UNIVERSITAT JAUME • I

# Machine learning-based techniques for indoor localization and human activity recognition through wearable devices.

Ph.D. Thesis

**Emilio Sansano Sansano**

**Supervisors:** Dr. Raúl Montoliu Colás (Universitat Jaume I)
Dr. Óscar Belmonte Fernández (Universitat Jaume I)

Dissertation submitted to the Doctoral School in partial fulfillment of the requirements for the Degree of Doctor by the University Jaume I

**Castelló de la Plana (Spain)**
**November 2020**

UNIVERSITAT JAUME•I

Doctoral Programme in Computer Science

**Escola de Doctorat de la Universitat Jaume I**

# Machine learning-based techniques for indoor localization and human activity recognition through wearable devices.

Dissertation submitted by Emilio Sansano Sansano in fulfillment of the requirements for the Degree of Doctor by the University Jaume I

| Author | Supervisor | Supervisor |
|---|---|---|
| Emilio Sansano Sansano | Raúl Montoliu Colás | Óscar Belmonte Fernández |

**Castelló de la Plana (Spain)**

**November 2020**

This work has been partially funded by:

- Spanish Ministry of Economy and Competitiveness: "Proyectos I + D Excelencia" programme (TIN2015-70202-P)
- Spanish Ministry of Economy and Competitiveness: "Proyectos I + D Retos investigación" programme (RTI2018-095168-B-C53)
- Spanish Ministry of Economy and Competitiveness: "Redes de Excelencia" (TEC2015-71426-REDT)
- Jaume I University: "Research promotion plan 2017" programme (UJI-B2017-45)

*My family*

# Publications

The present work is structured as a compendium of scientific publications that have been published as papers in journals indexed by the Journal Citation Report (JCR), or as book chapters published by a renowned publisher. This section enumerates the four works that make up this thesis, including authors, title, journal/editorial, D.O.I., and impact factor.

- Chapter 1:

  **Sansano-Sansano, Emilio**; Montoliu, Raúl; Belmonte-Fernández, Óscar; Torres-Sospedra, Joaquín.

  *Indoor Positioning and Fingerprinting: The R Package ipft*.

  The R Journal, 2019, vol. 11, núm. 1, p. 67-90 [142]

  doi: 10.32614/RJ-2019-010

  scopus: **Q1** (2019) 9/227 in Statistics and Probability

  jcr: **Q1** (2019) 7/124 in Statistics and Probability

- Chapter 2:

  Montoliu, Raúl; **Sansano-Sansano, Emilio**; Belmonte Fernández, Óscar; Torres-Sospedra, Joaquín.

  *IndoorLoc Platform: A Web Tool to Support the Comparison of Indoor Positioning Systems*. Geographical and Fingerprinting Data to Create Systems for Indoor Positioning and Indoor/Outdoor Navigation, Academic Press, pp 225-247, 2019. [104]

  isbn: 978-0-12-813189-3

  doi: 10.1016/B978-0-12-813189-3.00012-5

- Chapter 3:

  **Sansano-Sansano, Emilio**; Montoliu, Raúl; Belmonte Fernández, Óscar.

  *A study of deep neural networks for human activity recognition*.

  Computational Intelligence. 2020; 1– 27. [141]

  doi: 10.1111/coin.12318

  scopus: **Q2** (2019) 63/146 in Computational Mathematics

  jcr: **Q4** (2019) 111/136 in Computer Science, Artificial Intelligence

- Chapter 4:

  **Sansano-Sansano, Emilio**; Belmonte Fernández, Óscar; Montoliu, Raúl; Gascó-Compte, Arturo; Caballer.Miedes, Antonio.

  *Multimodal sensor data integration for indoor positioning in Ambient Assisted Living environments.*.

  Mobile Information Systems, 2020, vol. 2020, article id: 5204158; [140]

  doi: 10.1155/2020/5204158

  scopus: **Q2** (2019) 86/307 in Computer Networks and Communications

  jcr: **Q4** (2019) 118/156 in Computer Science, Information Systems

This thesis has been accepted by the co-authors of the publications listed above that have waved the right to present them as a part of another PhD thesis.

# Acknowledgments

First and foremost, I would like to express my most profound gratitude to my supervisors Raúl and Óscar, for always giving me useful suggestions, strong support, and inspiring talks during my Ph.D. studies. I'll always feel lucky to have had you as my directors.

I can not forget all the people that have been part of the GIANT group during these years. Looking back to the first year in the group, I am surprised about how much I have learned from all of you and the long way I have come through.

Last, but not at least, I am indebted to Henar and Aitana for supporting me and sharing my worries, frustrations, and happiness. This thesis certainly would not have been possible without the love, support, and understanding of my family.

# Introduction

## Motivation

Indoor positioning is perceived as one of the upcoming major applications which can be used in a wide variety of crucial location-based services, such as indoor navigation in airports, hospitals or malls, tracking of goods in warehouses, or assisted living systems for elderly care. While for outdoor environments Global Satellite Navigation Systems (GNSS) have become de facto standard, there is no equivalent system for indoor scenarios. Indoor positioning has been an object of research for the last years but there is currently no agreed solution that can compare to the performance of GNSS in outdoor environments.

Tracking technologies have several applications in contexts such as health care, and research on this technology is on the increase. These technologies often utilize wearable devices such as smartphones or smart-watches. However, the cost and effort needed to develop and deploy a reliable positioning system may become arduous for research purposes. Hence, effective, easy to use, and low-cost solutions are needed.

This thesis approaches the study of several machine learning techniques to improve the performance of indoor positioning systems, with a special focus on wearable and low-cost devices. It also presents some tools designed to facilitate the research in this field by means of the development of an open-source software framework for indoor positioning-related research, and the creation of a web platform committed to becoming a collaborative repository of data.

## Background

Indoor Positioning Systems (IPS) use radio waves, magnetic field, acoustic signals, images, or other information collected by sensors to locate and track people or objects inside buildings. In particular, RSSI (Received Signal Strength Indicator) positioning systems are based on measuring the intensities of the received radio signals of the emitting devices (beacons) that are available at a particular position. Currently, the most widely used technology for indoor localization is WiFi, which is readily available on almost all user devices. Since WiFi is commonly found in many buildings, indoor tracking and navigation using RSSI from opportunistic WiFi access points have been a rather common choice. Likewise, visible light and BlueTooth can be used as other viable options.

The methods used to derive the position of the user/device from the RSSI values can be divided into three general categories: proximity, triangulation, and fingerprinting. The proximity-based methods use the RSSI to locate the user assuming that the received signal with the highest value is from the closer emitter. The triangulation-based methods use the RSSI values from three or more emitters to estimate the distances to each of them based on a specific signal propagation model. Finally, fingerprinting-based are based on the assumption that, for a given indoor environment, a signal mapping exists, and that such map can be reconstructed measuring the RSSI signal at discrete locations of the mapped area. In the case of Wi-Fi fingerprinting, its main advantage relies on the fact that there already is an existing Wi-Fi infrastructure in the majority of urban areas. Therefore, the location of the user can be obtained without deploying any additional equipment.

The RSSI fingerprinting localization approach requires two phases of operation: a training phase, also known as off-line or survey phase, and a positioning phase, sometimes referred to as on-line, runtime, or tracking phase. In the training phase, multi-dimensional vectors of RSSI values (the fingerprints) are generated and associated with known locations. These measurements are used to build a data set (also known as radio map) that covers the area of interest. The given area has to be divided into smaller parts and each is defined by one specific fingerprint, which may be a time-consuming task, especially in large areas. The collected data set can include, along with the collected RSSI values and the location coordinates, many other useful parameters, as the device type used in the measurements or its orientation. Later, during the positioning phase, an RSSI vector collected by a device is compared with the stored data to generate an estimation of its position. Obstacles, reflections, multi-path interference, environmental changes, or device orientation are factors that affect signal propagation and can degrade the performance of IPS based on Wi-Fi fingerprinting.

Being such a critical technology, one of the major requirements for indoor localization systems is to use a technology that is readily available on the user's device. Furthermore, a future scenario where low access barriers and open standards play an important role in granting interoperability is most desirable. This is important for the wide-scale adoption of the technology. The first chapters of this thesis are focused on presenting some works aimed in this direction. In chapters 1 and 2 this work explores ways towards enhancing collaboration among researchers in indoor positioning by contributing with open source tools and a public repository of data. Allowing researchers easy access to an open-source implementation of commonly-used techniques and algorithms in this field can accelerate the progress of building complex, useful, and significant services on top of state-of-the-art machine learning algorithms for indoor positioning. The same considerations can be made about data. Since the recollection of quality data is costly, the existence of a public repository with accurate, complete, consistent, and reliable data may be an important help to researchers in this field.

One of the most relevant location-based services that indoor positioning technologies can offer is related to healthcare. The fact that virtually every country in the world is experiencing growth in the proportion of older persons in their population will accelerate an important social transformation in the coming decades. Many key enablers for the optimization of healthcare systems depend upon the implementation of more advanced systems in the healthcare industry, such as location awareness for patients (e.g. with dementia), nurses, doctors, etc.

Due to underlying and often debilitating health conditions that are associated with elderly people, aspects of everyday living can become physically and mentally challenging for them. Technology can be integrated into the health care of senior citizens to provide safe, high-quality lives, improve their health and happiness, and enable a longer period of independent living. Assistive technical applications should be easy to use, unobtrusive, suitably designed, and adaptable to changing needs and individual preferences.

The ubiquity of smartphones and smart-watches, and the availability of different wireless interfaces, such as Wi-Fi, 3G, and Bluetooth, make them an attractive platform for indoor monitoring. Smart home-based behavioral data have already been found to be useful in assisting older adults to live independently and to monitor health state and the onset and progress of age-related diseases and disorders such as dementia and Alzheimer's disease. Psychological health in older adults (loneliness, depression, or emotional states) has been assessed using such data too. Nevertheless, the level of technology readiness for home health monitoring technologies is still too weak to provide such services with reliable performance.

Before-mentioned services can be built upon Artificial Intelligence technologies such as Machine Learning and Deep Learning. In particular, Deep Learning has achieved many advances in recent years, dramatically improving the state-of-the-art in many artificial intelligence tasks like computer vision, language processing, speech recognition, and many more. One of the essential advantages of deep learning is its ability to automatically learn features from raw data. Traditionally, proposed schemes for pattern recognition have relayed on the design of handcrafted features. Although these schemes could achieve high accuracy, the requirement for domain knowledge limits its scalability. Finding a good set of features from the raw data is critical to isolate key information and highlight important patterns, but it requires expert knowledge and it is difficult and time-consuming. Deep learning eliminates the need for manual feature engineering.

To improve the accuracy and robustness of indoor positioning algorithms, data from wearable embedded sensors, such as smartwatches or fitness bands, can be leveraged to aid in the recognition of the user's activity. Recognizing the activity users are performing can be of much help to determine their position. Chapter 3 presents an extensive analysis among the most suited deep learning architectures for activity recognition to compare their performance in terms of accuracy, speed, and memory requirements.

Finally, chapter 4 is dedicated to showing how the use of other sensors available in wearable devices can help to improve the accuracy of indoor positioning systems. In the real world, indoor positioning accuracy is limited by the quality of the signals in the environment. This is due to the fact that indoor environments constantly experience dynamic changes, causing the radio signal strengths to fluctuate over time, which weakens the signal-spatial correlations of the RF fingerprints. Many things can interfere with signals and make the data incorrect. Things like walls, human bodies, pockets, or even proximity to several emitting devices at once can throw the measurements off. Furthermore, although many IPSs have been proposed in the literature, most of these have been evaluated in non-representative environments such as office buildings rather than in an operational environment. Chapter 4 outlines the characteristics of an IPS aimed at characterizing the behavior of elder people at their homes. Due to underlying and often debilitating health conditions that are associated with elderly people, aspects of everyday living can become physically and mentally challenging for them. Technology can be integrated into the health care of senior citizens to provide safe, high-quality lives, improve their health and happiness, and enable a longer period of independent living. The system described in this chapter has been designed to be as less intrusive as possible and has been tested in real environments. The work presents the results of the experiments conducted to assess the performance of the system, as well as some techniques that take advantage of the sensors included in low-cost wearable devices to improve the positioning accuracy of the system.

## Outline

The thesis is organized as follows:

Chapter 1 presents the R package `ipft`. This software has been developed to provide both researchers and industry with an easily extensible open-source framework for indoor positioning-related analysis, experimentation, and testing. The package has been built around a collection of fundamental algorithms and tools to manipulate, cluster, transform, create models and make estimations using indoor localization fingerprinting data. All included algorithms are highly customizable and extensible, allowing users to customize them with personalized parameters and functions to adapt the working mode to their particular research interests.

Chapter 2 presents the *indoorloc* platform, a public repository for comparing and evaluating indoor positioning algorithms and data sets. The idea behind the *indoorloc* platform is to serve as a collaborative repository for indoor positioning benchmark data sets. The platform serves also as an introductory tutorial for indoor positioning newcomers. It allows users to learn how some well-known algorithms work, study the source code of those algorithms, test the methods, and even upload results of the user's methods to check the accuracy when comparing

against the results provided by other methods.

Chapter 3 presents an evaluation of the performance of several different Deep Learning architectures for the specific task of Human Activity Recognition. The study is conducted using a great variety of public data sets acquired from several sources using a diversity of devices, such as smart-phones, smart-watches, or inertial motion units, in different environments. The purpose of this work is to assess the suitability of different deep learning architectures with respect, not only to their general accuracy but also to their memory footprint and computational requirements, since these aspects are decisive when considering resource-constrained wearable devices.

Chapter 4 presents a set of experiments designed to evaluate strategies to increase the accuracy of Wi-Fi fingerprinting-based indoor positioning systems. The experimental results show an improved room-level accuracy when using strategies such as data scaling and the use of consecutive Wi-Fi scanning. The results also demonstrate that the use of sensors such as Inertial Motion Units along with the Wi-Fi fingerprints can help to significantly increase the performance of indoor positioning systems. These techniques can be used to mitigate the variability in space and time of the perceived intensity of radio-frequency signals due to environment changing conditions. The results confirm that a more robust positioning estimation can be derived when implementing such strategies.

Finally, 5 presents a general discussion on the results of the exposed works, as well as a summary with the main conclusions that can be extracted from this thesis. In light of these conclusions, it also presents an overview of possible future lines of work.

**Keywords**: *Indoor positioning*, *Machine learning*, *Deep learning*, *Ambient assisted living*, *e-health*, *Big data*.

# Index

# Index of tables

# Index of figures

# Chapter 1

# Indoor Positioning and Fingerprinting: The R Package ipft

Methods based on the Received Signal Strength Indicator (RSSI) fingerprinting are at the forefront among several techniques being proposed for indoor positioning. This paper introduces the R package `ipft`, which provides algorithms and utility functions for indoor positioning using fingerprinting techniques. These functions are designed for manipulation of RSSI fingerprint data sets, estimation of positions, comparison of the performance of different positioning models, and graphical visualization of data. Well-know machine learning algorithms are implemented in this package to perform analysis and estimations over RSSI data sets. The paper provides a description of these algorithms and functions, as well as examples of its use with real data. The `ipft` package provides a base that we hope to grow into a comprehensive library of fingerprinting-based indoor positioning methodologies.

## 1.1 Introduction

Intelligent spaces, as a particularity of the concept known as Ambient Intelligence (AmI) [1, 165], where agents communicate and use technology in a non-intrusive way, have an interest in both open and closed environments. Since people spend 90% of time indoors [76], one of the most relevant aspects of AmI is indoor localization, due to the large number of potential applications: industrial and hospital applications, passenger transport, residences, assistance to emergency services and rescue, localization and support guide for the disabled, leisure applications, etc. It is expected that the global market for this type of location will grow from USD 7.11 billion in 2017 to USD 40.99 billion by 2022 [129], being among the key technologies in the future. This is a technology that has already awakened but that in a short period of time will suffer a big explosion, as happened with the systems of positioning by satellite in exteriors and its applications.

This paper introduces the R package `ipft` [137], a collection of algorithms and utility functions to create models, make estimations, analyze and manipulate RSSI fingerprint data sets for indoor positioning. Given the abundance of potential applications for indoor positioning, the package may have broad relevance in fields such as pervasive computing, the Internet of Things (IoT), or healthcare, among many others.

The main progress in indoor location systems has been made during the last years. Therefore, both the research and commercial products in this area are new, and researchers and industry are currently involved in the investigation, development, and improvement of these systems. We believe that the R language is a good environment for machine learning and data analysis related research, as its popularity is constantly growing [1], researchers related to indoor positioning have explicitly selected R as developing framework for their experiments [121, 62, 118], it is well maintained by an active community, and provides an ecosystem of good-quality packages that leverage its potential to become a standard programming platform for researchers. There are some open source applications and frameworks to build indoor positioning services, such as FIND [2], Anyplace [3] or RedPIN [4], based on fingerprinting techniques but, as far as we know, there is not any public framework or package that provides functions and algorithms to manipulate fingerprinting data sets and experiment with positioning algorithms.

RSSI (Received Signal Strength Indicator) positioning systems are based on measuring the intensities of the received radio signals of the emitting devices (beacons) that are available at a particular position, and comparing them with a previously built RSSI data set [81]. RSSI is used to measure the relative quality of a received signal to a client device, and each chipset manufacturer is free to define their scale for this term. The value read by a device is given on a logarithmic

[1] `https://stackoverflow.blog/2017/10/10/impressive-growth-r/`

[2] `https://www.internalpositioning.com#about`

[3] `https://anyplace.cs.ucy.ac.cy`

[4] `http://redpin.org`

scale and can correspond to an instant reading or a mean of some consecutive readings.

In this scenario, a fingerprint is an RSSI feature vector composed of received signal values from different emitting devices or beacons, associated to a precise position. In the last years, this technique is becoming increasingly important for indoor localization [87, 66], since Wi-Fi is generally available in indoor environments where GPS signals cannot penetrate, and the wireless access points (WAPs) can be used as emitting devices [85]. Other types of indoor localization RF emitters, such as Bluetooth [162], RFID [89], or Ultra Wide Band (UWB) [50], can be also used in combination with Wi-Fi access points or as a standalone positioning system.

The RSSI fingerprinting localization approach requires two phases of operation: a training phase, also known as off-line or survey phase, and a positioning phase, sometimes referred to as on-line, runtime, or tracking phase. In the training phase, multi-dimensional vectors of RSSI values (the fingerprints) are generated and associated with known locations. These measurements are used to build a data set (also known as radio map) that covers the area of interest. This data set can include, along with the collected RSSI values and the location coordinates, many other useful parameters, as the device type used in the measurements or its orientation. Later, during the positioning phase, an RSSI vector collected by a device is compared with the stored data to generate an estimation of its position (Figure 1.1).

Despite the increasing interest in RSSI positioning [170], this topic has not been explicitly covered yet by any publicly available R package. The proposed package has been developed to provide users with a collection of fundamental algorithms and tools to manipulate RSSI radio maps and perform fingerprinting analysis. While fundamental algorithms and similarity measurement functions are implemented to provide the main framework for research and comparison purposes, these are highly customizable, to allow researchers to tailor those methods with their own parameters and functions.

This paper describes these algorithms and their implementation and provides examples of how to use them. The remainder of the paper is structured as follows: Section 1.2 defines the fingerprinting problem statement and the nomenclature that will be used in the rest of the paper. An overview of the implemented algorithms is given in Section 1.3. Section 1.4 outlines some data wrangling techniques included

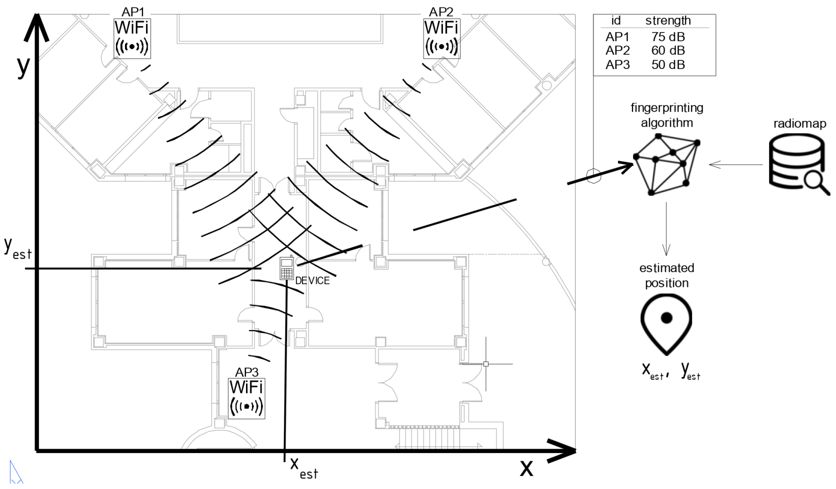


Figure 1.1: During the on-line phase, once the radio map has been built, the fingerprinting algorithm uses it to estimate the device's position by comparing the RSSI values heard by the device with the ones stored in the radio map.

in the package. Section 1.5 describes the implemented positioning algorithms. Section 1.6 presents the included methods for access point position estimation. Then, Section 1.7 discuses some tools and functions included to create clusters or groups of fingerprints. Section 1.8 illustrates the use of the plotting functions also included in the package. In all these sections, functions are described and explored using practical examples, and particular emphasis is placed on how to use them with real-world examples and data sets. Finally, the paper is summarized in Section 1.9.

## 1.2 Problem statement. Terminology and notation

This section provides a brief and general introduction to the principles of fingerprinting positioning, as well as a description of the notation and terminology that will be used in the next sections. The terms described here are related to general concepts of fingerprinting techniques, while the remaining of the paper describes the particular implementation of these concepts in the `ipft` package.

The main goal of the indoor localization techniques is to determine the position of a user in an indoor environment, where the GPS signal might not be received. This objective might require the use of existing infrastructure, the deployment of a new one, the use of the so-called signals-of-opportunity [172], or even a combination of some of these techniques. Many of these techniques take advantage of the radio-frequency signals emitted by devices, whose position can be known or not, to estimate the user's position from the perceived strength of these signals. There are many kinds of devices that can be used for this purpose, such as Wi-Fi access points, Bluetooth beacons, RFID or UWB devices, but for all of them, the information provided for a given position, the fingerprint, can be stored as a vector of received signal strength intensities (RSSI), whose length is determined by the number of detected emitters.

A radio map, or a fingerprinting data set, is composed of a set of collected fingerprints and the associated positions where the measurements were taken and may contain some additional variables, such as the type of device used or a timestamp of the observation, among any other useful data. Let $\mathcal{D}$ be a fingerprinting data set. Then:

$$\mathcal{D} = \{\mathcal{F}, \mathcal{L}\}$$

where $\mathcal{F}$ is the set of collected fingerprints and $\mathcal{L}$ is the set of associated locations.

For research purposes, a fingerprinting data set is usually divided into training and test sets. The training data set is used to store the fingerprints and location data to create models of the environment that can be used to estimate the position of a new fingerprint. The test data set is used to test the models obtained from the training data, and to compute the errors from the results of the position estimation.

Let $\mathcal{D}_{train}$ be a training data set:

$$\mathcal{D}_{train} = \{\mathcal{F}_{train}, \mathcal{L}_{train}\}$$

where

$$\mathcal{F}_{train} = \left\{\lambda_1^{tr}, \lambda_2^{tr}, ..., \lambda_n^{tr}\right\}$$

$$\mathcal{L}_{train} = \left\{\tau_1^{tr}, \tau_2^{tr}, ..., \tau_n^{tr}\right\}$$

$\mathcal{D}_{train}$ is composed of $n$ fingerprints, stored as $n$ vectors of RSSI measurements ($\lambda_i^{tr},\ i \in [1, 2, ..., n]$), and $n$ locations ($\tau_i^{tr},\ i \in [1, 2, ..., n]$), stored as vectors, representing the position associated with its correspondent fingerprint. Each fingerprint consists of $q$ RSSI values ($\rho^{tr}_{h,i}$, $h \in [1, ..., q]$), where $q$ is the number of beacons considered when building the training set:

$$\lambda^{tr}_i = \left\{\rho^{tr}_{1,i}, \rho^{tr}_{2,i}, ..., \rho^{tr}_{q,i}\right\},\ i \in [1, ..., n]$$

and each associated position is composed of one or more values, depending on the number of dimensions to be considered and the coordinate system used. The position can be given as a vector of values representing its coordinates, although on multi-floor and multi-building environments labels can be used to represent buildings, floors, offices, etc. Let $l$ be the number of dimensions of a position vector. Then:

$$\tau^{tr}_i = \left\{\nu^{tr}_{1,i}, \nu^{tr}_{2,i}, ..., \nu^{tr}_{l,i}\right\},\ i \in [1, ..., n]$$

The test data set is also composed of a collection of fingerprints associated with known positions. This data set is used for testing purposes, during research or during model building adjustments, to assess the model's performance by comparing its estimation of the positions with the ground truth.

The situation is different in real applications, where the goal is to estimate the unknown position of the receiver given the RSSI values detected at a particular location, using a previously built model. In this case, the test data set is just composed of a unique fingerprint, and the objective is to estimate the actual location of the receiver. Therefore, no information about its location is provided.

The test data set is composed of $m$ observations:

$$\mathcal{D}_{test} = \{\mathcal{F}_{test}, \mathcal{L}_{test}\}$$

where

$$\mathcal{F}_{test} = \left\{\lambda^{ts}_1, \lambda^{ts}_2, ..., \lambda^{ts}_m\right\}$$

$$\mathcal{L}_{test} = \left\{\tau^{ts}_1, \tau^{ts}_2, ..., \tau^{ts}_m\right\}$$

To be able to compare the test observations with the training fingerprints, the number of RSSI values of its respective fingerprints has to be the same, and the position in the RSSI vector must represent the same beacon in both data sets.

Therefore, each one of the $m$ observations of the test data set is composed of a fingerprint with $q$ RSSI values:

$$\lambda_j^{ts} = \left\{\rho_{1,j}^{ts}, \rho_{2,j}^{ts}, ..., \rho_{q,j}^{ts}\right\}, \; j \in [1, ..., m]$$

and a location vector with the same spatial dimensions as the training location vectors:

$$\tau_j^{ts} = \left\{\nu_{1,j}^{ts}, \nu_{2,j}^{ts}, ..., \nu_{l,j}^{ts}\right\}, \; j \in [1, ..., m]$$

The notation depicted above will be used in the remaining of the paper to represent the fingerprinting data. Symbols $i$ and $j$ will be used to represent iterations over the training and test data sets, respectively, while $h$ will be used to iterate over the beacons present in each fingerprint.

## 1.3 An overview of the implemented algorithms

This section presents an introduction to the main functions, included in the `ipft` package, that implement fingerprinting-based indoor localization methods. The package also provides two data sets for training and validation purposes that are briefly described in this section. The package is available at textttCRAN and can be installed like any other R package:

```
    install.packages("ipft")
```

The package has to be loaded into the main environment to use it for the first time in an R session:

```
    library("ipft")
```

The `ipft` package implements three algorithms to build models to estimate the position of a receiver in an indoor environment. Two of these implementations are based on the well known k-Nearest Neighbors algorithm (*knn*) [37] to, given an RSSI vector, select the $k$ most similar training examples from the radio map. The similarity between the RSSI value vectors can be measured, for example, as the $euclidean$ distance between them, but other distance functions may be used

[157]. The selection of a method to compute this measure can be provided to the function in two ways, either choosing one of the already implemented distance measurements ($euclidean$, $manhattan$, etc.) or by way of a reference to a function implemented by the user that returns the distance (the lower, the more similar or 'closer') between two matrices or vectors. Once the $k$ neighbors are selected, the location of the user is estimated as the weighted average of the neighbors' positions.

The first implementation, corresponding to the function `ipfKnn`, may behave in a deterministic way, finding the $k$ more similar neighbors using a deterministic similarity function such as the euclidean or manhattan distances, or in a probabilistic way, using similarity functions such as LDG (Logarithmic Gaussian Distance) or PLGD (Penalized Logarithmic Gaussian Distance) [39], that are based upon statistical assumptions on the RSSI measurement error. The similarity function can be chosen from the set of implemented options or provided by the user via a custom function. This implementation is discussed in Section 1.5.1.

The other implementation of the knn algorithm assumes a probabilistic nature for the received signal distribution [133] and uses collections of many fingerprints at each particular position, acquired during the training phase. Therefore, the radio map is composed of several groups, where a group is a set of fingerprints (vectors of RSSI values) that share the same location. Assuming that the RSSI value for a specific beacon can be modeled as a random variable following a normal distribution [58], any of these collections, or groups, of fingerprints can be represented by the statistical parameters of this distribution, in this case, the mean and the standard deviation. This implies that the original data set can be transformed into a new type of data structure by storing the mean and the standard deviation of every detected beacon for every group. All the original data for a group is transformed into two vectors, one storing the means and the other the standard deviations. The trustworthiness of the data in the new data set will depend on the number of measurements for every location of the original data. It is assumed that the more measurements for a particular location, the more reliable will be their inferred statistical parameters.

The implementation of this probabilistic-based method takes the original radio map and a set of group indices and fits these groups of measurements to a normal (Gaussian) distribution for every beacon and every location so that the signal intensity distribution is determined by the mean and the standard deviation of the

Gaussian fit. Then, given a test fingerprint, the algorithm estimates its position by selecting the *k* most probable locations, making explicit use of the statistical parameters of the data stored in the radio map to optimize the probabilities in the assignment of the estimated position by computing a similarity function based on a summatory of probabilities. This approach is implemented through the `ipfProbabilistic` function and is described in the Section 1.5.2.

Finally, the third implemented algorithm is based on a scenario where the location of the beacons is known, and an estimation of the fingerprint position can be made using the log-distance path loss model [144]. The strength of the received signal at a particular point can be modeled as a function of the logarithmic distance between the receiver and the emitter and some parameters related to the environment properties and the devices' characteristics. Therefore, as this method uses an analytical model to evaluate the position, no radio map is needed to train a model to compare fingerprints with, since the position might be estimated from the fingerprint data and the position of the beacons. This method is implemented by the function `ipfProximity` and is described in Section 1.5.3.

The previous functions `ipfKnn`, `ipfProbabilistic` and `ipfProximity` create models based on the training data and parameters provided. These models can then be evaluated using the `ipfEstimate` function, which internally detects the algorithm to apply based on the model that receives as a parameter.

The package also includes data from the IPIN2016[5] Tutorial data set. In the `ipftrain` data frame there are $n$ = 927 observations, including the RSSI values for $q$ = 168 wireless access points, the location, expressed in Cartesian coordinates, for the observation (x, y), and some other variables, such as timestamps for the measurements or an identifier for the user who took the survey. The `ipftest` data frame contains $m$ = 702 observations with the same structure, for testing and validation purposes. The fingerprints included in both data sets were taken in the same building and the same floor. The `ipfpwap` data frame contains the position of 39 of the WAPs included in the `ipftrain` and `ipftest` data sets. The unknown positions of the remaining WAPs are stored as `NA`. The characteristics of these data sets attributes are:

- `RSSI values`: Columns from 1 to 168. The values represent the strength of the received signal expressed in decibels, on a scale that ranges from

[5]http://www3.uah.es/ipin2016/

$-30$dBm to $-97$dBm in the training set, and from $-31$dBm to $-99$dBm in the test set. The closer the value to zero, the stronger the signal.

- `position`: Columns 169 (X) and 170 (Y). The position given in Cartesian coordinates, with its origin in the same corridor where the data was acquired.
- `user id`: A numeric value from 1 to 8 to represent each of the 8 users that acquired the train data set. The test dataset was acquired by a different user, represented by the value 0.
- `timestamp`: The UNIX timestamp of the observation, in seconds.

There are some other publicly available indoor location data sets that have been used to develop and test this package and that are not included for size reasons, as the UJIIndoorLoc Data Set [156] or the Tampere University data set [39].

The theoretical foundations of the algorithms and its uses are discussed in detail in Section 1.5. A description of the functions `ipfKnn`, `ipfProximity`, `ipfProbabilistic` and `ipfEstimate` is given while presenting some simulations to show how these algorithms can be useful in practice.

## 1.4 Data wrangling

An RSSI fingerprint is a vector composed of signal strength measurements from all the emitters received by a client device at a particular point and can be measured in any unit of power. It is often expressed in decibels (dBm), or as percentage values between 1-100, and can be a negative or a positive value. Typically these values are stored as negative figures, where the strongest signals are closer to zero.

Some algorithms are sensitive to the scale of the data. For example, Artificial Neural Networks generally work better [80] with data scaled to a range between [0, 1] or [$-1$, 1], since unscaled data may slow down the learning process and the convergence of the network parameters and, in some cases, prevent the network from effectively learning the problem. Thus, the first step before the data can be fed to a positioning algorithm may involve some kind of transformation, depending on the characteristics of the original data.

The data sets included in this package represent the RSSI data from a set of wireless access points as negative integer numbers from $-99$ (weakest detected signal) to $-30$ (strongest detected signal). When the RSSI of a WAP is not available, the value used is `NA`. This convention may be inconvenient for some calculations. For example, a similarity measure between two fingerprints as the euclidean distance will only take into account those WAPs that have been detected in both observations, causing a loss of information that otherwise could be utilized.

The `ipft` package contains some functions to manipulate and wrangle raw fingerprint data. The `ipfTransform` function mutates the given fingerprint data into a new data set with a specified range for the RSSI signals. The signature of the function is:

```
ipfTransform <- function(data, outRange = c(0, 1), outNoRSSI = 0,
    inRange = NULL, inNoRSSI = 0, trans = "scale", alpha = 24)
```

where:

- `data`: The input data set with the original RSSI fingerprints.
- `outRange`: A numeric vector with two values indicating the desired range of the output data.
- `outNoRSSI`: The desired value for not detected beacons in the output data.
- `inRange`: A numeric vector with two values indicating the range of signal strength values in the input data. If this parameter is not provided, the function will infer it from the provided data.
- `inNoRSSI`: The value given to a not detected beacon in the original data.
- `trans`: The transformation to perform over the RSSI data, either 'scale' or 'exponential'.
- `alpha`: The $\alpha$ parameter for the exponential transformation.

The *scale* transformation scales the input data values to a range specified by the user. The feature scaling is performed according to Equation 1.1:

$$\rho_{h,i}^{out} = \begin{cases} a + b \cdot \rho_{h,i}^{in}, & \text{if } \rho_{h,i}^{in} \neq inNoRSSI \\ outNoRSSI, & otherwise \end{cases} \tag{1.1}$$

$$b = \frac{outMin - outMax}{inMin - inMax}$$
$$a = outMin - inMin \cdot b$$

where:

- $\rho^{out}_{h,i}$ and $\rho^{in}_{h,i}$ are the output and input RSSI values, respectively, for the $h^{th}$ beacon from the $i^{th}$ observation
- $outMax$ and $outMin$ are the maximum and minimum values, respectively, specified for the output by the `outRange` parameter.
- $inMax$ and $inMin$ are the maximum and minimum values, respectively, of the input data.
- $outNoRSSI$ and $inNoRSSI$ are the values assigned in the fingerprint to represent a not detected beacon for the output and input data, respectively, specified by the parameters `outNoRSSI` and `inNoRSSI`.

The *exponential* transformation [157] changes the data according to the next equation:

$$\rho^{out}_{h,i} = \begin{cases} \exp(\frac{pos(\rho^{in}_{h,i})}{\alpha}), & \text{if } \rho^{in}_{h,i} \neq inNoRSSI \\ outNoRSSI, & otherwise \end{cases}$$

$$pos(\rho^{in}_{h,i}) = \begin{cases} \rho^{in}_{h,i} - inMin, & \text{if } \rho^{in}_{h,i} \neq inNoRSSI \\ 0, & otherwise \end{cases}$$

where $\alpha$ is a parameter for the exponential transformation. The authors establish $\alpha$ as a case-based parameter and find that 24 is a good value for RSSI fingerprinting data, but they did not study the effects of $\alpha$ in the transformed data.

The following code scales the `ipftrain` and `ipftest` data sets RSSI data, stored in the columns 1:168, to a positive range of values, from 0 to 1, with NA representing a not detected WAP. As a not detected WAP is represented by a `NA` value in the original data, this has to be indicated to the function so it can transform these values into the desired output:

```
trainRSSI <- ipfTransform(ipftrain[, 1:168], outRange = c(0.1, 1),
    inNoRSSI = NA, outNoRSSI = NA)
testRSSI <- ipfTransform(ipftest[, 1:168], outRange = c(0.1, 1),
    inNoRSSI = NA, outNoRSSI = NA)
```

The `ipfTransform` function returns a new data set with the same structure (vector, matrix, or data frame) as the input.

## 1.5 Positioning algorithms

This section describes three positioning algorithms implemented in the `ipft` package. The examples illustrating each description are based on the data previously scaled in Section 1.4.

### 1.5.1 The `ipfKnn` function.

The `ipfKnn` and `ipfEstimate` functions implement a version of the knn algorithm to select the $k$ nearest neighbors (the $k$ more similar vectors from the training set) to a given RSSI vector. Many different distance metrics [157] can be used to compare two RSSI vectors and measure how 'near' or similar they are.

The distance metrics implemented in the package include some typical functions, as the $L^1$ norm, or manhattan distance, or the $L^2$, or euclidean distance. The $L^u$ norm between two fingerprints with indices $a$ and $b$ is defined as follows:

$$L^u = \left( \sum_{h=1}^{q} |(\rho_{h,a} - \rho_{h,b}|^u \right)^{1/u}$$

The package also implements some fingerprinting specific distance estimation functions such as LDG and PLGD. The LGD between two RSSI vectors $\lambda_i^{tr}$ and $\lambda_j^{ts}$ of longitude $q$ is given by:

$$LGD(\lambda_i^{tr}, \lambda_j^{ts}) = -\sum_{h=1}^{q} \log\ \max(G(\rho_{h,i}^{tr},\ \rho_{h,j}^{ts}),\ \epsilon)$$

where $\epsilon$ is a parameter to avoid logarithm of zero, as well as having one beacon RSSI value influence the LGD only above a certain threshold. $G(\rho_{h,i}^{tr},\ \rho_{h,j}^{ts})$ represents the Gaussian similarity between $\rho_{h,i}^{tr}$ and $\rho_{h,j}^{ts}$, defined as

$$\mathrm{G}(\rho^{tr}_{h,i}, \rho^{ts}_{h,j}) = \begin{cases} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(\rho^{tr}_{h,i} - \rho^{ts}_{h,j})^2}{2\sigma^2}\right), & \text{if } \rho^{tr}_{h,i} \neq 0 \text{ and } \rho^{ts}_{h,j} \neq 0 \\ 0, & otherwise \end{cases}$$

The $\sigma^2$ parameter represents the shadowing variance [148]. Values for $\sigma$ in the range between 4 and 10 dBm are usually good for indoor scenarios [90].

The PLGD between two RSSI vectors $\lambda^{tr}_i$ and $\lambda^{ts}_j$ of longitude $q$ is given as:

$$PLGD(\lambda^{tr}_i, \lambda^{ts}_j) = LGD(\lambda^{tr}_i, \lambda^{ts}_j) + \alpha(\phi(\lambda^{tr}_i, \lambda^{ts}_j) + \phi(\lambda^{ts}_j, \lambda^{tr}_i))$$

where $\phi(\lambda^{tr}_i, \lambda^{ts}_j)$ is a penalty function for the beacons that are visible in the $i^{th}$ training fingerprint but not in the $j^{th}$ test fingerprint, $\phi(\lambda^{ts}_j, \lambda^{tr}_i)$ is a penalty function for the beacons that are visible in the $j^{th}$ test fingerprint but not in the $i^{th}$ training fingerprint, and are defined as follows:

$$\phi(\lambda^{tr}_i, \lambda^{ts}_j) = \sum_{h=1}^{q} T_{max} - \rho^{tr}_{h,i}, \text{ for } 0 < \rho^{tr}_{h,i} \leq T_{max} \text{ and } r_i = 0)$$

$$\phi(\lambda^{ts}_j, \lambda^{tr}_i) = \sum_{h=1}^{q} T_{max} - \rho^{ts}_{h,j}, \text{ for } 0 < \rho^{ts}_{h,j} \leq T_{max} \text{ and } r_j = 0)$$

$T_{max}$ is an upper threshold for the strength of the signal, and $\alpha$ is a scaling factor.

The similarity measurement method can be chosen by means of the parameter `method`, or by providing a custom function (parameters `FUN` and `...`). The signature of the `ipfKnn` function is:

```
ipfKnn <- function(train_fgp, train_pos, k = 3, method = '
   euclidean', weights = 'distance', norm = 2, sd = 5, epsilon =
    1e-3, alpha = 1, threshold = 20, FUN = NULL, ...)
```

where:

- `train_fgp`: A data frame of $n$ rows and $q$ columns containing the fingerprint vectors of the training set.
- `train_pos`: A data frame of $n$ rows and $l$ columns containing the positions of the training observations.

- `k`: The $k$ parameter of the $knn$ algorithm, the number of nearest neighbors to consider.
- `method`: The distance metric to be used by the algorithm. The implemented options are 'euclidean', 'manhatan', 'norm', 'LGD' and 'PLGD'
- `weights`: The weight function to be used by the algorithm. The implemented options are 'distance' and 'uniform'. The default 'distance' function calculate the weights from the distances as:

  $$w_{j,t} = \frac{1}{(1 + d_{j,t})\mathcal{W}_j}$$

  where $w_{j,t}$ is the weight assigned to the $t^{th}$ $(t \in [1..k])$ neighbor of the $j^{th}$ $(j \in [1..m])$ test observation, $d_{j,t}$ is the distance in the feature (RSSI) space between the $t^{th}$ neighbor and the $j^{th}$ test fingerprint, and $\mathcal{W}_j$ is a term used to normalize the values so that the total sum of the $k$ weights is 1.

  The 'uniform' function assigns the same weight value to each neighbor:

  $$w_{j,t} = \frac{1}{k}$$

- `norm, sd, epsilon, alpha, threshold`: Parameters for the 'norm', 'LGD' and 'PLGD' methods.
- `FUN`: An alternative function provided by the user to compute the distance.
- `...`: Additional parameters for the function `FUN`.

For a training data set of $n$ RSSI vectors (a data frame or a matrix named `tr_fingerprints`) and a data set of $n$ position vectors (a data frame or a matrix named `tr_positions`), the code for fitting a knn model with a $k$ value of 4 and the manhattan distance as the distance measurement method is:

```
knnModel <- ipfKnn(tr_fingerprints, tr_positions, k = 4, method = '
    manhattan')
```

This function returns an S3 object of class `ipftModel` containing the following properties:

- `params`: A list with the parameters passed to the function.

- `data`: A list with the fingerprints and the location data of the radio map.

To estimate the position of a new fingerprint, the `ipfEstimate` function makes use of the previously obtained model. An `ipfModel` object holds the data model needed by the `ipfEstimate` function to apply the selected algorithm and returns an estimation of the test fingerprints positions. The signature of `ipfEstimate` is:

```
ipfEstimate <- function(ipfmodel, test_fgp, test_pos = NULL)
```

where:

- `ipfmodel`: An S3 object of class `ipfModel`.
- `test_fgp`: A data frame of $m$ rows and $q$ columns containing the fingerprints of the test set.
- `test_pos`: An optional parameter containing a data frame of $m$ rows and $l$ columns with the position of the test observations.

The `ipfEstimate` function returns an S3 object of the class `ipfEstimation` with the following elements:

- `location`: A $m \times l$ matrix with the predicted position for each observation in the `test` data set.
- `errors`: If the actual location of the test observations is passed in parameter `test_pos`, and the data that represents the position is numeric, this property returns a numeric vector of length $n$ with the errors, calculated as the $euclidean$ distance between the actual and the predicted locations.
- `confusion`: If the actual location of the test observations is passed in parameter `test_pos`, and the data that represents the position is a factor, the estimation of the actual position is performed as a classification task, and this property returns a confusion matrix summarizing the results of this classification.
- `neighbors`: A $m \times k$ matrix with the indices of the $k$ selected neighbors for each observation in the `test` data set.

- `weights`: A $m \times k$ matrix containing the weights assigned by the algorithm to the selected neighbors.

