FRACTAL

Deliverable

D7.2 Prototypes of the reference designs

Deliverable Id:	D7.2
Deliverable Name:	Prototypes of the reference designs
Status:	Submitted
Dissemination Level:	Public
Due date of deliverable:	15. September 2022
Actual submission date:	15. September 2022
Work Package:	WP7
Organization name of lead contractor for this deliverable:	PLC2
Author(s):	Alexander Flick
Partner(s) contributing:	Bekim Chilku, SIEM Michael Gautschi, ACP Martin Rivas Caneiro, PROI Alfonso Gonzalez Gil, Zylk Inaki Paz,LKS Naga Venkata Harisyam Manda, AVL Bernhard Peischl, AVL Pascal Muoka, SIEG Daniel Onwuchekwa, SIEG Ester Sola, ZYLK Anabel Berenguer, UOULU
Reviewers:	Jerome Quevermont, THA Carles Hernandez, UPV

Abstract:
This report describes the actual prototypes that exhibit the required node building blocks for the reference FRACTAL cognitive nodes. These platform demonstrators are augmented by the building block to be verified.

ECSEL Joint Undertaking
Electronic Components and Systems for European Leadership

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 877056

Co-funded by the Horizon 2020 Programme of the European Union under grant agreement No 877056.

Contents

1 History

Version	Date	Modification reason	Modified by
v0.1	10.Mar.2022	Initial setup	Alexander Flick
v0.2	16.Mar.2022	Document structure proposal	Alexander Flick
v0.5	31.Aug.2022	Cleanup for internal review	Alexander Flick
v0.6	4.Sept.2022	Plain review comments	Alexander Flick
V0.7	8.Sept.2022	Comments resolved	Alexander Flick
V1.0	14. Sept. 2022	Submission candidate	Alexander Flick

2 Summary

This report is part of the milestone 5 deliverables and describes the prototypes built along the verification uses cases UC1 to UC4 to deploy specific building blocks that will prove the benefits of a set of FRACTAL features. The use cases aim to focus on the integration of the technical activities that all the partners undertake to unify the different modules that comprise the FRACTAL architecture. With this adaption of the components the demonstrators will help to align the building blocks for integration into the validation use cases of WP8, where an even wider application scope is used to show that FRACTAL can support the full set of use case requirements.

The verification focus of the WP7 use cases has already been laid out in D7.1, where the particular features used have been linked to the use case KPIs. This report D7.2 is organized along the use cases and details the specific demonstrator in each use case - or multiple demonstrators in case of UC1. To support the mapping of the features into a set of FRACTAL components the introductory Chapter 3 revisits the FRACTAL High Level Architecture. Following this systematic derivation, the use case chapters detail the prototype and demonstrator setup within their application context. Descriptions of the specific implementation are given with focus on the FRACTAL components that actually contribute to the use case evaluation scenario.

Along the project progress the technical demonstrators evolve with the integration of the technical work package (WP3 to 6) deliveries. D7.2 reports on the state of the implementation of demonstrators at the end of the second year of the FRACTAL project. Further insights through evaluation results and analysis will be discussed in the upcoming D7.3 report.

3 Introduction

This chapter provides an introduction to FRACTAL architecture in several levels of detail.

3.1 FRACTAL High Level Architecture

FRACTAL from a high-level point of view can be seen as a simple layered architecture with three layers:

A **node layer** that refers to the element that provides certain low-level characteristics such as AI Accelerators or computing power. Over the node layer, a **service orchestration layer** is built that enables to manage the services that run over the node. Finally, the **application layer** describes the business logic for the different applications / UCs.

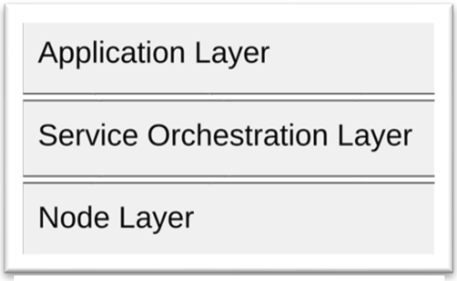


Figure 1 FRACTAL layered architecture

Fostering the notion of fractality, the node layer includes both the Cloud Node and the Fractal Edge Node, and the Fractal Edge Node references both the HW Edge Platform and the Low-Level SW Platform (which includes OS, drivers for HW, etc.).

Components on the application layer will be able to access characteristics provided by the Node Layer (such as AI accelerators).

3.2 FRACTAL Features

A FRACTAL feature is a distinguishing characteristic of FRACTAL, visible to users that will configure FRACTAL for their use cases. This is, features are high level concepts that may crosscut the distinct layers of the architecture and that a certain UC may (or may not) require. The following diagram presents the FRACTAL high-level features:

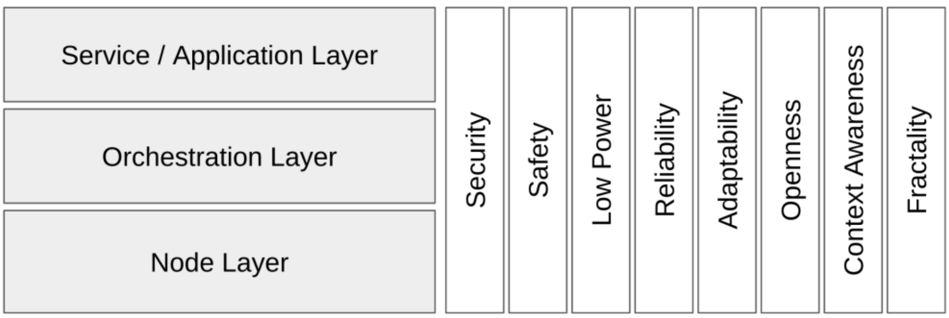


Figure 2 FRACTAL High Level architecture with FRACTAL features.

FRACTAL features provide different dimensions and characteristics to the High Level Architecture. These high-level features have been extracted from the concepts presented in the description DoA and have been refined in a Feature Diagram presented in D2.3 to provide additional details. Section 3.4 partially showcases the feature diagram for the adaptability feature.

3.3 FRACTAL Big Picture

Technical work packages (WPs 3 to 6) have been working to provide components or building blocks to be used by the UCs. In WP7 the integration of those contributions is undertaken with the support from all the participating partners.

The activities in WP7 Task 7.2 derive suitable demonstrator platforms to verify certain aspects of these FRACTAL features in the context of the UC1 to UC4 problem space as described in D7.1.

The generic set of available components is presented as a reference platform using the FRACTAL Big Picture Figure 3. As this is a superset of the defined hardware platform within the FRACTAL project, the actual demonstrator designs will need to focus on the specific supported components of the actually chosen hardware platform.

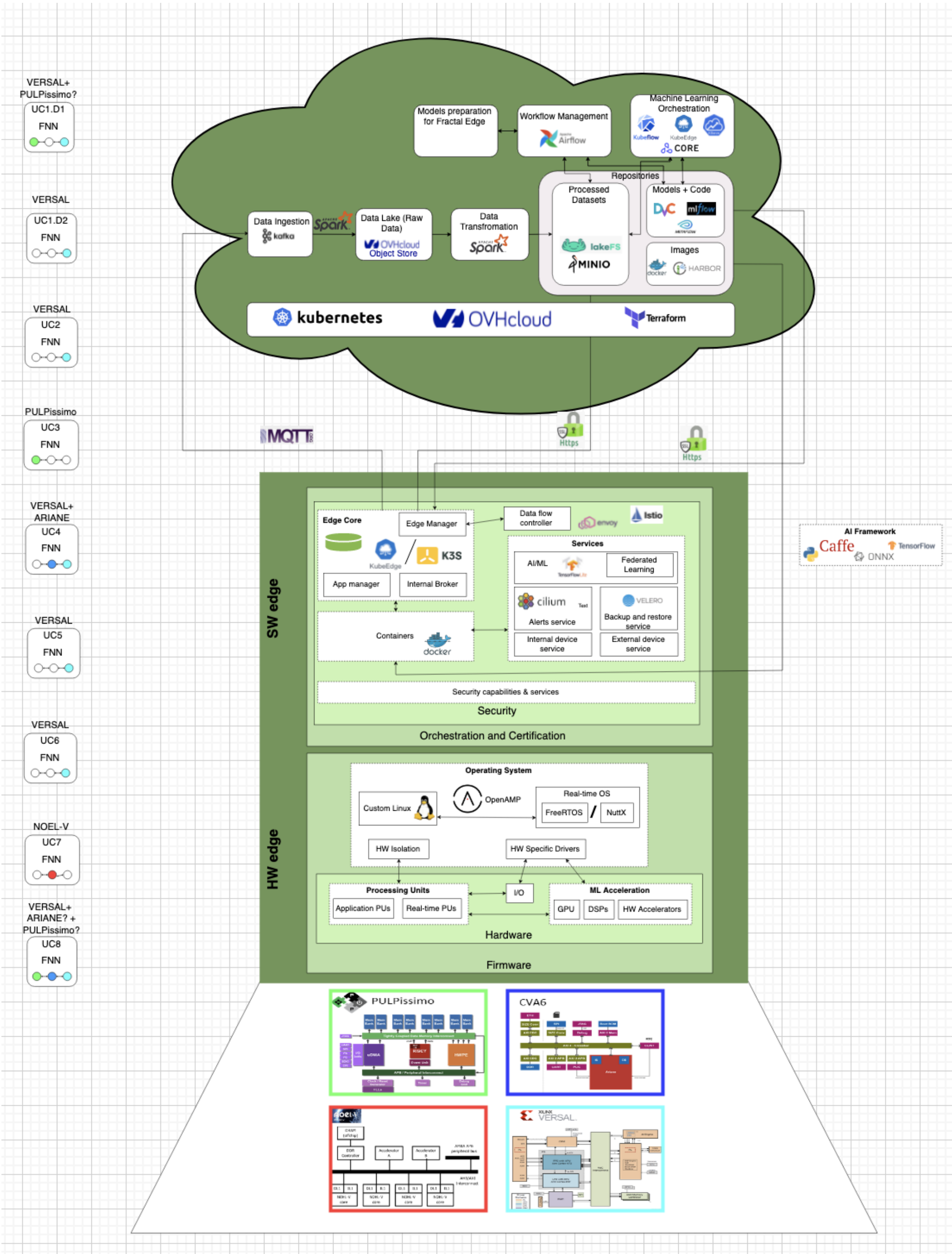


Figure 3 FRACTAL Big Picture

Following the High-Level Architecture, the node layer is reflected at the top and the bottom of the image: (1) by the cloud and the different services and capabilities it may provide, and (2) the HW Edge that depicts the HW elements and the low-level SW over the hardware. This is a generic view for the HW Edge, which has been customized for the 4 actual platforms (PULP, CVA6, Noel-V and Versal).

On the nodes, a service orchestration layer is depicted, that in the cloud node is clearly based on Kubernetes. Finally, over these orchestration layers, components are depicted to deal with Data Orchestration (IoT) and Model Orchestration and Inference (AI). Both part of the application layer.

3.4 Features and Use Case Demonstrators

In D2.3 a FRACTAL feature diagram has been presented that addresses the variable capacities in FRACTAL. Figure 4 depicts a partial view on the Adaptability feature decomposition.

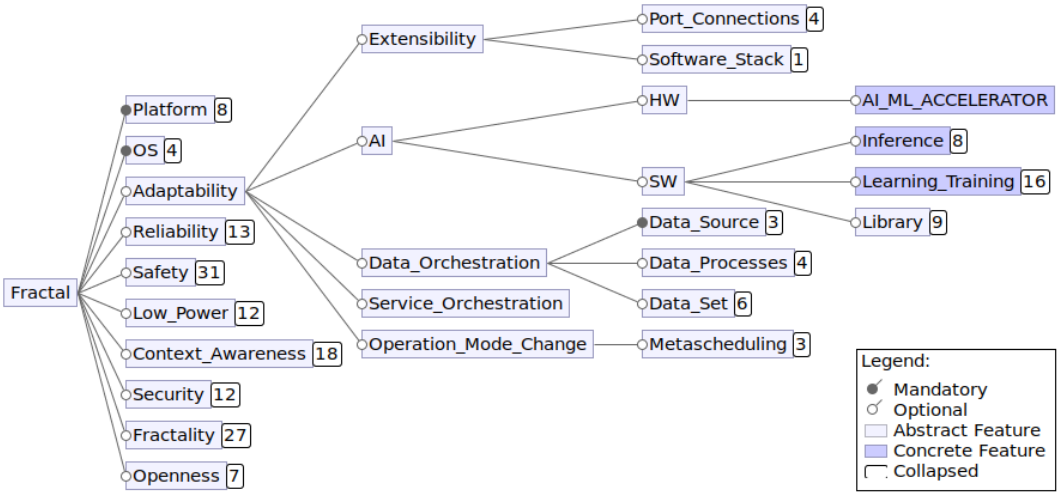


Figure 4 Partial feature diagram on Adaptability FRACTAL Feature

Each UC has been requested to select the desired features (the specific selection is called Bill of Features). On the other hand, features have been associated with components. Thus, a traceability path from UC needs to technical WPs components and building blocks has been defined. In this context, this document (D7.2) describes for each UC the selected set of components that the UC will use.

