Delft University of Technology
Master of Science Thesis in Embedded Systems

LoRa Localisation in Cities with Neural Networks

Tuan Anh Nguyen

TU Delft Delft University of Technology

LoRa Localisation in Cities with Neural Networks

Master of Science Thesis in Embedded Systems

Embedded and Networked Systems Group
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Mekelweg 4, 2628 CD Delft, The Netherlands

Tuan Anh Nguyen
t.a.nguyen-2@student.tudelft.nl
anhnguyentuan1610@gmail.com
TU Delft Student Number: 4742613

26/09/2019

Author
Tuan Anh Nguyen (t.a.nguyen-2@student.tudelft.nl)
(anhnguyentuan1610@gmail.com)
(**TU Delft Student Number:** 4742613)

Title
LoRa Localisation in Cities with Neural Networks

MSc Presentation Date
03/10/2019

Graduation Committee

Dr Fernando Kuipers (chairman)	Delft University of Technology
Dr Marco Zuniga (direct supervisor)	Delft University of Technology
Dr Claudia Hauff	Delft University of Technology

Abstract

Billions of wireless devices are interconnected to provide services to many aspects of life and form The Internet of Things. These devices which are often battery-powered and energy efficient can benefit greatly from an accurate localisation service that does not consume extra energy. Several localisation methods have been developed for Low-Power Wide-Area Networks (LPWANs), with LoRa being of particular interest thanks to its long range and cost effectiveness. Time Difference of Arrival (TDoA) is a common way to find location in a LoRa network which works well in open areas but poorly in the harsh radio environment of cities. In indoor settings where the radio environment is more complicated than outdoor, RSSI fingerprinting techniques have been sucessfully used for positioning using WiFi and Bluetooth, with state-of-the-art solutions employing Artificial Neural Networks (ANN).

This work aims to provide accurate localisation in an urban LoRa network, using an ANN-based fingerprinting approach. Two publicly available data sets collected in the cities of Utrecht and Antwerp are used to evaluate our method. We show that the ANN model can be trained on these data sets to predict location with mean errors between $411m$ and $581m$. We determine that the presence of gateways in the fingerprint plays a major role in the ANN's estimation but RSSI information is crucial in improving the accuracy. To realistically compare the ANN approach to TDoA, we train and test the neural network with chronologically split data. Our ANN approach achieves a mean error of $500m$ with 90% of cases having errors below $1070m$. This RSSI fingerprinting method is more effective than TDoA at limiting large localisation errors in cities.

Acknowledgement

I would like to express my sincere gratitude to my daily supervisor, Dr. Marco Zuniga, for his continuous support throughout the course of this work. His motivation, enthusiasm, and expert insights have helped me immensely in the research and writing of this thesis. I would like to thank Dr. Fernando Kuipers for his valuable expertise and for providing me with resources for the thesis. I would also like to thank my peers in the ENS group for their feedback and support whenever I presented my work. Finally, I wish to express my profound gratitude to my family and my partner for their dependable and loving support. Without their continuous engagement and encouragement throughout my years of study, this accomplishment would not have been possible.

Tuan Anh Nguyen

Delft, The Netherlands
26th September 2019

Contents

Chapter 1

Introduction

As frequently stated in literature, the Internet of Things (IoT) has seen steady growth in the number of connected devices in recent years, with many more expected in the near future. Some have even predicted over 20 billion devices will make up the IoT by the year 2020 [30, 44, 57]. These devices provide us with a multitude of applications, ranging from industrial and agriculture automation to smart light bulbs and toothbrushes. With this growth comes a demand for accurate localisation of wireless devices in both indoor and outdoor environment [17]. Not only will providing accurate localisation open the doors to a great number of new applications, doing so with existing technology is an attractive feature for the billions of devices already on the market. Many current solutions for IoT localisation are using GNSS, WiFi, and Bluetooth [38, 71] but these technologies all have significant drawbacks.

IoT devices are typically wireless and deployed in large quantities, thus they are generally required to be low cost and to consume little energy for extended lifetime. Both of these requirements make GNSS unsuitable for many IoT applications. The addition of a GNSS module also brings higher cost, so solutions that utilise the existing communication signal to provide localisation are preferred. Systems of this type have been developed for a range of wireless technologies.

LoRa is a long range, low power communication technology that is designed for IoT devices. Coupled with its low cost, this technology has been gaining popularity among the IoT landscape. With more than 100 LoRaWAN network operators in over 100 countries, localisation using LoRa can benefit a large number of devices and applications.

1.1 Problem Statement

There has been much research in providing accurate localisation with a wide range of wireless technologies, all having their strengths and weaknesses.

Solutions using LoRa and other long-range protocols are typically using ranging methods such as RSSI-ranging or Time-difference of Arrival (TDoA) to estimate distance and infer location. Ranging methods work well when devices have a clear line-of-sight but their accuracy deteriorates in urban locations. At the same time, RSSI fingerprinting techniques have been successful in overcoming the harsh radio environment found inside buildings, with state-of-the-art solutions using Artificial Neural Networks. An RSSI fingerprinting solution for outdoor localisation could therefore overcome the challenges that ranging methods face.

While some research has been done on localisation with Low-Power Wide-Area Networks (LPWAN) using RSSI fingerprinting or machine learning, many of these works were able to report low errors by limiting their solution to specific settings (e.g. small area, stationary devices, etc.). Few works have attempted to create a solution for more general use such as localisation in a city-wide area. The algorithm *k-nearest neighbours* is popular among these works and more powerful techniques such as Artificial Neural Networks (ANN) have been left largely unexplored.

This thesis aims to: *Investigate the feasibility of using Artificial Neural Networks to estimate the location of devices in a LoRa Wide-Area Network in an urban environment.*

This work made use of two data sets that are publicly available. One was provided by TTN Mapper, a community effort to map the global coverage of The Things Network's LoRaWAN. The other was published as part of a study, collected in a LoRa network operated by Proximus in Belgium [1].

1.2 Contributions

The key contributions of this thesis are:

- Development of Neural Network models for LoRa node localisation with data sets gathered in the cities of Utrecht and Antwerp (Chapter 4). The ANNs can learn from these sets and estimate locations with mean errors of $581m$ and $381m$ for Utrecht and Antwerp, respectively.
- Comparison of accuracy between our approach and TDoA, using a chronological split of our data sets (Chapter 5). We train the models with data in the past and evaluate their performance with data nearer to the present. With the Antwerp data set, the ANN achieves a mean error of $480m$.

1.3 Organisation

Chapter 2 presents the background knowledge and research works related to this thesis. Specifically, the LoRa communication protocol and the LoR-

aWAN network protocol are discussed, then an overview of existing localisation solutions is provided, and finally background knowledge on Artificial Neural Networks is given. Chapter 3 provides details of the software and the data sets that are used in this thesis. Chapter 4 delves into the steps taken to pre-process the data and selecting a suitable model for an ANN. In chapter 5, we study how different aspects of a LoRaWAN can influence the ANN performance. We also provide a better comparison between our approach and TDoA, by chronologically splitting our data sets. Finally, chapter 6 provides conclusions and potential future work related to this thesis.

Chapter 2

Background and Related work

2.1 LoRa and LoRaWAN

This section provides an overview of LoRa communication which consists of the LoRa physical layer and the LoRa Wide-Area Network (LoRaWAN) protocol. LoRa is a modulation technique developed by Cycleo and acquired by Semtech in 2012 [66]. LoRaWAN is a network protocol that allows LoRa devices to connect with applications over the Internet[47].

LoRa is a radio modulation scheme that utilises Chirp Spread Spectrum (CSS) to be robust against noise and reference, extending its range [56]. The current record for the longest distance a LoRa packet has been received is $766km$, set on the 13^{th} of July, 2019 [65]. Its low power consumption makes it even more attractive to IoT applications.

The LoRa Alliance of over 500 member companies developed the LoRaWAN specification to allow LoRa devices to connect with the Internet and its applications [47]. LoRaWAN defines the network architecture and the communication protocol used in the network [63]. Figure 2.1 illustrates the network topology of a LoRaWAN.

A message from a LoRa node is broadcasted and can be received by multiple LoRa gateways. These gateways forward the message, along with some metadata recorded by the receiver, to the network server. Via the network server, users of the LoRaWAN can have their application server access the data from their nodes. It is thanks to the metadata and the fact that messages can be received by multiple gateways that localisation is at all possible in a LoRaWAN. With TDoA methods, the timestamps at which different gateways receive the same message is crucial. For fingerprinting, the RSSI is the most important metric. Thanks to its long range, LoRa messages may be received by many gateways; the more gateways that receive the same message, the better a localisation algorithm can estimate its location, re-

gardless of methodology [6]. The star topology of a LoRaWAN is similar to that of GNSS. Where a GNSS uses satellites as anchor points to calculate positions, a LoRaWAN solution can use gateways as anchors.

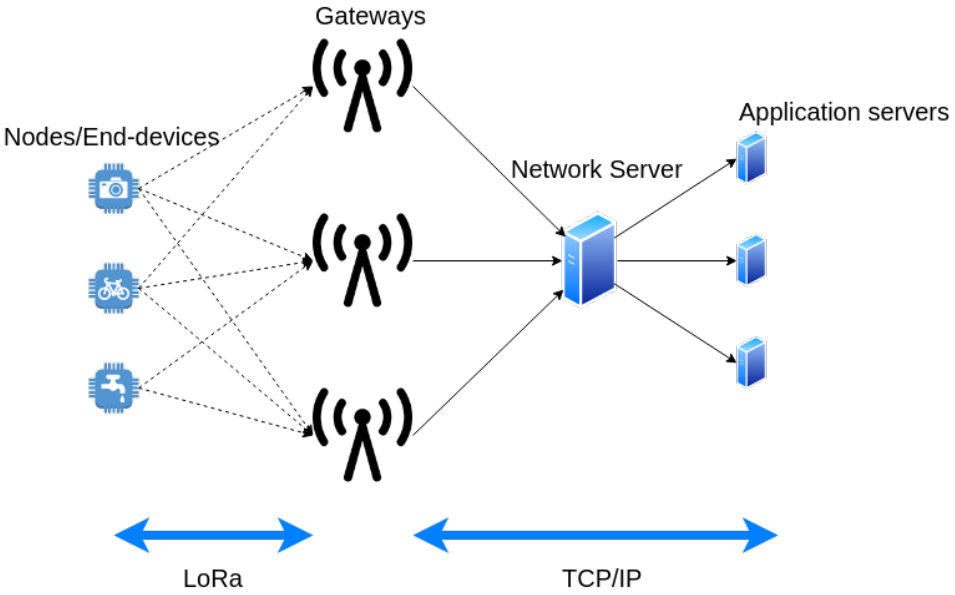


Figure 2.1: **Topology of a LoRaWAN.**

The Spreading Factor (SF) is an important parameter in LoRa that controls the data rate. A lower SF shortens the symbol duration and brings higher data rate as well as lower power consumption. The trade-off is that the signal becomes harder to decode. This trade-off effectively limits the range of devices that use lower SF. In a LoRaWAN, the SF can be used to provide Adaptive Data Rate (ADR). The ADR algorithm selects the lowest SF that still allows at least one gateway to receive messages from a LoRa device [53]. Therefore, LoRa devices with ADR enabled have a potential trade-off between data rate or battery life, and localisation accuracy.

2.2 Current solutions for localisation without GNSS

Satellite-based localisation technologies are currently the front-runner in accuracy for outdoors applications, with errors of less than $10m$ with GPS [14] and less than $4m$ with Galileo's Open Service [14]. On top of this, these systems utilise satellites and can thus provide continuous global coverage. This combination makes GNSS a very attractive solution for most applications. Unfortunately, for many devices in the IoT, GNSS modules consume too much energy. To give some perspective, a $250mAh$ battery can transmit 20000 LoRa messages, but will be depleted after 1000 GPS measurements [55]. Furthermore, IoT devices are often designed to be deployed in large

quantities; an additional GNSS module can drive up the cost significantly. A localisation solution which can use the communication modules already embedded in IoT designs is therefore highly desirable.

2.2.1 Range-based localisation

In the IoT, as well as in wireless sensor networks in general, many localisation techniques have been researched. A large number of these algorithms can be categorised as range-based, where distances are calculated using quantities such as Time-of-Arrival (ToA), Time-Difference-of-Arrival (TDoA), RSSI ranging, etc. Using multiple anchor points with known coordinates, the location of the transmitting node can be calculated using Trilateration, Triangulation, or multilateration [12, 43]. In an LPWAN such as LoRaWAN, the gateways may be used as anchor points.

