Exploring Impact of Location Granularity for Improving Indoor Localization Performance through Machine Learning Techniques

Thesis Submitted

By

Manjarini Mallik

Doctor of Philosophy (Engineering)

Department of Computer Science and Engineering

Faculty Council of Engineering & Technology

Jadavpur University, Kolkata, India

Index No - 121/22/E

1. **Title of the thesis:** Exploring Impact of Location Granularity for Improving Indoor Localization Performance Through Machine Learning Techniques

2. **Name, Designation and Institution of the Supervisor:**

Prof. Chandreyee Chowdhury
Department of Computer Science and Engineering Jadavpur University Kolkata,
West Bengal-700032, India

3. **List of Publications:**

(a) Journal publications:

1. **Mallik, M.**, Panja, A.K. and Chowdhury, C., 2023. Paving the way with machine learning for seamless indoor–outdoor positioning: A survey. *Information Fusion*, *94*, pp.126-151. Elsevier. https://doi.org/10.1016/j.inffus.2023.01.023 (Impact Factor: 15.5)

2. **Mallik, M.**, Das, S. and Chowdhury, C., 2023. Rank based iterative clustering (RBIC) for indoor localization. *Engineering Applications of Artificial Intelligence*, *121*, p.106061. Elsevier. https://doi.org/10.1016/j.engappai.2023.106061 (Impact Factor: 8.0)

3. **Mallik, M**. and Chowdhury, C., 2023. Characteristic analysis of fingerprint datasets from a pragmatic view of indoor localization using machine learning approaches. *The Journal of Supercomputing*,*79*(16), pp.18507-18546. Springer. https://doi.org/10.1007/s11227-023-05386-x (Impact Factor: 2.7)

4. **Mallik, M.**, Chakraborty, S., Sasidhar, K. and Chowdhury, C., 2025. VL-GAN: A Generative Classification Approach for Fingerprint-based Indoor Localization. *Expert Systems with Applications*, p.128400. Elsevier. https://doi.org/10.1016/j.eswa.2025.128400 (Impact Factor: 7.5)

5. Chakraborty, P., **Mallik, M.**, Kundu, A. and Chowdhury, C., 2025. Design of a Unified Indoor Localization System Integrating IoT Devices and Smartphones. *CCF Transactions on Pervasive Computing and Interaction*. Springer. https://doi.org/10.1007/s42486-025-00204-0 (Impact Factor: 2.0)

(b) Conference publications:

1. **Mallik, M.**, Chowdhury, C., 2023. IndoorGML Modeling for WiFi-Based Indoor Positioning and Navigation. In: Chakravarthy, V., Bhateja, V., Flores Fuentes, W., Anguera, J., Vasavi, K.P. (eds) Advances in Signal Processing, Embedded Systems and IoT. Lecture Notes in Electrical Engineering, vol 992. Springer, Singapore. https://doi.org/10.1007/978-981-19-8865-3_45

2. **Mallik, M.,** Chowdhury, C. (2026). DAMLoc: Data Augmentation for Minority Location Classes Applying CGAN for Indoor Localization. In: Bhattacharyya, S., Banerjee, J.S., Platos, J., De, I. (eds) Proceedings of the International Conference on Web 6.0 and Industry 6.0. WIN 2025. Lecture Notes in Networks and Systems, vol 1508. Springer, Singapore. https://doi.org/10.1007/978-981-96-8998-9_8

4. **Communicated Works:**

1. **Mallik, M**. and Chowdhury, C., 2025. GO-kDN: Granularity Optimization through k-Disagreeing Neighbor Concept for Machine Learning based Indoor Localization. *Pervasive and Mobile Computing,* Elsevier.

2. **Mallik, M**., Brahma, D., Khaskel, R. and Chowdhury, C., 2025. A resource-friendly indoor localization method using spatial feature analysis of WiFi fingerprints. *Measurement*, Elsevier [Under revision]

3. **Mallik, M.**, Manna, K., Pati, K., Thakur, T. and Chowdhury, C, 2025. Design of a Smartphone-based Real-time Seamless Localization System using Machine Learning. *Journal of The Institution of Engineers (India): Series B*, Springer.

5. **Other publications:**

1. Ganguly, R., **Mallik, M**. and Chowdhury, C., 2024. Design of a knowledge distillation network for wifi-based indoor localization. *Multimedia Tools and Applications*, pp.1-17. Springer. https://doi.org/10.1007/s11042-024-20212-z (Impact Factor: 3.0)

6. Software Copyrights:

1. **M. Mallik**, A.K. Panja, C. Chowdhury, JU-WIFI SCANNER: A WIFI RSSI FINGERPRINT DATA COLLECTION APPLICATION SOFTWARE FOR INDOOR LOCALIZATION, @copyright 2024 from Intellectual Property India, SW-19314/2024.

7. List of Patents:

1. Title: User Source Localization Framework for Indoor Environment
   Patent/Application Number: 202531115724
   Date: 2025/11/23
   Patent Office: India (Indian Patent Office)
   Inventors: A.K. Panja, C. Chowdhury, **M. Mallik**, P. Chakraborty, A. Kundu
   Status: Published

Statement of Originality

I, **Ms. Manjarini Mallik** registered on **17th May 2022** do hereby declare that this thesis entitled **"Exploring Impact of Location Granularity for Improving Indoor Localization Performance Through Machine Learning Techniques"** contains literature survey and original research work done by the undersigned candidate as part of Doctoral studies.

All information in this thesis have been obtained and presented in accordance with existing academic rules and ethical conduct. I declare that, as required by these rules and conduct, I have fully cited and referred all materials and results that are not original to this work.

I also declare that I have checked this thesis as per the **"Policy on Anti Plagiarism, Jadavpur University, 2025"**, and the level of similarity as checked by iThenticate software is **3%** .

Signature of Candidate: Manjarini Mallik

Date : 21.08.2025

Certified by Supervisor: Chandreyee Chowdhury 21/08/25

(Signature with date, seal)

Professor
Computer Sc. & Engg. Department
Jadavpur University
Kolkata-700032

(Dr. Chandreyee Chowdhury,

Professor,

Department of Computer Science and Engineering

Jadavpur University.)

Certificate from the Supervisor

This is to certify that the thesis entitled ***"Exploring Impact of Location Granularity for Improving Indoor Localization Performance Through Machine Learning Techniques"*** *submitted by* ***Ms. Manjarini Mallik****, who got her name registered on* ***17th May 2022****, for the award of Ph.D.(Engineering) degree of Jadavpur University, is absolutely based upon her own work under the supervision of* ***Dr. Chandreyee Chowdhury*** *and that neither her thesis nor any part of the thesis has been submitted for any degree/diploma or any other academic award anywhere before.*

Chandreyee Chowdhury

21/08/25

Dr. Chandreyee Chowdhury,
Professor,
Department of Computer Science and Engineering
Jadavpur University.
(Supervisor)

Professor
Computer Sc. & Engg. Department
Jadavpur University
Kolkata-700032

Acknowledgment

First and foremost, I extend my deepest gratitude and heartfelt thanks to my respected supervisor, Dr. Chandreyee Chowdhury, Professor at Jadavpur University. Her constant support, valuable research insight, dedicated guidance, continuous encouragement and active involvement during my entire PhD journey have been instrumental in the successful completion of my Doctor of Philosophy. Her enthusiasm, patience, acceptance and profound perspectives on research made this journey toward my PhD not only possible but also joyful. Her genuine support motivated me to dream big, and helped me to remain strong through all the ups and downs of my PhD journey.

I am sincerely grateful to the other members of my thesis committee for their invaluable and constructive feedback, which has substantially enhanced the quality of this thesis. Furthermore,I want to express my gratitude to several remarkable individuals whose contributions significantly influenced this research. Firstly, I would like to acknowledge Dr. Priya Roy and Dr. Ayan Kumar Panja for sharing their deep research insights in the field of indoor localization. I extend my thanks to Mr. Soumyajit Chakraborty, Ms. Sanchita Das, Mr. Ritabroto Ganguly and Mr. Prodipta Chakraborty for their collaborative efforts. I would like to express my sincere thanks to my fellow researchers and labmates, Dr. Asif Iqbal Middya, Mr. Sounak Banerjee, Mr. Subhayan Bhattacharya, Dr. Priyanka Saha, Ms. Bidisha Banerjee, Mr. Rohit Ghosal, Mr. Soumyadeep Sur, Ms. Arundhati Bhowal for providing a healthy, research-oriented and friendly lab environment.

I am deeply grateful to my parents, who never imposed a specific path on me but instead allowed me to explore my own way in my career at my own pace, and to my two elder sisters, who have always been the strongest pillars of my life, standing by me through every difficult situation.

Lastly, I extend my thanks to the Department of Computer Science and Engineering at Jadavpur University for providing me with this exceptional opportunity and for their exceptional support and guidance throughout my research journey.

Dedicated

To my beloved parents and sisters
To all the researchers who have contributed in this field

97% accurate localization is achieved on average. To address the challenge of signal fluctuations in case of fine-grained uniform granularity, a signal-to-image encoding algorithm is proposed to improve the localization performance using spatial analysis. When the encoded images are classified using proposed lightweight CNN model, around 99% accuracy is achieved. A new algorithm, GO-kDN, is proposed to update the granularity level of any fingerprint dataset based on instance hardness analysis to increase localization accuracy. For experimentation, a new dataset is collected from two metro stations, and it is made publicly available for further research. Applying the proposed approach, localization performance improved by 15% for at-grade-level and 35% for underground-level metro stations. To generate synthetic data and map it with proper location based on qualitative analysis, a Vectored labeled Generative Adversarial Network (VL-GAN) model is proposed. Up to 10% improvement in localization accuracy is observed after augmenting synthetic data generated by VL-GAN.

Acronym

ILS: Indoor Localization System
GNSS: Global Navigation Satellite System
GPS: Global Positioning System
BLE: Bluetooth low-energy
PDR: Pedestrian dead reckoning
CNN: Convolutional neural network
AR: Augmented Reality
ML: Machine Learning
DL: Deep Learning
DNN: Deep Neural Network
RSS: Received Signal Strength
UWB: Ultra-wideband
TDOA: Time Difference of Arrival
RSSI: Received Signal Strength Indicator
AP: Access Point
CSI: Channel State Information
kNN: k-nearest neighbor
SVM: Support vector machine
DT: Decision tree
RF: Random forest
ANN: Artificial neural network
RBIC: Rank based iterative clustering
GO-kDN: Granularity Optimization through k-Disagreeing Neighbor
GAN: Generative adversarial network
CGAN: Conditional generative adversarial network
ELM: Extreme Learning Machine
RFID: Radio-frequency identification
IMU: Inertial measurement unit
GMM: Gaussian Mixture Model

IERF: Image Encoding using RSSI Fusion

CDF: Cumulative Distribution Function

t-SNE: t-distributed Stochastic Neighbor Embedding

LTV: Label to vector

SRFG: Set wise fingerprint generation

IoT: Internet of Things

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the current era, technology and innovations have reshaped the living styles significantly, especially in metro cities and semi-urban areas. Smart navigation and wayfinding[1] is one of the key components of smart cities, helping residents and visitors move around efficiently and safely. To achieve real-time seamless navigation from outdoor to indoor and vice versa, the system must integrate efficient positioning or localization methods suitable for both indoor and outdoor environments. After decades of research, satellite signal-based ready-to-use applications for outdoor navigation and positioning has already come into widespread use like Google Maps[2], Komoot[3], Citymapper[4] etc. However, such worldwide standard applications are not available indoor or semi-indoor environment. Indoor areas are commonly enclosed by high-density building materials like concrete or brick walls, stone floors which the satellite signals cannot penetrate. Also, underground spaces like basements, underground parking garages, and subway stations do not have a direct line of sight to any Global Navigation Satellite System (GNSS) like Global Positioning System (GPS). Thus, estimating a user's location in indoor areas entirely depends on indoor-available sensors such as WiFi, Bluetooth, and magnetometers. Each of these sensors come with its respective real-life challenges, which becomes major or minor matter of concerns for different venues and adapted techniques. Currently, different group of researchers are working to investigate and address different domains of challenges in this regard. These research outcomes help to develop application system suitable for real life indoor positioning(coordinate-level location prediction) or localization(grid-level location

prediction). Some of the major application domains of indoor localization are presented as follows.

• **Navigation and wayfinding assistance:** User's continuous location prediction in a small interval of time is the building block of an indoor navigation and wayfinding system. The process mostly include radio frequency or inertial sensors and computer vision techniques [5]. The AlmaWhere[6] system was designed to assist disabled students to navigate within the campus area of the University of Bologna. It is based on Bluetooth low-energy (BLE) beacon technology, and for location estimation, a combination of fingerprinting and pedestrian dead reckoning (PDR) was applied. A vision-based indoor wayfinding system was proposed in [7] which use convolutional neural network(CNN) model for landmark image recognition. Although this helps visually impaired users to inform about some specified landmark areas (water closet, exit, disabled exit, confidence zone), the overall system is resource hungry as it is entirely vision based. Augmented reality (AR) can be a better approach to build a comparatively lightweight system, as proposed in the sensor fusion based indoor navigation system[8]. The work uses fusion of WiFi and magnetic field data along with a game development engine to create a visually appealing AR interface.

• **Emergency response:** An emergency situation may arise in many forms, and the victim needs to be visited at the earliest in each case. A visual simultaneous localization and mapping (vSLAM) based indoor localization system was proposed in [9] for victims of a sudden cardiac arrest. The goal was to locate the patient and provide an automated external defibrillator (AED) for initial treatment. Another emergency response approach was proposed in [10] to locate the victims inside a damaged building and group them around some predefined anchor points. The distances between the anchor points and rescue team was mapped to a weighted graph, and shortest paths were computed using graph theoretic algorithms. Augmented reality (AR) is playing a crucial role for developing emergency evacuation indoor solutions [11] [12]. Researchers in [11] proposed a machine learning(ML) based emergency evacuation system for indoors. Users locations were predicted using a deep neural network (DNN) model, trained on received signal strength (RSS) values from beacon nodes. The evacuation paths were computed using Q -learning method, integrated

with building structural knowledge and disaster context information. Another AR-based emergency evacuation approach was proposed in [12], where the system recommends the shortest path based on accelerometer data.

• **Asset tracking:** Asset tracking indoors is the process of monitoring the location, status, and movement of assets within an indoor facility. This is especially useful in environments like warehouses, hospitals, offices, and retail stores. In the work [13] a sensor-fusion based solution was proposed integrating WiFi, Ultra wideband (UWB) and Bluetooth low energy (BLE), specially to track assets in a dense industrial environment. Outliers were handled carefully using machine learning analysis. TinyML was considered in [14] to include machine learning for asset tracking solely based on BLE information, due to its usefulness in resource-constrained embedded devices. Time Difference of Arrival (TDOA) and the triangulation approaches are comparatively older, but still in use where UWB sensor is available, such as, in industrial areas for asset tracking purpose [15].

So, indoor localization is found to be significant across several application domains. Next section onwards we present the mechanisms of indoor localization mainly from the machine learning perspective and hence, build the motivation of this thesis work.

1.1 Fingerprinting Technique for Indoor Localization using Smartphone Sensors

In fingerprint based indoor localization method, precise areas on the floor are identified by some unique pattern of readings or their signal strengths or sensor values, called as 'fingerprint'. Three sensors that are commonly available in smartphones form the basis of this fingerprint-based indoor localization research– WiFi, Bluetooth, and magnetometer. Each of these sensors has their own strengths, limitations and applications, discussed as follows.

• **WiFi based fingerprinting:** Nowadays, urban and semi-urban areas are mostly covered with WiFi facility, and sometimes it is available in rural areas also. For this

Timestamp	Feature Set				Class
	AP1	AP2	AP3	AP4	Location ID
1669376067	-40	-55	-78	-58	L1
1669376068	-42	-56	-82	-59	L1
1669376069	-39	-57	-82	-57	L1
1669376070	-45	-52	-80	-59	L1

Figure 1.1: WiFi fingerprint structure

ready-to-use infrastructure, researchers have appreciated and widely used this method for various indoor localization purposes [16]. Received Signal Strength Indicator (RSSI) is a measurement used in wireless communication to indicate the strength of a received signal. Theoretically, 0 dBm is considered the strongest signal strength, though in practice the range varies from -25 dBm to -100 dBm in general. Thus, a particular location receives strong RSSI values from nearby access points (APs) and weak RSSI values from far-apart APs, and a vector of RSSI values are obtained as presented in Figure 1.1. Each AP acts as a feature, and it must stay fixed at its position. Multiple RSSI vectors collected at each location form the fingerprint vector of that location. Researchers have published over 50 open-access WiFi fingerprinting datasets [17] in different formats and dimensions, collected under various dynamic conditions. This dynamic patterns and non-linear mapping of data are greatly dealt by various machine learning (ML) approaches [18], making the ML-based WiFi fingerprinting an emerging research domain.

• **Bluetooth based fingerprinting:** Bluetooth Low Energy (BLE) signal is a popular option in recent indoor localization systems due to its low power consumption [19]. In Bluetooth fingerprinting, RSSI values are collected from fixed beacons placed within the indoor space, and accordingly a fingerprint pattern for each location is obtained. The RSSI range is similar to the WiFi range, -25 dBm to -100 dBm approximately. However, the BLE signal traverse a comparatively shorter range, typically around 30 meters, which demands deployment of a large number of beacons for localization in a large indoor space.

Also, a ready-to-use BLE infrastructure is comparatively difficult to find, as most of the public indoor spaces prefer WiFi for communication. Compared to WiFi, these practical concerns have delayed the growth of the Bluetooth-based indoor localization research field [20].

• **Magnetometer based fingerprinting:** Fingerprints can be built using magnetic field strengths data as location identifiers within an indoor space. Primarily a magnetic feature vector contains three components - m_x, m_y and m_z representing the magnetic field strengths for x, y and z axis respectively [21]. Then, more features can be extracted from these three basic features using statistical computations [22] to enhance the localization performance. The Earth's geomagnetic field strengths vary significantly across different locations, and this difference becomes more prominent in presence of static ferromagnetic materials, for which we get different location specific fingerprint patterns. However, this strength becomes weakness when an indoor space has sufficient amount of moving ferromagnetic materials, for example electric motor and generator rooms, conveyor belts in manufacturing plants and MRI rooms in hospitals.

• **Hybrid fingerprinting:** Hybrid fingerprinting approach can be a good option when some areas in an indoor space are not covered by a single signal source. For example, in the work [23] researchers proposed a hierarchical topological fingerprint strategy, integrating WiFi and BLE fingerprints. They deployed BLE APs in those places which are not covered by existing WiFi APs. Another hybrid approach was proposed in [24] combining Wi-Fi and BLE fingerprints, to investigate the sensitivity of RSSI patterns with different layouts of the indoor environments. Magnetic fingerprints are rarely fused with Wi-Fi or BLE because their features and location identification process differ significantly. This makes it difficult to divide the floor into uniformly sized grids such that each grid holds a strong fingerprint pattern for both magnetic and WiFi/BLE data. In the work [26], to fuse the WiFi and magnetic field data, their sampling rates were fixed to 0.25 Hz, which is a normal sampling rate for WiFi, but for magnetometer data, the normal sampling rate is around 25 Hz. However, it was a coarse-grained localization approach for region prediction; to

build a fine-grained localization approach with data fusion, challenges are required to be investigated with more depth.

1.2 Machine Learning framework for Indoor Localization

Different indoor spaces have varying layouts and infrastructures, and various kinds of obstacles differing in terms of materials, sizes and shapes. All of these static things affect the formation of fingerprints, along with some dynamic issues as well, like the time of experimentation, crowd density of the area, configurations of receiver devices and many others. Thus, the fingerprint patterns are greatly dynamic in nature. Again, for WiFi and BLE based approaches, APs, that is, fixed signal sources act as features in the datasets, which hold a direct relationship among them, resulting in non-linear feature relationship. For instance, a smartphone at one location cannot record strong signal strength from two APs installed at two different rooms. An indoor localization system must learn all these complex non-linear relations to be applicable in the real world. Machine learning algorithms can analyze and uncover this kinds of complex patterns, that are location specific and hence, could be utilized to distinguish among the location classes. Moreover, the ambient conditions also affect the WiFi signals that calls for an adaptive localization solution. Hence, machine learning provides a better alternative than other positioning approaches such as, angle of arrival or triangulation[28]. The basic pipeline of an ML-based indoor localization framework is presented in Figure 1.2, and each of the steps are detailed as follows.

1.2.1 Raw Data Collection

Data are collected in raw form, from fixed sources within a specified region considered for the localization. For WiFi and BLE-based location, the RSSI values received from different APs are recorded using a smartphone application. This RSSI vector forms the training data. The corresponding location label constitutes the ground truth. Usually two types of data acquisition processes are used – dedicated user-based and crowd-sourced. In dedicated user-based data collection, a group of volunteers (termed as 'subjects') collect data using smart devices. In crowd-sourced data collection, the dataset is built using a

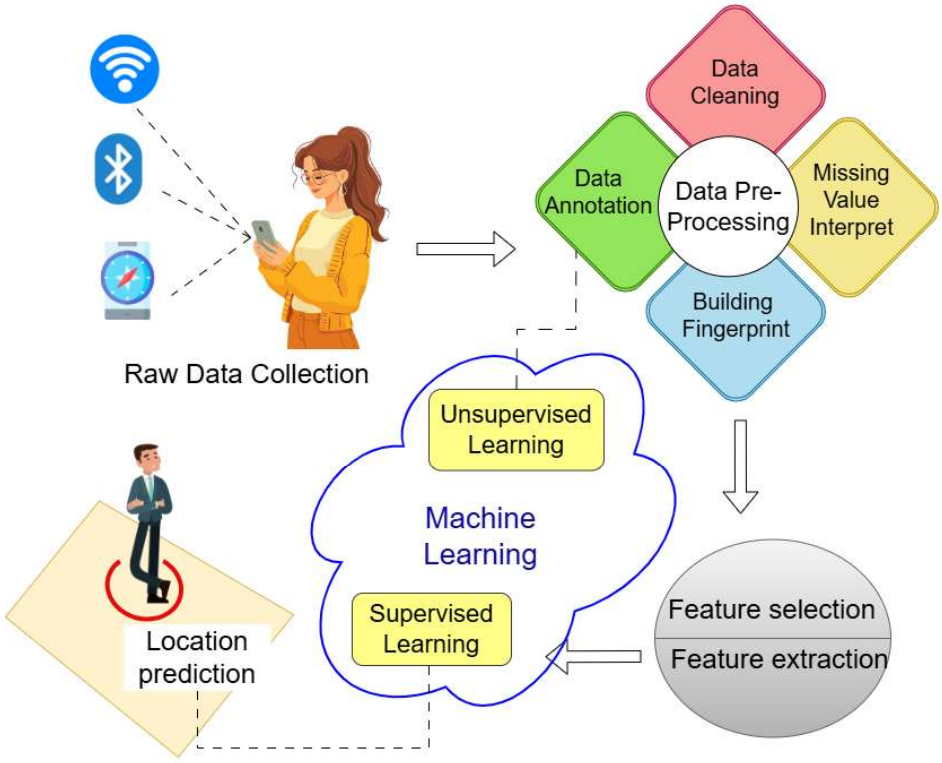


Figure 1.2: Machine Learning Framework for Indoor Localization

large group of people using individual mobile devices. The collected data are assigned to some particular location based on the users' feedback. In fact, volunteers could mark the location on the map while collecting the data at that location. Such labeled data forms the training set and the location label serves as the ground truth.

Robots are also used for automated data collection, connected with some IoT device like ESP32 or ESP8266. The robot navigates in the indicated direction and collects data for location points and tags those data. Channel State Information (CSI) is another WiFi signal measure that can be received through NodeMCUs [36]. CSI indicates channel quality and could also be utilized for positioning and navigation. However, this could not be measured through smartphones.

1.2.2 Data Preprocessing

Different raw datasets require different types of preprocessing techniques so that they can be used for constructing machine learning models. Raw data are reshaped into fingerprint

format, which is a vector of signal/sensor values with respect to each feature. When all data are labeled, there are some missing entries observed in case of WiFi or BLE fingerprint data. This happens because an AP cannot cover the entire experimental region. For instance, a smartphone would record very feeble or no RSSI from an AP installed at a different floor. Since a dataset cannot be fed with missing entries to ML model, a dummy value is used in those places to represent weakest signal logically. Alternatively, if for a particular AP, an instance contains empty value but all other instances contain data value, then location-specific mean or median values can be entered at the empty place [25]. Data are then cleaned by removing unwanted noise from the dataset,commonly known as filtering. Moving average filter and median filter are used to smooth inertial sensor data removing signal noises [27]. Outliers are identified and removed based on euclidean distance, Z-score [35] etc.

1.2.3 Feature selection and extraction

Features contain meaningful representation of the data for better classification. Large public indoor spaces like airport or shopping malls have numerous number of WiFi and Bluetooth sources, some of which are hot-spots coming from user's mobiles. Identifying stable APs and eliminating unstable APs or hot-spots is a crucial step in ML based indoor localization process. In magnetic fingerprinting, there are only three primary features which are the magnetic field components towards three dimensions. The number of features are often increased to enhance the localization performance by extracting more features from the tri-axial sensor readings applying statistical formula like mean, standard deviation, variance, skewness, and kurtosis of these magnetic field components.

1.2.4 Classification / Location prediction

The final step is classification. Fingerprint vectors are fed as training data and location labels are fed as class labels to the ML or DL classifier model. Some common classifiers used for this purpose are - k-nearest neighbor (kNN), support vector machine (SVM), decision tree (DT), random forest (RF), and artificial neural network (ANN). The entire dataset is divided into two parts, larger part (contains 70-80% of samples) is used to train

the classifier and smaller part (contains 20-30% of samples) is used to validate its learning efficiency. For extremely large spaces containing multi-building and multi-floor scenario comparatively complex classifier models are used like deep neural network (DNN) to learn the complex pattern over a significantly large feature space.

1.3 Motivation

The very first step of fingerprinting based indoor localization system (ILS) is data collection. Ideally, a real-life ILS should be trained on sufficient amount of data, distributed uniformly all over the region. A machine learning classifier can learn the class-respective data patterns only when each class has sufficient samples with limited intra-class variance [29]. Otherwise, the model leads to poor localization performance, especially in under-sampled or ambiguous areas [30]. High data density and consistent fingerprint patterns across all precise locations are ideal but often impractical. Some indoor areas, like control rooms or staff-only zones in metro stations, are restricted and accessible only during limited times, while public areas like platforms and ticket counters are generally accessible. Due to the high density of people and constant activity in such public transit spaces, repeating the labor-intensive data collection process is quite impractical. This in turn results in a class-imbalance in the dataset. Some classes have a greater number of data instances, and some have very limited number of data instances. As a result, ML model could get biased towards the majority classes. Interestingly, the WiFi RSSIs are sensitive to ambient contexts [31]. However, collecting data for a long duration considering different times and environmental contexts is not a realistic approach, as it is very time-consuming and requires a lot of human resource. Also, for crowded public spaces, data could only be grossly labeled and outliers could also get induced.

On the other hand, in most cases, the existing WiFi or Bluetooth infrastructure are reused for localization to make the system cost-effective [32]. Since the sole purpose of deployment of the APs are communication, and not localization, these APs are deployed in a non-uniform manner over the region based on requirements. Thus, it is quite obvious that some consecutive precise locations have their own strong RSSI fingerprints, but a portion of the entire floor may not be covered by sufficient no of stable APs. As WiFi

signals exhibit lesser distance sensitivity when they are weak, in such cases, two consecutive locations may exhibit similar RSSI fingerprints. This problem raises the query, then, how to decide the appropriate size of the grid? Or, what should be the level of granularity? In existing works [41] [33] researchers have used a fixed granularity level for the entire experimental region. The existing benchmark datasets on WiFi-based indoor localization also reflects this fact. Here, either the data is labeled as fixed sized grids as in [41], [33] or they are labeled by reference points as in [34].

Thus, scarcity of labeled data and non-uniform AP distribution are two chronic research challenges in ILS domain, especially for public indoor spaces which necessitates the investigation in specific two directions, presented as follows.

1. How to decide the granularity level of locations considering the non-uniform AP distribution over an experimental region?

2. How to address the issue of lack of sufficient amount of correctly labeled data across all locations?

1.4 Contribution

The above mentioned research challenges motivated us to lift two implicit assumptions of existing indoor localization works - (i) it is practically possible to collect sufficient labeled data from indoor public places beyond university lab setup and (ii) all areas inside an indoor space would receive strong signals from a significant number of WiFi AP or BLE transceivers. It is utmost important to design localization approaches to address these serious concerns. Accordingly, the contributions of this thesis are presented as follows.

- **Ensemble of clustering algorithm for grossly labeled data:** Fingerprint-based localization for large indoor areas is primarily contingent on laborious site surveys. To ease the flow of research, grossly labeled data collected using crowd-sourced methods could be clustered. However, the challenge here is to generate the clusters containing data samples that are really physically close. There may be some data points within a cluster that are physically positioned far away. Also, the count

and instances of these wrongly clustered data differ for different base clustering algorithms. To address this research gap, we have proposed an ensemble of clustering algorithm. The outcomes of different clustering algorithms are combined according to their rank to form an ensemble. Unlike conventional ensemble techniques, the proposed approach is executed in multiple iterations unless all the data points are clustered. In each iteration, the rank of the base clustering algorithms may change. Thus, the proposed approach aims to exploit the goodness of all the baseline clustering approaches. Thus, non-uniform clusters are obtained reflecting the non-uniform distribution of APs, and building properties. The gross labeling of data has been utilized to map the clusters to the respective zones in the indoor space. Finally, classification is performed in the online phase to predict the location of a user at a zone.

- **Signal to image encoding for two-phase indoor localization:** Existing research shows that fingerprint based indoor localization performance is greatly impacted by the fluctuations of WiFi signals. The instability in WiFi fingerprint pattern has been addressed in this chapter. Our contribution is three-fold. (i) An algorithm, namely, Image Encoding using RSSI Fusion (IERF), is proposed to generate RSSI images that emphasize the location-specific consistent RSSI glyph and de-emphasize the less frequent RSSI patterns. (ii) A lightweight CNN model CurveNet is proposed to predict locations classifying these encoded images. (iii)To make the entire localization system resource-friendly, localization is done in two phases-coarse-grained and fine-grained. At phase-1, a sub-region is identified in which a user is present and in phase-2, the user's fine-grained location within that sub-region is estimated. The output of Phase 1 (the predicted sub-region) is used as a gate or filter to determine which Phase 2 model (or sub-dataset) should be used for fine-grained positioning. For coarse-grained localization requirements where fine-grained location labels are not available, only phase-1 would be required.

- **Granularity optimization using k-disagreeing neighbor concept :** The precision of WiFi/BLE fingerprint-based indoor localization systems depend on the preciseness of data annotation. Since localization approach reuses the existing WiFi

infrastructure, non-uniform Access Point (AP) placement induces varying precision of localization service. To address this challenge, an algorithm, Granularity Optimization through k-Disagreeing Neighbor (GO-kDN) is proposed that analyzes the hardness of instances and combines instance spaces of different classes to obtain a coarse-grained class, only if the characteristics of their instance spaces are worth to combine. The non-uniformity of APs is thus translated to the varying granularity of target classes. GO-kDN enables the classifiers to perform equally well on such location classes having varying granularity levels. A benchmark dataset, MetroIndoorLoc has been designed[1] that contains RSSI fingerprints from available APs collected from two metro stations— one at grade level and another at underground level. Labeling errors or outliers are also handled through this kDN analysis method. However, due to non-uniform granularity, there is a significant chance of class imbalance that may arise in the final annotated dataset. To solve this, a data augmentation approach for minority classes is proposed using conditional generative adversarial network (CGAN).

- **Generating synthetic data and classify through Vectored Labelled Generative Adversarial Network (VL-GAN):** For wide-scale deployment of indoor location-based services, it is crucial to address the problem of laborious site-surveys for precise data annotation. Few recent works have focused on the applicability of Generative Adversarial Networks (GAN) to address the problem of limited fingerprint data. However, the presence of labeling errors and the limited number of data samples per class make the application of GAN challenging. To address this challenge, a Vectored labeled GAN (VL-GAN) model is proposed that introduces a vectored label generator acting as a classifier to address the above-mentioned challenges. This chapter makes three contributions: (i) a Label-to-Vector algorithm that transforms the location labels into distance vectors denoting the likelihood of a sample belonging to each location point (class); (ii) a VL-GAN model that consists of a generator, a discriminator, and a classifier (a label generator); (iii) a Set-wise RSSI Fingerprint Generation (SRFG) strategy to ensure higher-quality data generation

[1] https://drive.google.com/drive/folders/1lCxKDRTplT1TMykPXo5ikDft9K9wMUhV?usp=sharing

even for fine-grained locations. An adaptive loss strategy has been applied to the whole process to make it faster yet stable towards convergence.

1.5 Organization of the Thesis

The thesis makes original contribution in the domain of addressing data scarcity and non-uniform AP distribution, in the context of RSSI-fingerprint based indoor localization. Organization of the thesis is presented as follows.

- *Chapter 2:* Since this thesis addresses two research challenges commonly faced in public indoor spaces, in this chapter, the recent existing localization approaches for public indoor spaces are surveyed, research challenges are identified and common technologies used in this domain are discussed.

- *Chapter 3:* In this chapter, a semi-supervised localization solution is proposed for any grossly labeled RSSI fingerprint dataset. A Rank-based Iterative Clustering (RBIC) algorithm is proposed that provides a clustered dataset with negligible chance of containing physically apart location points within a common cluster. Hence, it provides a data annotation strategy so that localization can be achieved using any grossly labeled fingerprint dataset. As the region is clustered based on RSSI fingerprint characteristics, the non-uniform distribution of APs is reflected in the non-uniform clusters obtained as outcome of the ensemble of clustering.

- *Chapter 4:* In this chapter, a resource-friendly two-phase indoor localization solution is proposed. In 1^{st} phase, the larger sub-region on the floor is identified where user is present. In 2^{nd} phase, user's precise location is estimated. To obtain user's precise location, fingerprints are encoded into images using a proposed signal-to-image encoding algorithm using fusion of fingerprints, so that the location-specific consistent fingerprint pattern can be highlighted. Using a proposed lightweight CNN model these images are classified and user's fine-grained location is obtained.

- *Chapter 5:* In this chapter, the challenge of non-uniform distribution of APs over the experimental region is investigated. An algorithm is proposed that analyzes instance

hardness applying k-disagreeing neighbor concept, and accordingly merges consecutive small regions over the floor that lack significant strong fingerprint pattern. In this way, an asymmetric grid structure over the floor is obtained, maintaining non-uniform granularity level. A benchmark dataset is collected from two metro stations (one at grade level and another underground level) for experimentation and made available for research purpose. Moreover, after granularity optimization, synthetic data are generated for those areas using CGAN that lack sufficient amount of data samples to further enhance the localization performance.

- *Chapter 6:* In this chapter, synthetic RSSI fingerprint data generation using a Generative Adversarial Network (GAN) is investigated. Since GAN was originally designed for generating unlabeled data, at first we fed location-specific data separately to the model, and it was trained separately each time. Although augmenting these data enhanced the localization accuracy, the whole process seems very time consuming. To solve this issue, we proposed a new GAN variant, Vectored Labeled GAN (VL-GAN) which has three components - generator, discriminator and classifier. The generator generates synthetic RSSI fingerprint data samples, the classifier generates their corresponding labels in vectored form, indicating the likelihood of the sample belonging to each class. The discriminator does two job- it identifies a sample as either real or synthetic, and it also identifies a vectored label as real or synthetic.

- *Chapter 7:* In this chapter conclusion of the thesis is presented with summarized contribution and consolidated findings. Limitations and future scopes are also discussed.

Chapter 2

Survey of Machine Learning Based Localization in Public Indoor Spaces

Increased adoption of different machine learning techniques for indoor localization could be witnessed in the recent literature. Most of these existing works have chosen some public indoor space such as university, shopping mall, hospital etc for their experimental analysis. In this chapter, a survey is presented that highlights the different methods and technologies used in these existing related works, and summarizes the research gaps found from the survey.

2.1 Existing Machine Learning Based Solutions Using Different Technologies

Machine learning-based indoor localization widely relies on technologies like WiFi, BLE, UWB, and RFID for signal-based positioning, along with vision systems and inertial sensors. Hybrid approaches combining multiple technologies are often used for improving precision and robustness.

In [51], authors proposed an indoor localization system based on seven WiFi sources covering four rooms using a smartphone. Based on the RSSI value, the system decides in

which room the user is present. Supervised classifiers, namely, kNN, ANN, Naive Bayes, Decision Tree and SVM, have been used to train the system along with Extreme Learning Machine (ELM). kNN possesses the highest accuracy among all. These kind of systems solely based on limited number of WiFi sources might suffer with a problem if some APs are removed or their positions are changed. Researchers proposed a WiFi based indoor localization approach in [49] that identifies altered APs based on the relationship between AP-specific RSSI values. The RSSI values of the APs are predefined by the Gradient Boosting Decision Tree (GBDT) regression model. Mismatch of RSSI in the crowd-sourced data and predefined regression model determines the existences of altered AP. If altered APs are detected, an algorithm based on GBDT regression has been designed to update the radio map accordingly. Finally, a Weighted kNN (WkNN) algorithm is designed to evaluate the effectiveness of the radio map updating.

Another WiFi fingerprinting based localization approach has been proposed in [77] where kNN is used for classification. Distant location points having similar fingerprints or mismatch of observed fingerprints in the training and testing phase due to wide fluctuation of WiFi signals, may affect the performance of kNN. To solve these, a soft range limited K-nearest neighbors (SRL-kNNs) approach is used. The fingerprint distance is scaled by a range factor related to the physical distance between the reference point and the user's previous position. To solve the signal fluctuation problem, RSSI histogram is used in distance computation. Researchers proposed another WiFi based indoor localization system in [43] evaluated with multiple classifiers such as kNN, Decision Tree, Random Forest, and SVM. A fingerprint database is generated based on the signal strength received at known grid points. Comparing the user fingerprint and stored fingerprint, the best matching pattern in the radio map is found. Improved accuracy is observed for kNN and the Random Forest classifiers, whereas, SVM and the Decision tree do not perform well in the uncorrelated space.

One of the common problem of WiFi fingerprint based system is the manual adaptation of the large fingerprint dataset. In [56], Bluetooth low energy (BLE) is fused with WiFi and an algorithm named as i-KNN has been developed that filters the initial WiFi fingerprint dataset (the radio map) based on the distance between the user (carrying mobile device)

and the BLE devices. A subset of the initial fingerprint dataset is obtained containing possible locations of the user and this reduced dataset is used to estimate final positions.

Unsupervised learning is also found to be applied to this field. A WiFi RSSI based indoor localization method has been proposed in [57] that uses semi-supervised learning in a hierarchical way. In the system, at first k-means clustering is used that divides the environments in partially overlapping zones. Then, kNN and SVM have been used to predict the precise location. This hierarchical process simplifies the offline training and online localization phases and reduced overall computational complexities.

Some researchers have used ensemble learning mechanism to make a localization approach learn in repetitive way. In [58], a feature-based ensemble model was proposed to select subsets of APs in terms of optimal features to enhance the localization performance. Ensemble classifiers perform well in handling different types of heterogeneities also, for example, device heterogeneity, context heterogeneity, and temporal heterogeneity [59]. In [44] also, the well-known ensemble classifier model random forest has been used in which decision tree runs multiple times. The indoor localization system is based on Bluetooth technology that tracks user movements within a building to enhance smart building activities. For RSSI received from each iBeacon, multiple decision trees are used. Then, depending on their value, the decision is taken in the intermediate steps of the tree regarding which location points they indicate. Each of the leaf nodes of the decision tree denotes a unique class (location). All decision trees may not result the same, the result received from the majority is labeled as the right class. In [60], another indoor localization system has been proposed applying random forest. In that work, random forest based cross validation was used to train the system using WiFi fingerprint data. Problem of overfitting has been reduced to a great extent by using this random forest in the system. In [61], an ensemble of classifiers is designed based on the principle of weighted majority voting to mitigate the effect of different hardware sensitivities in different smartphones for WiFi based systems. The system applies Dempster-Shafer belief theory to calculate weights of the base learners so that a sustainable performance could be ensured even for smartphones for which no training data exists.

Use of ANN is observed increasingly in recent research works. In [52], researchers have

used integration of kNN and ANN with back propagation to reduce the error rate of an indoor localization system. Another similar approach is discussed in [53]. Here, ANN has been used to reduce the error in a RSSI based indoor localization system. The initial results of localization are fed into the ANN. A trained ANN needs to be trained again for changing environment. However, the amount of error reduction depends on the number of hidden layers of the ANN.

