Machine Learning-based MIMO Indoor Positioning

Qiyi Chen
qi2483ch-s@student.lu.se

Department of Electrical and Information Technology
Lund University

Supervisors:
Xuesong Cai, Lund University
Guoda Tian, Lund University

Examiner:
Michael Lentmaier, Lund University

February 12, 2023

Printed in Sweden
Tryckeriet i E-huset, Lund

Abstract

The most widely used positioning system is Global Navigation Satellite System (GNSS), which uses traditional positioning techniques and cannot achieve satisfactory positioning performance in indoor scenarios due to Non-Line-of-Sight (NLoS) transmission. Fingerprinting is a non-traditional positioning technique that is robust to NLoS transmission in indoor scenarios. Moreover, Applying Machine Learning (ML) to fingerprinting positioning can significantly improve positioning performance. Therefore the main objective of this project is to investigate the effect of different Multi-Input Multi-Output (MIMO) antenna topologies, the number of MIMO antennas, ML algorithms, and Channel State Information (CSI) fingerprints on the performance of ML-based fingerprinting positioning. The four open-source datasets used for investigation were measured on the Massive MIMO testbed of ESAT-TELEMIC at KU Leuven. Three datasets were collected when Uniform Rectangular Array (URA), Uniform Linear Array (ULA), and Distributed ULAs as Base Station (BS) under Line-of-sight (LoS) transmission, and one dataset was collected on URA BS under NLoS transmission.

The antenna topologies studied in this project are three 64-antenna topologies and five 8-antenna topologies. The ML algorithms studied are Support Vector Regression (SVR), Fully Connected Neural Network (FCNN), and Convolutional Neural Network (CNN). The fingerprints studied are Channel Impulse Response (CIR) and Channel Frequency Response (CFR). The number of antennas studied is 8-antenna ULA, 16-antenna ULA, and 32-antenna ULA. The positioning error measures the fingerprinting performance, which is the Euclidean distance between the predicted and ground truth coordinates. All comparisons are presented using the empirical Cumulative Distribution Function (CDF) curves of the positioning error.

The investigation results show that increasing the number of antennas of ULA improves positioning performance. CIR fingerprints and CFR fingerprints have comparable positioning performance, 64-antenna URA has the best positioning performance, and the 8-antenna random array has the best positioning performance. The two Deep Neural Networks (DNNs), FCNN and CNN, have much better positioning performance than the traditional ML algorithm, SVR. However, the difference between the positioning performance of the two DNNs is negligible.

Popular Science Summary

Over the past decades, Machine Learning (ML) has become increasingly popular as hardware computing power has increased. ML is widely used to implement Artificial Intelligence (AI), and unlike traditional computer programs, ML algorithms enable computer programs to achieve performance improvements as program inputs are updated automatically. This capability makes ML promising for a wide range of applications in other fields, including wireless fingerprinting positioning. The main objective of this project is to investigate the effect of different Multi-Input Multi-Output (MIMO) antenna topologies, the number of MIMO antennas, ML algorithms, and Channel State Information (CSI) fingerprints on the performance of ML-based fingerprinting positioning.

I will use an example to explain what is the machine learning concept. Suppose a teacher has two stacks of cards. The first pile has pictures of different animals on the front and the animal's name on the back of each card. The second set of cards also has pictures of the same group of animals, but they are taken from different angles and do not have the names of the animals on the back of the cards. The teacher shows the students the front and back of the first pile of cards and teaches them to match the pictures of the different animals with their names by identifying their features. For example, a panda has black and white fur and a big round head, and a giraffe has a long, thin neck and a huge orange-brown body. With the teacher's guidance, the students successfully mastered the ability to match the appearance of different animals with their names. The teacher then showed the students a second set of cards, and they tried to answer the names of the animals after looking at the pictures. In this type of test, students can make mistakes. After all, there is no such thing as a perfect student or teacher. For example, the teacher may not have found the best description of the animal, and the student may not be a good learner. Different students may be good at learning different subjects; some are good at learning from pictures, while others are better at learning from words. The above process of student learning is the same as the process of learning ML algorithms, where the teacher and the cards represent the data set in ML. The first pile of cards with name labels represents the training set, and the second pile of cards without name labels represents the data samples awaiting to be studied. Students represent ML algorithms, and students who are good at learning different subjects represent different ML algorithms.

In this project, the example becomes that of a teacher showing students a

series of CSI and their corresponding location coordinates. Students learn the correspondence between the two, and then students observe CSI they have never seen before and can predict the location coordinates corresponding to this new CSI. The following example can help us understand the relationship between CSI and location coordinates. Imagine a room with a mobile wireless transmitter and a fixed receiver. When a signal is transmitted, the transmitter is at location A, and the receiver receives version A of this signal. Then, the transmitter moves to another location B, in the room and sends the same signal again, and the receiver receives another version B of the same signal. Are the two versions of the received signal the same? The answer is no. During signal propagation, the same signal passes through the wireless channel differently due to the relative positions of the transceivers and the environment. These two wireless channels "distort" the signal differently to different degrees. So when the signal reaches the receiver, it already bears the imprint of the unique wireless channel it has experienced. This different imprint is the CSI, which can be associated with different transmitter locations since different transceiver relative locations constitute different wireless channels. Some ML algorithms are better at learning this correspondence than others, so this project aims to explore and compare which ML algorithms are better at learning the relationship between CSI and its corresponding location. In addition, the measurement system in this project consists of many antennas rather than a single antenna. A MIMO antenna is a receiver that can view the wireless channel from different angles. Topologies and the number of antennas also determine the richness of the CSI from the angle perspective. The CSI has different expressions, which are different fingerprints, and the choice of fingerprints also affects the positioning performance. Therefore, this project also investigates the effect of different topologies of MIMO antenna and the number of antennas and fingerprints on the positioning performance of different ML algorithms.

Acknowledgements

I want to thank my mom, Guoying, my dad, Liang, and my sister and brother-in-law, Meiyi and Tengyuan. Without the unconditional support of my family, I could not have persevered until today. I would also like to thank my supervisors, Xuesong Cai and Guoda Tian. They have provided me with much guidance regarding expertise and experience. I would also like to thank all the teachers at Lund University who taught me during my master's program. They are very professional and provide me with quality education. I am very grateful to Cecilia Bruhn, the international coordinator at Lund University, for all the help and care she has given me. Finally, I thank all my dear friends Yewen, Xunbing, Kean, Yijie, Florentin, and Marcel. I cherish and miss all the good times we had together.

Sincerely,
Qiyi Chen

List of Abbreviations and Acronyms

AI Artificial Intelligence

ANN Artificial Neural Network

AoA Angle of Arrival

BS Base Station

CDF Cumulative Distribution Function

CFR Channel Frequency Response

CIR Channel Impulse Response

CNC Computer Numerical Control

CNN Convolutional Neural Network

CSI Channel State Information

Distributed ULAs Distributed Uniform Linear Arrays

DNN Deep Neural Network

FCNN Fully Connected Neural Network

FPGA Field Programmable Gate Array

GCNN Gated-Convolutional Neural Network

GNSS Global Navigation Satellite System

IFFT Inverse Fast Fourier Transform

KNN K-Nearest Neighbors

LBS Location-Based Service

LoS Line-of-Sight

LR Logistic Regression

LTE Long-Term Evolution

MIMO Multi-Input Multi-Output

MLP Multi-Layer Perceptron

ML Machine Learning

NLoS Non-Line-of-Sight

OFDM Orthogonal Frequency Division Multiplexing

PCA Principal Component Analysis

RBF Radial Basis Function

RFID Radio Frequency Identification

RMSE Root Mean Square Error

RNN Recurrent Neural Network

RSS Received Signal Strength

SNR Signal-to-Noise Ratio

SVR Support Vector Regression

SVM Support Vector Machine

TDD Time Division Duplex

TDoA Time Difference of Arrival

ToA Time of Arrival

ToF Time of Flight

UE User Equipment

ULA Uniform Linear Array

URA Uniform Rectangular Array

USRP Universal Software Radio Peripheral

UWB Ultra-Wide Band

WLAN Wireless Local Area Network

WPAN Wireless Personal Area Network

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Background and Motivation

With the development of wireless communication technology, wireless positioning as an auxiliary wireless communication technology has been rapidly developed. Indoor positioning has always had an excellent economic prospect, and its market is expanding. Many fields include navigation, robotics, personal safety and health, industrial monitoring, and control, military, etc. The demand for high-precision Location-Based Service (LBS) is also increasing. Global Navigation Satellite System (GNSS) is a widely used wireless positioning system using traditional techniques. GNSS has excellent positioning performance in outdoor scenarios under Line-of-Sight (LoS) transmission conditions. However, the poor positioning performance when using GNSS for indoor positioning is because traditional positioning techniques use geometric mapping techniques and the physical properties of radio signals to achieve positioning purposes. Multipath interference, shadowing effects, fading, and transmission delays affect signal transmission in indoor environments [1].

The Machine Learning (ML) based Multi-Input Multi-Output (MIMO) fingerprinting localization technique has more advantages than the traditional localization techniques. It is promising for indoor scenarios because it can utilize multipath transmission for localization and is robust to Non-Line-of-Sight (NLoS) transmission environments. Moreover, since it does not rely on geometric mapping techniques, only a fixed Base Station (BS) is required for fingerprinting localization. Fingerprinting localization can take advantage of existing communication systems without rebuilding hardware. ML algorithm can improve the performance of fingerprinting localization. Due to the high angular resolution of Massive MIMO systems, their application in wireless localization techniques is promising. Using Channel State Information (CSI) measured by a Massive MIMO Orthogonal Frequency Division Multiplexing (OFDM) system based on cellular bandwidth as a fingerprint makes the fingerprinting technique compatible with current Long-Term Evolution (LTE), sub-6 GHz 5G, and WiFi communication systems [1]. In addition, fingerprinting positioning has been used in many indoor positioning systems and can provide good positioning services, such as Bluetooth and Ultra-Wide Band (UWB) systems. Different positioning technologies are often used together to provide better positioning services.

ML-based fingerprinting positioning has many advantages, but different ML algorithms are based on different mathematical principles and have different localization performances. The input to the ML algorithm affects its output and localization performance. Different fingerprints, array antenna topologies, and the number of array antennas indicate different forms of ML input. Therefore, this project investigates the impact of different ML algorithms, fingerprints, array antenna topologies, and the number of array antennas on fingerprinting localization performance. It is hoped that the findings of this project will bring insight to researchers in related fields.

1.2 Previous Works

The authors in [2] proposed a customized neural network structure for a CSI-based MIMO indoor localization system, which contains an additional phase branch as a feature extractor, minimizing the training data required compared to previous findings. An improved Convolutional Neural Network (CNN) algorithm was proposed in [3], also known as Gated-Convolutional Neural Network (GCNN), with a better ability to extract time-varying features embedded in the received signal. The GCNN can achieve Root Mean Square Error (RMSE) of less than 0.08 m and 0.3 m for 16 and 2 antennas, respectively. In addition, [3] evaluated the performance of the Fully Connected Neural Network (FCNN) algorithm for localization. The author in [4] presents a Recurrent Neural Network (RNN)for centimeter-level indoor localization that considers user trajectories and Signal-to-Noise Ratio (SNR) information. The conclusions show that the network proposed in [4] outperforms the state-of-the-art neural networks when a small training data set is used. An extensive comparison between neural network-based and decision tree-based localization methods was also performed. Simulation results show that the neural network has higher estimation accuracy than the decision tree-based approach. A Deep Neural Network (DNN)-based indoor localization fingerprinting system using CSI, called DNNFi, is proposed by [5]. Maintaining a single DNN at different reference points allows for faster online inference calculations. Experimental results show that DNNFi can effectively reduce localization errors compared to traditional CSI localization fingerprinting methods. The authors in [6] proposed an improved localization algorithm based on CSI and Received Signal Strength (RSS). RSS is used to estimate the distance and narrow the search range based on this estimate. The location is then accurately estimated based on the Support Vector Machine (SVM) algorithm. In addition, the effect of SVM key parameters on localization error is better evaluated by fingerprint database and online data. A sample expansion method is also proposed in [6]. Experimental and simulation results show that the improved localization algorithm can reduce the localization error within 2m. The authors in [7] proposed a K-Nearest Neighbors (KNN) method to estimate the location of CSI in indoor environments. The proposed method outperforms the three state-of-the-art methods based on Multi-Layer Perceptron (MLP) and CNN. This study's extension includes analyzing the estimated dataset's upper bound on the localization accuracy. [8] proposed an optimization algorithm for KNN. Theoretical analysis and experimental results show that the

KNN optimization algorithm has improved positioning accuracy compared with the original algorithm. The optimization algorithm improves positioning accuracy by sacrificing a part of the time cost without affecting the real-time performance of the positioning.

1.3 Purpose of Project

This project uses four open-source Ultra Dense Indoor Massive MIMO CSI datasets to evaluate the impact of different ML algorithms, fingerprint types, antenna topologies, and the number of Uniform Linear Array (ULA) antenna elements on fingerprinting localization performance. The investigated ML algorithms include the traditional ML algorithm Support Vector Regression (SVR) and two DNNs, FCNN and CNN, respectively. The two CSI fingerprints investigated are Channel Frequency Response (CFR) and Channel Impulse Response (CIR). Three 64-antenna topologies are investigated: ULA, Distributed Uniform Linear Arrays (Distributed ULAs), and Uniform Rectangular Array (URA). Five 8-antenna topologies are investigated: horizontal uniform linear array, vertical uniform linear array, diagonal uniform linear array, block array, and random array. 8-antenna ULA, 16-antenna ULA, and 32-antenna ULA were investigated to evaluate the effect of the number of antennas of ULA on positioning performance.

All investigations are based on datasets of real measurements. ML-based fingerprinting localization was implemented by the python-based scikit-learn and PyTorch libraries in this project. The KU Leuven large-scale MIMO testbed recorded the four open-source datasets used in this project. Each dataset contains 252004 CSI samples, and their corresponding User Equipment (UE) ground truth coordinates. The BS of the single-cell Massive MIMO OFDM testbed is equipped with 64 antennas and is designed for flexible topology changes. The testbed has four UEs moving in the 1.25 m $\times$ 1.25 m area. The UEs move 5 mm at a time and send pilot signals, where the spatial position of the UEs is recorded. The 64 antennas of the BS simultaneously receive pilot signals for channel estimation to obtain CSI. The pilot signals have 100 OFDM subcarriers spaced uniformly in frequency over a 20 MHz bandwidth. Three datasets were measured using URA, ULA, and Distributed ULA under the LoS transmission, and one dataset was measured using URA under NLoS transmission. The CSI obtained from a single measurement is a complex matrix with the shape of 64 $\times$ 100. The UE is shifted by five milliseconds per measurement so that the resulting dataset is fine-grained.

The investigation results show that the CIR and CFR fingerprinting localization performances are comparable, and 64-antenna URA and 8-antenna random arrays have the best localization performance. FCNN algorithm and CNN algorithm have comparable localization performance, but the localization performance of these two DNNs is much better than the localization performance of the SVR algorithm. Increasing the number of antennas of the antenna array can improve the performance of ML-based fingerprinting localization.

1.4 Thesis Outline

Chapter 2 of this thesis details the background and principles of traditional wireless positioning techniques and fingerprinting techniques. Chapter 3 details the background of ML and deep learning and the principles of three ML algorithms. Chapter 4 presents the detailed system setup and dataset collection details for CSI dataset measurement campaigns using a Massive MIMO OFDM communication system and data processing. Chapter 5 shows the key findings based on real measurement data, comparing and analyzing fingerprinting performance under different settings. Chapter 6 summarizes all the essential findings and insights of this project and discusses possible and promising further research in the future of this project.

Chapter 2
Wireless Positioning

2.1 Introduction

The essential function of a wireless communication system is to transmit information from one wireless communication device to another, and the two devices usually have different geographical locations. By exploiting the physical nature of radio signals, wireless communication systems have developed additional wireless positioning capabilities. Wireless communication systems can be used to determine the relative positions of the transmitter and receiver in space. Wireless positioning is an auxiliary communication service whose applications are often referred to as LBS, used in numerous fields such as navigation, robotics, personal safety and health, industrial monitoring and control, military, etc. With the development of communication technology and the increase in human demand for LBS, the requirements for the performance of wireless LBS are also increasing. LBS has become a necessary feature of wireless communication systems, and improving the positioning performance in different application scenarios has become an important research goal. The most widely used wireless positioning system is GNSS, which provides high-precision positioning services through long-range wireless links, and the accuracy of GNSS can reach a centimeter-level under outdoor LoS transmission. Due to NLoS transmission and multipath transmission, GNSS does not perform satisfactorily in indoor scenarios and urban environments with dense buildings. Therefore, it is significant to investigate wireless positioning technologies that can provide high positioning accuracy in indoor environments. Other common wireless platforms with positioning capabilities are Radio Frequency Identification (RFID), Wireless Local Area Network (WLAN), and Wireless Personal Area Network (WPAN)[9].