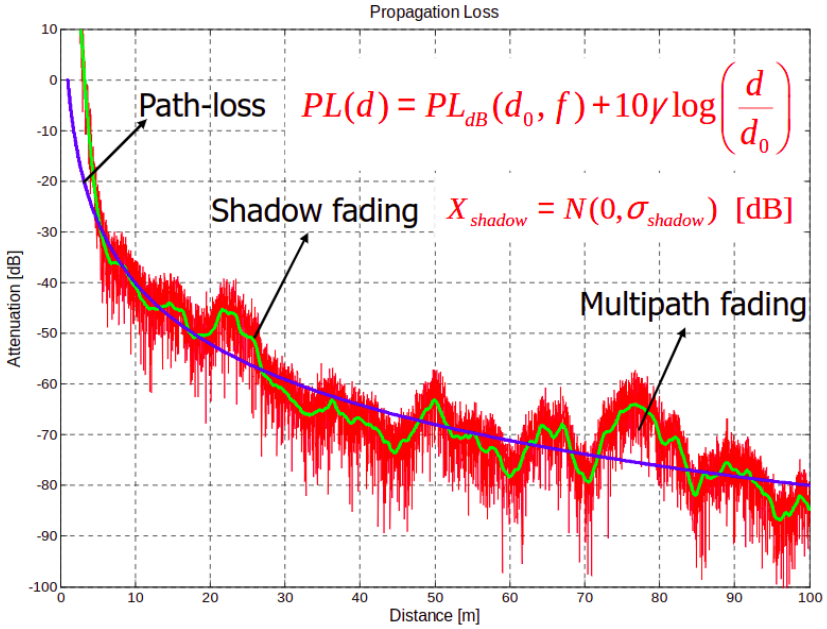


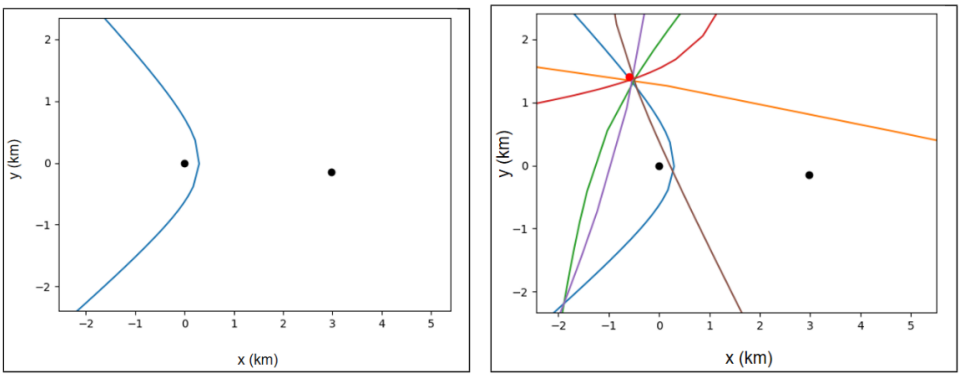
Figure 2.2: **Different types of losses in a wireless communication channel[19].**

In RSSI ranging, the distance between transmitter and receiver is estimated using the received signal strength and a signal propagation model. Figure 2.2 shows the effects of different types of losses in a wireless channel. In an ideal line-of-sight situation with no interference, the signal is attenuated following the path-loss curve. However, in realistic scenarios, shadow fading and multipath fading make the loss profile much more complicated and unpredictable [20, 52]. For example, in an ideal channel, an attenuation of $-70dB$ to the signal power means the transmitter is about $60m$ away

from the receiver. Due to multipath fading, this distance can vary between $30m$ and $90m$. Therefore, RSSI ranging localisation does not achieve high accuracy except for in open areas. Sigfox implements this method and could achieve an accuracy of less than $10km$ for 80% of cases with static devices.

ToA and TDoA methods require accurate time-synchronisation between devices. TDoA only requires that gateways have accurate timestamps and not the node which makes it better suited for IoT devices. Accurate and synchronised timing between gateways are essential in this scheme, because a small error of $1\mu s$ may correspond to an error of $300m$[6].

Another common method for ranging is using Time-Difference of Arrival (TDoA). When a LoRa node transmits, several gateways can receive the same message. Since each gateway is at a different distance away from the node, they will receive the message at different times. For each pair of gateways, we can get a measurement of this time difference. From this measurement, a collection of possible node locations can be calculated. If at least 3 gateways received the same transmission, we can form multiple hyperbolae that ideally would intersect at the position of the node [24]. Figure 2.3 shows a simulation of this method in measuring the distance difference and inferring a location using multiple measurements. TDoA requires accurate timestamps at the gateways. In a practical setting, interference such as error in timing can introduce many uncertainties. A timing error of $1\mu s$ may lead to an additional $300m$ localisation error [6].



(a) (b)

Figure 2.3: **Left: a pair of gateways (black) that received the same message, the hyperbola of possible node locations can be found from the TDoA measurement of this pair. Right: With multiple gateway pairs, resulting hyperbolae intersect at the node's position [6].**

The LoRa Alliance provides several case studies where TDoA has been used in LoRaWANs [46]. Many of these achieved good accuracy of under $100m$ in most cases. However, most of these only considered nodes that are

strictly inside a boundary created by the gateways used as anchor points. The cases that studied nodes beyond this perimeter found that performance degrades significantly. Accuracy is also better in all cases where the node is stationary.

Kim, S et al. applied TDoA method to a LoRaWAN and reported error above $300m$ [40]. Fargas et al. later achieved a significant improvement, with an error of about $100m$ for static nodes, using an iterative algorithm [15]. Podevijn et al. used TDoA to estimate location in the city of Eindhoven when 3 different modes of transport were used: walking, cycling, and driving [51]. A median error $200m$ was reported and significant improvements can be achieved when they applied an algorithm which takes map details such as roads into account.

	Median error	**Setting**
Fargas et al.[15]	$100m$	Stationary nodes, private LoRaWAN
Podevijn et al.[51]	$200m$	Mobile nodes, accuracy improved by using road details
KPN[6]	$174m$	Mobile nodes, large service area
Bissett[6]	$500.4m$	Mobile nodes, large service area

Table 2.1: **Comparison of median error and settings of TDoA research.**

In the Netherlands, the network operator KPN has deployed a proprietary solution using TDoA in their LoRaWAN [41]. KPN claims this algorithm offers accuracy within $100m$ for 90% of cases with static devices, in the Netherlands. Bissett used the KPN network to develop a modified TDoA approach for localisation and compared their result with what KPN's solution achieved [6]. Figure 2.4 shows the performance of these methods, using data gathered in the same study. For up to 75% of cases, KPN's algorithm is better, but Bissett's method is more effective in limiting the higher errors. Table 2.1 summarises the results of these works on TDoA with LoRA.

Similar to RSSI ranging, a major challenge for TDoA methods is the environment surrounding the location, especially with multipath. While a multipath channel varies the received signal strength in RSSI ranging, it affects TDoA measurements in a different manner. TDoA assumes a direct line-of-sight between the transmitter and receiver; in an urban area, this line-of-sight path may be completely blocked and the signal reaches the receiver after reflecting off a surface. Since this signal would take longer to reach the receiver, TDoA can overestimate the distance. A potential solution for this issue is to model the environment and take that model into account. Works such as those presented in [4] attempt to map the coverage of wireless technologies can provide a good basis for this. Another

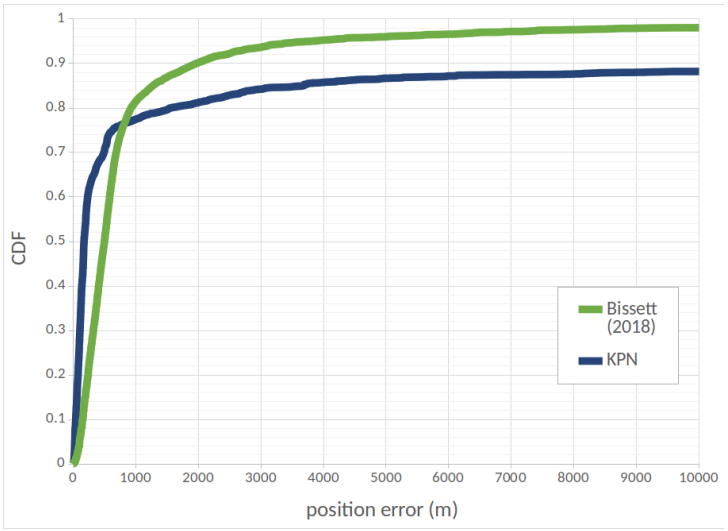


Figure 2.4: **Localisation error of Bissett (2018) compared with KPN's proprietary algorithm. [6]**

method that may overcome this challenge is RSSI fingerprinting, where a large number of transmissions are pre-recorded with known position, and localisation is achieved by matching a new reception with the record.

2.2.2 RSSI Fingerprinting

Much research has been done on using RSSI data to perform localisation but the wireless technologies and algorithms varies. A fingerprinting-based solution typically consists of an offline training phase and an online phase[13]. In the training phase, signal samples are collected in the area where localisation should be provided and an algorithm is then trained using this data. This area where data is collected shall be referred to as the *service area* throughout the rest of this thesis. In the online phase, the location of a new data point can be estimated by the algorithm using the signal characteristics. The first phase of this process where a lot of data needs to be collected presents a major challenge for this approach, especially for outdoor localisation because much time and effort are required to gather data for a large area. This is a reason why this method has mainly been used for indoor applications. Another reason is that the radio environment inside building is complicated, so ranging methods do not work well. If RSSI fingerprinting can overcome the multipath challenge inside building, it may be able to do so for an urban environment as well.

WiFi has been a popular choice for localisation thanks to its ubiquity[67, 26, 27, 7]. Many handheld devices are equipped with a WiFi interface and WiFi access points can be found all around, especially inside buildings. Fur-

thermore, the WiFi protocol makes data collection a simple task in public networks thanks to its Beacon Frame. Access points periodically send out a beacon frame to announce its presence and can be received by any device with a WiFi interface[36]. The fingerprint is made up of RSSI values from these beacon frames as received by the device. This differs from networks such as LoRaWAN, where the RSSI fingerprint is made up of receptions from multiple gateways. In a WiFi network with a fingerprinting localisation, a smartphone can record the fingerprint and find its own location via a web service for example. In a LoRaWAN, only the network server has the fingerprint data and can determine the node's location. This is not necessarily a limitation as many IoT applications do not require the device to know its location. Figure 2.5 visualises this difference between WiFi and LoRa.

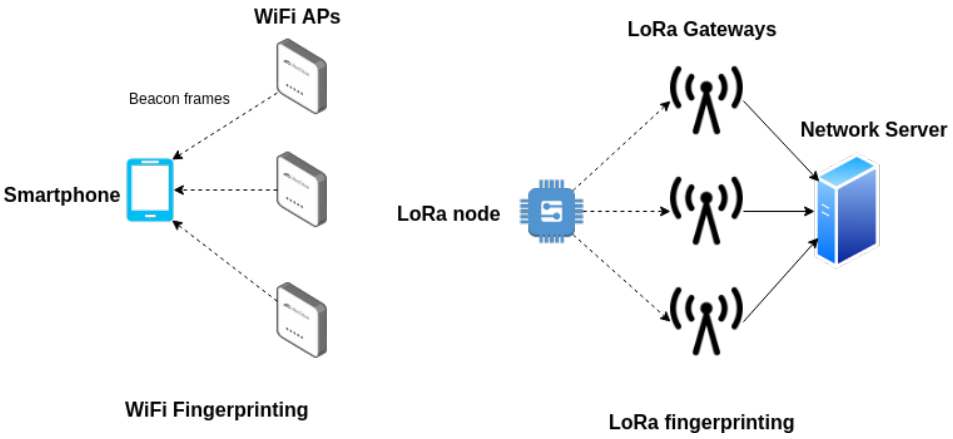


Figure 2.5: **The difference between WiFi (left) and LoRa (right) fingerprinting. In a WiFi network, the RSSI fingerprint can be recorded by the device, such as a smart phone. In a LoRaWAN, the fingerprint is compiled by the network server, from RSSI values at all gateways that received the same message.**

Many machine learning algorithms have been used to provide RSSI fingerprinting localisation for WiFi networks such as: k-nearest neighbours [67, 3, 29, 70], Artificial Neural Networks [68, 8], Support Vector Machines [69], Random Forest [54], etc. Other technologies that have been researched for indoor localisation using RSSI fingerprinting include Bluetooth[16, 45], RFID [11], etc. There are also works on hybrid solutions such as [34] which is based on WiFi and sound signals.