Within WP7 four use cases have been identified to establish the respective features and methods that would suit a verification of these features. These use cases are:

- **UC1** – Engineering and Maintenance Works led by PROIN
- **UC2** – Automotive-Airpath-Control led by AVL
- **UC3** – SmartMeter led by ACP
- **UC4** – Low-latency Object Detection as a generic building block for perception in the edge for industrial application led by SIEM

Along the verification plans devised in D7.1 the use case partners have set up suitable verification platforms and derive assessment metrics from experiments running on these demonstrators. In subsequent chapters, each use case will be covered based on the hardware platform specific version of the big picture and further detail the various components actually implemented in the demonstrator platform.

4 UC1 – Edge computing technologies applied for engineering and maintenance works

The construction business is characterized by being a very old-fashioned and poorly digitized sector, which means a very representative loss of efficiency, and which has an impact on the final costs of construction works. The implementation of new technologies, in turn, generates an initial rejection of them due to the tight timescales of construction works, added to the component of the need for regulation and certification of technological solutions for their validity and use in the works. Currently, construction companies are making an effort to use solutions that reduce the major problems arising from construction sites, where they are mainly advocating a zero-accident environment, through the improvement of health and safety systems and supervision, in addition to the aforementioned cost reduction and improved efficiency of the actions. UC1 seeks to globally improve safety conditions during the course of construction works, through the deployment of two demonstrators: UAV supervision of critical structures (maintenance focus) and Wireless Sensor Network (WSN) for safety at construction sites (real-time safety focus).

4.1 Demonstrator I – UAV supervision of critical structures

4.1.1 Hardware Platform Big Picture

The control and maintenance of large structures is technically complex, dangerous and expensive. Conventional inspection methods are mainly based on human visual approaches. Furthermore, the inspection of these components, that are usually hard to reach, supposes the need of special equipment.

The rapid technical improvement of UAVs in the last years suggests the opportunity of using them to improve the inspection and monitoring of infrastructures. The usage of this kind of vehicles can improve the efficiency and quality of the inspection, by gathering and analysing images on the edge and extracting detailed information of the pathologies. Thus, the present project incorporates the usage of UAVs to collect and analyse data (on the edge) of civil infrastructures, systematizing and improving the detection of pathologies or defects in the buildings, through the development of AI algorithms that analyse the collected images.

Figure 5 Concept of drone to supervise specific structures

There are different options to approach the correct algorithm to extract information from the images. Classical methods include approaches such as histogram thresholding (Otsu thresholding), analysis of local parameters and edge detection, growing algorithms and clustering methods (Zhu, 2011). However, in the recent years it has been shown the superior performance of deep convolutional neural networks (CNN) approaches. Concretely, three main deep learning approaches have been considered to face the instance segmentation problem (Terzopoulos, 2020):

- U-Net based architectures: This type of CNNs includes encoder-decoder structures that allow extracting more complex features of the image, and residual connections that result in a better propagation of the information through the neural network.
- Multi-scale models (FPN): The main advantage of this method is the great performance at extracting features at different scales.
- Attention-based models (R-CNN): These models propose a region of the image in which the object may be located, and a posterior segmentation process is carried out.

The selected algorithm is a deep learning model based on a convolutional neural network (CNN). In order to solve the segmentation problem, the architecture of the deep neural network is U-Net (and ResNet), with a custom loss-function (combination of Categorical Cross entropy and IOU). The deep learning model is initially trained with a dataset generated and labelled in an early stage. In addition, the model has the capability to improve its performance, by being retrained with new images collected during the lifecycle of the system.

One of the biggest challenges of the UC1 was the insight generation on the edge (on the UAVs). Deep neural networks are expensive algorithms in terms of consumption of resources such as energy and computing power. Thus, in order to achieve a good performance, special hardware must be used. Three options have been considered:

- Xilinx-Versal: Xilinx processor with ACAP (Adaptable Compute Acceleration Platform) architecture (on the same chip there are ARM processors, an FPGA and a vector calculation acceleration unit (inference with AI models)).
- PULP: PULP is an open hardware project developed by the universities of Bologna and ETH Zürich. The PULP project provides the microprocessor

architecture (RI5CY), and external companies develop chips with this architecture.

- Development of the prototype with other microcontrollers: When making a working proof of concept, other microcontroller options may be better suited for this UC in terms of price and ease of development. However, notice that this may not be the best option for production.

The commercial node Xilinx VERSAL ACAP was selected as hardware for the processing of the images. In addition, due to problems with the integration of the prototype board because of its size, there will be no direct communication between the drone and the processing environment, but the images will be downloaded for subsequent inference on the VERSAL board. Initially, the inference tests will be performed on a Jetson Nano board, since the VERSAL board is not available, and then the same solution will be developed on the VERSAL board to observe the performance improvement between the two boards.

The way to achieve the process of the model training, inference and re-training, is by using the following components from the FRACTAL platform:

Table 1 List of Components in UC1-Dem1

Component ID	Component name	Usage of component
WP3T32-10	Versal Accelerator Building Blocks	Support the AI developments for inference and the AI framework to use TensorFlow
WP3T34-03	Versal Model Deployment Layer	Model deployment on the Versal APU + DPU control from model repository images
WP4T42-02	Versal RPU Access to AI acceleration	Enable local AI acceleration deployment from RPU
WP6T61-01-03	Petalinux	
WP6T61-01-04	Vitis AI	

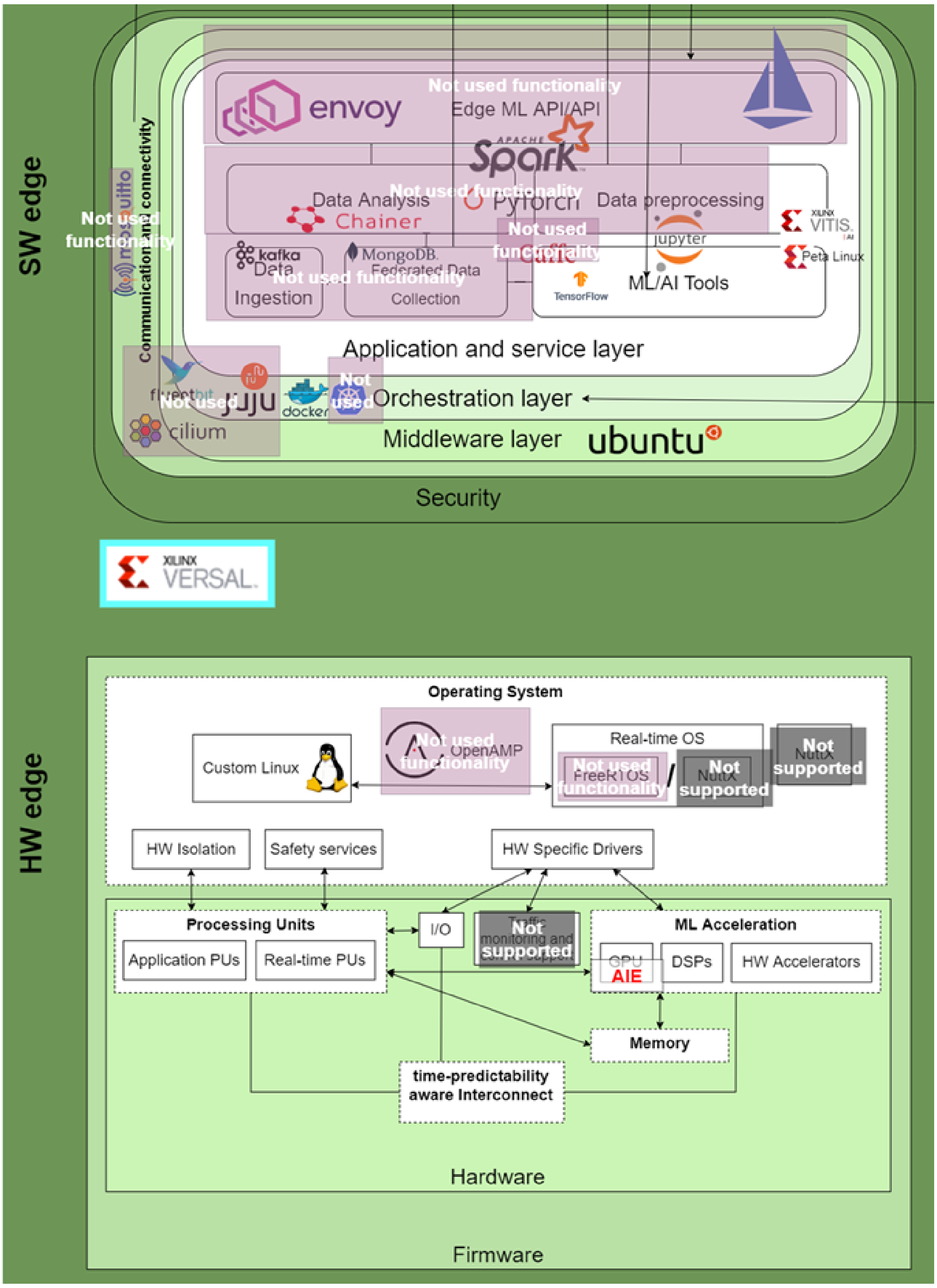


Figure 6 FRACTAL Big Picture: UC1-D1

As we can see in Figure 6, UC1-Demo1 will not use the cloud part of Fractal's big picture, focusing on the use of the components developed in the project associated with the VERSAL platform.

4.1.2 Demonstrator Setup

The goal of this demonstrator is the detection of cracks on reinforced concrete infrastructures. Concretely, this crack detection system will be used on structures such as bridges and constructions of limited access, on which it is difficult and expensive to carry out a human inspection. In order to get the insights of the cracks on concrete surfaces the following datasets have been considered:

- Deep Crack dataset – cracks on concrete infrastructures: https://github.com/yhlleo/DeepCrack
- Cracks on masonry: https://github.com/dimitrisdais/crack_detection_CNN_masonry
- To be studied: https://apps.peer.berkeley.edu/phi-net/

These datasets include images and their corresponding masks as shown below:

Figure 7 Original images and their created masks.

In addition to the datasets obtained on the Internet, images have been collected in different locations, so that the capture could be adapted to the training needs of the model used. In this sense, images have been obtained from the following locations:

Figure 8 Images collected with drone in different locations (in order from left to right and up to down): AUVISA, Ciudad Real (Spain), La Paz (Bolivia), A-31 highway (Spain), Níjar HST (Spain)

Once the images were collected, the model was trained. Because of its proven success on similar previous experiences, it has been decided to design a CNN with U-Net architecture. Furthermore, in order to make the CNN as configurable and adaptative as possible, a modular neural network has been designed, composed of the next sub-modules:

- **Residual block**: the residual block is the main and basic building block of our model. It is composed of convolutional layers followed by batch normalization and activation function layers. In order to allow a better propagation of different levels of information of the image, this block incorporates a skip connection (residual connection that connects the input and the output of the block).

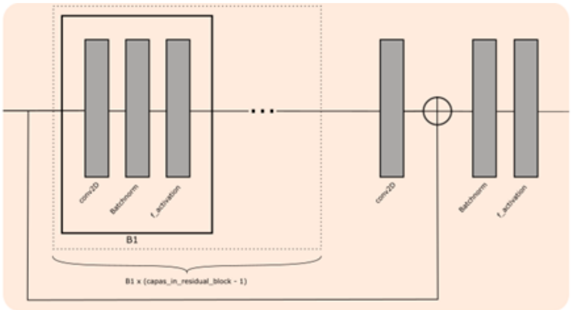


Figure 9 Residual block.

- **Encoder/Decoder block**: next, each level of the U-Net is composed by encoder/decoder blocks that gather several residual blocks into a single building unit. These blocks are connected between them via upsampling/downsampling layers and residual connections.

