D3.1

Enhancing deep learning / reinforcement learning

WORKPACKAGE	WP3	PROGRAMME IDENTIFIER	H2020-ICT-2020-1
DOCUMENT	D3.1	GRANT AGREEMENT ID	957246
REVISION	V1.0	START DATE OF THE PROJECT	01/10/2020
DELIVERY DATE	30/06/2021	DURATION	3 YEARS

DISCLAIMER

ACKNOWLEDGEMENT

This document is a deliverable of IoT-NGIN project. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 957246.

The opinions expressed in this document reflect only the author's view and in no way reflect the European Commission's opinions. The European Commission is not responsible for any use that may be made of the information it contains.

PROJECT ACRONYM	IoT-NGIN
PROJECT TITLE	Next Generation IoT as part of Next Generation Internet
CALL ID	H2020-ICT-2020-1
CALL NAME	Information and Communication Technologies
TOPIC	ICT-56-2020 - Next Generation Internet of Things
TYPE OF ACTION	Research and Innovation Action
COORDINATOR	Capgemini Technology Services (CAP)
PRINCIPAL CONTRACTORS	Atos Spain S.A. (ATOS), ERICSSON GmbH (EDD), ABB Oy (ABB), INTRASOFT International S.A. (INTRA), Engineering-Ingegneria Informatica SPA (ENG), Bosch Sistemas de Frenado S.L.U. (BOSCH), ASM Terni SpA (ASM), Forum Virium Helsinki (FVH), Optimum Technologies Pilroforikis S.A. (OPT), eBOS Technologies Ltd (EBOS), Privanova SAS (PRI), Synelixis Solutions S.A. (SYN), CUMUCORE Oy (CMC), Emotion s.r.l. (EMOT), AALTO-Korkeakoulusaatio (AALTO), i2CAT Foundation (I2CAT), Rheinisch-Westfälische Technische Hochschule Aachen (RWTH), Sorbonne Université (SU)
WORKPACKAGE	WP3
DELIVERABLE TYPE	REPORT
DISSEMINATION LEVEL	PUBLIC
DELIVERABLE STATE	**FINAL**
CONTRACTUAL DATE OF DELIVERY	30/06/2021
ACTUAL DATE OF DELIVERY	30/06/2021
DOCUMENT TITLE	Enhancing deep learning / reinforcement learning
AUTHOR(S)	Adrian Arroyo (ATOS), Terpsi Velivassaki (SYN), Hervé Bardisbanian (CAP), Artemis Voulkidis (SYN), Stavroula Borou (SYN), Daniel Calvo (ATOS)
REVIEWER(S)	Dimitrios Skias (INTRA), Terpsi Velivassaki (SYN)
ABSTRACT	SEE EXECUTIVE SUMMARY
HISTORY	SEE DOCUMENT HISTORY
KEYWORDS	Artificial Intelligence, Machine Learning, Machine Learning as a Service, IoT, Big Data, Deep Learning, Reinforcement Learning, Federated Learning

Document History

Version	Date	Contributor(s)	Description
V0.1	02/12/2020	ATOS	Toc and first draft
V0.1	03/01/2021	ATOS	Update on Use Case requirements
V0.1	03/02/2021	SYN	Federated Learning State-of-the-art update
V0.1.1	15/03/2021	ATOS	MLaaS platform updates
V0.1.2	23/03/2021	ATOS	Use Cases UML Diagrams
V0.2	10/05/2021	ATOS/SYN/CAP	UML diagrams update: Use Cases, Platform Logical View and sequence diagrams
V0.3	26/05/2021	ATOS	Platform logical view description, Deep Learning / Reinforcement Learning techniques state-of-the-art
V0.4	07/06/2021	ATOS/Living Labs	Living Labs Use Case data and AI techniques requirements.
V0.5	16/06/2021	ATOS	BDV reference model alignment. Introduction, ML techniques overview, conclusions.
V0.6	18/06/2021	CAP	MLaaS architecture framework details
V0.6.1	29/06/2021	SYN	Peer review comments
V0.6.2	29/06/2021	INTRA	Peer review comments
V1.0	30/06/2021	ATOS	Final version & quality check

Table of Contents

List of Figures

List of Tables

List of Acronyms and Abbreviations

AGV	Automated Guided Vehicle
AI	Artificial Intelligence
API	Application Programming Interface
AR	Augmented Reality
BDVA	Big Data Value Association
CD	Continuous Deployment
CI	Continuous Integration
CNN	Convolutional Neural Network
CSV	Comma-separated values
IDS	International Data Spaces
IoT	Internet of Things
DL	Deep Learning
DOA	Description of Action
DP	Differential Privacy
DSML	Data Science and Machine Learning Platforms
ETL	Extract, Transform, Load
ETP4HPC	European Technology Platform for High Performance Computing
EOSC	European Open Science Cloud
EV	Electrical Vehicle
FATE	Federated AI Technology Enabler
FL	Federated Learning
GCP	Google Cloud Platform
GDPR	Global Data Protection Regulation
GMM	Gaussian Mixture Models
GNSS	Global Navigation Satellite System
GPU	Graphical Processing Unit
GUI	Graphical User Interface
HCA	Hierarchical Clustering Analysis

HPC	High Performance Computing
HPDA	High Performance Data Analytics
HTTP	Hypertext Transfer Protocol,
HTTPS	Hypertext Transfer Protocol Secure
IoT	Internet of Things
JPEG	Joint Photographic Experts Group
JSON	JavaScript Object Notation
LL	Living Labs
MDP	Markov Decision Process
ML	Machine Learning
MLaaS	Machine Learning as a Service
MQTT	Message Queue Telemetry Transport
OPC UA	Open Productivity Collaboration Unified Architecture
PCA	Principal Component Analysis
PMU	Phasor measurement units
PPDL	Privacy-Preserving Deep Learning
RFID	Radio Frequency Identification
RL	Reinforcement Learning
RNN	Recurrent Neural Network
SDK	Software Development Kit
SMC	Secure Multiparty Computation
SME	Small and medium-sized enterprise
SRIA	Strategic Research and Innovation Agenda
SVM	Support Vector Machine
TCP	Transmission Control Protocol
TIFF	Tagged Image File Format
TL	Transfer Learning
TPU	Tensor Processing Unit
UC	Use-case
UML	Unified Modelling Language

UWB	Ultra-Wide Band
VLP	Visible Light Positioning
WP	Work Package

Executive Summary

This document constitutes Deliverable "D3.1: Enhancing deep learning / reinforcement learning" of the European H2020-ICT-2018-20 project "IoT-NGIN: Next Generation IoT as part of Next Generation Internet". D3.1 is an output of Work Package 3, entitled "Enhancing IoT Intelligence", which reports the activities of all the tasks of the Work Package until the ninth month of the project's lifetime. More concretely, the tasks of the work package are:

1. T3.1 (M1-M27): Big Data and Machine learning framework architecture.
2. T3.2 (M3-M27): Deep learning/reinforcement learning techniques to enhance training processes.
3. T3.3 (M3-M28): Confidentiality-preserving federated ML models.
4. T3.4 (M6-M30): Machine Learning model sharing.

The scope of this deliverable is: (i) to detail the Big Data and ML framework that enables MLaaS and supports decentralised / federated ML, and (ii) to study different deep learning and/or reinforcement learning techniques, including self-learning techniques and supervised/unsupervised machine learning procedures applied to IoT devices to select those that better fit the IoT-NGIN scenarios and enhance the ML models' performance.

Main results/findings

- A common terminology for Artificial intelligence related elements.
- A study of Artificial Intelligence needs of IoT-NGIN Living Labs.
- A design of a Machine Learning as a Service platform that includes architecture, actors and use cases; and which references to BDVA SRIA4.0.
- The application of the federated learning paradigm over the platform and the Living Labs.
- A study of machine learning techniques, including deep learning, reinforcement learning and self learning, and a mapping of those techniques to IoT-NGIN use cases.

1 Introduction

H2020 IoT-NGIN promotes the enhancement of the Internet of Things (IoT) technology from several points of view towards the next generation of IoT through scalability, openness and security, while supporting monetization. In order to achieve this, the project pushes the boundaries of IoT communication, federation, intelligence, security and privacy via concrete work packages (WP) the results of which are applied over four main application areas: Smart City, Smart Agriculture, Industry 4.0 and Smart Energy.

Concretely, WP3, entitled "Enhancing IoT Intelligence", provides the innovation with respect to Artificial Intelligence (AI) and Big Data analytics, where the partners of the WP will go beyond the state-of-the-art of the Machine Learning (ML) techniques applied to the IoT. More specifically, the main objectives of the work package are shown in Figure 1.

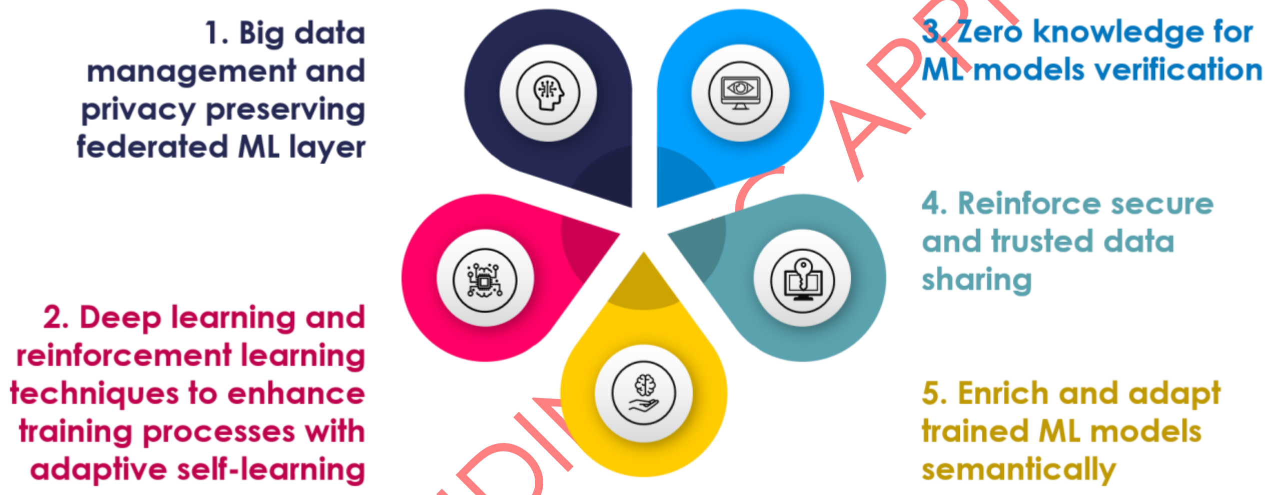

1. Big data management and privacy preserving federated ML layer
2. Deep learning and reinforcement learning techniques to enhance training processes with adaptive self-learning
3. Zero knowledge for ML models verification
4. Reinforce secure and trusted data sharing
5. Enrich and adapt trained ML models semantically

Figure 1- IoT NGIN WP3 objectives to enhance IoT intelligence

Considering this, Deliverable 3.1: "*Enhancing Deep learning / reinforcement learning*" reports the first activities that have been carried out in order to fill these objectives. Mainly, the platform, denominated as IoT-NGIN Machine Learning as a Service (MLaaS), that will allow the application of Artificial Intelligence techniques over IoT is explained through dedicated sections, and details with respect to its architecture, the management of data according to BDVA SRIA4.0, and the privacy preserving federated ML layer are given. In addition, the machine learning techniques, including deep learning and reinforcement learning, that will be implemented over the upcoming months of the project are reviewed.

1.1 Intended Audience

Since Deliverable 3.1 reports upon the application of Artificial Intelligence in the IoT, it offers an innovative point of view for developers and infrastructure managers who need to provide a higher level of intelligence in their IoT deployments. Moreover, data scientists and AI specialists will benefit from the results of the Deliverable, where details on how to build, deploy and maintain AI applications can be acquired.

The document will be also useful for project partners and the participants in IoT NGIN Open Calls since they will be able to get information about the functionalities and services to be implemented by IoT NGIN Big data and AI platform.

Finally, the whole European AI and IoT communities will be potential readers of the present deliverable due to its public nature.

1.2 Relations to other activities

In Figure 2 the work packages of IoT-NGIN are presented via a set of layers, where it can be estimated that the Big Data Analytics and ML layer influcences many other activities of the project.

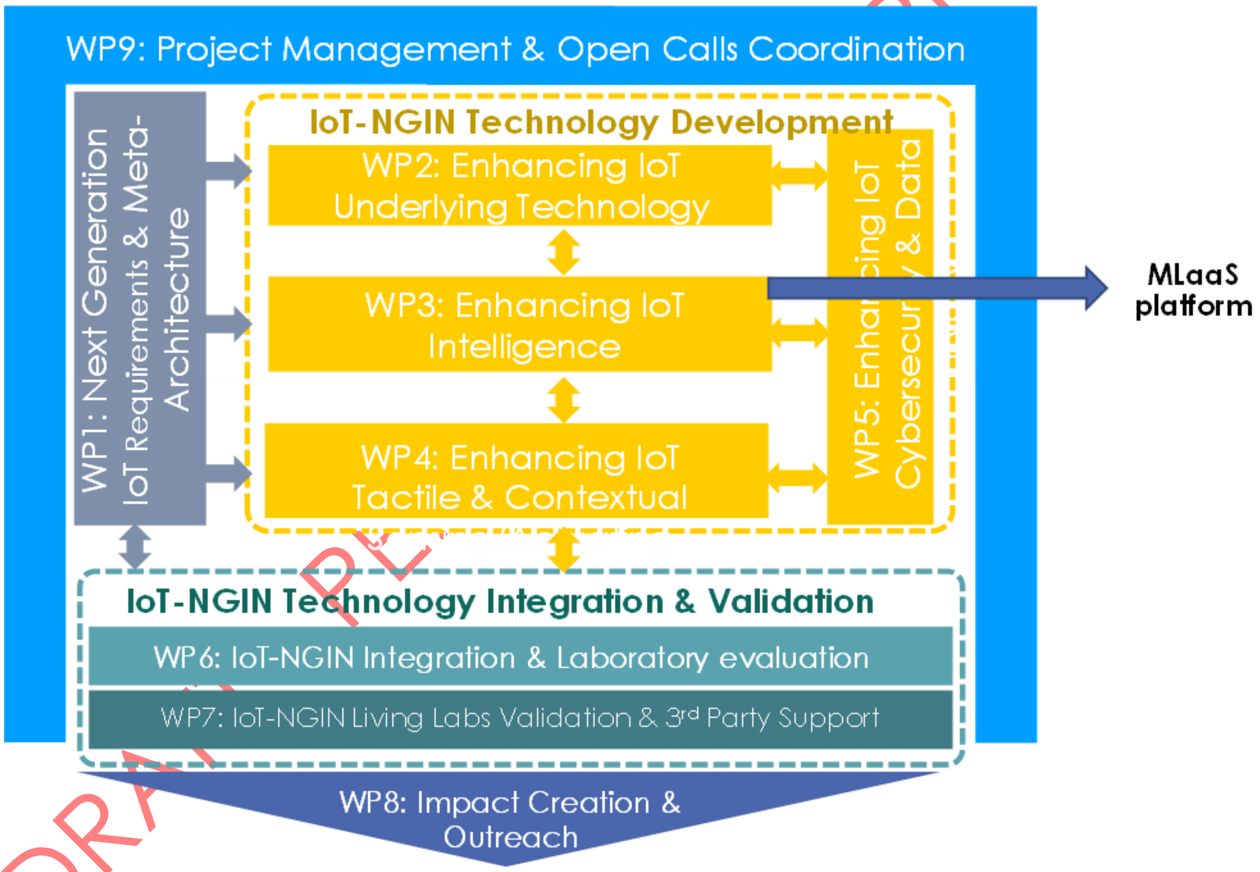

WP9: Project Management & Open Calls Coordination
WP1: Next Generation IoT Requirements & Meta-Architecture
IoT-NGIN Technology Development
WP2: Enhancing IoT Underlying Technology
WP3: Enhancing IoT Intelligence
WP4: Enhancing IoT Tactile & Contextual
WP5: Enhancing IoT Cybersecurity & Data
MLaaS platform
IoT-NGIN Technology Integration & Validation
WP6: IoT-NGIN Integration & Laboratory evaluation
WP7: IoT-NGIN Living Labs Validation & 3rd Party Support
WP8: Impact Creation & Outreach

Figure 2: Work Packages structure

Considering this, D3.1 and, generally, WP3, are related to the work packages and tasks described in Table 1.

Table 1: Relation of WP3 activities to other WPs and tasks

WP	Relation to WP3 and D3.1
WP1	Artificial Intelligence and Big Data analytics plays an important role on the next generation of IoT. In this regard, this role shall be represented in the meta-architecture resulting by the activities in WP1
WP2	The micro-services framework developed within WP2 shall be interconnected with the WP3 framework and its deployments. In addition, WP3 results, and more concretely trained AI models, are related to Task 2.4, where the Unikernels may support their deployment in an innovative environment
WP4	WP4 may need usage of several AI techniques to provide some of its functionalities. Specifically, Task 4.2 may benefit from the training of the AI models in charge of recognizing in real-time IoT devices
WP5	As the federation of the AI models training over several nodes may suffer from different cybersecurity attacks, WP3 clearly benefits from the mitigation of Federated Learning attacks in Task 5.1 and from the verification of AI models integrity in Task 5.2. In addition, the WP3's platform includes interaction with the Digital Twins, so results from Task 5.5 must be considered for a proper integration.
WP6	As the WP3's platform provides several components, the integration with rest of the project's technology and frameworks is clearly very important and will be continuously reviewed as the activities of WP3 iterate over their results. The WP3 platform will be exploited for application development, involving ML services. In addition, the infrastructure that enables CI/CD will provide the tools to develop the components of the WP in a safe a productive manner
WP7	WP3's framework will be used in several Living Labs to train, deploy and maintain their AI models, as it will be shown throughout the deliverable. In addition, WP3's platform will support 3rd parties by offering a specific set of functionalities

1.3 Document overview

The present deliverable is divided into seven chapters:

- **Chapter 1** introduces the **motivation and general objectives** for the document, its intended audience, relation to other project tasks and structure.
- **Chapter 2** proposes a **common terminology and definions** for all the components of the Artificial Intelligence stack and the involved stakeholders. It will guarantee the

homogeneity and consistency of the content of the document and the corresponding technical developments addressed within the scope of WP3.

- In **Chapter 3**, an exhaustive analysis of the pre-conditions and requirements imposed by the **Living Labs' use-cases** is done, extending and introducing specific details that will serve to guide the development of IoT-NGIN AI components.
- The main goal of **Chapter 4 is to provide a detailed specification of the functionalities to be implemented by IoT-NGIN MLaaS platform**. It is based on the 4+1 architectural view model [1] and covers the identification of actors interacting with the platform, modelling the functionalities through use cases, their translation to sequence diagrams and the composition of the platform's architecture from a logical point of view. In addition, the analysis of the state-of-the-art for available technologies and scientific trends is performed to ensure that the project is built considering the most advanced technical baselines. The **alignment with BDVA SRIA**[1] is also done, demonstrating compliance with the reference model.
- **Chapter 5 focuses on the different techniques** that will be implemented within project covering aspects like online learning and federated learning. A detailed vision of the current state-of-the-art and the improvement that will be brought by IoT-NGIN is included.
- The **conclusions and next steps** are explained in **Chapter 6**.

[1] https://www.bdva.eu/SRIA.

2 Artificial Intelligence Terminology in IoT-NGIN

As a result of the European & Korean joint DECENTER[2] project, a common terminology was identified to define an Artificial Intelligence application architecture. It is used also in IoT-NGIN to have a common ground for all the project partners.

2.1 Solution building blocks of Artificial Intelligence

Table 2 describes the terminology adopted to refer to each of the main building blocks that result from applying AI, going from the lowest level of complexity (a trained AI model) to the most complex one (AI application service).

Table 2: Solution building blocks of AI and their description

Solution building block type	Description
AI Model	A trained model (either a deep neural network or another Machine Learning model).
AI Method	An entity serving the inference of a trained model (runs the model on the ML engine). Refers to the actual method of the programming language that executes the ML engine to serve/offer the model.
AI Service	An entity serving a certain AI functionality (model serving layer) based on an AI Method. Refers to a software component that offers an AI functionality through an AI Method and other related methods.
AI Application	An entity invoking AI Methods to provide AI functionalities to the end-user through a GUI on a responsive and/or mobile app.
AI Application Service	Set of AI functionalities closely related comprising a software application or product function and delivered as a service to an end-user.

2.2 Functionalities related to an AI Application service

Table 3 provides a description for each type of service that an AI application service could provide, although it may be extrapolated for the rest of the building blocks described in the previous section.

[2] https://www.decenter-project.eu/

Table 3: Types of services provided by AI applications

AI application service kind	Description
Sensing	Data collection, monitoring
Analysis	Advanced analytics using AI, also known as perception in AI systems, may incorporate attention, data/sensor, image segmentation, object classification, object localization, object detection, scene classification, scene interpretation, etc.
Decision	Knowledge representation and reasoning, search / optimization, planning / scheduling, behaviour selection (reactive planning). Prescriptive machine learning, e.g. reinforcement learning
Action	Behaviour (decision) enaction. Control of the course of action or sequence of actuation (e.g. motor) commands.
Effecting	Execute a command (e.g. motor). Also means "actuation," to activate, or to put into motion; to animate.

Figure 3 is representing the described functionalities of an AI application service in a graph, where they are also linked to each other to define a higher level of complexity, like *Cognition*, *Perception* or *Inter-action*. With respect to the original figure, defined in DECENTER EU project, the main AI features that the IoT-NGIN project will implement and support have been added using bubbles.

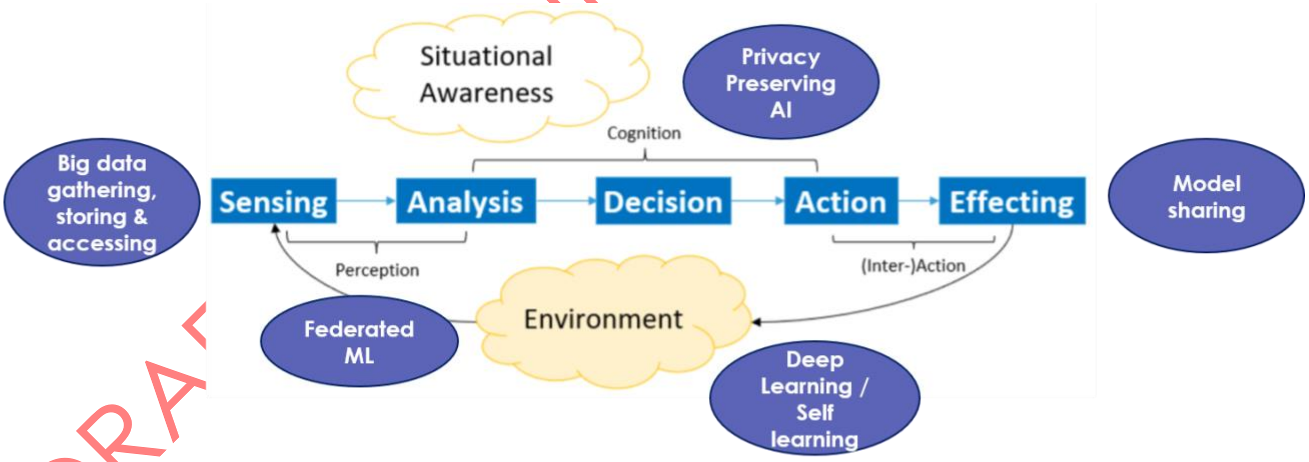


Figure 3: Adaptive behaviour control loop of an intelligent (adaptive) system, from DECENTER EU project, with IoT-NGIN main AI features.

3 Artificial Intelligence and Big Data preconditions from IoT-NGIN Living Labs

Throughout this section, the initial needs of IoT-NGIN use cases regarding Artificial Intelligence and Big Data are collected. The scope is to gather a set of requirements and information from the use cases so that the set of services that will be offered by the platform supports these requirements as best as possible, considering as well the objectives of the MLaaS platform defined in IoT-NGIN's Description of Action. Obviously, at the stage of the project in which this deliverable is released, the descriptions of the Living Labs are still to be refined and extended, so there could be information to be defined or that may vary in future iterations. For a more general overview of the requirements from the Living Labs for the whole IoT-NGIN project, the reader must refer to IoT-NGIN's Deliverable 1.1 [2].

The AI requirements from the use cases are analysed from four perspectives:

1. **Preconditions per application:** collects a list of conditions and objectives from each Living Lab use case (application) where the relevant AI needs are highlighted.
2. **Mapping to IoT-NGIN MLaaS tools**: one table per Living Lab which maps the highlighted AI needs from the previous section to the main components or services of the MLaaS platform to be built in WP3.
3. **Use Case data analysis**: these tables collect information regarding the data sources that are part of the Living Labs' use cases such as format, availability, communication protocol, etc. In addition, it collects information about the IoT-Devices which will be integrated into the use case.
4. **Use Case AI techniques analysis**: These tables collect information relevant to the AI models that are expected to be produced from the use case data, and that will participate in a successful deployment of the use case. They include attributes like the type of outcome (classification, regression), training techniques to be used (federated, online, ...), inference approach (real-time, on batches, ...), etc. Note that the information from these tables is under development; thus, some information might change or is still to be defined in future iterations.

The Living Labs and use cases of IoT-NGIN project are divided as represented in Table 4.

Table 4: IoT-NGIN Living Labs and Use Cases

Living Lab	Use Case
Human-Centred Twin Smart Cities	Traffic Flow Prediction & Parking Prediction
	Crowd management
	Co-commuting solutions based on social networks
Smart Agriculture	Crop diseases prediction. Smart irrigation and precision aerial spraying
	Sensor aided crop harvesting
Industry 4.0	Human-centred safety in a self-aware indoor factory environment

Smart Energy	Human-centred Augmented Reality assisted build-to-order assembly Digital powertrain and condition monitoring Move from Reacting to Acting in Smart Grid Monitoring & Control Driver-friendly dispatchable EV charging

3.1 Human-Centred Twin Smart Cities Living Lab

In this chapter, an analysis of the Artificial Intelligence needs from the three use cases under the Twin Smart Cities Living Lab is provided; including preconditions of each use case, mapping of those preconditions to MLaaS tools, a use case data analysis and a use case AI techniques analysis.

3.1.1 Preconditions per application

1. Traffic Flow Prediction & Parking Prediction
   a. In order to model and **train distributed AI models on traffic flow and parking prediction**, weather data (and how that affects road & traffic conditions) and road data (number of cars, velocity, fluctuation) will be used. The predictive model will also consume historical data and public transportation information.
   b. Traffic and parking prediction **ML models will be federated** at the edge cloud.
   c. Ideally, the driver could, with small behavioural changes, avoid traffic and know where available parking is located, without needing to look for it aimlessly.
   d. With the data collected the ML process will identify traffic patterns. This information will be provided in the least intrusive way to a willing user to avoid invasive notifications.
   e. 5G communications will be used for interconnecting sensors, gather IoT data in real-time and store it at the edge, and **run smart AI-based simulations to perform what-if analysis when transport is interrupted**, e.g.: due to extreme weather, man-made or technical hazards.
2. Crowd management
   a. Demonstrate the use of open data, user data and IoT data on traffic fluency through cameras and radars installed at the bottleneck intersections for **crowd steering based on the application of AI**.
   b. Demonstrate the use of AI on advanced crowd prediction and movement control.
3. Co-commuting solutions based on social networks
   a. Combine IoT data with virtual citizen-generated IoT data from social networks to demonstrate the use of advanced **AI in the provision of co-commuting solutions at the neighbourhood level and cross-border**.

b. UC will take avail of the Urban Open Platform and Lab (UOP.Lab) developed in the Finest Twins project[3].

3.1.2 Mapping to IoT-NGIN MLaaS tools

The preconditions collected in the previous section are mapped to IoT-NGIN's MLaaS platform in Table 5.

Table 5: Mapping of Human-Centred Twin Smart Cities Living Lab to IoT-NGIN MLaaS tools

Big Data and ML framework	**Deep Learning**	**Reinforcement learning**	**Federated learning**	**ML models sharing**
Open data, user data, IoT data, images from cameras and radars. Cross-border data models.	Predictive algorithms for traffic flow, parking, crowd management.	N/A	Prediction ML models will be federated at edge-cloud.	Transfer of models for SMEs.

For the Twin Smart Cities Living Lab two *AI Applications* will be developed, namely:

- "Traffic Flow Prediction & Parking Prediction", which contains two AI Services: "Traffic Flow prediction "and "Parking prediction".
- "Crowd Management", which contains two AI services: "Crowd size prediction" and "Route alternatives prediction".

