International Telecommunication Union

ITU-T

Technical Specification

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

(01/2019)

Focus group on Machine Learning for Future Networks including 5G (FG-ML5G)

FG-ML5G-ARC5G

Unified architecture for machine learning in 5G and future networks


International
Telecommunication
Union

Technical Specification ITU-T FG-ML5G-ARC5G

Unified architecture for machine learning in 5G and future networks

Summary

This document defines a unified architecture for Machine Learning in Fifth Generation and future networks. A comprehensive set of (architectural) requirements is presented, which in turn leads to specific architecture constructs needed to satisfy these requirements. Based on these constructs, a logical ML pipeline (along with the requirements derived and its realizations in various types of architectures) is discussed. Finally, key architectural issues facing the integration of such an ML pipeline into continuously evolving future networks are listed.

Keywords

Machine learning, 5G, architecture, ML, training, requirements, use cases, overlay.

Table of Contents

Technical Specification ITU-T FG-ML5G-ARC5G

Unified architecture for ML in 5G and future networks

1 Scope

This document defines a unified architecture for Machine Learning in Fifth Generation and future networks. The unified logical architecture establishes a common vocabulary and nomenclature for ML functions and their interfaces. By applying (or superimposing) this logical architecture to a specific technology, like 3GPP, MEC, EdgeX or transport networks, the corresponding technology-specific realization is derived. Such an overlay architecture for ML allows standardization and achieves interoperability for ML functions in 5G and future networks. In addition, the key issues for integration of such an overlay on 5G and future networks are studied and presented in this document.

2 References

[1] 3GPP TS 23501, System Architecture for the 5G System (Release 15).

[2] Broadband: Acronyms, Abbreviations & Industry Terms https://www.itu.int/osg/spu/ni/broadband/glossary.html

[3] Edgex Wiki https://wiki.edgexfoundry.org/display/FA/Introduction+to+EdgeX+Foundry

[4] IEC whitepaper on Edge Intelligence http://www.iec.ch/whitepaper/pdf/IEC_WP_Edge_Intelligence.pdf

[5] ETSI GS NFV-IFA 014 V2.3.1 (2017-08), Network Functions Virtualisation (NFV) Release 2; Management and Orchestration; Network Service Templates Specification.

[6] Intent NBI – Definition and Principles, Open Networking Foundation, ONF TR-523.

[7] ETSI GS MEC 003 V1.1.1 (2016-03).

[8] https://github.com/cncf

[9] Homing and Allocation Service (HAS) https://wiki.onap.org

[10] ETSI SOL002, Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; RESTful protocols specification for the Ve-Vnfm Reference Point.

[11] ETSI GS ZSM 001 V0.4.0 (2018-11), Requirements on the zero-touch end-to-end network and service management.

[12] Recommendation ITU-T Q.5001 (ex. Q.IEC-REQ), Signalling requirements and architecture of intelligent edge computing.

[13] 3GPP TS 28.554, Management and orchestration of 5G networks; 5G End to end Key Performance Indicators (KPI) (Release 15).

3 Definitions

3.1 Terms defined elsewhere

3.1.1 mobile edge computing (MEC): A mobile edge system that enables mobile edge applications to run efficiently and seamlessly on a mobile network.

NOTE – Source: [7].

3.1.2 EdgeX Foundry: A vendor-neutral open source project hosted by the Linux Foundation, which builds a common open framework for IoT edge computing.

NOTE – Source: [3].

3.1.3 capability exposure: The network exposure function (NEF) supports external exposure of capabilities of network functions (NFs). External exposure can be categorized as monitoring capability, provisioning capability, and policy/charging capability.

NOTE – For NEF details, see [1].

3.2 Terms defined in this document

3.2.1 machine learning pipeline: A set of logical entities (each with specific functionalities) that can be combined to form an analytics function.

Machine learning (ML) pipeline node: Each functionality in the ML pipeline is defined as a node (e.g., source, collector, pre-processor, model, policy, distributor and sink).

- src (source): This node is the source of data that can be used as input for the ML function. Examples of an src are: user equipment (UE), session management function (SMF) [1], access and mobility management function (AMF) [1] or any other entity in the network, including an application function (AF) [1].
- C (collector): This node is responsible for collecting data from the src. It can use specific control protocols to configure the src. Example: it may use the 3rd Generation Partnership Project (3GPP) radio resource control (RRC) protocol [1] to configure user equipment (UE) acting as an src. It may use vendor specific operations, administration and maintenance (OAM) protocols to configure an AMF acting as an src.
- PP (pre-processor): This node is responsible for cleaning data, aggregating data or performing any other pre-processing needed for the data so that are is in a suitable form for the ML model to consume it.
- M (model): This is an ML model. Example could be a prediction function.
- P (policy): This node provides a control for an operator to put a mechanism to minimize impacts into place on a live network, so that operation is not impacted. Specific rules can be put in place by an operator to safeguard the sanity of the network, e.g., major upgrades may be done only at night time or when traffic is low.
- D (distributor): This node is responsible for identifying the sinks and distributing the ML output to the corresponding sinks. it may use 3GPP RRC protocol [1] to configure a UE acting as a sink.
- Sink: This node is the target of the ML output, on which it takes action, e.g., a UE adjusting the measurement periodicity based on ML output.

NOTE – The nodes are logical entities that are proposed to be managed in a standard manner (by a machine learning function orchestrator (MLFO), see clause 3.2.3) and hosted in a variety of network functions (NFs).

The realization of such an ML pipeline (in, say, 3GPP release 16 (R16) or R17 networks) will result in a standard method of introducing and managing ML functionality in a 5G network.

When the symbol in Figure 1 is used, it denotes a subset (including proper subset) of nodes in the pipeline.

Figure 1 — Symbol used to denote the ML pipeline in general

3.2.2 interface 8: A multi-level, multi-domain collaboration interface between nodes of an ML pipeline that allows the ML pipeline to be disaggregated and distributed across domains, e.g., edge and core cloud.

NOTE 1 – This is a flexible, logical interface, whose realization may depend on extending some of the existing interfaces, say in 3GPP, mobile edge computing (MEC), EdgeX or other specific platforms.

NOTE 2 – See Figures 4 and 11.

3.2.3 machine language function orchestrator (MLFO): A logical orchestrator that can monitor and manage the ML pipeline nodes in the system. An MLFO selects and reselects the ML model based on its performance. The placement of various ML pipeline nodes, based on the corresponding capabilities and constraints of the use case, is the responsibility of the MLFO. (See [9] for a general discussion on placement)

3.2.4 chaining: The process of connecting ML functions or nodes together to form the complete ML pipeline. For example, an src instantiated in the distributed unit (DU) can require connection to a collector and PP in the centralized unit (CU) and they in turn, to a model in the core network (CN) to implement the mobility pattern prediction (MPP) use case. The chain itself is declared by the network operator (NOP) in the use case specification in the intent and its technology-specific implementation in the network is done by the machine learning function orchestrator (MLFO). The MLFO utilizes the constraints (e.g., timing constraints for prediction) defined in the intent to determine the placement and chaining.

NOTE – For "intent", see clause 3.2.6.

3.2.5 machine learning meta language (ML-ML): Language used to specify the constructs needed to add the ML use case and ML pipeline in a declarative fashion into the service design.

NOTE – For "service design", see [5].

3.2.6 intent: A declarative mechanism ([6]) which is used by network operator (NOP) to specify the machine learning use case. Intent does not specify any technology implementation methods, such as 3rd Generation Partnership Project (3GPP), or mobile edge computing (MEC) network functions (NFs) to be used in the machine learning (ML) use case. Hence the use cases are technology independent and provide a basis for mapping ML use cases to diverse technology-specific avatars. Intents can use machine learning meta language (ML-ML) to define use case constructs.

3.2.7 sandbox domain: This is a domain internal to the network operator (NOP) in which machine learning (ML) models can be trained, verified and their effects on the network studied. A sandbox domain can host the monitor-optimize loop, also called the closed-loop, and can use a simulator to generate data needed for training or testing, in addition to utilizing data derived from the network.

4 Abbreviations and acronyms

NOTE: See [2].

5GC	5G Core
AF	Application Function
AMF	Access and Mobility management Function
API	Application Programmer Interface
AR/VR	Augmented Reality/Virtual Reality
BOSS	Business and Operation Support System
C	collector (ML pipeline)
CN	Core Network
CNCF	Cloud Native Computing Foundation
CNF	Cloud-native Network Function
CSI	Channel State Information
CU	Centralized Unit
CUDA	Centralized Unit Data Analytics

DB	Database
DNN	Deep Neural Network
DU	Distributed Unit
DUDA	Distributed Unit Data Analytics
E2E	End-to-End
EGMF	Exposure Governance Management Function
EMS	Element Management System
FMC	Fixed Mobile Convergence
GPU	Graphic Processor Unit
HAS	Homing Allocation Service
IoE	Internet of Energy
IoT	Internet of Things
KPI	Key Performance Indicator
M	Model (ML pipeline)
MANO	Management and Orchestration
MEC	Mobile Edge Computing
mIoT	massive Internet of Things
MIMO	Multiple Input Multiple Output
ML	Machine Learning
MLFO	Machine Learning Function Orchestrator
ML-ML	Machine Learning Meta-Language
MPP	Mobility Pattern Prediction
MnS	Management Service
NEF	Network Exposure Function
NF	Network Function
NFV	Network Function Virtualization
NFVI	Network Function Virtualization Infrastructure
NFVO	Network Function Virtualization Orchestrator
NWDAF	Network Data Analytics Function
NMS	Network Management Subsystem
NOP	Network Operator
NSI	Network Slice Instance
OAM	Operations, Administration and Maintenance
ONAP	Open Networking Automation Platform
OSS	Operation Support System
OTT	Over The Top
P	Policy (ML pipeline)

P-GW	Packet Gateway
PNF	Physical Network Function
PoP	Point of Presence
PP	Pre-processor (ML pipeline)
QoS	Quality of Service
RAN	Radio Access Network
RCA	Root Cause Analysis
RDA	RAN Data Analytics
RRC	Radio Resource Control
SBA	Service-Based Architecture
SDO	Standards Developing Organization
Sim	Simulator
SMF	Session Management Function
SO	Service Orchestration
SON	Self-Optimizing Network
src	source (ML pipeline)
SS	Supporting Service
UE	User Equipment
V2X	Vehicle-to-everything
VIM	Virtualization Infrastructure Manager
VNF	Virtual Network Function
VNFM	Virtual Network Function Manager
VoLTE	Voice over Long Term Evolution
WIM	Wireless Infrastructure Network
ZSM	Zero Touch Network and Service Management

5 Conventions

In this document, 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 keywords "can optionally" and "**may**" indicate 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 Executive summary

There is a variety of ways in which ML can be incorporated into 5G and future networks. Any of the layers or strata or network functions (NFs) could act as touch points (either as a source of data that could aid ML or as a target for control mechanisms that implement ML decisions) for ML.

