UNIVERSITY OF BERN

BACHELOR THESIS

A Deep Learning Approach for Indoor Localization

Author:
Jonas FURRER

Supervisors:
Jose CARRERA,
Zhongliang ZHAO

Head of Research
PROFESSOR DR. TORSTEN BRAUN
Communication and Distributed Systems
Institute of Computer Science

June 21, 2020

Declaration of Authorship

I, Jonas Furrer, declare that this thesis titled “A Deep Learning Approach for Indoor Localization” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signature: ____________________

Date: ________________________

UNIVERSITY OF BERN

Institute of Computer Science

Bachelor of Science in Computer Science

A Deep Learning Approach for Indoor Localization
by Jonas Furrer

Abstract

Real-time localization systems become increasingly important due to the rapid growth of context-aware services and the Internet of Things (IoT). As there are already accurate approaches for outdoor environments, current research focuses on indoor localization. Most of the works in this field implement a system that runs the localization algorithms on the target device. However, this approach has to deal with limited computational resources of target devices (mostly smartphones). Thus, we introduce a Multi-Access Edge Computing (MEC) based localization system that allows offloading the heavy computations to an external server at the edge of the network. Our proposed localization algorithm achieves high room-level localization accuracy by fusing Wi-Fi and Ultra-Wideband (UWB) Received Signal Strength Indicator (RSSI), Earth Magnetic Field (EMF) values and environmental information in a combined Deep Learning model consisting of a Convolutional Neural Network (CNN) and a Recurrent Neural Network (RNN). We conduct extensive experiments in an office-like indoor environment along different trajectories. Evaluation results show that our system can achieve room-level prediction accuracy of 98.52% and, therefore, outperforms existing approaches in this field.

Contents

List of Figures

List of Tables

List of Abbreviations

AI Artificial Intelligence.

AN Anchor Node.

AOA Angle Of Arrival.

API Application Programming Interface.

CNN Convolutional Neural Network.

CSI Channel State Information.

DFL Device Free Localization.

DL Deep Learning.

DNN Deep Neural Network.

DoD Department of Defense.

DTDOA Differential Time Difference Of Arrival.

EMF Earth Magnetic Field.

FCC Federal Communications Commission.

FN False Negatives.

FNN Feedforward Neural Network.

FP False Positives.

GP-LVM Gaussian Process Latent Variable Model.

GPS Global Positioning System.

HDE Heuristic Drift Elimination.

HTTP Hyptertext Transfer Protocol.

IMU Inertial Measurement Unit.

IoT Internet of Things.

IP	Internet Protocol.
KNN	K-Nearest Neighbours.
LDPL	Log-Distance Path Loss.
LOS	Line-Of-Sight.
LSTM	Long Short-Term Memory.
MEC	Multi-Access Edge Computing.
ML	Machine Learning.
MR	Mobile Robot.
NLP	Natural Language Processing.
PDR	Pedestrian Dead Reckoning.
PHY	Physical Layer.
PRF	Pulse Repetition Frequency.
QoE	Quality of Experience.
RFID	Radio Frequency Identification.
RNN	Recurrent Neural Network.
RP	Raspberry Pi.
RSS	Received Signal Strength.
RSSI	Received Signal Strength Indicator.
TAG	Target Device.
TCP	Transmission Control Protocol.
TDOA	Time Difference Of Arrival.
TOA	Time Of Arrival.
TP	True Positives.
UWB	Ultra-Wideband.

Chapter 1

Introduction

In the past few years, the rising importance of context-aware applications and the growing ubiquitousness of the Internet of Things (IoT) have increased the attention in indoor location-based services. Context-aware systems are able to adapt their behaviour to the current environmental context. When it comes to using mobile devices, the most frequently used attribute of context is their current location [1]. Therefore, accurate real-time positioning systems are required. In outdoor environments, the Global Positioning System (GPS) is an effective and accurate approach. However, satellite based navigation systems suffer from positioning errors in indoor scenarios (e.g., complex buildings). GPS receivers determine their geo-positions using triangulation of radio signals transmitted by satellites. Due to the missing Line-Of-Sight (LOS) propagation between the satellites and the GPS receivers in indoor environments the travel time of the transmitted signal is affected and this can lead to a significant location estimation error [2].

Thus, alternative technologies are needed that provide higher accuracy indoors. Although there are a lot of theoretical and experimental works in this field there is not a simple and accurate solution for indoor positioning so far. So, indoor positioning can still be considered as an open challenging problem [3].

Radio-based positioning is one of the most widely used approaches for indoor localization due to the availability of a large variety of radio signals, such as Wi-Fi, Bluetooth, magnetic field, sound, light, etc. in indoor environments [4]. Wi-Fi signals are often used because of the advantage of reusing an already available infrastructure (wireless networks are available in most corporate environments and commercial buildings, such as universities, airports, restaurants, hospitals, etc.) [2]. Most of the existing radio-based indoor localization approaches use Wi-Fi Received Signal Strength Indicator (RSSI) as radio parameter. However, one problem of RSSI is that it suffers from the multipath effect. This means that there are multiple signals (not just the LOS signal) propagating through different paths from the transmitter to the receiver. This undesirable effect is even more severe in indoor environments due to the presence of different obstacles such as walls, ceiling, floor and furniture [5]. Additionally, the signal propagation is affected by several environmental factors such as the presence of people in the area of interest, relocation of furniture, temperature and humidity variations and opening and closing doors [6]. Thus, the RSSI is dependent on distance and environmental factors. The fluctuation of RSSI leads

to undesirable localization errors, which result in considerable reduction of the estimation accuracy [2]. In addition to Wi-Fi radio signals, the Earth Magnetic Field (EMF) is often used for localization purposes. Due to the presence of ferromagnetic materials the EMF distorts over space. These distortion patterns can also be used to estimate indoor positions [4].
Compared to Wi-Fi signals, Ultra-Wideband (UWB) has recently attracted attention due to its capabilities of reducing the localization errors to lower than one meter [7]. The main feature of the UWB technology is the use of a large frequency band (more than 500 MHz) and short-pulse, low-powered radio signals. In addition, UWB radios are able to distinguish pulses that are reflected from different objects what makes it robust to multipath effects. Therefore, UWB radios are often used for accurate indoor localization [8].

A large number of indoor positioning systems are running on client devices with limited computational power. Offloading the heavy computations to a centralized server could be a solution to this problem. However, the data transmission between the client device and the server could lead to increased latency and unreliable performance which is not tolerable for real-time applications [7]. An alternative solution is the Multi-Access Edge Computing (MEC) technology [9], which brings cloud computing capabilities to the edge of the network to satisfy application requirements of short latency and high resource-demanding. By applying the MEC paradigm, heavy computations are offloaded from the client devices to a near edge server. Due to the short device-server distance the data transmission time can be reduced and, therefore, the performance of the indoor localization system can be improved.

In this work, we consider the question *"What will be the impact of applying Deep Learning (DL) algorithms to indoor localization and how can the open technical challenges in this field be tackled?"*. Thus, we present a novel room-level localization approach by fusing Wi-Fi RSSI, UWB RSSI, EMF readings and environmental information in a discriminative DL model. To include information about previous positions of the client device we apply a Recurrent Neural Network (RNN) using Long Short-Term Memory (LSTM). In order to overcome the temporal and spatial instability of RSSI, we apply a Convolutional Neural Network (CNN) to filter out the environmental noise. Our system is implemented based on the paradigm of MEC. Thus, the localization algorithms are implemented in an external server at the edge of the network. We prototype our approach on Raspberry Pi (RP) both as client device and anchor nodes. To validate our indoor localization system, we conduct extensive experiments in a complex indoor environment along several trajectories. Evaluation results show that our proposed approach can achieve a prediction accuracy of 98.52%.

The main contributions of this work are summarized as follows:

- We propose a Deep Learning approach that fuses Wi-Fi and UWB RSSI, EMF readings and environmental information for room recognition.
- We implement and evaluate a MEC-based positioning system that runs our proposed localization algorithms on an external server.

- We conduct a set of extensive experiments in a real complex indoor scenario along several trajectories. Our proposed room recognition method can achieve 98.52% accuracy, which outperforms existing approaches in this field.

The rest of our work is organized as follows. Chapter II presents some related work in the field of indoor localization and introduces the background theory. The architecture of our proposed room recognition system is reviewed in Chapter III. Chapter IV shows the implementation details of the MEC-based localization system. In Chapter V the performance evaluation results are discussed and finally Chapter VI concludes the work.

Chapter 2

Related Work and Theoretical Background

Indoor positioning has been investigated for many years and became a hot topic for research as well as industry. Many solutions for accurate indoor positioning systems have been proposed. Most of the work in this field focus on inertial sensor-based tracking and radio-based positioning.

2.1 Overview of Indoor Localization Methods

As described in [10], previous indoor positioning research can be divided into two categories, 1) Pedestrian Dead Reckoning (PDR) based on Inertial Measurement Unit (IMU) and 2) Radio-based positioning.

2.1.1 Pedestrian Dead Reckoning (PDR)

With the fast development of mobile devices (over 5 billion of mobile users today [11]), PDR relying on IMUs has evolved into a widely used technique for indoor localization. PDR systems use inertial sensors in smartphones, e.g., accelerometer, magnetometer and gyroscope, to calculate the relative movement of the target by detecting steps, estimating stride length and heading orientation [10]. The position is incrementally estimated from a given starting point [12] by integrating the relative movement at sequential time intervals.

In [13], the authors propose an indoor tracking system that is decomposed into a stride length estimation part and a heading determination part. The stride length is calculated by using accelerometer readings whereas the heading direction is estimated by means of the gyroscope as it is much less noisy than accelerometers or magnetometers and relatively immune to environmental disturbances. A similar approach can be found in [14]. The stride determination is based on walking speed, walking frequency and acceleration magnitude. In order to estimate the heading direction, they use a combination of gyroscope and magnetic compass. In this integrated system, the gyroscope is able to correct the magnetic disturbances while the compass can compensate for the bias of the gyroscope and the initial orientation. The authors of [15] proposed a method named Heuristic Drift Elimination (HDE) that extends their previous work [16] so that the accumulated errors due to sensor drift are eliminated. This leads to near-zero heading errors. Nevertheless, HDE requires additional hardware (sensors on the foot of the pedestrian). In another work [12], the authors constructed a PDR system in which accelerometer readings and magnetometer readings are fused to detect steps and estimate the moving direction as well as the stride length. For the stride length estimation, they use different models for walking mode and for running mode, respectively. It is assumed that the user's activity mode is continuously classified into "walking", "running", and "stationary". In [17], the authors apply an accelerometer-based walking distance estimation. Gyroscope measurements are used to detect the physical turns made by the user.

As PDR systems estimate relative movements instead of absolute positions, that results in an accumulation of sensor errors over time which is a significant drawback of this approach. For that reason, PDR positioning systems must consider integrating additional information such as Wi-Fi signals or floor plan constraints to compensate for these errors.

2.1.2 Radio-based Positioning

In contrast to PDR that measures relative position changes, radio-based positioning systems estimate the absolute position of targets in a coordinate system [10]. Due to its ubiquitous availability, radio signals are often used in indoor positioning [18]. Different parameters of these radio signals can be extracted to localize the target. For instance in [19] and [20], the authors propose to use Received Signal Strength Indicator (RSSI). In contrary, the authors of [21] apply time information related to radio signal propagation. Radio-based positioning can be classified as range-free and range-based methods [10]. Range is defined as the propagation distance from the Target Device (TAG) to the Anchor Node (AN).

2.1.2.1 Range-free Methods

Fingerprinting is one of the most widely used range-free localization methods. It has the advantage that there is no need for a special infrastructure as ubiquitous signals (such as Wi-Fi or magnetic field) can be used that are already present in indoor environments [22]. Indoor localization systems based on fingerprinting consist of two phases [23], which are shown in Figure 2.1. In the training phase (off-line), the fingerprint database (also named radio map) is built by collecting various types of radio signals in the target environment. In the localization phase (on-line), the fingerprint (composition of radio signals) at an unknown position is measured and then compared with the stored fingerprints in the fingerprint database to determine the closest match [24]. In this phase, any discriminative learning model (i.e., classification model exclusively based on previously observed data [7]) can be applied.

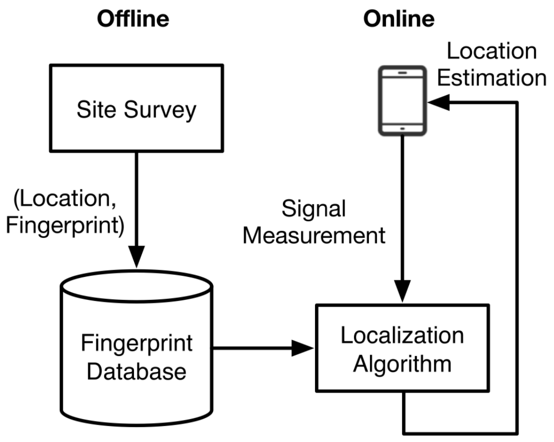


Figure 2.1: Fingerprinting phases [23]

There are two main kinds of learning methods according to [22].

- Probabilistic algorithms
- Deterministic algorithms

Probabilistic approaches are based on statistical inference between the target signal measurement and the stored fingerprints. Given a training set (set of fingerprints), maximum likelihood is applied to find the target's location [23]. In [25] for instance, the authors propose to apply Bayesian Graphical Models. In [26], Gaussian Process Latent Variable Model (GP-LVM) is used as probabilistic algorithm.

Deterministic approaches use distance or similarity metrices to compare online signal measurement and fingerprint data. The target position is then estimated based on the closest fingerprint. The ease of implementation is a major advantage of this method. In [27], K-Nearest Neighbours (KNN) is applied to find the most similar fingerprint. Also, some more advanced algorithms such as Support Vector Machine that is proposed in [28] belong to this category. A comparison of the performance of a large number of deterministic Machine Learning (ML) algorithms can be found in [29].

Despite its popularity in indoor localization, fingerprinting has also several drawbacks that have to be taken into account [22]. The main disadvantage is the expensive and time-consuming generation of the radio map. Especially in large areas of interest, many fingerprints need to be measured and stored in the fingerprint database. Furthermore, the radio map has to be maintained as any change in the infrastructure (e.g., displacement of access point) could affect the positioning results.

2.1.2.2 Range-based Methods

In range-based localization approaches, the creation and maintenance of a radio map is not necessary. Therefore, range-based positioning is often computationally less expensive compared to fingerprinting. The main idea is the derivation of ranges to different ANs from certain physical parameters which is named ranging [30]. In [30] and [31], the authors apply Time Difference Of Arrival (TDOA) and Differential

Time Difference Of Arrival (DTDOA) for ranging. Under LOS conditions, the usage of time information can provide satisfying accuracy. However, the signal's bandwidth limits this approach [30]. Another physical parameter that is used in ranging is RSSI. Often, the Log-Distance Path Loss (LDPL) model is applied [32] to calculate the propagation distance d using the following formula:

$$RSS = P_{\mathrm{t}} - (PL(d_0) + 10 \cdot \gamma \cdot \log_{10}(\frac{d}{d_0}) + X_\theta), \tag{2.1}$$

where P_{t} is the transmission power in dBm, $PL(d_0)$ is the path loss at reference point d_0, and γ is the path loss exponent. X_θ is a random variable with zero-mean reflecting shadowing attenuation in dB. In a rather novel approach called FILA [5], the authors proposed to extract Channel State Information (CSI) in order to estimate the range information. CSI can be considered as fine-grained values from the Physical Layer (PHY), which describes the amplitude and phase on each subcarrier in the frequency domain. That results in obtaining multiple CSIs at one time whereas with RSSI we only have one value per packet. The empirical results [5] show that this approach outperforms RSSI ranging in terms of accuracy as well as time tracking latency.

After this step, the ranging information needs to be converted into the location of the target. Different positioning algorithms such as trilateration and multilateration [30] can be applied to find the absolute positions.

2.1.3 Hybrid Localization Methods

Due to the mentioned drawbacks of both PDR-based and radio-based positioning, some works have been proposed that integrate these two approaches (named motion-assisted localization [23]). Figure 2.2 shows a typical system of motion-assisted localization. Wi-Fi fingerprinting is complemented by motion sensor data (accelerometer, gyroscope and magnetometer) in order to detect displacement and changes in direction. All the information is combined in a fusion algorithm, which provides the location estimation. For instance in [33], the authors presented a fingerprinting-based method by combining Wi-Fi information with accelerometer and magnetic field information. In another work [3], an enhanced particle filter that fuses range information estimated from RSSI, move detection using IMUs and floor plan constraints, was introduced.

