Politecnico di Torino

Master of Science in Mechatronic Engineering

Master's Degree Thesis

Machine learning based approach to Ultra-WideBand (UWB) indoor localization

Supervisors

Prof. Marcello CHIABERGE

Prof.ssa Marina MONDIN

Candidate

Alvin MATAROZZO

Academic Year 2021-2022

Abstract

Providing an accurate, reliable and low-cost estimate of indoor positioning remains an active area of research, despite the availability of popular localization techniques like Acoustic Systems, Infrared Systems, etc. The aim of this thesis is to introduce a new methodology for indoor localization combining Ultra-WideBand (UWB) technology with Artificial Intelligence (AI). Albeit UWB is not a new technology, it is now being revitalized and employed for wireless connections over short distances. Many companies such as Intel, Xiaomi, Sony, Samsung, Apple and Bosh claim that this technology could prove more successful than Bluetooth as it is faster, cheaper, less power consuming and more secure.

UWB is a short-range wireless communication protocol (like Wi-Fi or Bluetooth) using short radio pulses with large bandwidth. The resulting radio waves can pass through walls and other obstacles and do not interfere with different radio signals, such as those from cellular telephones. The only main limitation could be the short range, which could be easily overcome using multiple well-positioned receivers.

The localization that has been performed in this thesis uses the Channel Impulse Response (CIR) shape to understand in which subarea of the environment the antenna is positioned. The tracking is achieved by classification using Machine Learning (ML). Indeed, when the two antennas communicate with each other, what the receiver gets is a composition of the direct signal with all its reflections. The process consist in analyzing at which time these signals reach the target (or Tag), and, based on the reflections delay (which are dependent on the surrounding), estimating the position in the indoor environment.

The main steps performed in the thesis development are the collection of multiple datasets, the analysis and post-processing of the collected data, and the identification of the neural network able to offer the best classification performances.

The final selected network shows high validation accuracy when all the datasets have been combined with each other and successively split into training and validation samples. Conversely, performance deteriorates when the datasets are kept separate and used individually as training and validation sets.

Table of Contents

List of Tables

List of Figures

Acronyms

A

ACC	Accuracy
AI	Artificial Intelligence
ANN	Artificial Neural Network
AoA	Angle of Arrival
API	Application Programming Interface
AUC	Area Under Curve
autoML	Automated Machine Learning

B

BPM	Burst Position Modulation
BPSK	Binary Phase-Shift Keying
BW	Bandwidth

C

CAD	Computer Aided Design
CE	Counting Error
CIR	Channel Impulse Response
CIRE	Channel Impulse Response Estimate
CMOS	Complementary Metal Oxide Semiconductor
CNN	Convolution Neural Network
CSULA	California State University of Los Angeles

D

DL	Deep Learning
DR	DecaRanging
DT	Decision Tree

F

FCC	Federal Communication Commission
FN	False Negative
FNN	Deefforward Neural Network
FP	False Positive
FPR	False Positive Rate

I

IEEE	Institute of Electrical and Electronics Engineers
IC	Integrated Circuit
IG	Information Gain

K

KNN	K-nearest neighbor

L

LCD	Liquid Crystal Display
LOS	Line Of Sight
LR	Logistic Regression
LSTM	Long Short Term Memory

M

ML	Machine Learning

N

NLOS	Non-Line Of Sight
NLP	Natural Language Processing

O

OOK	On-Off Keying

P

PAM	Pulse Amplitude Modulation
PDP	Power Delay Profile
PDoA	Phase Difference of Arrival
PHR	Physycal Layer Header
PHY	Physical Layer
PPM	Pulse Position Modulation
PRF	Pulse Repetition Frequency
PSM	Pulse Shape Modulation

R

ResNet	Residual Neural Network
RF	Radio Frequency
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic
RSL	Received Signal Level
RSSI	Received Signal Strength Indicator
RTLS	Real-Time Location System
RX	Receiver

S

SFD	Start of Frame Delimiter
SNR	Signal to Noise Ratio
SPI	Serial Peripheral Interface
SST	Sensitivity
SVM	Support Vector Machine

T

TDoA	Time difference of Arrival
TN	True Negative
ToF	Time of Flight
TP	True Positive
TPR	True Positive Rate
TWR	Two-Way Ranging
TX	Transmitted

U

USB	Universal Serial Bus
UWB	Ultra-wideband

Chapter 1

Introduction

The term UWB signal had become always more common nowadays (particularly in the areas of radar, communications, electromagnetic interference and high-power directed energy), even if the technology itself has been discovered many years ago. The research on the UWB technology started right along with the discovery of the radio communication. It is said that the birth date of the UWB was around 1885 when Guglielmo Marconi used spark-gap transmitters for radio communication (Morse Code) over the Atlantic Ocean [1].
Of course, the usage of the UWB back then was quite different than how it is used nowadays in the commercial market, even if the main principles is the same.

Due to its military potentials applications, the use of UWB was prohibited in the USA until 1990. After 1990 first commercial, licensed products were released, leading to the general unlicensed operation (between 3.1 GHz and 10.6 GHz [2]) in 2002 from the Federal Communication Commission (FCC). Despite the public access, the technology did not become so popular in these years, mostly due to the high implementation cost and low initial performances. It took nearly two decades for the market to mature enough in order to get UWB chips available to the general public. Today, UWB chips are cheap and small enough to put them inside other devices like smartphones, becoming widely used.

1.1 UWB

UWB is a fast, secure and low power radio protocol used to determine short distance location with higher accuracy than any other wireless technology (within few centimeters). In addition, thanks to the low-power spectral density, UWB signals cause very little interference with existing narrow-band radio system.

This technology is a short-range wireless communication protocol, using radio waves as information messages (similarly to Bluetooth). In other words, UWB (standardized by IEEE 802.15.3) can also be defined as a data transmission method, in a wireless network, that offer a very large bandwidth. FCC has define these systems as those operating with an absolute BW ($BW = f_H - f_L$) larger than 500 MHz and at maximum power density at a central frequency (f_c) above 2.5 GHz (or, equivalently, with a fractional bandwidth greater than 0.25 and f_c lower than 2.5 GHz).

$$B_{frac} = \frac{(f_H - f_L)}{f_c} = 2 \times \frac{(f_H - f_L)}{(f_H + f_L)} \tag{1.1}$$

where f_H and f_L are respectively high and low frequencies at which the power spectral density is 10 dB below f_c. The main signal classification is performed by B_{frac}, as follow:

$$\begin{aligned} &Narrowband: && if\ 0.00 < B_{frac} \leqslant 0.01;\\ &Wideband: && if\ 0.01 < B_{frac} \leqslant 0.25;\\ &Ultra-wideband: && if\ 0.25 < B_{frac} \leqslant 2.00; \end{aligned}$$

1.1.1 Advantages of UWB

Differently from other radio frequency (RF) technologies, UWB was specifically designed to enable precise, secure and real-time measurements of location, while concurrently supporting two-way communication [3].
The key strength of this technology is that it calculates the distance between devices through the time-of-flight (ToF) of signals (see section 1.4 for better explanation of it). This simply difference with respect to other wireless technologies, such as Wi-Fi and Bluetooth (whose use the received signal strength indicator - RSSI) results in a more accurate precision.

As it possible to see in figure 1.1, UWB signals use much greater BW than prevalent narrowband technologies, leading to extremely short wavelength and short signals (due to the inverse relationship between time and BW: $t = \frac{1}{BW}$). Wider channel bandwidth allows more energy to be transmitted into the channel. This indicates that UWB signals have extremely high time resolution (typically in the order of nanoseconds), allowing accurate determination of ToF.

Moreover, UWB offers advantages with respect to:

- **Data transfer rate**: provides more than 100 Mb/s effective transfer rate compared with Bluetooth (1Mb/s max)
- **High performance in multipath channels**: due to the short pulse duration

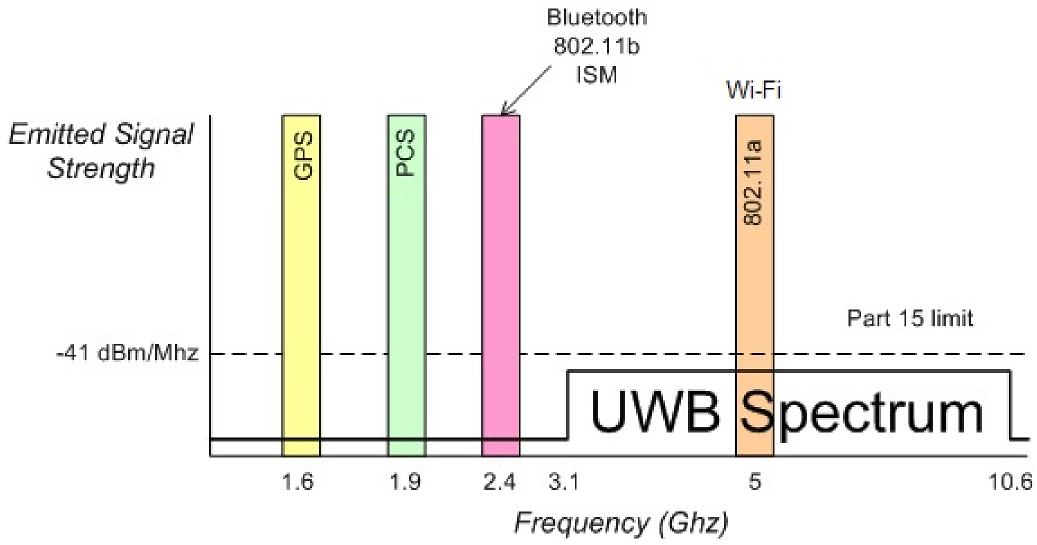


Figure 1.1: Spectrums comparison [4].

- **Low power consumption**: since the pulses are transmitted only during a small percentage of the transmission time, the average power emitted by the transmitter is very low, on the order of microwatts.
- **Simplicity of implementation**

1.1.2 IEEE 802.15.4

The IEEE established the 802.15.4 Study Group to define a new physical layer concept (which is responsible for data transmission and reception using a certain radio channel) for low-data-rate applications. The IEEE 802.15 is intended to operate in unlicensed, international frequency bands.

The working group considered many options, including using UWB technology at the air interface. The study group looks at novel applications such low-rate wireless personal area network and sensors, both demanding a moderate data throughput and long battery life.

1.1.3 Working principle

A UWB transmitter sends billions of pulses over the wide spectrum frequency (UWB was previously known as “pulse radio”). A corresponding receiver then converts the pulses into data ("0" or "1") by listening for a recognizable pulse sequence delivered by the transmitter. There are numerous "modulation techniques"

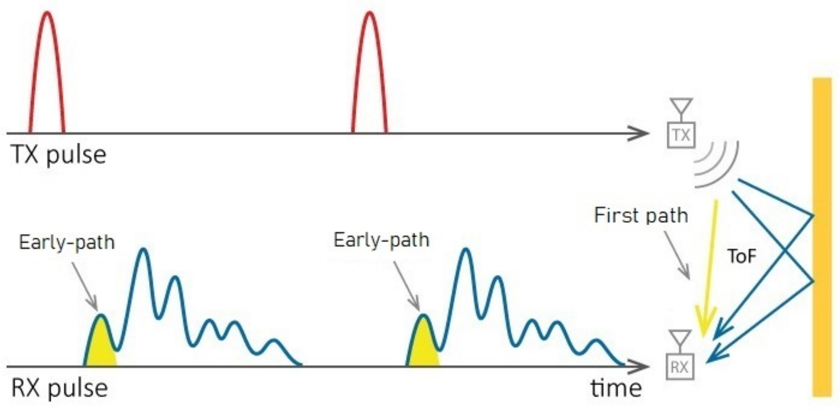


Figure 1.2: Working principle [5]

to encode pulse as digital numbers (see sec. 1.1.4).
The goal to send several pulses in very short time is to achieve the real-time accuracy. This is possible with UWB thanks to the fact that pulses are sent one every two nanoseconds.

1.1.4 Modulation techniques

Every time data need to be transmitted from one device to another, modulation takes a significant role. Modulation is the process of converting digital numbers (i.e. bits) to electrical signal (optimized for transmission).
There are several types of modulation, each of which is intended to change a certain feature of the carrier wave. Unlike other radio technologies, UWB does not use amplitude or frequency modulation to encode the information that its signals carry. Instead, UWB uses short sequences of very narrow pulses with a binary phase-shift keying (BPSK) and/or burst position modulation (BPM).

The main modulations techniques for UWB are depicted in figure 1.3, which are:

- **Pulse Amplitude Modulation (PAM)**: the classical binary PAM is implemented using two antipodal Gaussian pulses, as shown in figure 1.3 (a). The transmitted amplitude's pulse signal s(t), can be represented ad:

$$s(t) = \sum_{k=-\infty}^{\infty} a_k \, p(t - kT_f) \tag{1.2}$$

  where a_k is the amplitude of the pulse p(t) and T_f is the frame duration.

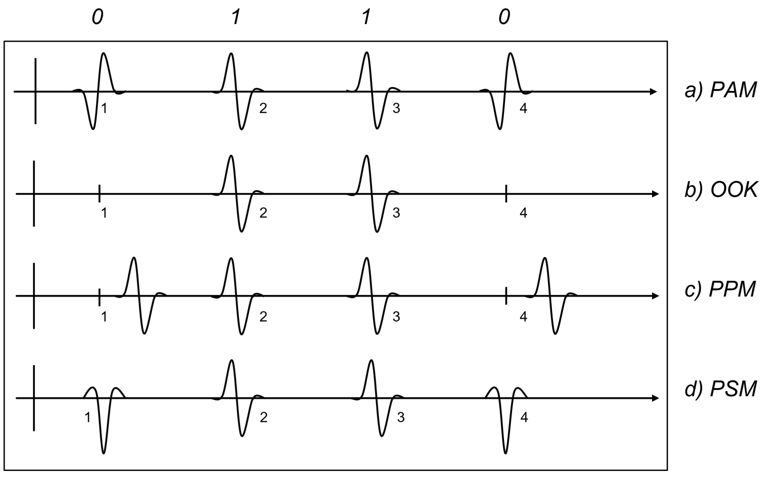


Figure 1.3: UWB modulations techniques

- **On-Off Keying (OOK)**: as illustrated in Fig. 1.3 (b), the pulse is sent only if represent "1", while no pulse to describe "0".
- **Pulse Position Modulation (PPM)**: with this modulation, the information of the data to be transmitted is encoded by the position of the impulse with respect to a nominal position (Fig. 1.3 (c)). More precisely, each pulse is delayed or sent in advance of a regular time scale, according to:

$$s(t) = \sum_{k=-\infty}^{\infty} a_k \; p(t - kT_f \pm \delta_{shift}) \tag{1.3}$$

  where, as for PAM, a_k is the amplitude of the pulse p(t), T_f is the frame duration and δ_{shift} is the pulse shift.
- **Pulse Shape Modulation (PSM)**: as depicted in Fig. 1.3 (d), PSM is an alternative to PAM and PPM modulations. Here, the information data is encoded by different pulse shape. This requires a suitable set of pulses for higher order modulations.

1.1.5 Topologies

In order to satisfy different needs, UWB technology has been thought in order to be implemented in different ways. Before proceeding describing all these methodologies, it is important to define "anchor" and "tag".
An anchor generally is an electronic UWB device with a well-known position, while tag refers to a mobile one [3]. The typical setup is, in general, fixing the anchor in

a wide area and the tag on the element needed to be localized. To establish the distance between them they have to exchange a lot information, which includes CIRs (par. 1.2) and ToF (par. 1.4).
It is worth to highlight that nowadays, most of the commercial devices can act either as an anchor, as a tag, or both. This include the boards used in this project to collect the dataset, where with a simple switch, they could change their setup (par. 1.5).

As mentioned before, there are several way to estimate the distance between two or more antennas, and depend mostly on the topology it has been decided to use. The main one are:

- **Two-way ranging (TWR):** this method determines the distance between a tag and an anchor by measuring the time UWB's RF signals takes to move back and forth between the two antennas (ToF). Once the ToF is stored, the distance computation become straightforward (eq. 1.4):

$$\begin{gathered} d = ToF \times c \\ ToF = \frac{T_{round1} \times T_{round2} - T_{reply1} \times T_{replay2}}{T_{round1} + T_{round2} + T_{replay1} + T_{replay2}} \end{gathered} \tag{1.4}$$

  where d is the distance [m], c is the speed light (299'792'458 [m/s]) and T_i are defined in figure 1.4.

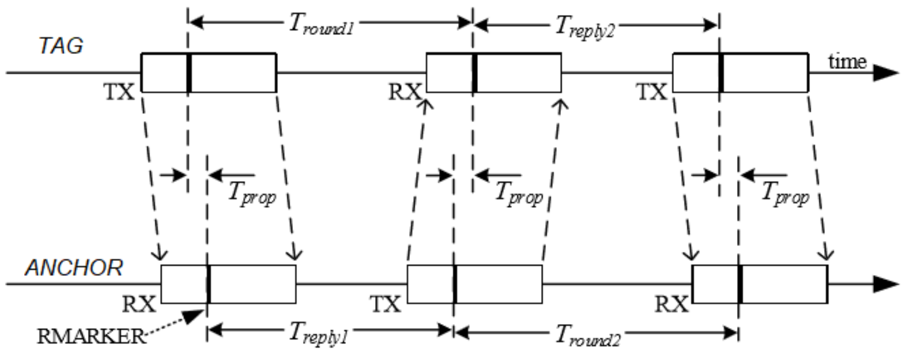


Figure 1.4: Two-way ranging between anchor and tag

As depicted in Fig. 1.4, the tag initiates TWR by sending a first signal (poll message) with the known address of an anchor. The anchor records the time it receives the poll message and send a response. When the tag receives the new message, it computes the ToF based on the round-trip time and reply time. Once the distance has been computed, the tag share the information through a final message (if required).

In case is desired to increase the accuracy, multiple anchors can be used. The disadvantages of using multi TWR for location measurements, is that the tag has to communicate contemporaneously with all the anchors, increasing the power consumption.
In this project, single TWR has been the selected setup.

- **Time difference of arrival (TDoA):** this setup is based on measuring the time difference between signals arriving at different anchor sensors. As it can bee seen in Fig. 1.5, multiple anchors are deployed in fixed and known locations and since the distance is computed on the relative difference, they must be accurately synchronised (running on the same clock).
  With TDoA, tags sends brief "blink messages" in regular intervals. Each anchors that receive the message, timestamps its arrival and forward it to a central location engine Real-Time Location System (RTLS), which runs multilateration algorithms to determine the device's locations.

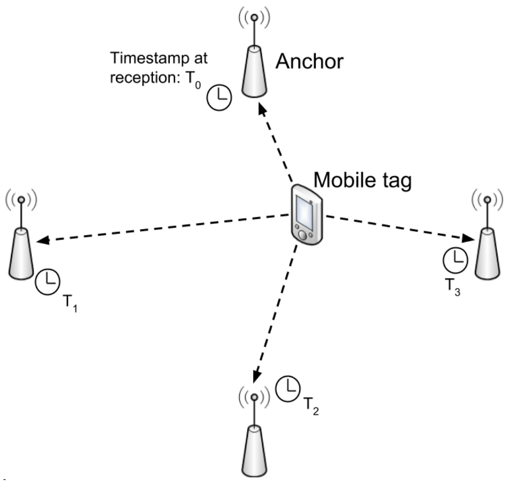


Figure 1.5: Time difference of arrival scheme [6]

- **Phase difference of arrival (PDoA):** the PDoA technology is used to determine the tag's coordinates (x, y). The required devices are 2 antennas (with known orientation and distance between each others) and a tag (Fig. 1.6). The tag sends the signal to both the antennas and once the signal arrives to the antennas, the difference in phase between them is measured, obtaining the angle of arrival (AoA). The final position is computed combining the latter information with the relative antennas' distance.

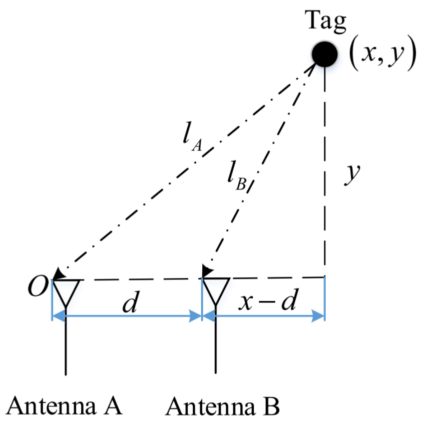


Figure 1.6: Phase difference of arrival scheme [7]

1.2 CIR

UWB channels can be measured by sounding the channel with pulses, and thereby obtaining the Channel Impulse Response (CIR). Since the 2 antennas communicate with electromagnetic signals, it's needed to compute the magnitude from the quadrature and phase component.
The main elements of the CIR are the multipath components, that are related on the type of transmitter and the surrounding area. The fact that multipaths are associated to the around environment gives the possibility to associate a single CIR, ideally speaking, to a unique point of the environment itself. Indeed, as it can see from figure 1.7, there are different peaks in the channel response, and all of them correspond to a specific multipath, which is a delay of the received signal due to objects reflections. Taking as reference figure 1.8, is possible to understand the relation between transmitted signal, multipath and CIR (fig. 1.7). The longer the path, the more delayed the peak appear in the final CIR.

The CIR could be divide in 3 main sections:

1. **Noises**: part of the CIR where there is no transmitted signal, and the receiver capture only noise.
2. **First path**: it represents the shortest path between transmitter and receiver. It is depicted in the CIR by the first peak and from its magnitude is possible to understand if it is LOS or NLOS.
3. **Multipath/delays**: they are all the peaks that from reflections on the surroundings reach the transmitted signal at different time. For example, looking at figure 1.8, the bottom red NLOS path will be represented as a

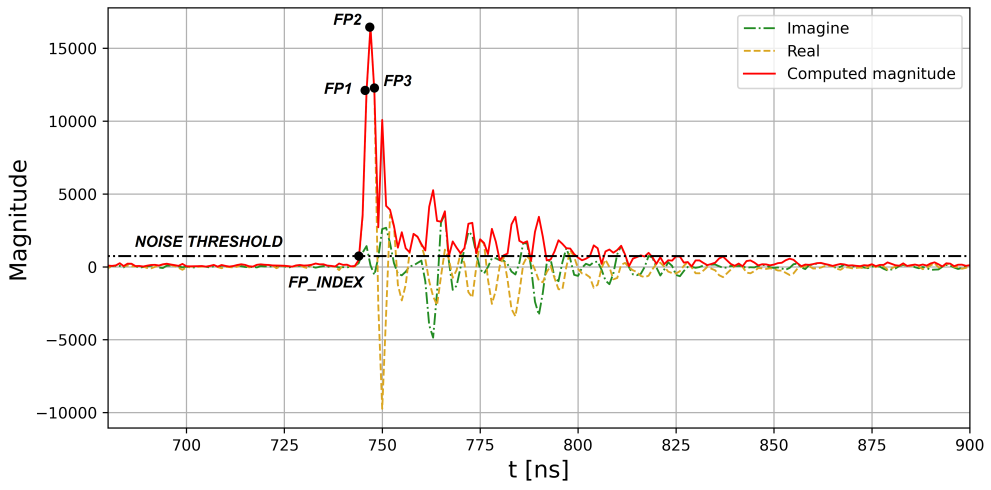


Figure 1.7: Example of a CIR

closer peak to the first-path with respect to the upper one, where the distance is longer.

The sections which have been analyzed for this thesis are "First path" and "Multipath". Indeed, the noise before and after the signal is somehow useless. This led us to consider only 157 samples, starting from FP_INDEX (time instant in which the signal overcome the noise threshold) to 157 successive samples. This CIR's subsection is the one which effectively provide information about the surroundings. In fig. 1.7, more than 157 samples have been reported, but only to show the signal with a portion of its noise.

As mentioned before, the acquired channel response is just a simple array of complex values. With reference to fig. 1.7, the green line is the plot of the imaginary values, the yellow line is the real values, while the red line is the computed magnitude values. It is also possible define 4 different values in the CIR:

- **NOISE THRESHOLD:** it is referred to the maximum magnitude that the noise could reach, after which it is no more noise, but it is recognized ad transmitted signal. An internal algorithm computes the threshold based on the noise/signal found in the first 200-300 samples of the accumulator.
- **FP_INDEX**: it is the index on which the received signal overcome the noise threshold.
- **FP1**: this is the amplitude of the first point after FP_INDEX. In case the

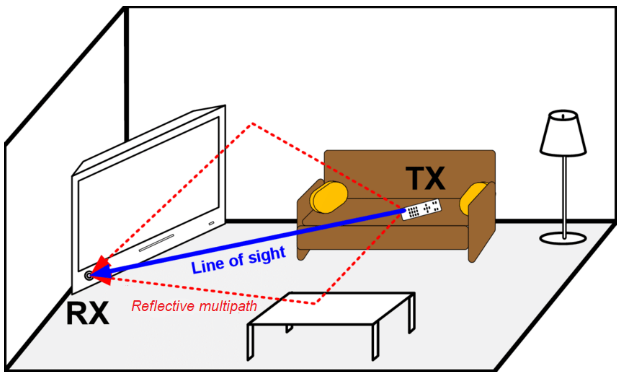


Figure 1.8: Example of multipath (LOS and NLOS) [8]

index is not an integer, it is rounded to the greater integer (ceiling operation).

- **FP2**: this is the amplitude of the second point after ceiling FP_INDEX.
- **FP3**: this is the amplitude of the third point after ceiling FP_INDEX.

These latter values are used to assess the signal power and estimate the receive signal power (sec. 1.5.5).
The indoor radio propagation channel can be modeled as a linear time-varying filter, with the following impulse response:

$$h(t,\tau) = \sum_{k=1}^{K} a_k(t)\delta(\tau - \tau_k(t))e^{j\theta_k(t)} \tag{1.5}$$

where τ is the delay, t refers to the impulse response at instant t and δ is the Dirac delta function transmitted at t = 0, while all the parameters at the k^{th} path are $a_k,\ \tau_k\ and\ \theta_k$. This is a generalized model that may be used to derive the channel impulse response to any sent signal s(t), by convolving s(t) with h(t, τ) and adding noise.

In case the the aim is specifically determine the UWB channel characteristic for a base-band UWB signal, equation 1.5 can be reduced as follow:

$$h(t,\tau) = \sum_{k=1}^{K} a_k(t)\delta(\tau - \tau_k(t)) \tag{1.6}$$

To calculate the estimate of the CIR data (fig. 1.9), $y^{'}(t)$ is down-converted to the baseband. Successively, a low-pass filter is used to remove irrelevant signal components.

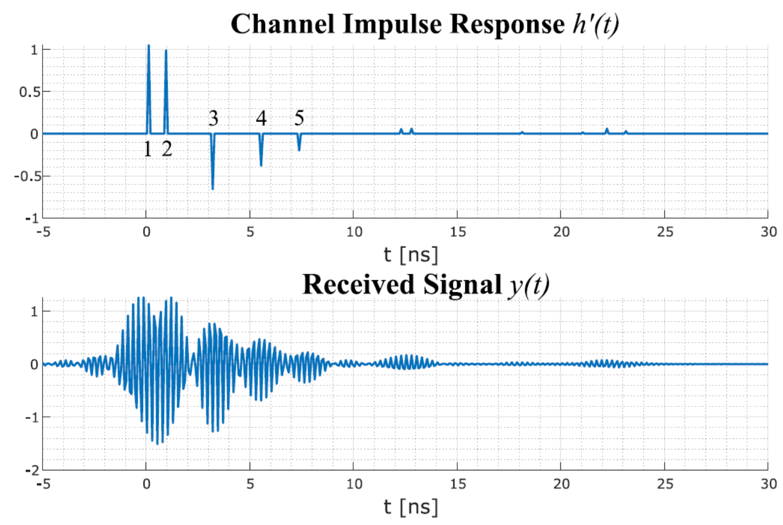


Figure 1.9: Convolution of an exemplary CIR $h^{'}(t)$ with reference pulse to estimate $y^{'}(t)$ [9]

LOS

LOS stands for Line of Sight. Communication in LOS is possible when there is no obstruction between transmitter and receiver. This condition provides strong signal strength and greater throughput due to less attenuation on the path.