The following `R` code shows an example of the usage of the `ipfKnn` function with the data set included in the package. This example takes the data previously scaled and generates a positioning model from the input data `trainRSSI` (the radio map) that is stored in `knnModel`. Then, the model is passed to the `ipfEstimate` function, along with the test data, to get an estimation of the position of the 702 test observations:

```
tr_fingerprints <- trainRSSI[, 1:168]
tr_positions    <- ipftrain[, 169:170]
knnModel        <- ipfKnn(tr_fingerprints, tr_positions, k = 7,
    method = "euclidean")
ts_fingerprints <- testRSSI[, 1:168]
ts_positions    <- ipftest[, 169:170]
knnEstimation   <- ipfEstimate(knnModel, ts_fingerprints, ts_
    positions)
```

Since the position of the test observations is known, the mean error for the 702 test observations can be calculated as follows:

```
> mean(knnEstimation$errors)
[1] 3.302739
```

The mean positioning error is one of the most common evaluation metrics used in indoor positioning [87] to assess the system's accuracy. This metric corresponds to the average Euclidean distance between the estimated locations and the true locations. As positions in the `ipftrain` and `ipftest` are expressed in meters, this metric represents the average error in meters for this scenario.

The neighbors selected from the training data set for the 6 first test fingerprints are:

```
> head(knnEstimation$neighbors)
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]   71  176  126  125  127  771  130
[2,]   71  176  126  125  127  771  130
[3,]  465  914  915  913  217   77  218
```

```
[4,]  465  914  915  176  913  461  217
[5,]  176  126  125  771  130  127  914
[6,]   77  914  915  217  176  465  218
```

where each row of the output corresponds to the indices of the $k$ = 7 more similar vectors from the training data set to the $i^{th}$ vector of the `test` data set.

As an example of how to use `ipfKnn` with a custom function, the next code shows the definition of a *C++* function that implements a modified version of the manhattan distance. The function needs at least two parameters, the two matrices representing the training and test data sets. A third parameter is here introduced to represent a penalization value. This function penalizes the computed distance between two RSSI measurements when one of the beacons is not detected (represented by the value $\emptyset$), by multiplying the resulting distance by a factor $F$. Given two fingerprints $\lambda_i^{tr}$ and $\lambda_j^{ts}$ of length $q$, the $myD$ distance is:

$$\mathrm{myD}(\lambda_i^{tr}, \lambda_j^{ts}) = \sum_{h=1}^{q} \mathrm{myd}(\rho_{h,i}^{tr}, \rho_{h,j}^{ts}),$$

where

$$\mathrm{myd}(\rho_{h,i}^{tr}, \rho_{h,j}^{ts}) = \begin{cases} |\rho_{h,i}^{tr} - \rho_{h,j}^{ts}|, & \text{if } \rho_{h,i}^{tr} \neq \emptyset \text{ and } \rho_{h,j}^{ts} \neq \emptyset \\ |\rho_{h,i}^{tr} - \rho_{h,j}^{ts}|F, & otherwise \end{cases}$$

The following code implements the `myD` function and shows an example of its usage with `ipfKnn`, as well as the results obtained. The function is coded in *C++* to improve its performance when using large data sets, although the method also accepts custom plain R functions. The `myD` function assumes that the fingerprints are in a positive range:

```
library('ipft')
library('Rcpp')
cppFunction('
  NumericMatrix myD(NumericMatrix train, NumericMatrix test, double F
      = 2.0) {
    NumericMatrix distanceMatrix(test.nrow(), train.nrow());
    double d = 0, pv = 0, rssi1 = 0, rssi2 = 0;
    for (int itrain = 0; itrain < train.nrow(); itrain++) {
      for (int itest = 0; itest < test.nrow(); itest++) {
          d = 0;
          for (int i = 0; i < train.ncol(); i++) {
            rssi1 = R_IsNA(train(itrain, i))? 0 : train(itrain, i);
            rssi2 = R_IsNA(test(itest, i))? 0 : test(itest, i);
            pv = (rssi1 != 0 && rssi2 != 0)? 1 : F;
            d = d + std::abs(rssi1 - rssi2) * pv;
          }
          distanceMatrix(itest, itrain) = d;
        }
      }
      return distanceMatrix;
    }'
)
customModel      <- ipfKnn(tr_fingerprints, tr_positions, k = 1, FUN
    = myD, F = 0.25)
customEstimation <- ipfEstimate(customModel, ts_fingerprints, ts_
    positions)

> head(customEstimation$neighbors)
     [,1]
[1,]  773
[2,]  773
[3,]  776
[4,]  773
[5,]  130
[6,]  130
```

The previous code outputs the selected neighbors for the first 6 observations in the test data set. As the `ts_positions` data frame contains the actual location of the observations, the absolute error committed by the model is returned in the `ipfEstimation` object:

```
> head(customEstimation$errors)
[1] 5.708275 5.708275 5.708275 5.708275 3.380000 3.380000
```

And the mean error with this custom similarity function is:

```
> mean(customEstimation$errors)
[1] 3.297342
```

An `ipfEstimation` object can be used directly to plot the Empirical cumulative distribution function of the error (function `ipfPlotEcdf()`) and the Probability density function (function `ipfPlotPdf()`). Figures 1.2 and 1.3 show the plots obtained from the following code:

```
> ipfPlotEcdf(customEstimation)
> ipfPlotPdf(customEstimation)
```

The plotting functions included in the package are described in detail in Section 1.8.

### 1.5.2 The `ipfProbabilistic` function.

Given the limitations of sensors accuracy [91] and the irregular character of signal propagation [3], the RSSI vector stored for a particular position cannot have completely reliable and accurate information about the emitters signal strength. This uncertainty is generally modeled by a normal distribution [58], but to do so many readings of the signals at the same position are needed to obtain a representative set of statistical parameters to model each RSSI present at that position.

Thus, the initial collection of RSSI observations associated to a particular point is transformed into a pair of vectors containing the means and the standard deviations of the RSSI for each beacon, and then the complete training data is stored as a set of statistical parameters that can be used to infer the location of a test observation as the one that maximizes a probability function.

Let $\widehat{\mathcal{D}}_{train}$ be the new training set obtained from the previous procedure:

$$\widehat{\mathcal{D}}_{train} = \left\{\widehat{\mathcal{F}}_{train},\ \widehat{\mathcal{L}}_{train}\right\}$$

$$\widehat{\mathcal{F}}_{train} = \left\{\widehat{\lambda_1^{tr}}, \widehat{\lambda_2^{tr}}, ..., \widehat{\lambda_g^{tr}}\right\}$$

$$\widehat{\mathcal{L}}_{train} = \left\{\widehat{\tau_1^{tr}}, \widehat{\tau_2^{tr}}, ..., \widehat{\tau_g^{tr}}\right\}$$

where $\widehat{\mathcal{F}}_{train}$ is the set of statistical parameters obtained from the fingerprints of the training set, $g$ is the number of groups of fingerprints with the same associated

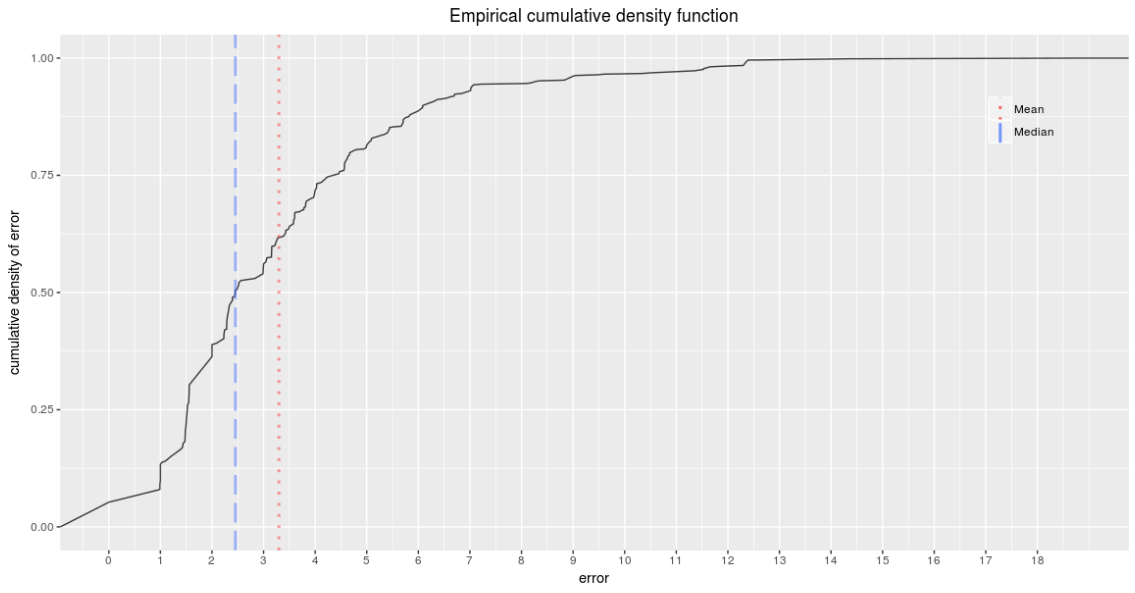


Figure 1.2: Funtion `ipfPlotEcdf`. Empirical cumulative distribution function of the error. The plot also shows the mean (red dotted line) and the median (blue dashed line) of the errors.

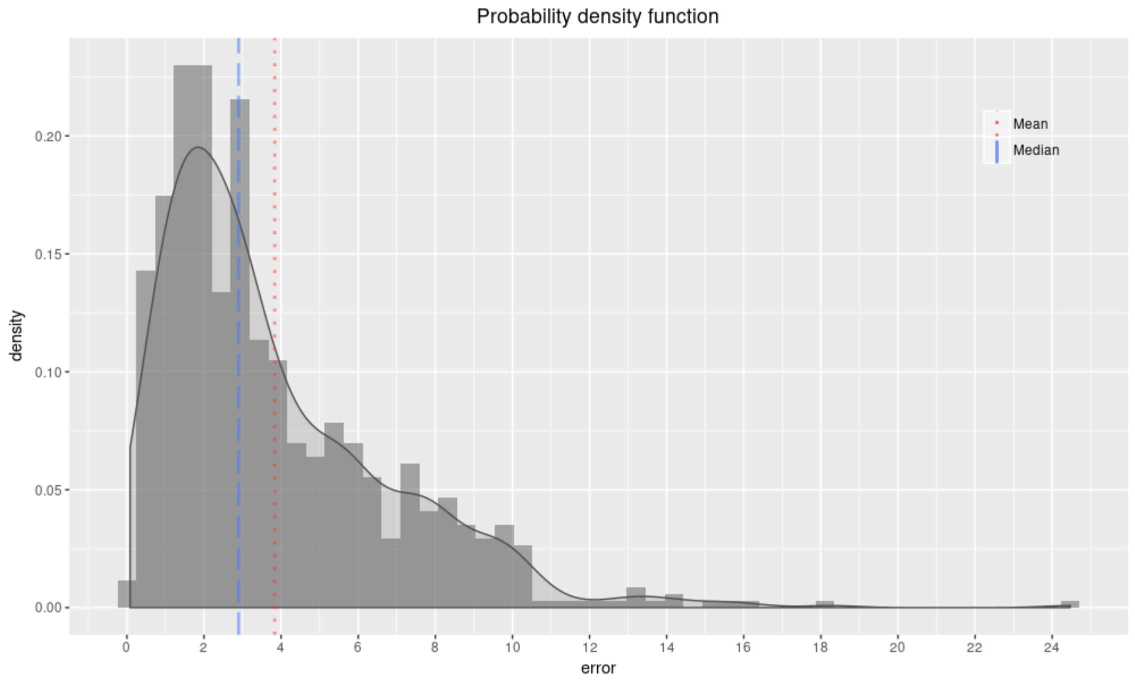


Figure 1.3: Funtion `ipfPlotPdf`. Probability density function. The plot shows the normalized histogram of the errors and its density function. The plot also shows the mean (red dotted line) and the median (blue dashed line) of the errors.

position, and $\widehat{\mathcal{L}}_{train}$ is the set of positions associated to each group. Each one of the $g$ observations of the training data set is now composed of a fingerprint with $q$ values:

$$\widehat{\lambda_i^{tr}} = \left\{\theta_{1,i}^{tr}, \theta_{2,i}^{tr}, ..., \theta_{q,i}^{tr}\right\},\ i \in [1, ..., g]$$

$$\theta_{h,i}^{tr} \sim \mathcal{N}(\mu_{h,i}, \sigma_{h,i}^2)$$

where $\mu_{h,i}$ and $\sigma_{h,i}^2$ are the mean and the variance, respectively, of the $h^{th}$ RSSI of the $i^{th}$ group of original fingerprints.

Let $\rho_{h,j}^{ts}$ be the $h^{th}$ RSSI measurement of the $j^{th}$ test fingerprint ($\lambda_j^{ts}$), and let $\mu_{h,i}$ and $\sigma_{h,i}^2$ be the mean and the standard deviation of the $h^{th}$ beacon distribution obtained for the $i^{th}$ position from the training set. The probability $p_{h,j}^{(i)}$, of observing $\rho_{h,j}^{ts}$ at the $i^{th}$ position is:

$$p_{h,j}^{(i)} = \int_{\rho_{h,j}^{ts}-\delta}^{\rho_{h,j}^{ts}+\delta} \frac{1}{\sigma_{h,i}\sqrt{2\pi}}\, \mathrm{e}^{-\frac{x-\mu_{h,i}}{2\sigma_{h,i}^2}}\, dx$$

where $\delta$ is a parameter to allow the discretization of the normal distribution (Figure 1.4).

The set of all probabilities $p_{h,j}^{(i)},\ h \in [1, ..., q]$ obtained for a given test observation $j$, expresses the similarity between the observation measurement and the training data for a particular location. An evaluation of the total similarity for every location can be computed as a function of these individual probabilities, like its sum or its product. In the `ipft` package, this algorithm is implemented by the `ipfProbabilistic` and `ipfEstimate` functions, and by default uses the sum of probabilities as default operator to evaluate the similarity:

$$\psi_j^{(i)} = \sum_{h=1}^{p} p_{h,j}^{(i)}$$

where $\psi_j^{(i)}$ is the similarity between the $j^{th}$ test observation and the $i^{th}$ distribution from the training data set. The function to evaluate the similarity can be passed to `ipfProbabilistic` as a parameter.

As well as the `ipfKnn` and `ipfProximity` functions, `ipfProbabilistic` returns a `ipfModel` object with the same data structure seen in Section 1.5.1, but

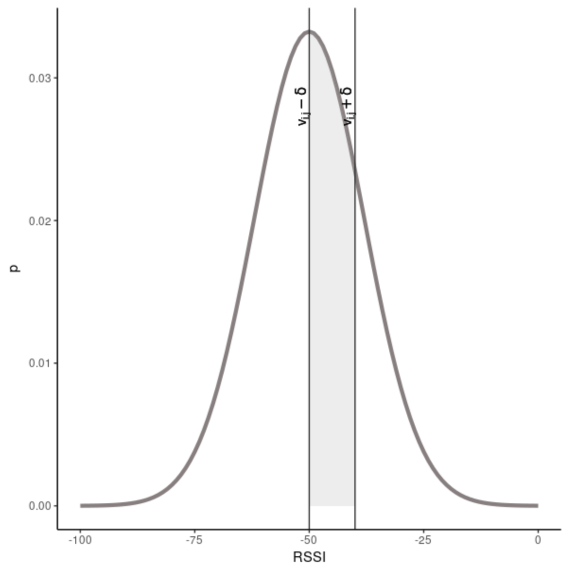


Figure 1.4: $\delta$ parameter for the probabilistic approach. This parameter sets the width of the discretization steps.

with the difference that now the `data` property returns the probabilistic parameters that define the fitted distributions for every group of fingerprints on the training set. The clustering or grouping of the training data is performed by default over the location data provided by the user, but this behavior can be customized by passing a parameter with the columns over which to group the data, or by passing the group indices directly. The `ipft` package implements two functions (`ipfGroup()` and `ipfCluster()`) to perform clustering tasks. These functions are described in Section 1.7.

The signature of the `ipfProbabilistic` function is:

```
ipfProbabilistic <- function(train_fgp, train_pos, group_cols = NULL,
     groups = NULL, k = 3, FUN = sum, delta = 1, ...)
```

where `train_fgp`, `train_pos` and `k` have the same meaning and structure as described in Section 1.5.1, and, given $n$ observations in the training set:

- `groups`: is a numeric vector of length $n$, containing the index of the group assigned to each observation of the training set. This parameter is optional.
- `group_cols`: is a character vector with the names of the columns to use as

criteria to form groups of fingerprints. This parameter is optional.

- `FUN`: is a function to estimate a similarity measure from the calculated probabilities.
- `delta`: is a parameter to specify the interval around the test RSSI value to take into account when determining the probability.
- `...`: are additional parameters for `FUN`.

The following code shows how to use the `ipfProbabilistic` function to obtain a probabilistic model from the `ipftrain` and `ipftest` data sets. The default behavior of `ipfProbabilistic` groups the training data attending at the position of each observation, in this case, its x and y coordinates:

```
> probModel <- ipfProbabilistic(tr_fingerprints, tr_positions, k = 7,
    delta = 10)
> head(probModel$data$positions)
     X     Y
1 -0.6 24.42
2 -0.6 27.42
3  0.0  0.00
4  0.4  0.00
5  0.4  3.38
6  0.4  6.81
```

Now the `ipfModel$data` property returns a list with 3 elements:

- `means`: a data frame with the means for every beacon and every group of fingerprints.
- `sds`: a data frame with the standard deviations for every beacon and every group of fingerprints.
- `positions`: a data frame with the position of each group of fingerprints.

To obtain an estimation from this model, the same code used in section 1.5.1 can be used to produce the estimated locations:

```
> ts_fingerprints <- ipftest[, 1:168]
> ts_positions    <- ipftest[, 169:170]
> probEstimation  <- ipfEstimate(probModel, ts_fingerprints, ts_
   positions)
```

and their errors and its mean value:

```
> mean(probEstimation$errors)
[1] 6.069336
```

An alternative function can be passed to `ipfProbabilistic`. The following code uses the maximum value of the probabilities as the similarity measure and passes a parameter to remove `NA`s from the data[6]:

```
> probModel <- ipfProbabilistic(tr_fingerprints, tr_positions, k = 9,
    delta = 10, FUN = max, na.rm = TRUE)
> probEstimation <- ipfEstimate(probModel, ts_fingerprints, ts_
   positions)
> mean(probEstimation$errors)
[1] 8.652321
```

### 1.5.3 The `ipfProximity` function.

When the location of the access points is known, it's possible to estimate the position of a fingerprint using the log-distance path loss model [144]. Given a set of $q$ beacons, and a fingerprint vector $\lambda = \{\rho_1, \rho_2, ..., \rho_q\}$ of length $q$, this model is expressed as:

$$\rho_h = P_{1m,h} - 10\alpha \log_{10} d_h - \gamma, \quad h \in [1, 2, ..., q]$$

where $\rho_h$ is the value of the received signal from the $h^{th}$ beacon, $d_h$ is the distance from the observation to the beacon, $P_{1m,h}$ is the received power at 1 meter from the emitter, $\alpha$ is the path loss exponent, and $\gamma \sim \mathcal{N}(0, \sigma_\gamma^2)$ represents a zero-mean Gaussian noise that models the random shadowing effects of the environment.

[6]The `ipfProbabilistic` function takes into account the `NA`s contained in the data when using the default function (sum), but the user needs to manage this situation when a custom function is provided. In this example, the data is not previously transformed, is passed as it is, with `NA`s for not detected WAPs, to illustrate this situation.

The estimator of the distance between the emitting beacon and the position where the signal is received is:

$$\hat{d}_h = 10^{\frac{\rho_h - P_{1m,h}}{10\alpha}}$$

This estimation follows a log-normal distribution that is:

$$\ln \hat{d}_h \sim \mathcal{N}(\ln d_h, \sigma_d^2)$$

where $\sigma_d = (\sigma_\gamma ln10)/(10\alpha)$.

The mean and the variance of the distribution are:

$$E[\hat{d}_h] = d_h \; e^{\sigma_d^2/2}$$
$$\textit{Var}[\hat{d}_h] = d_h^2 \; e^{\sigma_d^2} \; (e^{\sigma_d^2} - 1)$$

Note that the variance grows quadratically with the distance, making the estimation less reliable as the distance becomes larger. Therefore, the distances estimated from different beacons will have different accuracies. To take this into account, the algorithm estimates the position of a fingerprint as a minimization problem of the overall squared error of the estimated distances. The objective function to minimize is:

$$\underset{\tau}{min} J = \sum_{h=1}^{p} \omega_h (\hat{d}_h - \|s_h - \tau\|)^2$$

where $\tau$ is the position that minimizes the function, that is, the estimated position, $q$ is the number of beacons present in the fingerprint, and $\omega_h = 1/Var[\hat{d}_h]$ are the weights.

The functions `ipfProximity` and `ipfEstimate` implement this design, and uses the Broyden-Fletcher-Goldfard-Shano algorithm (BFGS) [30, 46, 53, 145], a quasi-Newton method, to minimize the previous function to make an estimation of the fingerprint position. The accuracy of the estimation is strongly dependent on the reliability of the emitters positions. When these positions are unknown, they can be estimated with the function `ipfEstimateBeaconPositions`. Section 1.6 details the implementation and usage of this function. The `ipfProximity` function returns an `ipfModel` object with the data needed by the `ipfEstimate` function to estimate a fingerprint position.

The signature of the `ipfProximity` function is:

```
ipfProximity <- function(bpos, rssirange = c(-100, 0), norssi = NA,
    alpha = 5, wapPow1 = -30)
```

where:

- `bpos`: a matrix or a data frame containing the position of the beacons, in the same order as they appear in fingerprints.
- `rssirange`: the range of the RSSI data in the fingerprints.
- `norssi`: the value used to represent a not detected beacon.
- `alpha`: the path loss exponent ($\alpha$).
- `wapPow1`: a numeric vector with the received power at one meter distance from the beacon ($P_{1m,h}$). If only one value is supplied, it will be assigned to all beacons.

In the following example, the goal is to estimate the position of the 702 fingerprints included in the test set, using the known position of the WAPs and the log-distance path loss model. The `ipfpwap` dataset contains the location of 39 of the 168 wireless access points of the `ipftrain` and `ipftest` data sets. The `ipfProximity` function returns a model that is used to estimate the position of the fingerprints. As the real position of the test fingerprints is known, this information can be also passed to the `ipfEstimate` function. Thus, the returned `ipfEstimation` object will contain, along with the estimated positions, the associated errors:

```
> proxModel      <- ipfProximity(ipfpwap, alpha = 4, rssirange = c
    (-100, 0), norssi = NA, wapPow1 = -32)
> fingerprints   <- ipftest[, 1:168]
> positions      <- ipftest[, 169:170]
> proxEstimation <- ipfEstimate(proxModel, ipftest[, 1:168], ipftest
    [, 169:170])
> mean(proxEstimation$errors)
[1] 8.0444
```

### 1.5.4 Positioning algorithms comparison

In a classical fingerprint-based positioning system, the radio map is constructed in accordance with the positioning algorithm to be used in the online phase. The knn algorithm follows a deterministic approach that performs well in most cases, while the probabilistic method is based on the assumption that there is enough training data for each particular position to obtain reliable parameters to model a distribution for each signal at each survey location. As regards the proximity algorithm, it is based on two assumptions; first, the ability to realistically simulate the propagation model of the signal, and second, the known positions of the emitter beacons. These conditions are not met in many scenarios, where changes in occupation, for example, modify the propagation model and thus the performance of the positioning system.

To illustrate the previous considerations, Table 1.1 shows the mean and the quartile errors in meters for the implemented algorithms, computed using the dataset included in the package. In this particular case, given the characteristics of the training data, knn performs better than the rest.

| | | Quartile error (m) | | | | |
|---|---|---|---|---|---|---|
| algorithm | mean error (m) | 0% | 25% | 50% | 75% | 100% |
| *knn* | 3.3027 | 0.15172 | 1.46891 | 2.61281 | 4.08992 | 19.84650 |
| *probabilistic* | 6.0693 | 0.14289 | 3.26988 | 5.63051 | 8.19933 | 17.93031 |
| *proximity* | 8.0444 | 2.49865 | 5.71055 | 7.42602 | 9.88427 | 20.12029 |

Table 1.1: Comparison of the algorithms' accuracy on the dataset included in the package

To compare the performance of the proposed implementation of the previous positioning algorithms, we ran a benchmark test of 1000 iterations on each function, using the dataset included in the package. The results for the model fitting functions are shown in Table 1.2. As can be seen, the proximity and knn algorithms are the fastest, as expected, since their model fitting process basically consists in storing the training data for later processing during the estimation stage. In contrast, the probabilistic algorithm has to fit a normal distribution for each signal received at each position, and thus, it takes longer to complete the process.

The outcomes are different when considering the results for the estimation

| function | elapsed (sec) | relative |
|---|---|---|
| `ipfKnn` | 0.031 | 1.409 |
| `ipfProbabilistic` | 1035.446 | 47065.727 |
| `ipfProximity` | 0.022 | 1.000 |

Table 1.2: Performance comparison of the model building functions

function (Table 1.3). The position estimation for the probabilistic algorithm is faster than the rest. For the knn algorithm, the estimation process could be improved using clustering techniques to avoid comparing the test fingerprint with all the instances in the training set. With regards to the estimation process for the proximity algorithm, the fact that the result is computed by solving an unconstrained nonlinear optimization through an iterative method highly penalizes its performance.

| model | function | elapsed (sec) | relative |
|---|---|---|---|
| *knn* | `ipfEstimate` | 2508.079 | 2.998 |
| *probabilistic* | `ipfEstimate` | 836.651 | 1.000 |
| *proximity* | `ipfEstimate` | 28259.110 | 33.776 |

Table 1.3: Performance comparison of the estimation functions on each model

## 1.6 Beacon position estimation

If the actual position of the beacons is unknown, it can be estimated in many ways from the RSSI data. Two basic methods for estimation of the beacons location have been included in the `ipft` package through the `ipfEstimateBeaconPositions` function. The 'centroid' and the 'weighted centroid' methods.

Both methods use the fingerprint data to guess the position of the beacons. Let $q$ be the number of beacons and $\tau^{\mathcal{B}}$ be the set of beacons locations:

$$\tau^{\mathcal{B}} = \left\{\nu^{\mathcal{B}}_{1,h}, \nu^{\mathcal{B}}_{2,h}, \nu^{\mathcal{B}}_{3,h}\right\},\ h \in [1, 2, ..., q]$$

the position of the $h^{th}$ beacon is given by:

$$\tau_h^{\mathcal{B}} = \left\{ \sum_{i=1}^{n} \omega_i \nu_{1,i}^{tr}, \sum_{i=1}^{n} \omega_i \nu_{2,i}^{tr}, \sum_{i=1}^{n} \omega_i \nu_{3,i}^{tr} \right\}$$

where $n$ is the number of fingerprints in the training set. The value of $\omega_i$ is:

$$\omega_i = \frac{1}{n}$$

for the 'centroid' method and:

$$\omega_i = \frac{\rho_{h,i}^{tr}}{\sum_{l=1}^{n} \rho_{h,l}^{tr}}$$

for the 'weighted centroid' method. Since the biggest weights have to be assigned to the strongest RSSI values, the fingerprint vector values should be positive, or at least, positively correlated to the beacon received intensity. This is checked by the function implementation so the input data is internally transformed to a positive range when needed.

This is the signature of the `ipfEstimateBeaconPositions` function:

```
ipfEstimateBeaconPositions <- function(fingerprints, positions, method
   = 'wcentroid', rssirange = c(-100, 0), norssi = NA)
```

where:

- `fingerprints`: is a data frame with the fingerprint vectors as rows.
- `positions`: a data frame with the position of the fingerprints.
- `method`: the method to use by the algorithm, either 'centroid' or 'wcentroid'.
- `rssirange`: the range of the signal strength values of the fingerprints.
- `norssi`: the value assigned in the fingerprints to a non detected beacon.

The following code uses the function `ipfEstimateBeaconPositions` with the 'weighted centroid' method to estimate the position of the wireless access points, under the assumption that this position is unknown. Finally, the function `ipfProximity` estimates the positions of the first 6 test fingerprints:

```
> bc_positions   <- ipfEstimateBeaconPositions(ts_fingerprints, ts_
   positions, method = 'wcentroid')
> proxModel      <- ipfProximity(bc_positions, rssirange = c(0.1, 1),
    norssi = NA)
> proxEstimation <- ipfEstimate(proxModel, fingerprints[1:6,],
   positions[1:6,])
> proxEstimation$location
        V1       V2
1 1.686950 12.02117
2 1.686950 12.02117
3 1.654255 10.91767
4 1.682121 10.96035
5 1.711448 10.88966
6 1.695007 10.09507
```

## 1.7 Data clustering

Clustering techniques can be used with the aim of enhancing localization performance and reducing computational overhead [39]. The `ipft` package includes some functions for cluster analysis and grouping of the fingerprinting and location data. These functions can be used to create or detect clusters based on the position of the observations, on its signal levels, or on any other criteria that might be useful to group the data by. Performing RSSI clustering before the positioning process groups a large number of reference points into various clusters that can be used to perform first-level classification. This allows assessing the testing point location by using only the fingerprints in the matched cluster rather than the whole radio map. Furthermore, given the amplitude attenuation that building partitions cause to electromagnetic signals, clusters usually can be related to physical spaces such as buildings, floors, or even rooms.

The main function for clustering tasks is `ipfCluster`. The more basic usage of the function takes the provided data and uses the k-means algorithm to classify it into $k$ disjoint sets of observations, by selecting a set of $k$ cluster centers to minimize the sum of the squared distances between the data vectors and their corresponding centers.

The k-means clustering procedure begins with an initial set of randomly selected centers and iteratively tries to minimize the sum of the squared distances. This makes the algorithm very sensitive to the arbitrary selection of initial centers and introduces variability in the results obtained from one execution to another. Besides, the number of clusters has to be established beforehand, and that may be inconvenient in some scenarios.

The signature of the `ipftCluster` function is:

```
ipfCluster <- function(data, method = 'k-means', k = NULL, grid =
    NULL, ...)
```

where

- `data`: is a data frame with the data to cluster. When using the *k-means* method, the data frame must not contain any `NA` values.
- `method`: the algorithm used to create clusters. The implemented algorithms are 'k-means' for k-means algorithm, 'grid' for clustering based on spatial grid partition, and 'AP' for affinity propagation algorithm.
- `k`: a numeric parameter for k-means algorithm.
- `grid`: a numeric vector with the size of the grid for the grid algorithm.

When using the default k-means algorithm, the function behaves as a wrapper around the k-means function of the `stats` package, and therefore, the usage can be further customized by passing extra parameters, as the number of iterations or the algorithm to be used ("Hartigan-Wong" is the default).

The following example will find $k = 30$ clusters of similar fingerprints in the `ipftrain` dataset. First, the data set of fingerprints is transformed to eliminate the `NA` values that represent a not detected beacon. Then, the data is passed to the `ipfCluster` function to find the 30 clusters using the 'MacQueen' algorithm:

```
> set.seed(1)
> cl_fingerprints <- ipfTransform(tr_fingerprints, inNoRSSI = NA,
    outNoRSSI = 0)
> clusterData     <- ipfCluster(cl_fingerprints, k = 30, iter.max =
    20, algorithm = "MacQueen")
> head(clusterData$clusters)
[1] 3 3 3 3 3 3
```

The outcome of the `ipfCluster` function is a list containing the indices of the $k$ clusters and its centroids. Given the previous example, `clusterData$centers` will return the $k$ centroids, and `clusterData$clusters` will return the cluster index $i \in [1, .., k]$ for every observation in `ipftrain`.

The `ipfCluster` function includes an implementation of the affinity propagation (AP) algorithm [47] that can be used to estimate the number of distinct clusters present in the radio map. AP does not require the number of clusters to be determined before running it. It finds members of the input set, known as 'exemplars', that are representative of clusters by creating the centers and the corresponding clusters based on the constant exchanging of reading similarities between the observations. This message-passing process continues until a good set of centers and corresponding clusters emerges.

The following code uses AP clustering to find groups of similar RSSI vectors from the `ipftrain` data set. With no further parametrization, it will classify the RSSI data into 43 distinct clusters:

```
> clusterData     <- ipfCluster(tr_fingerprints, method = 'AP')
> dim(clusterData$centers)
[1]  43 168
```

Now, `clusterData$centers` holds the 43 'exemplars', those RSSI vectors from the radio map that are representative of a cluster, and `clusterData$clusters` contains the indices that link every observation of the data set with its assigned cluster.

To perform a more simple grouping based on a precise set of variables, the `ipfGroup` function provides a method to group the data by column name. The function signature is:

```
ipfGroup <- function(data, ...)
```

where

- `data`: is a data frame with the data to group.
- `...`: The variables to group the data by.

The `ipfGroup` function returns a numeric vector with the same length as the number of observations contained in the `data` data frame, containing the index of the group assigned to each observation. The following example groups the data according to the position of the observations, that in the `ipftrain` and `ipftest` datasets are represented by the columns ’X’ and ’Y’:

```
> groups <- ipfGroup(ipftrain, X, Y)
> head(groups)
[1]  4  4  4  4 22 22
> length(unique(groups))
[1]  41
```

## 1.8 Plotting functions

Indoor positioning generally involves statistical analysis of datasets, and the `ipft` provides some useful functions to produce graphs for exploring data. All the graphic functions included in the package are built upon the `ggplot2` package [166], and return a `ggplot` object that can be plotted or further personalized with custom labels, theme, etc.

The `ipfPlotPdf` and the `ipfPlotEcdf` have already been introduced in Section 1.5.1. These functions will plot the probability density function and the empirical cumulative distribution function, respectively. Both functions take an `ipfEstimation` object to produce the plot, while the axis labels and plot tittle can be also supplied by the parameters `xlab`, `ylab` and `tittle`. Their respective signatures are:

```
ipfPlotPdf <- function(estimation, xlab = 'error', ylab = 'density',
    title = 'Probability density function')

ipfPlotEcdf <- function(estimation, xlab = 'error', ylab = '
    cumulative density of error', title = 'Empirical cumulative
    density function')
```

The function `ipfPlotLocation` will produce a plot of the location of the data. The following code shows its signature and presents an example of its use. The example calls the function with parameter `plabel` set to `TRUE`, to plot labels identifying each location, and `reverseAxis` set to `TRUE` to swap the axis. It also modifies the resulting object by changing the default `ggplot2` theme to the white one. The result is shown in Figure 1.5.

```
ipfPlotLocation <- function(positions, plabel = FALSE, reverseAxis =
    FALSE, xlab = NULL, ylab = NULL, title = '')
```

```
library(ggplot2)
ipfPlotLocation(ipftrain[, 169:170], plabel = TRUE, reverseAxis =
    TRUE) + theme_bw()
```

The function `ipfPlotEstimation` plots the estimated position of the test observations based on an `ipfModel` object and an `ipfEstimation` object, as well as the actual position (parameter `testpos`), if known, and the position of the $k$ selected fingerprints from the training set used to guess its location (parameter `showneighbors`). The green dots indicate the actual position of the observations, while the black dots indicate the estimated ones. The blue lines connect the estimated positions with the $k$ neighbors from which the location has been estimated, and the red arrows connect the actual position of the fingerprint with the estimated one. The following code shows the function signature and provides an example of its usage. The resulting plot is shown in Figure 1.6:

```
ipfPlotEstimation <- function(model, estimation, testpos = NULL,
    observations = c(1), reverseAxis = FALSE, showneighbors = FALSE,
    showLabels = FALSE, xlab = NULL, ylab = NULL, title = '')
```

```
library(ggplot2)
probModel <- ipfProbabilistic(ipftrain[, 1:168], ipftrain[, 169:170])
probEst   <- ipfEstimate(probModel, ipftest[, 1:168], ipftest[,
    169:170])
ipfPlotEstimation(probModel, probEst, ipftest[, 169:170],
    observations = c(61:62, 81:82), reverseAxis = TRUE, showneighbors
     = TRUE, showLabels = TRUE) + theme_bw()
```

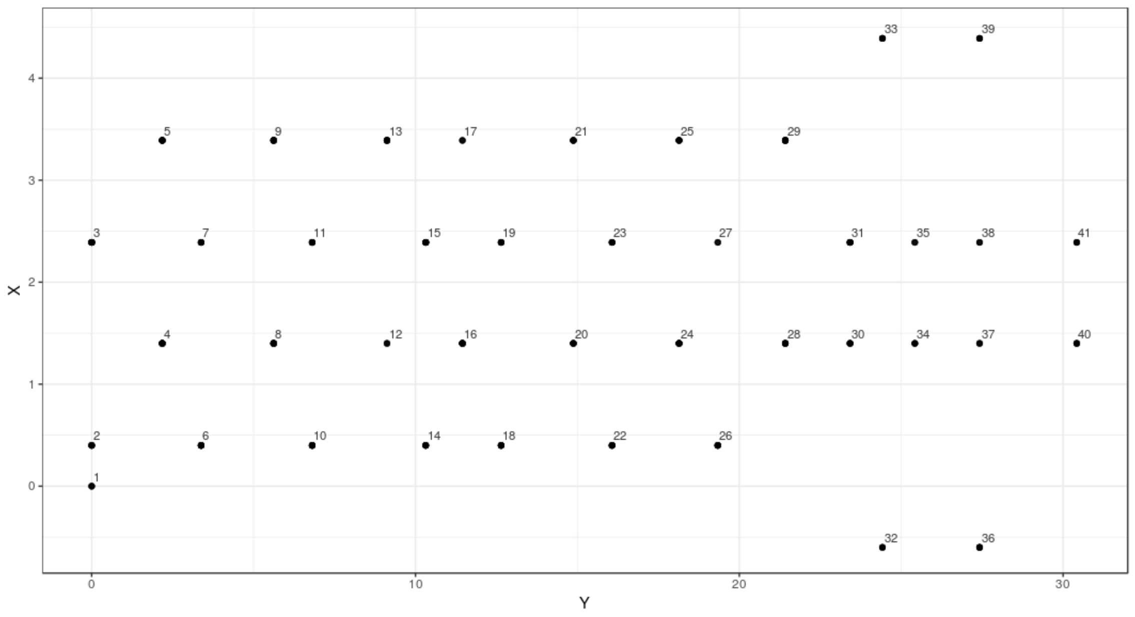


Figure 1.5: Location of fingerprints included in the `ipftrain` data frame. The labels indicate the group indices.

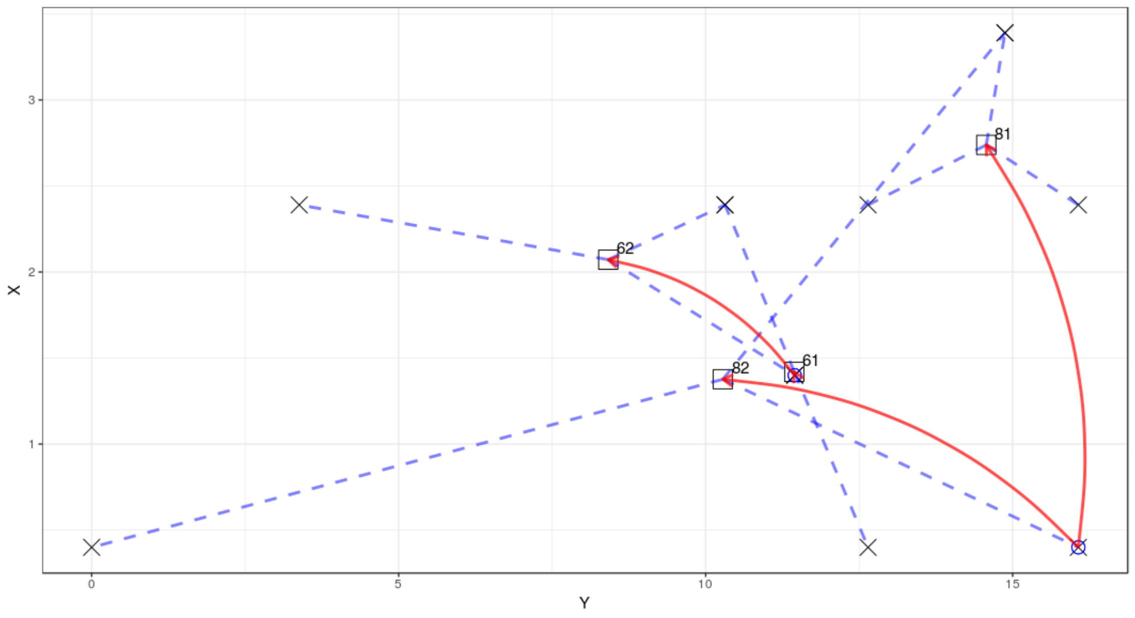


Figure 1.6: Estimated and actual positions of test observations 61, 62, 81 and 82 from the `ipftrain` data set. The circles indicate the actual positions of the observations. The squares show the estimated positions. The red arrows connect the actual positions with the estimated ones. The dashed lines connect the estimated positions with the $k$ neighbors from which the location has been estimated, represented by the crosses.

## 1.9 Summary

In this paper, the package `ipft` is presented. The main goal of the package is to provide researchers with a set of functions to manipulate, cluster, transform, create models, and make estimations using indoor localization fingerprinting data. This package enables researchers to use a well-established set of algorithms and tools to manipulate and model RSSI fingerprint data sets, and also allows them to customize the included algorithms with personalized parameters and functions to adapt the working mode to their particular research interests.

In this work, some of the fundamental algorithms used in indoor fingerprinting localization techniques have been formally presented and illustrated, while detailed examples and information about its usage and implementation have been provided.

## 1.10 Future work

This package is an ongoing work, and future versions will implement new algorithms and tools with the aim of providing a base framework for researchers, and become a reference library for fingerprinting-based indoor positioning research.

In particular, future lines of work should consider the implementation of deep learning-based algorithms. Many deep learning techniques can be exploited to try to obtain better positioning performance. Recurrent neural networks could be used to learn not only spatial but also temporal patterns of the received signals. Deep autoencoders can be implemented as a way to encode fingerprints and reduce their dimensionality to a few number of significant features. Their variational and generative extensions can be of use to better model the stochastic nature of RSSI data. These models can also be applied to generate new training data for deep learning-based classifiers, increasing the robustness of positioning systems, and trying to address problems caused by heterogeneity of devices.

## 1.11 Acknowledgements

The authors would like to thank the two anonymous reviewers for providing useful feedback that helped to improve the paper.

This work has been partially funded by the Spanish Ministry of Economy and Competitiveness through the "Proyectos I + D Excelencia" programme (TIN2015-

70202-P) and by Jaume I University "Research promotion plan 2017" programme (UJI-B2017-45).

# Chapter 2

# IndoorLoc Platform: A web tool to support the comparison of indoor positioning systems

The main objective of this document is to provide an introduction to the *IndoorLoc Platform*, a web tool to support the comparison and the evaluation of indoor positioning algorithms. The proposed web platform can be used to download data sets, learn how some well-known algorithms work, study the implementation of those algorithms, test the methods, and even upload indoor positioning estimations of the user's methods to check their accuracy in the same experimental conditions than other methods already included in a ranking, among other functionalities.

This paper also shows a comparative study of the accuracy of two well-known fingerprinting-based indoor localization algorithms using the data sets included in the platform. This comparative study can be performed using the tools included in the platform.

## 2.1 Introduction

Geolocation systems have been present during decades allowing navigation services that guide users by car, foot or bicycle, and many others such as evacuation services and social network services. The Global Navigation Satellite System (GNSS) are able to provide these services in outdoor environments, but in most of the situations, people spend a significant portion of their time in indoor environments such as offices, undergrounds, shopping malls, airports, etc., where these satellite-based positioning systems do not work. This is one of the reasons why the development of new indoor positioning and navigation systems has attracted the attention of many researchers in the last years.

This research effort has achieved the development of many different indoor positioning technologies, being the ones based on RSSI (Received Signal Strength Indicator) fingerprinting [65, 168, 60] among the most popular. This technique is based on the measurement of the intensity of the received radio signals of the emitting devices (beacons) that are available at a particular place and on the comparison of this measurement with a previously built RSSI data set (also known as radio map). In this scenario, a fingerprint is an RSSI feature vector composed of received signal values from different emitting devices or beacons, associated to a precise position. The similarity of the received signals (fingerprint) with some of the stored fingerprints can be used to guess the approximate position on the subject. This technique is becoming increasingly important for indoor localization, since Wi-Fi is generally available in indoor environments where GPS signals cannot penetrate, and the wireless access points (WAPs) can be used as opportunistic beacons. Other types of indoor localization beacons (Bluetooth, RFID, etc.) can also be used in conjunction with Wi-Fi access points or as a standalone positioning system.

