International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

Y.3179

(04/2021)

SERIES Y: GLOBAL INFORMATION INFRASTRUCTURE, INTERNET PROTOCOL ASPECTS, NEXT-GENERATION NETWORKS, INTERNET OF THINGS AND SMART CITIES

Future networks

Architectural framework for machine learning model serving in future networks including IMT-2020

Recommendation ITU-T Y.3179

ITU-T Y-SERIES RECOMMENDATIONS

GLOBAL INFORMATION INFRASTRUCTURE, INTERNET PROTOCOL ASPECTS, NEXT-GENERATION NETWORKS, INTERNET OF THINGS AND SMART CITIES

GLOBAL INFORMATION INFRASTRUCTURE	
General	Y.100–Y.199
Services, applications and middleware	Y.200–Y.299
Network aspects	Y.300–Y.399
Interfaces and protocols	Y.400–Y.499
Numbering, addressing and naming	Y.500–Y.599
Operation, administration and maintenance	Y.600–Y.699
Security	Y.700–Y.799
Performances	Y.800–Y.899
INTERNET PROTOCOL ASPECTS	
General	Y.1000–Y.1099
Services and applications	Y.1100–Y.1199
Architecture, access, network capabilities and resource management	Y.1200–Y.1299
Transport	Y.1300–Y.1399
Interworking	Y.1400–Y.1499
Quality of service and network performance	Y.1500–Y.1599
Signalling	Y.1600–Y.1699
Operation, administration and maintenance	Y.1700–Y.1799
Charging	Y.1800–Y.1899
IPTV over NGN	Y.1900–Y.1999
NEXT GENERATION NETWORKS	
Frameworks and functional architecture models	Y.2000–Y.2099
Quality of Service and performance	Y.2100–Y.2199
Service aspects: Service capabilities and service architecture	Y.2200–Y.2249
Service aspects: Interoperability of services and networks in NGN	Y.2250–Y.2299
Enhancements to NGN	Y.2300–Y.2399
Network management	Y.2400–Y.2499
Network control architectures and protocols	Y.2500–Y.2599
Packet-based Networks	Y.2600–Y.2699
Security	Y.2700–Y.2799
Generalized mobility	Y.2800–Y.2899
Carrier grade open environment	Y.2900–Y.2999
FUTURE NETWORKS	**Y.3000–Y.3499**
CLOUD COMPUTING	Y.3500–Y.3599
BIG DATA	Y.3600–Y.3799
QUANTUM KEY DISTRIBUTION NETWORKS	Y.3800–Y.3999
INTERNET OF THINGS AND SMART CITIES AND COMMUNITIES	
General	Y.4000–Y.4049
Definitions and terminologies	Y.4050–Y.4099
Requirements and use cases	Y.4100–Y.4249
Infrastructure, connectivity and networks	Y.4250–Y.4399
Frameworks, architectures and protocols	Y.4400–Y.4549
Services, applications, computation and data processing	Y.4550–Y.4699
Management, control and performance	Y.4700–Y.4799
Identification and security	Y.4800–Y.4899
Evaluation and assessment	Y.4900–Y.4999

For further details, please refer to the list of ITU-T Recommendations.

Recommendation ITU-T Y.3179

Architectural framework for machine learning model serving in future networks including IMT-2020

Summary

Recommendation ITU-T Y.3179 provides an architectural framework for machine learning (ML) model serving in future networks including IMT-2020, i.e., preparing and deploying ML models in different deployment environments to enable the application of ML model inference to ML underlay networks. The framework includes high-level requirements, and a high-level architecture description covering the definition of architectural components and reference points.

History

Edition	Recommendation	Approval	Study Group	Unique ID*
1.0	ITU-T Y.3179	2021-04-29	13	11.1002/1000/14614

Keywords

Architecture, future networks, IMT-2020, inference, machine learning, model optimization, reference points, requirements, serving ML model.

* To access the Recommendation, type the URL http://handle.itu.int/ in the address field of your web browser, followed by the Recommendation's unique ID. For example, http://handle.itu.int/11.1002/1000/11830-en.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications, information and communication technologies (ICTs). The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure, e.g., interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had not received notice of intellectual property, protected by patents/software copyrights, which may be required to implement this Recommendation. However, implementers are cautioned that this may not represent the latest information and are therefore strongly urged to consult the appropriate ITU-T databases available via the ITU-T website at http://www.itu.int/ITU-T/ipr/.

Table of Contents

Recommendation ITU-T Y.3179

Architectural framework for machine learning model serving in future networks including IMT-2020

1 Scope

This Recommendation provides an architectural framework for machine learning (ML) models serving in future networks including IMT-2020, i.e., preparing and deploying ML models in different deployment environments to enable the application of ML model inference to ML underlay networks.

The scope of this Recommendation includes:

– Background and motivations;

– High level requirements;

– High-level architecture description including the definition of architectural components, reference points and sequence diagrams.

2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

[ITU-T Y.3172] Recommendation ITU-T Y.3172 (2019), *Architectural framework for machine learning in 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*.

3 Definitions

3.1 Terms defined elsewhere

This Recommendation uses the following terms defined elsewhere:

3.1.1 machine learning model [ITU-T Y.3172]: Model created by applying machine learning techniques to data to learn from.

NOTE 1 – A machine learning model is used to generate predictions (e.g., regression, classification, clustering) on new (untrained) data.

NOTE 2 – A machine learning model may be encapsulated in a deployable fashion in the form of a software (e.g., virtual machine, container) or hardware component (e.g., IoT device).

NOTE 3 – Machine learning techniques include learning algorithms (e.g., learning the function that maps input data attributes to output data).

3.1.2 machine learning function orchestrator (MLFO) [ITU-T Y.3176]: A logical node with functionalities that manage and orchestrate the nodes in a machine learning pipeline.

3.1.3 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.

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.

NOTE 1 – Examples of parties include suppliers and users of ML models. Capabilities provided to users of ML models include functionalities to find, learn about, deploy (or download) and use ML models. Capabilities provided to suppliers of ML models (e.g., data scientist) include functionalities to share (on-board, upload), describe (learn about) and market their ML models.

NOTE 2 – A network operator may use a machine learning marketplace deployed internally and/or externally to the network operator's administrative domains. Internal and external marketplaces differ only in the deployment perspective. A marketplace which is internal to a network operator may act as an external marketplace to another network operator and vice versa.

3.1.5 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.

NOTE – The nodes of a machine learning pipeline are entities that are managed in a standard manner and can be hosted in a variety of network functions [b-ITU-T Y.3100].

3.1.6 machine learning underlay network [ITU-T Y.3172]: A telecommunication network and its related network functions which interfaces with corresponding machine learning overlays.

NOTE – An IMT-2020 network is an example of a machine learning underlay network.

3.2 Terms defined in this Recommendation

This Recommendation defines the following terms:

3.2.1 model inference: Process by which a deployed machine learning model generates a result.

NOTE – Examples of generated result from machine learning model are prediction or classification.

3.2.2 inference optimization: Optimization performed on a trained machine learning model for better performance during inference.

NOTE – Examples of better performance are improved latency and computing efficiency.

3.2.3 inference engine: Functionality that provides runtime environment for a machine learning model and exposes corresponding machine learning model inference capability.

3.2.4 ML model serving: A process of preparing and deploying machine learning models in different deployment environments to enable the application of model inference to machine learning underlay networks.

3.2.5 serving ML model: A machine learning model, obtained as a result of a machine learning model serving process, which is ready for performing inference in machine learning underlay networks.

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

AN Access Network

API Application Programming Interface

AUC Area Under Curve

CN Core Network

CPU Central Processing Unit

FPGA	Field Programmable Gate Arrays
GPU	Graphics Processing Unit
IoT	Internet of Things
MEC	Mobile Edge Computing
ML	Machine Learning
MLDB	Machine Learning Database
MLFO	Machine Learning Function Orchestrator
MSE	Mean Squared Error
NF	Network Function
QoS	Quality of Service
RAN	Radio Access Network
src	source (ML pipeline)
UPF	User Plane Function
VNF	Virtual Network Function

5 Conventions

In this Recommendation, 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.