Thanks to the recent rise in popularity of LPWANs, there has been more research in outdoor localisation. However, many of these works focus on specific settings and applications. Gotthard et al. developed an RSSI fingerprinting method with LoRa to locate used cars in a dealership and reported an error within 10 to 20 meters [22]. While this is good result, their applic-

ation is limited to stationary objects and the small area of a car park. Choi et al. generates fingerprint maps from sample points, then using these maps to predict location [13]. They reported a mean error of $24.1m$ in an experiment covering an area measuring $340m \times 340m$ with 4 LoRa gateways. Bibb et al. simulated the signals using a ray-tracing algorithm the authors developed [5]. They considered a building map of a part of Singapore for their experiment. While they reported a very low mean error of $2.8m$, this result is unlikely to be representative of a real scenario, where other interference such as cars, trees and other signal sources may be present.

More recently, there have been research efforts in general-purpose RSSI fingerprinting applications. Aernouts et al. published three data sets gathered for LPWAN in and around the city of Antwerp [1]. Two of these data sets were gathered with Sigfox and the third used LoRa. Janssen et al. reported a mean error of $340m$ using a k-Nearest Neighbours algorithm on one of the Sigfox data sets [31]. In this work, we used the LoRa data from [1] as one of two data sets to apply our method. Because results for LoRa RSSI fingerprinting are limited in literature, we will compare the performance of our solution with that of TDoA methods. During the final month of this thesis, a preprint [2] was made available that used ANN technique on the Antwerp data set, we will also discuss this work in later chapters. Table 2.2 summarises the results of research on RSSI fingerprinting in LPWAN.

	Error	**Setting**
Gotthard et al.[22]	$10m$ - $20m$	Stationary nodes, small area
Choi et al.[13]	$24m$ mean	Fingerprint map generation, small area
Bibb et al.[5]	$2.8m$ mean	Data from ray-tracing
Janssen et al.[31]	$340m$ mean	Large area, Sigfox data set

Table 2.2: **Accuracy of RSSI fingerprinting in LPWAN from related research.**

Like every system, RSSI fingerprinting methods have certain limitations.

- **Ground-truth inaccuracy:** RSSI fingerprints gathered for training an algorithm use GNSS solutions to provide the ground-truth coordinates. Since GNSS have inaccuracies, RSSI fingerprinting may never perform better than satellite-based systems.

- **Requires large data sets:** For an RSSI fingerprinting algorithm to learn and perform well, large data sets are needed. Collecting data samples to train a solution in a large service area is a task that can take a lot of time and effort.

2.3 Neural Network for RSSI fingerprinting

As discussed in section 2.2.2, the first phase of a fingerprinting method is collecting sample data and training an algorithm. Since this data must have the ground-truth location determined by GNSS, a supervised learning algorithm is a suitable choice. The fingerprints will serve as inputs to the algorithm which should learn to predict the output as a representation of geographical position. A supervised learning algorithm can tackle two sets of problems: *classification* and *regression*.

- **Classification:** The algorithm learns to predict the right categorical label for an input. An example of a classification application is a spam filter for emails which labels each email as 'spam' or 'not spam'. Another example is predicting in which room of a building a mobile phone is, based on its WiFi fingerprint. In the case of outdoor localisation, the service area can be divided with a grid, and the algorithm predicts which grid cell the device is in.

- **Regression:** In a regression problem, the algorithm learns to predict the outputs as continuous values. For example, a machine learning algorithm can be trained to predict the value of a house given the location, size, and age of the property. Another example is an algorithm that predicts coordinate values, given the RSSI fingerprint from LoRa gateways, which is the goal of this thesis.

An Artificial Neural Network (ANN) can be used as a supervised learning algorithm. An ANN consists of many artificial neurons, an example of these neurons is shown in figure 2.6. The inputs and output of the neuron are numerical values and each input connection is associated with a weight. The neuron computes the weighted sum of its inputs and applies a function, called the *activation function*, to that sum which produces the output [23]. By connecting many artificial neurons, we can create an ANN.

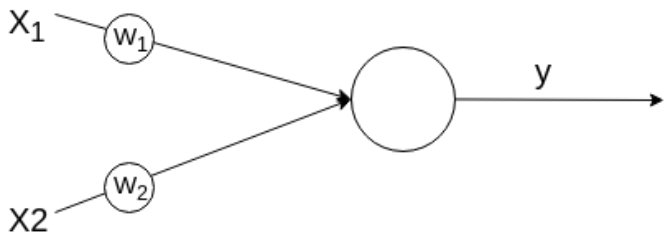


Figure 2.6: **An Artificial Neuron. The output of the neuron is calculated with the activation function of the neuron and the weighted sum of its inputs.**

There are different ANN architectures but we only consider the Multilayer Perceptron (MLP) in this work. An example of this is shown in figure 2.7; each neuron in one layer is connected to every neurons in the next layer. MLP is a simple but flexible architecture and can generally be used for a wide range of classification and regression problems [10]. If MLP can perform well, future works may focus on using other ANN architecture to further improve this approach.

Training the ANN is done by providing it an example input as well as the expected output. The ANN outputs a prediction and we can measure the error between this prediction and the expected output. The weights of the connections are then changed to reduce this error. With a large volume of training data, the ANN can learn and model patterns between the input and output so that it can reliably predict new data [23]. For localisation using ANN, we would provide the RSSI fingerprint at the input layer and the ANN should output an estimated position.

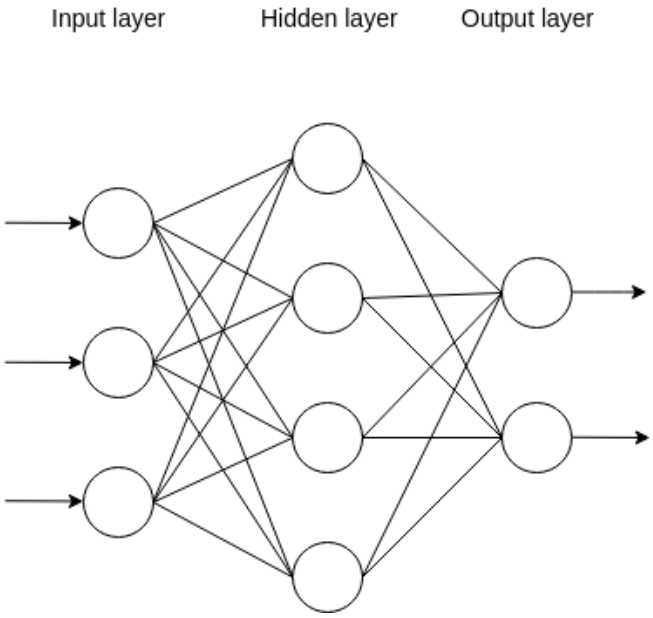


Figure 2.7: **A Multi-Layer Perceptron (MLP).**

We can influence the neural network's performance by tuning its *hyper-parameters*. These include the number of hidden layers and neurons, the learning algorithm, the learning rate, etc. A larger network may offer better performance but will also take more time to train. In selecting the number of hidden layer and neurons, we can start with the smallest network that can provide a reasonable performance. Then, more neurons and layers can be added until the neural network begins to *overfit* [50]. This mean the neural network has memorised the training samples and it is unable to predict new

fingerprints accurately. Now, we can focus on limiting the overfitting, by using regularisation.

Two common methods of reducing overfitting are: L1 or L2 regularisation and dropout. L1/L2 regularisation encourages the network to keep its weights small by penalising large weights [32]. Dropout excludes a number of neurons during each training step, making the neurons less reliant on each other, thus becoming less sensitive to small changes in their input [28, 58].

When we train a large ANN, the increased training time may pose a challenge. We can limit this time by using *early stopping*. This technique simply monitors the performance of the ANN after each training epoch and stops the training if the performance has not improved after some time. The best weights encountered during the training are then restored.

For training and evaluating an ANN and other supervised learning algorithms, it is recommended to split the available data into several sets [48]. These sets are typically: the *training set*, the *validation* set, and the *test* set. As the name suggest, the training set is the data used by the ANN to learn. The performance of a trained ANN can then be assessed using the validation set and design choices can be made to modify the ANN so that it performs better on the validation set. The final model can then be tested on the test set to provide a fair assessment of the model's performance. In other words, we use the validation set to select the architecture and tune our model and once a good model is trained, we evaluate its error using the test set. If we only had a training set and a test set, changing the architecture to perform well on the test set may produce a biased result since the test set was used as part of the design process [25].

Chapter 3

Tools and Data sets

Section 3.1 provides details on the tools used for developing the neural network models as well as the method for calculating the distances between points on the Earth's surface. The two data sets that were used in this work are discussed in section 3.2. The pre-processing steps performed on these data sets are discussed in details in chapter 4.

3.1 Toolkits used

Developed by Google for internal use and released publicly in 2015, *TensorFlow* is an open-source software library that is currently widely used for machine learning applications [60]. Running on top of *TensorFlow*, *Keras* is a library that provides high-level Application Programming Interfaces that allow for fast prototyping of neural network models [37]. These two pieces of software are used in this work to build and test the neural networks that will be discussed in chapter 4. The free machine learning library *scikit-learn* was also used for several of its useful functions.

geopy is a client for popular geocoding web services, aimed to simplify the process of locating the coordinates of addresses, cities, countries, and landmarks[18]. Geocoding is converting an address or landmark into geographical coordinates[21]. *geopy* contains a method to calculate the distance between two points on the surface of the Earth, which we used to calculate the localisation errors in this work. *geopy* provides two ways to calculate the distance between two points, using the geodesic distance or using the great-circle distance. The geodesic method uses an ellipsoidal model of the Earth, while the great-circle assumes the Earth is spherical. The geodesic distance is therefore, more accurate but takes more time to compute as it uses a more complicated model. Although the maximum error introduced by a spherical model is only 0.5% [49], the difference in computation time is insignificant. Thus, we used the geodesic distance for all of our error

analyses in this work. Specifically, we use the default geodesic algorithm in *geopy* which is given by Karney, Charles F.F. [35].

3.2 Datasets

Two data sets were used in this work. The first was acquired from TTN Mapper - a global community effort to map The Things Network's LoRaWAN coverage. The second was a data set collected and published by [1].

3.2.1 TTN Mapper - Utrecht

The Things Network (TTN) provides open-source tools as well as a global, open, and crowd-sourced network for IoT applications using LoRaWAN [61]. Currently there are around 8400 gateways across 141 countries connected to TTN. In The Netherlands alone, there are 869 gateways connected. Thanks to the country's relatively small size, these gateways provide a good LoRa coverage to the whole country, with many cities having over 100 gateways. TTN Mapper is an application that attempts to map the coverage of TTN gateways globally [64]. Users of TTN gateways can contribute to this effort by using LoRa devices with a GPS module or smartphones to measure the gateway's performance within its vicinity. All this data is aggregated and a global coverage map can be found on *https://ttnmapper.org/*.

The data on TTN Mapper is available for download. From them, we acquired a data set of LoRa transmissions in the city of Utrecht, The Netherlands for the period between August 2016 and April 2019. It should be noted that this data was gathered by many people for different applications, many of whom likely needed to measure the coverage for their own gateway in a specific area. Thus, this data may not fully and uniformly cover the area of the city. Furthermore, certain aspects are unknown such as the specific hardware used and the mode of transport. The data set also spans a rather long period of time (32 months) and the landscape of the city may have changed significantly. Nonetheless, this data is publicly available and provides a good basis to study RSSI fingerprinting in a LoRaWAN. If a good solution can be created for this data set, it can more easily be applied to other cities or regions since the data already exist.

Figure 3.1 shows a subset of data points from this data set on the city map. Due to limitations in the hardware and plotting software, this subset consisted of half the data points, selected at random. The entire data set is contained within an area of $160 km^2$. Several interesting elements can be seen from this map.

- **Clusters of data:** As expected, the data is not evenly distributed around the city. A large cluster of data points can be seen near the

central train station. Many data points were also gathered along the ring-road surrounding the city. Surprisingly, while the train station appears to be a hot-spot, not many data points were gathered along the railways. There are several smaller clusters of data around the city, indicating that many users measured the coverage in a small area of interest for their application only.

- **Lack of data for inner city:** In general, this data set lacks coverage of inner city streets and residential areas. Since most of the data points are clustered together, the fingerprinting algorithm may adapt very well to predicting data points near these clusters. The points that are scattered around the map will introduce some noise in the training phase, and are unlikely to predicted will high accuracy.
- **LoRaWAN outage:** The Leidsche Rijn tunnel along the west side of the ring-road shows a limitation of LoRa. Many data points can be seen at either exit of this tunnel, but very few are on the tunnel itself. While LoRa can provide long range communication, obstructions such as this can cause outage.

Chapter 4 discusses this data set in further details as well as describes the pre-processing steps used.

