

International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

Y.3531

(09/2020)

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

Cloud Computing

**Cloud computing – Functional requirements for
machine learning as a service**

Recommendation ITU-T Y.3531

ITU-T



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.3531

Cloud computing – Functional requirements for machine learning as a service

Summary

Recommendation ITU-T Y.3531 provides cloud computing requirements for machine learning as a service (MLaaS), which addresses requirements from use cases. MLaaS is a cloud service category in which the capability provided to the cloud service customer is the provision and use of a machine learning (ML) framework, which is a set of functionalities for provisioning ML data, as well as training, deployment and management of an ML model.

From the perspective of cloud computing service provision, Recommendation ITU-T Y.3531 provides the functional requirements for MLaaS to identify functionalities such as ML data pre-processing, ML model training and ML model testing. Also, Recommendation ITU-T Y.3531 is aligned with the cloud computing reference architecture specified in Recommendation ITU-T Y.3502.

History

Edition	Recommendation	Approval	Study Group	Unique ID*
1.0	ITU-T Y.3531	2020-09-29	13	11.1002/1000/14405

Keywords

BaaS, big data, big data as a service, cloud computing, MLaaS, machine learning, machine learning as a service.

* 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 received notice of intellectual property, protected by patents, 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 TSB patent database at <http://www.itu.int/ITU-T/ipr/>.

© ITU 2020

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

Table of Contents

	Page
1	Scope 1
2	References..... 1
3	Definitions 1
3.1	Terms defined elsewhere 1
3.2	Terms defined in this Recommendation 2
4	Abbreviations and acronyms 2
5	Conventions 3
6	Overview of machine learning..... 3
6.1	Introduction to machine learning..... 3
6.2	Generic process of machine learning 4
6.3	Machine learning ecosystem 5
7	Machine learning as a service..... 7
7.1	System context of MLaaS 7
7.2	CSN:machine learning data provider 8
7.3	CSN:machine learning model developer 9
7.4	CSP:machine learning service provider 9
7.5	CSC:machine learning service user 11
8	Functional requirements of MLaaS 11
8.1	ML data collection and storage requirements 11
8.2	ML data labelling requirements..... 12
8.3	ML data pre-processing requirements 12
8.4	ML data analysis and feature engineering requirements 12
8.5	ML model training requirements 13
8.6	ML model monitoring requirements 14
8.7	Trained ML model deployment and retraining requirements..... 14
9	Security considerations 15
Appendix I – Use case of MLaaS for operation perspectives..... 16	
I.1	ML data annotation and labelling management 16
I.2	Model training with user configuration 17
I.3	Report learning result and re-training ML model..... 18
I.4	Distributed training with multiple worker nodes..... 19
I.5	Model testing and optimizing the model quality includes hyperparameter tuning 21
I.6	Model monitoring to issue alerts of abnormal or unsuspected learning process 22
I.7	Model deployment and monitoring 23
I.8	Automated machine learning in cloud computing..... 24

	Page
Appendix II – Use case of MLaaS for application perspectives.....	25
II.1 Object recognition model development in the cloud computing environment.....	25
II.2 Traffic speed prediction and monitoring service.....	26
II.3 Image recognition.....	27
II.4 Face recognition.....	28
II.5 Image segmentation model development.....	29
II.6 Generative adversarial model development.....	30
Bibliography.....	32

Recommendation ITU-T Y.3531

Cloud computing – Functional requirements for machine learning as a service

1 Scope

This Recommendation provides system context, functional requirements and use cases for machine learning as a service (MLaaS).

In particular, the scope of this Recommendation includes:

- an overview of machine learning (ML);
- an introduction to MLaaS;
- functional requirements of MLaaS.

The use cases of MLaaS are developed to derive its functional requirements.

NOTE – Development of ML algorithms and methodologies lie outside the scope of this Recommendation.

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.2201] Recommendation ITU-T Y.2201 (2011), *Requirements and capabilities for ITU-T NGN*.
- [ITU-T Y.2701] Recommendation ITU-T Y.2701 (2007), *Security requirements for NGN release 1*.
- [ITU-T Y.3502] Recommendation ITU-T Y.3502 (2014), *Information technology – Cloud computing – Reference architecture*.
- [ITU-T Y.3600] Recommendation ITU-T Y.3600 (2015), *Big data – Cloud computing based requirements and capabilities*.

3 Definitions

3.1 Terms defined elsewhere

This Recommendation uses the following terms defined elsewhere:

3.1.1 activity [ITU-T Y.3502]: A specified pursuit or set of tasks.

3.1.2 cloud computing [b-ITU-T Y.3500]: Paradigm for enabling network access to a scalable and elastic pool of shareable physical or virtual resources with self-service provisioning and administration on-demand.

NOTE – Examples of resources include servers, operating systems, networks, software, applications, and storage equipment.

3.1.3 cloud service [b-ITU-T Y.3500]: One or more capabilities offered via cloud computing (3.1.2) invoked using a defined interface.

3.1.4 cloud service customer [b-ITU-T Y.3500]: Party which is in a business relationship for the purpose of using cloud services.

NOTE – A business relationship does not necessarily imply financial agreements.

3.1.5 cloud service partner [b-ITU-T Y.3500]: Party which is engaged in support of, or auxiliary to, activities of either the cloud service provider or the cloud service customer, or both.

3.1.6 cloud service provider [ITU-T Y.3500]: Party which makes cloud services available.

3.1.7 machine learning [b-ITU-T Y.3172]: Processes that enable computational systems to understand data and gain knowledge from it without necessarily being explicitly programmed.

NOTE 1 – This definition is adapted from [b-ETSI GR ENI 004].

NOTE 2 – Supervised machine learning and unsupervised machine learning are two examples of machine learning types.

3.1.8 machine learning model [b-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).

3.1.9 metadata [b-ITU-T H.752]: Structured, encoded data that describe characteristics of information-bearing entities to aid in the identification, discovery, assessment and management of the described entities.

3.1.10 role [ITU-T Y.3502]: A set of activities that serves a common purpose.

3.1.11 sub-role [ITU-T Y.3502]: A subset of the activities of a given role.

3.2 Terms defined in this Recommendation

This Recommendation defines the following terms:

3.2.1 machine learning as a service (MLaaS): A cloud service category in which the capabilities provided to a cloud service customer are the provision and use of a machine learning framework.

3.2.2 machine learning framework: A set of functionalities for provisioning machine learning data, as well as training, deployment and management of a machine learning model.

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

CPU	Central Processing Unit
CSC	Cloud Service Customer
CSN	Cloud Service partner
CSP	Cloud Service Provider
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
ML	Machine Learning
MLaaS	Machine Learning as a Service
MLDP	Machine Learning Data Provider

MLMD	Machine Learning Model Developer
MLSP	Machine Learning Service Provider
MLSU	Machine Learning Service User

5 Conventions

The following conventions are used in this Recommendation:

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

6 Overview of machine learning

6.1 Introduction to machine learning

Machine learning (ML) is a technique in computer science to enable machines or computers to learn how to perform tasks without being explicitly programmed. According to the definition, the mechanism of ML is distinct from explicit programming, which applies if-then statements to make decisions. The goal of ML is to improve their ability to solve tasks automatically through data.

The development of ML involves ML models, learning algorithms and training data. An ML model consists of mathematical representations for performing tasks with input data. The learning algorithm trains an ML model using training data for the task.

The ML developer configures the ML model and learning algorithm, and gathers training data based on setting up tasks. The tasks are designed to solve problems by performance evaluation metrics based on computational methods. An evaluation metric is a criterion for the measurement of the performance of an ML model. Examples of general tasks for ML are prediction, classification, clustering and sample generation.

ML tasks are classified according to the characteristics of the learning algorithm, such as the necessity for training data or feedback. Categories of ML task include: supervised learning; unsupervised learning; semi-supervised learning; and reinforcement learning.

- **Supervised learning:** a learning task with labelled data, which maps input data to desired output data.
- **Unsupervised learning:** a learning task with unlabelled data, which finds structures or patterns in the data.
- **Semi-supervised learning:** a task of learning with both labelled and unlabelled data, where there is a small amount of labelled data.
- **Reinforcement learning:** a task of learning in which an agent uses environment data to perform a task and then provides feedback.

6.2 Generic process of machine learning

The process is implemented by ML data acquisition, ML data processing, ML model development and ML model deployment. Figure 6-1 shows the generic process of ML.