The third application of this living lab, "Co-commuting solutions based on social networks" has no AI requirements at this stage of the project, so it is not included in the following analysis. The analysis of the data and the AI techniques that will be used to build these AI services is described in the following sections.

3.1.3 Use Case Data Analysis

The following tables describe, from a generic point of view, the features of the data produced in the Human-Centred Twin Smart Cities Living Lab that is relevant for the creation of AI Applications.

[3] http://www.finesttwins.eu/en.

Table 6: Data analysis for "Traffic Flow Prediction & Parking Prediction" use case

Use Case		**Human-Centred Twin Smart Cities Living Lab**	
AI Application		**Traffic Flow Prediction & Parking Prediction**	
AI Service		**Traffic Flow Prediction**	**Parking Prediction**
Data Sources	**IoT Devices**	Multispectral & Visual Cameras Radars Weather stations Sensors on Robo-buses & City Streets (road data)	
	Data Type(s)	JSON and GeoJSON	
	Data size	To be defined	
	Data communication protocols	HTTP MQTT	
	Data availability	Request only	
Potential Bias		N/A	
Heuristics/Assumptions		The number of cars on the roads correlates directly with environmental factors and time (weather, events, weekday, etc.)	
Data privacy level		API is not public access	

Table 7: Data analysis for "Crowd Management" use case

Use Case		**Human-Centred Twin Smart Cities Living Lab**	
AI Application		**Crowd Management**	
AI Service		**Crowd size prediction**	**Route Alternatives Prediction**
Data Sources	**IoT Devices**	Multispectral & Visual Cameras Radars Noise sensors	
	Data Type(s)	JSON XML	
	Data size	To be defined	
	Data communication protocols	HTTP MQTT	
	Data availability	On request	
Potential Bias		Passengers with luggage are very different from passengers without luggage. Distinction important to be made.	
Heuristics/Assumptions		Diverting a part of the incoming passenger flow helps directly improve bottleneck situations.	
Data privacy level		Private	

3.1.4 Use Case AI Techniques Analysis

The following tables describe, from a generic point of view, the techniques of AI that will be used in the Human-Centred Twin Smart Cities Living Lab for the creation of the relevant AI Applications.

Table 8: AI techniques analysis for " Traffic Flow Prediction & Parking Prediction " use case

Use Case	**Human-Centred Twin Smart Cities Living Lab**	
AI Application	**Traffic Flow Prediction & Parking Prediction**	
AI Service	**Traffic Flow Prediction**	**Parking Prediction**
Outcome type	Prediction Multi-class classification (car type, size, direction) Multidimensional regression (car type and count linearly related to parking spot amounts and traffic jams)	
Metrics: Acceptance / Failure Criteria	Minimum accuracy to be set	
Training strategy	Unsupervised ML/DL Auto-labelling	
Enhanced AI techniques	N/A	
Inference approach	To be set	
Serving approach	To be set	

Table 9: AI techniques analysis for " Crowd Management " use case

Use Case	**Human-Centred Twin Smart Cities Living Lab**	
AI Application	**Crowd Management**	
AI Service	**Crowd size prediction**	**Route Alternatives Prediction**
Outcome type	Multi-class classification Multidimensional regression	
Metrics: Acceptance / Failure Criteria	To be set	
Training strategy	Unsupervised ML/DL	
Enhanced AI techniques	N/A	
Inference approach	To be set	
Serving approach	To be set	

3.2 Smart Agriculture IoT Living Lab

In this chapter, an analysis of the Artificial Intelligence needs from the three use cases under the Smart Agriculture Living Lab is provided; including preconditions of each use case, mapping of those preconditions to MLaaS tools, use case data analysis and use case AI techniques analysis.

3.2.1 Preconditions per application

1. The preconditions identified as relevant for ML processes for the "Crop diseases prediction, Smart irrigation and precision aerial spraying" use case are identified as follows:
   a. **Crop diseases prediction is based on images and real-time video analysis** of the crop and the leaves captured from visual and **multi-spectral cameras** located on semi-autonomous **drones** flying over the orchard.
   b. Crop diseases' prediction also considers measurements acquired via SYN SynField[4] precision agriculture IoT nodes, integrating a variety of sensor modules.
   c. **Real-time video analysis** takes place either locally (on the drone), based on already trained ML models, or remotely (at the edge) based on **federated ML**.
   d. The drones will be able to **dynamically modify their trajectory** to introduce optimal, precision aerial spraying only in areas of interest.
   e. The agronomists can access the crop disease predictions and suggest irrigation and spraying rules.
   f. IoT-NGIN will leverage existing datasets available for Living Lab experimentation, to the extent possible. Alternatively, publicly available datasets will be used.
2. In addition, the preconditions assumed for the "Sensor aided crop harvesting" use case, in relation to ML processes, are the following:
   a. **AGLV serving as carrier machines** of crates, assisting the harvesting process, capable of calculating and following an appropriate trajectory from the harvesting location to the loading points.
   b. **AGLV** will be able to **locate and avoid workers** (for safety reasons) **and trees** (for operating reasons).
   c. The crates identification at the **loading points** will be based on **RFID readers** located there, as well as **RFID tags attached to the crates.**

3.2.2 Mapping to IoT-NGIN MLaaS tools

The Smart Agriculture Living Lab use cases will exploit various ML tools developed within IoT-NGIN. Specifically, the Big Data and ML Framework will be exploited for data acquisition and analysis, referring to the sensor measurements acquired via SynField devices, the images and videos captured from the drones, as well as the RFID readings of the crates. Moreover, deep learning techniques will be used for crop diseases' prediction, as well as for obstacle avoidance. Federated learning techniques will be employed for crop disease prediction, exploiting experience gained in other fields, as well as for near-real-time video analysis of drones' captures, in order to effectively calculate clear trajectories for drones' movement.

[4] https://www.synfield.gr/about/.

The exploitation of ML techniques in the Smart Agriculture Living Lab is summarized in Table 10.

Table 10: Mapping of Smart Agriculture IoT Living Lab to IoT-NGIN MLaaS tools

Big Data and ML framework	**Deep Learning**	**Reinforcement learning**	**Federated learning**	**ML models sharing**
Micro-climate measurements Images and real-time video analysis from drones RFID readings	Crop diseases prediction Obstacle avoidance.	N/A	Crop diseases prediction	During UCs' development/ validation

3.2.3 Use Case Data Analysis

For the Smart Agriculture IoT Living Lab, two *AI Applications* will be developed, namely the "Crop diseases prediction. Smart irrigation and precision aerial spraying" and the "Sensor aided crop harvesting", each one serving one of the LL's trials. The first one will use the "Crop diseases prediction" *AI Service*, while the latter will use the "V-SLAM" and "Obstacle Avoidance using Deep Learning" *AI Services*.

The datasets used for each *AI Service* of the Living Lab are described in Table 11 and Table 12.

Table 11: Data analysis for "Crop diseases prediction, smart irrigation and precision aerial spraying" use case

Use Case		**Smart Agriculture IoT Living Lab**
AI Application Service		**Crop diseases prediction. Smart irrigation and precision aerial spraying**
AI Service		**Crop diseases prediction**
Data Sources	**IoT Devices**	SynField devices measuring micro-climate data (e.g. air temperature, air humidity, wind direction, wind speed, rain volume, rain intensity), and soil and crop-related data (leaf wetness, soil type, soil temperature, soil humidity, soil conductivity) Drones (multi-spectral or RGB cameras)
	Data Type(s)	JSON Image (JPEG, TIFF) GNSS measurements
	Data size	N/A
	Data communication protocols	SynField API, HTTPS/REST
	Data availability	Streaming
Potential Bias		Class imbalance, more data of one class for drone Images
Heuristics/Assumptions		Data are strongly correlated with the time of the collection
Data privacy level		Private

Table 12: Data analysis for "sensor aided crop harvesting" use case

Use Case		**Smart Agriculture IoT Living Lab**	
AI Application Service		**Sensor aided crop harvesting**	
AI Service		**V-SLAM**	**Obstacle Avoidance using Deep Learning**
Data Sources	**IoT Devices**	Sensors on mobile robots	
	Data Type(s)	JSON GNSS, Images	
	Data size	N/A	
	Data communication protocols	HTTPS, pub/sub or REST	
	Data availability	Streaming	
Potential Bias		Homogeneity of the collected data	
Heuristics/Assumptions		Most data will be collected for autonomous navigation	
Data privacy level		Private	

3.2.4 Use Case AI Techniques Analysis

Diving deeper into the *AI services* of the Smart Agriculture LL from an ML perspective, the services are further analysed in the following tables. For each *AI Service,* several properties are highlighted, such as the expected outcome type, assessment criteria, the preferred training strategy, potential techniques for facilitating the ML process, the inference and the serving approach.

Table 13: AI techniques analysis for "Crop diseases prediction. Smart irrigation and precision aerial spraying " use case

Use Case	**Smart Agriculture IoT Living Lab**
AI Application Service	**Crop diseases prediction. Smart irrigation and precision aerial spraying**
AI Service	**Crop diseases prediction**
Outcome type	Classification
Metrics: Acceptance / Failure Criteria	Confusion matrix Minimum Accuracy F1 score
Training strategy	Unsupervised ML/DL, FL
Enhanced AI techniques	GPU or TPU
Inference approach	Near real-time responses
Serving approach	Streaming

Table 14: AI techniques analysis for "sensor aided crop harvesting" use case

Use Case	**Smart Agriculture IoT Living Lab**	
AI Application Service	**Sensor aided crop harvesting**	
AI Service	**V-SLAM**	**Obstacle Avoidance using Deep Learning**
Outcome type	Visual Mapping and Trajectory into space	Binary Classification (Obstacle or Non-Obstacle)
Metrics: Acceptance / Failure Criteria	Minimum Accuracy – Geometric Error Minimization	Minimum Accuracy F1 Score
Training strategy	Unsupervised ML/DL with Other Computer Vision Techniques	Unsupervised ML/DL with Other Computer Vision Techniques
Enhanced AI techniques	GPU	GPU or TPU
Inference approach	Real-time responses	Real-time responses
Serving approach	Streaming	Streaming

3.3 Industry 4.0 Living Lab

In this chapter, an analysis of the Artificial Intelligence needs from the three use cases under the Industry 4.0 Living Lab is provided; including preconditions of each use case, mapping of those preconditions to MLaaS tools, a use case data analysis and a use case AI techniques analysis.

3.3.1 Preconditions per application

1. Human-centred safety in a self-aware indoor factory environment
    a. **Edge computing** resources will be used to support a set of **virtual AI functions** that will process the **real-time location of the AGVs,** based on **the real-time stream coming from the safety cameras**. The AI functions will determine a **potential collision** between AGVs, or between a worker and an AGV and will issue an early warning.
    b. This can give a connected AGV fleet the ability to regulate the speed of vehicles instead of stopping suddenly to avoid a collision. AI algorithms can recalculate the optimal route in real time and provide alternative routes if the planned one has too many delays. In addition, the AGV use rate can be optimized by going where they are most needed instead of following a fixed pattern.
    c. IoT-NGIN will provide a high precision IoT localization layer merging real-time localizations obtained from Ultra-Wide Band (UWB) sensors and a solution providing Visible Light Positioning (VLP). In addition, safety cameras will be deployed to monitor areas with reduced visibility.
    d. The data from the sensors provide a full description of the environment and how it is changing. It allows to model moving objects and skeletonizes human bodies to detect the position of each body part and build a "safety shell" around it to ensure human-centred safety.
2. Human-centred Augmented Reality assisted build-to-order assembly
    a. This UC aims to assist human workers in the assembly line with the use of **Augmented Reality (AR). Machine learning and computer vision** techniques will be used to **detect product defects** and **differentiate** between different components and modules.
    b. IoT-NGIN will be able to **recognize the components and the stage of the assembly process using local ML trained models** and provide assistance and guided instructions, displaying the procedure and next stage of the customizable manufacturing, either using AR classes, mobile devices or small industrial screens.
3. Digital powertrain and condition monitoring
    a. Condition monitoring and predictive maintenance of powertrains and drive units using federated machine learning.
    b. Parameter tuning and optimization (e.g. in terms of energy consumption) of drive units using federated learning.

3.3.2 Mapping to IoT-NGIN MLaaS tools

The requirements collected in the previous section are mapped to IoT-NGIN's MLaaS platform in Table 15.

Table 15: Mapping of Industry 4.0 Living Lab to IoT-NGIN MLaaS tools

Big Data and ML framework	Deep Learning	Reinforcement learning	Federated learning	ML models sharing
High precision IoT localization layer, safety cameras, contextual IoT data, IoT localization, RFID sensors and camera analysis.	Potential collision between AGVs, or between a worker and an AGV. Recognition of components and stage of assembly.	N/A	Federated ML decision on constrained resources	N/A

For the Industry 4.0 Living Lab three *AI Applications* will be developed, namely:

- "Human-centred safety in a self-aware indoor factory environment", which contains two AI Services: "Collision Detection" and "AGV Route Planning".
- "Human-centred Augmented Reality assisted build-to-order assembly", which contains one AI service: "(AR) Object detection and classification".
- "Digital powertrain and condition monitoring", which contains two AI Services: "Predictive maintenance of powertrains" and "Energy consumption optimization".

The analysis of the data and the AI techniques that will be used to build these AI services is described in the following sections.

3.3.3 Use Case Data Analysis

The following tables describe, from a generic point of view, the features of the data produced in the Industry 4.0 Living Lab that is relevant for the creation of AI Applications.

Table 16: Data analysis for "human-centred safety in a self-aware indoor factory environment" use case

Use Case		Industry 4.0 Use Cases & Living Lab	
AI Application Service		Human-centred safety in a self-aware indoor factory environment	
AI Service		Collision Detection	AGV Route Planning
Data Sources	IoT Devices	Cameras Ultra-Wide Band (UWB) sensors AGV Sensors	
	Data Type(s)	JSON, OPC UA	
	Data size	To be set	
	Data communication protocols	TCP OPC UA	
	Data availability	Request-only access	
Potential Bias		None identified so far	
Heuristics/Assumptions		Data gathered to edge server	
Data privacy level		Private	

Table 17: Data analysis for "human-centred augmented reality assisted build-to-order assembly" use case

Use Case		Industry 4.0 Use Cases & Living Lab
AI Application Service		Human-centred Augmented Reality assisted build-to-order assembly
AI Service		(AR) Object detection and classification
Data Sources	IoT Devices	Public displays, tablet, Cameras Ultra-Wide Band (UWB) sensors AGV Sensors
	Data Type(s)	JSON, OPC UA
	Data size	To be set
	Data communication protocols	TCP OPC UA
	Data availability	Request-only access
Potential Bias		None identified sofar
Heuristics/Assumptions		Data gathered to edge server,
Data privacy level		Private

Table 18: Data analysis for "digital powertrain and condition monitoring" use case

Use Case		Industry 4.0 Use Cases & Living Lab	
AI Application Service		**Digital powertrain and condition monitoring**	
AI Service		**Predictive maintenance of powertrains**	**Energy consumption optimization**
Data Sources	**IoT Devices**	Variable speed drive, smart sensor, heat camera	
	Data Type(s)	JSON, OPC UA	
	Data size	To be set	
	Data communication protocols	TCP OPC UA MQTT	
	Data availability	Request-only access	
Potential Bias		Data is heavily affected by the powertrain's operating point (torque and speed).	
Heuristics/Assumptions		Data gathered to edge server, which can forward and adapt to different protocols	
Data privacy level		Private	

3.3.4 Use Case AI Techniques Analysis

The following tables describe, from a generic point of view, the techniques of AI that will be used in the Industry 4.0 Living Lab for the creation of AI Applications.

Table 19: AI techniques analysis for "human-centred safety in a self-aware indoor factory environment" use case

Use Case	Industry 4.0 Use Cases & Living Lab	
AI Application Service	Human-centred safety in a self-aware indoor factory environment	
AI Service	Collision Detection	AGV Route Planning
Outcome type	Prediction	
Metrics: Acceptance / Failure Criteria	Zero collisions / a collision with humans or human operated vehicles occurs	
Training strategy	Federated learning	
Enhanced AI techniques	N/A	
Inference approach	Real time, data stored for later analysis	
Serving approach	HTTP, others possible depending on implementation	

Table 20: AI techniques analysis for "human-centred augmented reality assisted build-to-order assembly" use case

Use Case	Industry 4.0 Use Cases & Living Lab
AI Application Service	Human-centred Augmented Reality assisted build-to-order assembly
AI Service	(AR) Object detection and classification
Outcome type	Classification
Metrics: Acceptance / Failure Criteria	To be defined
Training strategy	Federated learning
Enhanced AI techniques	N/A
Inference approach	Real time, data stored for later analysis
Serving approach	HTTP, others possible depending on implementation

Table 21: AI techniques analysis for "digital powertrain and condition monitoring" use case

Use Case	Industry 4.0 Use Cases & Living Lab	
AI Application Service	Digital powertrain and condition monitoring	
AI Service	Predictive maintenance of powertrains	Energy consumption optimization
Outcome type	Binary classification Anomaly detection	
Metrics: Acceptance / Failure Criteria	If data drifts away from a normal operation based on some statistical measure of normality or ML model bound.	
Training strategy	Federated learning model of 'normal operation'[5]	
Enhanced AI techniques	N/A	
Inference approach	Batched / Stored, later catching	
Serving approach	HTTP MQTT Script (Edge node can adapt)	

3.4 Energy Grid Active Monitoring/Control Living Lab

In this chapter, an analysis of the Artificial Intelligence needs from the three use cases under the Energy Grid Active Monitoring/Control Living Lab is provided; including preconditions of each use case, mapping of those preconditions to MLaaS tools, a use case data analysis and a use case AI techniques analysis.

3.4.1 Preconditions per application

1. Move from Reacting to Acting in Smart Grid Monitoring & Control
   a. **AI/ML-based analytics to train models tracking the health of the grid** and indicating that maintenance is required before obvious performance degradation or even failure, along with urban traffic scenario and traffic predictions
2. Driver-friendly dispatchable EV charging

[5] Assume that the powertrain is working normally for some days after being commissioned i.e. this data can be classified as normal.

a. **AI/ML-based analytics to train models** that are able to forecast the energy demand for the EV charges according to the data of the system.

3.4.2 Mapping to IoT-NGIN ML tools

The requirements collected in the previous section are mapped to IoT-NGIN's MLaaS platform in Table 22.

Table 22: Mapping of Energy Grid Active Monitoring / Control Living Lab to IoT-NGIN MLaaS tools

Big Data and ML framework	**Deep Learning**	**Reinforcement learning**	**Federated learning**	**ML models sharing**
High-tech power sensor	Health of energy grid, traffic predictions, local discrepancies, discharge detection.	N/A	Federated ML hosted on the IoT nodes	N/A

For the Energy Grid Active Monitoring/Control Living Lab two *AI Applications* will be developed, namely:

- "Move from Reacting to Acting in Smart Grid Monitoring & Control", which contains two AI Services: "Grid operation optimization", "Consumption prediction" and "Generation prediction".
- "Driver-friendly dispatchable EV charging", which contains one AI service: "Forecasting of energy demand".

The analysis of the data and the AI techniques that will be used to build these AI services is described in the following sections.

3.4.3 Use Case Data Analysis

The following tables describe, from a generic point of view, the features of the data produced in the Energy Grid Active Monitoring / Control Living Lab that is relevant for the creation of AI Applications.

Table 23: Data analysis for "move from reacting to acting in smart grid monitoring & control" use case

Use Case		Energy Grid Active Monitoring/Control Living Lab		
AI Application Service		Move from Reacting to Acting in Smart Grid Monitoring & Control		
AI Service		Grid operation optimization	Consumption prediction	Generation prediction
Data Sources	IoT Devices	6 Power quality analysers (PQA)	150 Smart meters	2 Phasor measurement units (PMU)
	Digital Twin	YES	YES	YES
	Data Type(s)	Csv - Json, Numerical data	Json, Numerical data	Json, Numerical data
	Data size	2 GB/year/device	1 GB/Year/device	5 GB/Year
	Data communication protocols	HTTP	MQTT	MQTT
	Data availability	Real-Time	Real-Time	Real-Time
Potential Bias		N/A		
Heuristics/Assumptions		N/A		
Data privacy level		Proprietary		

Table 24: Data analysis for "driver-friendly dispatchable EV charging" use case

Use Case		**Energy Grid Active Monitoring/Control Living Lab**
AI Application Service		**Driver-friendly dispatchable EV charging**
AI Service		**Forecasting of energy demand**
Data Sources	**IoT Devices**	Charging stations Electric vehicle OBD devices Near real-time smart meters
	Digital Twin	Yes
	Data Type(s)	String
	Data format	JSON
	Data size	Some KBs per day per device
	Data communication protocols	REST API and MQTT
	Data availability	Real-time
Potential Bias		More consumption during mornings
Heuristics/Assumptions		N/A
Data privacy level		Private / Open source

3.4.4 Use Case AI Techniques Analysis

The following tables describe, from a generic point of view, the techniques of AI that will be used in the Energy Grid Active Monitoring / Control Living Lab for the creation of AI Applications.

Table 25: AI techniques analysis for "move from reacting to acting in smart grid monitoring & control" use case

Use Case	**Energy Grid Active Monitoring/Control Living Lab**		
AI Application Service	**Move from Reacting to Acting in Smart Grid Monitoring & Control**		
AI Service	**Grid operation optimization**	**Consumption prediction**	**Generation prediction**
Outcome type	Optimized electrical parameter (voltage, current, power…)	Predicting values	Predicting values
Metrics: Acceptance / Failure Criteria	>0.8 close to nominal values	>0.8 close to real	>0.8 close to real
Training strategy	Continuous learning	Continuous learning	Continuous learning
Enhanced AI techniques	N/A	N/A	N/A
Inference approach	Real-time	Real-time	Real-time
Serving approach	Rest API	Rest API	Rest API

Table 26: AI techniques analysis for "driver-friendly dispatchable EV charging" use case

Use Case	**Energy Grid Active Monitoring/Control Living Lab**
AI Application Service	**Driver-friendly dispatchable EV charging**
AI Service	**Forecasting of energy demand**
Outcome type	Predicting day-ahead energy consumption values related to smart meters collected data
Metrics: Acceptance / Failure Criteria	>0.8 close to real
Training strategy	Federated learning Continuous learning
Enhanced AI techniques	GPU
Inference approach	Real-Time predictions
Serving approach	REST API

4 Big Data and ML framework: IoT-NGIN MLaaS Architecture

Throughout this section, the MLaaS platform to be developed and integrated over the IoT-NGIN project is going to be described. First, an initial analysis of the state-of-the-art considering this kind of platforms is provided. Then, an architecture for the platform is proposed, which includes the definition of:

- Actors of the platform
- Use cases for the platform
- Logical view of platform components
- Sequence diagrams for each platform use case considering the components

Then, an overview of how the data is managed in the platform following the BDVA architecture is provided in Section 4.3. Finally, the privacy-preserving federated learning framework that will be on top of the MLaaS platform is described.

4.1 MLaaS state-of-the-art definition

One of the promises of the "internet of things" is the use of device and sensor services to extract knowledge about how a system is performing and helps companies better understand what exactly is going on in various aspects of the system. The data from device and sensor services can be used immediately by a simple system, for example launching an alert if a value is above a specific threshold. However, the multiplication of these devices and sensors generates a growing amount of data and companies face the challenge of the 5 V's of Big Data: Velocity, Volume, Value, Variety, and Veracity. A simple system is no more able to cope with the complexity involved by these 5 V. New solutions are required to be able to react to complex data and to effectively and purposely captured valuable information from the data. With the development of new techniques, new algorithms and the increasing availability of computational power, it is now possible to not only get information about the current state of a system but also extract value from a huge amount of data and predict future states of a system. This will help the stakeholders better understand what exactly is going on in various aspects of the company and better plan for the future.

Primary users of the ML techniques are data science professionals which include expert data scientists, citizen data scientists, data engineers and machine learning engineers/specialists. These people need a platform that can provide all the necessary components to work on data, train ML models, share models and deploy models. Implementing and maintaining such a platform is complex, time-consuming, costly and companies may lack experience. So, one trend in the industry is to provide this kind of platform providing all the necessary services to build and execute ML in a ready-to-use form. In addition, it can be built as a custom-tailored ML system for some specific use cases. Such a platform is commonly referred to as ***Machine Learning as a Service (MLaaS)***. Using MLaaS allows a company to reduce the time and cost of integrating ML into its development and IT environment. By using MLaaS, Data Scientist can upload their data and model for training at the MLaaS platform and can focus on their core competency, i.e. ML development without taking care of the underlying infrastructure which is then provided and managed by another identity (such as in an as-a-service fashion).

There is no formal definition for an ML platform. For example, R. Bohm and G. Digital in [3] define ML platform as "*Broadly speaking, a platform is a set of interconnected services designed to enable all the parts of an application to work together. These platforms not only provide basic services, but they also create ecosystems so that new components can be added, and components can be offered for sale*". In another report by Louis Dorard [4], different types of ML platforms are presented: Pre-trained models as a service, Vertical ML as a service, Semi-specialized machine learning as a service, ML development platforms, ML deployment platforms. In another report by Gartner [5], Gartner defines a Data Science and Machine Learning Platforms (DSML) platform as "*a core product and supporting a portfolio of coherently integrated products, components, libraries and frameworks (including proprietary, partner and open-source)*". In report MLaaS: Machine Learning as a Service [6], authors wrote: "*Because multiple users will be using the same platform, computational resources can be shared or allocated on-demand, reducing overall costs. By specifying a well-defined interface, users can have access to the machine learning process efficiently from anywhere, at any time. Users must not be concerned with implementation and computing resources, focusing mainly on the data itself*."

ML platforms can come in different formats. They can be based on open source or commercial software, and they can run on-premises or in the cloud. For example, the major public cloud actors like Microsoft Azure, Google Cloud Platform (GCP) or AWS have MLaaS offers to provide different AI services. Several software vendors are also offering the MLaaS platform. Most of the literature discusses MLaaS as being in the cloud. However, in the context of IoT where there could be millions of sensors, intermittent connectivity, sensitive or private data, it may not make sense nor be even possible to push all the data to the cloud. Also, in real-life situations such as health monitoring, emergency response and other latency-sensitive applications, the delay caused by transferring the data or/and doing the prediction in the cloud may not be acceptable. Companies must decide what, when and where to send data versus keeping data locally and similarly must decide what, when and where ML model is trained, and prediction is done.

This has led to the notion of *Edge computing* and *Fog computing* where some local resources can be used to process data. Edge computing and fog computing share a lot of similarities and the distinction is not always clear. With edge, computing happens where data is being generated, right at "the edge" of a given network. Edge computer is connected to the sensors and controllers of a given device and then sends data to the cloud. Initially, Edge Computing paradigm devices had limited resources which limited its usage. Fog Computing is a new paradigm addressing this issue by providing a compute layer between the cloud and the edge. As explained in [7] Fog Computing has been defined in many ways. The definition from S. Yi, Q. Li, and C. Li in [8] fits well with the purpose of the IoT NGIN MLaaS: "*Fog Computing is a geographically distributed computing architecture with a resource pool consisting of one or more ubiquitously connected heterogeneous devices (including edge devices) at the edge of the network and not exclusively seamlessly backed by cloud services, to collaboratively provide elastic computation, storage and other services either in remote locations or in a large number of clients nearby*".

As stated in Machine Learning as a Service-Challenges in Research and Applications [9], there is no clear definition of what MLaaS is. Similarly, there are no standard functional requirements and no reference architecture for MLaaS. However, in most of the descriptions, the MLaaS platform provides services to store data, visualize data, perform Extract, Load, Transform (ETL), train ML models, allow transfer learning and leverage pre-trained model. Ideally, MLaaS must be scalable, provide substantial computational resources (e.g., high-performance graphics processing units (GPUs)) and allow for technical interoperability with other systems via standard APIs and standard network protocols.

MLaaS comes with new security concerns. As a shared platform MLaaS faces the challenge of data privacy of the data and security of the overall system. Owners must be sure that their data and model are not stolen, compromised nor being used by unauthorized users. This area is subject to quite a lot of research. For example, in survey Privacy-Preserving Deep Learning on Machine Learning as a Service [10], Privacy-Preserving Deep Learning (PPDL) is presented as a possible solution to this problem for Deep Learning specifically for MLaaS. In addition, the MLaaS platform, being connected to the network and in the case of IoT possibly using new network protocols, is subject to both common and new network attacks. Evaluation in [11] of both the attacks and defences dimensions of an MLaaS system reveals that there is an increasing interest from the research community on the perspective of attacking and defending against various attacks on Machine Learning as a Service platform. IoT-NGIN will participate in this effort by developing privacy-preserving federated Machine Learning (ML) and deep ML/ reinforcement learning techniques to enable ML training without moving sensitive data from its original sites, overcoming legal or privacy constraints, while novel cybersecurity techniques will mitigate poisoning attacks and ensure early attack detection in on-device federated ML.

