WEARABLE EDGE AI TOWARDS CYBER-PHYSICAL APPLICATIONS

Mateus Coelho Silva

Orientadores: Ricardo Augusto Rabelo Oliveira
Servio Pontes Ribeiro
Andrea Gomes Campos Bianchi

Ouro Preto
Janeiro de 2024

WEARABLE EDGE AI TOWARDS CYBER-PHYSICAL APPLICATIONS

Mateus Coelho Silva

Tese de Doutorado apresentada ao Programa de Pós-graduação em Ciência da Computação, da Universidade Federal de Ouro Preto, como parte dos requisitos necessários à obtenção do título de Doutor em Ciência da Computação.

Orientadores: Ricardo Augusto Rabelo Oliveira
Servio Pontes Ribeiro
Andrea Gomes Campos Bianchi

Ouro Preto
Janeiro de 2024

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

S586w Silva, Mateus Coelho.
Wearable Edge AI towards Cyber-Physical Applications. [manuscrito] / Mateus Coelho Silva. - 2023.
184 f.: il.: color., gráf., tab..

Orientador: Prof. Dr. Ricardo Augusto Rabelo Oliveira.
Coorientadores: Profa. Dra. Andrea Gomes Campos Bianchi, Prof. Dr. Servio Pontes Ribeiro.
Tese (Doutorado). Universidade Federal de Ouro Preto. Departamento de Computação. Programa de Pós-Graduação em Ciência da Computação.
Área de Concentração: Ciência da Computação.

1. Computação Vestível. 2. Edge AI. 3. Ecologia. 4. Cuidados com a Sáude. I. Oliveira, Ricardo Augusto Rabelo. II. Bianchi, Andrea Gomes Campos. III. Ribeiro, Servio Pontes. IV. Universidade Federal de Ouro Preto. V. Título.

CDU 004

Bibliotecário(a) Responsável: Luciana De Oliveira - SIAPE: 1.937.800

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E BIOLOGICAS
DEPARTAMENTO DE COMPUTACAO
PROGRAMA DE POS-GRADUACAO EM CIENCIA DA COMPUTACAO

FOLHA DE APROVAÇÃO

Mateus Coelho Silva

Wearable edge AI towards cyber-physical applications

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Ouro Preto como requisito parcial para obtenção do título de Doutor em Ciência da Computação

Aprovada em 01 de setembro de 2023

Membros da banca

Prof. Dr. Ricardo Augusto Rabelo Oliveira - Orientador - Universidade Federal de Ouro Preto
Prof. Dr. Fernando Augusto Teixeira - Universidade Federal de São João Del Rei
Prof. Dr. Jorge Miguel Sá Silva - Universidade de Coimbra
Prof. Dr. Luiz Henrique Andrade Correia - Universidade Federal de Lavras
Prof. Dr. Saul Emanuel Delabrida Silva - Universidade Federal de Ouro Preto
Dr. Vicente José Peixoto de Amorim - Dell Technologies

Prof. Dr. Ricardo Augusto Rabelo Oliveira, orientador do trabalho, aprovou a versão final e autorizou seu depósito no Repositório Institucional da UFOP em 13/12/2023

Documento assinado eletronicamente por **Ricardo Augusto Rabelo Oliveira**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 15/12/2023, às 11:47, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0642091** e o código CRC **3513E80B**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35402-163
Telefone: (31)3559-1641 - www.ufop.br

"The day we cease the exploration of the cosmos is the day we threaten the continuing of our species. In that bleak world, arms-bearing, resource-hungry people & nations would be prone to act on their low-contracted prejudices and would have seen the last gasp of human enlightenment... until the rise of a visionary new culture that once again embraces the cosmic perspective. A perspective in which we are one, fitting neither above nor below, but within."
Neil DeGrasse Tyson

Agradecimentos

Obrigado a todas as pessoas que estão do meu lado e seguraram minhas mãos. Primeiramente à minha família, por sempre me acompanhar ao longo da minha formação. Aos meus amigos, obrigado por estarem lá. À minha companheira, obrigado por tudo que você faz e fez, de graça e sem saber o quanto é importante.

Obrigado aos meus orientadores. De fato, posso dizer que tive orientadores durante toda pós-graduação. Fui desafiado constantemente a ser a melhor versão de mim. Obrigado por nunca ter deixado faltar essa fagulha que sempre se tornou incêndio.

Obrigado à equipe do IFMG Itabirito pela oportunidade profissional. Vocês me ajudaram a pavimentar mais uma etapa essencial no meu caminho de me tornar professor e pesquisador. Além disso, vocês me estenderam um tratamento humano e justo, que me ensinou como qualquer líder profissional deve ser.

Obrigado aos amigos do G6.1 da Universidade de Coimbra. Vocês foram bons amigos, companheiros fiéis de risadas, trabalhos e discussões. Me ajudaram a me sentir acolhido quando eu estava a muitos quilômetros de casa. Importante ressaltar que o laboratório era o mais latino-americano do continente europeu. Sobretudo, espero a visita de vocês em Ouro Preto para comer, beber e ser feliz por aqui também.

Obrigado às equipes dos laboratórios iMobilis, LEAF e XR4Good pela companhia, risadas, alegrias e angústias compartilhadas. Obrigado por se levantarem pra me ajudar quando eu precisei, e de confiar em mim em vezes que precisavam de ajuda. Obrigado por me ensinarem a ser mais profissional e mais humano, e por aguentar as minhas piadas sem graça.

Obrigado a todos que me ouviram quando eu precisava falar, obrigado a todos que se calaram quando eu precisava de silêncio. Obrigado a todos os abraços quando eu precisava de carinho, e todas as palavras que foram ditas quando eu precisava ouvir.

Obrigado pela estadia.

O autor gostaria de agradecer à CAPES, CNPq, a UFOP e a Universidade de Coimbra pelo fomento ao projeto de pesquisa apresentado. O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Resumo da Tese apresentada à UFOP como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

WEARABLE EDGE AI TOWARDS CYBER-PHYSICAL APPLICATIONS

Mateus Coelho Silva

Janeiro/2024

Orientadores: Ricardo Augusto Rabelo Oliveira
Servio Pontes Ribeiro
Andrea Gomes Campos Bianchi

Programa: Ciência da Computação

A aplicação de tecnologias em pesquisas e trabalhos em campo é impulsionada por tecnologias como a Internet das Coisas (IoT), *Edge Computing* e computação vestível. Nesse contexto, o uso de sistemas baseados em Inteligência Artificial (AI) é cada vez mais comum e uma tendência nos trabalhos mais recentes. Ambientes com pouca conectividade e dificuldade para transmissão de dados com baixa latência reforçam o uso de tecnologias de *Edge Computing* para tratamento de dados adquiridos. Contudo, não existe clareza na maneira como o uso de AI é transportado para a computação de borda em ambientes extremos, dada a complexidade dos demais requisitos. Essa lacuna é mais clara no contexto de dispositivos vestíveis (wearable), onde as restrições para desenvolvimento de sistemas são ainda mais complexas. Dessa forma, esse trabalho apresenta o protocolo de desenvolvimento e aplicações em casos de uso para criação de soluções baseadas em *Edge AI* no contexto de dispositivos vestíveis. Esse estudo ajuda a avaliar a criação do contexto de *Wearable Edge AI* como novo campo de pesquisa.

Abstract of Thesis presented to UFOP as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

WEARABLE EDGE AI TOWARDS CYBER-PHYSICAL APPLICATIONS

Mateus Coelho Silva

January/2024

Advisors: Ricardo Augusto Rabelo Oliveira
Servio Pontes Ribeiro
Andrea Gomes Campos Bianchi

Department: Computer Science

The creation of novel technologies to support field work and research has a major impact from technologies such as the Internet of Things (IoT), Edge Computing and wearable computing. In this context, Artificial-Intelligence-based systems became more common and a trend in recent work. Environments with low connectivity and high latency in data transmission enforce the usage of Edge Computing technologies in the treatment of acquired data. Nonetheless, there is no clarity in how to transport Artificial Intelligence (AI) to Edge Computing in extreme environments, given the complexity of the requirements. This gap is more clear in the context of wearable computing, where the systems restrictions for developing systems are even harder. Thus, this work presents a protocol for developing Edge AI appliances and some case-study applications in the context of wearable devices. This study helps to evaluate the creation of Wearable Edge AI context as a novel research field.

Contents

List of Figures

List of Tables

List of Abbreviations

AI Artificial Intelligence

ASL American Sign Language

CNN Convolutional Neural Network

CPS Cyber-Physical System

CPU Central Processing Unit

CWS Cooperative Wearable System

DNN Deep Neural Network

ECG Electrocardiography

EEG Electroencephalography

FPGA Field-Programmable Gate Array

GAN Generative Adversarial Network

GPE Gaussian Process Estimator

GRP Gaussian Random Process

HPS Hybrid Pelletized Sinter

HUD Head-Up Display

HW Hardware

IDS Intrusion Detection System

IMU Inertial Measurement Unit

IoT Internet of Things

LIDAR Light Detection and Ranging

LSTM Long Short-Term Memory

MCG Magnetocardiography

MEC Mobile Edge Computing

MLP Multi-Layer Perceptron

NLP Natural-Language Processing

PCB Printed Circuit Board

PPG Photoplesmiography

QoS Quality-of-Service

SVM Support Vector Machine

SW Software

UAV Unmanned Aerial Vehicles

VAE Variational Autoencoders

VANET Vehicular Ad Hoc Network

WBAN Wireless Body-Area Network

WLAN Wireless Local Area Network

WSN Wireless Sensor Networks

ZB Zettabytes

Chapter 1

Introduction

Cyber-Physical Systems (CPSs) are a nomenclature created to describe the applications that combine digital aspects with real-world applications. This integration happens through an integrated loop using sensors to add perception and often actuators to interact with the environment [7]. Some of the essential aspects that enable the development of such systems are hardware miniaturization and the increased number of devices with networking abilities [8].

Cyber-physical systems relate to some aspects of computer science, especially within the context of embedded computing and edge computing. Among these aspects, we highlight the following related context [8, 9]:

- Industry 4.0;
- Internet of Things (IoT);
- Big data;
- Cloud computing;
- Edge computing;
- Wearable devices;
- Mobile devices;
- Real-time systems;

Our understanding is that these concepts are linked through two fundamental features of cyber-physical systems. The first important concept we raise is the **context awareness**. Applications of CPSs must have means to perceive the context around them. Then, the second context is **environmental interaction**. These applications must be able to interact with the users, stakeholders, and the surrounding environment. Besides these main concepts, some of the features from CPSs are [10]:

- Closely integrated systems;
- Resource constrained;
- Networked;
- Executes in closed loops;
- Reliable and secure.

Modeling CPSs is a challenge [11]. Applications within this range must integrate system specifications, considering physical restraints, hardware and software integration, and the network interconnecting the devices. Also, software development in CPSs must consider its correctness and often timing constraints.

In this text, we will henceforth name the applications built within the CPS range as cyber-physical applications. Some examples of stakeholders from cyber-physical applications are smart cities, industries, aviation, environmental monitoring agencies and researchers, healthcare professionals and patients, and meteorologists [7–9, 12].

As stated before, wearable computing and edge computing are essential concepts within the range of CPSs. In this work, we explore how to use both these concepts together with artificial intelligence algorithms to create cyber-physical applications. In the following sections, we offer an initial individual perspective on each of these concepts before presenting the main objective of this thesis.

1.1 Wearable Computing

A relevant topic among CPSs is wearable computing, which creates several cyber-physical applications. Steve Mann and Thad Starner took some pioneer steps towards wearable computing in the late 80s and early 90s [13, 14]. Mann [13] enforced that a critical component in the creation of these systems was hardware miniaturization. Starner [14] had an early understanding of the issues of creating these applications, enforcing the difficulty in energy availability, weight, and size.

Since then, wearable computing has reached various stakeholders with various applications. Roggen et al. [15] pointed out several uses of wearable computing in robotics and automation, even with gesture and activity recognition. Jhajharia et al. [16] enforce the usage of such systems in healthcare and medical appliances, fitness, wellness, military, and industry.

Wearable computing shares some standard features among their applications [16]. Some of the main features of these systems are:

- **Reliabiliy:** Wearable computers and systems must be consistent and reliable;

- **Context awareness:** Wearable computing systems must add or enhance the user's environmental perception;
- **Ease of use:** Wearable systems must be easy to use, often not requiring any interaction to perform their duties;
- **Non-obstrusive and mobile:** Wearable systems must preserve the users' ability to move freely.

From this perspective, we notice that several of these features come from the broader field of CPSs. Therefore, within our context, we consider wearable computing systems to be examples of cyber-physical applications.

Wearable devices are trending topics in different areas such as healthcare and activity recognition [17–23], sports [24, 25], education [26, 27], industry [28, 29], human–computer interaction [30] and other areas. The rise of new concepts like the Internet of Things (IoT) [31–33] and Industry 4.0 [34] along with hardware miniaturization allow for the development of novel devices and solutions. Furthermore, the communication aspect of wearable systems is an essential aspect of novel developed solutions [35, 36].

1.2 Edge Computing

The following fundamental concept within this work is Edge Computing. This expression refers to using tools closer to the user's end and further away from the cloud. According to Varghese et al. [37], the motivations for this movement include decentralized computing, low latency, sustainable energy consumption, and smart computation techniques. These authors exemplify that the distance between Berkeley and Canberra causes a latency of 175 ms, which imposes an issue for latency-sensitive algorithms. Furthermore, the added network traffic can cause uncertainty.

Cao et al. [38] enforce that the expected global traffic on the internet in 2019 was 10.4 Zettabytes (ZB). They assert that 45% of this information would be stored, processed, and analyzed on the network's edge. By 2020, the number of connected devices was expected to surpass 50 billion. These authors also realized that cloud computing came short on some restraints, namely:

- Real-time;
- Energy consumption;
- Security;
- Privacy;

Both authors agree on the understanding that there are some issues that are not solved within the domain of cloud computing. Even with the 5G networks, these issues still need to be solved. El-Shorbagy [39] enforces that although 5G networks are faster and have low latencies in the ideal functioning scenario, there are issues as any physical obstacle blocks its signal and it has a lower range.

Khan et al. [2] also enforce that fast computing and quick application response are some of the features expected from edge computing, making it more suitable for real-time applications. He presents a division of edge computing among three main areas: *cloudlets*, *fog computing*, and *mobile edge computing*.

Cloudlets are infrastructures with high processing power that perform the role of cloud computing closer to the end-users. *Fog computing* is the virtualization of cloud-like services using a distributed computing environment through mobile devices. *Mobile edge computing* is the processing through isolated or restricted edge computing platforms providing processing power at the very edge of the network.

Khan et al. [2] also affirm that edge computing can have a dense geographical distribution, supports mobility, is closer to the end-user, enforces low latency solutions, and provides context-awareness. Finally, the authors display heterogeneity as an aspect of edge computing, given the diversity of devices involved in communication and processing.

Current technologies and environmental conditions still require storage, processing, and analyses at the network's edge. Edge computing is relevant in several cyber-physical applications, including cooperative wearable systems. In this study, this concept has a critical role in creating novel applications using wearable computing and AI techniques.

1.3 Artificial Intelligence (AI)

By the time this work was published, Artificial Intelligence had been a topic of research and development by scientists and engineers for almost 70 years. Jiang et al. [40] presented some projections for the near future of AI, stating that its market share is projected to reach over 190 billion US dollars by 2025. They exemplify some fields of acting from AI, such as:

- Speech recognition;
- Image processing;
- Natural-language processing (NLP);
- Smart robots;

- Autonomous vehicles;
- Healthcare;

Besides these, there are numerous fields in which AI has been applied. Jiang et al. [40] also assess some of the facts that enabled the uprise of AI. According to them, the increased number of theoretical proposals, the increasing amount of available data, the increasing computing power, and the outperformance of humans in some tasks of interest have contributed to an increasing interest over the years, starting from the 80s.

In a Q&A paper published in 2007, McCarthy [41] offers a simple definition of Artificial Intelligence. He states that AI is the science and engineering of making intelligent machines, especially intelligent software. This definition is a simple approach but requires a definition of intelligence. Still, according to McCarthy [41], intelligence is the ability to achieve goals in the world.

Shinde and Shah [1] present machine learning in computer science as the process of creating so-called "intelligent agents". In their perception, these agents are capable of perceiving the environment and acting to maximize its opportunity for success. Although artificial intelligence concepts came from the 1950s, the hardware and software advances allowed it to reach the desired potential later on. From the general concept of AI, these advances led to the uprise of machine learning and deep learning. Figure 1.1 displays the hierarchy of these concepts according to Shinde and Shah.

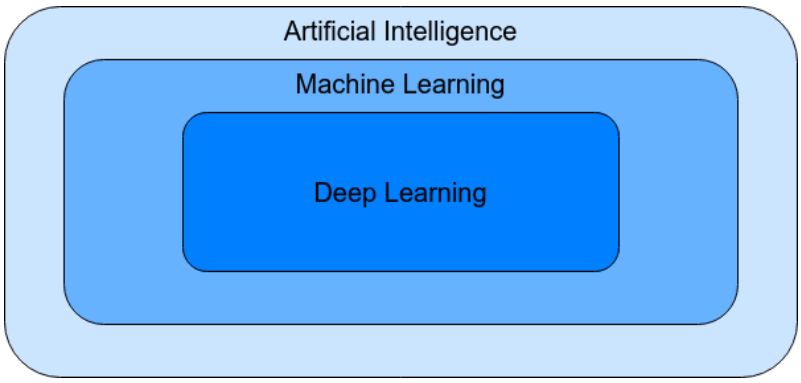


Figure 1.1: Machine Learning Hierarchy (inspired from Shinde and Shah [1])

Most AI applications follow the process described in the previous paragraph. The set of algorithms that follow this is called machine learning (ML) [42]. More recently, the most extensive set of algorithms developed is called deep learning (DL) [43, 44], a subset of machine learning algorithms with multiple processing layers to learn abstract data representations.

However, machine and deep learning often find limitations when integrating with edge computing. This condition happens especially when the computing power of

the edge application is constrained. Liu et al. [45] assert that although many research projects try to reduce this gap, machine, and deep learning often require high computational power and energy consumption. Sze et al. [46] also state that these applications require significant computational power, pushing applications toward the clouds.

1.4 Stakeholders

There are two main stakeholders to the technologies developed in this context. The first stakeholders are professors, students, practitioners, and ecology researchers. The technologies within this context were created aiming for canopy studies and entomology needs. These appliances are usually driven towards environmental perception. Figure 1.2 displays some applications developed for these stakeholders.

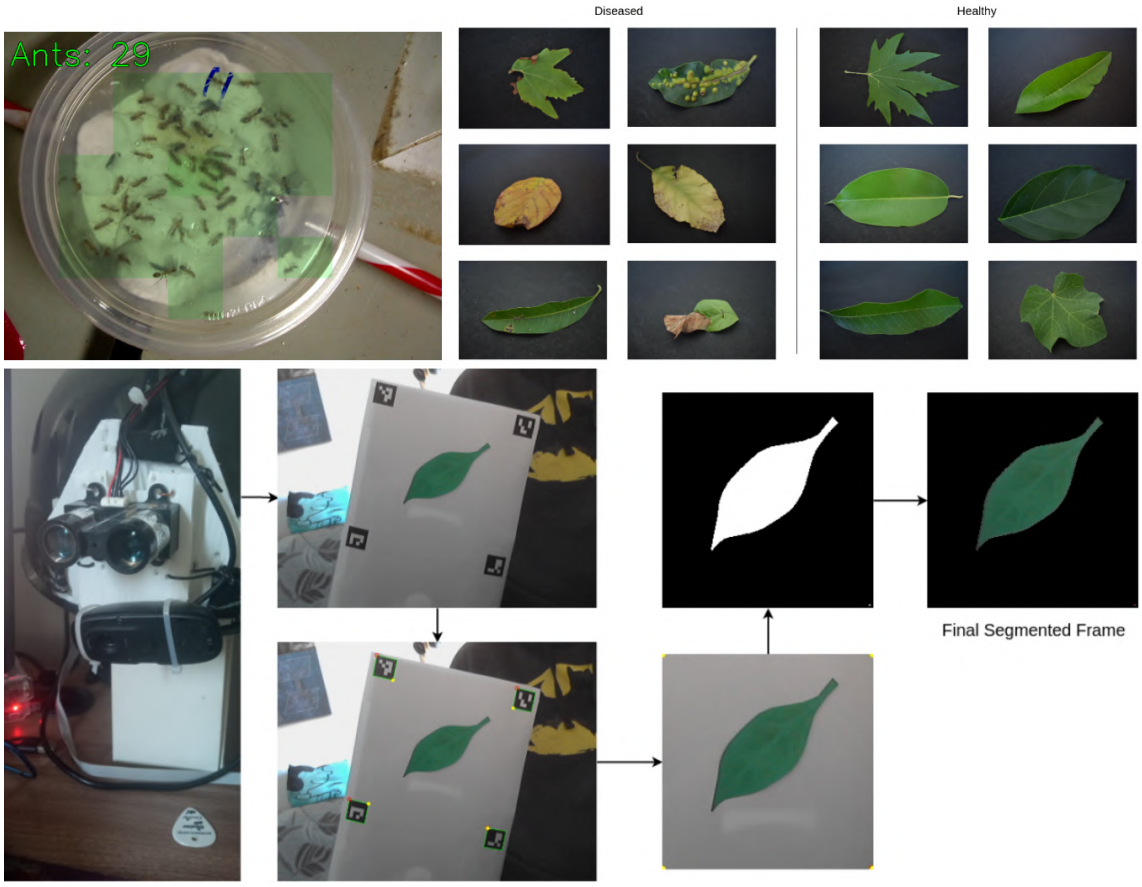


Figure 1.2: Examples of applications developed towards the first stakeholders. These solutions include automatic ant-counting using CNNs, leaf disease detection, and leaf shape estimation.

The second stakeholders are researchers, physiologists, medical practitioners, patients, and people who require health monitoring appliances. These technologies aim to monitor the users' health conditions and their activities. These applications

usually focus on perceiving the users' activities and conditions. Figure 1.3 displays some applications developed for the second group of stakeholders.

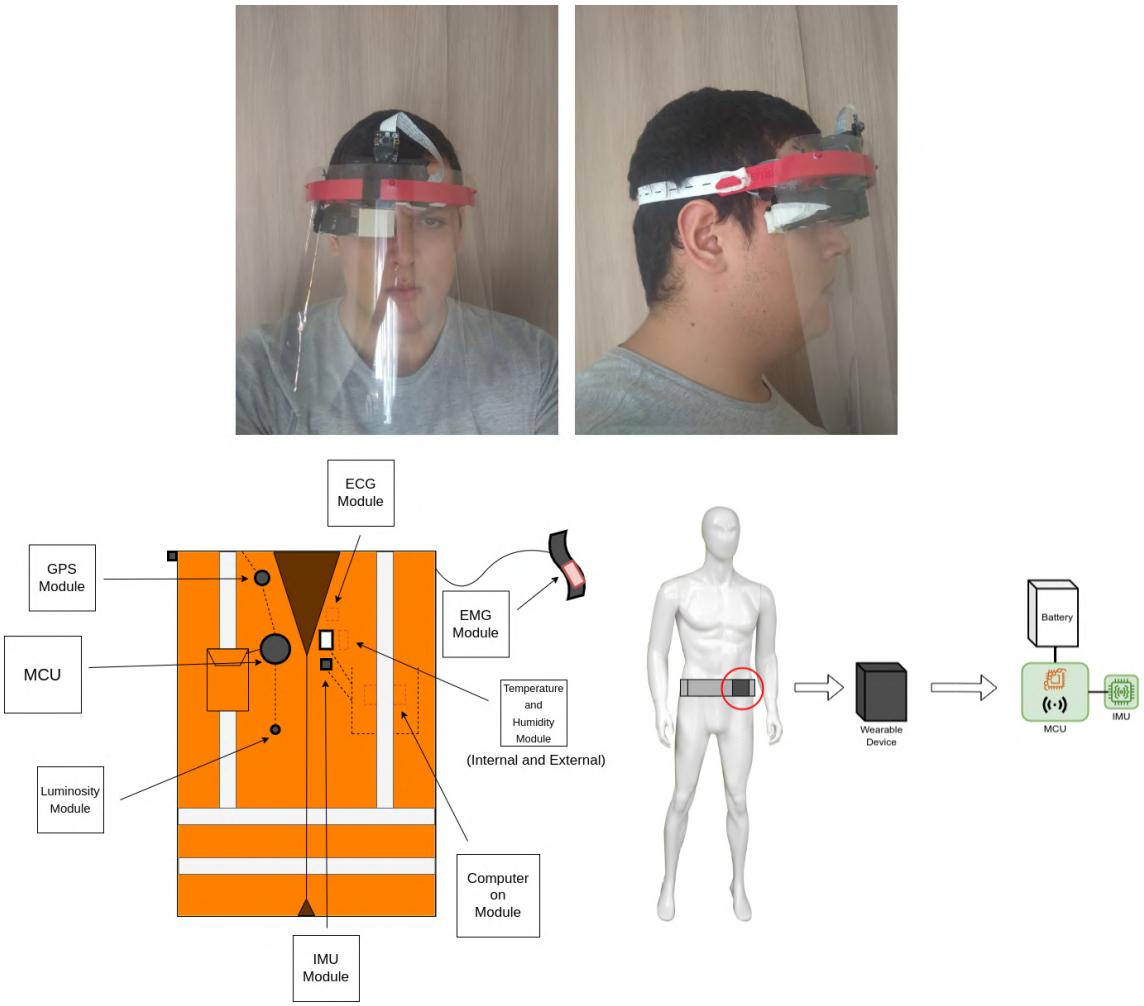


Figure 1.3: Examples of applications developed towards the second stakeholders. These solutions include health monitoring using a faceshield, a multi-sensored smart vest for industrial applications, and a human activity recognition (HAR) monitor.

1.5 Objectives

In this work, we focus on understanding how to mesh the concepts of edge computing, wearable computing, and AI to create a novel concept within Cyber-Physical Systems. A relatively novel concept named Edge AI works as a baseline for starting this establishment. Therefore, the main objective of this work is:

- Establish the concept of Wearable Edge AI and its design process to create cyber-physical applications.

We also contemplate a set of specific objectives withing this context. They are:

- Propose a novel taxonomy to describe the modalities of Edge AI;
- Develop a set of cyber-physical applications aiming ecological studies;
- Develop a set of cyber-physical applications aiming healthcare and human-activity recognition;

1.6 Text Organization

This text is organized into six chapters. In this first chapter, we introduce the subject of the study, motivation, stakeholders, objectives, and contributions. Chapter 2 reviews the Edge AI concept and establishes a taxonomy to describe it. Chapter 3 displays how we evolved the concept of Wearable Edge AI and defined its design patterns. Chapter 4 discusses the applications developed aiming at the ecology stakeholders. Chapter 5 discusses the applications developed aiming at healthcare stakeholders. Finally, we discuss the learned lessons, conclusions, and final remarks in Chapter 6.

1.7 Contributions

Within this work's context, we have published the following articles, with a total of 28 citations according to the Google Scholar metrics:

- An Automatic Ant Counting and Distribution Estimation System Using Convolutional Neural Networks [47];
- Designing a Multiple-User Wearable Edge AI system towards Human Activity Recognition (2022 XII Brazilian Symposium on Computing Systems Engineering (SBESC)) [48];
- Edge Computing Smart Healthcare Cooperative Architecture for COVID-19 Medical Facilities (IEEE Latin America Transactions 2022) [49];
  - 1 citation (Google Scholar);
- Bringing Deep Learning to the Fields and Forests: Leaf Reconstruction and Shape Estimation (SN Computer Science, 2022) [50];
- Wearable Edge AI applications for Ecological Environments (Sensors, 2021) [51];
  - 7 citations (Google Scholar);

- An Improved Deep Learning Application for Leaf Shape Reconstruction and Damage Estimation (Proceedings of the 23rd International Conference on Enterprise Information Systems - Volume 1: ICEIS, 2021) [5]
  - 4 citations (Google Scholar);
  - Best Student Paper Award in the Area of Artificial Intelligence and Decision Support Systems;
- Faceshield HUD: Extended Usage of Wearable Computing on the COVID-19 Frontline (Proceedings of the 23rd International Conference on Enterprise Information Systems - Volume 1: ICEIS, 2021) [52];
  - 1 citation (Google Scholar);
- IDiSSC: Edge-computing-based Intelligent Diagnosis Support System for Citrus Inspection (Proceedings of the 23rd International Conference on Enterprise Information Systems - Volume 1: ICEIS, 2021) [53];
  - 4 citations (Google Scholar);
- Leaf shape reconstruction and damage estimation using a U-net-based conditional GAN (Proceedings of the 36th Annual ACM Symposium on Applied Computing - SAC'21, 2021) [54];
- Constraints and Challenges in Designing Applications for Industry 4.0: A Functional Approach (Proceedings of the 22nd International Conference on Enterprise Information Systems - Volume 1: ICEIS, 2020) [55];
  - 1 citation (Google Scholar);
- Field Research Cooperative Wearable Systems: Challenges in Requirements, Design and Validation (Sensors, 2019) [4];
  - 10 citations (Google Scholar);

In this period, I have also collaborated with these works with relevant contributions to this topic's knowledge, with 20 more citations:

- Towards a mobile system with a new wearable device and an AI application for walking and running activities (Anais do L Seminário Integrado de Software e Hardware, 2023) [56];
- A Mobile Robot Based on Edge AI (Anais do L Seminário Integrado de Software e Hardware, 2023) [57];

- Towards Autonomous Mobile Inspection Robots Using Edge AI (Proceedings of the 25th International Conference on Enterprise Information Systems - Volume 1: ICEIS, 2023) [58];
- Towards a Novel Edge AI System for Particle Size Detection in Mineral Processing Plants (Proceedings of the 25th International Conference on Enterprise Information Systems - Volume 1: ICEIS, 2023) [59];
- Blockchain-Based Smart Contract and Edge AI Applied in a Multirobot System: An Approach (IEEE Robotics and Automation Magazine, 2023) [60];
- Using Mobile Edge AI to Detect and Map Diseases in Citrus Orchards (Sensors, 2023) [61];
  - 2 citations (Google Scholar);
- Towards novel smart wearable sensors to classify subject-specific human walking activities (Anais Estendidos do XII Simpósio Brasileiro de Engenharia de Sistemas Computacionais) [62];
- A novel intelligent mobile application using human-centered AR: A case study in orange inspection (Anais Estendidos do XXI Simpósio Brasileiro de Fatores Humanos em Sistemas Computacionais) [63];
  - 1 citation (Google Scholar);
- Enabling Digital Twins in Industry 4.0 [64];
  - 3 citations (Google Scholar);
- Edge Deep Learning Towards the Metallurgical Industry: Improving the Hybrid Pelletized Sinter (HPS) Process [65];
  - 1 citation (Google Scholar);
- Towards a novel wearable solution for citrus inspection using Edge AI (2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)) [66];
  - 1 citation (Google Scholar);
- Applying Edge AI towards Deep-learning-based Monocular Visual Odometry Model for Mobile Robotics (Proceedings of the 24th International Conference on Enterprise Information Systems - Volume 1: ICEIS, 2022) [67];
  - 2 citations (Google Scholar);

- Mask R-CNN Applied to Quasi-particle Segmentation from the Hybrid Pelletized Sinter (HPS) Process (Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2022) - Volume 4: VISAPP) [68];

- Deep Learning Approach at the Edge to Detect Iron Ore Type (Sensors, 2022) [69];

  - 1 citation (Google Scholar);

- Deep-Learning-Based Embedded ADAS System (Proceedings of the XI Brazilian Symposium on Computing Systems Engineering (SBESC)) [70];

- Deep-Learning-Based Visual Odometry Models for Mobile Robotics (Proceedings of the XI Brazilian Symposium on Computing Systems Engineering (SBESC)) [71];

  - 2 citations (Google Scholar);

- Synchronous and Asynchronous Requirements for Digital Twins Applications in Industry 4.0 (Proceedings of the 23rd International Conference on Enterprise Information Systems - Volume 1: ICEIS, 2021) [72];

  - 4 citations (Google Scholar);

- Edge Deep Learning Applied to Granulometric Analysis on Quasi-particles from the Hybrid Pelletized Sinter (HPS) Process (Proceedings of the 23rd International Conference on Enterprise Information Systems - Volume 1: ICEIS, 2021) [73];

  - 3 citations (Google Scholar);

We have also deposited a patent request, with its text already in public domain:

- ESCUDO FACIAL PARA PROTEÇÃO CONTRA O COVID-19 E DOENÇAS INFECCIOSAS COM MONITOR PORTÁTIL PARA MONITORAMENTO DA SAÚDE DO USUÁRIO E VARIÁVEIS DO AMBIENTE. 2022, Brazil. Registration number: BR1020220014477, INPI - Instituto Nacional da Propriedade Industrial, Brazil.

Our total contributions add up to 29 research papers and a deposited patent request, with 48 citations up to this date. We also have two more papers invited to become book chapters.

Chapter 2

Edge AI

Edge computing has gained several applications in academic, industrial, and commercial media. The fundamental applications of this concept have a vast reach, standing from mobile edge applications to Cloudlets [2]. On the one hand, this broad concept creates a large set of possible novel technologies. On the other hand, this vastness requires an organization of the surrounding concepts.

Several novel knowledge areas contribute to the uprise of edge computing. Khan et al. [2] enforce that some of these areas are the Internet of Things (IoT), 5G, and augmented reality, among others. They also present a set of several application areas of edge computing, including industrial automation, traffic management, and intelligent monitoring of several environments.

Shi and Dustdar [3] suggest that the IoT success has significantly impacted the need for solutions on edge. They discuss that the data processing power of cloud computing is not the only factor to consider when discussing the availability of processing services. Creators should consider that despite the computing power provided by cloud services, some other variables impact the need for edge computing, such as bandwidth and response time.

Satyanarayanan [74] also initially assesses the importance of the Internet of Things in the rise of edge computing. The author proposes that the IoT pushed the computing paradigm towards decentralization. The hardware miniaturization contributes to the increased processing power on edge devices. This author reinforces the importance of physical proximity in this context, which is a crucial value when arguing in favor of edge computing appliances.

These discussions show that edge computing is essential in academic and commercial matters. Also, this area is somehow organized, displaying a set of applications in which performing computing on edge is relevant. This text explores how edge computing evolved with artificial intelligence (AI) applications.

McCarthy [41] described artificial intelligence as the science and engineering to create intelligent machines. In the author's understanding, this intelligence is

described as the ability to learn how to achieve goals in the world through an algorithm. The usage of intelligence on edge pursuits to enforce the latest techniques to edge computing scenarios. In the latest years, the most significant advances and breakthroughs are in deep learning, raising a particular interest in performing its integration among different edge computing variations [75].

Before entering the realm of what we will define as Edge AI, it is necessary to choose an edge computing organization as a foundation. In this text, we present a classification similar to Khan et al. [2], which divides edge computing into three main groups:

- **Cloudlets**: In this paradigm, a resourceful infrastructure works similarly to the cloud but are physically near to the end-user;
- **Fog computing**: This paradigm represents virtualization of cloud services to distributed heterogeneous devices closer to the edge of the network;
- **Mobile edge computing**: This perspective considers isolated edge computing networks and services. The local resources and network should provide all the infrastructure.

From these three perspectives and concepts, we use these main groups as foundations to describe what Edge AI is and its features. For this matter, Section 2.1 will organize and provide a definition for Edge AI, and Section 2.2 will provide an overview of Edge AI applications found in the literature. In Section 2.3, we discuss and analyze the main taxonomies for both edge computing and AI, presenting how they contribute to the definitions of Edge Computing assessed above. Finally, we present our proposed taxonomy in Section 2.4, discussing the final aspects and conclusions in Section 2.5.

2.1 Defining Edge AI

In the first section of this text, we presented the relevance of integrating edge computing and AI algorithms. With such an importance, different authors discuss this integration with several names. Some of the names presented in the literature for this confluence are:

- Edge Intelligence [75–77];
- In-Edge AI [78];
- Edge AI [79, 80];

- AI on Edge [81];

While other authors discuss the convergence of edge computing and AI without a known name for this phenomenon. Regardless of the name, even in more general discussions, authors usually discuss these concepts within a limited scope of Intelligence algorithms or discuss an aspect of Edge AI/Edge Intelligence.

For instance, Shi et al. [3] discuss the communication aspect of Edge AI. Liu et al. [77] also discuss the networking and communication aspects. Li et al. [79] discuss deep learning inference acceleration through edge computing. Zhou et al. [76] present a survey based on deep learning applications in edge computing. Deng et al. [75] display an evaluation of the confluence of edge computing and deep learning. Wang et al. [78] discuss the convergence of edge computing and AI through federated learning. Although it is natural to overview the most trending AI technologies, the area requires a formal definition that applies to the confluence of artificial intelligence and edge computing regardless of the AI technology.

Concerning the classification of Edge AI devices, Shi et al. [3] separate the edge computing elements among two classes: edge devices and edge servers. Then, they discuss the communication-awareness of authors' proposals that developed algorithms and systems based on Edge AI. Their classification can be correlated to the one presented by Khan et al. [2]. Mobile edge computing can belong to both edge devices and edge server nodes. Fog computing and cloudlets belong only to the edge server class. Figure 2.1 displays a Venn diagram displaying these relationships. The union of these definitions displays a more general classification regarding the works of both authors.

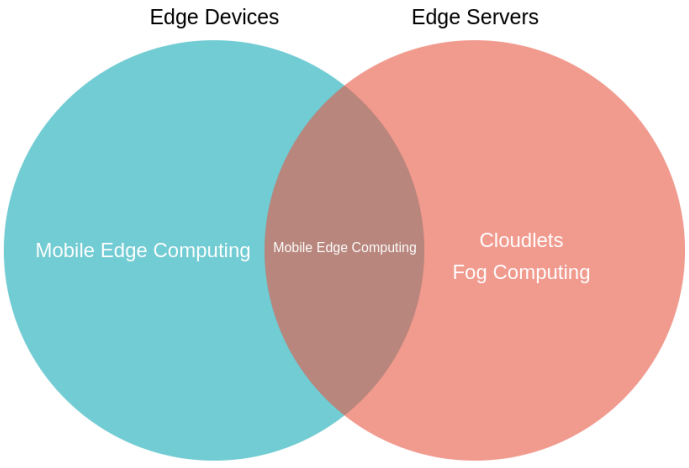


Figure 2.1: Correspondence to the classifications of edge computing devices and systems from Khan et al. [2] and Shi et al. [3]

Regarding a formal Edge AI definition, none of the authors present a formal

definition of Edge AI. As much as the definition seems evident at this point, there is no formal sentence defining the concept. For this matter, we establish an entry for the concept of Edge AI.

- **Edge AI Concept:** Edge AI is the set of methods that describes the design process and validation of solutions that combine edge computing and machine learning concepts in developing novel appliances, systems, and applications to solve real-world problems.

This definition bases itself on the main common aspects of all evaluated works. It is important to mention that this definition encompasses the other terminologies found in the literature, such as Edge Intelligence, and In-Edge AI, among others. From previous analyses, we also state some of the main features of Edge AI, which are:

- Regarding the **devices' role** in edge computing appliances, the devices can be *edge servers* or *edge devices*;
- Regarding the **applications' organization**, an application can be classified among *mobile edge computing*, *fog computing*, and *cloudlets*.

We use Edge AI as the official terminology in this work. Nonetheless, it is essential to clarify that this concept and features also refer to works that consider Edge Intelligence or other terminologies, as this is a general terminology. Finally, these terms work regardless of the technologies applied, as long as they can be classified as edge computing and AI together.

2.2 Mapping Edge AI applications

After defining the Edge AI, we also provide a mapping of the applications based on this concept. We initially assess and discuss some of the main works regarding Edge AI applications, their organization, and algorithms. Then, we provide some insights on classifying these systems according to the existing organizations and their contexts.

To perform this evaluation, we searched through the bibliography for papers describing applications based on Edge AI. For this matter, we used four different research strings to perform this search, performed on Google Scholar:

- "Edge AI";
- "Edge Intelligence";

- "Edge Computing Intelligent System";
- "Edge AI system";

We initially curated 70 articles describing appliances based on edge computing and AI algorithms. The criteria to select papers was that they should describe one or more case studies that are direct applications of Edge AI. After reviewing this selection, we remained with 54 papers. We excluded those that described general-use frameworks or did not describe specific use cases.

For evaluation purposes, we decided to provide a set of variables to be evaluated considering the edge computing and AI classifications. As discussed in the previous sections, edge computing systems can be classified into three classes according to their organization paradigm: cloudlets, fog computing, or mobile edge computing. Also, the devices running intelligent models can be classified into two main categories: edge devices and edge servers. For the AI, we will initially provide which algorithm or paradigm was used in the appliance. We will also briefly describe the process described in the paper.

2.2.1 Applications Overview

This subsection provides an overview of the applications. Each paragraph describes briefly a single work found in the literature. For each work, we evaluate the system's organization pattern (Mobile Edge, Fog, or Cloudlets), the AI paradigm applied in the solution, and whether the intelligence algorithms run on separate edge servers or edge nodes. The selection and classification criteria are described at the beginning of this section.

The text *iRobot-Factory: An intelligent robot factory based on cognitive manufacturing and edge computing [82]* describes an architecture of a multi-robot factory based on Edge AI. In this case, the edge devices are the robots and sensors involved in the production process. The intelligence algorithms are based on edge servers through direct connection or emulating cloud services. Their application targets the creation of intelligent and integrated industrial architectures.

The work *AI-enabled emotion-aware robot: The fusion of intelligent clothing, edge clouds, and robotics [83]* provides a proposal for an application integrating a robot and smart clothing to provide an emotion-aware system to mitigate mental health issues. The involved edge devices, in this case, were wearable devices and robots. Edge server nodes run a system called an emotion-cognition engine to provide AI-based actuation.

The authors of *An edge AI-enabled IoT healthcare monitoring system for smart cities [84]* assess the possibility of creating a healthcare solution based on edge

computing, AI, and the IoT. The edge devices involved in this proposal are personal end-users smart devices, patient monitoring devices, and connected ambulances. Intelligent algorithms run on edge servers based in cloudlets.

In *Energy efficient for UAV-enabled mobile edge computing networks: Intelligent task prediction and offloading [85]*, the authors provide a network-based application for unmanned aerial vehicles (UAVs) for intelligent tasks. In this work, the edge devices are UAVs, smart devices, and edge servers. The intelligent inferences occur in Edge Servers throughout a mobile edge computing organization.

The work entitled *Real-time strawberry detection using deep neural networks on embedded system (rtsd-net): An edge AI application [86]* provides a case study in which the authors apply computer vision and deep neural networks to detect strawberries in real-time. The edge devices involved in this perspective are UAVs and an edge server node. Intelligence algorithms run on the edge server node, provided by the Jetson Nano development board.

In the work named *Edge computing and artificial intelligence for real-time poultry monitoring [87]*, the authors propose and prototype an application for real-time poultry monitoring. The intelligent algorithm runs on a mobile edge computing server based on a Jetson Nano development board. The authors used ESP32 microcontrollers as the media for sensing the environment.

The text named *Edge AI-IoT pivot irrigation, plant diseases, and pests identification [88]* describes the usage of Edge AI within an IoT perspective to detect pests and diseases in plants. They also monitor environmental variables to detect anomalies. In their case, the edge devices are weather stations, remote sensors, and image capturing devices. The edge server runs several intelligent processes to provide insights into understanding environmental health.

The paper *Edge computing and artificial intelligence for landslides monitoring [89]* discusses the possibility of using Wireless Sensor Networks (WSNs), the IoT, and artificial intelligence to monitor landslides. The edge devices involved in the proposed architecture are multiple sensing nodes. The edge servers, in this case, are ODroid N2s and Jetson Nano boards, which run the intelligence algorithms.

In the article *Edge AI in smart farming IoT: CNNs at the edge and fog computing with LoRa [90]* the authors propose an appliance using edge and fog computing to provide Edge AI for smart farming. The edge devices are power-efficient IoT sensors that communicate through LPWAN networks. The intelligence applications are distributed among the fog and cloud layers.

In the paper *Edge Intelligence-Assisted Smoke Detection in Foggy Surveillance Environments [91]*, the authors discuss the training and deployment of an Edge-AI-based system to detect smoke in foggy surveillance environments. The edge intelligence proposed models are convolutional neural networks (CNNs) that classify

an image according to the presence of smoke. They search for models with lesser operations per second and size to enable the deployment of edge devices.

In the work named *Deep-Learning-Based Visual Odometry Models for Mobile Robotics [70]*, the authors propose the usage of CNNs and data from a simultaneous localization and mapping (SLAM) data to generate visual odometry models. The edge devices were robots with different capabilities: one can perform slam, while the other relies on visual odometry. The intelligence algorithm runs directly on the edge device.

The article *IDiSSC: Edge-computing-based Intelligent Diagnosis Support System for Citrus Inspection [53]* presents a proposal of an algorithm to perform the inspection of citrus fruits using Edge AI. The edge devices are image-capturing gear. The authors assess the possibility of using both edge devices or edge server hardware in the disease detection process.

In *Conveyor Belt Longitudinal Rip Detection Implementation with Edge AI [92]*, the authors assess the proposal of a novel conveyor belt rip detection system using edge AI for the mining industry. The edge devices, in this case, are image-capturing apparatuses. The authors' study considers the possibility of running the intelligence algorithms on both edge devices and edge servers.

The paper *Edge Deep Learning Applied to Granulometric Analysis on Quasi-particles from the Hybrid Pelletized Sinter (HPS) Process [73]* discusses a novel proposal for an edge-computing- and AI-based system to detect Quasi-particles in a steel industry process. The edge device involved in this proposal is an image capturing device, which also bears processing capabilities. The intelligence algorithms are CNNs running on the edge device.

In the work called *Investigating the Spread of Coronavirus Disease via Edge-AI and Air Pollution Correlation [93]*, the authors present an application based on information from an Edge AI system and Air Pollution data to investigate the spread of coronavirus disease. In this case, the authors evaluate the information flow from the cloud to the edge to perform predictions.

The paper *A Lossless Data-Hiding based IoT Data Authenticity Model in Edge-AI for Connected Living [94]* discusses the usage of Edge AI to verify and validate smart living IoT data before passing it through analytics. In this case, the edge devices are local IoT sensor nodes, mainly monitoring the user's health. The intelligent verification algorithm runs on an Edge AI server node.

The article *Edge Computing AI-IoT Integrated Energy-Efficient Intelligent Transportation System for Smart Cities [95]* displays a distributed multi-agent system based on edge computing, AI, and IoT for efficient and intelligent transportation. The intelligence systems run on cloudlets. Edge devices in this network are onboard devices gathering sensor data from transporting agents.

In *National Sports AI Health Management Service System Based on Edge Computing [96]*, the authors propose an AI sports health management system based on edge computing. Edge devices in this context are smart sensors and intelligent health systems. Intelligence algorithms run in cloudlet layers of the application.

The authors of *Towards AI-Based Traffic Counting System with Edge Computing [97]* discuss implementing an AI-based traffic counting system based on the intelligent transportation system concept. The edge devices in this context are cameras capturing live traffic images attached to AI-capable hardware. The edge devices run the intelligence models and feed a server with information.

Edge AI-Based Automated Detection and Classification of Road Anomalies in VANET Using Deep Learning [98] discusses the application of DNNs and edge computing to create an intelligent environment for road anomalies detection and classification. Edge devices, in this case, are image-capturing instruments distributed among the network. Edge servers integrate the Vehicular Ad Hoc Network (VANET), running intelligence algorithms.

The work named *Intelligent and Smart Irrigation System Using Edge Computing and IoT [99]* proposes the intelligent usage of ontology and sensor data together with a machine learning classification algorithm to create a smart irrigation system based on Edge AI. Edge devices are smart sensors capturing field data. Intelligent algorithms run on the edge server layer.

The paper entitled *AI at the Edge for Sign Language Learning Support [100]* suggests a novel method to recognize the American Sign Language (ASL) alphabet using CNNs and Fog Computing. Edge devices in this architecture are smart end devices responsible for capturing images. The proposed method deploys the intelligent algorithms on a fog computing layer.

In *AI on edge device for laser chip defect detection [101]*, the authors propose a novel edge system to detect defects in chips based on laser images. Intelligence algorithms run on the edge device, accelerated by a neural network stick. The edge devices are computer-on-modules with an attached microscope and a neural network accelerator.

The work *An AI-edge Platform with Multimodal Wearable Physiological Signals Monitoring Sensors for Affective Computing Applications [102]* displays a proposal for a novel platform to monitor user's emotions. The edge devices are a set of wearable sensors measuring electroencephalography (EEG), electrocardiography (ECG), and photoplesmiography (PPG). Intelligence algorithms run on an AI/IoT edge server based on FPGA and a RISC-V platform.

In *A New Deep Learning-Based Food Recognition System for Dietary Assessment on An Edge Computing Service Infrastructure [103]*, the authors propose a food recognition system based on edge computing and AI. Edge devices in this proposal

are mobile image capturing devices. The intelligent processing happens in a cloudlet layer.

The authors of *Implementation of Pavement Defect Detection System on Edge Computing Platform [104]* present a novel edge computing platform to detect defects in pavements. Edge devices are a composition of cameras and computing modules. Intelligence algorithms run on these specialized modules, which can be computer-on-modules or FPGAs.

Edge-AI-Based Real-Time Automated License Plate Recognition System [105] presents a new license plate recognition system based on Edge AI and IoT. The edge intelligence runs on the device, which has AI accelerators. The edge devices are image-capturing computer-on-modules.

In *A Wireless Multi-Channel Capacitive Sensor System for Efficient Glove-Based Gesture Recognition With AI at the Edge [106]*, the authors present a wearable-based edge system for ASL gesture recognition. Their edge device is a wearable system based on capacitive sensors. The edge intelligence runs in the wearable computer-on-module.

Blockchain-Based Trust Edge Knowledge Inference of Multi-Robot Systems for Collaborative Tasks [107] provides a proposal for a blockchain-based system to provide AI inferences in multi-robot systems. Edge devices are robots composing a multi-robot system. Edge servers, in this case, are local Edge AI nodes that run AI inference algorithms.

The paper *Deep Learning Models for Magnetic Cardiography Edge Sensors Implementing Noise Processing and Diagnostics [108]* presents an implementation of deep learning models to process noise and aid in diagnostics using magnetic cardiography edge sensors. Edge devices, in this case, are magnetocardiography (MCG) sensors. Signals are processed in a high-performance edge machine as an edge server.

In the work *Deep Reinforcement Learning for Collaborative Edge Computing in Vehicular Networks [109]*, the authors display a mobile edge computing system based on collaborative and decentralized intelligence in vehicular applications. Edge devices are onboard computers in vehicular networks. Edge servers are distributed edge clusters that run the intelligence algorithms.

The work named *Development and Validation of an EEG-Based Real-Time Emotion Recognition System Using Edge AI Computing Platform With Convolutional Neural Network System-on-Chip Design [110]* displays a solution to recognize emotions using electroencephalography (EEG) data. The edge device employed in this system is a field-programmable gate array (FPGA) board. For the edge server, the application uses networking to develop high-level appliances using the inferences from the edge devices.

In the article *Distributed Deep Learning Model for Intelligent Video Surveillance*

Systems with Edge Computing [111], the authors propose a novel model for video surveillance systems based on Edge AI. These are two layers of edge servers, organized as a mobile edge computing appliance and getting closer to cloud integration. In this context, the edge devices are video capturing devices.

Edge-AI in LoRa-based Health Monitoring: Fall Detection System with Fog Computing and LSTM Recurrent Neural Networks [112] presents the proposal of an ecosystem based on Edge AI and LPWAN connections toward remote healthcare monitoring. The edge devices are wearable IoT sensors. Edge servers provide the intelligence algorithms necessary to analyze signals and detect falls.

The article *Edge artificial intelligence for the industrial internet of things applications: an industrial edge intelligence solution [113]* presented a novel edge-computing-based model for developing intelligent industrial solutions. The edge devices present in this work are industrial end devices performing several tasks. The intelligence models are updated in the edge servers within the federated learning organization.

AI-Aided Individual Emergency Detection System in Edge-Internet of Things Environments [114] displays an innovative individual emergency detection system based on IoT and edge intelligence. The edge devices, in this case, are mobile phones, capturing their sensors' data throughout the usage time. The intelligence models run on edge servers.

In *Economic data analytic AI technique on IoT edge devices for health monitoring of agriculture machines [115]*, the authors provide a novel integration model for agricultural machines using Edge AI. In their case, the edge devices are agriculture machines (AgMs) producing data from interconnected sensors. Edge servers run in distributed mobile devices which perform inferences using lightweight models.

In the article named *Intelligent Edge Computing for IoT-Based Energy Management in Smart Cities [116]*, the authors present a design for an Edge-AI-based energy management system. In their perspective, the edge devices are IoT sensors distributed on the network. Mid-term edge servers provide intelligent services for energy management.

Intelligent Search and Find System for Robotic Platform Based on Smart Edge Computing Service [117] displays a novel edge service to provide intelligent task division algorithms. The edge devices in this perspective are autonomous robots. A base station contains a terminal that provides a self-design intelligence algorithm and interfaces with an operator.

In the article named *Intelligent Traffic Accident Detection System Based on Mobile Edge Computing [118]*, the authors display a traffic accident detection system based on Edge AI. The edge devices within this perspective are smartphones within a mobile edge computing (MEC) network. Mobile edge servers provide the intelli-

gence algorithms as a service.

The authors of *Low-Power HW Accelerator for AI Edge-Computing in Human Activity Recognition Systems [119]* propose an efficient hardware-accelerated system for human activity recognition. The edge device in this context is self-developed hardware to read inertial sensors and processes. The edge intelligence algorithms are hardware accelerated hybrid neural networks.

In the article named *pAElla: Edge AI-Based Real-Time Malware Detection in Data Centers [120]*, the authors propose a lightweight and scalable solution to monitor the presence of malware in data centers. The intelligence algorithms are designed to run in the edge devices, which feed inferences to an edge server. The edge devices are IoT computer-on-modules based on ARM processors.

The paper *Passban IDS: An Intelligent Anomaly-Based Intrusion Detection System for IoT Edge Devices [121]* displays a novel IoT intrusion detection system (IDS) based on smart edge computing. The edge devices in this context are IoT gateways. Machine learning models on these gateways provide predictions of potentially hazardous packages.

Real-Time Apple Detection System Using Embedded Systems With Hardware Accelerators: An Edge AI Application [122] presents a novel system to detect apples in orchards based on Edge AI devices. For edge devices, the authors propose several solutions based on embedded computers. These computers run intelligent models to detect the apples directly in frames taken from the orchard.

The paper *Towards the edge intelligence: Robot assistant for the detection and classification of human emotions [108]* provides a novel method to detect and classify human emotions using edge computing. The edge devices, in this case, are a wearable sensor and an Edge AI camera. The intelligence models run on the edge devices, which also feed an assistant robot with information.

With *Innovative Vineyards Environmental Monitoring System Using Deep Edge AI [123]*, the authors present a solution to monitor vineyards using edge devices with AI capability. Edge devices are computing nodes capable of running AI models and communicating using LoRaWAN. After acquiring data and predicting, the information is gathered and sent to a central prediction system.

The authors of *AI-doscopist: a real-time deep-learning-based algorithm for localizing polyps in colonoscopy videos with edge computing devices [124]* proposed an algorithm to localize polyps in colonoscopy videos through edge devices. The edge intelligence model uses convolutional neural networks to recognize situations of interest in colonoscopy exam videos. Due to the nature of the models, we analyze that a cloudlet architecture is more suitable for this application.

The researchers responsible for *Wearable IoT Smart-Log Patch: An Edge Computing-Based Bayesian Deep Learning Network System for Multi-Access Physical Monitor-*

ing System [125] proposes a system to monitor physical activities based on wearable patches and Edge AI. The edge devices are IoT smart-log patches that gather data from sensing the users' physical conditions and activities. Intelligence models run on local mobile devices or computer applications.

The authors of *Smart Video Surveillance System Based on Edge Computing [126]* provide an innovative surveillance system based on Edge AI. Edge devices are intelligent cameras capable of running AI models. Intelligence models run on specialized edge hardware that enables running AI models.

In the research named *Wearable Edge AI Applications for Ecological Environments [51]*, the authors provided a novel method to identify diseases in plants and evaluate the spread within canopies. The edge devices, in this case, are smart helmets with IoT capabilities. The intelligence models run on edge servers from a mobile edge computing perspective.

The authors of *Deep Learning Approach at the Edge to Detect Iron Ore Type [127]* display a new system based on intelligent sensors to monitor ore types in conveyor belts. The edge devices, in this case, are sensors integrated into the industrial plant with AI capabilities. The intelligence models run within the smart sensors, enabling instant actuation given the test results.

Green Citrus Detection and Counting in Orchards Based on YOLOv5-CS and AI Edge System [128] displays a new method to detect green citrus in orchards based on Edge AI. The edge device in this proposal is an AI-capable edge computer with a camera to capture images. The models run on the edge device, recognizing green citrus among the orchard.

In *An Edge Intelligence Empowered Recommender System Enabling Cultural Heritage Applications [129]*, the authors present a new recommendation system based on user information and Edge AI. Edge intelligence allows personalized recommendations based on the users' inputs. The edge devices in this perspective are mobile phones.

In the work *A semi-supervised learning approach for network anomaly detection in fog computing [130]*, the authors propose using a Support Vector Machine (SVM) through semi-supervised learning to detect anomalies in the network data flow. The intelligence model, in this case, is a One-Class Support Vector Machine (OCSVM). The semi-supervised learning happens in edge servers, forming a Fog Computing infrastructure.

