ICT-56-2020 - Next Generation Internet of Things

D7.8
Report on Open Call results

Document information	
Contract number	957197
Project website	www.vedliot.eu
Dissemination Level	PU
Nature	R
Contractual Deadline	31.10.2023
Author	Carola Haumann (UNIBI), Jens Hagemeyer (UNIBI)
Contributors	OC projects and All partners
Reviewers	Antonio Casimiro (FCUL), Micha vor dem Berge (Christmann)

The VEDLIoT Project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957197.

Changelog		
V 0.1	24.07.2023	Initial draft and ToC
V 0.2	27.08.2023	Added project interation section
V 0.3	07.10.2023	Version for internal review
V 0.4	30.10.2023	Incorporating feedback from reviewers
V 1.0	31.10.2023	Finalization

Table of contents

Executive summary

The goal of the Open Call in VEDLIoT was to involve additional use cases in the project utilizing the developed technologies, i.e., the VEDLIoT toolchain and the cognitive IoT hardware platform. After the evaluation phase from the 8th of May 2022 until the 10th of June 2022, 10 Open Call projects out of 30 submitted were selected to be funded within VEDLIoT. For selected projects, the topics range from smart agriculture, over pollen analysis, biosensors, and wearables until laser welding.

This report documents the Open Call (OC) execution, including the final technical report for the different open call projects. The final technical reports detail the use case architectures, developments, and benchmarking results. The active phase of the VEDLIoT Open Call projects started on the 1st of July 2022 (M21) and ran until the 30th of June 2023 (M33). The activity has been part of Task 7.8.

The key facts of the VEDLIoT open call are summarized below.

Call identifier: VEDLIoT – Open

Call title: Next-generation AIoT applications – VEDLIoT-Open

Publication date: 01/03/2022

Proposal Deadline: 08/05/2022 at 23:59h CEST

Number of Submitted/Selected projects: 30/10

Used budget for the call: 838.845,94 €

Reimbursement rate: 70 %

Total costs of all open call projects: 1.198.351,34 €

Indicative Duration of participation: 12 months

OC project site: https://vedliot.eu/use-cases/open-call/

The different open call projects have used VEDLIoT technology and provided direct feedback to the VEDLIoT developers, helping mature the developed technologies.

1 Introduction

This report documents the Open Call (OC) execution (from M21 to M33)). It includes the technical reports of the **10 selected Open Call projects**. As illustrated in Figure 1, from a global project perspective, the Open Call within VEDLIoT is located in WP7, as it brought additional use cases to the project.

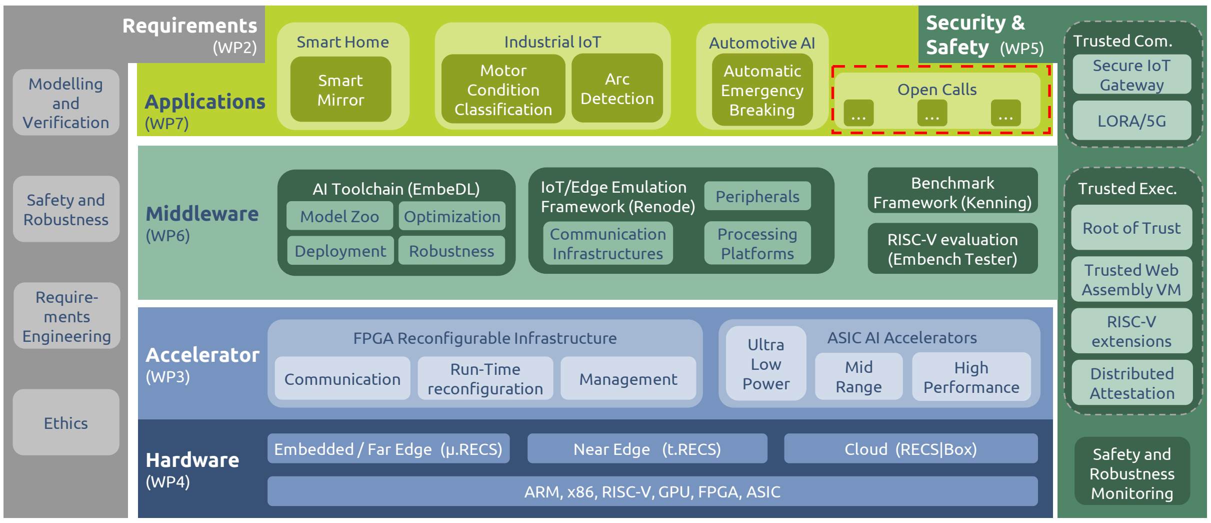


Figure 1: VEDLIoT project overview – this report is part of the activities in Task 7.8 within WP7

In the following, Chapter 2, after introducing the selected open call projects, gives an overview of the different key results for the individual open call projects, summarizing their final reports and their main results. Chapter 3 focuses on the interaction of the open call projects with the main VEDLIoT project, discussing which main assets have been used by the open call. In the Annex of this deliverable, the full final technical report of each project is available.

2 Open Call project results

The whole Open Call process was organized according to the cascading funding mechanisms, at which the Open Call preparation and publication from M16 to M18, including all relevant documents, is described in detail in *D7.6 Preparation documents for Open Call* [1].

VEDLIoT has cooperated with the Coordination and Support Action (CSA) Next Generation Internet of Things –(NGIoT) [2] already since the overall project started, e.g., by joining the "Open Calls Group". This cooperation included regular monthly meetings and reporting the status and outcome of the OC to the CSA. In a similar fashion, VEDLIoT is also active in the CSA EUCloudEdgeIoT [3].

In total, **30 OC** projects have been submitted with topics from **smart agriculture and cities**, over **pollen analysis**, **biosensors**, **neuromorphic processors** until **laser welding.** This includes **42 applying entities** (single, double and 3 partner consortia) from **15 countries.** The complete list of the 30 submitted proposals can be found in Annex I.

Out of the applications, **10 projects** (13 entities) **from 5 countries** were selected. The OC selection process was described in detail in *D 7.7 Evaluation and Selection Report for Open Call* [4]. Table 1 shows an overview of the selected projects, their respective IoT application domains, and the associated VEDLIoT Mentor. Each project had at least monthly meetings with the respective VEDLIoT mentor and the VEDLIoT coordinator to discuss the project's progress, current challenges, and documentation. The VEDLIoT mentor and coordinator have also been reviewing project deliverables and providing support with technical, organizational, and administrative advice.

Table 1 Selected VEDLIoT OC projects

Acronym	Title of project	Desciption	Type	Country	IoT Application domain	VEDLIoT Mentor
Honey.AI	Honey.AI 2.0 – The Evolved and optimized AI-related IoT solution for the honey industry.	AIoT Pollen analysis	Company	Spain	Agriculture	ANT
MushR	MushR: A Smart, Automated and Scalable Indoor Harvesting System for Gourmet Mushrooms	Automated Harvesting System for Mushrooms	Research	Germany	Agriculture	CHR
Power Edge RL	Control of electric power systems via edge computing-based reinforcement learning	Edge-based reinforcement learning for power electronics	Research	Germany	Industrial	UOS
DUNE RCO	Deep learning for multi-technology fusion in industrial indoor asset localization and tracking	Industrial indoor localization and tracking	Research	Spain	Industrial	RISE
BEAM_IDL	Multiple laser BEAM-shaping monitoring and IDentification boosted by deep-Learning algorithms	DL-based laser welding for Aluminium	Company + Research	Spain	Industrial	UGOT
AI_RIDE	Artificial Intelligence - driven RIding Distributed Eye	AI framework for driving courses on motorbikes	Research + Company	Italy	Automotive	VEONEER
FLEDGED	Feasibility of Low-energy Embedded Deep-Learning-Models Geared for Edge Devices	Wearables architecture for heathcare	Company	Ireland	Medical	EMBEDL
AccBD	Accelerated Biomarker Candidate Discovery	Accelerated Biomarker Candidate Discovery	Research	Germany	Medical	UNIBI
Edge4iwelli	Edge computing to support the iwelli ecosystem of services for smart home care and independent living	Edge-based healthcare for smart home	Company	Greece	Smart home	UNINE
FLAIR	Federated Learning Extension for Very Efficient Deep Learning in IoT	Federated Learning Extension for VEDLIoT	Research	Spain	General	CHALMERS

The final technical deliverable is a collection of all technical deliverables produced during the runtime of the open call project, together with some overarching aspects related to integration, exploitation, and sustainability aspects. While the complete final technical deliverable for each project is available in Annex I, a brief overview of the ten projects is provided in the following.

2.1 Honey.AI 2.0

The Evolved and optimized AI-related IoT solution for the honey industry.

Pollen analysis in the honey industry determines the floral source and verifies the authenticity of this precious food. Traditional methods involve a labor-intensive manual examination of honey on a glass slide under a microscope for 1-2 hours. A skilled laboratory technician identifies and counts each pollen grain species. This method is not only costly and time-consuming but also prone to human error, and results aren't immediate. Nevertheless, such evaluations are vital for the honey industry because they contribute to quality checks, detect fraud, and adhere to labeling regulations.

Honey.AI is an equipment designed to provide an on-the-spot, time-efficient IoT solution for industry stakeholders to gauge their honey's quality using robotics, computer vision, and deep learning. This digital microscope aims to simplify the process for those without specialized training, potentially completing multiple honey analyses in under an hour, right at the location. Sonicat's objective is to introduce AI and digital tools within the honey industry, which has been relatively slow in adopting advanced technological innovations.

Honey.AI features hardware that directs three precision motors and a digital camera. This pairs with a C++ application responsible for capturing and uploading thousands of images to cloud AWS GPUs. These images then undergo processing via TensorFlow-trained neural networks. However, initial particle identification and sorting, covering categories like pollen, yeasts, honeydew, starch, bubbles, and contaminants, take place within the microscope itself, ensuring only relevant images are sent to AWS.

The previous desktop application has transitioned from a user's external computer Windows version. Honey.AI has adeptly evaluated various edge devices and gauged their performance. Throughout this venture, Sonicat enhanced Honey.AI with additional features, expanded the datasets for identifying pollen and other particles, relocated an upgraded particle detection and classification system to the edge, enhanced prediction speed and accuracy, overhauled the APP, API, and AWS GPUs processing architecture, developed a fresh prototype for project testing with Jetson Nano and Xavier, and revised their business and monetization strategy.

For this initiative, the team employed tools from VEDLIoT technologies, specifically the test beds tailored to emulate edge device capabilities, refining models, and benchmarking with Kenning and EmbeDL. Both tools, designed specifically for AI-driven applications in IoT devices, have proven invaluable. Especially Kenning, an open-source tool, was a new discovery for the Sonicat team.

2.2 MushR

A Smart, Automated and Scalable Indoor Harvesting System for Gourmet Mushrooms

The gourmet mushroom industry is witnessing a shift towards digitization, positioning it as a game-changer for mushroom farming efficiency, productivity, and quality. This project set out to harness the power of digital technologies and smart automation, seeking to refine every phase of the cultivation process. Using controlled environment chambers furnished with intelligent sensors, factors like temperature and humidity were meticulously regulated, creating the optimal mushroom growth conditions. An integrated imaging system, paired with a Mask R-CNN model, facilitated continuous observation and precise determination of mushroom ripeness. Introducing a semi-automated harvesting mechanism further refined the cultivation workflow, ensuring meticulous extraction of ripe mushrooms.

With the aid of the VEDLIoT near-edge computing features, the image recognition mechanism oversees the mushroom growth trajectory, directing the robotic arm throughout the harvest phase. The system's modularity and expandability are apt for commercial operations, with VEDLIoT edge gadgets positioned in cultivation spaces, adjustable to the mushroom farming configurations of the establishment. Consequently, MushR significantly curtails the dependence on manual workforce in gourmet mushroom cultivation and harvesting, setting the stage for expansive automation, and elevating both mushroom yield and quality.

The outcomes of the project underscore the transformative capability of digital and automated solutions in reshaping the mushroom sector, leading to amplified yields, diminished labor reliance, superior product grade, and enriched analytical abilities. As global concerns about food sufficiency and ecological sustainability intensify, the adoption of digital tools in farming becomes ever more crucial. The MushR initiative stands as a beacon, illustrating the immense promise of digital transformation in refining mushroom farming methods and hinting at the future horizons of digital farming.

2.3 Power Edge RL

Control of electric power systems via edge computing-based reinforcement learning

Harnessing data for the intelligent control of power conversion systems, such as electric drives, presents an avenue for automating and accelerating the developmental trajectories of the future. Forefront methodologies in this domain are increasingly reliant on reinforcement learning (RL). This approach mandates the concurrent fulfillment of two distinct needs: firstly, the real-time execution of control decision-making (inference), and secondly, the ongoing refinement of this decision-making mechanism by interpreting continuous data streams acquired from interactions with the control plant. While the former demands real-time action, the latter can be efficiently delegated to edge computing hardware, thus enhancing the speed of the learning trajectory.

In the course of the VEDLIoT OC initiative, a cutting-edge computing toolchain was conceived and actualized, specifically tailored for the distributed inference and learning processes intrinsic to an RL-oriented drive control system. This innovative system underwent rigorous testing on real-world platforms, and its accomplishments are now being chronicled in scholarly publications. Adding another feather to its cap, the said toolchain was expanded to encompass a range of edge computing accelerators, spotlighting GPUs and FPGAs. These additions underwent empirical testing and were subsequently juxtaposed against traditional CPU-centric frameworks.

Conclusively, the Power Edge RL project has not only illustrated the viability of incorporating a data-centric control procedure within an IoT framework but has also accentuated the myriad advantages it brings to the table.

2.4 DUNE RCO

Deep learning for multi-technology fusion in industrial indoor asset localization and tracking

DUNE stands at the forefront of innovative asset tracking by integrating distributed computing across far-edge, edge, and cloud domains. This approach systematically deconstructs estimation processes to meet the stringent demands of real-time asset tracking applications.

Central to DUNE's advancements is its Real-Time Localization System (RTLS). It shines in pinpointing the direction of emitted Bluetooth Low Energy (BLE) signals through a sophisticated Deep Learning (DL) model, outclassing traditional signal processing techniques. A significant edge comes from the adept deployment of this DL model on the VEDLIoT hardware at the far-edge level. This approach sidesteps the intricate calculations of older methods and introduces remarkable scalability.

By consolidating estimates from these far-edge nodes, DUNE promises remarkable precision, achieving near-perfect position estimations at the combined edge-cloud interface. DUNE's horizon isn't limited to current capabilities. It's poised to embrace other wireless technologies like WiFi, RFID, and UWB. By harnessing DL, DUNE aims to design tailored positioning solutions that can be adjusted based on the chosen technology, setting the stage for a more versatile tracking framework.

The practical application of DUNE RTLS is evident in its real-world deployments. While pre-production environments cater to rigorous algorithm development and testing, resulting in the public release of DL models and associated datasets, the production setting offers a tangible demonstration of its prowess. Notably, DUNE is actively operational at the Universitat Oberta de Catalunya (UOC)'s interdisciplinary R&I hub, effectively monitoring a sprawling area and consistently tracking the movement and localization of a considerable daily populace.

In summary, DUNE RCO showcases the transformative potential of blending deep learning with multi-technology fusion, setting a new benchmark in industrial indoor asset localization and tracking.

2.5 BEAM_IDL

Multiple laser BEAM-shaping monitoring and IDentification boosted by deep-Learning algorithms

The Electric Vehicle (EV) manufacturing industry, particularly in areas like battery closures and chassis attachments, has seen a surge in the use of Aluminium (Al). To overcome intrinsic welding challenges associated with Al, dynamic laser beam methodologies are being adopted to enhance laser-material interactions. However, these innovative welding techniques necessitate cutting-edge in-process monitoring strategies that can efficiently recognize patterns and process events.

Stepping up to the challenge, BEAM_IDL introduces the integration of refined Deep Learning (DL) algorithms tailored for visible and infrared (IR) laser welding image recognition. This forward-thinking approach aims to merge feature extraction with process parameters, laying the groundwork for advanced process optimization in the future.

The technical report elaborates on the experimental tests and the implementation done during the project's duration, offering insights into the developed algorithms and their effectiveness in real-world welding situations. The foundational datasets, initially generated in a demonstrator setup, enriched the project by guiding the development and refinement of the DL algorithms, enhancing their overall performance.

Transitioning from theory to practice, the system architecture was realized into a tangible machine integration. Employing a robot cell at LORTEK and industrial optics curated by EXOM Engineering, the proposed technological solutions underwent rigorous testing on a demonstrative part, ensuring their practical applicability.

In a broader context, BEAM_IDL's offerings promise to enrich the welding portfolio, adding value to products from EXOM and services by LORTEK. As the industry faces stiff competition from cost-effective manufacturing solutions outside the European Union, integrating specialized AI-driven monitoring and control becomes pivotal to maintaining a competitive edge. In essence, BEAM_IDL underscores the transformative potential of marrying DL with laser welding, aiming to set new industry standards.

2.6 AI-RIDE

Artificial Intelligence - driven RIding Distributed Eye

AI-RIDE brings forth a novel proposition: leveraging a high-speed, embedded Artificial Intelligence framework to revolutionize motorcycle rider training, with a focus on Practical Driving Courses (PDC) and Driving Licence Exam (DLE) verification tools. This initiative aims to significantly outpace the existing instruments and methodologies currently prevalent in the PDC and DLE landscape, marking a paradigm shift in driving education techniques.

The success of AI-RIDE is intrinsically linked with the VEDLIoT hardware platform and middleware. This technology, with its disaggregated IoT, sensor-based network architecture, and distributed algorithm approach, empowers the AI-RIDE platform to offer a suite of groundbreaking features catered to various stakeholders in the driving education ecosystem, including learners, instructors, and examiners. VEDLIoT is recognized as the optimal choice for this venture, ensuring the necessary online computational power and fulfilling the stringent low-latency requirements crucial for real-time assistance.

Central to AI-RIDE's functionality is its advanced video and image processing capability. This allows seamless data fusion from multiple input sources, such as cameras and wearable devices, paving the way for enhanced visual target augmentation during driving test sessions. In essence, AI-RIDE underscores the potential of integrating AI into the realm of motorcycle training, presenting a vision of a smarter, more efficient learning experience.

2.7 FLEDGED

Feasibility of Low-energy Embedded Deep-Learning-Models Geared for Edge Devices

Managing chronic diseases often necessitates continuous health monitoring. Around one-fourth of Europeans are affected by such conditions, which invariably influence their daily activities. Among these, cardiovascular diseases, neurological disorders, and mental health challenges are prevalent, with epilepsy being a notable concern for many, often with limited treatment options.

Given this backdrop, there's a clear demand for dependable monitoring tools for individuals with chronic conditions like epilepsy. The majority of existing health monitoring tools tend to rely on centralized cloud infrastructures, which might not always be ideal, especially when real-time alerts for significant events, such as seizures or cardiac incidents, are essential. Traditional deep learning methods, while effective, can be resource-intensive in terms of computation and energy, a challenge in local, on-device settings. It's also anticipated that these computational demands will grow.

The FLEDGED initiative, led by the iBreve team, introduced a wearable technology focused on monitoring respiratory metrics and activity patterns. Paired with an app, it provides users with access to their health data and potential intervention strategies.

Beyond this, FLEDGED's goal was to investigate the potential of incorporating AIoT into healthcare, leveraging the capabilities of the VEDLIoT frameworks. The primary objective was to determine the possibilities for the next generation of wearables designed for ongoing health monitoring. Emphasis was placed on enhancing the efficiency of deep learning, reducing energy consumption, and strengthening security measures.

With the assistance of VEDLIoT, FLEDGED was able to explore and evaluate tools for a more adaptable and scalable platform suited for wearables in chronic health monitoring, underlining the feasibility of using edge-focused AI in the healthcare domain.

2.8 AccBD

Accelerated Biomarker Candidate Discovery

Modern medical research often involves analyzing data sets with a vast number of features but limited sample sizes. This presents unique challenges like the curse of dimensionality, and anomalies like skewed data distributions and outliers. Such challenges necessitate specialized algorithms that are typically computationally intensive.

The AccBD project focused on enhancing a feature selection workflow. It implemented an ensemble of embedded feature selection techniques optimized for the VEDLIoT hardware platform. The results were notable: an acceleration factor of 13.9X and an energy saving of 72%. However, integrating GPU for model training encountered a memory leak issue, which needs further resolution to fully leverage GPU acceleration.

A significant advancement in AccBD was the optimized implementation of the Yeo-Johnson transformation, essential for data preprocessing. The C-library version of this transformation boasted an 88.27% reduction in runtime compared to the Python scikit-learn library, along with an energy reduction of 89.5%. When this transformation was adapted to FPGA, a single module showcased a runtime decrease of 37.7% and an energy drop of 91.9% compared to the C-library version. Using multiple modules (five, in this instance) led to a runtime reduction of 81.4% and an energy saving of 95.7% against the C-library rendition.

In conclusion, AccBD demonstrated that, by harnessing optimized parallelization and leveraging alternative hardware, significant time and energy efficiencies can be achieved in the realm of medical research.

2.9 Edge4iwelli

Edge computing to support the iwelli ecosystem of services for smart home care and independent living

The increasing need for at-home healthcare is evident, and iwelli's IoT care package addresses this demand by providing a holistic suite of tools and applications to support home healthcare services. However, relying solely on cloud services for remote health monitoring brings forward issues of security, latency, and environmental sustainability. Enter the Edge4iwelli project, which seamlessly incorporates edge computing into the iwelli ecosystem to surmount these challenges.

Edge4iwelli's prime objective is to shift AI processing and data analysis closer to the source, thereby diminishing the need for extensive cloud services. By integrating advanced AI modules with edge computing into the iwelli IoT package, it crafts a secure and efficient care platform. This innovation allows for practical demonstrations of edge computing, optimization techniques, and machine learning tailored for home-based healthcare applications.

A notable innovation within the project is the Edge4iwelli smart health mirror. This tool, combined with the enhanced IoT care platform, forms the cornerstone of the project's aim: making home healthcare more accessible, efficient, and user-friendly. With tools like the smart health mirror, individuals can proactively manage their health, benefiting from the convenience and security of their home setting. In essence, Edge4iwelli is paving the way for a revolution in reducing healthcare expenses by promoting effective and efficient care right at home.

2.10 FLAIR

Federated Learning Extension for Very Efficient Deep Learning in IoT

In today's digital age, with an estimated 7 billion IoT devices and 3 billion smartphones in circulation, data-driven Machine Learning (ML) solutions are more relevant than ever.

Historically, these solutions have operated by funneling data to a centralized server where ML models are trained and tested. But a pressing concern arises: many IoT devices handle sensitive personal data, sparking public outcry over privacy issues.

Such concerns have prompted the introduction of stringent privacy and data-sharing regulations, like the European Commission's GDPR and the US Consumer Privacy Bill of Rights. Moreover, the vast range of computational and communication capabilities among IoT devices, especially those with constrained resources, presents further challenges for holistic AI solutions.

Enter the FLAIR project, championing the concept of Federated Learning (FL). This approach allows multiple end nodes, such as IoT devices, to collaboratively train an ML model without compromising their data privacy. With FL, the training data remains localized to the device, eliminating the need to centralize data storage on cloud platforms.

In numerous AI applications, such as voice-activated assistants, model training typically takes place on a centralized server, after which the trained models are sent out to end-user devices. However, FLAIR revolutionizes this by facilitating local training on individual client devices. Each device produces its own model, which then gets aggregated on a central server to create a global model. This ensures the continuous updating of the global model while honoring each device's local data privacy.

FLAIR, in essence, offers a forward-thinking approach to ML in IoT, prioritizing both efficiency and privacy.

3 VEDLIoT project interaction

The subsequent sections delve into the diverse applications of VEDLIoT technology across various open call projects. While these projects utilized the multifaceted offerings of VEDLIoT to enhance their operational efficiencies, they also served as crucial testing grounds for the technology itself. As these projects integrated VEDLIoT tools in real-world scenarios, they unearthed challenges and opportunities that provided pivotal feedback to the VEDLIoT developers. This feedback loop not only refined the VEDLIoT offerings but also facilitated its evolution, ensuring that it remains at the cutting edge of IoT advancements. By using various pieces from the project, the open call projects provided direct feedback to the VEDLIoT developers, helping mature the developed technologies.

3.1 VEDLIoT Hardware Platform

- **Honey.AI:** Initially, the integration of the u.RECS platform within a robot was contemplated to enhance parallel processing and reduce lead time. The idea was to merge a Jetson device with either an FPGA or a TPU accelerator. However, after a thorough assessment of the costs and technical feasibility, the solution was found to be cost-prohibitive. The primary requirement of Honey.AI is to process pictures with utmost clarity for accurate recognition. To meet this need, a solution based on a single Jetson device for classification was formulated using the u.RECS platform for prototyping the solution.
- **PowerEdge RL:** Modifications were made to the previously developed edge learning toolchain to make it compatible with the t.RECS equivalent Nvidia Jetson AGX Orin hardware. Preliminary assessments highlighted that the in-built SoC GPU was inadequate for RL or gradient descent acceleration. Consequently, a combination of the t.RECS platform and a Versal FPGA platform was employed.
- **MushR:** VEDLIoT's u.RECS evaluation board facilitated early-stage prototyping, enabling camera setup testing. The flexibility of MushR promotes industrial application, with VEDLIoT edge devices strategically positioned at mushroom cultivation facilities, ensuring scalability. Additionally, Christmann's cloud-hosted GPU training environment was utilized for machine learning model training, ushering in advancements in mushroom cultivation and harvesting.
- **DUNE RCO:** VEDLIoT's distributed computing capacities across various levels (far-edge, edge, and cloud) support the real-time application requirements of DUNE. Deep Learning (DL) plays a pivotal role in various DUNE positioning phases.
  - In the far-edge, u.RECS processes DL-based estimations of BLE signal angles and WiFi and BLE signal ranges, ensuring flexibility, scalability, and computational prowess for varied IoT contexts.
  - At the edge, DL integrates position estimates from varied technologies. VEDLIoT hardware ensures efficient processing in large deployments. Here, the t.RECS becomes the focal hardware, processing wireless signals and combining estimates from varied technologies.
- **AI_RIDE:** VEDLIoT's hardware platform aids in the development of systems and testbeds, inclusive of the final demonstration. The platform manages data inflow from various sensors and processes it for both training and real-time applications. Real-time functionality necessitates the VEDLIoT platform's presence at the AUG Track premises. The essential hardware components comprise:
  - A u.RECS for real-time feedback in the eye tracking use case, possibly placed within a motorbike.
  - A t.RECS for the path tracking use case, capable of handling multiple HD video streams.
- **AccBD:** AccBD's foundation lies in the RECS|Box, a modular microserver system that facilitates the amalgamation of various compute modules. The established cluster on

the VEDLIoT testbed spans three nodes, all operating on x86_64 architecture, complemented by a ZYNQ-7045 FPGA, completing the hardware test setup.

3.2 VEDLIoT Accelerators

- **Power-Edge-RL:** Power-Edge-RL employed VEDLIoT's STANN (Synthesis Templates for Artificial Neural Networks) software to craft an FPGA-based accelerator tailored for the DQN training regimen. This adaptation substantially augmented the speed of individual DQN training steps. The efficacy of this FPGA setup was cross-checked and authenticated via a Python-simulated motor control environment.
- **FLEDGED:** To gauge the practicality of leveraging hardware for speeding up deep learning implementations at the network's edge, an exhaustive review of suitable hardware fitting our stipulated performance and size metrics was undertaken. The project zeroed in on Silicon Witchery's 'S1 Bluetooth-FPGA System On Module' (anchored by Nordic's nrf52811) and the robust nRF52840 System on Chip by Nordic Semiconductor. The most notable find was Silicon Witchery's 'S1 Bluetooth-FPGA System On Module'. This module blends the capabilities of Nordic's nRF52811 SoC with the energy-efficient FPGA - Lattice iCE40 Ultra Plus.

3.3 Renode and Kenning

- **Honey.AI:** Honey.AI utilized Kenning's components: Dataset, ModelWrapper, ModelCompiler, and ModelRuntime to identify the most suitable EfficientNet model. This deep dive into the various modules facilitated an informed decision-making process, taking into account different trade-offs before finalizing the model.
- **BEAM_IDL:** BEAM_IDL recognized the potential fit of the Kenning Framework for their needs, contemplating its deployment for automating a deep-learning workflow. Additionally, they have expressed an interest in potentially integrating the EmbeDL optimization toolkit into their project.
- **FLEDGED:** Using the Renode Simulation Tool, FLEDGED simulated the Nordic Semiconductor nRF52840 System on Chip, allowing them to assess the wearable device with production binaries. The BLE-capable Renode radio model and the Zephyr biosignal monitor sample facilitated BLE traffic analysis. Renode's framework, endorsed by VEDLIoT, proved pivotal, especially for testing aspects like alarm latency and BLE communication distance. Antmicro further introduced dedicated support for the Nordic Semiconductor nRF52840 SoC in Renode.

3.4 Requirements Framework

- **BEAM_IDL:** By employing the VEDLIoT Requirements Framework, the BEAM_IDL project identified potential safety constraints early in the development phase. This proactive approach allowed the team to navigate potential pitfalls, ensuring the application was built with utmost safety standards in mind. Moreover, the framework provided a systematic approach, offering insights that directly influenced the structural and functional decisions of the system's architecture. The valuable feedback loop established through the framework enabled the team to adjust their strategies in real-time, ensuring alignment with the end-user's safety needs.
- **AI_RIDE:** The VEDLIoT Requirements Framework played a pivotal role in the AI_RIDE project. As the project navigated the complex landscape of integrating AI technologies into Driving Licence Exam (DLE) sessions, the framework was invaluable. It provided clarity on how AI technologies could be seamlessly and legally integrated, ensuring that the technology complemented, rather than disrupted, traditional examination processes. Through systematic analysis and assessment, the framework allowed the team to address potential legislative bottlenecks and

challenges proactively, paving the way for a more efficient and compliant system architecture.

- **Edge4iwelli:** In the realm of personal health and smart home devices, the ethical management of personal data is paramount. The Edge4iwelli project's smart mirror concept presented unique challenges in this respect. Using the VEDLIoT Requirements Framework, the project team was equipped with tools to critically assess and design a system architecture that upheld the highest standards of data privacy and ethics. The framework guided the team in identifying potential ethical concerns, ensuring that users' personal data was treated with the utmost respect and security. This proactive approach not only safeguarded user data but also strengthened trust in the smart mirror technology.

3.5 EMBEDL optimizer

- **Honey.AI:** By leveraging the detailed insights garnered from the Kenning analysis, Sonicat ascertained that the efficientnetB-1 model was best suited for their specific operational requirements. After the model was appropriately trained, it underwent further optimization. EmbeDL's Taylor pruning technique was utilized to enhance its efficiency, ensuring that the model was both lean and effective for their image processing tasks.
- **AI_RIDE:** Utilizing the sophisticated capabilities of the VEDLIoT toolchain, AI.Ride delves deep into the realm of distributed AI. The TensorFlow Frameworks, when paired with aptly chosen compilers and Runtime APIs, seamlessly facilitate the distribution of computational tasks across diverse hardware platforms, ensuring that each piece of hardware is optimally leveraged. This integrated approach not only harnesses the full potential of each component but also paves the way for a more efficient and scalable AI processing environment.
- **MushR:** In the domain of machine learning model development, EmbeDL was an invaluable asset to the MushR project. Its advanced methodologies facilitated the automated generation of raw image datasets, streamlining the initial stages of model development. Moreover, EmbeDL's tools played a pivotal role in devising efficient annotation workflows, ensuring that the machine learning model was trained with accurate and relevant data. This synergy of tools and methodologies elevated the overall quality and accuracy of MushR's AI-driven solutions.

3.6 Security: TEEs

- **Edge4iwelli:** Within the Edge4iwelli project, ensuring user privacy was paramount, especially given the sensitivity of data often processed in wellness and healthcare scenarios. To achieve this, they incorporated VEDLIoT's TEEs. These environments provide an isolated space within a device, ensuring that sensitive data is processed securely, shielded from potential threats or unauthorized access. This robust approach to data protection gives users confidence that their personal information remains confidential and secure, upholding the promise of 'Privacy by Design'.
- **HoneyAI:** For HoneyAI, the stakes were two-fold: the protection of proprietary machine learning models and safeguarding user data. Models, once developed, can become valuable intellectual property, and their protection against potential theft or tampering is crucial. By using VEDLIoT's TEEs, HoneyAI ensured that their models ran in a secure enclave, impervious to external intrusions. This not only protected the models but also the data they processed. Given that HoneyAI deals with imagery data, which can often be sensitive, the use of TEEs ensured that data confidentiality and integrity were maintained throughout the processing pipeline. This dual-layer of security fortifies both the product's business value and its commitment to user data protection.

3.7 VEDLIoT applications

- **Edge4iwelli:** Building upon the foundation laid by VEDLIoT's Smart Mirror application, Edge4iwelli implemented the shared datasets and user interface (UI) in their own smart mirror project. This facilitated a more efficient development process, ensuring that the new project capitalized on tried-and-tested components and insights from the VEDLIoT application.
- **AI_RIDE:** AI_RIDE looked to VEDLIoT's experience in the Automotive application realm, especially in terms of 5G architecture. By drawing from this prior knowledge, AI_RIDE was able to seamlessly integrate 5G technologies into their project, taking advantage of established best practices and insights to enhance their system's reliability and performance.

4 Summary

The Open Call in VEDLIoT aimed to incorporate more use cases into the project using its developed technologies such as the VEDLIoT toolchain and the cognitive IoT hardware platform. Out of 30 submissions between May 8th and June 10th, 2022, 10 were chosen to receive funding within VEDLIoT. The ten selected VEDLIoT Open Call projects were executed as planned (from M21 to M33). A set of documents was prepared to manage the OC reporting process. All reports were reviewed by the respective VEDLIoT mentors. This report gives and overview about the project execution, including the detailed technical reporting documents attached in Annex I. The participating projects have utilized VEDLIoT's innovative technologies and offered invaluable feedback to VEDLIoT's developers, which has been instrumental in refining the technologies further.

References

[1] VEDLIoT Project, „VEDLIoT Deliverable D7.6 - Open call preparation and publication,“ 2022.

[2] EU-IoT, „Next Generation Internet of Things,“ EU-IoT Consortium, 2023. [Online]. Available: https://www.ngiot.eu.

[3] EUCloudEdgeIoT, „EUCloudEdgeIoT,“ EUCloudEdgeIoTconsortium, [Online]. Available: https://eucloudedgeiot.eu/.

[4] VEDLIoT Project, „VEDLIoT Deliverable D7.7 - Evaluation and selection,“ 2022.

Annex I

Within the following sections, there is a comprehensive collection of the final technical reports from each individual project. These reports thoroughly detail the technical facets of each endeavor. To facilitate navigation through this extensive compilation, use the Table of Contents at the beginning of the document, along with embedded bookmarks.

VEDLIoT

Very Efficient Deep Learning in IoT

ICT-56-2020 - Next Generation Internet of Things

Final technical report OC_HoneyAI VEDLIoT Open

Document information	
Project website	https://www.vedliot.eu
Dissemination Level	PUBLIC
Nature	R
Contractual Deadline	30.06.2023
Author	Microfy Systems
Contributors	N/A

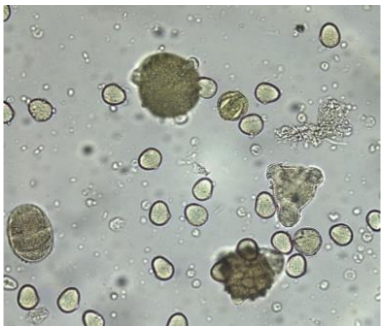
The VEDLIoT Project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957197.

Changelog		
Version	**Date**	**Type**
V 0.1	10/07/2023	Initial draft
V0.2	01/08/2023	Final Draft
…	…	…
V 1.0	…	Finalisation

Table of contents

1 Executive summary

Project Idea



Pollen analysis in honey industry is used to determine the floral source and authenticity of this highly valued food. It's a very manual test which involves studying a honey sample in a glass slide with a microscope for 1-2 hours, so all the existent pollen grains species are identified and counted by a widely trained laboratory technician. Manual counting is expensive, time-consuming, involves human error, and the results are obtained deferred. However, these analyses are essential and mandatory in the honey industry, since are used for quality assessment, fraud detection, and labelling legal directives.

Honey.AI is a novel equipment that allows on-site real-time cost-effective IoT device to honey industry stakeholders to assess their honey's quality with robotics, computer vision and deep learning tools.

We are introducing an AI-powered automated digital microscope, that is very easy-to-use for a non-trained user, to be used as a one-stop-shop for the honey industry. It is possible to carry out many different analysis in honey in less than 1 hour, and on-site. Our mission is to democratize AI and digital solution within honey industry, which has not introduce any disruptive technology to date.

Main outcomes & Achievements

The HoneyAI works with a PCB that controls 3 high-precision motors and a digital camera, coupled with an executable C++ application that is in charge of acquisition and uploading the thousands of pictures obtained per pollen test (from 1,000, to even 20,000) to the cloud AWS GPUs, where they are then processed with our TensorFlow trained neural networks.

However, preliminary particle detection and classification of families/types (pollen, yeasts, honeydew, starch, bubbles and dirt) is done on the edge, in the microscope itself, to avoid sending empty images to AWS.

The old desktop App has been migrated from the Windows version installed in the external computer of the user and we have been capable to analyze different edge devices and the performance obtain with each one of them.

During this project, we have: 1) incorporated two new functionalities to HoneyAI, 2) increased the datasets for pollen recognition and other particles, 3) migrated to the edge an improved particle detector and classifyier, 4) improved inference time and accuracy of predictions, 5) refactor the architecture of APP, API, and AWS GPUs processing, 6) designed and build a new prototype to be tested in the project with Jetson Nano and Xavier, and 6) updated the exploitation and commercialization strategy.

To carry out this work, our team have used 3 different tools from VEDLIoT technologies, specifically the test beds specifically designed to mimic edge devices performances, and improve models and benchmarking and pipelines with Kenning and EmbeDL. Both solutions specifically conceived for AI-powered applications in IoT devices have been very useful for our development, specifically Kenning which is open-source and our team did not know it.

2 Introduction

Honey is much more than a sweetener; it is a high-value functional food. Honey consists of various sugars, organic acids, vitamins, minerals, enzymes, and particles (as pollen) derived from honey collection. Among other benefits for humankinds, like its greatly valued flavour, honey has been consumed since ancient times due **to the proven health benefits it offers**, which make its more valued than other sweeteners. Thus, it is not strange that honey is of such great importance to human beings, **reaching 1.86 million tonnes of production worldwide in 2022, and a global market valued at €8,903 million,** making this product of great economic importance nowadays. In addition, **honey industry is an activity of vital importance to the sustainability of Europe, and plays an important role in agriculture**, since apiculture and bees are **responsible for more than €30 billion a year in crops, being responsible for more than 30% of world's food by providing pollination in about 1000 flowering species, therefore impacting on the whole food supply chain.**

Bees visit several different plants to collect nectar and pollen, and return to the hive, where the honey is produced. Thus, honey does not always have the same taste, colour, nor composition: **according to its predominant floral origin, honey is classified and labelled into *general multifloral*, or *specific monofloral***. Monofloral honeys are more demanded by consumers for its appreciated health benefits and greater taste, thus presenting **higher market price compared to standard multifloral honey**.

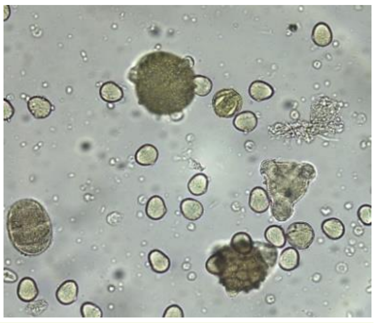
Although there are many different types of analysis to analyse very diverse honey quality parameters (colour, pH, sugars content, moisture, conductivity, enzyme content, etc.), the **predominant botanical origin is assessed with a pollen analysis, which confirms honey provenance and provide a quantitative measure of such floral origin.** Even monofloral honey contains nectar and pollen from more than one floral sources, in fact usually tenths. The analysis of the pollen content within the honey industry (called *Melissopalynology,* and done with light microscopy) constitutes an essential analytical test for all the stakeholders in the honey supply chain, and is carried out for 3 different purposes:

❶ **Labelling** ▸ In most of the EU countries, there is a regulated **minimum percentage of pollen grains of the specific floral source legally required for the classification and labelling of monofloral honeys** (Eucalyptus, Erica, Citrus, Lavandula, Thymus, etc.), thus the pollen concentration must be assessed **to meet honey marketing constraints**.

❷ **Processing and trading** ▸ Honey packers and traders purchase and process different batches (drums) of honey from different beekeepers to meet with market demand. **Floral origin and purity are used as a measure of product valorisation** when beekeepers and honey packers are negotiating the selling price of each honey batch: **the higher the pollen concentration is, the more valuable is considered the drum.**
On the other hand, **for retail, standardization is very important** from the market consumer point of view. Honey packers aim to keep their honey brands as uniform as possible in colour, flavour, and composition, so the consumer gets the same type of honey every time. To this end, honey producers and honey packers **routinely blend two or more types of different honey batches to keep this uniformity**. During the mixing process, beekeepers and honey packers must consider the resulting concentration of the HydroxyMetilFurfural compound (indicator of honey overheating which is legally regulated), the glucose-fructose-water ratio (quality indicator related with crystallization), and the **final pollen concentration of the mixture**. Considering the botanic source during decision-making process about honey mixtures help packers to better estimate the final honey's quality and price.

❸ **Fraud screening** ▸ Honey is one of the most adulterated food in EU. The EU coordinated action "From the hives" confirmed the initial assumption that a significant part of honey imported into the EU is suspicious of not complying with the provisions of the "Honey Directive" (46% based on 320 samples). it is clearly demonstrated that a significant percentage of the honey that all of us are consuming is often subject to **1) geographic or botanical mislabelling** (selling one honey kind as another type, more expensive) and **2) adulteration** (mainly by diluting honey with cheaper syrups, or sugar cane, that mimic honey's own sugar composition).

This is especially prevalent with monofloral honeys, as the most expensive ones. Apart from other complex analytical procedures for honey quality/authentication assessment, **a widely employed method for the authentication of honeys is melissopalynology**. Not only can we tell what botanical specie the honey is from, **melissopalynology can also be used to see whether honey has been diluted with grain syrups** – a common counterfeiting tactic, or even the **geographical origin, by means of identification of <u>endemic pollen species found in the sample</u>**.



Melissopalynology ▸ *As exposed, it is the oldest and the most essential tool globally used to determine the floral source of honey, and even the geographical origin, upon which the bees foraged to produce honey (because each flower species has its unique pollen grain). The official manual technique consists of diluting a sample of raw honey partly warm water and alcohol, centrifuge it, and the resulting residue examined under a light microscope at 400X. A trained specialist (often practicing during years) manually inspects the whole glass slide area to detect, identify, and count all the pollen grains within the concentrated residue. There are 3 different types of "particles" that can be found: i) dirt and honeydew elements (bee's parts, spores, fungal hyphae, parts of plants, sugar crystals, etc.), ii) countable pollen grains (those that come from nectariferous plants) and iii) non-countable pollen grains (those that come from non-nectariferous plants). Dirt, honeydew elements and non-countable pollen grains must be recognized, but dismissed from total counting, while the countable pollen grains are classified and considered for concentration counting. The expert must count, at least, 500 countable pollen grains in the sample to finish the analysis. The resulting concentration of the specific botanical floral source is obtained as: number of specific pollen grains of the predominant floral source, divided into the total number of countable pollen grains found.*

As can be appreciated, the pollinic analysis is an essential and useful tool within the honey industry. However, conventional pollen identification/counting by visual inspection with light microscopy proves complex, time-consuming, exhaustive, and cost-intensive:

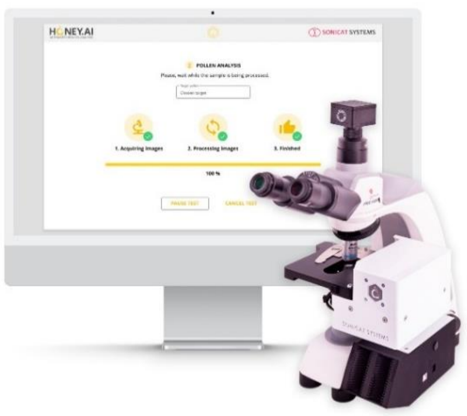
✕ It leads to long experimental times and cost-intensive workflow, due to the very low processing speed. Considering both, the sample preparation and the pollen grains identification and counting with the manual microscope, **the pollen analysis requires intensive human labour (around 1 – 1.5 hours per sample)**.

✕ The honey sample analysed usually presents many different pollen types, as well as small pieces of plants, parts of bee's wings, etc… In consequence, **the melissopalynologist should have wide expertise to conduct this complex manual test**. <u>Only the largest players in the honey market can conduct these tests at their premises</u>, because there is a **critical lack of trained staff, so they are difficult to find/hire.**

✕ Thus, most of beekeepers and honey traders are often forced to **externalize this service to the few specialized laboratories that have melissopalynology experts**, mainly Eurofins, Quality Services International, and Intertek. **The price paid per sample for the analysis is between 40 and 80 €** (depending on the laboratory and country). **A medium honey packer can typically perform 200 – 300 tests per year, while a large honey cooperative often requires more than 1,000 analysis. Apart from being a capital-intensive subcontracting,** honey samples must be first sent to the labs, so **the reported results are not received until 1-2 weeks later**, thus negatively impacting on their business, and on the honey processing chain during the mixing step.

✗ Last but not least, the overall assessment of the pollen analysis **strongly depends on the technician's skills and experience**, as well as the time invested in the cell-counting step, thus **affecting the final conclusions for the honey and test reproducibility**. In addition, **the results are subjected to human error**. As an illustrative experiment, our team sent a sample of the same monofloral honey to 4 different well-known laboratories, and the results obtained on pollen concentration for the predominant floral source **presented a deviation above 25% from one to others.**

With Honey.AI [www.honey-ai.com]**, we aim to automatize the tedious work of pollen analysis for honey's floral source authentication with higher accuracy, using artificial intelligence and an IoT robotized low-cost microscope. Our affordable solution standardizes the pollen counting measurement, reduces time, allow on-site real-time measurements, increases reproducibility/repeatability of results, and immensely reduces human dependency**. Honey.AI economically incentivise the production of genuine honey products, and potentially assist beekeepers to increase their economic margins (which would indeed indirectly be a support for bee populations).



This first functional version of Honey.AI device, developed in 2022, still had specific limitations that we aimed to solve. **With VEDLIoT Project we have been able to complete different milestones that we had in our development and commercialization roadmap:**

- ✓ Incorporate **two new functionalities** to HoneyAI: colour assessment & starch detection
- ✓ **Expand datasets** for pollen recognition and other particles
- ✓ **Migrate to the edge the improved particle detector and classifyier** and **assess different edge devices**.
- ✓ **Improve inference time** on the edge, **accuracy of predictions** and lead time per test
- ✓ **Refactor the whole architecture** and coding of APP, API, & AWS GPUs (queues system).
- ✓ **Design and build a new prototype** to be tested in the project with Jetson Nano and Jetson Xavier.
- ✓ Assess other different applications for HoneyAI, aside of the honey industry.
- ✓ Update the exploitation and commercialization strategy.

Within this document, we have included first the overall architecture of the soluion, then the description of the work done in the project, and how we have specifically used VEDLIoT tools to carry out specific tasks to facilitate decision making and improve performance.

3 Idea and architecture

3.1 Overall idea and objectives

The new version of Honey.AI has been conceived **including Edge-Computing technology integration, adapted for small/low-powered devices, coupled with an internal processor and a touchable screen**.

The APP still is the main user interface of Honey.AI, displayed on a 5-inch capacitive touch screen, and running on the SBC, which makes the device completely autonomous and does not depend on the external user's computer.

So, the existing AI pipeline BEFORE the project was:

1. Automated HoneyAI microscope automously scans the glass slide with the honey sediment in X-Y area, and also in the z-planes, in order to obtain the most focused images. There are 19 planes, of 2µm difference.
2. The pictures with empty contents (no particles) are dismissed with a simple OpenCV library that performs a first basic pre-filter.
3. The other pictures are processed, in the edge, and those images containing pollen grains are upload, through our API, to S3 Bucket in AWS.
4. The AI module in AWS cloud is mainly based on Convolutional Neural Networks architectures that were built in TensorFlow.
    a. the first model is a pollen detector, which detects the regions with pollen grains and feds them into the next pipeline. This double-step was established because in the edge had not enough accuracy (F1-score around 85% on validation set), so it was agreed to include a second pollen detector step in AWS to discriminate pollen grains from dirt, crystals and other particles as principal outlyiers. In this case, the accuracy increased to acceptable values.
    b. second and third models deployed in AWS are specialized in two specific pollen species recognition:
        i. First, the **target pollen specie**, which is the one that user specifies as monofloral honey. There is one different model trained for each specie of pollen desired. Depending on the honey chosen, specific models are loaded.
        ii. Second, to **discrimante those pollen grains coming from nectariferous plants or polliniferous plants**. It is important to further classify all pollen grains that are NOT target pollen into 2 families: nectariferous and polliniferous. This is because when providing the user with the concentration of the target pollen in the honey sample, it is calculated on a relative value only to nectariferous species. Poliniferous species are not considered in the global counting.

    The Deep learning models are required because : i) the pollen is three-dimensional and its shape is not symmetric in all its faces/axis, and ii) there are many pollen specimens sharing very similar morphological features.
5. At the end of a test, users receive a Results PDF at their inbox mail. The API also communicates with a MongoDB database where all the Results PDFs are stored, so the user can review and download the files on the Clients' Platform, the communication tool we use to coomunicate with the end-users. Our clients, through their user account within the web platform are able to check a historical archive of samples tested with results and a communication centre for the results received.

The overall lead time of a test is usually between 20 minutes and 2 hours. The lead time of an analysis is so variable due to the "criteria" defined by the DIN Directive for pollen analysis which stablishes that: pollen analysis is considered as correctly completed only once the expert has counted, at least, 500 pollen grains, which at least 400 of them are from nectariferous species. Also, in order to assure a good homogeneity of the sample in the glass slide, our API also include a third condition, in which, at least, 100 locations (XY positions) must be scanned before finishing the test.

So, for low concentrated honeys, and/or with low nectariferous content (infra-represented species), the finish criteria requires to scan a very large percentage of the glass slide, and most of the pictures are empty or without pollen grains. The percentage of these low concentrated honeys is significant. And our current customers, obviously, they want to decrease lead time per test. In consequence, this is the main reason why the particle detector and classification must be migrated to the edge within the microscope, basically to avoid sending thousands of images to the cloud. The benefits of edge migration are:

- ✓ Reduced lead time per analysis
- ✓ Reduced computing needs in AWS, so economic savings for us
- ✓ New honey analysis included in the local APP, processed with Edge GPUs, without the need to create AWS instances in the cloud: no costs associated for us and faster results for the user.
- ✓ For remote locations, with low bandwith, better overall satisfaction of users.
- ✓ And last but not least, if the microprocessor is independently connected to WiFi, to autonomously upload the pictures to the cloud, without the need of user's laptop, then HoneyAI becomes a standalone laboratory equipment which do not depend anymore on the OS it has, the antivirus, the firewall, the power speed, the administration permissions, etc.

The overall objectives of Honey.AI VEDLIoT Project were:

- Understand VEDLIoT technology and get familiar with the different tools available.
- Adapt Honey.AI's current system to the new architecture and mechanical requirements to integrate Jetson Xavier as additional different edge device.
- Re-code the APP to make it compatible with the new acquisition hardware and to compile in the new ARM architecture, as well as change the GUI to make it usable without a full OS running in the background (functions to configure, WIFI, network, OTA updates, power control etc.),
- Adapt our AI pipeline to the new improved pipeline to be further deployed in the embedded hardwares and compare different results using Kenning and EmbeDL.
- Migrate the **particle** detector and classifyier (not only pollen, but include all particles of interest) from current AWS cloud computing site to the specific HW selected, after detailed benchmarking done of the different alternatives available. Also integrate the colour and starch functionalities and new categories of "particles".
- Design, manufacture and assamble a prototype of Honey.AI 2.0, commission, test it in real operational field and assess its performance with the different edge devices and models.
- Update exploitation strategy and carry out specific market-oriented activities.

3.2 Architecture

As a general description, this is the architecture of our current solution:

1. **Scanning light microscope with a digital camera [the AI-powered IoT device]** ▸ Honey.AI is a standard commercial low-cost microscope coupled to a digital camera. It includes a robotized sample stage, built with three integrated high-precision stepper motors for X-Y-Z translational movement (2 micron precission), and a camera that take thousands of pictures per analysis. The main structure of the equipment is built from a commercial trinocular light microscope.The pollen analysis quantifies pollen concentration; thus the microscope must be capable to "scan" the glass slide area at 400X where the honey sample is spread. The pollen grains are three-dimensional and its shape is not symmetric in its faces/planes, thus the optics of the system needs to "focus" the image at different planes, so moving in Z plane to focus. It

includes a self-designed control PCB that is the responsible of controlling the 3 stepper motors (one for each axis) and the switches to calibrate the zero coordinates.

2. **Control APP** ▸ Through the HMI of a C++ application (that has been migrated from desktop application to the edge processor), the application oversees the communication and movement control of the positioning system (motorized axis in XYZ with precission of microns), which includes the board, as well as SDK communication with the digital camera. This APP also controls the communication with the cloud, using our API. This APP has complex routines to allow the autofocus, the glass slide scanning, and also now the AI edge processing particle detection and classification. This APP also includes a calibration protocol, and a simple OpenCV filter to dismiss “empty images” with nothing.This APP has been coded in C++, using QT framework, OpenCV for specific image processing (segmentation algorithms and contour analysis) and openSSL to secure the connection with the cloud.

3. **The API** ▸The API is the brain that receives and controls the test execution, delegating the analysis to the platform workers, and recovering the distributed computations to compute the pollen concentration (the system is flexible to accept other types of test apart from pollen analysis). The platform workers are Amazon EC2 instances that are created on-demand by the API. The API uploads the images to S3 so that the EC2 instance can process those images and return the number of pollens detected of each type through the API. The communication between the API and the worker was done before VEDLIoT project through a AMQP Broker, specifically RabbitMQ.

4. **The AI complete pipeline** ▸ Before the Project, as previously presented, the whole analysis was done in two different levels of GPU computing: a simple pollen detection step was already prepared to be done in the edge (with a Jetson Nano) and further pollen classification (with more than 25x2 models) was deployed with cloud computing with AWS GPUs on demand.

1. Before the Project, there were a filtering step in the edget in which it was identified in the pictures if not including pollen or honeydew elements, to further erase them and not uploading them to the cloud. Depending on the sample analized, empty pictures are between 10 and 60% of total. The first basic Project we did before Vedliot, was to migrate the pollen detector to Jetson Nano. However, our team developed a simpler model converted to TensorRT. This pre-

filter applies to every image obtained with the microscope within an analysis, including between 1k and 25k pictures, depending on the area of the glass slide scanned, although the average is 4k-8k. The threshold in this pre-filtering step is considered significantly low, to allow "doubtful" particles to enter into the pipeline.

2. The images that present pollen or honeydew elements that have been detected in this first Edge step are upload to the cloud and send to a second detection step, to eliminate the errors from the edge model.
3. These images, are now processed with two different streams.

Within this step 3, the decision about if the pollen grain is target or not it is based on thresholds scoring on confidence. Otherwise, it is further classified as pollineferous or nectariferous.

This pipeline is so complicated and innefficient because when the development of HoneyAI started by 2020, just with a very basic PoC, the database of pollen species that we had was extremely low, so detection and recognition had to be done in different steps instead on a large multiclasse (there are more than 300 species with honey market interest, only in EU).

Within this VEDLIoT Project, the pipeline has been evolved to an optimized more efficient one, eliminating replicated step and enhancing accuracy and processing times.

The work done, and further explain in section 4 is:

1. The pollen detector has been evolved to a **particle detector and classifier**. There are additional particles in the Honey that have interest, not only pollen, so:
    a. We have developed datasets for yeasts (different types) and starch (with and without polarizer). Yeasts indicates fermentation and starch can be an indicator of fraud, so both are very important. Also, in order to increase accuracy, we have included datasets for "bubbles" and "dirt" as negative classes for the models.
    b. We have evolved the simple detector to a **2-step particle-classification pipeline**
2. We have **eliminated the detectors in the cloud**, and only classifiers for target pollen and nectariferous/pollineferous are maintained.
3. We have migrated the whole pipeline to process high-resolution images, without any resizing on size after being collected with the camera. Original image size in the new pipeline is 2048 x 1536, instead of 1024x1024.
4. We have developed 2 additional analysis for honey quality analysis, both run on the edge:
    a. Starch detection analysis in honey
    b. Colour PFund measurement: During 2023 we have developed an R&D project with the Computer Vision Centre of Barcelona, who has helped us to improve and define a Pfund colour grading analysis in HoneyAI to mimic the values obtained with a Hanna Photometer (the gold-standard for honey sector). Our team already had developed a model before, but was very basic and the average error was not acceptable for commercial use yet. With this procedure, that we have migrated to C++ and integrated in the APP, we are capable to: 1) measure honey colour and confirm that fits botanical origin, 2) calibrate all the devices/cameras to obtain pictures at standardized RGB and brightness level, 3) apply a correction filter step to all the images taken with all the microscopes to reduce the variability between devices and assuring a colour-agnostic AI dataset.
5. We have extended the datasets for pollen trained. Currently there are more than 75 species.

6. We have refactored the architecture in the cloud and migrated to a Kubernetes elastic AWS GPU processing.
7. Our team has worked in the electromechanical and firmware part to adapt the last version of HoneyAI to completely adapt to new edge devices assessed (Jetson Nano and Jetson Xavier NX 8GB)

3.3 Relation to VEDLIoT

Honey.AI corresponds to a **clear IoT system built from an automated scanning microscope device, controlled by a specific firmware, and coupled with an artificial intelligence module hosted on the cloud**. Honey.AI provides a unique combination of the pillars of mechanical engineering, electronics, computer vision, deep learning, biology, and food technology.
After concluding a deep assessment based on commercial price, stocks policy, HW size, GPU computing capacity and CE compliance requirements of the different alternatives for the edge architecture our team had previously selected Jetson Nano.
With the VEDLIoT Project we aimed to boost our AIoT application in terms of performance, cost-benefit ratio, usability for AI deployment and neural networks accuracy improvement.

With this project we have take profit of VEDLIoT tools to:

- first analyzed the viability (performance and future price) of introducing μREC in HoneyAI device
- make use of Kenning to facilitate AI pipeline and decision-making on benchmarking and dataset pre-processing
- used EmbeDL to optimize the AI models and confirm the decrease on inference time on the edge for the most complex models
- and used the test beds available to expand the edge devices analysed: Xavier TX2

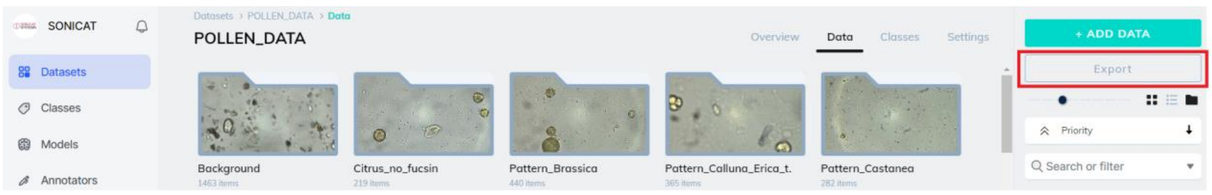
4 Implementation

4.1 Improvements of HoneyAI done in the project

4.1.1 Development of new datasets

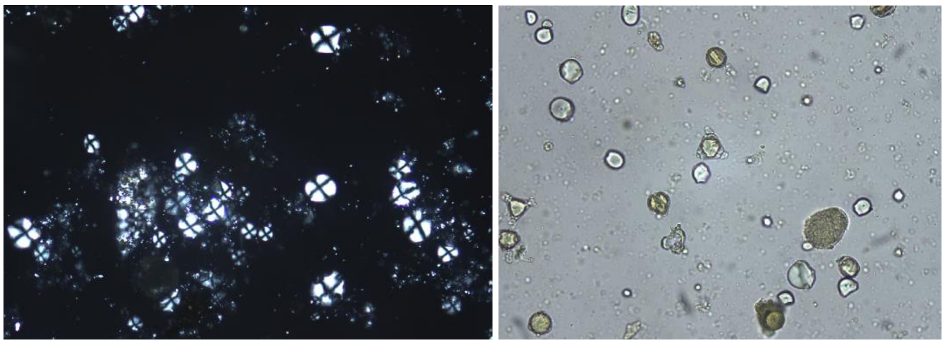
At Sonicat we have been analyzing different options during the last months to better implement dataset creatin with annotation platforms.

We have been assessing CVAT, Encord, and Darwin from V7. Different functionalities and releases have been appeared this period but finally selected V7 to continue with our work. Up to know, we have more than 75 species of pollen trained (nectariferous and pollineferous), and we have expanded the current database of pictures to more than 180,000 images.



Additionally, in order to to create the new functionalities, we have created the following datasets

- Dataset for yeasts, 1,200 instances, from different species potentially found in honey
- Dataset for starch grains with, and without polarizer. There are 4,560 instance without polarizer and 4,560 with polarizer. See examples of both below.



- Improved the dataset of honeydew elements
- Integrated new dataset of dirt and bubbles. The inclusion of new species annotated as negative classes has helped to increasy the accuracy of recognition capacity.

These new datasets have been elaborated at high resolution this time.

4.1.2 Development of starch screening functionality

As previously explained, starch particles are related with bad beekeeping practices, and also could be related to adulteration with sugar syrups.

Standard honeys should not include starch particles. However, up 15% starch/pollen grains can be considered as "usual", because of beekeepers feeding the bees sometimes with sugars. However, above 20% is a "warning", and above 30% is not recommended to put the honey in the market.

There are 2 different tests included:

a) **With polarizer -** In such a case, the detection and recognition is extremely simple and it is done with non-ML techniques. It is done autonomously of pollen analysis, because pollen cannot be seen with the polarizer.

b) **Without polarizer** – it is done during the pollen analysis, at the same time. As starch is now also recognized within the main particle detector/classifier, it is possible to include the relative concentration of starch within the pollen analysis functionality and PDF of results, with a specific note for the users in case the concentration is significantly high.

The pipeline in the APP has been updated and the the functionality included, as well as HMI of the APP. Also the PDF structure has been updated to include thise new information and introductory note for clients.

4.1.3 Development of pollen richness information

In order to mimic all the information provided by official laboratories when they do a pollen analysis, our team has also developed a new functionality to provide the client with the pollen richness. It indicates the quantity of pollen grains counted in 10 grams of honey.

The sample protocol has been updated to be prepared with a micropipette instead of a Pasteur Pipette to be able to better quantify the volume of sample spread in the glass slide, and the specific counting of pollen grains per specific number of locations included in the pipeline.

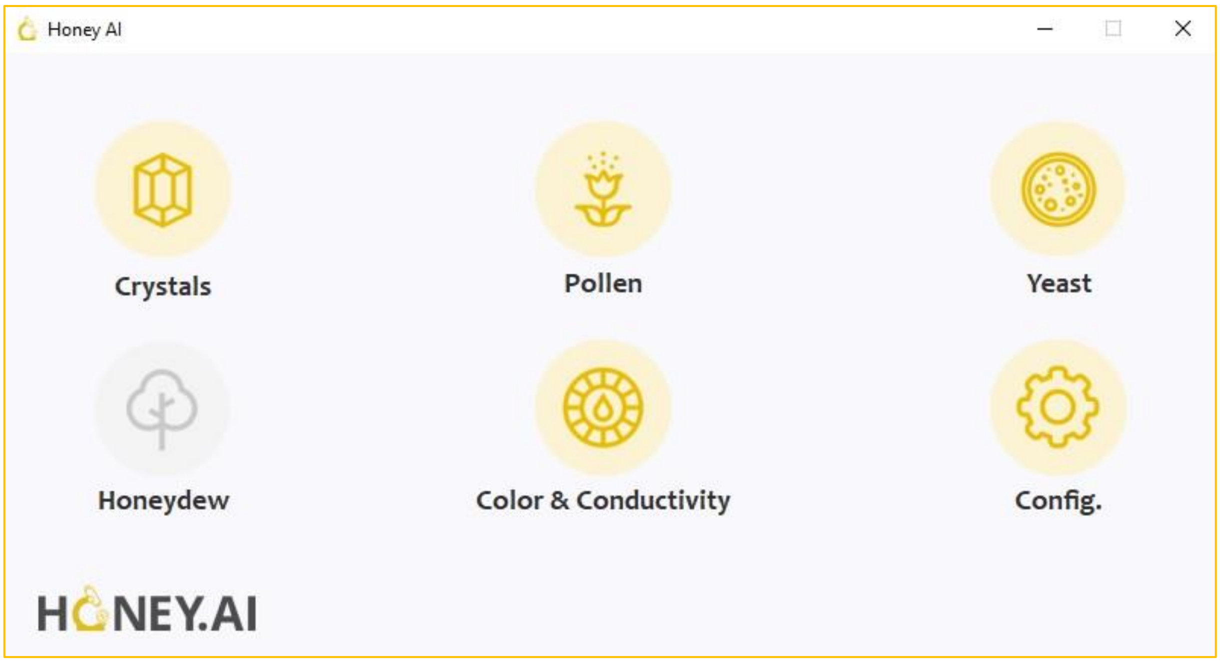
The pipeline in the APP has been updated and the the functionality included, as well as HMI of the APP. Also the PDF structure has been updated to include thise new information and introductory note for clients

4.1.4 Development of colour functionality

During the beginning of 2023, our team has developed an R&D joint project with Computer Vision Centre of Barcelona.

After 3 months of development, and a dataset of more than 250 honeys analyzed (done by Sonicat Systems) with the microscope and the Hanna Photometer, the overall solution is integrated.

The overall algorithm developed in Python to calibrate the microscope and to measure the values has been migrated to C++ and integrated in the pipeline. The pipeline in the APP has been updated and the functionality included, as well as HMI of the APP. Also the PDF structure has been updated to include thise new information and introductory note for clients.



4.2 VEDLIoT Tools

4.2.1 How we have used Kenning

One of the most important tasks in our pipeline was the selection of the model to be used for classification step on the edge. It would have been very cumbersome to test different models on different hyper parameters without the help of a proper pipeline.

After doing the initial tests we decided to go with EfficientNet model as it was the model which gave us the highest accuracy. The next step was to decide the variant of efficientnet model for which we took the help of Kenning pipeline and understand in depth various trade-off before finalizing the model. We tried and tested the following modules for Kenning:

Dataset ▸ Dataset module was created using the Kenning pipeline. Our dataset consists of cropped images (results from the detector), which consist of 5 classes : HDE, Starch, Yeast, Pollen and Dirt. Analyzing, splitting and pre-processing the dataset was done in this submodule.

ModelWrapper ▸ Using this module, we were able to define the model architecture and also specify the i/o specifications for the model for loading the model later. We had total of 3 models which we wanted to test

ModelCompiler ▸ In the model compiler part we converted the trained models into tflite int 8 format and also converted the pytorch model (which was compatible with embedl) into onnx format. The conversion was seamless and very easy to use.

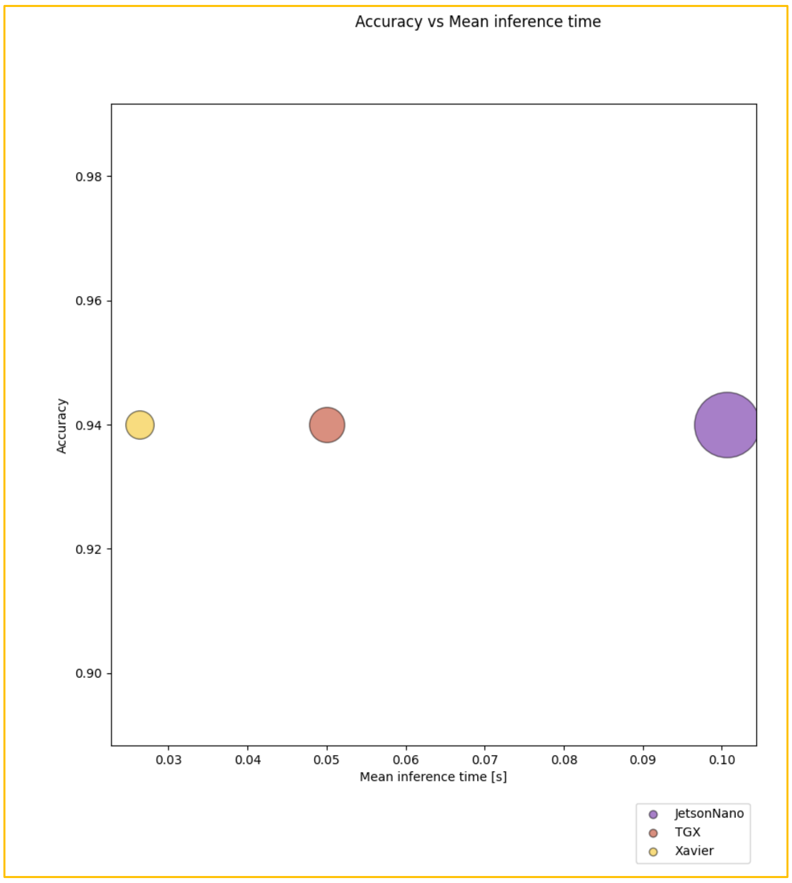
Though we had a problem because our pipeline is on tensorflow 2.4 (Jetson nano supports Cuda <= 10 and tensorflow > 2.4 needs Cuda 11), and while converting with tensorflow 2.4 we got a get_signature_runner error while converting it into tflite. However we changed our pipeline to higher version of tensorflow and this error was fixed.

ModelRuntime ▸ Using this class we loaded the tflite/onnx model, performed inference on native machine and also on the three hardwares and obtained the performance benchmarks.

Finally we used the render report functionality to get all the detailed reports summarized from the performance benchmarks and also to get an accurate overview of the comparison between different models.

For the initial hardware testing, we tested the int8 tflite models on three hardwares, Jetson Nano, Jetson Xavier and TX2 models. We mainly wanted to get an idea of how fast the models were running in the models the model which we were going to deploy as the final version.

The accuracy vs trade off graph model is given below tested against the three hardwares:



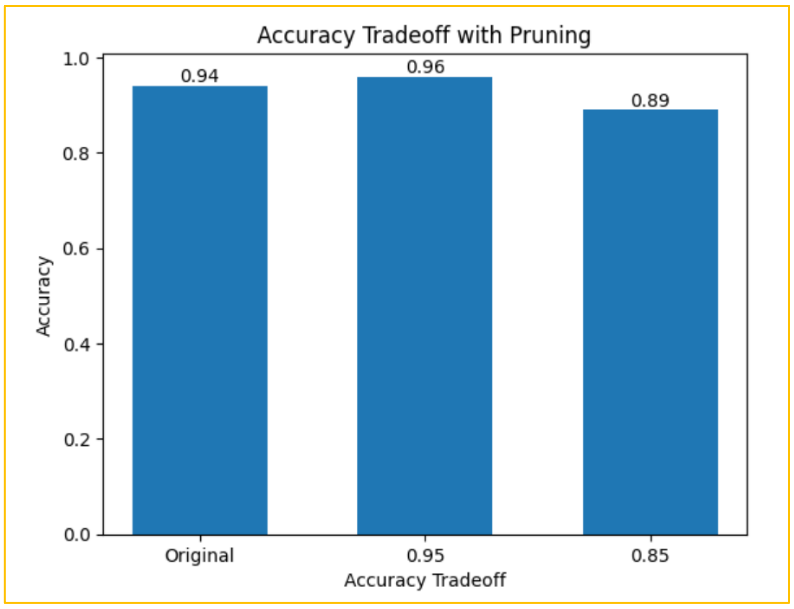
4.2.2 How we have used EmbeDL

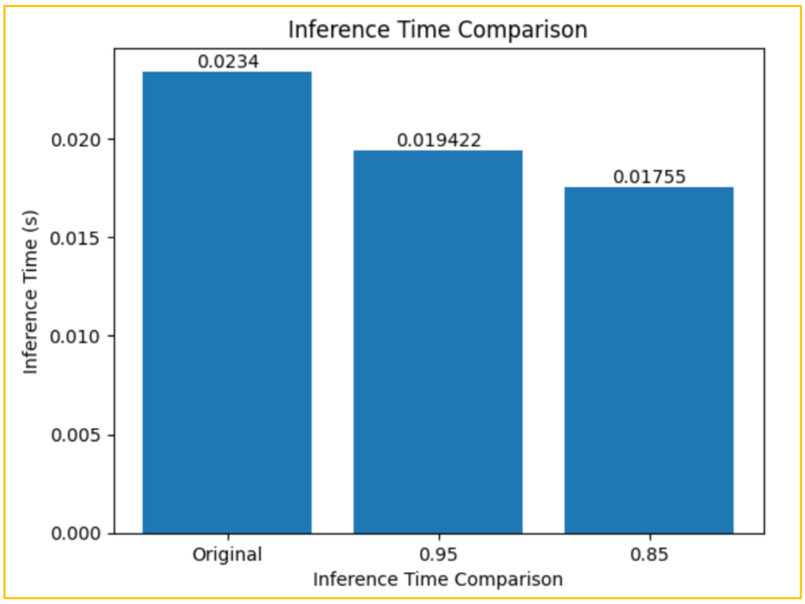
After the analysis carried out with Kenning we were able to determine that for our use case efficientnetB-1 would be the ideal model to carry out training and also test Embedl.

The following steps were carried out:

1. We migrated our training and inference pipelines to pytorch, (as it is supported by embedl).
2. First using the training module we trained a classifier with early stopping and patience as 3 (To get a fair comparison between the two models).
3. The trained model was then converted into onnx model using the Kenning pipeline.
4. The onnx model obtained was then tested in the three hardwares and the performance metrics was noted down.
5. The trained model was pruned using Taylor pruning of Embedl. Two models were obtained one pruned at 0.95 and the other at 0.85.
6. The pruned models were again trained with early stopping and patience set as 3.
7. Steps 3 and 4 was repeated.

Initially the two pruned models were taken and tested for its accuracy on Jetson Xavier. As we can see with pruned model 0.85 the accuracy is lowered, but with 0.95 we can see a slight increase in accuracy.





4.2.3 How we have used the Test Beds and edge hardware

Within the project, at the proposal stage, one of the ideas was to integrate µREC platform within the robot to be able to paralelize different steps and decrease lead time. The initial idea was to combine a Jetson device with a FPGA or a TPU accelerator.

However, after assessing potential solutions, and also assessing the commercial price of µREC platform coupled with the increase in price in Jetson devices, overall made the technical solution not economically feasible for our business case. Our target customers cannot afford so significant increase in price for the device. Furthermore, the overall improvement on the lead time would not be significant, since the analysis is limited in time by the mechanical speed of the motors and the acceleration/deceleration of each movement done.

HoneyAI does not work on video, but on pictures. And they must be extremely focused to ensure a good recognition precission, thus the microscope system must be completely stopped before taking the pictures. At this stage, only working with Jetson devices on the classification side, the use of further hardware integrated within μREC platform makes no sense for our solution.

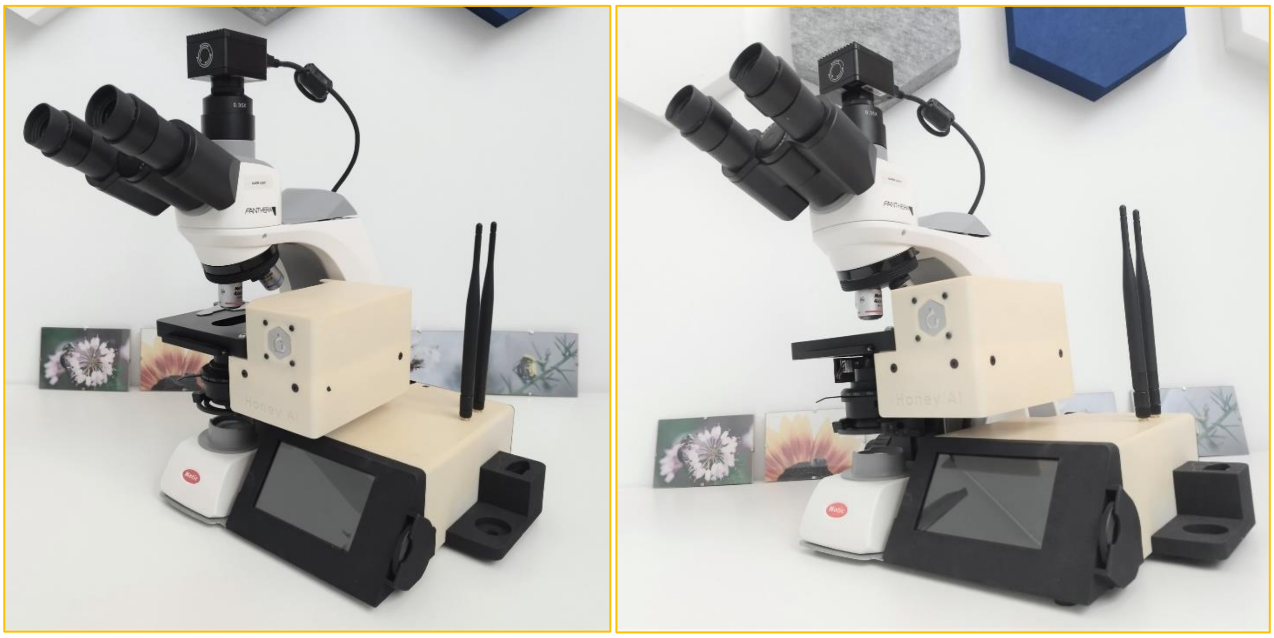
4.3 Manufacturing, testing, and commissioning

Within this project, there have been 2 different versions of HoneyAI to be tested with real analysis, integrating different devices. The Jetson Nano development kit and a Xavier NX 8GB module with a carrier board from another module.

Due to the larger dimensions of Jetson Xavier, it was required to adapt and adjust the CAD drawings to be able to enclose the AI-powered embedded device.

The mechanical components were re-designed and drawings obtained to purchase new parts and manufacture a new prototype for this project. Also within this project our team has analyzed different suppliers, engage new partners, and agreed on better conditions. Different materials also for the custom modules have been assessed.

The new prototype of HoneyAI has been manufactured and commissioned and tested in-house and validated all ubsystems interconnected.



5 Evaluation and characterization

5.1 Accuracy and Lead times

So, the total time is 1,302 secs per location. Considering 400 locations for instance as average, and the average of 8 particles per location, the result would be 8,7 minutes. However, the device is still limited on the hardware/APP side, which crearly is limiting decreasing the overall lead times.

The overall conclusions are that the AI pipeline can be still improved without impacting on the whole lead time due to the fact that the mechanical movement and image acquisition of HoneyAI is the limitant part, since it is required 14-18 minutes to cover 100 locations, which is the minimum locations to be covered, while AI would require around 9 (considering average values!).

There has been a significant increase in overall accuracy of the device with the improvements implemented and the new AI pipeline considered. There is still room for improvement by considering additional image processing steps on the edge.

On the other hand, our team is already focused into trying to reduce the operational time required by the APP and the mechanics, mainly based on developing a solution to allow reducing the number of planes to be captured when the location is empty, or when the pollen grains have been already effectively scanned. Also, it will be required to analyse in detail if 19 planes are required or can be reduced.

	Honey/pollen	Lead Time (min)	Result	Manual counting	Difference (%)
1	*Sunflower*	31	41,74	51,8	10,06
2	*Sunflower*	20	60,5	62	1,5
3	*Sunflower*	32	40,88	37,79	3,09
4	*Sunflower*	32	42,08	43,4	1,32
5	*Sunflower*	35	52,3	55,4	3,1
6	*Sunflower*	24	46,25	44	2,25
7	*Sunflower*	18	29,1	26,7	2,4
8	*Sunflower*	18	57,45	53,12	4,33
9	*Acacia*	65	18,93	20,3	1,37
10	*Acacia*	72	28,6	33,6	5
11	*Acacia*	54	49,86	61,49	11,63
12	*Acacia*	74	13,45	30,5	17,05
13	*Acacia*	85	22,02	24,67	2,65
14	*Acacia*	84	21,32	16,1	5,22
15	*Acacia*	71	26,97	33,74	6,77
16	*Acacia*	75	36,49	30,69	5,8
17	*Rosmarinus*	44	8	7,2	0,8
18	*Rosmarinus*	45	8,72	6,7	2,02
19	*Rosmarinus*	38	11,28	5,2	6,08
20	*Rosmarinus*	39	13,23	16,5	3,27
21	*Rosmarinus*	29	51,8	48,7	3,1
22	*Rosmarinus*	35	16,7	30	13,3
23	*Rosmarinus*	49	3,23	5,4	2,17
24	*Rosmarinus*	32	21,7	27,65	5,95
25	*Rosmarinus*	31	16,95	31,96	15,01

26	*Rosmarinus*	31	19,5	24,18	4,68
27	*Rosmarinus*	50	4,04	6,12	2,08
28	*Rosmarinus*	43	9,06	4,63	4,43
29	*Citrus*	74	9,09	8,37	0,72
30	*Citrus*	125	0,48	2,00	1,52
31	*Citrus*	119	1,50	2,00	0,5
32	*Citrus*	123	0,74	0,30	0,44
33	*Citrus*	124	0,69	1,60	0,91
34	*Citrus*	61	25,93	12,16	13,77
35	*Citrus*	87	3,60	2,50	1,1
36	*Citrus*	43	39,95	43,20	3,25
37	*Citrus*	85	5,45	8,97	3,52
38	*Citrus*	54	32,7	28,8	3,9
39	*Citrus*	45	51,75	51	0,75
40	*Thymus*	78	12,46	9,5	2,96
41	*Thymus*	110	1,71	6,3	4,59
42	*Thymus*	98	1,53	2,5	0,97
43	*Thymus*	68	11,94	47	35,06
44	*Thymus*	65	10,12	9,11	1,01
45	*Thymus*	89	5,25	2,3	2,95
46	*Thymus*	85	8,03	4,5	3,53
47	*Eucaliptus*	18	91,64	88,7	2,94
48	*Eucaliptus*	18	82	89,4	7,4
49	*Eucaliptus*	32	52,7	61,7	9
50	*Eucaliptus*	43	6,63	7,5	0,87
51	*Eucaliptus*	18	81,37	80,1	1,27
52	*Eucaliptus*	33	42,67	51	8,33
53	*Eucaliptus*	21	71,28	72,3	1,02
54	*Eucaliptus*	21	72,18	76,2	4,02
55	*Eucaliptus*	23	67,07	76,3	9,23
56	*Castanea*	33	56,03	56,14	0,11
57	*Castanea*	21	42,3	35,79	6,51
58	*Castanea*	20	56,03	56,14	0,11
59	*Castanea*	18	91,49	89,5	1,99
60	*Castanea*	18	94,18	92,1	2,08
61	*Castanea*	18	92,2	95,5	3,3
62	*Castanea*	18	96,06	98	1,94
63	*Castanea*	19	69	75,45	6,45
64	*Castanea*	18	97,06	98	0,94
65	*Castanea*	18	95,9	97	1,1
66	*Castanea*	19	64,96	62,19	2,77
67	*Lavandula*	45	8,76	12,13	3,37
68	*Lavandula*	56	4,11	9,2	5,09
69	*Lavandula*	43	7,01	7	0,01
70	*Lavandula*	68	2,45	1,2	1,25

71	*Lavandula*	34	29,7	27,62	2,08
72	*Lavandula*	37	11	8,08	2,92
73	*Lavandula*	50	4	3,53	0,47
74	*Lavandula*	58	6,18	2,6	3,58
75	*Lavandula*	41	14,54	6,02	8,52
76	*Lavandula*	98	4,54	0,59	3,95
77	*Erica*	46	22,3	14,68	7,62
78	*Erica*	57	28,74	17,2	11,54
79	*Erica*	38	44,88	40,88	4
80	*Erica*	95	3,76	1,1	2,66
81	*Rubus*	62	38,16	32,6	5,56

5.2 Colour functionality

The colour functionality has been tested with real samples in order to analyze the error compared to HANNA photometer. Overall the performance has been very good and the average error is below than the limit initially defined (±5 PFund).

HANNA	CVC	DESVIATION
0	0	0
1	1.4256	0.4256
2	1.4232	0.5768
3	0	3
6	8.5283	2.5283
8	2.999	5.001
16	21.0727	5.0727
16	12.8576	3.1424
19	18.383	0.617
22	25.6415	3.6415
26	27.0658	1.0658
30	31.609	1.609
36	39.7368	3.7368
40	32.8042	7.1958
43	41.3087	1.6913
58	59.8944	1.8944
60	61.0875	1.0875
64	66.9238	2.9238
67	74.3522	7.3522
74	74.3774	0.3774
77	74.2208	2.7792
78	90.4063	12.4063
80	93.143	13.143
86	71.5554	14.4446
95	96.2228	1.2228
108	111.7706	3.7706
113	114.8508	1.8508
119	119.7406	0.7406
123	118.908	4.092
135	133.058	1.942
137	134.844	2.156
142	138.0611	3.9389
150	150	0
	ERROR MIG	2.5283

6 Exploitation

6.1 Dissemination/communication/marketing/exploitation activities

6.1.1 Fairs and congresses and events

Since June 2022, our team has attended different events/congresses/fairs to disseminate and communicate HoneyAI. A list of them:

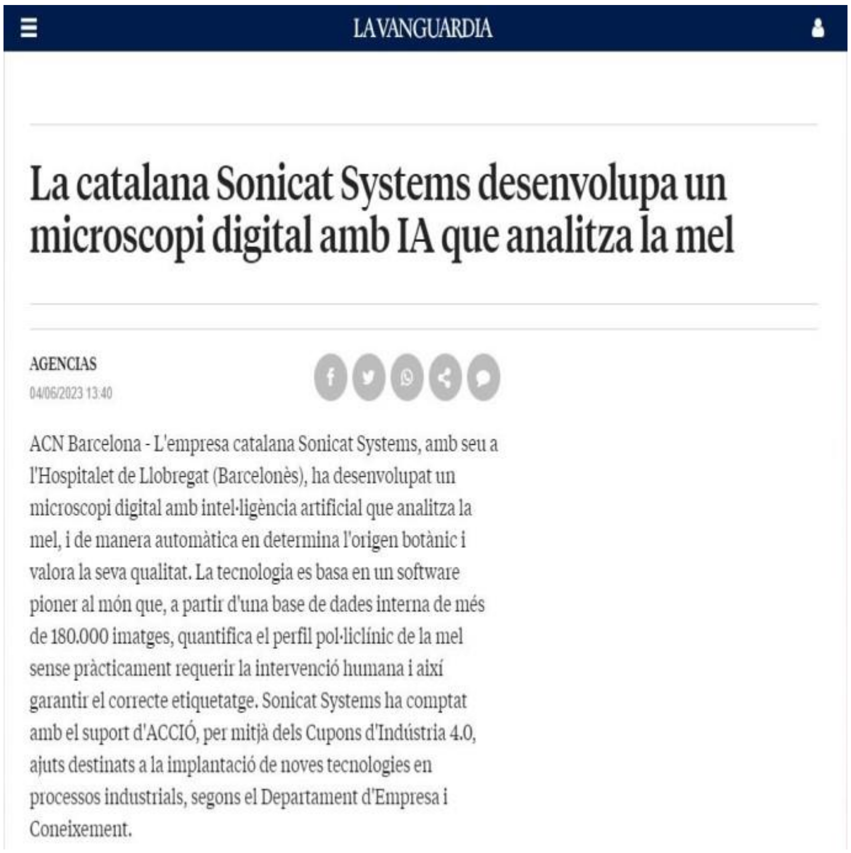
- Communication of VEDLIoT Project selection on the news tab
- Oral presentation in Apimondia 2022

- Honey.AI is presented at the Pastrana 2023 International Beekeeping Fair
- Honey.AI participated in the Meliza 2023 Beekeeping Fair

HONEY.AI
HONEY.AI
AUTOMATED POLLEN ANALYSIS
Análisis de calidad en miel
in-situ, automatizados
y en menos de 1 hora
Razones para invertir en Honey.AI

- Webinar for #FNAP [Federação Nacional dos Apicultores] in Portugal

- Press release – HoneyAI appears in many digital newspapers



LA VANGUARDIA

La catalana Sonicat Systems desenvolupa un microscopi digital amb IA que analitza la mel

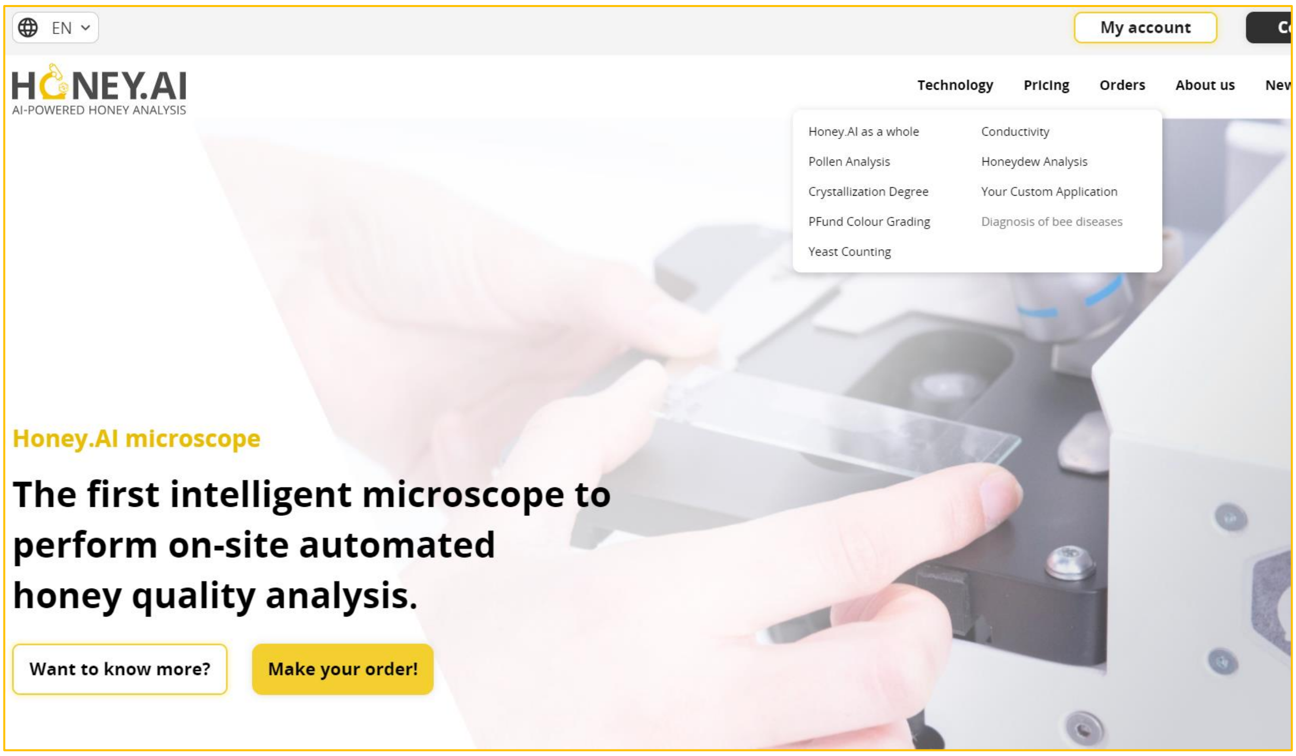
AGENCIAS
04/06/2023 13:40

ACN Barcelona - L'empresa catalana Sonicat Systems, amb seu a l'Hospitalet de Llobregat (Barcelonès), ha desenvolupat un microscopi digital amb intel·ligència artificial que analitza la mel, i de manera automàtica en determina l'origen botànic i valora la seva qualitat. La tecnologia es basa en un software pioner al món que, a partir d'una base de dades interna de més de 180.000 imatges, quantifica el perfil pol·liclínic de la mel sense pràcticament requerir la intervenció humana i així garantir el correcte etiquetatge. Sonicat Systems ha comptat amb el suport d'ACCIÓ, per mitjà dels Cupons d'Indústria 4.0, ajuts destinats a la implantació de noves tecnologies en processos industrials, segons el Departament d'Empresa i Coneixement.

- Oral presentation within the final event of Qualify European Project (invited by the Catalan Government to introduce HoneyAI as an example of AI-powered devices for fraund inspection in agrifood products.

6.1.2 Website

During 2023, we have updated the contents in our website to better explain the new functionalities of colour and also mentions of the starch. It has been included within a new “technology” tab in which the contents specifically describe the different analysis available with our system, the benefits of using them, and an example of results.



Honey.AI microscope

The first intelligent microscope to perform on-site automated honey quality analysis.

We have also included a Newsletter API on our website, of Mailjet, to allow potential customers to subscribe to HoneyAI updates.

6.1.3 Logos

Within our policy of one-stop-shop for the device, the tagline of our logo has been updated, in order to introduce the idea of "different analysis in honey", not only pollen.

Old tagline: Automated pollen analysis

HONEY.AI
AUTOMATED POLLEN ANALYSIS

New Tagline: AI-powered honey analysis

HONEY.AI
AI-POWERED HONEY ANALYSIS

We have also developed a logo for the other application that we are developing: FermentAI.

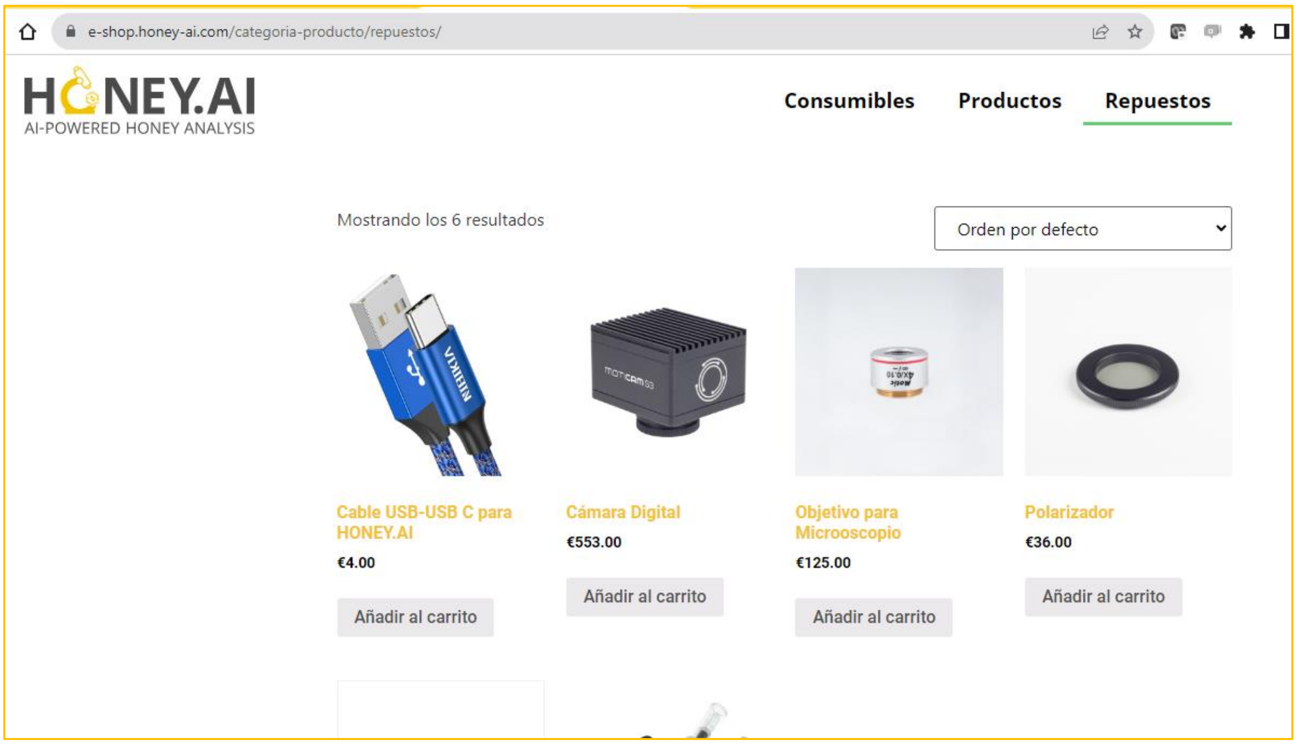
The overall concept is to keep the same typography, as well as the microscope icon (adapted to specific application).

Ferment.AI

6.1.4 e-shop integration

Thanks to a national small funding grant (Kit Digital), we have been able to subcontract a specific firm to integrate a wordpress woocommerce section for e-shop in our website.

Our team has prepared the pictures, contents, graphical design and preliminary information. The e-shop allows buying credits, and consumables. It is private, specifically conceived for the HoneyAI registered clients.



e-shop.honey-ai.com/categoria-producto/repuestos/

HONEY.AI
AI-POWERED HONEY ANALYSIS

Consumibles Productos Repuestos

Mostrando los 6 resultados

Orden por defecto

Cable USB-USB C para HONEY.AI	Cámara Digital	Objetivo para Microoscopio	Polarizador
€4.00	€553.00	€125.00	€36.00
Añadir al carrito	Añadir al carrito	Añadir al carrito	Añadir al carrito

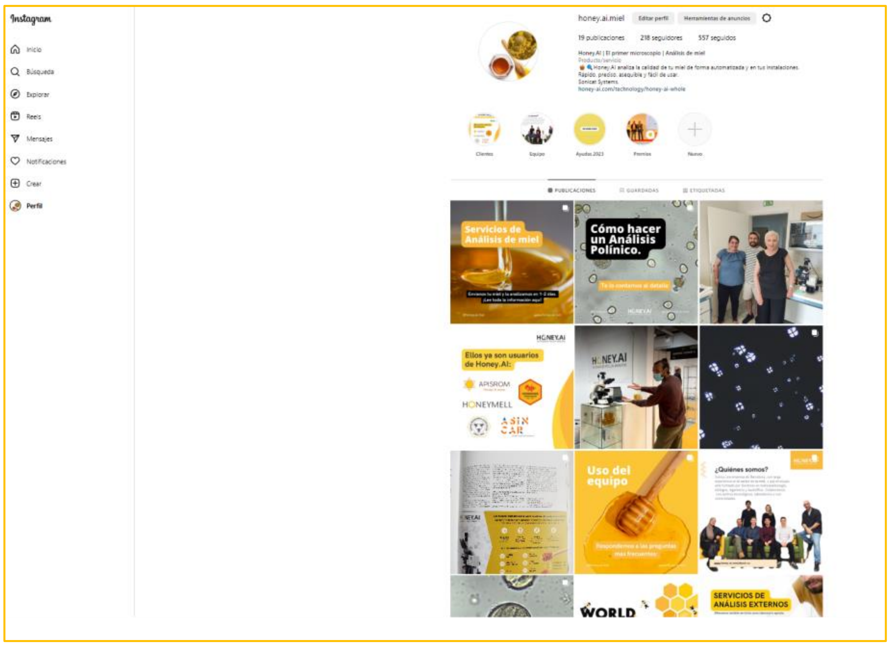
6.1.5 Promotional videos and Instagram profile

We have developed commercial videos of HoneyAI:

Promotional video: https://www.linkedin.com/showcase/82698416/admin/

Crystallization application: https://www.youtube.com/watch?v=sRmqsA5fCpI

We have opened a Instagram profile (Honey.ai.miel) to get closer to potential users and better explain how AI works in our device.



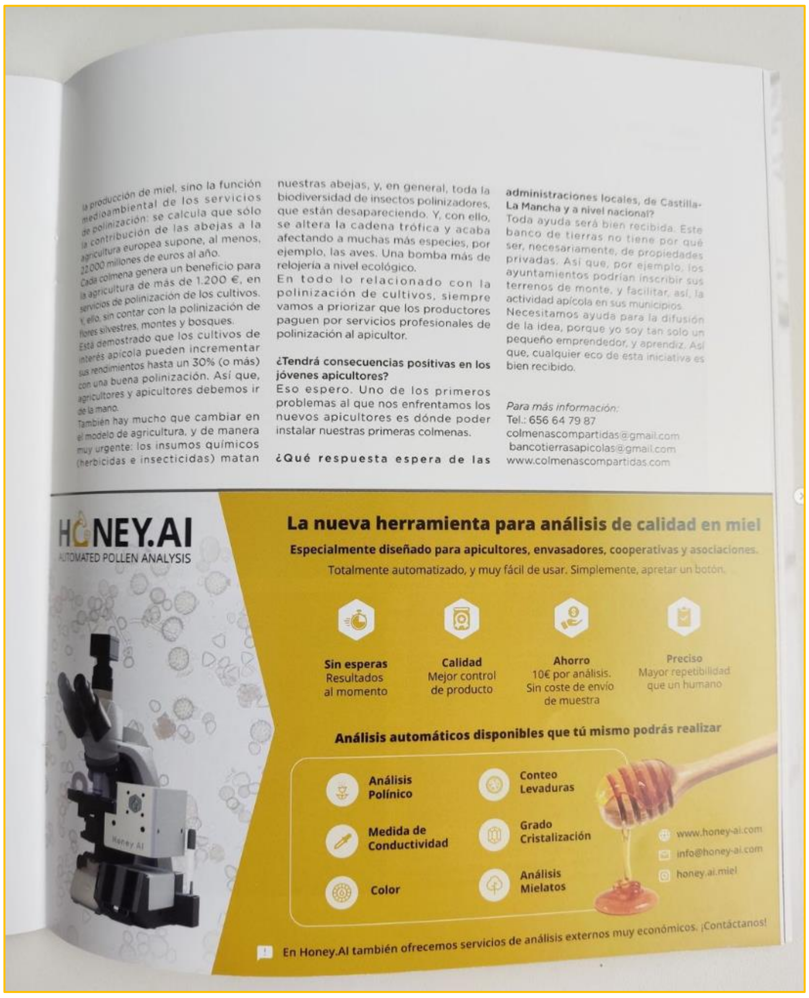
honey.ai.miel

19 publicaciones 218 seguidores 557 seguidos

Honey.AI | El primer microscopio | Análisis de miel
Producto/servicio
🔍 Honey.AI analiza la calidad de tu miel de forma automatizada y en tus instalaciones.
Rápido, preciso, asequible y fácil de usar.
Sonicat Systems.
honey-ai.com/technology/honey-ai-whole

6.1.6 Publicity in beekeeping journal

HoneyAI has been featured in the most important beekeeping journal in Spain: Vida Apicola:



la producción de miel, sino la función medioambiental de los servicios de polinización: se calcula que sólo la contribución de las abejas a la agricultura europea supone, al menos, 22.000 millones de euros al año.
Cada colmena genera un beneficio para la agricultura de más de 1.200 €, en servicios de polinización de los cultivos. Y, ello, sin contar con la polinización de flores silvestres, montes y bosques.
Está demostrado que los cultivos de interés apícola pueden incrementar sus rendimientos hasta un 30% (o más) con una buena polinización. Así que, agricultores y apicultores debemos ir de la mano.
También hay mucho que cambiar en el modelo de agricultura, y de manera muy urgente: los insumos químicos (herbicidas e insecticidas) matan nuestras abejas, y, en general, toda la biodiversidad de insectos polinizadores, que están desapareciendo. Y, con ello, se altera la cadena trófica y acaba afectando a muchas más especies, por ejemplo, las aves. Una bomba más de relojería a nivel ecológico.
En todo lo relacionado con la polinización de cultivos, siempre vamos a priorizar que los productores paguen por servicios profesionales de polinización al apicultor.

¿Tendrá consecuencias positivas en los jóvenes apicultores?
Eso espero. Uno de los primeros problemas al que nos enfrentamos los nuevos apicultores es dónde poder instalar nuestras primeras colmenas.

¿Qué respuesta espera de las administraciones locales, de Castilla-La Mancha y a nivel nacional?
Toda ayuda será bien recibida. Este banco de tierras no tiene por qué ser, necesariamente, de propiedades privadas. Así que, por ejemplo, los ayuntamientos podrían inscribir sus terrenos de monte, y facilitar, así, la actividad apícola en sus municipios.
Necesitamos ayuda para la difusión de la idea, porque yo soy tan solo un pequeño emprendedor, y aprendiz. Así que, cualquier eco de esta iniciativa es bien recibido.

Para más información:
Tel.: 656 64 79 87
colmenascompartidas@gmail.com
bancotierrasapicolas@gmail.com
www.colmenascompartidas.com

HONEY.AI
AUTOMATED POLLEN ANALYSIS

La nueva herramienta para análisis de calidad en miel

Especialmente diseñado para apicultores, envasadores, cooperativas y asociaciones.

Totalmente automatizado, y muy fácil de usar. Simplemente, apretar un botón.

Sin esperas Resultados al momento

Calidad Mejor control de producto

Ahorro 10€ por análisis. Sin coste de envío de muestra

Preciso Mayor repetibilidad que un humano

Análisis automáticos disponibles que tú mismo podrás realizar

Análisis Polínico
Conteo Levaduras
Medida de Conductividad
Grado Cristalización
Color
Análisis Mielatos

www.honey-ai.com
info@honey-ai.com
honey.ai.miel

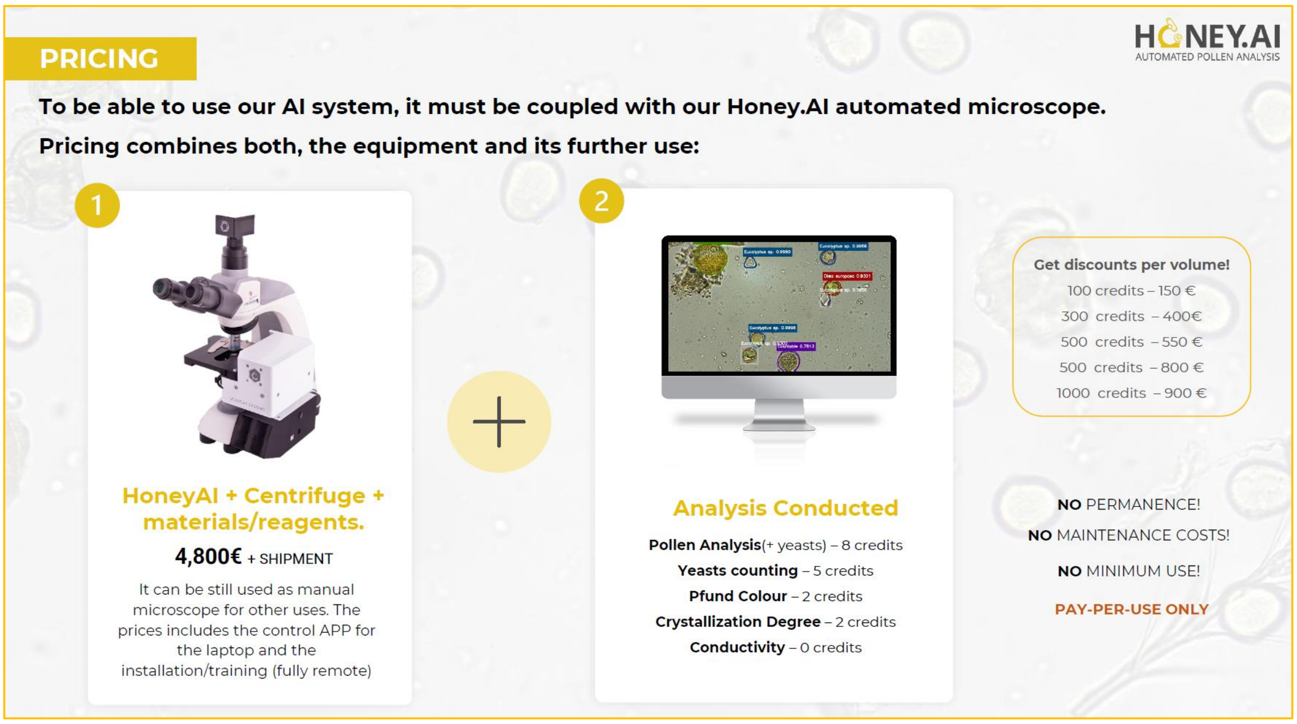
En Honey.AI también ofrecemos servicios de análisis externos muy económicos. ¡Contáctanos!

6.1.7 Pricing update

The pricing stretagy has been updated since the proposal stage. Now, there are only 2 revenue streams, not 3, specifically:

1. Sale of the device upfront, at 4,800 euros (both, the version with and without Jetson Nano), including also all the reagents, glass slides, lab materials, centrifuge and remote installation & training.

2. Pay-per-use, with a "credits" policy. The client buys a package of credits in our e-shop. The more credits they buy upfront (one year to use them), the higher the discount they get per credit. Then, the client can use the robot, so every time a test is conducted, the number of credits are discounted.


PRICING

HONEY.AI AUTOMATED POLLEN ANALYSIS

To be able to use our AI system, it must be coupled with our Honey.AI automated microscope.
Pricing combines both, the equipment and its further use:

1

HoneyAI + Centrifuge + materials/reagents.

4,800€ + SHIPMENT

It can be still used as manual microscope for other uses. The prices includes the control APP for the laptop and the installation/training (fully remote)

+

2

Analysis Conducted

Pollen Analysis(+ yeasts) – 8 credits
Yeasts counting – 5 credits
Pfund Colour – 2 credits
Crystallization Degree – 2 credits
Conductivity – 0 credits

Get discounts per volume!
100 credits – 150 €
300 credits – 400€
500 credits – 550 €
500 credits – 800 €
1000 credits – 900 €

NO PERMANENCE!
NO MAINTENANCE COSTS!
NO MINIMUM USE!
PAY-PER-USE ONLY

A new business model will be tested after summer in which a large discount (up to 50% discount) is negotiated with the client if they commit on a minimum subscription model per month, starting from 20 pollen analysis, at 8 credits each.

6.2 Suistanability

6.2.1 Sales and funnel

The commercialization of HoneyAI started by February 2023, in which our team initiated the commercial actions.

The version that we are selling now does not include yet the edge version, due to two reasons:

1. First, due to the problems with stocks and increase in prices of the embedded device. NVIDIA suffered shortages of components so was difficult and expensive to receive devices at acceptable prices. Also, our team still was waiting for final conclusions of this project to decide on the ai-powered embedded device to integrate in HoneyAI.

2. Second, the current version of the C++ APP is coded with Qt framework. We are using the open source option, since we only use specific basic libraries and we do not communicate the use of Qt framework and allow the user to easily access and modify the code. However, if migrating to the edge, it is not easy to allow the users to access the code, and also we do not want them to be able to access the classifier models for particle recognition. After several discussion with Qt sales team, it was decided to eliminate Qt from our APP (license is pay-per-device model and very expensive) and migrate to HTML5 and CSS for HMI and Neutralino for servers. It is planned to be done from September to November 2023.

Currently, there are 6 paying customers, and 4 additional clients under conversations. One of these clients is the central laboratory of la Xunta de Galicia, the Spanish Government of Galicia Region, managed directly by the Spanish Ministery of Agricuture, fishery and food.

We have already clients in Romania, Norway, Cyprus, Spain, Lebanon, Denmark and Saudi Arabia.

In addition, we have started to offer also external services as AI-powered laboratory, so the honey stakeholders can send us their honeys and our technicians process the samples and send the results at affordable price and short delivery times.

6.2.2 New name, and other applications

6.2.2.1 Microfy Systems

Sonicat Systems was the company name defined when started its activity by developing processing machines with high power ultrasounds.
However, we have recently pivoted and centered our activity in the automated microscope and the AI platform. In that sense, we have decided to change company's name and recently done by 31st July.
New name is Microfy Systems. Has been already registered, and the logo designed, in line with our corporative image for Honey.AI and Ferment.AI.

Microfy.AI

And the icon:

6.2.3 Funding and loans and grants

Our team is working not only on commercial sales and marketing, but also on:

- **Raising money** - a seed round of 400k is being requested. Our CEO has joined different events for startups and VCs, as ESADE BAN, SUPERNOVAS Event, DealFlow Agritech, etc.
- **Requesting specific R&D loan** – ENISA Agroinpulso in Spain, and IBEROEKA together with a Argentina cooperative that aims to help us to create the datasets for their region.
- **Grants** – actively submitting othere proposals for cascade funding and grants. Currently preparing an Eurostars, and 2 cascade fundings. In addition, we have been recently granted with a Mind4Machines cascade funding and DREAM cascade funding. The last one is to implement the multi-classe models, to be able to provide full spectrum of pollen analysis to clients.

7 Summary and next steps

For 2023-2024, there are specific goals/objectives to be met:

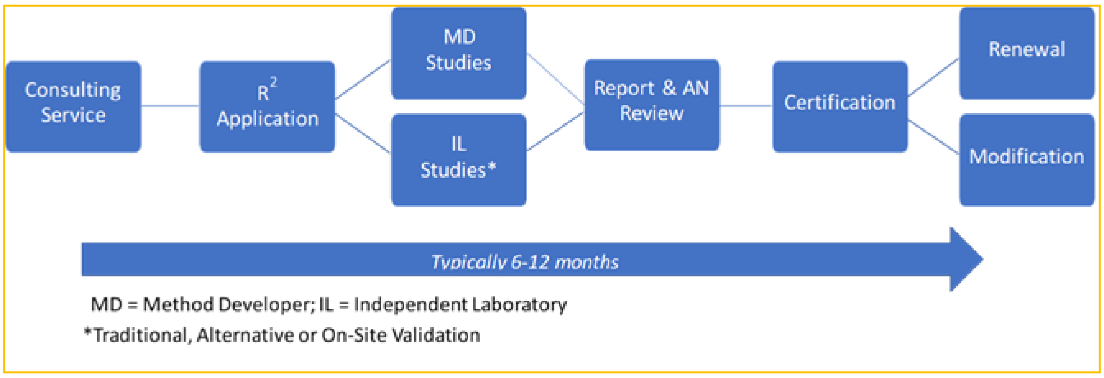
- ✓ **Increase accuracy** ▶ Still there is the need to improve the accuracy of the predictions. Due to the fact that the limiting time now is the hardware/mechanical/picture acquisitions, there is still room to improve the AI pipeline both, in the edge and also in the cloud. Our team will assess still other backbones and models to increase the accuracy and add additional image processing steps and/or further DL models.
- ✓ **Migrate the pipeline to multiclasse** ▶ As described on 4.1.4, the detectors have been eliminated of the AI pipeline. However, there is still room to improve by evolving the current architecture of target pollen individual search to a unitary multiclasse pollen classifier where all the pollen species (per region) are considered. The multiclasse would be hosted in the cloud, although we are planning to deveolop a specific version of HoneyAI in which we could integrate the whole pipeline on the embedded device. This version would target those clients located in remote areas with low bandwith, for instance those in Greece, where internet connection is quite bad. Within this stage, we will keep working with Kenning and EmbeDL to assess them.
- ✓ **Expand Honey Database** ▶ Our intention is to go global. In that sense, it is essential for us to increase the number of species trained in our AI-module. Also, for the multiclasse model is it important that the model has been trained with all the potential classes that can appear on the honey samples, otherwise the precission would decrease.
- ✓ **Develop Nosema application** ▶ during the end of 2023, we will implement the specific functionality of Nosema spores automated detection to diagnose Nosemosis in bees. The Spanish Ministery is assisting us to develop the functionality and as soon as it is ready, we already count with different clients that would invest in HoneyAI to carry out these veterinary analysis. Can be done with the same robot, but using x60 magnification, and a specific chamber similar to a hemocitometre.
- ✓ **Develop other applications** ▶ As presented, our intention is to keep working on developing the next applications according to the needs and requirements of the companies we are collaborating with.
- ✓ **Homologate Microval or AOAC** ▶

  The melissopalynology process is generally regulated by the official procedure defined into the “Harmonized methods of melissopalynology” and also the DIN 10760, which describes the sample preparation protocol and the classification and counting protocol for relative distribution definition of each pollen specie in a honey sample.

  The specialized laboratories that perform these analysis within the honey industry have been previously accredited by the specific accreditation body of their countries, namely ENAC for Spain, or DAkks for Germany. **It is not a must to be accredited to sell services of melissopalynology**, but it's a good practice in order to generate trust among clients.

  Honey.AI can be accredited, but it is very difficult to meet the criteria if you are not a laboratory. Tthe overall method and the performance of the system can be certified according to official organisms, such as Microval and AOAC.

The procedure from AOAC from instance has 6 steps. Once approved, the method is awarded a unique AOAC Reviewed and Recognized certification number. The Method Developer is licensed to use the AOAC Reviewed and Recognized certification mark. **The procedure has a cost of 21k€, and pending to add the cost of the IL procedure**.



- ✓ **Integrate e-billing and e-wallet ▶** finally, we also want to integrate a real IoT communication with the device in which the client can perform tests on the stand-alone device and every time that a test is done, with a token service, the unit connects with the e-wallet in the cloud and automatizally discounts the credits from the e-wallet. Otherwise the analysis is not processed. Now, this process is being done manually, but because the number of clients is still limited.

ICT-56-2020 - Next Generation Internet of Things

MushR Technical Report

Document information	
Contract number	957197
Project website	www.vedliot.eu
Dissemination Level	CO
Nature	R
Contractual Deadline	30.06.2023
Authors	Prof. Dr. Benjamin Leiding, Anant Sujatanagarjuna, Shohreh Kia
Contributors	Prof. Dr. Benjamin Leiding, Anant Sujatanagarjuna, Shohreh Kia
Reviewers	

The VEDLIoT project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957197.

Changelog		
Version	**Date**	**Type**
V 1.0	07.07.2023	Final Draft
V 1.1	21.07.2023	Final Draft (with corrections)

Table of contents

Executive Summary

The digitization of the gourmet mushroom production line has emerged as a transformative approach to enhance productivity, efficiency, and quality in mushroom cultivation. This project aimed to leverage digital technologies and intelligent automation to optimize every stage of the production process. By setting up controlled environment tents equipped with smart sensors, environmental conditions such as temperature and humidity were closely monitored and controlled, ensuring an ideal growth environment for the mushrooms. An image capture and storage system, integrated with a Mask R-CNN [1] model, enabled real-time monitoring and accurate detection of mushroom maturity levels. The implementation of a semi-automated harvesting system further streamlined the production line, automating the precise cutting of mature mushrooms. The project's outcomes showcased the potential of digital technologies and automation to revolutionize the mushroom industry, resulting in increased yield, reduced labor requirements, improved product quality, and enhanced decision-making capabilities. As the world faces the challenges of food security and sustainability, embracing digital solutions in agriculture becomes increasingly vital. This project serves as a testament to the potential of digitization in optimizing mushroom cultivation practices and paves the way for further advancements in digital agriculture.

1 Introduction

The digitization of production processes has revolutionized numerous industries, enabling increased efficiency, improved monitoring capabilities, and enhanced decision-making. In this report, we present a comprehensive overview of a project focused on the digitization of the production line for oyster mushrooms.

The primary objective was to leverage advanced technologies to streamline the cultivation, monitoring, and harvesting of oyster mushrooms. The project began with the setup of two tents within our workspace, where a specialized oyster mushroom cultivation system was implemented. The tents were equipped with racks containing square buckets, each filled with the necessary substrate for oyster mushroom growth. To monitor the development of the mushrooms, three cameras were strategically positioned within each tent, capturing images at regular intervals. These images were then stored in a cloud-based system and made accessible through a dedicated website, providing real-time updates on mushroom growth.

To enhance the monitoring process further, a deep learning model named "Mask R-CNN" was employed. This model was trained to detect the maturity levels of the oyster mushrooms by analyzing the captured images. Through extensive data collection and annotation efforts, we amassed a dataset of approximately 34,400 images to train and fine-tune the Mask R-CNN model. This enabled us to determine when the mushrooms were ready for harvest, avoiding both premature and overdue harvesting.

We also developed two semi-automated harvesting systems (pneumatic and motorized) to streamline the harvesting process. Throughout this report, we will provide a detailed account of the entire project, covering the setup of the production line, the implementation of the Mask R-CNN model, and the functioning of the semi-automated harvesting systems.

Additionally, we will discuss the results obtained from our experiments and highlight potential avenues for further improvements and future work. By leveraging advanced digital technologies, we aim to demonstrate the significant benefits of digitization production processes in the realm of oyster mushroom cultivation. The combination of real-time monitoring, intelligent mushroom maturity detection, and a semi-automated harvesting system holds the potential to revolutionize the industry, paving the way for increased efficiency, productivity, and profitability.

1.1 Background

Gourmet mushrooms like Shiitake, Oyster and Enoki are either harvested in the wild or grown indoors in controlled environments. Harvesting mushrooms outside is a seasonal activity and thus limited to a few months per year. Moreover, it is a labour-intensive process, and the changing outdoor conditions result in volatile harvests. Furthermore, climate change further limits outdoor mushroom growing- and harvesting opportunities [2] [3]. Indoor mushroom farms with controlled growth environments allow for an all-year-round growing and harvesting of mushrooms in sensor-controlled grow rooms and grow tents. However, it remains a labour-intensive process that requires skilled workers [4].

1.2 Objectives

MushR aims to introduce a modular and scalable gourmet mushroom growing and harvesting system that extends the state of the art – which only monitors and controls the growing environment – by introducing a novel image recognition system based on Mask R CNN that detects when and which mushrooms are ready to be harvested in combination with an automated mushroom harvesting mechanism, which will be added in follow-up projects, for harvesting the mushrooms without human interaction.

The modularity and scalability of the system allow for industry-level usage where the VEDLIoT edge devices are placed at the growing facilities and can be scaled according to the required mushroom growing systems within the facility. As a result, MushR drastically reduces the necessity of manual labour for gourmet mushroom growing/harvesting and allows for further industrial-scale automation and increased yields and quality of mushrooms.

In this project, we focused our study to a specific species of gourmet mushroom: *Pleutorus Ostreatus [5]*.

1.3 Significance

The project holds significant importance in the field of mushroom cultivation and automation. By employing computer vision and machine learning techniques, we aimed to provide growers with a reliable and efficient tool to determine the optimal harvest time. This would lead to higher yields, reduced labor costs, and improved overall quality of the harvested mushrooms.

2 Methodology

In this section, we will outline the methodology employed in the project, including the setup of the mushroom growth environment in 2.1, development of reusable mushroom pods in 2.2, digital twins for gourmet mushrooms in 2.3, machine learning model for oyster mushroom maturity detection in 2.4; including data collection, data pre-processing, annotation, model training, and finally the automated harvesting systems in 2.5.

Throughout the project duration, we received considerable help and guidance from VEDLIoT partners *Christmann Informationstechnik + Medien GmbH & Co. KG (CHR)* and *EMBEDL AB (EmbeDL).* Christmann Informationstechnik provided us with a VEDLIoT µ.RECS (micro-RECS) evaluation board which allowed us to perform early prototyping, including testing our camera setup; described in 2.1, and also compatibiliy of our digitaltwin implementation and harvesting mechanisms. With regards to the developed machine learning models, we received early consulting support from EmbeDL with regards to the forming the methodologies for the automated creation of the raw image datasets and creation of our annotation workflows for our machine learning model described in 2.4. Christmann Informationstechnik also gave us access to their cloud-hosted GPU training environment for hardware accelerated training, without which we would not have been able to train our machine learning model.

2.1 Grow Chamber Setup & Sensor Integration

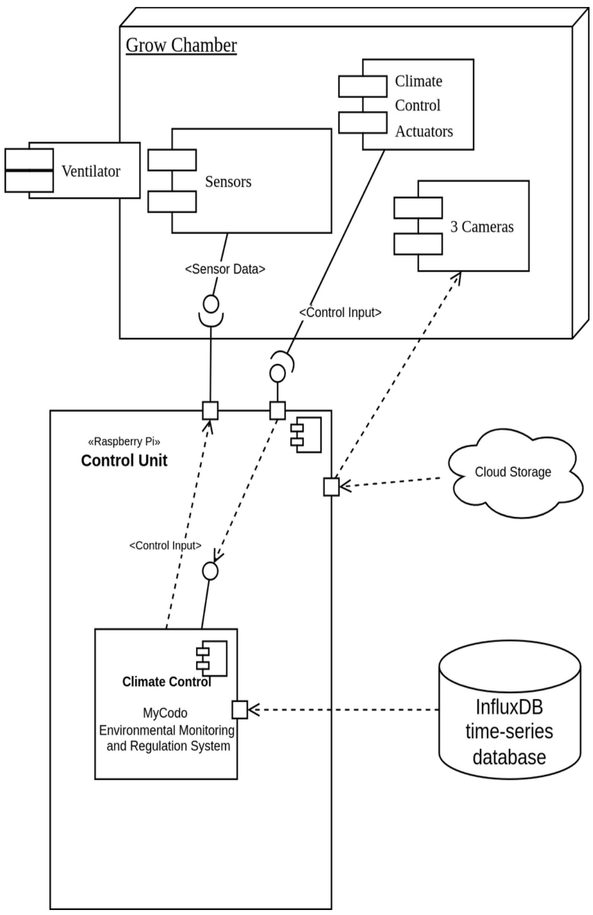


Figure 1: MushR Grow Chamber Setup and Sensor Integration

Figure 1 shows the experimental setup used for integrating sensors into our grow chambers. While this figure only shows the components used for one grow chamber, we scaled up our production environment to two grow chambers (which in our case shared the Ventilator and Raspberry Pi Control Unit) to ensure that we were able to capture adequate data (especially images) of Oyster mushrooms. This setup is shown in Figure 2. Inside each tent, a rack was installed to hold the square plastic buckets containing the mushrooms. A ventilator is positioned outside the tents, which continuously vents excess humidity from inside the tents to outside the building through ducts, firstly to prevent humid air from leaking into the interior of the building, and secondly to facilitate fresh air to circulate into the chambers.

The humidification system is composed of a 60L water tank (for each tent) that further contains a floating ultrasound humidifier, a fan mounted on it's lid, and a UV light. The UV light is positioned inside the tank so that the light can penetrate the water and kill any harmful bacteria or organisms that may be present. The fan attached to the lid, blows air into the tank when the humidifier is active, which forces the humidified air to flow out of the tank and into the tents through a duct.

We used DHT22 sensors for detecting temperature and humidity in the grow chambers. These readings are used as inputs to MyCodo [6], an environmental monitoring and regulations system that runs on the Raspberry Pi. This system also uses these sensor inputs to control the climate control actuators, namely a heater and a humidification system. For optimal mushroom growth, the system keeps the temperature close to 20°C, and between 85% and 90% relative humidity.

Three Raspberry Pi HQ Camera [7] modules (per tent) are used to capture images from inside the tents. These cameras are each additionally fitted with a 120° wide angle lens, and an mounted inside an acrylic dome case using custom 3D printed mounts for adequate waterproofing.

We have published all the source code used for the sensor integration and automated image capture, including the 3D printable mounts on GitHub [8].

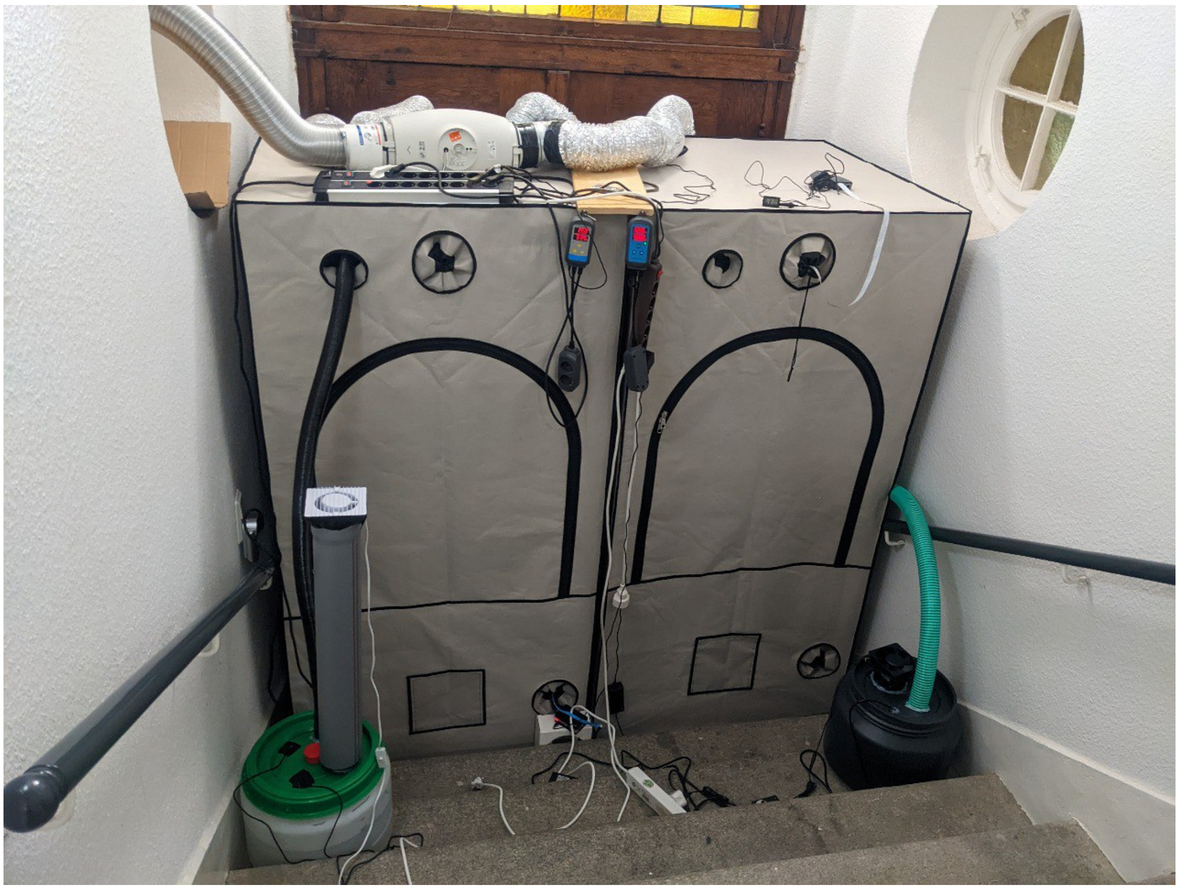


Figure 2: Pair of tents setup identically as grow chambers for Gourmet Mushrooms

2.2 Reusable Mushroom Pods

Gourmet mushrooms are commonly grown on a substrate in one-time-use plastic bags, which are cut open to start the fruiting period of the mushroom. While some mushroom growing bags claim to be bio-degradable, it is not clear whether they can really be decomposed or if the plastic particles become too small to be visible to the human eye.

Aiming to reduce, if not eliminate, the plastic waste created using one-time-use plastic (polypropylene) bags as substrate containers, we experimented with an alternative approach: reusable plastic buckets.

In our experimental setup, we use plastic (polypropylene) buckets; which, weighing in at 90g, can contain up to 3L of substrate. We drill 3.5cm holes on five sides of the bucket, which function as fruiting and ventilation holes. These holes are sealed with micro-porous tape during the incubation phase, which is removed for fruiting. Unlike one-time-use plastic bags, the plastic buckets, which we call mushroom pods, are never permanently damaged during the lifetime of the substrate contained therein.

We compute the environmental benefit of using reusable mushroom pods (in terms of materials) over one-time-use plastic bags by running life-cycle impact assessment calculations using OpenLCA [9]. To this end, we define the reference flow and functional unit for both container types in terms of “amount of polypropylene (g)” and “colonizable volume of the container in liters (L)”. We further quantify the relationship between the defined reference flow and the functional unit for both container types.

Through experimentation we estimated the colonizable volume of a 5L one-time-use plastic bag weighing 30g to be 3L (on average). This decrease is because the bags do not have a built-in method to seal the contained substrate. A substantial amount of the bag is used for this purpose, accomplished by folding the opening over itself several times. Unlike the bags, our mushroom pods are equipped with a sealable lid. As a consequence, the estimated colonizable volume of our 3L mushroom pods is hence, still 3L.

Using these calculations, we create a reference process for both these container types in OpenLCA. We sourced the Life-cycle Inventory (LCI) data pertaining to the manufacturing processes of polypropylene bags and buckets from the Agribalyse dataset [10]. Table 1 shows the Life-cycle impact assessment (LCIA) results computed using the BEES+ impact assessment method.

Table 1: Life-cycle Impact Assessment comparison for "Mushroom Substrate Bag/MushR Reusable Pods (Colonizable Volume)"

Impact	Reference unit	Mushroom	MushR Reusable Pods

category		Substrate Bag	
Acidification	H+ mmole eq	11.21	40.15
Eco-toxicity	g 2,4-D eq	2.07	0.72
Eutrophication	g N eq	0.43	0.34
Global warming	g CO2 eq	102.46	217.20
Habitat alteration	T&E count	1.88 E-16	-7.03 E-17
HH cancer	g C6H6 eq	0.15	0.48
HH criteria air pollutants	microDALYs	0.004	0.015
HH noncancer	g C7H7 eq	423.69	1124.07
Indoor air quality	g TVOC eq	0	0
Natural resource depletion	MJ surplus	0.26	0.92
Ozone depletion	g CFC-11 eq	2.64E-07	1.49E-06
Smog	g NOx eq	0.14	0.47
Water intake	liters	0.95	2.3

From Table 1 we can infer the benefit of our reusable mushroom substrate pods over the non-reusable substrate bags, with respect to the impact categories. Using “Global Warming” as an example, a 5L (3L usable) non-reusable bag has an impact of 102.46 g

CO_2 eq, compared to a 3L reusable pod with 217.20 g CO_2 eq. From this we can infer that one reusable pod will have less impact in this category than three non-reusable bags. This means that simply reusing our mushroom pods 3 times will result in them having a lower impact in this category than the non-reusable counterparts. Similar comparisons can be made with other impact categories. The full LCA calculations and results can be found in our GitHub repository [8].

The above figures indicate that the reusable Mushroom pods will have a lower environmental impact over time. These figures however are highly dependent on the specific materials used for producing the containers. These figures also might not scale linearly with the size of the container for mass-production of mushrooms.

2.3 Digital Twins for Gourmet Mushrooms

In the food industry, digital twins can provide better food quality, predictive maintenance, energy-use minimization, and higher transparency of the production processes [11]. The MushR digital twin implementation models various stages in the mushroom production process, including substrate and spawn production, inoculation, fruiting and harvesting.

2.3.1 Digital Twin Database Schema

The digital twin models the relations between various assets using a graph data-structure stored in Neo4j [12]. Figure 3 and Figure 4 give class diagrams of the defined nodes and relations.

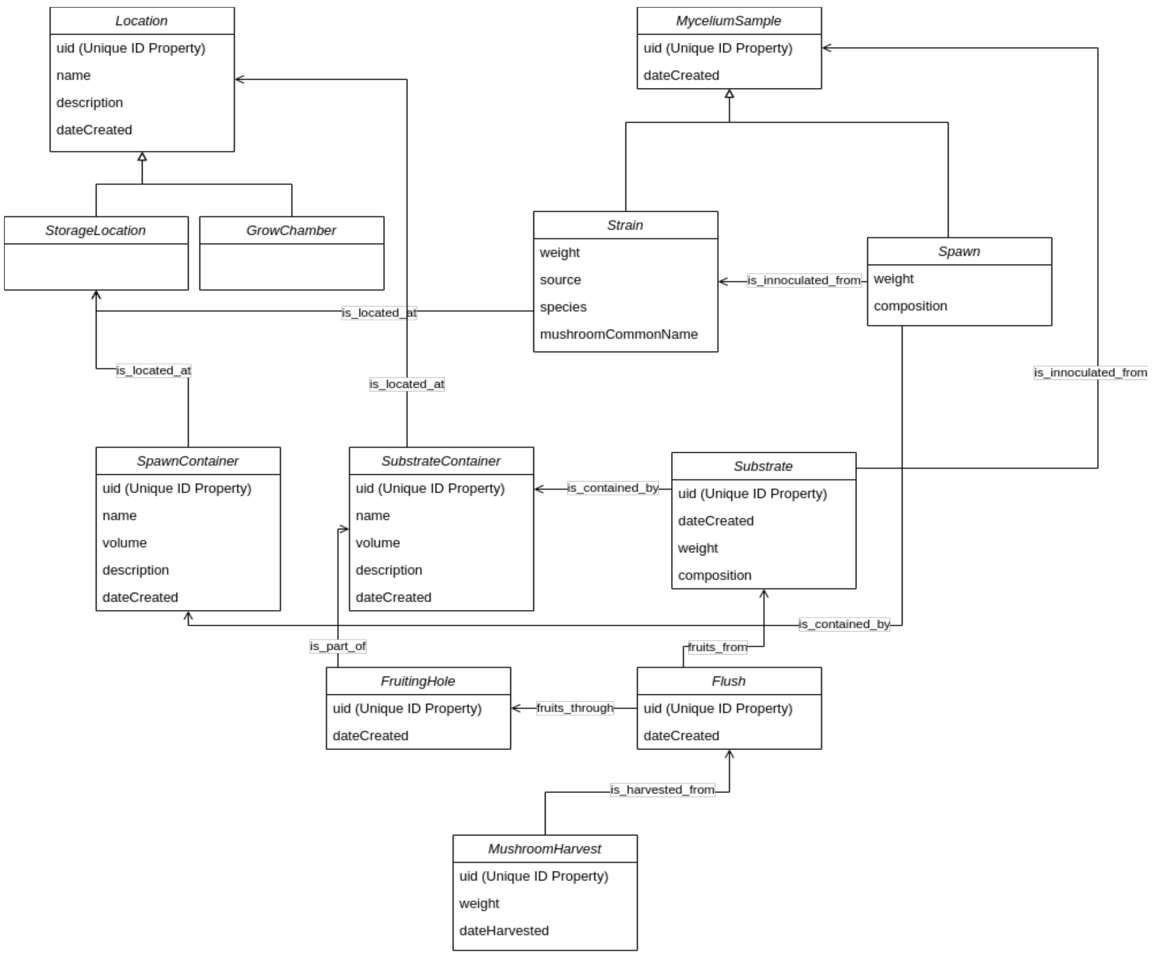


Figure 3: Class Diagram of Nodes defined for the MushR Digital Twin

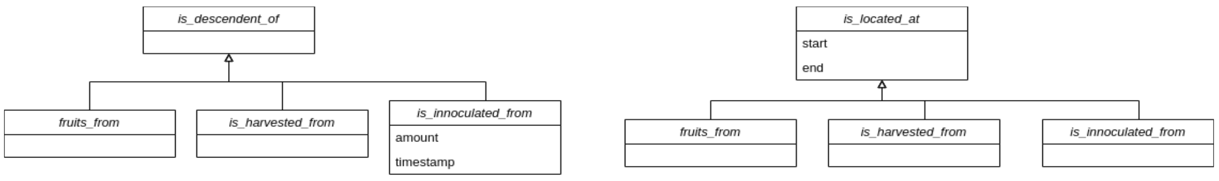


Figure 4: Class Diagram of Relations defined for the MushR Digital Twin

Neo4j is incapable of enforcing rigid class hierarchy or data-types of their attributes of nodes or relations. For this reason, we implement the class hierarchy using the Neomodel Object Graph Mapper (OGM) [13] for Neo4j, which is then exposed using a REST-API, the MushR DigitalTwin API. All interactions with the data, including the creation of new assets, are handled through this API. The defined schema allows retrieval of the state of the network at any previous instance in time. For example, the graph stores the state of a substrate container and contains information on where it is located, which can be either a storage location or a grow chamber. The relation between an instance of a substrate, with its substrate container implicitly gives its location history. Substrates are inoculated using samples of existing mycelium spawn or newly procured mushroom strains.

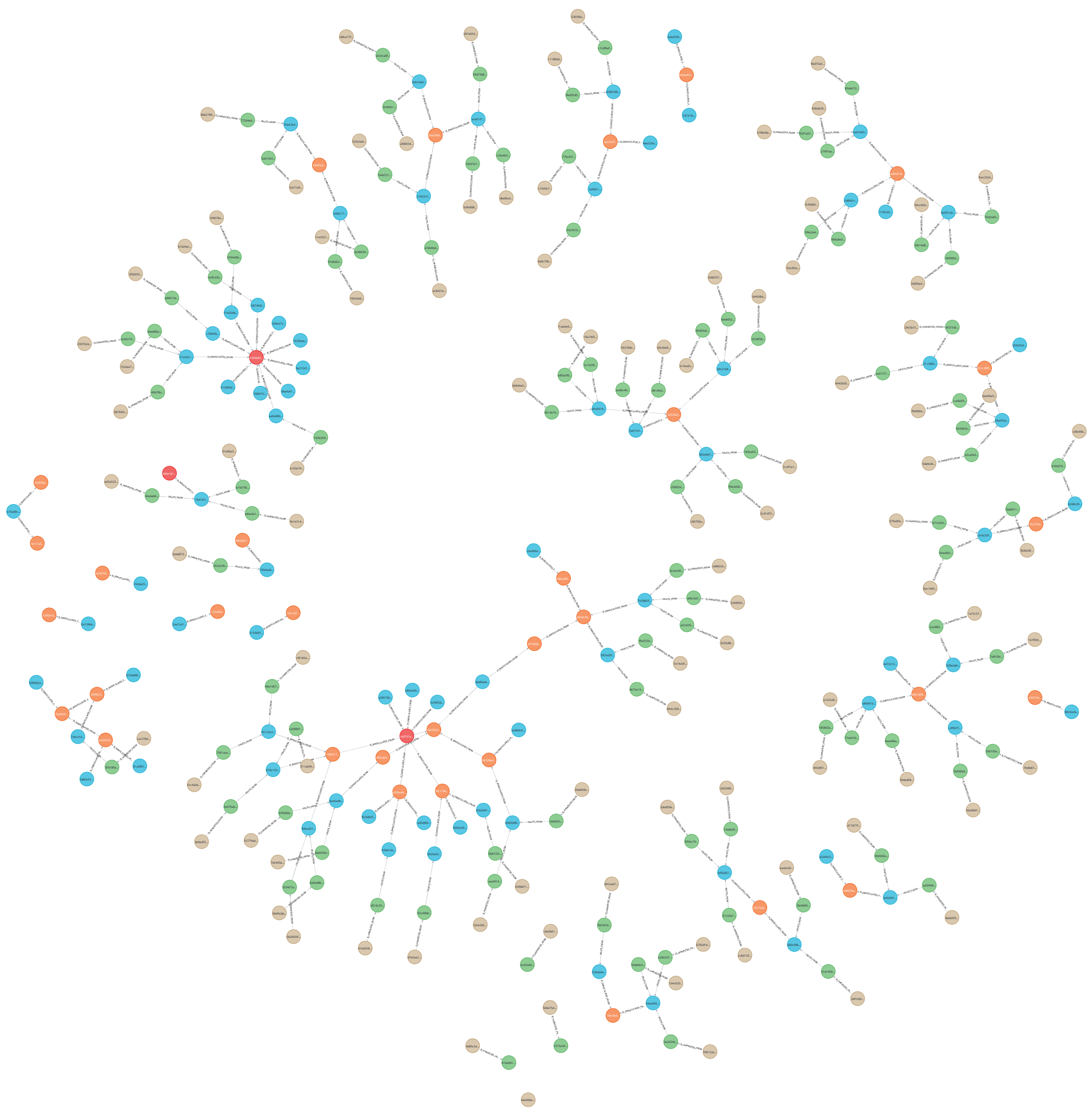


Figure 5: Digital Twin Graph Visualization showing provenance of mushrooms (brown) harvested

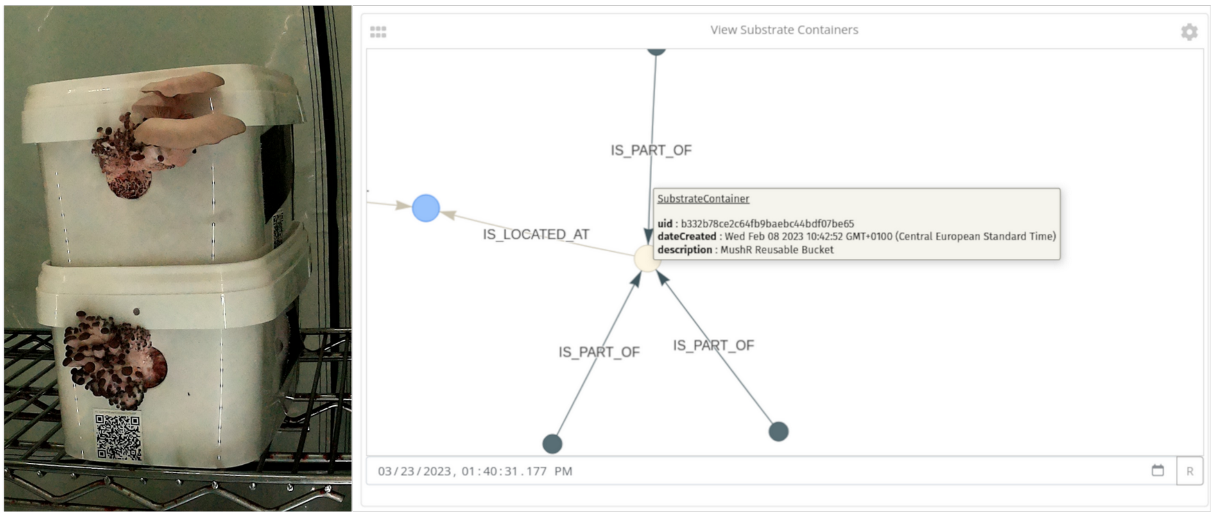
The provenance of a specific mushroom harvest can be traced using the digital twin by recursively traversing the relations. Figure 5 shows a graph visualization of provenance based on data recorded over eight months in 2022. The data is restricted to 300 nodes and relations pertaining to inoculation, fruiting and harvests. The digital twin database also stores location-related information so that the location of any of the assets can be accessed at any instance of time.