4.2 IoT-NGIN MLaaS Platform Concept

The following section details the architecture of the MLaaS framework from different perspectives: at the architecture level, at the actors level, at the components level and at the functionality level.

4.2.1 IoT-NGIN MLaaS Concept

Throughout this section the MLaaS platform is described at the architecture and business level, including the functional and non-functional requirements as well as the main interactions of the platform. In later sections the architecture is realized into use cases, logical components view and sequence diagrams.

4.2.1.1 Business Context & Motivation

One of the ambitions of Europe is to foster, strengthen and support the development and wide adoption of Big Data Value technologies to sustain the growth of Big Data and remain competitive. As stated in BDVA SRIA4.0, three dimensions (over seven) of a strong Big Data ecosystem relate to:

- Data. The availability of data and access to data sources are paramount concerns. There is a broad range of data types and data sources: structured and unstructured data; multilingual data sources; data generated from machines and sensors; data at-rest and data-in-motion. Value is created by acquiring data, combining data from different sources, and providing access to data with low latency while ensuring data integrity and preserving privacy. Pre-processing, validating and augmenting data, as well as ensuring their integrity and accuracy, add value.
- Skills. In order to leverage the potential of Big Data Value, a key challenge for Europe is to ensure the availability of highly and relevantly skilled people who have an excellent grasp of the best practices and technologies for delivering Big Data Value

within applications and solutions. There will be a need for data scientists and engineers who have expertise in analytics, statistics, machine learning, data mining and data management. These specialists should be combined with other experts who have strong domain knowledge and the ability to apply this know-how within organisations to create value.

- Technical. Key aspects such as real-time analytics, low latency and scalability in processing data, new and rich user interfaces, interacting with and linking data, information and content, all have to be developed in order to open up new opportunities and to sustain or develop competitive advantages. As well as having agreed approaches, the interoperability of datasets and data-driven solutions is essential to ensure wide adoption within and across sectors.

The goal of the *MLaaS architecture* is to define a **hybrid edge-cloud MLaaS (Machine Learning as a Service) framework** that will support these three dimensions by providing the data scientists and engineers with access to data and the tools they need for their role without taking care of the underlying infrastructure and by providing capabilities for real-time analytics, low latency and scalability in processing data. More information related to how the MLaaS platform follows the BDVA architecture can be found in Section 4.3: Data Storage and Data Management.

4.2.1.2 High-level IoT-NGIN architecture

IoT-NGIN aims to act as the engine that will drive the evolution to the next generation of IoT. As such, IoT-NGIN addresses multi-variate diverse network topologies and computing paradigms for IoT systems acroos a number of domains. With the increasing emergence of IoT devices and things, with diverse computing capabilities, computational loads could be executed even on device or offloaded to more powerful devices. At the same time, the energy efficiency constraints, combined with stringent delay requirement, depending on the underlying IoT application, often mandate for the computation to be executed as close as possible to the data sources. This has led to the adoption of edge computing, which usually refers to computation jobs being offloaded within the same LAN. However, heavier computations in less time-sensitive applications have led to the emergence of fog computing. In this paradigm, computation is done at more powerful devices in remote locations compared to the data sources, but still not referring to cloud resources. Last, but least cloud computational resources can be exploited, utilizing flexible plans of data centre resources.

IoT-NGIN covers all three computing paradigms, as illustrated in the network topology of Figure 4. As shown in the figure, computation tasks can be executed at the edge, fog or cloud layers. Indeed, computation offloading can be performed directly from the edge layer to other edge devices or fog or cloud resources, but also from fog to cloud resources.

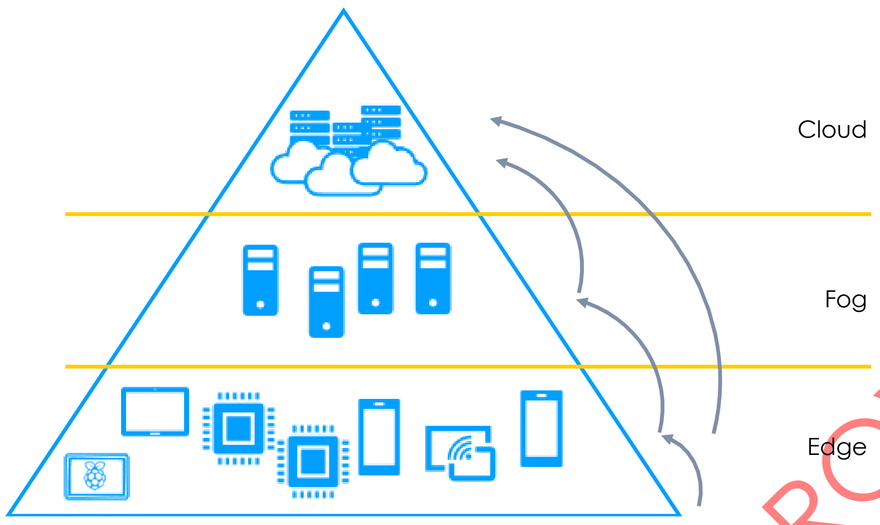


Figure 4: IoT-NGIN network topology

The complete IoT-NGIN functionality can be mapped to the network nodes, as depicted in the draft meta-architecture figure included in the the Description of Action (DoA), which is illustrated in Figure 5. The architecture specifies the services offered both in IoT devices and in more capable edge / fog / cloud nodes of an IoT-NGIN powered system, supporting a modular and flexible combination of next generation (5G, mesh and fog) communications and federations of IoT systems and Distributed Ledger Technologies (DLTs), along with Big Data Analytics and privacy preserving federated ML for distributed intelligence. As shown in the figure, five flexible meta-architectural functional groups are foreseen, namely Federated Communications, Microservices and Virtual Network Functions (VNF), Federated Data Sovereignty, Federation of Big Data Analytics & ML and Human-Centred Augmented Reality Tactile IoT.

The MLaaS function is plays a central role in the overall IoT-NGIN design, which includes various ML-based components. The MLaaS function is supported in the Big Data Analytics, as well as the primary ML-based components appearing in purple color in the figure.

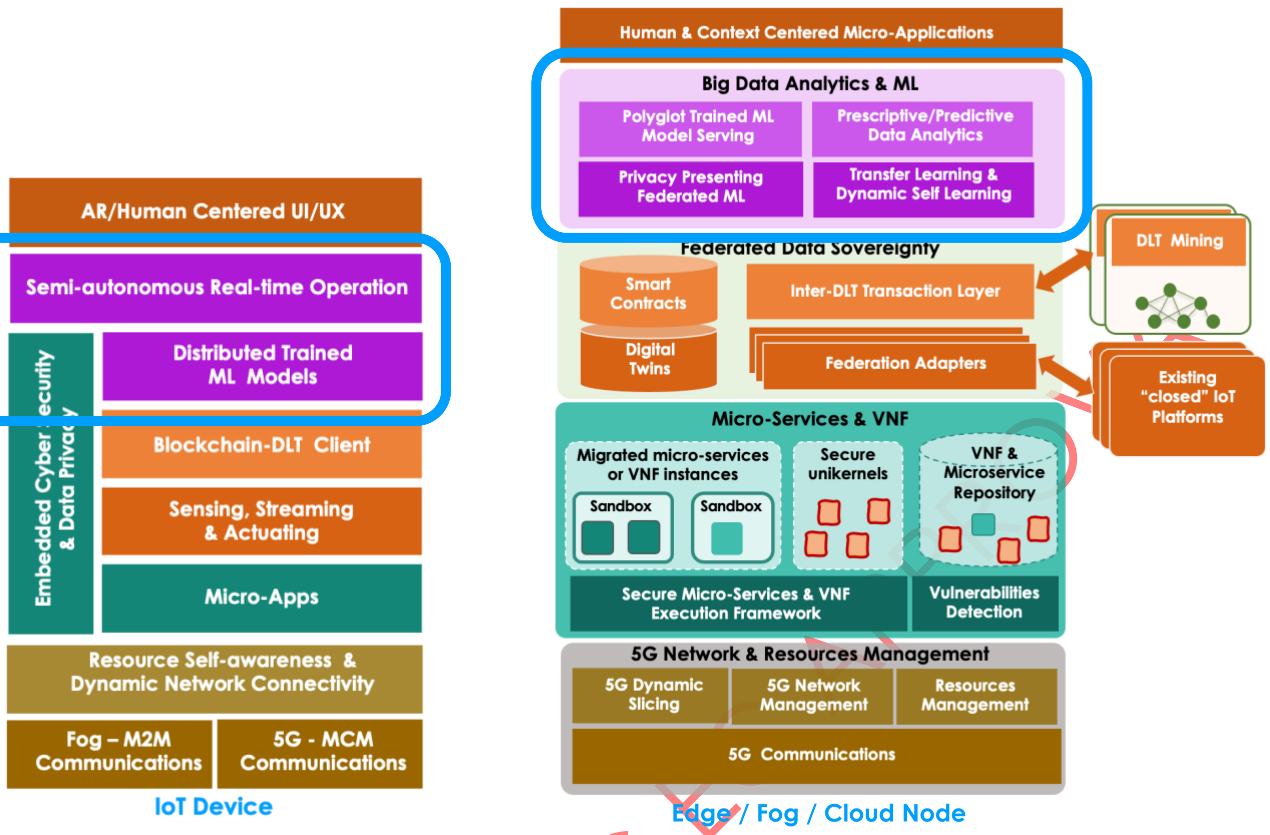

AR/Human Centered UI/UX
Semi-autonomous Real-time Operation
Embedded Cyber Security & Data Privacy
Distributed Trained ML Models
Blockchain-DLT Client
Sensing, Streaming & Actuating
Micro-Apps
Resource Self-awareness & Dynamic Network Connectivity
Fog – M2M Communications
5G - MCM Communications
IoT Device
Human & Context Centered Micro-Applications
Big Data Analytics & ML
Polyglot Trained ML Model Serving
Prescriptive/Predictive Data Analytics
Privacy Presenting Federated ML
Transfer Learning & Dynamic Self Learning
Federated Data Sovereignty
Smart Contracts
Digital Twins
Inter-DLT Transaction Layer
Federation Adapters
DLT Mining
Existing "closed" IoT Platforms
Micro-Services & VNF
Migrated micro-services or VNF instances
Sandbox
Sandbox
Secure unikernels
VNF & Microservice Repository
Secure Micro-Services & VNF Execution Framework
Vulnerabilities Detection
5G Network & Resources Management
5G Dynamic Slicing
5G Network Management
Resources Management
5G Communications
Edge / Fog / Cloud Node

Figure 5: IoT-NGIN proposed architecture, DoA version

4.2.1.3 IoT Device Classification for MLaaS

The IoT Device is not part of the MLaaS platform per se, but it can be a consumer of the platform and can have strong interaction with it. As there are many diverse types of IoT devices there will be different levels of possible interactions between the IoT device and the edge/fog/cloud node. The services used from MLaaS platform will depend on the IoT device capabilities.

The IoT devices could range from a simple Microcontroller (Arduino for example) to a large object with high compute capabilities (a Street Light or a Wind Turbine for example). Looking at the range of possible IoT devices, the characteristics of a micro-controller are:

- Has low CPU and memory capabilities
- Runs on battery which possibly should last several years
- Has limited communication capabilities (only low-power wide-area network – LPWAN, for example)
- Can only run simple ML model (TensorFlow Lite[6], etc.)

In addition, the characteristics of a Big Thing are:

- Has higher (or at least decent) CPU and memory capabilities
- Is continuously powered

[6] https://www.tensorflow.org/lite

- Has a lot of communication capabilities (4G/5G, Wifi, Ethernet, etc.)
- Can use complex AI models (Tensorflow[7], PyTorch[8] models, etc.)

As the IoT devices has increasing compute capabilities, it will be able to have higher interaction with the IoT-NGIN Edge node and higher potential autonomy. Figure 6 illustrates the differences in terms of capabilities between a simple IoT device (a micro-controller) and a bigger IoT device (a Robot) with regards to the services they can use from the IoT-NGIN edge node.

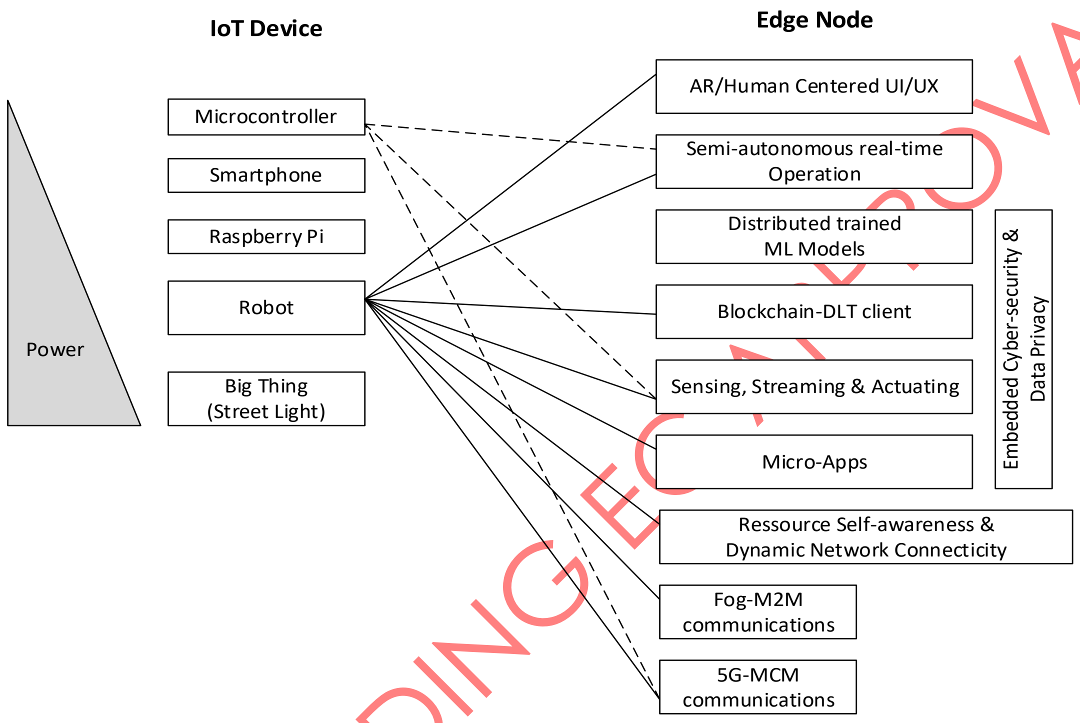

IoT Device
Edge Node
Power
Microcontroller
Smartphone
Raspberry Pi
Robot
Big Thing (Street Light)
AR/Human Centered UI/UX
Semi-autonomous real-time Operation
Distributed trained ML Models
Blockchain-DLT client
Sensing, Streaming & Actuating
Micro-Apps
Ressource Self-awareness & Dynamic Network Connecticity
Fog-M2M communications
5G-MCM communications
Embedded Cyber-security & Data Privacy

Figure 6: IoT Device – Edge Relationship

Indicatively, the *micro-controller* will not be able to use AR/Human centered UI/UX, micro-apps or Blockchain technology in contrast to a *Robot* which will have enough capabilities to use these services. It may be useful to classify the IoT devices based on their capabilities in order to evaluate the potential interaction with the IoT-NGIN services and especially with the MLaaS platform. Table 27 presents a classification with some of the capabilities of the devices.

[7] https://www.tensorflow.org/
[8] https://pytorch.org/

Class	Device type	Can do prediction?	Can do Federated ML?
1	Microcontroller	Prediction with only simple model and framework like TinyML	Likely not
2	Smartphone	Prediction only with simple model	Yes with simple model
3	Affordable low-powered computer (e.g.: Raspberry Pi)	Prediction with model of medium complexity	Yes with model of medium complexity
4	Robot with medium-powered computer	Prediction with possibly complex model (image / video recognition)	Yes
5	Big Thing	Prediction with complex model	Yes

Table 27: IoT device classification

It is worth noting that even in the case that the IoT device cannot do prediction or participate in Federated Machine Learning, it may still be possible to perform such tasks using a Digital Twin that will mimic the behaviour of the device.

4.2.1.4 MLaaS platform high-level overview

Considering the network topology and the draft IoT-NGIN meta-architecture described in section 4.2.1.2 "High-level IoT-NGIN architecture", the IoT device characteristics, as well as the IoT-NGIN requirements derived from the Living Lab use cases in deliverable document D1.1 "Definition analysis of use cases and GDPR Compliance" [2], the high-level architectural concept of the MLaaS platform is defined in this section. Specifically, Figure 7 maps the MLaaS functionality in the IoT devices, the Edge nodes and the cloud, highlighting basic interactions among them in the MLaaS workflow. These functionalities are mapped to logical platform components in Section 4.2.5 "MLaaS Platform Logical view".

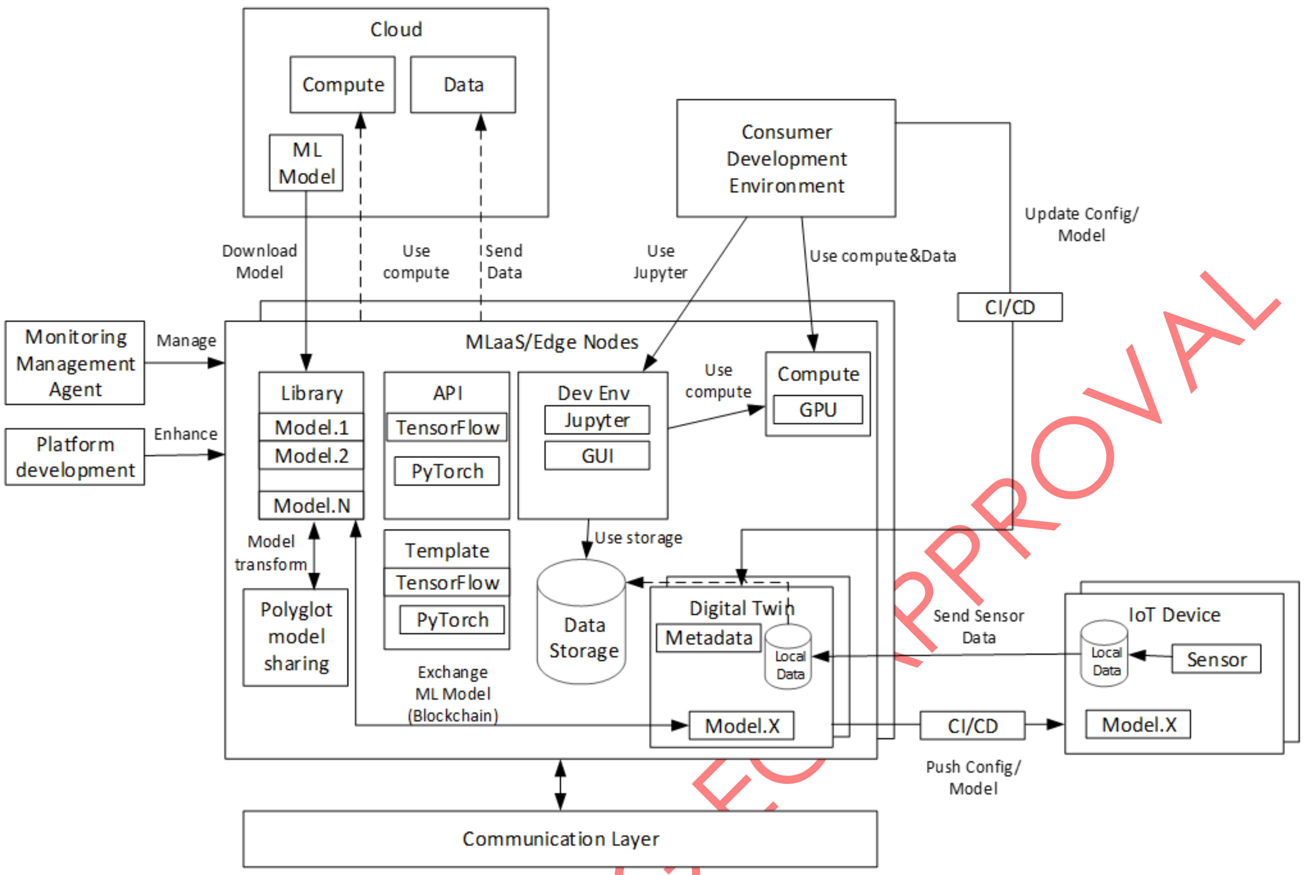


Figure 7: MLaaS platform architecture overview

From an architectural point of view, the platform provides the following functionalities and interactions, which are detailed and explained later throughout section 4.3:

- A set of edge nodes providing the MLaaS functions. This set of edge nodes can be located in the same location or distributed over several locations.
- Interaction with the Cloud:
  - The platform can interact with the cloud to download AI Models
  - The edge nodes may possibly use computation from the cloud when extra compute resources are required
  - The platform may send data to the cloud for data aggregation or data archiving
- Interaction with the Development Environment of a consumer of the platform (Data Scientist, AI developer)
  - The consumer can use some of the development tools of the platform (Jupyter notebooks for example in the figure)
  - The consumer can use compute service from the platform directly or via the development tool of the platform
  - The consumer can update a Digital Twin that would be hosted in the MLaaS platform. The consumer can update the devices via a Continuous Integration/Continuous Delivery pipeline
- Data Storage
  - The development environment can use the storage either to retrieve data (for training for example) or to store data (following ETL process for example)

- The Data storage can be used to store data (from sensor for example) by the Digital Twin (as illustrated in the figure) or directly by the IoT device in case the IoT system has no Digital twin
- The Data Storage is also used for secondary purpose like storing ML from the library or ML template

- Models Library
  - The ML library is used to store ready to use models by a consumer of the platform. These models could be used as is or for transfer learning.
- Polyglot model sharing
  - The Polyglot model sharing component is a software that reads the Model library and transforms models into an standard format. It may also offer a transfer learning API.
- API
  - The platform host standard API like Tensorflow or Pytorch so that a consumer can directly used these API from the platform
- Development Environment
  - The platform will have a limited integrated ready to use Development Environment (like Jupyter notebooks) that will allow a developer to create AI Models without having to setup his own development environment, e.g. directly on the platform as a Service
- Interaction from the Management Agent
  - Depending on user case and associated SLA, the platform can be monitored to ensure it is up and running and in good condition. Also the Monitoring Management Agent will ensure that the platform is up to date in terms of patching and software version. The Management Agent will also be in charge of ensuring the platform is and stays secure.
- Platform Development
  - The platform may be provided by an entity who will be responsible to develop the platform, i.e. add new functionalities, perform major upgrades of some of the components, change components, etc.
- Interaction with Digital Twin / IoT device
  - The Digital Twin or the IoT Device is not a component of the MLaaS platform but is is expected that it will have major interaction with the platform as it is described as part of the architecture of the solution. Such interaction is detailed in section 4.2.1.4.2 "Inference on IoT Device " and section 4.2.1.4.3 "Digital Twins
- Interaction with a communication Layer
  - The platform will be using a communication layer provided by the underlying infrastructure to exchange data with external components like a Cloud (Public or Private), a Digital Twin or an IoT Device, a third party, external data sources, etc. It is especially expected that 5G will be supported by the communication layer.

4.2.1.4.1 MLaaS Platform and Federated Machine Learning

As derived from the draft IoT-NGIN meta-architecture, the platform should support Federated Machine Learning. In this context, the MLaaS Platform may have one or several of the following functions:

- In the case of several disjoint MLaaS platform instances, those instances will act as Edge Nodes using data from the Data Storage. The Cloud or one of the MLaaS platform instances would then be the Aggregator Node.
- The MLaaS platform will act as the aggregator node in the case the Digital Twins or IoT Devices are the Edge Node
- The MLaaS platform will provide Compute capabilities to a Digital Twin or IoT Device during the training process

The exploitation of the platform will depend on the use case and the platform may have several roles (via separate services, for example).

4.2.1.4.2 Inference on IoT Device

One of the functions of the MLaaS platform is to collect data sent by the IoT device. The IoT device may also request for inferences to the MLaaS platform. However another key function is to create AI models that can be loaded onto the IoT devices. Using these models, the IoT devices should be then capable of doing local prediction using local data (data from a sensor measurement and small cache for example). Such local prediction may be required if low latency is needed or when the device has lost connectivity to the edge node. Interaction between the IoT device and the MLaaS platform is illustrated in Figure 8.

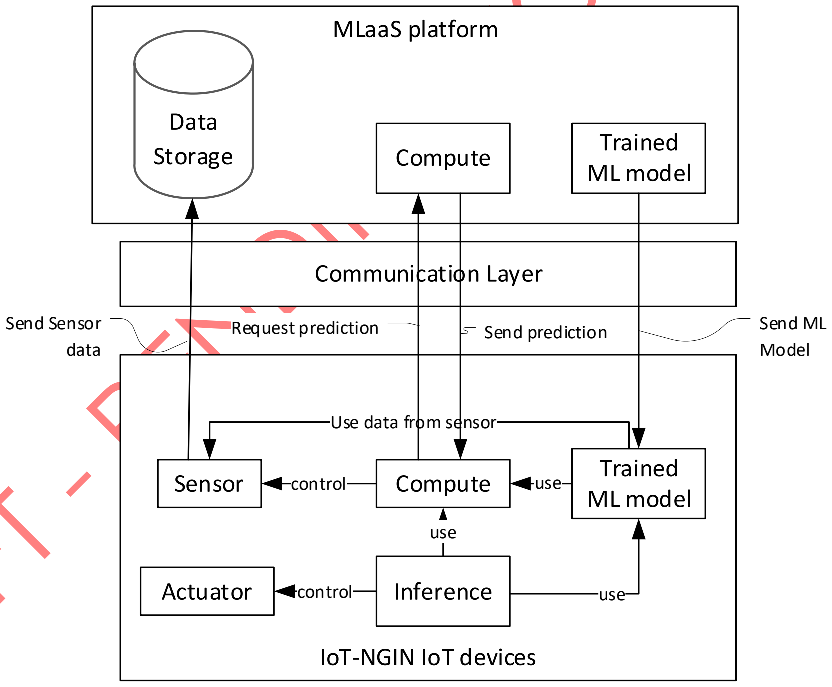


Figure 8: IoT Device – MLaaS platform interaction

Apart from the device sending sensor data and receiving AI Model from the MLaaS platform the figure illustrates how the IoT device can work in an autonomous mode. The compute part (an application) of the device controls a sensor to retrieve some data. The data received from the sensor is then used with the ML Model do perform a prediction (Inference). Based on the result of the prediction, the IoT device may choose to perform an action, like activating an actuator for example.

4.2.1.4.3 Digital Twins

One of the technological objectives of the IoT-NGIN project is to perform research towards a novel concept of DLT enabled Meta-Level Digital Twin (MLDT) to enable "by design" digital twins' scalability, flexibility and trust. This research will be conducted as part of WP5: Enhancing IoT Cybersecurity & Data Privacy. The Digital Twin will not be as such part of the MLaaS platform. However it is expected that there will be interaction between the Digital Twin and the MLaaS platform. This section gives a simplified and short introduction of the Digital Twin concept and its possible interaction with the MLaaS platform.

For the definition of the Digital Twin, we will use the definition from the Digital Twin Consortium[9]: "A digital twin is a virtual representation of real-world entities and processes, synchronized at a specified frequency and fidelity". Using Digital Twins could present several benefits:

- Digital twin systems transform business by accelerating holistic understanding, optimal decision-making, and effective action.
- Digital twins use real-time and historical data to represent the past and present and simulate predicted futures.
- Digital twins are motivated by outcomes, tailored to use cases, powered by integration, built on data, guided by domain knowledge, and implemented in IT/OT systems.

The Digital Twin Consortium sees two categories of digital models:

- A representational model which consists of structured information which generally represents the states of entities or processes.
- A computational simulation model which is an executable model of a process and consists of data and algorithms that input and output representational models.

Whatever the model is, a key function of a Digital Twin is to synchronize itself with the real world. This means:

- The virtual representation should match more closely the real world. This can be achieved via observation mechanisms (sensors, laser scans, satellite imaging, radar, videos)
- The real world should match the virtual representation of a desired state more closely. This can be achieved via intervention mechanisms (actuators, robots)
- The real world should match the virtual configuration of a desired state. This can be achieved via regular update of the real world via for example a CI/CD pipeline.