NLOS

NLOS, instead, stands for Non Line of Sight. It means that the communication happen when natural and/or man made structure block the path between the two antennas. Anyway, the signals are able to reach the receiver, with the only difference that they have to go through many obstruction, getting attenuate. A typical result of NLOS, is to obtain the first component smaller then the following multipath signals. This is the consequence of having an attenuation due to material's obstacle. Due to NLOS, multiple copies of the signals arrive at different times with different amplitudes.

It is clear that distint NLOS cases exist. For example, it is possible to differentiate a case where the signal is mainly obstructed by relatively low attenuation materials such as plasterboard and doors (soft NLOS), and a case where it is mostly obstructed by high attenuation materials, such as multiple concrete walls (hard NLOS).

1.3 SNR

SNR, or Signal-to-Noise Ratio, is defined as the ratio between the desired transmitted information (or power) of a signal and the undesired signal (background noise).

$$SNR = \frac{P_{signal}}{P_{noise}} \tag{1.7}$$

where P is average power [10]. Both signal and noise power must be measured at the same or equivalent points in a system, and within the same system bandwidth. Because many signals have a very large dynamic range, they are frequently represented in logarithmic decibels.

$$SNR_{dB} = 10 \log_{10}(\frac{P_{signal}}{P_{noise}}) = 10 \log_{10}(P_{signal}) - 10 \log_{10}(P_{noise}) \tag{1.8}$$

However, when the signal and noise are measured in volts (V) or amperes (A), which are amplitude measurements, they must first be squared to get a power proportional quantity.

In wireless technology, the key to device communication is the ability to distinguish the applied signals as legitimate information from any background noise, or signals on the spectrum.
As easy to guess, the goal in the transmission is to have SNR as higher as possible, meaning that the signal is strong enough to travel in the channel, and being easily distinguishable from the receiver.

1.4 ToF

Time of Flight (ToF) is the time an item, particle, or wave (whether acoustic, electromagnetic, etc.) takes to travel a distance across a medium. This information can then be used to measure velocity or path length, or as a way to learn about the particle or medium's properties (such as composition or flow rate) [11].
The basic way to compute the distance is the following:

$$d = c \cdot t \tag{1.9}$$

where c is the speed light [m/s] and t is the effectively time of flight (from TX to RX in fig. 1.10).

As anticipated in sec. 1.1.5 (and with reference to fig. 1.4), when a signal is transmitted it must be take care to use the real time of flight. It means that all the delays in the signal communication must not be considered in the final computation.

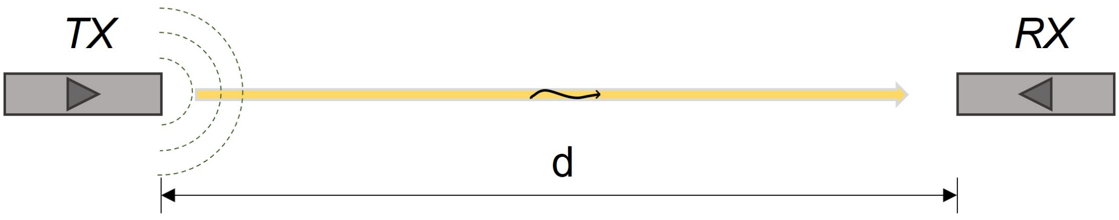


Figure 1.10: Schematic representation of signal transmission

1.5 EVK1000

The EVK1000 consist of a pair of EVB1000 boards, where each of the pair is configured to run a pre-programmed "DecaRanging" two-way ranging application. The main purpose of the application is to control the IC mounted on the boards to exchange messages, calculate ToF, estimate the resultant distance between the two boards and display the final result on the on-board display [12].

The boards may optionally be driven via USB interface using a PC version of the "DecaRanging" software, or through a simple DC power supply (or battery) with Voltage from 3.6 to 5.5V, if the program is already present on the IC.

1.5.1 EVB1000 board description

The EVB1000 evaluation board is 7 cm × 7 cm in size. Figure 1.11 depicts its two sides, which indicate the key components. The LCD display, situated in the front side, is used to show ranging information and the operating mode of the board, whose is possible to select from the DIP switch (S1). Moreover, while in use, it shows also information about the computed distance.

The rear side contains the DW1000 IC, the ARM IC, the ARM reset button, two DIP switches (S2 and S3), the JTAG connection header, the external SPI connection header, and various jumpers and power connectors for configuring the input powering mode. For more explanations on these components, or setup mode, see "evk1000_user_manual [12]".

1.5.2 DW1000 (UWB) transceiver IC

The DW1000 is a fully integrated low power, single chip CMOS radio transceiver IC compliant with the IEEE 802.15.4-2011 (par. 1.1.2) UWB standard.
The main characteristics of this transceiver, based on "DW1000 User Manual" [13], are:

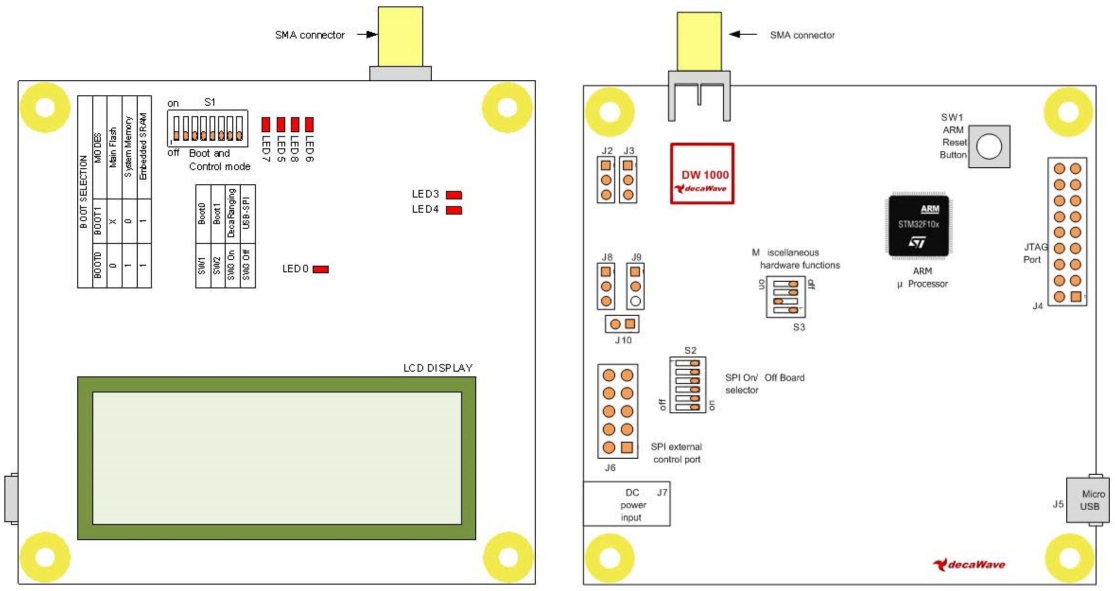


Figure 1.11: Main EVB1000 components

- Facilitation of the proximity detection to an accuracy of +/- 10 cm using TWR time-of-flight measurements.
- Spans of 6 RF bands from 3.5 GHz to 6.5 GHz
- Support of data rates of 110 kbps, 850 kbps and 6.8 Mbps
- Its high data rates allow it to keep on-air time short and thereby save power and extend battery lifetimes
- The ability to deal with severe multipath environments makes it ideal for highly reflective RF environments

The IEEE 802.15.4-2011 standard [14] specifies a chipping rate of 499.2 MHz. This frequency is the same the DW1000 system clocks are set at.

1.5.3 Transmitted signal

The UWB technology is based on transmission and reception of specific frames. The general structure of this frame is depicted in figure 1.12.
It starts with a synchronisation header (made up of single pulses) that includes the preamble and the SFD (Start of Frame Delimiter), followed by the PHY header (PHR), which determines the length (and data rate) of the frame's data payload. The data portion is transmitted as burst, with specific modulation (BPM or BPSK).

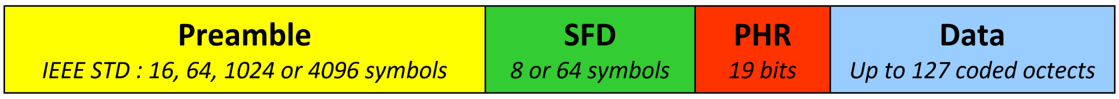


Figure 1.12: UWB PHY frame structure

Preamble

The preamble consist of a fixed sequence of pulses, without carrying the identity of the sender. Its length, standardize by IEEE 802.15.4a [14], is of 16, 64, 1024 or 4096 symbols and is strictly related to the positioning system demands and performance. Indeed, larger packet size helps low quality receivers to gain higher SNRs (because travel more information), while smaller packet size reduces the channel occupancy, leading to more efficient energy consumption and an increase of devices in the same channel. It needs to be chosen in conjunction with the data rate. There is no point of using a very long preamble with a fast data rate at long range because the receiver will not be able to receive the data irrespective of the length of the preamble. However at slow data rates longer preambles give an increase in operating range. To maximize range, a slow data rate in conjunction with a long preamble (2048) should be chosen.
The symbols used in the preamble part of a packet is one of the eight symbols

Table 1.1: Symbol set present in the preamble

Type	**Symbol**
1	-1000010-1011101-10001-111100-110-100
2	0101-10101000-1110-11-1-1-10010011000
3	-11011000-11-111001101000-10000-1010-1
4	00001-100-100-1111101-1100010-10110-1
5	-101-100111-11000-1101110-1010000-00
6	1100100-1-1-11-1011-10001010-11010000
7	100001-101010010001011-1-1-10-1100-11
8	0100-10-10110000-1-1100-11011-111101000

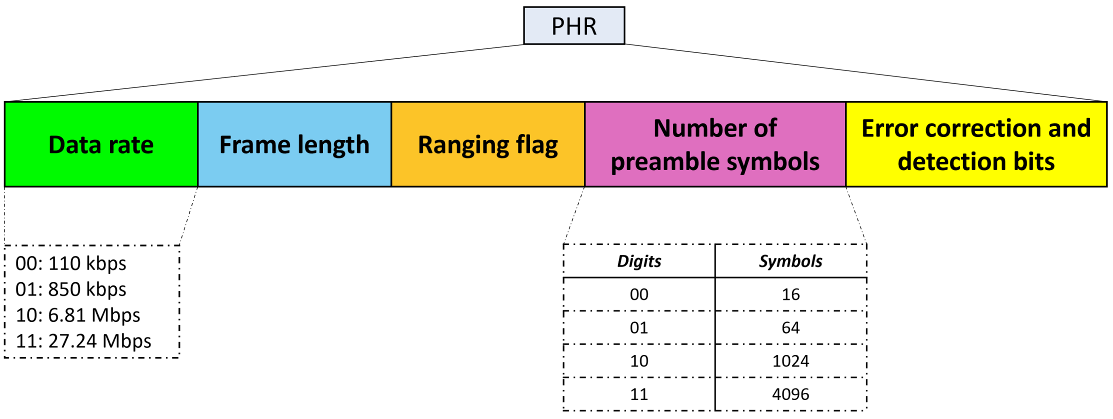
present in table 1.1. Each preamble symbol has an approximately duration of 1 μs. The important property they have, is the perfect periodic auto-correlation, which reduce the ranging error caused by multipath propagation.

Start of frame delimiter

The SFD, composed by only 8 or 64 symbols, is a short sequence used to established the completion of the preamble section of the frame, and the commencement of the payload section (PHR). The SFD is a narrow signal used to trigger starting and stopping of time counting, necessary for a precise timing.

PHR

The length of PHR is 19 bits. The information it carries is data rate, frame length, ranging flag, preamble length, error correction and detection bits. The information



Digits	Symbols
00	16
01	64
10	1024
11	4096

Figure 1.13: PHR structure

in the data rate and preamble length are represented as two bits, as shown in Fig. 1.13, while the whole PHR package is transmitted at the mandatory data rate. Lower data rates have longer range than higher data rates, so, to maximise range, the lowest data rate (110 kbps) should be selected.

1.5.4 UWB channels and sampling period

Despite the IEEE 802.15.4 standard UWB PHY defines 16 different channels, those supported by DW1000 are only 6 and are listed in table 1.2. The preamble codes specified by the standard for a specific channel has been chosen to have a low cross correlation factor with other channels, with the intention to let each channel operate independently from each other.

The DW1000 measures the CIR upon packet received with a sampling period $T_s = 1.0016$ ns and stores it in a large internal buffer. The time span of the CIR is the duration of the preamble symbol: 1016 samples for a 64 MHz pulse repetition

Table 1.2: UWB channels supported by DW1000

Channel number	**Centre frequency (MHz)**	**Bandwidth (MHz)**	**Preamble Codes (16 MHz PRF)**	**Preamble Codes (64 MHz PRF)**
1	3494.4	499.2	1,2	9,10,11,12
2	3993.6	499.2	3,4	9,10,11,12
3	4492.8	499.2	5,6	9,10,11,12
4	3993.6	1331.2	7,8	17,18,19,20
5	6489.6	499.2	3,4	9,10,11,12
7	6489.6	1081.6	7,8	17,18,19,20

frequency (PRF) or 992 for a 16 MHz PRF. Each sample is a complex number whose real and imaginary parts are 16-bit signed integers. In order to move from sampling period to sampling frequency, it is simply needed to compute the inverse: $T_s = 1.0016ns$ means that $f_s = \frac{1}{T_s} = \frac{1}{1.0016} = 998.4MHz \cong 1GHz$.

1.5.5 Quality of reception and RX timestamp

The DW1000 receiver is able of receiving messages under many different conditions. In a network, it may be beneficial to examine the quality of message receipt from a specific node in order to adjust network routing or settings associated to that node, such that the communication reliability could increase. For example to improve communications reliability the frame length might be shortened, or the data rate might be reduced, or the preamble length might be increased.

In order to assess the quality of a received message and any related timestamp, is possible to use different information:

- **Standard Deviation of Channel Impulse Response Estimate (CIRE) Noise:** the standard deviation of the noise can be used as an absolute value or it may be compared with the First Path Amplitude. Higher CIRE and more probably the quality of receive timestamp is poorer. This condition could be associated to the case when the real first path is irretrievable buried in the noise.
- **Estimate receive power and estimate power:** using these two calculations it may be possible to say whether the channel is LOS or NLOS. As a rule of thumb, if the difference between the estimate recevie power and the estimate

power is less than 6 dB, the channel is likely to be in LOS, while if the difference is greater than 10 dB the channel is likely to be NLOS [13].

Estimating the signal power in the first path

To estimate the power in the first path signal (in dBm), is possible to use eq. 1.10:

$$First\ Path\ Power\ Level \ = \ 10 \times \log_{10}(\frac{F1^2 + F2^2 + F3^2}{N^2}) - A \tag{1.10}$$

where:

- F1, F2 and F3 are the amplitude of the respectively point FP1, FP2 and FP3.
- A is the constant 113.77 for a PRF of 16 MHz, or, the constant 121.74 for a PRF of 64 MHz.
- N is the preamble accumulation count.

Estimating the receive signal power

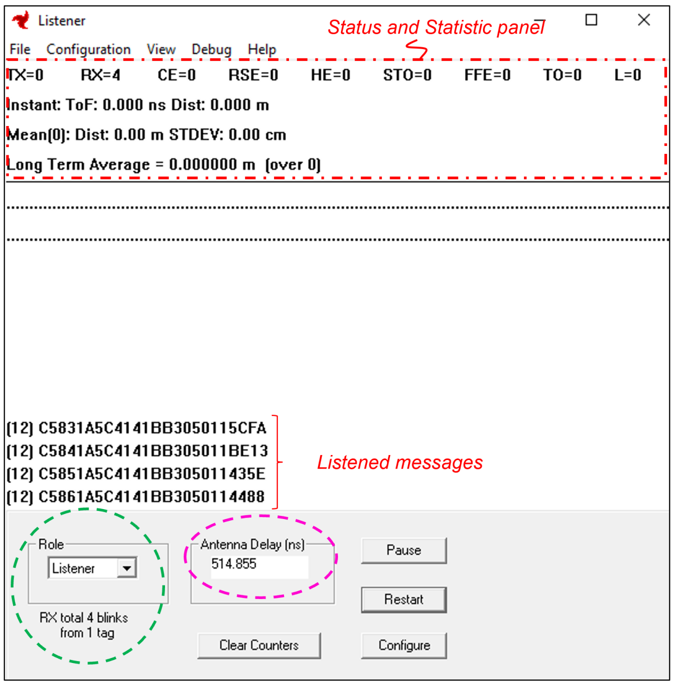
To calculate an estimate of the receive power level (in dBm), is possible to use:

$$RX\ Level \ = \ 10 \times \log_{10}(\frac{C \times 2^{17}}{N^2}) - A \tag{1.11}$$

where C is the Channel Impulse Response Power value which can be found as a 16-bit in the register file: 0x12 - RX Frame Quality Information, while A and N are the same constant of equation 1.10.

1.5.6 DW1000 APP

The EVK1000 board comes with the DecaRanging application, which is aimed to display in real time the CIR and others meaningful data. Additionally, from the application, is possible to log the data, in order to store them and perform post-processing or analysis. Since the basic DecaRanging application didn't store some variable of our interest, the code has been slightly changed, obtaining a modified DecaRanging app version.
First, the basic app will be evaluated in this section. Then, a comparison with the updated one will be shown.



```
TX=0    RX=4    CE=0    RSE=0    HE=0    STO=0    FFE=0    TO=0    L=0
Instant: ToF: 0.000 ns Dist: 0.000 m
Mean(0): Dist: 0.00 m STDEV: 0.00 cm
Long Term Average = 0.000000 m  (over 0)

(12) C5831A5C4141BB3050115CFA
(12) C5841A5C4141BB305011BE13
(12) C5851A5C4141BB305011435E
(12) C5861A5C4141BB3050114488
```

Antenna Delay (ns): 514.855

RX total 4 blinks from 1 tag

Figure 1.14: Main window of the DecaRanging app

Main DecaRangning Application window

To run the DecaRanging.exe executable the computer must be connected to the board. Once they are connected, the main display windows appear, as depicted in fig. 1.14. Initially, when the executable is launched, the software starts in listener mode, where it listens and reports all received message. To enable the ranging function the role must be reconfigured in order to switch one of the ends as a Tag and the other as an Anchor. This role selection can be done by the application using the drop-down list of the role group to the bottom left of the main DecaRanging window (fig. 1.14, green dashed circle) or manually (in case the other antenna is not working with the DR app) through the switches. Before switching one antenna as Anchor, it needs to spend some time in the initial Listener role to receive some blink messages from the tag. The number of blinks received is reported in text below the role selection windows.

When the ranging is running, the lines at the top of the main windows (fig. 1.14, red dashed box) give a status report of the measurements. The status box contains the following information:

- **TX:** is the number of frame transmitted.
- **RX:** is the number of frame received.

- **CE:** is the number of CRC errors.
- **RSE:** is a count of unrecoverable errors in the Reed Solomon decoder.
- **HE:** is count of uncorrectable errors detected in the PHY header SECDED code.
- **STO:** is count of SFD timeout events.
- **FFE:** is the number of frame filtering errors (when the destination address is incorrect).
- **TO:** is the number of times the Tag has timed out from receiving without getting a response frame. This results in another transmission attempt, increasing the number of TX signal.
- **L:** is a count of late TX enables and late RX enables. These can occur in the delayed transmission and reception. If this happens frequently then it probably means that the response time configuration needs to be increased.
- **Instant ToF:** is the measured Time of Flight, in nanosecond.
- **Dist:** is the conversion from ToF (in [s]) to distance in [m], by multiplying 299'702'547.0 [m/s], which is the speed of the light in air.
- **Mean (n) Dist:** is the average of the distance over (8) measures. Since in the figure no measurements have still be taken, it shown n = 0.
- **STDEV:** is the standard deviation of the calculated distance (of last 50 range values), in centimetres.
- **Long Term Average:** is the average distance over all the collected measurements, in meters. The number in brackets represent the number of measurements taken as average.

The main window (1.14) contains the controls button in the lower gray area. As already mentioned at the start of this section, it is possible to find the role button, which is used to switch the antenna as Listener, Tag or Anchor. On the right of the windows, 4 different buttons are present:

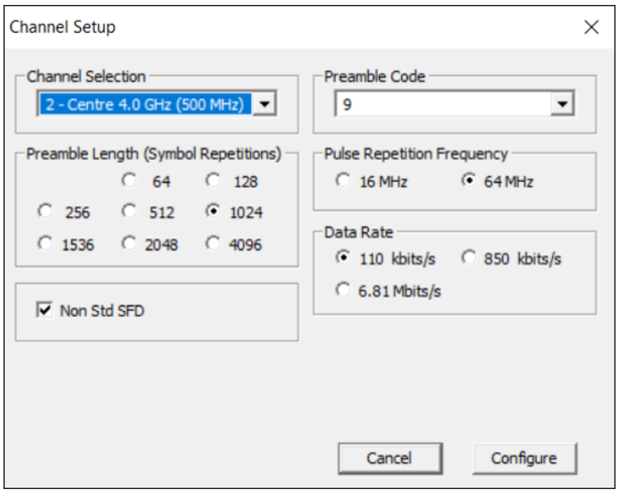
- **Pause:** it is used to pause the measurements collection and disable all activity in the lower layer application state machine, halting all sending and all receive processing.
- **Restart:** it is used to reset the software and start it again.
- **Configure:** it is used to access to the channel setup dialog box (fig 1.15).

- **Clear Counters:** it is used to clear the measurements averages.

Finally, in the center of the control area (fig. 1.14, purple dashed circle), is present the Antenna Delay section. This box is designed to change the delays to and from the antenna so that these may be subtracted from the round trip. Indeed, changing the antenna delay is a possible calibration method, where by trial and error is possible to find the right value for the respectively settings. The value specified here is used to correct the reported message sent and received times, half in the transmitter and half in the receiver, compensating for the system delays between physical timestamp and signal presence at the antenna.

Channel Setup Configuration Box

The Channel Setup dialog box allows to customize all the settings and parameters controlling the format of 802.15.4 messages. In tab. 1.3 have been explained all


Channel Setup

Channel Selection: 2 - Centre 4.0 GHz (500 MHz)

Preamble Code: 9

Preamble Length (Symbol Repetitions): 64 128 256 512 [✓ 1024] 1536 2048 4096

Pulse Repetition Frequency: 16 MHz [✓ 64 MHz]

Data Rate: [✓ 110 kbits/s] 850 kbits/s 6.81 Mbits/s

[✓ Non Std SFD]

Figure 1.15: Channel setup configuration dialog box

the parameters.

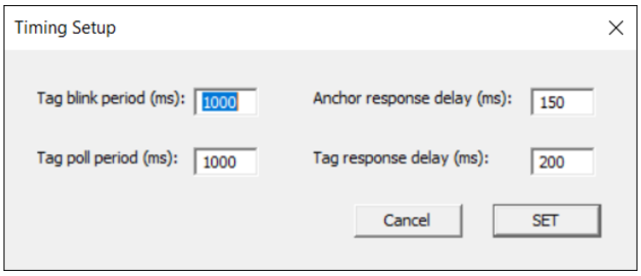
Timing Setup Configuration box

Before performing TWR measurements, certain parameters of the tag and anchor message interactions must be set. These parameters can be configured from the Timing Setup dialog box, which is represented in fig 1.16.

The configurable parameters are described in table 1.4.

Table 1.3: Channel Setup paramters [15]

Item	Quick description
Channel Selection	Contains the possibilities for the channels, with the information of center frequency and bandwidth.
Preamble Code	Defines the Complex Channel for the inter device communication. These codes depend on the channel selection.
Preamble Lenght	Preamble lengths of 64, 1024 and 4096 are defined in the IEEE 802.15.4 standard. The additional preamble lengths provided by the DW1000 allow designers more opportunity to optimise system performance trade-offs. Ideally, the same preamble length should be set at both ends to give a balanced communications range, set to be consistent with the data rate. For long range at the 110 kbps data rate, one would use a preamble of 1024 (or more) symbols, while at the 6.8 Mbps data rate a long preamble will bring no benefit and a shorter preamble would be used in practice to save power and air-time (increasing network capacity).
Pulse Repetition Frequency	This selects the Pulse Repetition Frequency to be used in the transmitter and receiver. NB: both units need the same PRF configuration setting.
Data Rate	The data rate may be selected to be any one of the data rates available here. NB: both units need the same data rate setting.
Non Std SFD	This tick box enables the use of a non-standard SFD. The SFD is the component of the IEEE 802.15.4 UWB frame marking the end of the preamble and the start of the data frame, which is actually a certain pattern of normal, inverted and deleted preamble symbols. Decawave has found an alternative SFD pattern that is more robust than the one defined in the IEEE 802.15.4 standard, giving an extra dB or so of performance.


Timing Setup

Tag blink period (ms): 1000

Anchor response delay (ms): 150

Tag poll period (ms): 1000

Tag response delay (ms): 200

Figure 1.16: Timing setup configuration dialog box

Table 1.4: Timing Setup paramters [15]

Item	Quick description
Tag blink period	This sets the time for which the tag waits between the blinks that are sent while the tag has not been paired with an anchor.
Tag poll period	This sets the time for which the tag waits between ranging attempts.
Anchor response delay	This sets the response delay used in anchor for sending the response to the tag poll message.
Tag response delay	This sets the response delay used in tag between receiving the anchor response and sending the final message.