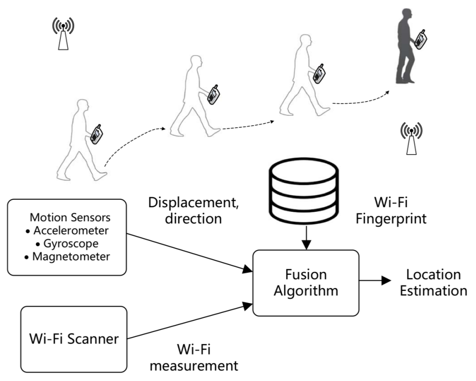


Figure 2.2: Illustration of motion-assisted system [23]

2.2 Deep Learning Algorithms

Over the past decade, Deep Learning algorithms have significantly advanced the performance of many supervised learning tasks, such as image classification [34], language modelling [35], healthcare [36], and online advertising [37]. Further applications can be found in [38]. As shown in Figure 2.3, Deep Learning is a subset of Machine Learning which is again a subset of Artificial Intelligence (AI) and can be defined as the set of algorithms that were developed in order to train multilayer neural networks more efficiently [39]. The purpose of neural networks is to mimic how the human brain works. Basically, neural networks use a hierarchy of layered filters in which each filter learns from the previous layer and then passes its output to the next layer, similar to the interaction of the neurons in the human brain where about 100 billion of neurons are connected to 100'000 of their neighbours each [40].

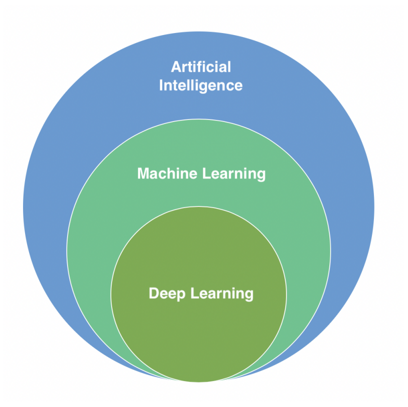


Figure 2.3: Overview of Artificial Intelligence [41]

This section provides an introduction to supervised learning and briefly introduces the different architectures of a Deep Neural Network (DNN): Feedforward Neural Network (FNN), Convolutional Neural Network and Recurrent Neural Network.

2.2.1 Supervised Learning

Neural networks are often applied to supervised learning problems. There is also intensive research in the area of unsupervised learning (e.g., clustering), for detailed information we refer to further reading [42] as this field is not covered in our work. The following dataset (Table 2.1) represents an example of supervised learning.

Living area (feet^2)	#Bedrooms	Price(1000$)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
⋮	⋮	⋮

Table 2.1: Supervised learning example [43]

Given data like this (living area, number of bedrooms and price for several houses in Portland, Oregon) we want to learn how to predict the prices of other houses in Portland, as a function of the size of their living area as well as the number of bedrooms [44]. To describe this supervised learning problem more formally, we denote the input variables (in this case the living area and the number of bedrooms) as $x^{(i)}$, also known as input features, and the output or target variable that we are trying to predict as $y^{(i)}$. So, a pair of $(x^{(i)}, y^{(i)})$ is called a training example and the set of m training examples $(x^{(i)}, y^{(i)}; i = 1 \dots m)$ is called training set. Additionally, X denotes the space of input variables and Y means the space of output variables. Therefore, the main goal is, given a training set, to find a function $h : X \to Y$ (h is also called hypothesis) so that $h(x)$ is a "good" predictor for the corresponding value of y.

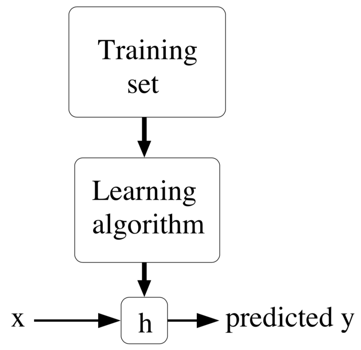


Figure 2.4: Process of finding hypothesis in supervised learning [43]

As shown in Figure 2.4, the hypothesis h is derived by a learning algorithm. This can be any discriminative (e.g., logistic regression) or generative machine learning algorithm (e.g., Gaussian Discriminant Analysis). Moreover, Deep Learning algorithms can be applied to approximate the function $h(x)$.

2.2.2 Feedforward Neural Networks

In Figure 2.5, a simple architecture of a Feedforward Neural Network is displayed. The reason why these networks are called feedforward is that the flow of information takes place only in the forward direction (in contrast to recurrent neural networks as we will see later) [45].

The displayed network architecture consists of an input layer, two hidden layers and an output layer. The network is fully connected as every unit of a layer is connected with every unit of the next layer. The learning process (training of the neural network) can be summarized in three steps [39]:

1. For every training example, the information (input features $x_1, x_2, \dots, x_n$) is propagated through the network from the input layer to the output layer (forward propagation) in order to make a prediction.
2. The predicted error is measured by a cost function between the predicted output and the desired output (real values).
3. In order to minimize this cost function, we apply gradient descent and update the weights in the network with the classical backpropagation algorithm [46].

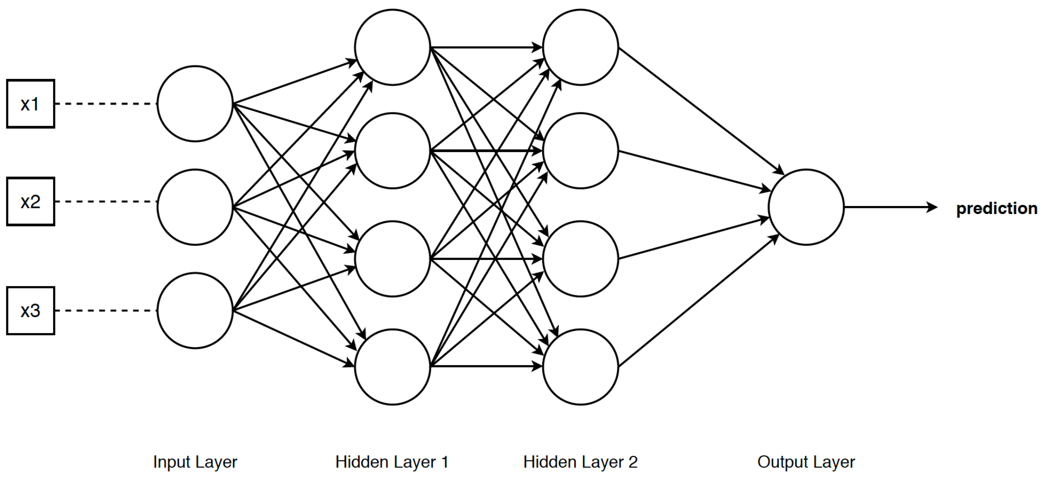


Figure 2.5: Architecture of Feedforward Neural Network

2.2.3 Convolutional Neural Networks

Convolutional Neural Networks belong to a family of models that try to imitate the functionality of the visual cortex in the human brain in order to visually detect objects [39]. The development of CNNs started back in the 1990s. At that time, the authors of [47] proposed a novel neural network architecture to classify handwritten digits. Due to their outstanding performance in image classification tasks, CNNs have received increasing attention over the past years.

The main goal of a CNN is to extract relevant features from the input data (in this case a handwritten digit) to construct a feature hierarchy. This means that low-level features (such as pixels and edges) are extracted in the first layers and then combined to high-level features which represent objects (such as a car or a dog) [39].

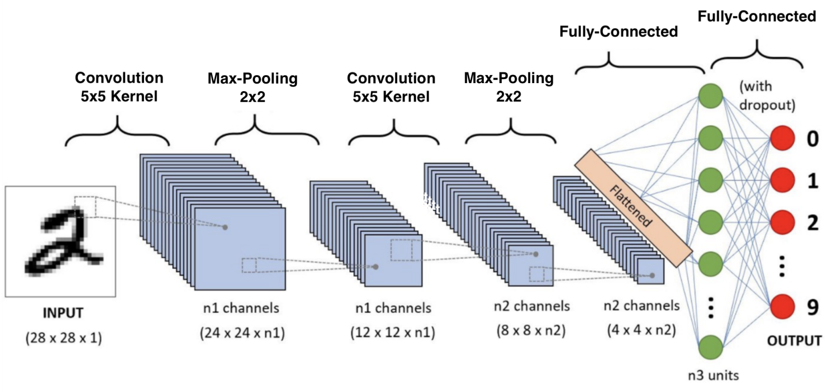


Figure 2.6: Application of Convolutional Neural Network to handwritten digit classification [48]

This can be achieved using a multilayer architecture as shown in Figure 2.6. The CNN for handwritten digit classification consists of three different types of layers:

1. **Convolutional Layer**
   A convolutional layer consists of multiple feature maps, each of which learns to extract a local feature regardless of its position in the previous layer [49].

A basic convolution operation is applied for the feature extraction. In Figure 2.7, the leftmost matrix represents the input data (e.g., 6x6 greyscale image). The 3x3 matrix in the middle is also called kernel or filter which is the second element involved in the convolution operation. Beginning in the top left-hand corner of the input matrix, the filter calculates the elementwise product between the area of the image over which the filter is hovering and the filter itself, followed by summing up the results to get a single number. Then, the filter moves to the right with a certain stride value (in this example we assume a stride value of 1) until it passes the complete width. Moving on, it jumps down to the beginning (left) of the next row of the image and repeats this process until the entire image is traversed [48]. However, a convolutional layer is not limited to one filter. There are often multiple filters which produce multiple convolved feature maps.

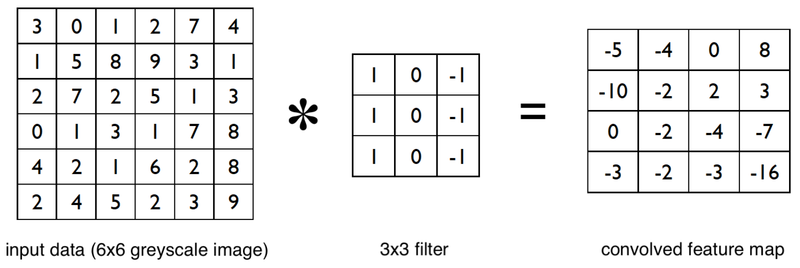

$$\begin{bmatrix} 3 & 0 & 1 & 2 & 7 & 4 \\ 1 & 5 & 8 & 9 & 3 & 1 \\ 2 & 7 & 2 & 5 & 1 & 3 \\ 0 & 1 & 3 & 1 & 7 & 8 \\ 4 & 2 & 1 & 6 & 2 & 8 \\ 2 & 4 & 5 & 2 & 3 & 9 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} -5 & -4 & 0 & 8 \\ -10 & -2 & 2 & 3 \\ 0 & -2 & -4 & -7 \\ -3 & -2 & -3 & -16 \end{bmatrix}$$

Figure 2.7: Convolution operation

2. **Pooling Layer**
   The convolved feature maps (results from convolutional layer) can be further processed by a pooling layer. Similar to convolutional layers, the purpose of a pooling layer is to reduce the spatial size of the convolved feature that decreases the computational power required to process the data. Additionally, it is useful for extracting dominant features [48] and suppresses the noise in the data. There are two different types of pooling as illustrated in Figure 2.8. Max pooling extracts the maximum value from the portion on the image that is covered by the kernel (here we use a 2x2 kernel). In contrast, average pooling returns the average of all the values from the portion.

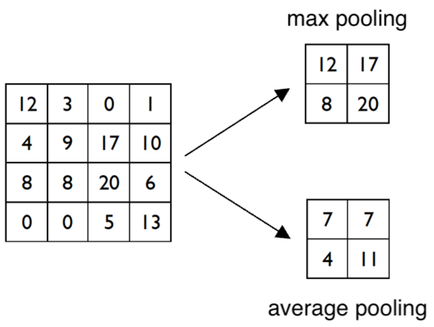

$$\begin{bmatrix} 12 & 3 & 0 & 1 \\ 4 & 9 & 17 & 10 \\ 8 & 8 & 20 & 6 \\ 0 & 0 & 5 & 13 \end{bmatrix} \quad \text{max pooling: } \begin{bmatrix} 12 & 17 \\ 8 & 20 \end{bmatrix} \quad \text{average pooling: } \begin{bmatrix} 7 & 7 \\ 4 & 11 \end{bmatrix}$$

Figure 2.8: Pooling operation

3. **Fully Connected Layer**
   After a series of convolutional and pooling layers, the final output is flattened (converted to a single column vector) and fed to a regular feedforward neural network. Therefore, we add one or multiple fully connected layers in order to perform the classification using the softmax function. In case of handwritten digit classification, there are ten different outputs representing the probabilities that the image belongs to a specific class (numbers from 0 to 9).

2.2.4 Recurrent Neural Networks

Recurrent Neural Networks are often used in time series prediction. Time series prediction can be defined as a process that extracts historical information and then determines future values [50]. A basic RNN architecture is depicted in Figure 2.9. In contrast to Feedforward Neural Networks where the information flows in one direction (i.e., no connection between the neurons within the same layer), RNNs allow neurons within the same hidden layer to be connected. Therefore, every hidden unit receives two inputs, the input at the current time step (denoted as $x^{<\text{t}>}$) and the activation from the previous time step (denoted as $a^{<\text{t-1}>}$). By means of this feature, the network is able to capture the effect of prior information on the current time step. In addition, there may be an output $\hat{y}^{<\text{t}>}$ at every time step. In an application of Natural Language Processing (NLP) for instance [51], the RNN takes the different words of a sentence as input $x^{<\text{t}>}$ and predicts for each word whether it is a name or not. Thus, we have a prediction $\hat{y}^{<\text{t}>}$ at every time step.

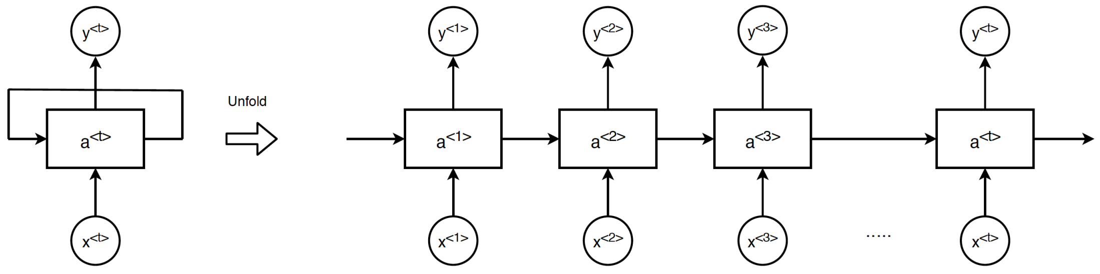


Figure 2.9: Architecture of Recurrent Neural Network

Similar to Feedforward Neural Networks, the parameters (weights to calculate activations and outputs) of an RNN are adjusted using backpropagation. So, we calculate a loss function (e.g., logarithmic loss function) by comparing the "real" values with the predictions and minimize the loss with gradient descent. However, there are two widely known issues when training a RNN by backpropagation, the vanishing and exploding gradient problems (for detailed information see [52]). Consequently, RNNs are unsuitable for modelling long-term dependencies. Long short-term memory networks can overcome this issue. LSTMs are a special category of RNNs that introduce the application of memory cells to store information in the long-term.