The analysis of various architecture variations has led to a unified, logical architecture that represents technology-agnostic requirements. This logical architecture could be instantiated using technology-specific realizations: 3GPP is a technology-specific realization, others could be MEC, EdgeX, etc.

The unified logical architecture establishes a common vocabulary and nomenclature for ML functions and their interfaces. By applying (or superimposing) this logical architecture to a specific technology, like 3GPP, MEC, EdgeX or transport networks, the corresponding technology-specific realization is derived. This provides a unique ability to analyse both technology-agnostic and specific issues, arrive at general solutions that can be standardized (in ITU) and reused elsewhere (3GPP, MEC, EdgeX).

This document lists the requirements for a ML overlay architecture. Based on these requirements, a unified logical architecture is introduced.

The high-level requirements, listed in Table 1, provide specifications for various aspects of proposed logical architecture. The corresponding aspect of each requirement is listed under the relevant "Req. applicant" row. These requirements are neither exhaustive nor mutually exclusive. Read these requirements as a set of specifications for the corresponding aspect of the architecture.

An ML application can be realized by instantiating logical entities of the ML-pipeline with specific roles (e.g., src, collector, sink), and distributing these entities among NFs specific to the technology (e.g., 3GPP virtual network functions (VNFs)), based on the corresponding requirements of the logical entities (e.g., a traffic classifier requires to be fed with data summaries every *x* ms) and capabilities of the node (e.g., computing power at the edge).

The flow of information in an ML-based use case can be represented by an ML-pipeline. Take MPP as an example which is used in Figure 3. The ML algorithms for MPP require data that are correlated to the location and data-usage patterns of the user. These data might be obtained from various levels of the network. For example, at the radio access network (RAN) level, the multiple input multiple output (MIMO) channel state information (CSI) measurements give the location bearing of the user with respect to the base station, while at the transport level, the user data-usage patterns can be obtained. The data collected at various collection points (source) need to be gathered (by a collector) and pre-processed (by a pre-processor) before feeding these data to the ML algorithm (model). The output of the ML algorithm is then used to apply policies (policy) that will be implemented (sink). Note that, although the location of source(s) and sink(s) depends on the ML-use case, location of other functionalities, e.g., collector, pre-processor, ML algorithm, are not specific to any use case. Instead these functionalities can be seen as logical entities whose placement (virtualized) depends on the capabilities of the network and requirements of the use case [9]. This logical representation of an ML-based network application is called the ML-pipeline.

The high-level logical architecture (see Figure 2) has three main components: the management subsystem, the multi-level ML pipeline, and the closed loop subsystem. Together, these subsystems facilitate the functionality and interfaces needed for ML in future networks. These functions defined in the logical architecture are logical functions, these blocks could be hosted in various technology-specific NFs and the corresponding interfaces could be realized using extensions of existing technology-specific interfaces. These technology-specific avatars of the logical architecture are depicted in Figures 3, 4, etc.

Several key issues related to ML in 5G and future networks are listed in clause 9. These key issues may give rise to new areas of research and extensions of current standards to support future networks, which are beyond the scope of this document. In essence, the following topics are important.

- Need for an ML-ML. This will provide an interoperable, declarative mechanism to specify the "intent" of the use case that uses ML in 5G.
- The level of capability exposure needed to enable dynamic ML-based use cases in 5G and future networks. The characteristics of capability exposure mechanisms for future networks need to be studied, especially in the context of migrating existing networks to future ML-based ones.
- MLFO: A logical orchestrator that can be used to monitor and manage the ML pipeline nodes in the system. Operations on ML in future networks will be controlled via the MLFO (e.g., compression, scaling, chaining, updates, optimizations).

In the remainder of this document, clause 7 specifies high-level requirements, which are derived from various use cases and architectural considerations. Clause 8 specifies both a logical architecture and its technology-specific realizations. Clause 9 lists key issues encountered while analysing these concepts.

7 High-level requirements

See Table 1.

Table 1 – High-level requirements

ML-unify-001	Multiple sources of data are recommended to be used to take advantage of correlations in data.
Req. applicant	Core/general.
Significance/description	In future networks, sources of data may be heterogeneous, integrated with different NFs, and may report different formats of data. These varied "perspectives" can provide rich insights upon correlated analysis. Example: Analysis of data from UE, RAN, CN and AF is needed to predict potential issues related to quality of service (QoS) in end-to-end user flows. Thus, an architecture construct to enable the ML pipeline to collect and correlate data from these varied sources is needed.
Traceability/examples	Examples of such sources of data are self-optimizing network (SON) modules that monitor and listen to all network alarms and key performance indicators (KPIs), and then takes proper action to clear alarms or enhance network KPIs, or give network design recommendations without human intervention.

ML-unify-002	Multiple technologies and network layers (RAN, core, transport, 2G, 3G, future networks) are required to be supported, even non-3GPP external elements are recommended to be interfaced with, to achieve end-to-end user experience.
Req. applicant	Core/general.
Significance/description	Future networks will have multiple technologies coexisting side by side, e.g., licensed and unlicensed wireless technologies, fixed mobile convergence (FMC) technologies, legacy and future technologies. The emergence of network slicing [1] is one example in which vertical technologies [e.g., vehicle-to-everything (V2X)] and their integration into future networks are important. Thus, it is important for that architecture to be capable of overlay with multiple underlying technologies (e.g., 3G, 4G, 5G) and even support application functions like in-car entertainment or streaming data from drones or augmented reality/virtual reality (AR/VR) headsets. The 5G end to end KPIs are defined in technology-specific standards (e.g., see [13]).

Traceability/example	In some use cases, the data from network elements beside some other external elements, e.g., sensors, power circuits and different Internet of things (IoT) modules, are utilized to control and monitor these elements. These data are then used in various types of network parameter optimizations to achieve gains in coverage, capacity and quality.

ML-unify-003	The network architecture is required to support multi-level and distributed instantiation of the ML pipeline. Data from different levels may be able to enrich multiple ML pipelines (algorithms) as needed. ML pipelines are required to be multi-level or multi-domain, connected via logical interface 8. An ML pipeline may be instantiated in multiple levels (e.g., core, edge, MEC).
Req. applicant	Core/general
Significance/description	Cloudification, service-based architecture (SBA), flexible homing allocation service (HAS) imply that various network functionalities can be placed dynamically in multiple levels, domains and clouds. Thus, the ML pipeline should also be able to span these levels and interface accordingly. The functionality of the ML pipeline would also span these levels, domains and clouds based on the homing allocation criteria.
Traceability/example	In some use cases, feature extraction (src) may be placed in a packet gateway (P-GW) and feature data are sent (over interface 8) to MEC where the deep learning model is hosted. Analytics (ML pipeline) may be hosted in the network at multiple levels (e.g., RAN, AMF), utilizing locally available data as input while performing MPPs or slice configurations.

ML-unify-004	When realizing an ML application using the ML pipeline, it is recommended that the number of logical entities impacted is kept to a minimum. In the ideal case, it is required to impact only the src and sink of the ML pipeline.
Req. applicant	Core/general
Significance/description	The use case definition for ML in future networks will be done based on existing (or new) services or functions in the network. The use case definition for ML has to be loosely coupled with the network service definition in future networks. Note: Based on the use case definition (see the intent), homing and other characteristics of the ML pipeline (e.g., chaining) are decided. The src and the sink are points of tight integration (e.g., application programmer interfaces (APIs)) with the technology-specific NFs (e.g., 3GPP RAN). The other nodes in the ML pipeline may be generic and do not have tight integration with technology-specific NFs. Clear interface points between the ML pipeline and the underlying technology are proposed (at the source or sink or MLFO).
Traceability/example	In some use cases, feature extraction (src) and traffic classification (sink) may be placed in the user plane of the P-GW. These are points where user plane data are handled by the ML pipeline nodes. The other nodes in the ML pipeline (e.g., the deep neural network (DNN) model) do not have such interface dependencies on the 3GPP NFs.

ML-unify-005	Logical entities of the ML pipeline are required to be capable of splitting their functionalities or be hosted on separate technology-specific nodes. Similarly, multiple logical entities are required to be capable of being implemented on single node.
Req. applicant	Core/general
Significance/description	In future networks, HAS for NFs will optimize the location and the performance accordingly. The network function virtualization orchestrator (NFVO) plays an important role in this. To carry forward such benefits to the ML use case, similar optimizations should also be applied to ML pipeline nodes. Moreover, the constraints applicable to an ML pipeline [e.g., training may need a graphic processor unit (GPU) and may need to be done in a sandbox domain] may be unique.
Traceability/example	Figure 3 gives scenarios in which, depending on the latency budget, data availability and other considerations for MPP, the ML pipeline could have the src and model hosted in the core, edge or MEC.