Basic vs Updated log-file

The log-file is an automated file that starts to be created when the "log channel response" button is clicked. It stores all SPI traffic. In fig. 1.17 is depicted an example of log-file, where all the changes with respect to the standard version are highlights through red boxes. Indeed, the adjustments have only be performed in order to store all the values of our interest, for the specific project.
The file is divided in 4 main sections:

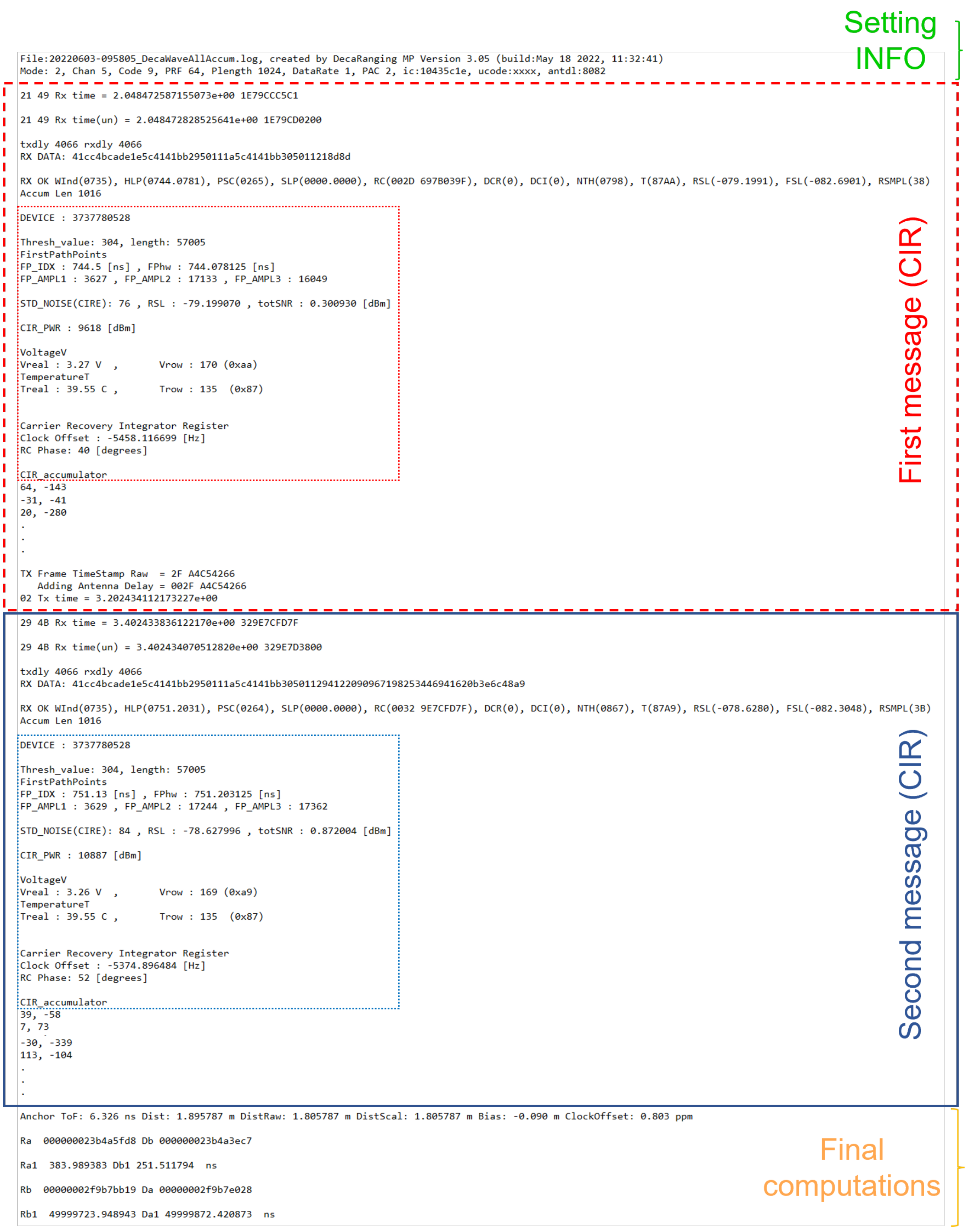
1. **Setting information:** in this first portion of the file is reported the name of the file (with time and day of the creation) and the channel setup configured variable.

2. **First received signal:** before sending the real main signal (the CIR), the board stores different information (like RX time, condition of the RX signal, FP_index, RSL, etc.). All these data could still be acquired from the standard version.

The updated one, instead, include all the information enclosed in the red-dotted box, that are: device id, threshold value, all the point relative to the first path, the CIRE, the signal's power, board temperature [°C], voltage consumption [V], clock offset [Hz] and recovery phase [°] (even if these latter two have not been successively used).
The core of the received message has been saved as a collection of real and imaginary number and are situated below "CIR accumulator". Its length is equal to the preamble length. The final information of this section is the TX time of the new signal.

3. **Second received signal:** since the working principle is two-way ranging, the antenna needs to exchange two signals in order to compute the ToF. For this reason this section is divided exactly as section 2 - First received signal.

4. **Final computations:** once the antenna have the information of the two messages, it could start to compute the ToF. Indeed, in the log-file it is reported every two exchanged signals. It is reported also the distance, in meter, which differ from ToF only by a constant (c: speed of light).



```
File:20220603-095805_DecaWaveAllAccum.log, created by DecaRanging MP Version 3.05 (build:May 18 2022, 11:32:41)
Mode: 2, Chan 5, Code 9, PRF 64, Plength 1024, DataRate 1, PAC 2, ic:10435c1e, ucode:xxxx, antdl:8082
21 49 Rx time = 2.048472587155073e+00 1E79CCC5C1

21 49 Rx time(un) = 2.048472828525641e+00 1E79CD0200

txdly 4066 rxdly 4066
RX DATA: 41cc4bcade1e5c4141bb2950111a5c4141bb305011218d8d

RX OK WInd(0735), HLP(0744.0781), PSC(0265), SLP(0000.0000), RC(002D 697B039F), DCR(0), DCI(0), NTH(0798), T(87AA), RSL(-079.1991), FSL(-082.6901), RSMPL(38)
Accum Len 1016
DEVICE : 3737780528

Thresh_value: 304, length: 57005
FirstPathPoints
FP_IDX : 744.5 [ns] , FPhw : 744.078125 [ns]
FP_AMPL1 : 3627 , FP_AMPL2 : 17133 , FP_AMPL3 : 16049

STD_NOISE(CIRE): 76 , RSL : -79.199070 , totSNR : 0.300930 [dBm]

CIR_PWR : 9618 [dBm]

VoltageV
Vreal : 3.27 V  ,       Vrow : 170 (0xaa)
TemperatureT
Treal : 39.55 C ,       Trow : 135  (0x87)


Carrier Recovery Integrator Register
Clock Offset : -5458.116699 [Hz]
RC Phase: 40 [degrees]

CIR_accumulator
64, -143
-31, -41
20, -280
.
.
.

TX Frame TimeStamp Raw  = 2F A4C54266
   Adding Antenna Delay = 002F A4C54266
02 Tx time = 3.202434112173227e+00
29 4B Rx time = 3.402433836122170e+00 329E7CFD7F

29 4B Rx time(un) = 3.402434070512820e+00 329E7D3800

txdly 4066 rxdly 4066
RX DATA: 41cc4bcade1e5c4141bb2950111a5c4141bb3050112941220909671982534469416 20b3e6c48a9

RX OK WInd(0735), HLP(0751.2031), PSC(0264), SLP(0000.0000), RC(0032 9E7CFD7F), DCR(0), DCI(0), NTH(0867), T(87A9), RSL(-078.6280), FSL(-082.3048), RSMPL(3B)
Accum Len 1016
DEVICE : 3737780528

Thresh_value: 304, length: 57005
FirstPathPoints
FP_IDX : 751.13 [ns] , FPhw : 751.203125 [ns]
FP_AMPL1 : 3629 , FP_AMPL2 : 17244 , FP_AMPL3 : 17362

STD_NOISE(CIRE): 84 , RSL : -78.627996 , totSNR : 0.872004 [dBm]

CIR_PWR : 10887 [dBm]

VoltageV
Vreal : 3.26 V  ,       Vrow : 169 (0xa9)
TemperatureT
Treal : 39.55 C ,       Trow : 135  (0x87)


Carrier Recovery Integrator Register
Clock Offset : -5374.896484 [Hz]
RC Phase: 52 [degrees]

CIR_accumulator
39, -58
7, 73
-30, -339
113, -104
.
.
.
Anchor ToF: 6.326 ns Dist: 1.895787 m DistRaw: 1.805787 m DistScal: 1.805787 m Bias: -0.090 m ClockOffset: 0.803 ppm

Ra  000000023b4a5fd8 Db 000000023b4a3ec7

Ra1  383.989383 Db1 251.511794  ns

Rb  00000002f9b7bb19 Da 00000002f9b7e028

Rb1  49999723.948943 Da1 49999872.420873  ns
```

Figure 1.17: Standard vs Updated version of the log channel response

Chapter 2

From Machine Learning to Deep Learning

Nowadays, intelligent system that offer Artificial Intelligent (AI) capabilities often rely on Machine Learning (ML). Machine learning refers to a system's ability to learn from problem-specific training data in order to automate the process of developing analytical models and solving associated tasks. ML is a subset of artificial intelligence, seeking to automatically lean meaningful relationships and pattern from large amount of data. The ability of such systems to solve complex problems is based on analytical models that create predictions, rules, responses, suggestions, or other comparable outputs. During the last decades, the field of ML has brought forth a variety of remarkable advancements in sophisticated learning algorithms and efficient pre-processing techniques. One of these developments was the evolution of Artificial Neural Networks (ANNs) into deeper and deeper neural network topologies with better learning capabilities, summarized as Deep Learning (DL). The hierarchical relationship between all of these terms is depicted in fig. 2.1.

In order to have a better conceptual distinction, AI includes any method that enables computers to replicate or outperform human decision-making to solve complicated problems on their own or with minimal human intervention. Machine learning overcomes such limitations. Indeed, ML means that a computer program's performance improves with experience with respect to some class of tasks and performance measures. As such, it aims at automating the task of analytical model building to perform cognitive tasks like object detection or natural language translation. This is achieved by applying algorithms that iteratively learn from problem-specific training data, which allows computers to find hidden insights and complex patterns without explicitly being programmed [16]. Thanks to the ability

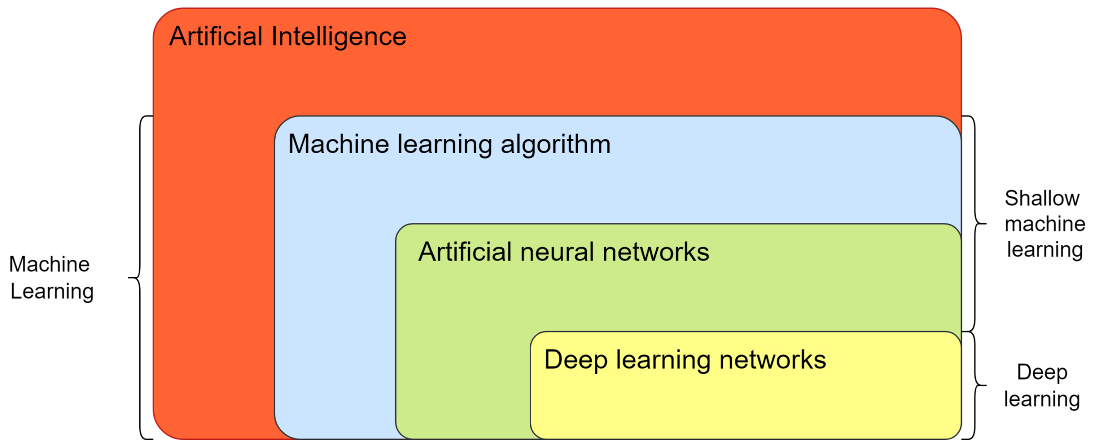


Figure 2.1: Artificial Intelligence classification

of learning from previous computations and extracting regularities from massive databases, it produces, if well designed, reliable and repeatable decisions. For this reason, ML algorithms have been successfully applied in many areas, such as fraud detection, credit scoring, next-best offer analysis, speech and image recognition, or Natural Language Processing (NLP).
Based on the available information, we can defines three types of machine learning:

- **Supervised learning:** it is defined by its use of labeled datasets to train the algorithm for classification. This means that it is known a priori how the training set is categorized. This training dataset includes inputs and correct outputs, which allow the model to learn over time. The algorithm measures its accuracy through the loss function, adjusting weights until the error has been sufficiently minimized.

- **Unsupervised learning:** this learning takes place when the learning system is supposed to detect patterns without any pre-existing labels or specifications. In this case, training data only consist of variables "x", with the goal of finding shared common properties (known as clustering) or data representations that are projected from a high-dimensional space into a lower one (known as dimensionality reduction).

- **Reinforcement learning:** here, instead of providing input and output pairs, we describe the current state of the system, specify a goal, provide a list of allowable actions and their environmental constraints for their outcomes, and let the ML model experience the process of achieving the goal by itself using the principle of trial and error to maximize a reward [16].

Inspired by the principle of the biological neural networks, that constitute animal brain, artificial neural networks, instead, are a mathematical representations of collections of connected units or nodes, called artificial neurons. Like synapses in a brain, each connection between neurons transmits signals whose strength can be amplified or attenuated by a weight that is continuously adjusted during the learning process. ANN could be differentiated by layers. Typically is possible to split them in 3: input layer, which receives the data input (e.g., images), output layer, which returns the final result (e.g., classification), and hidden layers, that are effectively responsible for learning a non-linear mapping between input and output (fig. 2.2).

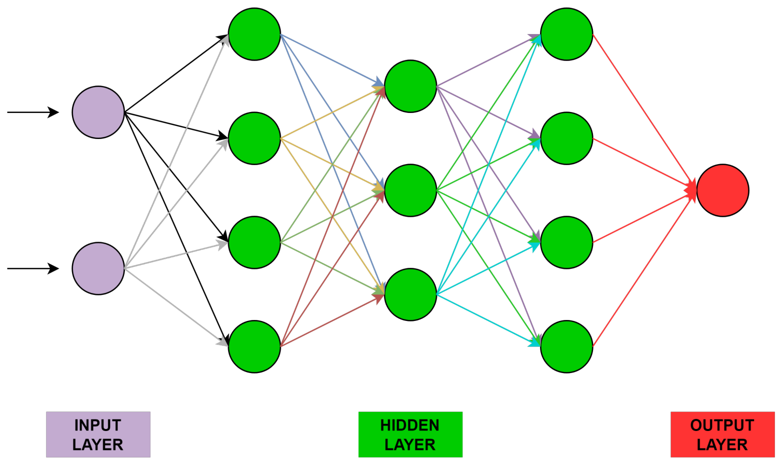


Figure 2.2: Example of an artificial neural network scheme

The output of the neuron is typically the weighted sum of all the inputs plus a bias term. The weighted sum is then passed through a (usually nonlinear) activation function to produce the output. The number of layers and neurons, as well as other internal parameters such as learning rate or activation function, cannot be learned by the algorithm. This means that they constitute a model's hyperparameters and must be set manually or determined by optimization procedures.

Deep neural networks is formed by ANN with typically more than one hidden layer, organized in deeply nested network architectures. The difference with the classical ANN is that they contains advanced neurons, besides that they perform also more complicated operation like convolutions or recurrent connections. These features allow deep neural networks to extract automatically the main information from the input, enabling the possibility to feed them with raw input data. DL is particularly useful when working with large and high dimensional data (e.g., most applications in which text, audio, image and video needs to be processed).

2.1 Activation function

The activation function take an important role in ML. Indeed, if the activation function is not used in a neural network, the result is a simple linear function which is a one degree polynomial. In other words, neural networks without an activation function acts as a Linear Regression model with limited performances.
Since almost every problem in real life can not be represented by linear functions, it is important to have non-linear functions. The usefulness of these functions is that they have to learn, represents and process any data and any arbitrary complex problem which maps the input to the output. An important feature of an activation function is that it must be differentiable so that backpropagation could be implemented [17].

To enable a limited amplitude of the output of a neuron different functions could be used [18]:

1. **Binary Step Function**
2. **Linear**
3. **Sigmoid**
4. **Tanh**
5. **ReLU**
6. **Leaky ReLU**
7. **SoftMax**

There are a lot of parameters to increase performances and decrease erroneous results, like number of hidden layers, training methods, hyperparameter tuning and activation function. The former represent one of the most important parameters to consider, even if choosing the right one may be a tedious process and may require a lot of research and studies. From studies has been discovered that both sigmoid and tanh functions are not suitable for hidden layers because the slope of the fucntion become very small as the input becomes very large or very small, leading to to slow down gradient. ReLu is the most preferred choice for applying in hidden layers as the derivative of it is 1

2.2 Underfitting and overfitting

The purpose of a machine learning model is to approximate an unknown function that map input elements to outputs one. Since the analytic expression of the function is unknown, when training, it is necessary to think about fitting the model but keeping it free to generalize when an unknown input is presented. This simple idea becomes to a very hard condition to fulfil, leading to two possible errors:

- **Underfitting:** when a model is underfitting the data, means that the selected function is not able to capture the dynamics shown by the same training set, and the same time, it is not able to model new data. Typically, underfitting is easy to detect given a good performance metric. The remedy is to move on and try alternate machine learning algorithms.

- **Overfitting:** it happens when the model has an excessive capacity in adapting to the details and noise of the training dataset, loosing the generalization ability. In other words, it can associate almost perfectly all the known samples to the corresponding output values, but when a new dataset is given as input, the prediction error become very high. Overfitting is more likely with non-parametric and non-linear models that have more flexibility when learning a target function. As such, many non-parametric machine learning algorithms also include parameters or techniques to limit and constrain how much detail the model learns.

In fig. 2.3 are depicted interpolations with low-capacity (underfitting), normal-capacity (correct fit), and excessive capacity (overfitting).

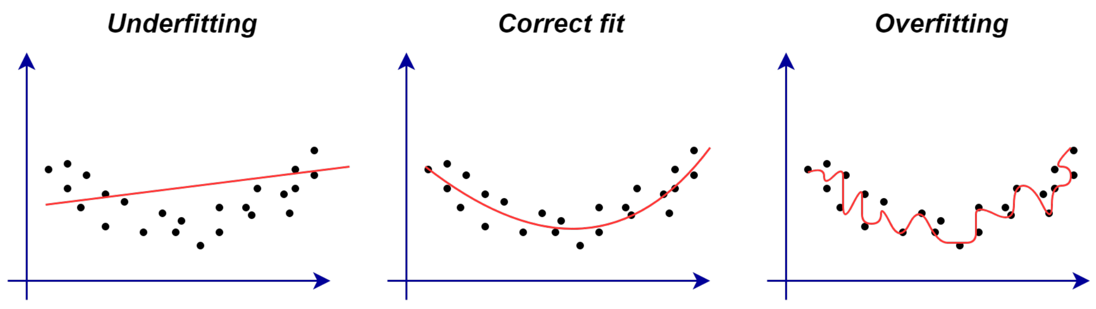


Figure 2.3: Underfitting vs Overfitting

In all the possible situations, it is very important to avoid both underfitting and overfitting. A generic rule of thumb is to check the residual error, since it must be always present to guarantee a good generalization. Indeed, it is very likely that models having validation accuracy similar at 99.999 percent are overfitting the dataset, that means they will not be able to predict correctly when never-seen input samples will be provided.

2.3 Validation metrics

One of the most important steps after developing a model is to evaluate its training and predictive performance. Different metrics exist and the use of one with respect to other depends on many factors. The main factor that drastically change the use of metrics is the final result of the ML algorithm, meant as if we are classifying or predicting. Since in this thesis project the final goal is classification, the analysis of predicting metrics has not been carried on.

It is important to highlight that all the validations techniques must be performed on a new dataset, different from the training one.

2.3.1 Confusion matrix

The confusion matrix is a N×N matrix, where N is the number of classes being predicted. The matrix gives an immediately visualization of the performance of a specific algorithm (fig. 2.4). On each row is specified the number of predicted class for each class, while on the columns is represented the number of actual true class.

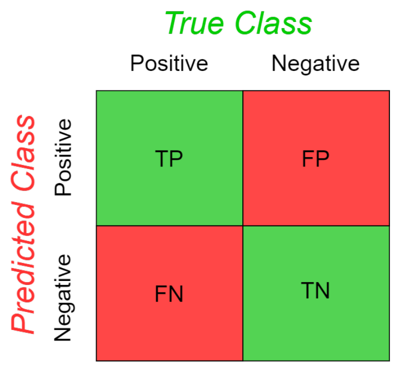


Figure 2.4: Confusion matrix for binary classification

where TP: True Positive, FP: False Positive, FN: False Negative and TN: True Negative. The goal is to correctly predict all the elements, which means get almost 0 value off the diagonal.

It is also possible to find confusion matrix in which Predicted and True class are swapped.

2.3.2 F1 score

The F1-score, also called the F-score, is a measure of a model's accuracy on a dataset. Contrary to the confusion matrix which can have multi classes, this metrics could be used only to evaluate binary classification systems, which classify examples into ‘positive’ or ‘negative’. The F1-score is a way of combining the precision and recall of the model, and it is defined as the harmonic mean of the model’s precision and recall. The formula for F1-score is as follow:

$$F1 = \frac{TP}{TP + \frac{1}{2}(FP + FN)} = \frac{2}{\frac{1}{recall} \times \frac{1}{precision}} = 2 \times \frac{precision \times recall}{precision + recall} \tag{2.1}$$

where:

- **Precision:** $\frac{TP}{TP + FP}$, is the fraction of true examples among all the examples classified as positive.
- **Recall:** $\frac{TP}{TP + FN}$, is the fraction of examples classified as positive among the total number of true class positive examples.

2.3.3 AUC-ROC

The receiver operating characteristics (ROC) curve is one of the most important and widely used performance metrics for the evaluation of classification models in terms of their goodness-of-fit. This method is a probability-based curve that can measure models at different thresholds. The threshold is the specific value with which the probability of belonging to a class is compared. Based on the comparison, the element is then classified as positive or negative. The curve plots the True Positive Rate vs the False Positive Rate, which are defines as follow:

- **True Positive Rate (TPR):** $TPR = \frac{TP}{TP + FN}$, which is a synonym of recall.
- **False Positive Rate (FPR):** $FPR = \frac{FP}{FP + TN}$

To build the ROC, the classifier must be run every time with different threshold. For example, in fig. 2.5, is possible to see that when the threshold is equal to 1 (impossible to overcome since the probability p is always bounded between 0 and 1) no classifications are performed. Contrary, with threshold equal to 0, the classification assign the element always to the same class. A perfect algorithm should have TPR closer as possible to 1, with low FPR.

The Area Under Curve (AUC), instead, represents the degree or measure of separability, in other words, it tells how much the model is capable of distinguishing between classes. It measures the entire two-dimensional area underneath the entire ROC curve (by integrating from (0,0) to (1,1)). AUC ranges in value from 0 to

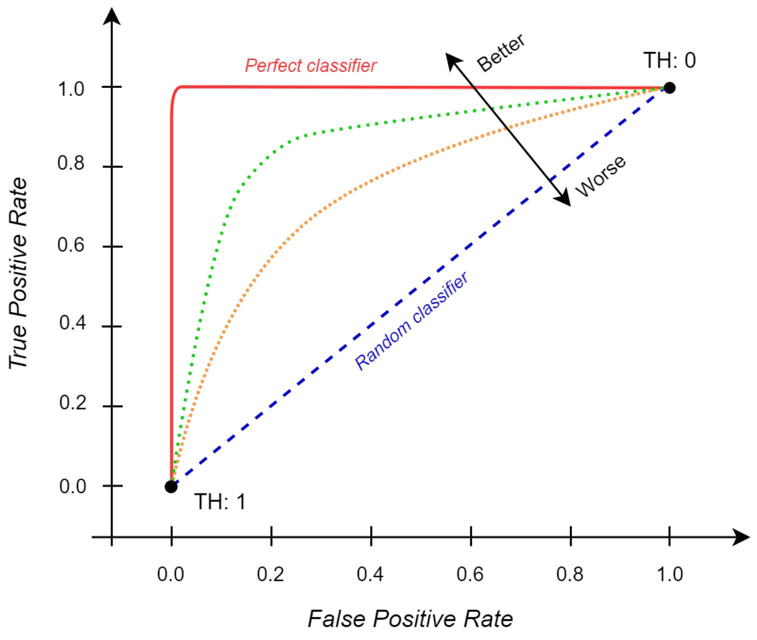


Figure 2.5: ROC Curves

1. A model whose prediction are 100% correct has an AUC of 1, while if are all wrong, it has an AUC of 0. The usefulness to use the AUC with respect to test the model with different threshold are:

- AUC is scale-invariant.
- AUC is a classification-threshold-invariant. It measures the quality of the model's predictions irrespective of what classification threshold has been chosen

However, these two characteristic are not always desired.

2.3.4 Other statistical metrics

The statistical metrics used for machine learning are useful in order to have a first quick idea on how the model is performing the classification. In the main used is present the threshold metric, which includes:

$$\textbf{Specificity} = \textbf{Recall} = \textbf{TPR:} \quad defined\ in\ sec.\ 2.3.2\ and\ 2.3.3$$

$$\begin{aligned} \textbf{Sensitivity:} \quad SST &= \frac{FP}{TN+FP} \\ \textbf{Accuracy:} \quad ACC &= \frac{TP+TN}{TP+TN+FP+FN} \end{aligned} \tag{2.2}$$

Using these performances metrics, is possible to investigate how well the model makes decisions when tested with unseen dataset (i.e., generalization ability).

2.4 Machine learning algorithms

Machine learning algorithm are all of those which takes decisions and does predictions or forecasting based on data and training operations. An example could be an algorithm able to forecast when a patient has cancer based on different feature (e.g., blood pressure, age, blood glucose values, etc.), or, simpler, when a mail is SPAM or not, looking, for example, at specific words.
Furthermore, algorithms that are designed for binary classification could be adapted to use with multi-class problems. This involves using a strategy of fitting multiple binary classification models for each class vs. all other classes (called one-vs-rest) or one model for each pair of classes (called one-vs-one).

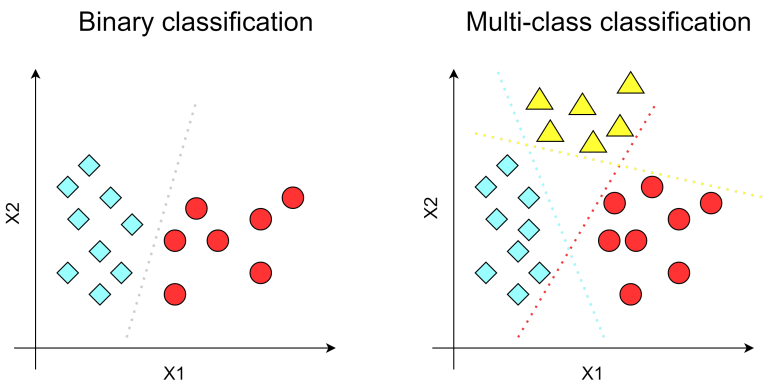


Figure 2.6: Binary and multi-class classification

2.4.1 Logistic regression

Logistic Regression (LR) is a linear and supervised model, used for classification's problem. It means that the space could be divided with line (2-D), hyperplane (3-D), etc. It is usually used for binary classification, but it can be easily adapted for multi-class classification.
Using the sigmoid as activation function, it return the probability of the item to belong to each class. The item will be then assigned to the class with higher probability.

Since the output of the logistic regression is a probability score, the threshold over which the classification take part is a very important variable. For example,

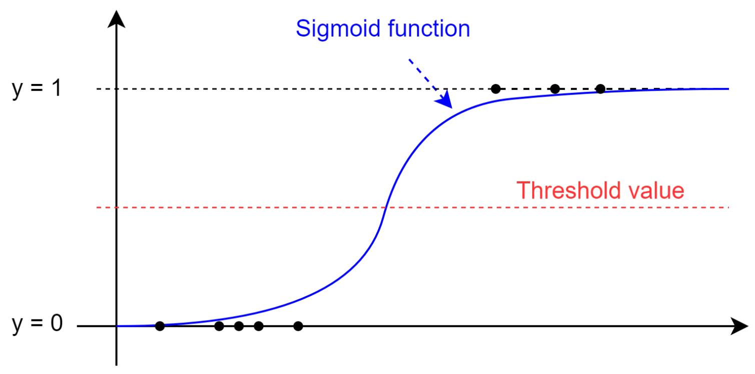


Figure 2.7: Logistic regression for binary classification

Table 2.1: Advantages and disadvantages using Logistic Regression

Advantages	**Disadvantages**
Simplicity of implementation	Inability to solve non-linear problem as its decision surface is linear
Computational efficiency	Prone to overfitting
Ease of regularization	
Input features scaling not required	

looking at figure 2.7, it has been used a threshold of 0.5, which means that if the probability $p \geq 0.5$ we get $y = 1$, $y = 0$ otherwise. The classification changes if a threshold value of 0.2 would be used.

2.4.2 Decision Tree

Decision Tree (DT), is a supervised ML approach to solve classification and regression problem, by continuously splitting data based on a certain parameter. In the classification the outcome is categorical (e.g., Yes or No), while in regression tree the decision variable is continuous.
It is possible to define:

- **Root Node:** it is the main node, representing the attribute with largest Information Gain.
- **Internal Node:** it is the decision node inside the main decision tree, representing other attributes values, with smaller IG value moving from the top to the bottom.

- **Leaf Node:** it represent the final class.

Different method exist on the selection of the splitting node. The most used one is the Information Gain function, known as "reduction in entropy", which try to order the nodes based on the impactful they have, that means choosing as the root node the feature which split most the data.