Several parameters could be used for the synchronisation. Which one are used and with which values depends on the context and would have to be decided on case-by-case basis. Example of possible parameters are:

- Observational synchronization frequency
- Interventional synchronization frequency
- Frequency by mechanism (i.e. sensors and laser scans)

The Digital Twin Consortium introduces the notion of a Digital twin system which is a way to implement a digital twin. It comprises functional subsystems that implement digital twin

[9] https://www.digitaltwinconsortium.org/

system features. Figure 9, from the Digital Twin Consortium website, gives an overview of a Digital Twin System.

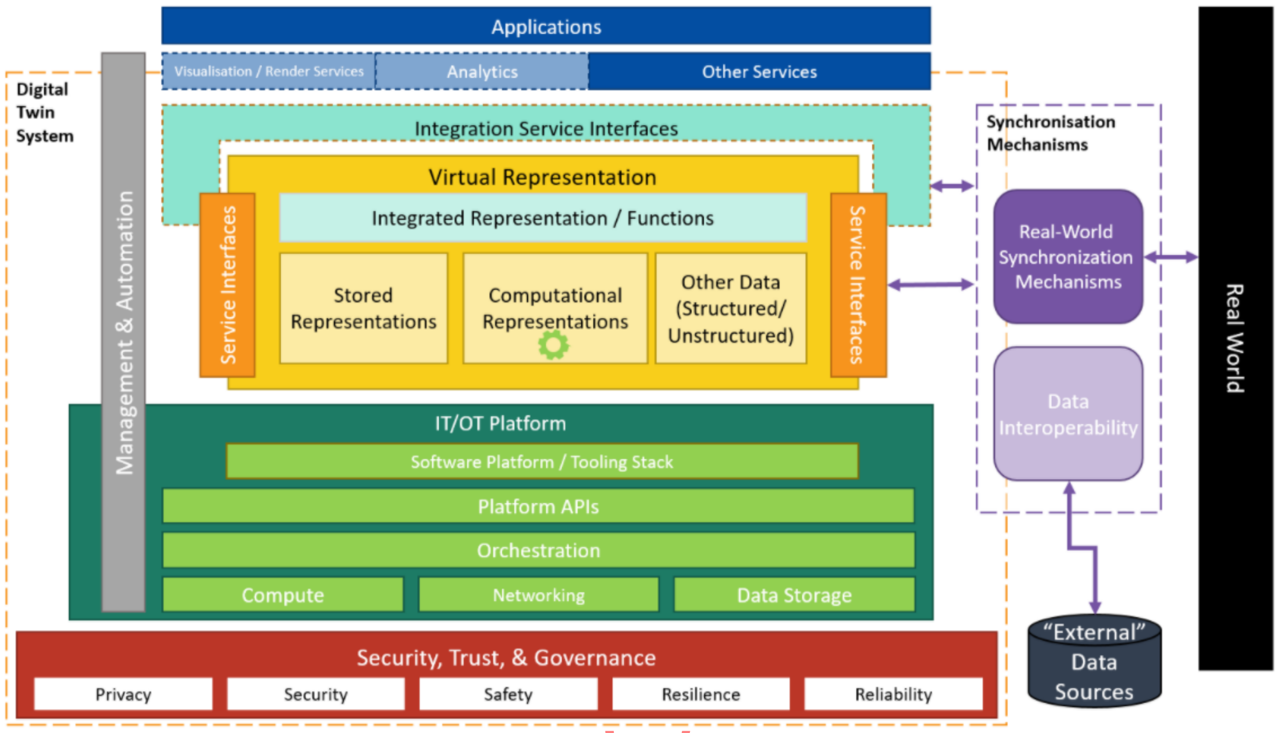


Figure 9: Digital Twin System[10]

In the context of MLaaS, a Digital Twin System may interact with the platform in different ways:

- Detect, prevent, predict and optimize through real time analytics based on data from real device
- Send request to the real device to activate an actuator following a prediction
- Federated ML using private data
- Update configurations of real IoT devices
- Check integrity of real world (real state vs desire state)
- Provide data privacy by keeping data local to the digital twin
- Keep some of the data private and share some of the data with the platform.

[10] https://www.digitaltwinconsortium.org/

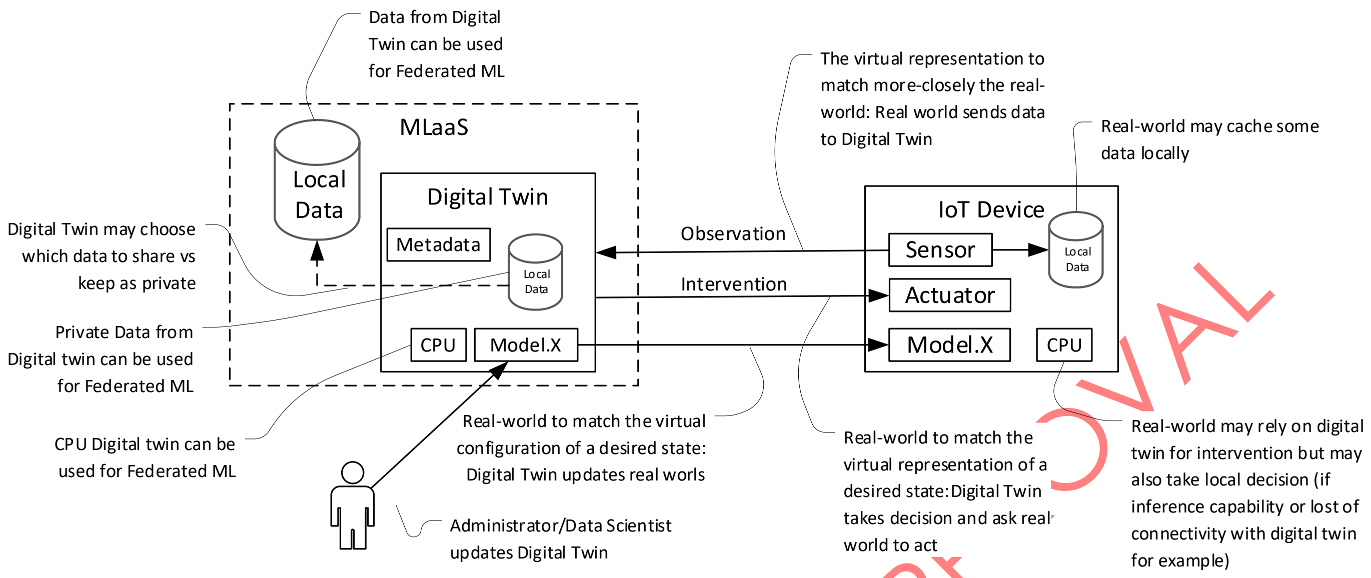

Data from Digital Twin can be used for Federated ML
MLaaS
Local Data
Digital Twin
Metadata
Local Data
CPU
Model.X
Digital Twin may choose which data to share vs keep as private
Private Data from Digital twin can be used for Federated ML
CPU Digital twin can be used for Federated ML
The virtual representation to match more-closely the real-world: Real world sends data to Digital Twin
Real-world may cache some data locally
Observation
Intervention
IoT Device
Sensor
Local Data
Actuator
Model.X
CPU
Real-world to match the virtual configuration of a desired state: Digital Twin updates real worls
Administrator/Data Scientist updates Digital Twin
Real-world to match the virtual representation of a desired state: Digital Twin takes decision and ask real world to act
Real-world may rely on digital twin for intervention but may also take local decision (if inference capability or lost of connectivity with digital twin for example)

Figure 10: MLaaS platform interaction with Digital Twin and IoT Device

From an implementation point of view the Digital Twin System could be from small digital twin (metadata only) to large digital twin (complete application/simulator). Dedicated container or application with multiple containers could be used to make the Digital Twin System. The Digital Twin System is not part of the MLaaS platform but complements the MLaaS schema, providing the required interface for the optimal utilization of the IoT devices.

4.2.2 Actors

The identification of the actors that will interact with the MLaaS platform is key for the specification of the use cases and the functionalities that will be provided. Table 28 and Table 29 describe the actors of the MLaaS platform, which can be separated into two main groups: human actors and non-human actors (systems).

Table 28: Human actors of the MLaaS platform

Human Actors	
Data scientist / AI developer	This actor will make use of the tools provided by the platform to: 1. Analyse and process the data. 2. Design and prepare AI models to be trained within the infrastructure. The actor may also use the platform to prepare an AI method or an AI Service.
AI Application developer	This actor is responsible for the creation of an AI Application by gathering a set of AI methods or AI Services that serve a common purpose. The AI Application may be offered as an AI Application Service, which can be consumed by other human or system actors.
AI User / Integrator	This actor will either consume single AI services or an entire AI Application Service offered directly from the platform. As an alternative, it will deploy AI Methods or AI Services into another infrastructure.
End-user (e.g., farmer, factory operator, etc.)	This actor refers to the final consumer, who is the one who benefits from the actual outcomes of the AI models and the one who will take different actions or decisions according to those results.
Platform provider / administrator (hw + sw)	This actor is the manager of the infrastructure, overseeing the in-premise deployment of the platform as well as its maintenance and updates.
Third parties / Data providers	Refers to companies or other users that benefit externally from the platform capabilities.

Table 29: Non-human actors, either hardware or software, using the MLaaS platform

Non-human actors	
Device (i.e. IoT devices, robots or drones)	Generally speaking, any device that produces data that can be used to train AI Models or feed AI methods, services and applications. Also, they can directly embed the AI Methods / Services or consume their results.
Data platform (e.g. used in the living labs)	A cloud data provider or any other kind of data platform providing data from the cloud or edge. It could also refer to the general data sources of the MLaaS platform.
AI Applications	Applications (e.g. to be developed within the living labs' use cases) that consume the offered AI models by means of AI Services or AI Application Services.
Digital Twins	Mimics of the UC devices or platforms, providing the right response when those are not available or are not desired to be accessed from the MLaaS platform (for example, due to a local electrical issue).
Blockchain network	The management of data in the platform can be integrated with a blockchain network to ensure data integrity immutability.
(IoT-NGIN) Orchestrator	Manager of the deployed instance of cloud-edge infrastructures.

4.2.3 MLaaS Platform Use Cases

The key functionalities which will be implemented as part of the IoT-NGIN MLaaS platform are represented through use case diagrams. The actors identified in the previous section will be the users of these functionalities. The use cases will be the baseline for the definition of the internal platform architecture and the specification of the interactions with external components and systems.

4.2.3.1 Data acquisition

Figure 11 depicts the Data Acquisition platform use case as a UML diagram. It explains the functionality of the MLaaS platform when it comes to getting the data from different data sources.

The use case is divided into five sub-use cases:

1. **UC1.1 Loads dataset**: a Data scientist / AI developer should be able to load a dataset (connects to local data storage and gets the data) from within the MLaaS platform and perform typical operations on it included in the rest of UC1 use cases.
2. **UC1.2 Specifies dataset parameters**: When loading a dataset, the Data scientist / AI developer should be able to specify some parameters to be applied to it, which should be previously defined in the MLaaS data templates. These parameters may include, for instance, the format of the dataset (structured data, images, etc.), the number of instances to load, among many others.

3. **UC1.3 Extracts data**: Once the loading parameters are defined, the Data scientist / AI developer can extract the data (exports it to the platform) from the corresponding data source(s): a Data Platform, a Digital Twin or a Device. Extracting this data can be done in two ways:
   a. **UC1.3.1 Retrieves batch information**: The dataset is loaded once or in some subsets or batches into the MLaaS platform storage.
   b. **UC1.3.2 Connects to data streams**: The dataset is not loaded entirely but rather the data source sends the data periodically to the MLaaS platform storage, which handles the connection to the stream seamlessly.
4. **UC1.4 Feeds AI Methods and Services**: Once the dataset is loaded and it has been processed by the Data pre-processing, a Data scientist / AI developer should be able to feed seamlessly an AI method from the data so that it is used as an ML model input for training or inference.

In this use case the following actors participate:

- **Data scientist / AI developer**: He or She should be able to load a dataset from one or some of the data sources and connect the dataset with an AI method.
- **Device**: Real IoT Devices, robots or another kind of devices that send data in batches or a stream to the MLaaS platform.
- **Digital Twin**: A data source provided by IoT-NGIN instead of the Devices that sends data in batches or a stream.
- **Data Platform**: A data source that is not a Digital twin or a Device, but instead, for example, a cloud provider or any other data provider that sends data to the MLaaS platform.

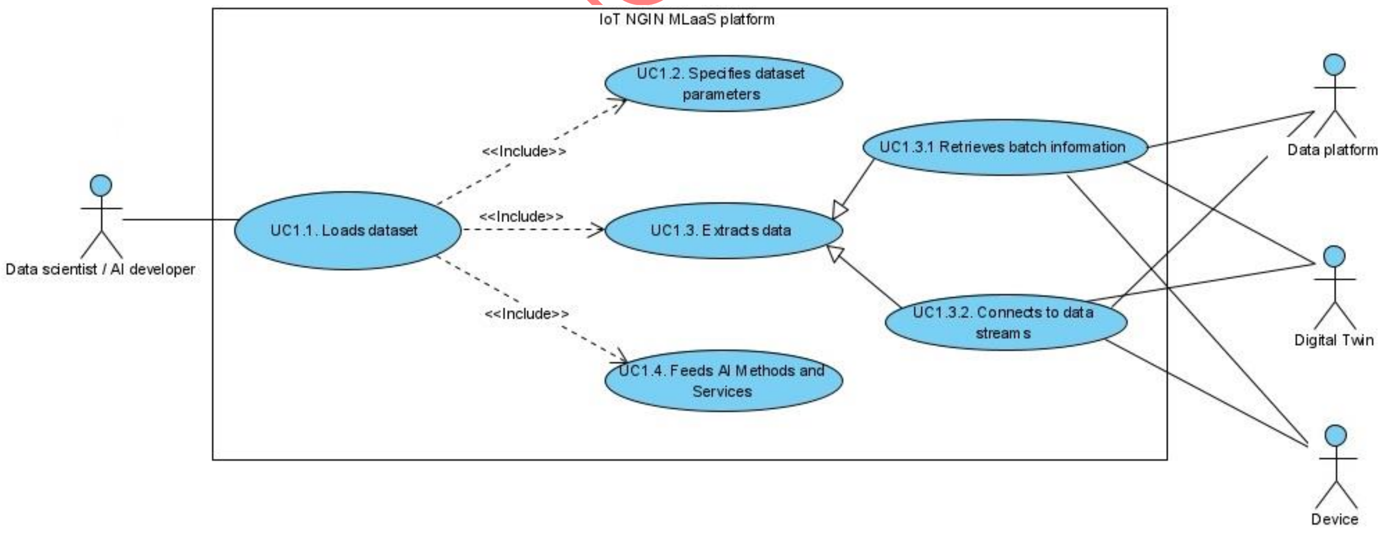

IoT NGIN MLaaS platform
UC1.2. Specifies dataset parameters
<<Include>>
UC1.3.1 Retrieves batch information
Data platform
Data scientist / AI developer
UC1.1. Loads dataset
<<Include>>
UC1.3. Extracts data
<<Include>>
UC1.3.2. Connects to data streams
Digital Twin
UC1.4. Feeds AI Methods and Services
Device

Figure 11: Platform Use Case 1: Data Acquisition

4.2.3.2 Data pre-processing

The Data preprocessing use case provides information related to how the analysis of the data is done in the platform and its preparation to be used as a source to train an AI model.

The use case is divided into seven sub-use cases:

1. **UC2.1 Exploratory Design Analysis**: a Data scientist / AI developer should be able to use data analysis tools and techniques to investigate, analyse, and summarize the main characteristics of a dataset. Typical operations refer to detecting obvious errors, identifying outliers, understanding relationships, unearthing important factors, and finding patterns within data.
2. **UC2.2 Data annotation:** data is annotated by a Data scientist / AI developer to prepare the raw data in order to be consumed by the ML platform. Example of annotation would be an image, video, text, audio annotation and 2D/3D bounding box. This may also include the division of the dataset into train and test sets.
3. **UC2.3 Data wrangling**: a Data scientist / AI developer should be able to restructure and transform data into a standard format. Part of Data wrangling operations are:
    a. **UC2.4 Data Cleaning:** the source data is converted into cleaned data by removing incomplete, errors, noise, duplicates data and inconsistencies.
    b. **UC2.5 Data normalization:** any unstructured data and redundancies that can exist in the data set is removed and eventually the data are grouped logically in order to end up with more structured data.
4. **UC2.6 Data sampling**: As it is not always possible to store the data in full or it is inconvenient to work with data in full, it could be faster to work with a compact summary. The Data scientist / AI developer may decide to obtain a smaller data set with the same structure from the full dataset.
5. **UC2.6 Data augmentation**: value can be added to the data by adding information derived from internal and external sources. Some of the common techniques that can be used are extrapolation, tagging, aggregation and probability technique.

In this use case the following actors participate:

- **Data scientist / AI developer:** He or She should be able to perform various transformation on the dataset in order to prepare the data to be consumed by the ML platform.
- **Third-party**: The third party may use the data from the platform and perform its own transformation on them.

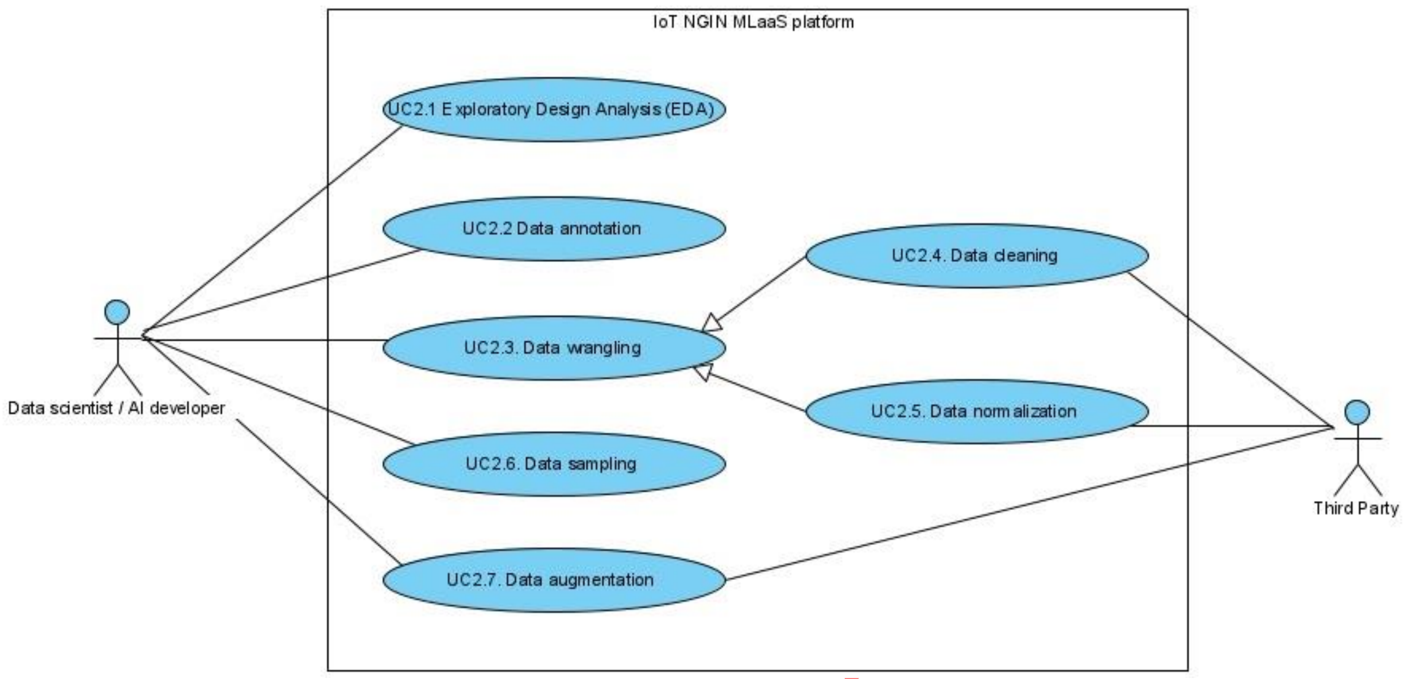


Figure 12: Platform Use Case 2: Data pre-processing

4.2.3.3 AI Modelling (edge)

The "AI Modelling" use case provides the set of actions that are necessary in order to develop an AI model that can be further deployed and tested. The use case is divided into the following sub-use cases:

1. **UC3.1 Feature engineering**: The efficacy of any ML model depends on successful feature engineering. This use case allows the Data Scientist/AI developer to derive better features from raw data, by enabling blending information together to make useful model inputs.
2. **UC3.2 Model selection**: "There is an old theorem in the machine learning and pattern recognition community called the No Free Lunch Theorem, which states that there is no single model that is best on all tasks," said Dr Jason Corso, who is a Professor of Electrical Engineering and Computer Science at the University of Michigan and the co-founder and CEO of Voxel51[11]. Via this use case, the Data Scientist/AI developer can select the algorithm that best fits their dataset, as well as their task. Based on the type of learning selected, the algorithm can change the shape it takes. The Data Scientist/AI developer can select among supervised learning, unsupervised learning, reinforcement learning and transfer learning. This use case is the generalization of the following use cases.
   a. **UC3.4 Supervised Learning**: Supervised learning involves the machine being trained on a given labelled dataset. The Data Scientist/AI developer can select the supervised learning algorithm in order to train accordingly.
   b. **UC3.5 Unsupervised Learning**: In this type of learning, the machine is trained over unlabelled datasets. The Data Scientist/AI developer selects unsupervised

[11] https://voxel51.com.

learning to let the machine find the links between the objects, potentially applying dimensionality reduction, where it reduces the number of variables to decrease the noise.

c. **UC3.8 Reinforcement Learning**: Reinforcement learning allows the Data Scientist/AI developer to train their machine, by learning from their own success and failures. This use case allows the selection and application of algorithms performing reinforcement learning techniques.
d. **UC3.10 Transfer Learning**: In this type of learning, training exploits the knowledge (patterns) learnt in one task to another related task, by actually retraining a pre-trained model in a different task than the original training took place. Through this use case, the Data Scientist/AI developer performs transfer learning.

3. **UC3.3 Model training**: This use case provides the next step in model creation, which is the training of the model. In order to do so, the Data Scientist/AI developer defines the training set, on which the algorithm will be trained. This use case is the generalization of the following use cases.
    a. **UC3.6 Federated Learning**: In federated learning, the training procedure is a product of "coopetition" among a set of nodes, trading off cooperation and competition in deriving an aggregated trained model. The Data Scientist/AI developer may select this type of training through this use case. Moreover, the Device, the Data Platform and the Digital Twin interact with the use case, as they may select to perform training as federated nodes.
    b. **UC3.7 Continuous Learning**: Continuous learning embeds the idea of continually updating an ML model with new data as they become available. This use case enables the Data Scientist/AI developer to perform online training, while it provides the baseline for active learning and multimodal and multitask learning. Moreover, the Device, the Data Platform and the Digital Twin participate in the use case, as they may perform online training on their continuous data streams.
4. **UC3.11 Validation**: The definition of performance metrics is valuable in the evaluation, comparison and analysis of results of training, which will help to further refine the ML models. Indicatively, the classification accuracy, which represents the number of correct predictions divided by the total number of predictions, and multiplied by 100, is an appropriate performance metric for classification problems. Also, the selection of validation dataset affects the model trustworthiness, overcoming the problem of overfitting. In this use case, the Data Scientist/AI developer defines the set of performance metrics against which the ML models will be validated, as well as the success criteria, i.e. the setpoints which will define whether the model can be considered appropriate for the use case it is aimed for. Also, through this use case, the validation takes place and the Data Scientist/AI developer is provided with the validation result.
5. **UC3.12 Model tuning**: In this use case the Data Scientist/AI developer applies model tuning, by configuring a set of hyperparameters, such as coefficients penalties, decision trees, number of layers in neural networks, etc. Model tuning is required in order to derive more accurate predictions for different datasets.
6. **UC3.13. Create AI Method**: In this use case, the Data Scientist/AI developer build saves the model and uses it to create an AI method.

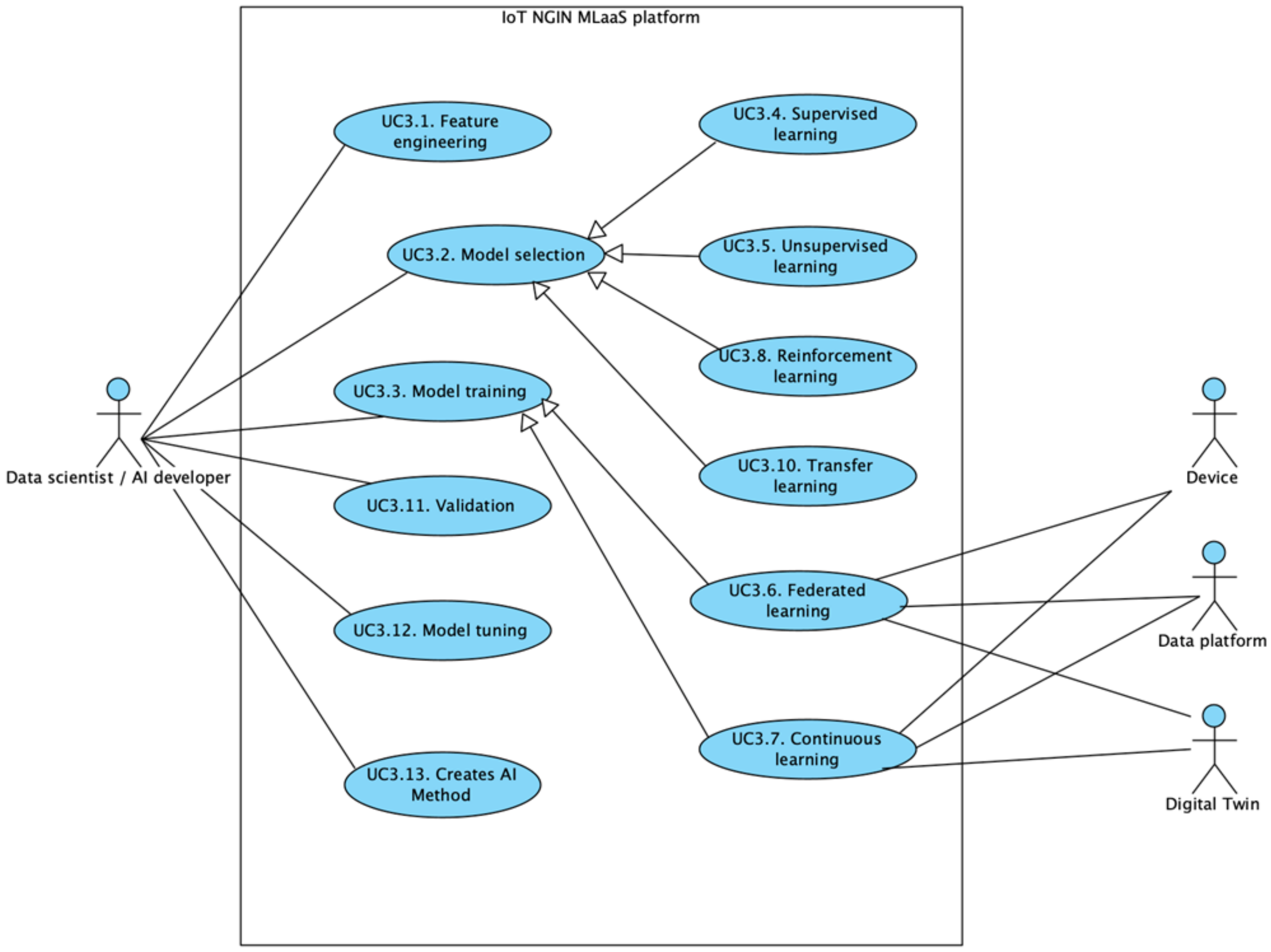

IoT NGIN MLaaS platform
UC3.1. Feature engineering
UC3.4. Supervised learning
UC3.2. Model selection
UC3.5. Unsupervised learning
UC3.8. Reinforcement learning
UC3.3. Model training
UC3.10. Transfer learning
UC3.11. Validation
UC3.6. Federated learning
UC3.12. Model tuning
UC3.7. Continuous learning
UC3.13. Creates AI Method
Data scientist / AI developer
Device
Data platform
Digital Twin

Figure 13: Platform Use case 3: AI Modelling

4.2.3.4 AI Model deployment

The AI Model deployment use case defines the set of actions to be carried out in order to package, optimize and deploy an AI model into a target infrastructure.