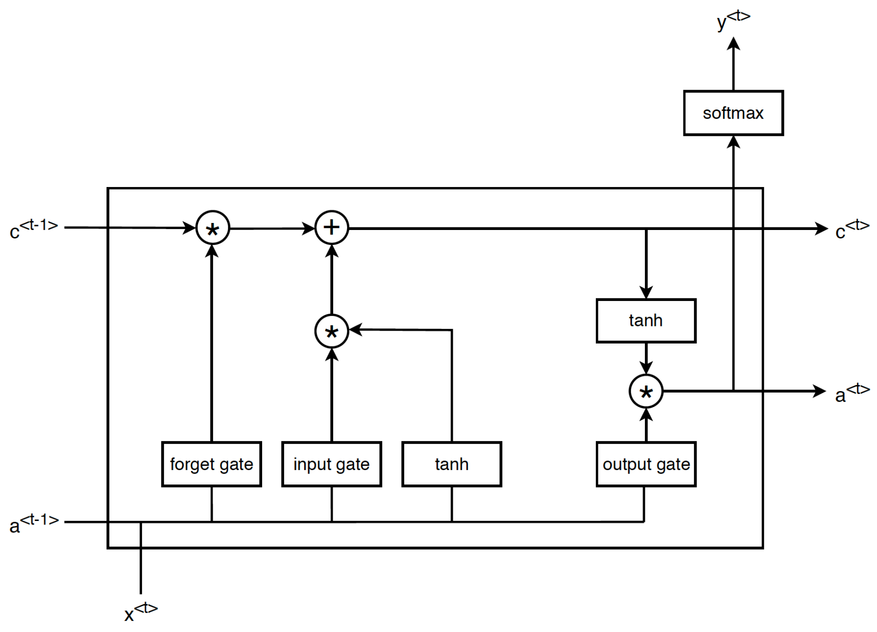


Figure 2.10: Architecture of Long Short-Term Memory cell

Figure 2.10 shows a typical architecture of a LSTM memory cell. As illustrated, the memory cell consists of three different gates classified as input gate, forget gate and output gate. The input gate controls how much information flows into the memory cell whereas the forget gate determines how much information still remains in the cell, and the output gate is responsible for deciding which information is used to compute the output activation of the memory block and further flows into the rest of the network [50]. These memory blocks are the main components of a LSTM network as they replace the standard RNN blocks (see Figure 2.11).

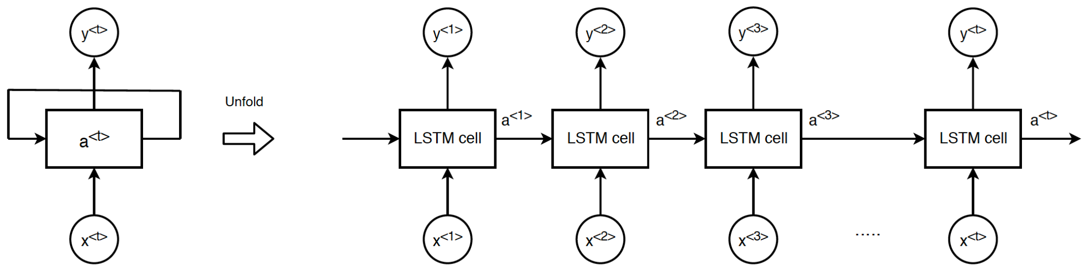


Figure 2.11: Architecture of LSTM network

2.3 Multi-Access Edge Computing Paradigm

In recent years, the importance of new computing applications, such as virtual reality and smart environments, has grown rapidly. These delay-sensitive applications have strong delay requirements which leads to a significant problem. This problem becomes more evident when multiple smart devices are integrated in human's life as for instance in case of smart cities or Internet of Things [53].

Cloud Computing solutions [54] are unable to meet these requirements. Therefore, Multi-Access Edge Computing technology [9] was introduced as an alternative solution to bring cloud computing capabilities to the network edge to fulfill the application demands of short latency and high resource-demanding [7]. Different definitions of MEC have been proposed. In [55], Multi-Access Edge Computing (formerly Mobile Edge Computing) is described as:

> *"Mobile Edge Computing is a model for enabling business oriented, cloud computing platform within the radio access network at the close proximity of mobile subscribers to serve delay sensitive, context-aware applications."*

When implementing the MEC architecture in an indoor localization system, compute-intensive tasks can be offloaded from client devices to near edge servers. Due to the short distance between the client and the server, the data transmission time can be reduced and, therefore, real-time localization performance can be guaranteed [7]. Instead of replacing the cloud computing model, MEC is rather a complement by executing delay-sensitive tasks on the MEC server while the delay-tolerant computation part can be executed on the cloud server [53].

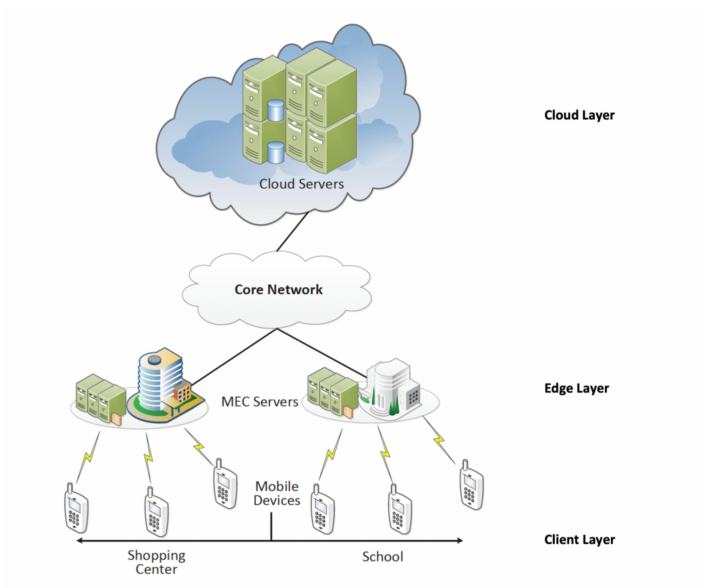


Figure 2.12: Overview of MEC-based system [55]

Figure 2.12 shows the architecture of a MEC-based system. The system consists of three different layers:

1. **Client layer:** including all types of devices (mobile devices and IoT devices) connected to the network
2. **Edge layer:** MEC servers (edge cloud) as the less resourceful cloud deployed in each of the mobile base stations
3. **Cloud layer:** cloud infrastructure hosted in the Internet that is able to store a large amount of data

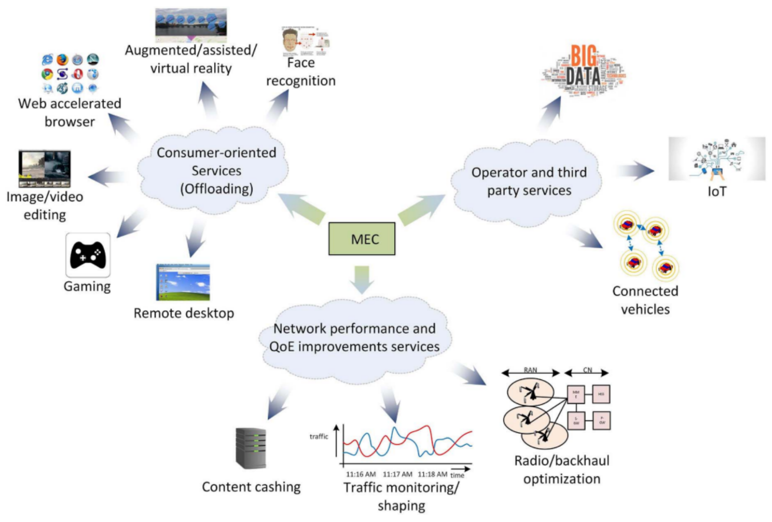

Augmented/assisted/ virtual reality
Face recognition
Web accelerated browser
Consumer-oriented Services (Offloading)
Image/video editing
Gaming
Remote desktop
MEC
Operator and third party services
IoT
Connected vehicles
Network performance and QoE improvements services
Content cashing
Traffic monitoring/ shaping
Radio/backhaul optimization
RAN
CN
traffic
11:16 AM
11:17 AM
11:18 AM
time

Figure 2.13: Use cases of MEC-based systems [9]

Due to the many advantages of MEC, there is a large number of applications that benefit from offloading heavy computations to near edge servers. Three main use cases can be distinguished according to [56] and [57]. These categories are shown in Figure 2.13 and shortly discussed in the next subsections.

2.3.1 Customer-Oriented Services

The first use case is consumer-oriented and, therefore, the end-users benefit from the MEC by offloading computational expensive applications to an external server near the network. In [58] for instance, the authors implemented a prototype named Edge Accelerated Web Browsing by executing most of the browsing functions, such as contents fetching, contents evaluation, layout of the contents component and rendering, on the edge server. In terms of performance (web contents rendering time), this approach is able to outperform the standard Android browser and Chrome browser. Furthermore, face recognition and NLP can benefit from a MEC architecture due to the large amount of required computational power and storage capacity. As a result, the execution time is decreased significantly [59]. Yet another work [60] showed the applicability of MEC for augmented reality. Conducting expensive experiments, the authors could achieve reducing the latency up to 88% and the energy consumption up to 93% by the computation offloading to the MEC server.

2.3.2 Operator and Third Party Services

The second use case category comprises applications that bring benefits for operators and third parties. One novel approach is described in [61] and [62] where the MEC paradigm is applied to IoT. IoT devices are connected through various radio technologies (e.g., 3G, LTE, Wi-Fi, etc.) using different communication protocols. So, MEC servers can be used as IoT gateways that intend to aggregate and deliver IoT services into highly distributed mobile base stations enabling applications to respond in real time. Moreover, Nokia has recently proposed a system based on MEC technology that is suitable for car-to-car and car-to-infrastructure communication [63]. Roadside applications are running in the edge layer and are able to receive local messages directly from applications in the vehicles and roadside sensors. The edge server then analyses this data and broadcasts warnings to nearby vehicles if necessary with very low latency [55].

2.3.3 Network Performance and QoE Improvement Services

The third category of applications aims to optimize network performance and improving Quality of Experience (QoE). For instance, MEC can be implemented in order to conduct radio network optimization. Network optimization is defined as the continuous process of enhancing the overall network quality [64]. Therefore, related information is collected from the client devices and processed in the edge layer. Adjusting parameters and hardware results in a more efficient scheduling and better support of the user demands. Additionally, MEC can also be used for mobile video delivery optimization [65] using throughput guidance for Transmission Control Protocol (TCP). TCP suffers from its difficulty to adapt to rapidly varying conditions on radio channels, which results in an inefficient use of resources. MEC can be used to address this problem by introducing an analytic application (running in the edge server) that provides a real-time estimation on the throughput to a video server.

2.4 Ultra-Wideband Radio Technology

Ultra-wideband wireless communication is a revolutionary technology for transmitting large amounts of data over a wide frequency spectrum using short-pulse, low-powered radio signals [66]. UWB has its origins in the late 1890s in the spark-gap transmission design by Marconi and Hertz. UWB radio communication was based on electromagnetic waves propagating from electrical sparks. The first systems were rudimentary and not efficient in terms of spectrum usage [67]. Research has been carried on mainly around 1960 when this technology was restricted to military and Department of Defense (DoD) applications [66]. A significant change in the development of UWB occurred in 2002, when the Federal Communications Commission (FCC) of the United States reserved the unlicensed frequency band between 3.1 GHz and 10.6 GHz for indoor UWB wireless communication systems. Afterwards, some industrial standards such as IEEE 802.15.3a (high data rate) and IEEE 802.15.4a (very low data rate with ranging) have been introduced based on UWB technology [68]. FCC proposed the following definition for UWB transmission [67]:

> *"Any signal, which has a fractional bandwidth (B_f) larger than* 0.2*, or which occupies a bandwidth (BW) greater than* 500 *MHz."*

The fractional bandwidth is the ratio of signal bandwidth (BW) and center frequency (f_c). As shown in Figure 2.14, conventional radio signals have much smaller fractional bandwidth as for instance the signal bandwidth is below the level of UWB. One important consequence of using a large frequency band and low power for communication is an increased channel capacity. This relationship can be explained through Shannon's formula [43]:

$$C = BW \cdot \log_2(1 + \frac{S}{N}) \frac{bits}{second} \tag{2.2}$$

This formula gives how many bits of information per time can be transmitted without error over a channel with a bandwidth of BW Hz and a limited average signal power S when the signal is exposed to an additive, uncorrelated noise of power N with a Gaussian probability distribution [66]. This shows that the increase of the channel capacity requires a linear increase in bandwidth whereas a similar capacity increase would require an exponential increase in power. Thus, UWB technology is able to transmit high data rates with very low power [66].

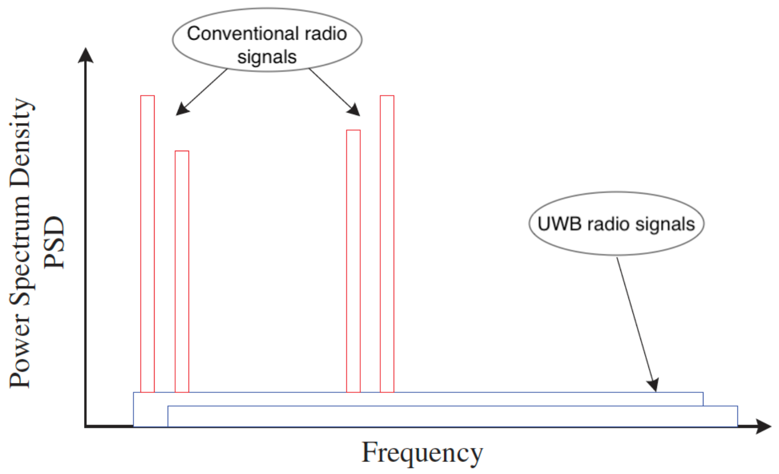


Figure 2.14: Power Spectrum Density (PSD) of different radio signals [67]

Due to these major characteristics, UWB can be applied in many fields. A classification of these applications is shown in Figure 2.15.

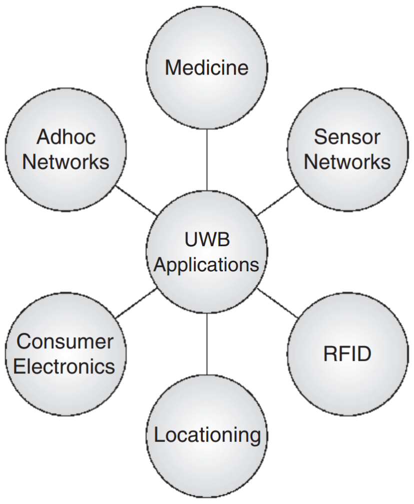


Figure 2.15: Overview of UWB applications [67]

Among many other applications, such as Wireless Sensor Networks, Radio Frequency Identification (RFID) or Wireless Adhoc Networking, Ultra-Wideband is used for indoor localization. Recently, a large number of works focusing on UWB indoor localization methods have been presented [8]. There are multiple reasons that explain this increase in popularity [69]. The main advantage over other radio technologies (such as Wi-Fi, Bluetooth, etc.) is that UWB radios are able to precisely differentiate between signals that are reflected from different objects. Therefore, this technology is very robust to the undesirable effect of multipath interference due to the fact that it transmits short pulses over a large bandwidth. Another characteristic is the ability of UWB pulses to effectively penetrate objects (such as walls, doors, etc.). The penetration capabilities come from the lower frequency components, which were mostly centered at 1 GHz for early UWB communication systems. However, the center frequency for most UWB systems has essentially increased with the 2002 ruling of FCC. Consequently, the penetration characteristics of the signals have decreased, especially compared to the IEEE 802.11b systems, which are centered at 2.4 GHz [70]. Furthermore, the high data rate of UWB (can reach 100 Megabits per second (Mbps)) is advantageous for indoor localization as it is suitable for near-field data transmission. A comparison of the maximal signal rate with other radio technologies can be found in [71]. In [72], the author adds one more positive characteristic of UWB to this list. Due to the low power of the signal and the even distribution over a wide frequency spectrum, it is difficult to intercept an UWB signal. In addition, there is no disturbance of other wireless technologies. According to [69], UWB is the most promising technology for indoor positioning and localization.

There are different positioning algorithms that can be used in order to apply this technology for indoor localization. In [69], these algorithms are classified into five main categories: (1) time of arrival (TOA), (2) time difference of arrival (TDOA), (3) angle of arrival (AOA), (4) received signal strength (RSS) and (5) hybrid algorithms.

2.4.1 TOA-based Algorithms

In TOA systems, the position of the target device is estimated based on the distance between the TAG and multiple receivers (intersection of circles as shown in Figure 2.16). The TAG transmits a time stamped signal towards the receiving beacons. When it is received, the distance is calculated from the measured transmission time delay and the corresponding speed of the signal. This method requires precise knowledge of the transmission start time. Therefore, all the receiving beacons are synchronized with an accurate time source [73]. One of the disadvantages of this approach is the need for precise time delay measurement. For this purpose, an additional server is required, which increases the cost of the localization system.

TOA-based algorithms have been used to locate targeted objects for various applications and environments. In [74] for instance, the author introduced an UWB-based localization system for coal mines. Using TOA-based algorithms, workers can be effectively located in case of an emergency. In another work [75], the design of an UWB navigation system was presented that allows mobile robots (MR) or other mobile vehicles to autonomously navigate in indoor environments. By means of TOA-based algorithms, the MR is able to estimate its position referred to fixed reference points with known locations. Experiment results show that in 95% of the measurements the estimation error is below $20cm$.

