POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Anomaly Detection in Elderly Behavior Using TinyML and UWB Radar

Tesi di Laurea Magistrale in

Computer Science Engineering - Ingegneria Informatica

Alessandra Fiore, 991745

Abstract:
In the context of elderly care, there is an emerging need for advanced technologies that can improve health care delivery. This is due to the fact that the elderly population is increasing and with it the challenges related to the health and well-being of the elderly. Machine Learning (ML) technologies can provide valuable support for health carers in identifying significant changes in the conditions of patients under their care. At the same time, the technologies in this field, in order to be used and accepted by the final user, require to be non-invasive, discrete and respectful of each one habits and lifestyle. Wearables and smart cameras are examples of technologies that showed this kind of rejection from the elderly population: wearables require, in fact, to be constantly carried by the users in order to work, while cameras are often perceived as too invasive of the final user privacy.
In this context, we propose an innovative Ultra-Wide-Band (UWB) radar-based distributed solution employing two distinct ML algorithms hierarchically working together, with the objective of enhancing privacy while taking into account the perception of older users. The two algorithms, namely the *in-sensor distance and presence estimation algorithm* and the *anomaly detection algorithm*, are designed to be run together on different devices of a distributed ML pipeline. The in-sensor algorithms have been specifically designed to match the severe constraints that characterize the TinyML environments. It is meant to be run on multiple UWB-radar sensors distributed in all the rooms of a house, and, inputing high-dimensional radar matrices, it outputs the presence and the distance of a target in a room. The *Anomaly Detection (AD) algorithm* is then executed on an edge device, gathering the output of all the smart sensors with the objective of detecting significant deviations from the standard habits of the patient. The presented distributed solution encompassing smart, tinyML enabled, UWB-radar sensors present a promising approach to the enhancement of elderly care, offering a privacy-preserving and non-invasive solution that can enhance the quality of life for seniors while at the same time limiting the burden on their carers.

Advisor:
Prof. Maneul Roveri

Co-advisors:
Massimo Pavan

Academic year:
2022-2023

Key-words: Tiny Machine Learning, Ultra-Wideband Radar, Anomaly detection, Indoor positioning, Deep Neural Networks

Contents

1 Introduction **3**
- 1.1 Problem and Formulation . . . 3
- 1.2 Thesis Structure . . . 4

2 Background **5**
- 2.1 Machine Learning . . . 5
  - 2.1.1 Deep Learning and Convolution Neural Networks . . . 5
  - 2.1.2 TensorFlow . . . 6
- 2.2 Introduction to TinyML . . . 6
  - 2.2.1 Model Compression Methods: Quantization and Pruning . . . 8
  - 2.2.2 TensorFlow Lite for Microcontrollers . . . 8
- 2.3 UWB Radar . . . 8
  - 2.3.1 Application of UWB radar in Localization problem . . . 9
- 2.4 One Class Classification for outliers detection . . . 9
  - 2.4.1 One Class Support Vector Machine . . . 9

3 Related Work **11**

4 Problem Formulation **12**

5 Proposed Solution **13**
- 5.1 In-sensor Presence Detection . . . 14
  - 5.1.1 Problem Formulation . . . 14
  - 5.1.2 Preprocessing, Algorithm, Output . . . 14
  - 5.1.3 Dataset . . . 16
- 5.2 In-sensor Distance Estimation . . . 16
  - 5.2.1 Problem Formulation . . . 17
  - 5.2.2 Preprocessing, Algorithm, Output . . . 17
  - 5.2.3 Dataset . . . 18
- 5.3 Anomaly Detection for Time Series . . . 19
  - 5.3.1 Problem Formulation . . . 19
  - 5.3.2 Preprocessing, Algorithm, Output . . . 20
  - 5.3.3 Dataset . . . 21
- 5.4 Evaluation . . . 21
  - 5.4.1 Accuracy . . . 21
  - 5.4.2 Mean Absolute Error . . . 22

6 Results **23**
- 6.1 Results of Presence Detection . . . 23
- 6.2 Results of Distance Estimation . . . 24
- 6.3 Results of Anomaly Detection . . . 25

7 Conclusion **29**

1. Introduction

1.1. Problem and Formulation

Machine learning has evolved into a fundamental element of contemporary technology. The combination of edge computing and the Internet of Things (IoT) offers a new avenue for applying machine learning techniques on resource-constrained embedded devices located at the edge of the network. Traditional machine learning typically demands significant power resources to perform predictions. However, the emergence of TinyML aims to transfer this burden from conventional high-end systems to low-end clients [31].
Machine learning technologies are also growing popular in elder care because to their effectiveness in patient management and monitoring, resulting in a higher standard of care.
UWB is an innovative radar technology mainly used for short-range communication and imaging [30]. UWB radar systems typically operate at low power levels and work on a wide frequency spectrum, enabling high resolution and precision in applications like imaging and localization. UWB technology is of particular interest in the context of home care since images produced by this type of sensor do not allow for recognition of the identity of a target, but are precise enough to track their activities. UWB-radar devices employing this type of sensors can be very discrete, considering they can even be embedded inside of walls, and non-invasive, as they do not need to be carried on the patient.
The primary goal of this thesis is to create an effective solution for monitoring the behavior of older individuals in home environments in order to detect behavioral anomalies early and deliver relevant notifications.
To achieve this goal, we designed a novel smart sensor solution, encompassing an Ultra-Wide-Band (UWB) radar sensor, an *in-sensor tinyML algorithm*, and an *anomaly detection algorithm* that uses the outputs of sensors deployed on all rooms of the house.
The *TinyML algorithm*, exploiting the UWB-radar data, allows us to assess both the presence and distance of an individual within the various rooms, thus enabling comprehensive and detailed monitoring of movement and activity within the home environment.
Successively, we propose the usage of an *anomaly detection algorithm* on an edge device that can identify deviations from the standard habits of the patient by collecting data in output from the TinyML algorithm.
These two technologies enable the privacy-preserving analysis of data coming from multiple sensors, reducing by a lot the communication of data outside of each device and of the whole ecosystem. A schematic representation of the system can be seen in Figure 1.

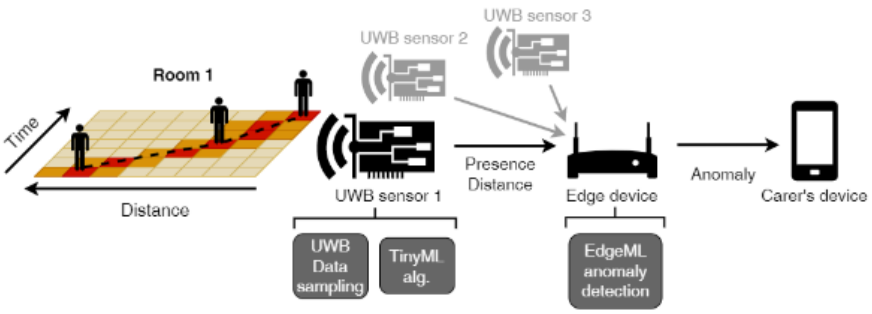


Figure 1: Architecture of the system.

The *TinyML algorithm* consists of a combination of two Convolutional Neural Networks, one for classification and one for regression, trained on data collected directly using the NXP UWB-SR 160 radar, the target hardware

that will be used also in the deployment phase.
The *anomaly detection (AD) algorithm* consists of a one-class SVM (OCSVM) trained directly on the data collected through the first period of use of the device personalizing it on the habits and schedules of the specific patient. All the data collected in this phase are considered representative of the standard behavior of the patient in the house.

After this phase, the algorithm becomes able to notify a carer about possible anomalies occuring in the house of the elder.

1.2. Thesis Structure

Chapter 2 of this thesis provides an introduction to machine learning, specifically delving into the topic of deep learning. It also lays out the fundamentals of TinyML and UWB radar technology, on which the research is based, and provides background on classification using SVM. Chapter 3 presents some articles focused on indoor positioning, highlighting how UWB radar technology offers significant advantages in this area of study. Chapter 4 describes the problem formulation, while chapter 5 describes the proposed solution, providing a more detailed description of the algorithms used and how they work together in a pipeline. Chapter 6 shows the results obtained from the proposed solution. Chapter 7 outlines the conclusions of the work.

2. Background

This section introduces essential topics for understanding the subsequent chapters. In particular, it will be explained what machine learning is and its advantages with respect to traditional algorithms. Subsequently, a special case of machine learning, deep learning, will be described, focusing on convolutional neural networks, essential for carrying out this thesis. Finally, some general information will be provided on the TensorFlow library, useful for designing neural networks.

2.1. Machine Learning

Within the broad field of artificial intelligence (AI), there is a special field called machine learning (ML). This subset is dedicated to fields of science and engineering that focus on the development of intelligent machines that can perform human-like tasks. Specifically, machine learning allows computers to learn on their own, eliminating the need for explicit programming. This is different from target programming, where functions are defined using formulated rules that clearly and statically define their behavior.
The advantage of the machine learning algorithm is obvious. A machine learning algorithm gathers knowledge from established data features and past experience (a learning process called training) to provide accurate predictions for new cases using knowledge from trained datasets, instead of time-consuming and trial-and-error methods, developing a unique program for each specific domain problem [13, 39].
In certain situations, the use of this customary method based on rules can create obstacles in extracting meaning from the data. In such cases, machine learning is used.
The growing demand for machine learning is due to the fact that many data sets are now available, and its ability to extract information from data is a significant advantage [23].

An artificial neural network (ANN) is a widely used machine learning algorithm inspired by the central nervous system and operating according to principles controlled by the human brain. ANNs typically consist of complex connections between neurons that allow messages to be passed between them.
The algorithm consists of three layers: an input layer for receiving input, a hidden layer for processing, and an output layer responsible for storing the final output. The connections between layers are formed by these weighted neurons, and during the training phase, the algorithm tries to modify these weights so that the system learns to create connections between the input and output layers [13].

2.1.1 Deep Learning and Convolution Neural Networks

Deep learning is a part of machine learning [34]. It involves neural network with multiple hidden layers, structured in deeply nested network architectures. Moreover, these networks generally contain advanced neurons, which distinguishes them from simple artificial neural networks. These features allow deep neural networks to receive raw input data and autonomously find the representation needed for the given learning task. Hence, deep learning proves to be particularly useful in fields dealing with extensive and high-dimensional data. This explains why deep neural networks outperform machine learning algorithms in various applications [14].