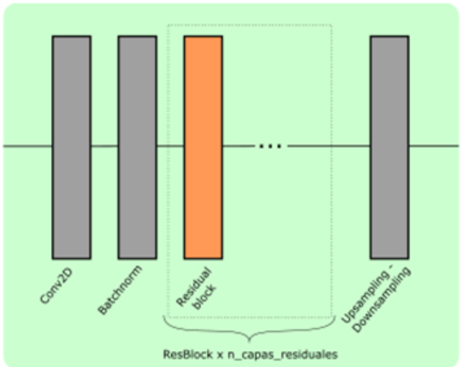


Figure 10 ED block.

- **Ensemble block:** the output of the CNN consists of an ensemble block. The main idea of this part of the model is that, starting from the information extracted from the neural network (final feature map), the combination of several small models will perform better than a single big model. Thus, several small models based on the combination of residual blocks are combined in parallel at the output of the model.

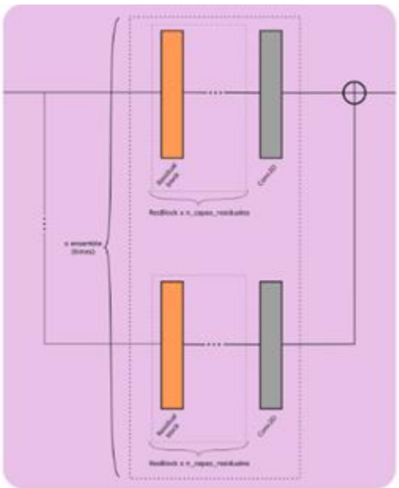


Figure 11 Ensemble block.

- **Model:** the final designed model will be based on a combination of the previous blocks. First, there is the U-Net architecture built with encoder/decoder blocks, followed by the ensemble block. The main advantage of this configuration is that the architecture of the neural network can be easily configured and optimized as if it was any other hyperparameter of the neural network.

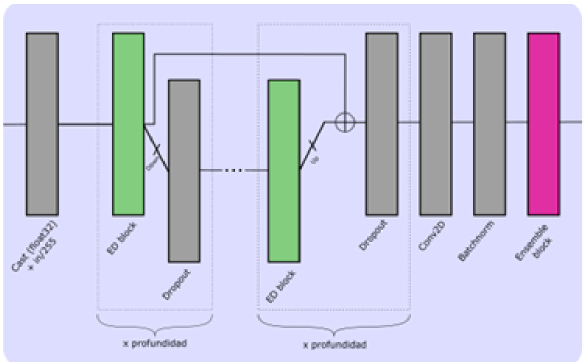


Figure 12 Neural Network (NN).

In order to obtain the best architecture of the neural network and the best configuration of parameters, a "pseudo" grid search has been carried out ("pseudo" because not all the possible combinations have been tested). In this way, several combinations of the components of the neural network and hyperparameters (learning rate, activation function) have been tested out, and the best one was selected. The following range of parameters has been considered:

- Input of RGB images of size 416x416x3. In some cases, in addition to the RGB images, the insertion of the images in HSV format (input of size 416x416x6) has been tested.
- Number of filters of the convolutional layers: The number of filters of the first layer ranges from 10 to 20.

- Depth: the tested depths of the U-Net part of the neural network range from 2 to 4 (2 to 4 downsampling/upsampling layers).
- Regularization: L2.
- The influence of batch normalization layers after each convolutional layer has been also tested.
- The number of layers of the residual block ranges from 1 to 3.
- The influence of downsampling by maxpooling vs downsampling by a convolutional layer with stride 2x2 has been tested.
- Filters of the convolutional layers of size 3x3 and 5x5 have been tested.
- Data augmentation: brightness augmentation and affine transformations.
- Loss function = cross-entropy + ɑ • IoU; with ɑ ranging from 0 to 1.

The model has been trained with 2879 images, validated and tested on a set of 159 images. The dataset has been made with cropped images from the Deep Crack dataset without resizing. In addition, in order to get the best configuration of hyperparameters, all the models have been trained 20 epochs.

The first results were obtained after the hyperparameter optimization process, with the following parameters:

- Three downsampling/upsampling phases (depth of three).
- Initial input layer with 16 filters of size 3x3. Notice that after each downsampling layer, the number of filters is multiplied by 2, and after each upsampling layer, the number of filters is divided by 2.
- Batch normalization layers after each convolutional layer, no L2 regularization and ReLU activation function.
- The model is trained with data augmentation (brightness augmentation and affine transformations).
- Loss function = cross-entropy + 0.3 • IoU.
- The HSV image has been combined with the RGB image at the input (input of size 416 x416 x 6).

A model with these parameters has been trained 60 epochs on a dataset composed by 2879 images, obtaining an IoU score on the validation set of 0.7308. In Figure 9 it is shown the inference results of this model:

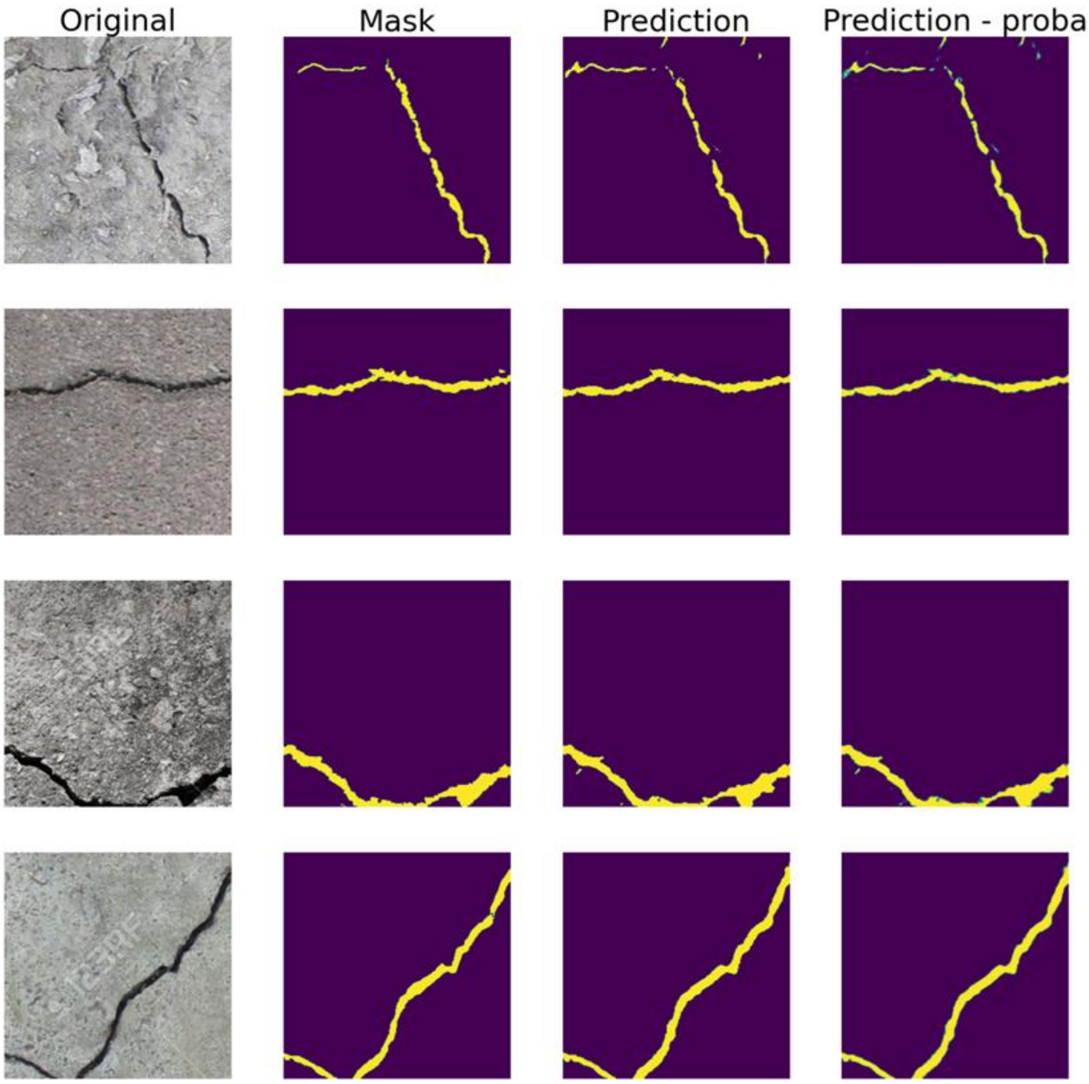


Figure 13 Result of the model trained 60 epochs.

The above figure shows the inference of our model on some images of the test set of the Deep Crack dataset. The first column shows the original image, the second shows the mask, the third one shows the prediction and the fourth column shows the predicted probability of a pixel of being a crack (this ranges from 0 to 1). As shown, our model success on the faced task. However, dark parts of the image are predicted as cracks even when they are not cracks (first image). This happens because in the original dataset, most dark parts of the image are cracks.

UC1-Demo1 has generated two components:

- Versal (VCK190) inference component
- Report generation component

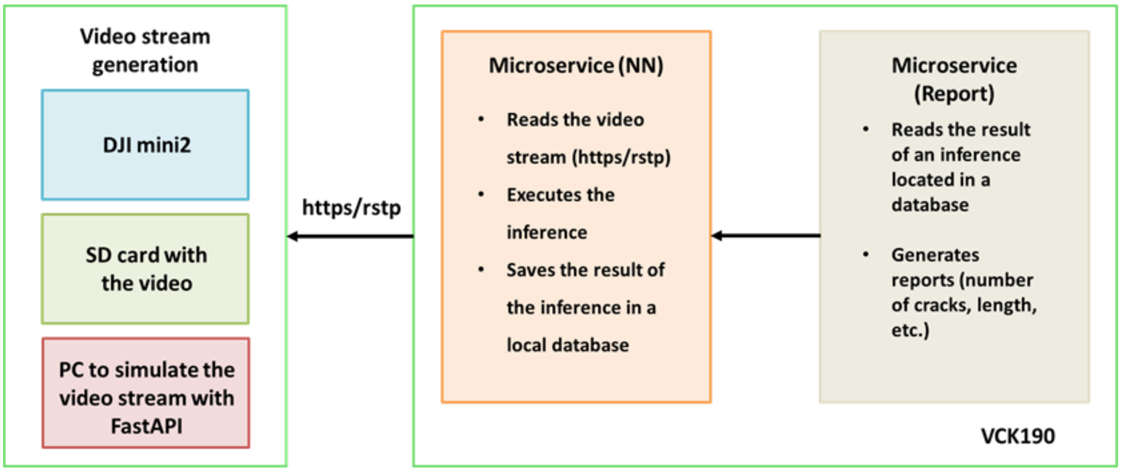


Figure 14 UC1-Demo1 components architecture

4.1.3 Demonstration,Verification and KPIs

4.1.3.1 Cognitive Awareness

The UAV will be controlled by an experienced operator, and the designed AI system will automatically process and analyze the images to gain knowledge of structural defects that may pose a future hazard to the construction. The designed system is based on a Convolutional Neural Network (CNN) with U-Net architecture and residual connections. In addition, the system has been improved by using image augmentation techniques (translations, rotations, superposition of different textures, etc.) and a manually labeled dataset. The training of the model has been done offline and several variations of the proposed architecture have been tested to better adapt the system to the use case. Also, the inference will run online and in real-time on the edge. In addition, the images taken by the UAV and the inference made by our model will be uploaded to the cloud, so they can be used by a structure's expert.

The system will be considered successful if it helps the technicians to better detect cracks. This can be measured in several ways. First, the quality of the Deep Learning Model will be evaluated with metrics such as IoU score and Categorical Cross Entropy. This model should reduce the costs and accidents inherent to "traditional" inspection methods that make use of special machinery. Moreover, it must help the technicians to better detect the number of cracks. Thus, it must result in a cognitive node that is aware of the defects of the structures (cracks) and introduces an improvement over a "traditional" method. This will be verified by technical metrics and, in a later stage, by the technicians who use this tool.

4.2 UC1 – Demonstrator II - UAV supervision of critical structures

During 2021 in Spain, 68.454 occupational accidents were recorded on construction sites, of which 118 people died, with construction being the sector with the highest incidence rate of fatal accidents. These data demonstrate the clear need to use and improve personal protective equipment and collective protection systems to make construction sites safer. The UC1-Demo2 is born from the double need that arises in construction sites, on the one hand, the reduction of occupational accidents and, on the other hand, the digitization of current health and safety systems. Thus, the demonstrator aims to reduce possible conflicts between the workforce and the machinery during their movements, as this is one of the main sources of accidents in construction field environments.