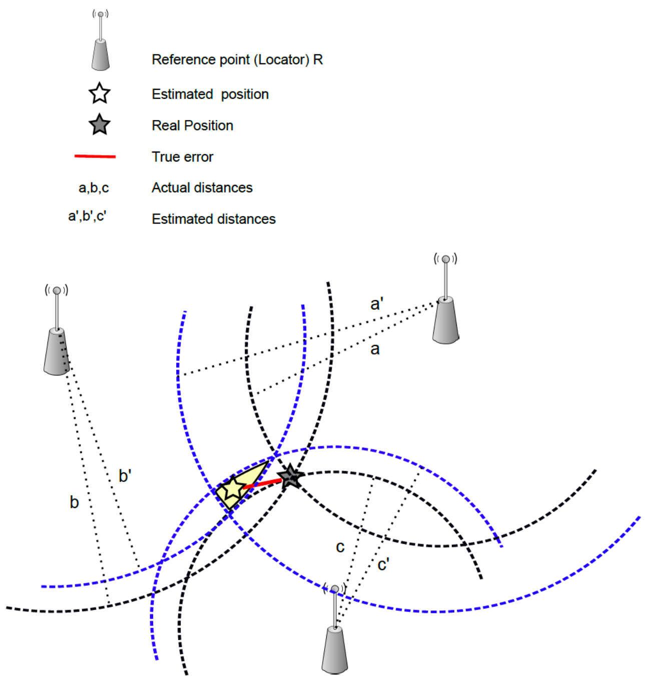


Figure 2.16: Position estimation using Time of Arrival (TOA) [69]

2.4.2 TDOA-based Algorithms

TDOA-based localization systems are based on measuring the time difference of arrival of a signal transmitted by a target object and received by three or more receivers, see Figure 2.17. It is also possible that a single receiver determines the time difference by measuring the delta in arrival times of two transmitted signals [69]. Each difference of arrival produces a hyperbolic curve in the localization space. By calculating the intersection of multiple hyperbolic curves the position of the target device can be estimated. This technique is called multilateration. In contrast to TOA, TDOA does not require the use of a synchronized time source of transmission in order to resolve timestamps and determine the location [73]. A disadvantage of these algorithms is the increased need of bandwidth in comparison with other algorithms due to the fact that multiple receivers share the measurement data and collaborate to find the location of the transmitter.

A large number of UWB localization systems have been developed that apply TDOA-based positioning algorithms. For instance in [76], the authors proposed a centimeter-accuracy, 3D localization system using UWB technology for tracking miniature mechanical parts within a metal enclosed space (airplane wheel). This is achieved by implementing two UWB transmitters in a mechanical tool. This tool rotates around the target object while the position of both transmitters is tracked by the localization system. The moving trajectories are then used to determine the 3D location of the mechanical part. The system implements a TDOA-based localization algorithm and consists of four receivers (minimum number of receivers required for 3D localization with TDOA). Two technical challenges were observed during the experiments: (1) angle-dependent waveform distortion and (2) path-overlap. The authors concluded that for medium to low location accuracies (e.g., greater than

10 cm) the error due to angle-dependent pulse distortion is negligible, regardless of the timing-detection method. However, for centimeter-accuracy localization, such errors must be reduced. In order to overcome the effect of path-overlap in dense multipath environments, a new range estimation method was developed. See [76] for more detailed information about these technical challenges.

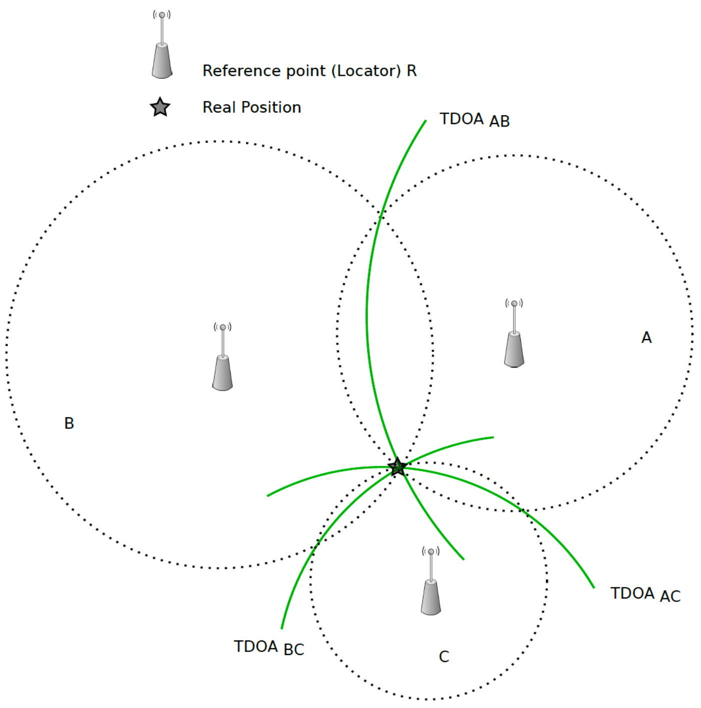


Figure 2.17: Position estimation using Time Difference of Arrival (TDOA) [69]

2.4.3 AOA-based Algorithms

In localization systems that implement AOA-based algorithms, the estimation of the signal reception angles (at least two beacons required) is compared with either the signal amplitude or carrier phase across multiple antennas [69]. The number of beacons can be increased (three or more) in order to improve accuracy [73]. As indicated in Figure 2.18, the location of the target can be calculated from the intersection of the angle line of each reference point. AOA-based localization techniques are limited in indoor environments. This is mainly due to the fact that they are affected by multipath interference and NLOS propagation of signals as well as reflections from walls and other objects. This can significantly change the direction of signal arrival (angle changes) and, therefore, decrease the accuracy of the localization system. Furthermore, the cost of the AOA system implementation is increased by the requirement of additional antennas with the capacity to measure angles [73].

AOA-based algorithms have been presented in many studies that evaluated the performance of this technique for different applications, environments, hardware, and configurations. In [77], a real-time localization system was introduced that is able to define the health status of an athlete objectively. The system allows a highly precise localization ($10 - 20cm$) of the moving target device, which is worn by the athlete. Additionally, four fixed receivers were deployed, which measure the AOA of the UWB impulses transmitted by the target. The estimated position of the target is then calculated from the intersection of the angle lines. Processing the collected data from different runs allows to conclude technical considerations about the performance of the athlete and, therefore, measure the progress or regress of his/her physical status.

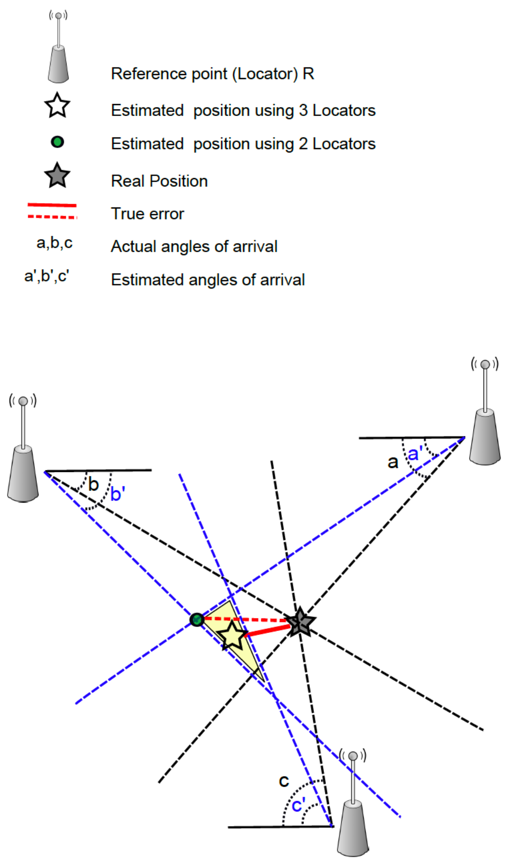

Reference point (Locator) R
☆ Estimated position using 3 Locators
● Estimated position using 2 Locators
★ Real Position
True error
a,b,c Actual angles of arrival
a',b',c' Estimated angles of arrival
a
a'
b
b'
c
c'

Figure 2.18: Position estimation using Angle of Arrival (AOA) [69]

2.4.4 RSS-based Algorithms

In RSS-based algorithms, signal strength of received UWB impulses is used as an estimator of the distance between the transmitters and the receiver (target device). Therefore, the target measures the received signal strength (RSS) of multiple UWB transmitters. Then, the position of the target can be estimated applying either trilateration or fingerprinting. Trilateration algorithms estimate the distances from the target device to three reference nodes. Given this information, the current location of the target can be calculated. In contrast, fingerprinting is based on collecting a dataset of RSS fingerprints in the target environment (offline phase). Online measurements are then compared with the entries in the fingerprint database to find the closest match and, thus, the position estimation of the target object [69].

UWB localization systems using RSS-based positioning algorithms were developed in a large amount of literature. The authors of [78] presented a device free localization (DFL) system that fuses UWB radios and RSS sensors in order to localize people through a building (e.g., in case of a tactical operation or crisis situation to help emergency personnel localize people in a room before they enter). The DFL system consists of radio transceivers (combination of transmitter and receiver) on all four sides of a target area. For every pair of sensor, the RSS is measured. Changes in the RSS measurements introduced by a person near the link between a transmitter and receiver are used for the localization process. The performance of this localization system can be improved by adding UWB radios to the setup. Combining UWB radios with RSS sensors, a person can still be localized through walls even when the devices are only placed on two sides of the target area.

2.4.5 Hybrid Algorithms

Hybrid algorithms combine multiple positioning algorithms. The advantage of this method is that the algorithms are able to complement each other and cover different parts depending on their capabilities and strengths. In general, the overall accuracy of a localization system based on hybrid algorithms will improve. On the other hand, complexity and cost of the system will increase as well.

In literature, there are a lot of studies focusing on the performance of localization systems that implemented hybrid algorithms. For instance in [79], the authors introduced an indoor localization system that can be applied in a medical environment. The aim of this work is to seamlessly combine indoor and outdoor localization in order to track medical staff, patients or instruments. By integrating GPS and UWB technologies along with TDOA and AOA as positioning algorithms, accurate localization is provided regardless of where the person or object is in indoor or outdoor. Experiments showed that the accuracy in indoor scenarios can be reduced to $15cm$. In contrast, the localization error outside is $10m$ or above.

Despite all the advantages of using UWB technology for indoor localization, there are still some open issues prior to becoming a successful indoor localization choice [8]. Most of the works in this field focus on improving the accuracy of the localization system. However, they rarely consider scalability aspects. Current UWB-based localization methods are typically designed for localizing at most tens of mobile devices and the experiments are conducted in small (office) environments or open space areas.

Chapter 3

System Architecture

This chapter presents the design details of our proposed MEC-based indoor localization system. Figure 3.1 gives an overview of the system architecture consisting of two different layers: Client layer and Edge layer. In our work, there is no need for a Cloud layer. The next sections describe some further details of each layer.

3.1 Client Layer

The client layer includes multiple hardware components, such as a target device, multiple Wi-Fi access points and UWB anchor nodes. We use a Raspberry Pi as target device (client) that needs to be located. It is equipped with external sensors (accelerometer, magnetometer, temperature, humidity and air pressure sensors) that allow the measurement of motion-based and environmental information. Additionally, the RP includes Wi-Fi and UWB modules so that RSSI data from every access point and anchor node can be collected. The Wi-Fi access points and UWB anchor nodes (also Raspberry Pis) are distributed homogenously in the area of interest in order to cover the largest possible area. More detailed information about the hardware components can be found in the next chapter. The data collection process is conducted constantly by the TAG. Instead of sending the collected data to the edge layer directly (as raw data), the client processes them locally to derive meaningful insights (i.e., Wi-Fi, UWB and EMF fingerprints) [7].

The processed information is then merged into an appropriate data format (in our case a JSON file) and sent to the edge layer via a data transmission module (WebSocket connection) for further processing. This architecture moves all the device-dependent data processing, such as Wi-Fi or UWB signal processing, to the client layer. Therefore, the edge layer is completely independent of the client device specifications which is a significant advantage as the system is able to support different client devices. For instance, the Raspberry Pi could easily be replaced by a smartphone without any major modifications in the client layer as well as edge layer [7].

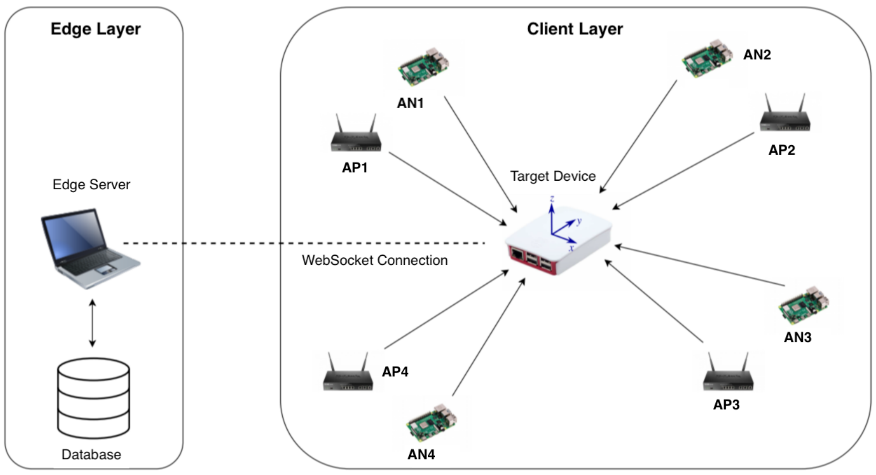


Figure 3.1: Overview of the localization system components

3.2 Edge Layer

The computationally heavy localization algorithms that provide real-time indoor locations of the target device are running in the edge layer due to the higher computation resources available in the edge server. The proposed room-level localization algorithm is illustrated in Figure 3.2. It takes the data periodically received from the client layer (sensor information, EMF values, Wi-Fi and UWB fingerprints) as input. As shown in Figure 3.2, the Deep Learning localization algorithm includes two different neural networks.

First, there is a Convolutional Neural Network which takes the input data and propagates this data through multiple layers (more details about the implementation of the algorithms can be found in the next chapter) in order to extract relevant information. In addition, the CNN aims to suppress the noise in the data due to the temporal and spatial instability of the fingerprints. The compressed data is then passed to the second neural network, the Recurrent Neural Network. Since we have a time series dataset as input (a series of datapoints ordered in time), the RNN composed of LSTM cells is capable of modelling long-term dependencies in the data. This means that historical information about the location of the target device is extracted and used to determine future locations. For instance, with the knowledge that the last 100 datapoints are all labeled with the same room, the algorithm learns that it is more likely that the next location of the target device will be in the same room or a room next to it instead of one far away. Therefore, the RNN computes probabilities for every room that the TAG could be located in. The room with the highest probability is chosen as prediction for the actual datapoint. Finally, all the real-time predictions are stored in a database in the edge server for further evaluation (performance metrices). The learning of the Deep Learning localization algorithm is an iterative process and consists of two phases, forward propagation and backpropagation. This learning process is described in detail in section 4.2.

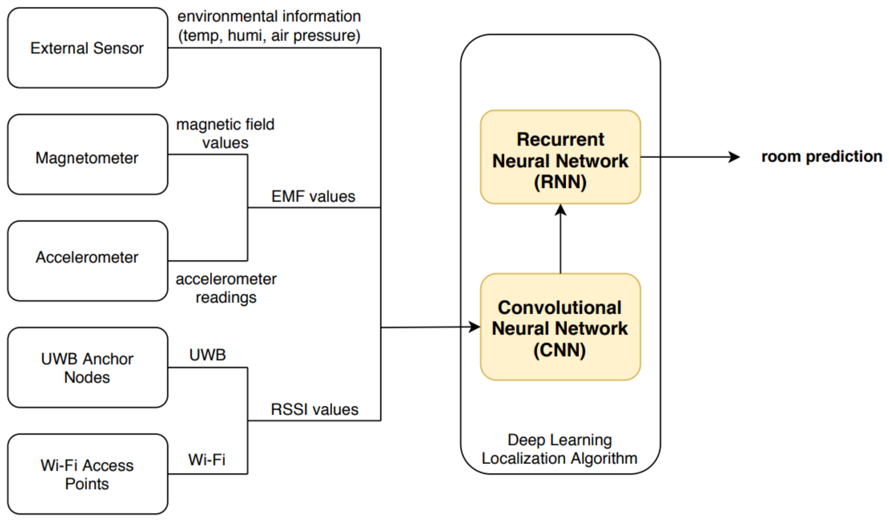


Figure 3.2: Overview of the localization algorithm components

3.3 Cloud Layer

Instead of storing the results of the localization algorithm in a local database in the edge layer, another layer (cloud layer) could be introduced which takes responsibility for the storage of historical localization information. The data flow would then be slightly different: The client device is continuously collecting data from on-site, then it passes this data to the edge layer for processing (i.e., localization of the target device). Afterwards, the processed data is sent to the cloud layer which is located in a different geographical location (e.g., cloud infrastructure hosted in the Internet). The main advantage of this approach is that the cloud layer enables high-order queries over the historical localization information to provide predictive analysis and business control. Besides this, the application could be scaled up to several client devices gathering data from multiple scenarios and making this data accessible anywhere in the world [7]. As we do not have to meet these requirements of worldwide accessibility of the data and scalability in our application, there is no cloud layer in our work.