ML-unify-006	An interface between ML pipelines of multiple levels may transfer trained models. (This may be one functionality of interface 8)
Req. applicant	Interface-8
Significance/description	Training models has certain specific needs, e.g., availability of certain kinds of processors, availability of data. Once the training is done, it has to be sent to the NF that is hosting the model. This could be UE, RAN or CN as examples from 3GPP. Training can be done separately from the live network. Thus, sending trained models across multiple levels (e.g., core, edge) is an important requirement.
Traceability/example	Figure 10 describes the scenario where distributed unit data analytics (DUDA) host real-time RAN data collection and pre-processing, prediction, parameter optimization and training tasks with low computational complexity in the DU. DUDA need to offer the data features requested for training prediction/decision models to the centralized unit data analytics (CUDA) after pre-processing, while CUDA can assist DUDA to conduct some computationally intensive model training tasks. The trained model can be sent to the DUDA for deployment. In some use cases, deep learning may be done at the MEC and trained model (classifier) updated at the P-GW.

ML-unify-007	Interface between ML pipelines of multiple levels may transfer data for training or testing models. (This may be one functionality of interface 8)
Req. applicant	Interface-8
Significance/description	Certain domains in which the data are available may not have the training capabilities (e.g., resource-constrained edge networks [4]). In such cases, there may be a need to send data for training or testing across to domains where the capacity for such operations is available (e.g., a central data centre).
Traceability/example	See Figure 10, Figure 13. Feature extraction from the user plane may be done at the P-GW and the feature data is then sent to the model at the MEC for training.

ML-unify-008	Potential to place/host the ML pipeline in a variety of NFs (e.g., CN, MEC, network management subsystem (NMS)) in a flexible fashion is required. Decoupling of the location of the logical ML pipeline nodes from their functionality, except in the case of performance constraints, is recommended.
Req. applicant	MLFO, placement
Significance/description	An orchestrator function that understands the needs and constraints of ML functions is needed to place or host the ML pipeline nodes at appropriate NFs. Constraints could include availability of data that is specific to the use case, data transformation capabilities, performance constraints, training capabilities and model characteristics (e.g., if the model is a neural network, then a GPU-based system is desirable). Capability exposure is needed for placement and MLFO exploits this to achieve placement.
Traceability/example	Figure 3 describes the case in which, based on the requirements of the use case, short-term or long-term predictions, the ML pipeline nodes may be hosted closer to the edge or in the CN. The placement may also be influenced by data availability considerations.

ML-unify-009	Certain interfaces (e.g., interface 8) may be realized or extended using existing protocols (e.g., RRC, GS MEC 011, management service (MnS) [7]. Src and Sink may need specific interfaces or APIs to extract data or configure parameters. For example, an src running in the UE may use specific APIs to extract data from a voice over long term evolution (VoLTE) client. The ML pipeline may use interfaces provided by an underlying platform (e.g., EdgeX services) as a source of data or sink of configurations. In that sense, these platform specific APIs may act as realizations of an interface to the src and sink. Traditional 3GPP interfaces like Ng and Xn may need extensions so that they can realize the needs of the ML pipeline. Inter domain interfaces between edge and core, and edge and edge, may be abstracted using interface 8, but realized using p2p platform specific interfaces (e.g., EdgeX interfaces).
Req. applicant	Realization, extension (e.g., to 3GPP, MEC, EdgeX)
Significance/description	There may be cases where a tight coupling at integration stage between the ML pipeline src and sink, and the NF may not be avoidable; e.g., consider the case where the src runs in the RAN but needs measurements from the UE. In this case, the RAN needs to configure the UE for this measurement using RRC. In certain cases, an extension of such interfaces may be needed to achieve the ML function in the use case.
Traceability/example	Figure 3 describes how the Ng interface needs to be extended to support the UE level- or flow level-related information interaction between the RAN (RDA/CUDA) and CN. See also Figures 5 and 9 Figures 6, 7 and 8 describes an EdgeX-based use case where an edge/core interface is needed for deep-learning done in the cloud and ML-based prediction in the edge cloud.

ML-unify-010	It is required that a standard method exist to translate the use-case specifications (e.g., Intention) into an analytic ML pipeline (defined as Intent in rest of the document) Use of a machine-readable format is recommended to instruct use-case specifications to MLFO in order to instantiate the ML pipeline.
Req. applicant	Design-time: Intent, ML-ML

Significance/description	Automation using intent specification and corresponding translation into configurations is a characteristic of future networks. Extending this technique to ML, intent specification of ML use case and correspondingly translate that into pipeline configurations should be supported. Note – Intent specification of the ML use case allows overlaying of ML on top of an existing declarative specification of network services, e.g., those defined in [5].
Traceability/example	Interpretation of the intent specification is a kind of solution for intelligent configuration. The interpretation function can translate the intent into the configuration that can be implemented by network devices.

ML-unify-011	Intention is required to specify the sources of data, repositories of models, targets/sinks for policy output from models, constraints on resources and use case.
Req. applicant	Design-time: Intent. ML-ML
Significance/description	The separation between technology agnostic part of the use case and technology-specific deployment (e.g., 3GPP) is captured in the design time of future network services. Intent specification for the ML use cases achieves this separation for the ML overlay. See clauses 3.2.5 and 3.2.6 for definitions.
Traceability/example	Figure 14 shows intent as a template that captures the requirement of the operator for a ML-related use case. The intention interpretation function may translate it into the configuration that can be implemented by network devices.

ML-unify-012	Any split of the ML pipeline is required to be flexible based on the use cases and constraints defined in the intent.
Req. applicant	Design-time: Intent, ML-ML
Significance/description	Platform capabilities can change (hardware can be added or removed), network capabilities can change (capacity can increase or decrease), NF's may be scheduled or (re)configured dynamically by the NFVO. These dynamic changes may necessitate a change in the split and placement of the ML pipeline (e.g., a decision may be taken to colocate the source and collector, based on changes in the link capacity, or a decision may be taken to instantiate a new source based on scale out of a VNF).
Traceability/example	See Figure 5

ML-unify-013	(NOP external) third party service providers are required to be able to describe the requirements and capabilities for an ML pipeline using intent.
Req. applicant	Design-time: Intent, ML-ML
Significance/description	Third party service providers may offer innovative services on top of future networks. For ML, it means new algorithms. A collaboration between third party providers and operators may bring new sources of data or aggregation mechanisms (e.g., a new smartphone application that interfaces with sensors on the UE). Intention as a declarative mechanism should extend the capabilities to include such third parties, and they should be able to include these nodes in the specification so that end users can enjoy such innovative services offered by third party service providers. Example of such a use case: a third party (e.g., Skype) wants to optimize call quality over the network by running an ML application that configures network parameters. The third party can set up this ML use-case using an interface to the ML pipeline.

Traceability/example	See Figure 5. In some use cases, deep learning may be provided as an MEC application.

ML-unify-014	Time constraints of use cases are required to be captured in the intent.
Req. applicant	Design-time: Intent
Significance/description	Different ML use cases have varied time constraints. At the tightest scale, RAN use cases like beamforming, scheduling, link adaptation would have 50 µs–100 µs latency criteria. These are followed by transport and 5GC use cases that need from 10 ms to a few seconds latency criteria. The least demanding in terms of latency are management level use cases, e.g., anomaly detection, coverage hole detection, that can afford minutes, hours or days of latency. These criteria form an important input to the MLFO while determining the placement, chaining and monitoring of an ML pipeline.
Traceability/example	See Figure 10.

ML-unify-015	Placing and split of ML pipeline nodes is required to consider various constraints (e.g., resource constraints of the NF, latency constraints specific to the use case) The model is required to have the capacity to be placed in a flexible manner in an NF that is most optimal for the performance of the use case (e.g., as per [13]) and constraints defined in the intention. The split of the ML pipeline is required to be flexible based on the use cases and constraints defined in the intention. Placement and hosting of the ML pipeline are required to be flexible based on constraints mentioned in the intent; e.g., placement could be in the core, RAN, or management and orchestration (MANO). It is recommended that the constraints for online training and prediction for real-time applications (e.g., 1 ms~10 ms) are captured in the intent. This may be input to placing these nodes in NFs that can provide optimal performance for the use case (e.g., as per [13]).
Req. applicant	MLFO, intent, constraints
Significance/description	The positioning and placement of ML pipeline nodes on to VNFs forms a major part of the realization of the ML use case with a specific technology (e.g., 3GPP). Thus, it forms the link between two domains: technology-agnostic ML pipeline (overlay) and tech specific network underlay (e.g., 3GPP). The needs, constraints and status of each domain need to be taken into consideration while making this mapping or linkage. Thus, these requirements form an important part of MLFO that achieves this mapping of overlay to underlay and provides a smooth migration path to the operator.
Traceability/example	See Figures 3, 5 and 9. In certain use cases, user plane data classification may be done using DNNs. Since this is a latency-sensitive application, the model may be hosted at the P-GW, whereas the training could be done at MEC.

ML-unify-016	Model selection is required to be done at the setup time of the ML pipeline, using data from the src.
Req. applicant	MLFO: setup

Significance/description	Advances in ML algorithmics suggest that in future networks there would be models with varied characteristics (e.g., a using variety of optimization techniques and weights) that are appropriate for different problem spaces and data characteristics. A ZSM [11] requirement brings in discovery and onboarding of sources of data dynamically. To extend the ML use case to such devices and sources of data, model selection has to be done dynamically, based on the data provided by the source.
Traceability/example	Figure 5 shows the requirement for model selection based on the requirement of the operator specified in the intent.