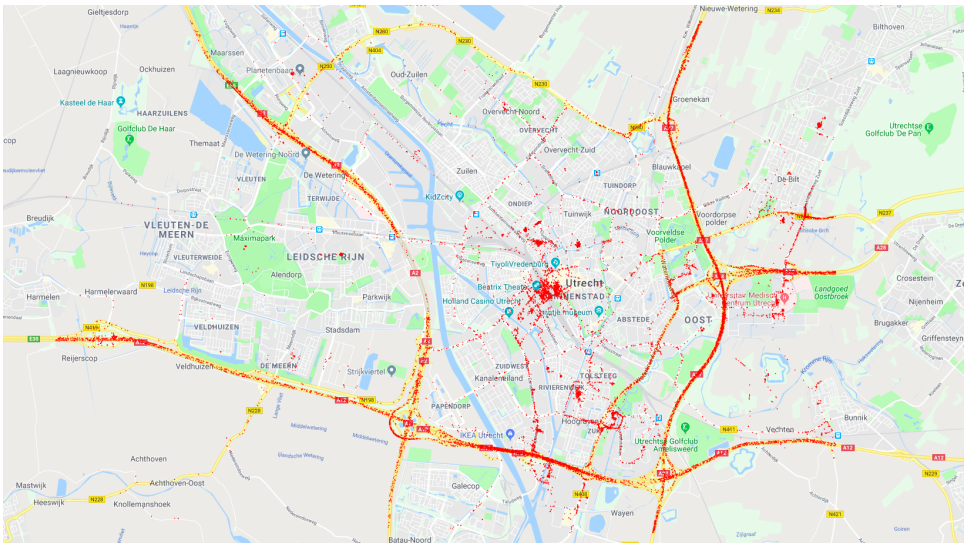


Figure 3.1: **A random subset of data points in Utrecht, acquired from TTN Mapper. The data is contained within an area of** $160km^2$

3.2.2 Antwerp

Aernouts et al. [1] published three data sets collected in and around the city of Antwerp. One of these was collected with a LoRaWAN. The author

gathered this data by attaching LoRa nodes to postal service vehicles for three months. During the collection period, these nodes were able to send a message every minute, giving a total of 123,529 data points. This is a similar amount of data compared to the Utrecht data set, which contains 140,593 messages after some pre-processing. Since this period is much shorter than the time frame of the Utrecht data set, the variation in the environment may not be significant between the data points.

The LoRaWAN used was deployed by a company called Proximus. Figure 3.2 shows a subset of this data on the map of Antwerp. This data set encapsulates a smaller area than that of the Utrecht dataset, covering only $52km^2$. Thanks to the methodological way of collection, this data set appears to be more evenly distributed between inner city streets.

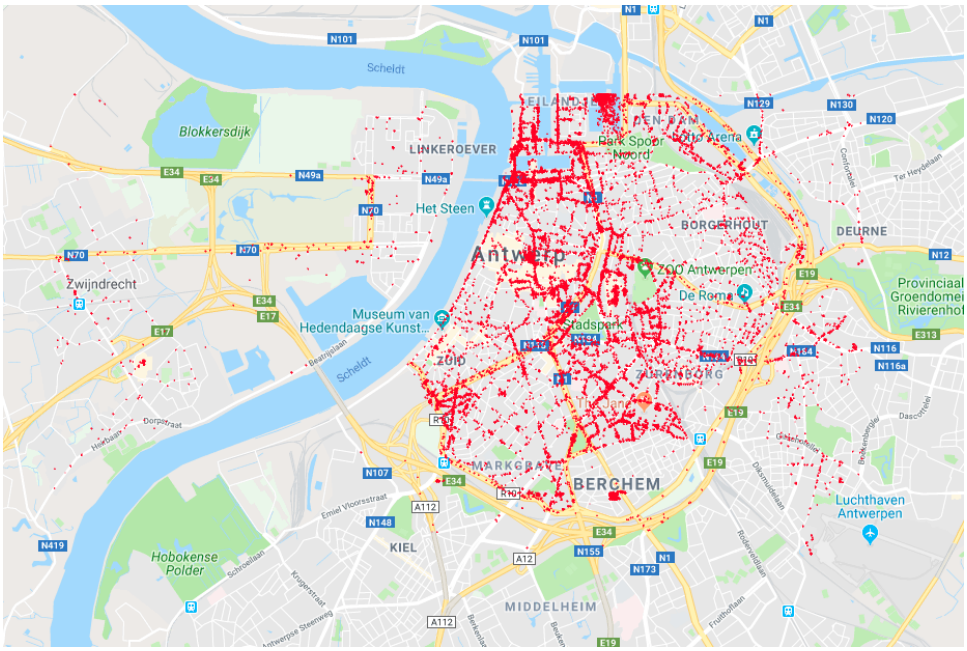


Figure 3.2: **A random subset of data points in Antwerp, published by [1].**

Chapter 4

Data Pre-Processing and ANN Model Selection

In this chapter, we first detail our approach to pre-process the data into a suitable format for training neural networks in section 4.1. Then, section 4.2 discusses the process of selecting, training and validating a suitable neural network model.

4.1 Data Pre-Processing

Utrecht

The data set for the Utrecht area from TTN Mapper contained a list of messages that were received by gateways in this region. Each line in this file is one reception of a message and contains several fields. The most important ones for our work are: the unique 'message ID'; the 'timestamp' at which this message was received; the 'node ID' of the LoRa device that sent the message; the 'gateway ID' that received the message; the 'received signal strength indicator' (RSSI); and the latitude and longitude of the LoRa node. From this file, we compiled a dataset that is suitable for neural networks and this process is detailed below.

Firstly, the data needed to be filtered from any message that was not useful for our neural networks. We removed two types of inconsistencies from the data:

1. **Many messages with the same timestamp:** a large number of messages all had the same timestamp at 2016-01-08 00:00:00. We discarded all of these receptions.

2. **Multiple locations for the same message:** There are groups of messages coming from the same nodes that have the same timestamp while also so having different locations. As a node cannot be in several locations at once, these groups were also discarded.

From the resulting data, we created the fingerprint dataset of this area. This dataset contains, for every message, the RSSI value at each gateway that received the message as well as the ground-truth coordinates. For every gateway that did not receive this message, we inserted the value of -200 as the RSSI. This step was done by sorting the data in time, then iterating through the data and finding groups of messages that were received at the same time by different gateways, while having the same 'node ID' and the same ground-truth coordinates.

Overall, 155 gateways were seen in this dataset and 140,593 data points were gathered.

Antwerp

The dataset for Antwerp from [1] had already been pre-processed into a similar format as the Utrecht dataset. Incidentally, the authors have also used a value of -200 to indicate when a gateway did not receive a message. This dataset contains 68 gateways and 123,528 data points.

Figure 4.1 shows the number of receptions for various number of gateways in both data sets. while some messages in the Utrecht data set were received by more than 7 gateways, the highest number of gateways that received the same message in Antwerp is only 6. Most of the receptions in Antwerp (92%) were also received by only 3 gateways. This surprising feature may be due to some selection process that is implemented by the network provider Proximus. In both data sets, most of the messages were received by only a small fraction of all the gateways in the service area. Apart from the possible limitation at the network server, this shows the effects that urban environments have in limiting the range of LoRa.

Figure 4.2 shows the distribution of RSSI values in our datasets. In section 3.2, we could see on the map that the Utrecht data set contains clusters of points. This is also reflected in this plot, where a portion of the data has RSSI values between $-80dBm$ and $-50dBm$. This suggests that some of the data was collected closer to gateways than the rest, showing again that some users were focused on gathering data for a small area.

For the neural network, we wish to use the ReLU activation function because it offers fast training using gradient-based training algorithms [42]. This activation function returns 0 for input values below 0, and returns the input value otherwise. Our raw RSSI data of negative values is thus not suitable. Furthermore, the absolute values of our data are large which can result in the weights of our network growing large and making the network unstable [9]. We rescaled the data to the range 0 to 1, using Min-Max Normalisation, shown in equation 4.1.

$$x_i = \frac{RSSI_i - RSSI_{min}}{RSSI_{max} - RSSI_{min}} \tag{4.1}$$

Where x_i is the normalised value, $RSSI_i$ is the raw RSSI value, and $RSSI_{min}$ and $RSSI_{max}$ are the minimum and maximum values of RSSI in the whole data set.

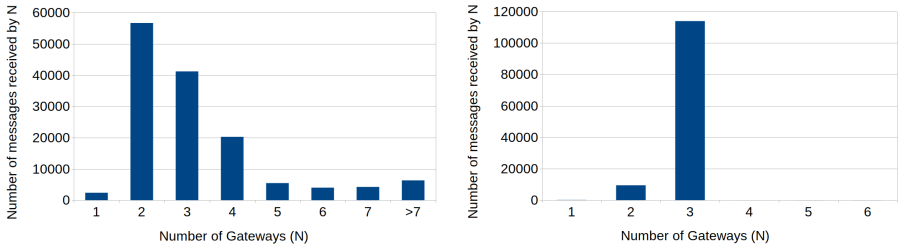


Figure 4.1: **Number of messages received by N gateways in Utrecht (left) and Antwerp (right).**

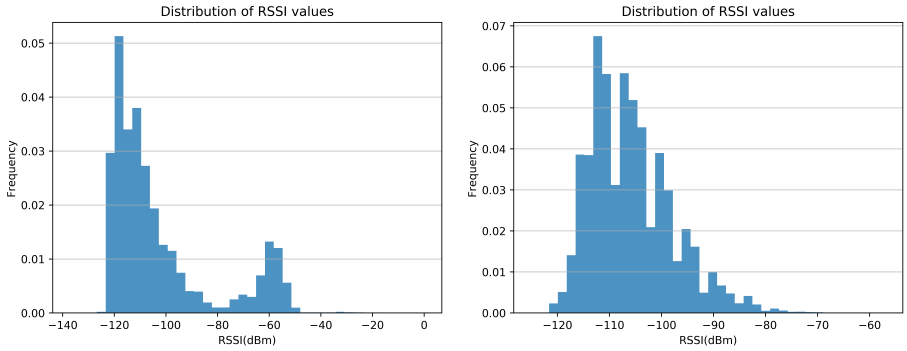


Figure 4.2: **RSSI Distribution of Utrecht (left) and Antwerp (right) datasets.**

Finally, for selecting and evaluating the performance of using Neural Networks on these data sets, we split the data into a training, a validation, and a test set containing 60%, 20%, and 20% of the data respectively. As discussed in chapter 2, the training set is used to train the neural network models and the performance of the trained model is evaluated with the validation set. The architecture of the neural network is altered based on its performance on the validation set. Once a model that can predict the validation set well is achieved, we can judge its performance on the test set.

At this stage, these data sets allowed us to study the use of ANN in learning and predicting RSSI fingerprint data. The process of designing the neural networks are discussed in section 4.2.

Splitting the data set in this manner does not take into account the fact that data was collected throughout an extended period of time. A real-world realisation of these algorithms would likely be trained on a set of data collected in the past, and be expected to predict new data points in

the present. In the next chapter, we will investigate the feasibility of using these methods in a real-world setting. For this, we split the data into subsets using timestamp instead. This way, the models can be trained with data in the past and tested with data collected later.

4.2 ANN Model Selection

In this section, the selection of the Neural Network model and hyperparameters is discussed. We consider the problem of localisation as both a regression and a classification problem. A Multilayer Perceptron (MLP) architecture where each layer is fully connected to the previous one was used. Figure 4.3 shows an example of the loss function over the training set, achieved by a selection of different network sizes for the regression problem. Adding a second hidden layer had a larger impact on the network's ability to learn from the training data than increasing the size of the first hidden layer. Networks larger than those shown in this plot did not achieve significantly better learning performance while taking much more time to train. Thus, we prioritise adding more layers instead of using very large layers.

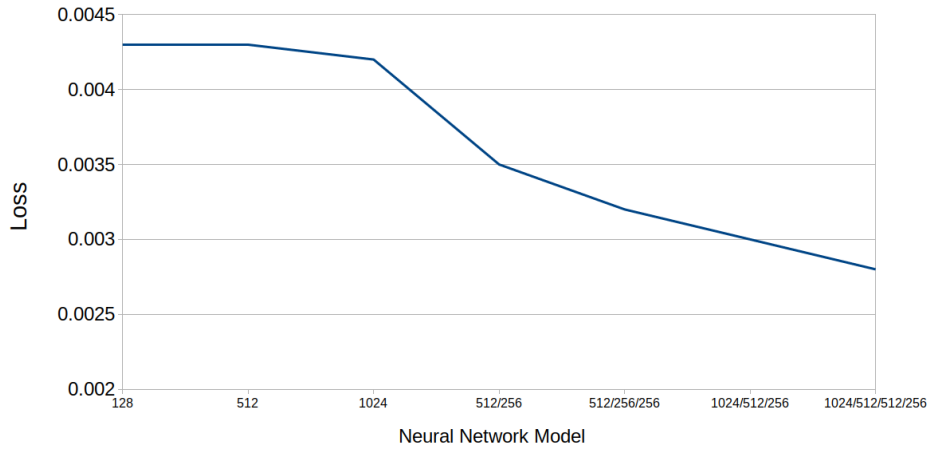


Figure 4.3: **Mean-squared error loss on the training set for different neural network sizes with Antwerp dataset.**

We start with training and evaluating a small network, then increasing the size by adding more neurons and layers until the network can learn the training data very well. The resulting neural network would not be able to predict the validation set well because it overfits the training set. In other words, the network simply memorises the training set and thus, not be able to correctly predict new data. Then we attempt to limit this overfitting so that it can predict the validation data.