ML pipeline – In this Recommendation, in alignment with the conventions of [ITU-T Y.3172] when the symbol shown in Figure 1 is used, this denotes a subset (including proper subset) of nodes in an ML pipeline. When this symbol is used in a figure, the symbol stands for the subset of an ML pipeline's nodes not explicitly shown in that figure.

Figure 1 – Symbol used to denote a subset of nodes in an ML pipeline

6 Introduction

This Recommendation aims at addressing challenges for ML models serving in future networks including IMT-2020, i.e., application of ML model inference to ML underlay networks. The challenges include the following:

– Enabling efficient ML models optimization mechanisms for heterogeneous deployment environments.

The existing ML marketplaces [ITU-T Y.3176], such as the ones available in open-source [b-LF Acumos], provide built-in packaging solutions which wrap the trained models using packaging mechanisms, like containers, without optimization considerations regarding the deployment environment. This may result in lack of efficiency, e.g., due to a high computational complexity and redundant parameters, while performing inference. Furthermore, diverse inputs and constraints from varied ML underlay networks may require specific optimization considerations.

– Ensuring flexible deployment of ML models for different use case scenarios.

Deployment requirements of ML models in network operators' networks may be specific to the use case requirements and deployment environment. The existing containerized deployment environments may not provide enough flexibility and portability to address the considerations of network operators' deployment preferences.

NOTE 1 – For example, a network operator may prefer deployment of ML models from internal ML marketplace implemented using Acumos or external ML marketplaces of third party players. Each marketplace supports specific deployment environments, e.g., Kubernetes cluster [b-Kubernetes]. However, the network operator may need to deploy ML models optimized for less resource consumption or quicker execution in the access network (AN) while ML models optimized for higher throughput may be implemented in the core network (CN). Flexible solutions are needed to help with deploying ML models into various hardware environments, e.g., into environments with limitations in terms of computing power and support of strict latency requirements.

– Providing effective interfaces in a ML pipeline when a ML model is deployed.

For some ML pipelines where the ML model is embedded in a ML application, e.g., a ML pipeline for an access network, the interaction between the ML model and other nodes in the ML pipeline is implementation-specific. While in other cases, model inference is exposed to the network as web services and a standardized mechanism for serving these ML models is needed to ensure the workflow of the ML pipeline.

To address the above listed challenges, this Recommendation specifies an architectural framework for ML models serving in future networks including IMT-2020 by considering three fundamental stages, which are inference optimization, model deployment and model inference.

The inference optimization stage is the process by which the trained ML models are modified for better performance when executing inference in a certain deployment environment according to the requirements of the use case and the current state of the network. This is essential to achieve better performances, e.g., improved latency, throughput, power efficiency and memory consumption. Inference optimization may be achieved by both model-targeted optimization techniques, e.g., pruning, quantization, structural compression, graph conversion and layer fusion, as well as hardware specific ones, e.g., compiler acceleration and parallel computing acceleration.

NOTE 2 – Metadata generated from the training and usage of ML models in the network operators' networks may be used for inference optimization of the ML models.

The model deployment stage is aimed at getting the ML model ready to run in a specific deployment environment, which is followed by the inference stage where the model inference output result is applied to ML pipelines.

Based on the high-level requirements in clause 7, clause 8 of this Recommendation provides a high-level architecture, including reference points and sequence diagrams, aiming to provide mechanisms for ML model serving in future networks including IMT-2020.

7 High-level requirements

7.1 Inference optimization

This clause describes high-level requirements related to inference optimization of ML models before the ML models are deployed in ML pipeline subsystem.

REQ-ML-OPT-001: It is required that inference optimization of a ML model be determined based on the deployment environment in the ML underlay network.

NOTE 1 – Examples of deployment environment include real time deployment environment in the AN and cloud computing environment in the CN.

REQ-ML-OPT-002: It is required that inference optimization of a ML model be determined based on the up-to-date characteristics and status of the ML model.

NOTE 2 – Examples of ML model characteristics are the type of learning and structure of the ML models.

NOTE 3 – Examples of ML model status are observed performance values and optimization status.

NOTE 4 – Monitored performance reports of a ML model may be correlated with the type of data that the ML model has handled in order to decide what type of optimization to perform (e.g., pruning of decision trees).

REQ-ML-OPT-003: It is required that inference optimization of a ML model be determined based on changes in input data patterns.

NOTE 5 – Example of changes in data patterns are frequency of outliers' appearance, like frequency of call-drops or packet drops in the network, and change in number of network users in a location.

REQ-ML-OPT-004: It is recommended to update ML models in the ML model repository along with their corresponding optimization information after inference optimization.

NOTE 6 – Updated ML models can be reused in other ML pipelines deployed on similar hardware environment. Optimization information helps in the matching to similar runtime environments.

NOTE 7 – Examples of optimization information include the target environment, changes to the original model and effects of the optimization.

7.2 Model deployment

This clause describes high-level requirements for deploying ML models in ML pipeline subsystems.

REQ-ML-DEP-001: It is required that model testing and evaluation be done in the ML sandbox subsystem before model deployment in a ML pipeline to make sure the ML model meets the requirements of the ML pipeline.

NOTE 1 – For example, the ML model may be evaluated with respect to specific performance requirements as laid out in the ML pipeline requirements.

REQ-ML-DEP-002: It is required to select model runtime environments based on the resource requirements of the ML models and the resources' status of the ML underlay network.

NOTE 2 – For example, some ML models may require graphics processing unit (GPU) capabilities to achieve the desired performance while some may be optimized for other specific capabilities, e.g., in terms of central processing unit (CPU) and field programmable gate arrays (FPGA) characteristics.

NOTE 3 – An example of ML underlay network resources' status is the size of memory available for use in the ML underlay network. The deployment should prepare a ML model with a size less than the memory available for use in the ML underlay network.

REQ-ML-DEP-003: It is required to select a model deployment option based on the use case specification as indicated in the ML Intent [ITU-T Y.3172].

NOTE 4 – A use case specification may include latency and throughput requirements, performance requirements and location requirements. Model performance metrics may include precision, recall, area under curve (AUC) [b-AUC in classification] for classification models and mean squared error (MSE) for

regression models. A use case specification may also include how the prediction service is provided, e.g., in the form of web service or real-time streaming analytics, in which case the input data would be a stream of events triggering the prediction.

NOTE 5 – Deployment options are detailed in clause 8.2.

REQ-ML-DEP-004: It is recommended that model deployment operations be time-synchronized with events in the ML underlay network.

NOTE 6 – Examples of other events in the network are upgrading, scaling and maintenance of network functions in the underlying network. Example of time-synchronization is scheduling of model deployment in a particular NF after scheduled software upgrade in the NF.

REQ-ML-DEP-005: It is required to deploy ML models taking into account their performance status as evaluated in the ML sandbox subsystem and the capabilities of the ML underlay network along with network operator's preferences and policies.

NOTE 7 – Examples of ML underlay network's capabilities are the capabilities of NFs and computing platforms of the ML underlay network. For example, a User Plane Function (UPF) [b-ITU-T Y.3102] may need a capability for feature extraction and execution of ML models, while a Multi-access Edge Computing (MEC) [b-ETSI GS MEC 003] platform may need a capability for updating ML models based on optimizations identified by the Machine Learning Function Orchestrator (MLFO).

NOTE 8 – An example of network operator preferences and policies is the scheduling of ML model updates in the ML pipeline subsystem.

REQ-ML-DEP-006: It is required to configure the nodes of a ML pipeline during the deployment of the corresponding ML models.

7.3 Model inference

This clause describes requirements for a model inference stage in a ML pipeline subsystem.

REQ-ML-INF-001: It is required to perform model inference in coordination with the other nodes of the ML pipeline.

NOTE 1 – Examples of the coordination are that data collector is configured to collect input data according to the input features of deployed ML models and pre-processor processes the raw data to align with the format of the input specifications of the deployed ML models. (For details of the nodes of a ML pipeline, see clause 8.1 in [ITU-T Y.3172]).

