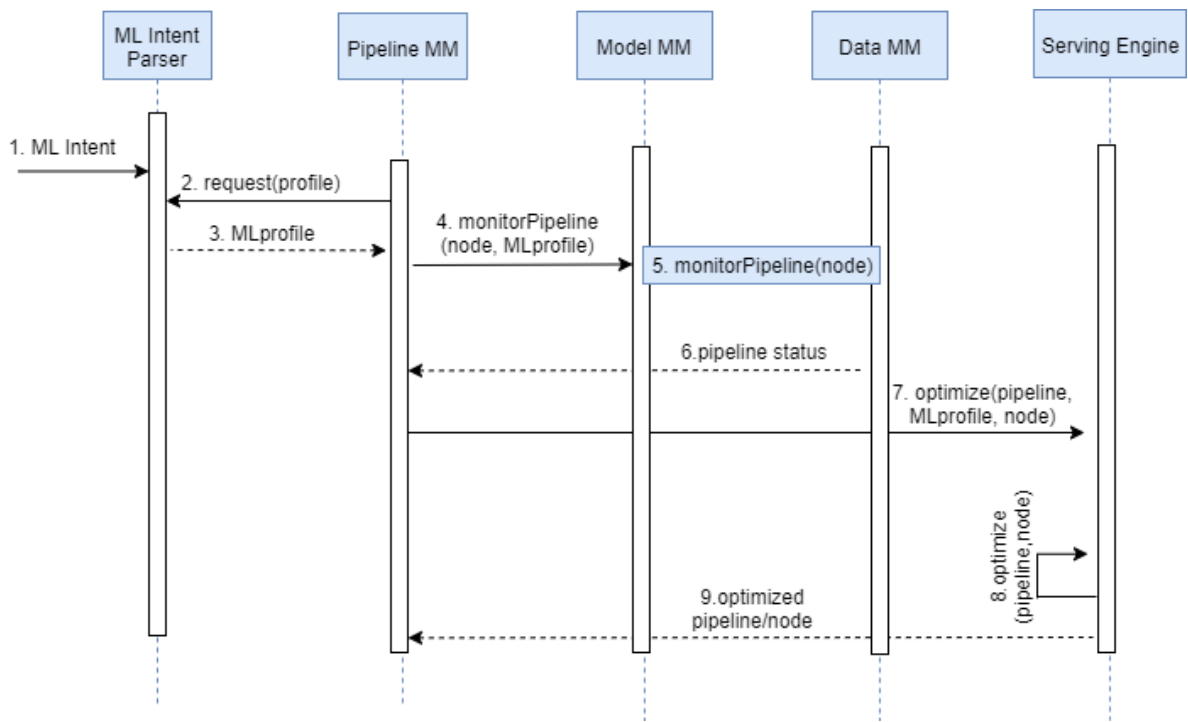


Figure 15 – Serving engine-assisted closed loop system

The steps for the closed loop system shown in Figure 15 are illustrated below:

- 1 Pipeline management module retrieves the ML profile from the intent parser to determine ML requirements and constraints.
- 2 ML pipeline management module initiates pipeline monitoring with the help of the ML profile across the model management and data management modules. This monitoring can be enabled on a single, or multiple, nodes of the pipeline. Certain statistics are collected during this monitoring, e.g., model accuracy and latency, etc.
- 3 Based on the performance status of the pipeline, the pipeline management module coordinates with the serving engine.
- 4 The serving engine optimizes the pipeline/node performance based on the ML profile, e.g., KPIs and other constraints. In certain cases, the serving engine can fall back to an older model to optimize the model performance.
- 5 The serving engine returns the optimized pipeline/node to the pipeline management module which can be deployed or published.

8.2.6 Model deployment

The procedure for the MLFO-enabled model deployment shown in Figure 16 is illustrated below:

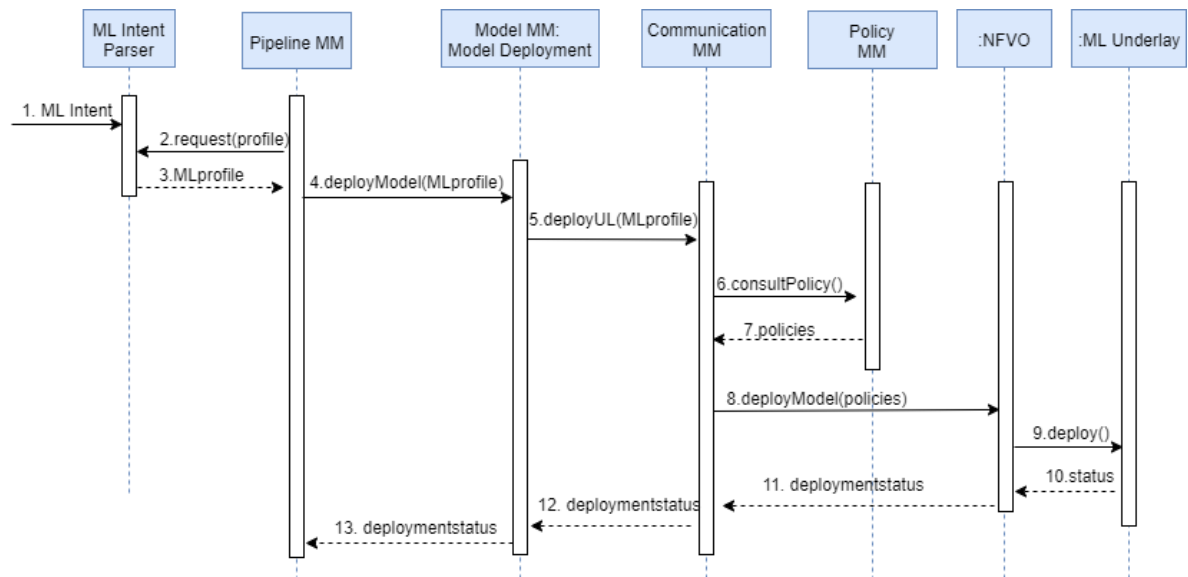


Figure 16 – MLFO-enabled model deployment

- 1 Pipeline management module queries the ML profile from the intent parser.
- 2 Based on the requirements specified in the ML profile, pipeline management module issues a deployModel() request to the model management module.
- 3 Model management module interfaces with the communication management module to deploy the model.
- 4 Communication management module consults the operator-specified policies from the policy management module.
- 5 Communication management module interfaces with the NFVO to deploy the model based on the policies.
- 6 NFVO deploys the model on the ML underlay.
- 7 NFVO returns the status of the ML deployment to the communication management module.
- 8 Communication management module informs the ML pipeline management module of the deployment status.

8.2.7 ML subprocedure: ML pipeline creation

Figure 17 illustrates the sequence of steps and can be triggered by the data handling framework, etc.

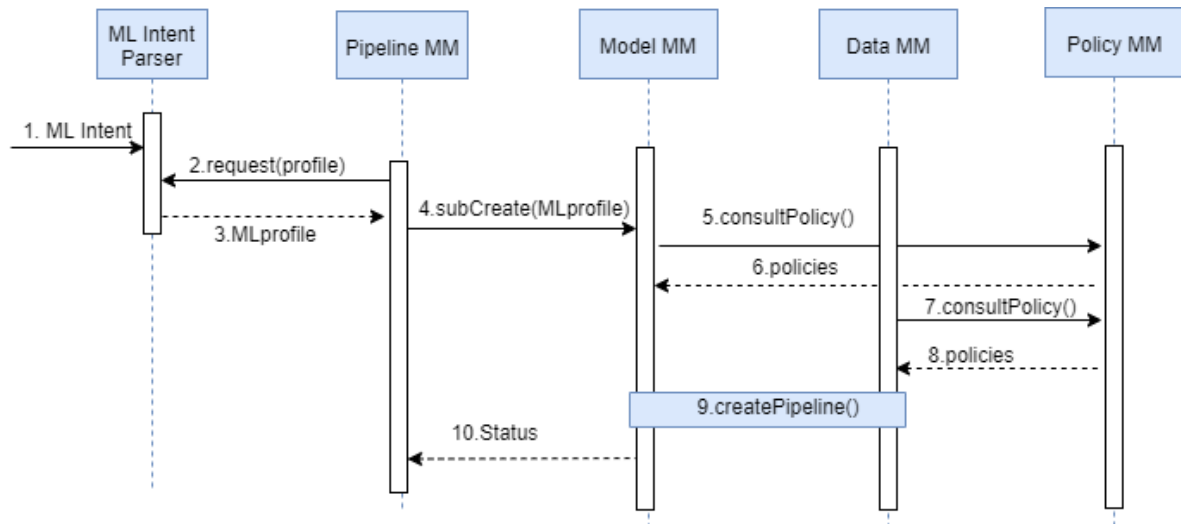


Figure 17 – Sub-procedure creation

- 1 Pipeline management module queries the ML profile from the intent parser for the subprocedure creation.
- 2 Based on the requirements specified in the ML profile, pipeline management module issues a subCreate() request to the model management and data management modules.
- 3 Model management and data management modules interface with the policy management module to consult the policies.
- 4 Based on the policies, both these modules create an ML pipeline to facilitate the subprocedure creation request.

8.2.8 ML sub-procedure: ML pipeline modification

The sequence diagram given in Figure 18 illustrates the sequence of steps required to facilitate a sub-procedure modification request:

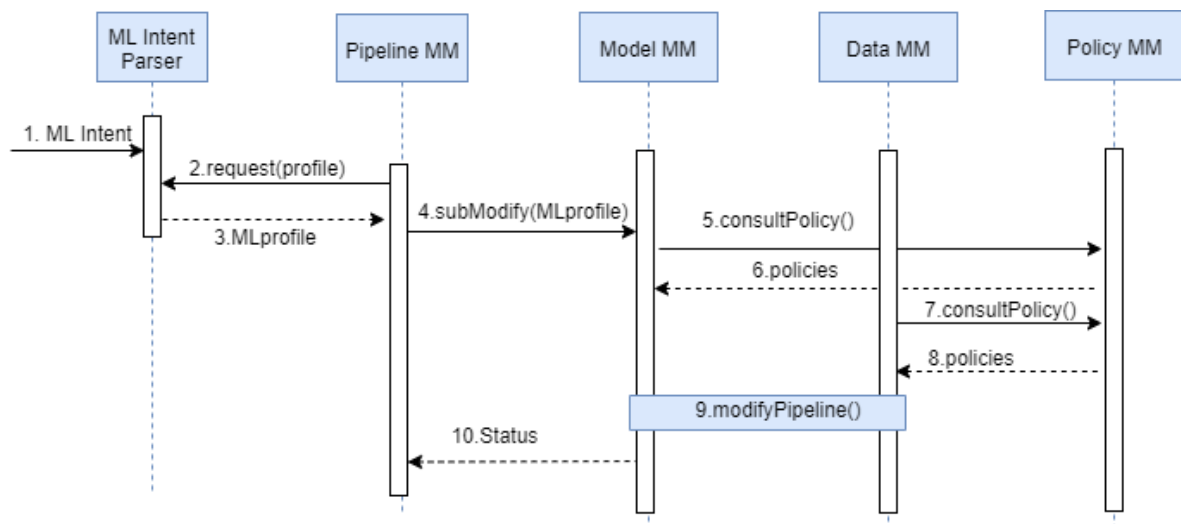


Figure 18 – Sub-procedure modification

- 1 Pipeline management module queries the ML profile from the intent parser for the subprocedure modification.
- 2 Based on the requirements specified in the ML profile, pipeline management module issues a subModify() request to the model management and data management modules.

- 3 Model management and data management modules interface with the policy management module to consult the policies.
- 4 Based on the policies, both these modules modify the ML pipeline.

8.2.9 ML subprocedure: ML pipeline monitoring

The sequence diagram in Figure 19 illustrates the sequence of steps required to facilitate a subprocedure monitoring request:

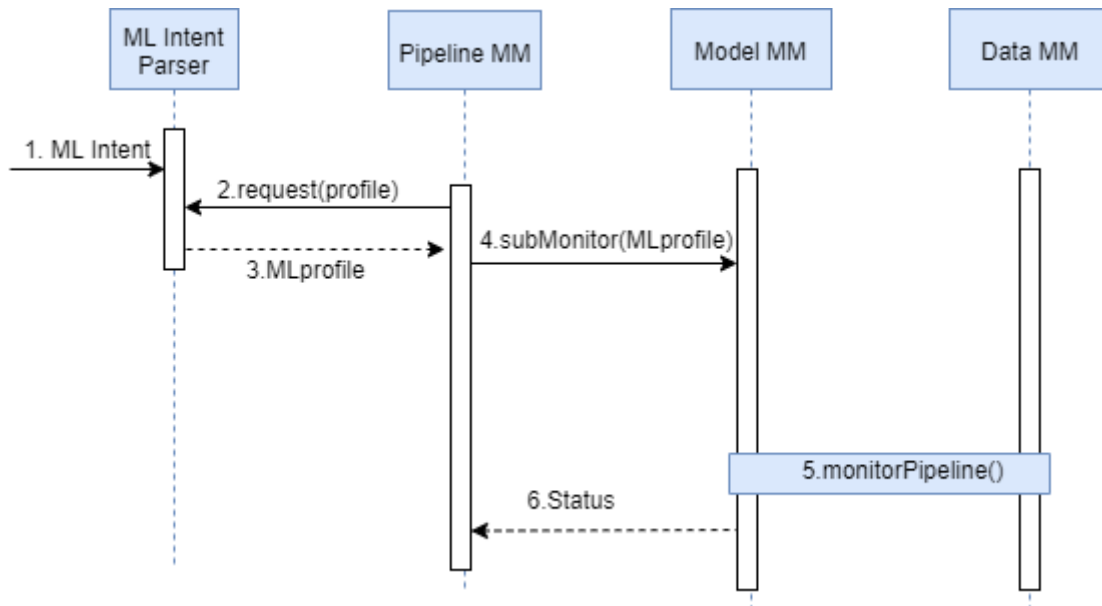


Figure 19 – Subprocedure monitoring

- 1 Pipeline management module queries the ML profile from the intent parser for the subprocedure monitoring.
- 2 Based on the requirements specified in the ML profile, pipeline management module issues a subMonitor() request to the model management and data management modules.
- 3 Model management and data management modules return the status of the ML pipeline.

8.2.10 Policy management

The sequence diagram in Figure 20 shows the sequence of steps required to facilitate provisioning and update of operator policies.

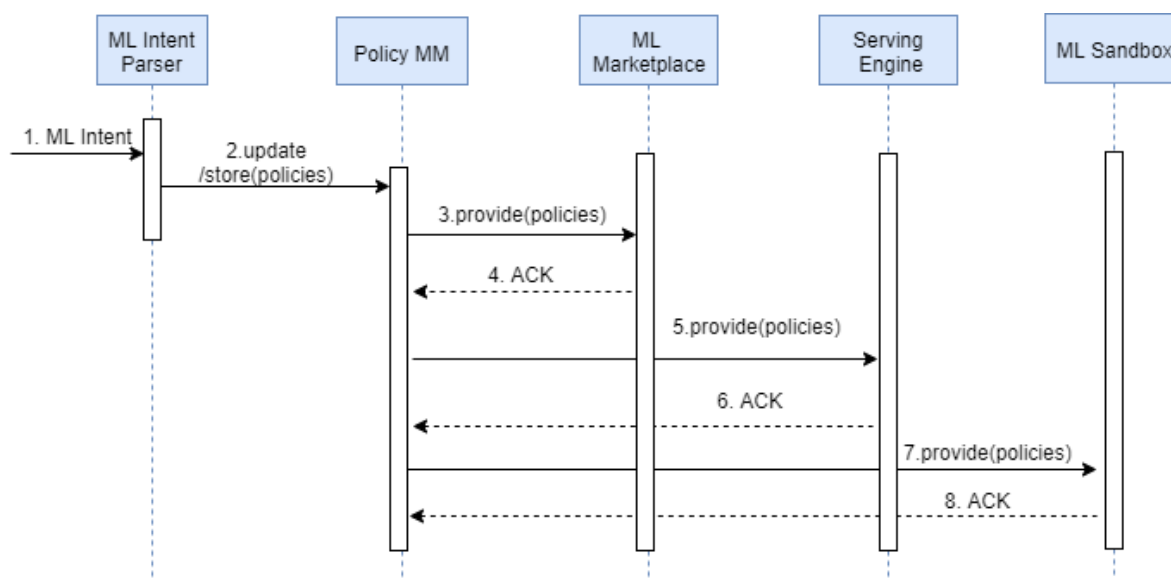


Figure 20 – Policy management

- 1 Operator specifies the policies by using a declarative mechanism, i.e., intent. These policies can include security and privacy requirements of operators.
- 2 The specified policies are stored or updated in the policy management module.
- 3 These policies are provided to the trusted parties including ML marketplace, serving engine, and ML sandbox.
- 4 These policies can be requested from the policy management module.

8.2.11 NFVO interface

The procedure in Figure 21 illustrates the sequence of steps required by the MLFO to interface with the NFVO to create a ML pipeline.

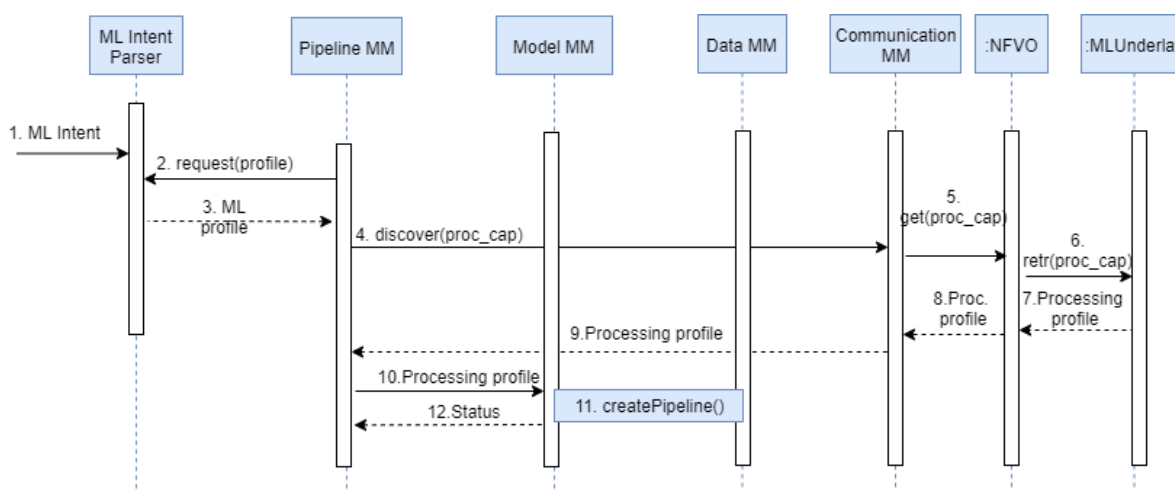


Figure 21 – NFVO interface

- 1 The pipeline management module retrieves the application requirements and constraints in the form of ML profile.
- 2 It interfaces with the communication management module to discover the processing or hardware capabilities of the ML underlay.