2.3.2 Digital Twin User Interface (UI)

2.3.2.1 *MyCodo Widgets*

We extended the functionality of MyCodo which we to integrate our grow chamber with sensors, by adding widgets to interact with the MushR Digital Twin. Furthermore, we publish our extension implementations with an open-source license, available on our GitHub repository [8].

We designed a collection of widgets for the MyCodo, Figure 6 shows a substrate container, it's fruiting holes, and it's location at a particular instance in time. Similarly the state of a various assets involved in the mushroom production process can be accessed with the graph visualization, for any instance in time.



SubstrateContainer

uid : b332b78ce2c64fb9baebc44bdf07be65
dateCreated : Wed Feb 08 2023 10:42:52 GMT+0100 (Central European Standard Time)
description : MushR Reusable Bucket

Figure 6: Left: MushR Reusable Pod, Right: Corresponding Digital Twin representation

The UI also includes further widgets such as a QR Code scanner, that allows a user to scan the QR Codes of the pods and their fruiting holes, making the processes of inoculation, fruiting, harvesting, etc easier to log into the digital twin database. For instance, the QR Code on a substrate container refers to the substrate container itself, rather than the substrate that might be currently contained in it. A user can log actions on the substrate, such as inoculation, by simply scanning the QR Code on the substrate container. The digital twin API associates the actions on the substrates based on the association of which substrate is currently located in the container.

2.4 Mask R CNN Machine Learning Model

This section describes the methodology we followed to train a computer-vision machine learning model to detect the maturity of Oyster mushrooms. However, since no such dataset exists to facilitate training, we aimed to create our own.

2.4.1 Dataset

Over a period of 10 months, we captured approximately 34,400 JPEG images, each with a resolution of 3040x4056 and all 3 color channels. We positioned three cameras on

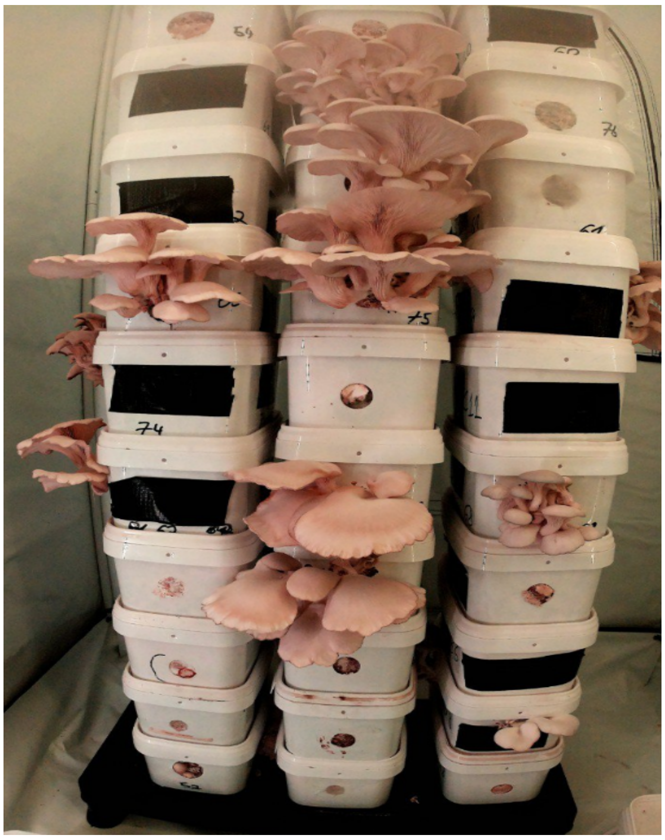


Figure 7: Hourly automated image capture for Oyster mushroom growth monitoring

one side of each or our two grow chambers; one camera in the center and the other two on either sides. We angled the cameras inwards to capture as many fruiting substrate containers as possible within each frame. Substrate containers that appear in the images are generally of three types: 5L and 7L substrate bags, large 5L reusable substrate buckets and 3L MushR reusable pods. While substrate bags cannot be stacked, unlike the other containers, they are always placed on a storage rack. Images were taken automatically at hourly intervals and sent to cloud storage and a website for real-time monitoring. The data collection process provided a comprehensive and diverse dataset for training and validation.

2.4.1.1 Data pre-processing

We cleaned the image data to ensure that the focus of our project, which is detecting the maturity of oyster mushrooms, is maintained. As part of this process, we removed images that are foggy (influenced by the humidification system), have a significant amount of noisy areas or are contaminated. In preparation for annotation, we further de-prioritized continuous image sequences which did not show enough apparent growth, in an effort to preserve the variety of images in our final dataset. Furthermore, since the background of mushrooms growing from substrate bags has significantly more noise than those of the reusable pods, we also prioritized images in which the majority of substrate containers are reusable pods. Our filtered and prioritized subset of images selected for annotation and creation of the training set resulted in only 1,566 images, roughly ~4.5% of the total number of available images.

However, it is important to note that we have preserved the original raw dataset on Kaggle [14] and will keep it freely available. This is done to provide a comprehensive dataset for developers, researchers and enthusiasts in the field of agriculture automation, that now can access and utilize the dataset for further analysis and experimentation. Since the images are not only of buckets and mushrooms, it can enable the development of other algorithms and approaches for mushroom maturity detection. In addition to the raw images, we provide annotations for a subset of the dataset on [8]. Annotations were carefully created to mark the regions of mushrooms within the images. The annotation approach involves manual labeling of mushroom maturity regions based on visual cues, such as the mushrooms' color, shape and size. The annotation provides a valuable resource for training and validating the Mask R-CNN model.

2.4.1.2 Annotation

To prepare the dataset for training, a subset of images were annotated and labeled using CVAT [6]. We labeled mushrooms as "ready," "not ready," or "overdue" based on their maturity level, shown in Figure 8. The annotations were exported to the COCO JSON, containing the bounding box coordinates, masks, and corresponding labels.

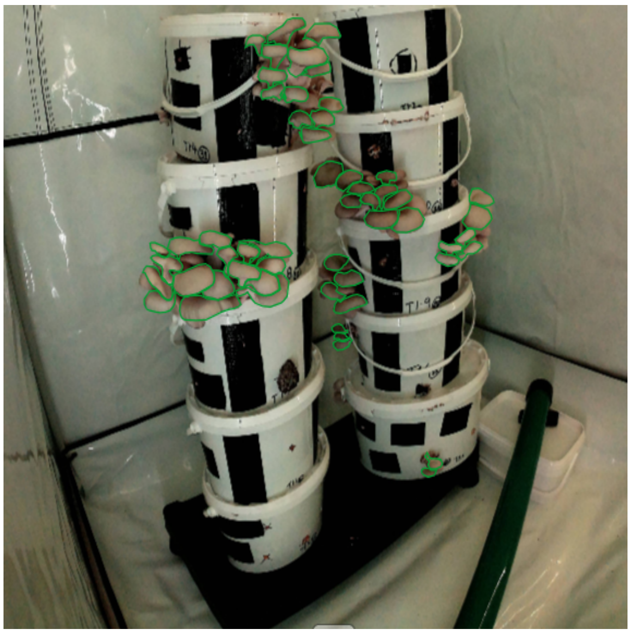


Figure 8: An example annotated image

Table 2: MushR Dataset Overview, number of annotated instances per class

Dataset	Not-Ready	Ready	Overdue	Total
Train	723	1344	692	2759
Test	251	198	541	541

2.4.2 Training

The Mask-R-CNN model was utilized for training the maturity detection algorithm. The annotated dataset was divided into training and test sets. The model trained and fine-tuned through multiple iterations to optimize its performance. The test dataset was used to assess the model's accuracy and make necessary adjustments to the training hyper-parameters.

To train the model, we relied on the Detectron2 [15] library, a popular computer vision framework. It provides a comprehensive set of tools and pre-defined architectures for object detection and instance segmentation tasks. In our research, the Detectron2 library was the foundation for training and fine-tuning the Mask R-CNN model on our custom dataset. It optimized the model’s performance specifically for oyster mushroom maturity detection.

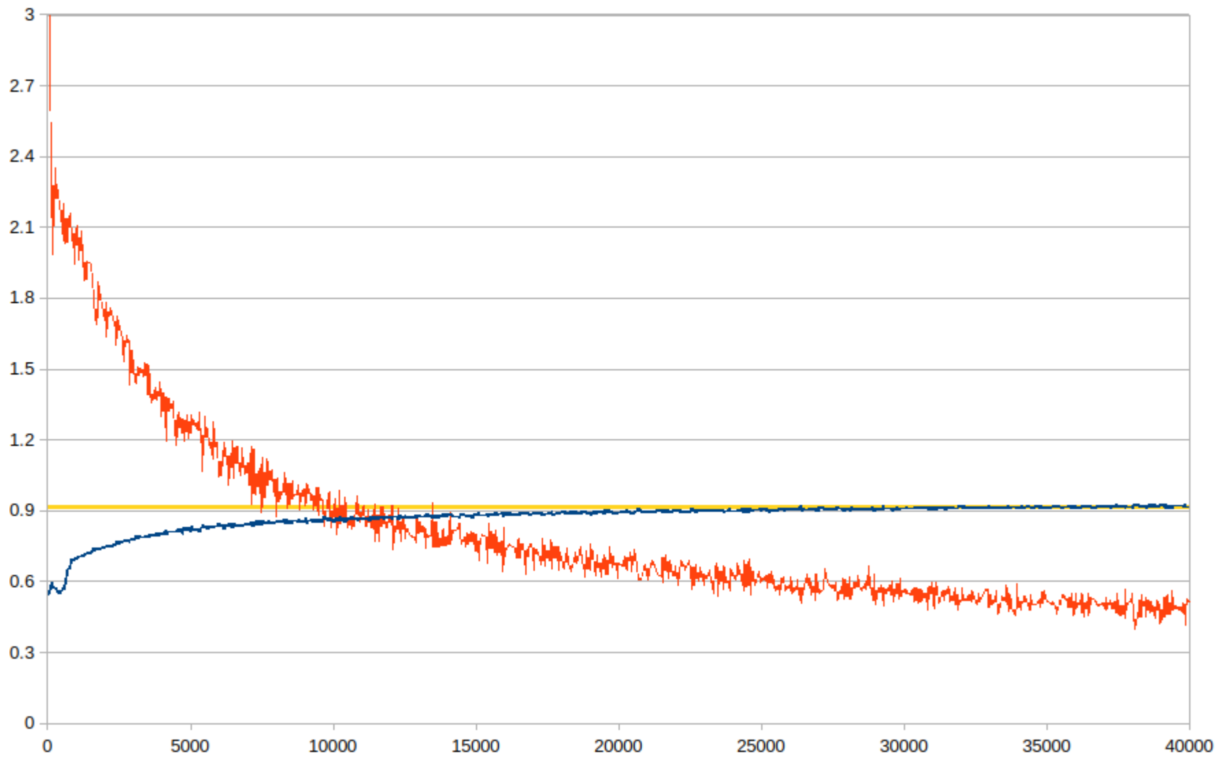


Figure 9: Mask R-CNN Training Curve showing Total Loss and Training Accuracy over training iterations.

We first initialized the model with weights pre-trained from the COCO dataset [16]. We then train the model on our custom training dataset. We adopted the default Detectron2 hyper-parameters [15], and used a base-learning rate (post-warmup) of 0.00015, with a batch size of 2 for a maximum of 40,000 iterations. The model trains on an Nvidia GTX 4090 using 2.7GB of memory at ten iterations per second, achieving a training accuracy of 91.72%.

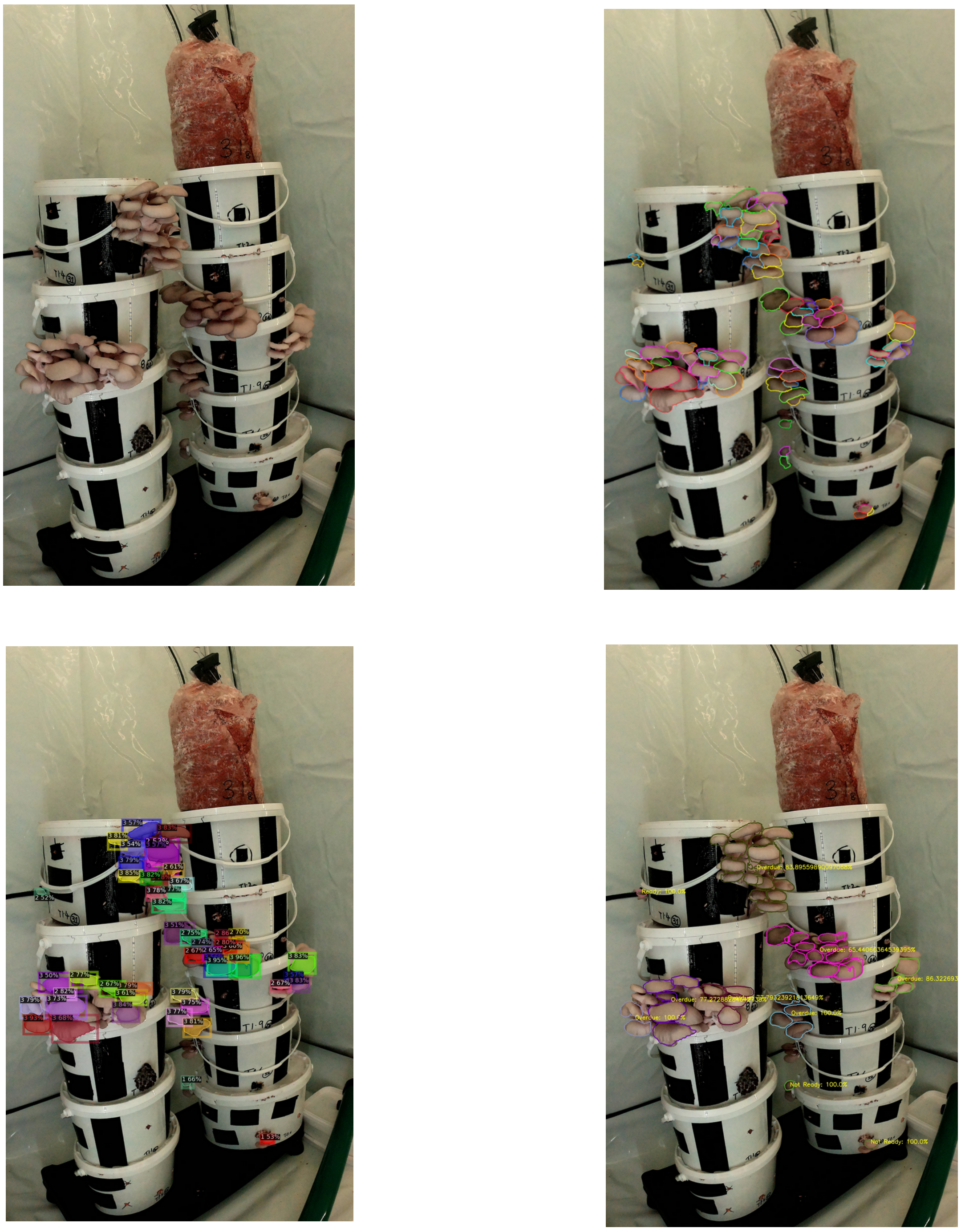
2.4.3 Clustering

Although the trained Mask R-CNN model segments and classifies individual mushrooms with decent accuracy; in mushroom production, entire flushes are always harvested together rather than individual mushrooms. Therefore, an additional step is required to cluster the predicted masks and obtain a final prediction for the entire flush. This clustering process helps to consolidate the individual mushroom predictions into a cohesive prediction for the entire harvest, ensuring efficient and accurate harvesting in mushroom production.

We use the DBSCAN [17] algorithm to cluster the predicted masks, computing the center of the masks to be used as the proxy. We use $\varepsilon = 2.5 \times 2 \times r$, where r is the average (approximate) radius of all the mask instance predictions in the input image. This value defines the neighborhood of a mask center-point, and any masks that do not center within this neighborhood are not considered part of the same flush. Figure 4 shows example results of our maturity prediction workflow performed on a single image. In addition to this clustering, we compute each cluster's total area of the instance classes, Not-Ready, Ready and Overdue. This computation improves the accuracy of the final maturity state prediction for the cluster over using a simple instance count. Figure 10 gives an overview of an example maturity prediction on a single image.

More information, such as the complete training configuration, train and test datasets used, and code used for clustering and visualization can be found in our GitHub Repository [8].

Figure 10: Example of oyster mushroom maturity prediction on a single image showing; a) Raw image used for training, b) Annotated ground-truth instances, c) Mask R-CNN inference results on trained model, with each instance labeled with the predicted class; Not-Ready(1)/Ready(2)/Overdue(3) and class probability (%), d) Final harvesting decision by clustering predicted instance masks, including class probability (%)



2.5 Automated Harvesting

In pursuit of the goal of a prototype harvesting system for gourmet, we followed two possible approaches: pneumatic and motorized harvesting, which are detailed in this section.

2.5.1 Pneumatic Harvesting

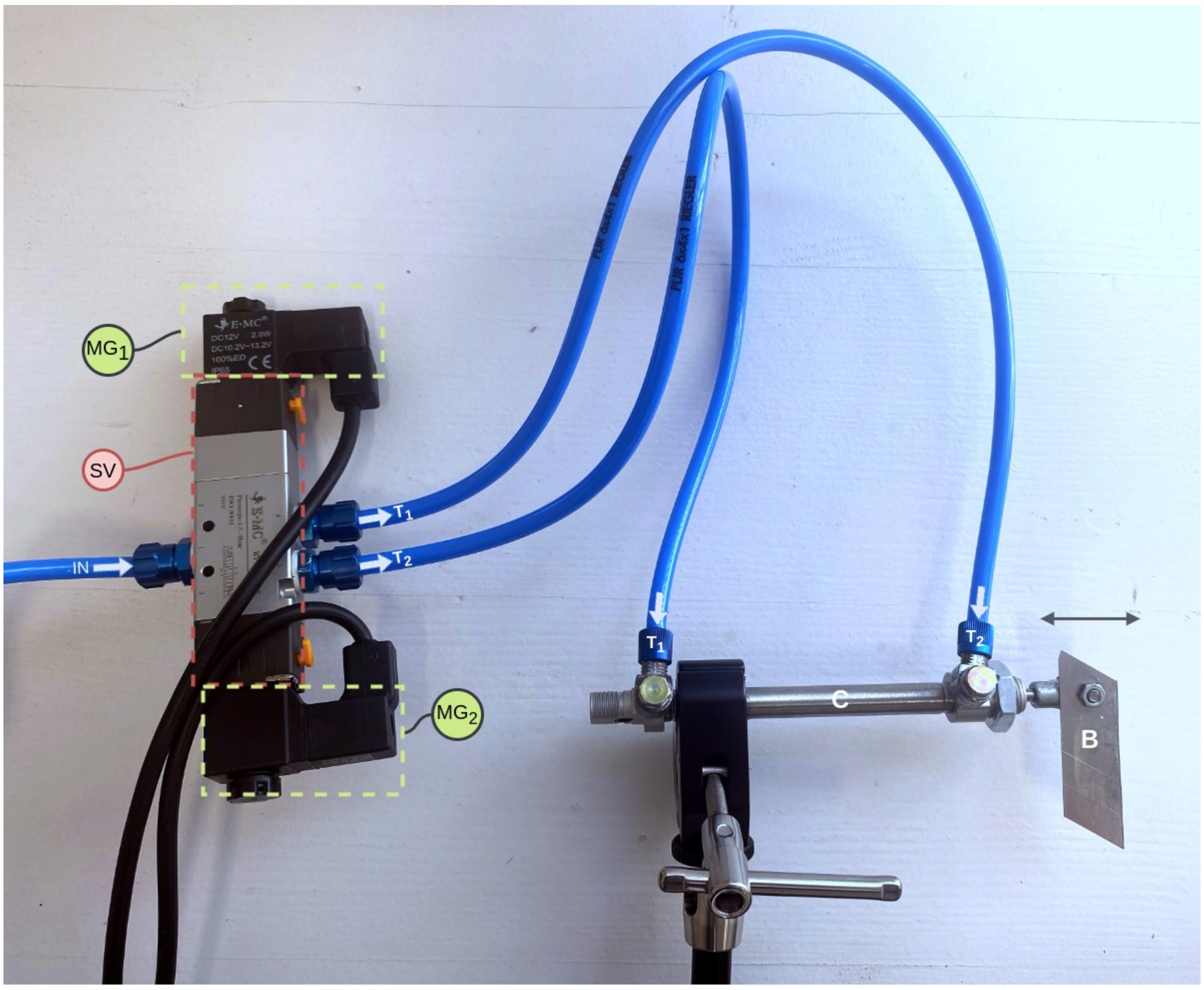


Figure 11: Pneumatic Harvesting System Components

Figure 11 shows the components used to prototype the pneumatic harvesting system. This system actuates a blade (B) which is mounted perpendicularly on a bi-directional piston rod cylinder (C). Figure 12 shows how the cylinder is set up to harvest oyster mushrooms growing from our developed mushroom pods. The pneumatic tubes T_1 and T_2 control the motion of the pneumatic cylinder. Depending on whether T_1 or T_2 is pressurized, the piston rod moves outwards or inwards from the cylinder, respectively. When both T_1 and T_2 are pressurized or depressurized, the piston is immobilized or free to move, respectively. For the purpose of harvesting, T_1 and T_2 must be pressurized and depressurized alternatively. This is accomplished by the 5/3-way solenoid valve (SV). Internally, two solenoids control the valves pressurizing T_1 and T_2 . These solenoids are activated by 12-volt DC magnetic coils MG_1 and MG_2 . These coils are digitally controlled by 5-volt relay switches connected to a Raspberry Pi 4B, programmatically controlling the entire setup. More information about our experimental setup can be found on our GitHub repository [8].

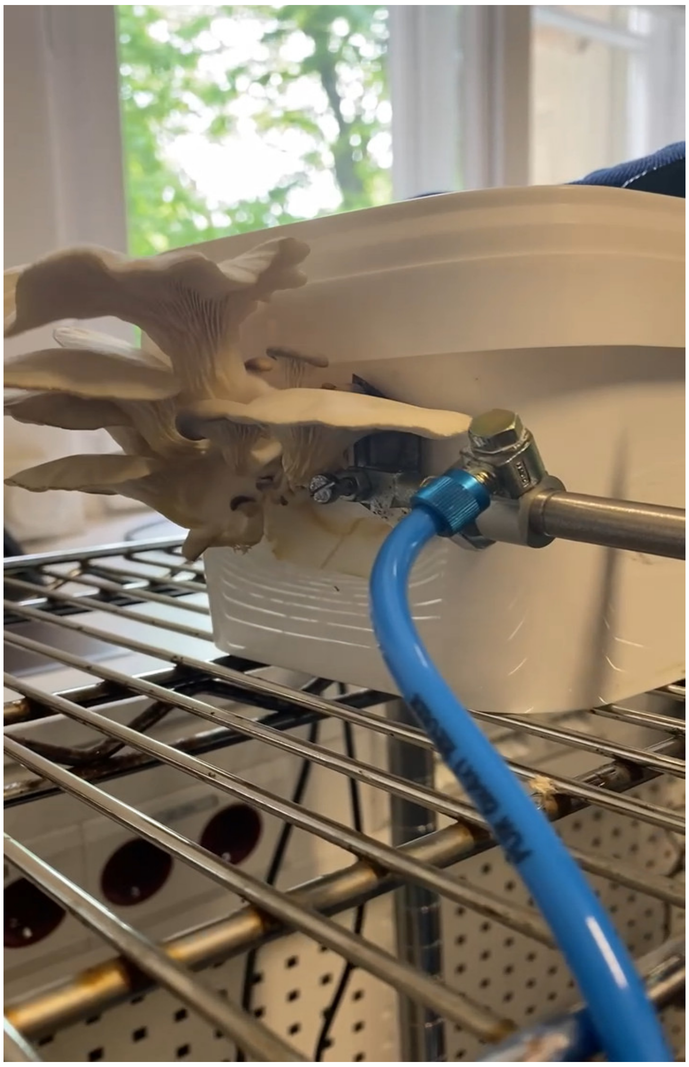


Figure 12: Pneumatic Harvesting System

2.5.2 Motorized Harvesting

Figure 13 shows the major components of the developed motorized harvesting system. It consists of a stainless steel cylinder (C) with drilled-in fruiting holes. It is suspended from the ceiling of an aluminum platform that is held together by machined aluminum columns. The cylinder contains a plastic bag filled with a substrate for oyster mushroom growth. The holes in the cylinder aligned with the perforation in the plastic bag, allowing the mushrooms to emerge and grow. A 3D-printed lid (L), matching the diameter of the cylinder, seals the bottom to prevent substrate spillage in case of damage to the bag. A stepper motor (S) is installed underneath the suspended cylinder to drive a 3D-printed plate, which vertically holds the blade (B).

The plate is specifically designed to accommodate the motor, ensuring stability during rotation. The stepper motor initiates rotation, causing the blade to cut the mushrooms protruding through the holes in the cylinder. We program the motor to rotate in either direction to ensure successful harvesting. Adjustments such as sharpening the knife and modifying the motor's speed can be made to optimize the system's performance.

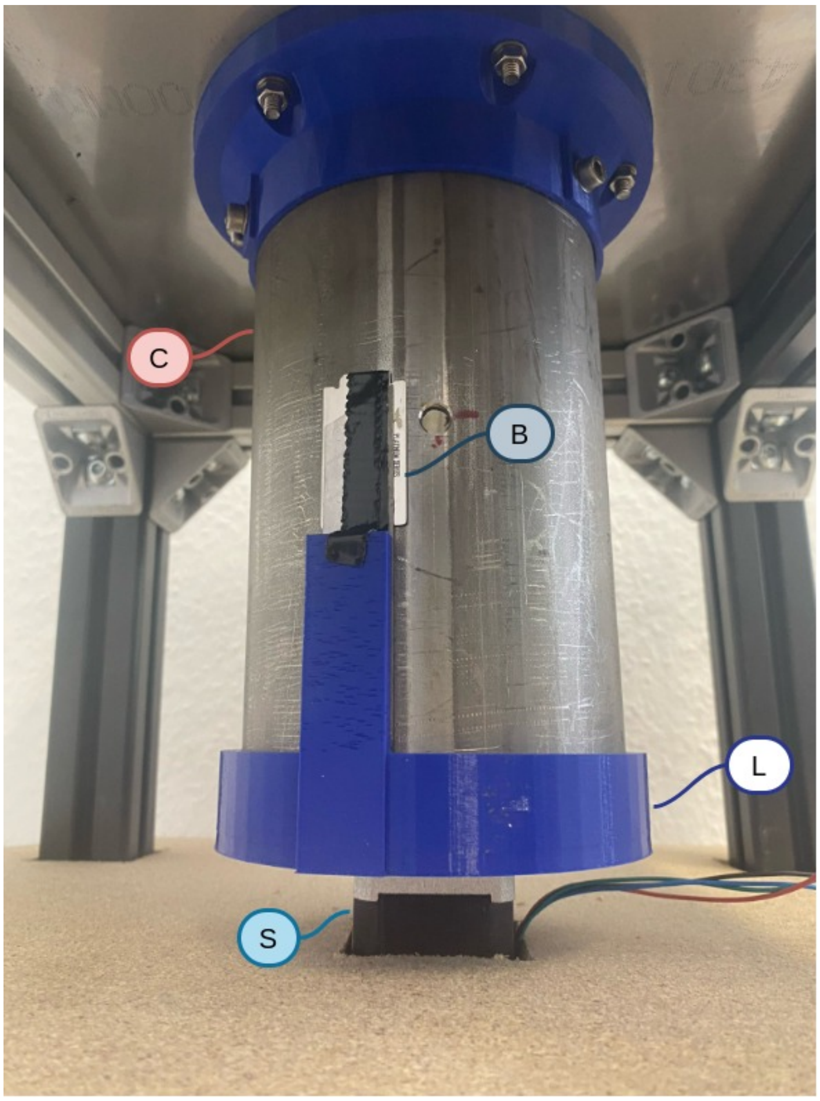


Figure 13: Motorized Harvesting System

3 Results and Findings

In this section, we will present the results and findings obtained from the project. This includes the performance of the Mask-R-CNN model, the effectiveness of the maturity detection system, and the impact on the harvesting process.

3.1 Mask R CNN Model Performance Evaluation

Table 3 and Table 4 evaluate the Mask R CNN model described in 2.4.2. We evaluated the trained model in terms of Average Precision (AP) on our test set using the COCO Evaluation Metric [16]. AP of the model for a specific class shown in Table 4, is defined as the area under the precision-recall curve, computed by performing inference over the test dataset. The AP of the bounding boxes are expected to be generally larger than those of segmentation, which is also true for our model. For a more holistic understanding of the model's performance on the test dataset, Table 3 gives the AP averaged over all classes, and computed for different Intersection Over Union (IOU) thresholds. For instance, $AP^{IOU=0.50}$ averages the AP over all classes, where each AP is computed by considering a proposed Region of Interest (ROI) to be a true positive only if it overlaps with the ground truth by at least 50% of their unionized area. AP is also computed across different instance sizes (in terms of pixel area), where only instances in a specific size range are considered for computing the average. A trend can be seen in Table 3 with AP across scales: AP^{small} is significantly lower than AP^{medium} and AP^{large} . Since mushroom fruiting bodies that are "Not-Ready" are usually smaller in size, this explains the decrease of AP for the “Not-Ready” class in comparison with the other classes.

Due to time-constraints, we were only able to utilize a small percentage of the total available images for the training and testing datasets. While, this leaves some room for improvement, our practical tests showed the model performing very well at correctly predicting the maturity of a single mushroom when it's closer to the point of being ready for harvest, due to it's relative size being larger. Regardless of whether the majority predicted class of an entire flush of mushrooms is “Ready” or “Overdue”, the decision made in this case is always to harvest. Due to this, combining the Mask R-CNN model's predicted instances with the clustering approach further improves the model's usability towards the actual goal of knowing if a flush (or cluster) of mushrooms fruiting from a fruiting hole are ready to harvest.

Table 3: Mask R CNN model evaluation using the COCO Evaluation Metric

Criteria	AP	$AP^{IOU=0.50}$	$AP^{IOU=0.75}$	AP^{small}	AP^{medium}	AP^{large}
Bounding Box	61.876	71.016	67.920	20.792	69.788	87.207
Segmentation	49.332	70.639	60.812	9.010	54.799	80.218

Table 4: Mask R CNN model evaluation (per-class)

Task	Class	AP
Bounding Box	Not-Ready	34.386
	Ready	74.683
	Overdue	76.558
Segmentation	Not-Ready	26.502
	Ready	61.083
	Overdue	60.410

3.2 Automated Harvesting Systems

2.5.1 and 2.5.2 describe our prototype pneumatic and motorized harvesting systems respectively. While they are capable of harvesting full flushes of mushrooms, the time required to harvest a single flush increases heavily on the water content and thickness of the flushes at the base of the fruiting holes. From our limited testing, the pneumatic harvesting mechanism setup, as shown in Figure 12, failed to produce any significant results below 2.1 bar of pressure. At 7.5 bar, we could reliably harvest flushes, albeit requiring up to 5 seconds of continuous application to successfully harvest the thickest fruiting flushes. On the other hand, since the fruiting holes in the motorized harvesting system are designed to be significantly smaller, fruiting flushes can be harvested nearly instantaneously, since it requires lesser force. The disadvantage of using a smaller fruiting hole, is a decrease in the fruiting speed and yield quantity of the flushes. However, this does not diminish it's potential for use in a smaller scale system. Our pneumatic system is unlikely to be very practical on a small-scale. The motorized system being dedicated to the substrate container, can be adapted to a small, integrated growth and harvesting system. This makes it the better contender for a small-scale, low throughput indoor mushroom production system.

Unlike the motorized harvesting mechanism, the pneumatic harvesting system is an independent modular component. In an industrial-grade setup, since our mushroom pods have flat surfaces, they can be moved from their growth chambers when the mushrooms are ready to harvest (as detected by our Mask R-CNN based maturity detection method) to the harvesting system via conveyor-belts (or other industrial mobility solutions). The number of mushroom pods being used in production can hence increase faster than the number of harvesting systems; the rate of the increase being dependent on the desired throughput of the production environment, thereby allowing the system to be very scalable.

4 Discussion

In this section, we will discuss the key contributions and implications of the project. We will also address any limitations encountered during the implementation and suggest potential areas for further improvement.

4.1 Key Contributions

The MushR project makes an important preliminary step towards developing a system capable of complete automation of gourmet mushroom production. Following are it's key contributions.

1. A digital twin for gourmet mushroom production, which is capable of storing the temporal state of any of the important assets, such as substrate, spawn and harvests involved in the mushroom production process, along with a visual user interface to interact with the graph-based structure of the digital twin representation.
2. A Mask R-CNN based model for the detection of oyster mushroom maturity.
3. Two semi-automated proof-of-concept prototypes for harvesting gourmet mushrooms:
    1. Pneumatic harvesting
    2. Motorized harvesting
4. An analysis of the environmental benefit of using reusable mushroom pods in favour of one-time-use-only plastic bags. In our use case, reusing the mushroom pods three times yields a more eco-friendly output than previous approaches relying on plastic bags.

4.2 Limitations

While the project achieved significant success, some limitations were encountered. Firstly, the digital twin implementation described in 2.3 did not use any process modeling to monitor or simulate the various production process involved in gourmet mushroom production.

While our Mask R-CNN model's performance is promising, there are certain limitations to consider. Factors such as lighting conditions, humidity, oyster mushroom variations, and occlusions affect the model's accuracy.

As mentioned in 2.4.1, we accumulated 34,400 images of oyster mushrooms. However, due to the immensely time-consuming endeavor of annotating the mushrooms with masks, we could only annotate a small portion of those images. This subsequently limits the potential of our Mask R-CNN model to learn to segment these mushrooms.

Lastly, our developed methods have yet to be tested in an industrial gourmet mushroom production environment, and the results presented in this paper have the possibility of being only applicable to our experimental setup for oyster mushroom production.

5 Conclusion

In this section, we will provide a summary of the project and its achievements, emphasizing the significance and benefits of the developed maturity detection system for oyster mushroom cultivation. The key accomplishments of the project include:

Development of a gourmet mushroom production environment with controlled temperature, humidity, and real-time monitoring capabilities.

Collection of a substantial dataset consisting of approximately 34,400 images of oyster mushrooms at different maturity stages.

Training of the Mask-R-CNN based model to accurately detect the maturity status of mushrooms, enabling real-time monitoring and decision-making.

Two prototype harvesting mechanisms, automating the cutting process based on the detection results.

This project has significant implications for the mushroom cultivation industry. The maturity detection system reduces the reliance on manual observation, minimizing human error and subjectivity in determining the optimal harvest time. The automation of the harvesting process has the potential to significantly improve productivity, reduces labor cost, and ensure precise harvesting, resulting in higher-quality mushroom production.

6 Acknowledgments

We would like to express our sincere gratitude to Dominique Fabio Briechle for his guidance, expertise, and support throughout the duration of this project.

We would also like to thank VEDLIoT for funding this project, providing the necessary resources and infrastructure. Without their support, this project would not have been possible.

We would also like to thank VEDLIoT Partners *EMBEDL AB (EmbeDL)* and *Christmann Informationstechnik + Medien GmbH & Co. KG (CHR)* for giving us access to various Nvidia hardware, and also guidance and support, without which developing our machine learning models would have not been possible.

Lastly, we extend our gratitude to all the team members Harish Gundelli, Theresa Sommer, Sepideh Sayadkouh, Johannes Mayer and Athira Mavoomkuttathil Sivachandran involved in the project for their hard work, dedication, and collaboration in achieving the project's goals.

7 Future Directions

In this section, we will discuss potential future directions and areas of further exploration that can build upon the achievements of this project.

7.1 Improvements to the AI Model

To address the identified limitations of the Mask R-CNN model, the following areas can be considered for further improvement:

- Increase the diversity and size of the training dataset to improve the model's robustness and accuracy in detecting mushroom maturity.
- Integrate additional sensors and data sources to gather more comprehensive environmental data, such as CO2 levels and nutrient content, for improved mushroom growth monitoring, and maturity detection; incorporating this to the Mask R-CNN based model.
- Expansion of the instance classes for recognizing contaminated substrate or mushrooms.
- Coupling the maturity detection with the control unit of the growth environment to manipulate growth-related parameters (e.g., temperature) so that mushrooms are ready to be harvested on pre-determined days.

7.2 Scaling and Commercialization

To expand the application of the maturity detection system to larger-scale mushroom cultivation operations and facilitate commercialization, the following steps can be taken:

- Conduct feasibility studies to assess the scalability and economic viability of implementing the system in commercial mushroom farms.
- Develop cost-effective solutions for integrating the detection system into existing cultivation setups, taking into account the specific requirements and constraints of different production environments.
- Collaborate with mushroom growers and industry stakeholders to gather feedback, address their specific needs, and ensure the system's practicality and compatibility with existing workflows.

7.3 Optimization of Harvesting Process

To further optimize the harvesting process and maximize efficiency, the following areas can be explored:

- Investigate the integration of robotic systems or automated mechanisms for mushroom harvesting, leveraging the maturity detection system to precisely cut and collect mushrooms.

- Develop intelligent algorithms that consider factors such as mushroom size, weight, and overall crop yield to optimize the harvesting sequence and prioritize the most mature mushrooms for harvesting.

7.4 Extension to other food types

While this project focused on oyster mushrooms, the developed maturity detection system can be extended to other mushroom varieties, and other food types. This would involve collecting specific datasets for each variety and adapting the detection algorithm to account for variations in appearance, growth patterns, and maturity indicators.

By pursuing these future directions, we can continue to advance the field of automated mushroom cultivation and indoor farming, enhance productivity and quality, and contribute to sustainable food production practices.

8 References

[1].....K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask r-cnn,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969. doi: 10.48550/arXiv.1703.06870.

[2].....X. Yang *et al.*, “Climate change effects fruiting of the prize matsutake mushroom in China,” *Fungal Divers.*, vol. 56, pp. 189–198, 2012, doi: 10.1007/s13225-012-0163-z.

[3]..............H. Kauserud, L. C. Stige, J. O. Vik, R. H. Økland, K. Høiland, and N. C. Stenseth, “Mushroom fruiting and climate change,” *Proc. Natl. Acad. Sci.*, vol. 105, no. 10, pp. 3811–3814, 2008, doi: 10.1073/pnas.0709037105.

[4].....M. Huang, L. He, D. Choi, J. Pecchia, and Y. Li, “Picking dynamic analysis for robotic harvesting of Agaricus bisporus mushrooms,” *Comput. Electron. Agric.*, vol. 185, p. 106145, 2021, doi: 10.1016/j.compag.2021.106145.

[5]....C. Sánchez, “Cultivation of Pleurotus ostreatus and other edible mushrooms,” *Appl. Microbiol. Biotechnol.*, vol. 85, pp. 1321–1337, 2010, doi: 10.1007/s00253-009-2343-7.

[6]CVAT.ai Corporation, “Computer Vision Annotation Tool (CVAT).” https://cvat.ai (accessed Jul. 02, 2023).

[7]Raspberry Pi Foundation, “Raspberry Pi High-Quality Camera.” https://www.raspberrypi.com/products/raspberry-pi-high-quality-camera/ (accessed Jul. 02, 2023).

[8]MushR Contributors, “MushR GitHub Repository.” https://github.com/ETCE-LAB/MushR/ (accessed Jul. 02, 2023).

[9].......GreenDelta, “OpenLCA: The open source Life Cycle and Sustainability Assessment software.” openlca.org (accessed Jul. 02, 2023).

[10]. .V. Colomb *et al.*, “AGRIBALYSE®, the French LCI Database for agricultural products: high quality data for producers and environmental labelling,” 2015, doi: 10.1051/ocl/20140047.

[11].....P. Verboven, T. Defraeye, A. K. Datta, and B. Nicolai, “Digital twins of food process operations: the next step for food process models?,” *Curr. Opin. Food Sci.*, vol. 35, pp. 79–87, 2020, doi: 10.1016/j.cofs.2020.03.002.

[12]....................................Neo4j Inc., “Neo4j.” https://neo4j.com/ (accessed Jul. 02, 2023).

[13]Robin Edwards and other contributors, “Neomodel Object Graph Mapper.” https://github.com/neo4j-contrib/neomodel (accessed Jul. 02, 2023).

[14]ETCE Lab, “MushR Project Raw Image Dataset.” https://doi.org/10.34740/KAGGLE/DSV/6062648 (accessed Jul. 02, 2023).

[15]Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, “Detectron2,” 2019. https://github.com/facebookresearch/detectron2 (accessed Jul. 02, 2023).

[16]....................T.-Y. Lin *et al.*, “Microsoft COCO: Common Objects in Context,” *CoRR*, vol. abs/1405.0312, 2014, [Online]. Available: http://arxiv.org/abs/1405.0312

[17]E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, “DBSCAN revisited, revisited: why and how you should (still) use DBSCAN,” *ACM Trans. Database Syst. TODS*, vol. 42, no. 3, pp. 1–21, 2017, doi: 10.1145/3068335.

ICT-56-2020 - Next Generation Internet of Things

Final technical report
Power-Edge-RL
VEDLIoT Open

Document information	
Project website	https://www.vedliot.eu
Dissemination Level	PU
Nature	R
Contractual Deadline	30.06.2023
Author	B. Haucke-Korber, M. Schenke, O. Wallscheid
Contributors	B. Haucke-Korber, M. Schenke, M. Rothmann, O. Wallscheid

The VEDLIoT Project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957197.

Changelog		
Version	**Date**	**Type**
V 1.0	2023-06-29	Finalisation
V 1.1	2023-07-20	Minor updates based on project partner inputs

Table of contents

Executive summary

Data-driven control of intelligent power conversion systems, like electrical drives, can automate and speed up future development processes. Upcoming concepts in this area are based on reinforcement learning (RL), which requires on the one hand to execute the control decision making policy in hard real time (inference) and on the other hand to improve this policy over time by evaluating the stream of measurement data obtained during control plant interactions. The latter does not require to be executed in hard real time, but can be outsourced to edge computing hardware to speed up the learning process. During this VEDLIoT OC project, an edge computing toolchain for distributed inference and learning of an RL-based drive control system was developed, successfully tested in a real-world test bench and is currently reported in scientific papers. Moreover, the mentioned toolchain has been extended to cover different edge computing accelerators, in particular GPUs and FPGAs, which have been empirically tested and compared against standard CPU-based implementations. This OC project therefore successfully demonstrated the feasibility and benefits of a data-driven control process integration into an IoT context.

1 Introduction

Reinforcement learning (RL) is becoming an increasingly popular approach for controlling complex dynamic systems. However, the actual learning process of RL requires extensive computational power, i.e., it is opposed to the real-time requirement of control processes when using cost-sensitive embedded hardware. To overcome this issue, splitting up the learning process and the control policy inference in an edge computing / IoT framework becomes a viable solution as the policy inference typically requires much less computational effort. By doing so, the computationally heavy data handling processes as well as gradient descent-based learning steps are executed in soft real time on an appropriate edge hardware device while only the RL policy inference is realized in hard real time on the embedded control hardware. Challenges of this edge computing RL approach include keeping the delay between the learning process executed on the edge device and the data generating / policy inference process on the embedded controller as small as possible, ensuring a reliable and safe data communication between both entities as well as minimizing energy consumption of the entire learning and control process as much as possible. Against this background, the given project investigated to what extent VEDLIoT hardware and software technologies can contribute to the stated objectives. The application scope is a power electronic-based energy conversion system, i.e., an electric drive system, utilizing real-world laboratory test benches.

This introduction is followed by 5 further chapters and a bibliography. Next, the general idea of the project is discussed. This is followed by the implementation in detail in the individual components of hardware and software. Connected to the implementation, the evaluation of the individual solutions is discussed. The following chapter provides a brief overview of the impact of this project so far and the continuation of work on the topics addressed. It concludes with a summary and outlook of the project.

2 Idea and Architecture

Controlling electrical power systems is a challenging problem faced by many industry branches such as industrial automation, automotive and aerospace engineering or energy generation and distribution. A standard control loop (cf. Figure 1) consists of physical components such as the plant process, the actuator and corresponding sensors, which are connected to the controller as the central decision center. The controller itself consists of a hardware component, typically an embedded hardware (microprocessor, FPGA,...) with comparatively low computing power, and the controller software, which contains the operational intelligence of the overall controlled process as a mathematical program.

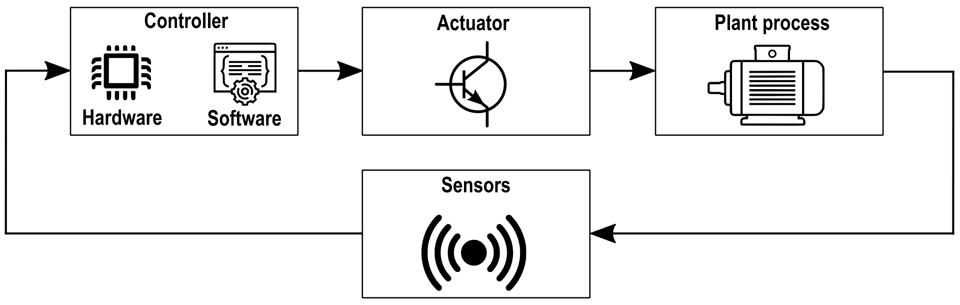


Figure 1: Principle sketch for the control of technical systems

State-of-the-art control engineering design patterns are highly dependent on expert knowledge, both regarding the specific plant system models and the domain-specific control methods. Consequently, the development process of a new power system's control unit is a tedious task binding both significant human resources as well as time (cf. Figure 2). The latter is typically due to the model-to-reality gap: modern control methods, such as model predictive control (MPC), rely on accurate models which are rarely available in industry projects such that systematic modelling errors must be compensated by trial-and-error controller tuning.

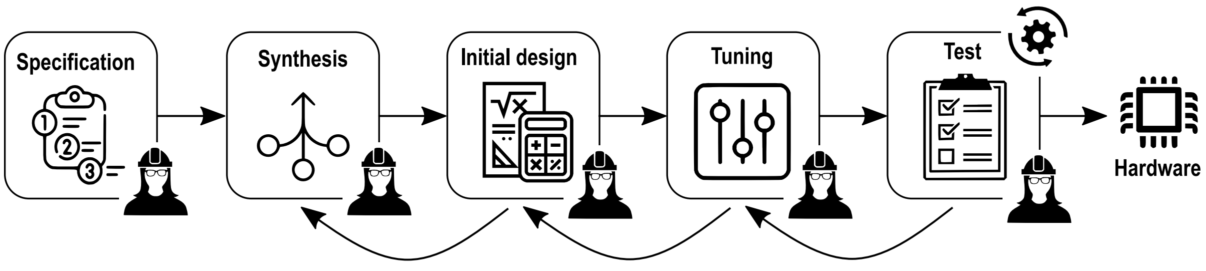


Figure 2: State-of-the-art controller design; iterative process based on expert knowledge and much manual work

Due to the advances of artificial intelligence (AI) inspired control algorithms making use of reinforcement learning (RL) techniques, this development process might be significantly accelerated in the future (cf. Figure 3). Instead of manual, expert-driven working steps, automated and learning algorithms are used to directly interact with the plant system. An optimal control policy is then learned in a model-free but data-driven fashion which can be considered an iterative and incremental optimization process (cf. Figure 4).

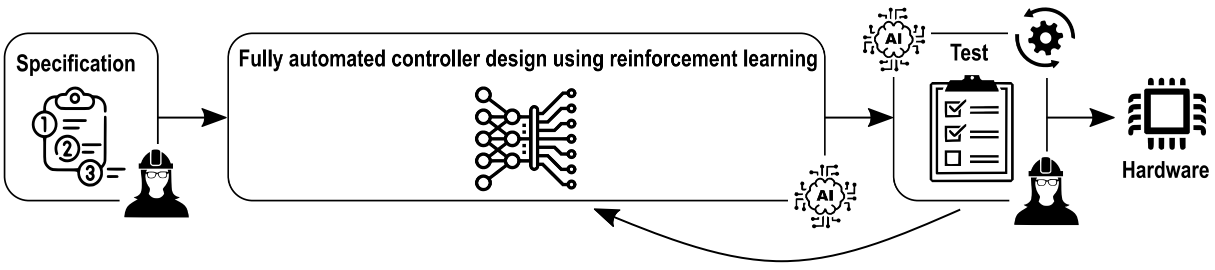

Specification
Fully automated controller design using reinforcement learning
AI
Test
Hardware

Figure 3: Possible future; AI-powered algorithms learn control strategies based on data

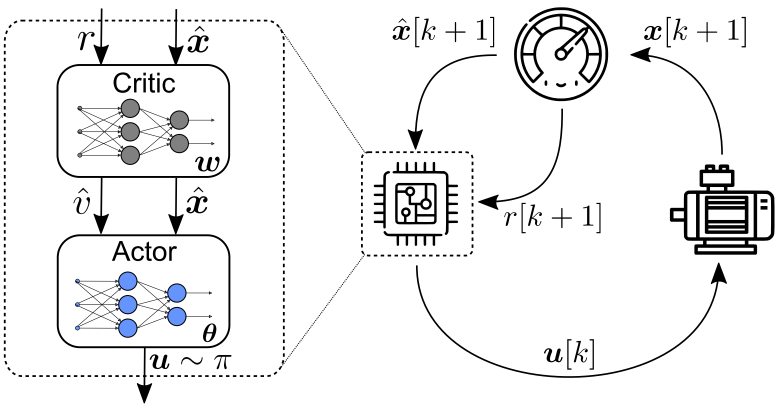


Figure 4: Actor critic-based reinforcement learning loop

The corresponding learning and optimization process from cf. Figure 4 can be summarized as follows: at each time step k, the controller applies a manipulated variable (action) $u[k]$ to the controlled system (shown here as an electric drive). The system's state reaction $x[k+1]$ is detected by sensors and evaluated with respect to the given control objectives using the so-called reward function $r[k+1]$. This contains a mathematical representation of the control requirements in terms of an optimization function. The better the controller has learned to manipulate the given technical system in terms of the specifications, the higher the reward will be. The resulting data is stored in real time in a corresponding data buffer, which is continuously analyzed by the RL-based controller in order to evaluate which action sequences have yielded a high reward in the past. Based on this analysis, the RL controller continuously adapts its control strategy, which is typically represented by an online adaptive artificial neural network (ANN). For this purpose, computationally expensive gradient descent methods are used in combination with algorithmic differentiation.

In addition, RL algorithms can also be used to automate the final controller test. In this case, an RL adversary is used, whose reward function consists in putting the actual RL controller out of step by generating particularly challenging operating scenarios. This realizes a mutual learning process between the controller and the test AI algorithms, which can be automated without the help of experts.

However, there is one key hurdle when developing RL-based algorithms for power system control: most modern power systems are based on power electronics which need to be operated at high sampling and control rates due to their speedy system dynamics. Typically, rates are in the range from single digit up to hundreds of kHz. Since any control algorithm must run under hard real-time constraints, this also applies to RL algorithm used for control purposes. This would require that the embedded control hardware must be able to execute both, the RL inference and training steps (i.e., on deep ANNs) in the microsecond range. This is in direct contrast to typical cost-optimized microcontrollers used in these applications, which are far away from the required computational performance level.

Proposed technology / innovation

To overcome this initially seemingly insurmountable hurdle, an edge computing / IoT-based solution can be approached: although an RL-based controller requires both the training and inference steps to learn optimal control policies, only the inference step (ANN forward execution) must be implemented on the embedded hardware device under hard real-time constraints. With respect to Figure 4 that would be the so-called actor network which represents the controller policy. End-of-line embedded hardware is already nowadays capable of performing the inference of small ANNs based on microprocessor architectures and up to medium-sized ANNs using cost-efficient FPGAs under hard real time, i.e., this part of the problem is manageable with contemporary technology. In contrast, the much more computationally demanding learning algorithm parts, such as the critic inference, as well as the gradient descent step (i.e., updating the ANNs' weights) cannot be updated in hard real time on such embedded hardware. But this is also not necessary, since state-of-the-art RL algorithms only update the ANNs' weights during training in a cautious way and with a comparatively small step size. Hence, a small delay in a soft real-time sense between receiving new data from the control plant and updating the controller policy ANN weights can be tolerated. This enables RL training to be performed remotely from the embedded hardware using edge computing / IoT technologies on specialized hardware (such as developed within the VEDLIoT project) that is executed asynchronously to the embedded controller process. This leads to a partitioning or decoupling of the most computationally intensive parts of the RL algorithms from the embedded hardware and thus also allows innovative deep RL approaches to be used for fast, dynamic power systems (cf. Figure 5).

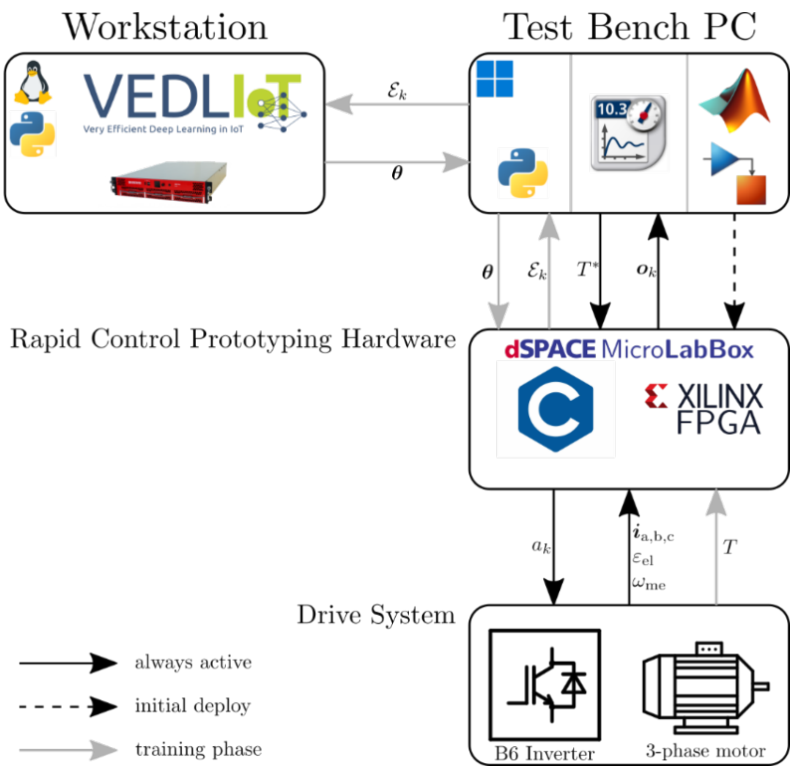


Figure 5: Schematic of the Power-Edge-RL pipeline

Nevertheless, this innovative learning and training process is not risk-free and many challenges have to be addressed when edge computing-based RL solutions are applied to safety-critical control tasks:

- The latency with which new RL ANNs' weights are sent from the edge / IoT platform to the embedded controller should be kept as low as possible to ensure fast and stable learning. This requires specialized hardware accelerators on the edge / IoT hardware to calculate the training update step as fast as possible.
- For the same reasons, the communication link between the embedded and IoT device must function with the lowest possible latency and without data loss.
- Since the energy efficiency of power systems is usually of great importance to the user, i.e., also the learning task has to be realized with minimal energy conversion losses, the edge / IoT device should be as energy-saving as possible.

3 Implementation

This section describes the technical implementation of the Power-Edge-RL pipeline in detail. This section will be structured in the different elements of the shown schematic in Fig. 5.

3.1 Drive System

Figure 6 shows the used experimental setup for this project. The device under test (DUT) is a permanent magnet synchronous motor (PMSM), where the reinforcement learning will be applied to control the motor. The current states of the DUT are measured via electrical and mechanical sensors to send the information to the rapid control prototyping hardware (RCPH). A commercially available drive system of Beckhoff Automation is used as load for the DUT.

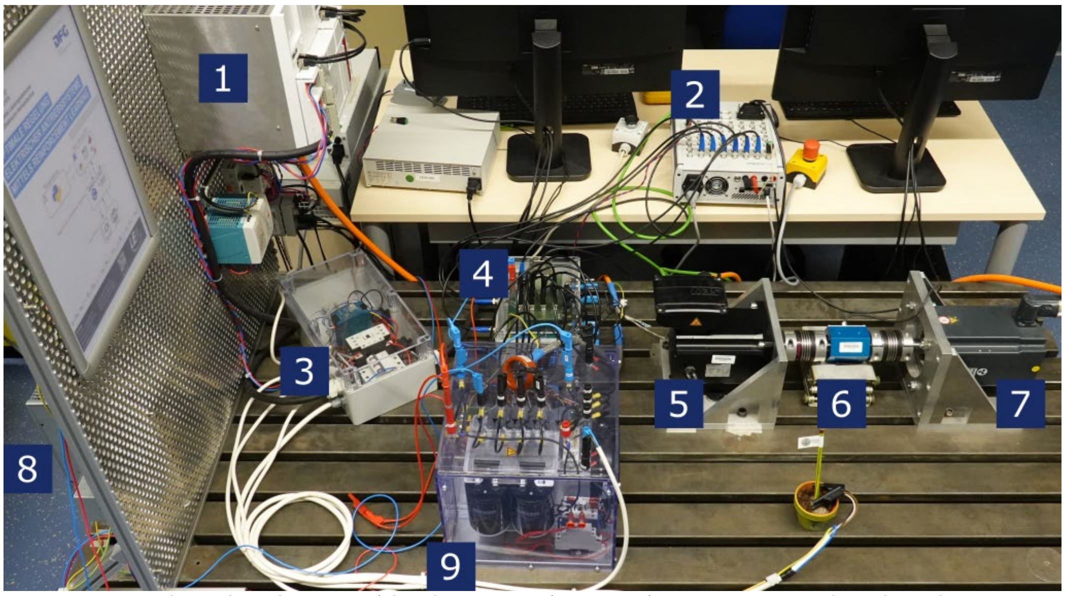
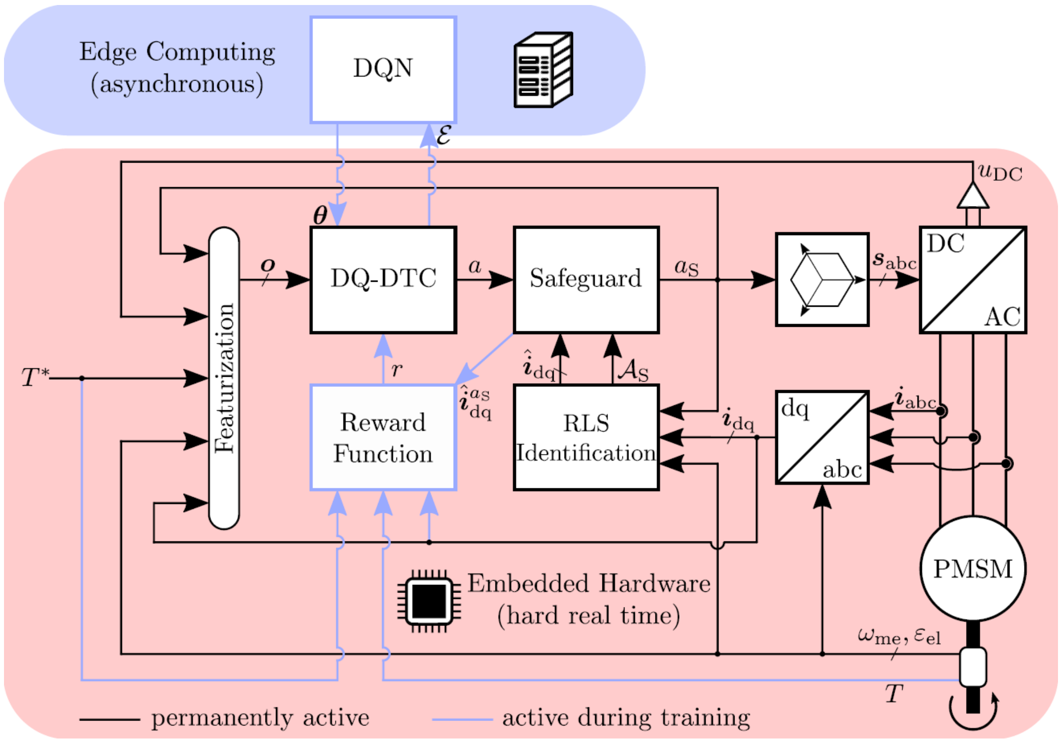


Figure 6: Experimental test bench setup; 1) load inverter, 2) RCPH, 3) protective switch and auxiliary power supply, 4) electric sensors, 5) DUT, 6) drive train and torque sensor (without safety cover), 7) load motor, 8) DC-link chopper, 9) DUT inverter

3.2 Rapid Control Prototyping Hardware and Hard Real Time Software

A dSPACE MicroLabBox is the used RCPH, which contains a programmable logic (PL) in form of an FPGA and two CPUs. The CPUs and the PL are coupled internally via a bus system. The RCPH executes the hard real time part of the applied RL control loop. The schematic of the implemented deep Q direct torque control (DQ-DTC) of [1] is shown in Figure 7. The proof of the DQ-DTC approach is performed in simulation in [3]. To transfer the experiment from simulation to real-world experiment, the forward path of the deep Q network (DQN) was transferred into the FPGA of the RCPH to meet the timing constraints of 50 µs in each sampling step of the DQ-DTC. Also, the inputs and outputs of the RCPH are mapped via the PL to the processors to name all components of the implementation in the PL. The processing of the measurement values takes place in the processor before the so-called feature vector or observation is the input to the DQN in the PL to receive the q-values as an output. With the q-values the policy can be applied to determine the optimal action of the agent.

The data-driven system identification via a recursive least squares (RLS) algorithm is used to apply a safeguard to guarantee that the drive system is not damaged during the training procedure of the RL controller. The RLS identifies a plant system model utilizing the same data as the RL agent which is used for predicting the system response on agent's actions before actually applying them on the real system. This is especially important at the beginning of the training phase when most of actions are chosen randomly by the policy of the DQ-DTC, which can lead to unsafe operation conditions (e.g., massive overcurrents). The safeguard ensures that unsafe actions chosen by the RL agent will be overwritten by safe alternative actions within the system constraints and controllable region. A more detailed description of the implementation and functionality is shown in [1].


Edge Computing (asynchronous)
DQN
$\mathcal{E}$
u_{DC}
$\boldsymbol{\theta}$
$\boldsymbol{o}$
DQ-DTC
a
Safeguard
a_{S}
$\boldsymbol{s}_{\mathrm{abc}}$
DC
AC
Featurization
T^*
r
$\hat{\boldsymbol{i}}_{\mathrm{dq}}$
$\mathcal{A}_{\mathrm{S}}$
$\hat{\boldsymbol{i}}^{a_{\mathrm{S}}}_{\mathrm{dq}}$
Reward Function
RLS Identification
$\boldsymbol{i}_{\mathrm{dq}}$
dq
abc
$\boldsymbol{i}_{\mathrm{abc}}$
PMSM
Embedded Hardware (hard real time)
$\omega_{\mathrm{me}}, \varepsilon_{\mathrm{el}}$
T
permanently active
active during training
Figure 7: Schematic of the safeguarded DQ-DTC structure

3.3 Test Bench PC

During operation, the test bench PC is the interface for the communication between RCPH and the workstation. The communication to the RCPH is performed with a XIL-API and from the test bench PC to the workstation a TCP/IP protocol is used. Further functionalities of the test bench PC are the human machine interface to control the RCPH via control desk and to program the RCPH with Matlab/Simulink.

3.4 Workstation

The workstation performs asynchronously the learning step of the DQN and sends the updated network weights back to the RCPH via the test bench PC.

Within this project three different workstations are investigated. First, a desktop OC is used, as can be seen in [1]. The system has an Intel Core i7-9700 CPU (3.00 GHz) processor with 32 GB RAM and runs with Windows 10 (64 Bit) as operation system. Second, a Nvidia Jetson Orin is used as a representative for the VEDLIoT t.RECS edge server. Third, an AMD Xilinx Versal eval board is used to perform the learning in an FPGA with the VEDLIoT toolchain for FPGA-based acceleration of deep neural networks STANN [4]. The first two workstation solutions use the python package “keras-rl2” [5]-[6] to implement the learning.

4 Evaluation and Characterization

In this section, the evaluation of the different solutions is shown. Beginning with the baseline, a near edge solution where the learning step (backward pass) is performed on a PC is discussed. The hard real-time necessary forward step is always implemented in the RCPH. For the further evaluation, the solution goes in the direction of far edge solutions with the implementation of the backward pass in the Nvidia system and afterwards in an FPGA system from AMD Xilinx. These solutions are also displayed in Figure 8 with an outlook to a fully integrated embedded solution, where backward and forward pass are performed in one system while controlling the drive system. To the current state, it is not possible to implement both, forward and backward pass, in the RCPH, because the size of the DQN allocates nearly all of the available resources of the FPGA in the RCPH.

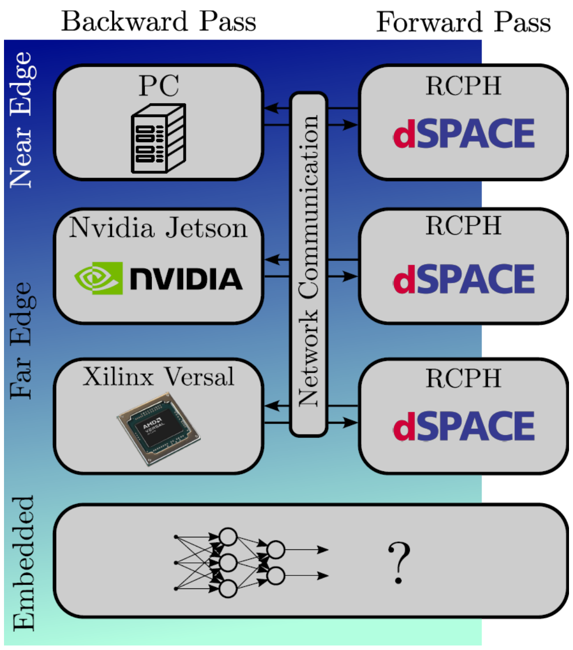


Figure 8: Classification of the individual IoT solutions

4.1 Evaluation of the Power-Edge-RL Pipeline

In [1] an in-depth evaluation of the Power-Edge-RL pipeline is described. The asynchronous learning pipeline in combination with a safeguarding-layer enables a torque control with no expert knowledge needed in a few minutes. A video of the training phase is available under https://youtu.be/hQ49Mc6LV78. In the following, a short summary of the reported evaluations from [1] is given.

The most basic validation is the rising reward over time in the learning curve to show that the Power-Edge-RL pipeline enables the training process successfully (cf. Fig. 9). Further, Fig. 10 shows the performance of one DQ-DTC agent at different times of a 10-minutes learning phase, first at the start, second in the middle and third in the end of the learning phase. It is clearly visible that the tracking in the first row improves from the left to the right column. Moreover, the last row shows that the number of overwritten actions by the safeguard decreases over time.

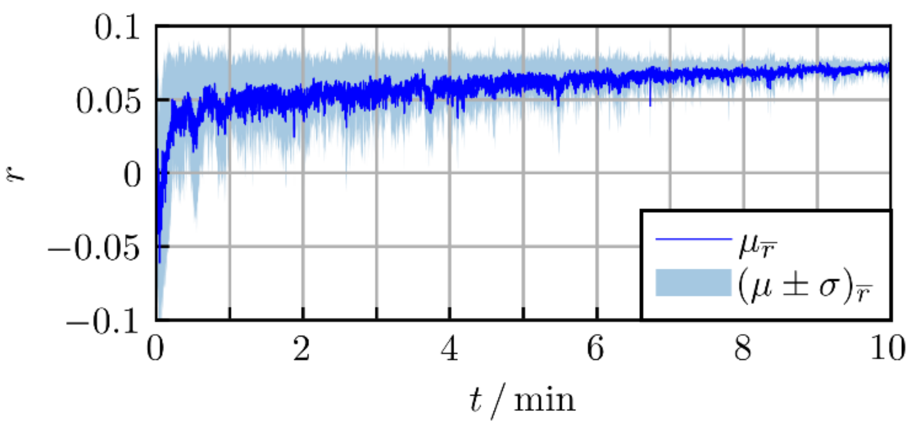


Figure 9: Mean learning curve over ten trainings of the DQ-DTC with the Power-Edge-RL pipeline

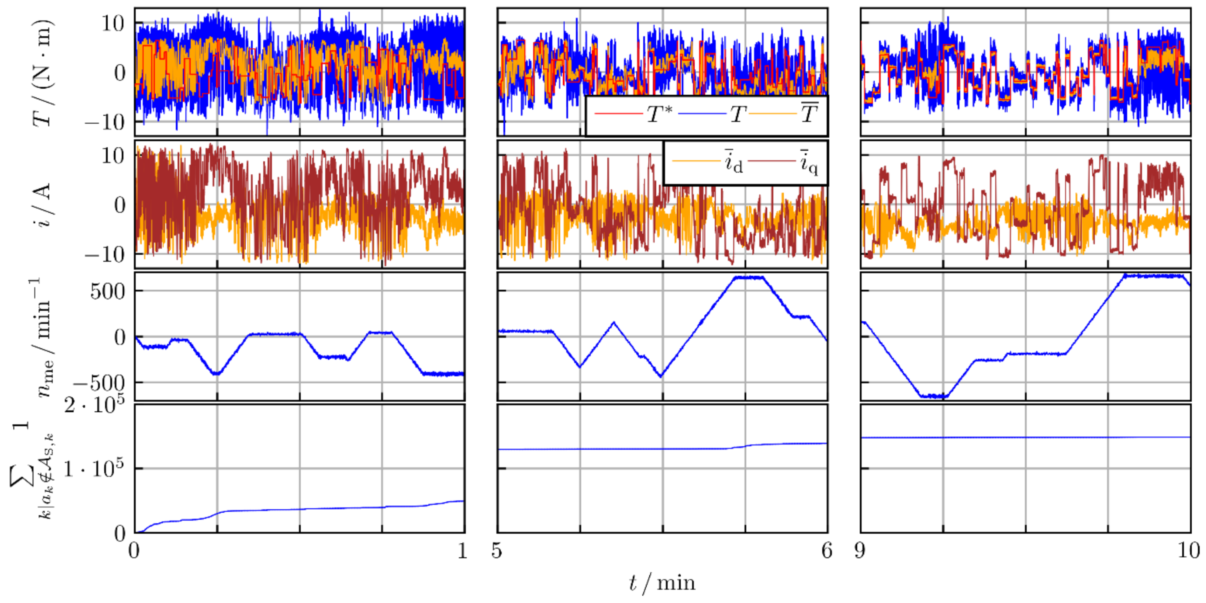


Figure 10: Control performance in an early (left), intermediate (center) and late (right) phase of an exemplary training

4.2 Evaluation Nvidia Jetson Orin

The previously developed edge learning toolchain has been adapted for usage on the t.RECS equivalent Nvidia Jetson AGX Orin hardware. Early evaluations hereon revealed that utilization of the built-in SoC GPU does not suffice to accelerate execution of RL / gradient descent steps, which is attributed to the rather low number of features that are sampled per timestep. As GPU performance commonly scales with increasing width of data, it seems that the break-even point, that would allow effective use of such resources, is not reached for the given scenario of power system control.

The further investigation, performed under utilization of the Jetson's CPU, did also not meet encouraging performance levels. This can be attributed to the limited computational power of the hardware. As these results did not encourage further testing in a real-world laboratory experiment, a precise analysis of computational time demand and resulting expectable training time has not been prioritized in favor of further work packages.

4.3 Evaluation of the FPGA Solution

The VEDLIoT Software STANN (Synthesis Templates for Artificial Neural Networks) [4] has been used to implement an FPGA-based accelerator for the DQN training process. As a result, the latency of a single DQN training step has been improved significantly. The desktop PC needs around 10ms per learning step, while the FPGA implementation achieves a latency of 1.45ms when implemented on the Xilinx Versal FPGA (vck190). The FPGA implementation was validated using a Python simulation of the motor control environment [7]. A neural network consisting of ten fully connected layers with 128 neurons in each layer and leaky ReLU activations has been used as the model for the DQN algorithm, and the training of this network is the most significant part of the DQN accelerator. The accelerator implements training based on stochastic gradient descent with a (configurable) mini-batch size of 32.

Table 1 shows a first evaluation of the accelerator based on timing and utilization reports generated by Xilinx Vitis High-Level Synthesis. The design has been implemented for a Xilinx Alveo U50 accelerator card and a Xilinx Versal Evaluation Board (vck190). The accelerator uses one compute engine for each layer, arranged in a pipelined fashion. Each compute engine uses a systolic array for matrix multiplication. The size of this systolic array is the number of PEs (processing elements) per layer. As expected, the resources used for the accelerator increase, while the latency decreases when we increase the number of PEs. Furthermore, it can be seen in the last row of Table 1 that the same architecture implemented on the Versal architecture is faster while using fewer DSPs. The Versal architecture includes floating point DSPs which are a significant advantage for the floating point value based training of the neural network.

Table 1: FPGA-based DQN Training Results

Hardware Platform	PEs per Layer	Latency (500 updates)	Latency (1 update)	LUTs	BRAM	URAM	DSPs
Alveo U50	1	39.75s	79.49ms	131804 (15%)	447 (16%)	130 (20%)	466 (7%)
Alveo U50	8	2.76s	5.51ms	199875 (22%)	474 (17%)	130 (20%)	610 (10%)
Alveo U50	16	1.00s	1.98ms	459151 (52%)	492 (18%)	130 (20%)	826 (13%)
Versal	16	0.73s	1.45ms	515739 (57%)	487 (24%)	130 (26%)	587 (29%)

The DQN accelerator is based on STANN, an HLS library for the implementation of neural network inference and training on FPGAs. In the real-world testbench, the interaction of the reinforcement learning agent with the environment and the training process are decoupled and happen on separate hardware. However, STANN could also be used to accelerate the neural network inference for this interaction. Table 2 shows an evaluation of the inference of the ten-layer neural network model with STANN. For the inference part, a batch size of one is necessary because the interaction of the agent with the environment happens sequentially. Thus only one observation is available for each inference. Here, the STANN-based accelerator achieves a low latency of 12μs, as appropriate for the real-time environment.

Table 2: Evaluation of Inference with STANN

PEs per Layer	Batchsize	Latency	LUTs	BRAM	DSPs
1	32	21ms	24822 (3%)	148 (10%)	142 (4%)
4	32	1.343ms	37895 (4%)	148 (10%)	187 (6%)
8	32	0.364ms	81141 (9%)	148 (10%)	367 (12%)
4	1	43μs	36666 (4%)	148 (10%)	187 (6%)
8	1	12μs	79912 (9%)	148 (10%)	367 (12%)

4.4 Comparison

The detailed comparison of the baseline with the FPGA-based training is currently ongoing. It is planned to write another journal publication about the upcoming test bench experiment. This article will include detailed information about the acceleration of the learning process and will also show an in-depth analysis of the control performance.

5 Exploitation

During the project, the following contributions have been published:

- Publication: S. Zhang, O. Wallscheid and M. Porrmann, "Machine Learning for the Control and Monitoring of Electric Machine Drives: Advances and Trends," in IEEE Open Journal of Industry Applications, doi: 10.1109/OJIA.2023.3284717.
- Publication: F. Book, A. Traue, M. Schenke, B. Haucke-Korber and O. Wallscheind, Gym-Electric-Motor (GEM) Control: An Automated Open-Source Controller Design Suite for Drives, IEMDC, 2023.
- Publication: B. Haucke-Korber, M. Schenke and O. Wallscheid, Deep Q Direct Torque Control with a Reduced Control Set Towards Six-Step Operation of Permanent Magnet Synchronous Motors, IEMDC, 2023.
- Publication: M. Schenke,B. Haucke-Korber, and O. Wallscheid. "Finite-Set Direct Torque Control via Edge Computing-Assisted Safe Reinforcement Learning for a Permanent Magnet Synchronous Motor." (currently under peer review at IEEE TPEL)
- Publication: M. Rothmann, M. Porrmann, M. Schenke,B. Haucke-Korber, and O. Wallscheid. "FPGA Acceleration of Reinforcement Learning-Based for Controlling Intelligent Power Systems." (currently under preparation)
- Open-source toolchain: https://github.com/max-schenke/EdgeRL

The work of this project will be continued by a funded DFG project and an upcoming BMBF founding for 3 years starting in October. Nevertheless, the project partner of University of Osnabrück and Paderborn University are looking for further funding possibilities to continue their research on reinforcement learning together. An upcoming research goal is to fill the last part of Figure 8 with a full embedded solution, which could be solved in this cooperation.

6 Summary and Next Steps

During the project the Power-Edge-RL pipeline was successfully implemented and evaluated, whereby the focus is on the IoT workstation integration, which performs the learning step (backward pass). A PC is used as baseline and near edge solution. During the runtime of the project, further solutions of the far edge category were considered. The FPGA-based solution could achieve a speedup of factor 10 in first studies, utilizing VEDLIoT hardware and accelerator implementations, but the final comparison is still pending.

In the next steps, the Power-Edge-RL pipeline will be applied to further control approaches like presented in [2], to enable real world experiments. Further, the research consortium is looking for a new funding to continue their collaboration to implement an embedded solution, where one device controls the drive system and performs the learning step of the ANN.

7 Literature

[1] M. Schenke, B. Haucke-Korber and O. Wallscheid, *Finite-Set Direct Torque Control via Edge Computing-Assisted Safe Reinforcement Learning for a Permanent Magnet Synchronous Motor*, in review process for journal publication, preprint available at https://doi.org/10.36227/techrxiv.22032578.v2

[2] B. Haucke-Korber, M. Schenke and O. Wallscheid, *Deep Q Direct Torque Control with a Reduced Control Set Towards Six-Step Operation of Permanent Magnet Synchronous Motors*, IEMDC, 2023.

[3] M. Schenke and O. Wallscheid, "A Deep Q-Learning Direct Torque Controller for Permanent Magnet Synchronous Motors," in *IEEE Open Journal of the Industrial Electronics Society*, vol. 2, pp. 388-400, 2021.

[4] M. Rothmann and M. Porrmann, "STANN - Synthesis Templates for Artificial Neural Network Inference and Training," *17th International Work-Conference on Artificial Neural Networks (IWANN2023)*, Ponta Delgada, Azores, Portugal, 2023

[5] M. Plappert, "Keras-RL," 2016. [Online]. Available: https://github.com/keras-rl/keras-rl

[6] T. McNally, "Keras-RL2," 2019. [Online]. Available: https://github.com/wau/keras-rl2

[7] P. Balakrishna et al., "gym-electric-motor (GEM): A Python toolbox for the simulation of electric drive systems," JOSS, vol. 6, no. 58, p. 2498, 2021.

ICT-56-2020 – Next Generation Internet of Things

FINAL TECHNICAL REPORT

DUNE

Real-Time Localization System

Version 1.1

Document Information	
Contract Number	957197
Project Website	https://vedliot.eu/
Dissemination Level	PU (Public)
Nature	R (Report)
Contractual Deadline	30 June 2023
Author	Guillem Boquet, Ivan Pisa, Borja Martinez, Xavier Vilajosana
Contributors	
Reviewers	Ivan Pisa, Borja Martinez, Simon Bouget

The VEDLIoT project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No 957197.

Change Log

Version	Date	Description of Change
0.01	2023-06-20	Initial draft.
0.10	2023-06-29	Internal review.
1.00	2023-06-30	Finalization.
1.10	2023-07-26	Correction of minor issues. Addition of sections 4.1.3 and 4.1.4.

Table of Contents

List of Figures

List of Tables

Acronyms

AI	Artificial Intelligence.
AoA	Angle of Arrival.
API	Application Programming Interface.
BLE	Bluetooth Low Energy.
CDF	Cumulative Distribution Function.
CEP	Circular Error Probability (CEP).
CNN	Convolutional Neural Networks.
CPS	Cyber-Physical Systems.
CTE	Constant Tone Extension.
DL	Deep Learning.
ECDF	Experimental Cumulative Distribution Function.
EWMA	Exponential Weighted Moving Average.
FCNN	Fully-Connected Neural Networks.
IMU	Inertial Measurement Unit.
IN3	Internet Interdisciplinary Institute.
IoT	Internet of Things.
KF	Kalman Filter.
LIDAR	Laser Imaging Detection and Ranging.
MQTT	MQ Telemetry Transport.
MUSIC	MUltiple SIgnal Classification.
OPEX	Operational Expenditure.
PE	Positioning Engine.
RF	Radio Frequency.
RFID	Radio Frequency Identification.
ROS	Robot Operating System.

RPI	Raspberry Pi.
RSSI	Received Signal Strength Indicator.
RTLS	Real-Time Localization System.
SDK	Software Development Kit.
SIG	Special Interest Group.
SMA	Simple Moving Average.
TCP	Transmission Control Protocol.
UCA	Uniform Circular Array.
UDP	User Datagram Protocol.
UOC	Universitat Oberta de Catalunya.
UWB	Ultra-WideBand.

Executive Summary

DUNE utilizes distributed computing capabilities at the far-edge, edge, and cloud levels to effectively break down estimation steps, addressing the demanding performance and scalability requirements of real-time asset tracking applications.

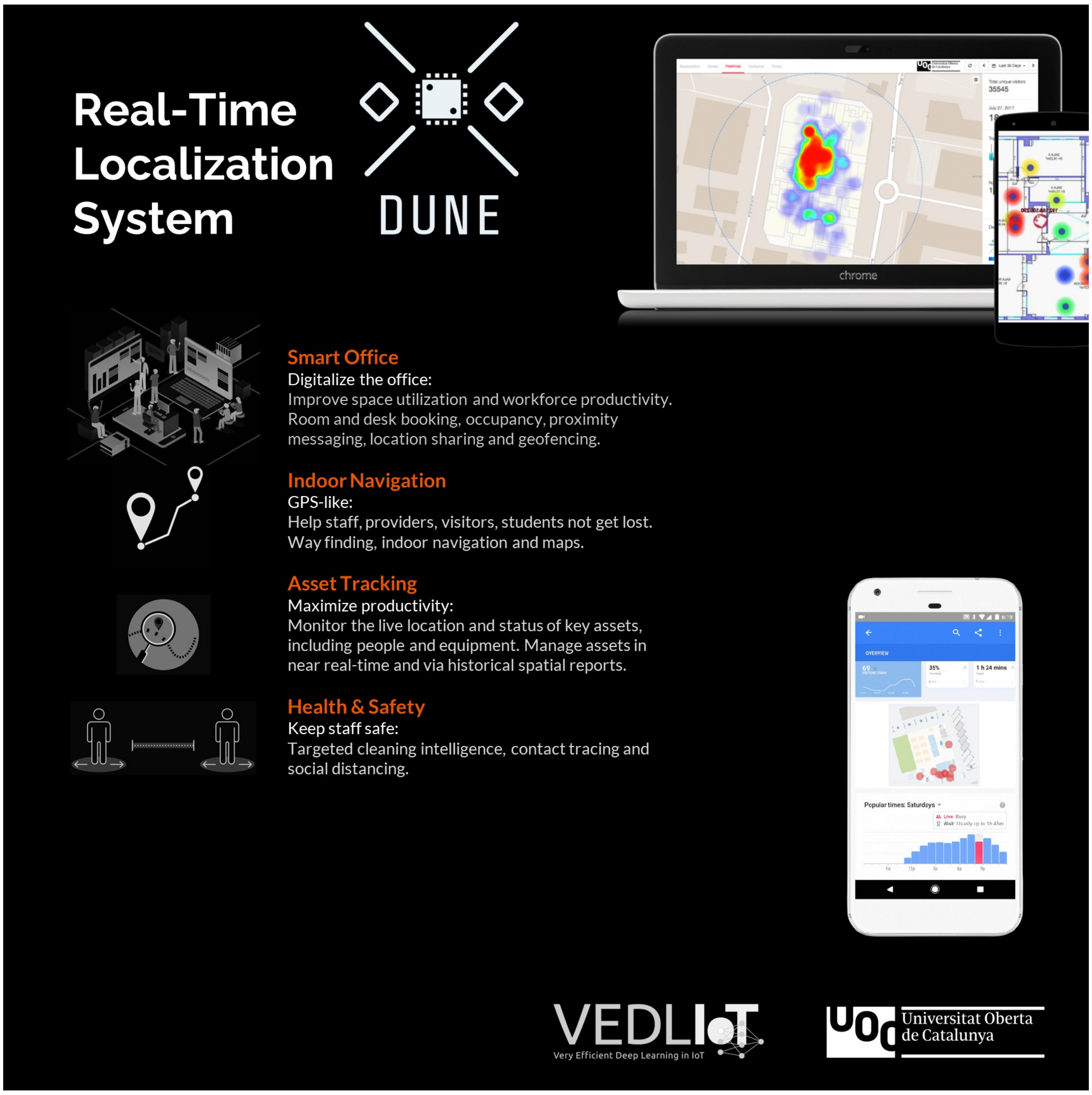
This document demonstrates that DUNE's Real-Time Localization System (RTLS) outperforms signal processing state-of-the-art methods in estimating the direction of emitted Bluetooth Low Energy (BLE) signals using a Deep Learning (DL) model. A key advantage lies in the efficient execution of the DL model on the VEDLIoT hardware at the far-edge, offering a clear contrast to conventional approaches that necessitate complex computation and lack scalability.

By combining estimates from far-edge nodes, DUNE achieves sub-meter accuracy in position estimation at the edge-cloud level. Looking ahead, DUNE envisions incorporating other existing wireless technologies like WiFi, Radio Frequency Identification (RFID), and Ultra-WideBand (UWB), employing DL to provide efficient and customized positioning solutions. Consequently, the design and implementation of DUNE have been carefully structured to facilitate seamless integration of these technologies.

DUNE RTLS has been successfully implemented in both pre-production and production environments. The former allows for algorithm development and testing in a controlled setting, leading to the open publication of DL models and datasets used for training.[1][2] The latter showcases DUNE's capabilities in a practical setting, where it is deployed on the first floor of the recently inaugurated Universitat Oberta de Catalunya (UOC)'s interdisciplinary R&I hub, spanning an area of 1200 m^2 and accommodating a daily average of 100 active individuals.[3]

A summary of DUNE RTLS service is shown in Figure 1 and Figure 2.

[1]https://bitbucket.org/wineuoc/aoa-ble-deeplearning
[2]https://bitbucket.org/wineuoc/aoa-ble-dataset
[3]https://www.uoc.edu/portal/en/news/actualitat/2022/269-opening-hub-research-UOC.html



Real-Time Localization System

DUNE

Smart Office

Digitalize the office:

Improve space utilization and workforce productivity. Room and desk booking, occupancy, proximity messaging, location sharing and geofencing.

Indoor Navigation

GPS-like:

Help staff, providers, visitors, students not get lost. Way finding, indoor navigation and maps.

Asset Tracking

Maximize productivity:

Monitor the live location and status of key assets, including people and equipment. Manage assets in near real-time and via historical spatial reports.

Health & Safety

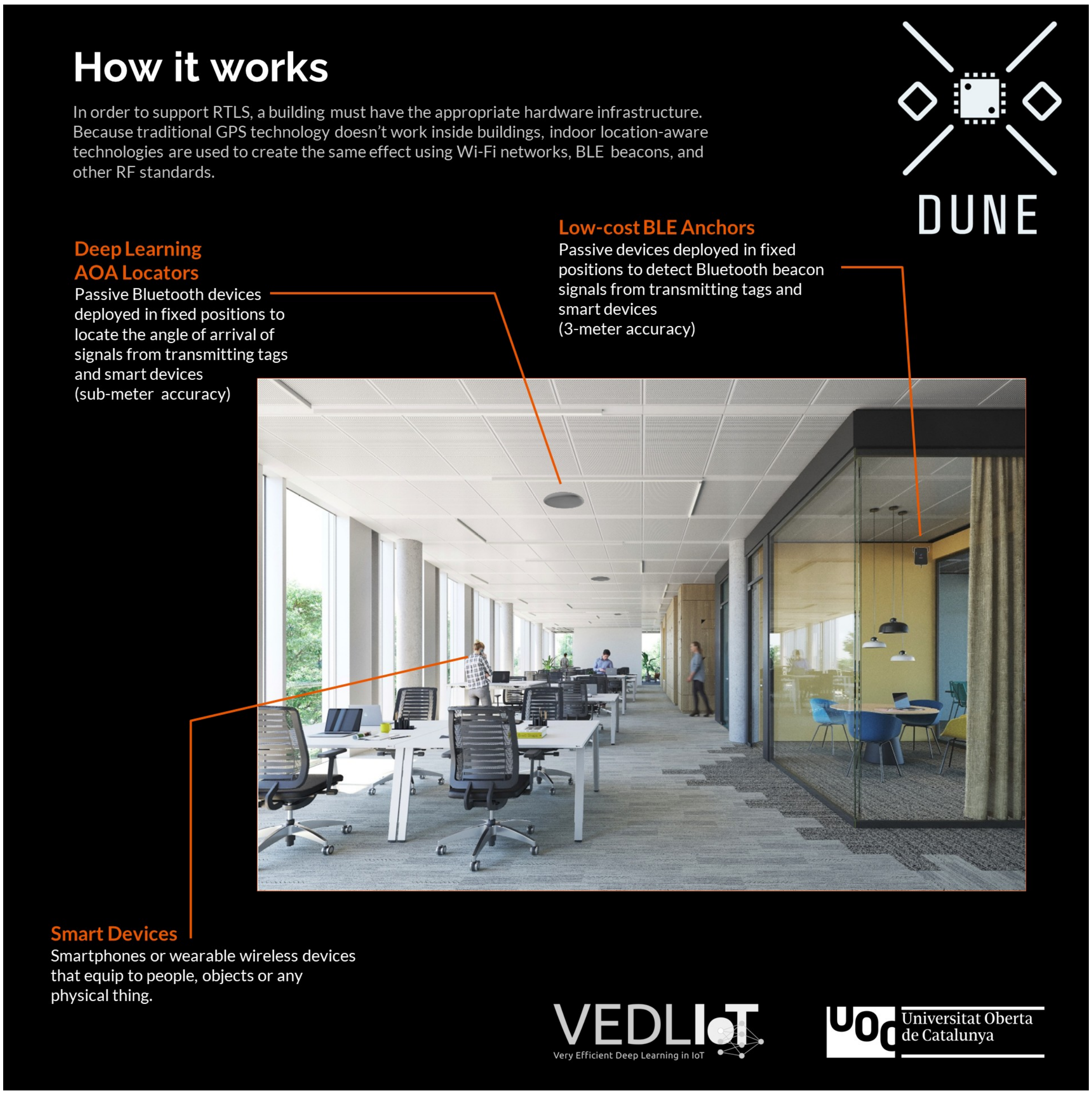
Keep staff safe:

Targeted cleaning intelligence, contact tracing and social distancing.

VEDLIoT Very Efficient Deep Learning in IoT

Universitat Oberta de Catalunya

Figure 1. Top part of DUNE RTLS datasheet.



How it works

In order to support RTLS, a building must have the appropriate hardware infrastructure. Because traditional GPS technology doesn't work inside buildings, indoor location-aware technologies are used to create the same effect using Wi-Fi networks, BLE beacons, and other RF standards.

Deep Learning AOA Locators
Passive Bluetooth devices deployed in fixed positions to locate the angle of arrival of signals from transmitting tags and smart devices (sub-meter accuracy)

Low-cost BLE Anchors
Passive devices deployed in fixed positions to detect Bluetooth beacon signals from transmitting tags and smart devices (3-meter accuracy)

Smart Devices
Smartphones or wearable wireless devices that equip to people, objects or any physical thing.

DUNE
VEDLIoT
Very Efficient Deep Learning in IoT
UOC Universitat Oberta de Catalunya

Figure 2. Bottom part of DUNE RTLS datasheet.

1 Introduction

The advancing capabilities and performance of Cyber-Physical Systems (CPS) have been instrumental in transforming various industries and social environments [1]. These advancements empower us to address increasingly complex challenges by harnessing automation and leveraging Artificial Intelligence (AI), resulting in services that enhance the overall quality of work and life [2]. Among the critical pillars of the ongoing digital revolution, indoor positioning technologies play a vital role, finding applications in diverse areas such as robotics, logistics, safety, and more [3]. In this context, DUNE RTLS aims to harness the power of diverse localization inputs and technologies to provide real-time asset tracking and positioning. Leveraging a combination of far-edge, edge, and cloud distributed computing capabilities alongside the VEDLIoT hardware, the system is engineered to meet the performance and scalability demands of various real-time industrial applications.

In 2019, the Bluetooth Special Interest Group (SIG) consortium standardized Direction Finding based on Angle of Arrival (AoA), enabling cost-effective indoor positioning solutions [4, 5]. Proof of that is that several companies, such as Texas Instruments, Silicon Labs, Nordic and u-blox are providing AoA solutions, while other companies such as Quuppa and Blueiot provide their own end-to-end positioning system with sub-meter accuracy. In this project, DUNE focused on harnessing AoA technology. Initially, we developed DL models for AoA estimation at the far-edge, comparing them with state-of-the-art algorithms based on signal processing, such as the renowned MUltiple SIgnal Classification (MUSIC) algorithm. We have implemented an experimental framework that enables systematic data retrieval from various scenarios. For this purpose, we created a controlled pre-production environment and generated a 5-day data set, which served as training data for DL models. The data set, along with the resulting models, has been made publicly available. In addition to AoA estimation, we developed several AoA-based positioning algorithms, extensively validated in both pre-production and production scenarios. The production environment comprises a 1200 m^2 RTLS deployed at the Internet Interdisciplinary Institute (IN3) research hub, offered as a service to other research groups at the university. We observed that DL models exhibited limitations in generalization when applied to far-edge nodes in production. However, through a fine-tuning process, DL model-based positioning eventually outperformed the MUSIC algorithm, leading to improved position estimation accuracy with a near 95% probability of sub-meter precision within the tested area. Looking forward, and anticipating a rise in the complexity and heterogeneity of indoor positioning technologies, we explored the combination of AoA position estimations with proximity detection using more affordable signal strength detectors in strategic areas. This fusion of positioning approaches represents an important step towards the envisioned DL positioning fusion mechanisms by DUNE. By leveraging various wireless technologies, the ultimate goal is to achieve cost-effective and optimal indoor location performance.

The document is structured as follows. Chapter 2 provides a comprehensive technical report on DUNE RTLS, explaining its core concept and detailing the system's designed architecture. In Chapter 3, the current implementation of DUNE, along with the architectural decisions, is described in detail. Subsequently, Chapter 4 presents an in-depth analysis of the system's performance, serving as a validation of DUNE's effectiveness as both a positioning tool and a successful hardware solution for VEL-DIoT. Chapter 5 discusses the impact of DUNE throughout the project's runtime. Concluding the document, Chapter 6 provides a summary of the project and outlines the next steps for future development.

2 Idea and Architecture

DUNE is a highly versatile and scalable real-time positioning system that offers both a preproduction environment for algorithm development and testing in controlled scenarios, as well as a production environment available as a service. Going beyond a simple localization tool, DUNE's true value lies in its capacity to deliver a cost-effective and customized localization solution tailored to meet specific customer needs. By seamlessly integrating various wireless technologies with positioning capabilities such as WiFi, BLE, and RFID, DUNE achieves sub-meter accuracy for indoor positioning. Leveraging the already deployed technologies on a customer's site, DUNE can efficiently provide asset localization. Moreover, when strict accuracy requirements are necessary, DUNE has the flexibility to incorporate additional technologies and fuse localization information for enhanced precision.

The system's architecture, as depicted in Figure 2.1, was developed during Work package WP2 and presented in Deliverable D2 of the VEDLIoT project. Far-edge, edge and cloud distributed computing capabilities of VEDLIoT are used for real-time application needs, functionally splitting the fusion methods to also meet the time, performance, and scalability demands of common industrial applications. DUNE employs DL in various stages of positioning as a key component of the system. In the far-edge, DL is utilized to estimate the AoA of BLE signals and the range of both WiFi and BLE signals. At the edge, DL is harnessed to fuse position estimates from different technologies for the same asset. This processing is designed to scale seamlessly and sustain real-time operation thanks to the efficient VEDLIoT hardware.

This section provides an in-depth presentation of DUNE's architecture and its essential building blocks. The components and their connections are described in two subsections: "Far-Edge" (Section 2.1) and "Edge-Cloud" (Section 2.2).

2.1 Far-edge

In the far-edge of the CPS system, a fully distributed architecture is established, consisting of various localization sensors and technologies. These sensors, referred to as locators, are strategically deployed within the tracking area of interest, generating raw data traces. The ultimate goal is to process this raw data efficiently to obtain accurate estimates of the assets' positions. This processing occurs in proximity to the receiver and the Internet of Things (IoT) radio interface/sensor at the far-edge.

Traditionally, handling the complex processing of Radio Frequency (RF) signals has been predominantly tackled using signal processing models in the cloud. However, DUNE's architecture introduces a novel approach by incorporating a set of software components. These components play a crucial role in transforming the raw data into meaningful localization information at the far-edge. As a result, high-speed and scalable complex processing become feasible, catering to the demands of the most challenging IoT use cases.

The main hardware parts of the far-edge architecture depicted in Figure 2.1 are:

- the BLE, WiFi and RFID locators as the technology-specific receivers and

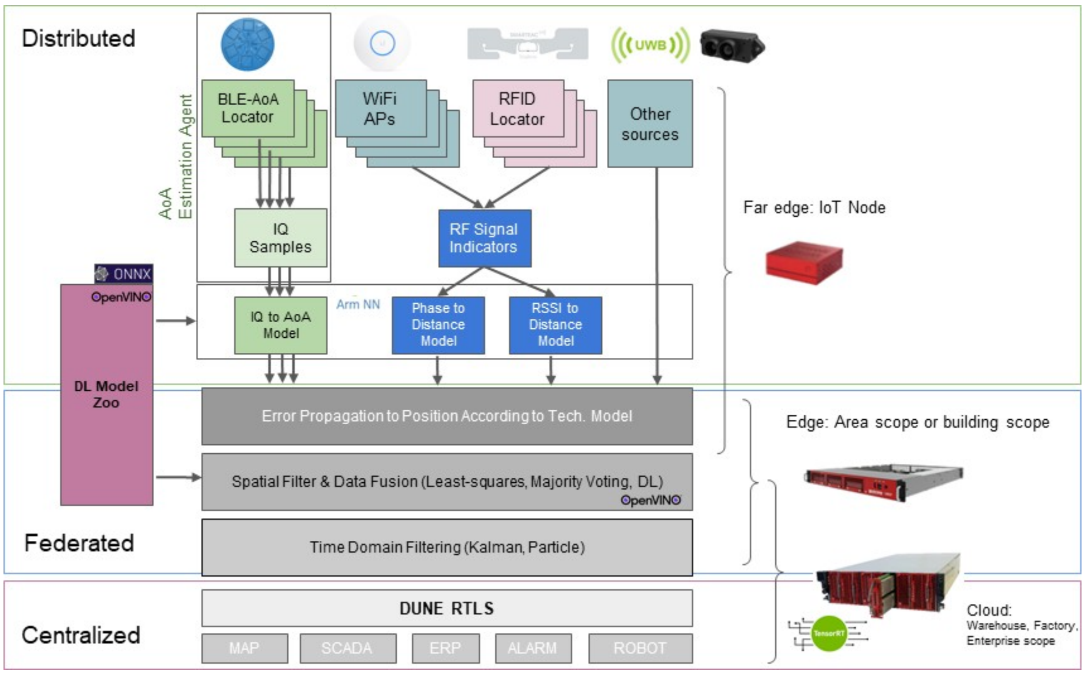


Figure 2.1. Full DUNE RTLS architecture.

- the u.RECS as the far-edge real-time processing units.

IoT use cases often require real-time tracking of multiple assets, with wireless signals transmitted every 100 ms. Note that usually multiple locators will be deployed in an indoor room to improve positioning accuracy. Each locator will receive the same data transmitted from potentially lots of assets. To accurately determine the assets' positions, estimations of the signals' AoA or range must be computed using complex models. Traditionally, this estimation is performed in the Cloud, leading to unwanted delays, the transmission of large amounts of data, limited scalability, and reduced tracking accuracy for highly mobile assets. To overcome these challenges, we designed DUNE with a far-edge AoA estimation approach. For this purpose, we leveraged the u.RECS node to host DUNE's far-edge services. The u.RECS node offers the necessary flexibility and computational power to address the diverse IoT scenarios considered in our work. It incorporates hardware accelerators essential for exploiting DL models and includes a mini PCIe slot to enable typical IoT communications [6]. The use of u.RECS ensures scalability and efficient processing of AoA estimations at the far-edge.

Regarding the locators used in our study, we employed WiFi access points and RFID readers, which are commonly available in the market or already deployed in various environments. The signal range estimation is based on the Received Signal Strength Indicator (RSSI) and phase information of the WiFi and RFID signals. We utilized fingerprinting methods or RSSI-to-distance models for this purpose. As these receivers are standard equipment, no further specific details are provided here. However, it is essential to highlight the distinct nature of BLE locators with aoA estimation capabilities. A BLE-AoA locator differs from the more common BLE-RSSI locator, as the former requires a specialized antenna array to perform AoA estimation. In contrast, the latter, being cheaper, can only sense RSSI and loosely estimate signal range. For our project, we employed the BLE-RSSI locator for locating assets within a 5-meter prox-

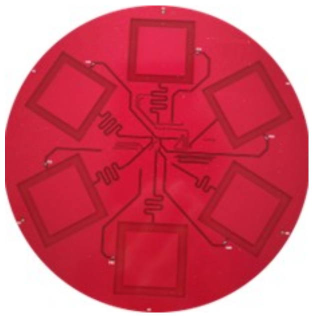
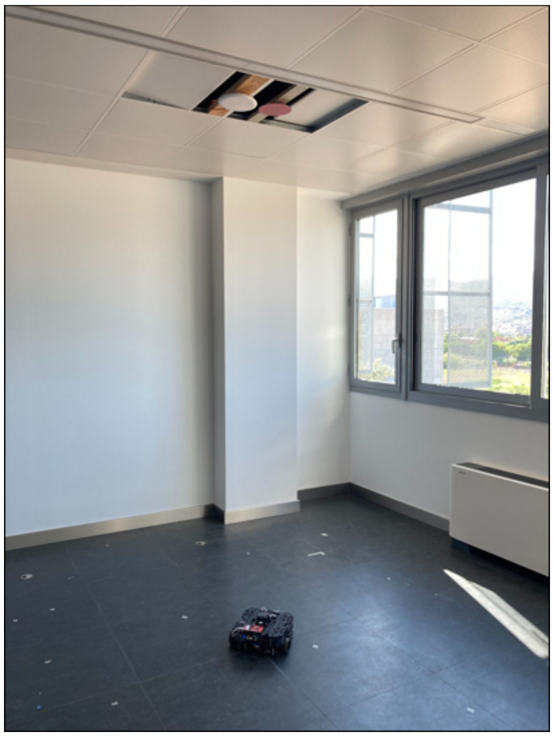



Figure 2.2. (left) Top view of DUNE's BLE-AoA locator. (right) Two close BLE-AoA and WiFi locators mounted on the ceiling of the preproduction environment, and a robot with BLE and WiFi technology.

imity of the sensor in closed areas like meeting rooms. To achieve higher accuracy, we developed a custom BLE-AoA locator, which, when combined with a u.RECS unit and advanced positioning algorithms, provides sub-meter accuracy. We have named this specialized solution the "AoA Estimation Agent."

The AoA Estimation Agent in the far-edge is composed of the AoA locator, IQ sampling and IQ to angle software blocks in Figure 2.1. The IQ sampling block samples the Constant Tone Extension (CTE) of the received BLE signal and sends the in-phase and quadrature components, known as IQ samples, to the u.RECS. The u.RECS unit is needed to be able to process in real-time the IQ samples using complex AoA estimation models, that is, to transform IQ samples to useful angles for positioning. The u.RECS runs Ubuntu Server with custom software to process IQ samples and derive the AoA of the received RF signal. The AoA locator is a BLE 5.2 receiver with AoA capabilities which is mounted in a fixed position on the ceiling of the room. Its main parts are:

- A custom 6 patch dual-polarized dipole antenna that forms a Uniform Circular Array (UCA) in Figure 2.2.
- A RF switch to sample the received signal from the 6 different antenna patches.
- The TI's LAUNCHXL-CC2640R2 BLE 5.2 board configured to use an external antenna. It uses TI's BLE SDK firmware adapted to match DUNE's custom UCA pattern. It controls the RF switch and samples the RF signal according to the BLE Direction Finding standard.

2.2 Edge-Cloud

In large deployments, DUNE real-time data processing and aggregation is done at edge-cloud. The edge-cloud processing takes advantage of filtering models in the spatial and temporal domain, that is, filtering timestamped batches of data coming from devices that processed the same information source but are located at different positions. This approach is known to lead to improved system performance because the edge filtering is capable of mitigating bad phenomena due to the characteristics of the wireless propagation medium, mainly multipath components and line-of-sight obstruction. Data fusion at the cloud is achieved through different approaches, such as majority voting schemes or DL methods trained to select the position estimations that better localize the objects under tracking. In DUNE, we implement specific components for data fusion, exploiting multi-technology, multi-source data flows in order to produce accurate object position estimations. The outcome of these methods is exposed by an Application Programming Interface (API), so end-user applications can exploit that capability.

In large deployments, DUNE utilizes real-time data processing and aggregation at the edge-cloud level. The edge-cloud processing capitalizes on filtering models in the spatial and temporal domains, handling timestamped data batches received from devices positioned at different locations but processing the same information source. This approach significantly improves system performance by mitigating issues arising from wireless propagation characteristics, especially multipath components and line-of-sight obstruction. Data fusion at the cloud is achieved using various approaches, including majority voting schemes and DL methods. These techniques are trained to select the most accurate position estimations for the tracked objects. DUNE implements specialized components for data fusion, leveraging multi-technology, multi-source data flows to generate precise object position estimations. The outcome of these methods is made available through an API, enabling end-user applications to harness this capability. The central hardware component of the edge architecture (Figure 2.1) is the t.RECS. In real-time, a single t.RECS manages subsequent filtering steps to exploit the multi-technology, time, and spatial characteristics of the received wireless signals within a specific tracking area. The t.RECS edge server is capable of running various data fusion and positioning algorithms, which combine estimates from different technologies. DL and computer vision algorithms are used for fusion by projecting the estimates and their uncertainties onto the positioning plane as ellipses surrounding each estimated position, treating them as images. These images are then processed with DL models to obtain points in the plane that provide the best position estimations for the assets, considering data from multiple sources. This computation is efficiently executed by t.RECS. On the other hand, the primary hardware component of the cloud architecture (Figure 2.1) is the Deneb RECS Box. This centralizes all position estimations, providing an API, multiple positioning services, and data analytics functionalities.

The main software components in Figure 2.1 can be summarized as follows:

- Error Projection Layer: This layer scales the estimates and associates an uncertainty based on their origin.
- Data Fusion Layer: The data fusion layer combines estimates in the spatial domain, considering their uncertainty.

- Time Domain Filtering Layer: This layer utilizes time dependencies and motion models to improve the estimates.

The edge-cloud tools manage the data fusion and processing pipeline. Far-edge devices use various transport technologies, such as User Datagram Protocol (UDP), Transmission Control Protocol (TCP) sockets, and MQ Telemetry Transport (MQTT), to send data to the edge devices. Upon arrival in the federated domain, the data is grouped to match the same event, considering different sources and slightly different time instants. A pre-processing step accommodates the data for the spatial domain filter. The spatial filter processes the received data, leveraging topological and geometrical characteristics of the setting and pre-calibrated information related to technological errors to provide precise object positions. Optionally, the spatial filter can be complemented by a time-domain filter, which takes advantage of subsequent samples in time to discard unfeasible positions and improve asset localization, particularly under moving conditions. The final position estimate is stored in the database and made available through an API.

3 Implementation

The implementation of DUNE RTLS is described in Section 3.1. The implementation is divided into a pre-production and production environment. The pre-production scenario implemented is described in Section 3.2. It is used to develop and extensively test all the algorithms, also to collect data for training the DL models. Finally, the production scenario implemented is presented in Section 3.3.

3.1 RTLS Operation

The operation of DUNE during the project duration focused on developing accurate BLE-AoA estimation and provide a functional end-to-end demo. AoA-based positioning using BLE signals is a relevant cost-effective solution for the IoT. BLE technology is low-cost and widely available, while the AoA-based positioning solution is able to provide indoors sub-meter accuracy most of the time. Note then in Figure 3.2 that WiFi and RFID are not integrated yet in DUNE. In that sense, the above layers have been designed to be independent of the underlying technology to facilitate the integration of those positioning technologies in the immediate future.

The current implementation of DUNE is depicted in Figure 3.1. But leaving aside the far-edge and edge-cloud architecture division, the RTLS operation can be summarized in terms of operation blocks. We then use the building blocks depicted in Figure 3.2 to follow the next subsections division and explanations.

3.1.1 Reference Navigation Agent

The wireless asset tracked by the system in real time shown in Figure 3.2 is any emitter transmitting a BLE signal with the added CTE defined in the BLE 5.2 standard. For testing and developing purposes, we used what we call the Reference Navigation Agent shown in Figure 2.2 and Figure 3.1. It is a development and experimentation tool of DUNE which consists of a Turtlebot3 –a Robot Operating System (ROS) standard platform robot– plus a BLE transceiver board added on top of it. The Navigation Agent is designed to be remotely controlled to follow preconfigured or random paths for indoor positioning experimentation. Its sensed data is processed to obtain accurate indoor position estimations that serve as groundtruth labels for DL models to learn from. The emitted BLE or Wi-Fi RF signals by the agent during the experiment and received by DUNE's infrastructure will be used as data features to train the DL models jointly with the labels for AoA estimation.

The robot exact model is named Turtlebot3 Waffle Pi. It allows flexible, reproducible and rapid experimentation covering most IoT indoor positioning use cases. The battery was replaced by a Li-Po with greater capacity to extend the duration of the experiments. The ROS software stack runs on Ubuntu Server in the Raspberry Pi (RPI) of the TurtleBot3. Acting as a ROS slave, the robot communicates with the cloud to send sensor data and for remote control. Laser Imaging Detection and Ranging (LIDAR), camera and Inertial Measurement Unit (IMU) data are sent in ROS topics during the experiment, while the cloud just subscribes to the topics. The data is there processed and stored in a database using a modified ROS package to obtain indoor position estimations with an average accuracy of 10 ± 0.1 cm that serve as groundtruth for benchmarking the other approaches and technologies. DUNE's ROS package includes

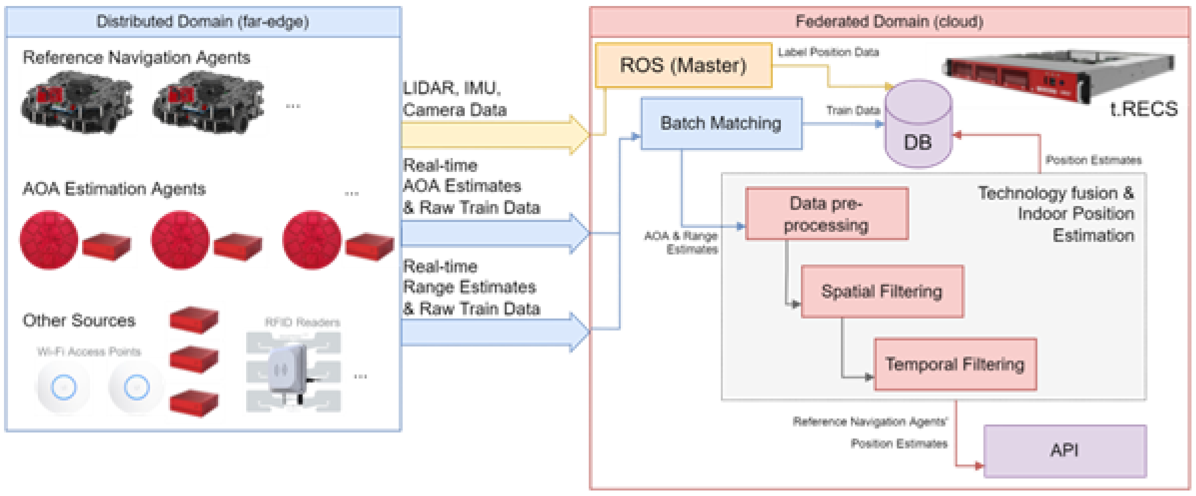


Figure 3.1. Current DUNE RTLS implementation.

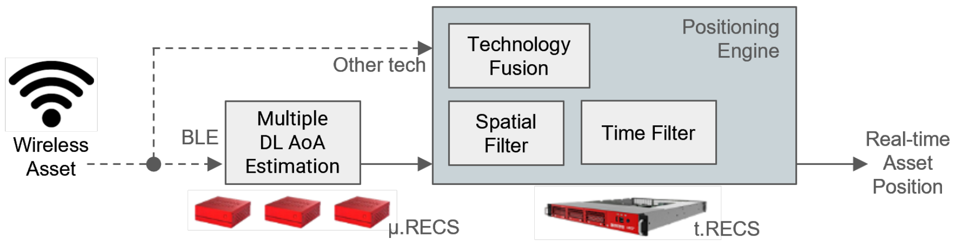


Figure 3.2. Operation summary of DUNE RTLS.

navigation in a known map and path-planning capabilities.

The Navigation Agent platform incorporates a plain TI's LAUNCHXL-CC2640R2 BLE board acting as a BLE transmitter. The board is attached to the TurtleBot3's platform and powered via the USB port of the RPI. LAUNCHXL-CC2640R2 was chosen as the development board for its flexibility, documentation, and ease of use. Any other BLE transmitter with AoA capabilities (BLE ≥ 5.1) can be used by DUNE's system without any additional change. As per the software side, the LAUNCHXL-CC2640R2 uses the BLE firmware from TI's SDK. The compatible TI's Software Development Kit (SDK) offers a fully qualified Bluetooth 5 protocol stack for single-mode BLE applications supporting high speed modes. It is configured to periodically transmit a BLE 5.2 packet with the CTE at the end that allows the AoA estimation at the receiver part.

3.1.2 AoA Estimation Agent

The multiple AoA estimation block shown in Figure 3.2 consists of all the AoA Estimation Agents shown in Figure 3.1 that translate the received BLE signal to angle estimates. These estimates are then feed to the Positioning Engine described in Section 3.1.2 to derive a position estimate.

The hardware and software specifics of the AoA Estimation Agent and how it interacts with the Reference Navigation Agent are shown in Figure 3.3 and Figure 3.4. Recall that these are far-edge components of DUNE. The AoA estimation procedure of each AoA Estimation Agent is explained next.

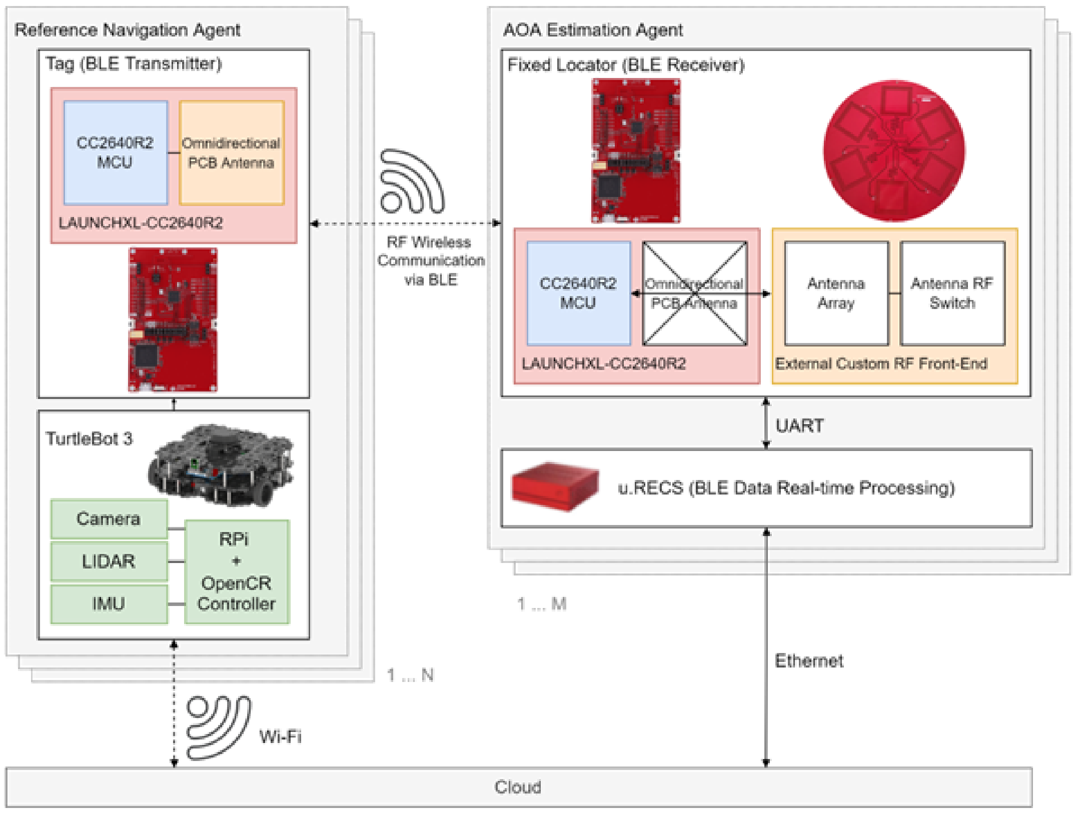


Figure 3.3. Hardware view of the far-edge BLE-AoA estimation of DUNE.

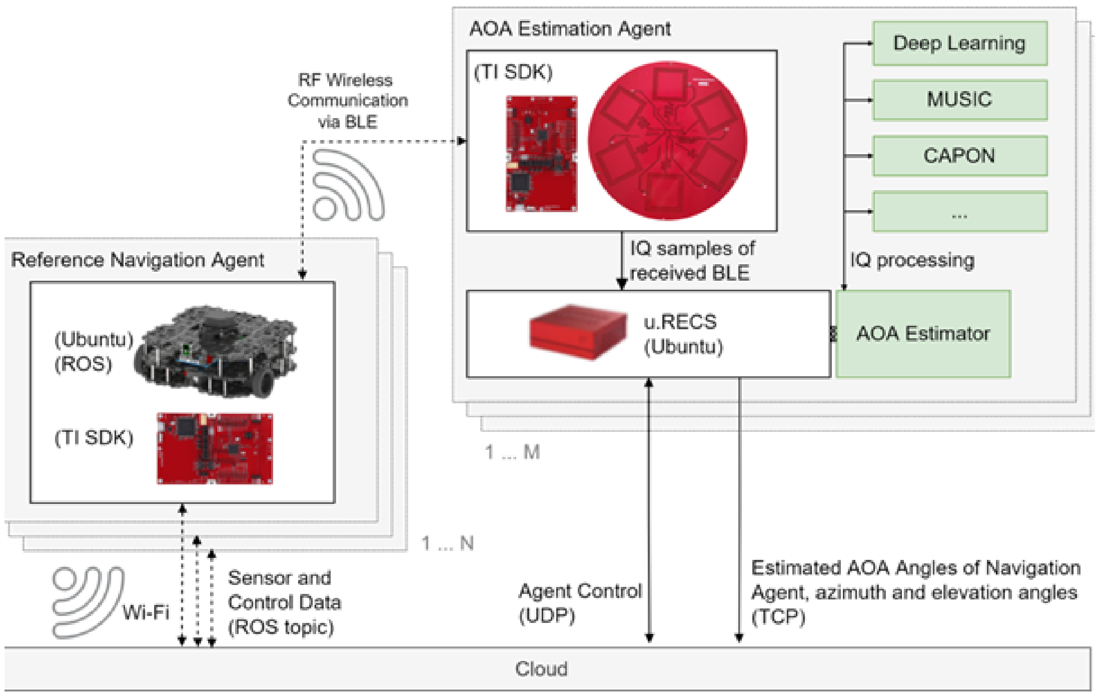

AOA Estimation Agent
(TI SDK)
Deep Learning
MUSIC
CAPON
...
IQ samples of received BLE
IQ processing
u.RECS (Ubuntu)
AOA Estimator
RF Wireless Communication via BLE
Reference Navigation Agent
(Ubuntu) (ROS)
(TI SDK)
1 ... M
1 ... N
Wi-Fi
Sensor and Control Data (ROS topic)
Agent Control (UDP)
Estimated AOA Angles of Navigation Agent, azimuth and elevation angles (TCP)
Cloud

Figure 3.4. Software view of the far-edge BLE-AoA estimation of DUNE.

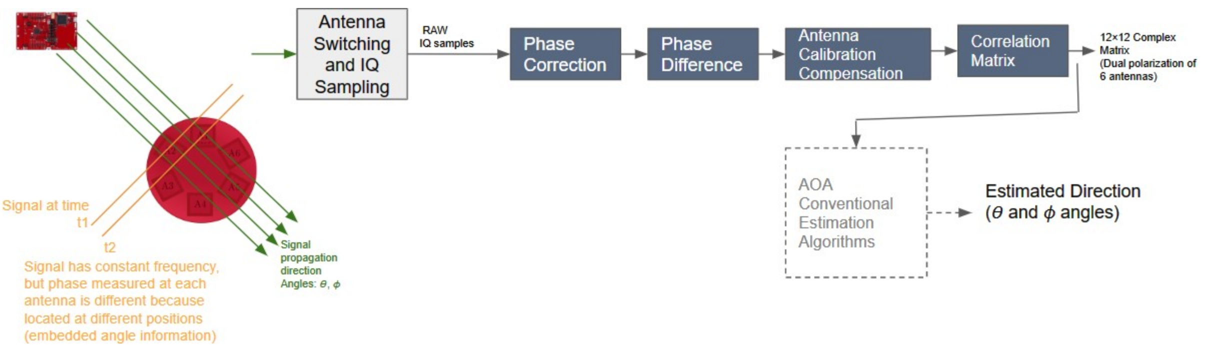

Antenna Switching and IQ Sampling
RAW IQ samples
Phase Correction
Phase Difference
Antenna Calibration Compensation
Correlation Matrix
12×12 Complex Matrix (Dual polarization of 6 antennas)
AOA Conventional Estimation Algorithms
Estimated Direction (θ and ϕ angles)
Signal at time t1
t2
Signal has constant frequency, but phase measured at each antenna is different because located at different positions (embedded angle information)
Signal propagation direction Angles: θ, ϕ

Figure 3.5. Block diagram of BLE-AoA estimation.

3.1.2.1 Baseline BLE-AoA Estimation

The BLE-AoA locator equips an antenna array that consists of 6 antennas, which are used to extract phase difference out of in-phase and quadrature components of the received BLE signals.

IQ sampling (also known as complex sampling or quadrature sampling) consists of digitally sampling the IQ components of the received signal, which combined define the whole signal. Antennas are controlled by an RF switch that accesses antennas sequentially. BLE employs a switch-then-sample mechanism to access the antenna array. Instead of sampling antenna elements at the same time, this mechanism reduces the required number of receivers to one. In each switch slot, the antenna switching is performed, while in the next sample slot the receiver samples the IQ components. Then, the AoA of received signals can be derived using the phase difference information that is created by the spatially separated antennas. Note that the impinging signal will reach each antenna at slightly different times, obtaining time and phase-delayed versions of the original signal. How the wave propagates and interacts with the fixed antenna elements can be mathematically modelled, thus, in some sense, estimation algorithms compare the sampled signal to many possible transmitted signals gone through the model. Finally, to allow phase difference calculation, BLE Direction Finding added in release 5.1 optionally adds the CTE at the end of a packet, which contains constant and un-whitened ones, that is, a single tone sine wave with a positive constant frequency deviation. As samples collected by the different antennas are not sampled in the same time instant, the constant frequency greatly simplifies the phase difference calculation.

Finally, the AoA of the impinging RF signal in 3D space is defined by two angles called azimuth θ and elevation φ. Following the conventional approach, we estimate the AoA first following the procedure of Figure 3.5 and then using the MUSIC signal processing algorithm [7] in the u.RECS. The specific MUSIC derivation that exploits the orthogonality between the signal and noise space can be found in Deliverable D3. This method is considered as the baseline.

3.1.2.2 DL Approach

The MUSIC algorithm and its derivates fail in the AoA estimation when non-idealities such as multipath effects or the antenna array imperfections are present [7, 8]. MUSIC requires the computation of an autocorrelation matrix and its eigendecomposition, which implies the computation of inverse matrices for each received signal.

Moreover, the steering vector definition is costly when dual-polarized antenna arrays are considered [9]. Efforts towards reducing the computation complexity have been recently proposed with the Propagator Direct Data Acquisition (PDDA) algorithm, which is only valid for linear and rectangular antenna arrays [10]. For that reason, we approached AoA estimation using DL. The adoption of DL models such as Fully-Connected Neural Networks (FCNN) and Convolutional Neural Networks (CNN) for AoA estimation is nowadays being assessed. Our state-of-the-art review can be found in Deliverable D3. The crucial point is that they only consume time during the training process, which usually is performed offline. Once trained, the DL models can provide an output whenever an input is provided by *simple* matrix multiplication. The required amount of time to obtain an estimation can be highly reduced using a trained DL model running in the efficient VEDLIoT u.RECS.

3.1.2.3 Implemented DL models

Seven different DL models were implemented and evaluated as AoA estimators in work package WP3 and Deliverable D3. The models were published as one of the outcomes of work package WP3 in https://bitbucket.org/wineuoc/aoa-ble-deeplearning. The proposed DL models consider feed-forward layers to allow their implementation in real time. Extensive details can be found in the *readme* file in the code repository. A summary of the implemented models is as follows:

- **FCNN-1day**: FCNN model shown in Figure 3.6. It was trained with measurements collected during a single day campaign. It consists of 16 layers. The input layer considers the following: the real and imaginary parts of the 45 raw IQ values, the average RSSI value of the IQ values and the one-hot encoding BLE channel number from which the signal was received. Note that this model skips all preprocessing steps of conventional estimation shown in Figure 3.5. The first Normalization layer standardize the input data towards zero mean and unit standard deviation. Four fully connected layers with 1024, 2048, 1024 and 512 hidden neurons are added after the normalization layer. Each one is accompanied by a Rectified Linear Unit (ReLU) non-linear activation function and a Dropout layer. The drop rate is set to 0.2 for each layer. Finally, two fully connected layers considering a single neuron and a linear activation function are in charge of providing (θ, φ) estimates. To avoid the over-fitting of the model, a L2 extra penalty equal to $5 \cdot 10^{-5}$ is considered as a regularization technique [11, Section 7.2].

- **FCNN-2days**: Same structure as FCNN-1day. Instead of a unique day, the model here is trained considering 2-day data. This is proposed to evaluate how the amount of trained data affects performance. Specifics of trained data can be found in Deliverable D3. The rest of the following models were all trained with 1-day data.

- **FCNN-R**: Same as FCNN-1day. The input, however, corresponds to the flattened estimated correlation matrix $\hat{\mathbf{R}}_\mathbf{x} \in \mathbb{C}^{12\times12}$ matrix derived in Deliverable D3. It consists of a vector of 288 real values.

- **CNN**: DL model shown in Figure 3.7 with 15 layers considering a CNN structure with its respective max pooling layer to process the correlation matrix $\hat{\mathbf{R}}_\mathbf{x}$. The input corresponds to two matrices of size (12×12). One matrix defines the real values of $\hat{\mathbf{R}}_\mathbf{x}$ whereas the other defines the imaginary part. The CNN layer is adopted to exploit the spatial relationships between samples by means of convolving slices of $\hat{\mathbf{R}}_\mathbf{x}$ with size 3×3 for the real and imaginary parts of $\hat{\mathbf{R}}_\mathbf{x}$. The Batch Normalization (BN) layer is adopted here to ease the convergence of the model. Then, the max pooling layer halves the output of the convolutional layer computing the maximum value of the pooling filter (2×2). The Flatten layer is in charge of transforming the 2-dimensional shaped outputs of the MaxPooling layer into a 1-dimensional shaped data. Later, these data are processed by fully connected layers. Besides, L2 extra penalty of value $5 \cdot 10^{-5}$ and Dropout are considered as regularization techniques. All the layers adopt the ReLU activation function, except for the output layers, which considers the linear one.

- **CNN-NS**: DL model derived from the CNN one shown in Figure 3.8. The novelty here is placed in the convolutional part of the model, where it considers two branches instead of a unique one. The former processes the $\hat{\mathbf{R}}_\mathbf{x}$ while the latter is in charge of the noise space $\mathbf{Q}_n$. In that way, the model processes the same information as the MUSIC algorithm. All the layers adopt the ReLU activation function, except for the output layers, which considers the linear one.

- **CNN-NS-IQ**: Derivation of the CNN-NS model considering three branches shown in Figure 3.9. The first one processes the $\hat{\mathbf{R}}_\mathbf{x}$, the second is in charge of $\mathbf{Q}_\mathbf{n}$, and the last one of the raw IQ samples. In that way, information regarding the raw samples, the covariance matrix and the noise space are processed simultaneously. The branches in charge of $\hat{\mathbf{R}}_\mathbf{x}$ and $\mathbf{Q}_\mathbf{n}$ follow the same structure as in CNN-NS model. The branch in charge of the IQ measurements considers 6 layers consisting of two fully connected layers with 1024 hidden neurons each, two ReLU layers and two DropOut layers with a drop rate of 0.2. Then, the output of each branch is concatenated to feed the fully connected layers.

- **FCNN-SP**: DL model processing the IQ samples and signal space $\mathbf{Q}_\mathbf{s}$ as inputs shown in Figure 3.10. It considers two branches in the input side to process the different inputs. Both consider the same structure: two fully connected layers of 1024 and 2048 hidden neurons. Each layer is accompanied by its ReLU activation function and DropOut layer. The outputs of each branch are concatenated and processed by a unique fully connected layer of size 1024 hidden neurons. As regularization techniques, this model considers the same L2 and early stopping techniques as the previously defined models.

3.1.3 Positioning Engine

The Positioning Engine (PE) was developed during work package WP4 and presented in Deliverable D4. PE processes the AoA and range estimates computed in the far-edge to derive a position estimate of the asset.

The far-edge estimates suffer certain variability, which depends on the scenario conditions, RF medium and the underlying technology and estimation algorithms. This variability translates into errors in the estimation of the position in the Cartesian plane. Obviously, this influences the performance of the system and impacts applications that require fine tracking of assets and accurate and stable positioning over time. By exploiting spatial and temporal diversity, the PE improves the performance of the localization system, mitigating the fading, interference and multipath effects

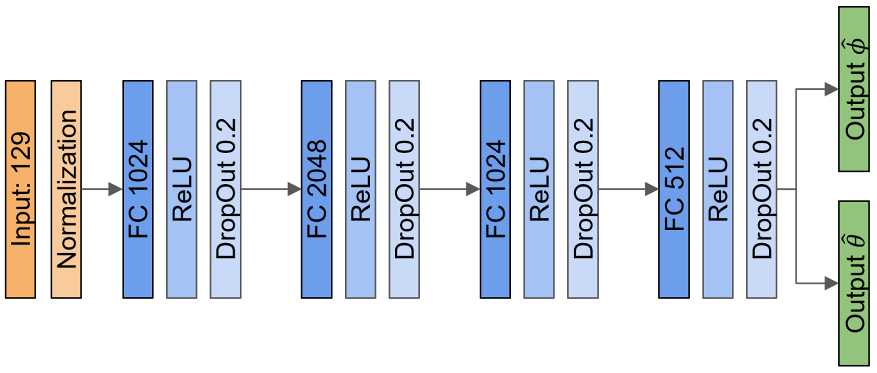


Figure 3.6. FCNN model.

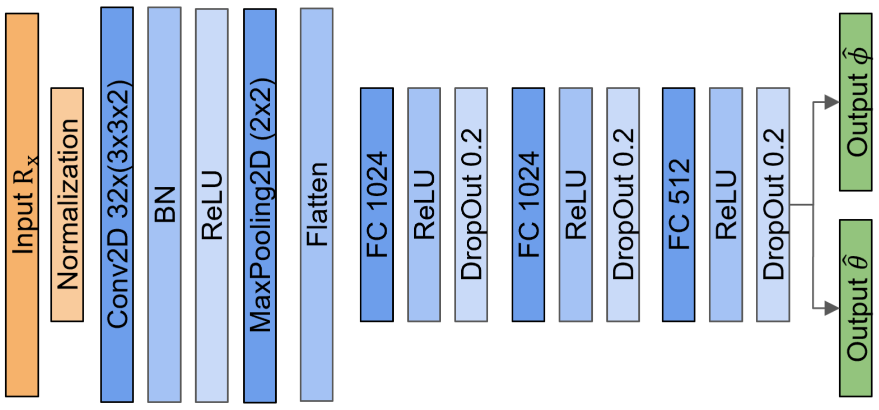


Figure 3.7. CNN model.

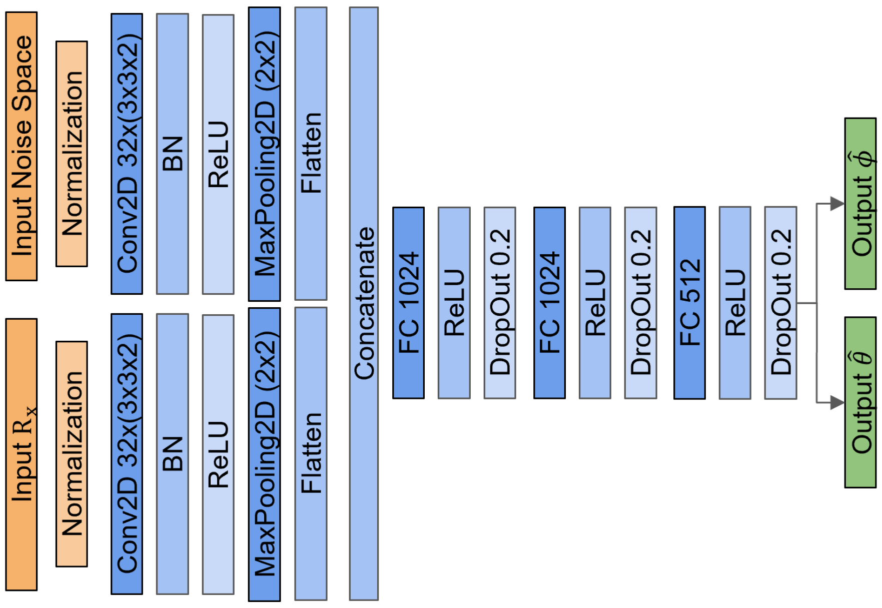


Figure 3.8. CNN-NS model.

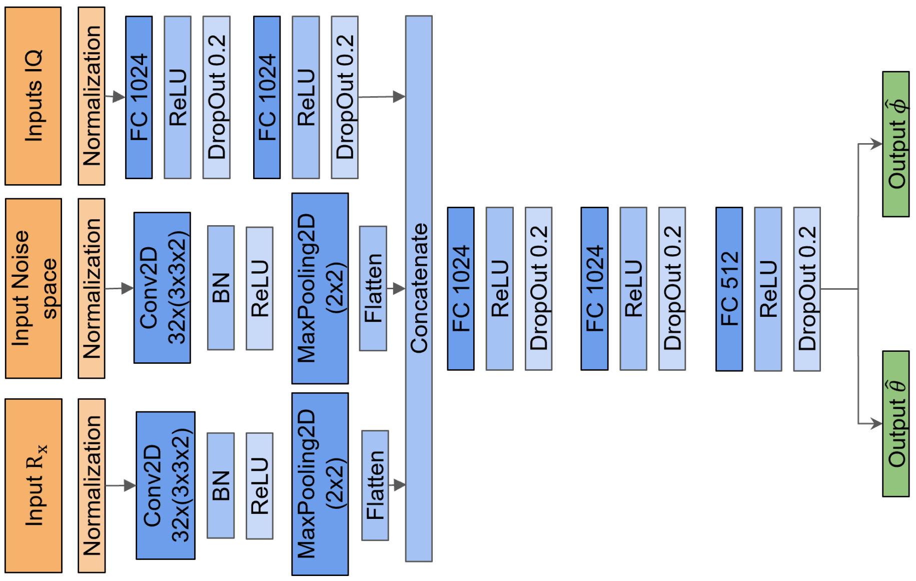


Figure 3.9. CNN-NS-IQ model.

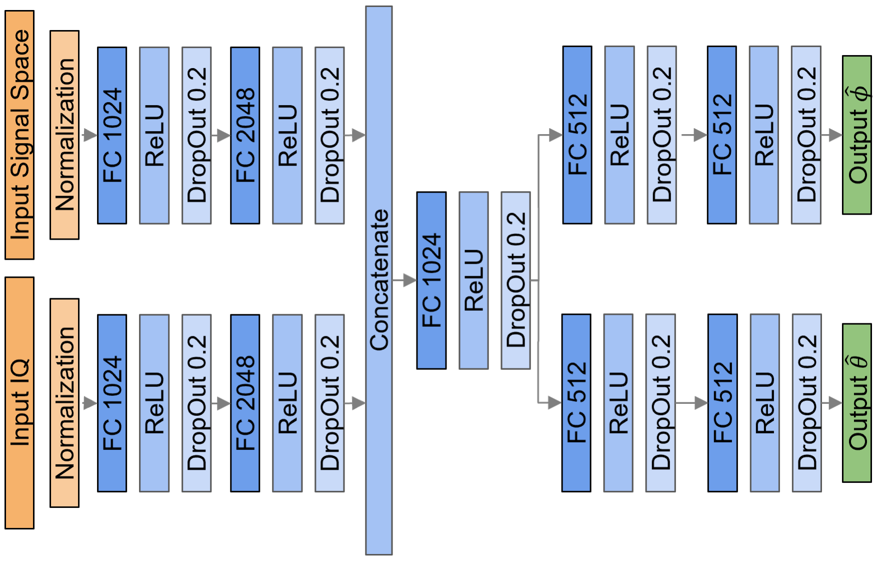


Figure 3.10. FCNN-SP model.

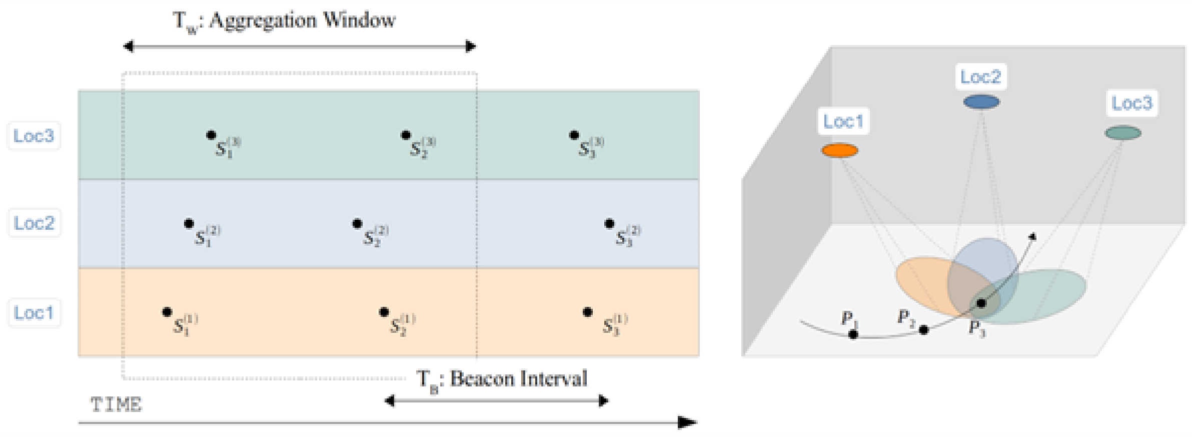


Figure 3.11. Discrete sampling. The filtering tools need to wait for a period of time T_w in order to exploit the same BLE packet received by different locators.

of signal wireless propagation. The exploit of information is done in real-time using the t.RECS hardware.

3.1.3.1 Real-time Implementation

Some considerations need to be addressed to run the PE in real time. These are handled by the Batch Matching and Data pre-processing blocks in Figure 3.1. The outputs of the AoA Estimation Agents Z_n need to be synchronized with the subsequent stage of the localization tool. A single emitted signal will be received almost simultaneously by the locators of each AoA Estimation Agent deployed. Due to the implementation restrictions and the serial operation of the processes, a small aggregation window T_w is needed to ensure that the time domain and spatial domain filters process at the same time the packet replicas of the emitter captured and processed by the different agents. Figure 3.11 exemplifies the case in which 3 BLE-AoA locators receive a BLE packet from the same emitter, where T_w is the minimum time window required to receive all the packet instances, $S_1^{(3)}$ denotes the BLE packet number one received by locator number 3 and T_B is the BLE packet interval (how often the emitter transmits).

After each time window T_w, the PE processes the received data in order to estimate the asset's position. When this T_w is large enough, more than one sample per emitter can be received during the same period and may be averaged. This can be beneficial or not depending on how fast the asset is moving, so T_w must be tuned.

3.1.3.2 Baseline AoA-based Position Estimation

To understand the non-linear relationship between AoA and position, we must define the positioning system and position solution with respect to the AoA based on Figure 3.12. Thus, let:

- $\mathbf{p} \triangleq [x_t, y_t, z_t]^\mathsf{T} \in \mathbb{R}^3$ denote the true position of the mobile asset in a Cartesian coordinate system.
- $A_n = \{\mathbf{s}_1, \ldots, \mathbf{s}_n\}$ be the set of $n \geq 1$ locator nodes, where the $\mathbf{s}_i \triangleq [x_i, y_i, z_i]^\mathsf{T} \in \mathbb{R}^3$ stand for their absolute fixed positions.
- $\boldsymbol{\theta}_n = \{\theta_1, \ldots, \theta_n\}$ and $\boldsymbol{\varphi}_n = \{\varphi_1, \ldots, \varphi_n\}$ be the respective sets of true azimuths and elevations describing the AoA of the BLE wireless signal to each of the n

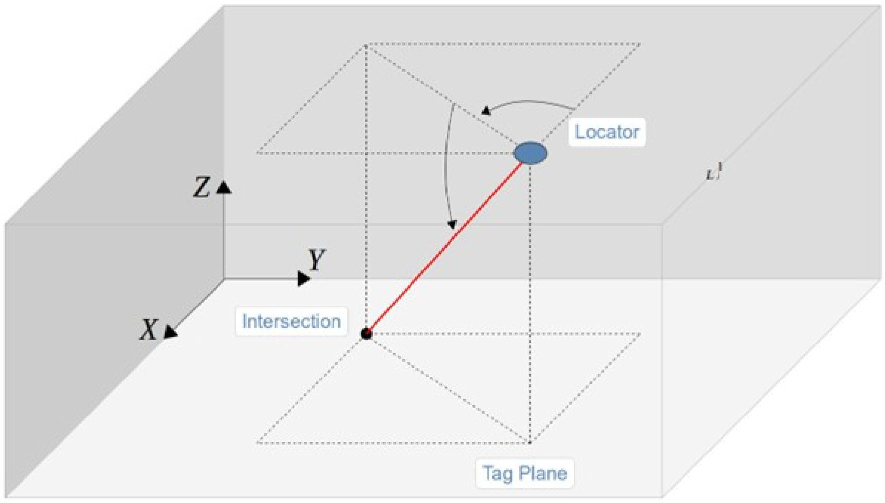


Figure 3.12. The geometry definition between the interaction of a BLE-AoA locator and a mobile asset (black circle) in an AoA-based indoor localization system. The azimuth angle θ of the vector locator-asset is the angle between the x-axis and the orthogonal projection of the vector onto the xy-plane. The angle is positive in going from the x-axis toward the y-axis. The elevation angle φ is the angle between the vector and its orthogonal projection onto the xy-plane. The angle is positive when going toward the positive z-axis from the xy-plane.

locators.