2.2.2 Preliminary Analyses

After studying the works in the literature, we provide an initial overview of some information extracted from the collected data. This information aims to surround

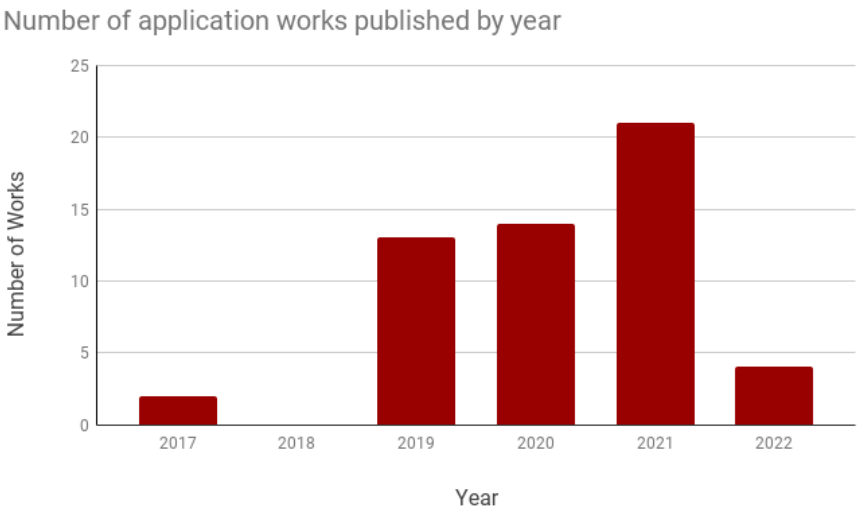


Figure 2.2: Number of application papers per year. The first applications in this context were proposed in 2017, but the it was established in 2019.

the main common aspects of the studied works. Initially, we provide an overview of how many papers were found each year. Figure 2.2 displays the results of this analysis.

From this result, we can initially state that the preliminary works that apply Edge AI concepts came from 2017. Nevertheless, the data indicate that the concept was actually established in 2019. After this year, the number of novel proposed applications is consistent and increased in 2021. Some Edge AI papers were already found in the first quarter of 2022.

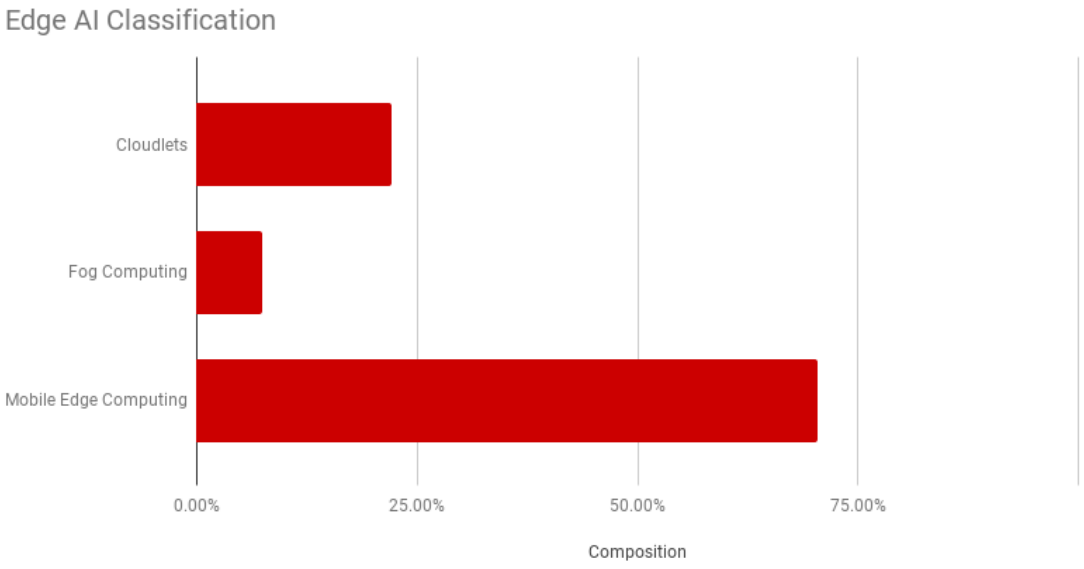


Figure 2.3: Contributions of each category of edge computing to the curated appliances. The majority of works apply "Mobile edge computing".

A second relevant analysis regards the edge computing classification. This section classified the works among "Mobile edge computing", "Fog computing", and "Cloudlets". Of the 54 curated articles, 38 applications used Mobile edge computing, 12 used Cloudlets, and four applied Fog computing. Figure 2.3 displays the classification data analysis in proportions.

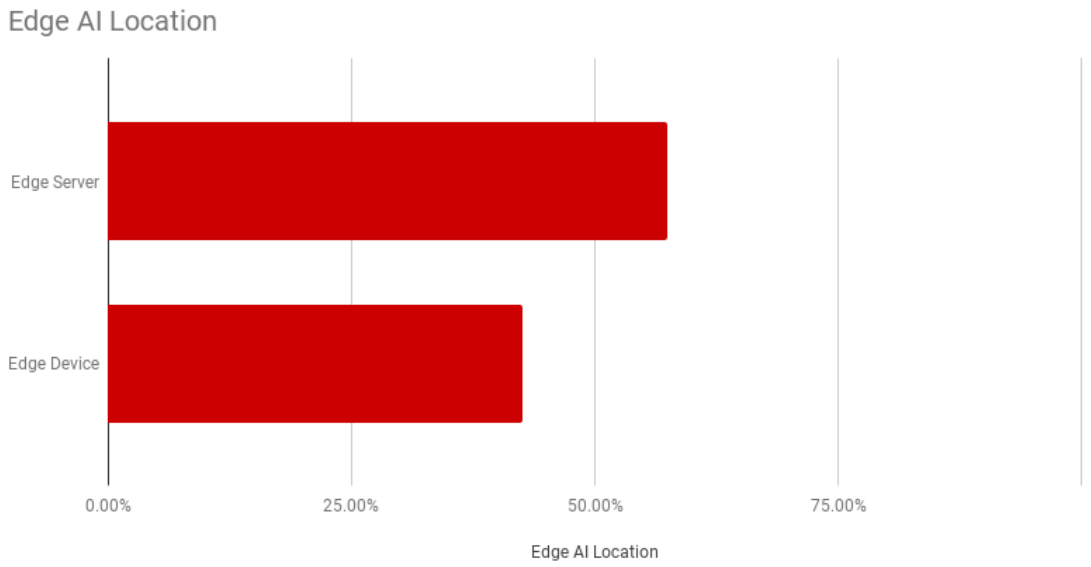


Figure 2.4: Location of the AI algorithms. The analysis display that appliances can deploy models on edge devices or edge servers.

Another feasible study is whether the appliances run the models in the edge devices or edge servers, according to our preliminary classification. A few more works deployed AI models on edge servers than edge devices. Nonetheless, the data is almost balanced, displaying circa 57% of the works with AI deployed on edge servers and 43% on edge devices. In this case, the two works that could run on both devices and servers were classified as "Edge Device". Figure 2.4 displays these results.

Finally, we assess the models used in the studied applications. After inspecting the works initially, we decided to classify the applications among the following classes: **CNNs**, **LSTMs**, **DNNs**, and **Others/Unspecified**. The three initial classes represent the most models which were individually identified in the papers found in the literature, while the last one treats the exceptions. If a system employs more than one algorithm, all models contribute individually to the count. The results, displayed in Figure 2.5, show that CNNs are the most popular models among Edge AI applications. Also, many authors did not specify the machine learning models applied, focusing their papers on the edge computing architecture. DNNs and LSTMs had relevant individual contributions to these numbers.

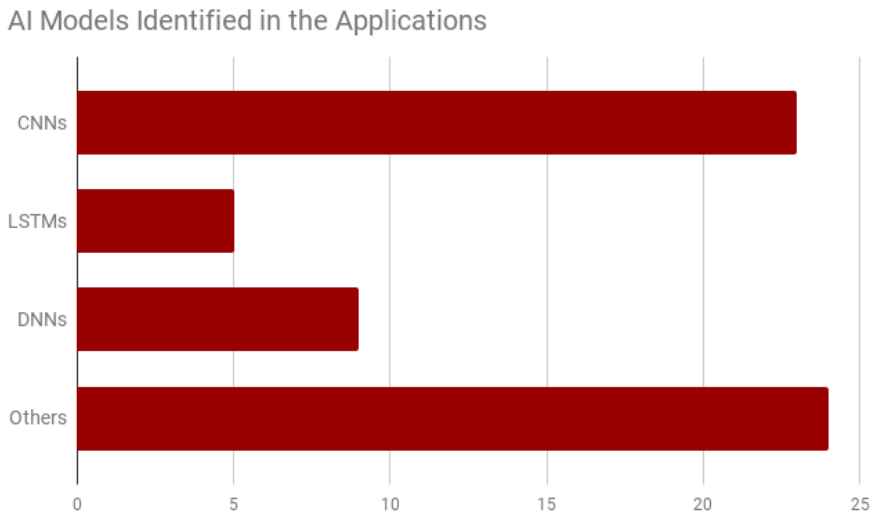


Figure 2.5: AI models identified in the articles. The three individual categories that contributed the most are CNNs, LSTMs, and DNNs. Many authors identify their appliances generically as "Machine Learning"

2.3 Edge Computing and AI taxonomies analysis

After analyzing several applications, the next stage in this research is to understand how the foundation areas are organized. For this matter, we provide a study of the taxonomies in both edge computing and AI means. After this stage, we expect to extract enough information to propose an Edge AI taxonomy.

2.3.1 Edge Computing Taxonomies

Initially, we explored a few works regarding edge computing taxonomies. Finding general information in edge computing taxonomies was not easy, as most texts are focused on single areas or applications. For instance, Beck et al. [131] focus on examining the taxonomy of mobile edge computing solutions. While his work is extensive, it refers to one single of the three main areas of Edge Computing. Thus, this work does not provide a general view of the area.

Regarding a general view of edge computing, Ahmed et al. [132] display a comprehensive taxonomy of information that defines edge computing. An essential aspect of this taxonomy within the context of this work is that the authors also evaluate the same computing paradigms as presented by Khan et al. [2], which were one of the bases of this study. The authors also generally classify the technologies among edge devices and edge servers, later using their taxonomy to evaluate the most critical enabling technologies for edge computing.

Dolui and Datta [133] also use the same three paradigms to classify edge computing applications. These three paradigms work as a baseline to understand the

nodes' organization, nodes location, software architecture, context awareness, geographical proximity, access, and communication. This result reinforces the usage of these three paradigms as the general line to classify edge computing applications.

The analysis of these studies, combined with the previous evaluations, leads to the conclusion that a set of technologies based on edge computing should be initially evaluated according to the three computing paradigms that compose the area. The separation among cloudlets, fog computing, and mobile edge computing allows a general classification of the solutions, leading further to the AI classification.

2.3.2 AI Algorithms Taxonomies

The ways of classifying AI are broader than edge computing. For instance, Sheth et al. [134] classify AI according to the learning paradigm (supervised learning, unsupervised learning, reinforcement learning). While we understand that this will later be important to evaluate Edge AI techniques, we search for a broader classification within this work. Another example of this specificity happens in Baltrušaitis, Ahuja, and Morency [135]. These authors create a taxonomy of specific deep learning relevant within the Multimodal Machine Learning context.

Similarly, Talbi [136] divides AI works according to the algorithms. They provide this classification in the context of evaluating how these algorithms can support metaheuristics. From this work, we understand that using the algorithm paradigms to classify these works is an interesting way of approaching the problem. Still, the way to perform this classification depends on the objective of the algorithms. After searching and evaluating works, we understand that the taxonomy within the AI context should consider the specificity of the desired tasks and concepts.

2.4 Edge AI Taxonomy

Given the knowledge built up to this point, we extracted enough information to propose a new taxonomy to classify Edge AI works and applications. Initially, we start with the edge computing domain, classifying the application among its classes. Then, we evaluate the application according to the AI domain.

The reviews indicate that the classification among the three main classes is adopted for the edge computing traits in both application works and taxonomies. Thus, the first layer of the Edge AI taxonomy evaluates if the works belong to *mobile edge computing*, *fog computing* or *cloudlets* domains. This separation is efficient for dividing edge computing works while separating among edge devices or edge servers only makes sense within *mobile edge computing* domain.

In the AI computing features, we initially observed several ways to separate AI

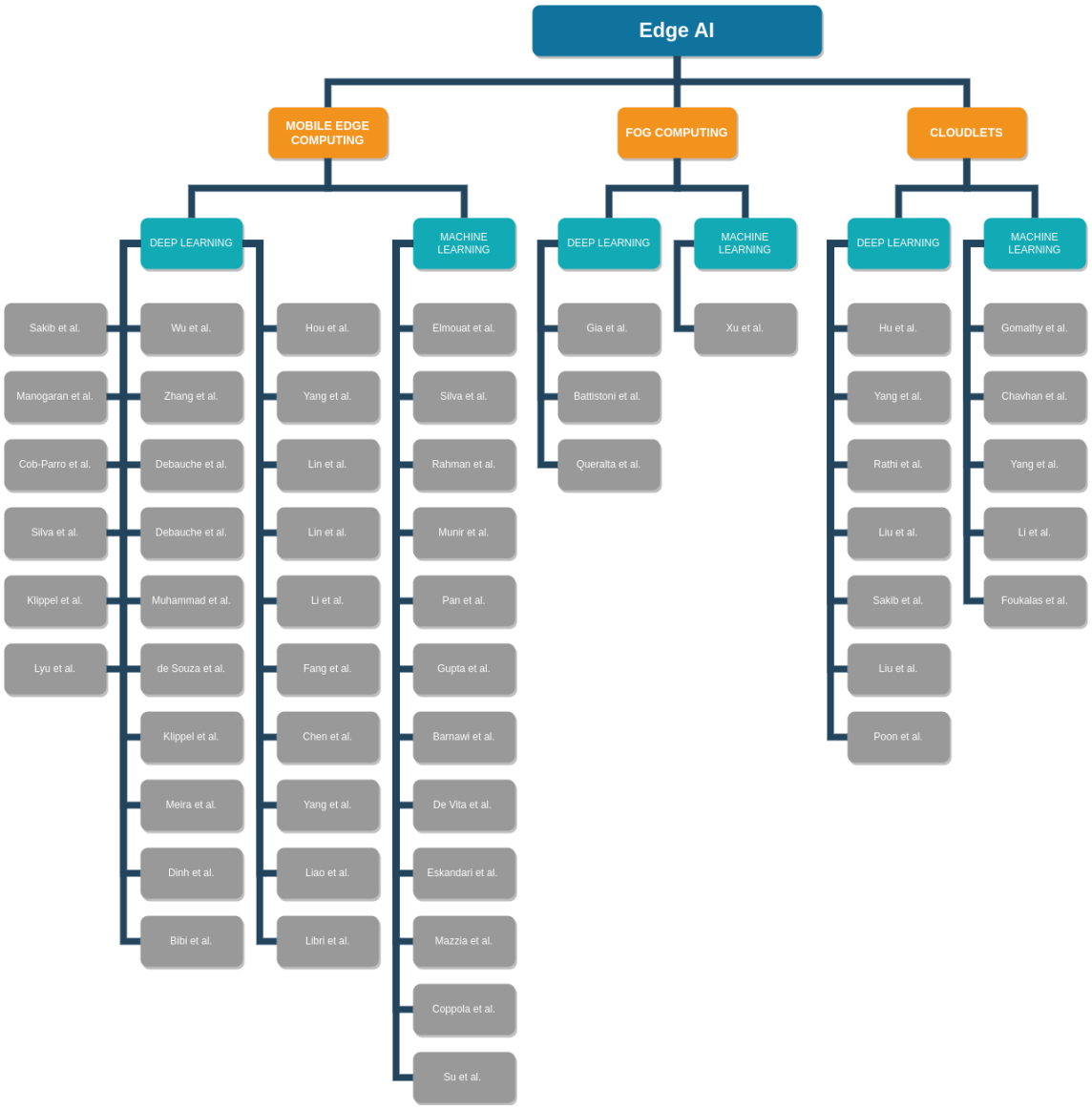


Figure 2.6: Classification of the works from Section 2.2 according to the proposed taxonomy. Each gray square is a single work evaluated in this section of the work.

algorithms. Thus, interpreting the data obtained in Section 2.2 was the critical aspect of defining this division. In Figure 2.5, we observe that most works use CNNs, LSTMs, or DNNs as the basis of the application. These models belong to the deep learning domain. A similar proportion of works employ other machine learning models or paradigms to employ AI on edge. Thus, we defined two classes: *deep learning*, containing the applications that employ algorithms belonging to the deep learning domain, and *machine learning*, encompassing the applications that employ other methods and paradigms.

In Figure 2.6, we display the reclassification of the works presented in Section 2.2 according to the proposed taxonomy. We manually evaluated each work in this section according to the proposed divisions. If the works employ methods from more than one domain, including deep learning, we consider them to belong to the *deep*

learning classes.

What we observe in this case is that the proportions indicate by Figures 2.3, 2.4, and 2.5 were reflected in this taxonomy. Most of the works from Edge AI belong to the *mobile edge computing* domain. Also, most employ algorithms belonging to the *deep learning* domain. The fewest amount of applications was found regarding the *fog computing* domain, while a reasonable amount was found regarding the *cloudlets* domain, with a more balanced division between *deep learning* and *machine learning*.

Finally, we organized a taxonomic table of the displayed works for a better understanding. In Table 2.1, we compiled the results displayed through Figure 2.6. This information complements what we presented previously, working as a catalog to find works related to each area.

Table 2.1: Taxonomic table compiling the collected information from papers.

Paper Title	Edge Computing Paradigm	AI Paradigm
Energy efficient for UAV-enabled mobile edge computing networks: Intelligent task prediction and offloading [85]	Mobile Edge Computing	Deep Learning
Real-time strawberry detection using deep neural networks on embedded system (rtsd-net): An edge AI application [86]	Mobile Edge Computing	Deep Learning
Edge computing and artificial intelligence for real-time poultry monitoring [87]	Mobile Edge Computing	Deep Learning
Edge AI-IoT pivot irrigation, plant diseases, and pests identification [88]	Mobile Edge Computing	Deep Learning
Edge Intelligence-Assisted Smoke Detection in Foggy Surveillance Environments [91]	Mobile Edge Computing	Deep Learning
Deep-Learning-Based Visual Odometry Models for Mobile Robotics [70]	Mobile Edge Computing	Deep Learning
Conveyor Belt Longitudinal Rip Detection Implementation with Edge AI [92]	Mobile Edge Computing	Deep Learning
Edge Deep Learning Applied to Granulometric Analysis on Quasi-particles from the Hybrid Pelletized Sinter (HPS) Process [73]	Mobile Edge Computing	Deep Learning
Towards AI-Based Traffic Counting System with Edge Computing [97]	Mobile Edge Computing	Deep Learning
Edge AI-Based Automated Detection and Classification of Road Anomalies in VANET Using Deep Learning [98]	Mobile Edge Computing	Deep Learning
AI on edge device for laser chip defect detection [101]	Mobile Edge Computing	Deep Learning
An AI-edge Platform with Multimodal Wearable Physiological Signals Monitoring Sensors for Affective Computing Applications [102]	Mobile Edge Computing	Deep Learning
Implementation of Pavement Defect Detection System on Edge Computing Platform [104]	Mobile Edge Computing	Deep Learning
Edge-AI-Based Real-Time Automated License Plate Recognition System [105]	Mobile Edge Computing	Deep Learning
Blockchain-Based Trust Edge Knowledge Inference of Multi-Robot Systems for Collaborative Tasks [107]	Mobile Edge Computing	Deep Learning
Development and Validation of an EEG-Based Real-Time Emotion Recognition System Using Edge AI Computing Platform With Convolutional Neural Network System-on-Chip Design [110]	Mobile Edge Computing	Deep Learning
Distributed Deep Learning Model for Intelligent Video Surveillance Systems with Edge Computing [111]	Mobile Edge Computing	Deep Learning
AI-Aided Individual Emergency Detection System in Edge-Internet of Things Environments [114]	Mobile Edge Computing	Deep Learning
Intelligent Traffic Accident Detection System Based on Mobile Edge Computing [118]	Mobile Edge Computing	Deep Learning
pAElla: Edge AI-Based Real-Time Malware Detection in Data Centers [120]	Mobile Edge Computing	Deep Learning
Towards the edge intelligence: Robot assistant for the detection and classification of human emotions [108]	Mobile Edge Computing	Deep Learning
Wearable IoT Smart-Log Patch: An Edge Computing-Based Bayesian Deep Learning Network System for Multi Access Physical Monitoring System [125]	Mobile Edge Computing	Deep Learning
Smart Video Surveillance System Based on Edge Computing [126]	Mobile Edge Computing	Deep Learning
Wearable Edge AI Applications for Ecological Environments [51]	Mobile Edge Computing	Deep Learning
Deep Learning Approach at the Edge to Detect Iron Ore Type [127]	Mobile Edge Computing	Deep Learning
Green Citrus Detection and Counting in Orchards Based on YOLOv5-CS and AI Edge System [128]	Mobile Edge Computing	Deep Learning
Edge computing and artificial intelligence for landslides monitoring [89]	Mobile Edge Computing	Machine Learning
IDiSSC: Edge-computing-based Intelligent Diagnosis Support System for Citrus Inspection [53]	Mobile Edge Computing	Machine Learning
A Lossless Data-Hiding based IoT Data Authenticity Model in Edge-AI for Connected Living [94]	Mobile Edge Computing	Machine Learning
Intelligent and Smart Irrigation System Using Edge Computing and IoT [99]	Mobile Edge Computing	Machine Learning
A Wireless Multi-Channel Capacitive Sensor System for Efficient Glove-Based Gesture Recognition With AI at the Edge [106]	Mobile Edge Computing	Machine Learning
Economic data analytic AI technique on IoT edge devices for health monitoring of agriculture machines [115]	Mobile Edge Computing	Machine Learning
Intelligent Search and Find System for Robotic Platform Based on Smart Edge Computing Service [117]	Mobile Edge Computing	Machine Learning
Low-Power HW Accelerator for AI Edge-Computing in Human Activity Recognition Systems [119]	Mobile Edge Computing	Machine Learning
Passban IDS: An Intelligent Anomaly-Based Intrusion Detection System for IoT Edge Devices [121]	Mobile Edge Computing	Machine Learning
Real-Time Apple Detection System Using Embedded Systems With Hardware Accelerators: An Edge AI Application [122]	Mobile Edge Computing	Machine Learning
Innovative Vineyards Environmental Monitoring System Using Deep Edge AI [123]	Mobile Edge Computing	Machine Learning
An Edge Intelligence Empowered Recommender System Enabling Cultural Heritage Applications [129]	Mobile Edge Computing	Machine Learning
Edge AI in smart farming IoT: CNNs at the edge and fog computing with LoRa [90]	Fog Computing	Deep Learning
AI at the Edge for Sign Language Learning Support [100]	Fog Computing	Deep Learning
Edge-AI in LoRa-based Health Monitoring: Fall Detection System with Fog Computing and LSTM Recurrent Neural Networks [112]	Fog Computing	Deep Learning
A semi-supervised learning approach for network anomaly detection in fog computing [130]	Fog Computing	Machine Learning
iRobot-Factory: An intelligent robot factory based on cognitive manufacturing and edge computing [82]	Cloudlets	Deep Learning
AI-enabled emotion-aware robot: The fusion of smart clothing, edge clouds and robotics [83]	Cloudlets	Deep Learning
An edge AI-enabled IoT healthcare monitoring system for smart cities [84]	Cloudlets	Deep Learning
A New Deep Learning-Based Food Recognition System for Dietary Assessment on An Edge Computing Service Infrastructure [103]	Cloudlets	Deep Learning
Deep Learning Models for Magnetic Cardiography Edge Sensors Implementing Noise Processing and Diagnostics [108]	Cloudlets	Deep Learning
Intelligent Edge Computing for IoT-Based Energy Management in Smart Cities [116]	Cloudlets	Deep Learning
AI-doscopist: a real-time deep-learning-based algorithm for localising polyps in colonoscopy videos with edge computing devices [124]	Cloudlets	Deep Learning
Investigating the Spread of Coronavirus Disease via Edge-AI and Air Pollution Correlation [93]	Cloudlets	Machine Learning
Edge Computing AI-IoT Integrated Energy Efficient Intelligent Transportation System for Smart Cities [95]	Cloudlets	Machine Learning
National Sports AI Health Management Service System Based on Edge Computing [96]	Cloudlets	Machine Learning
Blockchain-Based Trust Edge Knowledge Inference of Multi-Robot Systems for Collaborative Tasks [107]	Cloudlets	Machine Learning
Edge artificial intelligence for industrial internet of things applications: an industrial edge intelligence solution [113]	Cloudlets	Machine Learning

2.5 Final Remarks

Edge computing and artificial intelligence initially had conflicting requirements. As AI required more processing throughout time, edge computing developed mainly

over hardware miniaturization and increased mobility. Nevertheless, a recent trend displayed increasing applications applying both concepts together, especially in mobile edge computing. Although other research papers survey functional aspects [137], applications for specific areas [138], or even specific technologies [139], the area lacks a general conceptualization and taxonomy.

This trend had several names up to this moment, but every one of them had similar premises of uniting edge computing with AI algorithms. Moreover, a significant number of papers describing applications in this context were published from 2019 on. To define this area, we started from the fundamental areas behind the appliances: edge computing and AI. We named this novel concept Edge AI, one of the names used to describe these appliances in the literature.

Edge computing is divided into three main areas: cloudlets, fog computing, and mobile edge computing. Cloudlets refer to resourceful infrastructures built to provide AI closer to the end-user. Fog computing regards the distribution among heterogeneous devices and the virtualization of cloud-like services. Mobile edge computing describes more mobile and isolated infrastructure, where less powerful devices and more restricted resources are employed to provide AI.

AI can be divided in many ways. Mainly, it refers to machine learning processes where algorithms learn to solve a task from provided data. The more modern applications of machine learning are described in the literature as deep learning, having several specific algorithms in this domain. Within edge computing, AI can run in single-edge devices or as a service in edge servers. The first mode is only referred to in the mobile edge computing domain.

This information helped us create a taxonomy to classify applications described in papers from the literature. The works can belong to three edge domains: cloudlets, fog computing, and mobile edge computing. The first layer evaluates the edge computing classification. The second layer divides the AI among the deep learning domain or belonging to any other paradigm of AI.

Finally, some of the main future trends regarding Edge AI integrate concepts like Blockchain, 5G, 6G, and Federated Learning [78, 134, 140–142]. Future works can explore these aspects as the ground for further investigation. Also, as mobile edge computing contains the most developed area among these domains within Edge AI, it requires further investigations and classifications on its own, possibly as a "Mobile Edge AI" concept.

Chapter 3

Wearable Edge AI

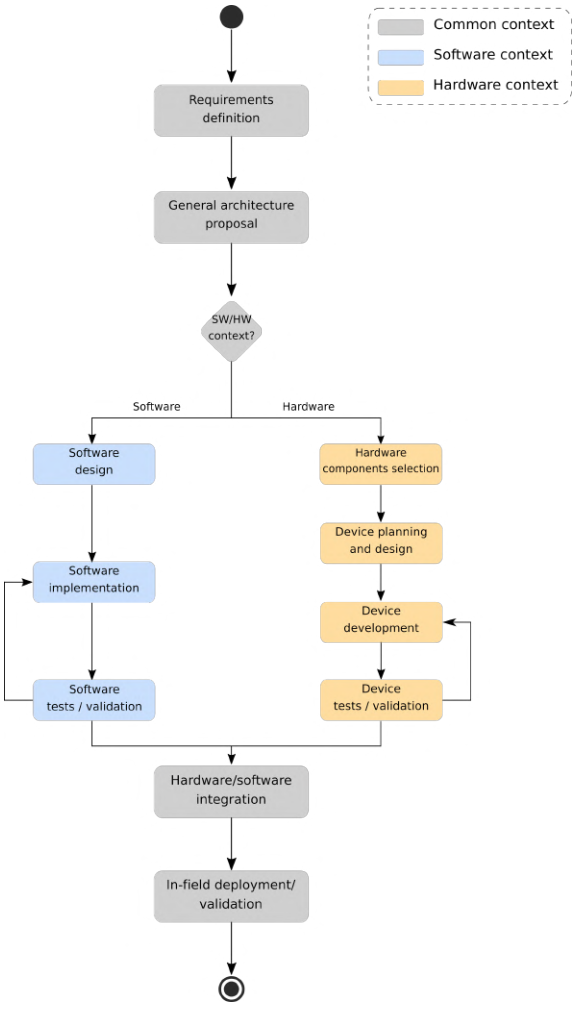


Figure 3.1: Original Hardware and Software co-design process, presented in [4]

This chapter discusses the creation of the *Wearable Edge AI* concept. Our discussion begins with an understanding of how cooperative wearable systems work. Then, we review the hardware/software co-design process. Finally, we assess the wearable edge AI concept, surrounding its constraints and its creation and valida-

tion processes.

Our concept starts from the wearable systems perspective. Thus, it is required to understand the design process behind them. As an inheritance from their embedded systems' origin, wearable devices and systems usually follow a process known as hardware and software co-design [4], exemplified in Figure 3.1.

As displayed, a series of steps are required to produce these solutions. Initially, there are two general steps to start projecting these appliances:

- **Requirements definition:** This stage involves collecting, assessing, and examining the system requirements. This is achieved by conducting a thorough analysis of the stakeholders' needs;
- **General architecture proposal:** During this stage, professionals utilize the data collected from the previous step to create a preliminary outline of the proposal. This initial blueprint necessitates defining the limitations for both software and hardware to strategize the flow of information.

Following these preliminary stages, there are several simultaneous activities related to the software and hardware characteristics. These steps occur concurrently since their outcomes are interdependent. They are:

- **Hardware:**
  - **Hardware components selection:** The initial step in the hardware phase is to list the necessary hardware components required to execute the proposed tasks and functions.
  - **Prototype planning and design:** Hardware components act as input data for devising and developing the final prototype. During this stage, connections between components are specified, and if required, the Printed Circuit Board (PCB) layout is outlined;
  - **Prototype development:** This phase involves the production of the wearable device. The primary Central Processing Unit (CPU) and hardware peripheral components are physically linked, potentially employing the final version of the PCBs created in the previous stage. Finally, the components can be attached to the garment;
  - **Device tests/validation:** Multiple rounds of on-site hardware testing and validation are conducted to authenticate the connections established with each component. If any issues arise during this stage, they can be rectified or altered in a new development round.
- **Software:**

- **Software design:** The previously stated requirements can also serve as a guide for devising the relevant software. This phase entails defining functionalities and how internal modules will communicate with one another;
- **Software implementation:** The design created earlier can now be utilized as a point of reference for precisely coding the software. Typically, specific and lightweight programming languages/frameworks can be employed during this stage, depending on the requirements of the entire solution;
- **Software tests/validation:** This module is responsible for testing and verifying whether the resultant software adheres to the previously stated requirements. If the solution fails to meet the necessary requirements or does not pass the validation tests, a new development round can be initiated.

The remaining blocks are used by both contexts:

- **Hardware/software integration:** This stage involves integrating the previously developed hardware and software components. This is a crucial step since both sections were separately developed until this point. Basic and complex functionalities can be assessed to determine whether they produce the desired output.
- **In-field deployment/validation:** This marks the final stage in the design of wearable systems. The wearable device, which now incorporates integrated hardware and software modules, is deployed and authenticated through field sessions. This moment is also utilized in numerous research projects to collect and retrieve empirical data.

3.1 Cooperative Wearable Systems

The *Wearable Edge AI* concept originates from the recognition that wearable systems do not function as isolated processing units. In Chapter 1, we highlighted that Wearable Computing systems can be incorporated into cyber-physical applications.

Augimeri et al. [143] and Fortino et al. [144] present four distinct propositions for collaborative body sensors. In two of these propositions, data collected from a single user provides information to one or several stations. In the other two, data collected from multiple users is utilized to feed one or several stations. Similarly, to comprehend the methods of collaborating using wearable devices, we categorize Cooperative Wearable Systems into two scenarios:

1. Single-User Cooperative Wearable Systems,

2. Multiple-User Cooperative Wearable Systems.

The first type happens when the same user utilizes multiple wearable devices to compose a system, feeding one or multiple applications in base stations. The second type occurs when many users wear the same equipment, with post-processed data and flexibility gained in the final appliances in individual or various consumer stations.

Single-User Cooperative Wearable Systems

Wearable compositions are crucial for monitoring both the user and the context of the environment. Mihovska and Sarkar [145] introduce a vital concept for this context: Human-Centric Sensing. According to these researchers, the Internet of Things (IoT) and connectivity of modern devices provide the necessary tools to establish cooperative systems around the user. These systems can monitor both user signals and the surrounding environment.

Zhang et al. [146] proposed a cooperative environment based on a glove-shaped pressure sensor and an armband to enable gesture recognition for Human-Computer Interaction devices. They conducted a series of tests to verify the recognition accuracy of their system.

Peng and Peng [147] assert that Body-Area Networks have become a critical tool for creating innovative healthcare solutions using collaborative wearable devices. They proposed a cooperative communication strategy to integrate multiple devices in a Wireless Body-Area Network (WBAN) and validated their proposal through simulations.

Nguyen-Huu et al. [148] present a combination of a wearable device and smartphone working together to monitor daily activities in an indoor environment. Their system utilizes both an armband and a smartphone to gather sensor data, process and recognize activities, and transmit this information to a web server for further analysis. They validated their system through performance evaluations on their activity recognition and lifelogging algorithms.

Most of the proposed systems and architectures employ wearable devices as IoT nodes in a Wireless Body-Area Network, and in some instances, smartphones serve as connection gateways. Finally, these works follow the same systematic process, beginning with requirement analysis, architecture proposal, implementation, and validation.

Multiple-User Cooperative Wearable Systems

Pimentel et al. [149] developed a system that monitors the stress levels of multiple surgeons by using commercial wearable devices for vital signs monitoring, such as ECG and actigraphy signals. The data gathered from these devices is sent to an Android application that marks events, creates reports, and stores the data in a database. The authors validated their proposal by conducting statistical studies on the self-assessment tools present in the Android application.

In another study, Prakash and Ganesh [150] established a communication environment for cooperative health monitoring in hospitals using wearable devices. They tested their proposal using network simulator applications to verify the packet transmission performance.

Pham et al. [151] proposed a wearable-based system environment to monitor older adults and patients with Parkinson's disease. Their architecture was based on IoT wearable devices with Inertial Measurement Units (IMUs) located on the user's lower limb. To validate their proposal, they tested their environment and system with actual target users and compared the results with video recognition using statistical analysis. These studies followed a similar systematic process, including analyzing the context of use and system requirements, proposing an architecture, assembling a prototype environment, and performing validation tests.

Even in a multi-user context, the current state-of-the-art works follow a systematic process in proposing wearable systems. This includes analyzing the context of use and gathering system requirements, proposing an architecture, assembling a prototype environment to test the proposal, and performing validation tests.

3.2 Wearable Edge AI

Field research environments usually lack the infrastructure required to integrate Edge wearable devices with computer systems. Wearable Edge AI is a concept that aims to provide services to ecological researchers and practitioners by offering local services based on artificial intelligence applications. The concept enables embedded devices to perform machine learning models that assist humans in the decision-making process in real-time.

The growing interest in machine learning, deep learning, and other computational intelligence applications has led to discussions on how to bring these algorithms to the edge. According to Chen and Ran [152], the main challenges of using machine and deep learning in this context are latency, scalability, and privacy. These technologies are typically used for Computer Vision and Natural Language Processing, and relevant features in these applications include cost, reliability, latency, and pri-

vacy, as stated by Wang et al. [153].

An increasing number of wearable computing applications use edge computing to provide insights based on machine learning. For instance, there are appliances in health monitoring [125, 154, 155], ergonomics [156], activity tracking [157], and so on. Most of these applications are user-centered and focus on monitoring the users' conditions rather than the environment. Although there are many applications and common features, the authors have yet to define Wearable Edge AI as a single topic. Therefore, this work aims to formalize the constraints and design patterns for such applications.

Considering these aspects, we produced an entry for the Wearable Edge AI concept. It considers the aspects raised in this chapter and Chapter 2 as foundations. We define Wearable Edge AI as:

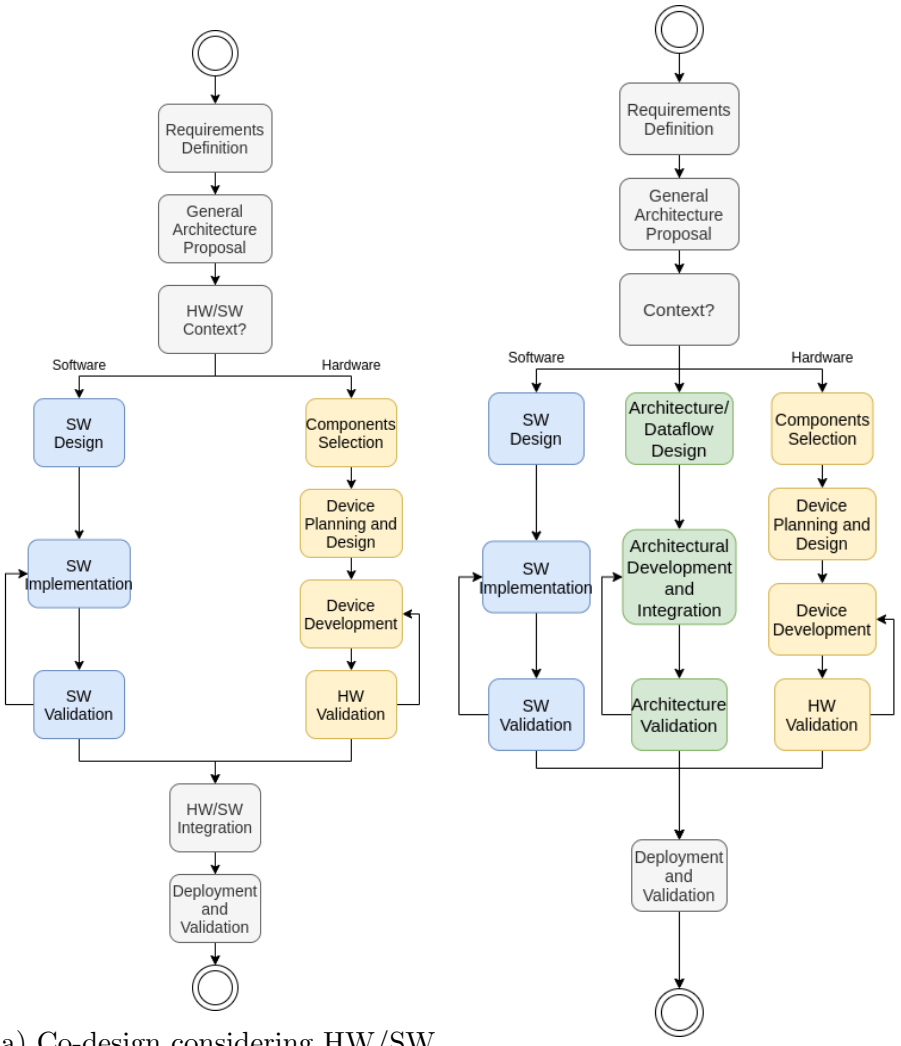
- Wearable Edge AI is the set of methods that describes the design process and validation of solutions that combine wearable computing, edge computing, and machine learning concepts in developing novel appliances, systems, and applications to solve real-world problems.

Although this concept is very similar to the one presented in Chapter 2, the presence of wearable computing concepts and constraints is a crucial difference. Wearable applications have constraints that are not included in edge computing by itself. These devices are especially constrained in resources and capabilities, reduced in size, but require the usage of machine and deep learning algorithms to perform inferences.

3.2.1 Rethinking the hardware/software co-design for Edge AI solutions

The co-design principle of hardware and software is crucial while designing new solutions for embedded and wearable systems [158, 159]. This principle guides the process of designing and validating parallel hardware and software aspects, which later integrate into novel systems.

However, in this text, we suggest a new approach to this pattern. We believe that architectural factors must also be validated parallelly during the design process in edge computing and IoT approaches. To illustrate this, Figure 3.2 displays the traditional and new diagrams for the co-design. While Figure 3.2a shows the traditional approach for the Hardware/Software (HW/SW) co-design pattern, Figure 3.2b explores a new branch for designing and validating the architecture in parallel with hardware and software constraints.



(a) Co-design considering HW/SW

(b) New co-design approach

Figure 3.2: Co-design principle diagrams. The traditional approach does not consider architectural aspects in parallel with the HW and SW design.

The traditional co-design approach typically begins with defining requirements and proposing a general architecture. Then, the constraints are segregated between hardware and software contexts, and the development of both aspects proceeds in parallel. After validating both components, the solution is integrated. This architecture is effective for designing single solutions for wearable and embedded systems, regardless of their abstraction levels. However, when dealing with multiple and variable architectures, this approach becomes fragile. If the validation after integration fails due to architectural traits, the solution becomes obsolete. This problem becomes even more critical when designing a Wearable Edge AI, where such factors cannot be ignored.

Thus, we suggest that architecture validation should be a new branch after context splitting. When designing new devices, identifying the essential aspects of architectural design and validating these designs during the proposal process is cru-

cial. Finally, the integration must happen in parallel, and the last stage is deploying and validating all systems together. In the architecture branch, represented by green blocks in Figure 3.2b, there are three new stages:

- *Architecture/Dataflow Design*: In this stage, the proposal must identify how the devices communicate within the network. In the context of IoT and Edge Computing, devices communicate with each other providing services, insights, and information. Integrating devices in the same WBAN/WPAN, or even multiple devices with multiple WLAN users, requires a dataflow design.

- *Architectural Development and Integration*: After defining the roles of each device within the network, as well as the integration protocols, the architecture must be developed in parallel with the integration of hardware components and individual software traits.

- *Architecture Validation*: Like the other branches, the architecture must also be validated using formally-defined tests. This aspect enforces the design process and identifies flaws in the development process that must be assessed.

As displayed, these conditions help assessing the possibility of creating solutions combining the concepts of wearable computing, edge computing, and AI. We have previously studied the development of an Edge AI concept. Now, we define Wearable Edge AI as the set of methods that describes the design process and validation of solutions that combine wearable computing and Edge AI concepts in developing novel appliances, systems, and applications to solve real-world problems.

Chapter 4

Case Study - Wearable Edge AI towards environmental studies

This chapter is dedicated to the evaluation of the case-studies developed towards our first stakeholders. In the introductory section, we defined these target audience as the researchers, students, professors, and practitioners within ecology. We developed applications within three main branches: leaf damage estimation, diseases evaluation and mapping, and ants distribution and couting estimation.

4.1 Leaf damage estimation

The first application in our context is the leaf damage estimation. This information is important for the stakeholders related to the ecological field. For instance, researchers use this variable as an indicator to analyze the ecosystem interactions [160, 161], or even to analyze the impact of predators in crops [162, 163].

4.1.1 Requirements

The first step in this analysis is evaluating the requirements for the proposed method. For this matter, we display a version of the co-design diagram presented in Figure 4.1, which is a simplification of the diagram presented in Figure 3.2b.

This representation displays the need to raise the constraints for the application and classify them into the hardware, software or architectural domain. The constraints identified for this matter are:

- The application must reconstruct the leaf shape using artificial intelligence [*Software*].
- The application must have a mean to extract a single leaf image from the environment [*Hardware*].

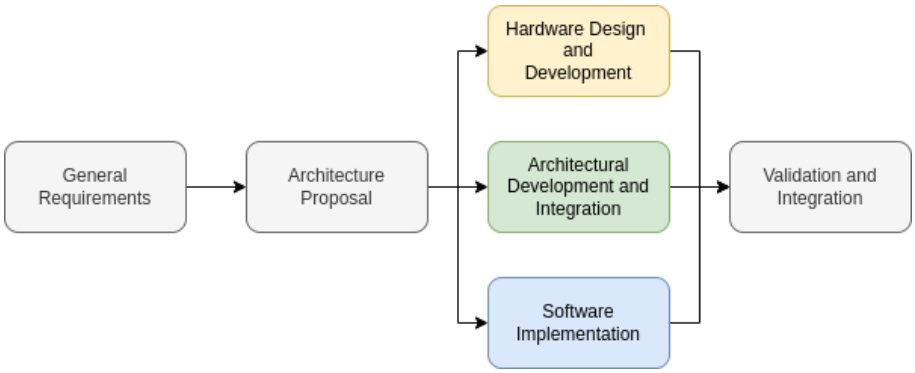


Figure 4.1: Simplified Co-design diagram.

- The application must move the captured image into an AI accelerated hardware [*Architecture*].
- The image from the leaf must be converted into a mask using image processing [*Software*].

Given these constraints, we proposed the usage of a conditional GAN using the Tensorflow environment, allowing the integration with embedded AI accelerated hardware. We also propose the means in which a user can extract a leaf image which can input into such algorithm.

4.1.2 Method overview

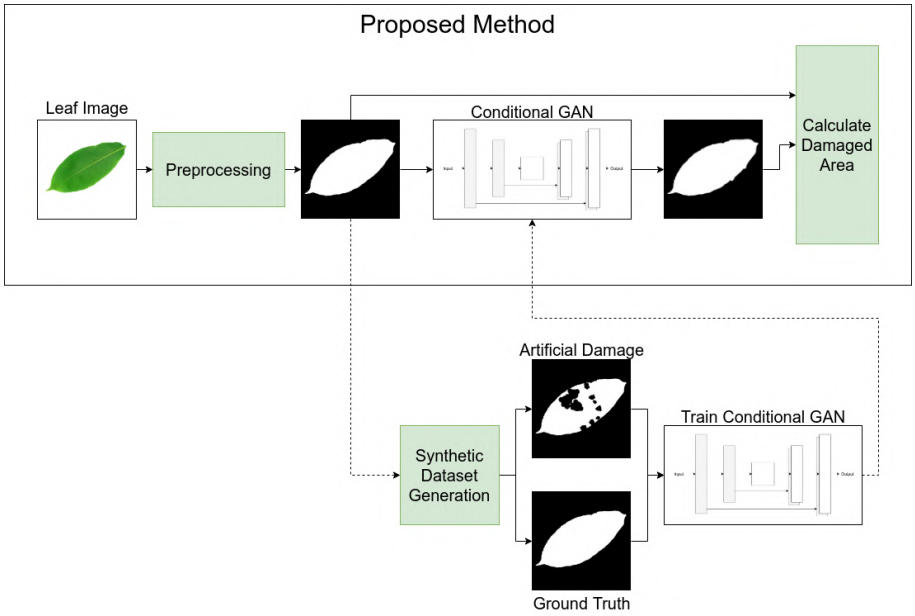


Figure 4.2: Proposed Method and Work Overview

The primary process of our proposed method begins with a preprocessing step that extracts a mask of the leaf area in the image, separating it from the background.

The segmented image is then fed into a Conditional GAN model trained to produce an estimated original leaf shape. Lastly, we compare the output with the input image to determine the estimated percentage of defoliation. Figure 4.2 provides a visual representation of this method.

In addition, we employed the preprocessing method to create a database of masks that includes the complete leaf shapes. These images were utilized to produce a synthetic database that includes leaf masks with artificially induced damage. The latter database was used to train the Conditional GAN method to obtain the test model, utilizing the original masks database as the ground truth. Figure 4.2 also illustrates this series of stages.

4.1.3 Datasets Description

In this study, we worked with two distinct databases. The first one, referred to as FLAVIA henceforth, was introduced by Wu et al. [164]. This dataset comprises 1907 colored images of leaves from 33 distinct plant species, with a resolution of 1600x1200 pixels. We utilized this dataset to create the synthetic database and for model training, validation, and testing purposes.

The second dataset we used is the Middle European Woods dataset presented by Novotny and Suk [165]. We will refer to this dataset as MEW 2012 in the rest of the paper. It consists of 9745 images of leaves from 153 different species, already binarized and available in various resolutions. We employed this dataset to conduct additional tests on the shape reconstruction and damage estimation process.

4.1.4 Preprocessing

As previously mentioned, the initial step in the data flow is preprocessing, which aims to extract the image from the background. We employed this technique to create a synthetic database that includes leaf masks with artificially induced damage. Additionally, we utilized the preprocessing method to generate a database of masks that includes the complete leaf shapes. The preprocessing stage comprises six consecutive steps:

1. Convert to grayscale;
2. Insert paddings to turn the image into a square shape;
3. Reduce the size of the image to 400x400;
4. Enhance the contrast using a radiometric transformation;
5. Calculate the threshold using Otsu's method;

6. Binarize the image;

The initial step involves converting the image colorspace to grayscale. We then add padding to the image to shape it into a square. The padding is selected based on the highest pixel value to improve binarization performance when using thresholding algorithms. Following this, we employ a radiometric transformation to enhance the image contrast, as per Equation 4.1. In this equation, $G_i(x, y) \in [0, 1]$, and $G_i(x, y) \in \mathbb{R}$ represent the normalized pixel values of the original image. It is worth noting that $G_f(x, y) \in [0, 1]$, and $G_f(x, y) \in \mathbb{R}$ represent the output process parameters.

$$G_f(x, y) = G_i(x, y)^{10}. \tag{4.1}$$

Following the contrast enhancement stage, we move on to binarization. To achieve this, we utilized Otsu's method [166] to identify the separation threshold between the leaf and the background. This method works by minimizing the intra-class variance function, as defined in Equation 4.2.

$$\sigma_b^2(k) = \frac{[\mu_T \omega(k) - \mu(k)]^2}{\omega(k)(1 - \omega(k))} \tag{4.2}$$

Where k is the highest number of all the possible threshold values and:

$$\omega(k) = \sum_{i=1}^{k} p(i); \tag{4.3}$$

$$\mu(k) = \sum_{i=1}^{k} i\dot{p}(i); \tag{4.4}$$

$$\mu_T = \sum_{i=1}^{L} i\dot{p}(i). \tag{4.5}$$

The values required for this method are obtained from the histogram, normalized as a probability density function represented by $p(i)$, for the L candidate threshold values in the histogram. This approach provides a reliable estimation for the threshold value required to segment the image from its background. In this equation, $\omega(k)$ represents the class probability, $\mu(k)$ represents the class means, and μ_T represents the global mean. The variable i represents all possible pixel values present in the histogram. Assuming every possible value of k falls within the range of $[0, 255]$ and is a natural number, all possible values of i should be $i \in [1, 255]$, $i \in \mathbb{N}$.

We also used this method to prepare the synthetic database, which was employed to train the conditional GAN method to obtain the test model using the original masks database as the ground truth. During the synthetic dataset generation, we

removed internal holes to create ideal leaf images. Additionally, we developed a novel method for creating randomly artificial damaged leaf images, which we used to generate a dataset for training the conditional GAN.

4.1.5 Synthetic Dataset Generation

As previously mentioned, the preprocessing pipeline was utilized in the synthetic dataset generation process. We employed this method to prepare the images from the dataset for the application. In this section, we present the pipeline involved in producing images with synthetic damage and the processes included in this pipeline.

Most leaves in the dataset are undamaged, while some may show slight damage or light reflection spots. To better represent the ideal leaf shape, we selected the largest contour recognized after binarization to create a complete leaf representation. Using this technique, we generated 1907 masks corresponding to the 1907 images in the dataset. To create a supervised learning dataset in the next stage, we needed to introduce artificial measurable damage into the leaf masks.

In this stage, we discuss how we created artificial random damage on the leaves. Similar to Da Silva et al. [167], we applied artificial damage techniques to generate a training dataset. We initially assumed that the leaf had a slightly higher probability of having damage at its borders. Therefore, we created a 2-D probability distribution, $g(x, y)$, centered on the (x_0, y_0) average center position of the x and y coordinates of the binarized leaf mask image. Equation 4.6 represents this 2-dimensional Gaussian distribution centered at (x_0, y_0), with a standard deviation of σ.

$$g(x, y) = e^{-\frac{(x-x_0)^2+(y-y_0)^2}{2\sigma^2}}. \tag{4.6}$$

Furthermore, we created a probability function $p(x, y)$, for the damage using $g(x, y)$ according to the following equation:

$$p(x, y) = \frac{1 - g(x, y)}{2} + P_0. \tag{4.7}$$

In this scenario, P_0 represents the minimum probability offset. The probability of damage beyond the leaf's boundaries in the image must be zero. This condition is achieved by multiplying the probability function by the leaf mask. During the first stage of this study, we selected a baseline value of $P_0 = 0.3$ and $\sigma = 100$, based on practical tests conducted on the databases. In the second stage, we opted for a baseline value of $P_0 = 0.6$ and $\sigma = 10000$, to generate more damage on average. Figure 4.3 illustrates the probability function for one of the leaves in our dataset.

The damage generated at a point follows a certain rule. Initially, we draw a

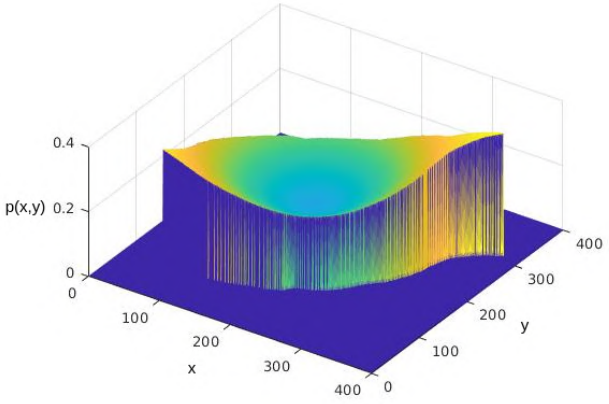


Figure 4.3: Example of damage probability density distribution. This function is used to generate the artificial damage.

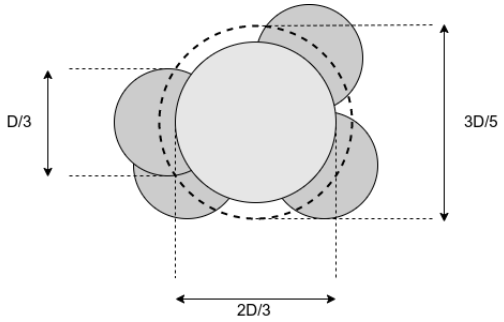


Figure 4.4: Illustration of the punctual artificial damage generation method [5].

circle with a diameter of $2D/3$ for a given reference size of D. Subsequently, we draw four circles with a diameter of $D/3$, centered at random points located over a virtual circle with a diameter of $3D/5$. Figure 4.4 depicts the punctual artificial generation method.

The artificial damage generation algorithm selects several random coordinates and checks the function to determine whether it should insert damage at that point. If the answer is positive, it injects the loss at the spot by randomly selecting a reference size.

To create the synthetic dataset, we generated 12 versions of each leaf with random losses. We selected a pixel located at a coordinate (x, y) as a candidate for receiving the artificial damage. Damage occurs only if the pixel is located within the leaf boundaries. For the first four images, we ran the method with 100 coordinates. For the fifth to eighth images, we executed the procedure with 200 coordinates. For the final four images, we performed the process with 300 coordinates. The resulting dataset consisted of 22884 shapes with varying levels of artificially generated damage.

4.1.6 Conditional GAN Architecture

Our implementation takes the work of Isola et al. [168] as a baseline. For this matter, we applied a U-Net-based conditional GAN architecture. This network has two main modules: a generator and a discriminator. At first, the generator takes an input image and produces a predicted output. Then, the discriminator evaluates the prediction.

In this work, the main architecture is based on a U-Net. U-Nets are generative models of deep neural networks. Originally, this technique was proposed to perform segmentation in biomedical images [169]. They are similar to Variational Autoencoders (VAEs) [170], and due to their generative capability, they can be used to reconstruct images pixel-by-pixel. These networks are applied for recognition and segmentation [171, 172] and for reconstruction [173, 174].

Generator

The generator's architecture is based on an Encoder-Decoder network. For this implementation, a U-Net was used, which has interconnected mirrored layers. The encoder consists of 8 layers (256x256, 128x128, 64x64, 32x32, 16x16, 8x8, 4x4, 2x2), with batch normalization in the intermediate layers. The output also has 8 layers (1x1, 2x2, 4x4, 8x8, 16x16, 32x32, 64x64, 128x128), with batch normalization in the intermediate layers.

Discriminator

On the other hand, the discriminator follows a PatchGAN architecture, which is similar to the encoding section of an encoder-decoder network. In this implementation, the discriminator consists of 5 layers (256x256, 128x128, 64x64, 32x32, 31x31), with batch normalization between the intermediate layers.

Training

The network uses a two-part training method. Initially, the discriminator is trained based on the baseline answers. After that, the generator weights are updated based on the baseline truth and the discriminator guess. The training algorithm was performed for 20 epochs, with the first results being better than the ones presented in the literature. However, to avoid overfitting, the network was trained for an additional 5 epochs, which resulted in a significant improvement. As a result, the error becomes much smaller, making it state-of-the-art on the proposed problem.

4.1.7 Damage Estimation

In the previous section, we discussed the network architecture and its training process. In the preprocessing stage, the image is first converted to grayscale and then subjected to a binarization process. The resulting image is then used as input for the conditional GAN, which produces a mask that represents the predicted original shape. Finally, to calculate the damage percentage, we use the following formula:

$$P_d = (1 - \frac{\sum_{i,j} Im_d(i,j)}{\sum_{i,j} Im(i,j)}) \times 100(\%). \tag{4.8}$$

Here, P_d represents the damage percentage, $\sum_{i,j} Im_d(i,j)$ represents the sum of the binarized value (0 or 1) of each pixel of the damaged leaf image, and $\sum_{i,j} Im(i,j)$ represents the sum of the binarized value (0 or 1) of each pixel of the baseline image. We used the original image mask as the baseline for calculating the ground truth values of damage, while the model's outputs were used to calculate the predicted damage.