The use case is divided into four sub-use cases:

1. **UC4.1 Model Packaging**: a Data scientist / AI developer should be able to put the AI models and their descriptions in one place and a standard format so that they can be easily shared and used through an API. This will enable to place the model in the corresponding AI services, which offer the API to the users.
2. **UC4.2 Model Optimization**: The model should be validated before it is put into production, which means optimizing its parameters before doing the final deployment. In addition, the deployment of models on low-powered devices or constrained hardware may require performing some optimization techniques on the model so that it operates smoothly in that hardware. With regards to this, this sub-use case conforms to the set of actions and operations to optimize (i) the model

hyperparameters after doing a validation operation against a validation dataset, and (ii) the optimization of the model binary according to the requirements of the target deployment infrastructure.

3. **UC4.3 Model Deployment**: a Data scientist / AI developer should be able to deploy the model to a device, a digital twin or a data platform once it has been packaged. The calls for inferences to this model are done via an API defined by the corresponding AI method.
4. **UC3.13 Creates AI Method**: In order to deploy a model, it must be created previously the corresponding AI Method that calls the AI model in the background.

In this use case the following actors participate:

- **Data scientist / AI developer**: He or She should be able to package, optimize and deploy a model.
- **Device**: Real IoT Devices, robots or other kinds of devices to which the data model can be deployed.
- **Digital Twin**: A data source provided by IoT-NGIN instead of the Devices to which the data model can be deployed.
- **Data Platform**: A data platform that is not a Digital twin or a Device, but instead for example a cloud provider or any other data provider to which the data model can be deployed to.

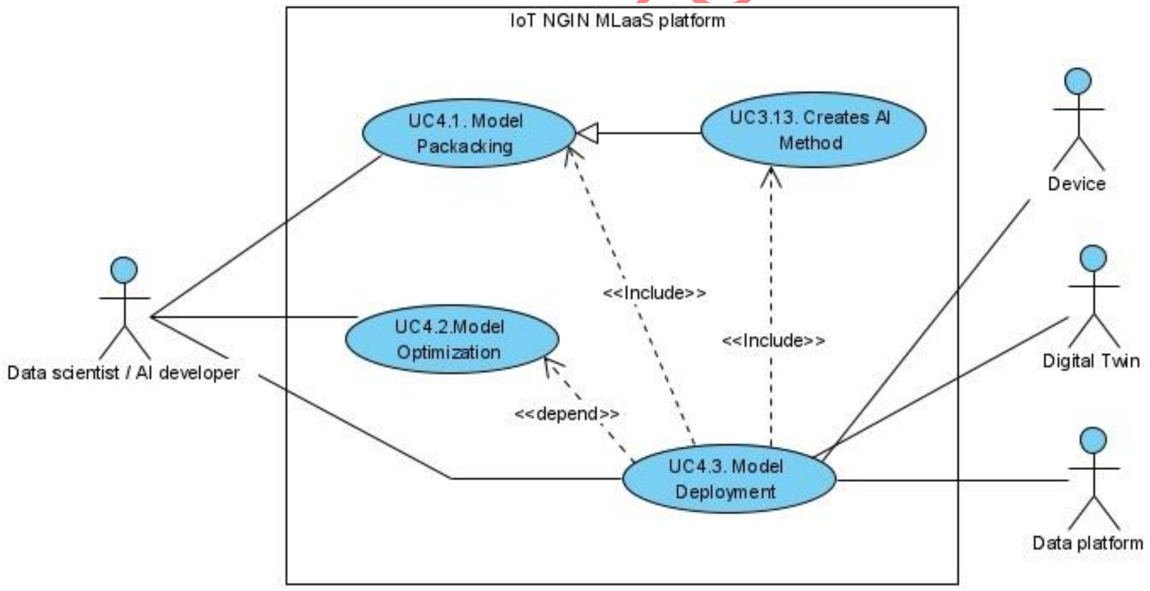


Figure 14: Platform Use Case 4: AI Model Deployment

4.2.3.5 Integration and Model Operation

In Figure 15 we may observe the UML diagram for the Integration and Model Operation platform use case. This use case depicts how the trained ML models are integrated into AI services after an AI method has been created, and how those AI services are deployed for their integration in the living labs or any other third-party platform operating IoT-NGIN. In addition, this platform use case describes how the model operates once it has been deployed. For that, different kind of actors will make use of it, some of them will even monitor its performance in production, as it is described below.

The platform use case is divided into six sub-use cases:

1. **UC5.1 Creates AI Service**: This sub-use case refers to the preparation of an AI service through loading the required libraries and frameworks for using the required AI method. The AI Application developer is in charge of creating this AI service by packaging the required libraries, as well as the AI method and the trained AI model, enabling seamless deployment in any compatible infrastructure.
2. **UC5.2 Deploys AI Service**: Once the AI service has been created, it may be deployed in the target infrastructure by the AI application developer. This use case extends from the UC4.3 Model Deployment, described previously, so that the target infrastructure deployment requirements are properly addressed before running the AI service.
3. **UC5.3 Uses AI Service**: The AI User / Integrator and any client's end-user can use the AI service through this sub-use case. It includes the UC5.4 described next.
4. **UC5.4 Consumes Inferencing API**: This sub-use case offers the deployed AI service API for its usage, either by the Digital Twins and/or IoT Devices deployed in the infrastructure or by external calls from other parts of the platform.
5. **UC5.5 Feeds AI method:** By means of this sub-use case, the Digital Twins and IoT Devices will be able to feed the AI method of the deployed AI service, as input data for the AI model and/or as a request for inference (prediction or classification).
6. **UC5.6 Monitors Performance:** The AI User / Integrator and/or the Data scientist / AI Developer could monitor the metrics of the deployed AI models by means of this sub-use case. The inferencing API will be consumed to obtain the required data for the monitoring.

In this use case the following actors participate:

- **AI application developer**: Person in charge of creating the AI services, by loading the specific AI methods and AI models, and deploying them into the target infrastructure.
- **End-user**: User or client of IoT-NGIN's deployed MLaaS platform that consumes the deployed AI services.
- **Data Scientist / AI developer**: Person that deals with the deployed AI models' monitoring and uses the relevant metrics to perform UC3.7 Continuous Learning on the deployed AI models, if applicable.
- **Digital Twin**: Entity consumer and/or feeder for the AI models.
- **Device**: Real device consumer and/or feeder for the AI models.
- **AI User / Integrator**: Person in charge of ensuring the correct usage of the AI services and monitoring the performance of the deployed AI models in the target infrastructure.

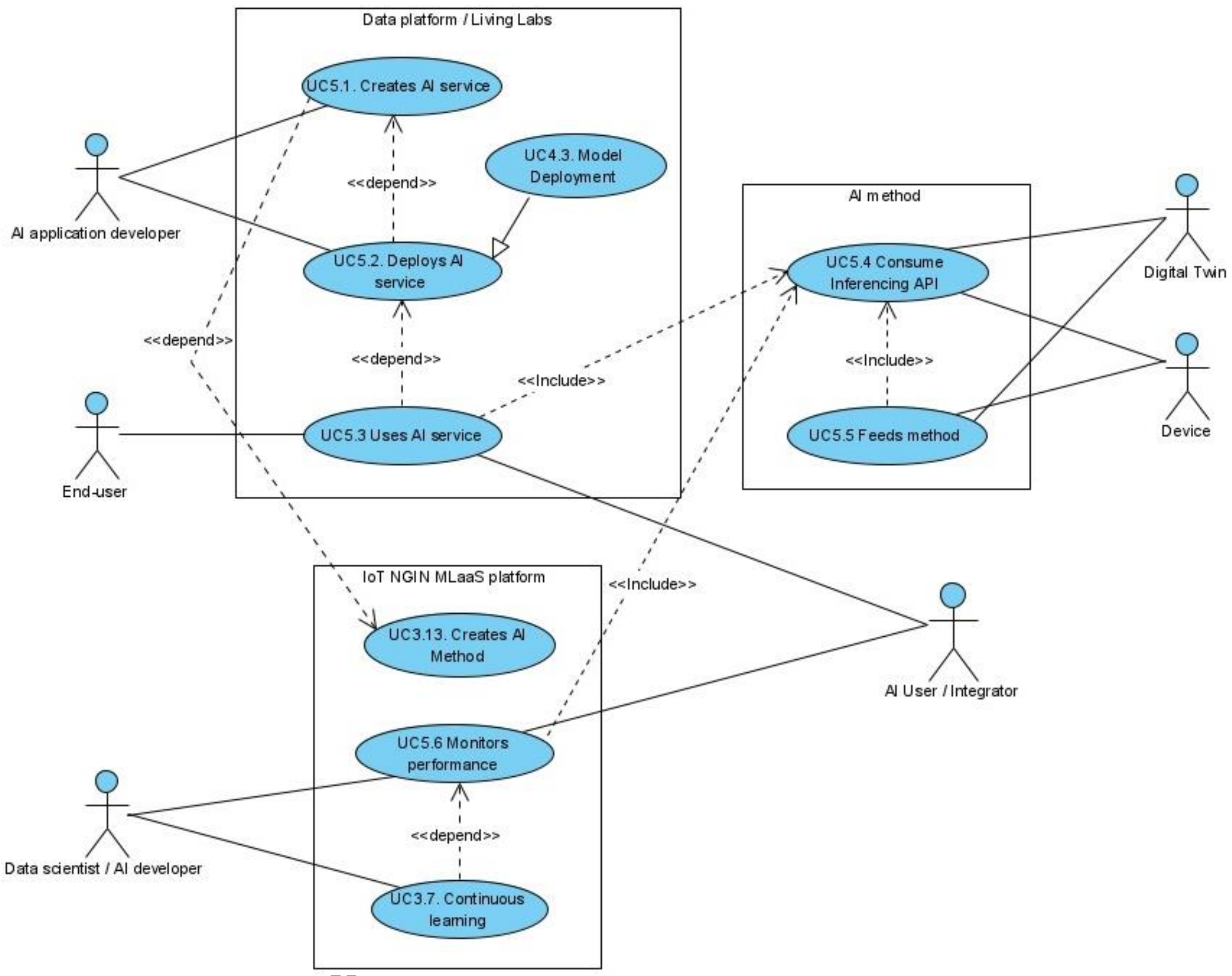


Figure 15: Platform Use Case 5: Integration and Model Operation

4.2.3.6 Model Sharing

Use Case 6 "Model Sharing" deals with making the model available to third-party developers in order to use the stored models in new applications or services or even use a direct API to make predictions using these models. This use case is realized via the following set of sub-use cases.

1. **UC6.1 Publishes model**: The Data Scientist/AI Developer publishes the model, i.e. shares the model with a user or a group of users or publicly, giving them the ability to see and run the model. Specifically, the Data Scientist/AI Developer tags a stable version of their model or API serving this model and it is made available as a new release.
2. **UC6.2 Controls access**: The Data Scientist/AI Developer controls the type of sharing, including "private" sharing with selected users or groups, as well as "public" which makes the model available with no additional access restrictions.

3. **UC6.4 Discovers models**: A Third-party (developer) searches ML models or APIs serving those models, which are available to them, based on their profile or access rights or availability restrictions of the models per se. This Third-party (developer) discovers models of their interest, being provided with data or metadata describing the model.
4. **UC6.5 Gets model**: The Third-party (developer) has the option to get the model of their interest in order to use it, selecting a specific tag (release) of it. This includes downloading the binary model, in order to use it in the AI method, service or application development or a direct API to an AI method that uses this model.
5. **UC6.6 Checkout model**: This use case is "included" in UC6.5 "Gets model". It refers to Third-party (developer) being able to download locally the selected tag of the model or API serving the model.
6. **UC6.3 Monetizes model**: This use case is not included in Use Case 6 "Model Sharing", as it represents an optional case of the Data Scientist/AI Developer monetizing the model. Also, this use case involves the Third-party (developer) paying the specified remuneration fee, as part of the checkout process, after selecting and before downloading/being granted access to the model.

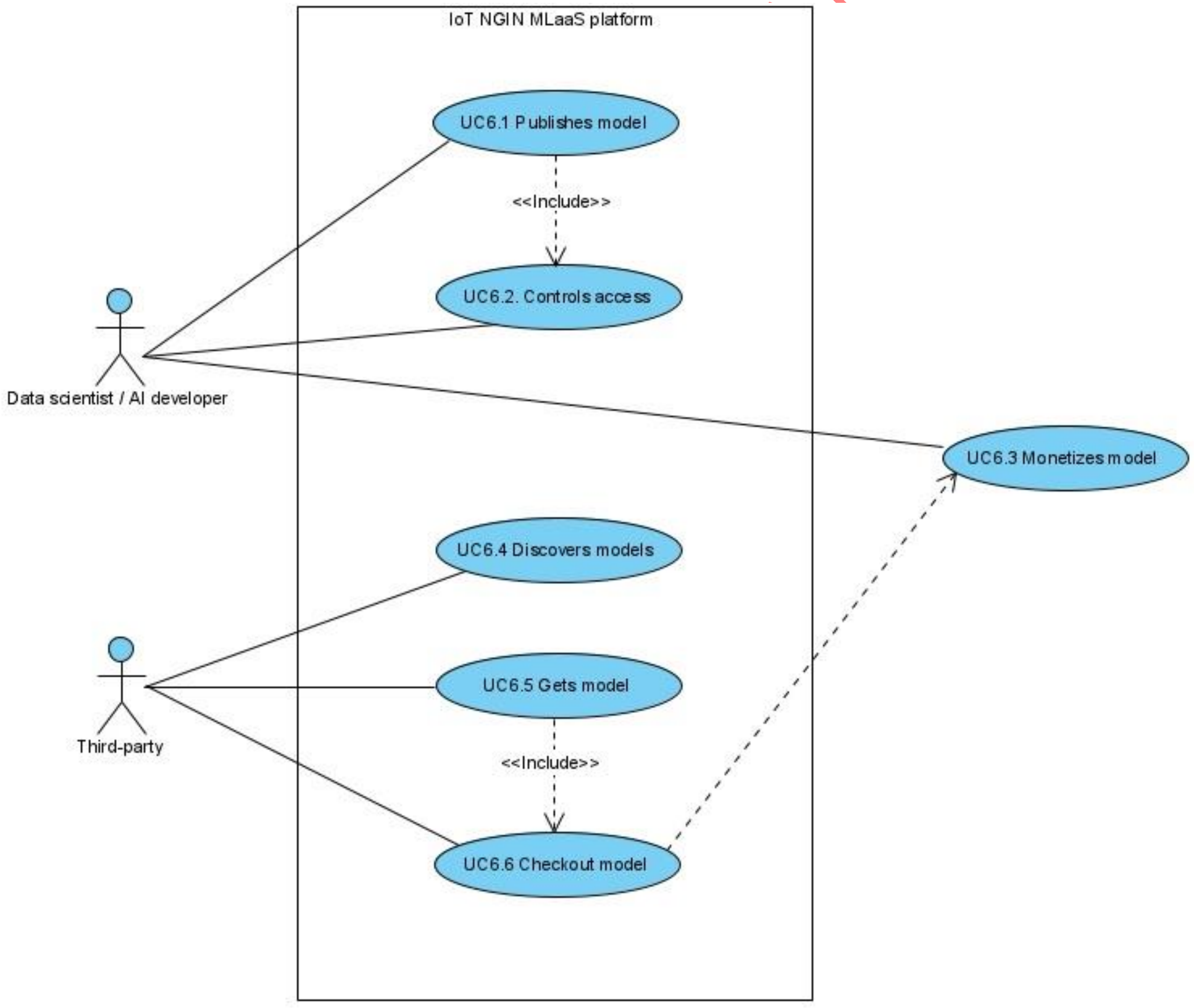


Figure 16: Use Case 6: Model Sharing

4.2.4 Functional and Non-Functional Requirements for MLaaS

In this section, both functional and non-functional requirements are elicited for the IoT-NGIN MLaaS platform, considering the platform use cases specified in section 4.2.3 "MLaaS Platform Use Cases" lists the functional requirements for the MLaaS platform.

Table 30: MLaaS platform functional requirements (1/2)

IoT-NGIN MLaaS Functional Requirements	
Data Storage	The platform should provide Data Storage. The Data Storage should allow for storing: 1. Data from Data Ingestion 2. Data for ML training 3. Data for feature processing
Data Visualisation	The platform should provide an environment to perform data exploration and data visualisation
Data Analytics	The platform should provide an environment for applications to request prediction
Data Processing	The platform should provide an environment to perform data preparation
Data Management	The platform should provide a collection of tools that help control and manage the data storage effectively to achieve centralized data coherence. Data management tools support functions performing control, protect, organize, retrieve, search, data lifecycle, etc. of the data
Feature engineering	The platform should provide an environment to perform Feature engineering
Data collection and ingestion	The platform should allow for IoT devices or Digital Twin to ingest and store data in the Data Storage
Model creation and training	The platform should allow for creating ML model and perform training
Model testing	The platform should allow for testing ML models prior to their deployments
Model Deployment	The platform should provide tools for the deployment of the trained model onto IoT devices or Digital Twin
Model library	The platform should provide a library storage of AI models that can be used as-is or for transfer learning by the platform
Support for Federated Machine Learning	The platform should provide support for Federated Machine Learning
Polyglot trained ML model sharing	The platform should provide polyglot trained ML model sharing component where AI models can be shared with third parties and stakeholders entirely or by applying transfer learning methodologies

Table 31: MLaaS platform functional requirements (2/2)

IoT-NGIN MLaaS Functional Requirements	
Interaction with Digital Twin	The platform should be able to collaborate with a Digital Twin System or meta-level Digital Twins: 1. To get access to data for Federated Machine Learning 2. To send prediction to the Digital Twin 1. To send command to the Digital Twin
Search and Discovery	The platform should allow clients to search for ML models and available (public) data

In addition, Table 32 and Table 34 list the non-functional requirements for the MLaaS platform.

Table 32: MLaaS platform non-functional requirements (1/2)

IoT-NGIN MLaaS Non-Functional Requirements	
Availability	The platform should be able to run 7 days a week, 24 hours a day
Data collection and ingestion protocols	The platform should support open interfaces for data ingestion via REST, MQTT and/or CoAP transfer protocols.
Data ingestion format	The platform should support data representation in the JSON, XML and/or CBOR format
Network communication	Platform support for data ingestion protocols should be independent from the underlying network communication layer, e.g. 4/5G, LTE/NB-IoT, LPWAN, Ethernet, WLAN, etc.
Platform protection	The platform should provide Role Based Access Control (RBAC) for protected access to the platform
Data protection at rest and in transit	The platform should provide secure data at-rest and in-transit. Traffic should be encrypted with TLS or DTLS.
Data sharing	The platform should provide a secure, trusted and controlled way to share data. Sharing should stay within the bounds of security and privacy policies defined by the stakeholders of the data
Data privacy	The platform should provide adequate protection to ensure privacy of the data
Data type	Platform should address data management across a data ecosystem comprising both open and closed data
Provide GPU	The platform should provide graphics processing unit (GPU) capabilities to accelerate ML training
Model Operationalization	The platform should allow adjustment of models to ensure their relevance over time (MLOps)
Management	The platform should allow to be maintained, monitored and managed

Table 33: MLaaS platform non-functional requirements (2/2)

IoT-NGIN MLaaS Non-Functional Requirements	
Scalability	The platform should be scalable to ensure it can grow over time according to the number of requests.
Regulatory requirements	The platform should be compliant according to the data hosted into the platform (GDPR, HIPPA, etc.)
Localization	The platform should support the English language. Depending on the tools used it will possibly support other languages
Implementation	The platform should preferably be installed using Infrastructure as Code (IaC)
Recoverability	The platform should have backup mechanism to be able to recover data and models in case the platform must be reinstalled
Open source	The platform should be based on Open source components
External access	The platform should be able to exchange data with external platforms like public/private cloud, external data provider (weather forecast for example) or social media
DevOps	The platform should provide tools for Continuous Integration & Continuous Deployment (CI/CD) for deployment of the AI models into IoT-NGIN IoT devices

4.2.5 MLaaS Platform Logical view

Considering the MLaaS requirements and the previously described platform use cases, the logical view of IoT-NGIN's MLaaS platform was conformed as it is shown in Figure 17. The components of the platform are not designed to be a software element itself, but rather they represent a set of functionalities that serve a common purpose provided through different software and/or hardware procedures. They are represented in the part of the network (Cloud/Edge) in which they are expected to function. However, some components can be executed and perform tasks along the cloud-edge continuum, thus, they are represented in the middle of the figure. three main blocks can be identified as follows:

- Orange boxes represent the functionalities that deal with AI models: training, deploying, optimizing, and sharing. The following components have been identified:
  - **Model training**: This represents the functionalities related to training an AI model with data handled by the platform in the Data Storage component and represents the use cases defined in Platform Use Case 3: AI modelling. Even though the training of those models is supposed to be federated, the aggregation operations of the federated learning servers can be executed in the cloud-edge continuum, so, in the figure, it appears in the middle of the network. Once an AI model is trained, its weights can be stored in the Models Library for later usage or sharing, or it can be directly deployed with the Model Deployment component. In addition, the Model Training component is

accessible from the Development Environment, and it works in cooperation with the Model optimization component.

- **Models Library**: This component represents the functionalities related to storing the AI models' weights in a database-like object. It will allow to verify the integrity of those models and to prevent disclosing any data related to the training phase, along with the capacity of loading and downloading.
- **Model deployment**: It represents the set of functionalities related to deploying an AI model (comprised of an AI Service) for inference. It also enables the set of APIs that can be used to request inferences and the set of APIs to enable the monitoring of the model's performance. In addition, it will allow using the continuous learning framework if the target environment can take advantage of this functionality. All in all, this component represents the use cases defined in Platform Use Case 4 "AI Model Deployment" and Platform Use Case 5 "Integration and Model Operation". Likewise, the Model deployment component is supposed to operate over the cloud-edge continuum; there will be cases in which the AI Model may be deployed directly on the IoT Devices or in the edge nodes, but in other cases, it may be deployed in the cloud instance of the platform.
- **Model optimization**: With respect to Platform Use Case 4 "AI Model deployment", this component is in charge of performing the required optimizations on the trained AI model to (i) validate the model hyperparameters against the validation data and (ii) to enable an optimal deployment on the target environment (different types of IoT Devices or Edge nodes that have different hardware capabilities). Thus, this component will enable the operation of those models optimally by leveraging a set of techniques implemented for (i) validation of the model's parameters before production deployment, and (ii) adaptation of the model to the hardware of the production environment.
- **Polyglot model sharing**: In addition to the rest of the operations with an AI model, the AI platform will define a component with the functionalities detailed in the Platform Use Case 6 "Model Sharing". It will be in charge of sharing the model with third parties or other external users in different AI frameworks (polyglot).

- Green boxes represent the functionalities to handle the data from the data sources: loading the data according to templates, performing data processing and storing the data in edge nodes. The following component conforms to the data-related functionalities of the platform:
  - **Data Management**: This component will conform to the main functionalities related to managing the data in the platform, allowing to access, view, modify and/or delete specific data to specific users, and to enable the operations in the ways explained in Platform Use Case 1 "Data Acquisition and platform" and Use Case 2 "Data pre-processing", from a common point of view. Since it is comprised of a set of functionalities, this component is represented in the cloud, where the users have access from the Development Environment component.
  - **Template Data Library**: This component enables the acquisition and pre-processing of the data according to some pre-defined templates, i.e. granting the users with pre-existing functionalities and schemes to load and visualize the data or to do some pre-defined operations over them.

- **Data Acquisition**: As defined in Platform Use Case 1 “Data acquisition”, this component will grant the users of the platform to load a dataset (or connect to a data stream) from the Data Storage according to some parameters that can be configured previously. These functionalities can be executed along the cloud-edge continuum.
- **Data pre-processing**: Once a dataset is loaded, the data can be processed in the ways explained in Platform Use Case 2 “Data Pre-processing”, before feeding this data into an AI model. Similar to the Data Acquisition component, these operations can be carried out along the cloud-edge continuum.
- **Data Storage**: This component represents the set of functionalities and capabilities of the platform to store the data locally, close to the edge. Not only that, but it also enables access to this data through a set of APIs or functionalities. The storage of the data in the MLaaS platform is supposed to be only on the edge, in order to preserve privacy and to prevent disclosing this data outside the scope of the local deployment.

- Blue boxes represent the data sources of the platform, that send data in batches or a stream. The following main data sources of the platform have been identified:
  - **IoT Devices**: This represents the data sources of the platform and, possibly, consumers of the AI models.
  - **Digital Twins**: Similar to the IoT Devices, the Digital Twins may access the AI models for inference, and they can feed the Data Storage of the platform.

Last, there is a white-marked component named “Development Environment”. This component will serve as the entry point of the platform for any user where, through a set of APIs and/or a graphical user interface (GUI), it will grant usage of the rest of the platform components and functionalities from a common place.

