ISSN 1688-2806

UNIVERSIDAD DE LA REPÚBLICA
FACULTAD DE INGENIERÍA

On the Application of Graph Neural Networks for Indoor Positioning Systems

TESIS PRESENTADA A LA FACULTAD DE INGENIERÍA DE LA UNIVERSIDAD DE LA REPÚBLICA POR

Facundo Lezama Carignani

EN CUMPLIMIENTO PARCIAL DE LOS REQUERIMIENTOS PARA LA OBTENCIÓN DEL TÍTULO DE MAGISTER EN INGENIERÍA ELÉCTRICA.

Montevideo
Wednesday 23rd November, 2022

On the Application of Graph Neural Networks for Indoor Positioning Systems, Facundo Lezama Carignani.

ISSN 1688-2806

This thesis was prepared in LaTeX using the iietesis class (v1.1).
It contains a total of 86 pages.
Compiled on Friday 4th November, 2022.
`http://iie.fing.edu.uy/`

Lo imposible cuesta un poco más, y derrotados son sólo aquellos que bajan los brazos y se entregan.

José Mujica

This page has been intentionally left blank.

Acknowledgments

The experiments presented in this work were partly carried out using Tryolabs' infrastructure.

This page has been intentionally left blank.

To my family, friends and those who supported me along the way.

This page has been intentionally left blank.

Abstract

Due to the inability of GPS (Global Positioning System) or other GNSS (Global Navigation Satellite System) methods to provide satisfactory precision for the indoor localization scenario, indoor positioning systems resort to other signals already available on-site, typically Wi-Fi given its ubiquity. However, instead of relying on an error-prone propagation model as in ranging methods, the popular fingerprinting positioning technique considers a more direct data-driven approach to the problem. First of all, the area of interest is divided into zones, and then a machine learning algorithm is trained to map between, for instance, power measurements from Access Points (APs) to the localization zone, thus effectively turning the problem into a classification one.

However, although the positioning problem is a geometrical one, virtually all methods proposed in the literature disregard the underlying structure of the data, using generic machine learning algorithms. In this work we consider instead a graph-based learning method, Graph Neural Networks, a paradigm that has emerged in the last few years and constitutes the state-of-the-art for several problems. After presenting the pertinent theoretical background, we discuss two possibilities to construct the underlying graph for the positioning problem. We then perform a thorough evaluation of both possibilities, and compare it with some of the most popular machine learning alternatives. The main conclusion is that these graph-based methods obtain systematically better results, particularly with regards to practical aspects (e.g. gracefully tolerating faulty APs), which makes them a serious candidate to consider when deploying positioning systems.

This page has been intentionally left blank.

Resumen

Debido a la incapacidad del GPS (*Global Positioning System* o Sistema de Posicionamiento Global) o de otros métodos de navegación por satélite (GNSS por sus siglas en inglés) de proporcionar un posicionamiento en espacios interiores con suficiente precisión, se suele recurrir a otras señales ya disponibles en el lugar, típicamente Wi-Fi por su gran adopción. Existen diversas técnincas que utilizan la señal de Wi-Fi para realizar el posicionamiento modelando la propagación de la señal para alcanzar el objetivo. Sin embargo, debido a su alta complejidad, estos modelos de propagación son propensos a errores. Una alternativa que se popularizó es el posicionamiento en base a huellas (*fingerprinting*) que considera un enfoque basado en datos más directo al problema. El método consiste en dividir el área de interés en zonas y entrenar un algoritmo de aprendizaje automático para establecer una relación entre, por ejemplo, las mediciones de potencia de los puntos de acceso (*Access Points* o APs) y la zona de localización, convirtiéndose así en un problema de clasificación.

Si bien el problema de posicionamiento es en última instancia un problema geométrico, prácticamente todos los métodos propuestos en la literatura ignoran la estructura subyacente de los datos, utilizando para su resolución algoritmos genéricos de aprendizaje automático. Este trabajo propone utilizar un método de aprendizaje basado en grafos (*Graph Neural Networks* o GNN), un paradigma que ha surgido en los últimos años y que constituye el estado del arte para varios problemas. Tras presentar el marco teórico pertinente, discutimos dos posibilidades para construir el grafo subyacente al problema de posicionamiento. A continuación realizamos una evaluación exhaustiva de ambas posibilidades y las comparamos con algunas de las alternativas más populares de aprendizaje automático. La principal conclusión es que estos métodos basados en grafos obtienen sistemáticamente mejores resultados, especialmente en lo que respecta a los aspectos prácticos (por ejemplo, tolerar fallas en APs), lo que los convierte en excelentes candidatos a considerar a la hora de diseñar e implementar sistemas de posicionamiento.

This page has been intentionally left blank.

Contents

Acknowledgments **iii**

Abstract **vii**

Resumen **ix**

1 Introduction **1**

2 Indoor Localization **5**

2.1 Localization Techniques 5

2.1.1 Time of Arrival (ToA) 5

2.1.2 Time Difference of Arrival (TDoA) 6

2.1.3 Angle of Arrival (AoA) 7

2.1.4 Phase of Arrival (PoA) 7

2.1.5 Received Signal Strength Indicator (RSSI) 8

2.1.6 Channel State Information (CSI) 9

2.2 Wi-Fi Fingerprinting 10

2.2.1 Deterministic approaches 12

2.2.2 Probabilistic approaches 13

3 Machine Learning for Indoor Localization **17**

3.1 Problem Statement 17

3.2 Baseline Methods 18

3.2.1 K-Nearest Neighbors 18

3.2.2 Fully Connected Neural Networks 19

3.3 Graph Neural Networks 21

3.3.1 Graph Notions 22

3.3.2 Learning on Graphs 23

3.3.3 GNNs 24

3.4 Proposed GNN Architectures 29

3.4.1 Homogeneous Graphs 29

3.4.2 Heterogeneous Graphs 30

3.5 Related Work 33

4 Implementation Details **35**
4.1 Datasets 35
4.1.1 UJIIndoorLoc 35
4.1.2 MNAV 37
4.2 Graph Construction 37

5 Experiments and Results **43**
5.1 On the size of the training set 45
5.2 Propagation environment changes 47
5.3 AP failures 49
5.4 Adding new zones 50

6 Conclusions and Future Work **57**

Bibliography **59**

List of Tables **64**

List of Figures **66**

Chapter 1

Introduction

The key enabler for location-based services is naturally an accurate positioning system. This becomes even more crucial in indoor location-based services, which are not simply an extension of outdoor localization, but rather offer a new range of applications. Although for outdoor scenarios Global Navigation Satellite Systems (GNSS) such as GPS are generally enough, the signal is not strong enough to provide a sufficient precision for indoor applications, since it can be blocked by physical obstacles like buildings. The resulting error is typically in the tens of meters, barring its usage on scenarios in which precision is critical.

High-precision indoor localization pays a crucial role in many personal and business applications. Let us consider the use of indoor positioning systems in an emergency (such as a fire). It can help rescuers or residents locate their positions and find the shortest way to escape from the building, where smoke might even block their field of vision [25]. Another example is its use to assist businesses to understand the pattern of customer visits in places like shopping malls. Such information can then be used to stock up certain sections, display custom advertisements or design the layout of products. For personal navigation systems, a low-precision localization result can guide users to an erroneous destination. In museums, an accurate localization of visitors can enable the possibility of offering self-guided tours, or provide accessibility to visually impaired people, helping them have an independent experience, just to name a few examples [22]. In medical scenarios, an accurate indoor positioning system is a crucial technology for uses that range from quickly locating and tracking personnel (e.g. the nearest nurse), to helping identify paths of infection, or finding equipment and other medical devices (e.g. respirators) inside a hospital or a building.

Several approaches have been proposed to address this problem, which basically consider other signals to infer the position of the mobile device. For instance, the received power or the Channel State Information from a set of Wi-Fi, Bluetooth, Ultra Wide-Band or radio-frequency identification tags (RFID) transmitters with a known and fixed location (thus generally known as anchor nodes) may be used to this end, constituting the so-called *ranging techniques* [44]. These are generally model-based, requiring in turn a precise model that relates the position of the mobile to the received power, a model which is generally unavailable.

Instead of collecting a large set of measurements to derive a channel model, the so-called *fingerprinting technique* takes a more data-driven approach to the problem at hand: directly learning to map, for instance, the received powers (from the anchor nodes) to the position of the mobile, transforming the problem into a regression one [50]. In fact, depending on the final application, the actual coordinates of both the anchor nodes and the mobile device may actually be unnecessary (or even unavailable). For instance, we may want to identify at what shop is a certain customer of a shopping mall, and not the precise coordinates. In this case, the problem turns into a classification one. That is to say, the area is divided into zones and the objective is, given the power measurements from the anchor nodes, inferring at which zone the mobile device is. Note that in both cases a measurement stage is still necessary in order to train a learning algorithm to provide this mapping.

Interestingly, the vast majority of the existing fingerprinting techniques consider vector-based learning algorithms, ignoring the geometric nature of the problem [51]. Following our ongoing example, if the number of anchor nodes is n, then the input to the learning algorithm may be a vector in $\mathbb{R}^n$ (the power received from each anchor node), thus dropping the spatial information, which is to be inferred again by the learning algorithm. This will limit the generalization power of the resulting method. To illustrate the importance of considering the structure of the data, suffice to say that it is one of the main reasons behind the success of Convolutional Neural Networks (CNNs) for image and audio processing. CNNs precisely take advantage of the geometric properties of a grid-like structure and using convolutions, they are able to learn spatial hierarchies of features.

In this work, we study how to take into account the geometric information of the problem as an *a priori* on the learning algorithm. We will be using Wi-Fi Access Points (APs) as anchor nodes and the measured power (Received Strength Signal Indicator, RSSI) from all APs as the input to the learning algorithm, which will map this input to a zone (i.e. a classification problem). Extending the proposed methodology to other technologies, inputs or to consider it a regression problem is straightforward. We will define a graph with APs as its nodes, where the existence of an edge and its weight is indicative of the distance between each pair of APs (e.g. the mean RSSI between each pair of APs). A given input (i.e. the RSSI of each AP as measured by the mobile node) is now actually a number associated to each node on the graph (or a two-dimensional vector, if the APs are dual-band), thus defining a signal on the graph. The proposed classifier is based on Graph Neural Networks, which basically extend the concept of CNNs to signals defined on graphs [16].

We will introduce two versions of the learning algorithm. In the first one the graph signal is mapped to a zone. In a nutshell, we will transform the positioning problem into a graph classification one. By means of two public datasets we will compare this method to some popular alternatives, namely K-Nearest Neighbors, a Fully Connected Neural Network, or the combination of methods used in the indoor positioning software FIND3 [1]. Results indicate that the proposed method

[1]`https://github.com/schollz/find3`

performs systematically better in terms of accuracy. For instance, in one of the datasets it performed above KNN using less than half of the training samples. The code generated in this work is available on GitHub[2].

These virtues notwithstanding, this first method's main drawback is that it considers the geometry between APs only, ignoring the zones. We may include the zones on the graph, resulting in a so-called *heterogeneous* graph. In particular, although we may define the same kind of edges (e.g. mean RSSI from each AP to each zone), the signal on nodes corresponding to zones is clearly not of the same kind as the one on each AP (they may even be of different dimensions). In any case, the problem now turns into a graph signal interpolation one, where given the signal on the nodes corresponding to APs, we want to estimate a probability distribution over the nodes corresponding to zones. This method obtained similar results than the homogeneous case, with the addition to showing much better robustness in the scenario where one or several APs fail. The heterogeneous method was also tested on a scenario where a new zone is defined, e.g. when a new exposition gets added in a museum, and we do not have new training data. In this complex scenario we found that the model had promising results, being able to correctly classify inputs to the new zone for some specific configurations. It is important to note that neither of the methods require a map to construct the graphs, and we resort to the training dataset only.

The remainder of this document is structured as follows. Chapter 2 discusses the problem of indoor localization, introduces necessary background concepts and related localization examples. Chapter 3 presents the use of machine learning for indoor localization and delves into the methods used. Chapter 4 describes implementation details including the datasets used in this work and the graph construction algorithms. Chapter 5 presents in detail the solution designed along with the results obtained. Finally, Chapter 6 concludes this work and discusses some possible future research lines.

[2]`https://github.com/facundolezama19/indoor-localization-gnn`

This page has been intentionally left blank.

Chapter 2

Indoor Localization

Indoor localization is the process of obtaining a device or user location in an indoor setting or environment. Indoor device localization has been extensively investigated over the last few decades, mainly in industrial settings and for wireless sensor networks and robotics. However, it is only less than a decade ago since the wide-scale proliferation of smart phones and wearable devices with wireless communication capabilities have made the localization and tracking of such devices synonym to the localization and tracking of the corresponding users. This enabled a wide range of related applications and services. User and device localization have wide-scale applications in health sector, industry, disaster management, building management, surveillance and a number of various other sectors [51].

In this chapter we will provide a detailed explanation of several indoor localization techniques and technologies. We based this overview mainly in available surveys [2, 46, 48, 51].

2.1 Localization Techniques

In this section we will present different techniques used for indoor localization. Each of these have their advantages and disadvantages, some are widely used while some others are poorly adopted, but they all serve as a starting point to understand how to solve indoor localization problems.

2.1.1 Time of Arrival (ToA)

The ToA or Time of Flight (ToF) approach involves finding the amount of time needed by a signal to travel from the unlocated device (UD), e.g. a smartphone, to the anchor nodes (ANs), e.g. Wi-Fi APs. The UD is assumed to be located at a distance represented as a circle centered on the AN with a radius d estimated through the ToA. Hence, to detect the exact location of the UD, at least three ANs are required. Figure 2.1 shows a simple example of this. In this case, the estimated position of the UD is simply within the region of intersection (if it exists) of the three circles. The actual estimated position could then be easily obtained through

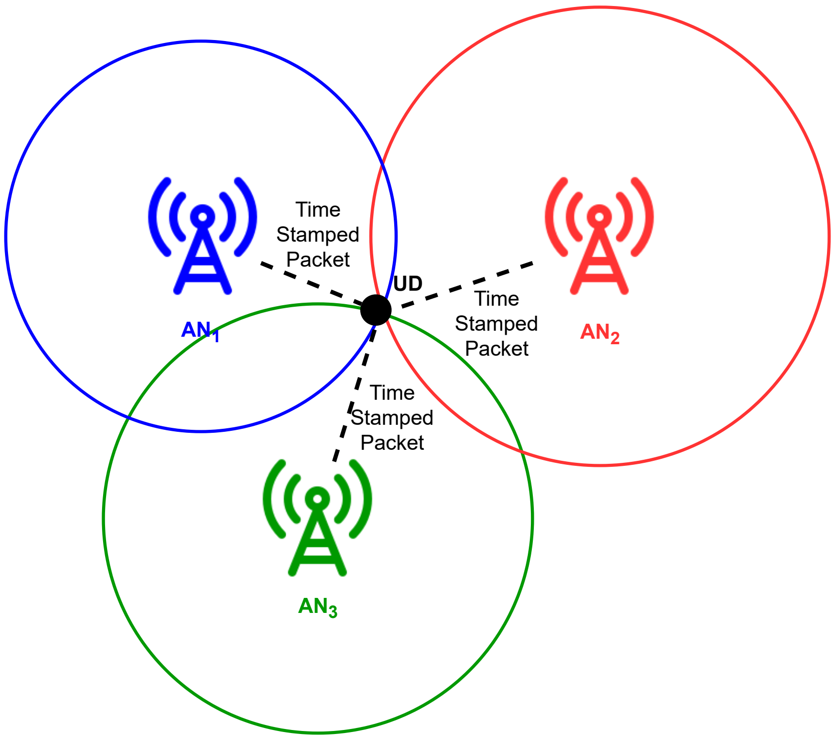


Figure 2.1: ToA based device localization example. The UD location is estimated as the intersection of the circles centered on the ANs.

any filtering technique such as Least Square or Weighted Least Square.

ToA requires synchronization between transmitters and receivers and timestamps to be transmitted with the signal (depending on the communication protocol). The key factors that affect ToA estimation accuracy are the signal bandwidth and the sampling rate. Low sampling rate (in time) reduces the ToA resolution since the signal may arrive between the sampled intervals. In multipath indoor environments, the larger the bandwidth, the higher the resolution of ToA estimation. Although large bandwidth and super-resolution techniques can improve the performance of ToA, still they cannot eliminate significant localization errors when the direct line of sight path between the transmitter and receiver is not available.

2.1.2 Time Difference of Arrival (TDoA)

TDoA examines the time difference at which the signal arrives at several ANs. The UD must lie on a hyperboloid for each TDoA measurement with a constant range difference between the two ANs. Such measurements are taken between multiple pairs of reference points with known locations. Also, relative time measurements are used at each receiving node as opposed to absolute time measurements. This is different from the ToA technique, where the absolute signal propagation time is used. No synchronized time source is needed by TDoA to perform localization, and synchronization is only needed at the ANs. The location to be estimated is

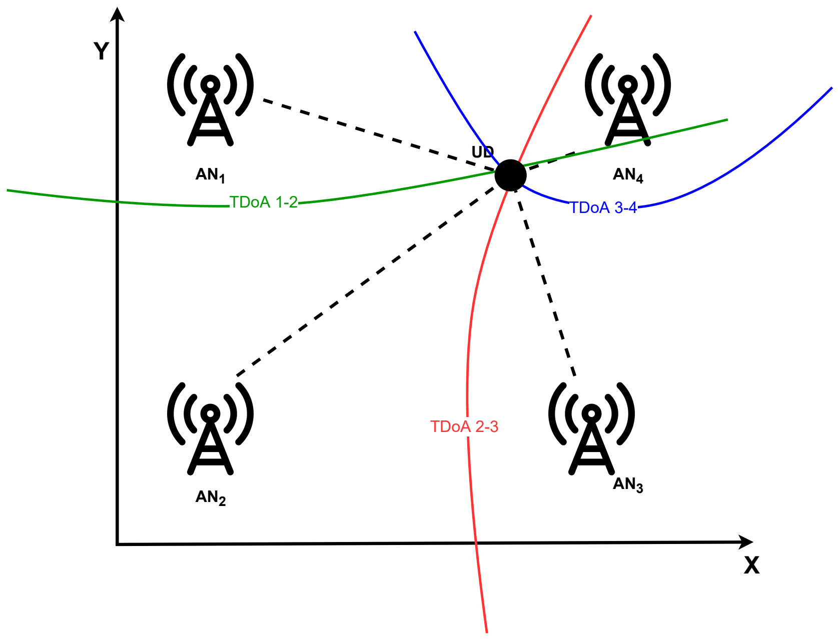


Figure 2.2: TDoA based localization and proximity detection. The UD position is obtained by estimating the intersection of the hyperbolic curves.

the intersection of many hyperbolic curves, as shown in Figure 2.2. This technique is referred to as multilateration.

2.1.3 Angle of Arrival (AoA)

AoA based approaches use several antennas (arrays) at the receiver side to estimate the angle at which the transmitted signal arrives on the receiver by exploiting and calculating the TDoA at individual elements of the array. The main advantage of AoA is that the UD location can be estimated with as little as two antennas in a 2D environment, or three in a 3D environment respectively. AoA can provide accurate estimation when the transmitter-receiver distance is small but it requires more complex hardware and careful calibration compared to other techniques such as RSSI (see Section 2.1.5). Also, its accuracy deteriorates with an increase in the transmitter-receiver distance where a small error in the angle of arrival calculation can translate into a huge error in the location estimation. Indoor environments are challenging for this technique due to multipath effects where the AoA in terms of line of sight is often hard to obtain. Figure 2.3 shows how AoA can be used to estimate the user location.