REQ-ML-INF-002: It is required that versions of serving ML models be managed by authorized management functions.

NOTE 2 – A ML model may have different versions; during inference, one of the versions can be selected based on criteria like interface compatibility and performance trade-offs.

REQ-ML-INF-003: It is required that model inference enables the continuous monitoring and evaluation of the performance of ML models.

NOTE 3 – An example of monitoring results is whether the models are experiencing performance degradation over time. Inputs, outputs and exceptions of each inference request may be stored and used for such analysis.

REQ-ML-INF-004: It is recommended that authorized management functions be notified when the model performance deteriorates during model inference to a certain threshold which depends on each use case.

NOTE 4 – MLFO is an example of authorized management functions.

REQ-ML-INF-005: It is recommended the serving ML models be managed upon requests from authorized management functions during model inference.

NOTE 5 – The serving ML models may be stopped, restarted, updated and deleted by the authorized management functions.

REQ-ML-INF-006: It is required that profiles of serving ML models be exposed to the ML underlay network during model inference so that serving ML models can be discovered by model consumers to compose a ML pipeline.

NOTE 6 – Profile of a serving ML model may include its current status as well as metadata as defined in [ITU-T Y.3176].

REQ-ML-INF-007: It is required that, during inference, changes in data collection requirements of a serving ML model be notified to authorized management functions.

NOTE 7 – For example, changes in data collection requirements of a serving ML model may be caused by ML model optimization.

REQ-ML-INF-08: It is recommended to enable the updating of ML models during their inference to continuously improve their performance based on incoming input data.

NOTE 8 – This applies especially for ML use cases where data characteristics change dramatically over time. Online learning [b-Online learning] is an appropriate approach for such use cases.

REQ-ML-INF-009: It is recommended to support the scaling of the resources allocated to serving ML models.

NOTE 9 – Examples of parameters for scaling are the volume of input data and the performance thresholds of the use case.

NOTE 10 – The scaling decision may be taken by authorized management function (e.g., MLFO).

REQ-ML-INF-010: It is recommended to support load balancing between multiple instances of a given serving ML model.

REQ-ML-INF-011: It is required to support the selection of the appropriate serving ML model among different available models for a given inference request.

REQ-ML-INF-012: It is required to support multiple serving types satisfying the requirements of different use cases.

NOTE 11 – Examples of serving types are online prediction and batch prediction. Online prediction is useful for use cases with high latency requirements. Online prediction refers to real-time prediction where data generated from the network is sent to the serving ML model for inference immediately. Batch prediction refers to the prediction for a high volume of data per inference request.

REQ-ML-INF-013: It is required that fault management be supported including detecting whether the ML model fails to perform inference or to apply output to ML pipelines, as well as reporting and recovering from such failures.

NOTE 12 – The fault management may be performed with the support of authorized management functions (e.g., MLFO).

REQ-ML-INF-014: It is required to that accounting and licensing management be supported when a ML model is deployed on a public cloud environment.

REQ-ML-INF-015: It is required that model inference be exposed only to authorized entities in the network to minimize the risks of attacks.

NOTE 13 – Examples of attacks are evasion attack (wrong inference output caused by adversarial data) and poisoning attack (interference to the update of ML model caused by malicious data during online learning).

8 High-level architecture

8.1 Architectural components for ML model serving

The architectural components for ML model serving are shown in Figure 2. This is a simplified view of the main components along with external components (management subsystem and model repository) needed for ML model serving in the network.

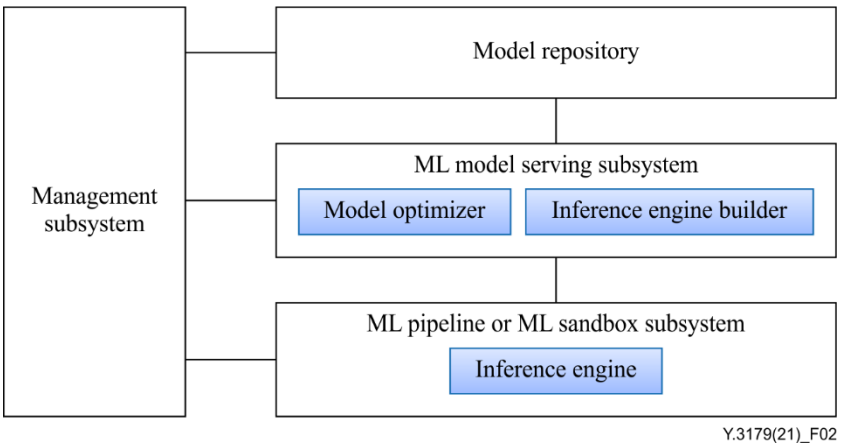

Figure 2 – Functional components for ML model serving

8.1.1 Model repository

This component is a repository of machine learning models [ITU-T Y.3174] from where ML models can be retrieved for the purpose of ML model serving.

NOTE – ML marketplace as described in [ITU-T Y.3176] may include a model repository.

8.1.2 ML model serving subsystem

The ML model serving subsystem, composed of model optimizer and inference engine builder, is used to adapt the model for specific deployment environments.

8.1.2.1 Model optimizer

The model optimizer component is responsible for inference optimization of a ML model that is ready to serve the ML underlay network. The inference optimization is performed based on the ML model and the deployment environment in order to achieve optimization goals, e.g., better throughput and latency. The inference optimization results in optimized ML models, the format of which may be different from the original ML models.

8.1.2.2 Inference engine builder

The inference engine builder component generates inference engines for ML models.

8.1.3 Inference engine

This functional component provides the following functionalities in ML pipeline or ML sandbox subsystem:

1) Runtime environment

Inference engine provides runtime environment for ML models. This enables:

– loading the ML model artifacts, including the model file and metadata.

– executing the operations defined in ML models so that ML models can accept input data and generate model inference output.

2) Model scheduling

Inference engine schedules the ML models based on the respective scheduling policies so that multiple serving ML models can perform inference efficiently.

3) Version management

– Inference engine manages the versions of the ML models according to the respective ML model update policy.

4) Model inference

The inference engine provides ML model inference capability for ML pipeline(s).

8.1.4 MLFO

The machine learning function orchestrator (MLFO) manages and orchestrates the model optimizer, inference engine builder and inference engine.

8.2 Deployment options

There are different kinds of inference hosts where the serving ML model may be hosted and inference be executed, e.g., an Internet of things (IoT) device, a server located at the network edge or in the cloud, or other NFs.

There are several options to deploy ML models into inference hosts in different hardware environments, as shown in Figure 3.

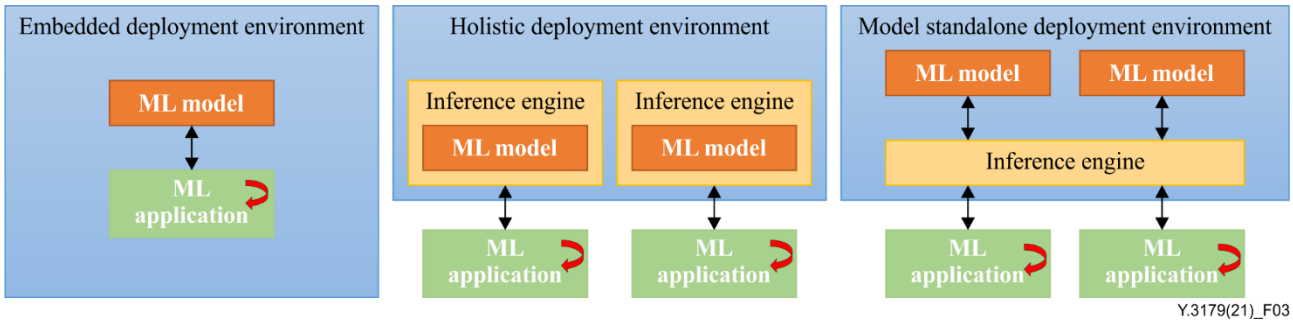

Figure 3 – Deployment options of ML models

1) Embedded deployment

Embedded deployment means ML models are built and packaged inside a ML application, where the ML pipeline is integrated within the ML application.