The Convolutional Neural Network (CNN) stands out as one of the most widely used deep neural networks, demonstrating exceptional performance in various machine learning problems. This is particularly suitable for applications involving image data. It derives its name from a mathematical linear operation among matrices known as convolution [1], that consist in taking input images and convolving them with filters or kernels to extract features [4].

Convolutional neural networks are made up of a series of layers, each of which plays a specific role in processing

input data. The five main layers that make up a CNN are:

- Input layer. Responsible for storing raw data,
- Convolution layer. Performs a dot product between the image and various filters to generate the output volume,
- Activation function layer. Apply an activation function to each element of the output of the convolutional layer,
- Pool layer. It is designed to improve the storage efficiency of the output of previous layers and reduce the computational cost,
- Fully connected layer. It takes input from previous layers and produces a one-dimensional array of computed class results [15].

2.1.2 TensorFlow

Originally created by researchers at Google, TensorFlow stands out among the many alternatives as a deep learning library.
In the expansive domain of deep learning, neural networks have achieved significant success, enjoying widespread popularity across various fields. Given its flexibility and scalability, this class of models has enormous potential for advancing data analysis and modeling in educational and behavioral sciences. However, the implementation of these complex models and optimization algorithms is a time-consuming and error-prone task.
TensorFlow is an important helper that greatly streamlines and accelerates both the research and application of neural network models. This tool proves invaluable in mitigating the challenges associated with the complex processes involved, making it a cornerstone for researchers and practitioners navigating the complex landscape of deep learning.
In particular, TensorFlow serves as a versatile and scalable software library designed for numerical computations through dataflow graphs. This collection of tools allows users to adeptly code, train, and deploy a variety of machine learning models, with a particular focus on neural networks. The functionality extends to production deployment, and TensorFlow offers APIs in multiple languages, with the Python API standing out as the most comprehensive and stable option.
Basically, a TensorFlow program involves two main phases: construction and execution.

- During the construction phase, TensorFlow functions are leveraged to create a computational graph that embodies a machine learning model. This graph, comprising edges and nodes, symbolizes the flow of data as tensors (e.g., vectors, matrices) through the system, according to the library's namesake. Nodes, called operations, describe computations (e.g., addition, multiplication) on tensors, taking inputs and generating outputs. TensorFlow provides fundamental components like fully connected and convolutional layers, along with nonlinear activation functions. The library also provides diverse loss functions, including cross-entropy and mean squared error (MSE).
- Upon entering the execution phase, the computational graph is made to move through a series of updating steps, aiming to train and enhance the model parameters.

This dual-phase structure highlights TensorFlow's role in both constructing intricate machine learning models and executing them for iterative improvement [29].

2.2. Introduction to TinyML

Whitin the domain of efficient machine learning, there exist two primary subdomains: edge machine learning (EdgeML) and cloud machine learning (CloudML).

CloudML is dedicated to improving latency and throughput on cloud servers, while EdgeML is focused on improving energy efficiency, latency, and privacy on edge devices. In particular, significant progress has been made in extending the scope of EdgeML to include ultra-low-power devices such as IoT devices and microcontrollers (MCUs), referred to as tiny machine learning (TinyML).

TinyML is a research topic focused on creating machine and deep learning models and algorithms that can run on resource-constrained devices including IoT or edge devices.

This approach offers several significant advantages. It facilitates machine learning with a minimal memory footprint, often utilizing only a few hundred kilobytes, which significantly reducing costs. By employing TinyML for ultra-low power tasks at the edge, a new approach called ML sensor emerges, providing a design where AI functionalities are not simply appended via cloud or mobile connections but are integrated within the sensor device itself. ML sensors provide a new paradigm for sensing by moving processing and analysis to the device rather than the cloud. This strategy gives priority to the proximity of data resulting in lower latency and increased data privacy as raw data is never transmitted. For this reason, the on-device processing of data proves advantageous in applications where real-time decision-making is paramount [22, 24, 38].

In TinyML, computational processes occur in close proximity to sensors, introducing new data analysis methods that were previously unavailable in resource-constrained settings. Key performance indicators that reinforce the effectiveness and necessity of TinyML can be delineated as follows:

a. Transition from Non-intelligent to Intelligent IoT Devices:
   Significant raw data generation from sensor systems limits cloud computing and the ability to process such data. Consequently, a significant portion of data remains unused at the edge without being delivered to the cloud. TinyML offers the ability to analyze this data in resource-constrained settings by integrating intelligence into each IoT device through the incorporation of machine learning algorithms.
b. Network Bandwidth:
   Compared to standard IoT services, TinyML offers greater independence, enabling low data transmission and potentially reducing bandwidth requirements. Analyzing raw data at the edge before transmission, particularly in dense IoT frameworks, significantly reduces the high bandwidth demand. This independence results in more efficient data transmission.
c. Security and Privacy:
   By avoiding extensive data transmission and limiting data to the device, TinyML enhances security and privacy. In TinyML, data either does not move or moves minimally, reducing exposure to potential attacks. Consequently, security and privacy features are embedded in TinyML by default and design.
d. Latency:
   The traditional IoT system involves the transfer of sensor data from IoT devices to cloud servers, culminating in receiving cloud-computed decisions or predictions. This sequence introduces significant latency, which requires analysis close to the device. TinyML serves as a foundation for providing significantly reduced, near-zero latency, as it relies less on external communication.
e. Reliability:
   TinyML has been identified as a solution for on-site tasks, especially in areas with limited cellular connectivity or internet access. This change significantly increases the reliability of IoT services.
f. Low Cost:
   Reducing data traffic, and consequently bandwidth requirements, leads to a reduction in data transmission and communication costs [8].

Despite the benefits, there are some challenges to navigate: the efficiency of deep learning models often comes with significant computational requirements, which prevent their integration into TinyML applications due to the severe resource limitations of devices like microcontrollers [22]. Creating Machine and Deep Learning

(MDL) solutions that can run on these small devices requires a complete overhaul and redesign of MDL models and algorithms. It is important to consider the significant constraints on memory, computation, and power consumption inherent in these devices. This is precisely the domain where TinyML becomes instrumental. It involves the design, development, and deployment of MDL models and algorithms tailored for small devices. TinyML typically introduces compact MDL architectures and approximate computing solutions (such as quantization, pruning) to accommodate the stringent technical limitations characterizing these small-scale devices [30].

2.2.1 Model Compression Methods: Quantization and Pruning

The challenges mentioned above can be successfully addressed using optimization methods such as network compression. Network compression is often achieved with minimal compromise in accuracy, and in some cases, has the potential for improvement, making it a viable solution. Two commonly used techniques for compressing networks are pruning and quantization [21]. The goal is to reduce memory and energy required to perform inferences over large networks and facilitate implementation on small devices [11].

- Network pruning stands as a crucial method for reducing both memory size and bandwidth. By removing redundant parameters or neurons that do not significantly contribute to accuracy, pruning addresses situations where weight coefficients are zero, close to zero, or duplicated. This process effectively reduces the computational complexity [21].
- Quantization is defined as the procedure of approximating a continuous signal with a collection of discrete symbols or integer values. This approach specifically aims to reduce the model's storage cost. In particular, weight quantization involves discretizing the range of weight values, allowing each weight to be expressed using a smaller number of bits [21, 44].

2.2.2 TensorFlow Lite for Microcontrollers

TensorFlow Lite Micro (TFLM) is a customized version of TensorFlow [40]. It is an open-source machine learning inference framework designed for running deep-learning models on embedded systems. TFLM addresses the challenges associated with limited resources and platform differences in embedded systems, challenges that typically prevent cross-platform compatibility. The framework adopts a unique interpreter-based approach, providing flexibility to overcome these specific challenges. In particular, a trained model, such as a TensorFlow model, can be transformed into a TensorFlow Lite '.tflite' model file using an exporter. Once converted, this file can be installed on a client device, like a mobile device or embedded system, and executed locally using the TensorFlow Lite interpreter [7].
TFLite is specifically designed to operate machine learning models on hardware with restricted computational capabilities. It employs techniques like quantization and pruning to decrease the model's size and improve its speed. Moreover, it features a fine-tuned runtime to ensure the efficient execution of models across various embedded devices [40].

2.3. UWB Radar

Ultra-Wideband (UWB) radar belongs to a class of radio systems characterized by very wide bandwidths, ranging from 3.1 to 10.6 GHz [5]. This means that it enables the frequency spectrum to be shared among different users.This means that it enables the frequency spectrum to be shared among different users, helping to manage interference and signal path complexity in radio communications [36].
An ultra-wideband radar functions as a sensor that measures range using time-of-flight-based techniques. It emits short pulses and captures reflected waves from objects in the surrounding environment. The resulting

echo waveform is the sum of these reflected waves, and the time delay for each wave depends on the distance between the sensor and the target. The received waveform is therefore a combination of individual echoes that vary based on the distance, material, shape and radar reflectivity of obstacles in the environment [41].

2.3.1 Application of UWB radar in Localization problem

The necessity for accurate location tracking has become more pressing, especially in environments with a lot of obstacles where the Global Positioning System (GPS) might not always be reliable or enough precise.
Ultra-wide bandwidth technology is an effective solution for achieving precise positioning in these challenging environments. Its unique ability to pass through obstacles makes UWB as a promising technology for ensuring accurate localization, particularly in cluttered and challenging scenarios [6].
UWB stands out for its exceptional accuracy, which allows it to precisely locate individuals and objects within a range of just a few centimeters. This level of precision far exceeds Bluetooth Low Energy (BLE) and Wi-Fi current implementations by a significant margin, making UWB technology 100 times more accurate than these existing alternatives [43].