2.1.4 Phase of Arrival (PoA)

In order to estimate the distance between the transmitter and the receiver, PoA approaches look at the phase or phase difference of carrier signal. It is often assumed for determining the phase of the signal at the receiver side that the

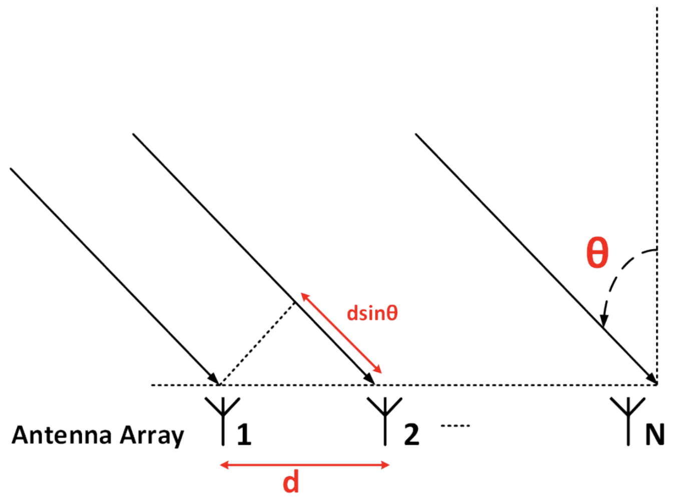


Figure 2.3: AoA based localization diagram [51]. The UD's signal arrives at several antennas in the antenna array and the angles at which it arrives on each antenna are used to intersect and calculate the UD position.

signals transmitted are of pure sinusoidal form and have the same frequency and zero phase offset. There are a number of techniques available to estimate the range or distance between the transmitter and the receiver using PoA. One of these is to assume that a finite delay exists between the transmitter and the receiver which can be expressed as a fraction of the signal's wavelength. Figure 2.4 shows a simple diagram illustrating the situation. The incident signals arrive with different phases at different antennas in the antenna array, which can be used to obtain the transmitter location.

2.1.5 Received Signal Strength Indicator (RSSI)

The RSSI based approach is one of the simplest and most widely used approaches for indoor localization [51]. The RSSI is the signal power strength received at the receiver. It is usually measured in dBm (decibel-milliwatts) and it can be used to estimate the distance between a transmitter and a receiver, where the higher the RSSI value the smaller the distance. The distance can be estimated using different propagation models given that the transmission power is known. Using the RSSI and a simple path loss propagation model, the distance d between the transmitter and the receiver can be estimated as:

$$RSSI = -10n\ log_{10}(d) + A \tag{2.1}$$

where n is the path loss exponent (which varies from 2 in free space to 4 in indoor environments) and A is the RSSI value at a reference distance from the receiver.

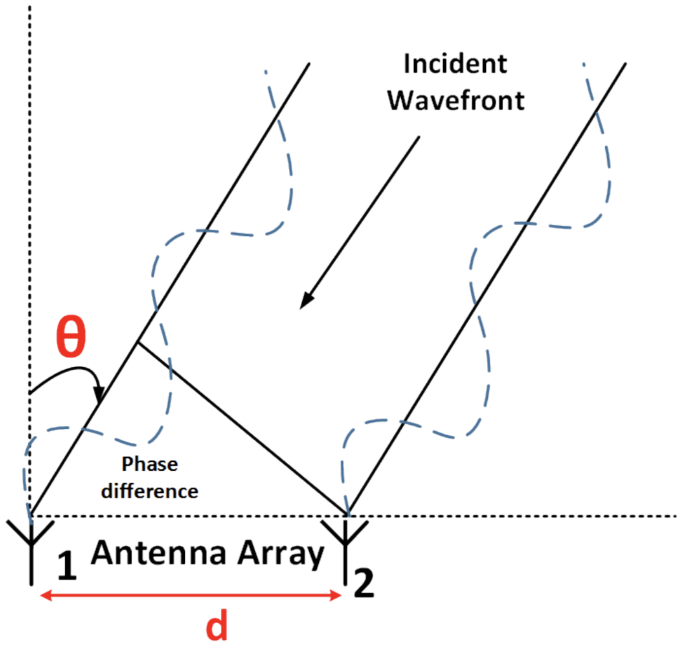


Figure 2.4: PoA based localization diagram [51]. Sinusoidal signals arrive at antennas in the antenna array with different phases. Knowing the signal's wavelength and assuming a finite distance between the transmitter and receiver, UD position can be estimated.

RSSI based localization requires trilateration, i.e., the RSSI at the device is used to estimate the absolute distance between the user device and three reference points. Then, similarly to ToA (see Section 2.1.1), basic trigonometry is applied to obtain the user device's location relative to the reference points as shown in Figure 2.5.

While the RSSI based approach is simple and cost efficient, it struggles, especially in non-line-of-sight conditions, due to additional signal attenuation resulting from transmission through walls, multipath fading and indoor noise.

2.1.6 Channel State Information (CSI)

Channel State Information (CSI) is another metric that can overcome the problem of multipath and temporal fluctuations especially in complex indoor environments. It represents the channel properties of a communication link. More specifically, CSI describes the propagation of signal from the transmitter to the receiver and gives information about the combined effect of, for instance, scattering, fading, and power decay with distance. Comparing to the RSSI, CSI is more resistant to environment changes.

In general, the CSI is a complex quantity and can be written as:

$$H(f) = |H(f)|e^{j\angle H(f)} , \tag{2.2}$$

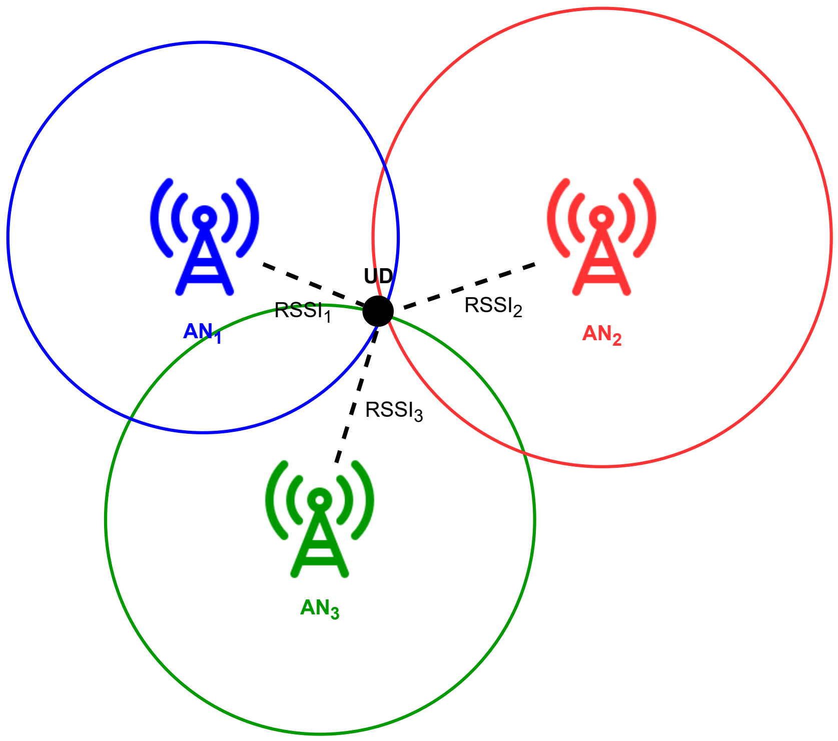


Figure 2.5: RSSI based localization. The UD location is estimated as the intersection of the circles centered on the ANs.

where $|H(f_i)|$ is the magnitude response and $\angle H(f_i)$ is the phase response of the frequency f_i of the channel.

This information can be used simply by trilateration or in a fingerprint scenario, as it will be described in the following section, to obtain the UD's location. Extensive evaluation results indicate that the scheme successfully copes with very complex indoor structure and simultaneously provides good performance in localization cost, and localization accuracy.

All of the presented techniques have something in common: they all need a physical model to estimate the UD's location. These models are complicated to estimate in indoor scenarios and many of the techniques include precise synchronization between devices. This opens an opportunity to use other more data-driven approaches for indoor localization. In the following section we present the approach used in this project: the widely studied fingerprinting approach.

2.2 Wi-Fi Fingerprinting

Although the techniques explained in the previous section can be applied in several network scenarios and technologies, when focusing on indoor scenarios we rapidly converge into one: WLAN (Wireless Local Area Network).

WLAN is one of the most popular solutions for indoor positioning and it provides better performance compared to other technologies, such as GPS and Bluetooth, in indoor scenarios. WLAN-based positioning systems are also much more adopted due to the fact that they do not require any additional software or hardware manipulation, but are able to perform localization based on the existing infrastructure. RSSI is widely used in WLAN-based positioning systems due to its easy extraction in 802.11 networks and its ability to run on off-the-shelf WLAN hardware, as opposed to CSI information that is harder to obtain. On the other hand, techniques such as ToA, TDoA, and AoA are less common to WLAN-based positioning systems since angular and time delay measurements are complex, as it was mentioned in previous sections. We will focus then on the use of RSSI signals for positioning. For simplicity and because it is one of the most popular WLAN technologies we are going to be referring to Wi-Fi from now on.

One of the most direct ways to obtain RSSI information in order to enable indoor localization services is by using the signals received on the process of Wi-Fi scanning. Wi-Fi scanning is the process through which available networks for connection are found. In a typical scenario, the scanning is performed at low rates since the set of available networks changes slowly. But when a device aims to estimate its position while acquiring Wi-Fi signals, several RSSI measurements are needed from the APs in order to minimize the positioning error. When the device is moving, a regular update is needed, hence, scanning for available APs is performed at a rate equivalent to the update rate. If a device is concerned about the positioning accuracy, it performs the scanning at a rate higher than the update rate so that the average of RSSI measurements reduces the effect of noise. On the other hand, a slower scanning rate than the update rate leads to reduced power consumption at the expense of positioning accuracy. Balancing the trade off between power consumption and positioning performance is the main driver for a device in selecting its parameters for scanning in Wi-Fi.

Although RSSI based methods are common and straightforward to use, there is still a limitation due to the physical modeling of the signal propagation. But RSSI measurements are not only used in the scenario described in 2.1.5. As we will see they are also leveraged on other data-driven localization techniques such as fingerprinting.

The most popular data-driven Wi-Fi indoor localization technique is based on the fingerprint approach. This is a technique that identifies the location of a user by characterizing the radio signal environment of the device. The process is described in 3 phases and can also be seen in Figure 2.6:

1. Is the so-called calibration phase, offline phase or training phase. This phase focuses on constructing a radio map by associating the fingerprints that were collected from RSSI measurements to their position.

2. An algorithm is developed to match new incoming measurements with the prerecorded radio map. The algorithm may be based on deterministic approaches, probabilistic approaches or their combinations.

3. Execution of positioning algorithm to estimate the UD's location.

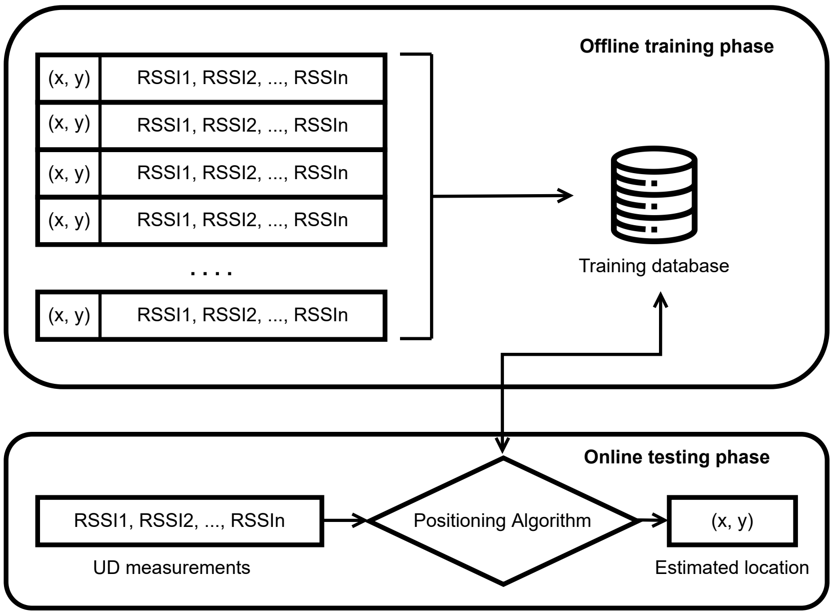


Figure 2.6: Fingerprinting process diagram [2]. The diagram shows the offline phase where training data is obtained and the database is constructed. The bottom part of the diagram describes the online phase where new RSSI measurements are received from an UD and the positioning algorithm outputs the estimated location.

The basic drawback of Wi-Fi fingerprint technology is the laborious, time consuming database construction process and a lot of human intervention. On the other hand RSSI is vulnerable to environment changes including the multipath problem and interference such as moving objects, door opening and closing. These constraints affect the database life cycle which requires another site survey and task calibration to be updated and adapted to the environment.

There are a number of algorithms available that can be used to match the offline measurements with the new online measurements. Some of these are discussed below.

2.2.1 Deterministic approaches

Deterministic methods use the raw data, associating a degree of similarity to each reference point independently. These methods are usually simple and useful for real time applications. One of the main advantages of the deterministic methods is their ease of implementation. For instance, the algorithm of K-Nearest Neighbors (KNN) is one of the simplest classification algorithms. Its principle is based on the search of neighbors and the determination of the device's position according to a similarity measure with its k-nearest neighbors. The only element needed is the definition of a distance between the new measurements and those in the database. This algorithm will be explained in depth in the following chapter.

2.2.2 Probabilistic approaches

Probabilistic methods rely on calculating the likelihood of the user being in a specific location provided the RSSI values obtained in online phase.

Artificial Neural Networks are used in a number of classification and forecasting scenarios. For localization, the neural network has to be trained using the RSSI values and the corresponding locations that are obtained during the offline phase. Once the neural network is trained, it can then be used to obtain the device's location based on the online RSSI measurements. The Multi-Layer Perceptron (MLP) network with one hidden layer is one of the simplest neural networks used for localization. In MLP based localization, an input vector of the RSSI measurements is taken as input and the obtained output is the estimated probabilities of the user's location. This method will be explained in depth in the following chapter.

A summary of the advantages and disadvantages of the techniques discussed in this chapter is presented here:

- **RSSI**
  - **Advantages:** Easy to implement, cost efficient, can be used with a number of technologies.
  - **Disadvantages:** Prone to multipath fading and environmental noise, lower localization accuracy, can require fingerprinting.
- **CSI**
  - **Advantages:** More robust to multipath and indoor noise.
  - **Disadvantages:** It is not easily available on off-the-shelf NICs.
- **AoA**
  - **Advantages:** Can provide high localization accuracy, does not require any fingerprinting.
  - **Disadvantages:** Might require directional antennas and complex hardware, requires comparatively complex algorithms and performance deteriorates with increase in distance between the transmitter and receiver.
- **ToA**
  - **Advantages:** Provides high localization accuracy, does not require any fingerprinting.
  - **Disadvantages:** Requires time synchronization between the transmitters and receivers, might require time stamps and multiple antennas at the transmitter and receiver. Line of Sight is mandatory for accurate performance.

- **TDoA**
  - **Advantages:** Does not require any fingerprinting, does not require clock synchronization among the device and AN.
  - **Disadvantages:** Requires clock synchronization among the ANs, might require time stamps, requires larger bandwidth.
- **PoA**
  - **Advantages:** Can be used in conjunction with RSS, ToA, TDoA to improve the overall localization accuracy.
  - **Disadvantages:** Degraded performance in the absence of line of sight.
- **Fingerprinting**
  - **Advantages:** Fairly easy to use.
  - **Disadvantages:** New fingerprints are required even when there is a minor variation in the space.

As we presented in this chapter, the problem of indoor localization is a widely studied topic and different methods and technologies have been used to address it. In [2] technologies such as the use of infrared sensors, ultrasound sensors, cameras and their subsequent processing, radio frequency technologies such as Radio Frequency Identification (RFID) and methods based on WLAN are presented. The latter has gained great interest in recent years due to the growth of Wi-Fi network deployments.

As we mentioned before, indoor positioning based on WLAN infrastructure is usually done by means of power measurements from the different network APs. Since it is not easy to fit a propagation model that accurately estimates the signal received at each point, fingerprint-based techniques have become the most popular approach to the problem [51]. In this case, the RF propagation model is not taken into account directly, but through RSSI measurements from mobile devices. This data-driven approach converts the problem into a machine learning one, turning the position estimation into a regression problem. It can be further simplified if, instead of the exact position, the goal is only to estimate the area where the mobile device is located, reducing the problem to a classification one. We will expand on this idea in the next chapter when we formally present the problem.

One of the most used methods to address this problem is K-Nearest Neighbors, due to its simplicity and good results [1, 40]. Other machine learning techniques used for this problem include Deep Learning [28], Random Forests, Support Vector Machine or Multi Layer Perceptron [4]. However, all of them are vector-based learning algorithms, where the model simplification ignores the geometric nature of the position estimation problem, particularly the relationship between the RSSI measurements at a certain point from the different APs.

A novel machine learning approach which we will explore next and that enables to take into account the geometric information as a model prior are Graph Neural

Networks (GNNs). For this purpose, a graph is defined with the network APs as nodes, and the edges weights should reflect the distance between each pair of APs. As detailed later on, in this model the RSSI values measured by the mobile devices will be signals defined on the graph. Although there are still not many indoor localization works based on GNNs in the literature yet, some promising results are already beginning to show that it is a successful approach to address the problem.

In the following chapter we will present both the basic notions of graph learning and the definitions of the studied GNN models, together with the constructed graphs.

This page has been intentionally left blank.

Chapter 3

Machine Learning for Indoor Localization

As stated in the previous chapter, KNN and Neural Networks based algorithms are widely used to approach the problem of indoor localization as a classification one. This is why we considered these algorithms as baselines to compare the results obtained in this work. In this Chapter we briefly present the ideas behind these algorithms along with a more thorough explanation of the studied approach: graph based learning and GNNs.

3.1 Problem Statement

As we mentioned before, in the fingerprinting approach the localization problem is usually turned into a classification one: given the RSSI measurements of the n_{AP} APs received by the device, the objective is to learn how to map these values to the corresponding zone. Figure 3.1 shows this idea on a simple diagram.

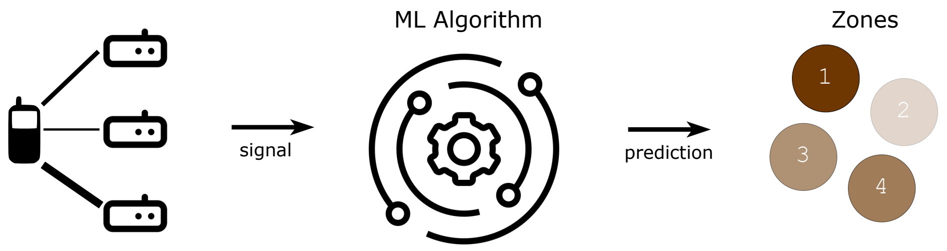


Figure 3.1: Example diagram showing the classification problem.

Formally, let us denote by $\mathbf{X} \in \mathbb{R}^{n_{AP} \times F_{in}}$ one of these measurements, where for instance $F_{in} = 2$ when measurements for both the 2.4 GHz and 5 GHz bands are available. We will use $\mathbf{x}_i \in \mathbb{R}^{F_{in}}$ to indicate the i-th row of $\mathbf{X}$, corresponding to the RSSI measurement from AP i (with a default value of, for instance, -100 dBm in case this particular AP's RSSI was below the sensitivity of the device). Given n_z

possible zones, we want to estimate the parameters of a function $\mathbf{\Phi} : \mathbb{R}^{n_{AP} \times F_{in}} \to \{1, \dots, n_z\}$ that minimizes a certain loss over the available training set.

We remark again that if other types of measurements are available (such as Channel State Information), they may easily be included in the framework by modifying the definition of $\mathbf{x}_i$ accordingly. Moreover, if we were interested in the actual coordinates of the device, we would simply change the codomain of the function $\mathbf{\Phi}$ and consider a regression problem; i.e. we would have $\mathbf{\Phi} : \mathbb{R}^{n_{AP} \times F_{in}} \to \mathbb{R}^3$ and the cost function would for instance be the Mean Squared Error instead of a Cross Entropy Loss used for classification. In any case, the family of functions $\mathbf{\Phi}$ typically chosen (e.g. a Neural Network) does not consider at all the underlying structure of the problem, which is expected to be learned from the training set instead. In this work we consider an alternative approach, where the geometric information is provided *a priori* by means of a graph.