4.1.8 Evaluation Methods

In the previous section, we discussed the neural network used to estimate the original shape of damaged leaves, as well as the image datasets and artificial damage generation process used to create the synthetic dataset. In this section, we will focus on the methods used to evaluate the prediction quality.

We started with 22884 original images and used the first 22833 for our analysis. These images were randomly divided into three separate sets: 10% for validation, 10% for testing, and the remaining 80% for training the algorithm. In the second stage, we repeated the process with modified parameters that allowed for more damage. We also used the 22884 images to create a dataset with the same proportions.

After training the model, we performed a round of predictions on the MEW 2012 dataset. To speed up the generation process, we reduced the images to 256x256 pixels and randomly applied 10 to 40 damage coordinates with a probability of $P_0 = 0.7$, resulting in a total of 38980 images. Although the generation process differed slightly, the images had to be resized to 400x400 pixels to be used with the model.

Damage Estimation Evaluation

Similar to Da Silva et al. [167], we calculated the real defoliation percentage d_r and the estimated defoliation percentage d_e. These values were measured on both the validation and test sets, as we generated the synthetic dataset from the ground

truth. We evaluated the Root Mean Square Error (RMSE), which is calculated using the following equation:

$$RMSE = \sqrt{\frac{1}{n}\sum(d_e - d_r)^2}. \tag{4.9}$$

Furthermore, we conducted a series of quantitative and qualitative analyses based on the prediction results.

Shape Reconstruction Evaluation

In the previous subsection, we discussed the evaluation method used for the damage estimation process. In addition to analyzing the quality of the defoliation estimation method, we also conducted a quantified evaluation of the image reconstruction process. To do this, we employed the dice coefficient, which is a widely used method for evaluating image similarity [175–179]. This measurement is used to compare areas and can be easily applied using binarized images. The dice coefficient (DC) is calculated for a pair of images (A and B) using the following equation:

$$DC = \frac{2\|(A \cap B)\|}{\|A\| + \|B\|} \tag{4.10}$$

The resulting coefficient value is always in the range of [0, 1]. A high dice coefficient value indicates that the images are highly similar. Therefore, we used this factor to measure the success of the shape reconstruction process by calculating the dice coefficient to compare the ground truth and model output images.

4.1.9 A Broader Evaluation on the Damage Estimation Results

In the previous section, we discussed the method used to evaluate the leaf damage predictions, which involves estimating the original leaf shape using a Conditional GAN. In this section, we provide a broader overview of the original and predicted data to demonstrate the robustness of the proposed solution.

The first important set of results came from analyzing the RMSE values, which were defined by equation 4.9. The validation dataset had an RMSE value of 0.92 ($\pm$ 1.90), while the test dataset had a value of 0.92 ($\pm$ 1.85). As we previously mentioned, both the validation and test datasets had similar results for the RMSE value. After the second training stage, the error values were even lower. The validation dataset had an RMSE value of 0.61 ($\pm$ 0.99), and the test dataset had a value of 0.52 ($\pm$ 0.73). As previously shown, these results alone represent a significant improvement over the state-of-the-art. Table 4.1 presents the results obtained.

Table 4.1: RMSE Results

	Validation Set	Test Set
Initial Round	0.92 (± 1.90) %	0.92 (± 1.85) %
Improved Round	0.61 (± 0.99) %	0.52 (± 0.73) %

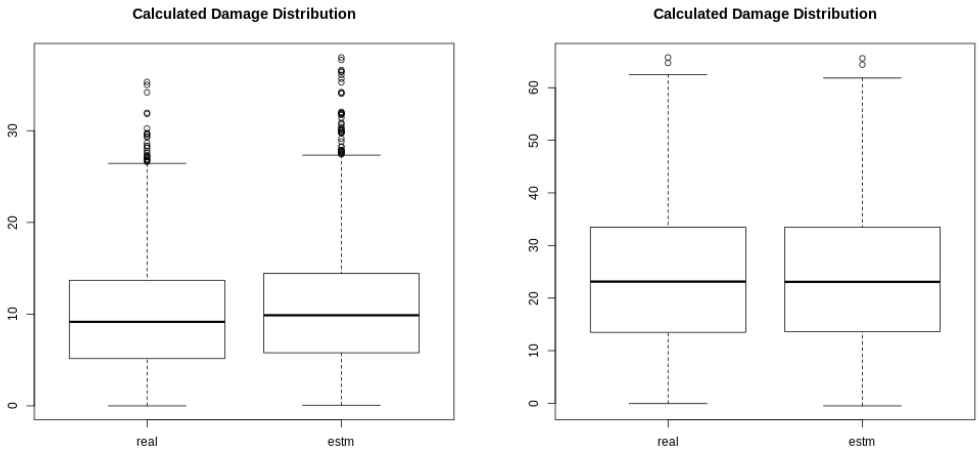


Figure 4.5: Validation Set - Damage Distribution for the Initial and Improved Rounds.

For the initial stage, the validation set comprised 2283 randomly selected images from the original dataset. The estimated average damage on this set was 10.68 ± 6.34%, with a maximum damage value of 37.99%. The real average damage was 9.86 ± 6.03%, with a maximum value of 35.31%. In the second stage, the validation set contained 2288 randomly selected images from the original dataset. The estimated average damage on this set was 23.88 ± 12.97%, with a maximum damage value of 65.59%. The real average damage was 23.84 ± 13.06%, with a maximum value of 65.76%. The distribution plots for this data are shown in Figure 4.5.

Additionally, we created a graph comparing the obtained data with the ground truth for both the initial and improved stages. Figure 4.6 displays the results for the validation dataset in both rounds.

Similarly, the test set comprised 2283 randomly selected images from the initial set. The average estimated damage in this set was 10.66 ± 6.44%, with a maximum value of 56.14%. The real damage distribution average and standard deviation were 9.84 ± 6.19%, with a maximum damage of 52.74%. In the second stage, the test set contained 2289 images. The average estimated damage in this set was 23.61 ± 12.99%, with a maximum value of 63.49%. The real damage distribution average and standard deviation were 23.68 ± 12.99%, with a maximum damage of 63.92%. The boxplots with the distribution of this data are shown in Figure 4.7.

We also created a graph comparing the obtained data with the ground truth for

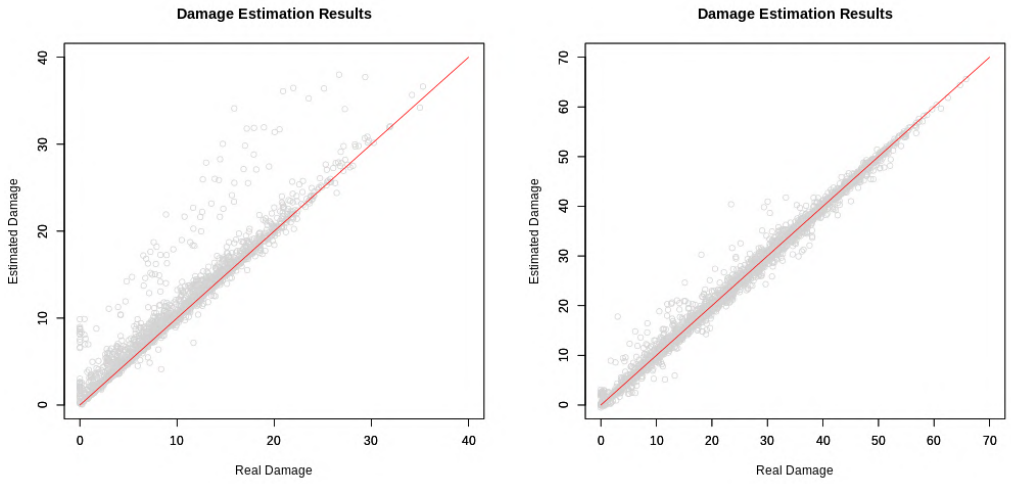


Figure 4.6: Validation damage estimation results for the Initial and Improved Rounds

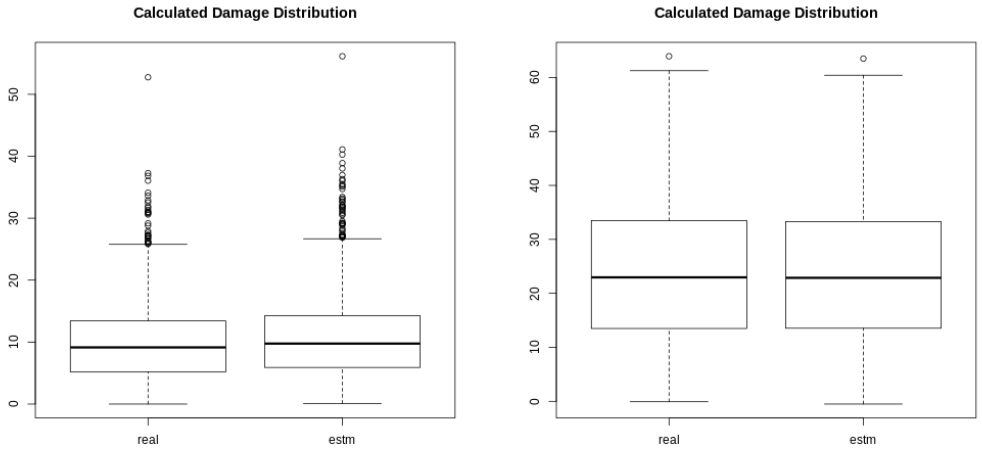


Figure 4.7: Test Set - Damage Distribution for the Initial Round

the test dataset in both the initial and improved stages. Figure 4.8 displays the results. A qualitative analysis of the results indicates that the distributions of the sets are similar, which reinforces the RMSE parameter results.

MEW 2012 Results

As previously mentioned, we also conducted predictions on another database containing different species from the ones used during training. We selected MEW 2012, which comprised 9745 images, and generated 38980 images with artificial random damage for this purpose.

The average estimated damage in MEW 2012 set is 5.05%, with a standard deviation of 4.43% and a maximum damage value of 37.90%. The real damage distribution average and standard deviation were 3.93 $\pm$ 4.39%, with a maximum

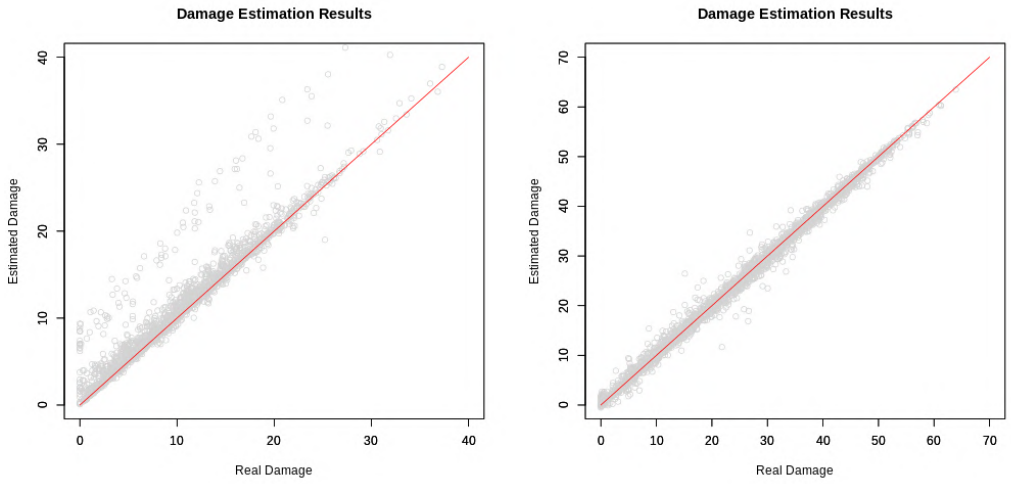


Figure 4.8: Test set damage estimation results for the Initial Round

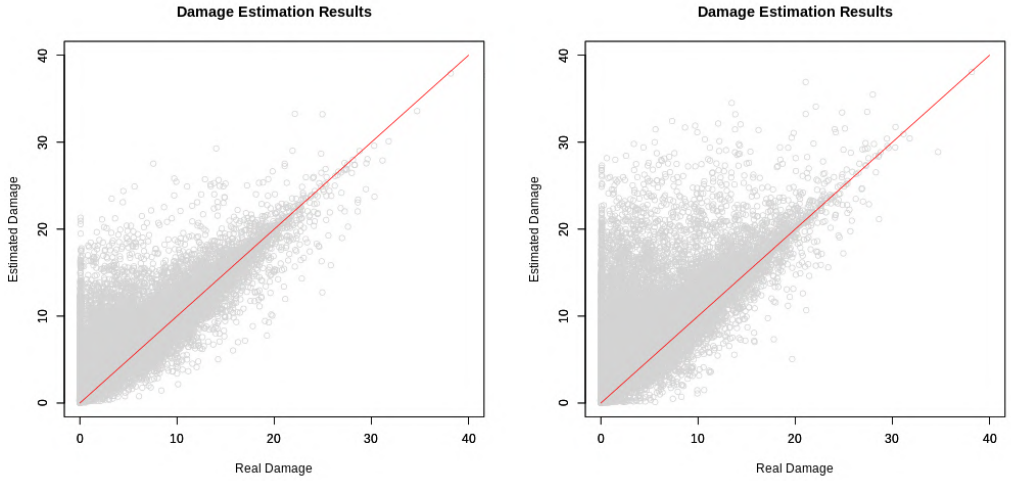


Figure 4.9: MEW 2012 set damage estimation results for the Initial and Improved rounds

value of 41.87%. The RMSE for this prediction was 1.76 ($\pm$ 3.02).

We also presented a graph comparing the obtained data with the ground truth. Figure 4.9 displays the results for the MEW 2012 dataset. As this dataset contained more species and samples, the distribution of predictions appeared wider during qualitative analysis. However, the RMSE result confirms that the prediction quality was similar, even with a dataset containing leaves from untrained species.

Shape Reconstruction results

To evaluate the shape reconstruction quality, we compared the network model's output with the ground truth initially generated or obtained from the datasets. We began by evaluating the distributions of the validation and test datasets and conducted a statistical analysis to determine if the predicted and original shapes

represented different populations based on their dice coefficient results distribution. The population distributions are presented in Figures 4.10, 4.11, 4.12, 4.13, and 4.14.

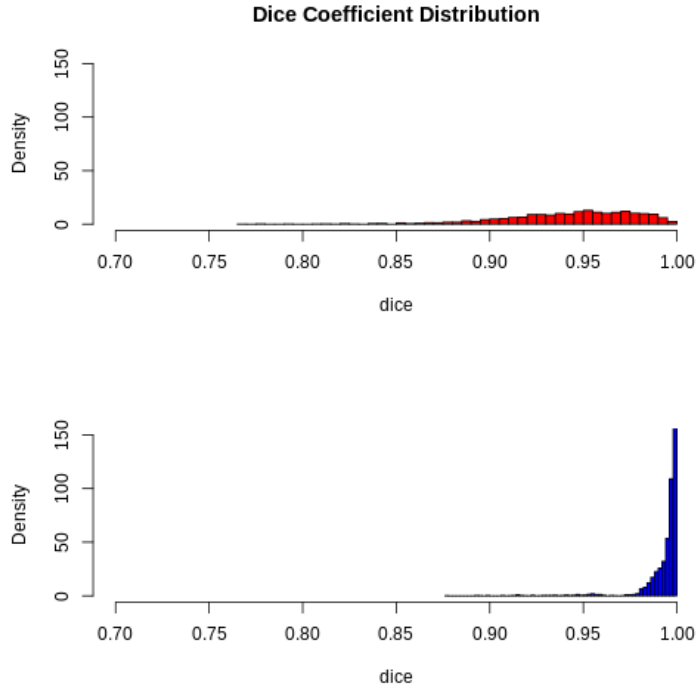


Figure 4.10: Dice coefficient distribution for the validation set - Initial Round

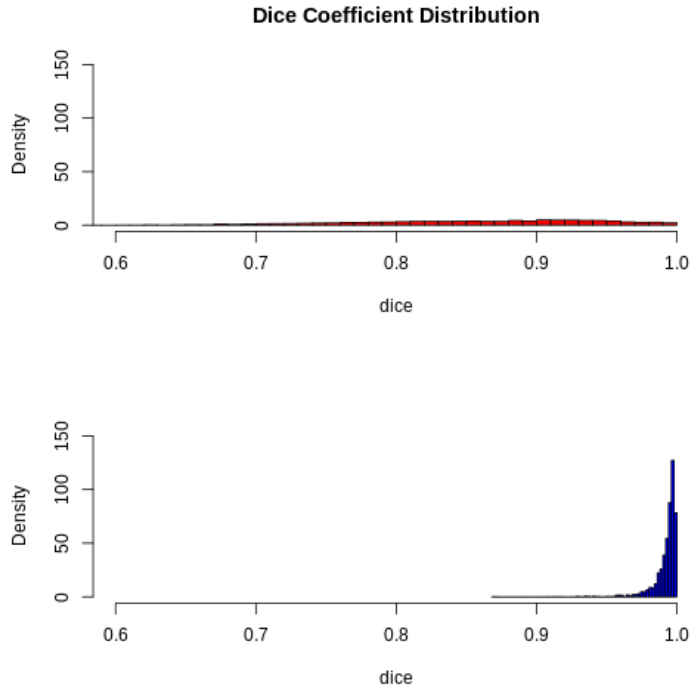


Figure 4.11: Dice coefficient distribution for the validation set - Improved Round

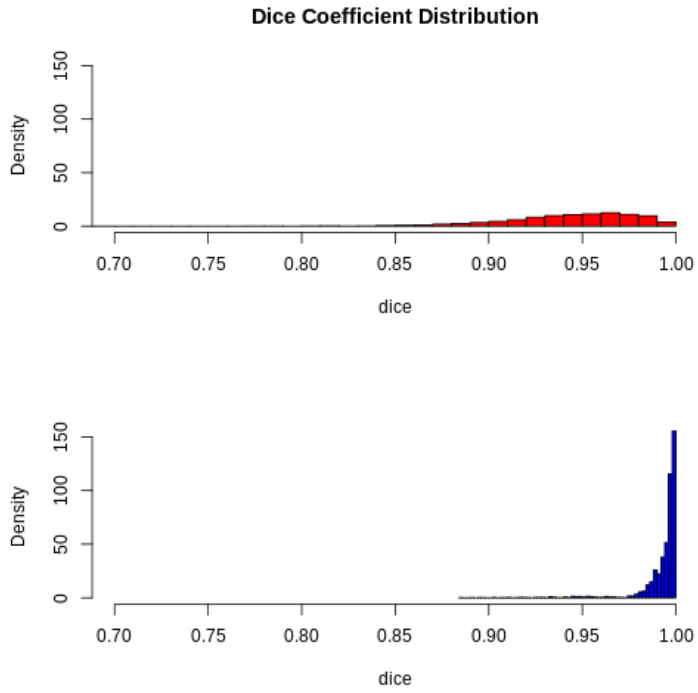


Figure 4.12: Dice coefficient distribution for the test set - Initial Round

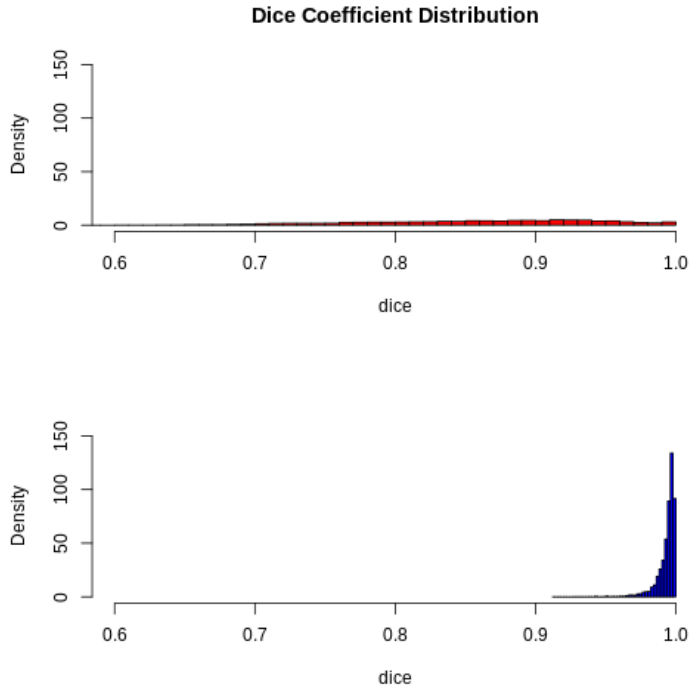


Figure 4.13: Dice coefficient distribution for the test set - Improved Round

In red, we plotted the dice coefficient comparing the damaged leaves with the original shapes, while in blue, we plotted the dice coefficient comparing the reconstructed leaves with the original shapes. The variances between the red and blue

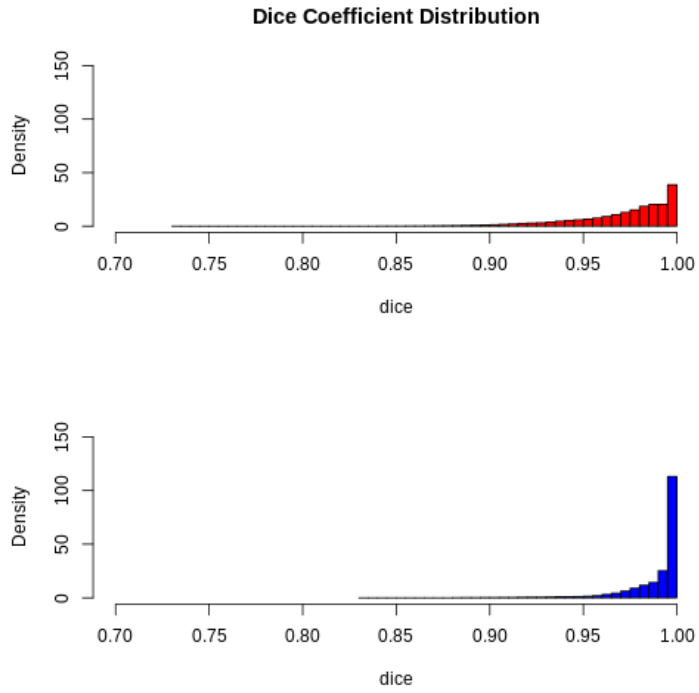


Figure 4.14: Dice coefficient distribution for the MEW 2012 set

populations were different. Therefore, we chose to apply Welch's t-test to compare the populations [180]. For all studied cases, the p-value was lower than 2.2×10^{-16}, indicating that the population means were not equal. In other words, the reconstruction process produced different shapes that were not caused by random events.

Regarding the reconstructed data, the average dice coefficient value for the validation set was 0.992 ± 0.008, while that for the test set was 0.993 ± 0.007. The worst-case values were 0.869 for the validation set and 0.912 for the test set. Finally, the average obtained from the MEW 2012 set was 0.988 ± 0.017.

4.1.10 Technical Evaluation: How to embed this solution?

In red, we plotted the dice coefficient comparing the damaged leaves with the original shapes, while in blue, we plotted the dice coefficient comparing the reconstructed leaves with the original shapes. The variances between the red and blue populations were different. Therefore, we chose to apply Welch's t-test to compare the populations [180]. For all studied cases, the p-value was lower than 2.2×10^{-16}, indicating that the population means were not equal. In other words, the reconstruction process produced different shapes that were not caused by random events.

Regarding the reconstructed data, the average dice coefficient value for the validation set was 0.992 ± 0.008, while that for the test set was 0.993 ± 0.007. The worst-case values were 0.869 for the validation set and 0.912 for the test set. Finally,

the average obtained from the MEW 2012 set was 0.988 ± 0.017.

1. Interest region identification;
2. Interest region segmentation;
3. Image binarization.

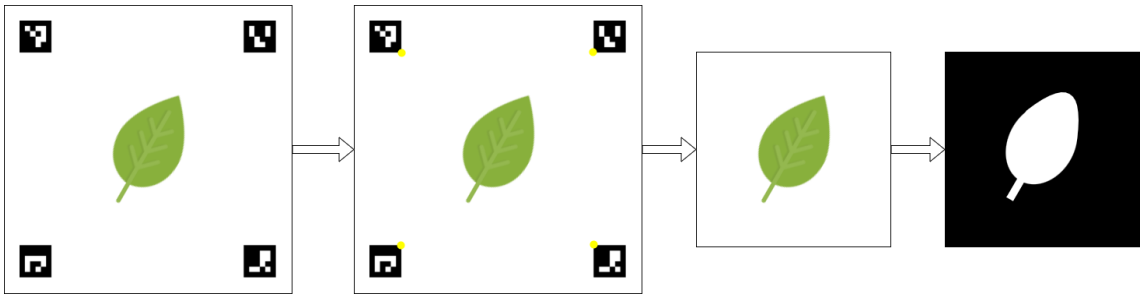


Figure 4.15: Complete segmentation pipeline proposal

The proposed pipeline is presented in Figure 4.15. The initial step in this process involves identifying the area where the user intends to place the leaf. This step is aided by the use of readily identifiable elements in the image, such as ArUco tags [181]. These tags are easily integrated into popular Computer Vision libraries and enable the identification of specific points.

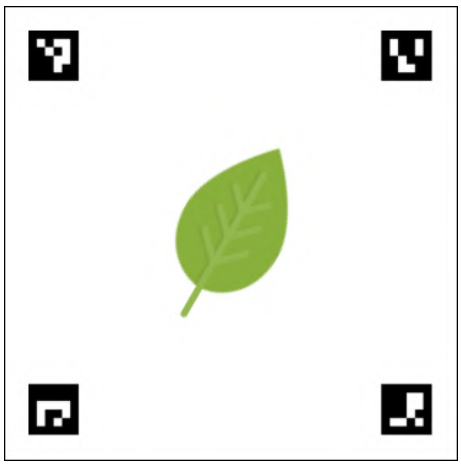


Figure 4.16: Illustration of the usage of ArUco tags to segment a map area.

We propose the use of four tags that delimit a square to aid in this process. The leaf should be positioned at the center of this square. This method helps to correct perspective issues that may arise due to camera tilt. An illustration of the proposed solution is presented in Figure 4.16.

To segment the region, the algorithm must identify the four tags and extract the desired coordinates from each of them. Then, a perspective transformation is

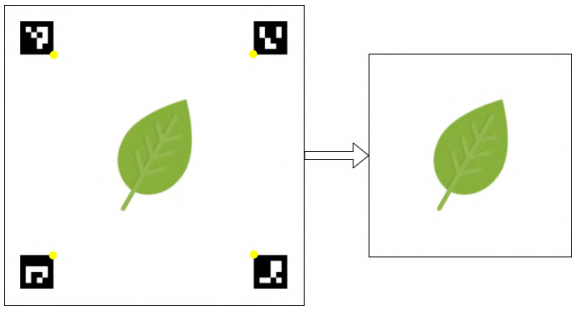


Figure 4.17: Region segmentation process illustration

performed to convert the plane into a square. The expected results after this stage are displayed in Figure 4.17.

After this stage, the algorithm obtains a square region with the leaf positioned above the background. At this point, we directly apply Otsu's binarization algorithm to the obtained segment. The expected result is displayed in Figure 4.18.

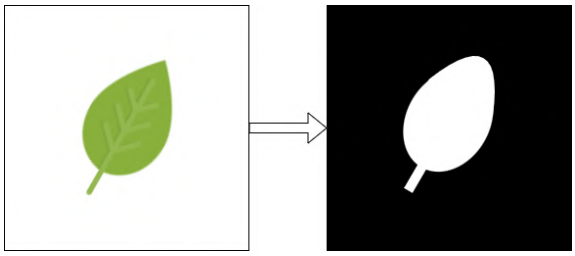


Figure 4.18: Binarization process illustration

We tested this pipeline under two different conditions. The first condition was a bench test. We produced the background according to the planned and provided a mockup leaf to test the segmentation process. Then, we also performed a small field test, in which we tried to capture and segment a leaf in the field using a camera and submitting it to this same process. Figure 4.19 displays the results for both tests. Our experiments displayed overall satisfactory results in both the bench and the field tests. The algorithm was able to segment the leaf from the background properly.

4.2 Evaluating and mapping diseases in forest canopies

The case study involves a triangulation approach, where three individual climbers conducted a cylinder-transect investigation. This method is akin to the recommendation made by Ribeiro, Basset, and Kitching [182] about density estimation. We employed this approach to develop the Edge AI system for identifying leaf diseases. The researchers initiate their approach by commencing at the highest point of the canopy and progressively descend downwards. They collect leaf samples along hori-

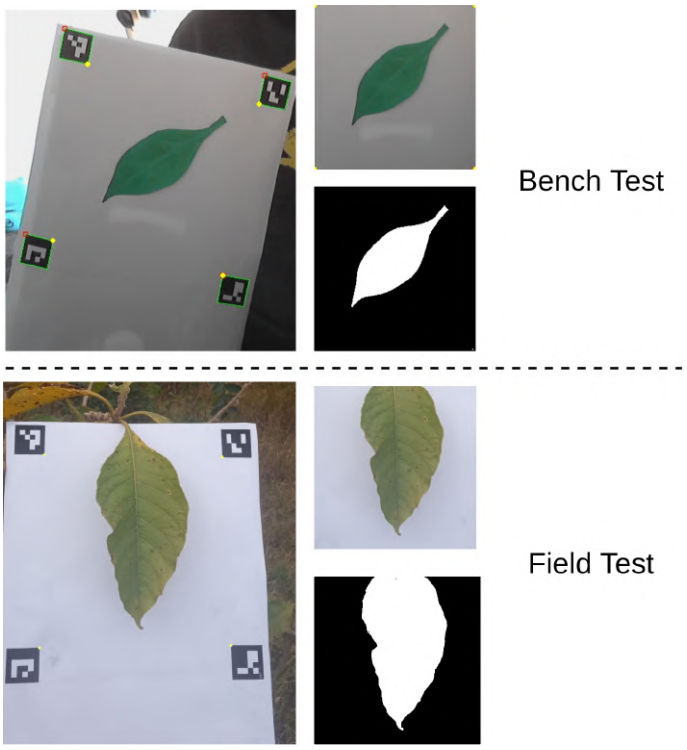


Figure 4.19: Experiments displaying the results of the proposed process. These experiments validate the usage of this technique to provide a mean to take this appliance onto the field.

zontal transects that are spaced at predetermined intervals until they reach the last stop. In this initial approach, to facilitate the division of data, we employ a backdrop template for sampling. The final destination is typically situated approximately 3 meters above ground level. The proposed methodology is depicted in Figure 4.20.

Leaf conditions are highly significant markers of ecosystem health, as previously mentioned. García-Guzman et al. [183] demonstrated that in Mexican wet forests, the prevalence of diseased leaves can reach 65% in highly infected areas, while it is only 2% in locations with low infection rates. Given this baseline, we anticipate that the disease will be dispersed in both high and low infection locations whenever a pathogen is present in a canopy. From this viewpoint, we simulate the transmission of disease by utilizing a probability density function (PDF) that is centered on the location with the largest percentage of infected individuals. Figure 4.21 depicts a visual representation of a density gradient determined by a centered maximum.

We assume that the distribution is Gaussian in shape, extending across the canopy. Therefore, the distribution can be represented by a probability density function that follows a geometric function based on the Gaussian distribution. The function is expressed in Equation 4.11. The function has the benefit of being able to represent the spread of the disease with only five parameters. The variable p_0 denotes the highest occurrence rate of the disease. The σ parameter, as always,

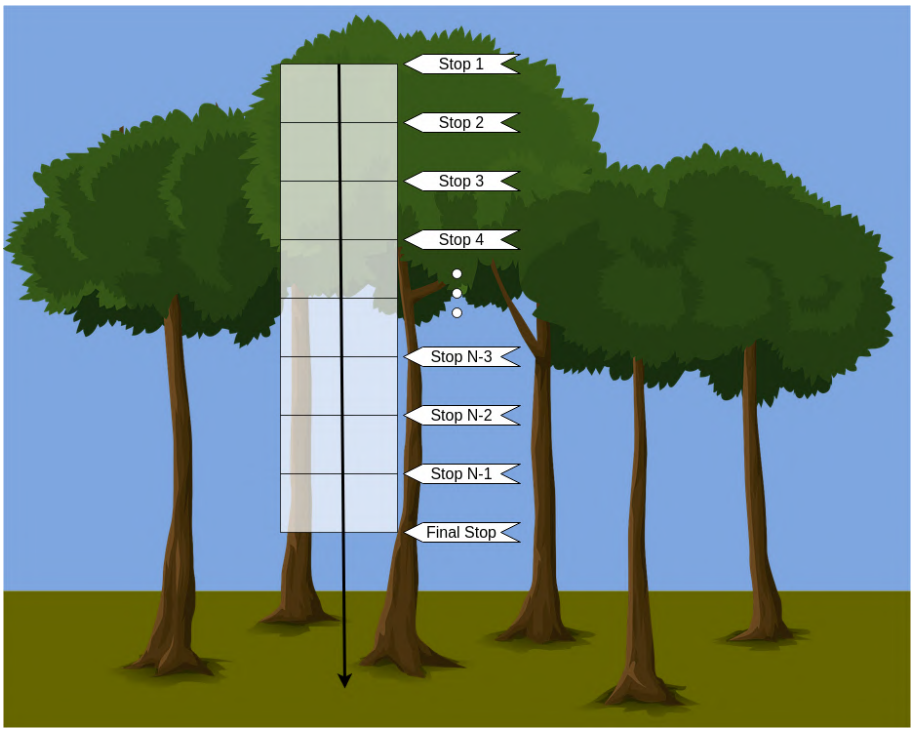


Figure 4.20: Illustration of the Cylinder-Transect study.

reflects the standard deviation. To simplify the analysis, we employed a uniform standard deviation across all three spatial dimensions. The (x_0, y_0, z_0) coordinates represent the central point of the distribution. The objective of this work is not to delve into the intricacies of the modeling process, but rather to present a case-study that is straightforward and can be easily replicated.

$$P(x, y, z) = p_0.e^{-\frac{(x-x_0)^2+(y-y_0)^2+(z-z_0)^2}{2\sigma}} \tag{4.11}$$

Several authors with prior experience in similar methodologies endorse the use of Gaussian-based models for disease transmission modeling. Soubeyrand, Enjalbert, and Sache [184] employed Gaussian-based modeling to simulate the circular diffusion of airborne plant disease. Pokharel and Deardon [185] conducted mathematical modeling of the transmission of infectious diseases using Gaussian distributions. In the midst of the COVID-19 pandemic, Ketu and Mishra [186] utilized Gaussian-based models to forecast the spread of the disease.

Despite the existence of similar methods in the literature, we have selected this modeling approach over Gaussian random processes (GRPs) or Gaussian process estimators (GPEs) due to its distinct essential features, as highlighted by some of the writers. The authors Soubeyrand, Enjalbert, and Sache utilized GRPs in their suggested model, as outlined in their publication [184]. The model involved the application of circular functions in a two-dimensional space to generate a preliminary representation. Our goal in this work is not to go into illness modeling. Therefore,

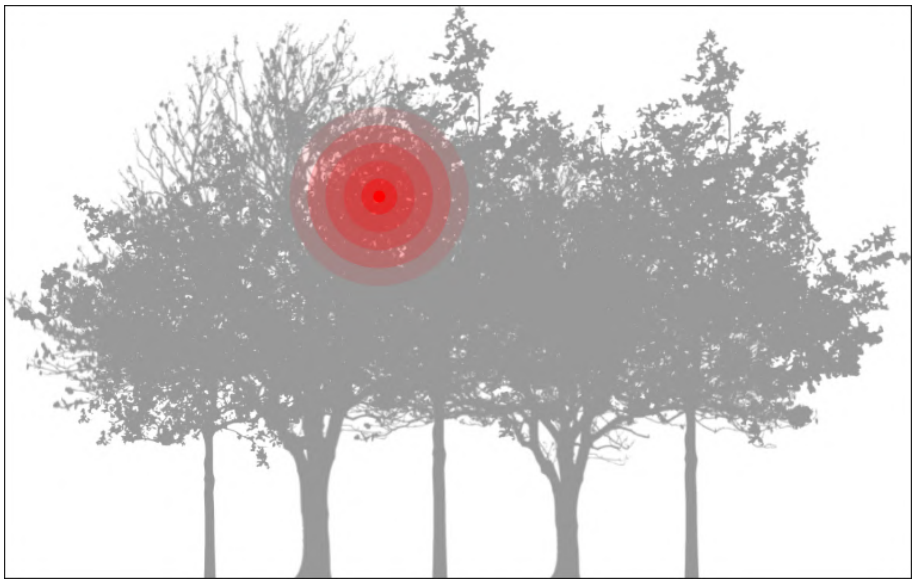


Figure 4.21: Example of a possible location for a disease spread. We model this spread using a spatially-distributed probability density function (PDF).

we have chosen to develop a simplified model that relies on a single spatial function. Pokharel and Deardon [185] suggest employing Gaussian process approximations to construct emulators (GPEs) for a two-dimensional dynamic disease spread model. The objective differs in that the authors aim to incorporate additional variables and processes that are not the focus of this study.

Researchers employing the cylinder-transect approach in the canopy acquire the spatial arrangement of damaged and healthy leaves using established coordinates. While the depiction may appear uncomplicated, the process of doing a regression from density points in three-dimensional space to a continuous-space function is not straightforward. Therefore, we suggest employing a heuristic approach to acquire the parameters that more accurately depict the original function.

4.2.1 Requirements

To begin this study, it is necessary to assess the prerequisites for the proposed approach. To address this issue, we provide a modified version of the co-design diagram shown in Figure 4.22, which is a simplified rendition of the diagram depicted in Figure 3.2b.

This representation displays the need to raise the constraints for the application and classify them into the hardware, software or architectural domain. The constraints identified for this matter are:

- As this system needs to be taken into the field, it needs to work for hours without a battery recharge or replacement. [*Hardware*].

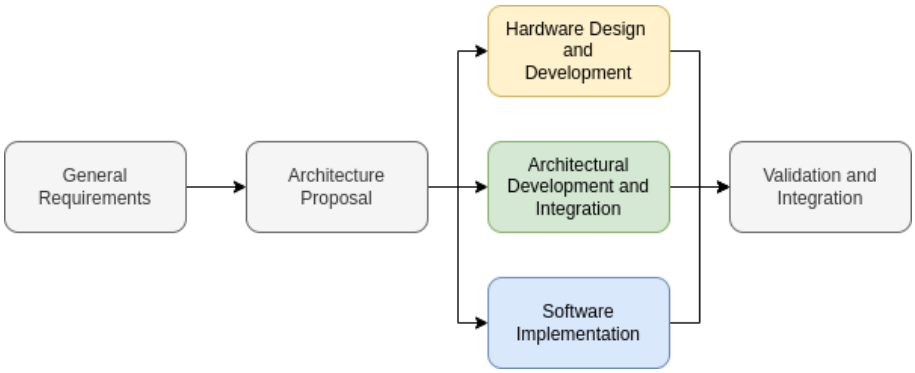


Figure 4.22: Simplified Co-design diagram.

- It needs to be robust enough to take hits from branches and falling seeds or nuts. [*Hardware*].
- As we propose a distributed system in a WBAN-Environment, both systems need to communicate with an application, working as web server nodes [*Architecture*].
- The communication needs to be efficient to stream the data through this local network [*Architecture*].
- The integrated architecture must present a mean to classify the leaves into diseased or healthy [*Software*].

In this case, we proposed a cooperative wearable system in which several users can input gathered data into a local wireless server which provides the AI using its hardware acceleration. This system is able to run both a traditional machine learning method and a convolutional neural network, according to the available resources.

4.2.2 General Architecture Proposal

The suggested method utilizes a wearable distributed system that operates in both Wireless Body-Area Network (WBAN) and Wireless Local Area Networks (WLAN). This system is designed to enable the extraction of information using techniques such as Data Fusion, Image Processing, and Computer Vision. The provided diagram, labeled as Figure 4.23, illustrates the suggested design for this system.

The initial component of this system is the tangible nucleus. Since we are suggesting a wearable system, it is crucial that it is inconspicuous and does not cause any interference or disturbance [187]. Consequently, we constructed the system using hands-free technology for the user. The helmet is the optimal choice for meeting the wearable and research needs in terms of its physical core. The paper by Silva et

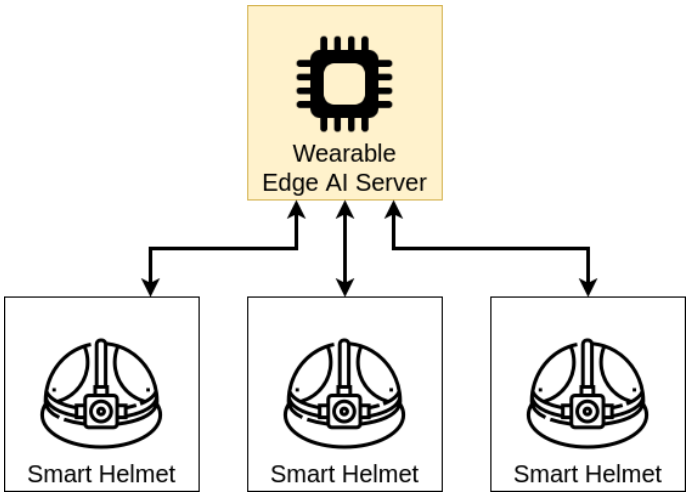


Figure 4.23: Proposed General Architecture. The smart helmets use the wearable Edge AI server to provide machine learning inferences.

al. [188] addressed the limitations imposed by certain appliances. Our main focus was to calculate the energy needs and usage for this particular system.

The subsequent component pertains to the fabrication of the sensor nodes that are based on the Internet of Things (IoT) technology. Every node is a Computer-on-Chip that has the ability to read sensor data, either single or numerous, process the data beforehand, and transmit it via either WBAN or WLAN. The selected sensors for enhancing environmental perception include a laser radar (LIDAR), a 9-Degree-of-Freedom Inertial Measurement Unit (9DoF IMU), and a conventional camera.

The Computer-on-Chip must possess the capability to access the necessary input/output ports from every sensor. Additionally, it is imperative that it possesses the capability to establish a wireless connection within the immediate vicinity of the body. Given the energy limitations of wearable systems, it is imperative that the Computer-on-Chip incorporates a processor with low power consumption. Therefore, ARM-based computer-on-chips are suitable solutions for this purpose.

Typically, the preferred devices to serve as primary applications for this system are ARM-based Computer-on-Chips. These devices should include various I/O ports to connect the necessary sensors, as well as a network card that can transmit the data via a local wireless connection.

We choose the Wi-Fi network standard (IEEE 802.11) as the interface for our WLAN/WBAN. The decision was made considering the simplicity of developing web server solutions, the speed at which data can be transmitted, and the coverage area for WBAN/WLAN. Furthermore, it relied on the wider bandwidth capacity to ensure the quality of the connection, particularly while handling camera streaming.

Each sensor node functions as a local webserver within this network. The ap-

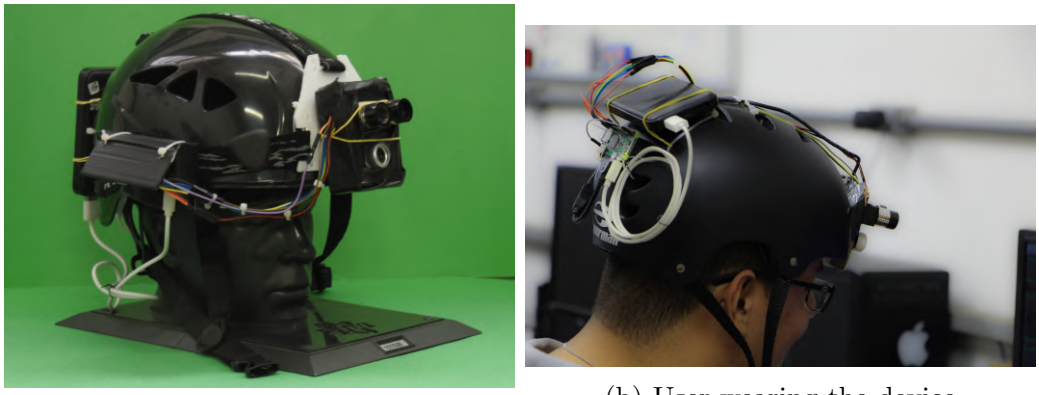
plication should explicitly request the sensor data from each node through the WBAN/WLAN. The application executes a data fusion technique. augments reality.

Hardware Specification

From the proposal, there are two main hardware element decisions: the *Smart Helmet Hardware* and the *Edge AI Node Hardware*. In order to develop the intelligent helmet, we required a flexible and verified solution that includes a built-in camera. Conversely, the stage of selecting Edge AI Node Hardware must effectively balance performance and portability.

Smart Helmet Hardware This project is a progressive development of a wearable gadget designed for the purpose of studying and monitoring the ecological environment. Additional publications were authored by members of the research group, featuring findings that were previously examined. The articles [188, 189] provide a detailed description of the hardware specification and its evaluation. The reproduction of the previously described evaluation is not the topic of this effort. Our objective is to include the suggested hardware into the Edge-AI architecture within this context.

This section offers a concise overview of the constructed wearable device and presents the necessary details and principles for the suggested integration. The hardware that has been created is a helmet consisting of sensors and a data processing unit. This tool was specifically designed for researchers and professionals who utilize the climb tree approach to gather data on natural habitats.



(a) Assembled Wearable Device

(b) User wearing the device

Figure 4.24: Prototype Assembled

The wearable prototype is depicted in Figure 4.24. Figure 4.24a displays a three-dimensional representation of the device, showing both the front and side views. Figure 4.24b depicts a user wearing the device as seen from the rear. The hardware

is equipped with a LIDAR sensor that is attached to a processing unit for accurately measuring the distance and estimating the 3D geometry of the objects being sensed. A Raspberry Pi Zero W was selected as the device for processing unit data, with power supplied by a 5V battery. During the development process, we took into account both the specifications for the wearable device and the needs for its use.

We utilized this prototype to do specific operations within this domain. Having already verified the sensing features of this system, our primary focus in this study was to conduct tests specifically evaluating its computational capability. Subsequent experiments have demonstrated that this prototype is capable of doing certain intended tasks, but its capabilities are restricted when confronted with more demanding processing requirements. Therefore, this viewpoint provides a rational basis for supporting the proposed Edge-based design. In the subsequent part, we will elaborate on the procedure of selecting the hardware for Edge Computing.

Edge AI server Node - Hardware selection and integration In addition to the smart helmet, another crucial component of the overall system is the Edge AI server node. Therefore, the hardware selection must take into account portable embedded systems that can facilitate machine-to-machine communication for this particular stage. Our case study focuses on implementing machine learning techniques within the setting of WBAN/WLAN to develop a viewpoint on Wearable Edge AI. This paper presents a comparison of the performance of four hardware components that have the capability to provide this utility.

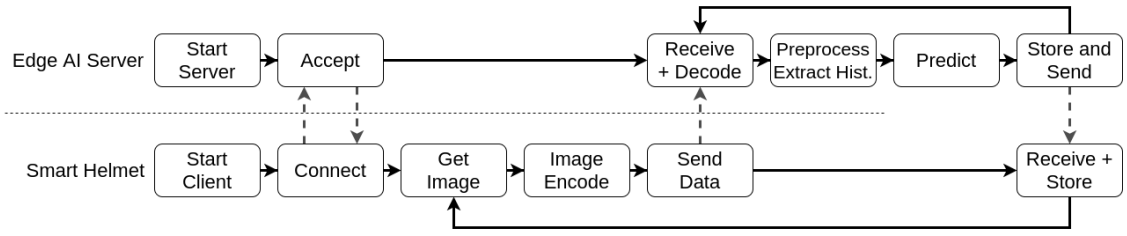


Figure 4.25: Edge AI service pipeline. In the proposed architecture, clients perform part of the processing, while the AI pipeline is provided by the Edge AI server node.

Initially, we established a pipeline that separates the components which are processed locally from those processed within the Edge AI server node. At first, the local systems obtain and convert the image, transmitting the converted data to the edge server. Within this server, the application initializes a trained machine learning model and continuously receives encoded frames. It then proceeds to decode, preprocess, and extract the pseudospectrum from these frames, thereafter evaluating it. The assessment outcome is thereafter stored by the Edge AI server and returned to the device for duplication. The diagram in Figure 4.25 illustrates the suggested pipeline for the Edge AI server, which is designed to accommodate a solitary client.

We evaluated multiple devices for developing the solution. In the context of this Wearable Edge AI solution, we evaluated the Raspberry Pi Zero W, Raspberry Pi 3B, Raspberry Pi 3B+, and Jetson Nano platforms as potential options for delivering Edge AI functionality. These solutions are all commercially available ARM-based computer-on-modules.

Edge AI Software

As stated in the introduction, the condition of leaves serves as crucial indicators of the overall health of the ecosystem. Therefore, we opted to assess the limitations of an Edge AI component that conducts leaf classifications into two categories: "normal" and "diseased." This section delves into the implementation of Edge AI software as demonstrated in the case study.

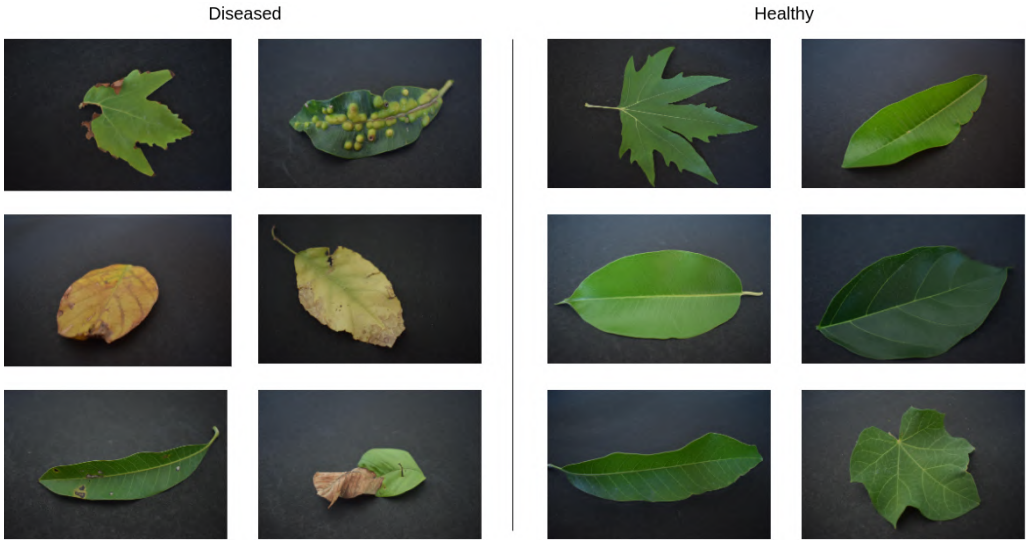


Figure 4.26: Sample of healthy and diseased leaf images obtained from the dataset.

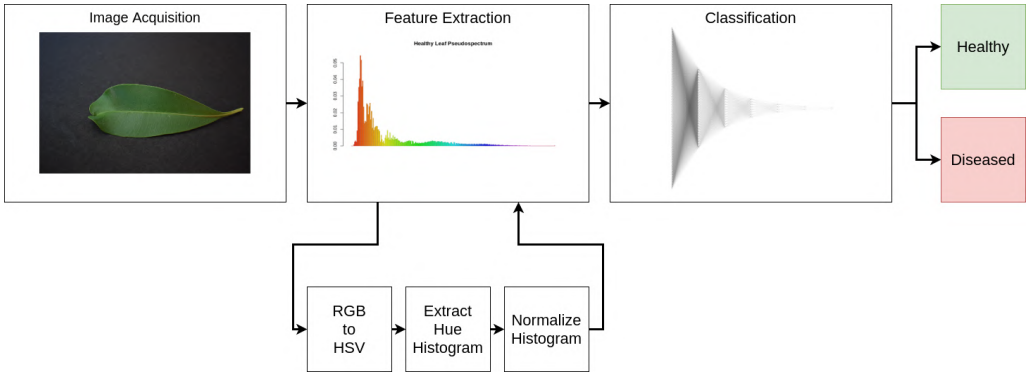


Figure 4.27: Data processing pipeline and associated substages. For the image extraction, the associated stages are the color space conversion and histogram extraction.

To begin, it is essential to locate a dataset that contains comparable information

to the required dataset. After conducting thorough research, we have chosen to utilize the dataset provided by Chouhan, Kaul, and Singh in their publication on leaf diseases [190]. The authors provide a dataset consisting of 4,503 photos depicting leaves exhibiting both healthy and diseased conditions. Out of this collection, there are 2,278 photographs depicting healthy leaves, whereas 2,225 images show damaged leaves. There are 12 distinct species represented by the leaves. Figure 4.26 showcases photos of both healthy and diseased leaves extracted from the original dataset.

The designers of this database ensure a clear distinction between infected and healthy leaves. Any variation in hue and texture is solely attributable to illness in all instances. The photos had a pixel resolution of 6,000x4,000. In order to enhance the test's speed and align the resolution with that of commonly used cameras in embedded systems, we reduced the resolution to 900x600 pixels. The improved resolution is equivalent to 15% of the original size.

The data extraction and classification method adheres to a traditional pipeline. Initially, we obtain the image and then proceed to extract a feature vector. Next, we employ a machine learning model to categorize the image based on its attributes, resulting in a binary classification outcome. The pipeline is depicted in Figure 4.27, illustrating the substages linked to each primary stage.

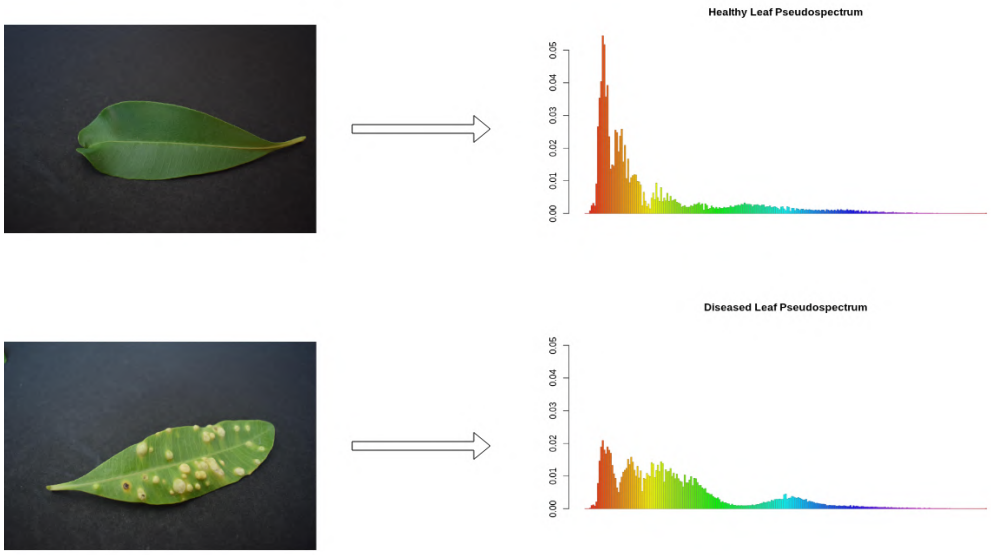


Figure 4.28: Pseudospectrum extraction samples

Within this particular framework, our method to addressing this issue involves the development of a pseudospectral analytic system [53]. In this process, the feature vector is a *pseudospectrum*, which is obtained by extracting the histogram from the Hue channel in the HSV color space. Ultimately, the histogram is divided by its total sum, resulting in a probability density function (PDF) that represents the distribution of colors. This PDF is then referred to as the pseudospectrum. Figure

4.28 illustrates several instances of the extraction of pseudospectra.

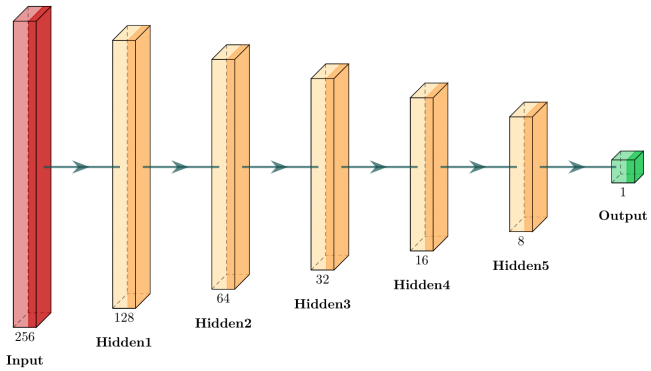


Figure 4.29: Neural network representation. The chosen model was a Multi-Layer Perceptron (MLP). All layers are fully connected. The number beneath the blocks represents the number of neurons in each layer.

Subsequently, a neural network is employed to categorize the leaf. In this instance, we employed a conventional Multi-Layer Perceptron (MLP) model to carry out the computations. Despite the model's lack of novelty, we selected it due to its simplicity, which results in improved performance on low-power devices in Edge Computing. Despite its simplicity, the model has demonstrated intriguing outcomes in the classification of citrus fruits [53]. Within this particular framework, we employed a network that received 256 inputs derived from the pseudospectrum. The network's hidden layers consisted of 128, 64, 32, 16, and 8 neurons, respectively. The result yielded a binary categorization of either healthy or diseased. Figure 4.29 presents a concise representation of the network's structure.