2.4. One Class Classification for outliers detection

The traditional approach to multi-class classification is designed to classify an unknown data object into one of several predefined classes, typically two in the case of binary classification. However, challenges arise when the data object doesn't fit into any of these predefined categories. In some cases, the goal of classification is not simply to assign a test object to predetermined classes, but rather to determine whether it belongs to a specific class or not. In the context of One-Class Classification (OCC), one class, arbitrarily labeled as the positive or target class, is well-defined by instances in the training data, while the other class, labeled as negative or outlier, either lacks instances or has very few of them.
Therefore, in a typical multi-class classification scenario, the definition of a decision boundary is done using data from two or more classes, supported by the presence of data objects for each class. Traditional classifiers generally assume relatively balanced data classes and struggle when faced with severe under-sampling or the complete absence of any class. In One-Class Classification scenarios, either the negative data objects are scarce or entirely absent, allowing only one side of the classification boundary to be determined based on positive data (or some negatives). The objective in OCC is to delineate a classification boundary around the positive class, trying to include as many objects as possible from the positive class while minimizing the probability of receiving an outlier.

Of particular interest to this thesis are one-class classification algorithms based on One Class Support Vector Machine (OCSVM) [16].

2.4.1 One Class Support Vector Machine

SVM, a recently developed data mining and machine learning approach, has gained wide recognition and popularity in various domains due to its overall superior capabilities compared to traditional intelligent methods such as neural networks. The fundamental concept of SVM involves creating a hyperplane to effectively separate two classes of data, maximizing the margin between them. This margin is defined as the minimum distance from the data points to the hyperplane. By transforming the datasets into a higher dimensional feature space, linear separability is achieved, and the hyperplane is constructed within this space.
One-class SVM represents a specialized version of SVM that utilizes only a regular feature dataset during the training phase. Its primary aim is to define a decision boundary with as much margin as possible between regular data points and the origin. Samples falling within this boundary are classified as normal points, while

those outside the boundary are identified as anomalies [45].

For all the above reasons, the OCC problem is used across various research themes, including anomaly and novelty detection [16].

3. Related Work

In the scientific literature, the problem of indoor location has received considerable attention, highlighting the importance of developing reliable indoor positioning systems.
Different methods consider the use of wearable sensors by the subject or the use of cameras to monitor the surroundings. However, one possible drawback to using this type of method in domestic context is that residents have conveyed dissatisfaction with cameras due to concerns related to privacy as well as they are reluctant to put on any sensor or label in the case of wearable sensors [27]. For example, the study conducted in [17] introduce a localization method based on radio-frequency identification (RFID) technology to accurately monitor the elderly. The RFID system consists of readers and tags. Precisely an RFID tag affixed to living beings comprises an antenna designed to both receive and transmit RF signals, while an RFID reader establishes communication with one or multiple tags within its range, transmitting the acquired data to the backend server for additional processing.
Differently, the use of UWB radar for indoor tracking, as in this study, represents a totally non-intrusive system, eliminating the need for wearable tags or sensors. This feature gives the system a significant advantage in terms of convenience and comfort for users, as it does not require any additional device to be worn.
A significant contribution in this area is the study conducted by Samuel G. Leitch et al., who delved into the evaluation of four different technologies within Indoor Positioning Systems (IPS): Wi-Fi, Bluetooth Low Energy (BLE), Inertial Measurement Unit (IMU) and Ultra-Wideband (UWB) on mobile phones.
The goal of the study is to achieve an average estimation error of less than 10 cm for indoor localization. The study has shown that among these technologies, only WiFi and UWB demonstrate the ability to achieve this level of accuracy independently [19].
Another study led by Stefania Monica and Federico Bergenti focuses exclusively on comparing UWB and WiFi technologies, specifically emphasizing the advantages and disadvantages associated with adopting each of these solutions. These technologies present varying degrees of localization accuracy and are distinguished by their diverse applications. UWB delivers highly precise localization information; however, its implementation necessitates a specialized infrastructure and is not yet prevalent in mobile devices. On the other hand, WiFi provides less accurate localization information, but it is seamlessly integrated into all contemporary mobile devices and does not demand a dedicated infrastructure [28].
These works, utilizing UWB ranging solutions, still rely on devices (mobile phones) used as tags to be carried by the monitored patients. On the other hand, the UWB-radar solution proposed in this thesis offers an alternative without the need for an additional phone or tag. This method allows monitoring without requiring patients to carry a specific device. Moreover, while these studies have investigated the use of UWB technology for indoor localization, none have designed and developed a solution specifically for tiny devices. Such a solution is able to perform operations or calculations quickly and efficiently using a limited amount of computational resources. Although there are several solutions that take advantage of UWB technologies integrated into resource-constrained devices, as demonstrated in the study on indoor presence detection in [30], the presented distributed solution encompassing smart, tinyML enabled, UWB-radar sensors present a promising and unique approach to the enhancement of elderly care, offering a privacy-preserving and non-invasive solution that can enhance the quality of life for seniors while at the same time limiting the burden on their carers.

4. Problem Formulation

The main objective of this research is to develop an advanced system for monitoring the behavior of elderly people within their homes so that any anomalies or deviations from normal behavior can be quickly identified. The problem concerns the analysis of data acquired from a UWB radar, particularly, the most recent k radar acquisitions, being k, a parameter specific to the application, that is chosen by the designer.
The problem can be reformulated as the design of a system $f_v(x_t, x_{t-1}, x_{t-2}...x_{t-(k-1)})$ able to map the frames of the radar $(x_t, x_{t-1}, x_{t-2}...x_{t-(k-1)})$ into the label A_t, being k defined as the length of the observation window, x_t the radar acquisition at time t with dimensions $N \times M$, where N denotes the number of pulses produced and, therefore, returned to the radar, and M denotes the number of bins, or discrete intervals of space, and A_t the label where $A_t \in \{abnormal, normal\}$. Abnormal in this context refers to a pattern of behavior that deviates significantly from the data collected in the first period of device use, which is aimed at collecting the subject's habits. An example of abnormal behavior might be prolonged standing in a specific position suggesting the possibility of an undesirable event such as a fall.

5. Proposed Solution

In the specific context of the research, the problem has been subdivided in two interconnetted sub-problems. One addressed the task of designing a sensor technology based on UWB and TinyML. This sensor, exploiting data from UWB radar, was designed to identify the presence of a subject within a room and determine its distance from the radar. In parallel, an algorithm capable of performing anomalous pattern detection in the time series composed of the outputs of these sensors in the various rooms was developed.
The developed algorithm was designed to recognize nonconforming or unusual patterns within the time series, thus signaling any risky situations or emergencies.
The proposed solution for the sensor problem involves the use of two TinyML algorithms, that are deployed on the embedded devices placed in each room and analyze the radar acquisitions:

- The *in-sensor presence detection algorithm*, that aims to evaluate the presence or absence of the subject.
- The *in-sensor distance estimation algorithm*, which provides detailed information on its position in the room in the case of detected presence.

The identification of the presence of the subject, alongside the detailed distance measurement, are important to understand where the elderly person is located in the room, thus providing a complete picture of their behavior and mobility within the home environment.
Subsequently, an edge device aggregates the results generated from the TinyML algorithms to compute information related to the collected data, such as the mean and standard deviation of distances recorded for each room in a given time window. These parameters provide a detailed overview of movement dynamics or statics over time in elderly subjects. Utilizing the mean of distances provides an approximation of the subject's average positioning within each time interval, while the standard deviation offers an indication of the variability of these positions and the amount of movement performed by the target. These metrics are crucial for understanding behavioral patterns and detecting any anomalies or significant alterations.
These new insights are then utilized to construct a time series, which will be fed into the multi-device *anomaly detection algorithm*. This algorithm will evaluate the time series to determine whether it deviates from the subject's typical behavior, consequently classifying it as anomaly or normal.

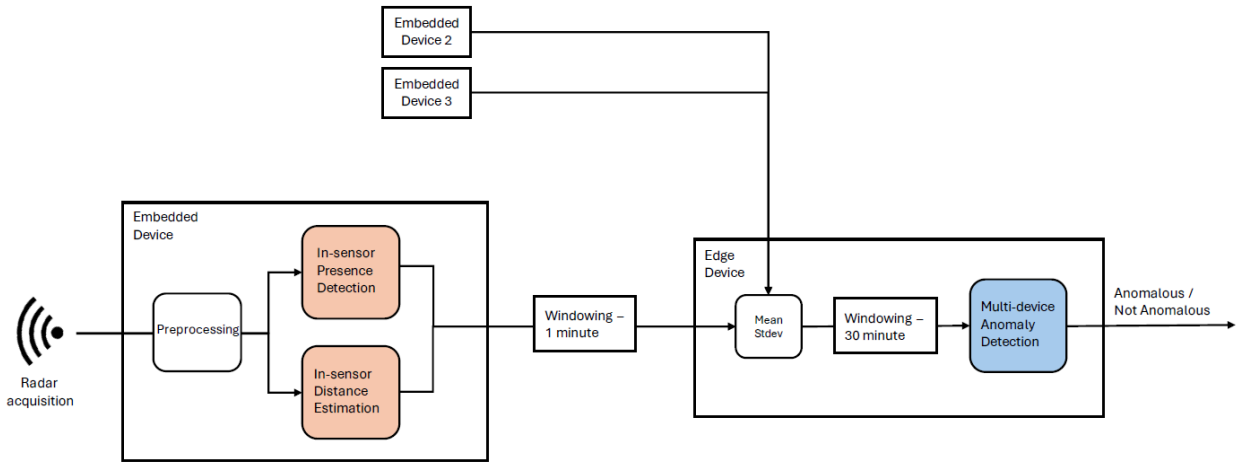


Figure 2: Proposed solution. In orange the TinyML algorithm. In blue the Anomaly Detection algorithm.

5.1. In-sensor Presence Detection

This section concentrates on the initial aspect of the problem, which is classifying the presence or absence of a subject. Here, we will introduce the problem, discuss the dataset utilized for the method, delve into the data preprocessing phase, and outline the algorithm itself.