To reduce overfitting, we use dropout. This technique selects a random set of neurons in a layer to be ignored during each training phase. This makes the neurons less reliant on each other as the connection between them are not always present during training time [59]. The dropout rate is the probability of a neuron to be ignored during a training phase and is a hyperparameter to be set for the model. A too high dropout ratio makes the network take longer to train and may not offer improved performance on the validation set.

We also employed early stopping in training our models. This technique simply stops the training if performance has not increased after a certain amount of time and restores the best weight values to the model. It monitors performance based on a loss function or a metric which can be calculated on the training or the validation set. The number of epoch to wait for before stopping is called the *patience* and we used a value of 50.

Table 4.1 shows the hyperparameter we selected for the classifier and regressor. In the rest of this section, we discuss other details surrounding our trained networks as well as their performance.

	Classifier	**Regressor**
Hidden layers' sizes	512/256/256	1024/512/256
Activation	ReLU with softmax output	ReLU
Loss Function	Cross-Entropy	Mean-squared Error
Optimizer	Adam	Adam
Dropout rate	0.15	0.2

Table 4.1: **Hyperparameters selected for classifier and regressor.**

Classification

For a classification model, the area covering the data points was divided with a grid into zones. These zones were numbered and the data points were labelled with the zone they are in. The trained classifier will output the predicted location of a node as its zone. We can expect the classifier to have lower prediction error with larger zones. As the result reported in [6] has median error of about $500m$, we can select a grid size that corresponds to this distance as a starting point. This means that the Utrecht area can be divided by a 20×20 grid, and Antwerp by a 10×10 grid. These correspond to zone sizes of $780m \times 510m$ and $940m \times 550m$ respectively. The classifier thus contains 400 output neurons for the case of Utrecht and 100 neurons for Antwerp.

In order to assess the performance of a classification model, we needed a way to measure the localisation error of each prediction. Instead of returning the predicted zone number, our classifier returns the centre location of the grid cell. The localisation error is then the distance between the actual

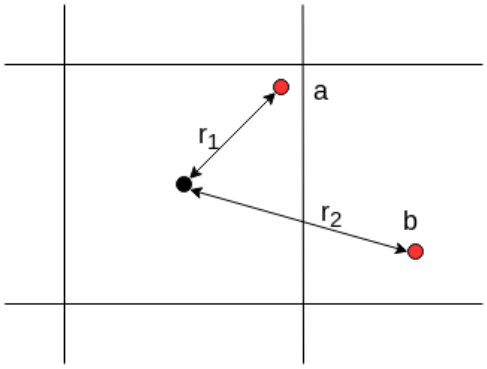


Figure 4.4: **Localisation error of classifier. The error is measured from the centre of the predicted zone (black) to the actual position of the point. r_1 and r_2 are the errors for points a and b, respectively.**

position and the centre of the cell. Figure 4.4 shows a visualisation of this error calculation. Figure 4.5 shows the classification error of this classifier on the two datasets, compared with results of KPN and [6].

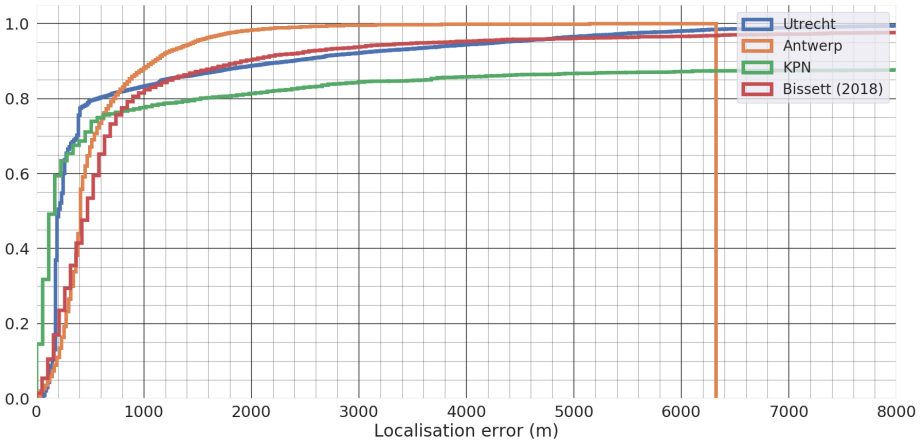


Figure 4.5: **CDF of localisation error, shown for Classifier Neural networks on both datasets and results from [6].**

The classifier appeared to perform well on the Antwerp dataset compared to the other methods from the 75^{th} percentile. It should be noted that the data for Antwerp was gathered inside a relative small area (of about $50km^2$) and thus, the maximum error that the classifier can get is limited by this size. The same is also true for the Utrecht dataset, although this set is inside a larger area (about $160km^2$).

Regression

In the regression neural network, the label used for each data point consisted of its latitude and longitude. The softmax layer is not needed. As the regressor predicts the output as latitude and longitude, the localisation error is simply the distance between the predicted location and the actual position.

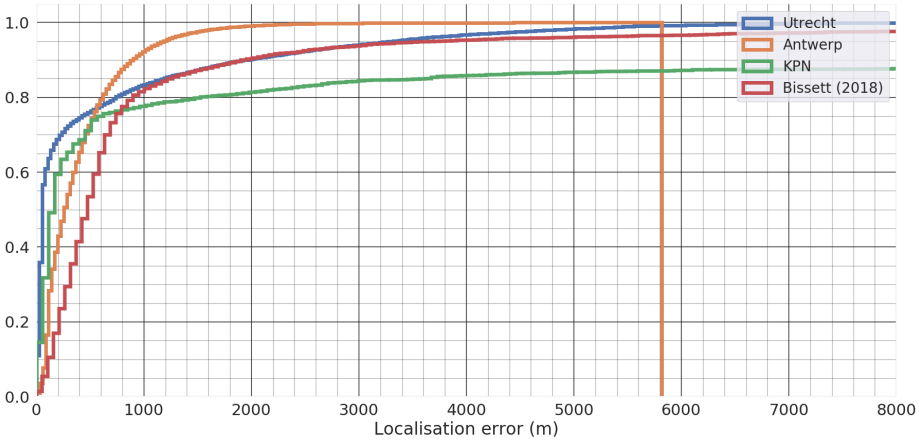


Figure 4.6: **CDF of localisation error, shown for Regression Neural networks on both datasets and results from [6].**

Figure 4.6 shows the error of our regressor on the two datasets, compared with results of KPN and [6]. Both the classifier and regressor achieved very low error for up to 80% of the test cases in Utrecht. The reason for this low error is due to the distribution of data in Utrecht:

- A large number of data points were collected in clusters and thus, the network was trained on a lot of data points from these clusters. This results in the network being able to predict points in and around these clusters very well.
- When the data set is split to make the test set, we randomly select the data points to be included. This means that a significant portion of the test set also came from these clusters. Because the network has learnt these clusters well, they can predict these points with high accuracy.

Figure 4.7 shows a comparison in accuracy between the classifier and regressor methods for both of our datasets. An overview of the performance, based on the median, mean, and 80^{th} percentile error of these models are given in table 4.2.

These models have provided us with a good basis to assess neural networks' performance in localisation. Both the regressor and classifier were

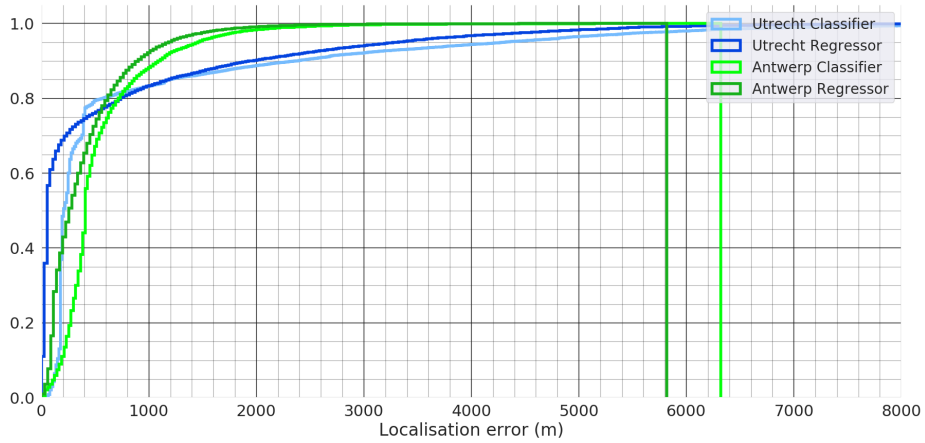


Figure 4.7: **Comparison of Classifier and Regressor performance.**

	Classification (m)		**Regression (m)**		**TDoA (m)**	
	Utrecht	Antwerp	Utrecht	Antwerp	KPN	Bissett
Median	216.4	419.9	**66.4**	277.4	174.2	500.4
Mean	779.2	553.9	581.1	**411.4**	6008.4	1667.1
80%	**580.1**	730.9	753.4	628.9	1557.1	924.7

Table 4.2: **Performance summary of the trained classifier and regressor compared to TDoA solutions.**

able to learn from the training data and predict the test data with adequate accuracy. Overall, the regression model offers better performance than the classifier.

The comparisons here only tell us that the neural network can learn to make predictions with these data sets. As the TDoA performances were evaluated with a different data set, and we cannot perform TDoA with our data due to inaccurate timestamps, our comparison may not be indicative of real-life performance. Furthermore, we expect that in a realistic setting, the accuracy of our method would not be the same, because a real system would learn from a data set in the past to predict location in real-time.

Chapter 5

Performance Assessment

This chapter discusses some characteristics of a LoRaWAN and how they contribute to the neural network's performance. First, we investigate how well the ANN can perform if we only provide the set of gateways that received a message and no RSSI is measured. Then, we consider two different methods of scaling the data that are more suitable for RSSI values than Min-Max Normalisation. We also look at the effect of using the Spreading Factor, a unique parameter of LoRaWAN, as an additional input. Finally, we split the training and test sets chronologically to more realistically assess our approach.

5.1 Reception without RSSI

Considering the fact that most of the receptions in our data sets were received by a small fraction of gateways as seen in figure 4.1, we wish to investigate whether the *combination of gateways* alone may provide a good location estimate with ANN. Pre-processing the data for this step is simply replacing every RSSI value with 1 and all non-receptions are set to 0. Table 5.1 summarises the result of this task for the Antwerp data set. Figure 5.1 shows a comparison between this result and TDoA performance.

	Median (m)	Mean (m)	90^{th} Percentile (m)
RSSI	277.4	411.4	907.4
Reception only	331.1	452.4	973.1
Error increase	**16.2%**	**9.1%**	**6.7%**

Table 5.1: **Performance of ANN when no RSSI is known, only which gateways received the message.**

For the Antwerp data set, there is a a 9% increase in mean error when only the reception of a message is considered, compared to when RSSI is used. The maximum error increased by almost $2000m$. In the case of Utrecht,

the mean error increases by 21%. This result using the Antwerp data set still appears better than the TDoA method proposed by Bissett [6]. The combination of gateways that received a message plays a major role in the decision of the network, but the RSSI at these receptions are crucial in improving the accuracy. By providing only the reception as input, the neural network may learn the coverage of each gateway and the prediction is made based on the intersecting coverage areas. Providing RSSI values in addition allows the network to learn a relationship between RSSI and distance, as well as patterns that may arise from the urban environment.