Use of advanced machine learning techniques is increasing in the field of localization. In [62], authors proposed a deep learning-based indoor localization approach (named as DeepPositioning) based on fingerprinting. For fingerprinting, a combination of RSSI of WiFi and the magnetic field value is taken. The system has been evaluated through practical experiments.

A deep learning based approach has been used in [63] for indoor localization using WiFi. The main objective of the proposed method, named as DLoc, is to detect the reflectors and obstacles present in the path of signal propagation. An automated mappig platform named as MapFind is designed that creates the indoor environmental map during collection of WiFi data. A 2D heatmap is fed to ANN containing the information of angle-of-arrival and time-of-flight. The ANN observes the impact of the environment on the relationship of wireless signals and ground truth locations and based on the implicit representation of the relationship learned by the ANN, unlabeled data points can also be identified. In [64], authors have proposed a deep learning based autonomous indoor robot navigation approach. The environment is represented by a graph with nodes indicating 'perceptual behaviours' including buildings, corridors, rooms and edges indicating the 'navigational behaviours' such as crossing a corridor, entering or leaving a room etc. The robot builds the environmental representation from a visual input. Then, it learns the semantic information regarding navigation using deep learning techniques (mainly, supervised imitation learning) and uses it to navigate autonomously. The proposed approach seems applicable for simple environments only where the human navigational behaviours are predictable, complex situation demands further experiments. Another indoor robot navigation approach has been proposed in [65] based on visual map learning and CNN. In-

stead of training the CNN with randomly initialized weights, it is pre-trained with transfer learning method. The knowledge gained on the large dataset ImageNet[66] is transferred to the CNN.

Table 2.1: Different ML Models Applied in Indoor Localization Solutions Using Various Technologies

Exist. work	**Year**	**Technologies**				**ML / DL Models**	**Performance Metric**
		WiFi	**BLE**	**IMU**	**RFID**		
[43]	2016	✓	✗	✗	✗	kNN, Decision Tree, Random Forest, SVM	Accuracy improved by 41.44%
[67]	2017	✗	✗	✗	✓	ELM	Accuracy 84.25%
[62]	2017	✓	✗	✓	✗	DNN	Accuracy improved by 30%
[51]	2018	✓	✗	✗	✗	kNN, ANN, Naive Bayes , Decision Trees, SVM, ELM	Accuracy 98.75%
[52]	2018	✓	✗	✗	✗	kNN, ANN	Loc. error<0.9 m
[53]	2018	✗	✓	✗	✗	ANN	--
[54]	2018	✓	✗	✗	✗	Affinity Propagation	Loc. error < 1.99 m
[68]	2019	✗	✗	✓	✗	DNN	Mean loc. error 2.84 m
[69]	2019	✓	✗	✗	✗	Transfer Learning	Loc. error 0.3749 m
[57]	2020	✓	✗	✗	✗	K- Means, kNN, SVM	Accuracy improved by 1.4-3.2%, error reduced by 10-22%
[63]	2020	✗	✓	✗	✗	ANN	Median Error: 94 cm
[37]	2020	✓	✗	✓	✗	Transfer learning	Accuracy 97.1%
[44]	2021	✗	✓	✗	✗	Random Forest	--
[55]	2022	✓	✗	✗	✗	Affinity Propagation	Mean error reduced by 1.5 unit
[70]	2022	✓	✗	✗	✗	K- Means	Average error 1.51 m

A radio-frequency identification (RFID) location tracking method using ELM has been presented in [67]. The method is based on the fusion of RSS and phase shift. A fingerprint

database is constructed using the RSS values of reference points. These data are normalized and fed to an ELM to estimate the positions. Due to extreme fast speed of ELM, the offline training and online localization phase is complete in much less time.

Different classification models applied in various indoor localization technologies are summarized in Table 2.1. It is observed that researchers are appreciating ubiquitous low-cost technologies for indoor localization purposes. WiFi is found to be very common – whether it is traditional supervised classification approaches or deep neural network or unsupervised learning like K-Means, affinity propagation, as well as recent emerging methodologies like transfer learning or extreme learning machine. Bluetooth, inertial sensors and RFID are also used for the purpose solely or in combination with some other technologies, but WiFi is found to be the popular standalone technology. Availability of ready-to-use infrastructure and a comparatively long range of coverage areas may be the reason. However, despite of lot of advantages, there are some research challenges also that come with these solutions, as presented in next subsection.

2.2 Research Challenges

Integrating indoor localization into various technologies and techniques and deploying it in various contexts calls for a number of important research challenges. Challenges that commonly arise in existing literature are discussed in the following subsections, along with a few currently proposed approaches to address them.

2.2.1 Identifying Varying Granularity of the Area

There is no common granularity level for indoor areas. The localization can be room level, reference point wise or location point wise where each location point is considered to be a grid of varying sizes, such as, $2m \times 2m$, $1m \times 1m$ for different experimental region. However, an indoor localization system deployed for some particular venue follow a pre-decided fixed granularity level throughout different floors of the buildings. Interestingly, the difficulty of localization increases with the demand for the precision. For example, detecting a person in a $1m \times 1m$ granularity is more challenging than detecting him/her at $2m \times 2m$ granularity or greater. Thus, accuracy varies at different granularity for the

same experimental region.On the other hand, the demand for granularity also varies with applications. Robot navigation or navigating autonomous vehicles at indoors demand more precision than user navigation at a hospital or underground markets. Precise positioning at a finer granularity can be achieved using a combination of sensors and considering latitude-longitude positioning along with virtual grids [39]. Stable source of sensors, for example, stable WiFi APs can provide consistent performance for varying granularity levels [41].

2.2.2 Multimodal Dataset Analysis

In any indoor area, sometimes there are not enough signal sources present from any particular one kind. For example, underground areas (like basements, subways) might lack enough number of WiFi or BLE source. Again, large public gatherings (concerts, stadiums etc) can sometimes overwhelm available WiFi, making it slow or unreliable. Thus, some situations and venues require a fusion of multiple technologies, and multimodal datasets where single type of sensor is not sufficient. Usually, there are some technologies commonly used solely for localization, such as, WiFi [42] [43], Bluetooth [44] [45], ultrasonic sensor[46], inertial sensor [47] [48]. However, they are fused with one another or with some add-on sensors whenever needed. Technologies such as light sensor, proximity sensor, thermometer, and barometer, these are not sufficient to perform the whole task alone and commonly have been used in combination with some other technologies.

The need for sensor fusion becomes more prominent when the system continues to estimate locations with changing environments, i.e. when seamless positioning takes place. For seamless indoor-outdoor localization environment detection, in [38], GPS signal features (elevation and SNR) and magnetometer reading both have been used whereas smartphone sensors to measure atmospheric pressure, temperature, relative humidity etc. have been used with GPS in [40]. In [37], WiFi RSSI values are considered along with mangnetometer reading as input data to a deep neural network. Multi-sensor datasets, especially, multimodal combinations require novel feature extraction and selection techniques in order to combine the best of each sensing modalities. Classification techniques can also be layered to incorporate features from different sensors in a hierarchical fashion.

Table 2.2: Research Challenges and Their Impact on Localization in Public Indoor and Outdoor Spaces

Challenge	Existing work	Remarks
Identifying Varying Granularity of the Area	[41] in 2019	Selction of stable APs helps to improve postioning granularity.
	[39] in 2021	
Multimodal Dataset Analysis	[40] in 2016	Additional sensor data such as atmospheric pressure, relative humidity, temperature help to distinguish the IO environment.
	[37] in 2020	Performance degradation due to signal fluctuation of sensors like WI-Fi can be balanced by using other sensor readings, such as a magnetometer.
Repetitive Site Survey for Manual Adaptation of Fingerprints	[50] in 2016	To automatically identify the altered APs, methods like Gaussian Regression, Gradient Boosting Decision Tree (GBDT) can be used.
	[49] in 2019	

2.2.3 Repetitive Site Survey for Manual Adaptation of Fingerprints

To validate a new approach or compare different localization strategies using standardized evaluation metrics, some benchmark datasets are needed covering an indoor area which is large enough. In fingerprinting based positioning, a site survey is a time-consuming and laborious procedure. For instance, in case of WiFi based positioning, if an Access Point (AP) is damaged or its position is changed, the stored fingerprints for that AP could not be utilized for training purpose. Thus, maintainance of accurate fingerprints appears to be an infeasible task.

Researchers proposed a crowdsourcing approach in [49] that automatically identifies the altered APs using Gradient Boosting Decision Tree (GBDT). Using the relationship between RSSI of the altered APs and the unaltered APs in the dataset, a prediction model is trained to update the fingerprints. In [50], a similar approach is proposed that uses Gaussian Regression to adapt the fingerprints of the altered APs. This method achieves around 20% positioning error reduction compared to the other traditional schemes. However, crowdsourcing calls for its own disadvantages, such as, precise data annotation and usage of different devices with varying sensitivity for data acquisition. The above mentioned research challenges are summarized in Table 2.2.

2.3 Summary

Analyzing the recent literature on machine learning based indoor localization, it is observed that the system relies on multiple technologies including WiFi, BLE, RFID, UWB, inertial sensors or inertial measurement unit (IMU) etc. WiFi is found to be one of the most common technologies for indoor localization, especially because of WiFi infrastructure are already installed in most public indoor environments (offices, shopping malls, airports, hospitals etc) and mobile devices (smartphones, tablets, laptops) have WiFi receivers built in, requiring no extra hardware. However, despite of string potentials of WiFi based fingerprinting methods, there are some significant challenges also, as explained earlier. In this thesis, two among the three mentioned research challenges are attempted to investigate and address – repetitive site survey and varying granularity of the area. Another challenge (multimodal dataset analysis) is more faced in seamless indoor-outdoor positioning rather than solely indoor localization, so in future it can be investigated in some research which is more focused towards seamless positioning. The next chapter presents a proposed work to address the repetitive site survey following a semi-supervised approach, using WiFi fingerprint data, as follows.

Published Work

Manjarini Mallik, Ayan Kumar Panja, Chandreyee Chowdhury, Paving the way with machine learning for seamless indoor–outdoor positioning: A survey, Information Fusion, Volume 94, 2023, Pages 126-151, ISSN 1566-2535, doi:10.1016/j.inffus.2023.01.023

Chapter 3

An ensemble of clustering algorithms using grossly labeled data

RSSI fingerprint-based localization for large public indoor areas is primarily dependent on laborious site surveys, to ensure sufficient amount of labeled data for training. To ease the flow of research, grossly labeled data are often collected using a crowd-sourced method. The real challenge is to annotate those grossly labeled data analyzing their RSSI characteristics with proper location information, so that a chunk of data annotated with common location class contains physically close samples. In this chapter, a two-phase semi-supervised indoor localization approach is proposed, which is general enough to be applied to indoor localization datasets. In the offline phase, a Rank-Based Iterative Clustering (RBIC) algorithm is proposed that generates a clustered dataset with a negligible chance of containing physically apart location points within a common cluster. RBIC can be viewed as a clustering ensemble model. Different clustering algorithms are selected as baseline algorithms and assigned unique ranks depending on well-known clustering scores to be fed as input to RBIC. In the online phase, the users' location is estimated in terms of cluster ID, using trained ML classifiers based on the dynamic RSSI vector received through its handheld smartphone.

3.1 Two-phase semi-supervised indoor localization approach

The proposed work is executed in two phases- offline and online. In the offline phase, clustering is applied to divide the experimental region into a number of zones that reflect the building properties and hence, similarity in RSSI fingerprints. A Rank-Based Iterative Clustering (RBIC) algorithm (Algorithm 1) is designed for this purpose. These zones are treated as annotated location points for the online phase. In the online phase, the user's location is estimated based on the test data values received in the user's device through a supervised classifier that is tuned based on the outcome of the training phase. The two phases are detailed as follows.

3.1.1 Offline Phase

The main goal of the offline phase is clustering the dataset into zones for which the RBIC algorithm is proposed. The workflow of the proposed algorithm is represented in Figure 3.1. The major steps of the algorithm is discussed as follows.

Selection of clustering algorithms: The selection of a set of clustering algorithms is the first step in the process. There must be a minimum of two algorithms, and this number cannot be increased beyond a certain point. Outstanding performance is not needed, but it is preferable if each algorithm performs at least satisfactorily on the chosen dataset because the final result will depend on how well each algorithm performs when combined. This can be done by assessing their output using a few internal validation clustering scores for different values of n (number of clusters). Any one of the clustering algorithms can be chosen if two or more of them have scores that are nearly identical, indicating moderate performance. In this work, we considered four base clustering algorithms, namely K-Means, Hierarchical, GMM, and Birch. However, researchers can investigate with any base clustering algorithm of their choice where the target number of clusters, i.e., n, is provided as an input.

Selection of number of clusters: The number of clusters can be selected based on an

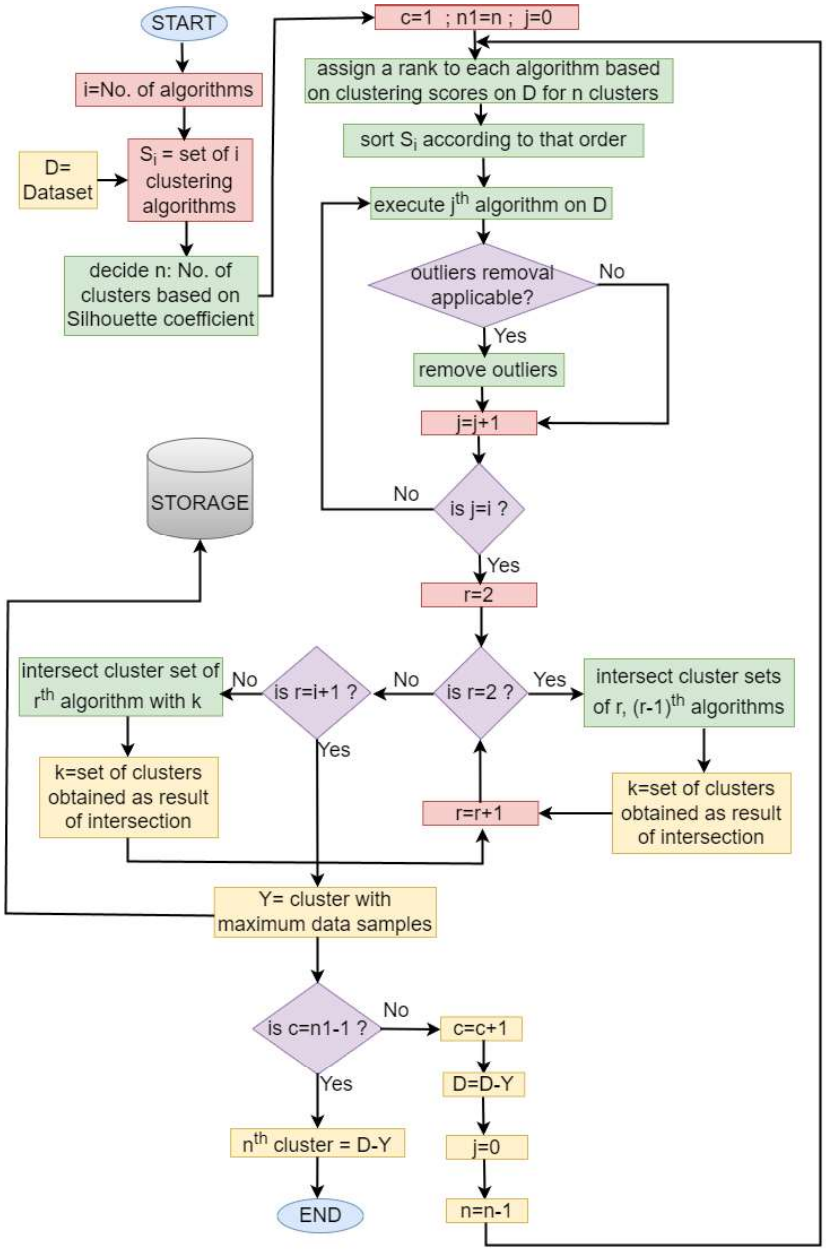


Figure 3.1: Workflow of Proposed Rank Based Iterative Clustering (RBIC) Algorithm

internal validation score, obtained as a result of experimentation with different numbers of clusters for different clustering algorithms. The internal validation indices that are considered include Elbow method, Silhouette coefficient and Calinski Harabasz index. The experimental observations with these evaluations are explained in the Section 3.2, for respective datasets. Since each clustering algorithm acts differently on specific datasets, it is difficult to obtain the best performance of all of the algorithms for common values of n. Hence, in this work, the n has been selected in such a way that the scores of all of the algorithms are at least satisfactory.

Outlier removal: Noise or outliers are practical problems that reduce a sensor-based system's performance. Before introducing the data to an ML method, researchers use a variety of techniques on it to solve the issue [72]. Since Wi-Fi operates on a radio frequency range, wireless interference can obstruct or interrupt the transmission. For this reason, there is a high chance of presence of few data samples in the dataset that do not reflect the characteristics of its corresponding location point in terms of RSSI values. These are the outliers, and needed to be identified and removed to improve the localization performance. Figure 3.2 represents the outliers of a WiFi fingerprint based indoor localization dataset. In the set of clustering algorithms, there must be at least one algorithm that is good enough to identify and remove outliers. Since this particular algorithm's result will be combined with that of other algorithms, its benefit of outlier removal will be reflected on the overall clustering result.

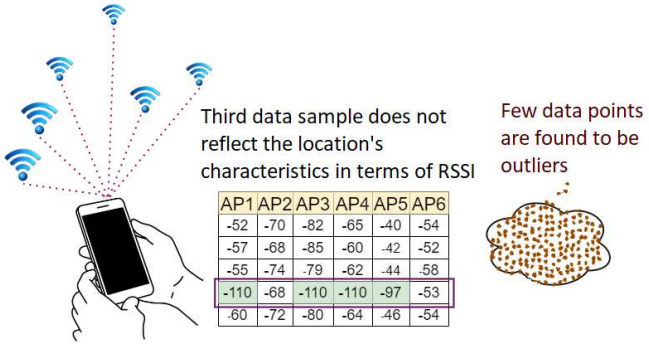


Figure 3.2: Outliers in WiFi Fingerprint Dataset

Algorithm 1: Rank Based Iterative Clustering Algorithm

```
Input: No. of clusters n
       Dataset D
       A set of i clustering algorithms S_i={A_1,A_2..... A_i}
Output: A set of n cluster
initialization;
n1←n
for c in 1 to n1-1
        for algo in S_i
                assign rank to algo
        sort S_i based on rank
        for algo in S_i
                execute algo(D) to generate n clusters
                if outlier removal is applicable then
                        remove_outliers(D , n)
        r←2
        while r < i+1
                if r=2 then
                   K ← intersect (clusters(S_i[r]) , clusters(S_i[r-1])
                else
                   K ← intersect (clusters(S_i[r]) ,K)
                increment r by 1
        if r=i+1
                Y←cluster with maximum data samples
                if c=n1-1 then
                   nth cluster=D-Y
                else
                   increment c by 1
                   D=D-Y
                   decrement n by 1
end;
remove_outliers(D, n):
        i_max=maximum iteration
        for i 1 to i_max:
                apply clustering algorithm on D with n clusters
                for each data point
                        APPEND (distance_vector , Euclidean distance(data point,
centroid[cluster]))
                dmax = distance_vector.max()
                T ← predefined threshold for outliers detection
        for each cluster
                outliers=(distance(centroid,row)/dmax>T
                newcluster = (distance(centroid, row) / dmax<=T
```

Ranking of clustering algorithms: When the number of clusters, n is fixed each baseline algorithm is assigned a rank based on a standard internal validation score to evaluate the goodness of the clustering structure for that specific value of n. In our work,

Silhouette coefficient and/or Calinski-Harabasz index have been used to obtain the internal validation score. The implementation of it is detailed in Section 3.2 with experimental observations.

Intersection of clusters in each iteration: To generate n clusters, the proposed RBIC algorithm iterates (n-1) times. In the first iteration, intersection is performed on the two sets of n clusters that are generated by the first and second ranked baseline algorithms. The intersection is performed among each possible pair of clusters (a, b), where a belongs to the cluster set of first ranked algorithm and b belongs to the cluster set of second ranked Baseline clustering Algorithm (BA). The reason is that Cluster 1 generated by BA1 may exhibit highest similarity to Cluster $i(\neq 1)$ generated by BA2. The temporary clusters obtained in the outcome of the 1^{st} intersection are stored separately. After that, these temporary clusters are intersected with the n clusters generated by the third ranked baseline algorithm. Here, while intersection, the data points that are there in the temporary cluster would be given higher precedence. If two datapoints a and b are in one temporary cluster and the BA3 has kept only a in one cluster, then, during the intersection, datapoint b may be retained in the outcome with probability p. This is done to put more emphasis on the outcome of the top ranked baseline algorithms on the outcome. However, if all the selected baseline algorithms perform equally well, then we can keep $p = 1.0$. Thus, ordering of the intersection would not matter in that case. This process goes on for all baseline clustering algorithms as represented in Figure 3.3. After the first iteration, one of the n final clusters is obtained, covering the maximum number of data points among all the clusters generated in the last intersection of the current iteration. All of these data points of the final cluster are stored separately and removed from the dataset before the next iteration. A reduction in the number of data points will in turn gradually reduce the overall intricacy of total intersections in consecutive iterations.

There would be a total of n iterations where after each iteration, one cluster of the final set would be generated and the corresponding datapoints would be removed from the input dataset. This entire clustering phase happens only once during training. After this is over, data would be annotated by the cluster ids that will indicate the different regions of the total experimental area. The intersection of different clustering algorithm ensure

better balancing of the bias of the individual outcomes.

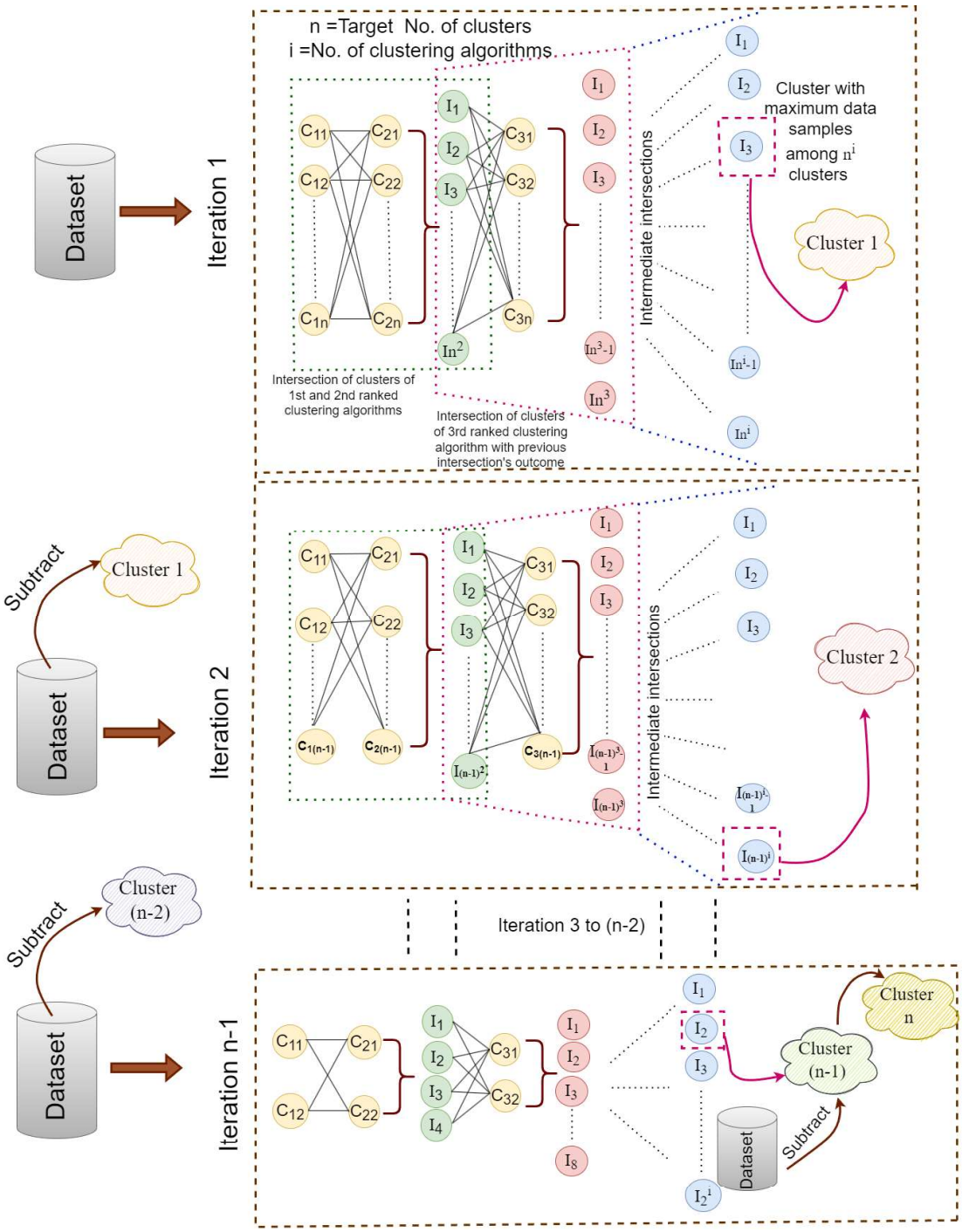


Figure 3.3: Intersection of Clusters in Proposed Rank Based Iterative Clustering (RBIC) Algorithm for n Clusters and i Clustering Algorithms

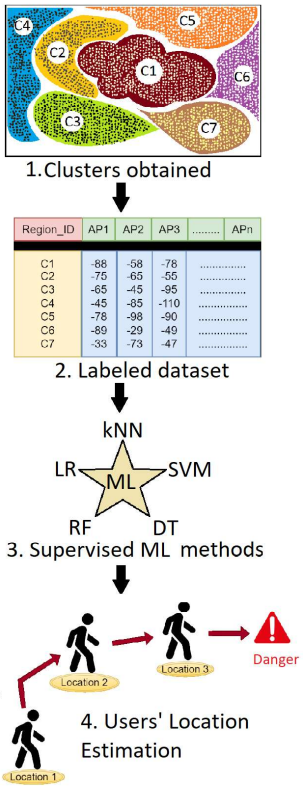


Figure 3.4: Location Estimation Based on Supervised Learning

3.1.2 Location Estimation

After successful clustering, the entire experimental region has been divided into n clusters as the building properties would get reflected in the unsupervised data analysis. Thus, these n clusters would signify designated zones of the indoor space and hence, can be treated as class labels. Thus, with the help of a building floorplan and gross data labels already available with the dataset, these n clusters can be assigned class labels that indicate designated regions in the floorplan. A location estimator model is now trained with this labeled dataset as the training data using supervised ML classifiers. This made the location estimator model capable of estimating unknown locations of a user moving along that experimental region, by collecting real-time test data in user's device. The pictorial presentation of the location estimation process is shown in Figure 3.4. Thus, during the online phase of the proposed framework, only the location estimator gets executed that is a supervised classification process.

3.2 Experimental Analysis

We have implemented the proposed approach on three benchmark WiFi fingerprint datasets, Dataset_1 [1] (JUIndoorLoc), Dataset_2 [2] (Nexus-5) and Dataset_3[3] (SODIndoorLoc) for which the floorplans and the ground truth locations on the floorplan are available. This is so chosen to show the validity of the algorithm on the floorplan. A small portion (20%) of the data is kept separate solely for the test purpose required by the location estimation phase. The datasets are labeled here, though the labeling information is not utilized by the offline clustering phase of the proposed framework. The labels are removed before the datasets are fed into the baseline clustering algorithms as input. The clusters are obtained based only on the RSSI characteristics. After execution of the proposed RBIC algorithm, each data point of each cluster is mapped to the original dataset to get the corresponding location point, and that location point is marked on the floorplan with a specific color shade indicating that particular cluster. Application of this domain knowledge is required to check the effectiveness of the clustering algorithms.

In this section, the experimental setup is described briefly, followed by performance analysis of individual clustering algorithms, RBIC algorithm and location estimation process for all three datasets.

3.2.1 Experimentation on Dataset_1

In Dataset_1[73], data was collected over three floors of a building, among which, experimentation is carried out on the data of one floor where the data shows many different but realistic patterns due to the presence of classrooms, a seminar hall, a laboratory, and faculty rooms.

3.2.1.1 Experimental Setup

In Dataset_1, RSSI values from 120 APs present in a floor was collected for different location points having 1×1 sq. m. area. The entire floor was divided into 42×21 grids, resulting in total 882 location points. Data was collected from these location points except

[1] https://drive.google.com/drive/folders/1_z1qhoRIcpineP9AHkfVGCfB2Fd_e-fD
[2] https://ieee-dataport.org/open-access/wifi-rssi-indoor-localization
[3] https://github.com/renwudao24/SODIndoorLoc

staircases, toilets and the upper-leftmost room. It is expected that consecutive location points receive signals of similar strengths from common APs, based on which a bunch of location points could be placed within a common cluster.

3.2.1.2 Performance Analysis of Individual Clustering Algorithm

K-means, Hierarchical, Birch, and Gaussian mixture model (GMM) algorithms were initially selected for experimentation on this dataset. At first, an optimal value of K was found using the Elbow method. The point of inflection is found at K=7 on the curve, which indicates that the model fits best at that point, as represented in Figure 3.5. The distortion score indicates the sum of squared distances from each data point to its cluster center.

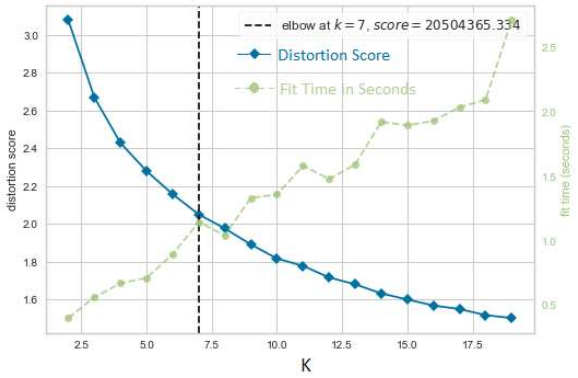


Figure 3.5: Elbow Method for Finding The Best Value of K for The K-means Algorithm on Dataset_1

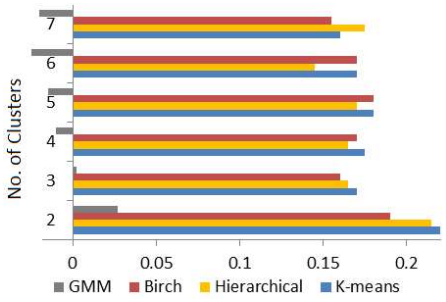


Figure 3.6: Performance Comparison of the Baseline Clustering Algorithms Based on the Silhouette Coefficient on Dataset_1

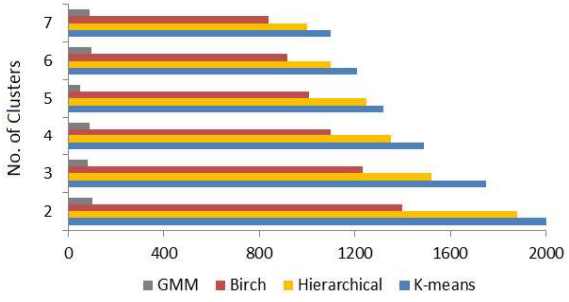


Figure 3.7: Performance Comparison of the Baseline Clustering Algorithms Based on the Calinski-Harabasz Index on Dataset_1

Since for K-means algorithm the optimal number of clusters is found to be 7, we have checked the goodness of performances of other three algorithms for n=2 to 7 with Sil-

houette coefficient [74] and Calinski-Harabasz index [75]. This analysis will indicate the influence of choosing different numbers of clusters for each of the base classifiers, which in turn will have an impact on RBIC's performance. In Figure 3.6, it can be observed that although K-means, Hierarchical, and Birch clustering methods performed satisfactorily for n=2 to 7, performance of GMM is poor. A positive coefficient value indicates good formation of distinguished clusters, whereas a negative coefficient indicates the existence of unrelated data points within a common cluster, or the clusters have been assigned in a wrong way. Figure 3.7 represents the performance of four clustering algorithms measured with Calinski-Harabasz index. This index represents a ratio of the total dispersion scores obtained for between-clusters dispersion and inter-cluster dispersion. A higher score indicates good clustering performance, whereas a lower score indicates poor clustering performance. Observation is similar with Silhoette coefficient, i.e., GMM is found to perform poorly and other two algorithms perform satisfactorily, securing a score of 1000 or higher.

To visualize the clustering performance applying domain knowledge, the outcome of K-means is plotted on the floorplan, as shown in Figure 3.8. In order to achieve this, each clusters' unlabeled data points were mapped to the original dataset, to obtain its corresponding location class. The obtained location point was given a distinctive colour specifying that particular cluster. In this way, all data points of seven clusters were plotted on the floorplan with seven distinct colors. All of the location points where data was collected, are found to be covered by one of the clusters.We can observe that the deep blue cluster (6^{th} from left) has a poor shape. Many of its location points are dispersed throughout the pink cluster (7^{th} from left). The reality that the data points within a cluster may not be physically closer together despite a particular unsupervised approach's satisfactory performance is effectively highlighted by this visualization. This in turn emphasizes the motivation to combine the clustering algorithms to retain the generality of the solution without too much bias on one algorithm.

From Figure 3.6 and Figure 3.7 we can observe that, to cluster the data in 7 clusters, Hierarchical clustering performs best according to Silhouette coefficient, and K-means performs best according to Calinski Harabasz index. GMM performs poorly for both. Since Birch method performed near to K-means and Hierarchical clustering and obtained

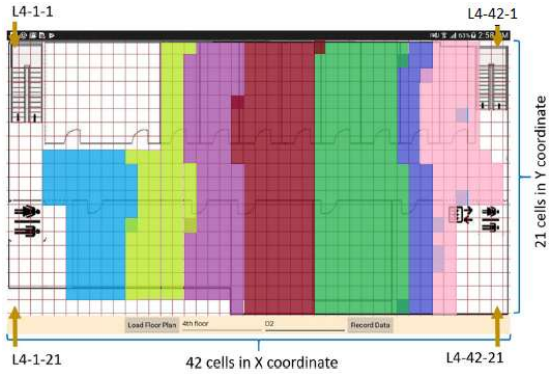


Figure 3.8: Outcome of the K-means Algorithm Plotted on JUIndoorLoc (Dataset_1) Floorplan. Dispersion of Cluster Data Points is Observed.

rank 3 for both the internal validation methods, it was not included in the final set of clustering algorithms. Since the number of clusters is significant, to make the RBIC algorithm implementable at the edge servers, the no of possible intersections should be kept manageable. Thus, to maintain the efficiency of the proposed framework, we select the outcomes of top 2 algorithms here as inputs to the proposed RBIC algorithm.

3.2.1.3 Performance Analysis of Proposed RBIC Algorithm

RBIC algorithm was executed on Dataset_1 with the selected baseline clustering algorithms. The clusters produced were plotted on the floorplan to demonstrate the correctness of the clustering. In Figure 3.9, although a few grids of consecutive cluster boundaries are found to be overlapped, it can be observed that each cluster mostly refers to the location points which are physically close.

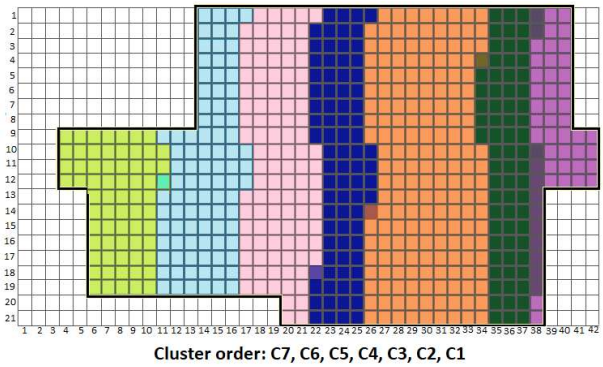


Figure 3.9: Proposed RBIC Algorithm's Result Plotted on JUIndoorLoc (Dataset_1) Floorplan

The performance of RBIC algorithm has been evaluated by three different internal

validation indices (Table 3.1). Silhouette score obtained with n=7 is 0.12, which indicates that the clusters are correctly assigned including data points of similar characteristics. The CH index obtained for RBIC algorithm with n=7 is 427.63, which is acceptable. The Davies Bouldin score is termed as the mean similarity measurement of individual clusters with its most similar cluster [76]. Here, similarity is defined as the ratio of within-cluster distances to between-cluster distances. The minimum score is 0, and increased with the degradation of the quality of the cluster. Davies Bouldin score obtained for RBIC algorithm with n=7 is 2.3, which is not much higher than 0, though there is a chance of further improvement.

Table 3.1: Evaluating performance of proposed RBIC using Internal Validation Indices on Dataset_1

Internal Validation Index	**Score**
Silhouette Coefficient	0.12
Calinski Harabasz	427.63
Davies Bouldin	2.30

Analyzing Importance of the APs According to Dominance Over the Clusters- Dataset_1 consists of 120 APs. This number is not uncommon for indoor public spaces and could even go up depending on local demand. It is therefore ideal to get an analytical perspective of how much each AP impacts cluster identification. If some APs in the future need to be replaced with new ones at a reasonable price due to damage, this knowledge might be helpful. First, intra-cluster analysis was performed by calculating the standard deviation of RSSI values received from each AP to a specific cluster in order to achieve this. A single AP's low intra-cluster standard deviation value suggests that the AP offers a stable signal strength throughout the cluster. The inter-cluster standard deviation of each AP for all seven clusters is obtained in order to discover the APs that are stable for a certain cluster but not equally stable for the other clusters.

Among 120 APs, the best 10 APs were selected, which have a lower intra-cluster score for a few clusters and the highest inter-cluster scores. Among these sets, an AP is selected as the dominating AP for a cluster if its inter-cluster score is greater than its intra-cluster score for that particular cluster. Figure 3.10 represents a network of all the dominant APs for all the clusters. We can observe that most of the consecutive clusters (refer to Figure 3.9) have some common dominating APs ; for example, clusters C4 and C5 have seven

common dominating APs. Interestingly, it is to be observed that the AP55 dominates three consecutive clusters, C5, C6, C7, and another far-cluster C1. Mean RSSI received from this AP is less than −100 dBm for all clusters (refer to Figure 3.11). That is, the signal is merely received all over the experimental region, but at certain points, due to the fluctuating nature of the WiFi signals, it may be received with a significant value, and the overall intra-cluster standard deviation of that cluster increases. However, such APs are not highly important and could be placed behind other dominating APs for some specific cluster in the priority list.

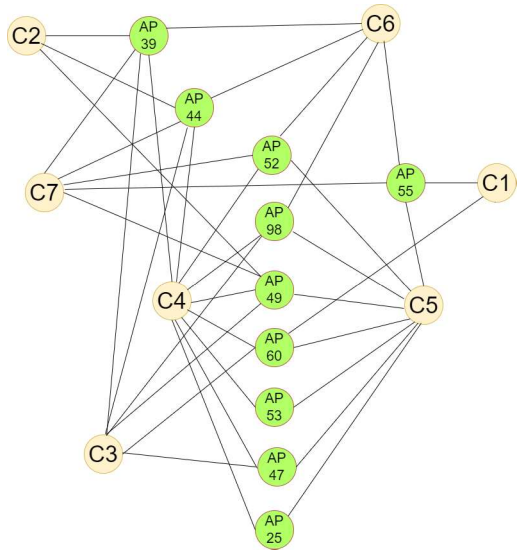


Figure 3.10: Dominating APs Connected to Respective Clusters

The mean RSSI values for each cluster from the best 10 APs are represented in Figure 3.11. For a specific AP, it can be seen that the mean RSSI value for each cluster differed considerably, which enhances the characteristic of each cluster that makes it easy to identify. Thus, a combination of cluster features such as, mean and standard deviation of the data points could be useful for identifying important APs for the clusters.