2.2 Wireless Positioning Techniques

Wireless positioning techniques can be classified as traditional positioning techniques and database comparison-based positioning techniques. Traditional wireless positioning techniques use the physical propagation characteristics of electromagnetic waves and geometric mapping techniques to determine location. Traditional wireless positioning techniques can be classified according to the different geometric mapping techniques as distance-based positioning techniques, such as Time of

Flight (ToF) and RSS, and angle-based positioning techniques, such as Angle of Arrival (AoA). In contrast, database comparison-based positioning techniques do not require geometric mapping techniques for position determination; it achieves positioning by mathematics algorithms that compare the CSI fingerprints measured in real-time with the CSI fingerprints in the database. Each fingerprint corresponds to a location in space, so this positioning technique is also called fingerprinting or pattern recognition. There are many options for algorithms used to compare fingerprints measured in real time with those in the database, and this project uses ML algorithms to implement fingerprinting. In practical applications, using only a single positioning technique is rare. A mixture of positioning techniques is usually used to achieve specific system requirements [9]. The choice of a specific positioning technique should be based on the needs of the application scenario, such as system complexity, positioning accuracy, and cost. Positioning tasks can be classified as two-dimensional and three-dimensional positioning, but the implementation of both tasks is based on the same principles. This project investigates only two-dimensional positioning, so the following methodological discussion is based only on the two-dimensional plane. In the two-dimensional positioning scenario, positioning techniques using geometric mapping are required to determine the mobile terminal's location using at least two fixed wireless terminals with known locations.

2.2.1 Time of Flight

The distance between the UE and the BS equals the signal's ToF multiplied by its propagation speed [9]. The ToF of a signal is the electromagnetic propagation time, while the propagation speed of the signal is the speed of light. Both Time of Arrival (ToA) and Time Difference of Arrival (TDoA) are extensions of the ToF technique. The ToA technique uses the transmission time of the signal between transceivers to find the distance, which requires precise time synchronization between the UE and the BS. The TDoA technique measures the difference in the signal's arrival time at the BSs to calculate the position, which requires precise time synchronization between multiple BSs [9].

Figure 2.1 shows how positioning can be achieved based on the geometric mapping. A, B, and D are known fixed BSs. C is a mobile UE whose location is unknown. In figure 2.1, all four terminals are in the same plane. The distance between AC and BC can be obtained by measuring ToA or RSS. The radius of the three circles in the figure is the distance between the three BSs to the UE. The intersection of the three circles is the position of C. If only two fixed BSs are used, an erroneous position is generated in most cases because the position relationship of the two circles sometimes leads to two intersections, so to solve this problem, either additional information about the UE is used or another fixed BS is added for eliminating the erroneous position.

2.2.2 Received Signal Strength

There is a relationship between the distance between transceivers and the RSS [10]. Generally, the RSS decreases as the distance between transceivers increases.

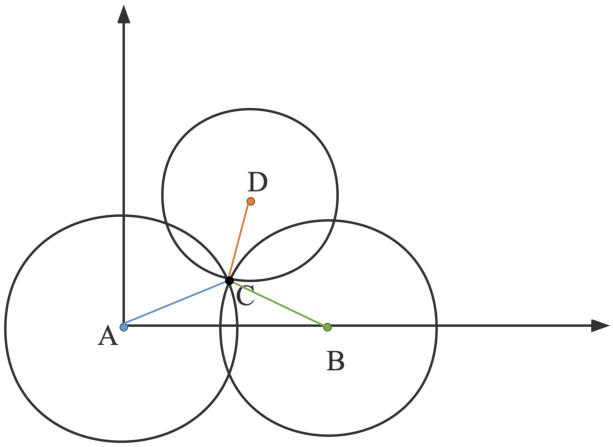

Figure 2.1: Distance-based geometric mapping.

RSS has several advantages over the ToF method: it can be directly applied to existing wireless communication systems and requires little addition or change to the hardware of existing wireless communication systems. Moreover, it does not require synchronization between transceivers, so its implementation cost is meager. However, the RSS method also has some disadvantages, as the signal experiences interference, multipath transmission, and NLoS transmission during the actual wireless communication. These channel effects can lead to large fluctuations in signal strength, so RSS methods are usually less accurate than ToF methods. In free space, the Friis equation (2.1) expresses the relationship between the received power and the transceiver distance [10].

$$P_r = \frac{P_t G_t G_r \lambda^2}{(4\pi)^2 d^2} \tag{2.1}$$

G_t and G_r are the antenna gains of the transmitter and receiver. λ is the wavelength of the transmitted signal, d is the distance between the transceiver, P_t is the transmit power, and P_r is the receive power. When the transmitter power and antenna gain are known in free space, the distance can be determined accurately from the RSS using the Friis equation. However, in non-free space, scatterers near the transmission path, including the ground, can change the relationship between received power and distance. The exponent of the distance d is the propagation law parameter. The environmental conditions can be accounted for by selecting the most suitable propagation law parameter for the region where the system is used [9].

2.2.3 Angle of Arrival

The AoA method is probably the easiest to implement, and the only hardware required for this method is a directional antenna. AoA is often used to locate illegal transmitters and to track wildlife. AoA is not subject to the conditions of

using other positioning methods. It does not require the cooperation of a target and can use any signal. The AoA method is an essential part of a radar system where the radar only needs one fixed device to determine the position, which requires using both AoA and ToF methods. When using AoA alone, at least two fixed devices are required to determine position, or one mobile terminal is used to make separate measurements at two locations.

The direction of the electromagnetic wave can be estimated by varying the known spatial radiation pattern of the transmitting or receiving antenna and the variation of the RSS because The wavefront of the transmitted signal is perpendicular to the direction of wave propagation [10]. The signal's AoA can be determined based on the maximum signal strength or the zero point of the signal strength when the antenna is rotated, depending on the position of the fixed BS [9]. In contrast to RSS, the AoA method does not require a specific transmit power value.

Figure2.2 shows the angle-based geometric mapping. A and B are fixed BSs at known locations, while T is a mobile UE at an unknown location. A and B are equipped with steerable unidirectional antennas, and T sends signals to A and B. A and B obtain the AoA from the received signals. Since the positions of A and B are known, the distance AB is known. When the AoA is determined, the intersection of the extensions of the sides of angles A and B is the position of T.

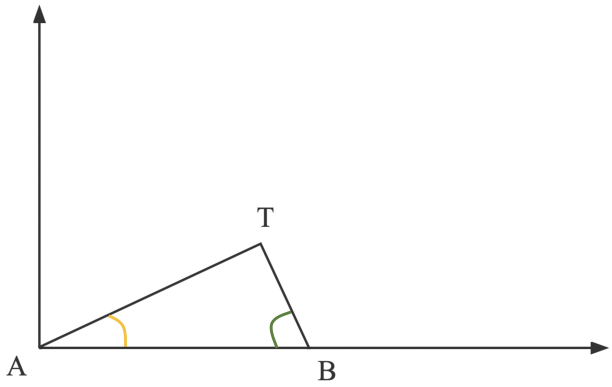

Figure 2.2: Angle-based geometric mapping.

2.2.4 Fingerprinting

NLoS and multipath transmission are the biggest obstacles for conventional positioning techniques to achieve high positioning accuracy. Positioning can be achieved using multipath transmission and is robust under NLoS transmission conditions using database comparison-based positioning techniques, which typically obtain the location of a mobile UE by comparing the CSI measured online in real-time with the CSI with location labels in a database built offline. The database comparison-based positioning technique can also be called fingerprinting or pattern recognition [9]. The fingerprinting method has two phases; the first phase, also called the offline measurement phase, aims to create a CSI fingerprint

database with location labels. The CSI fingerprint is unique for each UE location as the CSI changes over time due to changes in the location of the UE and the position of the scatterers in the propagation path. When an ML algorithm is used to compare fingerprints, the data collected in this phase is used to train the ML algorithm to obtain an ML model that can accurately predict the location in subsequent phases. The second phase of the fingerprint recognition method is the real-time online location phase. This phase measures the new CSI fingerprint in real-time and compares it with the fingerprints in the database for position estimation. The fingerprinting method only applies to the environment where the database is created. The physical changes affecting the radio propagation in that environment result in the need to create a new database [9]. The database is usually time-consuming and expensive to build. Several methods have been developed to compare real-time measurement data with offline databases, including the minimum Euclidean distance method, statistical methods, and the ML algorithms [9]. Fingerprinting has many advantages, such as it can be used for indoor and outdoor scenarios and requires only a fixed BS to achieve positioning. It requires minor hardware modification to existing communication devices. The propagation multipath and shadowing modifications are contained in the location-specific offline database information [9]. As fingerprinting localization has many advantages and ML techniques can improve the performance of fingerprinting localization, the main object of this project is the ML-based fingerprinting technique.

Chapter 3
Machine Learning

This project applies three ML algorithms for fingerprinting localization: SVR, FCNN, and CNN. Therefore the background and general knowledge of ML and the principles of the different ML algorithms will be introduced.

3.1 Introduction

ML is a multidisciplinary science involving probability theory, statistics, approximation theory, convex analysis, and computational complexity theory. ML is a way of implementing Artificial Intelligence (AI), i.e., using ML algorithms to solve AI problems. A significant advantage of ML is that it can help build concise programs that automatically adapt to new data and scenarios and are easy to maintain and perform well. Another advantage is that it can be used to solve problems that cannot be solved by traditional methods or for problems that require streamlined processes. ML can also help discover correlations and trends between data easily overlooked by humans, thus helping humans better understand the problems. The biggest challenge in ML is making well-trained models perform well on entirely new data, an ability known as generalization. ML is determined as a program that is said to learn from experience E if its performance measured by P on a task of class T improves as experience E increases [11]. ML learns to improve performance through data samples consisting of data features. ML is commonly used to solve two types of problems: classification problems and regression problems [12]. The goal of solving classification problems is to classify data samples into different categories according to specific rules, while the goal of solving regression problems is to estimate a real-valued function. Some ML algorithms originally developed for classification problems can also be used for regression tasks and vice versa. Performance metrics are used to evaluate the performance of ML algorithms, and different performance metrics are usually chosen depending on the particular task. Generally, the percentage of the correct classified results is used to evaluate since the model outputs only two types of results for classification tasks, correct and incorrect [13]. However, for regression problems, the RMSE is generally used to evaluate the model's performance.

ML can be classified as unsupervised learning, supervised learning, and semi-supervised learning based on the type of data set processed by the ML algorithm [14]. The dataset processed by supervised learning contains labels for the cor-

rect answers. Both classification and regression problems are typical supervised learning problems. ML algorithms can use these labels as feedback to enable the algorithm to achieve better performance. These labels are equivalent to a supervisor who can monitor changes in the performance of the algorithm. On the other hand, unsupervised learning is an algorithm that deals with data sets that do not contain the correct answer labels. Typical unsupervised learning problems are the cluster analysis problem and the anomaly detection problem. Unsupervised learning algorithms aim to learn a dataset's functional and structural properties. Semi-supervised learning algorithms are used to process datasets with only partial labels, which is the most common dataset in real-world applications. Most semi-supervised learning algorithms combine unsupervised and supervised learning algorithms. Some ML algorithms, such as reinforcement learning, are not trained on a fixed dataset. Reinforcement learning algorithms interact with the environment, so their systems can observe the environment and select actions using a policy, then execute the actions and get feedback, update the policy with the feedback, and iterate the process until the best policy is found.

ML algorithms can be classified according to the depth of the algorithm as shallow learning, also known as traditional ML algorithms, and deep learning algorithms. The structure of an ML algorithm usually consists of three parts, an input layer, hidden layers, and an output layer. A shallow learning algorithm is an ML algorithm with only a few or no hidden layers. Typical shallow learning algorithms are SVM, Artificial Neural Network (ANN), and Logistic Regression (LR). ANNs were first proposed in the 1940s and were initially inspired by neuroscience to build a simplified model of the basic computational units of the human cerebral cortex. However, modern ANNs are function approximators designed to achieve statistical generalization and are not intended to achieve perfect modeling of the human brain. ANNs have been an active area of research since their introduction and have produced many variants. Deep learning algorithms are ML algorithms that have a lot of hidden layers. One of the classical deep learning algorithms is DNN, which was proposed in 2006 [15]. DNN stands out from traditional ML algorithms for its excellent performance; deep learning frameworks have also been developed as a subfield of ML. DNNs approach high-complexity functions by adding hidden layers and adding neurons within the hidden layers.

In general, all current ML systems use the tensor as the data structure [16]. The core of the tensor concept is that it is a data container, which almost always contains numerical data. A tensor is a generalization of a matrix to any dimension [17]. A scalar is a zero-dimensional tensor that contains only one number. A vector is a one-dimensional tensor, and an array of numbers is called a vector. A matrix is a two-dimensional tensor, and an array of vectors is called a matrix. A three-dimensional tensor is a new array of multiple matrices, which can be intuitively understood as a cube of numbers.

3.2 Support Vector-Based Algorithms

3.2.1 SVM

SVM are the most influential traditional ML algorithms of the 21st century. SVM is most commonly used in two-class classification and multi-classification problems [18]. The SVR algorithm is a variant of SVM that can be used to solve regression problems. SVM is deeply rooted in statistics, optimization, and ML principles. SVM has the robustness, and good generalization SVM is a sparse kernel decision machine, and its learning model does not require the computation of posterior probabilities [13]. Sparsity means that all training data are used to determine the SVM model parameters during the training process. However, after the parameters are determined, the SVM relies on only a subset of the training data, i.e., all support vectors, to predict the labels of new data samples. The complexity of the SVM algorithm relies on the number of support vectors rather than the dimensionality of the input data space [19]. All support vectors are used to define the hyperplane that divides the categories, and finding support vectors is done by an optimization step using Lagrange multipliers [20]. The upper limit on the number of support vectors is 50 percent of the size of the training data set [13]. The SVM model reduces the computation time of the test set, which is beneficial in application scenarios with storage requirements. In addition to minimizing the error function based on the training set, SVM has another constraint: maximizing the margins.

It is more intuitive to explain the principle of SVM visually. The geometric representation of the linear SVM is shown in Figure 3.1. The goal of training SVM is to find a hyperplane in the input feature space that optimally divides the different classes of data samples. So SVM is a discriminative ML technique based on finding a discriminant function [12] that correctly predicts the labels of new instances. The advantage of the discriminative approach is that it requires fewer computational resources and training data. SVM aims to solve convex optimization problems analytically and always returns the same optimal hyperplane parameters. SVM learns many hyperplanes during the training phase, but only the optimal ones are saved. Therefore, the generalization of hyperplanes obtained when trained with representative data samples is better. The hyperplane of SVM can be expressed by the formula (3.1) [18].

$$\mathrm{y}(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}) + b \tag{3.1}$$

The function of the hyperplane can divide the input space into two parts, where $\boldsymbol{x}$ is the data sample vector. The $\boldsymbol{\phi}(\boldsymbol{x})$ represents the spatial transformation of the input data features, also known as the kernel mapping function, which can extend the linear SVM model to a nonlinear SVM model, and the kernel trick will be described in detail later. b is the bias parameter, and $\boldsymbol{w}$ is the weight vector. In the two-class classification problem, a data sample is represented using $\boldsymbol{x}_n$, which corresponds to a ground truth value t_n. There are only two possible values for t_n, 1 and -1. The predicted value corresponding to $\boldsymbol{x}_n$ is denoted by $y(\boldsymbol{x}_n)$. In the two-class classification problem, when $y(\boldsymbol{x}_n)$ is positive, then $\boldsymbol{x}_n$ is classified as class 1, and When $y(\boldsymbol{x}_n)$ is negative, then $\boldsymbol{x}_n$ is classified as class 2. Let the

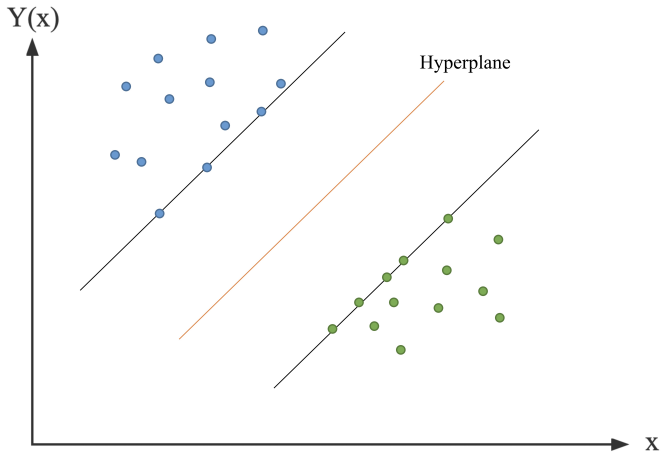

Figure 3.1: Geometric representation of linear SVM

support vector satisfy the equation (3.2) [12]. In this case, all data samples will satisfy the restriction in equation (3.3) [12].

$$t_n \left(\boldsymbol{w}^T \boldsymbol{\phi} \left(\boldsymbol{x}_n \right) + b \right) = 1 \tag{3.2}$$

$$t_n \left(\boldsymbol{w}^T \boldsymbol{\phi} \left(\boldsymbol{x}_n \right) + b \right) \geq 1, \quad n = 1, \ldots, N \tag{3.3}$$

The process of determining the best hyperplane for a two-class classification SVM is to solve for the values of the parameters $\boldsymbol{w}$ and b to maximize the margins under the constraints formula (3.3). The vertical distance between the hyperplane and the nearest data sample, i.e., the support vectors, is defined as the margin, and the constraint of maximizing the margin allows the training process to eventually find the hyperplane with the best generalization capability, i.e., a model that has satisfactory classification accuracy even on new data. The support vectors are represented by the blue dots and red circles in figure 3.2.