4.2.1 Hardware Platform Big Picture

Through the deployment of a network of personal wireless sensors in the workforce and individual devices on the machinery, a safe environment will be created on the construction site. The devices allow two safety radii to be established depending on the proximity between the workers and the machinery, so that a vibration and alarm is emitted on both devices in the event of a dangerous approach. A medium danger radius is defined, where workers must be aware of the danger and act accordingly, and another high danger radius where the machinery must stop instantly in the event of an imminent accident. In addition, the machinery vehicles are equipped with GPS positioning so that their movements within the site can be recorded.

The devices chosen were those provided by the Linde Safety Guard system from LINDE, whose functional description fits perfectly with the requirements established by the UC1-Dem2: "the Linde Safety Guard is a wireless assistance system for protecting people and objects in defined danger areas in industrial environments. The system wirelessly measures the distance between the component mounted on the industrial truck, components at fixed positions in the working area and the mobile components that individuals carry on their person. In this way, the assistance system can effectively use LED displays, warning sounds and vibrations to warn people of danger and help to avoid potential collisions with industrial trucks. For an early and effective warning, the Safety Guard display unit has two warning zones. The extended area can be configured in such a way that it covers a cone-shaped area to the front and rear. Within this area, the direction towards people with a mobile warning unit is displayed. The immediate vicinity covers a circular area immediately around the industrial truck. The size of the areas can be configured and must be adapted to the work environment before initial commissioning".

The information collected will be gathered and processed through data aggregation techniques, then fed to the corresponding ML models that will make their respective predictions. The outcoming data from the models and the processed data will then be represented through dashboards with information on alarms and machinery positioning, among other information, which will be used for a posteriori analysis to improve health and safety on site, and the generation of predictions of the most probable alarms, which will contribute to take quick measures.

The following components from WP4, WP5 and WP6 will be used in UC1-Dem2

Table 2 List of Components in UC1-Dem2

Component ID	Component name	Usage of component
WP4T44-05	IoT Gateway	Edge gateways will be installed for security and traffic ingress control purposes.
WP5T54-01-01	MLBuffet	MLBuffet is an open-source ML model server that allows easy storage and deployments of ONNX and Tensorflow models on the Edge.
WP5T54-01-02	Training module for ML Buffet	MLBuffet includes a Training module for the re-training and automatic re-deployment of ML models on the Edge, by using containerized environments.
WP5T54-02-01	Docker Swam	Docker Swarm is a built-in container orchestrator on Docker Engine which is available at the Edge and aims for ease of use and deployments.
WP6T54-02-02	Kubernetes-based Container Orchestrators for the Edge	Docker Swarm can be substituted with more stable and reliable K8S-based orchestrators on the Edge once the deployed solution is functional and has been tested.
WP5T54-03	MLOps Toolchain	MLOps tools will be included in the solution for agile and automated deployment of models, datasets, and other ML artifacts that require version control.
WP6T61-01-01	Operating System – Ubuntu	Ubuntu has been chosen as a widely used Linux distribution. Other distributions can be also applicable.
WP6T61-02-01	Docker	Docker Engine is a container runtime for managing, building and creating containers. Although not a hard requirement, it makes container provisioning and management agile and simple and is widely used.
WP6T61-03-02	Tensorflow	Tensorflow (V2) is an open-source ML library developed by Google, based on

		maximizing performance on GPU and TPU operations. The most popular TF wrapper library, Keras, will also be used.

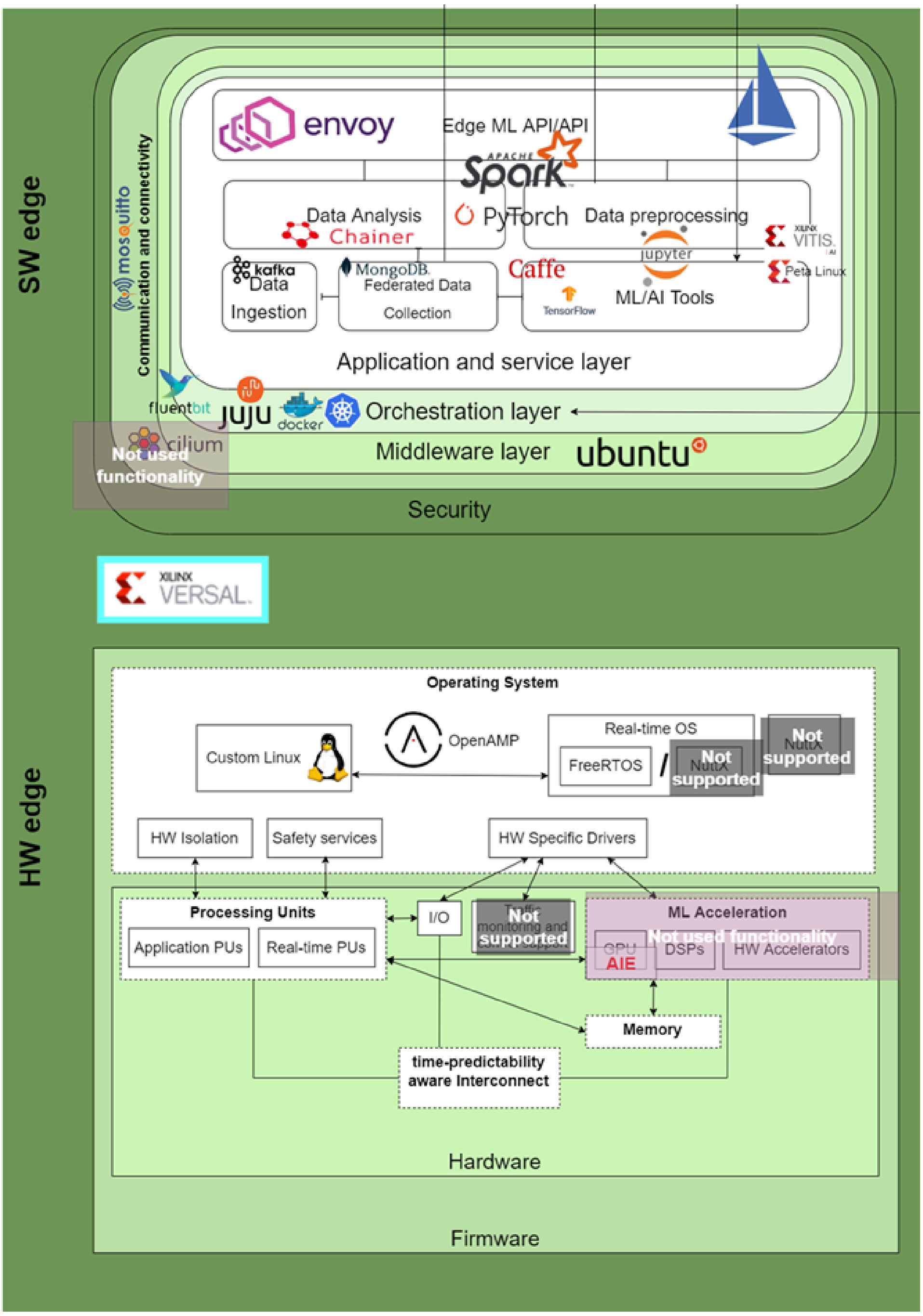


Figure 15 FRACTAL Big Picture: UC1-D2

As we can see in Figure 15, UC1-Demo2 will not use the cloud part of Fractal's big picture, focusing on the use of the components developed in the project associated with the VERSAL platform.

4.2.2 Demonstrator Setup

Demonstrator 2 is being developed in the HERNANI-ASTIGARRAGA high-speed railway project, which will link the Basque Country with Madrid. As part of this project, various concrete structures are being built. For this purpose, earthworks are being carried out where the machinery, together with the workforce, are working together on the daily tasks.

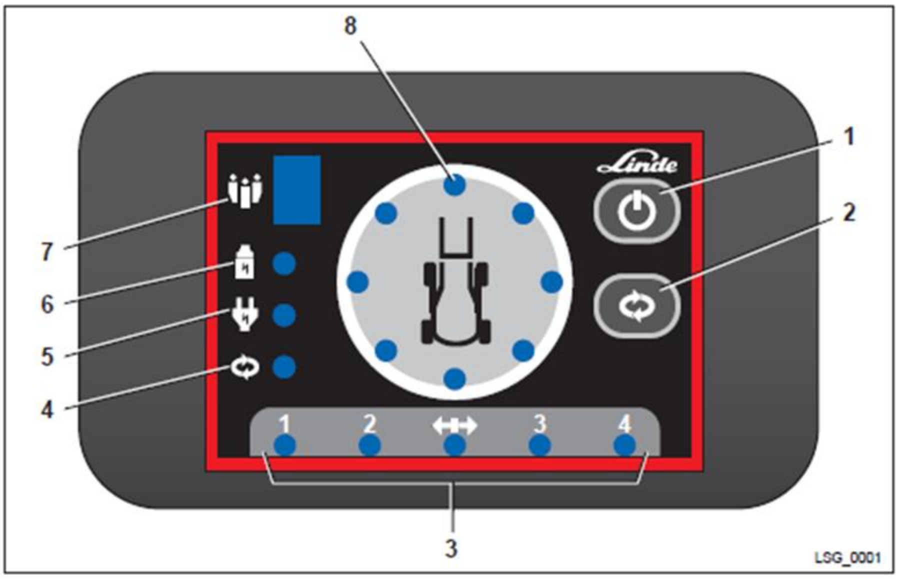
Figure 16 Demonstrator 2 scenario

In this ecosystem, a network of sensors will be deployed to collect information on possible dangerous proximity alerts through wearable devices.

This sensor network consists of three different devices:

- Device for machinery

Safety Guard display unit - Truck Unit

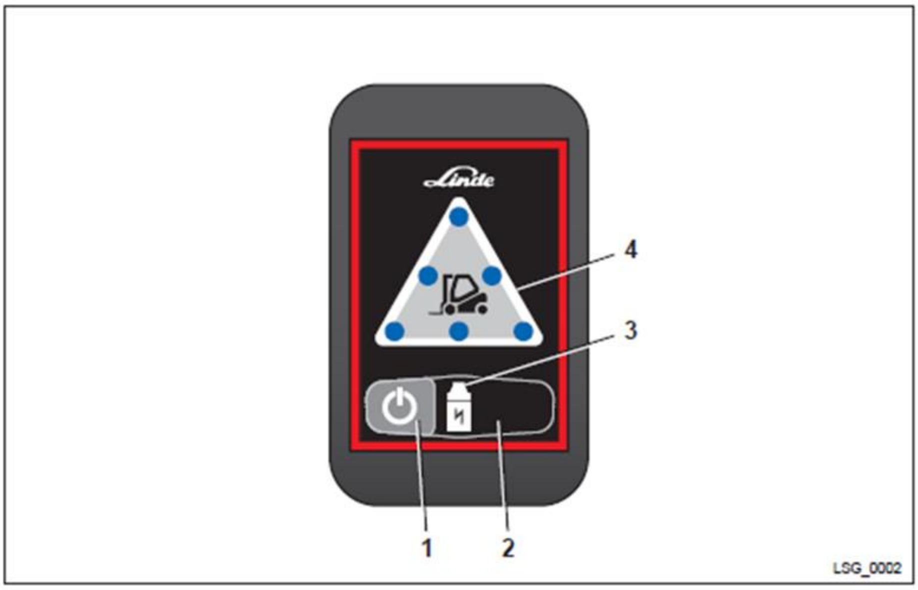


1 No function
2 No function
3 Sensors, switching output (middle)
4 Pairing active
5 Power supply
6 No function
7 Number of persons in the warning zone
8 Warning zones

- Machinery GPS

- Workforce devices

Safety Guard portable warning unit - Portable Unit



1 Switch on
2 No function
3 Rechargeable battery charge state
4 Warning zone

For the deployment of the pilot project, various procedures have been carried out with the construction company (Sacyr, S.A.) to ensure the success of the installation and subsequent monitoring of the technology. In relation to the machinery, it has been necessary to install three clamps for power in the machinery to be used. There are three types of machines, two of which will be equipped with sensorisation for the pilot project:

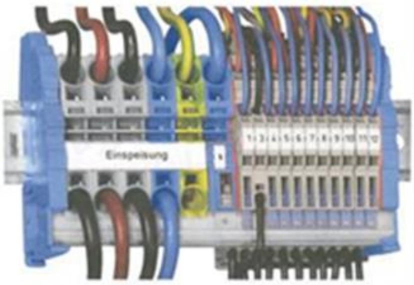
Vehicle	Self-propelled crane
Brand	LIEBHERR
Model	LTM1050/1
Location	Central viaduct of Hernani
Use	Steel framework elevation in the viaduct deck: Steel bars with different length (up to 12 m)

Vehicle	Self-propelled crane
Brand	LIEBHERR
Model	LTM1060/2
Location	Central viaduct of Hernani
Use	Prefabricated elements elevation

Vehicle	Backhoe excavator
Brand	CASE
Model	WX 185
Location	Diversion 3 excavation

Use	Land movement (excavation, load, widespread land, landfill, etc.)	