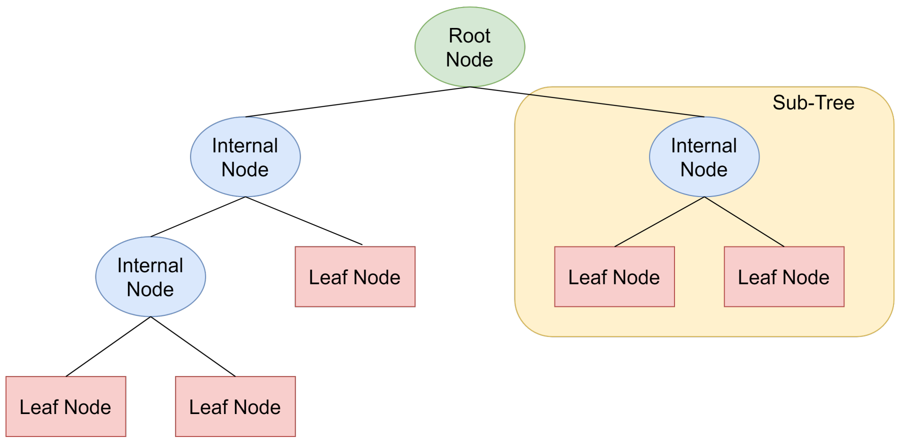


Figure 2.8: Components of a Decision Tree

Table 2.2: Advantages and disadvantages using Decision Tree

Advantages	**Disadvantages**
Simple to understand and interpret	Unstable, meaning that a small change in the data can lead to a large change in the structure of the optimal decision tree
High performance due to efficiency of tree traversal algorithm	Inaccurate
Ease of handling categorical and quantitative values	Calculations can get very complex

Moreover, Decision Tree might encounter the problem of over-fitting for which Random Forest is the solution, that is based on ensemble modelling approach.

2.4.3 Random forest

Random Forest is an ensemble method in ML. It is tree-based algorithm which leverages the outcome of multiple decision tree, to get a final decision. As the name suggest, the multiple decision trees are randomly created (via bootstrap aggregation, which only use a subset of variables when deciding how to split each node). Each tree predicts a classification independently and “votes” for the corresponding class. The majority of the votes decides the overall random forest prediction. The aggregate votes of several decision trees reduce the dependence to outliers with respect of using a single tree. Combination of bootstrap and random forest lead to the so called "bagging" techniques. One way to validate the model is to use the permutation test, which consist by measuring the increase of error if the variable under question are permuted across out-of-bag observations. This score is computed for each constituent tree, averaged across the entire ensemble and divided by the standard deviation.

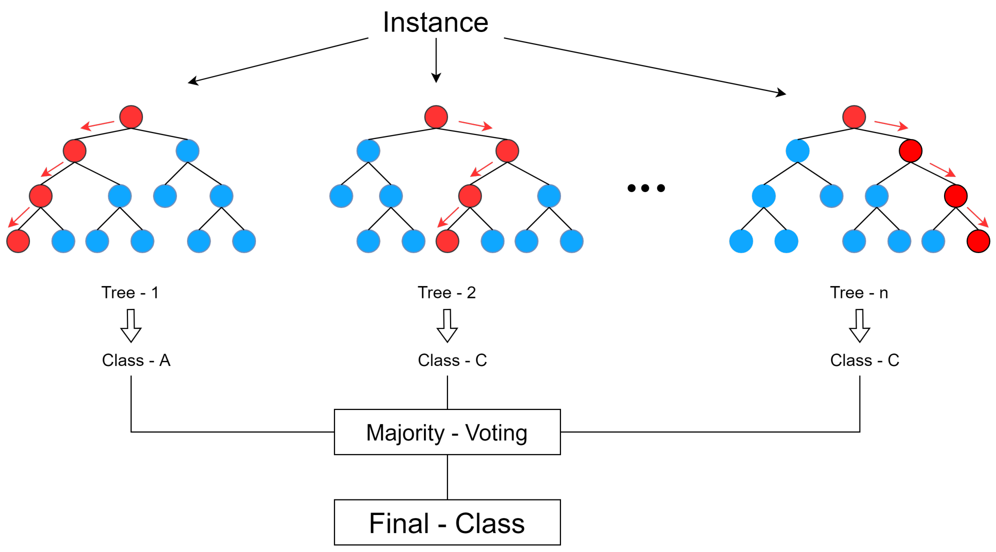


Figure 2.9: Random Forest scheme

2.4.4 Boosting

Boosting is an ensemble learning technique that uses a set of machine learning algorithm in order to convert, or combine, weak learners to strong learners, trying to increase the accuracy of the model. The main difference with the Bagging, which is an ensemble method as well, is that with the former we use weak learners over different dataset (parallel method), while in boosting we try to combine them (sequential method).

When using boosting, the entire dataset is fed to the algorithm, obtaining as a result the classification of all the item. With this technique more attention is focused on the miss classified data-points, increasing it’s weight. These steps will

Table 2.3: Advantages and disadvantages using Random Forest

Advantages	Disadvantages
Tend not to overfit. The processes of randomizing the data and variables across many trees means that no single tree sees all the data	A forest is less interpretable than a single decision tree
Accuracy calculated from out-of-bag samples is a proxy for using a separate test data set. The out-of-bag samples are those not used for training a specific tree and as such can be used as an unbiased measure of performance	A trained forest may require significant memory for storage, due to the need for retaining the information from several hundred individual trees
Works well "out of the box" without tuning any parameters. Other models may have settings that require significant experimentation to find the best values	

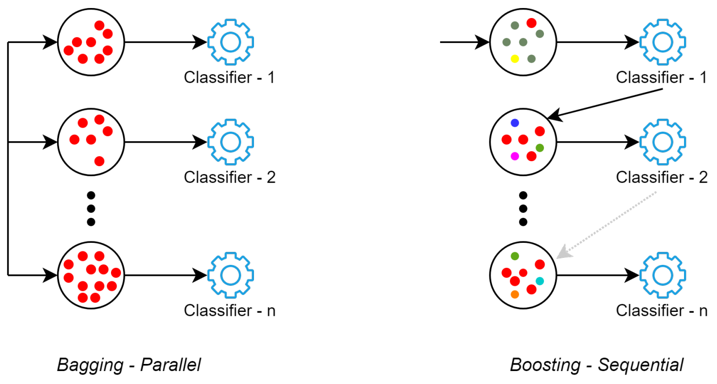


Figure 2.10: Bagging vs Boosting technique

be continuously repeated until all the wrong predicted samples will be correctly classified.

Let's understand better the followed steps in Boosting. Consider the example in figure 2.11, we have 2 classes, square and circle. At the beginning the algorithm assign at each data the same weight, and after a first analysis, it try to draw a decision stump (which is a single level decision tree). Successively, it checks for all the miss-classified data-points and assign them an higher weight (greater size in

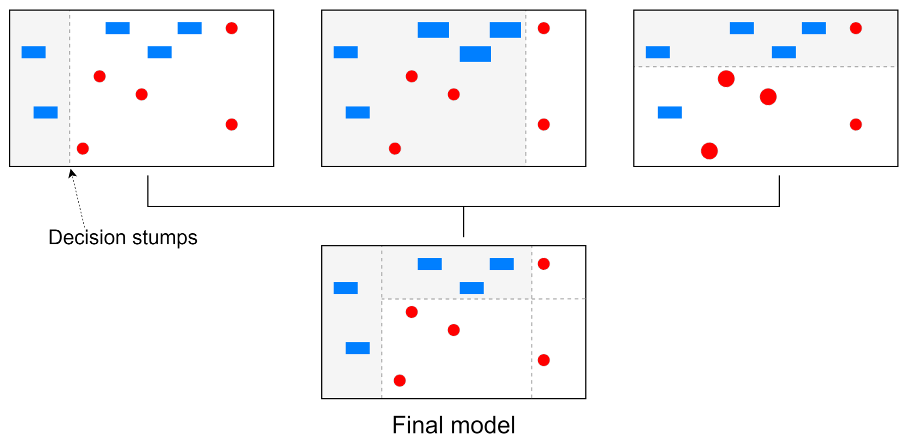


Figure 2.11: Adaptive boosting technique

the figure), in order to let classify them correctly in the next iterations. These 2 steps are repeated until all the elements are correctly classified.

There exist 3 main Boosting techniques:

- **Adaptive Boosting:** it is the technique used in the previous example, where it combine several weak learners into a strong learner.
- **Gradient Boosting:** in gradient boosting, base learners are generated sequentially in such a way that the present base learner is always more effective then the previous one (through loss function optimization).
- **XGBoost:** it is an advance version of the Gradient boosting method that is designed to focus on computational speed and model efficiency.

Table 2.4: Advantages and disadvantages using Boosting

Advantages	**Disadvantages**
Easy to read and interpret algorithm	Sensitive to outliers
Efficient prediction capability	Method impossible to scale up, since every estimator bases its correctness on the previous predictor
Less probability to encounter overfitting	

2.4.5 K-nearest neighbor

K-Nearest Neighbor (KNN) is frequently used for solving clustering problem. The KNN algorithm is positioned under the supervised type learning technique and is considered one of the easiest-to-use algorithms in machine learning, since it does not need any training data points for model generation. In KNN, K is the

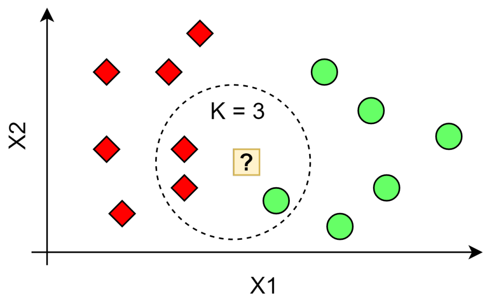


Figure 2.12: K-nearest neighbor scheme

number of nearest neighbors, and it is the core deciding factor. Indeed, the whole classification is based on the K value. When K = 1 (corresponding to the nearest algorithm) the new element is assigned to the same class of the closest data-point. In general, first is needed to find the K closest points to the new element, and then classify it by majority vote of that K neighbors.

Table 2.5: Advantages and disadvantages using KNN

Advantages	**Disadvantages**
Simple to implement	Expensive classification for unknown records
It is extremely flexible classification scheme and well suited for multi-modal classes	Distance computation of k nearest neighbors
Less probability to encounter overfitting	It does not work well when dataset is noisy
	Computationally intense with the growth of the training set size

2.4.6 Support Vector Machine

Contrary to the other methodologies, Support Vector Machine (SVM) needs the definition of the decision boundary, which is an hyperplane (element that separe different classes). In case the classes are not linearly separable, complex mathematical function called kernels are needed to map the elements in a different dimension which can be again easily separable. In fig. 2.13 are depicted 2 linearly separable classes with a maximum-margin hyperplane.

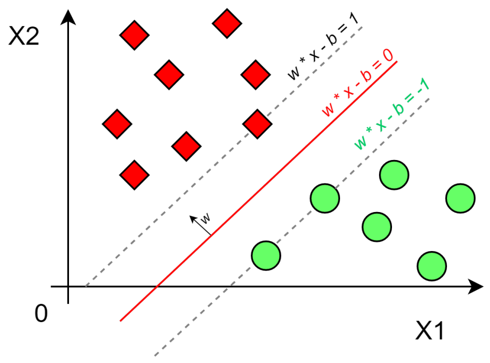


Figure 2.13: Maximum-margin hyperplane and margins for SVM

Table 2.6: Advantages and disadvantages using SVM

Advantages	Disadvantages
It handle both semi structured and structured data	Its performance goes down with large dataset due to the increase in the training time
It handle complex function if the appropriate kernel function can be derived	It will be difficult to find appropriate kernel function
Less probability to encounter overfitting	It does not work well when dataset is noisy
It can scale up with high dimensional data	It does not provide probability estimates
It does not get stuck in local optima	Understanding the final SVM model is difficult

2.4.7 Perceptron

Perceptron, or Fully Connected Layer, is the oldest neural network. It is a feedforward neural network in which the connections between nodes do not form loops. It accept multiple inputs, where each of them is then multiplied by different weights and all of them sum up. The role of the weight is to simulate synapses in biological neurons (to enhance or inhibit a signal). For every layer a bias term is added to the result, before the summation, result is passed to an activation function. A perceptron network has been already shown, indeed it corresponds to fig. 2.2.
This has been the basic for the development of Deep Neural Network, which include more complex interconnection.

2.5 Deep learning algorithms

Deep learning algorithms have developed in order to solve the problem of large dataset. Indeed, once the size of the data increases, it becomes challenging for traditional ML approaches to solve it, mostly due to the manual feature extraction. Deep Neural Network methods guarantee an opportunity to develop a more robust model to perform well on both small and large datasets. The most common networks are CNN, RNN, ResNet, Inception Time and LSTM. A quick analysis is carried out in the next sections.

2.5.1 Recurrent neural network

RNN is a popular algorithm used mostly for NLP and speech processing. Unlike traditional neural networks, RNN utilizes the sequential information in the network. In other words, this new topology of network insert for the first time an idea of memory, in which give importance also at the sequence of the input. For example, to understand a word in a sentence, it is necessary to know the context rather than the singular word. Therefore, an RNN can be seen as short-term memory unit which analyse the single word, remembering the context based on the previous inputs.
One main issue of an RNN is its sensitivity to the vanishing and exploding gradients [19]. In other words, since the algorithm involves the multiplications of small or big derivatives during the training, the gradient might decay (become too small to be used) or explode exponentially. This sensitivity reduces over time, meaning that the networks tends to forget the initial inputs with the entrance of the new ones. For this problem, Long Short-Term Memory (LSTM) has been developed by providing an additional block used to memorize specific elements in its recurrent connections.

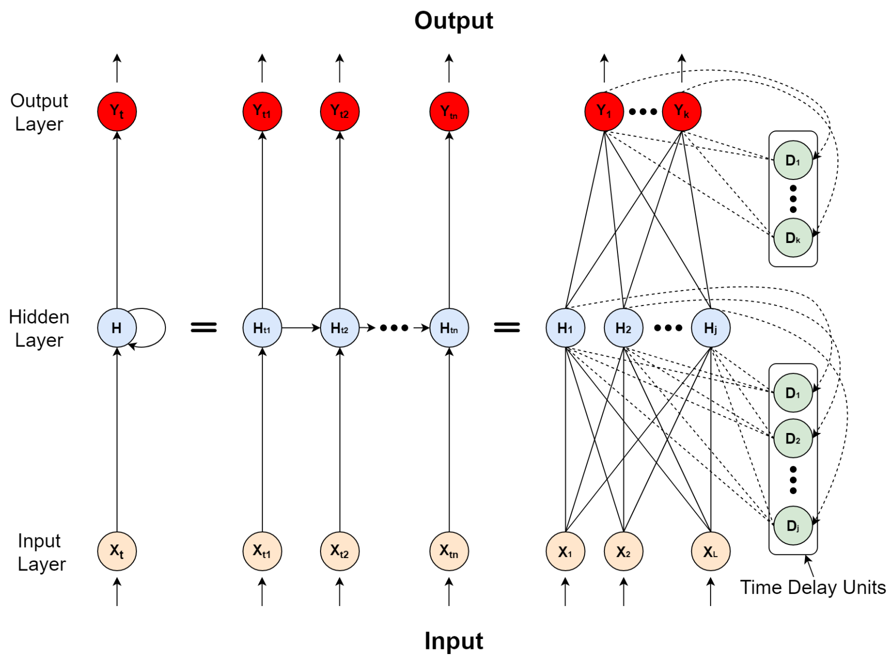


Figure 2.14: Recurrent neural network in: compact form (left), expanded in time (center) and completely expanded (right)

2.5.2 Convolution neural network

The main beneficial aspect of CNNs is reducing the number of parameters in ANN. For this reason it has been widely used in a variety of field related to pattern recognition, NLP, image processing, voice recognition, and so forth.
The main structure is based on series of convolution and-sampling layers, followed by a FNN and a normalization layer (e.g., softmax function) layer. The series of multiple convolution layer perform progressively more refined feature extraction at every layer moving from input to output layers. Sub-sampling or pooling layers are often inserted between each convolution layer to reduce the dimensionality of the problem. Different from the previous networks, CNNs takes a 2D n $\times$ n input, and each layer consist of group of 2D neurons called filters of kernels. Since the use of CNN will be for signal analysis (which are time series - 1D), the final network result is depicted in figure 2.15. For every layer, we have different depth, which corresponds to the number of applied filters.

Unlike other networks, neurons are not connected to all neurons in adjacent layer, reducing the number of parameters [20]. These factors speed up the learning and reduces the memory requirements for the network. The final classification is then performed only by the last layer, where neurons between the layers are fully connected. The main algorithm for parameter training is usually backpropagation,

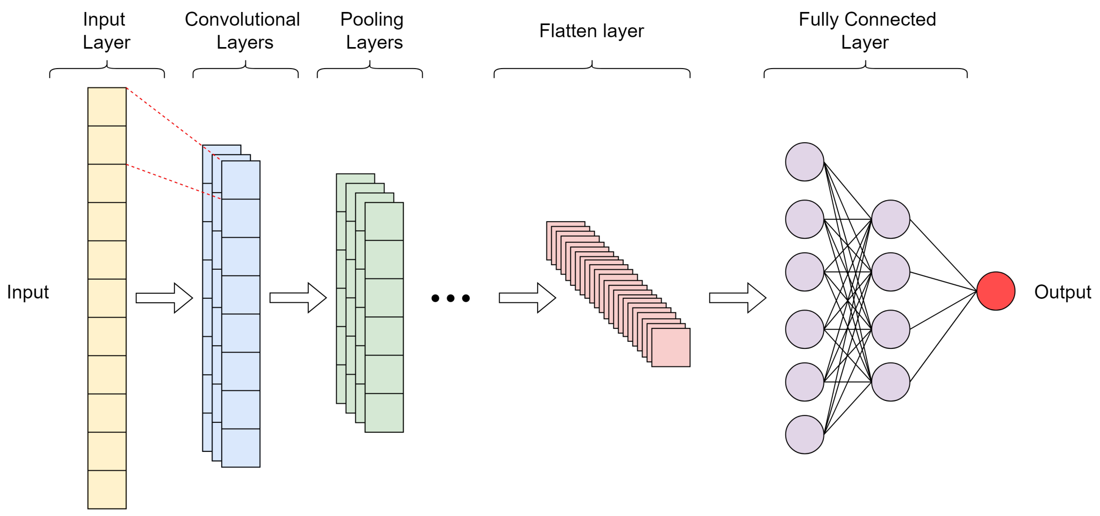


Figure 2.15: 1D CNN architecture

even if it will not be covered in this thesis.

2.5.3 Residual neural network

The modular unit of the generalized residual network architecture is a combination of CNN blocks, direct connection and skip connection (or shortcuts). Typical ResNet models are implemented with double or triple skips containing non linearity (activation function like ReLu) and batch normalization in between.

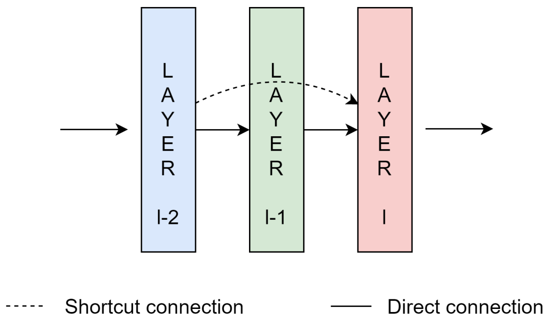


Figure 2.16: ResNet architecture

There are two main reasons to skip some connections:

1. to avoid vanishing gradients problem
2. to mitigate the degradation (or accuracy saturation) problem, since typically, adding layers to a suitable deep model leads to increase the training error.

2.5.4 Long short term memory

LSTM is an implementation of the Recurrent Neural Network aiming to solve the problem of vanishing gradients by letting gradients to pass unaltered. Unlike the earlier described feed forward network architecture, LSTM has the ability to retain knowledge of earlier states. As depicted in figure 2.17, LSTM consist of blocks of memory cell state, input cell, output gate and forget gate.

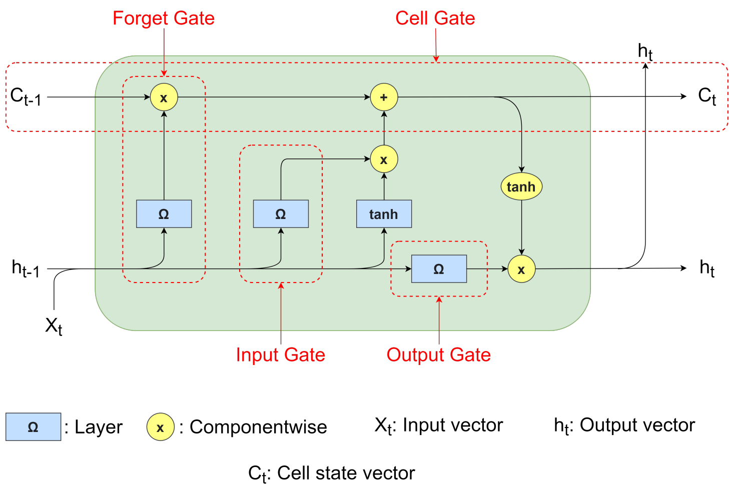


Figure 2.17: LSTM block unit

The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell. LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series.

2.5.5 Inception time

Inception was first proposed for end-to-end image classification. Now the network has evolved and it is able to work also with time series classification. The first invented module was simpler than the one shown in fig. 2.18. It was initially proposed without the dash boxes. The embedded of these three new boxes is related on dimensionality reduction. Indeed, since they are simple 1x1 convolutions layer, their aim is to reduce the space. For example, RGB images are store in matrices m-by-n-by-3 data array that defines red, green, and blue color components for each individual pixel. Applying a 1x1 convolution layer, we reduce the matrix to n-by-m-by-1, reducing the number of parameter needed in the next filter by a factor of 3.

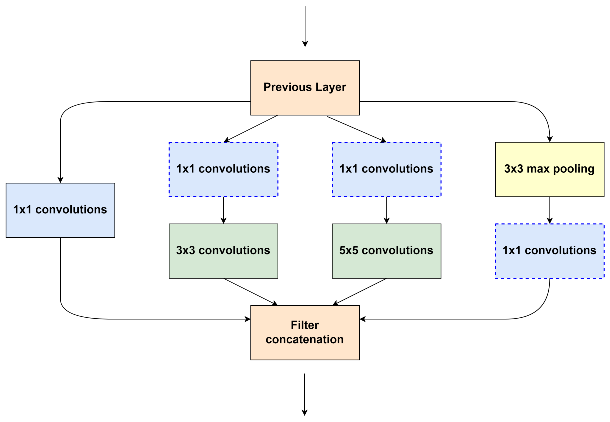


Figure 2.18: Inception time module with dimension reduction

The final version contains 3 dimensionality reduction blocks, 3 convolutions layer with different filter size (in order to extract different features from the same layer) and one max pooling filter.

Chapter 3

Measurements and logged data

Now, after that all the theoretical aspects have been covered, let's move to the beginning of the project. The first step has been analyse the environment over which the UWB localization would be performed. The selected area has been the third floor of the Engineering building, starting from the elevator and going up half of the hallway. It has been decided to divide the whole area in small rectangles of 100cm $\times$ 80cm (as depicted in fig. 3.1), getting a trade-off between measurements time-consuming and grid density.

The two boards had been configured and located in different locations:

- **Tag:** the board used as a Tag has been fixed to the showcase, and it has been charged by direct power supply. It is highlights in fig. 3.1 from the red box.
- **Anchor:** the antenna that plays the role of Anchor, instead, has been connected to a movable cart in order to easily reach any possible point in the selected environment. This antenna has been controlled by USB interface, over which the modified version of the DecaRanging application can operate as a simple USB to SPI controller.

Finally, the two antennas had been setup following the parameters present in tab. 3.1.

Table 3.1: Boards' setup

Channel	**Preamble C.**	**Preamble L.**	**PRF**	**Data Rate**	**Non Std SFD**
5	9	1024	64	110 kbits/s	Checked

Before starting to log all the measurements, the EVK1000 needs to be calibrated based on the selected frequency (channel).

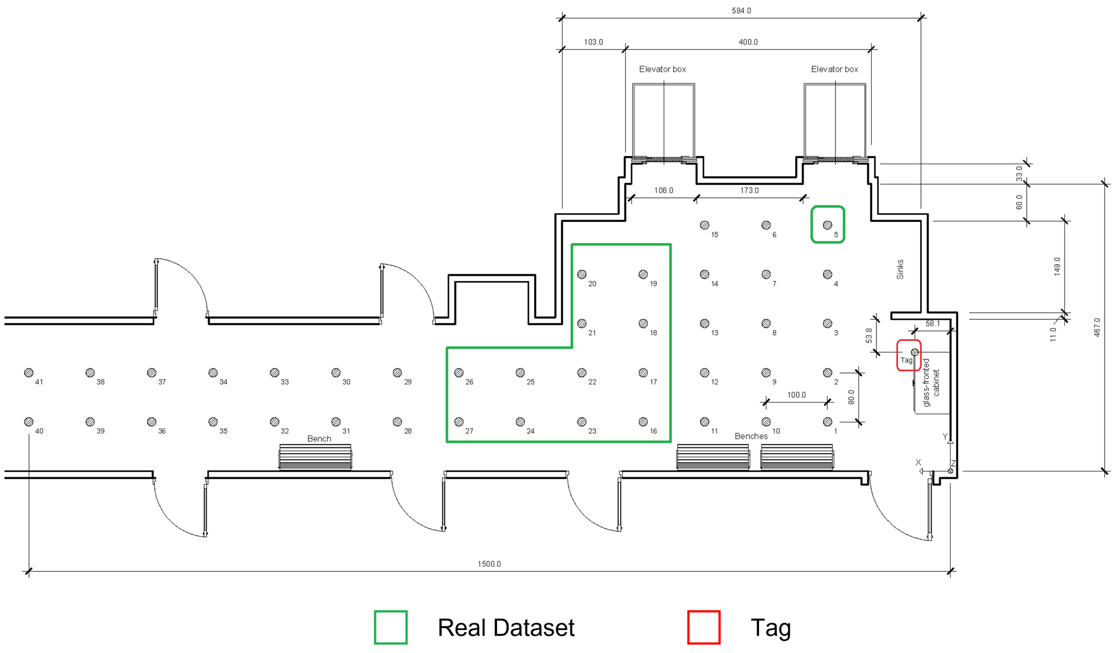


Figure 3.1: 2D model of the analysed environment

3.1 Calibration

The operating characteristics and performance of the DW1000 is dependent on the IC itself, on its external circuitry and on its operating environment. To give optimum performance it is necessary to calibrate the IC to account for factors which affect its operation [13]. Some calibration parameters may vary according to the operational environment of the DW1000. This include having a large variation in the ambient temperature (e.g., moving from a warm into a cold area) and/or a drastic changes in battery voltage supply.

There are mostly 3 elements of the DW1000 that may be subject to calibration:

- **Crystal trimming:** the DW1000 contains trimming capacitors that can fine tune the operating frequency of its crystal oscillator
- **Transmitted output power and spectrum:** the output spectrum of the DW1000 may be tuned to suit regional spectral standards and maximize output power to obtain the maximum operational range.

- **Antenna delay:** the DW1000 antenna delay may be fine-tuned to give best possible ranging or location

When the boards are brand-new, they already performed a calibration cycle by the company before being sold. For this reason, what has been done before starting the measurement process was only tune the two antennas through antenna delay, keeping crystal trimming and transmitter output power and spectrum at the original status.

The calibration procedure of the antenna delay is trivial. It is necessary only locates the two antennas at a known distance, and by trial and error modify the antenna delay until the known distance and the reported range match.