Chapter 4

System Implementation

In this chapter, we specify the hardware components of our system architecture in detail. In addition, we focus on the implementation of the proposed localization algorithm and discuss the underlying UWB communication.

4.1 Hardware Components

We have implemented a MEC-based system for accurate room recognition. The system architecture was presented in the last chapter (Figure 3.1) and includes four different hardware components: a target device (TAG), some commercial Wi-Fi access points (APs), some UWB anchor nodes (ANs) and an edge server.

The TAG is the device to be localized. It is a Raspberry Pi 3 Model B [80] running on the operating system Raspbian with a 1.2 GHz CPU and 1 GB RAM. Using a Grove Base Hat [81] we connected a Grove Temperature and Humidity Sensor [82] to the Raspberry Pi in order to measure environmental factors. Additionally, the 40-pin GPIO was equipped with a Sequitur InGPS Tag Chip from the company UNISET [83], which enables UWB communication along with acceleration and magnetic field readings as well as air pressure measurement. The UWB ANs that were distributed in the area of interest are also Raspberry Pis with the same specifications as the TAG. In contrast to the TAG, the ANs were equipped with a Sequitur InGPS Anchor Chip instead of a Tag Chip. As for the Wi-Fi APs, we use some commercial access points (D-Link DAP-2553 and TP-Link AC750) as well as the strongest access points (APs with the highest RSSI values) which are already present in the environment. The positions of the deployed ANs and APs in the environment were chosen to provide the maximum coverage in the area of interest.

Our proposed localization algorithm is running on the edge server, which is a mid-end commercial laptop with limited computational power. The communication between client layer and edge layer was implemented by using WebSocket technology [84]. WebSocket is a communication protocol that enables a two-way communication between a client and a host. The protocol consists of an opening handshake, followed by basic message framing.

WebSocket is simply a layer on top of TCP and does the following [84]: It

- adds a security model for browsers,
- adds an addressing and protocol naming mechanism that supports multiple services on one port and multiple host names on one IP address,
- layers a framing mechanism on top of TCP, and
- includes an additional closing handshake

In contrast to a HTTP connection where every action of the webserver requires a request of the client, WebSocket allows both the server and the client to send messages at any time without any relation to a previous request (bidirectional communication). Besides this, WebSocket enables full-duplex communication – client and server can send messages independently at the same time. These characteristics are suitable for our application since we have to be able to send the measurement data from the client device to the edge server at any time without opening and closing the connection every time (single TCP connection). Autobahn [85] was used to provide WebSocket client and WebSocket server was implemented in Python Django [86]. Table 4.1 shows the detailed specifications of each hardware component.

Component	**Specification**
Edge Server	**Model:** Samsung Series 9 Ultrabook **CPU:** 1.8 GHz Intel Core i5-3337U **RAM:** 4 GB **Architecture:** 64-bit **OS:** Ubuntu 18.04.1 LTS
Target Device	**Model:** Raspberry Pi 3 Model B **CPU:** Quad Core 1.2 GHz Broadcom BCM2837 **RAM:** 1 GB **Architecture:** 64-bit **OS:** Raspbian 4.14 **WLAN:** Wi-Fi b/g/n **UWB:** Sequitur InGPS Lite Tag
Access Points	**Model:** D-Link DAP-2553 and TP-Link AC750
Anchor Nodes	**Model:** Raspberry Pi 3 Model B **CPU:** Quad Core 1.2 GHz Broadcom BCM2837 **RAM:** 1 GB **Architecture:** 64-bit **OS:** Raspbian 4.14 **WLAN:** Wi-Fi b/g/n **UWB:** Sequitur InGPS Lite Anchor

Table 4.1: Hardware components

4.2 Localization Algorithms

4.2.1 Deep Learning Algorithm

In the room recognition method, we set up a Deep Learning algorithm (combination of Convolutional Neural Network and Recurrent Neural Network as seen in the previous chapter) using Python's high-level neural network API Keras [87] that runs on top of the open-source platform TensorFlow [88].

4.2.1.1 Data Collection

To build the fingerprint map, we placed the Raspberry Pi (target device) on a SunFounder PiCar [89] that was remote-controlled by a laptop. The fingerprint database entries were collected equally distributed over the whole area of interest in each room. During the data collection process we gathered 55000 data points in total, 5000 in each of the 11 rooms (independent of the size of the room). The layout of the rooms in the area of interest is illustrated in Figure 5.2. The data collection was restricted by the computational capabilities of the built-in Wi-Fi sensor of the Raspberry Pi, i.e., every fingerprint entry was collected at a rate of approximately 1 entry/second. Therefore, building the fingerprint database was very time-consuming and took about 15 hours.

Feature	**Description**
Wi-Fi RSSI	The TAG scans the surrounding Wi-Fi access points to obtain their RSSI values. Wi-Fi RSSI depends on the distance between the TAG and the AP as well as some environmental factors. RSSI is measured in dBm and the closer to 0 dBm, the stronger the signal is.
UWB RSSI	The TAG collects the RSSI values of the surrounding UWB anchor nodes by sending a request to every AN which then sends back its corresponding RSSI. The range of the UWB RSSI is identical to the Wi-Fi RSSI.
EMF values	Magnetic field values with respect to the device's coordinate system are measured by the Sequitur InGPS Tag Chip attached to the TAG. However, these values have to be converted into the earth magnetic field values (as described above).
Temperature Humidity Air Pressure	Due to the dependence of the RSSI not only on the distance between the TAG and the APs and ANs but also on environmental factors, we measure three different meteorological parameters that could possibly affect the RSSI. Temperature and humidity is measured by an external sensor and air pressure values are provided by the Sequitur InGPS Tag Chip.

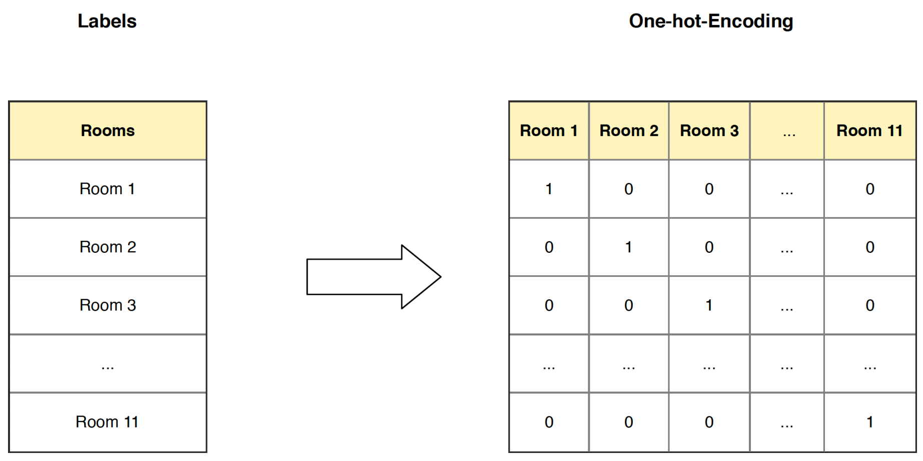
Table 4.2: Detailed description of the features

The fingerprint instances consist of different features. We collected the Received Signal Strength of the various Wi-Fi access points and UWB anchor nodes in the environment. In addition, the magnetic field values were measured along with some environmental factors, such as temperature, humidity and air pressure. A description of these features can be found in Table 4.2. Along with the features, every datapoint contains a label (enumerated rooms) of the room where the measurement was taken.

4.2.1.2 Data Preprocessing

Before training the model, data preprocessing was necessary in order to guarantee high-quality data. This can a have significant impact on the performance of the learning algorithm [39]. First of all, there were a few missing values in the dataset due to the unavailability of some sensors during collection. One way to handle missing values is removing the affected datapoints. However, this might lead to the loss of observations with valuable information. Therefore, we applied mean imputation method instead. This works by calculating the mean of the non-missing values in a column and then replacing the missing values (with the mean) within this column [90]. In addition, depending on the position of the TAG, some signals from Wi-Fi access points and UWB anchor nodes were too weak to be measured by the built-in sensors due to their large distance from the TAG. Signal strengths can range from approximately $-30dBm$ (strong connection) to $-110dBm$ (no signal). In our dataset, the lowest measured signal strength is $-109dBm$. Therefore, we decided to set all undetected signals to $-110dBm$.

Finally, we had to apply one-hot encoding [39] in order to transform the nominal feature 'room label' into numeric values. As illustrated in Figure 4.1, for each possible value (room 1, room 2, etc.) of this label, we introduced a separate dummy variable, which takes zero or one as values. Thus, an observation with room label 1 for instance has a value of one for the newly created variable 'room 1' and zero for all the other dummy variables.



Rooms
Room 1
Room 2
Room 3
...
Room 11

Room 1	Room 2	Room 3	...	Room 11
1	0	0	...	0
0	1	0	...	0
0	0	1	...	0
...	...	...	...	...
0	0	0	...	1

Figure 4.1: One-hot encoding of room labels

The conversion of the magnetic field values from the local coordinate system of the RP to the world coordinate system was adapted from the Android Developer Documentation [91]. The measured magnetic field values are represented by a vector with respect to the local coordinate system (x, y, z) where x is along the shorter side of the device pointing right, y is along the longer side pointing up and z is perpendicular to the surface pointing out (as indicated in Figure 3.1). However, when the target device stands still and only rotates around its own axis, the magnetic field values change and consequently the fingerprint also changes. That is not suitable for our application since in this case the position of the TAG is still the same (so fingerprint should not change). Therefore, we have to transform the magnetic field values to the fixed world basis (w_1, w_2, w_3) to get the earth magnetic field values. This can be achieved by means of a basis change where we multiply the magnetic field values with a rotation matrix. See more details about the derivation in [91].

4.2.1.3 Model Training

The localization algorithm was trained with the preprocessed data - we first standardized the features to zero-mean and standard deviation of one - to optimize the internal parameters (e.g., weights). We used 80% of the data as training data and 20% as test data. However, certain parameters (called hyperparameters) that have a significant impact on the performance of the learning-based algorithm were not optimized during the training process. We used a nested cross validation [39] instead to adjust them. Nested cross validation techniques consist of an inner and outer cross validation. The inner cross validation intends to find the optimal hyperparameters over a set of possible values whereas the outer cross validation estimates the generalization error. We applied ten-fold cross validation on both inner and outer cross validation [24]. According to the results of the nested cross validation, the proposed localization algorithm has the following optimal values of hyperparameters (listed in Table 4.3).

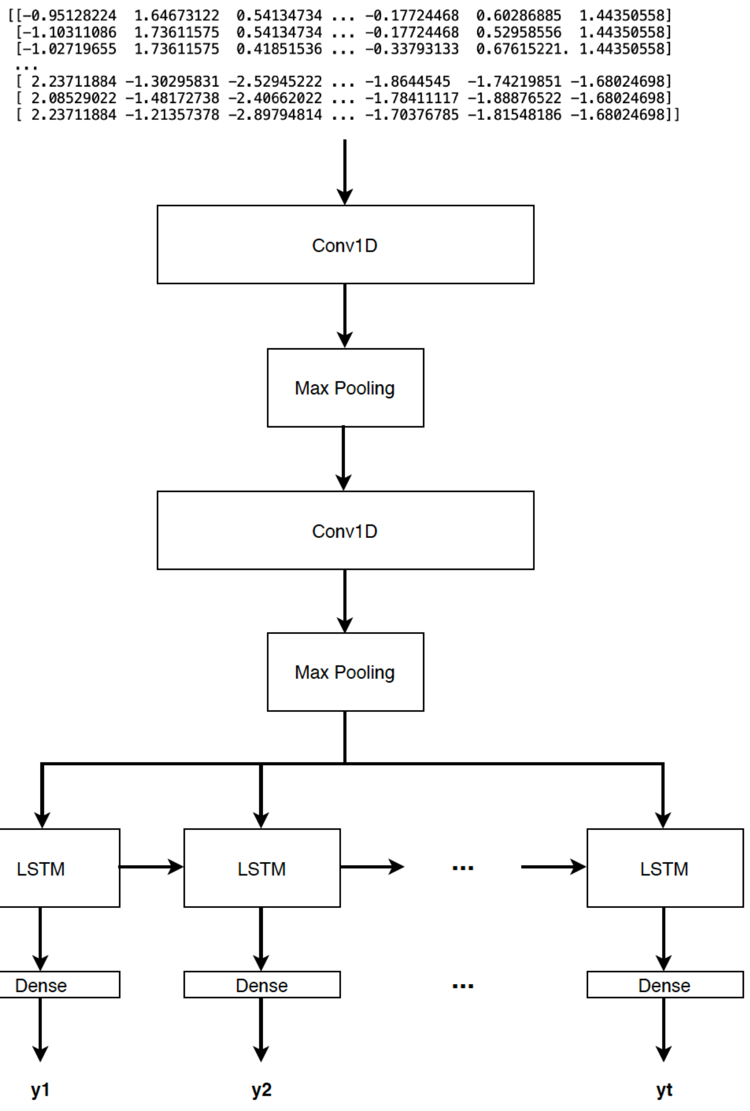
Hyperparameter	**Optimal Value**
Number of convolutional layers	2
Number of filters	10
Kernel size	3x3
Padding	same
Activation function	tanh
Optimizer	adam
Number of LSTM layers	1
Number of LSTM cells per layer	100

Table 4.3: Overview of optimal hyperparameters (combined model)

Figure 4.2 shows the implementation of the Deep Learning algorithm. When training the model with the training data, sample after sample is propagated through the network consisting of two convolutional layers (including a max pooling layer each), a LSTM layer as well as a dense layer. The convolutional part of the network is responsible for the feature extraction. This means that the data is decomposed and simple patterns are extracted. The LSTM layer is able to include historical information of samples that were propagated through the network earlier in order to calculate the output. The fully connected layer (dense layer) acts as a classifier on top of the features. It learns how to use the features provided by convolutions with the aim to correctly classify the samples (predict the room where it is located). After a specific number of propagated samples (called batch size), the weights of the network are updated. For this purpose, we use the categorical entropy as loss function [39]:

$$J = -\sum_{n=1}^{N} y_i * log(\hat{y}_i), \tag{4.1}$$

where N is the number of samples, $\hat{y}_i$ is a vector containing the calculated probabilities that the sample belongs to the different rooms and y_i is the ground truth. In order to minimize this cost function, the weights of the network are adjusted. After training the model with all samples (from training dataset), we have the final model, which can then be used for the online localization.



```
[[-0.95128224  1.64673122  0.54134734 ... -0.17724468  0.60286885  1.44350558]
 [-1.10311086  1.73611575  0.54134734 ... -0.17724468  0.52958556  1.44350558]
 [-1.02719655  1.73611575  0.41851536 ... -0.33793133  0.67615221. 1.44350558]
 ...
 [ 2.23711884 -1.30295831 -2.52945222 ... -1.8644545  -1.74219851 -1.68024698]
 [ 2.08529022 -1.48172738 -2.40662022 ... -1.78411117 -1.88876522 -1.68024698]
 [ 2.23711884 -1.21357378 -2.89794814 ... -1.70376785 -1.81548186 -1.68024698]]
```

Figure 4.2: Implementation of Deep Learning algorithm

4.2.2 Reference Algorithms

To be able to compare our indoor localization approach, we also implemented a standard Feedforward Neural Network and a Feedforward Neural Network that has differential RSSI values as input instead of raw values (as proposed in [2] and [6]). We will refer to these two neural networks as the reference algorithms. The optimized hyperparameters of these algorithms (using nested cross validation) are listed in the Table 4.4 below.