ML-unify-017	Model training is required to be done in the sandbox using training data. A sandbox domain is recommended to optimize the ML pipeline. Simulator functions hosted in the sandbox domain may be used to derive data for optimizations.
Req. applicant	Non-functional (sandbox)
Significance/description	Model training is a complicated function, it has several considerations: use of specific hardware for speed, availability of data (e.g., data lakes), parameter optimizations, avoiding bias, distribution of training (e.g., multi-agent reinforcement learning), the choice of loss function for training. The training approach used exploration of hyper parameters, for example. Moreover, in future networks, operators will want to avoid service disruptions while model training and updates are performed. These considerations point to the use of a simulator for producing the data for training the models, as well as its use in a sandbox domain.
Traceability	See Figure 5

ML-unify-018	The capabilities to enable a closed loop monitoring and update, based on the effects of the ML policies on the network, are required.
Req. applicant	Non-functional (closed loop)
Significance/description	Closed loop is needed to monitor the effect of ML on network operations. Various KPIs are measured constantly and the impact of the ML algorithm on them as well as on the ML pipeline itself (due to operations of the MLFO) are monitored and corrected constantly. These form inputs to the simulator that generate data. These data can cover new or modified scenarios accordingly in future (e.g., a new type of anomaly is detected in the network, the simulator is modified to include such data. which can also train the model to detect that data type).
Traceability	See Figure 5. In addition, Figure 12 describes the case in which continuous improvement of the automated fault recovery process workflows is important. Hence, not only is the root cause analysis (RCA) provided to the autonomous functions for configuring the NFVO, but also the effect produced by ML in autonomous functions are evaluated and used in a closed loop to optimize the autonomous function itself.

ML-unify-019	A logical orchestrator (MLFO: ML function orchestrator) is required to be used for monitoring and managing the ML pipeline nodes in the system. MLFO monitors the model performance, and model reselection is recommended when the performance falls below a predefined threshold.

Req. applicant	MLFO (monitoring)
Significance/description	The varied levels and sources of data (core, edge), including the simulator and the sandbox domain, imply that there could be various training techniques including distributed training. Complex models that are chained (or derived) may in fact be trained using varied data. The performance of such models can be determined and compared in the sandbox domain using a simulator. Based on such comparisons, operators can then select the model (based on internal policies) for specific use cases. This can be used in conjunction with the MLFO to reselect the model. Note: evaluation may involve network performance evaluation along with model performance.
Traceability	See Figure 5

ML-unify-020	Flexible chaining of ML functions is required to be done based on the hosting and positioning on different NFs and domains. This is to realize the hybrid or distributed ML functions (see the traceability row for details). ML pipeline deployment may be split and multi-level. Chaining of ML pipeline nodes across these levels may be needed to achieve the use case (e.g., the split of ML functions based on gNodeB-CU/gNodeB-DU architecture, see Figure 9). Chaining of logical functions may be used to build a complex analytic ML pipeline.
Req. applicant	MLFO (chaining)
Significance/description	The network function virtualization (NFV) architecture, along with SBA and the emergence of service orchestration mechanisms like the open networking automation platform (ONAP), will enable operators, in the near future, to rapidly design, develop and deploy network services. An MLFO needs mechanisms including flexible chaining to keep up with innovation in the underlay space. As underlying network services evolve and deploy rapidly, so does the ML pipeline on top of them, using these MLFO techniques. This requirement aims to give the ML pipeline overlay, the ability to adapt to dynamic service creation and orchestration.
Traceability	See Figures 5, 10 and 11

ML-unify-021	Support for plugging in and out new data sources or sinks to a running ML pipeline is a requirement
Req. applicant	MLFO (unstructured data)
Significance/description	Certain advanced network services to be defined in future networks, e.g., massive Internet of things (mIoT), require handling of unstructured data from a huge number of sources that may be under ZSM [11]. One such use case is the analysis of logged data for anomaly detection in networks. An MLFO needs mechanisms to perform operations like selecting models based on metadata derived from unstructured data and scaling ML pipeline nodes based on incoming data.
Traceability	See Figures 6, 7 and 8

ML-unify-022	A sharing mechanism for data and inputs between various nodes in the pipeline is recommended to be in the form of distributed, shared, highly performant data storage.
Req. applicant	General: core: database (DB)

Significance/description	Cross-layer, cross-domain sharing of data across various levels, domains and clouds is needed to take correlated ML decisions in future networks. Concepts like data lakes are emerging in future clouds and can also be exploited in operator clouds. Governance mechanisms for data are mandated by regulations in certain areas.
Traceability	See Figures 7 and 8.

8 Unified architecture

Unified architecture stands for a common high-level logical architecture for ML in future networks. However, to understand the deployment options in various technology domains, this architecture has to be considered along with its technology-specific realizations. These are described in clauses 8.1 and 8.2.

8.1 Unified logical architecture

The unified logical architecture is derived from the high-level requirements specified in Table 1. Reuse of existing standards wherever possible is a guiding principle applied while arriving at this architecture. This exercise allows us to study the gaps of existing standards. The level of abstraction used while deriving this logical architecture is such that, while all the basic requirements can be captured using these building blocks, extensions and technology-specific customizations are possible in each standard domain (e.g., 3GPP).

The three main building blocks of the unified logical architecture (Figure 2) are:

- Management subsystem: This includes orchestration, various existing management entities (e.g., virtual network function manager (VNFM), element management system (EMS)), management of platform [e.g., virtualization infrastructure manager (VIM)]. In addition, a new logical entity MLFO is also defined (see clause 3.2.3). Monitoring and management of these functions is achieved using a service-based architecture (SBA) defined in [1]. The intent (see clause 3.2.6) allows the operator to specify and deploy ML services on top of existing ones without tight coupling with the underlying technology used for realization.
- Multi-level ML pipeline: The ML pipeline (see clause 3.2.1) is a logical pipeline that can be overlaid on existing NFs (e.g., VNFs as defined by ETSI or a cloud-native network function (CNF) as defined by CNCF [8]). It uses the services of an MLFO for instantiation and setup. For lifecycle management, it uses the services of the NFVO. The deployment of an ML pipeline may span different levels and domains or clouds. The MLFO coordinates this deployment. In this context, interface-8 (see clause 3.2.2) is important to achieve the chaining of such a multi-level deployment. Specific integration aspects of such an overlay of an ML pipeline on a specific technology (e.g., how to integrate an ML pipeline across various NFs in 3GPP CN and RAN) may require extension of existing interfaces or definition of specific APIs.

NOTE 1 – Figure 2 shows three domains: CN, transport and RAN. These are treated as administrative domains. These may be owned, operated and administered by different entities (e.g., in roaming cases). These should be treated as examples while other domains are possible in the network.

NOTE 2 – Figure 2 shows an instance of the MLP (ML pipeline) overlay on these domains. However, based on specific use cases, other ways of distributing the MLP nodes are possible. These are shown in clause 8.2.

- Closed-loop subsystem: future wireless networks present a dynamic environment. Various conditions can change in the network (e.g., air interface conditions, UE position, platform capabilities and platform capacity). A closed loop subsystem allows the ML pipeline to adapt to this dynamic environment. It is driven by a simulator and monitored by the MLFO using the parameters defined in the intent. Such a "sandbox" environment allows operators to study the effect of ML optimizations before deploying them on a live system. As mentioned in

clause 3.2.6, updates from the network are fed back into the closed loop so that the ML pipeline can adapt to dynamically changing environment in the network.