3.2.1.4 Performance Analysis of Location Estimation

To analyze how accurately users' locations could be predicted when a localization system is trained on the clustered dataset obtained using RBIC, five supervised ML classifiers

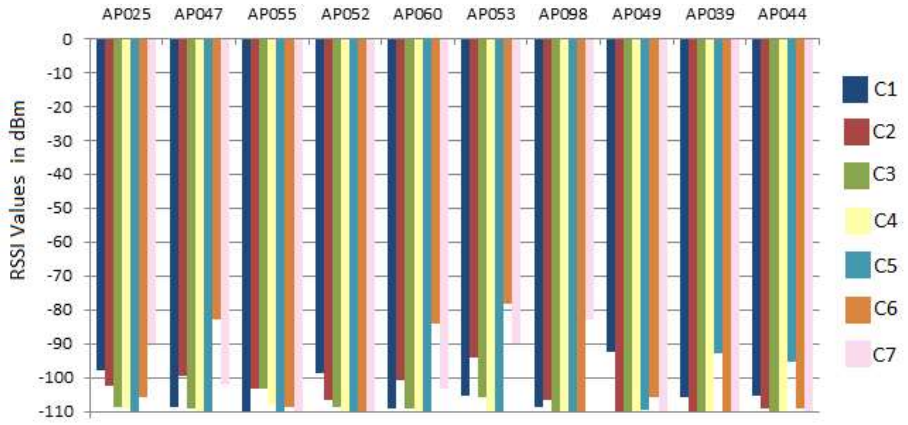


Figure 3.11: Mean RSSI Value Received From Each of the Best 10 APs for Each Cluster

have been used as shown in Figure 3.12. Here, the clusters are marked as class labels as they indicate specific zones in the indoor area as indicated in the previous subsection (Section 3.2.2.3). A portion of the floor data is treated as the test data for evaluation. As we can observe, the lowest accuracy obtained for logistic regression is 94%, which is quite acceptable, and could be increased as high as 99% using decision tree or random forest.

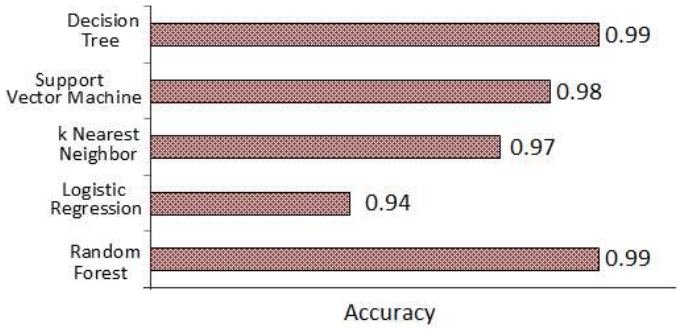


Figure 3.12: Location Estimation on Dataset_1 by Supervised Classifiers Trained on the Clustered Dataset Obtained Using the Proposed RBIC Algorithm

3.2.1.5 Comparison with State of-the-art Method K-DBSCAN

Researchers in [71] proposed a clustering algorithm which initially divides the entire data into given number of clusters with K-Means++, and then further divides each region with DBSCAN to subcluster the data points having common characteristics. After these two steps, distances between subclusters from different clusters are checked, if found lesser than the distance considered in DBSCAN, they are merged. We have applied this algorithm on

JUIndoorLoc and result is represented in Figure 3.13.

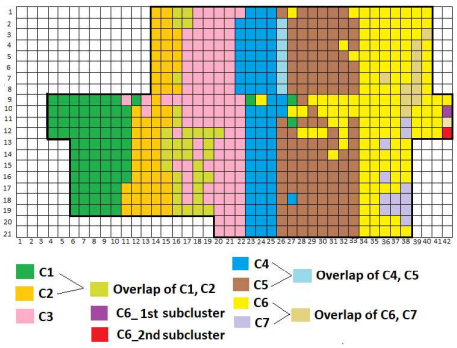


Figure 3.13: K-DBSCAN's Result Plotted on the Dataset_1 Floorplan

We can observe in Figure 3.13 that many data points of one cluster are scattered over some other clusters, degrading the overall performance. The primary reason is that in this method noises or outliers are ignored, which degrades the quality of the clusters. Also, most of the clusters did not break into subclusters with DBSCAN for a Euclidean distance parameter up to 0.5. The generated subclusters are very small and contain only one location point. Comparatively, the result of RBIC (Figure 3.9) is much neat than the result of K-DBSCAN, since it considers every base clustering algorithm's decision for each and every data point.

3.2.2 Experimentation on Dataset_2

In the case of Dataset_2 [77], data was collected by autonomous robots over the experimental region for which the floorplan is provided.

3.2.2.1 Experimental Setup

On the $21m \times 16m$ experimental region, data was reported to be collected over 345 location points following a path, identified by x-y coordinates. Six APs were reported as fixed WiFi sources by the authors [77], among which five APs have dual MAC address, for dual frequencies, 2.4 GHz and 5 GHz. Another AP operates only on 2.4 GHz frequency. Thus, total eleven features were provided by eleven MAC addresses, to form the feature set. Two devices were used to collect data, Nexus 4 and Nexus 5. In this work, the database of Nexus 5 has been used for experimentation.

3.2.2.2 Performance Analysis of Individual Clustering Algorithm

Initially, four clustering algorithms were applied on this dataset, among which K-means inidicated the optimal number of clusters to be 3, as represented in Figure 3.14. Before finalizing the set of clusters, performances of all four clustering algorithms were checked with Silhouette coefficient, as shown in Figure 3.15.

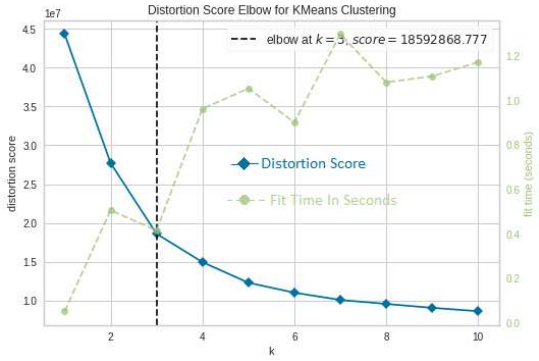


Figure 3.14: Elbow Method to Find an Optimal value of K for K-means Clustering Using Dataset_2

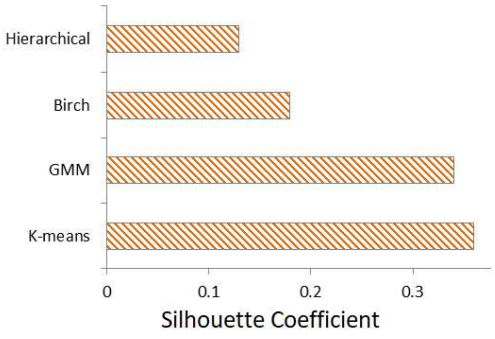


Figure 3.15: Rank of Base Clustering Algorithms Before Execution of Proposed RBIC on Dataset_2

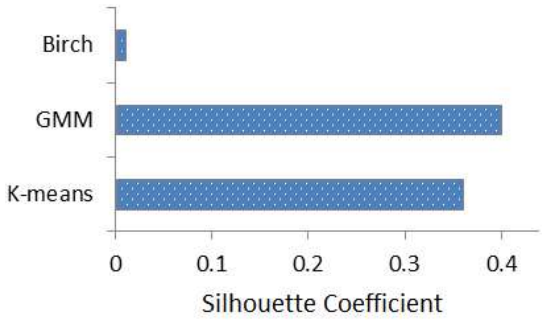


Figure 3.16: Rank of Base Clustering Algorithms After 1st Iteration of Proposed RBIC on Dataset_2

We can observe that K-means and GMM perform comparitively well for n=3 on this dataset, whereas Birch and Hierarchical clustering perform moderately. The score obtained for Birch and Hierarchical clustering are 0.18 and 0.14 respectively. Since performance of these two algorithms are close, we selected the better one among them for better accuracy. Since no of clusters is less, here we selected the outcome of the 3 best performing baseline clustering algorithms (K-means, GMM and Birch) to be fed as input to the RBIC algorithm.

After the 1^{st} iteration of RBIC algorithm, 1^{st} cluster was obtained, and the data points

of this cluster were eliminated from the whole dataset. Then, performance of these three clustering algorithms were checked again on the reduced dataset. Interestingly, the rank of the algorithms are found to be changed. GMM achieves 1^{st} rank by obtaining maximum score, while K-means achieves 2^{nd} rank. Birch algorithm maintains its 3^{rd} rank with a reduced yet acceptable (since positive) score. Hence, in the 2^{nd} iteration, the algorithms are executed following this altered ranking as shown in Figure 3.16.

3.2.2.3 Performance Analysis of Proposed RBIC Algorithm

The final outcome of the RBIC algorithm was plotted on the floorplan. In Figure 3.17, we can observe that, almost all of the data points are covered by the three clusters. Data points of same cluster are located in consecutive locations, and no data point of same cluster are present in different clusters, or in a far apart location. This visualization implies the success of the proposed approach on this second dataset.

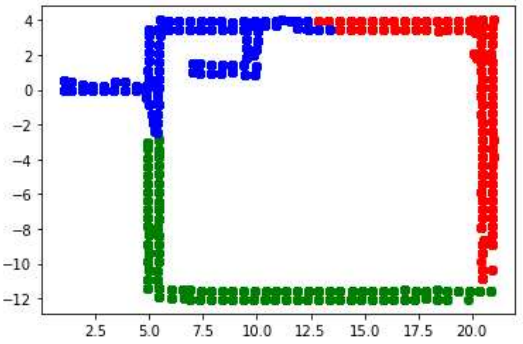


Figure 3.17: Proposed RBIC Algorithm's Result Plotted on Floorplan of Dataset_2

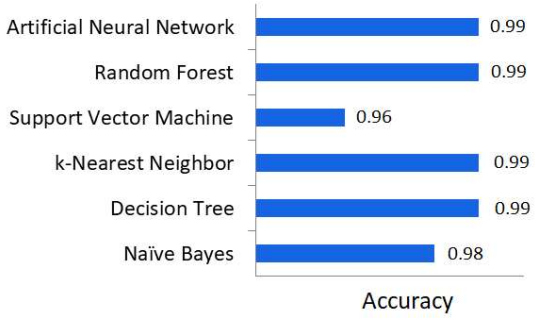


Figure 3.18: Location Estimation on Dataset_2 by Supervised Classifiers Trained on the Clustered Dataset Obtained from Proposed RBIC Algorithm

3.2.2.4 Performance Analysis of Location Estimation

To evaluate the process of location estimation, six supervised classifiers were trained using the clustered dataset, treating the cluster ids as class labels. Localization accuracy was obtained minimum of 96% for SVM, 98% for Naive Bayes, and maximum 99% for all of the other classifiers (Figure 3.18). Hence, user's position on the floor can be estimated easily by a localization system trained on the clustered dataset, obtained as the outcome of RBIC algorithm.

The performance of an indoor localization system is improved in this work using the Rank-Based Iterative Clustering (RBIC) technique. The likelihood of including two or more data samples from distinct physical locations in the same cluster is minimized by taking into account the combined decisions of various clustering algorithms, rank-wise (based on internal validation index).

3.2.3 Experimentation on Dataset_3

To strengthen the generalization of the proposed RBIC method, it was applied to another dataset. In the case of Dataset_3 [78], data was collected near two sides (south and east) of the floor. Among four base clustering algorithms, K-Means was found to perform best for K=4 (Figure 3.19). K-Means, Hierarchical, and GMM all produced Silhouette scores greater than 0.2, whereas Birch clustering produced a Silhouette score less than 0.2 as shown in Figure 3.20. Based on this observation, we choose K-Means, Hierarchical and GMM to contribute to RBIC.

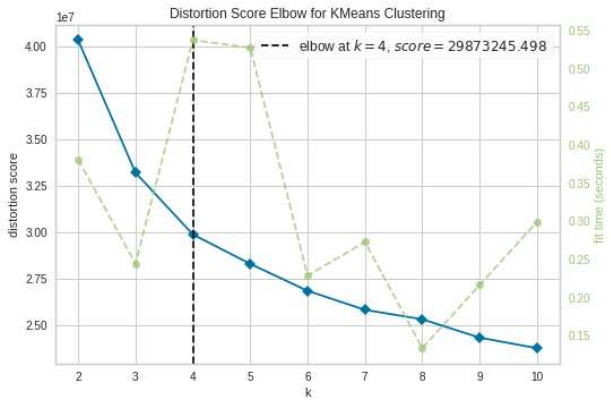


Figure 3.19: Elbow Method to Find Optimal value of K for K-means Clustering Using Dataset_3

In Figure 3.21, the RBIC algorithm's result is plotted according to coordinates of the floorplan. Data were neatly clustered into four clusters, without any overlapping. When this clustered dataset is used for localization using the cluster ids as location labels, accuracy was obtained within a range of 95% to 98% for the supervised classifiers, as shown in Figure 3.22.

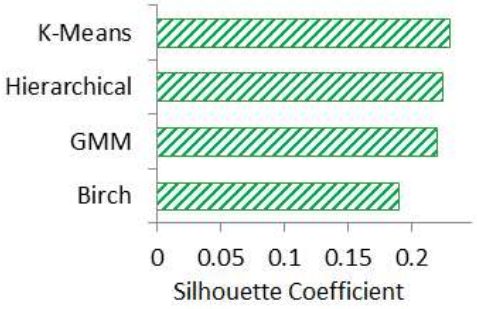


Figure 3.20: Rank of Base Clustering Algorithms Before Execution of Proposed RBIC on Dataset_3

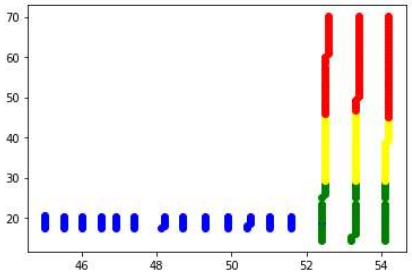


Figure 3.21: Proposed RBIC Algorithm's Result Plotted on Floorplan of Dataset_3

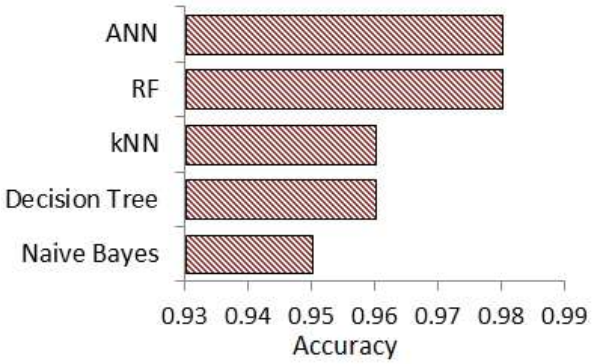


Figure 3.22: Location Estimation on Dataset_3 by Supervised Classifiers Trained on the Clustered Dataset Obtained from the Proposed RBIC Algorithm

3.3 Summary

In this chapter, a two phase training pipeline for indoor localization is proposed for grossly labeled fingerprint datasets. A Rank-Based Iterative Clustering (RBIC) method is proposed to improve the clustering solution for indoor localization that retains generality to be applied on different localization datasets. The decision of different clustering algorithms are combined, and that too priority wise (based on the rank of individual algorithms), which minimizes the chance of including two or more data samples in same cluster which are physically located far apart. In each iteration, the cluster containing the maximum number of data points at that moment is obtained and eliminated from the whole dataset, and the reduced dataset is fed to the next iteration. In this way, the computational cost is reduced with each iteration. Finally, all clusters obtained from the proposed RBIC method have been plotted on their respective floorplans for each of the three datasets to visually check the feasibility of the results.

An indoor localization system was trained using this clustered dataset including cluster ids as location labels. Here, the locations indicated by the cluster ids have been found out using the floormap and the gross labels present in the dataset. This mechanism would be able to work satisfactorily in presence of labeling errors and outliers as the cluster formation handles any individual bias exhibited by any of the baseline clustering algorithms. The technique of estimating the unknown locations was evaluated using ML classifiers. The experimentation is carried out on three publicly available benchmark datasets. Accuracy was within the range of 94% to 99% when tested using test data samples kept separate from the beginning, which is adequate for real-time location prediction systems. An analysis based on statistical measurement was presented to identify the important APs and their impacts on identifying each cluster.

The limitation of the work is that the clusters obtained applying RBIC are moderate to large in size, hence only a coarse-grained localization is possible using this proposed two-phased localization approach. This research gap is investigated and addressed in the next chapter, where a solution is proposed to obtain user's precise location within each sub-region (cluster) on the floor.

Published Work

Manjarini Mallik, Sanchita Das, Chandreyee Chowdhury, Rank Based Iterative Clustering (RBIC) for indoor localization, Engineering Applications of Artificial Intelligence, Volume 121, 2023, 106061, ISSN 0952-1976, doi:10.1016/j.engappai.2023.106061

Chapter 4

A novel signal-to-image encoding algorithm for lightweight CNN-based indoor localization

Reusing the existing WiFi infrastructure for indoor localization services is crucial for urban areas where satellite signals are not available. Machine learning approaches are commonly applied on signal fingerprints for localization. However, the performance is often hampered by the WiFi signal fluctuations due to multiple reasons. Existing works have used various smoothing and outlier removal techniques to overcome signal fluctuation effects, but the question is how to ensure that such fluctuation patterns are really outliers? Especially, when an entire system is dependent on a signal type for which fluctuation is a very common issue. In this paper, we have investigated this challenge and proposed an approach to deal with signal fluctuations without eliminating any instances as outliers. Our contribution is three-fold. (i) An algorithm, namely, Image Encoding using RSSI Fusion (IERF), is proposed to generate RSSI images that emphasize the location-specific consistent RSSI glyph and de-emphasize the less frequent RSSI patterns. (ii) A lightweight CNN model CurveNet is proposed to predict locations classifying these encoded images. (iii)To make the entire localization system resource-friendly, localization is done in two phases-coarse-grained and fine-grained. At phase-1, a sub-region is identified in which a user is present and in phase-2, the user's fine-grained location within that sub-region

is estimated. For coarse-grained localization requirements where fine-grained location labels are not available, only phase-1 would be required. We have experimented on two benchmark datasets – (i)SODIndoorLoc and (ii)Nexus-5 datasets.

4.1 Two-phase localization using encoded image fingerprints and lightweight CNN

In this section, the proposed image enconding algorithm IERF is presented in detail along with data preprocessing techniques applied. Then, the proposed two phase localization strategy along with the CNN architecture CurveNet is discussed in detail.

4.1.1 Data Preprocessing

WiFi fingerprint datasets for Indoor localization consist of RSSI vectors received from nearby APs at a given location point. APs are the features and location points are the class label generally denoted as (x, y). We have considered two fine-grained benchmark datasets in this work - (i) SODIndoorLoc Dataset and (ii) Nexus-5 dataset. In both the datasets, data are collected from consecutive coordinates, following a path around the floor, and labeled with (x,y) coordinates. To make the annotation process suitable for classification, we have annotated each unique combination of (x,y) with a unique integer label. Using exploratory data analysis (EDA), it is observed that in Nexus-5 dataset the absence of RSSI signal at any location is represented with a minimum value of -100, whereas in SODIndoorLoc dataset it is represented by a positive dummy value of 100. However, instead of representing absence of signal strength, a positive value like 100 logically indicates an impractically strong signal. In practice, RSSI values nearer to zero represents strong signal and a negative value in the range -80 to -100 dBm usually represents weak signal. For this reason, the missing values have been updated from 100 to -100 for a realistic interpretation in SODIndoorLoc dataset.

4.1.2 Formation of RSSI Fingerprint glyphs

A glyph is a visual symbol that represents a character, part of a character, or symbolic mark in writing systems. In machine learning, a glyph dataset typically refers to a collection of images of individual characters or symbols. For example, MNIST [100] dataset contains grayscale images of handwritten digits, where each class represents a specific digit between 0 to 9, and each instance contains a handwritten image of that specific digit. Since the handwriting style of different people are different, there will be minor to major variations in the glyph patterns for each digit, but there will be some specific anchor points that hold the digit-specific glyph pattern.

In WiFi fingerprinting method, if we consider its 2D representation taking RSSI magnitudes in y-axis and APs in x-axis, and draw a continuous lineplot, we will get location specific glyphs. For each AP, there is one anchor point which is its absolute RSSI value. For multiple fingerprints recorded from same location, these anchor points y-value can vary up to a certain extent, like 30 to 40 (for RSSI -30 dBm to -40 dBm), but x-value of these anchor points are always fixed. These anchor points hold the entire pattern, and since y-value can vary, the location specific glyphs also vary. Since RSSI received from an AP can fluctuate over time, different RSSI vectors collected at different timestamps for one specific location results in variation of the glyph pattern. One such image for each dataset is shown in Figure 4.1 and 4.3. In fact when data are collected from different devices or for different ambient conditions, the resulting RSSI glyph would also vary. Interestingly, an AP that imparts strong signal is less likely to environmental fluctuations but more sensitive to distance unlike an AP that imparts weak RSSI signal at an area. Thus, some portion of the plot that exhibits strong RSSI is likely to show a stable pattern across various ambient conditions.

Through the proposed algorithm Image Encoding using RSSI Fusion (IERF), presented in Algorithm 2, the different plots obtained from a location are superimposed in various combinations. Thus, the different training conditions have been encoded in the data itself. Such images exhibit some consistent portions on which classifiers could rely on for a number of different ambient conditions. To figure out the location-specific consistent pattern, combination is used to obtain various superimposed RSSI glyphs. For each location, from

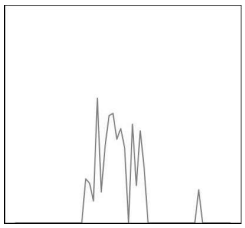


Figure 4.1: Image obtained from one RSSI vector for one specific location of SODIndoorLoc

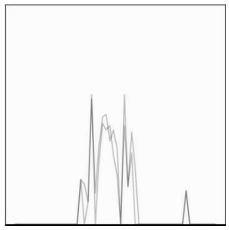


Figure 4.2: Image obtained by mathematical combination of six RSSI vectors for one specific location of SODIndoorLoc

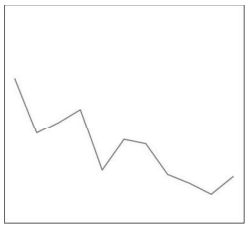


Figure 4.3: Image obtained from one RSSI vector for one specific location of Nexus-5 dataset

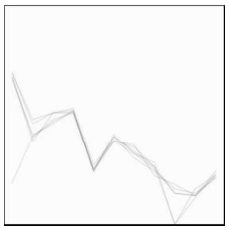


Figure 4.4: Image obtained by mathematical combination of six RSSI vectors for one specific location of Nexus-5 dataset

total n images, r images are selected at a time and an image is generated as output (steps 29-34 of Algorithm 2), which is the fusion of the selected images.

Frequently received RSSI values (direct path) from each AP makes a bold and prominent visibility in the corresponding area of the plot, whereas RSSI values from multi-path effect often leaves a feeble mark on it. Figure 4.2 and 4.4 represent one sample of generated images of SODIndoorLoc and Nexus-5 dataset respectively, after applying combination on selected r samples. In Figure 4.1 and Figure 4.3, a symmetric intensity all over the glyphs could be observed. However, in Figure 4.2 and 4.4, the glyphs are at some points bold and prominent indicating consistent RSSI values for different context, while at other points shaded and hazy capturing the context heterogeneity.

So, from data collection perspectives, r represents the no of RSSI samples required to form an image. Thus, it is also an indicator of the time interval a user is expected to stand at a location point during which test set samples should be collected. For a small value or very large value of r, frequent pattern of fluctuation could go overlooked. On the contrary, if we take a large no of all possible signal combinations then the total number of images will be huge after combination incurring high computational overhead. To address this challenge, a subset of the total ${}^{n}C_{r}$ images are selected, ensuring uniform contribution of

all samples. Initially, r consecutive samples are selected in such a way that, the 1^{st} image is generated using 1^{st} to r^{th} sample, 2^{nd} image is generated using 2^{nd} to $(r+1)^{th}$ sample and so on. Thus, initially time interval between selected samples is $d = 1$. After that, samples from larger time interval are also considered to capture unpredictable fluctuations due to ambient conditions. In each iteration, d is increased by 1 and can be increased up to $d = r - 1$. Following this computation, if required number of images are generated, then encoding method ends at that point. Otherwise in next phase, a random set of sample numbers are selected maintaining some difference in timestamp, and all possible combinations of r samples are obtained from that set. Superimposing those r samples one single image is generated at a time. Until required number of images are obtained, this process is repeated.

4.1.3 Two-Phase Localization

In fine-grained indoor localization system, usually the locations are close enough, like 1×1 square meter apart or 2×2 square feet apart. Thus, the number of classes (locations) is more than usual image datasets. Interestingly, fingerprint patterns of two consecutive locations could also be similar. Thus, location classification of the transformed image dataset would require a complex model. As a solution in this work location prediction task has been divided into two phases-phase 1 deals with gross localization to a sub-region while fine-grained localization within a sub-region is predicted in phase 2. In phase 1, the number of sub-regions is decided using proposed sub-region detection algorithm - represented as Algorithm 3.

The numeric RSSI features are fed as input to the k-means algorithm without fine-grained location label. Number of sub-regions is obtained from k-Means elbow method. Based on the RSSI characteristic similarities, data instances are grouped into clusters. For each cluster, its centroid is now mapped to its location label on the floor plan. This is feasible as all the RSSI vectors of the original dataset are labeled. Then, from the numeric RSSI dataset, the fine-grained physical locations are grouped to form individual sub-regions in such a way that each sub-region covers one unique cluster centroid. Thus, data samples per location point would be mapped to a sub-region of its nearby centroid

location.

Algorithm 2: IERF: Image Encoding using RSSI Fusion

Input: Complete RSSI fingerprint data $\boldsymbol{F}$
List of class labels $\boldsymbol{L}$
Total No. of samples per class $\boldsymbol{n}$
No. of RSSI samples to form one image $\boldsymbol{r}$
No. of images expected per class $\boldsymbol{m}$
Output: 2D Images of RSSI fingerprint data

1: **for** each class label $\boldsymbol{l}$ in $\boldsymbol{L}$
2: **for** each RSSI sample $\boldsymbol{s}$ assigned to label $\boldsymbol{l}$ in $\boldsymbol{F}$
3: **img**←plot of $\boldsymbol{s}$ with APs in x-axis, RSSI in y-axis
4: Annotate **img** with its label and sample count
5: Save **img**
6: **for** each class label $\boldsymbol{l}$ in $\boldsymbol{L}$
7: **count**←0
8: *//slide a window of length $\boldsymbol{r}$ over $\boldsymbol{F}$ to select $\boldsymbol{r}$ consecutive samples*
9: **for** $\boldsymbol{d}$ in range 1 to $\boldsymbol{r}$
10: **for** $\boldsymbol{i}$ in range 1 to ($\boldsymbol{n}$-(($\boldsymbol{r}$-1)×d+1)) *//new set of $\boldsymbol{r}$ samples*
11: Initialize empty matrix ***M_img***
12: **for** $\boldsymbol{s}$ in [$\boldsymbol{i}$, $\boldsymbol{i}$+$\boldsymbol{d}$, $\boldsymbol{i}$+2*$\boldsymbol{d}$, ... , $\boldsymbol{i}$+$\boldsymbol{r}$*$\boldsymbol{d}$]
13: ***s_img***←image encoded for the RSSI sample $\boldsymbol{s}$
14: ***s_img_arr***← matrix form of ***s_img***
15: ***M_img***← Superimpose (***M_img***, ***s_img_arr***)
16: ***final_img***←2D image of ***M_img***
17: Increment **count** by one
18: **New_label**← concatenate class label $\boldsymbol{l}$ and **count**
19: Annotate ***final_img*** with **New_label**
20: Save ***final_img***
21: **if count**=$\boldsymbol{m}$
22: **break**
23: **if count**=$\boldsymbol{m}$
24: **break**
25: **while count**<$\boldsymbol{m}$ //amount of encoded images is not enough
26: **S**← set of random non-consecutive samples of class $\boldsymbol{l}$
27: **k**← number of elements in **S**
28: **for** each combination $\boldsymbol{c}$ in ${}^{k}C_{r}$
29: Initialize empty matrix ***M_img***
30: **for** each sample $\boldsymbol{s}$ in $\boldsymbol{c}$
31: ***s_img***←image encoded for the RSSI sample $\boldsymbol{s}$
32: ***s_img_arr***← matrix form of ***s_img***
33: ***M_img***← Superimpose (***M_img***, ***s_img_arr***)
34: ***final_img***←2D image of ***M_img***
35: Increment **count** by one
36: **New_label**← concatenate class label $\boldsymbol{l}$ and **count**
37: Annotate ***final_img*** with **New_label**
38: Save ***final_img***

The sub-regions would not be uniformly placed over the experimental region as the corresponding AP placements are not uniform. However, the no of samples of two physically neighboring sub-regions would be somewhat balanced as per step 15-17 of Algorithm 3. Ties will be resolved randomly or based on the domain knowledge of the floor plan. This algorithm would be executed during the training phase only. Thus, in the online

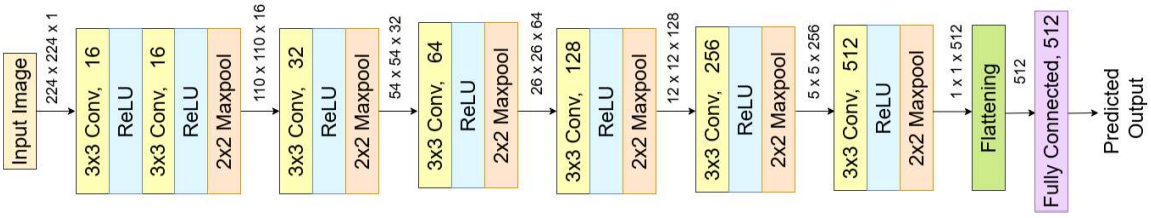


Figure 4.5: Architecture of the proposed CurveNet model

phase, phase-1 would classify the sub-region in which a user is currently present. For grossly labeled indoor localization datasets this phase would do the location prediction. For the datasets with fine-grained location labels, phase-2 would further classify the precise location of the user.

In phase-2, user's fine grained location within the predicted sub-region is estimated. For classification a lightweight CNN model is proposed, presented as follows.

4.1.4 CurveNet Model Architectures

To classify the encoded images, a lightweight CNN architecture is proposed. It contains lesser number of filters in the initial layers and more number of filters in deeper layers. The detailed architecture is represented in Figure 4.5. The kernel size is maintained as 3×3 in all convolutional layers to capture the fine-grained details of the superimposed RSS glyphs, followed by a 2×2 MaxPooling. After input layer, two consecutive convolutional layers were used instead of convolution-pooling pair, to initially capture the abstract features maintaining the high spatial resolution of the feature map. After that, the spatial resolution was decreased gradually by pooling in the following layers. A dropout rate of 0.5 was included to the final layer to reduce model complexity and generalize the model avoiding overfitting.

Algorithm 3: Sub-region detection algorithm

Input: Complete RSSI fingerprint data $\boldsymbol{F}$

Output: Sub-regions formed from $\boldsymbol{F}$

Identify number of sub-regions using K-Means elbow method
Obtain centroid of each sub-region
Map each centroid to a location class using Euclidean distance
C_loc ← list of centroid locations sorted in x or y or z axis
Loc ← list of all location classes sorted in same axis as **C_loc**
th ← predefined threshold distance
n ← maximum number of locations in a single sub-region
for each centroid location $\boldsymbol{c}$ in **C_loc**
    count←0
    Initialize empty set **S** of locations for a new sub-region
    for each location class $\boldsymbol{l}$ in **Loc**
        if $\boldsymbol{l} \neq \boldsymbol{c}$ and $\boldsymbol{l} \in$ **C_loc**
            break
        e ← Euclidean distance between $\boldsymbol{c}$ and $\boldsymbol{l}$
        If e $<$ **th** and **count** $<$ **n**
            Include $\boldsymbol{l}$ to **S**
            count ← **count** + 1
            Remove $\boldsymbol{l}$ from **Loc**

4.2 Experimental Result

Experimentation has been conducted on two benchmark datasets-SODIndoorLoc [96] and Nexus-5 [95]. The datasets are collected from different countries having varying building infrastructure. Detailed experimental analysis are presented in the following subsections. For SODIndoorLoc dataset 30 and for Nexus-5 dataset 50 images per location were generated. Each image size was kept at 224×224 in order to keep parity with other image datasets containing curves or glyphs.

4.2.1 Experiment on SODIndoorLoc Dataset

As mentioned earlier, the proposed indoor localization approach executes in two phases. To predict the broad sub-region on the floor where the user is present, the entire floor of SODIndoorLoc dataset is divided into 25 sub-regions. The number of sub-regions is decided from the K-means clustering elbow analysis as shown in Figure 4.6. Considering the distortion scores for different k values, the graph changes rapidly near $k = 5$, creating an elbow shape. However, the graph still has some small bumps after that, and it becomes almost parallel to the x-axis from $k = 24$ onward. Accordingly, the entire region is divided into 25 clusters as shown in Figure 4.7. To reduce the data dimension and improve clustering efficiency, principal component analysis (PCA) is used. As observed in Figure 4.7, clusters are well separated and not overlapping on each other. However, the size of the clusters varies resulting in an asymmetrical distribution of location points all over the clusters.

To analyze this phenomenon, centroids of all clusters are plotted on the floor as represented in Figure 4.8. Although not evenly distributed, centroids are found to cover the data collection path entirely. Finally, all the fine-grained labeled data are divided into 25 coarse-grained sub-regions each covering one centroid as shown in Figure 4.9. A few clusters are relatively small, most clusters are moderate, and one cluster (last cluster in the north-east) is significantly large in size covering 66 consecutive locations.

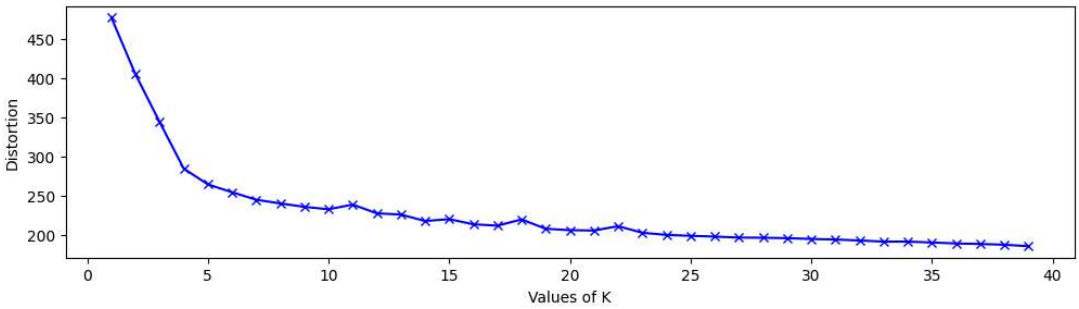


Figure 4.6: Identifying optimal value of k (number of clusters) usinh K-means elbow method on SODIndoorLoc dataset

One big cluster has two disadvantages-(i) the no of images required for fine-grained classification in phase 2 would be more that may not be resource-friendly and (ii) larger subregions result in less precise localization in Phase-1 for the coarse grained datasets where phase

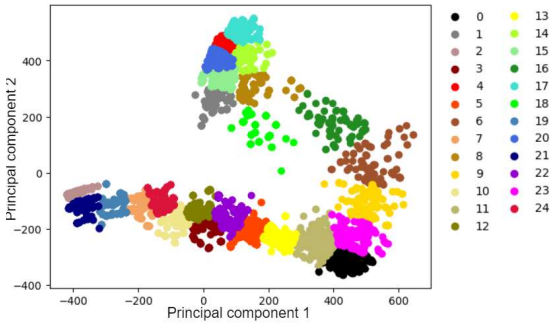


Figure 4.7: Clusters formed by K-Means on SODIndoorLoc data

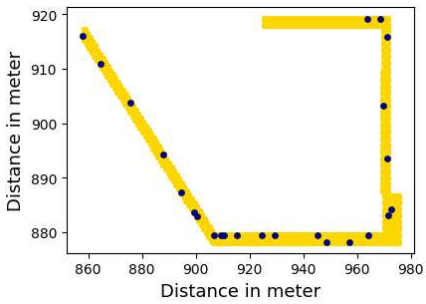


Figure 4.8: Locations of centroids of clusters formed using k-Means on SODindoorLoc dataset are plotted on the floor

2 could not be conducted. So, in this case, to keep the whole process resource-friendly, the big cluster is further divided into consecutive four clusters by applying the algorithm again now covering 16 to 17 locations each so that at a time the system need not load a huge number of images, or import a more complex CNN model.

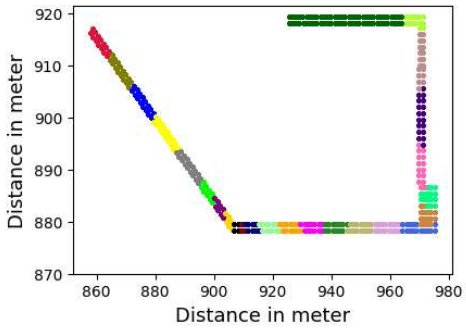


Figure 4.9: Clustered sub-regions mapped over the floor of SODIndoorLoc dataset

Phase-1 Localization

The entire dataset is first divided into two subsets-80% data are kept for training and validation while rest 20% data has been kept separate and used only for testing the trained classification model. In the 1^{st} phase of localization, the sub-region in which the user is present, is predicted.

The CurveNet model is trained and validated with 50% of images for each sub-region, maintaining a train-validation split of 80-20. In this phase, the initial image versions of the single RSSI vectors are used for classification instead of the fused RSSI images. The

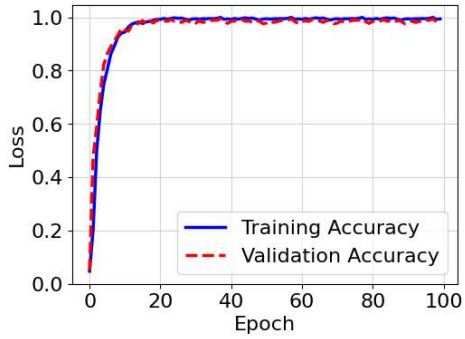


Figure 4.10: Training and validation accuracy by proposed CurveNet for sub-region prediction in SODIndoorLoc

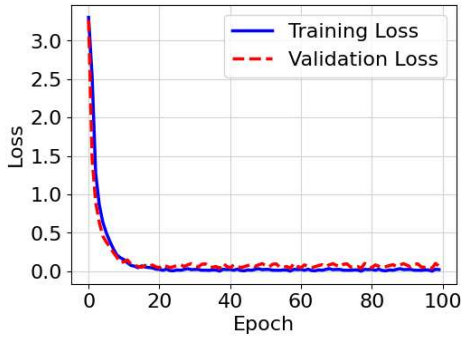


Figure 4.11: Training and validation loss by proposed CurveNet for sub-region prediction in SODIndoorLoc

reason is, in this case each sub-region is significantly large and variance in RSSI pattern between two consecutive sub-regions is very high. Thus, the effect of RSSI fluctuations among fine-grained locations within the sub-region do not degrade the classification outcome. Figure 4.10 represents the accuracy obtained by the CurveNet model during the training process. From 20 epochs onward, both the training and validation accuracies become stable and reach above 99.60%. The training and validation loss are represented in Figure 4.11. The loss becomes less than 0.01% resulting in a smooth curve parallel to the x-axis. This indicates that the model does not overfit to the data. After training the model for 100 epochs, it is saved and then loaded for testing with remaining images. An accuracy of 98.94% is obtained for sub-region detection in test case.

Phase-2 Localization

After predicting the sub-region in which the user is present, phase-2 that is, finegrained (within the cluster) localization begins. In this phase, users location is predicted precise to 1×1 square meter area. In real-time, the idea is to import the sub-region specific trained CurveNet model and predict users fine-grained location. Accordingly, the CurveNet is trained 28 times ($24 + 4$ as the 25th sub-region was farther divided into four parts).

The detailed analysis is presented in Table 4.1. As observed in Figure 4.9 and Table 4.1, the sub-regions are not equal in size, they cover different number of location points, varying from 3 to 20. Difficulty in classification process increases with increasing number of location points in the sub-region as reflected in the RAM size reported in the

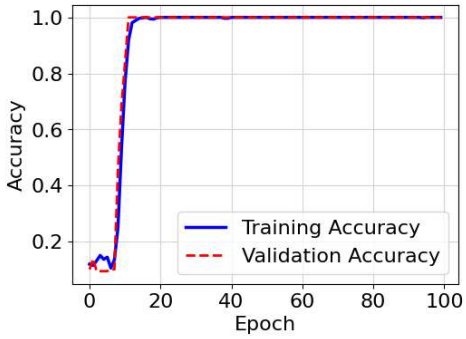


Figure 4.12: Training and validation accuracy by proposed CurveNet for fine-grained localization in SODIndoorLoc

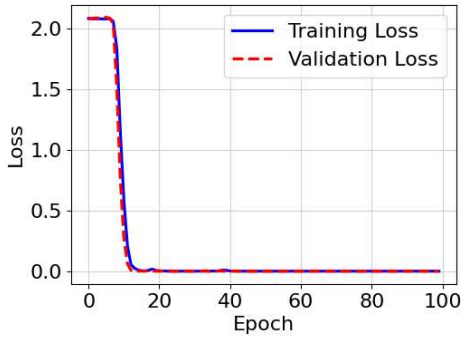


Figure 4.13: Training and validation loss by proposed CurveNet for fine-grained localization in SODIndoorLoc

table. For example, to predict user's fine-grained location in the sub-regions 8,9 and 10 covering 6,4 and 3 consecutive locations respectively 80 images per fine-grained location are sufficient. However, to do the same, 300 and 280 images per fine-grained locations were needed respectively for the larger sub-regions 22 and 23. The minimum number of fused images per fine-grained location is kept 80. The reason is, there are 30 samples per locations in the given dataset, and with a window size 6 to make fusion of each 6 consecutive RSSI samples, we get a total of 78 fused images per location. To utilize the information of the consecutive fused images fully, a minimum of 80 images are included in the CurveNet training process. However, different sub-regions of similar size may require significantly different number of images for fine-grained localization depending on the similarity of the RSSI characteristics among the fine-grained locations included. After the sub-region specific CurveNet models are trained and saved, they are fed with the test data kept separately in the beginning. For all sub-regions, test accuracy obtained is more than 99%.