Table 3.2: Suggested calibration distance for antenna delay

Channel	PRF (MHz)	Calibration Separation (m)
1	16	14.75
1	64	9.3
2	16	12.9
2	64	8.14
3	16	11.47
3	64	7.24
4	16/64	8.68
5	16	7.94
5	64	5.01
7	16/64	5.34

From the calibration process, it has been discovered that the measured distance changes based on if we are logging the data or if we are deploying the computed distance on the built-in LCD screen. Luckily, the aim of the project is to localize the UWB signal (CIRs), which not involve the computed distance, but it still remain an important concept for possible next development.
The reason why the calculated distance changes is because it takes longer time to store/log all the data in the ".txt" file. Indeed, when the measurement process starts, the deployed distance increase, on average, about 2.5 cm. Since the two boards will always be used logging the data (both when we have to store all the CIRs and when we have to localize our new measurement), it has been decided to calibrate them considering the result when the log option is on.

The channel selected and the used PRF are shown in tab. 3.1, which leads to select a calibration separation of 5.01 m (highlighted row in tab. 3.2). Multiple measure had been collected with different Antenna Delays.

The value which minimize the error had been the following:

$$\textbf{Calibrated Antenna Delay} = 516.025\ ns$$

3.2 Collected Dataset

Once the antennas have been calibrated and properly setup, the measurement of the environment begun. In order to collect datas as independent as possible, different dataset had been collected:

ID - **Ideal condition:** this dataset is made of 41 points, which cover the whole selected environment. For each point, 800 total CIRs had been collected, producing 400 whole distance measurements since the antennas work in two-way-ranging mode. This dataset, as the name says, has been collected in ideal conditions, which means that all the measures where in LOS and with static environment (e.g., no people in the surrounding).

R1 - **Real Condition 1st:** after collecting the dataset ID, it has been realized that ideal conditions are not really useful for a real localization since they are quite different from the reality. For this reason, a new dataset has been collected, trying to simulate as much as possible real condition (e.g., people walking between the antenna leading NLOS measurements, people moving in the surrounding, people seated on the benches or people using the elevator). This dataset, contrary to the previous one, has been collected for less points to speed up the measurements process, reducing the number of points from 41 to 13. In order to consider all the main points of the grid, the 13 points have been selected to cover both the main hall and the hallway, plus the point 5 that could be considered as special point since slightly NLOS and in front of the elevator, which could leads more interferences (green box in fig. 3.1). For each point a total of 2800 CIRs (1400 computed distance) had been collected, splitting them in specific conditions, such that it becomes easier analyse their influence on the specific point:

- 0-800 s.: ideal conditions
- 801-1000 s.: people walking
- 1001-2000 s.: seats on a bench
- 2001-2200 s.: antenna rotated about +45°
- 2201-2400 s.: antenna rotated about -45°

- 2401-2800 s.: waiting and using the elevator

R2 - **Real Condition** 2^{nd}**:** this new dataset is composed similarly to the R1 one (same selected points) with the only difference that the number of collected CIRs is 1000, and that the conditions are no more considered separately but they randomly overlap. The aim of collecting this new dataset, is just to have more real measurements collected in different moments to reduce possible offset errors.

R3 - **Real Condition** 3^{rd}**:** composed like dataset R2, with only difference in time instant of collecting the dataset. Indeed, also in this dataset, each point have almost 1000 CIRs.

To collect all the measurements, the Anchor (which connected to the movable cart, fig. 3.2 - a) has been moved such that its position matches with the one in the CAD file. The position has been taken with respect to the walls on the x and y-axes using a laser distance measurer (fig. 3.2 - b).

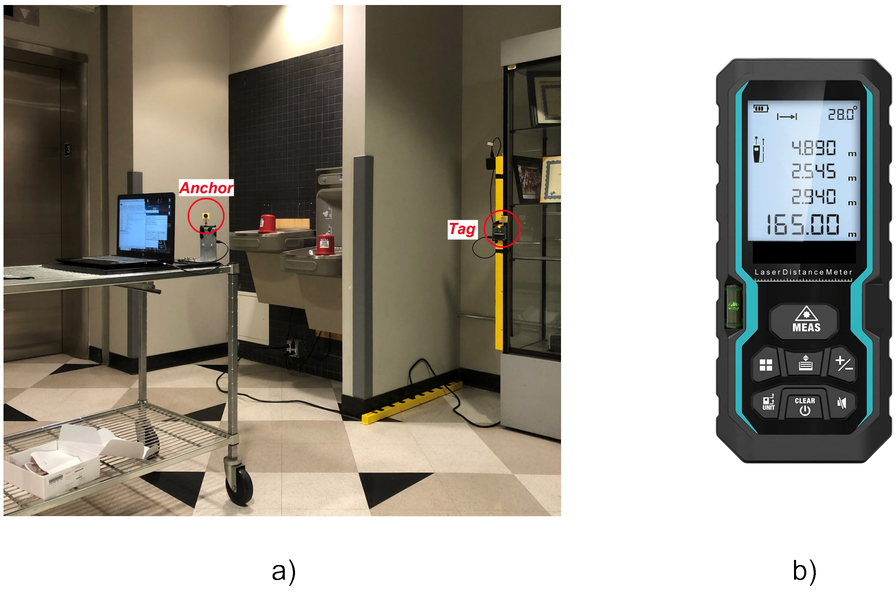


Figure 3.2: a) Measurements setup; b) Laser distance measurer

Once the Anchor reached the correct position, the modified version of the DecaRangning application has been run. All the setting has been changed to match the ones in tab. 3.1 and once all has been set correctly, it has been started to log all the measurements in a ".txt" file, named as the measured point.

3.3 Logged data

In order to log the data, the log button in the final version of DecaRangin application must be clicked. It automatically creates a new ".txt" file, where all the new TWR collected measurements will be stored. Due to the topology used, two CIRs are needed to complete a whole measurement estimator.

To extract all the needed information, a python script have been developed.

The data that have been finally used, with reference to fig. 1.17, are:

- **First Path Points:** it includes FP_IDX, FP_AMPL1, FP_AMPL2 and FP_AMPL3. As described in chap. 1, they give the information of amplitude overcoming the threshold noise, and the relative amplitudes of the first peak, which is associated to the LOS contribution.
- **CIR accumulator:** the accumulator is composed by two values, real and imaginary part, over which the final amplitude can be computed. It correspond to the final shape of the CIR, nevertheless the current localization information.

Data like SNR, CIR's power and carrier recovery have been extracted as well, but since they were not adding tracking information, it has been preferred to ignore them. On the other hand, Voltage and Temperature, even if not useful for localization, have been used for monitoring the antennas' correctness functioning. Indeed, have been considered as correct working conditions: $2.3 \leqslant V\ [V] \leqslant 3.75$ and $-40 \leqslant T\ [°C] \leqslant 100$.

Chapter 4

Dataset analysis

Before proceeding to feed the data to the network, is very important to analyse the data in their overall. In such a way, errors in the acquisition could be immediately discovered and correct. Moreover, knowing the data help for a deeper understanding of the problem. After that, comparison between different points have been carried out, in order to understand if data are effectively sensitive of the surroundings and/or at the dynamics caused by people.

From a first analysis, it shows up that the collected CIRs have the maximum peak always around 18/20k. Unfortunately, this means that their amplitudes do not add information about the strength of the signal (and so neither about the distance). Indeed, from the manual "DW1000 device driver application programing interface (API) guide" [21] is possible to read that an internal algorithm, called "LDE", which "details on the operation of the LDE algorithm are protected by IP and so are not publically available" [21], try to standardise the highest peak to 18/20k as a magnitude. Moreover, it can be highlighted analysing the amplitude of the direct path at the increase of the distance. From fig. 4.1, becomes easier to

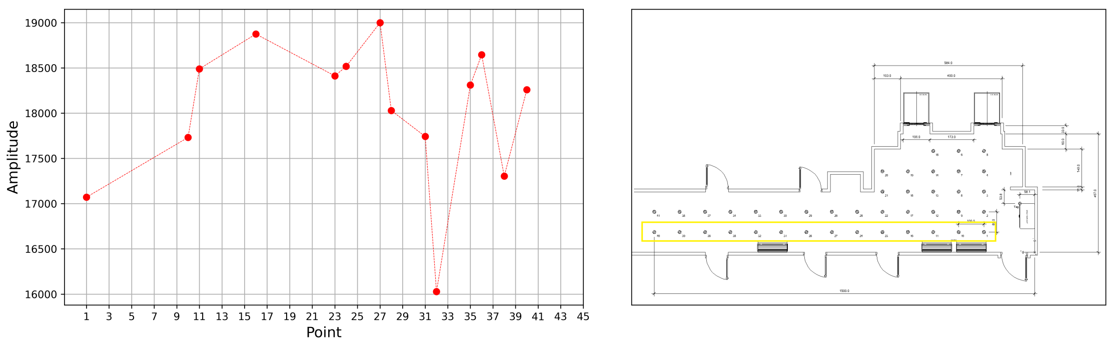


Figure 4.1: Amplitude at the variation of the distance

understand that the amplitude is not proportional to the distance. Except for the first four points, where seem that the amplitude increases at the increase of the distance, it randomly changes, remaining always nearby 18K.

4.1 CIRs' comparison

The first observation that has carried out has been on the magnitude's shape of the CIRs, where:

$$CIR_{amp} = \sqrt{CIR_{real}^2 + CIR_{imag}^2} \tag{4.1}$$

The aim has been trying to find some pattern via graphical intuition. As mentioned in paragraph 1.2, the useful and used samples from the whole CIR are only 157 time instant, rather than all 1016. It started analysing the dataset ID since it is related only on the surrounding, becoming easier to understand how CIRs change based on it. R1 and R2, instead, are affected from disturbances caused by people, leading to not have a direct correlation with the environment. This means that they will be slightly different from the ID, but with the same main shape.

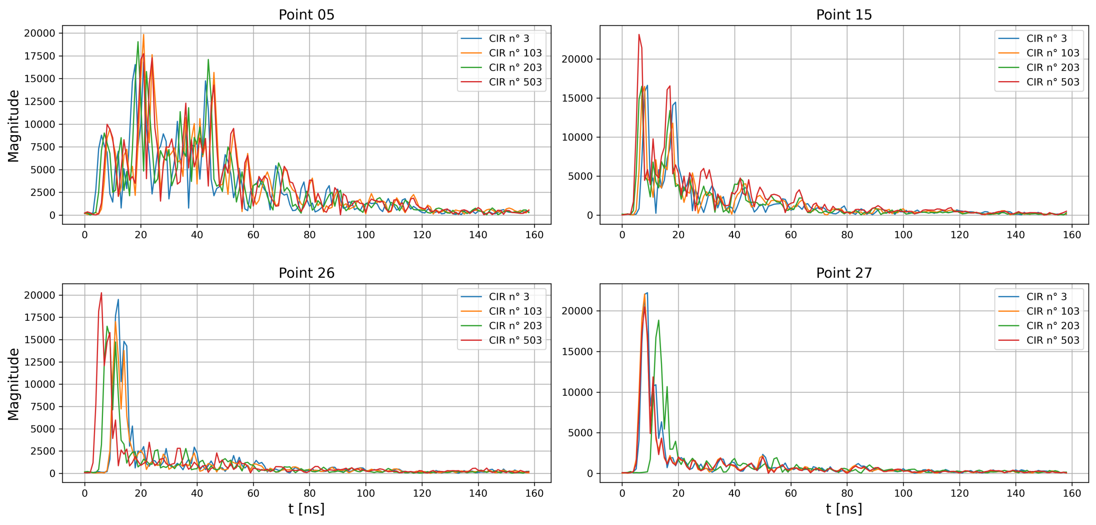


Figure 4.2: Comparison between different CIRs for different points

Since compare all the CIRs in a unique picture could be not clear, it has been preferred to use only four points (chosen as the most representative ones) and four random signals for each point. The result is depicted in fig. 4.2. The selected points are: 05, 15, 26 and 27 (with reference to fig. 3.2). All of them, as it is possible to see from fig. 4.2, are similar, but with specific different features. Point 05, indeed, is the only one having high magnitude multipaths. The reason why this happens is because this particular point is in front of the metal doors of the lift and slightly in NLOS. This leads to strong reflections. Point 15, instead, is completely in LOS but it is still close to the metal doors of the lift, which increase the multipaths' magnitude with respect to farther points. For example, point 26 and 27, which should not be affected by the elevators since in the hallway, have strong magnitude for the first peaks, but very low for all the others.

Another important detail to highlight is that these two latter points are at the same X location, distance 80 cm on the y axes. Even if structured in the same way, they still remain visible different (e.g., point 26 has higher magnitude in the second peak) which is a positive result since it means that the CIR is sensitive enough to sufficiently change for two adjacent points.

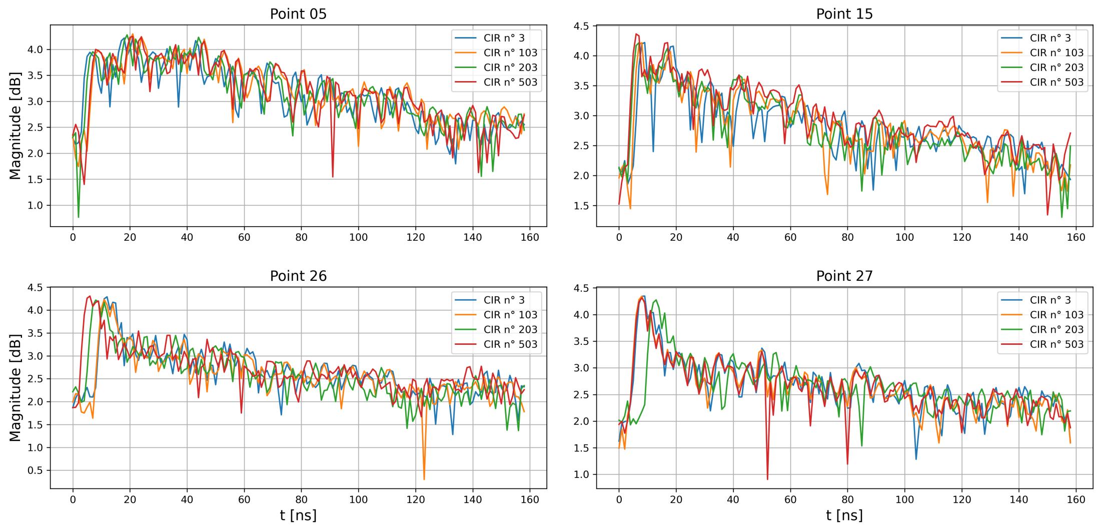


Figure 4.3: Comparison between different CIRs for different points, in $\log_{10}$ scale

In order to highlight also the differences when signals are close to 0, they have been represented in $\log_{10}$ scale. Unfortunately, even if from fig. 4.2 seems that the signals reach values close to 0, they still remain pretty high as magnitude (around 300) leading signals to look more similar, opposite of the initial goal. Indeed, looking at point 26 and 27 in fig. 4.3, they have more similar features than in normal scale. For this reason, it has been decided to not consider signals in $\log_{10}$ scale for all the next analysis, but keeping them as they are.

The following step has been comparing the behaviour of CIRs to the variation of the distance. For sake of simplicity, the average CIR has been computed for every point, obtaining a single final signal. First, analysis at the variation of the depth have been conducted (points 16, 23, 24, 27). Secondly, at the variation of the width (points 16, 17, 18, 19).

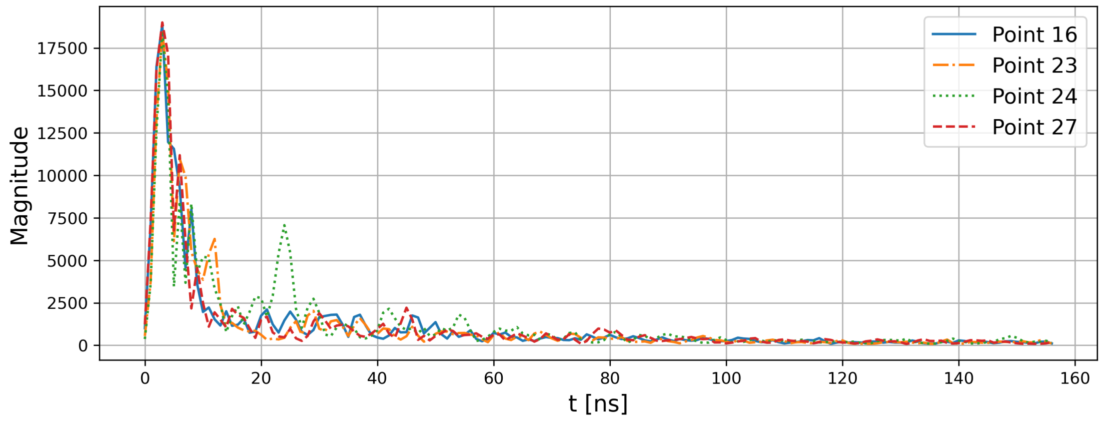


Figure 4.4: Averaged CIR's variation over x-axes distance

Analysis along the x-axis (depth)

The behavior of the averaged CIRs along the x-axes (moving along the depth of the hallway) is depicted in fig. 4.4. The main differences are located in the first 60 samples. The first peak, which correspond to the direct path, is almost the same for each point. This result is reasonable since the analyzed points are in line and the inside LDE algorithm try to equalize all the magnitudes. On the other hand, the second peak can be already considered different for the majority of cases. For example, in point 16, it is smoothed with respect to all the others. Another interesting behavior is that starting from the third peak (around the 10^{th} sample), until sample 60, all the peaks occur differently and with different magnitude.

Analysis along the y-axis (width)

Moving along y-axis, as for x-axis, produce different final CIRs. In this case, the closer we are to the walls, the higher magnitude the peaks will be. For example, looking at fig. 4.5, point 16 and point 19 are the points respectively at the wall and in front of the elevator and consequently they are the one with higher multipaths' magnitude. The second peak is an exception, which result to be lower with respect to the other 2.

In conclusion, moving along a single direction return slightly changes in the shape of the CIRs, which lead to think that ML algorithms should be able to properly classify the signals.

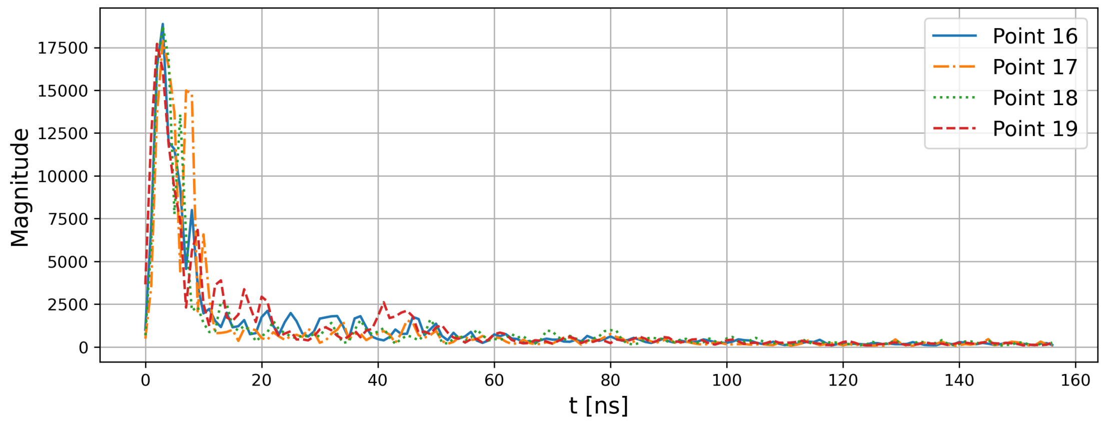


Figure 4.5: Averaged CIR's variation over y-axes distance

4.2 Power Delay Profile

It has been understood that multipath occurrences are one of the most important feature in order to characterize a point in a room, leading us to use the Power Delay Profile (PDP). PDP gives the intensity of a signal received through a multipath channel as a function of time delay. In other words, it focus the attention on the relative time and amplitude of the relative maximum peaks. The difference in travel time between multipath arrivals is the time delay. The abscissa is in time units (in this example, ns, which correlates to delays) and the ordinate is commonly in decibels (dB).

To plot the PDP of the averaged CIRs, at first they have been normalized as to get 0 dB for the point with the maximum amplitude (being the first path signal). Consecutively, all the other multipaths (peaks) have lower amplitude with respect to the first path signal.

Usually, the number of reflection paths are around 10 and, to achieve this result, a specific threshold value has been selected, cutting off all the other paths. From fig. 4.6, is possible to see that point 05 has much more than 10 path reflections. It happens because it has been decided to have a single threshold for all the points (1200 in normal scale), and since point 05 is the one in front the elevator (and with higher reflections), the number of paths will be more than the usual.

From this analysis, it is easier to highlight the difference in features between the 4 different points (compared to the simple CIRs of fig. 4.2). Indeed, comparing two adjacent points (e.g., point 26 and point 27), is possible to better understand

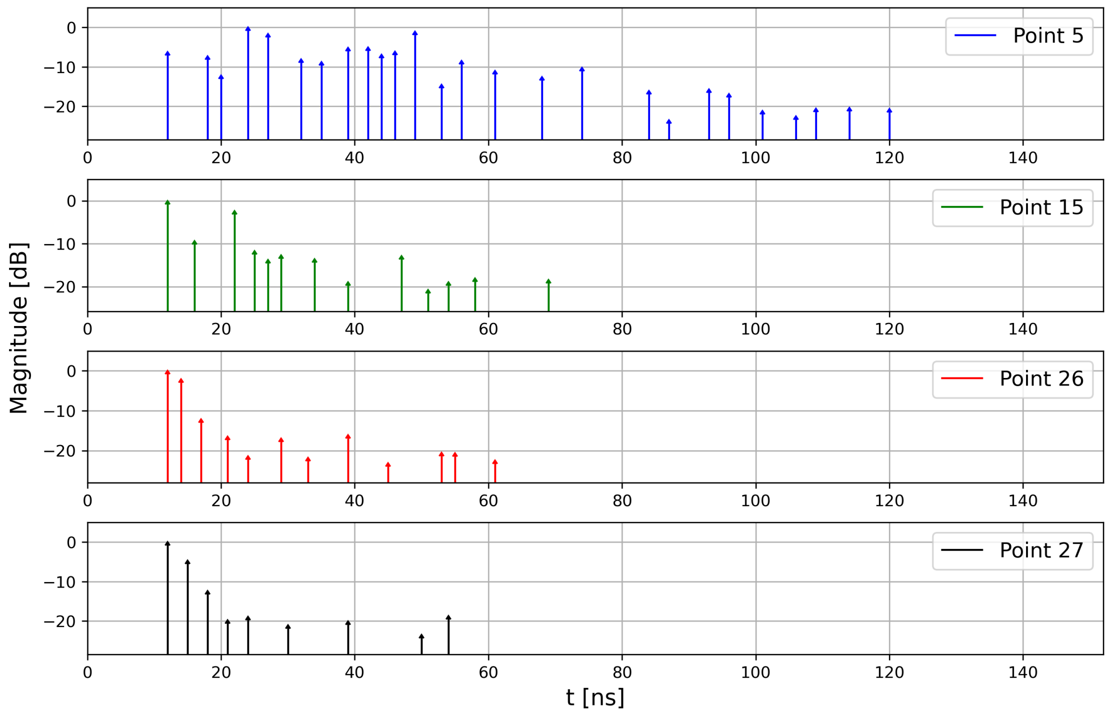


Figure 4.6: Power Delay Profile of 4 different points

their differences in paths. For example, point 26 has more reflection with higher amplitude. Moreover, all the 4 points have different time delay between one reflection to the other, characterizing in a unique way each point.

4.3 Histogram of delays

Another important analysis that can be carried out, is to analyse the frequency, or occurrences, of specific amplitude's multipath. The histogram of delays is the plot most relevant to show them.

As expected, from fig. 4.7 (a), is possible to see that for small amplitude (from 0 to 1000, first bin) the frequency of occurrence is very high. Unfortunately, this section of the histogram is not useful to the localization purpose, since small amplitude is mostly related to noises. A zoom on the rest of the graph has been performed (fig. 4.7 (b)). Contrary to what awaited, the histogram of delays do not give an easy interpretation on the feature of the points. Indeed, all the bins of amplitude higher than 7000 have an occurrences around 1 or 2. The peculiarity in this section of the graph, could be in understanding the exact amplitude on which there is the reflection, even if overlapping is present many times, which loose the

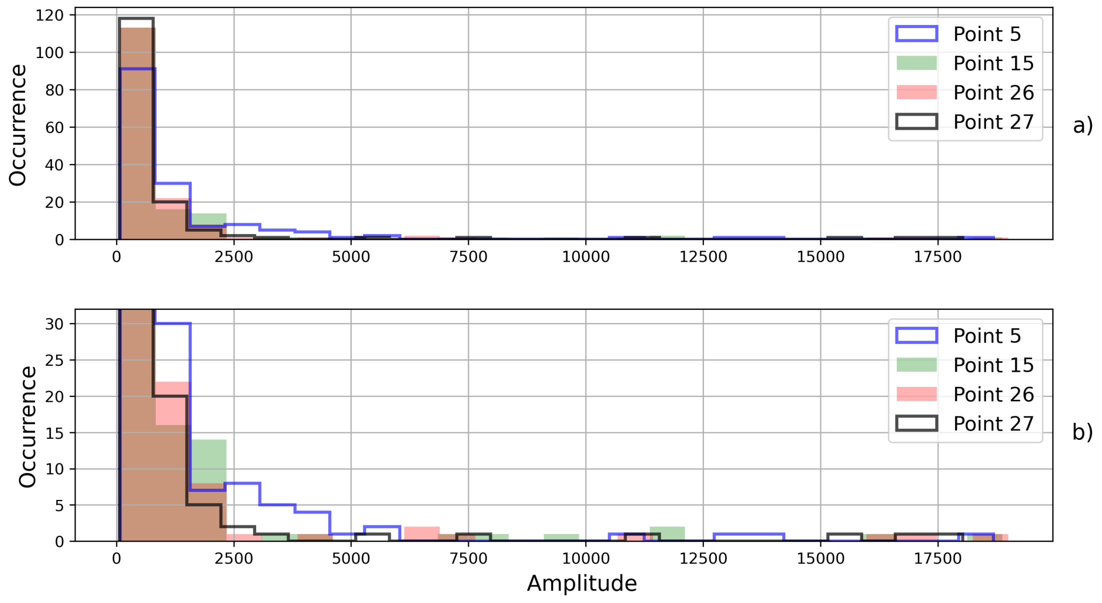


Figure 4.7: Histogram of delays of 4 different points

specificity of the point.

The bins whose represent the most distinct features are from the second to the amplitude value 7000. Here, occurrences ranges from 0 to 30 and all the 4 points have different trends. For example, point 05 (blue line), is the one having more reflection in this range of bins, giving immediately the idea of being the point in front of the elevator since metal material reflect more than normal walls. This means that the number of occurrences in this portion of the graph can give a first idea on how points are characterized.

4.4 What the non-idealities introduces

Until now, it has been used the ID dataset to perform all the comparison and analysis. The reason why it has been selected only this dataset is because it has been wanted to highlight how different points are dependent to the environment. This means that it has been desired to check the simple relation between specific points with respect to the surroundings, trying to avoid all the different noises coming from people's dynamic.

So, what people's dynamic means? In this thesis, people's dynamic is referred to reflection, attenuation or other kind of disturbances that people introduces when moving close to the antennas. As mentioned in chap 3 - sec. 3.2, the main introduced noises have been: people walking in the area and between the antennas, people sitting on benches, rotation of antenna about ± 45° (even if not related to persons) and people waiting and using the elevators.

In this section, the analysis of each specific disturbance has been carried out on R1 dataset, with the aim of understanding what they singularly introduce with respect to their absence. Since plot all the collected CIRs for each point could result in a not very clear graph, it has been decided to average all of them and compare the different conditions for only two point, which are point 05 and 26.