NOTE 1 – For this option, the integration of ML models within the ML application is out of scope and implementation dependent.

Such deployment reduces the latency to consume the model inference and is mostly used in resource constrained IoT scenarios.

2) Service-based deployment

Service-based deployment is usually used in cloud or edge computing platforms, where model inference is provided as a service via inference engine, which can then be consumed by ML pipelines in the networks. Management and orchestration functions, e.g., traffic routing, resource scaling and monitoring, and serving model management, may be provided for the inference engines in such deployments by the inference hosts.

NOTE 2 – The inference service may be an online service or batch prediction service.

– Holistic deployment

In the holistic deployment approach, a new inference engine integrating the ML model is created, and then deployed in the host as a whole. This type of deployment can be chosen, e.g., for cloud computing platforms that ML models are deployed at scale.

NOTE 3 – For this option, the ML model is hosted in the inference engine and the rest of the ML pipeline is integrated in the ML application.

– Model-standalone deployment

In the model-standalone deployment approach, ML models are loaded at run-time into already deployed service engines. The ML models can share the same, pre-deployed inference engine. This

type of deployment can be chosen, e.g., for edge computing platforms where ML models are transferred over low bandwidth connections or deployed over low memory budget.

NOTE 4 – For this option, the ML model is hosted separately from the inference engine and the rest of the ML pipeline is integrated in the ML application.

8.3 High level architecture

The architectural components for the serving of ML models in in future networks including IMT-2020 can be integrated in the architecture for ML marketplace specified in [ITU-T Y.3176], as shown in Figure 4. The reference points shown in Figure 4 correspond to the ones specified in [ITU-T Y.3172] and [ITU-T Y.3176].

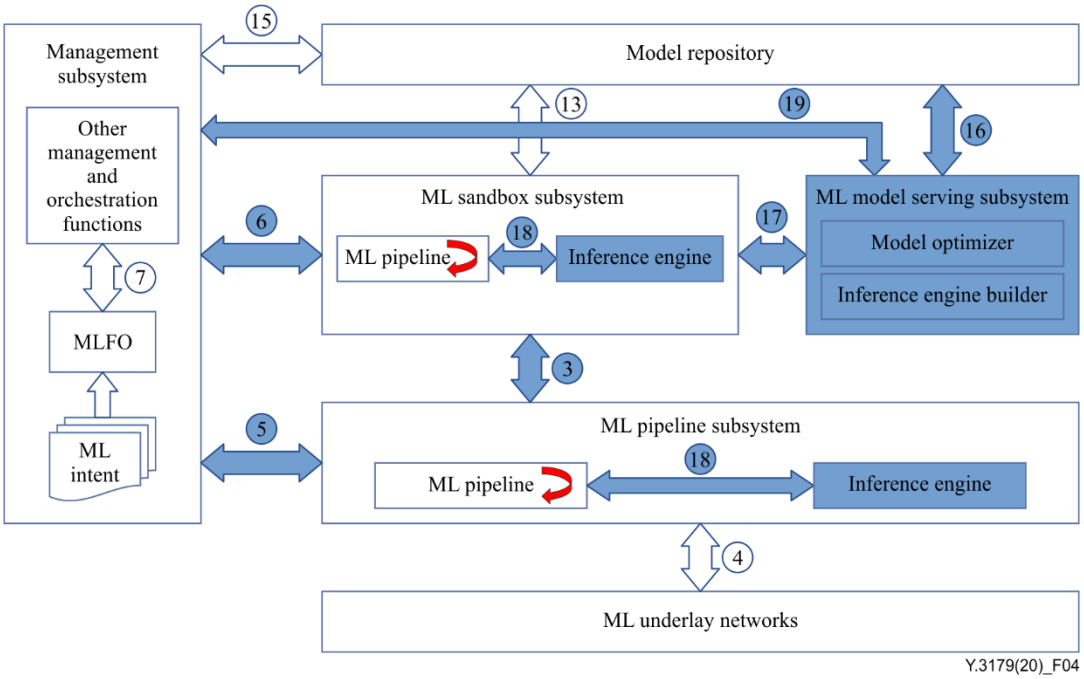

Figure 4 – High-level architecture for ML model serving

As shown in Figure 4, a ML model serving subsystem, which is comprised of model optimizer and inference engine builder, is introduced in the high-level architecture.

Model optimizer allows ML models obtained from a model repository via reference point 16 to be optimized.

Inference engine builder generates an inference engine for a ML model. The model is deployed in a ML sandbox subsystem for evaluation via reference point 17 and then is deployed as a serving ML model in the ML pipeline subsystem via reference point 3**.**

As far as the ML sandbox subsystem, the inference engine serves ML models for ML evaluation pipeline in the sandbox subsystem.

As far as the ML pipeline subsystem, the inference engine serves ML models for ML pipelines in the real network.

In the embedded deployment scenario, the ML application invokes the internal interface to load the embedded model and perform inference. In the service-based deployment scenario, instead of being tightly coupled with ML models, NFs that need ML capabilities consume the inference service exposed by the inference engine in a ML pipeline subsystem via reference point 18.

8.4 Reference points

Reference point 3, 5 and 6, which are used for ML model serving, are described in the following clauses.

8.4.1 Reference point 3 between ML sandbox subsystem and ML pipeline subsystem

This reference point is specified in [ITU-T Y.3172] and it is used for deployment or update of ML models in the ML pipeline subsystem.

8.4.1.1 Serving_model API

API description: When the ML pipeline subsystem receives the model deployment request from the MLFO, it can pull ML models (for service-based model-standalone deployment) or deploy inference engines (for service-based holistic deployment) from the ML sandbox subsystem.

Serving_model-request:

Direction: ML pipeline subsystem → ML sandbox

Information element	Type	Mandatory/Optional/ Conditional	Description
Deployment option	Enum	Mandatory	It is used to indicate the requested deployment option. Applicable values are service-based holistic deployment or service-based model-standalone deployment.
Model identifier	String	Conditional (See Note)	The model identifier for service-based model-standalone deployment.
Inference engine identifier	String	Conditional (See Note)	The inference engine identifier for service-based holistic deployment.
Version identifier	String	Optional	The version identifier specified for the serving ML model.
NOTE – Either the model identifier or inference engine identifier shall be present.			

Serving_model-response:

Direction: ML sandbox → ML pipeline subsystem

Information element	Type	Mandatory/Optional/ Conditional	Description
Result	Boolean	Mandatory	It indicates whether the serving request is successful.
Model identifier	String	Mandatory	The model identifier for service-based model-standalone deployment.
Inference engine identifier	String	Conditional	Inference engine identifier for service-based holistic deployment. NOTE – Only present in case of service-based holistic deployment

8.4.2 Reference point 5: Interface between management subsystem and ML pipeline subsystem

This reference point enables model management and monitoring functionality. After a ML model is pushed to ML pipeline subsystem, the MLFO triggers the monitoring operation in ML pipeline subsystem and receives the monitoring result reported from ML pipeline subsystem. The MLFO can also manage the registration and lifecycle of serving ML models.

8.4.2.1 Model_deployment API

API description: For a ML model that is evaluated in the ML sandbox subsystem, the MLFO can trigger the deployment from ML sandbox subsystem to ML pipeline subsystem.

Model_deployment-request:

Direction: MLFO → ML pipeline subsystem

Information element	Type	Mandatory/Optional/ Conditional	Description
Model identifier	String	Conditional NOTE	The model identifier for service-based model-standalone deployment.
Inference engine identifier	String	Conditional NOTE	The inference engine identifier for service-based holistic deployment.
Deployment configuration	<Attribute, value> array	Mandatory	The configuration data according to the requirements of the use case, e.g., deployment option (as described in clause 8.2), memory and CPU requirements, model scaling policy, batching policy, update policy, etc.
Serving model identifier	String	Mandatory	Identifier of the serving ML model.
Version identifier	String	Optional	The version identifier specified for the serving ML model.
NOTE – Either the model identifier or inference engine identifier shall be present.			