3.2 Baseline Methods

3.2.1 K-Nearest Neighbors

Nearest neighbor methods are very simple, straightforward and widely used in both classification and regression problems. At the basic level, the observations of the training set closest in the input space are used to classify the new instance. In other words, to classify a new object with input vector $\mathbf{x} \in \mathbb{R}^{n_{AP}}$ we examine the k closest data points in the training set and assign to the object the class that has the most points among all of the k samples. This notion is defined as follows:

$$\hat{y}(\mathbf{x}) = \text{argmax}_{z \in \{1, \dots, n_z\}} \sum_{\mathbf{x}_i \in N_k(\mathbf{x})} \delta(z, y_i) \tag{3.1}$$

where $N_k(\mathbf{x})$ is the neighborhood of $\mathbf{x}$ composed by its k closest points $\mathbf{x}_i$ in the training set, y_i is the class associated to $\mathbf{x}_i$, and $\delta(a, b) = 1$ if $a = b$ and 0 otherwise. Thinking of this approach for indoor localization we would assign to a device the most common zone among its k nearest observations in the n_{AP}-dimensional space.

Of course, this simple outline leaves a lot unsaid. In particular, we must choose a suitable value for k and close is defined here in terms of the n_{AP}-dimensional input space, so a metric needs to be used in order to calculate the distances. A common approach is to use the Euclidean Distance but other several such as the Minkowski Distance, Manhattan Distance or the Cosine Distance are widely used. The most basic approach takes $k = 1$, making the classifier rather unstable, with high variance and sensitive to the data. This can be seen in Figure 3.2 where the boundaries define the spaces where new instances may be classified as one or another class. Those small boundaries are the ones that make the classifier unstable.

Predictions can often be more consistent by increasing k, thus reducing the variance but as a trade-off it may increase the bias of the method. As k increases, the training data points included in the neighborhood are not necessarily very

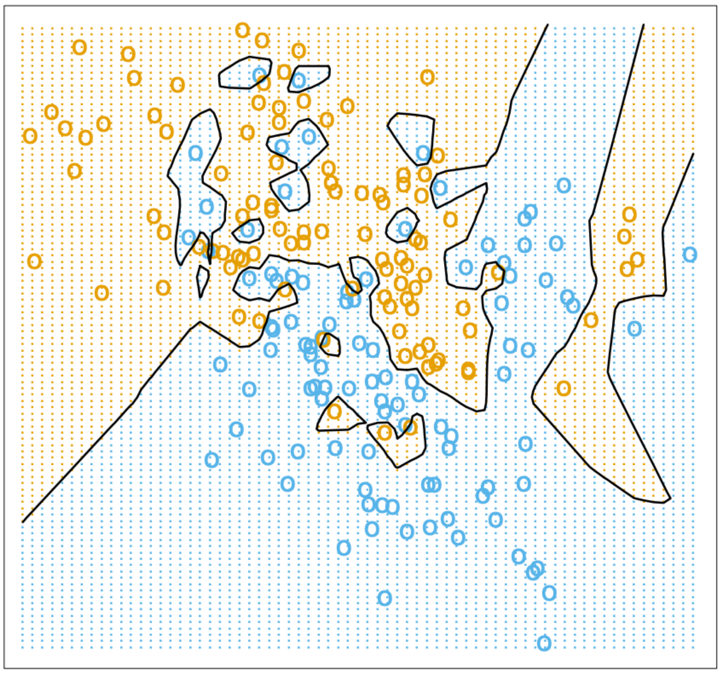


Figure 3.2: Classification example in two dimensions, the classes are coded as a binary variable (Blue = 0, Orange = 1), and then predicted by 1-nearest-neighbor classification. [18]

close to $\mathbf{x}$. An example of the regions defined by using $k = 15$ are shown in Figure 3.3, where the regions defined by the boundaries are less sparse.

3.2.2 Fully Connected Neural Networks

As stated in the previous Chapter, indoor localization problems do not escape from the widely adopted method of Deep Learning. In this work we evaluate a Fully Connected Neural Networks to use as a baseline to compare our results and that is why we include in this section a brief introduction.

A Fully Connected Neural Network (FCNN) is composed of one or more layers and is named that way because each layer connects every input feature to every output feature. Motivated by the explanation from [3], a layer is calculated by first constructing M linear combinations of the input features $x_1, x_2..., x_D$ as

$$a_j = \sum_{i=1}^{D} w_{j,i}^{(1)} x_i + w_{j,0}^{(1)}, \tag{3.2}$$

where $j = 1, ..., M$, and the superscript (1) indicates that the corresponding parameters are in the first layer of the network. $w_{j,i}^{(1)}$ are called the weights and $w_{j,0}^{(1)}$ are the biases.

The activations a_j are then transformed using a differentiable, nonlinear activation function $\sigma(\cdot)$ to obtain the outputs of the hidden units of the layers as

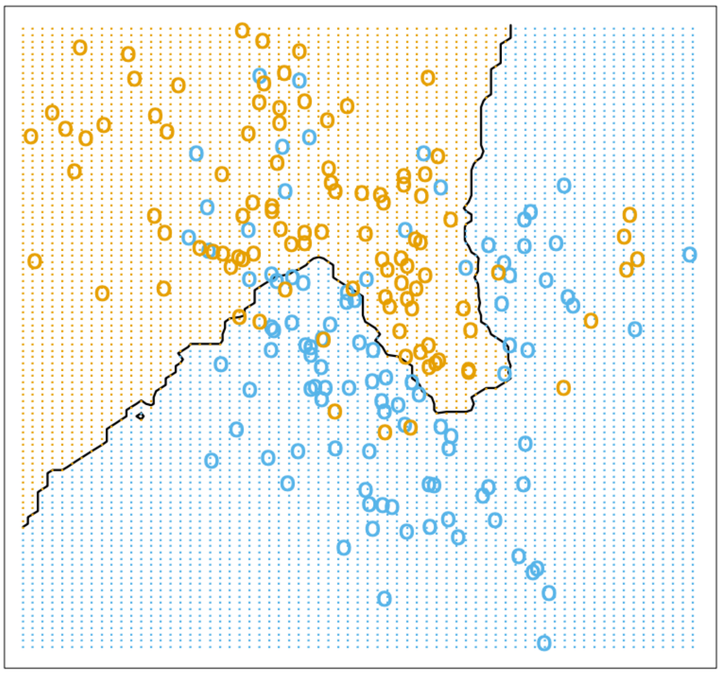


Figure 3.3: Classification example in two dimensions, the classes are coded as a binary variable (Blue = 0, Orange = 1). The predicted class is obtained by applying Equation 3.1 using the 15-nearest-neighbors. [18]

follows:

$$z_j = \sigma(a_j) \tag{3.3}$$

The nonlinear functions $\sigma(\cdot)$ are sometimes chosen to be sigmoidal functions such as the logistic sigmoid and $tanh$, or other popular ones such as ReLU.

Similarly, at the final layer of the network we have the output activations calculated by

$$a_k = \sum_{j=1}^{M} w_{k,j}^{(L)} z_j + w_{k,0}^{(L)}, \tag{3.4}$$

where $k = 1, ..., K$, and K is the total number of outputs. Finally, the output unit activations are transformed using an activation function resulting in the network outputs y_k. The activation function must be chosen accordingly to the nature of the problem for which the neural network is being used. For standard regression problems, the activation function is the identity so that $y_k = a_k$. On the other hand, for multiple binary classification problems the output unit activations are transformed by applying a logistic sigmoid. For multiclass problems such as the one studied in some parts of this work, a softmax activation function as described in the following equation is used.

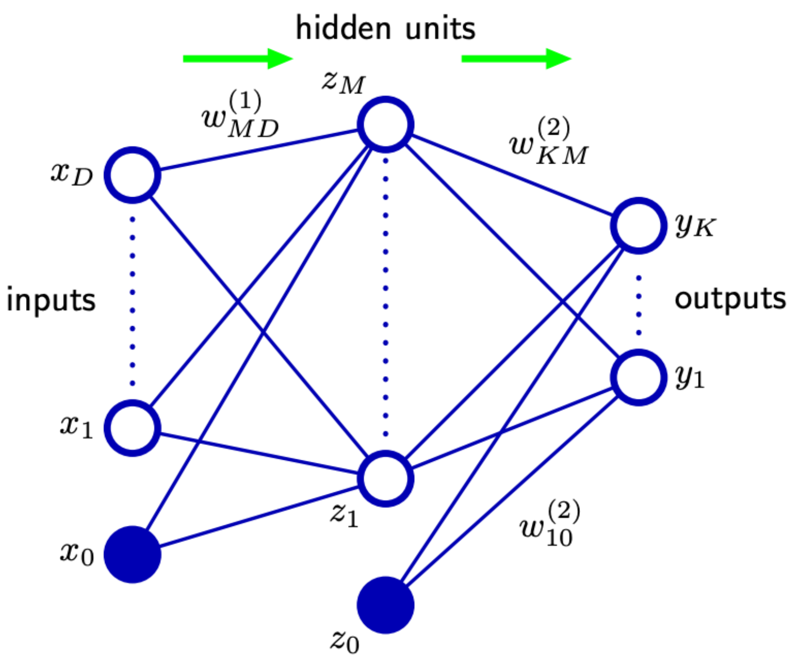


Figure 3.4: Network diagram for the two layer neural network corresponding to Equation 3.6. [3]

$$p(Ck|\mathbf{x}) = \frac{exp(a_k)}{\sum_j exp(a_j)} \tag{3.5}$$

Combining all these stages the overall network function, assuming two layers and sigmoid activation function at the output, takes the form

$$y_k(\mathbf{x}, \mathbf{w}) = \sigma\left(\sum_{j=1}^{M} w^{(2)}_{k,j}\sigma\left(\sum_{i=1}^{D} w^{(1)}_{j,i}x_i + w^{(1)}_{j,0}\right) + w^{(2)}_{k,0}\right), \tag{3.6}$$

where weights and biases have been grouped together into a vector $\mathbf{w}$. This can also be seen in Figure 3.4 where a two layer FCNN is shown. So the neural network model is simply a nonlinear function from a set of input variables x_i to a set of output variables y_k controlled by a vector $\mathbf{w}$ of adjustable parameters.

3.3 Graph Neural Networks

Learning from data represented by graphs or networks has received considerable attention over the last few years. Applications range from the analysis of social networks [43] to predicting properties of chemical compounds [8]. The most important challenge is that, differently to for instance audio or images, graph data is highly irregular and non-euclidean.

In this section we will introduce some basic graph theory, then we will present the network embedding problem and some graph learning applications, and lastly we will describe GNNs as it is the main learning algorithm used in this work.

3.3.1 Graph Notions

Let us first review some basic notions related to graphs so that we can further build on these concepts when explaining the different learning tasks and when introducing GNNs.

A graph or network G given as a pair $G = (V, E)$, is composed by a set of vertices (or nodes) $V = v_1, ..., v_n$ with $n = |V|$, connected by edges $E = e_1, ..., e_m$ with $m = |E|$, where each edge e_k is a pair (v_i, v_j) with $v_i, v_j \in V$. A graph is said to be weighted if there exists a weight function: $w : (v_i, v_j) \to w_{ij}$ that assigns weight w_{ij} to edge connecting nodes $v_i, v_j \in V$. Otherwise, we say that the graph is unweighted. A graph is undirected if $(v_i, v_j) \in E$ implies $(v_j, v_i) \in E$, i.e. the relationships are symmetric, and directed if the existence of edge $(v_i, v_j) \in E$ does not necessarily imply $(v_j, v_i) \in E$. Finally, a graph can be homogeneous if nodes refer to one type of entity and edges to one relationship. It can be heterogeneous if it contains different types of nodes and edges.

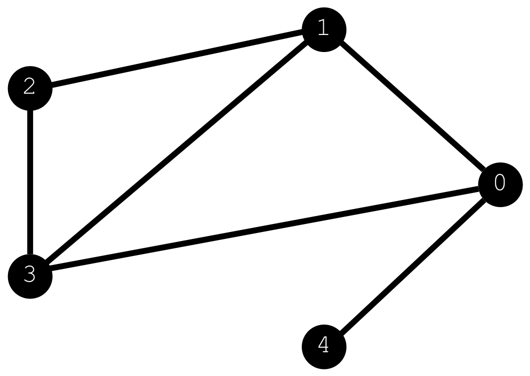


Figure 3.5: Example of an undirected graph. Nodes are represented by circles, which are labeled from zero to four. Edges between nodes are represented by lines.

An example can be social networks. These can be represented as undirected graphs, for example encoding symmetric relationships as friendship, or directed graphs encoding for example a follower/followed relationship in social media.

A path P from v_i to v_j of length k is a sequence of $k + 1$ vertices (v_i) and k edges (e_i) of the form v_0, e_1, v_1, e_2, v_2, ... , e_k, v_k, where each edge e_i connects v_{i-1} with v_i. A path is called simple if there are no repeated vertices. Otherwise, if a path visits a node more than once, it is said to contain a cycle.

Given two nodes (u, v) in a graph G, we define the distance from u to v, denoted $d_G(u, v)$, to be the length of the shortest path from u to v, or ∞ if there exists no path from u to v.

The degree, $deg(v_i)$, of a vertex v_i in an unweighted graph is the number of edges incident to it. Similarly, the degree of a vertex v_i in a weighted graph is the sum of incident edges weights. The degree matrix $\mathbf{D}$ of a graph with vertex set V is the $|V| \times |V|$ diagonal matrix such that $D_{ii} = deg(v_i)$.

A finite graph $G = (V, E)$ can be represented as a square $|V| \times |V|$ adjacency matrix, where the graph edges will be represented as the elements of the matrix. The adjacency matrix is binary for unweighted graphs, $\mathbf{A} \in \{0,1\}^{|V|\times|V|}$ (indicating whether pairs of nodes are adjacent or not), and non-binary for weighted graphs $\mathbf{A} \in \mathbb{R}^{|V|\times|V|}$ with $A_{ij} = w_{ij}$. Undirected graphs have symmetric adjacency matrices, in which case, $\tilde{\mathbf{A}}$ denotes symmetrically-normalized adjacency matrix: $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$, where $\mathbf{D}$ is the degree matrix.

The unnormalized Laplacian of an undirected graph is the $|V| \times |V|$ matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}$. The symmetric normalized Laplacian is $\tilde{\mathbf{L}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$. The random walk normalized Laplacian is the matrix $\mathbf{L}_{rw} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A}$.

The first order proximity between two nodes v_i and v_j is a local similarity measure, which captures the direct neighbor relationship between the vertices. For each vertex pair (v_i, v_j), if $(v_i, v_j) \in E$, the first-order proximity between v_i and v_j is w_{ij}; otherwise, the first order proximity between v_i and v_j is 0. In other words, the first-order proximity characterizes the local graph structure. The second order proximity between two nodes v_i and v_j measures the similarity of their neighborhood structures, capturing the global structure. Two nodes in a network will have a high second-order proximity if they tend to share many neighbors.

3.3.2 Learning on Graphs

Graphs are structures that represent complex relational information in a domain-agnostic way. To a large extent, it is their arbitrary structure what makes them so powerful, but also highly complex to use as learning data. To name just one example, operations like traditional convolutional filters are capable of efficiently processing data in Euclidean domains, but they are not as well-defined on irregular graph domains.

In an attempt to apply machine learning techniques to graph-structured data, new methods such as Graph Representation Learning (GRL) have been developed which aim to learn low-dimensional continuous vector representations (embeddings) for graph-structured data [5]. These methods take the graph elements (such as nodes, edges and local or global structure) to a lower dimensional space and preserve the information of graphs in the new embedding space. More specifically, network embedding aims at learning a mapping function from a discrete graph to a continuous domain. Formally, given a graph $G = (V, E)$ with weighted adjacency matrix $\mathbf{W} \in \mathbb{R}^{|V|\times|V|}$, the goal is to learn low dimensional vector representations $\{\mathbf{Z}_i\}_{i\in V}$ for nodes in the graph $\{v_i\}_{i\in V}$ to further use them in tasks such as the ones that we will describe in the following section.

As other learning techniques, network embeddings can be categorized in inductive and transductive, depending on whether the model can generalize to unseen data instances or not respectively.

Graph Embedding Applications

Graph embeddings can be applied to many tasks intrinsic to learning on graphs. Here we briefly explain some of these tasks.

Node classification is one of the main supervised graph applications, where the goal is to accurately predict node labels. For instance, in citation networks, node labels could be scientific topics and they have to be predicted based on citations. Exploiting the connection between nodes is a common approach to solve node classification. This can be done in multiple ways. There is the idea of homophilia, which is the tendency for nodes to share attributes (including labels) with their neighbors in the graph. In contrast, other networks behave according to the concept of heterophilia: nodes will be preferentially connected to nodes with different labels.

One can also predict edges over a graph, a task known as 'relation prediction', 'graph completion' or 'relational inference'. The basic idea is to determine the probability that an edge exists between two nodes. More advanced problems include predicting the type of edge, not just existence. A real world example of this task is recommending new content to social media users.

Tasks previously mentioned deal with inferring missing information, either labels or edges. A different task known as 'community detection' or 'clustering', on the other hand, identifies underlying communities amongst the nodes. Since there is no need to indicate known communities, it is often considered a unsupervised task.

Other applications on graphs involve classification, regression, or clustering problems over entire graphs. Similar to node classification, graph classification aims to predict graph-level labels given an input graph. As an example, [31] models the prediction of a molecule's odor as a graph classification problem. The molecule's embedding represents the underlying relationship between the molecule's structure (the graph) and the odor (the predicted class). See Figure 3.6 for an illustration. In this case, a GNN was used to learn the graph embedding. The next section will go over this technique in more detail.

3.3.3 GNNs

After covering *what* graph embeddings can be used for, let us discuss *how* they can be constructed. Many techniques have been proposed for generating effective graph embeddings, which can be grouped into two categories: traditional methods and Graph Neural Networks (GNNs) based methods. Examples of traditional methods include random walks, factorization methods, and temporal point processes.

Traditional techniques have been often referred to as the 'shallow' approach to generate graph embeddings, mainly due to its limitations. They are inherently transductive, meaning that they can only generate embeddings for nodes that were present during the training phase and are not able to generalize to unseen nodes after training [21].

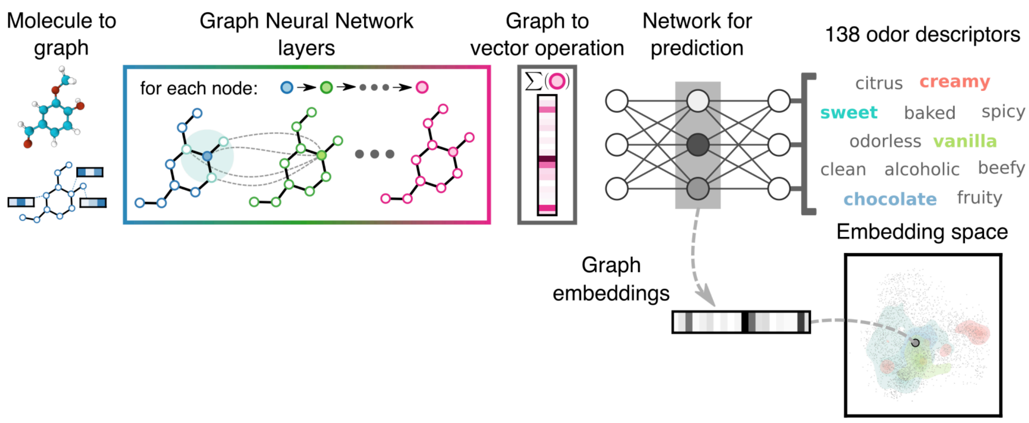


Figure 3.6: A molecule is represented as a graph and featurized by its constituent atoms, bonds, and connectivities. The graph embedding is used for predicting odor descriptors via a fully-connected neural network [31].

GNN, on the other hand, overcomes many of the traditional approaches' limitations. Mainly, it is able to generalize to unseen data (what we referred to as 'inductive'). The key idea behind a GNN is that the node embedding is generated by aggregating the node's neighbors embeddings.

Simply speaking, the GNN algorithm works as follows: at each iteration, every node aggregates information from its local neighborhood and at every step, further nodes are considered. After the first iteration, every node embedding contains information from its immediate neighbors. After the second iteration every node embedding contains information from its neighbors' neighbors and so on. This idea is illustrated in Figure 3.7. The information encoded is both structural and feature-based.