Table 4.1: Fine-Grained Localization Accuracy obtained using CurveNet on SODIndoorLoc

Sub-region ID	**No. of Locations**	**Maximum RAM(GB)**	**Test Acc.(%)**
1	14	5.1	99.98
2	16	4.8	99.32
3	16	6.5	99.91
4	17	10.9	99.89
5	17	10.9	99.83
6	9	3.7	99.86
7	8	3.5	99.98
8	6	3.5	99.99
9	4	3.3	99.99
10	3	3.2	99.97
11	8	4.0	99.67
12	12	10.0	99.17
13	12	3.8	99.85
14	13	6.0	99.29
15	13	5.6	99.92
16	15	8.7	99.89
17	16	9.2	99.73
18	16	9.7	99.85
19	16	9.3	99.93
20	16	10.6	99.82
21	13	6.2	99.95
22	20	9.1	99.18
23	19	7.4	99.17
24	14	7.5	99.95
25_1	16	5.7	99.98
25_2	16	5.8	99.86
25_3	17	10.2	99.97
25_4	17	6.8	99.95

Table 4.2: Comparison of Fine-Grained Localization Accuracy obtained on Numeric Data and Encoded Image Data for SODIndoorLoc

	Numeric	RSSI	Data		Encoded Image Data
GNB	kNN	DT	LR	1D-CNN	Proposed CurveNet
92.17	93.62	94.20	81.53	93.52	$99.57^{+}_{-}0.40$

Table 4.2 represents the comparison between localization accuracy obtained on numeric RSSI signals with the localization accuracy obtained on image data encoded using proposed IERF algorithm. Four machine learning classifiers - Gaussian naive bayes (GNB), k-nearest neighbor (kNN), decision tree (DT), logistic regression (LR) and one deep learn-

ing classifier - 1D Convolutional Neural Network (CNN) are applied on the numeric RSSI signal data of SODIndoorLoc. The highest accuracy obtained by DT- 94.20%. When the encoded image data of SODIndoorLoc are classified using proposed two-phase localization approach using proposed CurveNet model, around 5% improvement in localization accuracy is observed. This experimental analysis represents the effectiveness of the entire training pipeline of signal to image encoding followed by two-phase localization.

4.2.2 Experiment on Nexus-5 Dataset

To validate the generalization of the proposed signal to image fusion approach for localization, we have considered another benchmark localization dataset, Nexus-5 dataset. To calculate the sub-regions for Nexus-5 dataset, the entire floor is divided into 22 sub-regions as decided by the proposed k-means clustering-based algorithm (Figure 4.14). The graph becomes almost parallel to the x-axis from $k = 20$ onward. After applying PCA, the obtained clusters are plotted in Figure 4.15. The clusters are well separated and non-overlapping. Compared to SODIndoorLoc dataset, in Nexus-5 dataset, the size of the clusters are somewhat close resulting in a comparatively less disparate distribution of location points over the clusters. Cluster centroids are plotted on the floor as shown in Figure 4.16. Accordingly, all the fine-grained labeled data are divided into 22 coarse-grained sub-regions each covering one centroid following the proposed algorithm as shown in Figure 4.17.

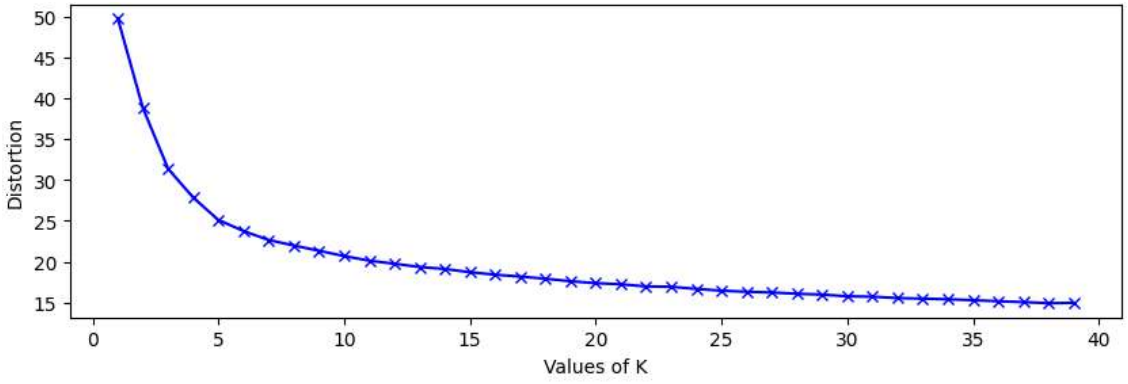


Figure 4.14: Identifying optimal value of k (number of clusters) using K-means elbow method on Nexus-5 dataset

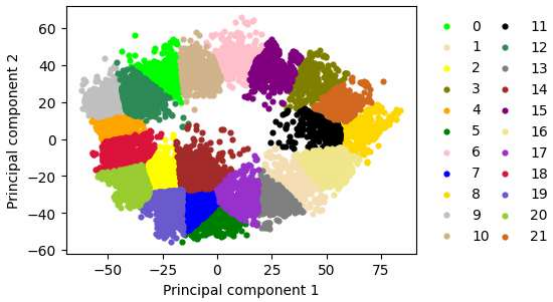


Figure 4.15: Clusters formed on Nexus-5 dataset using K-Means Clustering algorithm

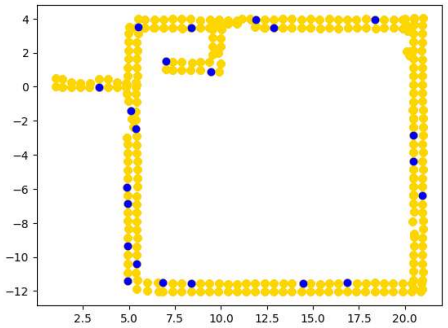


Figure 4.16: Locations of centroids of clusters formed using k-Means on Nexus-5 dataset are plotted on the floor

Phase-1 Localization

In the 1^{st} phase of localization, the user's current sub-region is predicted. Around 68% images are used to train and validate the CurveNet model and left 32% are used later for testing. The CurveNet is trained for 100 epochs. To utilize the available RAM efficiently, first 50 epochs are trained with odd number of images and next are trained with even number of images. The train-validation split was 85-15. The model starts to converge from 20 epoch onwards, as shown in Figure 4.18. To train the model, maximum RAM used is around 7.3 GB. This indicates the efficacy of the model for edge server implementations. When the trained CurveNet model is tested with rest of the images, 99.08% accuracy is obtained.

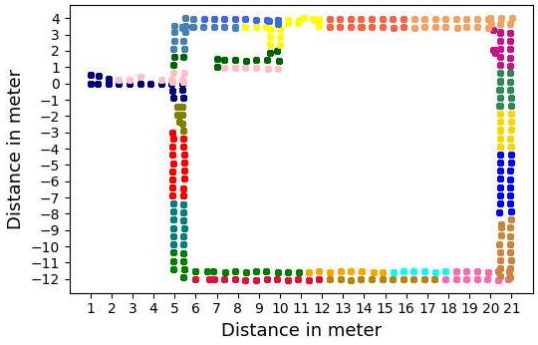


Figure 4.17: Clustered sub-regions mapped over the floor of Nexus-5 dataset

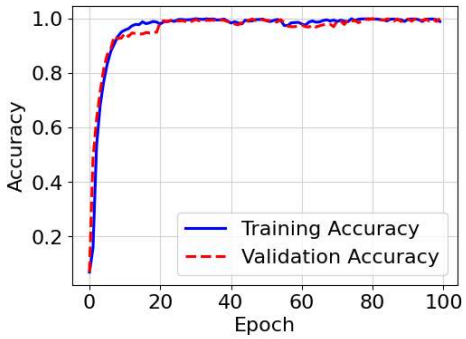


Figure 4.18: Training and validation accuracy by proposed CurveNet for sub-region prediction in Nexus-5 dataset

Phase-2 Localization

In this phase, users location is predicted between 1×1 square meter area approximately.

The CurveNet model is trained 22 times for each sub-region. Thus, sub-region specific trained model can be utilized for fine-grained localization within a sub-region. The detailed analysis is presented in Table 4.3. Compared to SODIndoorLoc dataset, larger number of images per locations are required in this case for localization. To balance the utilization of RAM, the batch size is kept as small as 4. The model is trained and validated with required amount of images then the trained model is tested with remaining images. For each sub-region, the fine-grained localization accuracy obtained was above 99%.

Figure 4.28 represents the visualizations of inter-layer outcomes obtained during classification process by CurveNet on one single sub-region of Nexus-5 dataset. In all the layers, the kernel size is fixed (3×3). Small kernel makes the model focus to concise region preserving more spatial resolution of the image. At the 1st convolutional layer, the number of filters is smallest, and with an increase in the layers of the model, the number of the filters gradually increases. This makes the model to abstract more complex and diverse features at different levels of abstraction from the different concise areas of the image. This enhances the ability of the model to capture the impact of fused signal patterns represented by varying pixel intensities. Significant improvement in spatial feature abstraction is observed from 1^{st} layer to last layer of the network.

4.2.3 Experiment on device independence based on multi-output regression

The Nexus-5 dataset was collected from the third floor of Engineering Office Wing (EOW), University of Victoria, BC, Canada [77]. From the same experimental region, data was collected with another device also, Nexus-4, on different days. However, the ground truth location coordinates of both devices do not exactly match, but they are significantly near.To experiment the performance of proposed approach in device independent situation, we have considered these devices' data and mapped it to a regression problem. For Nexus-4 there are 15 APs present in the dataset, among which 11 APs are found common in Nexus-5 dataset. We have considered those 11 common APs as features in our experiments. For Nexus-4, training and testing data are provided separately collected in different

times. For Nexus-5 dataset, we divided the entire Nexus-5 database into 80%-20% ratio for training and testing respectively in a stratified manner with respect to location points.

Single-phase localization using numeric fingerprint data

The localization performance on numeric fingerprint data in device independent scenario is experimented using six classifiers, which classify in distinct strategies. In Figure 4.19 and 4.20 we can observe that, the maximum distance using tree-based classification process (decision tree) is maximum, 15m and 19m which is quite high. Although, around 91% predictions are within 6m, for the rest 9% the localization error is a matter of concern. When ensemble of decision trees is considered, i.e. Random Forest is applied, maximum error gets reduced to 8m for Nexus-5 trained model, but remains significantly high (16m) for Nexus-4 trained model. The primary reason is, decision tree and random forest produce piece-wise constant predictions, and often face difficulties to generate smooth curves for continuous targets. The support vector machine (SVM) is used with radial basis function (rbf) kernel to capture the non-linear patterns among features, and gamma is set to 'scale' to normalize the influence of each feature automatically. These parameter tuning helped to get improved localization accuracy compared to decision tree and random forest classifiers. However, the maximum error are around 6m and 10m for Nexus-5 and Nexus-4 trained models respectively. Using the lazy-learner classifier k-nearest neighbor (kNN), the maximum errors are around 7m and 12m for Nexus-5 and Nexus-4 trained models. Euclidean distance is used for distance computation in this case. Artificial neural network (ANN) is used with adam optimizer and default learning rate 0.001, to handle noisy gradients and maintains stable weight updates. It was trained up to 1000 iterations to reach stable convergence. The error was up to 6m and 13m for Nexus-5 and Nexus-4 trained models. Finally, a one dimensional convolutional neural network (1D-CNN) model was considered, which includes 16 filters in convolution layer, with ReLu activation function and maxpooling size 2, followed by flattening and dense layer. It is trained using adam optimizer with default learning rate 0.001. The maximum errors using 1D-CNN were around 6m and 11m for Nexus-5 and Nexus-4 trained models.

The overall observation is, the models perform better when trained with Nexus-5 de-

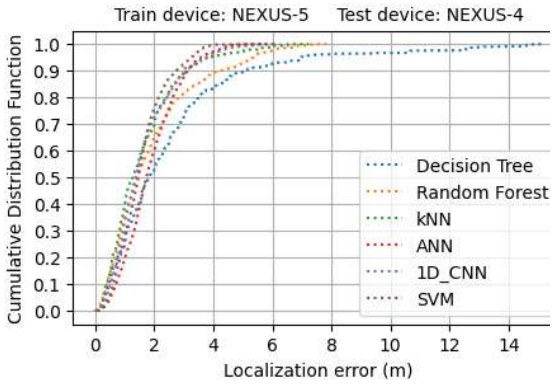


Figure 4.19: Single-phase localization trained using Nexus-5
and tested using Nexus-4

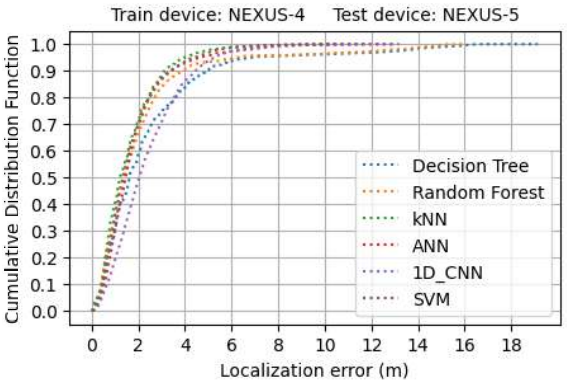


Figure 4.20: Single-phase localization trained using Nexus-4
and tested using Nexus-5

vice's data and tested with Nexus-4 device's data. One interesting fact is, Nexus-5 dataset has lower training data samples per location than Nexus-4, approximately half in count for most of the locations. The total sample count in Nexus-5 and Nexus-4 training datasets are 13817 and 30335 respectively. Thus, the chance for capturing signal fluctuations in Nexus-4 training samples are significantly high, which in turn increases the location specific intra-variance, and impacts localization performance in a negative way. Removing outliers (less frequent fingerprint patterns) using some traditional measurement techniques such as median absolute deviation or Z-score method could be one option, but since signal fluctuation is a practical and unavoidable problem in WiFi based indoor localization domain, we cannot guarantee that any of these less frequent fingerprint patterns will not return in the test case scenario. Thus, the challenge is to improve the localization performance in a balanced way without removing samples from the training datasets.

Two-phase localization in device independent scenario

The proposed localization process is a two-phase framework, where the output of phase_1 is used to activate the region-specific trained model in phase_2, presented as follows.

Phase-1 localization

To divide the entire experimental region into smaller regions, k-means clustering algorithm is applied to the training data, after removing the class labels. Both devices' data are utilized for this purpose. At first, the ideal number of clusters is identified from elbow

method, as shown in the Figure 4.21. It is observed that, up to k=5 the distortion score falls abruptly, then the rate of decrement in distortion score is reduced. Thus, 5 is chosen as the number of clusters and the entire data is divided into 5 clusters applying k-means clustering algorithm. After that, the centroids of the clusters are obtained, and their locations on the floor are mapped using the ground truth information of the dataset. After that, the entire set of ground truth locations are divided into 5 clusters in such a way that each cluster covers one centroid, near the center area of that region, as shown in Figure 4.22.

Sub-region prediction was experimented using CurveNet. From the confusion matrix shown in Figure 4.24 we can observe that, for same train-test device case, the classification accuracy is around 99.12%, and little misclassification occurred between consecutive sub-regions only. For device independent case, although the intensity of misclassification increases (Figure 4.23), the confusion is still limited among consecutive sub-regions. The fingerprints around adjoining locations of two consecutive sub-regions hold similar patterns.

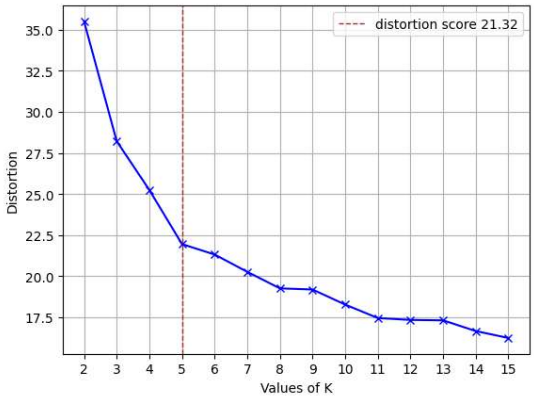


Figure 4.21: Identification of no. of sub-regions for device independent scenario using elbow method for k-means clustering

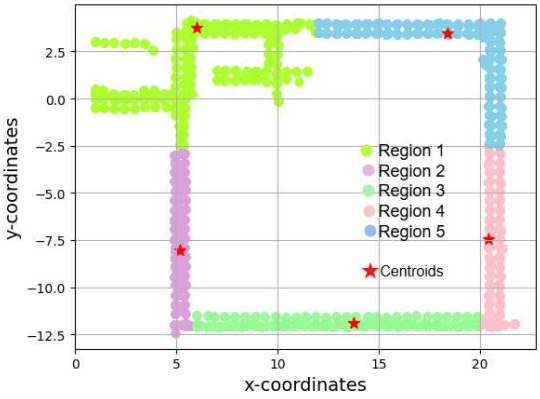


Figure 4.22: The entire floor divided into 5 regions, each centered around a cluster centroid

Phase_2 localization

After phase_1, to obtain fine-grained locations within each region, the encoded image dataset is used that has been obtained by applying the proposed IERF algorithm. For this dataset, a window of size 5 is chosen, and maximum time distance between fingerprint samples is chosen as 2 seconds. The x and y coordinate information is stored in image

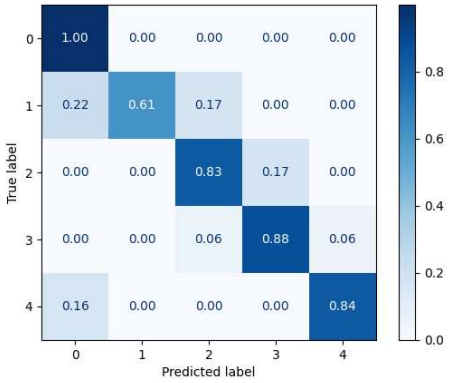


Figure 4.23: Sub-region prediction using CurveNet
train device: Nexus-5 test device: Nexus-4

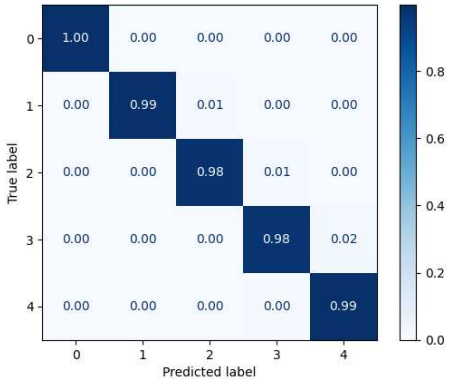


Figure 4.24: Sub-region prediction using CurveNet
train device: Nexus-5 test device: Nexus-5

name, and later extracted as labels for regression process. Two CurveNet models are trained for each clusters, one using Nexus-5 training data and another using Nexus-4 training data. Thus, there are 10 trained CurveNet models. During this process, 20% data from the training set is used for validation, and rest 80% is used to train the model. This 80-20 splits are done in stratified manner with respect to fine-grained locations for 9 among the 10 cases, except one. When CurveNet is trained using Nexus-5 data for cluster 5, this stratified split was not possible because for a few locations the number of samples was significantly smaller than other locations.

The CurveNet models are trained with a moderate batch size of 16 to maintain a balance between potentially faster convergence per epoch and avoiding noisy updates. Adam optimizer is used with default learning rate 0.001 to deal with the noisy gradients and maintains stable weight updates. The convergence of the model is ensured from the structure of its loss curve, where mean square error loss (MSE) is used as metric, presented in Figure 4.25. From 20 epoch onward the loss becomes minimum and stable. Learning performance is measured using mean absolute error (MAE), as shown in Figure 4.26, which also becomes stable from 20 epochs.

In Table 4.4, the localization performance by CurveNet in device independent scenario for each cluster is presented. It can be observed that, the maximum error is significantly reduced compared to single-phase localization using numeric RSSI data, as shown in Figure 4.19 and 4.20.

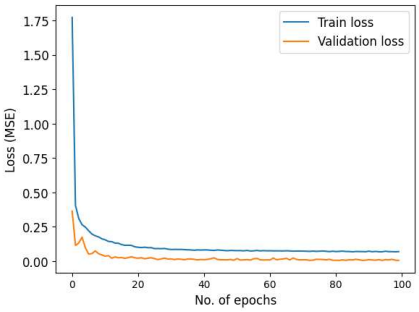


Figure 4.25: Training and validation loss for cluster 1
trained using Nexus-4 data

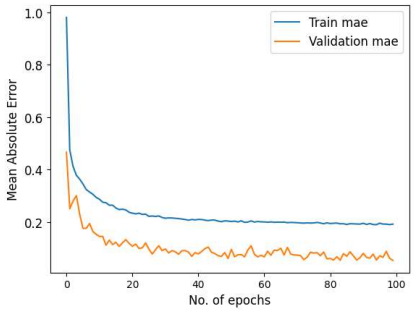


Figure 4.26: Training and validation error for cluster 1
trained using Nexus-4 data

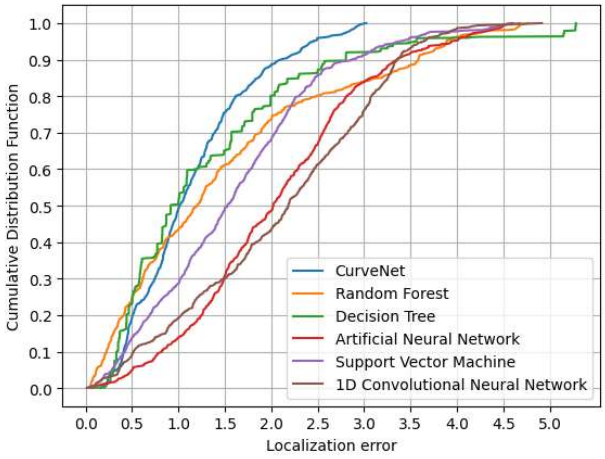


Figure 4.27: Comparison between the performance of proposed CurveNet model on image fusion data (obtained using proposed algorithm) and other classifiers on numeric RSSI data, Train device- Nexus-5, Test device- Nexus-4

The performance of CurveNet within a cluster is also compared with other classifiers' performances using numeric data, as shown in Figure 4.27. In this case, using CurveNet the maximum error is reduce by 2.5 meters and the overall localization performance is also better using CurveNet.

4.2.4 Performance of the proposed CurveNet on benchmark Glyph Datasets

To generalize the performance of our proposed CurveNet architecture, it is experimented with two standard benchmark datasets containing glyph images, as represented in Table 4.5. Both datasets contain grayscale images of handwritten digits (0-9). The Digits

Dataset [99] is smaller one, contains total 1797 samples, among which 90% images are used for training and validation, and 10% are used for testing. The MNIST dataset [100] is provided with 60000 training images and 10000 test images. Due to hardware constraints, only 1000 and 500 images per class are taken for training and validation respectively. However, the entire test set is used for testing. Classification accuracy obtained for MNIST and The Digits Dataset are 98.38% and 98.33% respectively, which is close enough to the accuracy obtained for fine-grained localization on the proposed image datasets. This result reflects the robustness and effectiveness of proposed CurveNet architecture, presenting its potential applicability to various curve-like image classification tasks.

Table 4.3: Fine-Grained Localization Accuracy Obtained using CurveNet on Nexus-5 Dataset

Sub-region ID	**No. of Locations**	**Maximum RAM(GB)**	**Test Acc.(%)**
1	13	6.1	99.92
2	11	5.3	99.95
3	12	5.5	99.95
4	18	6.7	99.95
5	8	3.1	99.99
6	6	2.5	99.91
7	18	6.8	99.95
8	12	3.7	99.90
9	21	7.1	99.85
10	17	5.2	99.76
11	7	3.6	99.91
12	17	9.8	99.95
13	15	5.7	99.90
14	15	4.38	99.97
15	16	6.60	99.87
16	19	9.8	99.74
17	13	3.63	99.87
18	24	6.5	99.90
19	24	6.6	99.92
20	22	8.2	99.84
21	24	8.8	99.71
22	11	3.3	99.88

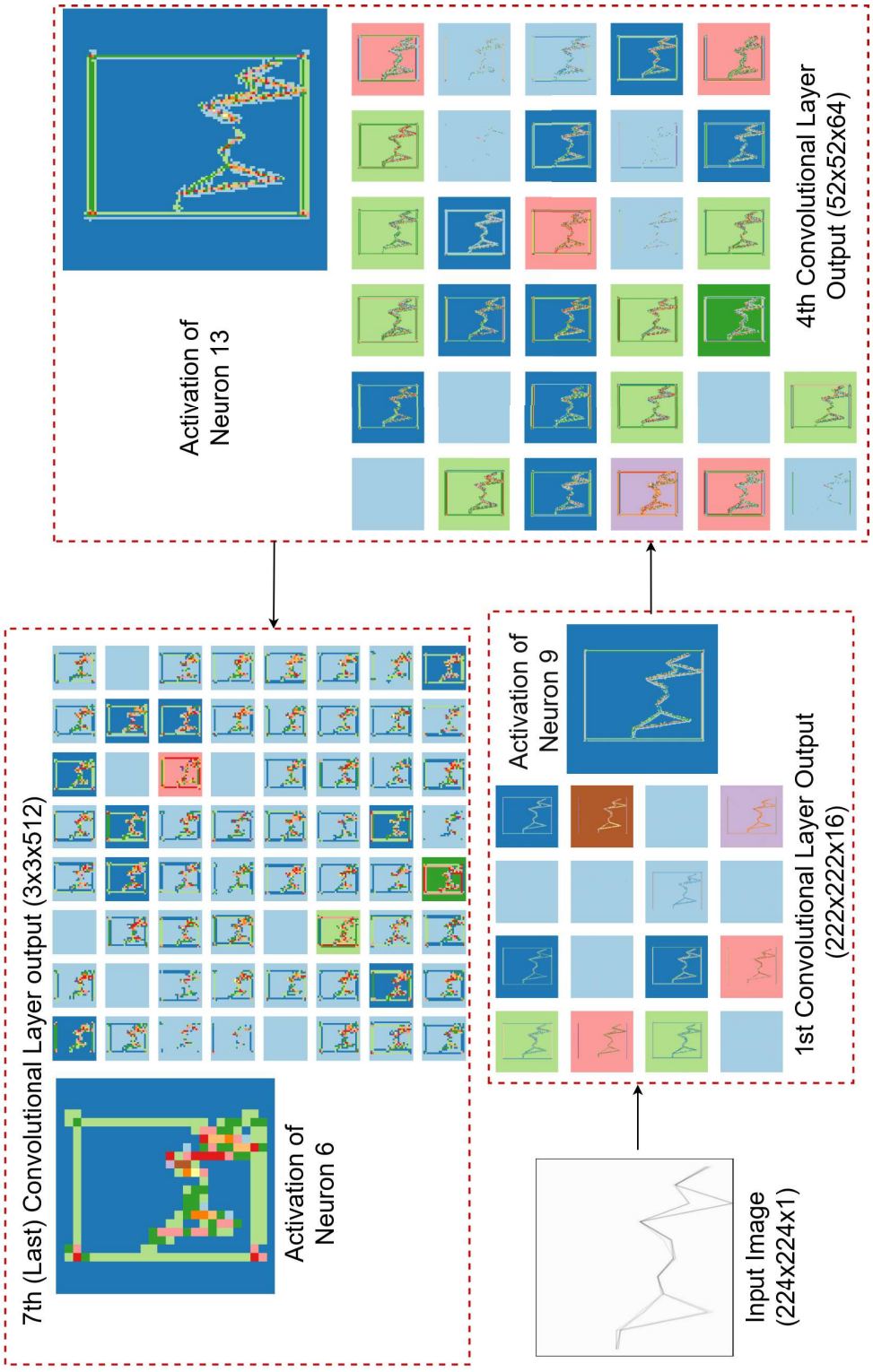


Figure 4.28: Visualization of the outputs of intermediate convolutional layers of CurveNet for a sub-region of Nexus-5 Dataset

Table 4.4: Fine-grained Localization Performance Obtained using CurveNet multi-output regression

Train device	Test device	Cluster ID	Accuracy	Max error (m)
Nexus-5	Nexus-4	1	95% predictions $\leq$3.5 m	4.5
		2	96% predictions $\leq$2.5 m	3.0
		3	89% predictions $\leq$3.2 m	4.9
		4	84% predictions $\leq$1.5 m	3.5
		5	97% predictions $\leq$2.5 m	3.4
Nexus-4	Nexus-5	1	95% predictions $\leq$3.5 m	4.2
		2	96% predictions $\leq$3.5 m	3.8
		3	95% predictions $\leq$3.2 m	4.6
		4	95% predictions $\leq$3.0 m	4.4
		5	70% predictions $\leq$3.2 m	5.2
Same device (Nexus-5)		1	97% predictions $\leq$0.51 m	2.60
		2	95% predictions $\leq$0.58 m	1.30
		3	94% predictions $\leq$0.58 m	1.38
		4	97% predictions $\leq$0.58 m	0.73
		5	91% predictions $\leq$0.98 m	2.56

Table 4.5: Comparison of classification accuracy of CurveNet on benchmark datasets

Dataset	Type	Content	Accuracy (%)
MNIST	Benchmark	Handwritten digits	98.38
The Digits Dataset	Benchmark	Handwritten digits	98.33
SODIndoorLoc image dataset	Proposed	Fusion of signal	99.57 ± 0.40
Nexus-5 image dataset	Proposed	Fusion of signal	99.85 ± 0.14

Table 4.6: Performance comparison of LeNet-5 and proposed CurveNet

Dataset	Model	#Images Train,Val	Accuracy (%)
SODIndoorLoc	LeNet-5	2400	99.92
SODIndoorLoc	CurveNet	800	99.98
Nexus-5	LeNet-5	1200	99.90
Nexus-5	CurveNet	600	99.91

4.2.5 Performance comparison of proposed CurveNet with LeNet-5

LeNet-5, a CNN variant was proposed in [101]. The primary goal of LeNet-5 is to classify

handwritten digits and letters that is, datasets containing strokes or glyphs. The CurveNet has been utilized for classifying the RSSI images containing glyphs. One sub-region from each dataset is selected to compare fine-grained localization accuracy.This experiment has two aspects. Firstly, it shows the effectiveness of the proposed image encoding algorithm as the transformed images could be classified appreciably by any CNN architecture such as, LeNet-5. Secondly, the efficacy of CurveNet is compared over LeNet. LeNet-5 is a heavier architecture (92.5 MB) than the proposed CurveNet (7.03 MB) with larger neurons and greater weight training. Minimum requirement of data is also larger for LeNet-5. As shown in Table 4.6, for both the datasets CurveNet requires smaller amount of images for training and validation. When tested, more than 99.90% accuracy obtained for both datasets. LeNet-5 also obtained impressive classification accuracy for both datasets, but due to larger architecture it requires larger number of training inputs. Thus, it can be concluded that proposed CurveNet is a lightweight model that performs as good as LeNet-5 for classifying curve like patterns, when trained with minimum amount of data.

4.3 Summary

The objective of this work is to propose a semi-supervised resource-friendly indoor localization framework to address the challenge of inconsistency in location-specific fingerprints. We propose an image encoding algorithm, IERF, to map 1D fingerprints into 2D space, reflecting the most consistent location-specific fingerprint pattern. In the proposed algorithm, mathematical combination is used to investigate the relative consistency and inconsistency present in the fingerprint pattern. A lightweight CNN model, CurveNet, is proposed to identify the broad region on the floor where the user is present and then the precise fine-grained location of the user within that broad region. This two-phase location prediction approach allows the model to be resource-friendly and suitable for real-time use. The performance of the proposed approach is evaluated using two benchmark datasets containing 1D fingerprints. With encoded images using the IERF algorithm and applying CurveNet for location prediction, around 99% accuracy is achieved, which is 5% better compared to 1D RSSI-based location prediction. The performance of the proposed CurveNet model is compared to the standard glyph-classifier model LeNet-5 us-

ing both RSSI fingerprint benchmark datasets. CurveNet is found to perform as good as the LeNet-5 model, though it is lighter and has a less complex architecture than LeNet-5. CurveNet was applied on two standard image datasets also – the MNIST and Digits datasets, containing glyphs – and standard baseline classification accuracy was achieved for both datasets.

In this chapter, we have proposed approaches to achieve the best possible localization performance in both fine-grained and coarse-grained manners. However, an important question still remains: is it always feasible to divide the entire region into fixed and symmetrical granularities, particularly in public indoor environments where access points (APs) are deployed in a non-uniform manner? This issue is further investigated in the following chapter.

Communicated Work

Mallik, M., Brahma, D., Khaskel, R. and Chowdhury, C., 2025. A resource-friendly indoor localization method using spatial feature analysis of WiFi fingerprints. Applied Soft Computing, Elsevier.

Chapter 5

GO-kDN: Granularity Optimization through k-Disagreeing Neighbor Concept

WiFi or Bluetooth based localization approaches utilize the existing infrastructure of already deployed APs as signal sources. As a result, the non-uniform placement of APs induces non-uniform coverage resulting in varying precision localization service. Existing machine learning approaches attempt to localize at a gross granularity (rooms, floors) or at a fine granularity (1mx1m) while performance in the later case would drop for some locations. However, the relation between spatial distribution of available APs and the resulting non-uniform localization by the classifiers remain mostly unexplored. Hence, an algorithm, Granularity Optimization through k-Disagreeing Neighbor (GO-kDN) is proposed that analyzes the hardness of instances and combines instance spaces of different classes to obtain a coarse-grained class, only if the characteristics of their instance spaces are worth to combine. The non-uniformity of APs is thus translated to the varying granularity of target classes. GO-kDN enables the classifiers to perform equally well on such location classes having varying granularity levels. A benchmark dataset, MetroIndoorLoc has been designed[1] that contains RSSI fingerprints from available APs collected from

[1]https://drive.google.com/drive/folders/1lCxKDRTplT1TMykPXo5ikDft9K9wMUhV?usp=sharing

two metro stations— one at grade level and another at underground level. Subsequently, CGAN has been applied to generate synthetic data for some locations in order to address class imbalance of such data collected from public areas.

5.1 Design of MetroIndoorLoc

In this section, a detailed design of MetroIndoorLoc is presented. At first, the design and implementation details of the WiFi-Scanner app are explained. After that, the data collection and preprocessing techniques used are discussed, followed by a detailed description of MetroIndoorLoc. Finally, the characteristics of the MetroIndoorLoc dataset are discussed, along with statistical analysis and visualization. Variations in signal strength at metro stations for different ambient conditions have also been examined.

5.1.1 Implementation and working principle of WiFi-Scanner app

To sense and capture WiFi signal strengths from nearby WiFi APs, the application software "WiFi-Scanner" is built. Although there are some existing applications that capture connected APs and sense signal strengths from those APs, these apps do not store this information automatically. Using these apps, the user can identify a location within a building or apartment where good signal strength is available from some nearby AP for internet connectivity. However, since the purpose of these apps is to find a better place for data transfer and not data collection for localization, no data is stored. It is not practical to manually record the data shown on the smartphone screen for every location repeatedly. Thus, "WiFi-Scanner" is built solely to collect and automatically store data for localization.

The "WiFi-Scanner" application has been built for data collection using Kivy [82], an open source library of Python. Using Kivy, it is possible to deploy apps on Linux, Windows, macOS, iOS and Android with a single codebase. We have deployed our WiFi-Scanner app on Android using the tool Buildozer [83]. The Buildozer tool eases the apk building process through setting up the prerequisites of python-for-android including

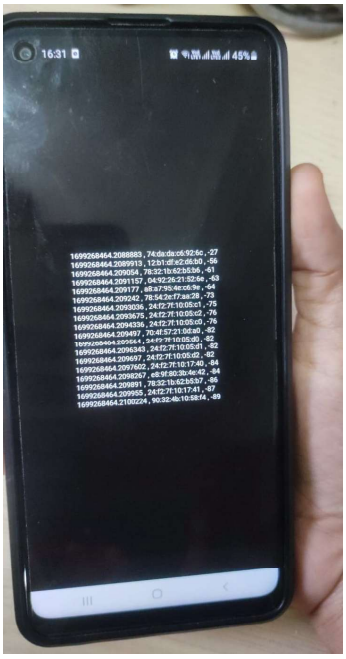


Figure 5.1: Data collection using implemented WiFi-scanner app

required versions of android NDK and SDK. WiFi and Location options must be enabled in the smartphone to make WiFi-scanner work. MetroIndoorLoc dataset has been built by collecting the data using this application. The working principle of the WiFi-Scanner

Timestamp	Mac	RSSI
1699268467	74:da:da:c6:92:6c	-27
1699268467	12:b1:df:e2:d6:b0	-56
1699268467	78:32:1b:62:b5:b6	-61
1699268467	04:92:26:21:52:6e	-63
1699268467	a8:a7:95:4e:c6:9e	-64
1699268467	78:54:2e:f7:aa:28	-73
1699268467	24:f2:7f:10:05:c1	-75
1699268467	24:f2:7f:10:05:c2	-76
1699268467	24:f2:7f:10:05:c0	-76
1699268467	70:4f:57:21:0d:a0	-82
1699268467	24:f2:7f:10:05:d0	-82
1699268467	24:f2:7f:10:05:d1	-82
1699268467	24:f2:7f:10:05:d2	-82
1699268467	24:f2:7f:10:17:40	-84
1699268467	e8:9f:80:3b:4e:42	-84
1699268467	78:32:1b:62:b5:b7	-86
1699268467	24:f2:7f:10:17:41	-87

Figure 5.2: Automatic storage format of collected data in a csv file

app is such that one needs to open the app at desired location, and keep it open as long as (s)he wants to collect data. MAC addresses of available nearby APs will be displayed on the screen along with respective RSSI values with respect to timestamps as shown in Fig.

5.1. This information will be stored in the same format automatically in the download folder of internal storage of the smartphone in a file named as "tracking.csv" as shown in Fig. 5.2. When the app is opened for the first time, "tracking.csv" file will be created automatically, and data will be appended afterwards. After collecting desired amount of data from a specific location, the file must be renamed indicating the location label, for example, "Location_1.csv". In this way, data from multiple locations can be collected and stored automatically in the device. However, the snippet of raw data shown in Fig. 5.1 and Figure 5.2 are not collected from the metro stations, because MAC addresses of the APs present at those stations are kept private. In the available datasets also, the n APs are presented as AP_1, AP_2, ..., AP_n to make those anonymous.

5.1.2 Data Collection

Data was collected by a team of volunteers using five different devices from two metro stations, one is at-grade level and another is underground metro station. The details of the devices are represented in Table 5.1. The RSSI values received at a particular location from a specific AP may vary when collected through different devices [33]. To deal with this challenge, training data is collected with various commonly available commercial smartphone devices of different configurations. When a number of RSSI patterns with device-related variations are present in the training set, a consistent representative signature pattern could be obtained.