5.1.1 Problem Formulation

Let $s_t \in \mathbb{R}^{N \times M}$ denote the acquisition obtained by the UWB radar at time t, with $s_t \in S$, where S refers to the complete collection of radar acquisitions. Here, $N, M \in \mathbb{N}$ represent the dimensions, where N denotes the number of pulses produced and, therefore, returned to the radar, and M denotes the number of bins, or discrete intervals of space.
The problem is assigning to data s_t the label y_t, where $y_t \in \{absent, present\}$.

$$y_t = P(s_t) = \begin{cases} 0 & absent \\ 1 & present \end{cases}$$

5.1.2 Preprocessing, Algorithm, Output

This algorithm is based on the use of data acquired by UWB radar. It is essential to perform a preprocessing step on the data before feeding it to the *in-sensor presence detection algorithm* in order to make it informative and clean, removing any noise or superfluous information.
Each data point acquired by the radar UWB, which is rich in useful information, has a size of 11x248, where $N = 11$ represents the time dimension dedicated to capturing the data point. This time interval reflects the temporal accuracy of the radar, providing a detailed capture window. $M = 248$ denotes the space dimension (spatial bins) and correspond to the information received from the return signal at each time step, evaluated at different spatial locations. The spatial dimension contains a representation of signal amplitude and phase alternated, expressed through complex numbers that include a real and an imaginary part, thus can be seen as 124 complex values.
Then absolute value was performed on the entire matrix. This operation, that is represented by the absolute value between the real and imaginary part, aims to refine the representation by preserving only the amplitude information of the signal, since the phase of the signal did not provide crucial information in this experimental context. Consequently, the matrix was reduced to a size of 11x124 real values. This transformation greatly simplified the data structure, retaining only the information essential for the analysis of the subject's motion and presence in the experiment conducted. The updated matrix configuration represents a significant step in centralizing relevant information and reducing data complexity, making it more suitable for specific research and interpretation needs.
Another crucial phase in processing the data involved employing the decluttering process, an essential technique designed to eliminate extraneous noise from the dataset, preserving only pertinent information. The employed declutter method is the *moving average filter* technique, that works by calculating the average over a given range of data and subtracting this average from the original data.
Following the decluttering process, an additional data processing step was performed. Specifically, each data matrix was subjected to targeted trimming to exclude the least informative parts of the record, that didn't provide information regarding the subject's behaviors, such as those contained in the near and far spatial bins. At the end of this phase, there was a total of 56 spatial bins, so a matrix of size 11x56 was generated.
In the last data preprocessing step, each sample undergoes normalization before being used as input for the algorithm.

Normalization of data is a pre-processing technique that involves either scaling or transforming the data to ensure each feature makes an equitable contribution.
In particular, min-max normalization was used. The approach involves linearly scaling the non-normalized data to predetermined lower and upper bounds. Typically, the data is rescaled to fall within the range of 0 to 1:

$$x_{norm} = \frac{x - min}{max - min}$$

This crucial pre-processing step aims to bring features within a common range, preventing larger numeric values from overshadowing smaller ones. When the relative importance of features is uncertain, normalizing the features in the dataset ensures that they are equally significant when predicting the output class of an unknown instance [35].

At the conclusion of this preprocessing step, the data I_t is ready to be used by the *in-sensor presence detection algorithm* to determine the presence or absence of the subject.
To handle the task of distinguishing between presence and absence, an approach involving the use of a Convolutional Neural Network functioning as a binary classifier was implemented. This choice was driven by CNN's ability to independently and efficiently acquire essential image features [20].
The model was formulated as Sequential Models utilizing TensorFlow's Keras API and consists of several layers that play a specific role in the feature extraction and classification process. Its input layer is a Conv2D layer specifically engineered to process input images formatted as two-dimensional matrices sized 11x56. It used 16 3x3 filters and relu activation function. Then, a second Conv2D, now with 32 filters, is employed for detecting more complex and abstract patterns of the input.
After each Conv2D layer, a MaxPooling2D layer was added to reduce the spatial dimensions of the input.
Next, through the Flatten layer, the data is transformed into a one-dimensional vector before moving to the Fully connected layer and the network output layer.
Before the output layer, the CNN includes a Dropout layer. The Dropout layer randomly removes some units and their connections from the neural network during the training phase. This reduces overfitting, a situation where the model fails to generalize successfully from observed to unseen data.[37, 46].
This dropout layer is included at a rate of 10%. This means that during training, 10% of the units are randomly deactivated at each iteration.
The class head of the neural network consists of a single dense layer with one node and sigmoid activation function. This layer produces a number between 0 and 1, with 1 denoting a high probability of presence and 0 a low probability. The L2 regularization, set to 0.0001, is incorporated in this layer. This regularization parameter adds a penalty as the complexity of the model increases, so the model generalizes the data and prevents overfitting [26].

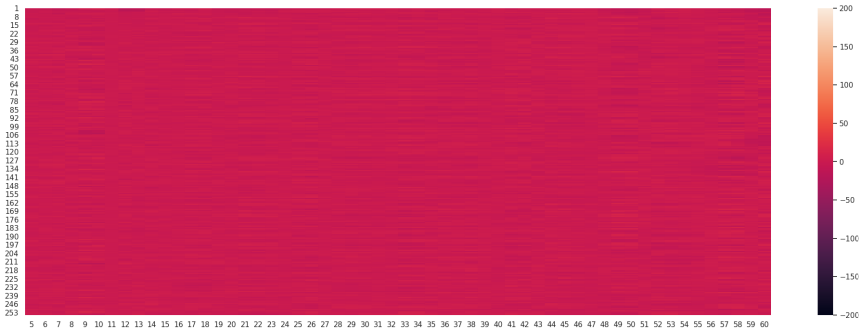
Layer	Hyperparameters
Conv2D	(16, (3,3), activation='relu', input_shape=(11, 56, 1))
MaxPooling2D	(2, 2)
Conv2D	(32, (3,3), activation='relu')
MaxPooling2D	(2, 2)
Flatten	()
Dense	(50, activation='relu')
Dropout	(0.1)
Dense	(1, activation='sigmoid', kernel_regularizer=l2(0.0001))

Table 1: CNN Architecture for Presence Detection.

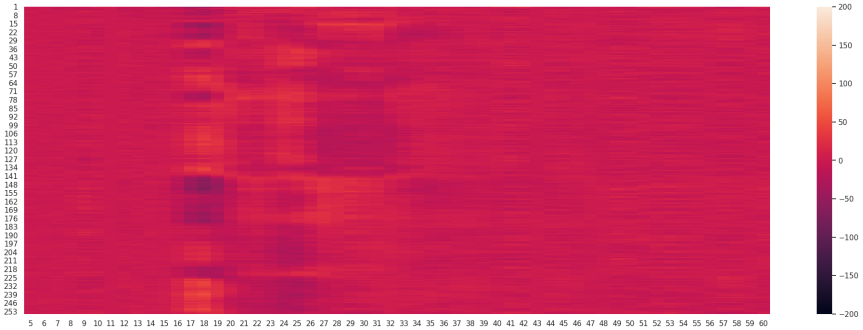
It is important to note that careful parameter selection was made in the design of the neural network model, as it directly affects the performance and robustness of the system. In this context, a thoughtful approach was taken in selecting key parameters, such as the number of filters for convolution layers, dropout rate and regularization term, carefully balancing performance requirements and efficiency in memory utilization.
After experimenting with numerous combinations of these parameters, the final setting was chosen based on the combination that provided the best balance of accuracy and occupied memory.

5.1.3 Dataset

For the training of the CNN, a dataset acquired through ultra-wideband radar was used. This approach aims to ensure that the CNN is finely tuned to process real-world data.
Specifically, a total of 805 data with $N = 11$ and $M = 56$ were acquired, each lasting 1.1 seconds thus with a frequency of 10Hz. These data were categorized into two distinct classes: the first includes situations in which a subject was positioned within the range of the radar, generating data in which the presence of an object was detected. The second class, on the other hand, includes data corresponding to situations in which there were no objects detected within the radar range, thus representing the absence condition. In particular, there are 322 cases corresponding to "non-presence" situations; the remaining number of data, 483, represents "presence" situations, so as to ensure a balanced representation, thus contributing to accurate classifier training. The division of data into the three sets - training, validation, and test sets - is described later in Chapter 6.1.



(a) 'Absent' data acquired.



(b) 'Present' data acquired.

Figure 3: Date acquired by radar in the presence and absence of a subject.

5.2. In-sensor Distance Estimation

This section delves into the challenge of estimating the distance between the subject and the radar. We will introduce the issue and illustrate the model utilized to tackle it. Furthermore, we will explore the rationale behind selecting this model and justify its suitability for resolving this specific problem.

5.2.1 Problem Formulation

This problem uses the same UWB radar acquisition $s_t \in \mathbb{R}^{N \times M}$ previously described for the presence detection problem in Section 5.1.1.
This time, the goal is to assign to the data s_t a value d representing the distance between the subject and the radar. In particular, distance d is used to determine the position of the subject inside the room with a range between values around 0.50 to approximately 2.50 meters.

$$d = D(s_t)$$

5.2.2 Preprocessing, Algorithm, Output

For data input to the *distance estimation algorithm*, the preprocessing step is identical to that used for data input to the *presence detection algorithm*.
A second Convolutional Neural Network was employed to solve the distance estimation problem. The parameters used were chosen following the same careful selection of parameters made for the *presence detection algorithm*. In particular, it has the same two Conv2D layers, but in contrast to the network utilized for presence/absence classification, this particular network was set up for regression. The network uses a dropout rate of 0 and as output layer it uses a single neuron with L2 regularization set to 0.0001 and liner activation function that produces a continuous value as output. This implies that its objective is to forecast a continuous variable (distance) rather than a discrete one.
In the output layer, the activation function is set to 'linear' instead of 'sigmoid'. Its definition is as follow $F(x) = ax$. In other words, a linear activation function is directly proportional to the input [33]. This choice reflects the nature of regression, where the goal is to predict continuous quantities rather than binary probabilities. "Linear" activation allows the network to produce outputs that can vary over a continuous range without any probability constraints.
In particular, this network was trained using data containing information about the actual distance from the radar. Configured to understand and forecast continuous distances, the network enables precise estimation of the subject's position relative to the radar.