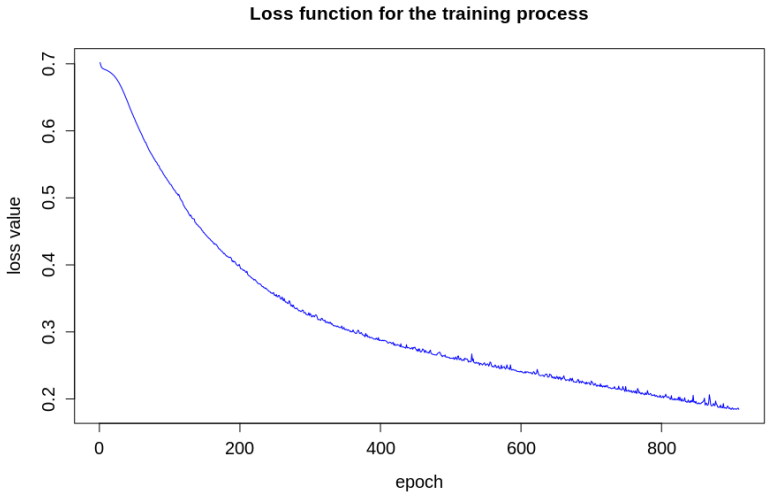


Figure 4.30: Loss function during the training process

The *scikit-learn* framework was employed to construct our model [191]. The training was conducted using a backpropagation technique, employing a cross-entropy

loss function. For the purpose of training, we partitioned the original dataset of photos into two distinct subsets. During the initial phase, we employed a random selection process to choose 10% of the photos depicting both damaged and healthy leaves from each species. These selected images were then used to create a test dataset. The remaining 90% constituted a training set. Our system was trained using 90% of the photos from the training set, while the remaining 10% were used for validation. The behavior of the cross-entropy loss during the training is depicted in Figure 4.30. This training concludes when the cross-entropy loss value does not improve by more than 10^{-5} for ten consecutive epochs, as determined by an arbitrary convergence criteria.

In addition to this model, we also evaluated the feasibility of utilizing a convolutional neural network (CNN) model for making predictions on the embedded hardware. This method is more contemporary, although necessitates a greater amount of computer capacity. Therefore, we put forward a test that assessed two aspects:

- How much improvement can a CNN obtain over an Computer Vision and MLP.
- How much performance the embedded system loses using this method over a traditional approach.

We created a simple CNN model that approaches this process. Figure 4.31 displays an illustration of this model.

This model utilizes five 2D-convolutional layers in the feature extraction stage. The layers have 16, 32, 64, 64, and 64 filters of size 3x3, respectively. Following each convolutional layer, there is a further max pooling layer with a 2x2 size. Following these phases, the output is compressed and sent through a dense layer consisting of 1024 neurons. So far, the convolutional and dense layers have employed a rectified linear unit (ReLU) activation function. Ultimately, the result is a solitary neuron that utilizes a sigmoid activation function. The loss function employed was the cross-entropy. The function was trained for a total of 12 epochs, a value determined empirically as the maximum number of epochs to prevent overfitting indications. In addition, we conducted tests on this model based on the hardware and software performance indicators in order to address the problems that were highlighted.

4.2.3 Validation Tests

Once we have put forth the hardware, software, and architectural components, it is essential to set metrics to verify the effectiveness of each stage. Within this area, we present the modeling and metrics utilized to assess each individual component. We assessed the performance of the Edge AI server across different platforms, focusing on the hardware components. We analyzed the metrics of the machine learning

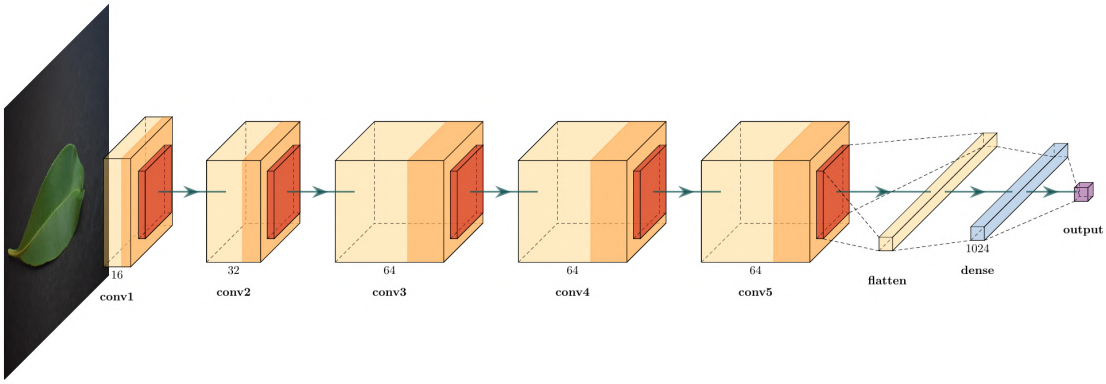


Figure 4.31: Proposed CNN model. The convolutional layers have 3x3 filters, with 2x2 pooling. The output is a single value obtained from a sigmoid activation function.

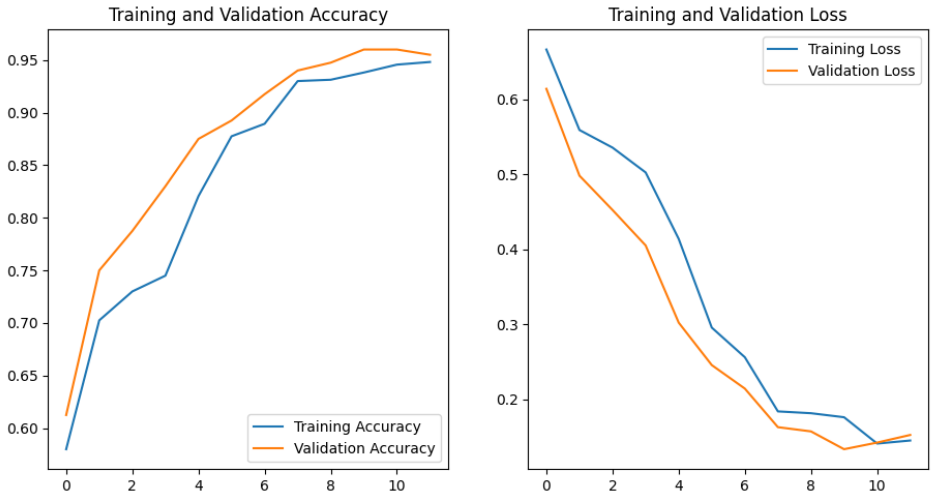


Figure 4.32: Values for accuracy and loss functions in the CNN training process.

software predictions for the suggested application in order to evaluate its software features. Regarding the design, we analyzed the timing limitations for numerous clients linked to the Edge AI server, taking into account a Real-Time Quality of Service (QoS) assessment.

Hardware Validation Tests

This appliance consists of two primary hardware components: a Smart-Helmet and an Edge AI server node. The helmet was originally designed to provide versatile applications in this field. Hence, the validation stage deems it imperative to validate the newly introduced component: the Edge AI node.

In Figure 4.25, we delineated the specific functions carried out by both the intelligent helmet and the Edge AI node. Initially, we must assess the hardware

components for each of the recommended solutions based on the distributor websites associated with them. The available options include of the Raspberry Pi Zero W, Raspberry Pi 3B, Raspberry Pi 3B+, and Jetson Nano. Table 4.2 presents the most relevant aspects about each solution.

Table 4.2: Hardware Specifications for the Edge AI server node candidates.

	Raspberry Pi Zero W	**Raspberry Pi 3B**	**Raspberry Pi 3B+**	**Nvidia Jetson Nano**
CPU	1x ARM11 @ 1GHz	4× ARM Cortex-A53 @ 1.2GHz	4× ARM Cortex-A53 @ 1.4GHz	4x ARM Cortex-A57 @ 1.43 GHz
RAM	512 MB	1GB	1GB	4GB
Storage	MicroSD card	MicroSD card	MicroSD card	MicroSD card
Nominal Power	5V over microUSB (max. 6W)	5V over microUSB (max. 12.5W)	5V over microUSB (max. 12.5W)	5V over P4 Jack Barrell (max. 5W/20W modes)
Network Platform	2.4GHz 802.11n	2.4GHz 802.11n	2.4GHz/5GHz 802.11b/g/n/ac	2.4GHz 802.11n (over USB)

To conduct this test, we execute the tasks outlined in the pipeline depicted in Figure 4.25 for every candidate. We assess the time delay required to complete all internal phases for each solution, executing identical code to receive input from a client, make predictions about the outcome (whether it is healthy or unhealthy), return the predictions, and store them in a text file. During the hardware evaluation, we just focus on the latency of the steps that are executed locally. The components of the system that rely on networking will be executed at a later time, taking into account the characteristics of the architecture. Additionally, we conducted a comparison of the two software techniques by testing the ratio of average predictions per second. This addresses one of the inquiries posed in the software proposal, namely evaluating the efficacy of the model on the embedded hardware.

Software Validation Tests

The unique software being presented is a machine learning-based prediction system that operates on an Edge AI server node. To address this issue, we utilized a Multilayer Perceptron (MLP) neural network model to accurately forecast whether leaf images exhibit signs of disease or are in a healthy state. Furthermore, we conducted an evaluation of the same measures for the CNN model in order to validate the enhancement in this characteristic by employing a more contemporary approach.

In order to successfully validate the program, it is imperative that we thoroughly comprehend the performance of this machine learning model in relation to the specific data at hand. Therefore, we employ conventional machine learning metrics to examine the data. We evaluate the confusion matrix, as well as the *Precision*, *Recall*, and *F1-Score* metrics. The subsequent equations illustrate the formulas for various measures. The above equations define the variables as follows: TP represents the

count of true positives, FP represents the count of false positives, TN represents the count of true negatives, and FN represents the count of false negatives.

$$Precision = \frac{TP}{TP + FP} \tag{4.12}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.13}$$

$$F1\text{-}Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4.14}$$

Architecture Validation Tests

It is necessary to take into account characteristics that assess the individual and overall performance of the proposed scenario for the validation tests of the architecture. Therefore, we devised an experiment that was specifically designed as a Real-Time Quality of Service (QoS) test, drawing inspiration from previous research on IoT and Wireless Sensor Networks [72, 192], with the aim of assessing the real-time constraint. This assessment assesses the proficiency in completing a series of tasks, taking into account both individual and network-related circumstances.

At first, we consider duration as discrete intervals, as the set $D = d_i, i \in \mathbb{N}$, where $d_{i+1} - d_i = \theta$, and θ is a constant sampling time. The soft real-time deadline will be represented by ϕ, where $\phi = k \times \theta, k \in \mathbb{N}^*$. Thereby, we establish the following definitions:

Definition 1. Let $D = d_i$ be the finite set of nodes performing IoT-dependant tasks, where $i \in \mathbb{N}$;

Definition 2. Let $E = e_i$ be the finite set of events that each node performs, where $i \in \mathbb{N}$;

Definition 3. Let $L = l_{g,e}$ be the length of time interval that the node g takes to perform an event e during the execution, where $g \in G$ and $e \in E$;

Definition 4. Let $P = p_i$ be the set of patterns of events to be observed in the devices, where $p_i = E_i$, $E_i \subset E$ and $i \in \mathbb{N}$. In this case, all client devices will perform the same events in the same pattern;

Definition 5. Let $O = o_i$ be the finite set of observations of a certain pattern $p_i \in P$ on each device;

The equation that represents the elapsed time λ to observe a particular pattern $p_i \in P$ is:

$$\lambda_{o_i} = \sum l_{g,e_k} | \forall e_k \in o_i, o_i = O_{p_i} \tag{4.15}$$

All client devices in the network composition will have the same ϕ soft real-time deadline. Given this equation, let $\hat{O}$ be a subset of O, where $\lambda_{o_i} \leq \phi$, $\forall o_i \in \hat{O}$. Finally, given the sets O and $\hat{O}$:

Definition 6. Let N be the number of elements on the set O;

Definition 7. Let N_h be the number of elements on the subset $\hat{O}$;

The quality factor Q_f will be represented by the following equation:

$$Q_f = \frac{N_h}{N} (\times 100\%) \tag{4.16}$$

This result represents how often the nodes execute a pattern of events without violating the soft real-time constraints. The clients represent the smart-helmets and will send data to the Edge AI server node in parallel on each test.

Case Study Validation for Deployment

In order to verify the system in the case study, we employed a test that relied on the probability distribution function outlined in Equation 4.11. The equation represents the likelihood of encountering damaged leaves during a sample procedure near the tree, as determined by the spatial coordinates. The function's maximum value is determined by the P_0 value, and the spatial location of the disease's epicenter is given by the (x_0, y_0, z_0) coordinate. The situation is depicted in Figure 4.21.

For this validation test, we examine a group of three climbers who collect leaf samples at different heights inside certain areas of the canopy called transects. The whereabouts of the three climbers are arbitrary but known. Additionally, the locations of the stops along the transect are known, enabling the mapping of their positions as three-dimensional points during the procedure. The researchers should be positioned within a random radius from the center of the tree trunk in the canopy to achieve a more optimal spatial dispersion. The organization for a 5-meter-radius and 9 stops is depicted in Figure 4.33.

We examined a procedure that allows for an arbitrary number of stops during the ascent, in accordance with the requirements of the transect method. During each stop, the researcher would collect samples of leaf images. The system autonomously categorizes the sampled leaves as either healthy or unhealthy based on the acquired methodology. Consequently, for every (x, y, z) coordinate at which a researcher

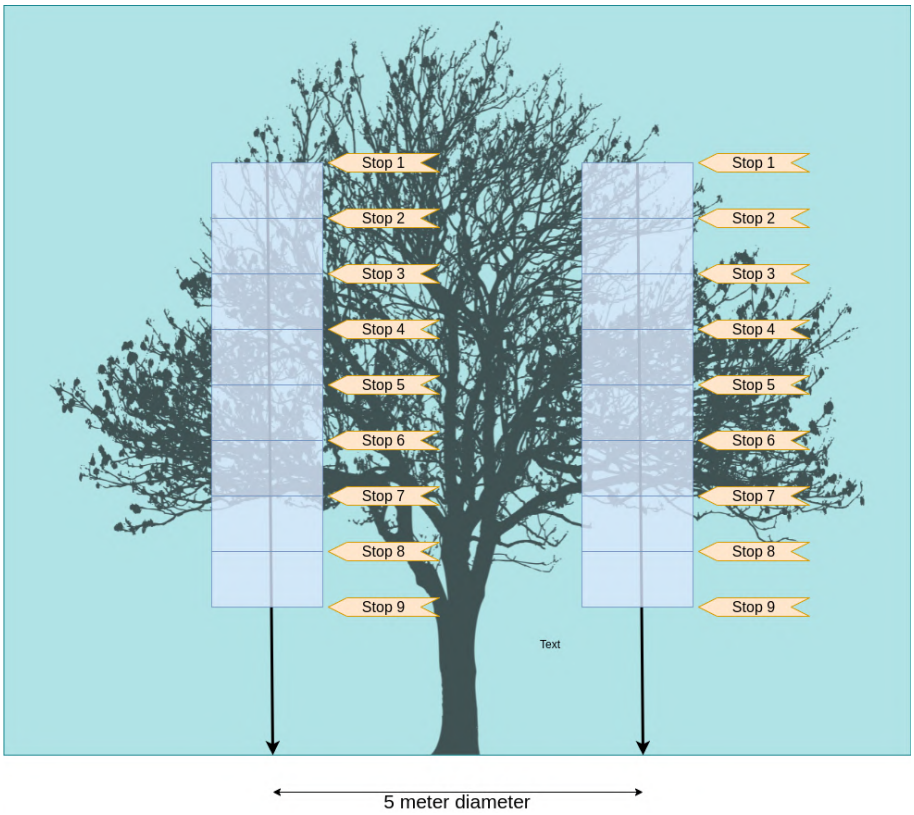


Figure 4.33: Sampling process illustration

employs the helmet and a backdrop template to sample the leaves, the system is capable of computing the proportion of unhealthy entities.

For climbers to take tiny goods to the canopy, we suggest using a background template. This template is a sturdy object with identification tags placed along its edges. This suggestion enables the algorithm to circumvent any influence from the background during the sampling procedure. We created a preliminary examination to showcase this problem utilizing the wearable camera. The prototype captures the image, detects specific tags within the backdrop template, applies a four-point transformation to isolate the region of interest, and utilizes Otsu's binarization technique to segment the image. The pipeline utilized in this investigation is depicted in Figure 4.34.

Using this data, the system does a regression analysis to fit the probability density function (PDF) outlined in Equation 4.11. To address this issue, it is necessary to acquire the parameters contained in the tuple $T = (p_0, \sigma, x_0, y_0, z_0)$. We opted to employ an Evolutionary Algorithm for the execution of this task, whereby the tuple candidates are regarded as the genotype and the mean squared error serves as the fitness function. The decision was made considering three primary factors:

- *Ease of use:* it is easier to perform regression for a smooth parametric arbitrary three-dimensional distribution function with an evolutionary algorithm than

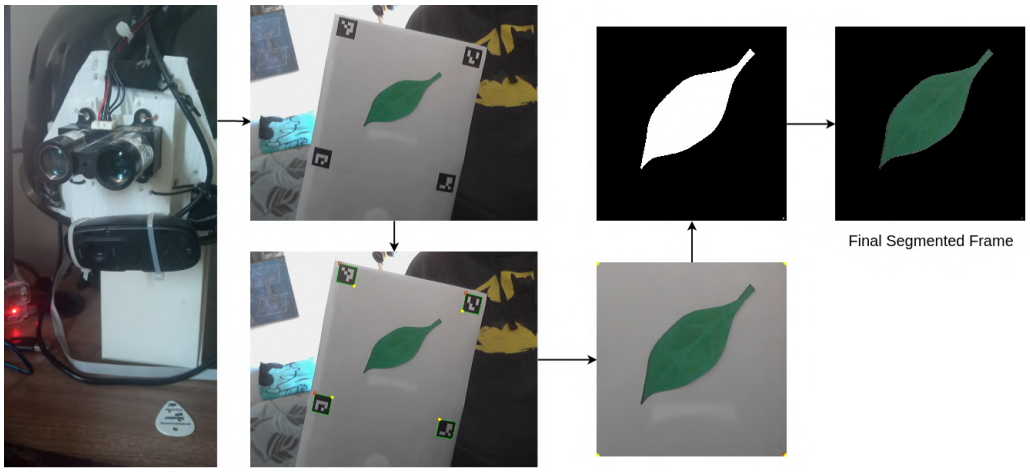


Figure 4.34: Demonstration of the segmentation process. The prototype used a USB camera to capture the data, which can be processed by the prototype itself or in the Edge AI server node.

designing an interpolation based in various parameters and kernel functions;

- *Flexibility:* the same process can be used to obtain a regression to any parametric model by just changing the input parameters on the same algorithm;
- *Robustness:* The regression algorithm displayed robust results, even with a change on its parameters.

In this case, the climbers would make nine stops, collecting 200 leaves at each stop. The system autonomously categorizes every leaf, transmitting data regarding each position and the compressed image of the leaf. To simulate the sampling process, we randomly selected 100 photos from the original dataset. Our selection approach utilizes a random number and evaluates its compatibility with the probability density function (PDF) defined by Equation 4.11 using arbitrary parameters. The value of p_0 was selected as 0.65, taking into account the highest occurrence of sick leaves observed in the study conducted by García-Guzman et al. [183]. Furthermore, we established the coordinates of the tree stem and ground as the origin $(0, 0, 0)$ and deliberately chose $(2, -2, 8)$ as the epicenter of the sickness. Ultimately, the value of our standard deviation (σ) was 5. Therefore, the ultimate random PDF for this exam is:

$$P(x, y, z) = 0.65.e^{-\frac{(x-2)^2+(y+2)^2+(z-8)^2}{10}} \tag{4.17}$$

Ultimately, the system categorizes and archives the data relative to the image. Our intention is to utilize the saved data to conduct an analysis that will yield the original PDF values through the implementation of an evolutionary algorithm. The Edge AI node is capable of conducting this analysis to offer on-site insights derived

from the collected data. The goal is to approximate the original values of Equation 4.17 as accurately as possible. Figure 4.35 illustrates the spatial distribution of the disease in the given arbitrary function. The likelihood of encountering diseased leaves at this position increases as the red circle becomes larger and more vibrant in color. The brown stick denotes the location of the primary tree trunk.

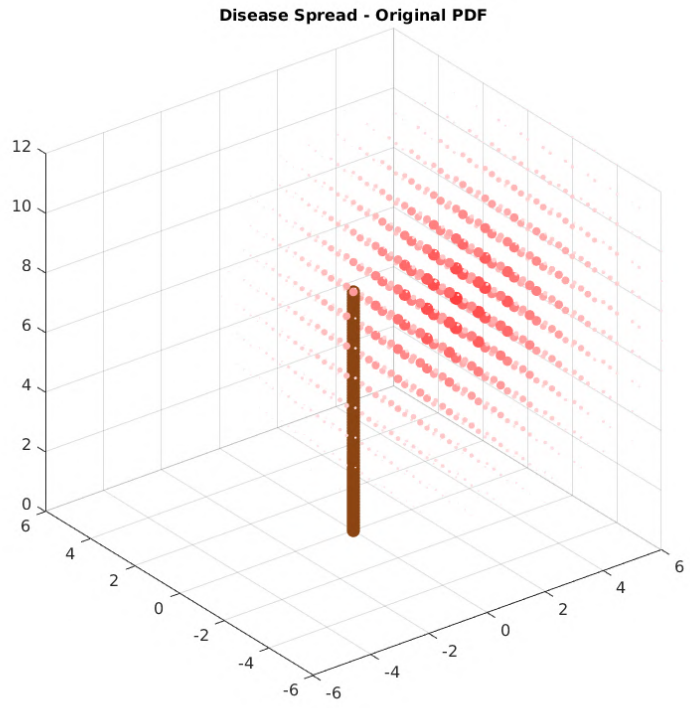


Figure 4.35: Arbitrary PDF display. The larger and more colorful red dots have a bigger probability density. The brown cylinder represents the main tree trunk.

4.2.4 Results

The preceding part provided a comprehensive overview of the hardware, software, and architecture components utilized in the proposed case study. In addition, we presented the assessment criteria employed to validate each branch of the co-design reviewed pattern. Ultimately, we deliberated on the verifications required for an application in the given case study. This section presents the results received from experiments conducted on the proposed elements.

Hardware Validation Tests

We present the hardware specifications for the Edge AI server node. All candidates are COTS computer-on-modules. To validate the hardware candidate, we assessed the candidates' performance in executing the internal Edge AI tasks. The pipeline

for the proposed test is illustrated in Figure 4.36. The internal duties for the Edge AI server are categorized into three stages: (i) preprocessing and extracting the feature vector, (ii) predicting the leaf condition, and (iii) storing the prediction data.

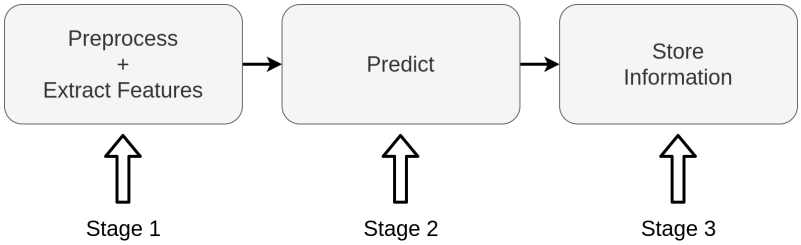


Figure 4.36: Pipeline for the hardware validation test.

We conducted the tests on all the candidates listed in Table 4.2. We conducted tests on the Jetson Nano in both the 5W and 20W power modes. We executed the subsequent pipeline on all 437 photos from the test set. The hardware candidates and configurations will be referred to as follows: *Zero W* (Raspberry Pi Zero W), *3B* (Raspberry Pi 3B), *3B+* (Raspberry Pi 3B+), *Jetson 5W* (Jetson Nano operating in 5W mode), and *Jetson 20W* (Jetson Nano operating in 20W mode).

The Zero W was evaluated as it serves as the computer in the helmet prototype. The *3B* and *3B+* models were evaluated due to their lower pricing compared to the Jetson Nano, although having similar processor configurations. Using the *Jetson 5W*, our goal is to compare the candidate with the highest cost but with limited hardware capabilities. In this economic operation mode, the operating system disables half of the CPU cores to conserve power. Ultimately, our objective was to confirm the disparity in performance between the priciest gear in the most powerful operational state and the performance of the other contenders.

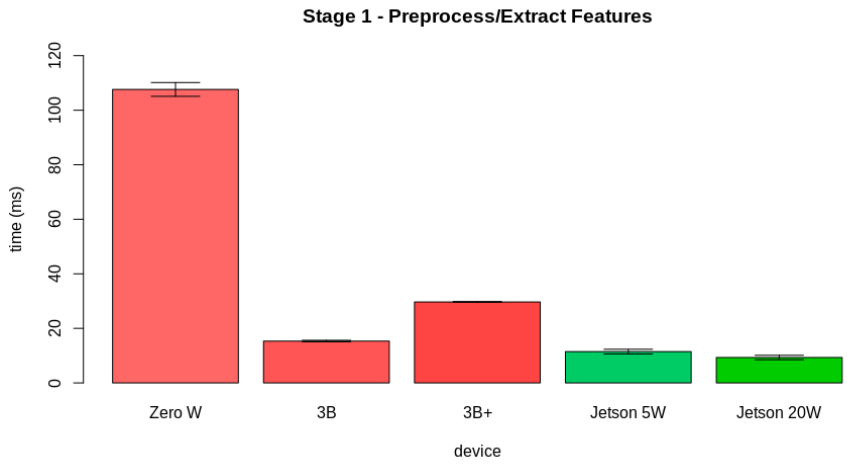


Figure 4.37: Latency results for the first stage.

Initially, we assessed the outcomes for Stage 1. During this stage, the hardware

does preprocessing on the image by converting its color space from RGB to HSV. Next, it retrieves the pseudospectrum from the Hue channel. The latency evaluation findings from the first stage are presented in Figure 4.37. The *Zero W* device completed the first stage in 107.61 ± 2.53 ms. The *3B* device took 15.34 ± 0.28 ms to perform this task. The *3B+* device took 29.69 ± 0.13 ms to complete this stage. The *Jetson 5W* device took 11.47 ± 0.87 ms to perform this part, while the *Jetson 20W* device took 9.32 ± 0.81 ms.

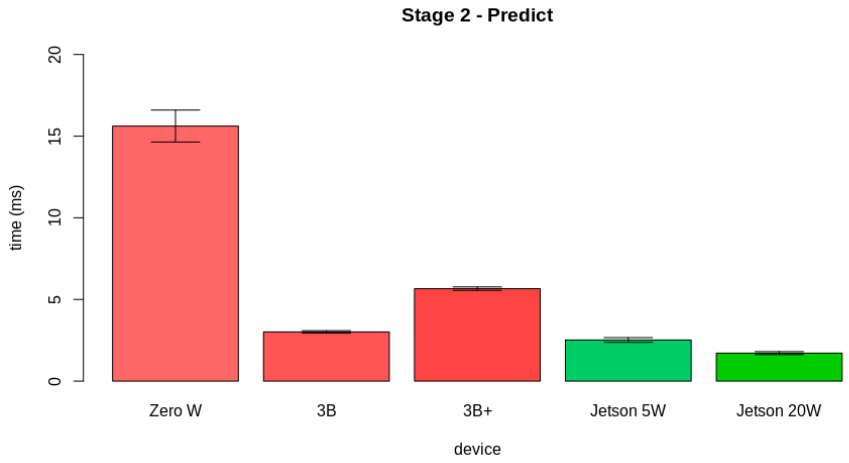


Figure 4.38: Latency results for the second stage.

Then, we assessed the results for the Stage 2. This stage corresponds to the prediction of the leaf condition using the model. Figure 4.38 presents the results obtained from the evaluation of the latency from the second stage. *Zero W* took 15.62 ± 0.98 ms to perform the first stage, *3B* took 3.01 ± 0.07 ms to perform this task, *3B+* took 5.66 ± 0.10 ms to perform this stage, *Jetson 5W* took 2.52 ± 0.14 ms to perform this part, and *Jetson 20W* took 1.71 ± 0.09 ms.

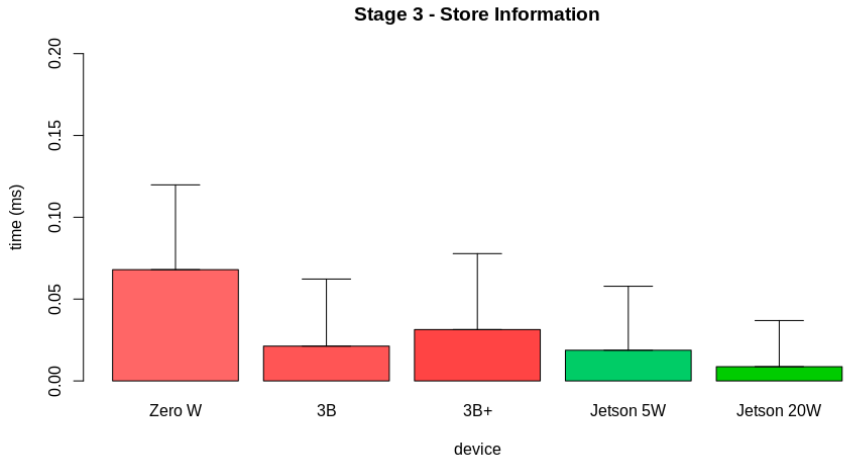


Figure 4.39: Latency results for the third stage.

At last, we analyzed the outcomes pertaining to Stage 3. This stage pertains to the retention of the information acquired from the preceding stages. The latency evaluation results from the second stage are depicted in Figure 4.39. The *Zero W* required 0.07 ± 0.05 milliseconds to complete the initial stage. The *3B* took 0.02 ± 0.04 milliseconds to accomplish this task. The *3B+* required 0.03 ± 0.05 milliseconds to perform this stage. The *Jetson 5W* took 0.02 ± 0.04 milliseconds to complete this part, whereas the *Jetson 20W* took 0.01 ± 0.03 milliseconds. The observed discrepancies in this instance arose due to the time interval of this particular stage being shorter than the minimum recorded value, resulting in numerous measurements being conducted within an infinitesimally little timeframe.

All of the suggested gear is capable of executing the intended task. Therefore, the decision is made by assessing the performance, which may be subsequently compared to the project cost. Based on the test findings, it is evident that the Jetson Nano outperformed the Raspberry Pi 3B and 3B+ even when operating in power saving mode. The Raspberry Pi Zero W has the least amount of computational capacity, resulting in the poorest overall performance. Despite the similarities in hardware specs, the Raspberry Pi 3B outperformed the Raspberry Pi 3B+ and exhibited a performance level similar to the Jetson Nano operating in the 5W mode. The graph labeled as Figure 4.40 illustrates the mean anticipated rate of predictions per second for each platform.

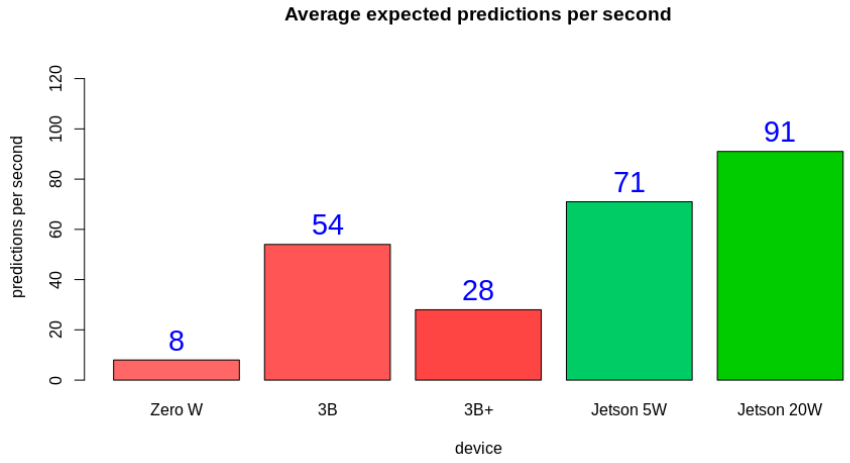


Figure 4.40: Average expected predictions per second ratio on each platform. The number in blue displays the expected ratio.

As anticipated, the performance of the *Zero W* was exceedingly poor. This supports the initial architectural idea to include an additional hardware component to handle the more demanding processing activities. Another anticipated outcome was the attainment of the highest level of performance using Jetson 20W. A notable finding is that, despite having a potentially superior processor, *3B* exhibited sig-

nificantly better performance than *3B+*. Another significant finding is that despite disabling two out of the four cores, the *Jetson 5W* exhibited higher performance compared to the *3B* and *3B+* models operating with all four cores and consuming over twice the amount of power. Given its ability to execute at a high level while operating under a power constraint, the *Jetson 5W* is an excellent choice for field processing. Based on these findings, we conclude that *3B* and *Jetson 5W* are the primary contenders for fulfilling this application, as they exhibit the most favorable balance between performance and power consumption.

Ultimately, we assessed the ratio of average predictions per second by executing the CNN and MLP pipelines. The sole distinction in the CNN pipeline, as depicted in Figure 4.36, is the absence of a feature extraction procedure, which is not necessary for the CNN. Therefore, this stage alone encompasses the alteration of input data to be provided to the model. The tests were conducted on the primary hardware options, namely Jetson 5W, Jetson 20W, and pi3B. The findings achieved for the predictions per second in the proposed setups were as follows:

- The average predictions per second ratio in pi3B was 54 ± 1 for the MLP pipeline and 5 ± 0 for the CNN pipeline;
- The average predictions per second ratio in Jetson 5W was 71 ± 5 for the MLP pipeline and 10 ± 0 for the CNN pipeline;
- The average predictions per second ratio in Jetson 20W was 91 ± 7 for the MLP pipeline and 15 ± 0 for the CNN pipeline;

Figure 4.41 also displays these results in the respective cited order. This data indicates that even in case of improvements on the software results, the CNN model is not adequate for time-restrictive tasks in the proposed configurations. This model is suitable to perform a later review of in-field captured results, but not to be integrated into a distributed constrained environment within the context of these tests.

Software Validation Tests

The software validation tests in the previous section incorporate conventional ML metrics. Within this particular framework, we assessed the metrics of *Precision*, *Recall*, and *F1-Score*.

To ensure simplicity, the validation set is randomly selected from the training data only once. It contains 10% of the total photos from the training data. The findings for the validation set are shown in Table 4.3. The findings indicate that the system successfully detected the sick leaves in 90% of the instances. Additionally, the Precision and Recall exhibit equilibrium, leading to a well-balanced F1-Score.

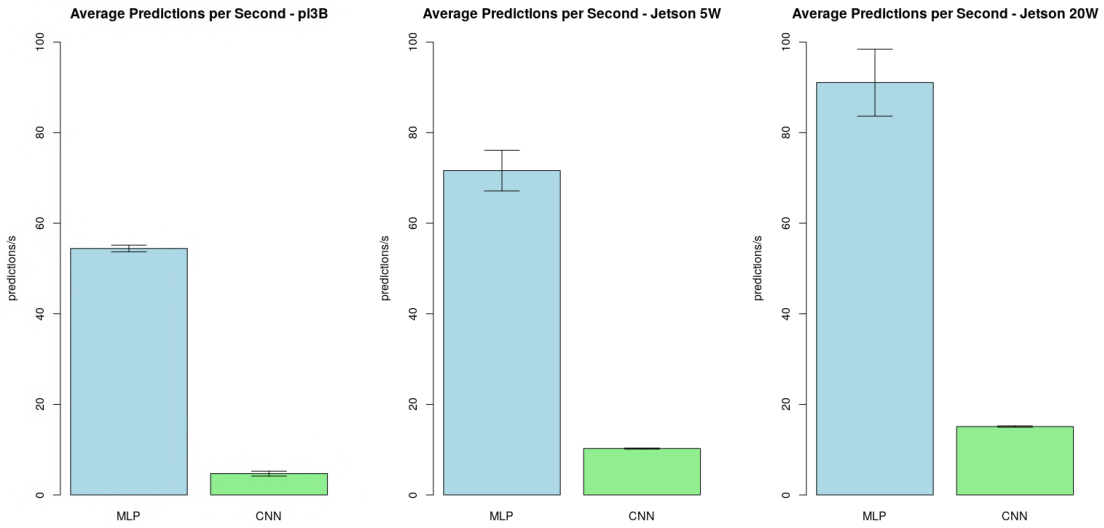


Figure 4.41: MLP and CNN performance comparison test results.

This outcome suggests that the number of incorrect positive and negative results is almost equal. The confusion matrix obtained from this stage is displayed in Table 4.4.

Table 4.3: Metric results for the validation dataset. This set was obtained separating 10% of the training data for validation.

Global Accuracy: 90%				
	Precision	Recall	F1-Score	Support
healthy	0.89	0.90	0.90	198
diseased	0.90	0.90	0.90	209

Table 4.4: Confusion Matrix for the validation data

	Healthy	**Diseased**
Healthy	178	20
Diseased	21	188

Additionally, we computed the global mean and the conventional metrics for the test dataset. Previously, the test set was partitioned by extracting 10% of the photos from the original dataset. The acquired results for the validation set are shown in Table 4.5, while the confusion matrix for this stage is displayed in Table 4.6. Once again, the results indicate that the system was able to accurately detect the sick leaves in approximately 90% of the cases. Despite a slight disparity, the Precision and Recall exhibit equilibrium, leading to a harmonized F1-Score. This outcome validates the practicality of the suggested method within the specified framework.

Based on these stages, we can infer that the method is valid for the intended purpose. It accurately distinguishes between diseased and healthy leaves with an

Table 4.5: Metric results for the test dataset. This set previously separated, taking 10% of all images.

Global Accuracy: 91%				
	Precision	Recall	F1-Score	Support
healthy	0.93	0.88	0.91	217
diseased	0.89	0.93	0.91	220

Table 4.6: Confusion Matrix for the test data

	Healthy	**Diseased**
Healthy	192	25
Diseased	15	205

approximate accuracy of 90%, and the outcomes are well-balanced. The following tests must verify the architectural characteristics of this solution. In addition, we conducted identical predictions taking into account the CNN. In this case, we utilized the identical test set to acquire the prediction outcomes.

Table 4.7: Metric results for the test dataset - CNN results. This set is the same previously separated for the MLP.

Global Accuracy: 96%				
	Precision	Recall	F1-Score	Support
healthy	0.96	0.95	0.96	217
diseased	0.95	0.96	0.96	220

Table 4.8: Confusion Matrix for the test data - CNN results

	Healthy	**Diseased**
Healthy	207	10
Diseased	9	211

As expected, the CNN performed better than the MLP. The results exhibit a 5% enhancement in precision compared to the previous results. This outcome reinforces the necessity of utilizing this approach in future investigations instead of the MLP model. From the standpoint of leveraging greater processing capacity for data analysis, the Convolutional Neural Network (CNN) is a more favorable model compared to the Multilayer Perceptron (MLP).

Architecture Validation Tests

The architecture validation test assesses the capacity to execute a task while adhering to a soft real-time limitation. It refers to a performance assessment that offers a comprehensive perspective of the scalability of the suggested architecture. In this case, we employed the Jetson Nano as an Edge AI server to execute the pipeline

illustrated in Figure 4.25. We designed a version of this system for the client that gives latency statistics for the steps marked in Figure 4.42.

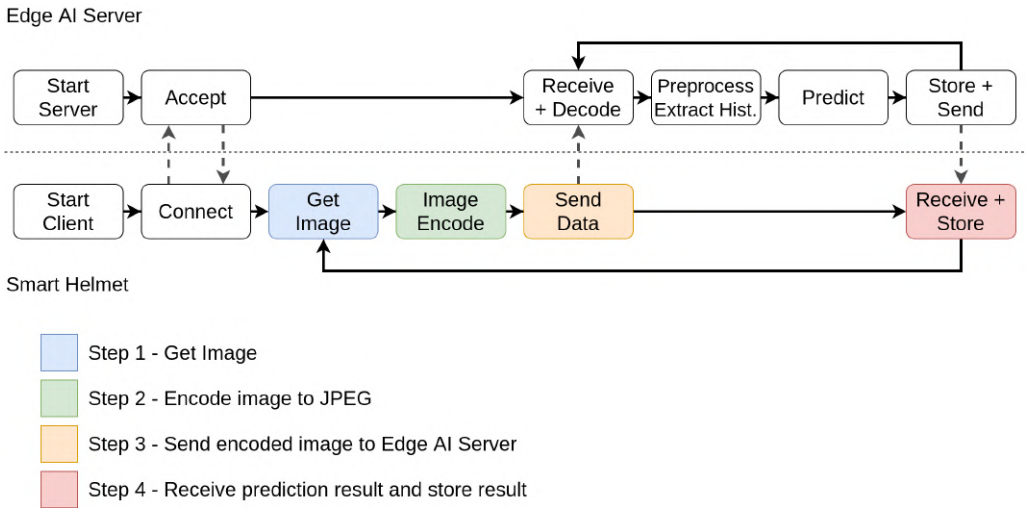


Figure 4.42: Stages considered in the architectural validation test.

All clients have access to a uniform set of events. The number of nodes executing the tasks is equivalent to the number of devices carrying out the IoT-dependent operations. This experiment involves augmenting the client count and assessing its impact on the soft real-time restriction.

Thus, it is necessary to initially assess the real-time criteria in relation to the pipeline depicted in Figure 4.42. Therefore, we conducted the test taking into account the latency of the procedures for a solitary customer. Additionally, the test takes into account a series of finite discrete time intervals. Within this framework, we have designated the smallest time interval as 1 millisecond. The delay for each step in a single-client test is shown in Figure 4.43. In order to establish the soft real-time constraint (ϕ), we assessed the minimum quantity of blocks required to deliver the service to a single client with a 100% level of quality ($Qf = 1.0$), along with an extra 10% allowance for loosening the criterion. We established the value of ϕ as 90 milliseconds using this approach.

Once ϕ was defined, we conducted multiple iterations of the test with 2 to 9 clients, all of whom were assigned the same assignment. The simulations were conducted on a computer machine that was linked to the WLAN network, serving as the Edge Server. During each test, every instance executed the identical test described in Figure 4.42, measuring the duration of each desired occurrence. Ultimately, we calculated the average and standard deviation of the quality factor, taking into account all the nodes that were part of the analysis. The test result is shown in Figure 4.44. This outcome demonstrates that the performance of the Edge-AI service deteriorates as the number of customers increases, while adhering to a specific real-time limitation. However, the system maintains a high level of quality despite having a

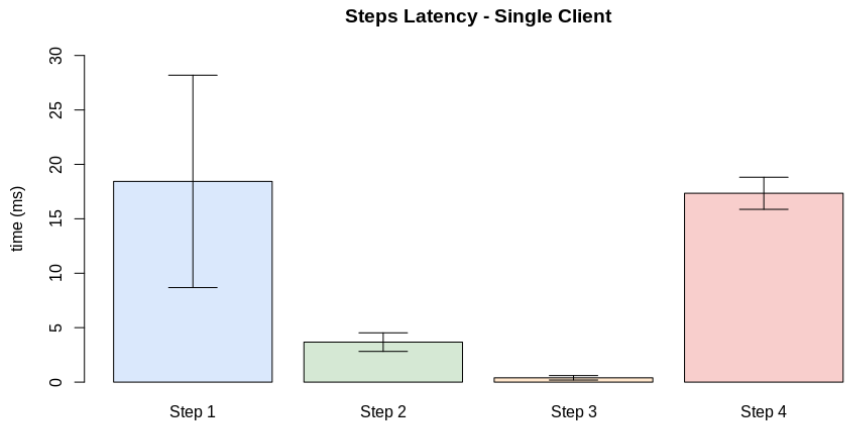


Figure 4.43: Latency for each of the steps presented in Figure 4.42

small number of connected clients.

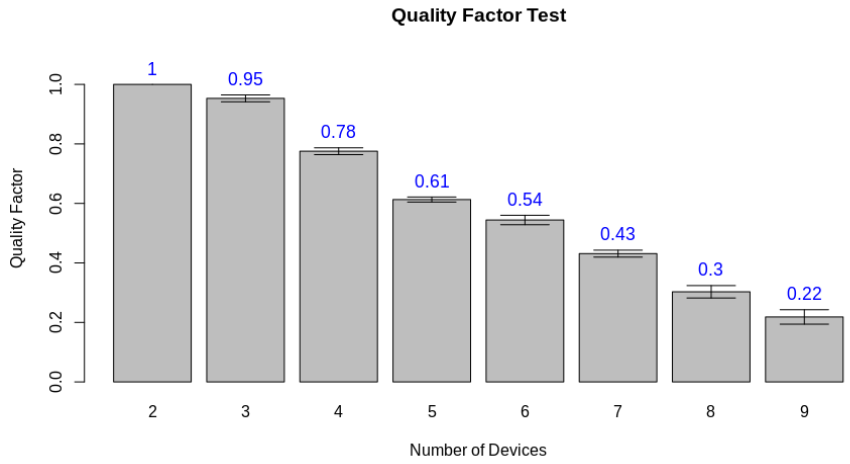


Figure 4.44: Quality Factor test result

Ultimately, we must ascertain whether the degradation in quality was a result of server overload or if other factors had an impact on the simulation program. In this regard, we quantified the mean latency of each stage while progressively increasing the number of nodes. Although steps one and two are contingent on the specific device being used, step three is vulnerable to potential network congestion, and the success of step four hinges on the efficiency of the Edge AI node. The findings for this analysis are displayed in Figures 4.45, 4.46, 4.47, and 4.48.

As anticipated, the latency in the initial two stages remained unaffected by the growing number of customers. These steps rely solely on the client carrying out its tasks. The third step introduces the initial action that relies on the network. The growing clientele may pose a challenge in the communication process. The findings indicate that this excessive load leads to an increase in delay for this particular

stage, albeit the effect on the ultimate outcome is negligible (about 2 milliseconds). Ultimately, stage 4 reveals that the excessive burden on the Edge AI node is the primary element contributing to a decline in quality as the number of clients increases. This level is associated with both networking and the process of machine learning inference.

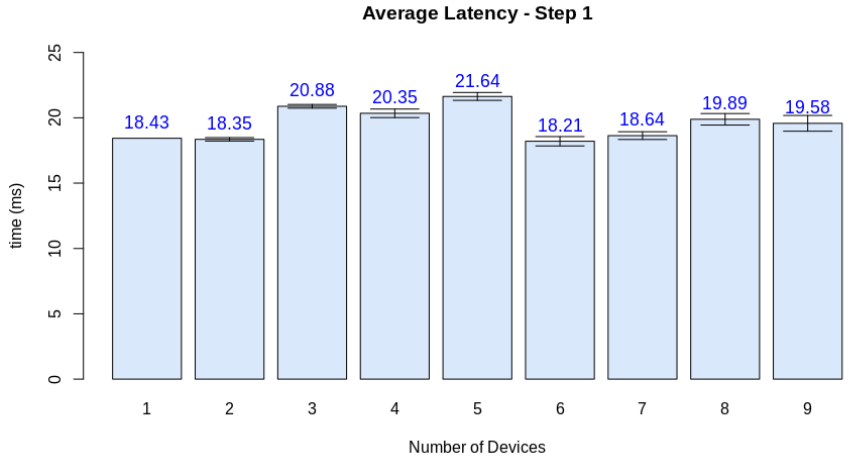


Figure 4.45: Latency test results for step 1

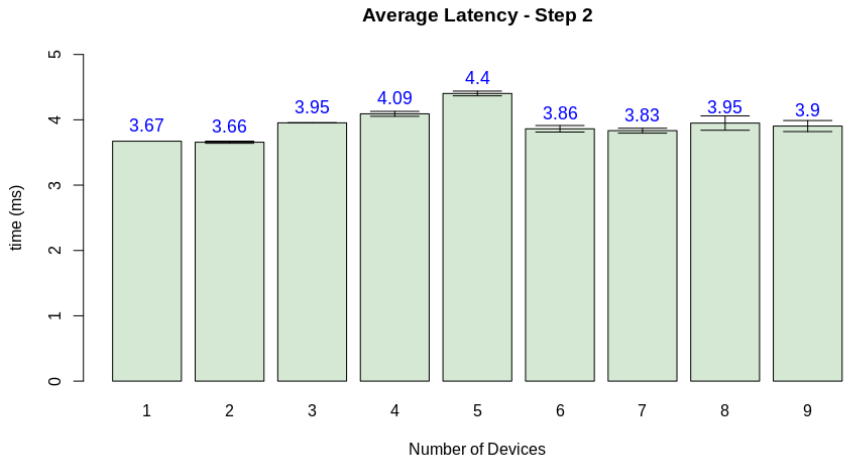


Figure 4.46: Latency test results for step 2

Case Study Validation for Deployment

We additionally conducted a validation phase for the entire solution. To address this issue, we created a simulation based on a case study appliance to test the proposed approach. Within this application, a trio of researchers employ the cylinder approach to sample 200 leaves at various predetermined heights. The Edge AI server node use predictive algorithms to determine the status of each leaf and subsequently records

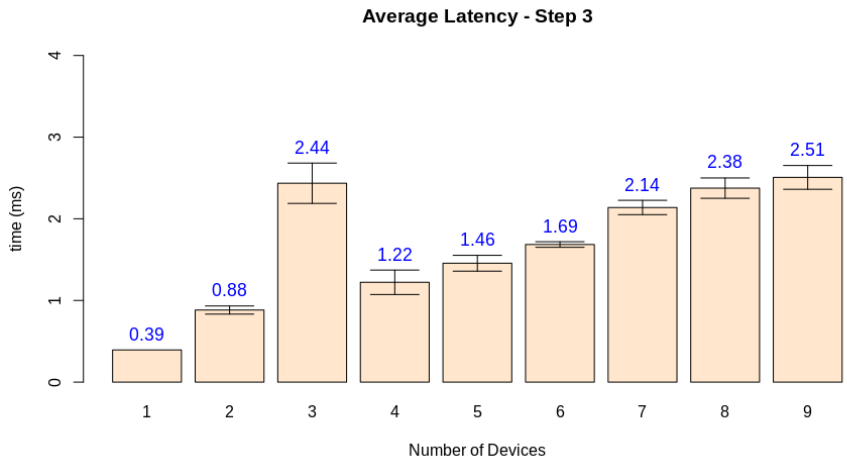


Figure 4.47: Latency test results for step 3

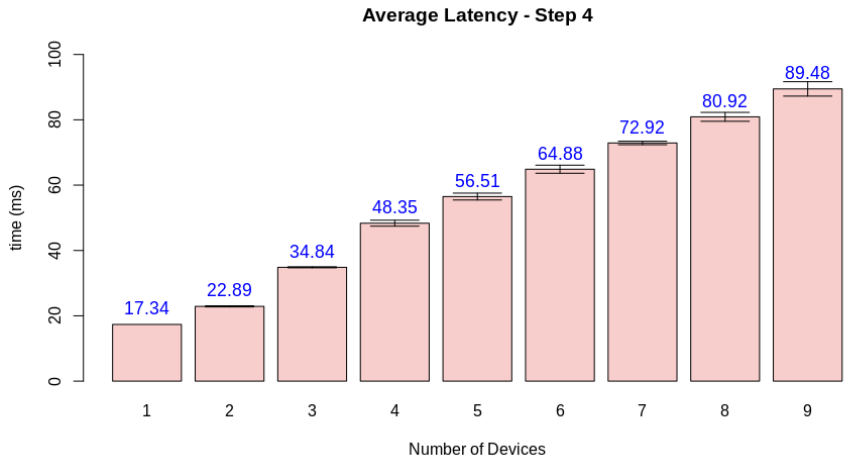


Figure 4.48: Latency test results for step 4

this information alongside the corresponding researcher coordinates. The researchers are positioned within a circular area with a diameter of 5 meters around the tree trunk in this device. Figure 4.49 illustrates the arrangement of various devices in relation to the tree trunk.

We utilized Equation 4.17 as the reference point for randomly choosing leaves from the sets of infected and healthy samples. Initially, the probability baseline was determined by computing the (x, y, z) coordinate of each researcher at the given position. Subsequently, the computer creates a random number within the interval of $[0, 1)$ for each of the 200 samples. If the value is below the baseline probability, the algorithm will choose a diseased leaf. Alternatively, it chooses a nutritious one.

After this process, we performed a test with the trained model. The test program uses each sample to estimate the leaf conditions for every device and location. Using this dataset, the application computes the proportion of infected leaves, resulting

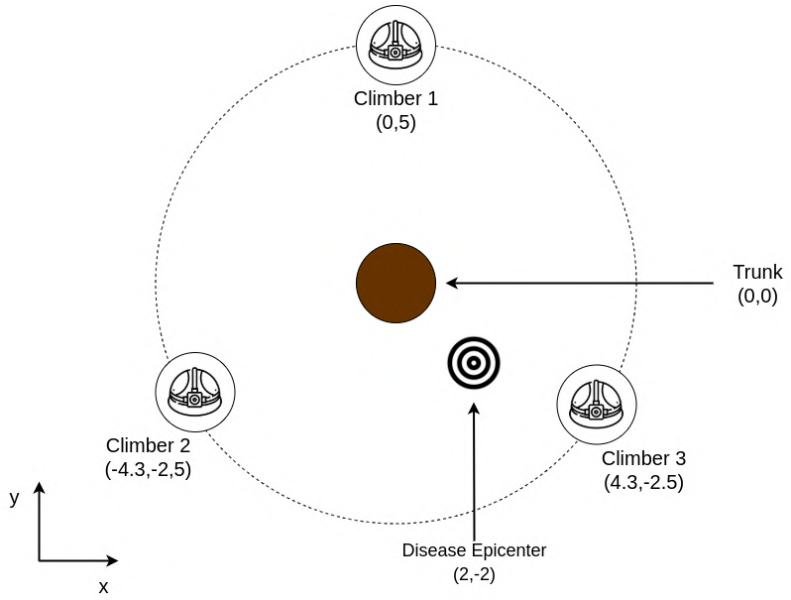


Figure 4.49: Upper view of the case study organization

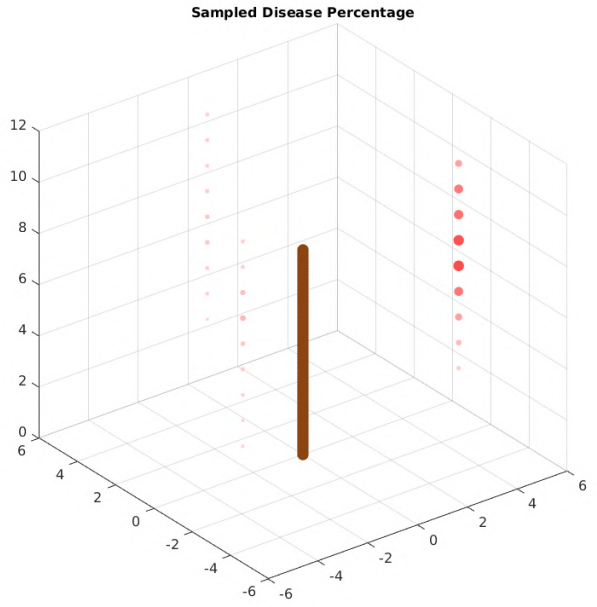


Figure 4.50: Case Study sampling distribution. The larger and more colorful red dots have a bigger percentage of diseased leaves. The brown cylinder represents the main tree trunk.

in a distribution sample. The outcomes of the sampling procedure are depicted in Figure 4.50, taking into account the organization shown in Figure 2.1. As depicted in Figure 4.35, there is a positive correlation between the size and color intensity of the red dots and the prevalence of diseased leaves.

Finally, we used an evolutionary algorithm to perform a regression to the parametric PDF presented in Equation 4.11 using the sampled data. Some features of

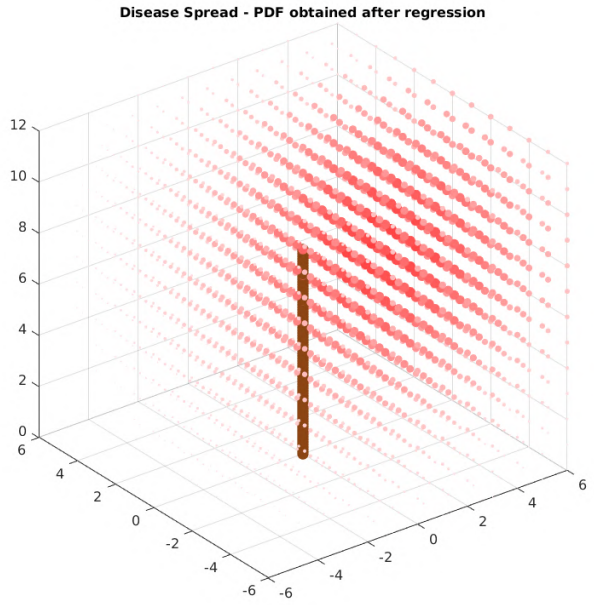


Figure 4.51: Estimated PDF display. The larger and more colorful red dots have a bigger probability density. The brown cylinder represents the main tree trunk.

this algorithm are:

- Each individual genotype is a $T = (p_0, \sigma, x_0, y_0, z_0)$ tuple;
- The population has 100 individuals;
- Each round generates 70 offspring (30% elitism);
- Each round has a complementary local search in half the population;
- The algorithm stops with a convergence criteria and RMSE lower than 0.05 (5%);

To understand how good would a prediction be, we ran the model 20 times, and evaluated the average value for each paramenter of the $T = (p_0, \sigma, x_0, y_0, z_0)$ obtained from the best individual of the population. The average responses obtained from this experiment are:

- $p_0 = 0.65 \pm 0.03$. The original value was 0.65.
- $\sigma = 12 \pm 0.86$. The original value was 5.
- $x_0 = 1.96 \pm 0.21$. The original value was 2.
- $y_0 = -1.52 \pm 0.35$. The original value was -2.