This local feature-aggregation behavior of GNNs is analogous to the behavior of the convolutional kernels in CNNs. However, whereas CNNs aggregate feature information from spatially-defined patches in an image, GNNs aggregate information based on local graph neighborhoods [17].

GNNs have been applied to a wide range of domains and their application can be classified as structural and non-structural, depending on the type of data. In the structural case, data has an inherent graph structure (e.g. social interaction networks or molecular networks). In the non-structural scenario, the structure in the data is not explicit and the graph needs to be constructed. This is the case of

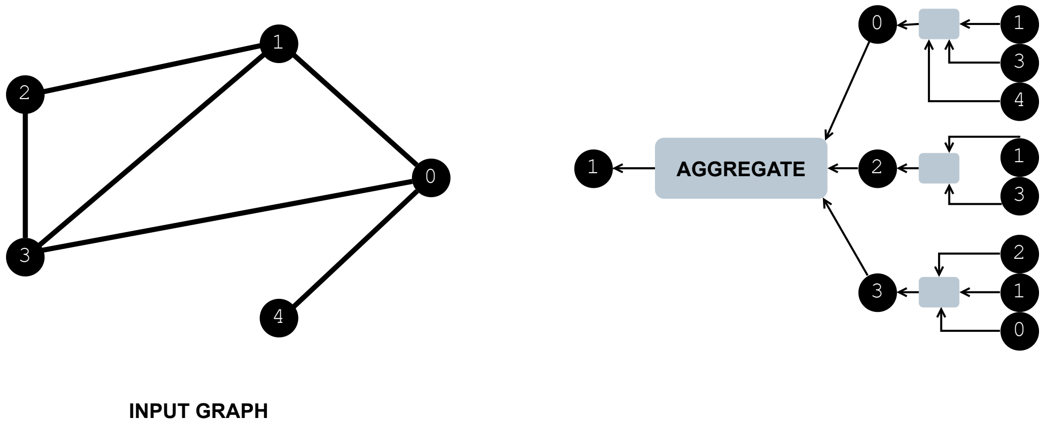


Figure 3.7: Overview of how a single node aggregates information from its local neighborhood. The model aggregates information from node 1's local graph neighbors (i.e. 0, 2 and 3), and in turn, the information coming from these neighbors are based on information aggregated from their respective neighborhoods, and so on.

this work, since multiple graphs can be created based on the same problem (indoor localization).

The motivation behind using GNNs to study the indoor localization problem is to take into account the underlying geometry. Although the utility of learned neural network embeddings in this problem is relatively unproven and to be confirmed, we still anticipate that a learned GNN embedding on an indoor localization task may produce a semantically meaningful and useful organization of the Wi-Fi deployment. As stated in [20], GNNs are suitable for problems in communication networks because of their strong learning ability to capture the spatial information hidden in the network topology and their generalization ability to be used in unseen topologies when the networks are dynamic.

Originally, some of the first works to propose Graph Neural Networks, which basically try to emulate the success obtained by CNNs onto the graph domain, represent node i (for $i = 1, \dots, n$) by a vector $\mathbf{x}_i \in \mathbb{R}^d$, which is iteratively updated by combining it with its neighbors' vector representation [32]. After a number of iterations, a final vector representation of each node is obtained, which is then used to, for instance, node classification.

An alternative approach to the problem was provided by taking a Graph Signal Processing perspective [35]. In particular, a spectral-based graph convolution was first considered [12], an operation that is extremely costly and numerically unstable. To remedy this, a Chebyshev polynomial on the Laplacian matrix was first proposed to approximate the spectral convolution [9]. Analogous to a discrete-time convolution, a first-order convolution layer for a GNN may be obtained by aggregating neighbors information (as shown in Figure 3.7) and can be formally defined by the following equation:

$$\mathbf{x}_i' = \sigma\left(\mathbf{\Theta}^T \sum_{j\in\mathcal{N}_i\cup\{i\}} S_{j,i}\mathbf{x}_j\right), \tag{3.7}$$

where $\mathbf{x}_i' \in \mathbb{R}^{d'}$ is the output of the layer, $\sigma(\cdot)$ is a point-wise non-linearity (e.g. the ReLU function), $\mathbf{\Theta} \in \mathbb{R}^{d\times d'}$ is the learneable parameter of this layer, $\mathcal{N}_i$ is the set of neighbors of node i, and $S_{j,i}$ is the j, i entry of matrix $\mathbf{S} \in \mathbb{R}^{n\times n}$, the so-called Graph Shift Operator (GSO). $\mathbf{S}$ is a matrix representation of the graph, which should respect its sparsity (i.e. $S_{i,j} \neq 0$ whenever there is an edge between nodes i and j). The adjacency matrix of the graph, its Laplacian or their normalized versions are all valid GSOs. Moreover, larger values of $S_{j,i}$ means that $\mathbf{x}_j$ will have more weight on Equation (3.7), and thus should be indicative that the signal on nodes i and j are more related to each other.

To grasp the rationale behind Equation (3.7) note that a discrete-time signal may be represented by a linear graph (successive samples are joined by an edge). If $d = d' = 1$, we would be linearly combining the previous and current samples, and then evaluating this with function $\sigma(\cdot)$. For a general graph we would be linearly combining the vector representation of the node's neighbors. As we concatenate K such layers, the final vector representation of node i will depend on its neighbors at most K hops away.

Let us now consider matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$, which basically stacks all nodes' vectors $\mathbf{x}_i$, and is termed a graph signal. Computing the matrix product $\mathbf{SX} = \mathbf{Y}$ we end up with another graph signal that aggregates at each node the information of its neighbors, corresponding to the first-order convolution we used in Equation (3.7) (albeit without parameter $\mathbf{\Theta}$, which we will include shortly). By writing $\mathbf{S}^K\mathbf{X} = \mathbf{S}(\mathbf{S}^{K-1}\mathbf{X})$ we may see that this way we aggregate the information K hops away. A general graph convolution is defined simply as a weighted sum of these K signals (i.e. $\sum_k \mathbf{S}^k\mathbf{X}h_k$, where scalars h_k are the taps of the filter). Notice that parameter $\mathbf{\Theta}$ in Equation (3.7) is now interpreted as a filter bank. Indeed, by considering a $d \times d'$ matrix $\mathbf{H}_k$ instead of the scalar taps, a single-layer GNN (or graph perceptron) results of applying a pointwise non-linear function $\sigma(\cdot)$ to this convolution [16, 19]:

$$\mathbf{X}' = \sigma\left(\sum_{k=0}^{K} \mathbf{S}^k\mathbf{X}\mathbf{H}_k\right), \tag{3.8}$$

whereas a deep GNN is constructed by concatenating several perceptrons. Figure 3.8 illustrates the concatenation for deep GNN.

In addition to having empirically shown to provide state-of-the-art performance in a number of important problems [52], GNN's theoretical properties have been intensively studied during the last few years. Important to our problem at hand, are their permutation equivariance (i.e. the output signal on each node is independent of the nodes' ordering), stability (i.e. small perturbations on the graph's edges lead to small perturbations on the output graph signal) and transferability (i.e. one may actually train in a small graph, and as long as the statistic is similar, the performance should remain the same on a larger one) [15, 30].

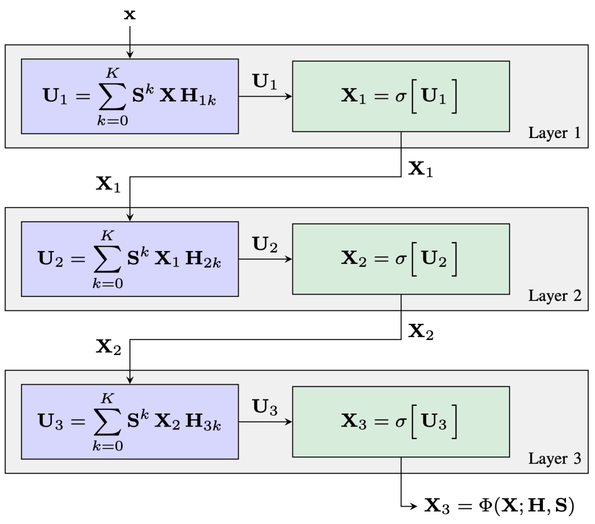

$\mathbf{x}$
$\mathbf{U}_1 = \sum_{k=0}^{K} \mathbf{S}^k \mathbf{X} \mathbf{H}_{1k}$
$\mathbf{U}_1$
$\mathbf{X}_1 = \sigma\left[\mathbf{U}_1\right]$
Layer 1
$\mathbf{X}_1$
$\mathbf{X}_1$
$\mathbf{U}_2 = \sum_{k=0}^{K} \mathbf{S}^k \mathbf{X}_1 \mathbf{H}_{2k}$
$\mathbf{U}_2$
$\mathbf{X}_2 = \sigma\left[\mathbf{U}_2\right]$
Layer 2
$\mathbf{X}_2$
$\mathbf{X}_2$
$\mathbf{U}_3 = \sum_{k=0}^{K} \mathbf{S}^k \mathbf{X}_2 \mathbf{H}_{3k}$
$\mathbf{U}_3$
$\mathbf{X}_3 = \sigma\left[\mathbf{U}_3\right]$
Layer 3
$\mathbf{X}_3 = \Phi(\mathbf{X}; \mathbf{H}, \mathbf{S})$

Figure 3.8: Deep GNN representation by concatenating several convolution layers [30]. In this case the nonlinearity is represented as a separate step applying $\sigma(\cdot)$ to the intermediate signal $\mathbf{U}_i$.

GNNs permutation equivariance property is presented and demonstrated in [15], and it states that graph signal processing with GNNs is independent of node relabeling. Let us formally present the proposition. We start by considering graph shift operators $\mathbf{S}$ and $\hat{\mathbf{S}} = \mathbf{P}^T\mathbf{S}\mathbf{P}$ for some permutation matrix $\mathbf{P} \in \mathcal{P}$. Given a bank of filters $\{\mathbf{H}_{lk}\}$ for each layer $l = 1, ..., L$ and a point-wise non-linearity $\sigma(\cdot)$, define a GNN Φ. Then, for any pair of corresponding graph signals $\mathbf{x}$ and $\hat{\mathbf{x}} = \mathbf{P}^T\mathbf{x}$ used as input to the GNN it holds that:

$$\Phi(\hat{\mathbf{S}}, \hat{\mathbf{x}}) = \mathbf{P}^T \Phi(\mathbf{S}, \mathbf{x}). \tag{3.9}$$

This property will motivate one of the proposed architectures detailed on the next section. As stated in [30], permutation equivariance not only applies to entire reordered graphs, but also to having locally similar structures within a graph. GNNs may learn good representations for a subgraph and transfer that knowledge to other subgraphs with similar structure, allowing to train on some data and expect decent performance on subgraphs that were not considered at training but are seen at inference time. This is also a motivation for some of the experiments detailed in Chapter 5 such as the addition of new zones.

Stability is also an interesting property that motivates some of the experiments from Chapter 5. In this work we present some studies that explore the robustness of the proposed methods in comparison to other more known ML approaches, so the stability property is of great importance.

3.4 Proposed GNN Architectures

3.4.1 Homogeneous Graphs

Let us then go back to our localization problem. We will first consider the APs graph. Although the details on how the graph can be constructed for our particular case are included in Section 4.2, suffice to say at this point that each node is an AP, an edge exist between nodes i and j if the received power between them is above a certain threshold, and that the edge's weight will be the corresponding mean RSSI plus this threshold (so as to turn larger weights to more related signals on the nodes).

Over this graph, we will consider the signal we defined before $\mathbf{X} \in \mathbb{R}^{n_{AP} \times F_{in}}$, corresponding to the RSSI measurements for each AP (with $F_{in} = 2$ if measurements for both $2.4GHz$ and $5GHz$ bands are available). The learning algorithm should then output on what zone (from the n_z possibilities) was that measurement taken. This thus corresponds basically to a graph classification problem.

Graph classification is typically achieved by taking the output of a GNN, passing it through a so-called readout layer (which transforms all nodes signals into a single vector that represents the whole graph) and then applying, for instance, an MLP followed by a softmax [31, 45] . Although typical readout layers (e.g. the sum) enforce permutation equivariance [27], it is clear that which AP had a certain representation is important to infer at which zone was the measurement taken. We have thus used as readout a stacking of the node's representation, resulting thus in a vector $\mathbf{y} \in \mathbb{R}^{n_{AP} F_{out}}$, where F_{out} is the last layer's dimension of the GNN. An illustrative example diagram is displayed in Figure 3.9.

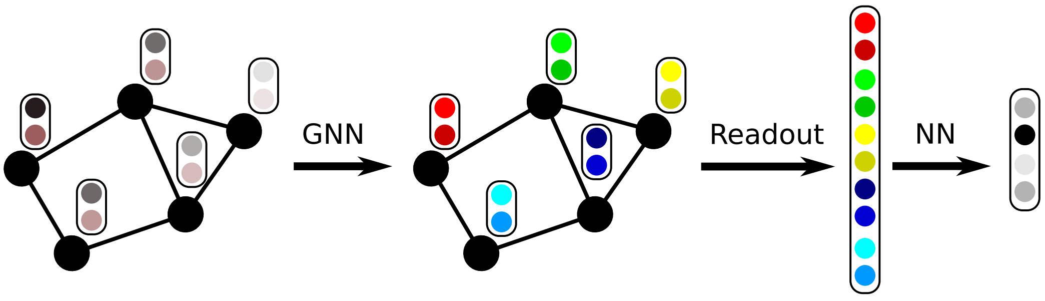


Figure 3.9: Illustrative diagram of the graph classification problem for localization. The signal of a graph with $n_{AP} = 5$ dual band APs (and thus each $\mathbf{x}_i$ has $F_{in} = 2$ dimensions) is processed by a GNN which produces a signal with $F_{out} = 2$ dimensions too. The readout layer then stacks all vectors and passes it through a NN which produces a probability distribution over the $n_z = 4$ zones. Note that training can be performed end-to-end.

It may seem that we have returned to a learning algorithm that drops the geometry of the data, as in traditional learning algorithms. Although this is actually true for the zones, a drawback we will address in the following section, just considering the geometry of the APs will have important performance advantages over traditional baselines, as will be further discussed on Chapter 5.

3.4.2 Heterogeneous Graphs

The question thus remains as to how to include the zones' geometry as an *a priori*. Let us then consider that zones are now nodes on the graph. The first thing to note is that we have no input signal associated to a zone. We may for instance associate an arbitrary scalar to all zones or a one-hot encoded vector [41]. On the contrary, we have an output signal (e.g. a 1 if the measurement was taken at that zone, or 0 if not) which is not present on the APs' nodes. We will thus consider the output signal only at the zones' nodes to compute the loss.

Furthermore, we have at least two types of edges now: the edges between APs (which we have been considering so far) and between an AP and a zone. Although we may define the latter just as the one we used for the inter-APs edges (i.e. the mean RSSI at that zone from that AP plus a threshold), it is clear that in terms of propagating information they should be considered different. We may even include edges between zones, considering for instance their distance or if it is possible to transit from one to the other.

In any case, we are in the context of *heterogeneous graphs* (or networks) and learning therein [10]. The most typical application is in relational data or knowledge graphs, such as academic graphs (where nodes may be papers, authors, conferences, etc.) [38] or recommender systems (where nodes may be users and films, as well as actors, directors, country of origin, etc.) [34]. In our particular case, we want to learn to estimate the zones' signal given the one on the APs, thus turning the localization problem into a graph signal interpolation one.

Graph convolution is easily extended to the heterogeneous case. For instance, Equation (3.7) is now written as [33]:

$$\mathbf{x}'_i = \sigma\left(\mathbf{\Theta}_0^T\mathbf{x}_i + \sum_{r\in\mathcal{R}} \mathbf{\Theta}_r^T \sum_{j\in\mathcal{N}_i^r} S_{j,i}^r \mathbf{x}_j\right), \tag{3.10}$$

where $\mathcal{R}$ is the set of possible relations (e.g. between APs or between an AP and a zone), $\mathcal{N}_i^r$ are the set of neighbors of node i of relation type r, and $S_{j,i}^r$ is the j, i entry in the GSO if the corresponding relationship is of type r (and 0 else). Note that we now have an specific learning parameter $\mathbf{\Theta}_r$ for each type of relation. An illustrative example diagram of the proposed signal interpolation system is presented in Figure 3.10.

Similarly to Figure 3.9, there are $n_{AP} = 5$ dual band APs and $n_z = 4$ zones. The graph now also includes nodes representing these zones (the white ones) and edges connecting them to the APs (the slashed ones), with a weight which may be the mean RSSI as measured in the corresponding zone. The input signal on the APs is the same as before, and we have arbitrarily set all input signals on the zones as a constant. This heterogeneous signal is processed by a heterogeneous GNN, which will produce a vector for each AP (which we will ignore in the training cost) in addition to a distribution over the zones' nodes. Note that information is propagated through all nodes and edges.

Going back to analyzing the properties described in the previous section, this new heterogeneous graph is highly motivated by the permutation equivariance

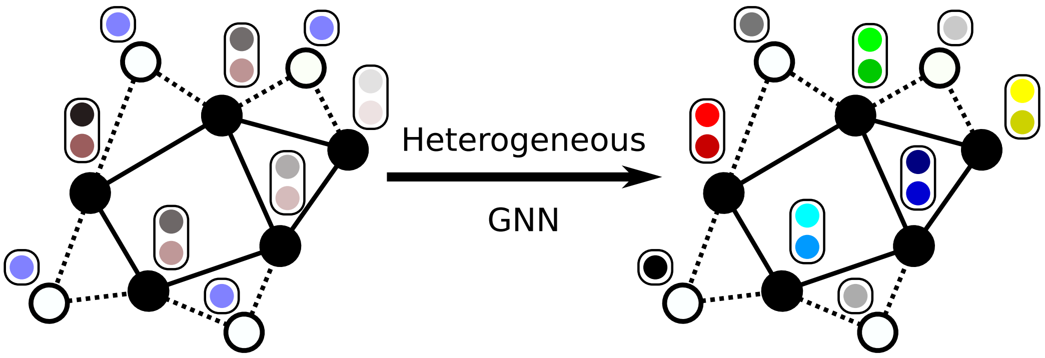


Figure 3.10: Illustrative diagram of the positioning system as a graph signal interpolation problem in the heterogeneous context.

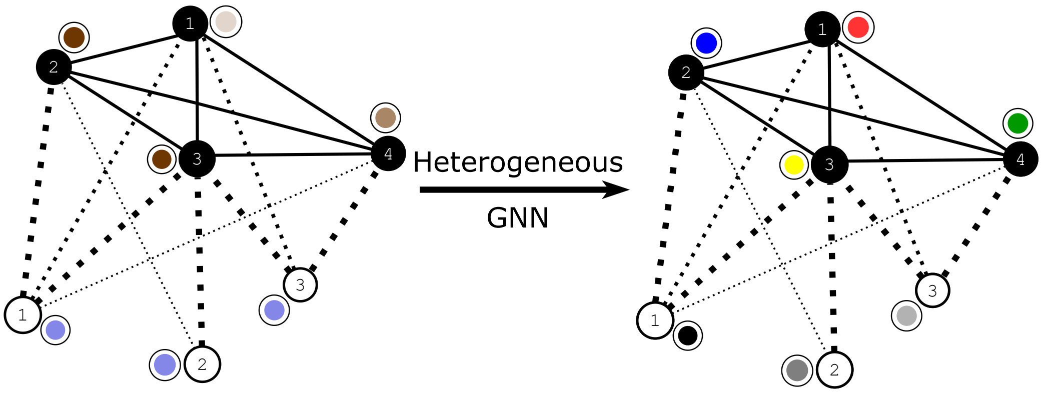


Figure 3.11: Heterogeneous graphs and GNN motivating example. Signal is strong on APs 2 and 3, and after passing through the heterogeneous GNN, the interpolated signal over the zone nodes indicates that the output zone is number 1.

property. Figure 3.11 illustrates the proposed scenario, where we have AP nodes (in black) and zone nodes (in white).