Then, the true azimuth and elevation angle pair can be expressed geometrically in terms of the corresponding asset position as

$$\theta_i = h_i^{\theta}(\mathbf{p}) = \text{arctan2}\left(y_t - y_i, x_t - x_i\right), \tag{3.1}$$

$$\varphi_i = h_i^{\varphi}(\mathbf{p}) = \arcsin\left(\frac{z_t - z_i}{d_i}\right), \tag{3.2}$$

where $d_i \triangleq \|\mathbf{p} - \mathbf{s}_i\|_2 > 0$ is the Euclidean distance between the locator and the asset. We choose $[\theta_i, \varphi_i]^{\mathsf{T}} \in \Omega \triangleq [-\pi, \pi) \times \left[-\frac{\pi}{2}, 0\right]$, which constrains φ to an indoor setting and avoids the indetermination of θ [12].

Let $\mathbf{z}_i \triangleq [\hat{\theta}_i, \hat{\varphi}_i]^{\mathsf{T}} \in \Omega$ be the AoA observed from the i-th locator, where $\hat{\theta}_i$ and $\hat{\varphi}_i$ refer to the respective measurements of the azimuth and elevation angles. We write the observation model of the localization system in vector form as

$$\mathbf{z}_i = \mathbf{h}_i(\mathbf{p}) + \mathbf{w}_i \tag{3.3}$$

with $\mathbf{h}_i(\mathbf{p}) \triangleq [h_i^{\theta}(\mathbf{p}), h_i^{\varphi}(\mathbf{p})]^{\mathsf{T}}$, and $\mathbf{w}_i \in \mathbb{R}^2$ a noise vector modeling the ensemble of unfavorable factors.

We now define the estimator $\hat{\mathbf{p}}$ of the asset position as

$$\hat{\mathbf{p}} = f\left(Z_n; A_n\right), \tag{3.4}$$

where the function $f : \Omega^n \mapsto \mathbb{R}^3$ maps a set $Z_n = \{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$ of AoA measured from the fixed-locator nodes A_n to an estimate of the position of the mobile asset.

The goal of a localization system is to derive a function f that maximizes the accuracy of $\hat{\mathbf{p}}$, i.e., minimizes the distance between $\hat{\mathbf{p}}$ and $\mathbf{p}$. However, the non-linear inversions in (3.1) and (3.2) –which are required to estimate **p** from the data Z_n via (3.3)– often amplify measurement noise and lead to estimates that are biased and that do not achieve minimum variance for small sample sizes [13]. Moreover, the computational complexity of these approaches hampers their application to real-time applications.

The implemented baseline estimator $\hat{\mathbf{p}}$ considering a single locator $n = 1$ is shown in Figure 3.12. The position estimate is calculated as the intersection between the red line and the height plane where the asset is assumed to be, where the red line is defined by the vector starting at the fixed locator position with its direction given by the estimated AoA. This is the only valid method when considering AoA information from only a single locator. Note that in most IoT applications, the asset is known to lie on a surface so that its altitude does not have to be estimated, e.g., tracking a shopping cart inside a supermarket. Multi-locator solutions implemented are described in Section 3.1.3.3.

There are multiple ways to characterize the positioning accuracy of a localization system. System-level metrics and key performance indicators used by DUNE are the Cumulative Distribution Function (CDF) of the error and the Circular Error Probability (CEP) (CEP):

$$\mathrm{CDF}(\;)\ \text{t:,}\ \mathrm{P}\,(I\mathbf{p} - \hat{\mathbf{p}}\,I_2 \le \;)\,, \tag{3.5}$$

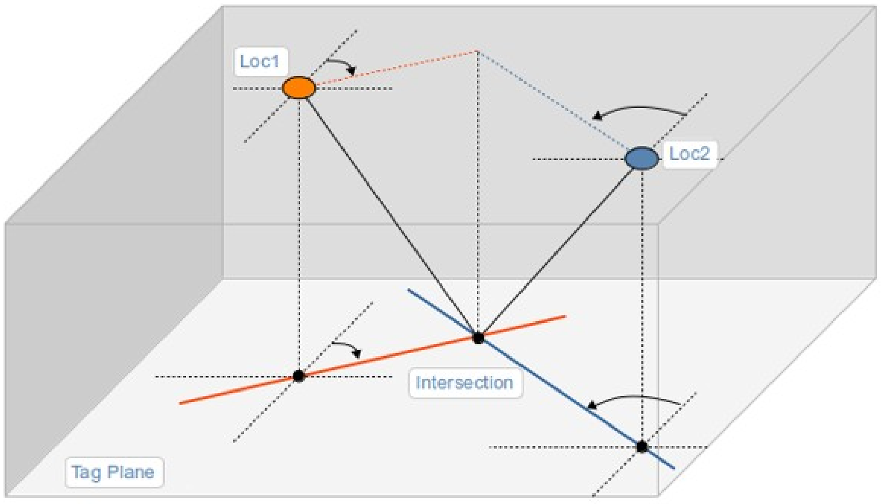
$$\mathrm{CEP}_{95}\ \text{t:,}\ \{\rho \mid \mathrm{P}\,(I\mathbf{p} - \hat{\mathbf{p}}\,I_2 \le \rho) = 0.95\}\,. \tag{3.6}$$

3.1.3.3 Spatial Filtering

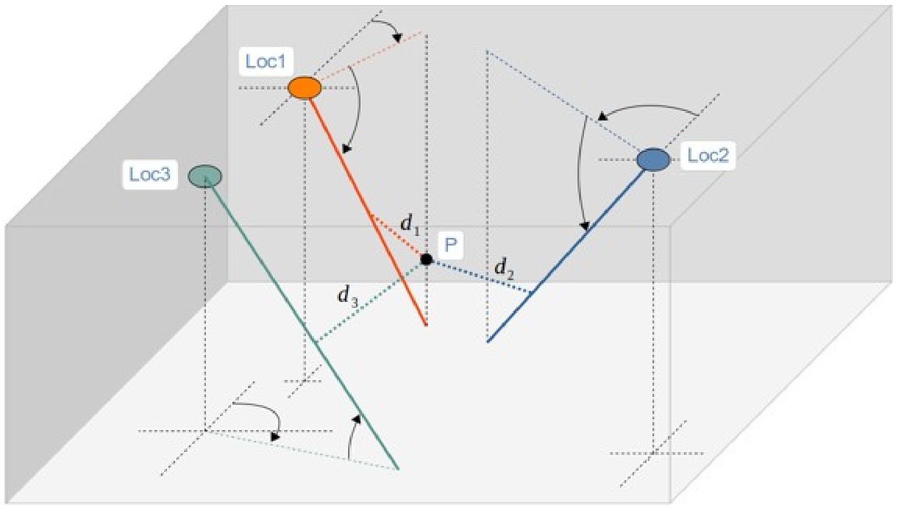
In practice, noise effects and deployment geometry often dictate the need for more than two locators to improve accuracy. The spatial domain filtering here exploits redundancy in the number of locators $n > 1$ to improve the positioning accuracy. So, DUNE deploys multiple AoA Estimation Agents to combine their information to overcome the limited performance of a single locator. Figure 3.13 shows the implemented least-squares solution which finds the closest position point to all the individual estimations.

We evaluated the performance of the different spatial filters based on the fusion of position estimations. Details were presented in Deliverable D4 jointly with a review of the state-of-the-art. For more than two locators, noise and finite angle resolution cause position estimation ambiguities that can only be conciliated through heuristic methods and multiangulation techniques. In our observations these methods often led to inaccurate results due to the impact of outlier measurements and the farther locators which contributed with very large position errors. The underperforming results achieved with plain aggregation lead us to evaluate locator selection mechanisms. The idea behind anchor selection is to select those locators that provide the more accurate estimates and discard the others. To that aim, several techniques based on RSSI, elevation angle and the pseudo-spectrum magnitude of the AoA estimation algorithm were also implemented and evaluated.

The current best performing spatial filter implemented is based on a weighted least squares solution with weights derived using the RSSI received. The RSSI is an available metric which gives a sense of proximity. The closer the asset is to the locator the best, as the errors from AoA estimates nonlinearly propagate to position errors. Therefore, the implemented method discards the worst locators' data based on the



(a) BLE-AoA locator 1 and 2 2D solution (labeled Intersection) using azimuth angles.



(b) BLE-AoA locator 1, 2 and 3 3D solution (labeled P) using azimuth and elevation angles.

Figure 3.13. The least-squares solution given a set of AoA estimates from different locators AoA Estimation Agents.

RSSI (the lowest RSSI) and determines the weighted position with the remaining measures from the other locators.

3.1.3.4 Technology Fusion

The first version of DUNE RTLS considers only BLE-based positioning with RSSI and AoA. The main idea behind the fusion algorithm is depicted in Figure 3.14. DUNE deploys BLE-RSSI locators in strategic areas such as meeting rooms where no accurate localization is needed. These locators are much cheaper but only detect the asset in proximity through the RSSI value. A DL model was trained that accurately classifies the asset within a 5-meter radius of the locator. BLE-RSSI-based localization can not be more accurate, however this information is also useful when combined with the more accurate BLE-AoA-based estimates. Figure 3.14 shows that two pairs of BLE-AoA locators disagree in the estimate of the position, but the problem is resolved thanks to the BLE-RSSI locator proximity estimate. The integration of other infor-

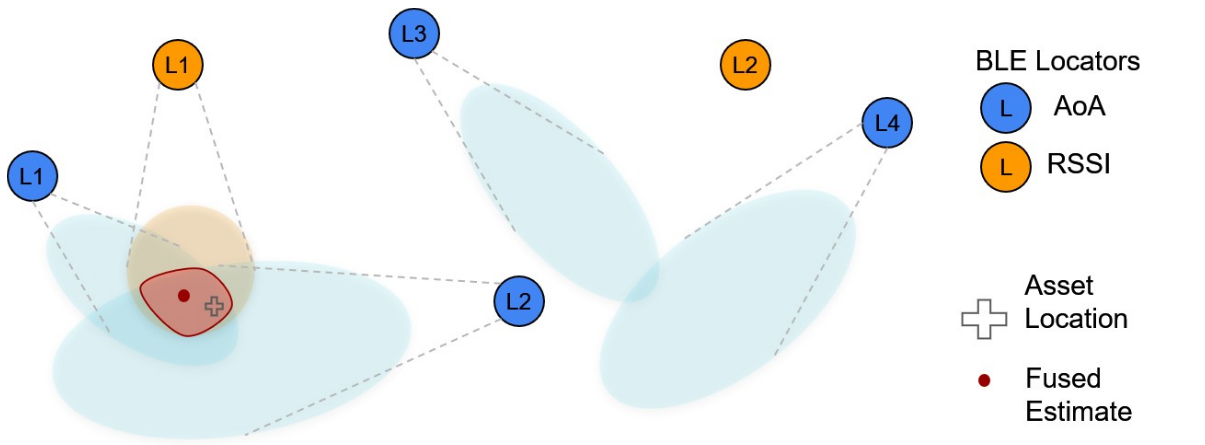


Figure 3.14. Information fusion algorithm idea. In this scenario, L1 and L2 estimations place the asset's position on the left side, whereas L3 and L4 estimations indicate the right side. Since L3 and L4 are located further away from the asset, they are more susceptible to estimation errors. Unfortunately, in practice, there is no way to determine this inherent bias. However, this challenge is effectively addressed by deploying cheaper RSSI locators strategically. When L1 detects the asset in proximity, it resolves the ambiguity, allowing for a valid and reliable estimate to be derived.

mation coming from other technologies is handled by the Technology Fusion block shown in Figure 3.2 and t.RECS hardware, which may be integrated in the Spatial Filtering process. Note that the ellipses shown in Figure 3.14 represent the 1-σ uncertainty associated to the position estimate.

It is worth mentioning that DUNE envisaged positioning service aims to provide an adaptable solution to the cost and accuracy required by the costumer. That is, if BLE accuracy is not enough for a specific application, DUNE can integrate other wireless technologies to increase the accuracy such as WiFi, RFID and UWB. Obviously, this has an extra cost that may not suit all customers, so there is a clear tradeoff. Therefore, future work involves the development of advanced algorithms using DL to fusion information from the different sources.

3.1.3.5 Time Filtering

Finally, time domain filtering exploits temporal redundancy to mitigate the impact of channel conditions or noise. These methods are used after the spatial domain filtering to achieve further performance. During the project, different methods have been implemented and evaluated:

- Simple Moving Average (SMA),
- Exponential Weighted Moving Average (EWMA) and
- Kalman Filter (KF).

The best-performing method in this context is the well-known KF [14]. This filter utilizes a series of measurements over time, taking into account statistical noise and other inaccuracies to produce more accurate estimates of the position compared to those based on a single measurement alone. It achieves this by estimating a joint

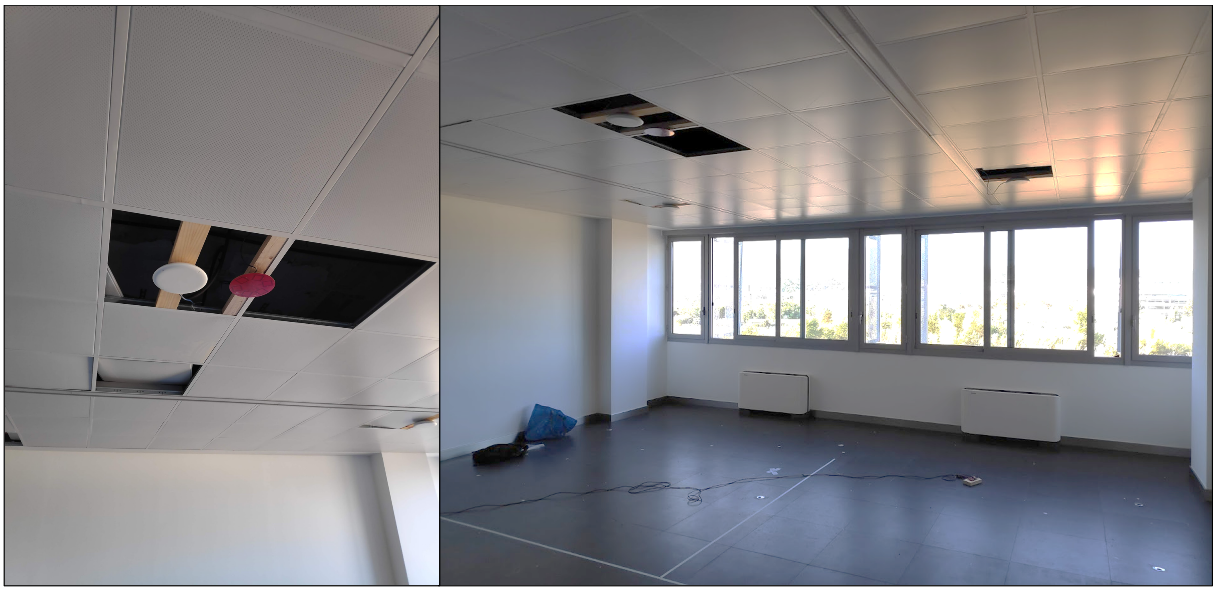


Figure 3.15. Pre-production RTLS with BLE-AoA locator and WiFi access points.

probability distribution over the variables for each timeframe. The KF algorithm operates through a two-phase process. During the prediction phase, it generates estimates of the current position along with their associated uncertainties. Subsequently, when the next measurement is observed, these estimates are updated using a statistical weighted average, with the weight being assigned based on a balance between the new measurement and the motion model's projection. The developed filter implements both first and second-order KF, with the option to assume constant velocity or constant acceleration models. It effectively tracks x and y positions, speed, and acceleration. A first-order Kalman Filter may offer good performance when objects are in static or very slow-moving conditions. On the other hand, the second-order filter is more suitable for accurately tracking moving objects.

3.2 Pre-production RTLS

The pre-production environment is DUNE's controlled scenario where algorithms are developed and extensively tested. This section describes the pre-production scenario and the datasets obtained and published.

The pre-production RTLS is deployed in a common business office room of dimensions 7.2 × 8.5 × 3 meters. The ceiling its made of metal plates while the walls are plasterboard, except one that it is completely covered by large windows. The room has furniture in it such as chairs, tables, laptops and a locker that are moved around for evaluation. WiFi access points and other Bluetooth devices transmit in the 2.4 GHz band from adjacent services in other rooms. Three to four anchors were deployed at different positions to understand their influence to AoA and position estimation. Figure 3.15 shows BLE-AoA locators and WiFi access points mounted on the ceiling of the pre-production scenario.

Five different datasets of raw BLE measurements were collected for training DL models and evaluating purposes. The datasets were published as one of the outcomes of work package WP3 in https://bitbucket.org/wineuoc/aoa-ble-dataset. In the repository, the reader will find the measurements obtained from several experiments in the considered scenario. A detailed explanation of the scenario, example code scripts, and the data format are also included in a *readme* file. This information is omitted in this deliverable to avoid clutter.

3.3 Production RTLS

This section describes DUNE RTLS as a permanent deployment over a real scenario in the first floor of the main headquarters of IN3 Research Hub with dimensions 55 × 22 × 3 meters. This deployment is the main outcome of work package WP4. The layout and positions of the hardware deployed over the production scenario is shown in Figure 3.17. The scenario considers two well-differentiated areas: (i) the left side, which consists mainly in the social area designed for social interaction and meetings between colleagues, and (ii) the right side, which is used for workers to sit and work using their laptop. In it there are fixed desks with PC screens and power sources. So most of the time people are not moving and when moving their dynamics are usually limited. The deployment considers six locators, which are attached to the ceiling of the office with specific 3D printed attachments.

Since the dynamics of people in the area are different, we have focused the efforts in the social area. Therefore, we increased the locator density on left side because it corresponds to the area where people are more active and higher interaction is expected. Recall that the asset tracked by the system in real time is any emitter transmitting a BLE signal with the CTE defined in the BLE 5.2 standard. For example, a smartphone, a low-cost and tiny BLE tag. If the emitter integrates other wireless technologies such as WiFi or RFID, DUNE RTLS will process in the immediate future the additional signals to derive a more accurate position. Finally, Figure 3.16 shows an actual picture of the area showing only 4 of the 6 locators deployed.

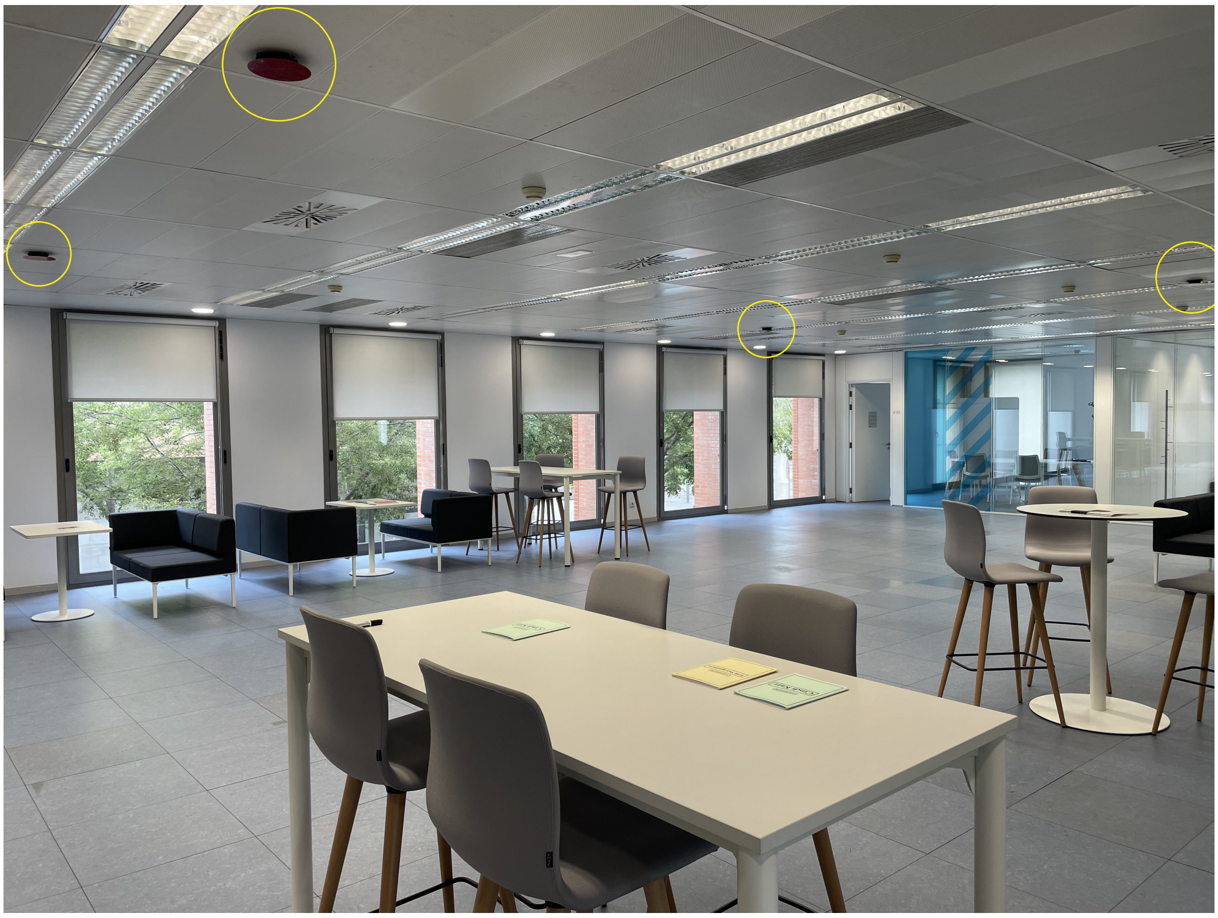


Figure 3.16. Social area covered by four AoA Estimation Agents highlighted in yellow.

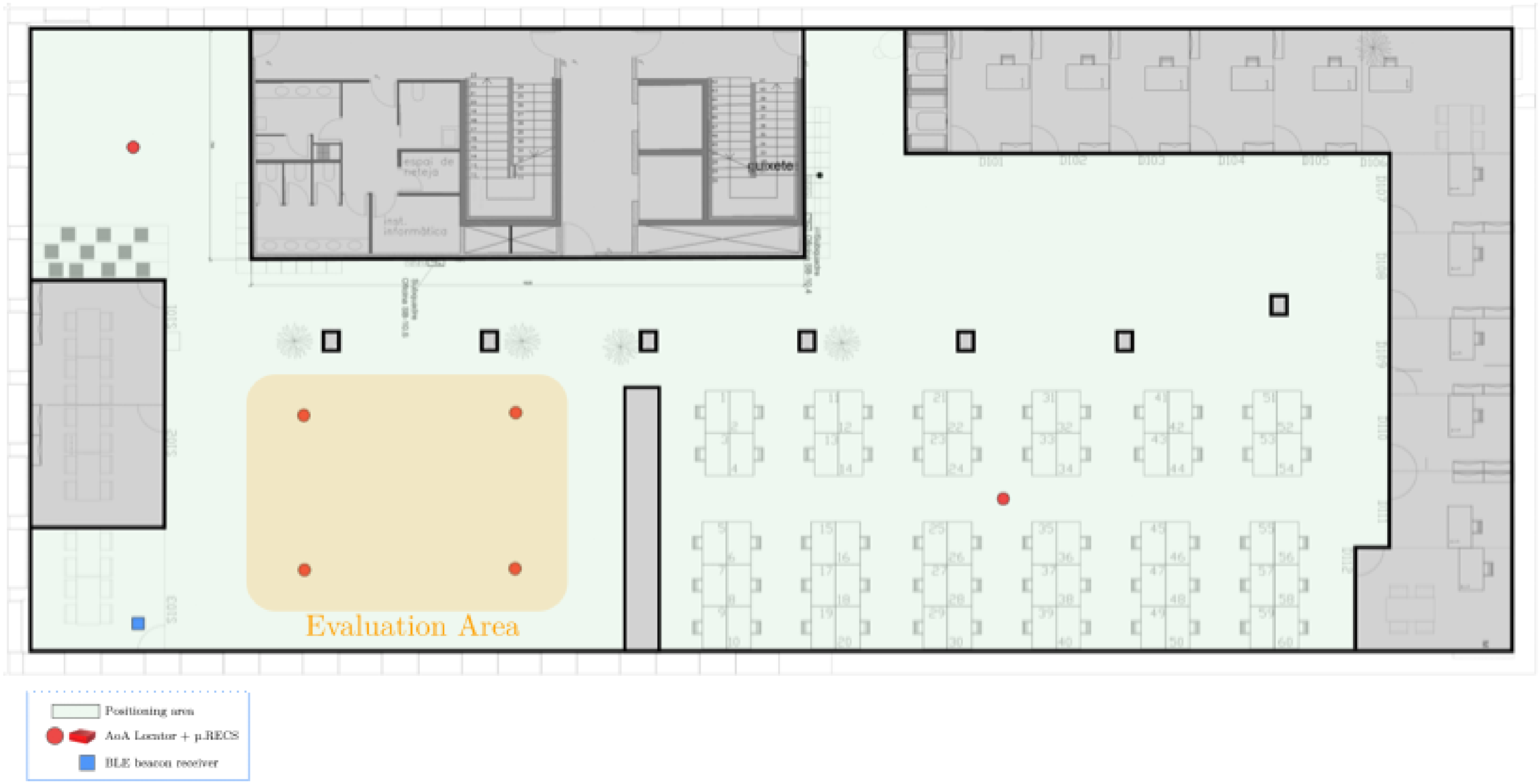


Figure 3.17. DUNE RTLS production deployment in IN3 Research Hub.

4 Evaluation and Characterization

This chapter presents the evaluation and characterization of the implemented DL models for AoA estimation and the performance of the positioning solutions of DUNE RTLS, which were described in Chapter 3. The results are divided into the outcomes of work package WP3 and WP4, which respectively correspond to the AoA Estimation Section 4.1 and the Position Estimation Section 4.2.

4.1 AoA Estimation

To evaluate the implemented DL models we first present the performance of the baseline algorithm.

Three different datasets of the preproduction scenario were processed with the MUSIC algorithm, obtaining the results shown in Figure 4.1. For the azimuth angle, the percentiles 90 of the errors correspond to 94.94, 91.31, and 24.00, for the datasets from days 11/10/22, 19/10/22 and 26/09/22, respectively. For the elevation angle, these percentiles equal to 35.29, 37.19, and 25.54. The bias of both angle estimates was close to 0º. However, large errors were found, mainly due to the multi-path problem. As the signal traveling from the emitter to the receiver can take several paths bouncing off walls or objects, it may happen that the direct signal path is received by the locator with less energy than the non-direct signal components. The signals can be added in the receiver in a constructive way according to their phase that depends on the distance they travel, which is different for each multi-path component. When this happens, the MUSIC algorithm does not estimate the AoA of the direct signal component, which is a well-known problem.

4.1.1 Accuracy

In terms of DL models trained with 19/10/22 train split data, the performance depicted in Figure 4.2 shows that all the implemented models are able to correctly estimate the angles against the test split of 19/10/22 data. The errors in the CDF were computed as the absolute minimum difference between the actual and the estimated angle. This shows that the models generalize with measurements under the same conditions of the day they were trained on. All the DL models were trained with the train split data of 19/10/22 and outperform the MUSIC algorithm.

However, errors shown in the CDF are too low, suggesting that models overfitted the data despite the techniques used to avoid this during training. The key point to assess is the generalization of the model when other conditions are observed. For example, when data gathered from an unseen day is fed to the DL models. To that aim, the performance of the models were tested considering the test split data from the other days 26/09/22 and 11/10/22.

Figure 4.3 shows the CDF of the angle errors for the implemented DL models against the test split data of 11/10/22. Note that all models were trained with data from 19/10/22, but only the model labeled as 2days was trained with the train split data of 11/10/22. In this case, the performance of all models expect FCNN-2days decreases. This means that DL models overfitted the training data but still outperform MUSIC when considering the variability in measurements from day to day. The similar per-

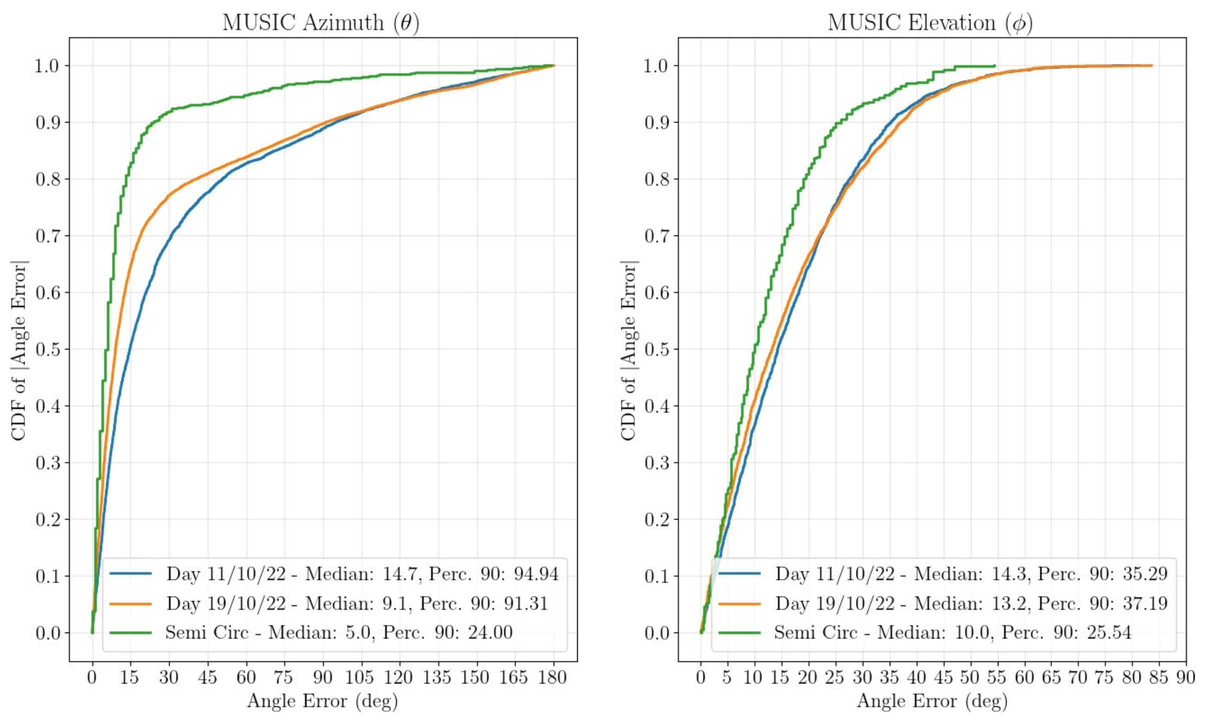


Figure 4.1. AoA estimation performance of baseline MUSIC algorithm.

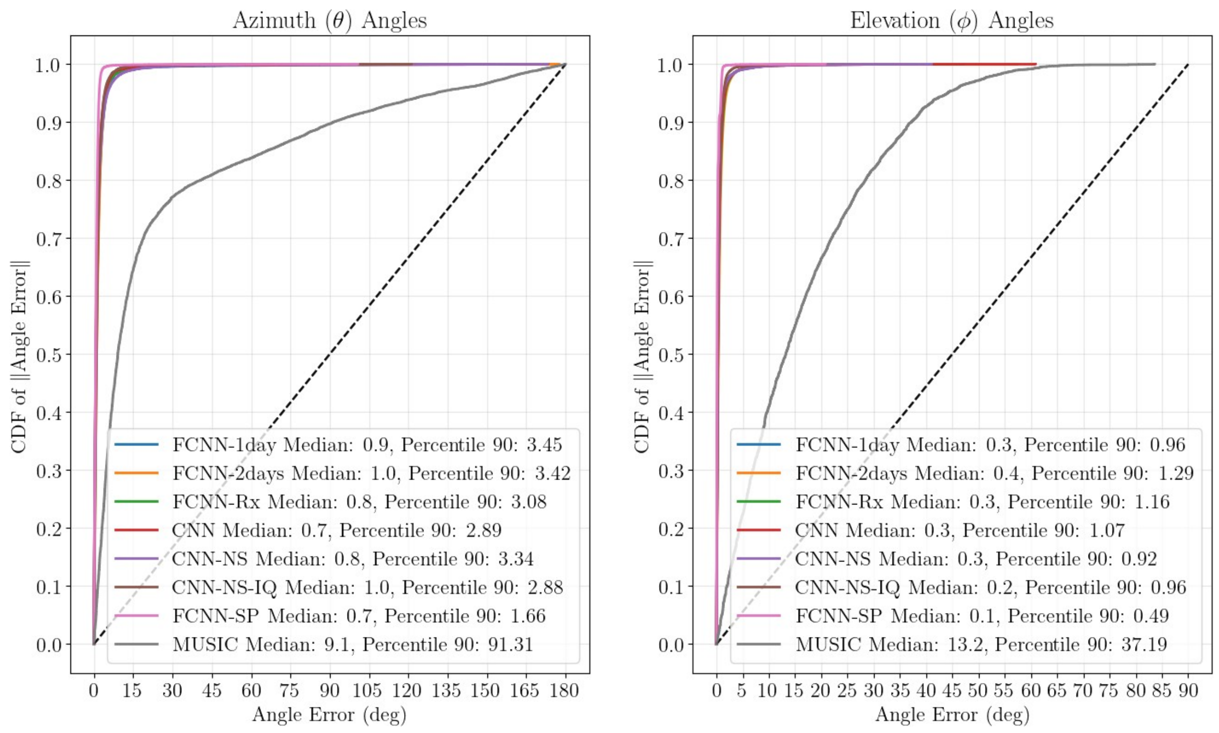


Figure 4.2. AoA estimation performance of DL models. Test split data of 19/10/22.

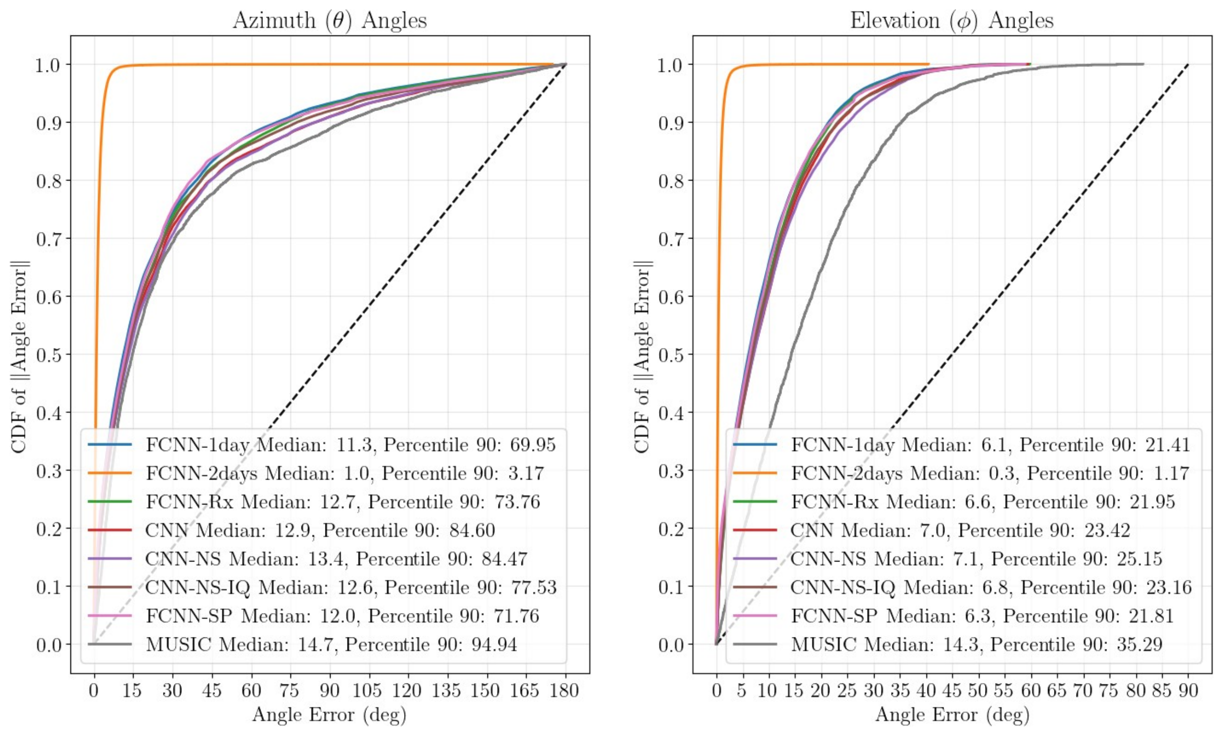


Figure 4.3. AoA estimation performance of DL models. Test split data of 11/10/22.

formance of FCNN-2days in Figure 4.3 compared to Figure 4.2 confirms that models overfitted, as it is the only model that has seen data during trained obtained in the same day of the test split.

Similarly, Figure 4.4 shows the CDF of the angle errors for the implemented DL models against the test split data of 26/09/22. None of the models have seen data from this day during training. Now, DL model performance is much similar to MUSIC but slightly higher.

Next, we explored how the amount of data the models are trained with influence the performance. For that reason, the DL models have been trained incrementally considering multiple day datasets. We labeled them as 1-day for 19/10/22, 2days for 11/10/22 and 19/10/22, 3days for 11/10/22, 19/10/22 and 01/03/23, and 4days for 11/10/22, 19/10/22, 01/03/23 and 09/03/23. The results are shown in Figure 4.5. It can be observed from the reported CDFs that the higher the amount of data for the training process, the better the performance. Besides, this also entails that a model trained with measurements from different days is more robust to changes in the environment, i.e., has better generalization.

Taking all this into account, Figure 4.6 the performance when all DL models are trained with 4 days of data and evaluated against unseen data from another day. In terms of each model, results show that the best performing is the FCNN-SP. The median of the errors in the estimates for azimuth and elevation angles is equal to 6.6 and 6.8, respectively. Moreover, the percentile 90 shows that the 90% of errors in the θ and φ estimates are not higher than 20.45º and 17.17º, respectively.

As a summary and in comparison with the state-of-the-art works, the proposed models demonstrate superior performance in estimating the AoA when compared to the

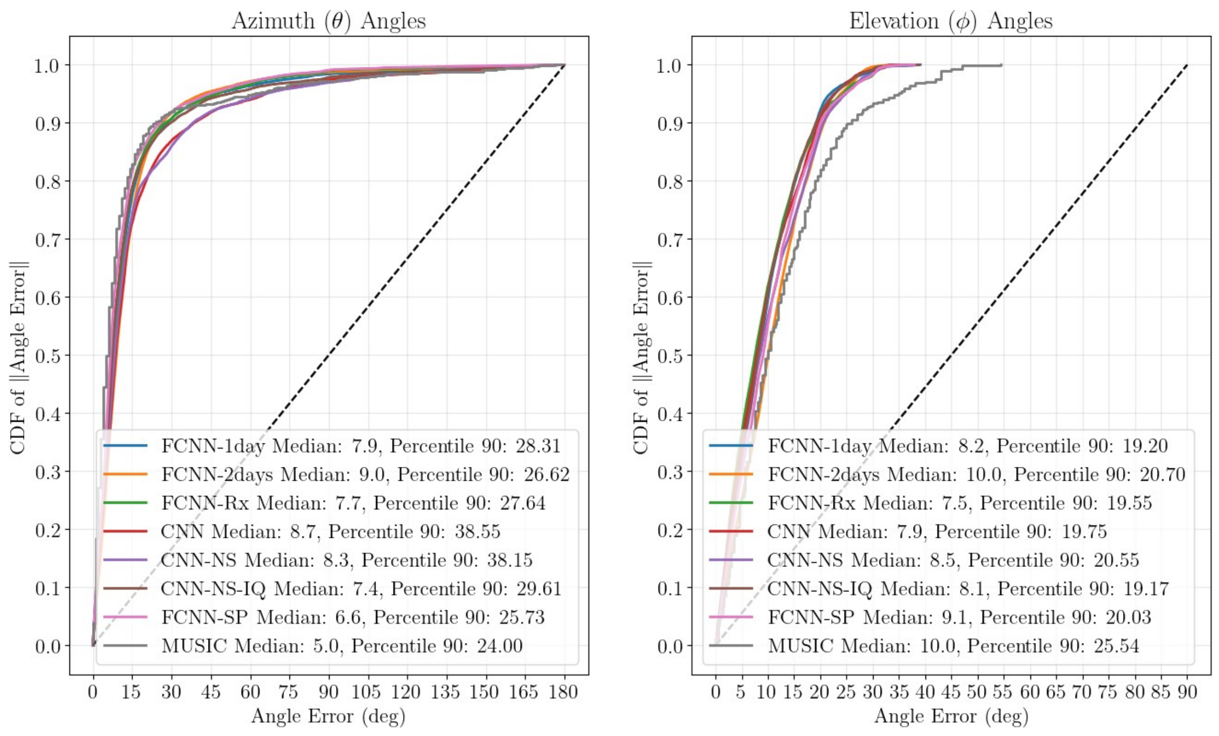


Figure 4.4. AoA estimation performance of DL models. Test split data of 26/09/22.

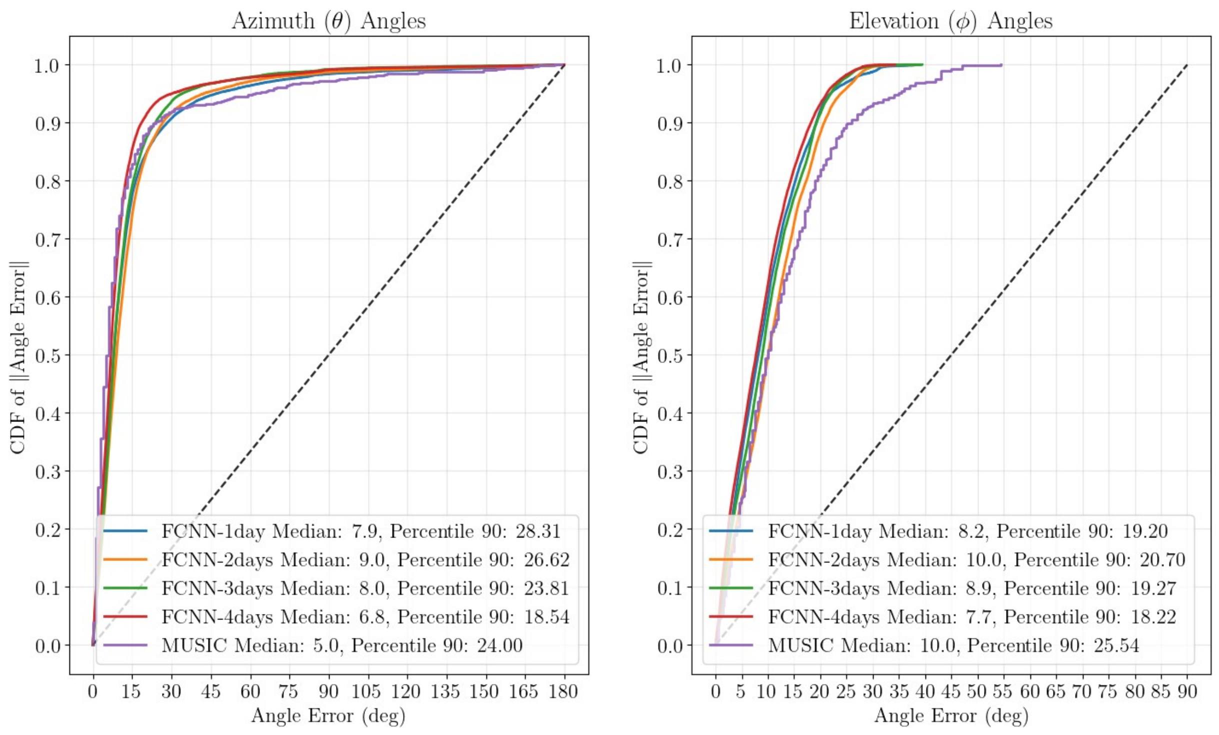


Figure 4.5. Performance of the FCNN model trained with different amounts of data. Test split data of unseen 26/09/22.

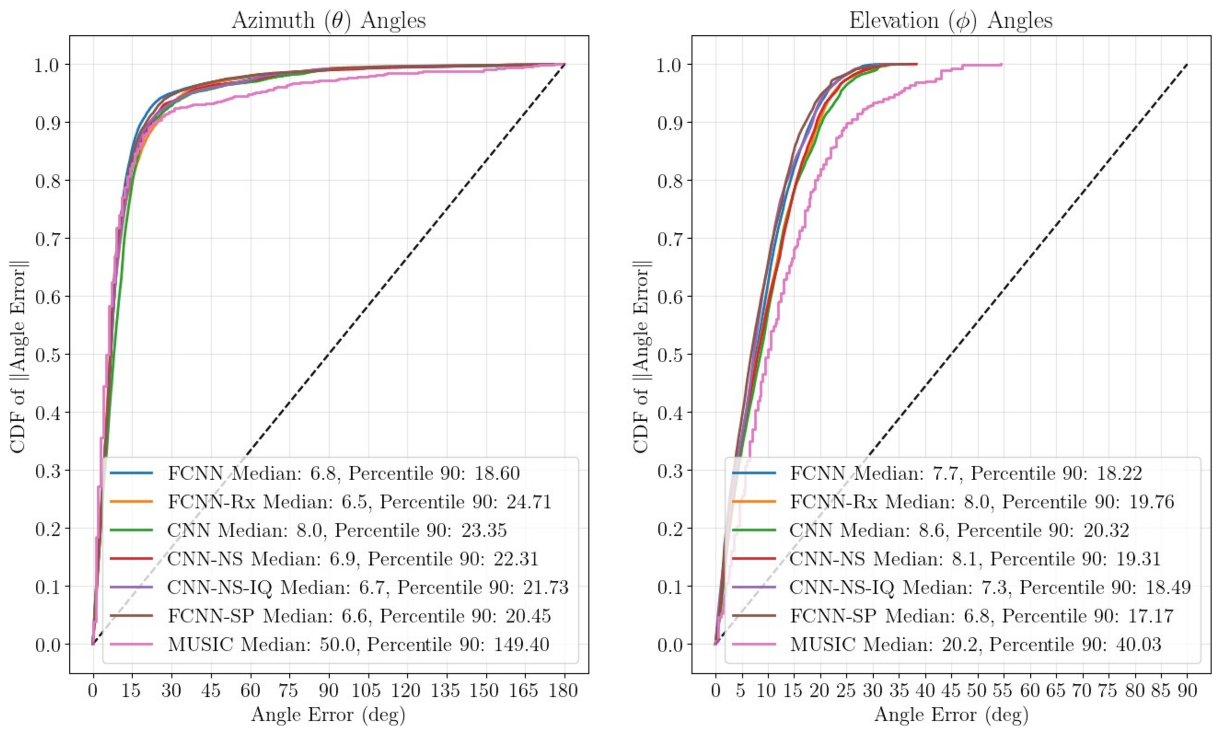


Figure 4.6. AoA estimation performance of DL models trained with 4-day data. Test split data of unseen 26/09/22.

MUSIC algorithm. These models were successfully validated in real scenarios in the presence of noise and multipath conditions.

4.1.2 Generalization and Fine-tuning

One of the well-known issues of DL models is their lack of generalization from one scenario to another. Therefore, prior to use a DL model in production, we have to ensure its generalization ability. We evaluated the AoA estimation performance for the models presented in Section 3.1.2.3 against data gathered in the production scenario. The generalization performance is showed next only in terms of θ to avoid clutter. The same happens for φ, but the azimuth angle is more critical. If it is not correctly estimated, the final positioning accuracy is jeopardized, even though the elevation angle is correctly estimated.

Figure 4.7 results show that the model that generalizes best is the CNN-NS, while FCNN-SP model was the best performing in the preproduction scenario. Those models were all trained with 4-day data of the preproduction scenario. Note that none of the DL models are able to generalize and outperform the baseline MUSIC when no action is performed in their transference from preproduction to production. MUSIC returns errors below the 15º with a probability of 80% whereas the CNN-NS returns the same errors with a probability of 60%. If the FCNN-SP model is considered instead, this probability is lowered until 20% approximately, showing that the model is not able to generalize for this scenario.

The poor generalization can be turned by fine-tuning the DL model. The fine-tuning process corresponds to a short, quick and easy retraining process of the DL model. Instead of starting the training with random weights and biases, their initial values

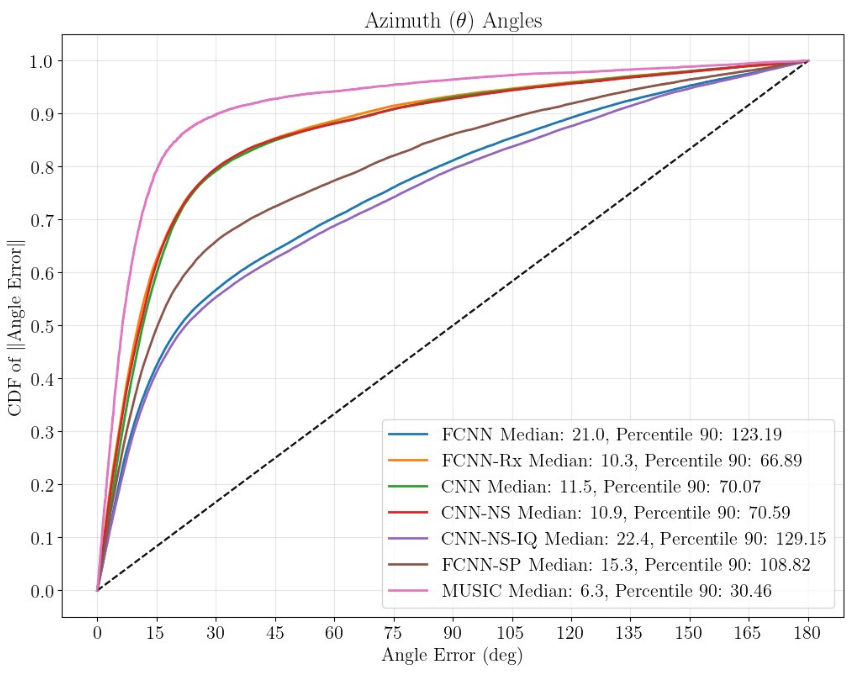


Figure 4.7. Azimuth accuracy in production for models trained in preproduction.

are given by the already trained models in the preproduction scenario. This is only performed for the preceding of the concatenate layer in Figure 3.8. In that way, less training data is need and convergence of the training process is faster. Moreover, not all the layers of the DL model are retrained. As a result, the AoA estimation performance is highly improved in Figure 4.8. If estimations from the fine-tuned DL models are considered, AoA estimation errors are below the 15º with a probability of 87% and a 98% for the CNN-NS and the FCNN-SP models, respectively. Focusing on the DL models, the medians are 4.3 and 1.2 for the CNN-NS and FCNN-SP models, respectively. Besides, the 90% of errors are below 20º for the DL model offering the lower performance, i.e., the CNN-NS. For the FCNN-SP these improvements are higher. Thus, by fine-tuning the DL model the AoA errors can be highly reduced, outperforming the baseline of the MUSIC algorithm.

The presented results highlight notable distinctions between the DL model and the MUSIC algorithm when deployed in an untrained scenario. The successful fine-tuning of the DL model effectively addressed this issue, leading to improved AoA estimation performance compared to the conventional MUSIC algorithm. However, the process of fine-tuning required DUNE to collect fresh data during the production phase.

4.1.3 Energy Efficiency

One of the key advantages of the DL models over conventional approaches is their efficiency. The proposed DL models leverage highly optimized matrix multiplication, making them fast and energy-efficient. In contrast, the MUSIC algorithm involves complex computations for each received signal, including the calculation of the autocorrelation matrix and its eigendecomposition, the extraction of the eigenvectors corresponding to the noise, the definition of the steering vector and the computation of the pseudospectrum for the 2D angle input range to find its maximum. This was validated by measuring the execution time during AoA estimation of MUSIC and the DL models implemented in OpenVINO. The relationship between computation

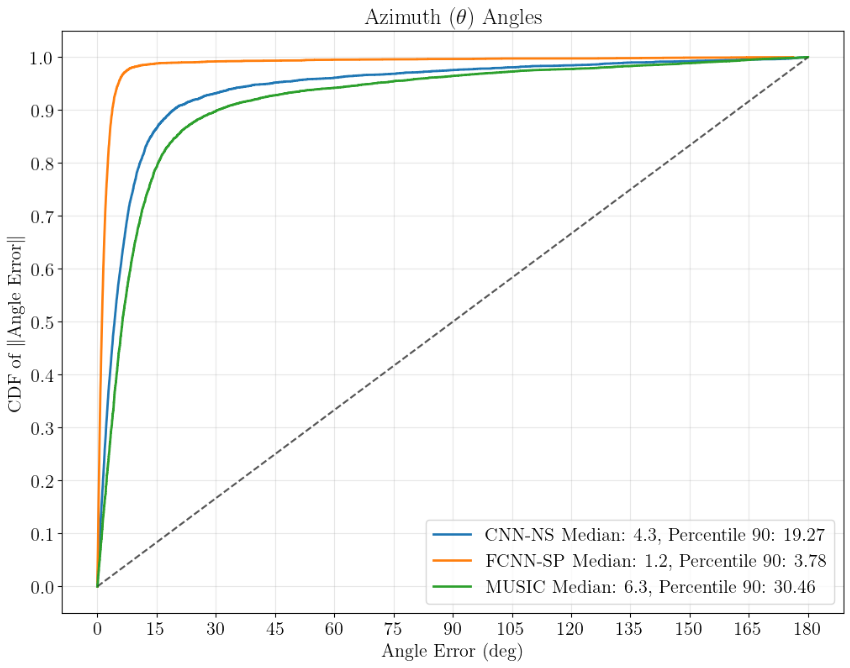


Figure 4.8. Azimuth accuracy in production for fine-tuned models.

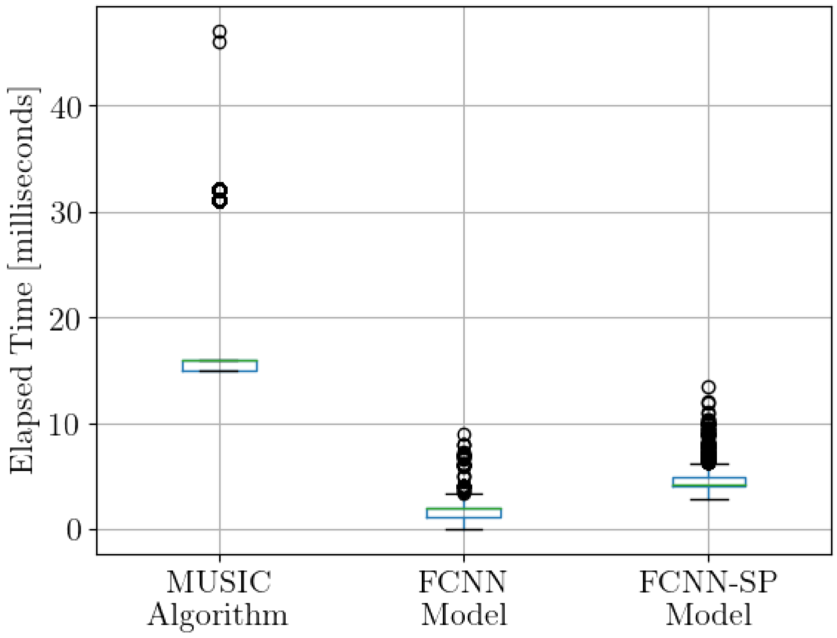


Figure 4.9. The execution times during AoA estimation of the MUSIC algorithm against two DL models implemented in OpenVINO.

time and energy efficiency is crucial in the context of modern computing systems. Generally, reducing computation time often leads to improved energy efficiency, as longer computation times require the system to be active for extended periods. Time results are shown in Figure 4.9. The FCNN and FCNN-SP models have 4,818,619 and 10,090,732 parameters, respectively. The FCNN model derives the estimation using straightforward matrix multiplication and the corresponding activation functions. On the other hand, the FCNN-SP model (best performing in terms of accuracy) includes a preprocessing step where the autocorrelation matrix and its eigendecomposition are computed and fed to the network. Despite the additional preprocessing in FCNN-SP, both DL models outperform MUSIC in terms of energy efficiency. The mean execution times and 95th percentile values for MUSIC are 16 ms and 31 ms, respectively. For FCNN, the corresponding values are 1.95 ms and 2.36 ms, while for FCNN-SP, they are 4.18 ms and 5.16 ms, respectively.

4.1.4 Overall Efficiency

DL models eliminate the need for antenna calibration, which is required for improving the performance of MUSIC. With DL models, variations in antenna behavior and frequency reception are automatically accounted for during training, thereby simplifying the overall process. On the other hand, it is essential to note that DL models require training, where known positions of various points are measured to generate accurate labels. Increasing the amount of training data leads to improved accuracy and generalization. For example, FCNN-SP model converges with 300 training epochs when trained on 70,000 data samples. In contrast, the MUSIC algorithm does not rely on training data, which can be seen as a notable advantage.

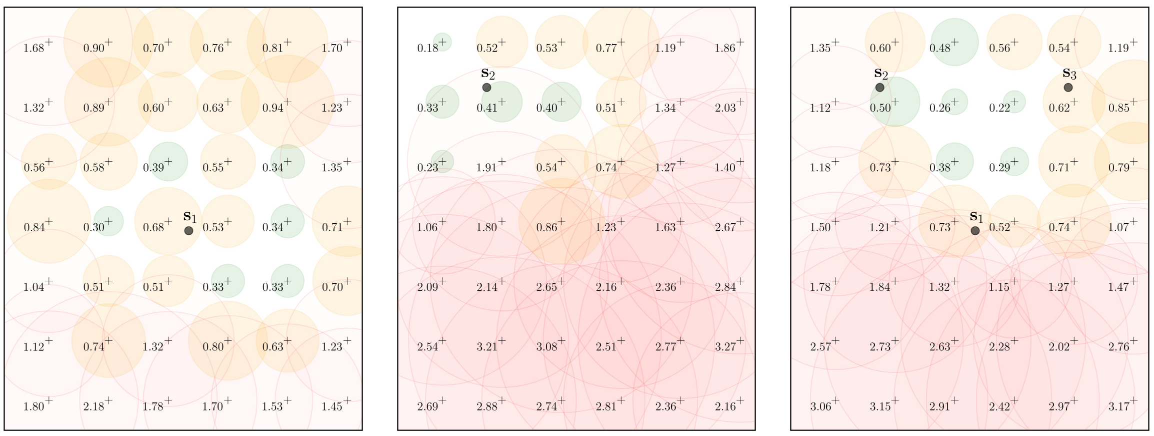
In conclusion, the proposed DL models outperform the MUSIC algorithm in AoA estimation. They offer advantages in terms of efficiency and no antenna calibration while requiring training data for optimal performance. Despite this requirement, their computational speed, accuracy and resource efficiency make them a suitable option for real-time and scalable AoA estimation in DUNE RTLS.

4.2 Position Estimation

In Section 4.2.1, we present the significant improvement in positioning accuracy achieved with the application of DUNE spatial filtering. Subsequently, in Section 4.2.2, we provide an overview of the overall performance of DUNE RTLS in production. Additionally, we discuss the performance gain obtained by employing the proposed DL models for AoA estimation. Lastly, in Section 4.2.3, we assess the system's production accuracy for various time filtering algorithms through evaluation against dynamic experiments.

4.2.1 Spatial Filter Performance

In terms of positioning, the performance is highly related to the amount of locators and their position. In the preproduction scenario under assessment, three BLE-AoA locators were deployed and 42 positioning measurements were considered as shown in Figure 4.10. Performance is highly dependent on the measurement position and the selected locators. As shown in Figure 4.10a, if only the central locator is used, only the positions close to it will be accurately estimated, showing errors whose median is placed in the range $[0, 0.5]$ meters. The same happens for the top left locator in Figure 4.10b. The positioning errors of targets placed in the bottom region of the sce-



(a) Single-locator s_1. *(b) Single-locator s_2.* *(c) Spatial filtering.*

Figure 4.10. DUNE spatial filtering performance against single-locator performance in preproduction. Black circles depict the locators involved. Cross markers depict the measurement points, and text labels explicitly show the CEP$_{50}$ metric (median) value of the errors. The radius of the circles is defined by the median of the errors. Color is green, orange or red when the median value is in $[0, 0.5]$, $(0.5, 1]$ *or* $(1, \infty)$ *meters, respectively.*

nario are higher. DUNE deploys multiple locators and uses spatial filtering to improve the limited performance of single locators. Consequently, a higher level of spatial performance is achieved for targets located within the convex hull of the locators, as shown in Figure 4.10c. For targets positioned outside the convex hull, more advanced outlier filtering algorithms are required to enhance performance. However, this challenge can be addressed by strategically deploying locators at the corners of the room.

4.2.2 Overall Performance and DL Gain

We now show the overall performance in production in terms of the Experimental Cumulative Distribution Function (ECDF) which hides the spatial insights of Figure 4.10 but retains all the statistical information of the errors. To compute the ECDF, 5-minute data were gathered in 42 different equaly-spaced positions in the evaluation area shown in Figure 3.17.

The performance depicted in Figure 4.11 is obtained using the weighed LS solution presented in Section 3.1.3.3 feeded with AoA estimates from the fine-tuned FCNN-SP, CNN-NS and MUSIC baseline. It clearly shows that DL position estimates are the ones providing a better positioning performance. This was expected because DL outperformed MUSIC in AoA estimation, but it does validate DL estimates for positioning. The FCNN-SP model is able to provide θ and φ estimates yielding a positioning error below 1 meter with a probability equal to an 95% and under 0.5 meters with a probability of 50%. These results clearly outperforms the MUSIC algorithm, even when considering CNN-NS model which was the one that generalized best before the fine-tuning. In conclusion, the better the DL AoA estimates the better the spatial positioning performance, and better DL AoA estimates are obtained training with more data which has the operational cost of gathering it.

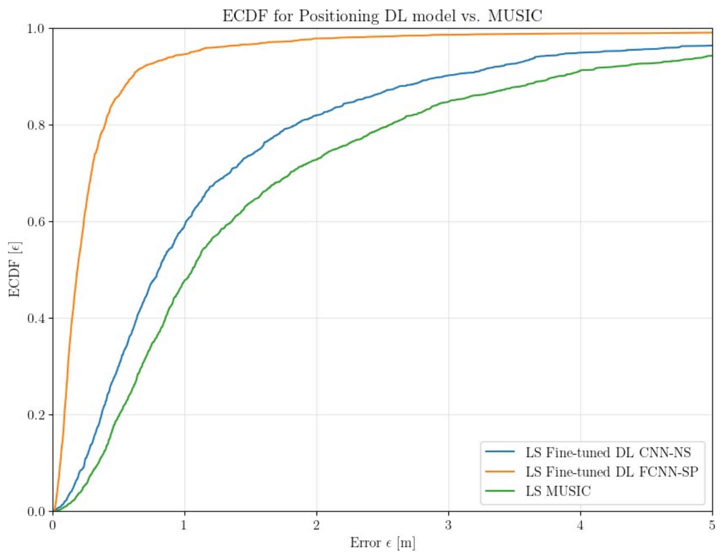


Figure 4.11. Positioning performance in production of DUNE RTLS.

4.2.3 Time Filter Performance

We also evaluated the positioning performance under dynamic conditions in production considering the time evolution. We used the Reference Navigation Agent described in Section 3.1.1 to compare against the RTLS output. The ground truth measurements of the evaluated trajectories were derived using data fusion between the LIDAR and IMU measurements of the robot, leading to centimeter-level accuracy. We recorded different trajectories in a representative area of 10 × 7 meters near two locators, which is highlighted in orange in Figure 3.17. This bounded area allowed controlled and reproducible experimentation and minimal disturbing of the normal operation of the floor.

All methods were evaluated based on the average location error across all trajectory points. The average location error is obtained by aggregating all errors for each of the estimated points. For a given point, the location error is defined as the Euclidean distance between the true coordinates (obtained from the ground truth) and the reported coordinates (obtained from DUNE's DL model). To ensure a more accurate assessment, we also considered the sample instant of time in the calculation of the location error. Specifically, we calculated the distance between the position of the real sample and the closest point on a section of the robot's trajectory. This section of the trajectory is defined by a time window centered around the sample's timestamp. This approach prevents the calculation of distance to the robot's path at a distant point in time if the robot passes through the same point again at a different instant. By incorporating the sample's timestamp, we obtain a more representative and relevant evaluation of the location error for each method.

Figure 4.12 shows the performance of DUNE RTLS while using the spatial and filtering algorithms already presented. Note that we only show the evaluation area shown in Figure 3.17, the rest of the room is not depicted for clarity. The black line depicts the groundtruth trajectory followed by the robot. The orange samples are the estimated positions processed by DUNE RTLS. A gray line connecting an orange sample

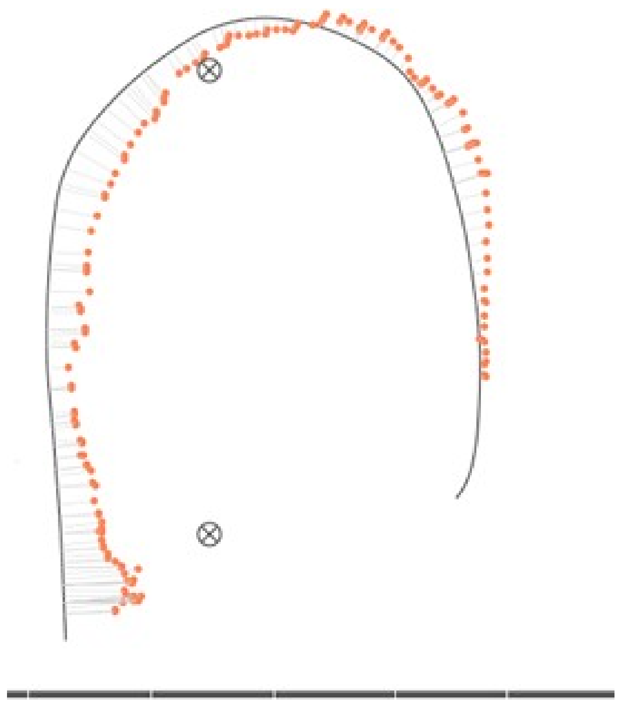


Figure 4.12. DUNE RTLS performance during a dynamic experiment.

to the robot's trajectory denotes that both positions correspond to the same instant of time. Its length is the Euclidean distance between both, that is, the error in the estimation used to compute the performance metric. The average position error of this experiment was 0.26 meters.

Figure 4.13 shows the same experiment as Figure 4.12, but it includes the performance of the different implemented time filter methods. Specifically, it includes the KF of second order, SMA with an averaging window of 2 seconds and EWMA with a decay factor of 0.1. Figure 4.14 shows the same methods but evaluated on a different trajectory. A summary of the results of both experiments is shown in Table 4.1.

The results demonstrate accurate asset tracking and validate the potential of DUNE RTLS. The experiments also indicate that the KF performs best among the evaluated algorithms. However, it is essential to note that the experiments aimed to evaluate the RTLS and developed filters specifically under moving conditions for particular trajectories. It is evident that due to the room's size, the infinite number of possible trajectories and the numerous deployment setups, it was not possible to experimentally cover all potential cases comprehensively. As a result, the focus was on specific trajectories, which provided valuable insights into the technology's performance but may not represent every possible scenario. The results presented in this section may not accurately reflect the behavior of other moving assets or different scenarios, and the generalization capabilities of the trained deep learning models to new scenarios need further examination.

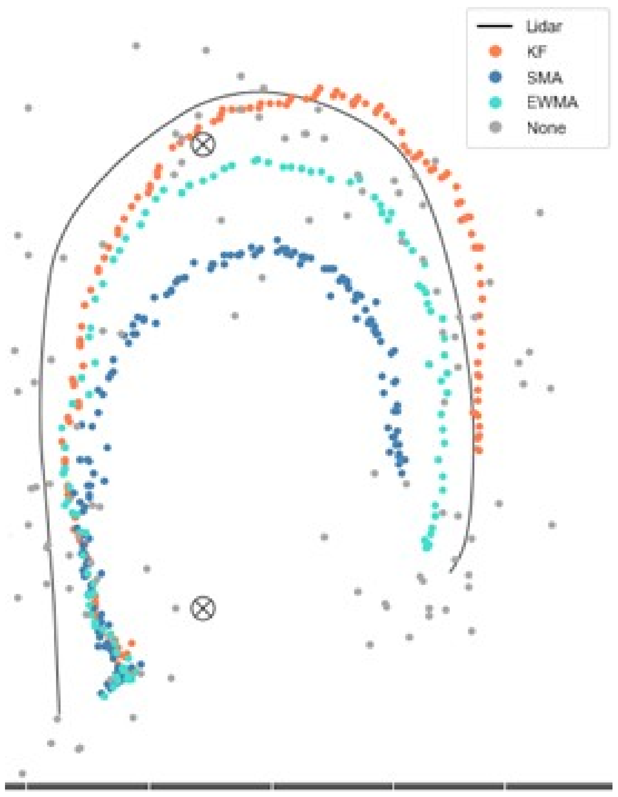


Figure 4.13. Performance of time domain filter during a dynamic experiment.

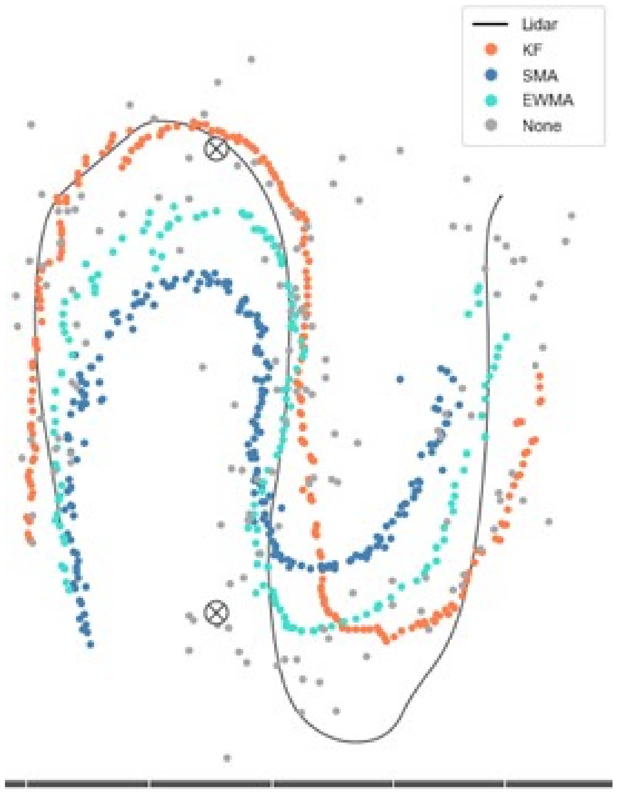


Figure 4.14. Performance of time domain filter under complex movement.

Table 4.1. Average position error in meters of the time filtering algorithms.

Time Filter	Figure 4.13 Experiment	Figure 4.14 Experiment
KF	0.258	0.213
SMA	0.625	0.517
EWMA	0.367	0.242
None	0.374	0.343

5 Exploitation

Positioning technologies play a crucial role in driving digital transformation within the industrial sector. DUNE's capability to accurately track assets in an industrial setting allows for optimized operations, leading to improved Operational Expenditure (OPEX). The implementation of indoor positioning in DUNE offers significant opportunities for enhancing safety and enabling automated operations. The project's focus has been aligned with this ambition, resulting in valuable outcomes that will benefit both academic and industrial communities.

The dissemination of DUNE's work has taken various forms, and the following list highlights the impact of the DUNE RTLS solution during the project's duration and in the immediate future:

- Open source publication of the developed datasets at the code repository https://bitbucket.org/wineuoc/aoa-ble-dataset, entitled "A Direction Finding (DF) Bluetooth Low Energy (BLE) indoor data set for angle of arrival (AOA) and position estimation". A screenshot is shown in Figure 5.1. These datasets will serve companies and researchers to easily develop and test their data-driven algorithms for AoA estimation and positioning without the need to collect their own data.

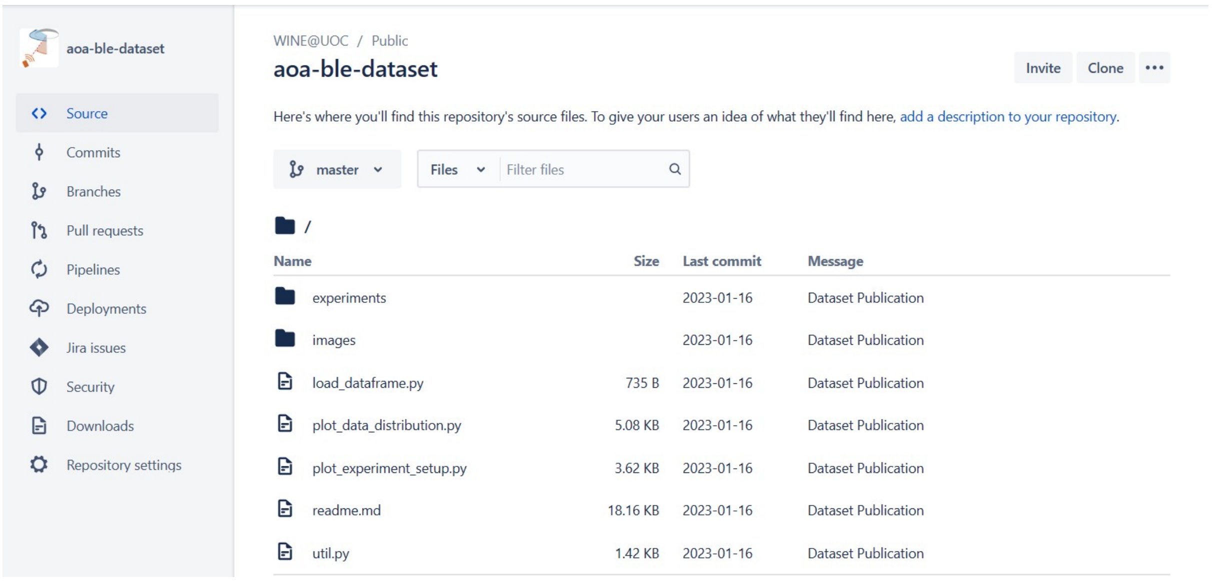
- Open source publication of the trained and used DL models for AoA estimation at the code repository https://bitbucket.org/wineuoc/aoa-ble-deeplearning, entitled "Deep Learning Models for Angle of Arrival (AoA) Estimation in Blue- tooth Low Energy (BLE) Standard". These models jointly with the data can be used to benchmark other proposed methods or improve them with new architecture or training proposals.

- A submitted manuscript to the IEEE Internet of Things jounal, entitled "Indoor Localization with Angle of Arrival: A Fast and Robust Solution for the Internet of Things". Currently under review. This work proposes a positioning algorithm which statistically discards outlier AoA estimates and accurately fuses the remain estimates. It avoids non-convex optimization of state-of-the-art showing better accuracy but with a faster and more efficient computation.

- Attendance and presentation of DUNE RTLS project at HiPEAC 2023 conference on January 2023 at Toulouse, France. A picture is shown in Figure 5.2.

- Confirmation of participation in VEDLIoT Open Call final event at IoT TechExpo at RAI Amsterdam, the 26th of September 2023.

- Preparation of another manuscript to be submitted to a relevant journal. The manuscript will present the main comparative study conducted in this project between DL-based AoA estimation and the conventional approach based on MUSIC. We will provide a deep knowledge of the performance and generalization capacity of DL models when changes of place, time, position, etc. occur. The study carried out will be the first of these characteristics, motivating the research community to evaluate and understand the suitability of DL for this



aoa-ble-dataset

Here's where you'll find this repository's source files. To give your users an idea of what they'll find here, add a description to your repository.

/

Name	Size	Last commit	Message
experiments		2023-01-16	Dataset Publication
images		2023-01-16	Dataset Publication
load_dataframe.py	735 B	2023-01-16	Dataset Publication
plot_data_distribution.py	5.08 KB	2023-01-16	Dataset Publication
plot_experiment_setup.py	3.62 KB	2023-01-16	Dataset Publication
readme.md	18.16 KB	2023-01-16	Dataset Publication
util.py	1.42 KB	2023-01-16	Dataset Publication

Figure 5.1. Open source code repository of DUNE datasets.

purpose. The main advantage of DL models is that they can be run in the far-edge very efficiently in specific and optimized hardware, contrary to MUSIC that has a high computational cost.

Research projects aimed at improving the solution and contributing to the community will ensure the long-term sustainability of the work beyond the end of this call. Additionally, DUNE now holds a significant place in our team's roadmap for developing a spin-off company. Once fully developed, DUNE RTLS will offer cost-effective and energy-efficient indoor positioning solutions tailored to meet the specific needs of our customers.

At present, DUNE RTLS is deployed in production at the UOC's main headquarters, permanently situated on the 1st floor of the IN3 research hub. This deployment serves three primary objectives: (i) advancing research in IoT indoor positioning at the Wireless Networks research group, (ii) providing a showcase of the end-to-end positioning solution to potential customers, and (iii) promoting multidisciplinary research at the IN3. For instance, the group of psychologists from the UOC has shown interest in utilizing DUNE as a tool for monitoring social interaction during their experiments.

Next steps

- Dataset published at: https://bitbucket.org/wineuoc/aoa-ble-dataset
- Evaluate other DL architectures:
  - Determine what of the inputs to the NN are relevant (RSSI, channel, raw IQ, preprocessed IQ.)
    - Traditional alg. (MUSIC) use a covariance matrix that correlates the phase variation between antennas. (Try with that)
  - IQ are complex numbers. Convert to polar coordinates and train the NN.
  - Try Recurrent NN (RNN), use batches of IQ (e.g 10 samples)
  - Convolutional NN (CNN), use the IQ samples covariance matrix
  - Complex NN, see if there is any advantage when dealing with Complex numbers.
- Comparte to state of the art methods such as MUSIC.

14

Figure 5.2. Professor Xavier Vilajosana presenting DUNE at HiPEAC 2023.

6 Summary and Next Steps

In this document, we present the primary outcome of the DUNE project in collaboration with VEDLIoT: the initial development and production deployment of DUNE RTLS.

The developed localization system transforms the estimated angles from individual locators into absolute positions relative to a map, offering improved localization accuracy through advanced filtering capabilities. Throughout this deliverable, we focused on spatial domain filtering, leveraging spatial diversity from multi-locator deployments. Additionally, we incorporated time domain filtering capabilities, utilizing subsequent samples in time to further enhance accuracy. The system underwent evaluation under both static and dynamic conditions, demonstrating promising sub-meter accuracy.

Looking ahead, future DUNE RTLS versions will be enhanced by incorporating other common wireless technologies such as WiFi, RFID, and UWB. This expansion aims to deliver an end-to-end cost-effective, energy-efficient, and multi-tech positioning solution.

7 References

[1] T.-M. Choi, S. Kumar, X. Yue, and H.-L. Chan, "Disruptive technologies and operations management in the Industry 4.0 era and beyond," *Production and Operations Management*, vol. 31, no. 1, pp. 9-31, 2022.

[2] F. Zafari, A. Gkelias, and K. K. Leung, "A survey of indoor localization systems and technologies," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2568-2599, 2019.

[3] P. S. Farahsari, A. Farahzadi, J. Rezazadeh, and A. Bagheri, "A survey on indoor positioning systems for iot-based applications," *IEEE Internet of Things Journal*, vol. 9, no. 10, pp. 7680-7699, 2022.

[4] M. Woolley, "Bluetooth core specification v5. 1," in *Bluetooth*, 2019.

[5] M. Woolley, "Bluetooth direction finding: A technical overview," *Bluetooth Resources*, 2019.

[6] M. Kaiser, R. Griessl, N. Kucza, C. Haumann, L. Tigges, K. Mika, J. Hagemeyer, F. Porrmann, U. Rückert, M. vor dem Berge, *et al.*, "Vedliot: very efficient deep learning in iot," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 963-968, IEEE, 2022.

[7] H. Krim and M. Viberg, "Two decades of array signal processing research: the parametric approach," *IEEE signal processing magazine*, vol. 13, no. 4, pp. 67-94, 1996.

[8] N. Wu, Z. Qu, W. Si, and S. Jiao, "Doa and polarization estimation using an electromagnetic vector sensor uniform circular array based on the esprit algorithm," *Sensors*, vol. 16, no. 12, p. 2109, 2016.

[9] G. Ran, M. Xing-Peng, L. Shao-Bin, W. Yi-Ming, and W. Xiu-Hong, "A fast doa estimation algorithm based on polarization music," *Radioengineering*, vol. 24, no. 1, p. 215, 2015.

[10] H. Ye, B. Yang, Z. Long, and C. Dai, "A method of indoor positioning by signal fitting and pdda algorithm using ble aoa device," *IEEE Sensors Journal*, vol. 22, no. 8, pp. 7877-7887, 2022.

[11] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press: Cambridge, MA, USA, 2016. http://www.deeplearningbook.org.

[12] L. Schmitt and W. Fichter, "Continuous singularity free approach to the three-dimensional bearings-only tracking problem," *Journal of Guidance, Control, and Dynamics*, vol. 39, no. 12, pp. 2673-2682, 2016.

[13] E. Naftali and N. C. Makris, "Necessary conditions for a maximum likelihood estimate to become asymptotically unbiased and attain the Cramer-Rao Lower Bound. Part I. General approach with an application to time-delay and Doppler shift estimation," *The Journal of the Acoustical Society of America*, vol. 110, no. 4, pp. 1917-1930, 2001.

[14] G. Welch, G. Bishop, *et al.*, “An introduction to the Kalman filter,” 1995.

ICT-56-2020 - Next Generation Internet of Things

Final technical report BEAM_IDL VEDLIoT Open

Document information	
Project website	https://www.vedliot.eu
Dissemination Level	PU
Nature	R
Contractual Deadline	30.06.2023
Author	…
Contributors	…

The VEDLIoT Project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957197.

Changelog		
Version	**Date**	**Type**
V 0.1	2023/06/26	Structure definition
V 0.2	2023/06/29	Add prior deliverables information
V 0.2	2023/07/10	Complete implementation and evaluation sections
V 1.0	2023/07/19	Finalisation
V 1.1	2023/09/27	Fix 1

Table of contents

Executive summary

The following report is the last deliverable of BEAM_IDL, which explains the technical activities carried out during the project. The document (D3.1 final report) contains information from previous deliverables, D1.1 requirements, D2.1 LBS identification solution and D2.2 BEAM IDL Business model.

In addition, in D3.1 the experimental part of last 3 month were added to complete and finalize the project report.

This document is a complementary report of the management final report, more focused in project administration tasks.

Introduction

Increasing demand for the use of Aluminium (Al) in the manufacture of Electric Vehicles (EV) –especially in battery closures and chassis attachments– requires optimized laser welding schemes. Intrinsic Al-associated welding issues are being overcome with new dynamic and adaptable laser beam approaches, mainly to increase laser-material interaction. Process control for these new laser manufacturing scenarios requires novel in-process monitoring strategies demanding pattern and process event recognition.

BEAM-IDL proposes the application of optimized DL algorithms for visible and infrared (IR) laser welding process image recognition. This novel approach will allow, in the future, to link feature extraction with process parameters, opening the gate to novel process optimization.

This document represents the last deliverable of the BEAM_IDL project, in which the state of the art of laser welding as well as monitoring equipment will be presented. The document explains also a definition of industrial constraints, hardware interfaces and software packages to finally developed and implemented hardware and software architecture design to solve the challenges of the BEAM_IDL approach.

The work presented in this document summaries the implementation and experimental test carried out during the project, explaining the algorithms developed and its performance in real welding operations. In order to improve system performance several datasets were generated first in a demonstrator setup, the information has allowed the developer to train and improve the algorithms performance.

Finally, the implemented architecture has being translated to a relevant machine integration, using a robot cell at LORTEK and an industrial optic developed by EXOM Engineering. The proposed technology and solution have been tested in a relevant demonstrator part.

Overall, the proposed solution and architecture will be part of a process monitoring and control system that provides a differential value to current welding portafolio, both for EXOM products and LORTEK services. The integration of ad-hock AI (artificial intelligence) approaches for monitoring and process control are crucial for competing against low value manufacturing solutions coming from outside EU.

Idea and architecture

Aluminium (Al) is 3 times less dense than steel (2.69 gr/cm3 vs 7.84 gr/cm3) it is an excellent approach for EV weight reduction. Moreover, its elasticity modulus is as well around 3 times lower (70 GPa vs. 193 GPa – higher modulus means stiffer material), providing notable resistance to collision and energy absorption capabilities. Aluminium is not a novel material in the car manufacturing industry, as has frequently been used in the production of body-in-white, chassis, suspension, and wheels.

Electric-Vehicle driving autonomy comes directly related to the energy density of the system as a whole, not only including the battery storage capacity but also the overall weight of the vehicle, in which battery structural light-weighting is the general trend nowadays for energy efficiency enhancement. It is mandatory to contribute to the general weight reduction, thus Aluminium is being included in the closure structure –weight savings of 40% compared to an equivalent steel design [1] –, placing tray, etc. The use of Al in these kinds of structures comes with several engineering challenges, like hermetic closures, structural strength in the event of an accident, etc.

As with any other assembly process, the union between different parts of the same and also different alloys requires a joining process, which in many cases means a welding process, due to its robustness, automation capability, and reliability for high-volume production in an industrial environment. Unlike steel, the welding of Al presents several challenges; the guidance of the heat flow and distortion control is much more challenging, it must be welded in a clean and dry environment, etc. [2] These issues become even more challenging when it comes to welding bigger parts like the aforementioned battery closures or trays.

To meet the increasing demand for improved aluminium welding operations, manufacturers have further developed and refined existing processes to offer greater control over the welding process. Here are some examples of the processes:

- Arc welding processes and their variants
- Laser-MIG-Hybrid Welding (LHW) - especially for structural components
- Single-wire or tandem wire MIG processes, both with (CMT or CMT Tandem) or without reversing electrodes
- Resistance spot welding (RSW)
- Friction stir welding (FSW)

Laser-based welding processes present several benefits when compared with more traditional thermal processes; (1) it results in less heat input –laser beam's high energy density–, (2) provides a high aspect ratio –penetration depth and weld seam width ratio– what contributes to (3) lower part distortion while (4) maximizing production. [3].

Laser Beam Welding (LBW) is suitable for highly conductive materials, due to high concentrated energy for melting. A laser beam is a non-contact flexible tool, and the power intensity distribution can be changed for specific purposes with enhanced productivity [4]. Low heat input provides a narrower heat-affected zone (HAZ), and thus, the strength may be enhanced. Laser beams may operate under two essential temporal modes, continuous wave (CW) and pulsed wave (PW). Modern laser beams have highly customisable pulsing capabilities with a pulse duration in the order of μs, very high peak powers in the order of megawatts, and complex pulse shapes with frequencies that greatly surpass that of pulsed arc welding. LBW is normally carried out without filler wire, especially for thinner sheets, and is mainly used in the automotive and aerospace industry [5].

Heat conduction laser welding usually operates under <106 W/cm2 energy density (see Figure 1a) and is more often used for thin plates (<1 mm), where the energy should not be sufficient to generate a keyhole but high enough to melt the material locally. Here, the melt flows are very dependent on the active surface elements and are similar to the tungsten inert gas (TIG) process weld pool [6]. Higher energy densities (>106 W/cm2) form a keyhole, providing much higher penetration depths, due to multiple reflections (see Figure 1b) and increased absorption. The threshold for transition from the heat conduction mode and keyhole varies and mainly depends on the wavelength and surface conditions. Based on Behler et al. [7], Nd:YAG requires two times lower power density to generate a keyhole, and its drilling time is several times faster, compared to the CO_2 laser. Similar is expected for the fiber/disk laser. There is also the transition mode between heat conduction and keyhole. Notably, the stability in the heat conduction mode is much higher than in the keyhole mode, due to a more stable weld pool with low porosity and spattering. The keyhole mode often has a spiking effect, as the keyhole collapses, causing porosity. Therefore, more optimisation of parameters is needed, but it is more attractive for thicker plates. [8]

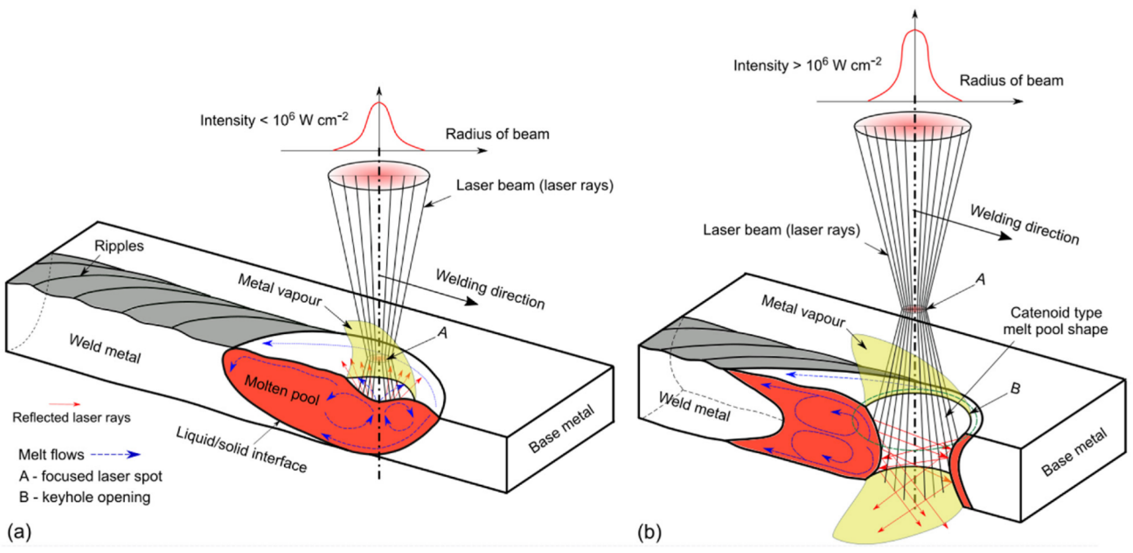


Figure 1: (a) Schematic illustration of laser beam physics (beam focused near the surface) during heat conduction mode with melt flows. (b) Schematic drawing of laser beam physics (defocused beam) during keyhole mode welding [5]

Due to the fast solidification speed, LBW may cause cracking, porosity, and poor metallurgical affinity in the fusion zone due to brittle phases. Microstructure in the fusion zone can be improved by using specific interlayers by coating components [9]. Disadvantages of LBW are high-cost investments, safety issues, high requirements of joint preparations of components to be joined, and more complex processes and variables to be adjusted.

Poor appearance of the weld bead and surface defects are frequent challenges in LBW, due to unstable melt pool formation, especially in the keyhole mode (see Figure 2). External weld seam defects are undercut and underfill (top and root), spattering, upper and root humping. The formation of undercuts, spattering and root humping is attributed to many factors, including process parameters, welding conditions, shielding gas composition, surface tension and viscosity of the used metals. International standards ISO 13919-1 [10] (for steel) and ISO 13919-2 [11] (for Al) specify the limits for different quality levels. Detailed information concerning various external weld pool defects and mechanisms can be found in [12].

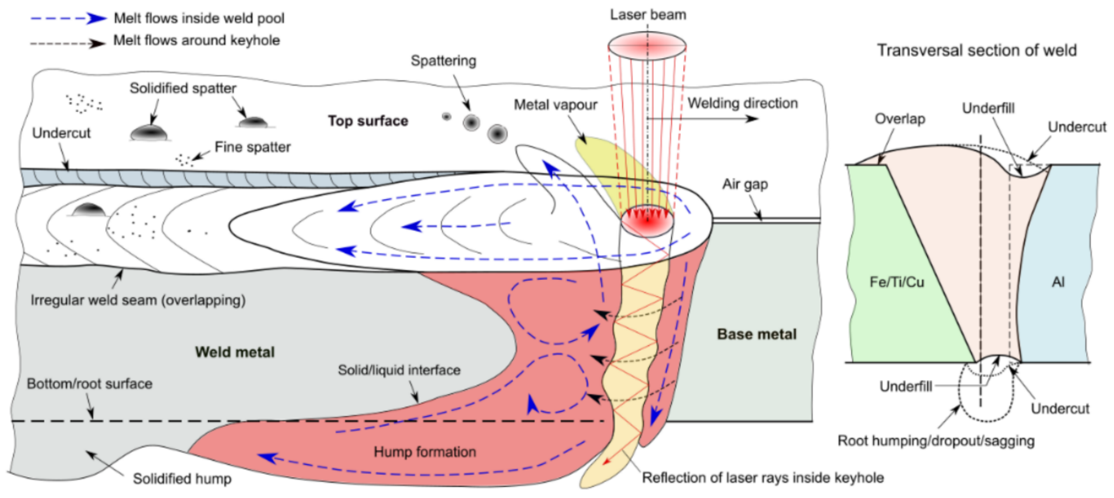


Figure 2: Common external imperfections in LBW during full penetration mode (with focused laser beam) using V-groove butt preparation [5]

S-o-t-A of monitoring equipment for laser welding of aluminium alloys

Optical and acoustical process signals can be measured from the emissions of the welding process using suitable sensors. Since the signals contain information about the beam-material interaction, welding defects or process deviations can be detected during the manufacturing and recorded for every single workpiece. Figure 3 shows a selection of detectable emissions, which can be used as process signals. The reflected laser is the amount of radiation of the laser source which is not absorbed by the material. Acoustic emissions, which can be divided into air-borne and structure-borne emissions, are originated from the stress waves induced by changes in the internal structure of a workpiece, the surface of the metal vapour and laser beam back reflection. The metal vapour and the molten pool emit continuous radiation whose spectrum varies with different laser applications. For instance, during IR-laser welding, the process radiation is in the visible and infrared range [13]. For CO_2 laser keyhole welding, the plasma generated is known to emit light with a wavelength between 190 nm and beyond 400 nm, and the spatter emits light with a wavelength between 1000 nm and 1600 nm [14]. In addition, the geometrical parameters of the keyhole and melt pool also contain useful information which can be used to inspect the welding quality.

It is highly desirable to diagnose laser process quality using these emissions, in particular, to understand the relationship between emission characteristics and weld quality characteristics.

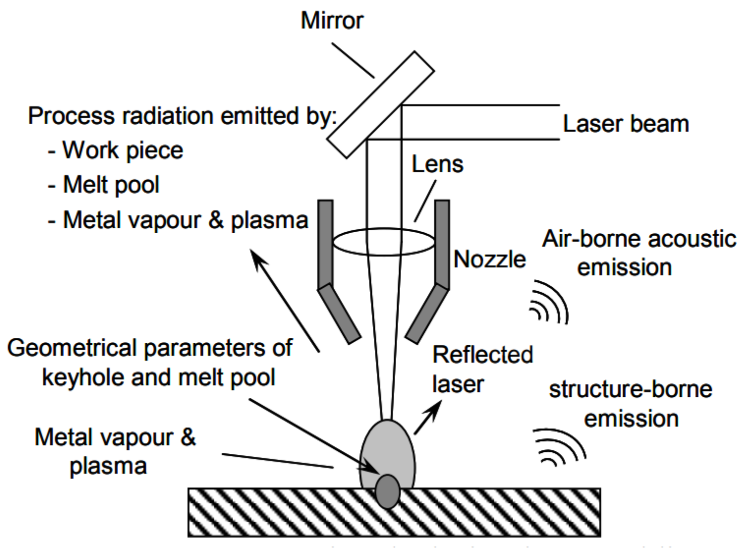


Figure 3: Process signals during laser welding [15]

Acoustic emission techniques

Acoustic emission (AE) involves a sensor which converts process sounds into electrical output to a measurable variable. Air-borne emission has a human audible range between 20 Hz and 20 kHz. The typical sensor for this emission is a microphone placed nearby the weld zone (Figure 4). The frequency of structure-borne emission is usually from 50 kHz to 200 kHz. Piezoelectric transducers are typically mounted at an acoustic mirror, acoustic nozzle and workpiece to measure AE (Figure 4).

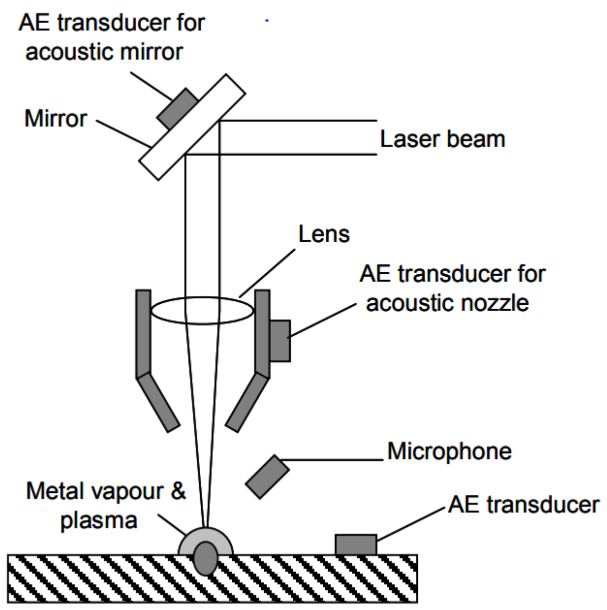

AE transducer for acoustic mirror
Mirror
Laser beam
Lens
AE transducer for acoustic nozzle
Microphone
Metal vapour & plasma
AE transducer

Figure 4: Typical experimental setups for acoustic emissions [15]

AE techniques are still investigated in academia, however, applications in this field have declined due to their limitations. For structure-borne emission monitoring, contact installation of transducers is needed, therefore, it is not suitable for mass production. For air-borne emission inspection, it is often difficult to acquire useful signals from hostile and noisy environments in the industry [15].

An alternative method recently used for laser welding applications is based on air-coupled detection of laser-generated ultrasound with a broadband optical microphone. While the detection bandwidth in this setup is limited by ultrasound attenuation in the air to a range of 10 Hz up to 1 MHz, it is still a factor 10 larger than what state-of-the-art air-coupled piezo transducers can offer. Consequently, ultrasound transients with single-microsecond durations can be reliably detected, temporally separated and spectrally analyzed [16].

Optical detector

Optical detectors, particularly ultraviolet (UV), visible or infrared (IR) detectors, have been widely used to convert the flux density of the radiation emitted by the laser welding process into an electrical signal. Optical filters are often placed in front of the detectors to confine the spectral ranges of the whole sensor system. Typical setups using co-axial and off-axial arrangements are illustrated in Figure 5

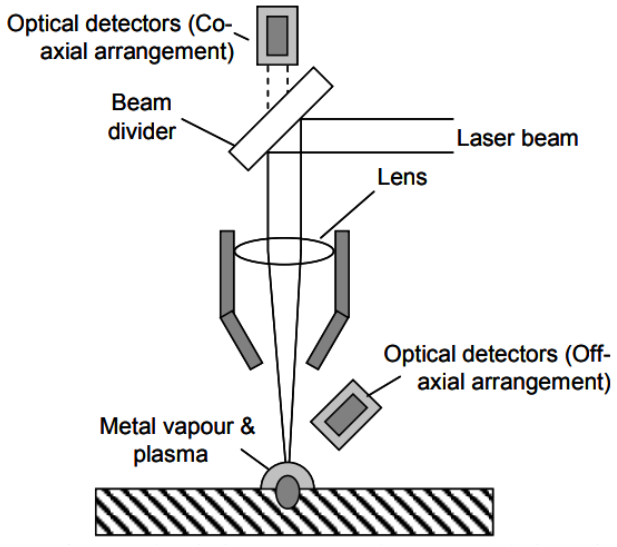

Optical detectors (Co-axial arrangement)
Beam divider
Laser beam
Lens
Optical detectors (Off-axial arrangement)
Metal vapour & plasma

Figure 5: Typical setups for optical detectors using co-axial and off-axial arrangements [15]

By far, image processing techniques are suggested to be the most intuitive and adaptable method as an in-process monitoring tool. Considering that during the welding process several thermal and mechanical phenomena arise, image processing techniques can be classified as thermal –captures the infrared (IR) radiation of the welded component and translate it to temperature distribution– and visible –allowing to observe melt pool, keyhole plume and spatter characteristics–. As seen, each of them will provide different process features, which converge in unique controlling and evaluation approaches. This separation has been overcome by a few approaches combining both thermal and visible techniques to benefit from the most suitable information from each of them [17]. Even, it has already been monitored the effect of dynamic and superimposed laser energy distributions within welding quality with visible and IR cameras [18], but, to the best of our knowledge, never been combined with a process control approach.

The main objective of the present project –entitled BEAM-IDL– is to develop a methodology capable of identifying, in-process, different laser beam shapes through vision and IR cameras, so each feature comes linked with process parameters and, consequently, allows to open the gate to a novel monitoring strategy for Dynamic Beam Shaping (DBS) welding approaches.

Definition of industrial constraints, hardware interfaces and software packages

Laser welding, unlike traditional welding methods that melt the material with the subsequent rapid solidification, evaporates material due to the high energy applied, creating a vapor capillary or keyhole. This evaporation generates pressure forces that interact with the liquid metal sustaining the formed keyhole. Consequently, any process parameter can unbalance these forces, destabilizing the keyhole or even collapsing it, leading to porosity, voids, unfilled keyhole cavities, surface craters and depressions, etc. Thus, the stability of this keyhole is crucial for achieving high-quality welds. In welding of Al –Copper (Cu) and new high-strength Al alloys, the keyhole can be inherently unstable mainly as liquid aluminium has less surface tension and viscosity than common metal alloys, which results in greater fluidity of the molten pool and chaotic fluid flowing [19].

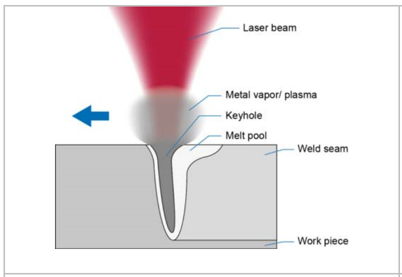
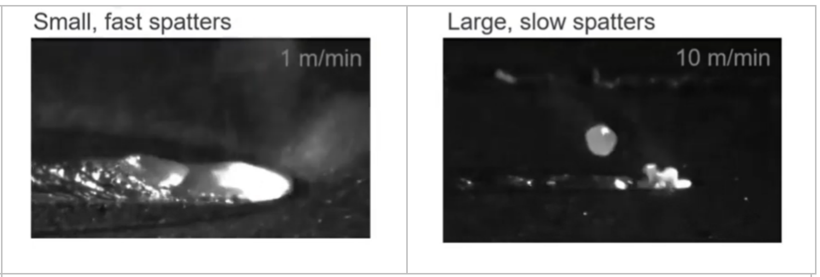
	
a)	b)

Figure 6: a) Parts and phenomena of a laser-based welding process; b) Different melt-pool dynamics and spatter behavior as a consequence of the keyhole effect.

Several strategies have been tested to optimize the keyhole behavior, from laser parameter variation –wavelength, pulse energy and duration, repetition, and feed rate– to more complex approaches centred on modifying the beam shape –e.g. Dynamic Beam Shaping (DBS)– and thus augmenting the laser interaction with the material – oscillating and wobbling [20], superimposed intensity distributions [21], dual-beam laser heads [22], diffractive optics [23], and Coherent beam combining [3]. Process parameters involved in this process might be laser power (cw lasers), energy distribution, pulse energy (in the case of pulsed lasers), duration, repetition frequency, and feed rate.

DBS process upscaling approach opens the gate to a new paradigm in welding optimization strategies and production throughput growth in, for example, aluminium welding in battery closures and placing trays, but comes linked to an increase in complexity in parameter configuration, as several mechanical, optical and laser settings should be modified and adapted, power, speed, amplitude, frequency, pattern, etc. As a consequence, in-process parameter adaptation comes upscaled in complexity, where processing dynamics modification have to be necessarily automatic. This is technically translated in a need of knowing at all moments which is the shape of the laser beam and how it is interacting with the material; i.e. this scenario requires a precise in-process monitoring approach capable of (1) determining what phenomena are arising based on the selected welding strategy –e.g. beam shapes– and (2) adapting and/or modifying process parameters in real-time.

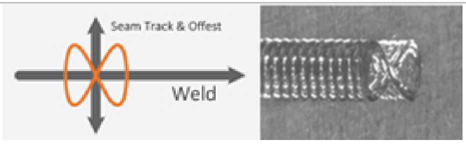
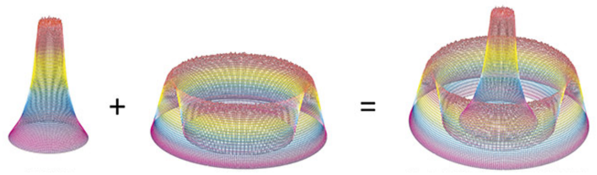
	
a)	b)

Figure 7: Some examples of a) "infinite" shape wobbling [20] and b) superimposed intensity distribution [21]

To provide machines with a feature or characteristic interpretation capacity, these machines must be "taught" so they can discern. The most widely used approach to date is known as Automatic Learning or Machine Learning (ML), where a computer looks at data, builds a model based on that data and uses that model as both a hypothesis about the world and a piece of software that can solve problems. Within these automatic learning approaches, Deep Learning (DL) is the one booting nowadays the development of artificial vision applications. DL is the technology that allows a computer to learn like a human being, enabling learning methods that are more robust, effective, and more like those of the human brain. Overall, DL allows already no to just classify an image with certain labelling –cat, dog, duck–, but to localize, detect or even discern these objects –or features– within the same image.

An important challenge when using artificial intelligence is that they require a high computational capacity due to the complexity of the algorithms, and a large amount of data to be processed in a short time. For this reason, some researchers have investigated how to minimize computational requirements when implementing artificial intelligence in embedded systems. Yang et al. [24] present in their work a "lightweight" transfer learning algorithm to retrain a CNN suitable for an embedded system in welding applications. Several previous undertakings have focused mainly on automatic image processing for quality control of parts, but this is one of the possible applications. In manufacturing processes, it is also necessary to actively control the process in question considering the information coming from the different sensors and/or camera images. In this way, it is possible to adapt the controllable parameters of the process seeking to reduce the defects during the same manufacturing of a piece before discarding it at the end of the processing line. In this sense, Franciosa et al. [25] determine the importance of closed-loop processing to meet industrial requirements. Bozic et al. [26] work in this direction using a convolutional network (CNN) as a fundamental element in closed-loop control in a laser welding process. In this investigation, the images were processed by the CNN to classify the penetration characteristics of the laser and feed the information collected to the PID control of laser power.

Thermal and visible image treatment for laser DBS feature extraction certainly has required the application of specific DL algorithms but as previously stated, the intrinsic rapidness of the laser-based manufacturing process has required the application of algorithm acceleration procedures –so to open the gate to real-time welding process control– meaning that a dedicated and embedded architecture should be specifically deployed. Aligned with this, some work were done in EXOM Engineering in the development of embedded electronics with basic algorithms for the monitoring and control of laser welding processes, as well as object classification using industrial cameras. In this sense, the wobble laser head allows real-time closed-loop temperature control with an IR camera and an adaptive scanner head positioning with a visible camera to perform precise melt pool monitoring and process traceability [27]. For that, a dedicated hardware structure deployed over the laser welding head:

- IR images: dedicated FPGA
- Visible images: NVIDIA® Jetson Nano™
- Real-time process control: Xilinx Ultrascale+ ZU3EG MPSoC

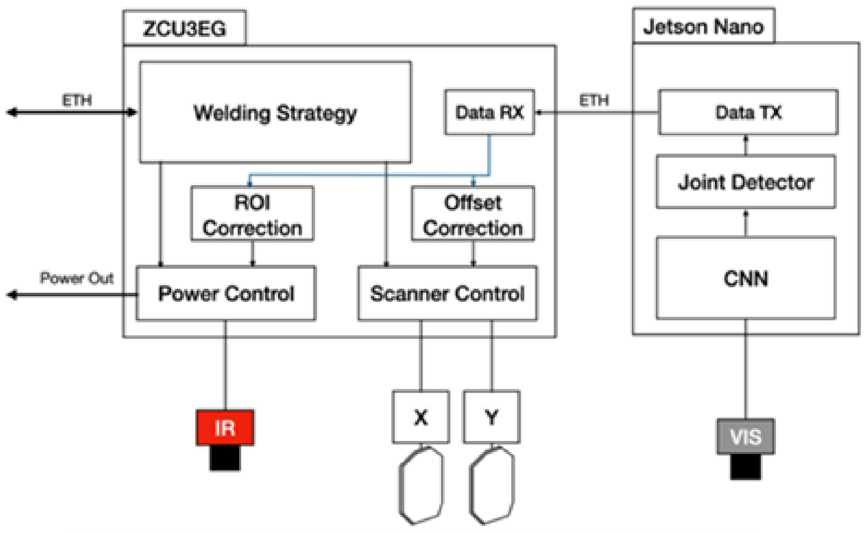


Figure 8: Process control and monitoring general HW and SW diagram – Implemented by EXOM Engineering

The parallel processing capability offered by FPGAs is particularly useful in accelerating hardware image processing. While GPUs are specialized to speed up calculations related to video or specific matrix operations thanks to new architectures that incorporate processors such as CUDA cores, FPGAs can be configured to implement highly specialized calculations beyond video processing. Thus, the acceleration of video processing algorithms with communication modules or logic circuits of different types were combined in the same design. With the help of direct memory access (DMA) modules, the transfer of data from the Software field executed in the ARM to the Hardware field executed in the programmable logic of the FPGA was facilitated. In this way, the frames of the videos were acquired by software through the libraries provided by the camera manufacturers and were transferred through a DMA module to the FPGA processing line by serializing the data of each frame. After processing the data in the programmable logic, depending on the application, these processed frames were sent back to the software process for display, additional processing or saving to a file, for example. However, in some applications, it was not necessary to send the processed frames to the software process, and they were used to perform hardware tasks such as the control of some process parameters with the information extracted from the processed images.

Besides the programmed temperature control tasks, the developed structure allows process visualization (visible and IR images), but it is not prepared for smart image analysis duties, as no classification or segmentation routines are implemented. For this, LORTEK provide valuable insight as counts with experience in the deployment of image analysis DL algorithms. LORTEK, as a Technological Centre with expertise in metal-based manufacturing processes and their digitalization for industrial network transferring, has deployed DL solutions for both laser-based welding process monitoring and welded part quality inspection:

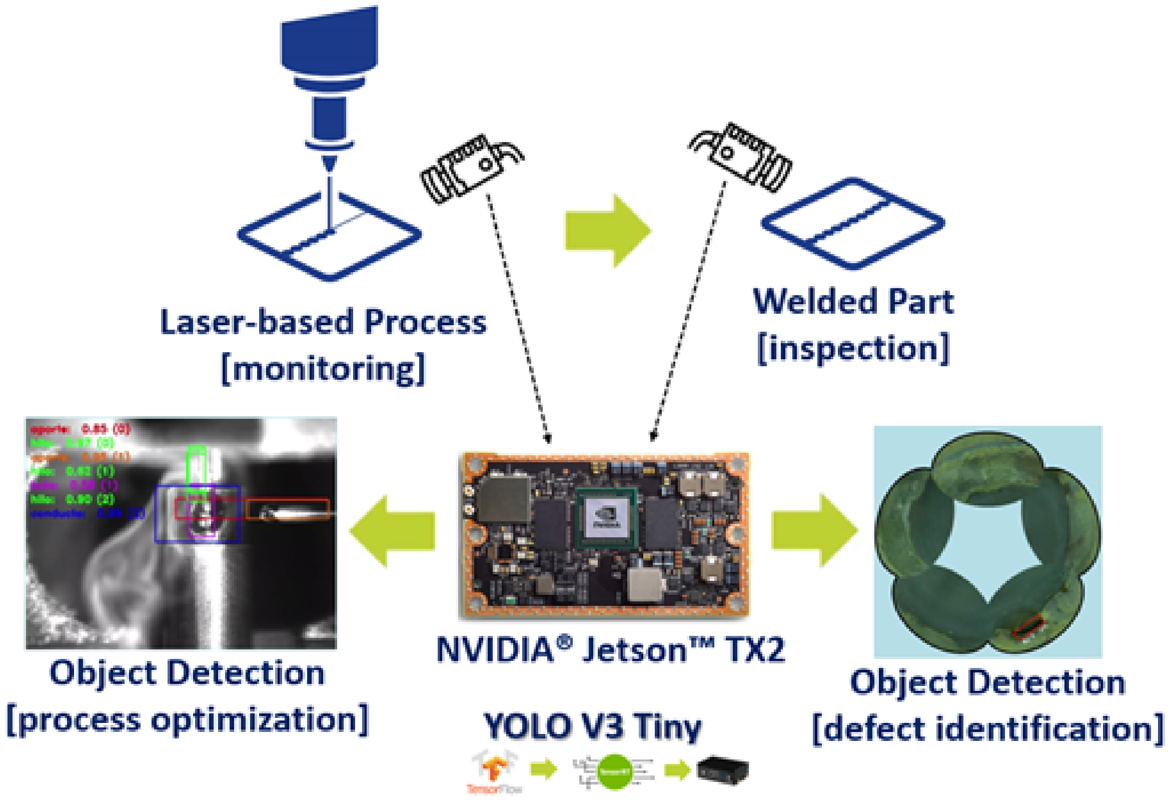


Figure 9: LORTEK’s laser-based object detection DL applications and toolchain structure.

In this approach, the deployment of DL algorithms was done using a TensorFlow and NVIDIA TensorRT flow –very similar to the one proposed in VEDLIoT–

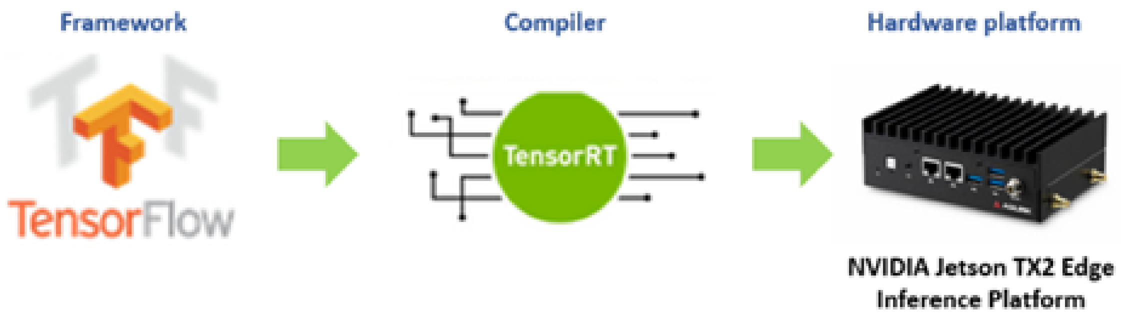


Figure 10: LORTEK's image classification and object detection DL's architecture workflow

Direct application of this structure for thermal and visible image classification and segmentation in laser-based processes count with some drawbacks. First, no optimization engine was used, limiting the real performance of the model in use. Second, and more importantly, this workflow was not implemented using a unique framework and the deployment of DL models in industrial environments became a time-consuming task. Therefore, the use of the VEDLIoT toolchain helps to overcome the drawbacks explained above.

All in all, the development of the BEAM-IDL project for laser beam shape IR and visible image classification and identification was grounded in EXOM's and LORTEK's experience in the development and deployment of monitoring systems, control devices and industrial DL algorithms for laser welding applications. The combination of both expertise supported by VEDLIoT's advancements in algorithm optimization and framework standardization duties.

The direct result of the BEAM-IDL project is a DL-based embedded solution (hardware and firmware/software) capable of being integrated into laser heads with visible and/or thermal monitoring cameras for beam-shape recognition purposes. For that, a complete DL pipeline to automate the workflow of image pre-processing, model training and evaluation, and model deployment to embedded hardware was developed.

Implementation

Hardware and software architecture design

Based on the previous information a hardware design based on the commercialized dual beam laser Wobble-Head by EXOM Engineering is proposed. The main components are summarized in the next Image.

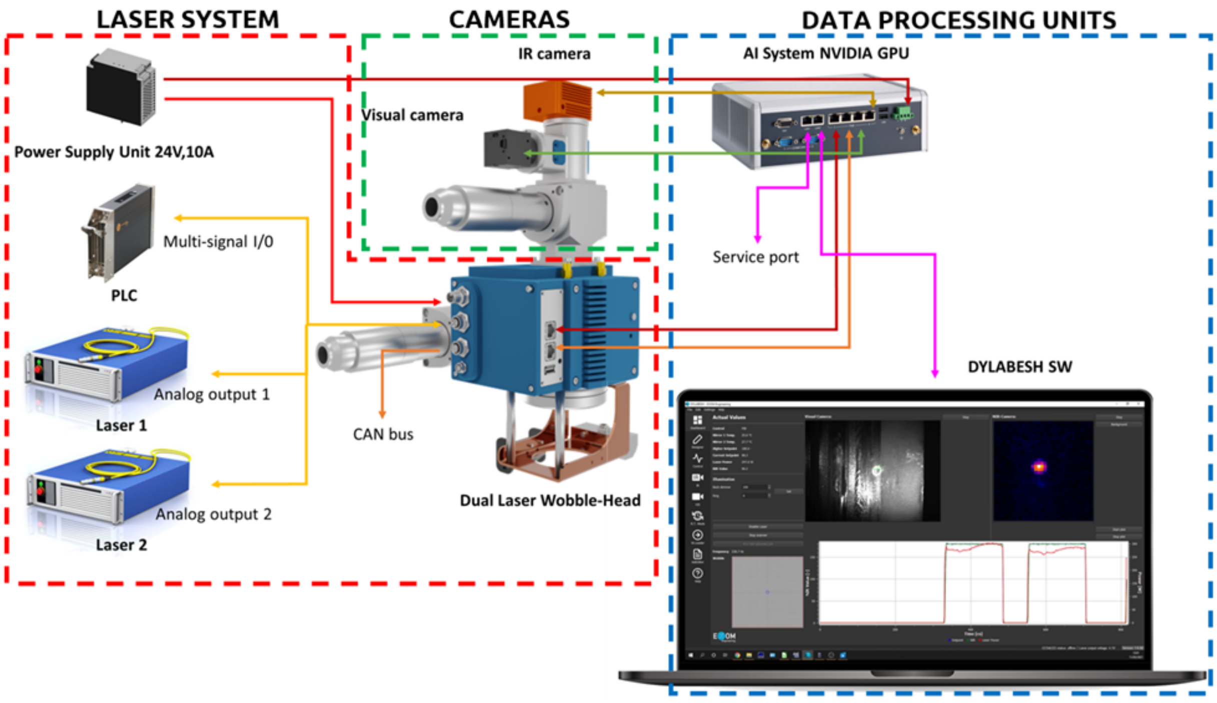


Figure 11: Hardware design and interfaces between the components

The laboratory set-up is represented in Figure 12 (left), together with initial results done on a steel tube Figure 12 (right).

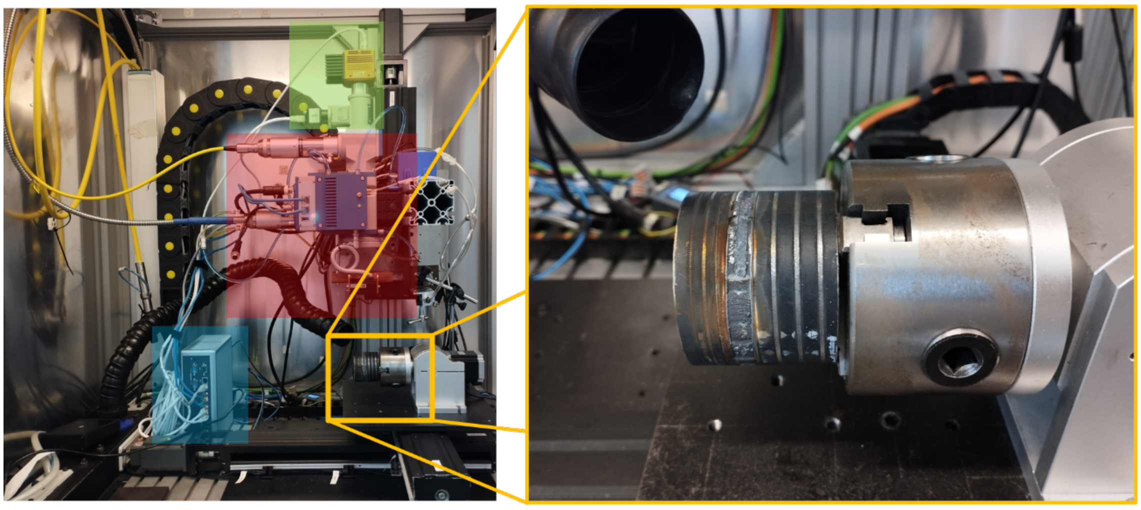


Figure 12: left) - Laboratory Set-up with the main components inside the machine; right) - Process results during setting up

As shown in Figure 11, the proposed system can be divided into three components/sub-systems.

Laser sub-system:

Compose by the Dual Laser Wobble-Head by Exom Engineering and accessories necessary for the start-up of the laser welding process. Multiple analogue and digital GPIOs will be used, here is a list of some of them:

- 1 USB3.0 Port
- 2 Ethernet: Gigabit Ethernet transceiver PHY (1 PoE)
- 5 Analog inputs: 20 kSPS, 0-10 VDC, 10 bits resolution
- 1 Analog output: 0-10 V (adjustable), 12 bits resolution, 50 mA max.
- 1 Digital input: 24 VDC
- 2 Digital outputs: 24 VDC, 500 mA max.
- 4 Digital outputs, 5 VDC, 10 mA, PWM Mode available (1 MHz max. Frequency)
- 2 CAN interfaces: supports 1 Mb/s operation

Camera module sub-system.

Two monitoring cameras, one for the visible spectrum and another for the infrared spectrum. The idea is to take advantage of the ability of the laser head to accommodate two cameras and maximize information extraction from the welding process. On one hand, a visible camera gives us information during the process about the interaction of the laser beam with the material being welded. On the other hand, the infrared camera extracts heat/energy distribution information around the welding zone. These two extracted features will change if the wobble strategy used in the welding process change.

In order to extract data from both camaras simultaneously the optical module was adapted implementing a dichroic mirror that separates the VIS (visible) spectrum from the IR wavelength, as it is shown in the Figure 13, additionally the lens module was adapted implementing a larger focal lens on the module, the purpose of the new optical design was to enhance the image spatial resolution captured by the cameras and adapt the camera module magnification to detect very small events during welding process.

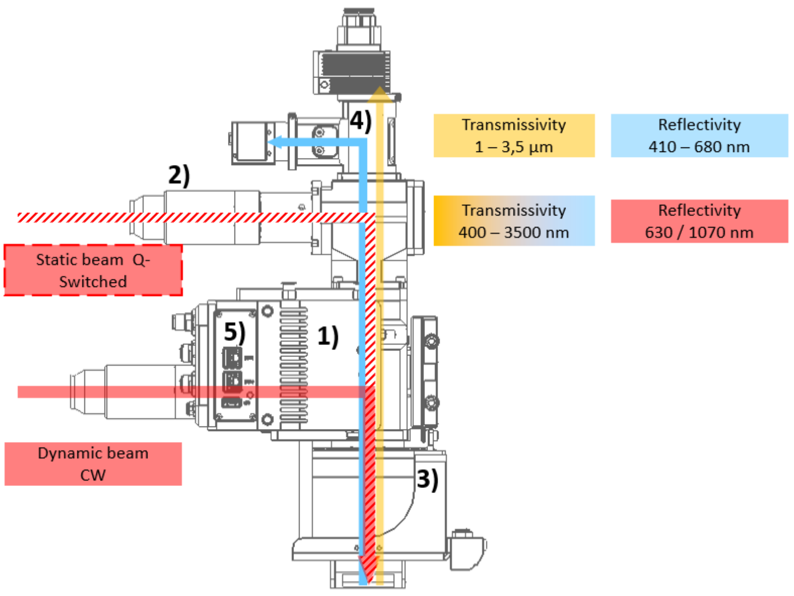

4)
2)
1)
5)
3)
Transmissivity 1 – 3,5 µm
Reflectivity 410 – 680 nm
Transmissivity 400 – 3500 nm
Reflectivity 630 / 1070 nm
Static beam Q-Switched
Dynamic beam CW

Figure 13: Dual laser welding head optical setup. 1) Scanner module, 2) Beam bender module, 3) Focusing module, 4) Camera module, 5) SdLED control card.

Detailed camera models and characteristics are shown in Table 1.

Table 1: Camera model and characteristics

	Visible Camera	Infrared camera
Model	Lucid Vision Lab TRI016S-MC	NIT Tachyon 16k+
Sensor	Sony IMX273	MWIR 1-5 µm
Resolution (px)	1440x1080	128x128
Pixel size (µm)	3.45x3.45	50x50
Maximum frame rate	70	4000
Communication interface	GigE Vision 2.0 (GeniCam compatible)	GigE Vision 2.0 (GeniCam compatible)

Both cameras were connected to the data processing unit using GigE interface, so available bandwidth was considered to define the frame rate and pixel format of the visible camera. The visible camera can acquire in both Mono8 (8-bit depth) and Mono16 formats. Acquiring at full resolution these were the required bandwidths.

- 1440x1080px, Mono8, 60fps ~ 90 Mb/s
- 1440x1080px, Mono16, 60fps ~ 180 Mb/s (not compatible with the actual solution)
- 1440x1080px, Mono16, 30fps ~ 90 Mb/s

Considering that the maximum value of GigE Vision standard [28] is 125 Mb/s, acquiring at mono16 to achieve maximum image quality limited the acquisition frame rate. To avoid this drawback and considering that the interest zone within the image was limited, the use of ROI features to acquire at least 60 fps using Mono16 pixel format was a practicable solution.

Due to the low resolution of the infrared camera, these limitations did not apply to it.

Data processing units.

As previously stated, cameras were connected to the data processing unit using GigE interface. The main component of this system was a Fanless Edge AI System powered by an Nvidia Jetson AGX Xavier [29] [30] . Here the µRecs platform developed within the VEDLIoT project can fit well [31]. The main purpose of this unit is to acquire both camera streams and make DL model inferences to achieve the required results (explained in detail in software components). These results are shown in a standard pc using Dylabesh Software (commercial software developed by EXOM Engineering and commercialized with their laser head).

Software components

Once the hardware design has been explained, the architectural design of software components are introduced. First, some acquisition software needed to acquire both visible and infrared camera images. In production, this image information is first pre-processed and then evaluated with the chosen Deep Learning algorithm. During the development phase, a dataset for Deep Learning model training and evaluation are created, so both camera images must be saved properly (explained in Evaluation and Characterization section). For this, a database schema to save all welding process parameters and acquired image metadata was defined. This database feed an annotation tool (coco-annotator[32]). Once a dataset was created an automated deep learning model training, evaluation and optimization workflow was developed, as well as a tool for deploying these models to Edge IA devices. To achieve these goals, tools developed within the VEDLIoT project were very helpful. The Kenning Framework [33] could fit these needs, so the idea was to use this framework to develop an automated deep-learning workflow. Optionally, EmbeDL optimization toolkit is of interest to this project.

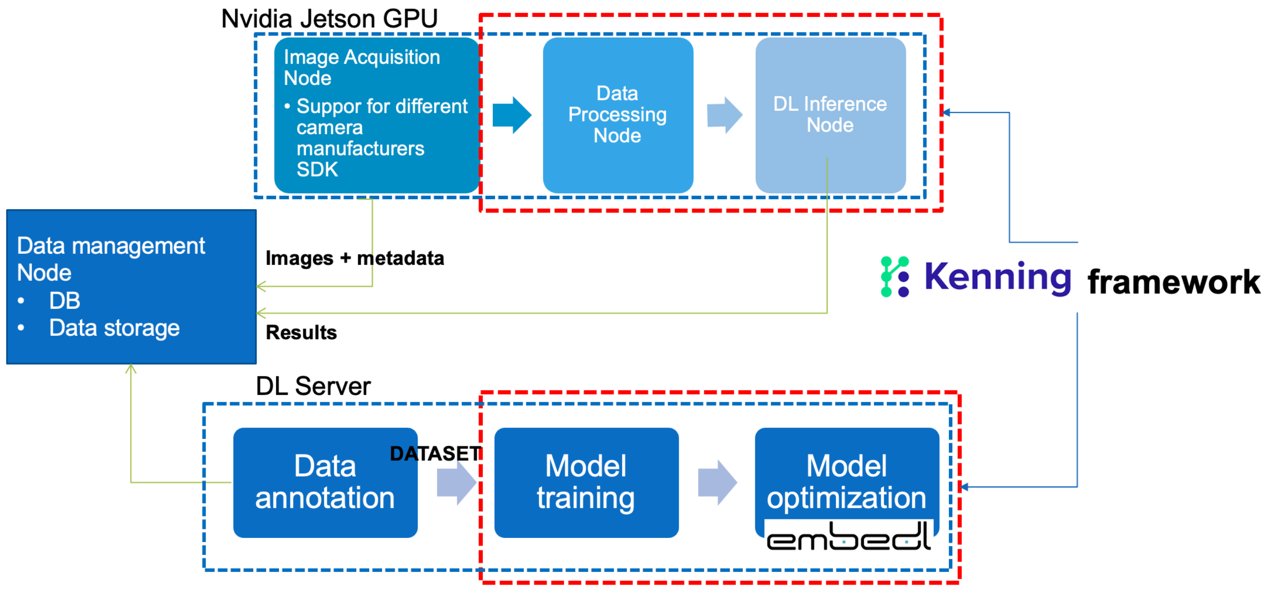


Figure 14: Software architectural design

The idea was to develop each software component separately and deployed using docker containers. For the image acquisition node, camera SDK was used to acquire image data in an automated way for phase 3 pre-industrial prototype. The goal was to acquire both cameras at 60 frames per second.

In the development phase of the project (Phase 2), no specific software was developed for image acquisition from cameras. This acquisition was carried out using commercial software provided by the cameras manufacturers. Instead, in the phase 3 of the project, we try to test the developments done in development phase 2 in a pre-industrial operational environment, so **we need to automatize the image acquisition process**. For this, we developed an application based on Aravis C++ Library [34] that captures the frame from both cameras and acts like a TCP server to serve these captured images to any connected client.

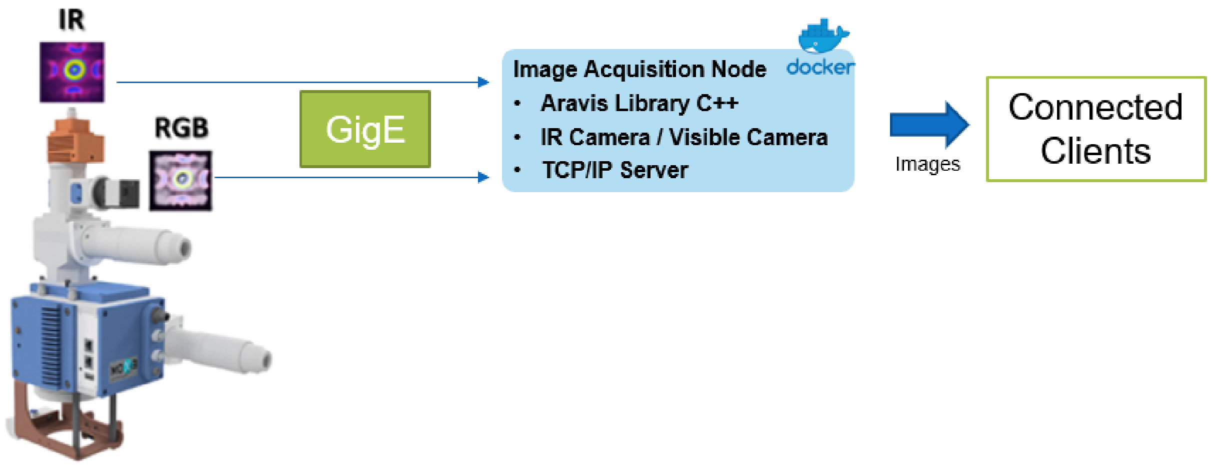


Figure 15. BEAM-IDL project acquisition node.

As previously mentioned, image data was stored properly to create a dataset for DL model training and evaluation. Depending on the task to be solved by the DL model the annotation tool used varies. For example, for object detection or segmentation tasks open-source COCO annotator project [32] was be used.

The development of an automated DL model training, evaluation and optimization workflow is the key component of this project. For this, we use Lortek´s ***deep-learning-framework***. Basically, is a framework created to simplify the dataset loading and processing, and the training of different model architecture (Figure 16). The idea is very similar to Kenning Framework that use a configuration file to define the whole training and deployment workflow. In case of deep-learning-framework, is much simpler than Kenning [34] and didn´t offer so much option. It´s only prepared to use Tensorflow [35] to train models for classification and object-detection tasks, and input dataset has to be saved in Tensorflow *TFRecord* format. In a near future, it will be able to also train segmentation architectures but for the scope of this project we decide to pause this development. It will be interesting in a future to try to merge Kenning developments with ***deep-learning-framework*** by Lortek. As we will explain in next section, classification models were trained for infrared images and object detection model for visible images.

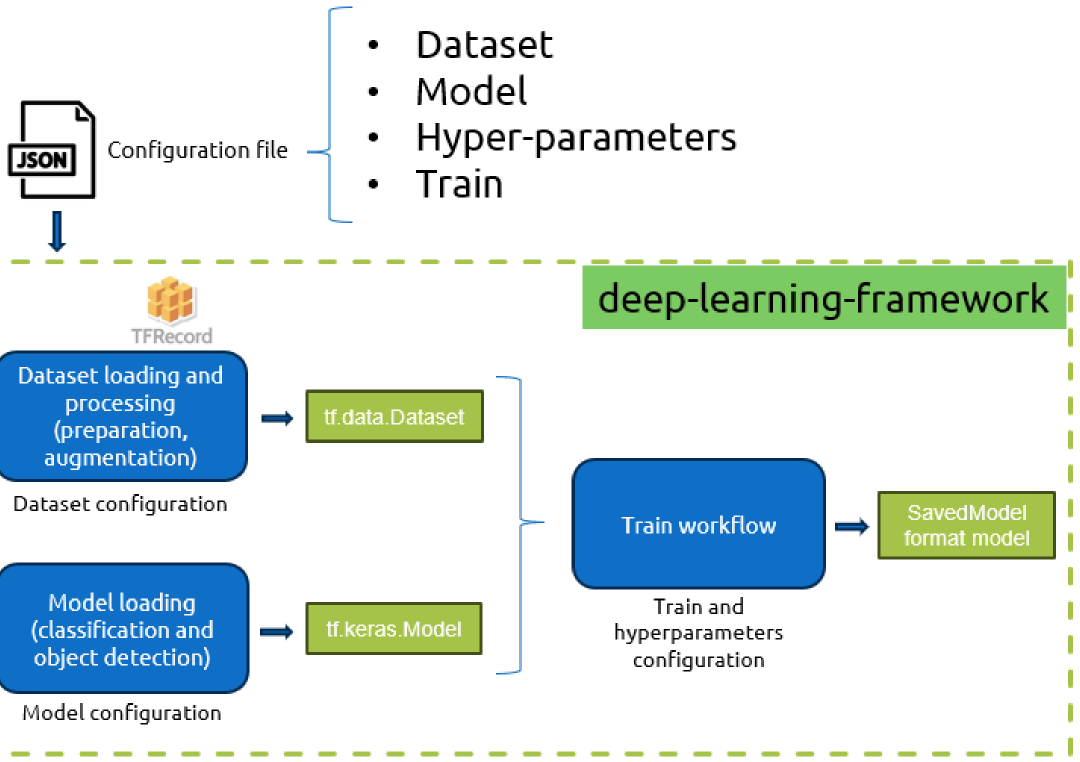


Figure 16. Lortek´s Deep-learning-framework workflow.

Using the *Configuration file,* we configure the dl model training workflow. First, we define the dataset to use (which *tfrecord* files to load) and the processing to perform to this dataset (normalization type and augmentations techniques). Using *model* section of the *configuration file*, we define the architecture to use and the parameters to build the model if needed. Also, we can define a weights file to load the weights of a pre-trained model. With the prepared dataset and the build model we perform the training using the parameters in both *hyper-parameters* section and *train* section of the *configuration file.* After training, we end up with a trained model saved in Tensorflow SavedModel format.

For production phase, trained model will be optimize using Nvidia TensorRT in target hardware (in this case Nvidia Jetson Xavier AGX module). This optimized model is then deployed for inferencing using Lortek´s ***deep-learning-gpu-server***. Basically, it is a TCP/IP server deployed in target hardware that accept clients that send images/data to analyze. In the case of this project, the acquisition node is the program that act as a server (Figure 15), so we modify ***deep-learning-gpu-server*** to act also as a TCP client.

In the next diagram we summarize the software components develop within BEAM-IDL project.

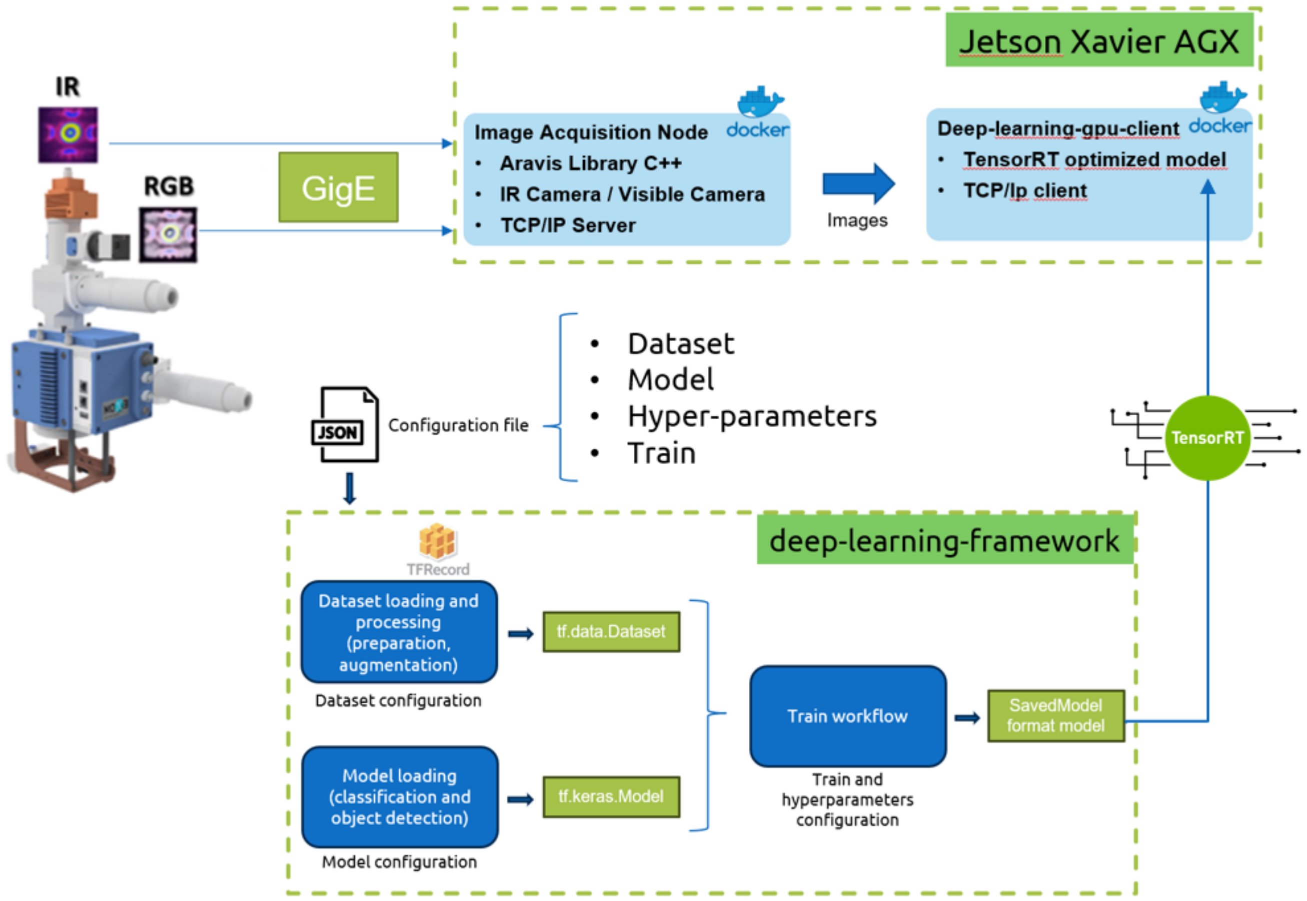


Figure 17. Software developed within BEAM-IDL project.

Evaluation and characterization

Dataset generation

The objective of BEAM-IDL project is to identify the configured laser beam shape in aluminum-aluminum laser welding process using Machine Learning algorithms that extract features for both captured images (thermal and visible) and classify to a laser beam shape. This can be done using supervised Deep Learning (DL) architectures that has demonstrated remarkable success in a multitude of tasks ranging from image classification to natural language processing.

To train a DL architecture is mandatory to have a collection of representative data for the task that want to be solved. There are several popular datasets commonly used in Deep learning research and applications covering different task solving, for example, **ImageNet** [36] for classification and object detection tasks or **COCO** [37] for object detection, segmentation and captioning. Other datasets for more specific tasks can be found in Kaggle datasets, Google Dataset Search, Dataset list or Open Images Dataset [38]. This generic dataset are used for evaluation of new DL architectures, or for the technique called Transfer Learning. Transfer learning is a technique in the deep learning domain that leverages the knowledge gained from one task to improve the performance on a different but related task. The idea is to use a pre-trained model, usually on a dataset mentioned above, as a starting point and fine-tune it for the target task. This approach can save significant time and computational resources, especially when the target task has limited labeled data. As we state, to use Transfer Learning, the task we want to solve should be related to the task these generic datasets were created for.

In our case, the objective is to retrieve data from laser welding process of aluminum-aluminum welding. In this welding processes, different parameters can be tuned to improve the welding quality as we state in deliverable 1.1. As we want to identify the configured laser beam shape, different laser beam shape (or wobble pattern) will be used, maintaining the other variables (laser power, welding velocity, etc.…) constant. Some common wobble strategies used in laser welding:

- Circular wobble: The laser beam follows a circular path around the central axis of the weld. This method distributes the energy more uniformly across the weld area, leading to better penetration and reduced porosity. Circular wobble is especially useful for welding materials with a high thermal conductivity or for butt joint welding.
- Spiral wobble: In this technique, the laser beam follows a spiral path around the central axis of the weld. Spiral wobble is useful when welding materials with a wide melting range or when dealing with complex joint geometries, as it ensures more uniform heat distribution.
- Zigzag wobble: The laser beam moves in a zigzag pattern along the weld line. This method can help reduce the heat-affected zone and minimize distortion in thin materials. Zigzag wobble is often used in lap joint welding.

- Linear wobble: The laser beam oscillates linearly along the weld line, moving back and forth. This strategy is useful for materials that are prone to cracking due to high cooling rates, as it helps to reduce stress and heat input.
- Scanning wobble: The laser beam scans across the weld area in a predefined pattern, such as a sinusoidal or rectangular shape. Scanning wobble can help control the size of the melt pool, ensuring a consistent and high-quality weld.

In the scope of this project, we choose three wobble pattern family: circular, ellipse and lemniscate or infinity pattern. In the case of circle and ellipse, 4 wobble pattern were used changing the amplitude to achieve a more challenging identification problem. In total, 5 wobble pattern were used (Figure 18).

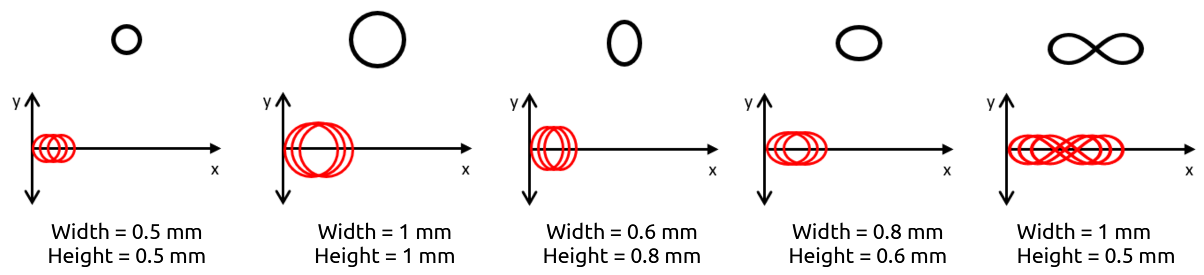


Figure 18. Selected wobble pattern for BEAM-IDL project identification algorithm.