Many different approaches have been the object of research and many papers have been published trying to solve this indoor localization problem. However, it is very difficult to compare results from different approaches, since every research presents its estimated results using its own experimental setup and measures, and it is very difficult to reproduce the particularities of every single experiment. In the Pattern Recognition and Machine Learning research fields, the common practice is to test the results of each proposal using several well-known data sets [49]. This allows researchers to fairly compare different methodologies in the literature. For instance, the UCI Machine Learning Repository [86] and the web *Kaggle* [52] are two well-known examples in this sense. However, in the fingerprint-based indoor localization research field, there is a limited number of such kind of databases [110, 158, 153, 155, 19, 99].

This paper consists of an introduction to the *IndoorLoc Platform*[1], a web tool to

[1] http://indoorlocplatform.uji.es

support the comparison and the evaluation of indoor positioning algorithms. The platform is a centralized website where researchers can the following actions:

1. Access to a public repository of data sets for RSSI fingerprinting.
2. Upload indoor positioning estimations on experimental setups included in the platform.
3. Include the estimation results in a ranking.
4. Analyze positioning methods.
5. Interact with the platform in a user-friendly environment to test the algorithms and data sets included.

In order to show a real example of the platform usage, this paper also presents a comparative study of the performance of two fingerprinting-based indoor localization methods included in the platform when using four of the data sets also included in the platform. All the experiments presented are easily reproducible using the tools included in the platform.

The two methods shown differ in the methodology used to solve the indoor localization problem. They are a deterministic-based and a probabilistic-based method. The four data sets differ in the type of scenario where data has been captured, as for instance: the number of samples, the size of the scenario, the density of the samples, etc.

A preliminary version of this work was published (as a conference paper) in [105]. This paper provides additional details of the *IndoorLoc Platform*, to help the reader to be aware of the different possibilities of the proposed web platform.

The rest of the paper is organized as follows. Section 2.2 reviews related work. Section 2.3 describes the main sections in which the platform is divided. Sections 2.4 and 2.5 explain the data sets and methods, respectively, included in the platform. Section 2.6 presents a set of experiments performed using the algorithms and data sets included in the proposed platform. Section 2.7 describes a real case of use of the usage of the platform during a fingerprinting-based indoor positioning course. Finally, the most important conclusions arisen from this work are presented in Section 2.8.

## 2.2 Related work

As it has been said in Section 2.1, most of the indoor positioning methods found in the literature present the experiments using their own experimental setup. A second related problem is that those data sets are not made available to the research community, making it impossible to reproduce the presented results. Both issues make a fair comparison of localization methods developed by different groups not feasible in a rigorous manner, since scenarios may change in an uncontrolled way.

A better way to compare positioning algorithms is to use the same experimental setup, and for that purpose, the use of a repository of prerecorded data in a large variety of buildings and contexts can be very useful. Some good examples of data repositories in the machine learning community are the UCI Machine Learning Repository [86] and *Kaggle* [52], both created for evaluating machine learning algorithms with common databases.

Other alternatives are competitions where several research groups should prepare their methods to obtain the best results using a common experimental setup, or even the same prerecorded data. Some examples of competition are: Microsoft-IPSN [92, 94, 95, 93], EvAAL [119] and EVARILOS [83]. The first off-site indoor location competition was the third track of the EvAAL-ETRI Indoor Location competition [160], called *Wi-Fi fingerprinting in large environments*, which was held during the Sixth International Conference on Indoor Positioning and Indoor Navigation (IPIN'15). In this event, the competitors had access to the *UJIIndoorLoc* [158] data set, which has been included in the proposed platform. A similar competition was held in the Seventh International Conference on Indoor Positioning and Indoor Navigation (IPIN'16) [159], where the data set used was more challenging since data provided by all sensors embedded in typical smartphones were included, acquired by different people moving in different types of buildings.

One of the main problems of such competitions is that when they finish, researchers cannot continue improving their methods. In addition, the different data sets are located on different web pages. The proposed web platform is focused on providing a common place for researchers to access to fingerprint-related data sets. Another of the main objectives is to provide a continuous competition without deadlines. Therefore, researchers will have not time restrictions to test their methods and submit their results to the platform.

The most similar work to the *IndoorLoc Platform* is [83], where the authors presented a web platform for evaluation of RF-based indoor localization algorithms with two core services: one focused on the storage of raw data and the other focused on automated calculation of metrics for performance assessment. They also include an SDK for convenient access to the platform from MATLAB and python. The two first characteristics are included in the proposed web platform. The SDK is not needed in our case since users can directly interact with the web platform to upload their results. The main differences in the proposed web platform with respect to [83] are as follows:

1. It is more focused on fingerprinting methods.

2. It also includes a dashboard section where researchers can make experiments using the methods and data sets included in the platform in a user-friendly environment.

3. There is a ranking section where researchers can check the accuracy of their method against the methods of other researchers. In addition, the proposed web platform has been designed in order to easily upload new methods and data sets.

The *IndoorLoc Platform* has been designed with a state-of-art visual style and with a user-centered interface making access to all the elements of the platform very intuitive. For instance, the home web page (see Figure 2.1) directly presents the main sections of the platform. Another example is that users can download a data set or upload a result with just a few mouse clicks. The platform is also responsive, and will automatically adapt to the device screen used to access it.

In addition, the platform has a high formative component, because even a user without programming knowledge can interact with the algorithms and data sets included. Although, it will be the users with a high programming skill who will be able to get better advantage of the platform because, probably, they will be able to improve the results that can be obtained with the algorithms included in the platform.

## 2.3 Overview of the platform

Figure 2.1 shows the homepage of the platform. The homepage displays a summary of the contents of the four main sections in which the platform is structured, while the menu in the upper section of the platform allows access to each one of these sections.

The four main sections of the platform are as follows:

- **data sets**: This section is a repository of several data sets stored in the platform. These data sets are available to download so users can use them in their experiments.
- **Ranking**: In this section, users can upload the results of their own algorithms to obtain an estimation of the accuracy of their methods when using the data sets included in the platform. In addition, the results can be included in the ranking, where the best results of each data set are showed sorted by accuracy.
- **Methods**: This section presents a set of well-known algorithms so users can study their implementation.
- **Dashboard**: In this section, users can test the algorithms included in the platform, using some of the data sets included, in a user-friendly environment.

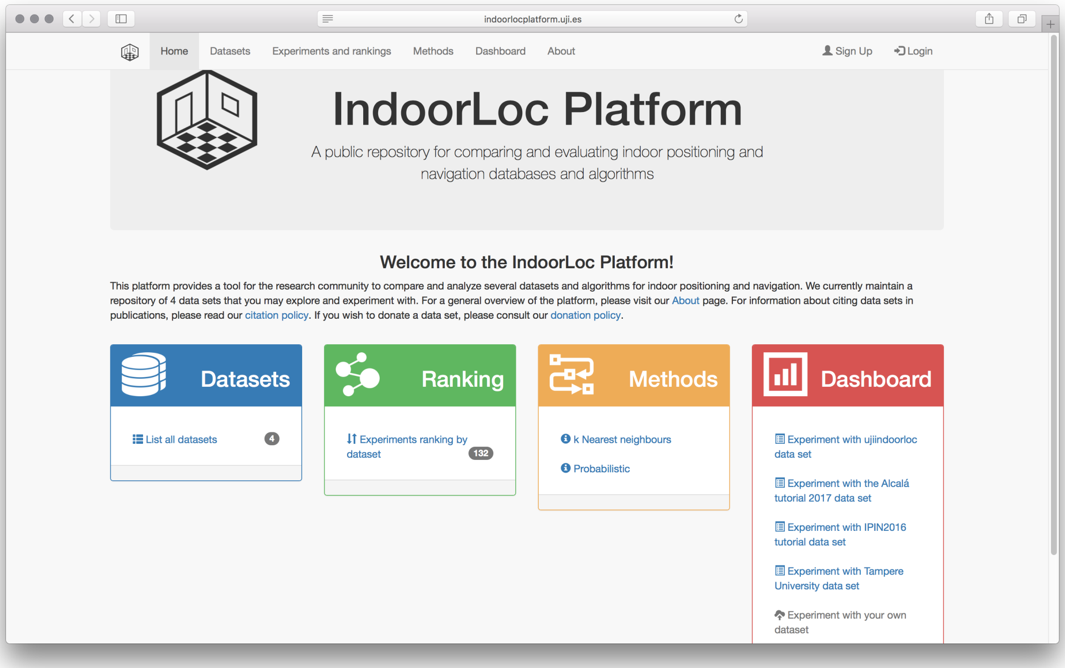
These sections are briefly described in the next subsections.

### 2.3.1 Data sets

Figure 2.2 shows the data sets section of the platform. The *data sets* section displays the basic information about all the data sets included in the platform. In addition, the links to download all the files related to each data set are also included.

Each data set can be composed of up to four files:

- **data set info**: A *pdf* file with information and features about the data set. The description includes the name of the donors, the contact information, general information about the data set, a description of the files included, the attributes description, the format of the result file and the citation request.



IndoorLoc Platform

A public repository for comparing and evaluating indoor positioning and navigation databases and algorithms

Welcome to the IndoorLoc Platform!

This platform provides a tool for the research community to compare and analyze several datasets and algorithms for indoor positioning and navigation. We currently maintain a repository of 4 data sets that you may explore and experiment with. For a general overview of the platform, please visit our About page. For information about citing data sets in publications, please read our citation policy. If you wish to donate a data set, please consult our donation policy.

Datasets
- List all datasets 4

Ranking
- Experiments ranking by dataset 132

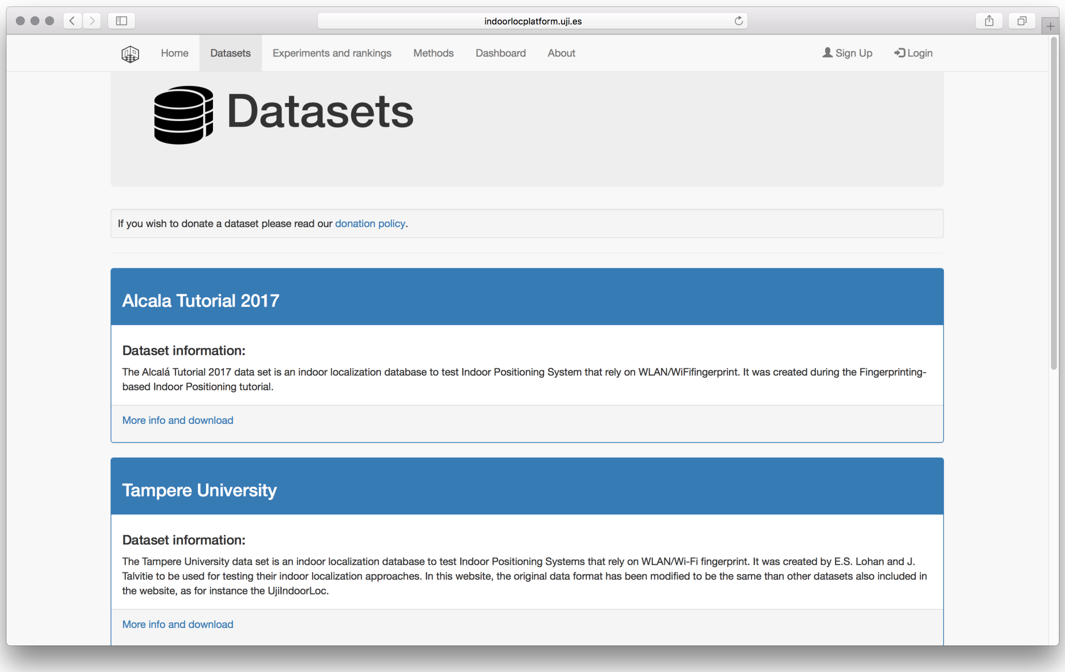
Methods
- k Nearest neighbours
- Probabilistic

Dashboard
- Experiment with ujiindoorloc data set
- Experiment with the Alcalá tutorial 2017 data set
- Experiment with IPIN2016 tutorial data set
- Experiment with Tampere University data set
- Experiment with your own dataset

Figure 2.1: Homepage of the Indoorloc platform

- **Training set**: A file with the samples to be used to train the localization models. It includes the localization of the samples.
- **Validation set**: This file is similar to the training set file, and also includes the localization of the samples. Should be used to assess the performance of the localization model created using the training set data.
- **Test set**: This file is also used to assess the performance of the localization model, but does not include the actual localization of the samples, only its fingerprint. To obtain an estimation of the accuracy of the model, users can run their methods to obtain an estimation of the localization of the samples of the test set, and then upload their results to the platform to get an evaluation of the performance of the model. The true localization of the samples are stored in the platform.

The training, validation and test files have a *comma-separated values* (CSV) file format. The three first files (info, training, and validation) are accessible to everyone. Only registered users are allowed to access the test file. Not all the data



Datasets

If you wish to donate a dataset please read our donation policy.

Alcala Tutorial 2017

Dataset information:

The Alcalá Tutorial 2017 data set is an indoor localization database to test Indoor Positioning System that rely on WLAN/WiFifingerprint. It was created during the Fingerprinting-based Indoor Positioning tutorial.

More info and download

Tampere University

Dataset information:

The Tampere University data set is an indoor localization database to test Indoor Positioning Systems that rely on WLAN/Wi-Fi fingerprint. It was created by E.S. Lohan and J. Talvitie to be used for testing their indoor localization approaches. In this website, the original data format has been modified to be the same than other datasets also included in the website, as for instance the UjiIndoorLoc.

More info and download

Figure 2.2: Data sets section of the Indoorloc platform

sets included in the platform have a validation set. In that case, users can use techniques as Cross-validation [27] to assess the performance of the localization model generated.

At the time of writing this article, there are six different databases included in the platform. Four of them are related to the Wi-Fi fingerprinting indoor localization problem. They are briefly described in Section 2.4.

Registered users can upload their own data sets following the instructions provided by the platform. Before to be definitively added to the platform, each new data set is rigorously examined by the administrators of the platform to ensure that it has the required quality.

### 2.3.2 Ranking

One of the main objectives of the *indoorloc* web platform is to provide a tool for the indoor localization community to compare their methods using well-known data sets. This section has been devoted to this purpose. For each data set, a list of the best methods, according to a figure of merit, is shown.

Registered users are allowed to upload the results of their methods following the instructions included in the description of the data set (see Section 2.3.1). Once the results file has been uploaded, the platform calculates the figure of merit for this data set using the estimated locations provided by the user and the ground truth internally (and privately) stored in the platform. After the figure of merit is calculated and displayed, users have the choice of including or not the result in the ranking.

Each entry in the ranking has a description field, provided by the user, showing info about the experiment performed to obtain such result, e.g. the parameters used or the algorithm details.

Figure 2.3 shows the ranking page for the *IPIN2016 Tutorial* data set. At the time that this text is written, the ranking is composed of two experiments performed by the same user. According to the notes written by the user, the result of the leader was obtained using the probabilistic method and the one in the second position using a knn algorithm.

Figure 2.4 shows the ranking for the *UjiIndoorLoc* data set. In this case, the four best results obtained in the third track of the EvAAL-ETRI Indoor Location competition [160], where this data set was used, have been manually introduced by the web creators to give a baseline reference.

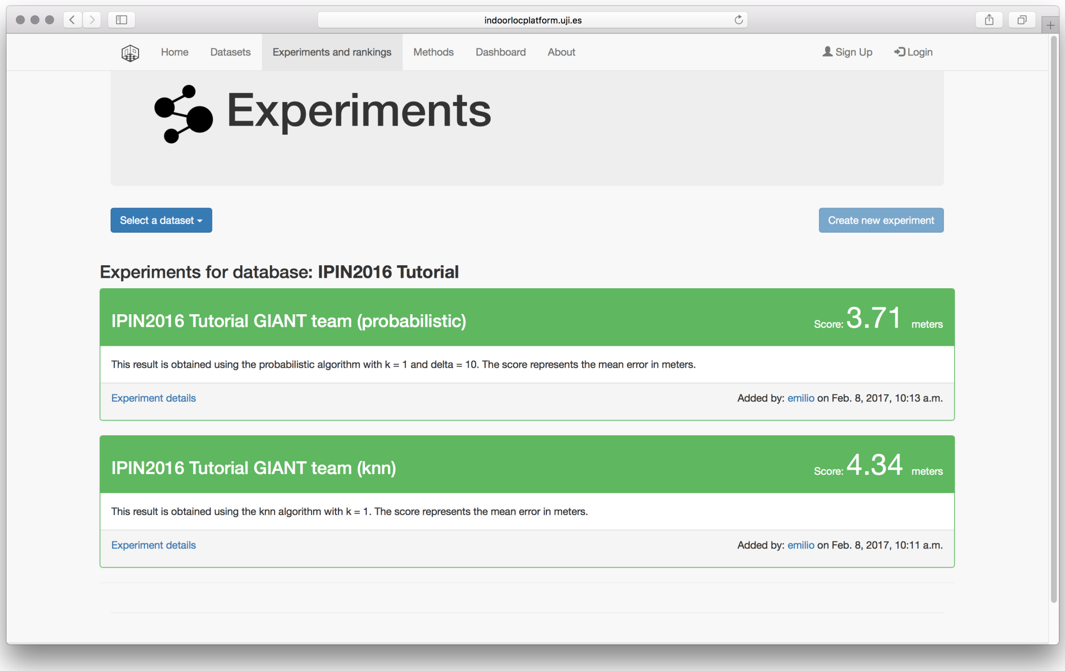
### 2.3.3 Methods

This section shows some basic information about the methods included in the platform. This information consists on the explanation of the method though R[2] source code using comprehensible examples. In addition, links to the Dashboard section, where users can test these methods, are also included.

At the time of writing this text, two methods have been included in the platform: deterministic-based [16] and probabilistic-based [174]. They are described in Section 2.5.

To add new methods to the platform, users must contact with platform administrators. Similarly to the data set case, each new method is rigorously examined by the administrators of the platform to ensure that it has the required quality.

[2]https://www.r-project.org/



Experiments

Experiments for database: **IPIN2016 Tutorial**

IPIN2016 Tutorial GIANT team (probabilistic) Score: 3.71 meters

This result is obtained using the probabilistic algorithm with k = 1 and delta = 10. The score represents the mean error in meters.

Added by: emilio on Feb. 8, 2017, 10:13 a.m.

IPIN2016 Tutorial GIANT team (knn) Score: 4.34 meters

This result is obtained using the knn algorithm with k = 1. The score represents the mean error in meters.

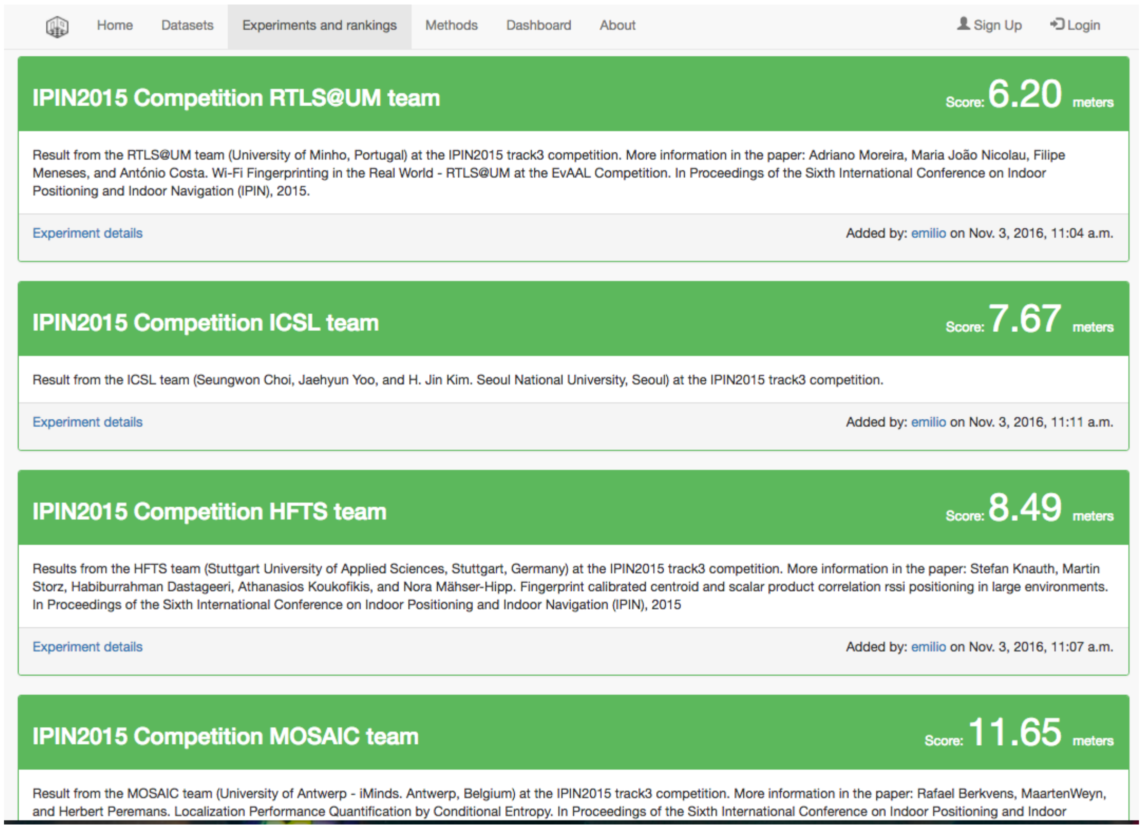
Added by: emilio on Feb. 8, 2017, 10:11 a.m.

Figure 2.3: Ranking webpage of the *IPIN2016 Tutorial* data set. Two experiments have been included in the ranking, the first one (according to the notes written by the contributor) using a probabilistic-based algorithm and the second one using a knn-based method.

### 2.3.4 Dashboard

In the Dashboard section, users can test the methods included in the platform, using the data sets also included in the platform, in a friendly user interface. Figure 2.6 shows an example of a dashboard for the *UjiIndoorloc* data set. In particular, the user selected the building 0, floor 1, the validation set (to estimate the locations), and the deterministic method. After clicking on the *Estimate error* button, the platform internally estimates the location of the validation samples and calculates some statistics, as the mean and the median of the estimation error. It also shows one figure with the error histogram and density, and another figure with the empirical cumulative density function.

Registered users are allowed to use their own data set using the methods included in the platform. This data set must be formatted using a set of rules specified in the web platform.



IPIN2015 Competition RTLS@UM team Score: 6.20 meters

Result from the RTLS@UM team (University of Minho, Portugal) at the IPIN2015 track3 competition. More information in the paper: Adriano Moreira, Maria João Nicolau, Filipe Meneses, and António Costa. Wi-Fi Fingerprinting in the Real World - RTLS@UM at the EvAAL Competition. In Proceedings of the Sixth International Conference on Indoor Positioning and Indoor Navigation (IPIN), 2015.

Experiment details Added by: emilio on Nov. 3, 2016, 11:04 a.m.

IPIN2015 Competition ICSL team Score: 7.67 meters

Result from the ICSL team (Seungwon Choi, Jaehyun Yoo, and H. Jin Kim. Seoul National University, Seoul) at the IPIN2015 track3 competition.

Experiment details Added by: emilio on Nov. 3, 2016, 11:11 a.m.

IPIN2015 Competition HFTS team Score: 8.49 meters

Results from the HFTS team (Stuttgart University of Applied Sciences, Stuttgart, Germany) at the IPIN2015 track3 competition. More information in the paper: Stefan Knauth, Martin Storz, Habiburrahman Dastageeri, Athanasios Koukofikis, and Nora Mähser-Hipp. Fingerprint calibrated centroid and scalar product correlation rssi positioning in large environments. In Proceedings of the Sixth International Conference on Indoor Positioning and Indoor Navigation (IPIN), 2015

Experiment details Added by: emilio on Nov. 3, 2016, 11:07 a.m.

IPIN2015 Competition MOSAIC team Score: 11.65 meters

Result from the MOSAIC team (University of Antwerp - iMinds. Antwerp, Belgium) at the IPIN2015 track3 competition. More information in the paper: Rafael Berkvens, MaartenWeyn, and Herbert Peremans. Localization Performance Quantification by Conditional Entropy. In Proceedings of the Sixth International Conference on Indoor Positioning and Indoor

Figure 2.4: Ranking webpage of the *UjiIndoorLoc* data set.

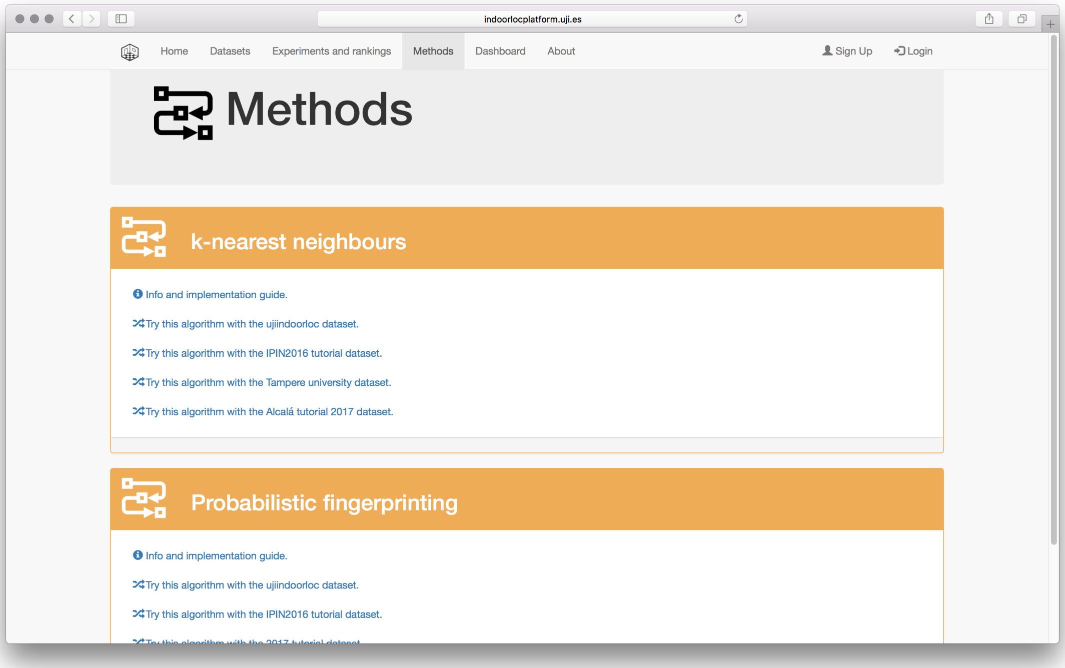
The platform can be easily improved adding more methods performance measurements and with other kinds of figures, thanks to the R Shiny environment.

### 2.3.5 Implementation details

The platform has been implemented using these open-source tools:

- **Django**[3]: A Python web framework to build the web application. Django follows the model-view-template (MVT) architectural pattern, and allows rapid development of database-driven websites.
- **Shiny**[4]: An R package that eases the building of interactive web *apps* using R code. The Shiny server hosts the *apps* on the platform that runs embedded in the dashboard page.

[3]https://www.djangoproject.com/
[4]https://shiny.rstudio.com/



Methods

k-nearest neighbours

Info and implementation guide.

Try this algorithm with the ujiindoorloc dataset.

Try this algorithm with the IPIN2016 tutorial dataset.

Try this algorithm with the Tampere university dataset.

Try this algorithm with the Alcalá tutorial 2017 dataset.

Probabilistic fingerprinting

Info and implementation guide.

Try this algorithm with the ujiindoorloc dataset.

Try this algorithm with the IPIN2016 tutorial dataset.

Try this algorithm with the 2017 tutorial dataset.

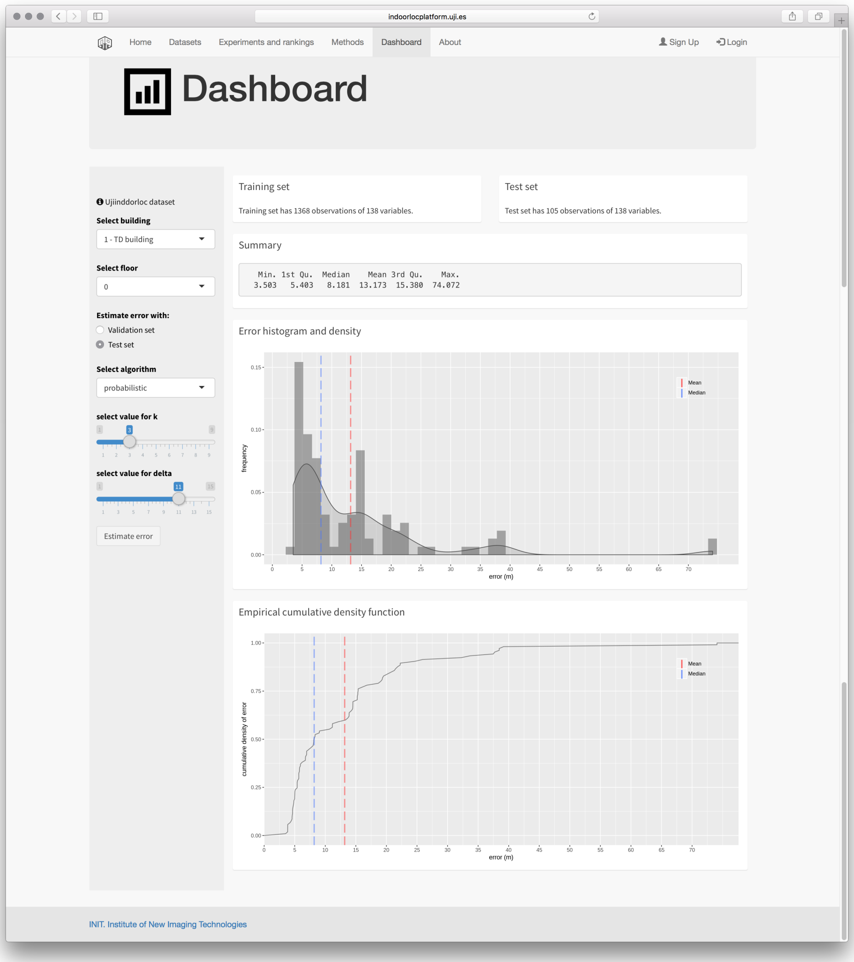
Figure 2.5: Methods section of the Indoorloc platform

- **RMarkdown**[5]: Documents created with the R Markdown technology are fully reproducible, and use a notebook interface to weave together text and code to produce elegantly formatted output. R Markdown uses multiple languages including R, Python, and SQL.

- **ipft R package**[142] [6]: This R package includes algorithms and utility functions for indoor positioning using fingerprinting techniques. These functions are designed for manipulation of RSSI (Received Signal Strength Intensity) data sets, estimation of positions, comparison of the performance of different models, and graphical visualization of data.

- **Apache**[7]: Open source HTTP web server that works on Unix-like systems (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh, and other platforms, and provides HTTP services in sync with the current HTTP standards.

[5]https://rmarkdown.rstudio.com/
[6]https://cran.r-project.org/web/packages/ipft/
[7]https://httpd.apache.org/



Ujiinddorloc dataset

**Select building**: 1 - TD building

**Select floor**: 0

**Estimate error with:** Validation set [✓ Test set]

**Select algorithm**: probabilistic

**select value for k**: 3

**select value for delta**: 11

Training set

Training set has 1368 observations of 138 variables.

Test set

Test set has 105 observations of 138 variables.

Summary

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  3.503   5.403   8.181  13.173  15.380  74.072
```

Figure 2.6: Example of the *Dashboard* section of the platform

## 2.4 Data sets included in the platform

At the moment of writing this document, the platform includes six different data sets, four of them dedicated to the Wi-Fi fingerprinting problem. They are briefly introduced in the next sections.

### 2.4.1 Wi-Fi based data sets

In the four Wi-Fi based data sets, Wi-Fi fingerprints are characterized by the detected Wireless Access Points (WAPs) and the corresponding Received Signal

Table 2.1: Main characteristic of the *UjiIndoorLoc* data set. # stands for "number of"

| | |
|---|---|
| # Buildings | 3 |
| # Floors | 4-5 |
| # WAPS | 520 |
| # training samples | 19937 |
| # validation samples | 1111 |
| # test samples | 4900 |

Table 2.2: Main characteristic of the *IPIN2016 Tutorial* data set. # stands for "number of"

| | |
|---|---|
| # Buildings | 1 |
| # Floors | 1 |
| # WAPS | 168 |
| # training samples | 927 |
| # validation samples | 0 |
| # test samples | 702 |

Strength Intensity (RSSI). The intensity values are represented as negative integer values near to $-100$ dBm (extremely poor signal) to $0$ dBm. The positive value 100 is used to denote when a WAP was not detected. Tables 2.1, 2.2, 2.3 and 2.4 show a summary of the main characteristics of each data set.

### UJIIndoorLoc

The UJIIndoorLoc [158] database covers three buildings of Universitat Jaume I[8] (Spain), with 4 or more floors and an area of almost 110000 m$^2$. It can be used for classification, e.g. actual building and floor identification, or regression, e.g. actual longitude and latitude estimation. It was created in 2013 and 2014 by means of more than 20 different users and 25 Android devices. The database consists of 19937 training/reference records and 1111 validation records. There is also a test file where the ground truth is not accessible.

The 529 attributes contain the Wi-Fi fingerprint, the coordinates (latitude, longitude, floor) and Building ID, and other useful information such as the particular

[8]http://www.uji.es

Table 2.3: Main characteristic of the *Tampere* data set. # stands for "number of"

| | |
|---|---|
| # Buildings | 2 |
| # Floors | 3-4 |
| # WAPS | 390-354 |
| # training samples | 1478-583 |
| # validation samples | 0 |
| # test samples | 589-175 |

Table 2.4: Main characteristic of the *Alcalá2017 Tutorial* data set. # stands for "number of"

| | |
|---|---|
| # Buildings | 1 |
| # Floors | 1 |
| # WAPS | 152 |
| # training samples | 670 |
| # validation samples | 0 |
| # test samples | 405 |

space (offices, labs, etc.) and the relative position (inside/outside the space) where the capture was taken, information about who (user), how (android device and version) and when (timestamp) Wi-Fi capture was taken, among other information. During the database creation, 520 different WAPs were detected. Thus, the Wi-Fi fingerprint is composed of 520 intensity values.

This data set was used in the off-site track of the EvAALETRI Indoor Localization Competition which was part of the Sixth International Conference on Indoor Positioning and Indoor Navigation (IPIN'15) [160]. The best results obtained in the competition have been included in the platform in the corresponding ranking.

Since the particular implementation of the localization methods included in the platform assumes that all the samples are in the same building and floor, the complete data set has been divided into 11 different data sets.

### IPIN2016 Tutorial

As an alternative to the *UJIIndoorLoc* data set, the *IPIN2016 Tutorial* data set is focused on the study of a small scenario. In particular, it covers a corridor of the

School of Engineering of the University of Alcalá[9] (Spain). It is the place where a tutorial on Wi-Fi fingerprinting was held during the IPIN2016 conference. The database consists of 927 training/reference records and 702 test ones. The 177 attributes contain the Wi-Fi fingerprint (168 WAPs), the coordinates where it was taken, and other useful information.

### Tampere University

This database [38] covers two buildings of the Tampere University of technology[10] (Finland), with 4 and 3 floors, respectively. In the first building, there are 1478 training/reference records and 489 test ones. The 312 attributes contain the Wi-Fi fingerprint (309 WAPs) and the coordinates (longitude, latitude, and height). In the second building, there are 583 training/reference records and 175 test ones. The 357 attributes contain the Wi-Fi fingerprint (354 WAPs) and the coordinates (longitude, latitude, and height). An important difference of this data set, with respect to the *UjiIndoorLoc*, is that in the former there is just one sample in each training location, while in the latter the number of samples is between 10 and 30.

Data from the two buildings can be considered as two separate data sets, with no relationship between respective WAP labels and real access points MAC addresses, meaning that two columns with the same WAP name in either data set may be assigned to different access points.

Similarly to the *UJIIndoorLoc* this data set has been divided into 7 different data sets.

### ALCALA2017 Tutorial

This data set was created during the 2017 Fingerprinting-based Indoor Positioning tutorial held in the School of Engineering of the University of Alcalá. Data were acquired in the same corridor as the *IPIN2016 Tutorial* data set. The main differences between both data sets are: 1) a thinner grid was used to capture training data; 2) some users made mistakes labeling the training fingerprints. These errors have not been eliminated since it is a situation that can occur in a real scenario. Users should take into account this situation in their methods.

The database consists of 670 training/reference records and 405 test ones. The

[9]https://www.uah.es/es/

[10]http://www.tut.fi/

154 attributes contain the Wi-Fi fingerprint (152 WAPs) and the coordinates where it was taken.

### 2.4.2 AmbiLoc data set

The *AmbiLoc* data set [117] is a collection of ambient radio fingerprints, collected in multiple predefined locations across several testbeds. Instead of Wi-Fi signals, the *AmbiLoc* data sets deals with ambient signals of opportunity, such as those from broadcasting TV and FM radio stations or GSM networks, that are almost always present on most indoor locations. This data set has been collected in multiple testbeds, including large-scale and multi-floor buildings, over the course of one year.

The platform provides some basic information of this data set and a link to the original source of the data[11].

### 2.4.3 magPIE data set

Magnetic field-based indoor positioning [84, 15, 101] is an infrastructure-less approach which is based on the uniqueness of the disturbances in the magnetic field produced by the structural elements present in a scenario. The uniqueness of the disturbances can be used as a fingerprint since it is stable over time.

*MagPIE* is a publicly available data set for the evaluation of indoor positioning algorithms that use magnetic anomalies [61]. This data set contains IMU and magnetometer measurements along with ground truth position measurements with centimeter-level accuracy. To produce this data set, the authors collected over 13 hours of data from three different buildings, with sensors both handheld and mounted on a wheeled robot, in environments with and without changes in the placement of objects that affect magnetometer measurements.

The platform provides some basic information on this data set and a link to the original source of the data[12].

[11]http://ambiloc.org/

[12]http://bretl.csl.illinois.edu/magpie/

## 2.5 Methods included in the platform

Two different approaches are considered here for the fingerprinting-based location process: a deterministic, or non-parametric method; and a statistical, or parametric method. In the first, no statistical behavior is assumed, and the location problem is solved according to a set of observations whose positions are known; while the second method makes explicit use of distributions and statistical parameters of the data stored in the radio map to optimize the probabilities in the assignment of the estimated position.

### 2.5.1 Deterministic-based approach

The deterministic approach [16, 97, 157] relies on the well known k-Nearest Neighbors algorithm (*knn*) [37] to, given an RSSI vector, select the $k$ more similar training examples from the radio map. The similarity between the RSSI value vectors can be determined, for example, as the $Euclidean$ distance between them, but other distance functions can be used instead [157]. Once the $k$ neighbors are selected, the method estimates the location of the user by calculating the weighted average of the neighbor's positions.

Figure 2.7 shows a possible R source code of this method (with $k = 3$). The R dataframe *training.set* contains the RSSI values and the localization of the training points. The last two columns are the longitude (column name LONG) and latitude (column name LAT) of those points. *The validation.set* dataframe has the same structure. The complete description can be found at: http://indoorlocplatform.uji.es/methods/knn/.

### 2.5.2 Probabilistic-based approach

Given the limitations of sensors' accuracy and the complex character of signal propagation, the RSSI vector stored for a particular position cannot have completely reliable and accurate information about the emitters' signal strength. This uncertainty has been usually modeled by a normal distribution [57], therefore many readings of the signals at the same position are needed to obtain a representative set of statistical parameters to model each RSSI present at that position. The more measurements for a particular location, the more reliable will be their inferred statistical parameters.

```
k               <- 3
n_observations <- nrow(training.set)
n_features     <- ncol(training.set) - 2
distances      <- matrix(0, 1, n_observations)
for (i in 1:n_observations) {
  distances[1, i] <- sqrt(sum((training_set[i, 1:n_features] -
      validation_set[1, 1:n_features])^2))
}
nearest       <- order(distances)[1:k]
weights       <- 1 / distances[nearest]
weights       <- weights / sum(weights)
est.longitude <- sum(training.set$LONG[nearest] * weights)
est.latitude  <- sum(training.set$LAT[nearest]  * weights)
```

Figure 2.7: A possible R source code for the deterministic method

In the Probabilistic-based approach [57, 174, 96], the initial collection of RSSI observations associated to a particular point is transformed into a pair of vectors containing the means and the standard deviations of the RSSI for each beacon, and then the complete training data is stored as a set of statistical parameters. Then, given a test fingerprint, for each beacon, it is possible to estimate a probability value that expresses the similarity between the observation measurement at this beacon and the training data for a particular location. An evaluation of the total similarity for every location can be computed as a function of these individual probabilities.

The algorithm selects the $k$ training samples with higher probability and, similarly to the deterministic method, it estimates the location of the user by calculating the weighted average of the selected samples positions.

Figure 2.8 shows a possible R source code of this method (with $k = 3$). Input data has the same structure than in the deterministic method. The complete description can be found at: http://indoorlocplatform.uji.es/methods/probabilistic/.