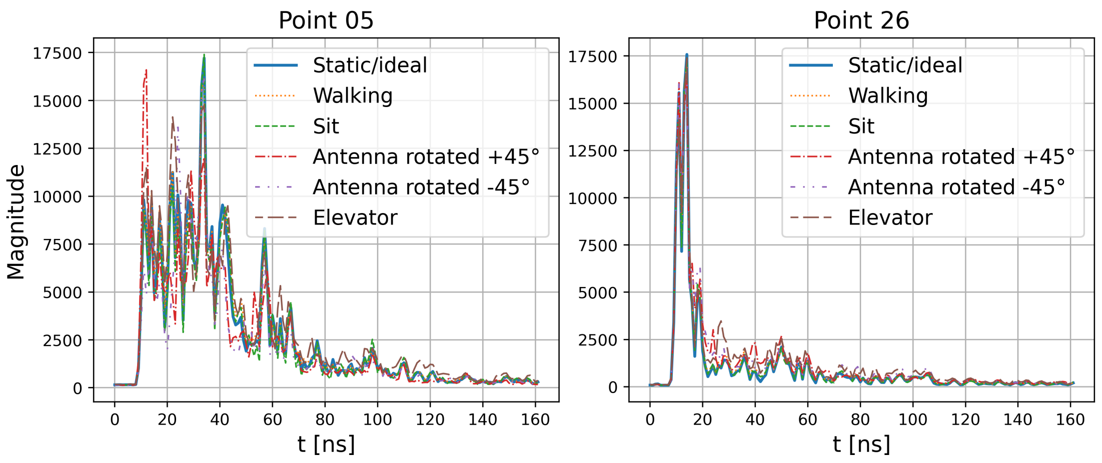


Figure 4.8: Comparison between ideality and non-ideality

Even if from fig. 4.8 is very complicated to understand which are the differences (due to overlapping of 6 similar graphs), it proves that, on average, the conditions change when people are in the surroundings. For a better interpretation on the introduced differences, multiple plots for two points (05 and 26) have been used.

Let's start considering ideal conditions with people walking (fig. 4.9). The main signal shape remains unaltered. Indeed, there are not so many differences between the blue line (ideal conditions) and the dashed red one (people walking in the surrounding). The reflections occur at the same time with small attenuation or amplification of some peaks. This trend takes place in both points, with more differences in point 05. It is important to remember that these are averaged signals, which means that, in general, multipaths could have higher and lower magnitude but, on average, they behave similarly to ideal conditions.

In fig. 4.10 is depicted the effect of people sitting on benches. As for people walking, the multipaths occur at the same time of ideal condition but, contrary to previous case, the magnitude tends to be slightly more attenuate. For example, in point 05, all the first reflections with people walking are attenuate with exception

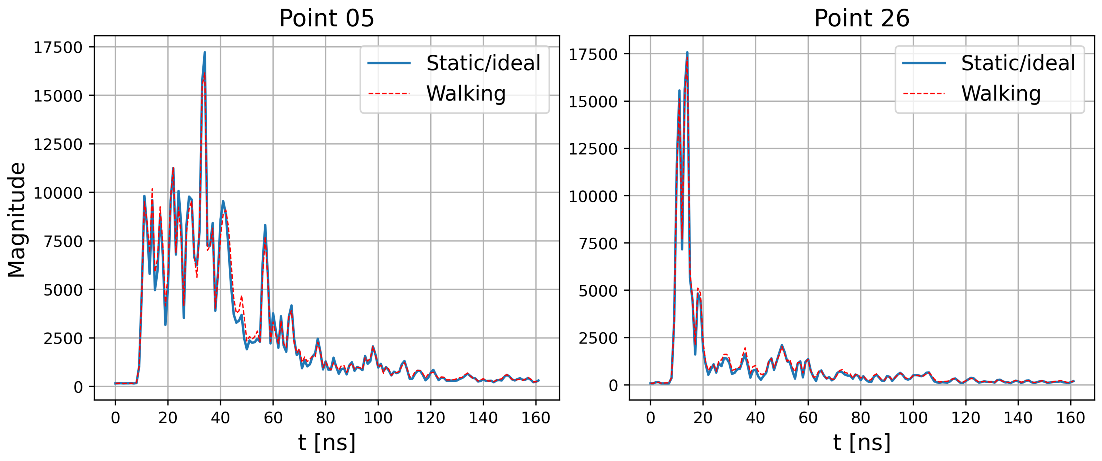


Figure 4.9: Comparison between ideal condition and people walking

of the direct path (higher peak). This exclusion is due to the fact that the LDE algorithm try to leverage the direct path around 18k. Point 26, instead, has the two conditions quite similar, with the signal almost perfectly overlapping. A possible reason why point 26 is not particularly affected by people sitting on the bench, could due its position in the environment. In other words, point 26 is located in front of the receiver (see fig. 3.1) with benches not very close to it, leading few reflections passing through people sitting on them.

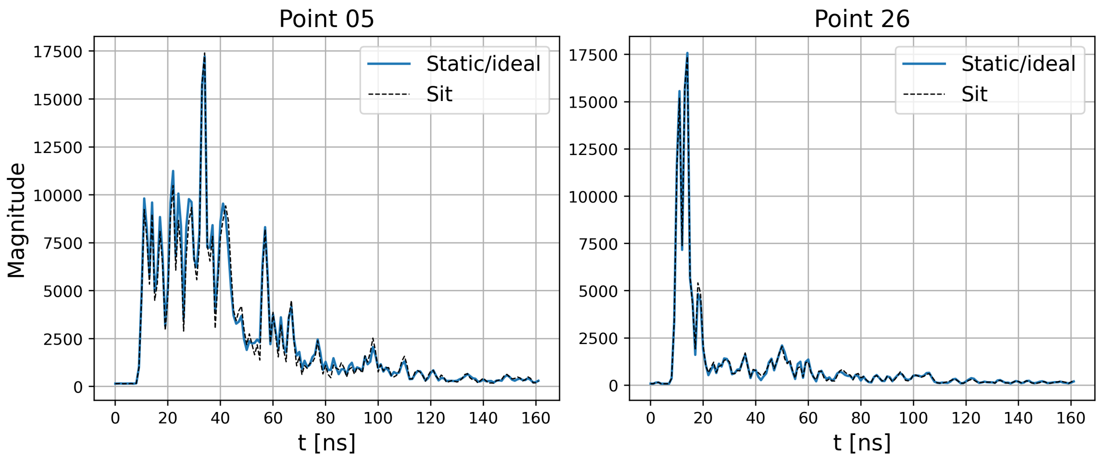


Figure 4.10: Comparison between ideal condition and people sitting

Next, the influence of people using the elevator (including people waiting in the hall and effectively using it) is depicted in fig. 4.11. This condition is the one who affect more the original CIR's shape. Luckily, also in this situation what change is

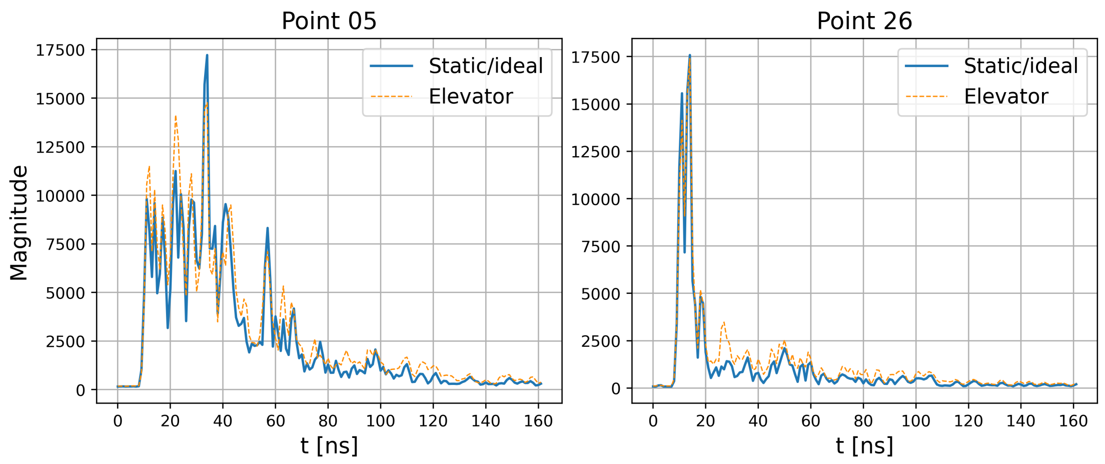


Figure 4.11: Comparison between ideal condition and people using elevators

not the shape itself, but is more the magnitude. Both points 05 and 26 kept their original shape with amplification of most of the reflections. This does not mean that the shape have been completely unaltered, indeed small variation are present, but still a minor part. In addition, also few peaks have been delayed in time, but always within 1 ns.

The last introduced non-ideality has been the rotation of the transmitter about ± 45°. The result is shown in fig. 4.12. Here, the two points behave slightly differently. Point 26 suffers a little less about the introduced disturbance. The shape remains mostly the same, with peaks amplification after the direct path. Instead, reflections in point 05, have been both amplified and attenuated (does not matter if the antenna are rotated about + or - 45°), with some peaks who change also in shape. Moreover, also the time when some multipath occur have been slightly shifted.

To sum up, is possible to state that the introduced non-idealities mostly change the magnitude of some multipath, leaving the main shape unchanged. This is a positive result since the ML algorithm can use the information of when the reflection happens and its shape. In such a way, even if it is slightly amplified or attenuated, it is still possible to extract unique features. Contrary, a critical situation could be when the antennas are rotated, since point 05 have been changed non only in amplitude but also in shape. Anyway, it does not mean that all the point will change their CIR's shape, but only that there is an higher probability. Indeed, point 26 has kept its shape, slightly modifying its amplitude.

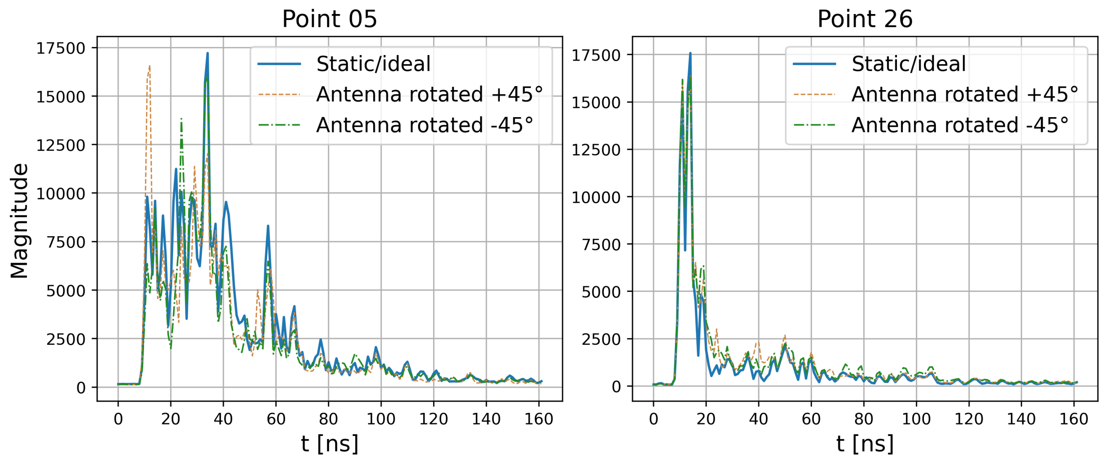


Figure 4.12: Comparison between ideal condition and antenna rotations

4.5 CIRs' correlation factors

After understanding how CIRs are different from each other and how they are related to the environments, the successive study has been on finding how much they are correlated between different points and different dataset.

First, the correlation formula (eq. 4.2) has been used to compare all the signals.

$$r = \sum_{i=1}^{N} \frac{(x_i - \overline{x})(y_i - \overline{y})}{(N-1)s_x s_y} = \frac{\dfrac{\sum(x-\overline{x})(y-\overline{y})}{N}}{\sqrt{\dfrac{(x-\overline{x})^2}{N}}\sqrt{\dfrac{(y-\overline{y})^2}{N}}} \tag{4.2}$$

where,

- **r** is the product-moment correlation and indicates the similarity's strength of the analyzed variable. It ranges from - 1.0 and + 1.0, where the sign indicate the direct or inverse relationship and the magnitude indicates the strength.
- s_x and s_y are the standard deviation of x and y respectively
- $\overline{x}$ and $\overline{y}$ are the respectively mean of x and y
- **N** is the number of cases

Then, it has been reported the correlation trend comparing each CIR of the same point (fig. 4.13 (a)) with different dataset (fig. 4.13 (b))

Fig. 4.13 a) shows how much CIRs are consistent for the same point. In other words, it depicts the average of the correlation coefficient for all the CIRs belonging

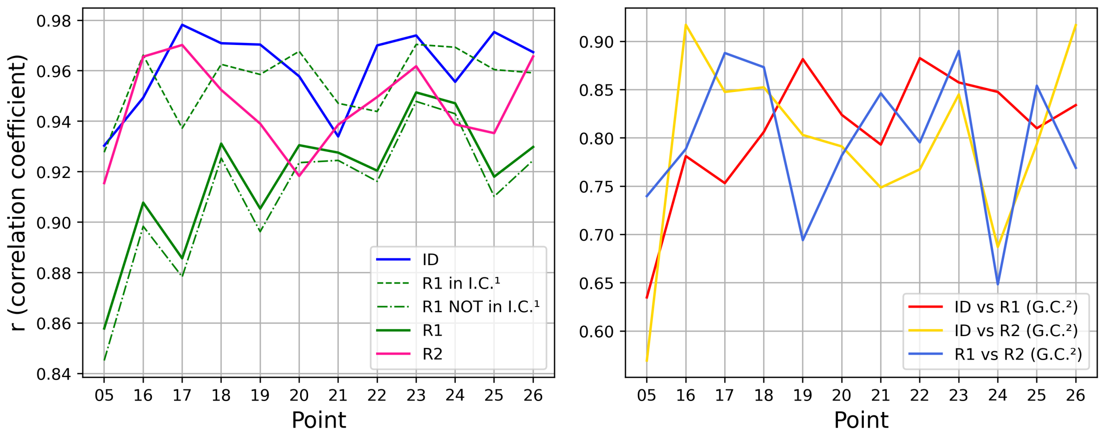


Figure 4.13: Correlation trend for CIRs of the same dataset (left - a)and for different dataset (right - b)

to the same point. All the correlation coefficient are bounded between 0.85 and 0.98. R1 is the dataset with lower r, while the most consisting one is ID (as it is reasonable to be since it includes only static measures).

In order to highlight the effect of people's dynamic, the global R1 trend (green line) has been split in two. The correlation has similar trend to the dataset ID when considering only the first part of the measures, which correspond to ideal conditions. This is a logical result since the environment and conditions are the same. Contrary, performances degrade (green dash-dotted line) when dynamic is introduces, decreasing, in the worst scenario, 9% of correlation (point 05). Since the overall trend is the average of all the correlation coefficients belonging to the same point, the final R1 trend remains closer to the dynamic condition because static and dynamic samples are not balanced. Indeed, (with reference to sec. 3.2) R1 contains three times more dynamic samples with respect to the static ones.

Points 05, 21, 22 and slightly 24 show the lower r, realizing that for these points it could be challenging collect consistent data, becoming for the ML network more difficult the classification.

In order to analyse the correlation between different dataset, the attention must be focused on fig. 4.13 (b). Here, all the CIRs belonging to the a point's dataset have

[1] I.C. = Ideal Condition

[2] G.C. = General Condition (used to indicated that includes both ideal condition and people's dynamic)

been compared with the collected CIRs of different dataset, for the same point. It is instantly noticeable the worsening of performances, having r bounded between 0.55 and 0.91. The most critical points seem to be the same of when analysing the same dataset (that are 05, 21, 22 and 24). Point 19, instead, result a "weak" point when comparing R1 with R2.

Moreover, looking deeper at the graph is possible to notice that when comparing both the dataset with dynamics, the results are more consistent. The reason behind this result is straightforward: dynamics affect the dataset in a similar way, or simpler, there are more similar samples in same condition. For example, when comparing ID with R1, we are comparing thousand of ideal condition's CIR of ID with hundred of ideal condition in R1, plus all the one with dynamics.

It is also worth to highlight that the ideal correlation coefficient must not be 1, otherwise, even if has been collected 2 millions samples, they do not add new information to the network. Similarly, it neither must be too small, otherwise the classification becomes impossible.

In short, the goal is to give as input to the neural network signals which are unique for each point, but with a lot of subtle differences, in order to generalize the classification and not overfit.

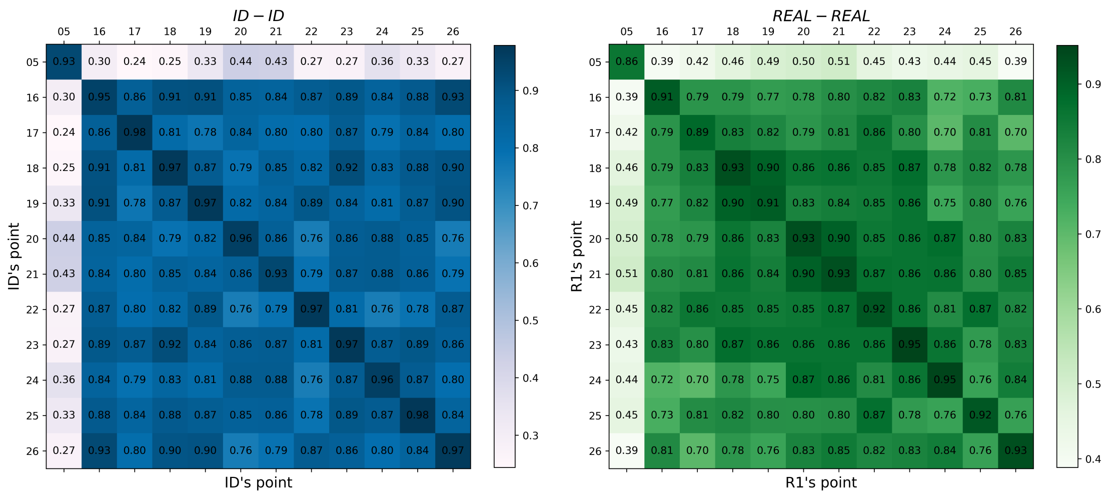


Figure 4.14: Correlation coefficient between ID-ID (left) and R1-R1 (right)

Finally, to have a broader perspective, the correlation coefficient has been computed between all the points of the three dataset, forming matrices where each cell represent the average r of the points on x and y-axes. Inside these matrices are present all the possible combinations between all the points and dataset. The values on the main diagonal of ID-ID, R1-R1 and R2-R2, correspond to the same

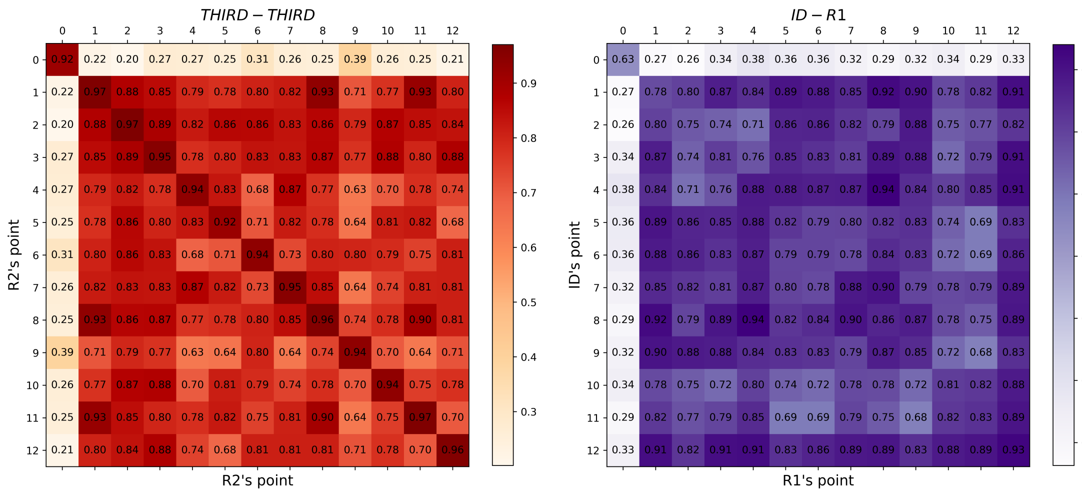


Figure 4.15: Correlation coefficient between R2-R2 (left) and ID-R1 (right)

values of fig. 4.13, with only difference of data view.

Point 05 is the most distinct from all the others points. This is true, since is the only one slightly NLOS and with high reflective material nearby. Indeed, in all the matrices, the correlations between point 05 and all the other points is always bounded between 0.20 and 0.50, which are less than half compared to the average. Albeit on the main diagonal the desired value is neither 1 nor values too small, the desired coefficient value off the diagonal is as smaller as

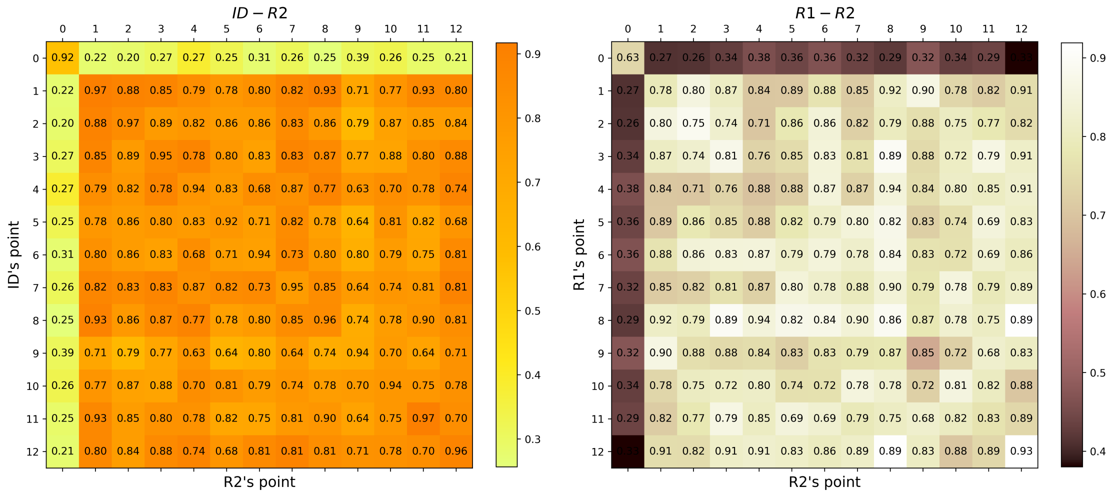


Figure 4.16: Correlation coefficient between ID-R2 (left) and R1-R2 (right)

possible. The optimal situation occurs when the CIRs of different points are mostly different. Obviously, it is impossible to get a correlation coefficient equal to 0 since the environment is the same and the points are distant few cm to each other.

Unfortunately, from fig. 4.14, 4.15 and 4.16, all the coefficients off the main diagonal remain pretty high (always above 0.70), by exception, as mentioned before, for point 05. This could lead to higher classification complexity, being harder to identify uniquely new unseen data.

4.6 ToF

The last performed analysis has been on the ToF.It has been decided to study how much consistent are these measurement. In fig. 4.17 are depicted all the collected ToF for dataset R1 (point 05 and 26 has been used as references, as showing the trend for all points could be confusing).

Before starting to analyze the trend, it is worth to remember that, to measure one ToF, two CIRs are needed. For this reason, with reference to sec. 3.2, there are half ToF's measures then CIRs.

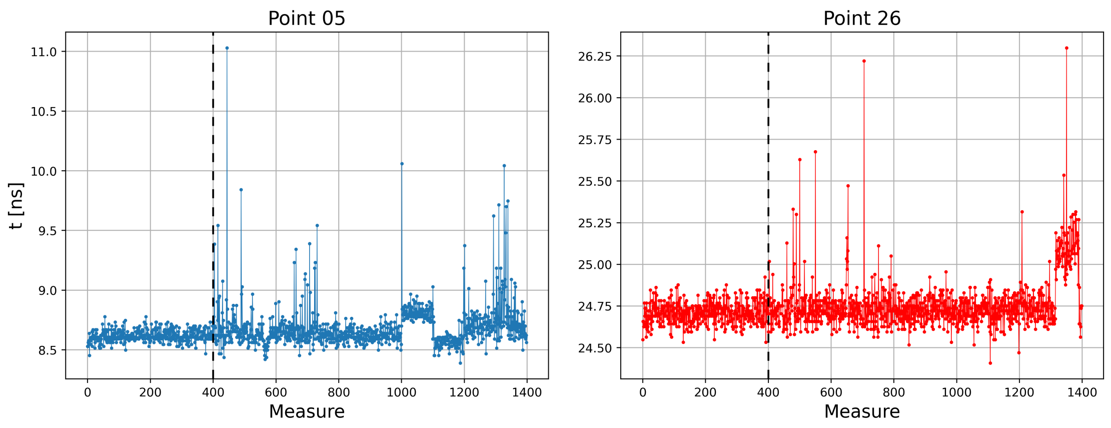


Figure 4.17: Measured ToF for point 05 (left) and 26 (right) - dataset R1

In both the graphs has been highlighted, with a dashed line, the parting line between measures collected in static (or ideal) condition (on the left) and measures collected in dynamic condition (right side). It is instantly recognizable that, in both the cases, dynamic alters a lot the final measure. In tab. 4.1 are shown the average ToF for the different conditions.

The different between static and dynamic condition may seem small. It is 0.068800 ns for point 5 and 0.014939 ns for point 26. The main problem here, is that to compute the distance in m, the ToF must be multiplied by the speed of

Table 4.1: Averaged ToF in different conditions - dataset R1

Point	**Static Cond.** [ns]	**Dynamic Cond.** [ns]	**Global** [ns]
05	8.6179	8.6867	8.6670
26	24.7101	24.7624	24.7474

light, which is 299'792'458 m/s. So, small delta in time, will result in significant distance error. For example, the different ToFs result in cm, are:

$$\Delta_{point\_05} = 0.068800 \cdot 10^{-9} \; * \; 299792458 = 0.020625 \; m = 2.0625 \; cm$$
$$\Delta_{point\_26} = 0.014939 \cdot 10^{-9} \; * \; 299792458 = 0.004478 \; m = 0.4479 \; cm$$

which, for indoor localization, could be a critical factor.

Moreover, when considering dynamic conditions (fig. 4.17), a large number of outliers are present (measures farther than 3σ from average). It could be due to situation in NLOS, where people passed through the two antennas, or due to interferences with other objects. So, it is important to collect big dataset, such that when training the network it is less affected from outliers.

In addition, for point 05, there has been a constant offset for measures from 1000 to 1100, corresponding to CIRs from 2000 to 2200 (antenna rotation about +45°) and a rise in number of outlier for the last 200 measures, corresponding to people waiting and using elevators.

Since it has been observed that ToF is very sensitive to dynamic, the indoor localization will be attempted only using CIRs.

Chapter 5

Pre-processing operations

After having an idea on how the data are formatted, some preliminary operations must be performed in order to have robust and consistent data for the network. It is well known that, nowadays, ML algorithms are strong enough to use raw data, but the more they get simplify, the easier will be the classification.

5.1 Alignment

During the dataset collection, it has been noticed that the main shape of CIRs were pretty similar for data of the same point, with the only exception of starting point. Indeed, almost all the CIR started delayed in time, and since the LDE[1] algorithm was the cause of it, it has been impossible to set it at priori. For this reason all the data have been post aligned.