Work has been carried out to prepare the machinery for the next connection of the devices by means of wiring.



UC1-Demo 2 has generated three building blocks for AI components:

- **Alert predictor**: the model determines if the relative positions of workers and machinery constitute a hazardous situation.
- **Alert classifier**: It tells the user the nature of the alarm (machine-machine, worker-machine, etc.).
- **Anomaly detector**: It detects what events in a time-series formatted dataset have special features. This model tells when the algorithm failed or succeeded in its predictions.

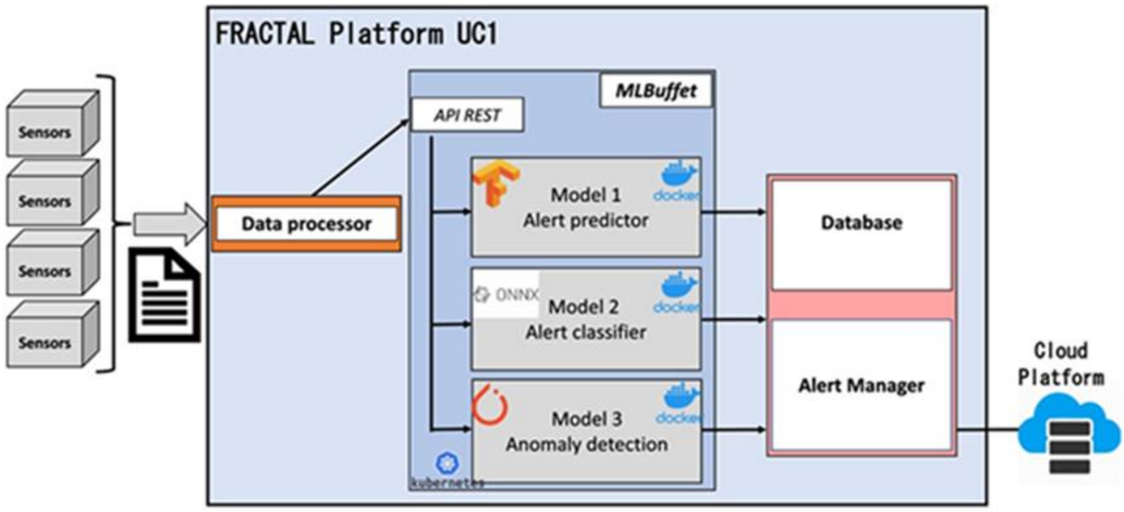


Figure 17 UC1-Demo2 data flow architecture

This architecture shows how the different AI Building blocks interact with each other. Firstly, the information from the LINDE sensor network goes to the Data processor, which is a system that formats the data accordingly to the models' expected input, and splits the different entries into HTTP requests to be sent to the MLBuffet's REST API holding the models. The models consist on Neural Networks trained with the Tensorflow v2 library from the historical gathered data from the LINDE sensor network. Once trained, the models can be retrained to improve their metrics and performance once enough data are available.

This REST API collects the inputs and feeds them into the different models, providing the results of the inference into a Database and the Alert Manager simultaneously. The Database will keep the data cached for a period of time, to make sure that the data are backed-up and stored for historical dataset creation. The Alert Manager is configured with a set of rules based on the models' outputs, and will generate alerts for the platform operator to take the appropriate actions.

These actions vary depending on the nature of the alerts, which could be high risk of hazardous situations predicted, type of alert predicted when a risk is encountered, or anomalous behavior being detected. Finally, the gathered data, together with the predictions and the LINDE raw data are collected and sent to an external Cloud Platform, which will only serve as data history, and post-processing of these data for further model refinement or re-training.

Notice that the external Cloud should be differentiated from the Fractal Cloud Platform, as the latter serves as a support for the Edge AI operations, but in this case, no support is needed from external Clouds, as the Fractal Platform designed for UC1-Dem2 should be independent and should work be able to work in environments with no internet connection.

This technological stack will be installed and deploy in a containerized approach, into the VERSAL board. Other Xilinx boards can be used as long as they support Linux distributions (Ubuntu or PetaLinux custom distributions) and have an ARM processor architecture.

4.2.3 Demonstration and Verification of KPIs

4.2.3.1 Cognitive awareness

The cognitive awareness capabilities will be enabled by a sensor network deployed over the machinery and wearable devices. These sensors will collect data about the interactions between the construction workers and the machinery, whenever an approach that can be considered hazardous happens between them. Notice that these sensors never collect personal information from the workers nor interact with them in any way (wearable sensors do not trace the positions of the workers out of alarm areas, do not collect information on how much time the worker has been on each location), so no GDPR rules are broken. This is a relevant aspect as data obtained from this sensor network is anonymized by default (the sensor ID is the only information present in the data, but not the worker's personal information), so in case a cyber-security leak happens, no personal data will be exposed.

4.2.3.2 Communication

The ML models must provide a quick response, and having real-time inferences is a must, because the safety of workers will be improved if a fast response is provided to forecasted hazardous events. These real-time capabilities are achieved by having a sensor network deployed at the edge, which collects data from the scenario, positions of workers and machinery, and sends alarms to the edge controller whenever the workers approach a machine from a dangerous area being monitored. These data can then be processed directly at the edge without a strong dependency to the cloud, which results in faster responses.

5 UC2 – Reinforcement Learning for Optimal Thermal Management of BEVs

5.1 Hardware Platform Big Picture

Currently, thermal management of Battery Electric vehicles (BEVs) is performed using traditional model predictive control (MPC) strategies which are modeled with the help of mathematical equations. These control strategies tend to fail or underperform when the vehicle is driven in scenarios which do not obey the boundary conditions set up by the control strategies. To solve this problem, and to make sure the optimal thermal management of a BEV is always maintained, UC2 has chosen to use reinforcement learning, a sub field of machine learning and the current state of the art in the field of Predictive Control. As such, UC2 demonstrates FRACTAL's adaptability feature.

It is worth noting, that aside of UC2 – mainly addressing adaptive thermal mgmt. at the application level – lower levels of the technology stack, also deal with adaptive behavior. Beyond the scope of UC2, for example, thermal management of the node (hardware) is also particularly relevant for safety-critical systems, where the execution of a system application generates heat in the node. Through the Hierarchical Adaptive Time-triggered Multi-core Architecture (HATMA), techniques such as Dynamic Voltage and Frequency Scaling (DVFS), clock gating and task re-distribution protect the node when cores exceed a thermal threshold. For example, while streaming inference data to the cloud does not require high-performance computation, running a reinforcement learning agent for inference on the edge does. Constantly running a core at higher performance levels naturally leads to overheating of the node. Therefore, the frequency of cores is scaled, or application tasks are re-distributed to manage node temperature without degrading system performance. In addition, HATMA protects the node where thermal constraints must be satisfied under all compute requirements and environmental conditions, contributing to the adaptability of the node.

Reinforcement Learning (RL) is "A way of programming agents by reward and punishment without needing to specify **how** the task is to be achieved" (Kaelbling, 1996). **Predictive Control using RL** is a method of process control that is used to control a process using a trained RL agent which has learned the dynamics of system through data. In this case the RL agent will be trained with hundreds or thousands of scenarios through simulated data and therefore enabling the agent to adapt to unseen scenarios based on experience it gained in the training process. In general, the RL process is formulated as a Markov decision process (MDP) which can be seen below

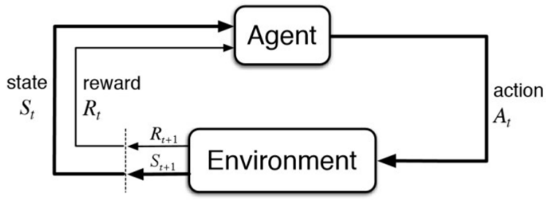


Figure 18 RL represented as a Markov Decision Process

In UC2, the RL agent will be trained on pre-existing data and will be deployed on the edge for inference, where the RL agent accumulates inference data as experience buffer and this experience buffer is streamed to the cloud for further improvement of the RL agent. To achieve the whole process of training, inference and re-training, UC2 uses the following components from the FRACTAL platform:

Table 3 FRACTAL Components currently used in UC2

Component ID	Component Name	Component Usage
WP3T34-03	Versal Model deployment layer (Vitis AI blocks)	Model deployment on the Versal APU + DPU control from model repository images
WP5T52-01-01	Data Ingestion Service using Kafka	Ingestion of streaming data from Edge devices to cloud
WP5T52-02-01	Raw data Object storage service	Storing of raw data on OVH S3 object storage
WP5T52-03-01	Data transformation	Data transformation of raw data into training dataset using Spark
WP5T52-04-05	Datasets version control	Dataset version control in the cloud
WP5T52-04-07	Images repository	Docker Image repository for storing different python docker jobs
WP5T52-04-08	Model repository	Model repository of storing different models
WP5T52-04-09	Machine learning pipeline	Pipeline for orchestrating ML training
WP5T52-05-02	Data pipelines and workflows orchestrator	Pipeline for data loading and transformation
WP5T52-07-01	Kubernetes-based cloud platform container orchestrator	Managed Kubernetes Service for containerized services orchestration.
WP5T52-07-02	Cloud container orchestrator services access	Allows to expose kubernetes services and routing requests to different endpoints.

WP6T61-02-01	Docker	OS-level virtualization to deliver software in packages called containers.
WP6T61-02-02	Mosquitto	open source (EPL/EDL licensed) message broker that implements the MQTT protocol
WP6T61-02-03	Microk8S	low-ops, minimal production Kubernetes and it is an open-source system
WP6T61-02-05	Ingress	API object that manages external access to the services in a cluster. Provides provide load balancing, SSL termination

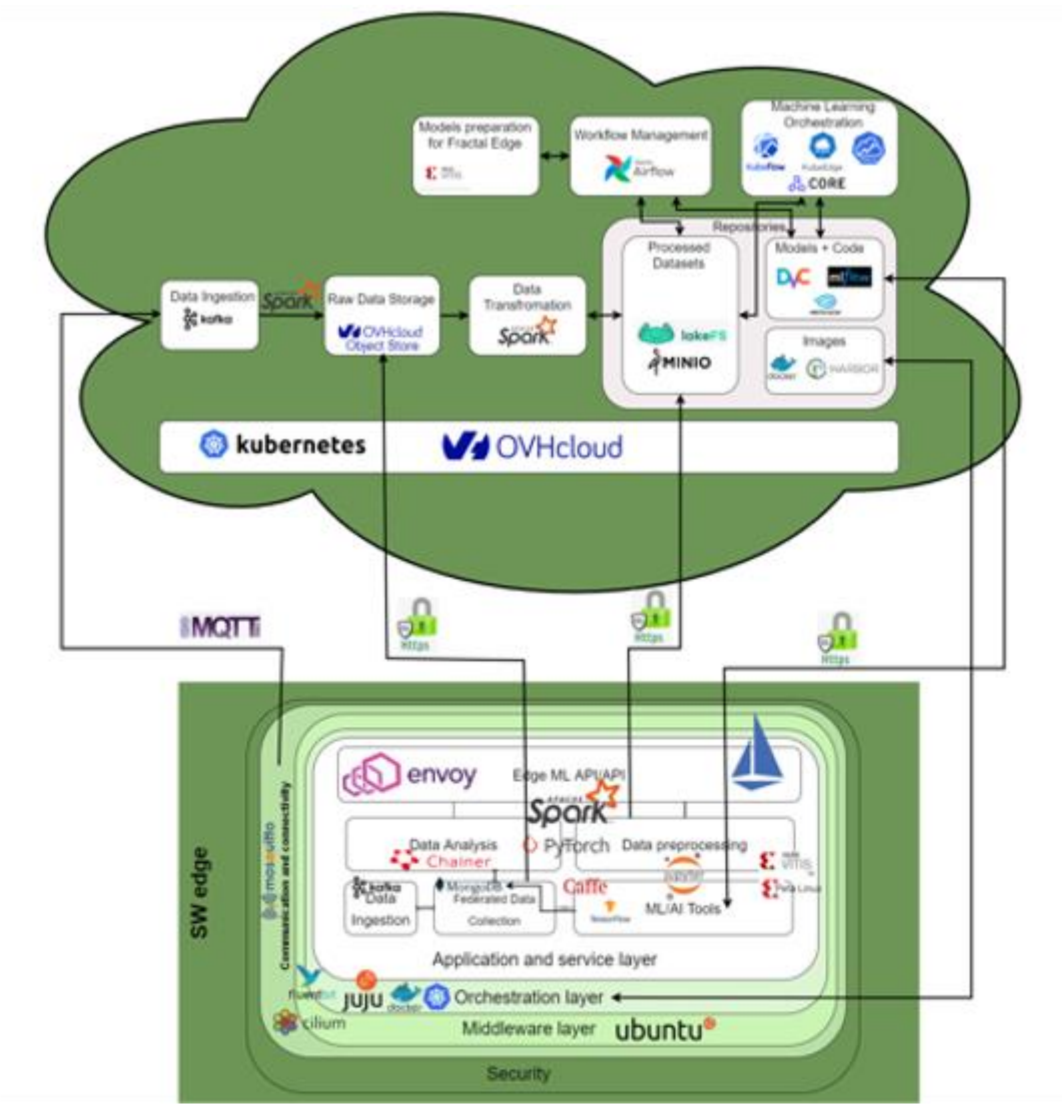


Figure 19 UC2 Components architecture

5.2 Demonstrator Setup

In UC2, as mentioned in section 5.1 the agent is trained using the MDP formulation with the main goal of obtaining a control policy that will maximize the observed rewards over the observation time of the agent (often called episode). For UC2, the environment (mentioned in Figure 18) is the plant model which models the dynamics of the BEV. The plant model (shown in Figure 20) is created using MATLAB Simulink which is later compiled into a Functional Mockup Interface Unit (FMU). The Reward function is basically the feedback from the environment, which in UC2 is the Cost-function based on thermal comfort, engine working temperature, system energy consumption.