```
library(dplyr)
k           <- 3
delta       <- 10
data.means <- training.set        %>%
              group_by(LONG, LAT) %>%
              summarise_each(funs(mean))
data.sds   <- training.set        %>%
              group_by(LONG, LAT) %>%
              summarise_each(funs(sd))
n_max       <- nrow(data.means)
n_waps      <- ncol(data.means) - 2
p           <- matrix(0, n_max, n_waps)
for (n in 1:n_max) {
    n_means <- as.numeric(data.means[n, 3:(n_waps + 2)])
    n_sds   <- as.numeric(data.sds[n, 3:(n_waps + 2)])

    for (j in 1:n_waps) {
      o1      <- validation.set[1, j] - delta
      o2      <- validation.set[1, j] + delta
      p1      <- pnorm(o1, mean=n_means[j], sd=n_sds[j], lower.tail=
         FALSE)
      p2      <- pnorm(o2, mean=n_means[j], sd=n_sds[j], lower.tail=
         FALSE)
      p[n, j] <- p1 - p2
    }
}
similarity    <- rowSums(p)
similar       <- order(similarity, decreasing=TRUE)[1:k]
weights       <- similarity[similar]/sum(similarity[similar])
est.longitude <- sum(data.means$LONG[similar] * weights)
est.latitude  <- sum(data.means$LAT[similar] * weights)
```

Figure 2.8: A possible R source code for the probabilistic-based method

## 2.6 Experiments

The two methods explained in Section 2.5 have been tested with the four Wi-Fi-based data sets described in the Section 2.4, using the tools included in the *Dashboard* section of the platform. Therefore, they are easily reproducible. All

Table 2.5: Mean positioning error (in meters) of both methods on the *UJIIndoorLoc* data set. # stands for number of.

| Building | Floor | # samples | Deterministic | Probabilistic |
|---|---|---|---|---|
| 0 | 0 | 17 | 4.26 | 7.83 |
| 0 | 1 | 17 | 5.65 | 6.77 |
| 0 | 2 | 60 | 6.06 | 5.79 |
| 1 | 0 | 105 | 9.62 | 11.26 |
| 1 | 1 | 147 | 7.65 | 20.42 |
| 1 | 2 | 132 | 5.40 | 8.99 |
| 1 | 3 | 140 | 8.16 | 11.0 |
| 2 | 1 | 20 | 6.64 | 10.09 |
| 2 | 2 | 19 | 7.96 | 9.07 |
| 2 | 3 | 19 | 3.88 | 4.57 |
| 2 | 4 | 22 | 12.50 | 21.31 |
| Mean | | 704 | 7.18 | 10.64 |

possible combination of the parameters have been tested. Only the combination of tuning parameters obtaining the best result is showed. In all cases, the test data set has been used to assess the performance of the algorithms. The figure of merit used to provide an estimation of the performance of the methods is the mean localization error between the estimated position and the real one (internally known by the platform) of all test samples.

Table 2.5 shows the results obtained using the *UjiIndoorLoc* data set. Note that there is no data set for the building 0, floor 3 and for the building 2, floor 0, since there are not samples for these floors in the test set.

Table 2.6 shows the results on the *Tampere* data set. In this case, only the results obtained with the deterministic approach are showed, since the probabilistic-based method can only be applied when there are enough samples at each position to calculate the estimation of the statistical parameters needed for the correct operation of this method.

Finally, Tables 2.7 and 2.8 show the results on the *IPIN2016 Tutorial* and *ALCALA2017 Tutorial* data sets. In both cases, all the samples are in the same building and floor, therefore it is not necessary to divide the data into subsets.

In the case of the *UJIIndoorloc* data set, the deterministic method provides

Table 2.6: Mean positioning error (in meters) of both methods on the *Tampere* data set. # stands for number of.

| Building | Floor | # samples | Deterministic |
|---|---|---|---|
| 1 | 1 | 156 | 9.83 |
| 1 | 2 | 110 | 14.21 |
| 1 | 3 | 118 | 8.01 |
| 1 | 4 | 105 | 13.03 |
| 2 | 1 | 61 | 15.87 |
| 2 | 2 | 77 | 8.38 |
| 2 | 3 | 37 | 6.74 |
| Mean | | 664 | 10.86 |

Table 2.7: Mean positioning error (in meters) of both methods on the *IPIN2016 Tutorial* data set. # stands for number of.

| # samples | Deterministic | Probabilistic |
|---|---|---|
| 702 | 4.21 | 3.55 |

Table 2.8: Mean positioning error (in meters) of both methods on the *ALCALA2017 Tutorial* data set. # stands for number of.

| # samples | Deterministic | Probabilistic |
|---|---|---|
| 405 | 5.03 | 2.53 |

better results than the probabilistic one in almost all the cases. The differences in the results obtained across buildings and floors depend on the quality of the radio map capture at each scenario and also on the structural characteristics of each scenario. According to the mean accuracy, the deterministic-based approach is preferable.

There is also a high variability across buildings and floors in the results obtained for the *Tampere* data set due to the same reasons than in the *UJIIndoorloc* data set.

In the case of the *IPIN2016 Tutorial* and *ALCALA2017 Tutorial*, the results are very similar and in both cases the probabilistic approach is preferable. Note that, in the *ALCALA2017 Tutorial* data set, the difference is quite significant since the probabilistic-based approach can deal with the unintentional mistakes introduced by some of the data set creators. Results obtained for these data sets are better than the ones obtained for the *UJIIndoorloc* and the *Tampere* data sets since the scenarios of the Tutorial data sets correspond to small areas and more fingerprints per $m^2$ than the other two. Therefore position error is lower.

Table 2.9 shows the Radio Map Inherent Difficulty (RMID) value of each data set [136]. This measure gives an estimation of the inherent difficulty of a radio map to obtain accurate estimates. According to this value, *IPIN2016 Tutorial* and *ALCALA2017 Tutorial* are scenarios where it is easier to obtain accurate results. However, as confirmed by the results showed in Tables 2.5 and 2.6, the RMID value of the *UJIIndoorloc* and *Tampere* data sets shows that both data sets are quite complex and therefore, it is quite difficult to obtain positioning estimation with high accuracy.

Table 2.9: Value of $RMID$ obtained for each data set. The data is ordered by ascending $RMID$ value.

| Data set | Building | Floor | $RMID$ |
|---|---|---|---|
| Alcalá2017 Tutorial | 1 | 1 | 0.05 |
| IPIN2016 Tutorial | 1 | 1 | 0.08 |
| UjiIndoorLoc | 0 | 1 | 0.22 |
| UjiIndoorLoc | 0 | 0 | 0.23 |
| UjiIndoorLoc | 2 | 2 | 0.29 |
| UjiIndoorLoc | 1 | 2 | 0.33 |
| UjiIndoorLoc | 2 | 3 | 0.33 |
| UjiIndoorLoc | 0 | 3 | 0.34 |
| UjiIndoorLoc | 2 | 1 | 0.35 |
| UjiIndoorLoc | 0 | 2 | 0.36 |
| UjiIndoorLoc | 2 | 0 | 0.49 |
| UjiIndoorLoc | 1 | 3 | 0.65 |
| UjiIndoorLoc | 1 | 1 | 0.66 |
| Tampere | 1 | 4 | 0.67 |
| UjiIndoorLoc | 1 | 0 | 0.67 |
| UjiIndoorLoc | 2 | 4 | 0.70 |
| Tampere | 1 | 3 | 0.72 |
| Tampere | 1 | 2 | 0.77 |
| Tampere | 1 | 1 | 0.80 |
| Tampere | 2 | 1 | 0.89 |
| Tampere | 2 | 2 | 0.90 |
| Tampere | 2 | 3 | 0.91 |

Taking into account all the data sets, the mean localization error of the deterministic-based method is $6.79$ meters without including the results of the *Tampere* data set, and $8.21$ including it. The mean error of the probabilistic-based method is 9.47. Therefore, according to the results, in general, it seems that the deterministic-based method is preferable. However, taking into account the differences of the four scenarios, the deterministic-based approach gets better results in big scenarios with low density of data (*UjiIndoorloc* and *Tampere* data sets), while the probabilistic based one is preferable in small ones with high density of data

(*IPIN2016 Tutorial* and *ALCALA2017 Tutorial* data sets).

Note that the results obtained with the methods included in the platform can be effectively improved using more sophisticated algorithms, and also using modern machine learning techniques. For instance, the ranking of the *UjiIndoorloc* data set include some better results than the presented in the Table 2.5, since they are the best results obtained in the *Wi-Fi fingerprinting in large environments* (IPIN'15) competition.

## 2.7 The platform in use

The performance of the platform was tested during the 2017 Fingerprinting-based Indoor Positioning tutorial held in the School of Engineering of the University of Alcalá. As an activity of the course, an indoor localization competition took place using the *ALCALA2017 Tutorial* data set.

The 15 attendees were invited to use the platform to upload the results of their proposals. Some of them used the *Dashboard* section of the platform to test different parameter configurations of the localization methods included in the platform, and others manually programmed their own method from the source code provided by the course instructors. After a very competitive and exciting competition, the winner team got an error of only 2.14 meters using a probabilistic method. This result is even better than the best one that can be directly obtained using the Dashboard included in the platform.

In general, tutorial attendees were able to easily use the platform, mainly the *data set* download section, the *Dashboard* section and, obviously, the *Ranking* sections. Almost no queries to the course introduction were produced, showing the effective user-centered design applied to the platform.

## 2.8 Conclusions

In this paper, the *IndoorLoc Platform* has been presented. It is a public repository for comparing and evaluating indoor positioning algorithms. The proposed web platform can be used to download data sets, learn how some well-known algorithms work, study the source code of those algorithms, test the methods, and even upload results of the user's methods to check the accuracy when comparing against the results provided by other methods already included in a ranking, among other functionalities.

To present a real example of the usage of the platform, a comparative study of the accuracy of two well-known fingerprinting-based indoor localization algorithms, using four of the data sets included in the platform, have been also presented. According to the results obtained, the deterministic-based approach gets better results in big scenarios while the probabilistic based one is preferable in small scenarios. These experiments are easily reproducible using the tools included in the platform.

This web platform is an ongoing project, and future versions will implement new algorithms and include more data sets, with the aim to provide an interesting tool for researchers and become a reference web platform for indoor positioning research. For this purpose, researchers are invited to include more methods and data sets on the platform.

## Acknowledgments

The authors gratefully acknowledge funding from the Spanish Ministry of Economy and Competitiveness in the projects: ‘Proyectos I+D Excelencia’ TIN2015-70202-P and ‘Redes de Excelencia’ TEC2015-71426-REDT.

# Chapter 3

# A study of Deep Neural Networks for Human Activity Recognition

Human activity recognition and deep learning are two fields that have attracted attention in recent years. The former due to its relevance in many application domains, such as ambient assisted living or health monitoring, and the latter for its recent and excellent performance achievements in different domains of application such as image and speech recognition. In this paper, an extensive analysis among the most suited deep learning architectures for activity recognition is conducted to compare their performance in terms of accuracy, speed, and memory requirements. In particular, convolutional neural networks (CNN), long short term memory networks (LSTM), bidirectional LSTM (biLSTM), gated recurrent unit networks (GRU) and deep belief networks (DBN) have been tested on a total of ten publicly available datasets, with different sensors, sets of activities and sampling rates. All tests have been designed under a multi-modal approach to take advantage of synchronized raw sensor' signals.

Results show that CNNs are efficient at capturing local temporal dependencies of activity signals, as well as at identifying correlations among sensors. Their performance in activity classification is comparable to, and in most cases better than, the performance of recurrent models. Their faster response and lower memory footprint make them the architecture of choice for wearable and IoT devices.

## 3.1 Introduction

The main goals of Human Activity Recognition (HAR) systems are to observe and analyze human activities, and to interpret ongoing events successfully. There are several application domains, such as video surveillance systems, ambient assisted living (AAL) systems for smart homes, or health care monitoring applications, that require a reliable activity recognition system.

HAR systems retrieve and process contextual data to classify human behavior into a set of complex activities (i.e. standing, walking, jogging). Image-based HAR analyzes human behavior using images or videos, which is generally considered a highly intrusive technique. On the contrary, sensor-based HAR systems study the motion data coming from wearable sensors such as accelerometers, gyroscopes, RFID, and so on. Besides the inclusion of such sensors, the ubiquity and unobtrusiveness of smart-phones and smart-watches, and the availability of different wireless interfaces, such as Wi-Fi, 3G, and Bluetooth, make them an attractive platform for human activity recognition.

A traditional HAR pipeline employs individual smart devices to collect raw data from embedded sensors. Then, the associated activity is extracted from the data by applying machine learning and data mining techniques. Typically, this process has been divided into three components:

- **Sensing**: The system monitors an actor's behavior by gathering relevant data through a series of sensors, such as accelerometers or gyroscopes.
- **Feature processing**: The sensors' data is processed into handcrafted features that extract relevant and discriminative characteristics from the raw data.
- **Classification**: The human activity is inferred by a machine learning model designed to recognize an activity from the previously obtained features.

The aforementioned process can be simplified through the use of deep learning (DL), a subfield of machine learning that has recently achieved large success. Although the main concept of deep learning has been around for a few decades, it has not been until lately that this technique has emerged, thanks to the achievement of its excellent empirical performance in several different domains of applications such as speech recognition, image recognition, and natural language processing [68, 98].

Deep learning finds features and classification boundaries through optimizing a certain loss function, employing a deep neural network architecture. The typical structure of a DL algorithm stacks multiple layers of simple neural network architecture to extract hierarchical abstractions. Each layer combines features derived from previous layers and transforms them via a non-linear function to produce a new feature set. This process builds a hierarchy where basic abstract features are detected in the first layers of the network and are combined in the deeper ones to form complex feature maps, giving the network the ability to automatically learn the best features for a specific problem domain. This feature set is learned by the model from the input data. There is no need for human intervention to manually craft explicative features. The training process of deep learning algorithms can extract key features from raw data and, at the same time, find the meaningful patterns that characterize each category of data (different activities in the case of HAR). This is the strong point of deep learning against traditional machine learning approaches.

The ubiquity of wearable devices provides an excellent platform to detect what activity is a user performing. In recent years, several works have explored the problem of identifying human actions using inertial signals from body-worn devices. Many of them investigated the use of manually engineered features extracted from the data as the input to different classical machine learning algorithms. More recently, the majority of research is centered on effectively applying DL to the detection of human activities.

The present work provides a broad comparison of the performance of five representative deep learning architectures for human activity recognition. The comparative is based on a set of premises:

- The main goal of this work is to compare the different performance of various deep learning architectures in the problem of human activity recognition using raw inertial data. Thus, we avoid the use of engineered features.

- To provide a broader scope for comparison, we use a total of ten publicly available benchmark datasets. The differences among the datasets, such as the set of recorded activities, the subjects' conditions, the sensors typology (dedicated IMUs, smart-phones, and smart-watches) or the on-body sensor placements, provide us with an enormous quantity of heterogeneous data that allows for reaching more meaningful and generalizable results.

- The data used for the experiments is composed of inertial signals from accelerometers and gyroscopes from a variety of devices worn on several positions, in different environments, and by various users.
- Previous works have shown that using accelerometer and gyroscope data provides better results than using either alone. Therefore, the data used as input to the algorithms is composed of the combined signals of both sensors.
- There is a significant number of different DL architectures, many of them aimed to solve very specific problems. We limit our comparison to five standard models that cover the majority of architectures used in recent works in the field of HAR and DL.
- We conducted an extensive set of experiments using ten different publicly available datasets containing motion data from accelerometer and gyroscope sensors.
- We compared the general performance of five different deep learning architectures (DBN, CNN, LSTM, biLSTM, and GRU) on human activity recognition tasks.
- We compared the computational cost and memory footprint among the best performing models of each architecture, to assess their suitability in environments with a scarcity of resources, such as wearable and Internet of Things (IoT) devices.

As far as we know, this is the first paper comparing the performance and suitability of various DL architectures over such a large number of different datasets. The diversity and heterogeneity of the considered data, acquired by different people in several environments using diverse devices and procedures, enforces the conclusions of this work.

The remaining of the paper is organized as follows: Section 3.2 describes the related work on DL and human activity recognition. Section 3.3 provides a brief introduction to DL and the basis of the DBN, CNN, and RNN architectures. Section 3.4 describes the datasets used in the study. The experiments description, methodology, and setup are presented in Section 3.5. Results are presented and discussed in Section 3.6. Finally, Section 3.7 presents the main conclusions.

## 3.2 Related work

Classical machine learning algorithms such as decision trees, Bayesian methods (naïve Bayes and Bayesian networks), k-nearest neighbors, neural networks (multilayer perceptron), support vector machines, fuzzy logic, regression models, Markov models (hidden Markov models, conditional random fields) and classifier ensembles (boosting and bagging) [77] have been used traditionally to address human activity recognition tasks. In the last years, following the wave of deep learning renaissance, several works have underlined the potential of a variety of deep learning architectures for activity recognition, such as deep convolutional neural networks (CNN), many types of deep feed-forward networks (FFN), such as deep belief networks (DBN) and Restricted Boltzman Machines (RBM), and different flavors of recurrent neural networks (RNN). These works show that DL algorithms are generally better than classical methods for activity recognition, with the added advantage of not requiring a preliminary stage of feature engineering.

Research on HAR using deep learning has been based on three different main approaches. In [116] researchers compared various algorithms for feature extraction using four datasets and accelerometer data. The results showed other techniques such as PCA+ECDF outperformed Restricted Boltzmann Machines [69] as feature extractors. Some later works [177, 4, 25, 123] employed DBNs and RBMs in classification tasks. These works report better accuracy results compared with traditional approaches such as $k$ Nearest Neighbors (kNN), Support Vector Machines (SVM), Hidden Markov Models (HMM), decision trees algorithms such as C4.5, logistic regression, etc. A more recent work [63] shows good results against classical algorithms but relies on a preprocessing stage to extract a set of features from sensor raw data.

Convolutional neural networks [79] have achieved state-of-the-art results in image recognition tasks, where the nearby pixels typically have strong relationships with each other. Similarly, in the human activity recognition realm, adjacent sensor readings are likely to be correlated. In multi-modal approaches, where more than one sensor are used to characterize an activity, correlations among distinct types of sensors may also have an impact on the correct interpretation of data. The CNN approach applied to HAR classification tasks has been proved to outperform previous state-of-the-art methods such as dynamic time warping (DTW), hidden Markov models (HMM), and support vector machines (SVM), among others. In one of the first works [42] to use CNN on raw sensor time-series data for gesture recognition, researchers obtained better results than classical methods, outperforming another deep learning model such as bidirectional LSTM. In [176], researchers used a single convolutional layer architecture to classify accelerometer data, giving some insights on the best values for some parameters, such as the sampling layer pooling size, or the dropout and weight decay values to prevent over-fitting and improve generalization. Later works, such as [173, 33, 72, 132, 175, 32, 106] use more complex CNN architectures, with three or more convolutional layers, and explore the influence that the number of layers and other parameters have on the classification performance. These experiments show that increasing the number of convolutional layers increases the performance of the model. The main conclusion to be drawn from the aforementioned works is that CNNs are able to capture local dependencies, both among sensor axes and among different sensors, and to build powerful features upon them that result in a performance boost from previous non deep learning methods. These results confirm that, although this architecture has been used prominently for computer vision tasks, it is also relevant for sequence processing tasks, with the additional benefit of their relatively cheaper computational cost.

Some recent works [82, 112, 55, 109] explore the performance of recurrent neural networks [135] on HAR classification tasks. This architecture has been specially conceived to work with sequential data, and thus, seems a good choice for HAR, since the input data is typically composed of a series of sensor readings. More advanced RNN architectures such as long short term memory (LSTM) units [70] and gated recurrent unit (GRU) [36] have eased the use of this type of neural networks, overcoming some important issues with the original RNN architecture. One of the first works [82] used bidirectional LSTM (biLSTM) to exploit the fact that activity recognition may depend on past and future context. The authors used a concatenation of accelerometer and gyroscope information synchronized in time, with no further feature engineering except the normalization of values between -1 and +1, as input to a biLSTM architecture. Authors in [112] used a combination of LSTM and CNN to perform activity recognition tasks on sensor data acquired from wearable devices. Their model outperformed previous results, including a

CNN baseline, using the *opportunity* and *skoda* benchmark datasets. The use of ensembles of LSTMs [55] has been explored, with better classification results than previous deep learning models, including [112]. Finally, [109] compared the performance of various LSTM models against CNN, sequential extreme machine learning (ELM), and SVM on five benchmark datasets, obtaining good results and demonstrating the suitability of RNN architectures for HAR.

As far as we know, there is only one work that compares the performance of different deep learning architectures on HAR. In their paper, Hammerla et. al. [59] measure the effectiveness of FFNs, CNNs, and two types of LSTM networks on three datasets, analyzing the influence of hyper-parameters on the performance of these algorithms, using accelerometer data as input. The conclusions of their work showed that CNN models obtained the best results for repetitive activities, while LSTM achieved better results on the *opportunity* dataset.

Table 3.1 shows a summary of previous works exploring deep learning effectiveness on HAR. The table shows the number of datasets used to asses the performance of the distinct algorithms, the deep learning architecture considered, and the typology of sensors used to classify the activities. The vast majority of previous studies use at least the accelerometer data, but many of them use it in conjunction with the gyroscope data since the use of both sensors produces better accuracy [169, 146]. Other elements such as magnetometers, light sensors, radio frequency identification (RFID), barometers, temperature sensors, or WiFi readings, are also used in some papers.

From the aforementioned works some conclusions can be drawn:

- Deep learning models that take into account local dependencies to build descriptive features, such as CNN and RNN, seem more suitable for HAR than fully connected models, since the data used as input for activity recognition tasks consist of time series from sensor readings. On the contrary, fully connected network architectures such as DBN assume that inputs are independent of each other, so in order to model a time series it is necessary some previous feature engineering to include temporal information in the input data.

- Both RNNs and CNNs provide state-of-the-art performance. Although their architectures can be over-engineered, and even combined, to boost their classification scores, it remains unclear which type is more adequate for

Table 3.1: Summary of previous works. Architectures are: deep belief networks (DBN), restricted Boltzmann machines (RBM), convolutional neural networks (CNN), recurrent neural networks (RNN), long short term memory networks (LSTM), gated recurrent unit networks (GRU) and deep feed-forward networks (FFN). Sensors are: accelerometer (acc), gyroscope (gyr) and other types such as magnetometer, barometer, light, temperature, WiFi, etc (other).

| **Authors and year** | **Architectures** | **Datasets** | **Sensors** |
|---|---|---|---|
| Plötz et al. (2011) [116] | DBN | 3 | acc |
| Lefebvre et al. (2013) [82] | LSTM | 1 | acc, gyr |
| Gjoreski et al. (2016) [51] | CNN | 2 | acc |
| Zeng et al. (2014) [176] | CNN | 3 | acc |
| Duffner et al. (2014) [42] | CNN | 1 | acc, gyr |
| Yang et al. 2015 [173] | CNN | 2 | acc, gyr, other |
| Chen & Xue (2015) [33] | CNN | 1 | acc |
| Jiang & Yin (2015) [72] | CNN | 3 | acc, gyr |
| Zhang et al. (2015) [177] | DBN | 3 | acc |
| Ha et al. (2015) [56] | CNN | 2 | acc, gyr |
| Alsheikh et al. (2016) [4] | DBN | 3 | acc |
| Bhattacharya & Lane (2016) [25] | RBM | 3 | acc, gyr, other |
| Ordóñez & Rogen (2016) [112] | LSTM+CNN | 2 | acc, gyr, other |
| Hammerla et al. (2016) [59] | FFN, LSTM, CNN | 3 | acc |
| Radu et al. (2016) [123] | RBM | 1 | acc, gyr |
| Ronao & Cho (2016) [132] | CNN | 1 | acc, gyr |
| Zebin et al. (2016) [175] | CNN | 1 | acc, gyr |
| Ravi et al. (2016) [125] | CNN | 4 | acc, gyr |
| Guan & Plötz (2017) [55] | LSTM | 3 | acc, gyr, other |
| Murad & Pyun (2017) [109] | LSTM | 5 | acc, gyr, other |
| Camps et al. (2018) [32] | CNN | 1 | acc, gyr, other |
| Moya et al. (2018) [106] | CNN | 3 | acc, gyr |

HAR tasks, both in terms of performance and resource consumption.

## 3.3 Deep Learning architectures

One of the core advantages of deep learning is its ability to automatically learn features from raw data. Previously proposed schemes for HAR remained in the conventional supervised learning paradigm that relies on the design of handcrafted features. Although these schemes could achieve high accuracy, the requirement for domain knowledge limits its scalability. Finding a good set of features from the raw data is crucial to isolate key information and highlight important patterns, but it requires expert knowledge and it is difficult and time-consuming. Deep learning eliminates the need for manual feature engineering.

There are several different architectures for deep neural networks (DNN). In general, they can be grouped into three main categories: Feed-Forward Networks, Convolutional Neural Networks and Recurrent Neural Networks. Their main characteristics are succinctly described in the next sections.

### 3.3.1 Deep Belief Networks

A deep belief network is composed of several fully connected layers. Its structure is essentially the same as for a multi-layer perceptron (MLP), where the only significative difference relies on the pretraining process. DBNs are formed by stacking restricted Boltzmann machines. RBMs are fully connected shallow neural networks composed by an input layer and a hidden layer, in which all the units are binary and stochastic. In an RBM, the visible units represent the observations and are connected to the hidden units using weighted connections. The nodes of any single layer do not communicate with each other laterally.

Restricted Boltzmann machines are symmetrical bipartite graphs. Their training process consists in learning to reconstruct data by themselves in an unsupervised approach, making several forward and backward passes between the visible and hidden layers. In the forward pass, the activations represent the probability of an output given a weighted input $x$: $p(a|x; w)$. In the backward pass, the result is an estimation of the probability of inputs $x$ given the weighted activations $a$: $p(x|a; w)$. These two estimates lead to the joint probability distribution of inputs $x$ and activations $a$: $p(x, a)$.

The training strategy for RBMs involves the concept of activation energy $E_i$. If there are $n$ visible units and $m$ hidden units, we can express the states of the visible and hidden layers with vectors $v$ and $h$. For a given state $(v, h)$ the energy in the RBM is:

$$E(v,h) = -\sum_{i=1}^{n} a_i v_i - \sum_{j=1}^{m} b_j h_j - \sum_{i=1}^{n}\sum_{j=1}^{m} v_i w_{ij} h_j \tag{3.1}$$

where $a_i$ is the bias of the $i^{th}$ visible unit, $b_j$ is the bias of the $j^{th}$ hidden unit, and $w_{ij}$ express the weight between the visible unit $i$ and the hidden unit $j$.

The marginal probability of the units, namely, the probability that the given weights will generate the visible units $v$ is:

$$P(v) = \frac{1}{\sum_{v,h} e^{-E(v,h)}} \sum_{h} e^{-E(v,h)} \tag{3.2}$$

The training process works by updating the weights using the contrastive divergence (CD) method to calculate the gradients. This method approximates the gradients of the log-likelihood based on a short Markov chain started at the last example seen. Each time CD is run, it's a sample of the Markov Chain composing the restricted Boltzmann machine. The weights are updated following the rule:

$$\Delta w_{ij} = \epsilon(< v_i.h_j >_{data} - < v_i.h_j >_{recon}) \tag{3.3}$$

where $\epsilon$ is the learning rate, $< v_i.h_j >_{data}$ represents the expectation of the observed data and the results of the weights in the training set, and $< v_i.h_j >_{recon}$ is the reconstruction distribution of the model.

Once all the RBMs of a DBN have been trained, the generative pretraining stage is finished. Their weights are used as the initial weights of the DBN. The generative pretraining process helps the discriminative training of the model to achieve better generalization solutions [43]. This stack of RBMs might end with a Softmax layer to create a classifier, or it may simply help cluster unlabeled data in an unsupervised learning scenario.

DBNs can be used to classify human activities from raw sensor data by pretraining the network layers individually in an unsupervised way and then fine-tunning the complete network with the backpropagation algorithm. In general, these architectures outperformed classical machine learning approaches, but today they are mostly out of favor and rarely used [54], at least for HAR classification tasks using raw inertial data. There is an inherent difficulty for this type of architecture to learn discriminative patterns from time-series data since it has separate parameters for each input feature. This forces the model to learn all the rules that characterize an activity separately at each position in the input, or in other words, at each time step. As an example, since a step detection can occur at any time step in the input data, in a feed-forward network architecture the detection of the step has to be learned for each position in the input layer. By comparison, recurrent neural networks share the same weights across several time steps, and the pooling layers of convolutional neural network architectures provide partial invariance to small local translations. These characteristics make them more appropriate for time series classification.

### 3.3.2 Convolutional Neural Networks

Convolutional neural network architecture is inspired by the hierarchical structure of the human visual cortex, which processes the visual information coming from the eye through a series of ordered and inter-connected visual areas that perform feature recognition, from simple edge detection in the first areas to more complex shape structures in the higher levels [78]. CNNs have gained popularity due to their ability to learn suitable representations and capture local dependencies from images or temporal series. In the last few years, the use of deep CNN models has led to very good performance on a variety of problems, such as visual and speech recognition. Human activity recognition is also a good field for convolutional architectures, especially when considering the translation invariance and temporally correlated readings of time-series signals from activities, and their hierarchical structure as a combination of small movements. Due to this potential to identify the representative patterns of HAR's signals, CNNs have recently been applied to human activity recognition in many research papers.

The operation performed by a convolutional layer consists of an element-wise product followed by a sum. The input, as a 2D matrix, is convoluted with a learnable kernel, a 2D matrix of a particular size, in a sliding-window fashion. The result of this operation forms the output feature map, which is another 2D matrix. Note that more than one kernel can be applied to the input, hence the output will be composed of as many feature maps as kernels are used. Different kernels will perform different convolution operations on the inputs, such as edge detection or sharpening. Each kernel can be considered as a specific feature detector, so the key task when training a convolutional neural network is to get it to learn the best kernels, those that extract the most meaningful features from the input. The convolution layer operation for a two-dimensional input can be expressed as:

$$S(i,j) = (K * I)(i,j) = \sum_{m}\sum_{n} I(i+m, j+n)K(m,n) \tag{3.4}$$

where $K$ represents the kernel and $I$ is the input of size $mxn$.

After the convolution, a nonlinear activation function is applied to enable a nonlinear transformation of the data. Then, a pooling layer is used to subsample the data, by sampling one input out of every region it looks into. The most commonly used subsampling strategy is max-pooling (taking the maximum value of the input), but other strategies can be considered, such as average-pooling (taking the average value of the input) or probabilistic pooling (taking a random value from the input). Besides turning the input into a smaller representation of the original data, the pooling layer makes the model invariant to small translations of the input data. Therefore, the pooling layer does not do any learning but introduces sparseness as well as translation invariance. Since it only considers the maximum or the average value in a local neighborhood, a small distortion in the input will not change the result of pooling.

Convolutional and pooling layers are the building blocks of convolutional neural networks. Many convolutional and pooling layers can be stacked to form a deep neural network architecture. These layers act as a hierarchical feature extractor, each one building feature detectors over the outputs of the previous layer. The lower layers obtain the local influence of the signals (for example, the characteristics of each basic movement in human activity), while the higher layers obtain a high-level representation features and patterns (for example, combinations of several basic movements). CNN can exploit the local dependency characteristics inherent in time-series sensor data and the translation-invariant nature of activities.

### 3.3.3 Recurrent Neural Networks.

Recurrent neural networks [135] are a family of neural networks specialized in processing a sequence of values. They have the ability to capture long-distance

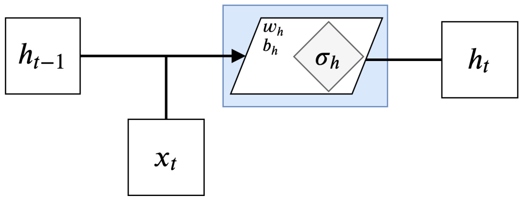


Figure 3.1: Vanilla recurrent neural network cell scheme.

dependencies in the input stream, or, in other words, to remember information about previous inputs. Recurrent neural networks have one or more cycles, so it is possible to follow a path from a unit back to itself [73]. These cycles make it possible for RNNs to model long-distance dependencies, as they can pass information among time steps. When unfolded in time, an RNN can be seen as a feed-forward multilayer neural network with all layers sharing the same weights. The deepness of this unrolled network can be potentially infinite, as each layer represents a step in time. Unrolling the network allows us to obtain a standard computation graph on which to apply forward computation and backward propagation, or backpropagation through time (BPTT) [130, 164, 107], to learn the parameters of the network. An RNN receives a sequence as input, which is processed in an internal loop over its elements (each sequence time step). In each step of the loop, the internal state of the RNN, a sort of 'memory' of previous time steps, is combined with the current time step input to produce an output. This operation can be expressed as follows:

$$h_t = \sigma_h(w_h[h_{t-1}, x_t] + b_h) \tag{3.5}$$

where $w_h$ represents the weights of the cell, $x_t$ is the input at current timestamp $t$, $\sigma_h$ is the activation function, and $h_t$ and $h_{t-1}$ are the outputs (states) of the cell at current and previous time steps, respectively. Figure 3.1 shows a scheme of the inputs, outputs and operations that conform a RNN cell.

When a sequence has been processed, the internal state of the network is initialized to zero before processing the next sequence. While processing the input data, the recurrent layers generate an output for each time step in the sequence. Since each output is based on the current and all previous time steps, the last output will provide the most accurate prediction. This last output can be connected to a soft-max layer to perform multi-class logistic regression, producing a probability distribution over the activity class labels.

Vanilla RNN architecture introduces some critical issues, like the vanishing gradient and the exploding gradient problems [24], which makes optimization a great challenge. The vanishing gradient problem states that, given that for traditional activation functions the gradient is bounded, when these gradients are computed by backpropagation through the chain rule, the error signal decreases exponentially within time steps. This makes it hard for RNN to account for long-term dependencies since the weights will not be updated beyond a few time steps. Depending on the activation function and the parameters used in the network, the problem can be the opposite, and the gradients can grow exponentially. The exploding gradient problem can be easily addressed by clipping the value of gradients to a predefined threshold [114], but the vanishing gradient is more problematic to correct. Regularization, the use of ReLU as the activation function, and proper initialization of the weights can reduce the effect of the problem.

Some alternative RNN architectures have been developed to address such issues. Long short term memory networks [70] and gated recurrent unit networks [36] are known to be good solutions to bypass the vanishing/exploding gradient problem and efficiently learn long-range dependencies. LSTMs are the most widely RNN architectures used today to process sequential inputs like speech and language. The operation of an LSTM cell can be described as follows:

$$\begin{aligned}
i_t &= \sigma(w_i[h_{t-1}, x_t] + b_i) \\
f_t &= \sigma(w_f[h_{t-1}, x_t] + b_f) \\
o_t &= \sigma(w_o[h_{t-1}, x_t] + b_o) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \tanh(w_c[h_{t-1}, x_t] + b_c) \\
h_t &= o_t \circ \tanh(c^t)
\end{aligned} \tag{3.6}$$

where $i_t$, $f_t$, $o_t$ are the output of the input, forget and output gates, respectively, $w_i$, $w_f$, $w_o$ are the weights for the input, forget and output gates, respectively, $w_c$ are the weights for the candidate layer, $b_i$, $b_f$, $b_o$ are the biases for the input, forget and output gates, respectively, $b_c$ is the bias for the candidate layer, $\sigma$ is the sigmoid function, $h_{t-1}$ is the output for the previous time step $t-1$, $x_t$ is the input at current time step $t$, $c_{t-1}$, $c_t$ are the cell states at time steps $t-1$ and $t$, $h_t$ is the output of the cell at current time step $t$, and $\circ$ is the Hadamard product. Figure 3.2 shows a scheme of the inputs, outputs and operations that conform an LSTM cell.

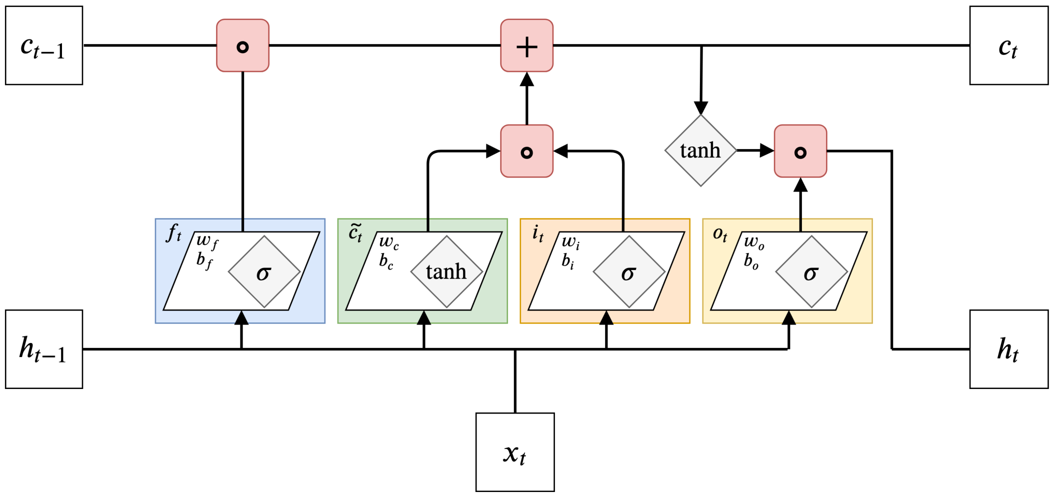


Figure 3.2: Long short-term memory cell scheme. Symbol $\circ$ represents the element-wise (Hadamard) product.

GRUs are simplified versions of LSTMs. Compared to LSTM, GRUs have fewer parameters and are usually used to conserve memory or computation time. The main difference with LSTMs is that a single gating unit simultaneously controls the forgetting factor and the decision to update the state unit. The operation of a GRU cell can be described as follows:

$$\begin{aligned} z_t &= \sigma(w_z[h_{t-1}, x_t] + b_z) \\ r_t &= \sigma(w_r[h_{t-1}, x_t] + b_r) \\ h_t &= z_t \circ h_{t-1} + (1 - z_t) \circ \tanh(w_h[r_t \circ h_{t-1}, x_t] + b_h) \end{aligned} \tag{3.7}$$

where $z_t$, $r_t$ are the output of the update and reset gates, respectively, $w_z$, $w_r$ are the weights for the update and reset gates, respectively, $b_z$, $b_r$ are the biases for the update and reset gates, respectively, $\sigma$ is the sigmoid function, $h_{t-1}$ is the output for the previous time step $t-1$, $x_t$ is the input at current time step $t$, $h_t$ is the output of the cell at current time step $t$, and $\circ$ is the Hadamard product. Figure 3.3 shows a scheme of the inputs, outputs and operations that conform a GRU cell.

## 3.4 Datasets

Focusing on a multi-modal approach by using a fusion of accelerometer and gyroscope data may be useful in accuracy-sensitive applications with a complex

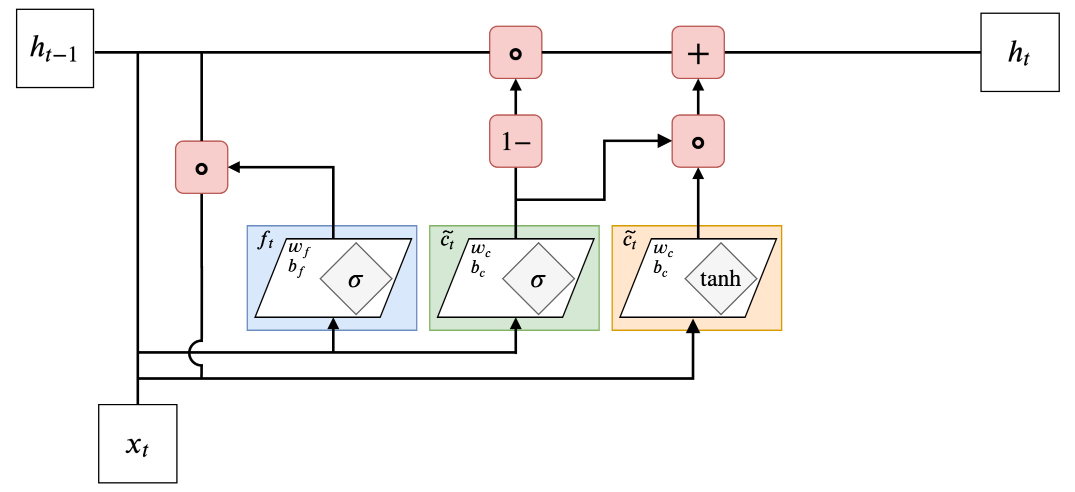


Figure 3.3: Gated recurrent unit cell scheme. Symbol $\circ$ represents the element-wise (Hadamard) product.

activity set, and allows for taking into account not only temporal dependencies but also possible dependencies among sensors.

To conduct the experiments to assess the performance of different RNN and CNN architectures, we used a total of ten publicly available datasets that have been widely used within the community. These datasets contain continuous sensor readings from inertial measurement units (accelerometers and gyroscopes) worn by participants of the particular studies at different positions on their bodies while performing typical tasks for human activity recognition. Some are realistic benchmark datasets, such as the *opportunity* dataset, while others might not be directly mirroring real-world situations but still are widely used in the research community, such as the *pamap2* dataset. The number of activities considered in each dataset, as well as the class balancing, show substantial variability among them, as shown in Table 3.2. This is usual for mobile application scenarios where there is a preeminence of static activities, such as sitting or standing, over dynamic activities such as walking or running. Examples of this scenario are health assessments, where problematic activities or behaviors are the rare exceptions within other more common activities.

For all the datasets we consider only recordings from on-body accelerometers and gyroscopes. We created 100 samples-wide sliding windows from the raw data, with 50% overlapping. Even though different overlap values can be used, an overlap of 50% has been shown to produce reasonable results [18, 120, 169]. The

Table 3.2: Number of datapoints per dataset. Columns 1-12 represent each activity.

| dataset | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | total |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| activemiles | 13648 | 7020 | 1288 | 784 | 16304 | 3408 | 8404 | - | - | - | - | - | 50856 |
| hhar | 39226 | 41936 | 39237 | 45514 | 37464 | 33873 | - | - | - | - | - | - | 237250 |
| swell | 2348 | 2348 | 2473 | 1455 | 1712 | 2264 | - | - | - | - | - | - | 12600 |
| fusion | 8950 | 8950 | 8950 | 8950 | 8950 | 8055 | 8950 | - | - | - | - | - | 61755 |
| usc-had | 3744 | 2518 | 2685 | 2048 | 1904 | 1695 | 1000 | 2545 | 2290 | 3680 | 1552 | 1552 | 27207 |
| mhealth | 1200 | 1200 | 1200 | 1200 | 1200 | 1106 | 1148 | 1146 | 1200 | 1200 | 1200 | 384 | 13384 |
| uci-har | 2257 | 2059 | 1880 | 2359 | 2582 | 2567 | - | - | - | - | - | - | 13704 |
| pamap2 | 22548 | 21634 | 22126 | 26378 | 11110 | 19208 | 21610 | 13638 | 12210 | 20560 | 27886 | 5424 | 224332 |
| opportunity | 28481 | 8116 | 16972 | 2945 | - | - | - | - | - | - | - | - | 56514 |
| realworld | 19375 | 22654 | 4695 | 31147 | 34777 | 27015 | 28987 | 27304 | - | - | - | - | 195954 |

time span of the window is fixed for each dataset and depends on the sampling rate of the raw data obtained from the sensors. For a rate of 50Hz, a window will contain 2 seconds of sensory data.

The main characteristics of the datasets considered in this experiment are as follows:

- **Activemiles** [126]: This dataset contains around 30 hours of labeled raw data from real-world human activities performed by 10 subjects. The data were collected using five distinct smartphones with independent device configurations, sensor brands, and sampling rates. No limitations were put on where the device was located (i.e., pocket, bag, or held in the hand). Annotations record the start time, end time, and a label describing the activity. For this work, both accelerometer and gyroscope raw data have been downsampled to 50Hz.

- **Fusion** [146]: The Fusion dataset collected data for seven basic motion activities in daily life (walking, running, sitting, standing, jogging, biking, walking upstairs and walking downstairs) from ten participants. Every subject performed each activity for 3–4 min, equipped with five smartphones on five body positions (left and right jeans pockets, right upper arm, right wrist, and right leg belt position). The same model of smartphone was used for all the positions, with different orientations depending on the device location. The data recorded included the readings from the accelerometer, the gyroscope, and the magnetometer, all collected at a sampling rate of 50Hz.

- **hapt (UCI HAR)** [6]: The data in this dataset was recorded by 30 volunteer subjects who performed six different activities while wearing a waist-mounted smartphone. The subjects performed a protocol composed of six basic activities: three static postures (standing, sitting, lying) and three dynamic activities (walking, walking downstairs, and walking upstairs). The raw accelerometer and gyroscope tri-axial signals were sampled at a rate of 50 Hz.

- **HHAR** [151]: The Heterogeneity dataset for Human Activity Recognition contains the readings of accelerometer and gyroscope sensors recorded while users executed activities carrying smartwatches and smartphones. The signals were sampled at the highest frequency the respective device allowed. A total of six different activities (biking, sitting, standing, walking, stairs up and stairs down) were performed by 9 subjects carrying 4 smartwatches, 2 on each arm, and 8 smartphones, all placed around the user's waist. Each participant conducted five minutes of each activity.

- **MHealth** [17]: Mobile Health dataset recorded 12 daily activities from 10 volunteers of diverse profiles. This data was acquired by means of four different types of sensors: 3 tri-axial accelerometer sensors, 2 tri-axial gyroscope sensors, 2 magnetometer sensors, and 1 two-lead Electrocardiogram sensor, at a sampling rate of 50Hz. The sensors were placed on the subject's chest, right wrist, and left ankle of the subjects.

- **Opportunity** [131]: The Opportunity Activity Recognition dataset consists of annotated recordings from a total of 4 participants, who wore 7 IMUs and 12 accelerometers placed on various body parts, and were instructed to carry out 18 different Activities of Daily Living (ADL), specifically focusing on kitchen routine. The sampling frequency for all IMUs was 30Hz. Every participant performed five different runs of the activities, following a loose description of the overall actions to perform. We only use the 4 activities from this dataset which are similar to the activities included in the rest of datasets; Stand, Walk, Sit and Lie, taking into account only the on-body sensors.

- **Pamap2** [128, 127]: The PAMAP2 Physical Activity Monitoring dataset contains data from 18 different physical activities, performed by 9 subjects wearing 3 inertial measurement units, attached to the hand, chest, and ankle, and a heart rate monitor. The participants were instructed to carry out a total of 12 activities of daily living, plus 6 optional activities, covering domestic routines and various sportive exercises (Nordic walking, running, etc). The data was recorded from inertial measurement units located on the hand, chest, and ankle of the participants, over a total of 10 hours.

- **Swell** [147]: This dataset contains raw accelerometer, magnetometer, and gyroscope signals acquired by means of four smartphones placed on four body positions: right jeans pocket, belt, right arm, and right wrist. It recorded six different physical activities, performed by four participants, at a sampling rate of 50Hz. The activities that the subjects conducted are walking, running, sitting, standing, walking upstairs, and walking downstairs. The dataset contains roughly 3-5 minutes of data per activity and participant.

- **USC-HAD** [178]: The USC human activity dataset is intended as a benchmark for algorithm comparison, particularly for health-care scenarios, and consists of 12 basic human daily activities: walking forward, walking left, walking right, walking upstairs, walking downstairs, running forward, jumping up, sitting, standing, sleeping, in elevator up, and in elevator down. The dataset was recorded by 14 subjects wearing a high-performance inertial sensor device, with a tri-axial accelerometer and gyroscope, located on the front right hip, at a sampling rate of 100Hz.

- **RealWorld** [152]: Realworld is a complete and realistic dataset that covers seven activities (climbing stairs down and up, jumping, lying, standing, sitting, running/jogging, and walking) performed by fifteen subjects. Each subject performed each activity roughly 10 minutes except for jumping ($\sim 1.7$ minutes). The dataset was recorded using seven wearable devices (smartphones and smartwatches) positioned on chest, forearm, head, shin, thigh, upper arm, and waist. The devices were attached to the relevant body positions using a sports armband case, trouser pocket, shirt pocket, head belt, or the bra. There was no further fixation of the device to closely resemble their use in everyday life. The sampling frequency for all sensors was 50Hz.

## 3.5 Experiments setup

The purpose of this stage is to obtain labeled segments from the continuous stream of data stored in each of the ten datasets used to evaluate the different deep learning architectures. These segments are sliding windows of sensor measurements, containing 100 consecutive readings from both sensors, accelerometer, and gyroscope, for a particular activity, and with an overlapping factor of 50%. The time span of the window depends on the sampling rate at which the data was recorded. While for the *opportunity* dataset, whose data was acquired at 30Hz, a window represents 3.33 seconds of a particular activity, for the USC-HAD dataset, recorded at 100Hz, a window is 1 second wide. Some authors add the magnitude of measurements (the norm of the three-axis) as an additional input to reduce the effect of sensor placement and orientation. This may increase the performance, but also the computational requirements and the training time. Since this experiment is intended to compare the performance of different architectures on the same data, we keep things simple and use only standardized raw data as input, with no further engineering.

Given the variety of sensors used in the benchmark datasets, with different typologies and brands, and thus distinct sensitivity and measurement ranges, it is convenient to scale the data to speed-up the convergence of DL algorithms. We performed preliminary tests to determine the best option among linear scaling in the range 0 to 1, linear scaling in the range -1 to +1, and standardizing to 0 mean and 1 standard deviation. The latter option showed the best results, with lower average convergence times for all architectures except DBN. Therefore, once the windows are extracted from the dataset, the values are standardized to have 0 mean and 1 standard deviation. In the case of DBN, to be able to work with real-valued data we use Gaussian distributed visible neurons and Bernoulli distributed hidden neurons in the first RBM. Consequently, input data has to be scaled in the range of 0 to 1.

Figure 3.4 shows the resultant standardized distributions of accelerometer and gyroscope values for each dataset. The gyroscope distributions look similar, with the majority of values around zero, except for the *mhealth* dataset, which shows great variability. The main reason for this might be the

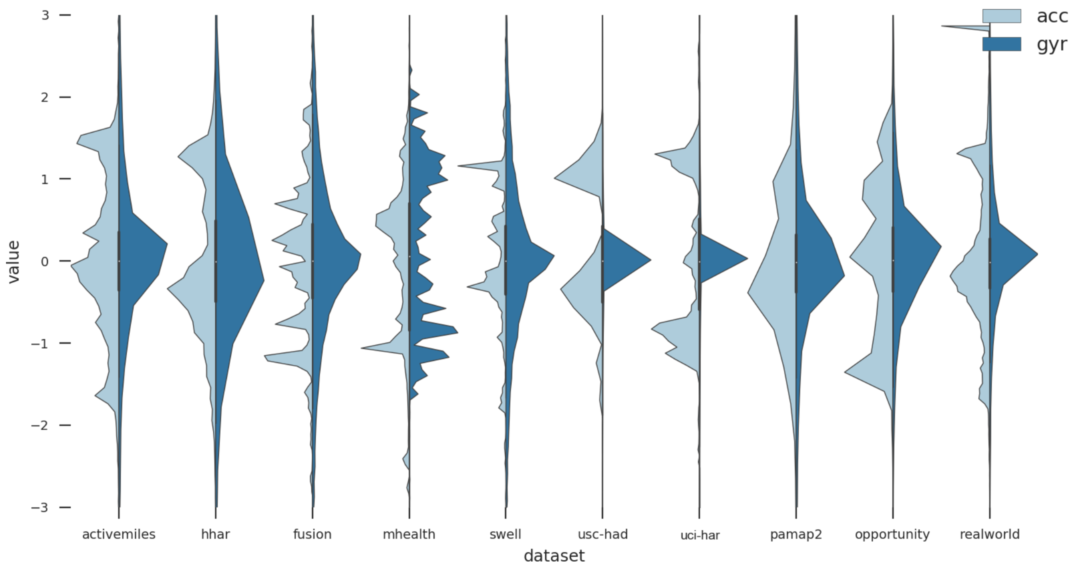


Figure 3.4: Distributions of accelerometer and gyroscope signals.

sensor used, a Shimmer3 IMU with an *Invensense MPU9150* gyroscope, with low RMS noise and good range, sensitivity, and resolution, in contrast with low-cost smartphone's sensors used in other datasets, combined with the small size of the dataset and the big number of activities (12) recorded. The accelerometer distributions present more diversity among datasets, with many peaks spread along with the range of values.

### 3.5.1 Experiment setup

Apart from standardizing the raw data and concatenating the accelerometer and gyroscope data together, sensor signals are used without employing any prior feature extraction methods, which is in line with the majority of recent deep learning-based analysis methods in HAR and with the objective of this work to determine the capacity of various DL architectures to find convenient features for recognition of human activities. All the models, except for the DBN architectures, have been designed to receive as input the same data. Each data point has a dimensionality of 100x6, which corresponds to 100 timesteps with 6 readings each, one for axis and sensor. In the case of DBNs, the input is reshaped to form a vector of 600 values. The composition of the different models has been designed taking into account previous works, but with the goal of using only simple architectures that are representative of each DL algorithm, to be able to compare the performance of each different structure without over-engineering the design to obtain state-of-the-art results. The guidelines when designing the models for each of the different deep learning architectures considered are as follows:

– DBN: The sizes of the models range from 1 to 6 RBM layers. In previous works [177, 4], researchers used layer sizes ranging from 256 to 2000 hidden units. We fixed the number of hidden units to an intermediate value of 512, as it is showed in Table 3.3. Once pretrained, RBMs are stacked forming a DBN that is then fine-tuned in a supervised way using backpropagation algorithm, with the output of the last RBM connected directly to a soft-max layer for classification.

– CNN: The models are composed of three 2D convolutional layers (with convolutional and pooling stages) connected to a fully connected neural network and a soft-max classifier. This three-layer convolutional architecture has been used in previous works, such as [173, 33, 132], with good results for human activity recognition tasks. The filter sizes of the first layer range from 19x3 to 3x3. The models are designed to study the influence of the temporal size of the filter (the number of time steps it encompasses) on the performance of the model. On the axis dimension of the input, the filter of the first convolutional layer encloses the three axes of each sensor, with a stride factor of 3. Consequently, this layer will not take into account any dependency among sensors, but only dependencies among axis of the same sensor. It is not until the last convolutional layer that local dependencies between accelerometer and gyroscope are taken into account. Table 3.4 shows the configuration parameters for each CNN model. The last fully connected layer is sent directly to a soft-max layer for classification.

– LSTM, biLSTM, and GRU: We use a similar architecture for all RNN models. The models are three layers deep and the number of hidden units ranges from 100 up to 600 for LSTM and GRU models, and from 100 to 500 for biLSTM models, as it is showed in Table 3.5. Previous works [82, 109] used a similar architecture, with 100 hidden units per layer. Other authors use more complex structures involving the use of

Table 3.3: DBN models. Table shows the number of hidden layers, the number of units in each layer, and the number of trainable parameters of the model.

| number of layers | hidden units per layer | number of parameters |
|---|---|---|
| 1 | 512 | 312320 |
| 2 | 512 | 620032 |
| 3 | 512 | 927744 |
| 4 | 512 | 1235456 |
| 5 | 512 | 1543168 |
| 6 | 512 | 1850880 |

mixed architectures [55] such as CNNs combined with LSTMs. The last prediction of the last recurrent layer is sent directly to a soft-max layer for classification.

In order to avoid potential over-fitting, and also to reduce the influence that a particular partition of data could have on the final result, we evaluated the performance of each model using a random 5-fold stratified cross-validation strategy. Prior to the training process of models, a test set, comprising 20% of the total data points, is extracted from each dataset. This set will be later used to assess the performance of the trained model. The remaining data is randomly divided into 5 equally sized and mutually exclusive folds. On each iteration, one fold is used as validation, and the remaining are used as training data. Once all the validation folds have been used, the performance is averaged among the five results obtained on the test set.

To further improve the training procedure and generalization performance, the following regularization techniques have been used:

– Dropout [150]: It is a regularization strategy that removes non-output units randomly from the original network, independently for each hidden unit and for each training case. Thus, applying dropout to a neural network is equal to sub-sampling a smaller and less complex neural network from it. Dropout was performed in all fully connected layers and in all recurrent layers, with a probability value of 0.5.

– Weight decay [115]: It is well known that more sparse neural networks

Table 3.4: CNN models. Table shows the size of the filter for the first convolutional layer of the models, the number of units in the fully connected layers (fcl1 and fcl2), and the number of trainable parameters.

| first filter size | fully connected layer 1 | fully connected layer 2 | number of parameters |
|---|---|---|---|
| 19x3 | 1536 | 100 | 1144676 |
| 17x3 | 1536 | 100 | 1176676 |
| 15x3 | 1536 | 100 | 1274212 |
| 13x3 | 1536 | 100 | 1306212 |
| 11x3 | 1536 | 100 | 748388 |
| 9x3 | 1536 | 100 | 845924 |
| 7x3 | 1536 | 100 | 943460 |
| 5x3 | 1792 | 100 | 673380 |
| 3x3 | 2816 | 100 | 742244 |

Table 3.5: LSTM, biLSTM and GRU models. All the models are composed of three recurrent layers and a soft-max output layer. Table shows the number of hidden units in each layer, and the number of trainable parameters of the model.

| hidden units per layer | LSTM parameters | biLSTM parameters | GRU parameters |
|---|---|---|---|
| 100 | 128400 | 256800 | 96300 |
| 200 | 496800 | 993600 | 372600 |
| 250 | 771000 | 1542000 | 578250 |
| 300 | 1105200 | 2210400 | 828900 |
| 350 | 1499400 | 2998800 | 1124550 |
| 400 | 1953600 | 3907200 | 1465200 |
| 450 | 2467800 | 4935600 | 1850850 |
| 500 | 3042000 | 6084000 | 2281500 |
| 550 | 3676200 | – | 2757150 |
| 600 | 4370400 | – | 3277800 |

generalize better. Weight decay penalizes large weights and limits the freedom of the model by adding an additional term in the weight update rule, forcing weights to be closer to zero. Based on previous works, such as [132], we chose a value of 0.00005 for weight decay.

- Early stopping: It is a simple technique that can be superior to other regularization methods in many cases, e.g. [45]. The validation error is used as an estimate of the generalization error, i.e., the validation set is used to anticipate the behavior on the test set, assuming that the error on both will be similar. The method consists in training the model on the training set and evaluating its performance every epoch on the validation set. Given an early stopping parameter $k$, the training process stops when $k$ epochs have passed without any performance increase. The final performance metric is evaluated on the test set using the model state that performed better on the evaluation set.

In the training process we use the AdaGrad [41] updating function, with a learning rate of 0.005, and a mini-batch size of 1000 data points, except for the most complex recurrent models, on which we used a mini-batch size of 500. These parameters have been determined based on previous experiments and precedent research works on deep learning and HAR. We also used the cross-entropy as the loss function. As our evaluation metrics, we employ both the accuracy and the f1-score, which is defined as the harmonic mean of precision and recall:

$$f_1 = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

Since more than two classes are considered, we report the weighted average of the individual f1-scores of all classes as the evaluation metric for each model, in line with standard practices in HAR research [26], since it is more resilient than accuracy on imbalanced datasets:

$$\hat{f}_1 = \frac{\sum_{i=1}^{c} w_i f_1^i}{\sum_{i=1}^{c} w_i}$$

where $c$ is the number of classes and $w_i$ is the weight (the number of instances) of the $i^{th}$ class.

## 3.6 Results and discussion

### 3.6.1 Results

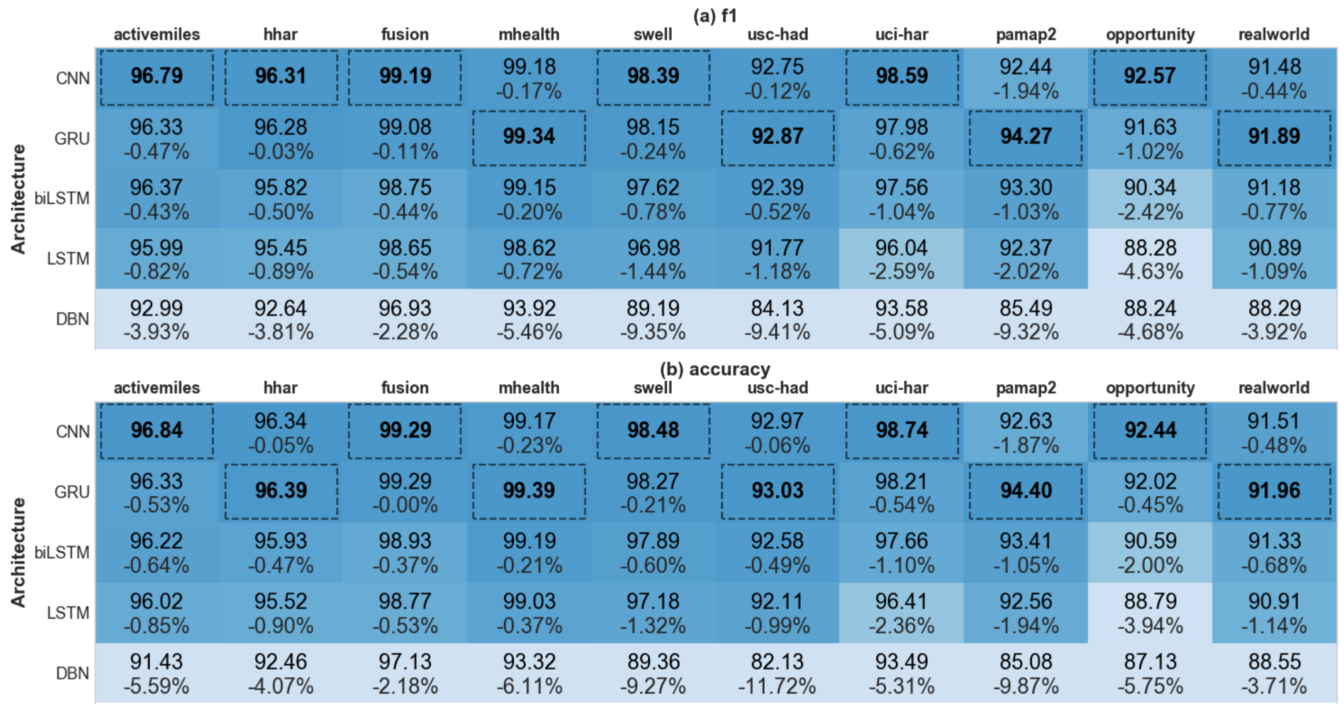
The results obtained are summarized in Figure 3.5, which shows the best f1 and accuracy scores for each architecture and dataset. Using both metrics, CNN performed better approximately if half of the datasets considered, while GRU performed better on the rest. In all the experiments, DBN was the worst-performing architecture.

Regarding the absolute scores obtained, one thing to note is that those datasets whose signals showed more variability (see Figure 3.4), such as *mhealth*, *fusion*, *hapt* or *swell*, achieved better general classification scores than the rest, rounding a maximum score of 99%. This highlights the important function that variability plays, making it easier for the models to find meaningful correlations among sensor signals, and among those and the correspondent activity, thus easing the discrimination among activities.

Figure 3.5 shows the performance loss in percentage over the best scores for each dataset. In those cases in which GRU is the best performing architecture, the average difference of performance over the CNN model is $0.67\%$. On the contrary, in those cases in which the CNN model obtains the best score, the best GRU model is $0.42\%$ worse on average. The worst performing architecture (DBN) is $6.04\%$ worse, on average, over the best architecture.

When comparing the speed and the memory footprint of the architectures, DBN is the fastest architecture, while CNNs are the most efficient in terms of memory use. Figure 3.6 shows the time that the best model of each architecture takes to process an input and generate a prediction, as a percentage over the time taken by the fastest model. In all cases the fastest architecture is DBN, which was an expected result since their architecture, regardless of the special training process, is the simplest.

Regarding memory footprint, Figure 3.6 shows the number of trainable parameters, as a percentage over the number of parameters of the least memory-consuming model, among the best for each architecture and dataset. In nine of the ten datasets, the best CNN model has a smaller memory



(a) f1

| Architecture | activemiles | hhar | fusion | mhealth | swell | usc-had | uci-har | pamap2 | opportunity | realworld |
|---|---|---|---|---|---|---|---|---|---|---|
| CNN | **96.79** | **96.31** | **99.19** | 99.18 -0.17% | **98.39** | 92.75 -0.12% | **98.59** | 92.44 -1.94% | **92.57** | 91.48 -0.44% |
| GRU | 96.33 -0.47% | 96.28 -0.03% | 99.08 -0.11% | **99.34** | 98.15 -0.24% | **92.87** | 97.98 -0.62% | **94.27** | 91.63 -1.02% | **91.89** |
| biLSTM | 96.37 -0.43% | 95.82 -0.50% | 98.75 -0.44% | 99.15 -0.20% | 97.62 -0.78% | 92.39 -0.52% | 97.56 -1.04% | 93.30 -1.03% | 90.34 -2.42% | 91.18 -0.77% |
| LSTM | 95.99 -0.82% | 95.45 -0.89% | 98.65 -0.54% | 98.62 -0.72% | 96.98 -1.44% | 91.77 -1.18% | 96.04 -2.59% | 92.37 -2.02% | 88.28 -4.63% | 90.89 -1.09% |
| DBN | 92.99 -3.93% | 92.64 -3.81% | 96.93 -2.28% | 93.92 -5.46% | 89.19 -9.35% | 84.13 -9.41% | 93.58 -5.09% | 85.49 -9.32% | 88.24 -4.68% | 88.29 -3.92% |

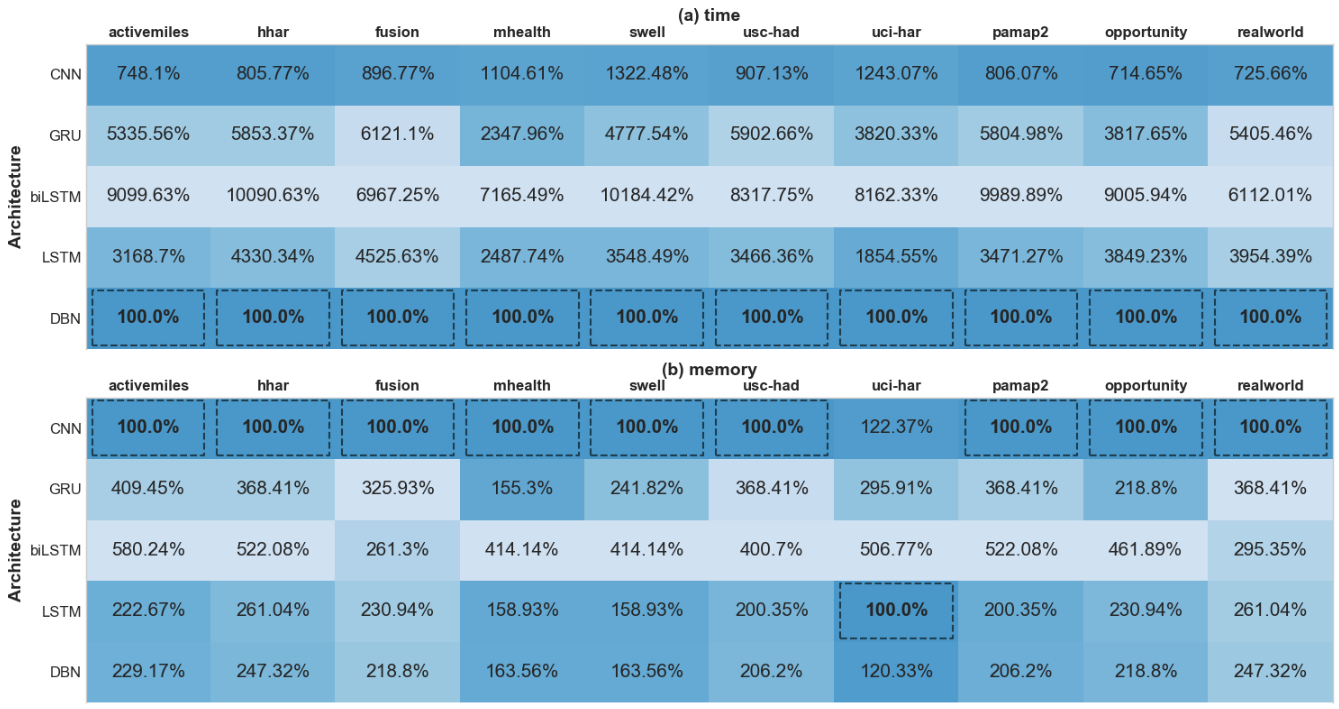
(b) accuracy

| Architecture | activemiles | hhar | fusion | mhealth | swell | usc-had | uci-har | pamap2 | opportunity | realworld |
|---|---|---|---|---|---|---|---|---|---|---|
| CNN | **96.84** | 96.34 -0.05% | **99.29** | 99.17 -0.23% | **98.48** | 92.97 -0.06% | **98.74** | 92.63 -1.87% | **92.44** | 91.51 -0.48% |
| GRU | 96.33 -0.53% | **96.39** | 99.29 -0.00% | **99.39** | 98.27 -0.21% | **93.03** | 98.21 -0.54% | **94.40** | 92.02 -0.45% | **91.96** |
| biLSTM | 96.22 -0.64% | 95.93 -0.47% | 98.93 -0.37% | 99.19 -0.21% | 97.89 -0.60% | 92.58 -0.49% | 97.66 -1.10% | 93.41 -1.05% | 90.59 -2.00% | 91.33 -0.68% |
| LSTM | 96.02 -0.85% | 95.52 -0.90% | 98.77 -0.53% | 99.03 -0.37% | 97.18 -1.32% | 92.11 -0.99% | 96.41 -2.36% | 92.56 -1.94% | 88.79 -3.94% | 90.91 -1.14% |
| DBN | 91.43 -5.59% | 92.46 -4.07% | 97.13 -2.18% | 93.32 -6.11% | 89.36 -9.27% | 82.13 -11.72% | 93.49 -5.31% | 85.08 -9.87% | 87.13 -5.75% | 88.55 -3.71% |

Figure 3.5: Per dataset heatmap of the maximum f1 *(a)* and accuracy *(b)* scores for each architecture. The best score for each dataset is boxed in dashed lines. The table also shows the percentage of performance loss over the best architecture for each dataset.

footprint. The best performing LSTM model for each dataset needs an average of 202% more parameters. The best GRU models have an average of 312% more memory needs than the best CNN models.

As for the structure of the models of each architecture, Figure 3.7, shows the influence of the number of hidden units on the model's performance and training time of the different architectures, as a percentage over the performance of the simplest model. As might be seen in the figures, in the case of recurrent models, the positive impact of increasing the number of hidden units decreases as their number grows. Each step also increases the network's complexity, reaching a point where adding more units will not increase the model's performance. Obviously, each increase in complexity causes a slowdown of the model along with a boost in the cost of training. Bidirectional LSTM models give its peak of average performance at 450 hidden units, while GRU models average performance peaks at 500 hidden



(a) time

| Architecture | activemiles | hhar | fusion | mhealth | swell | usc-had | uci-har | pamap2 | opportunity | realworld |
|---|---|---|---|---|---|---|---|---|---|---|
| CNN | 748.1% | 805.77% | 896.77% | 1104.61% | 1322.48% | 907.13% | 1243.07% | 806.07% | 714.65% | 725.66% |
| GRU | 5335.56% | 5853.37% | 6121.1% | 2347.96% | 4777.54% | 5902.66% | 3820.33% | 5804.98% | 3817.65% | 5405.46% |
| biLSTM | 9099.63% | 10090.63% | 6967.25% | 7165.49% | 10184.42% | 8317.75% | 8162.33% | 9989.89% | 9005.94% | 6112.01% |
| LSTM | 3168.7% | 4330.34% | 4525.63% | 2487.74% | 3548.49% | 3466.36% | 1854.55% | 3471.27% | 3849.23% | 3954.39% |
| DBN | **100.0%** | **100.0%** | **100.0%** | **100.0%** | **100.0%** | **100.0%** | **100.0%** | **100.0%** | **100.0%** | **100.0%** |

(b) memory

| Architecture | activemiles | hhar | fusion | mhealth | swell | usc-had | uci-har | pamap2 | opportunity | realworld |
|---|---|---|---|---|---|---|---|---|---|---|
| CNN | **100.0%** | **100.0%** | **100.0%** | **100.0%** | **100.0%** | **100.0%** | 122.37% | **100.0%** | **100.0%** | **100.0%** |
| GRU | 409.45% | 368.41% | 325.93% | 155.3% | 241.82% | 368.41% | 295.91% | 368.41% | 218.8% | 368.41% |
| biLSTM | 580.24% | 522.08% | 261.3% | 414.14% | 414.14% | 400.7% | 506.77% | 522.08% | 461.89% | 295.35% |
| LSTM | 222.67% | 261.04% | 230.94% | 158.93% | 158.93% | 200.35% | **100.0%** | 200.35% | 230.94% | 261.04% |
| DBN | 229.17% | 247.32% | 218.8% | 163.56% | 163.56% | 206.2% | 120.33% | 206.2% | 218.8% | 247.32% |

Figure 3.6: Per dataset heatmap of the time *(a)* taken to process an input and memory footprint *(b)* for the best model for each architecture, in percentage over the fastest. The best score for each dataset is boxed in dashed lines.

units and LSTM models at 550 hidden units.

Regarding the results obtained for the CNN models, Figure 3.7 shows the impact of the temporal size of the first convolutional layer filter. The best results are obtained with filters that are 11, 9, and 7 time steps wide, which for a sampling rate of 50Hz will correspond to roughly 0.2 seconds. As for the speed, the range between the worst and the best model (around 70%) is narrower than the range in recurrent models (around 500%), showing that changes in the filter size have a small impact on the speed of the model.

The results obtained for DBNs show that the best results are obtained with 3 hidden layers. As for the speed, as it can be expected, decreases as the complexity of the model grows.

The k-fold stratified cross-validation strategy followed in these experiments implies that the training, validation, and test sets have been drawn from the same distribution. This is, variables that can influence the parameters

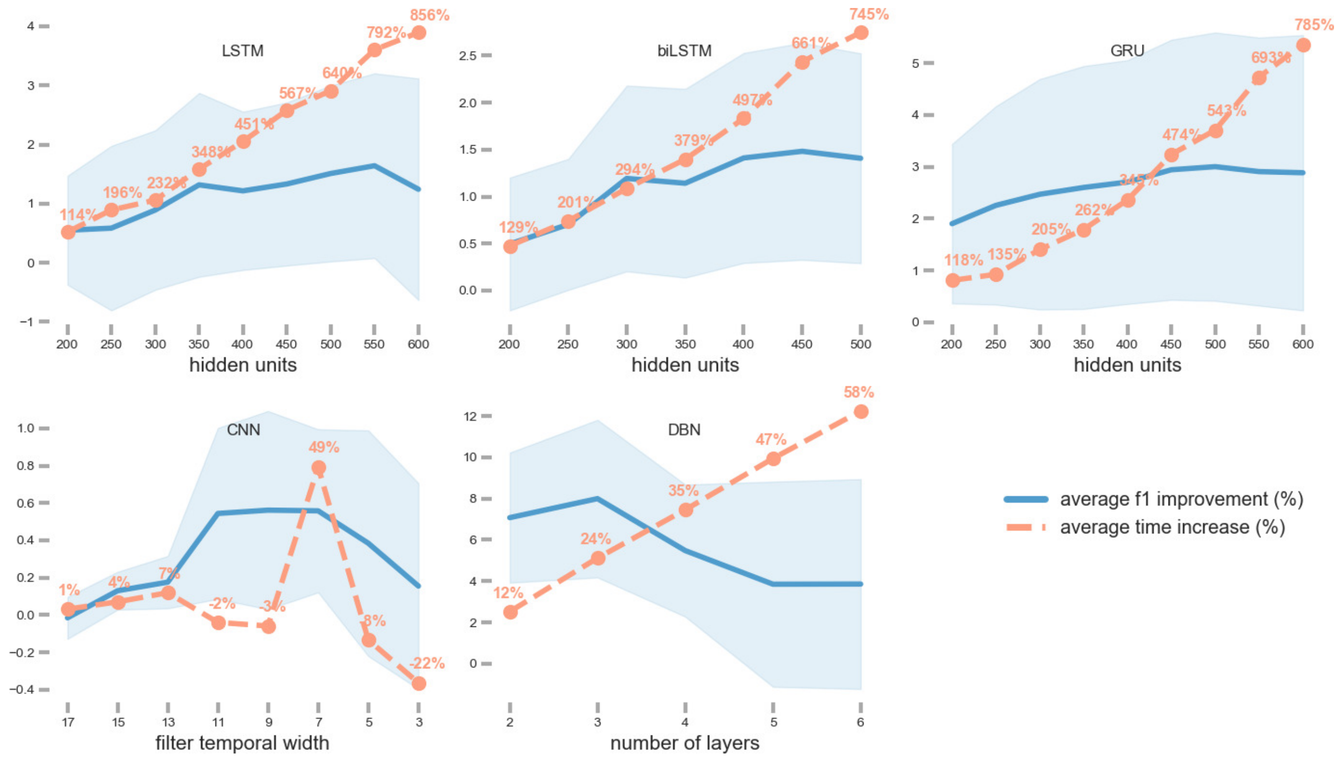


Figure 3.7: F1 score and time of prediction for each model, as a percentage over the simplest model. The grayed area represents the confidence level interval.

of the distribution for each set have been equally distributed, and patterns found in the training set will probably be found in the test data as well, thus facilitating the capacity of the trained model to achieve good results. One simple example of this kind of variable can be the user who acquired the data. Distinct users may show different peculiarities when performing activities, and using data from all users when training allows the model to learn these differences.

To assess the ability of the best-performing deep architectures to learn significative patterns in these types of scenarios, we selected three datasets to perform some additional tests. For each dataset, we trained the best performing model from the best architectures with a leave-one-out strategy, this is, we reserved the data from one user to evaluate the performance of the model and used the rest of the data to train the model. This process was repeated for each user. The reported result for each dataset and model is the average f1 score for all users. The number of users varies from 10 in the *fusion* and *mhealth*, to 30 users for the *uci-har* dataset. The results are presented in Figure 3.8. Results show a big difference in the general performance of deep models compared with previous results, where the data on which they were tested was very similar to the data on which they were trained. While these might be state-of-the-art models, with really high performance, it would be only on the very specific distribution represented by the training data. When used in a new task which might still be similar to the one they were trained on, as is the case of new data from a new user, they end up suffering a significant loss in performance. In these circumstances, neural networks fail to form a general understanding of the meaningful patterns that characterize each class. Combating this issue [111] is a difficult problem that generally requires more data to train and/or specific training procedures to prevent overfitting. Ironically, the use of engineered features can help, to some extent, to prevent these issues. Using the same leave-one-out strategy on the set of manually extracted features provided in the *uci-har* dataset with a classic machine learning algorithm such as SVM provides an average accuracy of 0.75. Although this simple test might not be significative, it seems reasonable to consider that a good set of well-selected features might be representative of the distinct characteristics of human activities, and more resilient to changes in the environment such as the user that is performing such activities.

Finally, in order to compare the performance of the trained deep learning models with various classical machine learning algorithms we performed some tests using the set of features provided in the *uci-har* dataset. Using three different machine learning classifiers (support vector machines, 3-nearest neighbors, and random forest) the results obtained were noticeably inferior to those achieved by convolutional and recurrent networks. Table 3.6 shows the results obtained. The support vector machines classifier obtained the best results.

### 3.6.2 Discussion

One of the key points of the CNN architecture is its shift-invariance property, that enables a pattern in the input to be recognized in any position. This type of DL architecture is able to successfully capture the temporal dependencies

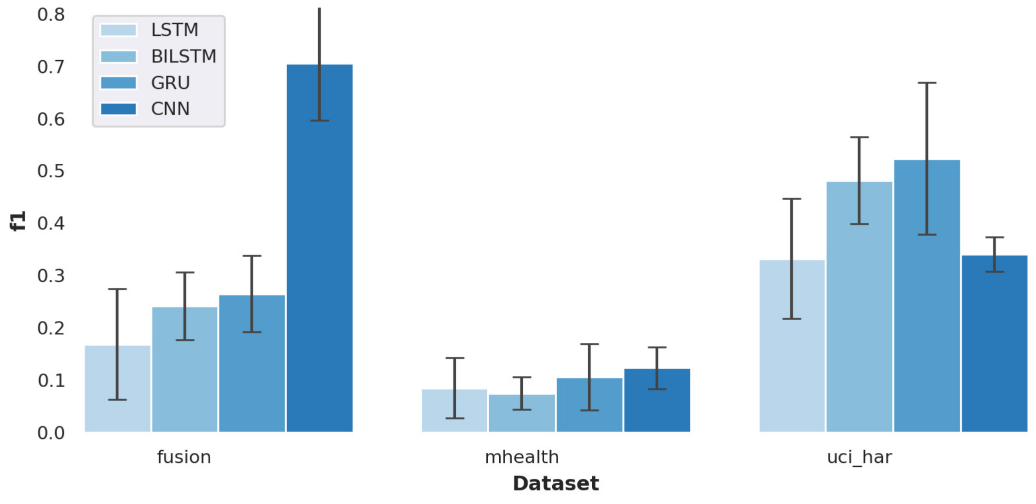


Figure 3.8: F1 score for each dataset and architecture when training with a leave-one-out strategy. Error bars show the standard deviation of the results.

Table 3.6: Accuracy and f1 for three classical machine learning algorithms on the uci-har dataset using the set of features provided by the authors.

| algorithm | accuracy | f1 |
|---|---|---|
| support vector machines | 0.9504 | 0.9504 |
| 3 nearest neighbors | 0.8907 | 0.8898 |
| random forest | 0.9253 | 0.9251 |

in raw sensor data structured as time series through the application of relevant filters. The pooling layers contribute to providing limited scale invariance properties, allowing for slight deformations in the input signals. Given that different people may execute an action with distinct paces (e.g., an elder person may walk at a slower pace than a young person), this scale deformation tolerance enables CNNs to effectively identify the pattern that characterizes the activity regardless of its pace of execution. The temporal size of the filter for the best models, of around 10 time steps, indicates that there is no need to capture long-distance dependencies in the input stream to obtain good results characterizing human activities. Architectures designed to do so, such as RNN, do not perform consistently better than CNN and have some disadvantages, such as their computational complexity and higher memory needs.

The unfolded computational graph of the RNN architecture results in the sharing of parameters across the deep network structure. This facilitates the use of the same parameters for different time steps, enabling the model to efficiently learn discriminative patterns regardless of their position in the input sequence. The results obtained with GRU models, which are comparable to the results achieved by CNN models, validate the use of GRU for human activity recognition tasks. This type of recurrent neural networks achieves better results than LSTM and bidirectional LSTM, at least using simple vanilla architectures, and come with the advantage of their relatively low cost in terms of memory and computation resources, at least compared with other recurrent architectures.

Finally, we see that the fully connected architecture of DBN limits their suitability for time series classification tasks, at least without previous manual feature extraction. The pretraining process using unlabeled data enables initializing layer weights in a way that should make feasible extracting significant features from the raw data. But the patterns that define the different activities can be present at any time stamp, and the fact that the weights are not shared implies that those pattern detectors have to be learned for each possible position in the input. However, since the availability of unlabeled data is significantly higher, exploring ways of leveraging this information to get better models is a research line on which these kinds of generative models can have their application. Either DBN or other architectures such as autoencoders or variational autoencoders could be employed as feature extractors trained in an unsupervised process, and then connected to a CNN or RNN model for fine-tuning with labeled data.

In summary, the results are compatible with the hypothesis that CNNs are efficient at capturing local temporal dependencies of activity signals, and are also capable of identifying multi-modal correlations among sensors. Their performance in activity classification is comparable to, and in some cases better than, RNNs. Their faster response and lower memory footprint make them the architecture of choice for wearable and IoT devices, where restrictions both in terms of power consumption, computational capabilities, and memory availability might play a decisive role.

### 3.6.3 Results reproducibility

To allow for the verification and validation of results, the software framework implemented to run the experiments has been made publicly available. The public repository [1] includes the log files of all the individual experiments and results for each model and architecture, as well as the scripts used to generate the plots and figures included in this paper. The software is easily extensible and has been implemented with the aim to allow for the reproducibility of the experiments included in this paper and to ease the creation of new ones to effortlessly extend this research. The repository provides details about the requirements, data sources, and how to run experiments and create new ones. The datasets are not included in the repository, although instructions on how to obtain them are provided.

The experiments were conducted on two identical machines with Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz, NVIDIA GeForce GTX 1080 Ti GPU and running Ubuntu v17.10 as the operating system, using Pytorch version 0.3 and CUDA version 9.0. It took more than 4000 hours to complete the total set of experiments.

## 3.7 Conclussions and future work

In this paper, we used a testbed of ten publicly available benchmark datasets to compare the performance, speed, and complexity of five different architectures of DNNs on HAR classification tasks. The results show that CNNs achieve very competitive performance scores with respect to GRU networks, and better results than biLSTM, LSTM, and DBN models. Regarding memory requirements, CNN is clearly the least memory demanding architecture.

The continuous progression of artificial intelligence into the mainstream due to the advent of enabling technologies such as machine learning, smart devices, cloud storage or high-speed networks, will require systems to be able to provide real-time insights in key fields such as ambient assisted living or health monitoring, often through the use of low-cost embedded devices

[1] `https://github.com/esansano/dl-for-har-comparison`

with strict restrictions on memory availability, power consumption, connectivity, and computational capabilities. Overall, both in terms of speed and memory requirements, the study allows confirming that CNNs are specially appropriated for activity recognition tasks in such scarce resource scenarios.

*Future work*. Given the extensive quantity of HAR data available, we plan to assess the possibility of using pre-trained models to speed up training and improve the performance on datasets where few data is available, decreasing thus the need for large volumes of new data and the time needed to adjust a new model. Transfer learning [113] makes use of the knowledge acquired by a model during the training process on a big dataset to solve a related classification problem where few labeled data is available. For example, a CNN can be trained with a big dataset such as *pamap2* or *realworld*, or a combination of both, and, once the training process has ended, be tuned to classify classes from a dataset with a limited amount of data and a different set of activities, such as *uci-har* or *swell*.

## Acknowledgments

This work has been partially funded by the Spanish Ministry of Science, Innovation and Universities through the “Proyectos I + D Retos investigación” programme (RTI2018-095168-B-C53) and by Jaume I University “Research promotion plan 2017” programme (UJI-B2017-45).

### Financial disclosure

None reported.

### Conflict of interest

The authors declare no potential conflicts of interest.

# Chapter 4

# Multimodal sensor data integration for indoor positioning in Ambient Assisted Living environments

A reliable Indoor Positioning System (IPS) is a crucial part of the Ambient Assisted Living (AAL) concept. The use of Wi-Fi fingerprinting techniques to determine the location of the user, based on the Received Signal Strength Indication (RSSI) mapping, avoids the need to deploy a dedicated positioning infrastructure but comes with its own issues. Heterogeneity of devices and RSSI variability in space and time due to environment changing conditions pose a challenge to positioning systems based on this technique.

The primary purpose of this research is to examine the viability of leveraging other sensors in aiding the positioning system to provide more accurate predictions. In particular, the experiments presented in this work show that Inertial Motion Units (IMU), which are present by default in smart devices such as smart-phones or smart-watches, can increase the performance of indoor positioning systems in AAL environments. Furthermore, this paper assesses a set of techniques to predict the future performance of the positioning system based on the training data, as well as complementary strategies such as data scaling and the use of consecutive Wi-Fi scanning to further improve the reliability of the IPS predictions. This research shows that a robust positioning estimation can be derived from such strategies.

## 4.1 Introduction

The current increase in the population's average age [75] leads to new requirements in the healthcare domain, particularly in aspects such as care-giving, home assistance, rehabilitation, early detection of diseases, or physical support [163]. The need for assistance and healthcare to the elderly is becoming more and more necessary for social as well as for economic reasons. This trend urges affording suitable assistance systems to improve the quality of life of older people [108] with the aim to help them live an active and productive aging at an affordable cost [124].