Using these wobble patterns, the next Design of Experiments was design.

Wobble	Width [mm]	Height [mm]	Laser Power [W]	Feed rate [mm/s]
Circle	0,5	0,5	1000	10
Circle	1	1	1000	10
Ellipse	0,6	0,8	950	10
Ellipse	0,8	0,6	950	10
Lemniscate	1	0,5	1000	10

Table 2. Design of experiments of BEAM-IDL project.

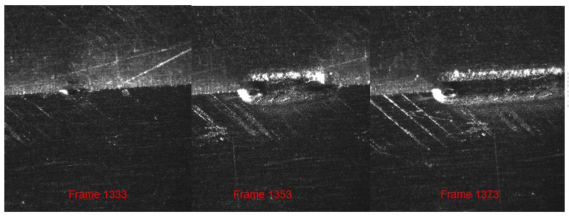
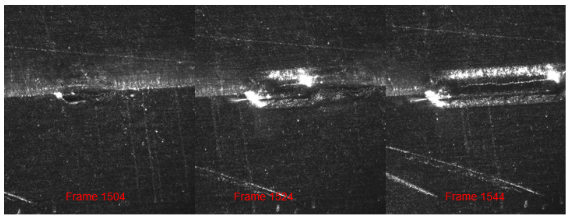
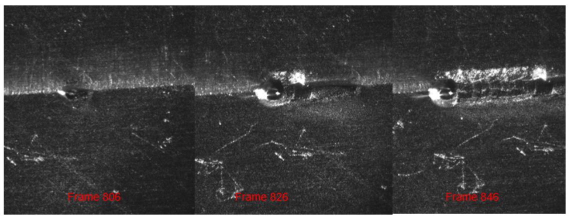
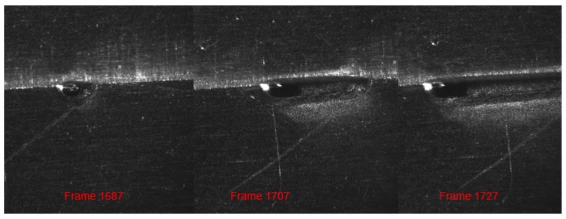
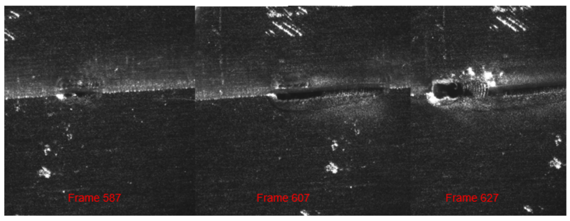
With this DoE (Table 2), several tests were carried out in EXOM Engineering facilities, and thermal and visible images were acquired. In this development stage, no software was developed for image acquisition, and commercial software was used for both cameras image acquisition (Table 3).

	Visible Camera	Infrared camera
Model	Lucid Vision Lab TRI016S-MC	NIT Tachyon 16k+
Sensor	Sony IMX273	MWIR 1-5 µm
Resolution (px)	1440x1080	128x128
Pixel size (µm)	3.45x3.45	50x50
Maximum frame rate	70	4000

Communication interface	GigE Vision 2.0 (GeniCam compatible)	GigE Vision 2.0 (GeniCam compatible)
Acquisition Software	Lucid ArenaView GUI	NIT SOFTWARE SUITE

Table 3. Visible and thermal cameras used in BEAM-IDL project. Characteristics and used commercial software for image acquisition.

In the next Figures we can see examples of acquired images with both cameras of Table 3.

	
Circle width=0.5 height=0.5	Circle width=1 height=1
	
Ellipse width=0.6 height=0.8	Ellipse width=0.8 height=0.6
	
Lemniscate width=1 height=0.5	

With these visible images, we try to identify the wobble pattern using object detector models. We will extent this in next sections.

Thermal data acquired with the infrared camera is stored in *.dat* files by NIT Acquisition software. For convenience, each acquired frame stored in *.dat* file is saved in *.jpg* format. In this conversion, each raw frame is pre-process. Finally, with these pre-processed frames a **dataset** is created.

Pre-processing of raw frame is done in the conversion stage. First, a background correction is done subtracting a mean background image acquired before the start of the welding process. Then, each frame is normalized between {0, 1}, and a ROI is defined to reduce the influence of non-interest zones of the image. This is because the field-of-view of the thermal image is much larger than the area of the welding area (Figure 19).

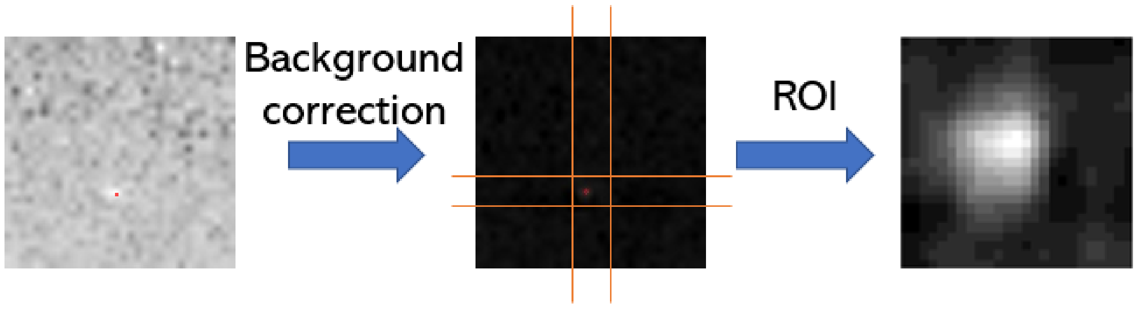
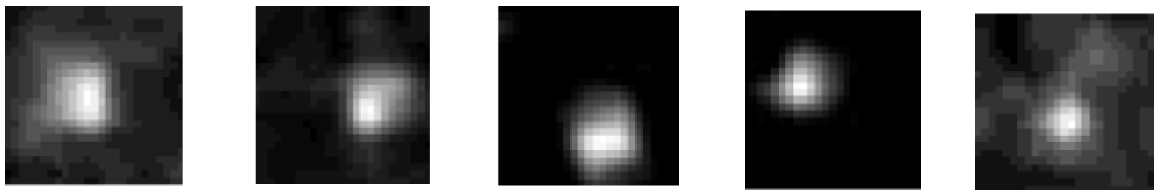


Figure 19. Thermal image pre-processing workflow.

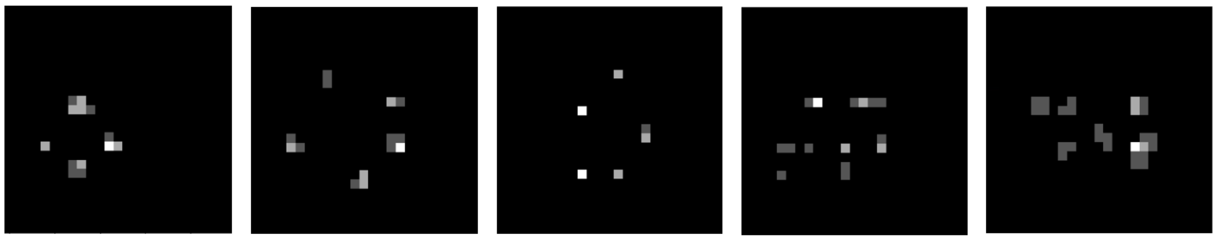
Using this process, we collect 2400 thermal frames for each of the wobble pattern used in this project and shown in Table 2. Each frame by itself doesn´t contain any differentiable feature between the wobble patterns as we can see in Figure 20.



Circle w0.5 h0.5	Circle w1 h1	Ellipse w0.6 h0.8	Ellipse w0.8 h0.6	Lemniscate w1 h0.5

Figure 20. Example of an acquired frame for each of the wobble pattern.

Instead, considering consecutive N images gives an idea of the pattern that the laser beam is making. In Figure 21, we try to demonstrate that showing in one image the maximum intensity pixels of each frame in a 12 consecutive images for each wobble pattern.



Circle w0.5 h0.5	Circle w1 h1	Ellipse w0.6 h0.8	Ellipse w0.8 h0.6	Lemniscate w1 h0.5

Figure 21. Representation in one image of the maximum intensity pixels in each image of the 12 images sequence.

This problem can be address in different ways (Figure 22).

1. Combine N consecutive images in one image to consider the time dependency of the problem. This can be understood as a pre-processing stage before feeding these processed images to the DL algorithm for wobble pattern identification. In this case, a simple Convolutional Neural Network (CNN) for classification task con be used.
2. As we will see in the next section, there are many DL architectures that admit an images sequence (a video) as input and can learn spatio-temporal features from this. This type of architectures can be used in our problem too.

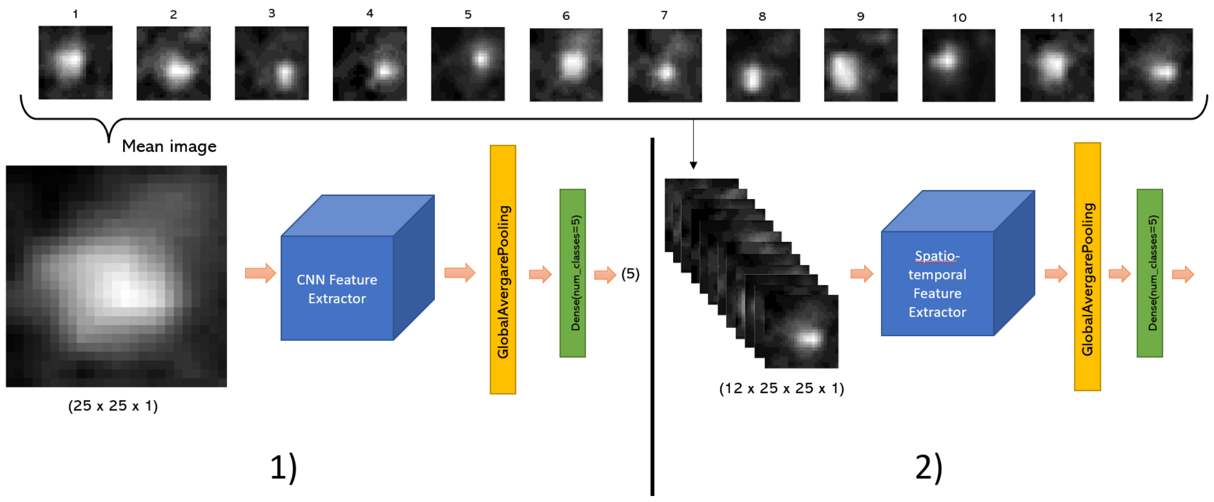


Figure 22. Different approaches to address BEAM-IDL problem.

To test different solutions (different DL architectures) to address the problem, two datasets were created for the case of the thermal images. One dataset (**dataset 1**) averages a 12 consecutive image sequence in one image (25 x 25 x 1). In the other one (**dataset 2**), 12 consecutive images were stored as a 4-dimensional tensor (12 x 25 x 25 x 1). In both datasets, there are 200 samples for each wobble pattern, and a typical 80:20 ratio was used for training and validation.

DL architecture

As we state in previous section, the objective of BEAM-IDL project is to identify the wobble pattern. This wobble pattern can be seen in consecutive images acquired during the welding process (ref figure of previous section), so we can say that it is a dynamic process that involve an analysis and prediction of time-dependent data. In such applications, the goal is to learn patterns and relationships in the data that change over time. Deep learning techniques have been successfully applied to dynamic processes with time-dependent data, with several architectures developed over the last years focusing on machine translation tasks, time series data forecasting or anomaly detection.

- **Recurrent Neural Networks (RNN)** [39] are designed to model sequences and handle temporal dependencies. They maintain an internal state that can capture information from previous time steps, making them well-suited for dynamic processes. RNNs are often used in time series forecasting, speech recognition, and natural language processing tasks.
- **Long Short-Term Memory (LSTM) networks** [39] [40] [41] are a specific type of RNN that addresses the vanishing gradient problem that vanilla RNN suffer, enabling them to capture long-term dependencies in sequences. They are particularly useful for modeling complex dynamic processes where relationships between elements might span across large time intervals. LSTMs have been widely used in applications such as machine translation, anomaly detection in time series data, and video analysis.
- **Gated Recurrent Unit (GRU) networks** [42] are another type of RNN that simplifies the LSTM architecture while maintaining the ability to capture long-range dependencies. GRUs have been applied to various dynamic processes, including speech recognition, time series prediction, and gesture recognition.
- In the last years, another promising architecture called **Transformers** [43] has emerged. Transformers use self-attention mechanisms to model relationships

between elements in a sequence, regardless of their positions. This architecture has shown significant improvements in natural language processing tasks, like machine translation and text summarization, and has been applied to other dynamic processes like time series forecasting and speech recognition.

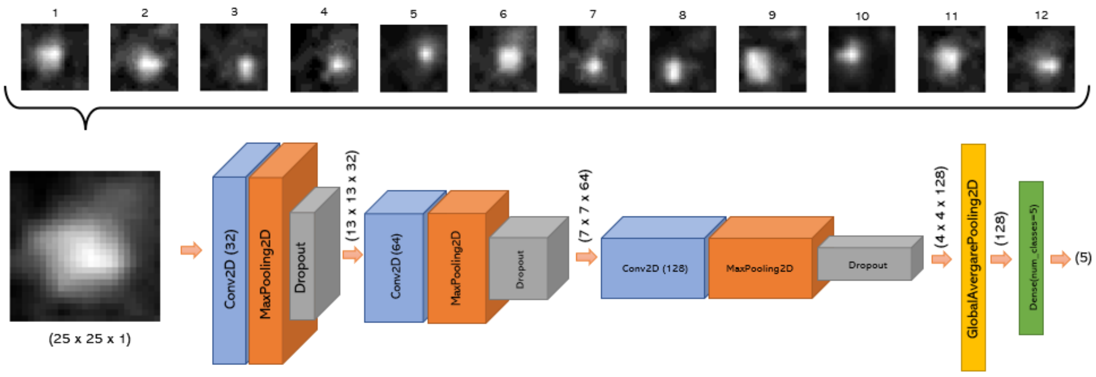
As we state, the majority of this architectures were developed for 1D data like text sequences or time-series data, but several of these architectures evolve to accept **image sequences** (a stack of images) in order to learn spatio-temporal features to capture both spatial information (e.g., objects, shapes, textures) and temporal information (e.g., motion, change over time). Here we mention some of these works.

3D Convolutional Neural Networks (3D CNNs) [44] extend traditional 2D CNNs by adding a third dimension (time) to the convolutional layers. They can learn both spatial and temporal features from a sequence of images or video frames. **Convolutional LSTM networks** [45] combine the spatial feature learning capabilities of CNNs with the temporal modeling capabilities of LSTMs. They replace the fully connected layers in LSTMs with convolutional layers, allowing the model to capture both spatial and temporal information. In **Two-Stream CNNs** [46], two separate CNNs are used – one for spatial features and one for temporal features. The spatial stream processes individual frames to extract spatial information, while the temporal stream processes optical flow information to capture motion. The outputs of both streams are combined to make predictions. **Spatio-Temporal Graph Convolutional Networks (ST-GCNs)** [47] model the spatial and temporal relationships in video data as a graph, where nodes represent image regions, and edges represent the connections between them. Graph convolutional layers are used to learn spatial features, while temporal information is captured by connecting nodes across different time steps. **Recurrent CNNs** [48] combine 3D CNNs with RNNs to learn both spatial and temporal features. A 3D CNN is used to extract spatial features from each frame in a video sequence, and the resulting feature maps are fed into an RNN (such as an **LSTM** or **GRU**) to model temporal dependencies. In this work other variants for the extraction of spatiotemporal features are explored.

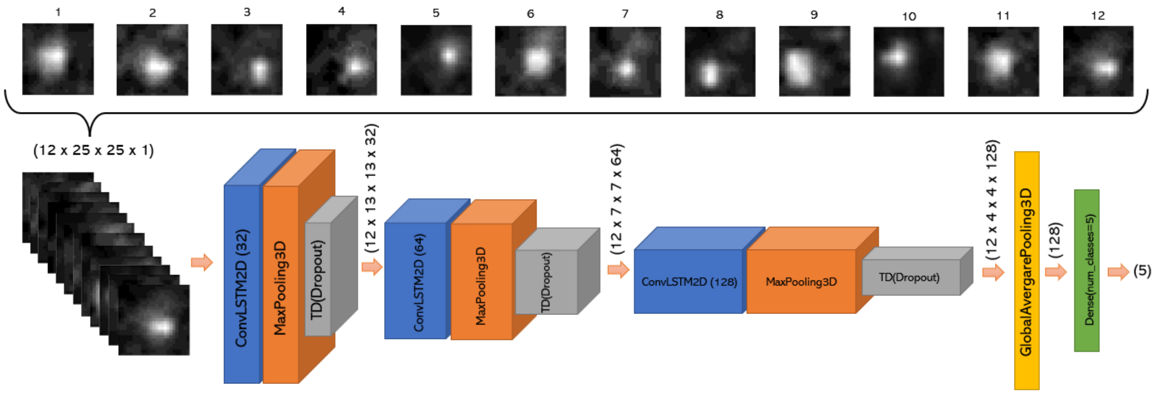
One of the main drawbacks of these approaches arise from the fact that using a stack of images create a bigger data volume as input to the DL model. This can create memory problems when training the model. Nevertheless, thermal image spatial dimension is very small (25x25), and despite a priori this could be a problem when training (less pixels, less information), in this case it can benefit the ability to train models that accept images sequence as input.

In BEAM-IDL project, three approaches have been tested for the case of thermal images.

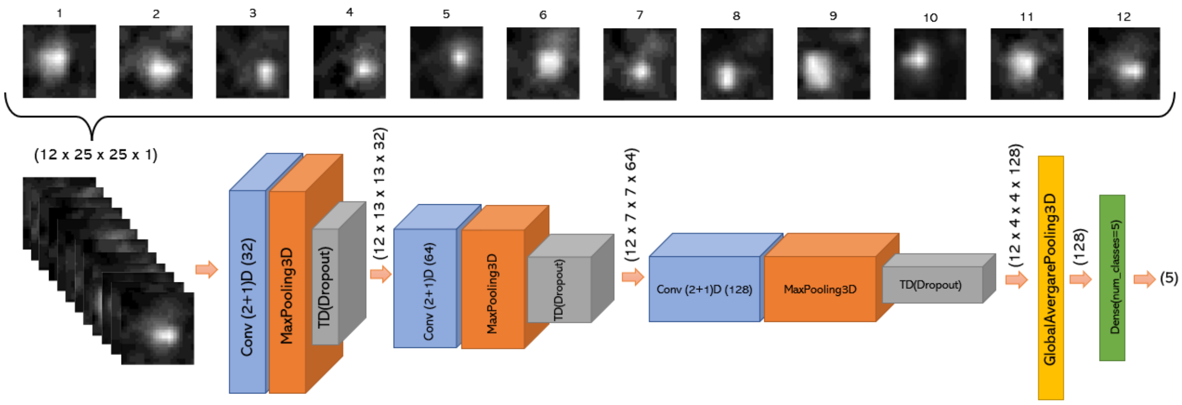
1. **Approach 1.** Use **dataset 1** with a simple Convolutional Neural Network designed using TensorFlow framework [35] to identify the wobble pattern. We use 3 blocks constructed with a 2d convolutional layer, a max polling layer and a dropout layer. The number of filters for the 2d convolutional layer are {32, 64, 128}, and the max pool layer use a pool size of 2 to decrease the spatial dimension to the half in both dimensions. The output of the last block is spatially average using a 2D Global Average Pooling layer and then a Dense layer of 5 units is used to get the final probability vector. Number of parameters = 18.981.



2. **Approach 2.** In this second approach, a simple network with Convolutional LSTM networks has been designed. In consist of three blocks, same as **approach 1**, replacing the 2d convolutional layers with a convolutional LSTM layer. As in the first approach, the output of the last block is average in both spatial and temporal dimension using a 3D Global Average Pooling layer, and a Dense layer of 5 units is used to get the final probability vector. In this case, the input is a 12 images sequence (**dataset 2**). Number of parameters = 1.145.477.



3. **Approach 3.** In [48] authors explore different ways of extract spatio-temporal features for Action Recognition task. The most obvious approach is to replace 2D convolutions of image feature extractors to 3D convolutions. Instead of this, they propose a "(2+1)D" convolution which processes the space and time dimensions separately (Figure X), using a 2d convolution for spatial dimension followed by a 1D convolution for spatial dimension. The main advantage of this approach is that it reduces the number of parameters comparing with a 3D convolution. For this reason, in this third approach we design the same architecture as in **approach 2** replacing the convolutional LSTM layers with custom (2+1)D convolutions. Number of parameters = 158.053.



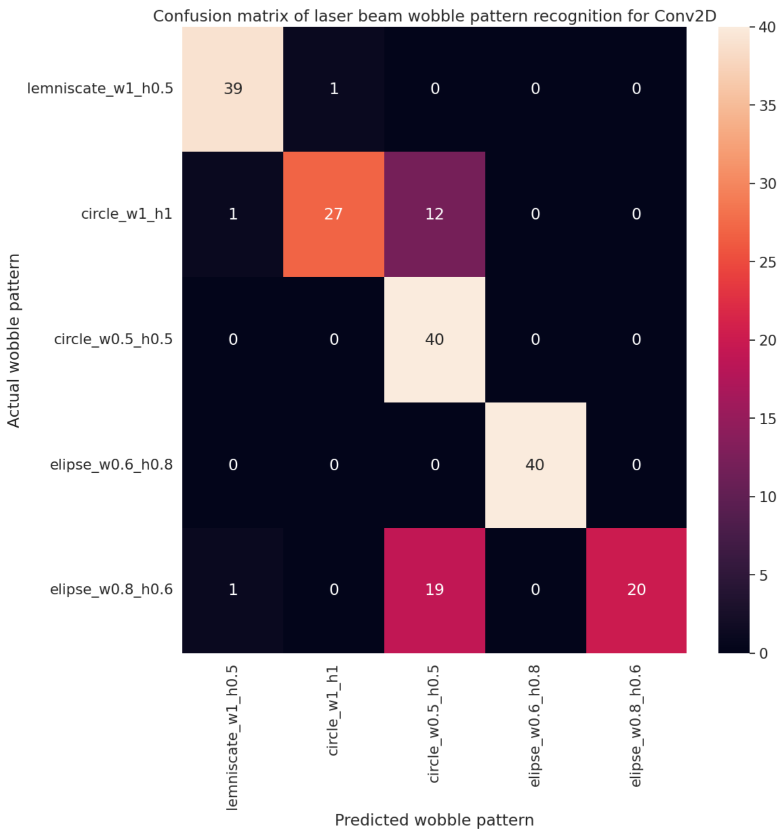
In the case of visible images, some proof of concept has been done using the latest YOLO family implementation, YOLOv8 [49]. The results are shown in the next section.

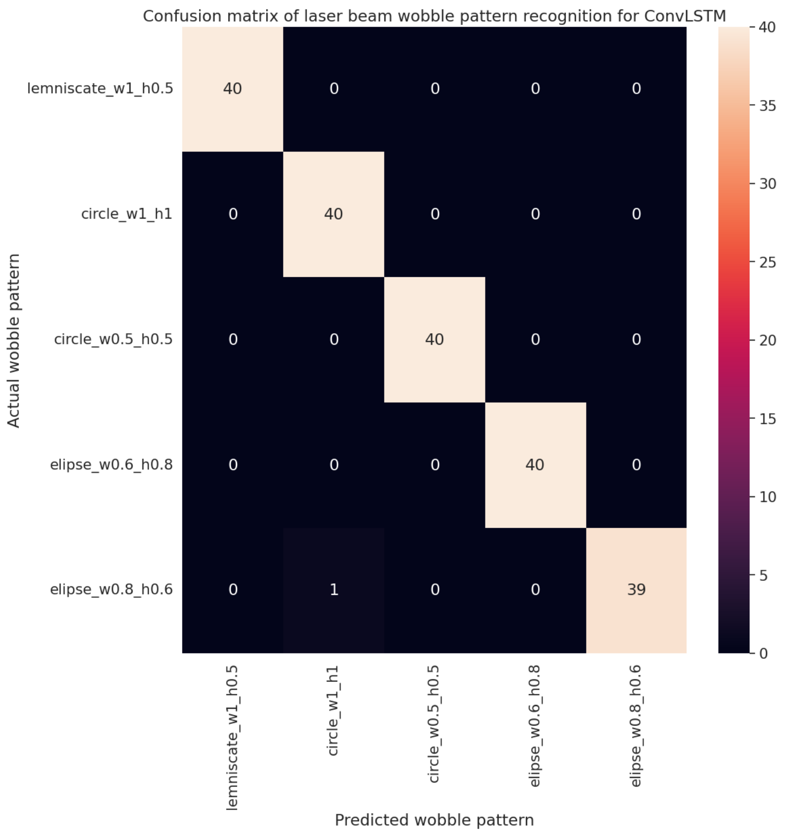
Training process and results

As we state, the three DL architectures for thermal images explained in previous section were designed using TensorFlow framework [35], and the training stage was carried out also using the same framework. For training, we use these hyperparameters and configuration.

- Batch size = 4
- Epoch number = 400
- Optimizer = **Adam** with Cosine decay learning rate scheduler and warmup phase.

In the next figure, we show the confusion matrix for each approach.





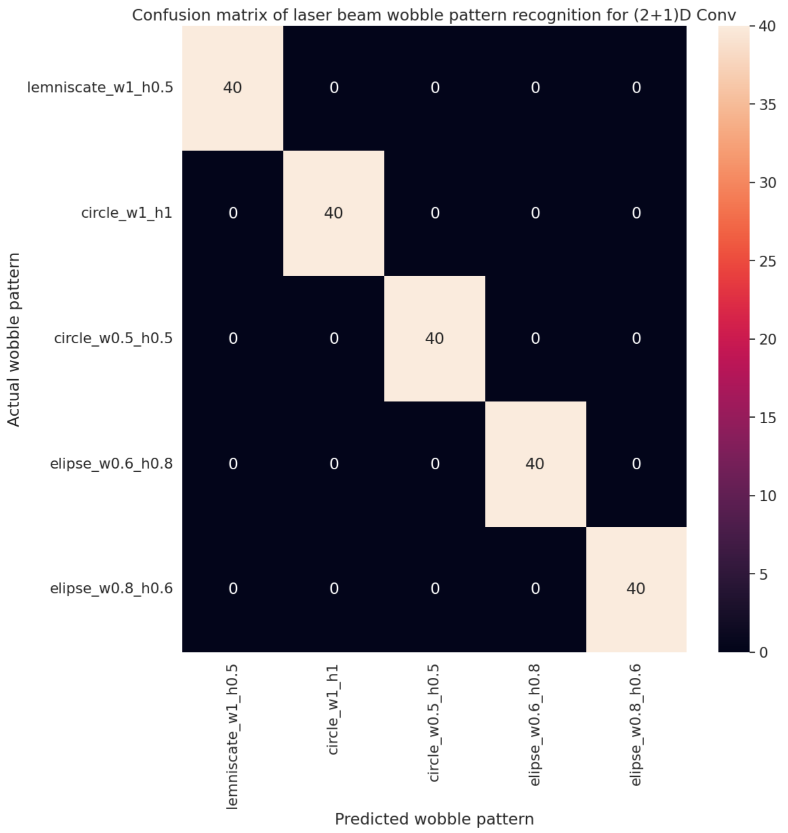


Figure 23. Confusion matrix for each DL model for wobble pattern identification in thermal images. From left to right approach 1, approach 2 and approach 3.

For the case of visible images, the used dataset is not the same, and only one circle type and one ellipse type is used. In the Figure 24 we share the confusion matrix for YOLOv8 model and in Figure 25 some prediction examples.

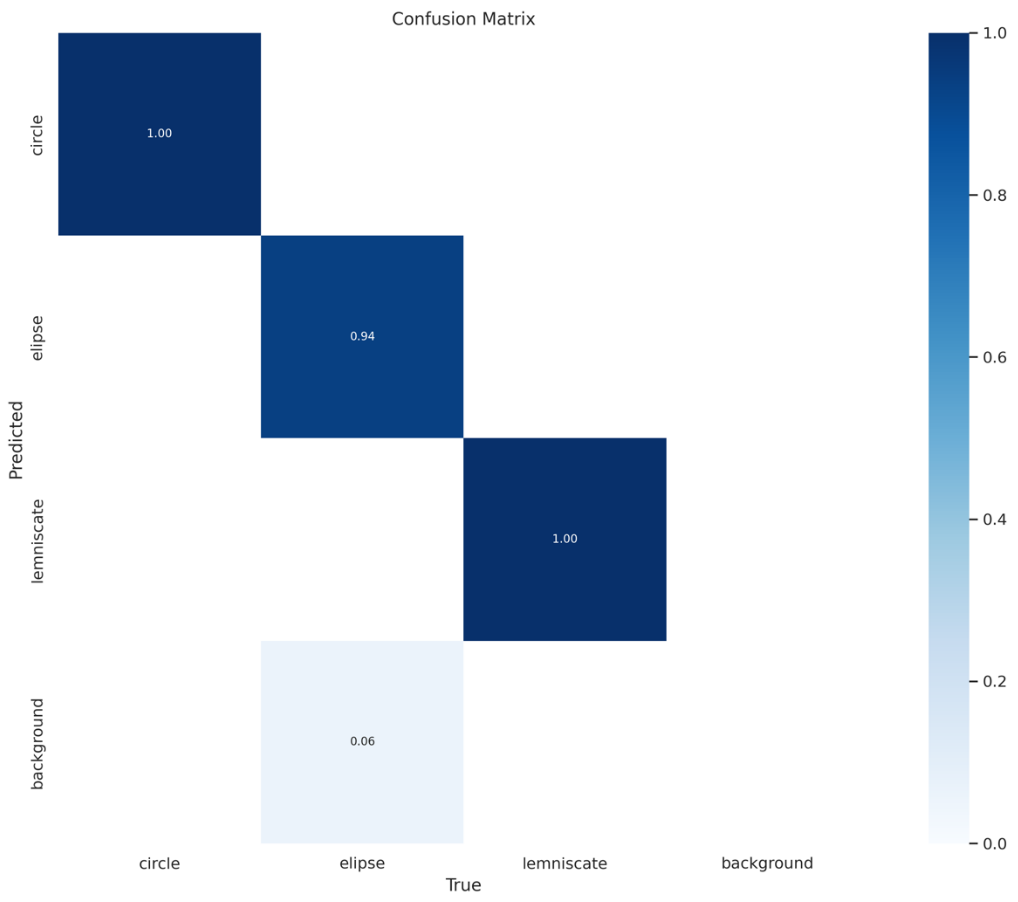


Figure 24. Confusion matrix of YOLOv8 model for wobble pattern identification in visible images.

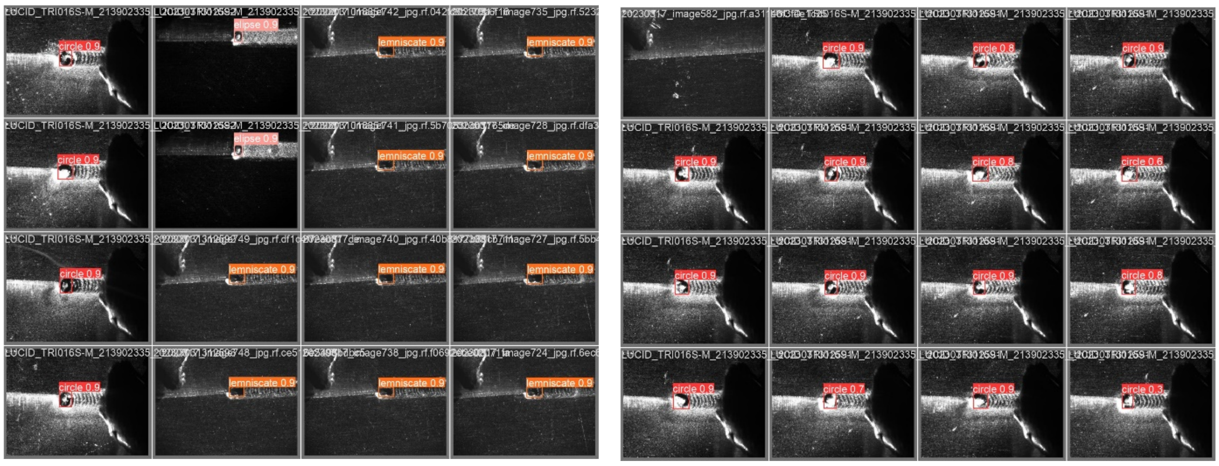


Figure 25. Prediction examples in visible images using YOLOv8 object detector.

Pre-industrial prototype

The developments described above have been tested in a pre-industrial environment in Lortek facilities as part of Phase 3 works. Lortek, has 3 laser welding cells prepared to carry out I+D projects. In the case of this project, one EXOM Engineering laser head was mounted in a KUKA robot in one of those cells. In Figure 26 we show this setup with the camera modules highlighted. In order to enhance visible image contrast on the VIS camera and avoid pixel saturation due to strong molted material glowing, a bandpass filter was mounted on the camera module, the process was at the same time externally illuminated using an external illumination laser source.

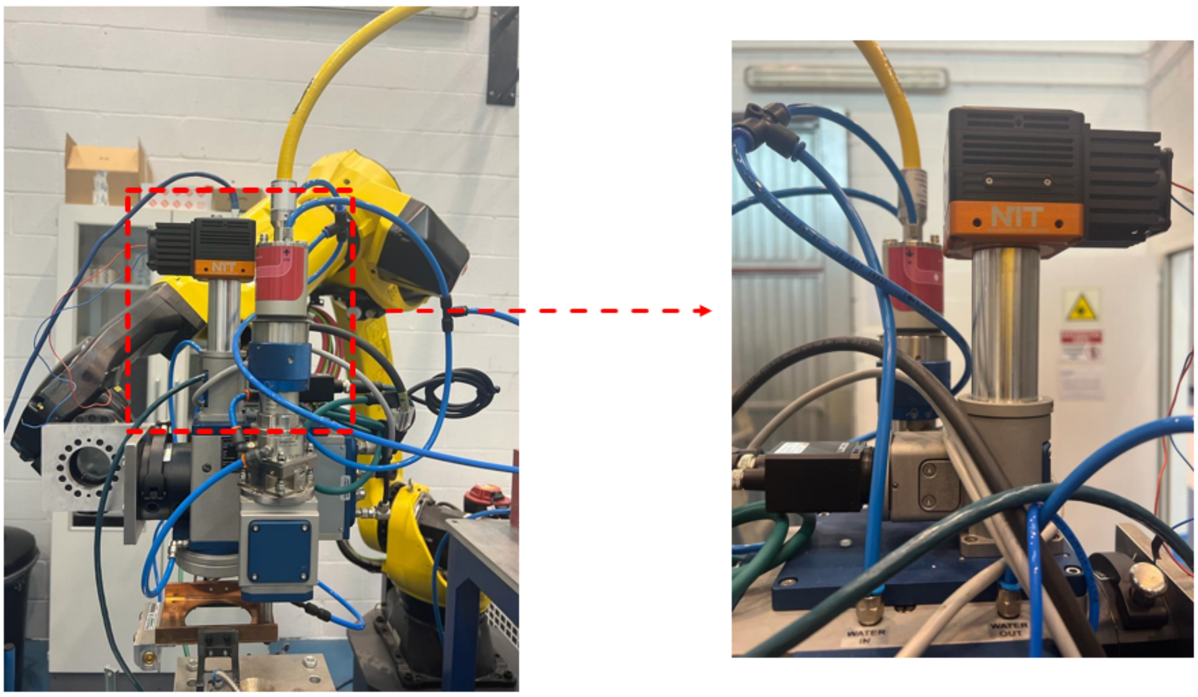


Figure 26. Pre-industrial prototype in Lortek´s laser cell. Detailed camera subsystem.

As we explained in implementation section, both cameras are connected to a Jetson Xavier AGX module using GigE connection. In this case, we use an ADLINK DLAP-401-Xavier with 2 GigE interfaces (Figure 27).

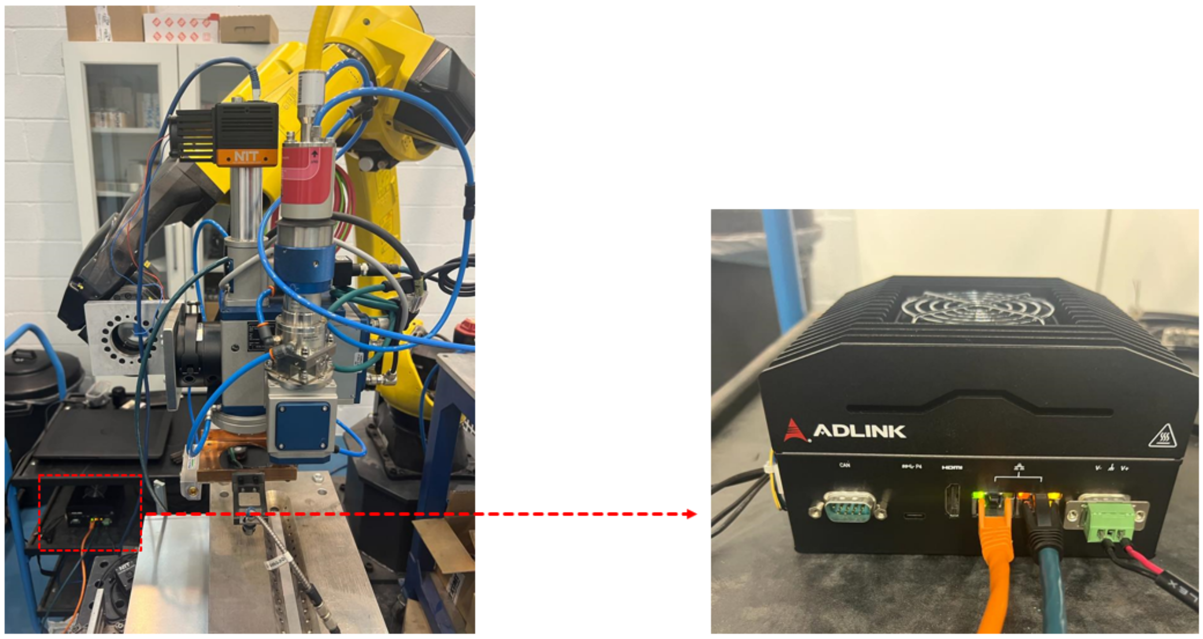


Figure 27. Pre-industrial prototype in Lortek´s laser cell with Nvidia Jetson Xavier AGX module.

In order to evaluate the developments done, we design a final batch of experiments based on the same DoE we use in the dataset generation explained in Implementation section. In this case, instead of using a specific wobble pattern in each welding, we try to change the wobble pattern at the middle of the welding. Specifically, we program 5 wobble pattern pairs shown in Figure 28 (small circle-big circle, vertical ellipse-horizontal ellipse, small circle-vertical ellipse, small circle-horizontal lemniscate and vertical ellipse-horizontal lemniscate).

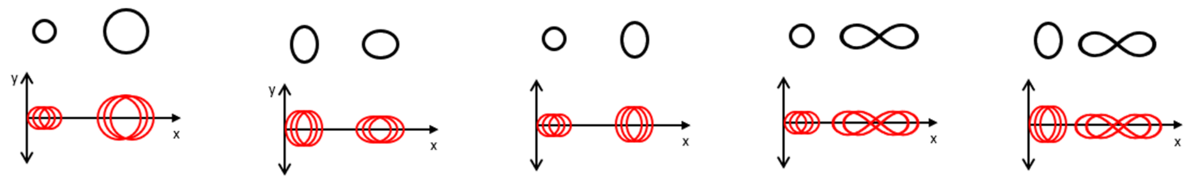


Figure 28. Chosen wobble patterns pairs.

The laser configuration used in development phase in Exom facilities and in pre-industrial phase in Lortek facilities were different. This variation led to a different thermal distribution on the welding zone, and this is critical for the information acquire by the thermal camera. For future works, we will study the variability on the thermal images created by different laser welding parameters configuration as laser power or feed rate, in order to develop a pre-processing algorithm prior to beam identification with the objective of strengthen the identification in different scenarios.

The objective of the test carried out was the automatic detection and recognition of the generated wobble and its sudden changes during a welding process. The generated data of the NIR and VIS cameras were compared with each other, concluding that the higher resolution of the VIS camera (1440x1080) provides more information of the welding quality (if we are really welding, possible defect identification, etc…) than the NIR camera (128x128), while the high-speed thermal camera detects faster the process events and is more suitable for wobble patterns identification with quick response. For this reason, we see a possibility to merge the information of both monitoring concepts in the future.

Exploitation

Impact and sustainability

The expected short-term impact of the planned project is the validation of the proposed approach for laser welding processes of aluminum alloys, heavily focusing on machine vision and artificial intelligence algorithms and their implementation in embedded computing devices for quality assurance and process control applications. Based on the actual set-up, an embedded heterogeneous multiprocessing system that combines a Xilinx Ultrascale+ ZU3EG MPSoC (FPGA+ARM) with an Nvidia System (GPU+ARM) for process control and monitoring, is planned to use the VEDLIoT framework to better exploit the edge-computing capacities of the laser solution. The objective of EXOM Engineering in the project is the further development of laser technology (both the method and the system/equipment) which allows a higher digitalization of the process, supporting the engineers using the solution during the definition of process parameters windows as well as identifying quality issues in manufacturing lines.

The exploitation of the project results will turn around the commercialization of products (algorithms, laser control card and scanning optics) which shall support the robustness and adoption of dynamic beam shaping technologies for aluminium welding in different industries, shortening the development and manufacturing cycle, reducing the consumption of time and material resources, and facilitating the use of aluminium laser welding for a wider range of applications also in other industries like aeronautics or energy production.

The following KPIs have been considered to measure the progress and success of the proposed project:

- Sensor fusion combining visual and thermal measurement in an embedded processing unit mounted on the laser welding head.
    - **During this project we process the information from both cameras in same hardware but in a separate way. Nevertheless, this idea of sensor fusion is considered a promising way to solve the problem of variability in this type of welding processes.**
- A real-time (under 5 ms) measurement, beam shape data processing and temperature distribution monitoring during the welding process. Beam shape recognition DL algorithm with real-time capabilities; ~30 fps.
    - **We can acquire from thermal camera at 1000 fps from Nvidia Jetson Xavier AGX module, and the classification model (approach 3 explained in Implementation and Characterization DL models section) inference time is ~12 ms without any optimization except TensorRT. We think that with an appropriate model pruning and quantization we can achieve this real-time processing objective. This work will be done on next steps of the collaboration between EXOM and LORTEK.**
- Increment of productivity during set-up for new materials (aluminium alloy) welding by at least 30%.
    - **This solution is not tested in a production environment so we cannot measure this KPI.**
- Demonstration joint manufactured with the final laser head solution.
- Beam shape recognition DL algorithm with minimum accuracy requirements for classification, detection or segmentation solving tasks:
    - 90 % accuracy for beam shape classification**. In the case of thermal camera, we use a classification based DL model to identify the beam shape. In the development phase the accuracy achieves this value.**
    - 0.75 mean Average Precision (mAP) for beam shape detection within the image and subsequent classification. **In this case, object detection was tested for visible images, but we decide to focus on thermal image and classification task, so we cannot measure this KPI.**
    - 0.75 Intersection Over Union (IOU) for beam shape segmentation. **Out of the scope of this project.**
- Decrease of DL workflow elapsed time –from dataset generation to model deployment– by at least 40 %.
    - **The development of deep-learning-framework by Lortek allows us to automatize the training process, leading to a reduction of 15-20% of time in this task. Due to time constraints the deployment of models is not included in this framework and will be done in future projects.**

The overall objective of the project goes in the direction of developing and validating algorithms based on AI for real time quality assurance and process control. Currently, EXOM Engineering has developed its own platform for supervising laser welding processes. The Hibrid Processing System, works as a hardware platform combining FPGA and GUP graphic card, and offers the possibility of combine information from different sensor (NIR cameras, visible cameras, distance sensors, among others) with the laser and a machine, employing a single processing unit. Therefore, the system can react in real time against any possible defect, improving the process performance and quality of the components.

Figure 29. HPS (Hybrid Processing System) AI platform for laser process control.

Summary and next steps

During the last 13 months the work done in BEAM_IDL has being focused on the definition and validations of a fast algorithms processing arquitechture for AI (Artificial Intelligence). The automatic detection of rapids events in laser processes is a major industrial challenge and represent an important industrial advantage in many welding applications. The arquitechture tested in BEAM_IDL has demonstrated the ability of detecting wobble types in real time, discerning small profile variations, as well as loosing melt-pool stability. Those among others, represent basic knowledge of a laser welding detection and control solution. Additionally, more work is already ongoing in this direction to validate novel capacities of the hardware/software solution.

Among others:

- Fillet detections and machine trajectory correction (hot path planning). Implementing a projected laser line or a line scanner.
- Quality assurance through defects recognition (this is a very wide and complex task, that has many variants depending on material or the type of union for example.

References

[1] R. GEHM, "SAE International," 19 02 2021. [Online]. Available: https://www.sae.org/news/2021/02/constellium-aluminum-ev-battery-enclosures. [Accessed 08 12 2022].

[2] "ams DIGITAL," [Online]. Available: https://automotivemanufacturingsolutions.h5mag.com/ams_autumn/aluminium_welding_goes_digital. [Accessed 08 12 2022].

[3] E. SHEKEL, "Dynamic Beam Lasers Reshape Materials Processing Applications," *Photonics Spectra,* pp. 40-45, 02 2022.

[4] S. Lee, T. Kim, S. Hu y W. Cai, «Joining technologies for automotive lithium-ion battery manufacturing—A review,» de *Proceedings of the ASME 2010 International Manufacturing Science and Engineering Conference (MSEC2010)*, Erie, PA, USA, 12–15 October 2010.

[5] I. Bunaziv, O. M. Akselsen, X. Ren, B. Nyhus, M. Eriksson and S. Gulbrandsen-Dahl, "A Review on Laser-Assisted Joining of Aluminium Alloys to Other Metals," *Metals,* no. 1680 [https://www.mdpi.com/2075-4701/11/11/1680], 2021.

[6] T. Zacharia, S. A. David, J. M. Vitek y T. Debroy, «Modeling of interfacial phenomena in welding,» *Metallurgical Transactions B,* vol. 21, nº [https://doi.org/10.1007/BF02667874], pp. 600-603, 1990.

[7] K. Behler, J. Berkmanns, A. Ehrhardt and W Frohn, “Laser beam welding of low weight materials and structures,” *Materials & Design, Volume 18, Issues 4–6,* Vols. ISSN 0261-3069, no. [https://doi.org/10.1016/S0261-3069(97)00085-X.], pp. 261-267, 1997.

[8] A. Otto, R. G. Vázquez, U. Hartel y S. Mosbah, «Numerical analysis of process dynamics in laser welding of Al and Cu,» *Procedia CIRP, Volume 74,* Vols. %1 de %2ISSN 2212-8271, nº [https://doi.org/10.1016/j.procir.2018.08.040], pp. 691-695, 2018.

[9] I. Mys y M. Schmidt, «Laser micro welding of copper and aluminum,» *Laser-based micropackaging,* Vols. %1 de %2SPIE, Vol. 6107, nº [https://doi.org/10.1117/12.648376], 2006.

[10] I. O. f. Standardization, “ISO. 13919-1—Welding—Electron and Laser-Beam Welded Joints—Guidance on Quality Levels for Imperfections—Part 1: Steel,” Geneva, Switzerland, 1996.

[11] I. O. f. Standardization, «SO. 13919-2—Welding—Electron and Laser-Beam Welded Joints—Guidance on Quality Levels for Imperfections—Part 2: Aluminium and Its Weldable Alloys,» Geneva, Switzerland, 2001.

[12] P. S. Wei, «Thermal Science of Weld Bead Defects: A Review,» *J. Heat Transfer,* nº [https://doi.org/10.1115/1.4002445], p. 22, Mar, 2011.

[13] A. Ostendorf, W. Specker, M. Stallmach and J. Zeadan, “3D-MID and process monitoring for micro joining applications,” *Proceedings of the society of photo-optical instrumentation engineers (SPIE),* vol. 4977, pp. 508-517, 2003.

[14] M. Ono, K. Nakada y S. Kosuge, «An investigation on CO2 laser-induced plasma,» *J Jpn Weld Soc,* vol. 10, pp. 239-245, 1992.

[15] J. Shao and Y. Yan, “Review of techniques for on-line monitoring and inspection of laser welding,” *Journal of Physics: Conference Series,* vol. 15, no. [https://iopscience.iop.org/article/10.1088/1742-6596/15/1/017], p. 101, September 2005.

[16] W. Rohringer, T. Heine, R. Sommerhuber, N. Lehmann y B. Fischer, «Optical Microphone as Laser-Ultrasound Detector,» de *DAGA* , München, 2018 .

[17] W. Jamrozik, M. Fidali, A. Bzymek and A. Timofiejczuk, “Application of fused visual and thermal images in monitoring and evaluation of welding processes,” *Welding International* , Vols. Volume 29 - Issue 6, no. [https://doi.org/10.1080/09507116.2014.937591], pp. 445-453, 2015.

[18] L. Wang, M. Mohammadpour, B. Yang, X. Gao, J.-P. Lavoie, K. Kleine, F. Kong y R. Kovacevic, «Monitoring of keyhole entrance and molten pool with quality analysis during adjustable ring mode laser welding,» *Applied Optics,* Vols. %1 de %2Volume 59 - Issue 6, pp. 1576-1584, 2020.

[19] K. Anderson, J. Weritz and J. G. Kaufman, Aluminum Science and Technology, ASM International, 2018.

[20] "Laser Systems - Process Heads," IPG Photonics, [Online]. Available: https://lasersystems.ipgphotonics.com/Technology/Process-Heads. [Accessed 08 12 2022].

[21] B. SAMSON, "Photonics Media," Photonics Spectra, [Online]. Available: https://www.photonics.com/Articles/Fiber_Lasers_Continuing_to_Power_Growth/a66584. [Accessed 08 12 2022].

[22] M. &. K. R. Harooni, «Laser Welding of Magnesium Alloys: Issues and Remedies,» *In (Ed.), Magnesium Alloys. IntechOpen,* 2017.

[23] S. Villumsen, Process time optimization of robotic remote laser cutting by utilizing customized beam patterns and redundancy space task sequencing, Aalborg: Aalborg Universitetsforlag, 2016.

[24] Y. Yang, R. Yang, L. Pan, J. Ma, Y. Zhu, T. Diao y L. Zhang, «A lightweight deep learning algorithm for inspection of laser welding defects on safety vent of power battery,» *Computers in Industry,* vol. 123, nº 103306, 2020.

[25] P. Franciosa, T. Sun, D. Ceglarek, S. Gerbino and A. Lanzotti, "Multi-wave light technology enabling closed-loop in-process quality control for automotive battery assembly with remote laser welding," in *Procedings of Multimodal Sensing: Technologies and Applications,*, Munich, Germany, 2019.

[26] A. Bozic, M. Kos y M. Jezersek, «Power Control during Remote Laser Welding Using a Convolutional Neural Network,» *Sensors ,* vol. 20, 2020.

[27] B. Arejita, J. F. Isaza Paz and A. Zuloaga, "Monitoring of Laser Welding and Cladding Processes with Edge Artificial Intelligence Combining Thermal and Visual Cameras," in *Lasers in Manufacturing Conference*, Munich, Germany, 2021.

[28] Association for Advancing Automation, «GigE Vision Standard,» [En línea]. Available: https://www.automate.org/a3-content/vision-standards-gige-vision.

[29] NVIDIA, "NVIDIA Jetson Xavier AGX," [Online]. Available: https://www.nvidia.com/de-de/autonomous-machines/embedded-systems/jetson-agx-xavier/. [Accessed 14 02 2022].

[30] Axiomtek , «https://www.axiomtek.com/Default.aspx?MenuId=Products&FunctionId=ProductView&ItemId=26095&C=AIE900-902-FL&upcat=350,» [En línea].

[31] VEDLIoT project, «VEDLIoT Deliverable 4.1 - First report on cognitive IoT hardware platform and microserver development,» 2022.

[32] J. Brooks, «COCO Annotator,» 2019. [En línea]. Available: https://github.com/jsbroks/coco-annotator/.

[33] Antmicro, «Kenning framework,» [En línea]. Available: https://github.com/antmicro/kenning.

[34] E. Pacaud y F. Olivier Pacaud, «Aravis,» [En línea]. Available: https://github.com/AravisProject/aravis.

[35] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu y X. Zheng, *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems,* 2015.

[36] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li y L. Fei-Fei, «ImageNet: A large-scale hierarchical image database,» de *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[37] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár y C. L. Zitnick, «Microsoft COCO: Common Objects in Context,» de *Computer Vision – ECCV 2014*, Springer International Publishing, 2014, p. 740–755.

[38] VEDLIoT, Deliverable 3.1, Evaluation of existing architectures and compilers for DL.

[39] A. Sherstinsky, «Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network,» 2018.

[40] B. Lindemann, T. Müller, H. Vietz, N. Jazdi y M. Weyrich, «A survey on long short-term memory networks for time series prediction,» *Procedia CIRP,* vol. 99, p. 650–655, 2021.

[41] C. B. Vennerød, A. Kjærran y E. S. Bugge, *Long Short-term Memory RNN,* arXiv, 2021.

[42] K. Cho, B. van Merrienboer, D. Bahdanau y Y. Bengio, *On the Properties of Neural Machine Translation: Encoder-Decoder Approaches,* arXiv, 2014.

[43] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser y I. Polosukhin, *Attention Is All You Need,* arXiv, 2017.

[44] D. Tran, L. Bourdev, R. Fergus, L. Torresani y M. Paluri, *Learning Spatiotemporal Features with 3D Convolutional Networks,* arXiv, 2014.

[45] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong y W.-c. Woo, *Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting,* arXiv, 2015.

[46] K. Simonyan y A. Zisserman, *Two-Stream Convolutional Networks for Action Recognition in Videos,* arXiv, 2014.

[47] S. Yan, Y. Xiong y D. Lin, *Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition,* arXiv, 2018.

[48] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun y M. Paluri, *A Closer Look at Spatiotemporal Convolutions for Action Recognition,* arXiv, 2017.

[49] Ultralytics, «YOLOv8 Ultralytics GitHub,» [En línea]. Available: https://github.com/ultralytics/ultralytics.

ICT-56-2020 - Next Generation Internet of Things

Final technical report **Artificial Intelligence - driven RIding Distributed Eye (AI-RIDE)** VEDLIoT Open

Document information	
Project website	https://www.vedliot.eu
Dissemination Level	PU
Nature	R
Contractual Deadline	30.06.2023
Author	Francesco Paolucci (CNIT), Marco Abbondandolo (AUG)
Contributors	

The VEDLIoT Project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957197.

Changelog		
Version	**Date**	**Type**
V 1.0	30-06-2023	Finalisation

Table of contents

Executive summary

The AI-RIDE project proposes the adoption of an accelerated, online and embedded Artificial Intelligence framework in the context of motorcycle rider training, particularly targeting the Practical Driving Courses (PDC) and Driving Licence Exam (DLE) sessions verification tools. The project targets a disruptive innovation step in the context of driving learning techniques, significantly going beyond the state of the art of the current instruments used in the PDC and DLE ecosystem.

The VEDLIoT hardware platform and middleware are the key driver to enable the AI-RIDE platform. Thanks to the disaggregated IoT and sensors-based network architecture, the distributed algorithm approach, high speed networking and embedded online AI at the edge, the AI-RIDE application is capable of performing a number of innovative functions supporting the whole driving learning ecosystem (i.e., practitioners, licence candidates, instructors, commissioners). The VEDLIoT are foreseen as the best candidate platforms guaranteeing adequate online computing capabilities, along with the low latency requirements needed for online assistance. For the scope of AI-RIDE, video and image processing is required to perform data fusion from different sources (cameras and wearables) and implement optimal vision targets augmentation during a drive test session.

1 Introduction

The AI-RIDE project is one of the VEDLIoT open call proposals that has passed the open call selection. The development of the AI-RIDE application employs the VEDLIoT project platforms technologies. Firstly, the VEDLIoT hardware platforms are used for system and testbed development, including the final demonstration. A sensor network, equipped with Near Edge and Far Edge platforms, processes incoming data from cameras, driver's sensors and headsets for both training phases and real-time applications. The feedback information towards the driver is computed, or alternatively, the exam score is calculated. Distributed AI is elaborated through the VEDLIoT toolchain, and the TensorFlow Frameworks are used, alongside suitable compilers and Runtime APIs to distribute the effort to various hardware platforms. A specific architecture to isolate specific AI-based processing functions and provide the full processing chain is required for the AI-RIDE solution. The final solution requires object recognition closer to the cameras, object tracking, possible re-identification of objects at different cameras, score computations using parallel or cascade AI or analytical tools.

The evolving state-of-the-art IoT sensors for driving/riding, driven by the automotive ecosystem and autonomous driving applications, generate vast amounts of data for trajectory and behavior predictions. However, these instruments are unsuitable for providing instructive feedback during human learning and training phases. Interacting with humans in driving school scenarios requires a different approach.

The project focuses on Practical Driving Courses (PDC) and Driving License Exams (DLE) for motorbikes. Factors such as performance time, trajectory precision, speed management, driver's posture, and motorbike position influence the exam outcome. Video cameras along the circuit track can provide data for computing these factors. However, cognitive features and specific human-interpretable feedback play a crucial role in improving performance.

Motorbike riding learning phases face a critical issue of practitioners struggling to track their eyes towards the correct street targets, hindering their acquisition of correct riding practices. This difficulty is challenging to address and leads to mistakes during practice and exam sessions, causing delays in obtaining the driving license. An automated mechanism that helps drivers target their eyes correctly would enhance their awareness and speed up the learning phase. Additionally, such a framework would provide instructors with interactive teaching tools and improve learning sessions. It would also facilitate automatic verification and auditability during license exams, offering objective performance evaluation parameters for AI-computed exam scores. This innovative toolkit would benefit the entire drive/ride license ecosystem. Currently, there are no available solutions in the market or scientific literature for automating motorbike training and testing procedures. Singapore has announced plans for AI-based automation expected to launch in 2024, but no details are available.

The AI-RIDE solution utilizes data fusion and AI processing to improve motorbike riding performance. It combines information from cameras on the track circuit and sensors/camera worn by the rider. The cameras provide details on body posture, vehicle position, speed/trajectory, while the innovative sensor/camera worn by the rider tracks line of vision and targets. The driver wears a helmet equipped with cameras and augmented reality sensors. An inner camera tracks the direction of the driver's eye pupil, similar to existing commercial solutions. The AI-RIDE solution augments this information by determining the optimal position of the pupil for optimal driving performance and safety. This position is computed in real-time by an AI system trained using performance trials from license instructors.

Additionally, the AI-RIDE system records key performance parameters and provides online feedback and test scores. It offers objective indicators to exam committees for evaluating test performance. The solution combines these pieces of information in different use cases:

1) **Use Case 1**: Training the AI model: Instructors and non-instructors wear AI-RIDE helmets to record pupil positions and route performance, classifying optimal, sub-optimal, and critical routes, to obtain reference Datasets.
2) **Use Case 2 (Practical Driving Course – PDC)** Practitioners wear AI-RIDE glasses with dynamic visualization (red+green spots) to evaluate and accelerate optimal driving learning, along with additional information such as optimal route trajectory in real time.
3) **Use Case 3 (Driving Licence Exam – DLE)** Exam candidates wear standard helmets without local augmented reality. Automatic performance evaluation, scores, and outcomes are provided remotely during exams thanks to a distributed camera-triggered video processing system, achieving 100% objective auditability without the need for physical attendance by exam commissioners.

In summary, AI-RIDE leverages data fusion, AI processing, and augmented reality to enhance motorbike riding performance, facilitate training, and automate exam evaluation. Moreover, it addresses the need for tailored feedback in driving training by adapting IoT sensors and data analytics tools to enhance human learning and improve performance in motorbike riding scenarios. The projects includes testbed and track setup, system design and implementation, evaluation result and a demonstration event at the end of the project.

To realize the AI-RIDE idea, design and implementation, including the final demo, a continuous feedback interaction was carried on with the VEDLIoT team. In particular, monthly-based discussion and reporting calls have been organized along the whole project duration between the AI-RIDE team and the VEDLIoT project representatives, including the coordinator (Jens Hagemeyer), one of the VEDLIoT partners focused on mobility and automotive-based solutions (Veoneer partner, Stefan Andersson and Anne Brandt) and the VEDLIoT technical team in support of hardware configuration (Jens Hagemeyer, Kevin Mika and René Griessl).

2 Idea and architecture

The AI-RIDE project intends to revolutionize motorcycle rider training by adopting an online and embedded Artificial Intelligence framework. The project focuses on enhancing the current instruments used in the PDC and DLE ecosystem and utilizes the VEDLIoT hardware platform and middleware to enable the AI-RIDE platform. The platform uses IoT and sensors-based network architectures, a distributed algorithm approach, and embedded online AI at the edge to perform innovative functions supporting the entire driving learning ecosystem. The VEDLIoT Far Edge Computing and/or Near Edge Computing platforms are deployed at motorcycle track facilities to ensure adequate online computing capabilities and low latency requirements for online assistance. The required VEDLIoT hardware includes CPU and GPU platforms for video and image processing in implementing augmented reality to optimize vision targets during a drive test session. The project relies on the VEDLIoT hardware through the open Call loan service support for specific use cases and application requirements. The hardware platforms have been installed and evaluated locally for low latency performance during driving test sessions.

Project innovations

The main innovation steps targeted by AI-RIDE include:

- Improving the state-of-the-art of IoT sensors for driving/riding, which has been significantly evolving in recent years due to the automotive ecosystem and next generation autonomous driving applications.
- Providing instructive feedback to anomalous, not eco-friendly, dangerous or inaccurate maneuvers during the human learning and training driving phases for motorbikes, which current instruments are not adequate for.
- Adapting the driving school scenario to interact with humans in a completely different approach.
- Developing AI capabilities that are specific to the Practical Driving Courses required to achieve Driving Licences for motorbikes, including the Licence Exam.
- Computation of performance factors such as trajectory precision, speed management, the driver's body posture and the motorbike position, using data analytics tools relying on video frames from external cameras.
- Assisting the practitioners, the instructors and the public officers to improving cognitive features by providing specific human-interpretable feedbacks in case of driving limited performance, to speed up the learning phase and to offer measurable, auditable and certified outcome results.

Gantt of the project

Project Month	1	2	3	4	5	6	7	8	9	10	11	12
Work packages and tasks												
WP1: Management and Dissemination												
T1.1 Admin and financial coordination	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
T1.2 Data, quality and risk management	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
T1.3 Dissem., Comm., and Exploitation	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
WP2: Technical Requirements, Architecture and Use Cases												
T2.1 Network, AI and Sensor Requirements, Use Case detailed definitions	🟦	🟦 **M1**	🟦									
T2.2 Definition of the AI-RIDE & VEDLIoT architecture	🟦	🟦	🟦	🟦	🟦 **M3**							
T2.3 Testbed setup	🟦	🟦 **M1**	🟦	🟦	🟦 **M3**							
WP3: AI-RIDE Implementation												
T3.1 Dataset and training acquisition			🟦	🟦 **M2**	🟦	🟦						
T3.2 Sensors, helmet and camera software			🟦	🟦	🟦	🟦	🟦	🟦	🟦			
T3.3 Design and implementation of the AI models at the edge					🟦	🟦	🟦	🟦	🟦	🟦		
T3.4 App design and implementation				🟦	🟦	🟦	🟦	🟦	🟦 **M4**	🟦		
WP4: Testing and Demo												
T4.1 Test sessions and feedbacks to the implementation						🟦	🟦	🟦	🟦	🟦	🟦	
T4.2 Demonstration										🟦	🟦	**D1**

Figure 1 Gantt Chart of AI-RIDE

The Gantt Chart of the project is shown in Figure 1. The project duration is 12 months and includes 4 Work Packages.

Work Package 1 is in charge of Management and financial coordination, Data management, Dissemination, Communication, and Exploitation.

Work Package 2 objectives are the following: the definition of the AI-RIDE use case applications, the definition of the main requirements of the software application, the testbed network, the AI engines and modules, the required VEDLIoT hardware flavours, the end-user sensors, the definition and prioritization mitigation strategies and means for setting realistic performance expectations, the definition of the AI-RIDE solution requirements and the framework architecture.

Work Package 3 objectives are the design, the development and prototyping of the whole AI-RIDE solution. WP3 is in charge of creating of a large dataset used for reference driving test sessions and the AI training, perform the design and prototype of software solutions at the driver's sensors for retrieving data and for enabling AR/VR real time information (e.g., eye pupil track), the design, prototype and implementation of AI models for exam scoring based on the analysis of external and driver's sensors, the design and implementation of the AI-RIDE whole application encompassing the different usage options: a) test sessions with augmented feedback, b) exam sessions with auditable scores and outcome. This WP performs initial prototyping of the algorithms, which are adjusted to the specific feedback provided by test session activities.

Work Package 4 objective is the evaluation of the AI-RIDE platform based on the selected Track Testbed and a test environment allowing integration and validation, as well as evaluation of performance such as AI, telemetry scalability in different conditions. Testing plans identify low-level system KPIs for performance evaluation. Test cases are designed and implemented for this purpose. Moreover, the WP oversees preparing and organizing the final demo, open to the VEDLIoT consortium and other institutional/industrial/research entities interested in the project outcomes.

Application Requirements

For all the AI-RIDE considered use cases, the AI-RIDE applications running in the VEDLIoT hardware/software systems will require:

1) Near real time feedback mechanism through analysis of the distributed sensor data along the Track (cameras, glasses)
2) Total feedback actuator trigger latency < 80 ms
3) Correct detection of key objects and key behaviors in the Track perimeter
4) Overall Licence Exam outcome time delivery < 2 minutes

Networking Requirements

In terms of network requirements, the need to perform video analysis requires the adoption of a Gigabit Ethernet – based infrastructure with limited wireless connectivity. To minimize the latency and, overall, the jitter of the connections between sensors and central edge node, it is important to provide wired connectivity between cameras placed along the track (UC1 and UC3). This choice will limit the latency/jitter burden to sensors and AI processing, while the network contribution will be constant and limited. Limited (and optional) wireless connectivity will be required in the UC2 case in the case the AI processing is performed onboard.

Use Case Definitions

UC1 Reference Datasets

Reference datasets need to be recorded and made available to the driver operators. The dataset will need to include key performance parameters used during the test exam evaluations. In particular, the motorbike position and route during the trial, the position of the body, the eye pupil track, the trial elapsed time will be considered as performance parameter set, classifying the trial by expert instructors (optimal performance classification), licensed riders (sub-optimal performance) and practitioners (non-optimal performance).

UC2 Practical Driving Courses (PDC)

The use case adopts the use of sensors onboard to provide direct feedbacks to the practitioner. The use case definition is depicted in Figure 2. The considered key performance metric of the use case is the actual position of the eye pupil of the driver during the test. The live feedback to the driver is a set of messages relying on the eye directions. The data sources are the glass video camera (pointing to the track in first person view) and the glass side cameras used to perform pupil eye tracking. Data are sent to the AI-RIDE software system running continuous score-based algorithm designed on the track features. The system, running in the VEDLIoT platforms, deploys a workflow that computes an instant test score evaluating the distance metric between the actual pupil eye target and the optimal one (provided in the video datasets by instructors). Moreover, the system near real time output sends continuous or event-based feedback alerts to driver's actuators, dynamically suggesting the most proper vision target. The system, optionally, sends offline data to the central edge node for further post-processing analysis, for dataset savings, allowing the practitioner to have a complete test recording repository, including scoring, eye tracking results along the track test, received alerts and to analyze the behavior of the driver after the alerts together with the instructor for improving the driving style.

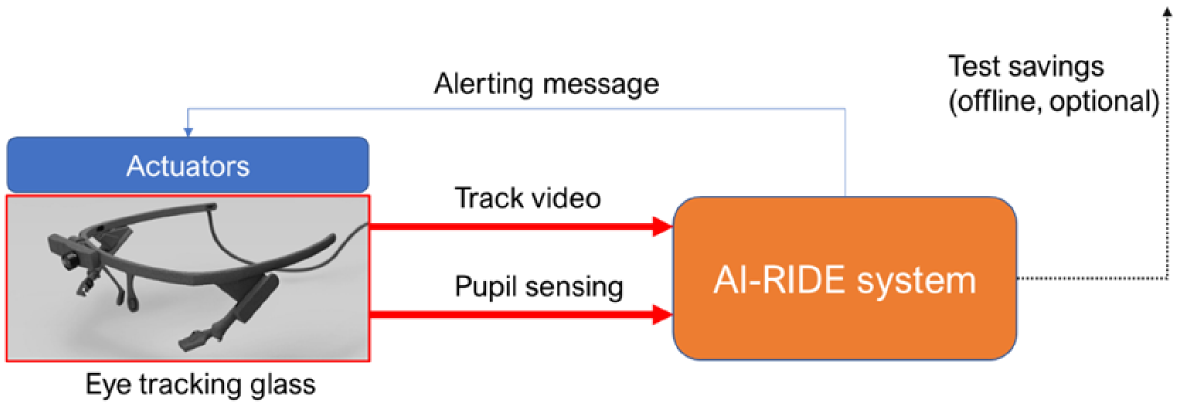


Figure 2 PDC Use Case definition

The system, optionally, sends offline data to the central edge node for further post-processing analysis, for dataset savings, allowing the practitioner to have a complete test recording repository, including scoring, eye tracking results along the track test, received alerts and to analyze the behavior of the driver after the alerts together with the instructor for improving the driving style.

UC3 Driving Licence Exam (DLE)

This use case adopts distributed sensors (i.e., IP cameras) along two different tracks. The sensors collect videos used by the AI-RIDE system to perform the automatic scoring of the Exam test. The motorbike licence exam (category A) is regulated in Italy by the Ministry of Infrastructure and Transports (MIT) Decree 2018, inherited by the new European Directive on the subject. The Exam includes the execution of two type of tests: low speed balance (LSB) test and high-speed agility test (HSA). The two tests are run along two different tracks, depicted in *Figure* 3. The LSB test is run on a 30 meters long track. The driver must be able to tackle a slalom at speed and then the passage in a narrow corridor. He will have to discard the first cone on the right and to follow the others, at the end of the slalom he must follow the most regular trajectory possible around the central cone and, therefore, enter the narrow corridor. The errors that may lead to exam rejection are the following: a) touch one or more cones; b) jump a cone during the slalom or exit the course; c) put one foot on the ground d) irregularly coordinate driving demonstrating poor skill; e) take less than 15 seconds to complete the route. Most of these errors will be considered in the DLE automatic scoring mechanism.

The HSA test is run on a 100 meters long track. The candidate driver is required to perform a slalom leaving the first cone on the right or left. At the end of the slalom there is a curve, a straight trajectory with suggested 50km/h peak speed and a final passage between 3 cones in the center of the track, going through the narrow corridor passing inside the cones placed at 1 meter distance from each other and, at the end, stop the vehicle so that the front wheel passes the first alignment, but not the next one. In this test the Decree specifies the irregularities that may lead to exam rejection: a) touch one or more cones; b) jump a cone during the slalom or exit the course; c) put one foot on the ground; d) irregularly coordinate driving, demonstrating poor skill; e) stop the motorcycle with the front wheel that has not passed the first alignment or has passed the second alignment; f) take more than 25 seconds to complete the route.

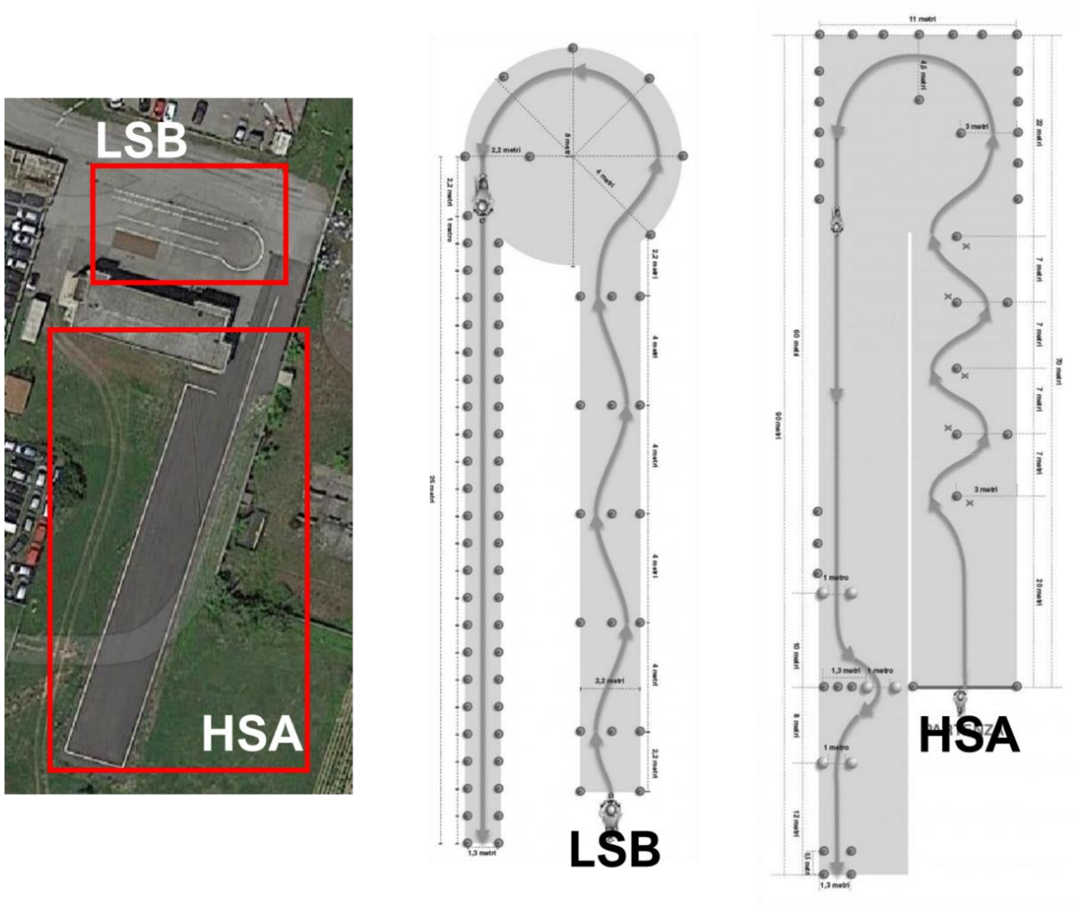


Figure 3 Tracks and DLE official tests

The use case definition is shown in Figure 4. The key performance metrics are a subset of the metrics defined by the two tests and include the time performance, the route adherence to the optimal trajectory, an evaluation of harmonic speed/acceleration patterns during the test, the detection of errors such as cone touches, foot on the ground, track area exit. The data source are videos collected by the different track cameras that will be installed by the project. Video streaming flows are received by the VEDLIoT edge node located at the Track premises. Videos are subject to analysis workflows, studied with deep details in WP3, including AI detection algorithms. The system outcome a scoring KPI detailing the different features of the test related to the different metrics. Thresholds will be set to obtain the exam acceptance/rejection outcome.

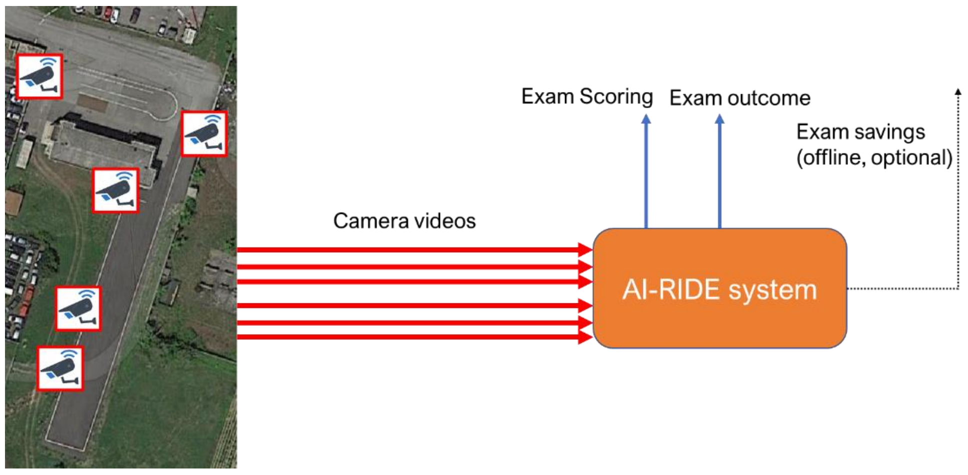


Figure 4 DLE use case definition.

3 Implementation

Brownfield Testbed Setup

A detailed recognition of the Track testbed Facility in Pontedera (AUG track) was held in two different steps in July 2022. The recognition was successful to identify the requirements, the constraints of the location. The identification of the required sensors and their location in the track to successfully cover all the use cases required to run a set of preliminary tests. To this end, the recognition included a preliminary test session deployed resorting to cameras of different type and flavours. The video cameras utilized for the recognition test are the following:

a) a low-cost Full HD webcam with 1920x1080 resolution at 30 fps,

b) a GoPro Hero 10 with up 2704x1520 resolution at 240fps and 3840x2160 resolution (4K) at 120fps.

We run a couple of acquisition days at the Testbed Track located in Pontedera during some students' training. As you can see in Figure 5 (left) the camera positions are marked by the label CX where X is the number of the camera. All the cameras were placed at the height of 6 meters and there is a total of 6 cameras: 4 for the short track in every direction and 2 for the long one only at the end and its beginning.

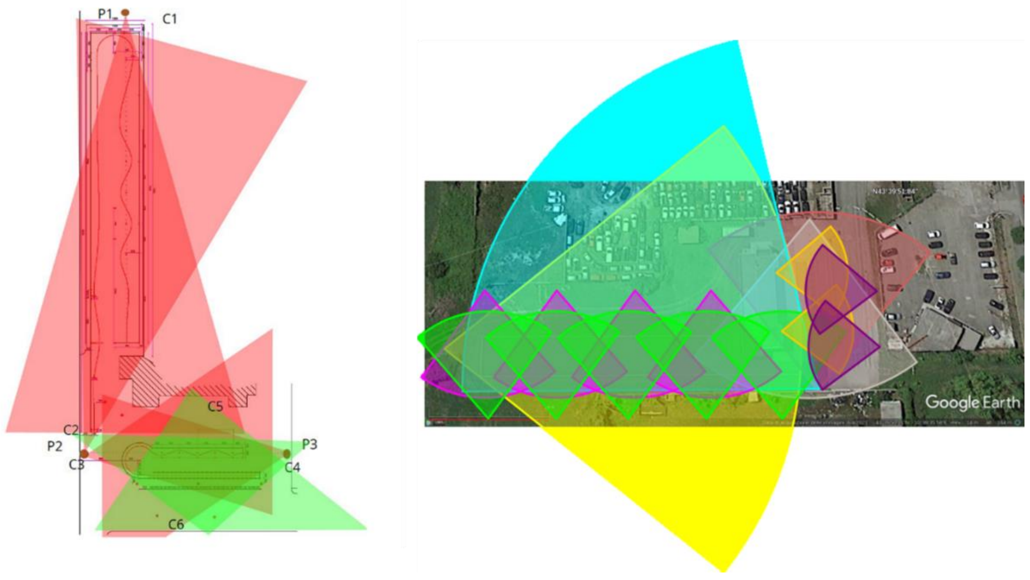


Figure 5 Camera placement in the recognition (left), optimal camera placement for 100% DLE (right).

From the preliminary workflow processing obtained on the videos, aiming at performing object detection (motorbike, person, cone) in the whole area of the tracks, and given the budget constraints, we have identified an optimal and a sub-optimal (final setup) possible camera placement. The optimal camera number and placement would include 2 wide-angle (6 meters height using dedicated poles) and 4 low-angle cameras for LSB track, and 2 wide-angle (6 meters height) and 9 low-angle cameras for HSA track. The need of the low-angle cameras would be required to perform the foot on the ground detection along the whole track. This type of recognition is particularly challenging and requires a detailed motorbike lateral detection.

Sensor/camera requirements, setup and placement

The pre-final setup has been selected after careful considerations and matching the AI-RIDE budget indications. The LSB setup is confirmed, while the HSA setup has two alternative setup placements that will require further recognition after the cameras purchase and will be assessed in M2.

A total of 8/9 cameras are identified as the sub-optimal solution allowing the foot on the ground feature detection covering 50% of the two tracks. The first sections of the tracks are the most critical ones due to the slalom trajectories. The AUG instructors confirmed that the foot on the ground event is much more likely to occur in this zone. The minimum requirements of the cameras are the following:

1) Outdoor IP camera with wired Ethernet connectivity (Fast/Gigabit Ethernet)
2) Resolution: 1920x1080 Full HD (2MPixel, 1080p)
3) Frame rate: 50fps

Given the minimum requirements, the candidate cameras are the following:

a) Dahua IPC-HDW5442T-ZE
b) Dahua IPC-HFW5541T-ASE
c) Hikvision DS-2CD7A26G0/P-IZHS

The placement of the cameras requires 2 new poles installation in the LSB track and 2/3 new poles in the HSA track. One pole will share two cameras, one for LSB and another one for HSA.

The picture in Figure 6 shows the final placement for LSB. Front and back cameras (i.e., pink and grey) are mounted on 6-meters high poles, centered on the axis of the aligned cones in the first track half. Two lateral cameras (i.e., yellow) are mounted on the AUG premises building and are used mainly to detect the foot on the ground event.

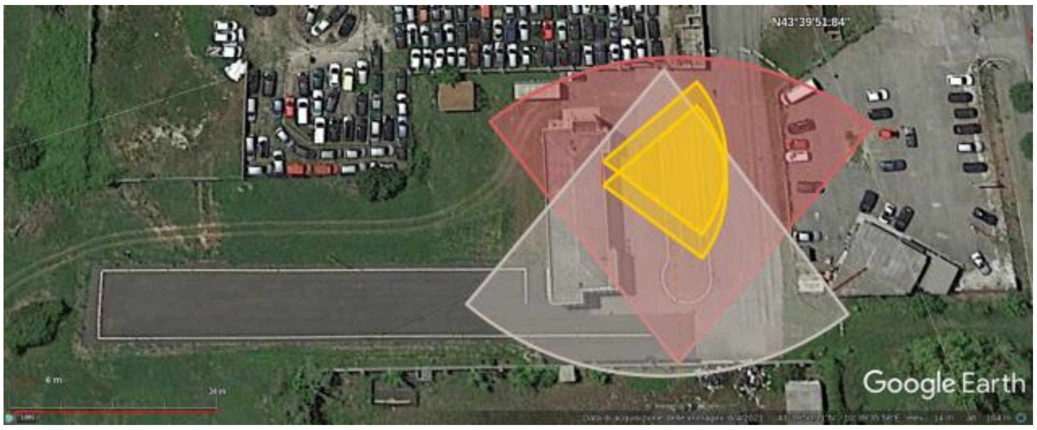


Figure 6 Final setup LSB

The picture in Figure 7 shows the Option 1 placement for HSA. Front and back (i.e., light blue and yellow) cameras are mounted on 6-meters high poles. The light blue camera is mounted on the same pole hosting the LSB pink camera. Lateral cameras (i.e., purple) are mounted on the first part of the circuit mainly for cone touch and foot on the ground detection.

The picture in Figure 8 shows the Option 2 placement. In this case the two lateral cameras are replaced with three cameras mounted on a L-shape pool over the track circuit (i.e., blue, orange and purple). Orange and purple cameras are low angle configured, while blue camera is placed vertically pointing to a specific cone slalom set. The three cameras can fully cover the first part of the circuit.

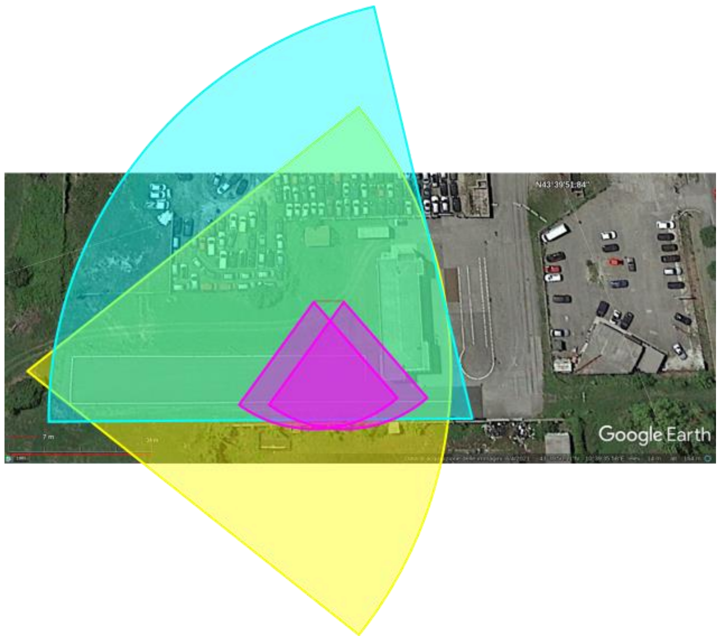


Figure 7 Setup HSA: Option 1 (lateral cameras)

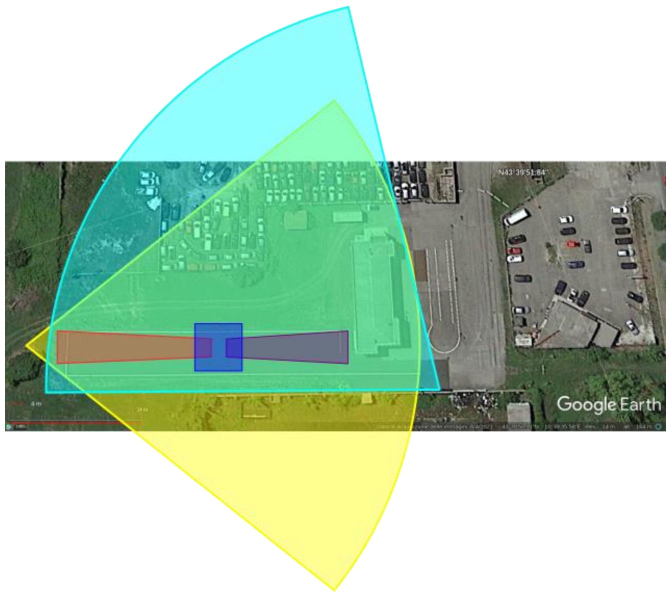


Figure 8 Setup HSA: Option 2 (main L-shape pole)

The networking connectivity requires an aggregation switch (at least 16 FE/GE ports, at most 4 10GE ports) to enable the full connectivity between the cameras and the VEDLIoT Edge Node hardware.

VEDLIoT hardware requirements

To assure near real time proper functioning of UC2 and UC3, the VEDLIoT platform needs to be located at the AUG Track premises. The target hardware includes, as a minimum set:

1) 1 u.RECS to be possibly boarded inside the motorbike for UC2 real time feedback

2) 1 t.RECS for UC3, able to process at least 4 HD 60fps video source streaming

The two hardware models need to be equipped with internal modules capable to perform efficient video analytics – oriented workflows including AI (object recognition, background subtraction, tracking). The candidate t.RECS internal modules include the Jetson Xavier AGX and Jetson Xavier NTX devices. The final candidate platforms have been discussed and selected after the first releases of the workflows with the installed cameras and sensors take place at the track premises to run the first dataset recordings, after M1 release.

Final testbed setup: PDC use case

In the PDC use case, sensors and processing nodes are fully onboarded on the motorcycle. To this end, a specific packaging system has been designed to host all the required sensors and enable the system to work properly inside the motorbike. The full system is referred to as AI-RIDE Vehicle Module (ARVM). The Pupil Labs glasses are connected by means of a USB-C cable to the ARVM, positioned inside the motorbike in a protected environment (e.g., motorcycle trunk). The ARVM is composed of 3D-printed plastic main box and a 3D-printed plastic cover designed to allow cables entering inside the module safely. The VM is packaged with a Li-IOn battery and is designed to host the local processing board. In the preliminary tests we have utilized an NVIDIA-based Single Board computer (i.e., Jetson Nano) and a x86 miniPC. In the tests carried out during the WP4 activities the ARVM will enclose the VEDLIoT u.RECS device, therefore the 3D-printed enclosure will be adapted and reprinted to fully support the u.RECS. In addition to this, an Inertial Measurement Unit with 6DOF is enclosed in the chassis, in order to provide information about the current vehicle inclination. Such data, combined with gaze and pupil track information, will provide improved awareness of the driver and vehicle status, supporting the module to output high quality feedback information. The feedback mechanism is realized during the WP4 tests. The preliminary idea is to use a LED strip triggering independent LED switch on/off configurations to indicate a correct eye track and position during the test.

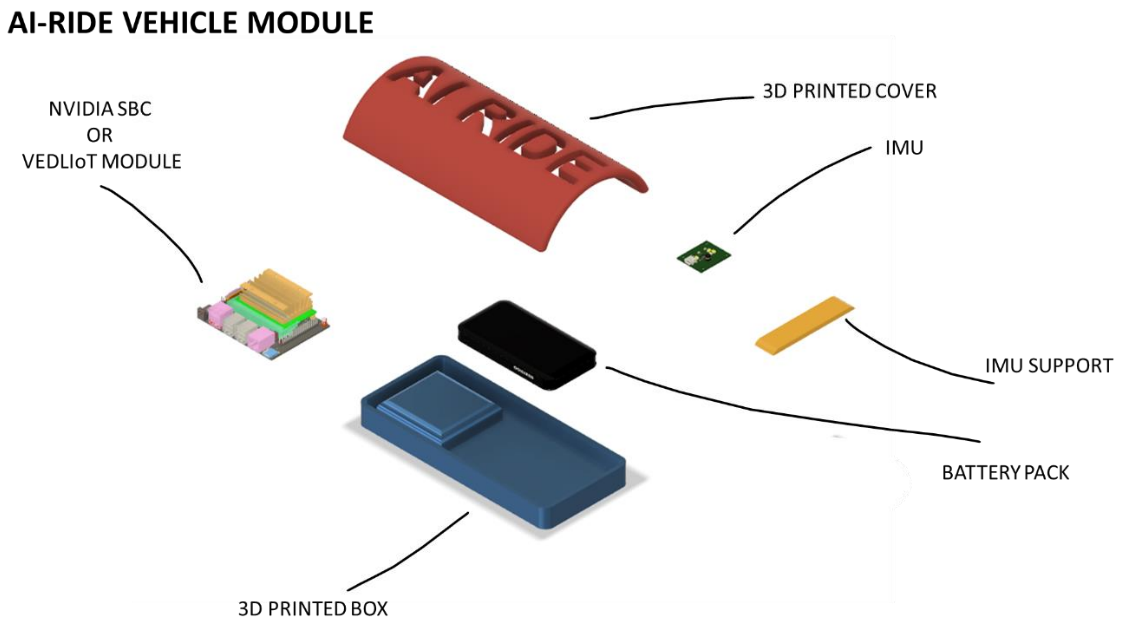
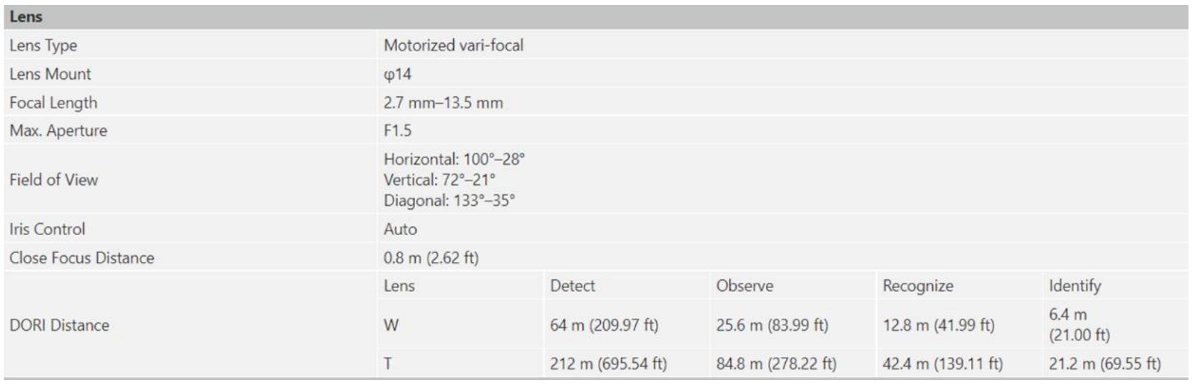


Figure 9 AI-RIDE Vehicle Module (ARVM)

Final testbed setup: DLE use case

In the DLE use case a number of cameras needs to be placed in the two tracks (LSB and HSA) in order to cover all the phases of the official exam session. In Milestone M1, we reported a study computing the optimal camera placement for each track with the assumption that the IP camera focal is fix (i.e., 3.6mm). The number of required cameras for optimal coverage was 19. A sub optimal solution was considered, employing just 8 cameras, satisfying the budget constraints.

Another solution came in place in December 2022. The use of motorized vari-focal IP cameras was considered since, at that date, they had become available in the market at a competitive price. The considered camera is the Dahua IPC-HF5541E-ZE, offering the same features satisfying the requirements identified in M1 (FullHD resolution at 50-60 fps, at least 100 Mbit/s with Ethernet interface), with the advantage of a configurable focal length ranging from 2.7mm up to 13.5mm. The camera technical specifications are summarized in Figure 10.



Lens					
Lens Type	Motorized vari-focal				
Lens Mount	φ14				
Focal Length	2.7 mm–13.5 mm				
Max. Aperture	F1.5				
Field of View	Horizontal: 100°–28° Vertical: 72°–21° Diagonal: 133°–35°				
Iris Control	Auto				
Close Focus Distance	0.8 m (2.62 ft)				
DORI Distance	Lens	Detect	Observe	Recognize	Identify
	W	64 m (209.97 ft)	25.6 m (83.99 ft)	12.8 m (41.99 ft)	6.4 m (21.00 ft)
	T	212 m (695.54 ft)	84.8 m (278.22 ft)	42.4 m (139.11 ft)	21.2 m (69.55 ft)

Figure 10 Dahua IPC-HFW5541E-ZE: lens technical specifications

Therefore, during the Pilot Test 4 carried out in December, a new placement evaluation has been carried out with the same budget constraints considered in M1. The outcome of the evaluation represents the final and definitive camera placement, driving the physical setup works carried out in January 2023.

The final camera placement with vari-focal IP cameras consists of 3 cameras covering the LSB track and 5 cameras covering the HSA track, for a total of overall 9 cameras. The placement is shown in Figure 11. In the LSB track, camera 1 is placed at the front side of the track, and cameras 2 and 3 are placed at the same lateral side, covering the start, the finish and the first half of the track, in both directions. In the HSA track, camera 4 is placed at the bottom of the track, covering the track start. Cameras 5 and 6 are placed at the lateral side at around 50% of the track length, each one pointing to different directions. Similarly, cameras 7 and 8 are placed at the lateral side at around 80% of the track length, able to cover the track edge. Finally, camera 9 is placed at the track end, thus covering the zone of the finish line.

With this new configuration it is possible to minimize the number of costly new poles to be installed. In fact, only 1 new 5 meters-height pole is needed to be mounted, hosting camera 1 and camera 9. Two existing light poles will be utilized to host cameras 5-6 and cameras 7-8. The remaining cameras are installed on the walls of the AUG track building (4 and 5 meters height). The details of the placement are reported in Table 1.

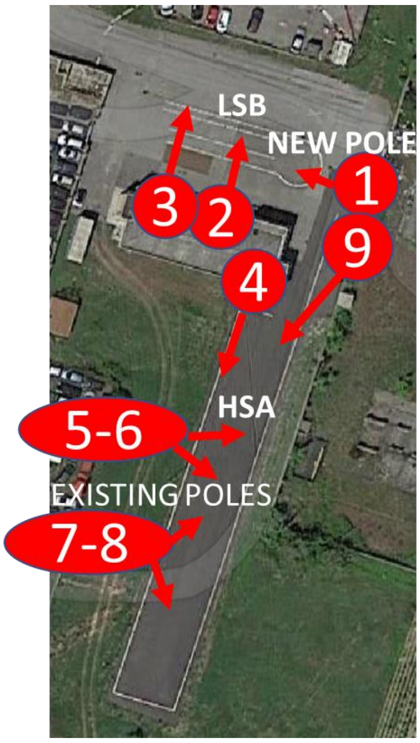


Figure 11 Final definitive placement of vari-focal IP cameras.

Table 1 Final Camera placement details

Camera #	**Track**	**Position**	**Camera type**	**Angle**	Support
1	LSB	Front side	Vari-focal	Aligned to first half axis	Pole
2	LSB	Lateral side	3.6mm focal	90°	Wall
3	LSB	Lateral side	Vari-focal	90°	wall
4	HSA	Bottom side (start)	Vari-focal	Aligned to first half axis	Wall
5	HSA	Lateral side	Vari-focal	≈ -45°	Pole
6	HSA	Lateral side	Vari-focal	≈ +45°	Pole
7	HSA	Lateral side	Vari-focal	≈ -45°	Pole
8	HSA	Lateral side	Vari-focal	≈ +45°	Pole
9	HSA	Finish line side	Vari-focal	≈ 60°	Pole

The technological upgrade works coordinated by AUG were carried out in January 2023. A new pole was founded to host cameras 1 and 9, as shown in Figure 12. All the cameras were mounted and connected to the central switch placed inside the track building. PoE connectivity was tested for all the cameras. Some of them are placed to a distance approaching 100m with respect to the switch, therefore some cameras were provided with independent power supply and standard Ethernet connectivity to avoid bad PoE performance or connectivity outages.

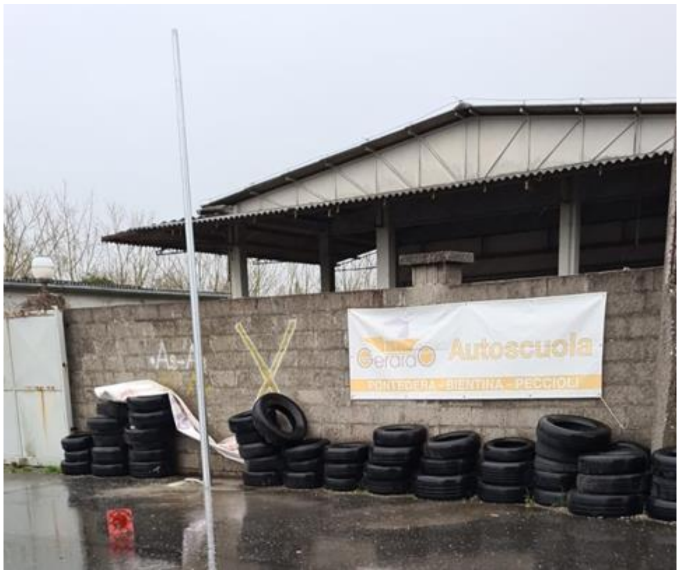
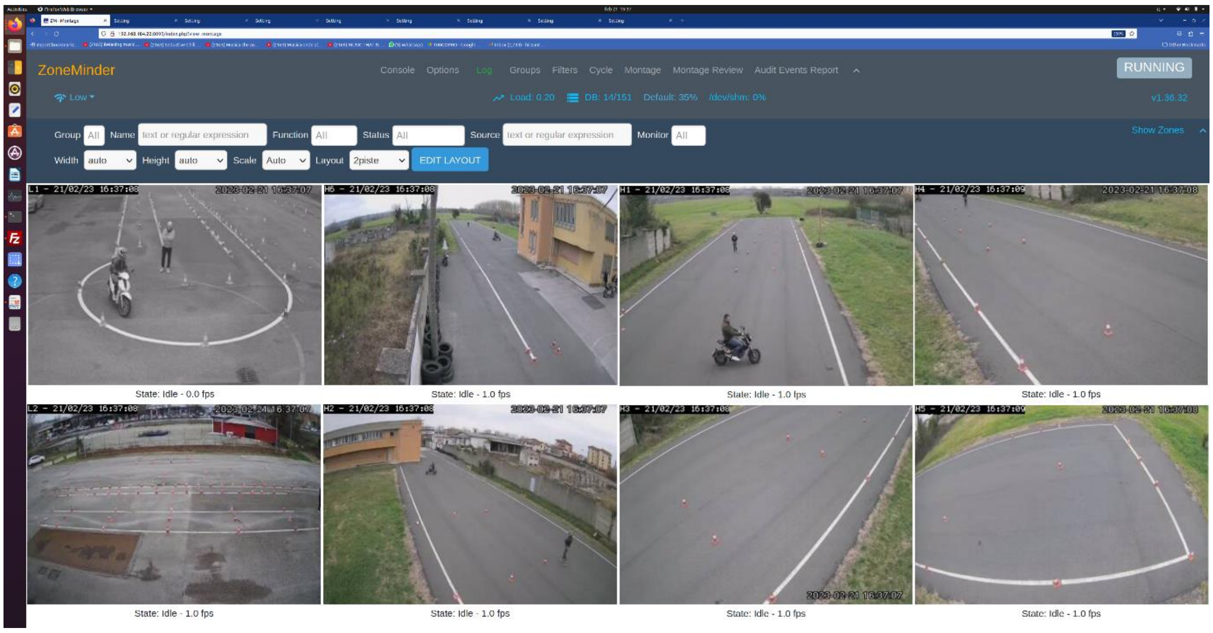


Figure 12 New pole founded to host camera 1 and 9.

After poles, cameras and connectivity installation, each camera have been connected to a central Juniper switch equipped with PoE-based Gigabit Ethernet interfaces. Each camera was updated with the latest firmware and configured with the main stream options set according to the requirements (60fps, full HD) and tuned from the point of view of ISO, aperture, contrast, brightness and shutter speed. Finally, continuous streaming configuration was set and a parallel monitoring system based on the secondary stream configuration (low resolution and bitrate) have been deployed relying on the ZoneMinder tool, to check the current status of each camera and monitor the tracks state in real time. The ZoneMinder dashboard with the active cameras is shown in *Figure* 13.


ZoneMinder
Console Options Log Groups Filters Cycle Montage Montage Review Audit Events Report
RUNNING
Low
Load: 0.20 DB: 14/151 Default: 35% /dev/shm: 0%
v1.36.32
Show Zones
EDIT LAYOUT

Group All Name text or regular expression Function All Status All Source text or regular expression Monitor All
Width auto Height auto Scale Auto Layout 2piste

L1 - 21/02/23 16:37:08 — State: Idle - 0.0 fps
H6 - 21/02/23 16:37:08 — State: Idle - 1.0 fps
H1 - 21/02/23 16:37:08 — State: Idle - 1.0 fps
H4 - 21/02/23 16:37:09 — State: Idle - 1.0 fps
L2 - 21/02/23 16:37:08 — State: Idle - 1.0 fps
H2 - 21/02/23 16:37:08 — State: Idle - 1.0 fps
H3 - 21/02/23 16:37:08 — State: Idle - 1.0 fps
H5 - 21/02/23 16:37:09 — State: Idle - 1.0 fps

Figure 13 ZoneMinder camera monitoring system.

Dataset acquisition and analysis for DLE

The preliminary DLE dataset acquisition was carried out during one of the Pilot Test sessions, held between October and November 2022 in the Pontedera Track. The aim of this first preliminary dataset is to start the training activities and all the workflow design and development activities based on a preliminary set of videos reproducing specific events in both the Low Speed Balance (LSB) and High Speed Agility (HSA) tracks.

The camera utilized for the videos is one Dahua IPC-HFW5541T-AS-PV camera. The camera follows the requirements identified in M1 for a correct video acquisition quality: 1) Main stream FullHD 1080p (1920x1080 resolution) guaranteeing at least 50 frame per second, 2) IP camera with at least 100 Mbit/s Ethernet or Power over Ethernet connectivity.

Before the dataset acquisition, the camera has been tested offline with different video acquisition software to check the correct values of resolution and exposition. During the dataset collection, the camera was placed at specific positions identified by the placement assignments discussed in M1. A mobile pole was used to attach the camera at around 5 meters height. The setup of the pole was the following: the camera is connected via Power over Ethernet to a Juniper EX3200 Switch equipped with 8 Gigabit PoE ports. A Linux laptop is connected to the switch via Gigabit Ethernet interface, running the *ffmpeg* and *gstreamer* video streaming acquisition software tools. The full setup of the mobile pole is shown in the photo of Figure 14 (a). The pole was moved during the whole dataset acquisition in the different positions where they will be mounted using dedicated fix poles. The photos of Figure 14 (b) and (c) show two different mobile pole locations at the LSB and HAS tracks, respectively.

The selected camera placement is a subset of the placement assignment as reported in M1. Specifically, video recordings related to the LSB track are taken from two different locations, place 1 (front side) and 2 (lateral side). HSA locations are place 3 (bottom side, end of the track), place 4 (bottom side, start of the track), place 5 (lateral side). The selected places of the different tracks are shown in Figure 15.

The preliminary Dataset video collection is made of 40 videos, each one reproducing a single exam test, performed by the same driver (i.e., an expert AUG driver instructor) using two different motorcycles (a motorbike and a scooter), in both tracks. The videos are classified based on different and particular events needed to be reproduced during the driving test. Each event has an influence on the metrics used to compute the final fully automated DLE score. Each video is classified based on a single tag, i.e., only one tagged event is executed in a single video. The classification tags used for this preliminary dataset are reported in Table 2, showing the number of videos recorded with the corresponding tag/event. For specific events, additional videos have been considered. For example, for the touched cone tag, the event was recorded in different videos affecting different cones in the track. Moreover, the irregular driving test were repeated using different motorcycles, since the trajectory is dependent on the size and the weight of the motorcycle.

Table 2 Preliminary dataset video classification

Tag	**# LSB video**	**#HSA video**
Perfect instructor exam	2	4
Touched/moved cone event	3	5

Skipped cone event	2	3
Out-of-track event	2	3
Irregular driving (motorbike, scooter)	2	3
Foot on the ground event	2	3
Poor time performance	2	3

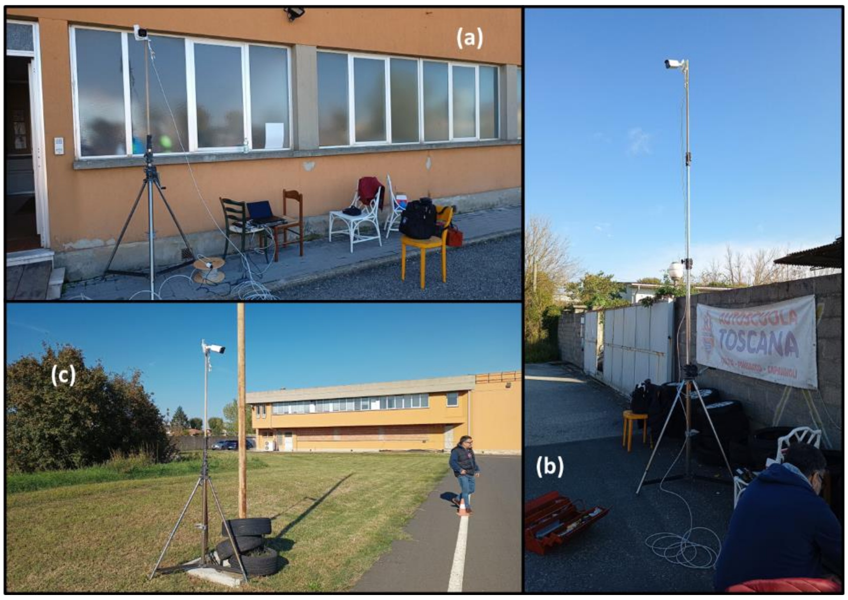


Figure 14 Preliminary Dataset acquisition: camera setup on the different tracks: pole setup (a), LSB front side placement (b), HSA lateral side placement (c).

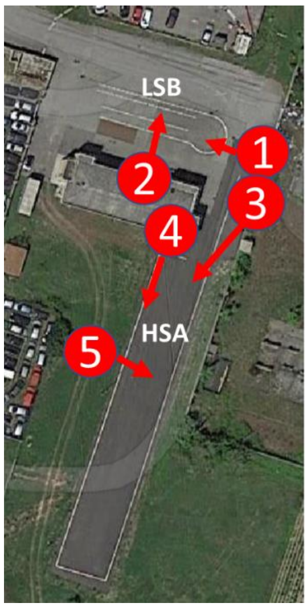


Figure 15 Locations of the IP camera used in the Preliminary Dataset acquisition.

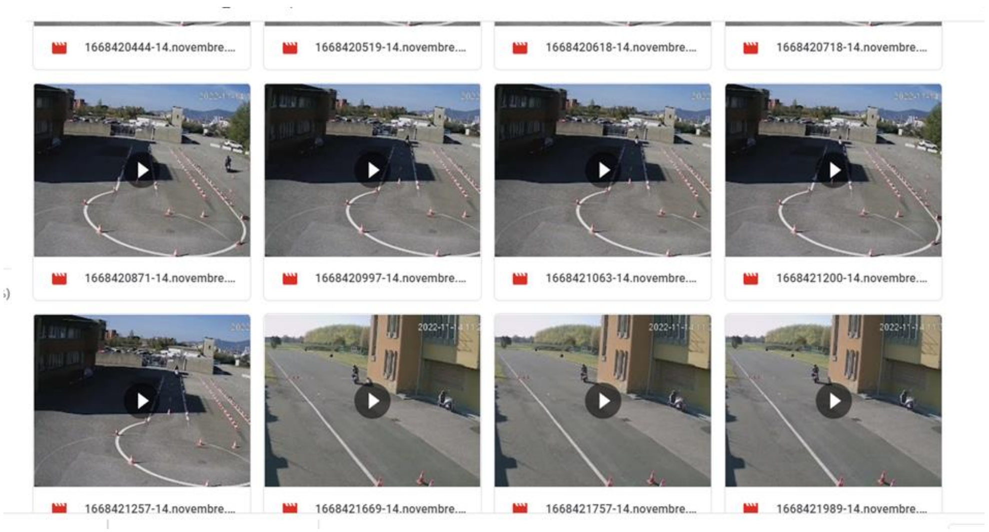

1668420444-14.novembre....
1668420519-14.novembre....
1668420618-14.novembre....
1668420718-14.novembre....
1668420871-14.novembre....
1668420997-14.novembre....
1668421063-14.novembre....
1668421200-14.novembre....
1668421257-14.novembre....
1668421669-14.novembre....
1668421757-14.novembre....
1668421989-14.novembre....

Figure 16 Excerpt of the dataset.

We have used the preliminary dataset to perform the first post processing analysis, that will be at the basis of the WP3 implementation of the DLE workflow inside the VEDLIoT edge node.

The picture in Figure 17 shows the processing elaboration of a single video, acquired from place 4 at the HSA track. The upper-left quadrant shows the full resolution original video. The upper-right quadrant shows only the Region of Interest (ROI) of the video, defined using any convex polygon (in this case 4 vertices but without the limitations of a simple rectangle). The ROI allows to reduce the effective processing area and eliminate possible noise. The red rectangle is the result of single step pass of background subtraction (BS) process which identifies a very noisy movement area (practically useless). In the bottom-left quadrant, the full BS filter is applied, that eliminates the noise and identifies only the largest floating objects (in this case, the target of our analysis, the person+motorbike object). The bottom-right quadrant shows the final result: the red rectangle indicates the area of the image where the object of interest moves. The long and marked shadow may be an issue that will require a dedicated post processing.

The picture in Figure 18 shows a similar processing procedure to a video acquired from place 5, offering partial HSA track lateral vision. With respect to the previous picture, the upper-right quadrant is the only one that changes compared to the other video. The ROI of the video consists of 5 vertices, allowing an improved adaptation to the track possibly exclude extraneous out-of-track objects (e.g., person at the track border). Moreover, an object detector is applied to perform motorbike recognition.

In these early experiments the applied Object Detector is SSDNet version 2, while BS and ROI use python-based scripts employing the OpenCV libraries. The typical video recording duration covering a single-track test is around 30s. The videos are processed using a Linux laptop employing Intel i7-10th gen CPU. With real-time constraint and without GPU, the output is performed at 6.7 fps. Without real-time constraint, recording at 50fps outputs results after 300s (5 minutes). Such measurements are encouraging, since the DLE are expected to use the significantly higher CPU+GPU resources provided by the VEDLIoT t.RECS.

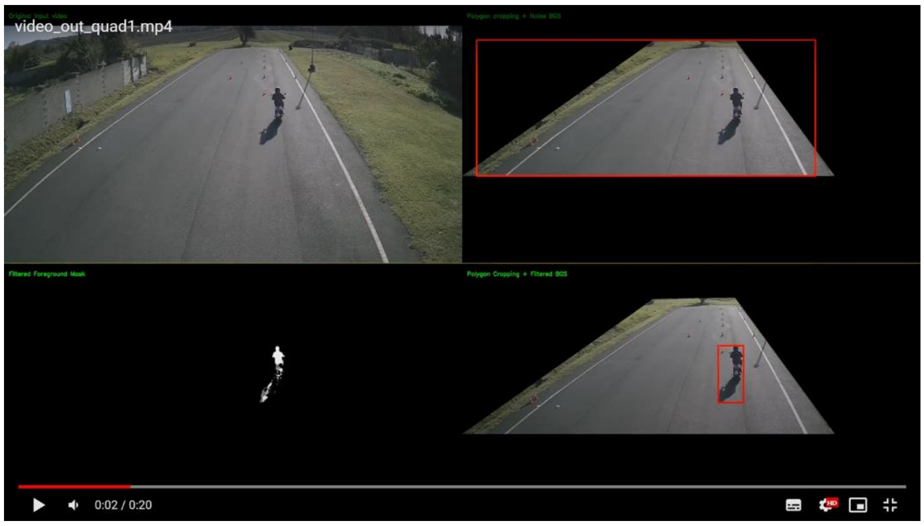


Figure 17 Background Subtraction, ROI.

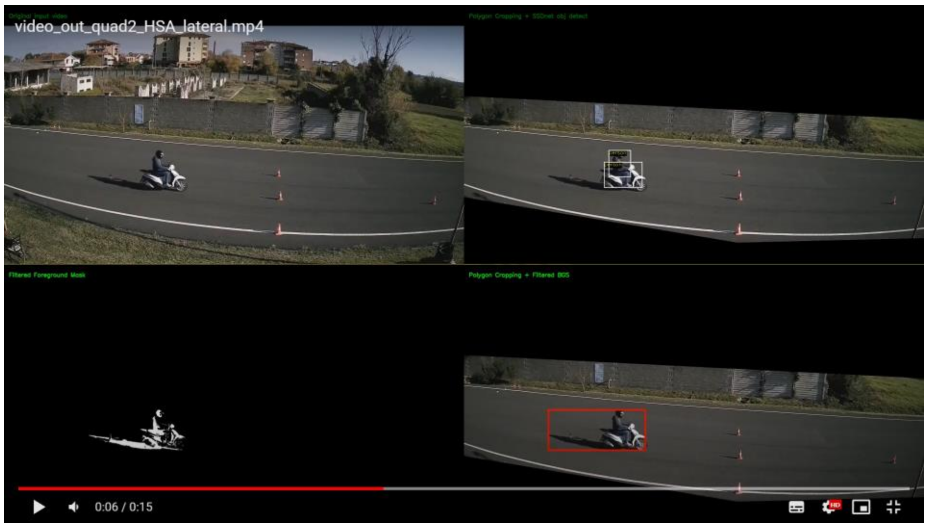


Figure 18 Background Subtraction, ROI and Object Detection.

Based on the workflow that will be elaborated and implemented by WP3 in T3.2, we have proposed the general structure of the target Final Dataset.

The dataset includes key performance parameters used during the test exam evaluations. In particular, the provided video recordings of the tests of an exam trial are associated to two group of features. The first group is referred to as pre-processing features (pre-PF) and includes metrics that are collected independently of the DLE performance evaluation of the test. Such metrics may have significant information impact for the training of future complex AI models. The pre-PF features include the driver profile (height, weight, age), the weather conditions, the type of motorcycle, the driver role (i.e., practitioner, exam candidate, instructor). The second group is referred to as post-processing features (post-PF) and includes all the metrics that will be elaborated as partial output in the DLE workflow implementation carried out in WP3. The post-PF metrics are strictly referred to the partial evaluation of key performance indicators that contributes to the final DLE exam outcome. For example, the touched/moved cone event feature will output a KPI feature, expressed in floating point unit, measuring the performance level of the test referred to the number and the gravity of those events (e.g., how many cones have been moved or touched, which ones are moved, correlated with a different scoring profile). The scoring profiles are implemented in WP3 based on the auditing of the exam instructors, since some behaviors and mistakes are weighted differently, based on the experience reported also by public examiners. The classification will include the final DLE scoring and outcome obtained by the AI-RIDE system and the actual exam outcome (Pass /Fail). Additional features may be added in the case the WP3 development identifies some key metrics of interest that were not included initially in the consolidated set of features.

In accordance with the VEDLioT team the final dataset has not been disclosed as open source tool for public usage. This comes from the significant interest in a future AI-RIDE product that AUG wants to explore in the near future. Feedbacks received from members of the Infrastructure and Transport Ministry and from the Civil Motorization suggest that there is a concrete possibility to evolve the AI-RIDE results into a complete software product certified for Exams that may be released for driving schools and that may receive the certification by the Ministry for conducting future official licence upgrade exams using this software as public outcome attestation. Therefore, commercial reasons led AUG to keep the dataset closed until a product version becomes available. However, for the sake of the VEDLIoT dissemination, AUG agrees that the dataset may be provided (only for scientific reasons) to specific partners requesting its usage through the consensus of AI-RIDE and VEDLIoT consortia.

Table 3 Dataset Structure proposal

Videos	**Pre-processing feature**	**Feature Unit**	**Post-processing feature (DLE)**	**Feature Unit**	**Classification**	Unit
MP4 file Per Track/ per Camera	Driver Age	Int	Touched/moved cone	TBD	DLE Exam Outcome	Float
	Driver Height	Int	Skipped cone	TBD	DLE Exam Pass/Fail	Bool
	Driver Weight	Int	Out-of-track	TBD	Real Exam Pass/Fail	Bool
	Weather Conditions	Enum	Foot-on-ground	TBD		
	Motorbike cylinder capacity	Int	Timing	TBD		
	Motorbike type	Enum	Irregular driving	TBD		
	Driver Type	enum	Additional DLE features	TBD		

DLE Workflow implementation

The implementation workflow follows the scheme of *Figure* 19. Cameras record video streams at 50fps with Full HD resolution.

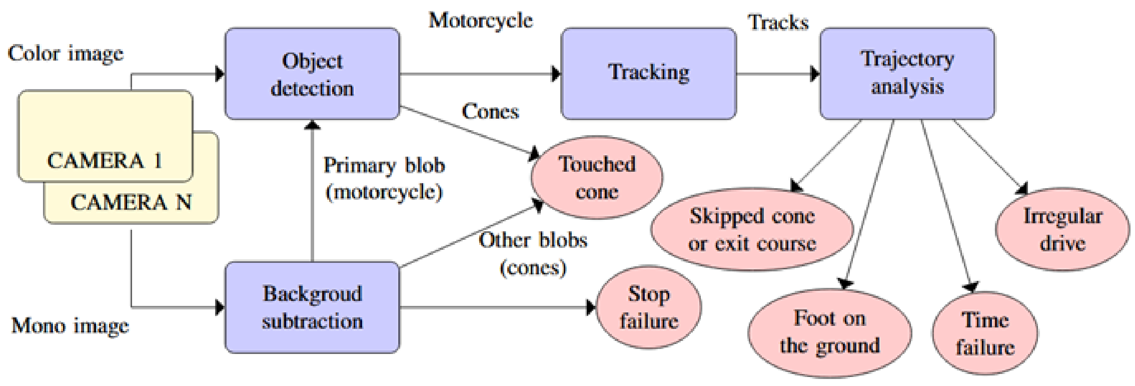


Figure 19 DLE Workflow implementation modules.

Object detection module is applied to all the frames to identify the motorbike, the pilot and the cones of the track. The detection module utilizes artificial intelligence libraries, in particular the YOLO version 5, trained using the frames of the video acquired along the whole dataset acquisition. For the detection of the motorbike only a cropped portions of the whole frame is used: the Foreground blob coming from the Background subtraction technique identifies the region of the frame that are changed with respect to the previous one; this improves the accuracy and the speed of the detection of the motorbike and enables the system to know where are the cones that can be occluded or touched.

Multiple training sessions have been employed to improve the detection, facing multiple issues, such as the rotation of the cone (see *Figure* 20), the invariance of the detection with respect to lightning and contrast image (the track is subject to different lightning conditions and shadows due to the near presence of buildings and the different sun positions during the day. The multiple training allowed to achieve excellent detection results for all the different track conditions, minimizing false positives and with accuracy > 95%. The example of *Figure* 21 shows accurate detection of all the objects in extremely different light conditions. In addition, differential cone detection has been performed to distinguish between untouched cones and cones on the ground, see *Figure* 21 with object detectors of different colors. This detection is crucial to detect exam mistakes (cones touched are critical, cones moved and fallen are more critical). Finally, to improve object detection and enable finer detections, object detection has been combined with semantic segmentation, to obtain not only the bounding box, but also the precise object profile at the pixel level. The segmentation is useful for tracking and to identify a motorbike reference point that allows the trajectory reconstruction. Semantic segmentation results are shown in *Figure* 22.

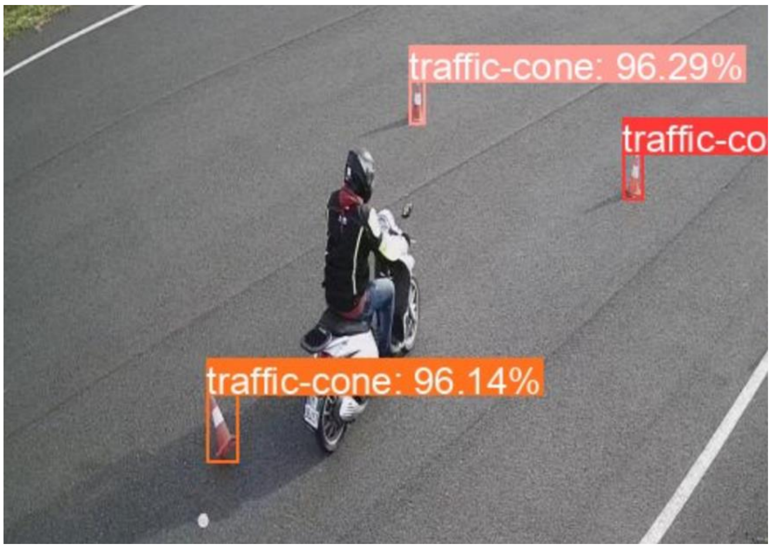


Figure 20 Object detection trials.

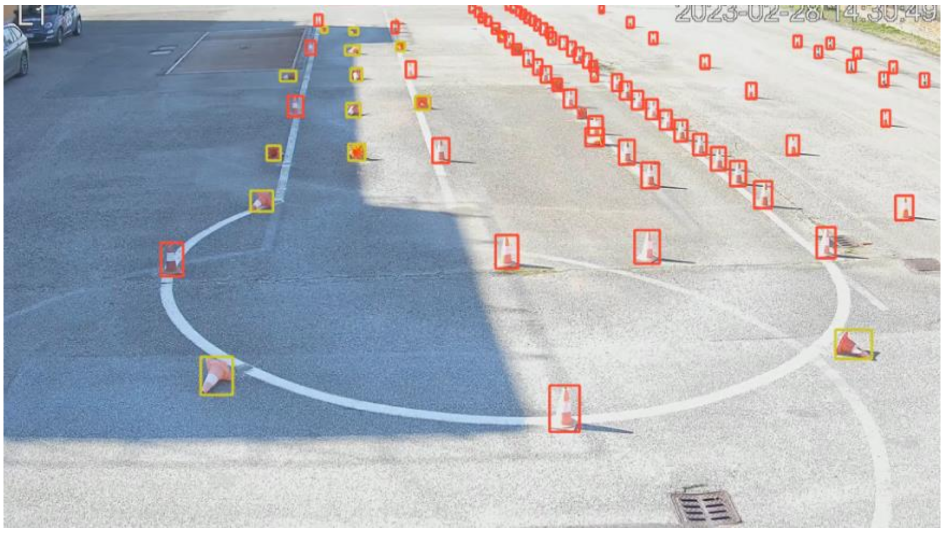


Figure 21 Differential object detection: standing cones (red), fallen cones (yellow).

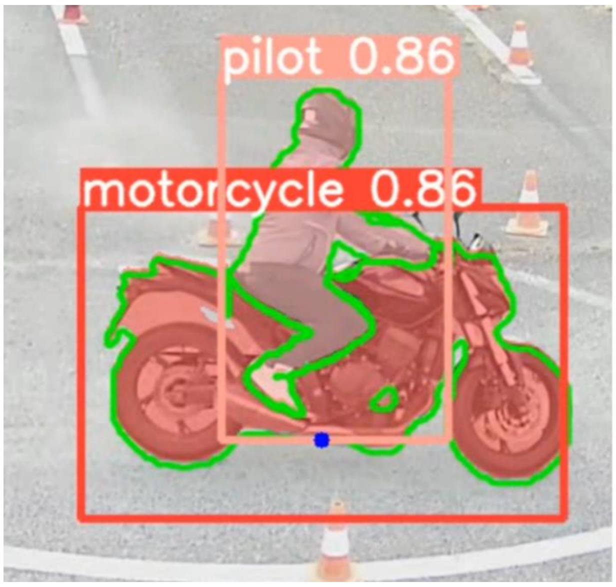


Figure 22 Semantic segmentation for motorbike and pilot. The blue point is the one used for position tracking. It is on the same vertical line of the center of mass, but very near to the ground reference plane.

Then, object tracking and trajectory analysis is carried out. For these modules, a prospective projection technique is applied. Each camera video is processed to obtain an orthogonal vision version (see *Figure* 23), thus each video contributes for a joint optimal orthogonal vision of the whole track during the test. In this way, it is possible to reconstruct with high precision the trajectory in both space and time domain, also relying on virtual zone sequencing (see *Figure* 24). With zone sequencing, it is easier to detect bad trajectories, skipped cones and out-of-track events. Moreover, a 3D fusion is processed to extract 3D visual using the 3D Unreal tool, as shown in *Figure* 25. To obtain a precise trajectory reconstruction, first the IP cameras have been calibrated in order to minimize the distortion effect of the cameras at the edges of the track. Then, the motorbike reference point coordinates are transformed into space coordinates using prospectic projection, filtered using a Kalman filter to eliminate noise and then interpolated using a set of bicubic splines functions. The result is an orthogonal continuous function describing the motorbike trajectory in the track. The exam trajectory is compared against an "optimal" trajectory in the space and time domain provided by a motorbike instructor. The computation result of the orthogonal trajectory is shown in *Figure* 26.

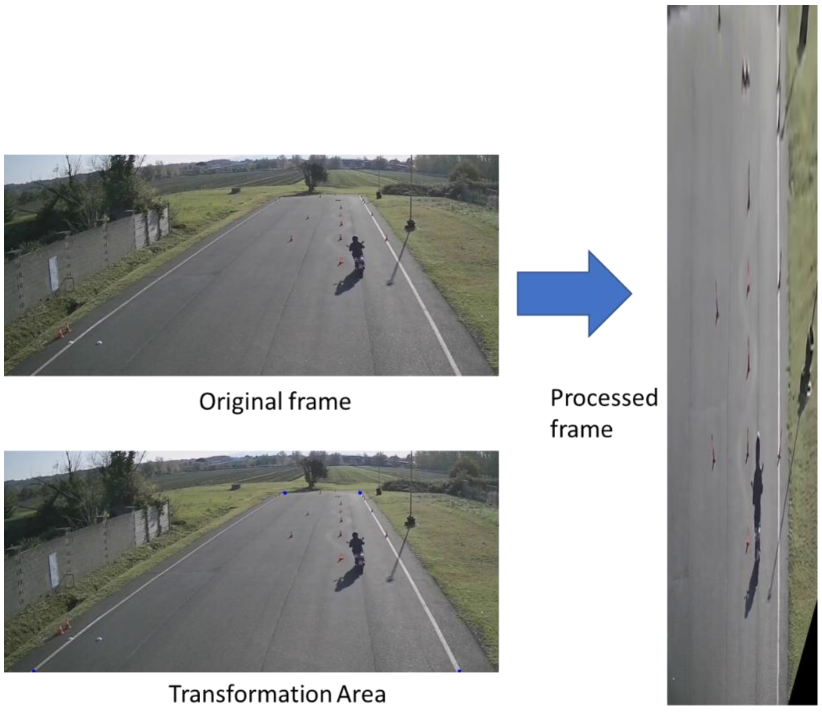


Figure 23 Orthogonal transformation of the track vision: prospective projection.

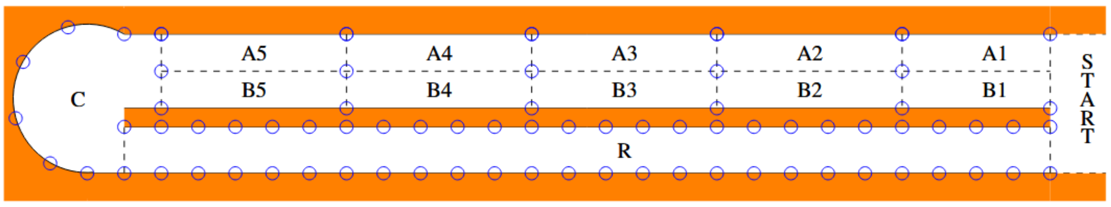


Figure 24 Track zone segmentation

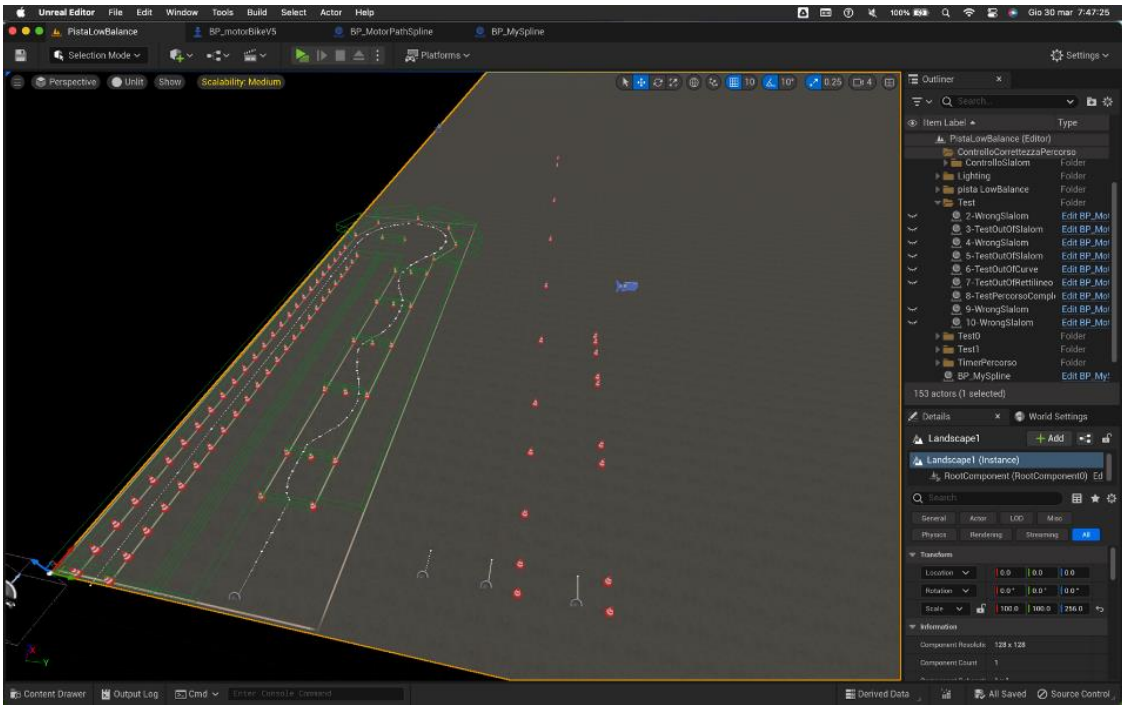


Figure 25 3D track trajectory processing.

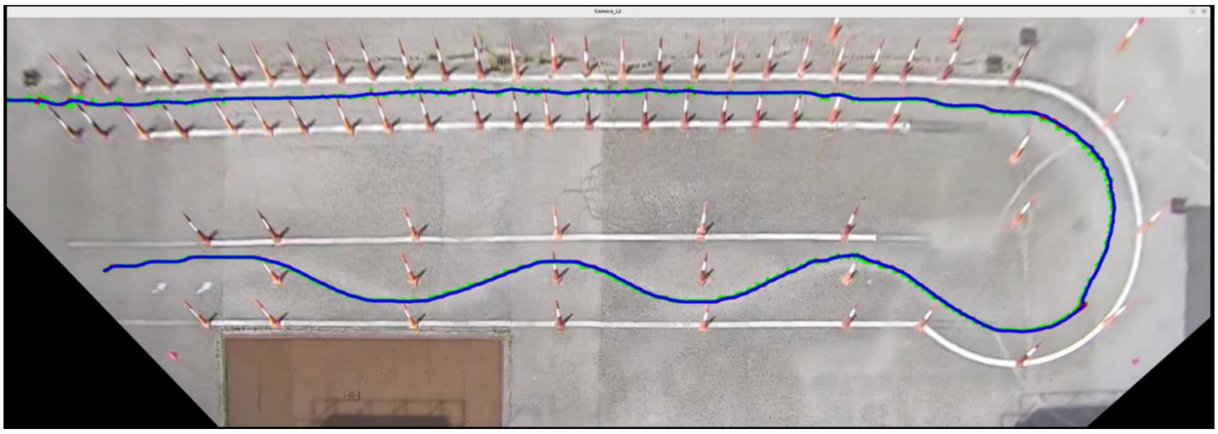
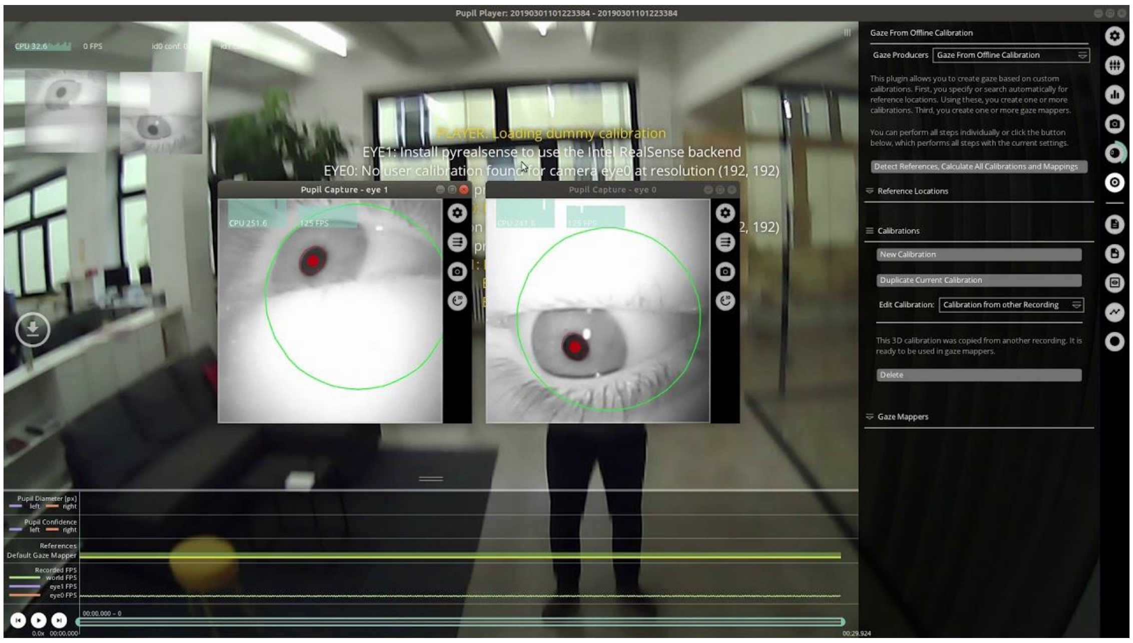


Figure 26 Final orthogonal trajectory elaboration and reconstruction

PDC Workflow implementation

The PDC use case employs the Pupil Glasses, enriched with feedback mechanisms, connected to the u.RECS co-located inside the motorbike to provide suggestions and offline analysis for driving improvements.

Figure 27 Pupil Labs Core glasses with eye pupil tracking system.


Pupil Player: 20190301101223384 - 20190301101223384

CPU 32.6 0 FPS

PLAYER: Loading dummy calibration
EYE1: Install pyrealsense to use the Intel RealSense backend
EYE0: No user calibration found for camera eye0 at resolution (192, 192)

Pupil Capture - eye 1
CPU 251.6 125 FPS

Pupil Capture - eye 0

Gaze From Offline Calibration

Gaze Producers: Gaze From Offline Calibration

This plugin allows you to create gaze based on custom calibrations. First, you specify or search automatically for reference locations. Using these, you create one or more calibrations. Third, you create one or more gaze mappers.

You can perform all steps individually or click the button below, which performs all steps with the current settings.

Detect References, Calculate All Calibrations and Mappings

Reference Locations

Calibrations

New Calibration

Duplicate Current Calibration

Edit Calibration: Calibration from other Recording

This 3D calibration was copied from another recording. It is ready to be used in gaze mappers.

Delete

Gaze Mappers

Figure 28 Pupil Labs Core calibration tools.

The packaging system designed for the AI-RIDE Vehicle Module (ARVM) is critical to ensure that all sensors required for the PDC use case are housed securely and appropriately inside the motorcycle. The ARVM is a complete system that facilitates the processing of data gathered by the various sensors. For the Pupil Labs glasses, the ARVM is responsible for connecting them to the motorcycle system. This connection is established through a USB-C cable, which ensures that the data from the glasses flows securely and reliably into the ARVM. The packaging and the full setup are shown in *Figure* 29.

Consisting of a 3D-printed plastic main box and a 3D-printed plastic cover, the ARVM has been specially designed to allow cables to enter the module safely. It also includes a Li-Ion battery and a local processing board, which in the preliminary tests were an NVIDIA-based Single Board computer (i.e., Jetson Nano) and a x86 miniPC. During the WP4 activities, the ARVM encloses the VEDLIoT u.RECS device, and therefore, the 3D-printed enclosure have been adapted and reprinted to fully support the u.RECS.

To provide improved awareness of the driver and vehicle status, an Inertial Measurement Unit with 6DOF is enclosed in the ARVM chassis. This component helps provide information about the current vehicle inclination, which, combined with gaze and pupil track information, allows the module to output high-quality feedback information during the tests carried out during the WP4 activities.

One of the most significant developments in the module is the introduction of the feedback mechanism. The preliminary idea is to use a LED strip to trigger independent LED switch on/off configurations to indicate a correct eye track and position during the test. This feedback mechanism is expected to provide a highly accurate and immediate response, allowing the driver to understand their own behavior on the road and make any necessary adjustments rapidly.

Overall, the ARVM is a significant contribution to the PDC use case, providing a robust and reliable platform for data gathering and processing. Its specially designed packaging system ensures that all the required sensors and processing systems are secure, and its feedback mechanism ensures that data is gathered accurately and efficiently, giving drivers greater insight into their driving behaviors.

The glasses for pupils detect the direction of gaze using two cameras directed towards the pupils. The detection algorithm can be disturbed by variations in environmental lighting, affecting the accuracy of pupil detection. During field testing, excessive sunlight was found to cause errors in gaze detection, which led to adjusting the exposure and filtering parameters of Pupil apps to improve accuracy. Tests have being conducted with various lighting conditions and parameter values to determine the optimal tuning for the glasses under all light conditions.

The onboard device collects two types of data: gaze data obtained from Pupil Labs glasses, which can be accessed through API calls, and IMU data that detects bike orientation and acceleration. To facilitate the analysis of the data, we have developed a compact Python application that seamlessly synchronizes the two sets of data, creating data files that connect the gaze data with the orientation and acceleration information gathered by the IMU. This Python application is versatile, as it can run on both Windows devices (currently our testing platform) and Vedliot ARM based Linux devices.

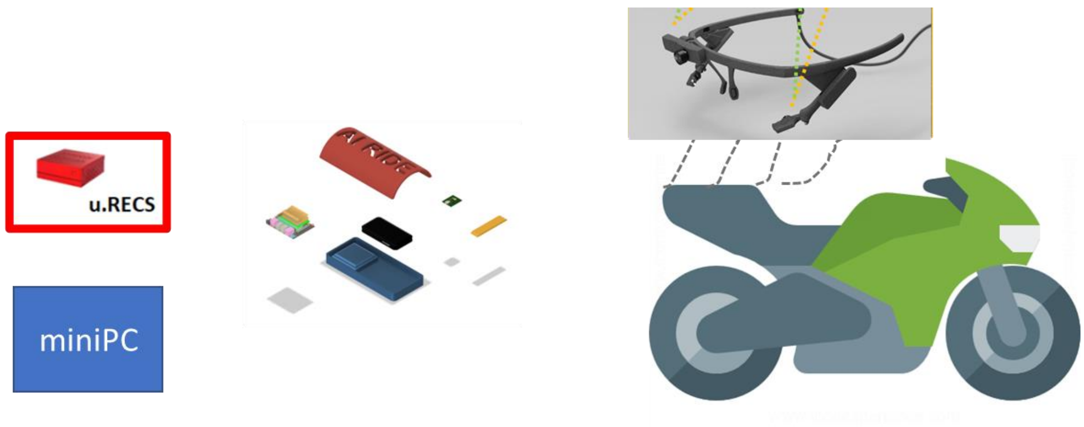


Figure 29 PDC devices onboard integration

4 Evaluation and characterization

DLE evaluation and characterization

The first evaluation concerns the custom object recognition models we trained to detect the characterizing elements of the scenario: the motorbike, the rider and the traffic cones. For this activity we have chosen the version YOLOv5 because, in addition to the excellent performances, it offers optimal support from the developing community.

We experimented different video resolutions and various starting models were taken into account with a variable number of parameters between 2 million and 80 million. It was possible to train very complex models thanks to the use of the Universal System for AI Infrastructure NVidia DGXA100, an highly parallel hardware specifically designed for Deep Learning tasks. We have chosen to use the Medium model (YOLOv5m) for a fair compromise between recognition accuracy and speed of processing.

As shown in Figure 30 the custom model achieves an accuracy of 99% for the recognition of traffic cones, 94% for the driver and 84% for the motorcycle. The latter value can be improved by increasing the number and models of motorcycles present in the dataset.