Layer	Hyperparameters
Conv2D	(16, (3,3), activation='relu', input_shape=(11, 56, 1))
MaxPooling2D	(2, 2)
Conv2D	(32, (3,3), activation='relu')
MaxPooling2D	(2, 2)
Flatten	()
Dense	(20, activation='relu')
Dense	(1, activation='linear', kernel_regularizer=l2(0.0001))

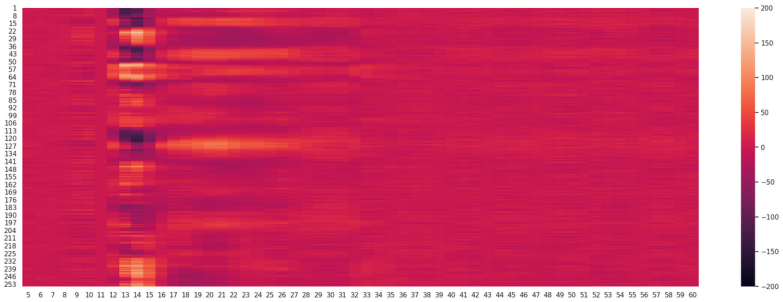
Table 2: CNN Architecture for Distance Estimation.

5.2.3 Dataset

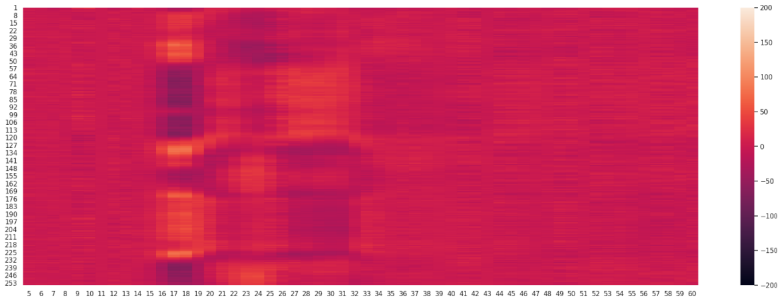
The dataset used in this phase of the problem were collected in a specifically designated space (the same for the presence detection problem) covering a maximum distance of 2.50 meters from the radar. This choice was guided by careful observation of the data: the results revealed that beyond this distance, the radar's ability to accurately detect the presence of an object decreases significantly. This limitation guided the selection of the data acquisition radius, ensuring that the information collected was reliable.

Figure 4: Target positioning in the test environment.

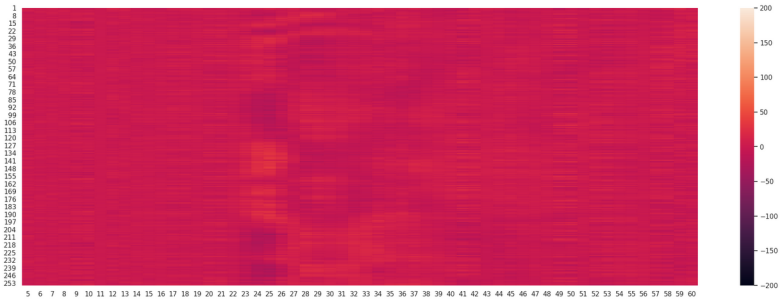
The image shows the test environment used for data collection. The distinctive marks on the floor in Figure 4 were purposely placed to identify the three target distances considered during the experimental tests: 0.50 meters, 1.25 meters and 2.50 meters.
The test environment, as shown in the photo, is limited in size, making it impractical to cover the complete radius during data gathering. As a result, only small angle ranges were considered: -30 to 30 degrees from the radar axis at 1.25 meters and -15 to 15 degrees at 2.5 meters. This selection of angles aims to make the best use of the constrained surroundings.
The data collection phase used to address regression problem focused on the three pre-specified target distances and corresponding angular ranges.
A total of 1725 data points have been recorded during data collection exploited for the CNN. These data were distributed among the three distances, with 437 collected at 0.50 meters, 644 at 1.25 meters, and a further 644 at 2.50 meters. The division of data into the three sets is described later in Chapter 6.2.
This method tries to enable the convolutional neural network to effectively estimate continuous distances within the certain range.
Examples of data can be seen in Figure 5.



(a) Data acquired at 0.50 meters.



(b) Data acquired at 1.25 meters.



(c) Data acquired at 2.50 meters.

Figure 5: Data acquired by radar over the 3 distances.

5.3. Anomaly Detection for Time Series

This chapter focuses on the last stage of the problem, which concerns the detection of anomalies in the behavior pattern of the elderly subject.

5.3.1 Problem Formulation

The problem aims to apply anomaly detection techniques in a multivariate time series T_i, which describes the behavior of the subject in an example apartment and in a specific time window. The time series T_i spans 30 time points and encompasses the time instant represented using hour and minute information and 3α variables, where α is the number of the rooms and 3 are the key information presence, mean distance, and standard deviation calculated from the collection of the outputs of the tinyML algorithms for each room and for each time step.

The *multi-device anomaly detection algorithm* assigning the value A_t to T_i, where A_t is the output of the

algorithm and $A_t \in \{abnormal, normal\}$.

$$A_t = AD(T_i) = \begin{cases} 0 & abnormal \\ 1 & normal \end{cases}$$

5.3.2 Preprocessing, Algorithm, Output

Using the outputs obtained with the two tiny NNs, if presence is detected, we compute the mean and standard deviation of the distances within each interval of granularity of 1 minute.
Particular attention was paid on selecting the correct level of temporal granularity. The granularity of the repetition interval δt must be determined in advance for a particular model and this parameter can be set according to the needs of the application [12]. For this reason, the granularity of 1 minute was chosen according to [3] that describes a problem very similar to this part of experiment.
Once the temporal granularity is established, emphasis is placed on calculating key information, mean and standard deviation of the distances, within each interval of this granularity.
Following this, we proceeded to structure this information into temporal windows, composed of the data collected during each granularity interval. These time windows function as multivariate time series, enabling for a more detailed analysis of behaviors over time. By examinating these time series, specific patterns can be identified and the presence of anomalous behaviors can be assessed. In particular, the time series are analyzed in windows of 30 minutes. The choice of the time window, again, depends on the needs of the application. If the window size is too large, the time window becomes more difficult to analyze. On the positive side, a larger window is effective in detecting longer anomalies such as an extended stay in a room. [2].

	23:00	23:01	23:02	23:03	23:04	23:05
1	1	1	1	1	1	1
2	2.4598227357707243	2.5716393779584714	2.5441118938146463	2.4674091116865213	2.5149021657514217	2.495306122516037
3	0.2130617748530524	0.18584310151187652	0.24642398459415943	0.09983998889037005	0.27272098233949116	0.17828143812497277
4	0	0	0	0	0	0
5	0	0	0	0	0	0
6	0	0	0	0	0	0
7	0	0	0	0	0	0
8	0	0	0	0	0	0
9	0	0	0	0	0	0

Figure 6: First 6 minutes of a 30-minute time series. The first three rows indicate presence/absence, average distance, and standard deviation for the bedroom; the next three rows represent the same attributes for the kitchen, and the last three rows represent the same attributes for the bathroom.

Before the time series is given as input to the *anomaly detection algorithm*, for each granularity interval δt in the series, the corresponding time is calculated in trigonometric format, that is, expressed in hours and minutes in the form of sine and cosine. This continuous representation of time highlights the cyclic nature of time, preserving temporal information and allowing the algorithm a more effective interpretation of time.

The anomaly detection algorithm consists of a One Class SVM used to identify unexpected behaviors. The OCSVM is trained only on normal data, allowing anything that deviates from the hyperplane of normal data to be identified as anomalous.
The model is structured using the RBF kernel and particular attention was employed in the choice of gamma and nu parameters. These parameter selections are adjusted to maximize the model's efficiency in accordance with the dataset's features. The parameter selection for the model was based on the combination that achieved the best accuracy on the validation set, a dataset used for evaluating model performance on unseen data during

training and adjust model parameters to improve model performance.
The kernel is a mathematical function that determines the characteristics of the hyperplane in the feature space [9]. In this case, the RBF (Radial Basis Function) kernel is employed, a popular choice in SVMs due to its adaptability in identifying nonlinear decision boundaries, while gamma is a specific parameter of the RBF kernel. After testing different gamma values for the model, a value of 0.05 was chosen. A key feature of OCSVM is nu that sets an upper and lower bound on the fraction of outliers and support vectors, and which was set to 0.1 for the model. The value selection of this parameter has a significant impact on the accuracy of a model built by the OCSVM algorithms [10].
Finally, after training the One-Class SVM model with these parameters, a threshold is set using the decision scores of the training data. If a data point's decision score is higher than this threshold, it's considered non-anomalous; otherwise, it's classified as an anomaly. The threshold is calculated as the 0.05th percentile of the decision scores, meaning around 5% of the training points will be labeled as anomalies.

5.3.3 Dataset

The dataset used for training the OCSVM model was generated by exploting the outputs of the *in-sensor presence and distance estimation algorithms*, simulating their usage on a real-world apartment composed of 3 rooms. This dataset consists of 30-minute time series, each represented by a 13x30 matrix. Of the 13 rows of the matrix, the first 9 rows represent the presence, mean, and standard deviation information for the 3 rooms; the last 4 rows are used for the trigonometric representation of time. Each column of the matrix corresponds to a single minute summarizing values of the sensors (presence, average distance, standard deviation).
To create these data, typical behavior of a subject during the day was assumed so the informations listed above were entered based on this pattern. Moreover, these informations were entered by first studying their values on the actual data. For example, the standard deviation values used are all in the range of 0.09-0.3 to simulate static behavior during the minute, and in the range of 0.65-0.75 to simulate movement between different positions. To choose these values, the standard deviation was first calculated on the data representing the static presence of a subject in front of the radar at a given distance, and then on data where the position changed between the 3 different distances.
Finally, the dataset used for the OCSVM has 2160 data in total, including 2150 normal data and 10 anomalies used to test the model's ability to identify outliers. The data are then divided into the three sets as described in Chapter 6.3.

5.4. Evaluation

In the process of evaluating neural networks and the OCSVM, it is crucial to adopt appropriate metrics that effectively reflect the performance of the model with respect to the specific objectives of the task. Two distinctive metrics were used, Accuracy (Acc) for evaluating the OCSVM together with the convolutional neural network intended for classification and Mean Absolute Error (MAE) for evaluating the convolutional neural network intended for regression.