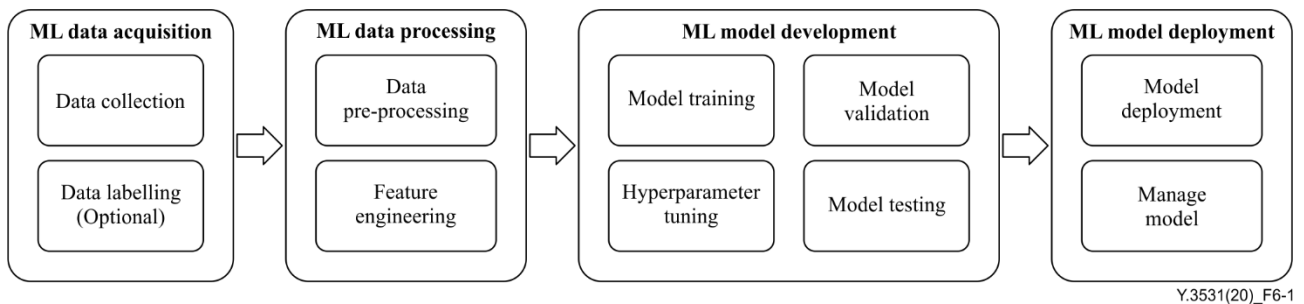


Figure 6-1 – Generic process of machine learning

- **ML data acquisition** collects data for training. The collected data are grouped into datasets for training, validation and test before ML model development.
 - **Data collection** gathers raw data, including that which is structured, unstructured and semi-structured. Within all types, data can exist in formats such as text, spreadsheet, video, audio, image and map. [ITU-T Y.3600].
 - **Data labelling** optionally generates labelled data for supervised learning or semi-supervised learning. Human resources and labelling tools are required to annotate data. The tools use tagging units, e.g., image data use a tagging unit of a bounding box to indicate a target object and video data use a tagging unit of 5 s videos for labelling.
- **ML data processing** handles data to improve learning performance or create meaningful information from data.
 - **Data pre-processing** transforms data to resolve inaccurate, incomplete or outlier data. Noise and bias in raw data are removed or mitigated for training data.
 - **Feature engineering** finds and determines the useful features for an ML model from the data. Feature engineering algorithms to implement ML include: feature selection; scaling; and extraction.
 - **Feature selection** obtains a subset from the original features of those with the same or similar analytical results. Examples of feature selection are: genetic algorithm; greedy forward selection; and correlation feature selection.
 - **Feature scaling** normalizes the range of data in the original features. Examples of feature scaling are: mean normalization; and min-max normalization.
 - **Feature extraction** derives new from original features by set reduction. Examples of feature extraction are: principal component analysis; and dimensionality reduction.
- **ML model development** trains and optimizes the ML model. Model training, model validation, model testing and hyperparameter tuning are iteratively performed to meet the purpose of the ML model.
 - **Model training** fits the parameters of the ML model to the training dataset. Training algorithms are implemented to update parameters such as backpropagation and gradient descent. The parameters are adjusted or optimized automatically by feeding in a training dataset.
 - **Model validation** verifies the performance of the ML model with a validation dataset. The model validation process gives the opportunity to optimize hyperparameters before ML training is completed.

- **Model testing** measures the performance of the final trained ML model with the test dataset. Model testing gives the final performance report to deploy the trained ML model. Model testing prevents re-learning when the ML model performs poorly after the ML model is deployed.
- **Hyperparameter tuning** adjusts the hyperparameters of ML model training. The hyperparameters are adjustable values for controlling ML model training. The hyperparameters include: iteration time; batch size; and epoch.
- **ML model deployment** utilizes the model to perform tasks in applications. ML model deployment involves deployment and management of the model, including retraining performance monitoring and management.
 - **Model deployment** loads the trained ML model into the application or hardware. The ML developer uses the trained model to develop ML applications.
 - **Manage models** updates or monitors the ML model. In addition, manage models requests model retraining.

Figure 6-2 shows an example of trained ML model development that includes the life cycle following the generic process of ML.

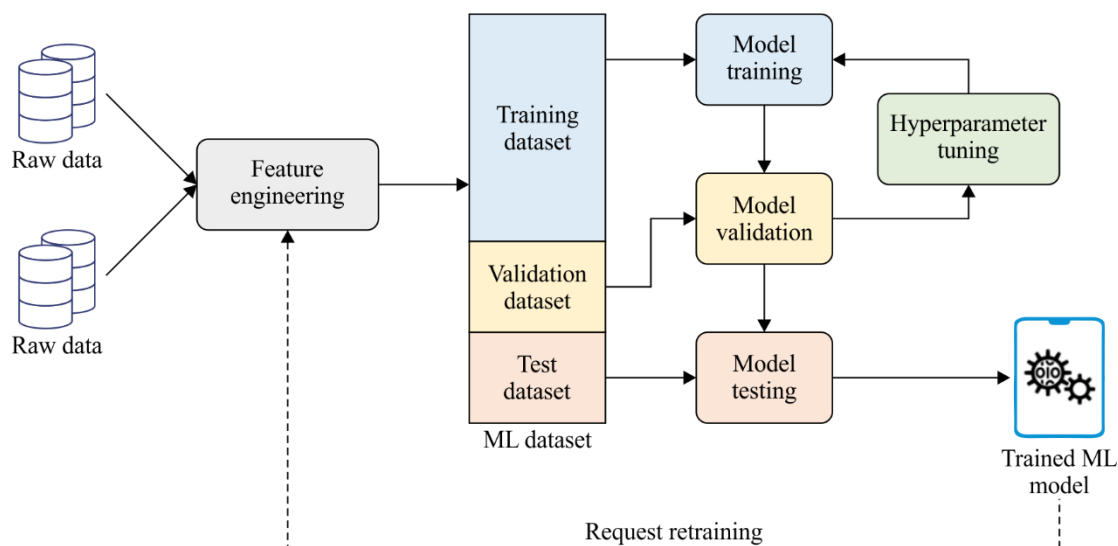


Figure 6-2 – Example of generic ML process

In Figure 6-2, the final output of a ML process is a trained ML model that has the ability to perform tasks on devices. If a deployed trained ML model falls below acceptable performance, then the model needs to be retrained or reengineered. The deployed model is retrained using a new dataset to meet the performance required for a given purpose.

To achieve its goal, ML requires well-designed models and learning algorithms, as well as an amount of data for training. Several learning algorithms and ML models have been developed to improve task-solving performance.

6.3 Machine learning ecosystem

The ML ecosystem consists of stakeholders that provide ML data, the ML model and ML framework. In this ecosystem, developers can effectively build their own ML or application.

This clause describes an ML ecosystem through stakeholder roles and sub-roles. It describes the activities required for the ML roles to provide and consume ML, as well as the relationships among the roles.

The ML ecosystem includes the following roles:

- data provider;
- ML model provider;
- ML framework provider;
- ML framework customer.

The ML ecosystem is shown in Figure 6-3.

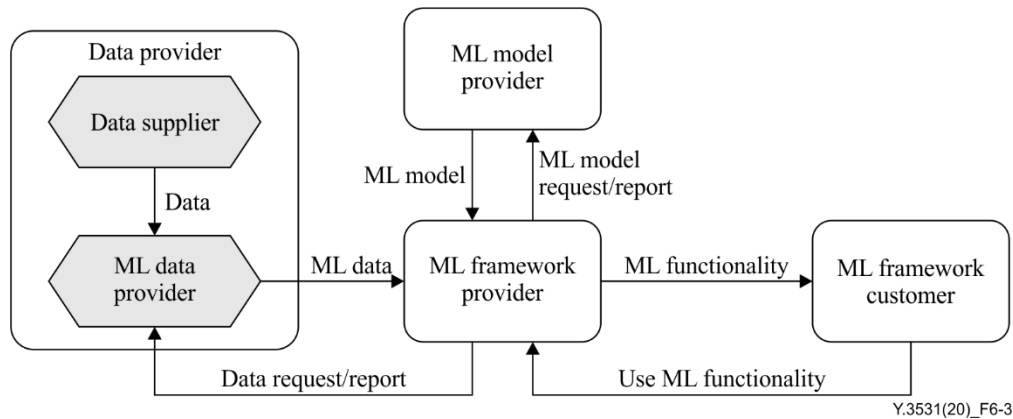


Figure 6-3 – High-level machine learning ecosystem

6.3.1 Data provider

The data provider (DP) role consists of two sub-roles:

- data supplier;
- ML data provider.

6.3.1.1 Data supplier

A data supplier provides data from different sources and performs activities specified in [ITU-T Y.3600]. The data supplier's activities include:

- data generation;
- creation of metadata information describing data source(s) and relevant attributes;
- publication of metadata information to access metadata.

6.3.1.2 ML data provider

An ML data provider acquires data and performs data processing for ML processing. It receives data from a data supplier [ITU-T Y.3600], then sends data for ML processing to an ML framework provider. The ML data provider supports various kinds of data, e.g., structured, unstructured and streaming.

The ML data provider's activities include:

- analysis of features of data;
- dataset generation for ML.

NOTE 1 – Activities such as analysing data and pre-processing are described in [ITU-T Y.3600].

NOTE 2 – Analysis of features from data includes labelling tasks by human resources. This activity is optional if the ML model requires labelled data for training.

6.3.2 ML model provider

An ML model provider provides an ML model to an ML framework provider. Developing an ML model involves ML algorithms for handling data, learning models and evaluating models in the process, as well as designing the schema of the ML model.

The ML model provider's activities include:

- ML model development;
- ML model provision.

6.3.3 ML framework provider

An ML framework provider provides ML functionalities to an ML framework customer. The ML framework includes an interface for using ML functionalities. The ML framework provider also manages and reports activities to an ML framework customer to develop an ML model.

The ML framework provider's activities include:

- training an ML model with ML data;
- ML model management and reporting;
- ML model deployment for an ML application;
- ML functionality provision to an ML framework customer.

6.3.4 ML framework customer

An ML framework customer uses ML functionalities from an ML framework provider for business, e.g., decision making, business process automation and customer interaction. The ML framework can be an end-user or a system that performs ML-enabled applications.

The ML framework customer's activities include:

- ML framework access and utilization.

7 Machine learning as a service

7.1 System context of MLaaS

MLaaS is a cloud service category in which the capabilities provided to a cloud service customer (CSC) are the provision and use of an ML framework. ML processing needs a large amount of computing power and resources for ML model training due to the large amount of training data and the highly complex computation involved in ML model training. MLaaS resolves the problem by providing elastic computing capabilities and resources in cloud environments based on CSC requests.

The system context of MLaaS provides additional sub-roles and activities based on the architectural user viewpoint established in [ITU-T Y.3502]. This clause describes how cloud computing supports the four main roles in an ML ecosystem: ML data provider, ML model provider, ML framework provider and ML framework customer.