Both stations were grided with local (x,y) coordinates as shown in Fig. 5.3 and Fig. 5.4, where each unit of x coordinates corresponds to a pillar present at the station. The experimental region of the at-grade level metro station platform was 128.44 meter long along the x axis and 5.46 meter wide along y axis. The experimental region of the underground metro station platform was 152.10 meter long along x axis and 5.16 meter wide along y axis. Both platforms were divided into 27 equal parts along the x axis, each containing a pillar, and 4 equal parts along the y axis to indicate the path for four users. As a result, each location point of at-grade level metro station covers an area of (4.75 $\times$ 1.37) square meter and each location point of the underground metro station covers (5.63

Table 5.1: Specification of smartphones used for data collection

Model name	Technology	Configuration
Poco x3 pro	GSM/HSPA/LTE, HSPA 42.2/5.76 Mbps	Android 11, upgradable to android 12,6GB RAM Qualcomm Snapdragon 860, Octa-core (1x2.96 GHz & 3x2.42 GHz & 4x1.78)
Redmi note 10 pro	GSM/HSPA/LTE, HSPA 42.2/5.76 Mbps	Android 11, upgradable to android 12, 6GB RAM Qualcomm SM7150 Snapdragon 732G, Octa-core (2x2.3 GHz & 6x1.8 GHz)
Samsung Galaxy A03 Core	GSM/HSPA/LTE, HSPA 21.1/5.76 Mbps	Android 11,2 GB RAM, Unison SC9863A, Octa-core (4x1.6 GHz & 4x1.6 GHz)
Samsung galaxy A21s	GSM/HSPA/LTE, HSPA 42.2/5.76 Mbps	Android 10, 6 GB RAM, upgradable to android 12, Exynox 850, Octa-core (4x2 GHz & 4x2 GHz)
Samsung galaxy M12	GSM/HSPA/LTE, HSPA 42.2/5.76 Mbps	Android 11, 6 GB RAM, Exynox 850, Octa-core (4x2 GHz & 4x2 GHz)

$\times$ 1.29) square meter. 77 and 63 APs were found at the at-grade level and underground metro station respectively. Collected data are analyzed with respect to APs and location points in Section 5.1.4.

5.1.3 Dataset Details

Raw data are collected in the triplet format [Timestamp, Mac, RSSI] as shown in Fig. 5.2. For each (x, y) location such a file is formed. After collecting data from all (x, y) location points, these raw data are preprocessed. Both the preprocessed datasets have the set of attributes shown in Fig. 5.5. User represents unique user id to identify four subjects. Y and X attributes represent the axis value along y-axis and x-axis respectively. Combining Y and X value, the cell_id is obtained, for example, "C_4_12" the location at $(x = 12, y = 4)$. Optimized label is a numeric label signifying its granularity level, assigned to each updated cell, obtained as a result of the proposed GO-kDN algorithm, explained in the following section. Cell_id (C_Y_X) and Optimized_label both are class attributes. How localization accuracy varies for these two classes, are analyzed in Section 5.3. AP_1 to AP_n represent n fixed AP sources.

When this dataset is fed to any ML or DL classifier algorithm, RSSI values received

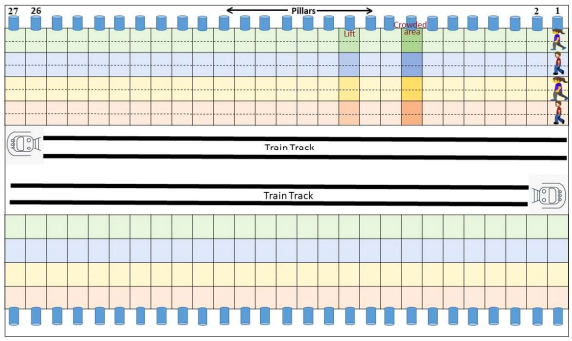


Figure 5.3: Grided structure of at-grade level metro station platform

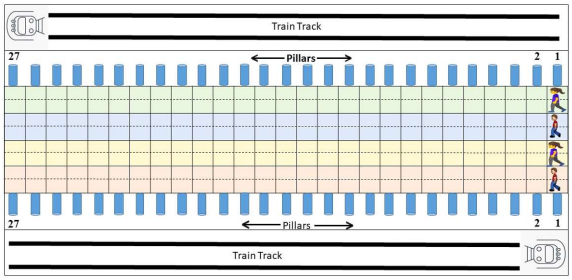


Figure 5.4: Grided structure of underground metro station platform

from all APs at some particular location form the RSSI vector i.e. the feature value vector for that location. Cell_id is the class label at the initial phase of the GO-kDN, and after that these class labels are merged according to GO-kDN algorithm's output until optimized phase is reached. Optimized label is the final class label of the dataset. Time, User, Y, X columns are dropped before classification, as these are not the features and provided only for clarification. RSSI values near zero indicate a strong signal and closer proximity of that

Timestamp	User	Y	X	Cell_id (C_Y_X)	Optimized _label	AP1 .....AP47.....APn
1654587997	U3	4	12	C_4_12	42	-75..... -110 .....-31

Figure 5.5: Attributes of preprocessed datasets

location to that AP. In both datasets, the strongest RSSI value received is -28 dBm. The weakest RSSI values received at at-grade level and underground station are -95 dBm and -92 dBm, respectively. However, at some locations, signal strength may not be received at all from some APs. Such missing values are replaced with a dummy value of -110, indicating absence of signal. There are total 1600 data samples in the underground metro station dataset that pertains to 102 different class labels (location point) and total 3031

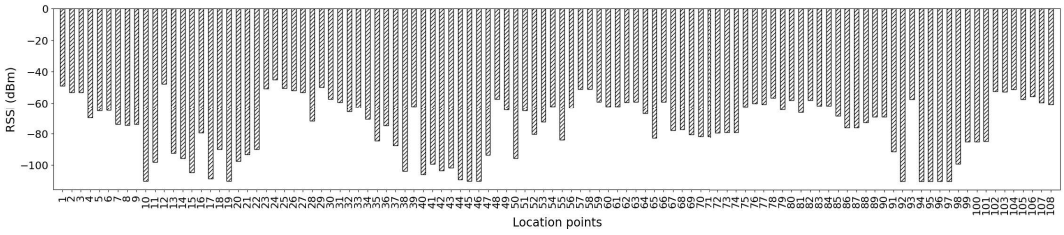


Figure 5.6: Variations of RSSI values throughout all locations of at-grade level metro station, received from a particular AP

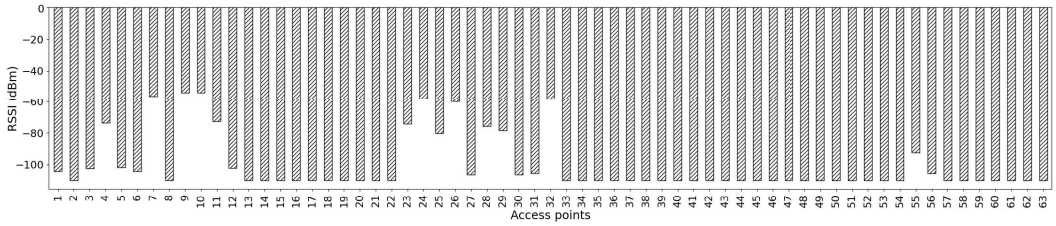


Figure 5.7: Variations of RSSI values received from all APs of underground metro station at a particular location point

data samples in the at-grade level metro station dataset that pertains to 108 different class labels (location point). There are around 16 and 20 samples per class in the underground and at-grade level metro stations respectively.

5.1.4 Discussion on MetroIndoorLoc

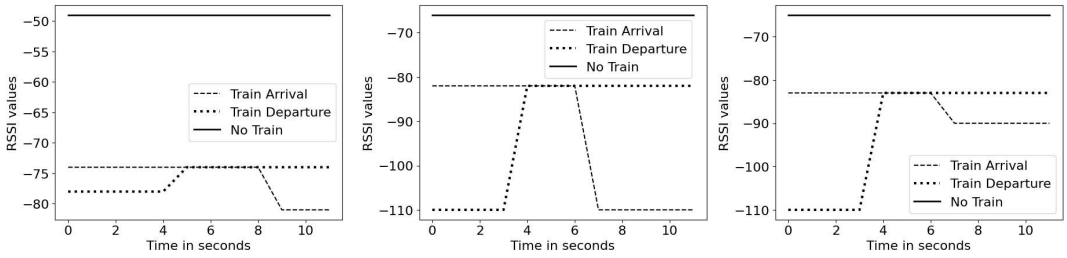


Figure 5.8: Variations of RSSI values of three different APs at one location point of At-grade level metro station with arrival, departure and absence of train

It is important to visualize the data first to look for overall patterns and understand the important aspects of it. How the RSSI signal varies across the platform for an AP is investigated first as shown in Fig. 5.6. It can be observed that, at a few location points the AP provides comparatively strong signal (-40 to -50 dBm). These location points are supposed to be closer to that AP. On the other hand, rest of the location points receive comparatively weak signal, (-75 dBm to -95 dBm) or no signal at all. This clearly shows

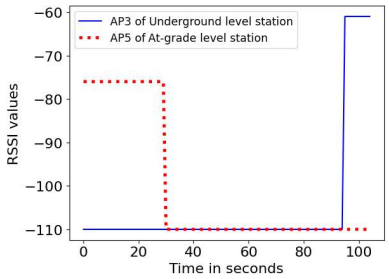


Figure 5.9: Variation of RSSI values of two APs fixed at two platforms, when user U1 is traveling on train from at-grade level metro station to underground level metro station

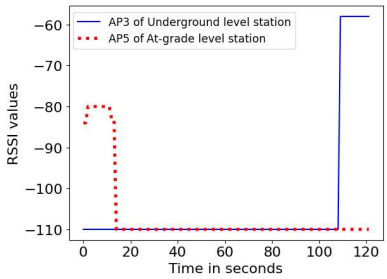


Figure 5.10: Variation of RSSI values of two APs fixed at two platforms, when user U2 is traveling on train from at-grade level metro station to underground level metro station

the distance sensitivity of the RSSI signal across the location classes. However, for two consecutive location points, the RSSI values could be similar. This in turn reduces the precision of the localization service.

Fig. 5.7 represents the variation of RSSI values received from all APs at a particular location point. It can be observed that, from a few APs (dominant APs) strong to moderate signal strength is received, and rest APs provide no signal at all. For any other location point, the set of dominant APs would differ. This is true for all the 5 devices irrespective of their different WiFi sensitivities. Thus, there is a distinguishing pattern of feature vectors for each location point that makes this data fit for classification.

Arrival and departure of train are two mostly occurring ambient conditions to study localization for metro stations. On the at-grade level metro station as shown in Fig. 5.8, during the arrival and departure of trains, signal strength is decreased by a range. APs that provide comparatively strong signal strength such as -50 dBm in absence of train, tends to provide weaker signal strength like -75 to -80 dBm during arrival or departure of train. Again, APs that provide comparatively weak signal strength like around -70 dBm in absence of train tend to provide mild or no signal strength at all in presence of train.

When a user is traveling from at-grade level metro station to underground level metro station as shown in Fig. 5.9 and 5.10, initially, RSSI signal strengths are available from APs fixed at the at-grade level metro station. After around 20 seconds there is no signal strength found from any AP. Then again, as the train reaches closer to the underground level metro station, signal strengths are received from APs that are fixed at underground level station.

5.2 Granularity Optimization through kDN

k-Disagreeing Neighbors (kDN) [81] is a well-known method to measure the hardness of an instance. In this method, at first, for any specific instance $\boldsymbol{i}$ from class L, its k-Nearest Neighbors (kNeN) are identified using Euclidean distance [85]. Among those k neighbors, the number of neighbors that belong to different classes other than L, are the kDNs of $\boldsymbol{i}$. The mathematical formula for kDN is expressed as follows.

$$kDN(i) = \frac{|j : j \in kNeN(i) \wedge t(j) \neq t(i)|}{k} \tag{5.1}$$

The kDN of a data instance i is the percentage of its kNeNs that holds a different target class value than that of i. When a sufficient amount of kDN instances of a particular class l_a shares a common target class label l_b, RSSI pattern similarities between these classes are reflected in this finding. In 5.1, j is the number of data instances that exist in the kNeNs of i and their target class $t(j)$ is different from the target class $t(i)$ of the data instance i. In this work, the algorithm GO-kDN (Algorithm 4) has been proposed that identifies the optimized granularity level for an indoor localization dataset. Here, the term "optimized" means, while tracking a user in a large indoor region, (s)he will be located within an area as precisely as possible, based on the RSSI characteristics around that area.

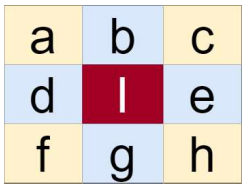


Figure 5.11: S: {a,b,c,d,e,f,g,h} is the octet-neighbor set of the location l; Each element of S is directly connected to l by eight directions - north, south, east, west, north-east, north-west, south-east, south-west

Algorithm 4: GO-kDN : Granularity Optimization using kDN

Input: RSSI dataset $\boldsymbol{D}$ with fine-grained labels
k: Number of nearest disagreeing neighbors
Output: RSSI dataset $\boldsymbol{D}_{new}$ with optimized granularity labels
//Check the percentage of instances that are in kDN list of some particular location, is below threshold percentage
for each label l_a in $\boldsymbol{D}$
  for each label l_b in $\boldsymbol{D}$
    if $l_a \neq l_b$ and l_b is in octet-neighbor of l_a **then**
      $n_1 \leftarrow$total number of instances of l_b in $\boldsymbol{D}$
      $n_2 \leftarrow$number of instances of l_b that are kDN of l_a
      if $\frac{n_2}{n_1} \times 100 < t_h$ **then**
        remove l_b from kDN list of l_a
a$\leftarrow$localization accuracy on $\boldsymbol{D}$
$\boldsymbol{D}_{new} \leftarrow \boldsymbol{D}$
for $\boldsymbol{i}$ in 1 to k
  $h \leftarrow (\frac{i}{k}) \times 100$ //percentage of hardness
  $\boldsymbol{D}_1 \leftarrow$set of instances of $\boldsymbol{D}$ having $h\%$ hard instances
  divide $\boldsymbol{D}_1$ in groups based on fine-grained labels
  for each group $\boldsymbol{g}$ in $\boldsymbol{D}_1$
    $\boldsymbol{x} \leftarrow$fine-grained label of $\boldsymbol{g}$
    $\boldsymbol{new\_x} \leftarrow \boldsymbol{x}$
    initialize empty list **Q** //to store labels of kDNs of $\boldsymbol{x}$
    for each instance $\boldsymbol{j}$ in $\boldsymbol{g}$
      $\boldsymbol{S} \leftarrow$set of labels of kDNs of $\boldsymbol{j}$
      for each label $\boldsymbol{l}$ in $\boldsymbol{S}$
        if $\boldsymbol{l}$ not in **Q** **then**
          append $\boldsymbol{l}$ to **Q**
    for each label n_l in **Q**
      if n_l is in octet-neighbor of $\boldsymbol{L}$ **then**
        $\boldsymbol{new_x} \leftarrow$merge($\boldsymbol{new\_x}$, n_l)
    update $\boldsymbol{x}$ as $\boldsymbol{new\_x}$ in $\boldsymbol{D}_1$ and $\boldsymbol{D}$

  $\boldsymbol{D}'' \leftarrow \boldsymbol{D}$ //$\boldsymbol{D}''$ is phase $\boldsymbol{i}$ dataset
  a$'$ $\leftarrow$Classifier($\boldsymbol{D}''$)
  if a$'$ <a **then**
    break
  else
    a$\leftarrow$a$'$ //update best localization accuracy
    $\boldsymbol{D}_{new} \leftarrow \boldsymbol{D}''$

This algorithm's core is to study the distribution of the data in a class and hence, analyze instance hardness through the kDN concept (equation 5.1). It is a measure that indicates how much distinguishable the class instances are. Since the AP distribution across indoor regions are not uniform, there would be some places that is not covered sufficiently by the APs. The reflection of distribution of available APs on the localization

classification process is investigated through this instance hardness. The proposed GO-kDN algorithm takes into account the effect of such non uniform distribution of APs and designs an optimally granular nonuniform localization.

As represented in Algorithm 4, Two classes l_a and l_b are considered for merging if both these conditions are satisfied- (i) l_a contains a sufficient no of hard instances that are closer to l_b and (ii) class labels of l_a and l_b denote neighboring locations in real-life (line no. 1-3). Otherwise, l_b will be removed from kDN list of l_a (line no. 7). Then, the entire dataset is divided into groups, each containing instances of same hardness. There will be (k) groups with distinct hardness intensities, for an integer value of k selected for kDN. Accordingly, there will be (k + 1) phases in the proposed algorithm. The initial phase is phase_0, it contains the preprocessed data with fine-grained labeling details. From phase_1 to phase k (line no. 10), the level of granularity is updated step-by-step. In each of these k phases, for each instance of current dataset that belongs to the class label L_i, the class label L_{kDN} of its each kDN data instance is checked. If L_{kDN} is physically located in the octet-neighbor set of L_i, as shown in Fig. 5.11, then L_i and L_{kDN} are merged together to form a new class label covering a location point with coarser level of granularity (line no. 25). L_i and L_{kDN} are replaced by the new label throughout the entire dataset. After updating the granularity level at a certain phase j with a certain level of hard instances, localization accuracy (a' at line no. 29)is computed using ML and DL classifiers. Then, in phase (j + 1) granularity level is updated analyzing the harder instances compared to the previous phase, and again, localization accuracy is computed. As represented in Algorithm 4 at line no. 30-34, if an improvement in accuracy is observed or it remains the same, GO-kDN goes to the next phase. Otherwise, the process stops and that particular level of granularity is considered optimized. In the Algorithm 4, D_{new} represents the final dataset obtained after granularity optimization.

To check the percentage of number of instances of kDNs for all locations, the complexity is $O(l^2)$ where l is the total number of location points. Since GO-kDN executes in k phases, in each phase the execution continues considering a class label-specific case, and in each case, each instance is visited one by one. Thus, for coarsening the granularity asymmetrically, the time complexity in the worst case is $O(k \times l \times n)$ where n is the max-

imum number of instances per class. Again, in each of the k phases, when the coarsening of granularity is done, execution of classifier takes place. This results in a time complexity of O($k \times c$) where c is the complexity of the classification algorithm.

The algorithm is executed during the training phase only. For the localization phase, only the pretrained classification models would be executed to classify among the optimized no of location classes. Thus, the algorithm would not hamper the real-time localization performance.

5.3 GO-kDN Experimental Results

The experimentation is carried out on MetroIndoorLoc and two benchmark datasets: JUIndoorLoc[2] [84] and BLE-dataset[3] [86]. Four machine learning and one deep learning classifiers commonly used for indoor localization, are used in this work to compute localization accuracy and error. Before localization, all datasets were labeled with optimized granularity labels using proposed GO-kDN algorithm. The value of k is desired to be at least half of the sample count per class of the dataset, so that if a sufficient number of k disagreeing neighbors belong to some specific class, it will be reflected. GO-kDN was executed for k=10, thus, resulting in 11 phases. The first phase is 0^{th} phase with fine-grained location labels. Phase 1, 2, ..., 10 are the phases where 10%, 20%, ..., 100% hard instances are analyzed respectively, to update the granularity levels. An experiment has been conducted to show the effect of the individual phases of the algorithm.

Table 5.2: Comparison of IPC algorithm with Proposed GO-kDN for at-grade level metro station

Algorithm	**Parameters**	**DT**	**kNN**	**SVM**	**RF**	**1D-CNN**
IPC	$\alpha = 0.0$	78.25	71.82	77.26	78.58	75.45
	$\alpha = 0.1$	75.45	71.82	72.81	75.28	71.66
	$\alpha = 0.2$	75.45	71.82	72.81	75.28	66.23
GO-kDN	initial phase	77.75	75.61	77.10	77.75	74.30
	optimized phase	92.75	90.93	92.58	92.91	92.13

In case of the at-grade level station of MetroIndoorLoc, localization accuracy at the initial phase is maximum 77.75% for tree-based classifiers, and it is gradually increased up to

[2]https://drive.google.com/open?id=1_z1qhoRIcpineP9AHkfVGCfB2Fd_e-fD
[3]https://www.kaggle.com/datasets/mehdimka/ble-rssi-dataset?resource=download

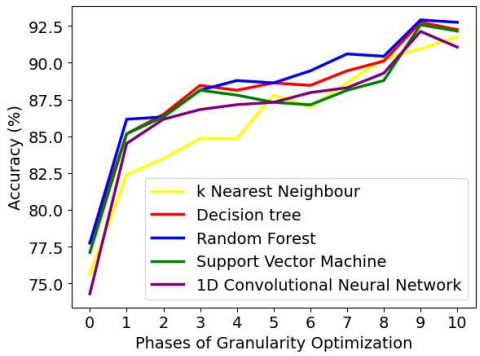


Figure 5.12: At-grade level station of MetroIndoorLoc: granularity optimization point found at 9th phase with localization accuracy 92.91%

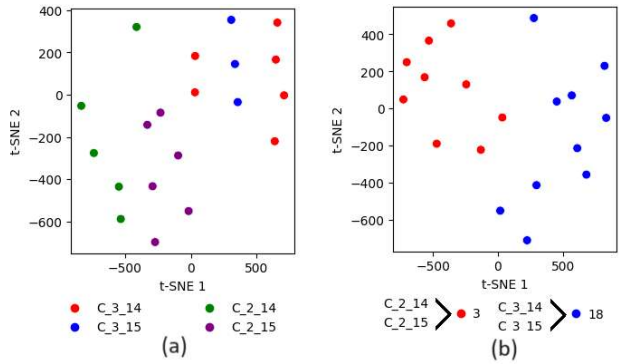


Figure 5.13: t-SNE visualization of neighbor location points in (a) finest and (b) optimized granularity level of at-grade level station of MetroIndoorLoc

9th phase reaching to an accuracy of 92.91%, for Random forest classifier. In next phase, the accuracy is decreased a bit for four among five classifiers, resulting in a knee point as shown in Fig. 5.12. Thus, 9^{th} phase can be considered as the phase of optimization. In Fig. 5.13(a) t-SNE visualization of four neighbor location points are shown. It is observed that location C_2_14 holds very close RSSI characteristics to location C_2_15. The data points of locations C_3_14 and C_3_15 have been mixed up in a common 2D visualization space. In the 9^{th} phase of GO-kDN algorithm, C_2_14 is merged with C_2_15 and named as class label 3. Similarly, C_3_14 is merged with C_3_15 and named as class label 18 (Fig. 5.13(b)).

The positions of location points 3 and 18 can be found in Fig. 5.14, in which the floormap is represented with optimized granularity location points. There are total 38 location points among which two cover large area, rest cover small or medium area. If one wants to achieve fine-grained localization throughout the station, concerned areas can be identified

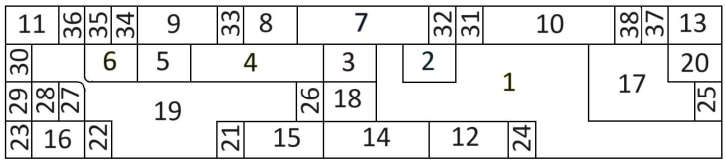


Figure 5.14: Floormap of at-grade level station of MetroIndoorLoc, after granularity optimization at phase 9 using proposed GO-kDN algorithm

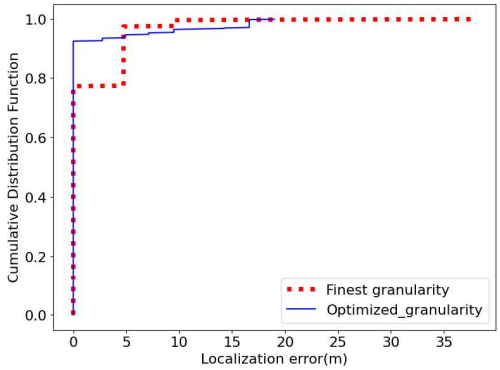


Figure 5.15: Cumulative Distribution Function of localization error in finest and optimized granularity level of at-grade level station of MetroIndoorLoc

from this floormap.

The performance of proposed GO-kDN algorithm has been compared to another recent indoor positioning approach proposed in [87]. The similarity between two approaches is that, both perform instance level analysis to finalize the labels of all data instances. In [87], researchers proposed improved public c-means (IPC) algorithm. IPC algorithm computes the likelihood that each instance belongs to each class. This likelihood is represented by membership value, just like traditional Fuzzy-C-means [88] clustering algorithm. If a data instance i is found to hold membership values (m_a, m_b) for class A and class B and $(m_a - m_b) \leq \alpha$, then i is considered as member of both the classes, A and B. For at-grade level station of MetroIndoorLoc, the training dataset was fed to IPC algorithm three times for three different values of α. The accuracy obtained in each case is represented in Table 5.2. It can be seen that, localization accuracy for IPC with $\alpha = 0.0$ is slightly improved than initial phase of GO-kDN, but GO-kDN performs much better in its optimized phase.

The localization errors at the finest and optimized granularity levels of GO-kDN are presented in Fig. 5.15. Here, error is the distance between the centres of the actual location point and the predicted location point in metres. As a result, the ground truth

error between two consecutive location points is smaller in finest granularity level, and comparatively larger in the optimized granularity level. From the CDF plot it is visible that after granularity optimization more than 90% locations are identified without any error and the maximum error value is also reduced significantly.

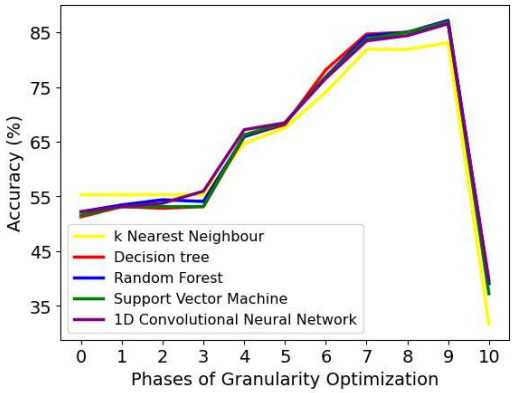


Figure 5.16: Underground station of MetroIndoorLoc: granularity optimization point found at 9th phase with localization accuracy 87.18%

The next experiment demonstrates the localization accuracies for phases of GO-kDN on underground station dataset of MetroIndoorLoc shown in Fig. 5.16. Since this station is at underground level, localization process deals with a harder situation. Accuracy is only around 55% at the initial phase, and after applying GO-kDN, it is increased up to 87.18% using Random forest classifier. After 9^{th} phase, a sharp decrease in accuracy is observed after the knee point (Fig. 5.16). This may happen when, as a result of coarsening granularity, two consecutive location points are large enough and they cover areas that are far contradictory in RSSI characteristics, just like they hold a distinguishable relationship with physically far apart location points. In such a case, if these two neighboring locations are merged, the classifier will be highly confused, and accuracy will fall sharply.

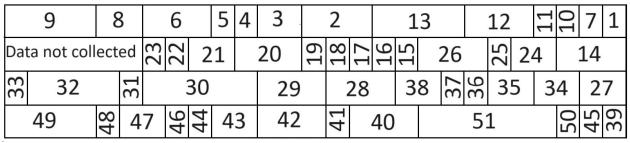


Figure 5.17: Floormap of underground station of MetroIndoorLoc, after granularity optimization at phase 9 using proposed GO-kDN algorithm

Fig. 5.17 represents the floormap of underground station dataset of MetroIndoorLoc, with optimized granularity at phase 9. Due to unavoidable circumstances, data was not

collected from a portion of the floor, as indicated in the Fig. 5.17, and the rest part was divided into 51 location points. Comparing Fig. 5.14 and 5.17, it can be observed that, in case of the underground station a finer level of granularity is obtained after optimization.

To check the generalization of proposed GO-kDN algorithm beyond station platforms, it has been executed on the benchmark dataset JUIndoorLoc [84]. This dataset contains WiFi RSSI data collected (the shaded portion) from 172 APs on three different floors of the university, maintaining a granularity level $1m \times 1m$. In the 4^{th} floor, diverse RSSI characteristics were observed from 120 APs, and we have selected this floor for experimentation.

Initially, the localization accuracy is around 65% and remains the same in phase 1 also, as there was no 10% hard instance in the dataset. After that, a sharp increase in accuracy is observed from phase 6 to 7 for all classifiers. The reason is that a lesser number of instances has been found that are 60% or less hard. From phase 7, accuracy continued to increase up to the 9^{th} phase, and then the curve becomes almost parallel to the horizontal axis. Using GO-kDN, localization accuracy is improved up to 97.51% at the 9^{th} phase, as the knee point is observed in Fig. 5.18. This improvement in accuracy is obtained even after a few location points maintain the finest granularity level, as shown in Fig. 5.19. This shows how granularity can be optimized in a non-uniform pattern utilizing the non-uniform distribution of APs.

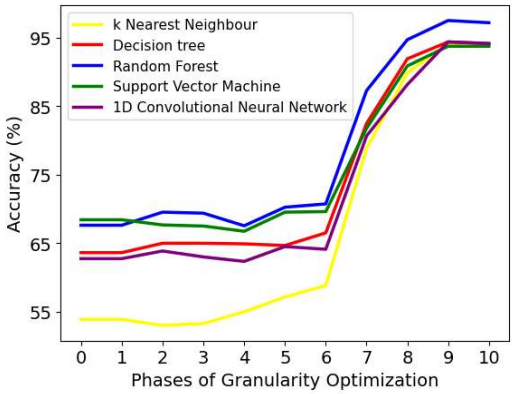


Figure 5.18: JUIndoorLoc- phase-wise accuracy improvement optimization point observed at 9th phase

Finally, GO-kDN algorithm is executed on BLE dataset [86] to show its performance on small area subject to discontinuous fingerprints. In BLE_ dataset [86], the overall area

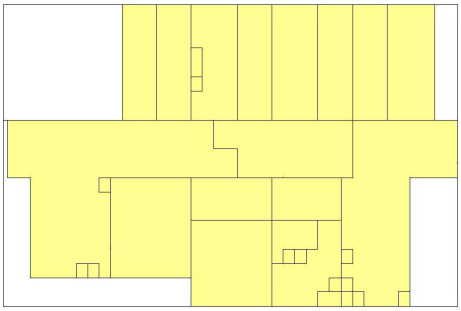


Figure 5.19: Floormap of JUIndoorLoc after granularity optimization at phase 9 using proposed GO-kDN algorithm

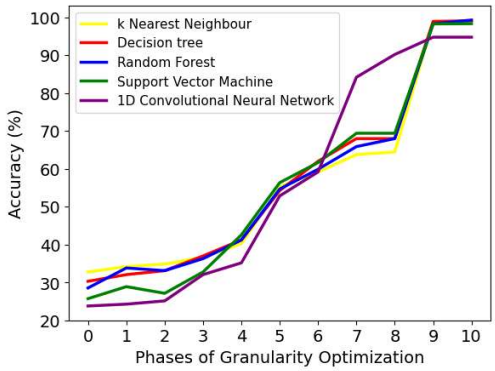


Figure 5.20: BLE_Dataset- phase-wise accuracy improvement

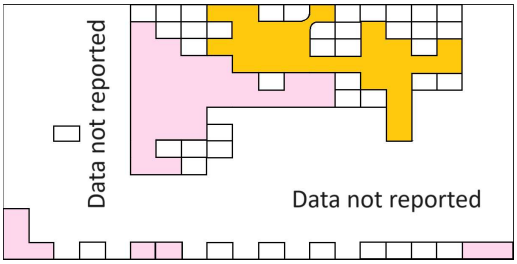


Figure 5.21: Floormap of BLE dataset using GO-kDN algorithm optimization at phase 6 using proposed GO-kDN algorithm

is small, $(20m \times 15m)$ and data was collected in labeled and unlabeled manner, provided separately. In the labeled dataset, the location provided are discontinuous. In general, the Bluetooth signal covers shorter area and variation of signal strength with respect to distance is also less than that of WiFi. As a result, at initial stage, the localization accuracy is very poor when the level of granularity is fine as shown in Fig. 5.20. Using GO-kDN algorithm, accuracy gradually increases and becomes considerable at phase 6, when all classifiers provide greater than 55% accuracy. However, due to the steady characteristic of

Bluetooth signal over the relatively small experimental region, most of the locations merge into one single location from phase 7 (except some locations near the border area), and all location points lie within one single location point at phase 10. In such a case, observing the combination of localization accuracy and granularity level, the optimization phase can be decided. If a certain desired level of localization accuracy is achieved and the location points are coarse enough, that phase can be considered as an optimized granularity phase, as shown in Fig. 5.21. In Fig. 5.21, two coarse location points are observed, and rest are separated fine-grained location points.

5.4 Data augmentation for minority classes after Granularity Optimization

After granularity optimization, class imbalance is an expected phenomenon in the optimized granularity dataset. Coarse-grained locations has greater sample count and fine-grained small locations have lower sample count. To bring a balance in the dataset, we have proposed a data augmentation strategy for minority classes using Conditional GAN (CGAN). The entire approach is designed and executed in steps, as presented in Algorithm 5 named as DAMLoc. The workflow is presented in Figure 5.22 and explained in detail as follows.

Statistical analysis of class imbalance: When the idea of increasing sample counts in a dataset comes to mind, the very first question to answer is: adding how many data samples would be sufficient yet practical? Maintaining this balance is important. For this reason, two things are computed- the most frequent number of samples per class (statistical mode M_o), and, the median value among all of the sample counts per class (statistical median M_e). In ideal case, all locations should contain equal number of samples. Thus, to increase the sample counts of required locations up to M_o is ideally correct. However, if most of the locations contain a small number of samples which is even less than the median sample count value M_e, in that case required locations' sample counts are increased up to M_e. This analysis is presented in the lines 2 to 13 of proposed Algorithm 5, DAMLoc.

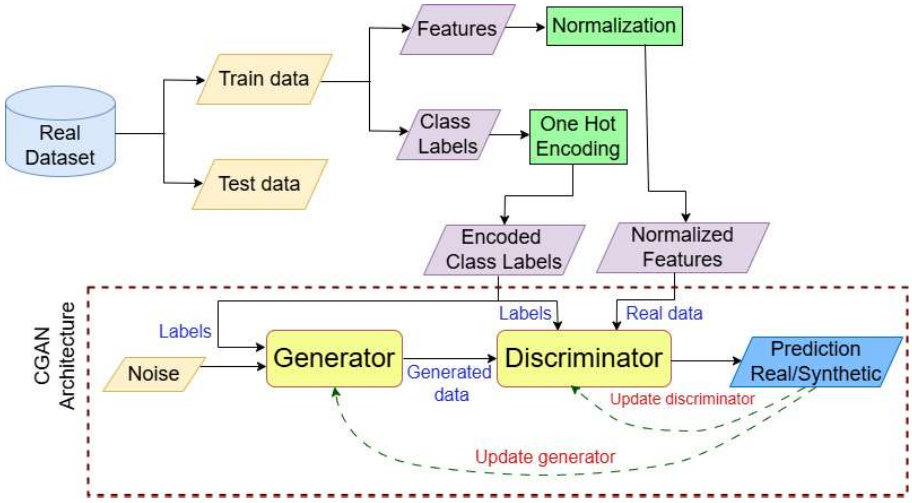


Figure 5.22: Workflow of the CGAN-based fingerprint data generation method

Data Normalization: The WiFi RSSI fingerprint values usually range around -10 dBm (strongest signal) to -110 dBm (weakest/no signal). In order to meaningfully reflect the small and large data variation patterns across the location classes subject to different conditions, the input dataset is normalized to the 0 to 1 range before feeding it to the CGAN model.

Label encoding: In a fingerprint dataset, locations represent the class labels which are often presented in categorical format. Before feeding to CGAN, these labels are encoded to provide the model with explicit and structured information about the conditions under which the generative process should occur. In this work, using OneHotEncoder, the categorical labels are converted into class-specific unique binary vectors.

Training the CGAN model: The CGAN model is used to generate synthetic RSSI fingerprint data, following a game-theoretic approach of zero-sum game in which one player's gain is balanced by the other player's loss. CGAN has two neural network components—a generator and a discriminator, which act as players in the game, adversarial to each other. The gain of the generator causes the loss of the discriminator, and vice versa.

The generator receives a random noise and the class labels representing location points

Algorithm 5: DAMLoc : Data Augmentation for Minority Locations

Input: Real RSSI fingerprint dataset D
  No. of location points n in D
  Set S of all locations in D
  Trained CGAN model M_{CGAN}
Output: Augmented RSSI fingerprint dataset D'

$D' \leftarrow D$
Initialize empty list *sample_count*
for $i \leftarrow 1$ to n **do**
    $c \leftarrow$ sample count for i^{th} location
    Append c to the list *sample_count*
end for
$M_o \leftarrow$ Mode value of *sample_count*
$M_e \leftarrow$ Median value of *sample_count*
if $M_o \geq M_e$ **then**
    $M \leftarrow M_o$
else
    $M \leftarrow M_e$
end if
for all locations l in S **do**
    $c \leftarrow$ sample count for location l
    if $c < M$ **then**
        $x \leftarrow (M - c)$
        Generate x samples for class l using M_{CGAN}
        Append generated x samples for class l in D'
    end if
end for

as inputs. The noise is a random vector of predefined size, drawn from a Gaussian distribution. The encoded location (class) labels are given as conditions to the generator. These two inputs are concatenated into a single vector to condition the data generation process. The concatenated input is then passed through a Dense layer with ReLU activation. This layer learns the relation between noise and class label, and generate feature vectors accordingly in the following output layer. The discriminator is responsible for distinguishing a data sample as real or synthetic. The discriminator is fed with both (real data, class label) and (generated data, class label) pairs. One crucial thing is to be noted that, only the train set real data samples are fed to discriminator, and real test set are kept aside to validate the model later. This in turn measures the efficiency of real-time user localization. The discriminator's loss is computed from two losses giving equal weights—classifying real data as real/synthetic and classifying generated data as real/synthetic. The generator is

fed with a loss computed by the entire CGAN model, which is the combined loss from the generator and discriminator.

The generator is updated to minimize this loss, thus improving its ability to generate more realistic data. Mathematically, $G(z, y)$ represents the generator's output for a random noise input z and a condition (location class) y. Then, $D(G(z, y), y)$ represents the probability predicted by the discriminator that, the generated sample is real with respect to the condition y. Generator's goal is to maximize $D(G(z, y), y)$ and discriminator's goal is to minimize it and maximize $D(x, y)$, where x represents real data for the given condition y.

Generating synthetic data for minority classes: When the CGAN model is trained up to a certain number of epochs, it is saved. Then, the trained model is utilized to generate required number of synthetic data for the minority locations to reach the required sample count per class. At this point, real training samples of minority classes only are fed to the CGAN.

5.4.1 Data Augmentation using CGAN : Experimental Result

We have experimented proposed DAMLoc Algorithm on the underground metro station dataset. There are total 51 classes. After the dataset is divided into 80%-20% ratio, the mode and median are computed from the 80% training dataset, as shown in Figure 5.23. The median and mode of the training dataset are found to be 16 and 8 respectively. Thus, we identified all the locations having sample count n where $n < 16$ and generated $16 - n$ synthetic samples for each locations. The Figure 5.24 represents the loss curve of CGAN during training, only real training dataset has been used to train the model. The generator and discriminator loss starts from a common point, but gradually generator loss is decreased and discriminator loss is increased, and the model reaches a convergence state.

After the model is converged, synthetic data for required locations are generated and augmented to the real training, and localization accuracy is evaluated using two classifiers - decision tree (DT) and artificial neural network (ANN). These two classifiers are chosen

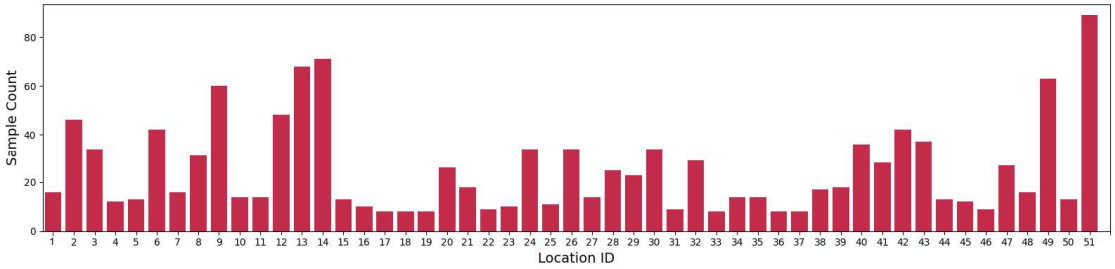


Figure 5.23: Sample count in the underground metro station training dataset

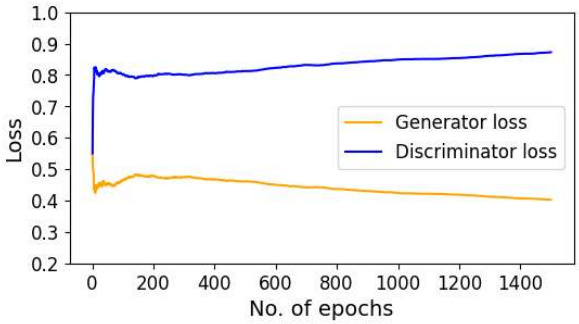


Figure 5.24: Training and validation loss in CGAN for underground metro station dataset

as their processing is completely different. DT follows tree structure pattern and feature-based classification takes place. Its important to understand how deep the tree can go, how many levels of splits it can make. Limiting maximum depth forces the model to stay simpler and focus on the most important patterns, reducing overfitting. In Figure 5.25, after augmenting synthetic data the model is capable to learn the training data pattern with limited maximum depth. On the other hand, ANN is an instance-based classification process. It treats each instances with equal importance. It is important to understand how the model is learning over iterations. When number of iterations is very high, model keeps and if the dataset is small or not very noisy, it might start memorizing the training data instead of learning general patterns. A limited number of iterations may cause underfitting. When ANN is trained using only real data, accuracy is around 80.6%, and after data augmentation it increases up to 82.2%.