As explained before, the goal is to interpolate the signal on the zone nodes. Figure 3.11 shows a signal example on some AP nodes and a reference signal on zones. We can notice that the signal on APs 2 and 3 is stronger than in the rest of the APs based on the color intensity. After passing this signal through the heterogeneous GNN we will have an interpolated signal over the zone nodes which indicates the zone where the device is most probable to be in. In this case, that is zone number 1.

If we better observe the graph structure, we can notice there are similar structures or subgraphs within the heterogeneous graph. These similar subgraphs are often called *motifs* and are defined by a particular pattern of interactions. Figure 3.12 highlights a relevant motif in this example graph. On the left we can see zone number 1 strongly connected (edges' width represent their weights) to APs number 2 and 3, while also having a weaker connection to AP number 1. The image on the right shows another highlighted subgraph, with a very similar structure to the one described before. Zone number 3 has strong connections to APs number

3 and 4, and a weak connection to AP number 1.

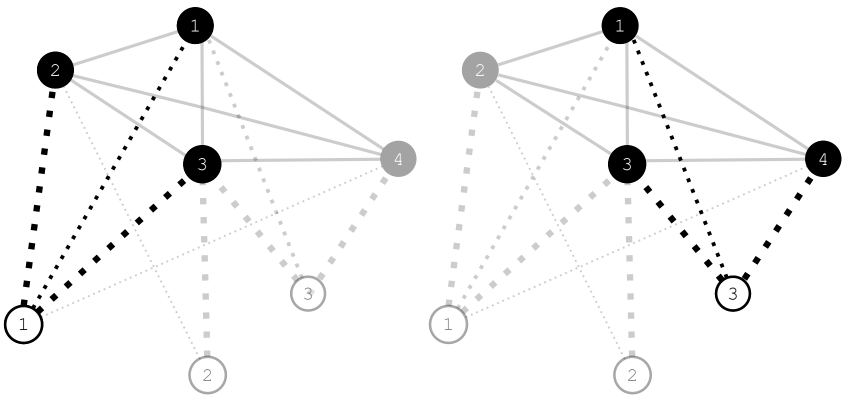


Figure 3.12: Example motif on motivating example graph. On the left we can see zone number 1 strongly connected to APs number 2 and 3. The image on the right shows a similar subgraph highlighted where zone number 3 has strong connections to APs number 3 and 4.

Based on the permutation equivariance property we can expect to achieve decent performance when classifying instances in zone number 3 if we train our model so that it achieves good performance when classifying instances in zone number 1. This example can be seen in Figure 3.13.

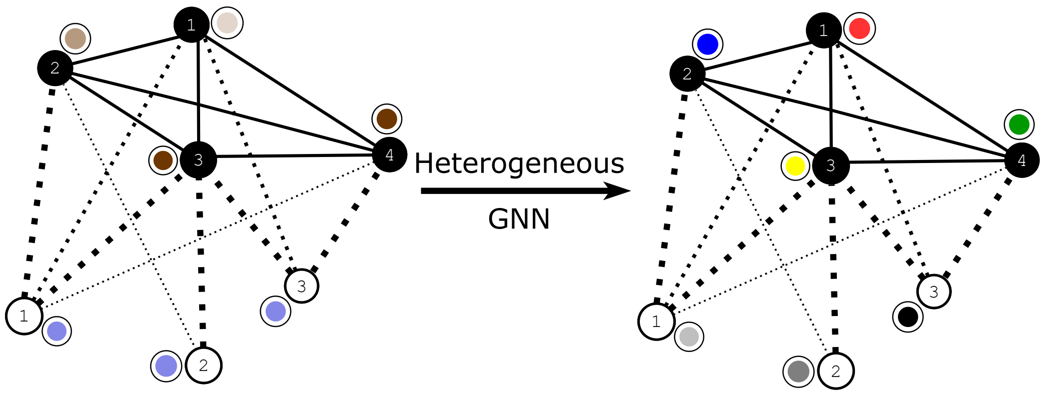


Figure 3.13: Heterogeneous graphs and GNN motivating example. Expected result based on permutation equivariance and stability properties. The signal over the APs is a permutation from the original one and the interpolated signal over the zones indicates that the output zone is number 3.

We have the same graph as in the previous example, but this time the signal on the AP nodes has changed. A strong signal can be found on APs number 3 and 4, while a weaker signal is associated to APs 1 and 2. After applying the heterogeneous GNN to this example we expect to have as an output the interpolated signal over the zone nodes, in particular a strong one over zone number 3.

It is worth to mention that these motifs may not be exactly the same. Based on the construction algorithm that will be described in Section 4.2, and on the deployment of APs and the zones defined we cannot expect to have a perfect match between these subgraphs. But based on the stability property previously mentioned, these motivating examples still make sense and motivate the further exploration done in this work.

3.5 Related Work

As we mentioned before, although there are still not many indoor localization works based on GNNs in the literature yet, they are starting to appear.

In [47] a GNN based method is presented, analyzing a simulated scenario where the distance between nodes is modeled with noisy values of the Euclidean distance between them. Results with a real dataset like [39] (which we present later in this document) can be found in [37], which uses a Graph Convolutional Network (GCN) to address the problem. An advantage of the graph based approach discussed in [6] is the possibility of using transfer learning, adapting a previously trained network with data from other location, thus requiring less number of measurements to train the model.

Finally, it is worth highlighting other variants of the problem, which are different applications where the GNN-based approach is still very useful. On the one hand, [7] studies the case where the signals are not measurements of the power of a Wi-Fi network, but images from multiple cameras of the mobile device. It is clear that the graph model (named Graph Location Networks in this context) still suits the problem, in this case modeling the relationship between images taken at different locations. Another problem extension is addressed by [24], which uses a GNN to estimate the device next location, taking advantage of the graph to model the trajectories given by the device locations sequence.

In the following Chapter we will dive into the datasets used and the implementation details; we will explain how we built these graphs (homogeneous and heterogeneous) based on the available data.

This page has been intentionally left blank.

Chapter 4

Implementation Details

4.1 Datasets

As in every machine learning problem, datasets were a key aspect in this work. As we stated before, building a good dataset is a complex and time consuming task. We should gather a great amount of fingerprints taking care of the diversity of devices used, the distribution of the fingerprints over the area, and the quality of the measurements. Since building a dataset was not in the scope of this work, we opted for using publicly available datasets to test the GNNs. As a result, two datasets were used: MNAV [4] and UJIIndoorLoc [39]. We now briefly describe them.

4.1.1 UJIIndoorLoc

This dataset can be found at *Kaggle*[1] and was used as the official dataset of the IPIN2015 competition [39]. This dataset was designed to test WLAN fingerprinting techniques. The data was acquired in three buildings of the University of Jaume I, each with 4 floors or more and covering an area of over 110,000 m^2. Figure 4.1 shows a map of the buildings. In total, it has 19,937 training samples and 1,111 test samples acquired 4 months after the training data. The dataset was collected by more than 20 users using 25 different models of devices. In addition to RSSI measurements from 520 APs, each measurement is labeled with the corresponding building and floor, along with its longitude and latitude (which were not taken into account here). Note that in this case the positions of the APs are unavailable, highlighting the practical importance of the method to construct the graph we will discuss in Section 4.2.

We started by looking at the available APs to see if we could identify those that contributed with meaningful information. Figure 4.2 shows a histogram of the mean RSSI for the 520 APs available in the dataset. We can see that the majority of the APs have a mean value of -105 dBm and that in the best case the mean values are situated around -99 dBm. This behavior is explained by the

[1] https://www.kaggle.com/giantuji/UjiIndoorLoc

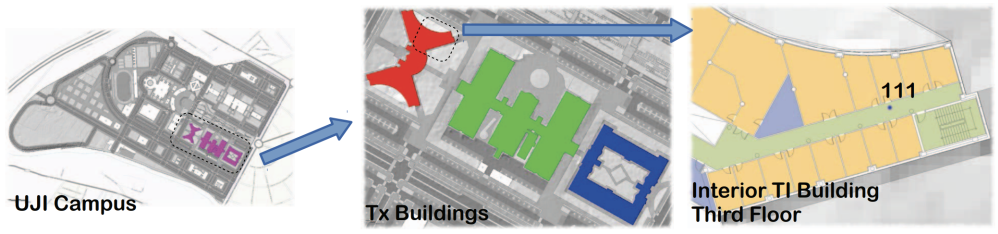


Figure 4.1: UJIIndoorLoc dataset map showing the different buildings and its indoor distribution. [39]

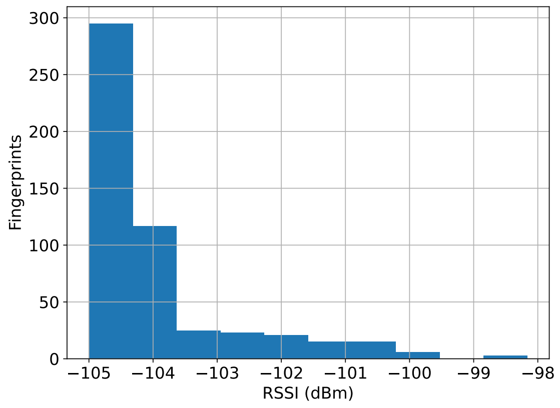


Figure 4.2: Histogram of the mean RSSI for the 520 APs from UJIIndoorLoc dataset.

fact that the area covered by the APs deployment is large and most of the APs can only cover a small part of the total area. This leads to fingerprints having good RSSI values for few APs and near -105 dBm values to almost every other AP. This information is not enough to filter out some APs.

In Figure 4.3 we can see a histogram of the standard deviation RSSI for the 520 APs. It is simple to notice that, although there is a great number of APs that have almost 0 dBm standard deviation, there are some others that can provide useful information. Using this information, together with that from Figure 4.2, we filtered out those APs that did not provide useful information (i.e. the ones that have smaller standard deviation). This resulted in using 253 APs out of the original 520 APs available.

Different definitions of zones may be explored in this case. For instance, a simple and coarse case-scenario is to predict at which building is the device located. We have considered the more interesting scenario where we want to predict both the building and the floor (resulting in 13 zones by combining the building and floor IDs).

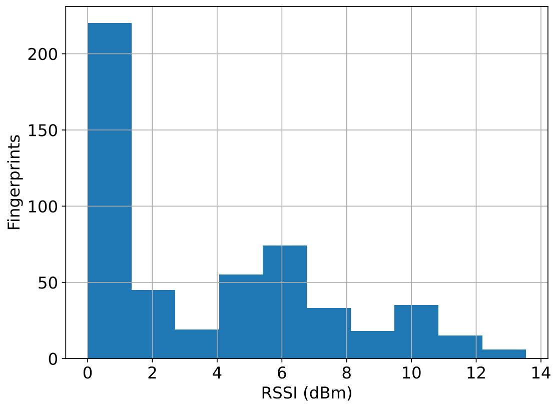


Figure 4.3: Histogram of the standard deviation RSSI for the 520 APs from UJIIndoorLoc dataset.

4.1.2 MNAV

The second dataset we considered here was created within the framework of [4], which sought to provide an indoor localization system to the *Museo Nacional de Artes Visuales* (MNAV, National Museum of Visual Arts) in Uruguay using fingerprinting techniques with Wi-Fi. The dataset is available on GitHub [2]. Figure 4.4 shows a map of the museum including both floors, the position of the deployed APs and the 16 areas that were defined.

The dataset has 10,469 measurements (which we splited 80%-20% for training and testing) from 188 APs, each labeled with the corresponding zone. Inside the museum there are 15 APs, each one using both the 2.4 GHz and 5 GHz bands, thus defining 30 of the 188 APs available in the dataset. The rest are APs outside the museum that the devices found while searching for Wi-Fi networks. In this work, we decided to use only the features corresponding to the APs within the museum, discarding the rest since we did not have control over them and data was noisy.

Figure 4.5 shows the distribution of RSSI for each of the 15 APs on the 2.4 GHz band. It can be noticed that there are some APs, such as the last one, that provide little information since most of its measurements are near -100 dBm. This may be of interest when analyzing the results, as some of the APs may become more relevant to the decision of classifying devices into the zones.

4.2 Graph Construction

Before presenting the results we obtained with the proposed methods, let us discuss how we built the graphs underlying them. Recall that in the graph nodes are the APs (and zones in the heterogeneous case) and the edges' weights should be related

[2]`https://github.com/ffedee7/posifi_mnav/tree/master/data_analysis`

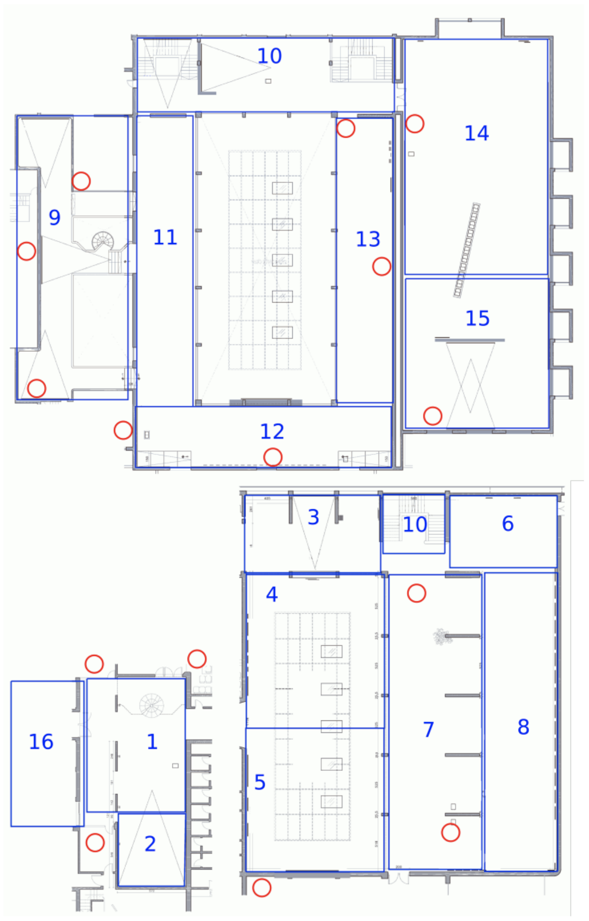


Figure 4.4: MNAV dataset map showing both floors, the distribution of the deployed APs in red circles and the different zones defined in blue numbers. [4]

to the distance between them. If a map with the position of the APs and zones is available, then we may build the graph by considering the inverse (or some other decreasing function) of the distance between them. Note however that we are using the RSSI from all APs as the graph signal in order to infer the corresponding zone. It may well happen that two points are very near to each other but, for instance, separated by a thick wall, meaning that the RSSI as measured in each point may be significantly different.

It would then be more sensible to use an edge weight related to a 'RF distance' between points. To this end, we have used precisely the RSSI between them. If we had access to the area where the positioning system is deployed, we may for instance measure the received RSSI at a given AP from all the rest. However, this was not the case here, as we only had access to the datasets.

We have thus proceeded as follows to construct the homogeneous inter-AP graph. As mentioned before, nodes correspond to the APs and edges connect each node with all other nodes in the graph. Available information (instances

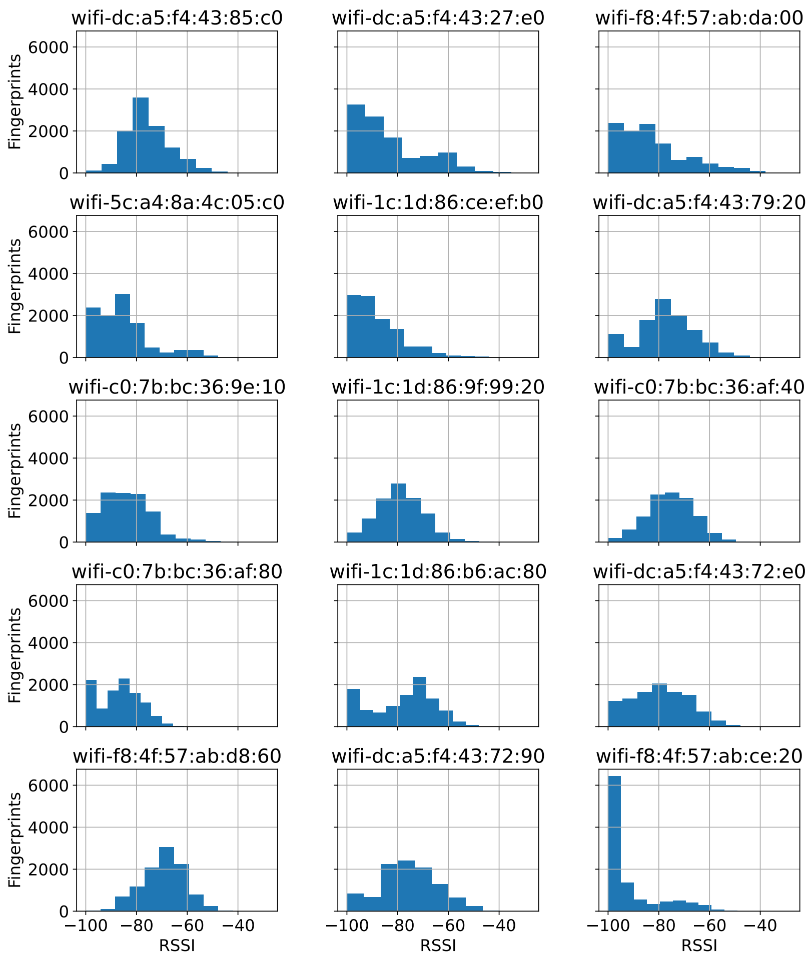


Figure 4.5: Distribution of RSSI measurements on the 15 APs considered (only $2.4GHz$ measurements are plotted) from MNAV dataset.

of our datasets) is used to estimate the weights of the edges. Let us consider a single weight estimation, that of the edge between AP_i and AP_j, as an example, since the same logic can be extended to all edges. We want to set a weight that represents in some way the RSSI between AP_i and AP_j (which as we know, is not directly observable). For that, we only consider the instances $\mathbf{X}$ in the training set which have 'high' RSSI values for AP_i, the rationale being that these would be the best representation of the power measurements that AP_i would take. In practice, this was done by setting a threshold (highest power signal to AP_i minus a certain value named $rssi\_threshold$) and only considering the subset of measurements for which $\mathbf{x}_i$ was above said threshold. Once that subset is obtained, the edge's weight between AP_i and AP_j is simply the mean over this subset of the RSSI for AP_j (i.e. the mean of the corresponding $\mathbf{x}_j$ on this subset).

One last step that was considered during this process was the possibility of applying pruning. This consisted of eliminating edges with small weights (based on a certain *prune_threshold*) since they do not provide much information. Also, this avoids having a complete graph that is heavier to process. The pseudo-code for the proposed method is provided in Algorithm 1.

Algorithm 1 Homogeneous graph construction

```
Input: $train\_data$, $aps$, $rssi\_threshold$, $prune\_threshold$
  for $ap \in aps$ do
      # Get highest power signal to AP $ap$
      $max\_val\_ap \leftarrow \max(train\_data[ap])$
      $ap\_data \leftarrow [\,]$
      for $instance \in train\_data$ do
          if $instance[ap] > (max\_val\_ap - rssi\_threshold)$ then
              $ap\_data.\text{append}(instance)$
          end if
      end for
      # Get mean power signal from AP $ap$ to other APs
      $mean\_ap\_to\_aps \leftarrow \text{mean}(ap\_data)$
      $graph\_edges \leftarrow \text{prune\_edges}(mean\_ap\_to\_aps, prune\_threshold)$
      $\text{build\_edges}(ap, graph\_edges)$
  end for
```

Note that in the case when there are measurements for both bands (i.e. $F_{in} = 2$) we will obtain two weights per edge. Although this is easily accommodated for the methods we discussed before, results do not change significantly when using either of them or both, so we will focus on the results of using a single weight per edge (the one corresponding to the 2.4 GHz band). Furthermore, in order to work with positive weights, we have subtracted the minimum RSSI value to all measurements as a pre-processing step. Note that this way AP pairs may be disconnected on the graph (with a weight equal to zero), effectively reflecting they are far apart. Finally, note that the resulting graph is not necessarily symmetric.

In the case of the heterogeneous graph, edges between zones and APs can be constructed similarly. We have simply considered as an edge weight the mean RSSI as measured at that zone (minus the minimum RSSI). Furthermore, we have used 0 as the input signal on the nodes corresponding to zones (very small changes were obtained using other alternatives such as using ones instead of zeros).