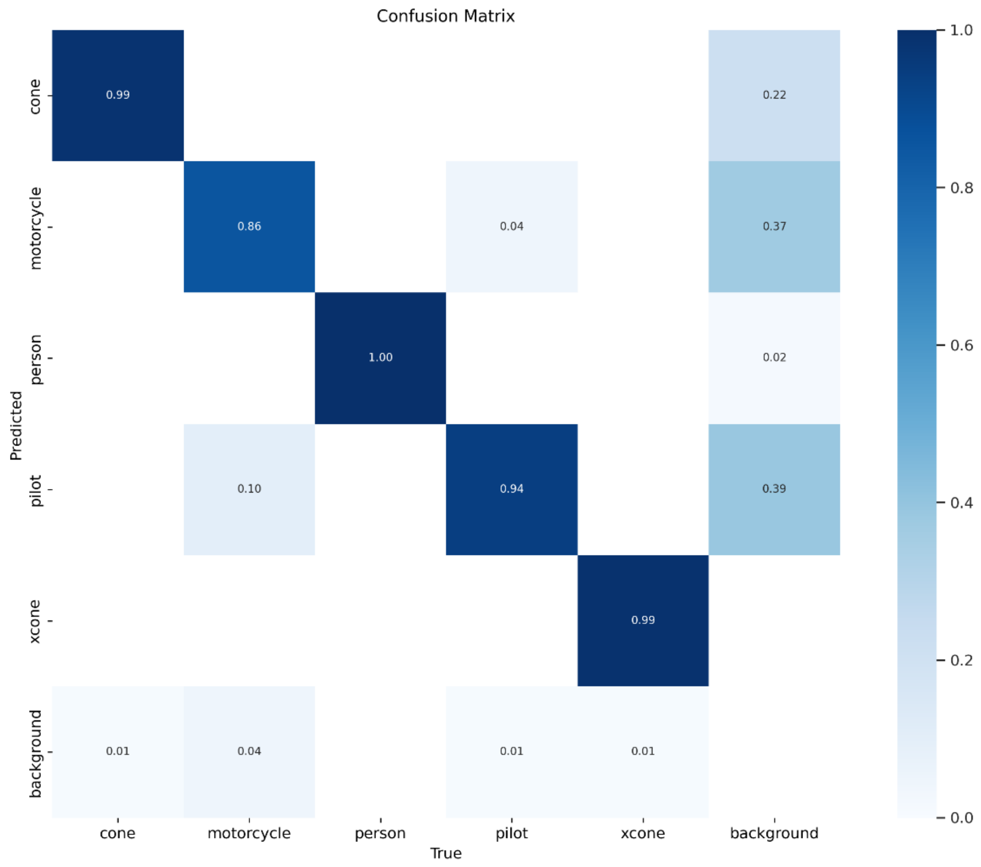


Figure 30 DLE Confusion Matrix: the custom model achieves an accuracy of 99% for the recognition of traffic cones, 94% for the pilot and 84% for the motorcycle.

An example of this recognition accuracy is shown in Figure 31 where the motorbike just touched a cone which is detected in yellow with the Xcone label (fallen cone).

All the data coming from the real world is represented into a 3D model (see Figure 32). The information contained in the 3D model allows the automatic evaluation system to assign a score to the examinee.

The parameters acquired from the real world concern the position of the bike and its projection in the virtual world and events such as a cone touched or a cone dropped.

Real world events are modeled in the virtual world as events in the virtual world and their occurrence invalidates the exam paper giving a low score to the examinee.

The calculation of the quality of the driving takes place by means of the analysis of the route: the maximum and minimum accelerations and the difference between the route traveled by the instructor and by the candidate are taken into account.
If the two tracks are identical, a score of 1 is obtained, as we move away from the path followed by the instructor, the score degrades towards 0. In addition to the trivial cases just presented, the final scoring is given by a series of evaluations between 0 and 1. The value is generally calculated as follows:

$$v_i = \frac{s_p}{s_p + \left|\left|s_p - s_e\right|\right|}$$

Where v_i is the computed value on parameter i , s_p is the score of the pilot and $s_e$the score of the student.
The score assigned to each v_i depends on the kind of analyzed data. Hit a cone or going off track provides a score 0 (the lack of penality has a score 1), the other value scores details are as follows:
- comparing the area covered by the two tracks (v1)
- comparing the completion time (v2)
- comparing the correlation coefficient between the two tracks for the axis x (v3)
- comparing the correlation coefficient between the two tracks for the axis y (v4)
- comparing the maximum speed (v5)
- comparing the minimum speed (v6)
- comparing the statistical variance computed on the speeds (v7)
- comparing the standard deviation computed on the speeds (v8)

The final score is simply computed as follows:

$$v = \prod v_i$$

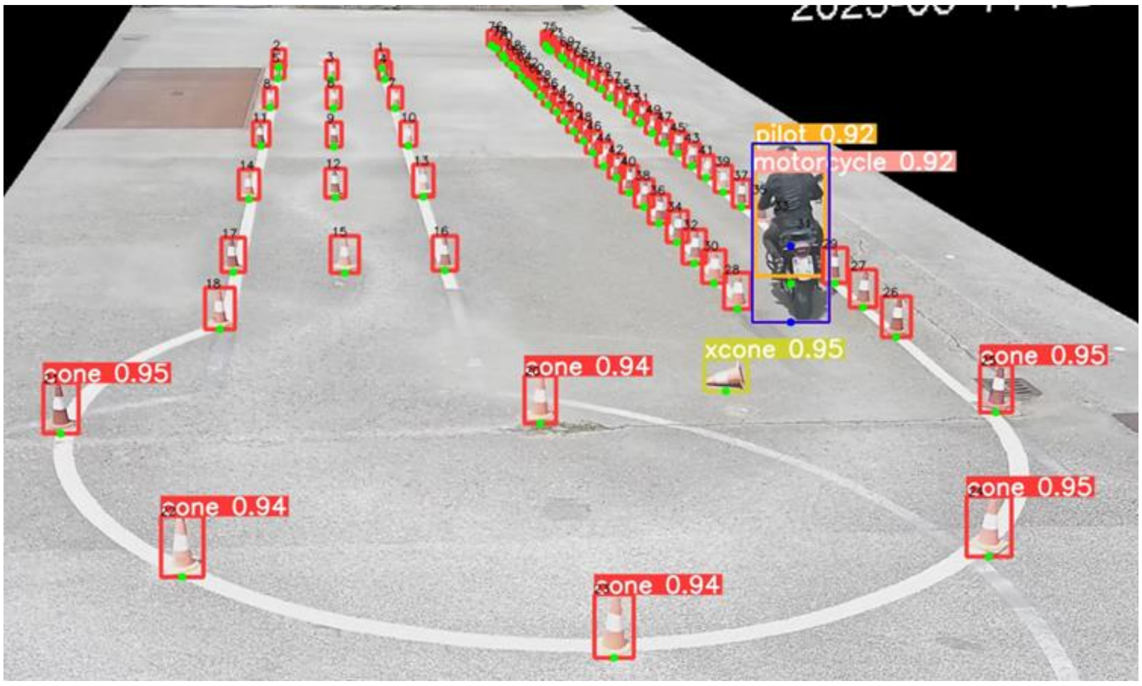


Figure 31 Example of all the class detected with a confidence greater than 92%(xcone represent the fallen cone).

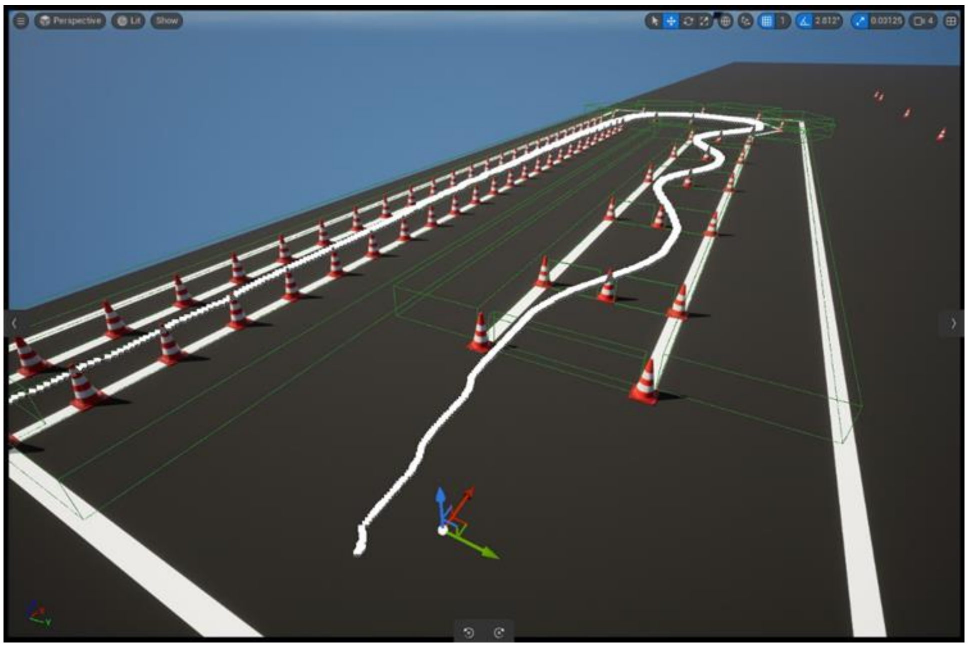


Figure 32 3D environment.

The guides used to test the system have the following characteristics:

- guide 1: guide carried out by an instructor (s_p)
- guide 2: guide judged satisfactory by the pilot instructor (s_{e1})
- driving 3: driving judged excessively aggressive and fast by the instructor pilot (s_{e2})
- driving 4: driving judged excessively slow and uncertain by the pilot instructor (s_{e3})

The values shown in Table 4 shows the evaluations obtained. In Figure 33 the graphical outcome of two different driving sessions are depicted by the system for the comparison evaluation.

Table 4 DLE Driving quality score

Drives	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_{tot}
S_p-S_{e1}	0.986	0.935	0.973	0.998	0.809	0.859	0.964	0.886	0.310
S_p-S_{e2}	0.974	0.388	0.931	0.994	0.885	0.311	0.644	0.601	0.03
S_p-S_{e3}	0.991	0.736	0.980	0.994	0.644	0.620	0.654	0.575	0.107

The scores obtained are preliminary. They separated good driving from bad driving demonstrating good performance although further study is needed to fine-tune them.

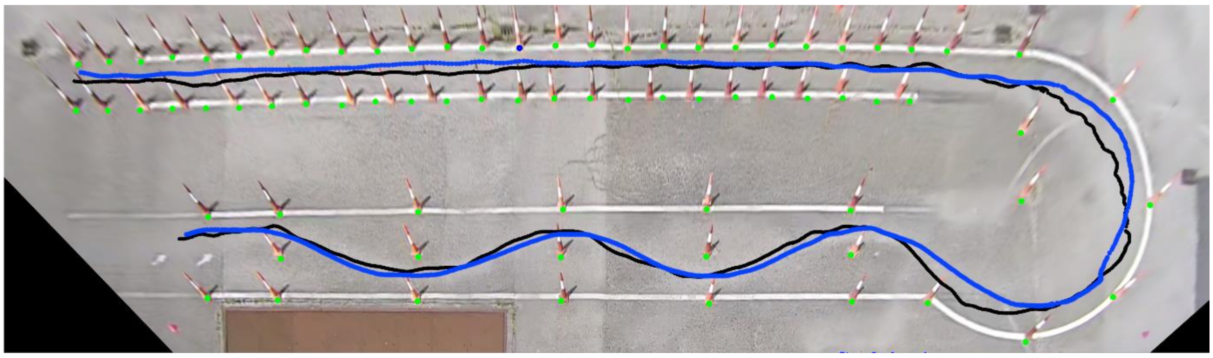


Figure 33 Driving quality: two sessions compared (session1 in blue and session 2 in black).

PDC evaluation and characterization

This use case aims to test a mobile device mounted on a motorbike, exploring the possibilities of a small edge low-power device that processes data.

The battery-operated device is connected to a local area network (LAN) through Wi-Fi, and is equipped with Pupil Labs glasses that detect the driver's gaze, and an inertial measurement unit (IMU) to catch the vehicle's inclination on three axes, the small battery can power the hardware for approximately 1 hour.

The selected device for the use case is a u.Recs appliance with two modules; the first is an x86-based computer with a Windows operating system that runs the software for the smart glasses, and includes two Python applications.

The first application is connected to the glasses software and detects whether the driver is looking in the correct direction. The second application sends the front camera image from the glasses to the second module.

The second computing module provides an NVIDIA GPU, and is directly connected to the first one via a high-speed Ethernet connection. It operates an artificial intelligence engine that runs an inference model for object detection, the results are sent to the first Python program.

Using the information from the artificial intelligence module, the program can determine the correct position of the driver's gaze based on the detected objects on the road, indicating whether the driver's gaze is valid.

Every data, including the driver's gaze, gaze errors, and object detection results, are transmitted in real-time to the driver instructor through the Wi-Fi connection.

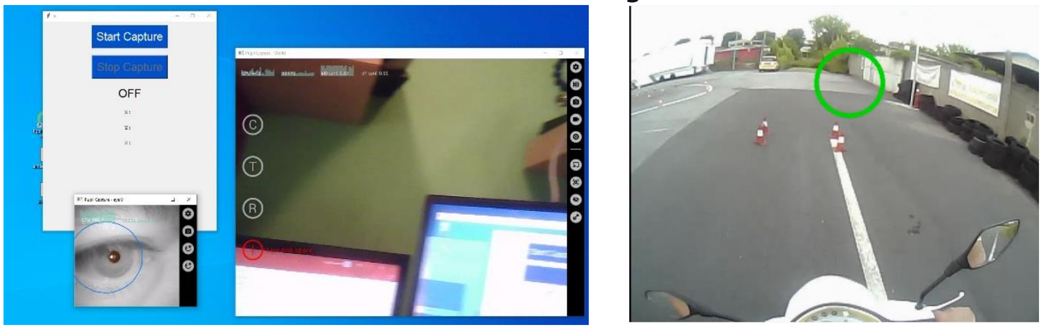


Figure 34 PDC instructor control board.

By employing optimization techniques, we have achieved the following results on the u.RECS:

- The IMU output from the Windows module can be registered at 30Hz, providing information about the vehicle's inclination on three axes.

- Using a neural network model optimized with NVidia's TensorRT software and 16-bit precision, by setting the power mode on the Jetson module to maximize the computational power, the Jetson module can run the object detection algorithm at 30/32 frames per second (fps).

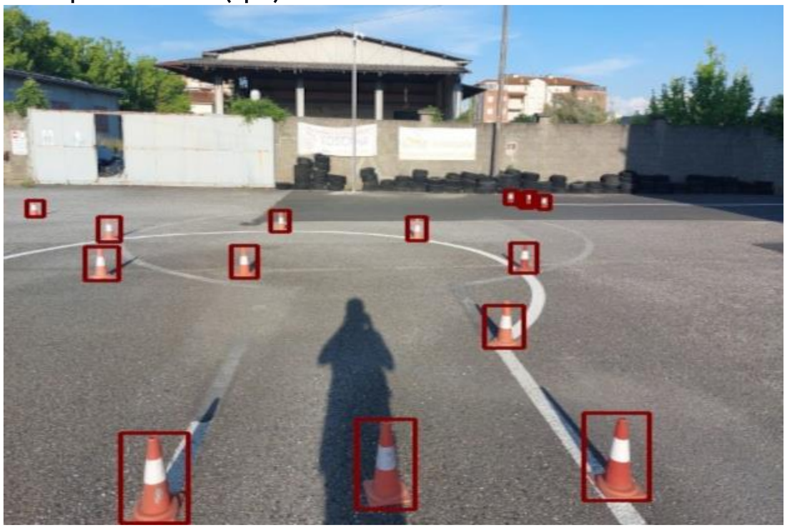


Figure 35 Object detection – traffic cones

The system can deliver the output to the driving instructor's tablet in real-time, allowing immediate feedback and assessment.
With this setup, we seek to enhance training and safety by monitoring the driver's gaze and correcting potential hazardous behavior during the activity on the road.

AI-RIDE Demo Event
The final demo event was held in Pontedera (Pisa, Italy) on June 14th, 2023 at the AUG trackside. A special paddock has been used to host the demo sessions, the speakers and the audience. Invited people from CNIT, Scuola Superiore Sant'Anna university, CNR. The coordinator of the VEDLIoT project, Jens Hagemeyer, attended the event and presented the VEDLIoT project.
The following list of representatives of different stakeholders have been invited: Paolo Colangelo (CONFARCA president, association of Italian Driving Schools), Fausto Fedele (Ministry of Infrastructure and Transports, general director of Transportation General Direction), Emiliano Malagoli (Association of disabled people drivers), Roberto Tommasi (Civil Motorization), Fausto Nosenzo (Handytech).
The demo showed a complete session of the two use cases running in the LSB track. The PDC use case have been run using the VEDLIoT u.RECS, the Pupil Labs Core glasses worn by Gianmarco Corona, AUG instructor. After calibration, multiple track laps were covered. Video, pupil detection and feedback (green or red spot on the video) were fully processed onboard in the u.RECS and data were sent using WiFi connectivity and projected in real time to the audience, showing different driver behavior capabilities in the straight part of the track and in the curve. The DLE use case have been run resorting to the workflow running over a single IP camera of the LSB track. .With respect to the planned use of both VEDLIoT u.RECS (PDC use case) and t.RECS (DLE use case), in accordance with the VEDLioT team the t.RECS has not been considered for shipping due to the need of dedicated GPU resources inside the edge node with PCIE4 bus bandwidths and speeds. Future model deployments, more suitable for product releases, will consider the adoption of lighter GPU resources allowing the use of Jetson-like external devices attached to the edge node.

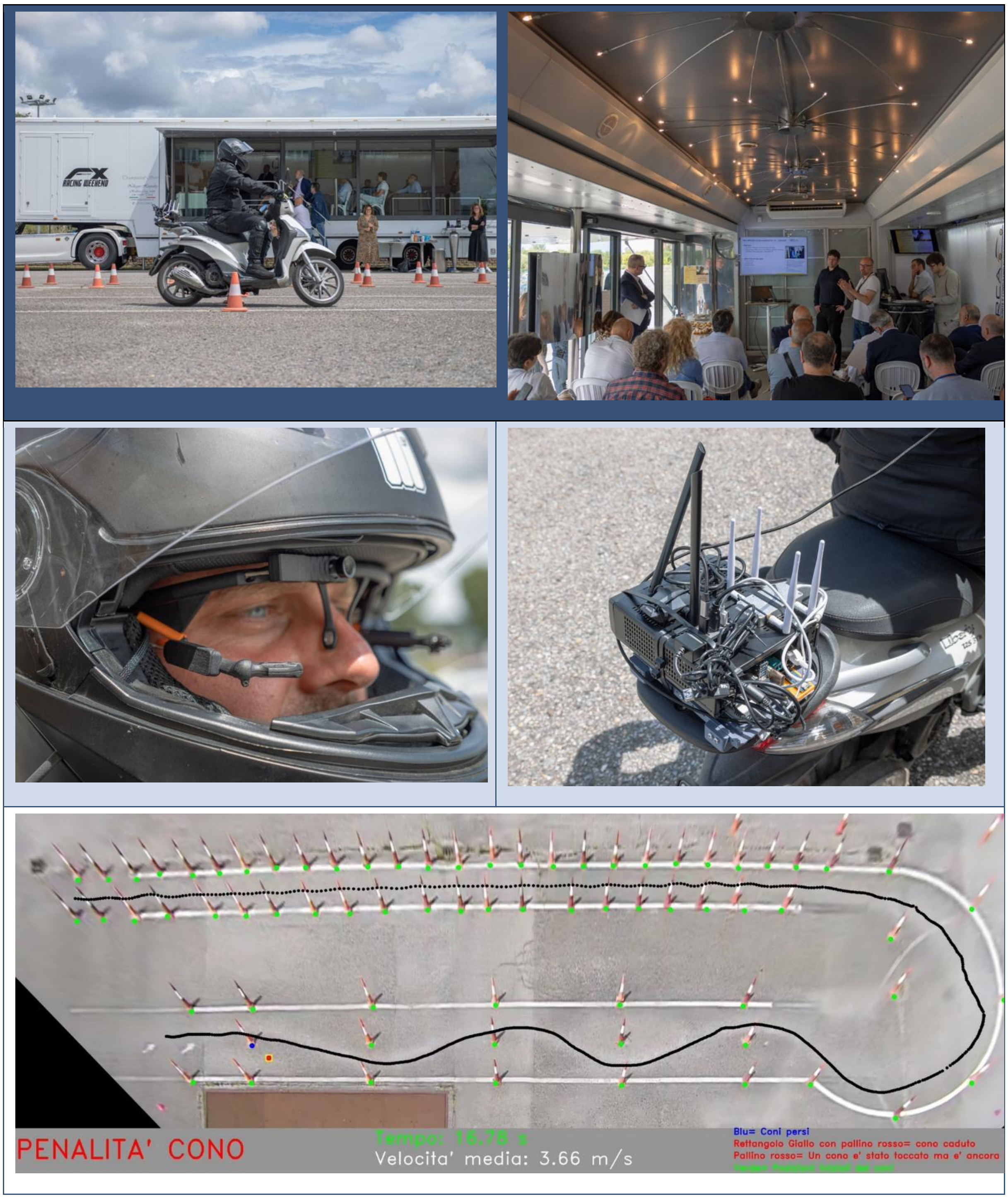

PENALITA' CONO
Tempo: 16.78 s
Velocita' media: 3.66 m/s
Blu= Coni persi
Rettangolo Giallo con pallino rosso= cono caduto
Pallino rosso= Un cono e' stato toccato ma e' ancora

Figure 31 Demo event pictures: paddock and track, event session, instructors driver equipped with Pupil Labs Core glasses, onboard edge node equipped with u.RECS, DLE demo trial with touched cone penalty.

5 Exploitation

The two AI-RIDE solutions has encountered a significant interest from different type of stakeholders. The AI-RIDE project aimed to design and test automated solutions for enhancing the training and testing of motorcycle practical driving courses and driving license

exams. In terms of training, the lack of tools and automated solutions has resulted in practitioners receiving limited feedback on their driving performance, leading to considerable time and resource investment in training. The AI-RIDE project seeks to address these limitations by providing practitioners with efficient and automated driving training that is personalised to their learning needs. This will not only reduce training hours but also improve their driving skills and safety on the road.
The current state of the art system, mainly based on scores given by instructors and examinators, lacks standardised, measurable, and auditable means for evaluating practical exams. This has resulted in significant variations in exam failure rates across different regions and countries. To address this issue, the AI-RIDE project provided a first contribution to a reliable and rigorous evaluation methods that ensure fair and equitable treatments for all practitioners. Additionally, the project will enable the use of a reliable scoring system that provides more accurate and measurable information on typical exam errors and unsafe driving procedures. The impact of the AI-RIDE project was recognized especially from the public sector in Italy. The invited representatives of the Ministry of Infrastructures and Transport and the Civil Motorization during the final Demo Event highlighted that such kind of system has the potential to become a revolution in the field of Driver Licence Exams. The ideas and the workflows of AI-RIDE have the potential to become a real application product available to driving schools and certified by the Ministry. Moreover, the interactive driving by measn of wearables and the near-edge processing solution proposed by AI-RIDE has raised the interest of driving schools and professionist drivers, whereas such systems may have a significant applicability for professional drivers, but also for disabled drivers.
The ideas and the proof-of-concepts of AI-RIDE have the potential to revolutionise the motorcycle practical driving training and testing landscape by providing automated and reliable solutions that enhance safety, efficiency, and equity.

The AI-RIDE solutions, developed as initial proof-of-concept demonstrators, will be carried out through additional financing opportunities, ranging from Italian public sector financing, industrial sector financing and European project opportunities. More than 350.000 motorcycle test exams take place every year in Italy. Since 2006 (Directive 2006/126/EC of the European Parliament on driving licences), very similar exam procedures have been defined in EU Countries (and worldwide). Thus, the AI-RIDE solution, subject to possible Country-specific minor adaptations/retraining, has a potential market applicability over millions of exams/training sessions and implementation over thousands of test sites. AI-RIDE Partners have identified key short term customers for the AI-RIDE over VEDLIoT solution to push the business towards success. These customers include public and private driving institutions and associations interested in automating driving teaching and examination procedures. It is important to note that current procedures will soon become obsolete and not be able to cope with the evolution of vehicles and V2X technologies.

Other customers and partners have also been identified, including large transportation companies looking to improve safety conditions, reduce accident-related costs, and improve ecological driving, leading to 15% cost reductions in vehicle maintenance. Insurance dealers aim to profile driving styles and adapt ranges to individual driver risk and ability/skills. Key players in the market of motorcycle-related solutions and advanced driver-assistance systems also show interest. Telco and Telco-OTT aim to provide new innovative services through the 5G infrastructure.

The final AI-RIDE&VEDLIoT demonstration has been targeted to specific transportation authorities and associations. AI-RIDE partners are part of an active industrial ecosystem including worldwide leaders in motorcycle production, Ericsson, and UFO. Further research activities are envisioned to enable real-time interaction with the vehicle, introduce additional IoT sensors and integrate it with the motorbike telemetry system, exploit low-latency 5G/6G connectivity, and enable real-time interaction with smart wearables and tactile solutions.

6 Summary and next steps

This document provided an overall description of the activities carried out in the AI-RIDE project, focused on motorbike practical courses and licence exam automation techniques using near real time edge node reousrce provided by the VEDLIoT project. Starting from the beginning of the project, the proposed activities of the three different use cases have been planned, with special focus on the track setup activities, with a careful study of the required tools, the type and placements of the cameras and the wearable open libraries study. The project aimed at designing and developing the proposed use cases, including the PDC use case, where data acquired by a number of different sensors from the Pupil Labs glasses were used to augment the driving course perception with real time driving suggestion, and the DLE use case, where a federation of external cameras covering the exam tracks send video streams to perform data fusion and compute a set of graphic-based metrics and outcomes. The AI-RIDE & VEDLIoT project's final demonstration showed live performance of the two use cases and achieved a noticeable interest to stakeholders such as transportation authorities and associations and industrial partners involved in the motorbike ecosystem. Future research activities will focus on real-time vehicle interaction, integration of additional IoT sensors with the motorbike telemetry system, leveraging low-latency 5G/6G connectivity, and enabling real-time interaction with smart wearables and tactile solutions.

VEDLIoT
Very Efficient Deep Learning in IoT

ICT-56-2020 - Next Generation Internet of Things

Final technical report FLEDGED VEDLIoT Open Call

FLEDGED

Feasibility of **L**ow-energy **E**mbedded **D**eep-Learning-Models **G**eared to **E**dge **D**evices

Document information	
Project website	https://www.vedliot.eu
Dissemination Level	PU
Nature	R
Contractual Deadline	30.06.2023
Author	iBreve Ltd
Contributors	FLEDGED Team

The VEDLIoT Project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957197.

Changelog		
Version	**Date**	**Type**
V 0.1	12.06.2023	Initial draft
V 0.2	23.06.2023	Revision 1
V 0.3	27.06.2023	Revision 2
V 1.0	29.06.2023	Finalisation

Table of Contents

List of Figures

List of Tables

List of Tables	
Table 1	Quantization types with expected size and accuracy reductions
Table 2	Overview of advantages and disadvantages by deployment type
Table 3	Comparison of widely used Nordic Semiconductor SoCs

List of Abbreviations

List of Abbreviations	
VEDLIoT	Very Efficient Deep Learning in IOT
FLEDGED	Feasibility of Low-energy Embedded Deep-Learning-Models Geared to Edge Devices
ML	Machine Learning
AI	Artificial Intelligence
LSC	Linear shaped-charge
GDPR	General Data Protection Regulation
COPD	Chronic obstructive pulmonary disease
ACOS	Asthma-COPD overlap syndrome
ROC	Receiver Operating Characteristic
RR	Respiratory Rate
RAM	Random-access memory
CPU	Central Processing Unit
FPGA	Field Programmable Gate Arrays
BLE	Bluetooth Low Energy
SoC	SoC
PMIC	Power Management Integrated Circuits
CI	Continuous integration
API	Application Programming Interface
UART	Universal asynchronous receiver-transmitter
AES	Advanced Encryption Standard

Executive summary

This report provides a comprehensive overview of the FLEDGED project from the technical point of view.

Section one introduces the idea behind the project and provides the reader with a broader understanding of the needs for health monitoring with wearable medical devices in chronic diseases.

Section two highlights the overall goal and approach the project has taken. The positioning within the VEDLIoT project is provided through an architectural system overview.

Section three is dedicated to the implementation throughout the project. It is divided into three subsections, starting with computations in the central cloud and working its way towards edge devices and finally over the edge & far-edge devices.

Subsequently, section four contains evaluation & characterization of the implementation and highlights key results & analysis of the use case.

In section 5, the exploitation of the project is described and the impact specified.

Finally section 6, provides a summary & next steps.

1. Introduction

Living with chronic diseases is much more than just taking medicine. Chronic diseases must be managed 24 hours a day, 7 days a week, 365 days a year and this often requires continuous health monitoring. Nearly 1 in 4 Europeans lives with a chronic illness. This drastically diminishes their overall quality of life and affects almost every aspect of their daily routine.

Some of the most common ones are cardiovascular diseases, neurological diseases and mental conditions. One of the most common disorders of the nervous system is epilepsy affecting over 50 million people. There is no cure and for many patients none of the available medication really works to stop their seizures.

For people affected by chronic diseases like epilepsy, reliable tools for continuous monitoring are very important. Here, novel technologies that enable deep learning & edge computing can be of immense help.

In health monitoring the data quality and computational requirements can be very high. Most current approaches are only available in controlled environments or use a central cloud for analysis instead of locally on the device.

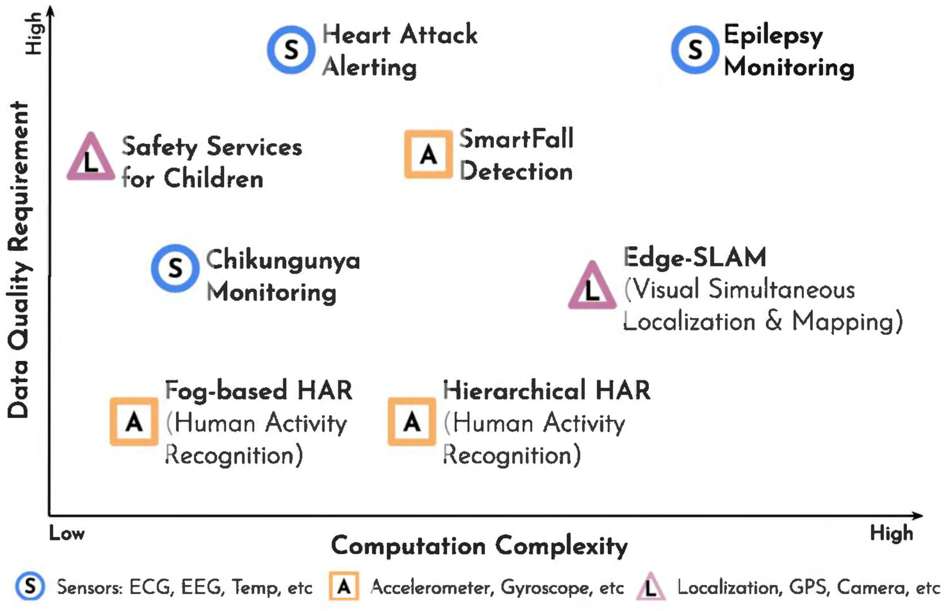


Figure 1: Relation of Data Quality Requirements on Computational Complexity for Health Monitoring Use Cases [1]

However when it comes to alert functions like for an epileptic seizure or a heart attack it can be crucial to move computation 'over the edge' directly to the end device. On top, the use of deep learning requires a lot of computing power and energy. Both are often unavailable or very limited locally on site. And if we think about the future, the amount of computing and energy power required will increase even more.

[1] Figure based on Xinqi Jin, Lingkun Li, Fan Dang, Xinlei Chen, and Yunhao Liu. 2022. A survey on edge computing for wearable technology. Digit. Signal Process. 125, C (Jun 2022). https://doi.org/10.1016/j.dsp.2021.103146

We developed a patent pending wearable technology to continuously analyze small variations in respiratory rate, breathing patterns and activity. Our system consists of a wearable device that is connected via bluetooth to our app. Through our app users can access data insights and dedicated exercises.

When fully developed we aim to support patients with different chronic diseases to personalize interventions based on the monitoring data.

So why breathing and not heart rate or brainwaves? Our respiratory rate is actually a fundamental vital sign. Breathing is one we can actively control but which also works on its own. It is very sensitive to many different adverse health events like clinical deterioration, during a seizure or cardiac events, but also to different other stressors like emotional stress, cognitive load, heat & cold and physical effort. What is interesting is that the sensitivity of respiratory rate to these conditions can actually be superior compared to that of other vital signs.

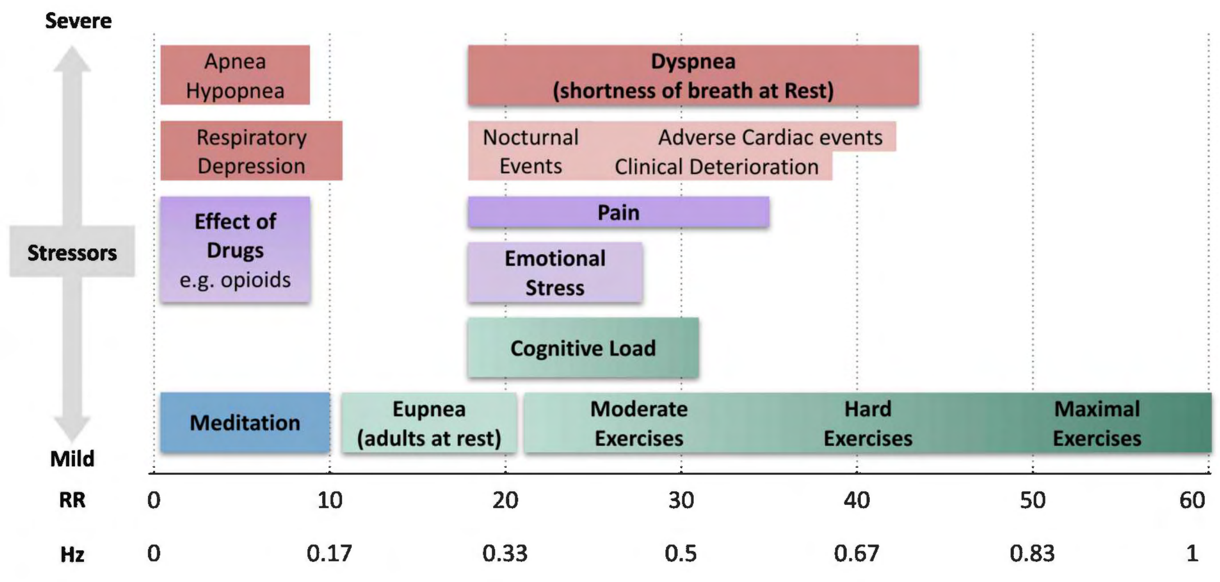


Figure 2: The Impact of Stressors on the Respiratory Rate [2]

Healthy adults normally are breathing around 12 to 18 breaths per minute at rest. Physical & mental activity can increase this, but studies have shown that also pain and clinical deterioration are highly linked with an increased breathing rate. Accurately monitoring our breathing rates has very high potential in many different clinical fields.

[2] Figure based on Nicolò A, Massaroni C, Schena E, Sacchetti M. The Importance of Respiratory Rate Monitoring: From Healthcare to Sport and Exercise. Sensors. 2020; 20(21):6396. https://doi.org/10.3390/s20216396

2. Idea and Architecture

2.1 Overall Idea

The overall goal of our FLEDGED project was to assess the feasibility of AIoT for the healthcare use case and VEDLIoT frameworks and technologies to advance the next-generation wearables for a deployment in healthcare for continuous monitoring. We were particularly interested in exploring the potential to optimize performance related to deep learning, energy consumption and security improvements.

We chose our project name as Fledged itself is defined as raising a young bird until it is ready to leave the nest and become able to fly. So we thought it very fitting for the challenge of making the next generation healthcare wearables ready to take off.

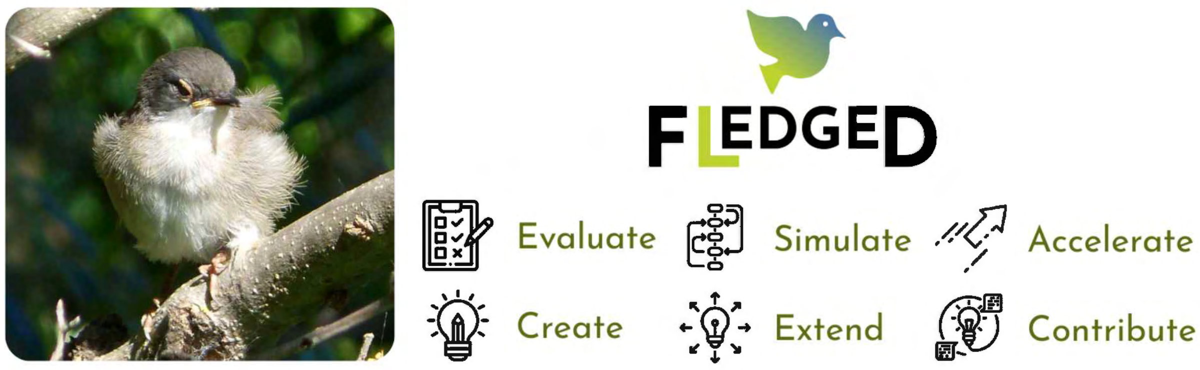


Figure 3: Overall Project Objectives [3]

We divided the 12-month project into 3 subsequent phases. Phase 1 was dedicated to the assessment of the needed use case specific requirements adaptations. In Phase 2 the VEDLIoT concepts were translated to be adopted into the project's use case analysis for hardware & software based optimizations. Within Phase 3 the analysis was concluded, the results documented and exploited. Communication & dissemination activities were performed throughout the project & intensified during Phase 3. The Work Plan is structured into 5 work packages (WP) with four functional WPs and a project management WP. For a detailed description see also the section on the overall management structure of the project in the management report.

[3] icons: Flaticon.com

2.2 VEDLIoT Relation & Architecture

Depending on the particular health monitoring scenario, a computing continuum is required for AI and Deep Learning applications from the cloud to the edge. VEDLIoT has enabled us to assess important tools for a modular, dynamic, configurable and scalable platform for wearable deployments in chronic disease monitoring.

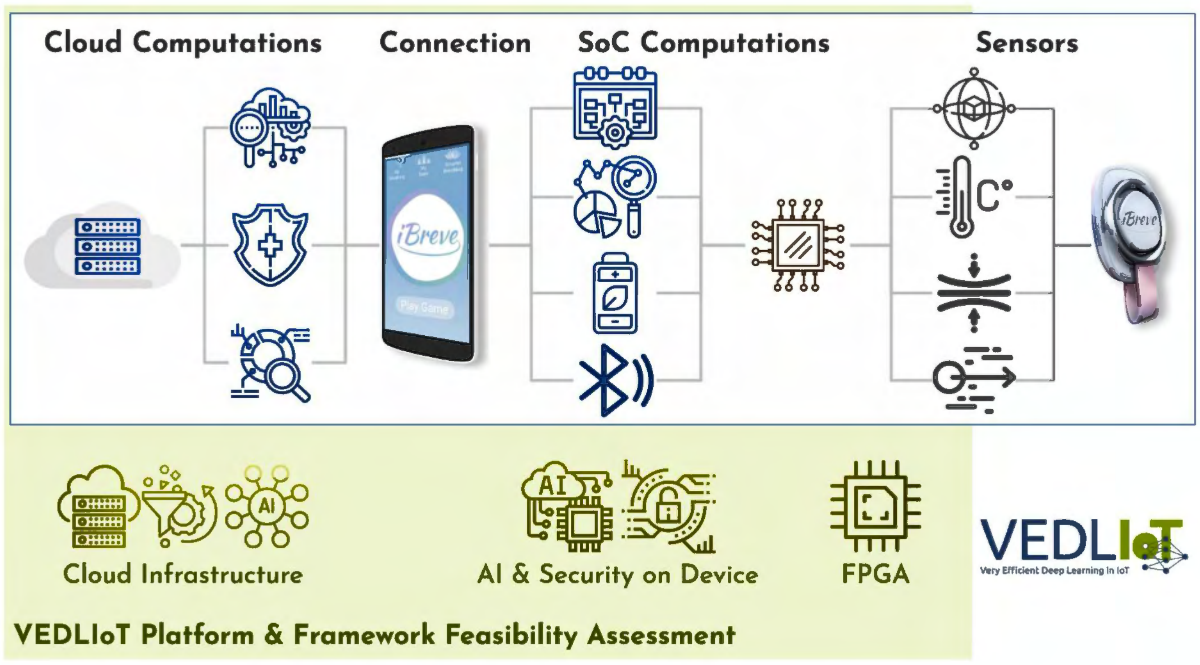


Figure 4: Architecture and FLEDGED relation to VEDLIoT [4]

[4] icons: Flaticon.com

3. Implementation

3.1 Project Plan

During the project, we assessed three main VEDLIoT technology areas, which can tremendously improve wearable systems spanning from infrastructure & tools in the centre, to the edge of the system and even going beyond (far-edge). For a detailed description of tasks see also the section on the explanation of the work per work package in the management report.

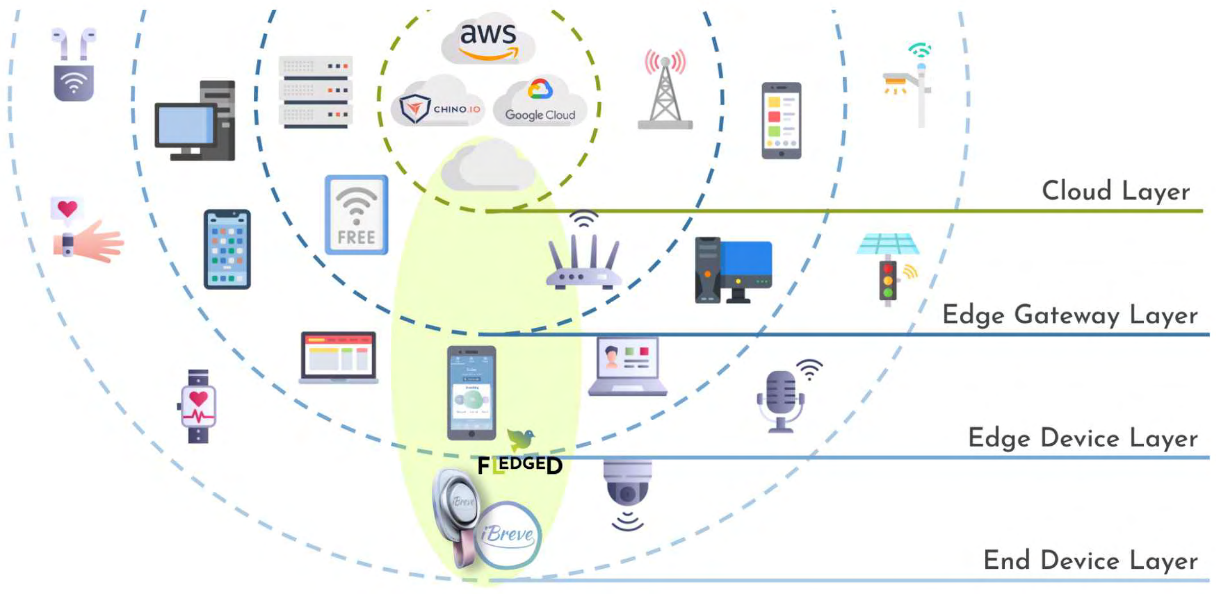


Figure 5: Layers from the cloud towards the edge [5]

Within the project we took a multi step approach that included the training of the ML algorithms of respiratory time series data in the cloud to then evaluate the potential for optimising the model for an edge device. Through an in-depth analysis the data quality was assessed to evaluate the robustness of the model, identify the key features for good accuracy predictions and gather insights for future data collection methods.

Further the feasibility of moving the models to the edge of the edge has been assessed. For this hardware requirements were assessed and potential upgrades identified.

3.2 Centre

3.2.1 Cloud Infrastructure for Health Data Analysis

For continuous health monitoring vast amounts of data have to be processed to train complex deep learning models across multiple patient groups. For this cloud servers are one of the primary go-to-infrastructure. However, due to their centralized nature they have

[5] icons: Flaticon.com

drawbacks related to privacy & security and it is key to ensure the protection of sensitive data.

Within a regional cloud infrastructure set up specific attention needs to be given to the resource requirements, scalability parameters and security aspects of the system. The highest value for the integration of the system into a regional cloud server is achieved in the medical use case for in-hospital on-site environments by increasing compliance with regulations for the handling of medical data.

For training ML models with large sets of fully anonymized training data cloud servers like Google Cloud[6], Amazon Web Services[7] or Microsoft Azure[8] offer the highest agility & scalability. To further comply with GDPR and specialised medical regulation data security solution providers for digital health companies like Chino.io[9] can be integrated into the cloud system architecture.

3.2.2 Training an ML Model for the Prediction of Respiratory Diseases

To build out the backend cloud infrastructure a labelled dataset of respiratory recordings was used.

The set contains recordings[10] from an electronic stethoscope of 112 people for the classification of respiratory diseases. It was chosen due to its feature similarity to the time series data that will later be generated & assessed on the edge device.

The data set includes respiratory sounds from several ailments including asthma, heart failure, pneumonia, bronchitis, pleural effusion, lung fibrosis, and chronic obstructive pulmonary disease (COPD) as well as normal breathing. The diagnosis and data generation was performed by a specialist physician. Each subset was replicated three times, each corresponding to different frequency filters. In total, the dataset comprises respiratory recordings from 112 individuals, 43 of whom are female and 69 male, ranging in age from 21 to 90 (mean ±SD of 50.5 ±19.4). Out of these, 35 are healthy and 77 unhealthy.

Three different types of filters were employed. The Bell mode filter amplified sounds in the range of [20-10 0 0] Hz, with a focus on low frequency sounds between [20-200] Hz. The diaphragm mode filter increased the volume of sounds in the range of [20-20 0 0] Hz, with an emphasis on frequency sounds between [10 0-50 0] Hz. The extended mode filter amplified sounds in the range of [20-10 0 0] Hz, with a special emphasis on frequency sounds from [50-500] Hz.[11]

For the purpose of developing automated methods that can detect pulmonary diseases from respiratory recordings a LSC model was trained to predict one of seven types of respiratory diseases.

[6] https://cloud.google.com
[7] https://aws.amazon.com
[8] https://azure.microsoft.com
[9] https://chino.io
[10] https://www.kaggle.com/datasets/arashnic/lung-dataset
[11] Kênia KP Menezes, Lucas R Nascimento, Louise Ada, Janaine C Polese, Patrick R Avelino, Luci F Teixeira-Salmela, Respiratory muscle training increases respiratory muscle strength and reduces respiratory complications after stroke: a systematic review, Journal of Physiotherapy, Volume 62, Issue 3, 2016, Pages 138-144, ISSN 1836-9553, https://doi.org/10.1016/j.jphys.2016.05.014.

The following shows an example of a LSC Model Pipeline & classification of respiratory disease types. The pipeline consists of creation of the data set with labelled data, a custom training job, an evaluation of the model and its upload and finally if better then previous registered models a deployment to the endpoint.

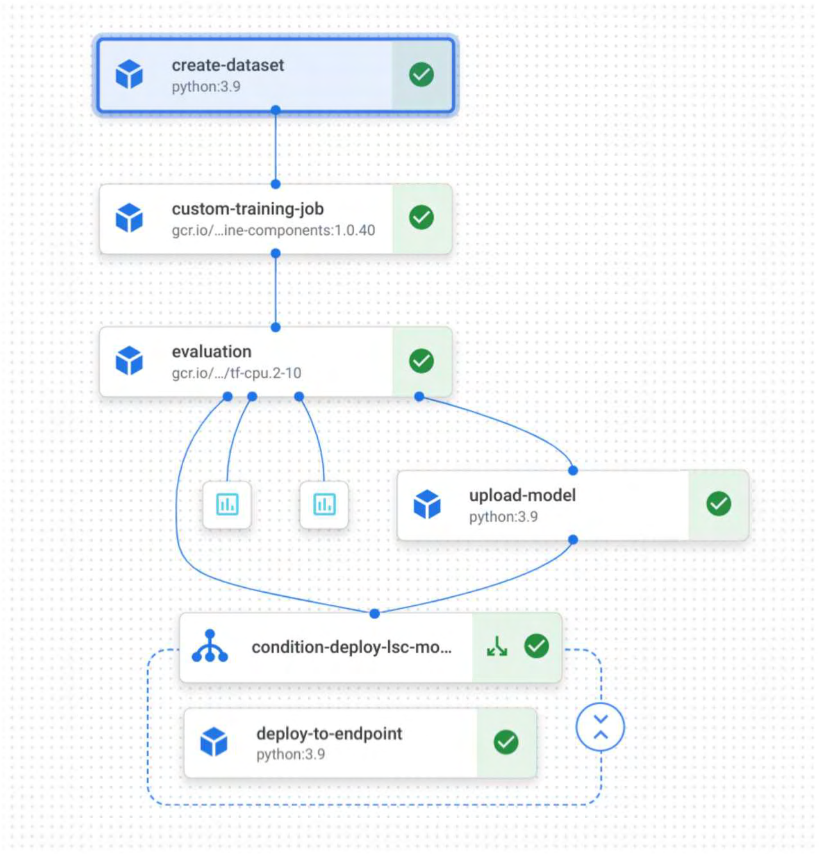


Figure 6: ML Pipeline from data set creation over training, evaluation, upload and deployment to endpoint

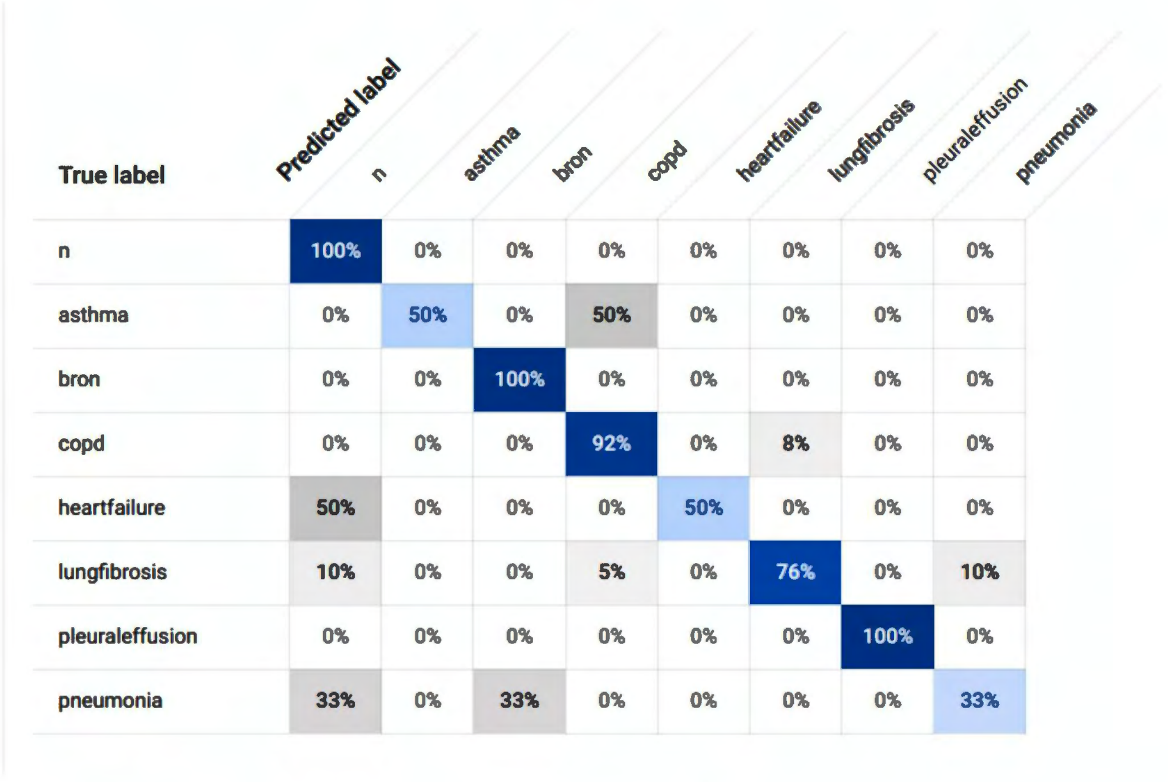
Automatic Model Deployment & Results

Based on the accuracy score we created a conditional cloud function (Condition-deploy-lsc-model) so we can upload the new model to the production and it is automatically deployed (Deploy-to-endpoint).

Further, we created a model registry to check versions & the exact datasets the model was trained on and what type of model is currently deployed.

After training 25 epochs we achieved an overall accuracy of 85%. The model had problems predicting asthma versus COPD. This is also very common in the healthcare setting as both disease symptoms can be very similar. In fact there is Asthma-COPD overlap syndrome (ACOS) and is diagnosed when a patient has symptoms of both asthma and COPD.

The following figure shows a confusion matrix with a table on how often the model classified each label correctly (blue coloured) and which labels were most often confused for that label (grey coloured). This is important for the classification potential.



True label \ Predicted label	n	asthma	bron	copd	heartfailure	lungfibrosis	pleuraleffusion	pneumonia
n	100%	0%	0%	0%	0%	0%	0%	0%
asthma	0%	50%	0%	50%	0%	0%	0%	0%
bron	0%	0%	100%	0%	0%	0%	0%	0%
copd	0%	0%	0%	92%	0%	8%	0%	0%
heartfailure	50%	0%	0%	0%	50%	0%	0%	0%
lungfibrosis	10%	0%	0%	5%	0%	76%	0%	10%
pleuraleffusion	0%	0%	0%	0%	0%	0%	100%	0%
pneumonia	33%	0%	33%	0%	0%	0%	0%	33%

Figure 7: Confusion Matrix

The Receiver Operating Characteristic (ROC) curve is a graph showing the performance of a classification model. The curve plots true positive rate vs false positive rate at different classification thresholds.

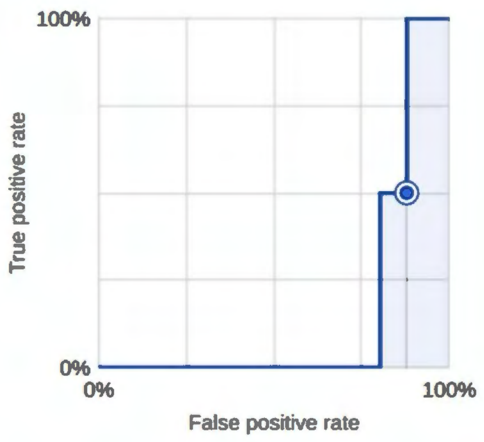


Figure 8: ROC Curve

Takeaways

1. The classification problem is non-linear and solvable by a neural network.
2. A small neural network is deployable on edge devices, in particular on mobile phones.
3. A larger data set is required for more accurate predictions.

3.3 Towards the Edge

3.3.1 Generating Labelled Data from the Edge Device

To build robust ML models it is of utmost importance to use high quality labelled data sets. For this a process to facilitate an automated data labelling process was designed.

Healthy volunteers recorded respiratory time series data with the wearable device in several sitting and standing scenarios with different cognitive loads like reading or playing cognitive games.

Goal

The main goal for the model was the classification of diaphragmatic and intercostal respiration with a neural network consisting of 2 convolutional layers.

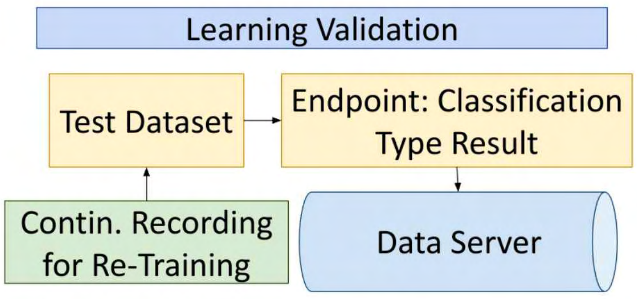

Learning Validation
Test Dataset
Endpoint: Classification Type Result
Contin. Recording for Re-Training
Data Server

Figure 9: Learning Validation on Runtime Level

Further this test was important to determine whether major changes would be needed in future data collection and measurement setups. During the analysis, a pre-feasibility test is done to understand the classification problem more in-depth.

3.3.2 Addressing the Black-Box Problem of Convolutional Neural Networks

Given the application field of the FLEDGED use case in health monitoring it is crucial to have good insights into key features to be able to understand how a model arrives at the final output.

For this reason we did a deeper analysis and built an automated script that enables a scalable replicability with each extension of our datasets.

Automated Preprocessing & Noise Filtering using Fourier filters

After interpolation of time stamps and normalization, we made use of Fourier filters to remove high frequencies and frequencies that are low in absolute amplitudes.

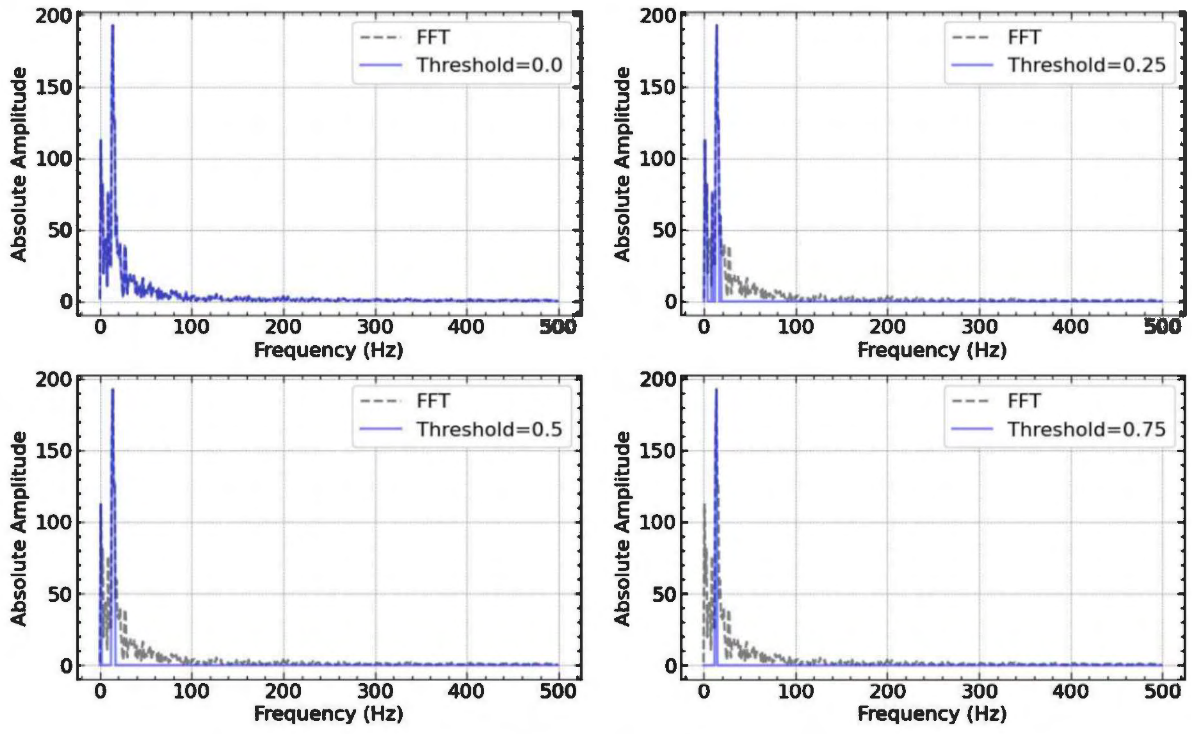


Figure 10: Pre-processing and Fourier Filters

We achieved good results already with lower thresholds.

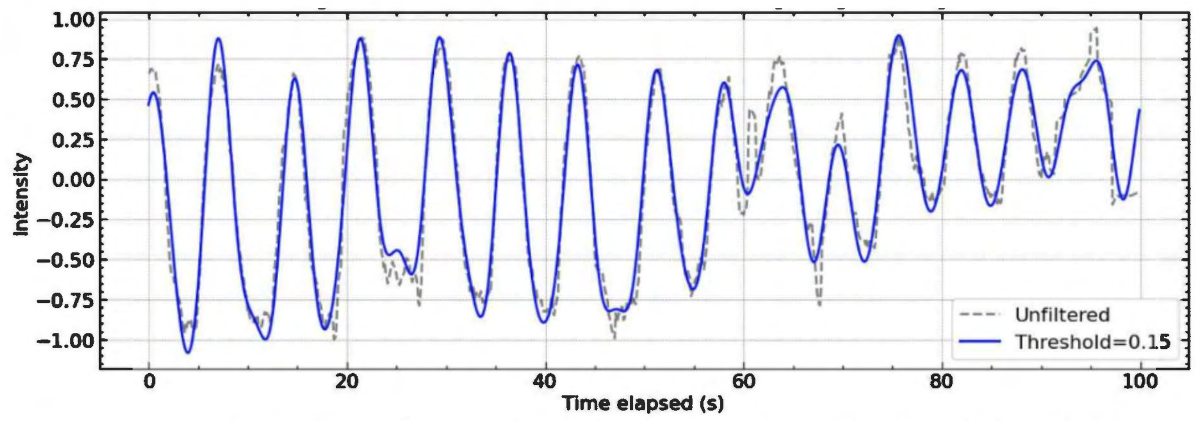


Figure 11: Example graph for Unfiltered vs Fourier filter with a threshold of 0.15

Automated recognition of in-breaths and out-breaths

We created an automated recognition system for differentiating between in-breaths and out-breaths. This can be important to predict characteristics for certain diseases and changes from baseline respiratory patterns.

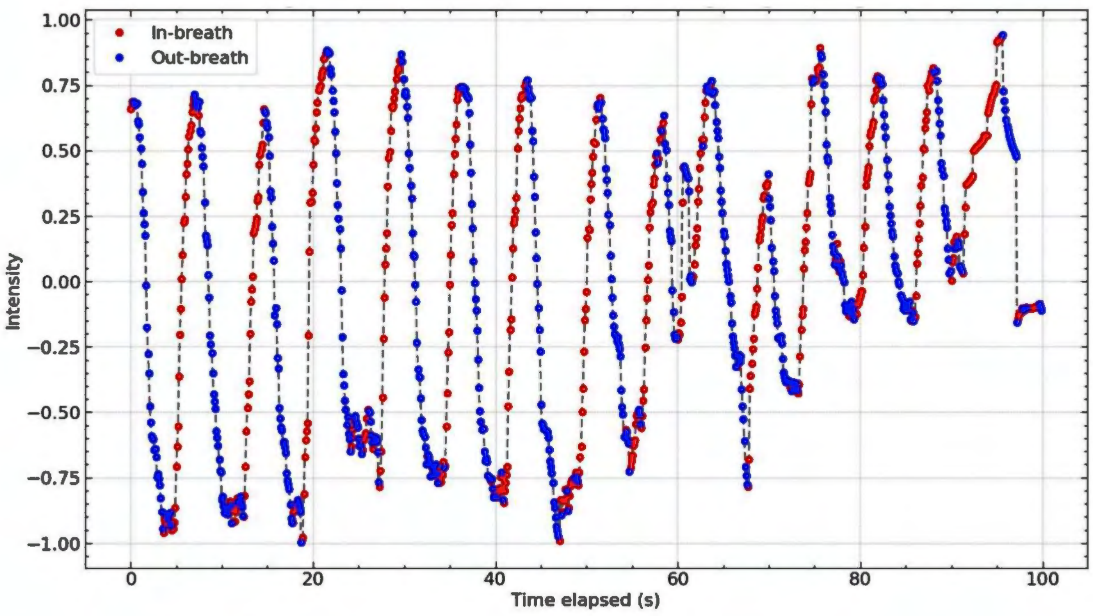


Figure 12: Automated recognition of in-breaths and out-breaths

Also here we used fourier filters for pre-processing to enhance the results.

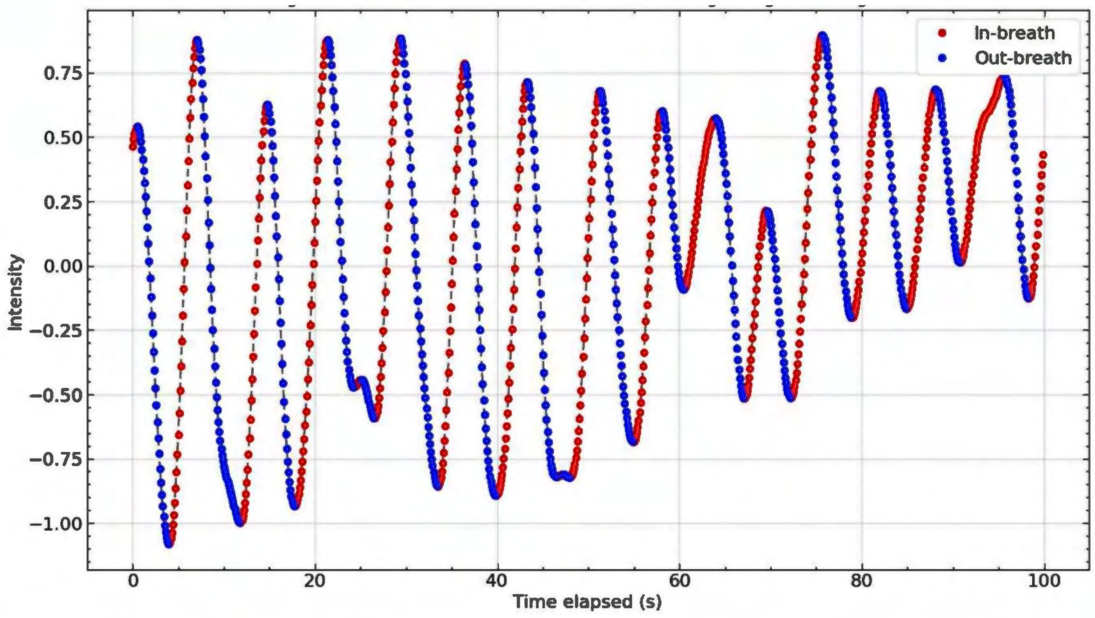


Figure 13: Automated recognition of in-breaths and out-breaths deployed on Fourier filtered time-series

Automated recognition pauses in between in-breaths and out-breaths

Further we created an automated recognition system for detecting pauses of breaths, in particular to detect the small pauses in between breaths, which can be an indicator for relaxation.

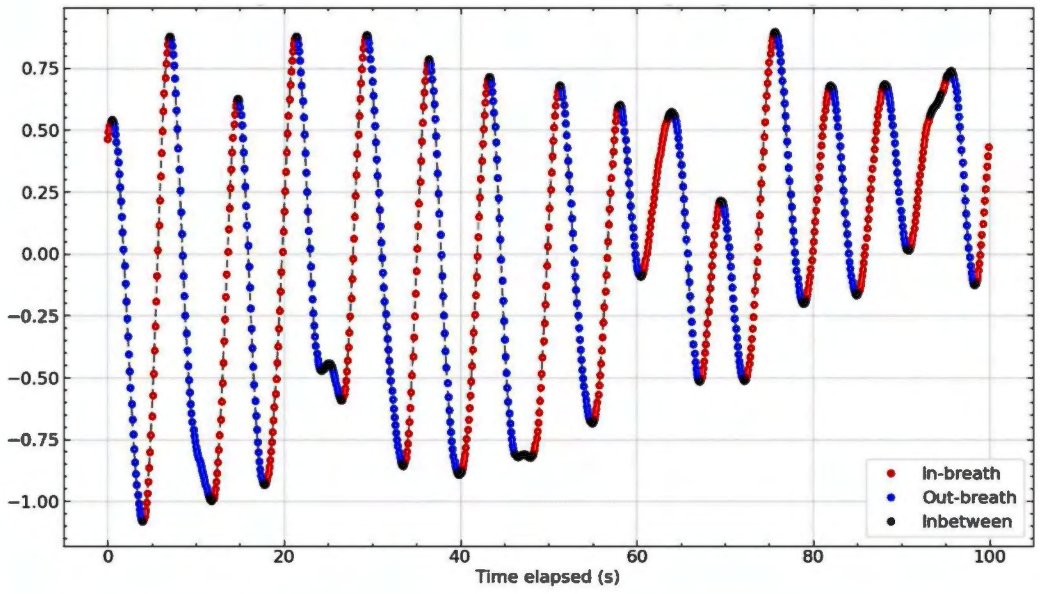


Figure 14: Automated recognition of pauses in between in-breaths and out-breaths

Correlation Matrix

Our calculated correlation matrix shows no obvious correlations between breathing type and activity. A correlation matrix is a table showing correlation coefficients between variables.

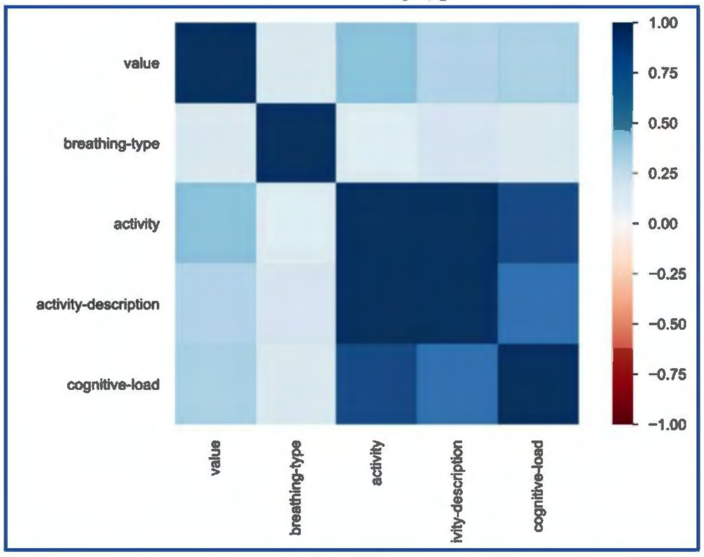


Figure 15: Correlation Matrix - the matrix shows correlation coefficients, where 1 is considered a strong relationship between variables, 0 a neutral relationship and -1 a not strong relationship.

3.3.3 Federated Learning Testing

Designing secure remote monitoring systems that also focus on privacy preservation is immensely important in healthcare and beyond. By moving deep learning towards the edge and the devices themselves, novel privacy-promoting concepts like federated learning could be integrated.

Federated learning is a machine learning technique that trains algorithms across multiple decentralized edge devices holding local data samples without exchanging them and thus enable multiple actors to build a common, robust machine learning model while also addressing data privacy, data security & data access rights.[12]

Status Quo & Existing Challenges

Traditionally, ML & AI models are often trained centrally but on data generated somewhere else. In fact, in healthcare data often comes from multiple places, patients and institutions. Within this process all data is sent to a central server, which in turn can increase vulnerability to the data. One data breach can allow access to a large set of sensitive data. Thus, a central approach has to take adequate measures to protect the privacy & security of sending and storing sensitive data. Sometimes it is not even possible to move all data into one central location, for example out of legal reasons or even the sheer amount of data. In particular, regulations like the GDPR in the EU specify what data cannot be moved from one country to another. Other countries and regions have similar regulatory frameworks like the Personal Information Protection and Electronic Documents Act (PIPEDA) in Canada or the California Consumer Privacy Act (CCPA) in California.

Federated Learning - moving the computation to the data

Instead of sending data to a central computation, federated learning moves the computation to the data. Within a federated learning AI model pipeline, a global AI model is initialised & sent to a selected set of devices, phones or wearables (client nodes), each with their own local dataset. Each node then trains its custom AI model and sends the updated AI model back to a central or local server, which is then aggregated into a new global model that is sent back to the respective nodes for more training. This process can be repeated until the AI model reaches the required standards & accuracy thresholds. This results in an AI model that has been trained on the data of all participating devices without the need to send all the data into a central location and thus is able preserve privacy of each node / user in an enhanced way compared to non-federated learning approaches.

Within healthcare, federated learning offers great potential for improving the accuracy and effectiveness of medical AI models by enabling different clinical stakeholders and hospitals to work together and share data in a privacy-preserving manner. Thus, by training advanced AI models on distributed datasets from different medical institutions, federated learning is able to help with some of the challenges of main personalized medicine. Further, systems built with privacy-by-design can put the user in control, what data is shared and what never leaves the local node. In the future, this will also integrate more and more wearables & remote medical monitoring devices.

[12] Wikipedia, Federated Learning; https://en.wikipedia.org/wiki/Federated_learning

3.4 Edge of the Edge

3.4.1 The Relevance of FPGA's in Health Monitoring with Wearable Devices

Wearables are essentially the endpoints of a network as they are located at the far edge, where bandwidth, latency and energy constraints are most demanding. Here hardware-based acceleration like FPGAs could increase the performance needed to strengthen the computational capabilities for the next generation of healthcare wearables and ensure long term sustainability through the adaptability of the hardware.

FPGAs' parallel architecture and pipelined processing offer high bandwidth & better scale for high throughput apps. They offer increased performance and secure offloading of data paths from CPUs. Beyond bandwidth, latency & security requirements, FPGAs can advance AIoT sustainability due to their programmability & re-configurability meaning hardware can be re-purposed/upgraded when new functionality is required.

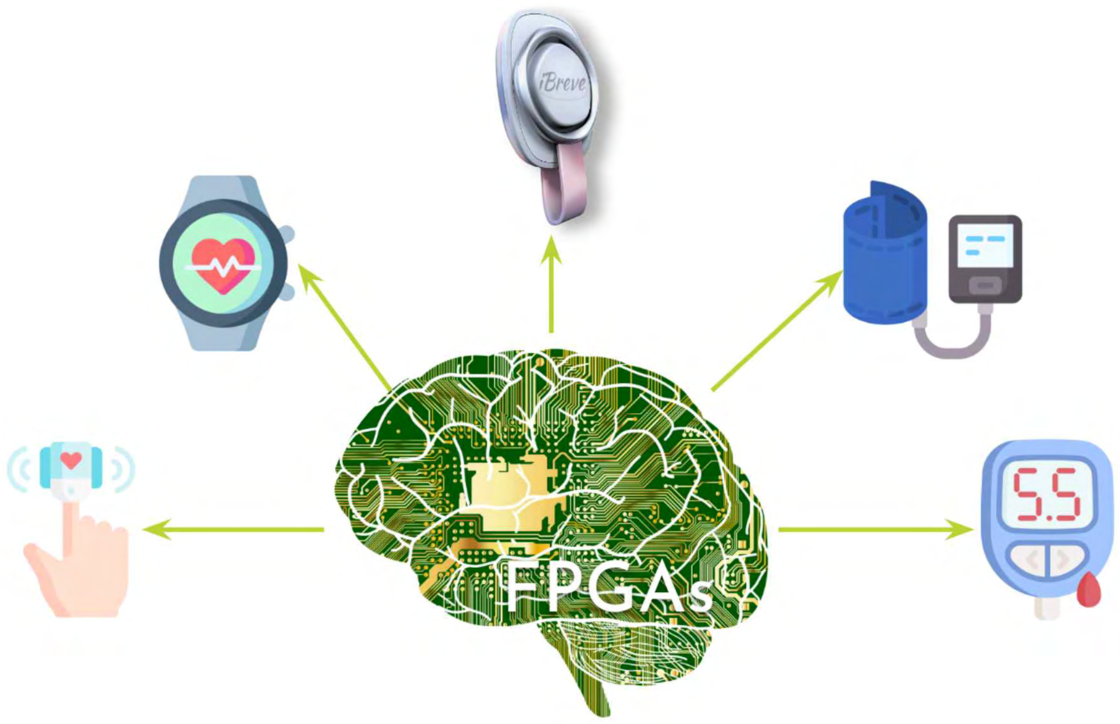


Figure 16: FPGAs for the future of health monitoring [13]

To evaluate the feasibility of hardware-based acceleration for deep learning deployments at the network edge, we conducted an assessment of the available hardware options that fit our overall performance & size requirements. In particular we assessed Silicon Witchery's S1 Bluetooth-FPGA System On Module (powered by Nordic's nrf52811) and Nordic Semiconductor's powerful nRF52840 System on Chip.

[13] icons: Flaticon.com

Silicon Witchery's S1 Bluetooth-FPGA System On Module

In our research we identified a promising FPGA module for our application that has been released by Silicon Witchery, the 'S1 Bluetooth-FPGA System On Module'. It is based on Nordic's nRF52811 SoC and the low power FPGA - Lattice iCE40 Ultra Plus.

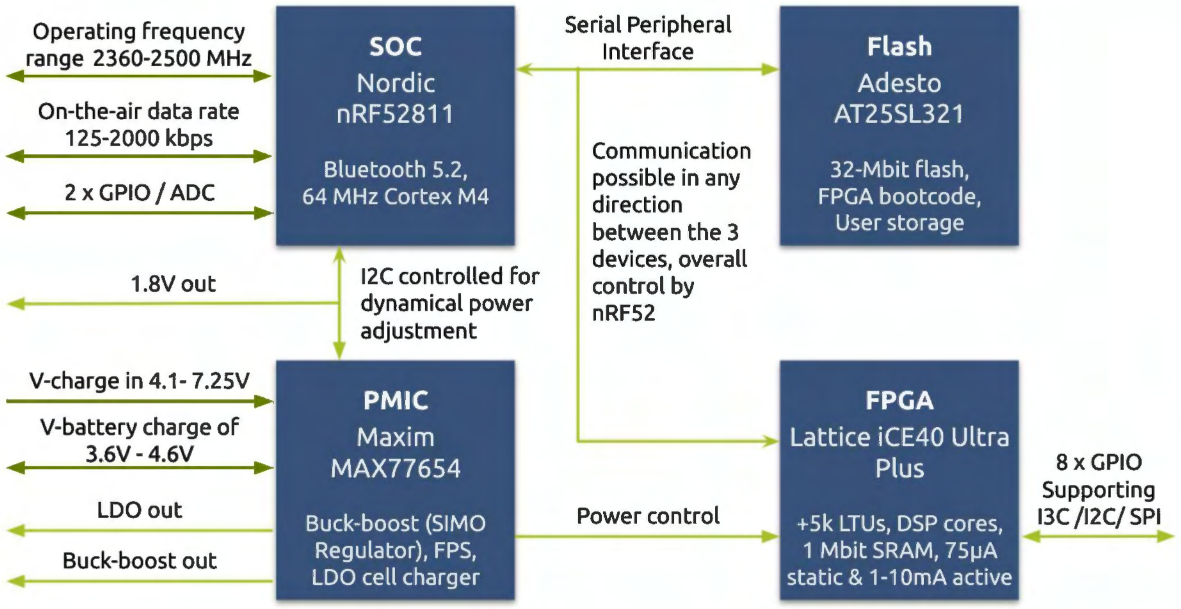

Operating frequency range 2360-2500 MHz
On-the-air data rate 125-2000 kbps
2 x GPIO / ADC
1.8V out
V-charge in 4.1- 7.25V
V-battery charge of 3.6V - 4.6V
LDO out
Buck-boost out
SOC Nordic nRF52811 Bluetooth 5.2, 64 MHz Cortex M4
I2C controlled for dynamical power adjustment
PMIC Maxim MAX77654 Buck-boost (SIMO Regulator), FPS, LDO cell charger
Serial Peripheral Interface
Communication possible in any direction between the 3 devices, overall control by nRF52
Power control
Flash Adesto AT25SL321 32-Mbit flash, FPGA bootcode, User storage
FPGA Lattice iCE40 Ultra Plus +5k LTUs, DSP cores, 1 Mbit SRAM, 75μA static & 1-10mA active
8 x GPIO Supporting I3C /I2C/ SPI

Figure 17: Block Diagram of the S1 module by Silicon Witchery combining Bluetooth, FPGA and power management [14]

Its ultra small form factor with a footprint of 6 x 11.5 x 1.6mm targets very small battery-powered products, like our wearable. By being designed to process edge applications that require demanding algorithms in an energy constraint space this module opens interesting development opportunities for wearables that are used in healthcare applications.[15] However, at the time of writing the module was not yet pre-qualified for EMC compliance or medical device regulations.[16]

Nordic Semiconductor nRF52840 System on Chip

To assess the nRF52840 the VedLIoT Renode Simulation Tool was used. Renode enables simulation of the wearable device, using the real, production binaries. By using the BLE-capable Renode radio model and the Zephyr biosignal monitor sample we inspected the BLE traffic between the simulated machines with Wireshark testing and CI processes with the Robot Framework. A more detailed description of our testing & simulation approach is provided in section 4.3.

[14] Based on https://docs.siliconwitchery.com/s1-module/s1-module; www.latticesemi.com/en/Products/DevelopmentBoardsAndKits/iCE40UltraPlusMobileDevPlatform, https://www.analog.com/en/products/max77654.html#product-overview & https://infocenter.nordicsemi.com/index.jsp?topic=/struct_nrf52/struct/nrf52811.html

[15] Nordic Semiconductors, News, May 2022, Nordic-powered module supports low power edge processing applications; www.nordicsemi.com/News/2022/05/Silicon-Witcherys-S1-Bluetooth-FPGA-System-On-Module-employs-Nordic-nRF52811-SoC

[16] Silicon Witchery, S1 Module, https://docs.siliconwitchery.com/s1-module/s1-module

3.4.2 TinyML models like Tensorflow Lite for Microcontrollers for over the Edge & far-Edge Devices

By introducing machine learning capabilities to microcontrollers, millions of wearable devices & medical devices could gain more intelligence without dependable internet connections, since edge devices are often limited in bandwidth, power and energy supply. TinyML models can also contribute to increased privacy as they can be set up in a way that data won't be leaving the device with a federated learning approach.

We explored TensorFlow Lite for Microcontrollers[17], working on supported Arm Cortex cores, and can be deployed locally on edge devices:

Workflow

1. First we need to train the model making sure it fits our target device and only contains supported operations (not on device).
2. We convert the model into TensorFlow Lite using TensorFlow Lite Converter
3. If the microcontroller does not support native file systems, we can use a C array and compile it into our program. After the file is generated we can directly use it in the program. Here it is advisable to change the array declaration to a const to make as efficient as possible use of the limited memory.
4. Run inference using the C++ library and process the results directly on the device.

Conversion of Models for far-Edge Devices

As microcontrollers on edge devices have very limited storage and RAM, machine learning models are very constrained in size. Thus, often only a limited subset of functions and operations are supported. In effect, not all model architectures are deployable on edge devices.

Model Architecture on far-Edge Device

While creating a model for a microcontroller, the size, intensity, and type of operations used should be taken into account.

Size of the Model for far-Edge Devices

To create an efficient model, we need to find the right balance between model size and complexity by using fewer layers that are appropriately sized to fit within our device's memory constraints. While a larger model will help to prevent underfitting, it may also be too strenuous on the processor and require too much workload. For example, the core runtime for TensorFlow Lite for Microcontrollers could fit within 16kB.

Complexity, Workload & Energy Efficiency on Edge Devices

The size and sophistication of the model directly influences the amount of workload required. If the model is very large and complex, the duty cycle of the device's processor will be greater, which implies that the device will be using more energy and emitting more heat. Within wearables & medical devices this could be a major problem as battery life time is drastically reduced.

[17] https://www.tensorflow.org/lite/microcontrollers

Conversion of Models on the example of TensorFlow

After the model is trained, we can convert a model to be embedded on microcontrollers using the TensorFlow Lite converter Python API.[18] The result is a reduced model size as it is converted into FlatBuffer and converter operations into TensorFlow Lite.

Limitations

TinyML are so programmed that they work on microcontrollers in far-edge devices. Oftentimes, when working with cores that have more performance another framework might be more suitable. Main limitations of tinyML like tensorFlow Lite include limited support for only a small number of devices, no on-device training, in case of tensorflow only a limited subset of functions and operations, and manual memory management required with low-level C++ API.

Evaluation of the Suitability of the Model for TinyML

There are several steps to consider for the evaluation of the model before converting it. First, we need to also assess if our model is suitable for use on mobile, edge or far-edge devices with respect to its data size, hardware processing requirements, and its overall complexity and size. For many models, a pre-configured converter like the TensorFlow Lite converter works without any additional steps. However, due to the subset of TensorFlow core operators that TensorFlow Lite supports, some models may require additional steps before being able to be converted to TensorFlow Lite. Additionally, some operations supported by TensorFlow Lite have certain usage requirements in order to retain performance. To ensure that our model is compatible for conversion, we check the operator compatibility guide to determine if any changes need to be made before conversion. There are three main flags: (1) Compatibility flags enable customization of operators for the conversion. (2) Optimization flags dictate the optimization technique used during conversion, such as post-training quantization (see below). (3) Metadata flags add extra information to the converted model for generating platform-specific wrapper code when deploying models on devices.

Optimising the Performance of the tinyML Model

There are multiple ways to optimize a tinyML model. One of the key optimizations are reducing the overall size of the model and/or reducing latency. A reduced size has several advantages. First it takes less storage space. This is particularly important for far-edge devices like wearables as they have limited storage. Further to update models, it takes less bandwidth to transfer or download the model on the device and thus also saves energy consumption. Further they are less idle while the update is processed. Lastly, smaller models use less RAM, which is extremely limited on edge devices. This leads to better performance and also robustness in terms of stability. Quantization can also reduce the amount of memory needed to store a model by reducing the bit-depth for storing certain parameters. While pruning and clustering can provide an exact size reduction, typically quantization will reduce the size of the model only approximately.

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

Techniques for Model Size Reduction

The size of a model can be reduced by quantization, potentially compromising some accuracy. Further reduction can be achieved through pruning and clustering, which both make the model more compressible and thus more suitable for download.

By pruning, parameters within a model can be removed that only have a minor effect on its predictions. This helps to reduce the model's download size, without changing its size on disk or its runtime latency. Therefore, pruning is a useful technique for reducing the size of a model.

Clustering groups the weights of each layer in a model into a specified number of clusters, then having the same centroid weight values. This decreases the amount of unique weights in the model, thus diminishing its complexity. Consequently, clustering allows for improved compression of the model, granting the same deployment benefits as pruning.

A major downside for both is that this can lead to a loss of accuracy, which can be a crucial drawback in healthcare.

Optimizing Latency

Another option is to optimize latency. Latency is the amount of time it takes to perform a single inference with a given model and can be reduced through optimization techniques. Quantization is currently the most popular method of reducing latency, as it simplifies the calculation process during inference, though this may come at the cost of a certain amount of accuracy. However, latency can also affect power consumption, which is a major drawback for edge devices.

The optimization of individual models can lead to changes in accuracy that are hard to foresee. Generally, models optimized for size or latency will experience a slight decrease in accuracy. Whether this has an effect on the use case or not depends on the application. On rare occasions, some models may even gain accuracy during optimization.

Quantization and Expected Model Size Reductions

The process of quantization reduces the precision of the numbers used to represent a model's parameters (which are typically 32-bit floating point numbers) to a smaller size. This leads to a reduction in model size and faster computation. Estimated size reduction can be considerable while having only minor accuracy losses.

Table 1: Quantization types with expected size and accuracy reductions

Type	Expected Reduction in Model Size (TensorFlow Lite)	Expected Reduction in Model Accuracy	Requirements
Quantization-aware Training	up to 75%	Minimal loss in accuracy	labelled data
Float16 Quantization (Post Training)	up to 50%	Minor	NA
Integer Quantization (Post Training)	up to 75%	Minor	unlabelled data but representative data sample
Dynamic range Quantization (Post training)	up to 75%	Minimal loss in accuracy	NA

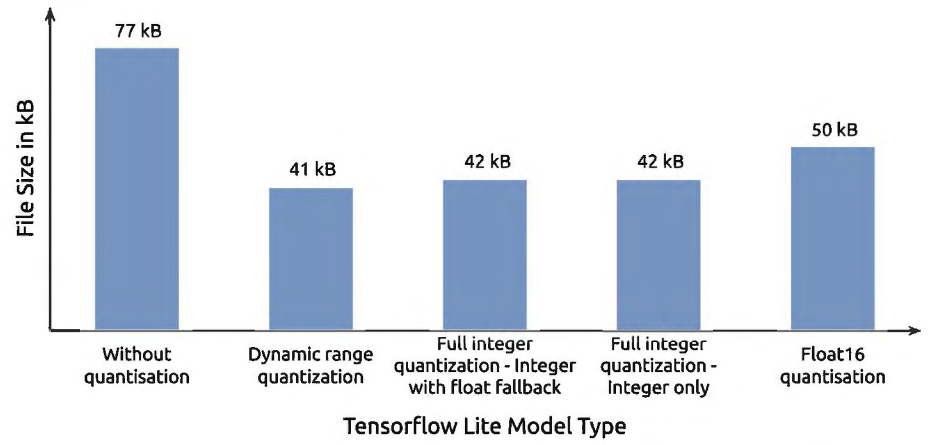


Figure 18: Comparison of quantization techniques and impact on model size. From Optimising Deep Learning at the Edge for Accurate Hourly Air Quality Prediction[19]

[19] Wardana, I.N.K.; Gardner, J.W.; Fahmy, S.A. Optimising Deep Learning at the Edge for Accurate Hourly Air Quality Prediction. Sensors 2021, 21, 1064. https://doi.org/10.3390/s21041064

4. Evaluation and Characterization

4.1 Feasibility & Comparison in the Wearable Use Case

In general there are three main alternatives available to deploy machine learning models for wearable health monitoring use cases. Depending on the specific monitoring scenario the advantages & disadvantages of each have to be evaluated.

Table 2: Overview of advantages and disadvantages by deployment type

Cloud Deployment	Edge Device Deployment	Wearable Device Deployment
+ very high computational resources + on-demand scalability + very high storage capacity + suitable for large size machine learning models	+ higher computing power compared to wearable device + lower latency compared to cloud + low power use by reduced cloud transmissions + near real-time processing	+ high privacy preservation + low classification latency + real-time feedback & alert functions + independent of connectivity
- increased costs for computing - higher latency - challenges around privacy - dependent on connectivity	- limited for large size machine learning models - limited for high training and testing data streams - dependent on connectivity to wearable device	- limited computing power - reduced battery life through always-on features - limited storage capacity

The wearable use case provides three main aspects & challenges of the system, which are oftentimes conflicting in its implementation or performance. While on-device & local edge computing allows a low latency environment, aggregated streaming of data instead of raw data on the other hand often don't allow the use of more advanced AI models & post-analysis benefits in the cloud. On-device AI models on the phones are more and more scalable as phones get better and better chips and performance attributes. Stand-alone device capabilities enhance security and reliability, which are particularly important in the healthcare environment and are less prone to errors of failed bluetooth communication.

For healthcare monitoring use cases that require instant alarms (e.g. falls or epileptic seizures) it is recommended to optimize machine learning models for a direct wearable or edge device deployment. For use cases that require inputs from multiple sensors, locations & devices a cloud deployment could offer more possibilities.

4.2 Hardware Comparison

Within the project we have compared three different SoCs developed by Nordic Semiconductors. Nordic is "specialising in designing ultra-low-power wireless communication semiconductors and supporting software for engineers developing and manufacturing IoT (Internet of Things) products."[20]

[20] Wikipedia; Nordic Semiconductor; https://en.wikipedia.org/wiki/Nordic_Semiconductor

Table 3: Comparison of widely used Nordic Semiconductor SoCs[21]

	nRF52832	NRF52840	NRF52811
Radio	Bluetooth Low Energy / Bluetooth 5.3 / Bluetooth mesh / ANT 2.4 GHz transceiver	BLE/Bluetooth mesh /Thread/ Zigbee/ANT /IEEE 802.15.4 2.4 GHz transceiver	Bluetooth Low Energy/ ANT/ IEEE 802.15.4 2.4 GHz transceiver
Processor	Arm® Cortex®-M4 32-bit processor with FPU, 64 MHz	ARM® Cortex®-M4 32-bit processor with FPU, 64 MHz	ARM® Cortex®-M4 32-bit processor, 64 MHz
Memory	512 KB flash 64 KB RAM	1 MB flash 256 kB RAM	192 kB flash 24 kB RAM
security & Crypto	AES Engine	AES Engine CryptoCell™310	AES Engine
Power management	1.7 V–3.6 V supply voltage range 0.3 µA at 3 V in System OFF mode, no RAM retention 1.9 µA at 3 V in System ON mode, no RAM retention, wake on RTC 0.7 µA at 3 V in System OFF mode with full 64 kB RAM retention	1.7 V to 5.5 V supply voltage range 0.4 µA at 3 V in System OFF mode, no RAM retention 1.5 µA at 3 V in System ON mode, no RAM retention, wake on RTC	1.7 V to 3.6 V supply voltage range 0.3 µA at 3 V in System OFF mode, no RAM retention 1.4 µA at 3 V in System ON mode, no RAM retention, wake on RTC 1.5 µA at 3 V in System ON mode, with full 24 kB RAM retention, wake on RTC
GPIO	Up to 32 general purpose I/O pins	Up to 48 general purpose I/O pins	Up to 32 general purpose I/O pins
Digital interfaces	Up to 3x SPI master and slave 1 × UARTE	4 × SPI master and slave 2 × UARTE	2 × SPI master and slave 1 × UARTE
NFC Tag	YES	YES	NO

4.3 Testing and Simulation

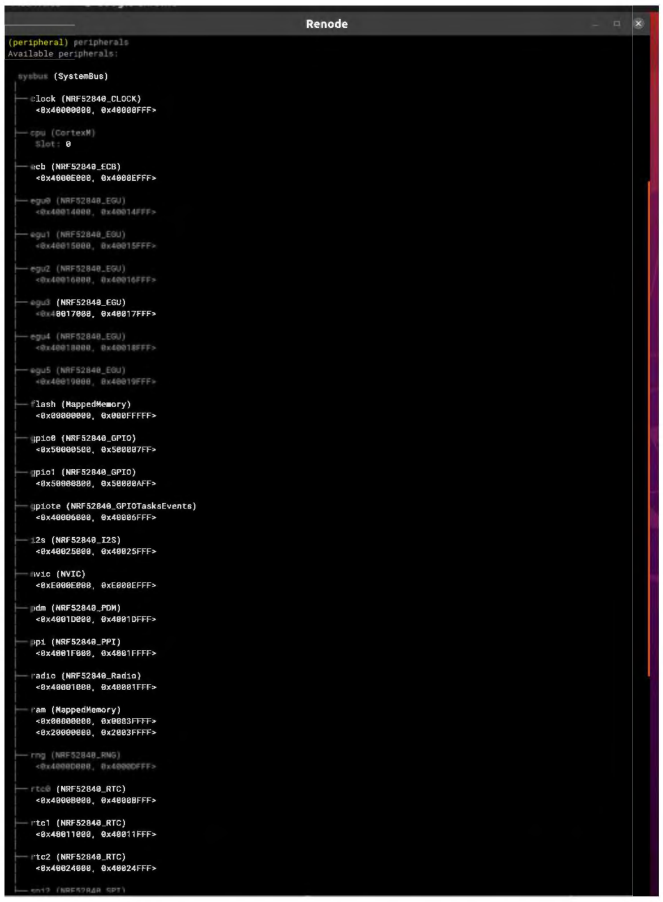
Development of multi-node test setups is a very complex & expensive process. In bluetooth low energy (BLE) solutions reliability & robustness is key and hard to implement. When working with hardware in real environments certain test scenarios cannot be fully isolated. Thus for device testing, simulation and debugging a development framework is needed. Here the Renode open source framework supported by VEDLIoT brings many advantages.

Renode enables simulation of the whole wearable device, using the real, production binaries. Thus, one can test and simulate both the hardware access layer and the more complex software running on top of systems, including end-to-end ML pipelines. One

[21] Based on infocenter.nordicsemi.com/index.jsp?topic=/ug_gsg_ses/UG/gsg/chips_and_sds.html

example scenario is to test alarms and their latency or you could test the maximum distance between phone and wearable device for reliable BLE communication.

Antmicro implemented a dedicated support for the Nordic Semiconductor nRF52840 SoC as the first platform with a BLE-capable Renode radio model.

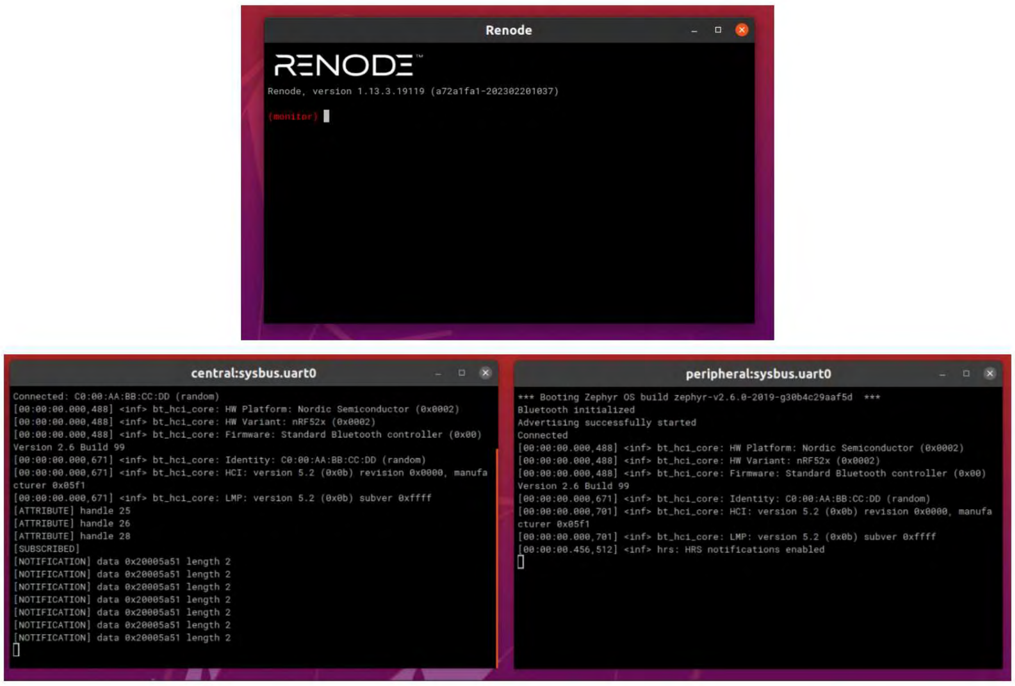


```
(peripheral) peripherals
Available peripherals:

  sysbus (SystemBus)
  │
  ├── clock (NRF52840_CLOCK)
  │     <0x40000000, 0x40000FFF>
  │
  ├── cpu (CortexM)
  │     Slot: 0
  │
  ├── ecb (NRF52840_ECB)
  │     <0x4000E000, 0x4000EFFF>
  │
  ├── egu0 (NRF52840_EGU)
  │     <0x40014000, 0x40014FFF>
  │
  ├── egu1 (NRF52840_EGU)
  │     <0x40015000, 0x40015FFF>
  │
  ├── egu2 (NRF52840_EGU)
  │     <0x40016000, 0x40016FFF>
  │
  ├── egu3 (NRF52840_EGU)
  │     <0x40017000, 0x40017FFF>
  │
  ├── egu4 (NRF52840_EGU)
  │     <0x40018000, 0x40018FFF>
  │
  ├── egu5 (NRF52840_EGU)
  │     <0x40019000, 0x40019FFF>
  │
  ├── flash (MappedMemory)
  │     <0x00000000, 0x000FFFFF>
  │
  ├── gpio0 (NRF52840_GPIO)
  │     <0x50000500, 0x500007FF>
  │
  ├── gpio1 (NRF52840_GPIO)
  │     <0x50000800, 0x50000AFF>
  │
  ├── gpiote (NRF52840_GPIOTasksEvents)
  │     <0x40006000, 0x40006FFF>
  │
  ├── i2s (NRF52840_I2S)
  │     <0x40025000, 0x40025FFF>
  │
  ├── nvic (NVIC)
  │     <0xE000E000, 0xE000EFFF>
  │
  ├── pdm (NRF52840_PDM)
  │     <0x4001D000, 0x4001DFFF>
  │
  ├── ppi (NRF52840_PPI)
  │     <0x4001F000, 0x4001FFFF>
  │
  ├── radio (NRF52840_Radio)
  │     <0x40001000, 0x40001FFF>
  │
  ├── ram (MappedMemory)
  │     <0x00800000, 0x0083FFFF>
  │     <0x20000000, 0x2003FFFF>
  │
  ├── rng (NRF52840_RNG)
  │     <0x4000D000, 0x4000DFFF>
  │
  ├── rtc0 (NRF52840_RTC)
  │     <0x4000B000, 0x4000BFFF>
  │
  ├── rtc1 (NRF52840_RTC)
  │     <0x40011000, 0x40011FFF>
  │
  ├── rtc2 (NRF52840_RTC)
  │     <0x40024000, 0x40024FFF>
  │
  └── spi2 (NRF52840_SPI)
```

Figure 19: Available peripherals in Renode with the NRF52840

In a characteristic wearable scenario like ours, the BLE architecture involves an asymmetry in which the devices are assigned with 'central' or 'peripheral' roles. Usually, initiators of the bluetooth communication are named central nodes, while the nodes to which the central node chooses to connect are peripheral nodes. The 'peripheral' devices like our wearable are able to send biosignal data to the smartphone and then enable sleep mode to save energy until the next read is performed.

We created two devices. Both machines were equipped with UART analyzers and we connected them to a single network. We sent example biosignal data for testing purposes over Bluetooth Low Energy with one device "sensing" them and a second one that reads the data and reports the reading to a console.



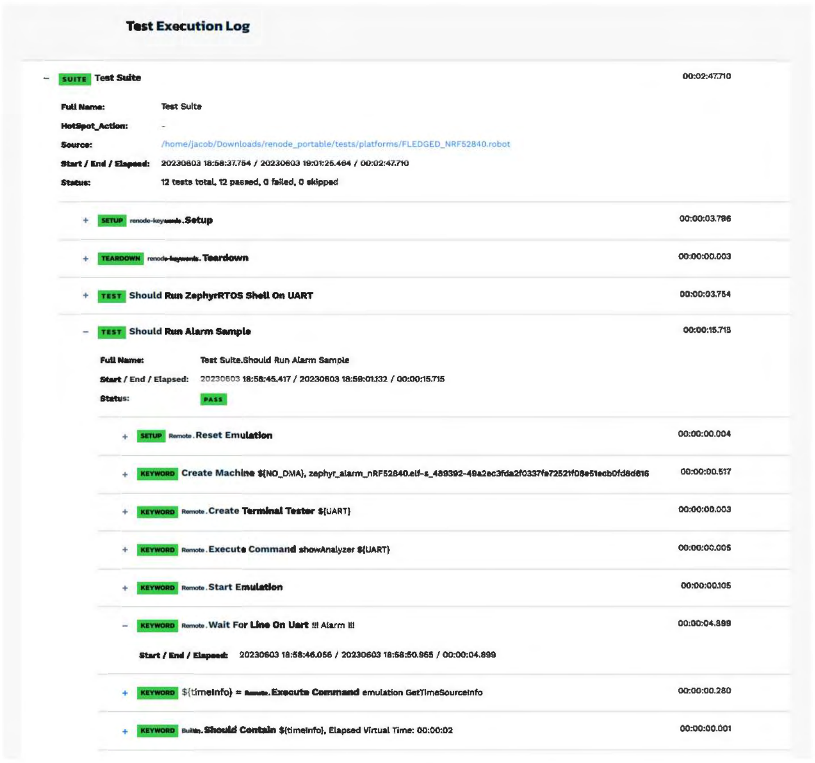
```
Renode

RENODE™
Renode, version 1.13.3.19119 (a72a1fa1-202302201037)

(monitor)
```

```
central:sysbus.uart0

Connected: C0:00:AA:BB:CC:DD (random)
[00:00:00.000,488] <inf> bt_hci_core: HW Platform: Nordic Semiconductor (0x0002)
[00:00:00.000,488] <inf> bt_hci_core: HW Variant: nRF52x (0x0002)
[00:00:00.000,488] <inf> bt_hci_core: Firmware: Standard Bluetooth controller (0x00)
Version 2.6 Build 99
[00:00:00.000,671] <inf> bt_hci_core: Identity: C0:00:AA:BB:CC:DD (random)
[00:00:00.000,671] <inf> bt_hci_core: HCI: version 5.2 (0x0b) revision 0x0000, manufa
cturer 0x05f1
[00:00:00.000,671] <inf> bt_hci_core: LMP: version 5.2 (0x0b) subver 0xffff
[ATTRIBUTE] handle 25
[ATTRIBUTE] handle 26
[ATTRIBUTE] handle 28
[SUBSCRIBED]
[NOTIFICATION] data 0x20005a51 length 2
[NOTIFICATION] data 0x20005a51 length 2
[NOTIFICATION] data 0x20005a51 length 2
[NOTIFICATION] data 0x20005a51 length 2
[NOTIFICATION] data 0x20005a51 length 2
[NOTIFICATION] data 0x20005a51 length 2
[NOTIFICATION] data 0x20005a51 length 2
```

```
peripheral:sysbus.uart0

*** Booting Zephyr OS build zephyr-v2.6.0-2019-g30b4c29aaf5d  ***
Bluetooth initialized
Advertising successfully started
Connected
[00:00:00.000,488] <inf> bt_hci_core: HW Platform: Nordic Semiconductor (0x0002)
[00:00:00.000,488] <inf> bt_hci_core: HW Variant: nRF52x (0x0002)
[00:00:00.000,488] <inf> bt_hci_core: Firmware: Standard Bluetooth controller (0x00)
Version 2.6 Build 99
[00:00:00.000,671] <inf> bt_hci_core: Identity: C0:00:AA:BB:CC:DD (random)
[00:00:00.000,701] <inf> bt_hci_core: HCI: version 5.2 (0x0b) revision 0x0000, manufa
cturer 0x05f1
[00:00:00.000,701] <inf> bt_hci_core: LMP: version 5.2 (0x0b) subver 0xffff
[00:00:00.456,512] <inf> hrs: HRS notifications enabled
```

Figure 20: Simulation on Renode with 2 devices communicating and sending biosignals over Bluetooth Low Energy

We used a test scenario based on Zephyr and analysed the BLE traffic with Wireshark between the simulated devices. This enabled us to observe the connection and log the BLE traffic. For example, we could analyse byte length of sent BLE data, used protocol type of data, write/read requests etc.



```
Test Execution Log

– SUITE Test Suite                                                              00:02:47.710
    Full Name:            Test Suite
    HotSpot_Action:       -
    Source:               /home/jacob/Downloads/renode_portable/tests/platforms/FLEDGED_NRF52840.robot
    Start / End / Elapsed: 20230603 18:58:37.754 / 20230603 19:01:25.464 / 00:02:47.710
    Status:               12 tests total, 12 passed, 0 failed, 0 skipped

  + SETUP renode-keywords.Setup                                                 00:00:03.786
  + TEARDOWN renode-keywords.Teardown                                           00:00:00.003
  + TEST Should Run ZephyrRTOS Shell On UART                                    00:00:03.754
  – TEST Should Run Alarm Sample                                                00:00:15.715
      Full Name:            Test Suite.Should Run Alarm Sample
      Start / End / Elapsed: 20230603 18:58:45.417 / 20230603 18:59:01.132 / 00:00:15.715
      Status:               PASS

    + SETUP Remote.Reset Emulation                                              00:00:00.004
    + KEYWORD Create Machine ${NO_DMA}, zephyr_alarm_nRF52840.elf-s_489392-49a2ec3fda2f0337fa72521f08e51ecb0fd8d616   00:00:00.517
    + KEYWORD Remote.Create Terminal Tester ${UART}                             00:00:00.003
    + KEYWORD Remote.Execute Command showAnalyzer ${UART}                       00:00:00.005
    + KEYWORD Remote.Start Emulation                                            00:00:00.105
    – KEYWORD Remote.Wait For Line On Uart !!! Alarm !!!                        00:00:04.899
        Start / End / Elapsed: 20230603 18:58:46.056 / 20230603 18:58:50.955 / 00:00:04.899
    + KEYWORD ${timeInfo} = Remote.Execute Command emulation GetTimeSourceInfo  00:00:00.280
    + KEYWORD BuiltIn.Should Contain ${timeInfo}, Elapsed Virtual Time: 00:00:02   00:00:00.001
```

Figure 21: Test Results

5. Exploitation

Mid-and Long-term Impact of Wearables & AI in healthcare

Wearables and AI will revolutionize healthcare, by providing valuable insights into physical activity, health, and providing support for digital health delivery (see also section 4 on impact and sustainability in the management report).

Their impacts include:

1. Improved patient monitoring: Wearables and AI are effective tools for continuous patient monitoring, helping health care providers & clinicians quickly identify potential medical problems and intervene early.

2. Improved diagnosis & health care: Wearables and AI can record and analyze a large range of data, from vital signs and movement to calorie intake and real-time information, providing health care providers with more comprehensive understanding about patients' needs.

3. Better decision-making: Doctors and nurses can use this data to make better informed decisions based on a patient's history, resulting in more effective treatments and better care.

4. More personalized care: By utilizing advanced AI and machine learning algorithms, these devices can track health trends of patients over time, enabling healthcare professionals to customize the care for them.

5. Reduced cost for healthcare systems & more efficient medical services: Wearables and AI make healthcare providers more efficient by automating tedious manual processes, thus reducing costs and improving patient care.

Sustainability & Work beyond the Project

AIoT & wearables are an important field in healthcare and are expected to grow in Europe alone with a compound annual growth rate (CAGR) of 43.9%.[22] A combination of AI and wearables is crucial for the future of healthcare, as it enables data-driven decision making in real-time. AI-driven wearables can connect to medical databases to accurately and quickly diagnose medical conditions. This information can be used to create personalized treatment plans, allowing patients to receive the best possible care. In addition, data from wearables provides insight into the current state of a patient, allowing healthcare providers to intervene more quickly if needed. With AI, healthcare providers can also gain insights into trends in European patient populations to help make better, early decisions and potentially save lives.

22 https://www.graphicalresearch.com/industry-insights/1777/europe-healthcare-artificial-intelligence-market

6. Summary and Next Steps

Our use case aimed to contribute to the VEDLIoT project by testing its feasibility for wearable devices that also operate on the 'edge of the edge' and explore important VEDLIoT concepts, design processes & architectures.

The wearable use case provides three main aspects & challenges of the system, which are oftentimes conflicting in its implementation or performance. While on-device & local edge computing allows a low latency environment, which are important for certain healthcare use cases, aggregated streaming of data instead of raw data on the other hand often don't allow the use of more advanced AI models & post-analysis benefits in the cloud. Sufficient performance for certain operations is achieved on the wearable device itself while for advanced AI models the training can be performed in the cloud and/or with a federated learning approach. On-device AI models on the phones are more and more scalable as devices get better and better chips and performance attributes. Stand-alone device capabilities enhance security and reliability, which are particularly important in the healthcare environment and are less prone to errors of failed bluetooth communication. Storage of raw data on device and synchronisation of data via bluetooth low energy allows both a federated learning approach and advanced AI models in the cloud.

A federated learning approach has several advantages as privacy, data security & data access aspects can be adapted depending on the use case. Within healthcare, federated learning offers great potential for improving the accuracy and effectiveness of medical AI models by enabling different clinical stakeholders and hospitals to work together and share data in a privacy-preserving manner. Thus, by training advanced AI models on distributed datasets from different medical institutions, federated learning is able to help with some of the challenges of personalized medicine.

Novel digital concepts and particularly health monitoring will play a major role in our lives. The FLEDGED use case is scalable and can be broadened to other healthcare use cases. In terms of effectiveness, AI and wearables can improve healthcare outcomes by providing accurate medical care and real-time disease monitoring. AI technology can be used to quickly diagnose or suggest treatments for illnesses, analyze and interpret data more accurately than humans, and synthesize information to more quickly draw conclusions. Similarly, wearables such as medical monitors and smart wearables can be used to capture more accurate medical data and provide real-time updates to healthcare providers. In terms of costs & sustainability, AI and wearables can be used to reduce and manage healthcare costs by providing affordable and accessible healthcare solutions to patients and healthcare systems. AI can be used to automate labour-intensive tasks and reduce administrative costs, while wearables can provide real-time medical data to doctors and other staff, reduce time-consuming medical consultations, and improve diagnosis accuracy, thus reducing unnecessary medical treatments and tests.

Further the FLEDGED use case can be transferred to other areas beyond healthcare, where remote monitoring with alarm functions in low energy environments are important. This can be from Smart Cities, Agritech or Environmental Monitoring.

Very Efficient Deep Learning in IoT

ICT-56-2020 — Next Generation Internet of Things

FINAL TECHNICAL REPORT

AccBD VEDLIoT Open

Version 1.0

Document Information	
Contract Number	957197
Project Website	`https://vedliot.eu/`
Dissemination Level	PU (Public)
Nature	R (Report)
Contractual Deadline	31 July 2023
Author	Sigrun May, Frank Klawonn
Contributors	Jerome Brenig, Daria Kober
Reviewers	René Griessl

The VEDLIoT project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No 957197.

Change Log		
Version	**Date**	**Description of Change**
0.1	2023-06-19	Initial structure
1.0	2023-07-31	Final version

Table of Contents

List of Figures

List of Tables

1 Executive summary

Data analysis for modern medical research with a very small sample size and a large number of features faces specific challenges. One of them is related to the curse of dimensionality. In addition, skewed data distributions and outliers are common in these biological data sets. Algorithms designed specifically for this use case are computationally intensive. The AccBD project successfully accelerated a feature selection workflow by implementing an ensemble of embedded feature selection methods, optimized for parallelization on the VEDLIoT hardware platform. The achieved results were remarkable, with an acceleration factor of 13.9X and an energy saving of 72%. Although, including model training on GPU is theoretically possible it currently suffers from a memory leak. Further investigation and resolution of the memory leak issue are required to fully exploit the potential of GPU acceleration in this context.
Furthermore, the Yeo-Johnson transformation, an integral part of the data preprocessing phase, was implemented as an optimized C-library as well as an FPGA implementation. The C-library version demonstrated a runtime reduction of 88.27% compared to the Python `scikit-learn` library, accompanied by an energy reduction of 89.5%. The FPGA version, consisting of a single module, exhibited a runtime reduction of 37.7% and an energy reduction of 91.9% when compared to the C-library version. In the case of multiple modules (five, in our scenario), the runtime reduction achieved was 81.4%, with a further energy reduction of 95.7% compared to the C-library version.
Overall, this project demonstrated the feasibility of optimizing parallelization and using alternative hardware solutions to save time and significantly reduce energy consumption in the field of medical research.

2 Introduction

This introduction provides background information on the bioinformatics challenge and discusses its connection to the Internet of Things (IoT). It introduces the concept of ensemble feature selection and highlights the motivation behind the research. Finally, the introduction concludes with an outline of the report's structure.

2.1 Bioinformatics Background

In recent years, the field of bioinformatics has witnessed significant advancements with the increasing availability of high-dimensional biological data sets. These data sets contain valuable information that can be harnessed to gain insights into biological processes and facilitate discoveries in various domains, including drug development and personalized medicine. So-called biomarkers are measurable indicators for medical risk factors, for a biological condition, to study a disease, to predict a diagnosis, to determine the state of a disease or the effectiveness of a treatment. To obtain potential biomarker candidates (features), multiple sources are possible. In addition to classical laboratory measurements, social-demographic values and imaging data, modern high-throughput technologies like mass spectrometry or next-generation sequencing provide large amounts of additional measurements. In "omics" fields like proteomics, metabolomics and transcriptomics, thousands of proteins, genes, and metabolites can be measured. Although such measurements provide valuable information for diagnostic and prognostic purposes, they usually lead to high-dimensional data with very small sample size. The small sample sizes result from the fact that sample processing for high-throughput technologies can be laborious and expensive. A robust selection method for biomarker candidates could lead to cheaper clinical tests since fewer markers would have to be tested [19]. For example, biomarker pilot studies may include well over 10,000 biomarker candidates (features) but fewer than 100 samples. The sheer volume and complexity of these data sets pose a challenge for traditional data analysis techniques. Advanced analysis of such high-dimensional data is computationally expensive.

2.2 IoT

Identifying potential biomarkers is a crucial aspect of advancing smart health applications. The experience of the COVID-19 pandemic has underscored the value of monitoring pathogens in wastewater to contain outbreaks effectively. The primary objective of the AccBD project is to facilitate real-time monitoring of biomarker candidates, enhancing our ability to make informed decisions during future pandemics and aiding in the analysis of more prevalent diseases such as influenza. To achieve this, an extensive network of monitoring points needs to be strategically positioned within the wastewater system to ensure data reliability. Traditional cloud-based processing results in substantial data transfers and delays, making local data processing at the monitoring points a more suitable approach. In our evaluation of IoT computing, we utilized VEDLIoT hardware, specifically the edge server, demonstrating that complex algorithms can be accelerated, particularly through the use of FPGA technology. The VEDLIoT project offers a comprehensive range of heterogeneous computing platforms, spanning from the cloud to IoT. In forthcoming efforts, we will extend the acceleration demonstrated in AccBD to these IoT devices, enabling on-site biomarker candidate calculations at the monitoring points.

2.3 Feature Selection

In our particular context, features are potential biomarker candidates. In general, feature selection plays a crucial role in enhancing model performance, eliminating redundant features, and constructing simpler and more efficient models [55, 36]. Unlike feature extraction, which involves transforming data into a new data space, feature selection focuses on selecting the optimal subset of features without changing the data [54]. In our bioinformatics domain, the objective is to eliminate irrelevant features. Due to the statistically insufficient sample size, it is almost impossible to overcome the curse of dimensionality [54] and select a final, clinically usable feature subset.
Feature selection can be categorized into three basic types:

1. **Filter methods:** Filter methods operate directly on the data set, providing fast and classifier-independent feature selection [20]. They encompass techniques such as correlation-based feature selection (CFS), information gain, chi-squared, entropy, symmetrical uncertainty (univariate), and the RELIEF algorithm (multivariate) [39]. Classical filter methods are CFS, FCBF, the INTERACT algorithm, Information Gain, ReliefF, and mRMR (minimum Redundancy Maximum Relevance). Although filter methods tend to yield inferior results compared to other methods, they are often employed for feature selection on high-dimensional,

low sample size data such as microarray data [26].

2. **Wrapper methods:** Wrapper methods involve greedy search algorithms that explore the space of attribute subsets to find the best performance [20]. Wrapper methods generally outperform filter methods but can be computationally intensive. For a complete ranking of N features, they require N^2 classification models [55]. However, due to their high computational complexity and the risk of overfitting, wrapper methods are not suitable for the data sets considered in this project [55, 26].

3. **Embedded methods:** Embedded feature selection methods utilize algorithms that inherently incorporate feature selection. They combine the benefits of both filter and wrapper methods. These methods consider attribute correlations (like wrapper methods) and are faster (like filter methods), resulting in improved accuracy compared to filter methods [20]. Examples of embedded feature selection methods include LASSO Regression, Ridge Regression, Random Forest, and Extra Trees [56, 35, 34, 28, 33]. Random Forest is a popular embedded feature selection method for high-dimensional data sets with low sample size, as it can handle missing data and unbalanced classes effectively [26, 22]. In Random Forest, feature importance is measured by randomly permuting the feature within the out-of-bag samples and calculating the percentage increase in misclassification rate compared to the rate with all variables intact [55]. Another well-known embedded method suitable for feature selection in data sets with a higher number of features than samples is LASSO Regression[61]. For these machine learning models, highly correlated features can have a negative impact on the analysis, and therefore, LASSO Regression ($\alpha = 1$) is preferred over Ridge Regression ($\alpha = 0$) in the presence of many irrelevant attributes.

2.4 Ensemble Feature Selection

In the bioinformatics domain, the feature selection performance is as important as the selection robustness, as robustness can enhance the confidence of domain experts [55, 38]. Enhanced robustness can be achieved by ensemble feature selection, which aims to aggregate the outputs of multiple feature selection methods for improved results [26]. To aggregate the results of feature selection methods, various aggregation methods are available. Examples of basic data aggregation methods are Sum, Arithmetic Mean, Median, Min, Max, Count, or Weighted Averaging:

- **Sum**: sums the soft labels of the feature selection methods. The highest S features are kept. [27]
- **Arithmetic Mean**: the arithmetic mean of the soft labels of the feature selection methods is calculated. The S highest features are kept. [43]
- **Median**: the median of the soft labels of the feature selection methods is calculated. The S highest attributes are kept. [27]
- **Min**: the output of the aggregation is the minimum value the feature got by the feature selection methods. [27]
- **Max**: the output of the aggregation is the maximum value the feature got by the feature selection methods. [27]

- **Count**: creates a $n \times m$-matrix T with n : number of attributes and m : number of feature selection methods. The matrix is filled as follows:

  $$t_{ij} = \begin{cases} 1 & \textit{if attribute i has a soft label bigger than 0} \\ & \textit{from feature selection method j} \\ 0 & \textit{otherwise} \end{cases}$$

  Afterwards, the features are ranked by summing the rows of the matrix T. The highest S features are kept.

- **Weighted Averaging**: in this method the soft labels of the feature selection methods are averaged by weighting them according to the accuracy of the predictive model. The weights have to be normalized so that their sum is equal to 1. [43]

Voting aggregation methods include Borda Count, Reciprocal Ranking, Feature Occurrence Frequency, Instant Runoff Voting, Coombs Voting and Bucklin Voting [32, 24, 43]:

- **Borda Count**: sums the ranks of a feature over all feature selection methods:

  $$r(f) = \sum_{j=1}^{N} r_j(f)$$

  where f : feature, $r_j(f)$: rank of feature f according to method j. The lowest S features are kept. [32, 25]

- **Reciprocal Ranking**: aggregates the rankings of the single feature selection methods based on the equation

  $$r(f) = \frac{1}{\sum_j \frac{1}{r_j(f)}}$$

  with f : feature, $r_j(f)$: rank of feature f according to method j. The lowest S features are kept. [32]

  Even though Reciprocal Ranking is not commonly used for feature selection, 2021 Effrosynidis and Arampatzis showed that it can lead to better performance in contrast to other methods and has a high stability. [32]

- **Feature Occurrence Frequency**: creates a $n \times m$-matrix T with n : number of attributes and m : number of feature selection methods. The matrix is filled as follows:

  $$t_{ij} = \begin{cases} 1, & \textit{if attribute i has been selected by classifier j} \\ 0 & \textit{otherwise} \end{cases}$$

  Afterwards, the features are ranked by their occurrence frequency by summing the rows of the matrix T. The highest S features are kept. [24]

- **Instant Runoff Voting**: is a voting method for elections. In the case of feature selection the features represent the candidates and the feature selection methods represent the voters. If one feature reaches over 50 % of the votes it is the winner of the election. As long as there is no such feature the feature with the least amount of first votes is removed and replaced by the feature that is next preferred until one feature gets more than 50 % of the votes. To get a ranking using this method, the winner of each iteration gets the corresponding rank and is removed from the data set the votings are based on. [32]

- **Coombs Voting**: is an election voting method closely resembling Instant Runoff Voting. However, unlike the latter, Coombs Voting does not eliminate the feature with the fewest first votes when no candidate or feature in the case of feature selection surpasses 50% of the votes. Instead, it eliminates the feature with the highest number of last-place votes. [32]
- **Bucklin Voting**: is a voting method for elections as well. In this method if there is no candidate (feature) that received the majority by summing the first choice votes, the votes of the next preference are added and so forth until one candidate holds the majority. [32]

2.5 Motivation and Outline

While ensemble feature selection is employed to achieve better and more stable results, it also has disadvantages. The interpretability of ensemble results is more challenging compared to single feature selection methods, and incorrect method selection can potentially yield worse results. However, the main disadvantage is the computational cost. Ensemble feature selection in high-dimensional data to reduce the feature dimension can be time-consuming or even unmanageable in a reasonable time without acceleration. But as its advantages are desirable [44] we aim to massively accelerate the ensemble feature selection workflow for high-dimensional data with tiny sample size. Additionally, we intend to build a framework to explore the different aggregation methods in order to find the most promising option for the given use case.
In this report, we especially focus on optimizing the Yeo-Johnson transformation as an integral part of our data preprocessing pipeline. Starting with an initial implementation in Python, we go on to develop a high-performance C library and an accelerated version using Field-Programmable Gate Arrays (FPGAs). The feature selection workflow described in Chapter 3 serves as the foundation for our research and subsequent optimization efforts. In Chapter 4 we discuss the mathematical foundations of the Yeo-Johnson transformation and its role in normalizing skewed data distributions. Subsequently, we describe a novel C library and an implementation on Field-Programmable Gate Arrays (FPGAs) of the Yeo-Johnson transformation algorithm. We evaluate and characterize its performance comparing it to other implementations and present our experimental results. After that, we present the acceleration of the bioinformatics workflow through parallelization and optimization techniques in Chapter 5. We discuss the implementation details and highlight the performance and efficiency improvements achieved. In Chapter 6, we explore the potential exploitation of our research findings. We conclude the report by summarizing the key findings of our research. Additionally, we outline the next steps for further exploration and optimization.

3 Feature Selection Workflow

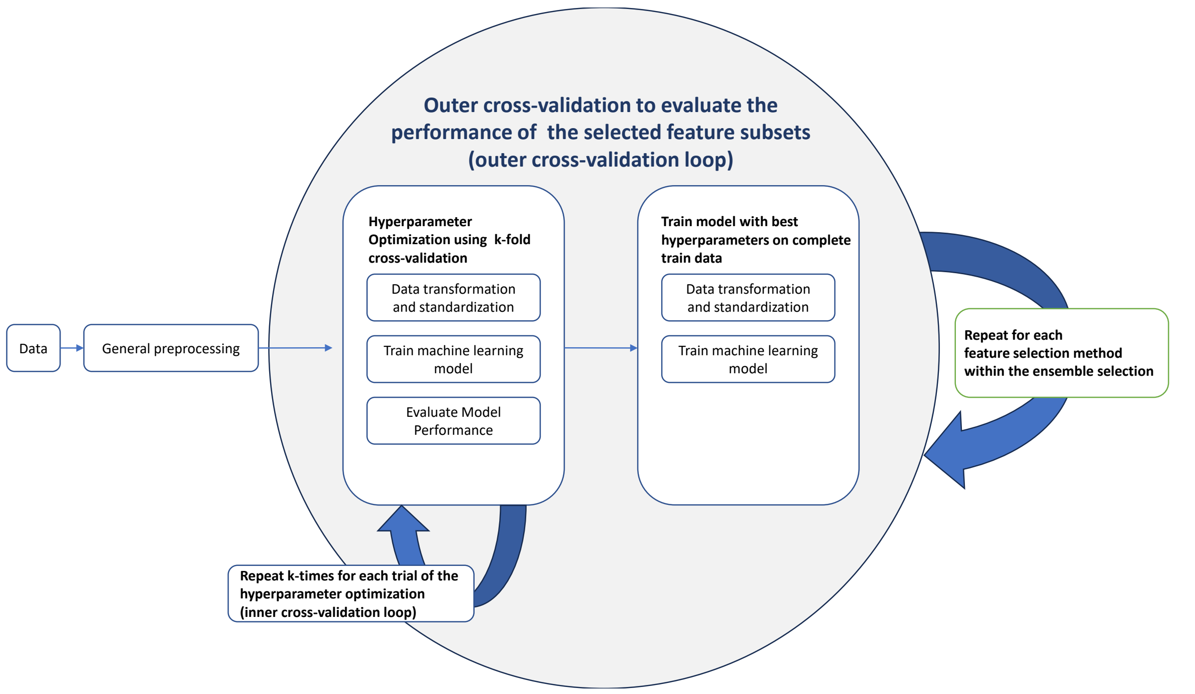

Outer cross-validation to evaluate the performance of the selected feature subsets (outer cross-validation loop)
Data
General preprocessing
Hyperparameter Optimization using k-fold cross-validation
Data transformation and standardization
Train machine learning model
Evaluate Model Performance
Train model with best hyperparameters on complete train data
Data transformation and standardization
Train machine learning model
Repeat for each feature selection method within the ensemble selection
Repeat k-times for each trial of the hyperparameter optimization (inner cross-validation loop)

Figure 3.1. Feature Selection Workflow

One approach to achieve feature dimension reduction with ensemble feature selection is applying an ensemble of embedded feature selection methods [42, 57, 53]. Embedded feature selection algorithms have built-in feature selection techniques performed during model building. The effectiveness of a model in fitting the data is commonly evaluated using performance evaluation metrics. For cases with small sample sizes, cross-validation is often employed to ensure that the trained model does not overfit the training data. Specifically, k-fold cross-validation divides the training data set into k equal parts, where each part is subsequently used as a validation/testing set. For instance, in the case of 5-fold (k = 5) cross-validation, the data set is split into five equivalent segments. The model is trained on four of these segments while the performance is assessed on the remaining one. This process is repeated five times, ensuring that each segment serves as the testing set at least once. Finally, the average performance of the model across all testing sets is calculated [52].
However, when only a limited number of samples is available, k-fold cross-validation can yield strongly biased performance estimates [58, 29]. To address this issue, Vabalas et al. [58] propose the use of nested cross-validation, which provides robust and unbiased performance estimates regardless of the sample size. Nested cross-validation consists of an outer and an inner loop, resulting in a total number of outer folds multiplied by the number of inner folds ($folds_{\text{outer-cv}} \times folds_{\text{inner-cv}}$) steps. In each step, a new model is built. However, as the number of features increases, the runtime of the model building process also increases. Consequently, constructing a large number of models with a high feature dimension leads to long computation times.
To ensure that each machine learning model is well-suited for its specific task, the individual hyperparameters of the model must be tuned. The choice of tuned hyperparameters can directly impact the model's performance [62]. In order to reduce overfitting, it is advisable to regularize the model building process by adjusting the corresponding hyperparameters [50]. Considering the hyperparameter dimensions

that can be optimized, the search space can be large, making it challenging to evaluate every configuration. Consequently, finding an optimal combination of hyperparameter values often requires many trials. Additionally, time-consuming analyses, such as computing SHAP values [41] to determine feature importances [45] based on the trained models, may be necessary. [47]
Parallelization can significantly accelerate the standard workflow (see Figure 3.1). However, in the standard workflow using libraries such as `scikit-learn` [51], `LightGBM` [37] or `Optuna` [21], only the model training process or the hyperparameter optimization is parallelized. In scenarios where a large number of models are involved, these parallelization layers might not be the most efficient ones. Generally following parallelizations are possible for the given workflow as illustrated in Figure 3.2:

1. Outer cross-validation
2. Yeo-Johnson transformation
3. Calculation of a correlation matrix
4. Individual embedded feature selection methods of the ensemble selection
5. Hyperparameter optimization
6. Inner cross-validation
7. Model Training

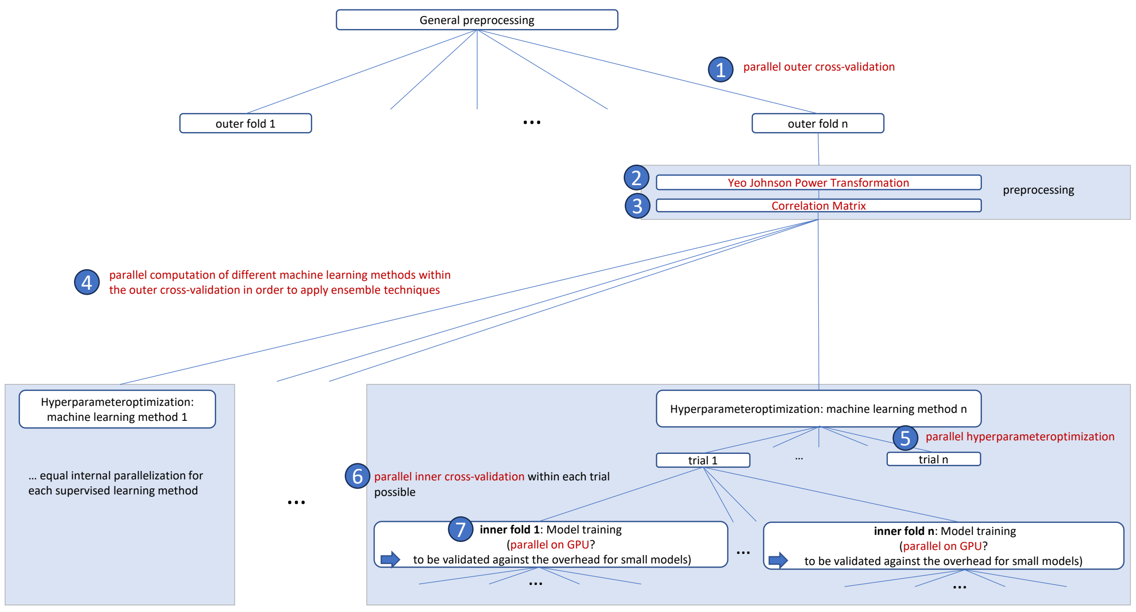

General preprocessing
1 parallel outer cross-validation
outer fold 1
...
outer fold n
2 Yeo Johnson Power Transformation
3 Correlation Matrix
preprocessing
4 parallel computation of different machine learning methods within the outer cross-validation in order to apply ensemble techniques
Hyperparameteroptimization: machine learning method 1
... equal internal parallelization for each supervised learning method
...
Hyperparameteroptimization: machine learning method n
5 parallel hyperparameteroptimization
trial 1
...
trial n
6 parallel inner cross-validation within each trial possible
7 inner fold 1: Model training (parallel on GPU? to be validated against the overhead for small models)
...
inner fold n: Model training (parallel on GPU? to be validated against the overhead for small models)
...
...

Figure 3.2. Parallelization options

Due to the constraints of limited hardware resources, it is not useful to parallelize all options simultaneously. By strategically clustering tasks into larger units, it can be possible to minimize overhead and optimize resource utilization. In this report, we will discuss the various options for parallelization mentioned above.

The first option, parallel outer cross-validation, does not provide significant benefits when implemented as a standalone parallelization technique due to the underutilization of hardware resources. In our test scenario we have opted for a 6-fold outer cross-validation. Consequently, this approach would only involve running 6 parallel processes. Thus, integrating this option into a nested parallelization framework could be more meaningful.
The second option, parallelizing the Yeo-Johnson power transformation as part of the preprocessing, will be extensively discussed in the following Chapters. Chapter 5 will then evaluate all parallelization options and present the best solution.

4 WP1: Yeo-Johnson power transformation on FPGA

Biological data typically includes outliers [30]. Given that outliers can significantly influence LASSO Regression [46] as one of the feature selection methods, data transformation is an essential step in the preprocessing pipeline.

4.1 Yeo-Johnson Transformation

The Yeo-Johnson transformation is a power transformation used to improve the normality or symmetry of a distribution [63]. Power transformations are a group of functions that allow for a monotonic transformation and approximate a symmetric distribution. The Yeo-Johnson transformation is particularly notable because, unlike other power transformations, it can also be applied to negative values. It is based on the Box-Cox transformation [31], which applies the transformation parameter λ to the strictly positive values of the distribution Y_t. The parameter c is usually 0 but can be varied to shift negative values within a distribution into a strictly positive range. The formula for the Box-Cox transformation is given by:

$$Y_t^{(\lambda)} = \begin{cases} \frac{(Y_t+c)^\lambda - 1}{\lambda} & \lambda \neq 0 \\ \ln(Y_t + c) & \lambda = 0 \end{cases}$$

Figure 4.1 shows the Box-Cox transformation, where the green axis represents the parameter λ, the red axis represents the input value Y_t, and the blue axis represents the result $Y_t^{(\lambda)}$. For $\lambda = 1$, the distribution is mapped to the identity minus the constant 1. For $\lambda = 0$, the logarithmic function is applied. For negative λ, values $Y_t > 1$ are shrunk but remain positive. Values with $0 \leq Y_t < 1$ and $\lambda \to -\infty$ are mapped to increasingly smaller negative values. As λ increases and Y_t rises, a monotonically increasing behavior can be observed.

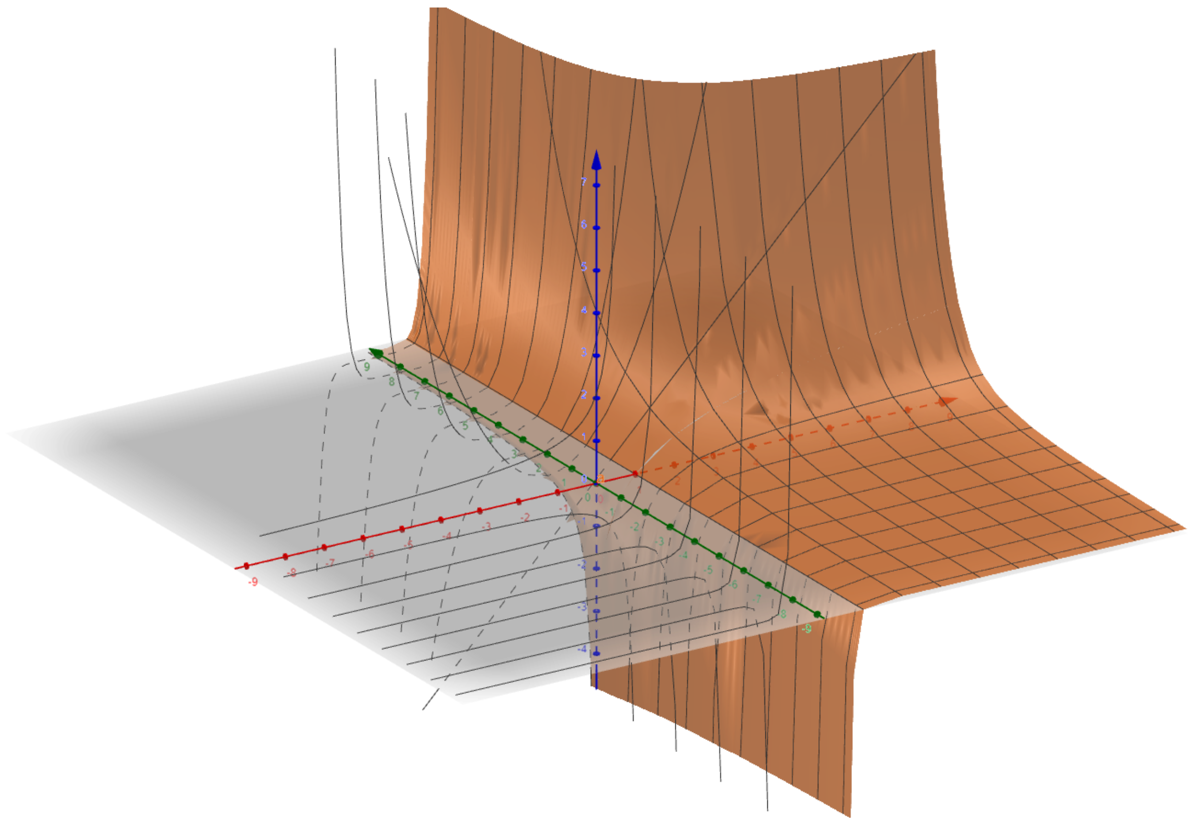


Figure 4.1. Box-Cox Transformation

The formula for the Yeo-Johnson transformation is as follows:

$$Y_t^{(\lambda)} = \begin{cases} (Y_t+1)^{\lambda} - 1/\lambda & \lambda \neq 0, Y_t \geq 0 \\ \ln(Y_t+1) & \lambda = 0, Y_t \geq 0 \\ -((-Y_t+1)^{(2-\lambda)} - 1)/(2-\lambda) & \lambda \neq 2, Y_t < 0 \\ -\ln(-Y_t+1) & \lambda = 2, Y_t < 0 \end{cases}$$

Figure 4.2 illustrates the behavior of the Yeo-Johnson transformation for λ on the green axis, Y_t on the red axis, and the resulting $Y_t^{(\lambda)}$ values on the blue axis. Similar to the Box-Cox transformation, a linear relationship is observed for $\lambda = 1$. However, in the case of the Yeo-Johnson transformation, this linear relationship corresponds to the identity of the input distribution, thus mapping to the identity function for $\lambda = 1$. For $\lambda = 0$, the Yeo-Johnson transformation, like the Box-Cox transformation, corresponds to the logarithmic function. The monotonically increasing behavior of the Box-Cox transformation for increasing λ and Y_t values is preserved in the Yeo-Johnson transformation but with an additional reflection across the angle bisector of the green and red axes for negative values. The logarithm complements the undefined regions for $Y_t \geq 0, \lambda = 0$, and $Y_t < 0, \lambda = 2$. Therefore, the Yeo-Johnson transformation is continuous throughout $\mathbb{R}$. Unlike the Box-Cox transformation, the Yeo-Johnson transformation does not employ a variable parameter c, as there is no need to shift the spectrum. The fixed parameter value is set to 1.

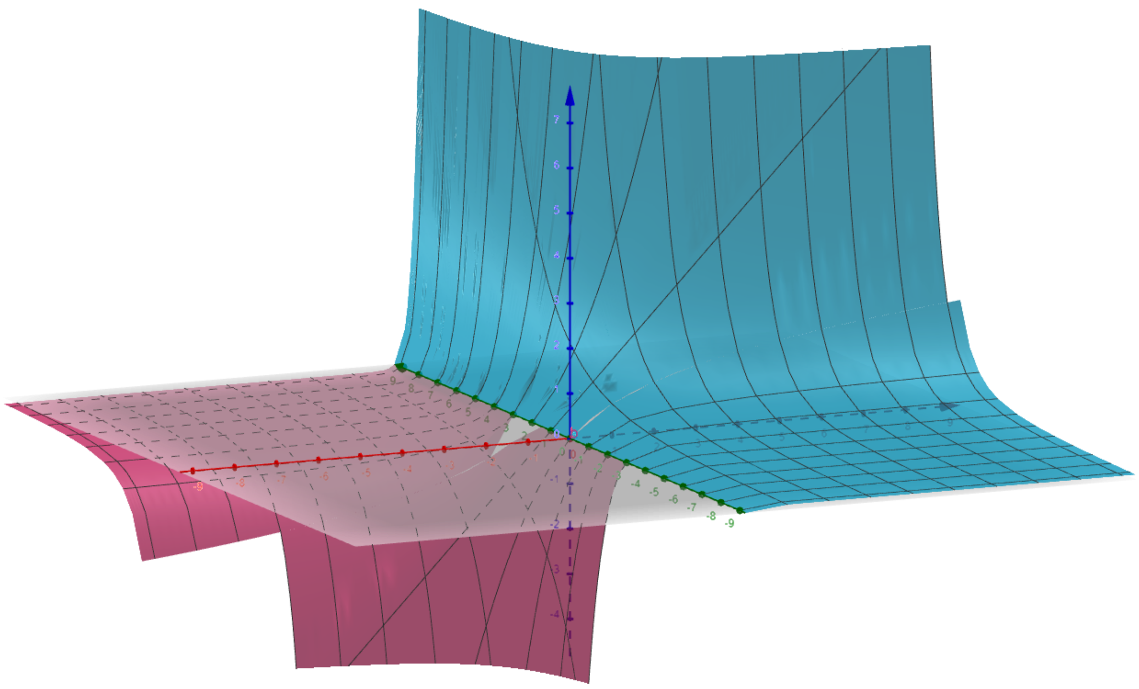


Figure 4.2. Yeo-Johnson

Since a λ is needed to find a transformation for a vector y_i, a search for an optimal λ is necessary to optimally influence the skewness of the distribution. Skewness is a statistical parameter that indicates the deviation from a symmetric distribution.

4.2 Skewness of a Distribution

The skewness of a distribution provides information about its asymmetry. A skewness value of zero indicates symmetry and therefore a normal distribution.

To calculate the skewness, both the mean and the standard deviation are needed. They are calculated as follows:

$$mean = \frac{1}{n} * \sum_{n=1}^{N}(x_n)$$

$$variance = \frac{1}{n} * \sum_{n=1}^{N}(x_n - mean)^2$$

$$standarddeviation = \sqrt{variance}$$

The formula for calculating the skewness of a distribution is as follows:

$$skewness = \frac{1}{n} * \sum_{n=1}^{N}(\frac{x_n - mean}{standarddeviation})^3$$

A distribution is considered positively skewed if its skewness is greater than zero. Conversely, a negatively skewed distribution is identified by a skewness smaller than zero. There is no fixed threshold to identify a distribution as approximately symmetrical. We define a range of $[-0.5, 0.5]$ as approximately symmetrical. However, a skewness close to the optimum of zero is aimed and used as a basis for comparison.

4.3 Standard Python Library: Scikit-learn - PowerTransformer

The Python library `sklearn.preprocessing.PowerTransformer` [51] provides the capability to apply both the Box-Cox and Yeo-Johnson transformation to a distribution. The ”optimal” λ values are estimated using the maximum likelihood method. The term ”optimal” is not further defined within this library, so it is assumed here to refer to the skewness as the parameter of interest. The maximum likelihood method determines the value that maximizes the probability of influencing the skewness of the distribution as effectively as possible.

By comparing Figure 4.3 and Figure 4.12, an improvement in the symmetry of the distributions can be observed. This is also confirmed by the skewness in Figure 4.5, as the majority of the skewness now lies within the range of $[-0.2, 0.2]$. Therefore, most distributions can be considered approximately symmetric after applying the `sklearn.preprocessing.PowerTransformer` library.