- 3 The communication MM interfaces with the NFVO to discover the processing or hardware capabilities of the ML underlay.
- 4 NFVO retrieves processing or hardware acceleration capabilities of the ML underlay.
- 5 NFVO returns these capabilities back to the communication management module in form of processing profile.
- 6 The communication management module returns these capabilities to the pipeline management module.
- 7 The pipeline management module creates an ML pipeline across the model management and data management module based on the processing profile for resource allocation.

8.2.12 Management of static ML profile

Figure 22 presents the sequence of steps required by the MLFO to create the static component of ML profile. The steps are given as follows:

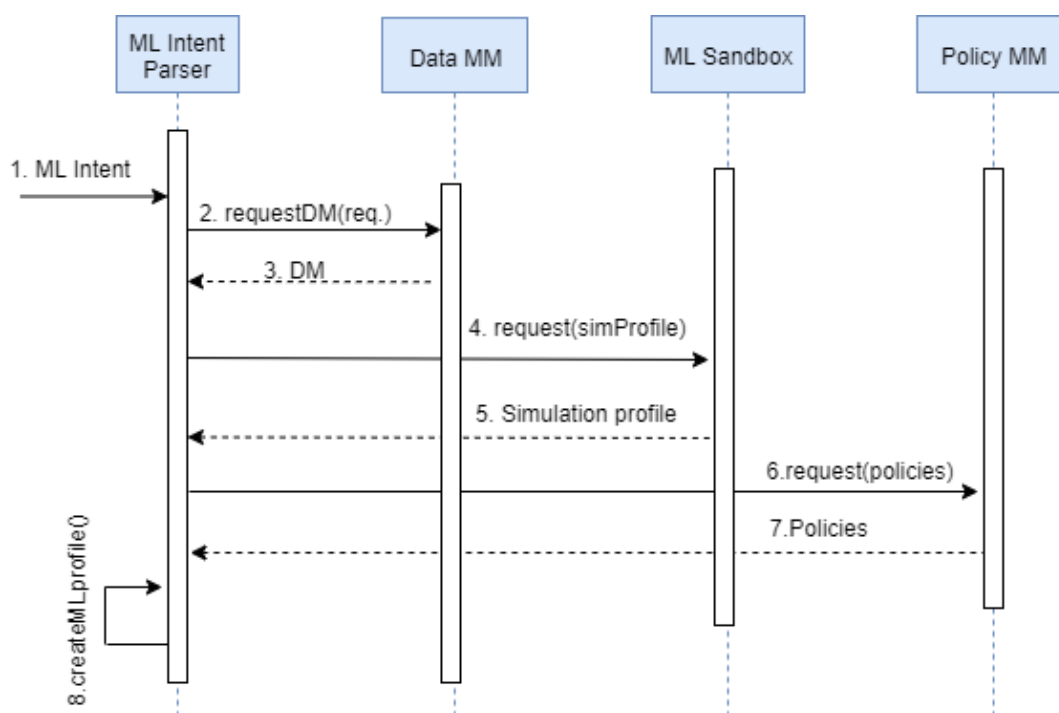


Figure 22 – Management of static ML profile

- 1 ML intent parser parses the requirements and constraints specified in the intent.
- 2 It requests a data model from the data management module according to the requirements.
- 3 Data MM returns the DM based on the requirements.
- 4 Intent parser requests simulation profile from the ML sandbox.
- 5 ML sandbox returns simulation profile according to the requirements of the ML application.
- 6 Intent parser requests operator-specified KPIs or other relevant policies from the policy MM.
- 7 The policy MM returns the requested policies.
- 8 ML intent parser creates an ML profile based on requirements, constraints, data model, simulation profile, and policies.

8.2.13 Management of dynamic ML profile (support for distributed instance of MLFO)

The management of the dynamic component of the ML profile by the MLFO is shown in Figure 23 and is illustrated below:

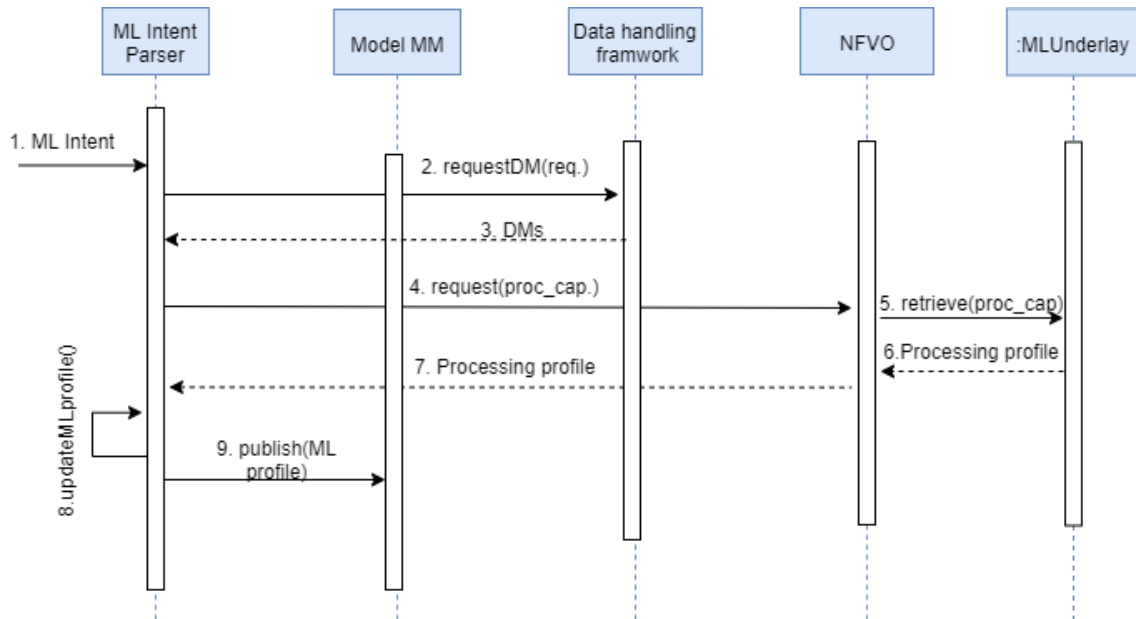


Figure 23 – Management of dynamic ML profile

- 1 ML intent parser requests the appropriate data models from the data handling framework based on ML application requirements.
- 2 Data handling framework returns data models based on application requirements.
- 3 ML intent parser enquires the processing capabilities of ML underlay with an interface to NFVO.
- 4 NFVO retrieves ML underlay processing capabilities.
- 5 NFVO returns it to the ML intent parser.
- 6 ML intent parser updates the profile based on requirements, constraints, ML underlay processing capabilities.

NOTE 1 – The dynamically updated ML profile can be used to create and assist with the distributed operation of MLFO in coordination with the NFVO.

- 7 ML intent parser publishes the ML profile with the help of model MM.

NOTE 2 – The ML profile can be subscribed by trusted parties, e.g., ML sandbox, Serving engine, and ML marketplace.

8.2.14 Pipeline testing

Figure 24 illustrates the steps to test the pipeline in coordination with the ML sandbox. The steps are given below:

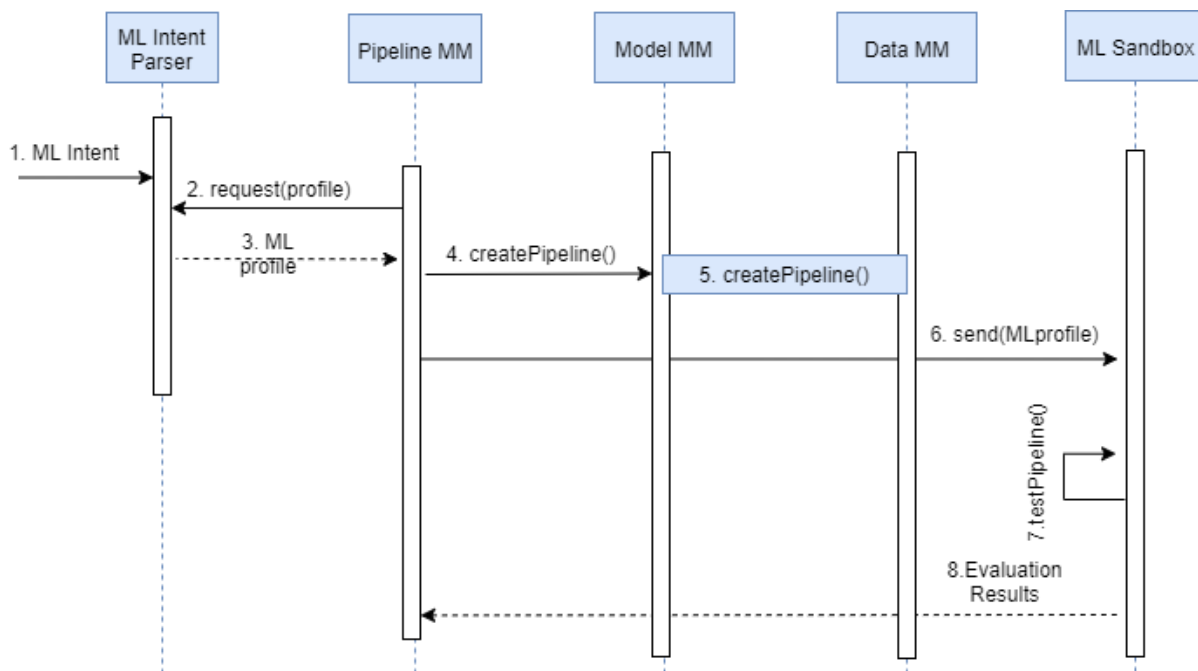


Figure 24 – ML pipeline testing

- 1 Pipeline management module requests the ML profile from the ML intent parser.
- 2 ML intent parser returns ML profile which includes constraints and ML application requirements.
- 3 Pipeline management module creates an ML pipeline across the model management and data management module.
- 4 Pipeline management module provides the ML profile to the ML sandbox to prepare a simulation profile.
- 5 ML sandbox tests the pipeline in the sandbox based on the simulation profile.
NOTE – ML pipeline testing in the ML sandbox can be in parallel based on the sandbox processing capabilities and ML application constraints (given in the ML profile).
- 6 ML sandbox returns the results of pipeline evaluation to the pipeline management module which can be used for the pipeline deployment decision.

8.2.15 ML sandbox-assisted model training/retraining

The sequence diagram given in Figure 25 illustrates the sequence of steps required for the model training or retraining with the help of ML sandbox. The steps are as follows:

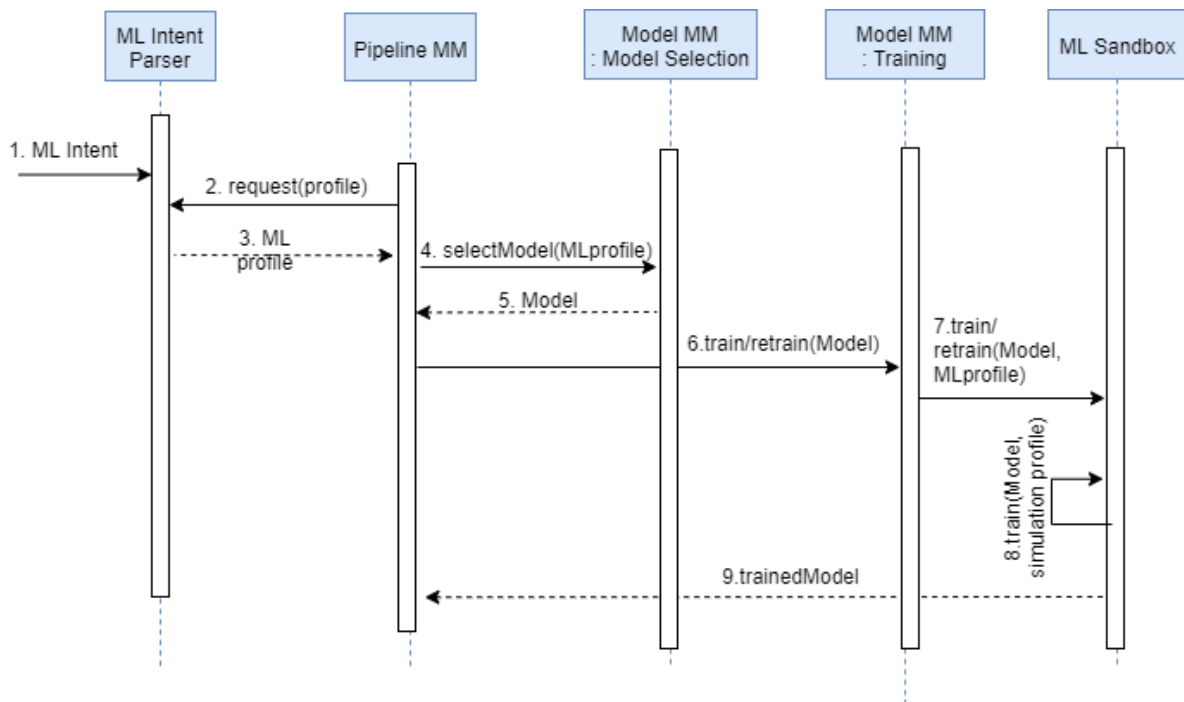


Figure 25 – ML sandbox-assisted model training/retraining

- 1 Pipeline management module requests the ML profile from the ML intent parser.
- 2 ML intent parser returns the ML profile which includes constraints and ML application requirements.
- 3 Pipeline management module selects a model from the model selection module. The model is selected from the pool of available models or is requested from the ML marketplace according to the ML profile.
- 4 Pipeline management invokes training of the model through the model management module by passing the ML profile.
- 5 Model management module passes the model and ML profile to the ML sandbox.
- 6 ML sandbox maps the ML profile to the simulated ML profile and trains/retrains the model. This training or retraining can be carried out in parallel based on ML application constraints and simulated processing capabilities.
- 7 ML sandbox returns the trained model to the ML pipeline module, which can be deployed or published.
NOTE – ML pipeline testing in the ML sandbox can be carried out in parallel based on the sandbox processing capabilities and ML application constraints (given in the ML profile).
- 8 ML sandbox returns the results of pipeline evaluation to the pipeline management module, which can be used for the pipeline deployment decision.

8.2.16 Model update in marketplace

The steps for the model update to the ML marketplace are illustrated in Figure 26 and given below:

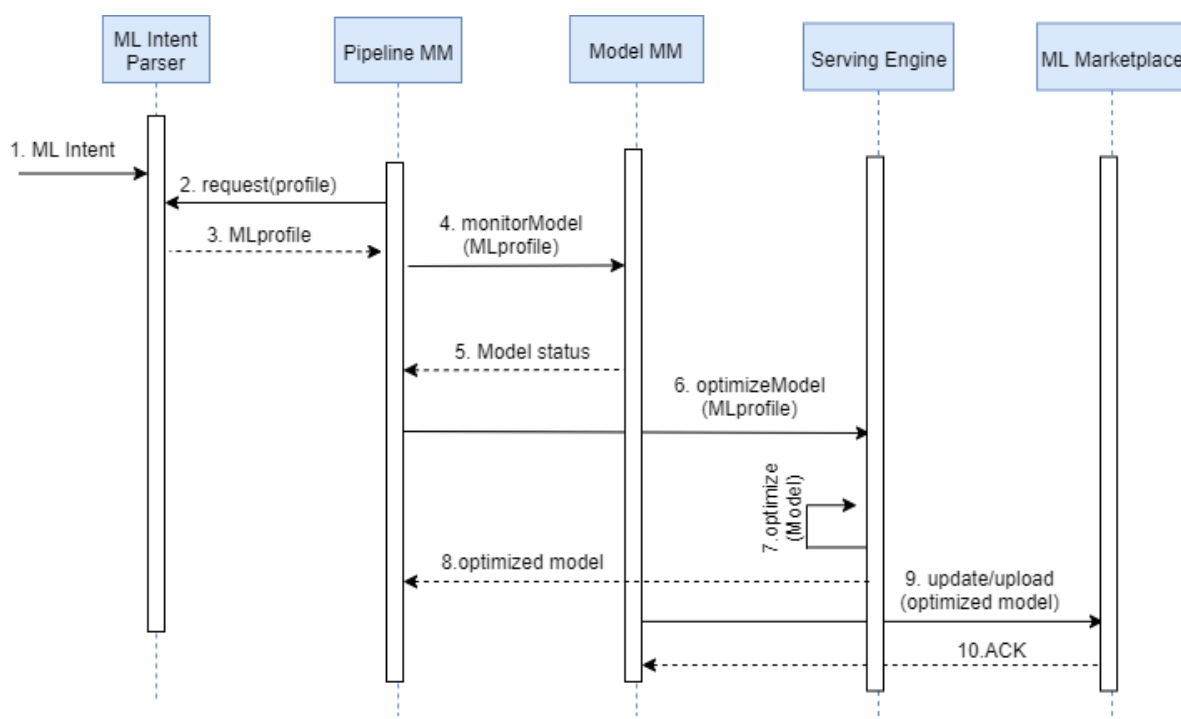


Figure 26 – Model update on ML marketplace

- 1 Pipeline management module monitors the performance of the deployed model through model management module.
- 2 Pipeline management module retrieves the ML profile from the intent parser to determine ML requirements and constraints.
- 3 ML pipeline management module initiates model monitoring with the help of ML profile. Certain statistics are collected during this monitoring, e.g., model accuracy and latency, etc.
- 4 Based on the performance status of the model, pipeline management module coordinates with the serving engine to optimize the model performance.
- 5 The serving engine optimizes the model performance based on ML profile, e.g., KPIs and other constraints. In certain cases, serving engine can fall back to an older model to optimize the model performance.
- 6 The serving engine returns the optimized model to the pipeline management module.
- 7 The pipeline management module uploads or updates the optimized model with a version number and ML profile to the ML marketplace.