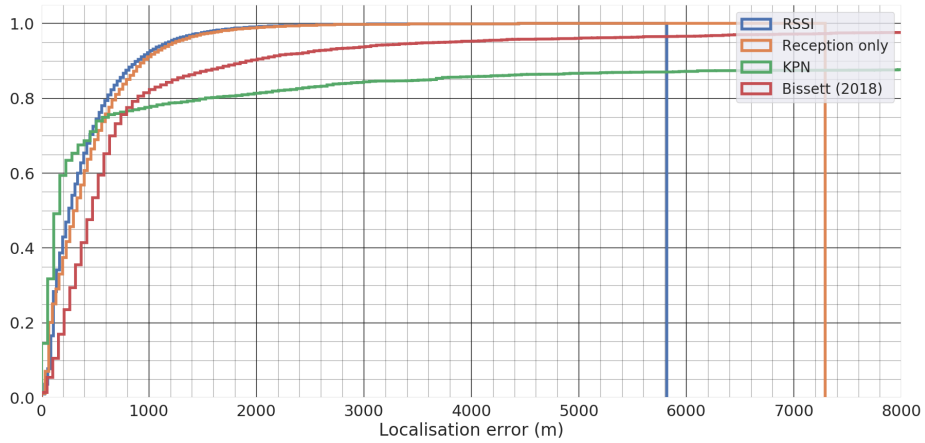


Figure 5.1: **Performance of ANN with and without RSSI information. Also shown are TDoA results from Bissett [6].**

5.2 RSSI Scaling

In many previous works with RSSI fingerprinting, researchers simply use the raw RSSI values as input to their algorithms[67]. In chapter 4, we have treated the RSSI in a similar way, simply normalising the values to be in the range between 0 and 1. A drawback of this normalisation is that the data is linear and thus does not represent the logarithmic nature of RSSI. For example, considering RSSI values between $0dBm$ and $-10dBm$. A difference of $-3dBm$ is a 50% reduction in power, but with normalised RSSI, this reduction is only 30%. In this section, we explore two different methods of scaling RSSI data that are proposed by Torres-Sospedra et al. [67].

RSSI is an indicator of signal power at the receiver reported in dBm. It is calculated from the signal strength with equation 5.1.

$$RSSI_{dBm} = 10 \times log_{10}(\frac{Power_{mW}}{1mW}) \tag{5.1}$$

Torres-Sospedra et al. concluded that researchers should consider the scaling of RSSI values as input when proposing fingerprinting techniques for localisation [67]. The scaling methods proposed by them showed improvements when used with a k-nearest neighbours classifier with WiFi data. They provided two representations named *exponential* and *powed* which are shown in equations 5.2 and 5.3.

$$Exponential_i(x) = \frac{\exp(\frac{Positive_i(x)}{\alpha})}{\exp(\frac{-min}{\alpha})} \tag{5.2}$$

$$Powed_i(x) = \frac{(Positive_i(x))^\beta}{(-min)^\beta} \tag{5.3}$$

Where min is the minimum RSSI in the data minus 1, this it to take into account gateways that did not receive a transmission, in which case their corresponding value in the data set is some out-of-range value. In both of our data sets the out-of-range value is -200. The representation $Positive_i(x)$ is used for both of these equations, which sets out-of-range values to 0, otherwise: $Positive_i(x) = RSSI_i - min$; where i is the gateway identifier. α and β are parameters to be set for specific settings.

Figure 5.2 shows a comparison between Min-Max normalisation and using the two proposed representations, for RSSI values in the range of $0dBm$ to $-140dBm$. With these new representations, small fluctuations of high signal values will result in a large changes in processed value, and small differences of weaker signals make little difference. For example, the difference between $-10dBm$ and $-20dBm$ is much larger than between $-110dBm$ and $-120dBm$, whereas in the Normalised data, this difference is the same. In this manner, the processed data can better represent the logarithmic nature of measured RSSI.

However, the authors considered these representations for WiFi which typically has higher RSSI than LoRa. They also did not consider the highest RSSI value seen in the data set, so a maximum of $0dBm$ was assumed. Since the maximum RSSI in our data sets are lower than that, using these representations as they are will effectively constrain our data to a section of the curves in figure 5.2. We made a minor modification to include the maximum RSSI by changing the way $Positive_i(x)$ is calculated. The equation to calculate this quantity, which we will refer to as $ModPositive$, is shown in equation 5.4.

$$ModPositive_i(x) = RSSI_i - min - max \tag{5.4}$$

Effectively, this makes the highest RSSI take on the absolute value of the lowest RSSI and vice versa. The effect of using $ModPositive$ can be seen in

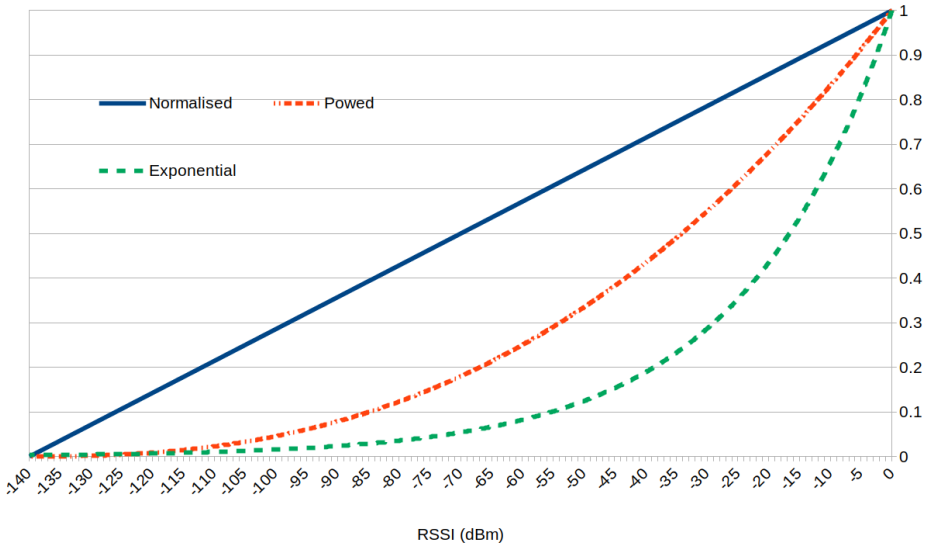


Figure 5.2: **Comparison between different RSSI representations after scaling, as proposed by Torres-Sospedra et al. [67].**

figure 5.3. By using *ModPositive*, the whole range between 0 and 1 is used, and the curves look much more similar to what was originally proposed. The next step is to select the parameters α and β. We find the values such that the scaled data set fully fits in the range 0 and 1. The parameters chosen for our data sets are shown in table 5.2.

	α	β
Antwerp	15	4
Utrecht	40	2.5

Table 5.2: **Selected values of α and β for our data sets.**

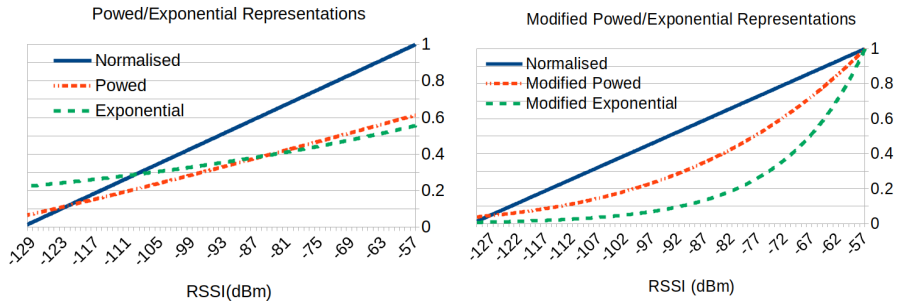


Figure 5.3: ***Exponential* and *Powed* representation with (right) and without (left) using *ModPositive*.**

Using these representations, we gained a minor improvement in performance with the Antwerp data set. The difference between using *Powed* and *Exponential* is minute. Table 5.3 presents the improvement with *Powed* compared to normalised RSSI values when used for Antwerp.

	Median error (m)	Mean error (m)	90^{th} Percentile (m)
Normalised RSSI	277.4	411.4	907.4
Powed RSSI	276.5	395.1	877.2
Improvement	**0.3%**	**3.97%**	**3.3%**

Table 5.3: **Summary of improvement when the *Powed* scheme is used with $ModPositive$ to scale the Antwerp input data.**

In the Utrecht data set however, performance deteriorates. The reason for this can be explained with figure 4.2 of the RSSI distribution in this data. We can see that a large portion of the data points have RSSI values between $-140dBm$ and $-80dBm$, another group of data points have RSSI in the range $-80dBm$ to $-40dBm$. Also, there is a small number of data points with very high RSSI. When we scale the data with these methods, most of the data are scaled to values very close to zero, because these RSSI values are much smaller than the maximum. In turn, the neural network will have difficulties in learning this data set because most of the data it sees is so close to zero.

We did not consider adapting this task to the Utrecht data set further in this work, but two ways to improve the performance may be:

- **Remove high RSSI values:** The fact that so few data points had RSSI values as high as $-5dBm$ indicates that these values are highly unlikely to appear in a realistic setting. A likely reason for these data points is because the user wished to test a gateway right after installation, standing very close to it with a LoRa node.
- **Use a different scaling method:** instead of using a scaling scheme that emphasizes on large RSSI, perhaps one that focuses more on smaller values may suit this application better. This new scheme would not take into account the logarithmic nature of RSSI, but we might expect that most LoRa messages are received with low power thanks to the long range capability.

5.3 Spreading Factor

The reception information and RSSI scaling techniques discussed so far are common to many LPWAN, or any wireless network with a similar architecture to LoRaWAN. In this section, we consider a detail that is unique to LoRa, that is, the Spreading Factor.

In LoRa, the Spreading Factor (SF) is used to provide Adaptive Data Rate (ADR). A lower SF means a shorter symbol duration which gives a higher data rate. The trade-off is that the minimum SNR required to decode the symbol is higher which lowers the range of the signal [56, 39]. Conversely, a high SF brings lower data rate but longer range.

The SF can potentially be used as an additional input for our method. As not all data points in the Utrecht set contained a Spreading Factor this step was only done for the Antwerp data set. There are certain technical aspects to consider when using the Spreading Factor. As the data set for Antwerp was collected over a proprietary network by Proximus in Belgium, the specific technical details of the ADR algorithm are not publicly available. However, we could use the ADR implementation of The Things Network to gain some insight since it is based on Semtech's recommended algorithm.

Spreading Factor	**One-hot encoding**					
SF7	1	0	0	0	0	0
SF8	0	1	0	0	0	0
SF9	0	0	1	0	0	0
SF10	0	0	0	1	0	0
SF11	0	0	0	0	1	0
SF12	0	0	0	0	0	1

Table 5.4: **One-hot encoding of Spreading Factor for use as Neural Network input.**

In this ADR scheme, the network takes measurements for the 20 most recent messages. The SNR at the gateway that had the best reception is used to calculate a margin which determines how much the data rate can be increased. Devices in a LoRaWAN typically should follow guidelines on the duty cycle, or how much time in a day the device can transmit. In The Things Network, this duty cycle is 30 seconds per day per node. With a message size of 10 bytes, this translates to about 20 messages per day at SF12 or 500 messages per day at SF7. This means that LoRa devices gathering data continuously may not be able to send a message often. If these devices are constantly travelling, the data rate may not change fast enough to adapt to the channel. Therefore, The Things Network recommends that ADR be enabled for static nodes, while mobile devices should only use ADR when they are stationary for an extended amount of time [62]. According to the authors of the Antwerp data [1], the LoRa nodes used can send a new message every minute without violating duty cycle regulations. If the same ADR algorithm as discussed before was used, the SF may change every 20 minutes. However, because the nodes are often moving, the changing environment may not allow the ADR algorithm to change the SF. Only when the postal service vehicle is stationary for a while should we expect a

change. While this is not the recommended way to use ADR, the Spreading Factor in this data set may still give us some insights.

Neural networks is a machine learning algorithm that operates on numeric data and requires all input and outputs to be numeric. Since the spreading factor can take one out of six possible values, it can be considered a categorical variable. A suitable way to represent this variable as a numerical input to the neural network is by using One-hot encoding [33]. Six more inputs were added to the neural network, each representing a spreading factor. For each data point, one of these six inputs, corresponding to the SF, will take the value of 1 and the others are 0. This encoding scheme is shown in table 5.4.

There was a small reduction in the median and mean error when SF was used which is shown in table 5.5. This result indicates that the spreading factor could be an important input to providing RSSI fingerprinting localisation with higher accuracy. As discussed, the data was gathered by mobile LoRa nodes, which are not recommended to have ADR enabled. The SF may provide an even better result if this data was taken by static nodes.

In chapter 3, we discussed how the SF can affect the range of LoRa and thus, can perhaps be used as a range indicator. This improvement may correspond to the fact that the Neural Network has learned some relationship between SF and distance. For example, with a low SF (high data rate, short range), the neural network may attach great importance to the gateway with the highest RSSI. While with higher SF values, it may add more weight to several gateways instead.