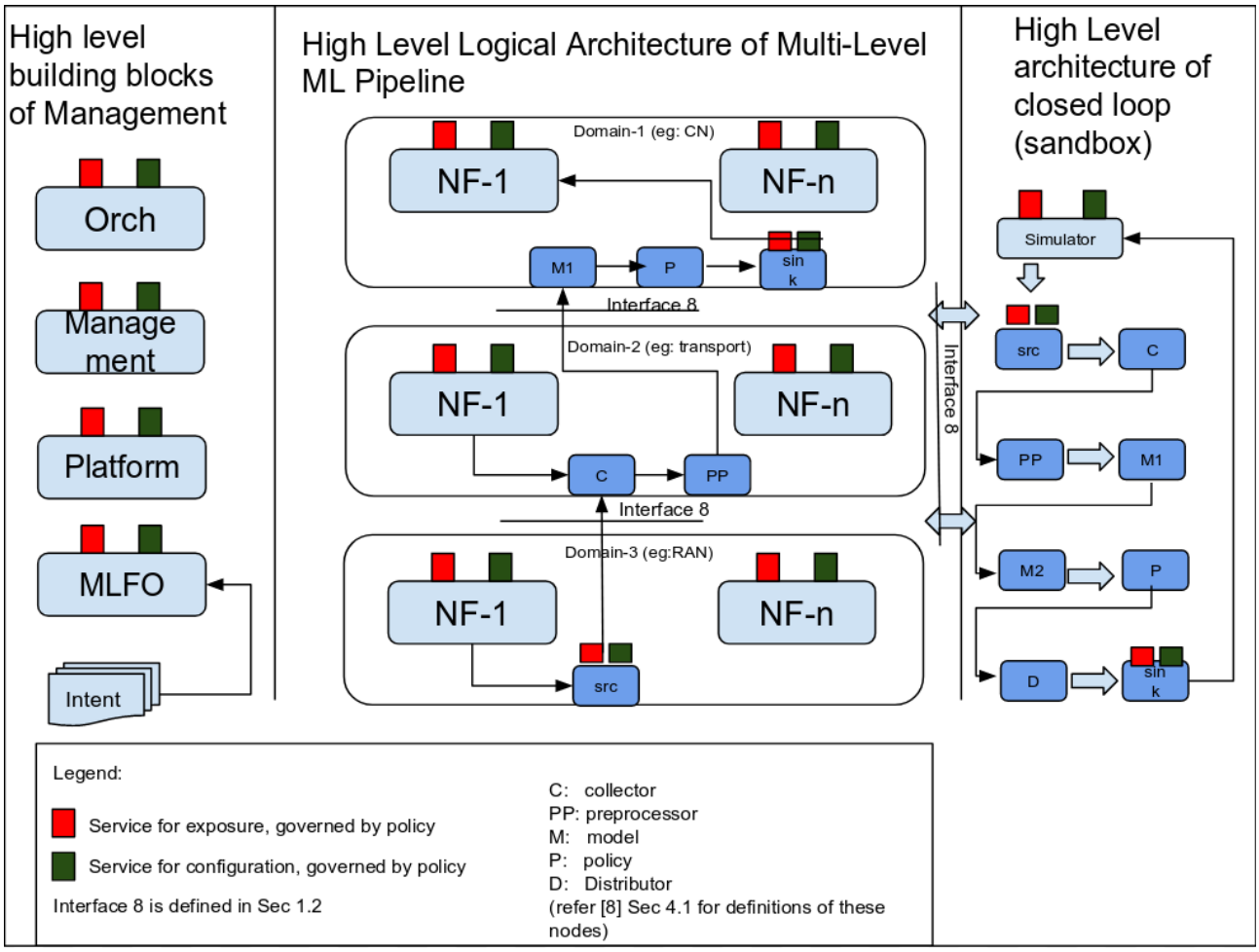

Figure 2 – Unified logical architecture

8.2 Realization(s) of the logical architecture

Figure 3 gives an instance of realization of the logical architecture on a 3GPP system along with MEC and management systems. The realization is achieved in the following manner.

- Use the ML pipeline (see clause 3.2.1) to show the positions in this realization wherever the nodes in the ML pipeline can be hosted. e.g., CN, RAN, MEC, NMS.
  - Consider arrows: 1→2→4→ML pipeline1: This pipeline uses inputs from the UE to make predictions at CN (e.g., MPP-based use cases).
  - Consider arrows: 9→2→4→ML pipeline1: This pipeline uses inputs from the RAN and possibly a combination of UE and RAN, to make predictions at CN (e.g., MPP-based use cases).
  - Consider arrows: 10→7→ML pipeline2→8: This pipeline uses inputs from the MEC platform to make predictions at the edge and to apply them to the MEC. It can also use side information from the UE and RAN (e.g., caching decisions made at the MEC, local routing decisions at the MEC).
  - Consider arrows: 3→4→ML pipeline1→5: This pipeline uses inputs from CN and possibly a combination of UE and RAN inputs to make predictions at CN and applies it to NMS parameters that can in turn affect configurations in different domains (e.g., SON decisions made at the CN).

  - Consider arrows: ML pipeline3→6: local predictions at the NMS that can in turn affect configurations in different domains (e.g., parameter optimizations based on data analytics).
- Call out extensions in 3GPP interfaces or MEC interfaces where applicable.
  - Consider arrow 1 in Figure 3: this can be realized as an extension of RRC.
  - Consider arrow 2 in Figure 3: this can be realized as an extension of N2 interface [1]
  - Consider arrow 5 in Figure 3: can be realized via a reuse of ETSI SOL002 [10]
  - Consider arrow 10 in Figure 3: this can be realized via a reuse or extension of GS MEC 011 [7]
- Give instances of constraints applicable for placement of the ML pipeline in 3GPP.
  - UE is a resource-constrained device, hence only a source is instantiated in the UE.
  - As mentioned in ML-unify-014, RAN and MEC might have latency constraints on their use cases. Hence those models are hosted in the RAN itself as ML pipeline 2. Those data from the RAN and UE that are not used in such latency-bounded use cases are sent to the CN via arrow 2 in Figure 3.

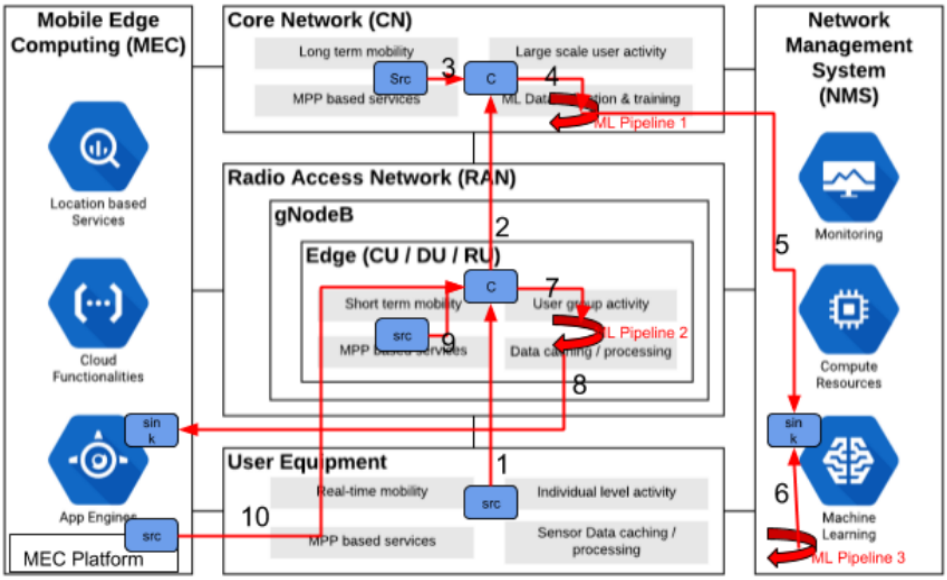

Figure 3 – Hosting of multi-level ML pipeline in 3GPP, MEC

Figure 4 (see also [1]) gives the interface points for the logical ML pipeline with various technology services. The interface points are achieved in the following manner.

- The ML pipeline is loosely coupled with 3GPP and other technologies. The ML pipeline has clear interface points at whih it interacts with 3GPP Network services. This allows the ML pipeline to evolve separately from underlying technologies, while allowing all forms of 3GPP and non-3GPP networks, even simulated ones, to benefit from ML services.
- In Figure 4, MnSx stands for producer of analytics services (and consumer of data) whereas MnSx' stands for consumer of analytics service (and producer of data).

NOTE – Data may be shared as described in ML-unify-022.

- This also provides an interface point for third party service providers who may provide innovative services on top of future networks. These may be ML based, e.g., new ML algorithms or optimization mechanisms, or an over the top (OTT) service (e.g., VoLTE). A plug-in mechanism for these third party providers is needed to handle the ML needs of such services. Intention as a declarative mechanism should extend the capabilities to include such

third parties and they should be able to include these nodes in the specification, so that end users can enjoy such innovative services. Please see ML-unify-13 for an example of such a service.

- The NOP may introduce services (e.g., SON based on analysis of data from the network) which takes advantage of the ML pipeline. These too, interface with the ML pipeline instances via MnS interface points.

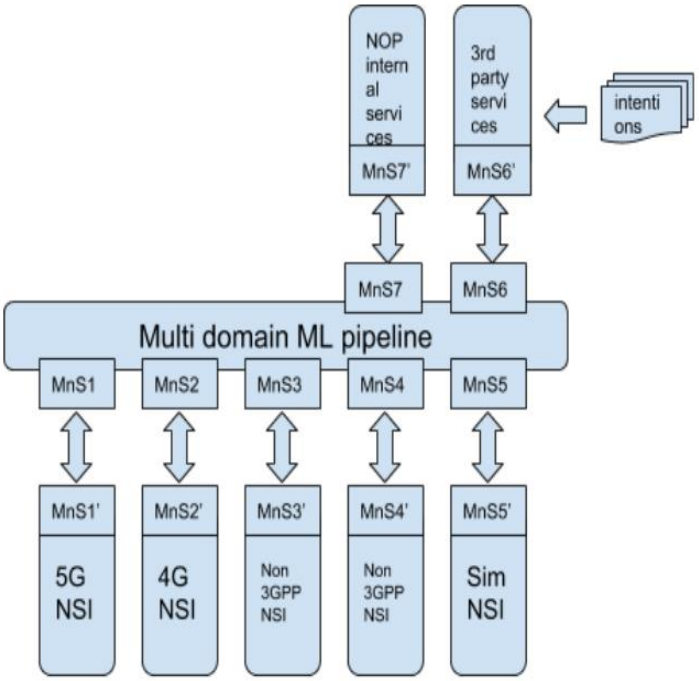

Figure 4 – MnS-based interface points for ML pipeline

Figure 5 gives another instance of a realization of the ML pipeline in a 3GPP network. Here the focus is on the following.

- Standard mechanisms for configuration of the ML pipeline using MLFO. MLFO in turn may use intent as an input.
- The MLFO interface (see arrow 1 in Figure 5) includes monitoring and management of the ML pipeline using MLFO (including model selection and reselection, see also ML-unify-019).
- Distributed and multi-level placement of src, sink and ML pipeline in general, using MLFO.
- Slice creation based on user data, integration with application logic are examples of use cases that can be achieved by placement of src and sink as shown in Figure 5. Such placement of src and sink may require specific interfaces that integrate the data collection (see arrows 2, 3, 5) and network configuration (see arrow 4), respectively.

Figure 5 – Management of multi-level ML pipeline

In Figure 6, the realization of the ML pipeline in the EdgeX [3], which is a vendor-neutral open source software platform, is demonstrated. This enables interaction with devices, sensors, actuators, and other IoT objects. It provides a common framework for Industrial IoT edge computing. The supporting service (SS) layer of the EdgeX layer provides edge analytics and intelligence.