The distance between the data sample $\boldsymbol{x}_n$ and hyperplane is expressed by $\frac{t_n y(\boldsymbol{x}_n)}{\|\boldsymbol{w}\|}$ [18]. So to obtain the optimal parameters $\boldsymbol{w}$ and b, it is necessary to maximize this distance, and the maximum margin solution is found by the following formula (3.4) [18].

$$\arg\max_{\boldsymbol{w},b} \left\{ \frac{1}{\|\boldsymbol{w}\|} \min_n \left[t_n \left(\boldsymbol{w}^T \boldsymbol{\phi} \left(\boldsymbol{x}_n \right) + b \right) \right] \right\} \tag{3.4}$$

Soft-edge two-class classification SVMs are proposed to improve the model's generalization ability, which means that some training data samples are allowed to be misclassified. Soft-edge SVMs do not provide error-free classification; in this case, soft-edge SVMs classify most of the data correctly, allowing the model's errors to exist near the classification boundary. The regularization ability is measured by

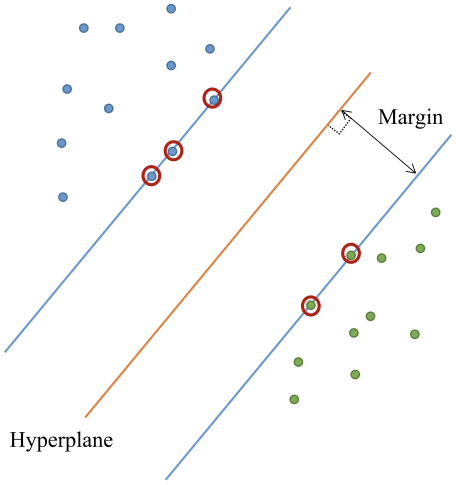

Figure 3.2: Support vectors

the parameter C, which varies according to the optimization objective. C increases and the margin becomes shorter [12]. C decreases, allowing more samples of the offending data to exist. However, the goal of the SVM is still to maximize the margin between different classes. The geometric representation of the soft-margin SVM classifier is shown in Figure 3.3.

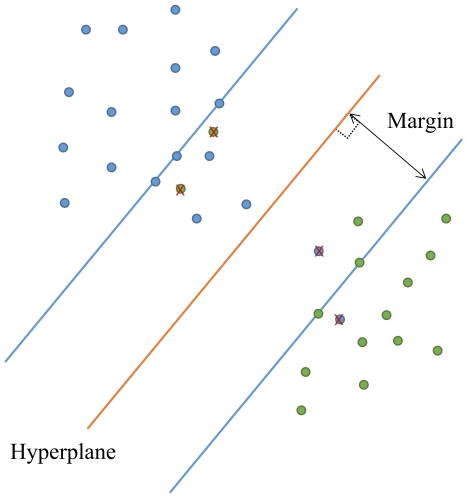

Figure 3.3: Geometric representation of soft-margin SVM

3.2.2 SVR

SVMs for solving classification problems can be generalized to SVR algorithms for solving regression problems. SVR is an effective tool for estimating real-valued functions so that SVR models return continuous-valued outputs rather than finite outputs. Like SVMs, SVR is characterized by kernel tricks and sparse solutions. SVR has many advantages, such as the computational complexity of SVR is not determined by the dimensionality of the input feature space but by the support vector, and SVR has good generalization capabilities. SVR introduces a ϵ insensitive region [12] to the SVM model. This region is called the ϵ tube. It is symmetrically distributed around the function to be estimated, so SVR uses a symmetric loss function to penalize overestimation and underestimation equally. Only the points outside the tube receive a penalty. The points inside the tube are not penalized in any case. The geometric interpretation of the nonlinear SVR model is shown in Figure 3.4, where the two blue dashed lines represent the ϵ tube.

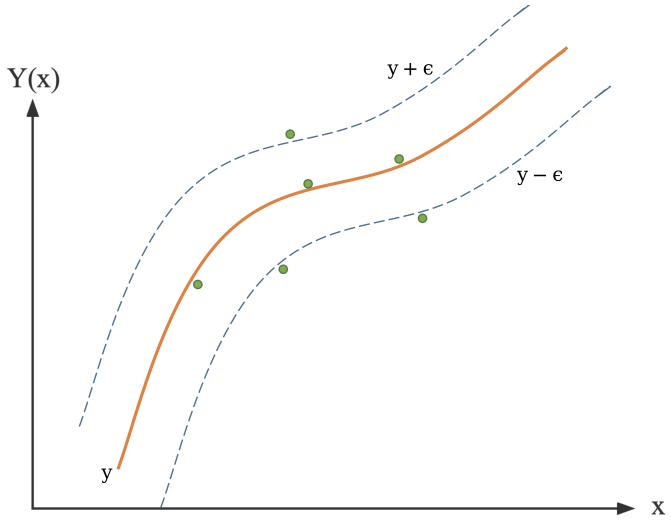

Figure 3.4: Geometric representation of SVR

The hyperplane of SVR is also defined in terms of support vectors, which determine the shape of the ϵ tube. The support vector in SVR is the data sample closest to the ϵ tube boundary and located outside the tube. SVR uses an ϵ insensitive loss function to penalize predictions that are farther from the desired output than ϵ. The value of ϵ determines the width of the tube, and a smaller value of ϵ indicates that the model is less tolerant of error. The value of ϵ also affects the number of support vectors and, thus, the sparsity of the solution. If ϵ decreases, the boundary of the tube moves inward, so more data samples are located outside the tube, which indicates an increase in the number of support vectors. Similarly, an increase in ϵ leads to a decrease in the number of data samples near the outside boundary, which is the number of support vectors. The introduction of ϵ insensitive regions makes the model more robust. The process of training SVR is determining the values of the parameters $\boldsymbol{w}$ and b so that the ϵ

tube can contain as many data samples as possible. Thus the optimization problem in SVR is to minimize a regularized error function shown by the equation (3.5) [12].

$$C \sum_{n=1}^{N} E_\epsilon \left(y(\boldsymbol{x}) - t_n \right) + \frac{1}{2} \|\boldsymbol{w}\|^2 \tag{3.5}$$

The ϵ insensitive error function [18] is used to find the sparse solution, and it can be expressed by the formula (3.6) [12]. Suppose the absolute value of the difference between the prediction result and the target is less than ϵ. In that case, this error function considers the error as 0, where ϵ is a positive number and C is the regularization parameter.

$$E_\epsilon(y(\boldsymbol{x}) - t) = \begin{cases} 0, & \text{if } |y(\boldsymbol{x}) - t| < \epsilon \\ |y(\boldsymbol{x}) - t| - \epsilon, & \text{otherwise} \end{cases} \tag{3.6}$$

3.2.3 Kernel Mapping

The concept of kernel mapping comes from [21]. In most problems waiting to be solved using ML algorithms, the data samples are often linearly inseparable from the original input space. Therefore linear hyperplanes are generally unable to classify nonlinear data samples correctly. Finding a nonlinearly separable hyperplane in the original input space can significantly increase the computational requirements. Linear SVM and SVR models can solve nonlinear classification problems and estimate nonlinear functions using kernel mapping functions. The kernel mapping function uses a defined kernel function to map the original input data samples to a new multi-dimensional data space where the data is linearly separable. Finding a linearly separable hyperplane in this new space is easy. The kernel trick can be used to extend many linear ML models. The kernel trick also has drawbacks in that its computation time increases exponentially with the number of samples. Even so, the computational effort of kernel mapping is trivial compared to finding a nonlinear hyperplane in the original input space. The kernel should be a Hermitian and positive semi-infinite matrix and needs to satisfy the Mercer theorem, which implies that the kernel or Gram matrix is evaluated as positive semi-infinite over all data point pairs, forming the formula (3.7) [12], where $\varphi(x)$ belongs to the Hilbert space [12].

$$K(x, u) = \sum \varphi_r(x) \varphi_r(u), \tag{3.7}$$

Using inner products of feature spaces to represent kernels extends many nonlinear ML models. When the input data sample vector $\boldsymbol{x}$ of some ML algorithm is a scalar product, it is possible to replace this scalar product with some other kernel. The popular kernel functions are the linear kernel, polynomial function, hyperbolic tangent function, Gaussian Radial Basis Function (RBF), variance RBF kernel, Etc. The choice of specific kernel functions depends heavily on the specifics of the data input space. For example, the linear kernel is the simplest and most useful in sizable sparse data vectors. Polynomial kernels are widely used for image processing, while variance RB kernels are often used for regression tasks. The

Gaussian RBF is the general kernel, and its application is mainly in the absence of prior knowledge [12]. If the kernel matrix is diagonal, then the feature space is redundant, and another kernel should be tried after feature reduction [12]. It is worth noting that the computation of the kernel matrix requires much memory and computational resources when the kernel is used to transform the feature vector from the input space to a linearly inseparable dataset in the kernel space [12]. When the size of the dataset increases, it becomes very inefficient or even infeasible to compute the kernel matrix, which makes SVM or SVR with kernels impractical for big data. The kernel function used in this project is the Gaussian RBF, which can be expressed by the formula (3.8) [12].

$$K(x, u) = \exp\left(-\frac{\|x - u\|^2}{\sigma^2}\right) \tag{3.8}$$

3.3 Deep Learning Algorithm

3.3.1 Introduction

A typical example of deep learning is DNN. One of the main reasons for promoting deep learning is the poor generalization of traditional ML algorithms to specific problems, such as speech recognition or object recognition. Traditional ML could perform better in generalization [14], and the computational cost is often huge when dealing with high-dimensional input data samples. In contrast, deep learning can overcome this challenge. The DNNs used in this project are a typical example of a deep learning framework. However, DNNs have limitations, such as the curse of dimensionality. Another challenge is the nonlinear dataset, which complicates the separation task. Therefore, in the 1990s and early 2000s, DNNs were much less advantageous than shallow learning models such as SVMs. As computer computing power increased, DNNs were brought back into the public and are practiced and flourishing in many fields. DNNs aim to learn multiple levels of abstraction of knowledge representation hierarchically to achieve powerful AI models. Information in DNNs is propagated through higher levels to accumulate knowledge, so higher-level learning is defined by and builds on lower-level statistical learning. We can understand the combination of backpropagation algorithms with DNNs as a forerunner of deep learning architectures.

3.3.2 DNN

The basic structure of a DNN is the neural layer, which consists of a series of artificial neurons. As shown in Figure 3.5, the shape and function of artificial neurons are similar to biological neurons [12]. An artificial neuron can be summarized by a mathematical formula (3.9).

$$y = \theta(\boldsymbol{W} \cdot \boldsymbol{x} + b) \tag{3.9}$$

The input data $\boldsymbol{x}$ is connected to the neuron through a weighted connection, and y is the neuron's output. $\boldsymbol{W}$ is the weight, b is the bias, $\theta(.)$ is the activation function which is a nonlinear function that determines how a neuron outputs data.

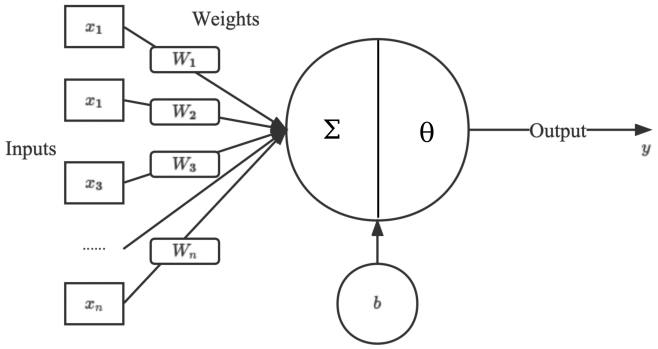

Figure 3.5: Artificial neuron.

There are many different choices of activation functions, the most common of which are hard limiter, saturated linear function, log sigma function, and hyperbolic tangent sigma function [16].

An artificial neural layer can be thought of as a data filter. Each neural layer extracts representations contributing to the problem from the input data. It means that some data is fed into the layer, Then the output from that layer, and the output is more valuable than the input data. A deep learning model is like a data processing sieve containing increasingly finer data filters. A neural network layer consists of a bunch of neurons, each with a weight, bias, activation function, and output. A neural network layer is shown in Figure 3.6. The output of this layer is an output vector consisting of the individual outputs of neurons. A neuronal layer can be conveniently expressed in matrix notation as the formula (3.10) [12].

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_i \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} \theta\left(\sum_{j=1}^{M} \boldsymbol{W}_{11}\boldsymbol{x}_1 + b_1\right) \\ \vdots \\ \theta\left(\sum_{j=1}^{M} \boldsymbol{W}_{ij}\boldsymbol{x}_j + b_i\right) \\ \vdots \\ \theta\left(\sum_{j=1}^{M} \boldsymbol{W}_{NM}\boldsymbol{x}_M + b_N\right) \end{bmatrix} = \theta(\boldsymbol{W} \cdot \boldsymbol{x} + \boldsymbol{b}) \tag{3.10}$$

The input of the layer is $\boldsymbol{x}$, and the output of the layer is $\boldsymbol{y}$. $\boldsymbol{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_N \end{bmatrix}$, and $\boldsymbol{W}_{ij}$ denotes the connection weight of the j_{th} input to the i_{th} neuron, and y_i is the output of the i_{th} neuron.

Figure 3.7 shows an ANN consisting of three hidden neural layers. The current layer's output is the next layer's input; each layer of the neuronal network shown in the figure uses the same activation function. However, different layers in the network can use different activation functions in other applications. The number

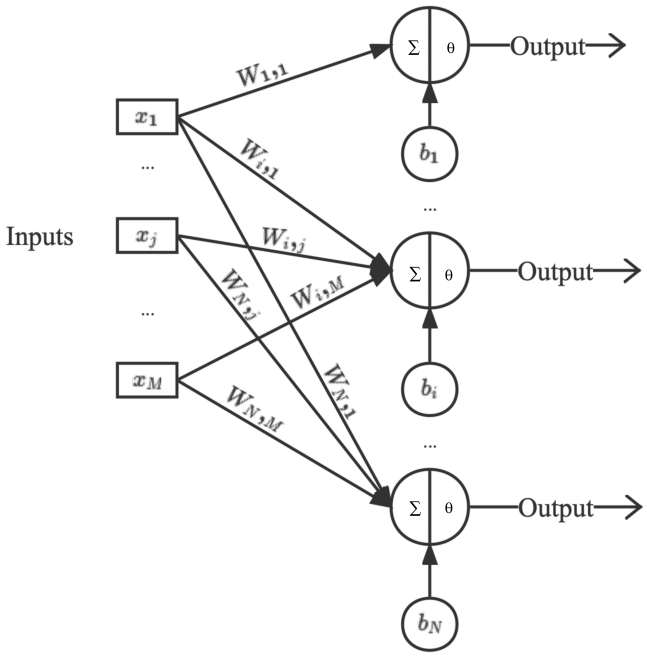

Figure 3.6: An artificial neural layer.

of hidden layers and the number of neurons in each neural layer of an ANN with the best performance is not quantifiable and depends entirely on the practical application [14]. Each neuron in a hidden layer divides the input data space into multiple subspaces, and the boundaries of these subspaces are defined by the hyperplane associated with each neuron. The fewer neurons in the hidden layer, the fewer subspaces are created, and the more the network tends to cluster and map data samples to the same output [12]. The output of each neuron is a nonlinear transformation of the hyperplane. When the number of neurons is too large, there is a greater risk of overfitting, which degrades the generalization performance, i.e., the model performs well on the training set but poorly on new data. The training set must be large enough to ensure that the subspace obtained in each hidden neuron layer can correctly separate the data samples. The topology of a DNN consisting of neural layers is a directed acyclic graph [16]. The most common network topology is a simple linear stack of neural layers. Such a structure maps a single input to a single output. Other common network topologies are two-branch networks, and multi-head networks [16]. The topology of a network defines a hypothesis space. Choosing a network topology means restricting the hypothesis space to specific operations on data samples and mapping input data to output data. The training goal is to find a suitable set of values for the weights of this operation [16].

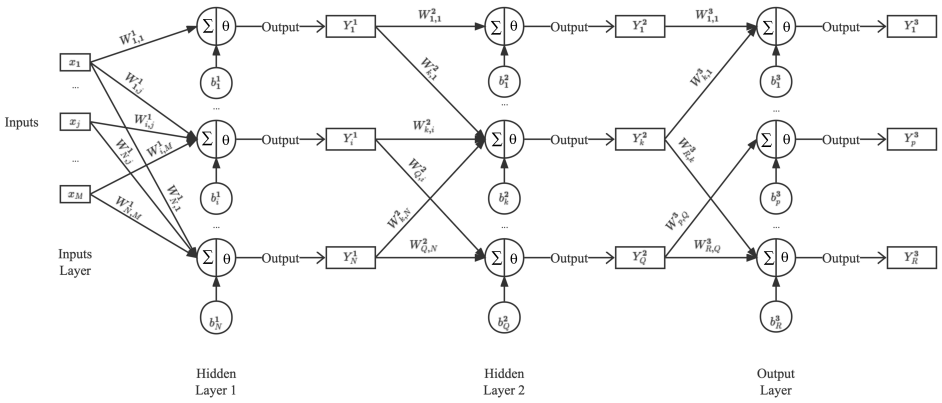

Figure 3.7: An artificial neural network with three hidden layers.