To do the alignment, it has been decided to use the maximum correlation. It has been used a random CIR as reference, and all the other have been shifted in order to have most similarity to the referenced one. Equation 5.1 shows the formula for the correlation's computation:

$$c_k = \sum_n x_{n+m} \cdot \overline{y_n} \tag{5.1}$$

with $-\infty < n < +\infty$ and $\overline{y_n}$ denoted the complex conjugate of y.

The result is an array containing the value of correlation for each shifted sample. The index containing the maximum value, identify how much the CIR must be shifted.

[1]It is the DW1000 internal algorithm responsible of signal identification, over which not details are provided from the company

As example, in fig. 5.1 is reported the obtained result for point 26.

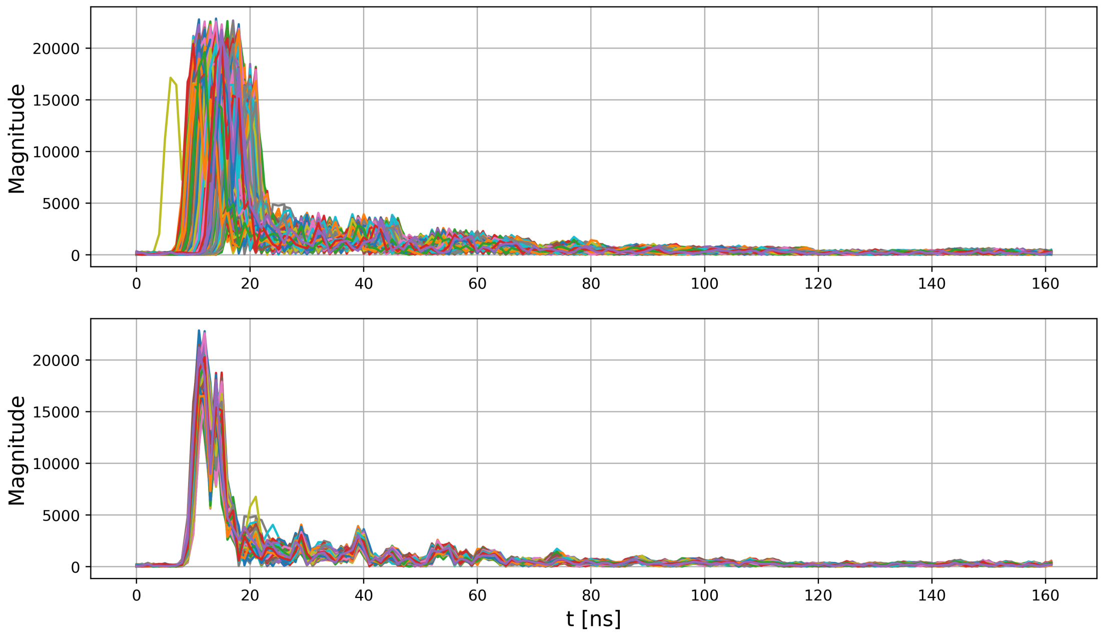


Figure 5.1: Before (up) and after (down) the alignment processing, for point 26 - dataset ID

5.2 Normalization

Normalizing input data aims to create a set of features that are all on the same scale, as well as to avoid common problems known as vanishing and exploding gradient problem. This means that normalization is not used to get normally distributed data, but it is only related to obtain data that work better with the network algorithm.

The goal is usually to recenter and rescale data such that they are between 0 and 1, or -1 and 1, depending on the data itself.

In this thesis' project, data have been normalizing by scaling each CIR to a given range [$r_{min} = 0$ - $r_{max} = 1$]. All the collected CIRs have been scaled in this range. The transformation is given by eq. 5.2:

$$\begin{gathered} x_{std} = \frac{x - x_{min}}{x_{max} - x_{min}} \\ x_{normalized} = x_{std} * (r_{max} - r_{min}) + r_{min} \end{gathered} \tag{5.2}$$

In fig. 5.2 is shown a random CIR of point 26, both in normal scale (left axes) and after the normalization (right axes).

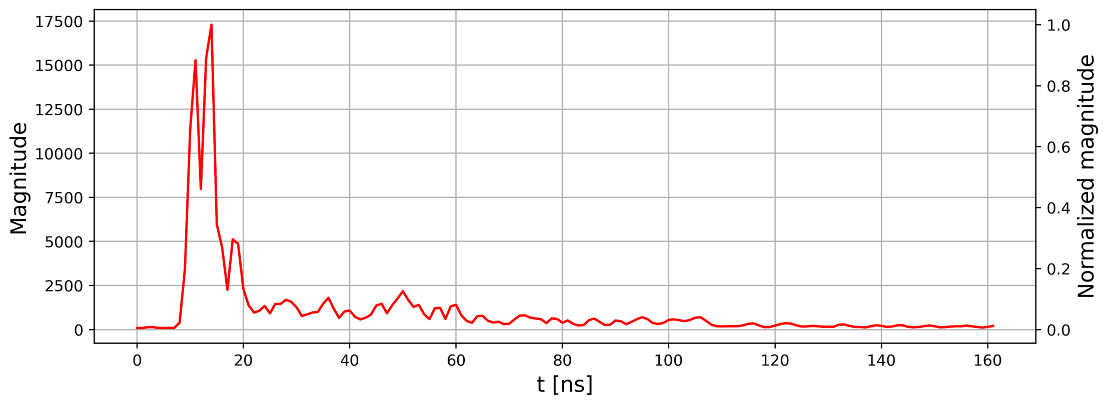


Figure 5.2: Example of normalization on a random CIR of point 26 - dataset R1

As it is a simple normalization, the shape remain unchanged.

Chapter 6

Network selection

Neural network selection remains a difficult problem whose solution can only be achieved through trial and error. The main reason why it remains challenging, is because the training process is solved using optimization algorithms containing lot of parameters, where each of these can improve or worsen the final result. This means that the user must try to find the best values to achieve the best final result, and since there are no equations for its detection, choosing the most promising hyperparameters and well-functioning model architectures continues to be a challenge also for experts.

These issues have led to approach the machine learning area from a different perspective, entering in a new area of research called autoML. The aim of this research is to search some well-performing ML models or algorithms with the automated choice of hyperparameters.

6.1 What is mcfly

Mcfly[1] is an autoML software tool that focuses on deep learning for multivariate time series classification. It is a Python library developed by the Netherlands eScience Center to generate different deep learning networks and train them on specific dataset. A peculiarity of mcfly is that it provides the user the ability to choose particular ranges of values for the hyperparameters (depending on domain knowledge and data science) or to explore a large number of networks [22].

Mcfly is a packaging around the Keras[2] API in Tensorflow[3]. Even though the

[1]https://github.com/NLeSC/mcfly

[2]Keras is an effective high-level neural network library that runs on top of Tensorflow.

[3]Tensrflow is an end-to-end open-sourced deep learning framework, developed by Google and

Keras API in Tensorflow is a powerful and user-friendly API, it still requires the user to define the architecture of the model and other hyperparameters (e.g., learning rate). What mcfly propose, is searching architectures and other hyerparamethers through random search[4][22].

The steps performed have been:

- **Prepare input data:** the input data format accepted by mcfly coincide with the one accepted by the Keras API, but limited to matrices representing single or multi-channel time series data. All sequences in the dataset should have the same length. The input data X should be of shape (num$_{CIR}$ $\times$ num$_{timestamp}$ $\times$ num$_{channel}$), while the output data y should be (num$_{CIR}$ $\times$ num$classes$), as a binary array for each collected CIR.
- **(Re)generate model(s):** there are four types of networks architecture that are available in mcfly: CNN, DeepConvLSTM, ResNet and InceptionTime. In case it is preferred to use only a subset, it is possible to set which one to use in the mcfly algorithm.
- **Find the best model:** the best model is obtained looking over the performances of all the random generated model. Few optional settings are tunable for training the models in mcfly, such as the number of epoch and early stopping (whether the training process should stop when performances does not increase).
- **Fine tuning and store the model:** once the best model has been found, a final tuning with the whole dataset must be performed. If the model still returns good performances, the model can be used for the definitive classification.

6.2 Selection process

Let now deeper analyze the steps mentioned in the previous section.

STEP 1 - Data split

Before start generate different models, it is important to have proper subsets on which train the networks. Indeed, once all the data have been properly processed

released in 2015.

[4]Different studies[23] affirm that random search is more efficient than other commonly used techniques, as grid search.

and organized in structures compliant with mcfly's algorithm, three different subset have been extracted. It has been decided to use only dataset R1 for the model research. So, the three subsets have been split as follow:

- Training: 70%
- Validation: 20%
- Testing: 10%

These are common percentage in Machine Learning.

STEP 2 - Generate models

The second step has been generate different models through random search to investigate which architecture is most suitable for our data and classification task. The number of models is an user parameter, which means it must be specified in mcfly by the user. In order to have a wider analysis, it has been decided to generate 12 different random models. The number of generated models could be arbitrary chosen to the user. Below, is reported the code to generate models.

```
    num_classes = y_train_enc.shape[1]
    metric = 'categorical_accuracy'
    models_1st = mcfly.modelgen.generate_models(X_train.shape,
                                    number_of_classes=num_classes,
                                    number_of_models = 12,
                                    metrics=[metric])
```

The 12 generated models are reported in tab. 6.1, highlighting their hyperparameters, which are: learning rate (that determines how fast moving toward a minimum of a loss function for each iteration) and regularization rate (that controls the regularization applied to the model, modifying bias and variance)

STEP 3 - Train models

Once all the models architectures have been generated, the weights have been tuned by training on the training dataset and evaluating on the validation subset.

In order to quickly train the models and get good results, early stopping (which is the parameter that will stop the training once the validation accuracy is not improved) has been set to 10, the subset size has been set 3000 and the epoch (that is the number of times the subset is iterated over) to 100.

Below, the code used to train all the models has been reported.

Table 6.1: List of generated models

Model n.	**Model type**	**Learning rate**	**Regularization rate**	**Tot params**
1	ResNet	0.02432507	0.00138735	3’280’521
2	InceptionTime	0.00136961	0.00258567	603’954
3	DeepConvLSTM	0.02796922	0.00044054	66’419
4	CNN	0.01285124	0.01368969	23’150’482
5	ResNet	0.00892321	0.00395036	723’810
6	InceptionTime	0.00041757	0.00102521	2’195’470
7	CNN	0.00296089	0.02229411	8’957’966
8	DeepConvLSTM	0.00198869	0.00521629	177’670
9	DeepConvLSTM	0.00284828	0.02172788	294’100
10	InceptionTime	0.01074923	0.00051785	724’122
11	CNN	0.00474171	0.00307608	5’108’995
12	ResNet	0.08381990	0.00103132	7’238’832

```
histories_1st, val_accuracies_1st, val_losses_1st =
                    train_models_on_samples(X_train, y_train_enc,
                                            X_val, y_val_enc,
                                            models_1st, nr_epochs=100,
                                            subset_size=3000,
                                            early_stopping_patience=10)
```

STEP 4 - Models performances

In STEP 2, when training the model, four different parameters have been computed and stored, which are: training accuracy, training loss, validation accuracy and validation loss. All of these are used to asses the performances of the model.

The model’s performances are shown in tab. 6.2.

The evaluation of the best model, must be performed on the validation dataset, which are data not used in training. It means that when comparing different networks, an important parameter to observe is the validation accuracy.

Based on this observation, it has been decided to narrow the model research by considering only types whose validation accuracy reached, in at least most of the cases, good percentage. "DeepConvLSTM" has been the worse model type among the four, with validation accuracy of 7.01% and 6.87% (model 3 and 9). So, it has been decided to not consider it for the following steps. All the other types, at least

Table 6.2: Performances of generated models

Model n.	Model type	Training accuracy	Training loss	Validation accuracy	Validation loss
1	ResNet	0.6813	0.9216	0.3423	7.5334
2	InceptionTime	1.0000	5.14E-05	0.9936	0.0252
3	DeepConvLSTM	0.0729	2.6464	0.0701	2.6516
4	CNN	0.9380	0.5753	0.6292	1.7881
5	ResNet	0.9916	0.0282	0.5680	4.0723
6	InceptionTime	1.0000	3.42E-07	0.9961	0.0193
7	CNN	0.9693	0.2900	0.0781	59.7078
8	DeepConvLSTM	0.8190	0.8585	0.6906	1.2550
9	DeepConvLSTM	0.0769	2.6392	0.0687	2.6403
10	InceptionTime	0.9969	0.0129	0.8515	0.7041
11	CNN	0.9840	0.2343	0.8277	0.7229
12	ResNet	0.0753	2.7651	0.0684	2.6858

2/3 have validation accuracy higher than 50%.

STEP 5 - Narrowing model generation

Once it has been decided to examine only "CNN", "InceptionTime" and "ResNet" as model types, 20 new random models have been generated. Below, the code used to generate the new model.

```
1  num_classes = y_train_enc.shape[1]
2  metric = 'categorical_accuracy'
3  types = ['CNN', 'ResNet', 'InceptionTime']
4  num_models = 20
5  max_layers = 12
6  models_2nd = mcfly.modelgen.generate_models(X_train.shape,
7                                  number_of_classes=num_classes,
8                                  number_of_models = num_models,
9                                  model_types = types,
10                                 cnn_max_layers = max_layers,
11                                 resnet_max_network_dept = max_layers,
12                                 IT_max_network_dept = max_layers,
13                                 metrics=[metric])
```

In addition, as it can seen in line 5, the maximum number of layers have been limited to 5. The reason behind it, is that, in STEP 1, the generated models were really deep. Using an upper limit in the number of layers reduces the network's complexity.

The new generated models are reported in tab. 6.3.

Table 6.3: List of generated models

Model n.	**Model type**	**Learning rate**	**Regularization rate**	**Tot params**
1	CNN	0.00021631	0.00562004	6’082’935
2	ResNet	0.03080126	0.00022648	3’304’299
3	InceptionTime	0.00024731	0.00056710	81’503
4	CNN	0.00037876	0.00040303	4’568’105
5	InceptionTime	0.00134682	0.02950281	876’530
6	ResNet	0.01086536	0.00643040	479’743
7	CNN	0.00406230	0.00027741	27’405’936
8	InceptionTime	0.05177687	0.01408421	376’364
9	ResNet	0.00385275	0.00224652	309’965
10	InceptionTime	0.07135029	0.02012080	1’155’438
11	CNN	0.00024522	0.02227372	9’802’394
12	ResNet	0.08095274	0.00848389	2’850’792
13	ResNet	0.00034802	0.05356596	2’953’447
14	InceptionTime	0.00098791	0.02546353	928’898
15	CNN	0.00173172	0.00010569	1’628’097
16	InceptionTime	0.00264728	0.07459546	331’173
17	CNN	0.04015362	0.09334354	8’254’467
18	ResNet	0.00061688	0.00374441	2’761’336
19	InceptionTime	0.00015089	0.04938399	255’762
20	ResNet	0.08027593	0.06263171	992’494

It is possible to see that the total number of parameters between all the models range from ten thousand until reaching millions. These are a large number of parameters.

So, even if nowadays, computer are able to perform complex computation in very little time, it has been preferred to consider number of parameters as crucial factor in the model selection, trying to get it as small as possible.

STEP 6 - New model performances

In tab. 6.4 are reported all the performances of the 20 models.

In order to pick the best model, different aspects have been considered. The best model must best fulfill the requirements listed below:

- High validation accuracy

Table 6.4: Performances of the new 20 generated model

Model n.	Model type	Training accuracy	Training loss	Validation accuracy	Validation loss
1	CNN	0.9163	0.9147	0.2122	7.3786
2	ResNet	0.8767	0.3979	0.3962	11.6138
3	InceptionTime	1.0000	0.0003	0.9795	0.0743
4	CNN	1.0000	0.0333	0.9902	0.0633
5	InceptionTime	1.0000	4.26E-06	0.9934	0.0269
6	ResNet	0.9740	0.0872	0.7840	2.0212
7	CNN	0.9970	0.0363	0.4909	4.2755
8	InceptionTime	1.0000	3.75E-05	0.9864	0.0694
9	ResNet	0.9903	0.0355	0.8868	0.4637
10	InceptionTime	1.0000	2.11E-05	0.9891	0.0526
11	CNN	0.9987	0.1969	0.9870	0.3451
12	ResNet	0.0747	2.7524	0.0745	2.7243
13	ResNet	1.0000	7.93E-05	0.9925	0.0301
14	InceptionTime	1.0000	1.12E-05	0.9952	0.0201
15	CNN	1.0000	0.0164	0.9885	0.0505
16	InceptionTime	1.0000	2.65E-06	0.9955	0.0192
17	CNN	0.7997	3.0394	0.7735	3.6095
18	ResNet	1.0000	0.0001	0.9887	0.0445
19	InceptionTime	1.0000	2.79E-05	0.9938	0.0198
20	ResNet	0.2797	1.9953	0.0732	26.6911

- Small number of parameters (it has been decided to use an upper limit of 1 million)
- No overfitting or underfitting

To display performances more easily, in fig. 6.1, it has been reported the trend of all the models at each iteration.

Now, analysing tab. 6.4 and fig. 6.1, 5 models (model n. 1, 2, 12, 17, 20) have been discarded since perform poorly on the training set.

Moving forward, it has been analyzed the validation accuracy (tab. 6.4 and fig. 6.2). Here only 12 models (model n. 3, 4, 5, 8, 10, 11, 13, 14, 15, 16, 18, 19) have a validation accuracy higher than 97%. So, all the other have not been considered.

Model n. 8, 10, 13 and 18 follow a particular behaviour. Indeed, looking at fig. 6.2, even if most of the iteration have an acceptable validation accuracy, they oscillated for the first 35/50, 19/38, 15/27 and 17/28 iterations, respectively. Since other models have better response, it has been decided to reject all of them, keeping

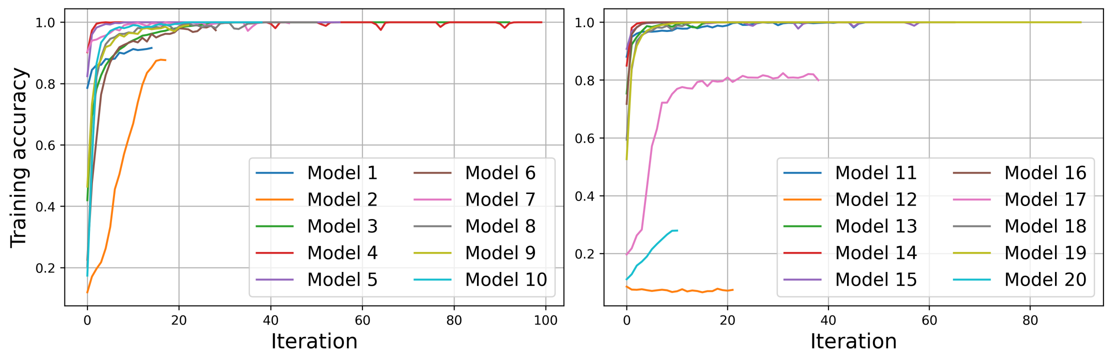


Figure 6.1: Performance on training set

models n. 3, 4, 5, 11, 14, 15, 16 and 19 as a final possible networks. This means that all of these 8 models are a good fit to be the best model.

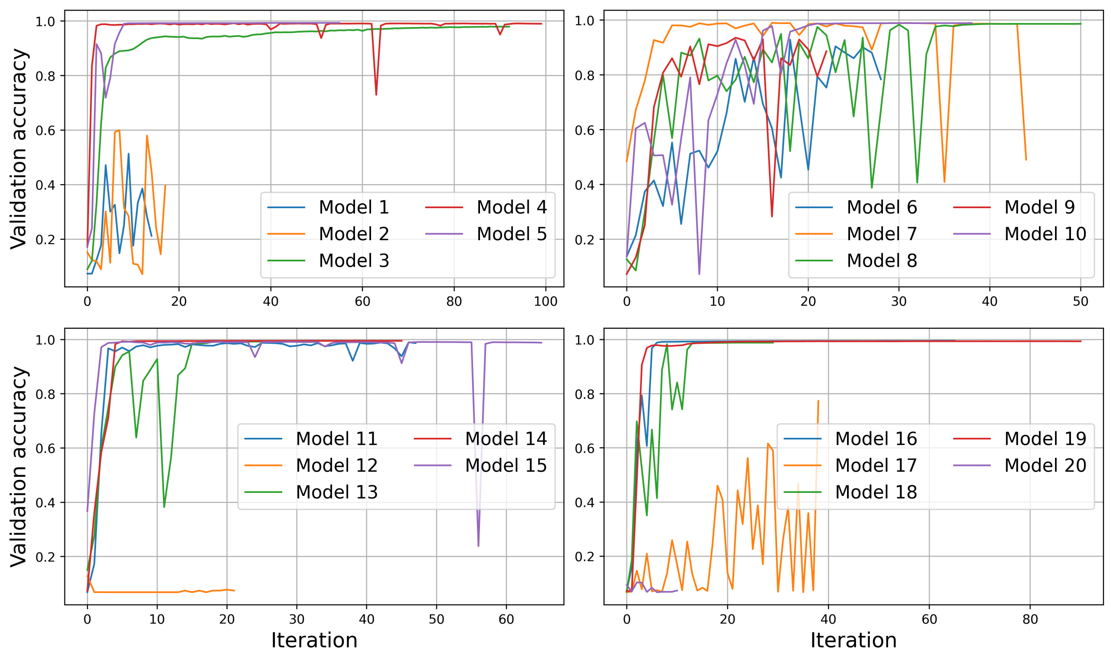


Figure 6.2: Performance on validation set

Successively, the number of parameters have been considered (tab. 6.3). Here, only models n. 3, 5, 14, 16 and 19 have number of variables less than 1'000'000 (1 million). Between these five, the best model has been chosen as the one with higher validation accuracy.

STEP 7 - Selection of the model

Following all previous considerations, the "InceptionTime" model (number 16) as been considered as the final best model.

After the model has been selected, another test has been performed: it has been checked if the model overfit or underfit the data, by looking at the loss functions.

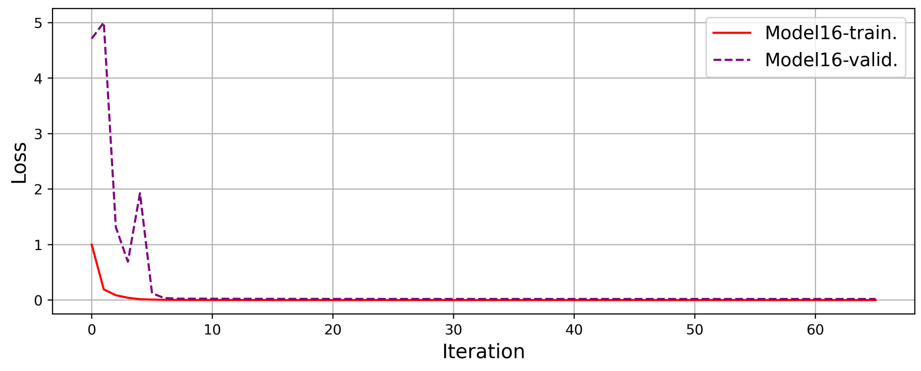


Figure 6.3: Comparison of training and validation loss of model 16

In fig. 6.3, the training loss and validation loss both decrease and stabilize at a specific point, which means model 16 optimally fit the data.

Now that the model has been identified, its final shape is reported in fig. 6.4, and its hyperparameters are reported in the following:

$$'learning\_rate' : \ 0.00264728226737, \ 'regularization\_rate' : \ 0.0745954647986,$$
$$'network\_depth' : \ 3, \ 'filters\_number' : 55, \ 'max\_kernel\_size' : \ 46$$

STEP 8 - Longer training of the best model

To maximize the final selected model performance, it has been trained on more data (removing the subset size) and with more epochs.

```
#Making a copy of the model and training with the whole dataset
best_model_index = 16 - 1 #-1 to get its index
_,_,_ = train_models_on_samples(X_train, y_train_enc,
                                X_val, y_val_enc,
                                [models_2nd[best_model_index]],
                                nr_epochs=200,
```

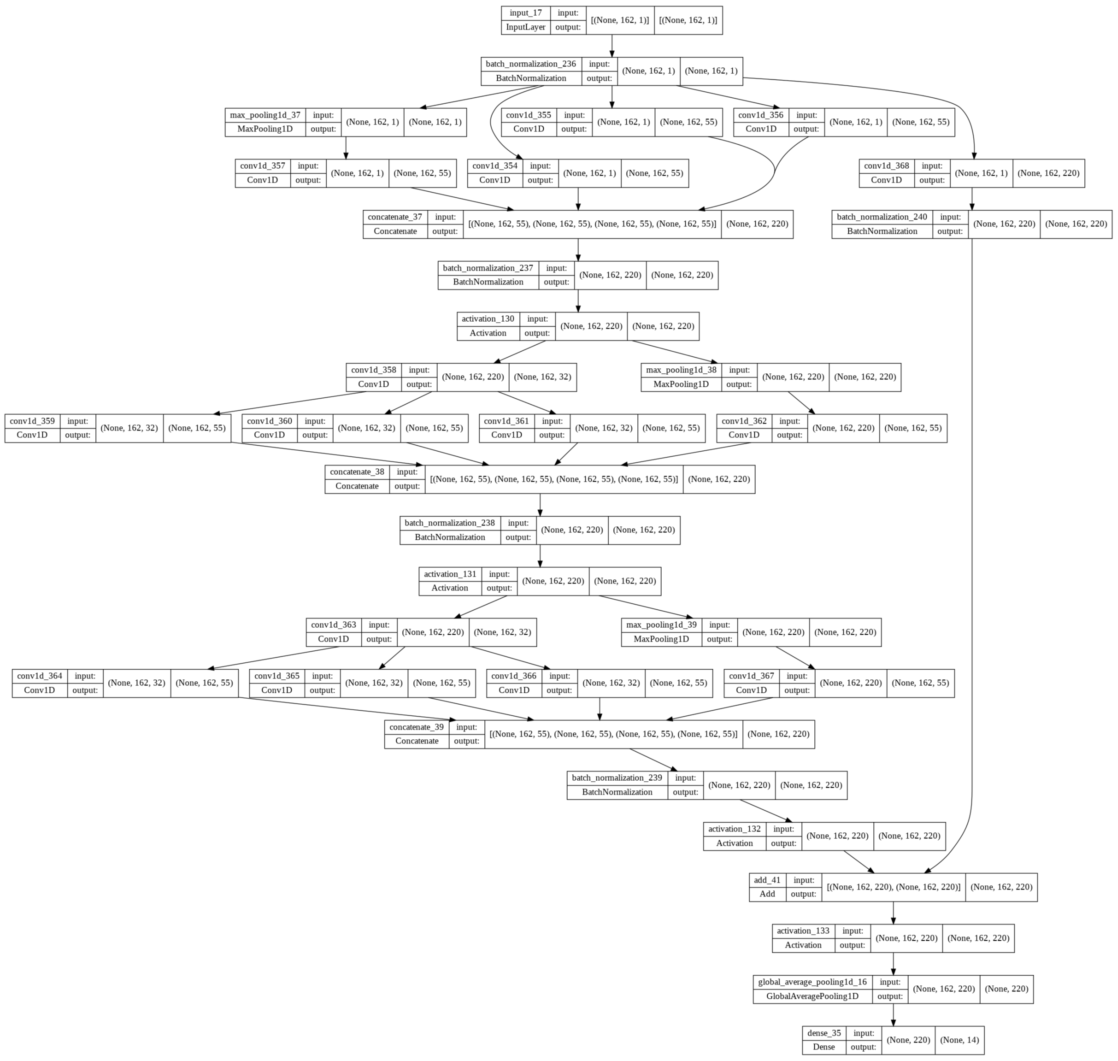

input_17 | input: | output: | [(None, 162, 1)] | [(None, 162, 1)]
InputLayer
batch_normalization_236 | input: | output: | (None, 162, 1) | (None, 162, 1)
BatchNormalization
max_pooling1d_37 | input: | output: | (None, 162, 1) | (None, 162, 1)
MaxPooling1D
conv1d_355 | input: | output: | (None, 162, 1) | (None, 162, 55)
Conv1D
conv1d_356 | input: | output: | (None, 162, 1) | (None, 162, 55)
Conv1D
conv1d_357 | input: | output: | (None, 162, 1) | (None, 162, 55)
Conv1D
conv1d_354 | input: | output: | (None, 162, 1) | (None, 162, 55)
Conv1D
conv1d_368 | input: | output: | (None, 162, 1) | (None, 162, 220)
Conv1D
concatenate_37 | input: | output: | [(None, 162, 55), (None, 162, 55), (None, 162, 55), (None, 162, 55)] | (None, 162, 220)
Concatenate
batch_normalization_240 | input: | output: | (None, 162, 220) | (None, 162, 220)
BatchNormalization
batch_normalization_237 | input: | output: | (None, 162, 220) | (None, 162, 220)
BatchNormalization
activation_130 | input: | output: | (None, 162, 220) | (None, 162, 220)
Activation
conv1d_358 | input: | output: | (None, 162, 220) | (None, 162, 32)
Conv1D
max_pooling1d_38 | input: | output: | (None, 162, 220) | (None, 162, 220)
MaxPooling1D
conv1d_359 | input: | output: | (None, 162, 32) | (None, 162, 55)
Conv1D
conv1d_360 | input: | output: | (None, 162, 32) | (None, 162, 55)
Conv1D
conv1d_361 | input: | output: | (None, 162, 32) | (None, 162, 55)
Conv1D
conv1d_362 | input: | output: | (None, 162, 220) | (None, 162, 55)
Conv1D
concatenate_38 | input: | output: | [(None, 162, 55), (None, 162, 55), (None, 162, 55), (None, 162, 55)] | (None, 162, 220)
Concatenate
batch_normalization_238 | input: | output: | (None, 162, 220) | (None, 162, 220)
BatchNormalization
activation_131 | input: | output: | (None, 162, 220) | (None, 162, 220)
Activation
conv1d_363 | input: | output: | (None, 162, 220) | (None, 162, 32)
Conv1D
max_pooling1d_39 | input: | output: | (None, 162, 220) | (None, 162, 220)
MaxPooling1D
conv1d_364 | input: | output: | (None, 162, 32) | (None, 162, 55)
Conv1D
conv1d_365 | input: | output: | (None, 162, 32) | (None, 162, 55)
Conv1D
conv1d_366 | input: | output: | (None, 162, 32) | (None, 162, 55)
Conv1D
conv1d_367 | input: | output: | (None, 162, 220) | (None, 162, 55)
Conv1D
concatenate_39 | input: | output: | [(None, 162, 55), (None, 162, 55), (None, 162, 55), (None, 162, 55)] | (None, 162, 220)
Concatenate
batch_normalization_239 | input: | output: | (None, 162, 220) | (None, 162, 220)
BatchNormalization
activation_132 | input: | output: | (None, 162, 220) | (None, 162, 220)
Activation
add_41 | input: | output: | [(None, 162, 220), (None, 162, 220)] | (None, 162, 220)
Add
activation_133 | input: | output: | (None, 162, 220) | (None, 162, 220)
Activation
global_average_pooling1d_16 | input: | output: | (None, 162, 220) | (None, 220)
GlobalAveragePooling1D
dense_35 | input: | output: | (None, 220) | (None, 14)
Dense

Figure 6.4: Best model network

```
                                    subset_size= None,
                                    early_stopping_patience = 10)
```

After the final training has been completed, the model reached its final shape, containing the best computed weights.