5.4.1 Accuracy

Accuracy is a measure of classification models' accuracy. Informally, this metric is the fraction of predictions that our model made correctly, but in binary classification accuracy can also be calculated as positives and negatives. In this case, it is calculated as the ratio of the number of correct predictions ($TP + TN$) to the total number of occurrences (N) in the test dataset:

$$Acc = \frac{TP + TN}{N}$$

Where TP represents the True Positives, TN the True Negatives, and N is the sum of all instances in the test dataset [18]. This metric provides an assessment of the model's ability to correctly classify different categories of inputs. A high Accuracy indicates an excellent ability of the model to discriminate between classes, while a lower value suggests possible areas for improvement.

5.4.2 Mean Absolute Error

The Mean Absolute Error (MAE) is a metric used to evaluate the accuracy of predictions in regression problems. This metric is calculated as the average of the absolute differences between model predictions (y) and actual values (x) in the test data set (N):

$$MAE = \frac{\sum_{i=1}^{N} |y_i - x_i|}{N}$$

[25].
A lower MAE indicates greater accuracy in predictions of numerical values because discrepancies between predictions and actual values are minimized.

The joint use of Accuracy and MAE metrics provides a comprehensive assessment of the performance of the models. Accuracy reflects the model's ability to correctly classify different categories of input, while MAE measures accuracy in predicting numerical values. Combining both metrics contributes to an in-depth understanding of the model's capabilities and limitations in diverse scenarios, ensuring an overall assessment of its performance.

6. Results

This section discusses the results obtained with the models described in the previous chapters.

6.1. Results of Presence Detection

Of the 805 data points in the dataset utilized for the *presence detection algorithm*, 552 were designated for the training set. This subset is used during the model's training phase to enhance comprehension of relationships and features within the data. Of the total, 115 were allocated to the validation set. This dataset is crucial for adjusting model parameters and preventing overfitting, helping to ensure that the model generalizes well even on new data not seen during training. The test set consisted of the remaining 138. This dataset remained untouched during both the training and validation phases of the model. It stands as the ultimate benchmark, assessing the actual performance of the model on previously unseen data.

Table 3 demonstrates the accuracy obtained by the CNN employed in the presence detection problem across the three different sets (training, validation, test).

train accuracy	val accuracy	test accuracy
0.9475	0.9304	0.8768

Table 3: CNN classifier results: accuracy values.

The optimizer 'adam' and the loss function 'binary_crossentropy', typically used in binary classification problems, are employed in the neural network. An optimizer aims to minimize the loss function, which represents the difference between predicted and expected values. Minimization is the process of determining the set of architecture parameters that produce the greatest results in the desired tasks [42].
In this setup, the neural network is trained for 400 epochs with early stopping, which stops the training process if no improvements are seen for 30 consecutive epochs. Early stopping is an approach used to avoid overfitting and assure the model's optimal performance throughout training.
The next table reports the loss values obtained for the training, validation and test sets.

train loss	val loss	test loss
0.1373	0.2264	0.2894

Table 4: CNN classifier results: loss values.

The confusion matrix below, on the other hand, shows the algorithm's predictions made about each datum in the test set versus their actual result. To convert the probability into a binary decision (presence or absence), a classification threshold was established. In this implementation, a threshold of 0.35 was selected. Essentially, all predictions made above this threshold will be classified as presence; conversely, they will be calssified as absence. This threshold choice was optimized based on results obtained through data acquired with the radar.

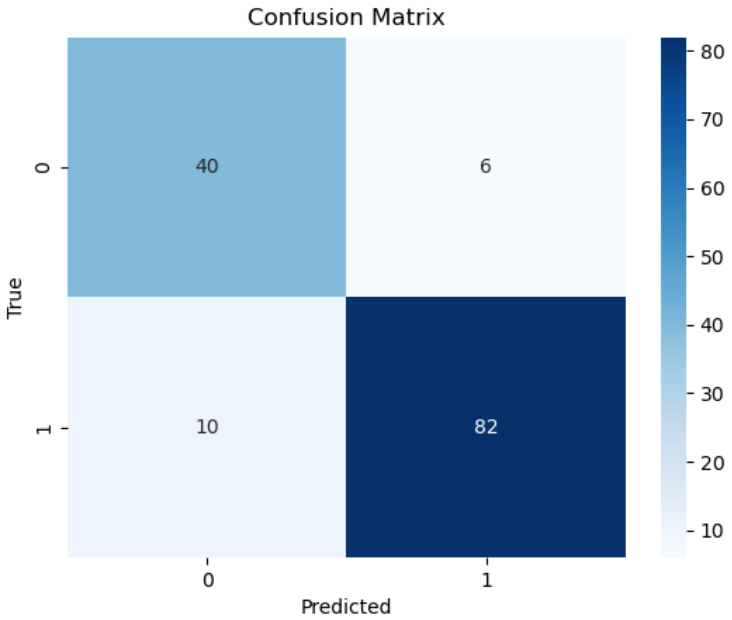


Figure 7: Confusion matrix for test set. The squares at the top indicate the number of data represented absence that were predicted as absence (labelled with 0), the true negatives, and those that were predicted as presence (labelled with 1), the false positives. The squares at the bottom indicate the number of data represented presence that were predicted as absence, the false negatives, and those that were predicted as presence, the true positives.

After training, the model was converted using the TFLite library. Specifically, quantization is applied to the model's weights, converting the values from floating point to integers, thus reducing model memory consumption.

```
converter = tf.lite.TFLiteConverter.from_saved_model(export_dir)
converter.optimizations = [tf.lite.Optimize.DEFAULT]
tflite_model = converter.convert()
```

The `DEFAULT` option refers to a set of predefined optimizations, including the quantization of weights from `float` to `int8`. This results in a reduction in the size of the model: from a memory occupancy of 136360 B for the original model to only 28936 B without any loss of accuracy in the model. This claim was verified by running inference on the same test set after conversion, showing that the quantized model successfully retains its predictive ability.

A simple method was tried for the purpose of classification before using the machine learning model. This method calculated the peak value for each data in the dataset and compared this maximum with a threshold (maximum peak value among all data representing the "non-presence" in the dataset). If the calculated peak exceeded the threshold, the data was classified as "presence," otherwise "non-presence". However, this method achieved a lower accuracy than the adopted ML model, with an accuracy value obtained on the test set of 0.7609, highlighting how such a simple approach is unable to understand the complexity of the data.

6.2. Results of Distance Estimation

The dataset used for *distance estimation algorithm*, which includes 1725 data points, is divided into the three sets as follows: 1219 for training, 253 for validation, and 253 for test.

Table 5 shows the mae obtained by the original model and the mae obtained by the quantized model after the

convertion in a tflite model.

train mae	val mae	test mae	quantized mae
0.1243	0.1199	0.1451	0.8769

Table 5: CNN regression results: mae values.

This model underwent identical quantization as the CNN model employed for presence detection. Subsequent to quantization, the memory footprint was substantially reduced from 89800 B to 16992 B. Nevertheless, this led to a degradation in the model's accuracy, as seen in Table 5 showing the increase in mae on the test set. Given the still reasonable amount of memory required by the non-quantized model, and the large difference in the MAE metric, we oped for deploying the non-quantized model.

The loss function for this neural network was 'mean_squared_error' instead of 'binary_crossentropy' because it was utilized for a regression problem rather than a binary classification. It as well was run for 400 epochs, with an early stop after 15 epochs. The values of the loss function computed are presented in the table below:

train loss	val loss	test loss
0.0370	0.0326	0.0549

Table 6: CNN regression results: loss values.

A simple approach was also tried for the distance estimation problem before using the ML algorithm. This method aimed to detect the bin where the data had the peak value and, based on the peak bin, to estimate the discrete position of the subject (near, middle, far). Although the level of accuracy obtained is good (0.9368 on the test set), this approach was not able to provide a continuous distance value, but rather only provides a discrete label.

6.3. Results of Anomaly Detection

The dataset used for the OCSVM model is divided into the three sets as follow: 1920 are used for the training set, 137 for validation set and 103 for test set, which included all the anomalous data.

The model has demonstrated strong competence in identifying outlier data, confirming the effectiveness of the methodology used, with an accuracy on the test set of 98.06%. The test set includes normal and anomalous data to test the model's ability to identify as anomalous the data that deviate from those used for training.

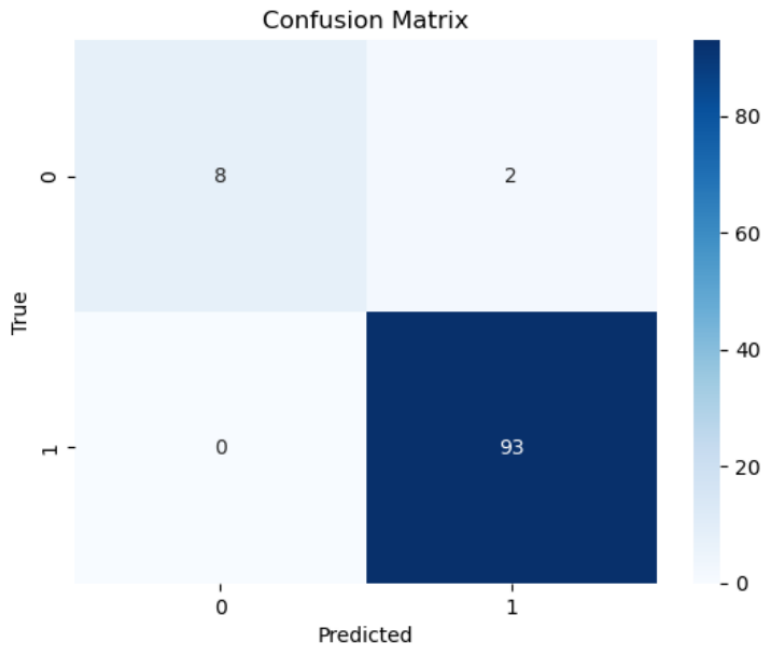


Figure 8: Confusion matrix for test set of the OCSVM model. The squares at the top indicate the number of anomalous data that were predicted as anomalous (labelled with 0), the true negatives, and those that were predicted as normal (labelled with 1), the false positives. The squares at the bottom indicate the number of normal data that were predicted as anomalous, the false negatives, and those that were predicted as normal, the true positives.