8.2.17 Asynchronous operation execution by MLFO

Figure 27 shows the sequence of steps for the asynchronous operation execution by the MLFO. The steps are as below:

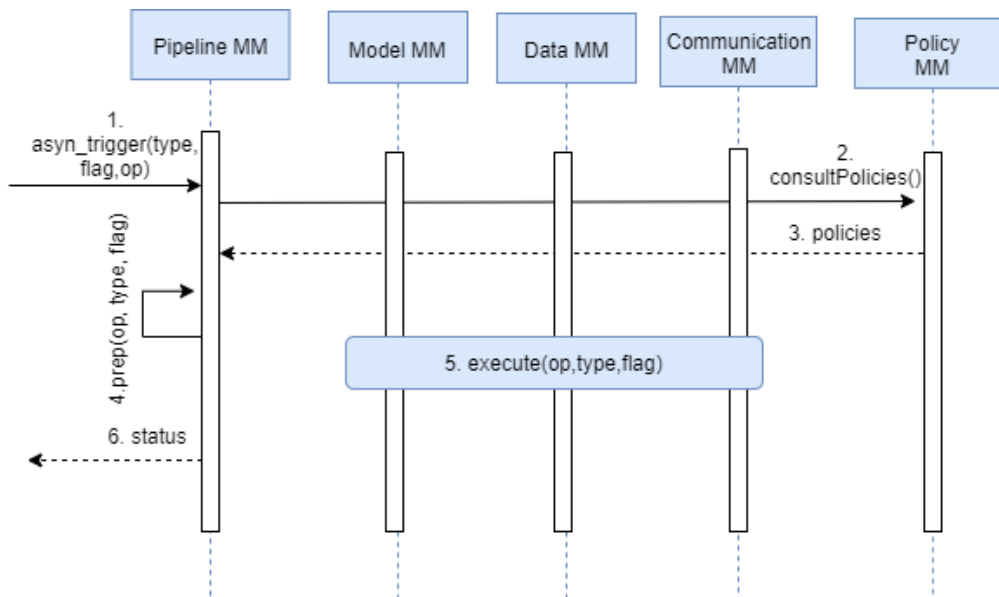


Figure 27 – Execution of asynchronous triggers by MLFO

- 1 ML pipeline management module receives an asynchronous operation request from the trusted entities, e.g., data handling framework, serving engine, or ML sandbox.

NOTE 1 – These asynchronous requests can include threshold-based event monitoring, data model mapping, preconfigured events, metering events, security violation events, data-related policy violation events, failure events including model or simulations, load thresholds, or data lifetime expiry events.

- 2 ML pipeline management module consults the policy management module to prepare execution of these asynchronous events.

- 3 Policy management module returns the policies relevant to the execution of these asynchronous operations.

- 4 Pipeline management module prepares the asynchronous operation.

- 5 Pipeline management module executes the asynchronous operation with the assistance of data management, model management, and communication management module.

NOTE 2 – The asynchronous operation can also include stop, start, instantiate data sources/collector, or expunge data from data store.

- 6 Pipeline management module returns the status of the asynchronous operation execution to the requested entities.

8.2.18 MLFO-triggered synchronous operations

Figure 28 shows the sequence of steps for the MLFO-triggered asynchronous operations. These steps are illustrated below:

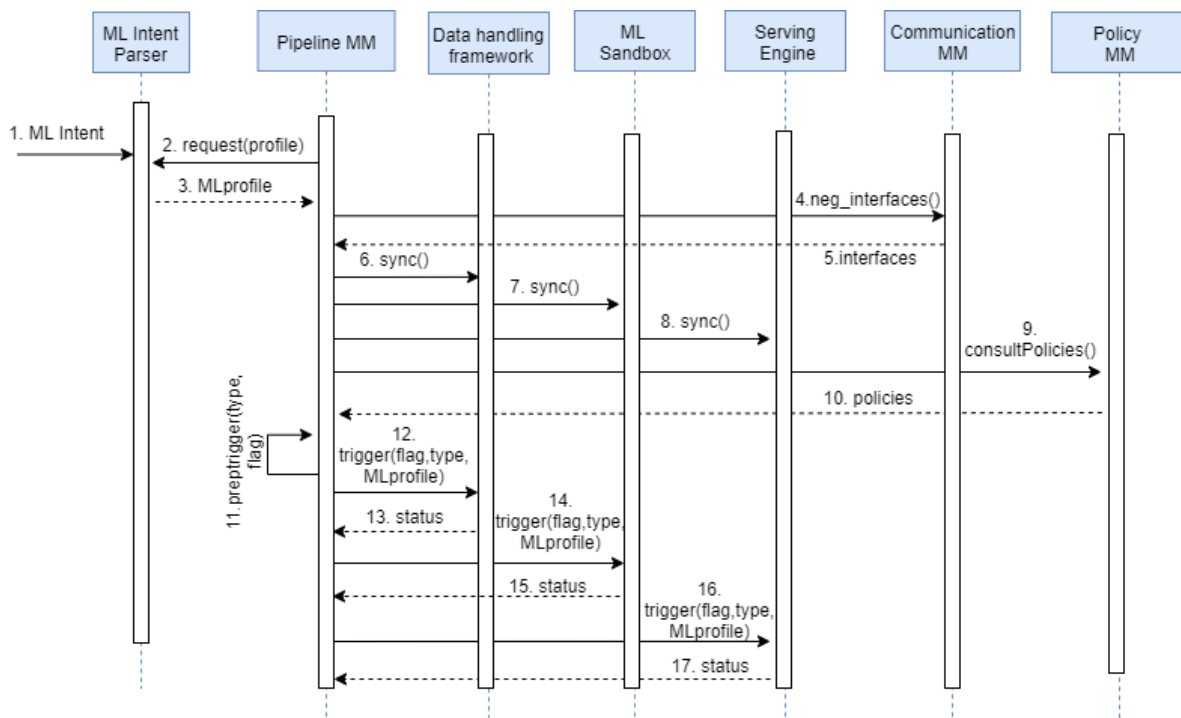


Figure 28 – MLFO-triggered synchronous operations

- 1 ML intent parser receives an ML intent.
- 2 Pipeline management module retrieves ML profile from the intent parser which includes ML use case requirements and constraints.
- 3 Pipeline management module interacts with the communication management module to negotiate required interfaces to interact with the trusted entities, e.g., ML sandbox, serving engine, or data handling framework.

NOTE 1– Interface negotiation achieved through the communication management module includes data formats or protocols, ML profile, ML model metadata, and policies.

NOTE 2 – Time synchronization is required prior to the trigger and involves MLFO and the communication management module, which interfaces with the NFVO.

- 4 Pipeline management module synchronizes with the ML sandbox.
- 5 Pipeline management module synchronizes with the data handling framework.
- 6 It synchronizes with the serving engine.
- 7 Pipeline management module consults relevant policies from the policy management module for the synchronous trigger preparation.

NOTE 3 – Dynamic operations are triggered on the ML pipeline nodes in the sandbox and ML underlay.

- 8 It prepares the synchronous trigger based on the ML profile.
- 9 It initiates a trigger to the required entity, e.g., ML sandbox, serving engine, or data handling framework for the desired operation.

NOTE 4 – These operations can include stop, start, format conversions and storage/saving of models, storage/saving of data, configuration/parameter updates, triggering of workflows on the ML sandbox/serving engine, data collection, pre-processing tasks, data storage and retention, transfer of ML metadata, operations on ML knowledge, optimization of overall model and data management, configuration of monitoring thresholds, metering configuration, security/keys config, energy

optimization, resource management and optimization, failover, and load balancing for resilient operation, etc.

- 10 The entities, e.g., ML sandbox, serving engine, and data handling framework execute these operations and return the status of the operation execution to the MLFO.

NOTE 5 – The operation execution in trusted entities is hidden from the MLFO, however, a status or confirmation is required.

Appendix I

Scenarios for machine learning function orchestration

Usage scenario: use case 4.1: Traffic classification

- MLFO enables specification of the use case to collect real-time traffic data.
- MLFO may specify pre-processing tasks such as labelling of data according to the application and protocol type, which may be used to train the model. It may include balancing of inputs to avoid bias.
- MLFO may also specify data storage and its management to store collected data.
- MLFO should be able to enable and configure sources of data based on the specification of the ML application.

NOTE 1 – E.g., the data may include network information, the transport layer payload, and some relevant protocol fields in the data.

- MLFO should enable selection of different types of models for an ML application based on the characteristics of ML application, data, and other constraints.

NOTE 2 – For example, CNNs, RNNs, or hybrid models may be selected for the use case.

- MLFO should support monitoring to view real-time status of data, collectors, packets, flows, and ML models.
- MLFO should support configuration of data models according to the source and the type of model used.
- MLFO should support update and retraining of ML models with minimal impact on the performance of the ML application.
- MLFO should support security requirements for data handling.

NOTE 3 – MLFO must be aware of any encryption/decryption that has been performed on the data and this information must be specified within the ML intent. The above information in the ML intent will be used by the MLFO to configure the data sources, accordingly.

Usage scenario: use case 4.2: RAN-assisted transmission control protocol window optimization

- MLFO should enable the collection of data required for the use case in real-time.
NOTE 1 – This data may include MAC and PDCP layer data, MAC data rate, PDCP buffer size, BS load, service type, and status. The output data may include TCP window decision model and TCP packet transmission rate.
- MLFO may enable specific learning mechanisms e.g., supervised learning model, required for the use case along with appropriate training mechanisms e.g., offline training of the corresponding model.
- Interface between the MLFO and the management plane of the ML underlay may be used to understand the data and service capabilities of the ML underlay, e.g., network topology, metadata of data models, and service orchestration capabilities provided.

NOTE 2 – Such metadata may be used for the interoperability in data handling [b-ML-Schema-Gustavo] by deriving the schema information which is supported by the ML underlay. MLFO understands the technology-agnostic data model specified in the ML intent, queries the management plane for the corresponding technology-specific interface, and then configures the DBr-CP accordingly.

NOTE 3 – DBr-CP (data broker control plane (CP)) facilitates the use of correct APIs towards the ML underlay network functions and maps to the incoming data to data model used in the ML overlay (if needed) [b-ML5G-148].

- Based on the capabilities of the NF or the AF, MLFO may enable specific interface towards the NF or the application function (AF) for ML data collection or interface for ML output-based configurations.

NOTE 4 – This may be used for example for configuring TCP window in the AF. This may be implemented in the form of an interface plugin. The interface plugin may be specified in the ML intent as an input to the MLFO.

Usage scenario: use case 4.3: ML-based mobility pattern prediction

- MLFO enables specification of the use case for real time data collection from different levels of a network. Application-specific latency constraints may be specified in the ML intent.

NOTE 1 – Some of the data sources may be relevant to position information, e.g., GPS data.

NOTE 2 – This data may include network performance data, cell status query, sleep/wake-up parameters, context data, environmental information, configuration information, and other current network interactions.

- MLFO selects a deep learning or hybrid ML models, e.g., RNN, reinforcement learning, or deep reinforcement learning based on the requirements of the use case, e.g., for local predictions of a user trajectory.
- It also selects and specifies an appropriate deep learning model, e.g., LSTM for network-wide global predictions of a user trajectory based on local predictions.
- MLFO may enable a caching module to store data and updated results of local and global predictions.

NOTE 3 – For example, this may be used to store a user's predicted trajectory.

- MLFO configures an execution module to generate configuration scripts. This script is communicated to the BS and other management subsystems. e.g., VNFM, NMS. This may form an ML output.

NOTE 4 – An application of ML output may be used for certain optimizations, e.g., handovers while minimizing the number of dropped calls.

- MLFO may enable training/re-training of deep learning or hybrid ML models based on data to improve accuracy and effectiveness of selected models both for the local and global predictions.

NOTE 5 – The training data may include historical information such as reference signal received power/quality (RSRP/RSRQ) or real time data.

NOTE 6 – Other historical information which can be used to train the model includes network load, throughput, interruption time, awareness and prediction of user equipment (UE) position, moving direction, and speed, etc.

- MLFO specifies and/or selects an appropriate ML model.

NOTE 7 – This selection may be based on the different levels of data. For example, UE level ML model may be used for UE level predictions.

NOTE 8 – RNNs can predict mobility pattern predictions better compared to traditional ANNs (e.g., FNNs) as they are able to remember important information about the received input due to their small memory. MLFO may enable simulated data sources.

- MLFO shall enable configuration of data privacy policies for mobility data from individual users and user trajectory prediction of individual users.

NOTE 9 – MLFO may chain and configure the sources and sinks based on the data privacy policies.

- MLFO may enable application of ML output at different levels in the network.
NOTE 10 – e.g., high level mobile edge computing (MEC) and low level MEC may be examples where such ML output may be applied.

Usage scenario: use case 4.4: Load balance and cell splitting/merging

- MLFO enables the specification of the use case to collect real-time data from the current and neighbouring cells.
NOTE 1 – This information includes number of UEs, RB utilization, PDCP package number, number of RRC connection requests, and active E-RAB number of current and neighbor cells.
- MLFO configures storage of the collected data in an appropriate ML data store for pre-processing.
- MLFO enables pre-processing and appropriate data analytics including data labelling.
- MLFO selects ML model to realize the ML application. It also uses appropriate training mechanism to train the model.
NOTE 2 – ML model may predict a feasible map between UEs and cells, and interface with the management plane to balance number of UEs connected to different BS.
NOTE 3 – ML model may analyse and predict the status of a cell. Based on the prediction, ML output is used in the management plane to take further action, i.e., cell splitting or cell merging. The ML output may also be used to select appropriate configuration parameters to execute cell splitting/merging process.
- MLFO may enable application-specific learning mechanisms, e.g., supervised learning or hybrid. It also selects appropriate training mechanisms, e.g., offline training to train the model.
NOTE 4 – Selection of learning mechanisms may be based on the characteristics of the data in the ML data store.
- MLFO may enable application of ML output in the NMS.

Usage scenario: use case 4.5: Cognitive heterogeneous networks

- MLFO enables the collection of data at various levels in the network using external devices.
NOTE 1 – This data may include radio access network (RAN) data, core network data, or GPS data. The radio access network data may include location information, radio conditions, resource allocation information, connection information, traffic type, service target indicator, cell/node resource utilization, cell/node KPIs, i.e., throughput, drops, accessibility, and power. The core network data may include core network KPIs, core traffic figures, and network utilization KPIs, CDRs, and log files.
NOTE 2 – Examples of external devices are IoT modules, sensors, drones, and application servers, etc.
- MLFO may enable pre-processing and labelling of the data.
NOTE 3 – This may be to assist network to be more aware and conscious about mobile network problems, user behaviours, and environmental aspects.
- MLFO may apply the ML output to different levels of the network and to external devices.
NOTE 4 – The ML output may be applied for example to different levels of management subsystems and orchestrators.
- MLFO interface with the management plane of ML underlay may be used to understand the services of central and distributed orchestrators to modify network parameters and set them automatically.

NOTE 5 – this understanding may be used to assist network aware operations, e.g., platform aware placement decisions for ML models.

- MLFO selects an appropriate supervised or unsupervised, or hybrid models to optimize different operations of the cognitive heterogeneous networks.

NOTE 6 – E.g., RNN, Fuzzy neural network, unsupervised learning technique or a hybrid model.

NOTE 7 – ML model may optimize quality of service (QoS) provisioning to various services.

NOTE 8 – The ML model also can enable dynamic adjustment of transmit power level and dynamically turn it on/off to optimize connection and QoS performance.

- MLFO enables specification of a training mechanism to train the ML models, e.g., offline or online according to the constraints in the ML intent.

NOTE 9 – MLFO can select platform-aware training, e.g., based on accelerators.

- MLFO understands the ML application specified in the ML intent, queries the management plane for technology-specific interfaces for appropriate source and sink [ITU-T Y.3172].
- MLFO may export trained model or training data across different levels of network [ITU-T Y.3172].
- MLFO selects the interface between nodes in the ML pipeline based on the specification given in the ML intent.

NOTE 10 – API may be used, for example, or specific low-latency interface may be selected and used, based on the capabilities of the ML underlay and the ML pipeline nodes.

Usage scenario: use case 4.6: ML-based self-organizing networks

- MLFO enables the collection of data and ML output configuration across various types of ML underlay networks.

NOTE 1 – ML underlay networks may include different wireless technologies of different generation, i.e., 3G, 4G, etc.

- MLFO enables ML pipeline across ML underlays from different vendors.
- MLFO enables ML pipeline across different levels of ML underlays.
- MLFO enables the collection of data related to network and enables the storage of this data in an appropriate ML data store for later use.

NOTE 2 – This data may include radio access network and core network data. The radio access network data may include location information, radio conditions, resource allocation information, connection information, traffic type, service target indicator, cell/node resource utilization, cell/node KPIs, i.e., throughput, drops, accessibility, and power. The core network data may include core network KPIs, core traffic figures, and network utilization KPIs.

- MLFO interfaces to various ML pipeline nodes and ML data stores based on latency requirements of the ML application.

NOTE 3 – A low latency and shared DB interface may be selected for interface between ML pipeline nodes.

- MLFO may enable pre-processing to label the data.
- MLFO may allow specification and training of the ML model based on the ML application.

NOTE 4 – E.g., a fuzzy Q-learning [b-pervez-Jaber17] in each eNB to predict performances or behaviours of the network. Based on the prediction of behaviours, the model optimizes network parameters including coverage, capacity, handover, and interference.

NOTE 5 – The learning mechanisms may be supervised learning model, transfer learning, reinforcement learning, or a hybrid model. MLFO selects an appropriate ML technique based on the needs of the ML application.