Cloud computing sub-roles can be mapped to ML roles as shown in Table 7-1. The sub-roles of a cloud service provider (CSP), cloud service partner (CSN) and CSC mapped to the roles and sub-roles of an ML ecosystem.

Table 7-1 – Mapping of a machine learning ecosystem to an MLaaS system context

Machine learning ecosystem	MLaaS system context
ML data provider	CSN:MLDP (machine learning data provider)
ML model provider	CSN:MLMD (machine learning model developer)
ML framework provider	CSP:MLSP (machine learning service provider)
ML framework customer	CSC:MLSU (machine learning service user)

Figure 7-1 illustrates the cloud computing sub-roles in MLaaS. Figure 7-1 also identifies activities specific to ML and assigns them to cloud computing sub-roles. MLaaS utilizes other sub-roles of CSP and CSN.

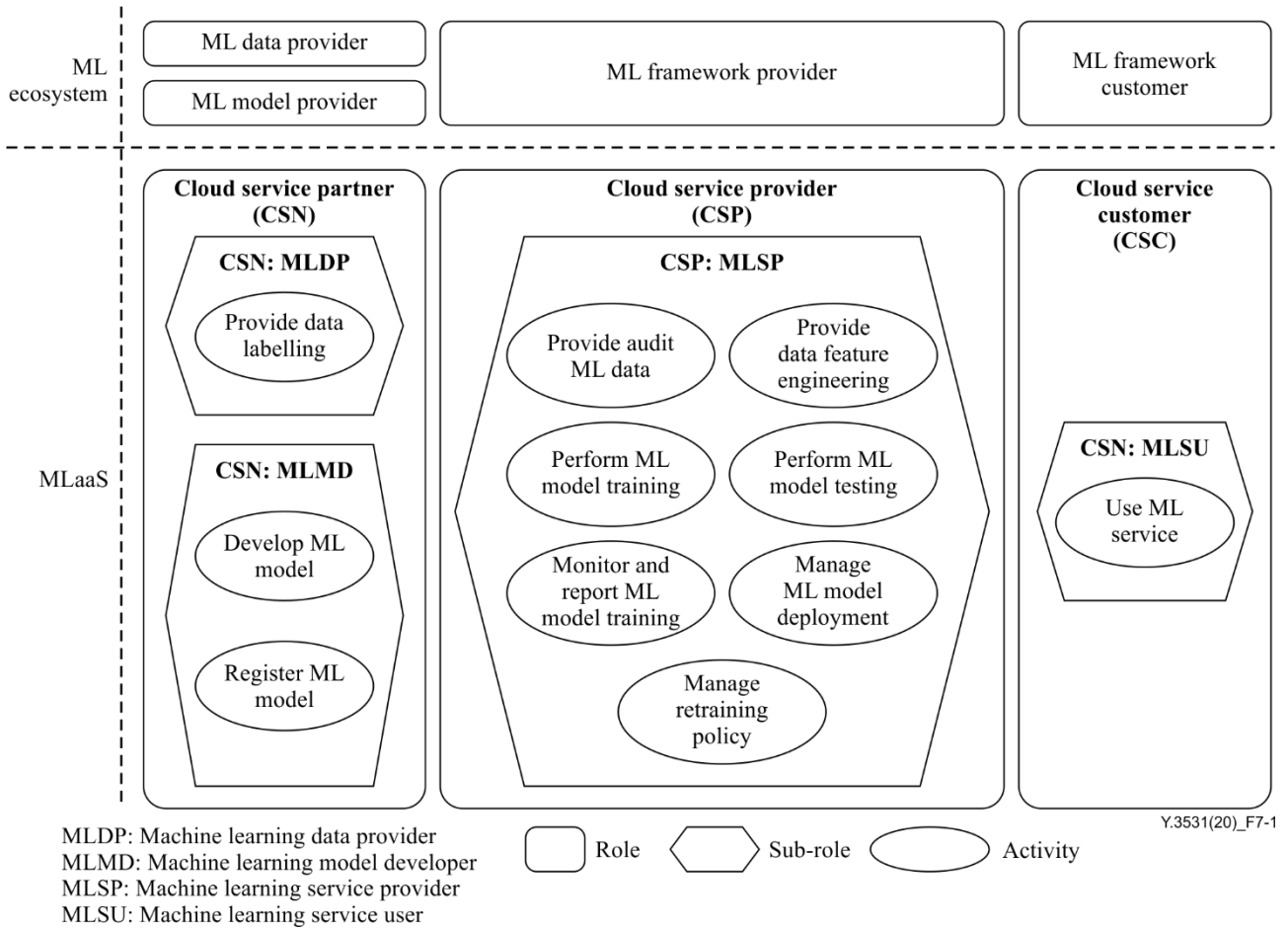


Figure 7-1 – MLaaS system context

7.2 CSN:machine learning data provider

CSN:MLDP is a sub-role of the CSN, which provides data labelling and activities of CSN:DP as specified in [ITU-T Y.3600]. The activities of CSN:DP for data include generation, publication and brokerage. Additional activities of CSN:MLDP include:

- data labelling provision.

7.2.1 Data labelling provision

Data labelling provision involves the generation of labelling information with the tools for this purpose. Human labellers manually annotate data in accordance with guidelines given by the tools. This activity involves:

- labelling generation in accordance with guidelines from the CSP:MLSP;

- metadata provision for labelled data;
- CSN:DP catalogue updating for a CSP:MLSP to search ML data.

7.3 CSN:machine learning model developer

CSN:MLMD is a sub-role of a CSN, which supports an ML model in the solution of various learning tasks. A CSN:MLMD generates an ML model catalogue for discovery and utilization of a ML model. An ML model catalogue includes ML model metadata, such as scheme, version, usage and evaluation metric.

NOTE – ML model usage include the applicable learning task of an ML model.

The activities of an CSN:MLMD for an ML model include:

- development;
- registration.

7.3.1 ML model development

ML model development involves developing and updating ML models, and publication of the ML model with its information. This activity involves:

- ML model development by setting initial values of a parameter;
- ML model updating with feedback and reporting information;
- hyperparameter configuration provision for ML model training;
- ML model metadata generation including its usage, input/output data format and expected performance.

7.3.2 ML model registration

ML model registration is the provision of an ML model catalogue to a CSP:MLSP. The ML model catalogue includes ML model information with metadata.

NOTE – The ML model can be provided with a structured format established by ML frameworks.

This activity involves:

- ML model access information provision to the CSP:MLSP;
- catalogue provision for the CSP:MLSP to search an appropriate ML model.

7.4 CSP:machine learning service provider

CSP:MLSP is a sub-role of the CSP, which provides MLaaS including infrastructures and tools for ML model training, deployment and management. In addition, a CSP:MLSP supports the collection and auditing of ML data.

NOTE – CSP:MLSP can provide the activities of CSP:BDIP specified in [ITU-T Y.3600]. The related activities of CSP:BDIP are data collection, data storage, data pre-processing and data integration.

The additional activities of CSP:MLSP include:

- ML data audit;
- data feature engineering;
- ML model training;
- ML model testing;
- ML model training monitoring and reporting;
- ML model deployment management;
- retraining policy management.

7.4.1 ML data audit

ML data audit supports labelling tasks by providing universalized and managed tools for them. This activity encourages improvement in labelling quality and reliability of the ML dataset. This activity involves:

- providing labelling environments and tools for a CSN:MLDP;
- auditing ML data for labelling quality;
- reporting feedback to the CSN:MLDP.

7.4.2 Data feature engineering

Data feature engineering employs methods, such as selection, scaling and extraction from ML data, for features. Feature engineering supports expedition of the training process and manages the risk of an underperforming ML model.

NOTE – The overfitting problem is an example of an underperforming ML model. Overfitting occurs when the model fits the data too closely, and fails to generalize for prediction.

7.4.3 ML model training

ML model training executes the training process for an ML model, including its validation. This activity involves:

- creation of a virtual machine and storage for ML model training;
- triggering and operation of ML model training with input for training relevant information, such as hyperparameter configuration;
- provision of configurations for partitioning the ML data into datasets for training, validation and test;
- ML model performance validation;
- registration and storage of the results of ML model training.

7.4.4 ML model testing

ML model testing evaluates a trained ML model before it is deployed, ensuring that its performance and quality target the tasks.

7.4.5 ML model training monitoring and reporting

ML model training monitoring and reporting support the model training process by providing functionalities such as measuring resource utilization, alerting abnormalities and automatic stopping. The report history can be utilized to detect errors in the ML model scheme and determination of retraining policy. This activity involves:

- measurement and monitoring resource utilizations during ML model training;
- reporting a problem if ML model training cannot be maintained with the ML model provided;
- reporting the performance of the ML model during ML model training;
- automatic stoppage if the configured threshold is exceeded.

NOTE – A CSC:MLSU configures threshold values for automatic stopping, such as resource utilization overloads.

7.4.6 ML model deployment management

ML model deployment management supports export of the trained ML model for deployment in the target computing environment. A CSP:MLSP supports transformation into multiple formats of an ML model for various computing environments. In addition, the exported models are registered for retraining and updating of the ML model catalogue.

NOTE 1 – The trained ML model can be updated in the catalogue and provided to service an ML model for transfer learning.

NOTE 2 – Transfer learning is a technique of application of a pre-trained ML model to a related learning task.