Another method tested for the purpose of anomaly detection was the use of different SVM models each aimed at identifying a specific type of anomaly that the designer knows to be common in this context. The SVM models are trained to distinguish between normal data and specific anomalous data, using anomalous data in training to create a hyperplane separating normal data from anomalous data. In particular, three SVMs were employed for this purpose:

- SVM_{double} for double appearances in two different rooms anomaly. The anomaly occurs when presence is detected simultaneously in two different rooms. This type of anomaly could indicate the presence of unauthorized people in the house in addition to the presence of the patient.
- SVM_{abs} for absence at unscheduled times anomaly. The anomaly arises when individuals are not detected within the home environment during periods that are not planned or scheduled such as at night. Such an anomaly could imply that an individual left the house at a time that was unexpected.
- SVM_{diff} for presence in a room other than the scheduled room anomaly. The anomaly occurs when a presence is detected in a room other than the one in which the individual is expected to be based on his typical behavior. An example would be the detection of presence in a room other than the bedroom during the night for a long period of time.

Each of the three SVMs is structured using the RBF kernel, but with different selections of gamma and C parameters. After testing different gamma values for the models, a value of 0.05 was chosen for the SVM_{abs} and SVM_{double} and 0.1 for the SVM_{diff}.
C parameter in SVMs is a regularization parameter, a sufficiently large value of C creates a separating hyperplane that minimizes the number of training errors while maximizing the margin for successfully categorized pattern vectors [32]. Finally, a C value of 1.0 was chosen for the SVM_{double} and SVM_{diff} and 1.5 for SVM_{abs}

Specific datasets were also created for the three SVM models. In particular, each dataset used for the SVMs

contains normal data and anomalous data representing the specific anomaly for which we are training the SVM. The dataset used for the SVM_{diff} consists of 1170 data points in total, with 508 anomalies and the remaining 662 data representing normal data, the dataset for the SVM_{abs} includes 1472 data points, of which 556 are anomalies, while the dataset for the SVM_{double} includes 1148 data, with 422 anomalies. The datasets used for SVMs are divided into the three sets as follows:

- Of the totals 1170 data in the SVM_{diff}, 700 are used for training set, 320 for validation set and 150 for test set.
- Of the totals 1472 data in the SVM_{abs}, 972 are used for training set, 350 for validation set and 150 for test set.
- Of the totals 1148 data in the SVM_{double}, 685 are used for training set, 311 for validation set and 152 for test set.

After training the SVMs to recognize specific anomalies through the training sets, and after choosing the model parameters by evaluating the performance on the validation sets, the three models were tested through the test sets for their ability to identify specific anomalies. On the same test sets, we evaluated also the OCSVM previously trained on the other dataset.

	Specific SVMs	OCSVM
Double Presence	0.9539	1.0
Unexpected Absence	1.0	0.9333
Different Room	0.9933	1.0
Average	98.24	97.78

Table 7: SVMs and OCSVM results on the three test sets.

The SVMs used for detecting specific anomalies have also shown high rates of accuracy. Such high accuracy are the result of manual data generation and the presence of multiple SVMs. By working with real world, more noisy data, we can expect the results to be slightly worse. Furthermore, each of these SVMs is tasked with addressing specific anomalies, resulting in simpler, more specialized models that contribute to higher performance in anomaly detection.

But, the choice to use the OCSVM model for the purpose of anomaly detection was driven by several reasons. First, the use of different SVMs requires a priori knowledge of anomalous behavior, thus failing to generalize should a new, nonspecific anomaly arise. Second, these models would require more attention during the training phase, which requires the presence of specific abnormal data in the dataset that should be collected on labeled by the final user. In the case of OCSVM, which requires the use of only normal data, this phase is done simply by recording the subject's behaviors in the first period of device use.

In favor of the OCSVM model, a new type of anomalous data representing presence at unscheduled times was tested on the other three SVMs, which were designed to detect other types of anomalies, before being submitted to the OCSVM. The anomaly indicates an unexpected or unscheduled presence in a certain area of the home, which could be caused by an unauthorized intrusion or by the patient's own presence at a time when the absence was scheduled that could indicate an abnormal patient situation such as a fall. This procedure showed the inability of the SVM to generalize on other type of data, due to their specific design to detect specific types of anomalies. In contrast, the OCSVM showed superior capabilities in detecting the presence of unusual anomalies, with significantly better results than the other three SVMs. Table 8 shows the percentages of outlier data denoting unexpected presence recognized by the SVM models.

Double Presence	Unexpected Absence	Different Room	OCSVM
0.6471	0.0	0.9216	1.0

Table 8: Comparison of SVMs on detecting a new type of anomaly.

7. Conclusion

In conclusion, we have seen how Tinyml combined with UWB technology leads to an excellent aftermath in the field such as presence detection and indoor localization, enabling accurate and noninvasive monitoring, and how this combination is therefore suitable in the world of healthcare and indoor monitoring.
Of particular importance to this thesis is the use of OCSVM for anomaly detection. It plays a key role in detecting abnormal behavior patterns. Its ability to distinguish normal behavior from potential risks greatly increases the effectiveness of our system in keeping the elderly safe.
Despite some limitations, such as the test environment for data collection being limited in size or the manual generation of data for the anomaly detection algorithm, the effectiveness of this technology has been demonstrated.

The research lays a solid foundation for future developments. First, the use of a more suitable data collection environment for conducting the experiment, or the possibility of acquiring data from real behavioral models learned from actual subjects, could provide a more authentic and accurate assessment of the performance of UWB and TinyML technologies.
In addition, although the study conducted focuses on the detection of anomalies, currently an alert system that notifies such anomalies has not yet been developed. Therefore, another future step could be dedicated to the development of an effective alert system that is capable of timely notification of detected anomalies.

Finally, we developed a distributed solution that can combine the use of UWB and TinyML for patient monitoring and an anomaly detection algorithm to detect deviations in patient behavior. In the future, we expect that as these technologies advance, these solutions will be increasingly used in the elderly care context to meet the growing demand of an aging population.
The success of TinyML and UWB technology bode well for future technological advancements especially within the healthcare and indoor monitoring sectors. Future research needs to tackle the limitations that have been identified. Continued advancements can be achieved through dedicated research and development initiatives.

References

[1] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6, 2017.

[2] Julien Audibert, Pietro Michiardi, Frédéric Guyard, Sébastien Marti, and Maria A. Zuluaga. Usad: Unsupervised anomaly detection on multivariate time series. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, page 3395–3404, New York, NY, USA, 2020. Association for Computing Machinery.

[3] Aritz Bilbao-Jayo, Xabier Cantero, Aitor Almeida, Luca Fasano, Teodoro Montanaro, Ilaria Sergi, and Luigi Patrono. Location based indoor and outdoor lightweight activity recognition system. *Electronics*, 11(3), 2022.

[4] Rahul Chauhan, Kamal Kumar Ghanshala, and R.C Joshi. Convolutional neural network (cnn) for image detection and recognition. In *2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC)*, pages 278–282, 2018.

[5] Chia-chin Chong, Fujio Watanabe, and Hiroshi Inamura. Potential of uwb technology for the next generation wireless communications. In *2006 IEEE Ninth International Symposium on Spread Spectrum Techniques and Applications*, pages 422–429, 2006.

[6] Davide Dardari, Chia-Chin Chong, and Moe Win. Threshold-based time-of-arrival estimators in uwb dense multipath channels. *IEEE Transactions on Communications*, 56(8):1366–1378, 2008.

[7] Robert David, Jared Duke, Advait Jain, Vijay Janapa Reddi, Nat Jeffries, Jian Li, Nick Kreeger, Ian Nappier, Meghna Natraj, Shlomi Regev, Rocky Rhodes, Tiezhen Wang, and Pete Warden. Tensorflow lite micro: Embedded machine learning on tinyml systems. *CoRR*, abs/2010.08678, 2020.

[8] Dr. Lachit Dutta and Swapna Bharali. Tinyml meets iot: A comprehensive survey. *Internet of Things*, 16:100461, 2021.

[9] Theodoros Evgeniou and Massimiliano Pontil. Support vector machines: Theory and applications. volume 2049, pages 249–257, 09 2001.

[10] Zahra Ghafoori, Sutharshan Rajasegarar, Sarah M. Erfani, Shanika Karunasekera, and Christopher A. Leckie. Unsupervised parameter estimation for one-class support vector machines. In James Bailey, Latifur Khan, Takashi Washio, Gill Dobbie, Joshua Zhexue Huang, and Ruili Wang, editors, *Advances in Knowledge Discovery and Data Mining*, pages 183–195, Cham, 2016. Springer International Publishing.

[11] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, 2016.

[12] Ramaswamy Hariharan and Kentaro Toyama. Project lachesis: Parsing and modeling location histories. In Max J. Egenhofer, Christian Freksa, and Harvey J. Miller, editors, *Geographic Information Science*, pages 106–124, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[13] Salma Jamal, Sukriti Goyal, Abhinav Grover, and Asheesh Shanker. *Machine Learning: What, Why, and How?*, pages 359–374. Springer Singapore, Singapore, 2018.

[14] Christian Janiesch, Patrick Zschech, and Kai Heinrich. Machine learning and deep learning. *Electronic Markets*, 31(3):685–695, Sep 2021.

[15] Manjunath Jogin, Mohana, M S Madhulika, G D Divya, R K Meghana, and S Apoorva. Feature extraction using convolution neural networks (cnn) and deep learning. In *2018 3rd IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT)*, pages 2319–2323, 2018.

[16] Shehroz S. Khan and Michael G. Madden. One-class classification: taxonomy of study and review of techniques. *The Knowledge Engineering Review*, 29(3):345–374, 2014.

[17] Soo-Cheol Kim, Young-Sik Jeong, and Sang-Oh Park. Rfid-based indoor location tracking to ensure the safety of the elderly in smart home environments. *Personal and Ubiquitous Computing*, 17(8):1699–1707, Dec 2013.