NOTE 6 – E.g., genetic algorithm [b-yang-Li18] for cell planning, coverage, and power adjustments. MLFO may enable an ML model for pattern identification or grouping.

NOTE 7 – E.g., Self-organizing Maps (SOMs) or Fuzzy C-means [b-Li-Zhang16] to identify patterns or group of nodes with similar characteristics based on the criteria defined in the ML intent.

- MLFO enables specification of an autonomy level to automatically configure parameters in the network.
- Interface between the MLFO and the management plane of ML underlay can be used for automatic configuration.
- MLFO specifies an appropriate learning mechanism, e.g., supervised learning to identify a fault, classify it, and takes the necessary actions to recover from it, e.g., removal of failures and automatic adjustment of parameters.

Usage scenario: use case 4.7: ML-based QoE optimization

- MLFO enables data collection from various parts of the network, including service requirements for user services. It may store data in a ML data store for analytics.

NOTE 1 – The important data which needs to be collected includes user experience and QoS/QoE feedback data. The user experience data may consist of rebuffering time, staying time, and blockage frequency. QoS/QoE feedback data may include additional information of latency and jitter.

- MLFO enables pre-processing on the data.

NOTE 2 – This may include refining raw network status data and to remove protocol dependent features.

- MLFO selects a feature selection mechanism to extract the relevant features from the data, e.g., rebuffering time, and blockage frequency from the feedback and user experiences.
- MLFO selects an ML model, e.g., supervised or hybrid, and train it by using preferable training approach.
- MLFO may also use a reinforcement learning approach to consider real-time network states and service quality feedback from the users to perform configurations to increase QoE optimizations.
- Based on the QoE score, MLFO interfaces with the management subsystem to configure the network within a given time-limit.
- MLFO enables user privacy and security for the data collected from the user.
- MLFO enables application of ML output to multiple levels of the network.

NOTE 3 – Examples of multiple levels in this case may be MEC and NMS.

Usage scenario: use case 4.8: Radio resource management for network slicing

- MLFO performs data collection at multiple levels for real-time data. It may store data in a ML data store.

NOTE 1 – The collected data may include slice level performance measurement parameters, slice level network performance measurements and alarm data, cell status query, user mobility and radio channel conditions data, and user service request, QoS and data monitoring.

- MLFO should also specify an ML model to predict resource allocation in the network.

NOTE 2 – e.g., CNN to predict the required allocation trajectory based on predicted per cell and per slice resource requirements.

- MLFO enables the application of ML output at various levels in the network.

NOTE 3 – e.g., NMS, CU, or distributed unit (DU) to optimize per slice performance with the help of optimal slice planning and appropriate slice radio resource scheduling.

- MLFO may use a policy to interface with the management plane to communicate with the RAN, network core elements, and network managers.
- MLFO may interface with a simulator to control and generate simulated events for training the model and it will minimize the impact on the network for such events.

NOTE 4 – This may include generation of rare events and it may include the use of a sandbox.

Usage scenario: use case 4.9: ML-based network management for industry 4.0

- The specification of ML application used as input to the MLFO should include the capabilities of device nodes to support APIs that enable setting/monitoring/measuring device parameters and network related functions.

NOTE 1 – This may include physical and link layer APIs, network formation/routing APIs, power and device state APIs.

NOTE 2 – This may include protocols to drive schedule delivery, network resource allocation, gateway association information, etc.

NOTE 3 – This may also include data formats and protocol specification/standardization to provide interoperability among devices from different manufacturers.

- Accuracy, latency, and functional requirements of a specific service may be input to the MLFO.
- Estimated additional overhead imposed by the collection of data for the ML application may be output from the MLFO based on the specification of the ML application and the topology.
- Timing requirements on data collection, ML processing, and delivery of the output to the network may be specified to the MLFO.
- MLFO enables resilience in the framework used in processing events for the ML application so that any failures in the framework do not impact the ML output.
- Security requirements of the data and messages should be specified to the MLFO.

Usage scenario: use case 4.10: ML-based correlations between transport KPIs and radio KPIs

- Continuous collection of performance measurements at different levels of network may be enabled by MLFO.

NOTE 1 – this may include radio and transport networks.

- ML application specification will include a granularity and frequency of collection of inputs from the data sources.
- ML application will specify any specific interfaces to be used towards the sink.

NOTE 2 – Active probing monitoring systems connected to the network elements may support specific APIs or interfaces for transport KPIs (e.g., FLR, delay and jitter).

NOTE 3 – Collecting and consolidating the data from all NMS's and active probes.

- Compute, storage and network requirements of the ML application may be taken into consideration while selecting the placement of the ML model.
- MLFO may enable data collection from real customer experience measurements from application specific readings (e.g., web browsing, video streaming, VoIP, online gaming, VR/AR, etc.). This data would be retrieved from crowdsourcing applications, or region-specific drive tests.

Usage scenario: use case 4.11: Retention and storage intelligence function

- MLFO should be provided with any input requirements from ML applications on data storage and any constraints on data storage.

NOTE 1 – Constraints may be in the form of size or latency.

NOTE 2 – Constraints may include the position of data e.g., core vs. edge.

NOTE 3 – This may be used to automatically store data in slow or fast storage.

- MLFO may interface with orchestration function in the ML underlay to understand the data storage capacity and capability.

NOTE 4– This may include the network capability in terms of latency for data transfer and memory capacity in terms of size or other storage characteristics.

- NOTE – This may include appropriate storage decisions within the network.
- MLFO may use policy inputs for storage and sharing decisions. This may be input for specific ML applications.

NOTE 5 – Such policy inputs may be used to comply with regulations.

NOTE 6 – Such policy inputs may be used to control the data retention.

NOTE 7 – Data retention may be controlled also by the resource limitations.

- MLFO may monitor the performance of ML models and optimize the location and delete/archive of data storage accordingly.

NOTE 8 – Relativity of the data may be used to perform proactive archiving/deletion decisions.

NOTE 9 – This may include identification of data that is technically obsolete to make a purge decision.

Usage scenario: use case 4.12: End-to-end network operation automation - Service design

- MLFO inputs the service requirements of the ML application and determines the network configurations based on it.

NOTE 1 – This may include the selection of a data model.

- MLFO monitors the performance of the ML application and modifies the network configurations accordingly.
- MLFO interfaces with the management plane to enable predicted network service deployment, mapped from the logical network requirements.

Usage scenario: use case 4.13: End-to-end network operation automation – Logical network design and deployment

- MLFO should enable the collection of large number of design intents and design results, e.g., TOSCA/OASIS.

NOTE 1 – These intents may consist of logical network descriptions with functional/non-functional requirements along with concrete design information in a machine-readable format.

NOTE 2 – The logical network descriptions may include logical network descriptions with concrete design information including components, mapping to physical nodes, configurations parameters, etc.

- MLFO should enable appropriate storage of these intents and design results in a ML data store.
- MLFO interfaces with the management plane to interface with a workflow generator to generate an executable workflow to safely reconfigure the current network infrastructure status to the desired status.
- MLFO understands the technology-agnostic data model specified in the intent, and interfaces with the management plane which in turn interfaces with an executor to allow deployment of the designed logical network on the physical network.

Usage scenario: use case 4.14: End-to-end network operation automation - Fault detection and recovery

- MLFO should have an appropriate mechanism to collect performance, events, topology, and configuration and failure data from the orchestrator of the ML underlay.
NOTE 1 – This data may include performance data related to network function (NF) and NFV infrastructure (NFVI) (e.g., CPU utilization, memory utilization, Disk I/O, etc.) described in a common data format. Other data may include traffic information.
NOTE 2 – This may include queries to the management plane for the corresponding technology-specific interface to perform fault detection, RCA, and automated recovery.
- MLFO monitors and analyses the performance of ML models.
NOTE 3 – Testing of ML models may be enabled based on simulated data.
NOTE 4 – RCA for failures in ML models may be performed based on collected data.
- MLFO should enable an appropriate generation mechanism for training data based either on test environment or previous knowledge data related to fault detection and recovery procedure.

Usage scenario: use case 4.15: End-to-end network operation automation - Network resource adaptation

- MLFO enables collection of resource information from NFs including control, service plane, and data plane information.
NOTE 1 – Control information may include information of control requests from an external controller or orchestrator. Service-plane information may include state data along with identification and configuration information, e.g., computational resources such as CPU, memory, and storage for network function virtualization (NFV) and SFC. State data in data-plane includes information about network resources, e.g., physical switch and link along with information of identifications and configurations.
NOTE 2 – This may be achieved using an interface with the orchestrator of the ML underlay.
- MLFO interacts with the management plane to interface with the NFVO to enable NF autoscaling, migration, and placement decisions.

Usage scenario: use case 4.16: Application-specific network slicing through in-network deep learning

- MLFO enables data collection and processing from NFs, where packets are tagged with the information of the application transmitting them.
NOTE 1 – As an example, this may be done using a set of UEs that may be programmed to tag the packets with the application information. Another example may be a simulated NF which tags the packets with the application information. To ensure privacy, information like < dst_ip, dst_port, protocol, ttl, packet_size > can be used as flow features without deep packet inspection.
- MLFO may enable deep learning in the user plan to extract relevant features, to perform labelling, and classification features in a train of packets contained in the flow.
NOTE 2 – The above function may be performed in packet gateway (P-GW) or MEC, for example.
NOTE 3 – This may be done without looking into the payload of packets.
NOTE 4 – To enable application-specific RAN spectrum allocation, TCP SYN/ACK packets with zero-payload can be tagged with application information.
- MLFO should enable training of a deep learning model using appropriate tagged data to be used as training data.
- MLFO enables classification of packets using ML output at various places in the network.

NOTE 5 – ML output for classification may be applied at MEC or AN.

Usage scenario: use case 4.17: Smart traffic mirror – An ML-assisted network service

- MLFO should enable collection and processing of video stream in real-time.
NOTE 1 – This may be done at the edge.
- MLFO may enable pre-processing to extract useful features from the collected data.
NOTE 2 – MLFO may enable these pre-processing capabilities as a capability out of the network.
NOTE 3 – It is not essential to collect data from all streams, only a small part of the stream may be enough (without incurring any timing constraints) should be enough to re-train the ML model.
- MLFO enables specification of an appropriate ML model on the edge or cloud based on the stringent application requirements.
NOTE 4 – E.g., CNN or RNN to predict in less time.
- Input to MLFO may specify the appropriate training of the selected ML model.
NOTE 5 – E.g., offline training based on the collected data or specified data in the ML intent.
- MLFO enables the distribution of training, data collection and transfer of knowledge.
NOTE 6 – Training may be done at the central core (based on limited data), data collection may be done at the edge (for training and inference), and transfer of knowledge may be done from the central core to the edge.

Usage scenario: use case 4.18: ML-based E2E network management for 5G and beyond

- MLFO enables collection of data regarding network management.
NOTE 1 – The collection of data may be from multiple domains of radio, transport, and core networks.
NOTE 2 – The data consists of information from events, inventory, topology, and storage. The information for end-to-end network management may include service request, network resources, path characteristics, and connection information.
NOTE 3 – Usage of standard data model is critical for this purpose.
- MLFO interfaces with the management plane to select the corresponding technology-specific interface so that the application of ML output is enabled in the ML underlay networks.
NOTE 4 – E.g., for fault prediction, RCA, and automated recovery.

Usage scenario: use case 4.19: ML-aided channel modelling and channel prediction

- MLFO enables collection of data with the assistance of data acquisition APIs on BS or UE that follows standard or partially standard format. Context of the data or metadata may be recorded along with the data.
NOTE 1 – Metadata may include scenario-specific channel measurements, environmental data, and system configuration for the measurement equipment or the BS/UE configuration.
- MLFO should be able to perform pre-processing to handle time-series data, and data with different dimensions.
- MLFO should enable protection of the data, and ensure its integrity, accessibility, and availability.
- MLFO may enable specification of an ML model or mechanism to perform channel modelling.
NOTE 2 – The ML model must ensure that ML output is generated in a format to be linked later to form a unified channel model.

- MLFO should also facilitate with federated learning or distributed learning to address the computational demands of specific domains.
NOTE 3 – E.g., wide-band channel data and massive MIMO data.
- MLFO should ensure appropriate training of the deployed ML models.
NOTE 4 – E.g., a distributed model may be trained offline and then updated locally.
- MLFO should interface with the management plane to query a technology-agnostic interface to the underlying physical processes.
NOTE 5 – E.g., to reduce the pilot.

Usage scenario: use case 4.20: ML-based link adaptation optimization

- MLFO enables collection of data from multiple levels of the network.
NOTE 1 – The data may include layer 1 UE level feedback, layer 1 and layer 2 BS measurements. Layer 1 UE feedback may include CQI, RI, PMI, downlink RSRP measurement, and ACK/NACK. Layer 1 BS measurements may consist of MIMO precoder, Uplink RSRP measurements CQI/PMI/RI feedback delay, and estimated doppler frequency offset.
NOTE 2 – Layer 2 BS measurements which should be collected include PRB usage ratio, scheduled MCS, scheduled RANK, user number, and interference from neighbouring cells.
- MLFO specifies an ML model to learn complicated relationships between modelling parameters across network layers.
NOTE 3 – E.g., measurement like BLER with the CQI, PMI, RI, RSRP, interference, ACK/NACK, etc. and predict an appropriate RANK/MCS for a link.
- MLFO queries an appropriate interface from the management plane to interface with the ML underlay networks to apply the relevant ML output.
NOTE 4 – This may include execution of predicted MCS and RI selection policy.

Usage scenario: use case 4.21: ML-based end-to-end network slicing for 5G

- MLFO enables appropriate collection of data.
NOTE 1 – The collection of data may include the data from the radio access network, core network, and transport network.
NOTE 2 – The radio access network data may include GPS, measurement reports for radio condition, resources allocated, connection signalling events, traffic type classified, service target indicator per user/node. It may also include cell/node resources utilization and cell/node KPIs.
NOTE 3 – The core network data includes KPIs, traffic figures, network utilization KPIs, and additional CDRs, and log files. The transport network data may include transport utilization, failures, and bandwidth.
- MLFO enables an ML mechanism to predict the best RA to allocate slices based on the service type. A slice may reflect an optimized service with respect to minimum QoS guarantees according to the service type.
NOTE 4 – This may be using ML mechanism e.g., supervised learning.
- MLFO coordinates with the management plane to interface via with the orchestration mechanism of ML underlays to configure network resources.
NOTE 5 – The configuration may be applied to network slices.

Usage scenario: use case 4.22: Data-driven architecture for ML at the edge

- MLFO enables collection of contextual information data in real-time from ML underlay networks.
NOTE 1 – E.g., from RAN controllers or MEC.

NOTE 2 – This data may include network state data including Identification information, number of users per cell, and number of handovers. MLFO enables the use of spatial information that is introduced by user mobility.

- The pre-processing extracts the relevant metrics that can be used to train the ML algorithm.
- MLFO enables the training of the ML model based on real-world data sets by using a training method, the training data must capture correlations such as real-world data that are difficult to model in artificial data sets.

NOTE 3 – Simulated data can be used to augment the training.

- MLFO enables selection of an appropriate ML mechanism to run on RAN controller at the edge, to control the network itself and provide new services to the users.

NOTE 4 – E.g., supervised, semi-supervised, reinforcement learning, or hybrid learning mechanisms.

- MLFO shall select and coordinate the positioning of ML models.

NOTE 5 – The RAN controllers, deployed at the edge, are associated with a cluster of base stations, and are thus responsible not only for RAN control, but also for running the ML infrastructure with a point of view that captures the spatial correlation. The network controller, placed in the operator's cloud, orchestrates the operations of the RAN controllers. MLFO shall select and coordinate the positioning of such ML models.

NOTE 6 – The model can perform real-time predictions and optimizations based on network conditions including congestion, load of base stations, and mobility patterns.

Usage scenario: use case 4.23: Long-term traffic forecasting

- MLFO enables collection of data with the assistance of dedicated measurement infrastructure.

NOTE 1 – E.g., plugins at gNB.

NOTE 2 – The collected data may include measurements at the base station level for a sufficiently long duration to ensure training can be performed with high accuracy.

NOTE 3 – MLFO should be able to collect episodic observations of the traffic demand. Additional context information that may be relevant includes: cell ID, number of antennas, and average number of connected users/intervals.

- MLFO assures appropriate storage of collected data.
- MLFO enables pre-processing on collected data to process it in a form that can be fed to the model for training.

NOTE 4 – Further, MLFO may represent episodic traffic updates through metadata as summaries sampled e.g., every 5-10 minutes.

- MLFO provides specification to train the deep learning model using offline training based on processed data.

NOTE 5 – Once trained, MLFO should support online training based on summary samples of traffic demand.

NOTE 6 – MLFO may enable parallel training capabilities to exploit parallel computing capabilities of GPUs.

- MLFO selects specification of an ML model to perform network-wide traffic forecasting at a central location.

NOTE 7 – E.g., 3D-CNNs or ConvLSTMs to perform network-wide traffic forecasting at a central location, e.g., P-GW.

Usage scenario: use case 4.24: ML-based utility maximisation of sliced backhails

- MLFO enables collection of data including detailed measurements at network entry points. MLFO can adjust the measurement granularity based on delay sensitivities of each application.

NOTE 1 – The collected data may include information from the uplink/downlink in the backhaul. The information includes the point of entry, e.g., gNB, flow path within the backhaul, flow type, and target KPIs.

- MLFO enables specification of a deep learning model.

NOTE 2 – E.g., CNN to predict an optimal resource allocation to maximise the utility of multi-service backhails networks.

- MLFO enables the training of the DL model centrally.

NOTE 3 – E.g., gateway, with appropriate training method, e.g., offline training along with a set of hyperparameters.

- MLFO may exploit data parallelism/model parallelism to use parallel computing capabilities of GPUs at a central node.

NOTE 4 – P-GW may be an example of a central node in this case.

- MLFO interfaces with the management plane to distribute the ML output to different parts of the network.

NOTE 5 – This may include optimal signalling allocations to each hop along relevant paths.

Usage scenario: use case 4.25: Emergency services based on ML

- MLFO can enable event-driven collection of data which can be triggered based on real-time alerts.