7.4.7 Retraining policy management

Retraining policy management determines the process of retraining for a CSC:MLSU. Retraining ensures and maintains the performance of an ML model or improves its performance. Retraining policy considers the expected performance of a deployed model, monitoring time period, subsequent following actions, etc.

7.5 CSC:machine learning service user

CSC:MLSU is a sub-role of the CSC, which utilizes an ML framework and cloud services to develop ML applications according to the user's intention.

CSC:MLSU activity includes:

- ML service use.

7.5.1 ML service use

ML service use involves invoking and using a ML framework and cloud service to develop ML applications.

8 Functional requirements of MLaaS

8.1 ML data collection and storage requirements

[ITU-T Y.3600] provides data collection and storage functional requirements in terms of big data. The functionalities of MLaaS also involve data collection and storage, and several functional requirements of these ML activities do not differ from those of big data specified in [ITU-T Y.3600].

The following functional requirements are inherited from those for big data specified in [ITU-T Y.3600] by changing the role from big data to ML, such as CSP:BDIP to CSP:MLSP and CSN:DP to CSN:MLDP.

The data collection and storage requirements include the following.

- 1) It is required that a CSP:MLSP support the collection of data from multiple CSN:MLDPs in parallel.
- 2) It is recommended that a CSN:MLDP expose data to the CSP:MLSP by publication of metadata.
- 3) It is recommended that a CSP:MLSP support data collection from different CSN:MLDPs with different modes.

NOTE 1 – Data can be collected in different modes, such as pull mode in which the data collection process is initiated by CSP:MLSP or push mode in which the data collection process is initiated by the CSN:MLDP.

- 4) It is recommended that a CSN:MLDP provide a brokerage service to the CSP:MLSP for searching accessible data.

NOTE 2 – Brokerage provides a data catalogue that has data information such as data specification, data instructions, electronic access methods, licence policy and data quality.

- 5) It is required that a CSP:MLSP support different data types with sufficient storage space, elastic storage capacity and efficient control methods.
- 6) It is required that a CSP:MLSP support storage for different data formats and data models.

NOTE 3 – Data formats include text, spreadsheet, video, audio, image and map. Data models include relational models, document models, key-value models and graph models (as described in clause 6.1 of [ITU-T Y.3600]).

8.2 ML data labelling requirements

The ML data labelling requirements include the following.

- 1) It is required that a CSP:MLSP provide tools for labelling tasks for CSN:MLMD.
NOTE 1 – The tools of labelling tasks provide different types of tagging unit for data formats such as image, video and text.
- 2) It is required that a CSP:MLSP provide audit data for labelled ML data.
NOTE 2 – The audit data for labelling are utilized to enhance data labelling quality such as accuracy of labelling and consistency of labelling.
NOTE 3 – The audit data is reported to a CSN:MLDP as feedback when the request arrives or when the labelling quality is poor for developing ML model.
- 3) It is required that a CSN:MLDP provide information about data labelling tasks.
NOTE 4 – Information about data labelling tasks includes data type, unit of labelling and consensus majorities of labellers.

8.3 ML data pre-processing requirements

As in clause 8.1, the following requirements are partially inherited from big data functional requirements in [ITU-T Y.3600], derived by changing the role from big data to ML such as CSP:BDIP to CSP:MLSP.

The ML data pre-processing requirements include the following.

- 1) It is required that a CSP:MLSP support data aggregation.
NOTE 1 – Data from different sources can be organized in the same model or data format, as described in clause 6.1 of [ITU-T Y.3600].
- 2) It is recommended that a CSP:MLSP support unification of data collected in different formats.
NOTE 2 – Unification of data is used for example to unify data about persons, locations or dates extracted from web pages, pictures, videos, SNS data and calling logs to text format.
- 3) It is recommended that a CSP:MLSP support extraction of data from unstructured data or semi-structured data into structured data.
NOTE 3 – This requirement can be applied also to data storage.

The additional requirements of ML data pre-processing not described in [ITU-T Y.3600] include the following.

- 4) It is required that a CSP:MLSP provide a configuration for splitting ML data into datasets for training, validation and test.
NOTE 4 – The training, validation and test datasets are divided completely independently.

8.4 ML data analysis and feature engineering requirements

As in clause 8.1, the following requirements are inherited from big data functional requirements in [ITU-T Y.3600], derived by changing the role from big data to ML such as CSP:BDIP to CSP:MLSP.

The ML data analysis requirements include the following.

- 1) It is required that a CSP:MLSP support analysis of various data types and formats.
- 2) It is required that a CSP:MLSP support association analysis.
NOTE 1 – Association analysis is the task of uncovering relationships among data.

- 3) it is required that a CSP:MLSP support different data analysis algorithms.
NOTE 2 – Data analysis algorithms include those for classification, clustering, regression, association and ranking.

In addition, preferences for ML data feature engineering not described in [ITU-T Y.3600] include the following.

- 4) It is recommended that a CSP:MLSP provide feature selection to determine a subset of relevant features of ML data.
- 5) It is recommended that a CSP:MLSP provide feature scaling to normalize the range of features of ML data.
- 6) It is recommended that a CSP:MLSP provide feature extraction to generate new improved features from the original features of ML data.

8.5 ML model training requirements

The ML model training requirements include the following.

- 1) It is required that a CSN:MLMD provide a registry of the ML model and catalogue to CSP:MLSP.
- 2) It is recommended that a CSP:MLSP provide feedback on ML model usage to CSP:MLMD.
NOTE 1 – The feedback includes applied learning tasks with high performance experienced by a CSC:MLSU. The CSP:MLMD use the feedback to update the ML model.
- 3) It is required that a CSP:MLSP configure hyperparameter values.
- 4) It is recommended that a CSN:MLMD provide default configuration hyperparameter values.
- 5) It is recommended that a CSN:MLMD provide a restricted range of values for hyperparameter adjustments.
NOTE 2 – The values in a restricted range are the adoptable hyperparameter values for the ML model.
- 6) It is recommended that a CSP:MLSP provide a visualization of learning results.
NOTE 3 – Examples of visualization are a chart, table, distribution plo, and graph, which show analytic information about a learning result.
- 7) It is required that a CSP:MLSP provide ML model training operations.
NOTE 4 – ML model training operations include initiate, stop and resume ML model training.
- 8) It is required that a CSP:MLSP provide an ML model validation process with a validation dataset.
- 9) It is recommended that a CSP:MLSP provide learning status monitoring during ML model training.
NOTE 5 – Learning status includes expected learning time, applied hyperparameter set and memory usage.
- 10) It is required that a CSP:MLSP provide performance evaluation for a trained ML model.
- 11) It is required that a CSP:MLSP store evaluation results of trained ML models with an applied hyperparameter set.
- 12) It is recommended that a CSP:MLSP provide hyperparameter optimization methods.
NOTE 6 – Hyperparameter optimization involves choosing a set of optimal hyperparameters for ML model training. Examples of hyperparameter optimization methods are grid search, Bayesian optimization and evolutionary optimization.
- 13) It is recommended that a CSP:MLSP provide automated ML model search methods.
NOTE 7 – Automated ML model search finds the design of an ML model so as to increase the performance for the target learning task.

- 14) It is recommended that a CSP:MLSP provide a transformation of ML models for use in other ML frameworks.
- 15) It is recommended that a CSP:MLSP provide distributed ML model training.
NOTE 8 – Distributed ML model training is the training of an ML model in multiple worker nodes to accelerate production of results.

8.6 ML model monitoring requirements

ML model monitoring requirements include the following.

- 1) It is required that a CSP:MLSP provide monitoring of resource utilization during ML model training.
NOTE 1 – Resource utilization includes a processing unit (such as a central processing unit (CPU) or graphics processing unit (GPU)), memory, storage and network utilization.
- 2) It is recommended that a CSP:MLSP issue resource utilization overload alerts during ML model training.
- 3) It is required that a CSP:MLSP report the history of resource utilization with timestamps.
- 4) It is required that a CSP:MLSP provide the automatic stoppage by detecting learning failure or measuring unpromising model performance.
NOTE 2 – Detection of learning failure includes ML parameter update failures.
- 5) It is recommended that a CSP:MLSP set threshold values for automatic stoppage.
NOTE 3 – Automatic stoppage is executed with user-defined threshold values to avoid unwanted early results. Unwanted results include overtraining and decreasing performance.
- 6) It is recommended that a CSN:MLMD provide the default threshold values for automatic stoppage.
- 7) It is required that a CSP:MLSP store the history of learning failure during ML model training.
NOTE 4 – The history includes a log of resource utilization, performance measurement and execution of automatic stoppage.

8.7 Trained ML model deployment and retraining requirements

The trained ML model deployment and retraining requirements include the following.

- 1) It is required that a CSP:MLSP provide a registry of a trained ML model.
NOTE 1 – A trained ML model is registered with a schema including applied input/output data and ML model structures of.
- 2) It is required that a CSP:MLSP provide trained ML model metadata.
NOTE 2 – Trained ML model metadata includes the evaluated performance and applied hyperparameter set.
- 3) It is required that a CSP:MLSP export a trained ML model with a format applicable to target hardware deployment.
- 4) A CSP:MLSP can optionally provide performance monitoring for the ML model deployed.
- 5) It is required that a CSP:MLSP provide retraining policy to manage the performance of a trained ML model.
NOTE 3 – The retraining policy includes reset learning parameter, add new data for learning to optimize the ML model.
- 6) It is recommended that a CSP:MLSP provide ML model retraining according to measured ML model performance.

9 Security considerations

It is recommended that the security framework for cloud computing described in [b-ITU-T X.1601] be considered for the MLaaS. [b-ITU-T X.1601] analyses security threats and challenges in the cloud computing environment and describes security capabilities that could mitigate these threats and meet security challenges.