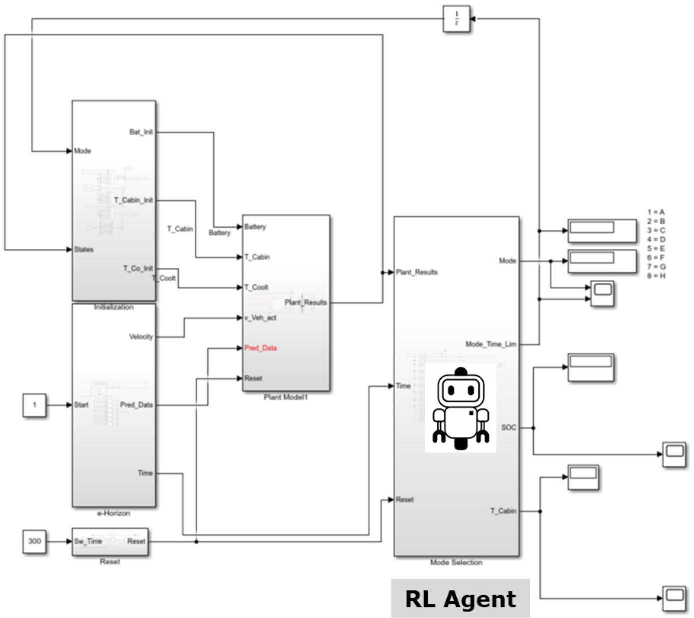


Figure 20 Plant Model with RL Agent connections in MATLAB Simulink

In UC2, to train the RL Agent, vehicle drive cycle data have to be used. The drive cycle data generally consist of the vehicle sensor data, the traffic information, the geolocation information and the topographical information. The drive cycles are generally collected when the test BEVs are driven on certain routes or test tracks. To train the RL Agent, UC2 uses "Ray (RLlib)" an open-source python package for formulating the RL MDP process and "FMPy" an open-source python package for reading a compiled FMU of the plant model. A typical training loop is shown below.

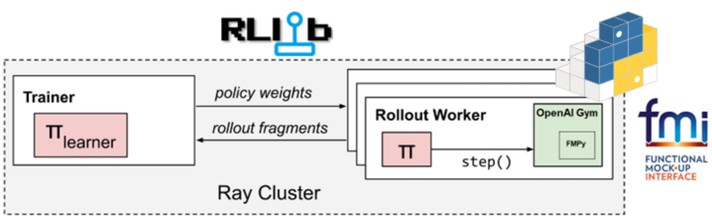


Figure 21 Training Loop for RL agent

In UC2, after training of the RL agent with enough drive cycles, the model has to be deployed onto the edge (Xilinx Versal board) for inference. All the data which the agent received is stored in an experience buffer which is streamed to Kafka for re-training of the RL Agent. The RL Trainer with the FMU vehicle model trains using data coming from various vehicles and evaluates the best RL policy and updates the RL model if necessary (Over The Air Update). This process basically closes the Inference and re-training cycle of the RL model development. The whole process entails the following steps:

1. Train an initial RL agent with FMU plant model on the cloud
2. Use 2 to 3 validation drive cycles on the Xilinx board to simulate a vehicle driving
3. Stream through the validation drive cycles with RL agent in inference mode
4. Store the online inference results of the model on the Xilinx board as experience buffer
5. Stream the experience buffer using MQTT to cloud residing Kafka service
6. Collect the Experience buffer from Kafka and store it in OVH Object storage and transform it as an experience dataset and store it in LakeFS
7. Trigger re-training on the cloud, when sufficient amount of experience dataset is available. Then evaluate models using policy evaluator
8. Based on the policy evaluator, send an updated model back to the IoT platform

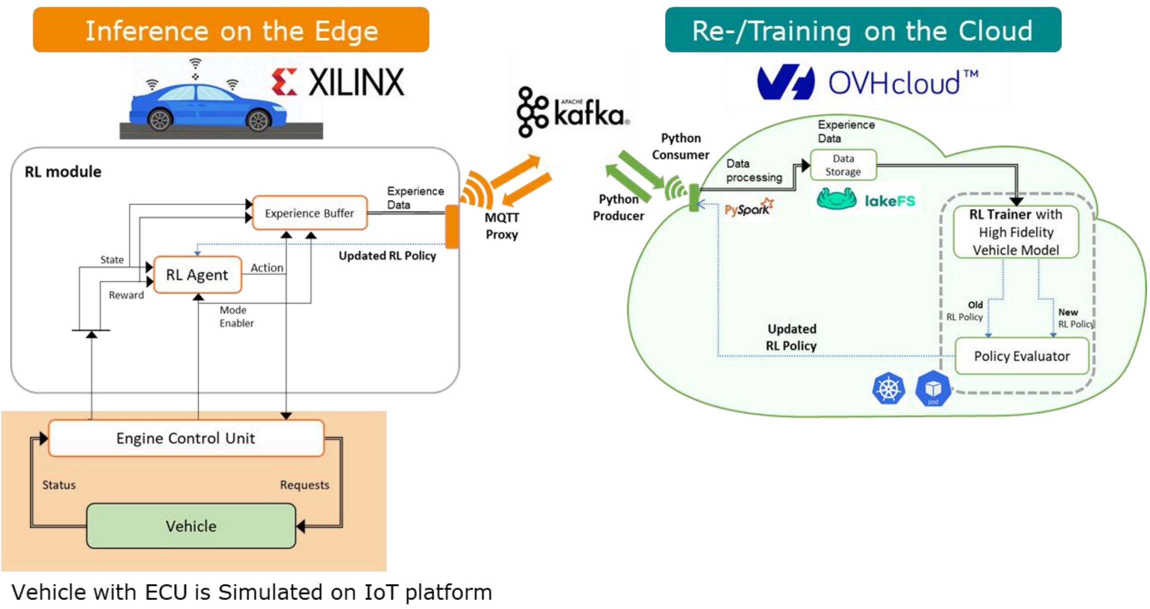


Figure 22 Complete UC2 RL demonstrator setup

This whole process in Figure 22 depicts the UC2 demo for the RL predictive control. Alongside the components listed in Table 3, we plan to use the following components and experiment with them in the next phase:

Table 4 FRACTAL components currently under consideration for next phase

Component ID	Component Name	Component Usage
WP4T42-02	Versal RPU access to AI acceleration	Enable local AI acceleration deployment from RPU
WP4T44-05	IoT Gateway	IoT network Gateway for external communication monitoring
WP4T44-06	GDPR Compliance	Data Protection Impact Analysis
WP5T52-04-01	Models version control	Version Control for models in the cloud
WP5T52-06-01	Model preparation for Fractal Edge (Versal Xilinx Vitis AI)	Workflows to compile models for Versal with Xilinx Vitis AI, add containerized toolchain to the cloud
WP5T54-02-02	Kubernetes-based Container Orchestrators for the Edge	Open-Source orchestrator for cluster management and container orchestration.
WP6T61-02-08	Prometheus	systems and service monitoring system
WP6T61-02-12	Fluent Bit	open-source Log Processor and Forwarder

5.3 Demonstrator and Verification of KPIs

5.3.1 Adaptability KPI

UC2 contributes to the development of AI-enabled control strategies and enables a controlled system to continuously evolve depending on perceived context. This basically highlights FRACTAL's technical feature of "Adaptability". This feature can be verified by comparing the control policy achieved by the trained RL Agent to the control policy obtained using Dynamic Programming (DP) which is employed in traditional Model Predictive Control (MPC). The cumulative error of the RL policy should fall within the range of 5% of the control policy generated by DP. To achieve this the RL agent would be trained with different drive cycles till convergence. The RL agent's neural network architecture will also be adjusted based on crossvalidations using different validation drive cycles. This makes the RL agent generalize well on different driving scenarios. This enables UC2 to achieve another very important business KPI i.e. "Responsiveness to unseen scenarios".

The development of RL agent can be parallelized, and can be trained on GPU accelerated cloud Virtual Machines. This makes the development of RL control policy much faster compared to the development of traditional MPC. This would thereby reduce the calibration effort. This business KPI can be verified by measuring the total development and calculation time needed for development of a trained RL agent and comparing it to the time taken for obtaining a traditional calibrated model using MPC.

5.3.2 Cloud Communication KPI

UC2 also contributes to the FRACTAL feature of "Cloud Communication" as it uses Apache Kafka for data routing when it comes to re-training of the ML model. The cloud communication happens, during the inference mode of the RL model. All the data obtained during the inference mode is stored in an experience buffer on the IoT platform. This experience buffer is streamed to Kafka, using this experience data the RL model degradation can be measured. To verify this feature, the streaming frequency to Kafka has to be monitored and measured in "Prometheus" a monitoring tool. In this tool streaming frequency metric can be defined which would send notifications to the developer when it is less than 100 messages per day per IoT device.

6 UC3– SmartMeter led by ACP

State of the art smart meters directly read the meters through digital interfaces (I2C, SPI, etc.), but cannot read older analog meters. The platform that is proposed within UC3 can read digital sensors thanks to a rich variety of digital interfaces, but at the same time, can also read a meter by taking a picture of the meter display and analyzing it.

6.1 Hardware Platform Big Picture

The existing platform (the ACP modem) consists of the PULP node paired with a digital front end (DFE) plus the full RF part of a wireless transceiver, all in one chip. This platform allows to establish a wireless connectivity to the cloud through NB-IoT or GPRS. Thanks to its peripherals, it also allows to read digital meters. It does however not have the compute power to efficiently analyze an image. For this task, the fractal node is used. And hence, throughout this project the smart meter prototype will consist of two platforms, the modem plus the fractal node. The final goal is to merge the fractal platform and modem into a single chip solution that allows to establish wireless connectivity and analyze images directly on the node. To make the platform secure, it requires different security features such as a secure boot process and encryption/decryption services which will be demonstrated separately.