NOTE 1 – IoT sensors, cameras, people alert or authority warning may be examples of real-time alerts.

NOTE 2 – This data may include specific measurements based on an event, e.g., position estimates, external network detection information, and additional information from users, e.g., text, image, or video.

- MLFO enables an ML mechanism, to correlate information from images, external sources, IoT devices, sensors, and alarms, etc. to predict an emergency.

NOTE 3 – E.g., CNN may be used as an example of an ML mechanism.

- MLFO may enable a distributed learning approach, where parts of learning are enabled on the edge and core.

NOTE 4 – E.g., MLFO may specify a deep learning model for accurate prediction at the core and a lightweight ML model at the edge to handle emergencies in real-time.

- MLFO interfaces with the management plane to query interfaces for the radio network, core network, or an ML server to send the warning messages and notifications to people or connected devices.

Usage scenario: use case 4.26: Energy efficient trusted multi-tenancy in 5G-crosshaul

- MLFO enables collection of information from the monitored environment for situational and operational awareness.

NOTE 1 – MLFO may perform this collection with the help of knowledge representation and representation strategies.

NOTE 2 – This may include information across the network for each slice, e.g., throughput, congestion, transmission delays, availability, risk level, and registered events, etc.

NOTE 3 – Sensors and external data repositories may also act as critical sources of factual knowledge.

NOTE 4 – A flexible data model may be specified in the ML intent for an appropriate data collection.

- MLFO may enable essential data pre-processing tasks including event-related correlations.

NOTE 5 – These pre-processing tasks may include normalization, aggregation/fusion, verification, prioritization, and reconstruction.

NOTE 6 – MLFO may consider case-based or attribute-based reasoning to find dependencies between events and resources.

- MLFO may enable specification of adaptive machine learning mechanisms or ML models to support secure, trusted, and energy-efficient multi-tenancy.

NOTE 7 – Multitenancy is defined in [ITU-T Y.3111]

NOTE 8 – MLFO may enable adaptive reinforcement learning, deep learning, or hybrid ML mechanisms.

NOTE 9 – To support multi-tenancy, these ML mechanisms can allocate virtual resources to different tenants according to SLAs with appropriate configurations while ensuring appropriate isolation.

NOTE 10 – These ML mechanisms may enable crosshaul resource orchestration, i.e., provide resource orchestration at the FH, BH, and midhaul at the tenant level.

NOTE 11 – MLFO may calibrate/recalibrate lightweight ML mechanisms to filter security threats at the midhaul (FH may be resource-constrained).

NOTE 12 – MLFO may enable these ML mechanisms at the BH to countermeasure future threats, its replicas, and possible propagation holes.

NOTE 13 – These ML mechanisms may provide anticipatory responses according to future expected events, evolution, and propagation of discovered threats.

NOTE 14 – The ML models may consider a continuous and collaborative decision-making process to offer a service in a multi-stakeholder scenario.

- MLFO may select these mechanisms/models to enforce contextual adaptation of trust models, e.g., timely detection of virtual network function (VNF) anomalies.

NOTE 15 – MLFO may enable deep learning ML models to countermeasure cloud-level threats at the tenant-level to prevent their propagation.

- MLFO may be augmented with a closed loop system, where expected key performance indicators (KPIs) can be assessed and appropriate corrective tactics can be taken.

NOTE 16 – These tactics may include modification of statistical models, data pre-processing rules and inference rules, and recalibration of heuristics.

- MLFO may enable a metacognition layer to optimize its operation under unforeseen events/threats.

NOTE 17 – E.g., this layer can monitor a reinforcement learning-based ML model to optimize it.

NOTE 18 – This layer should be able to dynamically acquire and validate the consequences of decision making with the help of operational knowledge.

NOTE 19 – The metacognition layer may incorporate efficient onboarded security and energy policies complemented with situational awareness to take appropriate resource allocation decisions for each tenant.

- MLFO may enable ML mechanisms for flow-level security driven by authorization/privacy policies at the edge in real-time.
- These mechanisms should comply with the trust model and energy-saving policies required by different 5G actors, e.g., slice owners, Telcos, and verticals.

- MLFO may deploy ML mechanisms to audit data integrity of each slice, e.g., resilience, and VNF anomaly detection.
- MLFO ML mechanisms are compliant with ETSI enhanced high-level NFV security management framework [b-ITEH] to entail energy-efficient security measures.
NOTE 20 – E.g., supervised or reinforcement learning or deep reinforcement learning.
NOTE 21 – E.g., convolutional neural networks (CNN) can be used to enable trust by enforcing appropriate levels of security and trust, thus enabling trusted execution environment (TEE).
- MLFO may enable training/retraining mechanisms for deep learning models to ensure high accuracy.

Usage scenario: use case 4.27: Network slice SLA assurance based on ML

- MLFO enables continuous collection of (large amount) of traffic data or measurements provided by operation, administration, and maintenance tools (OAM) tools.
NOTE 1 – MLFO may use this collected data to calculate service measurements, i.e., KPIs.
- MLFO may enable a ML mechanism to predict the level of assurance for each slice.
NOTE 2 – E.g., a supervised learning mechanism to predict the assurance level based on KPIs.
- MLFO interface with the management module of ML underlay can be used to interact with the data collection frameworks,
NOTE 3 – E.g., for service-based architecture (SBA)/SBI to the SBI bus system.
- MLFO may enable an execution environment to train/retrain the ML models according to the collected data.

Usage scenario: use case 4.28: ML service management for smart cities

- MLFO enables a ML mechanism or model to offer the services based on the application type and constraints.
NOTE 1 – This ML mechanism can be used to optimize the slices according to the user requirements.
- MLFO may enable pre-processing to label the packets according to the application type.
NOTE 2 – E.g., traffic types may include mMTC, URLLC, and eMBB.
- MLFO enables a ML mechanism/model for cross-slice pollination to provide the smart behaviour in the network.
NOTE 3 – E.g., smart cities.
- MLFO may enable interoperable transfer of knowledge from one operator to another.

Usage scenario: use case 4.29: Automated testing of services

- MLFO may monitor the performance of different ML models with respect to different services in a 5G network.
- MLFO interfaces with the management module of ML underlay be used to interact with the testbed for automated testing of ML models.
NOTE – This testing should be standalone without affecting the operator KPIs.
- MLFO may verify ML models in the testbed in compliance with the local regulatory policies before deploying them in the general availability network.

Usage scenario: use case 4.30: Combating use of counterfeit ICT devices – ML-assisted network service

- MLFO enables continuous collection of traffic data at different levels of the network.

NOTE 1 – This may include data from the radio access network including device characteristics, e.g., number of antennas, MIMO, etc. Other information may include IMEI/model of the device.

- MLFO may enable pre-processing to classify the traffic data according to the capabilities listed against the IMEI.

NOTE 2 – The capabilities may include internet access, antenna capability, and MIMO capability, etc.

- MLFO enables an ML model or mechanisms to identify the cloned IMEI based on the capabilities associated with the IMEI available in the core network node.

NOTE 3 – MLFO may deploy these ML models/mechanisms at the edge to prevent cloned devices from accessing the network.

- MLFO may train/retrain the models to continuously update them for faster and accurate actions against the cloned IMEI devices.

Usage scenario: use case 4.31: ML-based identification of illegal exchanges using the simulator (SIM) boxes

- MLFO enables collection of continuous real-time data at the network edge.
- MLFO may enable pre-processing tasks to classify the real-time traffic based on SIM number.
- MLFO enables an ML model or mechanism to predict illegal SIM exchanges at the edge.

NOTE 1 – This ensures that illegal traffic does not pass through the network.

- MLFO enables training/retraining of ML models based on the traffic patterns from illegal SIM exchanges.

NOTE 2 – E.g., VOIP traffic patterns.

- MLFO interfaces with the ML underlay may be used to communicate the ML output to the operator configured authority.

Appendix II

MLFO requirements analysis for the use cases

The requirements for MLFO are derived from the usage scenarios described in Appendix I. Table II.1 shows the MLFO requirements analysis for the use cases.

Table II.1 – MLFO requirements analysis for the use cases

Corresponding use case in [ML5G-I-227-R2]	Requirement
4.1 Traffic classification	- MLFO is required to take as input, the nature of data collection. NOTE 1 – E.g., real-time, batch, etc. - MLFO is recommended to take as input, the specification of pre-processing tasks. NOTE 2 – This may include tasks such as labelling of data according to application and protocol type to train the model. - MLFO is recommended to manage data storage to store collected data. - It is required that MLFO should be able to enable and configure sources of data based on the specification of an ML application. NOTE 3 – E.g., the data should include network information, the transport layer payload, and some relevant protocol fields in the data. - It is required that MLFO enables the selection of different types of models for an ML application based on the characteristics of ML application, data, and other constraints. NOTE 4 – For example, CNNs, RNNs, or hybrid models can be selected for a use case. - It is required that MLFO monitors various ML pipeline nodes and provide their real-time status to the operator. - It is required that MLFO supports the configuration of data models. NOTE 5 – This may be according to the source and the type of ML model used or according to other criteria such as the type of data sources used. - It is required that MLFO supports the update and retraining of ML models with minimal impact on the performance of an ML application. NOTE 6 – The performance KPI of the ML application may be defined in the ML intent. - It is required that MLFO supports security requirements for data handling. NOTE 7 – MLFO must be aware of any encryption/decryption that has been performed on data and this information must be specified within the ML application specification. It will be used by the MLFO to configure the data sources, accordingly.
4.2 RAN-assisted transmission control protocol window optimization	MLFO is required to enable collection of data required for the use case in real time. NOTE 1 – This data may include provisioning of real-time interfaces at different levels of the network.

Table II.1 – MLFO requirements analysis for the use cases

Corresponding use case in [ML5G-I-227-R2]	Requirement
	- MLFO is recommended to enable specific learning mechanisms required for the use case along with appropriate training mechanisms NOTE 2 – E.g., supervised learning and offline training of the corresponding model. - MLFO is recommended to interface with the management plane of the ML underlay to understand the data and service capabilities of the ML underlay. NOTE 3 – E.g., network topology, metadata of data models, and service orchestration capabilities of the management plane of the ML underlay may be examples of such capabilities of the underlay. NOTE 4 – Such metadata may be used for interoperability in data handling [b-ML-Schema-Gustavo] by deriving schema information that is supported by the ML underlay. NOTE 5 – MLFO understands the technology-agnostic data model specified in the ML intent, queries the management plane for the corresponding technology-specific interface, and then configures the DBr-CP accordingly. - Based on the capabilities of the NF or AF, MLFO is recommended to enable a specific interface towards the NF or AF for ML data collection or ML output-based configurations. NOTE 6 – E.g., This may be implemented in the form of an interface plugin. The interface plugin may be specified in the ML intent as an input to the MLFO.
4.3 ML-based mobility pattern prediction	- Application-specific KPI requirements are recommended to be specified in the ML intent. - MLFO is required to enable collection of data at various network levels. NOTE 1 – Some of the data sources may be relevant to environmental information, e.g., GPS data. NOTE 2 – This data may include network performance data, cell status queries, sleep/wake-up instructions, context data, environmental information, configuration information, and other current network interactions. - MLFO is recommended to select the type of ML model and the type of learning. NOTE 3 – this may include deep learning or hybrid ML models, e.g., RNN, reinforcement learning, or deep reinforcement learning for local predictions of a user trajectory. NOTE 4 – It also selects and specifies an appropriate deep learning model, e.g., an LSTM for network-wide global predictions of a user trajectory based on local predictions. - MLFO is recommended to manage the storage requirements of the ML application. NOTE 5 – This may include providing or enabling a caching module to store data and updating results of local and global predictions.

Table II.1 – MLFO requirements analysis for the use cases

Corresponding use case in [ML5G-I-227-R2]	Requirement
	NOTE 6 – For example, this may be used to store a user's predicted trajectory. - MLFO is recommended to enable the SINK to interface with the corresponding management modules in the underlay to configure the ML output. NOTE 7 – This may include an execution module to generate man-machine language, which is then to be communicated to a BTS and other elements of the core and managers, e.g., vNFM, SDNV, etc. - MLFO is recommended to enable training/retraining of deep learning or hybrid ML models based on real-time data. NOTE 8 – This may be used to improve the accuracy and effectiveness of the selected models both for the local and global predictions. - MLFO is recommended to enable the use of stored data, including historical information for training of ML models. NOTE 9 – This historical information includes reference signal received power/quality (RSRP/RSRQ). NOTE 10 – Other historical information which can be used to train the model includes network load, throughput, interruption time, awareness and prediction of UE position, moving direction, and speed, etc. NOTE 11 – RNNs can predict mobility patterns predictions better than traditional ANNs (e.g., FNNs). RNNs have small memory, which may store historical information. - MLFO is recommended to enable simulated data sources. - MLFO is recommended to enable ML models based on the different levels of data. NOTE 12 – For example, a UE level ML model may be used for UE level predictions. - MLFO is required to enable configuration of data privacy and security policies from different authorized sources. NOTE 13 – This may include policies for handling mobility data and user trajectory prediction of individual users. This may have geographic regulations and may be related to the storage and use of data. NOTE 14 – MLFO may chain and configure the sources and sinks based on the data privacy policies. ML sink may include MEC. - MLFO is recommended to enable the application of ML output at different levels in the network. NOTE 15 – E.g., high-level MEC and low level MEC may be examples where such ML output may be applied.
4.4 Load balance and cell splitting/merging	MLFO is required to have a data collection mechanism to collect information of a cell and its neighbouring cells. NOTE 1 – E.g., real-time data, etc.

Table II.1 – MLFO requirements analysis for the use cases

Corresponding use case in [ML5G-I-227-R2]	Requirement
	NOTE 2 – The information can include the number of UEs, RB utilization, PDCP package number, the number of RR connection requests, and the active E-RAB number of current and neighbour cells. NOTE 3 – This data can be collected from the gNB and the network management system. • MLFO is recommended to take a pre-processing specification as input. NOTE 4 – E.g., a pre-processing task can dictate application-specific labelling of data to aid the training of a model. • It is required that MLFO enables the selection of different ML models based on application requirements, data, and constraints. NOTE 5 – These models can include supervised, unsupervised or hybrid learning models. NOTE 6 – These models can be used by the MLFO to perform different predictions including load balancing, cell splitting, cell merging, cell status prediction, and feasible mapping between UEs and cells. • It is required that MLFO monitors various ML pipeline nodes and may provide output of one ML model as input to another model to make certain predictions. • It is required that MLFO regulates training and retraining of models based on any changes in data to retain the prediction accuracy and performance promises of ML models. NOTE 7 – MLFO can use either offline or online training mechanisms to adhere to performance KPIs and constraints as defined in the ML intent. • It is recommended that based on the prediction results, MLFO can choose a set of configuration parameters to execute the cell splitting/merging process. • It is recommended that MLFO communicates the prediction results to the network management system through an interface with the management plane of the ML underlay. NOTE 8 – The network management system in turn can directly control/balance the UEs connected to the eNodeBs.
4.5 Cognitive heterogeneous networks	• It is recommended that the MLFO is supplied with the application requirements and constraints as an ML intent. • MLFO is required to enable specification and collection of data from the network devices in real-time. NOTE 1 – This data can include radio access network, core network and transport data. The radio access network data may include location information, radio conditions, resource allocation information, connection information, traffic type, service target indicator, cell/node resource utilization, cell/node KPIs. The core network data may include core network KPIs, core traffic figures, network utilization KPIs, CDRs, and log files. The transport

Table II.1 – MLFO requirements analysis for the use cases

Corresponding use case in [ML5G-I-227-R2]	Requirement
	data to be collected can include transport utilization, failure rate, and bandwidth. • MLFO is recommended to specify a pre-processing mechanism to assist in ML model training. • It is recommended that the MLFO can enable specification/selection of different ML models or combination of them. NOTE 2 – MLFO can select RNN or fuzzy neural network models to allocate crowd blocks from different communication networks access nodes. NOTE 3 – MLFO can select supervised, unsupervised, or a combination of ML models for dynamic adjustment of parameters, e.g., transmit power and dynamic on/off embedded nodes to optimize coverage, capacity, and quality. NOTE 4 – MLFO can select supervised, unsupervised, or a hybrid model to assist the central orchestrator to support optimized E2E actions. • It is required that the MLFO can enable training/retraining of ML models to incorporate network dynamics and application requirements. NOTE 5 – E.g., MLFO can select either offline or online training for an ML model depending on the constraints as specified in the intent. • It is recommended that the MLFO has an interface with the management plane of the ML underlay. NOTE 6 – MLFO can query the management plane for technology-specific interfaces for appropriate user behaviour-aware operations. NOTE 7 – This interface can help to understand the services of central and distributed orchestrators (MEC) to modify network parameters and settings automatically. • It is recommended that the MLFO is equipped with distributed learning capability. NOTE 8 – E.g., MLFO can perform ML model training on a centralized node and then distribute the trained model to the edge to respond in real-time to certain applications, e.g., drones and connected vehicles.
4.6 ML-based SON	• Application-specific requirements and constraints are recommended to be specified in the ML intent. • MLFO is required to enable collection of data related to the past behaviour of the network. NOTE 1 – This data may include radio access, core network, and transport data. The radio access network data may include location information, radio conditions, resource allocation information, connection information, traffic type, service target indicator, cell/node resource utilization, cell/node KPIs, i.e., throughput, drops, accessibility, and power. The core network data may include core network KPIs, core traffic figures, and network utilization KPIs. The transport data to be collected can include transport utilization, failure rate, and bandwidth.