Model_deployment-response:

Direction: ML pipeline subsystem → MLFO

Information element	Type	Mandatory/Optional/ Conditional	Description
Result	Boolean	Mandatory	It indicates whether the deployment was successful.
Serving model identifier	String	Mandatory	Identifier of the serving ML model.
Version identifier	String	Optional	The version identifier specified for the serving ML model.

8.4.2.2 Model_registration API

API description: After a ML model is deployed in ML pipeline subsystem, its profile, including metadata, configuration, location and other basic information about the serving ML model is registered by inference engine so that model consumers can discover it from the registration information. This registration can be done in the MLFO via this API, i.e., the MLFO maintains the

profile of the model. Furthermore, if the status of the deployed model is updated, its profile in the MLFO will change accordingly. Similarly, inference engine can deregister a model from the MLFO and the MLFO will remove its profile.

NOTE 1 – The update of the model profile in the MLFO can be notified to the model consumer by polling or pub/sub mechanisms.

NOTE 2 – The MLFO can deregister a serving ML model when it is not available anymore because the MLFO cannot receive the periodic heartbeat sent from the serving ML model.

Model registration-request:

Direction: Inference engine in ML pipeline subsystem → MLFO

Information element	Type	Mandatory/Optional/ Conditional	Description
Serving model identifier	String	Mandatory	Identifier of the serving ML model.
Model metadata	<Attribute, value> array	Mandatory	The metadata of the serving ML model.
API information	String	Mandatory	It provides information about how the inference service is performed so that consumers of the inference service can discover the model.
Serving status	Boolean	Mandatory	It indicates whether the inference service of the serving ML model is available.
Version identifier	String	Mandatory	The version identifier specified for the serving ML model.

Model registration-response:

Direction: MLFO → Inference engine in ML pipeline subsystem

Information element	Type	Mandatory/Optional/ Conditional	Description
Result	Boolean	Mandatory	It indicates whether the registration was successful.
Serving model identifier	String	Mandatory	Identifier of the serving ML model.

8.4.2.3 Serving_model_management API

API description: This API is exposed by inference engine in ML pipeline subsystem to manage the serving ML models. Examples of model management operations enabled by this API are model start, stop, upgrade, scale, modify and delete operations, which may be used by an authorized management subsystem, e.g., MLFO.

Serving_model_management-request:

Direction: MLFO → Inference engine in ML pipeline subsystem

Information element	Type	Mandatory/Optional/ Conditional	Description
Serving model identifier	String	Mandatory	Identifier of the serving ML model.
Operation	Enum	Mandatory	The requested operation on the serving ML model. This includes model start, stop, upgrade, scale. modify, and delete operations.
Additional information	<Attribute, value> array	Mandatory	Additional information or configuration for the requested operation. For example, the "modify" operation may have additional information such as information on memory setting and model version.
Authentication information	String	Mandatory	Information for authenticating the operation requestor. If the authentication fails, the request is rejected.

Serving_model_management-response:

Direction: Inference engine in ML pipeline subsystem → MLFO

Information element	Type	Mandatory/Optional/ Conditional	Description
Result	Boolean	Mandatory	Indicates whether the model management operation was successful.
Serving model identifier	String	Mandatory	Identifier of the serving ML model.

8.4.2.4 Model_monitoring_subscription API

API description: After the ML model is deployed in ML pipeline subsystem, model monitoring can be exposed by this API, e.g., to the MLFO.

Model_monitoring_subscription-request:

Direction: MLFO → Inference engine in ML pipeline subsystem

Information element	Type	Mandatory/Optional/Conditional	Description
Serving model identifier	String	Mandatory	Identifier of the serving ML model whose monitoring event is subscribed to.
Notification Target Address	String	Mandatory	The address where the monitoring notification will be sent.
Monitoring event information	String list	Mandatory	Monitoring information the MLFO wants to be notified of, including model status; model performance (e.g., based on AUC and MSE); and exceptions and triggering events (e.g., based on periodicity or threshold).

Model_monitoring_subscription-response:

Direction: Inference engine in ML pipeline subsystem → MLFO

Information element	Type	Mandatory/Optional/ Conditional	Description
Result	Boolean	Mandatory	Indicates whether the model monitoring subscription was successful.
Serving model identifier	String	Mandatory	Identifier of the serving ML model.

8.4.2.5 Model_monitoring_event API

API description: When subscribed monitoring events are triggered, notification will be sent to all the subscribers.

Model_monitoring_event-notification:

Direction: Inference engine in ML pipeline subsystem → MLFO

Information element	Type	Mandatory/Optional/ Conditional	Description
Monitoring event information	String list	Mandatory	The information corresponding to the subscribed monitoring events.

8.4.2.6 Health_check API

API description: This API is used to check whether the serving ML model on the inference engine is active to process inference requests.

Health_check-request:

Direction: **MLFO**→Inference engine in ML pipeline subsystem

Information element	Type	Mandatory/Optional/ Conditional	Description
Serving model identifier	String	Mandatory	Identifier of a serving ML model.

Health_check-response:

Information element	Type	Mandatory/Optional/ Conditional	Description
Result	Boolean	Mandatory	Indicates whether the inference service is active.

8.4.3 Reference point 6: Interface between management subsystem and ML sandbox subsystem

This reference point is used for the management subsystem to manage the models pushed from the ML model repository to the ML sandbox subsystem. For example, the MLFO may trigger and monitor the training, optimization, chaining and deployment process of a ML model in the ML sandbox subsystem.

8.4.3.1 Model_monitoring_subscription API

API description: This API is similar to the model monitoring subscription API in clause 8.4.2.4, this one is used to monitor the ML model in the ML sandbox subsystem.

8.4.3.2 Model_monitoring_event_notification API

API description: This API is similar to the model monitoring event notification API in clause 8.4.2.5, this one is used to send monitoring notification of the ML model in the the ML sandbox subsystem.

8.4.3.3 Model_deployment API

API description: For a ML model that is trained in the ML sandbox subsystem, the MLFO can trigger model deployment in the ML sandbox subsystem for validation and evaluation.

Model_deployment-request:

Direction: MLFO → ML sandbox subsystem

Information element	Type	Mandatory/Optional/ Conditional	Description
Model Identifier	String	Mandatory	Identifier of the ML model.
Deployment configuration	<Attribute, value> array	Mandatory	The configuration data according to the requirements of the use case, e.g., deployment option (as described in clause 8.2), memory and CPU requirements, model scaling policy, batching policy, update policy, etc.
Version identifier	String	Optional	The version identifier specified for the ML model.

Model_deployment-response:

Direction: ML sandbox subsystem → MLFO

Information element	Type	Mandatory/Optional/ Conditional	Description
Result	Boolean	Mandatory	Indicates whether the deployment was successful.
Model identifier	String	Mandatory	Identifier of the ML model.
Inference engine identifier	String	Conditional	Identifier of the inference engine. NOTE – Present if deployment configuration in the model-deployment-request specifies service-based holistic deployment.

8.4.4 Reference point 16

This reference point is used for the interaction between ML model serving subsystem and model repository.

8.4.4.1 Model_Push API

The Model_Push API of reference point 14 in [ITU-T Y.3176] is reused for this API. It pushes trained ML models from model repository to model optimizer in ML model serving subsystem for optimization.

8.4.5 Reference point 17

This reference point is used for the ML sandbox subsystem to optimize ML models and to generate inference engine that can be deployed in the ML pipeline subsystem.

8.4.5.1 Serving_model API

This API is used for the ML sandbox subsystem to get the ML model or inference engine for evaluation from the ML model serving subsystem, similar to clause 8.4.1.1.

8.4.5.2 Model_push API

This API is used for the ML sandbox subsystem to push ML models to ML model serving subsystem for optimization, similar to Model_Push API in reference point 16.