- $z_0 = 8.1 \pm 0.16$. The original value was 8.

The estimated spatial distribution of the disease based on these factors is shown in Figure 4.51. The obtained values closely align with the anticipated outcomes. The distance between the predicted epicenter of the disease and the original PDF is 0.48 meters. The greatest predicted percentage closely approximates the original value. The variability of the predicted model is greater than that of the original module. The variation in results can be attributed to the inherent uncertainty of the leaf categorization model, which is approximately 10%. Despite the prevailing ambiguity, the model yielded a reliable estimation for the parameters of disease propagation, based on the collected data. We conducted experiments by manipulating the algorithm's parameters to determine if the resulting outcomes would be altered. The findings of our study indicate that variations in population size, number of offspring, and maximum epochs had a negligible effect on the outcomes. This outcome demonstrates the high level of resilience in the process of acquiring the model parameters.

4.3 Ant distribution and counting estimation

Comprehending the actions of groups of ants is a crucial obstacle in the field of ecology. Helanterä et al. [193] assert that unicolonial ant populations are the largest cooperative units in nature. According to them, these species are able to construct extensive, interconnected nests. Furthermore, the authors assert that researchers in this domain can generate valuable data by understanding the dynamics of these colonies.

McGlynn [194] states that insect colonies are mobile entities, moving nests through their lifetime. The authors assert that understanding the factors that influence the mobility of the researched species requires knowledge of several aspects, such as its genetics, life-history evolution, and the effect of competition. Specifically, the authors assert that ant migration patterns are often unclear.

Regarding the methods of understanding the migration patterns of ant colonies, Hakkala et al. [195] state that reliable data capture of the colony motion is needed. Furthermore, they assert that by integrating this data with environmental information, it is feasible to understand how the setting influenced their migration. Technology solutions serve as a method to improve data collection and develop creative solutions for this issue.

Majer and Heterick [196] also evaluate the subject of devising experiments to achieve this objective. According to the authors, prolonged observation is crucial for doing research on invertebrates. This element also ensures that the development

of innovative technology instruments aimed at this objective has a favorable impact on researchers in this field.

This study investigates the development of an innovative instrument that enables researchers to assess the dynamics within ant colonies. Our anticipation is to utilize the developed technology to derive data pertaining to quantities and distribution. Our objective was to develop an automated system capable of quantifying the number of ants in the solution. The solution also enables an assessment of the approximate distribution of the ants inside the scene.

4.3.1 Requirements

The first step in this analysis is evaluating the requirements for the proposed method. For this matter, we display a version of the co-design diagram presented in Figure 4.52, which is a simplification of the diagram presented in Figure 3.2b.

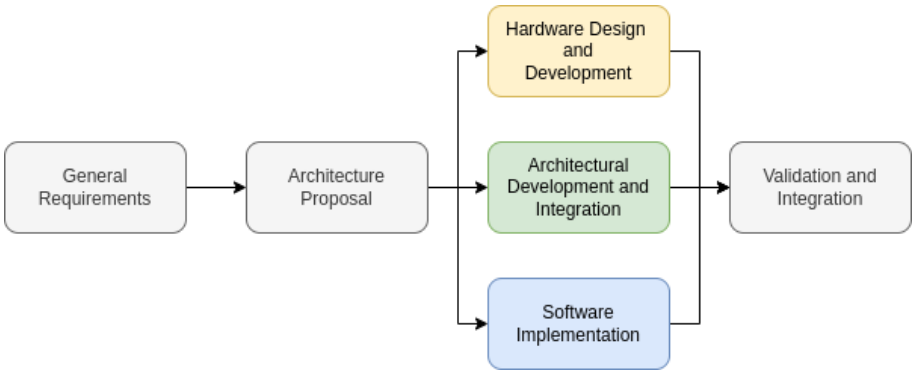


Figure 4.52: Simplified Co-design diagram.

This picture illustrates the necessity of increasing the limitations for the application and categorizing them into the domains of hardware, software, or architecture. The limits identified for this matter are as follows:

- This system must use lower-processing CNNs to match embedded edge server devices [*Hardware*].
- This system must be able to approach how many ants are present in a scene and estimate a spatial distribution. [*Software*].
- The application must use similar validated tools to allow its integration into a cooperative schema [*Architecture*].

In order to address this issue, we decided to conduct experiments using various Convolutional Neural Networks (CNNs) that have limited memory and processing requirements. The purpose of these experiments is to take a partially quantitative approach to assessing the significance and dispersion of ants. A crucial element of this concept is the attainment of the hardware restriction through a software design.

4.3.2 Methods overview

In the preceding sections, we evaluated the significance and originality of the proposed solution. There is no previous example in the literature of a similar solution being produced. Within this part, we delve into the intricacies of the suggested resolution. We thoroughly examine the offered solution in the beginning. Next, we will examine the dataset generation tool. In addition, we evaluate the training process of the backbone, providing specific information about the method used for training. Ultimately, we present the assessment criteria for each phase.

General solution

The proposed solution tries to estimate the number of ants present in each area of the image. For this matter, the employed algorithm has four main steps to estimate the number of ants from a picture. The steps involved in this algorithm are:

1. Transform the image size to 1024x1024;
2. Divide the image into a grid of squares of size 128x128;
3. Evaluate semi-quantitatively how many ants are present in each square;
4. Submit the results to an approximation formula for estimation;

Initially, it is necessary to resize the image dimensions to 1024x1024 pixels. This stage facilitates the assessment of diverse images, given that our dataset comprises photos with varying resolutions. By taking this action, we standardize the quantity of assessed areas for every image, which then leads to the next phase. During this stage, the image is partitioned into areas measuring 128x128 pixels each. This first processing facilitates the generation of 64 assessment zones on each image. The deep learning model evaluates each region separately and classifies it into one of ten classes that represent quantity bands ranging from 0 to 45 ants per region. Following this evaluation, we utilize the model's output for each segment to reconstruct the image, taking into account the density of each area, and subsequently carry out the counting process. The diagram labeled as Figure 4.53 provides a comprehensive representation of the entirety of the suggested solution.

This study is a novel and inventive approach to this task, as previously mentioned. Hence, several steps are necessary to accomplish this activity. We require an initial dataset generated by scholars in the field of ecology. The organization and structuring of this dataset necessitate the use of a computational tool. Subsequently, some necessary measures must be taken to train the AI, which encompass the selection of a backbone model for the CNN. Ultimately, it is necessary to set certain criteria to assess the proposed job.

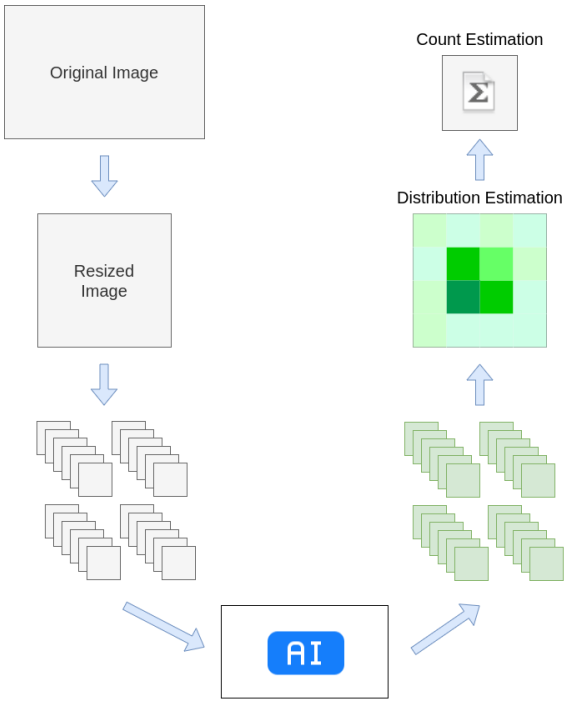


Figure 4.53: Proposed system overview

Dot map generation

As previously said, this is an unresolved issue without an accessible dataset. Consequently, we developed a tool with the purpose of producing a well-organized dataset. Similarly to the dataset used by Wan et al. [197], we chose to create a dot map representing the presence of individual ants on each part of the image. We developed a Guided User Interface (GUI) to do the task. Figure 4.54 depicts a diagram illustrating the workflow of a software.

The program consists of three primary screens. The first screen is the primary interface where the user sets up the input and output directories. Within this display, there exist two inputs for selecting a path. The initial parameter accepts the directory path where the user want to locate the photos for counting. The second parameter specifies the desired file path for the structured CSV file that will contain the output of the markings' information. The dataset is stored in a file called "result.csv" in the output directory. The initial screen arrangement is depicted in Figure 4.55. Upon completing the software configuration, the user is required to initiate the program by pressing the "start" button.

The second window is the counting screen, where users place a dot on each unit they wish to mark. This display features multiple commands. Users are required to select the desired location on the screen to place their dot. The software will record the coordinates and display a red dot at each marked location. To remove the most recent marking, users should click the "Undo" option. Once the marks on the image

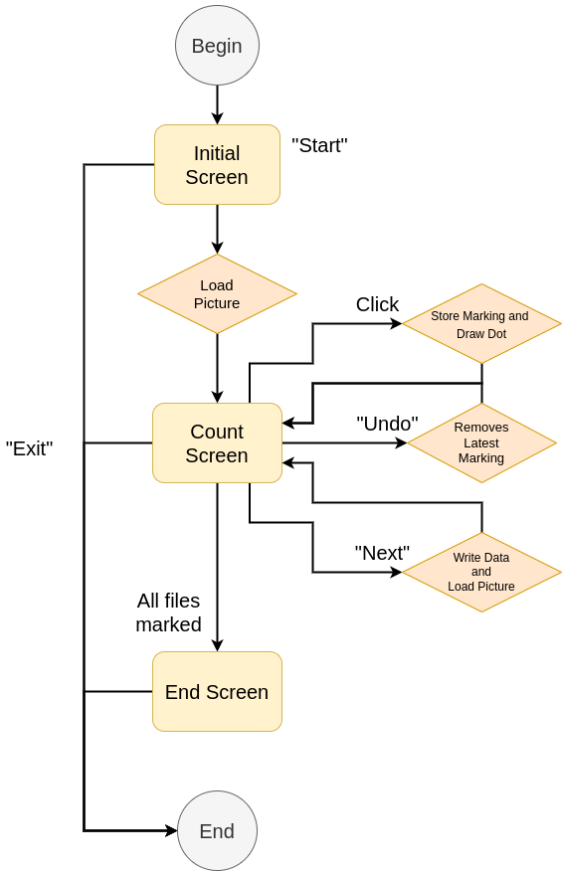


Figure 4.54: Dataset generation software diagram

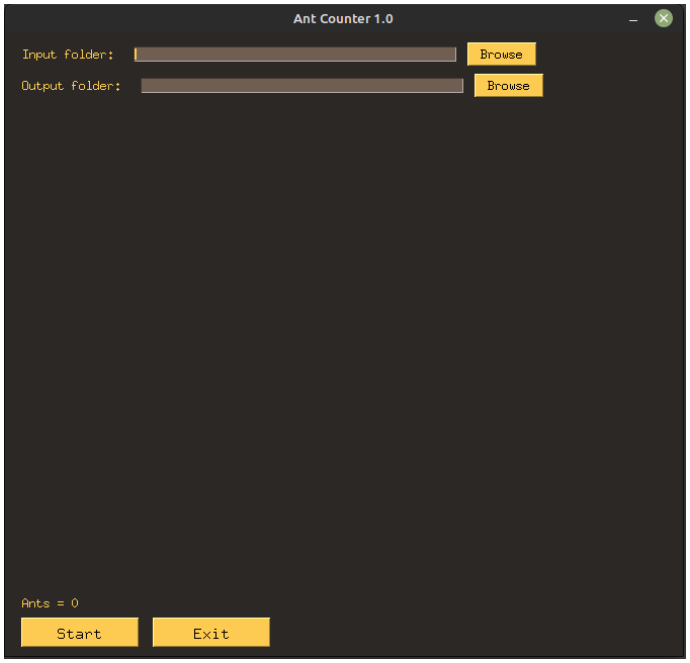


Figure 4.55: Initial Screen

are completed, users may simply click the "Next" button. This action will prompt the program to save the markings onto the disk and load the subsequent image. The screen is depicted in Figure 4.56.

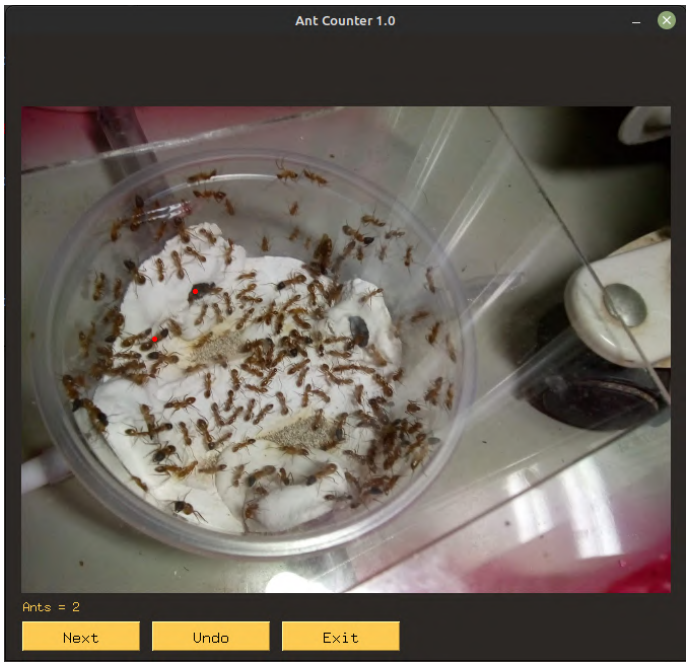


Figure 4.56: Counting Screen

The end screen, in which the program warns the user they have marked all images and finishes the execution. It only gives the option to end the execution. Figure 4.57 shows how this screen is configured.

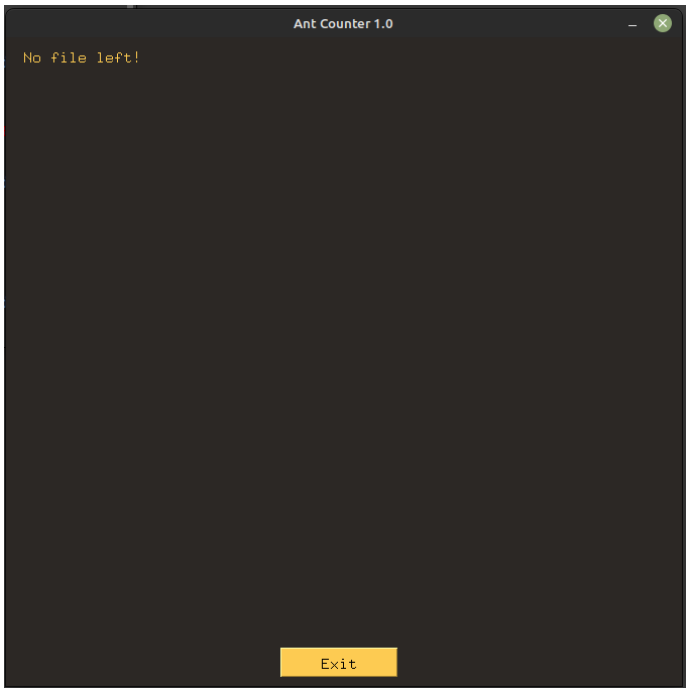


Figure 4.57: Ending Screen

A total of 134 photos were tagged by the laboratory members using this tool, resulting in the creation of dot maps for both sparse and dense scenes of ant colonies. The image with the minimum number of ants has only one, while the image with

the maximum number of ants contains a total of 460. The boxplot in Figure 4.58 illustrates the distribution of the number of ants per image, ranging from sparse to dense scenes.

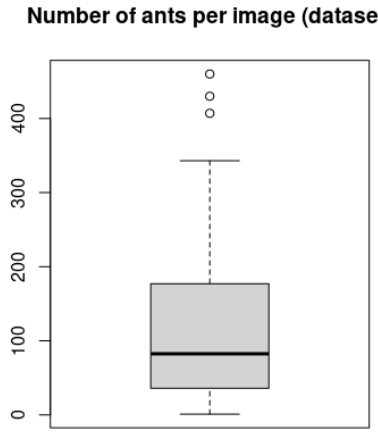


Figure 4.58: Number of Ants per Image Distribution

By utilizing structured annotations, we transformed every image into the 1024x1024 format, accurately mapping the markings to their respective coordinates. As a result of this phase, each image was able to produce 64 zones that included different quantities of ants. In order to generate a semi-quantitative representation that is appropriate for the purpose, we categorized them into 10 distinct classes. The initial class is designated for areas devoid of ants. Each class corresponds to a group of ants ranging from 1 to 5, 6 to 10, 11 to 15, and so on. The final class reflects the highest number of ants per region, which is 45. Any zone with more than 45 ants would be limited to this maximum number. The semi-quantitative classification convolutional neural network was trained using 8576 frames generated from 134 annotated photos.

Data augmentation

Following our preliminary findings, we conducted an investigation into implementing a data augmentation process to validate the capabilities of the model. In this case, we implemented a rotational rule during the process of creating the dataset. This rule was applied after dividing the original image, as explained in the following manner:

1. Store original segment;
2. Perform first rotation ($+90^{o}$);
3. Store rotated segment;

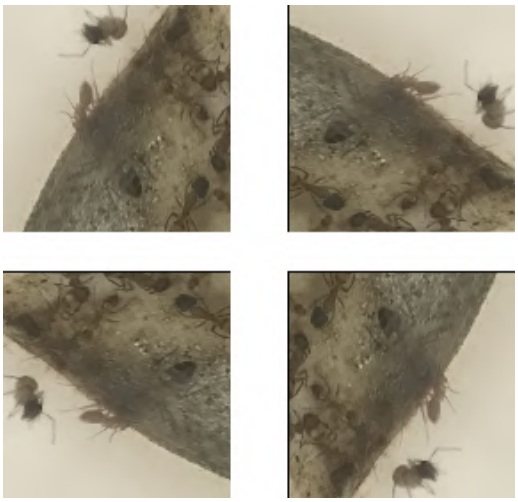


Figure 4.59: Data Augmentation Process Example

4. Perform second rotation ($+90^{o}$);
5. Store rotated segment;
6. Perform third rotation ($+90^{o}$);
7. Store rotated segment;

Following this procedure, we get a dataset that has four times the quantity of photos. By allowing ants to move unrestricted within the space, this action also generates a dataset that is more comprehensive, considering the constraints of the initial data. Figure 4.59 illustrates an instance of this procedure.

AI model training and counting system

As previously said, we initiated this phase with a total of 8576 photos of locations that needed to be categorized into ten distinct classes. The challenge was executed using a convolutional neural network (CNN) as the computational framework. For testing reasons, we examined two high-performance convolutional neural networks (CNNs) as the main frameworks for this strategy. Two models are mentioned: the MobileNet [198] and the EfficientNet V2-B0 [199]. Both models are lightweight convolutional neural networks (CNNs), which are well-suited for executing computationally intensive tasks and can be easily integrated into embedded systems. The training gear is equipped with an i5-9600K central processing unit (CPU) and has a total of 32 gigabytes (GB) of random access memory (RAM). Additionally, it is equipped with an NVidia GeForce RTX 2060 Super graphics card, which provides GPU acceleration specifically for machine learning tasks.

The model consists of an input layer, a backbone without the final classification layer, a dense layer with 32 neurons and linear activation function, and a final dense

classification layer with 10 neurons and "softmax" activation function. Both dense layers employ L1 kernel regularization with a λ factor of 0.01.

Out of the initial 8576 photos, we allocated 80% for training, 10% for validation, and 10% for testing. Due to the imbalanced nature of the dataset, we employed class weights as a means to improve the classification accuracy for the underrepresented classes. To prevent the weights from becoming excessively high or low, we utilized the square root of the initial balanced class weights. We utilized the Adam loss function during the training process.

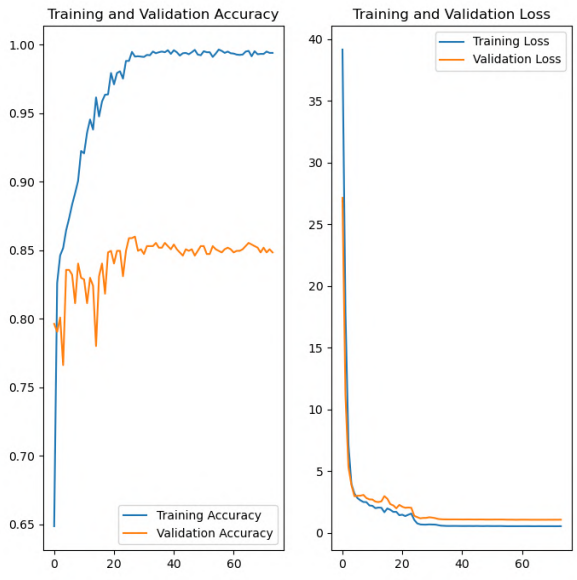


Figure 4.60: MobileNet Training Graph

The training commenced with an initial learning rate of 1×10^{-4}, which was then decreased to 10% of its original value upon identifying plateaus lasting 5 epochs. Ultimately, the algorithm will terminate prematurely upon encountering a period of 15 consecutive epochs where the validation loss remains constant. The graph in Figure 4.60 illustrates the loss functions and accuracy achieved during the training of MobileNet. The functionalities for the EfficientNet V2-B0 are depicted in Figure 4.61. Both figures demonstrate that the architectural design and training measures effectively prevented overfitting. The accuracy achieved in the validation set was verified when evaluating the data using the test set.

Once the CNNs have been trained, the counting system evaluates the output of these networks for each region in the image in order to carry out the counting process. The classification model produces a number ranging from 0 to 9 as its output, determined by the *argmax* function, which identifies the class with the highest probability of classification. Denoting C_i as the classification integer derived

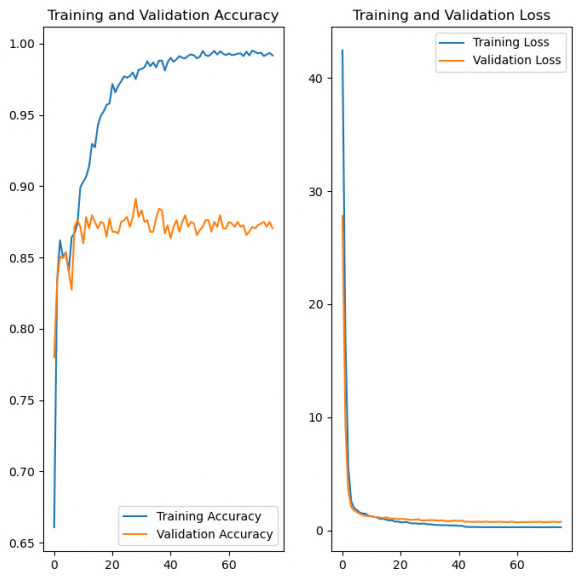


Figure 4.61: EfficientNet Training Graph

from the i-th region of an image in the dataset, the quantity of ants N_i in that region is:

- $N_i = 0$, if $C_i = 0$;
- $N_i = 1$, if $C_i = 1$;
- $N_i = 4 \times C_i$, if $2 \leq C_i \leq 6$;
- $N_i = 5 \times C_i$, if $C_i > 6$.

The number of ants per image A, considering each i region on the image, is given by the equation:

$$A = \sum^{i} N_i \tag{4.18}$$

Evaluation Metrics

Once the techniques for forecasting the quantity of ants in each segment of the dataset have been determined, it is necessary to develop assessment criteria for each phase of the process. Our primary emphasis is on two crucial components of the algorithm: region categorization and counting. The task of area classification involves categorizing regions based on their characteristics. Counting is classified as a regression problem.

As stated, the first stage is a classification problem. For this matter, we used the traditional machine-learning metrics towards classification: *Precision*, *Recall*, and *F1-Score*. They are defined by the True Positive (TP), False Positive (FP), and False Negative (FN) samples from each class. The equations which define each metric are:

$$Precision = \frac{TP}{TP + FP} \tag{4.19}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.20}$$

$$F1\text{-}Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4.21}$$

In addition to these metrics, we also assessed the global average and the confusion matrix as quantitative and qualitative measures of the model's performance.

In addition to defining the measurements for the classification problem, it is necessary to specify the metrics for the regression. Usually, the coefficient of determination R^2 serves as a measure of the accuracy of regressions. The coefficient is derived using the residual sum of squares (SS_r) and the total sum of squares (SS_t). Optimally, the count would approximate the function $f(x) = x$, where $f(x)$ represents the number of ants detected by the AI, and x represents the true value.

The residual sum of squares can be defined as the sum of the squared differences between the ground truth x_n and the model output $\hat{f}_n(x_n)$ for the n-th image. The equation that represents the residual sum of squares (SS_r) is:

$$SS_r = \sum^{n} (\hat{f}_n(x_n) - x_n) \tag{4.22}$$

Similarly, the total sum of squares can be calculated from the mean output value $\overline{\hat{f}}$ and all $\hat{f}_n(x_n)$ values obtained as the model outputs. The equation which represents the SS_t is:

$$SS_t = \sum^{n} (\hat{f}_n(x_n) - \overline{\hat{f}}) \tag{4.23}$$

The equation gives the coefficient of determination R^2:

$$R^2 = 1 - \frac{SS_r}{SS_t} \tag{4.24}$$

We conducted 10 iterations for each backbone to assess the coefficient of determination and see whether there are any statistically significant variations between the models. We conducted a comparison between the average error, the standard deviation of the error, and the median of the error for both backbones. Ultimately, we conducted a comparison of the duration required for each prediction on the entire dataset utilizing both Convolutional Neural Networks (CNNs). We conducted a statistical analysis utilizing the paired t-Test to assess the differences.

4.3.3 Experimental Results

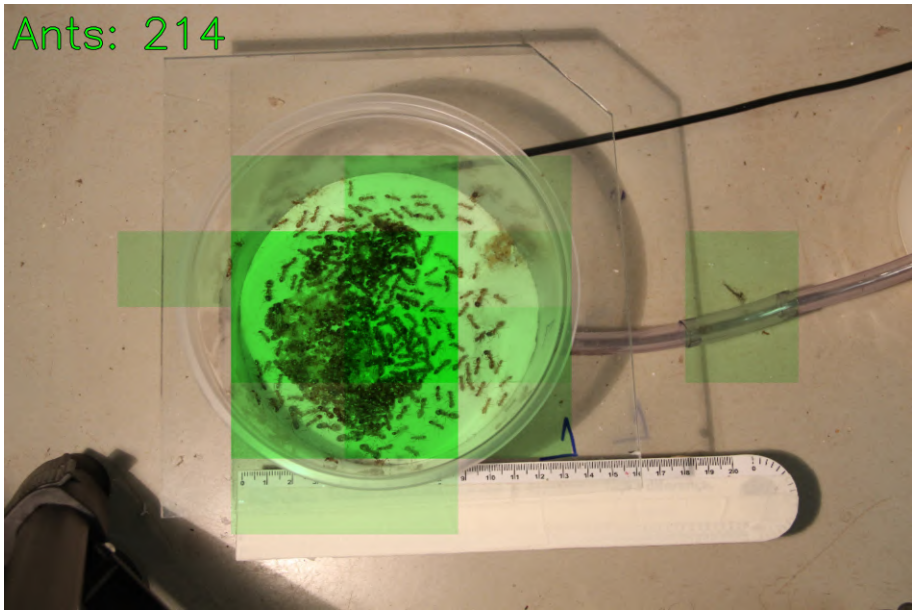


Figure 4.62: Application output example

Once the metrics for evaluating the system were established, we proceeded to train and test it using the proposed algorithm. The preliminary assessment is derived from the underlying convolutional neural networks (CNNs). Figure 4.62 displays the output of a program. As mentioned earlier, we assess it both quantitatively, using standard metrics of classification algorithms, and qualitatively, by utilizing the confusion matrix as a reference point.

The initial assessment is based on quantitative measures. The categorization metrics for the experiments testing the MobileNet as the backbone are summarized in Table 4.9. The overall accuracy was approximately 86%. The measurements indicate a decline in the model's accuracy while predicting classes with increased density. The presence of samples of this size is lower, which accounts for these results.

Given that the issue arises from a somewhat quantitative approach, it is imperative to assess the impact of any inaccuracies by employing a more qualitative

Table 4.9: MobileNet classification metrics

	Precision	**Recall**	**F1-score**	Support
0	0.92	0.96	0.94	584
1	0.81	0.70	0.75	202
2	0.56	0.67	0.61	36
3	0.58	0.50	0.54	14
4	0.40	0.67	0.50	3
5	0.50	0.29	0.36	7
6	0.17	0.25	0.20	4
7	0.25	0.25	0.25	4
8	0.60	0.43	0.50	7
9	0.50	0.67	0.57	3
Accuracy	86%			
Macro avg.	0.53	0.54	0.52	864
Weighted avg.	0.86	0.86	0.86	864

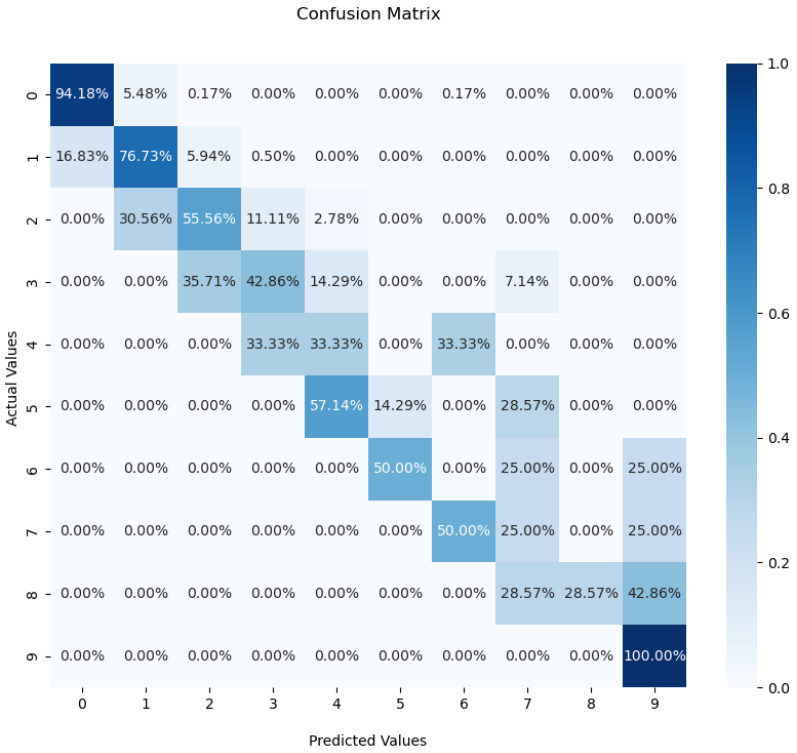


Figure 4.63: Confusion Matrix for the MobileNet

method. In this context, we assess the confusion matrix as a valuable source of information. The confusion matrix obtained utilizing the MobileNet as the backbone is depicted in Figure 4.63. As indicated by the graphic, the majority of errors occur either above or below a single category, leading to errors that are confined within a range of five units.

These first findings indicate that the proposed method is capable of achieving a satisfactory estimation for completing the primary counting tasks. Furthermore, it implies the ability to accurately determine the density of ants in any specific region.

Subsequently, the EfficientNet V2-B0 will be assessed using identical measures. The overall accuracy in this instance was 88%. The findings achieved from training

Table 4.10: EfficientNet V2-B0 classification metrics

	Precision	**Recall**	**F1-score**	support
0	0.94	0.96	0.95	584
1	0.84	0.78	0.81	202
2	0.72	0.72	0.72	36
3	0.64	0.50	0.56	14
4	0.14	0.33	0.20	3
5	0.12	0.14	0.13	7
6	0.12	0.25	0.17	4
7	0.00	0.00	0.00	4
8	0.50	0.43	0.46	7
9	0.50	0.33	0.40	3
Accuracy	88%			
Macro avg.	0.45	0.45	0.44	864
Weighted avg.	0.88	0.88	0.88	864

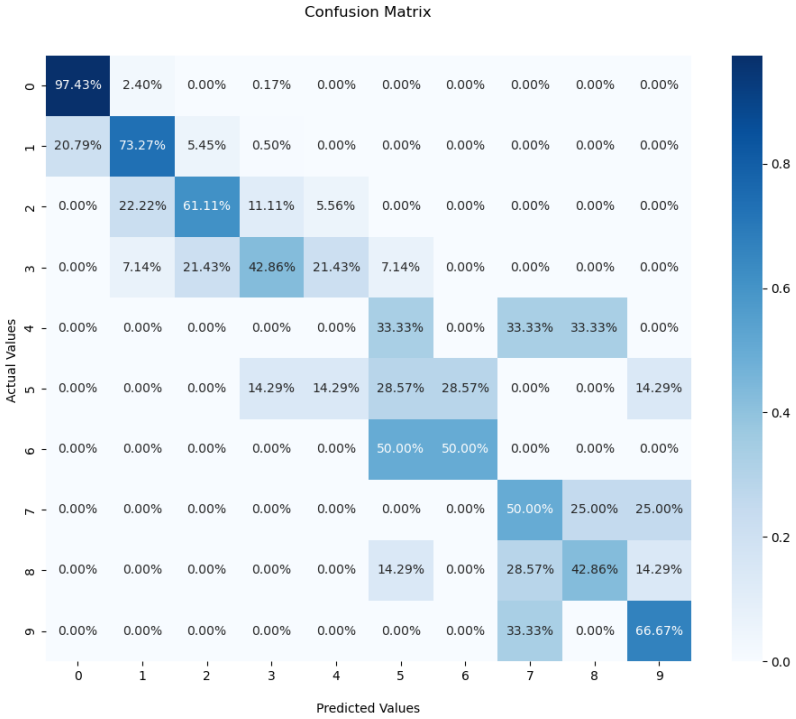


Figure 4.64: Confusion Matrix for the EfficientNet V2-B0

this network are displayed in Table 4.10. Despite having a higher global average, it first exhibits certain problems with certain classes. Similar to the previous situation, the majority of problems are connected to the classes that have the lowest representation.

Furthermore, the similarities and variances underscore the necessity for an additional qualitative assessment utilizing the confusion matrix. The confusion matrix evaluating the test set is shown in Figure 4.64. Once again, it is observed that the majority of errors occur in classes that are similar to the right classification, suggesting that this tool can be effectively utilized in the counting process. The subsequent procedures aim to assess the performance of these techniques in the specific setting of the counting application.

As indicated in the previous section, the counting task has resemblance to a regression problem. However, we are aware of the desired function that we wanted the data to conform to. Thus, we formulated our metrics, as presented in the previous section, taking into account the coefficient of determination for this optimal fitting function.

We conducted ten iterations of training and testing using the identical dataset, with each backbone being employed for separation. The objective of this experiment is to assess the functionality of both systems during a counting stage and determine if there are any statistically significant disparities between the use of each backbone model.

At first, we assessed the metrics by utilizing MobileNet as the underlying framework. The results obtained from these tests are shown in Table 4.11. The results demonstrate consistency, as indicated by an average error of approximately ten ants. The median error is approximately eight ants. The mean coefficient of determination was 0.9783, which remained constant over all ten iterations, with a standard deviation of roughly 10^{-3}. This outcome demonstrates the tool's capability to accurately count objects in situations that range from having few to many objects. The scatter plot from the most recent run is shown in Figure 4.65. The majority of points converge towards the optimal count, as denoted by the red indicator.

Table 4.11: Counting metrics for the MobileNet

	Median error	**Mean error**	**SD error**	R^2
	8	10.34	10.36	0.9774
	8	10.61	10.61	0.9773
	7.5	9.91	9.91	0.9797
	7.5	10.61	10.69	0.9777
	7.5	10.56	10.72	0.9766
	7.5	10.17	10.23	0.9778
	7	9.86	9.83	0.9799
	7.5	10.00	10.22	0.9785
	7	9.94	10.23	0.9787
	7.5	10.17	10.43	0.9792
Average	7.5	10.22	10.32	0.9783

We further examined the metrics derived from utilizing the EfficientNet V2-B0 as the underlying framework. The results from the second series of testing are presented in Table 4.12. The data additionally exhibit consistent behavior, suggesting that substituting the backbone also yielded a viable solution. The mean coefficient of determination was 0.9792 and remained constant over all ten runs, with a standard deviation of roughly 10^{-3}. The mean error was approximately 10 ants, and the median error was approximately seven ants.

Initially, the findings appear to be comparable to the prior tests, with a few of them showing a slight enhancement in the second set. Upon evaluating the data, it was found that this improvement did not exhibit statistical significance. The sole

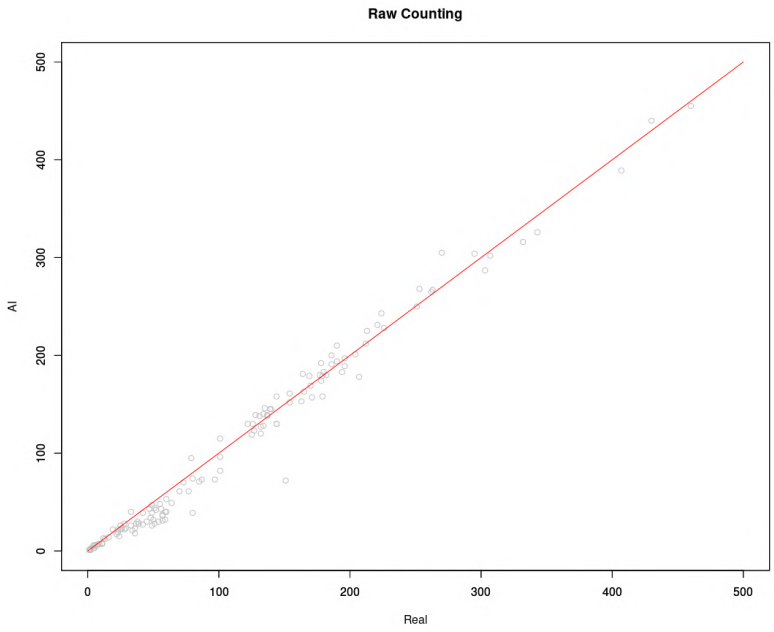


Figure 4.65: Scatter plot from the counting samples for the MobileNet. The red line indicates the ground truth.

outcome that showed a statistically meaningful enhancement was the coefficient of determination R^2, with a p-value of 0.065 when compared to the baseline using a paired t-Test. Additionally, we present the scatter plot of the most recent execution in Figure 4.66. The graphic indicates that the outcomes closely resemble those of the previous model.

Table 4.12: Counting metrics for the EfficientNet V2-B0

	Median error	**Mean error**	**SD error**	R^2
	7	10.05	10.12	0.9798
	7	10.33	10.85	0.9779
	7	9.91	9.50	0.9811
	8	10.34	10.56	0.9777
	7	10.14	10.24	0.9789
	8.5	10.34	10.68	0.9783
	7	9.62	10.09	0.9798
	7.5	9.92	10.19	0.9793
	8	10.20	9.91	0.9799
	7.5	9.82	10.08	0.9795
Average	7.45	10.07	10.22	0.9792

The most recent analysis conducted in this particular situation was the immediate and continuous understanding of the situation. This investigation is conducted by assessing the time intervals required to count each image. We utilized a dataset including 134 photos and conducted the evaluation using both models.

The mean duration for doing all measurements using MobileNet as the underlying framework was 0.410 ± 0.118 seconds. The application, which utilized the EfficientNet V2-B0 as its backbone, had an average execution time of 0.474 ± 0.122

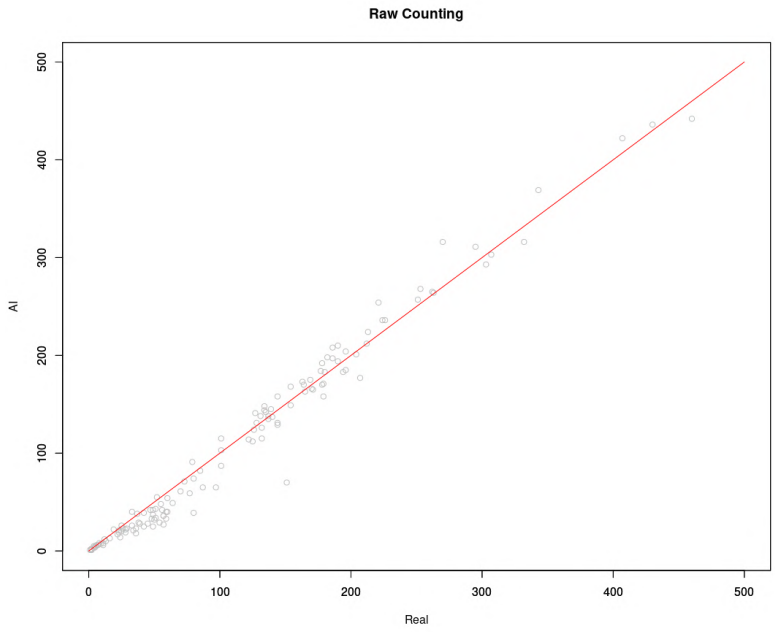


Figure 4.66: Scatter plot from the counting samples for the EfficientNet V2-B0. The red line indicates the ground truth.

seconds. The paired t-test demonstrated a statistically significant difference between these times (p ¡ 0.05).

The findings suggest that the program, utilizing the EfficientNet V2-B0 model as its core, has the capability to make around 182278 predictions within a 24-hour period. Additionally, the program has the capability to execute 210731 predictions every day utilizing MobileNet as its core architecture, without any noticeable degradation in quality. When utilizing this technology for real-time sampling, it is imperative to take these limitations into account.

The conclusive findings from the series of experiments provide initial proof that a system employing this methodology is viable for the tasks of counting and predicting density. Both the assessment of the model and the ultimate tally demonstrate encouraging results, bolstering the advancement of this technology. The same algorithms can be applied in future applications to accomplish counting jobs in dense and sparse environments inside various contexts.

Results after data augmentation

Following the original set of tests, we conducted the studies again using the expanded dataset. We assessed the training outcomes and counting outcomes using identical indicators. The validation and test datasets consist of 3430 photos, whereas the training dataset has 27445 images. Firstly, we examine the training outcomes for each network utilizing the supplemented dataset.

The accuracy and loss function during the training of both approaches are de-

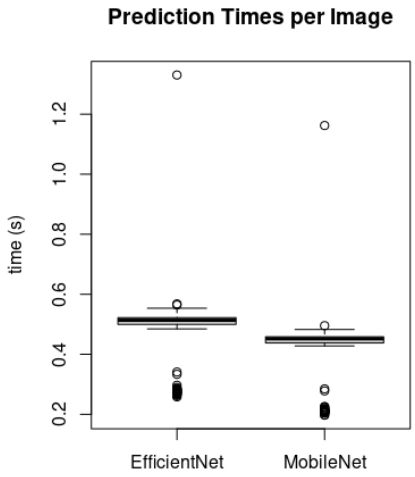


Figure 4.67: Boxplots indicating the time per using each backbone

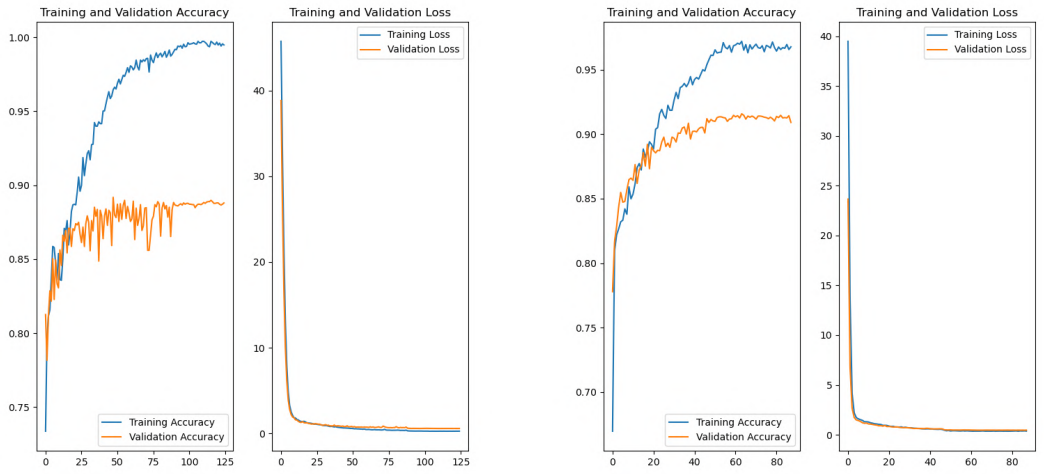


Figure 4.68: Training information using the augmented dataset. On the left, we display the results for the MobileNet. On the right, we display the results for the EfficientNet-V2B0.

picted in Figure 4.68. The resultant outcome was comparable to the initial findings. This material demonstrates the strength and effectiveness of the proposed methods. Subsequently, it is vital to comprehend the caliber of the prognostications.

Confusion matrices for the validation and test sets utilizing the MobileNet are shown in Figures 4.69 and 4.70. This provides additional evidence to corroborate the initial conclusions. Similar outcomes may be witnessed with the EfficientNet-V2B0, as depicted in Figures 4.71 and 4.72.

Finally, we also evaluated the counting process. We also obtained results with a coefficient of determination of circa 0.98 for both models. Figures 4.73 and 4.74

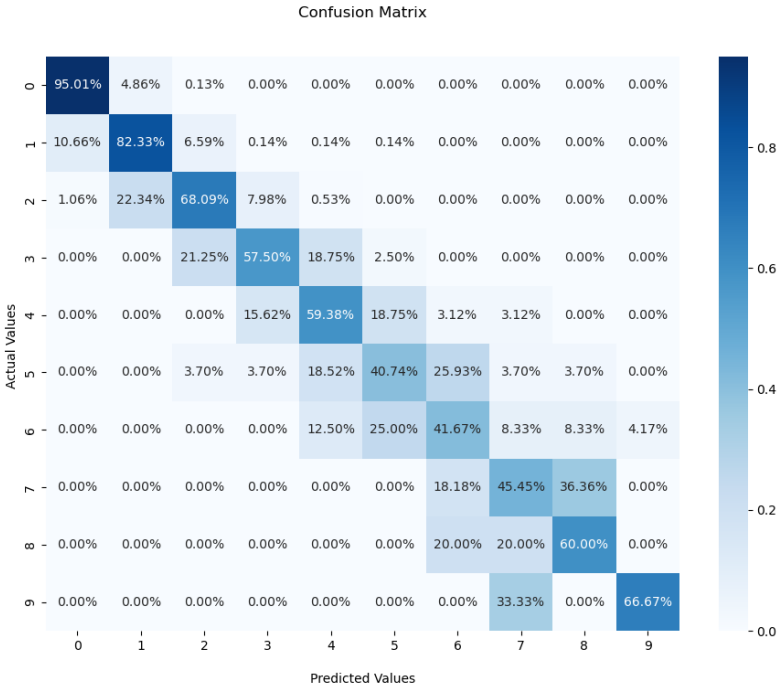


Figure 4.69: Confusion Matrix for the validation set using the MobileNet backbone

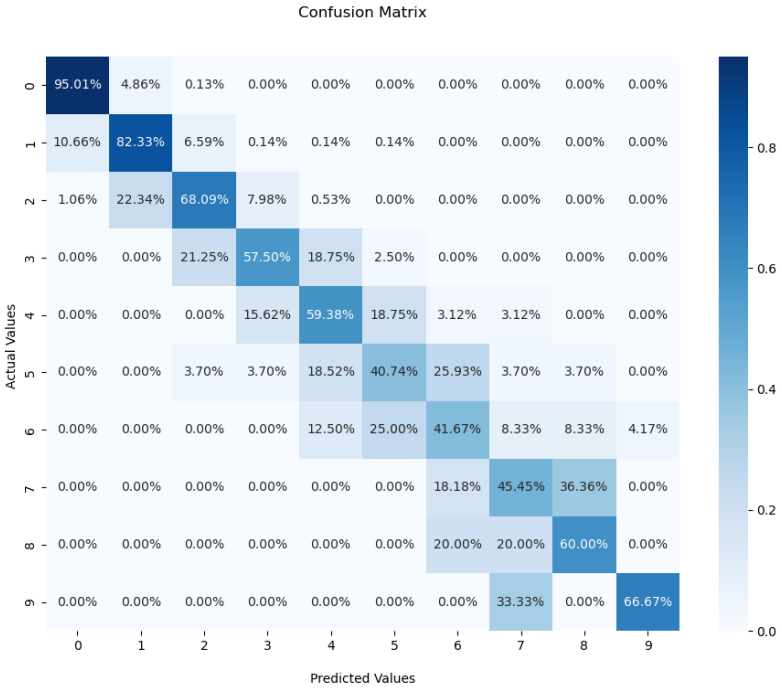


Figure 4.70: Confusion Matrix for the test set using the MobileNet backbone

display the results for the counting process using the MobileNet and EfficientNet-V2B0 as backbones.

This set of resutls present another collection of evidences of the process ro-

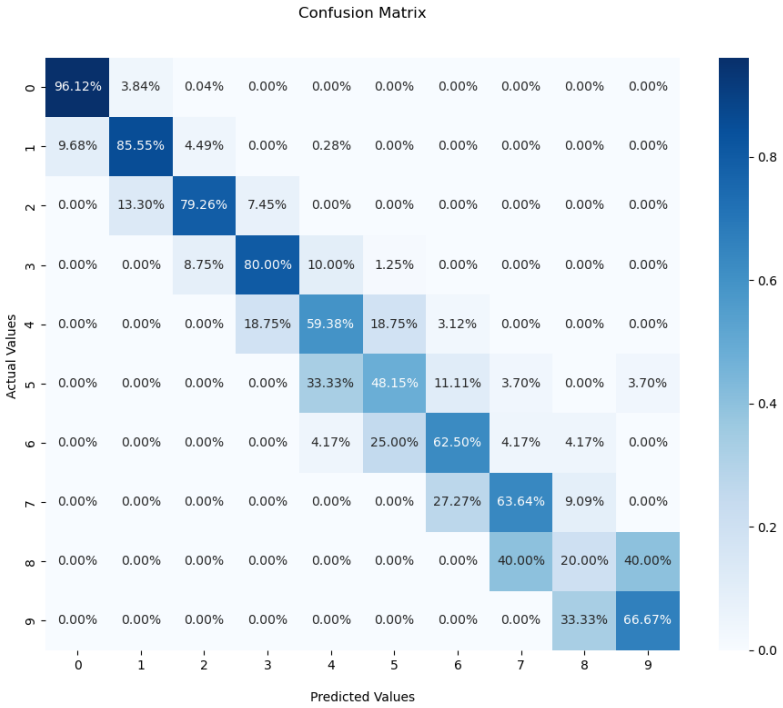


Figure 4.71: Confusion Matrix for the validation set using the EfficientNet backbone

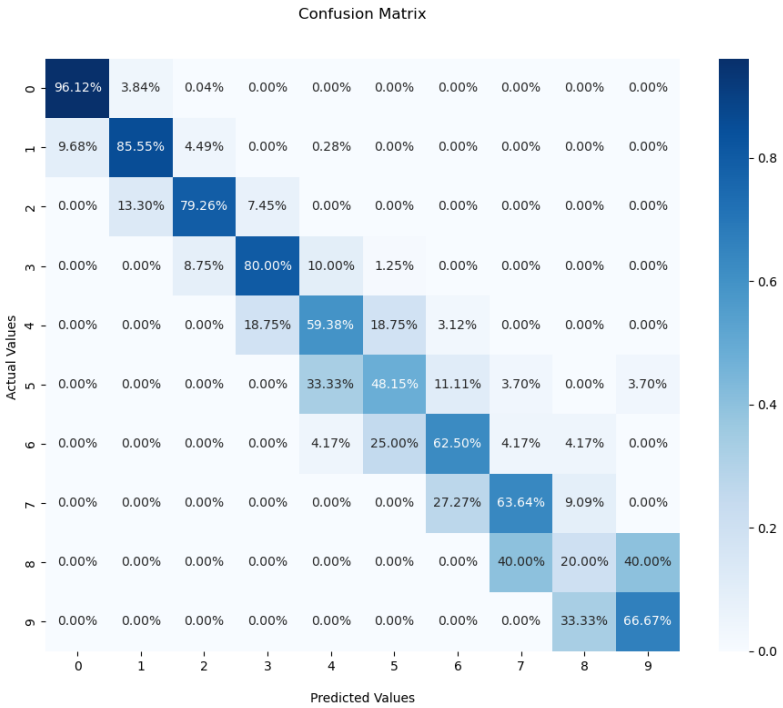


Figure 4.72: Confusion Matrix for the test set using the EfficientNet backbone

bustness. Although it does not solve the dataset limitation, it enforces the early conclusions obtained in previous tests.

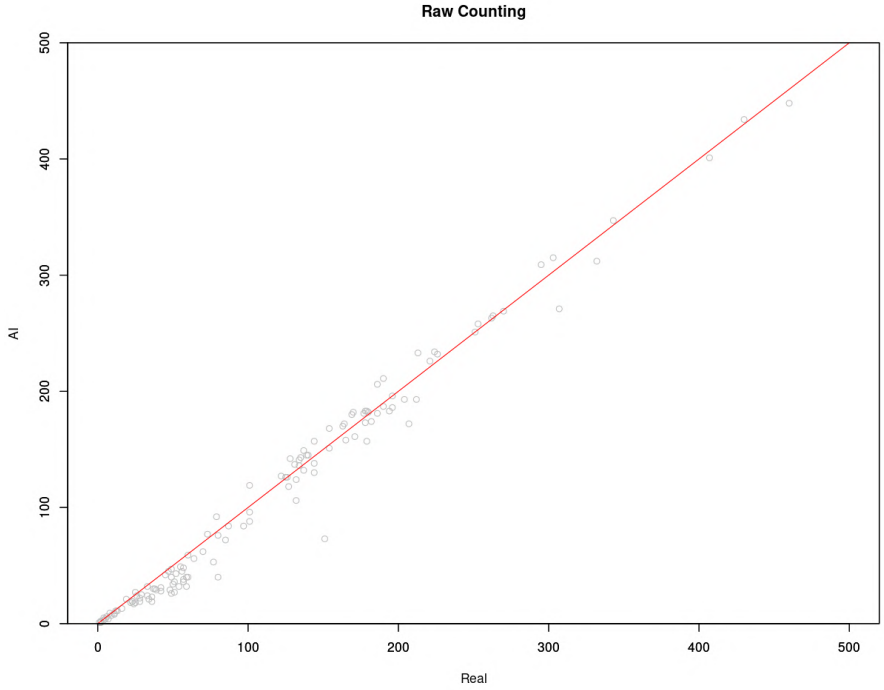


Figure 4.73: Counting graph for using the MobileNet backbone

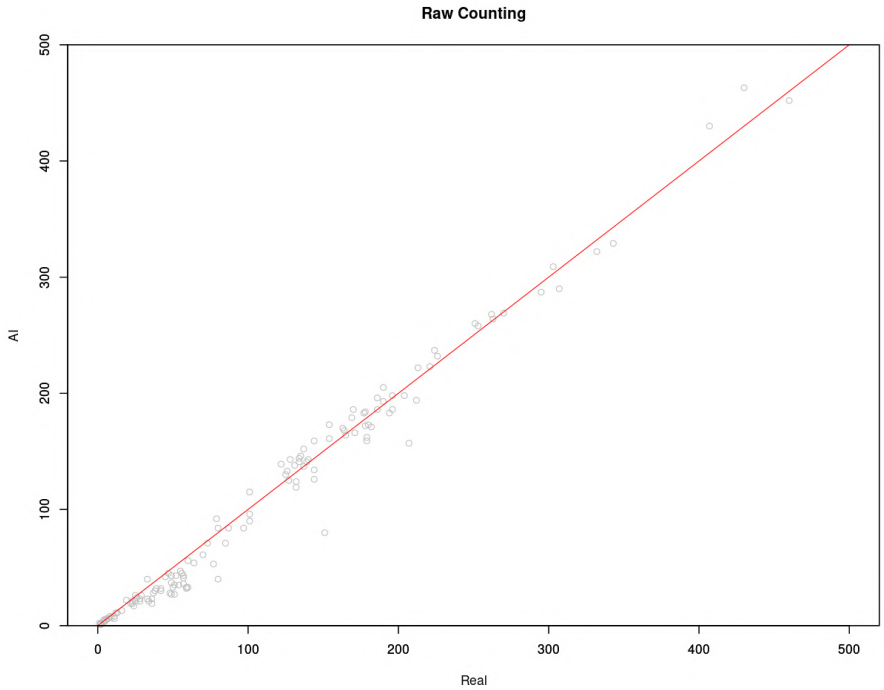


Figure 4.74: Counting graph for using the EfficientNet backbone

Chapter 5

Wearable Edge AI towards healthcare applications

In this chapter, we evaluate the applications developed towards our second stakeholders. We defined them as researchers, physiologists, medical practitioners, patients and people which require health monitoring appliances. We also developed cyber-physical applications within three branches: physical condition monitoring, smart wearable systems in the context of COVID-19, and wearable-based human activity recognition.

5.1 Physical condition monitoring in field

The first approach in this context was applying the Wearable Edge AI context to monitor workers' conditions during field activities. This application relates to the healthcare monitoring section, but interfaces with the first interests. This happens as this study was developed as an integration from both stakeholder groups.

5.1.1 Requirements

The first step in this analysis is evaluating the requirements for the proposed method. For this matter, we display a version of the co-design diagram presented in Figure 5.1, which is a simplification of the diagram presented in Figure 3.2b.