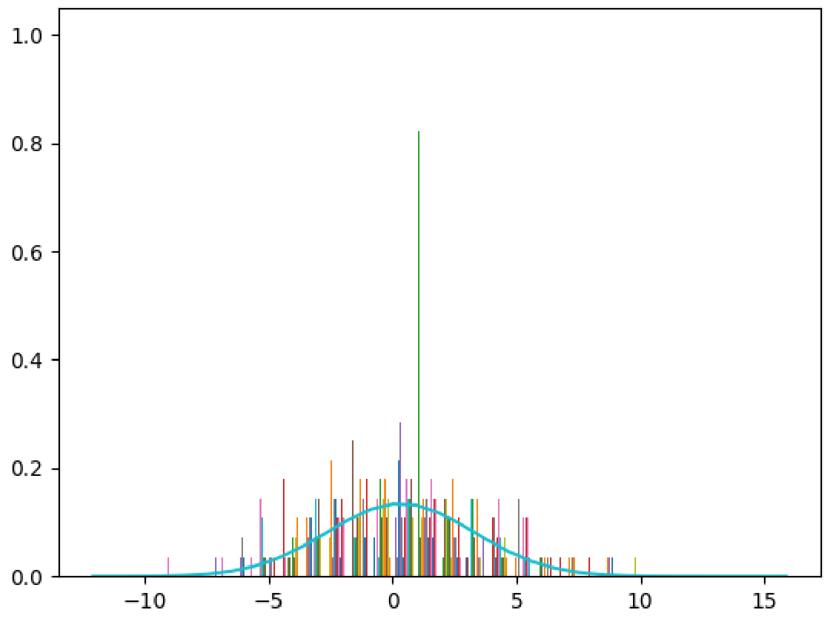


Figure 4.3. Data - Transformed with Python

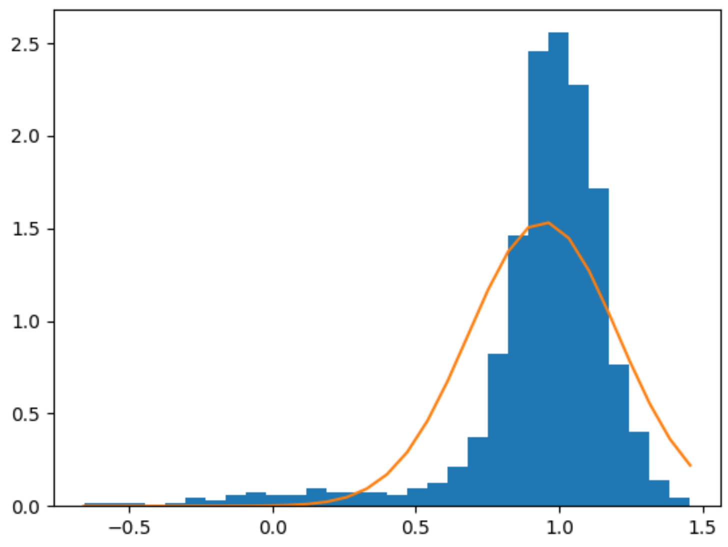


Figure 4.4. λ - Python

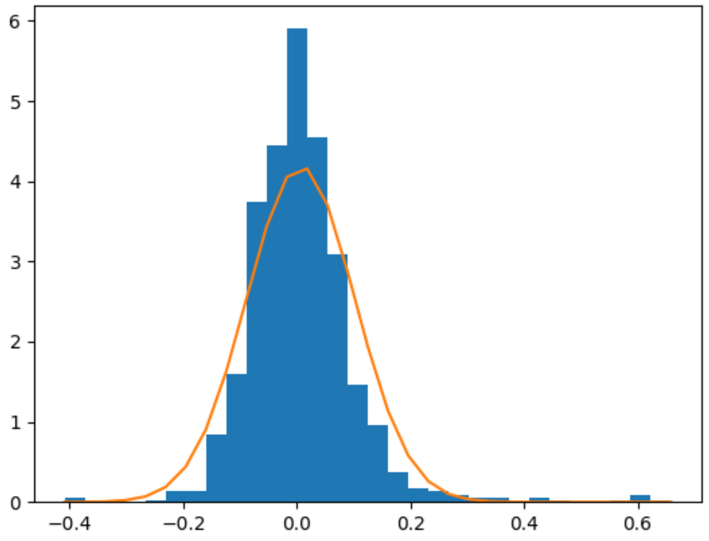


Figure 4.5. Skewness - Python

One major advantage of this library is its user-friendliness and stable behavior in the presence of outliers, as the best transformation is automatically selected for them. A disadvantage is the runtime of the function. The functionality, except for certain subcomponents of the used libraries, is not parallelized.

4.4 C Library: Task 2

To accelerate the Yeo-Johnson transformation, the initial stage of optimization involves the development of a novel C library providing an alternative algorithm. Diverging from the Python library, this novel implementation does not apply the maximum likelihood method. Instead, it iterates over a fixed range and applies a data transformation for each value. Subsequently, it determines the skewness of the transformed data and chooses the transformation parameter that achieves the best skewness value. The library provides three variations to execute this procedure.

The first variation we implemented iterates over a fixed range with a defined step size. Since there is no termination criterion to end the search prematurely, a fixed number of iterations is required for each distribution. This number is derived from the difference between the upper and lower bounds of the range and the step size ($\frac{\textit{upper\_bound}-\textit{lower\_bound}}{\textit{step size}}$). In this case, the sampling rate, i.e., the spacing between selected λ values, would significantly impact both the runtime and accuracy. For instance, a range of $\lambda = [-3, 3]$ with a sampling rate of $i = 0.001$ would require 6,000 cycles to find the optimal λ within the range. This is due to the fact that it cannot be inferred whether a decrease in skewness towards zero indicates a reduction in the distance to the "optimal" λ, as there is currently no knowledge regarding the behavior of skewness over such a range of values.
Furthermore, floating-point and double data types, which are represented as exponential values with a base and an exponent, introduce rounding errors through arithmetic operations. This directly affects the accuracy of the results. However, this problem can be circumvented by only using λ values that are powers of 2.

4.5 Optimized Algorithm: Task 4

To reduce the search time, we implemented an alternative approach, which no longer samples the entire interval. Initially, the interval is sampled in a broad range, and then the search continues within a decreasingly smaller range. In each iteration, the search range is divided into n values, and the step size is halved. The upper and lower bounds are adjusted to be smaller/larger than the λ value obtained within that iteration. The upper bound is shifted according to the following formula:

$$\textit{intermediate}\ \lambda + \textit{step\_size}$$

while the lower bound is shifted as follows:

$$\textit{intermediate}\ \lambda - \textit{step\_size}.$$

Depending on a starting λ, which is initially set to zero, n values are generated, and the transformed values are computed using the Yeo-Johnson transformation. Then, the skewness is determined, and the λ value that yields better skewness becomes the starting λ for the next iteration. This approach gradually reduces the search range and approaches the optimal λ value. Consequently, the number of iterations over the search range can be reduced. This second variation requires the initial definition of a precision variable that indicates how many times the algorithm should be repeated.

A third implementation extends this functionality by utilizing threads, enabling the simultaneous processing of multiple vectors in a matrix. Since a longer list of vectors is passed, rather than a single vector with multiple values, this is the more efficient step for parallelization compared to parallelizing over the values of a single vector. The user can determine the number of parallel threads.

Additionally, the option to adjust the transformed data using z-standardization is provided. This means that the calculated transformation values are arranged around zero. Furthermore, a parameter is implemented to measure the time it takes to perform the transformation for result verification.

Figure 4.6 shows that the transformed data using the C library is less aggregated compared to the Python library (see Figure 4.3). This is evident in the significantly smaller range of the data ($[-3, 3]$ for the C library versus $[-10, 10]$ for the Python library). Moreover, the skewness values of the C library (see Figure 4.8) exhibit, on average, significantly less deviation from zero compared to those of the Python library (see Figure 4.5). While the skewness values of the Python library range from $[-0.2, 0.2]$, the skewness values of the C library, without outliers, fall within the range of $[-0.002, 0.002]$.

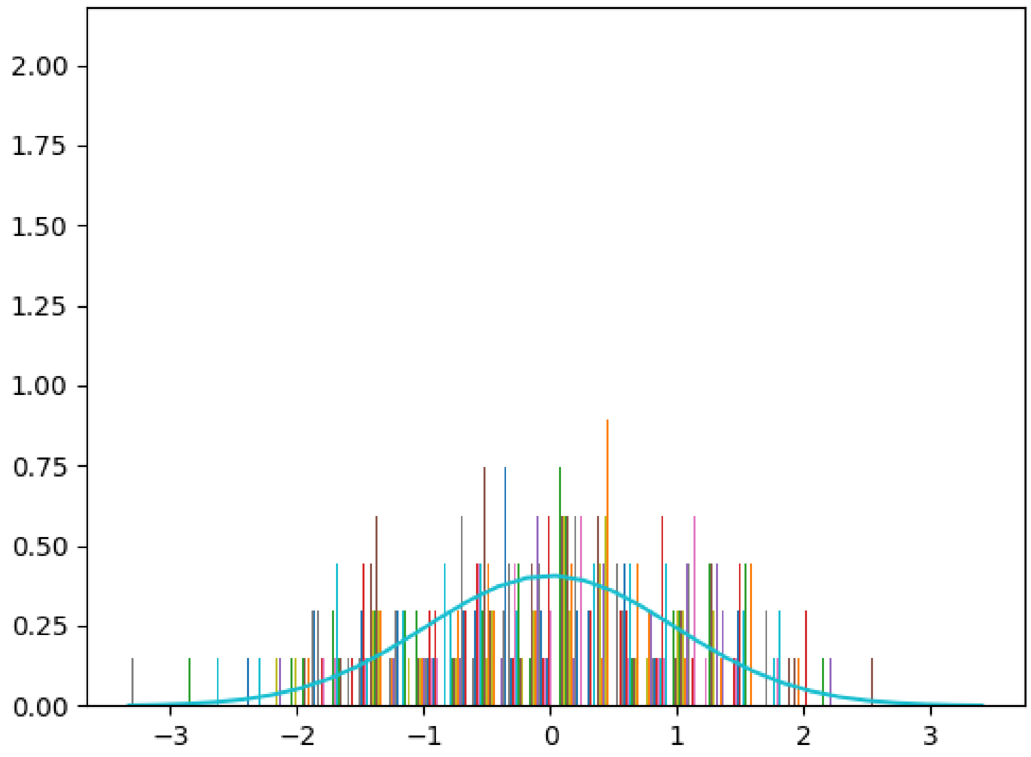


Figure 4.6. Data - Transformed with C

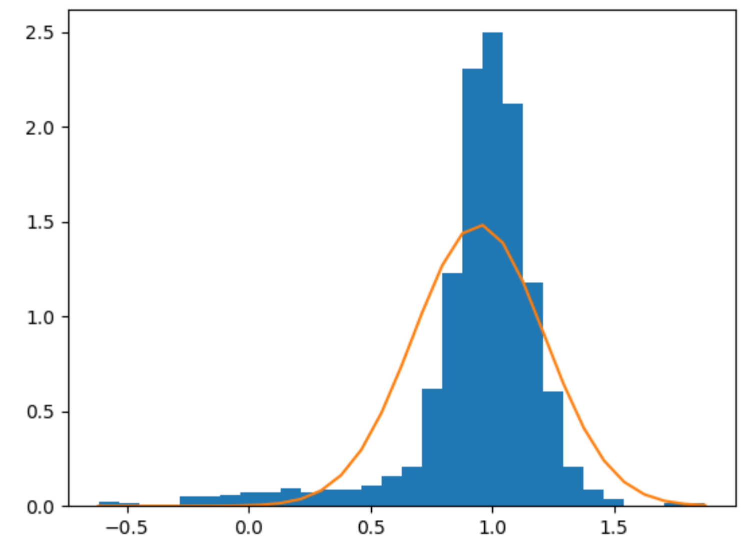


Figure 4.7. λ - C

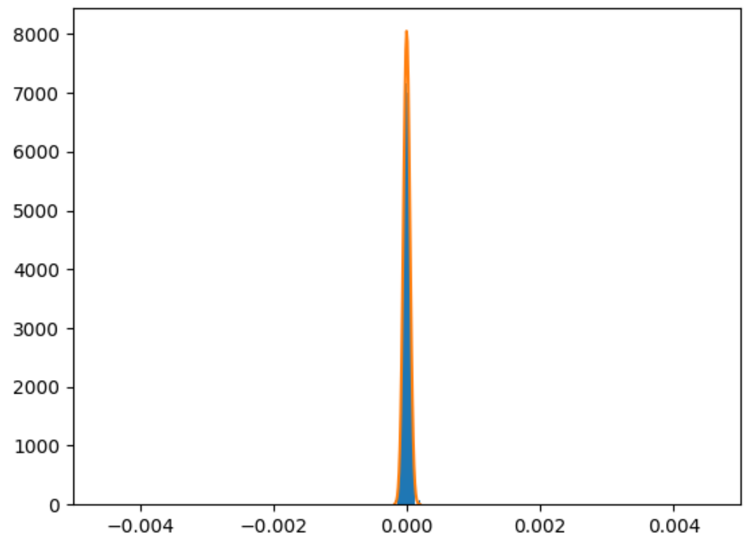


Figure 4.8. Skew - C

The distribution of λ values (see Figure 4.7 for C and Figure 4.4 for Python) does not exhibit a significant difference. However, the distribution of λ values for the C library is slightly steeper compared to the Python library. While the C library may yield λ values greater than 1.5, the Python library does not. A direct comparison of λ values results in an average deviation of 0.1178 and a maximum deviation of 1.6136 across all test data. Over 100 repetitions, the C library took 2.927 seconds per repetition on average to transform the test data and compute the λ values.

4.6 Python interface for C Library: Task 5

In order to integrate the library into Python, we developed an interface to enable the usage of compiled C shared object files (SOF) for Linux or dynamic link libraries (DLL) for Windows. The library `ctypes` [2] was selected for this purpose, as it can open and utilize both DLLs and SOFs. The Python interface allows it to load and integrate the corresponding libraries for the Linux and Windows operating systems.

The compiled library must be linked differently for different operating systems. For Linux, the parameter `-pthread` must be specified, while for Windows, `-pthread -static` is required. The latter results in the `pthread` library, which is typically installed by default on Linux and being statically included in the compiled library for Windows, as this library is not readily available by default on Windows. Additionally, Linux utilizes a different file format for libraries, known as SOF. The distinction between dynamic linking on Linux and the necessary static linking for Windows results in the library requiring less memory for Linux, as the `pthread` library is not included within the compiled SOF by default.

Access to the functions implemented in C is provided through the file `src_c/c_accesspoint.py` and is importable as a Python library (see GitHub yeo-johnson-transformation). The structure MATRIX is created and replicated using data types from the Python library `ctypes` [2]. This is necessary because a clearly defined form is required to translate Python objects for C. The private function `_construct_c_matrix` creates a MATRIX struct from a numpy array.

The private function `_yeo_johnson_output` accepts the following parameters:

`yeo_johnson_c` function from the C library

`c_data_type` data type

`matrix` numpy array

`interval_start` start of the search interval

`interval_end` end of the search interval

`interval_parameter` step size between start and end or recursion

`standardize` flag indicating whether standardization should be

`time_stamps` flag indicating whether time measurement should be done

`thread_count` number of threads to create

If `thread_count` is zero, this variable is not passed to the function. Using these parameters, the passed function (`yeo_johnson_c`) is executed on the matrix, and a numpy array is returned containing all the λ values of the vectors and their corresponding resulting skewness. `yeo_johnson` is the public function that should be filled with the user's selected parameters.

4.7 FPGA Implementation: Task 3

The previously described C library served as the basis for the FPGA implementation of the Yeo-Johnson algorithm. However, due to the impracticality of handling dynamically sized input values on an FPGA, the maximum number of input vectors was limited to 50 values.
While usually fixed-point arithmetic is more efficient on an FPGA, we decided against implementing a design that utilizes the fixed-point data type. The nature of the calculation necessitates a very large range of numbers, which can be better represented using a floating-point data type.
For a single CPU implementation, the order in which a result A or B is calculated is not of great significance, as it operates sequentially. However, in an FPGA hardware implementation, it is more efficient to first calculate intermediate results generated with the same variables and then compute the final results. This approach can save time by allowing the parallel generation of intermediate results instead of waiting for one result or performing the calculation multiple times. The first following optimization steps refer to this principle.

4.7.1 Skewness Optimization: Task 4

The implementation of the skewness calculation is not trivial, as explained in Chapter 4.2. Both the mean and the standard deviation are required to compute the skewness. For a software implementation, the approach would be as follows:

```
float mean(float vector[50], int rows) {
    float result = 0;
    for (int i = 0; i < rows; i++) { result += vector[i]; }
    return result/rows;
}

float variance(float vector[50], int rows, float mean) {
    float result = 0;
    for (int i = 0; i < rows; i++) { result += pow(vector[i] - mean, 2); }
    return result/rows;
}

float skew(float vector[50], int rows) {
    float result = 0;
    float mean = mean(vector, rows);
    float variance = variance(vector, rows, mean);
    float standard_deviation = sqrt(variance);
    for (int i = 0; i < rows; i++) {
        result += pow((vector[i] - mean) / standard_deviation, 3);
    }
    return result/rows;
}
```

However, a problem arises in this implementation: there is a dependency between the sequential calculation of the mean, variance, and standard deviation. The variance requires the mean as input, and the standard deviation requires the variance.

Therefore, the hardware compiler would compute these results one after another. Additionally, each value in the vector is read multiple times, significantly impacting latency. The goal is to minimize the number of memory accesses so that each value is ideally read only once. Furthermore, as many calculations as possible should be performed in parallel, requiring the formulas to be decomposed into sub-results.
When considering the formula for the mean, it becomes evident that no optimization is possible, as it requires the sum of all values in the vector.

$$mean = \frac{1}{n} * \sum_{i=1}^{n}(x_i)$$

However, for the formula of variance, the second binomial formula can be applied.

$$variance = \frac{1}{n} * \sum_{i=1}^{n}(x_i - mean)^2$$

Accordingly, after applying the second binomial formula and resolving the sum, the following formula is obtained:

$$variance = \frac{1}{n} * (\sum_{i=1}^{n}(x_i^2) - 2 * mean * \sum_{i=1}^{n}(x_i) + n * mean^2)$$

When applying the new formula, it is no longer necessary to wait for the calculation of the mean, as the partial results $\sum_{i=1}^{n}(x_i^2)$ and $\sum_{i=1}^{n}(x_i)$ can already be generated during the calculation of the mean. This way, each value of the vector only needs to be read once. Subsequently, the mean can be inserted, the square root can be taken, and the calculation can be completed.

A similar approach is possible for the calculation of the skewness, but in this case, the third-order binomial formula must be applied.

$$(a - b)^3 = a^3 - 3a^2b + 3ab^2 - b^3$$

In order to apply the third-order binomial formula, the formula for skewness needs to be modified.

$$skewness = \frac{1}{n}\sum_{n=1}^{N}\left(\frac{x_n - mean}{standarddeviation}\right)^3$$

By dividing within the sum, the following formula is obtained:

$$skewness = \frac{1}{n}\sum_{n=1}^{N}\left(\frac{x_n}{standarddeviation} - \frac{mean}{standarddeviation}\right)^3$$

Applying the binomial formula of the third order, we get (in the following formula, the mean is abbreviated as m):

$$skewness = \frac{1}{n}\left(\frac{\sum_{i=1}^{n}(x_i^3) - 3m\sum_{i=1}^{n}(x_i^2) + 3m^2\sum_{i=1}^{n}(x_i) + nm^3}{standarddeviation^3}\right)$$

This modification leads to additional intermediate results that can be generated in parallel to the calculation of the mean. The sums $\sum_{i=1}^{n}(x_i^2)$ and $\sum_{i=1}^{n}(x_i)$ were already required for calculating the variance, so only the sum $\sum_{i=1}^{n}(x_i^3)$ needs to be added.

With this optimization, the number of data accesses for calculating the skewness is minimized. For the transformation, this means that each value of the vector needs to be computed only once per iteration.

4.7.2 Optimization of Yeo-Johnson Transformation: Task 4

The Yeo-Johnson transformation is a composite function. The software-level implementation would appear as follows:

```
float yeo_johnson_transformation(float y, float lambda) {
    if (y >= 0) {
        if (lambda != 0) {
            return (pow((y + 1),lambda)-1)/lambda;
        } else {
            return log(y + 1);
        }
    } else {
        if (lambda != 2) {
            return -(pow((-y + 1),(2-lambda))-1)/(2-lambda);
        } else {
            return -log(-y + 1);
        }
    }
}
```

The current implementation suffers from three issues. Firstly, it utilizes nested if-else branches, leading to a substantial control logic that can be avoided through an alternative programming style. Secondly, there are multiple return statements, resulting in increased complexity of the output. Lastly, the same functions are called in different branches, resulting in excessive memory usage without any performance improvements. To address these issues, the return statements can be moved to the end of the function to save on logical operations. The nested if statements can be resolved by manipulating the input values. If y is less than zero, the y-value can be multiplied by -1, λ can be replaced with $2 - \lambda$, and a factor equal to -1 can be set. This ensures that the logic behaves the same way for positive and negative y-values.

```
float yeo_johnson_transformation(float y, float lambda) {
    float result;
    float return_factor = 1.0;
    if (y < 0) {
        y = -y;
        lambda = (float) 2.0 - lambda;
        return_factor = -1.0;
    }
    y = y + 1.0;
```

```
        if (abs(lambda) <= 0.001) {
            result = return_factor * hls::logf(y);
        } else {
            result = return_factor * ((hls::powf(y, lambda) - 1.0)/lambda);
        }
        return result;
    }
```

4.7.3 Yeo-Johnson Transformation Module

The module for calculating the Yeo-Johnson transformation uses a data stream as input parameter for the data to be transformed, along with three integer parameters: the number of rows per vector, the size of the data stream (50 values per packet), and a precision parameter. The precision parameter determines how many times the algorithm should be repeated, affecting both the achieved results and the runtime. The number of rows per vector is limited to a maximum of 50 values, as dynamic memory management would introduce high latency and cannot be implemented on the FPGA. On the other hand, the size of the data stream is not bound by the resources of the FPGA, as it can theoretically be read from off-chip memory.

By limiting the depth of the data streams to 50 values, FPGA resources can be saved without significantly impacting the runtime, as only a maximum of 50 values can be read per iteration of the packet loop. The variables `rows`, `size_of_stream`, and `return` are connected to the control bus to access these input parameters later on.

By partitioning the stored vector and parallelizing for three loop iterations, it is possible to calculate three transformations and their skewness in parallel with a single read access to the vector. When reading the values from the input data stream for a precision greater than or equal to one, the values need to be stored since data from the stream can only be read once. During the storing process, partial results are already computed, which are later needed for calculating the standard deviation and skewness. As a result, the final computation of skewness requires only a few clock cycles. An implementation without loops would be desirable, but it is not feasible due to the limited resources of the FPGA being used. Without loops, the implementation would become approximately 50-100 times as large.

A helper function (`compare_skew`) determines the λ value that achieves a lower skewness. Based on this result, the starting λ is adjusted. The chosen precision determines how many times the algorithm is applied to the stored vector. Finally, the output is the λ value that achieves skewness as close to zero as possible. The following code demonstrates the implementation of the described module:

```
void auto_search(    hls::stream< trans_pkt > &in_stream,
                     hls::stream< trans_pkt > &out_stream,
                     int rows, int size_of_stream, int precision) {

// Definition of inputs, outputs, and control registers
#pragma HLS INTERFACE mode=axis port=in_stream depth=50
#pragma HLS INTERFACE mode=axis port=out_stream depth=50
#pragma HLS INTERFACE s_axilite port=rows bundle=CTRL_BUS
#pragma HLS INTERFACE s_axilite port=size_of_stream bundle=CTRL_BUS
```

```
#pragma HLS INTERFACE s_axilite port=precision bundle=CTRL_BUS
#pragma HLS INTERFACE s_axilite port=return bundle=CTRL_BUS

    // Packet loop
    for (int packet = 0; packet < size_of_stream; packet++) {
        float vector[50];
#pragma HLS ARRAY_PARTITION variable=vector type=complete
        float sum1[3] = {0, 0, 0};
#pragma HLS ARRAY_PARTITION variable=sum1 type=complete
        float sum2[3] = {0, 0, 0};
#pragma HLS ARRAY_PARTITION variable=sum2 type=complete
        float sum3[3] = {0, 0, 0};
#pragma HLS ARRAY_PARTITION variable=sum3 type=complete
        float mean[3] = {0, 0, 0};
#pragma HLS ARRAY_PARTITION variable=mean type=complete
        float skew[3] = {0, 0, 123456};
#pragma HLS ARRAY_PARTITION variable=skew type=complete
        float lambda = 0;
        float step_value = 2;

        // First read access per packet
        for (int i = 0; i < MAX_ROW; i++) {
#pragma HLS PIPELINE
            read_v = in_stream.read(); // Read
            float value = *(float *) &read_v.data;
            vector[i] = value; // Store
            // Transformation
            float v1 = yeo_johnson(value, -2);
            float v2 = yeo_johnson(value, 2);
            float v3 = yeo_johnson(value, 0);
            // Partial results
            sum1[0] += v1;
            sum2[0] += v1 * v1;
            sum3[0] += v1 * v1 * v1;
            mean[0] += v1 / rows;
            sum1[1] += v2;
            sum2[1] += v2 * v2;
            sum3[1] += v2 * v2 * v2;
            mean[1] += v2 / rows;
            sum1[2] += v3;
            sum2[2] += v3 * v3;
            sum3[2] += v3 * v3 * v3;
            mean[2] += v3 / rows;
        }
        for (int i = 0; i < 3; i++) { // Skewness
#pragma HLS UNROLL
            float m = mean[i];
            float s1 = sum1[i];
            float s2 = sum2[i];
            float s3 = sum3[i];
            float mean2 = m * m;
            float mean3 = mean2 * m;
            float variance = (s2 - 2 * m * s1 + rows * mean2) / rows;
            float standard_deviation = hls::sqrt(variance);
            float skew_dividend =
                (s3 - 3 * m * s2 + 3 * mean2 * s1 - rows * mean3);
            float sd3 = standard_deviation *standard_deviation *standard_deviation;
            float skewness = (skew_dividend/sd3)/rows;
            skew[i] = hls::absf(skewness);
        }
        int flag = compare_skew(skew, &skew[2]); // Comparison
        lambda += step_value * flag;
        step_value /= 2;
        for (int i = 0; i < precision; i++) { // Precision loop
            transform(vector, &skew[2], &lambda, &step_value, rows,
            sum1, sum2, sum3, mean, skew);
        }
        // Write result
        read_v.data = *(int *) &lambda;
        out_stream.write(read_v);
    }
}
```

4.7.4 Module Control

Using Direct Memory Access (DMA), data can be transferred as a stream from the FPGA's memory to the module, and the module's outputs can be directly stored in the memory through DMA.

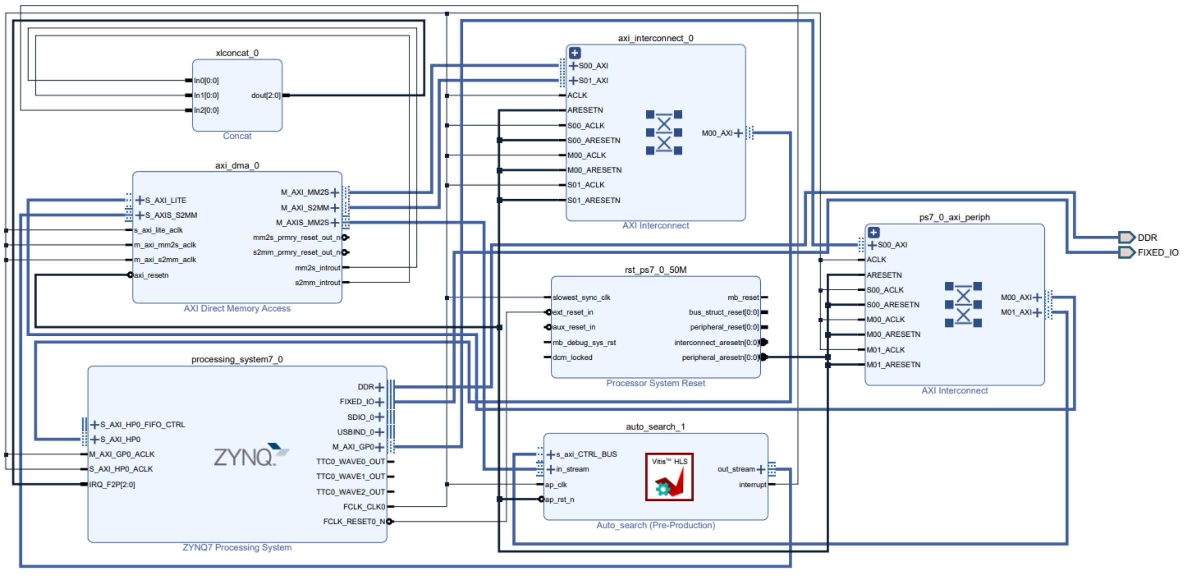


Figure 4.9. Block Diagram

Based on the block diagram shown in Figure 4.9 we developed the software that generates the data stream, initializes the peripherals and stores the results. The module is instantiated, and the addresses of the memory regions are defined. Receive and transmit buffers for DMA are created and assigned to these memory regions. During the development phase, the data is provided via an SD card, which is exchanged in a larger computer network by accessing memory outside the FPGAs. For the purpose of functionality verification, the transfer of an address is simulated by reading the SD card and writing the data into a predefined memory.

Upon startup, the peripherals are initialized, including the `Auto_Search` module, the DMA connection, and the SD card simulating the external memory. Since only unsigned integer values can be transferred via the DMA connection, floating-point numbers need to be transformed into integer values while preserving the original floating-point value. Direct type conversion would result in a loss of the fractional range, so the address of the read floating-point number must be reinterpreted to avoid automatic type conversion. This is implemented as follows:

```
float a = 3.0;
int b = *(int *) &a;
```

In this way, the floating-point value is converted into the decimal value 1077936128, which contains the binary data of the floating-point number. The calculation of the real value is derived from the sign, mantissa, and exponent. The newly interpreted values are written into a data stream, which is then made available to the module. The generated results are stored in the output buffer. Once the transfer is complete and the module reports that it has finished its activity, the collected data is written back to the designated memory address.

```
// Defines
#define MEM_BASE_ADDR^^I0x1000000
#define TX_BUFFER_BASE^^I(MEM_BASE_ADDR + 0x00100000)
#define RX_BUFFER_BASE^^I(MEM_BASE_ADDR + 0x00300000)
#define FLOAT_SIZE 4
#define COLS 50000
#define DATA_LENGTH (COLS * ROW)
#define INPUT_FILE_SIZE (FLOAT_SIZE * DATA_LENGTH)
// Module Parameter
#define PACKETS 50000
#define SIZE_ARR 50*PACKETS
#define PRECISION 7
#define ROW 30

// Function prototypes
static int SD_Init();
static int SD_Eject();
static int ReadFile(char *FileName, u32 DestinationAddress);
static int WriteFile(char *FileName, u32 size, u32 SourceAddress);

// Globals
XAuto_search auto_search;
```

```
XAuto_search_Config *auto_search_cfg;
XAxiDma axiDMA;
XAxiDma_Config *axiDMA_cfg;
FATFS fatfs;
int inStreamData[SIZE_ARR];
char filebuffer[INPUT_FILE_SIZE];

// Functions
static int SD_Init() {
    ...
}

static int SD_Eject() {
    ...
}

static int ReadFile(char *FileName, u32 DestinationAddress) {
    ...
}

static int WriteFile(char *FileName, u32 size, u32 SourceAddress) {
    ...
}

void initPeripherals() {
    // Initialize Auto_Search Module
    auto_search_cfg = XAuto_search_LookupConfig(0);
    if (auto_search_cfg) {
        int status = XAuto_search_CfgInitialize(&auto_search, auto_search_cfg);
        if (status != XST_SUCCESS) {
            printf("Error initializing auto_search core\r\n");
        } else {
            printf("init auto_search success\r\n");
        }
    }
    // Initialize DMA
    axiDMA_cfg = XAxiDma_LookupConfig(XPAR_AXIDMA_0_DEVICE_ID);
    if (axiDMA_cfg) {
        printf("Look up done\r\n");
        int status = XAxiDma_CfgInitialize(&axiDMA, axiDMA_cfg);
        if (status != XST_SUCCESS) {
          printf("Error initializing AxiDMA core\r\n");
        } else {
            printf("init axiDMA success\r\n");
        }
    } else {
        printf("Axi Dma unable to load config\r\n");
    }
    // Interrupt disable
```

```
        XAxiDma_IntrDisable(&axiDMA, XAXIDMA_IRQ_ALL_MASK, XAXIDMA_DEVICE_TO_DMA);
        XAxiDma_IntrDisable(&axiDMA, XAXIDMA_IRQ_ALL_MASK, XAXIDMA_DMA_TO_DEVICE);
        // Mount SD Card
        int rc = SD_Init(&fatfs);
        if (rc != XST_SUCCESS) {
           xil_printf("ERROR: SD Init failed\n");
        }
    }

    // float to int without transformation
    int floatToInt(float a) {
        return *(int *) &a;
    }

    // int to float without transformation
    float intToFloat(int a) {
        return *(float *) &a;
    }

    int main()
    {
        // Pointers to DMA TX/RX Adresses
        int *m_dma_buffer_TX = (int *) TX_BUFFER_BASE;
        int *m_dma_buffer_RX = (int *) RX_BUFFER_BASE;
        // Initialisierung der Peripherie
        initPeripherals();
        // read Data from File
        int rc = ReadFile("output.bin", (u32) filebuffer);
        if (rc != XST_SUCCESS) {
            printf("ERROR: Read file failed\n");
            return XST_FAILURE;
        }
        // Create Datastream
        int counter1 = 0;
        int counter2 = 0;
        for (int j = 0; j < PACKETS; j++) {
            for (int i = 0; i < 50; i++) {
                if ((i % 50) < ROW) {
                    inStreamData[counter1++] = ((int *) &filebuffer)[counter2++];
                } else {
                    inStreamData[counter1++] = 0;
                }
            }
        }
        // Start Module / Initialize
        XAuto_search_Set_rows(&auto_search, (u32) ROW);
        XAuto_search_Set_size_of_stream(&auto_search, (u32) PACKETS);
        XAuto_search_Set_precision(&auto_search, (u32) PRECISION);
        XAuto_search_Start(&auto_search);
        // Clear Cache
        Xil_DCacheFlushRange((u32) inStreamData, SIZE_ARR*sizeof(float));
        Xil_DCacheFlushRange((u32) m_dma_buffer_RX, SIZE_ARR/50*sizeof(float));
        // DMA Transfer - Send
        XAxiDma_SimpleTransfer(&axiDMA, (u32) inStreamData, SIZE_ARR *
            sizeof(float), XAXIDMA_DMA_TO_DEVICE);
        // DMA Transfer - Receive
        XAxiDma_SimpleTransfer(&axiDMA, (u32)m_dma_buffer_RX, (SIZE_ARR/50) *
            sizeof(float), XAXIDMA_DEVICE_TO_DMA);
        // wait for Transfer
        while(XAxiDma_Busy(&axiDMA, XAXIDMA_DEVICE_TO_DMA));
        // Invalidate Cache to avoid reading Garbage
        Xil_DCacheInvalidateRange((u32)m_dma_buffer_RX, SIZE_ARR*sizeof(float));
        // Wait for Module to finish
        while(!XAuto_search_IsDone(&auto_search));
        // Save Results
        for (int i=0; i < 50000; i++) {
            ((float *) filebuffer)[i] = intToFloat(m_dma_buffer_RX[i]);
        }
        // Write Results
        rc = WriteFile("result.bin", PACKETS * FLOAT_SIZE, (u32) filebuffer);
        if (rc != XST_SUCCESS) {
            printf("ERROR: Could not write to file\n");
        }
        // Dismount SD Card
        int status = SD_Eject(&fatfs);
        if (status != XST_SUCCESS) {
            printf("SD card eject failed\n");
        }
        return 0;
    }
```

4.7.5 Verification

To assess the quality of the generated results, the aim is to minimize the skewness as much as possible. Upon examining the skewness distribution for a precision of one (see Figure 4.10(1)), it can be observed that an improvement has occurred compared to the previous skewness values. However, the range of $[-1, 1]$ is still too large, resulting in poorer values than those obtained using the Python library. The chosen precision achieved an average runtime of 0.64 seconds for the specified test data with one module. As the time savings increase proportionally with each additional module, only a single module is used for verification.

Figure 4.10 depicts the distribution of skewness for different precisions. (1) illustrates the distribution for a precision of one, (2) for a precision of two, and so on. It is evident that the range of values narrows around zero for higher precisions. Additionally, (8) demonstrates the increase in density around zero for higher precisions. In this case, the blue histogram represents the distribution of skewness for a precision of one, the green histogram represents a precision of four, and the purple histogram represents a precision of seven. The average deviation of the resulting skewness values is listed in the following table:

Precision	Deviation
1	0.31642
2	0.28414
3	0.17671
4	0.08945
5	0.04474
6	0.02227
7	0.01116

Table 4.1. Skewness Deviation - C Library Comparison

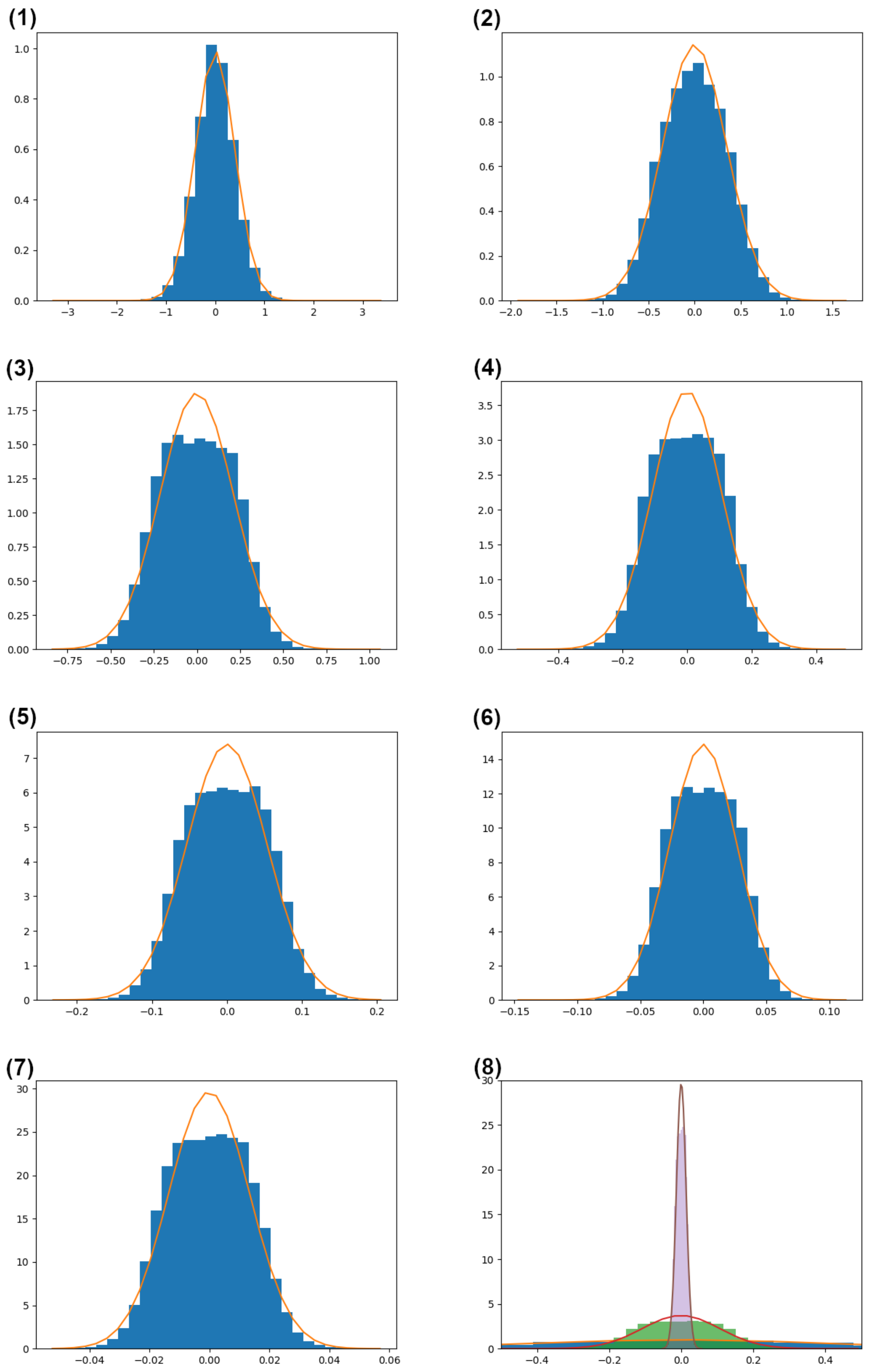


Figure 4.10. Skewness - FPGA

It can be observed that the skewness tends toward zero as the precision increases.

The runtime of the calculations exhibits an approximately linear behavior, as shown in Figure 4.11. Since the same logic is repeatedly applied with increasing precision, it is unrealistic to expect this type of repetition to result in an acceleration leading to logarithmic time complexity.

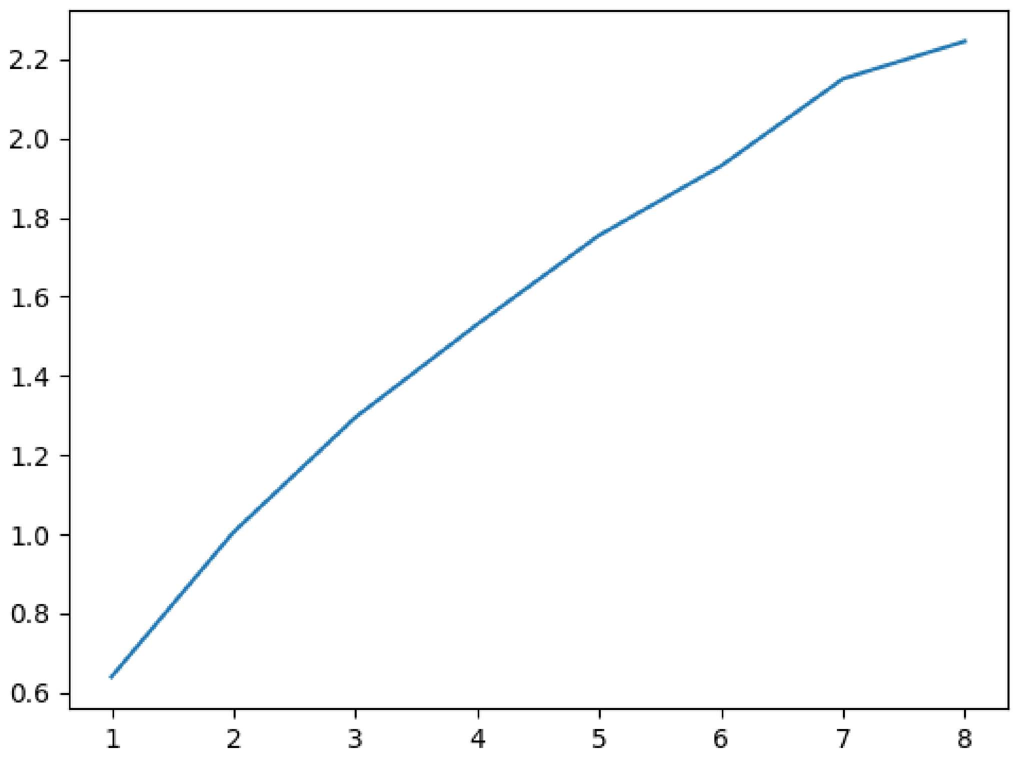


Figure 4.11. Latency - FPGA

4.7.6 Python Interface for FPGA Version: Task 5

The Python interface to establish a connection with the FPGA functionality uses the `Paramiko` Python library. It enables the execution of a script on the computer to which the FPGA is connected.
Unlike the C library, the FPGA implementation solely necessitates two parameters: the data and a precision parameter. The latter specifies how many times the algorithm should be applied to the data. As this precision parameter increases, the skewness of the transformed vectors approaches zero. Only the two mentioned parameters are required because no threads are created and variable bounds for the search cannot be defined. Parallelization occurs on the FPGA rather than the software level, eliminating the need for thread creation. The variable bounds for searching a λ have been replaced with fixed bounds. This is because results outside the defined range are very rare, and using fixed bounds reduces latency, as no large numbers need to be computed. Furthermore, the FPGA implementation only returns an array of λ values. Hence no transformed data is returned, which, accordingly, cannot be standardized.

Before use, it must be ensured that the FPGA is initialized. To do so, the function `init_fpga` is called, which loads the bitstream onto the FPGA. Subsequently, an array, along with a precision parameter, can be passed using the function `yeo_johnson_fpga (np_array, precision)`. The FPGA generates the results, which are returned as an array by the function. Username and password to establish the ssh connection must be provided.

To provide the data to the FPGA, the passed array is stored as a binary file at a predefined location. A script is used to write the stored data into the FPGA's memory. The results from the FPGA are copied from its memory to another predefined location using the same script. They are then read and returned as an array.

4.7.7 Evaluation and Characterization

4.7.7.1 Test System

The following development benchmarks were all measured on the same system. For both the Python and C implementation a CPU with a base clock frequency of 3.0 GHz and 8 physical cores was used. It has a thermal design power of 65 watts, with a power consumption of 30.9 watts during operation and 12.4 watts in standby mode. Furthermore, the maximum power of the zedboard ZYNQ-7000 used for testing one singular module is 24 watts, with a measured power consumption of 5.2 watts. The tests for the implementation involving multiple modules were conducted within the VEDLIoT testbed on the ZYNQ-7045. Which has a maximum power of 24 watts and during the tests exhibits a measured power of 7.1 watts.

4.7.7.2 Test Data

A fixed quantity of 50,000 vectors, each consisting of 30 values, is used to ensure better comparability between the different methods. The distribution of this artificial test data is illustrated in Figure 4.12. As can be observed, the majority of values are concentrated near zero, but there are also some outliers that reach up to a value of 100.

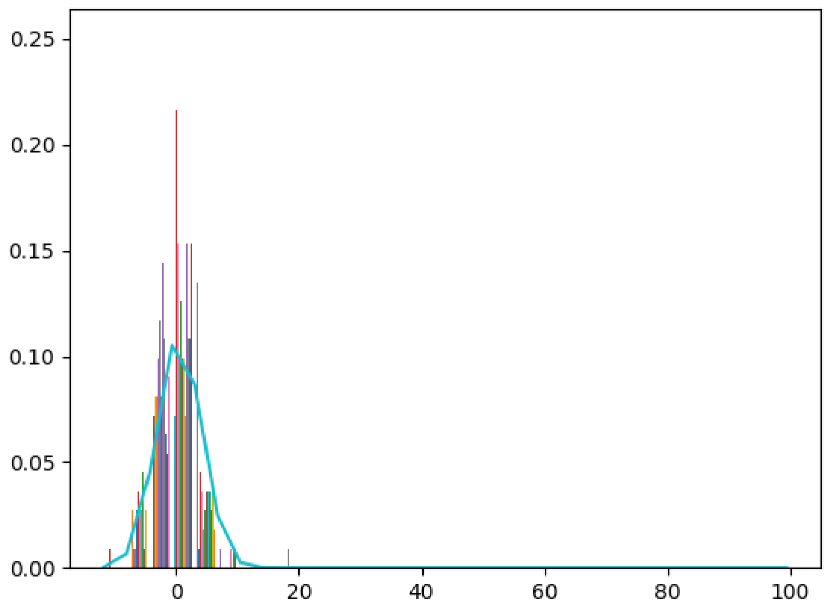


Figure 4.12. Distribution - Untransformed Data

4.7.7.3 Benchmark Energy Consumption and Duration

The calculation of λ values and the transformation of the test data using the `sklearn.preprocessing.PowerTransformer` library took 40.6534 seconds, averaging over 100 repetitions. Figure 4.13 illustrates the latency times relative to the different implementations. It is evident that both the C library and the FPGA implementation offer significant time savings compared to the Python library. However, the time difference between the C library and the FPGA implementation with a single module is not significantly larger when considering the same precision. This can be attributed to the limited resources of the development FPGA used.

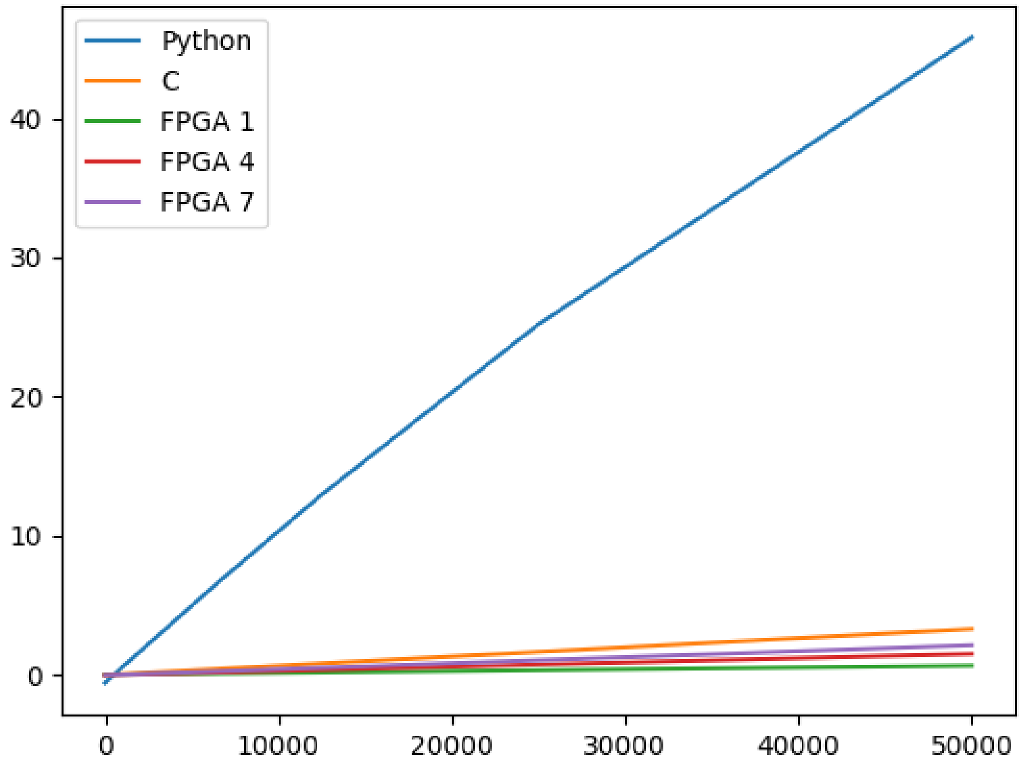


Figure 4.13. Latency Comparison: Python vs. C vs. FPGA

By utilizing an FPGA with more resources, the throughput of the design can be further enhanced. This can be achieved, for instance, by instantiating the described module multiple times and distributing the data stream across these modules. As a result, the data stream can ideally be continuously read while each module performs parallel computations and simultaneously supplies the next module with values.

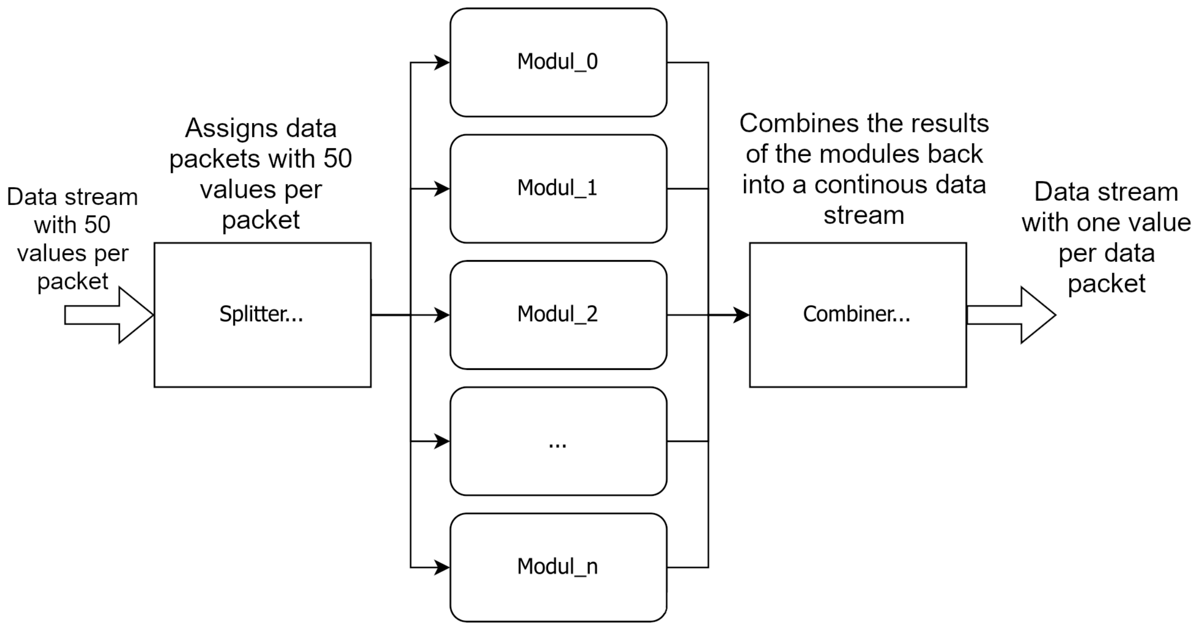

Data stream with 50 values per packet
Assigns data packets with 50 values per packet
Splitter...
Modul_0
Modul_1
Modul_2
...
Modul_n
Combines the results of the modules back into a continous data stream
Combiner...
Data stream with one value per data packet

Figure 4.14. Extended design

Although this procedure can be continued with any number of modules, ultimately, a bottleneck arises when reading the data through the splitter. Each module requires a certain time (t) to produce a result. If the splitter takes more time than t to return to the module it has just supplied with a data packet, all modules are not fully utilized. In such cases, additional modules do not increase the throughput but only consume additional resources on the FPGA.

This approach is performed on a ZYNQ-7045 FPGA. The FPGA is accessible through a Linux operating system and is located in a computer network connected to the main memory. The data to be processed is read from the main memory of the computer network and loaded into the FPGA's RAM through a process. Parameters such as the number of rows per vector, the length of the data stream, and the accuracy or number of iterations are passed as a parameter list to the process. This information is stored in the corresponding registers of the FPGA module. Subsequently, the module is started, and the results are stored in the main memory. Each module utilizes 201 DSPs, 73,139 flip-flops, 21 BRAMs, and 51,832 look-up tables. According to the datasheet of the ZYNQ-7045, a total of 900 DSP slices, 218,600 look-up tables, 437,200 flip-flops, and 545 BRAMs (each 36 Kb blocks) are installed. Thus, it is possible to deploy four modules to fully utilize the FPGA's resources.

The division of the data stream into multiple substreams, instantiation of multiple modules, and subsequent merging of the individual substreams are performed within the IP core referred to as *auto_search_1* in the block diagram 4.9. Therefore, there is no need to modify the software that writes to pre-defined memory addresses of the FPGA. Instead, only the IP core in the block diagram needs to be replaced.

As mentioned earlier, splitting the original data stream among multiple modules enables their parallel processing. Hence, doubling the number of modules is expected to cut the runtime in half. This is depicted in table 4.2, where the processing time decreases with an increasing number of modules for different precisions. It is evident that the runtime indeed halves as expected.

The following table presents the determined times in seconds for various precisions and numbers of modules.

Module	Precision=2	Precision=4	Precision=8	Precision=16
1	1.03522	1.70079	3.0275	5.68394
2	0.49962	0.82966	1.4914	2.81094
3	0.33459	0.55118	0.9907	1.86605
4	0.25026	0.41462	0.7447	1.39872

Table 4.2. Times for 1 to 4 modules

Similarly, in the case of the C library, the use of a CPU with more cores could be considered to further reduce latency. Therefore, to determine efficiency, the comparison between FPGA and C library is made across all test data with a precision of seven, one thread for the CPU, and one module for the FPGA implementation.

For the same calculation, the C library on a CPU with a base clock frequency of 3.0 GHz and the previously defined input parameters takes 4.77 seconds, with a total time of 5.742 seconds. Consequently, the call from Python resulted in a delay of 0.972 seconds. The FPGA implementation with a singular module requires 2.15 seconds for the same calculation, making it 2.62 seconds faster than the C library.

The implementation, which utilizes 5 modules, further reduces this to 0.67 seconds for the computation. However, due to the necessary data transfer between the system and the FPGA for implementation on the ZYNQ-7045, the time for writing the data (0.206 seconds) and reading the data (0.0065 seconds) must also be taken into account. Accordingly, the total runtime amounts to 0.883 seconds, and this time is used and referenced as the computation time going forward. Although the time required for data access to the FPGA and main memory needs to be added, this delay can be disregarded for calculating energy consumption since the FPGA can be continuously supplied with data, and this delay only occurs once at the start and end.

The maximum power consumption of the FPGA used for this test series is 24 watts, with the measured power during operation at 5.2 watts. The CPU used has a power consumption of 65 watts (TDP = Thermal Design Power), with a measured power of 30.9 watts during calculation and 12.4 watts in standby mode. Thus, the FPGA implementation requires a maximum of 51.6 watt-seconds and a minimum of 11.18 watt-seconds for the calculation of λ values, while the CPU requires a maximum of 310.05 watt-seconds and a minimum of 147.393 watt-seconds. Accordingly, the Python library requires a maximum of 2967.055 watt-seconds and a minimum of 1410.492 watt-seconds, representing nearly ten times the amount compared to the C library. An overview of these power consumption values is presented in Table 4.3.

5 WP2: Accelerating Overarching Bioinformatics Workflow

In this section, we discuss the assumptions and optimizations involved in the acceleration of the overarching bioinformatics workflow. Additionally, we present the inte-

	Sklearn	C-Version	ZYNQ-7000	ZYNQ-7045
calculation time	40.65s	4.77 s	2.97s	0.883s
energy consumption (max.)	2967.05 J	310.05 J	51.6 J	21.19 J
energy consumption (min.)	1410.49 J	147.39 J	11.18 J	6.27 J

Table 4.3. Results single-threaded/module

gration of basic embedded feature selection methods and the concept of "reverse" feature selection. Finally, we provide our benchmarking results.

5.1 Python libraries

Our bioinformatics workflow is implemented in Python, a widely used programming language in the field of data science and machine learning. The accelerated ensemble feature selection workflow is based on the following standard libraries: The Python library `Ray` [49][23] for building a cluster, and `Optuna`[21] for hyperparameter optimization. `Ray` is a unified framework for scaling AI and Python applications. It is able to scale workloads from a single machine to a large cluster. Due to a known memory leak in the latest stable `Ray` version 2.5.0 [17] we used the `Ray` version 3.0.0.dev0. To enable the distribution and execution of `Ray` applications across multiple nodes, a `Ray` cluster had to be established on the VEDLIOT testbed. A `Ray` cluster comprises a group of worker nodes interconnected with a central `Ray` head node. These clusters can either possess a fixed size or dynamically scale up and down based on the resource requirements of the applications executing within the cluster. In our particular case, the VEDLIoT test cluster consists of three nodes (see Chapter 5.3.2 for further details).
The selected feature selection methods include the `scikit-learn` LASSO Regression [15], Random Forest [16] and Extra Trees [14]. Random Forest and Extra Trees are also integrated as `LightGBM` [3][37] implementation. `LightGBM` provides a fast C++ implementation and a CUDA version [64] for parallel GPU training (further details can be found in Chapter 5.3.4).

5.2 Accelerated Workflow: Task 1

In order to minimize redundant computations, an initial optimization strategy involves caching preprocessed data as remote objects in `Ray`'s distributed shared memory object store. `Ray` provides one object store per node in the cluster. In the cluster environment, a remote object can reside on one or more nodes, regardless of who owns the object reference(s) [8]. The previously discussed Yeo-Johnson transformation is essential for every data set within the nested cross-validation process associated with LASSO Regression calculations. Therefore, caching reduces the number of thousands of λ optimizations (for details see Chapter 4.1) to 36 for the entire workflow. This calculation assumes six outer and five inner loops within the nested cross-validation. 30 optimizations are required for all inner cross-validation splits. In addition, six more are needed to compute the feature importance with the optimized hyperparameters using the full training data of each outer cross-validation split. An equal number of correlation matrices is also necessary for subsequent correlation analysis.
The acceleration of the Yeo-Johnson transformation itself was extensively discussed in the previous chapters. Additionally, the calculation of the correlation matrix can

be parallelized within the preprocessing. To demonstrate the feasibility, we developed a proof of concept for an FPGA-accelerated version. Further development and refinement will be carried out after the completion of the project.

In addition to the parallelized preprocessing and caching, there exist five more potential layers of parallelization (see Figure 3.1) as detailed in Chapter 3:

- Outer cross-validation
- Inner cross-validation
- Individual embedded feature selection methods of the ensemble selection
- Hyperparameter optimization
- Model training

As a first option, the elements of the nested cross-validation could be parallelized. However, employing outer or inner cross-validation as standalone parallelization strategies would not effectively utilize the available resources, as explained in Chapter 3. As an additional aspect, pruning during hyperparameter optimization means that not all iterations of the inner cross-validation have to be completed. Consequently, this kind of parallelization can lead to unnecessary model training.
The problem of unused resources due to an insufficient number of parallel processes equally applies to the individual embedded feature selection methods of the ensemble selection. In our ensemble feature selection example we integrate six feature selection methods. However, the number of includable methods is variable and could be expanded or reduced. Usually, the number of methods is much smaller than the available cores of a cluster.
Our selection includes the LASSO Regression as well as Random Forest and Extra Trees. In addition, LASSO Regression, Random Forest, and Extra Trees are also integrated into an alternative feature selection method which we refer to as ”reverse” feature selection. When applying the ”reverse” feature selection, instead of directly predicting the class from the training data, we predict each feature individually. Based on the importance of the included class label for the prediction and the regression error, we calculate a weight for each specific feature. Our assumption is that the higher the weight, the more relevant the feature is.
Therefore, the used feature selection methods can be divided into two groups: the standard embedded feature selection and the ”reverse” feature selection. For the standard embedded feature selection methods, parallelizing the hyperparameter optimization or model training is a viable approach to leverage the available resources. However, considering the large number of relatively small models in comparison to the overall workload, over-parallelizing with overly fine-grained tasks would be counterproductive [7]. Parallelizing a function that executes quickly can negatively impact speedup due to higher overhead compared to using normal functions. Furthermore, distributing the hyperparameter optimization across multiple nodes would further increase the overhead, as the trials of the hyperparameter optimization depend on each other and the results need to be synchronized. `Ray Tune` [11] serializes the result of each trial, while `Optuna` provides communication using a database [5]. Both options involve read/write operations that are slower than performing calculations

in memory within a single node. In the case of `Ray Tune`, we encountered a "Too many open files" error, as the large number of rapidly calculated models and hyperparameter value combinations posed challenges in writing the necessary files. None of the considered libraries provides native support for parallel processing within the memory of a single node.
As individually none of the listed parallelization options is an optimal strategy for our use case, we developed a combined approach. To accelerate the feature selection workflow and reduce overhead, we cluster tasks into larger units and apply nested parallelism [9]. The top level of nested parallelization includes the different standard embedded feature selection methods. Since these methods do not rely on each other and require no communication, they can be computed asynchronously. As `Ray` scheduling strategy for this we select `SEPARATED` [10], which aims to distribute the methods across different nodes. For each method, results are saved in a central storage independently. The remaining levels of nested parallelization are scheduled using their respective default strategies.
The second level of parallelization is the outer cross-validation. If more resources are available, model training can be parallelized as the lowest level. Parallelized model training is not controlled by `Ray`, but is already integrated into the respective libraries. Since we use ASHA[40] as a comparison-based pruning strategy to accelerate the hyperparameter optimization for the tree-based methods, trials executed in parallel could be terminated earlier. The reason is a fast detection of unpromising trials especially at the beginning of the optimization. Although hyperparameter optimization as the third level of nested parallelism would be even more beneficial, the selected libraries currently do not support this option as explained above. Therefore, it could be a possible next step to evaluate alternative hyperparameter optimization libraries.

Reverse feature selection, on the other hand, requires a different parallelization approach. As each calculation for a feature includes a complete embedded feature selection process, tasks can be clustered at the feature level. With 5,000 features, we would have 5,000 parallel tasks. However, starting too many tasks increases overhead and can lead to program abortion. Therefore, we utilize batching by grouping features based on the number of available cores and parallelize only the batches. Reverse feature selection is synchronized, as the order of outcomes is relevant for later result interpretation. Subsequently, the evaluation of the selected feature subsets can be performed as an uncoupled step. This analysis is quick and does not require further acceleration. The parallelization itself can be configured and thus adapted to reconfigurable hardware.

5.3 Evaluation and Characterization

This section provides an overview of the test data used in the evaluation process. We discuss the hardware components used within the VEDLIoT testbed and detail the techniques and tools employed for power measurement during our experiments. Subsequently, we compare the overhead of GPU training and CPU training for the `LightGBM` machine learning library. Finally, we present the benchmark results for energy consumption and duration of the accelerated feature selection workflow.

5.3.1 Test Data

For method development and benchmarking, we use artificial data to know the expected outcome [59]. While artificial data allows us to have a clear understanding of the underlying information, when dealing with real biological data, the inherent information and random noise remain largely unknown. In order to test and advance the ensemble feature selection technique for wide data sets, we have implemented and published a data generator that simulates high-dimensional biological data which is published at `https://github.com/sigrun-may/artificial-data-generator`. This generator incorporates artificial biomarkers with specific class distributions and outliers. For our following benchmarking experiments, we utilized a test data set with 30 samples and 50,000 features including 30 artificial biomarkers. These artificial biomarkers are positioned at the beginning of the columns, while the last columns are populated with random noise. This setup allows us to selectively choose subsets of features, thus enabling the simulation of varying feature sizes.

5.3.2 VEDLIoT Test Environment

A modular microserver system (RECS®|Box [60]) that allows arbitrary combinations of x86, ARM, GPU and FPGA compute modules is the basis for the used VEDLIoT testbed hardware. RECS®|Box is the selling name of the Microserver chassis sold by Christmann [1]. The `Ray` cluster (see Chapter 5.1) established on the VEDLIoT testbed includes three nodes. All three nodes have an x86_64 architecture. The head node (Node 4) has an Intel(R) Xeon (R) E-2176M CPU@2.70GHz with six cores. To prevent oversubscription on the head node, only five physical cores were integrated into the cluster resources. The first worker node (Node1) includes an AMD EPYC 3451 16-Core Processor and the second worker node (Node3) an Intel(R) Xeon(R) CPU D-1577 @ 1.30GHz. Both of these nodes offer a pool of 32 available cores. Node4 additionally provides a Tesla P100 GPU. As already introduced in Chapter 4.7.7.1 a ZYNQ-7045 FPGA completes the hardware test environment.
The maximum number of parallel threads on CPUs would be 76, with 32 cores on Node1, 32 cores on Node3, and 12 cores on Node4. However, since the `Ray` head node, the underlying operating system, and the measurement of energy consumption also utilize this hardware, we have chosen to include only 74 cores in our benchmark example.

5.3.3 Power measurement

There are different software interfaces available to monitor the status of the RECS® system [60]. These interfaces include a Management WebGUI and a REST API providing XML based monitoring and management functionality [60]. By utilizing the RECS®|Box Management API, all hardware components within the cluster are available as XML trees in the software. The API allows for retrieving data on the current power consumption of each node's individual hardware components. We extract the necessary information for GPU and CPU power consumption from the XML tree and update it every second for short-term experiments and every minute for long-running experiments. Simultaneously, we keep track of the number of measurements to calculate the average power consumption.

5.3.4 Investigating the Performance of GPU Training vs. CPU Training for LightGBM: Task 2

A performance and efficiency analysis comparing the usage of GPUs and CPUs for training machine learning models with high-dimensional data and small sample sizes was the objective of Task 2 in WP2. The primary focus was to investigate the impact of communication overhead. To accomplish this, we utilized the Python library

`LightGBM` 3.3.5, which provides both GPU and CPU versions for the model training process [48][64]. But during the calculations on the GPU, we encountered a memory leak issue in the library, resulting in the termination of the program (see Figure 5.1). This issue persisted even when working with a small number of only 49 features. We observed this behavior across all three implemented feature selection methods: Random Forest, Extra Trees, and gradient-boosted decision trees. Notably, these feature selection methods successfully ran in parallel on the GPU. This indicates the potential for nested parallel feature selection methods and model training. Unfortunately, no calculation with a significant feature size was able to complete successfully. Training a large number of models without freeing memory exhausted the GPU's memory resources [18]. The number of models to be calculated for each feature selection method is given by:

outer cross-validation iterations * (*inner cross-validation iterations* * *hyperparameter combinations* + *outer cross-validation iterations*)

In our test scenario, this value was calculated as $6 * (5 * 100 + 6) = 3036$. So we expected to train a total of 3036 models, considering 100 trials for each hyperparameter optimization, six outer cross-validation iterations, and five inner cross-validation iterations. However, due to the memory not being released after training, the GPU reached its memory limit (see Figure 5.1).

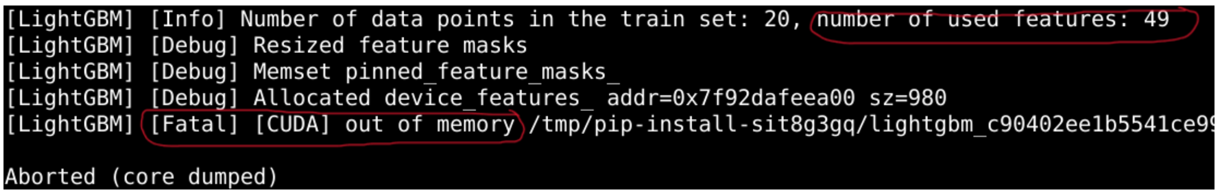

```
[LightGBM] [Info] Number of data points in the train set: 20, number of used features: 49
[LightGBM] [Debug] Resized feature masks
[LightGBM] [Debug] Memset pinned_feature_masks_
[LightGBM] [Debug] Allocated device_features_ addr=0x7f92dafeea00 sz=980
[LightGBM] [Fatal] [CUDA] out of memory /tmp/pip-install-sit8g3gq/lightgbm_c90402ee1b5541ce9

Aborted (core dumped)
```

Figure 5.1. `LightGBM` *memory leak in "CUDA" version*

To nevertheless investigate the communication overhead, we reduced the number of trials for each hyperparameter optimization to an unrealistic value of 15 trials. With this decrease in trials, we could demonstrate that training on the GPU (Tesla P100) is less efficient than training on a relatively small number of 6 CPU cores (Intel(R) Xeon (R) E-2176M CPU@2.70GHz) for the tested data shapes. Table 5.1 shows the measured energy consumptions and runtimes for the feature selection method `LightGBM` Extra Trees. The number of samples remained stable at 30 for all experiments. A seed for training and hyperparameter optimization was set, and all hyperparameter values were kept equal to ensure comparability. Only the number of features and the training hardware were varied. The power consumption was measured approximately every second. The duration was multiplied by the average power consumption to determine the energy consumption.

The communication overhead for the data shapes considered when using the GPU for model training is higher than the benefit for acceleration. However, the runtime difference between CPU and GPU training becomes smaller the higher the feature count is. At the same time, the total energy consumption also increases when using the GPU. But even though the communication overhead is higher and using the GPU

500 features, 15 trials HPO	CPU training	GPU training
duration (seconds)	4.27	7.93
average PEG card power consumption (W)		32.27
average node power consumption (W)	56.86	58.56
average total power consumption (W)	56.86	90.84
total energy consumption (Joule)	**242.83**	**720.37**

5000 features, 15 trials HPO	CPU training	GPU training
duration (seconds)	34	58.43
average PEG card power consumption (W)		33.15
average node power consumption (W)	54.52	51.72
average total power consumption (W)	54.52	84.87
total energy consumption (Joule)	**1856.21**	**4958.93**

50000 features, 15 trials HPO	CPU training	GPU training
duration (seconds)	1031.35	1270.28
average PEG card power consumption (W)		37.38
average node power consumption (W)	38.49	38.73
average total power consumption (W)	38.49	76.1
total energy consumption (Joule)	**39696.33**	**96673.13**

Table 5.1. GPU vs. CPU model training for `LightGBM`

is less energy efficient, it might make sense to use existing GPUs while running CPU-based computations in parallel on the remaining CPUs. This could make efficient use of the available heterogeneous hardware, which consumes energy in standby mode anyway.

5.3.5 Benchmark Energy Consumption and Duration

Our aim is to evaluate the performance and effectiveness of a parallelized version utilizing standard parallelization methods that are readily available in comparison to our optimized version previously described in Chapter 5.2. To achieve this, we employ an internally provided parallelized version offered by the libraries under study. We explore the parallelization capabilities of `Optuna` and two `scikit-learn` libraries as well as the CPU based `LightGBM` library by manipulating the respective parameters, which allow control over the level of parallelism.
The `scikit-learn` library offers several estimators and utilities that parallelize computationally expensive operations across multiple CPU cores. However, the documentation regarding the specifics of where and how parallelization occurs is lacking [13]. It is acknowledged that the documentation in this regard is currently insufficient. Nevertheless, our observations indicate that parallelism in the LASSO Regression estimator can be controlled through the global setting of the number of `OpenMP` threads. As for the Random Forest and Extra Trees classifiers, we set the number of parallel jobs using the `n_jobs` parameter, with a value of -1 indicating the use of all available processors [16] [14]. Further information on this parameter can be found in the Glossary [12], which defines that setting `n_jobs` to -1, all CPUs are utilized, and even when `n_jobs` is set to 1, low-level parallelism through `Numpy` and `OpenMP` might still be employed under certain configurations. Consequently, managing and controlling parallelism in `scikit-learn` libraries can be challenging.
In contrast, `LightGBM` provides a well-documented parameter to control parallelization [4]. When set to zero, the default number of threads defined by `OpenMP` are automatically utilized. To achieve optimal speed, the documentation recommends configuring this parameter to match the number of physical CPU cores, rather than the number of threads. Since most CPUs employ hyper-threading to generate two threads per core, the latter option would not yield the optimal results. Furthermore, it is worth noting that the value should not be excessively large when dealing with smaller data sets. It is documented, that it is normal for task managers or similar CPU monitoring tools to report underutilization of CPU cores in such scenarios.
Similarly, hyperparameter optimization in the context of our study utilizes the `n_jobs` parameter, which is also set to -1 [6]. This value corresponds to the number of available CPUs. However, it is important to note that this form of parallelization relies on threading and may encounter limitations due to Python's Global Interpreter Lock (GIL). The documentation advises the use of process-based parallelization if the target function is CPU bound.

Our experimental results comparing the different parallelization approaches show a remarkable speedup and improvement in energy efficiency achieved by our optimized version. Considering a data set with 500 features and 30 samples, our optimized solution achieves a substantial time reduction of 2 days and 47 hours. Moreover, we observe a significant decrease in energy consumption by 5.5 megajoules. These outcomes correspond to an acceleration factor of 13.9X and an energy saving

500 features	Standard	Optimized Parallelization
duration (d:h:m:s)	2:4:33:46	0:3:46:46
duration (seconds)	189226	13606
average total power consumption (W)	40,25	155,56
total energy consumption (Joule)	7616346.5	2116549.36
total energy consumption (MJ)	**7,62**	**2,12**

Table 5.2. Standard versus optimized feature selection workflow.

of 72%. Detailed results can be found in Table 5.2. These findings underscore the need for further research and improvements in the field of parallel computing for individual machine learning applications.

6 Exploitation

The generated impact during the project run-time is the publication of two open source projects. To ensure the sustainability of the work beyond the funding period, all developed artifacts are public and made available to the open source community. This has the general advantages associated with open source projects, as discussed in the Management Report's impact and sustainability chapter (Chapter 4.3 KER Overview).
Additionally, a teaching module for ensemble feature selection is planned to be developed, providing practical applications derived from the project's outcomes. To facilitate user-friendly use of the published software, we also created a prototype of a graphical user interface during the VEDLIoT project. Furthermore, the gained insights will be disseminated through one or more scientific papers.
However, there are potential risks associated with the exploitation of the solution. The complex workflow requires a certain level of expertise, and the research field of biomarker pilot studies is highly specialized, limiting the number of potential users. In contrast, the accelerated Yeo-Johnson transformation, on its own, has a broader scope of application, as it is a more frequently employed method. The limitation, in this case, lies in the utilization of FPGAs as specialized hardware.

7 Summary and next steps

This final technical report first provides an executive summary that offers a concise overview of the research findings. Within the introduction in Chapter 2, the background of bioinformatics is explored, explaining its relevance and interconnectedness with the Internet of Things (IoT). In addition, the introduction addresses feature selection, a key aspect of the research project. To enhance the feature analysis, the report introduces the concept of ensemble feature selection. Chapter 3 describes the feature selection workflow and provides a step-by-step overview of the process for removing irrelevant features. The report then continues with the development for WP1 (Yeo-Johnson power transformation on FPGA). Chapter 4 explains the Yeo-Johnson transformation and addresses the concept of skewness of a distribution. In addition, the standard Python library `scikit-learn`, in particular the `PowerTransformer` component, is introduced. Subsequently, the implementation and optimization of a new C library (Task 2 of WP1) and the subsequent FPGA implementation (Task 3 of WP1) are described in detail. Finally, the section concludes with an evaluation and characterization of the obtained results. Further on, the Accelerating Overarching Bioinformatics Workflow (WP2) is discussed. In Chapter 5, at first, the Python libraries used in the acceleration process are presented. Then, the accelerated workflow (Task 1 of WP2) is presented in detail and an in-depth analysis of its implementation is given. As for WP1, an evaluation and characterization of the results present the research findings. Finally, Chapter 6, Exploitation, provides insights into the potential practical applications and benefits arising from the research project.

The project results suggest several options for further research that could be explored. One possibility is to delve into the parallelization of hyperparameter optimization (HPO), with the goal of discovering more suitable libraries or adapting existing ones. Moreover, a deeper investigation into scheduling techniques could yield valuable insights and optimization potential. Additionally, improvements could possibly be achieved by storing untransformed data only once in the data store and accessing it through slicing operations. Another aspect concerns model training on the GPU. Since the performance analysis comparing GPU and CPU usage in model training for machine learning models with small sample sizes revealed a memory leak issue in the GPU CUDA version of `LightGBM` 3.3.5., further investigation and resolution of the memory leak issue are required to fully exploit the potential of GPU usage in a heterogenous hardware context.
Comparative analyses of individual feature selection methods are also interesting areas to investigate enabled by the project's outcome. It reveals new research options to evaluate the robustness of ensemble feature selection methods as well as single feature selection methods. However, feature selection robustness and classification performance should always be studied together, as robustness alone cannot compensate for lower classification performance [55].
To enhance usability, the implementation of a graphical user interface (GUI) for parameter validation and parallelization control would be beneficial. Furthermore, developing a training module that educates users on system usage and provides an explanation of its application in biological or medical settings would improve accessibility for these users.

8 References

[1] Christmann informationstechnik + medien. https://christmann.info/. Date of access: 28.06.2023.

[2] ctypes - a foreign function library for python. https://docs.python.org/3/library/ctypes.html. Date of access: 29.11.2022.

[3] Lightgbm documentation. https://lightgbm.readthedocs.io/en/latest/Python-Intro.html. Date of access: 28.06.2023.

[4] Lightgbm: Parameters. https://lightgbm.readthedocs.io/en/latest/Parameters.html. Date of access: 28.06.2023.

[5] Optuna: Easy parallelization. https://optuna.readthedocs.io/en/stable/tutorial/10_key_features/004_distributed.html. Date of access: 28.06.2023.

[6] Optuna: Optimize study. https://optuna.readthedocs.io/en/stable/reference/generated/optuna.study.Study.html#optuna.study.Study.optimize. Date of access: 28.06.2023.

[7] Ray anti-pattern: Over-parallelizing with too fine-grained tasks harms speedup. https://docs.ray.io/en/latest/ray-core/patterns/too-fine-grained-tasks.html. Date of access: 28.06.2023.

[8] Ray objects. https://docs.ray.io/en/latest/ray-core/objects.html. Date of access: 28.06.2023.

[9] Ray pattern: Using nested tasks to achieve nested parallelism. https://docs.ray.io/en/latest/ray-core/patterns/nested-tasks.html. Date of access: 28.06.2023.

[10] Ray scheduling. https://docs.ray.io/en/latest/ray-core/scheduling/index.html. Date of access: 28.06.2023.

[11] Ray tune: Hyperparameter tuning. https://docs.ray.io/en/latest/tune/index.html. Date of access: 28.06.2023.

[12] scikit-learn - glossary n_jobs. https://scikit-learn.org/stable/glossary.html#term-n_jobs. Date of access: 28.06.2023.

[13] scikit-learn - parallelism, resource management and configuration. https://scikit-learn.org/stable/computing/parallelism.html. Date of access: 28.06.2023.

[14] Scikit-learn extratreesclassifier. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html. Date of access: 28.06.2023.

[15] Scikit-learn lasso. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html. Date of access: 28.06.2023.

[16] Scikit-learn randomforestclassifier. `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`. Date of access: 28.06.2023.

[17] Apparent object store memory leak, 2023.

[18] Memory leaks / is better memory management possible?, 2023.

[19] Thomas Abeel, Thibault Helleputte, Yves van de Peer, Pierre Dupont, and Yvan Saeys. Robust biomarker identification for cancer diagnosis with ensemble feature selection methods. *Bioinformatics (Oxford, England)*, 26(3):392–398, 2010.

[20] Abhigyan. Feature selection for dimensionality reduction(embedded method). *Analytics Vidhya*, 5/7/2020.

[21] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. *CoRR*, abs/1907.10902, 2019.

[22] Ali Anaissi, Paul J. Kennedy, and Madhu Goyal. Feature selection of imbalanced gene expression microarray data. pages 73–78.

[23] Anyscale. Ray. `https://github.com/ray-project/ray`, 2023. Date of access: 2023-06-13.

[24] Afef Ben Brahim. Stable feature selection based on instance learning, redundancy elimination and efficient subsets fusion. *Neural Computing and Applications*, 33(4):1221–1232, 2021.

[25] Afef Ben Brahim and Mohamed Limam. Robust ensemble feature selection for high dimensional data sets. pages 151–157.

[26] V. Bolón-Canedo, N. Sánchez-Maroño, A. Alonso-Betanzos, J. M. Benítez, and F. Herrera. A review of microarray datasets and applied feature selection methods. *Information Sciences*, 282:111–135, 2014.

[27] Verónica Bolón-Canedo and Amparo Alonso-Betanzos. Ensembles for feature selection: A review and future trends. *Information Fusion*, 52:1–12, 2019.

[28] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001.

[29] Gavin C. Cawley and Nicola L. C. Talbot. On Over-fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation. *Journal of Machine Learning Research*, 11(70):2079–2107, 2010.

[30] Didier Concordet, Anne Geffré, Jean-Pierre Braun, and Catherine Trumel. A new approach for the determination of reference intervals from hospital-based data. *Clinica Chimica Acta*, 405(1-2):43–48, July 2009. Thanks to Elsevier editor. The original PDF of the article is available at http://bit.ly/c7q1eD.

[31] Takashi Daimon. *Box–Cox Transformation*, pages 176–178. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[32] Dimitrios Effrosynidis and Avi Arampatzis. An evaluation of feature selection methods for environmental data. *Ecological Informatics*, 61:101224, 2021.

[33] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely Randomized Trees. *Machine Learning*, 36:3–42, 2006.

[34] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 1, pages 278–282 vol.1, 1995.

[35] Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[36] Nazrul Hoque, Mihir Singh, and Dhruba K. Bhattacharyya. Efs-mi: an ensemble feature selection method for classification. *Complex & Intelligent Systems*, 4(2):105–118, 2018.

[37] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30:3146–3154, 2017.

[38] Utkarsh Mahadeo Khaire and R. Dhanalakshmi. Stability of feature selection algorithm: A review. *Journal of King Saud University - Computer and Information Sciences*, 34(4):1060–1073, 2022.

[39] Igor Kononenko. Estimating attributes: Analysis and extensions of relief. volume 784, pages 171–182.

[40] Liam Li, Kevin Jamieson, Afshin Rostamizadeh, Ekaterina Gonina, Moritz Hardt, Benjamin Recht, and Ameet Talwalkar. A System for Massively Parallel Hyperparameter Tuning. In *Proceedings of Machine Learning and Systems 2 (MLSys 2020)*, volume 2, pages 230 – 246. mlsys.org, October 2020. arXiv: 1810.05934v5.

[41] Scott M. Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M. Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. From local explanations to global understanding with explainable AI for trees. *Nature Machine Intelligence*, 2(1):56–67, 2020.

[42] Shuangge Ma and Jian Huang. Penalized feature selection and classification in bioinformatics. *Briefings in Bioinformatics*, 9:392–403, September 2008.

[43] Charu Makhijani. Ensemble learning techniques - towards data science. *Towards Data Science*, 29.9.2020.

[44] Charu Makhijani. Advanced ensemble learning techniques - towards data science. *Towards Data Science*, 5/10/2020.

[45] Wilson E. Marcilio and Danilo M. Eler. From explanations to feature selection: assessing SHAP values as feature selection mechanism. In *2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 340–347, Recife/-Porto de Galinhas, Brazil, 2020. IEEE.

[46] Layla Matter. The effect of outlier on lasso estimators and regressions. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, 12:2480–2490, 05 2021.

[47] Sigrun May, Sven Hartmann, and Frank Klawonn. Combined pruning for nested cross-validation to accelerate automated hyperparameter optimization for embedded feature selection in high-dimensional data with very small sample sizes. *CoRR*, abs/2202.00598, 2022.

[48] Microsoft. Lightgbm gpu tutorial, 2023.

[49] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, William Paul, Michael I. Jordan, and Ion Stoica. Ray: A distributed framework for emerging AI applications. *CoRR*, abs/1712.05889, 2017.

[50] Klaus Nordhausen. The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition by Trevor Hastie, Robert Tibshirani, Jerome Friedman. *International Statistical Review*, 77:482–482, 2009.

[51] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[52] Nicholas Pudjihartono, Tayaza Fadason, Andreas W. Kempa-Liehr, and Justin M. O'Sullivan. A review of feature selection methods for machine learning-based disease risk prediction. *Frontiers in Bioinformatics*, 2, 2022.

[53] Yanjun Qi. Random Forest for Bioinformatics. In Zhang Cha and Yunqian Ma, editors, *Ensemble Machine Learning: Methods and Applications*, pages 307–323. Springer New York, NY, Boston, MA, 1st edition, 2012.

[54] Papia Ray, S. Surender Reddy, and Tuhina Banerjee. Various dimension reduction techniques for high dimensional data analysis: a review. *Artificial Intelligence Review*, 54(5):3473–3515, 2021.

[55] Yvan Saeys, Thomas Abeel, and Yves van de Peer. Robust feature selection using ensemble feature selection techniques. volume 5212, pages 313–325.

[56] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[57] Rodrigo Torres and Robert L. Judson-Torres. Research Techniques Made Simple: Feature Selection for Biomarker Discovery. *Journal of Investigative Dermatology*, 139:2068–2074.e1, 2019.

[58] Andrius Vabalas, Emma Gowen, Ellen Poliakoff, and Alexander J. Casson. Machine learning algorithm validation with a limited sample size. *PLOS ONE*, 14:e0224365, 2019.

[59] Kai Vahldiek, Libing Zhou, Wenfeng Zhu, and Frank Klawonn. Development of a data generator for multivariate numerical data with arbitrary correlations and distributions. *Intell. Data Anal.*, 25(4):789–807, 2021.

[60] Micha vor dem Berge, Stefan Krupop, and Jens Hagemeyer. D2.5 – final report on next-generation microserver system development. Technical report, H2020 Project M2DC, 2019. available at `https://m2dc.eu/en/documents/`.

[61] Makoto Yamada, Wittawat Jitkrittum, Leonid Sigal, Eric P. Xing, and Masashi Sugiyama. High-dimensional feature selection by feature-wise kernelized lasso. *Neural computation*, 26(1):185–207, 2014.

[62] Li Yang and Abdallah Shami. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415:295–316, November 2020.

[63] In-Kwon Yeo and Richard A. Johnson. A new family of power transformations to improve normality or symmetry. *Biometrika*, 87(4):954–959, 12 2000.

[64] Huan Zhang, Si Si, and Cho-Jui Hsieh. Gpu-acceleration for large-scale tree boosting, 2017.

Very Efficient Deep Learning in IoT

ICT-56-2020 - Next Generation Internet of Things

Final technical report Edge4iwelli VEDLIoT Open

Document information	
Project website	https://www.vedliot.eu
Dissemination Level	PU
Nature	R
Contractual Deadline	30.06.2023
Author	Homer Papadopoulos
Contributors	Imirtziadis Theodoros

The VEDLIoT Project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957197.

Changelog		
Version	**Date**	**Type**
V 1.0	30/06/2023	Finalisation

Table of contents

Executive summary

The iwelli IoT care package is a comprehensive toolset of systems and applications designed to promote healthcare services at home, aiming to reduce the overall cost of healthcare. The need for remote healthcare monitoring has become increasingly apparent, however, relying solely on cloud services for this purpose poses security, latency, and environmental concerns. To address these challenges, the Edge4iwelli project integrates edge computing systems into the iwelli IoT care package.
The Edge4iwelli project focuses on transferring and executing AI and analysis models in-house, minimizing the reliance on cloud services. By embedding edge computing systems and advanced AI modules into the existing iwelli IoT care package, a more secure and efficient IoT care platform is created. This platform enables proofs-of-concept in edge computing, optimization, and machine learning in home contexts, specifically for healthcare services.
One of the key developments within the project is the Edge4iwelli smart health mirror.
The Edge4iwelli project presents a significant step towards reducing healthcare costs by promoting care at home. The smart health mirror, along with the secure and efficient IoT care platform, enables remote healthcare monitoring and interaction, empowering individuals to manage their health effectively while benefiting from the convenience and comfort of their own homes.

1 Introduction

In the era of the COVID-19 pandemic, the global healthcare landscape is undergoing a profound transformation. Strategies aimed at reducing the overall cost of healthcare have become increasingly important, and one such approach involves promoting care at home rather than in hospitals. With this goal in mind, we have developed the iwelli IoT care package, a comprehensive toolset of systems and applications designed to support healthcare services within the comfort of one's own home.

However, the iwelli IoT care package relies heavily on cloud services, which present inherent challenges such as security concerns, delays, and environmental impacts. To address these issues and further enhance the efficiency of home care services, we have developed the Edge4iwelli platform. This innovative solution leverages edge computing systems and advanced AI modules to minimize reliance on the cloud, ensuring faster response times, enhanced security, and reduced environmental footprint.

At the core of the Edge4iwelli platform lies the Edge4iwelli Smart Health Mirror. This interactive device seamlessly transforms an ordinary mirror into a sophisticated tool for healthcare monitoring. By placing a semi-transparent sheet of glass over a digital screen and connecting it to a computer, we have created a unique hardware configuration that enables the mirror to serve as an intelligent interface between users and the healthcare system.

The Smart Health Mirror comprises three main components. Firstly, it features a programmable mirror with a digital display, providing users with easily accessible and relevant information. Secondly, it incorporates two cameras—an RGB webcam and a hyperspectral camera—that enable real-time analysis of video data containing users' faces and bodies, facilitating automatic monitoring of physiological parameters. Finally, an edge computing system, wirelessly connected to an internet hub, performs the necessary video analysis, saving the data for further analysis in a secure local database.

Privacy is of utmost importance, and to ensure user confidentiality, the Smart Health Mirror allows users to control the camera and microphone, granting them the ability to close or open these features as desired. Moreover, the system's form factor is compact, the costs are low, and its energy efficiency is remarkable, with power consumption limited to just 15W.

Through the integration of cutting-edge technologies in optics, machine learning, computer vision, and medicine, the Edge4iwelli Smart Health Mirror represents a significant advancement in remote healthcare monitoring. Its implementation holds tremendous potential for the elderly population, particularly those living independently, as it empowers them to monitor their physiological parameters easily and automatically.

By embedding edge computing systems and advanced AI modules into the iwelli IoT care package, we have created a more secure, efficient, and cost-effective IoT care platform. This platform, with the Smart Health Mirror as its centerpiece, enables the practical utilization of AI and edge computing technologies within the context of smart homes and home care services.

The Edge4iwelli project stands as a testament to our commitment to innovation and our dedication to revolutionizing healthcare services. Through the development of new tools and application libraries, we are opening doors to diverse capabilities in AI and edge computing, propelling the field of home care services into a new era of improved efficiency, accessibility, and patient-centric care.

2 Idea and architecture

The software and hardware components introduced within the Edge4iwelli system architecture are analyzed in detail below. See figure below for the list of all the components.

2.1 Hardware Components

The Edge4iwelli health smart mirror is made from an LCD display covered with a semi-transparent glass. The screen connects to a logic unit/microcomputer that manages all the functions of the mirror via a GNU/Linux operating system. We utilize and experiment with different designs for the microcomputer using jetson nano and/or Raspberry Pi 4 with two Google Colab accelerators or a combination of the two systems. We also intend to experiment with the Nvidia Jetson system and our aim is to develop our own logic unit combining different systems.

Following are the specification for hardware devices which are needed for EDGE4IWELLI system

Table: Hardware choices for the health smart mirror

Name	*Mirror Hardware*
Two Way mirror	Basically, a mirror is a smooth polished surface where the image is created by the refection. In this work, a two-way mirror is used in which one side is transparent and the other side shows its reflection. The concept of using a two-way mirror 70/30 is similar

	to using a usual mirror and that same mirror also acts as a functional object. For the first prototype within the realm of the Edge4iwelli pilot we will work with the following Mirror Dimensions: Height (without the IP camera installed): 44cm Width: 30cm Depth: 12,5cm Weight : 5,8kg
Functional & non-functional description	The health smart mirror represents the device to allow the end user to communicate with the AI systems, collect the data and analyse the data and send the high level events to the med.iwelli.com server so that other persons (doctor, carers or relatives) can have access showing and giving awareness information
Hardware requirements	

No.	Name	Specifications/Model Number	Estimated Price
1	**Display**	In order to present the basic information a display monitor will be used. An HDMI display - LCD screen size ≈ 10'', display resolution ≥ 1280 x 800, will be considered for this implementation.	€70
2	**Frame**	Metallic frame or wooden frame (eg IKEA Ribba) for the prototypes	€40
3	**Glass 70/30**	1 square meter	€180
4	**Micro-computer**	Nvidia Jetson/nano Or Nvidia Orin nano	€120 €400
5	**Microphone**, speakers	Mini usb microphone for raspberry WM8960 Audio HAT Module for Raspberry Pi 4	Up to €30
6	**RGB camera (Manual lens cover for privacy.)**	TP-Link SC3130G, Logitech C922 Pro Stream Web Camera Full HD 1080p Raspberry Pi Camera Module 5MP 72° - Day & Night Vision or similar	Up to € 30
7	IR camera	Raspberry Pi Infrared Camera for temperature and other vitals	€20

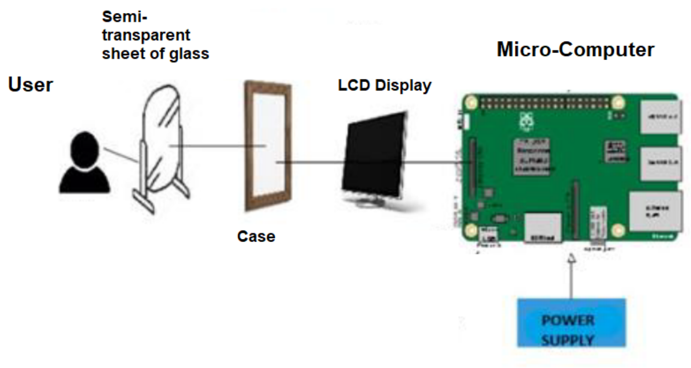
8	**Supportive device to the Micro Computer unit**	Raspberry 4	€70
9	Disposables units	Eg cables	€10
10	Extra communications sensors	To be defined	€40
11	Microcontroller	ESP32	€10
12	IR sensor	for presence detection	Up to €4

Communication		WiFi, Bluetooth, Zigbee
Involved systems		Local DB, security gateway
Execution		Through power on (hardware)
Data going in		Video imaging, audio, information from Web
Data going out		High level events regarding the health status of the end user
User Interface		Magic Mirror 2 customized UI with extra modules
Available Products	**Indicative Open Source, Commercial**	https://stanhom.com/, https://verconsmartmirror.com/projects/, https://helentronica.com/2016/01/11/magic-mirror-with-motion-detector/, https://www.care-os.com/, https://www.twowaymirrors.com/
	Adaptation	We will create our own prototype
Feasibility		Yes
Risks		Personal data security. Poor detection capabilities in dim light. There is a need for adequate lighting in order to allow the correct recording of the user's face and the subsequent reliable extraction from it of the necessary features as events for the system. The camera should be positioned in a place that should allow different lighting conditions from extremely dim to full overhead and daytime light. We have to set up the Smart Mirror lighting conditions eg with extra leds so that in dim lighting conditions it will provide reliable results. This requirement should be noted as a guideline for the installation of the camera and the mirror.
Available Standards		FHIR, communication standards

Possible reasons for crashing	Falling on the floor or similar Firmware faults
In Case of failure	Restart device, Replace device
Remarks	The installation of the mirror will be easy and different versions will be provided for different needs

On a basic implementation, the screen can display real-time information about: Time, weather, calendar, alarm, news update, music, and traffic update. For the authenticated users the mirror will display extra information like: email, To-do list, phone notification.

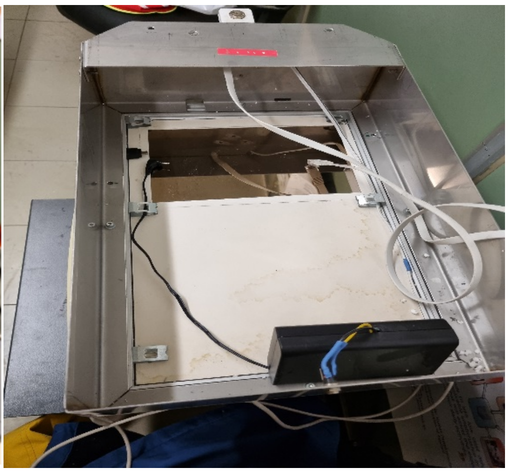
Text appears on a portion of the rear-view mirror to aid information. The edge4iwelli smart health mirror goes one step further utilizing edge computing and monitoring systems with personalized information and computer vision to help individuals achieve their health goals and have a more significant role in healthcare.
The edge4iwelli smart health mirror intends to embed multimodal sensors—multiple cameras (RGB and IR), motion detection, microphones, speakers—as well as software based on artificial intelligence (AI) into the original design and include the capacity to capture and communicate with multiple sources of data, linking them to the broader ecosystem of smart products and the cloud.
The Edge4iwelli smart mirror is able to capture physiological measurements non-invasively and to create interactive capabilities based on tracking and recognizing gestures. One of our objectives is to keep the architecture of the Edge4iwelli smart mirror open so to enable stakeholders to collect large quantities of biomedically relevant data, and to develop a wide range of applications that address many healthcare challenges.



Edge4iwelli health smart mirror design

Below the prototypes we are working on to set up the final devices are presented:

The prototype where we are doing the implementations



2.2 Data Collection and Data Processing Layer

The final health smart mirror should be able to provide multiple services that could be relevant for different medical use cases. We adopt a modular architecture for the software part of the mirror something that enables us to apply different features on the health smart mirror that complement its healthcare role, such as time display, news, weather forecasts, environmental data, face recognition and other.

Our intention is to embed in the health smart mirror different open-source services linked to healthcare e.g., mood detection as a measure of mental health and facial recognition for authorization and personalization.

Our main aim is to create a health smart mirror architecture that can be tuned by a user based on his/her needs.

The main health data the health smart mirror intends to monitor concern emotional states and anxiety levels, infer an individual's risk for cardio-metabolic diseases, other vital signs etc. all of which data can be captured by standing in front of the mirror.

Edge4iwelli believes that the recent advances in machine intelligence are creating the opportunities to develop a smart health mirror that goes beyond a prototype. Computer vision driven by deep learning algorithms can help inexpensive cameras achieve image acquisition performances comparable with high-end devices, enabling new opportunities to embed medical-quality image acquisition in at-home devices, such as mirrors. We believe that beyond the emotion and cardiovascular vitals recorded by remote-PPG there are other opportunities in data collection. For example, ophthalmology and dermatology can change the traditional patient-physician interaction since retinal photographs analyzed by deep learning networks can measure potential cardiovascular risk factors and systolic blood pressure, and identified diabetic retinopathy and diabetic macular edema while neural networks can classify skin images that indicate malignant growths.

For the purpose of face recognition, we will use an RGB camera and an IR camera. The mirror leverages face recognition technology to match the user's face to their profile. The user creates a profile the first time his face is detected by the mirror and is not recognized as an existing user. For the face recognition service we will try :

- And the face-api.js which has a JavaScript API for face detection and face recognition in the browser implemented on top of the tensorflow.js core API (tensorflow/tfjs-core).

All these models will run in the node Js server we will set up locally on the edge4iwelli micro-computer.

Furthermore, the health smart mirror is able to recognize human affects through the analysis of facial behavior. Specifically, given a single face image, Face API model predicts one discrete emotional state among the six basic, happiness, sadness, surprise, anger, fear, and disgust. The library also provides face detection model, a Point Face Landmark Detection Model and a Face Recognition Model. Furthermore a part of the Video analysis is responsible for the elderly user's facial analysis in order to extract parameters such as pupils' equality, focusing ability, buccal angle, and skin colour. The raw video feed is only available for analysis from the model and only the conclusions from video can be sent off-site.

To utilize the features of the voice-based activity, a microphone is needed in the designed model. A speaker is also attached to the system to receive feedback. However, all the output results can be received through the speaker. Although we can use the Google Cloud speech-to-text API for the speech to text conversion we will try:

- The Voice Assistant Rhasspy tool (https://rhasspy.readthedocs.io/en/latest/) that employs the speech to text transcription service, the intent recognition service and the intent handling service. For speech recognition, a model with English and Greek languages will be trained with DeepSpeech which is a speech recognition engine based on a trained model using machine learning techniques (https://github.com/mozilla/DeepSpeech

- and the transcription model whisper (the tiny model with 39 Million Parameters) https://github.com/openai/whisper. The Whisper models are trained for speech recognition and translation tasks, capable of transcribing speech audio into the text in the language it is spoken. Although it not working with streaming audio we believe that the model with the 3.75 second timeslot can give us enough quality for the command part.

The next version of the health smart mirror will allow the end user to control IoT devices for home automation and display information from these devices. For that an ESP32 micro-controller will be in charge of the home control.

Zig-ah-zig-ah! (zzh!)

The ZigBee technology seems to be the most appropriate one for communicating the IoT devices in house since it guarantees a low cost and low energy consumption, For that we will use the open-source project, known as ZigBee2MQTT[1] that provides a way to interconnect devices of different manufacturers with ZigBee connectivity through MQTT event bridge[2]. For the communication bridge we will use the USB stick zig-ah-zig-ah! or zzh! device[3] that is compatible with ZigBeeMQTT. The device will provide the ZigBee communication interface to the computing node, turning it into a gateway, therefore working as a protocol coordinator.

Data Fusion: The Data fusion component infers high-level events based on both video, audio and NLP analysis of the text transcribed by the end user. A stream of time-stamped low-level events detected on sensor data. The system will be executed when the Edge4iwelli system is started. It checks periodically a local Database for low-level events in order to perform high-level event recognition. We will implement it in YAP Prolog platform embedded in Ubuntu Linux. For this module available CPU is not enough so that the AI accelerators will be used for. The high-level events will be sent to the EHR using the FHIR protocol.

2.3 Security Layer

In the Edge4iwelli sensitive data will be transmitted and stored so as the mission of the system can be fulfilled. In order to provide the necessary services, electronic information has to be processed on a daily basis. Without the implementation of appropriate security measures and controls, the data are subject to potential damage and information could be compromised. This section defines the basic security policies to be implemented within the technology architecture of the Edge4iwelli system.

Edge4iwelli operational process:

The health smart mirror has three different states:
1) off-state: the mirror behaves like a standard mirror.
2) stand-by-state: the mirror is ready to be activated. The mirror, through its proximity sensor and

[11] www.zigbee2mqtt.io
[2] https://mqtt.org
[33] www.zigbee2mqtt.io/information/supported_adapters.html

microphone, senses the surrounding environment and detects the presence of human(s) in front of the mirror, detects events of the user eg crying or laughing and helping with recognition of the events of monologue (the subject speaking to him/herself) and helps record the speaking.
3) on-state: the mirror reacts to the surrounding environment by recognizing human identities and monitoring
emotions.

The operating process of the health smart mirror is following:

- The mirror will perform the face detection.
- Based on the recorded data there will be a user authentication check.
- If the user is unkown then basic features for general users will be shown.
- If the user is authenticated then more personalized information will be displayed in the mirror.

Local database:

We will apply a local database server that will be responsible for storing the monitoring from the mirror data locally within the home. The database will be a lightweight server with performance sufficient to handle possible peak traffic and will consists of:

- A conceptual model (ontology) for data
- HTTP server providing network access to the database
- A set of client APIs for different programming languages (wrapping HTTP communication with programmatic functions)
- Self-cleaning function, configurable (rule-based) to define how long data should be kept in the database (may be different for different kinds of data). Deleted data can be archived into files.

The mirror logs sensors data and information processed by the monitoring models. In particular, the information logged and stored in the local database are:

- daily emotions
- daily vitals
- daily mirror usage
- face recognition descriptors

This information is processed and sent to the personal record of the user over the cloud.

Threats and Vulnerabilities

There are some public concerns related to the use of the technology for data sharing in the health and personal data domains. These include the difficulty of respecting privacy and confidentiality when third parties have a strong interest in getting access to electronically recorded and stored personal health data, as well as the difficulty of ensuring the security of shared personal monitoring data. The threats to the security of personal health information that arise from the processing, the transmission and, in particular, from the necessary access to the personal records, include malware and database attack. Unauthorized access takes many forms (running a password-cracking application, sniffing/eavesdropping on network traffic, social engineering, etc.).

Security Measures

The basic security measures which will be implemented in the Edge4iwelli system are described below.

Virus Control

All servers eg Node server etc will have anti-virus software installed which will be kept up to date.

Protection of Data from Hardware Loss

Backups of data and system programs for the data stored from the mirror in house will be taken on a regular predetermined automatically. The exact backup system will be defined in the next steps of the mirror development.

Protection of Data from Unauthorized Access

The following are some of the countermeasures against unauthorized access:

- User Authentication: Authentication is required, e.g. to ensure, that the one standing in front of the Edge4iwelli mirror system is the intended user and not a visitor. The user needs an account to log in to the Edge4iwelli system. After successful registration, every time an existing user wants to have access to the data provided by the system, he or she must be authenticated. We will utilize the face recognition service for authenticating the user. For the web based personal records we are using utilize OAuth2 protocol.
- User Authorization
- Timeout Policy: When a User is inactive for a certain time, it should be logged out.
- Encryption: All communication must be encrypted using cryptographic protocols such as TLS/SSL
- The intel SGX enclave will be tested to support scenarios of trusted keys storage and exchange. We will use the Intel SGX to allocate private regions of memory, enclaves, to be protected from processes running at higher privilege levels. For that we will try to host a WebAssembly application in WAMR for key management.
- For the smart health mirror and the communciation of the high level events with the med.iwelli.com platforms and the inter-communcation of the data in house we will consider the VPN solution utilising the concept of security IoT gateway. The VPN solution shields the network from sniffing unencrypted data, and restricts unwanted communication to third-party servers. Besides securing the network, it also provides convenient features for secure communication among network devices over the internet. To enable these VPN connections we will work with the OpenVPN to access home network through the Wi-Fi Routers.
- Communication between devices: Edge4iwelli will apply wireless technologies like WLAN and Bluetooth. We will use the safe encryption standard for WLAN the WPA2 (Wi-Fi Protected Access) and for the bluetooth devices we will pair the devices before exchanging data.
- Local storage of sensor data: Data stored inside the home will be secured via cryptographic mechanisms.

A set of software libraries will be used for consistently applying the security policies throughout the Edge4iwelli system.

- A set of client APIs for communication with the Local database, supporting HTTPS.
- A set of client APIs for calling functions of the Server Web API, supporting HTTPS.
- Components handling authentication of software components, i.e. core services and applications

Privacy

The edge4iwelli is taking care of the potential privacy violations regarding information about an individual's health especially when the mirror is using cameras and microphones to collect data, as well as the transfer of these data to cloud storage systems. Privacy is important for the mirror's hardware and software design architecture. We implement a multi-tier privacy plan that includes different steps to process and encrypt data.

- The health smart mirror platform will proceed with the necessary actions to get a security clearance.
- The microcomputer will pre-process the data collected by the mirror locally (low level events), and then
- upload the processed information (high level events) to the cloud eg EHRs a secure and encrypted protocol.
- We will equip the health smart mirror with extra security system control eg a switch that will allow engaging or disconnecting embedded sensors.
- We will adopt GDPR compliant legal framework
- We will provide the ability to deactivate camera manually

2.4 Home-to-server upload

After starting up of the Edge4iwelli system at home, runs continuously. Most of time in stand-by mode, but wakes up at a predefined interval and performs the upload job.

All health-related data originated within home: Measurements, Person Events, Alerts will be stored and processed before uploading to the web. Home-to-server upload is an in-home software service (daemon-like) responsible for proactively uploading monitoring data and other relevant data (e.g. data-fusion produced high-level events) from the local database to the med.iwelli.com and to the mydata.iwelli.com servers. Allows configuration of uploading intervals and (rule-based) what data to upload. The API that will be used to upload the data will be the FHIR API.

Personal Health Record

The patients can store their data either in their mobile phone or upload their data in their own cloud space personal health record in the Fhir PHR mydata.iwelli.com portal. The med.iwelli.com and the mydata.iwelli.com platforms, based on artificial intelligence, allow patients to both obtain a personalised risk prediction and personalized guidance for their treatment plan management.

Electronic Health Record

The Fhir EHR med.iwelli.com platform is part of the iwelli.com ecosystem of applications enabling doctors to store heterogeneous data (vital signs from their patients, documents, pictures like MRI, photos of medical exams, document file types such as pdf, textual content like the medical history of the patient, financial records, audio files and many others).

Med.iwelli.com FHIR EHR

The med.iwelli.com is more than an EHR since it allows for more complex data to be captured and stored.

The doctors can access the med.iwelli.com cloud based intelligent EHR to review test results, lab results, allergies, prescriptions medications, immunization history, and other health information. They can analyze data with calculators and machine learning to understand recorded Data, link it to clinical outcomes and identify the health scores of their patients and identify patients risks by correlating collected data sources. They can schedule and manage appointments, including in-person visits and video visits. The med.iwelli.com addresses the health care challenges of patient monitoring and Risk-stratification and Decision Support.

2.5 Application Layer

Health monitoring measurements

The iwelli mirror utilises software AI models that can analyse video and monitor physiological among the others parameters of persons, through real-time analysis of video containing their bodies and faces, as captured by a common web camera and the IR camera. Given that a person stands relatively still in close distance of a common camera, the cameras of the mirror detects persons' pulse and breathing rate, pupil size, skin redness colour, eye blinking rate and duration and eye redness. We are using computer-vision and AI-powered technology for data acquisition, analysis, and interpretation of vital signs in real-time. We are using different monitoring methods like the remote photoplethysmography (rPPG) – a contactless optical measurement technique of recording skin blood pulsations at different vascular depths. The process we are using concerns:

- Visual Data monitoring: RGB and IR video of facial skin is captured from the cameras.
- Signal Processing: By employing data processing techniques, the captured images are converted. As a result, RGB color information is transposed into rPPG signal in order to extract a low-noise heart pulse signal for further analysis of every pixel of the skin image. Such an optimized model is a rich information source for the Machine Learning (ML) pipeline that analyzes each pulse wave's timing, shape, amplitude, and other features.
- AI analysis: Machine learning (ML) models use the extracted signals to precisely measure HR and other vital signs – the system is able to assess the risk of cardiovascular diseases (CVD) or those related to skin or eye health.

The monitoring techniques we are using together with the AI models allow us to capture and analyze the following parameters:

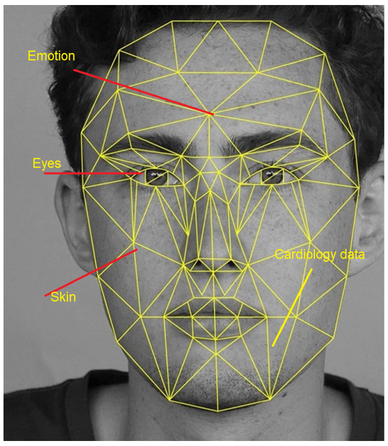
High TRL

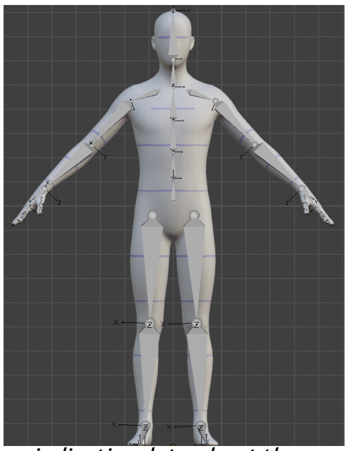
1. **Cardiovascular** parameters/vitals: Heart Rate (The number of times your heart beats), Blood Pressure (The pressure of blood of arteries), Heart Rate Variability (HRV- The variation between your heartbeats), Respiratory Rate (The number of breaths we take), pulse transit time, pulse, Temp
2. Emotion and mental markets: stress level (The level of stress we are experiencing), energy level, anxiety, basic emotions
3. Body Type, Shape – OpenCV (Yolo V4 to be tested)
4. Age, Gender, Height (ultrasonic sensors to be discussed)
5. Q&A with avatar (how to you feel? – Transformers). Following VEDLIOT project a voice assistant based only on local processing will be considered as well.

Low TRL.

The future features also include parameters like:

a. Dermatological: skin variation, skin type and colour, hydration, early prediction of skin diseases
b. Ophthalmological: abnormalities of the iris, inflammations, macular degeneration, anisocoria, eyes focusing, buccal angle
c. Mouth inspection – hygiene
d. Vitals: SpO2 (The amount of oxygen in our blood), Body Weight
e. IR camera – Temp and support other vitals – low TRL study





In the following tables we provide some indicative data about the monitoring data/events and their representation to the Edge4iwelli system.

Table. Measurement categories

Category	**Description**	**Arguments** (Additional arguments for all Measurements: sensor ID, timestamp, duration, confidence value)
	Physiological monitoring	
heart rate	The heart rate of a person has been measured	The person ID and the heart rate (beats per minute)
blood pressure	The blood pressure of a person has been measured	The person ID and the blood pressure (two mmHg values: systolic and diastolic)
body temperature	The body temperature of a person has been measured	The person ID and the temperature (degrees Celsius)
skin colour	The skin colour of a person's face has been evaluated	The person ID and the colour description
buccal angle	The buccal angle of a person has been evaluated as either normal or abnormal	The person ID and Boolean value ("true" means normal)
eyes focusing	The eyes focusing ability of a person has been evaluated	The person ID and the numeric value according to a scale
anisocoria	The pupils of a person have been detected to be equal or unequal	The person ID and Boolean value ("true" means equal)
	Emotion	
emotion	Elderly user has be detected to be in an emotional state	
laughing	Elderly user is laughing, for a duration	
crying	Elderly user is crying, for a duration	
monologue	Elderly user speaking to him/herself	
gesture	Elderly user performing a specific gesture, e.g. indicative of emotion	
	Objects	
lights on	A light has been turned on	
lights off	A light has been turned off	

Table. System event categories

Category	Description
Emergency alert sent	An emergency alert is issued to health professionals or other carers
communication event	A communication event between two system users took place, with its parameters recorded
system control command	Other kind of system control command issued
data access	A system user accesses privacy-sensitive information
system fault	A fault occurred in the system

Several state of the art machine learning and video analysis techniques are combined to analyse the video signal and estimate each one of these parameters.
Operation functionalities of the Smart Health Mirror:

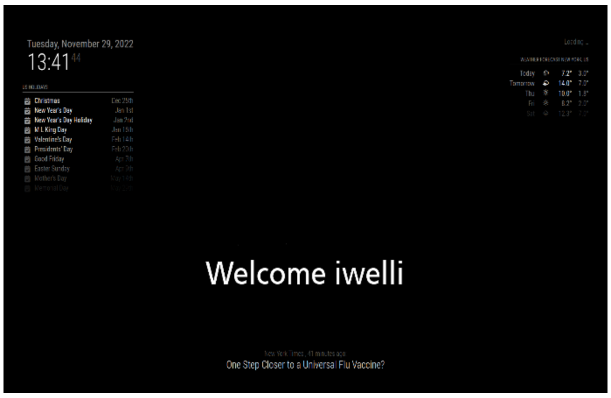
- Functionalities: No internet connection (https)
- Button for turning off the internet
- Due to privacy issues computation in the cloud is not desirable and local edge systems are essential. Ability to operate offline
- Architecture: Proxy server Secure-IoT Gateway and a secure data path
- Transmit over the cloud only high-level data (RSA)
- Low Energy consumption: Ability of the edge systems not to consume too much energy in order to be ecologically justifiable and affordable
- Natural user experience (low latency, high number of video frames)
- The communication interfaces that will be used are the Ethernet and the WiFi.

2.6 User Interfaces Layer

MagicMirror² (https://magicmirror.builders/) is Open Source, free smart mirror platform based on JavaScript language. With a growing list of installable modules, the MagicMirror² allows to convert a semi-transparent mirror into a personal assistant. The core of MagicMirror² contains a strong API which allows 3rd party developers to build additional modules.

The architecture is modular and each single module can be easily integrated in the MagicMirror2 platform that is using javascript language. We designed our functionalities with the aim to be easily integrated in the next future within the MagicMirror2 platform as third-party module. Some of the functionalities we intend to add to the magic mirror 2 framework are following:

- Medication Reminder module
- Emergency Help module
- Guided Health Check-Up module
- User Registration module



Tuesday, November 29, 2022
13:41 44

US HOLIDAYS

Christmas Dec 25th
New Year's Day Jan 1st
New Year's Day Holiday Jan 2nd
M L King Day Jan 16th
Valentine's Day Feb 14th
Presidents' Day Feb 20th
Good Friday Apr 7th
Easter Sunday Apr 9th
Mother's Day May 14th
Memorial Day May 29th

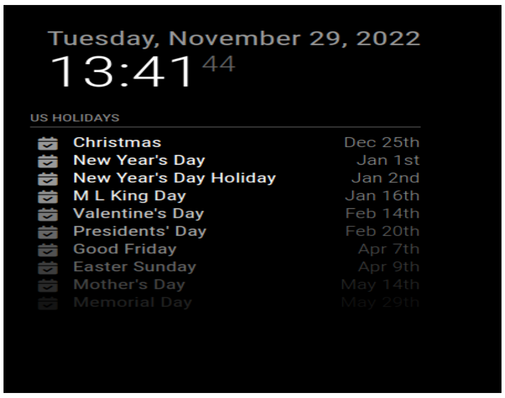
Loading ...

WEATHER FORECAST NEW YORK, US

Today 7.2° 3.0°
Tomorrow 14.0° 7.0°
Thu 10.0° 1.8°
Fri 8.2° 2.0°
Sat 12.3° 7.0°

Welcome iwelli

New York Times, 41 minutes ago
One Step Closer to a Universal Flu Vaccine?



Tuesday, November 29, 2022
13:41 44

US HOLIDAYS

Christmas Dec 25th
New Year's Day Jan 1st
New Year's Day Holiday Jan 2nd
M L King Day Jan 16th
Valentine's Day Feb 14th
Presidents' Day Feb 20th
Good Friday Apr 7th
Easter Sunday Apr 9th
Mother's Day May 14th
Memorial Day May 29th

Examples of the Edge4iwelli UI

2.7 Analysis Micro-computer

We intend to develop a customized edge4iwelli micro-computer module adopting the idea of modularity by supporting multiple computing systems and AI accelerators. PCIe-based accelerators can be flexibly connected to the computing modules. The system will supports many communication standards like USB 3.x , RJ45, wifi, Bluetooth, zigbee required by the use-cases. We are experimenting with RaspberryPi, the jetson nano and the Nvidia orin nanon microcomputers.

Combining Raspberry Pi 4 and Nvidia nano

2.8 VEDLIOT project and issues to be adopted in the edge4iwelli project:

One RGB and one IR camera are used in the Edge4iwelli project as the visual input devices. Cameras such as the Intel RealSense, which provide additional depth information as 3D point clouds will be tested as well.
Following the VEDLIOT deliverables we will adopt some of the functionalities and the modules of the magic mirror4 such as the Audio Stream module, the YOLO V4 for gesture detection together with the YOLO v7 algorithm that achieves the highest accuracy among all other real-time object detection models – while achieving 30 FPS or higher using a GPU V100. and others. Other models like the MobileNet-SSD, DeepSpeech, FaceNet 5 through the Voice Assistant Rhasspy tool we will validate, and others will be tested as well.
For the testing we will use a Raspberry pi with two Google Coral systems (in the two usb inputs) and we will also experiment with Nvidia nano and if necessary, we will test the Nvidia Jetson Xavier system. Although the IoT gateways6 systems using the FriendlyARM NanoPi R2S7 are not ready yet we will try to find methods to develop or customise an IoT gateway to provide a secure remote connection from the connected devices to data centers. From my conversation with Erik, they are using .

4 https://magicmirror.builders/.
5 Z. Qin, Z. Zhang, X. Chen, C. Wang and Y. Peng, "Fd-Mobilenet: Improved Mobilenet with a Fast Downsampling Strategy," 2018 25th IEEE International Conference on Image Processing (ICIP), pp. 1363-1367, 2018.
A. Y. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates and A. Y. Ng, "Deep Speech: Scaling up end-to-end speech recognition," CoRR, 2014.
F. Schroff, D. Kalenichenko and J. Philbin, "FaceNet," A unified embedding for face recognition and clustering, 2015.
66 https://embedded.christmann.info/secure-iot-gateway
77 https://openwrt.org/toh/hwdata/friendlyarm/friendlyarm_nanopi_r2s#dataentry

Following the VEDLIOT deliverable D2.3 we will try to work with a latency of less than 30 ms together with a throughput of at least 30 frames per second to get a more natural user experience.
The capabilities of FPGAs as acceleration devices using the software infrastructure will be explored. The value of this approach has to be discussed with the VEDLIOT team.

We will try to utilize the VEDLIOT relevant metrics for the smart mirror:

• Performance Metrics
  o Execution time
  o Latency (latency below 30ms)
  o Achieved performance
• Cost Metrics:
  o Resources
  o Cost
  o Power
  o Energy

• Quality Metrics:
  o Accuracy

• Combined Metrics:
  o (Power) Efficiency (Number of 30 FPS at 50 Watts when running all image processing frameworks simultaneously).
  o Cost Efficiency

Aspects to be considered within the edge4iwelli project are efficiency, security, data security and reliability (eg lighting conditions should not prohibit the system to monitor human skin colour).
Since ML models must be applied locally and not in the cloud (due to data privacy), optimization of these networks and usage of specialized hardware are essential. Within this aspect we will try the EmbeDL toolchain to optimize some of our models.
We will try to implement the attested edge trusted execution environments (TEEs) for voice recognition. Instead of uploading the voice to the cloud, we intend to do recognition in the devices deployed next to the user. By using attested TEEs, we can make sure that users' voice data never leaves their premises.

3 Implementation

The following text explains the edge4iwelli implementation and activities on the various layers (information viewpoint, system layers and logical components, computation viepoint) as well as describes how the health smart mirror features.

3.1 Information Viewpoint

The information viewpoint is focused on the information flow within the system. It is concerned with the types of information entities involved and the relationships between them.

The following table lists the main types of the information entities that are managed (recorded into databases, exchanged between system components, etc.) and processed within the Edge4iwelli system.

Table. Information entity types in the Edge4iwelli system

Entity	**Description**
Person	Information about a person, which can be an elderly user. Encapsulates properties of a person such as "name", "type", as well as dynamic properties such as "actual emotion", "actual vitals", and other.
Data Producer	Information about a sensor (here a mirror). Has the following properties: "type" and "location" (can be static or dynamic).
Event	Data received from a *Data Producer*, a common super-class of *Measurement*, *Person Event* and *System Event*. Has the following properties: "category", "value", "timestamp", "duration", and "confidence value" (certainty, from 0 to 1, that the reported event really took place, if can be estimated).
Measurement	A subclass of *Event* specifying a measurement event that took place. Can state, for example, that the heart rate of the monitored subject at 8am was 65bpm.
Person Event	Information about a non-quantified situation with a *Person*, such as a recognition of some action performed by or status change occurred to a *Person*.
System Event	Information about a digital event that occurred within the Edge4iwelli system rather than physical event that happened to a *Person*. Examples are incoming text form the person, record of an analysis performed, system fault, and so on.
Person Alert	Information about an exceptional situation with a *Person*. Typically generated based on a *Rule* after some *Person Events* or *Measurements* (or a series of those) are observed. Has the same properties are a *Person Event,* plus "criticality" (from 0 to 1, with higher number meaning more critical).
Rule	Information about a decision making logic to be followed by a software component, e.g. for generating *Person Alerts* from *Events*. Often hidden within the software programming code, but may also be explicitly represented as an information entity (i.e. encoded in a data structure to be stored and/or communicated between system components).
Schedule	A special case of a *Rule*, which is time-based. Information about the required timing for some actions to be taken by the system, such issuing *Messages* or *System Control Commands*. An example is a regular medication reminder.
Message	Message issued to a *Person* from an application, another component of the system, or another *Person*. For example, can be a medication reminder.
System Control Command	Instruction issued towards a device or a system component to perform some action, caused by an action of a person or by system decisions. For example, can be a command to to show specific information on the mirror.

The relationships between the Edge4iwelli information entities are depicted in the following figure. The arrows with triangle-shaped head have the meaning of the "subclass of" relationship (e.g. Person Event is a subclass of Event). This relationship is transitive. A subclass inherits all the properties of the its superclass (e.g. System Control Command has the same properties as Event, i.e. timestamp, category, etc.) as well as can participate in the same relationships as the super-class (e.g. Schedule can issue System Control Commands to send Messages).

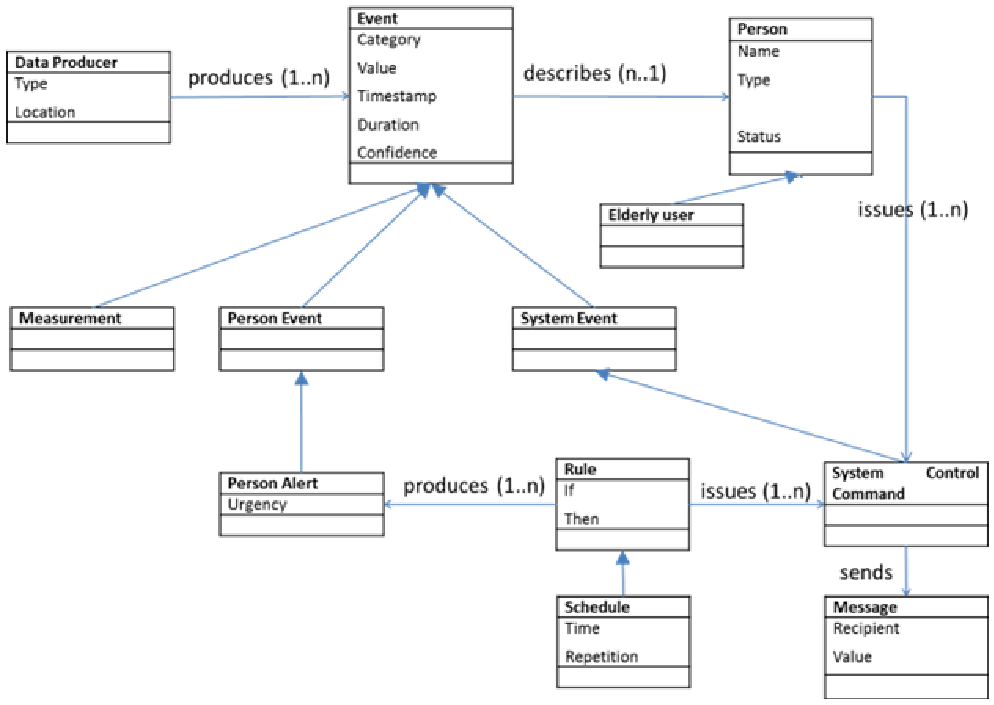


Figure. Information entities and their relationsips

3.2 Computation Viewpoint

The Computation viewpoint is focused on the processing performed within the system of various kinds of information (defined above from the Information Viewpoint).

The majority of the computational processes involved in the Edge4iwelli health smart mirror are overviewed in the following figure.

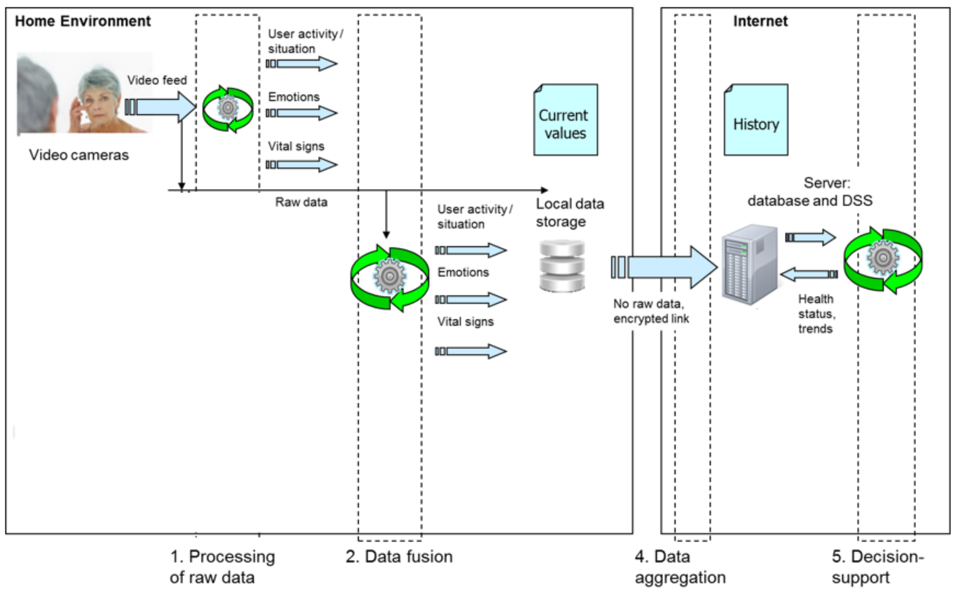


Figure. Data processing architecture

The Edge4iwelli computational processes:

1. **Processing of raw data (video, text, audio), filtering and abstraction.** This processing is algorithmic in nature where the raw data is first pre-processed to deal with missing, dropped and erroneous data as well as to remove noise and artefacts. Then, it is transformed into Measurements or Person Events as described above. The following processing happens mostly on these outputs, not the raw data. This fact, combined with the fact that this processing step will be performed close to the cameras/sensors, is also important from the privacy considerations. In particular, video does not leave the house, only the conclusions from video. Some raw data may be made available to data fusion step and even to local services, but cannot be send off-site.
2. **Data fusion.** The Edge4iwelli mirror attempts to infer some high-level Events based on several sources, for example person's activities from both video text and audio. Therefore, the data fusion is a processing step where the level of abstraction of information is further raised, potential conflicts between Measurements and Person events coming from different Data Producers are resolved, and the combined precision of the data is improved. The output of this process is a unified description of the Person's state and situation which is made available to the decision support processing step.
3. **Local service computing, rules and thresholds.** All Measurement and Person Events, potentially enhanced through data fusion, are made available to the local software services. Such services can perform various computations based on this data, apply Rules, and, in result, issue Person Alerts (e.g. when an extreme emotion state is registered), Messages to the elderly user or System Control Commands, or produce some higher-level Events based on recorded lower-level ones.
4. **Data aggregation**. Measurements and Person Events for the same elderly user but with different timestamps are collected into the user's personal history at the mydata.iwelli.com server. Also the data from similar users (e.g. suffering from similar condition) is combined to provide a uniform input to the decision support processing step.
5. **Decision-support related computing**. Statistical analysis (i.e. data mining) as well as rule-based analysis is applied to aggregated history data. The output of this step is evaluation of person's health status, estimates of trends, prognoses, and possible treatment suggestions.
6. **Server-side service and application computing**. Similarly to local services, server-side services and applications can process the Person Events and Measurements, but, more importantly, they also have access to the conclusions from the decision support step. They can apply Rules, generate Messages to the elderly user or Alerts to carers.

3.3 System layers and logical components

From the engineering viewpoint, the Edge4iwelli architecture is divided into a set of layers as follows:

1. **Data Collection layer** comprises logical elements responsible for the in-flow of information into the Edge4iwelli mirror, i.e. various sensing systems.
2. **Data Processing layer** comprises logical elements responsible for processing and enriching the information.
3. **System Core Services layer** consists of the core services that enable the operation of the Edge4iwelli mirror as a distributed system, including making the data available to the higher layers, enforcing the Edge4iwelli communication and security infrastructures.
4. **Utility Core Services layer** consists of re-usable software services that are used as building blocks by the applications.
5. **Applications layer** consists of the Edge4iwelli applications that deliver the actual added-value to the Edge4iwelli system stakeholders.

6. **User Interfaces layer** comprises various interfaces exposes to the system users, including elderly users, health professionals, and others.

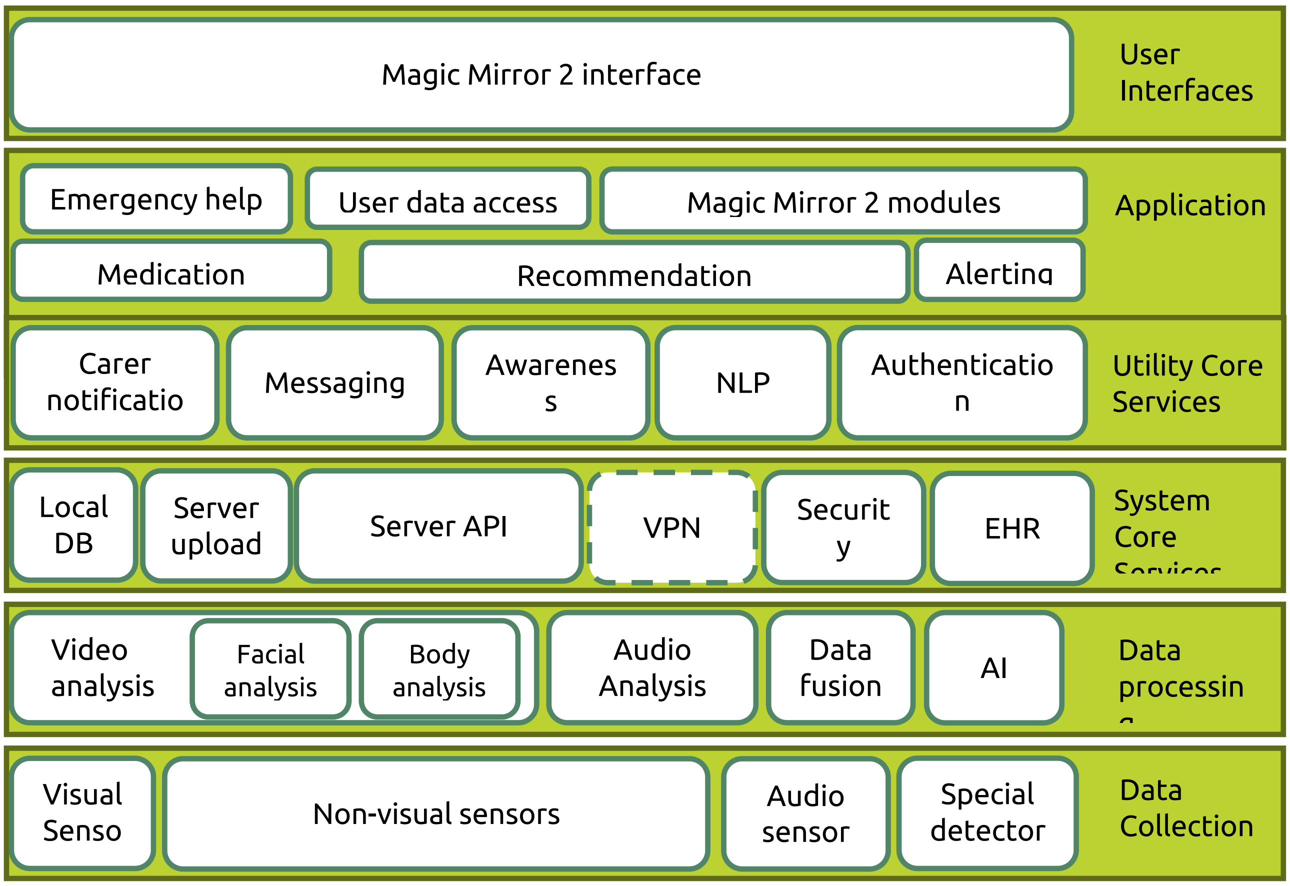


Figure. Overview of the logical components of the Edge4iwelli system

The following figure depicts the overall flow of information within the Edge4iwelli health smart mirror systems and between the mirror and the web portals.

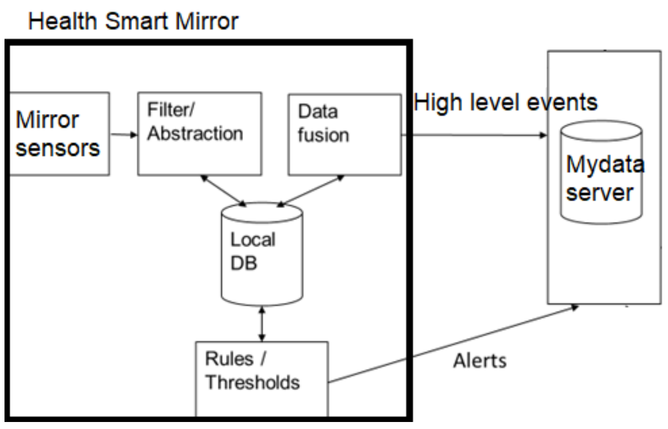


Figure. Overall Edge4iwelli smart mirror information flow

3.4 Setting Up Smart Health Mirror

Smart Health Mirror will be pre-configured with the required apps and graphical assets before shipping out to customer site. It will be ready to plug-and-play, so setting up the Smart Health Mirror will be very straightforward and only involve a few steps:

A. Unpack the Smart Health Mirror
B. Setup the Smart Health Mirror
   B.1 Carefully mount it in an appropriate surface
   B.2 Connect the power supply adaptor to the Smart Health Mirror
   B.3 Connect to the Internet through WiFi

3.5 Unpack the Smart Health Mirror

Inside the box, there will be 1 unit of Smart Health Mirror, and an accessory pack (1 external Power Supply Adaptor, Power Cable, Mini Wireless device - Wi-Fi Antenna).

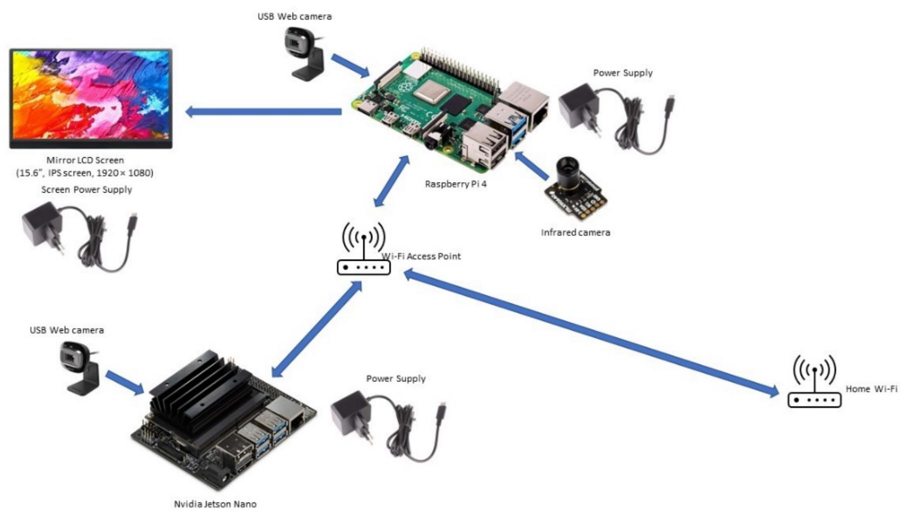


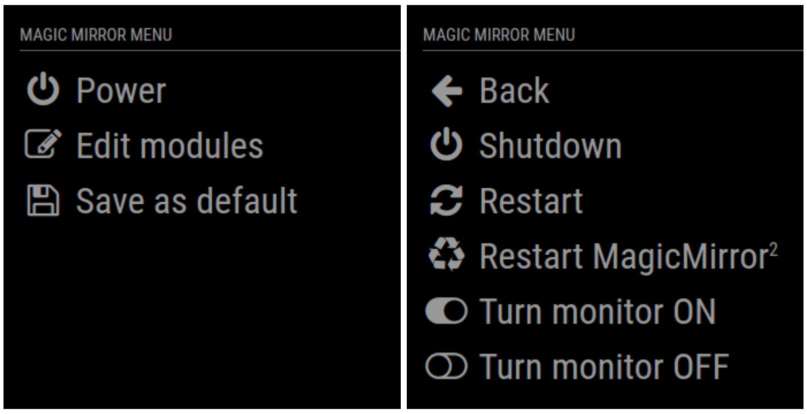
Figure 1: Internal structure of the Smart Health Mirror

3.6 Setup the Smart Health Mirror

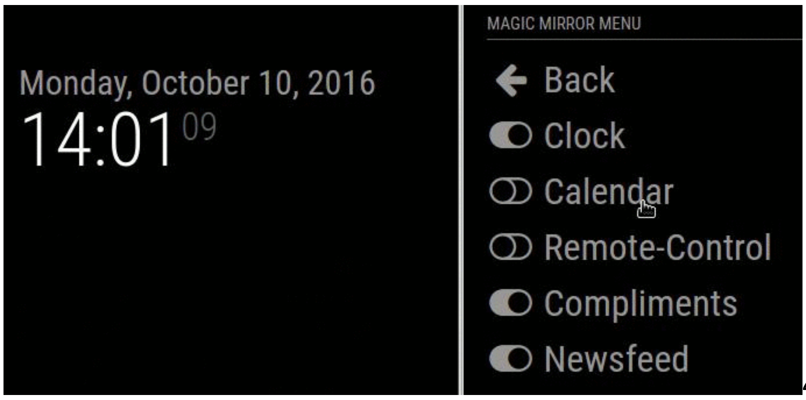
To successfully setup the Smart Health Mirror follow the steps below:

1. Ensure that the Smart Health Mirror is safely mounted in an appropriate surface that can support its weight.
2. Plug in the Power Supply Adaptor to a wall socket and the other end of the cable to the Smart Health Mirror.
3. Press the Power Button to power up the Smart Health Mirror.
4. Wait approximately 1 minute for the initial startup sequence.
5. Connect to the Wi-Fi.
6. Stand still in front of the Smart Health Mirror for approximately 30 seconds or until you start getting results from the Smart Health Mirror.

Personalizing your Smart Health Mirror



Magic Mirror Menu (https://magicmirror.builders/)

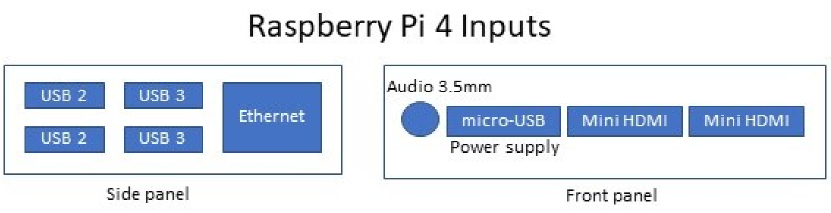


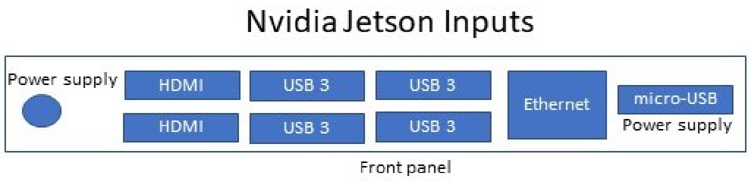
4

Magic Mirror Menu (https://magicmirror.builders/)

You can also personalize the Smart Health Mirror user interface using any device that is connected in the same network with the Smart Health Mirror. Using your network connected device you can:

- Shutdown/restart Smart Health Mirror.
- Turn on/turn off Smart Health Mirror screen.
- Change Smart Health Mirror's screen brightness.
- Show/hide any of the Smart Health Mirror's modules.