	Median (m)	Mean (m)	90^{th} Percentile (m)
Normalised-No SF	277.4	411.4	907.4
***Powed*-No SF**	276.5	395.1	877.2
***Powed*-With SF**	260.9	381.8	878.1
Improvement	**5.64%**	**3.37%**	**-0.1%**

Table 5.5: **Comparison of error between neural networks with and without Spreading Factor as inputs, using the Antwerp data set.**

5.4 Chronological Data Split

So far, the neural networks have been trained and tested with a random split of the whole data sets. In a realistic implementation of this approach, the neural network would be trained on a set of data gathered in the past, and it would be expected to estimate new data points in real-time. In this section we investigate this implementation by dividing the test and training set in time.

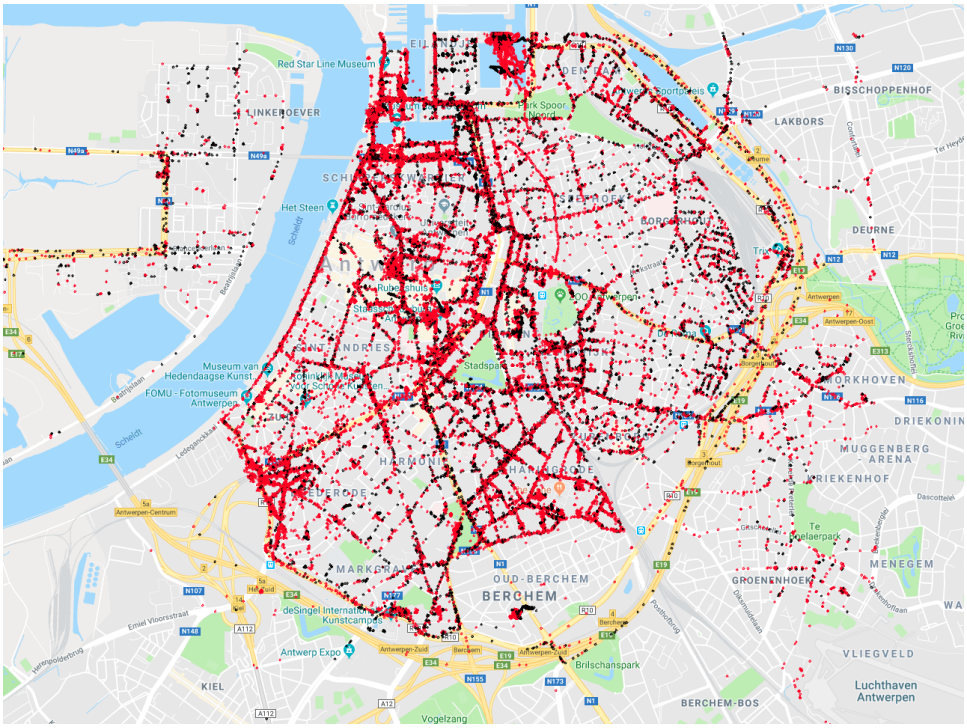


Figure 5.4: **A subset of the training and test set for Antwerp data set after splitting in time. Training set points are shown in red and test set points in black.**

The data sets are now first sorted chronologically and then split into the training and test sets with ratio 80:20 and with the training set being further in the past. The training set is then randomly split to make the validation set so that we end up with a 60:20:20 split as before. This results in a period of 12 days being used for testing in the Antwerp data set and a period of 74 days in the case of Utrecht.

Considering these data sets, we can expect the test set for Antwerp to be very similar in nature to the training set while it is not necessarily the case for Utrecht. The reasons for this expectation are:

- **Data collection method:** postal service vehicles likely follow a similar route every week, so data gathered on one week should be similar to the next in the case of Antwerp. For Utrecht data set however, we know nothing of the type of applications or method behind collecting data. Many of these data points may have been collected by users who were only interested in testing the coverage for a small area. Splitting this data in time will likely result in the test set containing data gathered for completely different purposes than those of the training set.

- **Shorter time period:** the Antwerp data set was collected in a shorter time frame of only 3 months. In this time, the landscape and environment of the city is less likely to have drastic changes compared to the 32-month data of Utrecht.

Figure 5.4 and 5.5 show these sets for Antwerp and Utrecht respectively. Indeed, while this splitting on Antwerp produces a test set that overlaps well with the training set, the same cannot be said about Utrecht. Although the training data as shown in figure 5.5 is a random subset of the whole training set, we can still see that the test data is present in some areas where there is little training data. For example, the area around the University Medical Centre, circled on the image, is where a large number of test data was gathered, but has very few training points. We expect that the neural network trained on this set will not be able to predict the test set well.

Figure 5.6 shows the result of these applications compared to when the ANN models were trained on the whole data set. As expected, the application performs significantly worse in Utrecht. In Antwerp, the accuracy also worsened, although much less than in the Utrecht case.

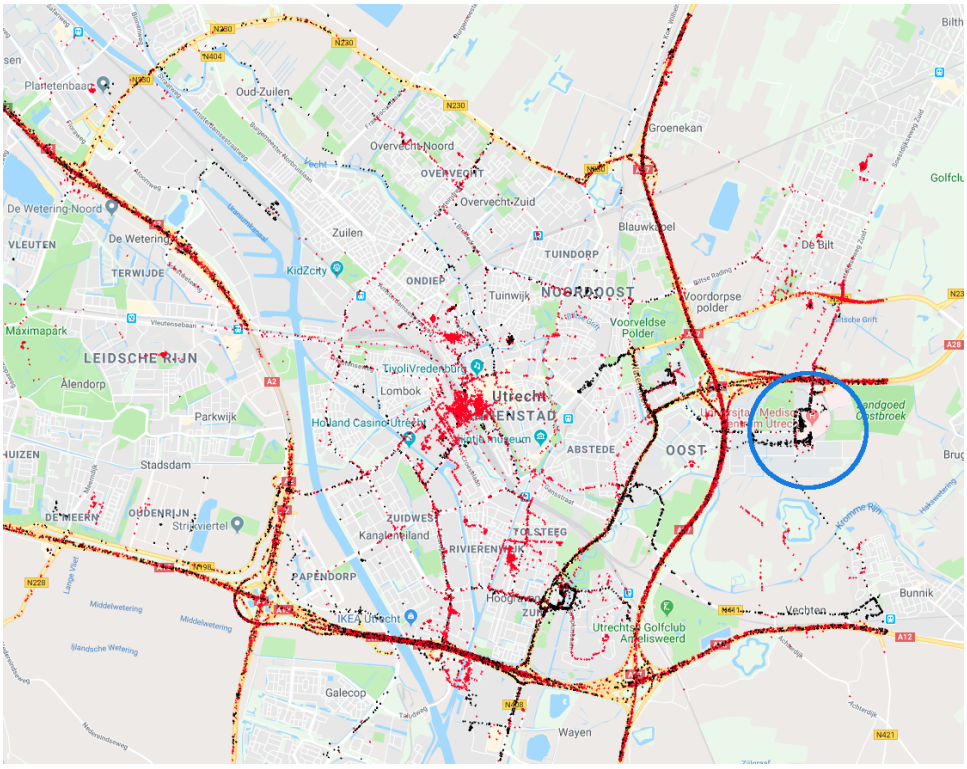


Figure 5.5: **A subset of the training and test set for Utrecht data set after splitting in time. Training set points are shown in red and test set points in black.**

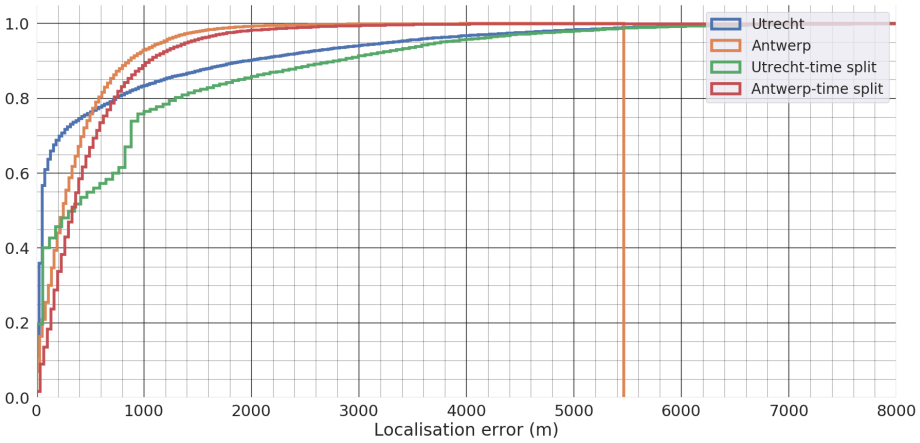


Figure 5.6: **Performance of ANN trained on chronologically split data compared with a random split of the whole data sets.**

As we discussed, the two main reasons behind the poor performance in Utrecht are the data collection method and the difference in time between the training and test set. Looking at the result from Antwerp, a proper method to collect the data, such as using postal vehicles, can go a long way in making this method better. Another benefit of using service vehicles is that the physical size of the LoRa device is less limited. A large battery can be used, or the device can even utilise the car's power to collect data for a longer period.

Providing a robust RSSI fingerprinting method for a longer time may be a challenging task. The environment and landscape of the city will inevitably change over time, with buildings, roads, tunnels, bridges, etc. being built and taken down. A data set collected over one year may not reflect the city a few years in the future. The solution may be to continuously collect new data and continuously update the model with it.

5.5 Comparison of Results

Figure 5.7 compares the result of our approach using chronologically split data, with TDoA results as reported by Bissett [6]. KPN offers the best accuracy for up to about 75% of samples. Beyond that, the Antwerp application has the best performance, with 80% of cases being located within $750m$. While the accuracy of our method suffered with this application, it provides the most realistic basis for comparing with TDoA method.

We should note that these performances were not measured with the same set of data and this is a known limitation in our approach. On the one hand, the TDoA data was gathered around the Netherlands, covering a much larger

area than either of our data sets, this data set is also not large enough to train a Neural Network. On the other hand, the data sets for our method do not have sufficiently accurate timestamps to perform TDoA. Comparing these two approaches using the same data set may be a good setting for future work.

While KPN's method gives the best accuracy for about 75% of cases, we may assume that these were data points where the gateways had line-of-sight with the node. The larger errors in TDoA will have arisen from multipath effects which is very likely to be present in data points collected in cities. Since the maximum errors of TDoA methods are very large, RSSI fingerprinting with ANN may be providing better localisation in cities overall.

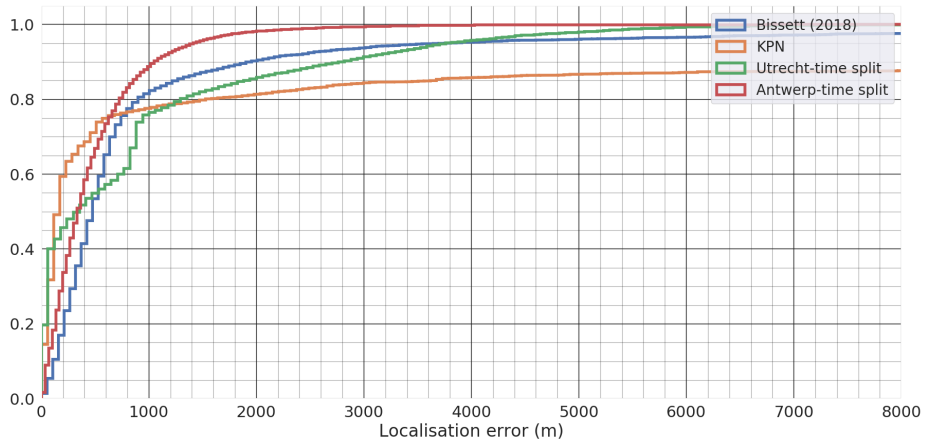


Figure 5.7: **Performance of ANN with chronologically split data compared to TDoA result reported by Bissett [6].**

In using data sets that were collected in a known, we are imposing some prior knowledge in the system, i.e. limiting the possible location of a LoRa node to within a city. We trained and tested our models using data points that we know are within this limit, thus limiting the maximum error we can encounter. On the other hand, TDoA methods can be used in a large area and can potentially have very high maximum error.

We also compare our result with that of a kNN-based approach for the Antwerp data set. Varying k in the range between 3 and 15, we select the value that gives the best mean validation error. For the Antwerp data set, $k = 11$.

Although the ANN approach on the Antwerp data set still achieves the best performance overall, the kNN model on this same data set comes quite close. The mean error using ANN is 5.3% lower than using kNN. Overall, RSSI fingerprinting methods can reduce the large errors with TDoA, but

the added complexity of using a Neural Network may not provide significant improvements over a simpler method like k-Nearest Neighbours. However, the Multilayer Perceptron used in this work is a simple ANN architecture, the size of the network and the hyperparameters selected may not be optimal for this application. A thorough study of different architectures and an extensive search of hyperparameters may bring much better performance.