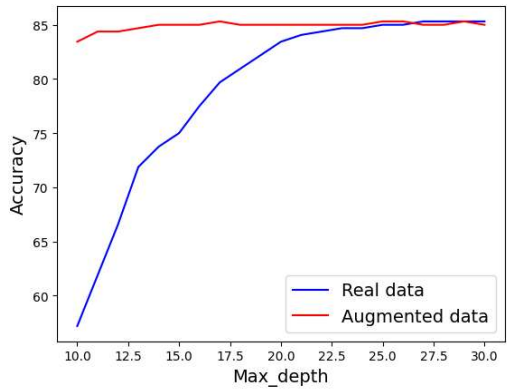


Figure 5.25: After data augmentation decision tree converges faster, with lower max_depth

5.5 Summary

Indoor localization is a growing domain of research, and significant contributions in various building areas are highlighted in existing works. However, there is a lack of experimentation in public indoor areas outside buildings. In this work, a new dataset, MetroIndoorLoc, is provided that consists of two metro station datasets: one at-grade level and another underground level. Experimental analysis of MetroIndoorLoc using different ML and DL classifiers ensures the feasibility of fingerprint-based localization in such areas. Moreover, the algorithm GO-kDN is proposed that updates the granularity level of any fingerprint dataset based on instance hardness analysis to improve localization accuracy. GO-kDN has been executed on MetroIndoorLoc and two other benchmark datasets, and in each case, significant improvements in localization accuracy and depletion in localization error were observed. Proposed approach is also found to perform in an impressive way when compared with another recent state-of-the-art indoor localization method. Label-specific synthetic data generation is a current trend in fingerprint-based localization to improve localization performance. Now, it is yet to be seen if the generation of synthetic data using optimized granularized labels further improves localization performance.

Accepted Work

Mallik, M. and Chowdhury, C., 2025. DAMLoc: Data Augmentation for Minority Location Classes applying CGAN for Indoor Localization. Web 6.0 and Industry 6.0 (WIN 6.0

2026), Springer.

Communicated Work

Mallik, M. and Chowdhury, C., 2025.GO-kDN: Granularity Optimization through k-Disagreeing Neighbor Concept for Machine Learning based Indoor Localization. Engineering Applications of Artificial Intelligence, Elsevier.

Chapter 6

Generating Synthetic Fingerprint Data using Generative Adversarial Network

Since site survey is a time-consuming and laborious task, collecting enough amount of labeled data from each location point is practically difficult and infeasible. In Chapter 3, we have proposed a semi-supervised coarse-grained localization approach that considers unlabeled data also, collected using crowd-sourced method, in order to increase overall volume of training data. In Chapter 4 we stepped into further detail, and proposed an approach for fine-grained localization using encoded image dataset obtained from limited amount of fine-grained labeled RSSI data. Although these two approaches have shown promising results, there is still a requirement for investigation for enhanced fine-grained localization using numeric RSSI fingerprint data.

In this chapter, we propose an approach for generating synthetic fingerprint data using Generative Adversarial Network (GAN). GAN is a deep learning model widely used for synthetic image data generation; we applied the same logic for generating numeric data instead. At first, GAN is separately trained each time for different locations, since it is an unsupervised model and cannot learn location-specific data pattern. Then we designed and implemented a new GAN variant, named as Vectored labeled GAN (VL-GAN) that not only generates synthetic data, but also generates corresponding label in vectored format.

Moreover, it is capable to generate labels for real unlabeled data samples as well, thus can be used as a data-annotator model on requirements.

6.1 Unlabeled fingerprint data generation and augmentation using GAN

In this section, the workflow and experimental analysis for separate training of GAN for each different location is presented.

6.1.1 Generative Adversarial Networks (GAN) Architecture:

GAN [79] is a deep learning model, consists of two neural network components, the generator and the discriminator. GAN works on the zero-sum gaming principle. The generator is given a random noise vector from a latent space and maps it to the same dimension as the real data. Initially it generates synthetic data samples based on the given random input, without understanding the semantics of the input data features. Later through training, it learns to capture the distribution and features of the real data, including semantic relationships, in a statistical sense. Both the original and generated data are fed as input to the discriminator, which is responsible for the identification of real and fake data. It learns the representation of the original data samples keenly and guides the generator by sending feedback. That means, GAN works on the indirect training method through the discriminator, which predicts how realistic the input seems. The two blocks work in an adversarial way and improve each other. The generator uses the discriminator's feedback to produce data that increasingly resembles the real data. Ideally, the model converges when the discriminator can no longer distinguish real from generated data. At this point, the generator is producing highly realistic samples, although in practice, stable convergence can be difficult to achieve and may not be reflected clearly in loss values. The basic architecture of GAN is represented in Figure 6.1.

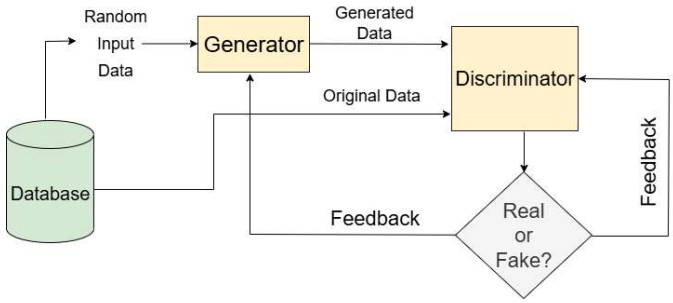


Figure 6.1: Generative Adversarial Networks (GAN) Architecture

6.1.2 Proposed workflow for fingerprint data augmentation using GAN

We have proposed a simple workflow for fingerprint data generation and augmentation as represented in Figure 6.2. At first, the dataset is divided into separate groups, each group containing data for a specific class label (location point). From each group, around 80% samples are selected for training purpose and rest of the sample are kept for test purpose. Only the train data samples have been fed as input to the discriminator of GAN for each location point at a time, to produce new data samples for that class. When samples for all the location points are generated, they are merged with the original training data samples to form the complete augmented training dataset. Augmented samples are added only to the train data, the test data remains the same.

The discriminator model is fed with a n-dimensional input and a one-dimensional output, where n is the number of features (APs in this case) of the dataset. The discriminator consists of four hidden layers, composed of 256, 128, 64 and 32 neurons respectively, followed by Rectified Linear Unit (ReLU) activation function. After each hidden layer, dropout has been used to avoid overfitting. The output layer is composed of a single neuron along with the sigmoidal activation to represent a probability value. The structure of discriminator model is represented in Figure 6.3.

The generator model is fed with the n-dimensional input, which will receive sample points of specific class labels $(p_1, p_2, ..., p_n)$, and a n-dimensional output providing $(q_1, q_2, ..., q_n)$ points resembling from the training data. The generator is composed of two hidden layers with 128 and 256 neurons respectively, each followed by ReLU activation function, and a linear activation output layer that consists of n neurons. In this way, the

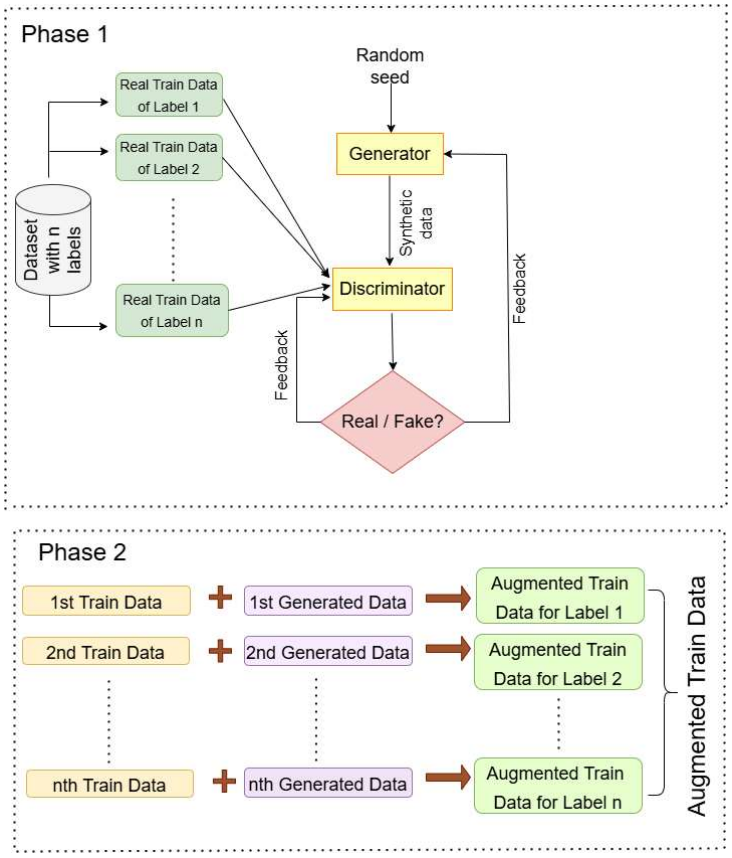


Figure 6.2: Proposed workflow for data augmentation

output is generated with a vector of n elements containing any value between negative infinity to infinity. The structure of generator model is represented in Figure 6.4.

6.1.3 Experimental result for unlabeled data augmentation using GAN

In our experiment, the GAN model is implemented using PyTorch library. To have a deterministic randomness, a static seed value has been specified for intialization of a pseudo random number generator. Static seed value is used to obtain the same pattern of data every time. With a learning rate 0.001 the model is trained repetitively 200 times with binary cross-entropy loss function and Adam optimizer. Since the learning of GAN depends on the data pattern and data augmentation is practically applied to small datasets, it will be beneficial to examine the data augmentation process with different smaller data portions of different contexts. We have selected two benchmark datasets of different contexts - one university dataset JUIndoorLoc Dataset_1, and another Shopping

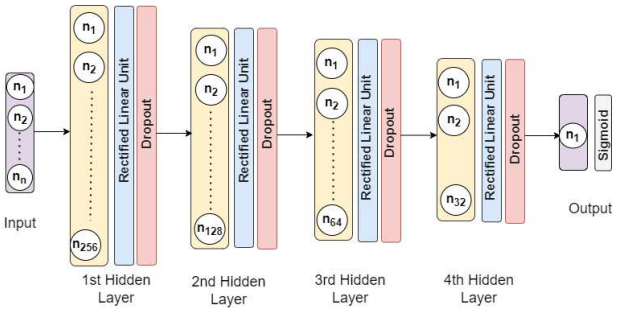


Figure 6.3: Discriminator model structure

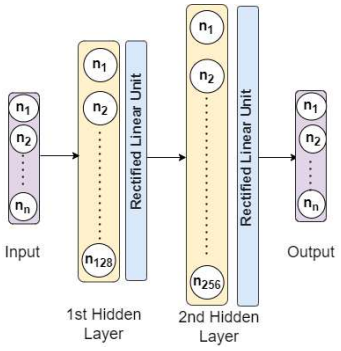


Figure 6.4: Generator model structure

mall dataset, Dataset_2. Two rooms from Dataset_1 and one floor from Dataset_2 have been selected for experimentation. In Figure 6.5, it can be observed that initially the loss of the discriminator is high and that of the generator is less. The reason is that the generator generates random samples on its own and the discriminator is completely unaware of the expected pattern of the generated data. After just a few epochs, the discriminator starts to learn from the original and generated data samples, and sends feedback to it accordingly. Approximately after 125 epochs, the discriminator and generator both come to a convergence.

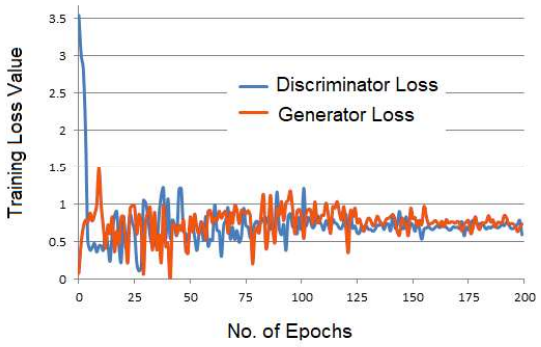


Figure 6.5: Training Loss of Generator and Discriminator

For Dataset_1, classification results are represented in Figure 6.6 and 6.7. For Dataset_2, different amounts of generated data are added gradually. It can be observed in Figure 6.8 that the accuracy improves with a spike after adding 25% data, and then the rate of improvement slows down. A maximum increment is found for ANN.

Although there may be a few cases where the augmentation is not sufficient to improve the accuracy, for example, SVM in Figure 6.6, the overall implementation results of GAN

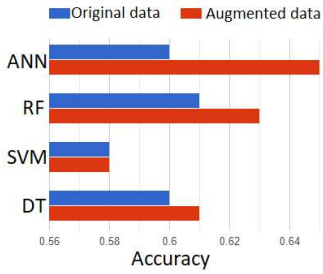


Figure 6.6: Localization performance is improved for augmented data for 1st room of Dataset_1

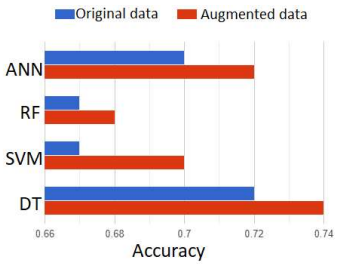


Figure 6.7: Localization performance is improved for augmented data for 2nd room of Dataset_1

indicate that the data generated for each location strongly inherits the fingerprint features of the original data of that location. To validate the closeness among the statistical distributions of the generated data and the original data, Kullback-Leibler divergence (KL-divergence) [80] is used, as shown in Table 6.1. KL-divergence represents the similarity between the data distributions of two vectors. After generating equal amounts of data, KL-divergence is computed from each i^{th} RSSI vector of the original dataset to the i^{th} RSSI vector of the generated dataset, and its mean is obtained. For each of the three cases, the mean KL-divergence is between 0.4 and 0.5, indicating a negligible difference between the data distributions of both datasets.

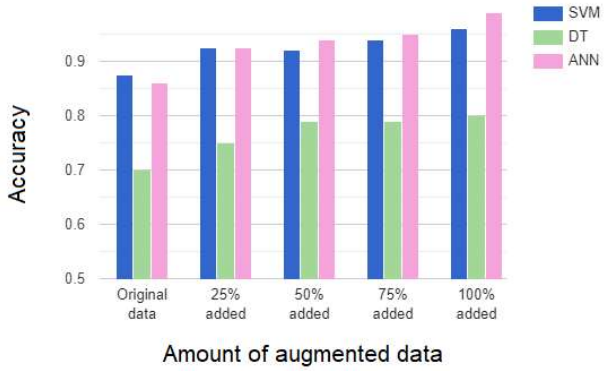


Figure 6.8: Localization performance is gradually improved for different amount of augmented data for a floor of Dataset_2

Table 6.1: Mean KL-divergence between original data and generated data by GAN

Dataset	Mean KL-divergence
Room 1 Dataset_1	0.41
Room 2 Dataset_1	0.47
Dataset_2	0.40

6.2 Vectored labeled Generative Adversarial Networks (VL-GAN)

In this work, a three-player generative model, VL-GAN, has been proposed. VL-GAN approaches the classification problem in a generative manner. In VL-GAN, we build the concept of data augmentation and labeling from another point of view. The primary idea of VL-GAN is to establish a relationship between latent space and data space like GAN, but also establish the data dimension and class dimension which will effectively benefit the classification of data points. VL-GAN is a three-player min-max game model where along with the obvious generator-discriminator duo, we have an additional generator model which acts as the classifier, and generates class labels. The generative classifier learns the distribution of the samples in the data space and generate vectored labels accordingly. The concept of vectored location label is introduced through a Label-to-Vector (LTV) algorithm by computing the distance between the centroids of each pair of classes. Thus, assignment of labels is least affected by noises and the vectored label becomes a measure of the relative class similarity of a data point. The vector contains numeric scores between -1 to 1, indicating the likelihood of the sample belonging to each class. At first, the classifier is trained with real data samples and generates vectored labels accordingly. By comparing generated and real vectored labels for real samples we can measure the classifier's accuracy and loss. Then, this generative classifier can be used to generate labels for the newly generated unlabeled data samples.

The data pre-processing mechanisms (shown in Figure 6.9) are discussed first followed by a detailed description of the proposed model.

6.2.1 Data Pre-processing for VL-GAN

Indoor localization datasets consist of WiFi signal RSSI vectors that are received from different APs at a given location point. APs are the features and location point is the class label generally denoted as (x, y) or (x, y, z). Three benchmark datasets are considered in this work- SODIndoorLoc dataset (Dataset_1)[91], Hybrid dataset (Dataset_2)[24], and

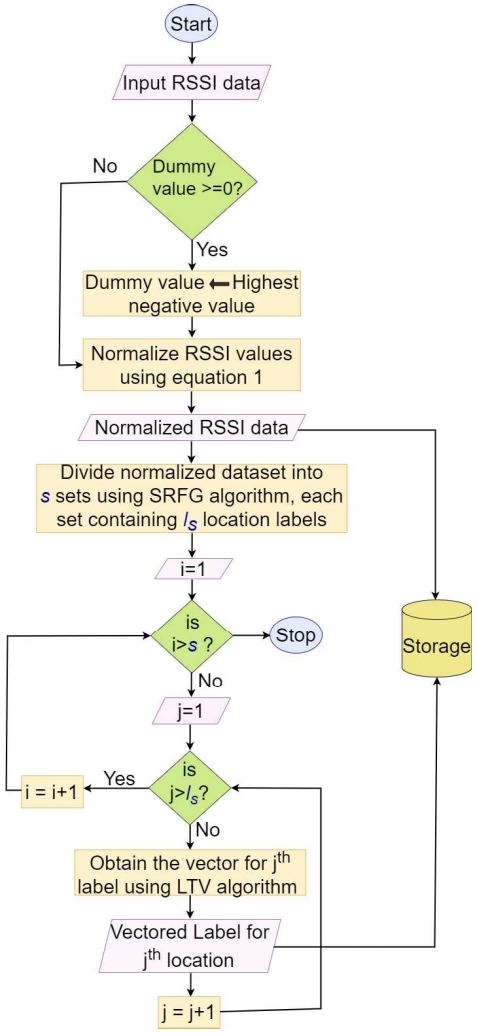


Figure 6.9: Flowchart for Data Preprocessing

Nexus-5 dataset (Dataset_3)[92]. These datasets share a common characteristic of this domain: the data are collected from consecutive coordinates, resulting in a large number of classes. Due to the congested nature of the location points (classes), neighboring classes have similarities among their fingerprint patterns. This is a characteristic feature of precisely labeled indoor localization datasets that makes the learning process of any DL model difficult. Although Dataset_2 contains BLE data along with WiFi data, in this work, we consider the WiFi features only to maintain consistency among the three datasets and reach a generalized conclusion.

Using exploratory data analysis (EDA), it is observed that Dataset_2 and Dataset_3 represent the absence of RSSI signal at any location with a minimum value of -110 and

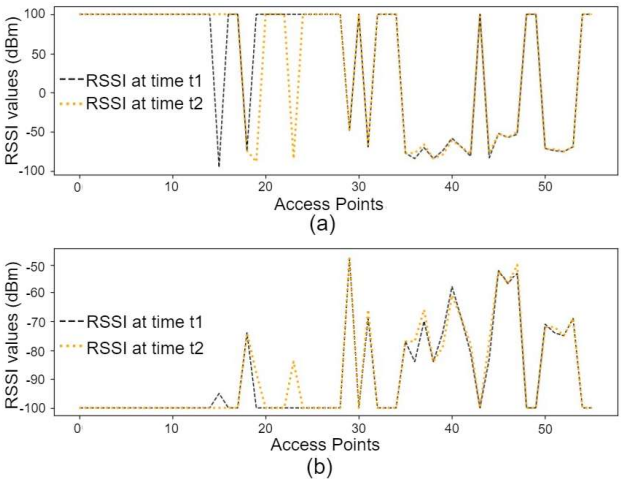


Figure 6.10: RSSI patterns of the same location, collected at different time, when missing signal strength is interpreted as (a)100 dBm and (b) -100 dBm.

-100, respectively, whereas in Dataset_1 it is represented by a positive dummy value of 100. Different RSSI vectors collected over different times at same location must reflect the AP-specific RSSI similarities. In Figure 6.10 (a), it is observed that two RSSI patterns of the same location differ by almost 90 dBm for AP 20, 23, and 24 because absence of signal strength near to -95 dBm was represented by 100 in the dataset. However, when the missing values are represented by -100, a more realistic location specific RSSI pattern would be captured as shown in Figure 6.10 (b). Accordingly, the missing values have been updated for a realistic interpretation.

To ease the learning process, original data has been normalized for (0,1) range maintaining the original statistical distribution. Any RSSI value R_v is scaled to R'_v as follows.

$$R'_v = \frac{|R_v| - |R_{max}|}{|R_{min}| - |R_{max}|} \tag{6.1}$$

Here, R_{max} and R_{min} are the strongest and weakest RSSI values present in the dataset respectively.

For fine-grained localization datasets, the class boundaries overlap especially when different ambient conditions are considered. So, for faster training of the GAN generator, the complete dataset is divided into multiple sets following Set-wise RSSI Fingerprint Generation(*SRFG()*) listed as Algorithm 6. The goal is to divide the entire dataset in a way that one set would contain fingerprints from those location points that are preferably

not nearby ones. Based on the total number of location points $\boldsymbol{n}$, and preferred number of location points l_s in each set, $\lceil \frac{\boldsymbol{n}}{l_s} \rceil$ sets are obtained. Thus, the class-wise distinguishing patterns could easily be learned for the fingerprints present in a set. This encourages an early stabilization of the VL-GAN model. This is essential because, for RSSI fingerprint dataset, the location-specific pattern is stored in a 1D vector. Multiple vectors represent multiple samples of the same location. Thus, while training the VL-GAN the batch size must be very small (ideally, 1) to make the generated samples characteristic-wise very close to the original one. However, small batch size increases the learning time, so total number of training samples must not be huge. The SRFG algorithm fulfills this requirement as well, since at any given time only one single set's data and vectored labels are fed to VL-GAN. For each subset, vectored representation of each location is obtained using Algorithm 7, Label to Vector ($LTV()$). Each label's vector has a length i, where i is the number of class labels. Each value in the vector lies within the range (-1, 1), representing the affinity to centroid of each class in reverse way. That means, i^{th} value of corresponding vector for location i is -1 indicating maximum affinity. The value for any other location in the same i^{th} vector is near to 1, if the class centroid holds distant RSSI characteristics from the class centroid of location i. This difference among the class centroids is measured by Euclidean distance [90].

6.2.2 VL-GAN model architecture

The proposed VL-GAN model has three components - two generators and one discriminator as shown in Figure 6.11. The first generator is responsible for generating RSSI fingerprint data and the second generator, known as classifier, generates the vectored label for each RSSI fingerprint. In conditional-GAN (CGAN) [89], the generator is trained to generate data by taking noise and labels as input, and discriminator investigates the pair (data, label) to declare the combination as real or fake. In the proposed model, generator and discriminator work in a similar way like CGAN, whereas, the additional generator that is acting as a classifier generates the vectored label for each data sample. The likelihood that a given data sample will fall into each class is represented by the vectored label that corresponds to that sample.

The generator G is a conditional generator, provided with conditional information (location point labels in this case) along with the latent vector. The discriminator D takes as input a data and a vectored label, and applies a linear transformation to both of them. The transformed data and the labels are then combined together to obtain a score. This score is passed through a Tanh activation function and then through a softmax layer to obtain attention probabilities in transformed dimension. Finally, the output is computed as the element-wise multiplication of the probabilities and the transformed data. In order to thoroughly understand the relationship between the data and the vectored labels, the attention mechanism is utilized.

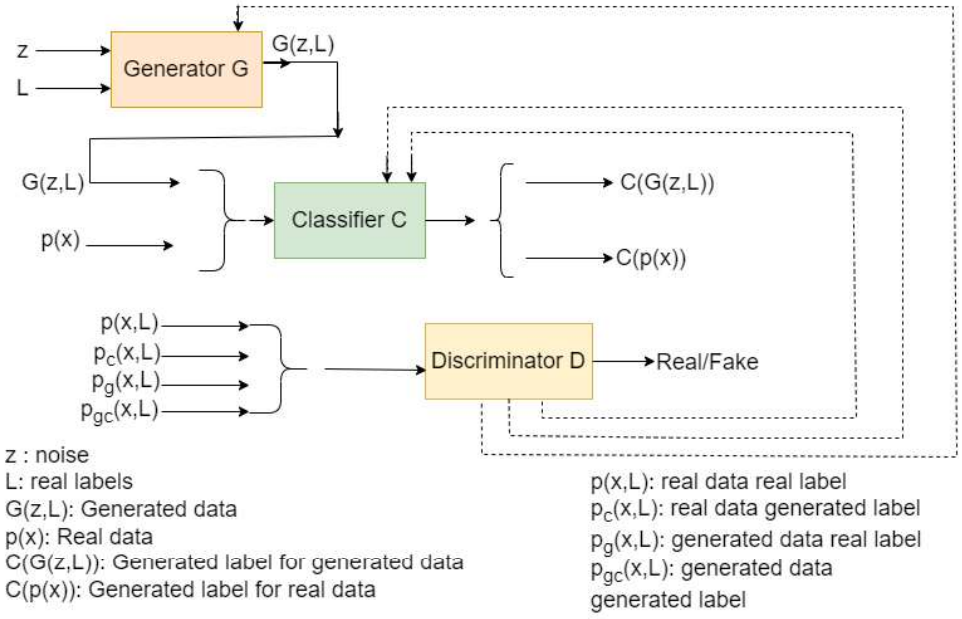


Figure 6.11: Architecture of the proposed VL-GAN model

The data is simply responsible for the vectored label being generated. The attention mechanism is kept in the first layer because it helps the discriminator to capture how the data is responsible for the vectored labels. This is the reason why both data and the label are converted to another dimension. Also, the addition of this layer ensures a more stable training of VL-GAN. The resulting scores are normalized using the SoftMax function. The output dimension is the number of classes, so that vectored form of the class labels can fit to it.

Algorithm 6: SRFG : Set-wise RSSI Fingerprint Generation

Input: Complete training RSSI fingerprint data $\boldsymbol{F}$

No. of location points $\boldsymbol{n}$ in $\boldsymbol{F}$

No. of location points l_s in each set

Output: Set-wise grouped RSSI fingerprint data

for $\boldsymbol{i}$ in 1 to $\lceil \frac{\boldsymbol{n}}{l_s} \rceil$

    initialize empty set of labels S_i

    initialize empty set of RSSI fingerprints F_i

    for $\boldsymbol{j}$ in 0 to (l_s-1)

        $\boldsymbol{loc} \leftarrow \boldsymbol{i} + (\boldsymbol{j} \times \lceil \frac{\boldsymbol{n}}{l_s} \rceil)$ // compute location id

        append $\boldsymbol{loc}$ to S_i

        append RSSI fingerprint data of $\boldsymbol{loc}$ to F_i

    $S_i' \leftarrow$ **LTV** (S_i , l_s) // vectored form of labels

    VL-GAN (F_i, S_i')

VL-GAN probability distribution: If a dataset has f number of features and n number of classes, i.e. x $\in R^f$ and L $\in R^n$ then the joint probability distribution p(x,L) for a sample-label pair (x,L) is given by $p(x, L) = p(L)p(x|L) = p(x)p(L|x)$ where p(x) and p(L) are the probability distributions of data and class labels respectively. The generator tries to match the conditional distribution to make $p_g(x|L) \approx p(x|L)$, thus, $p_g(x, L) = p(L)p_g(x|L)$. The classifier tries to match the conditional distribution to make $p_c(L|x) \approx p(L|x)$, thus, $p_c(x, L) = p(x)p_c(L|x)$.

Algorithm 7: LTV($\boldsymbol{L}$, $\boldsymbol{c}$) : Label-to-Vector

Input: Set of locations points $\boldsymbol{L}$

No. of location points $\boldsymbol{c}$

Output: Set of vectored labels for location points $\boldsymbol{V}$

initialize empty set $\boldsymbol{A}$ of class centroids
for $\boldsymbol{i}$ in 1 to $\boldsymbol{c}$
    $R_i \leftarrow$ RSSI fingerprints of location $\boldsymbol{i}$
    $C_i \leftarrow$ Mean RSSI vector of location $\boldsymbol{i}$ //class centroid
    $\boldsymbol{A[i]} \leftarrow C_i$
initialize empty matrix $\boldsymbol{B}$ // to store distance b/w centroids
for $\boldsymbol{i}$ in 1 to $\boldsymbol{c}$
  for $\boldsymbol{j}$ in 1 to $\boldsymbol{c}$
    $\boldsymbol{B[i,j]} \leftarrow \boldsymbol{e}_d(\boldsymbol{A[i],A[j]})$ //Euclidean dist. b/w centroids
$\boldsymbol{B} \leftarrow \frac{\boldsymbol{B}}{\boldsymbol{B}_{max}}$ //normalize centroid distances from 0 to 1
for $\boldsymbol{i}$ in 1 to $\boldsymbol{c}$ //set a maximum threshold of distance
    for $\boldsymbol{j}$ in 1 to $\boldsymbol{c}$
      if $\boldsymbol{B[i,j]} > t_h$ **then**
        $\boldsymbol{B[i,j]} \leftarrow t_h$
$\boldsymbol{B} \leftarrow \frac{\boldsymbol{B}}{(\boldsymbol{B}_{max} \times 2) - 1}$ //normalize the distances between -1 to 1

VL-GAN three-player gaming concept: There are three kinds of games being played from the discriminator's perspective. Due to the conditional learning of VL-GAN, in all the games, instead of only the data sample, a conditional pair of (data, label) is investigated and discriminated by the discriminator. This means the discriminator should be able to identify not only the synthetic data but also the synthetic labels. In the first game, the discriminator tries to distinguish between real label coming from the input sample (real data- real label) pair and the generated labels coming from the output of classifier. In the second game, the discriminator tries to distinguish between the data generated by the generator and the real data samples conditioned by the actual labels. In the third game, the discriminator tests how well it can distinguish between real and generated data when

the classifier generates the synthetic labels based on the data produced by the generator. The discriminator tries to see how well it distinguishes the synthetic data and synthetic label pairs compared to the synthetic data and real label pairs.

VL-GAN adaptive loss computation : In CGAN, performance improves gradually by comparing the synthetic data with real data, conditioned to real labels. Real data and real labels remain unchanged throughout the training period. Generator in VL-GAN also gets trained in the same way, and for this reason, generator's loss is computed statically, as follows.

$$G = G_{real}^{gen} = [E[logP(S = gen|[X_G, L_{RR}])]] \tag{6.2}$$

In equation 6.2, generator loss G is represented by–

G_{real}^{gen} – Loss to generate synthetic data provided real labels

X_G – Generated synthetic data

L_{RR} – Real labels in vectored form for real data

The discriminator's task is not only based on real data and real labels, but on four combinations : (real data, real label), (real data, generated label), (generated data, real label) and (generated data, generated label). Since during the begining of training, synthetic data generated by G is like noise, and distant from the real data distribution. The loss is initially high and updates rapidly, thus requires high concern. After certain period of training, this is reduced. Analyzing this whole scenario, the discriminator's loss is computed in an adaptive way so that initially the model gets highly influenced by the loss obtained for discriminating real labels versus generated labels, both conditioned for generated data, by a factor of α. α is maximum initially, then gradually decreases, and becomes zero at the last epoch. On the other hand, the loss obtained for real labels and generated labels, both conditioned for real data is computed by a factor of β. This β is zero at the beginning of VL-GAN model training, then gradually increases and gets maximum at the end. For larger number of epochs, the adaptive rate is slow. When the fingerprints of different locations hold close RSSI pattern generator needs larger time to

win. In such cases total number of epochs is also larger and thus, adaptive rate of loss is slow. The sum of α and β is always 1. When total number of epochs is $\boldsymbol{E}$, at each of the current epoch $\boldsymbol{e}$, $\alpha = \frac{\boldsymbol{E}-\boldsymbol{e}}{\boldsymbol{E}}$ and $\beta = \frac{\boldsymbol{e}}{\boldsymbol{E}}$. The discriminator loss can be expressed as:

$$\begin{aligned} D &= \frac{\alpha}{2} \times \{D_{real}^{gen} + D_{gen}^{gen}\} + \frac{\beta}{2} \times \{D_{real}^{real} + D_{gen}^{real}\} \\ &= \frac{\alpha}{2} \times [E[logP(S = gen|[X_G, L_{RR}])] \\ &\quad + [E[logP(S = gen|[X_G, L_{GG}])]] \\ &\quad + \frac{\beta}{2} \times [E[logP(S = real|[X_R, L_{RR}])] \\ &\quad + [E[logP(S = gen|[X_R, L_{RG}])]] \end{aligned} \tag{6.3}$$

In equation 6.3, discriminator loss D is represented by–

D_{real}^{gen}– Loss to discriminate synthetic data with real label

D_{real}^{real}– Loss to discriminate real data with real label

D_{gen}^{real}– Loss to discriminate real data with synthetic label

D_{gen}^{gen}–Loss to discriminate synthetic data with synthetic label

X_G – Generated synthetic data

X_R – Real data

L_{RR} – Real labels in vectored form for real data

L_{RG} – Generated labels in vectored form for real data

L_{GG} – Generated labels in vectored form for generated data

The classifier generates labels for the data, and for this the data must follow the original distribution pattern of any class. Otherwise, the classifier will generate random vectors with poor values for all class-specific scores, that will be of no use. For this reason, initially more importance is given to the loss obtained for generating labels conditioned to real data and less importance is given to generate labels for synthetic data. With increase in the training time, synthetic data quality is expected to improve. Thus, gradually the importance on different kinds of losses also reverses. The classifier's loss is expressed in equation 6.4.

$$
\begin{aligned}
C &= \frac{\alpha \times C_{gen}^{real} + \frac{\beta}{2} \times C_{real}^{gen} + \frac{\beta}{2} \times C_{gen}^{gen}}{3} \\
&= \frac{1}{3}[\alpha \times [E[logP(S = gen|[X_R, L_{RG}])]] \\
&\quad + \frac{\beta}{2}[E[logP(S = gen|[X_G, L_{RR}])]] \\
&\quad + \frac{\beta}{2}[E[logP(S = gen|[X_G, L_{GG}])]]]
\end{aligned}
\tag{6.4}
$$

In equation 6.4, classifier loss C is represented by–

C_{gen}^{real} – loss for generating labels conditioning real data

C_{gen}^{gen} – loss for generating labels conditioning generated data

C_{real}^{gen} – loss for real labels conditioning generated data

X_G – Generated synthetic data

X_R – Real data

L_{RR} – Real labels in vectored form for real data

L_{RG} – Generated labels in vectored form for real data

L_{GG} – Generated labels in vectored form for generated data

Selection of generated data for augmentation: For each generated data sample obtained from the generator's output, there is a labeled vector obtained in reverse representation, from the classifier's output. After multiplying that labeled vector with -1, the reverse property is removed and the index containing the highest value in that vector is the label for that data, and the highest value indicates the affinity of the generated data to the original data of that label. To ensure improvement in classification accuracy, only those generated data should be selected that have a label with a good tanh score in the labeled vector.

Denormalization : Before augmentation, the synthetic data were denormalized to obtain the original RSSI form equivalent to the tensor generated. Each value R'_v in the tensor is denormalized to R_v following the formula:

$$
R_v = (-1) \times \{\{R'_v \times (|R_{min}| - |R_{max}|)\} + |R_{max}|\} \tag{6.5}
$$

In equation 6.5, R_{max} and R_{min} are the strongest and weakest RSSI values present in the dataset, respectively similar to eqn. 6.1. After denormalization, 0 is mapped to the strongest RSSI value, and 1 is mapped to the weakest RSSI (usually the dummy) value.

The entire method for generating synthetic data with respect to corresponding locations using proposed VL-GAN model is presented in Algorithm 8.

Theorem 1: The equilibrium of U(C,G,D) is achieved if and only if $p(x,y) = p_g(x,y) = p_c(x,y) = p_{gc}(x,y)$.

Proof: Since, we use adaptive loss, initially more weightage is given on the generation of synthetic samples as close as the real ones. As the execution of the algorithm progresses, the focus is more on the generation of labels with respect to both real and synthetic data.

So, for generation of data to be optimal: $P(x,y) = p_g(x,y)$

For the generation of labels to be optimal: $P(x,y) = p_c(x,y)$

We also know that: $P(x,y) = p(x|y)p(y)$

And, $P(x,y) = p(y|x)p(x)$

Thus, for generating label-specific data, we can write :

$P_g(x,y) = p_g(x|y)p(y)$

Again, to generate labels for real or synthetic data samples, $P_c(x,y) = p_c(y|x)p(x)$

So, $p_c(x,y) = p(x,y) \implies P_c(y|x) = p(y|x)$

Similarly, $p_g(x,y) = p(x,y) \implies P_g(x|y) = p(x|y)$

Therefore,

$p_{gc}(x,y) = p_c(x,y)$

$= p_c(y|x)p(x)$

$= p_c(y|x)\frac{p(x,y)}{p(y|x)}$

$= p_c(y|x)\frac{p_g(x|y)p(y)}{p_c(y|x)}$

$= p_g(x|y)p(y)$

$= p_g(x,y)$

$= p(x,y)$ hence, proved.