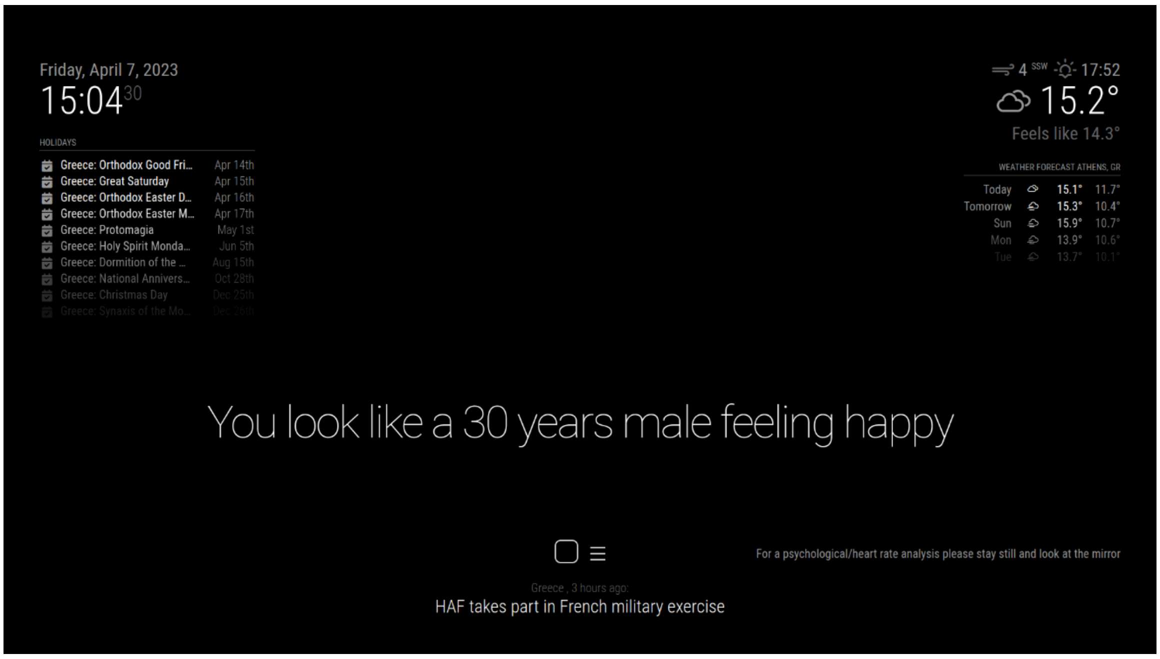


3.7 OPERATIONS

The user can stand in front of the mirror and get the following information at time intervals of 30 seconds.

3.7.1 face-api.js

- Face Expression Recognition
    - Surpised
    - Disgusted
    - Fearful
    - Sad
    - Angry
    - Happy
- Gender Recognition (Male, Female)
- Age Estimation (in years)



Friday, April 7, 2023
15:04 30

HOLIDAYS

Greece: Orthodox Good Fri...	Apr 14th
Greece: Great Saturday	Apr 15th
Greece: Orthodox Easter D...	Apr 16th
Greece: Orthodox Easter M...	Apr 17th
Greece: Protomagia	May 1st
Greece: Holy Spirit Monda...	Jun 5th
Greece: Dormition of the ...	Aug 15th
Greece: National Annivers...	Oct 28th
Greece: Christmas Day	Dec 25th
Greece: Synaxis of the Mo...	Dec 26th

4 SSW 17:52
15.2°
Feels like 14.3°

WEATHER FORECAST ATHENS, GR

Today	15.1°	11.7°
Tomorrow	15.3°	10.4°
Sun	15.9°	10.7°
Mon	13.9°	10.6°
Tue	13.7°	10.1°

You look like a 30 years male feeling happy

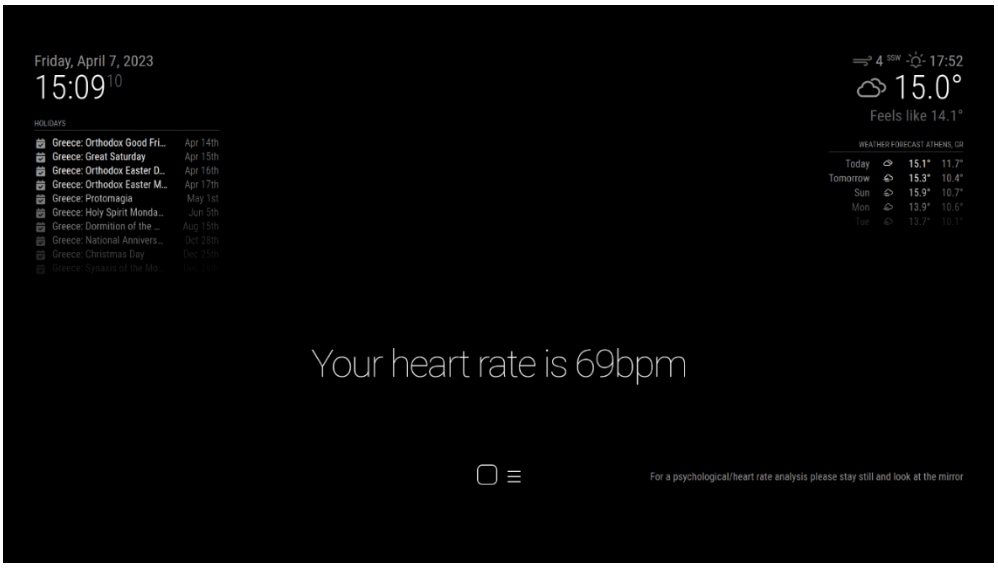
For a psychological/heart rate analysis please stay still and look at the mirror

Greece , 3 hours ago:
HAF takes part in French military exercise

A low cost RGB pc camera with face-api libraries provides the heart rate

3.7.2 Heart rate estimation

The smart mirror extracts Blood Volume Pulse (BVP) and Beats Per Minute (BPM) from a video of your face.



Friday, April 7, 2023
15:09 10

HOLIDAYS

Greece: Orthodox Good Fri...	Apr 14th
Greece: Great Saturday	Apr 15th
Greece: Orthodox Easter D...	Apr 16th
Greece: Orthodox Easter M...	Apr 17th
Greece: Protomagia	May 1st
Greece: Holy Spirit Monda...	Jun 5th
Greece: Dormition of the ...	Aug 15th
Greece: National Annivers...	Oct 28th
Greece: Christmas Day	Dec 25th
Greece: Synaxis of the Mo...	Dec 26th

4 SSW 17:52
15.0°
Feels like 14.1°

WEATHER FORECAST ATHENS, GR

Today	15.1°	11.7°
Tomorrow	15.3°	10.4°
Sun	15.9°	10.7°
Mon	13.9°	10.6°
Tue	13.7°	10.1°

Your heart rate is 69bpm

For a psychological/heart rate analysis please stay still and look at the mirror

A low cost RGB pc camera with rPPG algorithms provides the heart rate

To accurately extract the BVP and BPM a video of at least be 420 frames at 30 frames per second, meaning 14 seconds of video is recorded. The application will scan the video to find your face, and then extract RGB values from the recorded video. To ensure precise analyses of the video, we assign patches on different areas of the face that we track, enabling us to accurately map changes in RGB values in specific areas of the face.

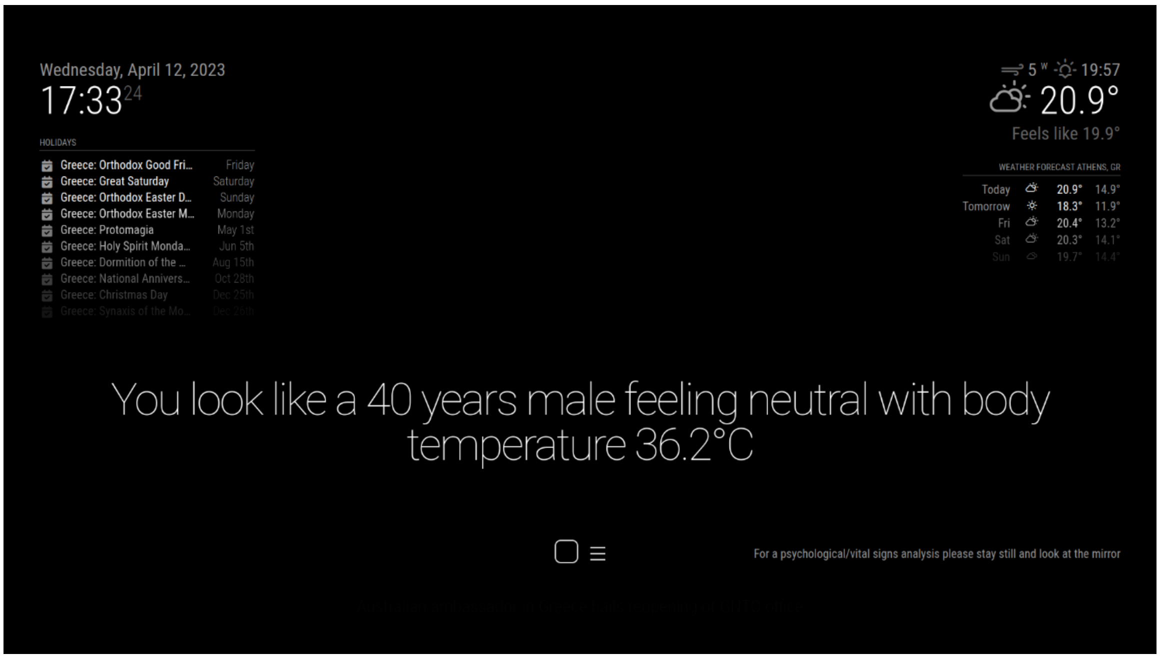
To achieve optimal results for the Remote photoplethysmography (rPPG) task, we use the POS rPPG method. It is crucial to note that for optimal results, the person being recorded should remain stationary and maintain eye contact with the camera during the video recording. This ensures that the RGB values are accurately extracted and can be used for the analyses.

The estimated time to receive results from the start of recording to the end of analyses is approximately 35-45 seconds. Once the analyses are complete, the process will restart, and the user can initiate another recording. If no face is detected during the recording, no values will be returned.

The analyses and the video recording uses the Nvidia Jetson Nano 4GB device. The results are then sent to the Raspberry Pi device using a GET Request.

3.7.3 Body Temperature

The smart mirror utilizing and IR camera extracts Body Temperature from a video of your face.


Wednesday, April 12, 2023
17:33 24

HOLIDAYS

Greece: Orthodox Good Fri...	Friday
Greece: Great Saturday	Saturday
Greece: Orthodox Easter D...	Sunday
Greece: Orthodox Easter M...	Monday
Greece: Protomagia	May 1st
Greece: Holy Spirit Monda...	Jun 5th
Greece: Dormition of the ...	Aug 15th
Greece: National Annivers...	Oct 28th
Greece: Christmas Day	Dec 25th
Greece: Synaxis of the Mo...	Dec 26th

5 W 19:57
20.9°
Feels like 19.9°

WEATHER FORECAST ATHENS, GR

Today	20.9°	14.9°
Tomorrow	18.3°	11.9°
Fri	20.4°	13.2°
Sat	20.3°	14.1°
Sun	19.7°	14.4°

You look like a 40 years male feeling neutral with body temperature 36.2°C

For a psychological/vital signs analysis please stay still and look at the mirror

Infrared Raspberry Pi Camera provides the body temperature

3.8 APPS

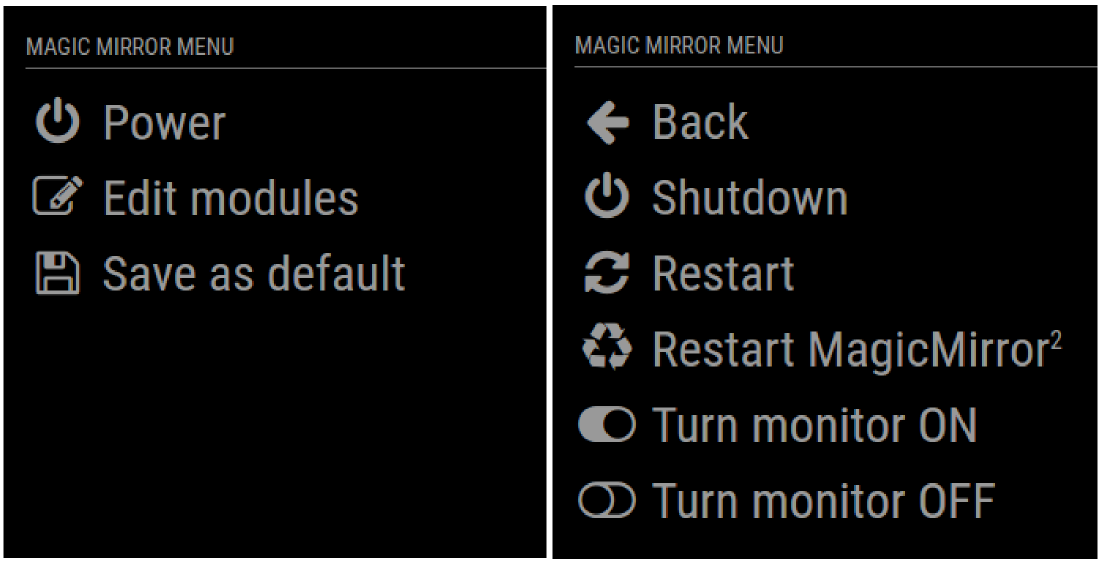
The applications of the Smart Health Mirror have been produced utilizing the MagicMirror² [8]open source modular smart mirror platform.

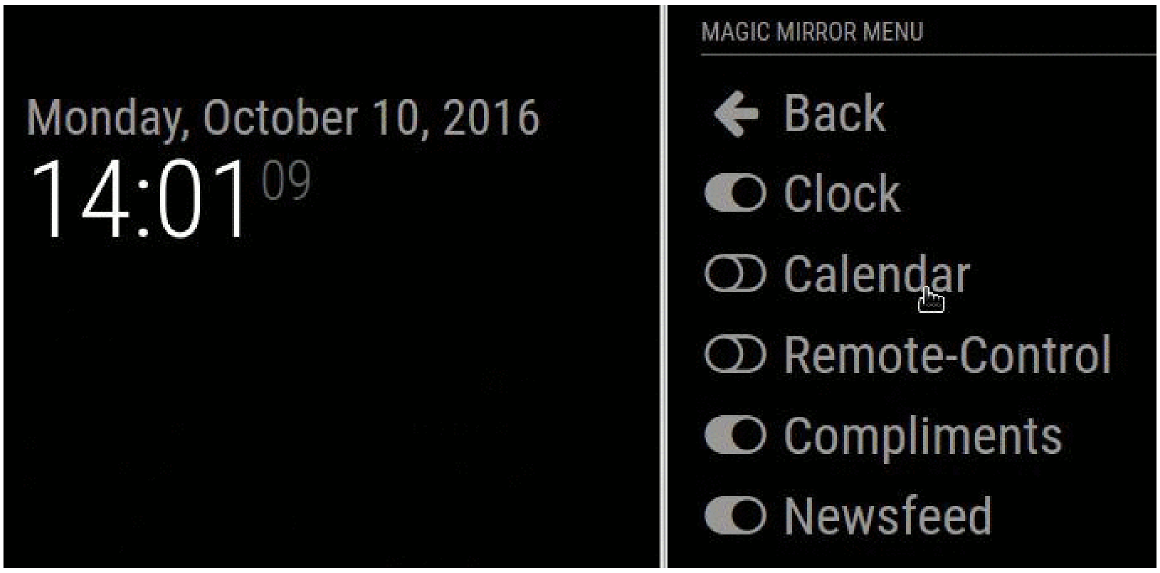
Apps	**Description**
Newsfeed	This module displays news headlines based on an RSS feed. Scrolling through news headlines happens time-based (updateInterval) but can also be controlled by sending news feed specific notifications to the module.
Weather	This module is used as a current weather view, or to show the forecast. This way the module can be used twice to fulfill both purposes.
Notifications	This module displays a notification of user's heart rate, facial expressions, age and gender.
Calendar	This module displays events from a public. ical calendar. It can combine multiple calendars.
Clock	This module displays the current date and time. The information will be updated realtime.
Remote Control w/ RESTful API	This module allows you to quickly shutdown your mirror through a web browser. The remote control should work fine on any device (desktop, smart phone, tablet, ...). Additionally, you can hide and show modules on your mirror and do other cool stuff

[8] https://magicmirror.builders/

3.9 CONTROL PANEL

3.9.1 Mirror Remote Control





3.9.2 List of actions

System Control:

Action	Description
SHUTDOWN	Shutdown your RaspberryPi
REBOOT	Restart your RaspberryPi
MONITORON	Switch your display on. Also sends a "USER_PRESENCE": true notification.

MONITOROFF	Switch your display off. Also sends a "USER_PRESENCE": false notification.
MONITORTOGGLE	Toggle the display on or off (with respective "USER_PRESENCE" notification.
MONITORSTATUS	Report back the monitor status (on or off)

MagicMirror Control:

Action	Description
RESTART	Restart your MagicMirror
REFRESH	Refresh mirror page
UPDATE	Update MagicMirror and any of it's modules
SAVE	Save the current configuration (show and hide status of modules, and brightness), will be applied after the mirror starts
BRIGHTNESS	Change mirror brightness, with the new value specified by value. 100 equals the default, possible range is between 10 and 200.

MagicMirror Electron Browser Window Control:

Action	Description
MINIMIZE	Minimize the browser window.
TOGGLEFULLSCREEN	Toggle fullscreen mode on and off.
DEVTOOLS	Open the DevTools console window.

Module Control:

Action	Description
HIDE	Hide a module, with the name (or identifier--see MODULE_DATA action) specified by module in the payload. You can also send module: "all" to hide all modules.
SHOW	Show a module (see above for how to specify which one).
TOGGLE	Toggle a module's visiblity (see above for how to specify which one).
FORCE	Force a module to show (see above for how to specify which one).
MODULE_DATA	Returns a JSON format of the data displayed in the UI, including all valid identifiers for the HIDE and SHOW action.

Alerts and Notifications:

Action	Description
SHOW_ALERT	Show Default Alert/Notification
HIDE_ALERT	Hide Default Alert/Notification

USER_PRESENCE	Will send a notification "USER_PRESENCE" = true or false (according to "value" to all other modules. See examples above
NOTIFICATION	To send a notification to all modules, see the example in the API README
DELAYED	Send any of the above nested inside a "DELAYED" call to delay the action. Default is 10s. See Delayed Actions below.

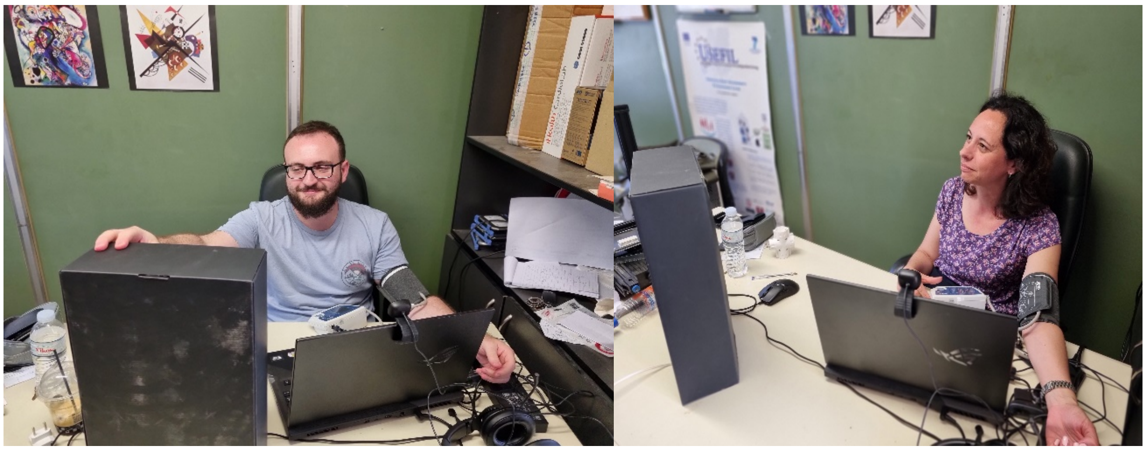
3.10 Troubleshooting

Component	**Problem**	**Possible Causes**	**Solution**
Edge computing devices (Raspberry pi 4 and Nvidia nano)	Computing devices are not working after the main power is switched on	The power cable was not plugged in properly	Ensure the power cable is plugged into the power socket securely. Unplug and re-plug the cable if necessary.
		Power supply of computing devices are not functioning well	Ensure the power supply is sufficient for our mirror. The required power voltage is 100 - 240V.
Touch Screen & Display	No display on main screen	The power cable was not plugged in properly	Ensure the power cable is plugged into the power socket securely. Unplug and re-plug the cable if necessary.
		The main switch of display is not turned on	Ensure the power button is switched on. Click again if necessary.
	Display not bright enough	Brightness is not adjusted based on environment setting	Adjust the brightness of the display using control until optimal.
Applications	System Crashed or Not Responding		Relaunch Smart Health Mirror by referring to section 1. for start-up. Power off the display and Wait for 5-10mins Turn on main power, start Edge computing devices and launch Smart Health Mirror

4 Evaluation and characterization

The objective of the validation is to assess the accuracy and reliability, of the systems in monitoring emotions, gender and age data, health vital signs of heart rate and body temperature.

We conducted the trials in the lab with engineers and we used an A&D Blood Pressure device for monitoring the Heart Rate with a CE certified medical device and an analog thermometer that monitored the body temperature. The participants in these lab tests are engineers form the lab (Papadopoulos Homer, Antonis Korakis, George Balaskas and Areti Katsamagkou) to ensure a representative sample.



In lab validation of the magic health mirror

We collected data on emotions, vital signs (heart rate and body temperature) from the magic health mirror devices and compare these data with reference measurements obtained using standard medical devices (A&D blood pressure and analog thermometer). Initially we obtained baseline vital sign measurements from the in lab participants using the reference medical devices to establish a starting point for comparison

We conducted the trials over an appropriate period of a 3 weeks to capture different scenarios and fluctuations in vital signs. The feedback we got from the in lab personnel helped us to assess the usability, satisfaction, and perceived value of the magic health mirror. We consider the lab as a controlled environment, to minimize external factors that could influence vital sign measurements.

4.1 Validation Methodology of the edge computing modules and the software tools that will run the vital signs monitoring (heart rate and body temperature) and the emotion analysis

The validation trial report of the integrated magic health mirror system presents some first results that small scale validation studies produced (conclusions of trial results, acceptability, and usability). To assess the effectiveness and usability of a magic mirror using the Nvidia Jetson Nano and Nvidia Orin nano systems we applied the following validation methodology, trials, and results for the magic health mirror that monitors among the others health vital signs for older people:

1. Initially we defined the objectives of the validation process. Because of the inability to test different edge computing systems and the fact that we wanted to keep the cost low we bought nvidia jetson nano and nvidia orin nano systems and we run the

algorithms in these two systems. We identified the aspects of the systems that run the machine learning algorithms to monitor the vital signs, we wanted to validate, **such as their performance, accuracy, user experience, and any specific features or functionalities.**

2. We tried to evaluate the performance of the magic mirror (how the algorithms perform in the two nvidia jetson nano systems) by assessing its ability to accurately monitor and display heart rate information. Use the machine learning algorithm running on both the NVIDIA Jetson Nano and Nvidia Orin nano systems to measure the accuracy of heart rate detection. We compared the results obtained from the two systems to determine any differences in performance.
3. We also conducted in-lab usability tests to assess the user experience of the magic mirror. Within these tests we interacted with the mirror interface, adjusting settings, navigating menus, and interpreting heart rate and body temperature data displayed. We collected feedback from the end users regarding the mirror's ease of use, intuitiveness, and overall satisfaction with the interface.
4. We analyzed the data collected during the validation activities. For the performance validation we compared the accuracy of heart rate monitoring obtained from the Nvidia Jetson Nano and Nvidia Orin nano systems. We looked for any discrepancies or variations in the results.
5. We also compared the body temperature coming from the IR camera with the analogue temperature device.
6. For usability testing of the magic health mirror, we analyzed user feedback and identified common issues, concerns, or suggestions for improvement.
7. This report documents the validation process, including the objectives, methodology, data collected, analysis, and any improvements made. Summarize the findings, including the performance comparison between the Nvidia Jetson Nano and Nvidia Orin nano systems, as well as the usability test results.

We intend to further extend the validation trials in the next months. More specifically we intend to:

1. Take the magic health mirror to real-world environments or scenarios where it will be used. This could include installing the mirror in a home or healthcare setting where individuals can interact with it in their daily routines. Monitor the mirror's performance and gather feedback from users over an extended period to assess its practicality and reliability.
2. Because the validation is an iterative process, and it may require multiple rounds of testing and improvements to achieve the desired performance and user experience we will adopt iteration validation cycles of improvement. Based on the analysis of the validation results in the real world scenarios, we will make any necessary improvements to the magic health mirror. This could involve fine-tuning the machine learning algorithm, optimizing the hardware setup, refining the user interface, or addressing any usability issues identified during testing. Offcourse we have to notice that it's important to follow ethical guidelines, seek appropriate approvals for real scenario validation trials, and involve experts in the field to ensure rigorous validation and accurate interpretation of the results.

4.2 Edge4iwelli Magic Health Mirror, Face-API and the body temperature monitoring component

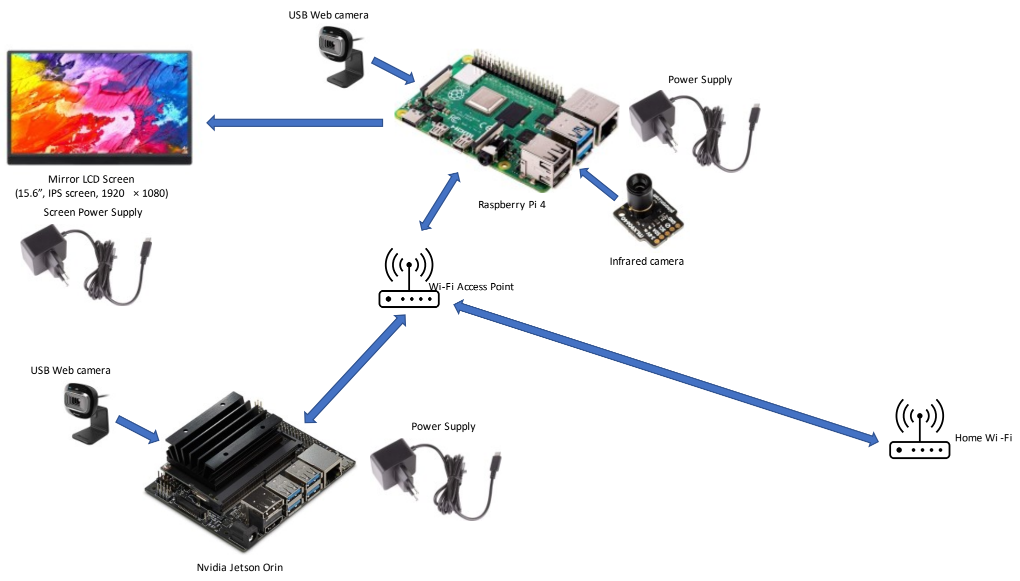
4.2.1 Magic Health Mirror

The Magic Health Mirror device is intended to monitor and interact with the subject. The interaction enables the subject to view useful information easily and quickly and to automatically monitor his/her physiological parameters.

The whole system consists of components:

1. A magic mirror with programmable applications and digital display

2. Two web cameras which enables the subject to monitor his/her physiological parameters, through real-time analysis of video containing his/her face, as captured by the web camera and one Infrared camera for the body temperature monitoring



The main systems that have been validated

The data acquired through the magic health mirror can be controlled, customized and stored via a web portal individually for each user if the users decides to register in the mydata.iwelli.com portal. The user can view some useful information in the mirror's screen. The cameras that are integrated in the mirror streams the audio and video data to the edge computing systems (raspberry 4 and Nvidia Orin nano), where the analysis is performed, using the Face-Api video analysis tool, the IR camera tool and the machine learning for the heart rate monitoring. The data are then saved in the web portal for further analysis if the user agrees for that. To ensure that the user's privacy is not compromised, the user can close and open the camera and the microphone, whenever he/she wants.

4.2.2 Face-API

face-api is a JavaScript library created by Vincent Mühler., to detect faces via browser. It is built over tensorflow.js core API. It supports Face Detection, Face Recognition, Face Expression, Age, and Gender Detection. You can clone the following face-api repository git clone https://github.com/justadudewhohacks/face-api.js.git

Validation results

Magic Health Mirror camera/ face-api js and the temperature monitoring component	Poor detection capabilities in dim light	Magic health mirror camera is not working and face-api and temperature monitoring software do not work properly in the presence of dim light.	Optimal positioning of the Mirror camera

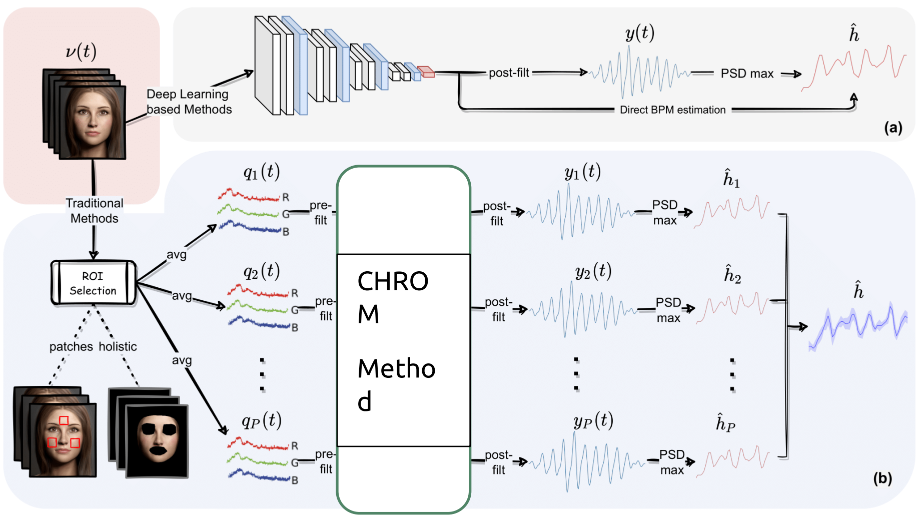
4.2.3 Body Temperature monitoring system

For the body temperature we use the IR Camera: MLX90640: https://makersportal.com/blog/2020/6/8/high-resolution-thermal-camera-with-raspberry-pi-and-mlx90640 and for monitoring and analysing the data we use the pythin data library of the Adafruit (adafruit_mlx90640).

4.2.4 Package pyVHR: Python library for Heart Rate monitoring

PyVHR is a Python framework for remote photoplethysmography for studying methods of pulse rate estimation relying on video, also known as remote photoplethysmography (rPPG). pyVHR provides APIs for handling 11 publicly available video datasets, i.e. *PURE*, *LGI-PPGI-DB*, *UBFC-1*, , *UBFC-2*, *UBFC-Phys*, *ECG-Fitness*, *MAHNOB Vicar-PPG-2*, *V4V*, *VIPL-HR* and *COHFACE*, usually adopted to benchmark rPPG methods.

We validated all the nine classical rPPG methods (namely *ICA*, *PCA*, *GREEN*, *CHROM*, *POS*, *SSR*, *LGI*, *PBV*, *OMIT*, as well as the recent Deep Learning-based model *MTTS-CAN) with our edge computing devices and we concluded that the CHROM rPPG method is the one that provided the most accurate results using our specific system implementation.*



4.3 Validation Trials Results

We analyze the data collected from the magic health mirror and compare it with reference measurements to assess the accuracy and concordance of the vital sign measurements. The analysis of the heart rate with the two different edge computing devices is following.

We analyzed the feedback from participants regarding the usability, ease of use, and overall experience of the magic mirror. The identified areas of improvement and potential enhancements based on user suggestions were among the others the position of the magic health mirror (in the same level of the head), the duration of the each measurement and the frequency of the measurements, the light intense that has to suggest to end users to get better temperature measurements.

To summarize the validation results indicated that the accuracy of the vital signs is an area where we have to further work on, the magic health mirror is reliable enough, and the usability of the magic mirror in monitoring health vital signs for older people is something that is still under development.

4.3.1 Limitation 1: Poor detection capabilities in dim light

In the development of the face-api and the body temperature component the lighting requirements that were proposed were those of adequate lighting in order to allow the correct recording of the user's face and the subsequent reliable extraction from it of the necessary features as events for the system. In our test setup the mirror with the camera was positioned in a place that allowed different lighting conditions from extremely dim to full overhead and daytime light. When testing this Magic Health Mirror set up dim lighting conditions provided unreliable results sometimes failing to provide any measurement at all. Consistently reliable results were provided only by good lighting conditions, that is clear overhead lighting or daytime light without heavy curtains or shutters pulled down.

This requirement was recorded and is noted as a guideline for the installation of the cameras and mirror. This limitation is not considered a significant one since the use case for the mirror and its camera involve its installation in the user's salon or bedroom which is usually a very well, artificially, lit area of the house. Thus the only significant care that should be taken is not to position the recording camera in a place with no clear view between the user and it.

4.3.2 Limitation 2 Suboptimal data rate from face-api, temperature and heart rate to the edge computing for further analysis

The face-api module's purpose is to provide physiological information about the user as this is extracted by the recorded signs that are apparent on her/his face (expression of the face etc.). The body temperature and the heart rate monitoring intend to generate vitals monitoring data. The storing of these data happens through the creation of events that are sent to the edge computing -raspberry 4 device. From our testing and from the size of the data that were stored in the raspberry 4 device after some realistic testing it became clear that the face-api and the other software components send events at a rate that is not practically. While the second to second recording of the facial-api and the vitasl would be appropriate possibly for security uses, in the context of health monitoring a far lesser rate would be appropriate. Suggestions included averaging of signals over small time periods (30 or 60 seconds) or even data production and transmission only when the difference between subsequent measurements present significant change or a definitive trend. All these solutions would greatly reduce the load data throughput and would also reduce the load of bulk data stored to the raspberry 4 device.

4.4 Analysis of the Validation trials or the Nvidia edge computing devices

We conducted the following validation analysis to compare the performance of Nvidia Jetson Nano and Nvidia Orin nano for running a machine learning algorithm that monitors heart rate. We defined the specific requirements of the heart rate monitoring machine learning algorithm (the results should be similar to the A&D BP device that is measuring the HR). Consider factors such as the desired accuracy, real-time processing capability and any other relevant considerations. We chose the machine learning model we describe below for heart rate monitoring and we ensured that the model is optimized and performs well on the chosen hardware.

We set up both the NVIDIA Jetson Nano and NVIDIA Orin nano devices according to their respective instructions. Install the necessary software frameworks, libraries, and dependencies for running the machine learning algorithm.

We evaluated the performance of both devices using the testing portion of our dataset. We measured the metrics and we run multiple trials to ensure reliable results.

We analyzed the collected performance data and compare the results between the two devices. Based on the results and analysis, the Nvidia Orin nano system outperformed the Nvidia Jetson nano for the heart rate monitoring machine learning algorithm. Therefore we adopted and installed in the magic health mirror the Nvidia Orin nano system.

The analysis of the performance of both the devices is following:

4.4.1 Blood Volume Pulse and Beats Per Minute extraction.

The magic health mirror extracts Blood Volume Pulse (BVP) and Beats Per Minute (BPM) from a video of your face.

To accurately extract the BVP and BPM a video of at least be 900 frames at 30 frames per second, meaning 30 seconds of video is recorded. The application will scan the video to find your face, and then extract RGB values from the recorded video. To ensure precise analyses of the video, we assign patches on different areas of the face that we track, enabling us to accurately map changes in RGB values in specific areas of the face.

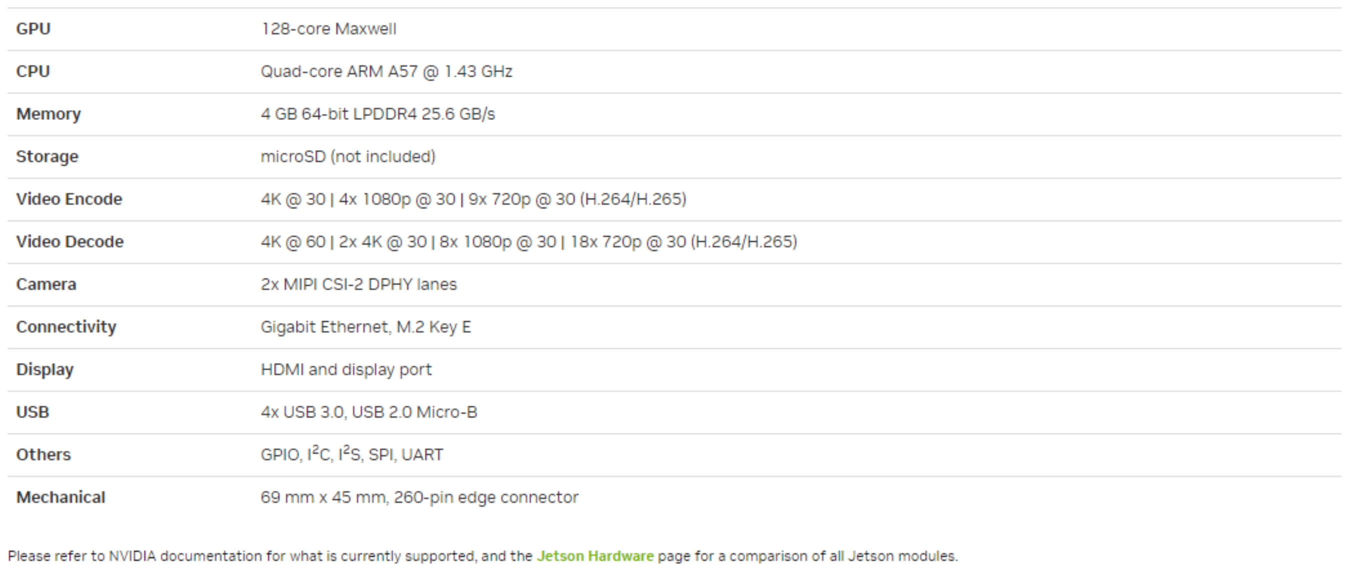
To achieve optimal results for the Remote photoplethysmography (rPPG) task, we use the *POS* rPPG method. It is crucial to note that for optimal results, the person being recorded should remain stationary and maintain eye contact with the camera during the video recording. This ensures that the RGB values are accurately extracted and can be used for the analyses. The results are then sent to the Raspberry Pi device using a GET Request.

The estimated time to receive results from the start of recording to the end of analyses depends on the device used. Once the analyses are complete, the process will restart. If no face is detected during the recording, no values will be returned.

We have conducted hardware tests with two different Nvidia devices, the Nvidia Jetson Nano Dev Kit 4GB device and the Nvidia Jetson Orin Nano Dev Kit 8GB device, both devices were overclocked to the maximum using the *jetson_clocks* script provided by Nvidia. The aim of the test is to find the device that provides the best performance in terms of speed and accuracy.

4.4.2 Nvidia Jetson Nano 4GB

The Nvidia Jetson Nano is a small portable device that was released in March of 2019. The technical specifications of the Jetson Nano 4GB are the following:



GPU	128-core Maxwell
CPU	Quad-core ARM A57 @ 1.43 GHz
Memory	4 GB 64-bit LPDDR4 25.6 GB/s
Storage	microSD (not included)
Video Encode	4K @ 30 | 4x 1080p @ 30 | 9x 720p @ 30 (H.264/H.265)
Video Decode	4K @ 60 | 2x 4K @ 30 | 8x 1080p @ 30 | 18x 720p @ 30 (H.264/H.265)
Camera	2x MIPI CSI-2 DPHY lanes
Connectivity	Gigabit Ethernet, M.2 Key E
Display	HDMI and display port
USB	4x USB 3.0, USB 2.0 Micro-B
Others	GPIO, I^2C, I^2S, SPI, UART
Mechanical	69 mm x 45 mm, 260-pin edge connector

Please refer to NVIDIA documentation for what is currently supported, and the Jetson Hardware page for a comparison of all Jetson modules.

Figure: Jetson Nano 4GB Technical Specifications

Due to the limited RAM memory on the Nvidia Jetson Nano we had to first record 900 frames of video and then analyse it. When the recording and analyses happened simultaneously, the device was not able to keep up with a constant recording. Due to the high RAM usage the video was not constant, and the analyses suffered. By making process of extracting BVP and BPM sequential we were able to get more accurate and stable results at the cost of the overall run time.

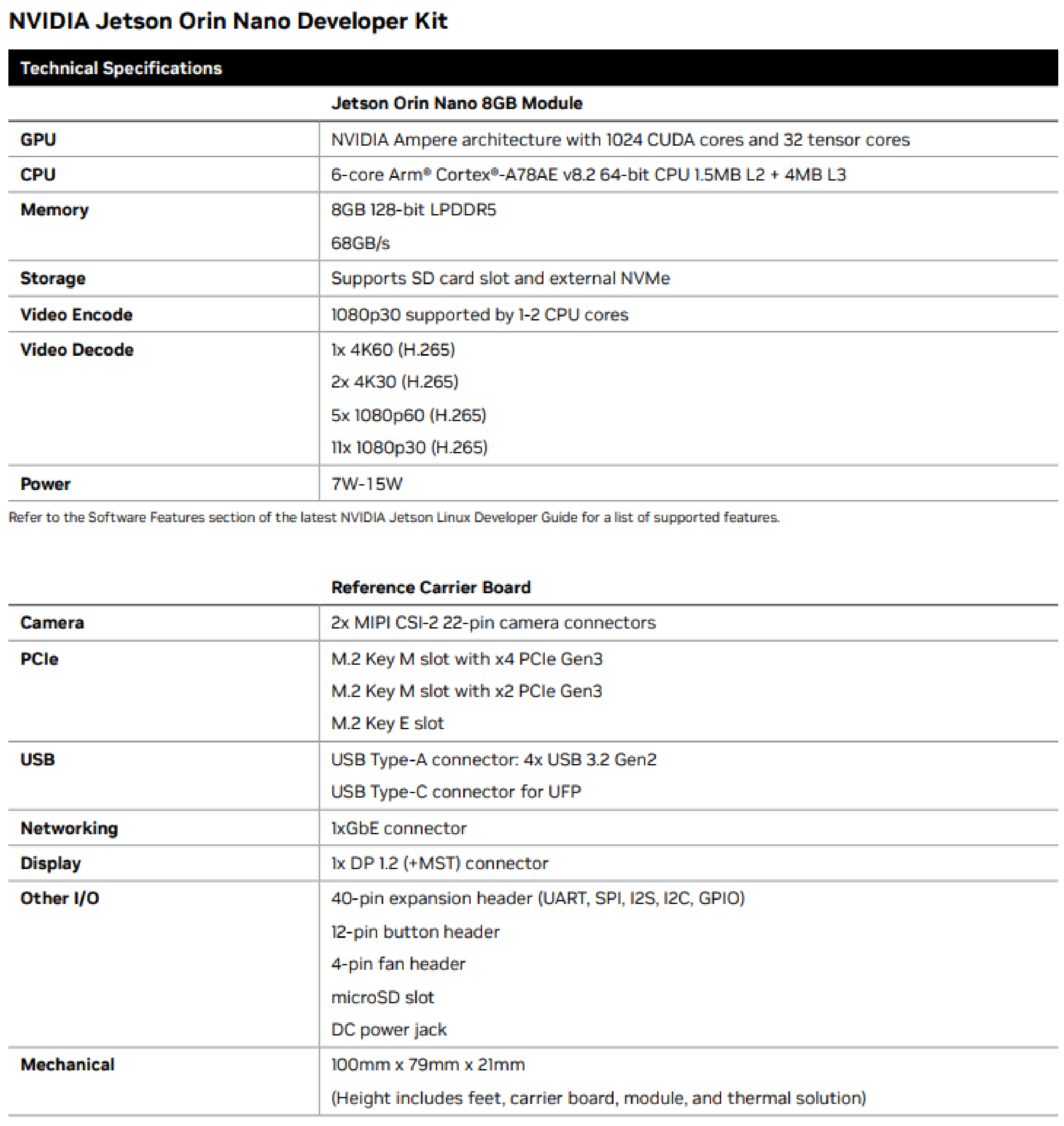
ID	BPM	Run Time	seconds	frames
0	66	177.1019	30	900
1	78	191.72	30	900
2	76	181.3609	30	900
3	70	195.653	30	900
4	63	191.5142	30	900
5	54	167.0198	30	900
6	75	162.4128	30	900
7	67	171.9826	30	900
8	78	175.6632	30	900
9	81	162.5667	30	900
10	71	177.4912	30	900
11	67	172.9353	30	900
12	71	187.462	30	900
13	70	172.3246	30	900
14	71	175.9871	30	900
15	65	183.5503	30	900
16	83	177.9027	30	900
17	73	206.4733	30	900
18	64	171.9285	30	900
19	60	171.3788	30	900
20	72	181.9291	30	900

4.4.3 Sequential Evaluation Jetson Nano 4GB

As we can see from Appendix 1, the time to extract the BVP and BPM is significant with an average run time of 165.15+-8.9 seconds for 900 frames or 30 seconds of video. Additionally, we extract an average of 85.7 +-9.72 BPM over 82 test runs. Indicating that everything is functioning as expected, albeit at a in a not so timely manner.

4.4.4 Nvidia Jetson Orin Nano 8GB

The Nvidia Jetson Orin Nano is a small portable device that was released in March of 2019. The species of the Jetson Orin Nano 8GB are the following:



NVIDIA Jetson Orin Nano Developer Kit

Technical Specifications

	Jetson Orin Nano 8GB Module
GPU	NVIDIA Ampere architecture with 1024 CUDA cores and 32 tensor cores
CPU	6-core Arm® Cortex®-A78AE v8.2 64-bit CPU 1.5MB L2 + 4MB L3
Memory	8GB 128-bit LPDDR5 68GB/s
Storage	Supports SD card slot and external NVMe
Video Encode	1080p30 supported by 1-2 CPU cores
Video Decode	1x 4K60 (H.265) 2x 4K30 (H.265) 5x 1080p60 (H.265) 11x 1080p30 (H.265)
Power	7W-15W

Refer to the Software Features section of the latest NVIDIA Jetson Linux Developer Guide for a list of supported features.

	Reference Carrier Board
Camera	2x MIPI CSI-2 22-pin camera connectors
PCIe	M.2 Key M slot with x4 PCIe Gen3 M.2 Key M slot with x2 PCIe Gen3 M.2 Key E slot
USB	USB Type-A connector: 4x USB 3.2 Gen2 USB Type-C connector for UFP
Networking	1xGbE connector
Display	1x DP 1.2 (+MST) connector
Other I/O	40-pin expansion header (UART, SPI, I2S, I2C, GPIO) 12-pin button header 4-pin fan header microSD slot DC power jack
Mechanical	100mm x 79mm x 21mm (Height includes feet, carrier board, module, and thermal solution)

Figure: Jetson Orin Nano 8GB Technical Specifications

The Jetson Orin Nano is able to extract BVP and BPM using both a sequential and parallel approach. The sequential approach is the same as the one used in the Jetson Nano experiments. Its advantage is a more stable BVP and BPM reading, while its disadvantage is a lower run speed as we can see in Table 2. The parallel approach analyses frames as they are recorded. This means that if the device freezes for any reason the video is not continuous. In practise this means that this type of analyses is faster, but the results provided have a higher degree of variation as we can see in Table 3.

4.4.5 Sequential Evaluation of Jetson Orin Nano 8 GB

As we can see from Appendix 2, the time to extract the BVP and BPM is significantly reduced when compared to the Jetson Nano 4GB with an average run time of 66.02 +-0.44 seconds for 900 frames or 30 seconds of video. Additionally, we extract an average of 86.3 +-9.48 BPM over 99 test runs. Indicating that everything is functioning as expected, albeit at a in a not so timely manner.

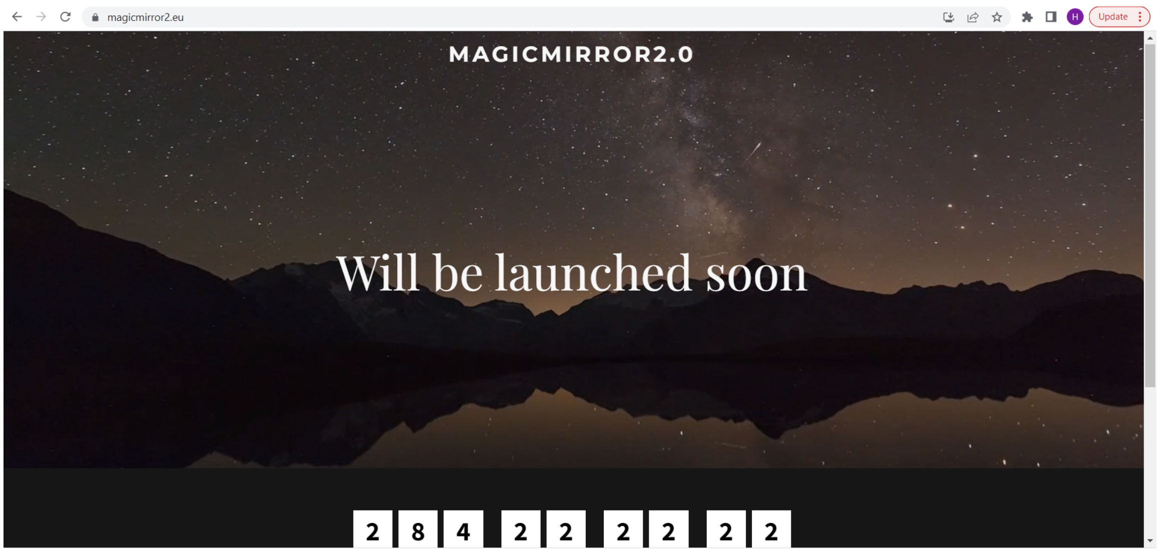
4.4.6 Parallel Evaluation of Jetson Orin Nano 8 GB

As we can see from Appendix 3, the time to extract the BVP and BPM is significantly reduced when compared to the Jetson Nano 4GB and the Jetson Orin Nano 8GB with the sequential methodology. An average run time of 44.63 +- 2.02 seconds for 900 frames or 30 seconds of video. We also note a more volatile measurement for BPM, with an average of 83.68 but with a standard deviation of 15.06 over 152 runs.

Detail analysis of the benchmarking trials and the data we collected can be found in the Deliverable D6- Proof of Concept report for the edge4iwelli.docx in the relevant annexes.

5 Exploitation

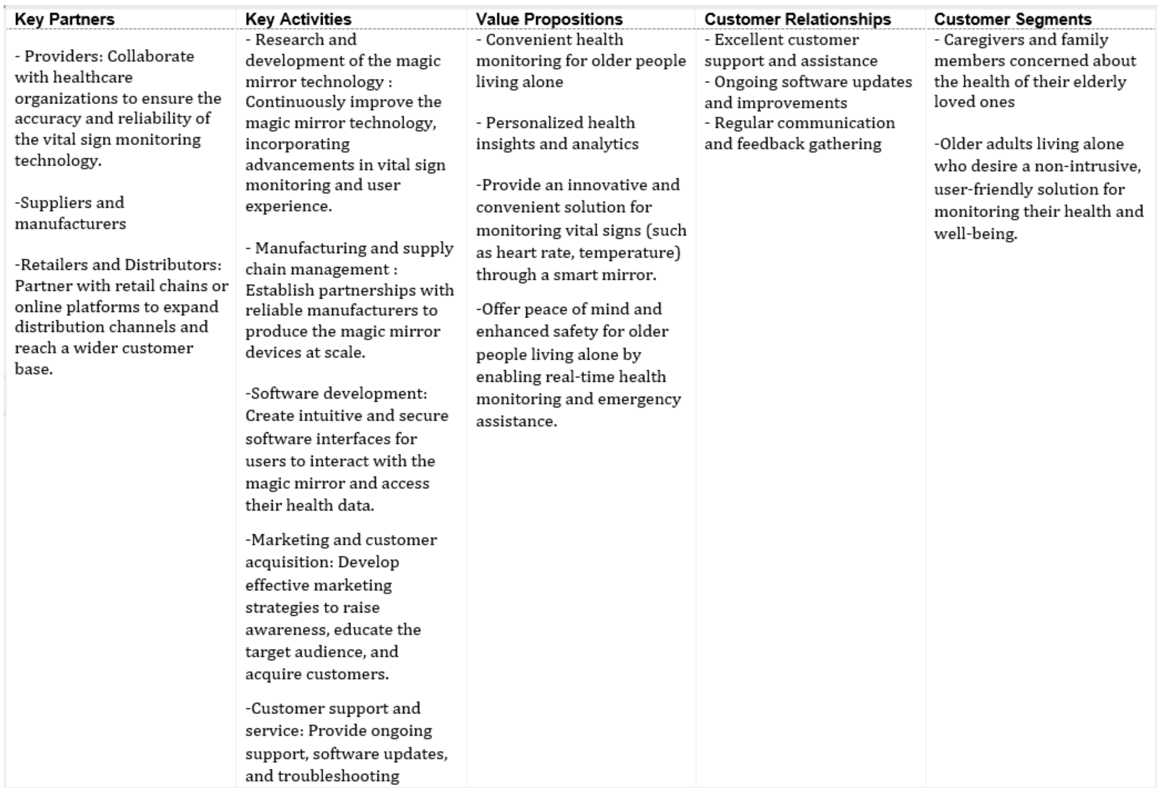
Syndesis intends to exploit the health smart mirror device to expand its health care services to more target groups and industries like the tourism industry targeting the high end market eg 4-5 star Hotels, high-end Yachts, Cruise Ships etc as well as stores and supermarkets to monitor customers for eg. infections.
A portal will be launched describing the main functions of the health smart mirror and provide information for the product, ability to buy online etc.
The www.magicmirror2.eu domain name will be used for that purpose.



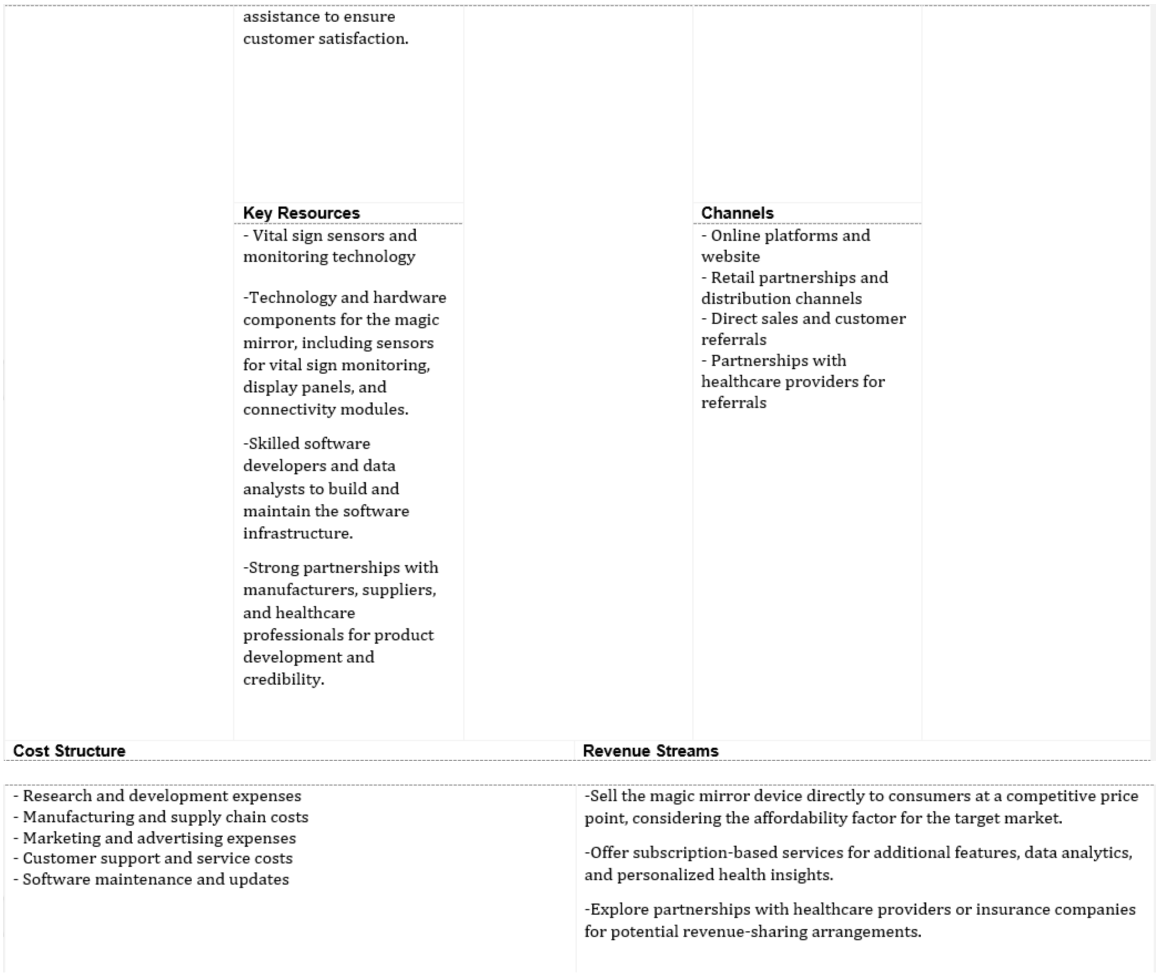
5.1 Magic Health Mirror Business modelling

By implementing a business model, the magic health mirror can effectively target older people living alone, offering them a valuable solution for health monitoring and enhanced safety while also ensuring long-term sustainability and growth for the business.

The analysis we have conducted created a business model canvas that provides an overview of the key elements involved in the magic health mirror business, including the value proposition, target customer segments, revenue streams, and key activities. It serves as a visual representation of the various components necessary for a successful business model in the context of monitoring health parameters in older people.



Key Partners	Key Activities	Value Propositions	Customer Relationships	Customer Segments
- Providers: Collaborate with healthcare organizations to ensure the accuracy and reliability of the vital sign monitoring technology. -Suppliers and manufacturers -Retailers and Distributors: Partner with retail chains or online platforms to expand distribution channels and reach a wider customer base.	- Research and development of the magic mirror technology : Continuously improve the magic mirror technology, incorporating advancements in vital sign monitoring and user experience. - Manufacturing and supply chain management : Establish partnerships with reliable manufacturers to produce the magic mirror devices at scale. -Software development: Create intuitive and secure software interfaces for users to interact with the magic mirror and access their health data. -Marketing and customer acquisition: Develop effective marketing strategies to raise awareness, educate the target audience, and acquire customers. -Customer support and service: Provide ongoing support, software updates, and troubleshooting	- Convenient health monitoring for older people living alone - Personalized health insights and analytics -Provide an innovative and convenient solution for monitoring vital signs (such as heart rate, temperature) through a smart mirror. -Offer peace of mind and enhanced safety for older people living alone by enabling real-time health monitoring and emergency assistance.	- Excellent customer support and assistance - Ongoing software updates and improvements - Regular communication and feedback gathering	- Caregivers and family members concerned about the health of their elderly loved ones -Older adults living alone who desire a non-intrusive, user-friendly solution for monitoring their health and well-being.



	assistance to ensure customer satisfaction.			
	Key Resources		**Channels**	
	- Vital sign sensors and monitoring technology -Technology and hardware components for the magic mirror, including sensors for vital sign monitoring, display panels, and connectivity modules. -Skilled software developers and data analysts to build and maintain the software infrastructure. -Strong partnerships with manufacturers, suppliers, and healthcare professionals for product development and credibility.		- Online platforms and website - Retail partnerships and distribution channels - Direct sales and customer referrals - Partnerships with healthcare providers for referrals	

Cost Structure	**Revenue Streams**
- Research and development expenses - Manufacturing and supply chain costs - Marketing and advertising expenses - Customer support and service costs - Software maintenance and updates	-Sell the magic mirror device directly to consumers at a competitive price point, considering the affordability factor for the target market. -Offer subscription-based services for additional features, data analytics, and personalized health insights. -Explore partnerships with healthcare providers or insurance companies for potential revenue-sharing arrangements.

Business Model Canvas for the edge4iwelli

To support the business model we have created the following Key Metrics to monitor the business success:

- Number of magic health mirror devices sold
- Customer acquisition and retention rates
- Subscription renewal rates
- Customer satisfaction and feedback
- Revenue growth and profitability

5.2 Cloud based Services that will complement the magic health mirror

The customers of the Magic Health Mirror will be able to assign to the whole service that the iwelli platform will offer to them with extra cost. More specifically the health and wellness services that will support the Magic Health Mirror business model:

After starting up of the Edge4iwelli system at home, runs continuously. Most of time in stand-by mode, but wakes up at a predefined interval and performs the upload job.

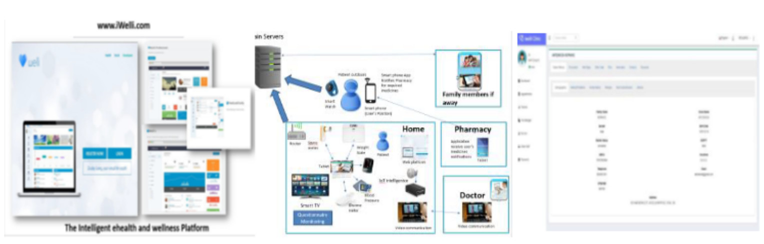
All health-related data originated within home: Measurements, Person Events, Alerts will be stored and processed before uploading to the web. Home-to-server upload is an in-home software service (daemon-like) responsible for proactively uploading monitoring data and other relevant data (e.g. data-fusion produced high-level events) from the local database to the med.iwelli.com and to the mydata.iwelli.com servers. Allows configuration of uploading intervals and (rule-based) what data to upload. The API that will be used to upload the data will be the FHIR API.

Personal Health Record

The patients can store their data either in their mobile phone or upload their data in their own cloud space personal health record in the Fhir PHR mydata.iwelli.com portal. The med.iwelli.com and the mydata.iwelli.com platforms, based on artificial intelligence, allow patients to both obtain a personalised risk prediction and personalized guidance for their treatment plan management.

Electronic Health Record

The Fhir EHR med.iwelli.com platform is part of the iwelli.com ecosystem of applications enabling doctors to store heterogeneous data (vital signs from their patients, documents, pictures like MRI, photos of medical exams, document file types such as pdf, textual content like the medical history of the patient, financial records, audio files and many others).



Med.iwelli.com FHIR EHR

The med.iwelli.com is more than an EHR since it allows for more complex data to be captured and stored.
The doctors can access the med.iwelli.com cloud based intelligent EHR to review test results, lab results, allergies, prescriptions medications, immunization history, and other health information. They can analyze data with calculators and machine learning to understand recorded Data, link it to clinical outcomes and identify the health scores of their patients and identify patients risks by correlating collected data sources. They can schedule and manage appointments, including in-person visits and video visits. The med.iwelli.com addresses the health care challenges of patient monitoring and **Risk-stratification and Decision Support.**

5.3 Analysis of the whole service that the magic health mirror will be part of it

Competitors and competitive solutions: Having performed a market search we have identified the key advantages of the full iwelli home care service with the magic health mirror compare to the competition which partly covers the iwelli services since IoT intelligent services for older population are still not mature. The most disruptive digital health solutions9 are summarized to highlight iwelli care package competitive advantage. The online competitive health platforms support monitoring, targeting wellness, disease self-management market (chronic diseases) and urgent care dramatically lowering health care costs. Some of them eg teladoc, have launched medical AI consultation based on personal medical history and common medical knowledge, while others simplify medical administration and refer patients to the right level of care. Most of the existing platforms connect doctors with patients for live, on-demand video visits over the internet where doctors diagnose the patient and then write prescriptions or refer to a specialist if required and handles all the administration, security, and record keeping that modern healthcare requires. Furthermore these platforms offer APIs to connect monitoring devices with their cloud based platforms although they have adopted a closed ecosystem approach (data belong to the vendor).

Table / Company	iwelli	Indicative Competitors:

9 www.healthtap.com, www.americanwell.com, www.teladoc.com, www.zocdoc.com, www.idoc24.com, http://angels.uams.edu, www.avizia.com, www.babylonhealth.com, www.doctorondemand.com

competitive Info solution		www.rempark.eu, www.aal-europe.eu. www.safemove-project.eu, Samsung IoT, LG IoT, Apple health kit
Product/Service: Holistic approach - support many use cases and consolidate IoT and eHealth into one single platform / Address Medical needs of elderly: prevent the consequences of adverse health events. / Address Social care needs, Social Connection to family friend, entertainment, novel business home care model, Family involvement:1.tight link with family members 2.Immediate notification in abnormal situations	X	Partially support. None of the existing technologies covers all as iwelli intends to do
Technology: One single point of access (plug and play approach). / Research developments (mirror, actuators easily adaptable and customizable)	X	-
Price: Low Cost self-management home care services	X	-
Technical Characteristics: Open source platforms (universaal), Web based technology, open APIs, intelligent IoT systems in house	X	Usually the products in the market have adopted a closed ecosystem approach (data belong to the vendor)
Main advantages: Unobtrusive technology. / Easy to use and user-friendly / Technology innovative sensors: for daily monitoring (smart mirror), Intelligent algorithms (in the cloud but in the in house modules)	X	-
Main weakness(es): Security / scalability / Content for developers	-	X (the full iwelli homecare service with the magic health mirror covers these weaknesses and goes the market)

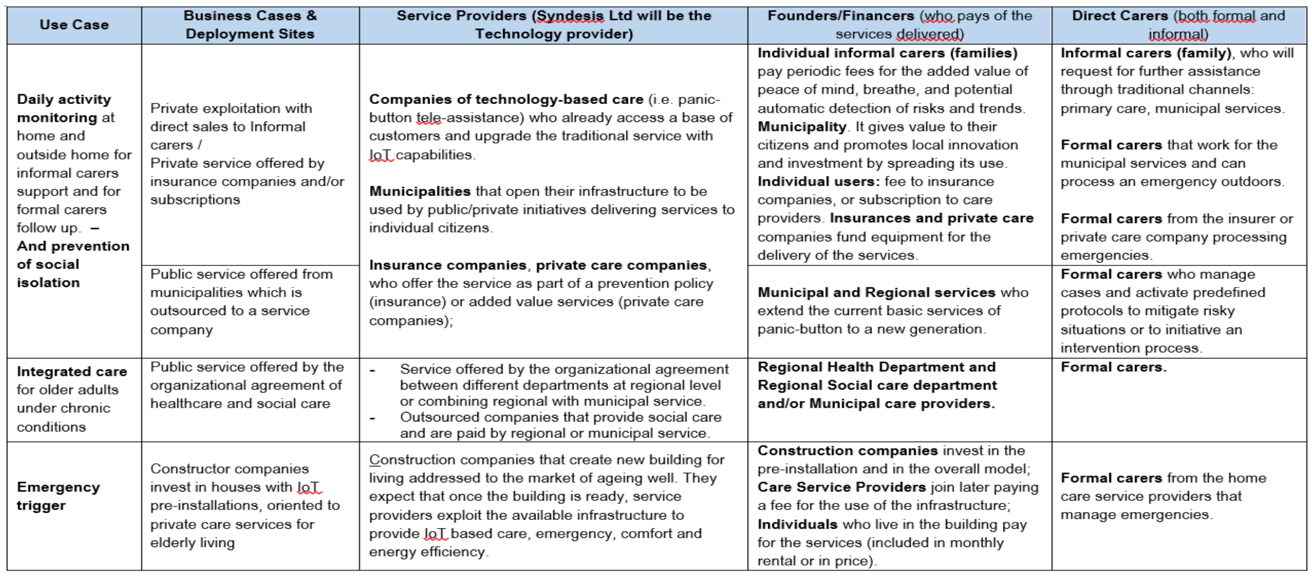
Comparison of the full iwelli home care service with the magic health mirror with indicative competitors

Our approach is unique from that of the competitors who have attempted to engage healthcare professionals and support ehealth services selectively. In contrast, we are trying to achieve meaningful self-management results by integrating multiple new proprietary technologies and open source platforms, heterogeneous but easily implemented services and mix of business models to engage healthcare professions.

Target Market : In Europe, 9 people out of 10 die[10] of a chronic disease (heart disease, stroke, cancers, respiratory diseases, diabetes, hypertension, allergies, kidney and liver diseases). 52 million EU citizens aged 55-74 report[11] having a longstanding illness or health problem which is about half of all people in this age group, while around 32.0 % of all the EU residents declare[12] having a chronic disease. Although potential customers of the iwelli toolset and services are all EU residents having a chronic disease and need a long lasting self-management home care service, iwelli commercial services and marketing startegy will initially target *elderly people (65+)* present with multiple chronic diseases living

[10] http://www.alliancechronicdiseases.org/home/
[11] http://ec.europa.eu/health/major_chronic_diseases/docs/rivm_report_retirement_en.pdf
[12] http://ec.europa.eu/eurostat/statistics-explained/index.php/Quality_of_life_in_Europe_-_facts_and_views_-_health#cite_note-15

independently that should adhere to long term medical plans. *Elderly people* present with multiple chronic diseases and often require complex long-term treatment. It is estimated that 50 million European citizens suffer from multi-morbidity13 which account for more than two-thirds of healthcare costs[14]. Among people over the age of 65 about 65% has multiple chronic diseases; among people over the age of 85 this is estimated[15] at 85%. Furthermore, the elderly are the greatest consumers of prescription drugs and are responsible for 60% of medication-related costs[16].
The business cases for the full iwelli homecare service with the magic health mirror for the care recipients (older people who live independent have been identified as following:



Use Case	Business Cases & Deployment Sites	Service Providers (Syndesis Ltd will be the Technology provider)	Founders/Financers (who pays of the services delivered)	Direct Carers (both formal and informal)
Daily activity monitoring at home and outside home for informal carers support and for formal carers follow up. – **And prevention of social isolation**	Private exploitation with direct sales to Informal carers / Private service offered by insurance companies and/or subscriptions	**Companies of technology-based care** (i.e. panic-button tele-assistance) who already access a base of customers and upgrade the traditional service with IoT capabilities. **Municipalities** that open their infrastructure to be used by public/private initiatives delivering services to individual citizens. **Insurance companies**, **private care companies**, who offer the service as part of a prevention policy (insurance) or added value services (private care companies);	**Individual informal carers (families)** pay periodic fees for the added value of peace of mind, breathe, and potential automatic detection of risks and trends. **Municipality**. It gives value to their citizens and promotes local innovation and investment by spreading its use. **Individual users:** fee to insurance companies, or subscription to care providers. **Insurances and private care** companies fund equipment for the delivery of the services.	**Informal carers (family)**, who will request for further assistance through traditional channels: primary care, municipal services. **Formal carers** that work for the municipal services and can process an emergency outdoors. **Formal carers** from the insurer or private care company processing emergencies.
	Public service offered from municipalities which is outsourced to a service company		**Municipal and Regional services** who extend the current basic services of panic-button to a new generation.	**Formal carers** who manage cases and activate predefined protocols to mitigate risky situations or to initiative an intervention process.
Integrated care for older adults under chronic conditions	Public service offered by the organizational agreement of healthcare and social care	- Service offered by the organizational agreement between different departments at regional level or combining regional with municipal service. - Outsourced companies that provide social care and are paid by regional or municipal service.	**Regional Health Department and Regional Social care department and/or Municipal care providers.**	**Formal carers.**
Emergency trigger	Constructor companies invest in houses with IoT pre-installations, oriented to private care services for elderly living	Construction companies that create new building for living addressed to the market of ageing well. They expect that once the building is ready, service providers exploit the available infrastructure to provide IoT based care, emergency, comfort and energy efficiency.	**Construction companies** invest in the pre-installation and in the overall model; **Care Service Providers** join later paying a fee for the use of the infrastructure; **Individuals** who live in the building pay for the services (included in monthly rental or in price).	**Formal carers** from the home care service providers that manage emergencies.

Business Cases for the full iwelli home care service with the magic health mirror

Business Model of the full iwelli homecare service with the magic health mirror:
Although the detail analysis of the business model will be conducted within the realm of the project, major determinants of the business model for the full iwelli home care service with the magic health mirror have been identified as following:

Target Market	**What is the need**
Potential customers of the iwelli care package and services are all individuals having: • an emerging and moderate risk to develop a medical condition such as a chronic disease and need to start a wellness programs to manage eg increased cholesterol, monito diet, increase activities like walking or decrease anxiety and stress etc, • or at least one medical condition eg chronic disease and need to adhere to a Disease self-management program. 52 million EU citizens aged 55-74 report having a longstanding illness or health problem which is about half of all people in	• Anxiety for their health • Early-Risk identification • Concern to adhere to their plan/objective • Cost concern

[13] http://ec.europa.eu/health/ageing/docs/ev_20151007_frep_en.pdf
[14] Smith SM, Soubhi H, Fortin M, Hudon C, O'Dowd T. Managing patients with multimorbidity: systematic review of interventions in primary care and community settings. Cochrane Database of Systematic Reviews 2012; 4.
[15] Marengoni A, Angleman S, Melis R, Mangialasche F, Karp A, Garmen A, Meinov B, Fratiglioni L. Aging with multi-morbidity: a systematic review of the literature. Aging Research Reviews 2011; 10:430-439.

[16] http://apps.who.int/medicinedocs/en/d/Js4883e/7.3.html

this age group, while around 32.0 % of all the EU residents declared having a chronic disease. • Insurance package (private or public) It is estimated that more than 20% of the insured population that need to start a welness or a disease management program account for more than two-thirds of healthcare costs17.	
Product/Service	Benefits - unique proposition
• Automation of the self-management process of the healthcare program (digitization of the health service) • Shop in a shop (Marketplace of services) • Low cost but high quality expert information services (knowledge) • Telehealth (24/7 availability) – convenient and low cost service • Digital enhanced connectivity (patients-doctors-carers)	Convenience – time saving – Reminder – Empowerement – Forget about appointments – Pharmacy Knowledge analytics (visualization)
Channels to deliver the service	
• Solution/service providers (Doctors, insurance companies, Pharma, care providers, healthcare providers) • Direct selling through Doctors • Through Web	We develop a redistributed channel of revenues
Revenue models	
Revenue sharing with healthcare professionals : they will register their patients (a revenue share model will be applied with them) Layer player: Sell to external bodies eg, customize iwelli for hospitals Ecommerce (directly purchased by end users through the www.iwelli.com platform): • Cross selling – devices , 23and me, pharmacy, health wearables, • Hidden revenues (Ads) • Freemium (premium customers will pay for extra services) eg DSS Telehealth sessions, second opinion (Hospitals abroad) • Lock in model (diabetics disposables)	We offer the technology to coordinate revenue sharing

Determinants of Business models for the full iwelli homecare service with the magic health mirror

To account for differences in customer willingness to pay we have to choose a mix of revenue mechanisms. Indicative Revenue Mechanisms are following although experimentation with different revenue mechanisms to ensure the viability of the specific mechanisms are necessart:

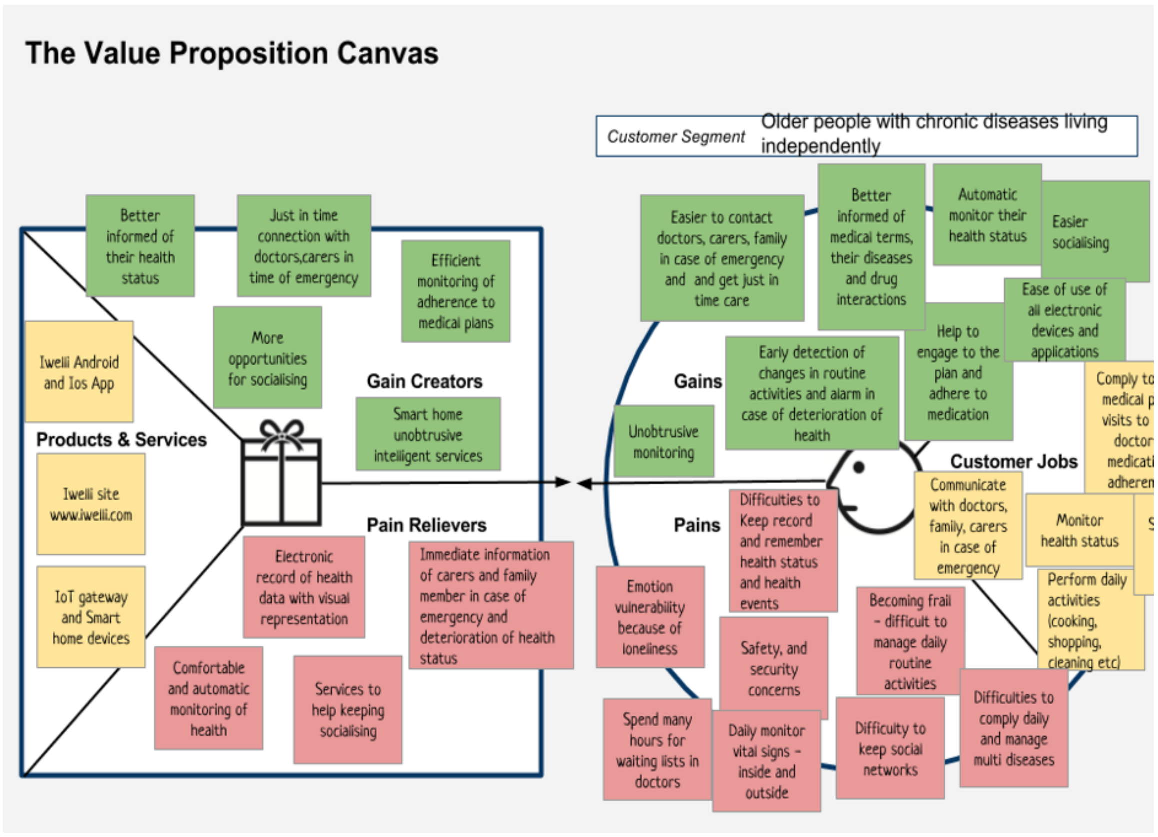
Type of digital offering	**Constraints**	**Revenue Mechanisms**
Basic monitoring services and cloud services (diagrams, history, storage of up to 1GB of data, communication functions, ehealth services, mobile application, atomization of the	Customers willingness to pay is extremely low	Advertising although ads alone may not be enough to generate the income needed

[17] Smith SM, Soubhi H, Fortin M, Hudon C, O'Dowd T. Managing patients with multimorbidity: systematic review of interventions in primary care and community settings. Cochrane Database of Systematic Reviews 2012; 4.

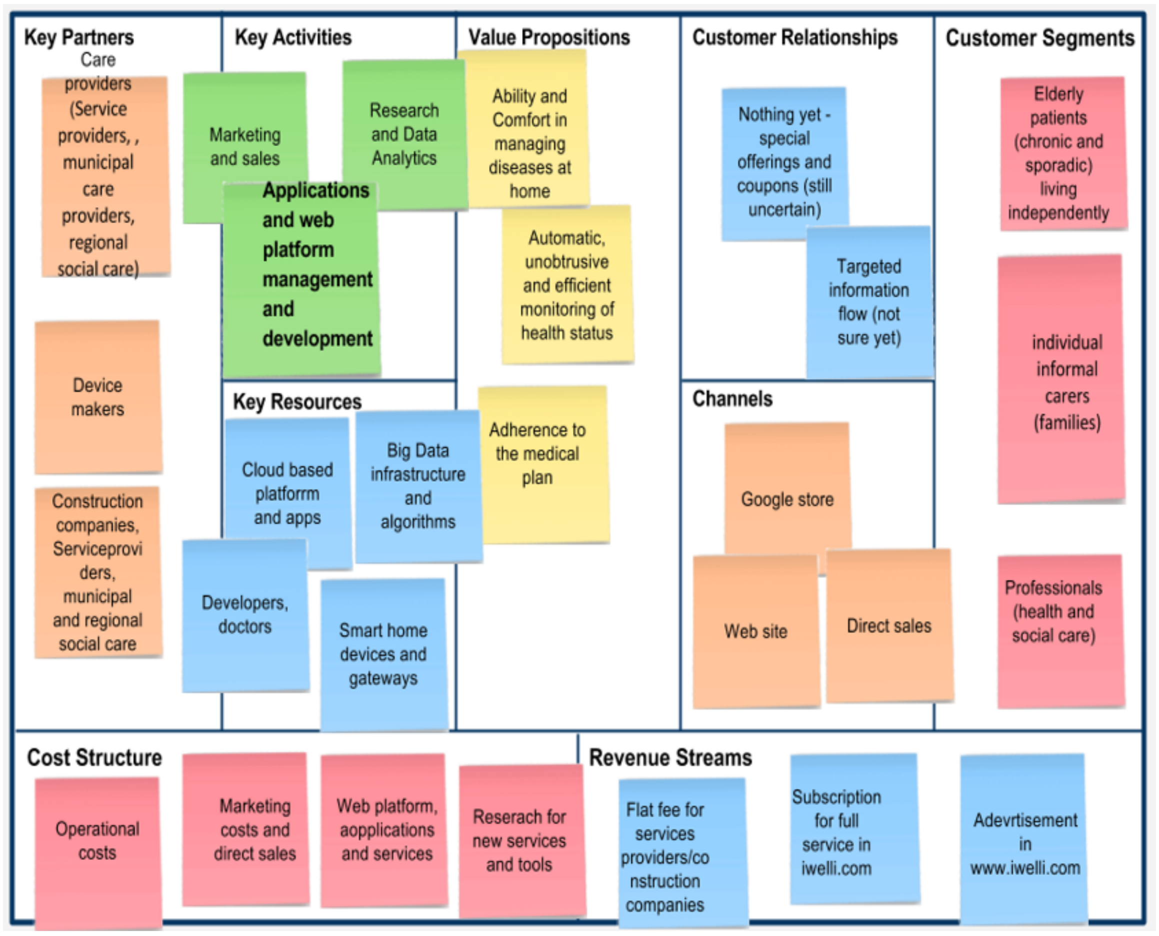
healthcare program, information services)		
Digitization of data (storage of up to 1GB of data)	Maybe - Not known in advance- differences in end users willingness to pay (depend on the disease)	Pay per use /basic package - Freemium
Decision support services	Maybe - Not known in advance	Pay per use for end users (free to doctors)
IoT care package		Subscription fee (low cost)
Added value services in the platform (eg consultation for Gene analysis)	willingness to pay (high end)	Pay per use

Revenue models for the full iwelli home care service with the magic health mirror

Furthermore a value proposition Canvas for this category of end users (older people) and a business model Canvas[18] have been constructed as following:


The Value Proposition Canvas
Customer Segment
Older people with chronic diseases living independently
Better informed of their health status
Just in time connection with doctors,carers in time of emergency
Efficient monitoring of adherence to medical plans
More opportunities for socialising
Gain Creators
Smart home unobtrusive intelligent services
Iwelli Android and Ios App
Products & Services
Iwelli site www.iwelli.com
IoT gateway and Smart home devices
Pain Relievers
Electronic record of health data with visual representation
Immediate information of carers and family member in case of emergency and deterioration of health status
Comfortable and automatic monitoring of health
Services to help keeping socialising
Easier to contact doctors, carers, family in case of emergency and and get just in time care
Better informed of medical terms, their diseases and drug interactions
Automatic monitor their health status
Easier socialising
Ease of use of all electronic devices and applications
Help to engage to the plan and adhere to medication
Early detection of changes in routine activities and alarm in case of deterioration of health
Gains
Unobtrusive monitoring
Comply to medical p visits to doctor medicati adheren
Customer Jobs
Communicate with doctors, family, carers in case of emergency
Monitor health status
Perform daily activities (cooking, shopping, cleaning etc)
Pains
Difficulties to Keep record and remember health status and health events
Emotion vulnerability because of loneliness
Safety, and security concerns
Becoming frail - difficult to manage daily routine activities
Spend many hours for waiting lists in doctors
Daily monitor vital signs - inside and outside
Difficulty to keep social networks
Difficulties to comply daily and manage multi diseases

[18] Osterwalder, Pigneur & al. (2010) Business Model Canvas: nine business model building blocks.


Key Partners
Care providers (Service providers, , municipal care providers, regional social care)
Device makers
Construction companies, Serviceproviders, municipal and regional social care
Key Activities
Marketing and sales
Research and Data Analytics
Applications and web platform management and development
Key Resources
Cloud based platforrm and apps
Big Data infrastructure and algorithms
Developers, doctors
Smart home devices and gateways
Value Propositions
Ability and Comfort in managing diseases at home
Automatic, unobtrusive and efficient monitoring of health status
Adherence to the medical plan
Customer Relationships
Nothing yet - special offerings and coupons (still uncertain)
Targeted information flow (not sure yet)
Channels
Google store
Web site
Direct sales
Customer Segments
Elderly patients (chronic and sporadic) living independently
individual informal carers (families)
Professionals (health and social care)
Cost Structure
Operational costs
Marketing costs and direct sales
Web platform, aopplications and services
Reserach for new services and tools
Revenue Streams
Flat fee for services providers/construction companies
Subscription for full service in iwelli.com
Adevrtisement in www.iwelli.com

Value proposition canvas and Business Model Canvas of the full iwelli home care service with the magic health mirror for the Older patients

<u>Commercialisation Roadmap for the full iwelli homecare service with the magic health mirror</u>

Syndesis envisages the close collaboration with a stakeholder community (e.g. Healthcare, Insurance, constructors for IoT Sectror) enforcing a full commercialization potential. An early evidence of Syndesis activities concern cooperation with health care providers to provide free of charge services to some of their patients using the full iwelli home care service without the magic health mirror. Municipalities Oriaokastro, Hrakleia to mention but a few, already have adopted the Iwelli ehealth services for their elderly citizens. This will help Syndesis to start a regional scaling of the full iwelli homecare service without the magic health mirror. A first commercialization roadmap has been constructed that provides an assessment of which countries/markets first should be addressed and the processes we should adopt to adjust to different cultural contexts and which steps we should take to financially support the large scale roll-out of the proposed methodology.

Indicative elements of the marketing plan for the full iwelli home care services with the magic health mirror
Place - The regulation issues, cultural issues, limited budget and competitive products made us decide that we will start our advertising actions in Greece within 2024. This choice has to do with the demographics of Greece which we believe will facilitate our marketing strategy.

Selected target groups: above 65 having at least 2 chronic diseases and need to adhere to long term therapies and polypharmacy
Customer groups and use cases: we will focus on Cardiovascular diseases
Market size (eg. Greece): Following OECD statistics (www.oecd.org/greece/47877676.pdf) approximately 18.7% of the Greek population is aged over 65 (OECD average 15%) with 4.3% of the population over 80 (OECD average 4%), while multimorbidity (at least two chronic diseases) of patients above 65 equals 27%. We expect a market size equal to 350.000 potential users of the iwelli services.
Market Share (Europe): The company intends to avoid aggressive sales tactics aiming to acquire not more than 1% share of the European market of the 45 million European citizens[19] suffer from chronic diseases (Cardiovascular, Diabetes, respiratory diseases) and are under self-management healthcare program. The exact numbers, as well as the number of the people we will target and are under a wellness program with a risk for chronic disease, the marketing and strategy to acquire this share will be calculated within the feasibility market study.
Agreements : We will include a series of targeted bilateral meetings with the relevant stakeholders in order to study the market and to reach preliminary strategic agreements like Joint-Venture or other forms of Collaboration. Already Syndesis discuss to sign MoUs with many regional companies eg. Sigmasoft[20] and Anats[21]. Exploitation of the results will also be demonstrated at selected business oriented events of IoT[22], CES[23], CEBIT[24] and Medica[25] to mention but a few.
Partners Acquisition Strategy: Syndesis will adopt a penetration pricing strategy to become cost leader in the above market. More specifically Syndesis with the iwelli ecosystem of tools and services like the clinic.iwelli.com platform the mobile apps and IoT gateways will adopt a hybrid model (selling products and services) and set up agreements with all possible vendors of the above market from all over the world.

Beyond the IoT for self management health services market with the ageing population magic health mirror intends to be positioned in many more markets in the next phase of development:

- The tourism high end market
- The Silver Economy market
- The market that concerns solution for stay safe from infections like Covid-19

For these markets the customers' acquisition strategy is the following: The business model strategy to create the critical mass of customers in B2B (high end yacht companies, Hotels and Cruise Ships) and B2C (direct selling to end users eg family members that taking care of older people) concerns the adoption of direct and indirect marketing methods.

[19] http://www.euro.who.int/__data/assets/pdf_file/0020/77510/RC56_edoc08.pdf
[20] https://www.sigmasoft.gr/iwelli-care/
[21] http://www.anats.gr/
[22] http://www.iot-world.fr/
[23] https://www.ces.tech
[24] http://www.cebit.de/en/#intro
[25] www.medica-tradefair.com

6 Summary and next steps

This report describes the development of the Edge4iwelli health smart mirror, which aims to promote home care services as a cost-effective strategy for healthcare. The Edge4iwelli project focuses on embeding edge computing systems and advanced AI modules into the existing iwelli IoT care package. By minimizing reliance on cloud services, the project aims to optimize home care services and improve the overall efficiency of the IoT care platform.

The Edge4iwelli smart health mirror transforms a regular mirror into a smart mirror enabling remote monitoring of users' physiological parameters through real-time video analysis.

The Edge4iwelli project and the health smart mirror aim to enhance home care services through the utilization of edge computing and AI. The development of a smart health mirror as part of the project allows for remote monitoring of users' physiological parameters in a cost-effective and energy-efficient manner.

The next steps involve further development and optimization of the smart mirror system, including software programming, integration of sensors, and testing with end users. Furthermore next steps involve the design of different sizes of end products (different sizes of mirrors), the agreements with vendors of microcomputers to manage lower cost of the devices, the set up of digital marketing activities and finaliseng the product site.

The markets that the health smart mirror will be positioned are:

- The IoT for self management health services market,
- The tourism high end market
- The Silver Economy market
- The market that concerns solution for stay safe from infections like Covid-19

The business development actions that Syndesis will adopt, beyond the advertising to create brand popularity for the health smart mirror, are Partners Acquisition Strategy and Customers’ acquisition Strategy.

VEDLIoT
Very Efficient Deep Learning in IoT

ICT-56-2020 — Next Generation Internet of Things

D4
FLAIR Final Report, User Manual and Code Drop

Version 2

Document Information	
Contract Number	957197
Project Website	`https://vedliot.eu/`
Dissemination Level	PU (Public)
Nature	M (Manual)
Contractual Deadline	31 June 2023
Author	Ilker Demirkol (UPC)
Contributors	Anass Anhari (UPC), Daniel Alamillo (UPC), Sebastià Vila (UPC), Francisco del Àguila (UPC)
Reviewers	Pedro Petersen (CHALMERS), Dr. Waqar (CHALMERS)

The VEDLIoT project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No 957197.

Table of Contents

1 Introduction

1.1 Motivation

The era of Big Data, thanks to the proliferation of the 7 billion IoT devices and 3 billion smartphones [5], allows the use of Machine Learning (ML)-based solutions for numerous applications. However, traditionally, to implement Machine Learning (ML) models, the data is transferred to a central server where it is stored and used for training and testing ML models. Nowadays, the main challenge **is that many IoT devices collect privacy-sensitive data**, for which there are growing consumer privacy concerns.

Consequently, the policy makers have imposed privacy and data-sharing policies such as the European Commission's Global Data Protection Regulations (GDPR) and the US Consumer Privacy Bill of Rights. In addition to the privacy-concerns, another challenge for IoT AI solutions is the **diverse computational and communication capabilities of IoT devices** (some with very limited hardware and communication resources), which would benefit from collaborative approaches to ML.

To tackle these issues, a promising solution is the Federated Learning (FL), where the end nodes, e.g., IoT devices, collaboratively train an ML model without sharing their private data. Federated Learning enables end nodes to cooperatively learn a shared prediction model while keeping all the training data on the device, decoupling the ability to do machine learning from the need to store the data in the cloud.

In many AI use cases, such as voice activated assistants, the training stage is done on a central server and then trained models are deployed to end-user devices. However, using a FL approach, the training procedure could be done locally on each client, obtaining a local model for each one. Furthermore, a central server would aggregate all the different models deploying the global model to the end-clients. Therefore, the global model would be continuously updated, and most importantly, preserving the privacy of the local data of each client.

1.2 The FLAIR solution

Many applications can benefit from FL, such as healthcare, smart home, and transportation systems. Based on the importance of FL for many IoT use cases, and the absence of such a privacy feature within the VEDLIoT solutions, our first objective in the FLAIR project is to implement and develop a software architecture within a FL scenario.

On top of the FLAIR software solution for FL, the project will implement a voice controlled IoT devices use case (based on speech command recognition). The main objective is to classify speech data collected from a user, in other words, to train a ML model for classifying voice commands. Since user's speech data are mostly considered as private information, a FL solution is the only possible privacy-preserving solution.

Different IoT devices can collaborate to create a global model without sharing their local data. There are several datasets available for voice commands, such as the one from the TensorFlow Speech Recognition Challenge 4. Hence, no new data collection is necessary within the FLAIR project. The FL clients are assigned the different speakers data, which will represent their local data that is not shared with the global FL server.

One of the main challenges of a FL solution is the communication overhead produced by the repetitive model parameter exchanges. This overhead can be significant based on the model size and the total number of rounds. For this reason, the FLAIR project will implement the FL use case into a 5G communication network. This mobile communication technology is currently being deployed worldwide and is the planned successor to the 4G networks. 5G has higher bandwidth and download speeds, hence, it will be interesting to use it in a real FL application. In addition, it also allows implementing virtual networks (subnetting, etc.) supporting a larger number of simultaneously connected devices compared to 4G networks. One of the use cases targeted with 5G is IoT, namely massive Machine Type Communication (mMTC). There have been several enhancements applied to 5G for such a support. For example, the number of devices a Base Station can support has been increased, the medium access methods of the devices are redesigned to support intermittently connecting devices, etc. Overall, 5G network implementation will benefit many IoT applications.

2 System architecture

2.1 Context

Federated Learning reproduces a centralized machine learning scenario but in distributed client nodes. These client nodes contribute to learning of a global model without sending any data to the central server. The main function of the server is to aggregate client local updates, using a specific aggregation method.

The popularization of ML-based solutions, due to their great performance in the multitude of use cases has produced the arrival of a variety of tools that implement it, such as TensorFlow, Scikit Learn, Theano, Pytorch...However, most of it does not include native implementation of Federated Learning (apart from TensorFlow, which has TensorFlow Federated only for simulation purposes). After some research, the FLAIR team decided to develop the system architecture with the Flower FL framework ([1]).

Flower framework enables a more seamless transition from experimental simulation to system research on real edge devices and offers a stable, language and ML framework agnostic implementation of the core components of an FL system. Flower provides higher-level abstraction, meaning that it simplifies and streamlines the process of implementing the core components of an FL system, such as federated aggregation and communication, without being tied to any particular machine learning framework. This allows researchers to experiment and implement new ideas on top of a reliable stack. So, the combination of Flower and an ML-tool, such as TensorFlow, will allow implementing a real FL scenario.

Conceptually, in the FLAIR project there are two main challenge that need to be faced:

- Model creation, validation and implementation $\rightarrow$ `TensorFlow`
- Federation implementation and its configuration $\rightarrow$ `Flower`

This pair of tools cover the project requirements. Various simple FL setups were tested to check the viability of the framework ensuring that it covers the desired objectives and requirements.

2.2 Overall architecture

In this section, we present the system architecture design, which is independent of the hardware setup. According to the Federated Learning architecture, our system will have two types of nodes as shown in Figure 2.1:

- **Aggregator server**: It will be in charge of sharing the global model to the clients at the beginning of the federation and update it at every training iteration from the selected clients results, using a federated aggregation algorithm (e.g. Federated Averaging). In other words, it will locally aggregate the tunned models of the selected clients into a new global model iteratively until convergence or some fatal error.

- **Collaborator client**: By design, it is the only component that has access to the local data. It recieves the global model from the Aggregator Server and its main role is to train locally the shared global model with its local data and inform back to the server the updates of the training results.

Every local training iteration produced at the client nodes is called epoch and every model aggregation produced at the server node is called round.

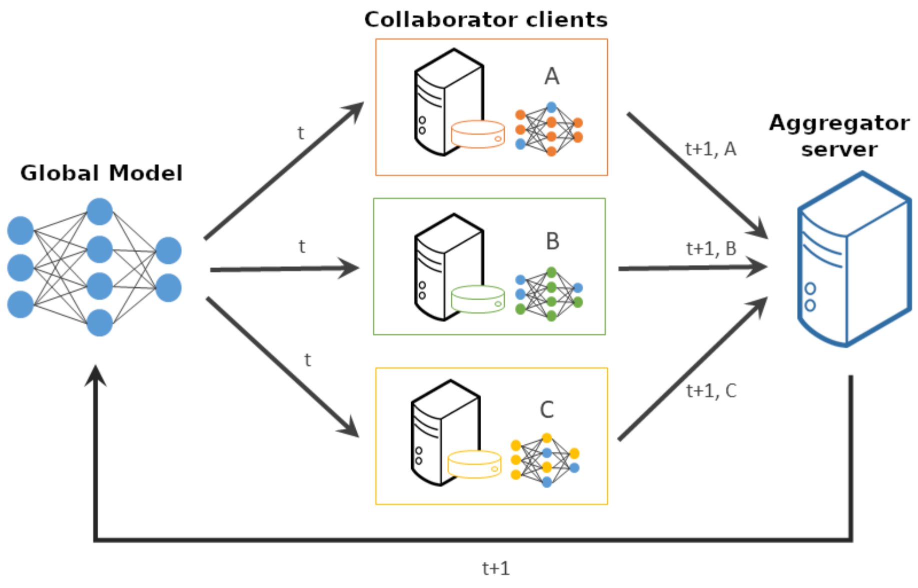


Figure 2.1. System Architecture conceptual diagram. t is the round number, A, B and C are the client's locally trained models.

In this project, we will reproduce this conceptual architecture design in various hardware setups. However, there are some other aspects related with connection quality and hardware acceleration that can alter the system behaviour such as the convergence time based on the connection quality and the number of available clients.

Such scenarios are defined to evaluate the FL performance on realistic scenarios, enabling/disabling the accelerators at the IoT devices, and the effect of having various IoT devices with differents connections (narrow-band vs. wideband).

Finally, illustrated in Figure 2.2, we have an overall view of the software structure and its related hardware components. Concretely, we can divide the entire architecture for the FL implementation and integration in three main structure components:

- Server: It will have the Flower framework integrated with to obtain the FL server software. Therefore, the server will have the global aggregated model and a specific evaluation data.

- 5G Base Station & 5G Core: As we have said, the FLAIR's 5G network implementation will use open-source software for the nodes. Specifically, we will use OpenAirInterface [2] for the 5G Base Station and the 5G Core Network. Moreover, the 5G Base Station will use an FPGA-based SDR (*Software Defined Radio*) allowing the use of software for the modulation/demodulation and the processing of radio signals.

- 5G IoT Device(s): Several IoT devices with the Flower framework, to obtain the FL clients software.

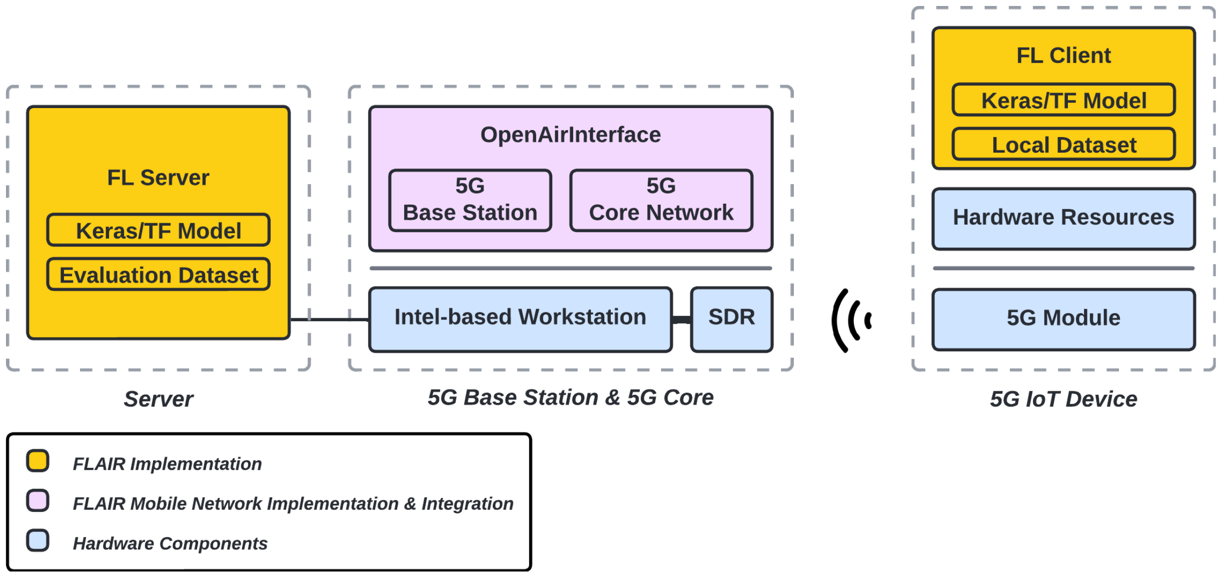


Figure 2.2. Overall architecture and its components

3 Software architecture

First of all, for implementing a federated learning solution to our problem we need to fully understand the background of *TensorFlow*, as it is the most common ML framework.

TensorFlow: is a framework to develop and train machine learning models, providing a high-level API and a low-level API.

Keras: is high-level deep learning API library for neural networks build to be modular and developing and implementing models faster.

A key difference is that Keras has a simple architecture and needs a backend engine to perform low-level computations. TensorFlow has a more capable and complex architecture, allowing a more precise debugging (challenging to do) and also managing low-level computations generating computational graphs and so on.

Briefly, Keras provides high-level building blocks for developing deep learning models. It does not handle low-level operations (convolutions, tensor products, etc.). For this reason, "other" libraries and frameworks can act as a "Keras backend engine" implementing and optimizing the low-level operations. This means that a Keras Model can be plugged into different backend engines, for now:

- TensorFlow (strongly active and maintained)
- Theano (discontinued in 2017)
- CNTK (discontinued in 2019)

However, the latest Keras release (*v2.9.0*) defines that "Keras is a high-level API of TensorFlow2", meaning that we should use the bundled Keras Version of TensorFlow.

3.1 Data preprocessing and Models

For speech keyword detection use case we have used the Google Speech Commands v2 dataset [8], which is a dataset and benchmark for speech recognition developed by Google. It consists of a collection of 105000 short audio clips, each containing a spoken command such as *stop*, *go* or *yes*. The dataset encompasses a total of 35 different keywords, and these keywords are spoken by various speakers. However, in various experimental scenarios we will keep only a few subset of commands, specifically to study the data heterogeneity. Our solution contemplates different dataset modes:

- Full Dataset Mode (`full_ds`): The 35 commands from the original dataset are kept.
- 30 commands mode (`30cmd`): Only 30 commands are kept.
- 26 commands mode (`26cmd`): Only 26 commands are kept.
- 20 commands mode (`20cmd`): Only 20 commands are kept.

In the dataset, each speaker contributes at least one audio file, and it is possible for a speaker to contribute multiple audio files containing different or the same commands. This diversity in speakers helps capture variations in pronunciation, accents, and speech patterns, making the dataset more representative of real-world scenarios. We will use this speaker variability to generate non-IID data partitions in a federated learning setup (only in simulated scenarios).

It is worth noting that not every speaker in the dataset covers all 35 commands. Only a subset of speakers contains audio files for all the commands in the dataset. The commands subset selection of every dataset mode has been done to maximize the number of speakers that contain samples from all the commands.

This variation in command coverage among speakers adds another level of complexity to the data preprocessing. We will add another non-IID dimensionality splitting data by commands, filtering the data to keep only the speakers that contain all the commands in the dataset.

So, our dataset preprocessing implementation allows three data spliting methods:

- IID data: Same amount of data samples for each client (tested in real network setups).
- Non-IID data seggregated by speakers: Each client has a fixed number of speaker in its local data (tested in simulation setups).
- Non-IID data seggregated by speakers and commands: Each client has a fixed number of speakers in its local data and train only a fixed number of commands (tested in simulation setups).

The model used in our solution to classify speech keywords is MobileNetV2 [7]. It is an image classification model, so it is necessary to convert the audio files to mel-spectrograms before introducing them into the model. A mel-spectrogram (Figure 3.1) is a visual representation of the power spectrum of an audio signal, where the frequency axis is transformed to better align with human auditory perception.

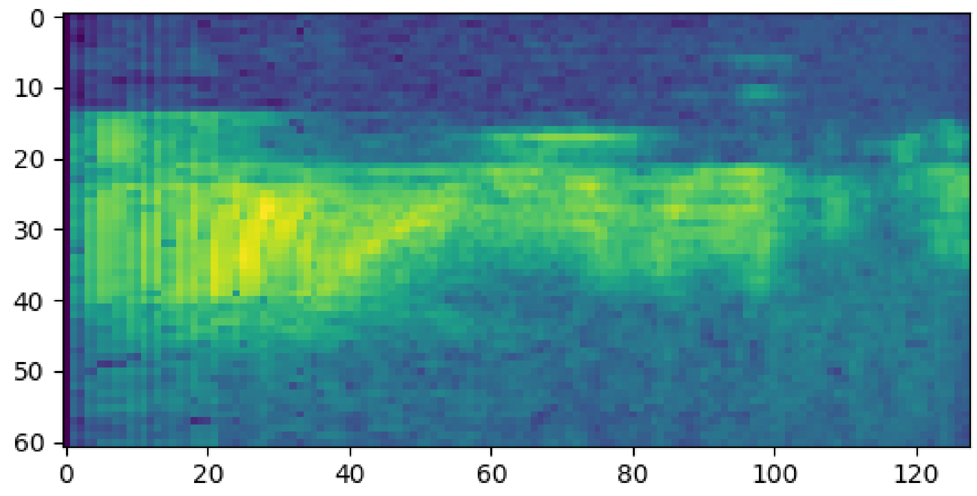


Figure 3.1. Generated mel-spectrogram ('go' command from the speaker '0a2b400e')

At the early stages of the project, another model (a simple convolutional neural network (CNN), from TensorFlow tutorial) was tested in a centralized learning environment to prove its efficiency in our use case. We could observe that MobileNetV2 offers a better performance in speech keyword detection task. For this reason, the federated evaluations provided in this report are all the output of MobileNetV2 model. However, our solution code also allows to load the simple CNN as model of the federated learning system.

MobileNetV2 is a lightweight convolutional neural network (CNN) architecture developed by Google. It is specifically designed for efficient deployment on mobile and embedded devices with limited computational resources. In general, MobileNetV2 achieves a good balance between model size and accuracy. Keras API provides an already implemented version of MobileNetV2 and allows to modify the input and the output layer to adapt it to a wide variety of image classification tasks (Code Snippet 17). More information about Keras MobileNetV2 at `https://keras.io/api/applications/mobilenet/`.

The data preprocessing and model loading has been implemented into two modules, `DatasetClasses.py` and `DatasetPreprocessing.py`, located in `src/common/dataset` directory. The functionalities in these modules are controlled through `src/common/config/Settings.py` file, which contains configuration options that affect the behavior of the whole FLAIR solution (more detail about `Settings.py` options in Chapter 5). Furthermore, the data pipeline is implemented through `tf.keras.utils.Sequence` in `SpeechGenerator.py` module. The `SpeechGenerator` object loads dynamically the audio files in batches, preventing memory overflow.

3.1.1 `DatasetClasses.py`

`DatasetClasses.py` module gives an object-based perspective of the speech keyword detection task. It implements the following Python dataclasses:

- `Label`: Represents a label of the dataset. It defines the following attributes:
  - `str_id: str` → The alphabetical name of the label (e.g. 'go', 'left'...).
  - `num_id: int` → The numerical value assigned to the label. It varies depending on the chosen `DatasetMode`.
  - `path: str` → The path where the labels audio files are located. By default, it is set to `DATASET_PATH/self.str_id`, where `DATASET_PATH` represents the default dataset path specified at `Settings.py`.
- `DatasetMode`: Represents the mode and version of the dataset and decide the subset of `Labels` (aka. voice commands) that will be used.
  - `mode: str` → String representing an specific dataset mode. The modes available are: `full_ds`, `30cmd`, `26cmd`, `20cmd`. By default it is set to `DATASET_MODE` from `Settings.py`.
  - `ds_version: int` → Decides which version of Google Speech Commands is used (v1 or v2). Only v2 has been used in our evaluations. By default it is set to `DATASET_VERSION` option from `Settings.py`.
  - `labels: set` → Set of `Label` objects corresponding to the chosen `self.mode`.

- `DatasetSamples`: Represents the dataset samples for the chosen `DatasetMode`. Provides methods for listing speakers, filtering speakers, getting speaker samples and getting all the samples in the dataset mode. The defined attributes and methods are:
  - Atributes:
    - `ds_mode: DatasetMode` → The Dataset mode.
    - `samples: Dict[str, Label]` → Samples dictionary containing files path as keys and its associated `Label` object as values.
  - Methods:
    - `list_speakers() -> Dict[str, Set]`: Returns a dictionary that contains all speaker IDs (keys) and their related label objects (values). It iterates over the samples and extracts the speaker ID from the sample paths. If a speaker ID is encountered for the first time, it creates a new entry in the dictionary. If the speaker ID already exists, it adds the label to the existing set of labels for that speaker ID. The resulting dictionary is returned.
    - `filter_speakers() -> List[str]`: Returns the list of speakers that contain all the speech commands defined in the current DatasetMode. It obtains all the commands from the `ds_mode.labels` attribute and compares them with the labels associated with each speaker in the dataset. If a speaker's labels match all the commands, the speaker ID is added to the filtered speakers list. The list of filtered speakers is returned.
    - `get_speaker_samples(sp_list)`: Returns a dictionary containing the speaker IDs from `sp_list` as keys and their samples dictionary (`spec_path0: label0, spec_path1: label1, ...`) as values. It iterates over the samples and extracts the speaker ID from the sample paths. If the speaker ID is in the `sp_list`, it adds the sample path and corresponding label to the speaker's samples dictionary. The resulting dictionary is returned.
    - `get_samples(filter_speakers, val_split, test_split)`: Retrieves the training, validation, and testing samples. If `filter_speakers` is set to True, it filters the speakers based on the `filter_speakers()` method. Otherwise, it includes all speakers. It calculates the number of validation and testing samples based on the provided split ratios (`val_split` and `test_split`). It randomly selects the samples for validation and testing and removes them from the remaining samples. The remaining samples are used for training. The samples are returned as a dictionary with keys 'train', 'validation', and 'test', where each key maps to a dictionary with 'files' (a list of sample paths) and 'labels' (a dictionary mapping sample paths to label IDs).
- `Speaker`: Represents a speaker with attributes:
  - `id: str` → Alphabetical identifier of the speaker.
  - `samples: Dict[str, Label]` → Speaker samples dictionary containing files, path as keys and its associated `Label` object as values.
  - `labels: Set[Label]` → Set of `Label` objects. Contains the `Labels` for which speaker has at least one sample.

- `TrainSpeaker`: Represents a training speaker with additional attributes `val_split`, `val_samples`, and `val_labels`. Inherits from `Speaker`.
  - `id: str` → Alphabetical identifier of the train speaker.
  - `samples: Dict[str, Label]` → Speaker train samples dictionary containing containing files path as keys and its associated `Label` object as values.
  - `labels: Set[Label]` → Set of `Label` objects. Contains the `Labels` for which train speaker has at least one train sample.
  - `val_split: float` → Specify the percentage of samples allocated in the validation set.
  - `val_samples: Dict[str, Label]` → Speaker validation samples dictionary containing containing files path as keys and its associated `Label` object as values.
  - `val_labels: float` → Set of `Label` objects. Contains the `Labels` for which train speaker has at least one validation sample.
- `Dataset`: Represents a dataset consisting of a set of training and testing speakers (and its associated samples and labels). It is used in only in simulated scenarios. It provides functionality to manage and manipulate the dataset.
  - Attributes:
    - `samples: Dict[str, Dict[str, Label]]` → A dictionary that stores the samples in the dataset. The samples are grouped into subsets, including `train`, `validation`, and `test`. Each subset is represented by a dictionary where the keys are sample filenames, and the values are Label objects indicating the corresponding label or command associated with the sample.
    - `speakers: Dict[str, Dict[str, Speaker]]` → A dictionary that stores the speakers in the dataset. Similar to the samples, the speakers are also grouped into subsets (`train` and `test`). Each subset is represented by a dictionary where the keys are speaker IDs, and the values are Speaker objects containing information about the speaker and their associated samples.
    - `labels: Set[Label]` → This attribute is a set that stores all the unique Label objects present in the dataset. It provides a convenient way to access and manage the labels associated with the samples.
  - Methods:
    - `get_samples()` → This method retrieves the samples in the dataset along with their corresponding labels. It returns a dictionary containing subsets ('train', 'validation', and 'test'), and for each subset, it provides a dictionary with 'files' and 'labels'. The 'files' key stores a list of sample filenames, and the 'labels' key stores a dictionary where the keys are filenames, and the values are the numeric label IDs. It prepares the samples to be introduced directly into the `SpeechGenerator` object, as we will see in Section 3.1.3.
    - `add_speaker(sp)` → This method adds a Speaker object to the dataset. It takes a Speaker object as input and appropriately updates the speakers and samples attributes. If the speaker is a TrainSpeaker, it adds the

samples to the 'train' subset as well as the 'validation' subset. If the speaker is a regular Speaker, it adds the samples to the 'test' subset.

* `get_samples(sp_id)` → This method removes a speaker from the dataset based on the speaker ID. It removes the corresponding speaker from the 'train' or 'test' subset and updates the samples and speakers attributes accordingly. It also regenerates the dataset labels to ensure consistency.
* `get_label_distribution()` → This method calculates the distribution of labels across the samples in the dataset. It returns a dictionary where the keys are the numeric label IDs, and the values are the counts of samples associated with each label.
* `remove_label(label_id)` → This method removes a label from the dataset based on the label ID. It removes the corresponding samples from the 'train', 'validation', or 'test' subsets and updates the samples and labels attributes accordingly.
* `remove_labels(labels_to_remove)` → This method removes multiple labels from the dataset based on a list of label IDs. It iterates over the provided label IDs, removes the corresponding samples and labels, and updates the dataset accordingly.
* `restore_labels()` → This method restores the original labels of the dataset by regenerating them from the speaker data. It rebuilds the labels attribute based on the updated samples and speakers attributes.
* `split_dataset(train_sp_per_ds, test_sp_per_ds, shared_speakers)` → This method splits the dataset into multiple datasets based on the specified number of training and testing speakers per dataset. It also provides an option to include shared speakers between datasets. The method returns a list of splitted datasets, each represented by a new Dataset object.

The Figure 3.2 shows the relationship between all dataclasses in `DatasetClasses.py`.

Another extra functions are created inside `DatasetClasses.py`. They provide various features for handling datasets and generating data splits for federated learning. Here is a summary of the API for each function:

- `init_dataset(filter_speakers=True, test_sp_split=0.2)`: Initializes a `Dataset` object with data. It takes parameters such as `filter_speakers` to determine if speakers are filtered, and `test_sp_split` to determine the ratio of test speakers in the dataset. Returns the initialized `Dataset` object.

- `test_num_samples(sp_per_client=None, cmd_to_remove=None)`: Counts the total number of samples considering the number of speakers per client and the commands to remove. It takes parameters such as `sp_per_client` tuple to specify the number of `(train, test, and shared speakers)` per client, and `cmd_to_remove` to specify the commands to remove from the dataset. Returns a dictionary with the number of samples per subset and a set of labels.

- `generate_ds_for_each_label(ds_list: List[Dataset], shuffle=False)`: Distributes a single `Dataset` object per command, maximizing the total number of samples.

It takes a list of splitted `Dataset` objects and an optional `shuffle` parameter to introduce randomness to the data preprocessing. Returns a dictionary containing the labels numerical IDs as keys and the splitted `Dataset` objects as values.

- `set_label_distribution(ds_list: List[Dataset], num_commands: int)`: Sets the label distribution for a list of `Dataset` objects. It takes the number of commands and adjusts the label distribution for each `Dataset` object in the list.
- `store_fl_dataset(fname=DATASET_EXPORT_FILE, num_clients=4, export=True)`: Stores the federated learning dataset in a file. It takes parameters such as the filename (`fname`), the number of clients (`num_clients`), and whether to export the dataset (`export`). Returns a dictionary representing the data split.
- `load_client_fl_dataset(node_id: str, fname=DATASET_EXPORT_FILE)`: Loads the federated learning dataset for a specific client from a file. It takes the client's node ID (`node_id`) and the filename (`fname`). Returns the dataset split for the client.

3.1.2 `DatasetPreprocessing.py`

The module `DatasetPreprocessing.py` gets the dataclasses defined at `DatasetClasses.py` to implement some additional features. The following functions are implemented:

- `download_dataset(forceDownload=False)`: Downloads the Google Speech Commands dataset (v0.01 or v0.02, depending on `DATASET_VERSION` value) and extracts its contents to the specified `DATASET_PATH`.
- `get_spectrogram(waveform)`: Converts waveforms to mel-spectrograms using the Short-Time Fourier Transform (STFT). It takes a waveform as input and returns the corresponding spectrogram representation.
- `store_spectrogram(label_path, audio_file)`: Stores the spectrogram of an audio file as a NumPy array. It takes the path to the directory where the spectrogram file will be stored and the name of the audio file as inputs.
- `generate_spectrograms(multicore=True)`: Obtains WAV files from the dataset and generates spectrograms for each file. The generated spectrograms are saved as NumPy arrays.
- `create_dataset_files(multicore=True)`: Downloads the dataset, generates spectrograms, stores it in a NumPy array and creates necessary files. It also creates a 'PROCESSED.txt' file to indicate that the spectrograms have been obtained and saved. It uses multiprocessing by default, but in some devices this feature may not work. Setting `multicore` to False disable multiprocessing and resolve these issues.
- `get_input_shape() -> Tuple[int]`: Returns the shape of the input spectrogram for the model.
- `get_labels() -> Set[Label]`: Obtains the list of Label objects (representing the labels of the dataset) for the current DatasetMode.

- `get_model(model=MODEL)`: Returns a spectrogram recognition model for image classification tasks. The specific model returned depends on the `model` parameter, which can be either 'cnn' (a simple convolutional neural network) or 'mn2' (MobileNetV2 model). By default load `MODEL` from `Settings.py` file.

3.1.3 `SpeechGenerator.py`

Once the data has been processed, we need to fed the model with it through a data pipeline. That is the main objective of the `SpeechGenerator`, load dynamically the data from memory. The `SpeechGenerator` class is a custom generator class that inherits from `keras.utils.Sequence`. It is designed to load batches of data on the fly for training or evaluation purposes in a speech recognition task. In our solution, the mel-spectrograms are stored in memory as numpy array files (`.npy`) and `SpeechGenerator` loads them in batches. The Code Snippet 18 shows the entire `SpeechGenerator` class.

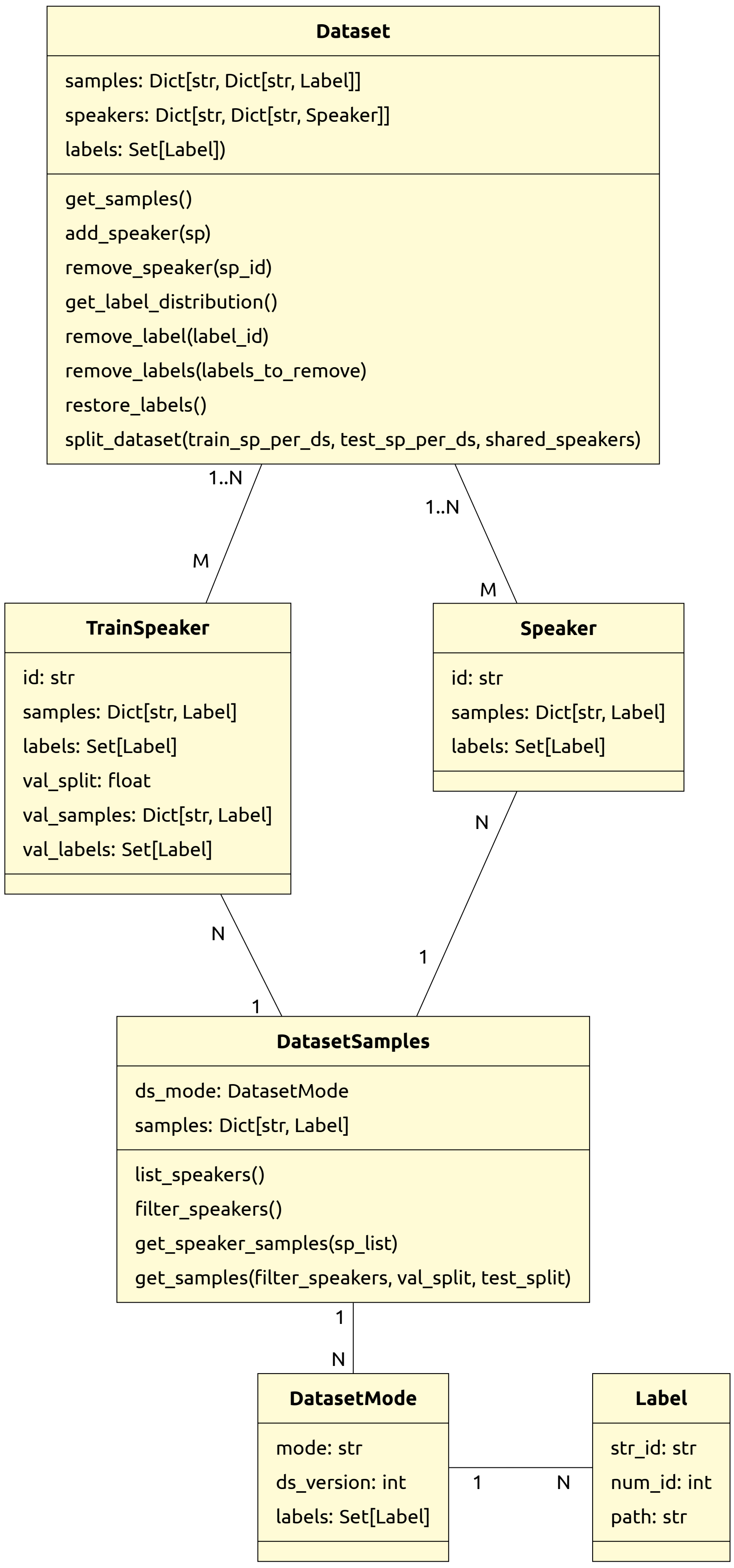


Figure 3.2. `Datasetclasses.py` *dataclasses relationship*

3.2 Flower Federation Implementation

Before exposing the experiments, it is necessary to explain briefly how *Flower* framework architecture works.

FL can be described as an interplay between global and local computations. Global computations are executed on the server-side and are responsible for orchestrating the learning procedure over connected clients. Local computations are executed on individual clients and have access to training data [3]. The Flower architecture (Fig. 3.3) reflects that perspective.

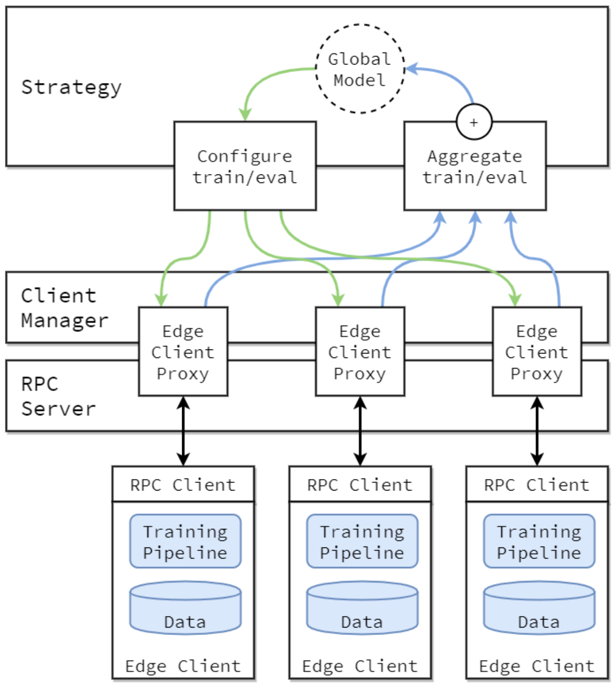


Figure 3.3. Flower architecture. Source Flower Architecture

As in all FL setups, client-side and server-side need to have the same model graph. However, the training stage is performed locally in each client with its local data. The server does not store any type of data and aggregates the model updates from clients.

Global logic for client selection, configuration, parameter update aggregation, and federated or centralized model evaluation is represented through the *Strategy* abstraction that has been defined by Flower. A *Strategy* instance implements a single FL algorithm and decides the behaviour of the server in Flower. The framework provides a variety of already implemented Strategies, which rely on popular FL aggregation algorithms, such as Federated Averaging. Every already implemented *Strategy* contains its own set of predefined configuration options. In the experiments described above, `server.strategy.FedAvg` *Strategy* instance will be used on the server-side. Its relevant configuration parameters are:

- `min_available_clients`: The miniumum number of total clients in the system. If the connected clients count is under this value, the server will not start the FL loop and will remain in an idle state, waiting for new connections. By default, it is set to 2.

- `min_fit_clients`: Minimum number of clients used during training. If the connected clients count is under the specified value, the training/aggregation stage will not start.

- `min_evaluate_clients`: Minimum number of clients used during validation. Only for simulation purposes. In a real scenario, the server cannot evaluate the global model because it does not contain any data. If the required number of connections is reached, the server will call to `evaluate_fn` at the end of every round.

- `fration_fit`: The fraction of clients used during training. In case `min_fit_clients` is larger than `fraction_fit` ∗ `available_clients`, `min_fit_clients` will still be sampled.

- `fration_evaluate`: The fraction of clients used during evaluation. In case `min_fit-_evaluate` is larger than `fraction_evaluate` ∗ `available_clients`, `min_evaluat-e_clients` will still be sampled.

- `evaluate_fn`: Optional function callback used for validation. Only for simulation purposes.

- `on_fit_config_fn`: Function callback used for configure training at the next federation round.

- `initial_parameters`: Initial global model parameters. At the beginning of the federation, this model's internal parameters are passed to clients

The FL loop asks the *Strategy* to configure the next round of the federation, send those configurations to the affected clients, receive the resulting in updates (or failures) from clients and delegates result aggregation to the *Strategy*, which implements a specific FL aggregation algorithm [3]. So, the system is orchestrated by the FL training loop, also called *round*, and coordinated by *Strategy* defined at the server-side.

Local logic is mainly concerned with model training and evaluation on local data partitions. There are a wide variety of existing ML pipelines, so Flower offers an ML framework agnostic way to federate these. In this project, TensorFlow is used as an ML pipeline. The FL clients in Flower are implemented via `flwr.client.Client` or `flwr.client.NumPyClient` base classes. So we create clients by implementing subclasses of them. They contain a set of default methods that need to be implemented to run the federation with Flower. For `flwr.client.NumPyClient`, these are:

- `get_parameters`: Return the current local model parameters.

- `fit`: Receive model parameters from the server, train the model parameters on the local data and return the updated model parameters to the server.

- `evaluate`: Receive model parameters from the server, evaluate the model parameters on the local data, and return the evaluation result to the server.

The communication between server and clients is based on *Remote Procedure Call* software protocol and it is implemented with the gRPC Python framework. The Client Manager handles client connections and generates a proxy for each one to communicate with.

3.3 Federated strategy

For strategies like *FedAvg* or *FedProx*, as it is shown in Figure 3.4 the server follows a specific sequence of methods. First, it awaits the results or possible failures from the clients. Once received, it proceeds with aggregating the results. Next, the server returns the aggregated model parameters and the instructions for configuring the next round of training to the clients. By default, the server typically returns fixed training parameters for the next round, such as the number of local epochs and the batch size. Additionally, it may include an adaptive learning rate based on the current server round.

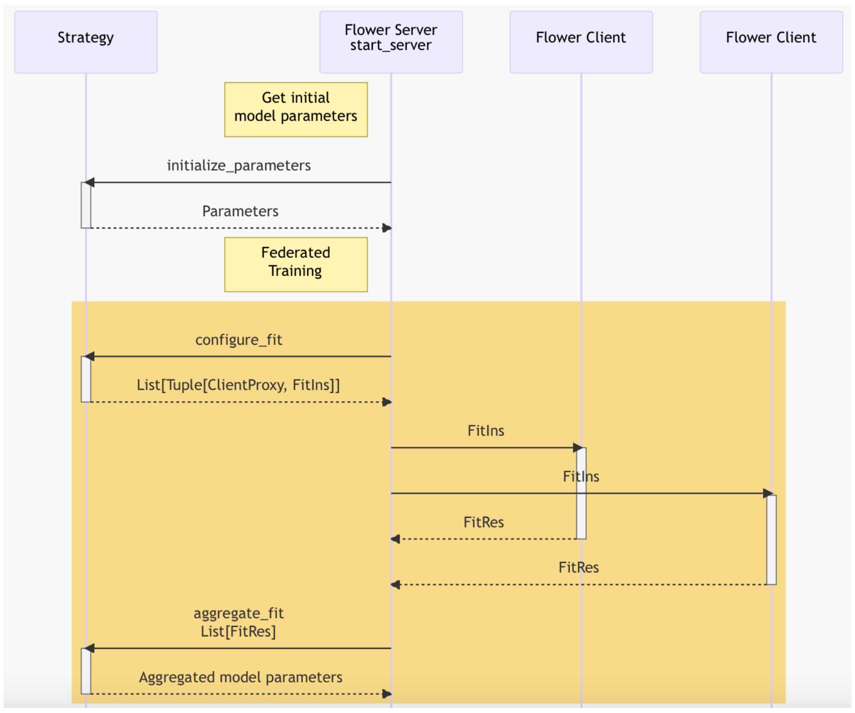

Strategy
Flower Server start_server
Flower Client
Flower Client
Get initial model parameters
initialize_parameters
Parameters
Federated Training
configure_fit
List[Tuple[ClientProxy, FitIns]]
FitIns
FitIns
FitRes
FitRes
aggregate_fit List[FitRes]
Aggregated model parameters

Figure 3.4. Server strategy calls. Source: Flower

However, with federated learning, new solutions can be implemented. In a federated learning scenario, all clients may have different computational capabilities and communication link quality. Additionally, Flower provides an API as specified in Code Snippet 19, and it is evident that the design of the API allows to parameterize the training configuration for each client. Hence, it is interesting to develop a custom and more advanced strategy to take advantage of the the client's capabilities, such as the having a GPU, the number of CPU cores, and other factors, during the federated training.

3.3.1 The FLAIR manager

As shown in Figure 3.5, the FLAIR layer implements all the logic on top of the Flower API, redefining the configuration method for the next round.

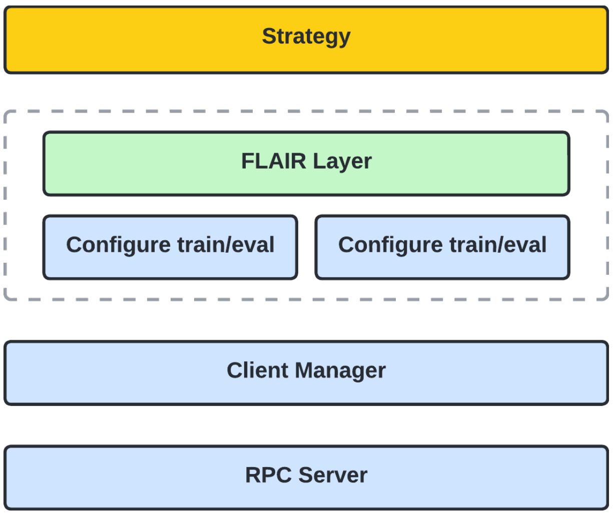


Figure 3.5. FLAIR layer

In this project, is interesting to benchmark FL strategies. It is particularly interesting to manage the FL training process and log/register information, including client properties and different kinds of metrics. To achive this, a FLAIR manager and client have been implemented, as illustrated in Figure 3.6. The FLAIR manager is responsible for managing the FLAIR clients, having a historical data, such as the fit time of each client and other information. With such information and history, complex strategies can be implemented.

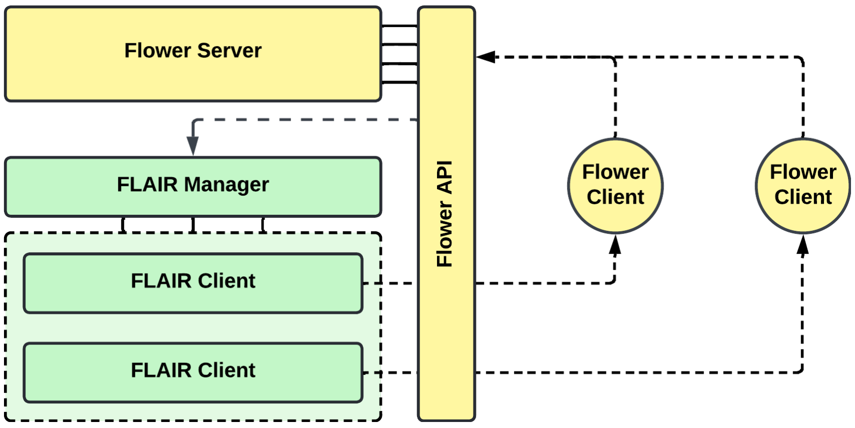


Figure 3.6. FLAIR strategy architecture

In Code Snippet 21, an example is presented where the FLAIR manager is integrated with the default Flower method strategy. When the clients complete a FL round and return their results, the FLAIR manager logs the information and maintains a history of the training stage. This historical data can be accessed at any point during the training process enabling the possibility to develop a more advanced analysis and strategy.

3.3.2 The FLAIR strategy

As mentioned earlier, federated learning offers the opportunity to implement new solutions. In a federated learning scenario, clients often have different computational capabilities and communication link quality. Considering this, the FLAIR team has proposed a new strategy to address these scenarios.

During a training round, the selected clients may have significantly different computational capabilities. This can result in some clients, with powerful resources such as GPU acceleration, finishing their tasks much earlier than other clients using IoT devices with limited capabilities. Consequently, while clients with poorer capabilities are still training, those with powerful resources may have already completed their assigned work.

To tackle this challenge, the FLAIR team aims to make the fit time for each client close to each other, ensuring an earlier converging model. This can be achieved by a variable configuration training process, where more or less epochs are performed in each round.

A possible strategy could be implemented with the following pseudocode:

```
slowest_client, slowest_fit_time = slowest_client()

for client, client_fit_time in clients:
  if client != slowest_client:
    ratio = slowest_fit_time / client_fit_time

    if server_round <= 2:
      num_local_epochs = FIXED_LOCAL_EPOCHS * round(ratio)
      set_client_next_epochs(client, num_local_epochs)

    else:
      previous_local_epochs // From a history (server_round - 1)
      num_local_epochs = previous_local_epochs * round(ratio)
      set_client_next_epochs(client, num_local_epochs)
```

3.4 5G implementation

For the project, we will need to configure and run a 5G end-to-end setup using SDRs and Openairinterface5G, an Open Source software. For this reason, we will need to configure:

- *OAI CN5G*. The core network of the 5G Network that we will be setting up. Hence, a database will be required to be able to register SIM's as a normally done in cellular networks for controlling data, traffic, and so on.
- *OAI gNB*. The 3GPP 5G Next Generation base station which supports the *5G NR* (*5G New Radio*). Therefore, we will setup the base station and configure it to be able to integrate the SDR, specifically, an URSPB200-mini.
- *Quectel RM500Q-GL + Quectel 5G-M2 EVB*. A 5G module will be required to test the 5G Network Setup, being able to test the association between the 5G module (with a SIM) and the gNB (base station). This module and evaluation board will be used as UE (user equipment).

3.4.1 OAI CN5G

3.4.1.1 Setup

Note: It is crucial to work within a native python3 environment. For example, testing OAI within an Anaconda environment the setup could not be completed due to multiple environment dependencies errors.

First of all, we will need to install the necessary tools and dependencies as shown in the Code Snippet 1.

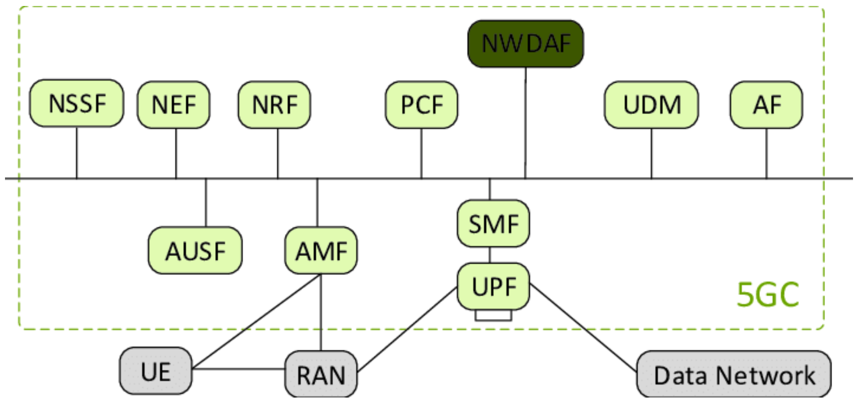


Figure 3.7. 3GPP 5G System Architecture. Source: ResearchGate

```
sudo apt install -y git net-tools putty

sudo apt install -y apt-transport-https ca-certificates curl
↪  software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add
↪  -
sudo add-apt-repository "deb [arch=amd64]
↪  https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
sudo apt update
sudo apt install -y docker docker-ce

# Add your username to the docker group, otherwise you will have to run in
↪  sudo mode.
sudo usermod -a -G docker $(whoami)
reboot

# https://docs.docker.com/compose/install/
sudo curl -L
↪  "https://github.com/docker/compose/releases/download/v2.12.2/docker-
↪  compose-$(uname -s)-$(uname -m)" -o
↪  /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
```

Code Snippet 1. OAI CN5G

Then, we need to clone the `oai-cn5g-fed` repository and pull and tag all the docker images (2). All these containers represent the components/modules defined at the 5G architecture (Fig. 3.7).

```
# Git oai-cn5g-fed repository
git clone https://gitlab.eurecom.fr/oai/cn5g/oai-cn5g-fed.git
↪  ~/oai-cn5g-fed

# Pull docker images
docker pull oaisoftwarealliance/oai-amf:develop
docker pull oaisoftwarealliance/oai-nrf:develop
docker pull oaisoftwarealliance/oai-smf:develop
docker pull oaisoftwarealliance/oai-udr:develop
docker pull oaisoftwarealliance/oai-udm:develop
docker pull oaisoftwarealliance/oai-ausf:develop
docker pull oaisoftwarealliance/oai-spgwu-tiny:develop
docker pull oaisoftwarealliance/trf-gen-cn5g:latest

# Tag docker images
docker image tag oaisoftwarealliance/oai-amf:develop oai-amf:develop
docker image tag oaisoftwarealliance/oai-nrf:develop oai-nrf:develop
docker image tag oaisoftwarealliance/oai-smf:develop oai-smf:develop
docker image tag oaisoftwarealliance/oai-udr:develop oai-udr:develop
docker image tag oaisoftwarealliance/oai-udm:develop oai-udm:develop
docker image tag oaisoftwarealliance/oai-ausf:develop oai-ausf:develop
docker image tag oaisoftwarealliance/oai-spgwu-tiny:develop
↪  oai-spgwu-tiny:develop
docker image tag oaisoftwarealliance/trf-gen-cn5g:latest
↪  trf-gen-cn5g:latest
```

Code Snippet 2. `oai-cn5g-fed` repository setup

Lastly, we copy a pair of configuration files (the main configuration file and the database) to their respective paths.

- Copy `docker-compose-basic-nfr.yaml` to `/oai-cn5g-fed/docker-compose`

  ```
  wget -O ~/oai-cn5g-fed/docker-compose/docker-compose-basic-nrf.yaml
  https://gitlab.eurecom.fr/oai/openairinterface5g/-
   ↪  /raw/develop/doc/tutorial_resources/docker-compose-basic-
   ↪  nrf.yaml?inline=false
  ```

- Copy `oai_db.sql` to `/oai-cn5g-fed/docker-compose/database`

  ```
      wget -O ~/oai-cn5g-fed/docker-compose/database/oai_db.sql
      https://gitlab.eurecom.fr/oai/openairinterface5g/-
   ↪  /raw/develop/doc/tutorial_resources/oai_db.sql?inline=false
  ```

Note: In the event of having a SIM Card Programmer, we can modify its internal parameters using the application uicc-v2.6 from Open Cells Project (Code Snippet 3). Otherwise, we will need to have an already programmed SIM and modify the CN database values to fit into SIM ones.

```
sudo ./program_uicc --adm 12345678 --imsi 001010000000001 --isdn
00000001 --acc 0001 --key fec86ba6eb707ed08905757b1bb44b8f --opc
C42449363BBAD02B66D16BC975D77CC1 -spn "OpenAirInterface"
--authenticate
```

Code Snippet 3. SIM Programming with uicc-v2.6

3.4.1.2 Configuration

Once having OAI CN5G installed with the default setup, we will need to change the default configuration with our specific configuration. Briefly, OpenAirInterface5G has virtualized all the components/modules of the 5G system architecture (Fig. 3.7), in specific, every component has been implemented in its own docker container. Hence, its is important not only to configure the 5G network (MCC, MNC, ...) but we also need to check the connection between the docker containers, each with its own configuration file.

From now on, we will consider that we have an already programmed SIM and we need to adapt the CN5G to it.

First, we will modify the main configuration file (`docker-compose-basic-nfr.yaml`) to:

- Change MCC and MNC values according to the programmed SIM card. In our case, `<MCC_SIM>` and `<MNC_SIM>` are 208 and 92, respectively.

  From the `oai-amf` and `oai-spgwu` services, we will need to set properly the following fields:

  - `oai-amf`:
    * `MCC = <MCC_SIM>`
    * `MNC = <MNC_SIM>`
    * `SERVED_GUAMI_MCC_0 = <MCC_SIM>`
    * `SERVED_GUAMI_MNC_0 = <MNC_SIM>`
    * `PLMN_SUPPORT_MCC = <MCC_SIM>`
    * `PLMN_SUPPORT_MNC = <MNC_SIM>`
  - `oai-spgwu`:
    * `MCC = <MCC_SIM>`
    * `MNC = <MNC_SIM>`

- For the authentication procedure to suceed we need to add the information of our SIM card to the database. We can follow two strategies:

  1. Before starting the CN5G and creating the docker containers, we can modify the default configuration file (/oai-cn5g-fed/docker-compose/oai_db.sql):
  2. If the docker containers have already been created, we can access the `mysql` container and update or insert new UE subscriptions to the database. The info can be inserted using the mysql command line client:

```
    sudo mysql -h 192.168.70.131 -u test -p
    Enter password: test

    mysql> USE oai_db;

    mysql> INSERT INTO `AuthenticationSubscription` (`ueid`,
`authenticationMethod`, `encPermanentKey`, `protectionParameterId`,
`sequenceNumber`, `authenticationManagementField`, `algorithmId`,
`encOpcKey`, `encTopcKey`, `vectorGenerationInHss`, `n5gcAuthMethod`,
`rgAuthenticationInd`, `supi`) VALUES (...);

    mysql> INSERT INTO `SessionManagementSubscriptionData` (`ueid`,
    `servingPlmnid`, `singleNssai`, `dnnConfigurations`) VALUES (...);

    mysql> exit;
```

Code Snippet 4. OAI CN5G `mysql` container

3.4.1.3 Starting the CN5G

Lastly, we can run the CN5G executing the following commands:

```
cd ~/oai-cn5g-fed/docker-compose
python3 core-network.py --type start-basic --scenario 1
```

Code Snippet 5. Running the OAI CN5G

3.4.2 OAI gNB

The gNB (base station) must also be installed and configured. We installed it on the same PC as the core network, to facilitate the communication between them.