[b-ITU-T X.1631] provides guidelines supporting the implementation of information security controls for CSCs and CSPs. Many guidelines aid CSPs to assist CSCs in implementing the controls and guide them to implement such controls. Selection of appropriate information security controls and the application of the implementation guidance provided depends on a risk assessment, as well as any legal, contractual, regulatory or other cloud-sector specific information security requirements.

Relevant security requirements of [ITU-T Y.2201], [ITU-T Y.2701] and applicable ITU-T X, ITU-T Y and ITU-T M series of Recommendations need to be taken into consideration, including access control, authentication, data confidentiality, data retention policy, network security, data integrity, availability and privacy.

Appendix I

Use case of MLaaS for operation perspectives

(This appendix does not form an integral part of this Recommendation.)

The use cases in this appendix provide examples of operating MLaaS functionalities and related functional requirements of MLaaS.

I.1 ML data annotation and labelling management

Title	ML data annotation and labelling management
<p>Description</p>	<p>This use case describes the management procedure for ML data, which includes assigning data, generating annotation, reporting result and merging ML data. The following are specific steps for managing ML data with annotators.</p> <ol style="list-style-type: none"> 1) CSP:MLSP sets the ML data server and collects raw data from CSN:DP. 2) CSP:MLSP requests annotation to be assigned to raw data. 3) CSN:MLDP annotates raw data using the annotation method provided by CSP:MLSP. 4) CSN:MLDP provides the annotated dataset to CSP:MLSP. 5) CSP:MLSP applies the decision policy to the annotated data. <ol style="list-style-type: none"> A. CSP:MLSP decides 'accept' or 'reject' for annotated data from each annotator. B. CSP:MLSP saves or reports the results of quality of annotated data. C. CSP:MLSP merges the accepted data into the ML dataset
<p>Role or sub-role</p>	<p>CSN:MLDP</p>
<p>Figure (optional)</p>	<p>The diagram illustrates the workflow for ML data annotation and labelling management. On the left, a box labeled 'CSN: MLDP' contains a stack of cylinders representing 'Data #1', 'Data #2', and 'Data #N'. An arrow points from this box to a central box labeled 'CSP: MLSP' which contains a person icon and a cylinder labeled 'ML data set'. From the 'CSP: MLSP' box, three arrows labeled 'Request annotation' point to three separate boxes, each labeled 'CSN: MLDP' and containing a person icon and labeled 'Annotator #1', 'Annotator #1', and 'Annotator #K' respectively. Each 'Annotator' box has an arrow labeled 'Annotate' pointing to a cylinder representing 'ML data #1', 'ML data #2', and 'ML data #N' respectively. Dashed lines from these 'ML data' cylinders converge and point to a box at the bottom containing three numbered steps: '1) Accept or reject annotated data', '2) Report annotated results from annotator', and '3) Merge accepted data into ML data set'. Dashed arrows from this box point back to the 'CSP: MLSP' box. A small reference code 'Y.3531(20)_Fl.1' is located at the bottom right of the diagram area.</p>
<p>Pre-conditions (optional)</p>	<p>CSN:MLDP operates ML data server to schedule or assign data resources to annotators. CSN:MLDP searches and requests raw data for annotation from CSN:DP</p>
<p>Post-conditions (optional)</p>	<p>CSN:MLDP provides an interface for reporting ML results and learning progress to CSC:MLSU.</p>

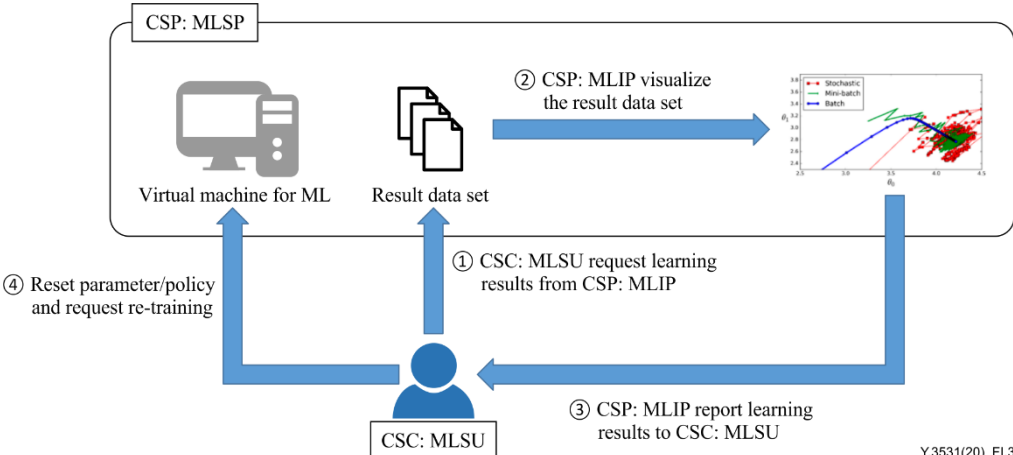
Title	ML data annotation and labelling management
	CSN:MLSP stores the trained or optimized ML model for developing ML applications
Derived requirements or recommendations	<ul style="list-style-type: none"> – Clause 8.2 items 1), 2), 3) – Clause 8.1 items 1) ,3) ,5), 6)

I.2 Model training with user configuration

Title	Model training with user configuration
Description	<p>This use case describes the model training procedure in cloud computing. A single machine for training is considered as default training. The following are general steps for model training in this use case.</p> <ol style="list-style-type: none"> 1) CSP:MLSP installs a virtual machine with a machine learning engine that has an interface for model training with the user. 2) CSN:MLMD provides an ML model for ML training to CSP:MLSP. 3) CSN:MLMD provides appropriate ML data for ML model to CSP:MLSP. NOTE – For the preparation of an ML model and data pair, CSC:MLSU requests the ML model and appropriate ML data for the model, or CSP:MLSP provides pair of ML model an ML data for CSC:MLSU. This preparation scenario lies outside the scope of this use case. 4) CSC:MLSU configures and sets the learning policy and parameters for machine learning training. 5) CSP:MLSP trains the ML model and tracks the performance of the training result for reporting to CSC:MLSU. 6) CSP:MLSP saves the trained model and training result in the designated source
Role or sub-role	CSN:MLMD CSN:MLDP CSP:MLSP CSC:MLSU
Figure (optional)	<p>The diagram illustrates the model training process. At the bottom, a user icon labeled 'CSC:MLSU' sends a request (Step 4) to 'CSP:MLSP'. The request includes learning policy and parameters. 'CSP:MLSP' then provides the ML model (Step 2) and ML data (Step 3) to a virtual machine (Step 1). The virtual machine trains the model and saves learning results (Step 5), producing a trained ML model and a result data set (e.g., learning history). The diagram also shows 'CSN:MLMD' providing the ML model and 'CSN:MLDP' providing the ML data. A legend at the bottom right identifies the components: 'Result data set (e.g., learning history)' and 'Trained-ML model'.</p>

Title	Model training with user configuration
Pre-conditions (optional)	CSC:MLSU installs virtual machine with CSP to build an ML model. CSP provides an ML framework or platform tools to build an ML model. CSN:MLSU searches and requests ML data and model from CSN:MLDP and CSN:MLMD
Post-conditions (optional)	CSP:MLSP provides the interface for reporting ML results and learning progress to CSC:MLSU. CSP:MLSP stores the trained or optimized ML model for developing ML applications
Derived requirements or recommendations	<ul style="list-style-type: none"> – Clause 8.1 items 1), 2), 4) – Clause 8.3 items 1), 2), 3), 4) – Clause 8.5 items 1), 3), 6), 7), 9), 10), 11)

I.3 Report learning result and re-training ML model

Title	Report learning result and re-training ML model
Description	<p>This use case describes the reporting of learning results from CSP:MLSP to CSC:MLSU, and re-training an ML model. The objective of reporting learning results is generally to give information about handling and managing an ML training configuration for CSC:MLSU. CSC:MLSU can optimize ML learning parameters and allow manual modification of ML learning policy; in addition, CSC:MLSU can request a re-training option. The following are general steps for this use case.</p> <ol style="list-style-type: none"> 1) CSC:MLSU requests a learning report from CSP:MLSP. 2) CSP:MLSP transforms and visualizes the learning report on appropriate interfaces. NOTE 1 – Visualization options can be provided to CSC:MLSU from CSP:MLSP. The raw data of the training result is the default option. 3) CSP:MLSP reports the learning result to CSC:MLSU 4) CSC:MLSU analyses the learning report and manages and optimizes the learning policies. NOTE 2 – The reset and re-training steps are the same as those in 'model training with user configuration' use case (clause I.2)
Role or sub-role	CSP:MLSP CSC:MLSU
Figure (optional)	 <p style="text-align: right; font-size: small;">Y.3531(20)_FI.3</p>
Pre-conditions (optional)	CSN:MLSP already performs ML learning and stores backup result data for ML model
Post-conditions	