The following components from WP3-6 will be used in UC3:

Table 5 List of Components in UC3

Component ID	Component name	Usage of component
WP3T32-02	PULPissimo platform	Base platform for fractal node
WP3T32-03	PULP trainings	Trainings how to use pulp platforms, adopt, run software on it.
WP3T32-04	FreeRTOS	Operating system running on the PULPissimo node
WP3T32-08	Real-time cache	New HW module that allows to run user application and real-time critical code from the modem on the same platform
WP3T32-12	TL2AXI protocol adapter	Protocol adapter to connect OpenTitan to PULP type systems
WP3T32-11	Smart interrupt unit	New HW module that allows to run user application and real-time critical code from the modem on the same platform

WP3T36-03	Nuttx on PULP	Alternative operating system running on a RISC-V platform.
WP4-T41-03	Low power services	Services to save power on the PULPissimo node

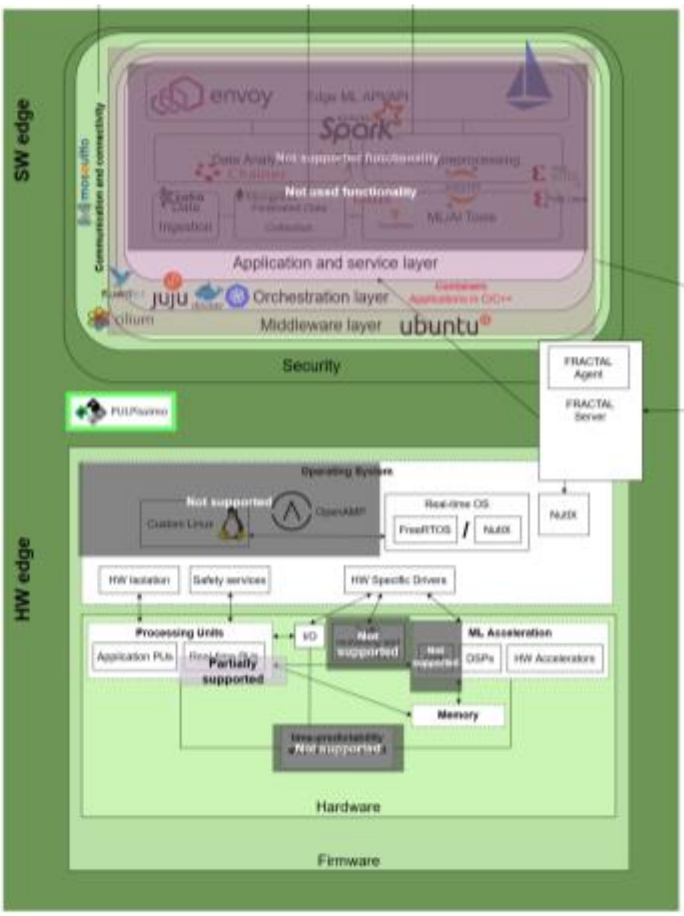


Figure 23 UC3 Big Picture

As can be seen in Figure 23, the focus in this use case is on hardware development on the platform and low-level software.

6.2 Demonstrator Setup

6.2.1 Demonstrator setup to show security components

The free and open-source OpenTitan platform was ported to a genesys2 FPGA board to be able to develop a secure boot process on the board using the OpenTitan hardware blocks. The programmable logic consists of a RISC-V processor, memory, and several hardware accelerators for hashing, encryption, etc.[1]

The main blocks that are required for a secure boot process are: a hash accelerator to compute the digest of the firmware, the OpenTitan Big Number Accelerator (OTBN) for the elliptic curve cryptography (ECC) functions signature verification functions, and the RISC-V processor.

The full boot process looks as depicted in Figure 24. Upon power up and reset release, the code in the ROM is executed. This first stage boot process verifies the first section of the flash content called *ROM_EXT*. Upon successful verification, the core jumps to the *ROM_EXT*, the second stage in the boot process which verifies the application firmware and launches the application if it has been successfully verified. This two-stage secure boot process can be extended with additional stages if needed.

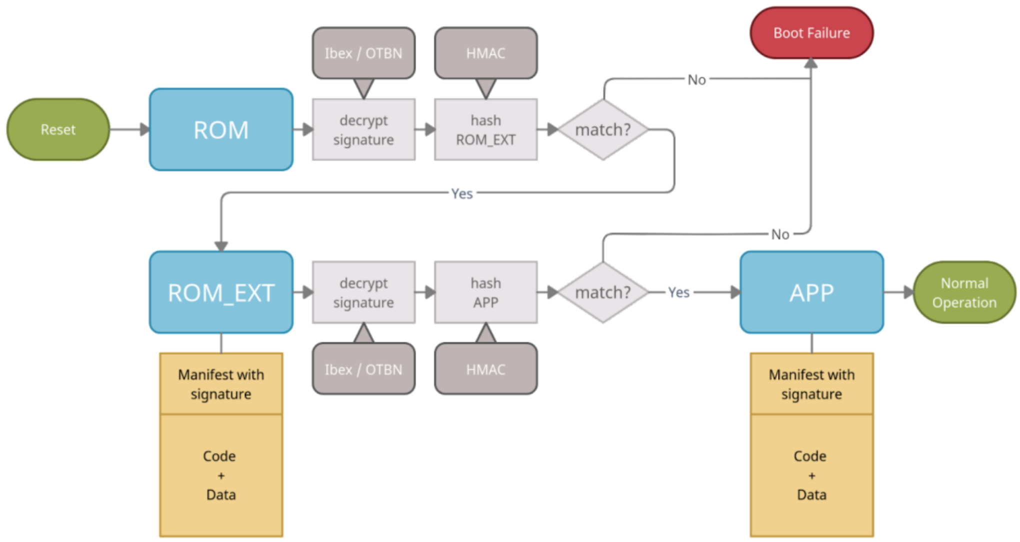


Figure 24 Secure boot process

The FPGA board with a loaded bitstream of the OpenTitan architecture is shown in Figure 25. On the FPGA, the flash is emulated with SRAM, and instead of a true random number generator, a LFSR is used to generate initial random numbers for initial secrets. The only external connections that are required apart from a power supply, are a UART interface for the command line interface and debug output and a JTAG interface to load the initial bitstream.

1 More details can be found here: https://docs.opentitan.org/hw/top_earlgrey/doc/#detailed-specification

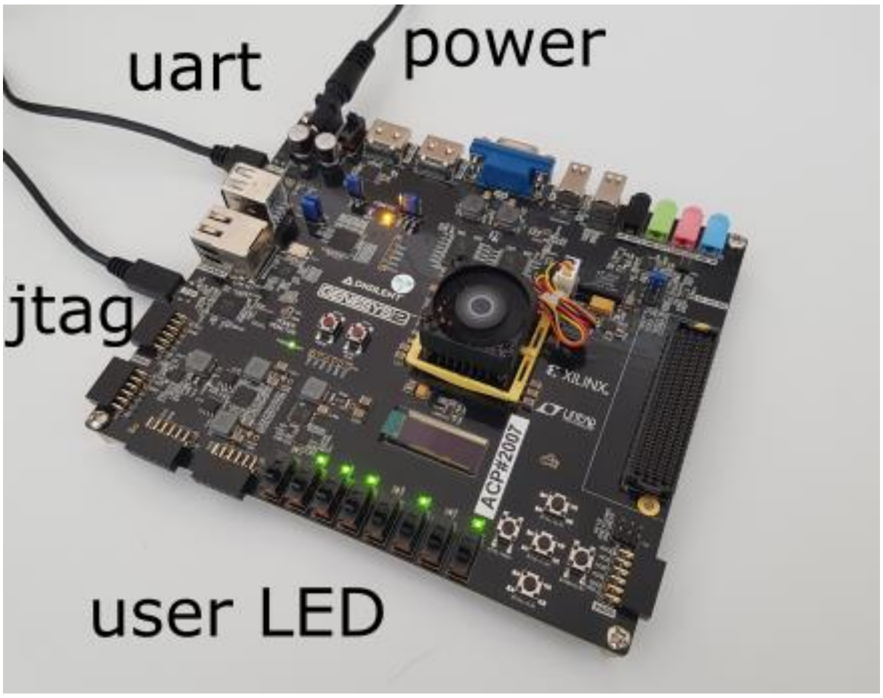


Figure 25 OpenTitan on genesys2 FPGA board

6.2.2 Demonstrator setup to show wireless connectivity and low-power components

Wireless connectivity is established with ACP's modem. It can be controlled with standard AT-commands over a UART interface. It allows to open a TCP/IP connection or directly use MQTT subscribe and publish commands to connect to a MQTT server. The setup in Figure 26 shows two boards. On the left, the PULP based FRACTAL node in form of an integrated circuit, and on the right the modem with an antenna. The two boards are connected through a 2-wire UART interface that is used to control the modem. The PULP board is also used for the ML inference demonstration that is used in UC3 to extract a number from a taken picture and allows to determine how long it takes to run the feature extraction directly on the board. The duration is an important parameter to be able to determine the battery lifetime.

Figure 26 Fractal node (left) connected to modem for wireless connectivity (right).

While this setup does not yet include the most recent hardware developments of the FRACTAL project, it can be used to verify the connectivity to the cloud and the power consumption in deep sleep and active state.

The deep sleep power is a good representative of the final deep sleep power consumption because the only circuit that needs to remain active is a real-time clock, everything else can be shut down with power gates.

The active current however is drawn twice, once in the modem and another time in the FRACTAL node which is not ideal, especially because the two platforms have a lot in common as they are both based on the PULP system. This means that a lot of area and power can be saved by merging the two platforms into one as depicted in Figure 27. Throughout WP3, a real-time aware instruction cache and an interrupt unit have been developed that allow to execute real-time critical code (the cellular protocol stack) and a user application (the meter reading) on the same platform without interfering with real-time constraints. To determine the power consumption of such a combined platform, the relevant blocks, including the real-time cache and smart interrupt unit, have been integrated in the place and route flow and the power is analyzed with the use of gate level simulations.

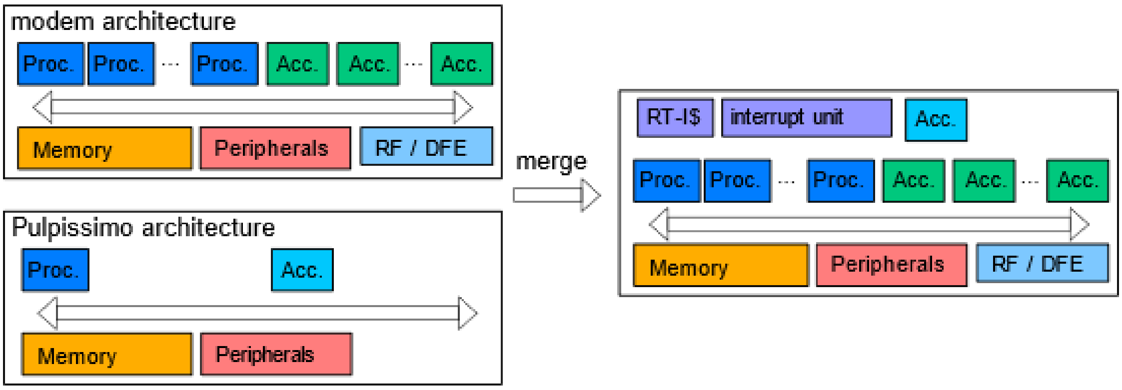


Figure 27 PULPissimo and modem platform merge.

6.3 Demonstration and Verification of KPIs

6.3.1 Security KPI

All KPIs related to security can be demonstrated and verified with the demonstrator setup of section 6.2.1. To verify the execution times, a clock frequency of 100MHz is assumed and the cycle counter of the processor can be used to determine the execution time from the start of the ROM execution until the end. On a final chip, the boot time not only depends on the execution time of the processor, but also on analog circuits for clock and power management. This time is typically in the range of milliseconds and can be assumed to be constant and allows to precisely determine the time that is required for a full secure boot process.

6.3.2 Low power KPI

The main KPI is a 5+ year battery lifetime. It mainly depends on the battery capacity and the following three parameters:

- Active time
- Deep sleep power consumption
- Average power consumption during active time

Since the size of the final device must be in the range of square centimeters, it is not possible to just use a much bigger battery. To verify this KPI, a medium sized battery with a capacity of 2200 mAh is assumed.

The active time consists of the boot process, taking a picture of the meter, analyzing it, establishing a connection to the cloud, and transmitting the data. This time can be measured with the existing setups of section 6.2.

The power consumption depends a lot on the actual tasks that are performed and can only be accurately determined once the system is complete. An upper bound for the average power consumption can however be determined by measuring different workloads on the already existing chip solutions, as well as by simulating the different modes in gate level simulations and by subsequent power analysis in the place and route tool.

To verify that a multiyear battery lifetime can be achieved, the three parameters are measured separately.

6.3.3 Connectivity KPI

To verify that a connection can be established in poor SNR conditions, the modem has been tested in the lab with the help of special equipment that can simulate different conditions. In addition, the setup of section 6.2 can be used to test the wireless functionality in different environments such as basements.

7 UC4 –Low-latency Object Detection

Machine Learning has shown huge success in the area of object recognition by exceeding the accuracy of human's eyes. Such a feature can be utilized in different industrial area to improve automation like in production lines for detection of anomalies on new products, or as auxiliary component for industrial robots to guide the motion of the arm, or as a safety observer to detect and stop the production line if a person enters in area where one be harmed from machinery.

7.1 Hardware Platform Big Picture

UC4 implements the object recognition feature as part of the Fractal platform in form of a generic building block. This is achieved by utilizing both, the Fractal Edge Node and the Fractal Cloud as part of the building block as shown in Figure 28. Fractal Edge Node runs constantly the AI model in form of inference for object detection. If a new object needs to be added for detection, then the model is pushed into the Fractal Cloud for training. The training cannot be performed on the edge side due to architecture of the Fractal Edge Node and the differences in data representation. On edge, the hardware is customized to run the data only in forward direction without possibility for feedback which is required for training. For data representation, the inference uses on edge fixed-point representation while the training applies float point for higher accuracy.