3.4.2.1 Setup

There are some prerequisites to set up the base station. First of all, we need to build the UHD drivers (free & open-source software driver and API for the Universal Software Radio Peripheral SDR platform). This can be done using the following commands:

```
sudo apt install -y libboost-all-dev libusb-1.0-0-dev doxygen
↪  python3-docutils python3-mako python3-numpy python3-requests
↪  python3-ruamel.yaml python3-setuptools cmake build-essential

git clone https://github.com/EttusResearch/uhd.git ~/uhd
cd ~/uhd
git checkout v4.3.0.0
cd host
mkdir build
cd build
cmake ../
make -j \$(nproc)
make test # This step is optional
sudo make install
sudo ldconfig
sudo uhd\_images\_downloader
```

Code Snippet 6. Build UHD from source

Once this step has been completed, we can proceed and build the OAI gNB (Code Snippet 7).

3.4.2.2 Configuration

In order to get the base station to work, some changes must be made to the default configuration, so we modified the `gnb.sa.band78.fr1.106PRB.usrpb210.conf` file. This configuration file changes depending on the band and SDR being used. In our case we are using the n78 band and the USRP B200 SDR, and we modified some extra settings:

- Set the proper MCC and MNC, in this case 208 and 92, respectively.
- Check that the gNB and AMF IPs are properly set. In our case, we had to change the gNB IPs shown in the file to $192.168.1.129/24$. The AMF IP was already set correctly by default ($192.168.1.132/24$, the IP of the AMF docker container).

```
# Get openairinterface5g source code
git clone https://gitlab.eurecom.fr/oai/openairinterface5g.git
↪  ~/openairinterface5g
cd ~/openairinterface5g
git checkout develop

# Install OAI dependencies
cd ~/openairinterface5g
source oaienv
cd cmake\_targets
./build\_oai -I

# Build OAI gNB
cd ~/openairinterface5g
source oaienv
cd cmake\_targets
./build\_oai -w USRP --ninja --nrUE --gNB --build-lib all -c
```

Code Snippet 7. Build OAI gNB

3.4.2.3 Starting the gNB

The starting procedure depends on the SDR used and the band. In our case, for the USRP B200 and band 78 we used the following commands:

```
cd ~/openairinterface5g
source oaienv
cd cmake_targets/ran_build/build
sudo ./nr-softmodem -O ../../../targets/PROJECTS/GENERIC-NR-
↪  5GC/CONF/gnb.sa.band78.fr1.106PRB.usrpb210.conf --sa -E
↪  --continuous-tx
```

Code Snippet 8. Running the OAI gNB

3.4.3 UE: Quectel RM500Q-GL + EVB

The Quectel RM500Q-GL is a 5G module optimized specially for IoT/eMBB applications. It supports both 5G NSA and SA modes, and it comes with an evaluation board for testing and debugging purposes. The important things that the EVB has are:

- USB-C port to provide power and connection to the serial port. It is used to send AT commands to configure the module. For power it also has a DC barrel jack input. Both must be connected, as the USB does not provide enough power to the evaluation board.
- Ethernet expansion card. It allows us to use the board as a modem through an Ethernet cable.
- Coaxial connectors and cables for up to 6 antennas. In this case, the module only uses four of them. The antenna pins (Fig. 3.8) must be connected to the antennas using the cables provided by the evaluation board cables. Each antenna function can be seen in Fig. 3.9.

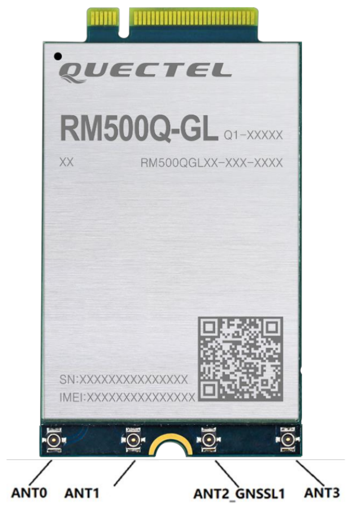
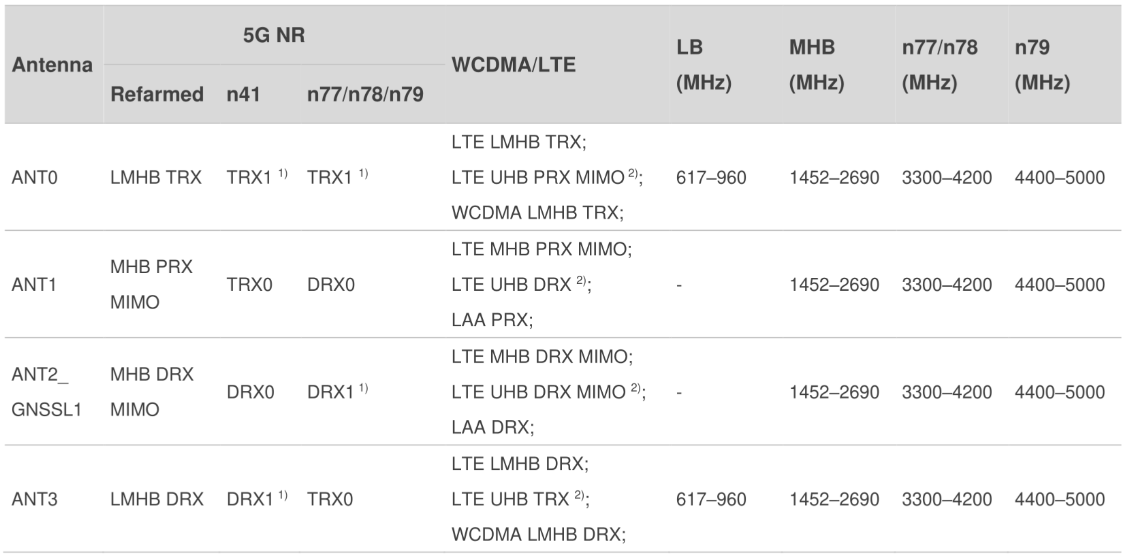


Figure 3.8. RM500Q-GL antenna pins.

3.4.3.1 Pre-configuration

If it is meant to be used in Windows, we need all the drivers that are available to download in the Quectel official web.

Although Windows is recommended by OpenAirInterface5G, it is also possible to setup the module with Ubuntu, and it doesn't even need special drivers to work with.



Antenna	5G NR			WCDMA/LTE	LB (MHz)	MHB (MHz)	n77/n78 (MHz)	n79 (MHz)
	Refarmed	n41	n77/n78/n79					
ANT0	LMHB TRX	TRX1 [1]	TRX1 [1]	LTE LMHB TRX; LTE UHB PRX MIMO [2]; WCDMA LMHB TRX;	617–960	1452–2690	3300–4200	4400–5000
ANT1	MHB PRX MIMO	TRX0	DRX0	LTE MHB PRX MIMO; LTE UHB DRX [2]; LAA PRX;	-	1452–2690	3300–4200	4400–5000
ANT2_ GNSSL1	MHB DRX MIMO	DRX0	DRX1 [1]	LTE MHB DRX MIMO; LTE UHB DRX MIMO [2]; LAA DRX;	-	1452–2690	3300–4200	4400–5000
ANT3	LMHB DRX	DRX1 [1]	TRX0	LTE LMHB DRX; LTE UHB TRX [2]; WCDMA LMHB DRX;	617–960	1452–2690	3300–4200	4400–5000

Figure 3.9. RM500Q-GL antenna mapping.

3.4.3.2 Setup

All the AT commands have to be sent through the serial port with 115200 baudrate and 1 stop bit. It is recommended to use “Cutecom”, a graphical serial terminal.

```
# MUST be sent at least once everytime there is a firmware upgrade!
AT+QMBNCFG="Select","ROW_Commercial"
AT+QMBNCFG="AutoSel",0
AT+CFUN=1,1
AT+CGDCONT=1,"IP","oai"
AT+CGDCONT=2
AT+CGDCONT=3

# (Optional, debug only, AT commands) Activate PDP context, retrieve IP
↪  address and test with ping
AT+CGACT=1,1
AT+CGPADDR=1
AT+QPING=1,"openairinterface.org"
```

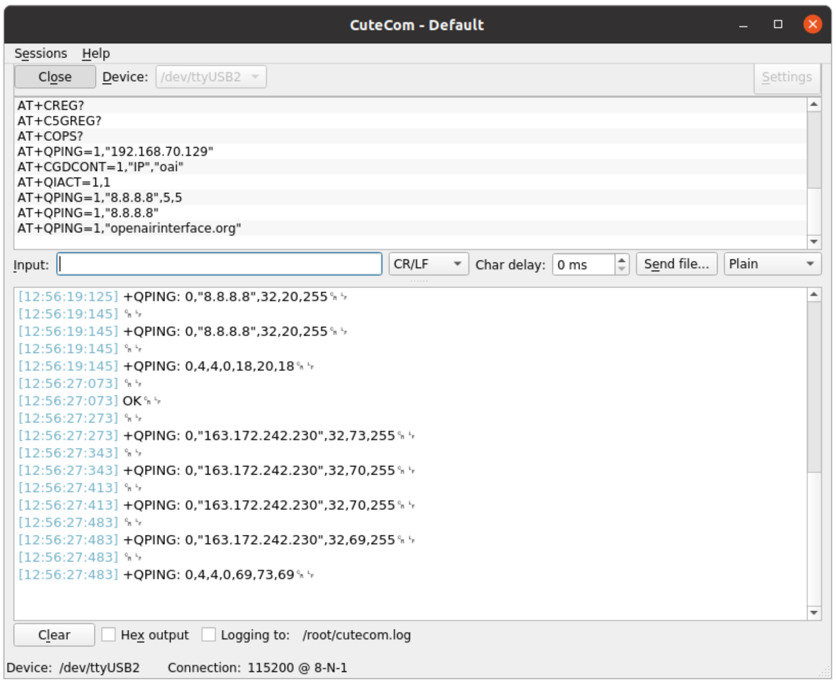
Code Snippet 9. OAI default Quectel setup

First of all, we need to configure which SIM slot will be used. The command to change the slot is “AT+QUIMSLOT=<num>”, in which “<num>” has to be the number of the slot to be used (1 or 2).

Then we need to enable both the SIM card detection and insertion status report with the commands “AT+QSIMDET=1,1” and “AT+QSIMSTAT=1” respectively. This is needed in order to force a reconnection from the UE to the base station by removing and inserting the SIM.

Finally, once enabling and setting up everything, as we can see at Fig. 3.10 and Fig.

3.11, we have been able to successfully setup a 5G network.

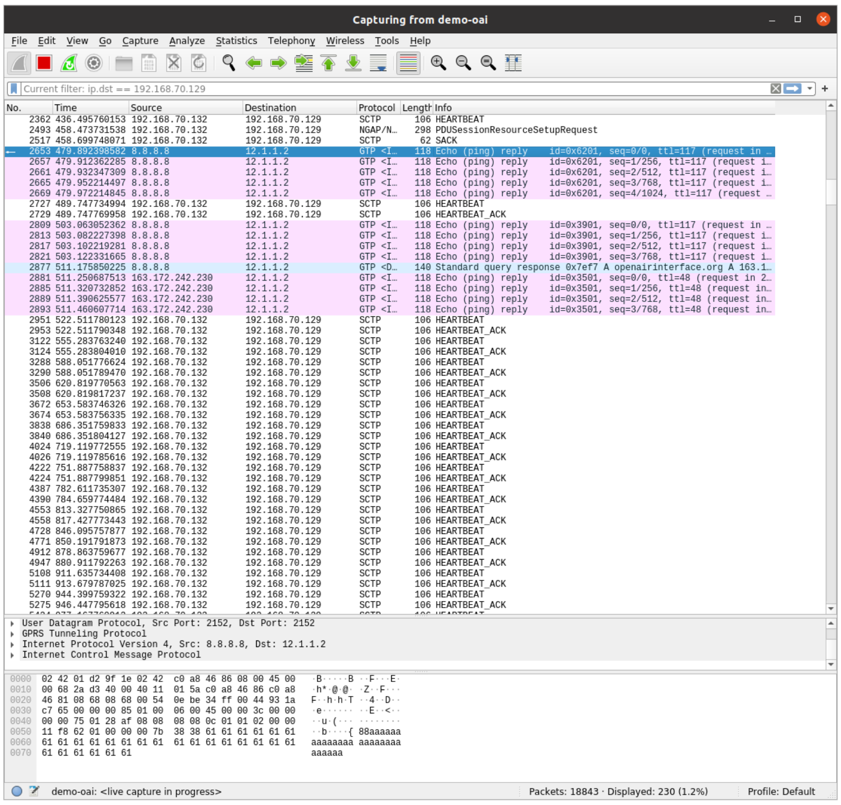


```
AT+CREG?
AT+C5GREG?
AT+COPS?
AT+QPING=1,"192.168.70.129"
AT+CGDCONT=1,"IP","oai"
AT+QIACT=1,1
AT+QPING=1,"8.8.8.8",5,5
AT+QPING=1,"8.8.8.8"
AT+QPING=1,"openairinterface.org"
```

```
[12:56:19:125] +QPING: 0,"8.8.8.8",32,20,255
[12:56:19:145]
[12:56:19:145] +QPING: 0,"8.8.8.8",32,20,255
[12:56:19:145]
[12:56:19:145] +QPING: 0,4,4,0,18,20,18
[12:56:27:073]
[12:56:27:073] OK
[12:56:27:273]
[12:56:27:273] +QPING: 0,"163.172.242.230",32,73,255
[12:56:27:343]
[12:56:27:343] +QPING: 0,"163.172.242.230",32,70,255
[12:56:27:413]
[12:56:27:413] +QPING: 0,"163.172.242.230",32,70,255
[12:56:27:483]
[12:56:27:483] +QPING: 0,"163.172.242.230",32,69,255
[12:56:27:483]
[12:56:27:483] +QPING: 0,4,4,0,69,73,69
```

Device: /dev/ttyUSB2 Connection: 115200 @ 8-N-1

Figure 3.10. Multiple pings from the Quectel to the gNB


Capturing from demo-oai
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Current filter: ip.dst == 192.168.70.129

```
No.   Time           Source           Destination     Protocol  Length Info
 2362 436.495760153 192.168.70.132    192.168.70.129  SCTP      106 HEARTBEAT
 2493 458.473731538 192.168.70.132    192.168.70.129  NGAP/N…   298 PDUSessionResourceSetupRequest
 2517 458.699748071 192.168.70.132    192.168.70.129  SCTP       62 SACK
 2653 479.892398582 8.8.8.8           12.1.1.2        GTP <I…   118 Echo (ping) reply    id=0x6201, seq=0/0, ttl=117 (request in …
 2657 479.912362285 8.8.8.8           12.1.1.2        GTP <I…   118 Echo (ping) reply    id=0x6201, seq=1/256, ttl=117 (request i…
 2661 479.932347309 8.8.8.8           12.1.1.2        GTP <I…   118 Echo (ping) reply    id=0x6201, seq=2/512, ttl=117 (request i…
 2665 479.952214497 8.8.8.8           12.1.1.2        GTP <I…   118 Echo (ping) reply    id=0x6201, seq=3/768, ttl=117 (request i…
 2669 479.972214845 8.8.8.8           12.1.1.2        GTP <I…   118 Echo (ping) reply    id=0x6201, seq=4/1024, ttl=117 (request …
 2727 489.747734994 192.168.70.132    192.168.70.129  SCTP      106 HEARTBEAT
 2729 489.747769958 192.168.70.132    192.168.70.129  SCTP      106 HEARTBEAT_ACK
 2809 503.063052362 8.8.8.8           12.1.1.2        GTP <I…   118 Echo (ping) reply    id=0x3901, seq=0/0, ttl=117 (request in …
 2813 503.082227398 8.8.8.8           12.1.1.2        GTP <I…   118 Echo (ping) reply    id=0x3901, seq=1/256, ttl=117 (request i…
 2817 503.102219281 8.8.8.8           12.1.1.2        GTP <I…   118 Echo (ping) reply    id=0x3901, seq=2/512, ttl=117 (request i…
 2821 503.122331665 8.8.8.8           12.1.1.2        GTP <I…   118 Echo (ping) reply    id=0x3901, seq=3/768, ttl=117 (request i…
 2877 511.175850225 8.8.8.8           12.1.1.2        GTP <D…   140 Standard query response 0x7ef7 A openairinterface.org A 163.1…
 2881 511.250687513 163.172.242.230   12.1.1.2        GTP <I…   118 Echo (ping) reply    id=0x3501, seq=0/0, ttl=48 (request in 2…
 2885 511.320732852 163.172.242.230   12.1.1.2        GTP <I…   118 Echo (ping) reply    id=0x3501, seq=1/256, ttl=48 (request in…
 2889 511.390625577 163.172.242.230   12.1.1.2        GTP <I…   118 Echo (ping) reply    id=0x3501, seq=2/512, ttl=48 (request in…
 2893 511.460607714 163.172.242.230   12.1.1.2        GTP <I…   118 Echo (ping) reply    id=0x3501, seq=3/768, ttl=48 (request in…
 2951 522.511780123 192.168.70.132    192.168.70.129  SCTP      106 HEARTBEAT
 2953 522.511790348 192.168.70.132    192.168.70.129  SCTP      106 HEARTBEAT_ACK
 3122 555.283763240 192.168.70.132    192.168.70.129  SCTP      106 HEARTBEAT
 3124 555.283804010 192.168.70.132    192.168.70.129  SCTP      106 HEARTBEAT_ACK
 3288 588.051776624 192.168.70.132    192.168.70.129  SCTP      106 HEARTBEAT
 3290 588.051789470 192.168.70.132    192.168.70.129  SCTP      106 HEARTBEAT_ACK
 3506 620.819770563 192.168.70.132    192.168.70.129  SCTP      106 HEARTBEAT
 3508 620.819817237 192.168.70.132    192.168.70.129  SCTP      106 HEARTBEAT_ACK
 3672 653.583746326 192.168.70.132    192.168.70.129  SCTP      106 HEARTBEAT
 3674 653.583756335 192.168.70.132    192.168.70.129  SCTP      106 HEARTBEAT_ACK
 3838 686.351759833 192.168.70.132    192.168.70.129  SCTP      106 HEARTBEAT
 3840 686.351804127 192.168.70.132    192.168.70.129  SCTP      106 HEARTBEAT_ACK
 4024 719.119772555 192.168.70.132    192.168.70.129  SCTP      106 HEARTBEAT
 4026 719.119785616 192.168.70.132    192.168.70.129  SCTP      106 HEARTBEAT_ACK
 4222 751.887758837 192.168.70.132    192.168.70.129  SCTP      106 HEARTBEAT
 4224 751.887799851 192.168.70.132    192.168.70.129  SCTP      106 HEARTBEAT_ACK
 4387 782.611735307 192.168.70.132    192.168.70.129  SCTP      106 HEARTBEAT
 4390 784.659774484 192.168.70.132    192.168.70.129  SCTP      106 HEARTBEAT_ACK
 4553 813.327750865 192.168.70.132    192.168.70.129  SCTP      106 HEARTBEAT
 4558 817.427773443 192.168.70.132    192.168.70.129  SCTP      106 HEARTBEAT_ACK
 4728 846.095757877 192.168.70.132    192.168.70.129  SCTP      106 HEARTBEAT
 4771 850.191791873 192.168.70.132    192.168.70.129  SCTP      106 HEARTBEAT_ACK
 4912 878.863759677 192.168.70.132    192.168.70.129  SCTP      106 HEARTBEAT
 4947 880.911792263 192.168.70.132    192.168.70.129  SCTP      106 HEARTBEAT_ACK
 5108 911.635734408 192.168.70.132    192.168.70.129  SCTP      106 HEARTBEAT
 5111 913.679787025 192.168.70.132    192.168.70.129  SCTP      106 HEARTBEAT_ACK
 5270 944.399759322 192.168.70.132    192.168.70.129  SCTP      106 HEARTBEAT
 5275 946.447795618 192.168.70.132    192.168.70.129  SCTP      106 HEARTBEAT_ACK
```

```
▸ User Datagram Protocol, Src Port: 2152, Dst Port: 2152
▸ GPRS Tunneling Protocol
▸ Internet Protocol Version 4, Src: 8.8.8.8, Dst: 12.1.1.2
▸ Internet Control Message Protocol
```

```
0000  02 42 01 d2 9f 1e 02 42  c0 a8 46 86 08 00 45 00   ·B·····B ··F···E·
0010  00 68 2a d3 40 00 40 11  01 5a c0 a8 46 86 c0 a8   ·h*·@·@· ·Z··F···
0020  46 81 08 68 08 68 00 54  0e be 34 ff 00 44 93 1a   F··h·h·T ··4··D··
0030  c7 65 00 00 00 85 01 00  06 00 45 00 00 3c 00 00   ·e······ ··E··<··
0040  00 00 75 01 28 af 08 08  08 08 0c 01 01 02 00 00   ··u·(··· ········
0050  11 f8 62 01 00 00 00 7b  38 38 61 61 61 61 61 61   ··b····{ 88aaaaaa
0060  61 61 61 61 61 61 61 61  61 61 61 61 61 61 61 61   aaaaaaaa aaaaaaaa
0070  61 61 61 61 61 61                                  aaaaaa
```

demo-oai: <live capture in progress> Packets: 18843 · Displayed: 230 (1.2%) Profile: Default

Figure 3.11. Multiple pings received (gNB)

Note: If the default OAI Quectel setup does not work, for example, an error `561` in a `QPING` AT command, it is most likely to be an error with the activation of the `PDP CONTEXT`. Hence, using `AT+QIACT=1,1` it solved the problem, activating the `PDPCONTEXT`.

The AT+CGACT=1,1 command shown in the previous listing should work exactly the same way.

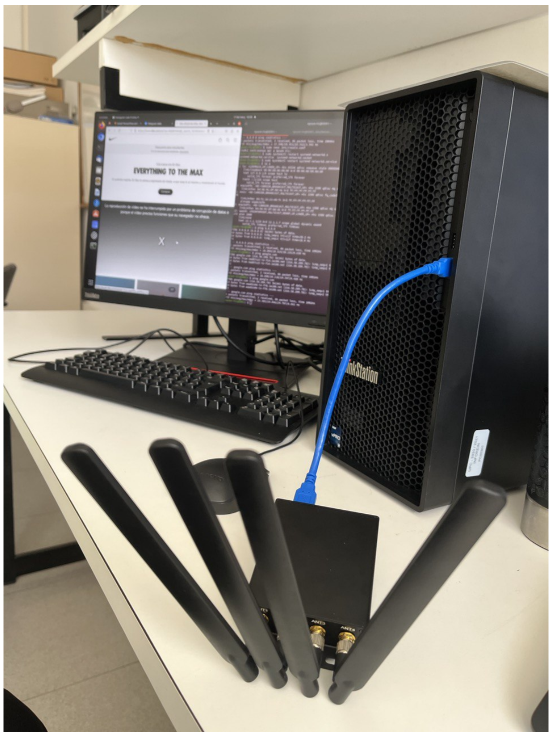
3.4.3.3 NetworkManager setup

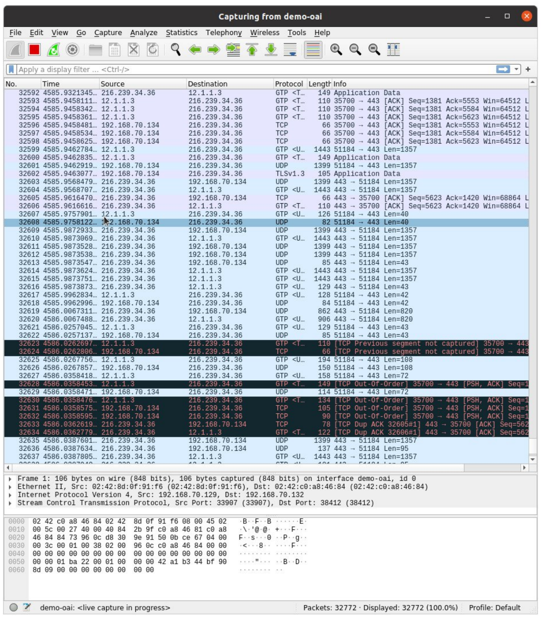
Once having the Quectel succesfully connected to the base station and being able to establish a communication, we will configure the Quectel module as a 5G modem for a host machine. Hence, we will have to configure NetworkManager (nmcli) to add a new *GSM* connection. Briefly, we will create a new configuration file in /etc/NetworkManager/system-connections with the *.nmconnection extension. Specifically, using PPP (Point-to-Point Protocol) for establishing a connection with the Quectel module over the GSM network via serial interface between the modem and th host machine (Code Snippet 10). As a result, we can see at Fig. 3.12, we have successfully established a connection between a host machine via the Quectel modem to the base station.

```
[connection]
id=quectel-oai-ppp
type=gsm
autoconnect=false

[gsm]
apn=oai
device=/dev/ttyUSB0

[ppp]
lcp-echo-failure=5
lcp-echo-interval=30
user=
password=

[ipv4]
method=auto

[ipv6]
addr-gen-mode=stable-privacy
method=auto
```

Code Snippet 10. NetworkManager GSM connection

Note: We encountered issues when connecting the Quectel RM500Q-GL modem to a regular laptop. Despite checking the compatibility of kernel versions and operating systems, the modem was never detected on the laptops for unknown reasons.


(a) Host machine (client)


Capturing from demo-oai

```
No.   Time           Source          Destination     Protocol Length Info
32592 4585.9321345… 216.239.34.36   12.1.1.3        GTP <T…   149 Application Data
32593 4585.9458111… 12.1.1.3        216.239.34.36   GTP <T…   110 35700 → 443 [ACK] Seq=1381 Ack=5553 Win=64512 L
32594 4585.9458342… 12.1.1.3        216.239.34.36   GTP <T…   110 35700 → 443 [ACK] Seq=1381 Ack=5584 Win=64512 L
32595 4585.9458361… 12.1.1.3        216.239.34.36   GTP <T…   110 35700 → 443 [ACK] Seq=1381 Ack=5623 Win=64512 L
32596 4585.9458481… 192.168.70.134  216.239.34.36   TCP        66 35700 → 443 [ACK] Seq=1381 Ack=5553 Win=64512 L
32597 4585.9458534… 192.168.70.134  216.239.34.36   TCP        66 35700 → 443 [ACK] Seq=1381 Ack=5584 Win=64512 L
32598 4585.9458625… 192.168.70.134  216.239.34.36   TCP        66 35700 → 443 [ACK] Seq=1381 Ack=5623 Win=64512 L
32599 4585.9462784… 12.1.1.3        216.239.34.36   GTP <U…  1443 51184 → 443 Len=1357
32600 4585.9462835… 12.1.1.3        216.239.34.36   GTP <T…   149 Application Data
32601 4585.9462919… 192.168.70.134  216.239.34.36   UDP      1399 51184 → 443 Len=1357
32602 4585.9463077… 192.168.70.134  216.239.34.36   TLSv1.3   105 Application Data
32603 4585.9568479… 216.239.34.36   192.168.70.134  UDP      1399 443 → 51184 Len=1357
32604 4585.9568707… 216.239.34.36   12.1.1.3        GTP <U…  1443 443 → 51184 Len=1357
32605 4585.9616470… 216.239.34.36   192.168.70.134  TCP        66 443 → 35700 [ACK] Seq=5623 Ack=1420 Win=68864 L
32606 4585.9616616… 216.239.34.36   12.1.1.3        GTP <T…   110 443 → 35700 [ACK] Seq=5623 Ack=1420 Win=68864 L
32607 4585.9757901… 12.1.1.3        216.239.34.36   GTP <U…   126 51184 → 443 Len=40
32608 4585.9758122… 192.168.70.134  216.239.34.36   UDP        82 51184 → 443 Len=40
32609 4585.9872933… 216.239.34.36   192.168.70.134  UDP      1399 443 → 51184 Len=1357
32610 4585.9873069… 216.239.34.36   12.1.1.3        GTP <U…  1443 443 → 51184 Len=1357
32611 4585.9873528… 216.239.34.36   192.168.70.134  UDP      1399 443 → 51184 Len=1357
32612 4585.9873538… 216.239.34.36   192.168.70.134  UDP      1399 443 → 51184 Len=1357
32613 4585.9873547… 216.239.34.36   192.168.70.134  UDP        85 443 → 51184 Len=43
32614 4585.9873624… 216.239.34.36   12.1.1.3        GTP <U…  1443 443 → 51184 Len=1357
32615 4585.9873751… 216.239.34.36   12.1.1.3        GTP <U…  1443 443 → 51184 Len=1357
32616 4585.9873873… 216.239.34.36   12.1.1.3        GTP <U…   129 443 → 51184 Len=43
32617 4585.9962834… 12.1.1.3        216.239.34.36   GTP <U…   128 51184 → 443 Len=42
32618 4585.9962996… 192.168.70.134  216.239.34.36   UDP        84 51184 → 443 Len=42
32619 4586.0067311… 216.239.34.36   192.168.70.134  UDP       862 443 → 51184 Len=820
32620 4586.0067488… 216.239.34.36   12.1.1.3        GTP <U…   906 443 → 51184 Len=820
32621 4586.0257045… 12.1.1.3        216.239.34.36   GTP <U…   129 51184 → 443 Len=43
32622 4586.0257137… 192.168.70.134  216.239.34.36   UDP        85 51184 → 443 Len=43
32623 4586.0262697… 12.1.1.3        216.239.34.36   GTP <T…   110 [TCP Previous segment not captured] 35700 → 443
32624 4586.0262806… 192.168.70.134  216.239.34.36   TCP        66 [TCP Previous segment not captured] 35700 → 443
32625 4586.0267756… 12.1.1.3        216.239.34.36   GTP <U…   194 51184 → 443 Len=108
32626 4586.0267857… 192.168.70.134  216.239.34.36   UDP       150 51184 → 443 Len=108
32627 4586.0358418… 12.1.1.3        216.239.34.36   GTP <U…   158 51184 → 443 Len=72
32628 4586.0358453… 12.1.1.3        216.239.34.36   GTP <T…   149 [TCP Out-Of-Order] 35700 → 443 [PSH, ACK] Seq=1
32629 4586.0358471… 192.168.70.134  216.239.34.36   UDP       114 51184 → 443 Len=72
32630 4586.0358476… 12.1.1.3        216.239.34.36   GTP <T…   134 [TCP Out-Of-Order] 35700 → 443 [PSH, ACK] Seq=1
32631 4586.0358575… 192.168.70.134  216.239.34.36   TCP       105 [TCP Out-Of-Order] 35700 → 443 [PSH, ACK] Seq=1
32632 4586.0358595… 192.168.70.134  216.239.34.36   TCP        90 [TCP Out-Of-Order] 35700 → 443 [PSH, ACK] Seq=1
32633 4586.0362619… 216.239.34.36   192.168.70.134  TCP        78 [TCP Dup ACK 32605#1] 443 → 35700 [ACK] Seq=562
32634 4586.0362779… 216.239.34.36   12.1.1.3        GTP <T…   122 [TCP Dup ACK 32606#1] 443 → 35700 [ACK] Seq=562
32635 4586.0387601… 216.239.34.36   192.168.70.134  UDP      1399 443 → 51184 Len=1357
32636 4586.0387634… 216.239.34.36   192.168.70.134  UDP       137 443 → 51184 Len=95
32637 4586.0387805… 216.239.34.36   12.1.1.3        GTP <U…  1443 443 → 51184 Len=1357

Frame 1: 106 bytes on wire (848 bits), 106 bytes captured (848 bits) on interface demo-oai, id 0
Ethernet II, Src: 02:42:8d:0f:91:f6 (02:42:8d:0f:91:f6), Dst: 02:42:c0:a8:46:84 (02:42:c0:a8:46:84)
Internet Protocol Version 4, Src: 192.168.70.129, Dst: 192.168.70.132
Stream Control Transmission Protocol, Src Port: 33907 (33907), Dst Port: 38412 (38412)

0000  02 42 c0 a8 46 84 02 42  8d 0f 91 f6 08 00 45 02   ·B··F··B ······E·
0010  00 5c 00 27 40 00 40 84  2b 9f c0 a8 46 81 c0 a8   ·\·'@·@· +···F···
0020  46 84 84 73 96 0c d8 30  9e 91 50 0b ce 67 04 00   F··s···0 ··P··g··
0030  00 3c 00 01 00 38 02 00  96 0c c0 a8 46 84 00 00   ·<···8·· ····F···
0040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0050  00 00 01 ba 22 00 01 00  00 00 42 a1 b3 44 bf 90   ····"··· ··B··D··
0060  8d 09 00 00 00 00 00 00  00 00                     ········ ··

demo-oai: <live capture in progress>          Packets: 32772 · Displayed: 32772 (100.0%)    Profile: Default
```

(b) Wireshark capture (gNB)

Figure 3.12. A connection between a client and the 5G base station (gNB)

4 Hardware architecture

4.1 GPU acceleration

FL is a machine learning approach that allows multiple clients to collaborate in training a global model without sharing their data. One interesting aspect of FL is the involvement of clients with different computational capabilities and resources. This diversity presents both opportunities and challenges for the project.

The participation of clients with varying computational capabilities allows us to explore the potential benefits of FL. Some clients may have high-performance GPUs, which can significantly accelerate training and inference processes. On the other hand, other clients may have limited computing resources, such as mobile devices or edge devices with lower computational power. This diversity presents real-world scenarios where FL can take advantage of this scenario.

In this project, we have selected two GPUs for our federated learning setup:

- NVIDIA Jetson TX2: Is a GPU specifically designed for embedded systems and edge computing. It provides high-performance GPU acceleration in a power-efficient package. However, it is important to note that the Jetson TX2 is considered an older model and is no longer supported by the latest NVIDIA JetPacks and software updates.
- NVIDIA GeForce RTX 3060: Is a more recent GPU model known for its powerful performance in gaming and machine learning applications.

4.1.1 Encountered issues

While testing with GPU acceleration, we faced many dependency issues with the *NVIDIA Jetson TX2*. Firstly, the Jetson platform includes small, power-efficient developer kits and production modules that offer high-performance acceleration of the NVIDIA CUDA-X™ software stack. However, some products such as the *NVIDIA Jetson TX2* have been discontinued being unable to update to the latest NVIDIA JetPacks. As a consequence, dependency issues may start to occur, e.g., in our case, issues between *Python3* and *TensorFlow* versions for experimenting with GPU-based acceleration (Fig. 4.1 for reference) have been observed. Finally, Fig. 4.2 intends to summarize the entire setup, specifically, the requirements and the hardware conditions for a feasible execution of the whole software for the two Nvidia platforms that were considered to be lent during the project.

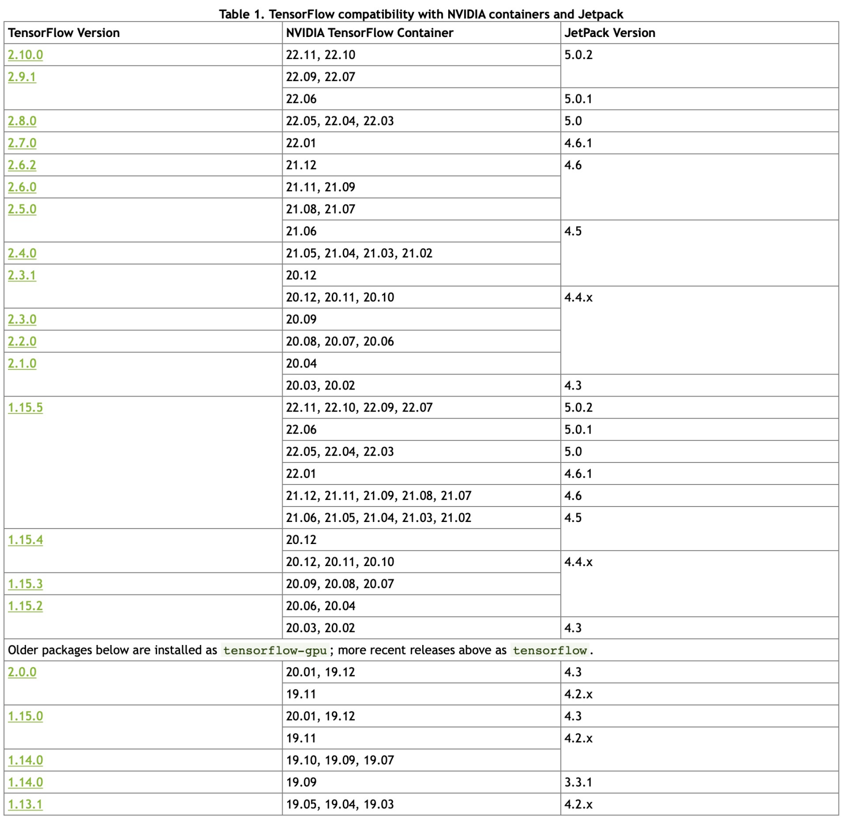


Table 1. TensorFlow compatibility with NVIDIA containers and Jetpack

TensorFlow Version	NVIDIA TensorFlow Container	JetPack Version
2.10.0	22.11, 22.10	5.0.2
2.9.1	22.09, 22.07	
	22.06	5.0.1
2.8.0	22.05, 22.04, 22.03	5.0
2.7.0	22.01	4.6.1
2.6.2	21.12	4.6
2.6.0	21.11, 21.09	
2.5.0	21.08, 21.07	
	21.06	4.5
2.4.0	21.05, 21.04, 21.03, 21.02	
2.3.1	20.12	
	20.12, 20.11, 20.10	4.4.x
2.3.0	20.09	
2.2.0	20.08, 20.07, 20.06	
2.1.0	20.04	
	20.03, 20.02	4.3
1.15.5	22.11, 22.10, 22.09, 22.07	5.0.2
	22.06	5.0.1
	22.05, 22.04, 22.03	5.0
	22.01	4.6.1
	21.12, 21.11, 21.09, 21.08, 21.07	4.6
	21.06, 21.05, 21.04, 21.03, 21.02	4.5
1.15.4	20.12	
	20.12, 20.11, 20.10	4.4.x
1.15.3	20.09, 20.08, 20.07	
1.15.2	20.06, 20.04	
	20.03, 20.02	4.3
Older packages below are installed as `tensorflow-gpu`; more recent releases above as `tensorflow`.		
2.0.0	20.01, 19.12	4.3
	19.11	4.2.x
1.15.0	20.01, 19.12	4.3
	19.11	4.2.x
1.14.0	19.10, 19.09, 19.07	
1.14.0	19.09	3.3.1
1.13.1	19.05, 19.04, 19.03	4.2.x

Figure 4.1. TensorFlow compatibility with NVIDIA containers and Jetpack. Source: TensorFlow For Jetson Platform

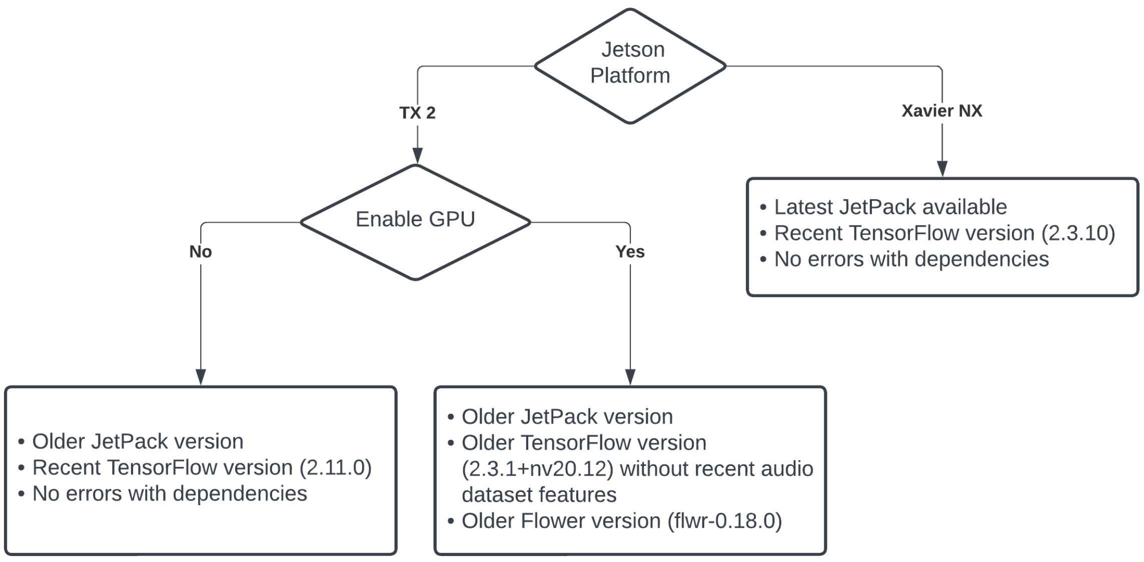

Jetson Platform
TX 2
Xavier NX
Enable GPU
No
Yes
• Latest JetPack available
• Recent TensorFlow version (2.3.10)
• No errors with dependencies
• Older JetPack version
• Recent TensorFlow version (2.11.0)
• No errors with dependencies
• Older JetPack version
• Older TensorFlow version (2.3.1+nv20.12) without recent audio dataset features
• Older Flower version (flwr-0.18.0)

Figure 4.2. Hardware conditions for a feasible execution of the whole software. Source: Own

4.2 5G network

In research studies, having a controlled environment and the ability to manipulate conditions is important. However, when working within a 5G network that operates through radiofrequency, it is evident that interferences can occur, and the behavior of the network may not remain constant. Consequently, the environment becomes practically uncontrolled.

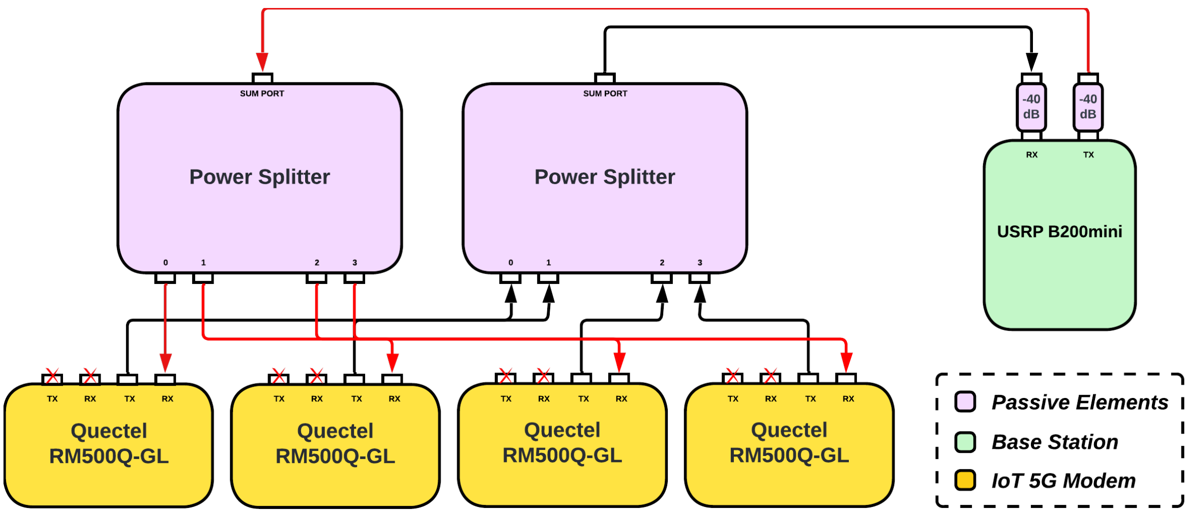


Figure 4.3. Wired 5G network

Hence, the Figure 4.3 represents a wired 5G network with a base station (based on the platform *USRP B200mini*) along with multiple 5G IoT modems (*Quectel RM500Q-GL*) that will establish connections with clients. This particular 5G network, provides us a complete control over the network, also minimizing external interference. This configuration allows us to modify individual client conditions, such as manipulating the modulation of each client by attenuating the received signal, among other possibilities. As a result, in Figure 4.4 and 4.5 we can see the final 5G network setup, with the corresponding signal attenuation and power splitting across all the 5G IoT modems.

Finally, while the setup mentioned earlier provides an environment with absolute control, it also lacks the flexibility and realism of a typical 5G network. In this setup, all clients must be in close proximity to each other, as shown in Figure 4.4. Additionally, the complexity of the setup increases as it requires additional equipment such as power splitters, attenuators, and coaxial cables. These components are necessary to establish the wired connections and manipulate the conditions of each client. The addition of such equipment adds to the overall complexity and infrastructure requirements of the setup.

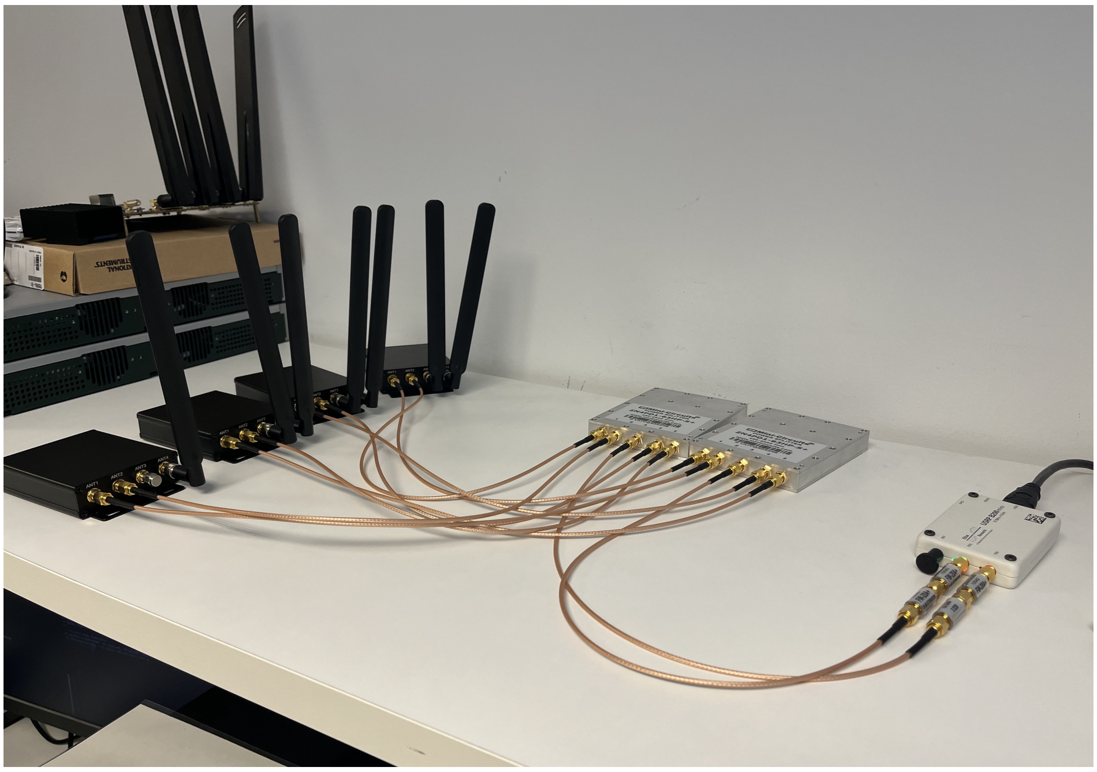


Figure 4.4. Client's 5G modems connected to the basestation

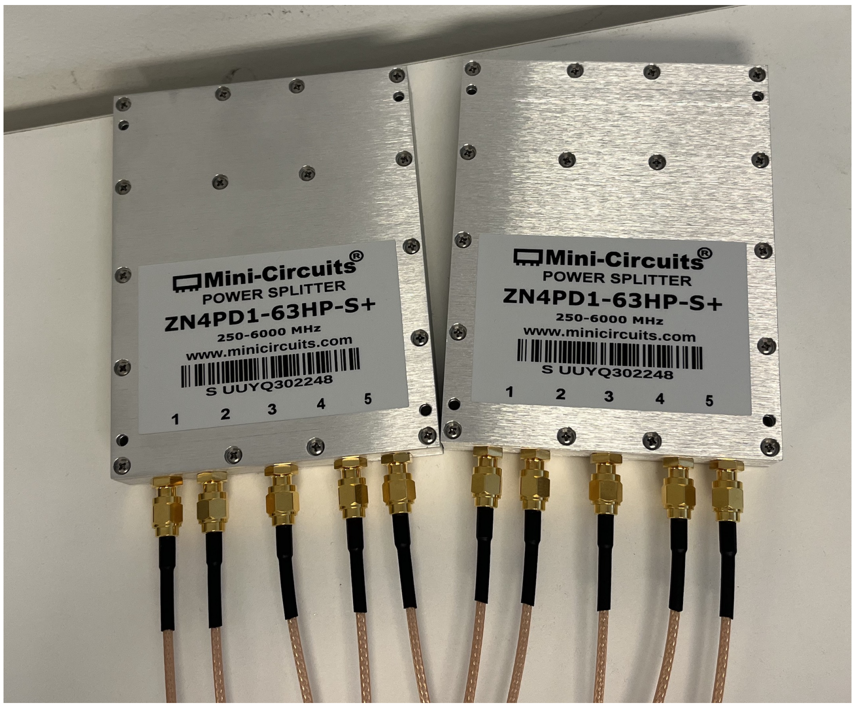


Figure 4.5. Signal management though power splitter

5 FLAIR Solution User Manual

5.1 Initialization

First of all, it is necessary to set up a Python 3 environment and install the required pip dependencies for the project (available in https://github.com/FLAIR-UPC/FLAIR_PROJECT). Once the environment is properly configured, the software defines the following key modules:

- `common.client`: Defines the `SpeechClient` class, which is used in all evaluations for the speech commands dataset.
- `common.server`: Contains the method calls for the Flower server. This module is specifically used in simulation evaluations.
- `common.config`: A file that defines various parameters, including the server's IP address, the number of FL clients, and other default values that can be customized.
- `common.dataset`: Defines the manipulations and preprocessings for the dataset, including handling audio files and generating spectrograms for each audio. It also introduces specific dataset modes to study the homogeneity in Federated Learning.
- `common.flair`: Contains the FLAIR manager and FLAIR client. The manager, as the name indicates, handles the FLAIR clients on the server side. It awaits the results from the Flower server and saves them, enabling the definition and implementation of custom and advanced strategies based on the history of each real FL client.

NOTE: Before executing anything, it is necessary to add the project path into the `PYTHONPATH` environment variable. This can be done by adding the following line to the `.bashrc` file in system home directory or by executing it in every single terminal instance:

```
export PYTHONPATH=~<path_to_project>/FLAIR-Project/src:$PYTHONPATH
```

Code Snippet 11. Dataset initialization and preprocessing

Hence, the first important thing is to download and generate the spectograms of all the dataset. For this reason this can be acompliseh by running directly one of the main programs defined in the `Project/` directory or by executing the following:

5.2 Simulated scenario

To execute a simulated Federated Learning scenario with multiple virtualized clients, the configuration file `common/config/Settings.py` defines the following important parameters:

```
cd $PROJECT_DIR/src/common/dataset
python3 -i DatasetPreprocessing.py
>>> create_dataset_files()
```

Code Snippet 12. Adding the project to the `PYTHONPATH`

- `BATCH_SIZE`: Sets the batch size for simulation training.
- `NUM_FL_CLIENTS`: Specifies the number of FL clients in the simulation.
- `LOCAL_EPOCHS`: Sets the number of local epochs for simulation clients.
- `NUM_ROUNDS`: Specifies the number of simulation rounds.
- `FL_ALGORITHM`: Specifies the FL algorithm to use ('FedAvg' or 'FedProx').
- `TRAIN_SPEAKERS_PER_CLIENT`: Sets the number of train speakers per client.
- `TEST_SPEAKERS_PER_CLIENT`: Sets the number of test speakers per client.
- `SHARED_SPEAKERS_PER_CLIENT`: Sets the number of shared speakers per client.
- `CMDS_PER_CLIENT`: Sets the number of commands per client (it takes effect when `ONE_DS_PER_CMD` is False).
- `ONE_DS_PER_CMD`: Determine if the simluation is executed with only one client (local dataset) per command. With that option enabled, the number of client depend of the chosen dataset mode.
- `NUM_CPUS_CLIENT`: Decides the number of cores allocated to each client in a simulation. It affects to the number of clients that are executing concurrently and depends on the number of cores of the machine that is executing the simulation.

Once these options are set, we can go to the `src/Project/federated_simulated` directory and execute `simulation.py` file.

```
python3 simulation.py
```

Code Snippet 13. Dataset initialization and preprocessing

5.3 Real Network scenario

To execute a realistic Federated Learning (FL) scenario involving multiple clients, the module `common/config/Settings.py` defines the following important parameters:

- `FL_SERVER_ADDR`: Specifies the IP address of the server which the FL clients will connect for FL training.
- `FL_SIMPLE_STRATEGY`: Determines whether to use or not the default simple strategy defined by Flower. If not, it uses the FLAIR strategy.
- `FL_NUM_CLIENTS`: Represents the maximum number of clients that the server will allow to collaborate in the FL scenario.
- `FL_NUM_ROUNDS`: Defines the total number of rounds that will be executed during the FL training process.
- `FL_LOCAL_EPOCHS`: Specifies the number of local epochs that each client will perform in each round.
- `FL_BATCH_SIZE`: Defines the default batch size that each client will have during the FL training.

In this project, the FL scenario has been tested to perform benchmarking studies based on the clients computational capabilities for optimizing the training process taking advantage of the clients with more resources to do «more work».

Whenever defining a new FL scenario, it is necessary to provide an ”experiment” name:

```
cd $PROJECT_DIR/Project/federated/
python3 server.py --exp <experiment name>
```

Code Snippet 14. FLAIR FL server initialization

Then, for the client side, every client should define a `CLIENTS_PROPERTIES.json` file with at least the following keys (also with the possibility to add other kinds of custom keys):

```
{
    "id (str)": "The client id, it does not have to be strictly unique",
    "gpu (bool)": "Whether the client has an accelerator or not",
    "cpu_cores (int)": "The number cores of the client's cpu",
    "ram (int)": "The amount of RAM that the client have installed"
}
```

For starting a FL client it is necessary to provide a client id (cid):

```
cd $PROJECT_DIR/Project/federated/
python3 client.py --cid <client id>
```

Code Snippet 15. FLAIR FL client initialization

When the FL training is done or even if the training stage is forced to stop, the history of the training will be saved in logs/<exp_name> containing a `metrics` and `plots` directories with the following information:

- `clients_properties.json`: It contains a json file with the information that has been logged and registered by the FLAIR manager.
- `clients_metrics.json`: Contains the received clients metrics for each round, information such as the batch size, the fit time, the datetime and the fit time per epoch.
- `server_metrics.json`: Saves the global model loss and accuracy of each round.

Finally to obtain the plots of the training stage is necessary to perform the following action which will generate and save the plots to be evaluated.

```
cd $PROJECT_DIR/Project/federated/
python3 plotting.py --exp <experiment name>
```

Code Snippet 16. FLAIR FL plotting result

5.4 Additional options

In `src/common/Settings.py` exist other additional options that control the device configuration, the dataset configuration…These are the following:

- `DEVICE`: Specifies the device used for computation (e.g., 'CPU' or 'GPU').
- `LIMIT_VRAM`: Determines if VRAM usage should be limited when using GPU. It is useful when executing multiple clients and the server in a real network scenario, but using the same machine (setting server address to the loopback address).
- `VRAM_LIMIT`: Sets the VRAM limit (in MB) when `LIMIT_VRAM` is enabled.
- `AUDIO_LENGTH`: Defines the length of the audio samples in the dataset (in samples).
- `DATASET_PATH`: Specifies the path to the dataset directory.
- `DATASET_VERSION`: Specifies the version of the dataset.
- `DATASET_EXPORT_FILE`: Defines the path where is exported the dataset, when it is stored as a file. Used in `store_fl_dataset()` and `load_client_fl_dataset()` function from `DatasetClasses.py` module.

- `DATASET_MODE`: Sets the dataset mode to determine the subset of commands loaded. Options available: ’full_ds’, ’30cmd’, ’26cmd’, ’20cmd’.
- `RANDOM_STATE`: Sets the random state for reproducibility.
- `MODEL`: Specifies the model to load for training and evaluation. Options available: ’cnn’, ’mn2’.
- `LOG_PATH`: Specifies the path for storing logs.

Our solution incorporates also a centralized learning setup in `src/Project/centralized` directory, executing `centralized.py` file. It has the following configuration options:

- `BATCH_SIZE_CENTRALIZED`: Sets the batch size for centralized training.
- `EPOCHS`: Specifies the number of epochs for centralized training.
- `EXPORT_MODEL`: Determines if the trained model should be exported.

6 Evaluation results

6.1 Experimental scenarios

The following sections present the experimental results obtained from the server evaluations of the implemented solution. The server contains unseen data for benchmarking purposes (to test the performance of the global model at each round). All the results have been evaluated using MobileNetV2 model and Google Speech Commands v2 dataset. We can distinguish the following experimental scenarios:

- Real Network scenario: The FL system is communicated through a real network, 5G or not. The program uses the Flower Edge-Client-Engine (ECE) architecture.

  In that scenario the dataset is identically distributed among clients (IID-Data). That means that each client has the same audio files as the others. The whole dataset is divided among clients without filtering speakers.

- Simulated scenario: The FL system communicates through a virtualized network, using Virtual Proxys to communicate with each client. The program uses the Flower Virtual-Client-Engine (VCE) architecture.

  In that scenario we evaluate the performance of FL in non-IID data setup (dividing the data by speakers and by commands). In that case, the speakers are filtered, we only keep the speakers that contain all the commands in the chosen dataset mode. Two aggregation algorithms are used: FedAvg and FedProx.

Figures 6.1 & 6.2 show the results of the training stage at a centralized environment using '26cmd' dataset mode. The first one uses the whole dataset and the second one filter the speakers that have all the commands. We can use these results as reference to compare the performance of centralized setup and the FL setup.

In a centralized environment, the training process and data reside on a single computing system. In this context, a centralized environment refers to a scenario where all training data and computation are concentrated and managed by a single entity. The computing system typically receives and processes the entire dataset, executes the training algorithm, and updates the model parameters based on the complete dataset.

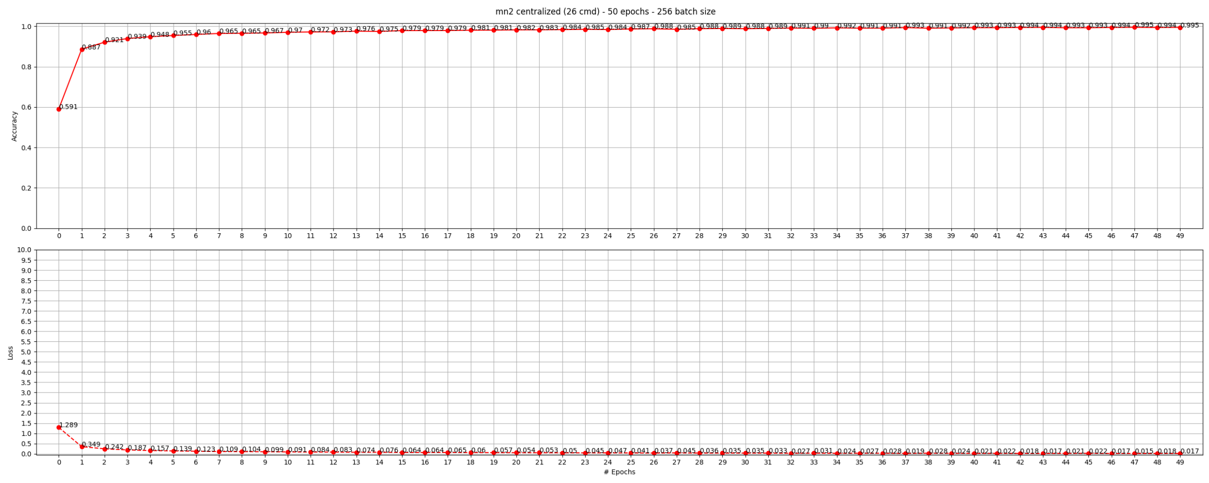


Figure 6.1. MobileNetV2 training results (26 cmd) in a centralized environment.

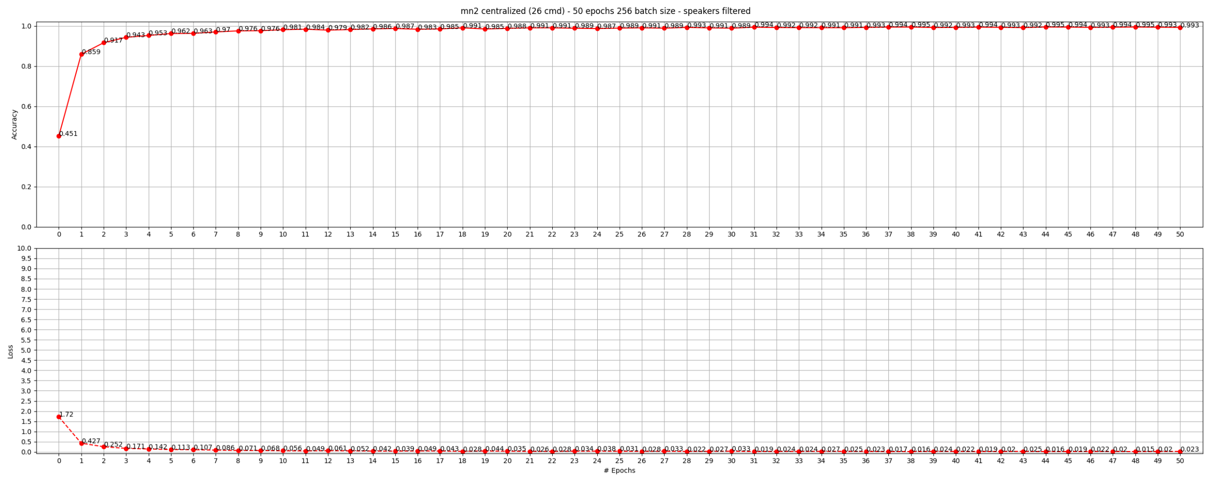
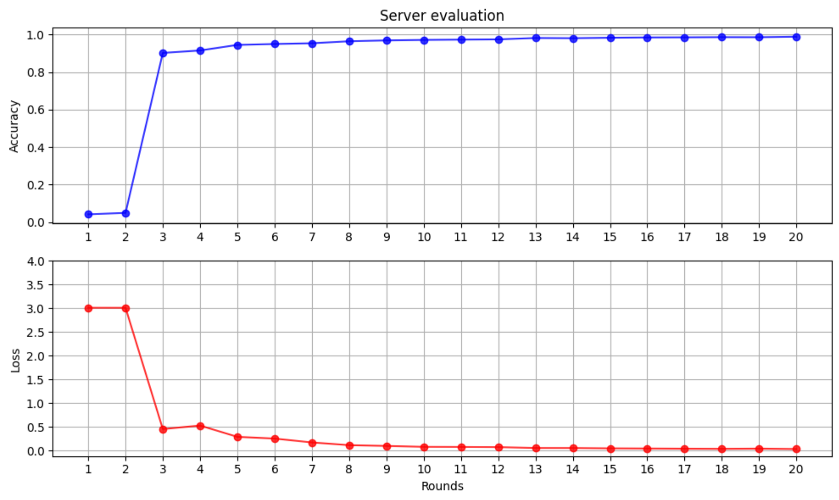


Figure 6.2. MobileNetV2 training results (26 cmd) in a centralized environment, speakers filtered.

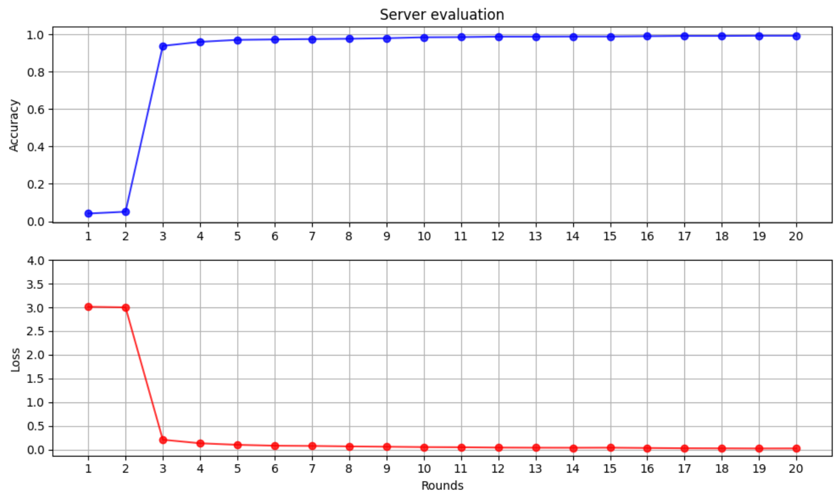
6.2 Real Network scenario

6.2.1 FLAIR strategy

As shown in Figure 6.3, the convergence time using 20 randomly shuffled speech commands across 4 federated learning (FL) clients is not significantly notable. In the 3rd and 4th epoch, the FLAIR strategy shows slightly improved accuracy and loss. However, we cannot exctract solid conclusions based on this information. Nevertheless, we belive that customizing a strategy to optimize the FL resources across all clients could potentially lead to an improved model convergence time.



(a) Simple FedAvg *strategy*

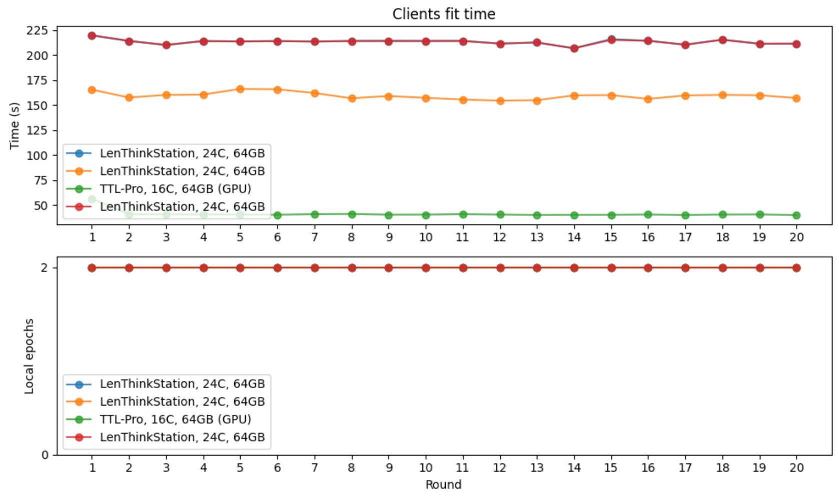


(b) FedAvg *applying the FLAIR strategy*

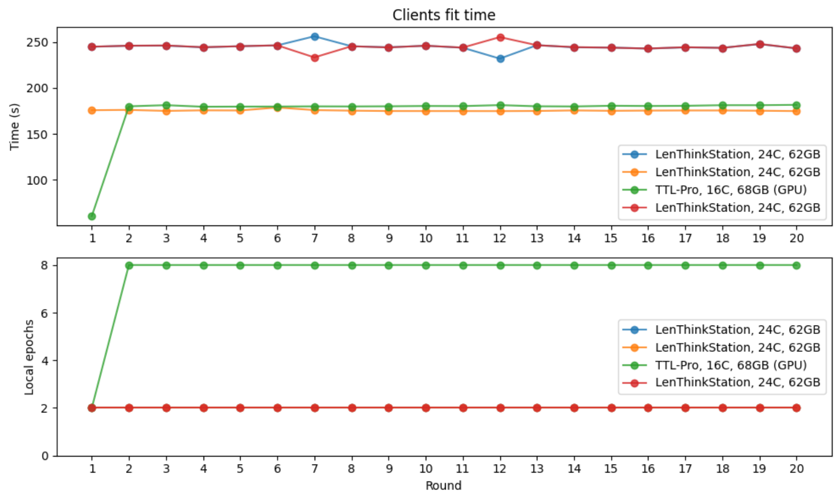
Figure 6.3. Differences between the FL strategies

In Figure 6.4, the FLAIR strategy notices in the initial epochs that the client *TTL-Pro*, which has GPU acceleration, has significantly lower fit times compared to the other clients. Consequently, the FLAIR strategy determines that this client should perform additional work by configuring it to perform more local epochs per round. This initial adjustment is a good starting point for the strategy.

However, the fit metrics and the assigned local epochs may not fully equalize the fit time across all clients, as desired, as shown in Figure 6.5. This indicates that improvements can be made to optimize the configuration distribution. Additionally, it is crucial to take into account potential future edge cases that have not been considered in this specific scenario to archive a better overall effectiveness of the FLAIR strategy.



(a) Simple FedAvg *strategy (local epochs per client)*



(b) FedAvg *applying the FLAIR strategy (local epochs per client)*

Figure 6.4. Differences between the FL strategies

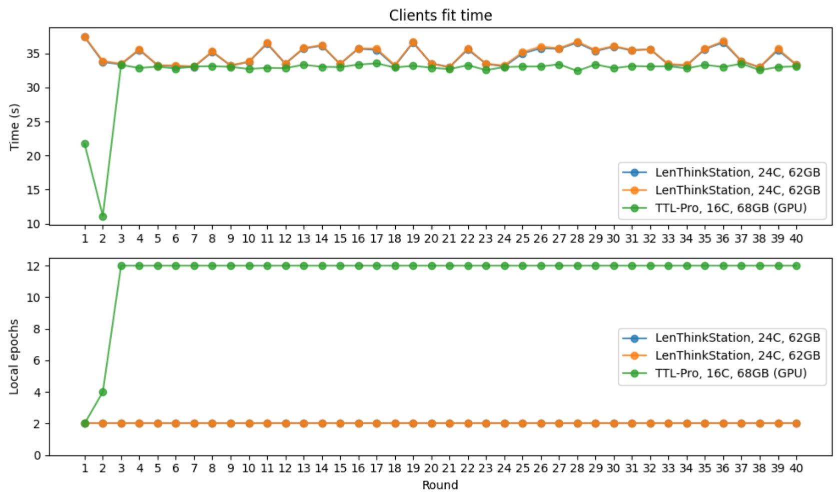
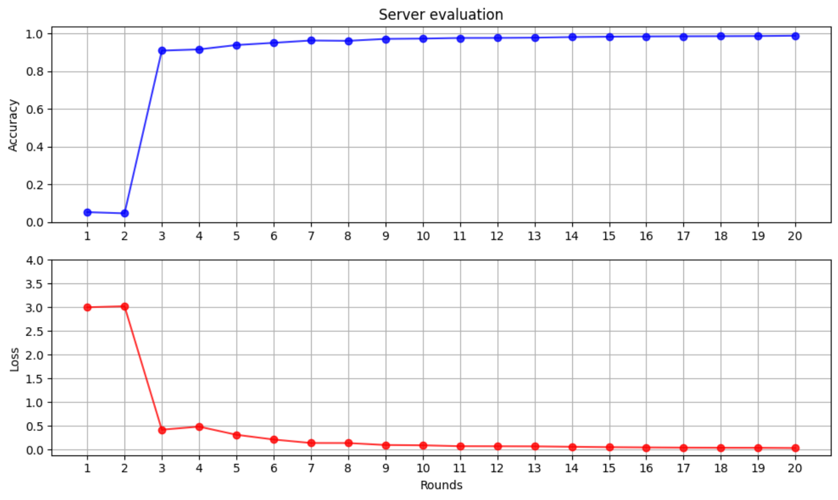


Figure 6.5. Clients resources optimization using the FLAIR strategy

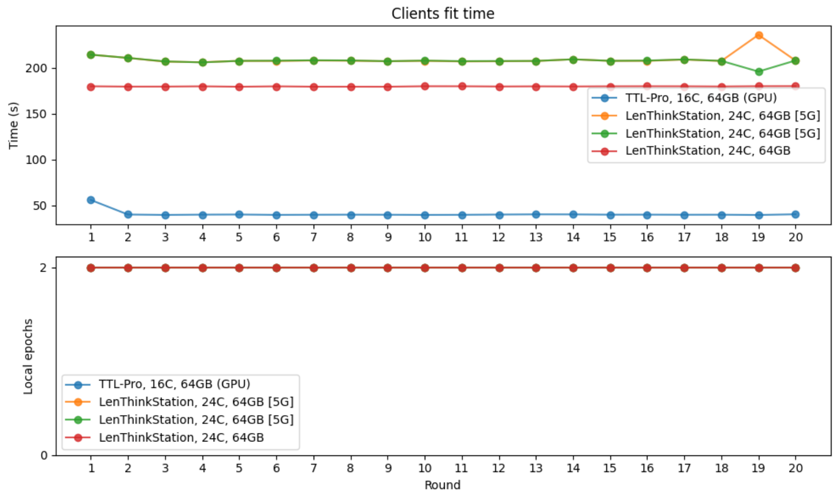
6.2.2 5G scenario

In this particular scenario, it is interesting to explore and demonstrate the flexibility of performing a FL training over a 5G network, while ensuring a feasible execution of the training process. As illustrated in Figure 6.6, the same experiment as before, involving four clients with randomly shuffled datasets of 20 commands, was successfully performed. The FL training process obtained similar results as before demonstrating that it is both viable and possible to execute FL training over a 5G network.

Therefore, this scenario introduces a diverse network setup where some clients are connected via 5G while others are not. This flexibility in connectivity options enables various possibilities and allows flexibility in performing FL experiments.



(a) Simple FedAvg *strategy (local epochs per client)*



(b) FedAvg *applying the FLAIR strategy (local epochs per client)*

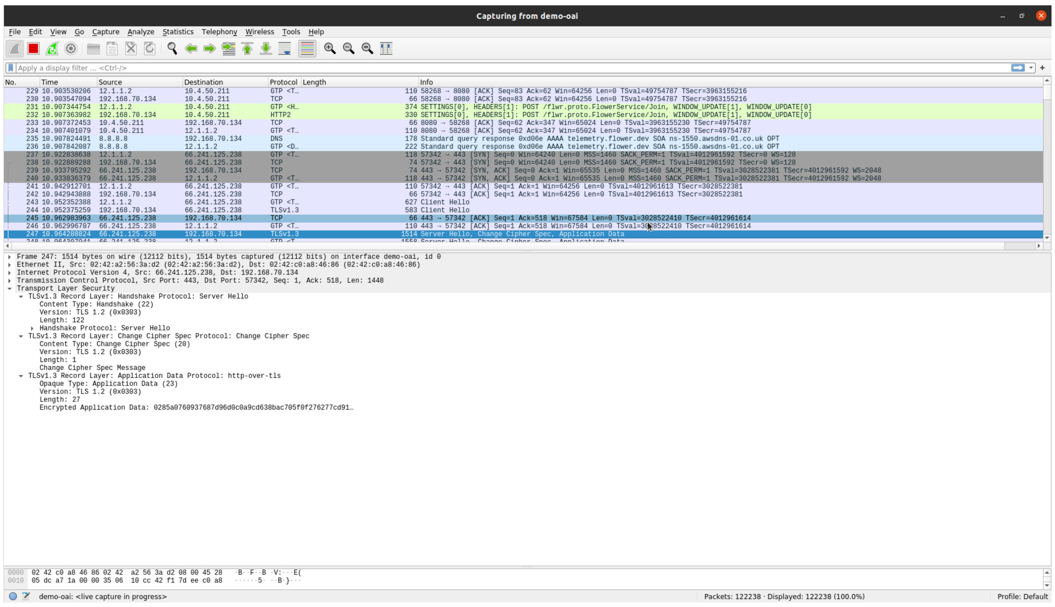
Figure 6.6. FL training with client clients connected though 5G

Finally, in Figure 6.7 is shown the FL server handshake with a FL client, then a protocol buffer is indeed needed to perform the method calls over the gRPC network that is defined by Flower as seen in Figure 6.8 and to end up in Figure 6.9 when a FL round is finished the FL client returns the model parameter exchange, in this case is sequenced in mane segments.

Figure 6.7 illustrates the handshake process between the FL server and an FL client.

This handshake is necessary to establish communication and synchronization between the server and client. To perform the method calls over the gRPC network, a protocol buffer (protobuf) is used, as shown in Figure 6.8. The protobuf defined by Flower defines the structure and format of the data exchanged between the server and client.

After completing a FL round, as shown in Figure 6.9, the FL client returns the model parameter exchange to the server. In this case, the model parameter exchange is divided into multiple segments surely for enabling efficient transmission and handling of large amounts of data (parameters).



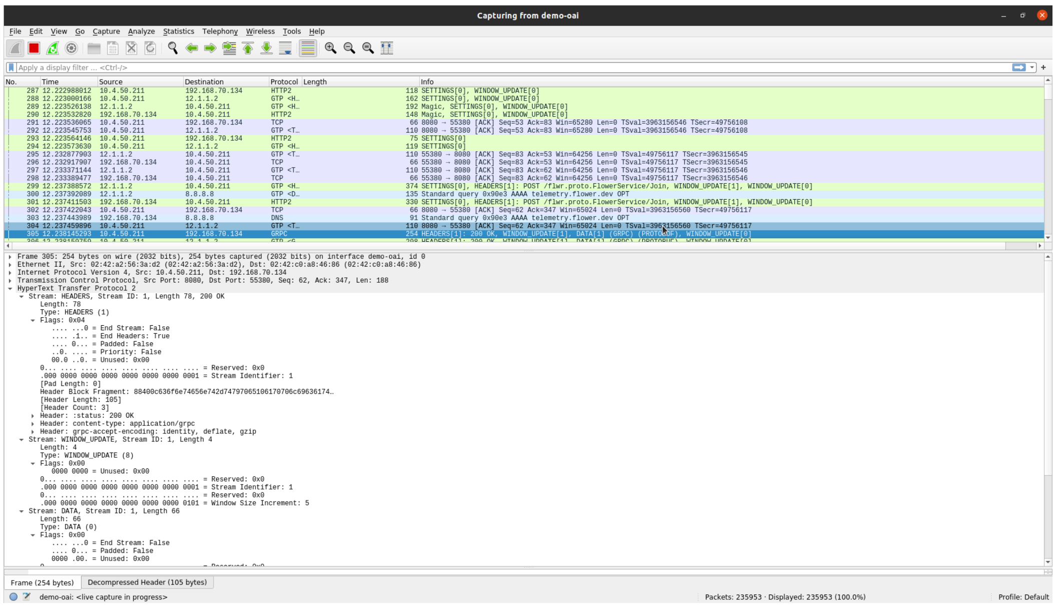
```
No.  Time         Source          Destination     Protocol Length  Info
 229 10.903530206 12.1.1.2        10.4.50.211     GTP <T…      110 58268 → 8080 [ACK] Seq=83 Ack=62 Win=64256 Len=0 TSval=49754787 TSecr=3963155216
 230 10.903547094 192.168.70.134  10.4.50.211     TCP           66 58268 → 8080 [ACK] Seq=83 Ack=62 Win=64256 Len=0 TSval=49754787 TSecr=3963155216
 231 10.907344754 12.1.1.2        10.4.50.211     GTP <H…      374 SETTINGS[0], HEADERS[1]: POST /flwr.proto.FlowerService/Join, WINDOW_UPDATE[1], WINDOW_UPDATE[0]
 232 10.907363982 192.168.70.134  10.4.50.211     HTTP2        330 SETTINGS[0], HEADERS[1]: POST /flwr.proto.FlowerService/Join, WINDOW_UPDATE[1], WINDOW_UPDATE[0]
 233 10.907372453 10.4.50.211     192.168.70.134  TCP           66 8080 → 58268 [ACK] Seq=62 Ack=347 Win=65024 Len=0 TSval=3963155230 TSecr=49754787
 234 10.907401079 10.4.50.211     12.1.1.2        GTP <T…      110 8080 → 58268 [ACK] Seq=62 Ack=347 Win=65024 Len=0 TSval=3963155230 TSecr=49754787
 235 10.907824491 8.8.8.8         192.168.70.134  DNS          178 Standard query response 0xd06e AAAA telemetry.flower.dev SOA ns-1550.awsdns-01.co.uk OPT
 236 10.907842087 8.8.8.8         12.1.1.2        GTP <D…      222 Standard query response 0xd06e AAAA telemetry.flower.dev SOA ns-1550.awsdns-01.co.uk OPT
 237 10.922838638 12.1.1.2        66.241.125.238  GTP <T…      118 57342 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=4012961592 TSecr=0 WS=128
 238 10.922889288 192.168.70.134  66.241.125.238  TCP           74 57342 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=4012961592 TSecr=0 WS=128
 239 10.933795292 66.241.125.238  192.168.70.134  TCP           74 443 → 57342 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 SACK_PERM=1 TSval=3028522381 TSecr=4012961592 WS=2048
 240 10.933836379 66.241.125.238  12.1.1.2        GTP <T…      118 443 → 57342 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 SACK_PERM=1 TSval=3028522381 TSecr=4012961592 WS=2048
 241 10.942912701 12.1.1.2        66.241.125.238  GTP <T…      110 57342 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=4012961613 TSecr=3028522381
 242 10.942943888 192.168.70.134  66.241.125.238  TCP           66 57342 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=4012961613 TSecr=3028522381
 243 10.952352388 12.1.1.2        66.241.125.238  GTP <T…      627 Client Hello
 244 10.952375259 192.168.70.134  66.241.125.238  TLSv1.3      583 Client Hello
 245 10.962983963 66.241.125.238  192.168.70.134  TCP           66 443 → 57342 [ACK] Seq=1 Ack=518 Win=67584 Len=0 TSval=3028522410 TSecr=4012961614
 246 10.962996707 66.241.125.238  12.1.1.2        GTP <T…      110 443 → 57342 [ACK] Seq=1 Ack=518 Win=67584 Len=0 TSval=3028522410 TSecr=4012961614
 247 10.964288824 66.241.125.238  192.168.70.134  TLSv1.3     1514 Server Hello, Change Cipher Spec, Application Data
```

```
▸ Frame 247: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface demo-oai, id 0
▸ Ethernet II, Src: 02:42:a2:56:3a:d2 (02:42:a2:56:3a:d2), Dst: 02:42:c0:a8:46:86 (02:42:c0:a8:46:86)
▸ Internet Protocol Version 4, Src: 66.241.125.238, Dst: 192.168.70.134
▸ Transmission Control Protocol, Src Port: 443, Dst Port: 57342, Seq: 1, Ack: 518, Len: 1448
▾ Transport Layer Security
  ▾ TLSv1.3 Record Layer: Handshake Protocol: Server Hello
      Content Type: Handshake (22)
      Version: TLS 1.2 (0x0303)
      Length: 122
    ▸ Handshake Protocol: Server Hello
  ▾ TLSv1.3 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
      Content Type: Change Cipher Spec (20)
      Version: TLS 1.2 (0x0303)
      Length: 1
      Change Cipher Spec Message
  ▾ TLSv1.3 Record Layer: Application Data Protocol: http-over-tls
      Opaque Type: Application Data (23)
      Version: TLS 1.2 (0x0303)
      Length: 27
      Encrypted Application Data: 0285a0760937687d96d0c0a9cd638bac705f0f276277cd91…
```

```
0000  02 42 c0 a8 46 86 02 42  a2 56 3a d2 08 00 45 28   ·B··F··B ·V:···E(
0010  05 dc a7 1a 00 00 35 06  10 cc 42 f1 7d ee c0 a8   ······5· ··B·}···
```

```
demo-oai: <live capture in progress>          Packets: 122238 · Displayed: 122238 (100.0%)          Profile: Default
```

Figure 6.7. Server and clients handshake

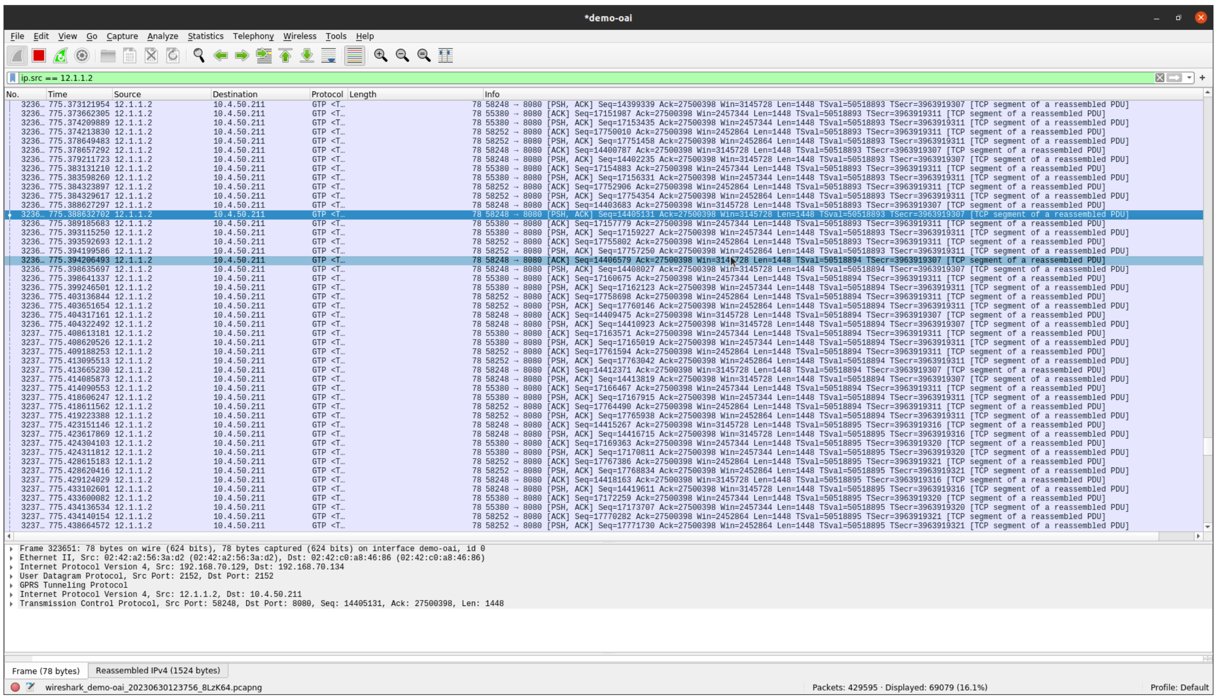


```
No.  Time         Source          Destination     Protocol Length  Info
 287 12.222988012 10.4.50.211     192.168.70.134  HTTP2        118 SETTINGS[0], WINDOW_UPDATE[0]
 288 12.223000166 10.4.50.211     12.1.1.2        GTP <H…      162 SETTINGS[0], WINDOW_UPDATE[0]
 289 12.223526138 12.1.1.2        10.4.50.211     GTP <H…      192 Magic, SETTINGS[0], WINDOW_UPDATE[0]
 290 12.223532820 192.168.70.134  10.4.50.211     HTTP2        148 Magic, SETTINGS[0], WINDOW_UPDATE[0]
 291 12.223536065 10.4.50.211     192.168.70.134  TCP           66 8080 → 55380 [ACK] Seq=53 Ack=83 Win=65280 Len=0 TSval=3963156546 TSecr=49756108
 292 12.223545753 10.4.50.211     12.1.1.2        GTP <T…      110 8080 → 55380 [ACK] Seq=53 Ack=83 Win=65280 Len=0 TSval=3963156546 TSecr=49756108
 293 12.223564146 10.4.50.211     192.168.70.134  HTTP2         75 SETTINGS[0]
 294 12.223573630 10.4.50.211     12.1.1.2        GTP <H…      119 SETTINGS[0]
 295 12.232877903 12.1.1.2        10.4.50.211     GTP <T…      110 55380 → 8080 [ACK] Seq=83 Ack=53 Win=64256 Len=0 TSval=49756117 TSecr=3963156545
 296 12.232917907 192.168.70.134  10.4.50.211     TCP           66 55380 → 8080 [ACK] Seq=83 Ack=53 Win=64256 Len=0 TSval=49756117 TSecr=3963156545
 297 12.233371144 12.1.1.2        10.4.50.211     GTP <T…      110 55380 → 8080 [ACK] Seq=83 Ack=62 Win=64256 Len=0 TSval=49756117 TSecr=3963156546
 298 12.233389477 192.168.70.134  10.4.50.211     TCP           66 55380 → 8080 [ACK] Seq=83 Ack=62 Win=64256 Len=0 TSval=49756117 TSecr=3963156546
 299 12.237388572 12.1.1.2        10.4.50.211     GTP <H…      374 SETTINGS[0], HEADERS[1]: POST /flwr.proto.FlowerService/Join, WINDOW_UPDATE[1], WINDOW_UPDATE[0]
 300 12.237392089 12.1.1.2        8.8.8.8         GTP <D…      135 Standard query 0x90e3 AAAA telemetry.flower.dev OPT
 301 12.237411503 192.168.70.134  10.4.50.211     HTTP2        330 SETTINGS[0], HEADERS[1]: POST /flwr.proto.FlowerService/Join, WINDOW_UPDATE[1], WINDOW_UPDATE[0]
 302 12.237422043 10.4.50.211     192.168.70.134  TCP           66 8080 → 55380 [ACK] Seq=62 Ack=347 Win=65024 Len=0 TSval=3963156560 TSecr=49756117
 303 12.237443989 192.168.70.134  8.8.8.8         DNS           91 Standard query 0x90e3 AAAA telemetry.flower.dev OPT
 304 12.237459896 10.4.50.211     12.1.1.2        GTP <T…      110 8080 → 55380 [ACK] Seq=62 Ack=347 Win=65024 Len=0 TSval=3963156560 TSecr=49756117
 305 12.238145293 10.4.50.211     192.168.70.134  GRPC         254 HEADERS[1]: 200 OK, WINDOW_UPDATE[1], DATA[1] (GRPC) (PROTOBUF), WINDOW_UPDATE[0]
```

```
▸ Frame 305: 254 bytes on wire (2032 bits), 254 bytes captured (2032 bits) on interface demo-oai, id 0
▸ Ethernet II, Src: 02:42:a2:56:3a:d2 (02:42:a2:56:3a:d2), Dst: 02:42:c0:a8:46:86 (02:42:c0:a8:46:86)
▸ Internet Protocol Version 4, Src: 10.4.50.211, Dst: 192.168.70.134
▸ Transmission Control Protocol, Src Port: 8080, Dst Port: 55380, Seq: 62, Ack: 347, Len: 188
▾ HyperText Transfer Protocol 2
  ▾ Stream: HEADERS, Stream ID: 1, Length 78, 200 OK
      Length: 78
      Type: HEADERS (1)
    ▾ Flags: 0x04
        .... ...0 = End Stream: False
        .... .1.. = End Headers: True
        .... 0... = Padded: False
        ..0. .... = Priority: False
        00.0 ..0. = Unused: 0x00
      0... .... .... .... .... .... .... .... = Reserved: 0x0
      .000 0000 0000 0000 0000 0000 0000 0001 = Stream Identifier: 1
      [Pad Length: 0]
      Header Block Fragment: 88400c636f6e74656e742d74797065106170706c69636174…
      [Header Length: 105]
      [Header Count: 3]
    ▸ Header: :status: 200 OK
    ▸ Header: content-type: application/grpc
    ▸ Header: grpc-accept-encoding: identity, deflate, gzip
  ▾ Stream: WINDOW_UPDATE, Stream ID: 1, Length 4
      Length: 4
      Type: WINDOW_UPDATE (8)
    ▾ Flags: 0x00
        0000 0000 = Unused: 0x00
      0... .... .... .... .... .... .... .... = Reserved: 0x0
      .000 0000 0000 0000 0000 0000 0000 0001 = Stream Identifier: 1
      0... .... .... .... .... .... .... .... = Reserved: 0x0
      .000 0000 0000 0000 0000 0000 0000 0101 = Window Size Increment: 5
  ▾ Stream: DATA, Stream ID: 1, Length 66
      Length: 66
      Type: DATA (0)
    ▾ Flags: 0x00
        .... ...0 = End Stream: False
        .... 0... = Padded: False
        0000 .00. = Unused: 0x00
```

```
demo-oai: <live capture in progress>          Packets: 235953 · Displayed: 235953 (100.0%)          Profile: Default
```

Figure 6.8. Profocol buffer (gRPC)


*demo-oai
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
ip.src == 12.1.1.2

No.     Time          Source      Destination   Protocol Length  Info
3236… 775.373121954 12.1.1.2    10.4.50.211   GTP <T…   78 58248 → 8080 [PSH, ACK] Seq=14399339 Ack=27500398 Win=3145728 Len=1448 TSval=50518893 TSecr=3963919307 [TCP segment of a reassembled PDU]
3236… 775.373662305 12.1.1.2    10.4.50.211   GTP <T…   78 55380 → 8080 [ACK] Seq=17151987 Ack=27500398 Win=2457344 Len=1448 TSval=50518893 TSecr=3963919311 [TCP segment of a reassembled PDU]
3236… 775.374209889 12.1.1.2    10.4.50.211   GTP <T…   78 55380 → 8080 [PSH, ACK] Seq=17153435 Ack=27500398 Win=2457344 Len=1448 TSval=50518893 TSecr=3963919311 [TCP segment of a reassembled PDU]
3236… 775.374213830 12.1.1.2    10.4.50.211   GTP <T…   78 58252 → 8080 [ACK] Seq=17750010 Ack=27500398 Win=2452864 Len=1448 TSval=50518893 TSecr=3963919311 [TCP segment of a reassembled PDU]
3236… 775.378649483 12.1.1.2    10.4.50.211   GTP <T…   78 58252 → 8080 [PSH, ACK] Seq=17751458 Ack=27500398 Win=2452864 Len=1448 TSval=50518893 TSecr=3963919311 [TCP segment of a reassembled PDU]
3236… 775.378657292 12.1.1.2    10.4.50.211   GTP <T…   78 58248 → 8080 [ACK] Seq=14400787 Ack=27500398 Win=3145728 Len=1448 TSval=50518893 TSecr=3963919307 [TCP segment of a reassembled PDU]
3236… 775.379211723 12.1.1.2    10.4.50.211   GTP <T…   78 58248 → 8080 [PSH, ACK] Seq=14402235 Ack=27500398 Win=3145728 Len=1448 TSval=50518893 TSecr=3963919307 [TCP segment of a reassembled PDU]
3236… 775.383131210 12.1.1.2    10.4.50.211   GTP <T…   78 55380 → 8080 [ACK] Seq=17154883 Ack=27500398 Win=2457344 Len=1448 TSval=50518893 TSecr=3963919311 [TCP segment of a reassembled PDU]
3236… 775.383598260 12.1.1.2    10.4.50.211   GTP <T…   78 55380 → 8080 [PSH, ACK] Seq=17156331 Ack=27500398 Win=2457344 Len=1448 TSval=50518893 TSecr=3963919311 [TCP segment of a reassembled PDU]
3236… 775.384323897 12.1.1.2    10.4.50.211   GTP <T…   78 58252 → 8080 [ACK] Seq=17752906 Ack=27500398 Win=2452864 Len=1448 TSval=50518893 TSecr=3963919311 [TCP segment of a reassembled PDU]
3236… 775.384329617 12.1.1.2    10.4.50.211   GTP <T…   78 58252 → 8080 [PSH, ACK] Seq=17754354 Ack=27500398 Win=2452864 Len=1448 TSval=50518893 TSecr=3963919311 [TCP segment of a reassembled PDU]
3236… 775.388627297 12.1.1.2    10.4.50.211   GTP <T…   78 58248 → 8080 [ACK] Seq=14403683 Ack=27500398 Win=3145728 Len=1448 TSval=50518893 TSecr=3963919307 [TCP segment of a reassembled PDU]
3236… 775.388632702 12.1.1.2    10.4.50.211   GTP <T…   78 58248 → 8080 [PSH, ACK] Seq=14405131 Ack=27500398 Win=3145728 Len=1448 TSval=50518893 TSecr=3963919307 [TCP segment of a reassembled PDU]
3236… 775.389185683 12.1.1.2    10.4.50.211   GTP <T…   78 55380 → 8080 [ACK] Seq=17157779 Ack=27500398 Win=2457344 Len=1448 TSval=50518893 TSecr=3963919311 [TCP segment of a reassembled PDU]
3236… 775.393115250 12.1.1.2    10.4.50.211   GTP <T…   78 55380 → 8080 [PSH, ACK] Seq=17159227 Ack=27500398 Win=2457344 Len=1448 TSval=50518893 TSecr=3963919311 [TCP segment of a reassembled PDU]
3236… 775.393592693 12.1.1.2    10.4.50.211   GTP <T…   78 58252 → 8080 [ACK] Seq=17755802 Ack=27500398 Win=2452864 Len=1448 TSval=50518893 TSecr=3963919311 [TCP segment of a reassembled PDU]
3236… 775.394199586 12.1.1.2    10.4.50.211   GTP <T…   78 58252 → 8080 [PSH, ACK] Seq=17757250 Ack=27500398 Win=2452864 Len=1448 TSval=50518893 TSecr=3963919311 [TCP segment of a reassembled PDU]
3236… 775.394206493 12.1.1.2    10.4.50.211   GTP <T…   78 58248 → 8080 [ACK] Seq=14406579 Ack=27500398 Win=3145728 Len=1448 TSval=50518894 TSecr=3963919307 [TCP segment of a reassembled PDU]
3236… 775.398635697 12.1.1.2    10.4.50.211   GTP <T…   78 58248 → 8080 [PSH, ACK] Seq=14408027 Ack=27500398 Win=3145728 Len=1448 TSval=50518894 TSecr=3963919307 [TCP segment of a reassembled PDU]
3236… 775.398641337 12.1.1.2    10.4.50.211   GTP <T…   78 55380 → 8080 [ACK] Seq=17160675 Ack=27500398 Win=2457344 Len=1448 TSval=50518894 TSecr=3963919311 [TCP segment of a reassembled PDU]
3236… 775.399246501 12.1.1.2    10.4.50.211   GTP <T…   78 55380 → 8080 [PSH, ACK] Seq=17162123 Ack=27500398 Win=2457344 Len=1448 TSval=50518894 TSecr=3963919311 [TCP segment of a reassembled PDU]
3236… 775.403136844 12.1.1.2    10.4.50.211   GTP <T…   78 58252 → 8080 [ACK] Seq=17758698 Ack=27500398 Win=2452864 Len=1448 TSval=50518894 TSecr=3963919311 [TCP segment of a reassembled PDU]
3236… 775.403651654 12.1.1.2    10.4.50.211   GTP <T…   78 58252 → 8080 [PSH, ACK] Seq=17760146 Ack=27500398 Win=2452864 Len=1448 TSval=50518894 TSecr=3963919311 [TCP segment of a reassembled PDU]
3236… 775.404317161 12.1.1.2    10.4.50.211   GTP <T…   78 58248 → 8080 [ACK] Seq=14409475 Ack=27500398 Win=3145728 Len=1448 TSval=50518894 TSecr=3963919307 [TCP segment of a reassembled PDU]
3236… 775.404322492 12.1.1.2    10.4.50.211   GTP <T…   78 58248 → 8080 [PSH, ACK] Seq=14410923 Ack=27500398 Win=3145728 Len=1448 TSval=50518894 TSecr=3963919307 [TCP segment of a reassembled PDU]
3237… 775.408613181 12.1.1.2    10.4.50.211   GTP <T…   78 55380 → 8080 [ACK] Seq=17163571 Ack=27500398 Win=2457344 Len=1448 TSval=50518894 TSecr=3963919311 [TCP segment of a reassembled PDU]
3237… 775.408620526 12.1.1.2    10.4.50.211   GTP <T…   78 55380 → 8080 [PSH, ACK] Seq=17165019 Ack=27500398 Win=2457344 Len=1448 TSval=50518894 TSecr=3963919311 [TCP segment of a reassembled PDU]
3237… 775.409188253 12.1.1.2    10.4.50.211   GTP <T…   78 58252 → 8080 [ACK] Seq=17761594 Ack=27500398 Win=2452864 Len=1448 TSval=50518894 TSecr=3963919311 [TCP segment of a reassembled PDU]
3237… 775.413095513 12.1.1.2    10.4.50.211   GTP <T…   78 58252 → 8080 [PSH, ACK] Seq=17763042 Ack=27500398 Win=2452864 Len=1448 TSval=50518894 TSecr=3963919311 [TCP segment of a reassembled PDU]
3237… 775.413665230 12.1.1.2    10.4.50.211   GTP <T…   78 58248 → 8080 [ACK] Seq=14412371 Ack=27500398 Win=3145728 Len=1448 TSval=50518894 TSecr=3963919307 [TCP segment of a reassembled PDU]
3237… 775.414085873 12.1.1.2    10.4.50.211   GTP <T…   78 58248 → 8080 [PSH, ACK] Seq=14413819 Ack=27500398 Win=3145728 Len=1448 TSval=50518894 TSecr=3963919307 [TCP segment of a reassembled PDU]
3237… 775.414090553 12.1.1.2    10.4.50.211   GTP <T…   78 55380 → 8080 [ACK] Seq=17166467 Ack=27500398 Win=2457344 Len=1448 TSval=50518894 TSecr=3963919311 [TCP segment of a reassembled PDU]
3237… 775.418606247 12.1.1.2    10.4.50.211   GTP <T…   78 55380 → 8080 [PSH, ACK] Seq=17167915 Ack=27500398 Win=2457344 Len=1448 TSval=50518894 TSecr=3963919311 [TCP segment of a reassembled PDU]
3237… 775.418611562 12.1.1.2    10.4.50.211   GTP <T…   78 58252 → 8080 [ACK] Seq=17764490 Ack=27500398 Win=2452864 Len=1448 TSval=50518894 TSecr=3963919311 [TCP segment of a reassembled PDU]
3237… 775.419223388 12.1.1.2    10.4.50.211   GTP <T…   78 58252 → 8080 [PSH, ACK] Seq=17765938 Ack=27500398 Win=2452864 Len=1448 TSval=50518894 TSecr=3963919311 [TCP segment of a reassembled PDU]
3237… 775.423151146 12.1.1.2    10.4.50.211   GTP <T…   78 58248 → 8080 [ACK] Seq=14415267 Ack=27500398 Win=3145728 Len=1448 TSval=50518895 TSecr=3963919316 [TCP segment of a reassembled PDU]
3237… 775.423617869 12.1.1.2    10.4.50.211   GTP <T…   78 58248 → 8080 [PSH, ACK] Seq=14416715 Ack=27500398 Win=3145728 Len=1448 TSval=50518895 TSecr=3963919316 [TCP segment of a reassembled PDU]
3237… 775.424304103 12.1.1.2    10.4.50.211   GTP <T…   78 55380 → 8080 [ACK] Seq=17169363 Ack=27500398 Win=2457344 Len=1448 TSval=50518895 TSecr=3963919320 [TCP segment of a reassembled PDU]
3237… 775.424311812 12.1.1.2    10.4.50.211   GTP <T…   78 55380 → 8080 [PSH, ACK] Seq=17170811 Ack=27500398 Win=2457344 Len=1448 TSval=50518895 TSecr=3963919320 [TCP segment of a reassembled PDU]
3237… 775.428615183 12.1.1.2    10.4.50.211   GTP <T…   78 58252 → 8080 [ACK] Seq=17767386 Ack=27500398 Win=2452864 Len=1448 TSval=50518895 TSecr=3963919321 [TCP segment of a reassembled PDU]
3237… 775.428620416 12.1.1.2    10.4.50.211   GTP <T…   78 58252 → 8080 [PSH, ACK] Seq=17768834 Ack=27500398 Win=2452864 Len=1448 TSval=50518895 TSecr=3963919321 [TCP segment of a reassembled PDU]
3237… 775.429124029 12.1.1.2    10.4.50.211   GTP <T…   78 58248 → 8080 [ACK] Seq=14418163 Ack=27500398 Win=3145728 Len=1448 TSval=50518895 TSecr=3963919316 [TCP segment of a reassembled PDU]
3237… 775.433102601 12.1.1.2    10.4.50.211   GTP <T…   78 58248 → 8080 [PSH, ACK] Seq=14419611 Ack=27500398 Win=3145728 Len=1448 TSval=50518895 TSecr=3963919316 [TCP segment of a reassembled PDU]
3237… 775.433600082 12.1.1.2    10.4.50.211   GTP <T…   78 55380 → 8080 [ACK] Seq=17172259 Ack=27500398 Win=2457344 Len=1448 TSval=50518895 TSecr=3963919320 [TCP segment of a reassembled PDU]
3237… 775.434136534 12.1.1.2    10.4.50.211   GTP <T…   78 55380 → 8080 [PSH, ACK] Seq=17173707 Ack=27500398 Win=2457344 Len=1448 TSval=50518895 TSecr=3963919320 [TCP segment of a reassembled PDU]
3237… 775.434140154 12.1.1.2    10.4.50.211   GTP <T…   78 58252 → 8080 [ACK] Seq=17770282 Ack=27500398 Win=2452864 Len=1448 TSval=50518895 TSecr=3963919321 [TCP segment of a reassembled PDU]
3237… 775.438664572 12.1.1.2    10.4.50.211   GTP <T…   78 58252 → 8080 [PSH, ACK] Seq=17771730 Ack=27500398 Win=2452864 Len=1448 TSval=50518895 TSecr=3963919321 [TCP segment of a reassembled PDU]

Frame 323651: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface demo-oai, id 0
Ethernet II, Src: 02:42:a2:56:3a:d2 (02:42:a2:56:3a:d2), Dst: 02:42:c0:a8:46:86 (02:42:c0:a8:46:86)
Internet Protocol Version 4, Src: 192.168.70.129, Dst: 192.168.70.134
User Datagram Protocol, Src Port: 2152, Dst Port: 2152
GPRS Tunneling Protocol
Internet Protocol Version 4, Src: 12.1.1.2, Dst: 10.4.50.211
Transmission Control Protocol, Src Port: 58248, Dst Port: 8080, Seq: 14405131, Ack: 27500398, Len: 1448

Frame (78 bytes)   Reassembled IPv4 (1524 bytes)
wireshark_demo-oai_20230630123756_8LzK64.pcapng      Packets: 429595 · Displayed: 69079 (16.1%)      Profile: Default
```

Figure 6.9. Model parameter exchange

6.3 Simulated scenario

In the simulated scenario all the speakers are filtered, keeping only the ones whose contain all the commands. Two main dataset modes will be used ('26cmd' and '20cmd') and two main aggregation algorithm were tested ('FedAvg' [6] and 'FedProx' [4]).

FedAvg and FedProx are popular algorithms in Federated Learning. FedAvg aims to train a global model by aggregating local model updates from multiple client devices while preserving data privacy. It achieves this by averaging model parameters in each round of communication. FedProx addresses the challenge of non-IID data in FL by introducing a regularization term that encourages local models to be close to the global model. This helps mitigate the impact of non-IID data and improves the global model's generalization.

The following simulation setups were executed:

- Speaker segregation:
  - 5 train speakers per client:
    - FedAvg ('26cmd', 43 clients, 142 rounds, 32000 audio files) $\rightarrow$ Figure 6.10
    - FedProx $\mu = 0.01$ ('26cmd', 43 clients, 142 rounds, 32000 audio files) $\rightarrow$ Figure 6.11
  - 2 train speakers per client:
    - FedAvg ('26cmd', 108 clients, 300 rounds, 32000 audio files) $\rightarrow$ Figure 6.12
    - FedProx $\mu = 0.01$ ('26cmd', 108 clients, 300 rounds, 32000 audio files) $\rightarrow$ Figure 6.13

- 1 train speaker + 1 shared speaker per client:
    - FedProx $\mu = 0.01$ ('26cmd', 216 clients, 300 rounds, 32000 audio files) $\rightarrow$ Figure 6.14
- 1 train speaker per client:
    - FedAvg ('26cmd', 216 clients, 500 rounds, 32000 audio files) $\rightarrow$ Figure 6.15
    - FedProx $\mu = 0.01$ ('26cmd', 216 clients, 500 rounds, 32000 audio files) $\rightarrow$ Figure 6.16
    - FedProx $\mu = 0.001$ ('26cmd', 216 clients, 500 rounds, 32000 audio files) $\rightarrow$ Figure 6.17
    - FedProx $\mu = 0.01$ ('20cmd', 367 clients, 900 rounds, 50000 audio files) $\rightarrow$ Figure 6.18

• Speaker seggregation + Command segregation:

- 1 train speaker per client + 1 command per client:
    - FedAvg ('20cmd', 20 clients (one for each speech command), 3000 rounds, 400 audio files) $\rightarrow$ Figure 6.19
- 5 train speaker per client + 1 commands per client:
    - FedAvg ('20cmd', 20 clients (one for each speech command), 4500 rounds, 700 audio files) $\rightarrow$ Figure 6.20
    - FedAvg ('20cmd', 73 clients, 4000 rounds, 3000 audio files) $\rightarrow$ Figure 6.21
- 5 train speaker per client + 10 commands per client
    - FedAvg ('20cmd', 73 clients, 1850 rounds, 25000 audio files) $\rightarrow$ Figure 6.22
- 10 train speaker per client + 1 commands per client
    - FedAvg ('20cmd', 20 clients (one for each speech command), 4500 rounds, 1100 audio files) $\rightarrow$ Figure 6.23

In the speaker segregation scenario, where clients have different sets of train speakers, the simulations showed no clear difference in performance between FedAvg and FedProx. We believe that more heterogeneity needs to be introduced in the simulations to observe a noticeable distinction between the two algorithms. All the simulation have converged, but incrementing the number of client adds more complexity to the FL system, needing more rounds to converge.

In the speaker segregation combined with command segregation scenario, where clients have both distinct train speakers and commands, some of the simulation have not converged using only FedAvg. Notably, simulations with a smaller quantity of data samples (below 25000 audio files) did not reach convergence. This sugests that achieving convergence with a limited amount of data is a challenge in FL systems. Future work could focus on addressing this challenge and developing strategies to effectively train models with smaller datasets in federated settings.

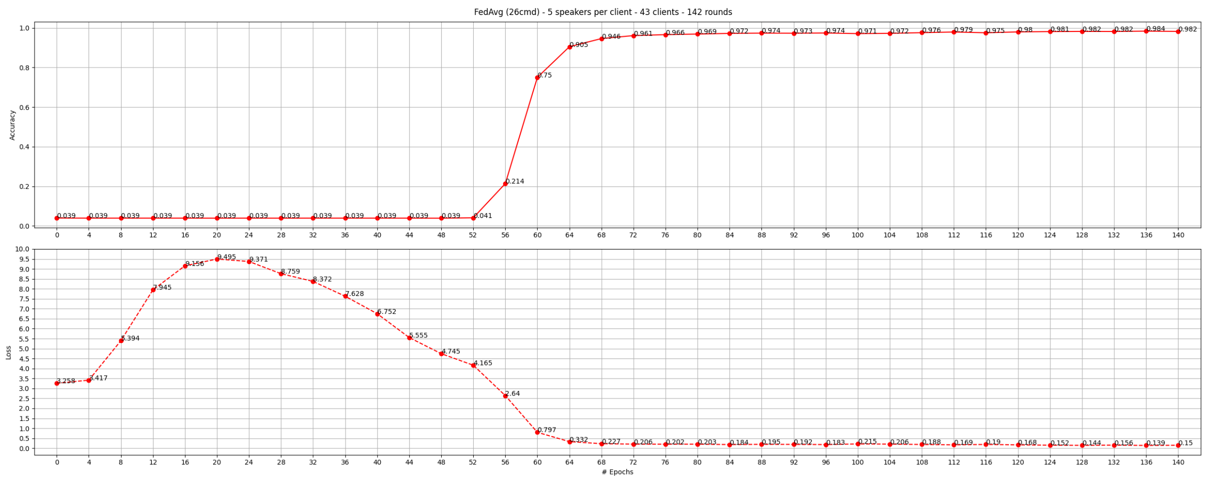


Figure 6.10. 5 train speakers per client: FedAvg ('26cmd', 43 clients, 142 rounds)

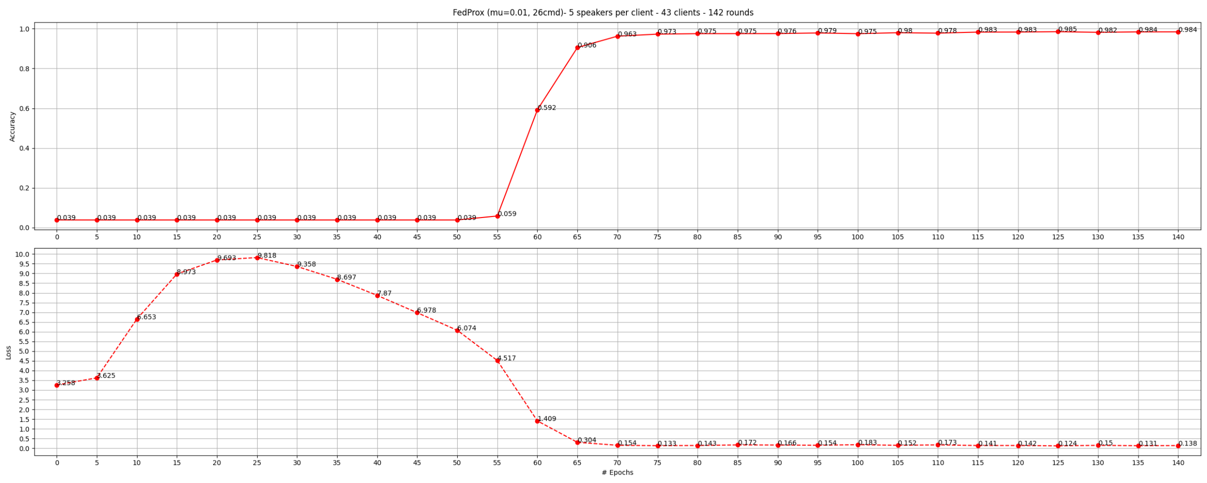


Figure 6.11. 5 train speakers per client: FedProx $\mu = 0.01$ ('26cmd', 43 clients, 142 rounds)

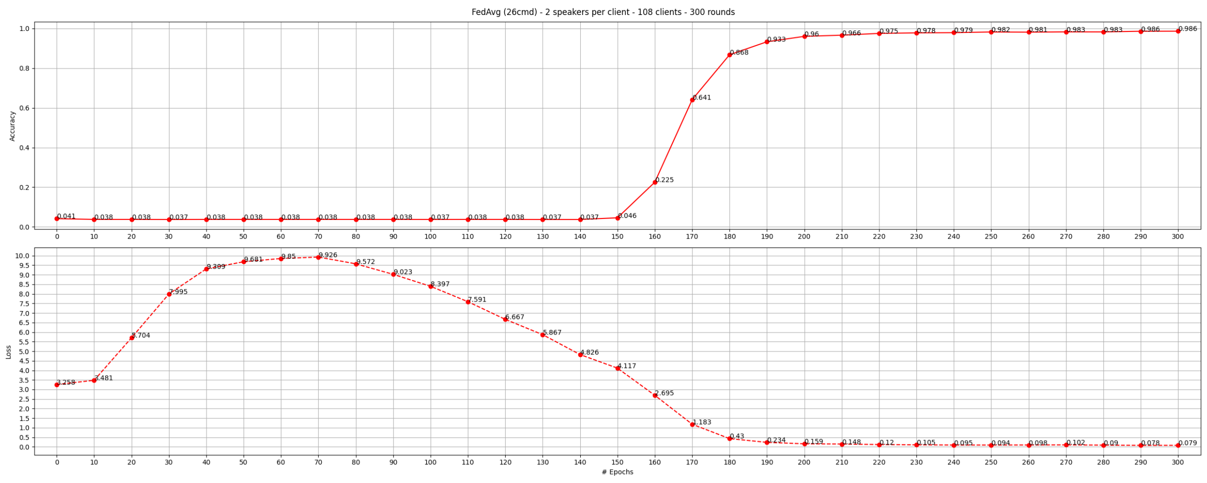


Figure 6.12. 2 train speakers per client: FedAvg ('26cmd', 108 clients, 300 rounds)

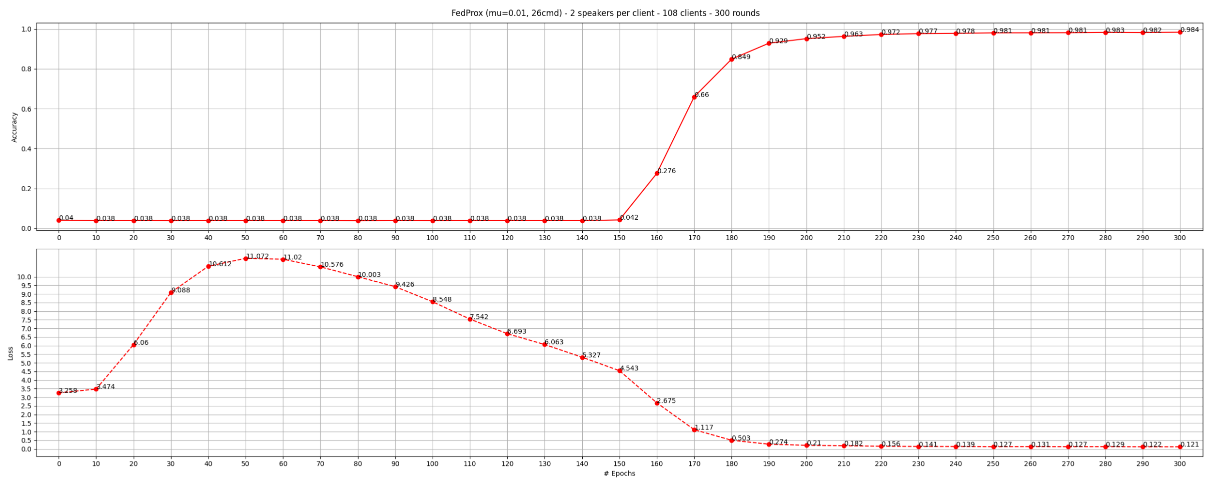


Figure 6.13. 2 train speakers per client: FedProx $\mu = 0.01$ ('26cmd', 108 clients, 300 rounds)

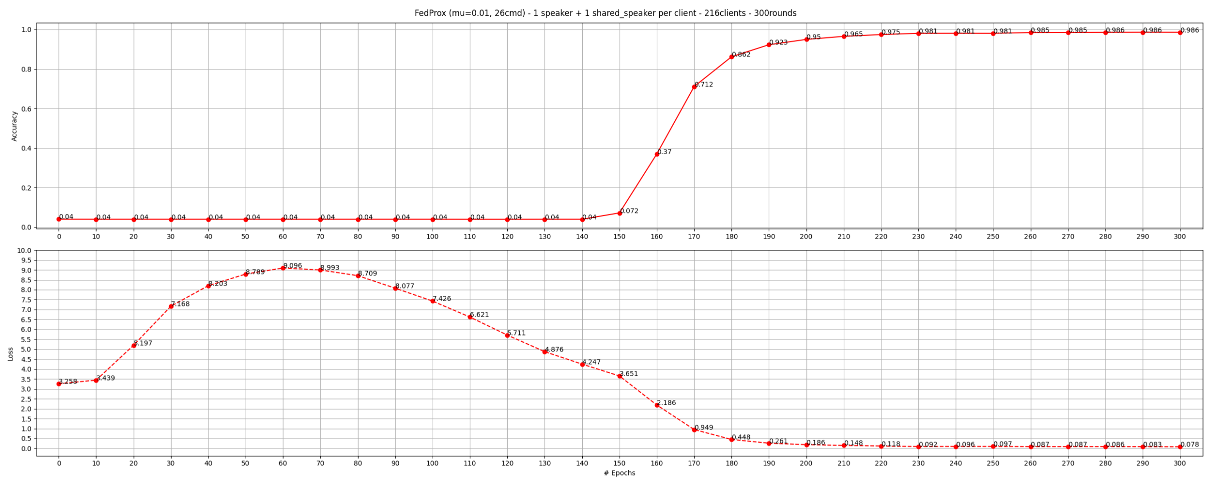


Figure 6.14. 1 train speaker + 1 shared speaker per client: FedProx $\mu = 0.01$ ('26cmd', 216 clients, 300 rounds)

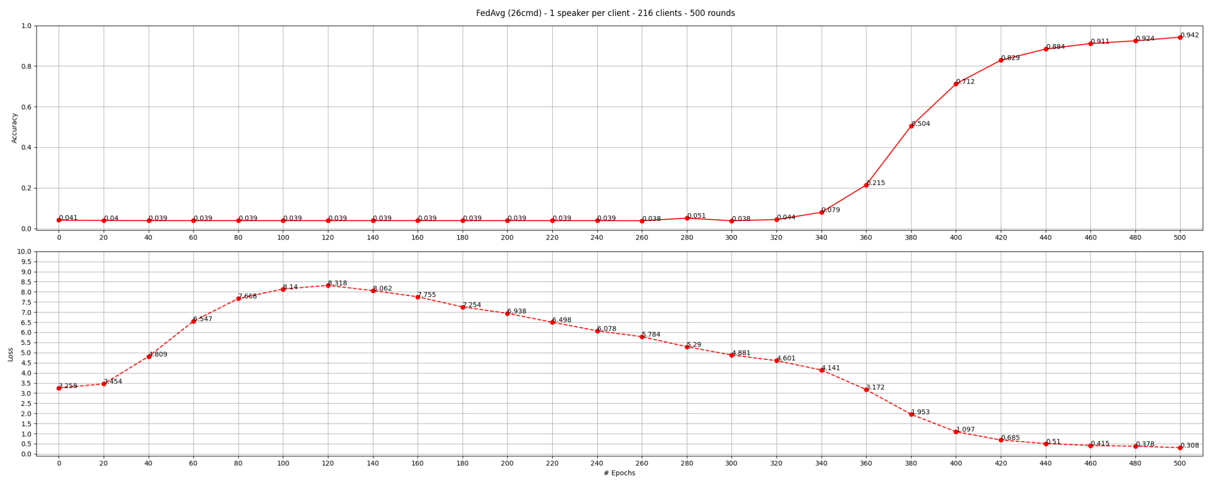


Figure 6.15. 1 train speaker per client: FedAvg ('26cmd', 216 clients, 500 rounds)

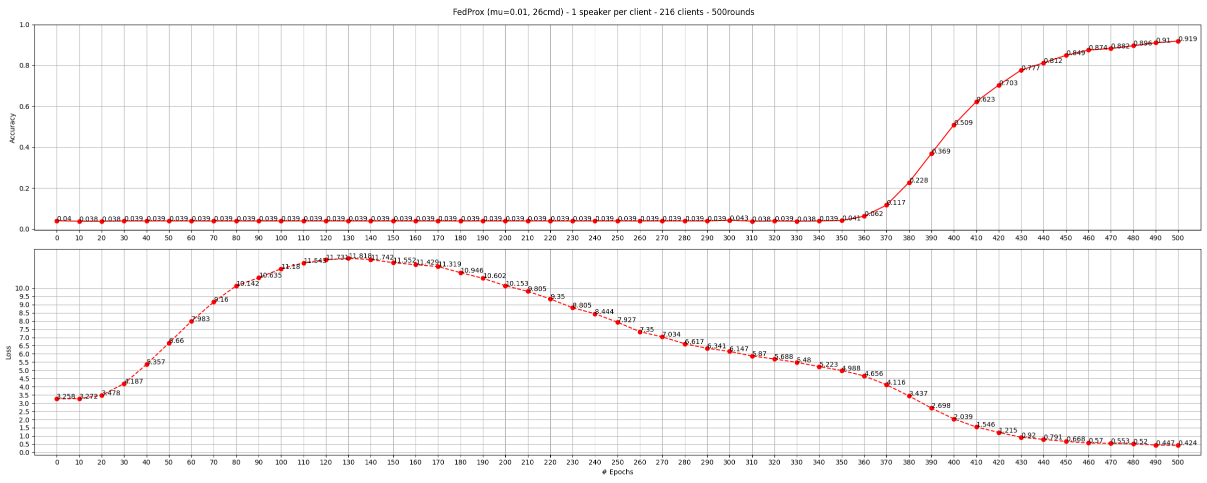


Figure 6.16. 1 train speaker per client: FedProx $\mu = 0.01$ ('26cmd', 216 clients, 500 rounds)

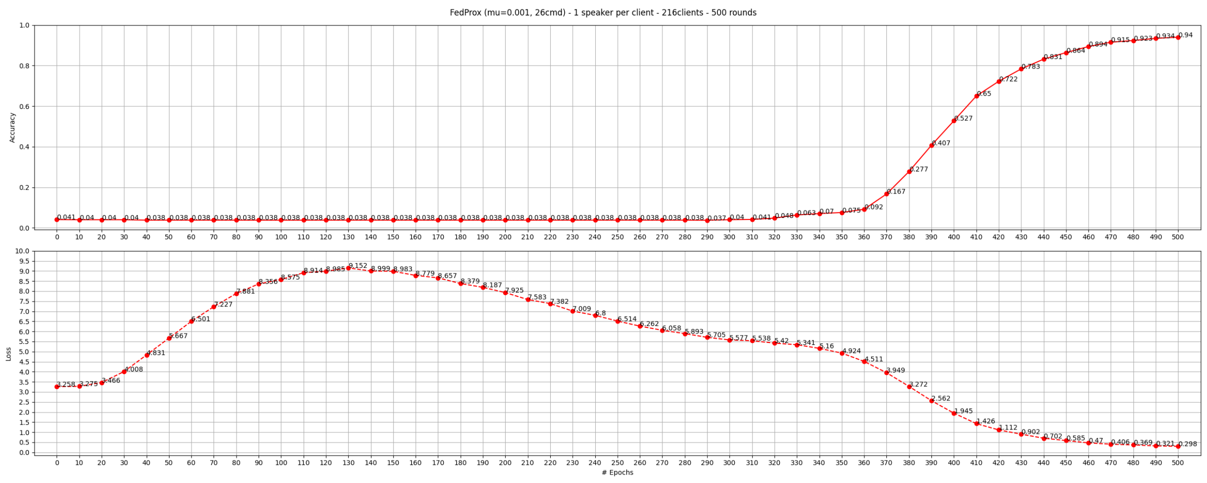


Figure 6.17. 1 train speaker per client: FedProx $\mu = 0.001$ ('26cmd', 216 clients, 500 rounds)

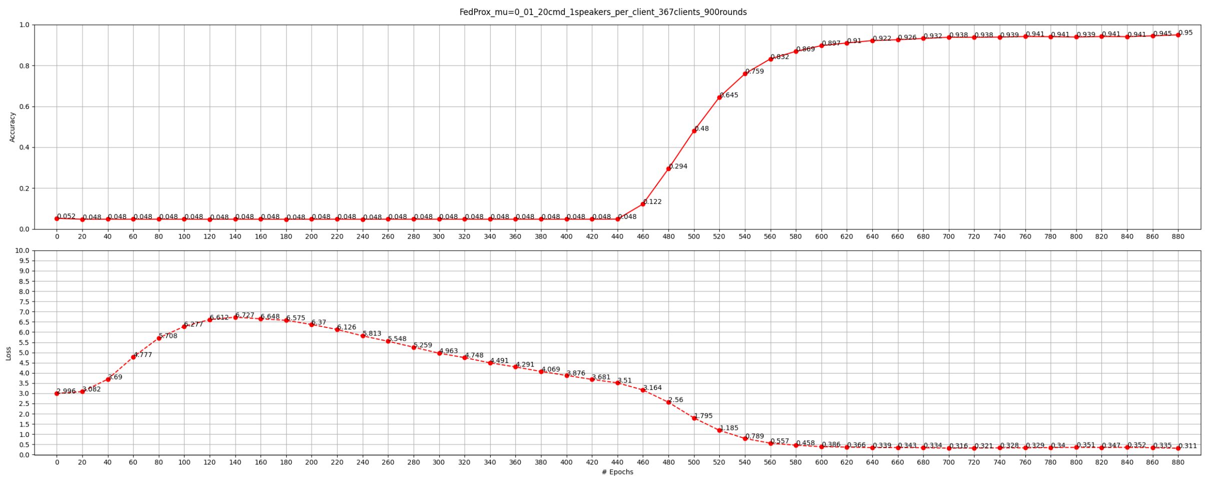


Figure 6.18. 1 train speaker per client: FedProx $\mu = 0.01$ ('20cmd', 367 clients, 900 rounds)

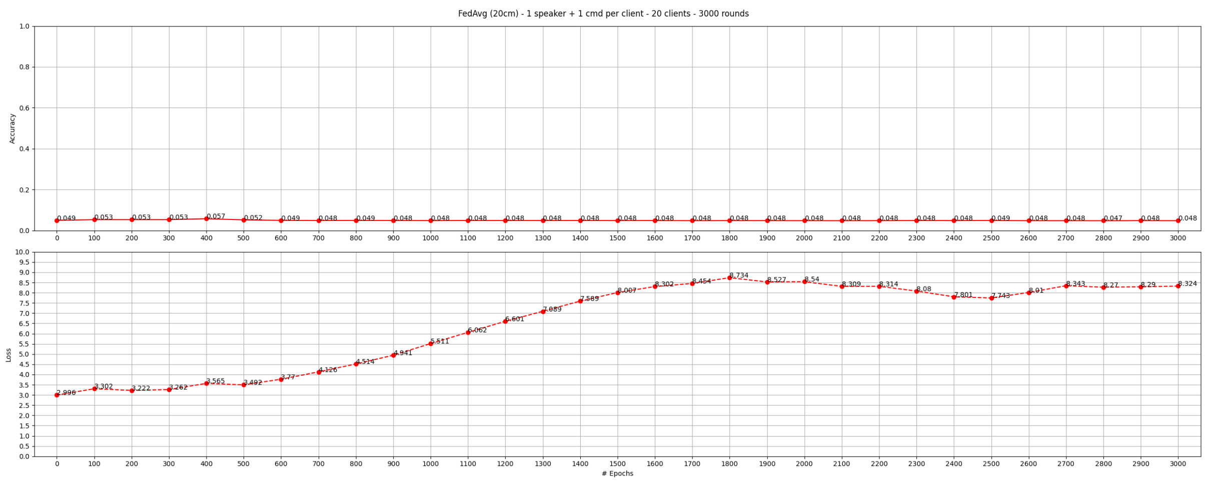


Figure 6.19. 1 train speaker per client + 1 command per client: FedAvg ('20cmd', 20 clients (one for each speech command), 3000 rounds)

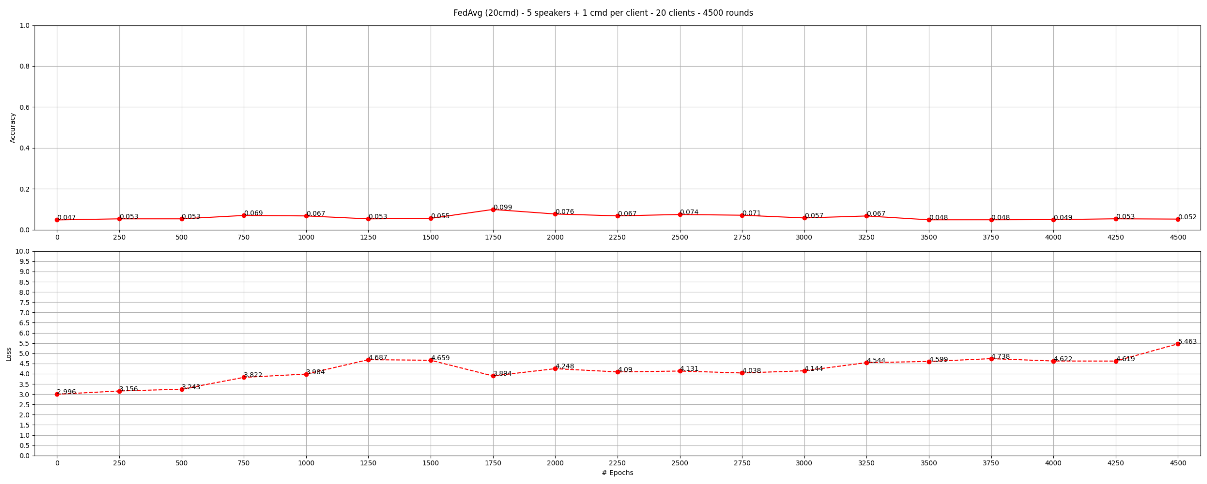


Figure 6.20. 5 train speaker per client + 1 commands per client: FedAvg ('20cmd', 20 clients (one for each speech command), 4500 rounds)

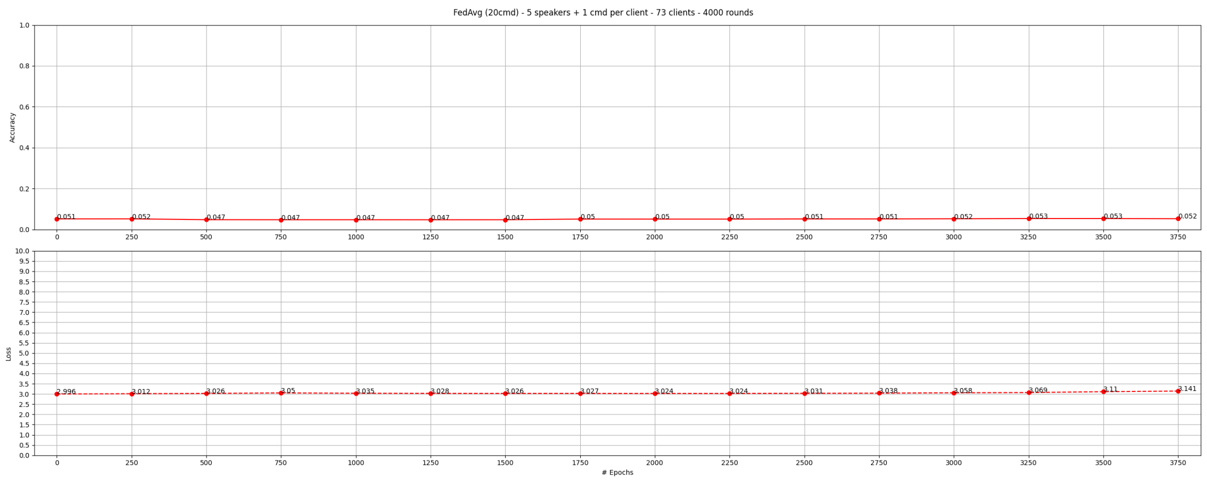


Figure 6.21. 5 train speaker per client + 1 commands per client: FedAvg ('20cmd', 73 clients, 4000 rounds)

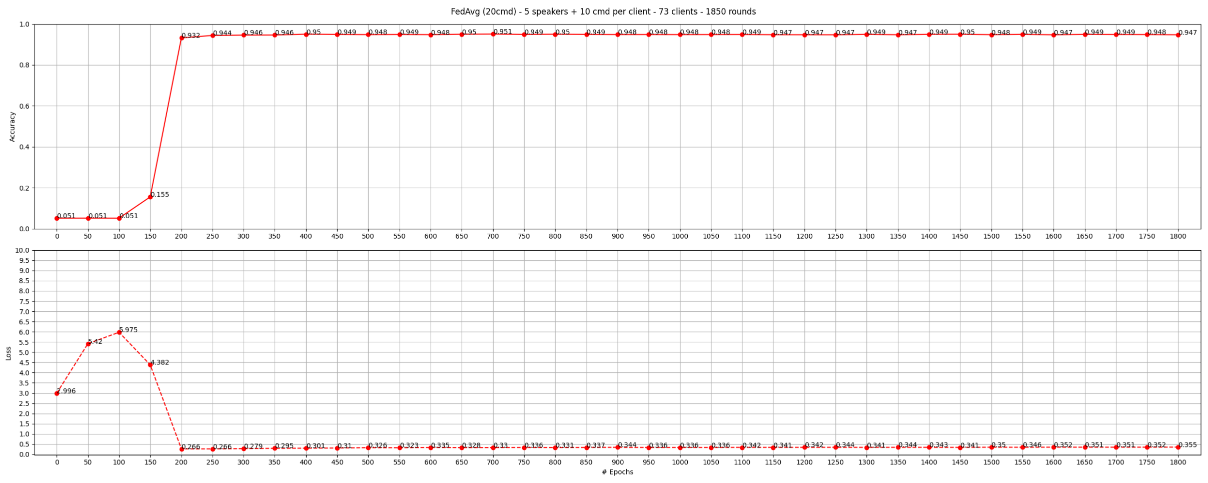


Figure 6.22. 5 train speaker per client + 10 commands per client: FedAvg ('20cmd', 73 clients, 1850 rounds)

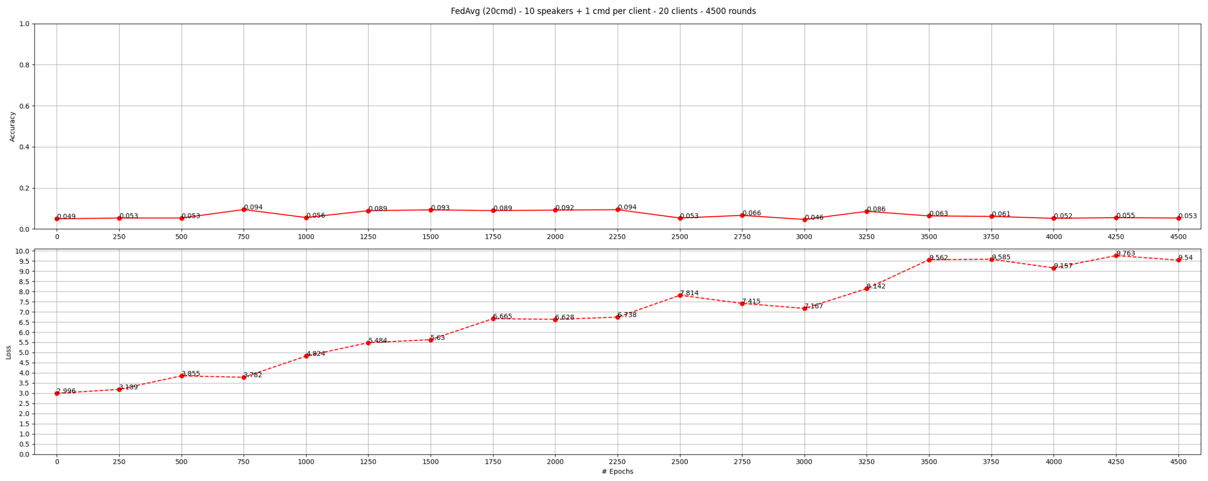


Figure 6.23. 10 train speaker per client + 1 commands per client: FedAvg ('20cmd', 20 clients (one for each speech command), 4500 rounds)

7 Conclusions

In conclusion, the evaluations conducted in this project have provided insights into the performance of federated learning (FL) algorithms in different scenarios (real network and simulated).

In the real network scenario, we build an FL system in separated devices in a real-world context. We achieve to connect our system through a 5G network and perform evaluations with successful results. Moreover, the added optimizations in Flower default strategy provide improvements in FL round time, speeding up the convergence of the model.

In the simulated scenario, we have tested the performance of FL setup with non-IID data. In speaker segregation setup, where clients have different sets of train speakers, no significant difference in performance was observed between FedAvg and FedProx. However, it was noted that introducing more heterogeneity in the simulations may reveal distinct advantages of one algorithm over the other. The simulations successfully converged, but the complexity of the FL system increased with a higher number of clients, requiring more rounds to achieve convergence.

Furthermore, in the speaker segregation combined with command segregation scenario, it was found that some simulations did not converge when using only FedAvg. Particularly, simulations with a smaller quantity of data samples, such as those below 25000 audio files, faced challenges in reaching convergence. This emphasizes the need to address convergence issues when working with limited datasets in FL settings, which could be explored in future research.

Overall, these findings contribute to the understanding of FL algorithms' performance under various conditions and highlight the importance of data heterogeneity, dataset size, and network connectivity in FL systems. Further investigations and optimizations can improve the efficiency and convergence of FL algorithms, enabling their practical implementation in real-world applications. The integration of FL with emerging technologies like 5G networks opens up exciting possibilities for distributed machine learning and collaborative data analysis.

8 References

[1] Flower a friendly federated learning framework, 2023.

[2] Openairinterface the fastest growing community and software assets in 5g wireless, 2023.

[3] Daniel J. Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Javier Fernandez-Marques, Yan Gao, Lorenzo Sani, Kwing Hei Li, Titouan Parcollet, Pedro Porto Buarque de Gusmão, and Nicholas D. Lane. Flower: A friendly federated learning research framework, 2020.

[4] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks, 2020.

[5] Knud Lasse Lueth. State of the IOT 2018: Number of IOT devices now at 7B – market accelerating, May 2022.

[6] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data, 2023.

[7] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks, 2019.

[8] P. Warden. Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. *ArXiv e-prints*, April 2018.

A Software Architecture Code Snippets

A.1 Data preprocessing and Models

```
model = tf.keras.applications.MobileNetV2(
    include_top = True,
    input_tensor = tf.keras.layers.Input(shape=get_input_shape()),
    weights = None,
    classes = len(get_labels()),
    classifier_activation = 'softmax'
)

model.compile(optimizer=tf.keras.optimizers.Adam(),
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
    metrics=['accuracy']
)
```

Code Snippet 17. MobileNetV2 load

A.2 SpeechGenerator.py

```
class SpeechGenerator(keras.utils.Sequence):
    def __init__(self, list_IDs, labels, batch_size=BATCH_SIZE,
     ↪  dim=get_input_shape(), shuffle=True):
        self.dim = dim  # Spectrogram size
        self.batch_size = batch_size
        self.labels = labels
        self.list_IDs = list_IDs
        self.shuffle = shuffle
        self.on_epoch_end()

    def __len__(self):
        """Denotes the number of batches per epoch"""
        return int(np.floor(len(self.list_IDs) / self.batch_size))

    def __getitem__(self, index):
        """ Generates one batch of data """
        indexes = self.indexes[index * self.batch_size:
         ↪  (index+1)*self.batch_size]
        list_IDs_temp = [self.list_IDs[k] for k in indexes]
        X, y = self.__data_generation(list_IDs_temp)

        return X, y

    def on_epoch_end(self):
        """Updates indexes after each epoch"""
        self.indexes = np.arange(len(self.list_IDs))
        if self.shuffle:
            np.random.shuffle(self.indexes)

    def __data_generation(self, list_IDs_temp):
        """Loads the data containing batch_size samples. X: (n_samples,
↪   *dim, n_channels) """
        X = np.empty((self.batch_size, *self.dim))
        y = np.empty((self.batch_size), dtype=int)

        for i, ID in enumerate(list_IDs_temp):
            spec = np.load(f'{DATASET_PATH}/{ID}')
            X[i,] = spec
            y[i] = self.labels[ID]

        return X, y
```

Code Snippet 18. SpeechGenerator class

A.3 Federated Strategy

```
@dataclass
class FitIns:
    """Fit instructions for a client."""

    parameters: Parameters
    config: Dict[str, Scalar]

@dataclass
class FitRes:
    """Fit response from a client."""

    status: Status
    parameters: Parameters
    num_examples: int
    metrics: Dict[str, Scalar]

class Strategy(ABC):
"""Abstract base class for server strategy implementations."""

    @abstractmethod
    def configure_fit(
        self,
        server_round: int,
        parameters: Parameters,
        client_manager: ClientManager
    ) -> List[Tuple[ClientProxy, FitIns]]:
        """Configure the next round of training."""

    @abstractmethod
    def aggregate_fit(
        self,
        server_round: int,
        results: List[Tuple[ClientProxy, FitRes]],
        failures: List[Union[Tuple[ClientProxy, FitRes], BaseException]],
    ) -> Tuple[Optional[Parameters], Dict[str, Scalar]]:
        """Aggregate training results."""
```

Code Snippet 19. Strategy abstraction

A.4 The FLAIR Manager

```
@dataclass
class FlairClient:
    """
    Represents a Flair client for managing the history and metrics.

    Attributes
    ----------
    cid : str
        Client ID.
    properties : Dict[str, Any]
        Client properties.
    metrics : Dict[int, Dict[str, Any]]
        Metrics associated with the client.
    """

    cid: str
    properties: Dict[str, Any]
    metrics: Dict[int, Dict[str, Any]]

@dataclass
class FlairManger:
    """
    Represents a Flair manager for managing the history and metrics of all
    flair clients.

    Attributes
    ----------
    experiment_path : Path
        The experiment path for saving all kinds of metrics and logs.
    clients : Dict[str, FlairClient]
        Contains all the registered flair clients.
    server_eval : Dict[int, Tuple[float, ...]]
        A metrics evaluation dictionary with the server round as a key and
↪   multiple
        metrics as the value.
    """

    experiment_path: Path
    clients: Dict[str, FlairClient]
    server_eval: Dict[int, Tuple[float, ...]]
```

Code Snippet 20. FLAIR manager and client

```
def aggregate_fit(
self,
server_round: int,
results: List[Tuple[ClientProxy, FitRes]],
failures: List[Union[Tuple[ClientProxy, FitRes], BaseException]],
) -> Tuple[Optional[Parameters], Dict[str, Scalar]]:

logger.info('[aggregate_fit] FlairManager updating and saving metrics')
for client_proxy, fit_res in results:
    flair_manager.update_metrics(
        client_cid=client_proxy.cid,
        server_round=server_round,
        metrics=fit_res.metrics
    )

return super().aggregate_fit(server_round, results, failures)
```

Code Snippet 21. FLAIR manager, example of usage