Realization of the ML pipeline is demonstrated as follows.

- ML pipelines are instantiated in the core cloud and edge clouds. For example, in Figure 6, three ML pipeline instances are shown. They coordinate using interface 8 (see clause 3.2.2).
- Arrow 4 shows a local analytics service (ML pipeline 2) based on platform inputs.
- In the core cloud, an ML pipeline will have the core NFs as src. This may be used in correlation with data from the edge.
- Arrow 5 shows a local analytics service (ML pipeline 3) based on local inputs at the core cloud and forwarded inputs from edge cloud 1 via arrow 2.
- In the edge cloud, an ML pipeline will have the EdgeX platform service and its other services as src. These may be used in correlation with data from another edge (e.g., in the 3GPP V2X use case or mobility case mentioned in [12]).
- Arrow 1 shows a "store and forward" of data collected at edge cloud 2 to edge cloud 1. These data are then analysed by ML pipeline 1.
- Collaborative interfaces Figure 6 will be realized using interface 8 (see clause 3.2.2). This helps in enriching the data available at any pipeline with side info from other instances.
- Arrow 3 shows a local analytics service (ML pipeline 1) based on local inputs and forwarded inputs from edge cloud 2.

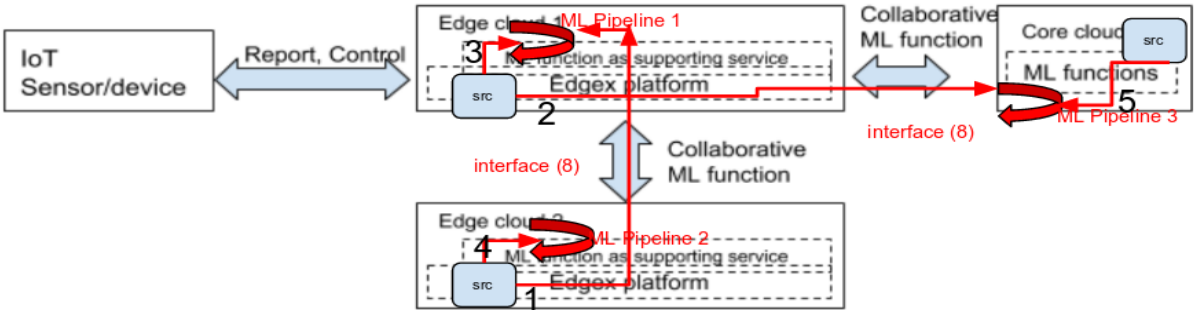

Figure 6 – EdgeX-based instance of ML pipeline

Figures 7 and 8 shows specific instances of Figure 6.

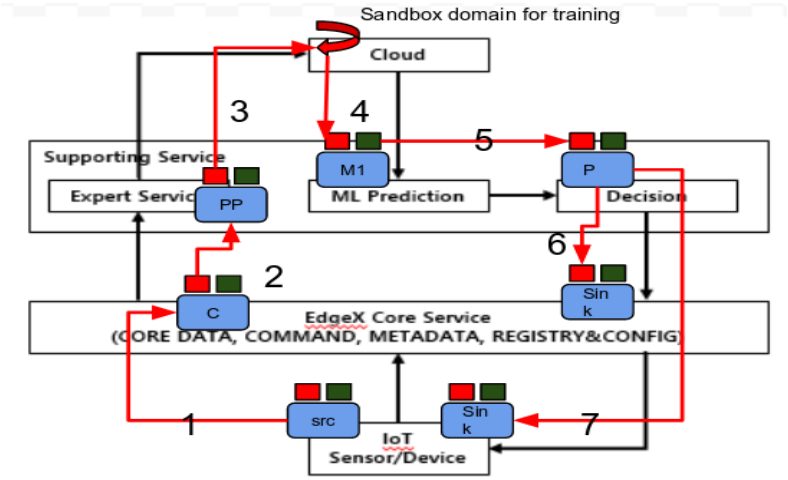

Figure 7 – ML-based edge computing workflow for prediction

Figure 7 shows Edge-platform-based ML prediction. It may include the following.

- Intelligent network traffic control technology edge computing technology based on EdgeX (Open source edge platform).
- Intelligent traffic analysis and ML-based prediction (e.g., deep learning, reinforcement learning).
- Hybrid intelligent network control approach combined by optimizing prediction and control.
- Figure 7 shows: (arrow 1) collecting data from a sensor/device; (arrow 2) stored data in EdgeX core service; (arrow 3) data pre-processing by EdgeX expert service; (arrow 4) transforming data-set to cloud for ML training in sandbox domain; (arrow 5) model serving with the trained model from ML prediction in EdgeX; (arrows 6,7) ML decision in EdgeX platform.

Figure 8 shows a real-time monitoring and control service based on edge computing. It may include the following.

- Intelligent edge-based monitoring and control system for analysing and processing real-time sensor optimization.
- ML-based intelligent IoE service and optimal control system using edge computing.
- Intelligent real-time edge computing solution.
- Figure 8 shows: (1) real-time sensor/device data collection by EdgeX device service via broker, followed by data collection and processing by an ML-based module; (2) data collection and saving in a DB (e.g., MongoDB) via the EdgeX core service; (3) real-time monitoring of the stored sensor/device data by the EdgeX-based monitoring client via the EdgeX expert service; (4) optimizing real-time control on the IoT sensor/device via the EdgeX core service.

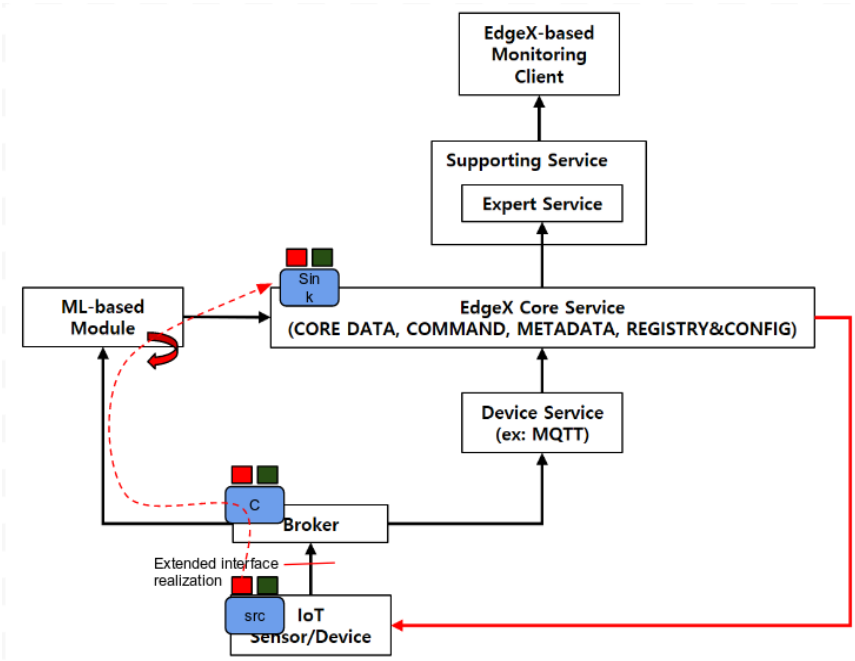

Figure 8 – Real-time monitoring and control system using EdgeX

Figure 9 shows an instance with the following characteristics.

- In an SBA, the ML pipeline is hosted in the network data analytics function (NWDAF) [1].
  - ML pipeline 1 may have AMF as src (arrow 4) and PCF as sink (arrow 5) to realize a particular use case (e.g., mobility-based policy decisions).
- A resource-constrained DU hosts part of the ML pipeline, but not the training. The training is done at the CU and the trained model is distributed to the DU, where it is hosted.
  - DU hosts M2 which is updated from the CU via arrow 3.
  - Data for training the model in the CU is provided via arrow 1.
- Collapsing of the interface between ML pipelines is an option as shown in the RAN data analytics (RDA) option. This brings out the need for flexibility in deployment. See ML-unify-012, ML-unify-015 and ML-unify-020.
  - M1 and M2 are hosted in CUDA and DUDA (in ML pipeline 2 and 3, respectively) in the 3GPP split deployment, whereas they are collapsed (merged) in the other 3GPP alternative deployment options [1].
- The extension of 3GPP interfaces for carrying information specific to ML pipeline execution and training is a requirement here.
  - For example: RDA is primarily used to support optimization in the RAN. It also needs to provide data subscription services for NWDAF and business and operation support system (BOSS), operation support system (OSS) or MANO; and
  - upload pre-processed subscription data to NWDAF and BOSS or OSS (arrow 8) for further big data operations and services;
  - RDA can also subscribe to the NWDAF data analysis results for the RAN-side service optimization (arrow 6).