This representation displays the need to raise the constraints for the application and classify them into the hardware, software or architectural domain. The constraints identified for this matter are:

- This solution must gather data from the users and sense environmental conditions that can affect them [*Hardware*].
- This solution must be integrated to personal protection equipment [*Hardware*].

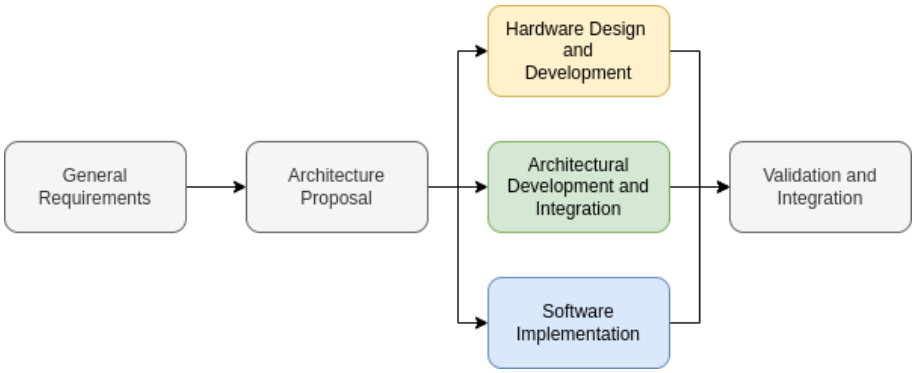


Figure 5.1: Simplified Co-design diagram.

- The wearable devices must be able to communicate with mobile applications which will interpret their data [*Architecture*].
- The communication needs to be efficient to stream the data through this local network [*Architecture*].
- Data obtained from the applications must be processed in real-time [*Software*].

In this case, we proposed a cooperative wearable system in which several users wear the proposed solution, which by itself serves for multiple purposes. In this case, often the answer for a constraint comes from the evaluation and development of others.

5.1.2 Context Overview

Figure 5.2: Wearable Device Prototype, proposed in [6].

The system is built upon a prototype suggested by Amorim et al. [6], as illustrated in Figure 5.2. We have not suggested any modifications to the hardware configuration as our goal is not to verify the device in isolation, but rather to assess the implementation of several devices. The sensors were previously employed in close proximity and evaluated during the prototype development phase.

We investigate the earlier proposal and validation of this solution. The development of innovative technologies necessitates a methodical procedure. In this context, we adhere to the notion of Multiple-User Cooperative Wearable Systems, which involves investigating the utilization of a single device across many applications.

Therefore, we utilize the understanding provided by the current sensors to outline innovative applications that can leverage this data to produce fresh insights. This study examines the progression of this system towards a Continuous Writing System (CWS), and more particularly, a French-English CWS (FR-CWS). This procedure adheres to the established protocols for the development and verification of wearable devices and systems. Moreover, we provide a hypothesis in which this system is capable of serving many devices.

This subsection provides the background in which we implement the suggested remedy. Initially, our wearable device gathers pertinent data from both the person and the immediate surroundings. Initially, this wearable device was specifically developed to provide assistance to workers in the open-sky mining sector [6]. Therefore, this study examines a more extensive hypothetical situation in the field of forest studies. The creation process was informed by typical functions seen in wearable gadgets and field research. Under such circumstances, a team with expertise in multiple disciplines collects diverse information from the gadget. The individuals comprising this team are:

1. A medic, monitoring the team's health [200, 201];
2. A biologist/ecologist, measuring sensor values for his research [202–204];
3. A physiologist, studying the physical effort in each kind of task [205–207];
4. A navigator, cross-checking the global position with physical or virtual maps [208–210].

Every member of this group wears identical safety field research equipment and has access to the collective data of the entire team's gear. Furthermore, every member is provided with a Smartphone application that is tailored to their specific professional responsibilities, along with specially-designed features. This architecture aims to utilize the adaptable nature of the data generated by the device to cater to many topics using the same wearable, as a distinguishing feature of a multi-user Context-Aware System (CWS).

5.1.3 Wearable computing requirements

This subsection provides a concise summary of the prerequisites for the specific circumstances in which this wearable system operates. The initial set of criteria is derived from the prevalent characteristics of wearable devices:

- The device must not block the user's common movements;
- It must provide information from sensors;
- It must detect context changes.

Moreover, the integration of new equipment in field research necessitates the development of innovative AHPs. Therefore, it is preferable to integrate the wearable system with widely-used protective equipment. Utilizing these gadgets hinders the need for implanting a new safety device. Therefore, we selected a safety vest as the reference point for our suggested approach, using it as a case study.

From now on, the team will be designated as M for the medic, E for the ecologist, P for the physiologist, and N for the navigator. In order to meet the requirements of the interdisciplinary team, this gadget must be equipped with sensors that offer:

- Body temperature and humidity (M, P);
- Heart rate and blood oxygenation (M, P);
- Environmental luminosity, temperature and humidity (M, P, E);
- Global Position and Altitude (E, N);
- Muscular Effort (P);
- Body Motion (M, P);
- Safety lights (M, E, N, P).

Each professional can obtain the required information from all the crew members, in an application designed with the specifications from each of them. This feature comes with the flexibility of CWS.

5.1.4 Device Architecture Description

In the preceding section, we outlined the prerequisites for the suggested CWS. Within this section, we will outline the structure and characteristics of the prototype utilized in this particular scenario. The diagram in Figure 5.3 illustrates the primary components of this prototype.

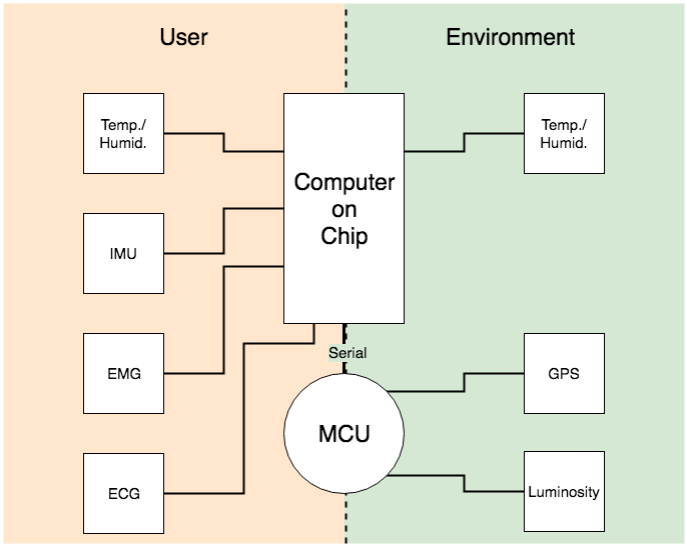


Figure 5.3: Proposed device architecture.

As mentioned, this device was previously presented by Amorim et al. [6]. It has sensors to monitor both the user and some environmental variables. These sensors are:

- User Sensors:
  - **Temperature/Humidity Sensor**—This sensor monitors the temperature and humidity internally. This sensor connects reading digital data from a GPIO pin;
  - **IMU Sensor**—This sensor monitors the user's body motion. It communicates using the I2C bus;
  - **EMG Sensor**—This sensor monitors the muscular effort from the user. It transmits the measured data using GPIO monitoring;
  - **ECG Sensor**—This sensor monitors the heart rate and blood oxygenation. It communicates using the I2C bus.
- Environmental Sensors:
  - **Temperature/Humidity Sensor**—This sensor monitors the temperature and humidity externally. This sensor connects reading digital data from a GPIO pin;
  - **GPS Sensor**—This sensor gathers the global position data and transmits it to the computer board through the MCU. The MCU uses a serial connection to communicate with this board;
  - **Luminosity Sensor** - This sensor gathers luminosity data and transmits it to the computer board through the MCU. The MCU uses I2C connection to communicate with this sensor.

In addition, the vest offers the possibility of activation. The MCU can utilize the brightness sensor to trigger the activation of safety lights in the event that it detects a dimly lit environment. To enhance comprehension of the arrangement of sensors in this device, Figure 5.4 depicts the spatial positioning of its components. As mentioned in the introduction of this section, this device underwent prior testing and validation in [6]. The safety vest accommodates all the elements without obstructing or causing discomfort to the user.

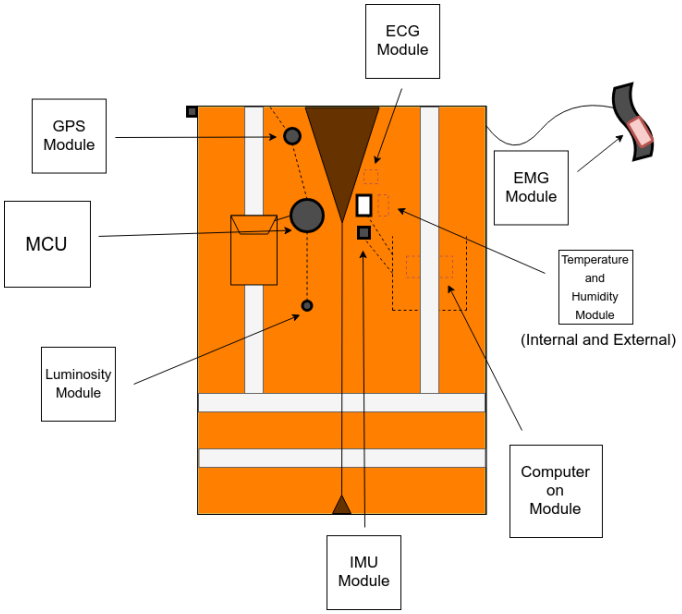


Figure 5.4: Wearable device illustration.

To have a more comprehensive understanding of the system's capabilities, it is crucial to chart the data acquisition time for each sensor. This information is also utilized to generate the simulated devices during the validation phase. In relation to this issue, we have condensed the duration of the procurement process for each sensor that is included in the prototype created by Amorim et al. [6]. The summary is presented in Table 5.1. Furthermore, we anticipate that the consuming applications will engage in additional post-processing of the data obtained from the sensors. Thus, the provided sampling times take into account the shortest interval necessary to obtain this essential information.

Table 5.1: Sampling time ratio for each sensor.

	Variable	Sensor	Sampling time
User	Temperature and Humidity	AM2302 (DHT22) [211]	2 s
	IMU	MPU6050 [212]	0.125 ms
	EMG	Myoware	System Sampling Rate [213] (~5 μs)
	ECG	MAX30100 [214]	1 ms
Environmental	Temperature and Humidity	AM2302 (DHT22) [211]	2s
	GPS	FGPMMOPA6H [215]	0.1 s
	Luminosity	TSL2561 [216]	2.5 μs

This analysis used the provided information from the sensors and the computer-on-chip datasheets [211–216]. Finally, we consider that all sensors were properly calibrated in a previous assembly stage with the correct methods. For a broader comprehension, we also present the general characteristic of the calibration process for each sensor. The AM2302 sensor needs a chemical process in a closed chamber for calibration, as it is an hygrometer–thermometer [217]. The MPU-6050 is a 6-DoF IMU, which requires a 3-axis motion and spinning movements [218]. As an EMG sensor, Myoware must be calibrated before the usage, considering reference levels of contraction signals [219]. As a GPS module, FGPMMOPA6H requires a factory calibration according to its antenna [220]. TSL2561 is a lux meter, and therefore must also be previously calibrated with its response curve [221]. Finally, MAX30100 is a pulse-oximeter, and thus requires factory calibration with the light wavelength response [222].

5.1.5 System Architecture

In the previous subsection, we outlined the key characteristics of the wearable devices that would form part of this system. Here, we introduce the suggested architecture for CWS. The architecture was derived from the collaboration of a multidisciplinary team. The crew consists of a medic (M), an ecologist (E), a physiologist (P), and a navigator (N), as previously stated.

This system adheres to the principle of Multiple-User Collaborative Writing System (CWS). In this particular setting, numerous individuals utilize a shared gadget. The increase in flexibility occurs at the application level, where the data obtained from the wearable devices will undergo post-processing.

Figure 5.5 depicts the integration of the CWS architecture. Each crew member possesses an application that collects data from the sensors over wireless communication protocols, as previously stated. The applications extract pertinent information for each individual involved in the process from the comprehensive dataset.

Wearable devices in contemporary applications can be regarded as Internet of Things (IoT) nodes, as evidenced by several studies [223–228]. Therefore, within the framework of this study, each wearable device is also represented as an Internet of Things (IoT) node that forms a Fog-Radio Cloud Wireless System (FR-CWS) architecture. Every crew member has the ability to utilize a smartphone that is connected to the network or a gateway in order to access and get data from each wearable device.

Every crew member application requires a specific sampling interval based on their professional specialty. However, the sampling rate must take into account both the interval at which the sensor readings are taken and the inherent delay in

communication, which is common in this type of system. Prior to proceeding, it is crucial to comprehend the communication alternatives for constructing a wireless network. Mahmoud and Mohamad [229] categorize the network protocols for the Internet of Things (IoT) based on the communication range. They use the following terminology:

- Proximity (up to 10 m);
- Wireless Personal Area Network (WPAN) (up to 100 m);
- Wireless Local Area Network (WLAN) (Up to 1,000 m);
- Wireless Neighborhood Area Network (WNAN) (up to 10 km);
- Wireless Wide Area Network (WWAN) (up to 100 km).

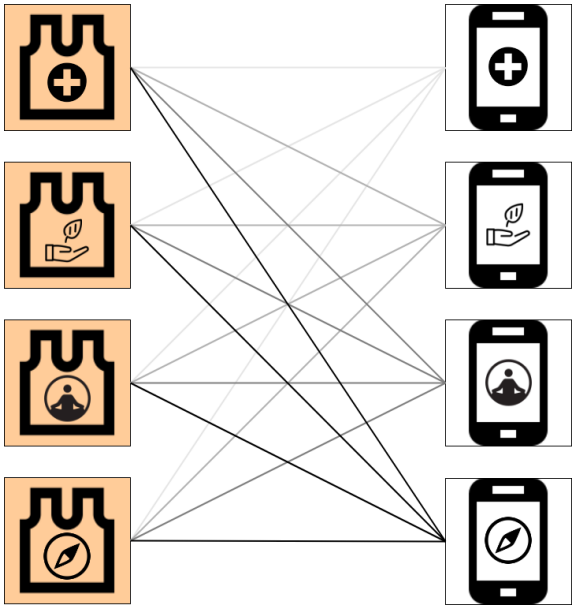


Figure 5.5: Field research cooperative wearable system architecture.

According to the conjectured scenario, we desire that the users have access to each other data in distances within the ranges of WLAN and WPAN. For this matter, this work also provides some examples of technologies in use in these scenarios. Table 5.2 presents these options.

Table 5.2: WPAN and WLAN connectivity technologies.

	Range	Technologies
WPAN	up to 100 m	bluetooth LE, ZigBee, Thread (6LoWPAM), Z-Wave, ANT^{+}, WirelessHART, ISA100.11a (6LoWPAM), EnOcean, ...
WLAN	up to 1,000 m	802.11a/b/n/ac, 802.11af, 802.11ah & 802.11p

In a test scenario, we use the WLAN as connection mean, as it is enough to manage the information exchange for a crew working in proximity.

5.1.6 Evaluation Methods

Given that the proposed system operates as a dispersed device network, a crucial element of its architecture pertains to its communication limitations. In order to verify the effectiveness of this approach as a collaborative multi-node system, it is necessary to assess the practicality of the suggested system. Furthermore, it is vital to comprehend the constraints associated with the data accessibility within this network prior to formulating any algorithmic suggestion. Therefore, in this assessment, we establish a mathematical framework grounded in Quality-of-Service (QoS), akin to the models proposed by Boukerche and Samarah [230] and Silva and Oliveira [192]. This Quality of Service (QoS) test assesses the availability of information, taking into account the time restrictions that need to be considered when developing consuming applications.

At first, we assume that the transmission time is divided in equally-sized timeslots, represented by the set $T = \{t_1, t_2, t_3, ..., t_m\}$, where $t_{i+1} - t_i = \lambda$ for $1 < i \leq m$.

Definition 8. Let $D = \{d_1, d_2, d_3, ..., d_n\}$ be the set of n wearable devices present in the network.

Definition 9. Let $p_j = d_r...d_s$ be a data acquisition pattern, where each d_i element is a device from D ($d_i \in D$).

Definition 10. Let $P = \{p_1, p_2, p_3, ..., p_k\}$ be a set of k desired observation patterns.

Definition 11. Let $F(p_a, x)$ be the number of observations of a p_a pattern within an x number of timeslots.

Definition 12. Let $F(p_a)$ be the total number of observations of a p_a pattern in the whole test period.

The quality parameter $Q_s(p_i, k)$ for a pattern $p_i \in P$ expected in a k number of timeslots is defined by the following equation:

$$Q_s(p_i, k) = \frac{F(p_i, k)}{F(p_i)}. \tag{5.1}$$

Each wearable device in our scenario will be emulated by utilizing a Raspberry Pi Zero W single-board computer connected to the network. The selection of these devices was based on their role as primary hardware nodes in the original prototype created by Amorim et al. [6].

Professionals may desire to receive communications from individual members, pairs of members, trios of members, or the entire crew in any given situation. If we define our pattern set P as the set of all permutations of arrangements from the elements of D according to this rule, then the size of our full pattern set is denoted by S:

$$S = P(4,1) + P(4,2) + P(4,3) + P(4,4),$$
$$S = \frac{4!}{3!} + \frac{4!}{2!} + \frac{4!}{1!} + \frac{4!}{0!},$$
$$S = 64.$$

Consequently, we conducted 30 simultaneous tests for each device, aiming to acquire identical results, for every conceivable combination of P. During each test, the consumer applications simultaneously attempt to retrieve data from each sensor based on the potential message patterns of P. Every consumer application is created as a universal client within the network, capable of connecting to each node through the network gateway and retrieving its sensor data. Furthermore, during each test, the consumer applications will endeavor to collect identical data. Throughout the entire process, the wearable devices will be readily accessible, as they are a necessary component of both wearable technology and the Internet of Things.

Upon completion of the testing, we conduct a thorough analysis of the data to assess the quality elements as per the specified timing requirements. The string holding the simulated data will be released at regular intervals of 4.2 seconds for simulation purposes.

5.1.7 Results

In the previous part, we introduced the validation test set for the suggested case-study. Here, we examine the developed modules and practical features of the implemented tests. The test set being proposed is a simulated representation of the real network environment. To address this issue, we created a server application on four distinct computer-on-chip nodes, which are interconnected by a WLAN network. Furthermore, we set up four distinct computers as clients to retrieve data from the server nodes.

The client nodes have the capability to execute a single query at any given time. This prevents a single application from causing the sensor nodes to become unresponsive after they are connected. As previously stated, every client application operates using identical code, and each sensor node carries out the same server function. The devices are disseminated throughout the local wireless network, as depicted in Figure 5.5.

We also said that the test encompasses every potential query case for each message. The evaluation approach takes into account the equation 5.1, which calculates the quality factor for a time frame consisting of k timeslots. According to the information provided in Table 5.1, it takes approximately 4.2 seconds to collect data

from all sensors. Therefore, we opted to examine the discrete-time slots in intervals of $\lambda = 1$s. Furthermore, the quality factor must take into account the quantity of queries in a pattern, as each question will require a minimum of five timeslots to be resolved. Consequently, once the k factor has been determined, the analysis employs the subsequent equation to compute the precise k_l factor, where n represents the quantity of queries in the p_j pattern, $p_j \in P$:

$$k_l = n.k.$$

Furthermore, as previously stated, there exist a total of 64 distinct patterns when examining the individual combinations of each device d_i from the set D. Hence, we gathered the durations for acquiring each pattern $p_j \in P$ that consists of a distinct arrangement of devices.

Each of the four client devices performed this method 30 times, simultaneously and concurrently retrieving data from the server nodes. Ultimately, we examined the data by taking into account various values of k, commencing with $k = 5$. The results obtained from the tests are displayed in Figure 5.6.

The purpose of this study is to comprehend the limitations of the proposed architecture and its components. The QoS factor represents a proportion of the overall occurrences of a pattern. Consequently, we display the outcomes in the form of percentages.

The QoS tests reveal that a minimum of nine timeslots, equivalent to nine seconds, is required to ensure the full delivery of patterns. As previously stated, a minimum of five slots is required to generate and transfer the data. Therefore, it is essential for each study application to take into account the utilization of acquisition rates ranging from 5 to 9 seconds per device.

The results show that the average QoS factor for $k = 5$ is 77.8%. The mean quality of service (QoS) factor for a given value of $k = 6$ is 95.0%. The mean quality of service (QoS) factor for a given value of $k = 7$ is 99.4%. The mean Quality of Service (QoS) factor for a value of $k = 8$ is 99.9%. The quality factor is 100% when the value of k is 9. As anticipated, decreasing the value of k in the analysis leads to a deterioration in the quality factor outcome. This occurs as a result of the delay in sensor acquisition and the simultaneous use of the network. The graph labeled as Figure 5.7 illustrates the trend of the average factor for each value of k.

Another conclusion from this result is that the gain is small starting from $k = 7$. Increasing the number of timeslots from $k = 5$ to $k = 6$ elevates the average quality factor by 17.1%. Increasing from $k = 6$ to $k = 7$ increases the quality factor in 4.4%. Increasing from $k = 7$ to $k = 8$ raises the quality factor only by 0.5%. Finally, increasing from $k = 8$ to $k = 9$ only elevates the quality factor by 0.1%.

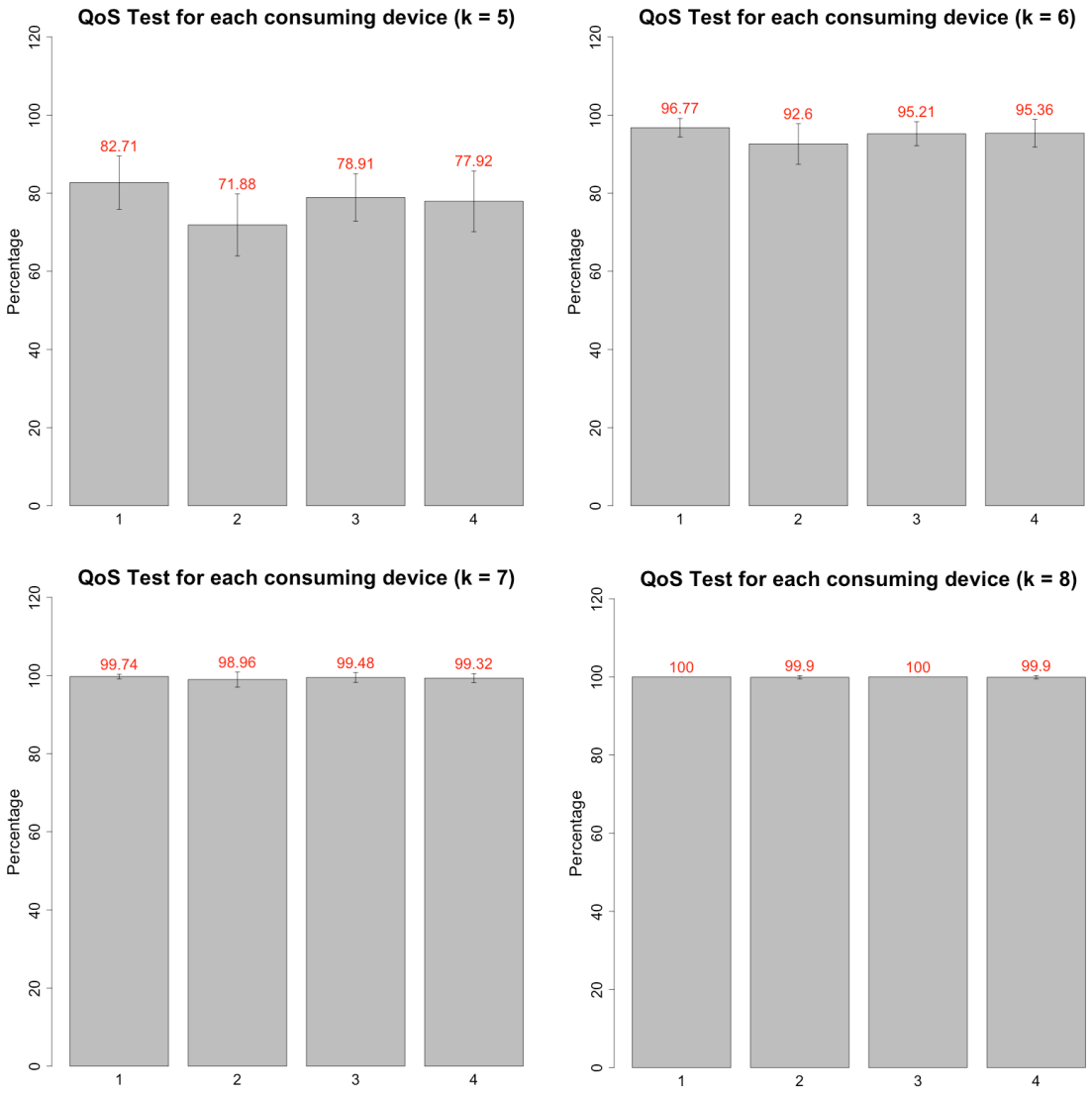


Figure 5.6: Results of the QoS tests on each device.

The initial inference drawn from these test results confirms the validation of the network architecture. The test demonstrates the ability to simultaneously query wearable node devices for messages in an IoT-like application using wearable devices, even with the anticipated delay in sensor readings. This outcome validates the practicality of developing a Field Research Cooperative Wearable System within the specific case study. Moreover, when developing applications to utilize the data from the wearable nodes, it is important to take into account the timing limitations specified by the QoS test. Typically, the devices should aim to achieve an acquisition rate of approximately 7 seconds per node. Within the context of this text, employing a 7-second capture rate ensures a Quality of Service (QoS) factor of 99.4%.

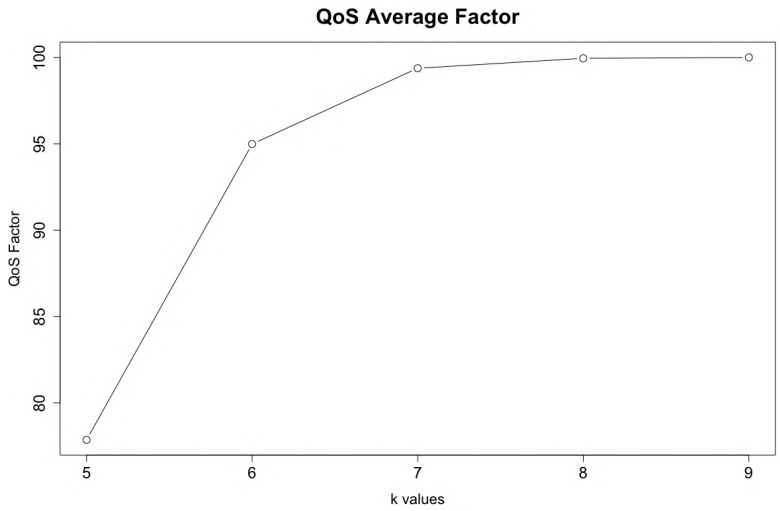


Figure 5.7: QoS average factor for each k value.

5.2 Smart wearable systems in the context of COVID-19

Due to advancements in hardware downsizing, the integration of Graphics Processing Units (GPUs), and the incorporation of Artificial Intelligence (AI) capabilities in System On Chip (SoC), wearable computers can now be categorized as edge computing devices as well (Chen, 2017). This viewpoint suggests improved and adaptable electronics and modular Computers-on-Chips. These devices have the capability to achieve greater involvement in local processing operations [232]. In addition, due to their advanced networking capabilities, they are capable of transmitting data with a greater level of abstraction to applications that are based on edge servers or cloud platforms [233].

Context-awareness is a crucial element in wearable computing [234]. Upon initial examination, this information pertains to the identification of alterations in the environmental circumstances within pervasive applications (Surve, 2017). Furthermore, an integral aspect of context-awareness is the monitoring of the user's conditions, sometimes referred to as user-awareness.

Kliger and Silberzweig [237] state that COVID-19 is a viral illness caused by a newly discovered strain of coronavirus. The primary recognized symptoms include fever, cough, myalgia, and weariness. According to Prachand et al. (2020), a significant issue in combating the pandemic is the healthcare personnel' susceptibility to contamination hazards. However, Kliger and Silberzweig [237] state that face masks and face shields are included in the list of recommended personal protection equipment (PPE) used by healthcare professionals to prevent contamination.

5.2.1 Requirements

The first step in this analysis is evaluating the requirements for the proposed method. For this matter, we display a version of the co-design diagram presented in Figure 5.8, which is a simplification of the diagram presented in Figure 3.2b.

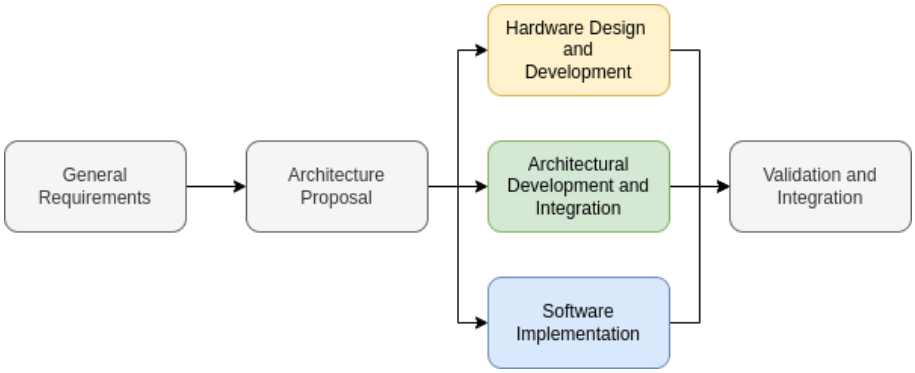


Figure 5.8: Simplified Co-design diagram.

This representation displays the need to raise the constraints for the application and classify them into the hardware, software or architectural domain. The constraints identified for this matter are:

- This solution must gather data from the users and sense gather information using external markers [*Hardware*].
- This solution must be integrated to personal protection equipment [*Hardware*].
- The wearable devices must have low current consumption for an increased autonomy [*Hardware*].
- The integration of data from multiple devices must happen within and edge server [*Architecture*].
- The communication needs to be efficient to stream the data through this local network [*Architecture*].
- Data obtained from the applications must be processed in real-time [*Software*].

In this section of our work, we propose the architecture for a novel wearable appliance to help the professionals in the frontline of the COVID-19 engagement. The proposed appliance has two main goals. The first one is to gather information from environment signals using a camera as a smart sensor. The other one is to monitor the medical professionals' health conditions using internal measurement sensors. We also prototyped a version of the proposed architecture to test its feasibility and features.

5.2.2 Architecture Proposal

Here, we utilize this knowledge to suggest a new advanced structure for the wearable device. We choose the protective face shield as the foundation for this architecture development. This item is a safety face shield that has been modified to include a Head-Up Display (HUD). This viewpoint advocates for the implementation of an additional protective barrier, as advised by the World Health Organization (WHO), to safeguard against direct contamination [239]. Moreover, the suggested appliance aims to offer context-awareness, taking into account both the environmental conditions and the user's awareness.

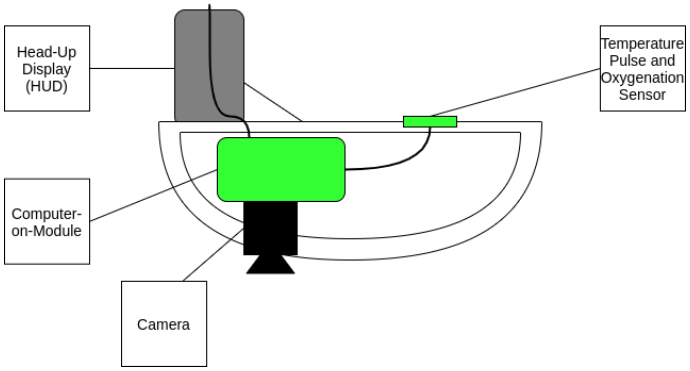


Figure 5.9: Schematic View of the Proposed Prototype

The suggested design consists of three main components: an environmental sensing element, a health monitoring sensor, and a heads-up display (HUD) interface. The integration of all these parts is achieved by the utilization of an integrated computer-on-module, which is powered by a battery. Figure 5.9 depicts a schematic representation of the arrangement of elements. To perceive the surroundings, we suggest use a camera. Initially, this sensor enables remote access to the medical records of patients.

The internal application utilizes a QR-Code to identify the patient and presents the most pertinent information via the Heads-Up Display (HUD). To detect the user's health status, we suggest utilizing a pulse-oximetry and temperature sensor. This module offers data regarding users' temperature, blood oxygenation, and pulse conditions over the course of usage.

The user awareness interface is an integrated heads-up display (HUD). The device utilizes a compact Organic Light-Emitting Diode (OLED) screen with a partially reflective surface and a lens to create the intended transparent look. Due to the compact size of the display, it can only accommodate a restricted amount of information. These pieces integrate with an ARM-based single-core computer-on-module. This board possesses wireless networking capabilities for seamless integration with the local network. This functionality enables the transfer of the user's data and the

retrieval of information regarding patients that are kept in the local servers.

5.2.3 Prototyping and Validation Tests

This section presents the produced prototype to validate this idea and the tests used to evaluate its performance.

Prototype Description

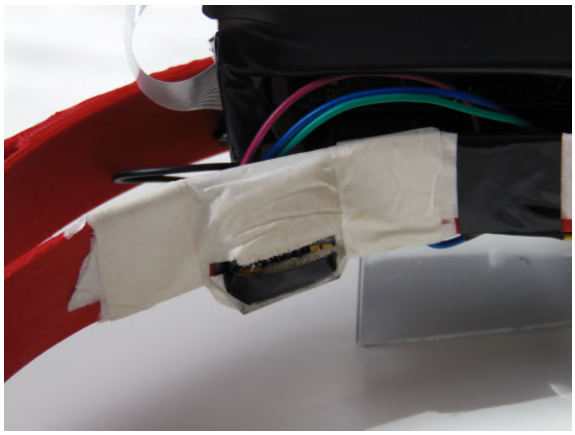


Figure 5.10: Pulse-Oxymeter and Temperature Sensor Placement

Initially, we commenced the production of the prototype by utilizing a 3D-printed face shield foundation. Volunteers utilize this basis to fabricate face shield masks. Upon this foundation, we established all the essential components required to develop the suggested application. The computer-on-module utilized was a Raspberry Pi Zero W. The solution is equipped with a solitary ARMv7 processing core, which is integrated within a Broadcom BCM2835 chipset. This CPU has a clock frequency of up to 1GHz. The device is equipped with 512MB of RAM and a wireless board that supports an 802.11 b/g/n WLAN connection, Bluetooth 4.1, and BLE protocols. We utilized a plastic enclosure to organize the computer-on-module and established connections to the remaining components via cables. The total weight of the solution, including the battery, is around 200g.

We employed a MAX30100 pulse-oximeter to detect and monitor the patients' health status. This sensor supplies data necessary for computing the users' heart rate, blood oxygen saturation, and body temperature. The device is powered by a 3.3V output generated from the computer-on-module and communicates through an I^2C/SMBus serial link. The positioning of this sensor in the prototype is seen in Figure 5.10. We employed a Raspicam V2 module[1] to perceive the surroundings. This device features an 8-megapixel picture resolution, with video formats of 1080p,

[1] `https://www.raspberrypi.org/documentation/hardware/camera/`

Figure 5.11: HUD See-through Display

720p, and 480p. It can be accessed using the V4L2 Linux driver. The device offers a horizontal field-of-view of 62.2 degrees and a vertical field-of-view of 48.8 degrees. This device establishes a connection with the central computer through the utilization of the MIPI camera serial interface.

The user interface is a Heads-Up Display (HUD) that presents information directly in front of the user's right eye. In this example, we utilized a 96x64 pixels OLED display[2] enclosed within a 3D-printed casing. This module utilizes an SPI serial interface for communication. A semi-reflexive membrane was employed to form the reflexive surface, positioned in front of an acrylic layer. We positioned a lens with the accurately computed focal length to exhibit the information at the precise distance. Figure 5.11 illustrates the information that may be accessed using this particular model of the transparent display.

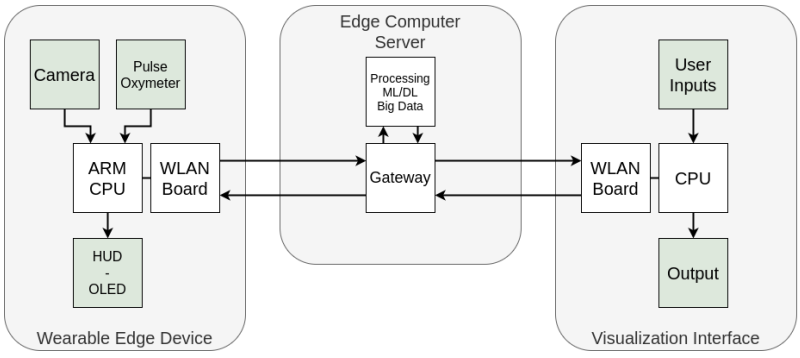


Figure 5.12: Data Flow for the Proposed Prototype

The computer-on-module consistently collects data from its sensors. Utilizing its wireless functionalities, it transmits this data to a server for storage and retrieves specific information based on the observed data. Ultimately, it generates the feedback frame of the HUD screen based on the received response and the data from

[2]https://img.filipeflop.com/files/download/Datasheet_SSD1331.pdf

the sensors. Figure 5.12 illustrates the flow of data within the systems, based on the information depicted in this section.

Validation Tests

Here, we introduce the test set that was utilized to validate the proposed solution. In order to evaluate the system, we will conduct tests on several elements. A wearable appliance refers to a gadget that has limited energy resources available to it [228, 240]. Furthermore, the utilization of processing power is a significant limitation in this particular situation [241]. Additionally, we aim to verify some functioning features of the system. Therefore, our test set takes into account:

1. A **current consumption profiling test**, to observe how the proposed device behaves due to processing charge;
2. A **full battery discharge test**, for probing the energy constraint and autonomy;
3. A **functional validation test**, to observe how the system reads the provided data.

In the initial two tests, we employed a data collecting device to deliver instantaneous data on consumption by utilizing a sensor and a microcontroller. The sensor utilized was an INA219 current consumption sensor, whereas the microcontroller employed was an Arduino Uno. The configuration for this probe is shown in Figure 5.13.

To obtain a singular output result, we calculate the mean measurement of 20 samples taken at an approximate rate of 100 samples per second. The ultimate sampling rate for acquiring a singular value was roughly 4.5 samples per second.

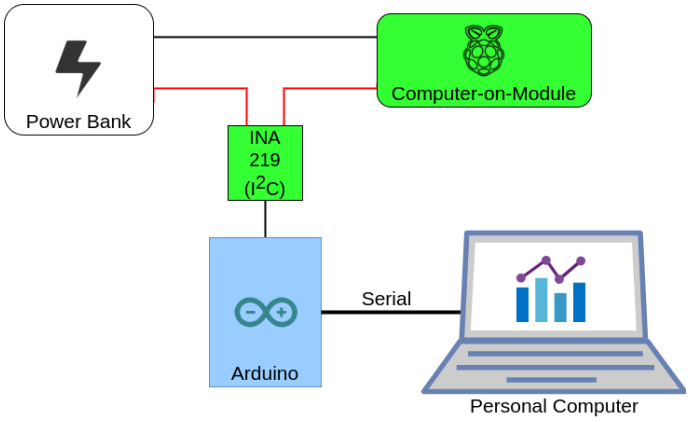


Figure 5.13: Current Consumption Probe Configuration

In the **current consumption profiling test**, we observe how the system consumes energy in different stages of its functioning. For this matter, we performed a

current consumption test considering various stages of the system functioning. We performed a 210-second run using a 5V power source. In this experiment, the device runs the following states in the approximate time intervals:

1. Device off – 0s-10s;
2. Boot – 10s-65s;
3. SSH enabled – Idle – 65s-110s;
4. Run application – 110s-180s;
5. SSH enabled – Idle – 180s-200s;
6. Device off – 200s-210s.

Using this metric, we anticipate examining the present consumption of the many potential states of the system. Furthermore, we anticipate gaining a more comprehensive understanding of the system's energy limitations. In order to enhance understanding, we also conduct the subsequent test as described.

During the **full battery discharge test**, we anticipate examining two distinct facets. Initially, our objective is to assess the level of autonomy of the system based on a certain power source. Additionally, we aim to assess the consistency of consumption during the entire duration of execution. This assessment takes into account both quantitative and qualitative characteristics. The current consumption, without significant increases, primarily demonstrates the resilience of the behavior. Furthermore, it is imperative to ascertain the level of autonomy exhibited by this system, taking into account the various constraints that have been mentioned.

Additionally, we do a **functional validation test**. Through an analysis of the system's essential components, we assess its practicality and explore potential extra features. We evaluate sensing technologies that are capable of detecting both user-awareness and environment-awareness. Our analysis focuses on the extraction and transmission of information to an external edge server appliance. We facilitate the utilization of Edge AI to assess the characteristics of the provided data within this device.

5.2.4 Validation Tests Results

In this section, we present the results for these tests and preliminary discussions and conclusions.

Current Consumption Profiling Test

The first proposed experiment is the current consumption profiling test. In this scope, we want to describe the functioning of the system throughout different stages. For this matter, we divided the test time into six stages: Device off (1), Boot (2), SSH enabled - Idle (3), Run application (4), SSH enabled - Idle (5), and Device off (6). This test represents roughly a "symmetrical" startup, execution, and shutdown from the prototype. Figure 5.14 displays the results obtained from this experiment.

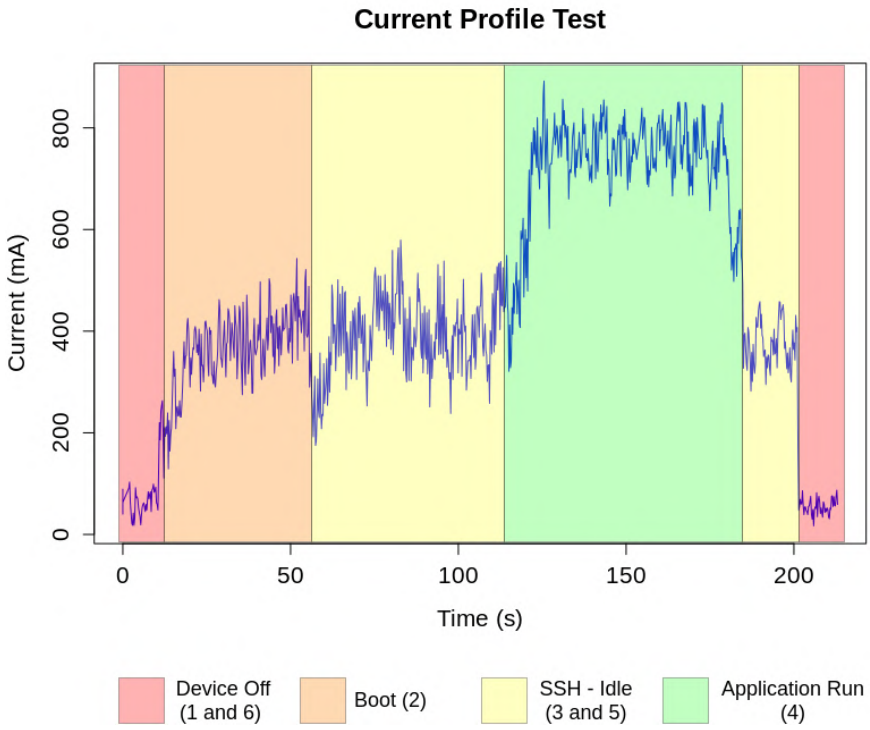


Figure 5.14: Current Consumption Profiling Test Result

In red, we display the probe readings when the system was off (Stages 1 and 6). These results display some noise but roughly represent the "zero-state" of this system. In orange, we display the current consumption results in the system boot (Stage 2). In this case, it is possible to see that the reading values increase until reaching a stable state.

In yellow, we display the results for the "SSH enabled - Idle" stages (3 and 5). In these intervals, the system was on, and the ssh connection was established. Nonetheless, the application was not running yet, configuring an idle state. At the connection establishment, we observe some instability in the current consumption, which reaches a stable state right after.

Finally, in green, we display the current consumption result for the application run time. In this case, the device starts the application, acquiring and transmitting data. In this case, the prototype is consuming the fully required resources. From

the data, it is possible to see that the current increases during the system start-up, reaching a stable state after some seconds. The system also displays a slope decrease on the current, reaching a stable level in the shutdown's idle state. Table 5.3 displays the average current consumption for each stage.

Table 5.3: Profiling Test Results

	Current (mA)		
	min.	max.	avg.
Stage 1	3	86	45.53 ± 22.27
Stage 2	56	529	331.4 ± 86.6
Stage 3	232	565	384.4 ± 65.8
Stage 4	744	927	831.4 ± 32.6
Stage 5	268	444	359.5 ± 45.0
Stage 6	3	73	40.63 ± 14.0

Full Battery Discharge Test

Following the analysis of the current consumption for each stage, we proceeded to conduct a discharge test. For this experiment, we utilize a compact battery as a potential power source for the device and assess its duration and mean electrical usage. Figure 5.15 exhibits the measurement outcomes during the entire duration of the test.

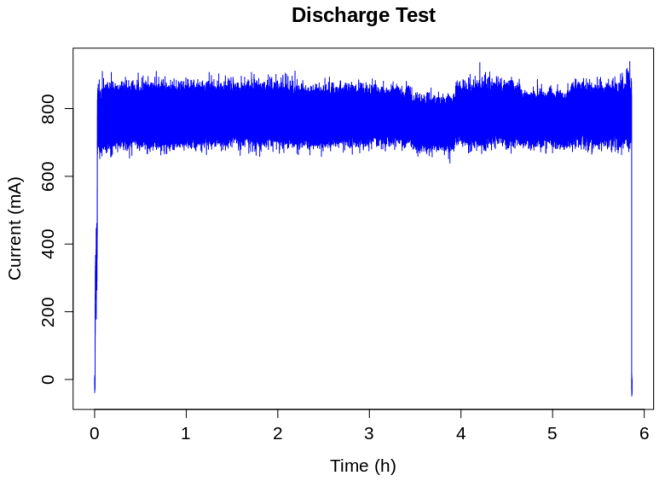


Figure 5.15: Discharge Test Result

For this test, we utilized a 4000 mAH power bank with a low weight as the primary battery for the system. The mean current during the test was 779.3 ± 58.7 mA, with a recorded maximum of 939 mA. The prototype exhibited consistent

performance throughout the whole test, indicating a high level of robustness for this device. Additionally, the autonomy was approximately 6 hours of continuous operation. These results support the findings of the profiling test, confirming the viability of the solution and its predictable behavior. This information also enables the choice of a suitable power source based on the current needs of healthcare personnel.

Functional Validation Test

Ultimately, we conducted a functional validation of the prototype to assess its viability in a situation that closely resembles the context of the end user. The intended appliance aims to retrieve two distinct types of information. Our objective is to utilize a camera to perceive data from the surroundings and employ a built-in sensor to retrieve information from the user.

Initially, we assessed the practicality of the external sensor. At first, it functions solely as a detector for a QR code to access information from other advanced sensors. Therefore, our hypothesis regards the wearable camera as a sensor that extracts images. The system obtains and transmits frames to an edge computing server for processing.

To address this issue, we incorporated a streamer program into the device. Therefore, the edge computing server has the capability to promptly establish a connection with the wearable device, get a single frame, and analyze it in order to locate a QR code containing an identifying query. Consequently, we created a straightforward program capable of extracting data from the wearable device and analyzing it for a QR code. The ultimate outcome for this example is depicted in Figure 5.16.

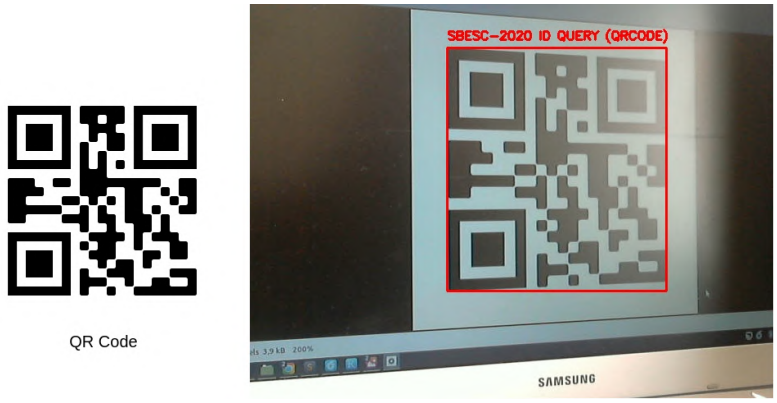


Figure 5.16: QR Code Acquisition Validation

Following the validation of the external data collecting procedure, it was necessary to also validate the acquisition of the user's health conditions data. For our prototype, we utilized the MAX30100, a device that offers data on temperature and pulse-oximetry. This application utilizes a wearable device to compute the blood oxygen saturation (SpO_2) data by analyzing measurements obtained from infrared

and red light emitting diode (LED) pulses. Hemoglobin exhibits differential absorption of red and infrared light depending on the presence or absence of oxygen. Figure 5.17 depicts the sampling readings of the probe sensor, which were taken at a rate of around 23 samples per second for a duration of approximately 15 seconds. The blue line represents the measurements of the infrared pulses, while the red line represents the measurements of the red light pulses.

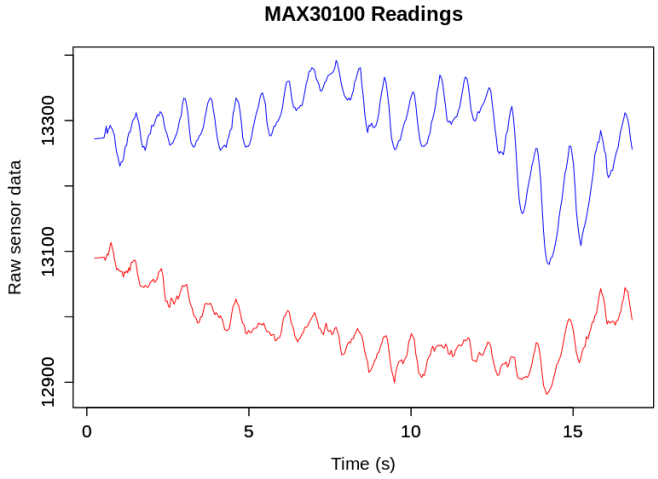


Figure 5.17: MAX30100 Probe Readings

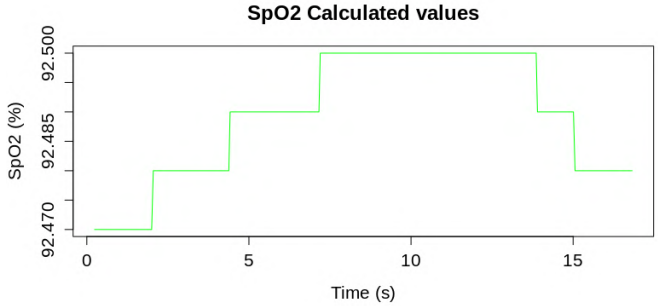


Figure 5.18: SpO_2 Readings Obtained from the Computer-on-Module

Usually, the data can be analyzed by considering the AC components that have been adjusted for the DC components [242]. The computer-on-module processes the readings as SpO_2. Figure 5.18 displays the data obtained from the computer-on-module. The pulse information can be acquired by simply enumerating the number of crests within a designated time interval. The collected data can be subsequently assessed on the edge computing server appliance using machine learning techniques to examine the temporal sequences.

The analyses conducted on the prototype ensure the functional and non-functional validation of every part of the proposed architecture. We conducted a functional

analysis to confirm that the prototype can be successfully integrated into the proposed appliance by verifying its key components. Non-functional analysis involves identifying the limitations that must be met for the proposed system to operate effectively.

5.2.5 Edge Computing - Architecture Proposal

Typically, edge devices are used in a wireless network environment to carry out cooperative duties, which is commonly linked to the Internet of Things (IoT) concept [243]. Therefore, the initial determination in this suggestion for the system's architecture is the networking environment. Given the need for rapid adaptation and deployment of structures like mobile field hospitals [244], an optimal approach would be to utilize a portable solution for swiftly establishing a management framework. The solution's network type is a Wireless Local Area Network (WLAN), as determined by the required connectivity ranges [229]. The main planned design is depicted in Figure 5.19.

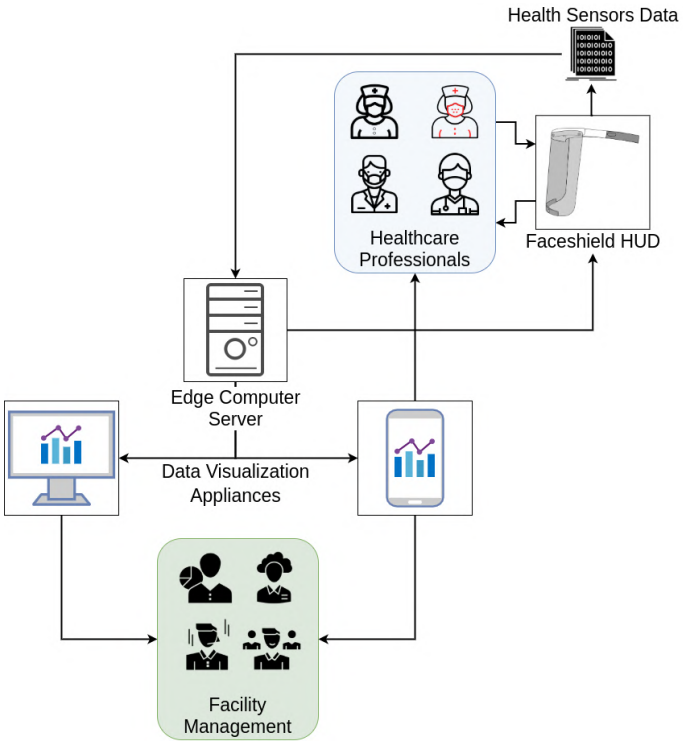


Figure 5.19: Overview of the proposed architecture

The WLAN-server functions as a nearby edge computing facility that serves as an access point for wearable devices and computer connections. This module collects data from all the sensor nodes in the network, conducts processing activities to extract information and insights, and delivers high-level feedback through wearable, mobile, and computer terminal interfaces. This option enhances the capabilities of

the Internet of Things appliances [245], enabling to reach higher-level insights from the acquired data.

Health sensors, in this particular context, refer to wearable devices that are worn by healthcare practitioners. Our solution involves sensor nodes consisting of sensors connected to single-core computer-on-modules, which are powered by power banks. They carry out basic processing duties such as collecting data, preparing it for analysis, and facilitating two-way communication to transmit the collected information and receive responses. Medical practitioners and facilities administrators can utilize computer stations and mobile devices as interfaces as well.

Edge Computing Devices Description

Figure 5.20: Face shield HUD Prototype.

We created a wearable gadget, specifically a prototype of the face shield HUD, for this project. We have already constructed and verified this prototype [52] as an obligatory safety apparatus in numerous healthcare establishments. The prototype that was created is shown in Figure 5.20.

The primary goal of this wearable gadget is to track and assess the user's health status and provide remote data retrieval. Using this device, we evaluate many aspects of the proposed solution. The primary pertinent part of the solution is the computer module. Wearable devices typically have limitations in terms of battery consumption and cost. Therefore, the initial choice is to utilize a solitary-core ARM-based computer-on-module as a foundation for creating the solution. A crucial component of the computer-on-module is a network board that has the capability to connect to a WLAN.

Another important aspect to take into account is the arrangement of the sensor. Historically, one of the primary advantages of wearable devices is enhancing the user's knowledge of their immediate surroundings and personal well-being [246].

This prototype is equipped with two distinct sensors. The first device is a pulse-oximeter and temperature sensor, which collects data on the user's health state. Another sensor present is a camera, which is used to gather data from the surrounding environment.

Within the healthcare facility, the camera may retrieve data from patients' medical records by utilizing QR codes. Lastly, the feedback interface is the final part of it. In this regard, we have suggested the implementation of a head-up display (HUD) to furnish the user with comprehensive information. A Heads-Up Display (HUD) is an optically transparent display that presents information directly within the user's line of sight [247]. Augmented reality enhances usability and context awareness by allowing hands-free interaction and presenting information in the user's visual field.

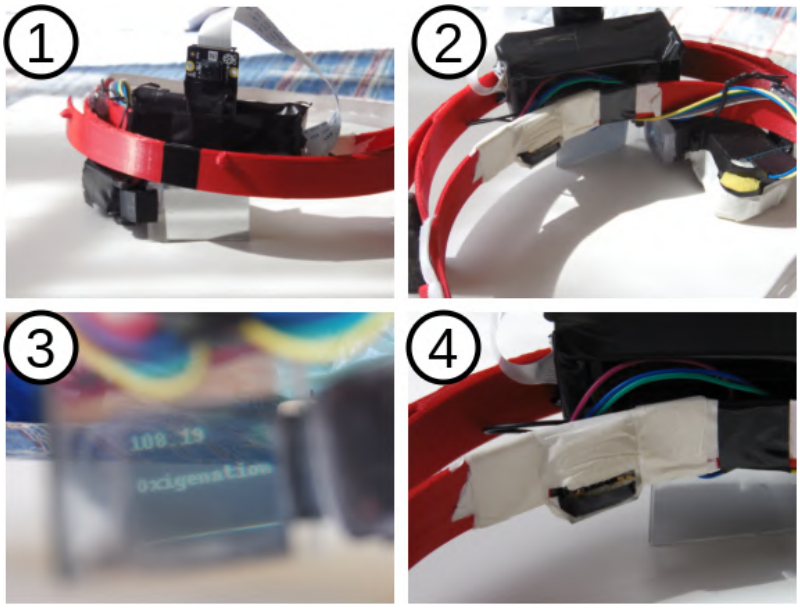


Figure 5.21: Overview of the elements of the produced prototype

The prototype's elements are depicted in Figure 5.21. In Figure 1, the primary mount is visible, featuring the camera positioned at the front and the HUD located below. In **2**, we can see the rearview, displaying the box containing the computer-on-module, the HUD system, and the monitoring sensors' location. Within **3**, the perspective from the HUD showcases an array of high-level data intended for the user's perusal. Ultimately, in **4**, the positioning of the health monitoring sensors and a portion of the wiring is visible.