Hyperparameter	**Optimal Value**
Number of layers	3
Number of units per layer	5
Activation function	tanh
Optimizer	adam
Kernel initializer	glorot uniform
Bias initializer	zeros

Table 4.4: Overview of optimal hyperparameters (neural network)

4.3 Ultra-Wideband Communication

For UWB communication between the TAG and the ANs we use Sequitur InGPS Lite (provided by the company UNISET), which enables wireless communication between neighbouring nodes in the same network via a UWB interface. However, UNISET does not provide full insight into the transmission techniques [92]. Hereafter, we describe the customizable parameters and known configuration.

The radio module of Sequitur InGPS Lite provides six different radio channels (listed in Table 4.5). They differ from each other in terms of central frequency and bandwidth. The central frequency ranges from about 3.5 GHz (channel 1) to 6.5 GHz (channel 6) whereas the corresponding bandwidth is between 500 MHz and 1300 MHz. Another customizable configuration parameter is the radio mode, which defines the data rate according to the IEEE 802.15.4a specification [93]. There are three available values of 110 kbps (mode 1), 850 kbps (mode 2) and 6.8 Mbps (mode 3) to choose for the data rate. It is important that different nodes in the network must be set on the same radio channel and radio mode to communicate correctly. In general, a lower frequency or lower data rate allows larger operating distances between the nodes. As a default, the Pulse Repetition Frequency (PRF) is set to 64 MHz for all the channels. Finally, the power level of the radio module can take values between 1 (lowest value) and 63 (highest value). The transmitting power is increased of 0.5 dB by every number.

In our application, we configured all the UWB devices (target and anchor nodes) to operate in channel 5 with a central frequency of 6489.6 MHz and a bandwidth of 500 MHz as well as radio mode 1 with a data rate of 110 kbps. The pulse repetition frequency (PRF) was set to the default value of 64 MHz and the transmission power level to a maximum of 63.

Channel Number	**Central Frequency [MHz]**	**Bandwidth [MHz]**
1	3494.4	500
2	3993.6	500
3	4492.8	500
4	3993.6	1300
5	6489.6	500
6	6489.6	1100

Table 4.5: Configuration options for the radio channels

Chapter 5

Performance Evaluation

In this chapter, we describe the setup of our experiments to test the localization algorithms. Furthermore, the experiment results are presented and analysed in detail.

5.1 Experiment Setup

We tested our localization system in an office-like indoor scenario along complex trajectories. The experiments were conducted in the third floor of the building of the Institute of Computer Science at the University of Bern. An area of 702 m^2 (39m x 18m) was chosen to deploy the 8 UWB ANs and 4 Wi-Fi APs (two additional access points were used that are already present in the environment). They were distributed to cover the largest area possible. The exact positions are marked in Figure 5.1.

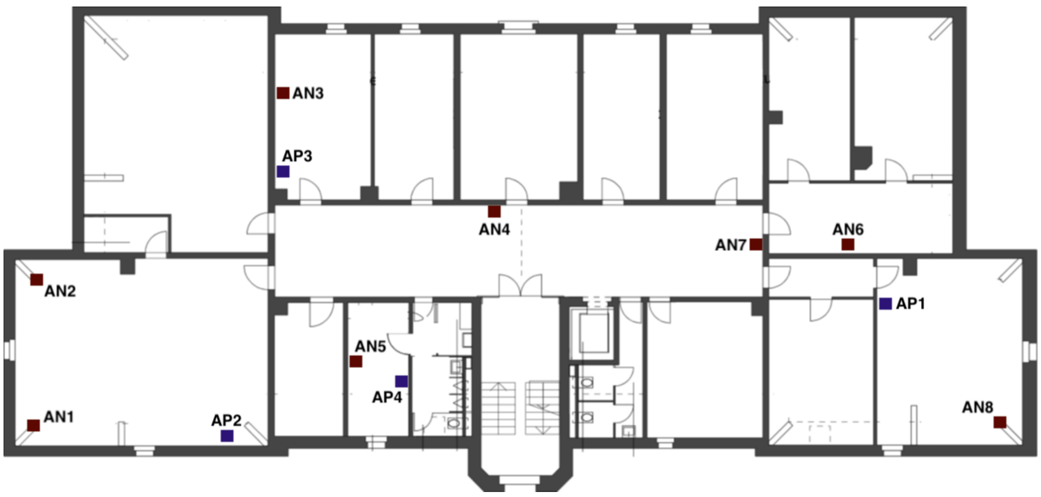


Figure 5.1: ANs and APs distribution in the area of interest

The target device (Raspberry Pi) was placed on a SunFounder PiCar moving along three different trajectories (Figures 5.3, 5.4 and 5.5). Every time a new fingerprint was available (approximately once per second), the TAG sent this data to the edge server where the room-level method was executed. This results in 388 measurements in trajectory 1, 419 measurements in trajectory 2 and 337 measurements in trajectory 3. The localization algorithms (proposed Deep Learning algorithm, neural network with raw RSSI values and neural network with differential RSSI values) predicted the room where the TAG is most likely to be located in. For that purpose, we defined 11 zones in the area of interest. Each zone is a wall separated area (i.e., room or corridor). Only one corridor was split into two zones (zone 5 and 9) due to its large size. The definition of the zones is displayed in Figure 5.2.

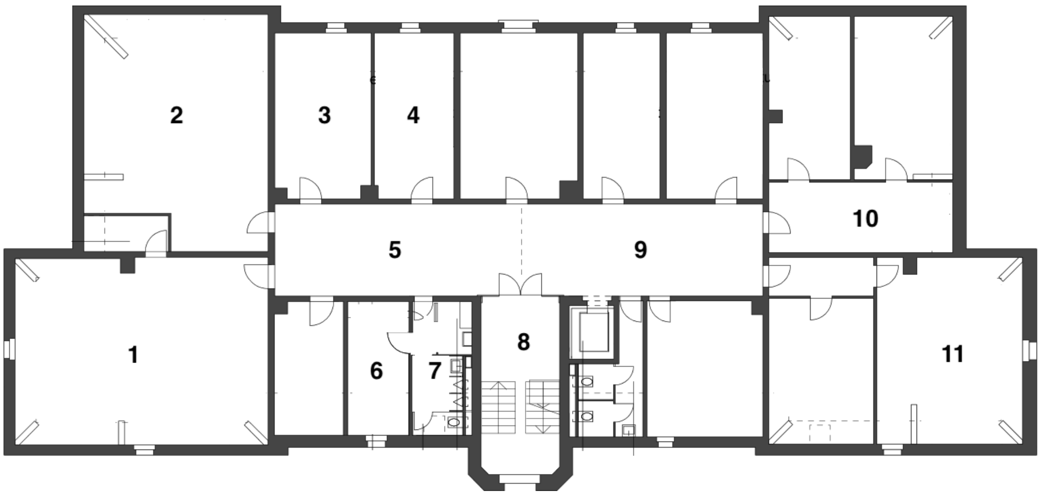


Figure 5.2: Zone definition in the area of interest

To evaluate the performance of our predictive models, we apply three different metrices: correct classification rate (prediction accuracy), F1 score and processing time. Prediction accuracy is defined as the ratio of the correctly predicted observations to the total number of observations. F1 score combines two performance metrices, precision and sensitivity (recall). Table 5.1 shows the relation between prediction and actual value. Precision is defined as the number of True Positives (TP) divided by TP and the number of False Positives (FP), i.e., the percentage of all observations that are predicted as positive which are actually positive.

$$Precision = \frac{TP}{TP + FP} \tag{5.1}$$

Sensitivity can be considered as the percentage of all actually positive samples which are predicted as positive. It is defined as the ratio of the number of TP to the number of TP and the number of False Negatives (FN).

$$Sensitivity = \frac{TP}{TP + FN} \tag{5.2}$$

Therefore, F1 score can be written as (harmonic mean of precision and sensitivity):

$$F1 = 2 \cdot \frac{sensitivity \cdot precision}{sensitivity + precision} \tag{5.3}$$

		Prediction		
		p′	**n′**	**total**
Actual Value	**p**	True Positive	False Negative	P
	n	False Positive	True Negative	N
	total	P′	N′	

Table 5.1: Relation between prediction and actual value of observations

5.2 Experiment Results

In the following, we present the experiment results of our proposed room recognition method in the indoor scenario. We analyse the localization results with reference to three different metrices (prediction accuracy, F1 score and processing time) and compare them with the results of a standard Feedforward Neural Network (one model with raw RSSI values and one model with differential RSSI values).

Figure 5.6 shows the accuracy of room recognition for the learning algorithms in the three trajectories. Figures 5.7 to 5.9 present the harmonic mean (F1 score) of precision and sensitivity for every room and in Figure 5.10, the average processing time per prediction is displayed. All classifiers are fed with the same fingerprint data (consisting of Wi-Fi and UWB RSSI, EMF values, temperature, humidity and air pressure). Furthermore, the hyperparameters of the models are optimized as described in Chapter 4.

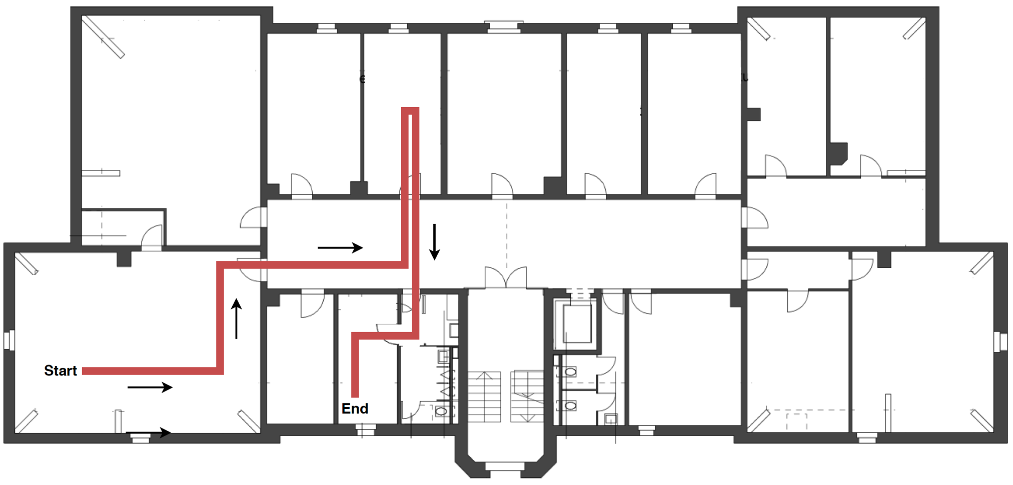


Figure 5.3: Path of trajectory 1

Considering the prediction accuracy, our proposed Deep Learning algorithm is able to outperform the reference algorithms in all trajectories. In the first trajectory (see 5.3), our room recognition algorithm achieves a prediction accuracy of 98.33% whereas the standard Feedforward Neural Network and the differential Feedforward Neural Network classify 91.67% and 92.50% of the observations correctly. Thus, the accuracy of our proposed algorithm is improved by 7.3% and 6.3% compared to the reference algorithms.

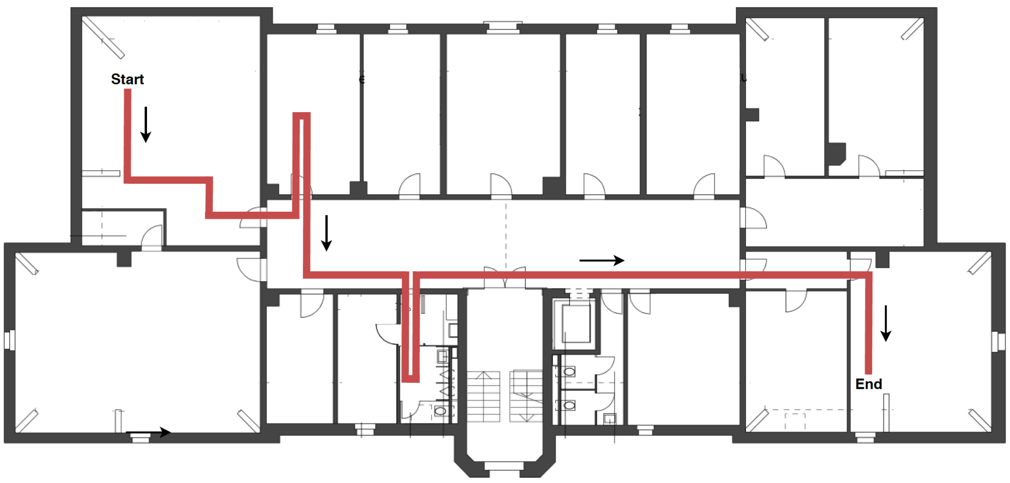


Figure 5.4: Path of trajectory 2

In trajectory 2 (see 5.4), the differences between the accuracy of the three localization algorithms are even more significant than in the first trajectory. The proposed algorithm achieves 99.38% accuracy compared to 90.62% and 94.38% of the standard FNN and differential FNN, respectively. This is an increase of 9.7% and 5.3%, respectively.

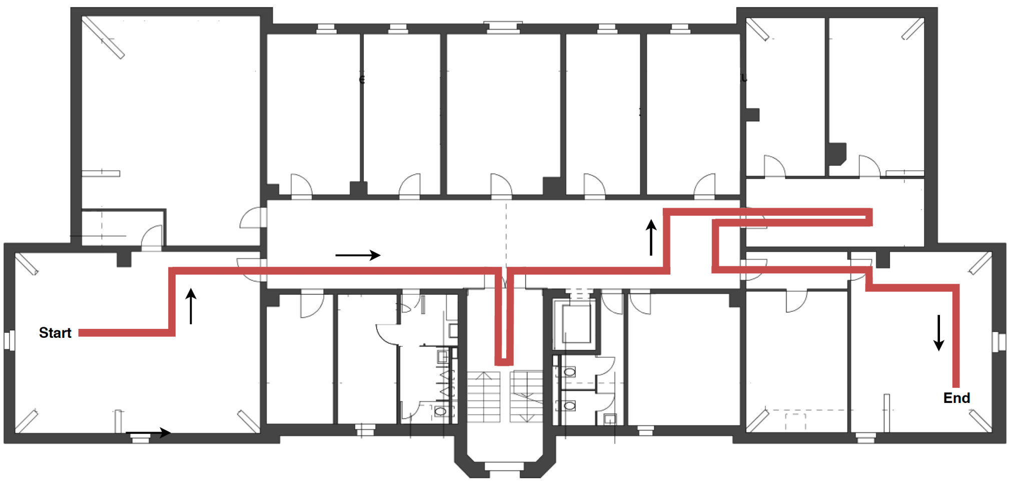


Figure 5.5: Path of trajectory 3

As for trajectory 3 (see 5.5), similar results can be observed. Here, our proposed algorithm outperforms the reference algorithms (standard FNN achieves 90.71% accuracy and differential FNN 95.71%) by 7.9% and 2.2%, respectively, with a prediction accuracy of 97.86%. This results in an average prediction accuracy of 98.52%.

In terms of precision and sensitivity of the models, Figures 5.7 - 5.9 show that our novel approach outperforms the existing solutions in all tested zones (i.e., rooms and corridors) of all trajectories. In most of the zones (1, 2, 3, 4, 6, 7 and 11), the localization algorithm is able to reach the highest F1 score possible (perfect precision and recall) which means that all observations in these zones were predicted correctly. However, there are other rooms (e.g., room 9) where a lower F1 score is achieved, i.e., room 9 is one of the hardest zones to classify correctly. This can be explained as zone 9 corresponds to a corridor where the collected fingerprints are similar to fingerprints in neighbouring zones. Especially in border areas, the localization algorithm is not able to differentiate between this corridor and an adjacent room. That leads to higher localization errors. Nevertheless, the proposed localization algorithm improves the F1 score in this zone by 18.3% and 5.4% (in trajacetory 2) compared to standard FNN and differential FNN, respectively.

In addition, we introduce another metric to measure the time it takes to make a prediction (per observation) as this is an important aspect of a real-time localization system. As indicated in Figure 5.10, the average prediction time is very different in all the models. However, our proposed algorithm is not able to reduce the processing time per prediction. In contrary, the prediction of one observation lasts $820\mu s$ and takes therefore the double and fourfold amount of time compared to the reference algorithms. The standard FNN achieves an average processing time of $431\mu s$ and the differential FNN reaches $258\mu s$. This is mainly due to the higher computational resources that are required to calculate a prediction with our model including two different Deep Learning algorithms, Convolutional Neural Network and Recurrent Neural Network. However, the average processing time could be decreased by offloading the heavy computations to the edge server in our work compared to running the localization algorithms on the target device itself as described in [24] for instance.