3.3.3 FCNN

A FCNN is a DNN that consists of a series of fully connected neural layers. A fully connected layer is also called a dense layer, which means that all neurons of one neural layer are connected to all neurons of the following neural layer. Dense layers learn global patterns [16] from the input feature space. FCNN is often applied to specific types of data, but this kind of network has some disadvantages in image recognition and classification. When the input of FCNN is an image, after FCNN learns a pattern in the image, when the pattern appears again in another corner of the image, FCNN can only relearn it but not recognize it directly. FCNN is computationally intensive, which tends to lead to model overfitting. The main advantage of FCNNs is "structure agnostic," i.e., which does not require specific assumptions about the input data. This structure-agnostic property makes the applicability of FCNN very wide. Figure 3.8 shows the structure of an FCNN.

3.3.4 CNN

A CNN [22] is a DNN dedicated to processing data with a grid structure, such as time series and images. Time series can be considered a one-dimensional grid formed by regular sampling on a time axis, and images can be considered a two-dimensional grid of pixels. CNNs, also known as convnet, consist of at least one convolutional layer, one Pooling layer, and one fully connected layer, and they perform well in many applications. Figure 3.9 shows the structure of the most straightforward CNN.

CNNs use the mathematical operation of convolution, which is a special kind of linear operation. In its usual form, convolution is a mathematical operation on two functions of real variables. The one-dimensional convolution operation can be represented by the formula (3.11) [14], where x is the input data, usually a multi-dimensional array. w is the convolutional kernel, a multi-dimensional array of parameters optimized by the learning algorithm. These multi-dimensional arrays are collectively called a tensor, and s is the output called the feature map.

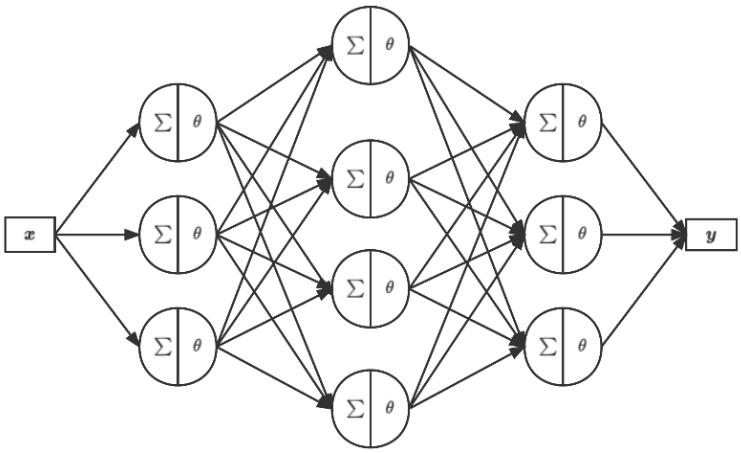

Figure 3.8: The structure of the FCNN.

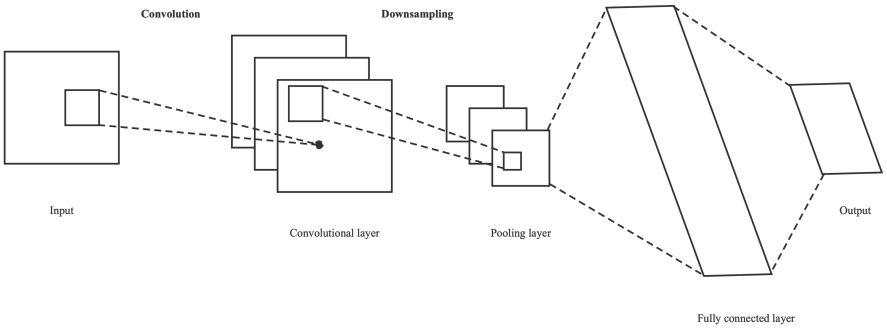

Figure 3.9: The structure of the CNN.

CNNs allow simultaneous convolutional operations in multiple dimensions. Convolutional operations in two-dimensional can be expressed by the formula (3.12) [14]. The convolution kernel is defined by two critical parameters: the size of the blocks extracted from the input and the depth of the output feature map, which is the number of filters. Convolution works by sliding a window over the input, stopping at each possible position, extracting the surrounding feature blocks, and performing a tensor product operation with the convolutional kernel.

$$s(i) = (x * w)(i) = \sum_{a=-\infty}^{\infty} x(a)w(i-a) \tag{3.11}$$

$$S(i,j) = (X * W)(i,j) = \sum_{m}\sum_{n} X(m,n)W(i-m, j-n) \tag{3.12}$$

The convolutional layer learns the local patterns of the input, an important property that gives CNN two exciting properties. The patterns learned by the CNN are translation invariant, which means that when the input is an image after the CNN has learned a pattern in the lower right corner of the image, it can recognize that pattern again anywhere in the image. It allows the CNN to use the data efficiently when processing images, requiring fewer training samples to learn a data representation with generalization capabilities. CNNs can learn patterns in spatial hierarchies. The first convolutional layer will learn smaller local patterns. The second convolutional layer will learn larger patterns consisting of features from the first layer. It allows CNNs to learn increasingly complex and abstract visual concepts efficiently [16].

Pooling is an operation that almost all CNNs use. The purpose of the pooling operation is to downsample the output feature map of the convolutional layer to reduce the number of elements of the feature map. The pooling function uses the overall statistical characteristics of the neighboring outputs at a given location to replace the network's output at that location. For example, the output of the maximum pooling function [23] is the maximum value within the adjacent rectangular region. Another pooling functions are average pooling. Regardless of which pooling function is used, when there is a small amount of translation of the input, pooling can help the representation of the input to be approximately constant because the pooling combines the feedback from all the neighbors.

3.3.5 Training Process of DNNs

Figure 3.10 illustrates the process of training a DNN, where the network is propagated forward and backward to achieve the training purpose. The loss function measures the network's performance on the training data. The optimizer is used to update the weights of the network. The weights of the network are also called trainable parameters. The weights contain the information the network learns from the training data. Each cycle is called a training epoch, and the network weights undergo an update at the end of each training epoch. The purpose of training is to keep repeating the training epoch until a model with satisfactory performance is trained.

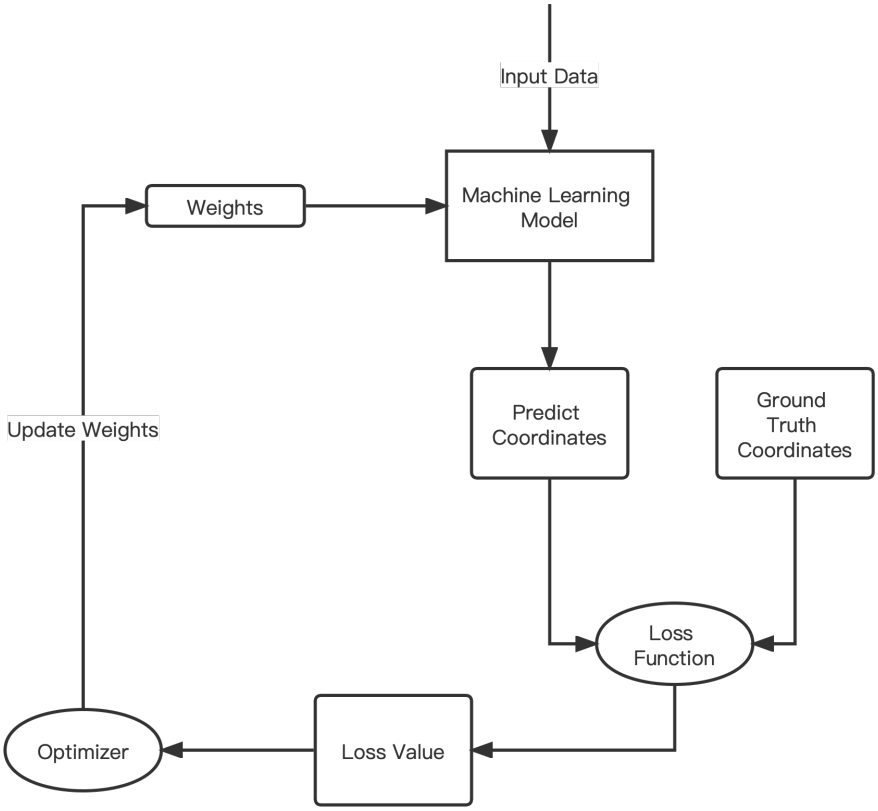

Figure 3.10: The process of training an ML model.

The input data is fed into the DNN, and the predicted value is obtained. The predicted value and the ground truth value are used as the input of the loss function to find the loss value, called forward propagation. The network weights are randomly initialized at the first training epoch, and the network's output with such weights is meaningless. However, the optimizer can gradually adjust these weights according to the feedback signal, i.e., the loss value. The loss function used in this project is RMSE, whose expression is the formula (3.13), where N is the number of data samples, y_i is the ground truth value, $\hat{y}_i$ is the output of the model.

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N} (y_i - \hat{y}_i)^2}{N}} \tag{3.13}$$

The input data is first forward propagated, and the predicted value is obtained. The predicted value and the ground truth value are input to the loss function to find the loss value, which is then fed to the optimizer. Since all the computations in the DNN are differentiable, the optimizer uses a stochastic gradient descent algorithm [22] to update the values of the weights to reduce the loss values; this process is called backpropagation.

Chapter 4

Measurement of The Datasets

4.1 The Measurement Environment of The Datasets

The four ultra-dense indoor Massive MIMO CSI datasets [24] used in this project were measured on the ESAT-TELEMIC Massive MIMO testbed at KU Leuven [1]. This testbed is a standard cellular bandwidth-based Massive MIMO OFDM communication system consisting of a BS and four UEs. The BS is equipped with a Massive MIMO array consisting of 64 patch antennas, all four UEs are equipped with Universal Software Radio Peripheral (USRP)s, and each UE uses a dipole antenna to transmit the pilot signal. The measured system has a center frequency of 2.61 GHz, a bandwidth of 20 MHz, and a transmit power of 15 dBm. This system uses a Time Division Duplex (TDD) based framework architecture with OFDM modulation, and demodulation via a Xilinx Field Programmable Gate Array (FPGA) [1].

The measurement site of the CSI datasets collection activity is shown in Figure 4.1 [1], which takes place in the MIMO lab at KU Leuven. All 64 antennas of the BS simultaneously receive the orthogonal pilot signal sent by the UE and perform channel estimation. The spatial position of the UE when it sends the pilot signal is recorded as the ground truth label for the CSI data.

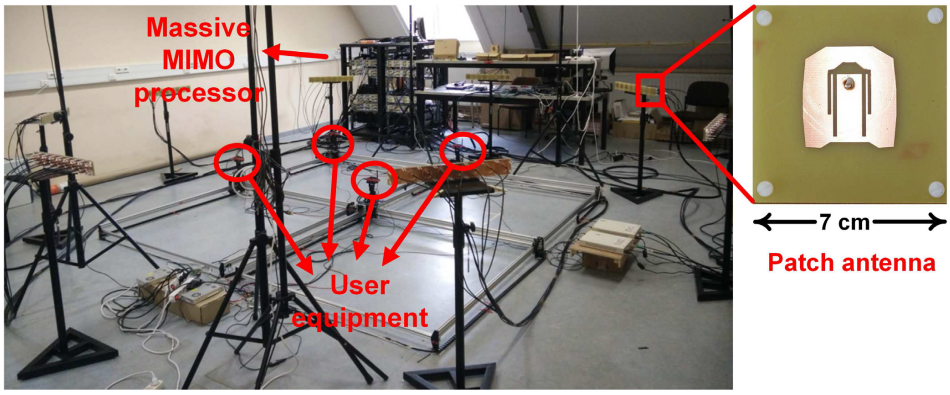

Figure 4.1: MIMO Lab at KU Leuven [1].

The measurement activity is carried out when the BS measures the CFR,

which can be considered CSI, using the LabVIEW communication MIMO application framework [1]. The pilot signal used for channel estimation consists of 100 subcarriers, which are uniformly spaced in frequency. Therefore, the CFR obtained from a single transmission measurement can be represented by the complex matrix H_{CSI}, which can be expressed by the formula(4.1) [1], where $n_r \in \{1, 2, \cdots, 64\}$, $n_k \in \{1, 2, \cdots, 100\}$ represent the array antenna and OFDM subcarrier indices, respectively. The structure of the CFR for a single measurement can be represented by Figure 4.2. The pilot signal processing chain used for channel estimation is shown in the dashed part of Figure 4.3 [1].

$$\mathbf{H}_{\text{CSI}} = \{H_{n_r,n_k}\} \in C^{64\times 100} \tag{4.1}$$

	100 OFDM Subcarriers				
0	(0.0-0.15625j)	(0.0-0.17578125j)	.....	(-0.015625-0.203125j)	(-0.0625-0.18359375j)
64	(0.12890625-0.015625j)	(0.1484375-0.02734375j)	.....	(0.03125+0.09765625j)	(0.01171875+0.12109375j)
Antenna	.....	.....	.....	.....	.....
Elements	.....	.....	.....	.....	.....
	(0.015625+0.13671875j)	(0.04296875+0.14453125j)	.....	(-0.03125+0.34375j)	(-0.078125+0.3203125j)
63	(0.10546875+0.05078125j)	(0.078125+0.04296875j)	.....	(0.15625+0.41796875j)	(0.2265625+0.4140625j)

99

Figure 4.2: The structure of the CFR data.

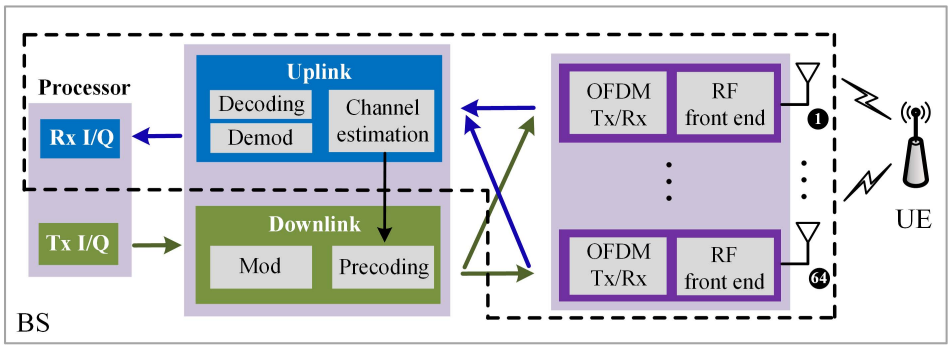

Figure 4.3: The signal processing chain for channel estimation [1].

The Massive MIMO testbed is designed for the flexible deployment of antenna arrays. It provides three 64-antenna array topologies for CSI collection, namely ULA with the shape of 1 × 64, URA with the shape of 8 × 8, and 8 Distributed ULAs, each ULA with the shape of 1 × 8, as detailed in Figure 4.4, Figure 4.5, and Figure 4.6.

The number next to the antenna elements in the figures is the order of CSI in the antenna domain, which is n_r in formula (4.1), and the spacing between adjacent antenna array elements is 7 cm. The origin of the space is defined as the middle of the URA. From this point in space, the X and Y positions of the UEs and the BS are measured. These positions are provided in the dataset in three dimensions. The height of both ULAs and Distributed ULAs is one meter, while

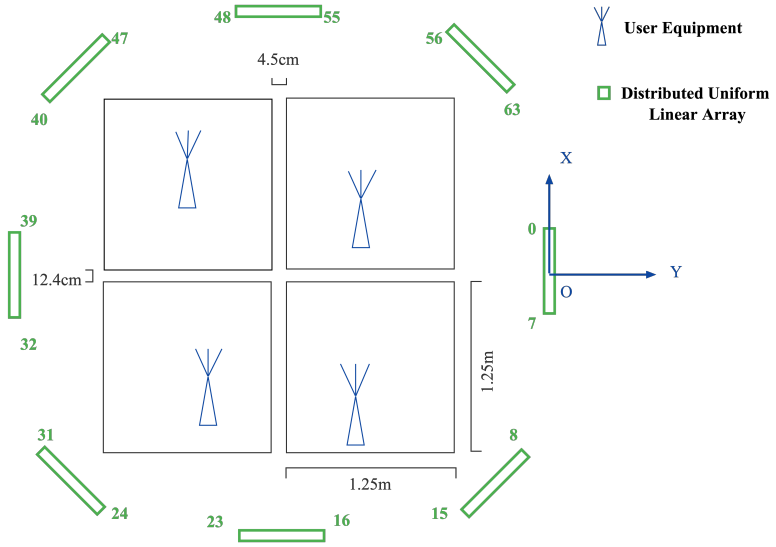

Figure 4.4: 8 Distributed uniform linear arrays.