Table II.1 – MLFO requirements analysis for the use cases

Corresponding use case in [ML5G-I-227-R2]	Requirement
	NOTE 2 – MLFO can enable data collection at various data sources. • MLFO is recommended to store the collected data in an ML data store for later use. • It is required that MLFO supports active probing across the network to collect transport KPIs, e.g., FLR, delay, and jitter. • It is required that MLFO consolidates the collected data from the network management system and active probes. • It is recommended that MLFO supports necessary pre-processing. NOTE 3 – E.g., MLFO can perform application-specific labelling on collected data to train the models. • It is required that MLFO enables the selection of ML models or a combination of them according to specified application requirements and constraints. NOTE 4 – MLFO can select fuzzy Q-learning in each eNB to predict performances or behaviours of the network. For pattern identification or grouping, MLFO can select ML models e.g., Self-organizing Maps (SOMs) or Fuzzy C-means. NOTE 5 – MLFO can select ML techniques, e.g., genetic algorithm for cell planning, coverage, and power adjustments. To support specific learning, MLFO can use transfer learning, reinforcement learning, or a hybrid model to automatically configure parameters to provide best services. NOTE 6 – MLFO may select supervised, unsupervised, or a hybrid model and may chain them to achieve global network optimizations. NOTE 7 – This may include supervised learning to identify a fault, classify it, and take necessary actions to recover from it, e.g., the removal of failures and automatic adjustment of parameters. • It is recommended that MLFO interface with the corresponding management modules in the ML underlay. NOTE 8 – This interface can be used for automatic configuration of underlying network. • MLFO is required to enable training/re-training of deep learning or hybrid ML models based on real-time data. NOTE 9 – this training can be performed in an online or offline manner to improve accuracy and effectiveness of ML models according to the application requirements and constraints. • It is recommended that MLFO interfaces with the management modules of ML underlay. NOTE 10 – This interface can be used by the MLFO to modify network parameters and settings through network management system. • It is recommended that MLFO interfaces with the management plane to connect with diverse data sources and databases via APIs.

Table II.1 – MLFO requirements analysis for the use cases

Corresponding use case in [ML5G-I-227-R2]	Requirement
4.7 ML-based QoE optimization	• It is recommended that application requirements and constraints be specified in the ML intent. • MLFO is required to enable data collection in real-time at various network entry points. NOTE 1 – MLFO may deploy mechanisms to carry out accurate network measurements, service quality, and the recording of raw network status samples. NOTE 2 – The data to collect may include user experience and QoS/QoE feedback data. The user experience data consists of rebuffering time, staying time, and blockage frequency, whereas, QoS/QoE feedback data may include information on latency and jitter. • MLFO is recommended to support specifications of any required pre-processing. NOTE 3 – e.g., pre-processing can be used to refine raw network status data and to remove protocol-dependent features. The pre-processing can also address missing information, e.g., QoS/QoE feedback information. NOTE 4 – MLFO can store these dependent features alongside the obtained KPIs in a network status database. • MLFO is recommended to select types of feature extraction mechanisms to extract relevant features from the data. NOTE 5 – E.g., rebuffering time, staying time, and blockage frequency from the feedback and user experience. • MLFO is required to support the selection of ML models, e.g., supervised, unsupervised, or hybrid. NOTE 6 – These ML models can be used by the MLFO to optimize the network configurations to maximize the QoE scores of network users while meeting network capacity and service requirements. • MLFO is required to select and deploy an ML mechanism at the sink node or core node. • MLFO is required to deploy and monitor different nodes from the ML pipeline at the sink node or core node. • MLFO is required to provide different re-training/training mechanisms to train the models to make sure predictions are stable and proactive. NOTE 7 – MLFO can use offline or online training mechanisms according to the application requirements and constraints. NOTE 8 –It can also use a reinforcement learning approach to consider real-time network states and service quality feedback from the users to perform configurations to increase the QoE optimizations. • MLFO is recommended to have an interface with the management modules in the underlay to configure the network and MEC within a proper granularity with the help of the network management system.

Table II.1 – MLFO requirements analysis for the use cases

Corresponding use case in [ML5G-I-227-R2]	Requirement
	MLFO is required to ensure mechanisms for privacy and security for data handling. NOTE 9 – E.g., MLFO can impose limits on the use and accessibility of data by other parties and for other purposes.
4.8 RRM for network slicing	- It is recommended that MLFO is provided with application requirements and constraints as an ML intent. - MLFO is required to enable data collection of real-time data at various network entry points. NOTE 1 – The collected data may include: slice-level performance measurement parameters, slice-level network performance measurements and alarm data, radio resource utilization patterns, cell status queries, user mobility and radio channel conditions data, slice profiles, user service requests, and QoS. - It is recommended that MLFO can generate data through simulations to improve training of ML models. NOTE 2 – E.g., this data can represent rare events for which enough real-time data is not available. Simulated data can be helpful for training the models before the wide commercialization of slice use cases. - It is recommended that MLFO supports any required pre-processing. NOTE 3 – E.g., labelling of data based on application type to train the model. - MLFO is required to select different ML models based on application requirements and constraints. NOTE 4 – MLFO may select supervised learning, unsupervised learning or a deep reinforcement learning model to predict in real-time per slice/per-cell radio resource requirements and the optimal radio resource allocation trajectory to optimize the network operation. e.g., CNN. These prediction models can also predict slice performance, optimal slice planning, and optimization strategies. - MLFO is required to enable a training/retraining mechanism to improve the accuracy and effectiveness of ML models. NOTE 5 – MLFO may use online training mechanisms to continuously collect data and train the ML models. - MLFO is recommended to have an interface with the management modules of the ML underlay. NOTE 6 – This interface can be used by the MLFO to interact with NMS, CU, or DU to enable data collection, access to non-real-time slice-related policies, configurations, and dynamic radio resource schedulers at both the CU and DU level. - MLFO is required to have an interface with the management plane to generate man-machine language (MML) commands.

Table II.1 – MLFO requirements analysis for the use cases

Corresponding use case in [ML5G-I-227-R2]	Requirement
	NOTE 7 – These commands can be used by the MLFO to communicate with the RAN, network core elements, and network managers to execute appropriate slice resource allocation plan.
4.9 ML-based network management for industry 4.0	• It is required that the MLFO is provided with the accuracy, latency, and other functional requirements of a specific application. • MLFO is required to enable data collection at various levels of the network. NOTE 1 – This data can include real-time operational and planning data. NOTE 2 – The data collection process is recommended to impose stringent timing requirements to assist the response to real-time events. NOTE 3 – This data may be measurement data from sensors within a manufacturing or production deployment including signal strength, packet loss rate, spreading factor/multiplier, modulation, coding, spatial streams/ antenna usage. Other information may include context information data, planning data, and production data. NOTE 4 – MLFO is recommended to consider data collection overhead during collector placement. • MLFO is recommended to specify pre-processing algorithms for input data. NOTE 5 – These algorithms can be used to perform labelling according to the application and protocol type of the input data. NOTE 6 – The data pre-processing should impose stringent timing requirements to assist the response to real-time events. • MLFO is required to select different types of ML models or learning mechanisms for different network-wide predictions. NOTE 7 – MLFO can select supervised, unsupervised, or hybrid ML modes to predict or forecast optimal network configurations in real time. NOTE 8 – MLFO can use multivariable linear regression as an error-based learning, whereas decision trees can be used as information-based learning by the MLFO. NOTE 9 – MLFO should consider the stringent timing requirements of the industrial applications for model selection, e.g., MLFO can select light-weight ML solutions on the edge. • MLFO is required to specify training/retraining mechanisms to train the ML models. NOTE 10 – E.g., offline or online according to the performance KPIs or unique requirements of industrial platform specified in the ML intent. NOTE 11 – These unique requirements may include reliability, availability, security and determinism. MLFO must meet the accuracy, latency, and functional requirements of a specific service. NOTE 12 – The training process is recommended to impose stringent timing constraints to assist the response to real-time events.

Table II.1 – MLFO requirements analysis for the use cases

Corresponding use case in [ML5G-I-227-R2]	Requirement
	• MLFO is required to deploy security measures to secure industrial plants-specific data and messages. • MLFO is recommended to interface with the management modules of ML underlay with minimum time. NOTE 13 – This interface can be used to support specific APIs, which can enable setting/monitoring/measuring device parameters and network related functions NOTE 14 – This interface can realize interoperability among devices from different manufacturers. NOTE 15 – This interface can also be used to support industrial platform protocols to drive scheduled delivery, network resource allocation, gateway association information, etc. • MLFO is recommended to efficiently deploy nodes of the ML pipeline across the network to respond to real-time events. NOTE 16 – This requires efficient deployment of collectors, pre-processors, ML models, and training across the network. • MLFO is recommended to provide predictions which are resilient and does not impact the management of the devices under any failures. NOTE 17 – E.g., nodes from the ML pipeline can be deployed in a redundant and distributed fashion across the network. • MLFO is recommended to support pre-specified/standardized data formats and protocols.
4.10 ML-based correlations between transport KPI and radio KPI	• It is recommended that MLFO be provided with the application specification including requirements and constraints. • MLFO is required to enable data collection from hourly instances to higher frequencies for longer periods of time. NOTE 1 – This data can include transport, network, and radio performance KPIs at different layers of the network. NOTE 2 – This data can include transport network topology data, transport measurements, and radio measurements. It may also include real customer experience measurements and data retrieved from crowdsourcing applications, or region-specific drive tests, e.g., Web browsing, Video streaming, VoIP, Online Gaming, VR/AR, etc. • MLFO is recommended to collect data through active probing to get accurate results for transport KPIs. NOTE 3 – E.g., FLR, delay and jitter. • MLFO is required to support configuration of data models. NOTE 4 – These models can be configured by using correlations between transport and radio KPIs at different layers of the network.

Table II.1 – MLFO requirements analysis for the use cases

Corresponding use case in [ML5G-I-227-R2]	Requirement
	- MLFO is recommended to manage the storage requirements of the big data platform to manage the large volume of data collected over a long period with low granularity. NOTE 5 – These storage requirements should be enough to store transport, network, and radio performance KPIs. - MLFO is recommended to perform necessary pre-processing tasks to the input data from various sources. NOTE 6 – MLFO pre-processing can filter the interactions. e.g., sometimes interactions over an hour/day can be more of greater interest than continuous interactions. NOTE 7 – MLFO pre-processing can also consolidate data from the NMS and active probes to train the ML models. - MLFO is required to select different ML model to predict correlations between transport and radio KPIs. NOTE 8 – E.g., MLFO can select supervised model to predict the performance of radio sites. NOTE 9 – MLFO can consider different models for predictions including CNNs, HMM, linear SVR, SVR Elastic Net, or hybrid models. - MLFO is required to use training/retraining mechanisms to meet the application requirements as specified in the ML intent. - MLFO is required to interface with the management plane of ML underlay to perform certain network-wide actions. NOTE 10 – These interfaces can help the MLFO to optimize the current thresholds of the transport network to control KPIs. NOTE 11 – This interface with the management plane of the ML underlay is required to interact with the network element manager system connected to different vendors to assist data collection. NOTE 12 – This interface can also be used by the MLFO to communicate with the active probing monitoring system connected to network elements to obtain accurate transport KPIs.
4.11 Retention and storage	- MLFO is recommended to take as input the specification of the use case including requirements and constraints. - MLFO is required to perform data collection at a large scale at various levels in the network during the entire lifecycle depending on the scenario. NOTE 1 – This data can be from key data sources from a core network including network parameters and service impact indicators, service usage and reference data from external sources, e.g., business specific, legal, or geographic. NOTE 2 – Other data of interest can include fault and performance data, service quality and sensor data, and processed data from a core node. - MLFO is required to manage storage of big data at various network levels.

Table II.1 – MLFO requirements analysis for the use cases

Corresponding use case in [ML5G-I-227-R2]	Requirement
	• MLFO is recommended to take as input the specifications of the pre-processing tasks. NOTE 3 – E.g., MLFO can perform labelling on data based on application and protocol type to assist in classification. • It is required that MLFO can select different types of ML models and learning mechanisms to perform different retention and storage intelligence functions. NOTE 4 – MLFO is required to select a model to classify the application based on service impact, user impact, regulatory impact and the basis of future influence. NOTE 5 – MLFO is required to use a supervised, unsupervised, or a hybrid model to process on-demand queries for data deletion or archiving. NOTE 6 – MLFO is required to enable a supervised model, e.g., decision trees to classify data to archive or delete. NOTE 7 – MLFO can also select a model to predict and proactively perform data clean-up under complex storage requirements. NOTE 8 – MLFO is required to select an ML model, e.g., supervised learning to identify priority service data preservation, and data that can be disposed of. NOTE 9 – MLFO is required to select a ML model to learn the predictability and relativity of data to make archive or delete decisions. NOTE 10 – MLFO is required to select an ML model to establish business rules, e.g., supervised learning to automatically store data in slow or fast storage. NOTE 11 – MLFO specifies an ML model to identify data that is technically obsolete to make a delete decision. • MLFO is required to train/retrain the ML models to ensure the prediction accuracy of storage and retention decisions. NOTE 12 – E.g., offline training method. • MLFO is required to interface with the management modules of ML underlay for storage or deletion decisions within the network. NOTE 13 – This interface can also be used by the MLFO to ensure interoperability and intelligence from one CSP to another CSP. • MLFO is required to consider dynamic business rules along with the output ML models to take an optimal decision regarding data use, storage, and state transition. NOTE 14 – MLFO should be able to work around dynamic business rules at all levels i.e., application/service, network and radio layer. • MLFO is required to support regional law regarding security and privacy. • MLFO is recommended to define the data retention to comply with regulations and privacy laws.

Table II.1 – MLFO requirements analysis for the use cases

Corresponding use case in [ML5G-I-227-R2]	Requirement
	NOTE 15 – E.g., the policy needs to have rules for data sharing with external agencies and other CSPs. • MLFO is recommended to perform time sensitivity analysis of proactive data clean-up to support complex storage requirements scenarios. • MLFO is recommended to perform location sensitivity analysis of storage, including hierarchical storage analysis from far-end/edge to core.
4.12 E2E network operation automation – Service design	• It is recommended that the MLFO is supplied with data models with specific service requirements as an intent. • It is required that the MLFO enables and configures data sources for data collection at various network entry points. • MLFO is recommended to manage data storage for collected data. • MLFO is recommended to specify any pre-processing tasks to transform the data into a reasonable format for training. NOTE 1 – E.g., MLFO pre-processing tasks can clean, format, or extract data from the raw data. • MLFO is required to select ML models or learning mechanisms for certain network-wide predictions. NOTE 2 – MLFO can select supervised or unsupervised learning to predict logical network requirements from the service user requirements stated in the data models, e.g., RNN or hybrid, etc. • MLFO is recommended to support configuration of data models as specified in the ML intent. • MLFO is recommended to update and retrain the ML models to ensure the accuracy of the ML predictions. NOTE 3 – MLFO can select an online or offline training mechanism according to the service requirements as specified in the ML intent. • MLFO is recommended to interface with the management plane of the ML underlay to perform network-wide configurations and deployments. NOTE 4 – This interface can be used by MLFO to perform network service deployment to reflect the logical network requirements to meet the service requirements as specified in the data model in the intent.
4.13 E2E network operation automation – Logical network design and deployment	• It is required that MLFO is provided with the application requirements and constraints as an intent. • It is required that the MLFO is provided with a collection of a large number of design intents and design results in a machine-readable format. NOTE 1 – These intents may consist of logical network requirements referred to a design intent with functional/non-functional requirements along with concrete design information, e.g., TOSCA/OASIS.

Table II.1 – MLFO requirements analysis for the use cases

Corresponding use case in [ML5G-I-227-R2]	Requirement
	NOTE 2 – The network design results may include concrete design information including components, mapping to physical nodes, and configuration parameters, etc. • MLFO is required to manage data storage to store collected design intents and design results. • MLFO is required to take as input any required pre-processing tasks. NOTE 3 – E.g., a pre-processing task can extract relevant data from the design intents and design results to train the models. • MLFO is required to update and train ML models to ensure reasonable accuracy. NOTE 4 – MLFO can select a training mechanism, e.g., offline training to train the ML model based on the updated datasets. • MLFO is required to select ML models and learning mechanisms according to application characteristics, data, and constraints. NOTE 5 – e.g., MLFO can select supervised learning, reinforcement learning, or hybrid model to select a template or script. • MLFO is required to interface with the management modules of the ML underlay. NOTE 6 – MLFO can use this interface to interact with a workflow generator to generate an executable workflow to safely reconfigure the current network infrastructure status to the desired status. NOTE 7 – MLFO can use this interface to interact with the infrastructure controller, e.g., OpenStack or Kubernetes to manage computing resources required to deploy logical network components. NOTE 8 – MLFO can also use this interface to interact with a workflow executor to allow deployment of designed logical network on the physical network. • It is required that the MLFO is provided network requirements in declarative form, i.e., as a design intent. NOTE 9 – The design intent represents logical network descriptions with functional/non-functional requirements and output design results.
4.14 E2E network operation automation – Fault detection and recovery	• MLFO is required to perform data collection at various network levels including network functions and infrastructure on a real-time basis. NOTE 1 – This data may include performance data related to network function (NF) and NFV infrastructure (NFVI), e.g., CPU utilization, memory utilization, disk I/O, etc. NOTE 2 – The collected data can include performance, event, topology, and configuration data from the NFV and NFVI in real time. NOTE 3 – It can include traffic information, e.g., traffic volume, discard counter error counter, etc.