Algorithm 8: VL-GAN($\boldsymbol{RD}_{[\mathbf{m}\times\mathbf{f}]}$, $\boldsymbol{RL}_{[\mathbf{m}\times l_s]}$)

Input: Real RSSI fingerprint data $\boldsymbol{RD}_{[\mathbf{m}\times\mathbf{f}]}$
Real labels in vectored form $\boldsymbol{RL}_{[\mathbf{m}\times l_s]}$
Batch size $\boldsymbol{b}$
No. of location points l_s
No. of epochs $\boldsymbol{E}$
Output: Generated RSSI fingerprint data $\boldsymbol{GD}_{[(\mathbf{b}\times\mathbf{E})\times\mathbf{f}]}$
Generated vectors each containing label-specific tanh-scores $\boldsymbol{GL}_{[(\mathbf{b}\times\mathbf{E})\times l_s]}$

initialize **Gendata** an empty dataset to store generated RSSI values
initialize **Genlabels** an empty dataset to store tanh scores wrt labels for each generated RSSI vector
for $\boldsymbol{e}$ in 1 to $\boldsymbol{E}$
  create noise $\boldsymbol{N}_{[\mathbf{b}\times\mathbf{f}]}$ for RSSI data
  $\boldsymbol{GD}_{[\mathbf{b}\times\mathbf{f}]} \leftarrow$ Generator ($\boldsymbol{N}_{[\mathbf{b}\times\mathbf{f}]}$, $\boldsymbol{RL}_{[\mathbf{b}\times l_s]}$)
  $\boldsymbol{GL}_{[\mathbf{b}\times l_s]} \leftarrow$ Classifier ($\boldsymbol{RD}_{[\mathbf{b}\times\mathbf{f}]}$)
  D^{real}_{real}=Discriminator ($\boldsymbol{RD}_{[\mathbf{b}\times\mathbf{f}]} \times \boldsymbol{RL}_{[\mathbf{b}\times l_s]}$)
  D^{gen}_{real}=Discriminator ($\boldsymbol{GD}_{[\mathbf{b}\times\mathbf{f}]} \times \boldsymbol{RL}_{[\mathbf{b}\times l_s]}$)
  D^{real}_{gen}=Discriminator ($\boldsymbol{RD}_{[\mathbf{b}\times\mathbf{f}]} \times \boldsymbol{GL}_{[\mathbf{b}\times l_s]}$)
  D^{gen}_{gen}=Discriminator ($\boldsymbol{GD}_{[\mathbf{b}\times\mathbf{f}]} \times \boldsymbol{GL}_{[\mathbf{b}\times l_s]}$)
  $\alpha = \frac{\boldsymbol{E}-\boldsymbol{e}}{\boldsymbol{E}}$
  $\beta = \frac{\boldsymbol{e}}{\boldsymbol{E}}$
  //discriminator loss
  D= $\frac{\alpha}{2} \times\{ D^{gen}_{real} + D^{gen}_{gen}\} + \frac{\beta}{2} \times\{ D^{real}_{real} + D^{real}_{gen}\}$
  //generator loss
  G=Discriminator ($\boldsymbol{GD}_{[\mathbf{b}\times\mathbf{f}]} \times \boldsymbol{RL}_{[\mathbf{b}\times l_s]}$)
  $\boldsymbol{GL}_{\mathbf{real}[\mathbf{b}\times l_s]} \leftarrow$ Classifier ($\boldsymbol{RD}_{[\mathbf{b}\times\mathbf{f}]}$)
  $\boldsymbol{GL}_{\mathbf{gen}[\mathbf{b}\times l_s]} \leftarrow$ Classifier ($\boldsymbol{GD}_{[\mathbf{b}\times\mathbf{f}]}$)
  $C^{gen}_{real} \leftarrow$ Discriminator ($\boldsymbol{GD}_{[\mathbf{b}\times\mathbf{f}]} \times \boldsymbol{RL}_{[\mathbf{b}\times l_s]}$)
  $C^{real}_{gen} \leftarrow$ Discriminator ($\boldsymbol{RD}_{[\mathbf{b}\times\mathbf{f}]} \times \boldsymbol{GL}_{[\mathbf{b}\times l_s]}$)
  $C^{gen}_{gen} \leftarrow$ Discriminator ($\boldsymbol{GD}_{[\mathbf{b}\times\mathbf{f}]} \times \boldsymbol{GL}_{[\mathbf{b}\times l_s]}$)
  //classifier loss
  C $\leftarrow \frac{\alpha\times C^{real}_{gen} + \frac{\beta}{2}\times C^{gen}_{gen} + \frac{\beta}{2}\times C^{gen}_{real}}{3}$
  append $\boldsymbol{GD}_{[\mathbf{b}\times\mathbf{f}]}$ to **Gendata**
  append $\boldsymbol{GL}_{\mathbf{gen}[\mathbf{b}\times l_s]}$ to **Genlabels**

6.2.3 VL-GAN Experimental Result

Experiments have been carried out on three benchmark datasets, the details are given in Table 6.2. Each of these three datasets contains data for multiple floors or buildings. From each dataset, one single floor was selected for experimentation so that approximately

Table 6.2: Parametric Details of Indoor Localization Datasets

Datasets	n	s	l_s	f	s_d
Dataset_1[a]	379	38	10	56	11370
Dataset_2[b]	48	8	6	17	1440
Dataset_3[c]	345	43	8	11	17270

[a] https://github.com/renwudao24/SODIndoorLoc
[b] https://doi.org/10.5281/zenodo.7306455
[c] https://doi.org/10.21227/1yd5-rn96
n Total number of location points (classes)
s Number of sets
l_s Number of location points (classes) in each set
f Number of features
s_d Number of samples in dataset

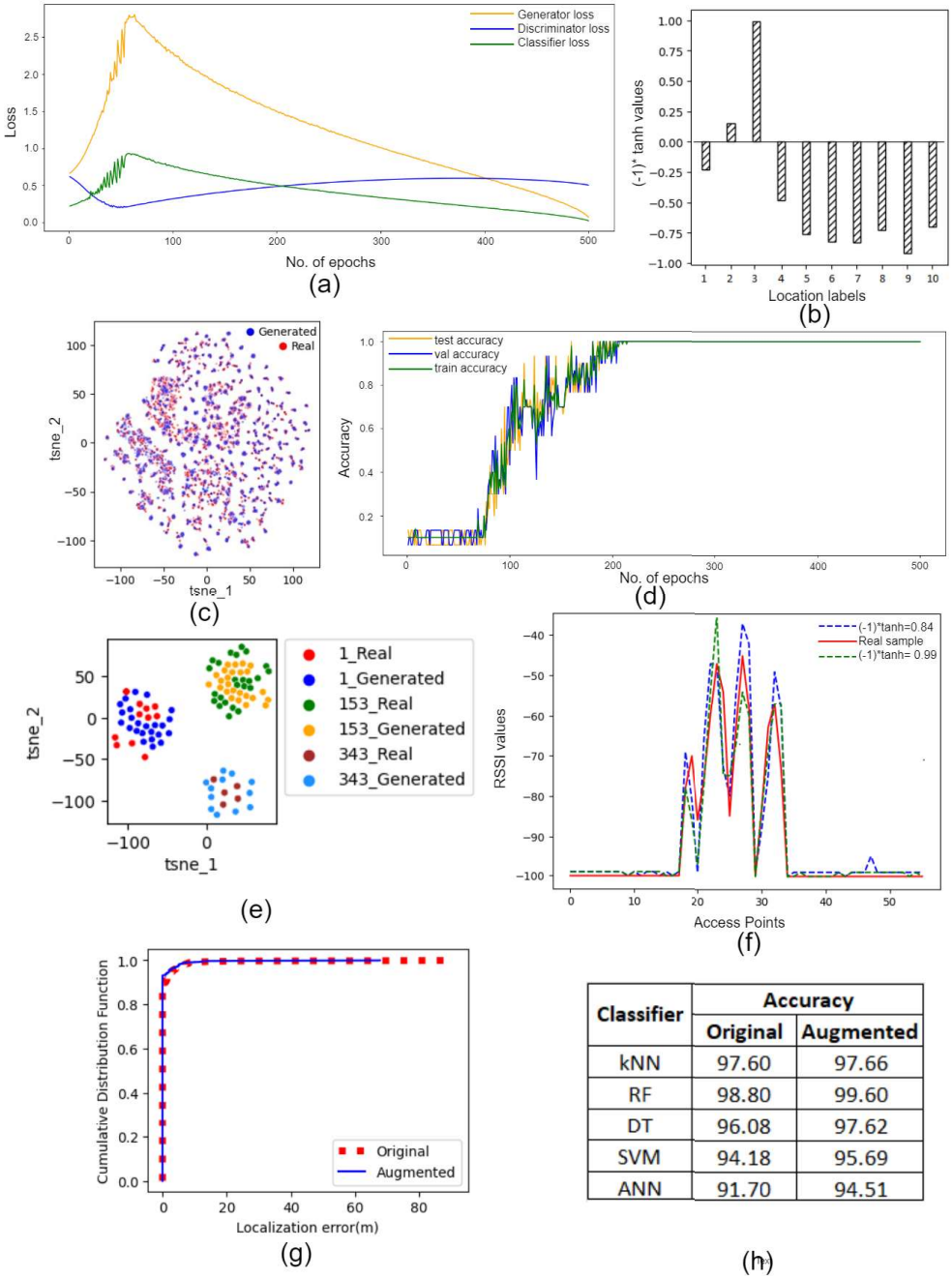
equidistant location points from the same floor could be fed into the proposed SRFG algorithm to generate data in a beneficial way. The proposed algorithms are implemented in Python.

6.2.3.1 Experimental result on Dataset_1

From the Dataset_1 [91], the data from building 2, 4th floor, was considered for experimentation. Data are provided with respect to local (x, y) coordinates in meters. We labeled each unique combination of (x, y) with an integer location id, and a total of 379 labels were obtained, each containing 30 data samples. A total of 11370 samples were divided into 80-10-10 train-test-validation splits in a stratified manner.

Following the SRFG algorithm, the train data was grouped into 38 different sets, each containing 240 samples of 10 labels (except the last set, it contains 9 labels), maintaining an interval of 38. For each set, vectored representation of these 10 labels were obtained using LTV algorithm. Then each set was fed to VL-GAN one at a time to generate new samples. 500 epochs were sufficient to converge the model.

The results of the experimentations conducted for Dataset1 is summarized in Figure 6.12. In Figure 6.12 (a), the loss curves of VL-GAN on a specific set among the 38 sets is represented. The pattern is similar for all 38 sets. Initially, the loss of generator and



Classifier	Accuracy	
	Original	Augmented
kNN	97.60	97.66
RF	98.80	99.60
DT	96.08	97.62
SVM	94.18	95.69
ANN	91.70	94.51

(h)

Figure 6.12: Experimental analysis on Dataset_1; (a) Loss of generator, discriminator & classifier (b) Vectored label (multiplied by -1) obtained as output of classifier for a specific generated data sample (c) 2D tsne representation of complete real and generated data (d) Accuracy of the classifier on train, test and validation stablizes at highest accuracy after 220 epochs (e) Labelwise 2D tsne representation of real and generated data (f) Generated data points for a specific label has maximum and minimum of reversed tanh value of 0.99 & 0.84 respectively. Both patterns are presented with dotted curve (g) CDF of localization error when classifying with ANN for real & augmented data. (h) Comparision of loalization accuracies for real and augmented data

classifier (which is also a generator) increases up to epoch 75 approximately, and the loss of discriminator decreases for the same time. It reflects the basic property of the zero-sum game of GAN, i.e. initially the generator generates random noises as data samples, and it is easy for the discriminator to distinguish real and generated samples. After this point, both the generators start to generate better samples gradually and it becomes a little difficult for the discriminator to distinguish between real and generated samples. As a result, a scissor shape is found between the classifier-discriminator pair with the crossing-point at epoch 200, and between the generator-discriminator pair with the crossing point at 400. Exactly at which epoch these crossing points will be found depends on the intra-class similarity and inter-class dissimilarities between RSSI vectors of each set. Thus, this stabilization indicator epoch varies for different sets of the same dataset. For Dataset_1, maximum of 400 epochs were needed to reach stabilization in loss considering all the sets.

From the crossing point of classifier-discriminator pair, the accuracy of the classifier stabilizes as can be seen in Figure 6.12 (d). This stabilization nature is observed for all the sets between epoch 200 to 300, when VL-GAN is executed for a total of 500 epochs. The set of 200 to 300 samples (1 sample per epoch) generated after stabilization of VL-GAN are considered as the generated synthetic data.

For each generated sample the classifier's output is a vector of length 10 each element representing the RSSI characteristic wise affinity or distance of that sample from original data of each label, in a reverse way. The label-specific scores are obtained from tanh activation function, thus lies within the range of -1 to 1. Which is then reversed by multiplying with -1. Such reversed tanh scores for one data sample is represented in Figure 6.12 (b). The reverse property indicates that, 1 represents highest affinity. It can be observed that, the scores for label 1 and 2 are very close to zero, which represents that the particular generated sample is neutral by nature to label 1 and 2. Similarly, tanh scores for label 4 to 10 are close to -1, that represents strong dissimilarities. Finally, a value very close to 1 ensures that the generated sample belongs to label 3. In this way, the maximum score for each generated sample is considered to identify its label.

For the generated samples of VL-GAN, the scores of the label vector lies within (0.70,0.99). Figure 6.12 (f) compares the pattern of one original RSSI data sample having

two generated samples with highest and lowest reversed tanh scores for that particular label. This figure shows the RSSI values received by a device at a location point from the different APs. Visibly, this location point is closer to the three APs from which it gets strong signals indicated by the three peaks. It is observed that the generated sample with a score 0.99 follows the original pattern more closely than the generated sample with score 0.84. For this reason, generated samples that have a score for the corresponding generated label equal to or greater than 0.99 were considered for augmentation only.

The t-SNE distribution of complete real and generated data in two dimensional space is represented in Figure 6.12 (c). The distribution of generated blue data points closely follows the real data points. Label-wise t-SNE distributions for real and generated data are also analyzed for three locations with six visibly distinguishable shades in Figure 6.12 (e). Generated data samples of each label are positioned close to the positions of their real data of same label. Finally, all labeled generated data from 38 sets having generated label with a score of minimum 0.99 were augmented to real data. 4086 generated samples were found with score equal to or greater than 0.99. This portion of data were appended to the original 9096 train data samples. Localization accuracies were obtained with different ML classifiers before and after augmentation of train data, as shown in Figure 6.12 (h). Here, the test data was the remaining 20% real data (that was kept separate all throughout).

Maximum hike in accuracy was found for ANN, and CDF plot of localization error for the same classifier is represented in Figure 6.12 (g). Distance between each pair of location points was obtained by Euclidean distance of their coordinates. The CDF curve indicates that majority of the location points were correctly predicted with a localization error of 0 meter, for both real and augmented data. However, there were very few location points that were wrongly classified with a localization error of upto 80 meters for real data, and this error is reduced by 10 meters after augmentation of generated data.

6.2.3.2 Experimental result on Dataset_2

From Dataset_2 [24], data from layout 1, 1^{st} floor, was considered for experimentation.

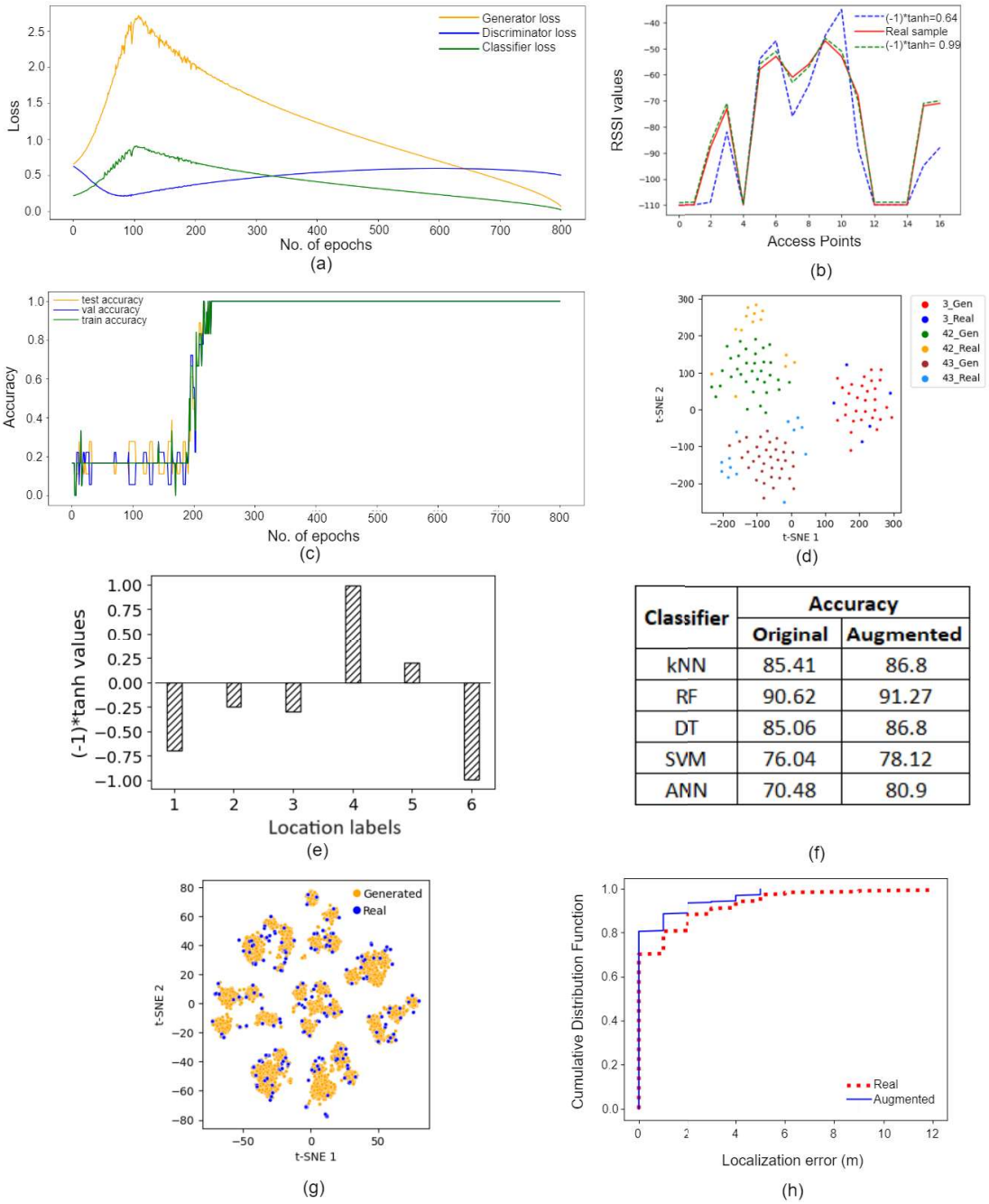


Figure 6.13: Experimental analysis on Dataset_2; (a) Loss of generator, discriminator and classifier (b) Generated data points for a specific label has a reversed tanh value of maximum 0.99 and minimum 0.64. Both patterns are represented with dotted curve (c) Accuracy of the classifier on train, test and validation stablizes at highest accuracy after 240 epochs (d) Labelwise 2D tsne representation of real and generated data (e) Vectored label (multiplied by -1) obtained as output of classifier for a specific generated data sample (f) Comparision of loalization accuracies for real and augmented data (g) 2D tsne representation of complete real and generated data (h) CDF of localization error when classifying with ANN for real and augmented data.

Data were collected by researchers from 42 BLE sources and 17 WiFi sources. BLE signal was found to be similar over a large region covering multiple location points compared to WiFi - thus, showing very less location sensitivity. So, to reach at a generalized conclusion from three datasets, we considered only WiFi sources for our experiments. RSSI values were provided with respect to local (x, y, z) coordinates in meters. We labeled each unique combination of (x, y, z) with an integer location id, and a total of 48 labels were obtained, each containing 30 data samples. A total of 1440 samples were divided into 80-10-10 train-test-validation splits in a stratified manner. The full dataset was divided into 8 sets by the SRFG algorithm, each containing 6 labels having an interval of 8. After converting the labels into corresponding vector format using LTV algorithm, each set was fed to VL-GAN one at a time. We experimented with 800 epochs since the distance between consecutive location points in a set is comparatively small, causing similarities between fingerprints of different labels.

In Figure 6.13 (a), the loss curves of VL-GAN of one single set is represented. The pattern is found to be similar for all the sets. Like Dataset_1, the basic property of the zero-sum game is reflected by the scissor-shaped curve between generator-discriminator and classifier-discriminator pairs. The peaks of the generator and classifier loss curve are approximately at 2.75 and 0.8 respectively near epoch 100. Upto this point, the game is in discriminator's hand and noise is generated as data and/or label. After that, the slope is downward and a quick improvement in accuracy (Figure 6.13 (c)) is observed between epoch 200 and 300, and the accuracy becomes stable from then to the last epoch. This fast stabilization is true for all the 8 sets, although the stablized accuracy curve fluctuates very little (roughly between 0.95 to 1.0) for some sets. Thus, for a larger

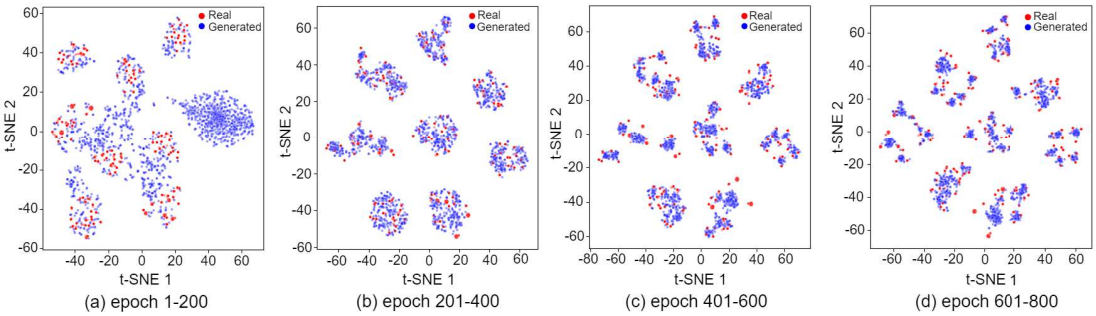


Figure 6.14: Improvement of generated data quality during training of VL-GAN on Dataset_2

number of epochs the model generates acceptable data and labels and the large quantity of generated data can be observed in Figure 6.13 (g). Generated data are also close to the real data distribution, and this property holds for location-specific distribution as well (Figure 6.13 (d)). Vectored representation of generated label for a specific RSSI vector that is considered for augmentation is shown in Figure 6.13 (e). Here, the assigned label is found to reach a reversed tanh score of 1 indicating good quality of generated data.

How this quality can vary, is shown in Figure 6.13 (b). The generated data of a specific location with a reversed tanh score of 0.99 follows the original RSSI curve very closely. Although the curves for real and generated data differ at some points by a few dBms, real pattern is highly maintained. However, the other generated RSSI vector with lower score greatly differs from real pattern. RSSI values vary up to 15 dBm approximately, near AP6. When all generated data having a label with a reversed tanh score ≥ 0.99 are augmented, a hike in localization accuracy is observed for all five ML classifiers, while ANN reports maximum improvement of around 10% (Figure 6.13 (f)). For ANN, CDF is computed for real data and augmented data as shown in Figure 6.13 (h). After data augmentation, localization error is reduced by 7 meters, and percentage of samples with zero or smaller localization error has been increased.

Vectored representation of real labels for one single set of Dataset_2 is represented in Figure 6.15. To visualize the relation among the data of these six classes, their two principal components are obtained by PCA and plotted. There are six vectors that correspond to one of the locations. In each vector L_i, the j^{th} position represents the distance between the centroids of i^{th} and j^{th} classes obtained using Algorithm 7. In vector L_i, the i^{th} value is -1, representing strong affinity to the centroid of i^{th} class. The class L_5 is visually distant from other classes and for this, it can be observed that 5^{th} position of all vectors (except L_5) holds a positive value near to 1 representing large distance from centroid of the class L_5.

How the quality of generated data is improved over the training period of VL-GAN is shown through TSNE plots in Figure 6.14. It can be observed that, at the beginning of training at epochs 1–200, the generated data points are basically noise and do not have

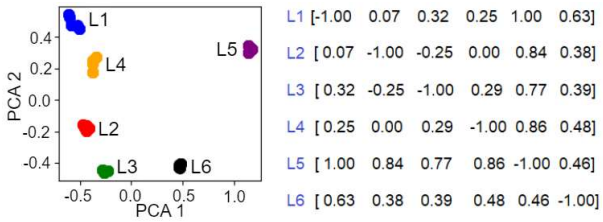


Figure 6.15: Vector representation of real labels Dataset_2

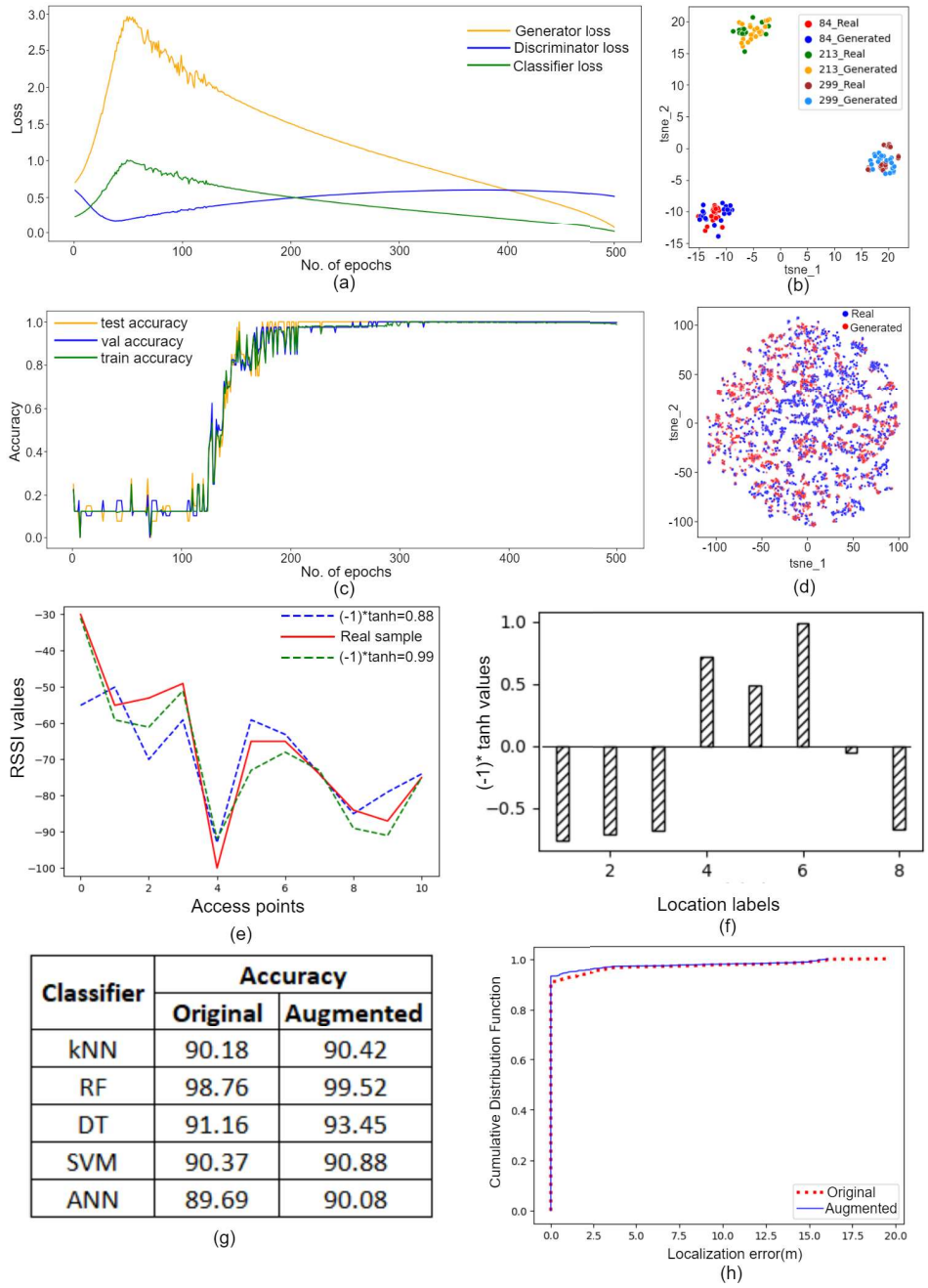
any similarity with the distribution of real data. Between the range of epochs 201–400, generated data started to fall within the bunch of real data, and no more sparse synthetic data was observed. The distribution of generated data becomes more and more similar to that of the real data in epochs 401–600 (Figure 6.14 (c)) and 601–800 (Figure 6.14 (d)), The difference between these two visualization is very small compared to the difference between Figure 6.14 (a) and (b). Thus, the data quality improves sharply at the beginning of the training and gradually becomes stable with more epochs.

6.2.3.3 Experimental result on Dataset_3

In Dataset_3 [92], WiFi fingerprints were collected from 6 APs, where 5 of them provide two different MAC addresses for communications channels of 2.4-GHz and 5-GHz, resulting in a total 11 number of features. Among available data files, we have used Nexus 5 data in this work for experimentation.

A total of 345 location labels were obtained, where most of the labels contain 50 data samples. A total of 17270 samples were divided in a stratified manner into 80-10-10 train-test-validation splits. 43 sets are obtained by the SRFG algorithm, each containing 8 labels having an interval of 43, except the last set that contains 9 labels. After converting the labels into corresponding vector format using LTV algorithm, each set was fed to VL-GAN one at a time, and, experimented with 500 epochs.

In Figure 6.16 (a), the loss curves of VL-GAN of one single set is represented. The pattern is found to be similar for all the sets. The basic property of the zero-sum game is reflected by the scissor-shaped curve between generator-discriminator and classifier-



Classifier	Accuracy	
	Original	Augmented
kNN	90.18	90.42
RF	98.76	99.52
DT	91.16	93.45
SVM	90.37	90.88
ANN	89.69	90.08

(g)

Figure 6.16: Experimental analysis on Dataset_3; (a) Loss of generator, discriminator and classifier (b) Labelwise 2D tsne representation of real and generated data (c) Accuracy of the classifier on train, test and validation stablizes at highest accuracy after 240 epochs (d) 2D tsne representation of complete real and generated data (e) Generated data points for a specific label has maximum reversed tanh value of 0.99 and minimum value 0.88. Pattern for both are represented with dotted curve (f) Vectored label (multiplied by -1) obtained as output of classifier for a specific generated data sample (g) Loalization accuracies for real and augmented data (h) CDF of localization error when classifying with ANN for real and augmented data.

discriminator pairs. The peaks of the generator and classifier loss curves are approximately at 3 and 1 respectively near epoch 50. After this point, the classifier and the generator are found to gradually improve their performance in learning the real pattern. The improvement in accuracy (Figure 6.16 (c)) is slow but along a steady slope.

Approximately, from epoch 225 the accuracy becomes stable near 1 and from this time generated data are stored. The generated data follow the original data distribution pattern as a whole and also label-wise, as can be observed in Figure 6.16 (d) and 6.16 (b) respectively.

Since the data were collected by the authors [92] following a close loop pattern, and labels are assigned by us for unique combination of (x,y), there can be a chance that in few sets obtained by SRFG algorithm, two or more labels are physically close location wise. In that case, generated data will have a vectored label in which two or more label-specific scores have strong likelihood as shown in Figure 6.16 (f). However, the strongest representitive label (6^{th}) is selected in such cases. Like previous datasets, in this case also the generated RSSI samples with reversed tanh score 0.99 and above follows the real RSSI pattern as shown in Figure 6.16 (e). When five ML classifiers were trained with real data and augmented data (on or above score 0.99), it can be seen in Figure 6.16 (g) that accuracy is increased for all the classifiers. Maximum accuracy is increased for decision tree, and for the same classifier CDF curve is plotted in Figure 6.16 (h). It can be observed that localization error is reduced around 3 meters after augmentation.

6.2.4 Comparison of VL-GAN with existing models

Performance of the proposed VL-GAN model has been compared with CGAN [89] and DeepFi[93] models on Dataset_2. VL-GAN works as a synthetic data generator and also as a classifier. Localization accuracy achieved by VL-GAN on Dataset_2 is 88.19%. However, since CGAN and DeepFi works as only synthetic data generators, performance of these three models have been compared only in terms of data augmentation.

CGAN is a variant of GAN where data is generated with respect to a provided condition, which is nothing but the class label. DeepFi is an Autoencoder-like structure,

where two neural networks encoder and decoder are used to learn the data representation using feature compression. For both models, the entire real data were divided into two parts. 80% data were used to train CGAN and DeepFi models and rest 20% data were kept for testing. To reach a generalized conclusion regarding data augmentation, before feeding the training data as input to CGAN and DeepFi models, it went through the same preprocessing method applied for VL-GAN.

After the models are trained for 800 epochs, synthetic data were generated which is half the amount of the 80% real training data, maintaining the class proportion. The generated data is augmented with the real training data, and four supervised ML classifiers were used to compute the localization accuracy as shown in Table 6.3. It is observed that, augmenting data generated by CGAN and DeepFi can increase localization accuracy, but when synthetic data generated by VL-GAN are augmented, the improvement in accuracy is maximum.

Table 6.3: Performance comparison of the proposed VL-GAN with CGAN and DeepFi models

Classifier	**Real Data**	**Augmented (VL-GAN)**	**Augmented (CGAN)**	**Augmented (DeepFi)**
kNN	85.41	86.80	85.76	85.42
Random Forest	90.62	91.27	90.62	90.97
Decision Tree	85.06	86.80	85.76	85.41
ANN	70.48	80.90	75.69	76.73

6.2.5 Discussion

Analyzing the thorough experimentation on three benchmark datasets some important observations are discussed as follows.

1. The proposed VL-GAN model is capable of generating labels for synthetic as well as real unlabeled data. How accurately labels are assigned to real unlabeled data can be understood from Figure 6.12(d), 6.13(c), and 6.16(c). On or before the half of the training period, the accuracy is more than 95% and this accuracy remains consistent throughout

the next epochs. This reflects the certainty of label predictions for real unlabeled data samples.

2. The adaptive loss updation resists the chance of overfitting. In Figure 6.12(a), 6.13(a) and 6.16(a) it is observed that the losses of both the generator and classifier sharply increase at the beginning, then start to decrease, and finally the decrement becomes stable and consistent. The improvement in learning is stable, there is no sudden fall or peak in the curve, only a consistent slope is observed.

3. The quality of generated samples is indicated by reversed tanh score T, which is obtained based class centroids. One can select a threshold T_h to select generated sample for augmentation. This threshold can be decided based on the intra-class RSSI similarities. When a single location point covers a comparative larger area, there will be many data points that are distant from the centroid of that class by RSSI characteristics. In that case, a synthetic data with comparatively lower T will be a little distant from the centroid of assigned class, but that will not violate the real data property rule. In this paper, all three benchmark datasets contains data of consecutive coordinates meters apart. Thus, a high T_h has been choosen to augment best quality synthetic data, not violating real data distribution rules.

4. Fingerprint data may contain noise. The noises are usually small in amount and distant from class centroids. The synthetic data with a good reversed tanh score is near the centroid of the assigned class. Thus, the chance of generating noise is negligible. It refers to the fact that after augmentation, accuracy will greatly increase for those data points with higher noise levels or whose class-specific data points contain varying RSSI patterns if each data point is given equal importance. For example, in Figure 6.13(f), the lowest accuracy obtained is 70.48% for ANN, whereas other classifiers obtain a higher accuracy. The reason is that in Dataset_2, the class-specific similarities between RSSI vectors are lesser in comparison to Dataset_1 and Dataset_3. kNN classifies based on distance among data points, and tree-based classifiers classify based on feature-oriented decisions one by one, where ANN classifies by initially assigning some weights to each data point. If noises

are significant in amount in the training data, there is a high chance these get significant weight repeatedly, resulting in comparatively poor accuracy. In such a case, generating synthetic data will result in a good accuracy improvement. For the very same reason, after data augmentation, classifiers fail in fewer cases where the true location and predicted location are physically near and hold similar RSSI characteristics. This, in turn, reduces the localization error.

6.3 Summary

In this chapter, the location-specific data generation process has been investigated in an qualitative manner. A three-player generative model, VL-GAN, is proposed that is capable of generating synthetic RSSI fingerprint data along with its label in vectored format. The vectored representation provides an understanding of how much the data relates to that class and how much it differs from other classes. During training, loss is computed in an adaptive way, which results in the expected scissor-shaped curve of any GAN-based model, indicating the clear winning of both generators opposing the discriminator. The proposed SRFG algorithm makes the whole process of fingerprint generation faster and simpler. Among all synthetic data, a portion is selected based on a threshold value indicating the quality of the generated data. This selection ensures improvement in localization accuracy and reduction of localization error when classified with five ML classifiers on three benchmark datasets from three different countries. This reflects the generalization of the proposed method. However, the number of generated data points for each class is not equal and cannot be predicted. We report maximum 10% improvement in localization accuracy for Dataset_2 using ANN, and localization error was reduced up to 20 meters on Dataset_1 as measured by CDF.

Published Works

[1] Manjarini Mallik, Chandreyee Chowdhury, Characteristic analysis of fingerprint datasets from a pragmatic view of indoor localization using machine learning approaches. J Super-

comput 79, 18507–18546 (2023). doi:10.1007/s11227-023-05386-x

[2] Manjarini Mallik, Soumyajit Chakraborty, Kalyan Sasidhar, Chandreyee Chowdhury, VL-GAN: A generative classification approach for fingerprint-based indoor localization, Expert Systems with Applications, Volume 290, 2025, 128400, ISSN 0957-4174, doi:10.1016/j.eswa.2025.128400.

Chapter 7

Conclusion and Future Scope

Indoor localization systems (ILS) aim to determine a user's location within an indoor space, where satellite signals are unavailable. Indoor-available sensors like WiFi and BLE are used in fingerprint-based ILS. Since WiFi and BLE infrastructures are built for communication purpose, these sources are present in non-uniform manner over any indoor area. Thus, the perimeter of the sub-region in which user's presence can be defined strongly varies according to the availability of effective fingerprint in that sub-region. In this thesis, we have investigated the impact of location granularity on localization performance from various perspectives. In previous chapters, solutions have been proposed to address various research challenges related to this matter, like scarcity of labeled data, fluctuations of signals, instance hardness and data annotation efficiency. Different supervised and unsupervised approaches are proposed like ensemble of clustering, signal to image encoding of fingerprints, asymmetric granularity investigation analyzing instance hardness, and synthetic data generation using vectored labeled GAN.

The overall research findings of this thesis are presented in this chapter. The open research scopes are also discussed in different directions.

7.1 Summary

Here is a summarized version of the thesis presented along with the related findings and conclusions.

• Machine learning and deep learning based algorithms are proposed in the domain of location granularity and synthetic data generation.

• Semi supervised learning algorithms along with statistical analysis have been used to obtain asymmetric coarse grained locations over an indoor space. Ensemble of clustering ensures to groupify physically consecutive fine locations in same sub-region.

• Asymmetric location granularity due to non-uniform AP distribution has been investigated from both perspectives– fine grained and coarse grained granularity levels. To investigate granularity optimization, a dataset MetroIndoorLoc is collected using five android devices from two metro stations, one at under-ground level and another at-grade level.

• Class imbalance has been reduced to improve localization accuracy, generating synthetic training data samples using CGAN. A balance is maintained between generated and real data sample count per class using statistical analysis.

• A new GAN variant, namely, VL-GAN is proposed to generate synthetic data as well as synthetic labels, and also quantify the relativeness of the assigned labels to the data samples.

• All of the proposed approaches are evaluated on multiple benchmark datasets to get generalized insight regarding their performance.

• The objective of this thesis is to achieve precise indoor localization utilizing existing WiFi/BLE infrastructures. To achieve this, two primary challenges - non-uniform distribution of APs and scarcity of labeled training data have been addressed.

7.2 Summary of contributions

In this work, each chapter is organized to improve on the information presented in its previous chapter, and/or to address the research gap left over in previous chapters. The chapters are summarized as follows.

• Fingerprint-based indoor localization stands on the primary assumption that, it is feasible to collect sufficient labeled ground truth data from various public indoor places. Often these collected data are grossly labeled to avoid laborious site surveys, i.e. with respect to some distant static reference points. In such case, it is difficult to draw the boundary of each distinct location, and specify the level of granularity. In chapter 3, an ensemble of clustering approach is applied to address this challenge and groupify related data (fingerprint characteristic wise) into varying granularity levels, according to the distribution of APs. However, this raises a challenge to cluster the collected data that are really physically close. There may be some data points within a cluster that are physically positioned far away. Also, the count and instances of these wrongly clustered data differ for different base clustering algorithms. Proposed ensemble of clustering addresses this research gap. The system is evaluated with three benchmark datasets for WiFi based indoor localization. For the first dataset, JUIndoorLoc, 94% to 99% localization accuracy is achieved for individual supervised classifiers. For the second and third datasets, the ranges of obtained localization accuracies are 96% to 99% and 95% to 98% respectively.

• Existing research shows that fingerprint based indoor localization performance is greatly impacted by the fluctuations of WiFi signals. This often limits the precision of localization up to coarse-granularity level, when a significant number of precise areas lack stable fingerprints. In chapter 4, a two-phase approach is proposed in which, in the first phase user's location is obtained in terms of coarse-granularity level, and in the second phase user's fine-grained location is obtained. An algorithm, namely, Image Encoding using RSSI Fusion (IERF), is proposed to generate RSSI images that emphasize the location-specific consistent RSSI glyph and de-emphasize the less frequent RSSI patterns. A lightweight CNN model CurveNet is proposed to predict locations classifying these encoded images. The performance of the proposed approach is evaluated using two benchmark datasets. With encoded images using the IERF algorithm and applying CurveNet for location prediction, around 99% accuracy is achieved, which is 5% better compared to 1D RSSI-based location prediction.

• In chapter 5, the impact of multi-level asymmetric granularity has been investigated, to deal with the challenge of non-uniform AP distribution. A new public indoor dataset, MetroIndoorLoc, is provided that consists of two metro station datasets: one at-grade level and another underground level. Experimental analysis of MetroIndoorLoc using different ML and DL classifiers ensures the feasibility of fingerprint-based localization in such areas. Moreover, the algorithm GO-kDN is proposed that updates the granularity level of any fingerprint dataset based on instance hardness analysis to improve localization accuracy. GO-kDN has been executed on MetroIndoorLoc and two other benchmark datasets, and in each case, significant improvements in localization accuracy and depletion in localization error were observed. Proposed approach is also found to perform in an impressive way when compared with another recent state-of-the-art indoor localization method. GO-kDN has been executed on MetroIndoorLoc and two other benchmark datasets. Significant improvement of localization accuracy was observed in each case, 15% for at-grade level metro station and 35% for underground metro station.

•It is crucial to address the problem of laborious site-surveys for precise data annotation, for both fine and/or coarse level granularity. To address the challenge of limited labeled data, in this chapter a Vectored labeled Generative Adversarial Network (VL-GAN) model is proposed. It introduces a third component, a vectored label generator acting as a classifier to the traditional GAN model. This chapter makes three contributions: (i) a Label-to-Vector algorithm that transforms the location labels into distance vectors denoting the likelihood of a sample belonging to each location point (class); (ii) a VL-GAN model that consists of a generator, a discriminator, and a classifier (a label generator); (iii) a Set-wise RSSI Fingerprint Generation (SRFG) strategy to ensure higher-quality data generation even for fine-grained locations. We report maximum 10% improvement in localization accuracy for Dataset_2 using ANN, and localization error was reduced up to 20 meters on Dataset_1 as measured by CDF. In the future, the concept of VL-GAN can be extended to multi-modal application domains. Furthermore, we aim to investigate if the model needs to be retrained for new environments or if a pre-trained model can be used instead. In this work, the proposed model is experimented on WiFi data. However, other sensors like Bluetooth and magnetometer data have a significant contribution in the

domaine of indoor localization. In the future, VL-GAN is aimed to be evaluated on other sensor-fusion based datasets.