8.4.6 Reference point 18

This reference point is used by ML pipeline to utilize the inference ability supported by the serving ML model. Through this reference point, model inference service is available to ML applications in a unified and modular manner, which ensures flexible integration of ML pipelines.

8.4.6.1 Inference API

API description: This API is used to get inference service from a specific serving ML model.

Inference-request:

Direction: ML inference consumer → Inference engine

Information element	Type	Mandatory/Optional/ Conditional	Description
Serving model identifier	String	Mandatory	Identifier of the serving ML model
Meta-data	<Attribute, value> array	Mandatory	The meta-data of the inference request, indicating the nature of the input and expected output, e.g., name and data format of the input data.
Input data	Byte array	Mandatory	The input data that needs to be passed to the serving ML model to get inference.

Inference-response:

Direction: Inference engine → ML inference consumer

Information element	Type	Mandatory/Optional/ Conditional	Description
Serving model identifier	String	Mandatory	Identifier of the serving ML model.
Result	Boolean	Mandatory	It indicates whether the request is successful.
Meta-data	<Attribute, value> array	Conditional	The meta-data of the response which can be used for the interpretation of the output data. NOTE 1 – Present if the result information element indicates the request is successful.
Output data	Byte array	Conditional	The output data of the inference request. NOTE 2 – Present if the result information element indicates the request is successful.

8.4.7 Reference point 19

This reference point is used for the management subsystem to manage the process in which a ML model is adapted to different ML pipeline subsystems.

8.4.7.1 Model_optimization API

API description: The newly trained model in the ML sandbox subsystem is triggered for optimization before it is deployed in the ML pipeline subsystem.

Model_optimization-request:

Direction: MLFO → model optimizer

Information element	Type	Mandatory/Optional/ Conditional	Description
Model Identifier	String	Mandatory	Used by the ML sandbox subsystem to identify the ML model.
Optimization info	<Attribute, value> array	Mandatory	Optimization information provided by the MLFO based on the ML Intent and the inference host capabilities. For example, target performance metrics, backend that the model will run on, operations and parameters of optimization, and framework to be followed for inference such as Adlik [b-Adlik] or TensorRT [b-TensorRT].

Model_optimization-response:

Direction: Model optimizer → MLFO

Information element	Type	Mandatory/Optional/ Conditional	Description
Result	Boolean	Mandatory	It indicates whether the optimization is successful.
Model Identifier	String	Mandatory	It indicates the identifier of the optimized model.
Result parameters	<Attribute, value> array	Conditional	Information of the changes to the ML model as a result of optimization. E.g., performance metrics like accuracy, latency and throughput NOTE - Present if the result information element indicates optimization is successful.

8.5 Sequence diagrams of the serving of ML models

This clause provides sequence diagrams related to the serving of ML models.

8.5.1 Model optimization in ML model serving subsystem

The sequence diagram for model optimization is shown in Figure 5.

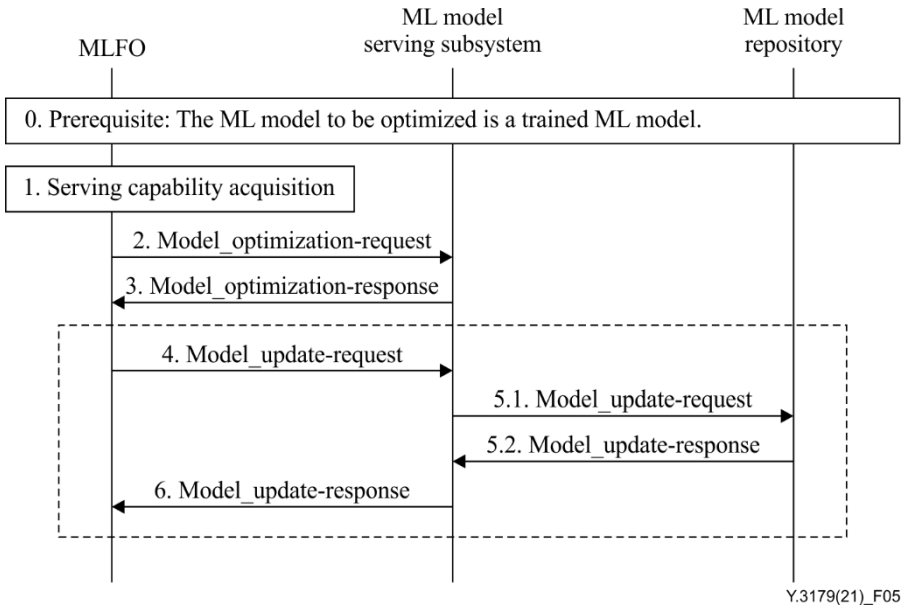

Figure 5 – Model optimization in ML model serving subsystem

Prerequisite: The ML model to be optimized is a trained ML model.

NOTE 1 – The ML model to be optimized can be either a ML model that was trained in the ML sandbox subsystem in a "model training" procedure and then pushed to serving subsystem via reference point 17, or a well-trained model that can be pushed from ML model repository via reference point 16 as indicated in "Model selection and push" procedure defined in [ITU-T Y.3176].

1. The MLFO acquires the serving capability information of the inference host in the ML underlay network through the other management and orchestration functions via reference point 7 in [ITU-T Y.3174]. The serving capability information includes the following:

– Network domain for the deployment, e.g., AN or CN.

– The information of NF properties, if the ML model has to be hosted inside a NF.

– The information of computing resources that can be used in the inference host, e.g., CPU cores, GPU capacity.

– The information of storage resources that can be used in the inference host.

– The information of the networking communication capabilities of the inference host, e.g., network bandwidth and quality of service (QoS).

2) The MLFO sends a model_optimization-request to the model optimizer in the ML model serving subsystem to indicate which model needs to be optimized. The request also includes some other optimization information according to the requirements implied in the ML Intent and the information on the serving capabilities.

3) Model optimizer determines the optimization operations for the ML model, tunes the optimization parameters and generates the output model, which is then evaluated in ML model serving subsystem to verify if the optimized performance meets the optimization goal. Then the ML model serving subsystem sends a model_optimization-response to the MLFO to indicate the result of the optimization.

Steps 4 to 6 happen only when the MLFO decides to update the optimized model as well as the optimization information to model repository based on network operators' ML model update policy and the result of the optimization. The MLFO triggers the model update procedure as indicated in clause 8.4.4 of [ITU-T Y.3176].

NOTE 2 – The difference between the model update in model optimization and that in model training lies in the content to update. Model parameters are updated in the model training whereas model structure is

updated in the model optimization. For example, model optimization may generate an optimized model with sparse connections by model pruning and fused layers by combining the processing of multiple layers of the model [b-Fused layer].

8.5.2 Model deployment in ML sandbox subsystem for evaluation

The sequence diagram for model deployment in the ML sandbox subsystem is shown in Figure 6.