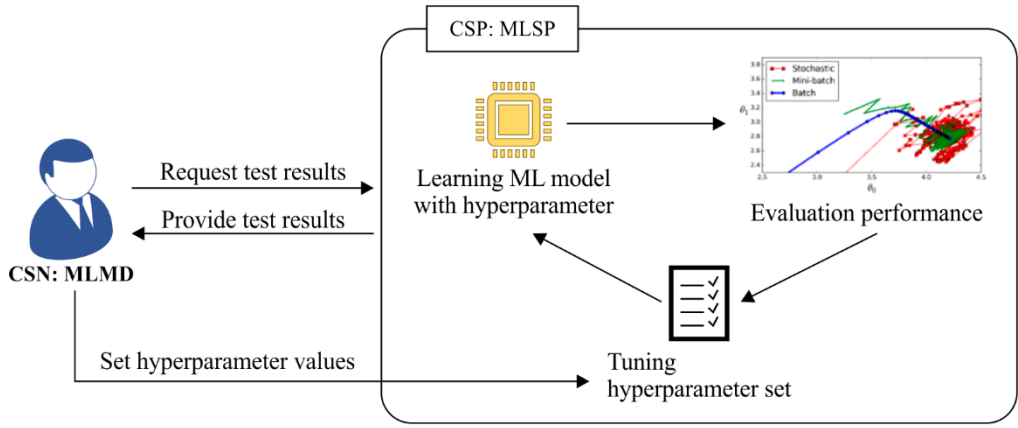
Title	Report learning result and re-training ML model
(optional)	
Derived requirements or recommendations	<ul style="list-style-type: none"> – Clause 8.6 items 1), 2), 3), 7) – Clause 8.7 item 5)

I.4 Distributed training with multiple worker nodes

Title	Distributed training with multiple worker nodes
Description	<p>This use case describes model training with multiple worker nodes to support parallel and distributed learning. CSC:MLSU can organize multiple worker nodes for ML training. This option has many advantages such as reductions in data size for each worker node and division of data into private and public. The following are general steps for distributed training.</p> <ol style="list-style-type: none"> 1) CSC:MLSU designs or organizes the architecture of ML worker nodes. 2) CSP:MLSP provides ML management for distributed training to CSC:MLSU. 3) CSC:MLSU sets the distribution policy for an ML model or data and schedules policy for assigning resources and ML parameters. 4) CSP:MLSP assigns resources and parameters using the configured policy from CSC:MLSU. 5) CSP:MLSP requests ML training from each virtual or local worker node and collects the training results. 6) CSP:MLSP iterates steps 4) and 5) until training is completed
Role or sub-role	<p>CSP:MLSP CSC:MLSU</p>

Title	Distributed training with multiple worker nodes
Figure (optional)	<p>1) Set learning parameter scheduling 2) Set connection topology</p> <p>CSC: MLSU</p> <p>1) Allocation of learning data to multiple worker nodes</p> <p>2) Distributed learning and updating parameter to ML management</p> <p>Y.3531(20)_FI.4</p>
Pre-conditions (optional)	<p>CSP provides virtual server to manage ML parameters and control distributed learning policy.</p> <p>CSP provides network connection with multiple worker nodes on other cloud and local environment</p>
Post-conditions (optional)	
Derived requirements or recommendations	<p>– Clause 8.5 items 3), 6), 7), 8), 9), 10), 15)</p>

I.5 Model testing and optimizing the model quality includes hyperparameter tuning

Title	Model testing and optimizing the model quality includes hyperparameter tuning
Description	<p>This use case describes the model testing procedure in cloud computing. The model testing or validation process is usually performed to optimize or generalize a model's performance and quality. The testing may require an iterative process to tune some model perspective resources and hyperparameters. Generally, the poor performance of an ML model results from: 1) lack of feature prediction; 2) non-optimal hyperparameters; and 3) abnormal learning data. The purpose of model testing is usually related to the resolution of problems 1) and 2). In that context, this use case mainly targets resolution of problems 1) and 2) experimentally.</p> <p>The following are general steps for model testing in this use case.</p> <ol style="list-style-type: none"> 1) CSN:MLMD requests learning result from CSP:MLSP. 2) CSP:MLSP reports the initial or previous learning result to CSN:MLMD. NOTE – The model testing process can be performed iteratively. 3) CSN:MLMD optimizes the ML model with a given dataset. <ol style="list-style-type: none"> a) CSN:MLMD tunes the hyperparameters of an ML model. b) CSN:MLMD adds features for ML data. 4) CSN:MLMD tests the ML model until the performance of ML model is qualified
Role or sub-role	CSN:MLMD CSP:MLSP CSC:MLSU CSN:MLDP
Figure (optional)	 <p style="text-align: right; font-size: small;">Y.3531(20)_FI.5</p>
Pre-conditions (optional)	CSP:MLSP installs a virtual machine and builds an ML model for testing. CSP provides an ML framework or /platform tools for testing the ML model
Post-conditions (optional)	CSN:MLDP provides the ML dataset for learning and testing. CSP:MLSP stores the testing history of ML model for debugging ML. If CSN:MLMD fails to optimize the ML model, then CSN:MLMD requests testing ML data from CSN:MLDP
Derived requirements or recommendations	– Clause 8.5 items 2), 3), 4), 5), 6), 12)

I.6 Model monitoring to issue alerts of abnormal or unsuspected learning process

Title	Model monitoring to issue alerts of abnormal or unsuspected learning process
Description	<p>This use case describes the function of monitoring services in MLaaS. Abnormalities during learning may be caused by many aspects that can result in failure to learn, so the CSC:MLSU or CSN:MLMD should be able to recognize an abnormal state of learning by monitoring its systems. Abnormalities during the learning process can be detected through resource overload in the hardware, such as CPU and GPU, excessively abnormal prediction results and synchronization errors of parameters. The following are general steps for model testing in this use case.</p> <ol style="list-style-type: none"> 1) CSC:MLSU or CSN:MLMD requests job details for learning process. 2) CSP:MLSP reports the job details which shows current job statuses. NOTE – Job statuses include CPU or GPU utilization, memory usage and parameter update or synchronization status. 3) CSC:MLSU sends a request to stop the learning procedure to CSP:MLSP. <p>CSN:MLMD can also set an automatic stop when a learning abnormality is clearly detected. In that case, CSN:MLMD can set the values that are served by CSP:MLSP or customized by CSN:MLMD</p>
Role or sub-role	CSN:MLMD CSP:MLSP CSC:MLSU CSN:MLDP
Figure (optional)	
Pre-conditions (optional)	CSP:MLSP installs virtual machine and builds ML model for testing
Post-conditions (optional)	CSP:MLSP provides the interface for reporting the testing result to CSN:MLMD. CSN:MLDP provides the ML dataset for learning and testing. CSP:MLSP stores the testing history of ML model for debugging ML. If CSN:MLMD fails to optimize the ML model, then CSN:MLMD requests testing ML data to CSN:MLDP
Derived requirements or recommendations	<ul style="list-style-type: none"> – Clause 8.5 item 10) – Clause 8.6 items 1), 2), 3), 4), 5), 6), 7)

I.7 Model deployment and monitoring

Title	Model deployment and monitoring
Description	<p>This use case describes model deployment and monitoring procedures in the cloud computing environment. Once a model has been trained, then it is deployed in production. A model-monitoring process is also needed to maintain the performance of a deployed model. In a broad sense, model deployment and monitoring in this use case includes registering, managing and monitoring stages in the cloud computing environment.</p> <p>The following are general steps for model deployment and monitoring described in this use case.</p> <ol style="list-style-type: none"> 1) CSP:MLSP registers a trained model after a training stage. 2) CSP:MLSP provides information about the trained model used by CSP:MLSP to develop an application service. 3) CSP:MLSP provides an application service using the deployed model, and monitors its performance continuously. 4) CSP:MLSP requests model retraining or reengineering according to the measured performance of the model and predefined policy. 5) CSP:MLSP retrains or reengineers a model when asked. 6) CSP:MLSP notifies CSP:MLSP using the model for an application service when it is retrained or reengineered. 7) CSC:MLSU uses the application service using the deployed model
Role or sub-role	CSP:MLSP CSC:MLSU
Figure (optional)	<p>The figure consists of two flowcharts illustrating model monitoring and retraining/reengineering processes.</p> <p>Top Flowchart: Model monitoring and retraining a model (Y.3531(20)_F1.7a)</p> <ul style="list-style-type: none"> CSP: MLSP: <ul style="list-style-type: none"> Manage ML data (white box) feeds into Retrain/test ML model (orange box, step 2). Retrain/test ML model feeds into Deploy ML model (orange box, step 3). Retrain/test ML model feeds into Monitor ML model's performance (orange box, step 1). Deploy ML model feeds into Use updated ML model (orange box, step 4). Monitor ML model's performance feeds into Use updated ML model. Use updated ML model feeds into Results (white box). Results feeds into Use ML service (white box). Use ML service feeds back into Monitor ML model's performance. Monitor ML model's performance feeds back into Retrain/test ML model (Request for model retraining when needed). CSC: MLSU: <ul style="list-style-type: none"> Use ML service (white box) feeds into Use updated ML model (orange box, step 4). Use updated ML model feeds into Results (white box). Results feeds into Use ML service. <p>Bottom Flowchart: Model monitoring and reengineering a model (Y.3531(20)_F1.7b)</p> <ul style="list-style-type: none"> CSP: MLSP: <ul style="list-style-type: none"> Manage ML data (orange box, step 2) feeds into Retrain/test ML model (orange box, step 3). Retrain/test ML model feeds into Deploy ML model (orange box, step 4). Retrain/test ML model feeds into Monitor ML model's performance (orange box, step 1). Deploy ML model feeds into Use updated ML model (orange box, step 5). Monitor ML model's performance feeds into Use updated ML model. Use updated ML model feeds into Results (white box). Results feeds into Use ML service (white box). Use ML service feeds back into Monitor ML model's performance. Monitor ML model's performance feeds back into Retrain/test ML model (Request for model reengineering). CSC: MLSU: <ul style="list-style-type: none"> Use ML service (white box) feeds into Use updated ML model (orange box, step 5). Use updated ML model feeds into Results (white box). Results feeds into Use ML service.
Pre-conditions (optional)	CSP:MLSP provides a trained model to be deployed to develop an application service
Post-conditions (optional)	CSC:MLSU uses an ML application service provided by CSP:MLSP
Derived requirements and recommendations	– Clause 8.7 items 1), 2), 3), 4), 5), 6)