Figure 4.6 shows a resulting homogeneous graph for MNAV dataset without applying edge pruning. The graph is overlaid on the map shown in Figure 4.4 where each node corresponds to an AP and they are numbered according to the number of AP. The width of the edges is proportional to the weight.

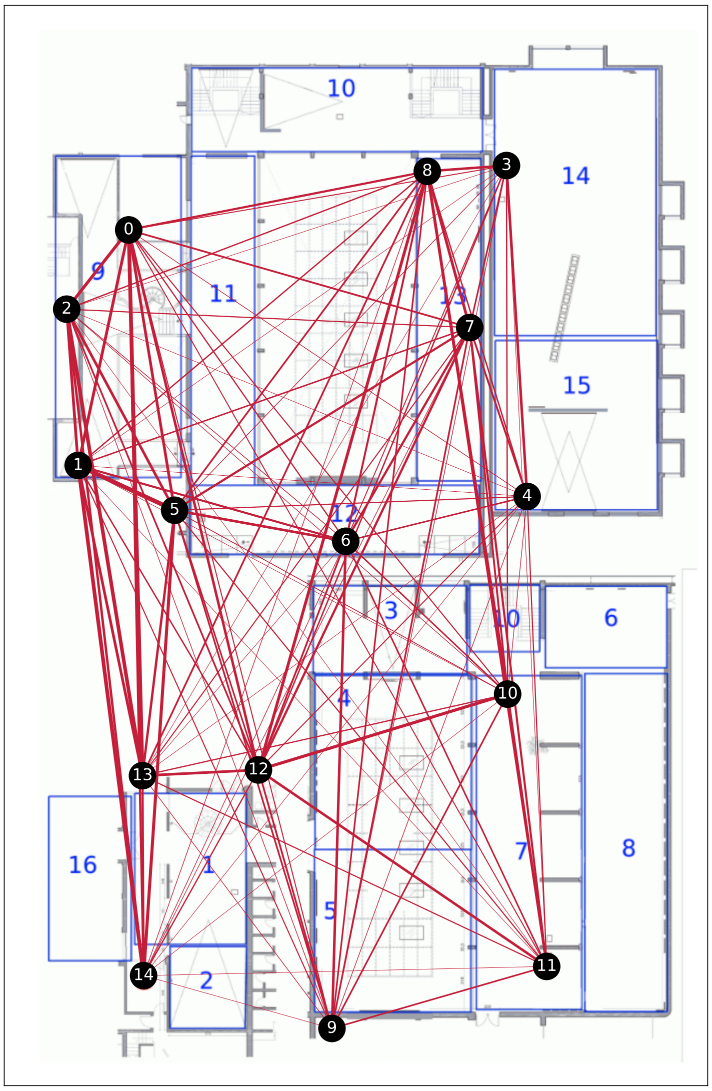


Figure 4.6: Graph for MNAV dataset constructed using the method explained in Algorithm 1 and overlaid on the map shown in Figure 4.4. Some strong connections can be seen between APs while the majority of the APs are connected by thin lines corresponding to weak connections.

This page has been intentionally left blank.

Chapter 5

Experiments and Results

Let us now present the experimental results of the proposed methods, along with some popular baselines. Regarding the latter, we have used KNN, a FCNN and the combination of methods discussed in [4] (which in turn is a small variation on the combination used in the popular FIND3 software[1]), implemented through Scikit-Learn [29].

Since GNNs have gained traction in the last couple of years, several Python libraries appeared to breach the gap between theory and implementation. Some of these were well suited for specific tasks, mostly related to the areas of investigation of the authors. But it quickly raised the need for more stable, robust and complete libraries. This is where some big companies came into the scene and developed libraries such as Pytorch Geometric [14] (now PyG), Tensorflow GNN [13] and Deep Graph Library [42]. We began this work by using the library [2] developed by the Alelab group [3] from the University of Pennsylvania but later decided on using PyG based on the maturity of the library.

Regarding the homogeneous case, the structure of the model consists of two layers with an output dimension of 20, and a filter length of $K = 2$ for UJIIndoorLoc and $K = 3$ for MNAV. In particular, we used the implementation of the architecture proposed in [11] (cf. Equation 3.8). After the readout we used an MLP with the same size as the number of zones (cf. Figure 3.9). In the heterogeneous case we had to resort to the simpler architecture studied in [26], basically a single-tap filter (generalized to heterogeneous graphs in [33] as presented in Equation 3.10), as the more general architecture we used before did not support heterogeneous graphs in PyG. In any case, the final architecture has 4 layers, all with an output dimension of 20 (except, naturally, the last one, which has dimension equal to 1). We also explored a simplified heterogeneous architecture in some of the experiments to be able to better understand the results. As we know, interpretability in neural networks is not straightforward, and GNNs are not the exception. Figure 5.1 shows a simple abstraction of the two heterogeneous GNNs variants.

Hyper-parameters (including mini-batch sizes, learning rates and weight de-

[1]`https://github.com/schollz/find3`

[2]`https://github.com/alelab-upenn/graph-neural-networks`

[3]`https://alelab.seas.upenn.edu/`

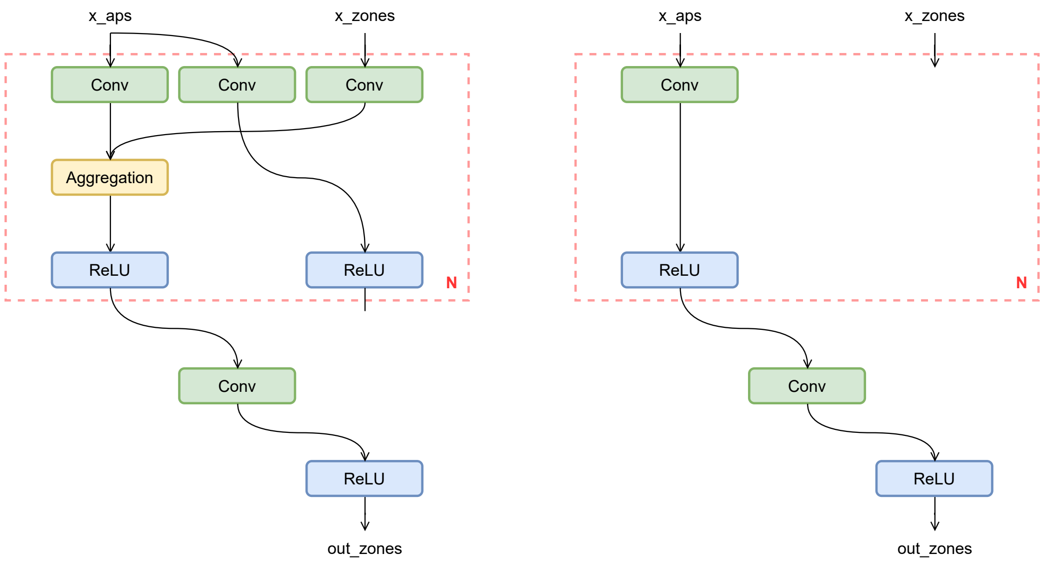


Figure 5.1: Simple diagrams showing the steps involved on the heterogeneous GNNs. The diagram on the left shows how the convolutions are applied on both signals corresponding to APs and to zones. This is done for every hidden layer as represented by the dotted red line. Diagram on the right shows a simplified version where on the hidden layers only convolutions over the inter-APs subgraph are calculated. The last layer is the same on both alternatives. The output interpolated signal on the zone nodes is the result of the convolution applied on the APs-zones subgraph.

cay) were obtained through cross-validation. This was done for the general problem described in Section 3.1 and those hyper-parameters were left unchanged for the entire experimentation phase. We avoided re-calculating hyper-parameters for every subsequent experiments. The idea behind this reasoning is that the experiments included variations in the dataset or in the graph construction phase but were not significant enough to alter the expeted results. All code generated for the experiments is available on GitHub [4].

For several analysis in this work we have used the accuracy metric (Equation 5.1) to compare results. Even though this metric has its drawbacks such as not being well suited for working with imbalanced classes, it is a standard metric that has been extensively used, in particular in works done on the mentioned datasets. It is only fair to compare using the same metric and that is what we have done. In most of the results presented we have also calculated F1-score as a sanity check and understood that there were not discrepancies with accuracy results.

Accuracy is one metric for evaluating classification models. Informally, accuracy is the fraction of predictions our model got right. Formally it is defined as follows:

[4]`https://github.com/facundo-lezama/gnns-indoor-localization`

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.1}$$

where TP = True Positives, TN = True Negatives, FP = False Positives, and FN = False Negatives.

The F1 score can be interpreted as a harmonic mean of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal. The following equation formally defines it.

$$F1 = 2 * \frac{precision * recall}{precision + recall} \tag{5.2}$$

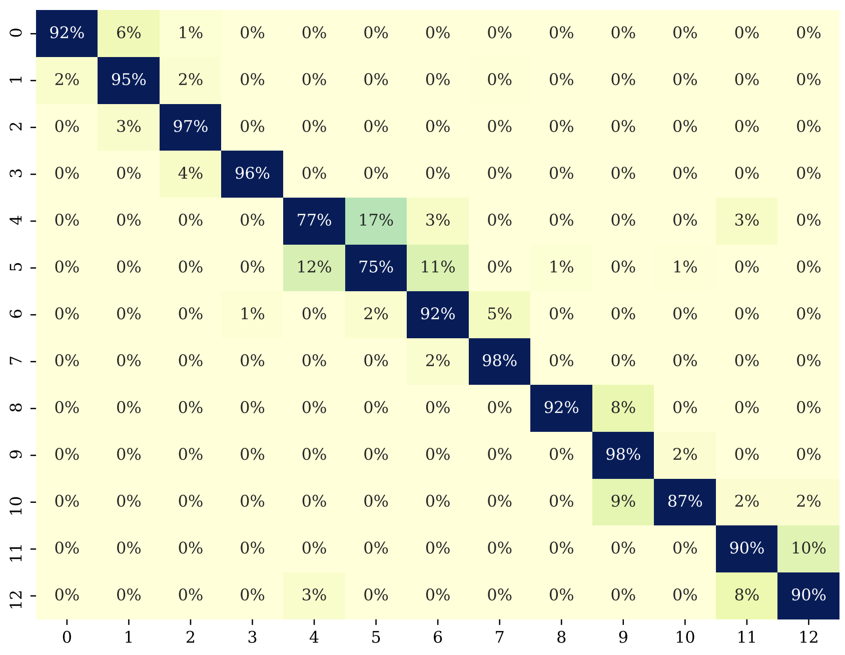
5.1 On the size of the training set

Let us first discuss the overall test accuracy of the different methods on both datasets. Table 5.1 presents these results. There are three important conclusions that may be drawn from the resulting accuracies. Firstly, the performance of all methods is relatively high, with a minimum accuracy of 87.7% for KNN on the UJIIndoorLoc dataset. Secondly, that the homogeneous GNN systematically outperforms the rest of the methods, particularly in the UJIIndoorLoc dataset. Figure 5.2 displays the corresponding confusion matrix for UJIIndoorLoc, where it can be seen that the bigger errors occur when classifying zone 4 as 5 (corresponding to two floors of the second building). Lastly, the heterogeneous GNN presents very competitive results, coming third on the UJIIndoorLoc case (only 0.6% below the FCNN), and second on the MNAV one.

Table 5.1: Classifier accuracy on both datasets.

	Dataset	
Method	UJIIndoorLoc	MNAV
KNN	87.7%	96.2%
FCNN	90.1%	95.9%
Bracco et al. [4]	87.9%	96.3%
GNN (homogeneous)	**93.0%**	**96.7%**
GNN (heterogeneous)	89.5%	96.3 %

Note that, as stated for example in [4], the fingerprint gathering stage is time-consuming and represents a non-negligible part of the total cost of the system. It is then pertinent to evaluate the merits of the different methods in terms of how many training samples they require. To this end, we have considered sub-samples of the training set with varying sizes (uniformly chosen among zones, so that we



	0	1	2	3	4	5	6	7	8	9	10	11	12
0	92%	6%	1%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
1	2%	95%	2%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
2	0%	3%	97%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
3	0%	0%	4%	96%	0%	0%	0%	0%	0%	0%	0%	0%	0%
4	0%	0%	0%	0%	77%	17%	3%	0%	0%	0%	0%	3%	0%
5	0%	0%	0%	0%	12%	75%	11%	0%	1%	0%	1%	0%	0%
6	0%	0%	0%	1%	0%	2%	92%	5%	0%	0%	0%	0%	0%
7	0%	0%	0%	0%	0%	0%	2%	98%	0%	0%	0%	0%	0%
8	0%	0%	0%	0%	0%	0%	0%	0%	92%	8%	0%	0%	0%
9	0%	0%	0%	0%	0%	0%	0%	0%	0%	98%	2%	0%	0%
10	0%	0%	0%	0%	0%	0%	0%	0%	0%	9%	87%	2%	2%
11	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	90%	10%
12	0%	0%	0%	0%	3%	0%	0%	0%	0%	0%	0%	8%	90%

Figure 5.2: Confusion matrix obtained by the homogeneous GNN on the UJIIndoorLoc dataset.

do not incur in an imbalance) and measured the obtained accuracy on the testing set.

In particular, for each sub-sample's size (measured as a percentage of the original training set's size) we have constructed 10 random training sub-samples and report the test results in Figures 5.3 and 5.4 in the form of a boxplot for both datasets. For the sake of presentation clarity, as a baseline we are only showing KNN in this figure, since results are similar or worse for the other methods. For instance, this case-study was also carried out in [4] for their method in the MNAV dataset, resulting in an accuracy as low as 90% when using 30% of the dataset, and 95% when using 70%. For this dataset, all three methods in Figure 5.4 obtain a similar performance but using approximately 50% of the samples, and fare well above an accuracy of 90% when using only 30% of the samples. The superiority of the GNN-based method is clearer in the UJIIndoorLoc dataset (Figure 5.3), where we may see that the homogeneous GNN's performance is above that of the KNN, even when using 30% of the samples. On the other hand, the heterogeneous architecture requires roughly 70% to surpass KNN.

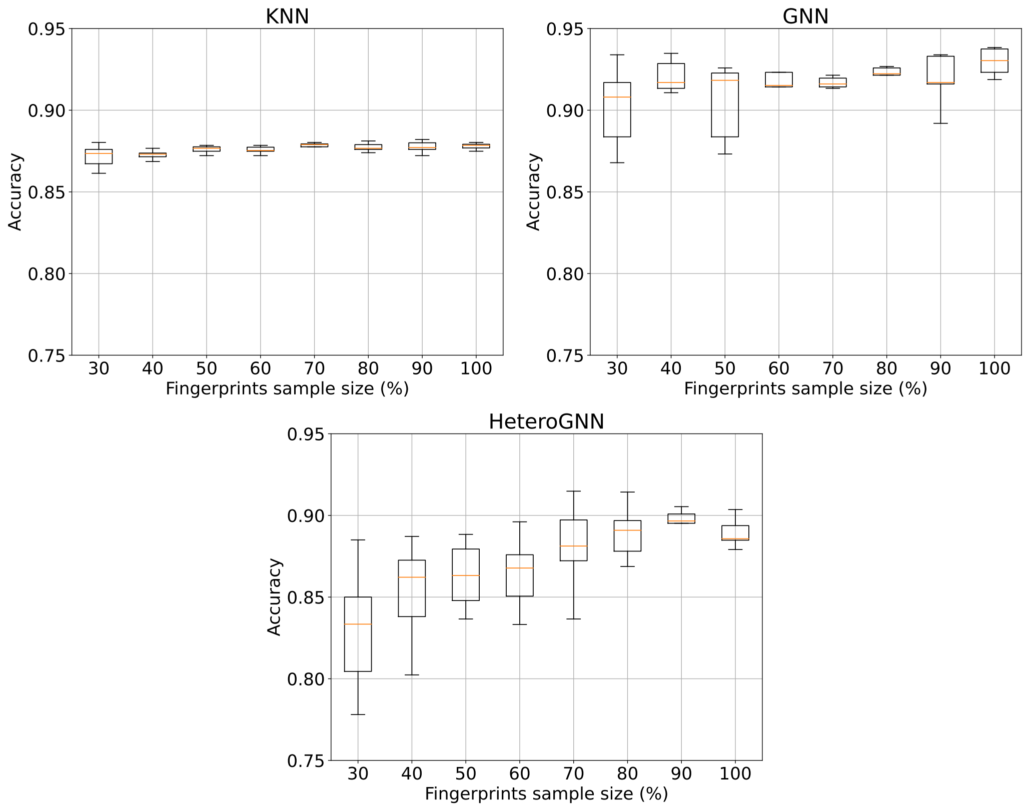


Figure 5.3: UJIIndoorLoc dataset: Accuracy using different amount of fingerprints for KNN (top left), homogeneous GNN (top right) and heterogeneous GNN (bottom). GNN based methods maintain an excellent performance even with very few training samples.

5.2 Propagation environment changes

One of the main drawbacks with fingerprint-based localization methods is the need of updating the system in terms of the propagation environment. For instance, if a wall is built, it will change the RSSI received at certain zones, thus potentially resulting in a lower performance of the localization method. This is remedied by periodically taking new measurements (or at least when such changes occur), which in turn further increases the cost of these deployments.

In this subsection, we will consider a variant of this scenario, where the RSSI received from a certain AP is lower (by a certain constant) than expected. This is achieved by simply decreasing the RSSI corresponding to this AP only on the test set. This scenario may occur if, after the system is trained and deployed, for instance the AP configuration is changed and starts transmitting at a lower power, its antennas are rotated, or simply because it is lowered in height.

Figure 5.5 shows the results for the MNAV dataset when subtracting 5 or 10 dBm to the received RSSI of a single AP. Each boxplot represents the resulting 15 accuracies (one for each AP in the dataset). Note how in this case KNN and the one studied in [4] are the methods that obtain the best results, although

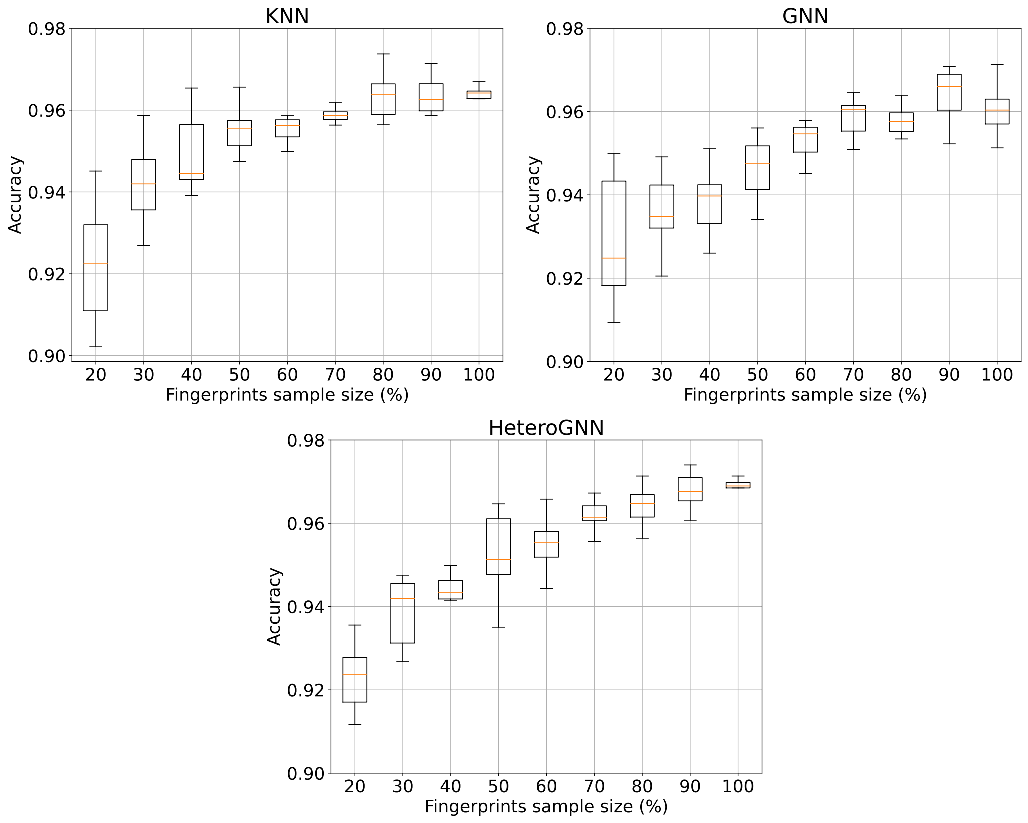


Figure 5.4: MNAV dataset: Accuracy using different amount of fingerprints for KNN (top left), homogeneous GNN (top right) and heterogeneous GNN (bottom). GNN based methods maintain an excellent performance even with very few training samples.

they are closely followed by the homogeneous GNN. Furthermore, although the heterogeneous graph does perform competitively well in the -5 dBm example, its performance is significantly degraded when the offset is -10 dBm. Finally, it is interesting to highlight that FCNN is the method that performs worst in these examples. This shows the importance of considering the structure of the data in the form of the underlying inter-APs graph.