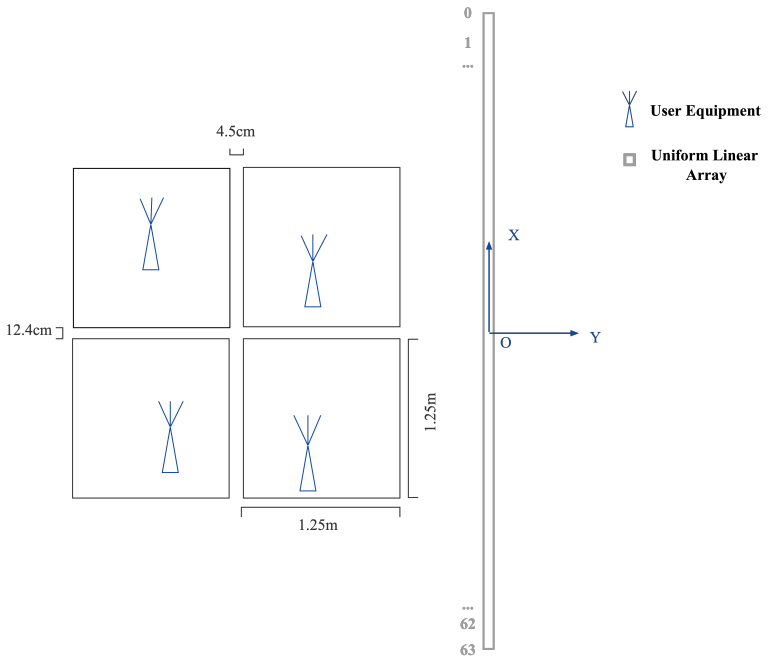

Figure 4.5: Uniform linear array.

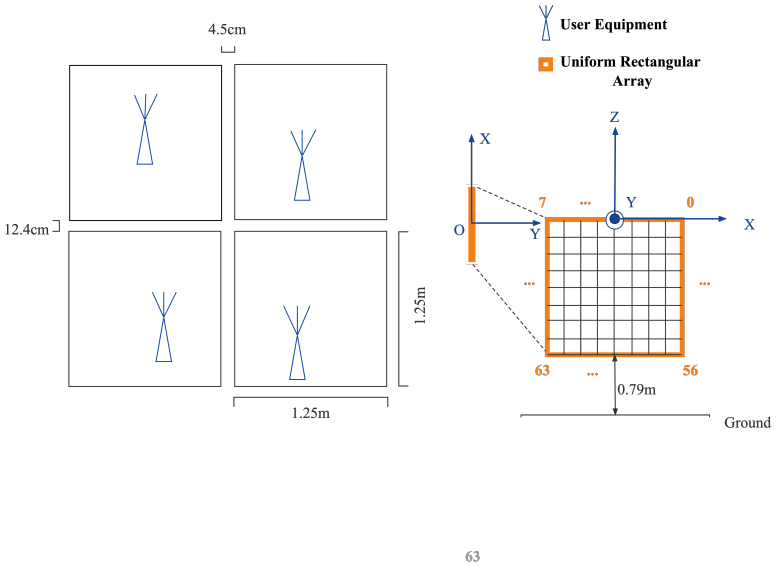

Figure 4.6: Uniform rectangular array.

the height of the lowest antenna element of the URA is 79 cm. The four UEs have a height of 40 cm. the ULA and URA are deployed on one side of the target area, while the Distributed ULAs are deployed around the four UEs. A Computer Numerical Control (CNC) X-Y table was used to control the four UEs so that they moved along a herringbone trajectory [1]. The four UEs moved in steps of 5 mm each, which resulted in an error of less than 1 mm in the ground truth coordinate labels of UEs, which resulted in highly accurate and fine-grained datasets. 252004 CSI samples and their corresponding UE ground truth coordinates labels were collected for each dataset. In all cases, the antennas are placed 1 m from the XY station, which is much smaller than the Rayleigh distance, so the communication system operates in a near-field environment. The four black-bordered rectangles in the figures depict the 1.25 m $\times$ 1.25 m area into which the XY station can move the UEs [1].

A total of four datasets were collected using this testbed, three of which were collected using ULA, Distributed ULAs, and URA as the BS under LoS transmission conditions. One dataset was collected using URA as the BS under NLoS transmission conditions, and metal blockers were placed between the four UEs and BS to create the NLoS transmission environment.

4.2 Utilization and Processing of Datasets

4.2.1 Generation of Fingerprints

This project investigates the impact of different fingerprints: CFR and CIR, on fingerprinting positioning performance. CFR is acquired directly by the measure-

ment system, so the Inverse Fast Fourier Transform (IFFT) algorithm was used to convert the CFR to CIR. An image of CFR is shown in Figure 4.7. CFR fingerprint has 100 valid features in the frequency domain. The image of CIR is shown in Figure 4.8, and it can be seen that the valid features of the CIR are less than one-third of the 100 features in the time domain, so only the 30 valid features of the CIR is retained for CIR fingerprints. Therefore the CIR dataset occupies significantly less memory than the CFR dataset.

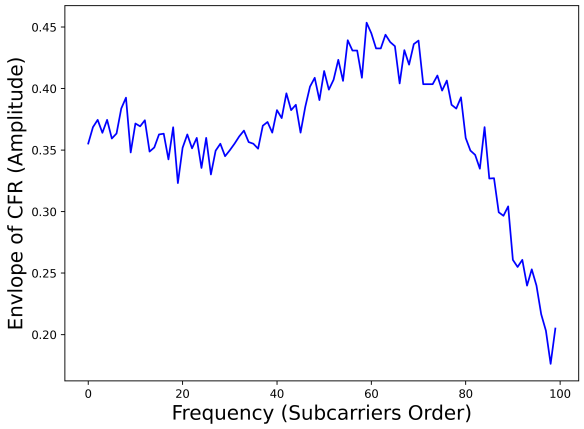

Figure 4.7: The CFR fingerprint.

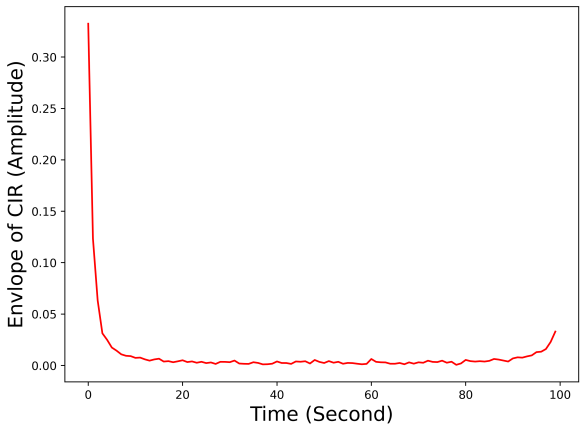

Figure 4.8: The CIR fingerprint.

After converting the directly collected CFR into CIR and the effective features extracted, CIR is represented as a 64×30 complex matrix, and CFR is a 64×100

complex matrix. Since ML algorithms can only process real-valued data, the real and imaginary parts must be separated and all represented by real values before feeding into the algorithms. The new structure of the CFR that can be fed directly into the algorithm is shown in Figure 4.9. The processing of CIR is based on the same principles and will not be repeated.

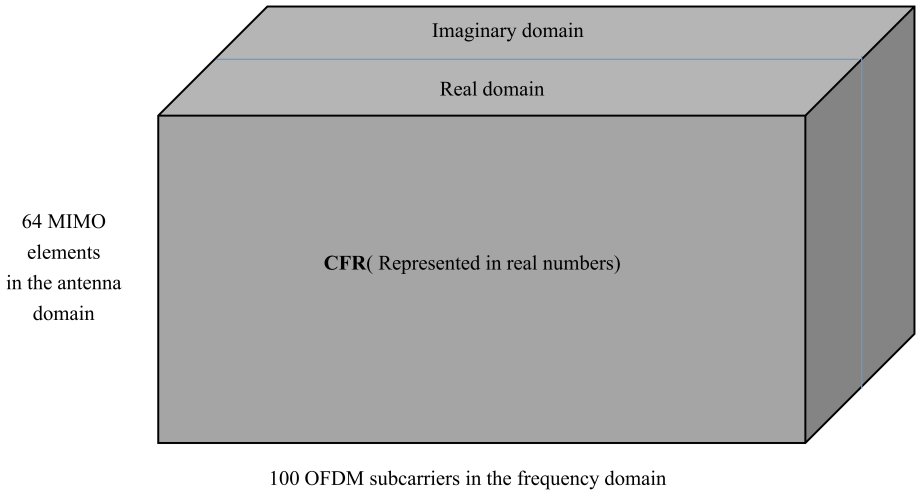

Figure 4.9: The structure of the CSI data is fed directly into the algorithm.

4.2.2 Generation of Antenna Topologies

This project investigated and compared the positioning performance of three different 64-antenna topologies and five 8-antenna topologies, respectively. The three 64-antenna topologies are ULA, Distributed ULA, and URA, introduced in the previous section. In practical situations, not all fixed BS in application scenarios can be equipped with 64 antennas, so it makes sense to investigate 8-antenna topologies. In this project, eight antennas are selected from 64 antennas of the URA, and five different 8-antenna topologies are formed by using the characteristics of the URA structure. These five 8-antenna topologies are horizontal uniform linear array, vertical uniform linear array, diagonal uniform linear array, block array, and random array. The five 8-antenna topologies are shown in Figure 4.10, Figure 4.11, Figure 4.12, Figure 4.13, and Figure 4.14. The eight antennas of the random array are randomly chosen from the 64 antennas of URA.

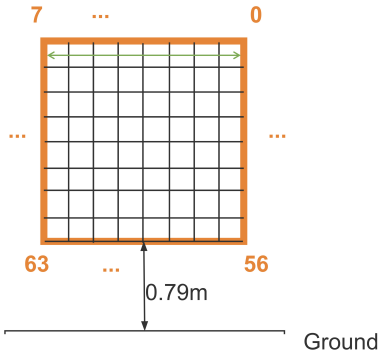

Figure 4.10: Horizontal uniform linear array.

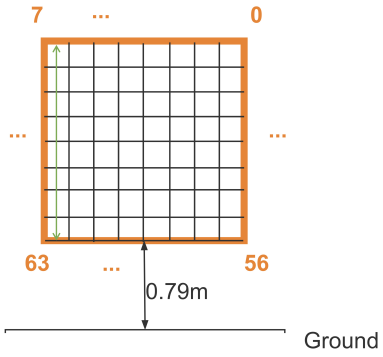

Figure 4.11: Vertical uniform linear array.

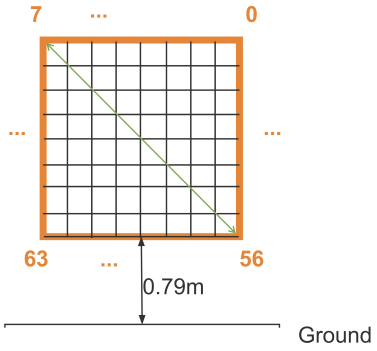

Figure 4.12: Diagonal uniform linear array.

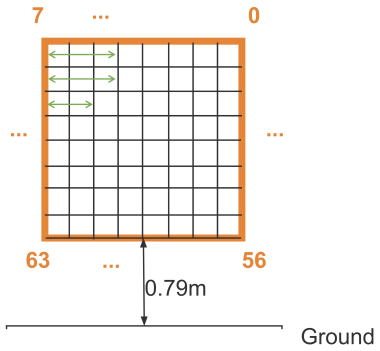

Figure 4.13: Block array.

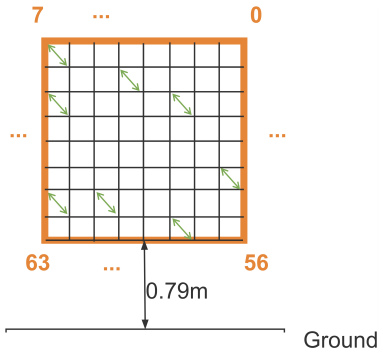

Figure 4.14: Random array.

4.2.3 Generation of ULAs with Different Numbers of Antennas

To investigate the effect of the different numbers of antennas on fingerprinting positioning performance. The antenna structures of ULA were used to form 8-antenna ULA, 16-antenna ULA, and 32-antenna ULA, respectively. All the selected antenna elements are adjacent to each other, as shown in Figure 4.15, Figure 4.16, and Figure 4.17.

Figure 4.15: 8-antenna ULA.

Figure 4.16: 16-antenna ULA.

Figure 4.17: 32-antenna ULA.

Chapter 5

Performance Comparison and Analysis Based on Real Measurements

5.1 Implementation of ML-based Fingerprinting.

To investigate the effect of ML algorithms on positioning performance, three different ML models, SVR, FCNN, and CNN, were implemented and trained using the python-based scikit-learn and PyTorch libraries. The previous chapter introduced how to use and process the dataset. To implement ML-based fingerprinting localization, a portion of the processed dataset must be selected as input to the designed ML algorithms to train the algorithm and obtain the best-performing ML models. The parameters of the algorithms used for training and the algorithm topologies are then presented.

5.1.1 SVR for Training

The hyperparameters' settings and the kernel function choice for the SVR algorithm used for training are shown in Table 5.1. The SVR algorithm used in this project is built based on the scikit-learn library and uses the NuSVR class. The kernel function chosen is the Gaussian RBF. The parameter nu takes the value interval $(0, 1]$, which is set to 0.5, representing a lower bound of 0.5 for a fraction of the number of support vectors. C is the penalty parameter of the error term, which is set to 1.

C	nu	Kernel
1	0.5	rbf

Table 5.1: Parameter setting for SVR model.

5.1.2 DNNs for Training

The two DNNs were built using the PyTorch library in this project. The FCNN used in this project consists of ten dense layers with linear kernels and Leaky Relu activation functions. The Leaky Relu function used in this project is seen

in Figure 5.2, where the slope $a = -0.3$. The three-dimensional tensor needs to be expanded into a one-dimensional tensor before being fed into the dense layer. Moreover, the network's final output is the predicted planar coordinates of UE. Figure 5.1 shows the topology of this FCNN.

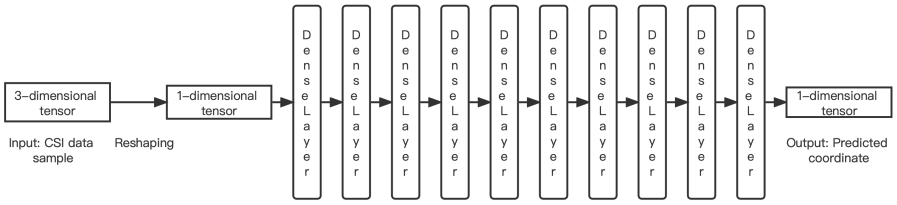

Figure 5.1: Fully connected network topology.

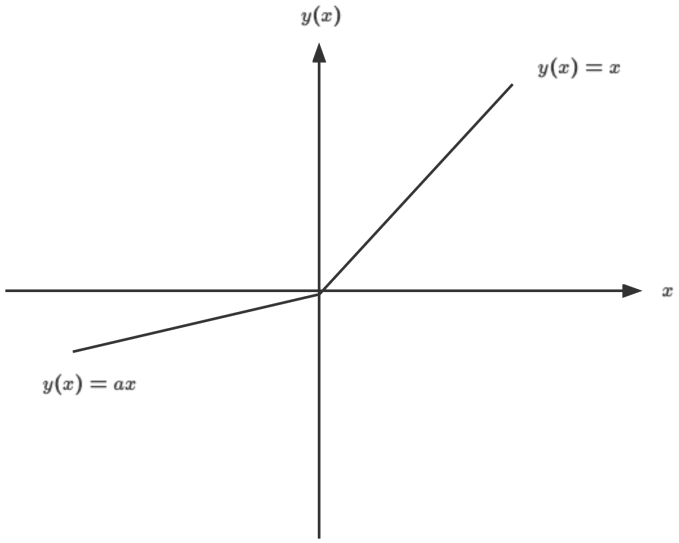

Figure 5.2: Leaky relu function.

The topology of the CNN used in this project is shown in Figure 5.3. The data are fed in parallel into a 3-layer FCNN and a CNN consisting of one convolutional layer and one pooling layer. The outputs of these two networks are combined and fed into a 7-layer FCNN, and the network's final output is the predicted planar coordinates of UE.

The FCNN and CNN used in this project share specific standard parameters, the settings of which are shown in Table 5.2. The convolutional and pooling layers have particular parameters that distinguish them from the dense layers. The settings of these parameters for CNN are shown in Table 5.3.

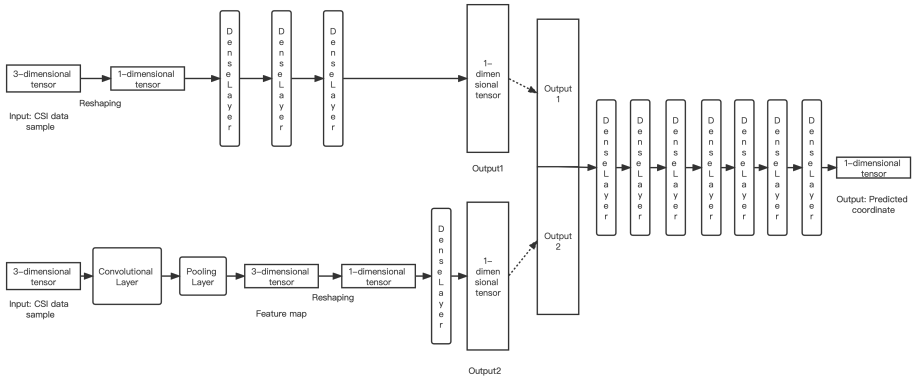

Figure 5.3: Convolutional neural network topology.

Batch size	Learning rate	Weight decay	Epoch
64	0.00005	10e-7	100

Table 5.2: Parameter setting for CNN and FCNN models.

Layer type	Kernel size	Stride	Padding
Conv layer	3	1	1
Max pool layer	2	2	0

Table 5.3: Parameter setting for CNN model only.