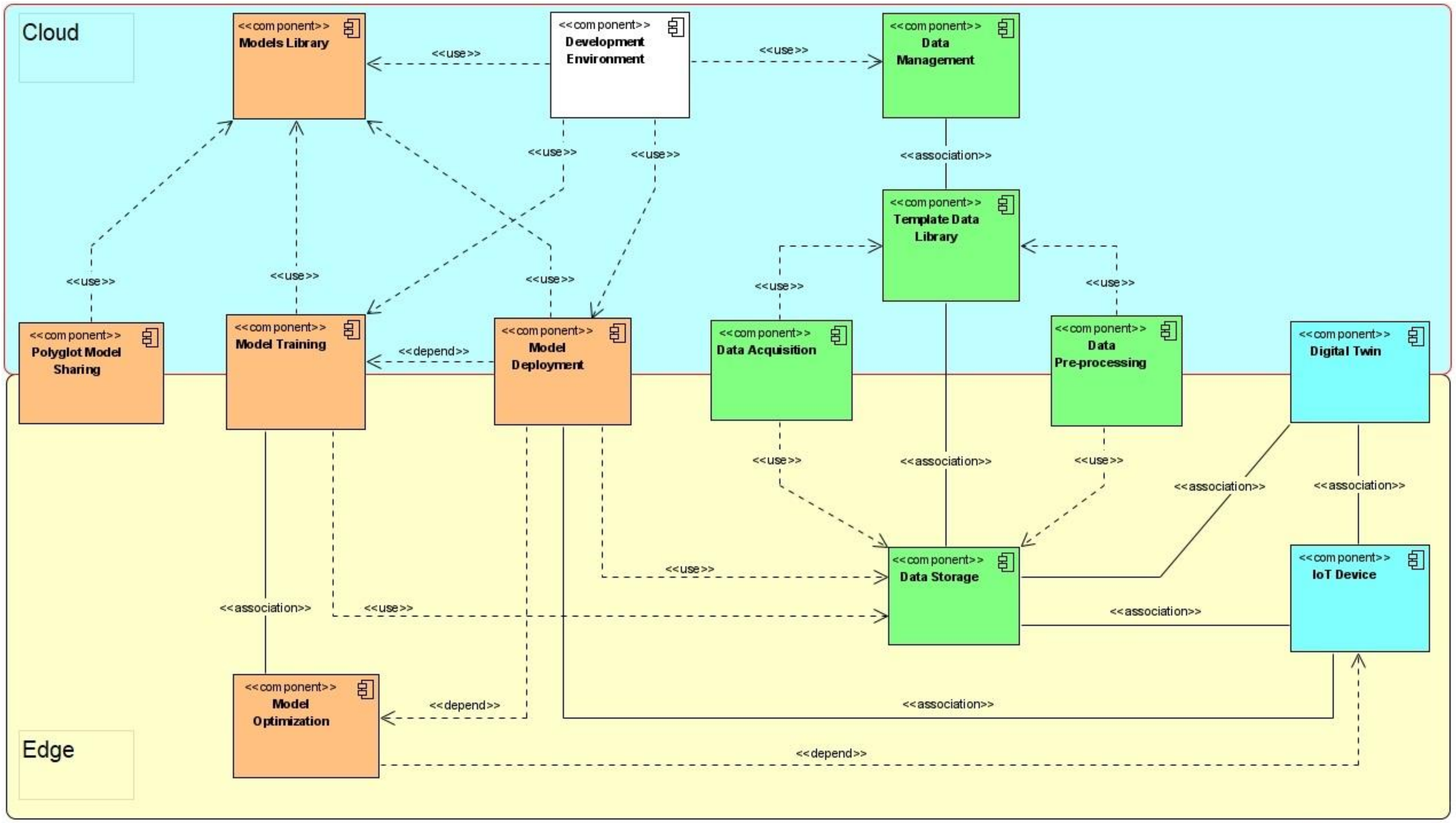

Cloud
<<component>> Models Library
<<component>> Development Environment
<<component>> Data Management
<<component>> Template Data Library
<<component>> Polyglot Model Sharing
<<component>> Model Training
<<component>> Model Deployment
<<component>> Data Acquisition
<<component>> Data Pre-processing
<<component>> Digital Twin
<<component>> Data Storage
<<component>> IoT Device
<<component>> Model Optimization
<<use>>
<<depend>>
<<association>>
Edge

Figure 17: MLaaS Platform Components - Logical view

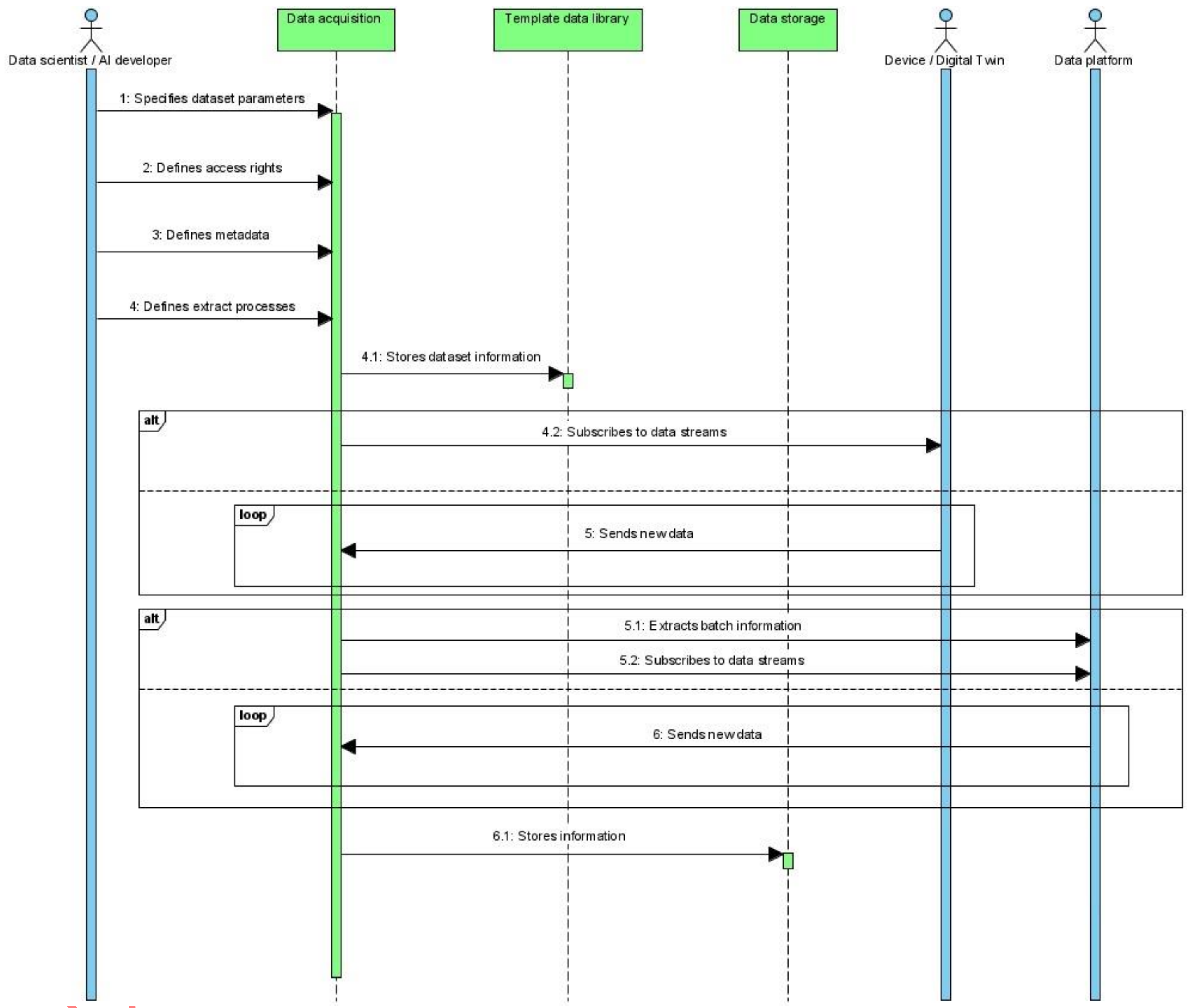
4.2.6 MLaaS sequence diagrams

In this section, the sequence diagrams of the MLaaS platform are described following the UML pattern. The main purpose of these diagrams is to link each platform use case with the platform's logical view in Figure 17, showing how the actors and components interact arranged in a time sequence way.

4.2.6.1 Data acquisition

Figure 18 depicts the sequence diagram for the data acquisition platform use case. The workflow is as follows:

1. The first actor involved in this diagram is the **Data Scientist / AI Developer**, who interacts with the **Data Acquisition** component to perform some preliminary operations on the dataset to be stored or processed, such as setting the parameters for the following extraction processes.
2. Once the preliminary parameters are set, the **Template Data Library** is used to store this information for later usage when the real data is received. There are two ways of getting this data: either directly from the **Devices** or the **Digital Twins** as a stream, or from the **Data Platforms**, which may also be as a stream or in batches.
3. Finally, after the successful extraction of this data, it is stored in the **Data Storage** component, which should be local/close to the edge.



Data scientist / AI developer
Data acquisition
Template data library
Data storage
Device / Digital Twin
Data platform
1: Specifies dataset parameters
2: Defines access rights
3: Defines metadata
4: Defines extract processes
4.1: Stores dataset information
alt
4.2: Subscribes to data streams
loop
5: Sends new data
alt
5.1: Extracts batch information
5.2: Subscribes to data streams
loop
6: Sends new data
6.1: Stores information

Figure 18: Sequence diagram for Data acquisition use case

4.2.6.2 Data pre-processing

Figure 19 depicts the Sequence Diagram for the Data preprocessing platform use case. The workflow is as follows:

1. The first actor involved in this diagram is the **Data Scientist / AI Developer**, who connects to the **Development Environment** of the platform before accessing tools and compute resources required for the data pre-processing. The Development Environment will check if the requester has adequate rights to use the platform.
2. Once authorized in the platform, the Data Scientist / AI Developer will start performing **Exploratory Design Analysis**. He will load the data s/he wants to explore from the platform's Data Storage. The platform will check if the Data Scientist / AI Developer has the rights to access these data. The Data Scientist / AI Developer will use data analysis tools and techniques available in the platform to investigate, analyse, and summarize the main characteristics of the dataset. This can be an iterative process where the Data Scientist / AI Developer refines her/his exploration of the data via several iterations. Once done the Data Scientist / AI Developer will save the updated data if they have been modified in the process.
3. Once the Data Scientist / AI Developer has completed the Data Exploration, s/he will perform **data wrangling** to possibly restructure and transform data into a standard format. She ornHe will load the data saved in the previous step and will perform **data cleaning** and **data normalization**. Updated data are saved to the data storage.
4. A second actor in this diagram is a third party that could want to use the data from the platform. This third party will request the data prepared by the Data Scientist / AI Developer and, if authorized, s/he will download the data from the data storage.
5. The Data Scientist / AI Developer may want to **augment the data** with external data from a third-party source. She or He will request the third-party data and, if authorized, will load the data in the development environment. This can be an iterative process where the Data Scientist / AI Developer augments the data via several external sources.
6. Once all data are fine, the Data Scientist / AI Developer will **annotate the data** to prepare the raw data to be consumed by the platform for ML training. Example of annotation would be an image, video, text, audio annotation and 2D/3D Bounding Box. Annotated data are saved in the data storage for future used for the ML training process.
7. In the case, the full data are big for long term storage or it is inconvenient to work with data in full, it could be faster to work with a compact summary. The Data scientist / AI developer may decide to **create a smaller data set** with the same structure as the full dataset. The Data scientist / AI developer will perform data sampling and will store the sampled data on the data storage for future use for ML training for example.
8. Once the data are ready, the Data scientist / AI developer will split the data into a training and a test set.

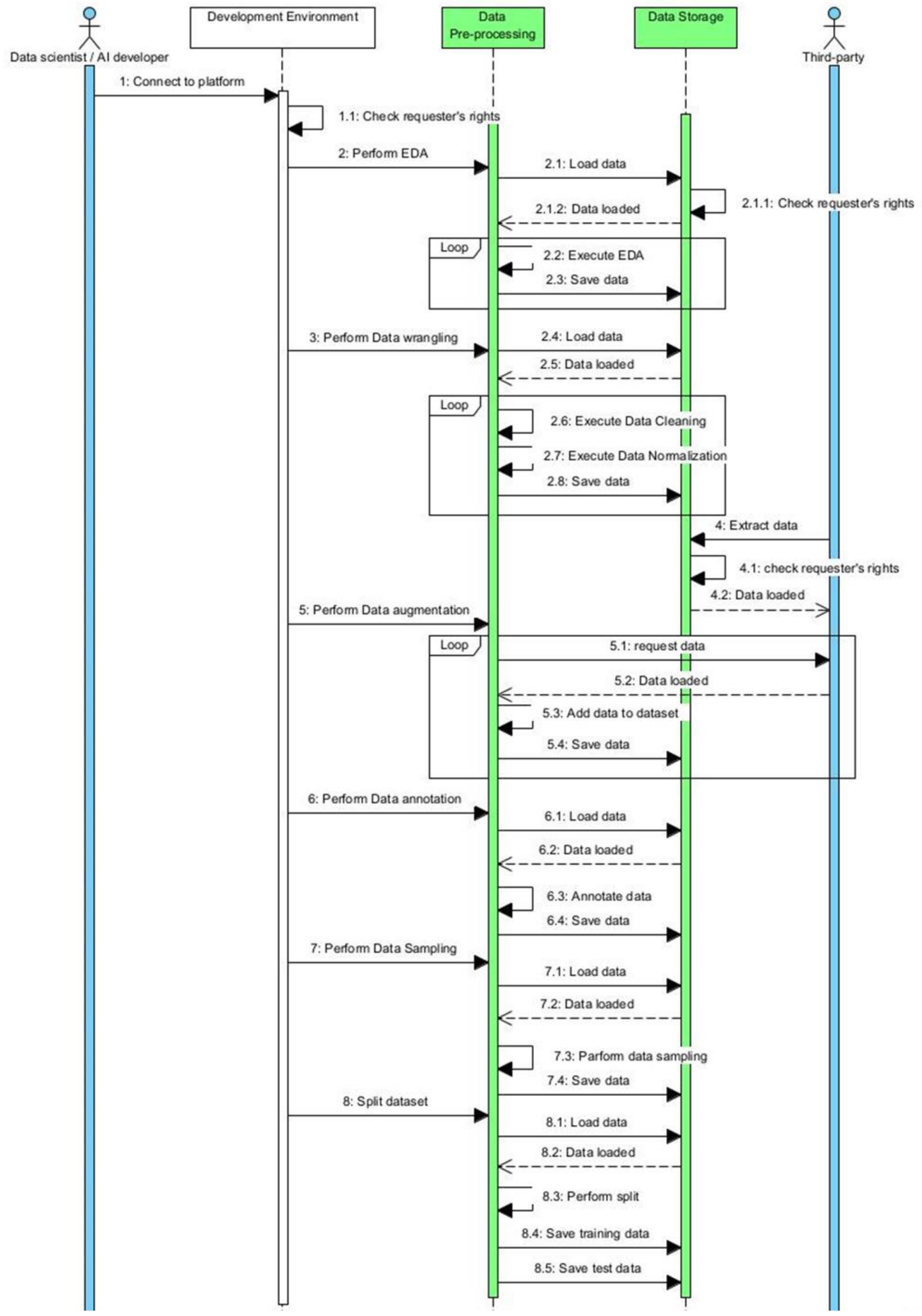


Figure 19: Sequence Diagram for Data pre-processing use case

4.2.6.3 AI Modelling (edge)

The AI modelling process describes the activities following the Data Preprocessing tasks and focused on the delivery of an ML model which is appropriate for the predictive modelling problem in question. In IoT-NGIN, the ML modelling process includes the following steps:

- Feature engineering,
- Model selection
- Model training
- Model validation
- Model tuning
- Model saving

The sequence of processes through AI Modelling that takes place in IoT-NGIN is analysed in the sequence diagram depicted in Figure 20 and Figure 21. The process starts with "feature engineering", which refers to the definition of appropriate properties or calculations which can be used to extract useful insights over the given data set for a given predictive problem. The *Data scientist/AI developer* interacts with the *Development Environment* in order to define and apply a new feature engineering process over the dataset already acquired and pre-processed, e.g. via the *Data Acquisition* and *Data Preprocessing* processes. The feature engineering process is handled by *Data Management*, which undertakes to store the process in the *Template Data Library* and apply it on the dataset loaded by the *Data Storage*. After *Feature Extraction* is finished, the feature metadata are stored in the *Data Storage* and are also provided back to the *Data scientist/AI developer.*

The next step includes the processes that lead to model selection, i.e. the model definition, training and assessment. First, the model hyperparameters are defined and the candidate model is stored in the *Model Library*. Then, the *Data scientist/AI developer* calls model training via the *Development Environment* and *Model Training* loads the model from the *Model Library*, as well as the data and features from the *Data Acquisition* and perform the model training. The trained model is stored in the *Model Library* and the result is returned to the *Data scientist/AI developer*. Then, validation takes place, after the *Data scientist/AI developer*'s call, which is passed to the *Model Optimization*. This component performs the validation of the model -taken from the *Model Library*- and the validation dataset -acquired via *Data Acquisition*. The validation result is returned to the *Data scientist/AI developer*. In case the validation has failed, s/he performs model tuning and the updated model hyperparameters are saved in the *Model Library*.

If validation has been successful, the *Data scientist/AI developer* selects and saves the trained model in the *Model Library*. Now, the model is available for developing an AI method, using it for inference.

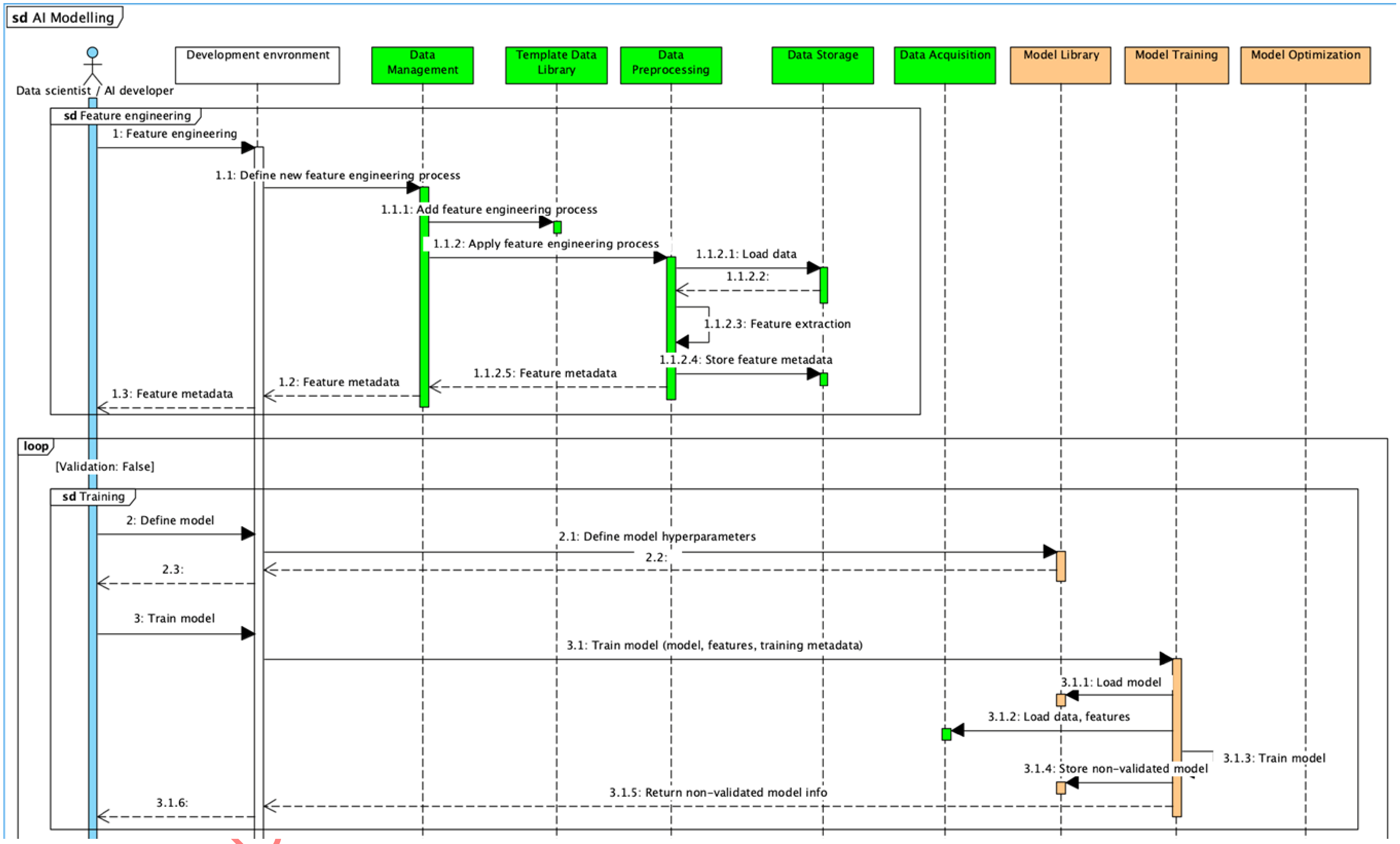

sd AI Modelling
Data scientist / AI developer
Development envronment
Data Management
Template Data Library
Data Preprocessing
Data Storage
Data Acquisition
Model Library
Model Training
Model Optimization
sd Feature engineering
1: Feature engineering
1.1: Define new feature engineering process
1.1.1: Add feature engineering process
1.1.2: Apply feature engineering process
1.1.2.1: Load data
1.1.2.2:
1.1.2.3: Feature extraction
1.1.2.4: Store feature metadata
1.1.2.5: Feature metadata
1.2: Feature metadata
1.3: Feature metadata
loop
[Validation: False]
sd Training
2: Define model
2.1: Define model hyperparameters
2.2:
2.3:
3: Train model
3.1: Train model (model, features, training metadata)
3.1.1: Load model
3.1.2: Load data, features
3.1.3: Train model
3.1.4: Store non-validated model
3.1.5: Return non-validated model info
3.1.6:

Figure 20: Sequence diagram for AI Modelling use case (1/2)

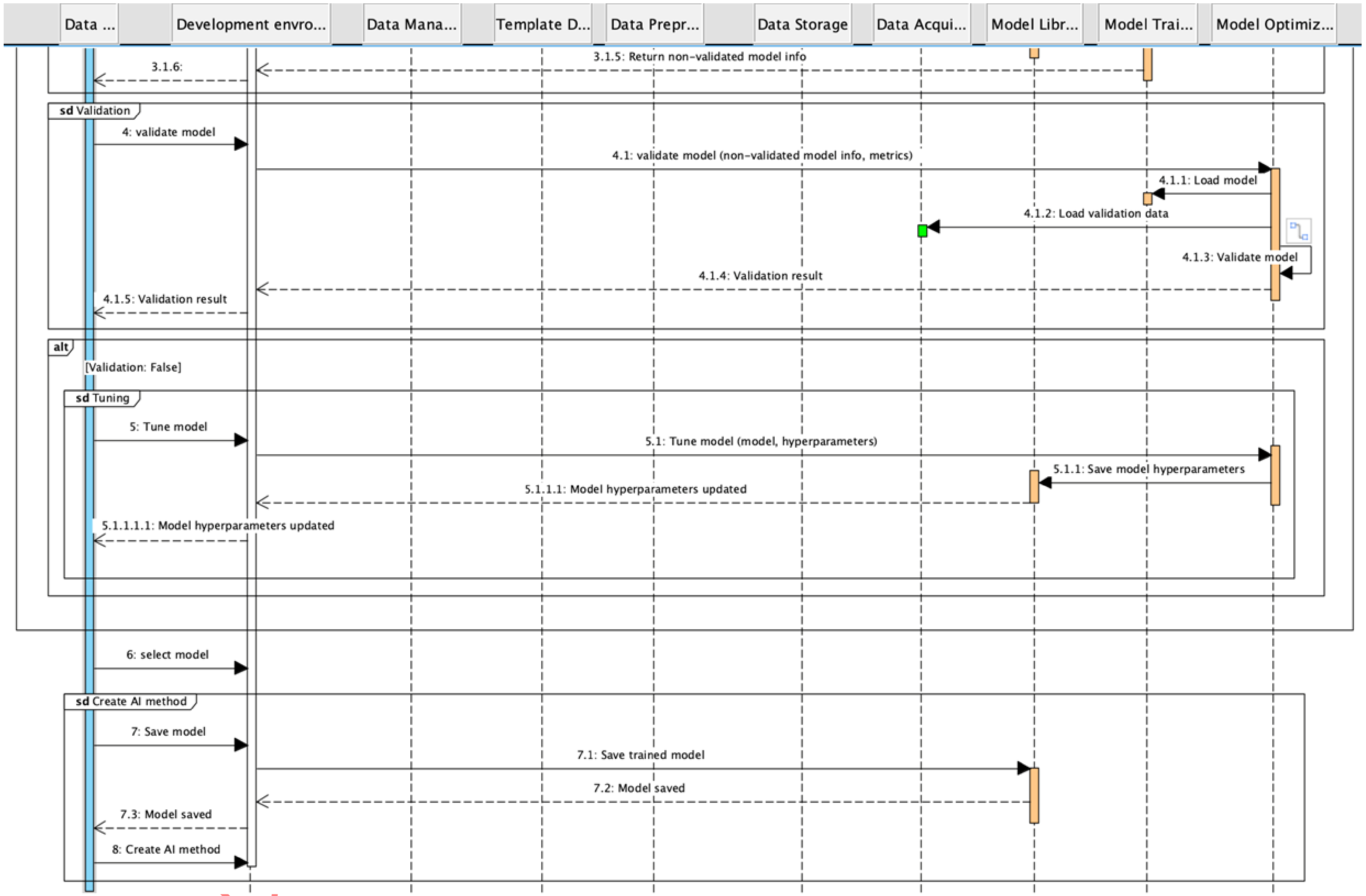


Figure 21: Sequence diagram for AI Modelling use case (2/2)

4.2.6.4 AI Model deployment

Figure 22 depicts the sequence diagram for the AI Model Deployment use case. The workflow is as follows:

1. The **Data Scientist / AI Developer** starts by loading an AI method that has been previously created.
2. The Data Scientist / AI Developer **packages the AI model** in a standard format and with a description. The packaged model is saved in the Model library so it can be shared and reused.
3. Optionally, The Data Scientist / AI Developer may need to **optimize the model** for a specific use case like deployment on low-powered devices or constrained hardware. Once optimized the model is saved in the Model Library.
4. Once the model is ready to be deployed, the Data Scientist / AI Developer will **deploy the model** onto the physical device or the digital twin or another data platform. Optionally (but preferably), the Data Scientist / AI Developer could use the **CI/CD system** of the platform to deploy the model onto the physical device or the digital twin or another data platform by updating a CI/CD repository associated with the device, the Digital Twin or the data platform.

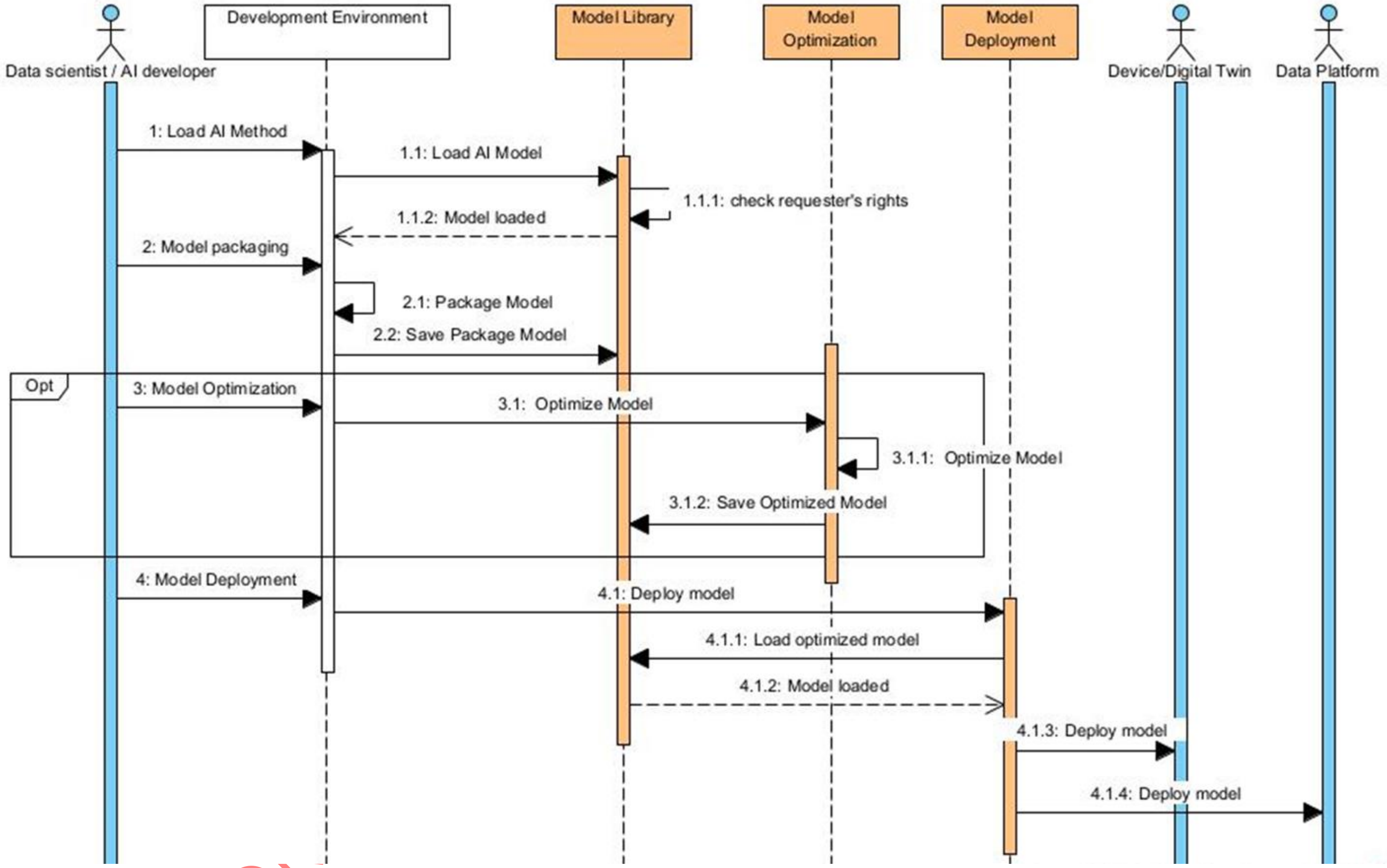


Figure 22: Sequence diagram for Model Deployment use case

4.2.6.5 Integration and Model Operation

Figure 23 depicts the Sequence Diagram for the Integration and Model Operation platform use case. The workflow is as follows:

5. The **Data Scientist / AI Developer** starts using the **Development Environment** to create an AI service. It includes the AI model, loaded from the **Models Library**, and the AI Method corresponding to that model. Once the AI model is loaded and the AI method is prepared, the AI service is created encapsulating them through the **Development Environment**.
6. Once an AI service is ready, it is deployed by the **Model Deployment** component, which is connected to the corresponding **Data Storage**. The AI service deployment takes place, according to the **Data Scientist / AI Developer**'s input, in the Living Labs of IoT-NGIN or any client Data Platform.
7. The AI service can be operated in different ways. However, in order to do so, the AI model needs some input data, which can be obtained either directly from the **Devices** or **Digital Twins**, or via requests to the **Data Storage**. Similarly, the **Devices** and **Digital Twins** can get the inference either directly from the AI Service, or via reading the stored inference in the **Data Storage**.
8. In addition, as the inferences are stored in the **Data Storage**, the AI model can be monitored by the **Data Scientist / AI Developer** and the **AI User / Integrator**.
9. Last, the **AI User / Integrator** can make use of the AI Service, similarly, feeding or consuming the AI model in order to get inferences.