Note that the way we constructed the homogeneous graph (cf. Section 4.2) the tag of which zone corresponds to each measurement is not necessary. This means that we may actually update the edges' weights as clients send the measured RSSI (assuming a centralized scheme, where clients send their measurements and the server answers with the estimated zone). By doing this, we can have a more precise graph given the new state of the deployment. At this point it may be obvious, but the GNN would not need to be modified to work with the new graph, filters can remain unchanged.

A reasonable question is to what point do this updated GSO affects the resulting accuracy in this scenario. Sadly, our experiments show a somewhat marginal gain, of approximately 1%.

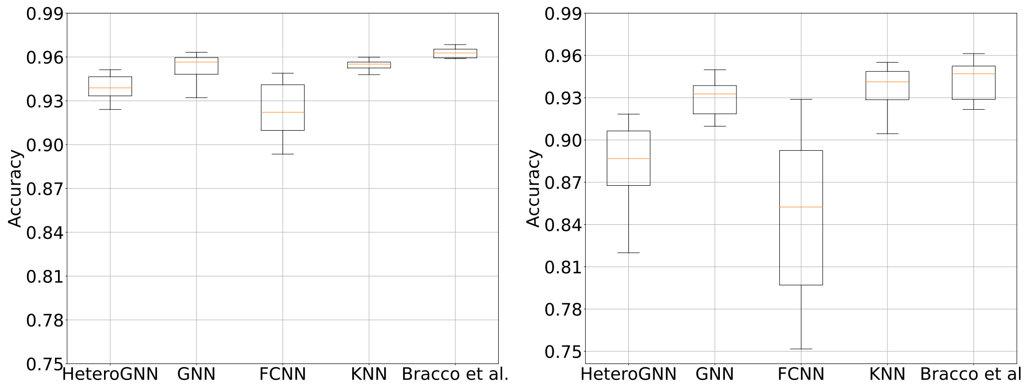


Figure 5.5: Accuracy on the MNAV dataset when an offset of -5 dBm (left) or -10 dBm (right) is applied to the received power of one AP during testing only. Although KNN and [4] perform best, the homogeneous GNN method obtains competitive results.

5.3 AP failures

Let us now consider a typical failure scenario, where one or several APs cease operation. This may happen due to a faulty AP or an electrical power problem on the building. In the latter case, it would affect a group of APs, which are not necessarily all of the ones used for localization. Although this type of problems are typically transient and (should) last for a limited amount of time, it is important to verify if the system still works reasonably well during these problems because we might not have control over its use or the environment. In some application scenarios such as museum guidance through indoor localization, a failure in an AP is not critical to the museum's operation, which would continue normally and localization should still work as good as possible for that precise reason.

Note that this AP failure scenario may be considered as a variation of the one we studied in the previous subsection, where we are basically dropping to zero the transmitting power of a certain group of APs. Quite interestingly, results show that differently from that case, KNN now performs much worse.

We will consider that a certain number of APs fail (1, 2 or 3), which is simulated by simply placing the minimum default RSSI on the entries corresponding to the faulty APs on the test dataset. All possible combinations of APs are considered, and results are reported in the form of a boxplot in Figure 5.6.

The main conclusion to draw from Figure 5.6 is that GNN-based methods tolerate much more gracefully AP failures. Whereas with a single AP failure these methods may still obtain over 90% accuracy (and this may even be the case in the two AP failure scenario, depending on which APs fail), this is not at all the case for baselines (only KNN is shown for the sake of clarity of the figure, but similar results were obtained with the other methods). Even in the case of a single failure, results are typically below 90%, whereas with two AP failures they obtain a performance similar to, or even worse than, the GNN-based methods when three APs failed.

This is a remarkable result since it says the model is fault tolerant, at least

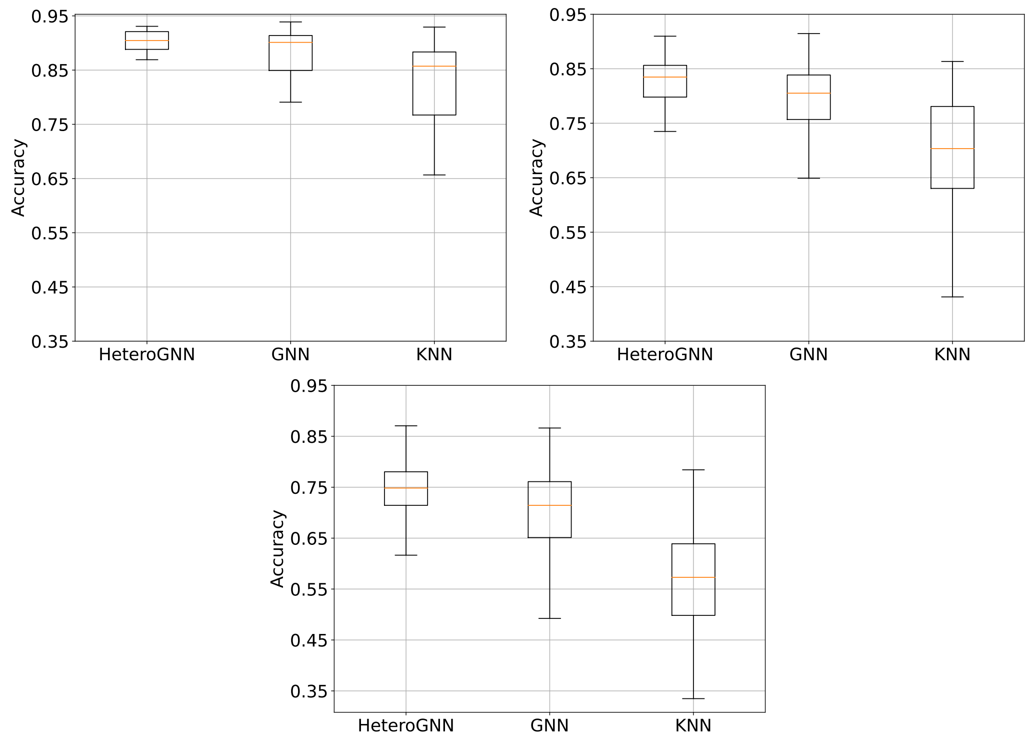


Figure 5.6: Test accuracy on the MNAV dataset when a 1 (left), 2 (middle) or 3 (right) APs fail. All possible combinations of faulty APs are considered. GNN-based methods (particularly the heterogeneous one) perform much better than the baseline.

to a very common fault. Also, as we stated in previous sections, data gathering is one of the most time-consuming tasks and it would add a big effort to take new measurements to accommodate for this transitory scenario. These results are promising and start to validate the initial idea behind most of this work: using spatial information can be leveraged to get better results.

5.4 Adding new zones

It is often the case that deployments change their structure (e.g. a new exposition gets added, walls get teared out, rooms get enlarged) and in this direction we studied the behavior of the model in the scenario of adding a new zone. In the case of having a KNN model, if we do not include new training data classified as the new zone we cannot expect the model to classify any new data with the recently added class. Another case could be having a FCNN. When modifying the last layer to include one new class, the weights associated to this new neuron will be arbitrary and thus it would be hard for the model to correctly classify new instances with the recently added class.

On the other hand, when using the heterogeneous graph we can add a new zone by simply adding a new node in the graph. Based on GNN's theory discussed in Chapter 3, there is no need to modify anything in the GNN to handle this addition. Tap filters remain unchanged and we can try to use the already trained network to classify instances in the new zone. It is clear that to get the best results we should retrain the model, but we want to analyze the performance of the GNN without retraining to understand if we can achieve reasonable results considering the benefits of the model. As we saw in Chapter 3, the permutation equivariance property could lead the model to good results on the new zone if a local structure around it is similar to other in the same graph.

Simulating this scenario was not straightforward. We opted to leave out a zone in the training phase and add it later in the testing phase. For this purpose we simulated leaving out a zone by setting every edge-weight corresponding to the specified zone to 0. Thereby, no signal would be propagated to that node. The training data was also modified to leave out those instances corresponding to the specified zone. On the testing phase, the graph was modified to include edge-weights *from* and *to* the specified zone node. Everything else was left unchanged regarding the graph used for training. Also the testing data included instances labeled with the specified zone.

For this experiment we decided to modify the structure of the heterogeneous GNN to make it simpler. Getting to understand what is actually happening on the heterogeneous GNN as the signal goes through the layers is difficult, so we opted for simplifying it to try to explain its behavior. The new structure of the network makes the convolutions to only consider the inter-APs subgraph, as in the case of the homogeneous graph and homogeneous GNN. Only after the stack of these convolutional layers comes a final convolutional layer where the entire heterogeneous graph is considered and information is aggregated into the zone nodes. This architecture is shown in Figure 5.1.

We started by implementing the addition of a new zone on a simple heterogeneous graph as the one presented on the motivating example in Section 3.3.3 (Figures 3.11 3.12 3.13). To simplify the readers' task we are presenting the same example in Figure 5.7. This controlled scenario gave us the possibility of validating our initial motivation for this specific study. The results were as expected. Figure 5.8 shows the confusion matrices related to training and testing phases. The testing confusion matrix shows that instances corresponding to the new zone are correctly classified, even when the GNN did not see a single instance for that class when training. We also tried some variations to the new zone's edge weights to 'break' the motif and the result was that the GNN could not classify correctly into the new zone (getting almost 0% accuracy and F1 score). This was as expected and encouraged us to continue the exploration on a real scenario.

We tested this scenario for every zone in the MNAV dataset. We chose this dataset since it is simpler than the UJIIndoorLoc one where the number of APs is considerably higher.

The result was somehow as expected. Working with a small graph as the one built from the MNAV dataset, we certainly did not expect to see good performance on every experiment. We saw good performance on some zones (more specifically when adding zones 1, 8 and 15), meaning that performance was not degraded

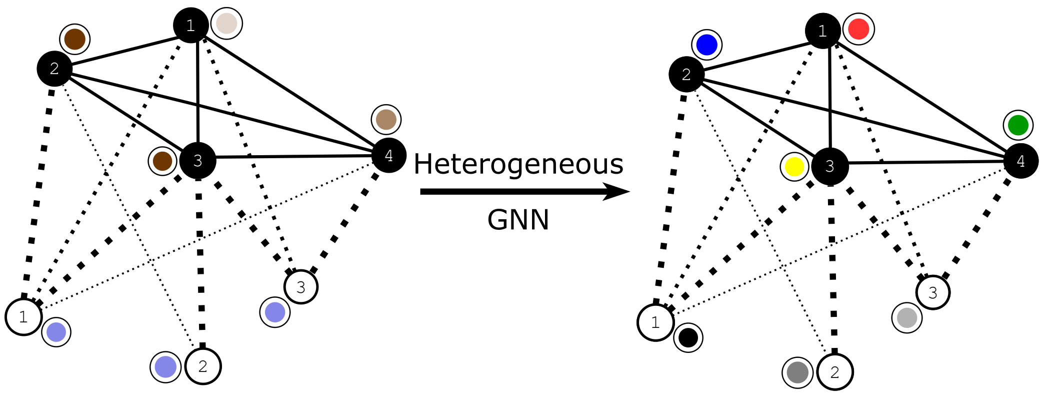
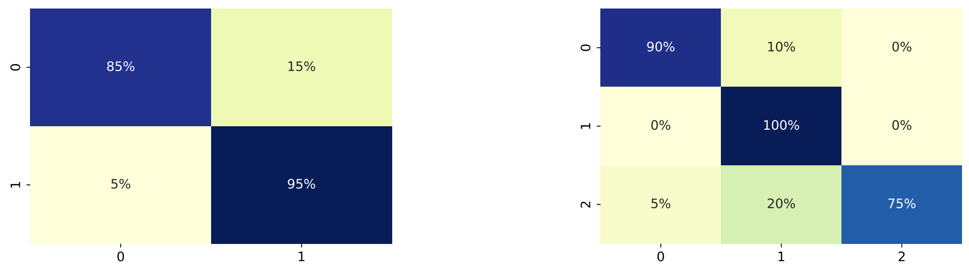


Figure 5.7: Simple heterogeneous graph from Section 3.3.3 motivating example. This example was used for the analysis of adding new zones. Zone number 3 was removed in the training phase and added later for testing.



	0	1
0	85%	15%
1	5%	95%

	0	1	2
0	90%	10%	0%
1	0%	100%	0%
2	5%	20%	75%

Figure 5.8: Confusion matrices corresponding to the addition of a new zone on the motivating example. Matrix on the left shows the training results. Matrix on the right shows the test results where zone number 2 was added on the test phase. The GNN clearly understands how to clasify into this new zone even when it was not trained with that node.

much on the zones that were seen in the training phase and also that the model performed above 50% both in accuracy and F1 score for the newly added zone. On the other hand, the accuracy when adding the rest of the zones was 0% for the added zones while in most cases performance had a small decay on the rest of the zones seen at training. This last result is in agreement with what was observed in the analysis on the simple graph, where the GNN performed poorly if the motif structure changed. One possible explanation may be that the zones that perform poorly do not associate to a motif in the graph. That is, the most relevant subgraph for classifying the added zone does not match the relevant subgraphs for the rest of the zones, so it is difficult for the network to learn to classify the new zone without seeing its instances during training.

We aimed to go further in the interpretation of the results and tried to discover those allegedly similar structures for those zones which achieved good performance. This was not straightforward, specially due to the heterogeneous graph. Figure 5.9 shows how strongly connected are AP nodes and zone nodes. Rows represent APs and columns represent zones, matrix values correspond to the edges' weights

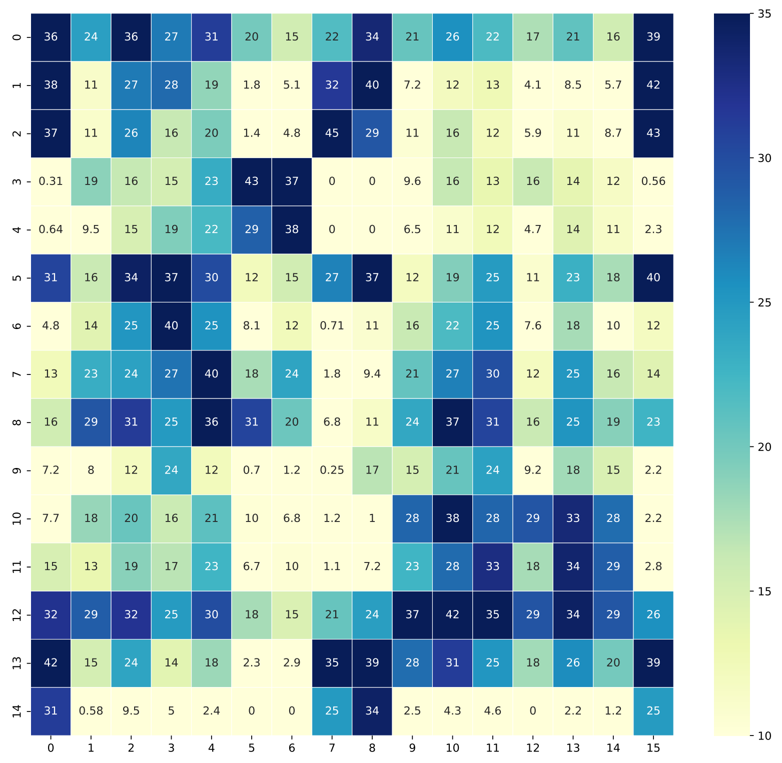


Figure 5.9: This matrix shows how strongly connected AP nodes and zone nodes are. Rows represent APs and columns represent zones while colors represent the magnitude of the edges' weights. Stronger connections have higher numbers. Some small structures between APs and zones can be seen.

which are colored according to a heat-map to better visualize the magnitudes. We can notice some motifs or similar structures composed of connections between zone nodes and AP nodes. For instance, column number 0 (corresponding to zone number 1) has strong connections to APs number 0, 1 and 2, and this structure can also be seen in column number 13 when looking at its connections to APs number 10, 11 and 12. But this is not necessarily enough to ensure good performance as there are several other zones with similar structures and the model does not perform well when adding those zones. This means that in order to have full understanding of the behavior we need to also address the connections between APs and the signals on the APs subgraph. Putting it all together and arriving to conclusions from that information is a difficult task.

Another interesting explainability tool that has been used mainly for computer vision networks are the saliency maps [36]. It relies on calculating the gradient of the loss function for the class we are interested in with respect to the input pixels. In our case, we can calculate in an analogous way the gradient with respect to the input signal on the inter-APs subgraph. With that idea in mind, we analyzed which APs were considered the most important when classifying a zone (by means of saliency maps). We combined that information with the connection strength for each one of those APs to see if they matched (i.e. the most important APs are the ones with the strongest connections). Results showed that in some cases that was true, but in others it was not. The following example illustrates the above. Figure 5.10 shows the contribution of the different APs (rows) to the classification of the zones (columns) obtained using saliency maps in the scenario of adding zone number 15 (column number 6), which resulted in high accuracy and F1 score. Observing column number 6 we can notice that the most important APs are number 4, 8 and 14. When trying to match this information to that in Figure 5.9 we notice that APs 4 and 8 are strongly connected to the node corresponding to zone 15. Nevertheless, AP number 14 has no connection. All in all, it was not possible to conclude an explanation that would hold true for all cases. What is more, the relationship between these two properties (importance and connection strength) was not always consistent; even in cases in which the classification was accurate, the most important node was not among the ones with the strongest connection. This gives us the hint that perhaps analyzing it from this point of view is too simplistic and disregards existing complexities.

There are other studies that have delved into this problem [49], claiming that graphs are problematic structures to address with a simple linear combinations of individual contributions. For example, in our case an AP node might become important because of its neighborhood (the other nodes it is connected with) rather than by its connection strength. Interesting tools specifically for GNN explainability have been developed, like GNNExplainer [49] where an optimization approach is used to find the most influential subgraphs for each classification. However, the implementation for heterogeneous GNNs still needs to mature in order to be used. Explainability is part of what would be interesting to continue exploring to better understand the results and the potential of GNNs for this kind of scenarios.

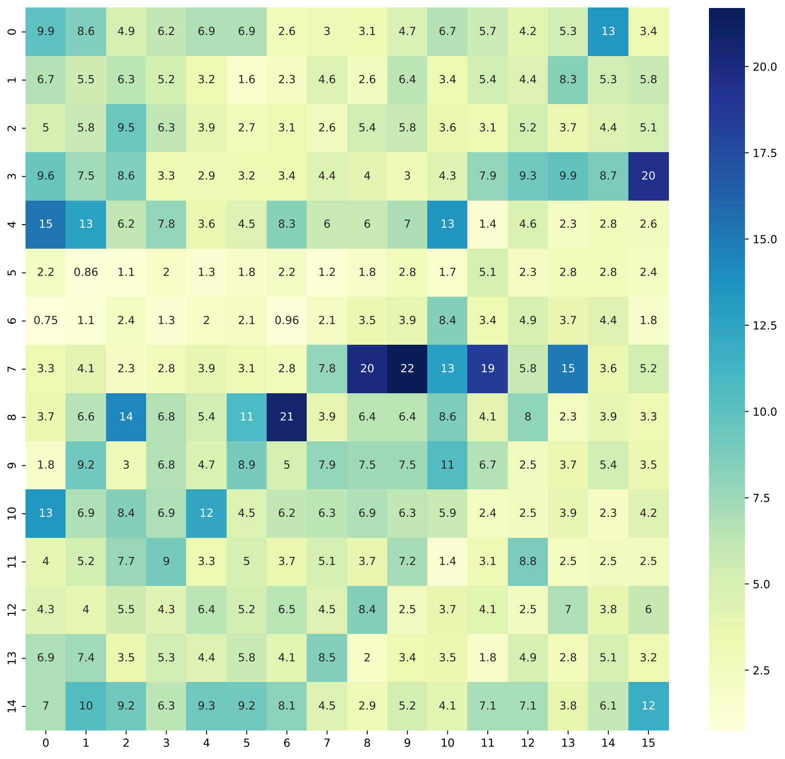


Figure 5.10: This matrix shows how important the different APs are in the classification of the different zones based on saliency maps. Rows represent APs and columns represent zones while colors represent the importance. These values correspond to the scenario of adding zone number 15 (column 6), which resulted in high accuracy and F1 score.