Due to underlying and often debilitating health conditions that are associated with elderly people, aspects of everyday living can become physically and mentally challenging for them. Technology can be integrated into the health care of senior citizens to provide safe, high-quality lives, improve their health and happiness, and enable a longer period of independent living. Assistive technical applications should be easy to use, unobtrusive, suitably designed, and adaptable to changing needs and individual preferences.

Ambient Assisted Living (AAL) concept has been defined as a set of products and services aimed to build intelligent environments in the assistance of these groups of people [122]. One of the goals of AAL is to provide reliable and meaningful information to health professionals, caregivers, psychologists or family members. AAL applications consist of networks of heterogeneous information appliances and smart artifacts that can assist people with special needs in several areas such as daily task facilitation, mobility assistance, health-care and rehabilitation, and social inclusion and communication. With the help of Artificial Intelligence (AI), AAL facilitates the use of technology in a non-intrusive way to support safe, high-quality, and independent lives for the frail and elderly.

AAL platforms strongly rely on an accurate underlying localization system in order to provide timely and reliable services to elderly users. Knowing their position and actions is vital for medical observation, timely accident prevention, behavioral pattern characterization, or anomaly detection [34].

While the problem of localization in outdoor environments has been solved by the use of satellite positioning systems such as GPS or Galileo, which provide an acceptable level of accuracy and precision, indoor positioning remains an open issue. Satellital signals are not available inside buildings, since they are attenuated and scattered by roofs, walls and other building elements, and are unable to reach the user's device with enough intensity to provide precise positioning services. Researchers and industry are currently involved in the investigation, development, and improvement of reliable indoor positioning systems. Although significant progress has been made, there is not an accurate and widely accepted solution for this topic.

Indoor Positioning Systems (IPS) are systems that locate and track people or objects inside buildings using radio waves, magnetic field, acoustic signals, images, or other information collected by sensors [29]. A suitable IPS system for AAL has to be able to localize the assisted person in an indoor environment, with accuracy and performance high enough to reliably monitor his/her activity and provide meaningful assistance and services. These systems have to be deployed at the user's living place, in real scenarios where the particular constructive characteristics and peculiarities of the building may affect the way that signals propagate. This type of scenarios are very different from controlled experiments where environmental conditions are known in advance. The fact that homes are diverse, with different layouts and varied architectural particularities makes it complex, time-consuming, and expensive to model the propagation of radio frequency signals each time the positioning system has to be installed. Furthermore, technical proposals for AAL should be easy to use, unobtrusive, and inexpensive, so deployment has to be as simple as possible.

In order to predict the position of an agent in an indoor environment, traditional approaches rely on the construction of reliable models for signal propagation, which is a complex topic. The performance of these methods depends on the correct assumptions about the underlying rules governing the observed signals. If the assumptions are wrong, the model will not describe the observations and will not be able to make solid predictions. On the other hand, Machine Learning (ML) algorithms allow the identification of correlations in data sets without the need for a proper determination of the underlying model. In other words, ML techniques treat the model as a black box for which an explicit characterization is unknown. The training process of the supervised ML algorithms is able to uncover meaningful and characteristic patterns directly from the data and to build effective and predictive models that perform well on unseen data.

Wi-Fi fingerprinting, which uses ML techniques to take advantage of an already deployed infrastructure, is a good choice for such systems [143, 22]. Nevertheless, factors such as channel interference, sensor orientation, or multipath propagation and fading introduce a level of indeterminacy in Wi-Fi sensor readings, impacting negatively in the performance of this positioning method [154]. In scenarios where accuracy at room level is enough to provide relevant services, selecting an adequate classifier algorithm, and collecting data appropriately can significantly improve precision. Furthermore, taking into consideration readings from other sensors, such as Inertial Motion Units (IMU), along with Wi-Fi fingerprints, can help to account for changes in the user's location, providing valuable information that can be utilized to improve the IPS results.

This paper presents the results of a study to assess the impact on the IPS performance of strategies based on the utilization of motion sensor readings to detect user states. This work also presents some preliminary data study techniques that can help to predict the quality of the data recorded by the users, which directly affects the accuracy of the IPS system. Furthermore, we also carried out a set of experiments to evaluate the effectiveness of a series of actions aimed to reduce the influence of Wi-Fi signal uncertainty and to select the most appropriate ML algorithm for the positioning system. As a summary, the main contributions of this paper are as follows:

1. We perform a set of data analysis techniques that help to predict the performance of the positioning system based on the characteristics of the training data recorded by the user.
2. We assess the performance gain of considering readings from body-worn inertial sensors to recognize room transitions.
3. We compare the impact of some strategies to increase the positioning

performance of the positioning algorithms and to reduce uncertainty in Wi-Fi signals.

4. We also compare the performance of four machine learning algorithms in room-level indoor localization tasks.

A preliminary version of this work, entitled *"Improving positioning accuracy in Ambient Assisted Living environments. A multi-sensor approach."* [138] was presented at the *15th International Conference on Intelligent Environments (IE19)*. With respect to the preliminary version, we have extended and partially rewritten all sections. Furthermore, we have added a new section dedicated to exploring data characteristics and their relationship with the performance of the classification algorithms used in the experiments.

The organization of this paper is as follows: Section 4.2 shortly provides context on Ambient Assisted Living and Indoor Positioning Systems. Section 4.3 presents an overview of the positioning system used to perform the experiments. Section 4.4 presents a study of the data and discusses the outcomes. Section 4.6 describes the experiments performed and Section 4.7 discusses their results. Finally, Section 4.8 underlines the main conclusions of this work and explores possible future lines of research.

## 4.2 Background

General AI and Machine Learning (ML) based systems are being developed and used in areas such as context-awareness, agent-based technologies or computer vision, to provide more intelligent, flexible, natural and supportive services for health-care. Some examples of how services based on AI and ML techniques can be used in healthcare services are:

- Human Activity Recognition (HAR): Systems can combine data from multiple sensors to recognize user's activities and identify behavioral patterns. The performance of daily activities can be used as a measure of the cognitive and physical condition of the elderly [14, 8, 32].
- Anomaly Detection: Anomaly detection techniques can expose declining health conditions. Changes and anomalies in the user behavior can be of use in chronic diseases monitoring [35] and early depression

detection [40], and can denote elder-specific illnesses such as cognitive decline, Alzheimer, dementia or functional impairment [64, 48].

– Decision Support: Decision support systems assemble different types of data from multiple patients and help doctors and healthcare professionals to organize their work, to analyze people personal needs or to survey some common phenomenon.

Some of the key aspects of deploying an IPS for Ambient Assisted Living are related to choosing the right positioning technology while implementing the system in a passive, device-free, and unobtrusive way. This objective might require the use of an existing infrastructure, the deployment of a new one, the use of the so-called signals-of-opportunity, or even a combination of some of these techniques. Many of these techniques take advantage of the radio-frequency signals emitted by devices, whose position can be known or not, to estimate the user's position from the perceived strength of these signals, the Received Signal Strength Indication (RSSI). RSSI is used to measure the relative quality of a received signal to a client device. The value read by a device is given on a logarithmic scale and can correspond to an instant reading or a mean of some consecutive readings, but each chipset manufacturer is free to define their own scale for this term. There are many kinds of devices and technologies that can be used for positioning purposes, such as Wi-Fi access points, Bluetooth beacons, Radio Frequency Identification (RFID), or Ultra-Wide Band (UWB) devices [2]. The effectiveness of these techniques can be improved by leveraging the use of other sensors that are commonly present in wearable devices.

Beyond WiFi or Bluetooth signals, the use of the Earth's magnetic field to build an indoor positioning system has been explored in recent years in several research works. Man-made constructions cause disturbances that alter the magnetic field. These magnetic anomalies are location specific and temporally stable, and can be leveraged to build an indoor positioning framework. In many works, this approach is combined with other sensors to enhance its performance. For example, in [149] authors use the magnetic field along with opportunistic WiFi signals to achieve a 90 percentile accuracy of 3.5 m for localization. The use of different deep learning architectures, such as deep neural networks (DNN) or convolutional neural networks (CNN) has also been proved to achieve good localization accuracy [12, 9, 10].

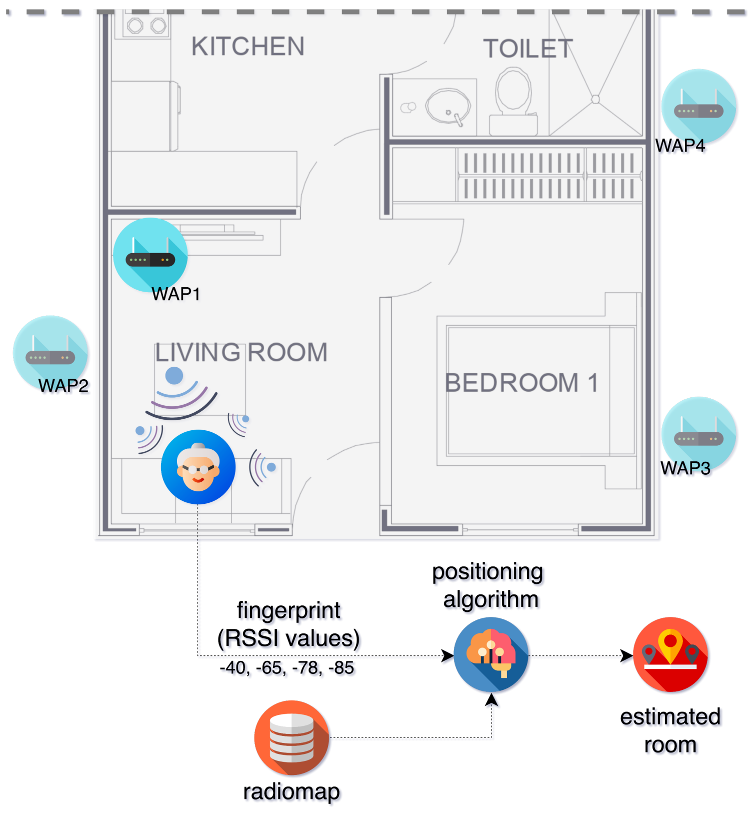


Figure 4.1: During the on-line phase, once the radio map has been built, the fingerprinting algorithm uses it to estimate the device's position by comparing the RSSI values heard by the device with the ones stored in the radio map.
(Icons made by Freepik from www.flaticon.com)

Using inertial sensors' data has been applied previously to solve diverse problems related to localization. In [5] authors present a pedestrian dead reckoning tracking system that relies on two modules; a step counter and a stride length estimator. Although the reported results are good, their solution is based on a homogeneous walking, which can not be assumed in some small indoor scenarios such as homes. Combining radio-frequency signals along with other sensors' data has been implemented in several previous works. For example, Xie et. al. [171] achieve good accuracy in large indoor buildings using magnetic field fingerprinting together with an augmented particle filter. The use of particle filters to fuse data from various sensors has been a common practice for indoor localization systems [180, 71], generally providing good results. In [67] authors use magnetic field readings along with WiFi to create a Spatio-temporal signal fusion graph to identify crowd-flows in large indoor scenarios such as malls or airports. This technique can be applied to applications or services like advertisement and recommendation or urban-flow monitoring systems.

Techniques used for indoor location can be divided into three general categories: proximity, triangulation, and fingerprinting [44].

– **Proximity** methods compare the RSSI value from different transmitters and determine the position of the client assuming that the received signal with the highest value is from the closest access point. The accuracy is generally low and relates to the density of deployed beacons and its signal range.

– **Triangulation** methods use the geometric properties of triangles to determine the target location. When the position of at least three transmitters is known, the position of the mobile node can be estimated calculating its distance to each device. The difficulties come with the task of finding the right model for transforming RSSI to distance. Triangulation methods can be divided into two groups; lateration techniques such as Time-of-Arrival (ToA), Time-Difference-of-Arrival (TDoA) or Round-Trip-Travel-Time (RTTT), based on the measurement of the propagation time, and angulation techniques such as Angle-of-Arrival (AoA), based on the angle of the arrival wave. These technologies are not available to inexpensive positioning infrastructures due to the need for antenna arrays or time synchronization [31].

– **Fingerprinting** methods assume that, for a given indoor environment, a signal mapping exists and that such map can be reconstructed measuring the RSSI signal at discrete locations of the mapped area. In the case of Wi-Fi fingerprinting, its main advantage relies on the fact that there already is an existing Wi-Fi infrastructure in the majority of urban areas. Therefore, the location of the user can be obtained without deploying any additional equipment. Obstacles, reflections, multi-path interference, environmental changes, or device orientation are factors that affect the signal propagation [74] and can degrade the performance of IPS based on Wi-Fi fingerprinting.

Mapping fingerprinting assumes that an RSSI map exists, and it is constructed by measuring the RSSI at some locations of interest. The radio map, or fingerprinting data set, is composed of a set of collected fingerprints and the associated positions where the measurements were taken and may contain some additional variables, such as the type of device used or a timestamp of the observation, along with any other useful data. This stage in which the data is acquired to construct the radio map is known as training, off-line, or survey phase. During operation, once the radio map is completed, the IPS will use this map as a database for location purposes [66]. This stage is known as the online phase (see Fig. 4.1).

Determining the location of a receiver device at room level based on the RSSI mapping can be seen as a classification problem, where the classes are the mapped rooms and the features are the RSSI signals. However, there are some issues that make it difficult to achieve good classification performance in IPS. The main problems are caused by the heterogeneity of devices and RSSI variability in space and time due to environment changing conditions. Regarding the former, since RSSI is a not standardized indication of power level being received by a wireless device, any device manufacturer may implement its measuring in a different way. Therefore, RSSI value can vary depending on the hardware, software driver libraries, operating system or software monitoring implementation. With respect to the latter, RSSI is sensitive to dynamic environmental conditions such as channel noise, interference, reflection, and attenuation. This can degrade the performance of the IPS when circumstances change from the offline to the online phase. There have been some proposals to tackle different aspects of the heterogeneity problems. For example, in [179] authors find that the relation in the order of RSS values from different APs at a fixed location is more stable than the values themselves, and propose the use of an algorithm that uses this relation to construct a more stable fingerprint. Other works [28] propose hybrid systems based on the use of pyroelectric infrared sensors to process sets of zone-based fingerprints with the goal of excluding outliers due to device diversity or shadowing effects. Other authors [11] disregard the traditional approach for fingerprinting and propose a system that exploits the WiFi access points coverage area uniqueness and the coverage area overlap to calculate the user's current position while mitigating the impact of using heterogeneous devices.

Furthermore, collecting and maintaining a radio fingerprint database is a high cost and time-consuming task. This cost can be reduced considerably in household environments when room level positioning is enough to provide most AAL services. In those cases, the training stage has to be performed at least once in each room. In a typical house with 6 rooms, this process can take 10-20 minutes.

The ubiquity of smart-phones and smart-watches, and the availability of different wireless interfaces, such as Wi-Fi, 3G, and Bluetooth, make them an attractive platform for indoor monitoring. Smart home-based behavioral data have already been found to be useful in assisting older adults to live independently and to monitor health state and the onset and progress of age-related diseases and disorders such as dementia and Alzheimer's disease [7]. Psychological health in older adults (loneliness, depression, or emotional states) has been assessed by means of such data too [13]. Nevertheless, the level of technology readiness for home health monitoring technologies is still low [88].

When choosing a particular device to implement an IPS in AAL, one of the most important factors to consider is the fact that it has to be as unobtrusive as possible and do not modify, disturb, limit or interfere in the user's daily activities or lifestyle. Most Wi-Fi fingerprinting location systems are based on the use of a smart-phone. Nevertheless, to track the user location implies the device to be permanently attached to the user, which may not be applicable in home daily living. For instance, forgetting the device on the top of the bedside table would point the IPS to assume that the user is still in bed.

Smart-watches can be seen as an extension of a smart-phone which looks like a common watch. A smart-watch is always attached to the user, so it is less likely to be forgotten on top of the bedside table than a smart-phone. In addition, it is a non-obtrusive, relatively cheap, and easy to use tool, which can also provide direct communication between the user and caregivers, nurses, or general practitioners.

As most smart-phones do, smart-watches also embody several sensors such as accelerometer, gyroscope, ambient light intensity, compass, and so on. On the connectivity layer, most of them also embody Bluetooth, NFC, and Wi-

Fi communications, which allows the use of Wi-Fi fingerprinting technology as a suitable positioning candidate to be deployed in such devices. Moreover, most of these devices also include a GPS chip. This sensor can be used along with an IPS to provide the location of the user both outdoors and indoors.

## 4.3 System overview

The Indoor Positioning System designed to perform these experiments is part of the research project '*Senior Monitoring*' [100], which is aimed to provide solutions for monitoring elderly people behavior and to detect short-term issues (falls) as well as long-term issues (cognitive decay). The IPS consists of a smart-watch, which is worn by the user who is being monitored, and a paired smart-phone, which is used to configure and control the smart-watch behavior and to communicate with a central cloud server (see Fig. 4.2). The server stores the sensory data gathered through the smart-watch and offers assistance to provide decision support services by performing analysis tasks such as indoor positioning, activity recognition, or anomaly detection.

### 4.3.1 Hardware

The IPS described in this paper requires the use of a smart-watch attached to the user's wrist and a smart-phone that communicates with the former through a user-friendly application in the following way:

- **Smart-watch**. The wearable device used is the model SmartWatch 3, manufactured by Sony. This device runs Android Wear as its operating system and embodies a Wi-Fi chip along with GPS, accelerometer, compass, gyroscope, and ambient light sensors. Connectivity is supported through Wi-Fi, NFC, and Bluetooth. The resolution of its 1.8" screen is 320x320 pixels. This device runs an application that can be set up to continuously scan for any nearby Wireless Access Points (WAP) signal, as well as to record readings from some other sensors. This application is controlled via a paired Android smart-phone that runs its corresponding version of the application.

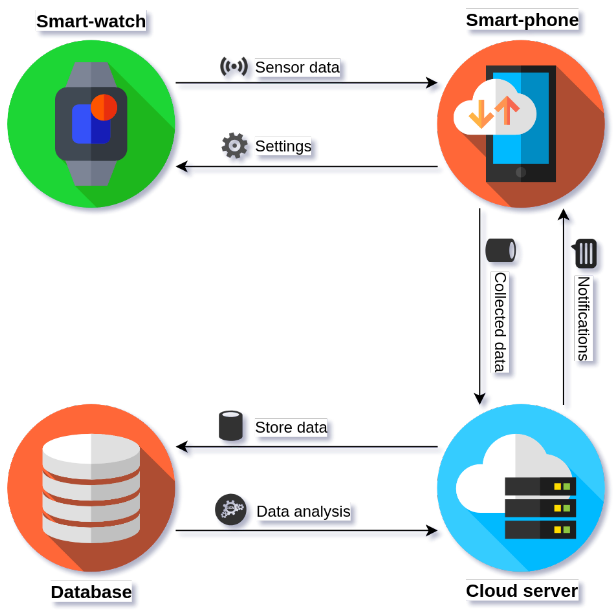


Figure 4.2: Senior monitoring IPS overview. The smart-watch sends sensors data to the smart-phone. An application installed in the smart-phone is the interface through witch the user or the caregiver configures the smart-watch. The smart-phone sends the data to a cloud server. This server performs data analysis and can communicate with the user through notifications.
(Icons made by Freepik from www.flaticon.com)

– **Smart-phone**. The smart-watch is paired to a smart-phone that controls its behavior. The smart-phone is used to configure some sensor options such as scan intervals, number of consecutive scans, sensor activation, etc. Both devices communicate through Bluetooth. All the smart-watch readings are sent to a central server through the smart-phone. In case the devices are not in range, the smart-watch buffers the data to be sent when a network connection becomes available.

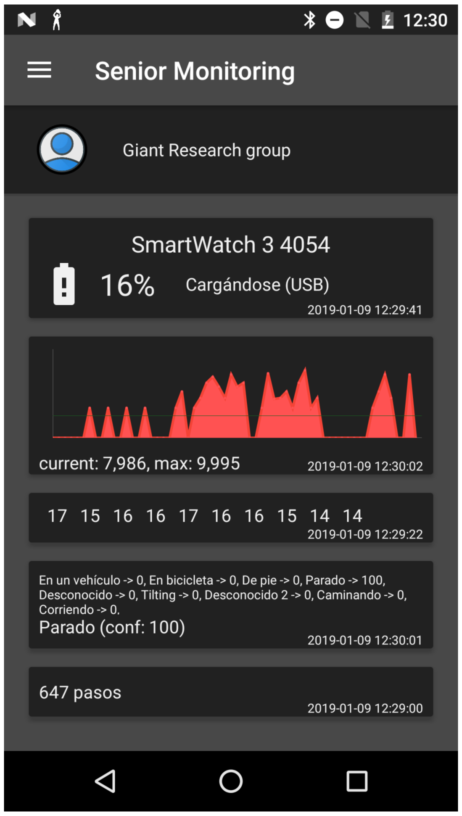
### 4.3.2 Software

The Android application that runs in the smart-watch is in charge of collecting the sensor data. The configuration and behavior of this application are controlled by means of its reciprocal application installed in the paired smart-phone. Figure 4.3 shows the main screen of the smart-phone application. This version of the software is used for research purposes, so it shows some information relevant only to this purpose, along with general information that is useful for end-users. The smart-phone is the interface through which the elderly user can perform tasks such as checking the smart-watch status, viewing his/her level of physical activity, observing the readings and status of active sensors or responding to notifications delivered by his/her caregivers, health professionals, psychologists or automatic healthcare services provided by the analysis system.

The smart-phone sends the sensory data to a cloud server using the MQTT protocol. The server stores the data for posterior analysis using Elasticsearch as a NoSQL database. The data provided through user interaction, such as login data or interaction through notifications, is sent to a REST cloud server and stored in the same database.

### 4.3.3 Sensors

The goal of the structure described so far is to collect meaningful sensory data to build systems able to provide reliable AAL services to its users. To this end, the software previously outlined has been designed to make use of the following sensors:



SmartWatch 3 4054
16% Cargándose (USB)
2019-01-09 12:29:41

current: 7,986, max: 9,995
2019-01-09 12:30:02

17 15 16 16 17 16 16 15 14 14
2019-01-09 12:29:22

En un vehículo -> 0, En bicicleta -> 0, De pie -> 0, Parado -> 100, Desconocido -> 0, Tilting -> 0, Desconocido 2 -> 0, Caminando -> 0, Corriendo -> 0.
Parado (conf: 100)
2019-01-09 12:30:01

647 pasos
2019-01-09 12:29:00

Figure 4.3: The application main screen shows detailed information about the smart-watch status, the levels of physical activity of the user in the previous hour, and other useful information from other active sensor such as Wi-Fi or step counter.

The goal of the structure described so far is to collect meaningful sensory data to build systems able to provide reliable AAL services to its users. To this end, the software previously outlined has been designed to make use of the following sensors:

– **Wi-Fi**. This sensor constitutes the base of the positioning system. The smart-watch performs a given number of consecutive Wi-Fi scans every minute. The default number of scans is 5, but this setting can be modified through the smart-phone app. The procedure is described as follows:

  1. The app sends a *startScan* command to the Wi-Fi module to scan for nearby AP signals.
  2. The Wi-Fi module performs a scan and stores its results in the cache. A notification is sent to the operating system.
  3. The operating system notifies the app when a scan is completed. The app then sends a *getScanResults* command to request the scanning results stored in the cache.

  When a scan is performed, the Wi-Fi module updates some data in the cache while keeping some intact. Some WAPs may be present in the scan results although they have not been detected in the most recent scan. The details of the cache updating algorithm are unknown, but outdated data may persist during some scans. Moreover, in highly crowded WAP environments, channel interference is very likely. This means that some WAP signals, especially those whose RSSI is low, may appear and disappear stochastically. Other circumstances such as heating, ventilation, etc. have their own impact on the radio signals. Because of the aforementioned conditions, signals collected from incorrect locations at incorrect times are likely to happen, introducing errors in data analysis. To minimize the impact of this behavior and lessen stochasticity, the application completes a default number of 5 consecutive scans, each one taking approximately one second to complete. For the smartwatch model used in the experiments, these settings allowed for around 15 hours of battery duration, long enough to collect data during day time and recharge the device during the night.

– **Significant Motion Sensor**. The user physical activity can be determined with the use of inertial sensors such as accelerometer and gyroscope. Both sensors are capable of measuring human motion and estimate body position, allowing to determine the physical activity the user is performing, such as walk, run, sit, etc [77]. The main drawback of the use of these sensors in a smart-watch is its energy cost. Continuously monitoring inertial readings keeps the system from going into low power/sleep mode and drastically reduces the battery duration to less than a whole day, which is a minimum requirement for monitoring applications.

  An alternative is the use of the Significant Motion Sensor (SMS), a virtual sensor that uses the physical accelerometer, but only triggers when it detects a motion that might lead to a change in the user's location. Thus, though this sensor does not allow to determine the activity the user is performing, it provides a way to detect a possible change in his/her location. Inversely, if the SMS has not been triggered during an interval of time, it may be assumed that the user has not changed its location during that period.

– **Step counter**. This sensor detects the number of steps taken by the user since the last time the sensor was activated. The application automatically resets the counter every day at midnight. Similarly to the SMS, the step counter could help to detect intervals during which the user is not walking.

– **Activity recognition**. In order to automatically monitor the user's activity, at least one inertial sensor, preferably the accelerometer, has to be continuously monitored and its data analyzed in search of patterns that characterize the activities of interest. This would cause a considerable battery drain, seriously compromising the device's usefulness. To remedy this situation, the Android API allows registering for activity recognition updates. To keep the power usage to a minimum, the activity detection is done by periodically waking up the smart-watch and reading short bursts of motion sensor data. It can detect if the user is currently on foot, in a vehicle, on a bicycle or still, but the accuracy of the prediction depends on the update interval. Larger interval values will result in fewer activity detections, while smaller values will result in more frequent activity detections, but will consume more power. Each detection result contains a list of activities sorted by a probability that indicates how likely that activity is.

To prevent excessive battery use, the activity reporting service may stop when the device is motionless for an extended period of time. Once the device moves again, which is detected through the SMS, the service will resume.

– **Magnetic field**. Geomagnetic fingerprinting (GF) is a technique that maps disturbances of the Earth's magnetic field caused by the metal construction of buildings and uses this data to achieve indoor localization through pattern matching [161].

  The 3D magnetometer of the smart-watch measures the magnetic field in its coordinate system. As the smart-watch may be oriented arbitrarily in the user's wrist, the measurements have to be transformed into the coordinate system of the indoor plan, which can be done with the aid of inertial sensors such as the accelerometer and the gyroscope. An alternative to this transformation is to only use the module of the signals, thus eliminating the need for other sensor reading but compromising the quality of the localization.

  Geomagnetic fingerprinting can be integrated with some other positioning technology in a sensor fusion system to improve localization. For instance, Wi-Fi fingerprinting can be used to determine the location at room level, and GF to estimate the most likely position within the room.

  The smart-watch scans the magnetic field continuously and sends the collected data to the server every minute.

## 4.4 Data

When the system is deployed in a home, users manually create the radio map while wearing the smart-watch and following the indications of the smart-phone application. The users first select a set of rooms and then the software guides them to collect training data in certain points of the selected rooms, such as the center or any commonly used location. When this process finishes, the collected training data is sent to the server. During

Table 4.1: Data distribution for each data set

| Data set | WAPs | Instances | Room 1 | Room 2 | Room 3 | Room 4 | Room 5 | Room 6 |
|---|---|---|---|---|---|---|---|---|
| user1 Train | 45 | 4900 | 800 (16.33%) | 800 (16.33%) | 800 (16.33%) | 900 (18.35%) | 800 (16.33%) | 800 (16.33%) |
| user1 Test | | 29227 | 3909 (13.37%) | 8930 (30.55%) | 1214 (4.16%) | 1010 (3.46%) | 1870 (6.40%) | 12294 (42.06%) |
| user2 Train | 115 | 3600 | 800 (22.22%) | 800 (22.22%) | 500 (13.89%) | 600 (16.67%) | 900 (25.00%) | - |
| user2 Test | | 36718 | 31700 (86.33%) | 778 (2.12%) | 2790 (7.60%) | 0 (0.00%) | 1450 (3.95%) | - |
| user3 Train | 93 | 3350 | 750 (22.39%) | 1000 (29.84%) | 500 (14.93%) | 500 (14.93%) | 600 (17.91%) | - |
| user3 Test | | 74731 | 27444 (36.72%) | 32065 (42.91%) | 5779 (7.73%) | 3488 (4.67%) | 5955 (7.97%) | - |
| user4 Train | 36 | 2600 | 600 (23.08%) | 600 (23.08%) | 400 (15.38%) | 600 (23.08%) | 400 (15.38%) | - |
| user4 Test | | 8322 | 5134 (61.69%) | 328 (3.94%) | 1125 (13.52%) | 610 (7.33%) | 1125 (13.52%) | - |

the system's normal operation, the data acquired by the device sensors are sent every minute to the paired smartphone, which in turn dispatches it to the server to be stored and analyzed.

The data used to perform these experiments were collected by four users, two males and two females, at their homes for two months. During this period, the users manually reported many intervals of time at which they where in a particular room performing activities of their daily living. This labeled information constitutes the test data used to assess the accuracy of the predictions. Table 4.1 shows some of the characteristics of each data set, such as the number of data points for each room, the total number of access points detected, or the number of rooms that were selected by each user.

## 4.5 Data exploration

Since data has been labeled at room level and given that usually rooms are separated by walls that attenuate the perceived intensity of the WiFi signal, the feature space of the data should reflect this, that is, we should be able to find a way to separate the RSSI data into a number of clusters equal to the number of labels. Each one of these clusters is formed by signals that have high similarity among them but are dissimilar to signals in other clusters. Therefore, we can have a measure of the predictive quality of the data by finding these clusters and comparing them to the actual labels. The more similar the clusters are to the labels, the more feasible it will be for a machine-learning algorithm to find these discriminative patterns between classes and achieve a good classification accuracy.

The well-known k-means clustering algorithm works by grouping data into a given number of clusters by calculating the Euclidean distance among data instances and assigning each observation to the cluster with the nearest mean. The algorithm iteratively minimizes within-cluster squared Euclidean distances until the solution converges, that is, there are no changes from the previous iteration, or until the maximum number of iterations have been reached.

In order to find if the training data is well segmented and to know if we can expect good classification accuracy, we apply the k-means algorithm to the data from each user and then compare the obtained clusters with the actual labels. Figure 4.4 shows four heatmaps with the results, with darker colors representing higher room-cluster correlation and values denoting percentage. A perfect correlation would show 100% on each diagonal value, this is, each cluster being composed only by data from the correct label (room).

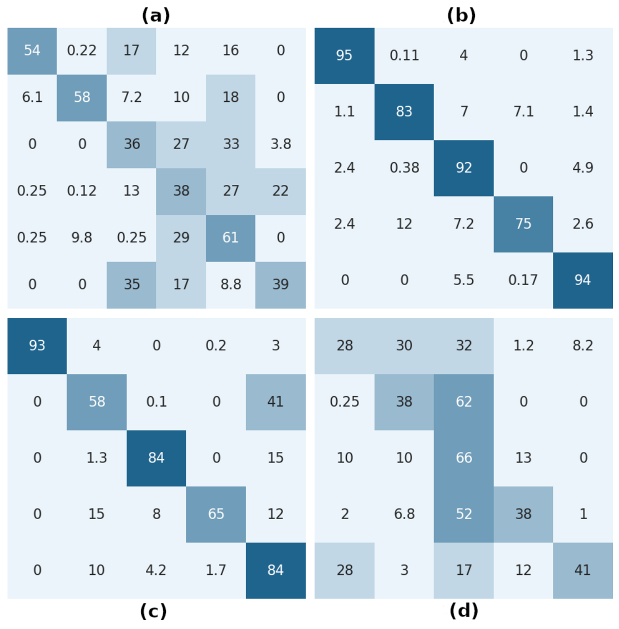


Figure 4.4: Confusion matrices for users' clustered data. The vertical axis represents true labels, the horizontal axis represents predicted labels. (a) user 1, (b) user 2, (c) user 3, (d) user 4

Since users usually did not spend the same time in all rooms, the train and test data set may be imbalanced. Therefore, we adopt the f1 metric as the metric for classification performance, since it is more resilient than accuracy on imbalanced data sets:

Table 4.2: Accuracy and f1 metrics for k-means clustering of training data

| user | accuracy | f1 |
|---|---|---|
| user1 | 0.4769 | 0.4950 |
| user2 | 0.8801 | 0.8792 |
| user3 | 0.7674 | 0.7740 |
| user4 | 0.4220 | 0.4291 |

$$f_1 = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

When more than two classes are considered, we report the weighted average of the individual f1-scores of all classes as the evaluation metric for each model:

$$\hat{f}_1 = \frac{\sum_{i=1}^{c} w_i f_1^i}{\sum_{i=1}^{c} w_i}$$

where $c$ is the number of classes and $w_i$ is the weight (the number of instances) of the $i^{th}$ class.

The calculated metrics for each user, shown in table 4.2, can be seen as a predictor of the quality of data. Higher values will indicate well defined boundaries between classes, revealing potentially useful hidden predictive information that will make it easier for a classifier to assign the correct room to a new instance of data. From these results, it is clear that the k-means clustering algorithm was able to find a better partitioning of the data space for users 2 and 3 than for users 1 and 4. Therefore, we may expect better classification results for these users.

Users 2 and 3 are also users with a higher number of WiFi access points detected, which could partially explain the results obtained. Since the sensor used to record the data is the same for all users and leaving aside factors such as each particular house layout, which are unknown, we can assume that the number of WAPs clearly influences the ability of the k-means algorithm to differentiate between classes.

A visual representation can make it easier to detect meaningful patterns and outliers in groups of data. To be able to find a structure in data in a way that can be visualized we need to reduce its dimensionality while trying to keep most of the knowledge. There are many techniques available to automatically reduce the complexity of high dimensional data. Some of these techniques are:

- Principal Components Analysis (PCA) is an unsupervised technique that finds the components that hold most of the variance (information) of the data. Each component has both direction and magnitude. The direction represents across which principal axes the data has most variance, and the magnitude expresses the amount of variance that is captured of the data when projected onto that axis. Each subsequent principal component is orthogonal to the previous and has less variance. The final result is a set of uncorrelated principal components.
- Linear Discriminant Analysis (LDA) identifies a suitable low-dimensional representation of original data by finding not only the component axes that maximize the variance of the data (PCA) but also the axes that maximize the separation between multiple classes, thus maintaining the class-discriminatory information. LDA is a supervised technique since it needs label information to determine a suitable feature space in order to distinguish between patterns that belong to different classes.
- t-Distributed Stochastic Neighbor Embedding (t-SNE) is an unsupervised, non-linear technique primarily used for exploration and visualization of high-dimensional data. It differs from PCA by preserving only small pairwise distances or local similarities whereas PCA preserves large pairwise distances to maximize variance. The algorithm calculates a similarity measure between pairs of instances in the high dimensional space and in the low dimensional space and tries to minimize the difference between these two similarity measures using gradient descent and the Kullback-Liebler divergence (KL) as the cost function.

Figure 4.5 shows the visualization obtained for each user and algorithm. PCA does not seem to reveal any clear pattern for any user. For users 2 and 3 there are some rooms that seem to be well segmented, but there's still some confusion with the remaining groups. With respect to LDA, it has been able to find a good separation between classes for dataset 1, and specially for 2 and 3. For user 4 the representation found looks more cluttered. And finally, the t-SNE algorithm shows some structure for datasets 2 and 3, where we can visualize a clear separation between some classes. On the other hand, the plots corresponding to data from users 1 and 4 looks more chaotic.

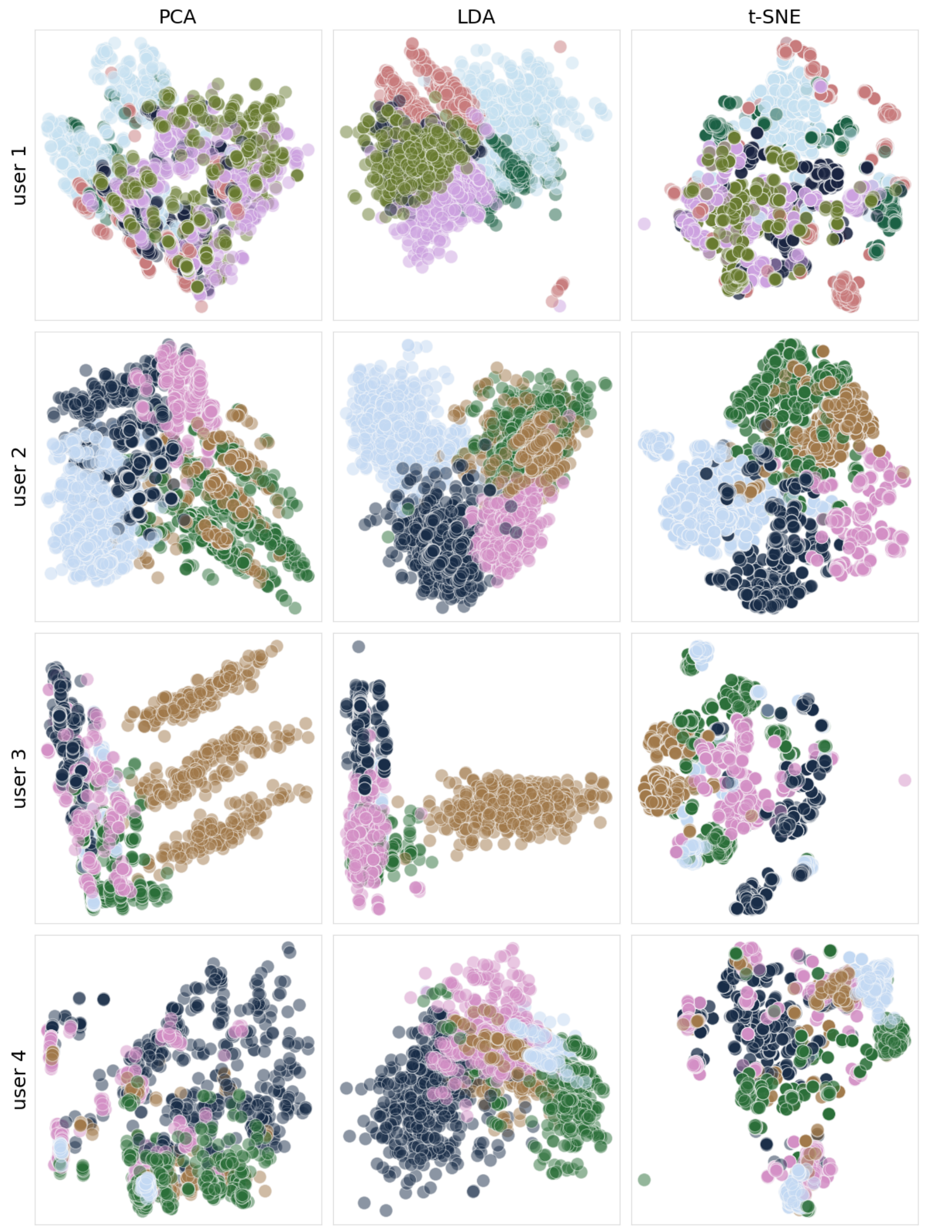


Figure 4.5: Visualization of data using feature reduction algorithms (PCA, LDA and t-SNE)

In order to get a numeric evaluation of these figures, we use the Silhouette metric [134]. The silhouette analysis can be used to study the separation distance between the resulting clusters, as a measure of the quality of clustering achieved. This value measures the space between clusters with a value in the range -1 to 1. If cluster cohesion is good and cluster separation is good, the value will be close to 1. On the other hand, if samples have been assigned to the wrong clusters, the score will be near to -1. Figure 4.6 shows the values obtained for this metric for each one of the algorithms used for visualization.

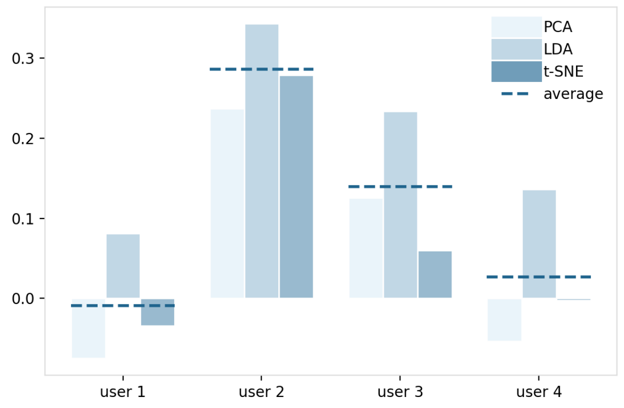


Figure 4.6: Silhouette metric for PCA, LDA and t-SNE algorithms. Dashed line shows the average result for each user.

The conclusions that arise from these visualization and silhouette plots are consistent with the results obtained with the k-means clustering. We can expect machine learning classifiers to have more difficulty finding discriminative patterns for those data sets on which clustering and visualization techniques have not been able to find significant differentiation among groups of instances belonging to distinct rooms. In particular, silhouette values predict a better classification accuracy for users 2 and 3 with respect to users 1 and 4, for which the algorithm could not find clear boundaries between clusters.

## 4.6 Experiments description

The goal of the experiments is to evaluate the influence of a set of parameters in the accuracy of the positioning system, as well as to assess the impact of considering the lack of motion as a constraint for its predictions. Each experiment consists on the evaluation of the classification metric for a particular data set and a given set of parameter values. The parameters that will determine each experiment are the following:

- **Classifier**. A total of four classification algorithms have been tested: Decision Tree (DT), k-Nearest Neighbors (kNN), Neural Net (NN) and Random Forest (RF). The best parameters for each classifier where determined through a series of tests before the experiments:
  - DT: *max. depth = 20*
  - kNN: *k = 3, distance = euclidean*
  - NN: *5 hidden layers, units = 50, act = RELU*
  - RF: *max. depth = 20, max nodes = 50*
- **Scaling**. The RSSI values from the Wi-Fi fingerprints are usually in the range (-100, -30). One common strategy [157] to ease the work of classification algorithms and increase their performance is to scale those values into the range (0, 1), where 0 would mean that the WAP is not present in the fingerprint, and 1 would represent the maximum value for a RSSI (see equation 4.1). We compare the performance of this strategy against feeding the classifiers without pre-processing the data.

$$RSSI_{scaled} = \frac{RSSI + 100}{70} \tag{4.1}$$

- **Reducing stochasticity** As stated in Section 4.3, the IPS described in this work performs a number of consecutive scans to minimize the impact of uncertainty in the RSSI values of Wi-Fi fingerprints. This scan instances, five by default, are passed to the classifiers and the predictions are determined by a majority vote. We assess the performance impact of this strategy against classifying only the first scan instance.
- **Minimum Interval without Significant Motion** (MISM) To improve indoor localization accuracy, specially in room level applications, motion sensors can play an important role, since the detection of steps or any significant motion could be used to discover transitions from one room to another. If a body worn sensor does not register a significant motion in