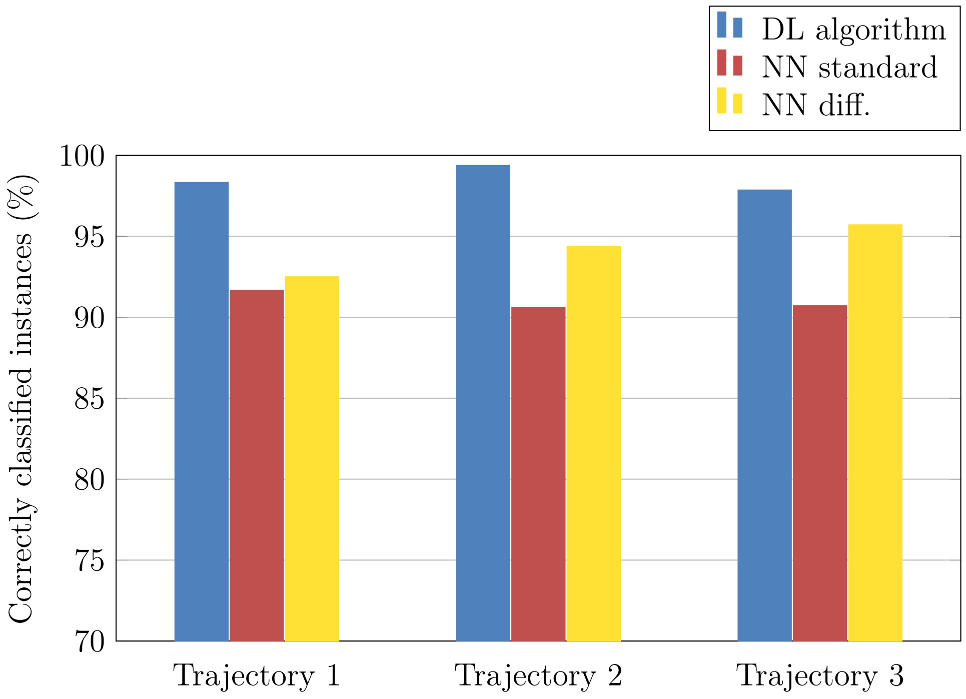


Figure 5.6: Predictive model accuracy

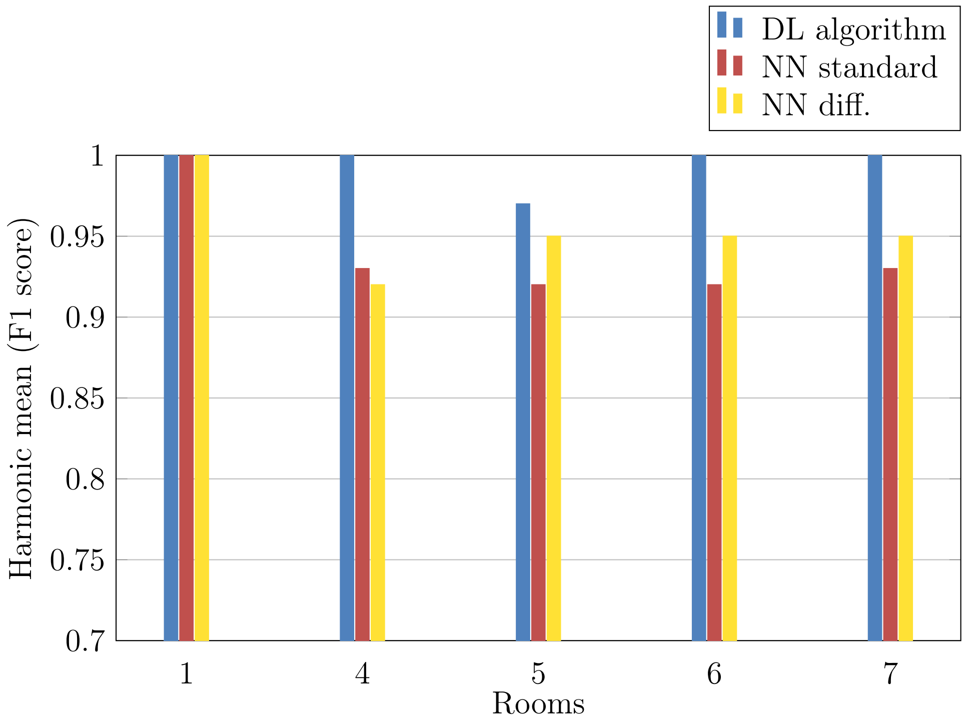


Figure 5.7: Room recognition F1 score (harmonic mean) for trajectory 1

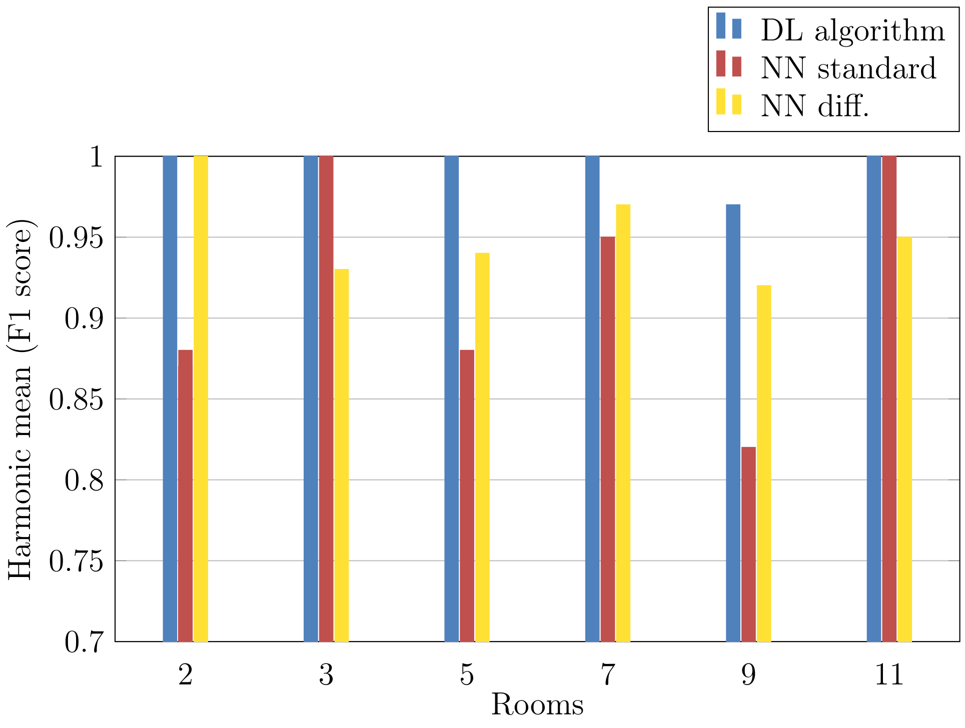


Figure 5.8: Room recognition F1 score (harmonic mean) for trajectory 2

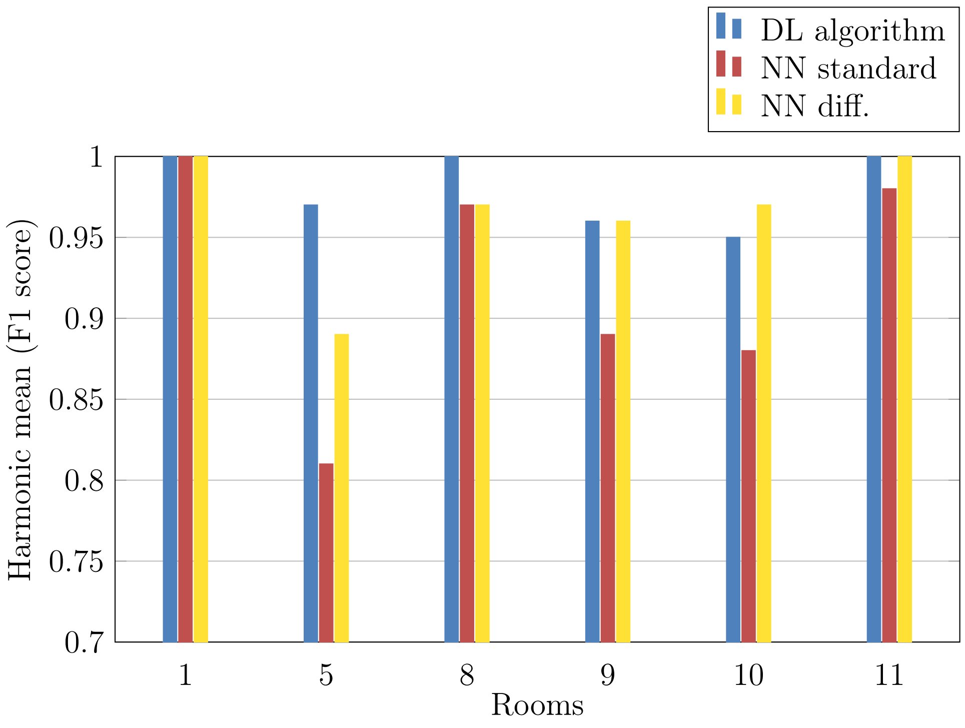


Figure 5.9: Room recognition F1 score (harmonic mean) for trajectory 3

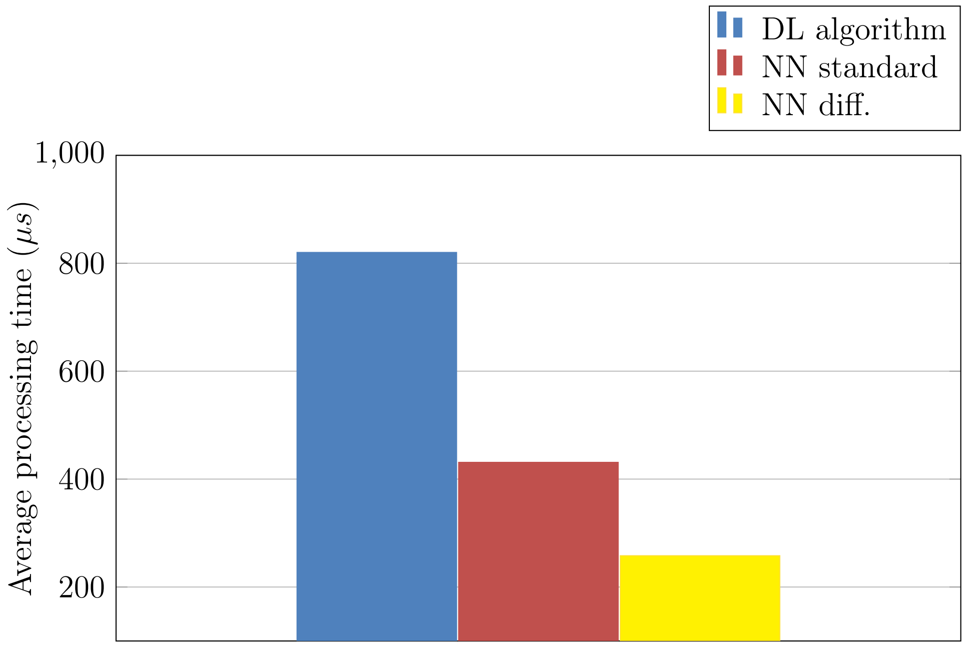


Figure 5.10: Average processing time (per prediction)

Chapter 6

Conclusion and Future Work

In this section, we conclude our work and summarize our findings. Furthermore, some aspects are discussed that could be tackled in future research to further improve the room level localization system.

6.1 Conclusion

In our work, a MEC-based room level indoor localization system is presented to achieve high room recognition accuracy in complex indoor scenarios. The Deep Learning approach fuses Wi-Fi and UWB radio signals along with EMF values and environmental information in a model that includes a Convolutional Neural Network to filter out the environmental noise and a Recurrent Neural Network using LSTM cells to model long-term dependencies in the data. The localization algorithm is running on an external server near the network edge to offload the heavy computations. Experiment results show that our approach achieves high room level localization accuracy (average of all three trajectories is 98.52%). Compared to previous works in this field (e.g., [24]), our implementation is able to accomplish higher localization accuracy and, therefore, outperforms existing approaches. Due to the short distance between target device (Raspberry Pi) and edge server, the data transmission time can be reduced and, therefore, the performance of the indoor localization system (processing time per prediction) can be improved. As we include UWB radio signal information, our model is robust to multipath effects since UWB radios are able to differentiate between pulses that are reflected from different objects. However, this requires the deployment of additional hardware to enable UWB communication. To further increase the performance of our system, some improvements could be made which are discussed in the next section.

6.2 Future Work

One problem that occurred in our implementation was the failure of UWB communication between the TAG and the ANs due to long transmission distances. It happened that at a given point in the indoor environment, we could not receive the response of every single UWB anchor node as they were too far away from the target. The reason for that is a low UWB-AN density (i.e., low number of UWB ANs in a large area of interest). To tackle this UWB communication issue, the UWB-AN density could be increased by either reducing the size of the area of interest or deploying more UWB ANs. With such a modification of the system, a larger amount of data would be available for the estimation and, therefore, the localization accuracy could be improved. Another approach is to apply a different wireless technology to communicate between the TAG and the ANs. For instance, ZigBee or Wi-Fi could provide longer transmission ranges [71].

Our work is limited to the localization of one target device at a time. In order to apply indoor localization in complex buildings with multiple users, the system should be able to scale up. To enable scalability, a cloud layer can be introduced. This additional layer is responsible for storing historical localization information as well as enabling high-order queries over the historical localization information to provide predictive analysis and business control. Furthermore, multiple client devices can be introduced to gather data from multiple scenarios. The connection between edge layer and cloud layer could be established using WebSocket technology for instance.

Finally, it is possible that this work will be integrated with an enhanced indoor tracking system (similar to [7]) that applies our proposed algorithm as room recognition method, which provides coarse-grained accuracy. Fusing with additional information (e.g., floor plan information) in a particle filter, the tracking algorithm is then triggered to localize the target within the predicted zone to provide highly accurate results. We can see that our proposed localization approach is promising for the future and can be envisioned as a basis for accurate indoor localization.

Bibliography

[1] M. Baldauf, S. Dustdar, and F. Rosenberg, "A survey on context-aware systems," *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 2, no. 4, pp. 263–277, 2007.

[2] N. Chang, R. Rashidzadeh, and M. Ahmadi, "Robust indoor positioning using differential wi-fi access points," *IEEE Transactions on Consumer Electronics*, vol. 56, no. 3, pp. 1860–1867, 2010.

[3] J. L. Carrera, Z. Li, Z. Zhao, T. Braun, and A. Neto, "A real-time indoor tracking system in smartphones," in *Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, ser. MSWiM '16, 2016, pp. 292–301.

[4] H. Abdelnasser, R. Mohamed, A. Elgohary, M. F. Alzantot, H. Wang, S. Sen, R. R. Choudhury, and M. Youssef, "SemanticSLAM: Using environment landmarks for unsupervised indoor localization," *IEEE Transactions on Mobile Computing*, vol. 15, no. 7, pp. 1770–1782, 2016.

[5] K. Wu, J. Xiao, Y. Yi, M. Gao, and L. M. Mi, "FILA: Fine-grained indoor localization," *Proceedings IEEE INFOCOM*, pp. 2210–2218, 2012.

[6] J. Wang, G. Qinghua, H. Wang, H. Chen, and M. Jin, "Differential radio map-based robust indoor localization," *EURASIP Journal on Wireless Communications and Networking*, vol. 2011, no. 1, pp. 1–12, 2011.

[7] J. L. Carrera, Z. Zhongliang, M. Wenger, and T. Braun, "MEC-based UWB indoor tracking system," in *IEEE/IFIP 15th Wireless On-demand Network systems and Services Conference (WONS 2019)*, 2019, pp. 138–145.

[8] M. Ridolfi, S. Velde, H. Steendam, and E. De Porter, "Analysis of the scalability of UWB indoor localization solutions for high user densities," *Sensors*, vol. 18, no. 6, 2018.

[9] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[10] Z. Li, D. B. Acuña, Z. Zhao, J. L. Carrera, and T. Braun, "Fine-grained indoor tracking by fusing inertial sensor and physical layer information in WLANs," in *IEEE International Conference on Communications (ICC)*, 2016, pp. 1–7.

[11] S. Kemp. (2019). Digital 2019: Internet trends in q3 2019, [Online]. Available: `https://datareportal.com/reports/digital-2019-internet-trends-in-q3` (visited on 07/23/2019).