[18] Google. Machine Learning. Classification: Accuracy. `https://developers.google.com/machine-learning/crash-course/classification/accuracy`.

[19] Samuel G. Leitch, Qasim Zeeshan Ahmed, Waqas Bin Abbas, Maryam Hafeez, Pavlos I. Laziridis, Pradorn Sureephong, and Temitope Alade. On indoor localization using wifi, ble, uwb, and imu technologies. *Sensors*, 23(20), 2023.

[20] Qing Li, Weidong Cai, Xiaogang Wang, Yun Zhou, David Dagan Feng, and Mei Chen. Medical image classification with convolutional neural network. In *2014 13th International Conference on Control Automation Robotics Vision (ICARCV)*, pages 844–848, 2014.

[21] Tailin Liang, John Glossner, Lei Wang, Shaobo Shi, and Xiaotong Zhang. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing*, 461:370–403, 2021.

[22] Ji Lin, Ligeng Zhu, Wei-Ming Chen, Wei-Chen Wang, and Song Han. Tiny machine learning: Progress and futures [feature]. *IEEE Circuits and Systems Magazine*, 23(3):8–34, 2023.

[23] Batta Mahesh. Machine learning algorithms -a review, 01 2019.

[24] Armando Caltabiano Massimo Pavan and Manuel Roveri. On-device subject recognition in uwb-radar data with tiny machine learning. In *CPSW'2022: Fourth Cyber-Physical Systems Summer School Workshop*, 2022.

[25] Medium. Mean absolute error. `https://medium.com/@20__80__/mean-absolute-error-mae-machine-learning-ml-b9b4afc63077`.

[26] Medium. Regularization — understanding l1 and l2 regularization for deep learning. `https://medium.com/analytics-vidhya/regularization-understanding-l1-and-l2-regularization-for-deep-learning-a7b9e4a409bf`.

[27] Ghassem Mokhtari, Qing Zhang, Chad Hargrave, and Jonathon C. Ralston. Non-wearable uwb sensor for human identification in smart home. *IEEE Sensors Journal*, 17(11):3332–3340, 2017.

[28] Stefania Monica and Federico Bergenti. A comparison of accurate indoor localization of static targets via wifi and uwb ranging. In Fernando de la Prieta, María J. Escalona, Rafael Corchuelo, Philippe Mathieu, Zita Vale, Andrew T. Campbell, Silvia Rossi, Emmanuel Adam, María D. Jiménez-López, Elena M. Navarro, and María N. Moreno, editors, *Trends in Practical Applications of Scalable Multi-Agent Systems, the PAAMS Collection*, pages 111–123, Cham, 2016. Springer International Publishing.

[29] Bo Pang, Erik Nijkamp, and Ying Nian Wu. Deep learning with tensorflow: A review. *Journal of Educational and Behavioral Statistics*, 45(2):227–248, 2020.

[30] Massimo Pavan, Armando Caltabiano, and Manuel Roveri. Tinyml for uwb-radar based presence detection. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2022.

[31] Partha Pratim Ray. A review on tinyml: State-of-the-art and prospects. *Journal of King Saud University - Computer and Information Sciences*, 34(4):1595–1623, 2022.

[32] DM Reeves and GM Jacyna. Support vector machine regularization. *Wiley Interdisciplinary Reviews: Computational Statistics*, 3(3):204–215, 2011.

[33] Siddharth Sharma, Simone Sharma, and Anidhya Athaiya. Activation functions in neural networks. *International Journal of Engineering Applied Sciences and Technology (IJEAST)*, 4(12):310–316, 2020.

[34] Pramila P. Shinde and Seema Shah. A review of machine learning and deep learning applications. In *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, pages 1–6, 2018.

[35] Dalwinder Singh and Birmohan Singh. Investigating the impact of data normalization on classification performance. *Applied Soft Computing*, 97:105524, 2020.

[36] K. Siwiak. Ultra-wide band radio: introducing a new technology. In *IEEE VTS 53rd Vehicular Technology Conference, Spring 2001. Proceedings (Cat. No.01CH37202)*, volume 2, pages 1088–1093, 2001.

[37] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, jan 2014.

[38] Matthew Stewart, Pete Warden, Yasmine Omri, Shvetank Prakash, Joao Santos, Shawn Hymel, Benjamin Brown, Jim MacArthur, Nat Jeffries, Sachin Katti, Brian Plancher, and Vijay Janapa Reddi. Datasheets for machine learning sensors: Towards transparency, auditability, and responsibility for intelligent sensing, 2024.

[39] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S. Emer. Efficient processing of deep neural networks: A tutorial and survey. *CoRR*, abs/1703.09039, 2017.

[40] Paulo Sá, Rafael Bessa Loureiro, Fernanda Lisboa, Rodrigo Peixoto, Lian Nascimento, Yasmin Bonfim, Gustavo Cruz, Thauan Ramos, Carlos Montes, Tiago Pagano, Oberdan Pinheiro, and Rafael Borges. Efficient deployment of machine learning models on microcontrollers: A comparative study of quantization and pruning strategies. pages 181–188, 10 2023.

[41] Eijiro Takeuchi, Alberto Elfes, and Jonathan Roberts. *Localization and Place Recognition Using an Ultra-Wide Band (UWB) Radar*, pages 275–288. Springer International Publishing, Cham, 2015.

[42] Ange Tato and Roger Nkambou. Improving adam optimizer. 2018.

[43] Truesense. Ultra wide band. `https://ultrawideband.truesense.it/`.

[44] Frederick Tung and Greg Mori. Deep neural network compression by in-parallel pruning-quantization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(3):568–579, 2020.

[45] Shen Yin, Xiangping Zhu, and Chen Jing. Fault detection based on a robust one class support vector machine. *Neurocomputing*, 145:263–268, 2014.

[46] Xue Ying. An overview of overfitting and its solutions. *Journal of Physics: Conference Series*, 1168(2):022022, feb 2019.

Abstract in lingua italiana

Nel contesto dell'assistenza agli anziani, sta emergendo la necessità di tecnologie avanzate che possano migliorare l'erogazione dell'assistenza sanitaria. Ciò è dovuto al fatto che la popolazione anziana sta aumentando e con essa le sfide legate alla salute e al benessere degli anziani. Le tecnologie di Machine Learning (ML) possono fornire un valido supporto agli operatori sanitari nell'identificare cambiamenti significativi nelle condizioni dei pazienti da loro assistiti. Allo stesso tempo, le tecnologie in questo campo, per essere utilizzate e accettate dall'utente finale, richiedono di essere non invasive, discrete e rispettose delle abitudini e dello stile di vita di ciascuno. I dispositivi indossabili e le telecamere intelligenti sono esempi di tecnologie che hanno mostrato questo tipo di rifiuto da parte della popolazione anziana: i dispositivi indossabili richiedono, infatti, di essere portati costantemente con sé dagli utenti per funzionare, mentre le telecamere sono spesso percepite come troppo invasive della privacy dell'utente finale.
In questo contesto, proponiamo un'innovativa soluzione distribuita basata su radar in banda ultralarga (UWB) che impiega due distinti algoritmi di ML che operano gerarchicamente insieme, con l'obiettivo di migliorare la privacy tenendo conto della percezione degli utenti più anziani. I due algoritmi, ovvero *algoritmo di stima della distanza e della presenza in-sensor* e *algoritmo di rilevamento delle anomalie*, sono progettati per essere eseguiti insieme su diversi dispositivi di una pipeline ML distribuita. Gli algoritmi in-sensor sono stati specificamente progettati per soddisfare i severi vincoli che caratterizzano gli ambienti TinyML. È pensato per essere eseguito su più sensori radar UWB distribuiti in tutte le stanze di una casa e, inserendo matrici radar ad alta dimensione, restituisce la presenza e la distanza di un bersaglio in una stanza. L'*algoritmo di rilevamento delle anomalie (AD)* viene eseguito su un dispositivo edge, raccogliendo l'output di tutti i sensori intelligenti con l'obiettivo di rilevare deviazioni significative dalle abitudini standard del paziente. La soluzione distribuita presentata, che comprende sensori UWB-radar intelligenti e abilitati a tinyML, rappresenta un approccio promettente per il miglioramento dell'assistenza agli anziani, offrendo una soluzione non invasiva e rispettosa della privacy che può migliorare la qualità della vita degli anziani, limitando al tempo stesso il carico di lavoro di chi li assiste.

Parole chiave: TinyML, Radar Ultrawideband, Rilevamento delle anomalie, Localizzazione indoor

Acknowledgements

Desidero esprimere la mia sincera gratitudine al Prof. Manuel Roveri per il suo prezioso sostegno e la sua guida durante lo sviluppo di questa tesi. Desidero inoltre ringraziare Massimo Pavan il cui contributo e suggerimenti hanno arricchito notevolmente il contenuto di questa tesi.

Vorrei ringraziare la mia famiglia, per il loro sostegno incondizionato lungo tutto il percorso. Grazie di cuore per avermi sempre incoraggiato a perseguire i miei interessi e per avermi lasciato la libertà di seguire le mie passioni. La vostra fiducia in me è stata una fonte costante di ispirazione e mi ha dato la forza necessaria per raggiungere questo traguardo.

Grazie a mia sorella Valentina, per essere la donna che ammiro. Grazie per la tua saggezza e la tua determinazione. Il tuo sostegno ha reso il mio percorso più significativo. Grazie per quello che fai.

Grazie alle mie amiche per essere ancora una volta qui al mio fianco. Crescere insieme a voi è stata la mia più grande fortuna e condividere questo traguardo con voi è la cosa che più desidero. La vostra presenza rende questo momento ancora più speciale.

Grazie alle mie coinquiline, siete diventate amiche con cui condividere gioie, preoccupazioni e momenti indimenticabili. Siete state le persone che più intensamente hanno vissuto questo periodo con me, e in ogni situazione avete saputo tirarmi su il morale con la vostra presenza. Non potrei essere più grata per tutto ciò che abbiamo condiviso, le serate insieme, le risate, la complicità.

Grazie a tutte le persone che con il loro supporto hanno contribuito al raggiungimento di questo traguardo. Vi voglio bene.

Alessandra