Terminal and Mobile Interfaces

Another component of the suggested architecture is the management interfaces. These applications assist facility managers in the decision-making process. This technology facilitates the surveillance and identification of initial indications of pol-

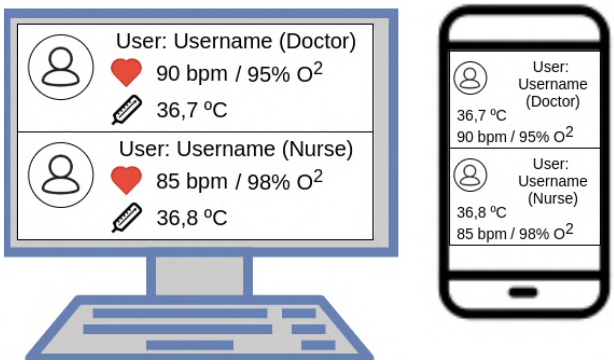


Figure 5.22: Proposed Interfaces Illustration

lution in healthcare practitioners [248]. The instances of real-time monitoring of healthcare workers' situations are depicted in Figure 5.22.

Edge Computer Server

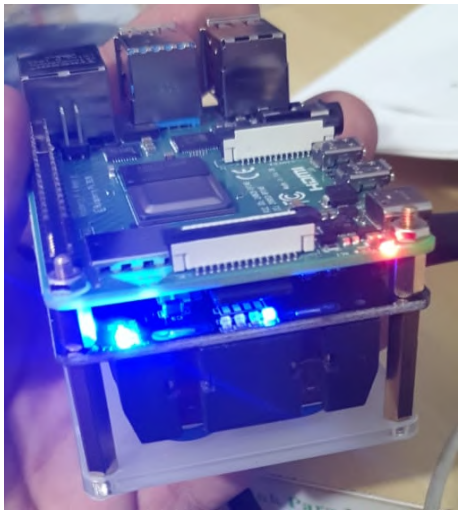


Figure 5.23: Edge Computing Server Node

The network edge consists of devices that manage computational operations in edge computing. The edge device must be engineered to efficiently handle such activities and meet requirements such as reliability. The integration of IoT components into the system occurs via an edge computer server. This device carries out two distinct functions inside this system. The initial component serves as a conduit for receiving, storing, and disseminating data from the wearable devices and the management terminal interfaces. The second objective is to input the obtained data into information extraction algorithms. In order to accomplish this goal, we suggested utilizing a portable device known as an edge computer server, which possesses the specified hardware prerequisites depicted in Figure 5.23. The depicted image showcases a Raspberry Pi4 model equipped with 4GB of RAM, 64GB of stor-

age, and WiFi capabilities. The tests conducted with this gadget demonstrated a level of autonomy of approximately 12 hours under conditions of intensive usage.

System Integration

In order to comprehend the operation of this system, we moreover furnish an examination of the data flow. The diagram in Figure 5.12 illustrates the key components of this architecture and depicts the flow of data. This depiction facilitates comprehension of the anticipated conduct of each constituent in accordance with its individual elements. Additionally, it is beneficial to analyze the temporal limitations associated with each stage of the procedure.

The suggested architecture consists of three distinct sorts of elements. The initial component is the wearable edge computing device, exemplified by the faceshield HUD. This component collects data from the sensors and performs initial processing to transform the data into meaningful information.

The focal component is the Edge AI Computer/WLAN Server. This component serves as a central hub for data processing in an embedded edge computer, while also overseeing the network connections. In addition to serving as an access point on a computer, it is also capable of executing parallel data fusion and analysis activities. This method yields comprehensive data for the other components of the design. Lastly, the management interface serves as the final component in this arrangement. The generic interface block in Figure 5.12 represents this element. Both mobile devices and computer terminals consist of essential components including user inputs, WLAN connectivity, processing, and output.

5.2.6 Experimental Tests

A relevant aspect of distributed architectures is networking performance. This feature directly affects the quality of the provided services in IoT-based systems, having consequences in the system's real-time capabilities [249]. Thus, the experimental setup evaluates the timing constraints for the data flow and processing.

Real-Time as Quality-of-Service

To evaluate these aspects, we perform a QoS-based timing constraint test. The experiment was designed as a QoS formalization, presented on similar studies concerning IoT and Wireless Sensor Networks [192] to evaluate soft real-time constraints as network timing constraints.

At first, we divide the experiment time in discrete intervals, as the set $T = t_i, i \in \mathbb{N}$, where $t_{i+1} - t_i = \theta$, where θ is a constant sampling time. The soft real-

time deadline will be represented by ϕ, where $\phi = k \times \theta, k \in \mathbb{N}^*$. From these primary statements, we establish the following definitions:

Definition 13. Let $D = d_i$ be the finite set of nodes consuming and producing data from the middleware node, where $i \in \mathbb{N}$;

Definition 14. Let $E = e_i$ be the finite set of events that each node performs, where $i \in \mathbb{N}$;

Definition 15. Let $L = l_{d,e}$ be the length of time interval that the node d takes to perform an event e, where $d \in D$ and $e \in E$;

Definition 16. Let $P = p_i$ be the set of patterns of events to be observed in the devices, where $p_i = E_i$, $E_i \subset E$ and $i \in \mathbb{N}$;

Definition 17. Let $O = o_i$ be the finite set of observations of a certain pattern $p_i \in P$ on the devices;

The equation that represents the elapsed time λ to observe a particular pattern $p_i \in P$ is:

$$\lambda_{o_i} = \sum l_{d,e_k} | \forall e_k \in o_i, o_i = O_{p_i} \tag{5.2}$$

In this case, each device in the network composition can have its single ϕ_i soft real-time deadline. Given this equation, let $\hat{O}$ be a subset of O, where $\lambda_{o_i} \leq \phi_i$, $\forall o_i \in \hat{O}$. Finally, given the sets O and $\hat{O}$:

Definition 18. Let N be the number of elements on the set O;

Definition 19. Let N_h be the number of elements on the subset $\hat{O}$;

The following equation will represent the quality factor Q_f:

$$Q_f = \frac{N_h}{N} (\times 100\%) \tag{5.3}$$

The nodes will try to gather or update data from the server node in parallel on each test. This result represents how often the nodes execute a pattern of events without violating the soft real-time constraints. This perspective allows experimenting on how increasing the number of devices producing and consuming data affects the network quality factor.

The proposed experiment is divided in 3 stages:

- **Stage 1:** Defining the soft real-time constraint;

- **Stage 2:** Evaluating the effect of stressing the system by increasing the number of edge devices;
- **Stage 3:** Evaluating the effect of stressing the system by increasing the number of client interfaces.

During **Stage 1**, we execute a simplified version of the appliance, which includes a single element from each class depicted in Figure 5.12. To determine the minimum ϕ_i soft real-time constraint, we utilize the provided data to achieve a quality factor of 0.95 (95%). This limitation will serve as a benchmark for assessing the system's performance while scaling up the number of devices, with a block length of $\theta = 2$ ms. Furthermore, we utilize this data to assess the intrinsic timing restrictions of each device for the purpose of simulation.

During **Stage 2**, we utilize the limitations established in the initial stage to assess the performance of the system as the quantity of edge devices is augmented. The computer-on-module boards will be utilized to replicate the behavior of various devices, as done in the production of the prototype. The timing for each job on the simulated device is derived from the assessment conducted in the previous stage.

In **Stage 3**, we assess the impact of increasing the quantity of terminal interfaces. To address this issue, we create many simulated terminals as separate processes on a machine that is linked to the network.

These stages serve as a means of validating the proposed architecture and providing an overview of the limitations when it comes to scaling up the number of devices and customers in this system.

Server Behavior Evaluation

For the second experiment, we consider the influence of patients and medical professionals in the environment. Also, in this case, we evaluate the timing aspect. We evaluate how the overload of patients and medical devices in the network influence the server capability of answering requests. For this matter, we consider the dataflow presented in Figure 5.24.

In this instance, we replicated the actions of numerous customers that generated and utilized data from the edge server. The patient appliance is regarded as a sensor for measuring temperature and pulse-oximeter readings, and it has the same temporal constraints as in the previous stage.

From this standpoint, the solution has five distinct groups of elements. The primary and pivotal component is the edge computing server. The system operates an application that receives connections from several customers, storing their data and providing it for requests from management apps in the wireless local area network. The management clients consist of the second and third aspects. The patient

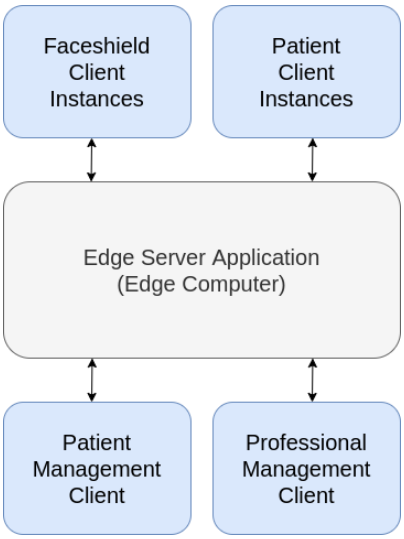


Figure 5.24: Data Flow for the Experimental Setup on the Second Test

management client retrieves data from a specific patient by using their name as an identifier. The key worker management client collects data from all professionals employed at the facility. Ultimately, the fourth and fifth aspects represent the occurrences specifically tailored for clients in the healthcare device industry. They operate in a similar manner, as they both necessitate data collection from identical sensors.

In order to assess the efficacy of this system within the network, we conducted an analysis of two distinct situations. Initially, we augmented the quantity of instances for the face shield device from 5 to 50, while maintaining a consistent count of 20 patient instances. Next, we conducted a second test scenario in which we augmented the quantity of patient device instances from 5 to 50, while maintaining a consistent number of 20 face shield device instances. In all cases, we ran one instance of the patient management client and another instance of the key workers' management client.

The parameter for this response is the server's response time, which is measured throughout the entire execution. In this regard, we assessed the mean response time and the rolling average during program execution. We conducted both tests for around 600 seconds, modifying the payload on the variable that induces stress in the test every 60 seconds.

5.2.7 Results

Devices Internal Constraints Evaluation

Firstly, we assess the inherent timing restrictions of the device. This stage represents the initial phase in establishing a validation environment that closely replicates

the conditions encountered in real-world scenarios. In this context, we provide a description of both the server apps and the clients, which include terminal and wearable devices. Furthermore, we empirically ascertain the temporal limitations in the prototype in order to establish authentic simulation environments.

Let us begin by discussing the server program. The server stores the latest data from all the clients in its memory for the experimental group, and then transmits the entire information to the asking terminal clients. Additionally, the server monitors the temporal limitations for experimental reasons.

In order to facilitate the connecting of several clients, we have created a server that utilizes multiple threads. Each thread is tasked with managing a solitary client. The application utilizes two essential processes: capturing the most recent health data from each wearable client and recording it in the log file's handler. The algorithm executed by each thread managing a single client connection is illustrated in Figure 5.25. The crucial sessions are highlighted in red blocks.

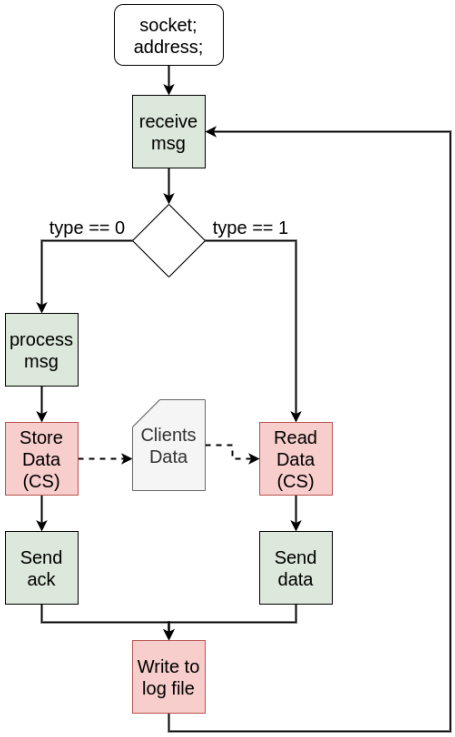


Figure 5.25: Edge Server Node Algorithm

The processes commence with the handlers responsible for the creation of connections and information management. The client address and port information serve as indices for the client data stored in memory. This device caters to two categories of users: terminal clients and wearable clients. Messages of terminal requests are identified by "type == 1", and messages from the wearable clients are identified by "type == 0".

The threads store the customers' information in a unified data structure. Hence,

the initial crucial segment involves engaging in reading and writing inside this framework. The data received from the clients is analyzed and kept. Terminals receive data from all available clients in response to requests. Ultimately, the threads utilize the identical log file. Therefore, it is likewise a second crucial segment.

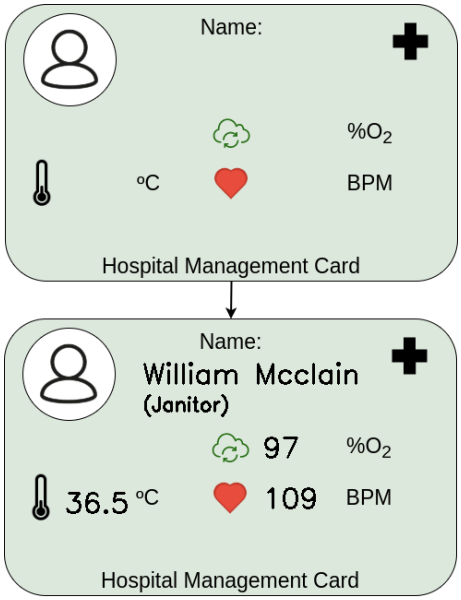


Figure 5.26: Visualization Prototype Application Example

The initial type of client is the terminal client. This client consists of two primary stages: (i) retrieving data from the server, and (ii) processing the data and presenting it on a screen using a background template and the collected information. The background and an example of a card made using simulated information are depicted in Figure 5.26. Ultimately, the threads utilize the identical log file. Therefore, it is likewise a second crucial segment.

The wearable client is responsible for three distinct tasks: (i) retrieving data from the sensors, (ii) transmitting data to the server and receiving the server's response, and (iii) analyzing the data and presenting a visual representation on the OLED display. Tasks (i) and (iii) are internal, however task (ii) relies on network connectivity. In order to ascertain the necessary timing specifications for executing this application, we conducted tasks (i) and (iii) on the prototype depicted in Figure 5.20 for a duration of 120 seconds, while monitoring the durations required to complete both internal operations. The results obtained, on average, show that:

- Task (i) takes an average time of 2.25 $\pm$ 0.10 ms;
- Task (iii) takes an average time of 11.8 $\pm$ 14.3 ms.

With this data, we created an application to emulate the client's behavior for the stress tests (Stages 2 and 3). The prototype also uses the application to determine the real-time constraint in Stage 1.

Stage 1: Real-Time Constraints Definition

In order to establish the real-time limitations, we assess the timing specifications of the tasks executed by each device inside the framework of this test. In this case, we utilize the definition of the quality factor. In this stage, we assess the basic setup by running only one piece from each class on the desired hardware. The server operates on a Raspberry Pi 3 Model B, the interface operates on a desktop computer, and the wearable application operates on an embedded Raspberry Pi Zero W, which is attached to the prototype face shield.

During the initial phase, our objective is to determine the soft real-time constraint ϕ. The experiment period is partitioned into discrete-time intervals of $\theta =$ 2 ms. To address this issue, we set a desired quality factor of $Qf = 0.95$. Next, we determine the minimal number of time blocks required to meet the desired restriction, which is a multiple of the number of blocks k needed to achieve the desired aim. For every class of device, we define a distinct limitation that corresponds to its specific set of functions.

Table 5.4: Real Time Constraint Definition Results

	Average time (ms)	Requirement (k)
WEARABLE	27.6 ± 27.4	37 blocks
INTERFACE	27.7 ± 33.3	33 blocks
SERVER	54.4 ± 50.4	65 blocks

In the subsequent phases, we employ the limitations outlined in Table 5.4 to assess the impact of exerting additional pressure on the system through an increased number of clients and interfaces. The stress tests are still being conducted on the Raspberry Pi 3 Model B server. The interface appliance is operated by one computer, while another computer executes many instances of simulated client applications.

Stage 2: Stressing the System with more Wearable Edge Devices

The primary goal of the initial test is to ascertain the impact of augmenting the quantity of wearable gadgets on network efficiency. We executed a range of 5 to 50 instances of the program that simulates the behavior of a wearable device on the network for the purpose of this test. Throughout the entire duration of the test, we ran a solitary interface application.

Initially, we derived the aggregate values for the quality factor based on the fixed elements. Based on our comprehensive testing during the entire duration, the results consistently indicate that:

- The quality factor for the interface was $Q_f = 0.989$;

- The quality factor for the server was $Q_f = 0.937$;

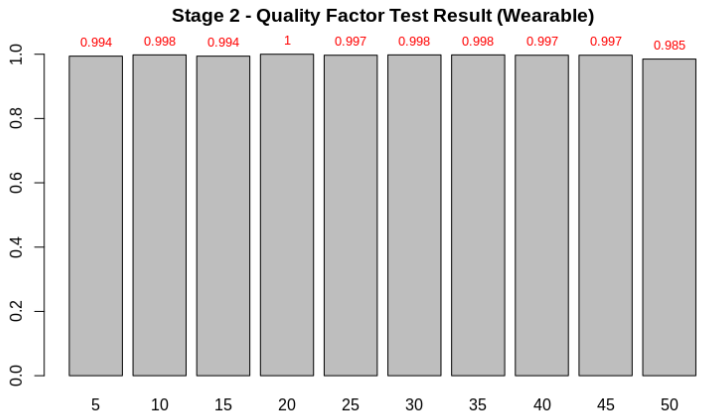


Figure 5.27: Quality Factor Test Results

The findings for the quality factor test are shown in Figure 5.27. The degradation in quality observed when scaling up the system was approximately 1%. This phenomenon demonstrates that the system design is capable of collecting and controlling data from different devices without violating the soft real-time limitation. In addition, the other limitations encountered a minimal level of compromise, so reinforcing this initial finding.

Stage 3: Stressing the System with more Terminal Devices

The purpose of this test is to examine the performance of data-intensive applications in conditions of concurrent stress. For this instance, we altered the quantity of emulated terminal devices. The terminal devices collect data from all connected clients for the management applications. We ran the application multiple times, ranging from 5 to 50 instances. Throughout the duration of the test, we simulated 20 wearable devices.

Initially, we derived the aggregate values for the quality factor based on the fixed elements. Based on our comprehensive testing during the entire duration, the results consistently indicate that:

- The quality factor for the wearable devices was $Q_f = 0.986 \pm 0.001$;
- The quality factor for the server was $Q_f = 0.885$;

The findings for the quality factor test are shown in Figure 5.28. Expanding the quantity of instances in this scenario greatly undermines the real-time capability of this program. This trend is reinforced by the deterioration in the quality factor

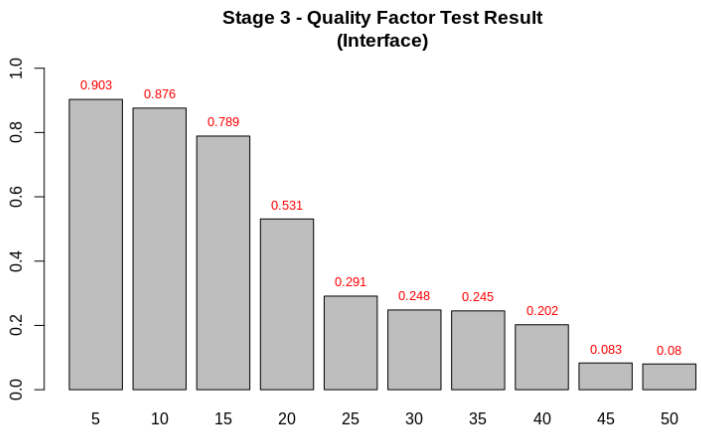


Figure 5.28: Quality Factor Test Results

on the server. This outcome suggests that it is preferable to do extensive data processing on the edge server whenever feasible, as transmitting substantial data quantities can compromise the reliability of the architecture.

Server Behavior Evaluation

This final test assesses the impact of client overload on the system's performance. The findings obtained for the testing are shown in Figure 5.29. The moving average results during the test are shown in gray. The red line represents the mean duration for the system to handle a request and transmit the outcome.

We calculated a moving average by averaging 100 consecutive samples for both tests. The average response time for a single sample in the initial test was 116.9 ± 19.11 milliseconds. The moving average indicates that the mean time is rather stable during the execution, with a few exceptional numbers. The initial outcome demonstrates the system's stability, even under increased device load. The second test yielded an average time limitation of 121.8 ± 73.28 ms. While the moving average generally indicates a similar pattern to the first test, with stability around the global average, it also identifies more outlier values, indicating a larger standard deviation.

5.3 Wearable-based human activity recognition

Within this section, we will introduce the suggested framework and its various components. Initially, it is imperative to comprehend the equipment implicated in the project. Furthermore, it is crucial in this solution to comprehend the integration process and the algorithm accountable for forecasting the actions. Our proposal involves utilizing a minimalist wearable device and a recurrent neural network (namely,

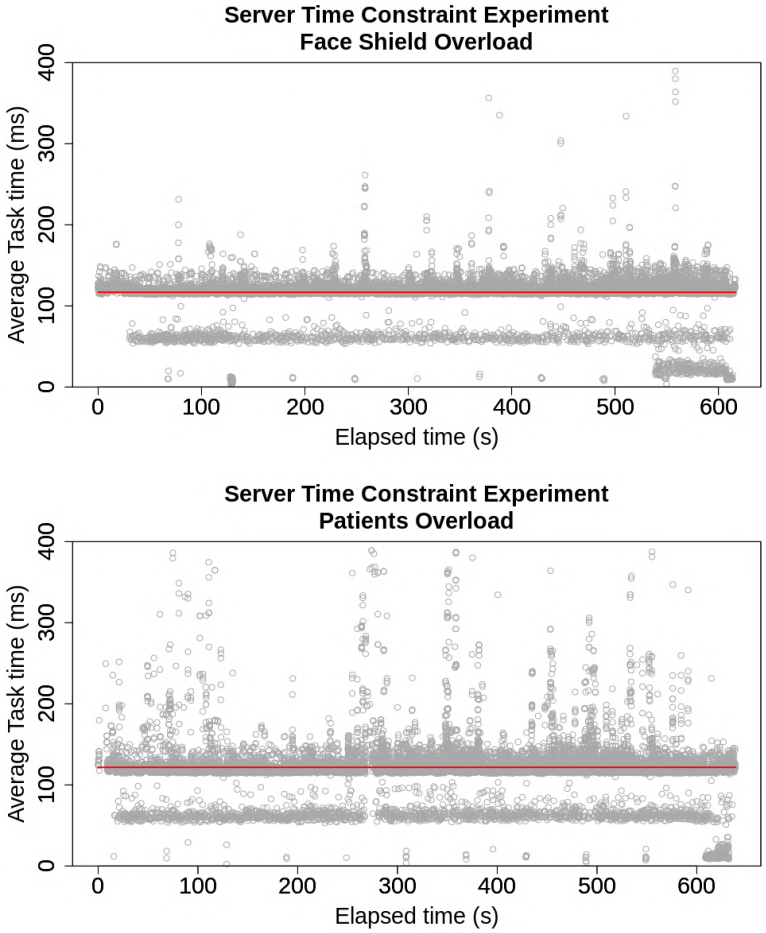


Figure 5.29: Performance Test Results

LSTM) to carry out the classification task. Ultimately, we assess the incorporation of cloudlets and mobile edge computing.

5.3.1 Requirements

To begin this study, it is necessary to assess the prerequisites for the proposed approach. To address this issue, we provide a modified version of the co-design diagram shown in Figure 5.30, which is a simplified rendition of the diagram depicted in Figure 3.2b.

This representation displays the need to raise the constraints for the application and classify them into the hardware, software or architectural domain. The constraints identified for this matter are:

- This solution must gather data from the users with small scale sensors [*Hardware*].
- The wearable devices must have low current consumption for an increased autonomy [*Hardware*].

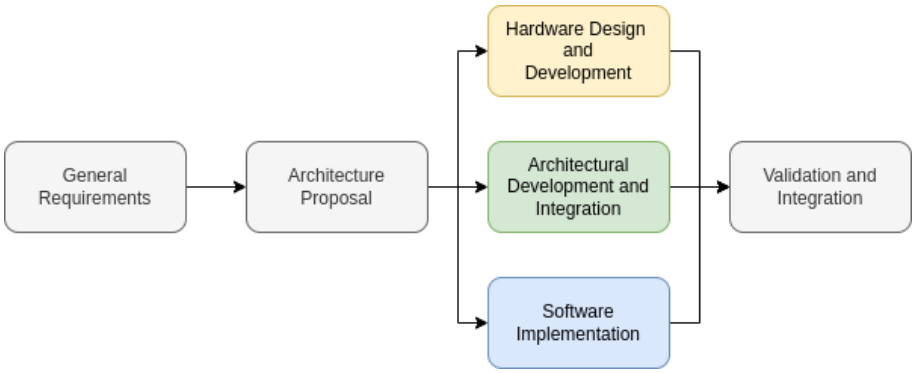


Figure 5.30: Simplified Co-design diagram.

- The integration of data from multiple devices must happen within and edge server [*Architecture*].
- The communication needs to be efficient to stream the data through this local network [*Architecture*].
- Data obtained from the applications must be processed in real-time [*Software*].

This solution shares comparable limitations with the preceding options. In this instance, we suggested incorporating small-scale wearable devices with microcontroller units (MCUs) that have wireless capabilities. We suggested employing recurrent neural networks with a wearable server to carry out the specified task.

Edge Computing Architecture

The architecture of edge computing consists of two primary levels. Within the initial stratum, uniform wearable gadgets generate data from individual users. The second layer is equipped with an edge server that receives data from the sensors, forecasts the current activity, and stores it in a database. Figure 5.31 depicts the layers of the proposed architecture.

The wearable gadget in this design possesses a straightforward configuration. The processing unit is a microcontroller (MCU) with the ability to communicate wirelessly using Wi-Fi (IEEE 802.11). This choice is made to ensure minimal processing power requirements throughout the development of this gadget. The sensor of the device consists of a solitary 6-DoF IMU. The proposed system is powered by a battery, serving as the final component in this idea. The schematic model of the proposed solution is depicted in Figure 5.32.

For the second layer, we suggest employing an edge server to execute the Edge AI algorithm. In this regard, we conducted experiments from two different viewpoints. One approach involves utilizing a cloudlet architecture to carry out this task. In this study, we conducted tests on the algorithm utilizing a desktop computer equipped

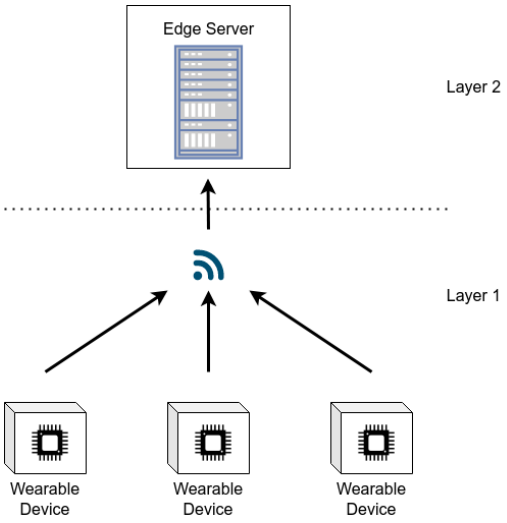


Figure 5.31: Architecture layers

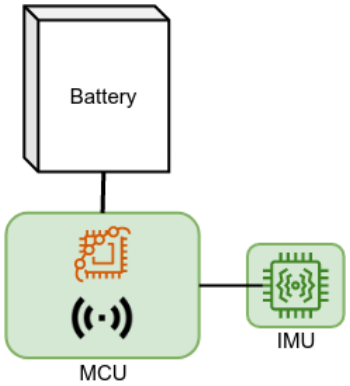


Figure 5.32: Wearable device schematics proposal

with a six-core i5-9600K CPU running at a speed of 3.70GHz, an RTX 2060 super GPU, and 32GB of RAM. We conducted tests on a mobile edge platform equipped with a quad-core ARM A57 processor running at 1.43 GHz, a 128-core Maxwell GPU, and 4GB of memory. The mobile edge platform utilized in this experiment is depicted in Figure 5.33.

Considering this proposed architecture and experimental setup, we evaluated the usage of a recurrent neural network exploring a theoretical data generation using data from a public dataset. Furthermore, we evaluate the temporal constraints for each edge server setup.

AI Model Proposal and Training

The gathered data consists of a collection of time series obtained from the sensors. For this particular scenario, we employ a Long Short-Term Memory (LSTM) model to forecast the behavior based on a collection of unprocessed time series data. Therefore, we opted for a recurrent neural network design to forecast the actions.

Figure 5.33: Mobile edge computing platform

The first stage involved selecting a publicly available dataset that possessed similar characteristics to the required dataset. We used the KU-HAR dataset [250] as our initial foundation. The dataset comprises samples from 18 distinct activities executed by a total of 90 people. The authors modified the data to achieve a sampling rate of 100 Hz, which serves as a foundation for developing the MCU-based wearable device. The data was partitioned using a sliding window of two seconds, with a half-second overlap. 90% of the data was allocated for training, while 5% each was allocated for validation and testing. Ultimately, the training set consisted of 108976 samples, the validation set contained 6111 samples, and the test set comprised 8313 samples. The table labeled as "Table 5.5" presents the collection of jobs outlined in this dataset.

We propose using a network with 128 LSTM cells in the first layer. Then the next layer is a dense neural layer with 256 neurons. Finally, the output has 18 layers with the "softmax" activation function, configuring a generative classification model. This classification results from the output that generates the highest probability. Figure 5.34 illustrates the proposed architecture.

The loss function was categorical cross-entropy. We trained the algorithm until the loss convergence. We configure the convergence condition as no improvement higher than 0.0001 for 15 epochs of training. We also reduced the learning rate as the model reached 4-epoch-plateaus. Figure 5.35 displays the graphs for loss and global accuracy during the training process.

Finally, we converted the model to the "tflite" format for embedding and testing. After defining the edge computing architecture, proposing the algorithm, and performing its training, the next step is establishing evaluation tools and metrics to

Table 5.5: Classes of activities in the KU-HAR dataset

Label	Activity
0	Stand
1	Sit
2	Talk-sit
3	Talk-stand
4	Stand-sit
5	Lay
6	Lay-stand
7	Pick
8	Jump
9	Push-up
10	Sit-up
11	Walk
12	Walk-backwards
13	Walk-circle
14	Run
15	Stair-up
16	Stair-down
17	Table-tennis

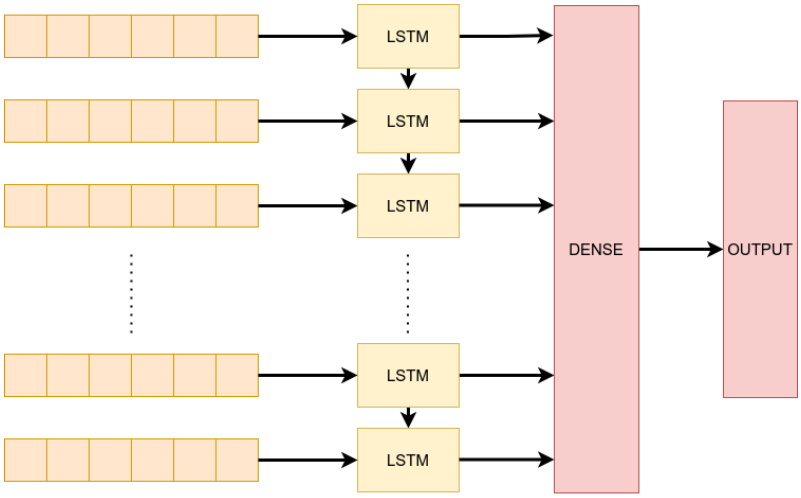


Figure 5.34: LSTM illustration

understand the performance of the proposed system.

5.3.2 Evaluation Tests

With the previously established proposal of an edge computing architecture and training of the proposed algorithm, we require the definition of tests and metrics to evaluate the experimental aspects of this work. Initially, we propose a set of metrics based on usual multi-class classification tasks. Then we propose evaluating the proposed solution's timing aspects in the cloudlet and mobile edge sets.

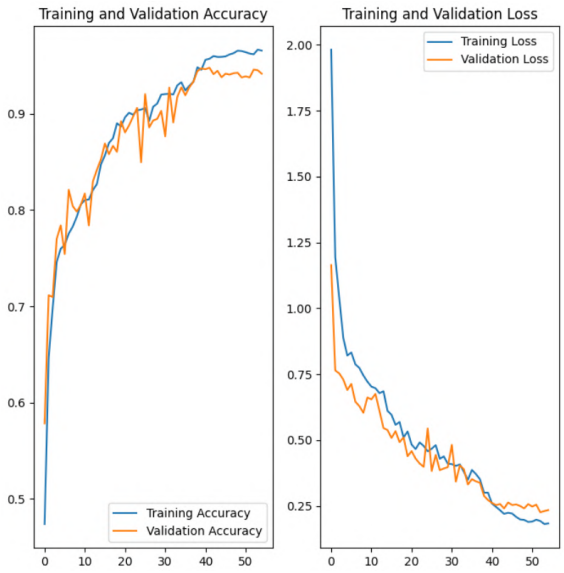


Figure 5.35: Training session result

Algorithm Evaluation

The problem stated in this case is a multi-class classification. To evaluate the model, we initially employ the most traditional multi-class classification metrics for AI. These metrics are *precision* (P_c), *recall* (R_c) and *F1-Score* ($F1_c$), calculated for each class c and defined by:

$$P_c = \frac{TP_c}{TP_c + FP_c}, \tag{5.4}$$

$$R_c = \frac{TP_c}{TP_c + FN_c}, \tag{5.5}$$

$$F1_c = 2 \times \frac{P_c \times R_c}{P_c + R_c}. \tag{5.6}$$

These equations depend on the variables TP_c (true positives for each class c), FP_c (false positives for each class c), and FN_c (false negatives for the class c). We also evaluate the global accuracy A_g, which can be calculated by the two forms below with the same result:

$$A_g = \frac{TP}{TP + FN}, \tag{5.7}$$

or

$$A_g = \frac{TP}{TP + FP}. \tag{5.8}$$

In these equations, the variables TP, FN, and FP are the sums of the variables they represent for each class, as displayed:

$$TP = \sum_{c} TP_c, \tag{5.9}$$

$$FP = \sum_{c} FP_c, \tag{5.10}$$

$$FN = \sum_{c} FN_c. \tag{5.11}$$

These metrics help to understand the quality of the model's predictions. In the next stage, we also evaluate timing constraints regarding the two edge computing perspectives.

Architecture Evaluation

Given an algorithm employing a solitary-queue prediction pipeline, it is necessary for each user to receive a prediction from the system every 0.5 seconds. If the worst-case situation occurs, the device will be faced with a complete queue that needs to be resolved within a 0.5 second timeframe. This formulation indicates the maximum number of devices that the edge server can process in an asymptotic limit. In relation to this issue, we assess the asymptotic limit in the following manner:

$$L_t = \frac{0.5}{\mu_t + 2\sigma} \tag{5.12}$$

In this context, L_t represents the ultimate number of users that the edge server can handle, μ_t denotes the average duration required for each prediction, and σ represents the variability in the prediction timings. The function is intended to encompass approximately 90% of the situations, taking into account the 2σ factor. The limit is a theoretical limitation of the system, which is based on a soft real-time constraint. It does not take outliers into account as critical circumstances. We also assess the timing restrictions in a qualitative manner using the obtained data.

5.3.3 Results and Discussion

The next part provides the outcomes acquired from the conducted experiments. At first, we evaluated the algorithm based on the predetermined metrics. Next, we assess the theoretical temporal limitations. Lastly, we will analyze the acquired outcomes and their consequences.

Algorithm Evaluation

In Figure 5.35 from the corresponding section, we presented the outcomes obtained after training the algorithm. The graphs suggest that the training procedure occurred without overfitting, based on qualitative analysis. Therefore, we assess the specified indicators at this point.

We conducted an initial assessment of the data obtained from the validation set. The results collected in this test are displayed in Table 5.6. The program achieved an accuracy rate of over 90% in correctly identifying the appropriate classification across most categories. Under the most unfavorable circumstances for this test, the algorithm achieved a prediction accuracy of 79%. The overall precision rate reached 94.7% on a worldwide scale. The results shown here surpass the initial findings reported by the dataset authors [250] (84.7%) and another study that references the work of Abid and Nahid [251] (89.5%). These findings demonstrate that this method is capable of performing the specified task to a significant degree. The findings collected in this step are displayed in Figure 5.36 and Table 5.6.

Table 5.6: Classification Metrics for the Validation Set

Activity	**Precision**	**Recall**	**F1-Score**	**Support**
Stand	0.95	0.94	0.94	617
Sit	0.90	0.96	0.93	619
Talk-sit	0.93	0.88	0.91	499
Talk-stand	0.92	0.99	0.95	478
Stand-sit	0.98	0.99	0.99	657
Lay	0.99	0.97	0.98	477
Lay-stand	0.95	0.95	0.95	443
Pick	0.97	0.90	0.93	383
Jump	0.99	1.00	0.99	158
Push-up	1.00	0.92	0.96	151
Sit-up	0.99	0.98	0.99	449
Walk	0.92	1.00	0.96	235
Walk-backwards	0.94	0.92	0.93	130
Walk-circle	0.85	0.95	0.90	81
Run	0.80	0.95	0.87	84
Stair-up	0.92	0.79	0.85	278
Stair-down	0.95	0.94	0.95	249
Table-tennis	0.97	0.97	0.97	123
Macro Avg.	0.94	0.94	0.94	6111
Weighted Avg.	0.95	0.95	0.95	6111
Global accuracy:	94.7 %			

Furthermore, we additionally verify the accuracy of the data obtained from the test set. The metrics gathered in this stage are presented in Table 5.7. Once again, the system accurately identified the correct classification with an accuracy rate of 90%. The lowest accuracy achieved by a single class in this situation was 83%. The global average achieved a minimal drop in the validation set's result, remaining at

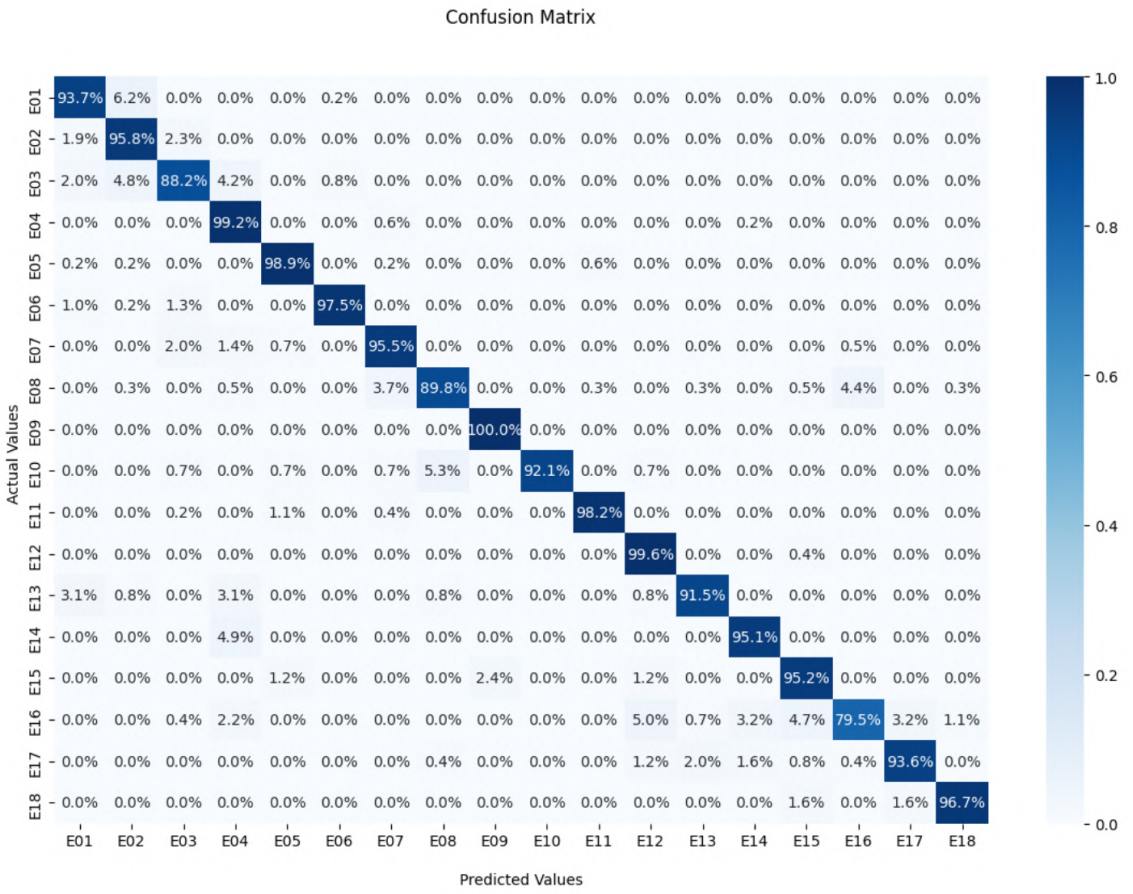


Figure 5.36: Confusion Matrix for the Validation Set

93.7%. These results provide proof that the algorithm demonstrates satisfactory generalization capabilities when making predictions on previously unknown data. The results collected in this stage are displayed in Figure 5.37 and Table 5.7.

Both tests exhibit a minimum global accuracy of 94%. These findings validate the viability of this algorithm as a predictive tool for determining the specific activity being performed by users of wearable devices. It is essential to comprehend the system's behavior from both the cloudlet and mobile edge perspectives.

Architecture Evaluation

Upon assessing the method, we further examined the number of distinct persons that may be analyzed by each modality of edge computing. We employed both a cloudlet-like architecture and a mobile edge device to evaluate the system. We conducted a test on both devices using the 8313 samples from the test set.

The preliminary experiments were carried out in the cloudlet test environment. We conducted the test and recorded the duration of each forecast. The mean forecast time was 6.7 milliseconds with a standard deviation of 0.2 milliseconds. Based on this outcome, the cloudlet system can handle a maximum of 71 people, as determined by

Table 5.7: Classification Metrics for the Test Set

Activity	**Precision**	**Recall**	**F1-Score**	**Support**
Stand	0.91	0.90	0.91	735
Sit	0.92	0.83	0.87	732
Talk-sit	0.78	0.97	0.87	747
Talk-stand	0.97	0.96	0.96	774
Stand-sit	0.97	0.98	0.98	714
Lay	0.97	0.90	0.94	678
Lay-stand	0.96	0.99	0.97	623
Pick	0.98	0.97	0.98	450
Jump	1.00	1.00	1.00	211
Push-up	0.97	0.97	0.97	183
Sit-up	0.98	0.94	0.96	498
Walk	0.90	0.95	0.92	307
Walk-backwards	0.99	0.97	0.98	129
Walk-circle	0.88	0.88	0.88	127
Run	0.96	0.83	0.89	269
Stair-up	0.99	0.93	0.96	348
Stair-down	0.98	0.95	0.96	307
Table-tennis	0.95	0.98	0.97	481
Macro Avg.	0.95	0.94	0.94	8313
Weighted Avg.	0.94	0.94	0.94	8313
Global Average:	93.7 %			

the measure defined in Equation 5.12. The distribution of these periods is depicted in Figure 5.38.

Subsequently, we conducted timing tests on the mobile edge devices. Once again, we conducted the test and recorded the duration for each forecast. The mean forecast time was 42.3 $\pm$ 0.9 ms. Based on this outcome, the maximum number of users that may be accommodated by the cloudlet system using this model is 11. The distribution of these times is depicted in Figure 5.39.

Ultimately, we graphed the data for both tests in parallel to gain a visual understanding of their relationship. Figure 5.40 illustrates the comparison of various tests. As seen, both the mean and standard deviation are greater in the mobile edge device. This outcome is anticipated, given that the computational capacity of this device is inferior to that of the cloudlet.

Discussion

During the initial round of tests, our goal was to assess the effectiveness of the proposed method based on its machine-learning metrics. In this regard, we assessed the *Precision*, *Recall*, and *F1-Score* metrics for each class, as well as the overall average precision.

The experiments demonstrate the viability of this algorithm in executing this task effectively. The method exhibited comparable performance on both the train

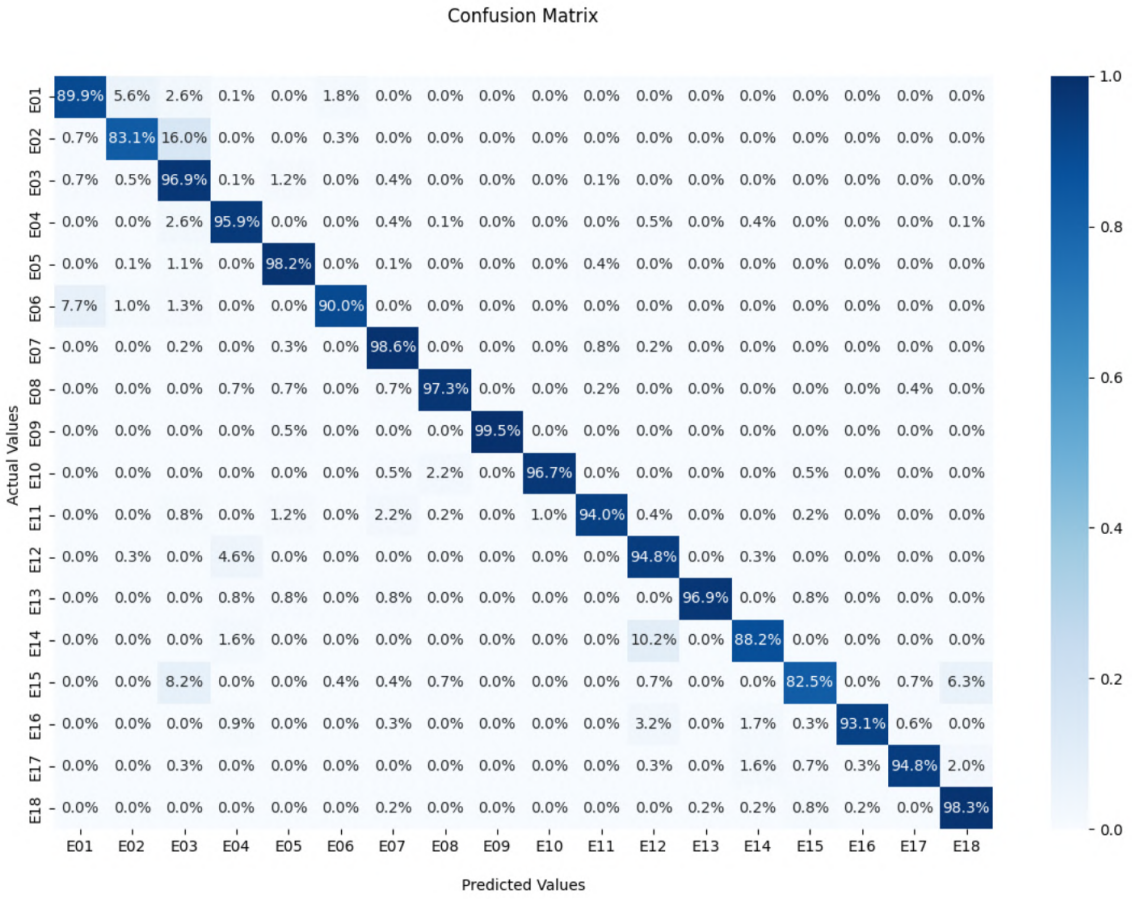


Figure 5.37: Confusion Matrix for the Test Set

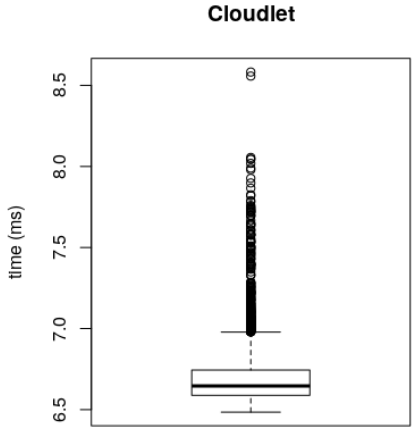


Figure 5.38: Times measured from the Cloudlet

and validation sets, yielding satisfactory results that are in line with the latest advancements in the field. In addition, we offer a thorough analysis of these indicators, addressing the lack of clarity found in most studies utilizing this dataset.

Subsequently, we assessed the asymptotic threshold for this program to run with-

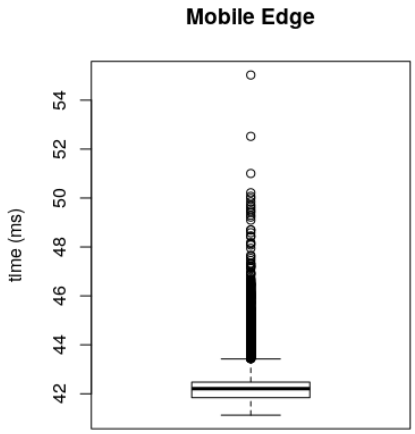


Figure 5.39: Times measured from the Mobile Edge Device

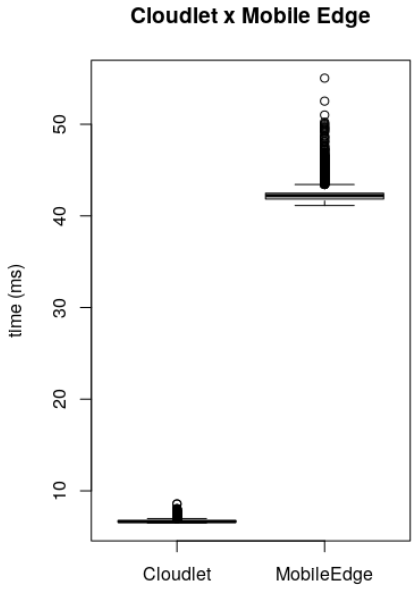


Figure 5.40: Comparison from the measurements in both tests

out any loss of data, taking into account a soft real-time situation. We have established a metric that quantifies the theoretical maximum limit, enhancing the ability to make 97.8% of forecasts without any loss of data. Unsurprisingly, the cloudlet infrastructure exhibited superior performance in comparison to the mobile edge device. However, employing mobile edge computing remains appropriate for limited populations.

Chapter 6

Conclusions and Final Remarks

This work discusses how to combine wearable computing, edge computing, and AI to define a new field of knowledge where these formerly antagonist methods can be employed together, namely Wearable Edge AI. Then, we use this new field to create new wearable technologies based on this concept. Thus, our main objective with this work is to establish a Wearable Edge AI concept and define how to design and create applications in this context.

With this text, we expect to understand how to create cooperative wearable systems based on edge computing, considering real-time aspects. We also review the HW/SW co-design process to evaluate architectural aspects. Finally, we performed a series of case studies to evaluate aspects of the risen concepts and methods.

Wearable computing, which has a central role in this work, is a concept that arose in the late 90s and early 2000s. In modern approaches, these applications use hardware miniaturization and the IoT to increase the number of possible applications within this perspective. Edge computing is another concept that uses the IoT to develop and rise. It comes to solving issues that cloud computing struggles with, such as real-time and low latency. This concept happens using proximity as a factor and in perspectives that provide the services with powerful computing, emulate the cloud services, or enhance the processing limits on mobile appliances. AI is a general concept of machines that can learn to solve real-world problems. More recently, ML applications have been pushed toward the clouds, given their requirement for high computational power.

Although wearable and edge computing have requirements that diverge from AI, an increasing number of authors are trying to combine these concepts to create novel solutions. This uprising trend still needed to have a general definition. Our case studies are some examples of how the proposed solutions seek to mesh the concepts of machine learning and edge computing.

To establish the Wearable Edge AI concept, we started with the grounding knowledge in edge computing and machine learning. Then, we defined Edge AI as the set

of methods that describes the design process and validation of solutions that combine edge computing and machine learning concepts to develop novel appliances, systems, and applications to solve real-world problems. Finally, we proposed a taxonomy method to classify Edge AI applications. The works are initially classified according to their edge computing paradigm (cloudlets, fog computing, or mobile edge computing) and then through an AI classification that divides them into "machine learning" and "deep learning" applications.

We continued by defining Edge AI and evaluating the concepts behind cooperative wearable systems. Then, we evaluated the design process to consider novel constraints based on the added edge computing process. Finally, we defined Wearable Edge AI as the set of methods that describes the design process and validation of solutions that combine wearable computing and Edge AI concepts in developing novel appliances, systems, and applications to solve real-world problems.

6.1 Theoretical Backbone

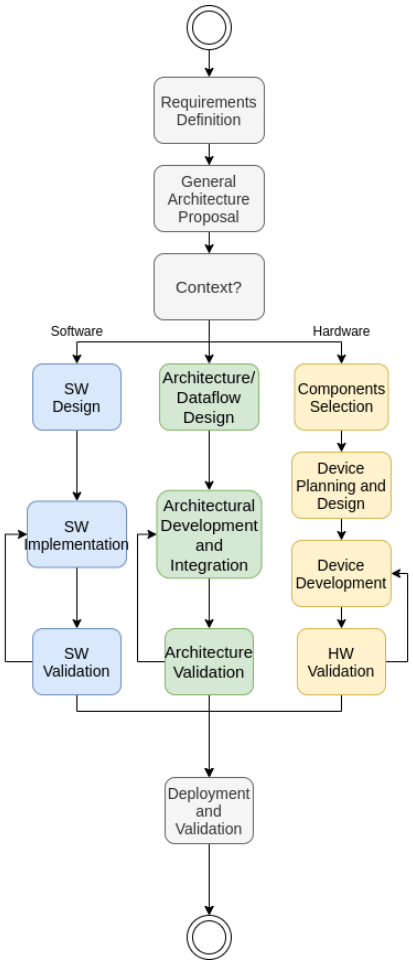


Figure 6.1: New co-design approach

Our design process definition is centered on the redefinition of the classical hardware/software co-design process. This concept comes from the development of embedded systems but required some updates to consider edge computing traits. Figure 6.1 displays the result of our evaluation.

This approach provided the means to propose solutions within the Wearable Edge AI concept for the two stakeholders from this work. Our experiments display strong evidence that this update provides a valuable tool for developing new solutions combining the concepts of Edge Computing, Wearable Computing, and AI. In all cases, we display how some of the constraints do not fit well within the hardware or software constraints, being specifically analyzed as architectural traits. Each application had a gain in evaluating the architectural traits in parallel with the hardware and software design:

- **Leaf damage estimation:** In this case, the architectural traits helped selecting the AI framework to develop the application. This appliance must be feasible to run in embedded hardware with accelerated AI, as it has the most computationally consuming algorithm within this context.
- **Evaluating and mapping diseases in forest canopies:** This application requires the concepts of cooperative wearable systems as a baseline to work. Integrating the wearable solutions within an Edge Server requires an evaluation under the optic of architectural restraints.
- **Ant distribution and counting estimation:** In this work, the algorithm design and backbone choices considered a constrained wearable edge AI environment. The model has a smaller memory footprint, with the feasibility of running in embedded edge servers.
- **Physical condition monitoring in field:** This application uses the cooperative wearable systems baseline to provide information for a multidisciplinary team. In this case, an understanding of the architecture is essential to designing an application and understanding its limits.
- **Smart wearable systems in the context of COVID-19:** This application design integrates several wearable devices within a local wireless network environment. Therefore, the proposal and results depend on the definition of architectural restrictions.
- **Wearable-based human activity recognition:** This application is another example of using multiple wearable devices within an interconnected edge computing architecture. Therefore, its capabilities also depend on the establishment of architectural restrictions.

6.2 Lessons learned

In this work, we explored a set of case studies in which the concepts of Wearable Edge AI apply. We have identified a set of lessons learned along this work. These lessons come from the conceptualization of Edge AI up to the application development. They are:

- **Wearable Edge AI requires Mobile Edge AI.** The constraints observed in developing Wearable Edge AI applications are similar to those of Mobile Edge AI. Nevertheless, these restrictions are increased by the nature of the applications.
- **Wearable Edge AI applications required a review of the Hardware/-Software co-design.** The original co-design process described how to create embedded applications but was not enough to consider the integration of co-operative devices within a network. Thus, this concept required a revision to design Edge AI applications.
- **Wearable Edge AI adds further constraints related to the architecture.** As its foundation is the Internet of Things (IoT), considering machine-to-machine communication, these applications have another set of constraints considering the architecture and machine-to-machine communication.
- **AI and computer vision usually add computationally constrained methods into Wearable Edge AI systems.** Timing is an essential constraint within Wearable Edge AI. Algorithms must have performance and avoid consuming too much computing resources during their functioning. Therefore, costly algorithms such as image processing and AI profoundly impact the applications' performance.
- **Hardware, Software, and Architectural constraints often are complementary.** Some constraints come from hardware requirements but will be surpassed by software decisions. A typical example in our cases was an edge server's processing and energetic constraint, which directly impacts the hardware choice and architecture design. Therefore, the branches of the co-design cannot be interpreted as isolated development stages.