[12] N. Kakiuchi and S. Kamijo, “Pedestrian dead reckoning for mobile phones through walking and running mode recognition,” in *Proceedings of the 16th International IEEE Annual Conference on Intelligent Transportation Systems (ITSC 2013)*, 2013, pp. 261–267.

[13] F. Hong, H. Chu, L. Wang, Y. Feng, and Z. Guo, “Pocket mattering: Indoor pedestrian tracking with commercial smartphone,” *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2012.

[14] J. Won Kim, H. J. Jang, D.-H. Hwang, and C. Park, “A step, stride and heading determination for the pedestrian navigation system,” *Journal of Global Positioning Systems*, vol. 3, no. 1, pp. 273–279, 2004.

[15] J. Borenstein and L. Ojeda, “Heuristic drift elimination for personnel tracking systems,” *Journal of Navigation*, vol. 63, no. 4, pp. 591–606, 2010.

[16] J. Borenstein, L. Ojeda, and S. Kwanmuang, “Heuristic reduction of gyro drift for personnel tracking systems,” *Journal of Navigation*, vol. 62, no. 1, pp. 41–58, 2009.

[17] A. Mariakakis, S. Sen, J. Lee, and K.-H. Kim, “SAIL: Single access point-based indoor localization,” in *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys ’14, 2014, pp. 315–328.

[18] J. L. Carrera, Z. Zhao, T. Braun, and Z. Li, “A real-time indoor tracking system by fusing inertial sensor, radio signal and floor plan,” in *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2016, pp. 1–8.

[19] M. Youssef and A. Agrawala, “The horus WLAN location determination system,” in *Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys ’05, 2005, pp. 205–218.

[20] P. Bahl and V. N. Padmanabhan, “RADAR: An in-building RF-based user location and tracking system,” in *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*, vol. 2, 2000, pp. 775–784.

[21] Z. Li, T. Braun, and D. C. Dimitrova, “A time-based passive source localization system for narrow-band signal,” in *IEEE International Conference on Communications (ICC)*, 2015, pp. 4599–4605.

[22] J. Conesa, A. Pérez-Navarro, J. Torres-Sospedra, R. Montaliu, R. Berkvens, G. Caso, C. Costa, N. Dorigatti, N. Hernandez, S. Knauth, E. S. Lohan, J. Machaj, A. Moreira, and P. Wilk, “Challenges of fingerprinting in indoor positioning and navigation,” in *Geographical and fingerprinting data to create systems for indoor positioning and indoor/outdoor navigation*, ser. Intelligent data-centric systems, 2019, pp. 1–20.

[23] S. He and S.-H. Gary Chan, “Wi-fi fingerprint-based indoor positioning: Recent advances and comparisons,” *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 466–490, 2016.

[24] J. L. Carrera, Z. Zhao, and T. Braun, “Room recognition using discriminative ensemble learning with hidden markov models for smartphones,” in *IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2018, pp. 1–7.

[25] D. Madigan, R. P. Martin, W.-H. Ju, P. S. Krishnan, A. S. Krishnakumar, and E. Einahrawy, “Bayesian indoor positioning systems,” in *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 2, 2005, pp. 1217–1227.

[26] B. Ferris, D. Fox, and N. Lawrence, “WiFi-SLAM using gaussian process latent variable models,” in *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, ser. IJCAI’07, 2007, pp. 2480–2485.

[27] J. Ma, X. Li, X. Tao, and J. Lu, “Cluster filtered KNN: A WLAN-based indoor positioning scheme,” *International Symposium on a World of Wireless, Mobile and Multimedia Networks*, pp. 1–8, 2008.

[28] C.-L. Wu, L.-C. Fu, and F.-L. Lian, “WLAN location determination in e-home via support vector classification,” in *IEEE International Conference on Networking, Sensing and Control*, vol. 2, 2004, pp. 1026–1031.

[29] D. Mascharka and E. Manley, “LIPS: Learning based indoor positioning system using mobile phone-based sensors,” in *13th IEEE Annual Consumer Communications Networking Conference (CCNC)*, 2016, pp. 968–971.

[30] Z. Li, T. Braun, and D. C. Dimitrova, “A passive WiFi source localization system based on fine-grained power-based trilateration,” in *IEEE 16th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2015, pp. 1–9.

[31] Z. Li, D. C. Dimitrova, D. H. Raluy, and T. Braun, “TDOA for narrow-band signal with low sampling rate and imperfect synchronization,” in *7th IFIP Wireless and Mobile Networking Conference (WMNC)*, 2014, pp. 1–8.

[32] V. Erceg, L. J. Greenstein, S. Y. Tjandra, S. R. Parkoff, A. Gupta, B. Kulic, A. A. Julius, and R. Bianchi, “An empirically based path loss model forWireless channels in suburban environments,” *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 7, pp. 1205–1211, 1999.

[33] P. S. Nagpal and R. Rashidzadeh, “Indoor positioning using magnetic compass and accelerometer of smartphones,” in *International Conference on Selected Topics in Mobile and Wireless Networking (MoWNeT)*, 2013, pp. 140–145.

[34] D. Ciregan, U. Meier, and J. Schmidhuber, “Multi-column deep neural networks for image classification,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3642–3649.

[35] M. Sundermeyer, H. Ney, and R. Schlüter, “From feedforward to recurrent LSTM neural networks for language modeling,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 3, pp. 517–529, 2015.

[36] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, “Dermatologist-level classification of skin cancer with deep neural networks,” *Nature*, vol. 542, no. 7639, pp. 115–118, 2017.

[37] J. Fu, L. Liang, X. Zhou, and J. Zheng, "A convolutional neural network for clickbait detection," in *4th International Conference on Information Science and Control Engineering (ICISCE)*, 2017, pp. 6–10.

[38] M. Dixit, A. Tiwari, H. Pathak, and R. Astya, "An overview of deep learning architectures, libraries and its applications areas," in *International Conference on Advances in Computing, Communication Control and Networking (ICACCN)*, 2018, pp. 293–297.

[39] S. Raschka and M. Vahid, *Machine Learning mit Python und Scikit-Learn und TensorFlow*. Frechen: mitp, 2017.

[40] A. Bonner. (2019). The complete beginners guide to deep learning, [Online]. Available: `https://towardsdatascience.com/intro-to-deep-learning-c025efd92535` (visited on 07/28/2019).

[41] J. Wu. (2019). Ai, machine learning, deep learning explained simply, [Online]. Available: `https://towardsdatascience.com/ai-machine-learning-deep-learning-explained-simply-7b553da5b960` (visited on 01/31/2020).

[42] M. F. Hussin and M. Kamel, "Document clustering using hierarchical SOMART neural network," in *Proceedings of the International Joint Conference on Neural Networks*, ser. 3, 2003, pp. 2238–2242.

[43] T. Braun, "Übertragungsmedien," Universität Bern, 2017, (visited on 08/23/2019).

[44] A. Ng, *CS 229 lecture notes*, Jun. 28, 2019. [Online]. Available: `http://cs229.stanford.edu/summer2019/cs229-notes1.pdf` (visited on 08/02/2019).

[45] Y. Upadhyay. (2019). Introduction to FeedForward neural networks, [Online]. Available: `https://towardsdatascience.com/feed-forward-neural-networks-c503faa46620` (visited on 07/28/2019).

[46] D. E. Rumelhart and J. L. McClelland, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*, 1987, pp. 318–362.

[47] Y. LeCun, B. Boser, J. S. Denker, R. Howard, W. Habbard, L. D. Jackel, and D. Henderson, "Handwritten digit recognition with a back-propagation network," in *Advances in Neural Information Processing Systems 2*, 1990, pp. 396–404.

[48] S. Saha. (2018). A comprehensive guide to convolutional neural networks — the ELI5 way, [Online]. Available: `https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53` (visited on 08/03/2019).

[49] D. Wang and J. Chen, "Supervised speech separation based on deep learning: An overview," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 10, pp. 1702–1726, 2018.

[50] Y. Hua, Z. Zhao, R. Li, X. Chen, Z. Liu, and H. Zhang, "Deep learning with long short-term memory for time series prediction," *IEEE Communications Magazine*, vol. 57, no. 6, pp. 114–119, 2019.

[51] P. M. Nadkarni, L. Ohno-Machado, and W. W. Chapman, "Natural language processing: An introduction," *Journal of the American Medical Informatics Association*, vol. 18, no. 5, pp. 544–551, 2011.

[52] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," *30th International Conference on Machine Learning (ICML)*, pp. 1310–1318, 2012.

[53] E. Ahmed and M. H. Rehmani, "Mobile edge computing: Opportunities, solutions, and challenges," *Future Generation Computer Systems*, vol. 70, pp. 59–63, 2016.

[54] A. Agarwal, S. Siddharth, and P. Bansal, "Evolution of cloud computing and related security concerns," in *Symposium on Colossal Data Analysis and Networking (CDAN)*, 2016, pp. 1–9.

[55] A. Ahmed and E. Ahmed, "A survey on mobile edge computing," in *10th International Conference on Intelligent Systems and Control (ISCO)*, 2016, pp. 1–8.

[56] ETSI. (2016). Mobile edge computing (MEC): Technical requirements, [Online]. Available: `https://www.etsi.org/deliver/etsi_gs/MEC/001_099/002/01.01.01_60/gs_MEC002v010101p.pdf` (visited on 07/26/2019).

[57] M. T. Beck, M. Werner, S. Feld, and T. Schimper, "Mobile edge computing: A taxonomy," *The Sixth International Conference on Advances in Future Internet (AFIN)*, pp. 48–54, 2014.

[58] N. Takahashi, H. Tanaka, and R. Kawamura, "Analysis of process assignment in multi-tier mobile cloud computing and application to edge accelerated web browsing," in *3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*, 2015, pp. 233–234.

[59] Y. Zhang, H. Liu, L. Jiao, and X. Fu, "To offload or not to offload: An efficient code partition algorithm for mobile cloud computing," in *IEEE 1st International Conference on Cloud Networking (CLOUDNET)*, 2012, pp. 80–86.

[60] J. Dolezal, Z. Becvar, and T. Zeman, "Performance evaluation of computation offloading from mobile device to the edge of mobile network," in *IEEE Conference on Standards for Communications and Networking (CSCN)*, 2016, pp. 1–7.

[61] O. Salman, I. Elhajj, A. Kayssi, and A. Chehab, "Edge computing enabling the internet of things," in *IEEE 2nd World Forum on Internet of Things (WF-IoT)*, 2015, pp. 603–608.

[62] X. Sun and N. Ansari, "EdgeIoT: Mobile edge computing for the internet of things," *IEEE Communications Magazine*, vol. 54, no. 12, pp. 22–29, 2016.

[63] (2019). Vehicle-to-everything communication, Nokia, [Online]. Available: `https://www.nokia.com/networks/solutions/vehicle-to-everything-communication/` (visited on 07/26/2019).

[64] R. Nazmul and H. U. Zaman, "Radio network optimization," in *International Conference on Advances in Electrical Engineering (ICAEE)*, 2015, pp. 194–197.

[65] P. Kamaraju, "Towards content delivery optimization in future wireless networks," in *IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2016, pp. 1–3.

[66] R. S. Kshetrimayum, “An introduction to UWB communication systems,” *IEEE Potentials*, vol. 28, no. 2, pp. 9–13, 2009.

[67] H. Nikookar and R. Prasad, *Introduction to Ultra Wideband for Wireless Communications*. Dordrecht: Springer, 2009.

[68] X. Shen, M. Guizani, R. C. Qiu, and T. Le-Ngoc, Eds., *Ultra-Wideband Wireless Communications and Networks*, Hoboken: John Wiley & Sons, 2007.

[69] A. Alarifi, A. Al-Salman, M. Alsaleh, A. Alnafessah, S. Alhadhrami, M. Al-Ammar, and H. Al-Khalifa, “Ultra wideband indoor positioning technologies: Analysis and recent advances,” *Sensor*, vol. 16, no. 5, pp. 1–36, 2016.

[70] M. Ghavami, L. Michael, and R. Kohno, “Basic properties of uwb signals and systems,” in *Ultra Wideband Signals and Systems in Communication Engineering*, 2007, pp. 9–24.

[71] J. Lee, Y. Su, and C. Shen, “A comparative study of wireless protocols: Bluetooth, uwb, zigbee, and wi-fi,” in *IECON 2007 - 33rd Annual Conference of the IEEE Industrial Electronics Society*, 2007, pp. 46–51.

[72] L. Parv. (2017). The story behind UWB technology and indoor positioning, [Online]. Available: `https://www.eliko.ee/uwb-technology-indoor-positioning/` (visited on 04/22/2020).

[73] Z. Farid, R. Nordin, and M. Ismail, “Recent advances in wireless indoor localization techniques and system,” *Journal of Computer Networks and Communications*, vol. 2013, pp. 1–12, 2013.

[74] G. Cheng, “Accurate toa-based uwb localization system in coal mine based on wsn,” *Physics Procedia*, vol. 24, pp. 534–540, 2013.

[75] M. Segura, V. Mut, and C. Sisterna, “Ultra wideband indoor navigation system,” *Radar, Sonar Navigation, IET*, vol. 6, pp. 402–411, 2012.

[76] R. Ye, S. Redfield, and H. Liu, “High-precision indoor uwb localization: Technical challenges and method,” in *2010 IEEE International Conference on Ultra-Wideband*, 2010, pp. 1–4.

[77] L. Mucchi, F. Trippi, and A. Carpini, “Ultra wide band real-time location system for cinematic survey in sports,” in *2010 3rd International Symposium on Applied Sciences in Biomedical and Communication Technologies (ISABEL 2010)*, 2010, pp. 1–6.

[78] M. McCracken, M. Bocca, and N. Patwari, “Joint ultra-wideband and signal strength-based through-building tracking for tactical operations,” in *2013 IEEE International Conference on Sensing, Communications and Networking (SECON)*, 2013, pp. 309–317.

[79] L. Jiang, L. N. Hoe, and L. L. Loon, “Integrated uwb and gps location sensing system in hospital environment,” in *2010 5th IEEE Conference on Industrial Electronics and Applications*, 2010, pp. 286–289.

[80] (2019). Raspberry pi: Product specification, [Online]. Available: `https://www.raspberrypi.org/products/raspberry-pi-3-model-b/` (visited on 08/15/2019).

[81] (2019). Grove base hat for raspberry pi, [Online]. Available: `http://wiki.seeedstudio.com/Grove_Base_Hat_for_Raspberry_Pi/` (visited on 08/15/2019).

[82] (2019). Grove temperature and humidity sensor, [Online]. Available: `http://wiki.seeedstudio.com/Grove-TemperatureAndHumidity_Sensor/` (visited on 08/15/2019).

[83] (2019). Sequitur InGPS lite, [Online]. Available: `https://www.unisetcompany.com/download/sequitur-ingps-lite-complete` (visited on 08/15/2019).

[84] I. Fette and A. Melnikov, "The WebSocket protocol," *Internet Engineering Task Force (IETF)*, vol. 6455, pp. 1–71, 2011.

[85] (2019). Autobahn documentation, [Online]. Available: `https://autobahn.readthedocs.io/en/latest/index.html` (visited on 08/16/2019).

[86] (2019). Python django framework, [Online]. Available: `https://www.djangoproject.com` (visited on 08/23/2019).

[87] (2019). Keras documentation, [Online]. Available: `https://keras.io/` (visited on 08/15/2019).

[88] (2019). Tensorflow documentation, [Online]. Available: `https://www.tensorflow.org/` (visited on 08/15/2019).

[89] (2019). Sunfounder picar, [Online]. Available: `https://www.sunfounder.com/picar-s-kit.html` (visited on 01/31/2019).

[90] W. Badr. (2019). 6 different ways to compensate for missing values in a dataset, [Online]. Available: `https://towardsdatascience.com/6-different-ways-to-compensate-for-missing-values-data-imputation-with-examples-6022d9ca0779` (visited on 05/03/2020).

[91] (2019). Documentation for app developers, [Online]. Available: `https://developer.android.com/reference/android/hardware/SensorManager` (visited on 08/13/2019).

[92] M. Wenger, "Indoor positioning using raspberry pi with UWB," Bachelorarbeit, Universität Bern, Bern, 2019.

[93] E. Karapistoli, F.-N. Pavlidou, I. Gragopoulos, and I. Tsetsinas, "An overview of the IEEE 802.15.4a standard," *IEEE Communications Magazine*, vol. 48, no. 1, pp. 47–53, 2010.