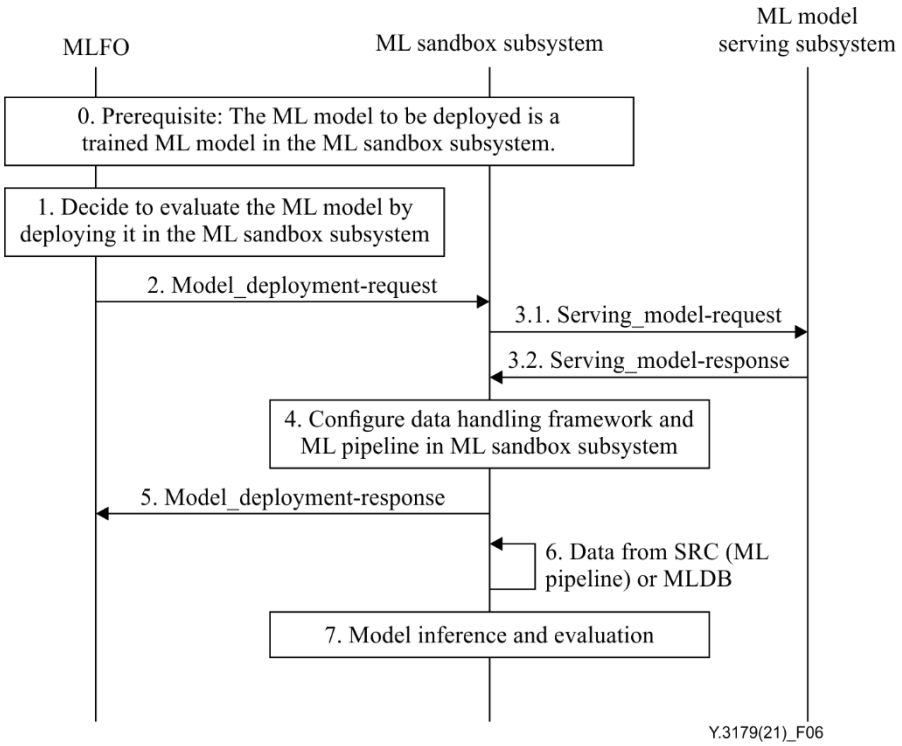

Figure 6 – Model deployment in ML sandbox subsystem for evaluation

Prerequisites: The model to be deployed is a trained ML model in the ML sandbox subsystem.

NOTE 1 – The model may be an already optimized model.

1) According to the criteria provided in the ML Intent, the MLFO decides to evaluate the ML model by deploying it in the ML sandbox subsystem.
2) The MLFO sends a model_deployment-request message to the ML sandbox subsystem which contains the model identifier, deployment configuration information and version identifier.
3) The ML sandbox subsystem sends a serving_model-request to the ML model serving subsystem via reference point 17. The inference engine builder in the ML model serving subsystem prepares an inference engine image for validation according to the inference configuration.
4) ML pipelines for data handling framework [ITU-T Y.3174] and simulated ML underlay networks [ITU-T Y.3172] are configured in the ML sandbox subsystem.
5) The ML sandbox subsystem sends the model_deployment-response to the MLFO to indicate the result of the deployment, including the inference engine identifier if it's service-based holistic deployment.
6) Data from the ML underlay network is used as input for model inference and evaluation.

8.5.3 Model deployment from ML sandbox subsystem to ML pipeline subsystem

The sequence diagram for model deployment from the ML sandbox subsystem to the ML pipeline subsystem is shown in Figure 7.

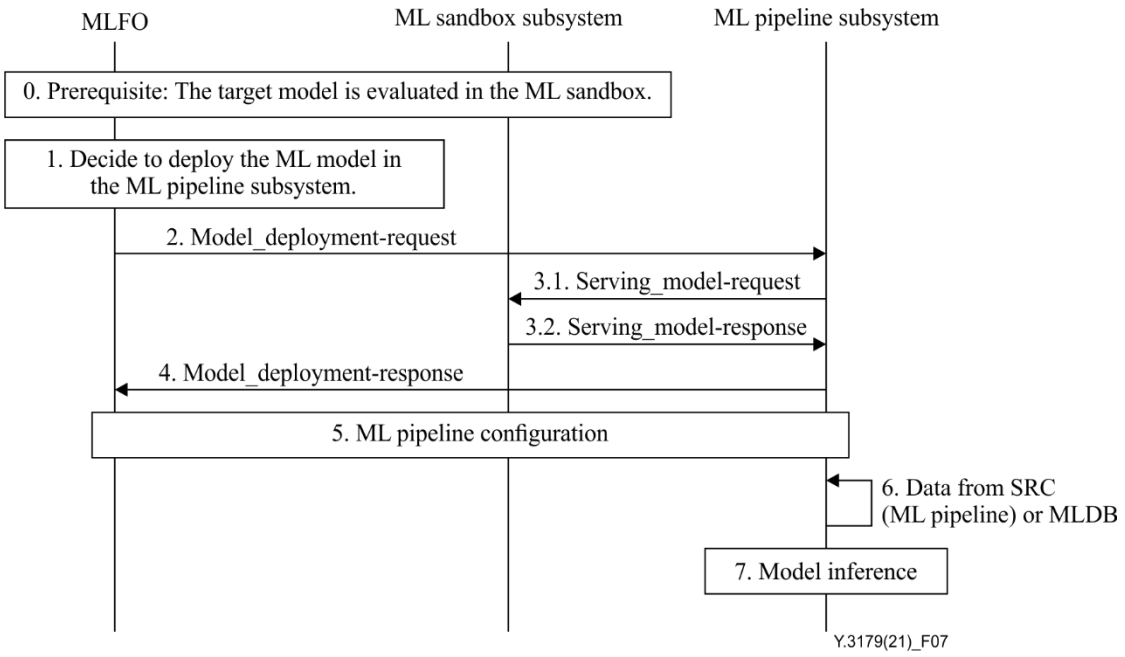

Figure 7 – Model deployment from ML sandbox subsystem to ML pipeline subsystem

Prerequisites: The ML model is evaluated in the ML sandbox subsystem.

1) Based on performance of ML model in the ML sandbox subsystem, the MLFO decides to deploy the ML model in the ML pipeline subsystem.
2) The MLFO sends a model_deployment-request message to the ML pipeline subsystem, including the serving model identifier, version identifier, deployment configuration, inference engine identifier (for service-based holistic deployment) or the model identifier (for service-based model-standalone deployment). The version identifier can be specified explicitly, or the latest version identifier will be chosen by ML sandbox subsystem.
3) ML pipeline subsystem sends a serving_model-request to the ML sandbox subsystem with the information of inference engine identifier (for service-based holistic deployment) or model id (for service-based model-standalone deployment), and the retrieved inference engine or ML model is deployed.
4) ML pipeline subsystem sends an asynchronous model_deployment-response to the MLFO to indicate whether the deployment is successful.
5) ML pipeline is configured. See the detailed procedure in clause 8.4.4.
6) Data from the ML underlay network or the MLDB [ITU-T Y.3174] is used as input data for model inference. The model inference consumer prepares the inference-request based on the input data, as specified in the profile of the serving ML model, and sends it to inference engine via reference point 18.
7) Inference engine then allocates the request to the serving ML model, and responds the inference consumer with the output inference data.

8.5.4 ML pipeline configuration in ML pipeline subsystem

The sequence diagram for the ML pipeline configuration in the ML pipeline subsystem is shown in Figure 8.

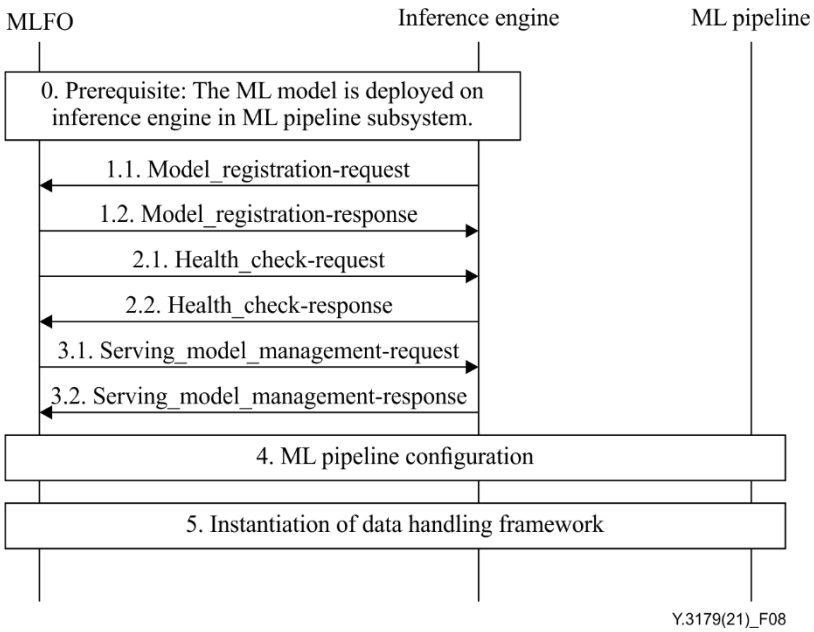

Figure 8 – ML pipeline configuration in ML pipeline subsystem with inference engine

Prerequisite: The ML model is deployed on inference engine in ML pipeline subsystem.