5.2 Performance Comparison and Analysis

After introducing the principle, structure, and training parameters of different algorithms, and dataset collection environments, it can be more intuitively understood that different algorithms and algorithm inputs have different degrees of impact on the localization performance. Therefore, this project investigates the effect of different fingerprints, antenna topologies, ML algorithms, and the number of array antenna elements on the localization performance. After getting the trained ML model, 25 percent of the data set is used as a test set to test the localization performance. The data in the test set are new data that the ML model has not learned. The localization error is the Euclidean distance between the predicted and ground truth coordinates. The localization performance in all cases is represented by the localization error's empirical Cumulative Distribution Function (CDF) curves.

5.2.1 The Impact of Fingerprints

The number of effective features of CIR is much less than that of CFR, so using the CIR as a fingerprint can save memory and computation time compared to CFR. Knowing the positioning performance of different fingerprints can guide the selection of fingerprints.

Comparison of Fingerprints Using FCNN

The FCNN algorithm is trained using 75 percent of the dataset, and the CIR and CFR fingerprint localization performance, is compared in four cases. The cases of URA, ULA, and Distributed ULAs as BSs under LoS transmission conditions and URA as BS under NLoS transmission conditions, respectively. The results are shown in Figure 5.4, Figure 5.5, Figure 5.6, Figure 5.7.

Figure 5.4 shows, under LoS transmission, the CIR and CFR fingerprint positioning performance when 64-antenna Distributed ULAs are used as the BS and FCNN is used as the fingerprint comparison algorithm. From figure 5.4, we know that there is almost no difference between the positioning performance of the CIR fingerprint and CFR fingerprint in this case.

Figure 5.5 shows, under LoS transmission, the positioning performance of the CIR fingerprint with CFR fingerprint when 64-antenna ULA is used as the BS and FCNN is used as the fingerprint comparison algorithm. From Figure 5.5, we know that the CIR fingerprint's positioning performance is better than that of the CFR fingerprint, and this difference cannot be neglected.

Figure 5.6 shows, under LoS transmission, the positioning performance of the CIR fingerprint versus CFR fingerprint when 64-antenna URA is used as the BS and FCNN is used as the fingerprint comparison algorithm. From Figure 5.6, we know that the CIR fingerprint positioning performance is slightly better than that of the CFR fingerprint. This difference is small enough to be negligible.

Figure 5.7 shows, under NLoS transmission, the positioning performance of the CIR fingerprint versus CFR fingerprint when 64-antenna URA is used as the BS and FCNN is used as the fingerprint comparison algorithm. From Figure 5.7,

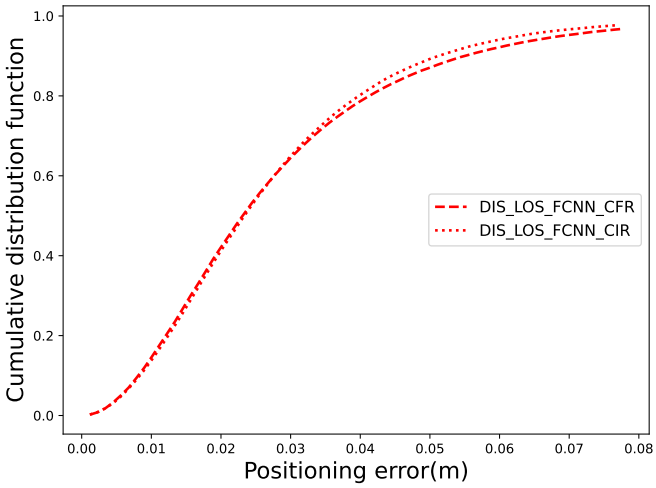

Figure 5.4: Positioning performance of CIR and CFR fingerprints using Distributed ULAs under LoS.

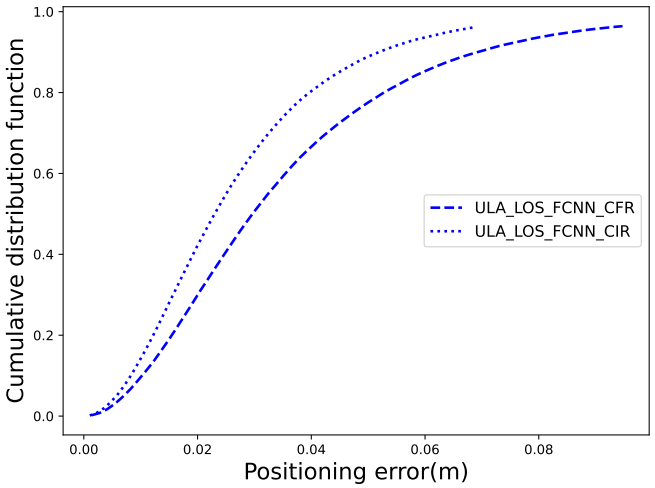

Figure 5.5: Positioning performance of CIR and CFR fingerprints using ULA under LoS.

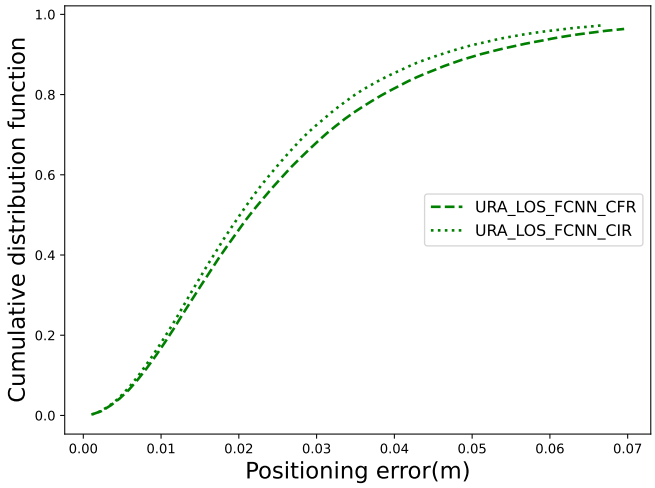

Figure 5.6: Positioning performance of CIR and CFR fingerprints using URA under LoS.

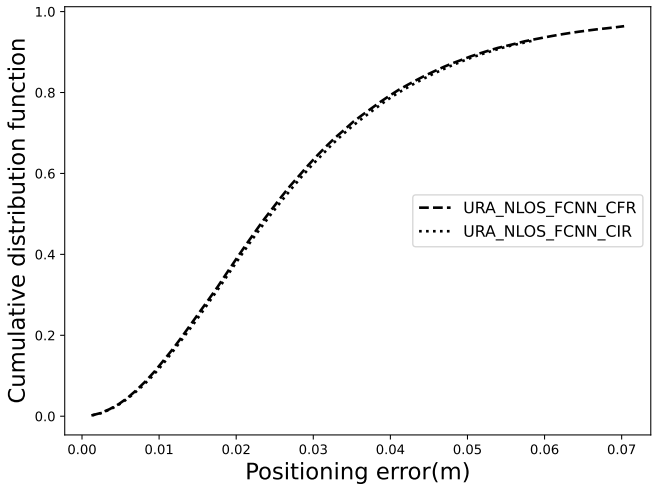

Figure 5.7: Positioning performance of CIR and CFR fingerprints using URA under NLoS.

we know that there is almost no difference between the positioning performance of the CIR fingerprint and CFR fingerprint in this case.

Comparison of Fingerprints Using CNN

The CNN algorithm model is trained using 75 percent of the dataset, and the positioning performance of the CIR fingerprint and CFR fingerprint is compared in four cases. The cases of URA, ULA, and Distributed ULAs as BSs under LoS transmission conditions and URA as BS under NLoS transmission conditions, respectively. The results are shown in Figure 5.8, Figure 5.9, Figure 5.10, Figure 5.11.

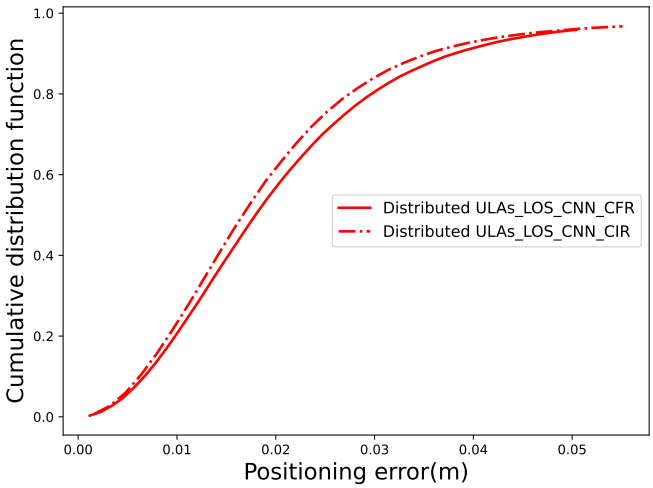

Figure 5.8: Positioning performance of CIR and CFR fingerprints using distributed ULAs and CNN.

Figure 5.8 shows, under LoS transmission, the positioning performance of the CIR fingerprint and CFR fingerprint when 64-antenna distributed ULAs are used as the BS, and CNN is used as the fingerprint comparison algorithm. From Figure 5.8, we know that the positioning performance of the CIR fingerprint is slightly better than that of the CFR fingerprint, but the difference between the two is minimal.

Figure 5.9 shows, under LoS transmission, the positioning performance of the CIR fingerprint versus CFR fingerprint when 64-antenna ULA is used as the BS and CNN is used as the fingerprint comparison algorithm. From Figure 5.9, we know that the CIR fingerprint's positioning performance is almost no different from that of the CFR fingerprint.

Figure 5.10 shows, under LoS transmission, the positioning performance of the CIR fingerprint versus CFR fingerprint when 64-antenna URA is used as the BS and CNN is used as the fingerprint comparison algorithm. From Figure 5.10, we

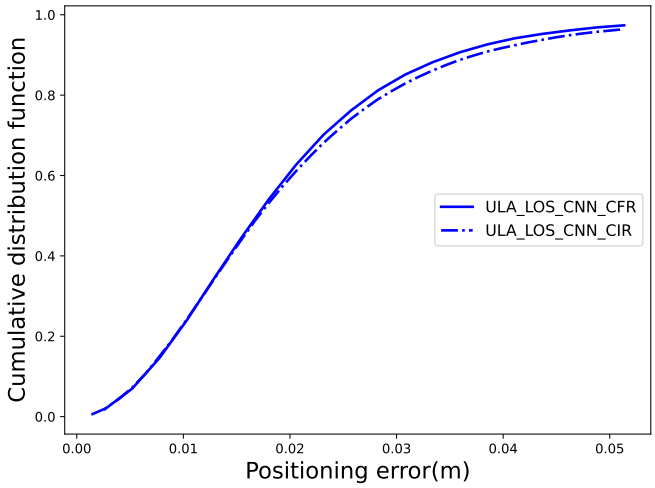

Figure 5.9: Positioning performance of CIR and CFR fingerprints using distributed ULAs and FCNN.

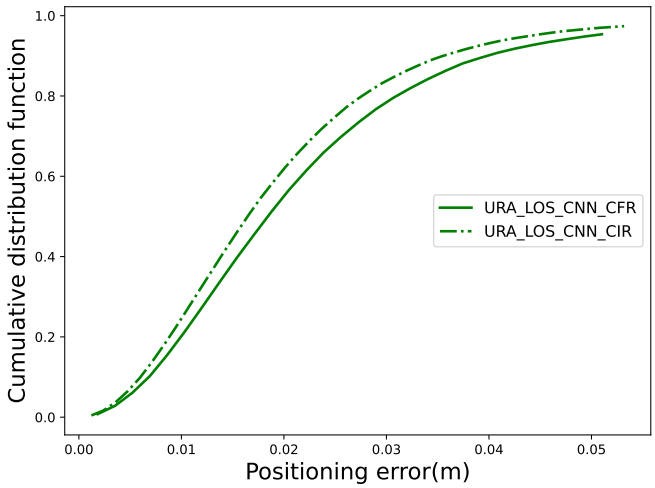

Figure 5.10: Positioning performance of CIR and CFR fingerprints using ULA and CNN.

know that the CIR fingerprint's positioning performance is slightly better than the positioning performance of the CFR fingerprint.

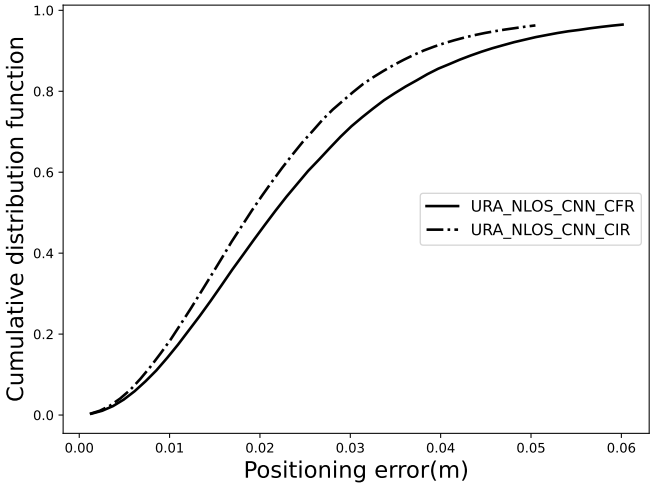

Figure 5.11: Positioning performance of CIR and CFR fingerprints using ULA and FCNN.

Figure 5.11 shows, under NLoS transmission, the positioning performance of the CIR fingerprint versus CFR fingerprint when 64-antenna URA is used as the BS and CNN is used as the fingerprint comparison algorithm. From Figure 5.11, we know that the CIR fingerprint's positioning performance is slightly better than the positioning performance of the CFR fingerprint.

Summary of The Impact of Fingerprints

In summary, the positioning performance of the CIR fingerprint is either no different from that of the CFR fingerprint or better than that of the CFR fingerprint, no matter which DNN is used to compare fingerprints. Moreover, the CIR fingerprint occupies less memory and takes less computation time. Hence, the CIR fingerprint is a better choice. In all subsequent investigations, only the CIR fingerprint is used as the input of the three ML algorithms, and this condition will not be repeated later.

5.2.2 The Impact of Array Antenna Topologies

Different antenna topologies can capture different types of angular information, have different aperture sizes, and therefore have different positioning performances. Understanding the positioning performance of different antenna topologies is a guideline for selecting antenna topologies.

Comparison of 64-antennas Topologies under LoS

In order to fully use the antenna resources, 75 percent of the dataset is used as the training set, and the positioning performance of 64-antenna ULAs, Distributed ULAs, and URA are compared using the FCNN and CNN algorithms. The results are shown in Figure 5.12 and Figure 5.13.

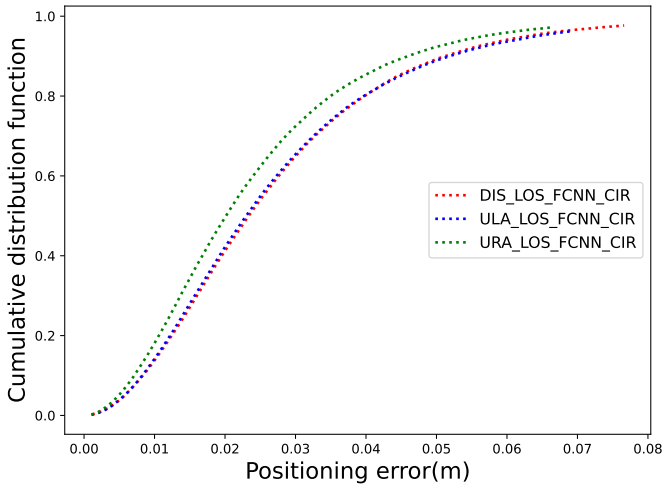

Figure 5.12: Positioning performance of ULAs, Distributed ULAs, and URA using FCNN.

Figure 5.12 shows, under LoS transmission, the positioning performance of 64-antenna Distributed ULA, ULA, and URA when using the FCNN algorithm. From Figure 5.12, we know that 64-antenna URA has the best positioning performance, and there is almost no difference between the Distributed ULA and ULA.

Figure 5.13 shows the positioning performance of the 64-antenna Distributed ULA, ULA, and URA when using the CNN algorithm. From Figure 5.13, we know that there is almost no difference in the positioning performance of the three 64-antenna topologies when the CNN algorithm is used.

Comparison of 8-antenna Topologies Under LoS

Since many application scenarios have limitations on the number of antenna elements, the unique structure of 64-antenna URA is used, from which eight antennas are selected to form 5 different topologies. Under LoS transmission, 75 percent of the dataset was used as the training set to train the FCNN and CNN models, and the positioning performance of the five 8-antenna topologies was investigated. The investigation results are shown in Figure 5.14 and Figure 5.15.

Figure 5.14 shows the positioning performance of the five 8-antenna topologies when the FCNN algorithm is used. From Figure 5.14, we know that when using

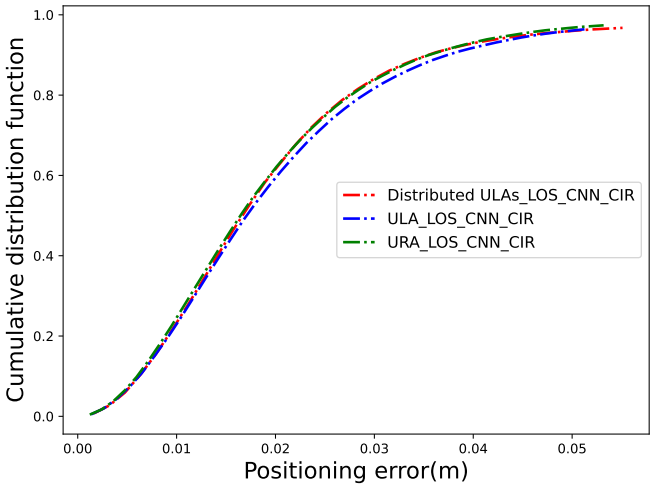

Figure 5.13: Positioning performance of ULAs, Distributed ULAs, and URA using CNN.