This page has been intentionally left blank.

Chapter 6

Conclusions and Future Work

In this work we have presented an introduction to different techniques used to solve the problem of indoor localization, further deepening in the Wi-Fi fingerprinting technique. This data-driven approach opens the door to the use of machine learning models, avoiding the laborious work of estimating propagation environments and synchronizing equipment.

In particular, we have explored the possibility of applying graph-based learning methods, Graph Neural Networks, to the indoor localization problem. To this day there are not many research works that use this method to approach the mentioned problem. One of our main motivations was to understand if taking the structure of the data and the Wi-Fi deployment into consideration, as it can be done with GNNs, could help in achieving better results.

We have presented the necessary theoretical background, and discussed two ways of constructing the underlying graph: the homogeneous graph and the heterogeneous graph. The homogeneous graph represented the APs deployment and the corresponding homogeneous GNN aimed to generate a graph embedding to further be classified by a MLP. In an attempt to include even more structural data and get rid of the MLP phase, we decided to study the use of an heterogeneous GNN and represented the data in an heterogeneous graph. This graph contains, not only APs as the previous one, but also the zones represented as nodes. We transformed the graph classification problem into a signal interpolation one. This was mainly motivated by the permutation equivariance and stability properties of GNNs.

Two datasets were used for the experimentation phase: MNAV and UJIIndoorLoc. Our first idea to build the graphs for these datasets was to consider neighbor nodes based on physical distances between APs. Since we could not manage to get all the necessary information regarding the Wi-Fi deployment related to those datasets, the graph construction phase was not trivial. We managed to leverage the available training data and estimated distance relationships between APs based on RSSI measurements.

We have then conducted a thorough performance evaluation of the homogeneous and heterogeneous methods, and compared them to popular methods. Results show that GNNs achieve systematically better results than previous methods. We have furthermore evaluated the practical advantages of GNNs by running different experiments. For instance, results show that GNNs gracefully tolerate faulty APs by maintaining high accuracy. We also studied the performance of GNNs when training on a small amount of data and got promising results, showing that even with small amounts, such as 30% of the dataset, accuracy did not suffer much. Lastly we evaluated the scenario of adding a new zone. We started by evaluating the scenario on a simple graph finding that the GNN could learn to classify new classes that were never seen at training. We then followed to evaluate the scenario on the MNAV dataset where results were harder to understand. The GNN performed well on some zones and could not generalize on others. Intuitively we can say that the zones that performed well were associated to motifs in the graph. We aimed to explain the behavior with saliency maps but it was not straightforward. Saliency maps gave us information of the allegedly most important nodes involved on each classification. We tried to combine this information with the graph's structure information to see if they were related. Some zones had strong relationships between both sources of information but some others did not, even in the case that they were classified correctly. Even though results were inconclusive, we understand that there is theoretical background that indicates that GNNs may be good at handling these changes and further work could be done in this direction using more complex tools.

In any case, these results are promising and encourage further research on the use of GNNs to approach indoor localization problems. For instance, other network architectures may be explored (e.g. attention-based), or temporal information may be included to the model. Also by evaluating these methods on other datasets we could better understand some results, since some of the datasets evaluated in this work were small. Another interesting future work would be to find a dataset that has deployment information and be able to construct the graphs without using the training data and compare to the results obtained by constructing the graph the way proposed in this work. In this direction, it would be interesting to explore other methods to build the graph, since it is not a trivial decision.

Part of the work done in this thesis was published as a paper [23] in the URUCON2021[1] conference as well as a chapter in a book that is yet to be published.

[1]`https://www.urucon2021.org/`

Bibliography

[1] Paramvir Bahl and Venkata N Padmanabhan. Radar: An in-building rf-based user location and tracking system. In *Proceedings IEEE INFOCOM 2000. Conference on computer communications. Nineteenth annual joint conference of the IEEE computer and communications societies (Cat. No. 00CH37064)*, volume 2, pages 775–784. Ieee, 2000.

[2] Chaimaa Basri and Ahmed El Khadimi. Survey on indoor localization system and recent advances of wifi fingerprinting technique. In *2016 5th International Conference on Multimedia Computing and Systems (ICMCS)*, pages 253–259. IEEE, 2016.

[3] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[4] Antonio Bracco, Federico Grunwald, Agustin Navcevich, Germán Capdehourat, and Federico Larroca. Museum accessibility through wi-fi indoor positioning. *arXiv preprint arXiv:2008.11340*, 2020.

[5] Ines Chami, Sami Abu-El-Haija, Bryan Perozzi, Christopher Ré, and Kevin Murphy. Machine learning on graphs: A model and comprehensive taxonomy. *arXiv:2005.03675 [cs.LG]*, 2020.

[6] Bing-Jia Chen and Ronald Y. Chang. Few-shot transfer learning for device-free fingerprinting indoor localization. *CoRR*, abs/2201.12656, 2022.

[7] Meng-Jiun Chiou, Zhenguang Liu, Yifang Yin, An-An Liu, and Roger Zimmermann. Zero-shot multi-view indoor localization via graph location networks. In *MM '20: The 28th ACM International Conference on Multimedia, Virtual Event / Seattle, WA, USA, October 12-16, 2020*, pages 3431–3440. ACM, 2020.

[8] Connor W Coley, Wengong Jin, Luke Rogers, Timothy F Jamison, Tommi S Jaakkola, William H Green, Regina Barzilay, and Klavs F Jensen. A graph-convolutional neural network model for the prediction of chemical reactivity. *Chemical science*, 10(2):370–377, 2019.

[9] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

[10] Yuxiao Dong, Nitesh V. Chawla, and Ananthram Swami. Metapath2vec: Scalable representation learning for heterogeneous networks. KDD '17, New York, NY, USA, 2017. Association for Computing Machinery.

[11] Jian Du, Shanghang Zhang, Guanhang Wu, José M. F. Moura, and Soummya Kar. Topology adaptive graph convolutional networks. *CoRR*, abs/1710.10370, 2017.

[12] Joan Bruna Estrach, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and deep locally connected networks on graphs. In *2nd international conference on learning representations, ICLR*, volume 2014, 2014.

[13] Oleksandr Ferludin, Arno Eigenwillig, Martin Blais, Dustin Zelle, Jan Pfeifer, Alvaro Sanchez-Gonzalez, Sibon Li, Sami Abu-El-Haija, Peter Battaglia, Neslihan Bulut, et al. Tf-gnn: Graph neural networks in tensorflow. *arXiv preprint arXiv:2207.03522*, 2022.

[14] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[15] Fernando Gama, Joan Bruna, and Alejandro Ribeiro. Stability properties of graph neural networks. *IEEE Transactions on Signal Processing*, 68:5680–5695, 2020.

[16] Fernando Gama, Antonio G. Marques, Geert Leus, and Alejandro Ribeiro. Convolutional neural network architectures for signals supported on graphs. *IEEE Transactions on Signal Processing*, 67(4):1034–1049, 2019.

[17] W.L. Hamilton. *Graph Representation Learning*. Synthesis lectures on artificial intelligence and machine learning. Morgan & Claypool Publishers, 2020.

[18] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[19] Elvin Isufi, Fernando Gama, and Alejandro Ribeiro. Edgenets:edge varying graph neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021.

[20] Weiwei Jiang. Graph-based deep learning for communication networks: A survey. *Computer Communications*, 185:40–54, 2022.

[21] Shima Khoshraftar and Aijun An. A survey on graph representation learning methods. *arXiv preprint arXiv:2204.01855*, 2022.

[22] Axel Küpper. *Location-based services: fundamentals and operation*. John Wiley & Sons, 2005.

[23] Facundo Lezama, Gastón García González, Federico Larroca, and Germán Capdehourat. Indoor localization using graph neural networks. In *2021 IEEE URUCON*, pages 51–54, 2021.

[24] Hao Lin, Guannan Liu, Fengzhi Li, and Yuan Zuo. Where to go? predicting next location in iot environment. *Frontiers Comput. Sci.*, 15(1):151306, 2021.

[25] Junjie Liu and R Jain. Survey of wireless based indoor localization technologies. *Department of Science & Engineering, Washington University*, 2014.

[26] Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):4602–4609, Jul. 2019.

[27] Nicolò Navarin, Dinh Van Tran, and Alessandro Sperduti. Universal readout for graph convolutional neural networks. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2019.

[28] Michał Nowicki and Jan Wietrzykowski. Low-effort place recognition with wifi fingerprints using deep learning. In *International Conference Automation*, pages 575–584. Springer, 2017.

[29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[30] Luana Ruiz, Fernando Gama, and Alejandro Ribeiro. Graph neural networks: Architectures, stability, and transferability. *Proceedings of the IEEE*, 109(5):660–682, 2021.

[31] Benjamin Sanchez-Lengeling, Jennifer N Wei, Brian K Lee, Richard C Gerkin, Alán Aspuru-Guzik, and Alexander B Wiltschko. Machine learning for scent: Learning generalizable perceptual representations of small molecules. *arXiv preprint arXiv:1910.10685*, 2019.

[32] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[33] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In Aldo Gangemi, Roberto Navigli, Maria-Esther Vidal, Pascal Hitzler, Raphaël Troncy, Laura Hollink, Anna Tordai, and Mehwish Alam, editors, *The Semantic Web*, pages 593–607, Cham, 2018. Springer International Publishing.

[34] Chuan Shi, Binbin Hu, Wayne Xin Zhao, and Philip S. Yu. Heterogeneous information network embedding for recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 31(2):357–370, 2019.

[35] David I Shuman, Sunil K. Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3):83–98, 2013.

[36] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[37] Yanzan Sun, Qinggang Xie, Guangjin Pan, Shunqing Zhang, and Shugong Xu. A novel GCN based indoor localization system with multiple access points. In *17th International Wireless Communications and Mobile Computing, IWCMC 2021, Harbin City, China, June 28 - July 2, 2021*, pages 9–14. IEEE, 2021.

[38] Jie Tang, Jing Zhang, Ruoming Jin, Zi Yang, Keke Cai, Li Zhang, and Zhong Su. Topic level expertise search over heterogeneous networks. *Machine Learning*, 82(2):211–237, 2011.

[39] Joaquín Torres-Sospedra, Raúl Montoliu, Adolfo Martínez-Usó, Joan P Avariento, Tomás J Arnau, Mauri Benedito-Bordonau, and Joaquín Huerta. Ujiindoorloc: A new multi-building and multi-floor database for wlan fingerprint-based indoor localization problems. In *2014 international conference on indoor positioning and indoor navigation (IPIN)*, pages 261–270. IEEE, 2014.

[40] Alex Varshavsky, Anthony LaMarca, Jeffrey Hightower, and Eyal De Lara. The skyloc floor localization system. In *Fifth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom'07)*, pages 125–134. IEEE, 2007.

[41] Clément Vignac, Andreas Loukas, and Pascal Frossard. Building powerful and equivariant graph neural networks with structural message-passing. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 14143–14155. Curran Associates, Inc., 2020.

[42] Minjie Yu Wang. Deep graph library: Towards efficient and scalable deep learning on graphs. In *ICLR workshop on representation learning on graphs and manifolds*, 2019.

[43] Peng Wang, BaoWen Xu, YuRong Wu, and XiaoYu Zhou. Link prediction in social networks: the state-of-the-art. *Science China Information Sciences*, 58(1):1–38, 2015.

[44] Kamin Whitehouse, Chris Karlof, and David Culler. A practical evaluation of radio signal strength for ranging-based localization. *SIGMOBILE Mob. Comput. Commun. Rev.*, 11(1):41–52, jan 2007.

[45] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2021.

[46] Jiang Xiao, Zimu Zhou, Youwen Yi, and Lionel M Ni. A survey on wireless indoor localization from the device perspective. *ACM Computing Surveys (CSUR)*, 49(2):1–31, 2016.

[47] Wenzhong Yan, Di Jin, Zhidi Lin, and Feng Yin. Graph neural network for large-scale network localization. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2021, Toronto, ON, Canada, June 6-11, 2021*, pages 5250–5254. IEEE, 2021.

[48] Ali Yassin, Youssef Nasser, Mariette Awad, Ahmed Al-Dubai, Ran Liu, Chau Yuen, Ronald Raulefs, and Elias Aboutanios. Recent advances in indoor localization: A survey on theoretical approaches and applications. *IEEE Communications Surveys & Tutorials*, 19(2):1327–1346, 2016.

[49] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32, 2019.

[50] Simon Yiu, Marzieh Dashti, Holger Claussen, and Fernando Perez-Cruz. Wireless rssi fingerprinting localization. *Signal Processing*, 131:235–244, 2017.

[51] Faheem Zafari, Athanasios Gkelias, and Kin K Leung. A survey of indoor localization systems and technologies. *IEEE Communications Surveys & Tutorials*, 21(3):2568–2599, 2019.

[52] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

This page has been intentionally left blank.

List of Tables

5.1 Classifier accuracy on both datasets. 45

This page has been intentionally left blank.

List of Figures

2.1 ToA based device localization example. The UD location is estimated as the intersection of the circles centered on the ANs. 6

2.2 TDoA based localization and proximity detection. The UD position is obtained by estimating the intersection of the hyperbolic curves. 7

2.3 AoA based localization diagram [51]. The UD's signal arrives at several antennas in the antenna array and the angles at which it arrives on each antenna are used to intersect and calculate the UD position. 8

2.4 PoA based localization diagram [51]. Sinusoidal signals arrive at antennas in the antenna array with different phases. Knowing the signal's wavelength and assuming a finite distance between the transmitter and receiver, UD position can be estimated. 9

2.5 RSSI based localization. The UD location is estimated as the intersection of the circles centered on the ANs. 10

2.6 Fingerprinting process diagram [2]. The diagram shows the offline phase where training data is obtained and the database is constructed. The bottom part of the diagram describes the online phase where new RSSI measurements are received from an UD and the positioning algorithm outputs the estimated location. 12

3.1 Example diagram showing the classification problem. 17

3.2 Classification example in two dimensions, the classes are coded as a binary variable (Blue = 0, Orange = 1), and then predicted by 1-nearest-neighbor classification. [18] 19

3.3 Classification example in two dimensions, the classes are coded as a binary variable (Blue = 0, Orange = 1). The predicted class is obtained by applying Equation 3.1 using the 15-nearest-neighbors. [18] 20

3.4 Network diagram for the two layer neural network corresponding to Equation 3.6. [3] 21

3.5 Example of an undirected graph. Nodes are represented by circles, which are labeled from zero to four. Edges between nodes are represented by lines. 22

3.6 A molecule is represented as a graph and featurized by its constituent atoms, bonds, and connectivities. The graph embedding is used for predicting odor descriptors via a fully-connected neural network [31]. 25
3.7 Overview of how a single node aggregates information from its local neighborhood. The model aggregates information from node 1's local graph neighbors (i.e. 0, 2 and 3), and in turn, the information coming from these neighbors are based on information aggregated from their respective neighborhoods, and so on. 26
3.8 Deep GNN representation by concatenating several convolution layers [30]. In this case the nonlinearity is represented as a separate step applying $\sigma(\cdot)$ to the intermediate signal $\mathbf{U}_i$. 28
3.9 Illustrative diagram of the graph classification problem for localization. The signal of a graph with $n_{AP} = 5$ dual band APs (and thus each $\mathbf{x}_i$ has $F_{in} = 2$ dimensions) is processed by a GNN which produces a signal with $F_{out} = 2$ dimensions too. The readout layer then stacks all vectors and passes it through a NN which produces a probability distribution over the $n_z = 4$ zones. Note that training can be performed end-to-end. 29
3.10 Illustrative diagram of the positioning system as a graph signal interpolation problem in the heterogeneous context. 31
3.11 Heterogeneous graphs and GNN motivating example. Signal is strong on APs 2 and 3, and after passing through the heterogeneous GNN, the interpolated signal over the zone nodes indicates that the output zone is number 1. 31
3.12 Example motif on motivating example graph. On the left we can see zone number 1 strongly connected to APs number 2 and 3. The image on the right shows a similar subgraph highlighted where zone number 3 has strong connections to APs number 3 and 4. 32
3.13 Heterogeneous graphs and GNN motivating example. Expected result based on permutation equivariance and stability properties. The signal over the APs is a permutation from the original one and the interpolated signal over the zones indicates that the output zone is number 3. 32

4.1 UJIIndoorLoc dataset map showing the different buildings and its indoor distribution. [39] 36
4.2 Histogram of the mean RSSI for the 520 APs from UJIIndoorLoc dataset. 36
4.3 Histogram of the standard deviation RSSI for the 520 APs from UJIIndoorLoc dataset. 37
4.4 MNAV dataset map showing both floors, the distribution of the deployed APs in red circles and the different zones defined in blue numbers. [4] 38
4.5 Distribution of RSSI measurements on the 15 APs considered (only $2.4GHz$ measurements are plotted) from MNAV dataset. 39

4.6 Graph for MNAV dataset constructed using the method explained in Algorithm 1 and overlaid on the map shown in Figure 4.4. Some strong connections can be seen between APs while the majority of the APs are connected by thin lines corresponding to weak connections. 41

5.1 Simple diagrams showing the steps involved on the heterogeneous GNNs. The diagram on the left shows how the convolutions are applied on both signals corresponding to APs and to zones. This is done for every hidden layer as represented by the dotted red line. Diagram on the right shows a simplified version where on the hidden layers only convolutions over the inter-APs subgraph are calculated. The last layer is the same on both alternatives. The output interpolated signal on the zone nodes is the result of the convolution applied on the APs-zones subgraph. 44

5.2 Confusion matrix obtained by the homogeneous GNN on the UJI-IndoorLoc dataset. 46

5.3 UJIIndoorLoc dataset: Accuracy using different amount of fingerprints for KNN (top left), homogeneous GNN (top right) and heterogeneous GNN (bottom). GNN based methods maintain an excellent performance even with very few training samples. 47

5.4 MNAV dataset: Accuracy using different amount of fingerprints for KNN (top left), homogeneous GNN (top right) and heterogeneous GNN (bottom). GNN based methods maintain an excellent performance even with very few training samples. 48

5.5 Accuracy on the MNAV dataset when an offset of -5 dBm (left) or -10 dBm (right) is applied to the received power of one AP during testing only. Although KNN and [4] perform best, the homogeneous GNN method obtains competitive results. 49

5.6 Test accuracy on the MNAV dataset when a 1 (left), 2 (middle) or 3 (right) APs fail. All possible combinations of faulty APs are considered. GNN-based methods (particularly the heterogeneous one) perform much better than the baseline. 50

5.7 Simple heterogeneous graph from Section 3.3.3 motivating example. This example was used for the analysis of adding new zones. Zone number 3 was removed in the training phase and added later for testing. 52

5.8 Confusion matrices corresponding to the addition of a new zone on the motivating example. Matrix on the left shows the training results. Matrix on the right shows the test results where zone number 2 was added on the test phase. The GNN clearly understands how to clasify into this new zone even when it was not trained with that node. 52

5.9 This matrix shows how strongly connected AP nodes and zone nodes are. Rows represent APs and columns represent zones while colors represent the magnitude of the edges' weights. Stronger connections have higher numbers. Some small structures between APs and zones can be seen. 53
5.10 This matrix shows how important the different APs are in the classification of the different zones based on saliency maps. Rows represent APs and columns represent zones while colors represent the importance. These values correspond to the scenario of adding zone number 15 (column 6), which resulted in high accuracy and F1 score. 55