1) Inference engine in the ML pipeline subsystem registers the ML model using a model_registration-request to the MLFO.

2) The MLFO sends a health_check-request to the inference engine to check whether the inference service is running.

   NOTE 1 – Based on the network operator's policy, the MLFO can decide whether to stop an inference service when there is no ML pipeline configured to consume this service.

   NOTE 2 – Health check may be done periodically as per the configuration in the MLFO.

3) If the service is not running, the MLFO sends a serving_model_management-request to inference engine to start and get the inference service ready for getting data input and perform inference.

4) The serving model is configured as the model node in the pipeline, along with other nodes.

5) The data handling framework is instantiated as specified in [ITU-T Y.3174] and ML pipeline starts to send the collected data to the inference service and do further operations on the inference output.

8.5.5 Model monitoring

The sequence diagram for model monitoring is shown in Figure 9.

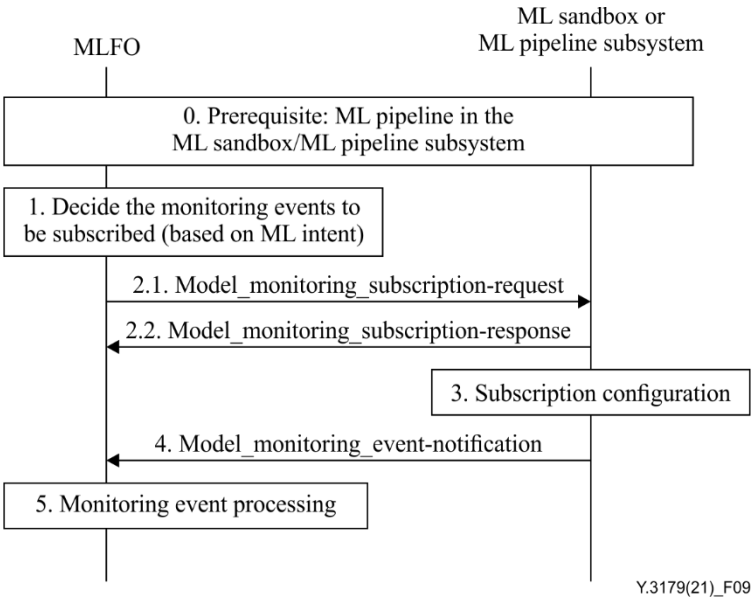

Figure 9 – Model monitoring

Prerequisites: ML pipeline (training or serving) is created in the ML sandbox subsystem (or the ML pipeline subsystem).

1) Based on the ML Intent, the MLFO selects the monitoring events to be subscribed.
2) The MLFO subscribes to the monitoring events of models in the ML sandbox subsystem (or the ML pipeline subsystem) using a model_monitoring_subscription-request.
3) Based on the subscription information, the MLFO is configured as a subscriber of the monitoring event of the ML model in the ML sandbox subsystem (or ML pipeline subsystem).
4) The ML sandbox or pipeline subsystem responds to the MLFO with a model_monitoring_event-notification. This may be a periodic event.
5) The MLFO processes the monitoring notification, e.g., the MLFO may trigger a model update, a scaling of serving instances.

8.5.6 Serving model update

Figure 10 shows serving model update. The sequence diagram for serving model update is shown in Figure 5.

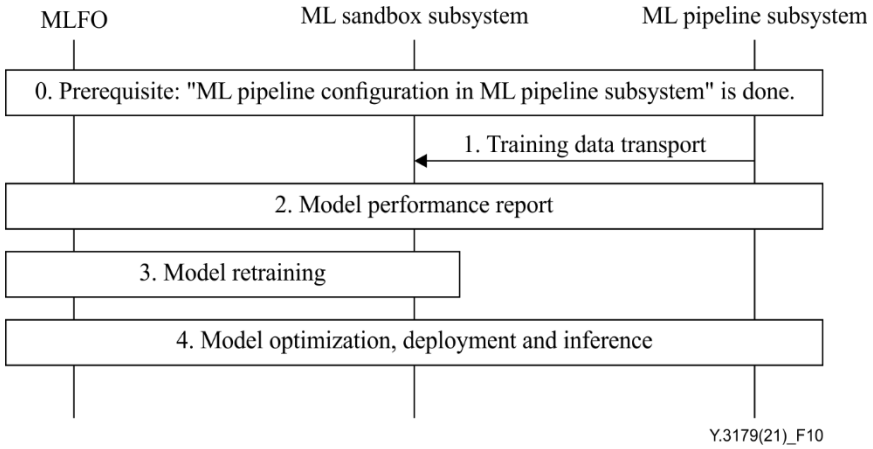

Figure 10 – Serving model update

Prerequisites: The ML pipeline configuration in ML pipeline subsystem is done.

1) The inference data during the inference stage is continuously sent to the ML sandbox subsystem as new training data according to the model update policy received from the MLFO in model_deployment-request in clause 8.5.3.
2) The model_monitoring_event-notification in ML pipeline subsystem is sent to the MLFO via model monitoring mechanism.
3) If the MLFO decides to update the model in the ML pipeline subsystem according to the model update policy and model performance status, the model is retrained in the ML sandbox subsystem.
4) The steps that follow are model optimization as specified in clause 8.5.1, deployment as specified in clause 8.5.3 and inference as specified in clause 8.5.4 based on the instruction of the MLFO.

9 Security considerations

The security considerations provided in [ITU-T Y.3172] are applicable to this Recommendation.

Additional specific security considerations concern model security evaluation (e.g., analysing the characteristics of ML model to evaluate risk of evasion attack). Moreover, to ensure a robust ML pipeline, the reliability of the inference results needs to be assessed before applying the output of a serving ML model to the network.

Bibliography

[b-ITU-T Y.3100] Recommendation ITU-T Y.3100 (2017), *Terms and definitions for IMT-2020 network.*

[b-ITU-T Y.3102] Recommendation ITU-T Y.3102 (2018), *Framework of the IMT-2020 network.*

[b-ITU-T Y-Sup.55] ITU-T Supplement 55 to Y.3170-series (2019), *Machine learning in future networks including IMT-2020: use cases.*

[b-Adlik] Adlik: *Toolkit for Accelerating Deep Learning Inference*. https://wiki.lfaidata.foundation/display/ADLIK (Referenced 10.03.2020).

[b-AUC in classification] Classification: ROC Curve and AUC. https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc (Referenced 10.03.2020).

[b-ETSI GS MEC 003] ETSI GS MEC 003 V2.2.1 (2020-12), *Multi-access Edge Computing (MEC); Framework and Reference Architecture.*

[b-Fused layer] M. Alwani, H. Chen, M. Ferdman and P. Milder, *Fused-layer CNN accelerators*, 2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), Taipei, 2016, pp. 1-12, doi: 10.1109/MICRO.2016.7783725.

[b-Kubernetes] Concepts of Kubernetes. https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/ (Referenced 10.03.2020).

[b-LF Acumos] Acumos AI Marketplace. https://www.acumos.org (Referenced 10.03.2020).

[b-Online learning] Online Methods in Machine Learning Theory and Applications. http://www.mit.edu/~rakhlin/6.883/ (Referenced 10.03.2020).

[b-TensorRT] NVIDIA TensorRT https://developer.nvidia.com/tensorrt (Referenced 10.03.2020).

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series D	Tariff and accounting principles and international telecommunication/ICT economic and policy issues
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Cable networks and transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Environment and ICTs, climate change, e-waste, energy efficiency; construction, installation and protection of cables and other elements of outside plant
Series M	Telecommunication management, including TMN and network maintenance
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling, and associated measurements and tests
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks, open system communications and security
Series Y	**Global information infrastructure, Internet protocol aspects, next-generation networks, Internet of Things and smart cities**
Series Z	Languages and general software aspects for telecommunication systems

Printed in Switzerland
Geneva, 2021