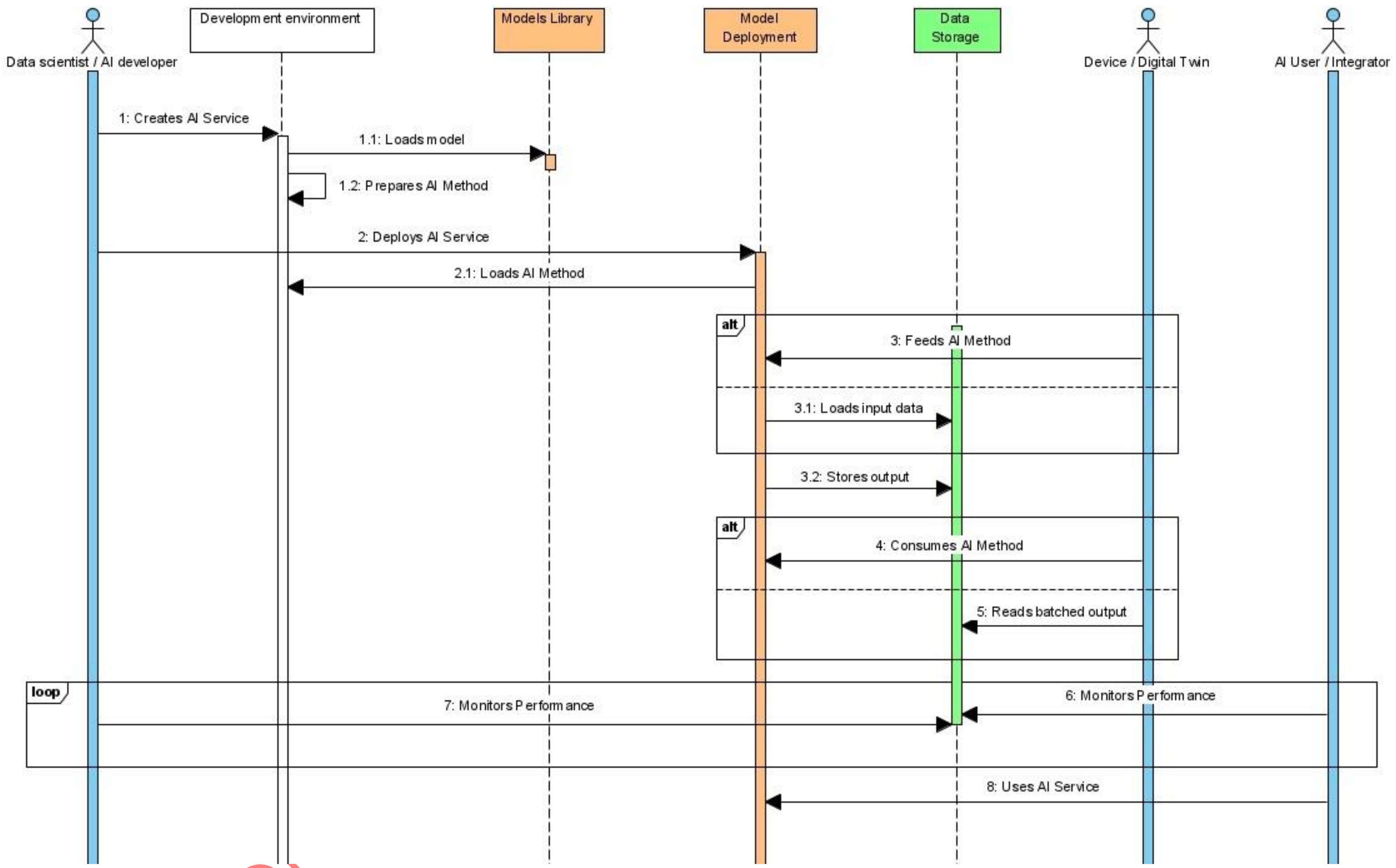


Figure 23: Sequence diagram for Integration and Model Operation use case

4.2.6.6 Model Sharing

The Model Sharing processes ensure that a published model will be available to third parties for use. Model Sharing in IoT-NGIN is analysed via the processes depicted in the sequence diagram of Figure 24. First, the *Data scientist/AI developer* publishes their model via the *Development Environment*. Accordingly, the model metadata referring to the sharing options of the model, such as sharing scope (public/private) and permission, are updated in the *Model Library*.

Some *Third-party* wishing to use a pre-trained model searches for models through the *Development Environment*. The request is communicated to the *Model Library*, which calls *Polyglot Model Sharing* for retrieving the models from the *Model Library*. The call returns the list of models, potentially filtered by the underlying technology, as there is polyglot support.

Then, the *Third-party* selects a specific model version, again via the *Polyglot Model Sharing*. Last, the *Third-party* checks out the model, i.e. they download locally the model from the *Model Library*.

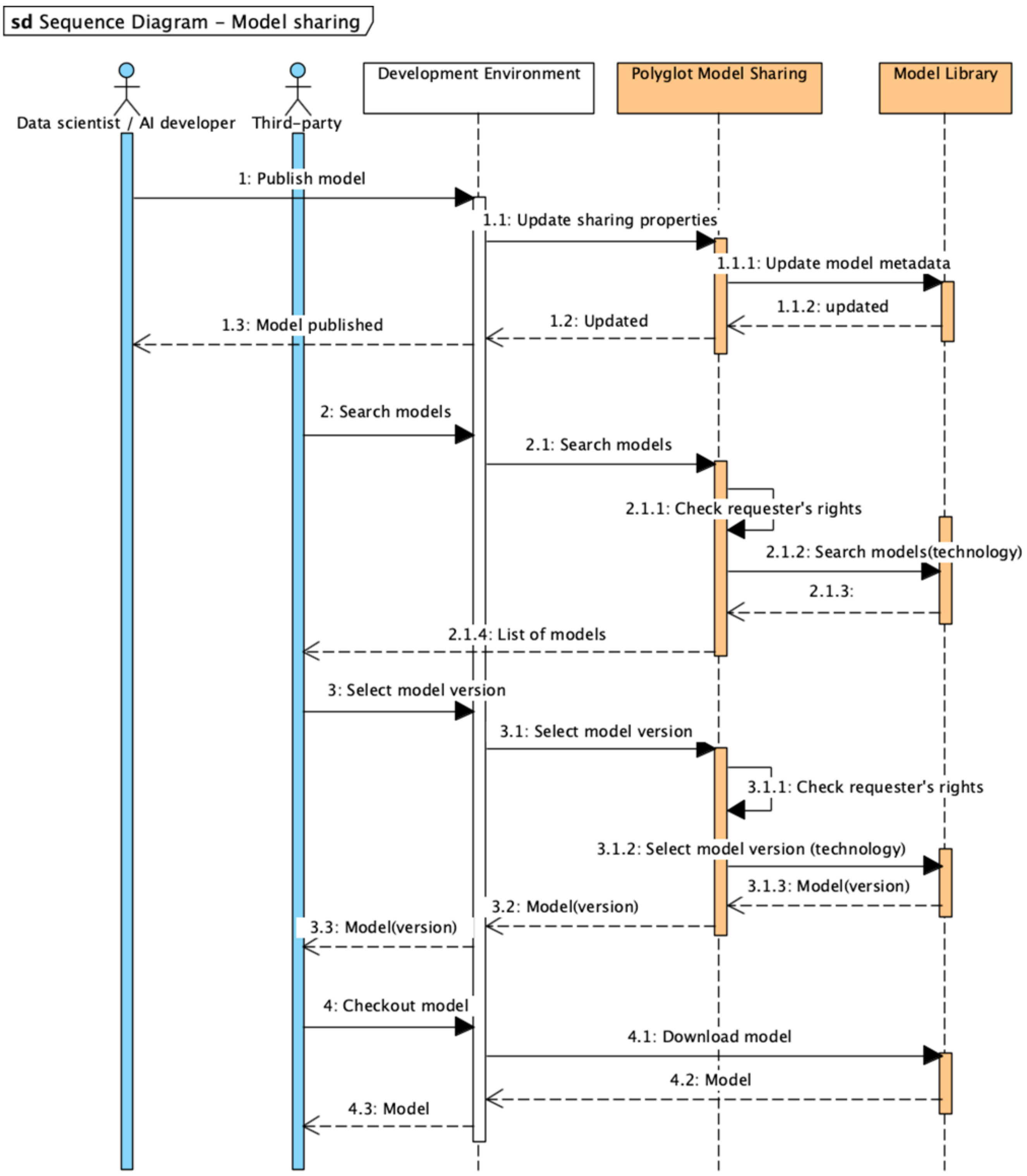


Figure 24: Sequence diagram for the Model Sharing use case

4.3 Data Storage and Data Management

The European Big Data Value Strategic Research and Innovation Agenda (SRIA), published in October 2017, sets up the basis for the European vision of the complete data value chain, proposing the Big Data Value Reference model included in Figure 25. The figure also shows the mappings with IoT-NGIN MLaaS and the main areas where WP3 will work.

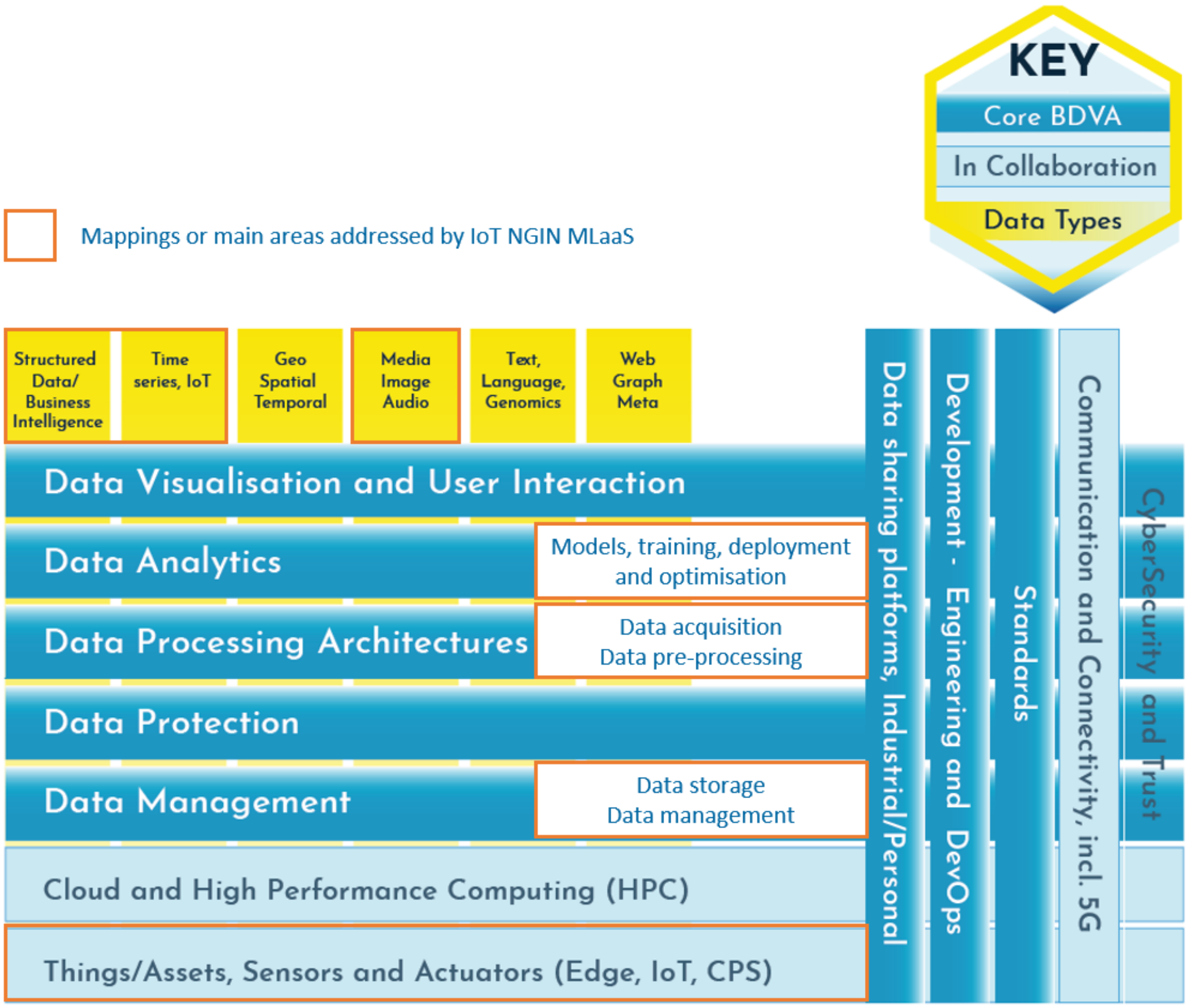

KEY
Core BDVA
In Collaboration
Data Types
Mappings or main areas addressed by IoT NGIN MLaaS
Structured Data/ Business Intelligence
Time series, IoT
Geo Spatial Temporal
Media Image Audio
Text, Language, Genomics
Web Graph Meta
Data Visualisation and User Interaction
Data Analytics
Models, training, deployment and optimisation
Data Processing Architectures
Data acquisition
Data pre-processing
Data Protection
Data Management
Data storage
Data management
Cloud and High Performance Computing (HPC)
Things/Assets, Sensors and Actuators (Edge, IoT, CPS)
Data sharing platforms, Industrial/Personal
Development - Engineering and DevOps
Standards
Communication and Connectivity, incl. 5G
CyberSecurity and Trust

Figure 25: Big Data Value Reference Model

The horizontal components of the reference model are used to identify the foundational aspects that must be taken into account during the complete data management life cycle while the vertical elements depict cross-cutting topics and/or non-technical issues. As can be derived, it also includes relationships with other technologies like IoT, High-Performance Computing or 5G connectivity which are relevant for the IoT-NGIN project.

4.3.1 Horizontal concerns

- **Data visualisation and user interaction:** presentation of the data so that they can be explored and understood by the final user or by the scientist. Interfaces must be able to represent huge amounts of multidimensional data in an intuitive fashion addressing the requirements of the users. Data visualisation techniques include data discovery, interactive and collaborative interfaces, interactive visual data exploration, etc.
- **Data analytics** that extract value from the data deluge generated by the digital platforms and solutions. Analytics cover areas like near-real-time interpretation applying knowledge-based analysis, models for validation of trustworthiness of datasets and data sources, advanced business analytics and intelligence, the usage of Machine Learning to obtain predictive and prescriptive analytics, exploiting the potential of HPC infrastructures through High-Performance Data Analytics (HDPA) and the creation of scalable frameworks for quality-aware processing considering batch and streaming information and distributed architectures.
- **Data processing architectures** that rely upon hardware infrastructures composed of heterogeneous devices and resources and which deal also with heterogeneous data, having to satisfy functional and non-functional requirements for a potentially high number of users, including components and technologies for processing data-in-motion and at rest. It considers specifically the need to adapt data processing architectures to environments including IoT devices and edge computing resources or executing big data workloads that require exploiting HPC resources or specialised hardware accelerators.
- **Data protection,** which implies taking care of sensitive information and compliance with regulations. The application of techniques to protect data privacy must preserve at the same time the usefulness of the data. Data protection covers topics like usage control, auditability, scalable anonymisation techniques, dealing with heterogeneous data, etc.
- **Data management:** semantic interoperability between heterogeneous data types collected and shared by different sources or platforms, harmonization of data formats and models, multilingualism, analysis of data quality and robustness, data lifecycle management and traceability.
- **Cloud and High-Performance Computing (HPC):** convergence between Big Data and HPC to make it possible to run more computationally intensive applications including deep learning workloads. BDVA is exploring this trend utilizing a close collaboration with ETP4HPC and EOSC.
- **IoT, CPS, Edge and Fog Computing.** IoT devices and cyber-physical systems have become a major source of information during the last years thanks to the progressive deployment of sensors and actuators in a wide variety of application domains. Although many of these devices are quite constrained in terms of hardware capabilities, some of them provide also enough computational power to allow running processing tasks directly at the edge or fog tiers.

Table 34: Mapping with Big Data Value Reference Model horizontal concerns

Big Data Value Reference Model	**IoT NGIN MLaaS platform**
IoT, CPS, Edge and Fog Computing	BDVA is currently pushing a close collaboration with AIOTI for this area. IoT NGIN will strongly contribute to this area through the project's results and the demonstration in the living labs.
The Cloud and High-Performance Computing (HPC),	Related to IoT NGIN, HPC computing is not within the scope of the project.
Data management	IoT NGIN MLaaS will not be focused on data management although some tools will be integrated for data exploration and analysis.
Data protection	IoT-NGIN MLaaS will implement privacy-preserving Federated Learning, new functionality that will suppose an important step forward for the training of deep neural networks directly at the edge without having to exchange personalised datasets containing sensitive information.
Data processing	The project will specifically aim to innovate in this area since IoT NGIN MLaaS will be demonstrated in several use-cases that require ingest and process heterogeneous streaming information coming from IoT devices and smart objects. The resulting ML models will be optimised and deployed directly at different levels of the edge tier.
Data analytics	IoT-NGIN will address the complete life cycle of Machine Learning models from the data collection to the final deployment and inferencing, considering the specific needs of IoT-based systems and decentralised architectures including edge computing resources. The project will provide new mechanisms to enhance the models' training process, i.e. self-learning, federated learning.
Data visualisation and user interaction	Although IoT-NGIN MLaaS may integrate some well-known libraries and tools for visual data exploration, it is not one of the priorities of WP3.

4.3.2 Vertical concerns

- **Data sharing platforms, industrial/personal:** the development of applications and services exploiting the data from a single individual or company is not enough anymore. In many use-cases, there are complex ecosystems with multiple stakeholders that establish close and symbiotic relationships. Also, the information collected from one actor can be relevant to other ones. Thus, data sharing and trading are going to become essential enablers for the development of data-driven systems in the near future. For industrial data platforms, new solutions should be able to guarantee secure and sovereign data exchange and monetisation. International Data Spaces (IDS) reference architecture has become a de-facto standard for the implementation of interoperable embryonic data spaces. Regarding personal data platforms, the major barrier is the protection of privacy and avoiding abuses by companies. Personal data spaces should enable citizens to maintain complete control of the usage that third parties do of their data, being fully compliant with the provisions of the General Data Protection Regulation (GDPR).

- **Development – engineering and DevOps:** Particularization of DevOps environments to the specific needs of data scientist and data engineers, new testing paradigms to increase the level of confidence of the resulting systems and to provide enough quality during operation in production environments. DevOps tools must also guarantee the productivity of the development and operation teams.
- **Standards:** BDVA SRIA proposes to rely on already existing standards and also to support them with the outcomes of the projects under its umbrella and the different task forces, e.g., ETSI, ISO, IEEE, CNELEC, etc. The Big Data Reference Model is already aligned with the ISO Big Data Reference Architecture described in ISO IEC JTC1 WG9 20547-3 and it serves as the common ground for most European research projects that address Big Data issues.
- **Communication and connectivity:** 5G is identified as the enabling technology that will provide communication and connectivity supporting the needs of BigData and AI applications. In addition, network management and automation could be improved also with these technologies, for instance, through the usage of machine learning to predict the status of the infrastructures and implement smart orchestration schemas.
- **Cybersecurity and trust:** as it happens with the communications area, cybersecurity and BigData can benefit from each other. On one hand, data and metrics collected from all systems' components, interfaces and communications links can be used to detect threats, attacks and anomalies. Even research has been done about their prediction which would result in safer and robust systems, which is required in many critical applications. At the same time, cybersecurity and privacy must be considered holistically by design, especially in those cases where personal or sensitive data is being collected and processed. Last but not least, the progress done during the last years in Artificial Intelligence techniques are enabling also new opportunities for adversarial attacks or data poisoning. Again, requirements identified concerning trustworthy AI must be enforced and even crystallized in new legislations and regulations.

Table 35: Mapping with Big Data Value Reference Model vertical concerns

Big Data Value Reference Model	**IoT NGIN MLaaS platform**
Standards	Standardisation will be explored through the activities of WP8: Impact Creation and Outreach.
Communication and Connectivity	The relationship between IoT NGIN MLaaS and communication technologies will be done through the interaction between WP2: Enhancing IoT Underlying Technology and WP3: Enhancing IoT Intelligence.
Cybersecurity	The functionalities and services provided by IoT NGIN MLaaS will be used also by WP5 to mitigate poisoning attacks in IoT devices and to detect adversarial attacks. Federated Learning will be the main asset to innovate in the level of cybersecurity and trust of IoT-based systems.
Engineering and DevOps for building Big Data Value systems	The vision for DevOps and development methodologies will be explored in collaboration with WP6: IoT-NGIN Integration & Laboratory evaluation.
Marketplaces, Industrial Data Platforms and Personal Data Platforms	Data sharing between several personal or industrial platforms is not within the scope of the IoT NGIN project.
Data types	BDV reference model identifies 6 types of BigData: 1. Structured data. 2. Time-series data. 3. Geospatial data. 4. Media, image, video and audio. 5. Text including genomics representations 6. Graph data, Network/Web data and Metadata. IoT NGIN will deal mainly with the first four data types as has been explained in subsection 3.3. Nevertheless, the platform will be applicable also to the rest of them.

5 Machine Learning, Deep Learning and Reinforcement Learning in IoT-NGIN

So as to enhance the underlying IoT intelligence, in addition to implementing privacy-preserving federated ML, IoT-NGIN plans to provide innovation in the AI and IoT world from two more perspectives: (i) enhancing the model's training processes and (ii) improving the resulting models automatically. Undoubtedly, this supposes a challenge due to the high innovation expected in linking AI methodologies with the underlying IoT technologies. Even though the state-of-the-art concerning these two topics is already stocked in solutions, it is significantly hard to find real-world implementations. In this regard, throughout this section, firstly, it will be provided with an overview of Machine Learning techniques and, secondly, it will be described how the IoT-NGIN project plans to provide a solution to the enhancement of the intelligence over the IoT landscape, considering the use case needs described in Section 3 and the current state-of-the-art solutions, that will help in the development of the MLaaS components in the upcoming months.

5.1 Overview and state-of-the-art of Machine Learning techniques

Machine Learning is a scientific discipline within Artificial Intelligence that allows creating systems that learn automatically from data. Machine Learning algorithms can analyse large datasets, identify patterns and extract knowledge that can be used then for different tasks. Two main groups of ML algorithms can be identified depending on the type of learning: supervised and unsupervised.

Supervised algorithms train models relying on labelled datasets, including features (i.e. input attributes) and labels (i.e. the output that will be predicted [12], [13]. Since the difference between the real value and the prediction can be obtained, it is possible to minimize the error. The models resulting from the training process are then able to obtain forecasts for new values. Supervised machine learning comprises classification where the goal is to predict to which category a new sample belongs, and regression, where continuous variables must be used as input. Examples of supervised learning algorithms are linear or logistic regression, decision trees, random forests, support vector machines (SVMs), Naïve Bayes and neural networks.

Unsupervised algorithms infer relationships between input data [14], [15]. Clustering algorithms are a clear example of this method that group homogeneous elements in a set of clusters so that the degree of correlation between members of the same cluster is high and low inter-cluster. The most common algorithms are k-means clustering, PCA (principal component analysis), hierarchical clustering analysis (HCA), probabilistic clustering based on Gaussian Mixture Models (GMMs) or autoencoders.

A variant of these two main groups is semi-supervised ML, where algorithms learn from datasets that are partially labelled [16], [17]. This, it is possible to reduce the effort and cost of the labelling, which is one of the main barriers for the development of some ML-based applications that need to deal with the huge amount of annotated data. Two main approaches can be also found within semi-supervised ML: semi-supervised classification which enhances supervised classification and semi-supervised clustering for obtaining better-define clusters than just using pure supervised classification.

Deep Learning is an evolution of ML technologies that replicate the working principles of the human brain. For that reason, its models are based on neural networks that can work with multidimensional data and execute complex tasks with high accuracy on top of Graphical Processing Units (GPUs). For instance, Convolutional Neural Networks (CNNs) are commonly used for computer vision, while Recurrent Neural Networks (RNNs) have demonstrated great potential in speech recognition. Transformers are also becoming more and more popular due to the capacity to solve problems that imply the transformation of an input sequence to an output one.

Reinforcement learning systems are based on algorithms that learn how to reach a certain goal through positive or negative incentives defined by the designer, following a cause and effect approach [18]–[20]. The advantage of RL is that it does not need to collect training datasets in advance, but the resulting models can learn in an online way.

Transfer Learning (TL) aims to extract the knowledge from one or more source tasks and applies the knowledge to a target task. It is a solution to solve problems for which the collection of high-quality, complete, and accurate training and validation datasets is not possible. Thus, instead of training a machine learning model from scratch, an already available model pre-trained for another task is reused with a smaller and low-quality dataset [21].

In addition, a comparison between these Machine Learning techniques is done in Table 36, which tabulates the main differences and challenges that each technique deals with.

Table 36: Matrix comparing different ML techniques

ML Technique	Reinforcement Learning	Online Learning / Incremental Learning	Batched Machine Learning / Deep Learning		
Type of learning	By rewards	Supervised (*)[12]	Supervised	Unsupervised	Semi-supervised
High-level definition	Algorithms that aim to maximize the rewards obtained by an agent encouraged to take actions in an environment	Algorithms that are able to adapt to new data sequentially and which compute the best model at every step	Algorithms that learn to map the input data to the outputs (classes or values)	Algorithms that are able to learn the data patterns without knowledge of the labels or values	Algorithms that are able to learn the patterns of the data against a few labelled samples and a large number of unlabelled ones
Learning procedure	In real-time when a reward is received after the action was performed	New adaptations are done incrementally and in real-time as new data arrives	A pre-defined dataset is fit entirely into an AI model until it is able to infer the classes or values as best as possible	A pre-defined dataset is fit entirely into an AI model until it is able to discover the output	
Expected input data over time	Context and reward after an action	High variation	Low variation		
Main drawbacks	Requires a very specific and active operational environment to obtain good results, depending directly on the rewards received	Adaptations to new data need to be controlled: new adaptations may lead to catastrophic forgetting; too drastic adaptations may lead to bad results	On the one hand, it needs a large amount of data to cover most of the data variation spectrum. On the other hand, it may lead to overfitting trying to cover all this spectrum. It may also lead to bad results when the training data is not representative enough or when it has a big bias towards one (or some) of the outputs		

[12] (*) Could be unsupervised in some cases

5.2 Privacy-preserving Federated Machine Learning in the MLaaS platform

Traditional machine learning approaches demand the training data be centralized on one machine or central server. For this purpose, those approaches collect a vast amount of data from devices (smartphones, laptops, IoT devices) and transfer it to central servers for training. However, data owners are not often willing to share their data, because it may contain sensitive information, which is subject to the GDPR. Thus, data privacy is a major concern. To address this problem, Federated Learning (FL) is introduced by Google [22], which is a distributed machine learning approach.

FL aims to build and train global models based on training datasets that are distributed across different remote devices while avoiding data leakage. Thus, as opposed to traditional approaches, FL inherently enhances privacy and security as the data is never processed on central servers, decoupling the machine learning process from the data sources.

In practice, FL solutions train an initial, generic machine learning model in a central server, which is a baseline to start with. Afterwards, the server sends this model to the user's devices, where the local copy of the model is trained using its own data. Then, the updated model parameters are sent back to the central server and the global model is updated. Therefore, FL approaches are capable of learning robust models from a huge amount of distributed data across devices without transferring and/or processing it on a central server.

FL approaches can be applied in several application areas, in which data privacy is required, such as health care, telecommunications as well as IoT networks. The term devices are referred to entities that are participated in the communication network of federated learning. Smartphones can be viewed as devices that commonly used in the FL approach, to jointly learn users' behaviour, while protecting their personal privacy by not sharing their data. The application in [23], learns a predictor in a large-scale smartphone network based on users' text data. Organizations/institutes can also be treated as devices in federated learning. For instance, hospitals are organizations that private data of patients, that should remain local. FL architecture presented in [24] can reduce privacy leakage and implement private learning between the different organizations. Additionally, FL can also be used on IoT networks to ensure privacy, enabling on-device machine learning solutions without the need to store private data from end devices to a central server.

Federated learning approaches can be categorized based on the distribution characteristics of the data [25]. The instances and the features of the data may differ among parties, so FL can be distinguished in horizontal FL, vertical FL and federated transfer learning, based on the way that data is distributed among parties in the feature and sample space. Specifically, at horizontal FL approaches the datasets to share the same features, but they differ in instances. While, vertical FL can be applied when the datasets share the same instances, but they present differences in features. Lastly, Federated Transfer Learning is applicable when the datasets differ in instances and features, with only a small portion of the features and instances overlapped.