7.3 Future Research directions

This work investigates the impact of different granularity levels on localization performance and addresses the challenge of scarce labeled data. Through this analysis, several key insights have emerged, highlighting potential avenues for future research, presented as follows.

7.4 Data distillation in RSSI fingerprint datasets using autoencoder

Some of the public indoor spaces contain multistorey buildings or platforms. If we target to develop a common indoor localization system for all of these buildings, it will result in a massive fingerprint dataset. To reduce storage and computational cost of the indoor localization system, dataset distillation approach can be considered. Autoencoder model is widely used for feature compression in various domains, the same logic can be used to compress data instances using transpose of the dataset. In this way, we can obtain a smaller and compressed version of the real dataset, retaining most of the learning potential of original dataset. Compressing the majority classes more than the minority classes could help to reduce class imbalance also.

An asymmetric autoencoder model can be utilized for generating synthetic distilled data. The general structure of an autoencoder with a single hidden layer y, input real data x and output synthetic data $\hat{x}$ having an encoder E(x) and decoder D(y) can be defined as follows.

$$\mathrm{y} = \mathrm{E(x)}$$

$$\hat{x} = \mathrm{D(y)}$$

or,

$$\mathrm{f(x) = D(E(x))}$$

The input x is a vector of real numbers with n components, and $\mathbb{R}^k$ is a lower-dimensional compressed representation with k features. $\mathbb{R}^m$ reconstructed space, that has the same dimension as the original input.

$$\mathbf{x} \in \mathbb{R}^n$$

$$\mathrm{E(x)}\colon\ \mathbb{R}^n \rightarrow \mathbb{R}^k\ [\mathrm{k<n}]$$

$$\mathrm{D(y)}\colon\ \mathbb{R}^k \rightarrow \mathbb{R}^m\ [\mathrm{m=n}]$$

$$\mathrm{f(x)} \approx \mathrm{x\ [full\ reconstruction]}$$

In case of distillation, instead of full reconstruction, data from any previous decoding layer can be taken based on the how much distillation is needed. How the intensity of distillation affects the quality of distilled data requires thorough investigation. Apart from autoencoder, other generative models can also be explored for this purpose.

7.5 Designing unified localization approach for IoT and smartphone based indoor localization

Existing fingerprint based indoor localization research works are mostly focused on smartphone-based user localization, a few existing works attempted to locate assets and robots using IoT devices. A unified localization system for both user and asset tracking can be proposed using both smartphones and IoT devices, to connect these two seemingly different research directions. Since there is no benchmark dataset combining both devices, it would be beneficial to collect real-life data from some IoT modules, along with smartphones of different device configurations. Investigating and analyzing distinguishing patterns from IoT devices, smartphones and their combinations can help to improve localization performance using machine learning and deep learning classifiers.

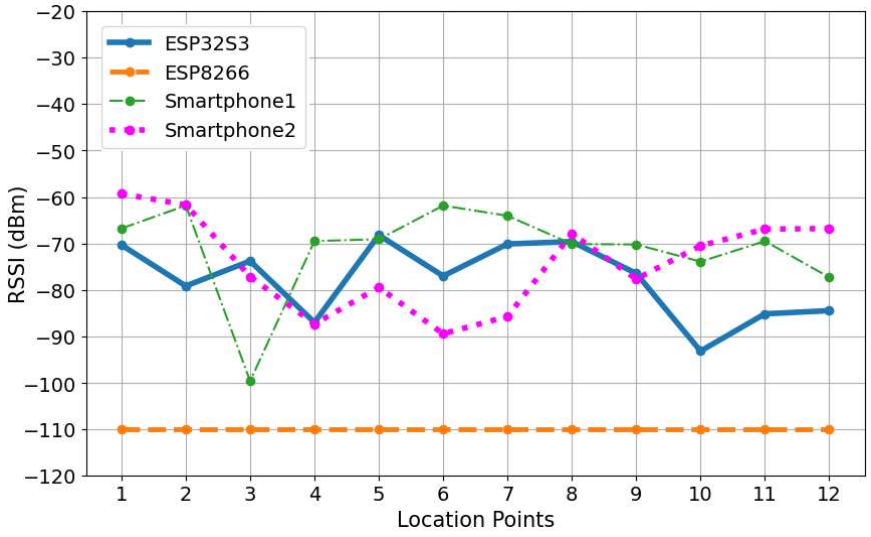


Figure 7.1: Variation of received signal strengths from an AP using different IoT devices at same location points

Table 7.1: Localization accuracy obtained by ML Classifiers for 7 cases, over an experimental region

Case	Train Device	Test Device	GNB	DT	kNN	SVM	RF	ANN	DNN
i	ESP32-S3, ESP8266, Smartphone	ESP32-S3, ESP8266, Smartphone	71.14	99.59	93.09	96.34	**99.70**	96.95	96.85
ii	ESP32-S3, ESP8266, Smartphone	ESP32-S3	81.44	99.50	96.53	97.03	**99.75**	86.39	95.54
iii	ESP32-S3, ESP8266, Smartphone	ESP8266	51.46	**99.99**	86.59	95.53	**99.99**	83.67	96.13
iv	ESP32-S3, ESP8266, Smartphone	Smartphone	94.71	99.03	99.03	99.03	99.17	96.38	**99.30**
v	ESP8266	ESP8266	98.98	**99.95**	92.88	98.22	99.94	97.46	96.51
vi	ESP32-S3	ESP32-S3	97.18	**99.95**	96.24	98.59	99.53	98.59	97.77
vii	Smartphone	Smartphone	98.15	99.63	99.01	99.59	**99.74**	99.27	99.65

From Figure 7.1 it is observed that different IoT devices are differently sensitive in capturing signals from a specific AP. Thus, collecting ground truth data using different IoT devices can capture more patterns that can in turn improve the localization performance at the online phase. In Table 7.1 it is observed that impressive accuracy can be achieved

in unified localization approach. In this case, two smartphones and two IoT devices have been included in the experiment. In future, combination of different IoT devices can be explored in detail.

Accepted Work

[1] Prodipta Chakraborty, Manjarini Mallik, Arkadev Kundu, Chandreyee Chowdhury, Design of a unified indoor localization system integrating IoT devices and smartphone. CCF Transactions on Pervasive Computing and Interaction, Springer.

Chapter 7

Conclusion and Future Scope

Indoor localization systems (ILS) aim to determine a user's location within an indoor space, where satellite signals are unavailable. Indoor-available sensors like WiFi and BLE are used in fingerprint-based ILS. Since WiFi and BLE infrastructures are built for communication purpose, these sources are present in non-uniform manner over any indoor area. Thus, the perimeter of the sub-region in which user's presence can be defined strongly varies according to the availability of effective fingerprint in that sub-region. In this thesis, we have investigated the impact of location granularity on localization performance from various perspectives. In previous chapters, solutions have been proposed to address various research challenges related to this matter, like scarcity of labeled data, fluctuations of signals, instance hardness and data annotation efficiency. Different supervised and unsupervised approaches are proposed like ensemble of clustering, signal to image encoding of fingerprints, asymmetric granularity investigation analyzing instance hardness, and synthetic data generation using vectored labeled GAN.

The overall research findings of this thesis are presented in this chapter. The open research scopes are also discussed in different directions.

7.1 Summary

Here is a summarized version of the thesis presented along with the related findings and conclusions.

• Machine learning and deep learning based algorithms are proposed in the domain of location granularity and synthetic data generation.

• Semi supervised learning algorithms along with statistical analysis have been used to obtain asymmetric coarse grained locations over an indoor space. Ensemble of clustering ensures to groupify physically consecutive fine locations in same sub-region.

• Asymmetric location granularity due to non-uniform AP distribution has been investigated from both perspectives– fine grained and coarse grained granularity levels. To investigate granularity optimization, a dataset MetroIndoorLoc is collected using five android devices from two metro stations, one at under-ground level and another at-grade level.

• Class imbalance has been reduced to improve localization accuracy, generating synthetic training data samples using CGAN. A balance is maintained between generated and real data sample count per class using statistical analysis.

• A new GAN variant, namely, VL-GAN is proposed to generate synthetic data as well as synthetic labels, and also quantify the relativeness of the assigned labels to the data samples.

• All of the proposed approaches are evaluated on multiple benchmark datasets to get generalized insight regarding their performance.

• The objective of this thesis is to achieve precise indoor localization utilizing existing WiFi/BLE infrastructures. To achieve this, two primary challenges - non-uniform distribution of APs and scarcity of labeled training data have been addressed.

7.2 Summary of contributions

In this work, each chapter is organized to improve on the information presented in its previous chapter, and/or to address the research gap left over in previous chapters. The chapters are summarized as follows.

• Fingerprint-based indoor localization stands on the primary assumption that, it is feasible to collect sufficient labeled ground truth data from various public indoor places. Often these collected data are grossly labeled to avoid laborious site surveys, i.e. with respect to some distant static reference points. In such case, it is difficult to draw the boundary of each distinct location, and specify the level of granularity. In chapter 3, an ensemble of clustering approach is applied to address this challenge and groupify related data (fingerprint characteristic wise) into varying granularity levels, according to the distribution of APs. However, this raises a challenge to cluster the collected data that are really physically close. There may be some data points within a cluster that are physically positioned far away. Also, the count and instances of these wrongly clustered data differ for different base clustering algorithms. Proposed ensemble of clustering addresses this research gap. The system is evaluated with three benchmark datasets for WiFi based indoor localization. For the first dataset, JUIndoorLoc, 94% to 99% localization accuracy is achieved for individual supervised classifiers. For the second and third datasets, the ranges of obtained localization accuracies are 96% to 99% and 95% to 98% respectively.

• Existing research shows that fingerprint based indoor localization performance is greatly impacted by the fluctuations of WiFi signals. This often limits the precision of localization up to coarse-granularity level, when a significant number of precise areas lack stable fingerprints. In chapter 4, a two-phase approach is proposed in which, in the first phase user's location is obtained in terms of coarse-granularity level, and in the second phase user's fine-grained location is obtained. An algorithm, namely, Image Encoding using RSSI Fusion (IERF), is proposed to generate RSSI images that emphasize the location-specific consistent RSSI glyph and de-emphasize the less frequent RSSI patterns. A lightweight CNN model CurveNet is proposed to predict locations classifying these encoded images. The performance of the proposed approach is evaluated using two benchmark datasets. With encoded images using the IERF algorithm and applying CurveNet for location prediction, around 99% accuracy is achieved, which is 5% better compared to 1D RSSI-based location prediction.

• In chapter 5, the impact of multi-level asymmetric granularity has been investigated, to deal with the challenge of non-uniform AP distribution. A new public indoor dataset, MetroIndoorLoc, is provided that consists of two metro station datasets: one at-grade level and another underground level. Experimental analysis of MetroIndoorLoc using different ML and DL classifiers ensures the feasibility of fingerprint-based localization in such areas. Moreover, the algorithm GO-kDN is proposed that updates the granularity level of any fingerprint dataset based on instance hardness analysis to improve localization accuracy. GO-kDN has been executed on MetroIndoorLoc and two other benchmark datasets, and in each case, significant improvements in localization accuracy and depletion in localization error were observed. Proposed approach is also found to perform in an impressive way when compared with another recent state-of-the-art indoor localization method. GO-kDN has been executed on MetroIndoorLoc and two other benchmark datasets. Significant improvement of localization accuracy was observed in each case, 15% for at-grade level metro station and 35% for underground metro station.

•It is crucial to address the problem of laborious site-surveys for precise data annotation, for both fine and/or coarse level granularity. To address the challenge of limited labeled data, in this chapter a Vectored labeled Generative Adversarial Network (VL-GAN) model is proposed. It introduces a third component, a vectored label generator acting as a classifier to the traditional GAN model. This chapter makes three contributions: (i) a Label-to-Vector algorithm that transforms the location labels into distance vectors denoting the likelihood of a sample belonging to each location point (class); (ii) a VL-GAN model that consists of a generator, a discriminator, and a classifier (a label generator); (iii) a Set-wise RSSI Fingerprint Generation (SRFG) strategy to ensure higher-quality data generation even for fine-grained locations. We report maximum 10% improvement in localization accuracy for Dataset_2 using ANN, and localization error was reduced up to 20 meters on Dataset_1 as measured by CDF. In the future, the concept of VL-GAN can be extended to multi-modal application domains. Furthermore, we aim to investigate if the model needs to be retrained for new environments or if a pre-trained model can be used instead. In this work, the proposed model is experimented on WiFi data. However, other sensors like Bluetooth and magnetometer data have a significant contribution in the

domaine of indoor localization. In the future, VL-GAN is aimed to be evaluated on other sensor-fusion based datasets.

7.3 Future Research directions

This work investigates the impact of different granularity levels on localization performance and addresses the challenge of scarce labeled data. Through this analysis, several key insights have emerged, highlighting potential avenues for future research, presented as follows.

7.4 Data distillation in RSSI fingerprint datasets using autoencoder

Some of the public indoor spaces contain multistorey buildings or platforms. If we target to develop a common indoor localization system for all of these buildings, it will result in a massive fingerprint dataset. To reduce storage and computational cost of the indoor localization system, dataset distillation approach can be considered. Autoencoder model is widely used for feature compression in various domains, the same logic can be used to compress data instances using transpose of the dataset. In this way, we can obtain a smaller and compressed version of the real dataset, retaining most of the learning potential of original dataset. Compressing the majority classes more than the minority classes could help to reduce class imbalance also.

An asymmetric autoencoder model can be utilized for generating synthetic distilled data. The general structure of an autoencoder with a single hidden layer y, input real data x and output synthetic data $\hat{x}$ having an encoder E(x) and decoder D(y) can be defined as follows.

$$\mathrm{y} = \mathrm{E(x)}$$

$$\hat{x} = \mathrm{D(y)}$$

or,

$$\mathrm{f(x) = D(E(x))}$$

The input x is a vector of real numbers with n components, and $\mathbb{R}^k$ is a lower-dimensional compressed representation with k features. $\mathbb{R}^m$ reconstructed space, that has the same dimension as the original input.

$$\mathbf{x} \in \mathbb{R}^n$$

$$\mathrm{E(x)}\colon\ \mathbb{R}^n \rightarrow \mathbb{R}^k\ [\mathrm{k<n}]$$

$$\mathrm{D(y)}\colon\ \mathbb{R}^k \rightarrow \mathbb{R}^m\ [\mathrm{m=n}]$$

$$\mathrm{f(x)} \approx \mathrm{x\ [full\ reconstruction]}$$

In case of distillation, instead of full reconstruction, data from any previous decoding layer can be taken based on the how much distillation is needed. How the intensity of distillation affects the quality of distilled data requires thorough investigation. Apart from autoencoder, other generative models can also be explored for this purpose.

7.5 Designing unified localization approach for IoT and smartphone based indoor localization

Existing fingerprint based indoor localization research works are mostly focused on smartphone-based user localization, a few existing works attempted to locate assets and robots using IoT devices. A unified localization system for both user and asset tracking can be proposed using both smartphones and IoT devices, to connect these two seemingly different research directions. Since there is no benchmark dataset combining both devices, it would be beneficial to collect real-life data from some IoT modules, along with smartphones of different device configurations. Investigating and analyzing distinguishing patterns from IoT devices, smartphones and their combinations can help to improve localization performance using machine learning and deep learning classifiers.

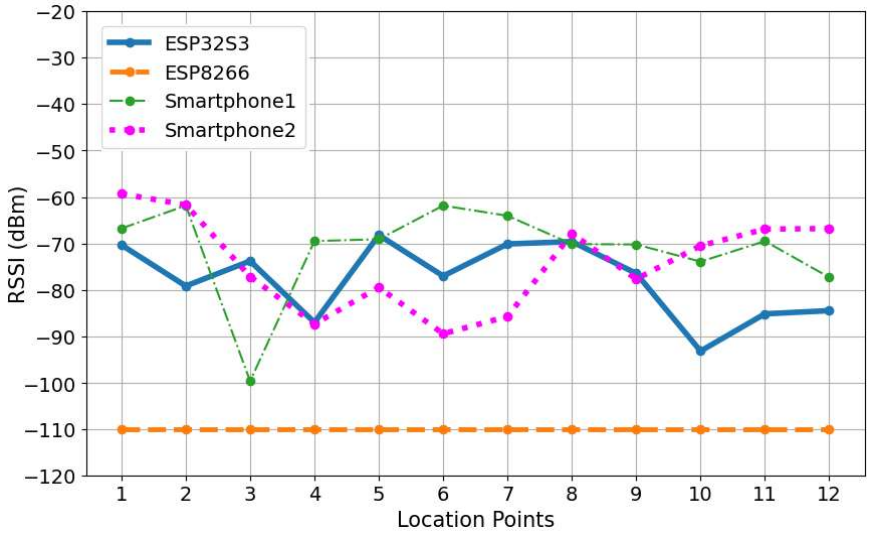


Figure 7.1: Variation of received signal strengths from an AP using different IoT devices at same location points

Table 7.1: Localization accuracy obtained by ML Classifiers for 7 cases, over an experimental region

Case	Train Device	Test Device	GNB	DT	kNN	SVM	RF	ANN	DNN
i	ESP32-S3, ESP8266, Smartphone	ESP32-S3, ESP8266, Smartphone	71.14	99.59	93.09	96.34	**99.70**	96.95	96.85
ii	ESP32-S3, ESP8266, Smartphone	ESP32-S3	81.44	99.50	96.53	97.03	**99.75**	86.39	95.54
iii	ESP32-S3, ESP8266, Smartphone	ESP8266	51.46	**99.99**	86.59	95.53	**99.99**	83.67	96.13
iv	ESP32-S3, ESP8266, Smartphone	Smartphone	94.71	99.03	99.03	99.03	99.17	96.38	**99.30**
v	ESP8266	ESP8266	98.98	**99.95**	92.88	98.22	99.94	97.46	96.51
vi	ESP32-S3	ESP32-S3	97.18	**99.95**	96.24	98.59	99.53	98.59	97.77
vii	Smartphone	Smartphone	98.15	99.63	99.01	99.59	**99.74**	99.27	99.65

From Figure 7.1 it is observed that different IoT devices are differently sensitive in capturing signals from a specific AP. Thus, collecting ground truth data using different IoT devices can capture more patterns that can in turn improve the localization performance at the online phase. In Table 7.1 it is observed that impressive accuracy can be achieved

in unified localization approach. In this case, two smartphones and two IoT devices have been included in the experiment. In future, combination of different IoT devices can be explored in detail.

Accepted Work

[1] Prodipta Chakraborty, Manjarini Mallik, Arkadev Kundu, Chandreyee Chowdhury, Design of a unified indoor localization system integrating IoT devices and smartphone. CCF Transactions on Pervasive Computing and Interaction, Springer.

Bibliography

[1] Prandi, C., Barricelli, B.R., Mirri, S. and Fogli, D., 2023. Accessible wayfinding and navigation: a systematic mapping study. Universal Access in the Information Society, pp.1-28.

[2] Mehta, H., Kanani, P. and Lande, P., 2019. Google maps. International Journal of Computer Applications, 178(8), pp.41-46.

[3] Vecchiato, M., Borasio, N., Scettri, E., Franzoi, V., Duregon, F., Savino, S., Ermolao, A. and Neunhaeuserer, D., 2025. Are Suggested Hiking Times Accurate? A Validation of Hiking Time Estimations for Preventive Measures in Mountains. Medicina, 61(1), p.115.

[4] Tavmen, G., 2020. Data/infrastructure in the smart city: Understanding the infrastructural power of Citymapper app through technicity of data. Big Data & Society, 7(2), p.2053951720965618.

[5] Kunhoth, J., Karkar, A., Al-Maadeed, S. and Al-Ali, A., 2020. Indoor positioning and wayfinding systems: a survey. Human-centric Computing and Information Sciences, 10, pp.1-41.

[6] Delnevo, G., Monti, L., Vignola, F., Salomoni, P. and Mirri, S., 2018, January. AlmaWhere: A prototype of accessible indoor wayfinding and navigation system. In 2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC) (pp. 1-6). IEEE.

[7] Afif, M., Ayachi, R., Said, Y. and Atri, M., 2021. Deep learning-based application for

indoor wayfinding assistance navigation. Multimedia Tools and Applications, 80(18), pp.27115-27130.

[8] Gandhi, A., Punde, P., Khaire, P., Bhattad, A. and Shaikh, N.F., 2024. A* Algorithm and Unity for Augmented Reality-based Indoor Navigation. Grenze International Journal of Engineering & Technology (GIJET), 10.

[9] Tseng, P.Y., Lin, J.J., Chan, Y.C. and Chen, A.Y., 2022. Real-time indoor localization with visual SLAM for in-building emergency response. Automation in Construction, 140, p.104319.

[10] Zubairi, J.A. and Idwan, S., 2022. Localization and Rescue Planning of Indoor Victims in a Disaster. Wireless Personal Communications, 126(4), pp.3419-3433.

[11] Yoo, S.J. and Choi, S.H., 2022. Indoor ar navigation and emergency evacuation system based on machine learning and iot technologies. IEEE Internet of Things Journal, 9(21), pp.20853-20868.

[12] Wakchaure, M., Tamboli, M. and Sonkar, S., 2022, August. Indoor navigation system for public evacuation in emergency situation. In Journal of Physics: Conference Series (Vol. 2327, No. 1, p. 012062). IOP Publishing.

[13] Elsanhoury, M., Nieminen, J., Välisuo, P., Siemuri, A., Koljonen, J., Elmusrati, M.S. and Kuusniemi, H., 2023, July. Indoor Asset Tracking in Dense Industrial Environments Using Low-cost Wireless Technologies. In WIPHAL.

[14] Avellaneda, D., Mendez, D. and Fortino, G., 2023. A tinyml deep learning approach for indoor tracking of assets. Sensors, 23(3), p.1542.

[15] Fatima, H.S., Zaman, L.U., Zia, H. and Khurram, M., 2022. High Precision Indoor Positioning System for Mobile Asset Management and Safety. Engineering Proceedings, 20(1), p.37.

[16] Shang, S. and Wang, L., 2022. Overview of WiFi fingerprinting-based indoor positioning. Iet Communications, 16(7), pp.725-733.

[17] Feng, X., Nguyen, K.A. and Luo, Z., 2024. A Review of Open Access WiFi Fingerprinting Datasets for Indoor Positioning. IEEE Access.

[18] Singh, N., Choe, S. and Punmiya, R., 2021. Machine learning based indoor localization using Wi-Fi RSSI fingerprints: An overview. IEEE access, 9, pp.127150-127174.

[19] Ji, T., Li, W., Zhu, X. and Liu, M., 2022, March. Survey on indoor fingerprint localization for BLE. In 2022 IEEE 6th information technology and mechatronics engineering conference (ITOEC) (Vol. 6, pp. 129-134). Ieee.

[20] Shi, T. and Gong, W., 2024, May. A Survey of Bluetooth Indoor Localization. In 2024 IEEE 10th Conference on Big Data Security on Cloud (BigDataSecurity) (pp. 71-77). IEEE.

[21] Chen, C.H., Chen, P.W., Chen, P.J. and Liu, T.H., 2021. Indoor positioning using magnetic fingerprint map captured by magnetic sensor array. Sensors, 21(17), p.5707.

[22] Rafique, H., Patti, D., Palesi, M. and La Delfa, G.C., 2025. Indoor localization by projecting magnetic field signals onto images with vision transformer. Computers and Electrical Engineering, 123, p.110074.

[23] Luo, R.C. and Hsiao, T.J., 2019. Indoor localization system based on hybrid Wi-Fi/BLE and hierarchical topological fingerprinting approach. IEEE Transactions on Vehicular Technology, 68(11), pp.10791-10806.

[24] A.N. Nor Hisham, Y.H. Ng, C.K. Tan, and D. Chieng, "Hybrid Wi-Fi and BLE Fingerprinting Dataset for Multi-Floor Indoor Environments with Different Layouts," *Data*, vol. 7(11), p.156, 2022.

[25] P. Roy, C. Chowdhury, A Survey of Machine Learning Techniques for Indoor Localization and Navigation Systems, Journal of Intelligent & Robotic Systems 101 (3) (2021) 1–34. doi:10.1007/s10846-021-01327-z.

[26] Junoh, S.A. and Pyun, J.Y., 2022. Region Classification using Wi-Fi and Magnetic Field Strength. In IPIN-WiP.

[27] Pereira, T., Santos, V., Gameiro, T., Viegas, C. and Ferreira, N., 2024. Evaluation of Different Filtering Methods Devoted to Magnetometer Data Denoising. Electronics, 13(11), p.2006.

[28] Furfari, F., Girolami, M., Mavilia, F. and Barsocchi, P., 2025. Indoor localization algorithms based on Angle of Arrival with a benchmark comparison. Ad Hoc Networks, 166, p.103691.

[29] Zhang, J., Wang, B., Hu, Z., Koh, P.W.W. and Ratner, A.J., 2023. On the trade-off of intra-/inter-class diversity for supervised pre-training. Advances in Neural Information Processing Systems, 36, pp.64193-64212.

[30] Yong, Y.F., Tan, C.K. and Tan, I.K., 2021, October. Smote for wi-fi fingerprint construction in indoor positioning systems. In 2021 IEEE International Performance, Computing, and Communications Conference (IPCCC) (pp. 1-6). IEEE.

[31] Wei, B., Gao, M., Li, F., Luo, C., Wang, S. and Zhang, J., 2024. rWiFiSLAM: effective WiFi ranging based SLAM system in ambient environments. IEEE Robotics and Automation Letters, 9(6), pp.5362-5369.

[32] Jaisinghani, D., Balan, R.K., Naik, V., Misra, A. and Lee, Y., 2018, November. Experiences & challenges with server-side wifi indoor localization using existing infrastructure. In Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (pp. 226-235).

[33] Ye, L., Wang, Y., Pei, S., Wang, Y., Zhao, H. and Dong, S., 2025. Multi-granularity and multi-modal feature fusion for indoor positioning. Symmetry, 17(4), p.597.

[34] Turgut, Z. and Kakisim, A.G., 2024. An explainable hybrid deep learning architecture for WiFi-based indoor localization in Internet of Things environment. Future Generation Computer Systems, 151, pp.196-213.

[35] Chikodili, N.B., Abdulmalik, M.D., Abisoye, O.A. and Bashir, S.A., 2020, November. Outlier detection in multivariate time series data using a fusion of K-medoid, standardized euclidean distance and Z-score. In International Conference on Information

and Communication Technology and Applications (pp. 259-271). Cham: Springer International Publishing.

[36] Todorov, A., Stoykova, V. and Zlatev, Z., 2023. Improving signal strength estimation in IoT using Wi-Fi network performance data. Applied Research in Technics, Technologies and Education, 11(4), pp.224-236.

[37] D. Li, Y. Lei, X. Li, H. Zhang, Deep learning for fingerprint localization in indoor and outdoor environments, ISPRS International Journal of Geo-Information 9 (4) (2020) 267.

[38] M. Okamoto, C. Chen, Improving gps-based indoor-outdoor detection with moving direction information from smartphone, in: Adjunct Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2015 ACM International Symposium on Wearable Computers, 2015, pp. 257–260.

[39] S. Rajak, A. K. Panja, C. Chowdhury, S. Neogy, A ubiquitous indoor-outdoor detection and localization framework for smartphone users, in: Advances in Intelligent Systems and Computing (IEMIS 2020), Springer.

[40] N. Kroll, J. Michael, S. Sebastian, Context-aware indoor-outdoor detection for seamless smartphone positioning (2016).

[41] P. Roy, M. Kundu, C. Chowdhury, Indoor localization using stable set of wireless access points subject to varying granularity levels, in: 2019 International conference on wireless communications signal processing and networking (WiSPNET), IEEE, 2019, pp. 491–496.

[42] G. Shtar, B. Shapira, L. Rokach, Clustering wi-fi fingerprints for indoor–outdoor detection, Wireless Networks 25 (3) (2019) 1341–1359.

[43] A. H. Salamah, M. Tamazin, M. A. Sharkas, M. Khedr, An enhanced wifi indoor localization system based on machine learning, in: 2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN), IEEE, 2016, pp. 1–8.

[44] P. S. Varma, V. Anand, Random forest learning based indoor localization as an iot service for smart buildings, Wireless Personal Communications 117 (4) (2021) 3209–3227.

[45] S. Iwata, K. Ishikawa, T. Takayama, M. Yanagisawa, N. Togawa, Robust indoor/outdoor detection method based on sparse gps positioning information, in: 2018 IEEE 8th International Conference on Consumer Electronics-Berlin (ICCE-Berlin), IEEE, 2018, pp. 1–4.

[46] I. Bisio, A. Delfino, F. Lavagetto, Poster: Detecting if a smartphone is indoors or outdoors with ultrasounds, 2015. doi:10.1145/2742647.2745907.

[47] A. Esmaeili Kelishomi, A. Garmabaki, M. Bahaghighat, J. Dong, Mobile user indoor-outdoor detection through physical daily activities, Sensors 19 (3) (2019) 511.

[48] I. Ashraf, S. Hur, Y. Park, Magio: Magnetic field strength based indoor-outdoor detection with a commercial smartphone, Micromachines 9 (10) (2018) 534.

[49] J. Yang, X. Zhao, Z. Li, Crowdsourcing indoor positioning by light-weight automatic fingerprint updating via ensemble learning, IEEE Access 7 (2019) 26255–26267.

[50] S. He, W. Lin, S.-H. G. Chan, Indoor localization and automatic fingerprint update with altered ap signals, IEEE Transactions on Mobile Computing 16 (7) (2016) 1897–1910.

[51] K. Sabanci, E. Yigit, D. Ustun, A. Toktas, M. F. Aslan, Wifi based indoor localization: application and comparison of machine learning algorithms, in: 2018 XXIIIrd International Seminar/Workshop on Direct and Inverse Problems of Electromagnetic and Acoustic Wave Theory (DIPED), IEEE, 2018, pp. 246–251.

[52] A. B. Adege, Y. Yayeh, G. Berie, H.-p. Lin, L. Yen, Y. R. Li, Indoor localization using k-nearest neighbor and artificial neural network back propagation algorithms, in: 2018 27th Wireless and Optical Communication Conference (WOCC), IEEE, 2018, pp. 1–2.

[53] G. Anand, V. Thanikaiselvan, Improving the performance of rssi based indoor localization techniques using neural networks, in: 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA), IEEE, 2018, pp. 249–253.

[54] P. A. Karegar, Wireless fingerprinting indoor positioning using affinity propagation clustering methods, Wireless Networks 24 (8) (2018) 2825–2833.

[55] A. Alitaleshi, H. Jazayeriy, J. Kazemitabar, Affinity propagation clustering-aided two-label hierarchical extreme learning machine for wi-fi fingerprinting-based indoor positioning, Journal of Ambient Intelligence and Humanized Computing 13 (6) (2022) 3303–3317.

[56] L. Kanaris, A. Kokkinis, A. Liotta, S. Stavrou, Fusing bluetooth beacon data with wi-fi radiomaps for improved indoor localization, Sensors 17 (4) (2017) 812.

[57] C. Zhang, N. Qin, Y. Xue, L. Yang, Received signal strength-based indoor localization using hierarchical classification, Sensors 20 (4) (2020) 1067.

[58] A. K. Panja, S. F. Karim, S. Neogy, C. Chowdhury, A novel feature based ensemble learning model for indoor localization of smartphone users, Engineering Applications of Artificial Intelligence 107 (2022) 104538.

[59] D. Ghosh, P. Roy, C. Chowdhury, S. Bandyopadhyay, An ensemble of condition based classifiers for indoor localization, in: 2016 IEEE International conference on advanced networks and telecommunications systems (ANTS), IEEE, 2016, pp. 1–6.

[60] R. Gomes, M. Ahsan, A. Denton, Random forest classifier in sdn framework for user-based indoor localization, in: 2018 IEEE International Conference on Electro/Information Technology (EIT), IEEE, 2018, pp. 0537–0542.

[61] P. Roy, C. Chowdhury, M. Kundu, D. Ghosh, S. Bandyopadhyay, Novel weighted ensemble classifier for smartphone based indoor localization, Expert Systems with Applications 164 (2021) 113758.

[62] W. Zhang, R. Sengupta, J. Fodero, X. Li, Deeppositioning: Intelligent fusion of pervasive magnetic field and wifi fingerprinting for smartphone indoor localization via deep learning, in: 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), IEEE, 2017, pp. 7–13.

[63] R. Ayyalasomayajula, A. Arun, C. Wu, S. Sharma, A. R. Sethi, D. Vasisht, D. Bharadia, Deep learning based wireless localization for indoor navigation, in: Proceedings of the 26th Annual International Conference on Mobile Computing and Networking, 2020, pp. 1–14.

[64] G. Sepulveda, J. C. Niebles, A. Soto, A deep learning based behavioral approach to indoor autonomous navigation, in: 2018 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2018, pp. 4646–4653.

[65] E. Ovalle-Magallanes, N. G. Aldana-Murillo, J. G. Avina-Cervantes, J. Ruiz-Pinales, J. Cepeda-Negrete, S. Ledesma, Transfer learning for humanoid robot appearance-based localization in a visual map, IEEE Access 9 (2021) 6868–6877.

[66] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE conference on computer vision and pattern recognition, Ieee, 2009, pp. 248–255.

[67] H. Ma, K. Wang, Fusion of rss and phase shift using the kalman filter for rfid tracking, IEEE Sensors Journal 17 (11) (2017) 3551–3558.

[68] I. Ashraf, S. Hur, S. Park, Y. Park, Deeplocate: Smartphone based indoor localization with a deep neural network ensemble classifier, Sensors 20 (1) (2019) 133.

[69] Y. Yin, X. Yang, P. Li, K. Zhang, P. Chen, Q. Niu, Localization with transfer learning based on fine-grained subcarrier information for dynamic indoor environments, Sensors 21 (3) (2021) 1015.

[70] Y. Chen, W. Liu, H. Zhao, S. Cao, S. Fu, D. Jiang, Bisecting k-means based fingerprint indoor localization, Wireless Networks 27 (5) (2021) 3497–3506.

[71] N. Gholizadeh, H. Saadatfar, N. Hanafi, K-dbscan: An improved dbscan algorithm for big data, The Journal of Supercomputing 77 (2021) 6214–6235.

[72] X. Chen, S. Wu, C. Shi, Y. Huang, Y. Yang, R. Ke, J. Zhao, Sensing data supported traffic flow prediction via denoising schemes and ann: A comparison, IEEE Sensors Journal 20 (23) (2020) 14317–14328.

[73] P. Roy, C. Chowdhury, D. Ghosh, S. Bandyopadhyay, Juindoorloc: A ubiquitous framework for smartphone-based indoor localization subject to context and device heterogeneity, Wireless Personal Communications 106 (2) (2019) 739–762.

[74] P. J. Rousseeuw, Silhouettes: a graphical aid to the interpretation and validation of cluster analysis, Journal of computational and applied mathematics 20 (1987) 53–65.

[75] T. Caliński, J. Harabasz, A dendrite method for cluster analysis, Communications in Statistics-theory and Methods 3 (1) (1974) 1–27.

[76] D. L. Davies, D. W. Bouldin, A cluster separation measure, IEEE transactions on pattern analysis and machine intelligence (2) (1979) 224–227.

[77] M. T. Hoang, Y. Zhu, B. Yuen, T. Reese, X. Dong, T. Lu, R. Westendorp, M. Xie, A soft range limited k-nearest neighbors algorithm for indoor localization enhancement, IEEE Sensors Journal 18 (24) (2018) 10208–10216.

[78] J. Bi, Y. Wang, B. Yu, H. Cao, T. Shi, L. Huang, Supplementary open dataset for wifi indoor localization based on received signal strength, Satellite Navigation 3 (1) (2022) 1–15.

[79] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[80] James M Joyce. Kullback-leibler divergence. In *International encyclopedia of statistical science*, pages 720–722. Springer, 2011.

[81] M.R. Smith, T. Martinez, and C.G. Carrier,"An instance level analysis of data complexity," Machine learning, 95, pp.225-256, 2014, doi:10.1007/s10994-013-5422-z.

[82] T. Barua, R. Doshi, and K.K. Hiran, "Mobile Applications Development: With Python in Kivy Framework," Walter de Gruyter GmbH & Co KG, 2020.

[83] A.F.M. Gad, "Building Android Apps in Python Using Kivy with Android Studio: With Pyjnius, Plyer, and Buildozer," Apress, 2020.

[84] P. Roy, C. Chowdhury, D. Ghosh, and S. Bandyopadhyay, "JUIndoorLoc: A ubiquitous framework for smartphone-based indoor localization subject to context and device heterogeneity," Wireless Personal Communications, 106, pp.739-762, 2019, doi:10.1007/s11277-019-06188-2.

[85] P.E. Danielsson, "Euclidean distance mapping,"Computer Graphics and image processing, 14(3), pp.227-248, 1980, doi:10.1016/0146-664X(80)90054-4.

[86] M. Mohammadi, A. Al-Fuqaha, M. Guizani, and J.S. Oh, "Semisupervised deep reinforcement learning in support of IoT and smart city services," IEEE Internet of Things Journal, 5(2), pp.624-635, doi:10.1109/JIOT.2017.2712560.

[87] J. Ren, Y. Wang, C. Niu, W. Song, and S. Huang, "A novel clustering algorithm for Wi-Fi indoor positioning," IEEE Access, 7, pp.122428-122434, 2019, doi:10.1109/ACCESS.2019.2937464.

[88] R. Suganya, and R. Shanthi, "Fuzzy c-means algorithm-a review. International Journal of Scientific and Research Publications," 2(11), p.1., 2012.

[89] M. Mirza, and S. Osindero, "Conditional generative adversarial nets," arXiv preprint arXiv:1411.1784, 2014.

[90] R. Fabbri, L.D.F. Costa, J.C. Torelli and O.M. Bruno, "2D Euclidean distance transform algorithms: A comparative survey," *ACM Computing Surveys (CSUR)*, vol. 40(1), pp.1-44, 2008.

[91] J. Bi, Y. Wang, B. Yu, H. Cao, T. Shi and L. Huang, "Supplementary open dataset for WiFi indoor localization based on received signal strength," *Satellite Navigation*, vol. 3(1), pp.1-15.

[92] M.T. Hoang, Y. Zhu, B. Yuen, T. Reese, X. Dong, T. Lu, R. Westendorp and M. Xie, "A soft range limited K-nearest neighbors algorithm for indoor localization enhancement," *IEEE Sensors Journal*, vol. 18(24), pp.10208-10216, 2018.

[93] X. Wang, L. Gao, S. Mao and S. Pandey, "CSI-based fingerprinting for indoor localization: A deep learning approach", *IEEE transactions on vehicular technology*, vol. 66(1), pp.763-776, 2016.

[94] Thanapol, P., Lavangnananda, K., Bouvry, P., Pinel, F. and Leprévost, F., 2020, October. Reducing overfitting and improving generalization in training convolutional neural network (CNN) under limited sample sizes in image recognition. In 2020-5th International Conference on Information Technology (InCIT) (pp. 300-305). IEEE.

[95] Hoang, M.T., Zhu, Y., Yuen, B., Reese, T., Dong, X., Lu, T., Westendorp, R. and Xie, M., 2018. A soft range limited K-nearest neighbors algorithm for indoor localization enhancement. IEEE Sensors Journal, 18(24), pp.10208-10216.

[96] Bi, J., Wang, Y., Yu, B., Cao, H., Shi, T. and Huang, L., 2022. Supplementary open dataset for WiFi indoor localization based on received signal strength. Satellite Navigation, 3(1), p.25.

[97] Soro, B. and Lee, C., 2019. Joint time-frequency RSSI features for convolutional neural network-based indoor fingerprinting localization. IEEE Access, 7, pp.104892-104899.

[98] Karakusak, M.Z., Kivrak, H., Ates, H.F. and Ozdemir, M.K., 2022. Rss-based wireless lan indoor localization and tracking using deep architectures. Big Data and Cognitive Computing, 6(3), p.84.

[99] Alimoglu, F. and Alpaydin, E., 1997, August. Combining multiple representations and classifiers for pen-based handwritten digit recognition. In Proceedings of the Fourth International Conference on Document Analysis and Recognition (Vol. 2, pp. 637-640). IEEE.

[100] LeCun, Y., 1998. The MNIST database of handwritten digits. http://yann. lecun. com/exdb/mnist/.