Table II.1 – MLFO requirements analysis for the use cases

Corresponding use case in [ML5G-I-227-R2]	Requirement
	NOTE 4 – It can be previously known data related to fault detection and recovery procedures. • MLFO is recommended to perform necessary pre-processing to bring the data into a proper format. NOTE 5 – It can be used to extract common topology and configuration models from the data to understand network topology for efficient RCA. • MLFO is recommended to generate simulated data based on a test environment. • MLFO is required to select ML models and mechanisms to meet the application characteristic, data, and constraints. NOTE 6 – MLFO can select a supervised learning model to detect faults of various types. NOTE 7 – MLFO can select an ML model or learning mechanism to detect impending failures. NOTE 8 – MLFO can also select a supervised learning model to predict the root cause of failure, e.g., a deep neural network (DNN) with autoencoders and autoregressive NN. NOTE 9 – MLFO can select a supervised or reinforcement learning ML mechanism e.g., a partially observable Markov decision process (POMDP) to automatically select an optimal recovery action to improve the automated workflow. • MLFO is recommended to provide training mechanisms based on training datasets including an actual or test environment. NOTE 10 – E.g., offline or online training mechanisms. MLFO is required to interface with the management modules of ML underlay to interact with network controllers. NOTE 11 – This interface can be used by the MLFO to perform fault detection, RCA, and automated recovery. • MLFO is recommended to perform a closed-loop function to improve the performance of the ML pipeline. NOTE 12 – E.g., results of RCA can be used by MLFO to improve the prediction accuracy of ML models for fault detection and recovery.
4.15 E2E network operation automation – Network resource adaptation	• It is recommended that MLFO is provided with the application requirements and constraints as an intent. • MLFO is required to enable data collection from network functions (NFs) at various levels including the control, service, and data planes. NOTE 1 – Control plane information may include control requests from an external controller or orchestrator. NOTE 2 – Service plane information may include state data along with identification and configuration information, e.g., computational resources

Table II.1 – MLFO requirements analysis for the use cases

Corresponding use case in [ML5G-I-227-R2]	Requirement
	such as CPU, memory, and storage for NFV and service function chains (SFCs). NOTE 3 – State data in the data plane may include information about the network resources, e.g., physical switches and links with identification and configuration information. • MLFO is recommended to support necessary pre-processing tasks on available data. NOTE 4 – E.g., labelling of data based on application and protocol type. • MLFO is required to enable the selection of different types of models for ML applications based on the characteristics of the application, data, and constraints as specified in the intent. NOTE 5 – MLFO can select GNN or deep reinforcement learning to predict and dynamically adjust resources according to service requirements and traffic dynamics. NOTE 6 – MLFO can select an ML model to predict the load on different NFs and can predict the candidate NFs to which the current load can be migrated, and therefore can make a migration decision. NOTE 7 – MLFO can select an ML model to predict the best order of NF placement and new network topology for a SFC. • MLFO is recommended to have the capability of agile closed-loop operation of resource control. NOTE 8 – Closed-loop operations of resource control can be used by the MLFO to adapt network resources to the dynamic environment. • MLFO is required to select different types of training/retraining mechanisms to train the ML models to continuously meet diverse QoS requirements of various types of applications. NOTE 9 – E.g., MLFO can select data parallelism or model parallelism to speed up the training process in a distributed fashion. • MLFO is required to interface with the management plane of the ML underlay to interact with the NFVO to perform NF autoscaling, migration, NF placement decisions, and closed-loop operations of resource control.
4.16 Application specific network slicing through in-network deep learning	• It is recommended that MLFO is provided with the specification of the ML application including any requirements, constraints, and data models. • MLFO is required to enable data collection at various network levels with or without the assistance of customized UEs. • MLFO is recommended to support different pre-processing tasks to transform data into a form suitable for learning. NOTE 1 – E.g., labelling of data based on application type. NOTE 2 – This labelling can be performed on specific UEs programmed to tag packets based on application information.

Table II.1 – MLFO requirements analysis for the use cases

Corresponding use case in [ML5G-I-227-R2]	Requirement
	NOTE 3 – This labelling can also be performed by a simulated NF. - MLFO is recommended to deploy mechanisms to ensure the privacy of data across the ML pipeline. NOTE 4 – MLFO can use header information such as < dst_ip, dst_port, protocol, ttl, packet_size > as flow features without deep packet inspection. - MLFO is required to select different types of deep learning models according to the application characteristics, constraints, and data. NOTE 5 – E.g., MLFO can deploy a deep learning model at the packet gateway (P-GW) or MEC to extract relevant features in a train of packets contained in the flow and can allocate resources accordingly. NOTE 6 – To enable application-specific RAN spectrum allocation, the deep learning model can use TCP SYN/ACK packets with zero payload to tag them with the application information. NOTE 7 – MLFO can employ an ML model to map slice/spectrum to uplink/downlink packets to or from the MEC, e.g., CNN or RNN at an eNB for application-specific spectrum allocation to downlink packets. - MLFO is recommended to have an interface with the management plane of the ML underlay to interact with the slice manager or controller for application-specific resource allocation for both uplink and downlink packets.
4.17 Smart mirror	- It is recommended that the MLFO is provided with the application requirements and data model. - MLFO is required to collect video stream data in real-time. NOTE 1 – It is not essential to collect all streams, only a small part of the stream without any timing constraints should be enough to retrain the ML model. - MLFO is recommended to manage the storage of data. - MLFO is recommended to take as input the specification of any pre-processing tasks. - MLFO is required to select different types of ML models or learning mechanisms based on stringent application requirements to make useful action predictions. NOTE 2 – E.g., CNN which is a supervised learning model can extract useful knowledge from the input information and can execute actions based on this. MLFO is recommended to make appropriate ML model deployment decisions based on stringent application requirements as given in the intent. NOTE 3 – E.g., moving image processing required to prevent accidents can be deployed at the edge while less critical processing can be shifted to the cloud to avoid longer delays. For example, CNN or RNN can predict events in less time.

Table II.1 – MLFO requirements analysis for the use cases

Corresponding use case in [ML5G-I-227-R2]	Requirement
	- MLFO is recommended to enable training/retraining mechanisms to ensure the accuracy of ML models. NOTE 4 – E.g., offline or online training mechanisms. NOTE 5 – To ensure a short turnaround time, MLFO may select only a sample stream to retrain or improve the ML models. - MLFO is recommended to interface with the management modules of the ML underlay to take actions based on inference while ensuring stringent timing requirements.
4.18 ML based E2E network management for 5G and beyond	- It is recommended that the MLFO is supplied with application requirements and data models in a generic format. - MLFO is required to enable big data collection at different network levels and entry points. NOTE 1 – E.g., the sources of data can include radio, transport, and core networks. NOTE 2 – The data to be collected may include information about alarms, events, inventory, topology, and storage. NOTE 3 – It may also include information related to end-to-end network management including service requests, the status of network resources, path characteristics, topology, and connection information. - MLFO is required to take as input the specification of pre-processing tasks. NOTE 4 – These pre-processing tasks can extract common topology, configuration, and data models from the big data platform. - MLFO is recommended to enable the selection of different types of ML models based on the application requirements, data models, and constraints as specified in the intent. NOTE 5 – MLFO can select supervised, reinforcement, or hybrid models, e.g., MLFO may select DNN, i.e., a supervised learning model with autoencoders and autoregressive NN for fault detection. NOTE 6 – MLFO may select a partially observable Markov decision process (POMDP) to automatically select an optimal recovery action. - MLFO is recommended to support the training and retraining of ML models using different types of training mechanism. NOTE 7 – E.g., offline training or online training. - It is recommended that MLFO has interfaces with the management modules of the ML underlay to perform fault prediction, RCA, and automated recovery at various points in the network. - MLFO is recommended to derive timing constraints for each service or application.

Table II.1 – MLFO requirements analysis for the use cases

Corresponding use case in [ML5G-I-227-R2]	Requirement
	NOTE 8 – MLFO can calculate this timing constraint by using the timing threshold of the application, the prediction time of the ML model, the time to communicate the prediction, and the time to act. MLFO is recommended to interface with the management modules of the ML underlay to controllers to deliver eMBB, uRLLC, and mMTC services.
4.19 ML-aided channel modelling and channel prediction	- MLFO is recommended to be provided with clear application requirements, constraints, or data models. - MLFO is required to enable collection of large amounts of real-time scenario-specific data. NOTE 1 – This data can include scenario-specific measurements, e.g., impulse response, which can be carried out both at the RAN and UE level. NOTE 2 – Scenario-specific data means that the measurements should be at various frequency bands across 0-100GHz, multiple antenna elements at both the RAN and UE, and different propagation environment, e.g., urban macro, urban micro, rural, high-speed train, and indoor, etc. NOTE 3 – The environmental data can include information on a 3D map of the propagation environments, transmitter and receiver locations, identified reflectors and scatters, material EM properties, and surface roughness, etc. NOTE 4 – Other measurements of interest can include system configuration for the measurement set-up, for instance, the antenna pattern and configuration, MIMO mode, operating frequency, transmit power, bandwidth, sampling rate, thermal noise level, and waveform used, etc. NOTE 5 – The UE-specific channel impulse responses estimated by the BS or reported by the UE are collected and stored with time stamps in the correct sequential order. NOTE 6 – It is important to collect information on UE and BS locations including GPS or channel-information-inferred distance, and angular information, etc. - MLFO is required to ensure privacy, security, integrity, accessibility, and availability of data. - MLFO is recommended to take as input the specification of the pre-processing tasks to manipulate data. NOTE 7 – The pre-processing task can convert the data to a standard format to enable information sharing. NOTE 8 – The pre-processing task can categorize the data according to certain attributes, such as environment, frequency, and purpose, etc. NOTE 9 – The pre-processing task should convert the time-series data and data with different dimensions into a format to be fed to the ML model for training. - It is required that the MLFO can select different ML models and learning mechanisms based on requirements and constraints given in the intent to

Table II.1 – MLFO requirements analysis for the use cases

Corresponding use case in [ML5G-I-227-R2]	Requirement
	accomplish different tasks related to channel modelling and channel prediction. NOTE 10 – E.g., MLFO can select clustering, feature extraction, and classification for channel modelling. NOTE 11 – MLFO can select different supervised learning mechanisms to predict the future channel response based on historic and user-specific data. NOTE 12 – Based on the input/output of ML models given in the intent, MLFO calls ML sub-models which are packaged as software/containers and can chain them to form a unified channel modelling process NOTE 13 – MLFO can use distributed learning based on the application requirements given in the intent to elevate the computation demand of a processing node. • MLFO is recommended to have an interface with the close-loop monitoring system. NOTE 14 – This interface can help to improve the performance of the model and be able to fall back to traditional channel estimation if required. • MLFO is required to select a training/retraining mechanism for ML models based on the accuracy and latency constraints as specified in the intent. NOTE 15 – E.g., an offline training mechanism can be used for increasing accuracy at the expense of high latency. • MLFO is required to have an interface with the management plane of ML underlay to interface with the physical layer processes NOTE 16 – This interface can help to configure measurement equipment, e.g., antennas, MIMO mode, frequency band, transmit power, and bandwidth. • MLFO is recommended to have an interface with a protocol which can enable subscription, distribution, and maintenance of a trained model.
4.20 ML-based link adaptation optimization	• It is recommended that the MLFO is given application specifications including requirements and constraints as an input. • MLFO is required to enable collection of data for model training. NOTE 1 – The data may include layer 1 UE level feedback and layer 1 and layer 2 BS measurements. NOTE 2 – Layer 1 UE feedback may include CQI, RI, PMI, downlink RSRP measurement, and ACK/NACK. NOTE 3 – Layer 1 BS measurements may include MIMO precoders, uplink RSRP measurements, CQI/PMI/RI feedback delay, and estimated doppler frequency offset. NOTE 4 – Layer 2 BS measurements which should be collected include PRB usage ratio, scheduled MCS, scheduled RANK, user number, and interference from neighbour cells.

Table II.1 – MLFO requirements analysis for the use cases

Corresponding use case in [ML5G-I-227-R2]	Requirement
	- MLFO is recommended to select different types of pre-processing tasks to handle multi-dimensional data to train the ML models. - MLFO is required to select different types of ML models for predictive link adaptation. NOTE 5 – E.g., MLFO can select a supervised learning mechanism, e.g., RNN to learn complicated relationship from processed data and predict appropriate RANK/MCS for a link. NOTE 6 – These complicated relationships can be between BLER with the CQI, PMI, RI, RSRP, interference, ACK/NACK, etc. - It is recommended that the MLFO can update, train, or retrain the ML model based on the application requirements and constraints as specified in the intent. - MLFO is recommended to have an interface with the management modules of ML underlay. NOTE 7 – This interface can be used by the MLFO to interface with the BS scheduler to deploy trained models. NOTE 8 – MLFO can use this interface to execute predicted MCS and RI selection policies on BS schedulers.
4.21 ML-based E2E network slicing for 5G	- MLFO is recommended to be provided with the application requirements, constraints, and data. - MLFO is required to enable collection of data at various network levels including radio access, core, and transport. NOTE 1 – The radio access network data may include geo-location reading from GPS, measurement reports for radio condition, allocated resources, connection signalling events, classified traffic types, service target indicators per user/node. It may also include cell/node resources utilization and cell/node KPIs. NOTE 2 – The core network data includes KPIs, traffic figures, network utilization KPIs, and additional CDRs, and log files. NOTE 3 – The transport network data may include transport utilization, failures, and bandwidth. NOTE 4 – MLFO can enable this collection with the help of an end-to-end orchestrator-connected universal system to collect network data, counters, and KPIs. NOTE 5 – MLFO can collect some results for transport KPIs, e.g., FLR, delay, and jitter with the integration of active probing monitoring systems connected to the network elements. - MLFO is recommended to perform necessary pre-processing actions on the collected data to transform it into a format suitable to train the ML models.

Table II.1 – MLFO requirements analysis for the use cases

Corresponding use case in [ML5G-I-227-R2]	Requirement
	NOTE 6 – E.g., the pre-processing labels the data according to the service identifier (linked to its ID) per service, which identifies the slice the user is connected to. • MLFO is required to select different types of ML models and learning mechanisms. NOTE 7 – E.g., MLFO can select an unsupervised learning technique, e.g., clustering technique to determine the best slice allocation based on the service type. • MLFO is recommended to deploy the ML models using a centralized or decentralized approach based on application-specific constraints. NOTE 8 – E.g., for latency-sensitive applications, MLFO can deploy the prediction models on the network edge. • MLFO is recommended to enable the specification of training/retraining mechanisms. NOTE 9 – E.g., offline or online training depending on application constraints and system dynamics. NOTE 10 – MLFO can retrain the models based on a model 's actual use and performance and the results of its decisions. • MLFO is recommended to have an interface with the management modules of ML underlay. NOTE 11 – This interface will be used by the MLFO to coordinate with the central orchestrator, distributed orchestrators, transport protocols, and with the NFV and SDN controllers.
4.22 Data-driven architecture for ML at the edge	• MLFO is required to collect network-generated data in real-time at various network levels. NOTE 1 – E.g., MLFO can collect this information on RAN controllers on MEC and by using cloud facilities. NOTE 2 – This data may include network state data including Identification information, number of users per cell, and number of handovers. • MLFO is recommended to perform pre-processing on the collected data without losing awareness of the spatial information. • MLFO is recommended to support different types of training and retraining mechanisms to train ML models. NOTE 3 – This training/retraining can be supported with the help of diverse real-world data sets which can be augmented with simulation-generated data to reflect unseen events. NOTE 4 – The training data must capture correlations such as real-world data that are difficult to model in artificial data sets. • MLFO is required to enable selection of different types of ML models and ML mechanisms to support big-data analytics.

Table II.1 – MLFO requirements analysis for the use cases

Corresponding use case in [ML5G-I-227-R2]	Requirement
	NOTE 5 – MLFO can select a model to control the network itself and provide new services to the users. NOTE 6 – MLFO can select an ML model for real-time predictions and optimizations based on network conditions including congestion, load of base stations, and mobility patterns. NOTE 7 – MLFO can select a model, e.g., supervised or reinforcement or hybrid to perform online predictions, clustering, and network management by capturing spatial correlations. • MLFO is recommended to take deployment decisions for ML models which can be deployed on RAN controller at the edge or cloud. NOTE 8 – E.g., supervised, semi-supervised, reinforcement learning, or hybrid. NOTE 9 – MLFO should also specify another ML model to predict and orchestrate the RAN controllers by using supervised or semi-supervised ML mechanisms. • MLFO is recommended to interface with the management modules of ML underlay to orchestrate the operations of the RAN controllers.
4.23 Long term traffic forecasting	• It is recommended that the MLFO is provided with the application specification including requirements and constraints as an intent. • MLFO enables data collection at different network levels. NOTE 1 – The collected data can include time-stamped measurements, geographical coordinates of the collecting entity. Additional context information that may be relevant includes: cell ID, number of antennas, and average number of connected users/intervals. NOTE 2 – MLFO can collect information with the assistance of dedicated measurement infrastructure (or measurement plugins at the edge, e.g., gNB) level. This may require hardware upgrades or software updates. • MLFO is recommended to manage the storage of data aligned with the privacy guidelines. NOTE 3 – Additional storage might be required at gNB level. Enough storage is required in the core, e.g., at P-GW level. • MLFO is required to enable data collection at base station level for a sufficiently long duration. NOTE 4 – MLFO can perform deployment-based data collection when ported from one deployment to another, e.g., from rural to city. • MLFO is recommended to take as an input the pre-processing tasks to be performed on the collected data. NOTE 5 – E.g., a pre-processing task should be able to represent episodic traffic updates through meta-data as summaries sampled at intervals e.g., every 5-10 minutes.

Table II.1 – MLFO requirements analysis for the use cases

Corresponding use case in [ML5G-I-227-R2]	Requirement
	- MLFO is required to select deep learning models according to various network deployments. NOTE 6 – Ideally, MLFO should select ML models which generalize well under different network deployments, e.g., rural and city. NOTE 7 – For example, MLFO can choose 3D-CNNs or ConvLSTMs to perform network-wide traffic forecasting at a central location, e.g., P-GW. - MLFO is recommended to support different training mechanisms to train the ML models with high accuracy. NOTE 8 – E.g., MLFO can train a model by using offline training and once trained it can use only meta-data to update the models. NOTE 9 – MLFO can enable parallel training capabilities to exploit parallel computing capabilities of GPUs. NOTE 10 – MLFO can retrain the model when ported from one deployment to another, e.g., from rural to city. - MLFO is recommended to take deep learning model deployment decisions according to resource constraints. NOTE 11 – MLFO can perform forecasting at a central location, e.g., P-GW to exploit the benefits of bulk data storage and parallel processing to train the deep learning models. - MLFO is recommended to consider agile resource allocation mechanisms based on past traffic forecasts. - MLFO is recommended to interface with the management modules of ML underlay. NOTE 12 – This interface can be used to communicate with the edge locations, e.g., gNB. NOTE 13 – This interface can be used by the MLFO to communicate with the central locations, e.g., P-GW.
4.24 ML-based utility maximization of sliced BH	MLFO is required to enable data collection of detailed measurements at various network entry points. NOTE 1 – These detailed measurements can include per-service demand. NOTE 2 – The measurement granularity can be adjusted by the MLFO according to application requirements, e.g., latency. NOTE 3 – MLFO can acquire the required measurements through signalling. NOTE 4 – MLFO can collect information with the assistance of dedicated measurement infrastructure (or measurement plugins at the edge, e.g., gNB) level. This may require hardware upgrades or software updates. NOTE 5 – The collected data at the entry point, e.g., gNB should include a node identifier, flow identifier, type of services, KPI required, and information about the capacity available on the link to the next hop.