Initially, the FL was introduced for mobile and edge devices applications. Those FL settings are referred to as "cross-device" [26]. This FL setting is applied in many consumer digital products. For example, Google widely uses FL in the Gboard mobile keyboard [23], [27], [28], while Snips applies cross-devices FL for hotword detection [29]. Additionally, the great interest in FL has led to applications, which might involve a small number of relatively reliable clients to train a model. Specifically, these FL settings are mentioned as "cross-silo" and they are applicable when several organizations or companies share and train a common model, without sharing the data directly due to confidentiality and legal constraints. A cross-silo setting can also be employed within a single company/organization when the data cannot be centralized between different geographical regions. Cross-silo applications have been explored in different domains, including drug discovery [30] and detection vaccine adverse event mentions [31]. Regarding the data partitioning, the cross-device setting is a horizontal FL approach, while the cross-silo can be either horizontal or vertical FL approach.

In the case of centralized FL, a central server orchestrates all the necessary steps of the algorithm and coordinates the participating clients during the learning process. This central server is responsible to aggregate the received model updates as well as sending model updates to nodes. In some more collaborative learning cases, the use of a powerful and reliable central server might not always be available and desirable [32]. A potential bottleneck of centralized FL is the communication traffic jam that can be occurred since all the nodes must communicate with the central node [33].

This bottleneck can be addressed by the decentralized federated learning approaches, in which individual nodes can communicate with each other to obtain the global model. The key difference is that in decentralized FL the communication with the central server is replaced by peer-to-peer communication between individual clients. The topology of decentralized FL techniques is represented as a connected graph, where the nodes are the clients and the edges are the communication channels between two nodes. Thus, devices only communicate with neighbours. Blockchain can be characterized as a popular decentralized platform. Authors in [34] propose a decentralized FL architecture based on blockchain for global model storage as well as model update exchange. Additionally, [35] presents a blockchain-based privacy-preserving FL platform for IoT devices, which trains an ML model at customers' data from various home appliances. Decentralized FL appears to outperform centralized FL, by reducing the high communication cost of a network with a central server.

Although FL enables on-device machine learning, it does not itself guarantee security and privacy. The fact that the private data are not shared with the central server is an advantage without doubts, still, there are ways to extract private information from data. After the shared model is trained at the user's device based on its own private data, the trained parameters are sent to the central server. Thus, it is possible to extract information about the private data from those trained parameters. For example, [36] demonstrates that it is possible to extract sensitive text patterns, like the credit card number, from a recurrent neural network that is trained on users' data. Therefore, additional methodologies are required to protect data from attack strategies, which are subject to privacy-preserving mechanisms on FL. The approaches that can be applied in FL for data protection are differential privacy, homomorphic encryption and secure multiparty computation.

Differential Privacy (DP) is a method that randomizes part of the mechanism's behaviour to provide privacy [37], [27]. For the FL scenario, a mechanism is considered the learning algorithm. The motivation behind adding randomness into a learning algorithm is to make it impossible to reveal behaviour patterns that correspond either to the model and the learned parameters or to the training data. Thus, the DP provides privacy protection against a wide range of privacy attacks (e.g., differencing attack, linkage attacks) [28]. The method of adding noise can result in great privacy but may compromise accuracy, therefore there is a trade-off between using differential privacy and achieving a high level of model accuracy. However, the authors in [29] present a method, which does not sacrifice accuracy to privacy.

Secure Multiparty Computation (SMC) is a well-defined cryptographic technique that allows a number of mutually distrustful parties to jointly compute a function while preserving the privacy of the input data [38], [25]. In the case of ML applications, the function can be the model's loss function at training, or it could be the model itself at inference. The challenge of applying SMC on a large-scale distributed system is the communication overhead, which increases with the number of participated parties.

Homomorphic encryption [39] secures the learning process by applying computations (e.g., addition) on encrypted data. Specifically, an encryption scheme is characterized as homomorphic, when standard operations can be applied directly to the cypher data, in such a way that the decrypted result is equivalent to performing analogous operations to the original encrypted data [40], [41]. For machine learning methods, homomorphic encryption can be applied when training and inference are performed directly on encrypted data. In scenarios, where large mathematical functions are implemented to cypher text space, the remarkable properties of homomorphic encryption schemes confront several limitations, related to the encryption performance.

Several open sources frameworks and libraries have been developed to facilitate the widespread use of FL in machine learning. Google's TensorFlow Federated[13] provides functionalities with a comprehensive set of features, which can help to perform research and to implement FL models. Another library is PySyft, which was introduced by OpenMined[14]. PySyft is suitable for research in FL and allows the users to perform private and secure Deep Learning. PySyft is also used in PyGrid[15], a peer-to-peer platform for federated learning and data privacy, which can be used for private statistical analysis on the private dataset as well as for performing FL across multiple organization's datasets. Additionally, FATE[16] (Federated AI Technology Enabler) framework supports FL architectures and secure computation of various machine learning algorithms.

5.3 Enhancing Machine Learning techniques in IoT-NGIN

According to IoT-NGIN living labs, there are three major types of data to be handled by the platform: images data, structured data and time-series data, but other types of data could be also expected in future iterations. From the IoT-NGIN living labs requirements of the Artificial Intelligence techniques to be exploited, we can extract the high-level results collected in Table 37. Considering these results, as the data is not only coming in predefined datasets or batches of known size, but also as a stream, it is reasonable, then, to investigate the machine learning techniques that have been proved to work well with these types of data, and to define how they can be integrated into a platform such as the one proposed in this project. As explained previously, the implementations to be made following two main patterns: enhancing the training procedures and improving the resulting models automatically once deployed. In each pattern there are different kinds of techniques, some of them have already been overviewed in the previous section, so only how they would be managed in the MLaaS platform will be tackled, but other techniques, such as the optimization of the AI models, are contemplated in the following sections.

[13] https://www.tensorflow.org/federated
[14] https://www.openmined.org/
[15] https://blog.openmined.org/what-is-pygrid-demo/
[16] https://fate.fedai.org/

Table 37: High-level features of IoT-NGIN living labs relevant to Artificial Intelligence

Living lab / Use Case		Data Type(s)	Outcome type(s)	ML Techniques
Smart City	**UC1**	Structured Images Timeseries	Prediction Multi-class classification Multidimensional regression	Unsupervised ML/DL Self-Learning (Auto-labelling)
	UC2		Multi-class classification Multidimensional regression	Unsupervised ML/DL
Smart Agriculture	**UC4**	Images Structured	Classification	Unsupervised ML/DL Federated learning
	UC5		Visual Mapping and Trajectory prediction Binary Classification	Unsupervised ML/DL Computer Vision Techniques
Industry 4.0	**UC6**	Structured Images	Prediction	Federated Learning
	UC7	Structured Images	Classification	Federated Learning
	UC8	Structured-time-series	Binary classification Anomaly detection	Supervised ML/DL Federated learning
Smart Energy	**UC9**	Structured-time-series	Classification Prediction	Supervised ML/DL Continuous/online learning
	UC10		Prediction	Supervised ML/DL Federated learning Online learning

5.3.1 Enhancing training processes

Enhancing the training of the AI models can be accomplished from different standpoints. Although one of the most relevant factors, when performance improvement matters, is the selection of the underlying ML or DL algorithm, one cannot forget about the actual training process. It may also be enhanced through different procedures, like using hardware capabilities, such as using GPUs or other novel ML accelerators, or by improving the actual software execution of the algorithm by means of quantization or binarization techniques.

Regarding the algorithm selection, the most influential factors to select one are: (i) the type of the training data and (ii) the type of outcome expected (classification, forecasting, anomaly detection). As for time series data, historically there have been applied different algorithms or network architectures depending on the task at hand [42]. The state-of-the-art shows that Recurrent Neural Networks (RNN), such as Long short-term memory (LSTM) networks [43], have proven to be one of the most effective architectures for time series data since they are designed to handle an internal state (memory) that grants them to understand the temporal connections between consequent inputs. Nevertheless, there has also been a success in using other architectures such as Deep Belief Networks, especially for forecasting tasks, or Convolutional Neural Networks (CNN), especially for classification tasks. In addition, for anomaly detection tasks, Autoencoders networks have had quite a success in fitting temporal data. Moreover, the diversity of time series datasets, as reported for instance in [44], shows that it is not possible to map only one kind of neural network with a specific type of time series dataset. Thus, in the context of MLaaS platforms that want to handle this kind of data, it is best to provide tools to build and test several kinds of neural network architectures, granting the user the freedom to select and try the one that could fit the type of the dataset. For that, there already exist several AI frameworks that already grant usage of these deep learning algorithms, such as the well-known Tensorflow [45] and PyTorch [46] frameworks. With respect to video/image data, neural network architectures have also been applied historically, and with a very high ratio of success. Undoubtedly, CNNs are one of the most applied architectures for this kind of issue, but the state-of-the-art shows that there exist lots of architectures that may be applied with results just as good as CNN's. What's more, a common approach is to combine different types of neurons and other kinds of methods such as pooling [47], which allows selecting the best representations of the convolutional neurons output, or batch normalization [48], that reduces the complexity of the output neurons with the intention of avoiding overfitting [49]. AI Frameworks like Tensorflow and PyTorch already provide tools to execute this kind of techniques on video/image data.

Moreover, as stated in [50] and other similar studies, the particular design of the circuits that execute the artificial intelligence operations is shown to affect the performance in terms of training and inference speed. Graphical processing units (GPU) have been historically one of the hardware devices selected to improve the training speed of AI models, due to their effectiveness in parallel operations. However, nowadays there exist other kinds of hardware devices that may perform similarly or even better; for instance: Google's Tensor Processing Unit (TPU)[17]. What's more, using those devices for executing the ML tasks does not suppose major effort from the point of view of the AI Developer, as most of the AI frameworks already offer seamless integration with this kind of devices. Thus, the capabilities of an MLaaS platform would clearly be boosted when it grants the usage of these hardware accelerators with the

[17] https://cloud.google.com/tpu/docs/tpus

supported AI frameworks. This will be considered for the developments to be carried out in the upcoming months of the project.

5.3.2 Strengthening the performance of the ML models deployed

Some IoT environments need their applications to tailor their performance at runtime, according to the variation of their data over time. Traditionally, data for training AI models has always been processed in batches, so the model has been aware of all the training data when computing the loss during the learning process. However, there exists also the case in which the data comes in a stream and, what is more, the variance of the data might be changing over time. This last scenario, which fits for instance in Twin Smart Cities Living lab or in the Smart Energy Living Lab, is not suitable for traditional machine learning techniques since this variation of fresh data makes pointless computing the loss on all the previous data, as it is not representative of the current data anymore. Just as the enhancement of the training procedures, there exist different ways to adapt the operational performance of the AI models, once they have been deployed, to the changes in their IoT environment. One approach to deal with this issue may be the so-called self-learning methodology, in which the AI models adapt themselves dynamically through different kinds of algorithms.

One relatively common approach to adapt models to changes is the Incremental or continuous learning paradigm, where input data (a variable batch of samples) is continuously processed by the AI model without forgetting the already obtained knowledge and without repeating the whole learning process. However, not every machine learning algorithm can apply incremental learning [51] since some of them, especially some neural networks, tend to forget past patterns as new data arrives, a problem known as catastrophic forgetting [52]. Despite this, there has been a success in applying some machine learning procedures for incremental learning with streaming and time-series data, as in [53]. In addition, in this incremental learning context, automatic data labelling could be applied. With this technique, new data is processed by the model as it arrives, getting an initial classification for it, so that it can be used later on as labelled training data in subsequent training stages. Ideally, the confidence of the classification of each new sample is stored along with the sample, enabling a human annotator to verify the labels of the data in order to prevent using errors in the new training step. This procedure has the advantage of accelerating the initial data labelling phase, but it also has the inconvenience of the model being low accurate at the firsts iterations and, thus, producing some errors at the firsts inferences that must be fixed by humans annotators.

Another approach to tackle the continuous adaptation of AI models is the machine learning technique named online (machine) learning, introduced in [54]. The main difference with other machine learning approaches is that the training examples are processed one at a time, which means that the loss is computed against only that sample. In the context of streaming data, each sample is processed by the algorithm as it arrives and, afterwards, the instance is discarded (not used anymore). There already exist some implemented algorithms to perform the online learning methodology, such as the Online Gradient Descent [55], or the updated version Adaptive Online Gradient Descent [56]. Despite the existence of these open-source implementations for this paradigm, they are not as extended as the other algorithms mentioned in the previous section and, what's more, they have not been integrated with existing AI frameworks, but rather isolated implementations to fit one specific scenario.

Lastly, another state-of-the-art approach to strengthen deployed AI models is the reinforcement learning paradigm. In this context, an agent learns the behaviour of a specific scenario through dynamic interactions, in which each movement (inference) of the agent receives a reward representing how good was this movement according to the current context of the scenario. More specifically, every interaction of the agent starts with an input of the current state of the environment and, then, the agent chooses an action to generate as output. This action changes the state of the environment, and the value of this change, which is computed as a scalar according to a pre-defined formula, is fed back to the agent as a reinforcement signal [18]. In order for this agent to fit well into the scenario, it must choose actions that increase the sum of values of these reinforcement signals. Since the reinforcement learning paradigm differs from supervised learning in that it is not told which action would be best to take to obtain the best rewards in the long-term, it is key for the agent to receive rewards actively from the environment to fit well with it. This means that this methodology needs a very specific context in which this scenario of dynamic action-reward can take place actively. Usually, the reinforcement learning problem is phrased as a Markov Decision Process (MDP), but there exist different algorithms that do the actual implementation [19].

As we have shown, and considering IoT-NGIN's deliverable 1.1 [2], the self-learning context fits well with some of the Living Labs descriptions. Thus, the IoT-NGIN project, and more specifically WP3: Enhancing IoT Intelligence, will try to bring innovation also from this perspective along with the Federated Learning framework, by designing and implementing a solution that will integrate the streaming data from the Living Labs into the MLaaS platform, and will execute incremental learning, online learning and/or automatic data labelling algorithms on selected pilots suitable for them.

However, because it is difficult to find real-world implementations outside the academic field that are integrated with most famous AI frameworks and, especially, due to this necessity for a suitable action-reward scenario, the implementation of the reinforcement learning paradigm in IoT-NGIN's project will be studied in the next months, considering the Living Labs requirements progression and use cases evolution.

In addition to making use of innovative machine learning techniques, the performance of the AI models can be strengthened in other ways. One of these means is the acceleration of the operations executed by the models, especially when the operational environment is at the edge of the network. Here, the constraints of the hardware that is in charge of executing those AI operations must be taken into consideration. Due to these constraints, some IoT environments cannot handle large AI models, since the computational and memory loads that are required to execute them might be too heavy for the existing hardware, leading to a large response time, or even no response at all. For this particular obstacle, there exist techniques that compress the AI models into lighter versions, which are more feasible for low-powered hardware. Examples of these techniques are, for instance: Channel pruning [57], which removes some of the weights of the layers of the AI model; activation function compression [58], which reduces the size of the activation layers of the neural networks; or model quantization [59], which reduces the floating-point operations of the AI model to integer-only arithmetic. These specific approaches have been already developed for some AI frameworks, such as the TensorFlow Lite[18] SDK, which makes it possible to be integrated into IoT-NGIN's MLaaS platform in the upcoming months.

[18] https://www.tensorflow.org/lite

6 Conclusions

This document has presented the work related to all WP3: Enhancing IoT Intelligence tasks until the ninth month of the project. It is clear that a focus has been put into Task 3.1 since it deals with the development of the MLaaS platform. However, results of the other tasks are included within the design of the platform itself and within the study of IoT-NGN living labs. This report will not only feed the upcoming activities of WP3 tasks, but also other tasks of the project that benefit from any Artificial Intelligence activity.

Overall, the Machine Learning as a Service platform has been designed, and its main functionalities have been detailed through several use case diagrams and sequence diagrams. In addition, a logical view of the platform has been provided, in which the relations and interconnetions between the components of the platform have been explained. In addition to this, the federated learning framework to be implemented over the platform has been overviewed. Furthermore, the machine learning techniques to be implemented and applied on the living labs have been analyzed, in which a focus has been put on techniques related to enhancing the training processes and related to improving the performance of the AI models deployed.

Although innovating in the field of IoT Intelligence supposes a great challenge, this deliverable contributes to the state-of-the-art with a considerable effort to facilitate the application of Machine Learning techniques, including deep learning, self learning and reinforcement learning, in the next generation of IoT. Upcoming deliverables of the work packge will boost these efforts by showing how the implementation of the relevant techniques fit into real-world scenarios, solving complex problems related to IoT.

7 References

[1] P. B. Kruchten, “The 4+1 View Model of Architecture.”

[2] IoT-NGIN, “D1.1: Definition analysis of use cases and GDPR Compliance,” 2021.

[3] R. Bohm and G. Digital, “Industrial Internet of Things for Developers,” 2018.

[4] L. D. D. C. Papis.io, “Machine Learning Platforms,” 2019. Accessed: Jun. 10, 2021. [Online]. Available: https://assets.ctfassets.net/nubxhjiwc091/48kgJWJ6s6p7QZWKOsKWhH/c92e2a2a1e85f6ef44d1bd19d012bf8e/Machine_Learnings_Platform_Digital_Catapult_paper.pdf.

[5] “Gartner Magic Quadrant for Data Science and Machine Learning Platforms,” *ID G00385005*. Feb. 11, 2020.

[6] M. Ribeiro *et al.*, “MLaaS: Machine Learning as a Service,” 2015. Accessed: Jun. 10, 2021. [Online]. Available: https://ir.lib.uwo.ca/electricalpubhttps://ir.lib.uwo.ca/electricalpub/86.

[7] C. Prabhu, *Fog Computing, Deep Learning and Big Data Analytics-Research Directions*. Springer, 2019.

[8] S. Yi, Q. Li, and C. Li, “A Survey of Fog Computing: Concepts, Applications, and Issues A Survey of Fog Computing: Concepts, Applications and Issues,” *dl.acm.org*, vol. 2015-June, pp. 37–42, Jun. 2015, doi: 10.1145/2757384.2757397.

[9] R. Philipp, A. Mladenow, C. Strauss, and A. Völz, “Machine Learning as a Service-Challenges in Research and Applications,” *dl.acm.org*, vol. 11, pp. 396–406, Nov. 2020, doi: 10.1145/3428757.3429152.

[10] H. Tanuwidjaja, R. Choi, S. Baek, K. K.-I. Access, and undefined 2020, “Privacy-Preserving Deep Learning on Machine Learning as a Service—a Comprehensive Survey,” *ieeexplore.ieee.org*, Accessed: Jun. 10, 2021. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9194237/.

[11] A. Qayyum, A. Ijaz, M. Usama, W. Iqbal, … J. Q.-F. in big, and undefined 2020, “Securing Machine Learning in the Cloud: A Systematic Review of Cloud Machine Learning Security,” *ncbi.nlm.nih.gov*, Accessed: Jun. 10, 2021. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7931962/.

[12] M. Kuhn, K. Johnson, and others, *Applied predictive modeling*, vol. 26. Springer, 2013.

[13] T. Hastie, R. Tibshirani, and J. Friedman, “Overview of supervised learning,” in *The elements of statistical learning*, Springer, 2009, pp. 9–41.

[14] Z. Ghahramani, “Unsupervised learning,” in *Summer School on Machine Learning*, 2003, pp. 72–112.

[15] P. Baldi, “Autoencoders, unsupervised learning, and deep architectures,” in *Proceedings of ICML workshop on unsupervised and transfer learning*, 2012, pp. 37–49.

[16] X. Zhu and A. B. Goldberg, “Introduction to semi-supervised learning,” *Synth. Lect. Artif. Intell. Mach. Learn.*, vol. 3, no. 1, pp. 1–130, 2009.

[17] O. Chapelle, B. Scholkopf, and A. Zien, “Semi-supervised learning (chapelle, o. et al., eds.; 2006) [book reviews],” *IEEE Trans. Neural Networks*, vol. 20, no. 3, p. 542, 2009.

[18] L. P. Kaelbling, M. L. Littman, and A. W. Moore, “Reinforcement learning: A survey,” *J. Artif. Intell. Res.*, vol. 4, pp. 237–285, May 1996, doi: 10.1613/jair.301.

[19] C. Szepesvári, “Algorithms for reinforcement learning,” in *Synthesis Lectures on Artificial Intelligence and Machine Learning*, Jul. 2010, vol. 9, pp. 1–89, doi: 10.2200/S00268ED1V01Y201005AIM009.

[20] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[21] K. Weiss, T. M. Khoshgoftaar, and D. Wang, “A survey of transfer learning,” *J. Big data*, vol. 3, no. 1, pp. 1–40, 2016.

[22] “Google AI Blog: Federated Learning: Collaborative Machine Learning without Centralized Training Data.” https://ai.googleblog.com/2017/04/federated-learning-collaborative.html (accessed Jun. 29, 2021).

[23] A. Hard *et al.*, “Federated Learning for Mobile Keyboard Prediction,” Nov. 2018, Accessed: Jun. 07, 2021. [Online]. Available: http://arxiv.org/abs/1811.03604.

[24] L. Huang, Y. Yin, Z. Fu, S. Zhang, H. Deng, and D. Liu, “LoAdaBoost: loss-based AdaBoost federated machine learning with reduced computational complexity on IID and non-IID intensive care data,” *PLoS One*, vol. 15, no. 4, Nov. 2018, Accessed: Jun. 07, 2021. [Online]. Available: http://arxiv.org/abs/1811.12629.

[25] Q. Yang, Y. Liu, T. Chen, and Y. Tong, “Federated machine learning: Concept and applications,” *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, Jan. 2019, doi: 10.1145/3298981.

[26] P. Kairouz *et al.*, “Advances and Open Problems in Federated Learning,” *Found. Trends® Mach. Learn.*, vol. 14, no. 1, p. 16, Dec. 2019, Accessed: May 31, 2021. [Online]. Available: http://arxiv.org/abs/1912.04977.

[27] M. Abadi *et al.*, “Deep Learning with Differential Privacy,” *Proc. ACM Conf. Comput. Commun. Secur.*, vol. 24-28-October-2016, pp. 308–318, Jul. 2016, doi: 10.1145/2976749.2978318.

[28] C. Dwork and A. Roth, “The algorithmic foundations of differential privacy,” *Found. Trends Theor. Comput. Sci.*, vol. 9, no. 3–4, pp. 211–487, 2013, doi: 10.1561/0400000042.

[29] Y. Wang, C. Si, and X. Wu, “Regression Model Fitting under Differential Privacy and Model Inversion Attack.”

[30] Z. Xiong *et al.*, “Facing small and biased data dilemma in drug discovery with federated learning,” *bioRxiv*, p. 2020.03.19.998898, Mar. 2020, doi: 10.1101/2020.03.19.998898.

[31] P. Kanani, V. J. Marathe, D. Peterson, R. Harpaz, and S. Bright, “Private Cross-Silo Federated Learning for Extracting Vaccine Adverse Event Mentions,” Mar. 2021, Accessed: Jun. 07, 2021. [Online]. Available: http://arxiv.org/abs/2103.07491.

[32] P. Vanhaesebrouck, A. Bellet, and M. Tommasi, “Decentralized Collaborative Learning of Personalized Models over Networks,” PMLR, Apr. 2017. Accessed: May 31, 2021. [Online]. Available: http://proceedings.mlr.press/v54/vanhaesebrouck17a.html.

[33] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, “Can Decentralized Algorithms Outperform Centralized Algorithms? A Case Study for Decentralized Parallel Stochastic Gradient Descent,” *Adv. Neural Inf. Process. Syst.*, vol. 2017-December, pp. 5331–5341, May 2017, Accessed: Jun. 07, 2021. [Online]. Available:

http://arxiv.org/abs/1705.09056.

[34] Y. Li, C. Chen, N. Liu, H. Huang, Z. Zheng, and Q. Yan, "A Blockchain-based Decentralized Federated Learning Framework with Committee Consensus," *IEEE Netw.*, vol. 35, no. 1, pp. 234–241, Apr. 2020, doi: 10.1109/MNET.011.2000263.

[35] Y. Zhao *et al.*, "Privacy-Preserving Blockchain-Based Federated Learning for IoT Devices," *IEEE Internet Things J.*, vol. 8, no. 3, pp. 1817–1829, Jun. 2019, Accessed: Jun. 07, 2021. [Online]. Available: http://arxiv.org/abs/1906.10893.

[36] N. Carlini, C. Liu, Ú. Erlingsson, J. Kos, and D. Song, "The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks," *Proc. 28th USENIX Secur. Symp.*, pp. 267–284, Feb. 2018, Accessed: Jun. 07, 2021. [Online]. Available: http://arxiv.org/abs/1802.08232.

[37] F. McSherry and K. Talwar, "Mechanism Design via Differential Privacy," Apr. 2008, pp. 94–103, doi: 10.1109/focs.2007.66.

[38] C. Zhao *et al.*, "Secure Multi-Party Computation: Theory, practice and applications," *Inf. Sci. (Ny).*, vol. 476, pp. 357–372, Feb. 2019, doi: 10.1016/j.ins.2018.10.024.

[39] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "ON DATA BANKS AND PRIVACY HOMOMORPHISMS," 1978.

[40] C. Gentry, "A FULLY HOMOMORPHIC ENCRYPTION SCHEME," 2009.

[41] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-Preserving Deep Learning via Additively Homomorphic Encryption," *IEEE Trans. Inf. Forensics Secur.*, vol. 13, no. 5, pp. 1333–1345, May 2018, doi: 10.1109/TIFS.2017.2787987.

[42] J. Gamboa, "Deep Learning for Time-Series Analysis."

[43] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[44] H. Anh Dau *et al.*, "The UCR Time Series Archive." Accessed: May 31, 2021. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8894743/.

[45] M. Abadi *et al.*, "TensorFlow: A system for large-scale machine learning," *Proc. 12th USENIX Symp. Oper. Syst. Des. Implementation, OSDI 2016*, pp. 265–283, May 2016, Accessed: May 31, 2021. [Online]. Available: http://arxiv.org/abs/1605.08695.

[46] A. Paszke *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," 2019. Accessed: May 31, 2021. [Online]. Available: https://arxiv.org/abs/1912.01703.

[47] Y.-L. Boureau, J. Ponce, J. P. Fr, and Y. Lecun, "A Theoretical Analysis of Feature Pooling in Visual Recognition," 2010. Accessed: May 31, 2021. [Online]. Available: https://www.di.ens.fr/willow/pdfs/icml2010b.pdf.

[48] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How does batch normalization help optimization?," in *Advances in Neural Information Processing Systems*, 2018, vol. 2018-December, pp. 2483–2493.

[49] D. M. Hawkins, "The Problem of Overfitting," *Journal of Chemical Information and Computer Sciences*, vol. 44, no. 1. American Chemical Society , pp. 1–12, Jan. 2004, doi: 10.1021/ci0342472.

[50] W. J. Dally *et al.*, *Hardware-Enabled Artificial Intelligence*. .

[51] G. M. van de Ven and A. S. Tolias, “Three scenarios for continual learning,” Apr. 2019, Accessed: May 31, 2021. [Online]. Available: http://arxiv.org/abs/1904.07734.

[52] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio, “An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks,” *2nd Int. Conf. Learn. Represent. ICLR 2014 - Conf. Track Proc.*, Dec. 2013, Accessed: May 31, 2021. [Online]. Available: http://arxiv.org/abs/1312.6211.

[53] S. Ahmad, A. Lavin, S. Purdy, Z. A.- Neurocomputing, and undefined 2017, “Unsupervised real-time anomaly detection for streaming data,” *Elsevier*, Accessed: May 31, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231217309864.

[54] G. Widmer and M. Kubat, “Learning in the presence of concept drift and hidden contexts,” *Mach. Learn.*, vol. 23, no. 1, pp. 69–101, Apr. 1996, doi: 10.1023/A:1018046501280.

[55] M. Zinkevich, “Online Convex Programming and Generalized Infinitesimal Gradient Ascent.” Accessed: May 31, 2021. [Online]. Available: https://www.aaai.org/Papers/ICML/2003/ICML03-120.pdf.

[56] P. Bartlett, E. Hazan, and A. Rakhlin, “Adaptive Online Gradient Descent,” *EECS Dep. Univ. California, Berkeley*, Jun. 2007, Accessed: May 31, 2021. [Online]. Available: https://repository.upenn.edu/statistics_papers/163.

[57] Y. He, “Channel Pruning for Accelerating Very Deep Neural Networks.” Accessed: May 31, 2021. [Online]. Available: http://openaccess.thecvf.com/content_iccv_2017/html/He_Channel_Pruning_for_ICCV_2017_paper.html.

[58] G. Georgiadis, “Accelerating Convolutional Neural Networks via Activation Map Compression.” Accessed: May 31, 2021. [Online]. Available: https://ai.intel.com/nervana-nnp/.

[59] B. Jacob *et al.*, “Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference.” Accessed: May 31, 2021. [Online]. Available: http://openaccess.thecvf.com/content_cvpr_2018/html/Jacob_Quantization_and_Training_CVPR_2018_paper.html.