a given period of time, it can be supposed that the user has not changed his/her location. In this scenario, all fingerprints received during this interval must correspond to the same room. Knowing this, the most reasonably procedure may seem to take the locations estimated by the IPS for that period and assume that all occurrences correspond to the location that occurs more frequently.

As an example, let's consider the user gets into the living room and sits on the sofa. Since the user was moving, we have a signal from the SMS. Now she stays on the sofa for 30 minutes and then goes to the bathroom. When she gets up from the sofa, we receive another signal from the SMS. We know that she was in the same room for 30 minutes, but we do not know which room it is. During this time, the WiFi sensor has been sending signals every minute, so we have 30 fingerprints. Let's suppose now that inferring the position of the user from the WiFi signals gives us these results: 22 in the living room, 6 in the kitchen, 2 in the bathroom. It is safe to assume, given the fact that we know that she did not move, that she was in the living room?

The MISM parameter represents the minimum period to consider when recognizing intervals at which the user has not made a significant movement, and therefore, is supposed to be in the same room. We considered 20 different intervals, between 10 and 200, in steps of 10 minutes.

– **Prediction Threshold** (PT) During the interval of time in which the user stays in a particular room, the IPS generates a series of predictions, specifically, one per minute. Due to the particularities of Wi-Fi signals, environment changes or user orientation, the predictions produced by the classification algorithm for this period may not be uniform and contain different predicted rooms. If we assume that the user has not changed his/her position, the best policy to determine the actual position may be to select the most commonly occurring prediction. In this experiment we evaluate the performance gain of following this strategy, in relation to the ratio of the most occurring prediction over the total number of predictions. To this end, we considered 50 values for this ratio, in the range (0.50, 1.00) with a step value of 0.01. A ratio value of 0.50 would correspond to a case where the most occurring prediction corresponds to the 50% of the IPS predictions. A ratio of 1.00 would correspond to a case where all predictions are equal. In this case, there would not be any improvement when considering motion sensors to amend the IPS predictions.

This configuration gives a total of 16000 experiment for each data set. This figure is broken down as follows:

$$4_{\ classifiers} \times 2_{\ scaling} \times 2_{\ maj.vote} \times 20_{\ MISM} \times 50_{\ PT} = 16000$$

To compare results to determine the best parameter values we used the Wilcoxon Signed-Rank Test [167], a paired difference test to evaluate the mean ranks differences that we applied to the f1 metric.

## 4.7 Results

Figure 4.7 shows a boxplot presenting the results obtained for each classifier on the four data sets. The Random Forest algorithm seems to perform better in all scenarios. To detect significant differences between the performance of the four algorithms and determine the most reliable option, we apply Wilcoxon signed rank test as a statistical method for testing the differences among the outcomes. The results of the test, comparing RF algorithm versus each one of the remaining classification algorithms for each data set, are shown in Table 4.3. In all cases, the Null Hypothesis ($H0$) of equivalence of means can be rejected ($p$-value $<$ 0.05). Therefore, the experimental results show an improved performance in room detection when using the RF algorithm.

These results also endorse the results obtained in Section 4.5. The Random Forest classifier obtains better results in the test data for those data sets that showed a significant structure or pattern when using a clustering or visualization technique. In particular, the best results are obtained for users 2 and 3, that show an average f1 of 0.88 and 0.89, respectively. On the other hand, results for users 1 and 4, with an average f1 of 0.83 and 0.76, also confirm the intuition that the correlation between clustering groups and rooms in training data is a good predictor for the performance of the positioning system on test data.

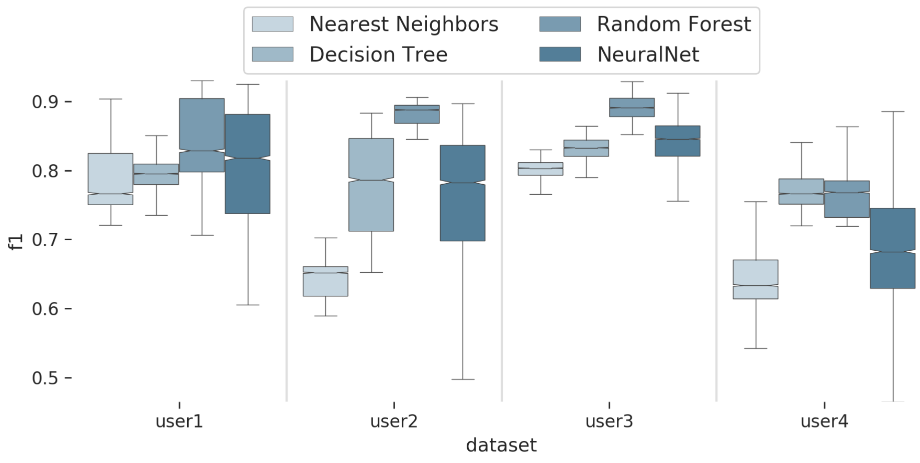


Figure 4.7: Boxplot of the f1 metric for each classifier algorithm and each data set.

Figure 4.8 displays a boxplot comparing the classification effectiveness of scaled data versus raw data. Table 4.4 shows the results of the Wilcoxon signed rank test used to compare the results. The outcome shows that for all the data sets the Null Hypothesis ($H0$) can be rejected, showing that scaling the data increments the performance of the algorithms.

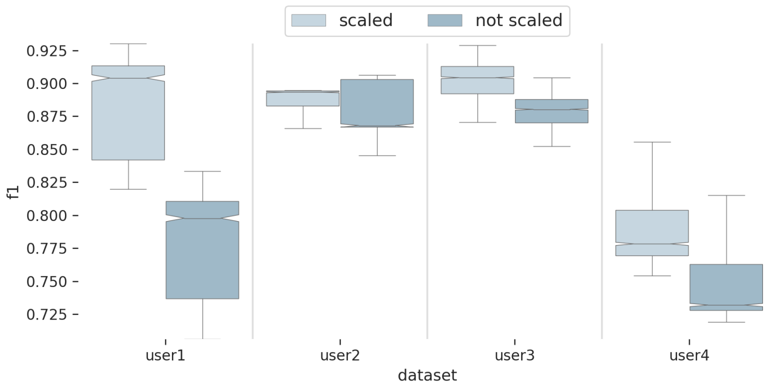


Figure 4.8: Boxplot of the f1 metric for scaled and raw data.

With regards to diminishing the influence of the Wi-Fi signal stochasticity, figure 4.9 displays a boxplot evaluating the use of the majority vote strategy applied to the consecutive samples acquired by the Wi-Fi sensors during each scan process. As is shown in Table 4.5, for all data sets the Null Hypothesis is rejected, indicating that a majority vote strategy significantly increments the accuracy of the positioning algorithm.

Table 4.3: Results of Wilcoxon signed rank test for algorithm comparison between RF and each other ML algorithm

| Data set | Algorithm | $p$-value | $H0$ rejection |
|---|---|---|---|
| user1 | DT | $3.2096 \times 10^{-165}$ | Yes |
| user1 | kNN | $3.2359 \times 10^{-165}$ | Yes |
| user1 | NN | $4.6380 \times 10^{-98}$ | Yes |
| user2 | DT | $2.3567 \times 10^{-167}$ | Yes |
| user2 | kNN | $2.0149 \times 10^{-165}$ | Yes |
| user2 | NN | $1.5510 \times 10^{-163}$ | Yes |
| user3 | DT | $3.3173 \times 10^{-165}$ | Yes |
| user3 | kNN | $3.3194 \times 10^{-165}$ | Yes |
| user3 | NN | $3.3259 \times 10^{-165}$ | Yes |
| user4 | DT | $0.035$ | Yes |
| user4 | kNN | $2.1312 \times 10^{-165}$ | Yes |
| user4 | NN | $4.9099 \times 10^{-131}$ | Yes |

Table 4.4: Results of Wilcoxon signed rank test for scaling strategy

| Data set | $p$-value | $H0$ rejection |
|---|---|---|
| user1 | $0$ | Yes |
| user2 | $1.6835 \times 10^{-138}$ | Yes |
| user3 | $0$ | Yes |
| user4 | $0$ | Yes |

Table 4.5: Results of Wilcoxon signed rank test for majority vote strategy

| Data set | $p$-value | $H0$ rejection |
|---|---|---|
| user1 | $2.6553 \times 10^{-165}$ | Yes |
| user2 | $1.1363 \times 10^{-166}$ | Yes |
| user3 | $3.2828 \times 10^{-165}$ | Yes |
| user4 | $2.6529 \times 10^{-164}$ | Yes |

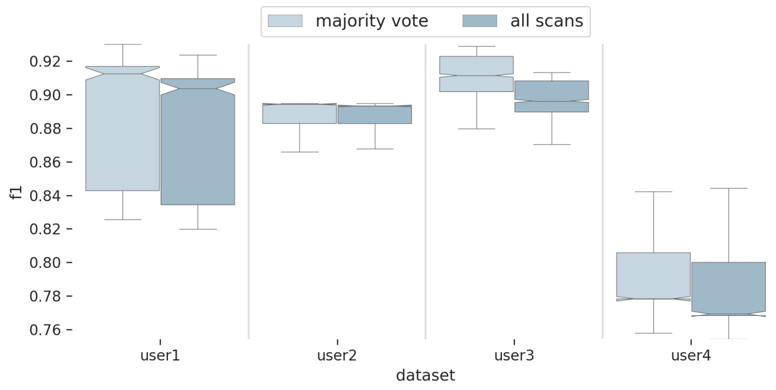


Figure 4.9: Boxplot of the f1 metric for majority vote strategy versus considering all Wi-Fi scans.

The previous tests helped to determine the best positioning algorithm and strategies to improve the positioning accuracy. In order to assess the impact of considering the SMS data to further improve the performance of the IPS, we used scaling and majority vote strategies, and RF as the selected classifier. Figure 4.10 shows the average performance increase obtained depending on the value for the PT parameter, for all possible values of MISM.

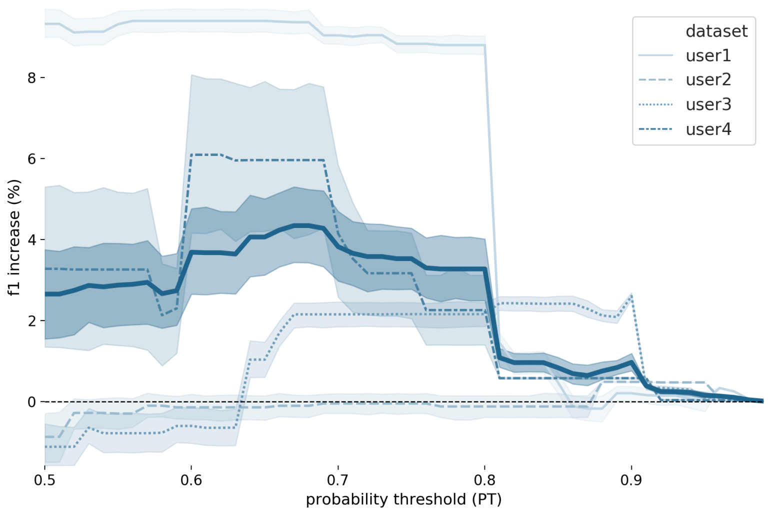


Figure 4.10: f1 average increase versus prediction threshold (PT) for all values of MISM. Each line represents a different data set and the confidence interval of the result. The thick line marks the average value and its confidence interval.

The averaged results show an f1 increase of around 3% in the range of 0.50 to 0.8. Therefore, results suggest that it is safe to assume the most predicted class as the outcome for the positioning system for a given period of time with no motion detected. Moreover, the performance increase is mostly independent of the MISM parameter.

The results for each particular data set show more variability in the range of PT between 0.5 and 0.7, where the accuracy cost of making an incorrect prediction is greater. This variability may be caused by the different house distribution of rooms on each data set. Spaces like open plan kitchen/dining rooms may need additional information, such as the use of Bluetooth beacons or magnetic field sensors, to help the IPS discriminate areas in the same open space. Nevertheless, the maximum cost in the accuracy of incorrect predictions is around 1%, and it occurs only for PT values lower than 0.65. Hence, for PT values greater than 0.65, there is a general performance increase in positioning system when using the SMS as an indicator of room/position changes.

In the process of continuously improving the performance of the positioning system, and with the goal of assessing the validity of these findings, we scheduled a new round of data collection four months after the data used for the previous experiments were recorded. These new data sets were recorded by seven elder users, two females and five males, who performed the training process while following the indications showed by the application. In the same way as with the previous data set, the process was conducted at their homes, where they used the positioning system during a period that varies between one and two months. Following the conclusions arisen from the previous experiments, we used these new data to validate the method of using the SMS as a landmark to detect possible room changes. The results are shown in Figure 4.11.

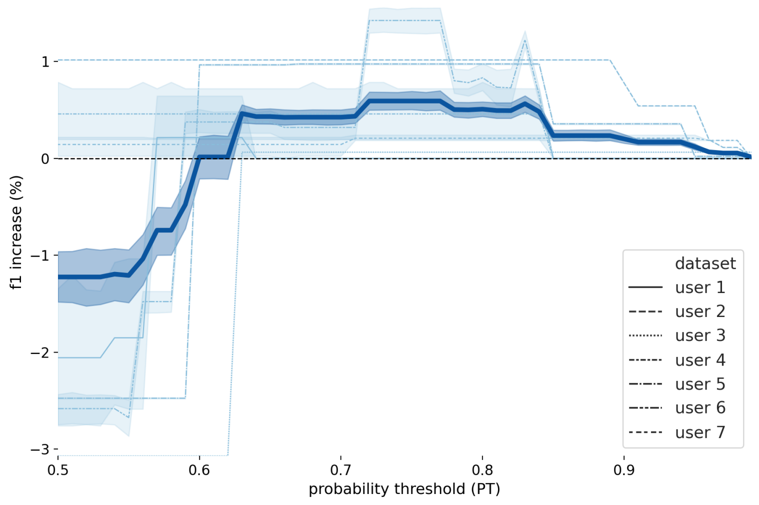


Figure 4.11: Results for stage 2; f1 average increase versus prediction threshold (PT) for all values of MISM. Each line represents a different data set and the confidence interval of the result. The thick line marks the average value and its confidence interval.

The results for the second stage show a similar pattern to the results shown earlier, but with increased variability in the results in the range of PT between 0.5 and 0.7. In this interval, there is not a general gain in performance, since the majority of the users report a decrease in the accuracy. This behavior was already detected for two data sets in the first stage, and now it happens for four out of seven users. As has been discussed earlier, this drop of performance in this interval of PT values can be expected, since it is risky assuming that the correct room can be predicted with only 50-65% of occurrences in a given period of time. For PT values greater than 0.65, the results validate the proposed approach, since there is a general gain in performance for all users.

## 4.8 Conclusions and future work

The experiments presented in this paper show an improved accuracy in room detection when using strategies such as data scaling and the use of consecutive Wi-Fi scanning. The results also demonstrate that the use of a significant motion sensor along with the Wi-Fi fingerprints can help to significantly increase the performance of indoor positioning systems.

As future work, more data from a variety of new users is being collected and will be used to validate the conclusions of this work, while providing more data to test new strategies such as the of the step counter and the activity recognition API to improve positioning accuracy and the use of the magnetic field readings to assess the possibility of determining the position of the user within the room.

## 4.9 Acknowledgements

This work has been partially funded by the Spanish Ministry of Science, Innovation and Universities through the "Retos Investigación" programme (RTI2018-095168-B-C53) and by the Universitat Jaume I “Pla de promoció de la investigació 2017” programme (UJI-B2017-45). Oscar Belmonte-Fernández had a grant from the Spanish Ministry of Science, Innovation and Universities (PRX18/00123) for developing part of this work.

A preliminary version of this work, entitled *"Improving positioning accuracy in Ambient Assisted Living environments. A multi-sensor approach."* was presented at the *15th International Conference on Intelligent Environments (IE19).* With respect to the preliminary version, we have extended and partially rewritten all sections, and added a new section dedicated to exploring data characteristics and their relationship with the performance of the classification algorithms used in the experiments.

We also would like to thank Pilar Bayarri Iturralde for her contribution to the organization of the experiments.

# Chapter 5

# Discussion and Conclusions

## 5.1 Discussion and Conclussions

The main progress in indoor location systems has been made during the last years. Therefore, both research and commercial products in this area are new. Researchers and industry are currently involved in the investigation, development, and improvement of solid indoor positioning systems. In chapters 1 and 2 we presented a set of tools aimed to propose a base framework for fingerprinting-based indoor positioning researchers to learn basic algorithms, contribute with new data sets, share methods and data and build indoor positioning systems based on a robust open-source library. These resources have been designed to be easily extensible, to allow future versions to implement new algorithms and tools to provide a growing reference for researchers.

The R language is a reliable environment for machine learning and data analysis related research. As its popularity is constantly growing, many researchers related to indoor positioning have explicitly selected R as the framework on which to develop their experiments. Even though Python and Matlab have traditionally been the languages of choice for many researchers in the indoor positioning area, the R language is continuously growing and is maintained by an active community. The number of packages available in the CRAN repository has been growing exponentially since its creation (see Figure 5.1), providing a rich ecosystem of good-quality packages that leverage

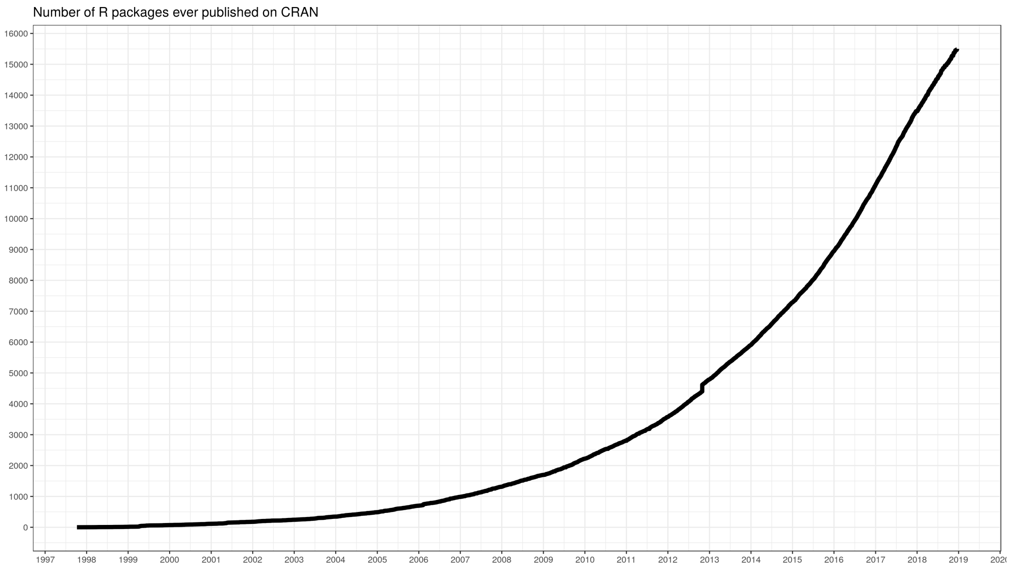


Figure 5.1: Number of packages in the CRAN repository by year.

the R language potential to become a standard programming platform for researchers of all fields. Although there are some open source applications and frameworks to build indoor positioning services, there is not any public framework or package that provides functions and algorithms to manipulate fingerprinting data sets and experiment with positioning algorithms.

The R package `ipft` is an open-source project to which any interested researcher can contribute. It aims to make scientific research in the field accessible to all levels, from amateurs or professionals. It has been designed as a collection of algorithms and utility functions to create models, make estimations, analyze and manipulate RSSI fingerprint data sets for indoor positioning. Given the abundance of potential applications for indoor positioning, the package may have broad relevance for researchers in fields such as pervasive computing, the Internet of Things (IoT), or healthcare, among many others.

Future updates of the package may be focused on the implementation of deep learning-based algorithms for indoor positioning. Many deep learning techniques can be exploited to try to obtain better positioning performance. Recurrent neural networks could be used to learn not only spatial but also temporal patterns of the received signals. Deep autoencoders can be implemented as a way to encode fingerprints and reduce their dimensionality to a few set of significant features. Variational and generative extensions of the autoencoder architecture can be of use to better model the stochastic nature of RSSI data. These models can also be applied to generate new training data for deep learning-based classifiers, increasing the robustness of positioning systems, and trying to address problems caused by the heterogeneity of devices and the various ways on which different devices interpret RSSI signals.

To lower entry barriers and accelerate progress in research, as important as the access to open-source frameworks is the availability of good quality data. Acquiring indoor positioning data is a costly process that demands many resources such as the availability of diverse devices or people to record the data. Furthermore, open access to benchmark data sets is important as a way for researchers to objectively measure how well they are doing on a particular problem. The *IndoorLoc Platform* has been conceived as a public hub for comparing and evaluating indoor positioning algorithms and as an intent to assemble a public repository for all types of indoor positioning-related data sets. The proposed web platform can be used to download data sets, learn the basics of indoor positioning interacting with a set of well-known algorithms and studying their source code, test the methods, and even upload results of the user's methods to check their accuracy against the results provided by other methods already included in the ranking, among other functionalities. As well as with the `ipft` package, the philosophy behind this platform is that anybody, from researchers to industry, should have access, and be able to contribute, to all the building blocks of indoor positioning technologies, such as algorithms, software, and data, to study it, modify it and redistribute it to others.

The performance of the platform was tested during the 2017 Fingerprinting-based Indoor Positioning tutorial held in the School of Engineering of the University of Alcalá. As an activity of the course, an indoor localization competition took place using a data set already included in the platform. A total of 15 teams participated in the contest either by uploading their results or using the *Dashboard* section of the platform to test different parameter configurations. In general, participants were able to easily use the platform, mainly the *Data set download*, *Dashboard*, and *Ranking* sections. Almost no queries to the course introduction were produced, showing the effectiveness of the user-centered design applied to the platform.

The second part of this thesis has been dedicated to the study and implementation of some building blocks to improve the performance of indoor positioning systems. As has been stated in the introduction, accurately identifying the activity a user is performing can help indoor positioning systems to increase their accuracy. For example, knowing that a person is climbing up/down the stairs, or going up/down in an elevator, would limit the range of possible locations in a known environment. Thus, certain activities can be used as landmarks for user location, since they can only be performed at certain locations. Besides using this information as a way to determine the user's position, it can also be exploited as a way to improve the robustness of the positioning system over time. If by identifying an activity the system is reasonably sure of the user's position, it can use this information to collect RSSI data at that location and automatically increment the size of the radio map. Being able to detect activities with high accuracy can lead to a collaborative building and maintenance of fingerprinting radio maps.

Human Activity Recognition and Deep Learning are both hot topics, and as such, several works have used DL algorithms to build recognition architectures with state-of-the-art performance. Chapter 3 provides a broad assessment of the performance of five representative DL architectures for human activity recognition. To improve the representativity of the study, the algorithms have been evaluated on a great quantity of heterogeneous data, using ten publicly available data sets with different characteristics. Even though there are several papers assessing the performance of specific DL algorithms for this task, this is the first study to include a comparison of the performance of the five most prominent architectures, and in doing so on such an extensive amount of data. The scope of the comparison is the ability of different algorithms to capture patterns over the raw data, instead of using pre-engineered features. In accordance with the principles of open science, all the code used to perform the experiments has been published to allow researchers to reproduce the experiments and assess the framework.

The final results obtained are not comparable in terms of accuracy with previous works. This is not the goal of this work since many previous state-of-the-art papers present more complex Deep Learning approaches that achieve very good results on their chosen data sets. The purpose of the study is to compare results among the different models designed as representative for different successful Deep Learning architectures. Therefore, even though results may be artificially good given the closed world set up and the similar distributions for training and test data, they are obtained under the same conditions for all the assessed models. This allows for comparing the results obtained to find out which models are better suited for the task, even though the outcomes may not be comparable with state-of-the-art results.

One of the conclusions that arise from the work presented in chapter 3 is that the particular architecture of convolutional neural networks enables a pattern in the input to be recognized in any position. This type of deep learning architecture can successfully capture the temporal dependencies in raw sensor data structured as time series through the application of relevant filters. Given that different people may execute an action with distinct paces (e.g., an elder person may walk at a slower pace than a young person), this scale deformation tolerance enables this architecture to effectively identify the pattern that characterizes the activity regardless of its pace of execution. Other architectures, such as RNN or DBN, do not perform consistently better than CNN and have some disadvantages, such as their computational complexity and higher memory needs. Thus, their performance, faster response, and lower memory footprint make CNN the architecture of choice for wearable and IoT devices, where restrictions both in terms of power consumption, computational capabilities, and memory availability might play a decisive role.

The results presented in Chapter 3 also reveal that when the amount of data is big enough to enable deep learning algorithms to learn significative patterns, their performance is much better than classical machine learning algorithms such as support vector machines, k-nearest neighbors, or random forest.

In addition to helping to recognize human activities, sensors embedded in wearables and IoT devices can be leveraged to improve indoor positioning systems in other ways. Especially, since embedded sensors usually have limited computational power or battery limitations, using complex algorithms may be unsuitable. The experiments described in Chapter 4 demonstrate that the use of a significant motion sensor along with the Wi-Fi fingerprints can help to significantly increase the performance of indoor positioning systems. The significant motion sensor has a low impact both in terms of computational and energy requirements, which makes it a good option for wearable devices with constant monitoring requirements, such as smart-watches or fitness bands. Other strategies such as data scaling and the use of consecutive Wi-Fi scanning can help as well to improve the accuracy of indoor positioning systems.

Leaving aside the aforementioned viewpoints about performance, Chapter 4 also serves as a showcase for a real indoor positioning system infrastructure deployment. In this case, the use case is the monitoring of older adults in their homes. Part of the chapter is centered around explaining the technical aspects and architecture of the deployment and the reasons behind design decisions. For these set of experiments, deployment and training of the positioning system was completed by experts, the researchers that designed the system. However, since this deployment was meant as a first test to assess the capacity and suitability of the proposed architecture, more data from a variety of non-expert users should be collected to validate the conclusions of this work.

Future work will be centered on testing other technologies for indoor localization, such as Bluetooth Low Energy. While ease of deployment is an important factor, using WiFi fingerprinting conditions the performance of the system to the suitability of an infrastructure that can not be directly controlled. Especially in the case of Ambient Assisted Living, the WiFi routers on which the positioning system depends are out of the administration of the final user. The quantity and position of the nearby WiFi base stations have a significant impact on performance in such systems, introducing too much uncertainty with respect to the expected performance of the positioning system. A compromise solution between control of the infrastructure and ease of deployment may be in the Bluetooth technology. Bluetooth Low Energy beacons consume considerably less power than traditional Bluetooth, which allows them to be considered as an alternative to WiFi access points since they can be easily deployed inside the user's house either using electrical outlets or by means of attached batteries.

# Appendix A

# Related publications

This Appendix is a summary of the related works in which I have participated during my Ph.D. studies, that were not included in previous chapters. Section A.1 shows a relation of papers, led by my thesis directors, that have been published in journals indexed by the Journal Citation Report (JCR), and in which I have contributed to a greater or lesser extent. Section A.2 presents the posters and conference papers in which I have been involved during this period. Finally, section A.3 resumes the work I did during my stay at NaverLabs Europe for five months in 2019.

## A.1 Journals with impact

### A radiosity-based method to avoid calibration for indoor positioning systems

Belmonte-Fernández, Óscar; Montoliu, Raúl; Torres-Sospedra, Joaquín; **Sansano-Sansano, Emilio**; Chía-Aguilar, Daniel

Expert Systems with Applications, 2018, vol. 105, p. 89-101 [21]

doi: 10.1016/j.eswa.2018.03.054

jcr: **Q1** (2018) 24/134 in Computer Science, Artificial Intelligence

scopus: **Q1** (2018) 12/195 in Artificial Intelligence

This work proposes the use of the radiosity model to describe the WiFi signal propagation in indoor scenarios. The use of an analytical propagation model allows generating the WiFi radio map used for fingerprinting indoor location systems, reducing acquisition costs in terms of time and people involved in that task. The proposed technique is based on the following hypothesis:

1. Given that WiFi radio waves are an electromagnetic signal, its propagation model can be simulated using the radiosity model.
2. A WiFi radio map can be analytically obtained from the radiosity model.
3. Walls are the most important structural elements to have into account when calculating the radiosity map of the WiFi signal.

The radiosity-based method provides the RSSI level for each point in the floor plan, avoiding the need to manually sampling RSSI WiFi signals at different positions to create the radio map. Additionally, removing manual data acquisition reduces the cost of creating WiFi maps. This might ease the use of positioning-based Expert Systems development in big scenarios where WiFi sampling is a high time-consuming task.

Experimental results, based on well known machine learning algorithms commonly used in expert systems development, showed that the accuracy of the presented method is close to the manual acquisition of data. Even in those cases where positioning systems are already working, the results presented in this paper show that adding new samples from the radiosity map to real samples improves the final accuracy in almost 10

## Indoor Positioning for Monitoring Older Adults at Home: Wi-Fi and BLE Technologies in Real Scenarios

Montoliu, Raúl; **Sansano-Sansano, Emilio**; Gascó-Compte, Arturo; Belmonte-Fernández, Óscar; Caballer-Miedes, Antonio

Electronics 2020, 9(5), 728 [103]

doi: 10.3390/electronics9050728

jcr: **Q2** (2019) 125/266 in Engineering, Electrical & Electronic

scopus: **Q3** (2019) 376/670 in Electrical and Electronic Engineering

This paper presents a real case of applying an indoor localization system for monitoring older adults in their homes. It showcases some of the problems that arise when real non-expert users deploy an indoor localization system and discuss some strategies to deal with such situations. This indoor localization system is part of a bigger project called *Senior Monitoring*, which is focused on the monitoring of older adults to study behavioral patterns as a tool for early detection of some degenerative diseases such as Alzheimer among others. The system may be useful for healthcare professionals, formal and informal caregivers, and families, especially in an aging society in which the number of older persons and people living alone is increasing.

Participants wear a smart-watch to collect samples using several sensors included in the device. The system has been designed to allow end-users to deploy it in their own homes easily. The performance of the positioning system will depend on the ability of the users to perform this task correctly. The evaluated system has been tested using two different technologies to provide indoor localization; WiFi and Bluetooth Low Energy. The results obtained suggest that the Bluetooth Low Energy-based approach is preferable in the proposed task.

## Anomaly Detection in Activities of Daily Living with Linear Drift

Belmonte-Fernández, Óscar; Caballer-Miedes, Antonio; Chinellato, Eris; Montoliu, Raúl; **Sansano-Sansano, Emilio**; García-Vidal, Rubén

Cognitive Computation 2020 [20]

doi: 10.1007/s12559-020-09740-6

jcr: **Q1** (2018) 25/134 in Computer Science, Artificial Intelligence

scopus: **Q1** (2019) 60/636 in Computer Science Applications

Recognition of changes in Activities of Daily Living (ADL) is a key aspect for e-health applications and is mainly performed by applying machine learning techniques to data provided by an in-home sensor network. In some cases, changes in ADL can signal a change in the person's health, either physical or

cognitive. Physical deterioration can be gradual, as well as cognitive decay, and may be difficult to detect.

This work models ADL as circular normal probability distributions and assesses the validity of this approach by performing experiments on two public datasets. In order to detect a change point in ADL, this study proposes extending the CUSUM algorithm in a way that enables the detection of a linear trend in such activities and using the Maximum Likelihood Estimation (MLE) algorithm to estimate the change point. Some examples of the application of the proposed method are: detection and estimation of the onset of cognitive or physical decline in monitored older adults, detection and estimation of a change in ADL as a result of a change in the medication.

The validity of these schemes is assessed through a set of extensive simulations that cover two main cases: abrupt change and linear drift change. Experimental results show the validity of the method in detecting and estimating the change point in time.

## A.2 Conferences

### A novel methodology to estimate a measurement of the inherent difficulty of an indoor localization radio map

**Sansano-Sansano, Emilio**; Montoliu, Raúl; Torres-Sospedra, Joaquín.

Published in: 2017 8th International Conference on Indoor Positioning and Indoor Navigation (IPIN) [139]

Date of Conference: 18-21 Sept. 2017

doi: 10.1109/IPIN.2017.8115939

The variables used to measure indoor localization methods' accuracy are dependent on the radio map used to test them. The estimated error for a positioning technique strongly depends on the characteristics of the scenario where it has been tested. Variables such as the dimmensions of the scenario, the number of available beacons or their positions, among many others, may affect the expected error for a given data set. Thus, since the estimated error not only depends on the method performance, but it is strongly related to the scenario attributes, a definitive conclusion on the method's performance cannot be obtained. This makes it hard to compare different methods' results.

This work presents a novel methodology to obtain a measure of the inherent difficulty of a scenario to obtain accurate localization results when testing an indoor positioning method. The proposed indicator can be used to obtain a difficulty measure from a fingerprinting data set. This indicator will show if the precision obtained with a positioning method, using that data set, can be considered a reliable measurement of the method's performance. It can be used to provide a measurement of how difficult it would be to get a good position estimation using a given data set of fingerprints. Therefore, it provides a fairer way to compare the performance of two different algorithms evaluated in different environments, as long as the difficulty of the respective radio maps used to test their accuracy is similar. It can also be used as a metric to measure the impact of any modification of the radio map, as the addition or removal of observations.

## A New Methodology for Long-Term Maintenance of WiFi Fingerprinting Radio Maps

Montoliu, Raúl; **Sansano-Sansano, Emilio**; Belmonte-Fernández, Óscar; Torres-Sospedra, Joaquín.

Published in: 2018 9th International Conference on Indoor Positioning and Indoor Navigation (IPIN) [102]

Date of Conference: 24-27 Sept. 2018

doi: 10.1109/IPIN.2018.8533825

Despite the benefits of WiFi fingerprinting techniques, one of the main problems of this technique for indoor positioning systems is the radio map maintenance. It is well known that the creation of the radio map is a tedious and long-time task. Besides, if after its creation, some access points are removed from the environment, the accuracy of the system can be dramatically affected. The fingerprint used to locate the user could be composed of values produced by a set of access points different from the one used to create the radio map. A common approach to deal with this situation is to use just the common access points received at the two different moments in time. This process discards the use of some useful information and therefore, the accuracy of the IPS can be drastically reduced.

This work proposes a new methodology to deal with this problem using regression-based imputation techniques. The main hypothesis is that there is a relationship in the signal strength values obtained for each access point with respect to the other existing access points in the environment. The regression techniques can take advantage of these correlations to impute a valid RSSI value for the removed access point. This paper presents an extensive set of experiments comparing different imputation techniques to demonstrate the benefits of using the proposed approach, showing that it can reduce the localization error in almost one meter with respect to a well-known solution.

### Improving Positioning Accuracy in Ambient Assisted Living Environments. A Multi-Sensor Approach

**Sansano-Sansano, Emilio**; Belmonte-Fernández, Óscar; Montoliu, Raúl; Gascó-Compte, Arturo; Caballer-Miedes, Antonio; Bayarri-Iturralde, Pilar

Published in: 2019 15th International Conference on Intelligent Environments (IE) [138]

Date of Conference: 24-27 June 2019

doi: 10.1109/IE.2019.00004

The primary purpose of this research is to examine the viability of leveraging other sensors in aiding a WiFi fingerprinting-based positioning system to provide more accurate predictions. In particular, the experiments presented in this paper show that the use of Inertial Motion Units (IMUs), which are present by default in smart devices such as smart-phones or smart-watches, can increase the performance of indoor positioning systems in AAL environments. Furthermore, this paper assesses complementary strategies such as data scaling and the use of consecutive WiFi scanning to further improve the reliability of the indoor positioning systems' predictions. This research shows that a robust positioning estimation can be derived from such strategies.

Moreover, this can be done without compromising important aspects such as battery duration or unobtrusiveness. This work also explores possible actions to reduce the influence of WiFi signal uncertainty as well as to select the most appropriate machine learning algorithm for the positioning system.

The results obtained from the set of experiments presented in this work show an improved accuracy in room detection when using strategies such as data scaling and the use of consecutive WiFi scanning. The results also demonstrate that the use of a significant motion sensor along with the WiFi fingerprints can help to significantly increase the performance of indoor positioning systems.

## Evaluation of Crowdsourcing Wi-Fi Radio Map Creation in a Real Scenario for AAL Applications

Belmonte-Fernández, Óscar; Gascó Compte, Arturo; **Sansano-Sansano, Emilio**; Quinde, Mario; Giménez Manuel, José Ginés; Augusto Juan Carlos.

Published in: 2019 15th International Conference on Intelligent Environments (IE) [23]

Date of Conference: 24-27 June 2019

doi: 10.1109/IE.2019.00005

Technology can be integrated into the health care of senior citizens to provide safe, high-quality lives, improving their health and happiness, and enabling a longer period of independent living. Indoor positioning technologies are destined to play an important part in these applications.

One of the main drawbacks of WiFi fingerprinting methods is the temporal cost involved in creating a radio map. Crowdsourcing strategies have been presented as a way to minimize the cost of radio map creation. This research presents an extensive study of the issues involved when using crowdsourcing strategies for that purpose. The results provided by extensive experiments performed in a real scenario by three users during two weeks show how joining data gathered by different users can improve the accuracy performance of a WiFi-based indoor location.

To prevent issues related to device diversity, the same device model was used by all users. To assess the location accuracy, the KNN, Bayes Network, and Random Forest machine learning algorithms have been tested. The results obtained in this study allow us to conclude that crowdsourcing data improves the accuracy of the location. The results show the feasibility of crowdsourcing data to create radio maps for the indoor location. One the second hand, accuracy decay along time was reported.

## Senior Monitoring: A Real Case of Applying a WiFi Fingerprinting-based Indoor Positioning Method for People Monitoring

Montoliu, Raúl; **Sansano-Sansano, Emilio**; Gascó-Compte, Arturo; Belmonte Fernández, Óscar; Caballer-Miedes, Antonio.

Published as *Work-in-Progress* paper in: 2019 10th International Conference on Indoor Positioning and Indoor Navigation (IPIN) [100]

Date of Conference: 30 September - 3 October 2019

This work showcases a real example of applying a Wi-Fi fingerprinting-based indoor localization system for monitoring elder people in their homes. The presented system is part of a broad project called *Senior Monitoring* where the main aim is to monitor elders to study behavioral patterns as a tool for early detection of some cognitive decay diseases. Since the system is used by real users, many situations can not be controlled by system developers and can be a source of errors. This study presents some of the problems arisen when real non-expert users use localization systems, and discuss some strategies to deal with such situations.

The experiments were conducted by 17 volunteers for two months on average in real scenarios, where the conditions are not controlled by the researchers. Participants had to create the radio map of his/her own house following the instructions provided by the system's developers. The present work contributes to a better understanding of the difficulties and problems that arise when implementing an indoor positioning system in real scenarios with real users.

## A.3 Stays

### NAVER Labs Europe

NAVER Labs Europe in Grenoble (France), supervised by Dr. Boris Chidlovskii from 18 March 2019 to 14 August 2019

During this stay, I worked on the development of an indoor positioning system based on the fusion of data from inertial sensors along with data from other sensors such as WiFi, barometer, or magnetic field. The system implements a deep learning-based pedestrian dead reckoning (deep PDR) model that provides an estimation of the relative position of the user and a WiFi fingerprinting module that provides a prediction of the user's absolute position. Both predictions, relative and absolute, are fused using a Kalman filter, and then projected on the possible paths taking into account the physical constraints (corridors, doors, etc.) of the environment.

The system took part in the off-site smartphone-based positioning track of the IPIN2019 competition, where the goal is to recreate a path traversed by a person holding a conventional modern smartphone, based on the readings from the smartphone's sensors. The system finished in second place of a total of 15 participants.

# Bibliography

[1] AARTS, EMILE and WICHERT, REINER: «Ambient intelligence». In: *Technology guide*, pp. 244–249. Springer, 2009. doi: https://doi.org/10.1007/978-3-540-88546-7_47.

[2] AL-AMMAR, MAI A; ALHADHRAMI, SUHEER; AL-SALMAN, ABDULMALIK; ALARIFI, ABDULRAHMAN; AL-KHALIFA, HEND S; ALNAFESSAH, AHMAD and ALSALEH, MANSOUR: «Comparative survey of indoor positioning technologies, techniques, and algorithms». In: *Cyberworlds (CW), 2014 International Conference on*, pp. 245–252. IEEE, 2014.

[3] ALI, ABDUL HALIM; RAZAK, MOHD RAZIFF ABD; HIDAYAB, MUZAIYANAH; AZMAN, SYUWARI ASHRAF; JASMIN, MOHD ZAIM MOHD and ZAINOL, MOHD AZMIR: «Investigation of indoor WIFI radio signal propagation». In: *Proceedings of the Symposium on Industrial Electronics and Applications, (ISIEA'10)*, pp. 117–119, 2010. doi: https://doi.org/10.1109/ISIEA.2010.5679486.

[4] ALSHEIKH, MOHAMMAD ABU; SELIM, AHMED; NIYATO, DUSIT; DOYLE, LINDA; LIN, SHAOWEI and TAN, HWEE-PINK: «Deep Activity Recognition Models with Triaxial Accelerometers.» In: *AAAI Workshop: Artificial Intelligence Applied to Assistive Technologies and Smart Environments*, , 2016.

[5] ALZANTOT, MOUSTAFA and YOUSSEF, MOUSTAFA: «UPTIME: Ubiquitous pedestrian tracking using mobile phones». In: *2012 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 3204–3209. IEEE, 2012.