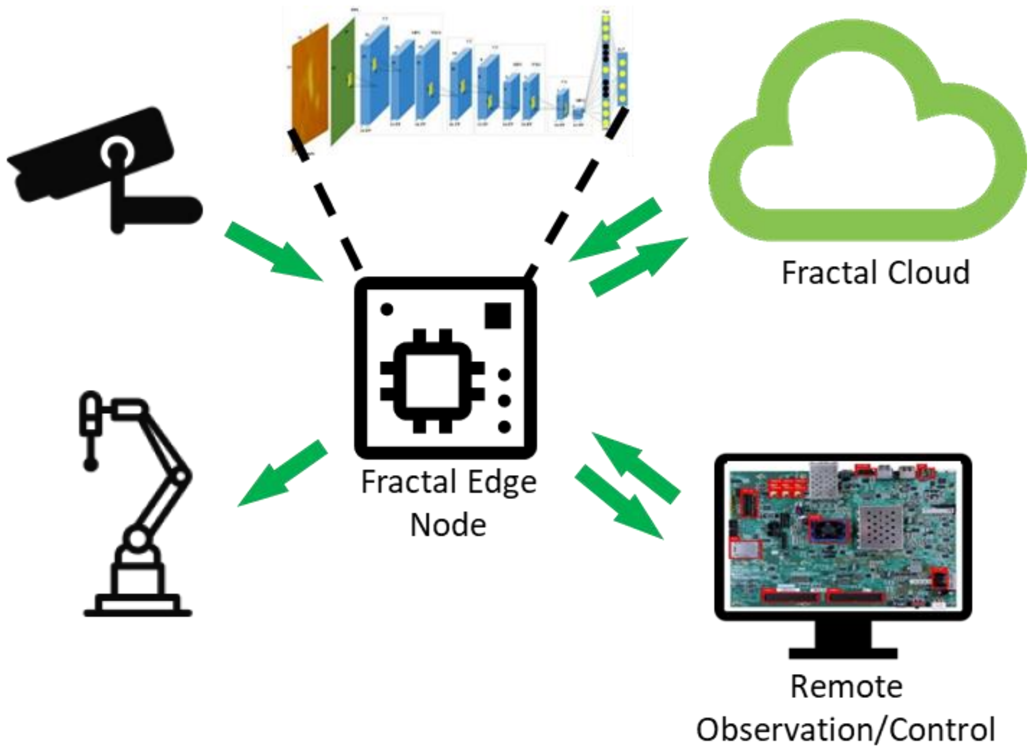


Figure 28: Fractal building block for object detection

As can be seen from the Big Picture (Figure 29) the implementation of the object detection feature in UC4 is achieved by utilizing components developed by different partners in the Fractal project. The Edge Node utilizes the components required to run the inference and communicate with the cloud, while the cloud part consist of components required for training of the AI model, preserving the previous AI model version and for keeping the data sets for new objects. Table 6 lists the names of components that are used in UC4.

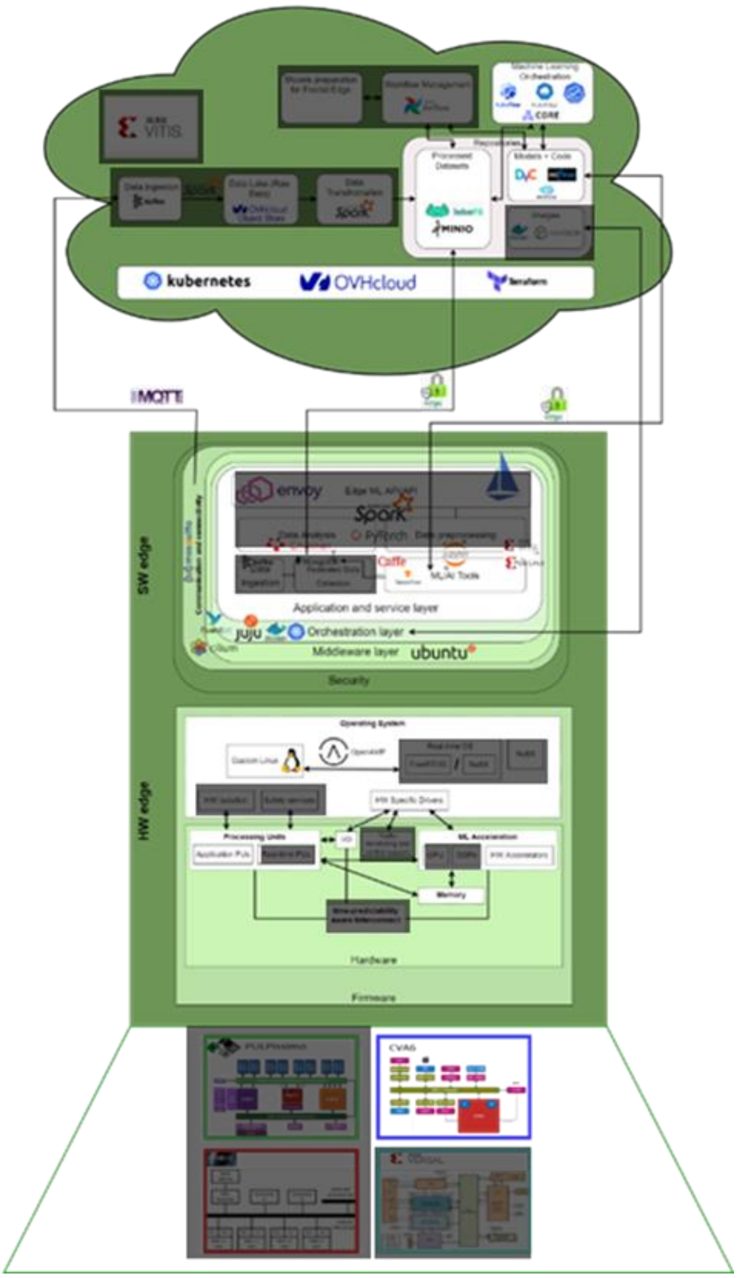


Figure 29: FRACTAL Big Picture – UC4

Table 6 List of Components in UC4

Component ID	Component name	Usage of component
WP3T32-01	HW accelerator (SIEFRACC)	HW accelerator for vision-based AI inference
WP3T33-01	Ariane RISC-V for ZCU102	Porting Ariane 64bit to Xilinx ZCU 102 board
WP3T35-01	SW driver for HW accelerator	SW driver for managing HW accelerator and data movements
WP3T36-01	Linux for CVA6 (former Ariane)	Linux support for CVA6 (both 32b and 64b): UBoot (boot loader) + OpenSBI (firmware) + Yocto (embedded Linux generation)
WP5T52-04-01	Models version control	Datasets version control, Model version control
WP5T52-04-03	S3 compatible data storage	Data Lake. High-performance, S3 compatible object (images, video) storage.
WP6T61-12	Neural network toolkit	open-source neural network framework (Darknet)

7.2 Demonstrator Setup

UC4 runs on Xilinx ZCU102 FPGA board (Figure 30) and OVHCloud. The edge part of the system with AI inference is mapped on the FPGA board. The components of the board are: CVA6 64-bit softcore processor, SIEFRACC AI hardware accelerator, SIEFRACC driver, CVA6 Linux OS and Darknet AI framework for inference.

Figure 30: Xilinx ZCU102 FPGA board

As shown in Figure 31 the CVA6 processor, HW accelerator, DMA and main memory are connected through AXI4 bus. The accelerator has in addition two signals connected directly to the CPU. One is to trigger the start of the HW accelerator, and the other signal notifies the processor when the HW accelerator has finished with execution of the convolution layer. DMA transfers the data between main memory and HW accelerator. Its configuration is controlled by Acc SW driver run on CPU.

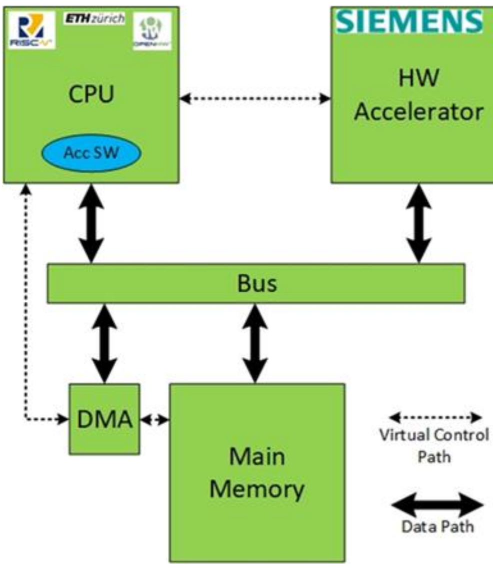


Figure 31: Block-level architecture of Fractal edge node

For retraining of the AI model, the UC pushes the AI model into the FRACTAL cloud. A new set of labeled data needs to be provided as well.

For demonstration of the object recognition feature in UC4 the following sequential steps will be performed:

- A set of input images will be collected for processing from where the objects of interest will be detected.
- The Darknet framework will perform preprocessing of the images by formatting their size to dimensions appropriate for the AI model.
- The memory layout of the processed image will be rearranged to fit more properly with the dataflow of the HW accelerator.
- The images and weight filters will also be fragmented due to small size of local memories in HW accelerator.
- The driver of the HW accelerator will prepare the descriptors and configure the DMA controller.
- The data transfer of images and weights from system memory to HW accelerator memory through DMA will be triggered.
- The start of HW accelerator will be activated.
- Transfer of the results from HW accelerator to the system memory through DMA will be triggered.
- Postprocessing of the output images will be performed.

7.3 Demonstration and Verification of KPIs

UC4 verifies cognitive awareness as a feature on the FRACTAL platform. Object recognition is crucial for cognitive awareness. It enables the system to observe the surrounding and recognize the environment. This feature can be verified through a test data set where the output from UC4 is compared to the labeled images. The cumulative error on not detecting objects for which the model is trained should be less than 5%.

The second verification is on image processing rate. On one side, the AI model requires a huge amount of processing power and on the other side the edge node is constrained on resources. To overcome this problem, Fractal edge node applies specially designed HW accelerator to increase the processing capability of the node and the rate of processed images. The goal is to achieve a rate higher than 5 fps.

8 Conclusions

This deliverable provides the description of the FRACTAL node prototypes as devised by UC1-UC4 to validate the FRACTAL *features* as identified in D7.1. The particular FRACTAL components that are deployed to support these features are shown in the context of the use case demonstrators.

The prototype implementation details are showing which part of the FRACTAL big picture each use case will actually cover. Within this coverage the prototypes are set up to derive specific metrics which in turn support the analysis of the KPIs the use cases are focused on. This determines the contribution that FRACTAL features add to the use cases application scope.

Based on these demonstrators the validation results will be presented in D7.3 as becoming available in the further progression of the project.

9 Bibliography

Kaelbling, L. L. (1996). Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, 4, 237-285.

Terzopoulos, S. M. (2020). *Image Segmentation Using Deep Learning: A Survey.* Von Arxiv.org: https://arxiv.org/abs/2001.05566

Zhu, Q. L. (2011). Image segmentation and major approaches. *2011 IEEE International Conference on Computer Science and Automation Engineering*, (S. 465-468).

10 List of figures

11 List of tables

12 List of Abbreviations

AI	Artificial Intelligence
BEV, BEVs	Battery Electric vehicle(s)
CNN	Convolutional Neural Network
DVFS	Dynamic Voltage and Frequency Scaling
DoA	Description of Action
EDL	Eclipse Distribution License
EPL	Eclipse Public License
FMU	Functional Mockup Interface Unit
FOTA	firmware update over the air
FPN	Feature Pyramid Networks
GDPR	General Data Protection Regulations
HATMA	Hierarchical Adaptive Time-triggered Multi-core Architecture
HICP	hierarchical interactive consistency protocol
HW	Hardware
IoT	Internet of Things
IOU	Intersection over Union
K8S	Kubernetes
MDP	Markov decision process
ML	Machine Learning
MPC	model predictive control
MQTT	Message Queuing Telemetry Transport
PULP	Parallel Ultra Low Power
R-CNN	Region-based Convolutional Neural Network
RL	Reinforcement Learning
RoT	Root Of Trust
RPU	Realtime Processing Unit (Versal)
SW	Software
TCP/IP	Transport Protocol over Internet Protcol
UAV	Unmanned Aerial Vehicle
UC	Use Case
WP	work package (in context of the FRACTAL project)