Table II.1 – MLFO requirements analysis for the use cases

Corresponding use case in [ML5G-I-227-R2]	Requirement
	NOTE 6 – The collected information of the central node may include routing topology information along with flow share, capacity constraints, and eventual scheduling conflicts. • MLFO is recommended to manage the storage of large volumes of meta-data at the entry points. NOTE 7 – E.g., gNB level. • MLFO is recommended to take as an input the specification of the pre-processing task on the collected data. NOTE 8 – For example, pre-processing tasks can formulate the timestamped summaries of continuous flow demand. • MLFO is required to enable selection of a deep learning model or distributed learning models according to the deployment scenario. NOTE 9 – For example, MLFO may select CNN to predict an optimal resource allocation to maximise the utility. • MLFO is recommended to enable training/retraining of deep models based on deployment to ensure required accuracy. NOTE 10 – E.g., offline training of the DL model centrally, e.g., gateway. NOTE 11 – Per-deployment re-training can reflect the particularities of a given topology. NOTE 12 – MLFO can take advantage of data parallelism/model parallelism to exploit parallel computing capabilities of GPUs at the central node (P-GW) using a distributed deep learning approach. • MLFO is recommended to interface with the management modules of ML underlay. NOTE 13 – This interface can be used to communicate with the edge node, e.g., gNB. NOTE 14 – This interface can be used by the MLFO to communicate with the central node, e.g., P-GW.
4.25 Emergency services based on ML	• MLFO is recommended to be provided with the specification of the application including requirements, constraints, and data. • MLFO is required to enable event-driven collection of data. NOTE 1 – The events can be triggered by IoT sensors, cameras, people alerts, or authority warning alerts. NOTE 2 – The collected data may include specific measurements based on an event, e.g., position estimates, external network detection information, and additional information from users, e.g., text, image, or video. • MLFO is recommended to take as an input the specification of the pre-processing tasks to bring the information in a suitable form to train the ML models.

Table II.1 – MLFO requirements analysis for the use cases

Corresponding use case in [ML5G-I-227-R2]	Requirement
	- MLFO is required to select different types of learning mechanisms and ML models to aid different emergency services. NOTE 3 – For example, MLFO can use CNN to correlate information from images, external sources, IoT devices, sensors, and alarms to predict an emergency and generate warning messages. NOTE 4 – MLFO can use RNN to predict a best and safe route based on collected route information. NOTE 5 – For example, MLFO may specify a deep learning model for accurate prediction at the core and a lightweight ML model at the edge to handle emergencies in real-time. - MLFO is recommended to take deployment decisions for the ML models according to the application requirements and constraints as specified in the intent. NOTE 6 – For example, MLFO may deploy a model on a centralized server to handle all inputs from different locations. NOTE 7 – It can deploy an ML model on the edge server located to close or inside an NR node. NOTE 8 – MLFO can also deploy ML models using a hybrid approach, i.e., distributed approach both on the edge and cloud. - MLFO is recommended to train/retrain an ML model based on accuracy and latency requirements of an application. NOTE 9 – For example, MLFO can use an offline training method to train a model in a centralized manner on a server, while edge servers can only initiate re-training of the ML model. - MLFO is recommended to interface with the management modules of ML underlay. NOTE 10 – This interface can help MLFO to interact with the common alerting protocol (CAP) to exchange hazard, emergency, and public warnings or alerts over all kinds of networks.
4.26 Energy efficient trusted multi-tenancy in 5G-crosshaul	It is recommended that the MLFO is provided with the application specifications including requirements, constraints, flexible data model, and service level agreements (SLAs). NOTE 1 – The SLAs can help the MLFO to allocate virtual resources to tenants based on the agreements. NOTE 2 – The specific SLAs required for multi-tenancy should be technology independent. MLFO is required to enable data collection to provide situational awareness of the environment. NOTE 3 – This may involve a set of activities to acquire operational environment data.

Table II.1 – MLFO requirements analysis for the use cases

Corresponding use case in [ML5G-I-227-R2]	Requirement
	NOTE 4 – MLFO should consider the collection of important information across the network for each slice including features such as: throughput, congestion, transmission delays, availability, risk level, and registered events. NOTE 5 – Sensors and external data repositories may act as critical sources of factual knowledge. • MLFO is recommended to support knowledge representation and reasoning strategy. NOTE 6 – This strategy can be used by the MLFO to fetch, manage, and process operational and factual knowledge along with meta-knowledge. • MLFO is recommended to support essential pre-processing tasks to process the data in an appropriate format. NOTE 7 – These tasks may involve labelling, normalization, aggregation/fusion, verification, prioritization, and reconstruction. NOTE 8 – Additionally, pre-processing is expected to form specific event-related correlations. NOTE 9 – The pre-processing tasks may consider case-based/attribute-based reasoning to find dependencies between events and resource vectors. • MLFO is required to enable selection of different types of ML models and learning mechanisms to enable trusted multi-tenancy across the 5G-Crosshaul. NOTE 10 – MLFO can select a supervised ML model to classify an application or service type and allocate virtual resources to tenants according to SLAs. • MLFO is required to enable isolation of virtual resources allocated to tenants. • MLFO is recommended to assure seamless allocation of virtual resources to tenants without any service disruption to existing virtual network operators (VNOs). NOTE 11 – This isolation should enable the VNO to specify its addressing space, deploy a network operating system of its own choice, along with its own virtual topology. • MLFO is required to calibrate different ML mechanisms to filter and countermeasure security threats. NOTE 12 – MLFO may select a lightweight ML algorithm at the midhaul to filter security threats. NOTE 13 – MLFO can enable a reinforcement learning-based model to facilitate context-aware recalibration at the FH. NOTE 14 – MLFO can enable adaptive ML models, e.g., reinforcement learning, deep learning, or hybrid at the BH to aid in countermeasure decisions with considerations of future events, threats, their replicas, and possible propagation holes in a multi-stakeholder multi-tenant environment.

Table II.1 – MLFO requirements analysis for the use cases

Corresponding use case in [ML5G-I-227-R2]	Requirement
	NOTE 15 – These ML models should provide anticipatory responses as actions according to future expected events, evolution, and propagation of discovered threats. NOTE 16 – In the context of beyond 5G, the ML models at the core should bring continuous and collaborative decision-making to offer a service in a multi-stakeholder scenario. • MLFO is recommended to support a closed loop system, where expected key performance indicators (KPIs) can be assessed and appropriate corrective tactics can be taken. NOTE 17 – These tactics may include modification of statistical models, data pre-processing, inference rules, and recalibration of heuristics. • MLFO is recommended to support a metacognition layer to optimize its operation under unforeseen threats/events. NOTE 18 – E.g., This layer can monitor ML models and control them. NOTE 19 – It can dynamically acquire and validate the consequences of decision-making with the help of operational knowledge. NOTE 20 – This layer should incorporate efficient onboarded security and energy policies complemented with situational awareness to take appropriate resource allocation decisions for each tenant. • MLFO is required to enable different types of ML models to enforce contextual adaptation of trust models for trusted execution environment (TEE). NOTE 21 – E.g., supervised learning, reinforcement learning, or hybrid. This will ensure timely detection of virtual network function (VNF) anomalies and will prevent the propagation of threats. NOTE 22 – Convolutional neural networks (CNN) can be used to enable trust by enforcing appropriate levels of security and trust. • MLFO is required to support multi-tenancy with self-awareness, self-healing, self-optimization, and self-protection. NOTE 23 – MLFO can ensure this with the help of lightweight ML models • MLFO is required to provide countermeasures to cloud-level threats at the tenant-level to ensure trust. NOTE 24 – MLFO can provide this with the assistance of deep learning ML. MLFO is recommended to provide flow-level security driven by authorization/privacy policies at the edge in real-time. NOTE 25 – These ML-enabled security mechanisms should comply with the trust model and energy-saving policies required by different 5G actors, e.g., slice owners, telcos, and verticals. • It is recommended that MLFO supports various ML mechanisms to audit data integrity.

Table II.1 – MLFO requirements analysis for the use cases

Corresponding use case in [ML5G-I-227-R2]	Requirement
	NOTE 26 – This data integrity can be provided for each slice, e.g., resilience, and VNF anomaly detection. • MLFO is required to comply with the ETSI enhanced high-level NFV security management framework to entail energy-efficient security measures. • MLFO is required to select different types of mechanisms to train/retrain the ML models depending on the resource constraints. NOTE 27 – Offline training may require dedicated physical infrastructure with virtualization capabilities equipped with enough computing resources. • MLFO is recommended to interface with the management plane of ML underlay to interact with the NFVO.
4.27 Network slice SLA assurance based on ML	• It is recommended that the MLFO is provided with the application requirements, constraints, data, and SLAs. • MLFO is required to enable continuous collection of (large amounts of) traffic data or measurements. NOTE 1 – MLFO can support this collection with the assistance of OAM tools. NOTE 2 – The MLFO data collection should constantly measure SLA-relevant (service) KPIs from RAN, CN, Transport or combinations. • MLFO is recommended to take as an input the pre-processing task to process the collected data. NOTE 3 – E.g., labelling of data based on service and application type. • MLFO is required to enable selection of different types of ML models and algorithms to provide the level of assurance for the slice according to the SLA. NOTE 4 – E.g., supervised learning mechanisms. NOTE 5 – KPIs obtained through the collected data can help to predict the level of assurance for the slice. NOTE 6 – MLFO can take actions to fulfil SLAs in case the agreed conditions for the slice are not fulfilled. • MLFO is recommended to train/retrain the ML models based on the resource constraints. NOTE 7 – E.g., offline training mechanisms. • MLFO is recommended to have an interface with the management modules of ML underlay. NOTE 8 – This interface can be used by the MLFO to interact with measurement systems, e.g., SBA/SBI.

Table II.1 – MLFO requirements analysis for the use cases

Corresponding use case in [ML5G-I-227-R2]	Requirement
4.28 Service management for smart cities	- It is required that the MLFO is provided with traffic and existing services specifications offered by the operators, i.e., labelled data. NOTE 1 – The traffic categories may include mIoT, URLLC and eMBB. - MLFO is required to provide interoperable transfer of knowledge among operators. NOTE 2 – E.g., cross-slice pollination of smart behaviour in the network.
4.29. Automated testing of services	- MLFO is required to support interoperable and open testing and monitoring interfaces. - MLFO is required to support automated testing of ML models in a standalone environment without affecting the operator KPIs. - MLFO is recommended to consider local regulatory policies to manage services and data before deploying the network.
4.30. Combating use of counterfeit ICT devices – ML assisted network service	- MLFO is required to enable continuous data collection. - MLFO is recommended to support pre-processing to label the traffic according to the IMEI and device capabilities. NOTE – These device capabilities may include antenna capability, MIMO capability, etc. - MLFO is recommended to continuously update edge nodes based on the classification information in a timely manner.
4.31. ML based identification of illegal exchanges using SIM boxes	- It is required that MLFO continuously collects the real time data. - MLFO is recommended to learn patterns of collected data to build traffic model for traffic. NOTE – These patterns may include VOIP traffic from illegal exchanges. - MLFO is recommended to share the information about the illegal exchanges to the operator-configured authority.

Appendix III

Examples of ML intent

A basic intent should contain the following:

Training phrases: These phrases can be included by the end-user. During the training phase, these phrases should be referred to by the model management module.

Pre-processing phrases: User can add expressions to specify any pre-processing requirements or constraints including data models.

Parameter phrases: The end-user can specify the parameters as expressions. The end-user can include an expression for the parameter which corresponds to an entity type that dictates how exactly data is extracted?

Responses: An intent parser can return responses to return to the end-user. These responses may ask the end-user for any further information.

Contexts: The MLFO intent parser can handle the context of any end-user expressions to correctly process an intent.

Policies: The end-user can specify any policies and the level on which these policies should be considered.

Appendix IV

Executive summary

The main objective of this deliverable is to extract and analyse high-level requirements from the detailed study of a wide range of use cases as given in the supplement document [ITU-T Y.Sup55]. These requirements are extracted after illustrating detailed usage scenarios of machine learning for 5G and beyond. This deliverable also extracts high-level requirements to facilitate integration and operation of the data handling framework, ML marketplace integration, mobile network intelligence evaluation, the serving framework for ML models and machine learning sandbox with the MLFO. The proposed MLFO architecture is intended to orchestrate the operation of machine learning pipeline across the network while providing integration with different ML frameworks.

The various elements of the deliverable are summarised as follows:

- The Technical Specification discusses the challenges experienced by network operators and data scientists in orchestrating the operation of the ML pipeline across the network. These challenges have been identified from the study of different ML use cases and ML frameworks. This deliverable also identifies other challenges related to the integration of the data handling framework, ML marketplace integration, mobile network intelligence evaluation, the serving framework for ML models, and the machine learning sandbox. The current ML pipeline orchestration solutions cannot be integrated with these frameworks.
- This Technical Specification also illustrates current ML pipeline solutions and the integration issues of these solutions with the data handling framework, ML marketplace integration, mobile network intelligence evaluation, the serving framework for ML models and machine learning sandbox. These solutions are not standard and therefore can result in complex and expensive ML pipeline orchestration across the network. It may also result in glue code, hidden dependencies, feedback loops, and pipeline nets. The current ML pipeline orchestration solutions do not offer ML model update, ML pipeline chaining, ML pipeline performance monitoring and evaluation, policy-based ML pipeline deployment, and management/coordination of multiple ML pipeline instances across the network.
- It also discusses the major motivation to offer machine learning pipeline orchestration as a standard solution to provide flexibility to network operators in terms of ML pipeline orchestration/management and integration with the existing ML frameworks.
- This deliverable also identifies high-level requirements which should be considered for the MLFO framework. MLFO architecture meets these requirements and addresses the challenges posed by diverse 5G and beyond use cases while integrating with data handling framework, ML marketplace integration, mobile network intelligence evaluation, serving framework for ML models and machine learning sandbox.
- It also presents the system sequence diagrams of the high-level requirements to depict the interactions in an ML pipeline and other ML frameworks. These interactions can be within an ML pipeline and can also be triggered by the MLFO or by other ML frameworks.
- It proposes a high-level architecture for the machine learning framework orchestrator that can be used as a reference to orchestrate the operation of an ML pipeline with easy integration with other ML frameworks including the data handling framework, ML marketplace integration, mobile network intelligence evaluation, the serving framework for ML models and the machine learning sandbox
- The deliverable also describes different reference points that are required between the components of MLFO and other ML frameworks.

Bibliography

- [b-ITEH] Network Functions Virtualisation (NFV); NFV Security; iTeh Standards
<https://www.etsi.org/deliver/etsi_gs/NFV-IFA/001_099/026/05.01.01_60/gs_NFV-IFA026v050101p.pdf>
- [b-ML-Schema-Gustavo] Gustavo Correa *et al.* (2018), *ML-Schema: Exposing the Semantics of Machine Learning with Schemas and Ontologies in Poster*, selected for the 2nd Reproducibility in Machine Learning Workshop at ICML, Stockholm, Sweden.
- [b-ML5G-I-148] China Mobile, ZTE, USTC, Turkcell, Vishnu Ram OV (2019), *Framework for data handling to enable Machine Learning in future networks including IMT-2020*, published 09/05/2019.
- [b-pervez-Jaber17] F. Pervez, M. Jaber, J. Qadir, S. Younis, and M. A. Imran, *Fuzzy Q-learning-based user-centric backhaul-aware user cell association scheme*, in Proc. 13th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC), Jun. 2017, pp. 1840–1845.
- [b-yang-Li18] S. Xu, R. Li, and Q. Yang (2018), *Improved genetic algorithm based intelligent resource allocation in 5G Ultra Dense networks*, in Proceedings of the 2018 IEEE Wireless Communications and Networking Conference (WCNC), pp. 1-6, Barcelona, April 2018.
- [b-Li-Zhang16] M Li, L Zhang, Z Xiang, E Castillo, T Guerrero, in 2016 International Conference on Progress in Informatics and Computing (PIC). *An improved fuzzy C-means algorithm for brain MRI image segmentation*. (IEEE Shanghai, 2016), pp. 336–339.
- [b-Dsilva] G. M. D'silva, A. Khan, S. Bari, *et al.* (2017), *Real-time processing of IoT events with historic data using apache kafka and apache spark with dashing framework*, in Recent Trends in Electronics, Information & Communication Technology (RTEICT), 2017 2nd IEEE International Conference on, pp. 1804–1809, IEEE, 2017.
- [b-Vera-Baquero] A. Vera-Baquero and R. Colomo-Palacios (2018), *Big-data analysis of process performance: A case study of smart cities*, in Big Data in Engineering Applications, pp. 41–63, Springer.
- [b-Rathore] M. M. Rathore, A. Ahmad, and A. Paul (2016), *Iot-based smart city development using big data analytical approach*, in Automatica (ICA-ACCA), IEEE International Conference on, pp. 1–8, IEEE.
- [b-Kwok] J. Kwok and Y. Sun (2018), *A smart iot-based irrigation system with automated plant recognition using deep learning*, in Proceedings of the 10th International Conference on Computer Modeling and Simulation, pp. 87–91, ACM.
- [b-Mohammadi] M. Mohammadi, A. Al-Fuqaha, M. Guizani, and J.-S. Oh (2018), *Semisupervised deep reinforcement learning in support of iot and smart city services*, IEEE Internet of Things Journal, vol. 5, no. 2, pp. 624–635.

- [b-ruiz18] C. Ruiz, S. Pan, A. Sadde, H. Y. Noh, and P. Zhang (2018), *Posepair: pairing iot devices through visual human pose analysis: demo abstract*, in Proceedings of the 17th ACM/IEEE International Conference on Information Processing in Sensor Networks, pp. 144–145, IEEE Press.
- [b-Sculley15] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, and D. Dennison (2015), *Hidden technical debt in machine learning systems*, in Advances in neural information processing systems, pp. 2503–2511.
-