[6] Anguita, Davide; Ghio, Alessandro; Oneto, Luca; Parra, Xavier and Reyes-Ortiz, Jorge Luis: «A Public Domain Dataset for Human Activity Recognition using Smartphones.» In: *Proceedings of the European Symposium on Artificial Neural Networks (ESANN'13)*, , 2013.

[7] Aramendi, Ane Alberdi; Weakley, Alyssa; Goenaga, Asier Aztiria; Schmitter-Edgecombe, Maureen and Cook, Diane J: «Automatic assessment of functional health decline in older adults based on smart home data». *Journal of biomedical informatics*, 2018, **81**, pp. 119–130.

[8] Arifoglu, Damla and Bouchachia, Abdelhamid: «Activity recognition and abnormal behaviour detection with recurrent neural networks». *Procedia Computer Science*, 2017, **110**, pp. 86–93.

[9] Ashraf, Imran; Hur, Soojung; Park, Sangjoon and Park, Yongwan: «DeepLocate: Smartphone Based Indoor Localization with a Deep Neural Network Ensemble Classifier». *Sensors*, 2020, **20(1)**, p. 133.

[10] Ashraf, Imran; Hur, Soojung and Park, Yongwan: «Application of Deep Convolutional Neural Networks and Smartphone Sensors for Indoor Localization». *Applied Sciences*, 2019, **9(11)**, p. 2337.

[11] ——: «Indoor positioning on disparate commercial smartphones using Wi-Fi access points coverage area». *Sensors*, 2019, **19(19)**, p. 4351.

[12] Ashraf, Imran; Kang, Mingyu; Hur, Soojung and Park, Yongwan: «MINLOC: Magnetic Field Patterns-Based Indoor Localization Using Convolutional Neural Networks». *IEEE Access*, 2020, **8**, pp. 66213–66227.

[13] Austin, Johanna; Dodge, Hiroko H; Riley, Thomas; Jacobs, Peter G; Thielke, Stephen and Kaye, Jeffrey: «A smart-home system to unobtrusively and continuously assess loneliness in older adults». *IEEE journal of translational engineering in health and medicine*, 2016, **4**.

[14] AVCI, AKIN; BOSCH, STEPHAN; MARIN-PERIANU, MIHAI; MARIN-PERIANU, RALUCA and HAVINGA, PAUL: «Activity recognition using inertial sensing for healthcare, wellbeing and sports applications: A survey». In: *23th International conference on architecture of computing systems 2010*, pp. 1–10. VDE, 2010.

[15] B. LI, C. RIZOS, T. GALLAGHER and DEMPSTER, A.G.: «Using Geomagnetic Field for Indoor Positioning». In: *Proceedings of the International Global Navigation Satellite Systems Society IGNSS Symposium*, , 2013.

[16] BAHL, PARAMVIR and PADMANABHAN, VENKATA N.: «RADAR: an in-building RF-based user location and tracking system». In: *Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'00)*, pp. 775–784, 2000.

[17] BANOS, ORESTI; GARCIA, RAFAEL; HOLGADO-TERRIZA, JUAN A; DAMAS, MIGUEL; POMARES, HECTOR; ROJAS, IGNACIO; SAEZ, ALEJANDRO and VILLALONGA, CLAUDIA: «mHealthDroid: a novel framework for agile development of mobile health applications». In: *Proceedings of the International Workshop on Ambient Assisted Living (IWAAL'14)*, pp. 91–98. Springer, 2014.

[18] BAO, LING and INTILLE, STEPHEN S: «Activity recognition from user-annotated acceleration data». In: *Proceedings of the IEEE International Conference on Pervasive Computing (PerCom'04)*, pp. 1–17. Springer, 2004.

[19] BARSOCCHI, P.; CRIVELLO, A.; ROSA, D.L. and PALUMBO, F.: «A multisource and multivariate dataset for indoor localization methods based on WLAN and geo-magnetic field fingerprinting». In: *Proceedings of the seventh Conference on Indoor Positioning and Indoor Navigation (IPIN'16)*, , 2016.

[20] BELMONTE-FERNÁNDEZ, ÓSCAR; CABALLER-MIEDES, ANTONIO; CHINELLATO, ERIS; MONTOLIU, RAÚL; SANSANO-SANSANO, EMILIO and GARCÍA-VIDAL, RUBÉN: «Anomaly Detection in Activities of Daily Living with Linear Drift». *Cognitive Computation*, 2020, pp. 1–19.

[21] BELMONTE-FERNÁNDEZ, OSCAR; MONTOLIU, RAÚL; TORRES-SOSPEDRA, JOAQUÍN; SANSANO-SANSANO, EMILIO and CHIA-AGUILAR, DANIEL: «A radiosity-based method to avoid calibration for indoor positioning systems». *Expert Systems With Applications*, 2018, **105**, pp. 89–101.

[22] BELMONTE-FERNÁNDEZ, ÓSCAR; PUERTAS-CABEDO, ADRIAN; TORRES-SOSPEDRA, JOAQUÍN; MONTOLIU-COLÁS, RAÚL and TRILLES-OLIVER, SERGI: «An indoor positioning system based on wearables for ambient-assisted living». *Sensors*, 2017, **17(1)**, p. 36.

[23] BELMONTE-FERNÁNDEZ, Ó.; GASCÓ-COMPTE, A.; SANSANO-SANSANO, E.; QUINDE, M.; MANUEL, J. G. G. and AUGUSTO, J. C.: «Evaluation of Crowdsourcing Wi-Fi Radio Map Creation in a Real Scenario for AAL Applications». In: *2019 15th International Conference on Intelligent Environments (IE)*, pp. 30–36, 2019.

[24] BENGIO, YOSHUA; SIMARD, PATRICE and FRASCONI, PAOLO: «Learning long-term dependencies with gradient descent is difficult». *IEEE transactions on neural networks*, 1994, **5(2)**, pp. 157–166.

[25] BHATTACHARYA, SOURAV and LANE, NICHOLAS D: «From smart to deep: Robust activity recognition on smartwatches using deep learning». In: *Proceedings if the IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom'16)*, pp. 1–6, 2016.

[26] BHATTACHARYA, SOURAV; NURMI, PETTERI; HAMMERLA, NILS and PLÖTZ, THOMAS: «Using unlabeled data in a sparse-coding framework for human activity recognition». *Pervasive and Mobile Computing*, 2014, **15**, pp. 242–262.

[27] BISHOP, CHRISTOPHER M.: *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., 2006.

[28] BITEW, MEKUANINT AGEGNEHU; HSIAO, RONG-SHUE; LIN, HSIN-PIAO and LIN, DING-BING: «Hybrid indoor human localization system for

addressing the issue of RSS variation in fingerprinting». *International Journal of Distributed Sensor Networks*, 2015, **11(3)**, p. 831423.

[29] BRENA, RAMON F; GARCÍA-VÁZQUEZ, JUAN PABLO; GALVÁN-TEJADA, CARLOS E; MUÑOZ-RODRIGUEZ, DAVID; VARGAS-ROSALES, CESAR and FANGMEYER, JAMES: «Evolution of indoor positioning technologies: A survey». *Journal of Sensors*, 2017, **2017**.

[30] BROYDEN, CG: «A new double-rank minimisation algorithm. Preliminary report». In: *Notices of the American Mathematical Society*, volume 16, p. 670. American Mathematical Society 201 Charles ST, Providence, RI 02940-2213, 1969.

[31] CAFFERY, JAMES J and STUBER, GORDON L: «Overview of radiolocation in CDMA cellular systems». *IEEE Communications Magazine*, 1998, **36(4)**, pp. 38–45.

[32] CAMPS, JULIÀ; SAMÀ, ALBERT; MART\'\IN, MARIO; RODR\'\IGUEZ-MART\'\IN, DANIEL; PÉREZ-LÓPEZ, CARLOS; AROSTEGUI, JOAN M MORENO; CABESTANY, JOAN; CATALÀ, ANDREU; ALCAINE, SHEILA; MESTRE, BERTA and OTHERS: «Deep learning for freezing of gait detection in Parkinson's disease patients in their homes using a waist-worn inertial measurement unit». *Knowledge-Based Systems*, 2018, **139**, pp. 119–131.

[33] CHEN, YUQING and XUE, YANG: «A deep learning approach to human activity recognition based on single accelerometer». In: *IEEE international conference on Systems, man, and cybernetics (SMC'15)*, pp. 1488–1492. Institute of Electrical and Electronics Engineers (IEEE), 2015.

[34] CHIARINI, GIOVANNI; RAY, PRADEEP; AKTER, SHAHRIAR; MASELLA, CRISTINA and GANZ, AURA: «mHealth technologies for chronic diseases and elders: a systematic review». *IEEE Journal on Selected Areas in Communications*, 2013, **31(9)**, pp. 6–18.

[35] CHIAUZZI, EMIL; RODARTE, CARLOS and DASMAHAPATRA, PRONABESH: «Patient-centered activity monitoring in the self-management of chronic health conditions». *BMC medicine*, 2015, **13(1)**, p. 77.

[36] CHUNG, JUNYOUNG; GULCEHRE, CAGLAR; CHO, KYUNGHYUN and BENGIO, YOSHUA: «Empirical evaluation of gated recurrent neural networks on sequence modeling». *arXiv preprint arXiv:1412.3555*, 2014.

[37] COVER, T. and HART, P.: «Nearest Neighbor Pattern Classification». *IEEE Transactions on Information Theory*, 1967, **13(1)**, pp. 21–27. doi: https://doi.org/10.1109/TIT.1967.1053964.

[38] CRAMARIUC, A. and LOHAN, E.S.: «Open-access WiFi measurement data and Python-based data analysis», 2016. `http://www.cs.tut.fi/tlt/pos/meas.htm`

[39] CRAMARIUC, ANDREI; HUTTUNEN, HEIKKI and LOHAN, ELENA SIMONA: «Clustering benefits in mobile-centric WiFi positioning in multi-floor buildings». In: *Proceedings of the 6th International Conference on Localization and GNSS (ICL-GNSS'16)*, pp. 1–6, 2016. doi: https://doi.org/10.1109/ICL-GNSS.2016.7533846.

[40] DORYAB, AFSANEH; MIN, JUN KI; WIESE, JASON; ZIMMERMAN, JOHN and HONG, JASON: «Detection of behavior change in people with depression». In: *Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence*, , 2014.

[41] DUCHI, JOHN; HAZAN, ELAD and SINGER, YORAM: «Adaptive subgradient methods for online learning and stochastic optimization». *Journal of Machine Learning Research*, 2011, **12(Jul)**, pp. 2121–2159.

[42] DUFFNER, STEFAN; BERLEMONT, SAMUEL; LEFEBVRE, GRÉGOIRE and GARCIA, CHRISTOPHE: «3D gesture classification with convolutional neural networks». In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'14)*, pp. 5432–5436. Institute of Electrical and Electronics Engineers (IEEE), 2014.

[43] ERHAN, DUMITRU; BENGIO, YOSHUA; COURVILLE, AARON; MANZAGOL, PIERRE-ANTOINE; VINCENT, PASCAL and BENGIO, SAMY: «Why does unsupervised pre-training help deep learning?» *Journal of Machine Learning Research*, 2010, **11(Feb)**, pp. 625–660.

[44] FARID, ZAHID; NORDIN, ROSDIADEE and ISMAIL, MAHAMOD: «Recent advances in wireless indoor localization techniques and system». *Journal of Computer Networks and Communications*, 2013, **2013**.

[45] FINNOFF, WILLIAM; HERGERT, FERDINAND and ZIMMERMANN, HANS GEORG: «Improving model selection by nonconvergent methods». *Neural Networks*, 1993, **6(6)**, pp. 771–783.

[46] FLETCHER, ROGER: «A new approach to variable metric algorithms». *The computer journal*, 1970, **13(3)**, pp. 317–322. doi: https://doi.org/10.1093/comjnl/13.3.317.

[47] FREY, BRENDAN J and DUECK, DELBERT: «Clustering by passing messages between data points.» *Science*, 2007, **315(5814)**, pp. 972–976. doi: https://doi.org/10.1126/science.1136800.

[48] GALAMBOS, COLLEEN; SKUBIC, MARJORIE; WANG, SHAUNG and RANTZ, MARILYN: «Management of dementia and depression utilizing in-home passive sensor data». *Gerontechnology: international journal on the fundamental aspects of technology to serve the ageing society*, 2013, **11(3)**, p. 457.

[49] GARCÍA, S.; MOLINA, D.; LOZANO, M. and HERRERA, M. F.: «A Study on the Use of Non-parametric Tests for Analyzing the Evolutionary Algorithms' Behaviour: A Case Study on the CEC'2005 Special Session on Real Parameter Optimization». *Journal of Heuristics*, 2009, **15(6)**, pp. 617–644.

[50] GIGL, THOMAS; JANSSEN, GERARD J.M.; DIZDAREVIC, VEDRAN; WITRISAL, KLAUS and IRAHHAUTEN, ZOUBIR: «Analysis of a UWB Indoor Positioning System Based on Received Signal Strength». In: *Proceedings of the 4th Workshop on Positioning, Navigation and Communication (PNC'07)*, pp. 97–101, 2007. doi: https://doi.org/10.1109/WPNC.2007.353618.

[51] GJORESKI, HRISTIJAN; BIZJAK, JANI; GJORESKI, MARTIN and GAMS, MATJAŽ: «Comparing deep and classical machine learning methods

for human activity recognition using wrist accelerometer». In: *Proceedings of the IJCAI 2016 Workshop on Deep Learning for Artificial Intelligence*, volume 10, 2016.

[52] GOLDBLOOM, A.; HAMNER, B.; MOSER, J. and CUKIERSKI, M.: «kaggle: Your Home for Data Science», 2017.
`https://www.kaggle.com/`

[53] GOLDFARB, DONALD: «A Family of Variable-Metric Methods Derived by Variational Means». *Mathematics of Computation*, 1970, **24(109)**, pp. 23–26. doi: https://doi.org/10.1090/S0025-5718-1970-0258249-6.

[54] GOODFELLOW, IAN; BENGIO, YOSHUA and COURVILLE, AARON: *Deep Learning*. MIT Press, 2016.
`http://www.deeplearningbook.org`

[55] GUAN, YU and PLÖTZ, THOMAS: «Ensembles of deep lstm learners for activity recognition using wearables». *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT'17)*, 2017, **1(2)**, p. 11.

[56] HA, SOJEONG; YUN, JEONG-MIN and CHOI, SEUNGJIN: «Multi-modal convolutional neural networks for activity recognition». In: *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC'15)*, pp. 3017–3022. Institute of Electrical and Electronics Engineers (IEEE), 2015.

[57] HAEBERLEN, A.; FLANNERY, E.; LADD, A.M.; RUDYS, A.; WALLACH, D.S. and KAVRAKI, L.E.: «Practical Robust Localization over Large-scale 802.11 Wireless Networks». In: *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking (ModiCom'04)*, pp. 70–84, 2004.

[58] HAEBERLEN, ANDREAS; FLANNERY, ELIOT; LADD, ANDREW M.; RUDYS, ALGIS; WALLACH, DAN S. and KAVRAKI, LYDIA E.: «Practical Robust Localization over Large-scale 802.11 Wireless Networks». In: *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking (MobiCom'04)*, pp. 70–84, 2004. doi: https://doi.org/10.1145/1023720.1023728.

[59] HAMMERLA, NILS Y; HALLORAN, SHANE and PLOETZ, THOMAS: «Deep, convolutional, and recurrent models for human activity recognition using wearables». *arXiv preprint arXiv:1604.08880*, 2016.

[60] HAN, DONGSOO; JUNG, SUK HOON; LEE, MINKYU and YOON, GI-WAN: «Building a Practical Wi-Fi-Based Indoor Navigation System». *IEEE Pervasive Computing*, 2014, **13**, pp. 72–79.

[61] HANLEY, D.; FAUSTINO, A. B.; ZELMAN, S. D.; DEGENHARDT, D. A. and BRETL, T.: «MagPIE: A Dataset for Positioning with Magnetic Anomalies». In: *Proceedings of the 2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN'17)*, , 2017.

[62] HARBICHT, ANDREW B.; CASTRO-SANTOS, THEODORE; ARDREN, WILLIAM R.; GORSKY, DIMITRY and FRASER, DYLAN J.: «Novel, continuous monitoring of fine-scale movement using fixed-position radiotelemetry arrays and random forest location fingerprinting». *Methods in Ecology and Evolution*, 2017, **8(7)**, pp. 850–859. doi: http://dx.doi.org/10.1111/2041-210X.12745.

[63] HASSAN, MOHAMMED MEHEDI; UDDIN, MD ZIA; MOHAMED, AMR and ALMOGREN, AHMAD: «A robust human activity recognition system using smartphone sensors and deep learning». *Future Generation Computer Systems*, 2018, **81**, pp. 307–313.

[64] HAYES, TAMARA L; ABENDROTH, FRANCENA; ADAMI, ANDRE; PAVEL, MISHA; ZITZELBERGER, TRACY A and KAYE, JEFFREY A: «Unobtrusive assessment of activity patterns associated with mild cognitive impairment». *Alzheimer's & Dementia*, 2008, **4(6)**, pp. 395–405.

[65] HE, SUINING and CHAN, S. GARY: «Wi-Fi Fingerprint-based Indoor Positioning: Recent Advances and Comparisons». *IEEE Communications Surveys & Tutorials*, 2016, **18(3)**, pp. 466 – 490.

[66] HE, SUINING and CHAN, S. H GARY: «Wi-Fi fingerprint-based indoor positioning: Recent advances and comparisons». *IEEE Communications Surveys & Tutorials*, 2016, **18(1)**, pp. 466–490. doi: https://doi.org/10.1109/COMST.2015.2464084.

[67] HE, SUINING and SHIN, KANG G: «Crowd-Flow Graph Construction and Identification with Spatio-Temporal Signal Feature Fusion». In: *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pp. 757–765. IEEE, 2019.

[68] HINTON, GEOFFREY; DENG, LI; YU, DONG; DAHL, GEORGE E; MOHAMED, ABDEL-RAHMAN; JAITLY, NAVDEEP; SENIOR, ANDREW; VANHOUCKE, VINCENT; NGUYEN, PATRICK; SAINATH, TARA N and OTHERS: «Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups». *IEEE Signal Processing Magazine*, 2012, **29(6)**, pp. 82–97. https://doi.org/10.1109/MSP.2012.2205597

[69] HINTON, GEOFFREY E; OSINDERO, SIMON and TEH, YEE-WHYE: «A fast learning algorithm for deep belief nets». *Neural computation*, 2006, **18(7)**, pp. 1527–1554.

[70] HOCHREITER, SEPP and SCHMIDHUBER, JÜRGEN: «Long short-term memory». *Neural computation*, 1997, **9(8)**, pp. 1735–1780.

[71] HUANG, HE; LI, WEI; LUO, DE AN; QIU, DONG WEI and GAO, YANG: «An improved particle filter algorithm for geomagnetic indoor positioning». *Journal of Sensors*, 2018, **2018**.

[72] JIANG, WENCHAO and YIN, ZHAOZHENG: «Human activity recognition using wearable sensors by deep convolutional neural networks». In: *Proceedings of the 23rd ACM international conference on Multimedia*, pp. 1307–1310. ACM, 2015.

[73] JORDAN, MICHAEL I: «Serial order: A parallel distributed processing approach». In: *Advances in psychology*, volume 121, pp. 471–495. Elsevier, 1997.

[74] KAEMARUNGSI, KAMOL and KRISHNAMURTHY, PRASHANT: «Properties of indoor received signal strength for WLAN location fingerprinting». In: *Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004. The First Annual International Conference on*, pp. 14–23. IEEE, 2004.

[75] KANASI, ELENI; AYILAVARAPU, SRINIVAS and JONES, JUDITH: «The aging population: demographics and the biology of aging». *Periodontology 2000*, 2016, **72(1)**, pp. 13–18.

[76] KLEPEIS, NEIL E.; NELSON, WILLIAM C.; OTT, WAYNE R.; ROBINSON, JOHN P.; TSANG, ANDY M.; SWITZER, PAUL; BEHAR, JOSEPH V.; HERN, STEPHEN C. and ENGELMANN, WILLIAM H.: «The National Human Activity Pattern Survey (NHAPS): a resource for assessing exposure to environmental pollutants». *Journal Of Exposure Analysis And Environmental Epidemiology*, 2001, **11**, pp. 231 EP –. doi: http://dx.doi.org/10.1038/sj.jea.7500165.

[77] LARA, OSCAR D; LABRADOR, MIGUEL A and OTHERS: «A survey on human activity recognition using wearable sensors.» *IEEE Communications Surveys and Tutorials*, 2013, **15(3)**, pp. 1192–1209.

[78] LECUN, YANN; BENGIO, YOSHUA and HINTON, GEOFFREY: «Deep learning». *nature*, 2015, **521(7553)**, p. 436.

[79] LECUN, YANN; BOSER, BERNHARD; DENKER, JOHN S; HENDERSON, DONNIE; HOWARD, RICHARD E; HUBBARD, WAYNE and JACKEL, LAWRENCE D: «Backpropagation applied to handwritten zip code recognition». *Neural computation*, 1989, **1(4)**, pp. 541–551.

[80] LECUN, YANN; BOTTOU, LEON; ORR, GENEVIEVE B. and MÜLLER, KLAUS ROBERT: *Efficient BackProp*. pp. 9–50. Springer Berlin Heidelberg. ISBN 978-3-540-49430-0, 1998. doi: https://doi.org/10.1007/3-540-49430-8_2.

[81] LEE, JOO-YUB; YOON, CHEAL-HWAN; PARK, HYUNJAE and SO, JUNG-MIN: «Analysis of Location Estimation Algorithms for Wifi Fingerprint-based Indoor Localization». In: *Proceedings of the 2nd International Conference on Software Technology (SoftTech'13)*, pp. 89–92, 2013.

[82] LEFEBVRE, GRÉGOIRE; BERLEMONT, SAMUEL; MAMALET, FRANCK and GARCIA, CHRISTOPHE: «BLSTM-RNN based 3D gesture classification». In: *Proceedings of the International Conference on Artificial Neural Networks (ICANN'13)*, pp. 381–388. Springer, 2013.

[83] LEMIC, F.; HANDZISKI, V.; WIRSTROM, N.; VAN HAUTE, T.; DE POORTER, E.; VOIGT, T. and WOLISZ, A.: «Web-based platform for evaluation of RF-based indoor localization algorithms». In: *In Proceedings of the 2015 IEEE International Conference on Communication Workshop (ICCW'15)*, , 2015.

[84] LI, B.; GALLAGHER, T.; DEMPSTER, A.G. and RIZOS, C.: «How feasible is the use of magnetic field alone for indoor positioning?» In: *3th International conference on Indoor Positioning and Indoor Navigation*, , 2012.

[85] LI, BINGHAO; SALTER, JAMES; DEMPSTER, AG and RIZOS, CHRIS: «Indoor positioning techniques based on wireless LAN». In: *Proceedings of the 1st IEEE International Conference on Wireless Broadband and Ultra Wide-band Communications (AusWireless'06)*, pp. 13–16, 2006.
`https://opus.lib.uts.edu.au/bitstream/2100/170/1/113_Li.pdf`

[86] LICHMAN, M.: «UCI Machine Learning Repository», 2013.
`http://archive.ics.uci.edu/ml`

[87] LIU, HUI; DARABI, HOUSHANG; BANERJEE, PAT and LIU, JING: «Survey of wireless indoor positioning techniques and systems». *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, 2007, **37(6)**, pp. 1067–1080. doi: https://doi.org/10.1109/TSMCC.2007.905750.

[88] LIU, LILI; STROULIA, ELENI; NIKOLAIDIS, IOANIS; MIGUEL-CRUZ, ANTONIO and RINCON, ADRIANA RIOS: «Smart homes and home health monitoring technologies for older adults: A systematic review». *International journal of medical informatics*, 2016, **91**, pp. 44–59.

[89] LIU, YONGBO; DU, HUAICHANG and XU, YE: «The Research and Design of the Indoor Location System Based on RFID». In: *Proceedings of the 4th International Symposium on Computational Intelligence and Design (ISCID'11)*, pp. 87–90, 2011. doi: https://doi.org/10.1109/ISCID.2011.123.

[90] LOHAN, ELENA SIMONA; KOSKI, KAROLIINA; TALVITIE, JUKKA and UKKONEN, LEENA: «WLAN and RFID Propagation channels for hybrid indoor positioning». In: *Proceedings of the 4th International Conference on Localization and GNSS, (ICL-GNSS'14)*, , 2014. doi: https://doi.org/10.1109/ICL-GNSS.2014.6934184.

[91] LUO, JIAYOU and ZHAN, XINGQUN: «Characterization of Smart Phone Received Signal Strength Indication for WLAN Indoor Positioning Accuracy Improvement». *Journal of Networks*, 2014, **9(3)**, pp. 739–746. doi: https://doi.org/10.4304/jnw.9.3.739-746.

[92] LYMBEROPOULOS, D.; CHOUDHURY, R.R.; YANG, X. and SEN, S.: «Microsoft Indoor Localization Competition (IPSN'14)», 2014. `https://www.microsoft.com/en-us/research/event/microsoft-indoor-localization-competition-ipsn-2014`

[93] LYMBEROPOULOS, D.; LIU, J.; BOCCA, M.; SEQUEIRA, V.; TRIGONI, N. and YANG, X.: «Microsoft Indoor Localization Competition - IPSN 2017», 2017. `https://www.microsoft.com/en-us/research/event/microsoft-indoor-localization-competition-ipsn-2017`

[94] LYMBEROPOULOS, D.; LIU, J.; YANG, X.; NAGUIB, A.; ROWE, A.; TRIGONI, N. and MOAYERI, N.: «Microsoft Indoor Localization Competition (IPSN'15)», 2015. `ttps://www.microsoft.com/en-us/research/event/microsoft-indoor-localization-competition-ipsn-2015`

[95] LYMBEROPOULOS, D.; LIU, J.; ZHANG, Y.; DUTTA, P.; YANG, X. and ROWE, A.: «Microsoft Indoor Localization Competition—IPSN 2016», 2016. `https://www.microsoft.com/en-us/research/event/microsoft-indoor-localization-competition-ipsn-2016`

[96] MADIGAN, D.; EINAHRAWY, E.; MARTIN, R. P.; JU, WEN-HUA; KRISHNAN, P. and KRISHNAKUMAR, A. S.: «Bayesian indoor positioning systems». In: *Proceedings of the 24th Annual Joint Conference of the*

*IEEE Computer and Communications Societies (INFOCOM'05)*, pp. 1217–1227, 2005.

[97] MARQUES, NELSON; MENESES, FILIPE and MOREIRA, ADRIANO: «Combining similarity functions and majority rules for multi-building, multi-floor, WiFi positioning». In: *Proceedings of the 2012 International Conference on Indoor Positioning and Indoor Navigation (IPIN'12)*, pp. 1–9, 2012.

[98] MIKOLOV, TOMÁŠ; DEORAS, ANOOP; POVEY, DANIEL; BURGET, LUKÁŠ and ČERNOCK\'Y, JAN: «Strategies for training large scale neural network language models». In: *Workshop on Automatic Speech Recognition and Understanding (ASRU'11)*, pp. 196–201. Institute of Electrical and Electronics Engineers (IEEE), 2011.

[99] MOAYERI, N.; ERGIN, O.; LEMIC, F.; HANDZISKI, V. and WOLISZ, A.: «PerfLoc: An Extensive Data Repository for Development and a Web-Based Capability for Performance Evaluation of Smartphone Indoor Localization Apps». In: *Proceedings of the 27th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC'16)*, , 2016.

[100] MONTOLIU, R; SANSANO-SANSANO, E; GASCO, A; BELMONTE-FERNÁNDEZ, O and CABALLER, A:. «Senior Monitoring: A Real Case of Applying a WiFi Fingerprinting-based Indoor Positioning Method for People Monitoring».

[101] MONTOLIU, R.; TORRES-SOSPEDRA, J. and BELMONTE, O.: «Magnetic Field based Indoor Positioning Using the Bag of Words Paradigm». In: *Proceedings of the International Conference on Indoor Positioning and Indoor Navigation (IPIN'16)*, , 2016.

[102] MONTOLIU, RAÚL; SANSANO-SANSANO, E; BELMONTE-FERNÁNDEZ, OSCAR and TORRES-SOSPEDRA, JOAQUÍN: «A new methodology for long-term maintenance of wifi fingerprinting radio maps». In: *2018 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pp. 1–7. IEEE, 2018.

[103] MONTOLIU, RAUL; SANSANO-SANSANO, EMILIO; GASCÓ, ARTURO; BELMONTE-FERNÁNDEZ, OSCAR and CABALLER, ANTONIO: «Indoor Positioning for Monitoring Older Adults at Home: Wi-Fi and BLE Technologies in Real Scenarios». *Electronics*, 2020, **9(5)**, p. 728.

[104] MONTOLIU, RAUL; SANSANO-SANSANO, EMILIO; TORRES-SOSPEDRA, JOAQUÍN and BELMONTE-FERNÁNDEZ, ÓSCAR: «IndoorLoc Platform: A Web Tool to Support the Comparison of Indoor Positioning Systems». In: *Geographical and Fingerprinting Data to Create Systems for Indoor Positioning and Indoor/Outdoor Navigation*, pp. 225–247. Elsevier, 2019.

[105] MONTOLIU COLÁS, RAUL; SANSANO-SANSANO, EMILIO; TORRES-SOSPEDRA, JOAQUÍN and BELMONTE, ÓSCAR: «IndoorLoc Platform: A Public Repository for Comparing and Evaluating Indoor Positioning Systems». In: *Proceedings of the 2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN'17)*, , 2017.

[106] MOYA RUEDA, FERNANDO; GRZESZICK, RENÉ; FINK, GERNOT A; FELDHORST, SASCHA and TEN HOMPEL, MICHAEL: «Convolutional neural networks for human activity recognition using body-worn sensors». In: *Informatics*, volume 5, p. 26. Multidisciplinary Digital Publishing Institute, 2018.

[107] MOZER, MICHAEL C: «A focused backpropagation algorithm for temporal». *Backpropagation: Theory, architectures, and applications*, 1995, p. 137.

[108] MURABET AMINA, EL; ANOUAR, ABTOY; TOUHAFI, ABDELLAH and TAHIRI, ABDERAHIM: «Towards an SOA Architectural Model for AAL-Paas Design and Implimentation Challenges». *International Journal of Advanced Computer Science and Applications*, 2017, **8(7)**.

[109] MURAD, ABDULMAJID and PYUN, JAE-YOUNG: «Deep recurrent neural networks for human activity recognition». *Sensors*, 2017, **17(11)**, p. 2556.

[110] NAHRSTEDT, K. and VU, L.: «CRAWDAD Dataset uiuc/uim (v. 2012-01-24)», 2012.
`http://crawdad.org/uiuc/uim/20120124`

[111] NEYSHABUR, BEHNAM; BHOJANAPALLI, SRINADH; MCALLESTER, DAVID and SREBRO, NATI: «Exploring generalization in deep learning». In: *Advances in Neural Information Processing Systems*, pp. 5947–5956, 2017.

[112] ORDÓÑEZ, FRANCISCO JAVIER and ROGGEN, DANIEL: «Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition». *Sensors*, 2016, **16(1)**, p. 115.

[113] PAN, SINNO JIALIN and YANG, QIANG: «A survey on transfer learning». *IEEE Transactions on knowledge and data engineering*, 2010, **22(10)**, pp. 1345–1359.

[114] PASCANU, RAZVAN; MIKOLOV, TOMAS and BENGIO, YOSHUA: «On the difficulty of training recurrent neural networks». In: *Proceedings of the 30th International Conference on Machine Learning (ICML'13)*, pp. 1310–1318, 2013.

[115] PLAUT, DAVID C and OTHERS: «Experiments on Learning by Back Propagation.» *Technical Report*, Computer Science Department, Carnegie-Mellon University, 1986.

[116] PLÖTZ, THOMAS; HAMMERLA, NILS Y and OLIVIER, PATRICK: «Feature learning for activity recognition in ubiquitous computing». In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'11)*, volume 22, p. 1729, 2011.

[117] POPLETEEV, ANDREI: «AmbiLoc: A year-long dataset of FM, TV and GSM fingerprints for ambient indoor localization». In: *Proceedings of the 2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN'17)*, , 2017.

[118] POPLETEEV, ANDREI; OSMANI, VENET; MAYORA, OSCAR and MATIC, ALEKSANDAR: «Indoor localization using audio features of FM radio signals». In: *International and Interdisciplinary Conference on*

*Modeling and Using Context*, pp. 246–249. Springer, 2011. doi: https://doi.org/10.1007/978-3-642-24279-3_26.

[119] POTORTÌ, F.; BARSOCCHI, P.; GIROLAMI, M.; TORRES-SOSPEDRA, J. and MONTOLIU, R.: «Evaluating indoor localization solutions in large environments through competitive benchmarking: The EvAAL-ETRI competition». In: *Proceedings of the Sixth Conference on Indoor Positioning and Indoor Navigation (IPIN'15)*, , 2015.

[120] PREECE, STEPHEN J; GOULERMAS, JOHN YANNIS; KENNEY, LAURENCE P J and HOWARD, DAVID: «A comparison of feature extraction methods for the classification of dynamic activities from accelerometer data». *IEEE Transactions on Biomedical Engineering*, 2009, **56(3)**, pp. 871–879.

[121] QUAN, YIMING; LAU, LAWRENCE; JING, FAMING; NIE, QIAN; WEN, ALAN and CHO, SIU-YEUNG: «Analysis and machine-learning based detection of outlier measurements of ultra-wideband in an obstructed environment». In: *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*, pp. 997–1000. IEEE, 2017. doi: https://doi.org/10.1109/INDIN.2017.8104909.

[122] QUEIRÓS, ALEXANDRA; SILVA, ANABELA; ALVARELHÃO, JOAQUIM; ROCHA, NELSON PACHECO and TEIXEIRA, ANTÓNIO: «Usability, accessibility and ambient-assisted living: a systematic literature review». *Universal Access in the Information Society*, 2015, **14(1)**, pp. 57–66.

[123] RADU, VALENTIN; LANE, NICHOLAS D; BHATTACHARYA, SOURAV; MASCOLO, CECILIA; MARINA, MAHESH K and KAWSAR, FAHIM: «Towards multimodal deep learning for activity recognition on mobile devices». In: *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing (PUC'16)*, pp. 185–188. ACM, 2016.

[124] RASHIDI, PARISA and MIHAILIDIS, ALEX: «A survey on ambient-assisted living tools for older adults». *IEEE journal of biomedical and health informatics*, 2013, **17(3)**, pp. 579–590.

[125] RAVI, DANIELE; WONG, CHARENCE; LO, BENNY and YANG, G: «A deep learning approach to on-node sensor data analytics for mobile or

wearable devices». *IEEE Journal of Biomedical and Health Informatics*, 2016.

[126] RAVI, DANIELE; WONG, CHARENCE; LO, BENNY and YANG, GUANG-ZHONG: «Deep learning for human activity recognition: A resource efficient implementation on low-power devices». In: *Proceedings of IEEE 13th International Conference on Wearable and Implantable Body Sensor Networks (BSN'16)*, pp. 71–76. Institute of Electrical and Electronics Engineers (IEEE), 2016.

[127] REISS, ATTILA and STRICKER, DIDIER: «Creating and benchmarking a new dataset for physical activity monitoring». In: *Proceedings of the 5th International Conference on Pervasive Technologies Related to Assistive Environments (PETRA'12)*, p. 40. ACM, 2012.

[128] ——: «Introducing a new benchmarked dataset for activity monitoring». In: *Proceedings of the 16th International Symposium on Wearable Computers (ISWC)*, pp. 108–109. Institute of Electrical and Electronics Engineers (IEEE), 2012.

[129] RESEARCH and MARKETS: «Indoor Location Market by Component, Deployment Mode, Application, Vertical and Region - Global Forecast to 2022». *Research and markets*, 2017.
`https://www.researchandmarkets.com/reports/4416241/indoor-location-market-by-component-deployment`

[130] ROBINSON, A J and FALLSIDE, FRANK: *The utility driven dynamic error propagation network*. University of Cambridge Department of Engineering, 1987.

[131] ROGGEN, DANIEL; CALATRONI, ALBERTO; ROSSI, MIRCO; HOLLECZEK, THOMAS; FÖRSTER, KILIAN; TRÖSTER, GERHARD; LUKOWICZ, PAUL; BANNACH, DAVID; PIRKL, GERALD; FERSCHA, ALOIS and OTHERS: «Collecting complex activity datasets in highly rich networked sensor environments». In: *Proceedings of the seventh International Conference on Networked Sensing Systems (INSS'10)*, pp. 233–240. Institute of Electrical and Electronics Engineers (IEEE), 2010.

[132] RONAO, CHARISSA ANN and CHO, SUNG-BAE: «Human activity recognition with smartphone sensors using deep learning neural networks». *Expert Systems with Applications*, 2016, **59**, pp. 235–244.

[133] ROOS, TEEMU; MYLLYMÄKI, PETRI; TIRRI, HENRY; MISIKANGAS, PAULI and SIEVÄNEN, JUHA: «A Probabilistic Approach to WLAN User Location Estimation». *International Journal of Wireless Information Networks*, 2002, **9(3)**, pp. 155–164. doi: https://doi.org/10.1023/A:1016003126882.

[134] ROUSSEEUW, PETER J: «Silhouettes: a graphical aid to the interpretation and validation of cluster analysis». *Journal of computational and applied mathematics*, 1987, **20**, pp. 53–65.

[135] RUMELHART, DAVID E; HINTON, GEOFFREY E and WILLIAMS, RONALD J: «Learning representations by back-propagating errors». *nature*, 1986, **323(6088)**, p. 533.

[136] SANSANO, E.; MONTOLIU, R. and TORRES-SOSPEDRA, J.: «A Novel Methodology to Estimate a Measurement of the Inherent Difficulty of an Indoor Localization Radio Map». In: *Proceedings of the 2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN'17)*, , 2017.

[137] SANSANO, EMILIO: *ipft: Indoor Positioning Fingerprinting Toolset*, 2017.
`https://cran.r-project.org/web/packages/ipft/index.html`

[138] SANSANO SANSANO, E.; BELMONTE-FERNÁNDEZ, O.; MONTOLIU, R.; GASCÓ-COMPTE, A.; CABALLER MIEDES, A. and BAYARRI ITURRALDE, P.: «Improving Positioning Accuracy in Ambient Assisted Living Environments. A Multi-Sensor Approach». In: *2019 15th International Conference on Intelligent Environments (IE)*, pp. 22–29, 2019.

[139] SANSANO-SANSANO, E; MONTOLIU, RAÚL and TORRES-SOSPEDRA, JOAQUÍN: «A novel methodology to estimate a measurement of the

inherent difficulty of an indoor localization radio map». In: *2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pp. 1–8. IEEE, 2017.

[140] SANSANO-SANSANO, EMILIO; BELMONTE-FERNÁNDEZ, ÓSCAR; MONTOLIU, RAÚL; GASCÓ-COMPTE, ARTURO and CABALLER-MIEDES, ANTONIO: «Multimodal Sensor Data Integration for Indoor Positioning in Ambient-Assisted Living Environments». *Mobile Information Systems*, 2020, **2020**.

[141] SANSANO-SANSANO, EMILIO; MONTOLIU, RAÚL and BELMONTE FERNÁNDEZ, ÓSCAR: «A study of deep neural networks for human activity recognition». *Computational Intelligence*, 2020.

[142] SANSANO-SANSANO, EMILIO; MONTOLIU COLÁS, RAUL; BELMONTE, ÓSCAR and TORRES-SOSPEDRA, JOAQUÍN:. «Indoor Positioning and Fingerprinting: The R Package ipft».

[143] SCHINDHELM, CORINA K and MACWILLIAMS, ASA: «Overview of indoor positioning technologies for context aware AAL applications». In: *Ambient Assisted Living*, pp. 273–291. Springer, 2011.

[144] SEYBOLD, J.S.: *Introduction to RF Propagation*. Wiley, 2005.

[145] SHANNO, D. F.: «Conditioning of Quasi-Newton Methods for Function Minimization». *Mathematics of Computation*, 1970, **24(111)**, pp. 647–656. doi: https://doi.org/10.1090/S0025-5718-1970-0274029-X.

[146] SHOAIB, MUHAMMAD; BOSCH, STEPHAN; INCEL, OZLEM DURMAZ; SCHOLTEN, HANS and HAVINGA, PAUL J M: «Fusion of smartphone motion sensors for physical activity recognition». *Sensors*, 2014, **14(6)**, pp. 10146–10176.

[147] SHOAIB, MUHAMMAD; SCHOLTEN, HANS and HAVINGA, PAUL J M: «Towards physical activity recognition using smartphone sensors». In: *Proceedings of the IEEE 10th international conference on Ubiquitous intelligence and computing and 10th international conference on autonomic and trusted computing (UIC/ATC'13)*, pp. 80–87. Institute of Electrical and Electronics Engineers (IEEE), 2013.

[148] SHRESTHA, SHWETA; TALVITIE, JUKKA and LOHAN, ELENA SIMONA: «On the fingerprints dynamics in WLAN indoor localization». In: *Proceedings of the 13th International Conference on ITS Telecommunications (ITST'13)*, pp. 122–126, 2013. doi: https://doi.org/10.1109/ITST.2013.6685532.

[149] SHU, YUANCHAO; BO, CHENG; SHEN, GUOBIN; ZHAO, CHUNSHUI; LI, LIQUN and ZHAO, FENG: «Magicol: Indoor localization using pervasive magnetic field and opportunistic WiFi sensing». *IEEE Journal on Selected Areas in Communications*, 2015, **33(7)**, pp. 1443–1457.

[150] SRIVASTAVA, NITISH; HINTON, GEOFFREY; KRIZHEVSKY, ALEX; SUTSKEVER, ILYA and SALAKHUTDINOV, RUSLAN: «Dropout: A Simple Way to Prevent Neural Networks from Overfitting». *Journal of Machine Learning Research*, 2014, **15**, pp. 1929–1958.
`http://jmlr.org/papers/v15/srivastava14a.html`

[151] STISEN, ALLAN; BLUNCK, HENRIK; BHATTACHARYA, SOURAV; PRENTOW, THOR SIIGER; KJÆRGAARD, MIKKEL BAUN; DEY, ANIND; SONNE, TOBIAS and JENSEN, MADS MØLLER: «Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition». In: *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems (SenSys'15)*, pp. 127–140. ACM, 2015.

[152] SZTYLER, TIMO and STUCKENSCHMIDT, HEINER: «On-body localization of wearable devices: An investigation of position-aware activity recognition». In: *Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PerCom'16)*, pp. 1–9. Institute of Electrical and Electronics Engineers (IEEE), 2016.

[153] TALVITIE, J.; LOHAN, E.S. and RENFORS, M.: «The effect of coverage gaps and measurement inaccuracies in fingerprinting based indoor localization.» In: *Proceedings of International Conference on Localization and GNSS 2014 (ICL-GNSS'14)*, , 2014.

[154] TORRES, JOAQUÍN; BELMONTE, ÓSCAR; MONTOLIU, RAÚL; TRILLES, SERGIO and CALIA, ANDREA: «How feasible is WiFi fingerprint-based

indoor positioning for in-home monitoring?» In: *Intelligent Environments (IE), 2016 12th International Conference on*, pp. 68–75. IEEE, 2016.

[155] TORRES-SOSPEDRA, J.; RAMBLA, D.; MONTOLIU, R.; BELMONTE, O. and HUERTA, J.: «UJIIndoorLoc-Mag: A New Database for Magnetic Field-Based Localization Problems». In: *Proceedings of the Sixth Conference on Indoor Positioning and Indoor Navigation (IPIN'15)*, , 2015.

[156] TORRES-SOSPEDRA, JOAQUIN; MONTOLIU, RAUL; MARTINEZ-USO, ADOLFO; AVARIENTO, JOAN P.; ARNAU, TOMAS J.; BENEDITO-BORDONAU, MAURI and HUERTA, JOAQUIN: «UJIIndoorLoc: A new multi-building and multi-floor database for WLAN fingerprint-based indoor localization problems». In: *Proceedings of the 5th International Conference on Indoor Positioning and Indoor Navigation (IPIN'14)*, pp. 261–270, 2015. doi: https://doi.org/10.1109/IPIN.2014.7275492.

[157] TORRES-SOSPEDRA, JOAQUÍN; MONTOLIU, RAÚL; TRILLES, SERGIO; BELMONTE, ÓSCAR and HUERTA, JOAQUÍN: «Comprehensive analysis of distance and similarity measures for Wi-Fi fingerprinting indoor positioning systems». *Expert Systems with Applications*, 2015, **42(23)**, pp. 9263–9278.

[158] TORRES-SOSPEDRA, JOAQUÍN; MONTOLIU, RAÚL; USÓ, ADOLFO MARTÍNEZ; AVARIENTO, JOAN P.; ARNAU, TOMAS J.; BENEDITO-BORDONAU, MAURI and HUERTA, JOAQUÍN: «UJIIndoorLoc: A new multi-building and multi-floor database for WLAN fingerprint-based indoor localization problems». In: *Proceedings of the 2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN'14)*, pp. 261–270, 2014.

[159] TORRES-SOSPEDRA, JOAQUÍN; JIMÉNEZ, ANTONIO R.; KNAUTH, STEFAN; MOREIRA, ADRIANO; BEER, YAIR; FETZER, TONI; TA, VIET-CUONG; MONTOLIU, RAUL; SECO, FERNANDO; MENDOZA-SILVA, GERMÁN M.; BELMONTE, OSCAR; KOUKOFIKIS, ATHANASIOS; NICOLAU, MARIA JOÃO; COSTA, ANTÓNIO; MENESES, FILIPE; EBNER, FRANK;

DEINZER, FRANK; VAUFREYDAZ, DOMINIQUE; DAO, TRUNG-KIEN and CASTELLI, ERIC: «The Smartphone-Based Offline Indoor Location Competition at IPIN 2016: Analysis and Future Work». *Sensors*, 2017, **17(3)**.

[160] TORRES-SOSPEDRA, JOAQUÍN; MOREIRA, ADRIANO; KNAUTH, STEFAN; BERKVENS, RAFAEL; MONTOLIU-COLÁS, RAÚL; ÓSCAR BELMONTE-FERNÁNDEZ; TRILLES, SERGIO; NICOLAU, MARIA JOAO; MENESES, FILIPE; COSTA, ANTONIO; KOUKOFIKIS, ATHANASIOUS; WEYN, MARTEEN and PEREMANS, HERBERT: «Realistic Evaluation of Indoor Positioning Systems Based on Wi-Fi Fingerprinting: The 2015 EvAAL-ETRI Competition». *Journal of ambient intelligence and smart environments*, 2017, **9**, p. 263–279.

[161] VANDERMEULEN, DRIES; VERCAUTEREN, CHARLES and WEYN, MAARTEN: «Indoor localization using a magnetic flux density map of a building». In: *The Third International Conference on Ambient Computing, Applications, Services and Technologies*, pp. 42–49, 2013.

[162] WANG, YAPENG; YANG, XU; ZHAO, YUTIAN; LIU, YUE and CUTHBERT, LAURIE: «Bluetooth positioning using RSSI and triangulation methods». In: *Proceedings of the 10th IEEE Consumer Communications and Networking Conference, (CCNC'13)*, pp. 837–842, 2013. doi: https://doi.org/10.1109/CCNC.2013.6488558.

[163] WANG, ZHIHUA; YANG, ZHAOCHU and DONG, TAO: «A review of wearable technologies for elderly care that can accurately track indoor position, recognize physical activities and monitor vital signs in real time». *Sensors*, 2017, **17(2)**, p. 341.

[164] WERBOS, PAUL J: «Generalization of backpropagation with application to a recurrent gas market model». *Neural networks*, 1988, **1(4)**, pp. 339–356.

[165] WERNER, WEBER; RABAEY, JAN and AARTS, EMILE H.L. (Eds.): *Ambient intelligence*. Springer-Verlag Berlin Heidelberg, 2005. ISBN 978-3-540-27139-0. doi: https://doi.org/10.1007/b138670.

[166] WICKHAM, HADLEY: «ggplot2». *Wiley Interdisciplinary Reviews: Computational Statistics*, 2011, **3(2)**, pp. 180–185. doi: https://doi.org/10.1002/wics.147.

[167] WILCOXON, FRANK: «Individual comparisons by ranking methods». *Biometrics bulletin*, 1945, **1(6)**, pp. 80–83.

[168] WU, CHENSHU; YANG, ZHENG; LIU, YUNHAO and XI, WEI: «WILL: Wireless Indoor Localization without Site Survey». *IEEE Transactions on Parallel and Distributed Systems*, 2013, **24(4)**, pp. 839–848.

[169] WU, WANMIN; DASGUPTA, SANJOY; RAMIREZ, ERNESTO E; PETERSON, CARLYN and NORMAN, GREGORY J: «Classification accuracies of physical activities using smartphone motion sensors». *Journal of medical Internet research*, 2012, **14(5)**.

[170] XIAO, JIANG; ZHOU, ZIMU; YI, YOUWEN and NI, LIONEL M.: «A Survey on Wireless Indoor Localization from the Device Perspective». *ACM Computing Surveys*, 2016, **49(2)**, pp. 1–31. doi: https://doi.org/10.1145/2933232.

[171] XIE, HONGWEI; GU, TAO; TAO, XIANPING; YE, HAIBO and LV, JIAN: «MaLoc: A practical magnetic fingerprinting approach to indoor localization using smartphones». In: *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 243–253, 2014.

[172] YANG, CHUN; NGUYEN, THAO and BLASCH, ERIK: «Mobile positioning via fusion of mixed signals of opportunity». *IEEE Aerospace and Electronic Systems Magazine*, 2014, **29(4)**, pp. 34–46. doi: https://doi.org/10.1109/MAES.2013.130105.

[173] YANG, JIANBO; NGUYEN, MINH NHUT; SAN, PHYO PHYO; LI, XIAOLI and KRISHNASWAMY, SHONALI: «Deep Convolutional Neural Networks on Multichannel Time Series for Human Activity Recognition.» In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'15)*, volume 15, pp. 3995–4001, 2015.

[174] YOUSSEF, MOUSTAFA and AGRAWALA, ASHOK: «The Horus WLAN Location Determination System». In: *Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services (MobiSys'05)*, pp. 205–218, 2005.

[175] ZEBIN, TAHMINA; SCULLY, PATRICIA J and OZANYAN, KRIKOR B: «Human activity recognition with inertial sensors using a deep learning approach». In: *Proceedings of the IEEE Sensors conference (SENSORS'16)*, pp. 1–3. Institute of Electrical and Electronics Engineers (IEEE), 2016.

[176] ZENG, MING; NGUYEN, LE T; YU, BO; MENGSHOEL, OLE J; ZHU, JIANG; WU, PANG and ZHANG, JOY: «Convolutional neural networks for human activity recognition using mobile sensors». In: *Proceedings of the 6th International Conference on Mobile Computing, Applications and Services (MobiCASE'14)*, pp. 197–205. Institute of Electrical and Electronics Engineers (IEEE), 2014.

[177] ZHANG, LICHENG; WU, XIHONG and LUO, DINGSHENG: «Recognizing human activities from raw accelerometer data using deep neural networks». In: *Proceedings of the IEEE 14th International Conference on Machine Learning and Applications (ICMLA'15)*, pp. 865–870. Institute of Electrical and Electronics Engineers (IEEE), 2015.

[178] ZHANG, MI and SAWCHUK, ALEXANDER A: «USC-HAD: a daily activity dataset for ubiquitous activity recognition using wearable sensors». In: *Proceedings of the ACM Conference on Ubiquitous Computing (UbiComp'12)*, pp. 1036–1043. ACM, 2012.

[179] ZHENG, ZENGWEI; CHEN, YUANYI; HE, TAO; LI, FEI and CHEN, DAN: «Weight-RSS: A calibration-free and robust method for WLAN-Based indoor positioning». *International Journal of Distributed Sensor Networks*, 2015, **11(4)**, p. 573582.

[180] ZHU, NAN; ZHAO, HONGBO; FENG, WENQUAN and WANG, ZULIN: «A novel particle filter approach for indoor positioning by fusing WiFi and inertial sensors». *Chinese Journal of Aeronautics*, 2015, **28(6)**, pp. 1725–1734.