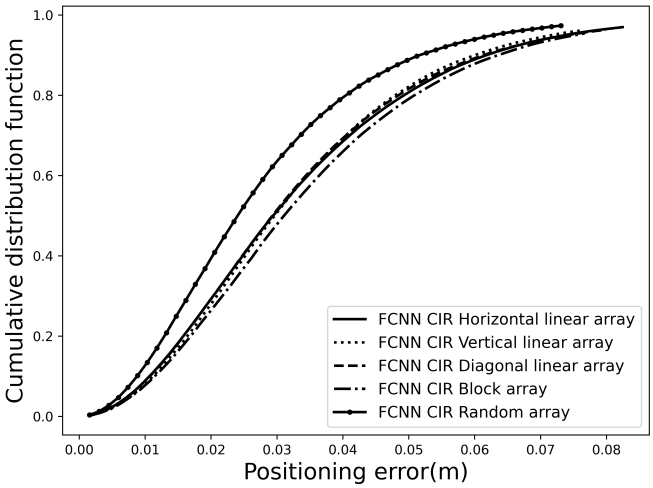

Figure 5.14: Positioning performance of different 8-antenna topologies using FCNN under LoS

the FCNN algorithm, the 8-antenna random array has the best positioning performance, and the 8-antenna block array has the worst positioning performance.

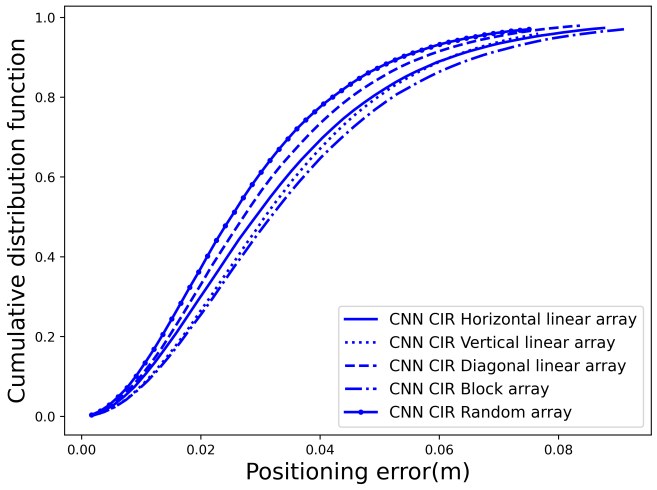

Figure 5.15: Positioning performance of different 8-antenna topologies using FCNN under LoS

Figure 5.15 shows the positioning performance of five 8-antenna topologies when using the CNN algorithm. From figure 5.15, we know that when using the CNN algorithm, the 8-antenna random array has the best positioning performance, and the 8-antenna block array has the worst positioning performance.

Comparison of 8-antenna Topologies Under NLoS

Specific application scenarios result in NLoS transmission and limit the number of antennas, so it is instructive to investigate the positioning performance of different 8-antenna topologies under NLoS transmission. Seventy-five percent of the dataset was used as the training set to train the FCNN and CNN models, and the positioning performance of five 8-antenna topologies is investigated under NLoS transmission. The investigation results are shown in Figure 5.16 and Figure 5.17.

Figure 5.16 shows, under NLoS transmission, the localization performance of five 8-antenna topologies when using the FCNN algorithm. From Figure 5.16, it can be known that there is almost no difference in the positioning performance of different 8-antenna topologies when using the FCNN algorithm.

Figure 5.17 shows, under NLoS transmission, the positioning performance of the five 8-antenna topologies when the CNN algorithm is used. From Figure 5.17, it is known that the 8-antenna horizontal uniform linear array and random array have the best positioning performance, and the 8-antenna block array has the worst positioning performance when the CNN algorithm is used.

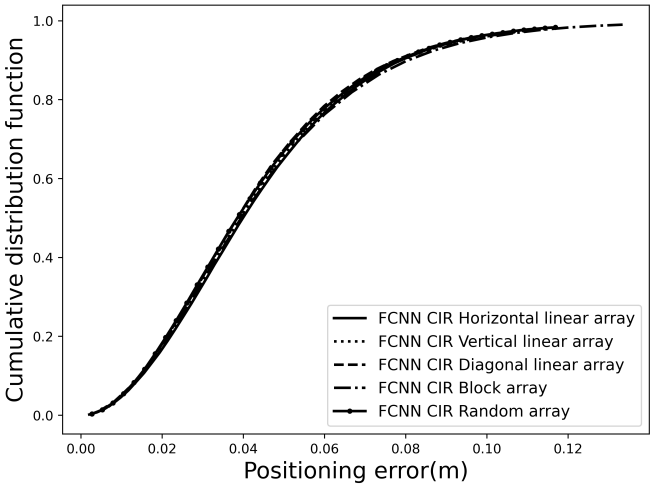

Figure 5.16: Positioning performance of 8-antenna topologies using FCNN under NLoS

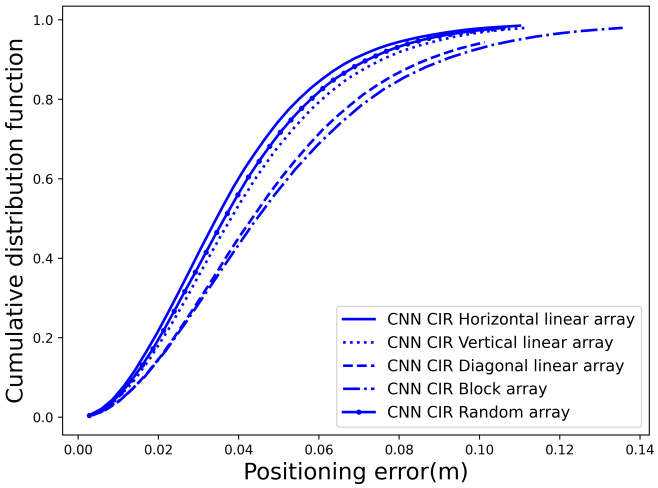

Figure 5.17: Positioning performance of 8-antenna topologies using CNN under NLoS

Summary of The Impact of Antenna Topologies

To sum up, choosing the URA in practical application is better. It has the best positioning performance when using the FCNN algorithm under LoS transmission because its unique structure can capture 3D angular information. URA may perform even better than this case when the positioning goal is 3D. The 8-antenna random array has the best positioning performance, and the block array has the worst positioning performance in most cases. It is because the random array has a bigger aperture but occupies more space because its structure is more distributed.

5.2.3 The Impact of ML Algorithms

Different ML algorithms have different structures and are based on different principles; they use different mathematical operations. Therefore, the prediction performance is different when they learn the same dataset. Investigating ML algorithms with better localization performance is instructive.

Comparison of Two DNN Algorithms

In real application scenarios, the positioning environment is sometimes LoS transmission and sometimes NLoS transmission. The positioning performance of FCNN and CNN is compared under LoS and NLoS transmission when using an 8-antenna random array. Seventy-five percent of the dataset was used as the training set, and the findings are shown in Figure 5.18 and 5.19.

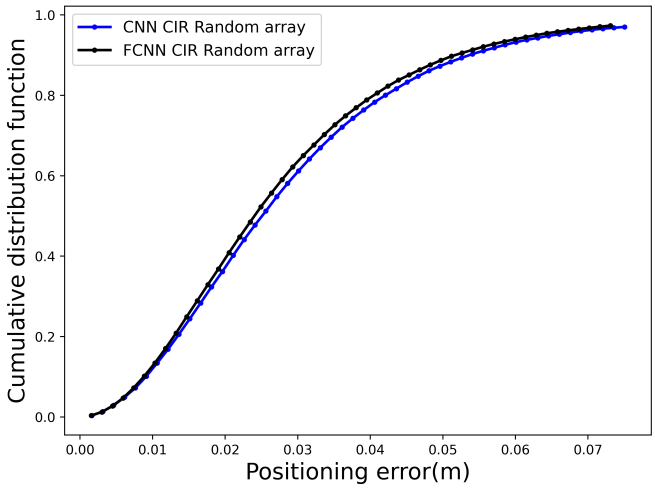

Figure 5.18: Positioning performance of FCNN and CNN algorithms using random array under LoS

Figure 5.18 shows the positioning performance of FCNN and CNN algorithms when using an 8-antenna random array as BS. From Figure 5.18, we know that the

CNN algorithm's performance is almost the same as the positioning performance of the FCNN algorithm when using an 8-antenna random array.

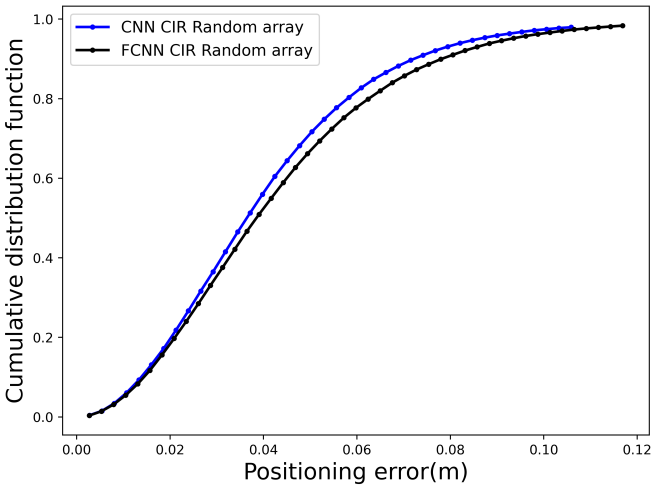

Figure 5.19: Positioning performance of FCNN and CNN algorithms using random array under NLoS

Figure 5.19 shows the positioning performance of the FCNN and CNN algorithms when using the 8-antenna random array. It can be learned that there is almost no difference in the positioning performance of the FCNN and CNN.

Comparison of Three ML Algorithms

In real application scenarios, the positioning environment is sometimes LoS transmission and sometimes NLoS transmission. Since the SVR algorithm is not suitable for large training sets, a 25 percent data set is used as the training set in order to compare the localization performance of the SVR algorithm and the two DNNs. The positioning performance of three different ML algorithms when using an 8-antenna random array as BS under LoS and NLoS transmission is investigated. The investigation results are shown in Figure 5.20 and Figure 5.21.

Figure 5.20 shows the positioning performance of three ML algorithms when using a random array of 8 antennas under LoS. From Figure 5.20, we know that the CNN algorithm's performance is almost the same as the positioning performance of the FCNN algorithm, and the positioning performance of both DNNs is much better than that of the SVR algorithm.

Figure 5.21 shows the positioning performance of the three ML algorithms. It can be learned that the positioning performance of the two DNNs is almost indistinguishable and much better than the positioning performance of the SVR algorithm.

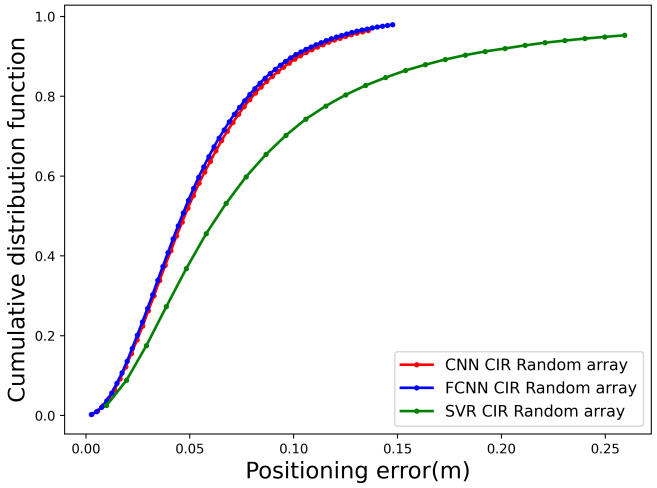

Figure 5.20: Positioning performance of ML algorithms using random array under LoS

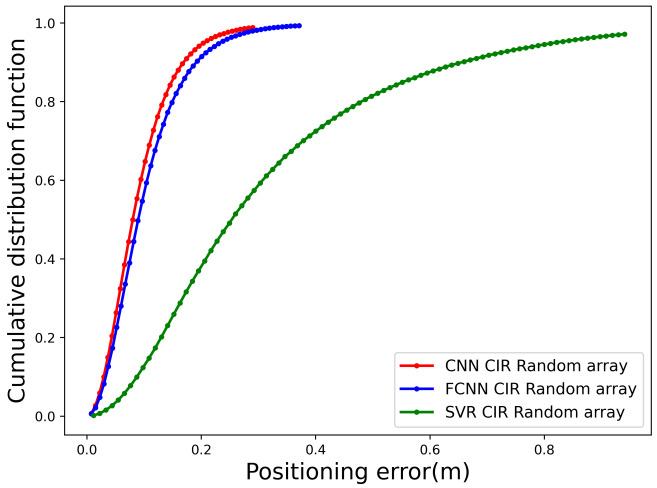

Figure 5.21: Positioning performance of ML algorithms using random array under NLoS

Summary of The Impact of ML Algorithms

In summary, the difference in positioning performance of two different DNNs algorithms is negligible when using 8-antenna arrays under LoS and NLoS transmission conditions, so it is better to choose the FCNN algorithm to save computation time and memory. The positioning performance of both DNN algorithms is much better than SVR, and SVR is unsuitable for big training sets to obtain better performance. By comparing Figure 5.19 and Figure 5.18, The localization performance of both DNNs is robust to NLoS transmission conditions when using a big training set. By comparing Figure 5.21 and Figure 5.20, it can be known that the NLoS transmission degrades the performance of fingerprinting localization when using a small training set. Moreover, we can also know that using a larger training set yields a DNN with better performance.

5.2.4 The Impact of Numbers of Antennas

The richness of angular information captured by ULA antennas with the different number of elements varies. Hence, it is instructive to understand the positioning performance of ULA antennas with different numbers of elements.

Comparison of Different Numbers of Antennas Using Large Training Sets

The positioning performance of 8-antenna, 16-antenna ULA, and 32-antenna ULA was compared using FCNN and CNN algorithms. Using 75 percent of the dataset as the training set. The results are shown in Figure 5.22 and Figure 5.23.

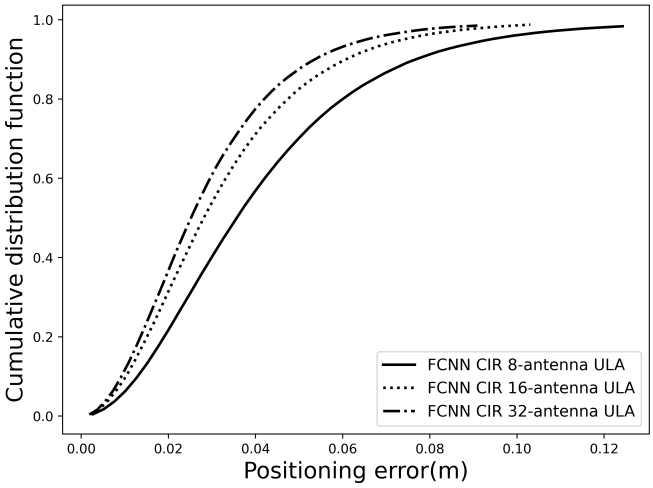

Figure 5.22: Positioning performance of different ULA antenna numbers using FCNN.

Figure 5.22 shows the positioning performance of 8-antenna ULA and 16-antenna ULA, and 32-antenna ULA when using the FCNN algorithm. From figure 5.22, we know that the positioning performance of 32-antenna ULA is better when using the FCNN algorithm.

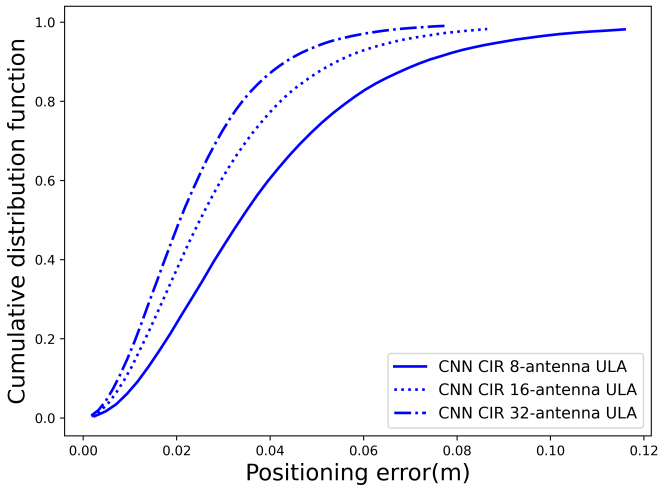

Figure 5.23: Positioning performance of different ULA antenna numbers using CNN.

Figure 5.23 shows the positioning performance of 8-antenna ULA and 16-antenna ULA, and 32-antenna ULA when the CNN algorithm is used. From figure 5.23, we know that the positioning performance of 32-antenna ULA is better when using the CNN algorithm.