I.8 Automated machine learning in cloud computing

Title	Automated machine learning in cloud computing
Description	<p>This use case describes automated machine learning procedures in the cloud computing environment. Automated machine learning supports three main functionalities of algorithms, namely automated feature engineering, ML model search and hyperparameter optimization. To execute automated machine learning, CSC:MLSU just configures the learning task and input data. Then, automated feature engineering algorithms construct features of ML data for automatic ML model training. After feature construction, ML model search algorithms are implemented to find an ML model design by exploring the ML model catalogue. Finally, hyperparameter optimization algorithms tune hyperparameter values to maximize ML model performance.</p> <p>The following are general steps for the model deployment in this use case.</p> <ol style="list-style-type: none"> 1) CSC:MLSU sets the learning task and input data. 2) CSP:MLSP executes feature engineering algorithms with input data. 3) CSP:MLSP executes ML model search and hyperparameter optimization algorithms iteratively. <p>NOTE 1 – ML model search and hyperparameter optimization algorithms can be combined. NOTE 2 – Validation of an ML model is performed during step 3).</p> <ol style="list-style-type: none"> 4) CSP:MLSP tests the ML model output from step 3). <ol style="list-style-type: none"> a) If test performance is evaluated under target performance, then repeat step 3). b) Otherwise, export the trained ml model.
Role or sub-role	CSP:MLSP CSC:MLSU
Figure (optional)	<p style="text-align: right;">Y.3531(20)_F1.8</p>
Pre-conditions (optional)	
Post-conditions (optional)	
Derived requirements or recommendations	<ul style="list-style-type: none"> – Clause 8.5 items 12), 13), 14) – Clause 8.6 items 4), 5), 6)

Appendix II

Use case of MLaaS for application perspectives

(This appendix does not form an integral part of this Recommendation.)

The use cases in this appendix provide scenarios for operating ML applications using MLaaS and related functional requirements of MLaaS.

II.1 Object recognition model development in the cloud computing environment

Title	Object recognition model development in the cloud computing environment
Description	<p>An AI software engineer wants to concentrate on developing and testing ML models for object recognition. However, the size of the labelled dataset is not sufficient to develop and test the model. In the cloud computing environment, the engineer can access a validated ML dataset for object recognition from other engineers or companies. With the data from the cloud, engineers can build, train and manage their own recognition model by testing the performance of the model.</p> <p>In this use case, the engineer who is represented as CSN:MLMD can develop an ML model in cloud computing systems with validated data from the CSN:MLDP. The following steps show the process of developing an ML model.</p> <ol style="list-style-type: none">1) CSN:MLMD registers the ML model with CSP:MLSP.2) CSN:MLDP prepares an ML dataset with data labelling.<ol style="list-style-type: none">2-1) CSN:MLDP requests raw data from CSN:DP.2-2) CSN:MLDP labels the data with an object detection algorithm.NOTE 1 – A different detection algorithm can be adopted. An example of a detection algorithm is rectangle detection and face detection with a facial landmark.3) CSP:MLSP requests ML data with the information from the detection algorithm adopted.4) CSP:MLSP iteratively trains the ML model with the ML dataset until it is optimized.<ol style="list-style-type: none">a) 4-1) CSP:MLSP trains the ML model with the ML dataset with default parameter.b) 4-2) CSP:MLSP evaluates the trained ML model with a validation dataset.c) 4-3) CSP:MLSP changes the model parameter and repeats a) and b) until the model is optimized.NOTE 2 – The ML model parameter can be different among ML models. An example of an ML model parameter is the Euclidean distance between the labelled data and validation data.5) CSP:MLSP reports the performance evaluation to CSN:MLMD
Role or sub-role	CSN:MLMD CSN:DP CSN:MLDP CSP:MLSP

Title	Object recognition model development in the cloud computing environment
Figure (optional)	<p>The diagram illustrates the object recognition model development process in a cloud computing environment. It is divided into two main parts: CSN:MLDP (top) and CSP:MLSP (bottom).</p> <p>CSN:MLDP (top): This component receives raw data from CSN:DP. It performs object detection (with a detection algorithm*) and data labelling. The detected objects are labeled as 'Cat' and 'Burger'. A note specifies: "NOTE – Detection algorithm* = ex) Rectangle detection, and face detection with facial landmark".</p> <p>CSP:MLSP (bottom): This component receives ML data and adopted detection algorithms from CSN:MLDP. It performs training with ML data and ML models, and then performs performance evaluation with a validation set. The trained ML model is stored. A note specifies: "NOTE – Model parameter* = ex) Euclidean distance threshold between the labelled and validation data".</p> <p>CSN:MLMD (left): This component provides ML models to CSP:MLSP. It receives performance evaluation data from CSP:MLSP and reports it back to CSN:MLDP.</p> <p>Process Flow:</p> <ol style="list-style-type: none"> 1) ML model is provided from CSN:MLMD to CSP:MLSP. 2) Raw data is sent from CSN:DP to CSN:MLDP, which performs object detection and data labelling. 3) ML data and adopted detection algorithms are sent from CSN:MLDP to CSP:MLSP. 4) CSP:MLSP performs training with ML data and ML models, and then performs performance evaluation with a validation set. 5) Performance evaluation data is reported from CSP:MLSP to CSN:MLMD, which then reports it back to CSN:MLDP. <p>Y.3531(20)_F11.1</p>
Pre-conditions (optional)	CSN:MLDP searches data and requests data from CSN:DP
Post-conditions (optional)	CSP:MLSP stores the trained or optimized ML model to develop ML applications
Derived requirements and recommendations	<ul style="list-style-type: none"> – Clause 8.2 items 1), 3), 4) – Clause 8.4 items 1), 3), 4) – Clause 8.5 items 1), 2)

II.2 Traffic speed prediction and monitoring service

Title	Traffic speed prediction and monitoring service
Description	<p>Traffic speed information on city roads are gathered from sensors. Traffic speed on each road after a given period, e.g., 15 min, is predicted, and the predictions used to solve traffic congestion in the city. The following is a procedure of this use case.</p> <ol style="list-style-type: none"> 1) Raw data such as traffic speed on city roads are gathered by CSN:DP. 2) The gathered raw data are featured and pre-processed by CSP:MLSP, and the data used for training and validating a model. 3) Some machine learning models developed by CSN:MLMD are provided, and those models are used to train models by CSP:MLSP. 4) A prediction model is trained and validated using a training dataset. 5) The trained model is deployed and used to develop an application. 6) Data for prediction are fed into the trained model or application, and predictions are returned to the service user. The service user can use the predictions, e.g., predicted traffic speed on each road, to solve traffic congestion by controlling traffic signals
Role or sub-role	<p>CSN:DP CSN:MLMD CSP:MLSP CSU:MLSU</p>

Title	Traffic speed prediction and monitoring service
Figure (optional)	<p style="text-align: right; font-size: small;">Y.3531(20)_FIL.2</p>
Pre-conditions (optional)	<p>CSN:MLSP can transform raw data used for predicting traffic speed. The transformed data are fed into a trained prediction model.</p> <p>The sensors for traffic observation are installed on city roads or in each vehicle to gather speed and location data from traffic using the roads</p>
Derived requirements and recommendations	<ul style="list-style-type: none"> - Clause 8.1 items 1), 3), 5) - Clause 8.3 items 1), 3), 4) - Clause 8.4 items 2), 6) - Clause 8.5 items 1), 3), 6), 7), 8), 9), 10), 11) - Clause 8.7 items 1), 2), 3), 4)

II.3 Image recognition

Title	Image recognition
Description	<p>CSC:MLSU who is an application developer wants to provide image recognition in applications, e.g., animal recognition. In order to improve development efficiency, CSC:MLSU can just focus on user interface interactions, and use MLaaS to implement the image recognition function. The process would include the following steps.</p> <ol style="list-style-type: none"> 1) CSP:MLSP collects animal images with different formats from different CSP:MLDPs and stores those images. 2) CSP:MLSP does some pre-process work on the images like: image classification by species (e.g., cats, dogs, pigs); image conversion into the same format (e.g., convert jpg, jpeg or bmp to png); image ranking by resolution. 3) CSP:MLSP sends pre-processed images to CSN:MLMD to develop an ML model and uses multiple worker nodes to get training acceleration. 4) After training, CSN:MLMD registers an image recognition model with the CSP:MLSP. 5) CSP:MLSP recognizes images from those submitted using the model registered by the CSN:MLMD. 6) CSP:MLSP displays the results of image recognition to the CSC:MLSU. 7) CSC:MLSP provides accuracy, CPU usage and time cost of image recognition as feedback to CSP:MLMD
Role or sub-role	<p>CSC:MLSU CSP:MLSP CSN:MLMD CSN:MLDP</p>