6.3 Future Works

In future works, we suggest using this set of rules to create a Wearable Edge AI development framework. This step consolidates the concepts raised within this text

into a set of tools to develop novel cyber-physical applications. Also, we encourage other researchers to create more applications within the context of the presented stakeholders, as we displayed some of their needs within this text.

Bibliography

[1] SHINDE, P. P., SHAH, S. "A review of machine learning and deep learning applications". In: *2018 Fourth international conference on computing communication control and automation (ICCUBEA)*, pp. 1–6. IEEE, 2018.

[2] KHAN, W. Z., AHMED, E., HAKAK, S., et al. "Edge computing: A survey", *Future Generation Computer Systems*, v. 97, pp. 219–235, 2019.

[3] SHI, W., DUSTDAR, S. "The promise of edge computing", *Computer*, v. 49, n. 5, pp. 78–81, 2016.

[4] SILVA, M. C., AMORIM, V. J., RIBEIRO, S. P., et al. "Field Research Cooperative Wearable Systems: Challenges in Requirements, Design and Validation", *Sensors*, v. 19, n. 20, pp. 4417, 2019.

[5] SILVA, M., RIBEIRO., S., BIANCHI, A., et al. "An Improved Deep Learning Application for Leaf Shape Reconstruction and Damage Estimation". In: *Proceedings of the 23rd International Conference on Enterprise Information Systems - Volume 1: ICEIS,*, pp. 484–495. INSTICC, SciTePress, 2021. ISBN: 978-989-758-509-8. doi: 10.5220/0010444204840495.

[6] JP AMORIM, V., C SILVA, M., AR OLIVEIRA, R. "Software and Hardware Requirements and Trade-Offs in Operating Systems for Wearables: A Tool to Improve Devices' Performance", *Sensors*, v. 19, n. 8, pp. 1904, 2019.

[7] ZANERO, S. "Cyber-physical systems", *Computer*, v. 50, n. 4, pp. 14–16, 2017.

[8] ALGULIYEV, R., IMAMVERDIYEV, Y., SUKHOSTAT, L. "Cyber-physical systems and their security issues", *Computers in Industry*, v. 100, pp. 212–223, 2018.

[9] BAHETI, R., GILL, H. "Cyber-physical systems", *The impact of control technology*, v. 12, n. 1, pp. 161–166, 2011.

[10] SHI, J., WAN, J., YAN, H., et al. "A survey of cyber-physical systems". In: *2011 international conference on wireless communications and signal processing (WCSP)*, pp. 1–6. IEEE, 2011.

[11] DERLER, P., LEE, E. A., VINCENTELLI, A. S. "Modeling cyber–physical systems", *Proceedings of the IEEE*, v. 100, n. 1, pp. 13–28, 2011.

[12] HAQUE, S. A., AZIZ, S. M., RAHMAN, M. "Review of cyber-physical system in healthcare", *international journal of distributed sensor networks*, v. 10, n. 4, pp. 217415, 2014.

[13] MANN, S. "Wearable computing: A first step toward personal imaging", *Computer*, v. 30, n. 2, pp. 25–32, 1997.

[14] STARNER, T. "Human-powered wearable computing", *IBM systems Journal*, v. 35, n. 3.4, pp. 618–629, 1996.

[15] ROGGEN, D., MAGNENAT, S., WAIBEL, M., et al. "Wearable computing", *IEEE Robotics & Automation Magazine*, v. 18, n. 2, pp. 83–95, 2011.

[16] JHAJHARIA, S., PAL, S., VERMA, S. "Wearable computing and its application", *International Journal of Computer Science and Information Technologies*, v. 5, n. 4, pp. 5700–5704, 2014.

[17] SOH, P. J., VANDENBOSCH, G. A., MERCURI, M., et al. "Wearable wireless health monitoring: Current developments, challenges, and future trends", *IEEE Microwave Magazine*, v. 16, n. 4, pp. 55–70, 2015.

[18] RISLING, T. "Educating the nurses of 2025: Technology trends of the next decade", *Nurse education in practice*, v. 22, pp. 89–92, 2017.

[19] VOGENBERG, F. R., SANTILLI, J. "Healthcare trends for 2018", *American Health & Drug Benefits*, v. 11, n. 1, pp. 48, 2018.

[20] LIU, L., PENG, Y., LIU, M., et al. "Sensor-based human activity recognition system with a multilayered model using time series shapelets", *Knowledge-Based Systems*, v. 90, pp. 138–152, 2015.

[21] ZHANG, Y., GRAVINA, R., LU, H., et al. "PEA: Parallel electrocardiogram-based authentication for smart healthcare systems", *Journal of Network and Computer Applications*, v. 117, pp. 10–16, 2018.

[22] QIU, S., WANG, Z., ZHAO, H., et al. "Body sensor network based robust gait analysis: Toward clinical and at home use", *IEEE Sensors Journal*, 2018.

[23] PACE, P., ALOI, G., GRAVINA, R., et al. "An edge-based architecture to support efficient applications for healthcare industry 4.0", *IEEE Transactions on Industrial Informatics*, v. 15, n. 1, pp. 481–489, 2018.

[24] KAYA, T., LIU, G., HO, J., et al. "Wearable Sweat Sensors: Background and Current Trends", *Electroanalysis*, v. 31, n. 3, pp. 411–421, 2019.

[25] CAMOMILLA, V., BERGAMINI, E., FANTOZZI, S., et al. "Trends supporting the in-field use of wearable inertial sensors for sport performance evaluation: A systematic review", *Sensors*, v. 18, n. 3, pp. 873, 2018.

[26] FU, Q.-K., HWANG, G.-J. "Trends in mobile technology-supported collaborative learning: A systematic review of journal publications from 2007 to 2016", *Computers & Education*, v. 119, pp. 129–143, 2018.

[27] CHANG, C.-Y., LAI, C.-L., HWANG, G.-J. "Trends and research issues of mobile learning studies in nursing education: A review of academic publications from 1971 to 2016", *Computers & Education*, v. 116, pp. 28–48, 2018.

[28] KONG, X. T., LUO, H., HUANG, G. Q., et al. "Industrial wearable system: the human-centric empowering technology in Industry 4.0", *Journal of Intelligent Manufacturing*, pp. 1–17, 2018.

[29] RICE, M., MA, K.-T., TAY, H. H., et al. "Evaluating an augmented remote assistance platform to support industrial applications". In: *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, pp. 592–597. IEEE, 2018.

[30] LI, Y., WANG, T., LI, L., et al. "Hand Gesture Recognition and Real-time Game Control Based on A Wearable Band with 6-axis Sensors". In: *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–6. IEEE, 2018.

[31] HAGHI, M., THUROW, K., STOLL, R. "Wearable devices in medical internet of things: scientific research and commercially available devices", *Healthcare informatics research*, v. 23, n. 1, pp. 4–15, 2017.

[32] BHATT, C., DEY, N., ASHOUR, A. S. "Internet of things and big data technologies for next generation healthcare", 2017.

[33] CONSTANT, N., BORTHAKUR, D., ABTAHI, M., et al. "Fog-assisted wiot: A smart fog gateway for end-to-end analytics in wearable internet of things", *arXiv preprint arXiv:1701.08680*, 2017.

[34] KONG, X. T., YANG, X., HUANG, G. Q., et al. "The impact of industrial wearable system on industry 4.0". In: *2018 IEEE 15th International*

Conference on Networking, Sensing and Control (ICNSC), pp. 1–6. IEEE, 2018.

[35] KOS, A., MILUTINOVIĆ, V., UMEK, A. "Challenges in wireless communication for connected sensors and wearable devices used in sport biofeedback applications", *Future generation computer systems*, v. 92, pp. 582–592, 2019.

[36] LI, J., PENG, Z., GAO, S., et al. "Smartphone-assisted energy efficient data communication for wearable devices", *Computer Communications*, v. 105, pp. 33–43, 2017.

[37] VARGHESE, B., WANG, N., BARBHUIYA, S., et al. "Challenges and opportunities in edge computing". In: *2016 IEEE international conference on smart cloud (SmartCloud)*, pp. 20–26. IEEE, 2016.

[38] CAO, K., LIU, Y., MENG, G., et al. "An overview on edge computing research", *IEEE access*, v. 8, pp. 85714–85728, 2020.

[39] EL-SHORBAGY, A.-M. "5G Technology and the Future of Architecture", *Procedia Computer Science*, v. 182, pp. 121–131, 2021.

[40] JIANG, Y., LI, X., LUO, H., et al. "Quo vadis artificial intelligence?" *Discover Artificial Intelligence*, v. 2, n. 1, pp. 4, 2022.

[41] MCCARTHY, J., OTHERS. "What is artificial intelligence", 2007.

[42] EL NAQA, I., MURPHY, M. J. "What is machine learning?" In: *machine learning in radiation oncology*, Springer, pp. 3–11, 2015.

[43] LECUN, Y., BENGIO, Y., HINTON, G. "Deep learning", *nature*, v. 521, n. 7553, pp. 436–444, 2015.

[44] GUNNING, D., STEFIK, M., CHOI, J., et al. "XAI—Explainable artificial intelligence", *Science Robotics*, v. 4, n. 37, pp. eaay7120, 2019.

[45] LIU, Y., LIU, S., WANG, Y., et al. "A survey of stochastic computing neural networks for machine learning applications", *IEEE Transactions on Neural Networks and Learning Systems*, v. 32, n. 7, pp. 2809–2824, 2020.

[46] SZE, V., CHEN, Y.-H., EMER, J., et al. "Hardware for machine learning: Challenges and opportunities". In: *2017 IEEE Custom Integrated Circuits Conference (CICC)*, pp. 1–8. IEEE, 2017.

[47] SILVA, M., FELISBERTO, B., BATISTA, M., et al. "An Automatic Ant Counting and Distribution Estimation System Using Convolutional Neural Networks". In: *Proceedings of the 25th International Conference on Enterprise Information Systems*. SCITEPRESS - Science and Technology Publications, 2023. doi: 10.5220/0011968900003467. Disponível em: <https://doi.org/10.5220/0011968900003467>.

[48] SILVA, M. C., BIANCHI, A. G. C., OLIVEIRA, R. A. R., et al. "Designing a Multiple-User Wearable Edge AI system towards Human Activity Recognition". In: *2022 XII Brazilian Symposium on Computing Systems Engineering (SBESC)*, pp. 1–8. IEEE, 2022.

[49] SILVA, M. C., BIANCHI, A. G. C., RIBEIRO, S. P., et al. "Edge Computing Smart Healthcare Cooperative Architecture for COVID-19 Medical Facilities", *IEEE Latin America Transactions*, v. 20, n. 10, pp. 2229–2236, 2022.

[50] SILVA, M. C., BIANCHI, A. G., RIBEIRO, S. P., et al. "Bringing Deep Learning to the Fields and Forests: Leaf Reconstruction and Shape Estimation", *SN Computer Science*, v. 3, n. 3, pp. 1–14, 2022.

[51] SILVA, M. C., DA SILVA, J. C., DELABRIDA, S., et al. "Wearable Edge AI Applications for Ecological Environments", *Sensors*, v. 21, n. 15, pp. 5082, 2021.

[52] SILVA., M., OLIVEIRA., R., D'ANGELO., T., et al. "Faceshield HUD: Extended Usage of Wearable Computing on the COVID-19 Frontline". In: *Proceedings of the 23rd International Conference on Enterprise Information Systems - Volume 1: ICEIS,*, pp. 893–900. INSTICC, SciTePress, 2021. ISBN: 978-989-758-509-8. doi: 10.5220/0010444308930900.

[53] SILVA, M., FERREIRA DA SILVA, J., OLIVEIRA, R. "IDiSSC: Edge-computing-based Intelligent Diagnosis Support System for Citrus Inspection". In: *Proceedings of the 23rd International Conference on Enterprise Information Systems - Volume 1: ICEIS,*, pp. 685–692. INSTICC, SciTePress, 2021. ISBN: 978-989-758-509-8. doi: 10.5220/0010444106850692.

[54] SILVA, M. C., BIANCHI, A. G., RIBEIRO, S. P., et al. "Leaf shape reconstruction and damage estimation using a U-net-based conditional GAN". In: *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, pp. 1106–1109, 2021.

[55] SILVA, M., MARTINS DE SOUSA, F., BARBOSA, D., et al. "Constraints and Challenges in Designing Applications for Industry 4.0: A Functional Approach". In: *Proceedings of the 22nd International Conference on Enterprise Information Systems - Volume 1: ICEIS,*, pp. 767–774. INSTICC, SciTePress, 2020. ISBN: 978-989-758-423-7. doi: 10.5220/0009570307670774.

[56] ALVIM, P. B., DA SILVA, J. C., AMORIM, V. J., et al. "Towards a mobile system with a new wearable device and an AI application for walking and running activities". In: *Anais do L Seminário Integrado de Software e Hardware*, pp. 155–166. SBC, 2023.

[57] SANTOS, R. C. C. D. M., SILVA, M. C., SANTOS, R. L., et al. "A Mobile Robot Based on Edge AI". In: *Anais do L Seminário Integrado de Software e Hardware*, pp. 191–202. SBC, 2023.

[58] SANTOS, R., SILVA, M., SANTOS, R., et al. "Towards Autonomous Mobile Inspection Robots Using Edge AI". In: *Proceedings of the 25th International Conference on Enterprise Information Systems*. SCITEPRESS - Science and Technology Publications, 2023. doi: 10.5220/0011972200003467. Disponível em: <https://doi.org/10.5220/0011972200003467>.

[59] CARDOSO, F., SILVA, M., MEIRA, N., et al. "Towards a Novel Edge AI System for Particle Size Detection in Mineral Processing Plants". In: *Proceedings of the 25th International Conference on Enterprise Information Systems*. SCITEPRESS - Science and Technology Publications, 2023. doi: 10.5220/0011748000003467. Disponível em: <https://doi.org/10.5220/0011748000003467>.

[60] GARROCHO, C. T., DE SOUSA, F. L., SILVA, M. C., et al. "Blockchain-Based Smart Contract and Edge AI Applied in a Multirobot System: An Approach", *IEEE Robotics & Automation Magazine*, 2023.

[61] DA SILVA, J. C., SILVA, M. C., LUZ, E. J., et al. "Using Mobile Edge AI to Detect and Map Diseases in Citrus Orchards", *Sensors*, v. 23, n. 4, pp. 2165, 2023.

[62] DA SILVA, J. C. F., DE AMORIM, V. J. P., DE OLIVEIRA LAZARONI, P. S., et al. "Towards novel smart wearable sensors to classify subject-specific human walking activities". In: *Anais Estendidos do XII Simpósio Brasileiro de Engenharia de Sistemas Computacionais*, pp. 68–73. SBC, 2022.

[63] DA SILVA, J. C., SILVA, M. C., DELABRIDA, S., et al. "A novel intelligent mobile application using human-centered AR: A case study in orange inspection". In: *Anais Estendidos do XXI Simpósio Brasileiro de Fatores Humanos em Sistemas Computacionais*, pp. 72–75. SBC, 2022.

[64] VITOR, R. F., KELLER, B. N., BARBOSA, D. L., et al. "Enabling Digital Twins in Industry 4.0". In: *International Conference on Enterprise Information Systems*, pp. 465–488. Springer, 2022.

[65] DE C MEIRA, N. F., SILVA, M. C., VIEIRA, C. B., et al. "Edge Deep Learning Towards the Metallurgical Industry: Improving the Hybrid Pelletized Sinter (HPS) Process". In: *International Conference on Enterprise Information Systems*, pp. 149–167. Springer, 2022.

[66] DA SILVA, J. C. F., SILVA, M. C., OLIVEIRA, R. A. "Towards a novel wearable solution for citrus inspection using Edge AI". In: *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*, pp. 966–971. IEEE, 2022.

[67] DE SOUSA, F. L. M., SILVA, M. C., OLIVEIRA, R. A. R. "Applying Edge AI towards Deep-learning-based Monocular Visual Odometry Model for Mobile Robotics." In: *ICEIS (1)*, pp. 561–568, 2022.

[68] MEIRA, N., SILVA, M. C., BIANCHI, A. G., et al. "Edge Deep Learning Applied to Granulometric Analysis on Quasi-particles from the Hybrid Pelletized Sinter (HPS) Process". In: *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2022) - Volume 4: VISAPP*, pp. 462–469. INSTICC, SciTePress, 2022. ISBN: 978-989-758-555-5. doi: 10.5220/0010836900003124.

[69] KLIPPEL, E., BIANCHI, A. G. C., DELABRIDA, S., et al. "Deep Learning Approach at the Edge to Detect Iron Ore Type", *Sensors*, v. 22, n. 1, pp. 169, 2022.

[70] DE SOUSA, F. L. M., DA SILVA, M. J., DE MEIRA SANTOS, R. C. C., et al. "Deep-Learning-Based Embedded ADAS System". In: *2021 XI Brazilian Symposium on Computing Systems Engineering (SBESC)*, pp. 1–8. IEEE, 2021.

[71] DE SOUSA, F. L. M., MEIRA, N. F. D. C., OLIVEIRA, R. A. R., et al. "Deep-Learning-Based Visual Odometry Models for Mobile Robotics". In:

Anais Estendidos do XI Simpósio Brasileiro de Engenharia de Sistemas Computacionais, pp. 122–127. SBC, 2021.

[72] VITOR, R., KELLER, B., BARBOSA, D., et al. "Synchronous and Asynchronous Requirements for Digital Twins Applications in Industry 4.0". In: *Proceedings of the 23rd International Conference on Enterprise Information Systems - Volume 2: ICEIS,*, pp. 637–647. INSTICC, SciTePress, 2021. ISBN: 978-989-758-509-8. doi: 10.5220/0010444406370647.

[73] MEIRA, N., SILVA, M., OLIVEIRA, R., et al. "Edge Deep Learning Applied to Granulometric Analysis on Quasi-particles from the Hybrid Pelletized Sinter (HPS) Process". In: *Proceedings of the 23rd International Conference on Enterprise Information Systems - Volume 1: ICEIS,*, pp. 527–535. INSTICC, SciTePress, 2021. ISBN: 978-989-758-509-8. doi: 10.5220/0010458805270535.

[74] SATYANARAYANAN, M. "The emergence of edge computing", *Computer*, v. 50, n. 1, pp. 30–39, 2017.

[75] DENG, S., ZHAO, H., FANG, W., et al. "Edge intelligence: The confluence of edge computing and artificial intelligence", *IEEE Internet of Things Journal*, v. 7, n. 8, pp. 7457–7469, 2020.

[76] ZHOU, Z., CHEN, X., LI, E., et al. "Edge intelligence: Paving the last mile of artificial intelligence with edge computing", *Proceedings of the IEEE*, v. 107, n. 8, pp. 1738–1762, 2019.

[77] LIU, Y., PENG, M., SHOU, G., et al. "Toward edge intelligence: Multiaccess edge computing for 5G and Internet of Things", *IEEE Internet of Things Journal*, v. 7, n. 8, pp. 6722–6747, 2020.

[78] WANG, X., HAN, Y., WANG, C., et al. "In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning", *IEEE Network*, v. 33, n. 5, pp. 156–165, 2019.

[79] LI, E., ZENG, L., ZHOU, Z., et al. "Edge AI: On-demand accelerating deep neural network inference via edge computing", *IEEE Transactions on Wireless Communications*, v. 19, n. 1, pp. 447–457, 2019.

[80] LEE, Y.-L., TSUNG, P.-K., WU, M. "Techology trend of edge AI". In: *2018 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, pp. 1–2. IEEE, 2018.

[81] GREENGARD, S. "Ai on edge", *Communications of the ACM*, v. 63, n. 9, pp. 18–20, 2020.

[82] HU, L., MIAO, Y., WU, G., et al. "iRobot-Factory: An intelligent robot factory based on cognitive manufacturing and edge computing", *Future Generation Computer Systems*, v. 90, pp. 569–577, 2019.

[83] YANG, J., WANG, R., GUAN, X., et al. "AI-enabled emotion-aware robot: The fusion of smart clothing, edge clouds and robotics", *Future Generation Computer Systems*, v. 102, pp. 701–709, 2020.

[84] RATHI, V. K., RAJPUT, N. K., MISHRA, S., et al. "An edge AI-enabled IoT healthcare monitoring system for smart cities", *Computers & Electrical Engineering*, v. 96, pp. 107524, 2021.

[85] WU, G., MIAO, Y., ZHANG, Y., et al. "Energy efficient for UAV-enabled mobile edge computing networks: Intelligent task prediction and offloading", *Computer Communications*, v. 150, pp. 556–562, 2020.

[86] ZHANG, Y., YU, J., CHEN, Y., et al. "Real-time strawberry detection using deep neural networks on embedded system (rtsd-net): An edge AI application", *Computers and Electronics in Agriculture*, v. 192, pp. 106586, 2022.

[87] DEBAUCHE, O., MAHMOUDI, S., MAHMOUDI, S. A., et al. "Edge computing and artificial intelligence for real-time poultry monitoring", *Procedia computer science*, v. 175, pp. 534–541, 2020.

[88] DEBAUCHE, O., MAHMOUDI, S., ELMOULAT, M., et al. "Edge AI-IoT pivot irrigation, plant diseases, and pests identification", *Procedia Computer Science*, v. 177, pp. 40–48, 2020.

[89] ELMOULAT, M., DEBAUCHE, O., MAHMOUDI, S., et al. "Edge computing and artificial intelligence for landslides monitoring", *Procedia Computer Science*, v. 177, pp. 480–487, 2020.

[90] GIA, T. N., QINGQING, L., QUERALTA, J. P., et al. "Edge AI in smart farming IoT: CNNs at the edge and fog computing with LoRa". In: *2019 IEEE AFRICON*, pp. 1–6. IEEE, 2019.

[91] MUHAMMAD, K., KHAN, S., PALADE, V., et al. "Edge intelligence-assisted smoke detection in foggy surveillance environments", *IEEE Transactions on Industrial Informatics*, v. 16, n. 2, pp. 1067–1075, 2019.

[92] KLIPPEL, E., OLIVEIRA, R., MASLOV, D., et al. "Embedded Edge Artificial Intelligence for Longitudinal Rip Detection in Conveyor Belt Applied at the Industrial Mining Environment". In: *Proceedings of the 23rd International Conference on Enterprise Information Systems*. SCITEPRESS - Science and Technology Publications, 2021. doi: 10.5220/0010447204960505. Disponível em: <https://doi.org/10.5220/0010447204960505>.

[93] GOMATHY, V., JANARTHANAN, K., AL-TURJMAN, F., et al. "Investigating the spread of coronavirus disease via edge-AI and air pollution correlation", *ACM Transactions on Internet Technology*, v. 21, n. 4, pp. 1–10, 2021.

[94] RAHMAN, M. S., KHALIL, I., YI, X., et al. "A Lossless Data-Hiding based IoT Data Authenticity Model in Edge-AI for Connected Living", *ACM Transactions on Internet Technology (TOIT)*, v. 22, n. 3, pp. 1–25, 2021.

[95] CHAVHAN, S., GUPTA, D., GOCHHAYAT, S. P., et al. "Edge Computing AI-IoT Integrated Energy Efficient Intelligent Transportation System for Smart Cities", *ACM Transactions on Internet Technology (TOIT)*, 2022.

[96] YANG, H., HAN, X. "National Sports AI Health Management Service System Based on Edge Computing", *Wireless Communications and Mobile Computing*, v. 2021, 2021.

[97] DINH, D.-L., NGUYEN, H.-N., THAI, H.-T., et al. "Towards AI-Based Traffic Counting System with Edge Computing", *Journal of Advanced Transportation*, v. 2021, 2021.

[98] BIBI, R., SAEED, Y., ZEB, A., et al. "Edge AI-based automated detection and classification of road anomalies in VANET using deep learning", *Computational intelligence and neuroscience*, v. 2021, 2021.

[99] MUNIR, M. S., BAJWA, I. S., ASHRAF, A., et al. "Intelligent and smart irrigation system using edge computing and IoT", *Complexity*, v. 2021, 2021.

[100] BATTISTONI, P., DI GREGORIO, M., SEBILLO, M., et al. "AI at the edge for sign language learning support". In: *2019 IEEE International Conference on Humanized Computing and Communication (HCC)*, pp. 16–23. IEEE, 2019.

[101] HOU, D., LIU, T., PAN, Y.-T., et al. "AI on edge device for laser chip defect detection". In: *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 0247–0251. IEEE, 2019.

[102] YANG, C.-J., FAHIER, N., HE, C.-Y., et al. "An ai-edge platform with multimodal wearable physiological signals monitoring sensors for affective computing applications". In: *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5. IEEE, 2020.

[103] LIU, C., CAO, Y., LUO, Y., et al. "A new deep learning-based food recognition system for dietary assessment on an edge computing service infrastructure", *IEEE Transactions on Services Computing*, v. 11, n. 2, pp. 249–261, 2017.

[104] LIN, Y.-C., CHEN, W.-H., KUO, C.-H. "Implementation of Pavement Defect Detection System on Edge Computing Platform", *Applied Sciences*, v. 11, n. 8, pp. 3725, 2021.

[105] LIN, C.-J., CHUANG, C.-C., LIN, H.-Y. "Edge-AI-Based Real-Time Automated License Plate Recognition System", *Applied Sciences*, v. 12, n. 3, pp. 1445, 2022.

[106] PAN, J., LUO, Y., LI, Y., et al. "A Wireless multi-channel capacitive sensor system for efficient glove-based gesture recognition with AI at the edge", *IEEE Transactions on Circuits and Systems II: Express Briefs*, v. 67, n. 9, pp. 1624–1628, 2020.

[107] LI, J., WU, J., LI, J., et al. "Blockchain-based trust edge knowledge inference of multi-robot systems for collaborative tasks", *IEEE Communications Magazine*, v. 59, n. 7, pp. 94–100, 2021.

[108] SAKIB, S., FOUDA, M. M., AL-MAHDAWI, M., et al. "Deep Learning Models for Magnetic Cardiography Edge Sensors Implementing Noise Processing and Diagnostics", *IEEE Access*, v. 10, pp. 2656–2668, 2021.

[109] LI, M., GAO, J., ZHAO, L., et al. "Deep reinforcement learning for collaborative edge computing in vehicular networks", *IEEE Transactions on Cognitive Communications and Networking*, v. 6, n. 4, pp. 1122–1135, 2020.

[110] FANG, W.-C., WANG, K.-Y., FAHIER, N., et al. "Development and validation of an EEG-based real-time emotion recognition system using edge AI computing platform with convolutional neural network system-on-chip

design", *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, v. 9, n. 4, pp. 645–657, 2019.

[111] CHEN, J., LI, K., DENG, Q., et al. "Distributed deep learning model for intelligent video surveillance systems with edge computing", *IEEE Transactions on Industrial Informatics*, 2019.

[112] QUERALTA, J. P., GIA, T. N., TENHUNEN, H., et al. "Edge-AI in LoRa-based health monitoring: Fall detection system with fog computing and LSTM recurrent neural networks". In: *2019 42nd international conference on telecommunications and signal processing (TSP)*, pp. 601–604. IEEE, 2019.

[113] FOUKALAS, F., TZIOUVARAS, A. "Edge artificial intelligence for industrial internet of things applications: an industrial edge intelligence solution", *IEEE Industrial Electronics Magazine*, v. 15, n. 2, pp. 28–36, 2021.

[114] YANG, T., LEE, S.-H., PARK, S. "AI-Aided Individual Emergency Detection System in Edge-Internet of Things Environments", *Electronics*, v. 10, n. 19, pp. 2374, 2021.

[115] GUPTA, N., KHOSRAVY, M., PATEL, N., et al. "Economic data analytic AI technique on IoT edge devices for health monitoring of agriculture machines", *Applied Intelligence*, v. 50, n. 11, pp. 3990–4016, 2020.

[116] LIU, Y., YANG, C., JIANG, L., et al. "Intelligent edge computing for IoT-based energy management in smart cities", *IEEE network*, v. 33, n. 2, pp. 111–117, 2019.

[117] BARNAWI, A., ALHARBI, M., CHEN, M. "Intelligent search and find system for robotic platform based on smart edge computing service", *IEEE Access*, v. 8, pp. 108821–108834, 2020.

[118] LIAO, C., SHOU, G., LIU, Y., et al. "Intelligent traffic accident detection system based on mobile edge computing". In: *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*, pp. 2110–2115. IEEE, 2017.

[119] DE VITA, A., PAU, D., PARRELLA, C., et al. "Low-power HWAccelerator for AI edge-computing in human activity recognition systems". In: *2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pp. 291–295. IEEE, 2020.

[120] LIBRI, A., BARTOLINI, A., BENINI, L. "pAElla: Edge AI-based real-time malware detection in data centers", *IEEE Internet of Things Journal*, v. 7, n. 10, pp. 9589–9599, 2020.

[121] ESKANDARI, M., JANJUA, Z. H., VECCHIO, M., et al. "Passban IDS: An intelligent anomaly-based intrusion detection system for IoT edge devices", *IEEE Internet of Things Journal*, v. 7, n. 8, pp. 6882–6897, 2020.

[122] MAZZIA, V., KHALIQ, A., SALVETTI, F., et al. "Real-time apple detection system using embedded systems with hardware accelerators: An edge AI application", *IEEE Access*, v. 8, pp. 9102–9114, 2020.

[123] COPPOLA, M., NOAILLE, L., PIERLOT, C., et al. "Innovative Vineyards Environmental Monitoring System Using Deep Edge AI". 2021.

[124] POON, C. C., JIANG, Y., ZHANG, R., et al. "AI-doscopist: a real-time deep-learning-based algorithm for localising polyps in colonoscopy videos with edge computing devices", *NPJ digital medicine*, v. 3, n. 1, pp. 1–8, 2020.

[125] MANOGARAN, G., SHAKEEL, P. M., FOUAD, H., et al. "Wearable IoT smart-log patch: An edge computing-based Bayesian deep learning network system for multi access physical monitoring system", *Sensors*, v. 19, n. 13, pp. 3030, 2019.

[126] COB-PARRO, A. C., LOSADA-GUTIÉRREZ, C., MARRÓN-ROMERA, M., et al. "Smart video surveillance system based on edge computing", *Sensors*, v. 21, n. 9, pp. 2958, 2021.

[127] KLIPPEL, E., OLIVEIRA, R., MASLOV, D., et al. "Embedded Edge Artificial Intelligence for Longitudinal Rip Detection in Conveyor Belt Applied at the Industrial Mining Environment". In: *Proceedings of the 23rd International Conference on Enterprise Information Systems*. SCITEPRESS - Science and Technology Publications, 2021. doi: 10.5220/0010447204960505. Disponível em: <https://doi.org/10.5220/0010447204960505>.

[128] LYU, S., LI, R., ZHAO, Y., et al. "Green Citrus Detection and Counting in Orchards Based on YOLOv5-CS and AI Edge System", *Sensors*, v. 22, n. 2, pp. 576, 2022.

[129] SU, X., SPERLÌ, G., MOSCATO, V., et al. "An edge intelligence empowered recommender system enabling cultural heritage applications", *IEEE Transactions on Industrial Informatics*, v. 15, n. 7, pp. 4266–4275, 2019.

[130] XU, S., QIAN, Y., HU, R. Q. "A semi-supervised learning approach for network anomaly detection in fog computing". In: *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pp. 1–6. IEEE, 2019.

[131] BECK, M. T., WERNER, M., FELD, S., et al. "Mobile edge computing: A taxonomy". In: *Proc. of the Sixth International Conference on Advances in Future Internet*, pp. 48–55. Citeseer, 2014.

[132] AHMED, E., AHMED, A., YAQOOB, I., et al. "Bringing computation closer toward the user network: Is edge computing the solution?" *IEEE Communications Magazine*, v. 55, n. 11, pp. 138–144, 2017.

[133] DOLUI, K., DATTA, S. K. "Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing". In: *2017 Global Internet of Things Summit (GIoTS)*, pp. 1–6. IEEE, 2017.

[134] SHETH, K., PATEL, K., SHAH, H., et al. "A taxonomy of AI techniques for 6G communication networks", *Computer communications*, v. 161, pp. 279–303, 2020.

[135] BALTRUŠAITIS, T., AHUJA, C., MORENCY, L.-P. "Multimodal machine learning: A survey and taxonomy", *IEEE transactions on pattern analysis and machine intelligence*, v. 41, n. 2, pp. 423–443, 2018.

[136] TALBI, E.-G. "Machine learning into metaheuristics: A survey and taxonomy", *ACM Computing Surveys (CSUR)*, v. 54, n. 6, pp. 1–32, 2021.

[137] MENDEZ, J., BIERZYNSKI, K., CUÉLLAR, M., et al. "Edge Intelligence: Concepts, architectures, applications and future directions", *ACM Transactions on Embedded Computing Systems (TECS)*, 2022.

[138] AMIN, S. U., HOSSAIN, M. S. "Edge intelligence and internet of things in healthcare: a survey", *IEEE Access*, v. 9, pp. 45–59, 2020.

[139] LALAPURA, V. S., AMUDHA, J., SATHEESH, H. S. "Recurrent neural networks for edge intelligence: a survey", *ACM Computing Surveys (CSUR)*, v. 54, n. 4, pp. 1–38, 2021.

[140] FANTACCI, R., PICANO, B. "Federated learning framework for mobile edge computing networks", *CAAI Transactions on Intelligence Technology*, v. 5, n. 1, pp. 15–21, 2020.

[141] NGUYEN, D. C., DING, M., PHAM, Q.-V., et al. "Federated learning meets blockchain in edge computing: Opportunities and challenges", *IEEE Internet of Things Journal*, 2021.

[142] AL-ANSI, A., AL-ANSI, A. M., MUTHANNA, A., et al. "Survey on intelligence edge computing in 6G: characteristics, challenges, potential use cases, and market drivers", *Future Internet*, v. 13, n. 5, pp. 118, 2021.

[143] AUGIMERI, A., FORTINO, G., GALZARANO, S., et al. "Collaborative body sensor networks". In: *2011 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 3427–3432. IEEE, 2011.

[144] FORTINO, G., GALZARANO, S., GRAVINA, R., et al. "A framework for collaborative computing and multi-sensor data fusion in body sensor networks", *Information Fusion*, v. 22, pp. 50–70, 2015.

[145] MIHOVSKA, A., SARKAR, M. "Cooperative Human-Centric Sensing Connectivity". In: *Internet of Things-Technology, Applications and Standardization*, InTechOpen, 2018.

[146] ZHANG, X., YANG, Z., CHEN, T., et al. "Cooperative Sensing and Wearable Computing for Sequential Hand Gesture Recognition", *IEEE Sensors Journal*, 2019.

[147] PENG, Y., PENG, L. "A cooperative transmission strategy for body-area networks in healthcare systems", *IEEE Access*, v. 4, pp. 9155–9162, 2016.

[148] NGUYEN-HUU, K., SONG, C. G., SEON-WOO, L. "Smartwatch/Smartphone Cooperative Indoor Lifelogging System", *International Journal of Engineering and Technology Innovation*, v. 8, n. 4, pp. 261, 2018.

[149] PIMENTEL, G., RODRIGUES, S., SILVA, P. A., et al. "A Wearable Approach for Intraoperative Physiological Stress Monitoring of Multiple Cooperative Surgeons", *International Journal of Medical Informatics*, 2019.

[150] PRAKASH, R., GANESH, A. B. "Establishment of network coded cooperative communication for clinical healthcare". In: *2016 2nd International Conference on Contemporary Computing and Informatics (IC3I)*, pp. 126–130. IEEE, 2016.

[151] PHAM, M. H., WARMERDAM, E., ELSHEHABI, M., et al. "Validation of a lower back "wearable"-based sit-to-stand and stand-to-sit algorithm for patients with Parkinson's disease and older adults in a home-like environment", *Frontiers in neurology*, v. 9, pp. 652, 2018.

[152] CHEN, J., RAN, X. "Deep Learning With Edge Computing: A Review." *Proceedings of the IEEE*, v. 107, n. 8, pp. 1655–1674, 2019.

[153] WANG, X., HAN, Y., LEUNG, V. C., et al. "Convergence of edge computing and deep learning: A comprehensive survey", *IEEE Communications Surveys & Tutorials*, v. 22, n. 2, pp. 869–904, 2020.

[154] UDDIN, M. Z. "A wearable sensor-based activity prediction system to facilitate edge computing in smart healthcare system", *Journal of Parallel and Distributed Computing*, v. 123, pp. 46–53, 2019.

[155] LIU, H., YAO, X., YANG, T., et al. "Cooperative privacy preservation for wearable devices in hybrid computing-based smart health", *IEEE Internet of Things Journal*, v. 6, n. 2, pp. 1352–1362, 2018.

[156] VEGA-BARBAS, M., DIAZ-OLIVARES, J. A., LU, K., et al. "P-Ergonomics Platform: Toward precise, pervasive, and personalized ergonomics using wearable sensors and edge computing", *Sensors*, v. 19, n. 5, pp. 1225, 2019.

[157] KUMARI, P., LÓPEZ-BENÍTEZ, M., LEE, G. M., et al. "Wearable Internet of Things-from human activity tracking to clinical integration". In: *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 2361–2364. IEEE, 2017.

[158] DEMICHELI, G., SAMI, M. *Hardware/software Co-design*, v. 310. Springer Science & Business Media, 2013.

[159] DE MICHELL, G., GUPTA, R. K. "Hardware/software co-design", *Proceedings of the IEEE*, v. 85, n. 3, pp. 349–365, 1997.

[160] MUIRURI, E. W., BARANTAL, S., IASON, G. R., et al. "Forest diversity effects on insect herbivores: do leaf traits matter?" *New Phytologist*, v. 221, n. 4, pp. 2250–2260, 2019.

[161] BENÍTEZ-MALVIDO, J., LÁZARO, A., FERRAZ, I. D. "Effect of distance to edge and edge interaction on seedling regeneration and biotic damage in tropical rainforest fragments: A long-term experiment", *Journal of Ecology*, v. 106, n. 6, pp. 2204–2217, 2018.

[162] SAIDOV, N., SRINIVASAN, R., MAVLYANOVA, R., et al. "First report of invasive South American tomato leaf miner Tuta absoluta (Meyrick)(Lepidoptera: Gelechiidae) in Tajikistan", *Florida Entomologist*, v. 101, n. 1, pp. 147–150, 2018.

[163] BAUDRON, F., ZAMAN-ALLAH, M. A., CHAIPA, I., et al. "Understanding the factors influencing fall armyworm (Spodoptera frugiperda JE Smith)

damage in African smallholder maize fields and quantifying its impact on yield. A case study in Eastern Zimbabwe", *Crop Protection*, v. 120, pp. 141–150, 2019.

[164] WU, S. G., BAO, F. S., XU, E. Y., et al. "A leaf recognition algorithm for plant classification using probabilistic neural network". In: *2007 IEEE international symposium on signal processing and information technology*, pp. 11–16. IEEE, 2007.

[165] NOVOTNỲ, P., SUK, T. "Leaf recognition of woody species in Central Europe", *Biosystems Engineering*, v. 115, n. 4, pp. 444–452, 2013.

[166] OTSU, N. "A threshold selection method from gray-level histograms", *IEEE transactions on systems, man, and cybernetics*, v. 9, n. 1, pp. 62–66, 1979.

[167] DA SILVA, L. A., BRESSAN, P. O., GONÇALVES, D. N., et al. "Estimating soybean leaf defoliation using convolutional neural networks and synthetic images", *Computers and electronics in agriculture*, v. 156, pp. 360–368, 2019.

[168] ISOLA, P., ZHU, J.-Y., ZHOU, T., et al. "Image-to-image translation with conditional adversarial networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1125–1134, 2017.

[169] RONNEBERGER, O., FISCHER, P., BROX, T. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

[170] HOU, X., SHEN, L., SUN, K., et al. "Deep feature consistent variational autoencoder". In: *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1133–1141. IEEE, 2017.

[171] DONG, H., YANG, G., LIU, F., et al. "Automatic brain tumor detection and segmentation using U-Net based fully convolutional networks". In: *annual conference on medical image understanding and analysis*, pp. 506–517. Springer, 2017.

[172] OKTAY, O., SCHLEMPER, J., FOLGOC, L. L., et al. "Attention u-net: Learning where to look for the pancreas", *arXiv preprint arXiv:1804.03999*, 2018.

[173] HYUN, C. M., KIM, H. P., LEE, S. M., et al. "Deep learning for undersampled MRI reconstruction", *Physics in Medicine & Biology*, v. 63, n. 13, pp. 135007, 2018.

[174] ANTHOLZER, S., HALTMEIER, M., NUSTER, R., et al. "Photoacoustic image reconstruction via deep learning". In: *Photons Plus Ultrasound: Imaging and Sensing 2018*, v. 10494, p. 104944U. International Society for Optics and Photonics, 2018.

[175] GENÇTAV, A., AKSOY, S., ÖNDER, S. "Unsupervised segmentation and classification of cervical cell images", *Pattern recognition*, v. 45, n. 12, pp. 4151–4168, 2012.

[176] SAMPAT, M. P., WANG, Z., GUPTA, S., et al. "Complex wavelet structural similarity: A new image similarity index", *IEEE transactions on image processing*, v. 18, n. 11, pp. 2385–2401, 2009.

[177] SHAMIR, R. R., DUCHIN, Y., KIM, J., et al. "Continuous dice coefficient: a method for evaluating probabilistic segmentations", *arXiv preprint arXiv:1906.11031*, 2019.

[178] MUN, J., JANG, W.-D., SUNG, D. J., et al. "Comparison of objective functions in CNN-based prostate magnetic resonance image segmentation". In: *2017 IEEE International Conference on Image Processing (ICIP)*, pp. 3859–3863. IEEE, 2017.

[179] NITSCH, J., KLEIN, J., DAMMANN, P., et al. "Automatic and efficient MRI-US segmentations for improving intraoperative image fusion in image-guided neurosurgery", *NeuroImage: Clinical*, v. 22, pp. 101766, 2019.

[180] SALKIND, N. J. *Encyclopedia of research design*, v. 1. sage, 2010.

[181] GARRIDO-JURADO, S., MUÑOZ-SALINAS, R., MADRID-CUEVAS, F. J., et al. "Automatic generation and detection of highly reliable fiducial markers under occlusion", *Pattern Recognition*, v. 47, n. 6, pp. 2280–2292, 2014.

[182] RIBEIRO, S. P., BASSET, Y., KITCHING, R. "Density of insect galls in the forest understorey and canopy: Neotropical, Gondwana or global patterns?" In: *Neotropical insect galls*, Springer, pp. 129–141, 2014.

[183] GARCÍA-GUZMÁN, G., DIRZO, R. "Incidence of leaf pathogens in the canopy of a Mexican tropical wet forest", *Plant Ecology*, v. 172, n. 1, pp. 41–50, 2004.

[184] SOUBEYRAND, S., ENJALBERT, J., SACHE, I. "Accounting for roughness of circular processes: Using Gaussian random processes to model the anisotropic spread of airborne plant disease", *Theoretical Population Biology*, v. 73, n. 1, pp. 92–103, 2008.

[185] POKHAREL, G., DEARDON, R. "Gaussian process emulators for spatial individual-level models of infectious disease", *Canadian Journal of Statistics*, v. 44, n. 4, pp. 480–501, 2016.

[186] KETU, S., MISHRA, P. K. "Enhanced Gaussian process regression-based forecasting model for COVID-19 outbreak and significance of IoT for its detection", *Applied Intelligence*, v. 51, n. 3, pp. 1492–1512, 2021.

[187] BONATO, P. "Wearable sensors/systems and their impact on biomedical engineering", *IEEE Engineering in Medicine and Biology Magazine*, v. 22, n. 3, pp. 18–20, 2003.

[188] SILVA, M., DELABRIDA, S., RIBEIRO, S., et al. "Toward the Design of a Novel Wearable System for Field Research in Ecology". In: *2018 VIII Brazilian Symposium on Computing Systems Engineering (SBESC)*, pp. 160–165. IEEE, 2019.

[189] SILVA, M. C., RIBEIRO, S. P., DELABRIDA, S., et al. "Smart-Helmet development for Ecological Field Research Applications". In: *Anais do XLVI Seminário Integrado de Software e Hardware*, pp. 69–80. SBC, 2019.

[190] CHOUHAN, S. S., KAUL, A., SINGH, U. P. "A Database of Leaf Images: Practice towards Plant Conservation with Plant Pathology". 2020. Disponível em: <`https://data.mendeley.com/datasets/hb74ynkjcn/4`>.

[191] PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., et al. "Scikit-learn: Machine learning in Python", *the Journal of machine Learning research*, v. 12, pp. 2825–2830, 2011.

[192] SILVA, M., OLIVEIRA, R. "Analyzing the Effect of Increased Distribution on a Wearable Appliance". In: *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, v. 2, pp. 13–18. IEEE, 2019.

[193] HELANTERÄ, H., STRASSMANN, J. E., CARRILLO, J., et al. "Unicolonial ants: where do they come from, what are they and where are they going?" *Trends in Ecology & Evolution*, v. 24, n. 6, pp. 341–349, 2009.

[194] MCGLYNN, T. P. "The ecology of nest movement in social insects", *Annual review of entomology*, v. 57, pp. 291–308, 2012.

[195] HAKALA, S. M., PERTTU, S., HELANTERÄ, H. "Evolution of dispersal in ants (Hymenoptera: Formicidae): A review on the dispersal strategies of sessile superorganisms", *Myrmecological News*, v. 29, 2019.

[196] MAJER, J., HETERICK, B. "Planning for long-term invertebrate studies–problems, pitfalls and possibilities", *Australian Zoologist*, v. 39, n. 4, pp. 617–626, 2018.

[197] WAN, J., WANG, Q., CHAN, A. B. "Kernel-based density map generation for dense object counting", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[198] HOWARD, A. G., ZHU, M., CHEN, B., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications", *arXiv preprint arXiv:1704.04861*, 2017.

[199] TAN, M., LE, Q. "Efficientnetv2: Smaller models and faster training". In: *International Conference on Machine Learning*, pp. 10096–10106. PMLR, 2021.

[200] SODHRO, A., SANGAIAH, A., SODHRO, G., et al. "An energy-efficient algorithm for wearable electrocardiogram signal processing in ubiquitous healthcare applications", *Sensors*, v. 18, n. 3, pp. 923, 2018.

[201] FIROUZI, F., RAHMANI, A. M., MANKODIYA, K., et al. "Internet-of-Things and big data for smarter healthcare: from device to architecture, applications and analytics". 2018.

[202] COSTA, L., FARIAS, R., SANTIAGO, A., et al. "Abiotic factors drives floristic variations of fern's metacommunity in an Atlantic Forest remnant", *Brazilian Journal of Biology*, v. 78, n. 4, pp. 736–741, 2018.

[203] ARENA, M. V., MARTINES, M. R., DA SILVA, T. N., et al. "Multiple-scale approach for evaluating the occupation of stingless bees in Atlantic forest patches", *Forest ecology and management*, v. 430, pp. 509–516, 2018.

[204] CALVÃO, L. B., JUEN, L., DE OLIVEIRA JUNIOR, J. M. B., et al. "Land use modifies Odonata diversity in streams of the Brazilian Cerrado", *Journal of insect conservation*, v. 22, n. 5-6, pp. 675–685, 2018.

[205] PEREIRA, A., NUNES, F., AICOS, F. P. "Physical Activity Intensity Monitoring of Hospital Workers using a Wearable Sensor". In: *Proceedings of the 12th EAI International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth'18). EAI. DOI: http://dx. doi. org/10.4108/eai*, pp. 20–4, 2018.

[206] JAHANBANIFAR, S., AKHAVIAN, R. "Evaluation of wearable sensors to quantify construction workers muscle force: an ergonomic analysis". In: *Proceedings of the 2018 Winter Simulation Conference*, pp. 3921–3929. IEEE Press, 2018.

[207] WEEKS, D. L., SPRINT, G. L., STILWILL, V., et al. "Implementing wearable sensors for continuous assessment of daytime heart rate response in inpatient rehabilitation", *Telemedicine and e-Health*, v. 24, n. 12, pp. 1014–1020, 2018.

[208] VELÁZQUEZ, R., PISSALOUX, E., RODRIGO, P., et al. "An outdoor navigation system for blind pedestrians using GPS and tactile-foot feedback", *Applied Sciences*, v. 8, n. 4, pp. 578, 2018.

[209] TJHAI, C., STEWARD, J., LICHTI, D., et al. "Using a mobile range-camera motion capture system to evaluate the performance of integration of multiple low-cost wearable sensors and gait kinematics for pedestrian navigation in realistic environments". In: *2018 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, pp. 294–300. IEEE, 2018.

[210] KISS, F., BOLDT, R., PFLEGING, B., et al. "Navigation systems for motorcyclists: exploring wearable tactile feedback for route guidance in the real world". In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, p. 617. ACM, 2018.

[211] "DHT22.pdf". `https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf`. (Accessed on 08/11/2023).

[212] "MPU-6000-Datasheet1.pdf". `https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf`. (Accessed on 08/11/2023).

[213] "Microsoft Word - BCM2835 ARM Peripherals.docx". `https://datasheets.raspberrypi.com/bcm2835/bcm2835-peripherals.pdf`. (Accessed on 08/11/2023).

[214] "max30100.pdf". https://www.analog.com/media/en/technical-documentation/data-sheets/max30100.pdf. (Accessed on 08/11/2023).

[215] "GlobalTop-FGPMMOPA6H-Datasheet-V0A-Preliminary". https://cdn-shop.adafruit.com/datasheets/GlobalTop-FGPMMOPA6H-Datasheet-V0A.pdf. (Accessed on 08/11/2023).

[216] "Document:". https://cdn-shop.adafruit.com/datasheets/TSL2561.pdf. (Accessed on 08/11/2023).

[217] WATERS, L. E., LANGE, R. A. "An updated calibration of the plagioclase-liquid hygrometer-thermometer applicable to basalts through rhyolites", *American Mineralogist*, v. 100, n. 10, pp. 2172–2184, 2015.

[218] YADAV, N., BLEAKLEY, C. "Fast calibration of a 9-DOF IMU using a 3 DOF position tracker and a semi-random motion sequence", *Measurement*, v. 90, pp. 192–198, 2016.

[219] ÖBERG, T. "Muscle fatigue and calibration of EMG measurements", *Journal of Electromyography and Kinesiology*, v. 5, n. 4, pp. 239–243, 1995.

[220] MENGE, F., SEEBER, G., VOLKSEN, C., et al. "Results of absolute field calibration of GPS antenna PCV". In: *PROCEEDINGS OF ION GPS*, v. 11, pp. 31–38. INSTITUTE OF NAVIGATION, 1998.

[221] SCHRAMA, C., REIJN, H. "Novel calibration method for filter radiometers", *Metrologia*, v. 36, n. 3, pp. 179, 1999.

[222] HAYES, M. J., SMITH, P. R. "A new method for pulse oximetry possessing inherent insensitivity to artifact", *IEEE Transactions on Biomedical Engineering*, v. 48, n. 4, pp. 452–461, 2001.

[223] WU, F., REDOUTÉ, J.-M., YUCE, M. R. "We-safe: A self-powered wearable IoT sensor network for safety applications based on LoRa", *IEEE Access*, v. 6, pp. 40846–40853, 2018.

[224] VARATHARAJAN, R., MANOGARAN, G., PRIYAN, M. K., et al. "Wearable sensor devices for early detection of Alzheimer disease using dynamic time warping algorithm", *Cluster Computing*, v. 21, n. 1, pp. 681–690, 2018.

[225] ROOPAEI, M., RAD, P., PREVOST, J. J. "A Wearable IoT with Complex Artificial Perception Embedding for Alzheimer Patients". In: *2018 World Automation Congress (WAC)*, pp. 1–6. IEEE, 2018.

[226] LI, B., DONG, Q., DOWNEN, R. S., et al. "A wearable IoT aldehyde sensor for pediatric asthma research and management", *Sensors and Actuators B: Chemical*, v. 287, pp. 584–594, 2019.

[227] NOUSIAS, S., TSELIOS, C., BITZAS, D., et al. "Uncertainty management for wearable iot wristband sensors using Laplacian-based matrix completion". In: *2018 IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, pp. 1–6. IEEE, 2018.

[228] GIA, T. N., SARKER, V. K., TCARENKO, I., et al. "Energy efficient wearable sensor node for IoT-based fall detection systems", *Microprocessors and Microsystems*, v. 56, pp. 34–46, 2018.

[229] MAHMOUD, M. S., MOHAMAD, A. A. "A study of efficient power consumption wireless communication techniques/modules for internet of things (IoT) applications", 2016.

[230] BOUKERCHE, A., SAMARAH, S. "A novel algorithm for mining association rules in wireless ad hoc sensor networks", *IEEE Transactions on Parallel and Distributed Systems*, v. 19, n. 7, pp. 865–877, 2008.

[231] CHEN, Z., HU, W., WANG, J., et al. "An empirical study of latency in an emerging class of edge computing applications for wearable cognitive assistance". In: *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, pp. 1–14, 2017.

[232] KIM, J., GUTRUF, P., CHIARELLI, A. M., et al. "Miniaturized battery-free wireless systems for wearable pulse oximetry", *Advanced functional materials*, v. 27, n. 1, pp. 1604373, 2017.

[233] REN, J., GUO, H., XU, C., et al. "Serving at the edge: A scalable IoT architecture based on transparent computing", *IEEE Network*, v. 31, n. 5, pp. 96–105, 2017.

[234] GRUBERT, J., LANGLOTZ, T., ZOLLMANN, S., et al. "Towards pervasive augmented reality: Context-awareness in augmented reality", *IEEE transactions on visualization and computer graphics*, v. 23, n. 6, pp. 1706–1724, 2016.

[235] SURVE, A. R., GHORPADE, V. R. "Pervasive Context-Aware Computing Survey of Context-aware ubiquitous middleware systems", *International Journal of Engineering10. 1*, 2017.

[236] AMFT, O. "How wearable computing is shaping digital health", *IEEE Pervasive Computing*, v. 17, n. 1, pp. 92–98, 2018.

[237] KLIGER, A. S., SILBERZWEIG, J. "Mitigating risk of COVID-19 in dialysis facilities", *Clinical Journal of the American Society of Nephrology*, v. 15, n. 5, pp. 707–709, 2020.

[238] PRACHAND, V. N., MILNER, R., ANGELOS, P., et al. "Medically-necessary, time-sensitive procedures: A scoring system to ethically and efficiently manage resource scarcity and provider risk during the COVID-19 pandemic", *Journal of the American College of Surgeons*, 2020.

[239] ORGANIZATION, W. H., OTHERS. *Advice on the use of masks in the context of COVID-19: interim guidance, 6 April 2020*. Relatório técnico, World Health Organization, 2020.

[240] HONG, S., GU, Y., SEO, J. K., et al. "Wearable thermoelectrics for personalized thermoregulation", *Science advances*, v. 5, n. 5, pp. eaaw0536, 2019.

[241] SÖRÖS, G., SEMMLER, S., HUMAIR, L., et al. "Fast blur removal for wearable QR code scanners". In: *Proceedings of the 2015 ACM International Symposium on Wearable Computers*, pp. 117–124, 2015.

[242] TREMPER, K. K., BARKER, S. J. "Pulse oximetry", *Anesthesiology: The Journal of the American Society of Anesthesiologists*, v. 70, n. 1, pp. 98–108, 1989.

[243] LONG, C., CAO, Y., JIANG, T., et al. "Edge computing framework for cooperative video processing in multimedia IoT systems", *IEEE Transactions on Multimedia*, v. 20, n. 5, pp. 1126–1139, 2017.

[244] CHEN, Z., HE, S., LI, F., et al. "Mobile field hospitals, an effective way of dealing with COVID-19 in China: sharing our experience", *BioScience Trends*, 2020.

[245] CHANG, C., SRIRAMA, S. N., BUYYA, R. "Indie fog: An efficient fog-computing infrastructure for the internet of things", *Computer*, v. 50, n. 9, pp. 92–98, 2017.

[246] MAJUMDER, S., MONDAL, T., DEEN, M. J. "Wearable sensors for remote health monitoring", *Sensors*, v. 17, n. 1, pp. 130, 2017.

[247] BETANCUR, J. A., VILLA-ESPINAL, J., OSORIO-GÓMEZ, G., et al. "Research topics and implementation trends on automotive head-up display systems", *International Journal on Interactive Design and Manufacturing (IJIDeM)*, v. 12, n. 1, pp. 199–214, 2018.

[248] SHAH, S., MAJMUDAR, K., STEIN, A., et al. "Novel use of home pulse oximetry monitoring in COVID-19 patients discharged from the emergency department identifies need for hospitalization", *Academic Emergency Medicine*, v. n/a, n. n/a. doi: 10.1111/acem.14053. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1111/acem.14053>.

[249] CAO, K., XU, G., ZHOU, J., et al. "QoS-adaptive approximate real-time computation for mobility-aware IoT lifetime optimization", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, v. 38, n. 10, pp. 1799–1810, 2018.

[250] SIKDER, N., NAHID, A.-A. "KU-HAR: An open dataset for heterogeneous human activity recognition", *Pattern Recognition Letters*, v. 146, pp. 46–54, 2021.

[251] ABID, M. H., NAHID, A.-A. "Two Unorthodox Aspects in Handcrafted-feature Extraction for Human Activity Recognition Datasets". In: *2021 International Conference on Electronics, Communications and Information Technology (ICECIT)*, pp. 1–4. IEEE, 2021.