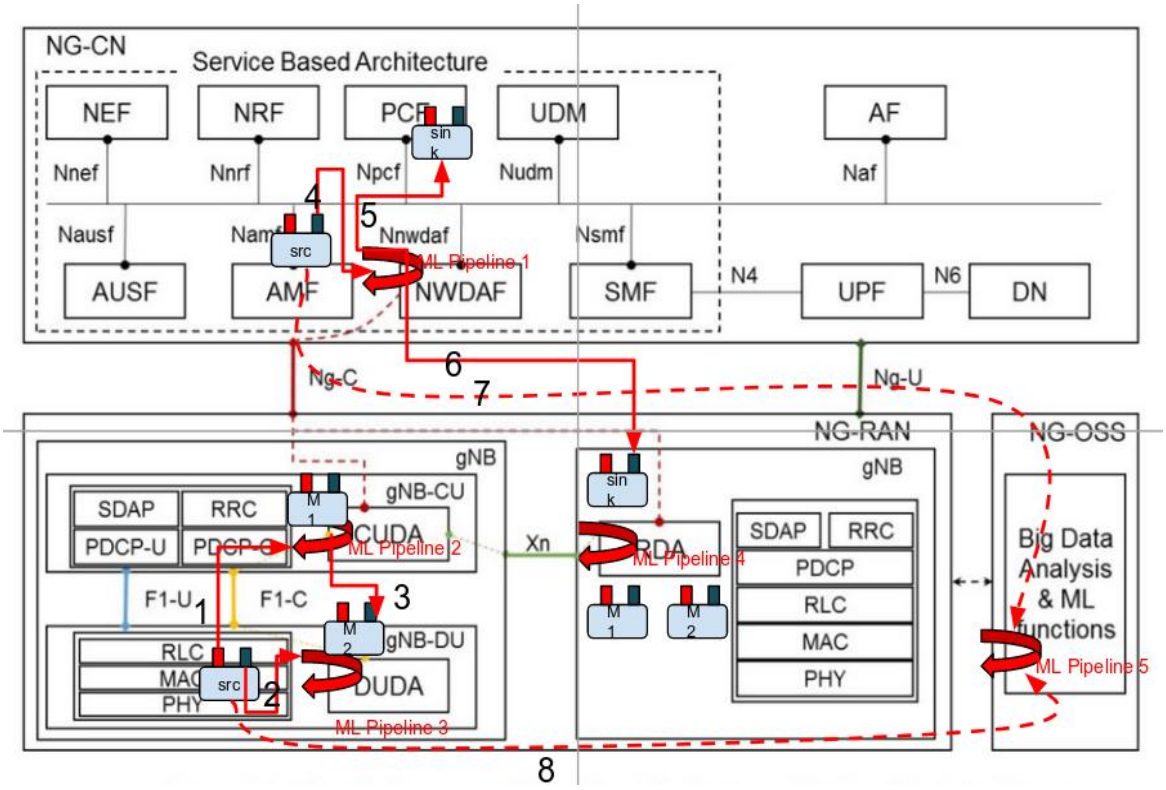

Figure 9 – Hierarchical distributed instance

In Figure 10, the mechanism of incorporating timing constraints of various 3GPP use cases into their realization using ML pipeline is shown. The timing constraints are captured in intents, which are in turn processed by MLFO to determine instantiation choices, like positioning of various ML pipeline nodes. In Figure 11, RAN use cases have the strictest latency constraints (50 µs–10 ms). Therefore, the MLFO may choose to position the entire ML pipeline 2 in the RAN. In contrast, use cases related to 5GC have 10 ms to a few seconds latency budgets. Hence, the MLFO may choose to enrich the data in ML pipeline 1 with side information from the RAN. The same is applicable to ML pipeline 3.

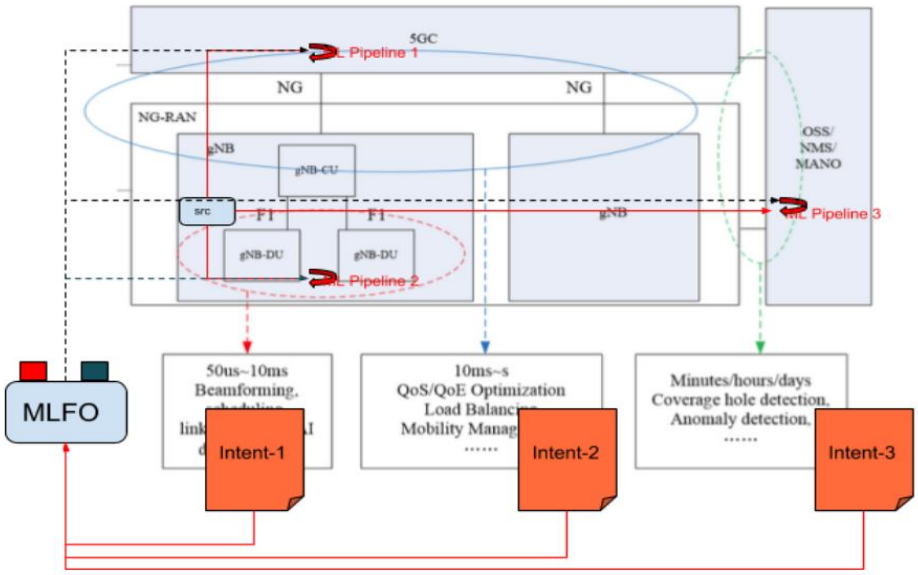

Figure 10 – Timing constraints and intents

Figure 11 shows a unique realization in which NWDAF functions are hierarchical across three domains: CN, AMF and RAN. This split allows certain specific data to be used for local decisions at these NFs. From an ML pipeline perspective, this would mean that the pipelines are chained, so that the output of one could feed into the input of another.

- Arrow 1, arrow 2, show control by the NS manager using the ML pipeline in NWDAF. This enables use cases like dynamic slice configuration using ML.
- Arrows 3,4, and 5,6 are local NWDAF functions in AMF and RAN respectively (e.g., AMF can customize connection management, registration management, mobility restriction management for UEs based on the long-term UE MPP).
- Arrow 7 shows chaining, so that the output of a remote NWDAF can feed into the local NWDAF as input (e.g., while performing short term MPP).

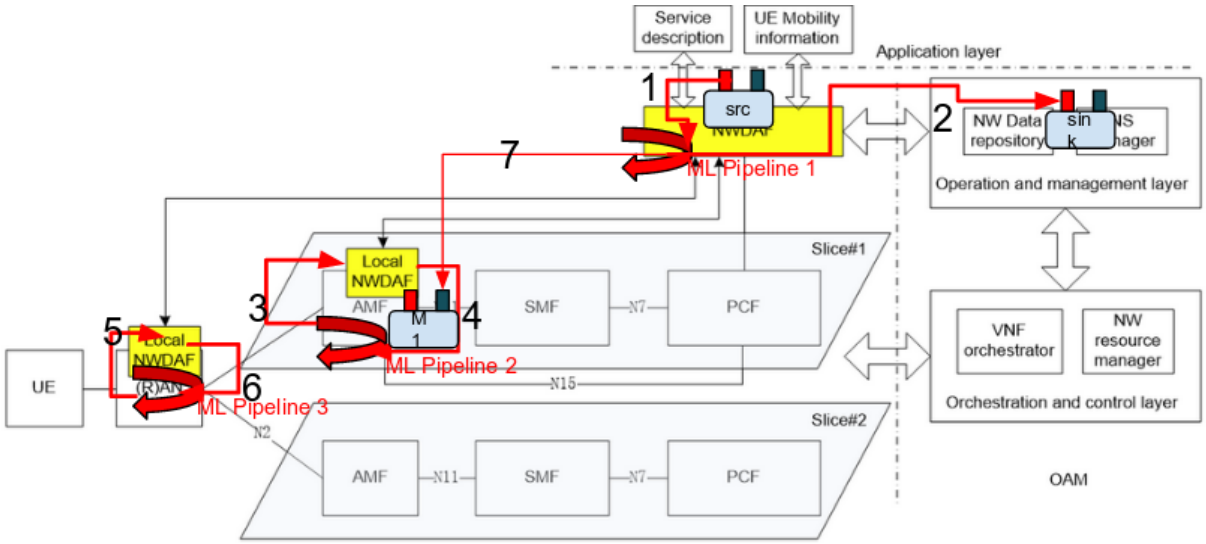

Figure 11 – Hierarchical instances

Figure 12 shows the proposed architecture to achieve closed loop automation in operation and management on 5G networks.

- The management system should be automated to the extent possible to promptly react to failures in the NFV. The operator wants to promptly discover such failures, which result in increasingly unstable behaviour before the process escalates into critical failure. RCA is also important to properly convey relationship information between failure type and location to automation function.
  - Line 1 in Figure 12 shows the src/collector interface. Line 2 shows the collector/ML pipeline interface. Line 3 conveys the result of the ML pipeline (RCA or predictive detection) to the policy function in the automation function.
  - NFVO is configured based on policy or workflows (line 4).
- Continuous improvement of the automated fault recovery process workflows is important.
  - Line 6 provides the output corresponding to this improvement to the sink hosted in the automation function.
- As mentioned in ML-unify-019, the ML pipeline is configured and monitored by MLFO via Line 5.

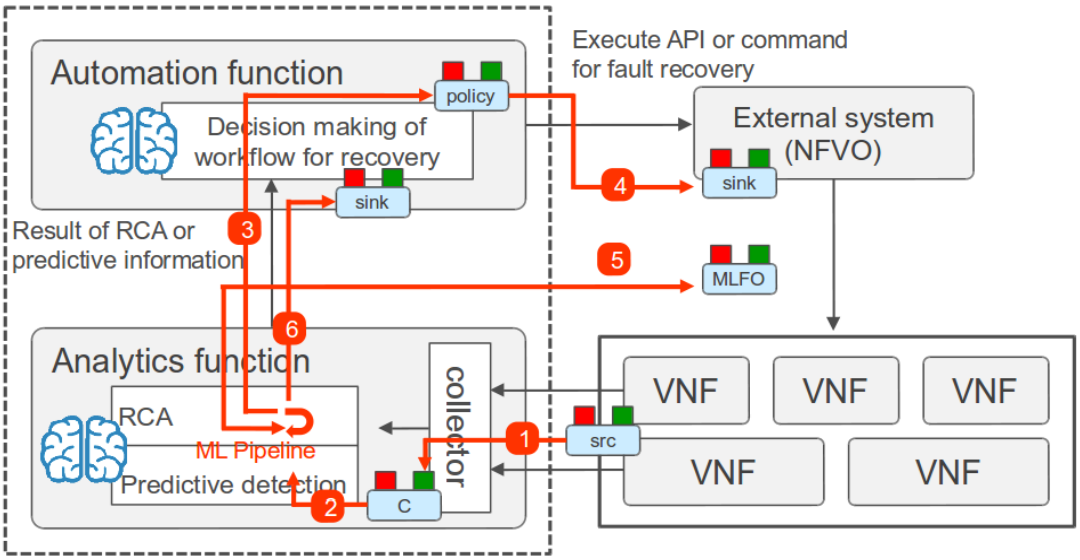

Figure 12 – ML in closed loop automation

Figure 13 shows the ML model being executed both on the cloud and the edge servers. In general, the knowledge acquisition (training) part is executed on the cloud server, as there is no severe timing requirement, but sufficient processing power is required. However, the knowledge application part, which requires severe timing conditions, is expected to be executed on the edge using the acquired knowledge, which is an output of ML executed on the cloud server. There may be transfer of learning or output between the models in the two domains.