Title	Image recognition
Figure (optional)	<pre> sequenceDiagram participant CSC as CSC: MLSU participant CSP as CSP: MLSP participant CSN1 as CSN: MLMD participant CSN2 as CSN: MLDP CSP->>CSN1: 1. Account opening CSN1-->>CSP: 3. Image recognition service selection CSP->>CSC: 4. Provide an image that needs to be recognized CSP->>CSN1: 5. An image needs to be recognized CSN1-->>CSP: 6. Image recognizing based on ML model CSN1-->>CSN2: 9. Record usage of image recognition service for billing CSN1-->>CSP: 7. Return image recognition result: "Cat" CSP-->>CSC: 8. Display image recognition result: "Cat" </pre> <p>The diagram shows a sequence of interactions between four entities: CSC: MLSU, CSP: MLSP, CSN: MLMD, and CSN: MLDP. The process starts with '1. Account opening' from CSP: MLSP to CSN: MLMD, followed by '3. Image recognition service selection' from CSN: MLMD to CSP: MLSP. Then, '4. Provide an image that needs to be recognized' is sent from CSP: MLSP to CSC: MLSU, which includes a small image of a cat. Next, '5. An image needs to be recognized' is sent from CSP: MLSP to CSN: MLMD, also including the cat image. CSN: MLMD then performs '6. Image recognizing based on ML model' and returns '7. Return image recognition result: "Cat"' to CSP: MLSP. Simultaneously, '9. Record usage of image recognition service for billing' is sent from CSN: MLMD to CSN: MLDP. Finally, CSP: MLSP sends '8. Display image recognition result: "Cat"' to CSC: MLSU. A reference 'Y.3531(20)_F11.3' is noted at the bottom right of the diagram area.</p>
Pre-conditions (optional)	
Post-conditions (optional)	
Derived requirements and recommendations	<ul style="list-style-type: none"> - Clause 8.1 items 1), 3), 5), 6) - Clause 8.3 items 1), 2), 4) - Clause 8.4 items 4), 5) - Clause 8.5 items 1), 2) - Clause 8.7 item 1)

II.4 Face recognition

Title	Face recognition
Description	<p>CSC:MLSU who is a door control system developer for a company needs to use face recognition for employee and visitor authentication. The most convenient way for the developer is to implement a face recognition function using MLaaS. The process would include the following steps.</p> <ol style="list-style-type: none"> 1) CSP:MLSU uses a camera to collect videos taken from different angles of employees. In order to detect live, videos with spoofed faces on a screen are also collected. 2) CSP:MLSU uploads those videos to CSP:MLSP and splits them into a real face training set, a fake face training set and testing set through the configuration reference point of CSP:MLSP. 3) CSP:MLSP marks the position of face and eyes in the training sets. 4) CSP:MLSP sends training sets to CSN:MLMD and configures hyperparameter values to start developing an ML model. During development, CSP:MLSP can obtain the learning status from CSN:MLMD. 5) After the learning process, CSN:MLSP validates the trained ML models with the testing set. If the validation results meet expectations, ML models will be registered and deployed. 6) When a visitor comes to the company, CSP:MLSU captures a video of the visitor. Then CSP:MLSP performs face recognition for the captured video using the model deployed. 7) The CSP:MLSP displays the result of face recognition to the CSC:MLSU and the visitor is authenticated based on the result

Title	Face recognition
Role or sub-role	CSC: MLSU CSP: MLSP
Figure (optional)	
Pre-conditions (optional)	
Post-conditions (optional)	
Derived requirements and recommendations	<ul style="list-style-type: none"> – Clause 8.2 items 1) – Clause 8.3 items 1), 4) – Clause 8.4 items 2), 6) – Clause 8.5 items 1), 2), 3), 4), 5), 7), 8), 11) – Clause 8.6 item 1) – Clause 8.7 item 1)

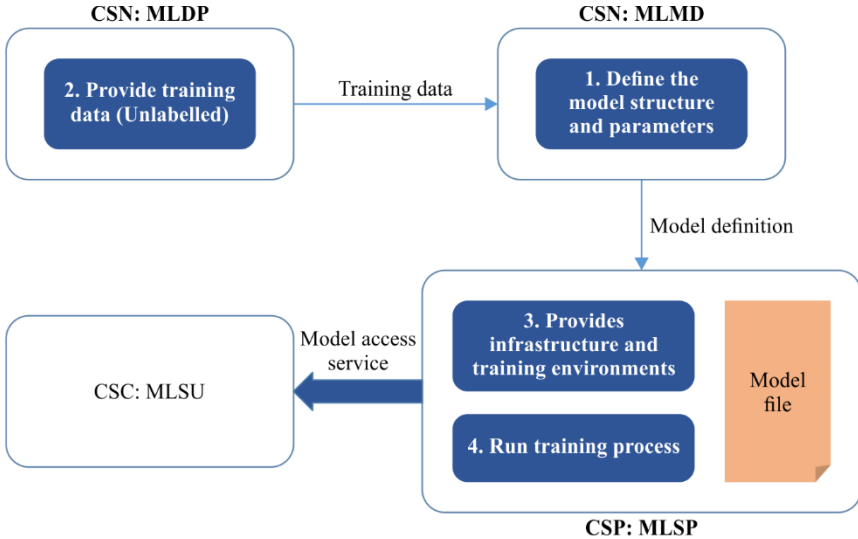
II.5 Image segmentation model development

Title	Image segmentation model development
Description	<p>Image segmentation is a fundamental task for high level vision tasks such as driving a vehicle and robot navigation. Recently developed deep neural networks can perform with a large amount of training data and powerful computation resources. Thus, it is a good choice for CSN:MLMD to train and deploy a model in cloud computing environments. This use case includes the following key steps.</p> <ol style="list-style-type: none"> 1) CSN:MLMD designs specific model parameters, such as network architecture and training loss function. 2) CSP:MLSP establishes the training data format and requests training data from CSN:MLDP. 3) CSN:MLMD requests CSP:MLSP to prepare resources with the necessary library installed (such as an ML library) for model training. 4) CSP:MLSP runs a training process for the model committed by CSN:MLMD and reports training status including training and validation errors. 5) After convergence, the trained model is registered and deployed in the cloud

Title	Image segmentation model development
Role or sub-role	CSN:MLMD CSN:MLDP CSP:MLSP
Figure (optional)	<pre> graph TD subgraph CSN_MLDP [CSN: MLDP] A[2. Provide training data (Labelled)] end subgraph CSN_MLMD [CSN: MLMD] B[1. Define the model structure and parameters] end subgraph CSP_MLSP [CSP: MLSP] C[3. Provides infrastructure and training environments] D[4. Run training process] E[Model file] end subgraph CSC_MLSU [CSC: MLSU] F[Model access service] end A -- Training data --> B B -- Model definition --> CSP_MLSP CSP_MLSP -- Model access service --> F </pre> <p style="text-align: right;">Y.3531(20)_F11.5</p>
Pre-conditions (optional)	
Derived requirements and recommendations	<ul style="list-style-type: none"> – Clause 8.1 items 1), 3), 4) – Clause 8.5 items 1), 2), 3), 4), 6), 9), 11), 12) – Clause 8.6 items 1), 3), 4) – Clause 8.7 item 1)

II.6 Generative adversarial model development

Title	Generative adversarial model development
Description	<p>Generative adversarial models can generate new images whose appearance is consistent with those in the training dataset, e.g., generate the image of an animal or a building. A simple generative model can be trained in an unsupervised manner, i.e., no labels are required. Neural network-based generative adversarial network (GAN) models are one effective method for image generation. However, training a GAN is usually tedious and requires lots of tuning skills. It would be convenient if the CSP can provide a training and deployment service for a CSN:MLMD. Such a use case can be fulfilled with the following steps.</p> <ol style="list-style-type: none"> 1) CSN:MLMD requests specific training data from CSN:MLDP according to the task. For instance, if the model is to generate animals, then the training data should only contain a large number of various animals. 2) CSN:MLMD establishes the network architectures of generative and discriminative modules with corresponding loss functions. 3) CSN:MLSP prepares computation resources and starts model training. A training process will stop if the convergence condition is satisfied or the maximum number of iterations has been reached. Retraining can be executed if necessary. 4) After convergence, the model access method is returned to CSN:MLMD
Role or sub-role	CSN:MLDP CSN:MLMD CSP:MLSP

Title	Generative adversarial model development
Figure (optional)	 <p style="text-align: right;">Y.3531(20)_F11.6</p>
Pre-conditions (optional)	
Derived requirements and recommendations	<ul style="list-style-type: none"> - Clause 8.1 items 1), 3), 4) - Clause 8.5 items 1), 4), 5), 6), 7), 9) - Clause 8.6 items 5), 6) - Clause 8.7 items 1), 5)

Bibliography

- [b-ITU-T H.752] Recommendation ITU-T H.752 (2015), *Multimedia content provisioning interface for IPTV services*.
- [b-ITU-T X.1601] Recommendation ITU-T X.1601 (2015), *Security framework for cloud computing*.
- [b-ITU-T X.1631] Recommendation ITU-T X.1631 (2015), *Information technology – Security techniques – Code of practice for information security controls based on ISO/IEC 27002 for cloud services*.
- [b-ITU-T Y.3172] Recommendation ITU-T Y.3172 (2019), *Architectural framework for machine learning in future networks including IMT-2020*.
- [b-ITU-T Y.3500] Recommendation ITU-T Y.3500 (2014), *Information technology – Cloud computing – Overview and vocabulary*.
- [b-ETSI GR ENI 004] ETSI GR ENI 004 V2.1.1 (2019), *Experiential networked intelligence (ENI); Terminology for main concepts in ENI*.
- [b-Han] Han, J., Kamber, M., Pei, J.. (2012), *Data mining: Concepts and techniques*, third edition., Burlington. MA: Morgan Kaufmann. 703 pp.
- [b-He] He, X., Zhao, K., Chu, X. (2019), *AutoML: A Survey of the state-of-the-art*, arXiv preprint arXiv:1908.00709v5.

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