STEP 9 - Model validation

The validation of the model consist in classifying data that have never been used by the network, in our case this correspond to use the testing subset. The testing accuracy can be computed by comparing the predicted output with true label.

Below, the code used to compute the testing accuracy.

```
datasize = X_test.shape[0]

#Evaluating the model on the test data
results = best_model.evaluate(X_test, y_test_enc, batch_size = None)
print("Test loss, Test accuracy: ", results)

# Generate prediciton (probabilities of the last layer)
probs = best_model.predict(X_test[:datasize,:,:], batch_size=1)
y_test_pred = probs.argmax(axis = 1)
```

The obtained results are:

$$Test\ loss:\ 0.0187443308532,\ Test\ accuracy:\ 0.9954163432121$$

This result established that the classification of a time-series using Deep Learning algorithm is possible with high precision.

Chapter 7

Classification results

When using Machine Learning and Deep Learning algorithms, a very important aspect is to validate the models in different scenarios, as closer as possible to the real application. For this reason, even if the obtained DL model (InceptionTime) reached a test accuracy of 99.5%, other test have been performed.

Three different scenario have been created:

CASE A - Same setup used in chap. 6, that is:

- **Training:** 70% of dataset R1
- **Validation:** 20% of dataset R1
- **Testing:** 10% of dataset R1

CASE B - With this scenario, the training and validation subsets remain unchanged, while the testing has been extracted from dataset R2, which has never been used before:

- **Training:** 70% of dataset R1
- **Validation:** 20% of dataset R1
- **Testing:** dataset R2

It has been decided keep training and validation subsets as they are in CASE A, such that it's not needed to retrain the network.

CASE C - In the last case the dataset R1 and R2 have been joined and shuffled, so to have a new bigger dataset. From it, training and validation subsets have been extracted, while the testing has been obtained from dataset R3, which has never seen from the network:

- **Training:** 70% of the new dataset (R1 + R2)

- **Validation:** 20% of the new dataset (R1 + R2)
- **Testing:** dataset R3

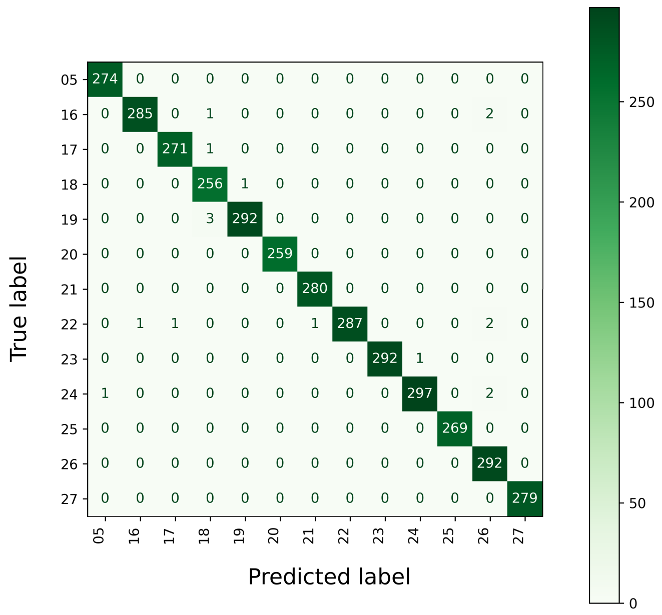
In this situation, two positive aspect can be highlight. First, the dataset is bigger, it contains more samples, so that the network have more information to analyze. Then, the training process uses dataset coming from two different measurements time instant, obtaining a dataset which have samples less related to each other.

Successively, also the ML algorithms explained in sec. 2.4 have been exploited to highlight their performances between all the 3 scenarios.

Let start analyzing the result of the obtained DL best model, InceptionTime-16.

7.1 InceptionTime-16 results

Once the mcfly's process has been concluded, the tested accuracy reached almost 100%, meaning that only few samples have been misclassified. In fig. 7.1 is shown its confusion matrix, which give a graphical interpretation on the wrongly predicted point.



	05	16	17	18	19	20	21	22	23	24	25	26	27
05	274	0	0	0	0	0	0	0	0	0	0	0	0
16	0	285	0	1	0	0	0	0	0	0	0	2	0
17	0	0	271	1	0	0	0	0	0	0	0	0	0
18	0	0	0	256	1	0	0	0	0	0	0	0	0
19	0	0	0	3	292	0	0	0	0	0	0	0	0
20	0	0	0	0	0	259	0	0	0	0	0	0	0
21	0	0	0	0	0	0	280	0	0	0	0	0	0
22	0	1	1	0	0	0	1	287	0	0	0	2	0
23	0	0	0	0	0	0	0	0	292	1	0	0	0
24	1	0	0	0	0	0	0	0	0	297	0	2	0
25	0	0	0	0	0	0	0	0	0	0	269	0	0
26	0	0	0	0	0	0	0	0	0	0	0	292	0
27	0	0	0	0	0	0	0	0	0	0	0	0	279

Figure 7.1: InceptionTime-16's confusion matrix on testing subset - case A

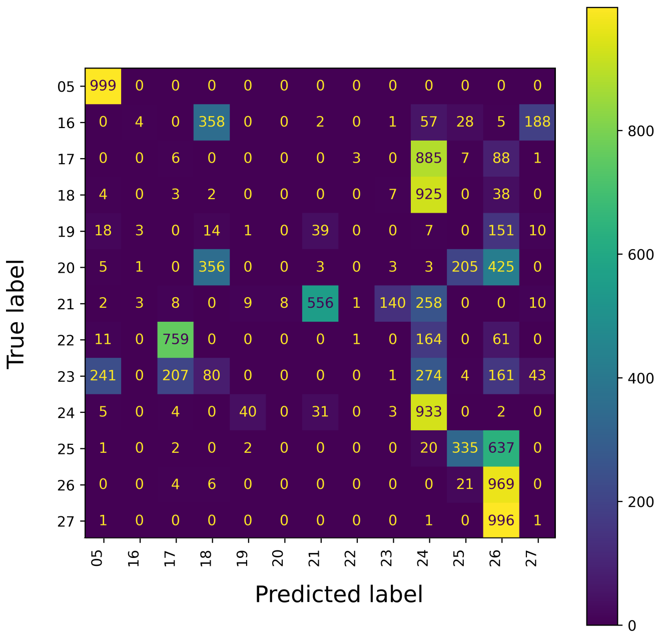


Figure 7.2: InceptionTime-16's confusion matrix on testing subset - case B

Based on the reached accuracy, it is predictable that only few point have been mislabelled. For example, point 22 has been correctly predicted 287 times, while it has been classified as point number 16, 17, 21, and 26 only few time, with an overall misclassification of 5 samples.

At a first sight, it could sound like a good result. Unfortunately, this is not the case. Indeed, when using Machine Learning algorithms, have a minimum error is a necessary condition to prove the correct operating. Too high accuracy could mean overfitting.

Moreover, it is also important to remember how that value of accuracy has been achieved. Indeed, when training the networks dataset R1 has been used and, from it, the three subsets (training, validation and testing) have been extracted. This means that the used samples for testing the network were unseen, but still coming from the same measurement campaign (CASE A).

When localizing a robot in an indoor environment, measures come from different time instant, being completely unrelated to the samples used in the training model. In this regard, and trying to verify the robustness to overfitting, it has been decided to move forward testing the model with the second scenario, CASE B. The obtained

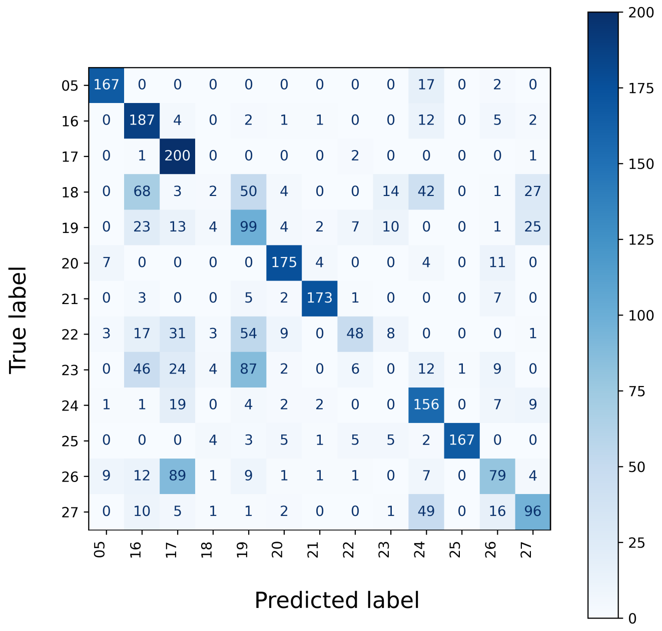


Figure 7.3: InceptionTime-16's confusion matrix on testing subset - case C

results have been:

$$Test\ loss:\ 12.27202606201,\ Test\ accuracy:\ 0.27134102582$$

Here, the first contradictory result. The testing accuracy drops from 99.5% to 27.1%, which means that when using data never seen before and coming from a different time instant, the model is no more able to predict correctly the position. For a wider understanding, in fig. 7.2 is shown its confusion matrix. From it, it is possible to see that almost all the samples have been wrongly classified, for example, most of them have been labelled as 26. Point 5 is the only one with all the sample correctly classified. This is a reasonable result since it is located in front of the elevator and slightly in NLOS, leading to a different CIR's shape.

After this poor result, it has been decided to understand which could be the effective reason of the degradation. The dataset have been collected in different time instant, but the surrounding conditions were pretty similar, which could not be the cause of it. The main problem could be that too few CIRs have been used for the training, leading the network to overfit the data. Indeed, using thousands of data for each point is not satisfactory. Most of the powerful networks used in big companies have been trained with millions, if not billions, of data before getting good results.

Unfortunately, since the acquirement process for a single dataset composed of thousand data requires different days, it has not been able to collect much more

CIRs. Only a fourth dataset (R3) has been collected in order to train the model with more data and still test it with unrelated dataset.

Keeping the InceptionTime-16 as it is, it has been merged dataset R1 and R2, and from it, the training and validation subsets have been extracted (CASE C). The new dataset R3, instead, has been used for testing the network. The obtained results have been:

$$Test\ loss:\ 4.26345968246,\ Test\ accuracy:\ 0.59645742177$$

with its relative confusion matrix in fig. 7.3.

With this small variation, accuracy almost double, increasing from 29.1% to 59.96%. This result confirm the idea that the network was previously overfitting a small dataset. So, to reach high performances with the selected network, more dataset must be collected in different moments and used in the training.

As mentioned before, it has not been possible in this thesis, but it is the purpose of the continuation of it.

7.2 ML results

Machine Learning algorithms are typically lighter than Deep Learning's one. On the other hand, the latter, are usually better performing and more robust. The goal of this thesis is to localize an autonomous robot in an indoor environment, which does not constraint on which network must be used. In this regard, an overview also on classical ML algorithms have been conducted.

To summarize the discussion, it has been decided to not include all the executed steps but only the main results, which are reported in tab. 7.1. As for the InceptionTime-16, all the model have been analyzed using three different scenarios (CASE A, CASE B and CASE C).

At first sight, all the 6 selected model performed optimally in CASE A, with testing accuracy always higher than 98%, with exception of "Decision Tree" model, which achieve 93.86% (that is still a good accuracy result). Performances completely change in CASE B (as it happened for InceptionTime-16). The testing accuracy reduces by 8 times, which means that there is not a single network able to localize properly when using a dataset collected in a different time instant.

Luckily, training the models with different datasets and an higher number of samples leads to achieve better results. Almost all the networks doubled their testing accuracy, with the highest percentage reached by the "Random Forest"

Table 7.1: ML model's performances, testing in 3 different scenarios

Model type	**Testing accuracy [%]**		
	Case A	**Case B**	**Case C**
Logistic Regression	98.22	11.58	29.46
Decision Tree	93.86	9.94	22.80
Random Forest	99.41	20.06	47.55
Gradient Boosting	99.31	16.82	41.43
K-nearest neighbor	99.67	14.98	42.29
Support Vector Machine	99.19	12.54	38.70

model (47.55%). On the other hand, "Logistic regression" and "Decision Tree" have been the models which obtained the lowest accuracy (29.46% and 22.80%, respectively).

Of course, it does not mean that all the other properly become acceptable for the localization. Indeed percentage remains below 50% which is still completely unacceptable for the project's purpose. To understand if all these model are suitable for the indoor localization, millions of more unrelated samples are needed.

Instead, regarding the "K-nearest neighbor" model, it is worth to highlight that the selected accuracy is the highest one. Indeed, this accuracy percentage has been achieved exploiting the number of neighbors points from 1 to 50 (fig. 7.4) and it has been selected the one with higher performance.

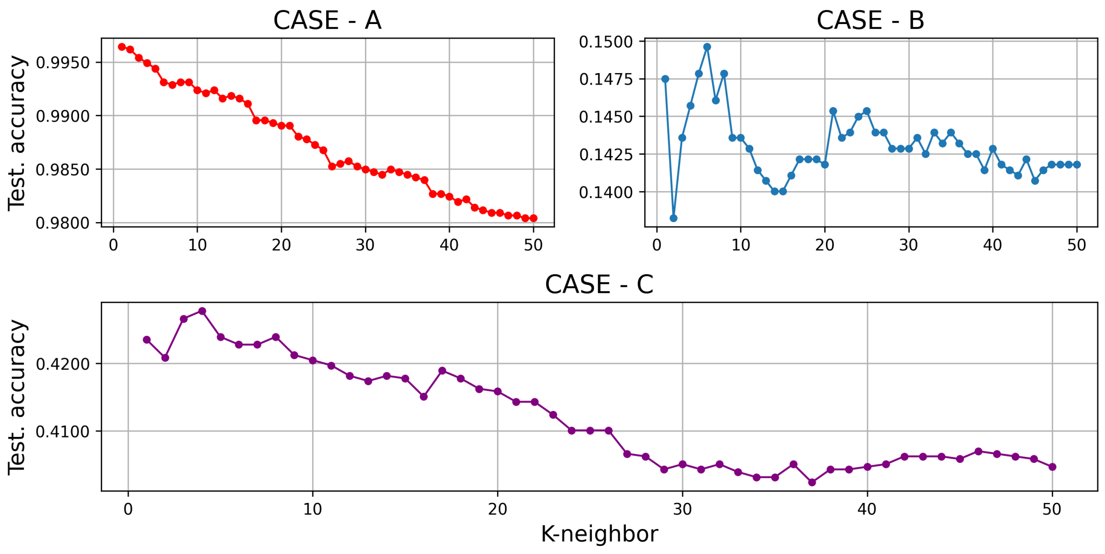


Figure 7.4: KNN's performances for the three cases

For example, the best testing accuracy are achieved picking: $n_{neighbor} = 2$ in CASE A , $n_{neighbor} = 5$ in CASE B and $n_{neighbor} = 4$ in CASE C.

Conclusions

This research has been aimed at exploring a new approach for indoor localization, trying to understand if, given a reference transmitter, the different points in an indoor environment could be characterized with a sort of "electromagnetic signature". The central questions considered in the thesis were:

1. Is it possible to characterize a point in a room using its complex impulse response (CIR) (or, more specifically, its amplitude)?

2. If an "electromagnetic signature" exists, could it be identified using AI techniques?

To determine the answers to these questions, two UWB transceivers have been used to perform CIR measurements in an indoor environment.

They have been configured to work as Anchor and Transmitter and, after having placed them on a wall and on a cart, respectively, a measurement campaign has been performed. Thousands of samples have been collected for every point, spaced roughly 1 m from each other, in front of the elevator area and in the hallway of the third floor of the Engineering building at the California State University of Los Angeles (CSULA). Several types of data have been collected, including ToF, SNR, and others, but only the CIR (that is the impulse response of the channel in the time domain) has been used to effectively perform the localization within this thesis.

Once different datasets had been collected, they have been analyzed and processed. From the preliminary analysis it has been observed that the CIR's shape is indeed related to the surrounding environment, and that moving from one point to another the CIR shape changes. It has however also been possible to observe that the changes are not only related to the position itself and to the fixed objects contained in the environment, but also to the non-static elements of the environment, such as people and mobile objects, that can create a very rich statistics.

Successively, the time-series data have been processed. CIRs associated to the same point have been shifted in order to align the starting point, with aim to increase the robustness of the dataset. Then, they have been normalized to avoid possible problem (e.g., vanishing gradient) in the Machine Learning and Deep Learning networks.

Once the input have been properly processed and organized in structures compliant with neural networks, the focus has been moved on the network itself. New questions have arisen: "Which network should be used?", "How to set the parameter to get the best performances?". In order to avoid mistakes due to inexperience, it has been decided to follow a new and different approach, called autoML.

The aim of autoML is to research some well-performing Machine Learning models or algorithms using an automated choice of hyperparameters. In this regard, the software mcfly has been employed, which focuses on Deep Learning algorithms for time series classification. Four different networks have been examined with this methodology: CNN, DeepConvLSTM, ResNet and InceptionTime. The hypeparameters have been optimized with a random search approach, that is finding the optimal value via random selection of them.

After an initial analysis, the DeepConvLSTM network has been the only model not achieving sufficiently result in the classification accuracy, and it has therefore been discarded. Successively, also CNN and ResNet have been eliminated since InceptionTime was the model with best trade off between testing accuracy and number of parameters.

To validate the model, three different scenarios have been considered, denoted respectively as CASE A, CASE B and CASE C. In the first one, training, validation and testing subsets all belong to the same dataset (named R1). In the second one, instead, only the testing subset has been modified with respect to CASE A, using data from R2. Finally, in CASE C, dataset R1 and R2 have been joined and shuffled, using them for training and validating the model, while a third dataset, R3, has been used to form the testing subset.

With these 3 different scenarios, both InceptionTime and Machine Learning models (e.g., Logistic Regression, Decision Tree, Random Forest, Gradient Boosting and SVM) have been tested.

Different results have been achieved in the 3 cases. All the model get high testing accuracy in CASE A. Indeed, the worse performing model has been the "Decision Tree" with test accuracy of 93.86%. Results completely change in CASE B. In this scenario, both InceptionTime and all the ML algorithms drop their performance by 8 times, obtaining testing accuracy in the range between 9.94% and 27.13%. These are very low percentages, which indicate that localization is impossible when using samples coming from a set of measurements performed in different conditions and never seen before by the network. Since this would be the final real application (a robot using real-time samples, coming from a measurement not belonging to the training set), it has been decided to find out more about the cause of problems. In this regard, a new dataset has been collected and the previous two have been used together in the training phase (CASE C). As expected, increasing the number of samples and training the model with data that are more unrelated to each other, increases the testing accuracy by a factor of 2. The best performance model has been the InceptionTime with a testing accuracy of 59.65%. Similarly, the same behavior have been followed by all the Machine Learning algorithms. The main difference has been the accuracy itself. Indeed, the model who performs better between all the ML algorithm has been the "Random forest" with a testing accuracy of 47.55%.

Of course the final computed accuracies remain low for a real localization goal. But thanks to this analysis, it has been clarified that, given the very rich statistics of the problem at hand, the collection of the samples is one of the most important contributions in order to have realistic Machine Learning models. Indeed, the main problem in our datasets, could have been having CIRs that are too similar when coming from the same dataset and when testing with a different dataset (measured in different conditions), samples looks random to the network. In other words, the different indoor locations have probably a distinctive “electromagnetic signature”, but the problem has an extremely rich statistical variability, that has not fully been captured by the limited measurement campaign that we were able to perform during the thesis development.

To better identify the “electromagnetic signature” of the environment, further studies are being performed, collecting larger dataset in different environmental conditions.

These conclusion are also verified by a parallel study were better performance have been obtained using a larger dataset (obtained not with additional measurements but with data augmentation techniques). Moreover, additional studies are also trying to improve the classification performance by exploiting the additional data (such as the ToF and SNR values), as well as considering larger indoor areas and datasets with finer spatial resolution.

Bibliography

[1] Michal Szczypior. *Ultra-wideband - old technology discovered again.* (`https://sii.pl/blog/en/ultra-wideband-old-technology-discovered-again-vol-1-uwb-potential/`). 2021 (cit. on p. 1).

[2] Jorge R Fernandes and David Wentzloff. *Recent Advances in IR-UWB - Transceivers - An Overview.* (`https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5537916`). 2022 (cit. on p. 1).

[3] QORVO. *Getting Back to Basics with Ultra-Wideband.* 2021 (cit. on pp. 2, 5).

[4] Mehrpouyan Hani. *UWB and UWB Channels.* Sept. 2022 (cit. on p. 3).

[5] 3db. *IMPULSE RADIO UWB PRINCIPLES AND REGULATION.* Nov. 2019 (cit. on p. 4).

[6] Ridolfi, Matteo and Velde, Samuel and Steendam, Heidi and De Poorter, Eli. «Analysis of the Scalability of UWB Indoor Localization Solutions for High User Densities». In: *Sensors* 18 (June 2018) (cit. on p. 7).

[7] Linfu Duan Yan Zhang. «A phase-difference-of-arrival assisted ultra-wideband positioning method for elderly care». In: *Measurement* 170 (2021) (cit. on p. 8).

[8] Sirikarn Woracheewan, Changhui Hu, Rahul Khanna, J. Nejedlo, Huaping Liu, and Patrick Chiang. «Measurement and characterization of ultra-wideband wireless interconnects within active computing systems». In: May 2011, pp. 1–4 (cit. on p. 10).

[9] Benjamin Matthews, Sven Ole Schmidt and Horst Hellbruck. *Understanding and Prediction of Ultra-Wide Band Channel Impulse Response Measurements.* 2019 (cit. on p. 11).

[10] Wikipedia contributors. *Signal-to-noise ratio — Wikipedia, The Free Encyclopedia.* [Online; accessed 27-September-2022]. 2022. URL: `https://en.wikipedia.org/w/index.php?title=Signal-to-noise_ratio&oldid=1104835005` (cit. on p. 12).

[11] Wikipedia contributors. *Time of flight — Wikipedia, The Free Encyclopedia.* [Online; accessed 27-September-2022]. 2022. URL: `https://en.wikipedia.org/w/index.php?title=Time_of_flight&oldid=1105580682` (cit. on p. 12).

[12] Decawave. *EVK1000 USER MANUAL.* 2016 (cit. on p. 13).

[13] Decawave. *DW1000 User Manual.* 2017 (cit. on pp. 13, 18, 50).

[14] IEEE 802.15.4-2011 or "IEEE Std 802.15.4™-2011" (Revision of IEEE Std 802.15.4-2006). *IEEE Standard for Local and metropolitan area networks – Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs).* 2011 (cit. on pp. 14, 15).

[15] Decawave. *Understanding and using the DecaRanging ranging demo (PC) application.* 2016 (cit. on pp. 22, 23).

[16] Christian Janiesch, Patrick Zschech & Kai Heinrich. *Machine learning and deep learning.* 2021. URL: `https://doi.org/10.1007/s12525-021-00475-2` (cit. on pp. 27, 28).

[17] Timothy P Lillicrap and Adam Santoro. «Backpropagation through time and the brain». In: *Current Opinion in Neurobiology* (2019). Machine Learning, Big Data, and Neuroscience. ISSN: 0959-4388. URL: `https://www.sciencedirect.com/science/article/pii/S0959438818302009` (cit. on p. 30).

[18] Siddharth Sharma, Simone Sharma and Anidhya Athaiya. *ACTIVATION FUNCTIONS IN NEURAL NETWORKS.* 2020 (cit. on p. 30).

[19] Xavier Glorot and Yoshua Bengio. *Understanding the difficulty of training deep feedforward neural networks.* 2010 (cit. on p. 43).

[20] AJAY SHRESTHA and AUSIF MAHMOOD. *Review of Deep Learning Algorithms and Architectures.* 2019 (cit. on p. 44).

[21] Decawave. *DW1000 DEVICE DRIVER APPLICATION PROGRAMMING INTERFACE (API) GUIDE NETWORKS.* 2016 (cit. on p. 55).

[22] D. van Kuppevelt, C. Meijer, F. Huber, A. van der Ploeg, S. Georgievska, V.T. van Hees. *Mcfly: Automated deep learning on time series.* 2020 (cit. on pp. 75, 76).

[23] James Bergstra and YoshuRandom Search for Hyper-Parameter Optimization. 2012 (cit. on p. 76).