Comparison of Different Numbers of Antennas Using Small Training Sets

Since the SVR algorithm is only applicable to a small training set and the computational time is also sensitive to feature number. The positioning performance of 8-antenna and 16-antenna was compared using SVR, FCNN, and CNN algorithms, respectively, using 25 percent of the dataset as the training set. The results are shown in Figure 5.24, Figure 5.25, and Figure 5.26.

Figure 5.24 shows the positioning performance of 8-antenna ULA and 16-antenna ULA when using the FCNN algorithm. From figure 5.24, we know that the positioning performance of 16-antenna ULA is better when using the FCNN algorithm.

Figure 5.25 shows the positioning performance of 8-antenna ULA and 16-antenna ULA when the CNN algorithm is used. From figure 5.25, we know that the positioning performance of 16-antenna ULA is better when using the CNN algorithm.

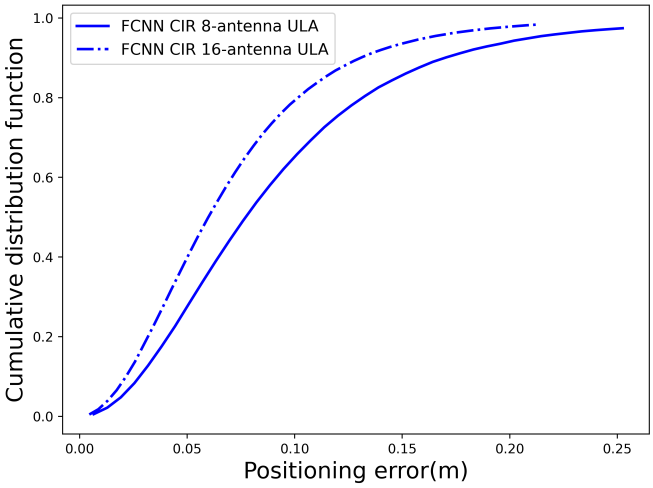

Figure 5.24: Positioning performance of different ULA element numbers using FCNN.

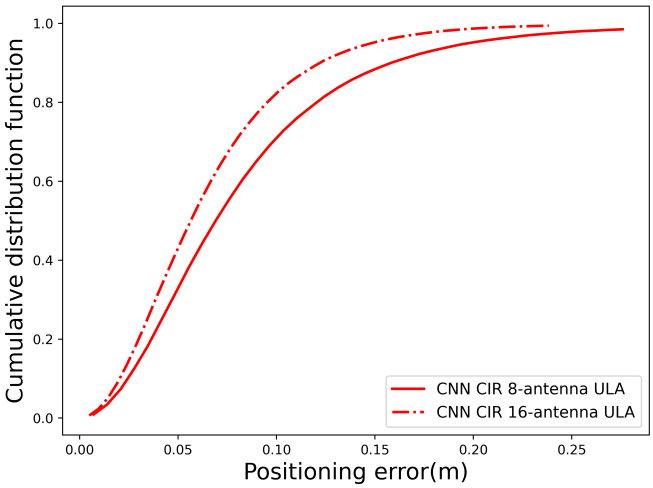

Figure 5.25: Positioning performance of different ULA element numbers using CNN.

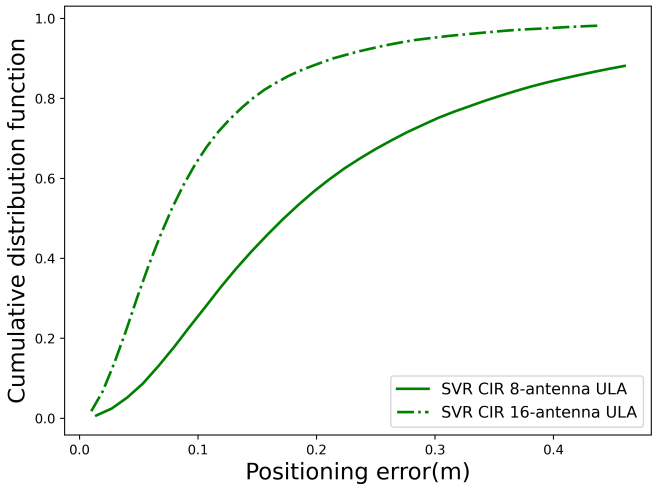

Figure 5.26: Positioning performance of different ULA element numbers using SVR.

Figure 5.26 shows the positioning performance of 8-antenna ULA and 16-antenna ULA when the SVR algorithm is used. From Figure 5.26, we know that the positioning performance of 16-antenna ULA is better when using the SVR algorithm.

Summary of The Impact of Numbers of Antennas

In summary, increasing the number of antenna elements of ULA can improve localization performance. Moreover, the SVR algorithm is more sensitive to the change in the number of antennas, and a considerable improvement in the positioning performance of SVR occurs when boosting the number of antennas.

Chapter 6
Conclusions and Future Works

The most widely used GNSS cannot achieve satisfactory positioning performance in indoor scenarios. Fingerprinting is robust to NLoS transmission for indoor scenarios. The application of ML technology to fingerprinting localization can significantly improve localization performance. KU Leuven has released four open-source ultra-dense massive MIMO indoor CSI datasets measured on the KU Leuven ESAT-TELEMIC massive MIMO testbed. Therefore, this project uses these four open-source datasets to investigate the impact of different antenna topologies, numbers of antennas, ML algorithms, and CSI fingerprints on the performance of ML-based fingerprinting localization. All comparisons are represented by empirical CDF curves of localization errors.

The CIR and CFR fingerprints' localization performance is compared under different conditions. It found that there is almost no difference between the two in most cases. The CIR fingerprint has fewer effective data features, saving memory and computation time. Therefore the CIR fingerprint is a better choice. The FCNN algorithm is a better choice because its localization performance is very satisfactory, and its performance is comparable to that of the CNN algorithm. The localization performance of the two DNNs is much better than the SVR algorithm. However, the model structure of the FCNN algorithm is simpler, so it has fewer parameters. The FCNN algorithm can save memory and computation time compared to the CNN algorithm. The FCNN algorithm using a big training set is robust to NLoS transmission. 64-antenna URA has the best localization performance when using the FCNN algorithm because its particular structure captures 3D angular information. In contrast, the other two ULAs can only capture 2D angular information. The 8-antenna random array has the best localization performance in most cases because it has the largest antenna aperture and a higher angular resolution. Therefore random array is the better choice. Increasing the number of antenna elements of ULA can improve localization performance because more elements mean more angular information, and ML algorithms can learn better when more valid data features as input. The SVR algorithm is more sensitive to the improvement in the number of antennas than the other two DNN algorithms. Furthermore, the ML-based indoor fingerprinting localization implemented in this project achieves satisfactory localization performance in all cases, partly because the four datasets used in this project are fine-grained and the channels of adjacent UE locations are highly correlated.

Due to time and computational power constraints, there was no opportunity

to study the localization performance of the SVR algorithm when using 75 percent of the data as the training set. Furthermore, since the SVR algorithm was sensitive to the boost in the number of antennas, the localization performance of the SVR algorithm may catch up with the DNN algorithm when using a larger number of antennas. In addition, the CNN in this project uses only one convolutional layer and one pooling layer; the FCNN uses only ten dense layers, so CNN with multiple convolutional layers and FCNN with more dense layers may be the next research contents. In this project, it has been demonstrated that the fingerprinting localization performance of both DNNs is very satisfactory. However, many other deep learning algorithms can be used for fingerprinting localization in the framework of deep learning that is worth exploring, such as deep belief networks, deep reinforcement learning, etc. In addition, the particular structure of the URA antenna can be used to study the effects of more different antenna topologies on fingerprinting localization performance in the future.

References

[1] C. Li, S. De Bast, E. Tanghe, S. Pollin, and W. Joseph, "Toward fine-grained indoor localization based on massive mimo-ofdm system: Experiment and analysis," *IEEE Sensors Journal*, vol. 22, no. 6, pp. 5318–5328, 2022.

[2] M. Widmaier, M. Arnold, S. Dorner, S. Cammerer, and S. ten Brink, "Towards practical indoor positioning based on massive mimo systems," in *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*, 2019, pp. 1–6. DOI: 10.1109/VTCFall.2019.8891273.

[3] W.-L. Chin, C.-C. Hsieh, D. Shiung, and T. Jiang, "Intelligent indoor positioning based on artificial neural networks," *IEEE Network*, vol. 34, no. 6, pp. 164–170, 2020. DOI: 10.1109/MNET.011.2000096.

[4] J. Yu, H. M. Saad, and R. M. Buehrer, "Centimeter-level indoor localization using channel state information with recurrent neural networks," in *2020 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, 2020, pp. 1317–1323. DOI: 10.1109/PLANS46316.2020.9109805.

[5] G.-S. Wu and P.-H. Tseng, "A deep neural network-based indoor positioning method using channel state information," in *2018 International Conference on Computing, Networking and Communications (ICNC)*, 2018, pp. 290–294. DOI: 10.1109/ICCNC.2018.8390298.

[6] L. Yin, T. Jiang, Z. Deng, and Z. Wang, "Improved fingerprint localization algorithm based on channel state information," in *2019 IEEE 1st International Conference on Civil Aviation Safety and Information Technology (ICCASIT)*, 2019, pp. 171–175. DOI: 10.1109/ICCASIT48058.2019.8973203.

[7] A. Sobehy, É. Renault, and P. Mühlethaler, "Csi-mimo: K-nearest neighbor applied to indoor localization," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6. DOI: 10.1109/ICC40277.2020.9149443.

[8] X. Ge and Z. Qu, "Optimization wifi indoor positioning knn algorithm location-based fingerprint," in *2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, 2016, pp. 135–137. DOI: 10.1109/ICSESS.2016.7883033.

[9] A. Bensky, *Wireless Positioning Technologies and Applications, Second Edition* (GNSS technology and applications series). Artech House, 2016, ISBN: 9781608079520.

[10] A. Molisch, *Wireless Communications* (IEEE Press). Wiley, 2010, ISBN: 9780470666692.

[11] T. Mitchell, *Machine Learning* (McGraw-Hill International Editions). McGraw-Hill, 1997, ISBN: 9780071154673.

[12] M. Awad and R. Khanna, *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers*. Apress, 2015, ISBN: 9781430259893.

[13] C. Bishop, *Pattern Recognition and Machine Learning* (Information Science and Statistics). Springer, 2006, ISBN: 9780387310732.

[14] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, ISBN: 9780262035613. [Online]. Available: https://books.google.co.in/books?id=Np9SDQAAQBAJ.

[15] G. E. Hinton, S. Osindero, and Y. W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, pp. 1527–1554, 2006.

[16] F. Chollet, *Deep Learning with Python*. Manning, Nov. 2017, ISBN: 9781617294433.

[17] B. Alexeev, M. A. Forbes, and J. Tsimerman, "Tensor rank: Some lower and upper bounds," in *2011 IEEE 26th Annual Conference on Computational Complexity*, 2011, pp. 283–291. DOI: 10.1109/CCC.2011.28.

[18] C. Cortes and V. Vapnik, "Support vector networks," *Machine Learning*, vol. 20, pp. 273–297, 1995.

[19] N. Cristianini and E. Ricci, *Support Vector Machines*, M.-Y. Kao, Ed. Boston, MA: Springer US, 2008, pp. 928–932, ISBN: 978-0-387-30162-4. DOI: 10.1007/978-0-387-30162-4_415. [Online]. Available: https://doi.org/10.1007/978-0-387-30162-4_415.

[20] H. W. Kuhn and A. W. Tucker, "Nonlinear programming," in *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability, 1950*, Berkeley and Los Angeles: University of California Press, 1951, pp. 481–492.

[21] Aizenman, Braverman, and Rozonoer, "Theoretical foundations of the potential function method in pattern recognition learning," *Automation and Remote Control*, vol. 25, pp. 821–837, 1964.

[22] Y. LeCun, B. Boser, J. S. Denker, *et al.*, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, pp. 541–551, 1989.

[23] Y.-T. Zhou, R. Chellappa, A. Vaid, and B. K. Jenkins, "Image restoration using a neural network," *IEEE transactions on acoustics, speech, and signal processing*, vol. 36, no. 7, pp. 1141–1151, 1988.

[24] S. De Bast and S. Pollin, *Ultra dense indoor mamimo csi dataset*, 2021. DOI: 10.21227/nr6k-8r78. [Online]. Available: https://dx.doi.org/10.21227/nr6k-8r78.

Appendix A

Training Loss Score versus Epoch Number

The training loss is the RMSE between the predicted coordinates of the machine learning and the ground truth coordinates after one training epoch. The training loss score measures the performance of an ML algorithm on the training set. In training DNNs, the training loss score increases as the number of epochs increases. However, when the number of training epochs reaches a certain level, the training score no longer increases and reaches saturation, so there is no need to increase the number of training epochs to improve the performance of the algorithm, using Figure A.1, Figure A.2, Figure A.3, and Figure A.4 to show the relationship between training loss score and epoch number. The training loss score is calculated as -10 $\log_{10} RMSE$.

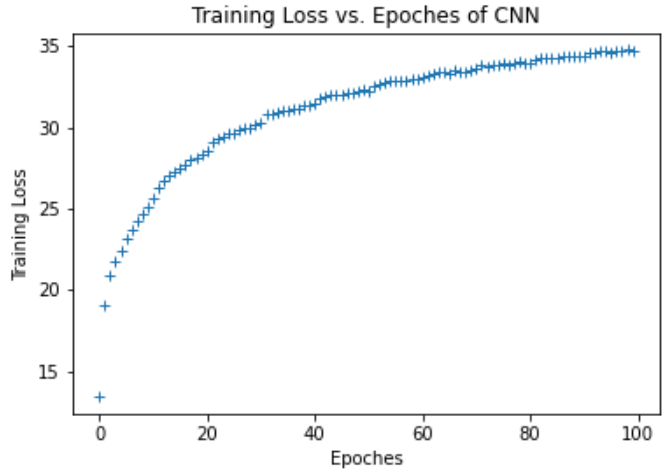

Figure A.1: Traning loss score versus epoch number using CNN and CFR.

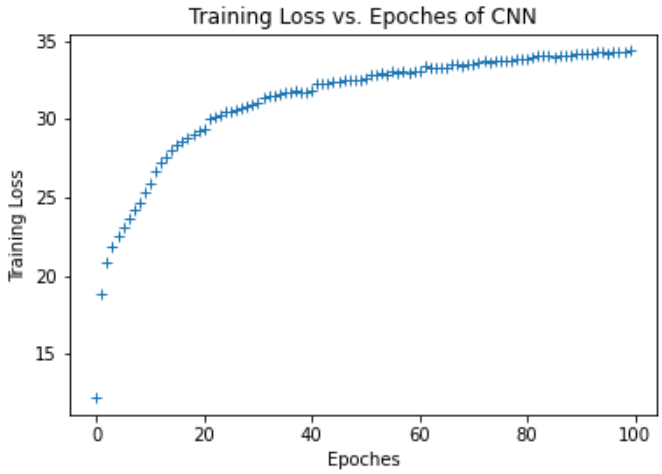

Figure A.2: Traning loss score versus epoch number using CNN and CIR.

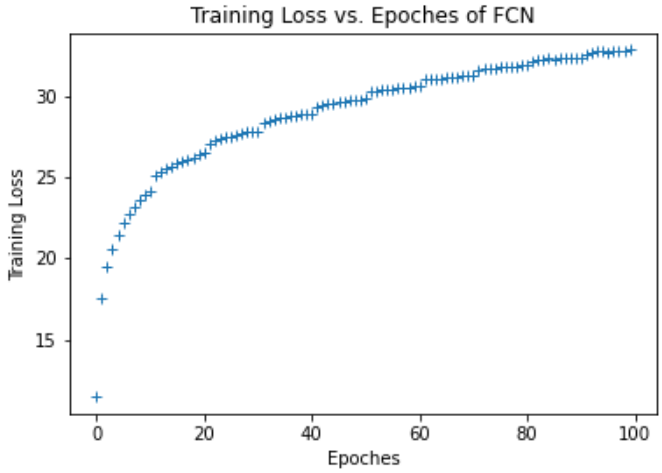

Figure A.3: Traning loss score versus epoch number using FCNN and CFR.

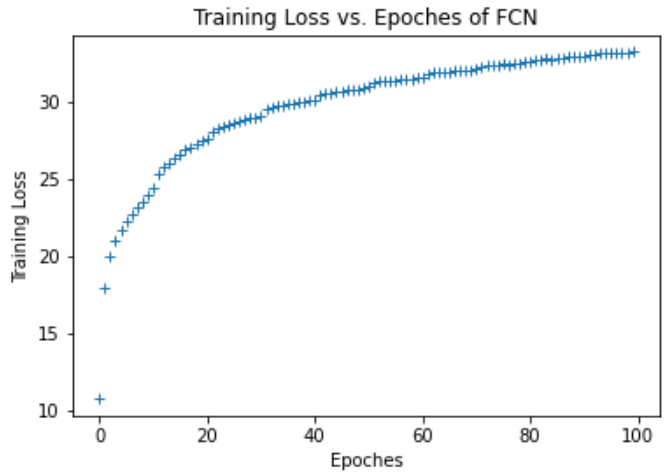

Figure A.4: Traning loss score versus epoch number using FCNN and CIR.