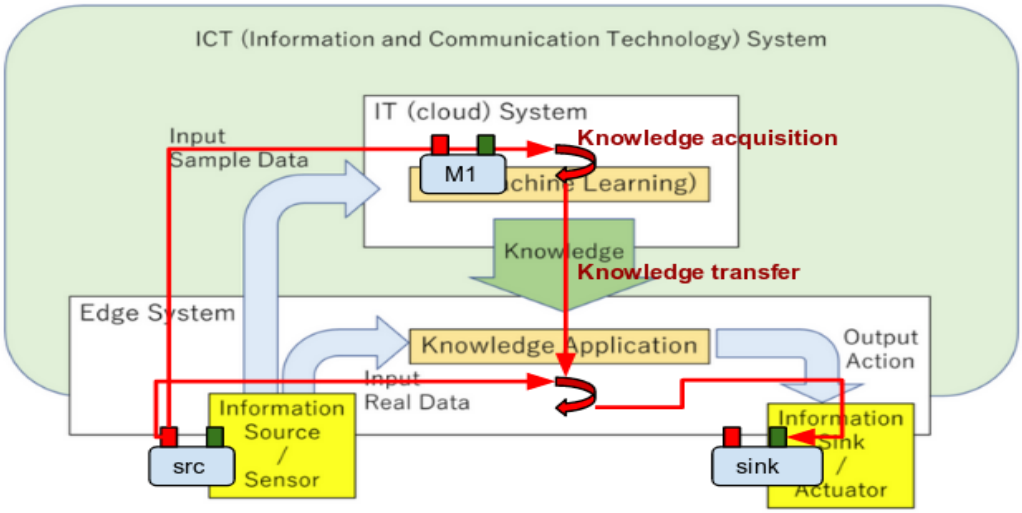

Figure 13 – Transfer of learning between domains

9 Informative Appendix: Key architectural issues for further study

9.1 Standardize a machine learning meta language

Description/significance:

- An ML-ML should be specified as a technology-agnostic, declarative specification language, that can be used to specify how an ML pipeline can be "composed" for a use case.

  For example, an NOP could specify using ML-ML that a new ML-based use case needs to be introduced –alarm analysis, RCA and network problem area prediction – by deploying a new src in the NMS subsystem, using a newly available model in a public repository, and directing the output to a fault recovery module via a policy function. Such a specification (intent) could be agnostic of its underlying implementation in specific technology (e.g., 3GPP) by various vendors.
- ML-ML is used to make declarative specification, from which flows the interpretation and realization of such a ML pipeline in, say 4G, 5G or any future networks, including simulated networks in a standard, predictable, interoperable manner, with no surprises.
- Service orchestration (SO) mechanisms, implemented by different vendors, need to understand the requirements specified in this format and control the steps in SO according to this input.
- It is important to differentiate the language (ML-ML) from the specifications (intent) written in that language.
- Prepare the mapping between ITU requirements and 3GPP, and other realizations of the use case (because of clear demarcation of use case specific nodes and ML pipeline nodes in generic architecture).

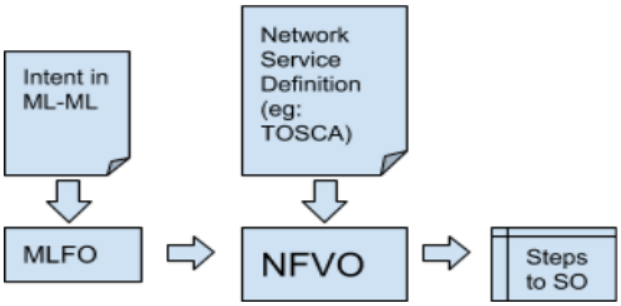

Figure 14 – The role of Intent and ML-ML

NOTE – A possible realization of ML-ML might use basic constructs in existing meta-languages and MLFO may be realized as a function as part of NFVO. Such implementation details are beyond the scope of the current study.

Current work and identified gaps:

- ETSI defines service orchestration. Network service catalogues for end-to-end (E2E) service description with VNF and physical network function (PNF) descriptors and service graphs are specified.
- Container-based orchestration platforms (e.g., Kubernetes) have their own mechanisms for service orchestration.
- Efforts are on to integrate these two, in ONAP.
- Current efforts are focused on providing E2E service. The impact of introducing an ML pipeline on to such service orchestration mechanisms should be studied. Reuse of existing mechanisms should be maximized, but at the same time, gaps in integrating ML in a descriptive fashion while orchestrating VNF, PNF and CNF should be studied.

9.2 Study the level of capability exposure needed to enable dynamic ML-based use cases in 5G and future networks

Description/significance:

- Capability exposure and creation of ML-based services in future networks are closely related to each other.
- ML use cases depend heavily on availability of data to analyse.
- Measurement data and context data related to the use case needs to be identified.
- Some of these data are standardized while others are not.
- Introduction of SBA into 3GPP implies that a means of obtaining these data is via services from MnS producers which expose such data. These data can be consumed by MnS consumers which use them for analytics.
- This may in turn be governed by an exposure governance management function (EGMF) as defined in 3GPP.
- Dynamic and rapid service creation can be achieved in future networks only by mapping the capability exposure with service creation, discovery and chaining.

Current work and identified gaps:

- 3GPP specifies [1] southbound interfaces between the NEF and 5GC NFs e.g., N29 interface between NEF and SMF or N30 interface between NEF and PCF.
- 3GPP NFs expose capabilities and events to other NFs via NEF.
- e.g., The AMF provides the UE mobility-related event reporting to NF that has been authorized to subscribe to the UE mobility event reporting service via NEF.

- However, these are neither dynamic, nor granular. Furthermore, cloud-native events [8] need to be supported along with discovery and chaining.
- In combination with dynamic creation and deployment of an MnS dynamically (third party or NOP internal), discovery, chaining and integration with NEF also have to happen dynamically. Requirements and mechanisms for these need further study.

9.3 Study the requirements for "division of ML labour" between clouds. Study the implementation of multi-level ML interface and its relation with the NFVO-NFVO interface

Description/significance:

- Multi-level interfaces are used in almost all use cases.
- Different orchestration abilities may be present in different clouds, e.g., edge clouds may not have all orchestrator functions.
- When designing, developing, testing, deploying and managing ML workloads across such multiple clouds, interface between them will be subject to specific needs, e.g., design time specification and runtime deployment of ML pipeline nodes across domains, monitoring ML pipeline nodes, decision of where analytics are done (based on the capability of the cloud, e.g., a resource-constrained edge node may not host training function, but it may host the runtime model predictor).
- Requirements of ML functions on such a cloud/cloud interface have to be studied to enable smooth deployment of ML functions across clouds.

Current work and identified gaps:

- Many open source forums are studying the implementation of generic cloud, e.g., ONAP, Akraino and openstack edge.
- In ETSI-NFV, IFA022 studies connectivity service instantiations between different network function virtualization infrastructure (NFVI) points of presence (PoPs) for network service life cycle management. A network service is instantiated by the interactions among BOSS, NFVO, WIM/VIM and network controllers. The current IFA022 analyses the interactions among BOSS, NFVO, wireless infrastructure network (WIM)/VIM with reference to the current ETSI-NFV standards specifications.
- draft-bernardos-nfvrg-multidomain-05 analyses the problem of multi-provider multi-domain orchestration, by first scoping the problem, then looking into potential architectural approaches, and finally describing the solutions being developed by the European 5GEx and 5G-TRANSFORMER projects.
- None of these addresses the needs of MLFO.

9.4 Study the relationship between ML pipeline nodes and 3GPP NFs. Understand the management of ML pipeline nodes

Description/significance:

- ML pipeline nodes are an overlay on top of 3GPP. They could be hosted on any 3GPP NF by orchestration methods. They are managed by MLFO which in turn is managed by NFVO. MLFO may use interfaces and coordination with 3GPP and non-3GPP to manage the pipeline.
- Further, an ML pipeline acts as a non-3GPP service and interacts with 3GPP NF using 3GPP-defined MnS. Requirements for nodes in an ML pipeline will be defined by ITU (and not by 3GPP).

- ML pipeline exposes only type A components towards 3GPP, because operation or notification has to be produced or consumed towards 3GPP NF MnS. Types B and C, even if exposed, may be emulated (not real 3GPP NF).
- Interaction with legacy 3GPP NFs needs to be studied using wrapper services. These wrapper services expose interfaces towards the ML pipeline using standard interfaces, but implement legacy or vendor-specific interfaces towards the NF. This aspect to provide a smooth migration path to operators needs further study.