	Median	**Mean**	90^{th} **Percentile**
ANN-Antwerp	341.4	**480.2**	**1048**
kNN-Antwerp	366	507	1118
TDoA-Bissett	500.4	1667.1	1961
TDoA-KPN	**174.2**	6008.4	23389.2

Table 5.6: **Performance overview of ANN fingerprinting, kNN fingerprinting, and TDoA methods in localising devices in a LoRaWAN.**

As mentioned in chapter 2, a preprint was published in the final month of this thesis [2]. The authors considered an MLP of seven layers with the following number of nodes: 1024,1024,1024,256,128,128,2. Some other parameters that they reported regarding the neural network such as activation function and optimiser are identical to this thesis. With this, they achieved a mean error of $358m$ and a median error of $204m$. The authors claimed that this result is reproducible and have made the specific data subsets public, however the authors have not published the code nor the trained model. We attempted to reproduce this result without success; there was no improvement in performance compared to our model.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis, we have presented a method, based on Artificial Neural Networks, to estimate the location of LoRa nodes in an urban environment. Existing solutions using TDoA perform well when gateways have line-of-sight with devices but their accuracies deteriorate inside cities. We evaluate the feasibility of using ANN for RSSI fingerprinting to overcome the challenges that TDoA faces in outdoor localisation.

We investigate neural network models that may be trained on the two data sets used in this thesis. For each data set, a classifier and a regressor are trained, and we find the regressors to perform consistently better. For about 80% of samples, the ANN achieved better accuracy with the Utrecht data set than with Antwerp. This is because a large portion of data in the Utrecht data set was collected in clusters. A likely reason is that the users who collected this data were mostly interested in testing the coverage of a certain gateway, in a certain area.

The thesis then studies the accuracy of the neural network when RSSI values are unknown. In this case, the mean error increases by 9% for Antwerp data, and 21% for Utrecht data. The RSSI information is therefore central in achieving better performance.

We consider two alternatives to Min-Max Normalisation for scaling RSSI data as well as using the Spreading Factor as an additional input. These changes account for a 7.2% improvement in mean error with the Antwerp data set. The best performance ANN for Antwerp achieves a median error of $260.9m$ and a mean error of $381.8m$. The Utrecht model does not benefit from the scaling method because of its data distribution and this data set does not contain a Spreading Factor for every point.

For a realistic comparison between our approach and TDoA, we split the data chronologically with the test set being nearer to the present. In our comparison, KPN's TDoA solution is the most accurate for 75% of cases but its mean error is up to $6008.4m$. Bissett's TDoA method achieves a better average of $1667.1m$ while our ANN approach with Antwerp data offers a mean error of $480.2m$. We conclude that our ANN-based approach can better limit the large errors that arise from an urban environment than TDoA methods, but TDoA is more accurate in a good radio channel.

We determine that the way the Utrecht data set was collected makes it unsuitable for RSSI fingerprinting. As this data set was gathered by many users and applications, the data at one point in time is not representative of the data at a later point. In implementing an RSSI fingerprinting algorithm such as this, the method of collecting data should therefore be carefully considered.

With the chronologically split data, we also compare the ANN performance with k-nearest neighbours fingerprinting. We find that ANN offers a 5.3% lower mean error than kNN with the Antwerp data. The ANN architecture we considered is a relatively simple one, and the hyperparameters may not be optimal for this problem. Therefore, the reported performance does not represent the best performance achievable by ANN. If simple ANNs such as the ones reported in this thesis can achieve this performance, using a different architecture as well as having an extensive search on hyperparameters may bring significant improvements.

6.2 Future Work

Localisation in a LoRaWAN is still a relatively new topic. Much more research can potentially be done to improve the accuracy of both TDoA and RSSI fingerprinting methods further. We propose some possible research direction to further explore this topic:

- **Direct comparison of ANN and TDoA:** Having a data set that allows localisation using both TDoA and RSSI fingerprinting would allow for a thorough and fair comparison between these two methods.

- **Using map details and motion model:** A lot of information can be utilised from incorporating map details into localisation methods. For example, if we know what a LoRa node is used for, we can constrain its possible location: a car is more likely to be found on the street or a parking lot than in the canal. Furthermore, if a node sends messages in quick intervals and it is moving, we can calculate its velocity and infer an estimation of its next position. A node that is moving is also more likely to be on a street.

- **Hybrid solutions:** TDoA methods work well in an open environment over long range but not in cities. RSSI fingerprinting can perform better in cities but is costly to scale to large areas. A hybrid solution using both of these can give us the best of both worlds.

Bibliography

[1] Michiel Aernouts, Rafael Berkvens, Koen van Vlaenderen, and Maarten Weyn. Sigfox and lorawan datasets for fingerprint localization in large urban and rural areas. *Data*, 3(2), 2018.

[2] Grigorios Anagnostopoulos and Alexandros Kalousis. A reproducible analysis of rssi fingerprinting for outdoor localization using sigfox: Preprocessing and hyperparameter tuning, 08 2019.

[3] 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)*, volume 2, pages 775–784 vol.2, March 2000.

[4] Priyanka Bhat. Analysis of remote sensing approaches for lora coverage estimation. Master thesis, Delft University of Technology, Delft, The Netherlands, 2019.

[5] D. A. Bibb, Z. Yun, and M. F. Iskander. Machine learning for source localization in urban environments. In *MILCOM 2016 - 2016 IEEE Military Communications Conference*, pages 401–405, Nov 2016.

[6] David Bissett. Analysing tdoa localisation in lora networks. Master thesis, Delft University of Technology, Delft, The Netherlands, 2018.

[7] J. Biswas and M. Veloso. Wifi localization and navigation for autonomous indoor mobile robots. In *2010 IEEE International Conference on Robotics and Automation*, pages 4379–4384, May 2010.

[8] Sinem Bozkurt, Gulin Elibol, Serkan Gunal, and Ugur Yayan. A comparative study on machine learning algorithms for indoor positioning. pages 1–8, 09 2015.

[9] Jason Brownlee. A gentle introduction to the rectified linear unit (relu). `https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/`. Last accessed: Sep. 21, 2019.

[10] Jason Brownlee. When to use mlp, cnn, and rnn neural networks. https://machinelearningmastery.com/when-to-use-mlp-cnn-and-rnn-neural-networks/. Last accessed: Sep. 17, 2019.

[11] Luca Calderoni, Matteo Ferrara, Annalisa Franco, and Dario Maio. Indoor localization in a hospital environment using random forest classifiers. *Expert Syst. Appl.*, 42(1):125–134, January 2015.

[12] Zhikui Chen, Feng Xia, Tao Huang, Fanyu Bu, and Haozhe Wang. A localization method for the internet of things. *The Journal of Supercomputing*, 63(3):657–674, Mar 2013.

[13] Wongeun Choi, Yoon-Seop Chang, Yeonuk Jung, and Junkeun Song. Low-power lora signal-based outdoor positioning using fingerprint algorithm. *ISPRS International Journal of Geo-Information*, 7(11), 2018.

[14] European Space Agency. Galileo performances. https://gssc.esa.int/navipedia/index.php/Galileo_Performances. Last accessed: Sep. 11, 2019.

[15] B. C. Fargas and M. N. Petersen. Gps-free geolocation using lora in low-power wans. In *2017 Global Internet of Things Summit (GIoTS)*, pages 1–6, June 2017.

[16] Silke Feldmann, Kyandoghere Kyamakya, Ana Zapater, and Zighuo Lue. An indoor bluetooth-based positioning system: Concept, implementation and experimental evaluation. In *International Conference on Wireless Networks*, 2003.

[17] Pedro Figueiredo e Silva, Ville Kaseva, and Elena Simona Lohan. Wireless positioning in iot: A look at current and future trends. *Sensors*, 18(8), 2018.

[18] geopy. Welcome to geopys documentation! https://geopy.readthedocs.io/en/stable/#. Last accessed: Sep. 15, 2019.

[19] Gerard J.M. Janssen. Wireless communication - lecture 2. Last accessed: Sep. 11, 2019.

[20] Andrea Goldsmith. *Wireless Communications*. Cambridge University Press, 2005.

[21] Google. Geocoding api - get started. https://developers.google.com/maps/documentation/geocoding/start. Last accessed: Sep. 15, 2019.

[22] P. Gotthard and T. Jankech. Low-cost car park localization using rssi in supervised lora mesh networks. In *2018 15th Workshop on Positioning, Navigation and Communications (WPNC)*, pages 1–6, Oct 2018.

[23] Aurlien Gron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 1st edition, 2017.

[24] F. Gustafsson and F. Gunnarsson. Positioning using time-difference of arrival measurements. In *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03).*, volume 6, pages VI–553, April 2003.

[25] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. The elements of statistical learning. Springer Series in Statistics, New York, NY, USA, 2001. Springer New York Inc.

[26] J.M.; Ocaa-M.; Kim E. Hernndez, N.; Alonso. Wifi-based indoor localization using a continuous space estimator from topological information. *Indoor Positioning and Indoor Navigation*, 10 2015.

[27] Noelia Hernndez, Manuel Ocaa, Jose M. Alonso, and Euntai Kim. Continuous space estimation: Increasing wifi-based indoor localization resolution without increasing the site-survey effort. *Sensors*, 17(1), 2017.

[28] Geoffrey Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint*, arXiv, 07 2012.

[29] Xuke Hu, Jianga Shang, Fuqiang Gu, and Qi Han. Improving wifi indoor positioning via ap sets similarity and semi-supervised affinity propagation clustering. *International Journal of Distributed Sensor Networks*, 11(1):109642, 2015.

[30] IoT Analytics. Lpwan emerging as fastest growing iot communication technology 1.1 billion iot connections expected by 2023, lora and nb-iot the current market leaders. `https://iot-analytics.com/lpwan-market-report-2018-2023-new-report/`. Last accessed: Sep. 10, 2019.

[31] T. Janssen, M. Aernouts, R. Berkvens, and M. Weyn. Outdoor fingerprinting localization using sigfox. In *2018 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–6, Sep. 2018.

[32] Jason Brownlee. Use weight regularization to reduce overfitting of deep learning models. `https://machinelearningmastery.com/`

weight-regularization-to-reduce-overfitting-of-deep-learning-models/. Last accessed: Sep. 17, 2019.

[33] Jason Brownlee. Machine learning mastery. https://machinelearningmastery.com/, 2017. Last accessed: Sep. 08, 2019.

[34] Zhiping Jiang, Wei Dong Xi, Xiuping Li, Jizhong Zhao, and Jisong Han. Hiloc : A tdoa-fingerprint hybrid indoor localization system. 2014.

[35] Charles F. F. Karney. Algorithms for geodesics. *Journal of Geodesy*, 87(1):43–55, Jan 2013.

[36] K. Kavitha Muthukrishnan, G.T. Koprinkov, Nirvana Meratnia, and M.E.M. Lijding. *Using time-of-flight for WLAN localization: feasibility study.* Number 06-28 in CTIT Technical Report Series. Centrum voor Telematica en Informatie Technologie, 6 2006.

[37] Keras. Keras: The python deep learning library. https://keras.io/. Last accessed: Sep. 15, 2019.

[38] Fekher Khelifi, A. Bradai, Abderrahim Benslimane, Priyanka Rawat, and Mohamed Atri. A survey of localization systems in internet of things. *Mobile Networks and Applications*, 08 2018.

[39] E.-K.; Kim J. Kim, D.-H.; Lee. Experiencing lora network establishment on a smart energy campus testbed. *Sustainability*, 2019.

[40] J. Kim, S.; Ko. Low-complexity outdoor localization for long-range, low-power radios. In *In Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services Companion*, page 44, June 2016.

[41] KPN. Lora — geolocation. https://zakelijkforum.kpn.com/lora-forum-16/lora-geolocation-8555. Last accessed: Sep. 12, 2019.

[42] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[43] P. Kumar, L. Reddy, and S. Varma. Distance measurement and error estimation scheme for rssi based localization in wireless sensor networks. In *2009 Fifth International Conference on Wireless Communication and Sensor Networks (WCSN)*, pages 1–4, 2009.

[44] In Lee and Kyoochun Lee. The internet of things (iot): Applications, investments, and challenges for enterprises. *Business Horizons*, 2015.

[45] Honggui Li. Low-cost 3d bluetooth indoor positioning with least square. *Wireless Personal Communications*, 78:1331–1344, 09 2014.

[46] LoRa Alliance. Geolocation whitepaper. `https://lora-alliance.org/sites/default/files/2018-04/geolocation_whitepaper.pdf`. Last accessed: Sep. 12, 2019.

[47] LoRa Alliance. Landing page. `https://lora-alliance.org/`. Last accessed: Sep. 11, 2019.

[48] University of Cambridge. Dataset splitting. `https://www.cl.cam.ac.uk/teaching/1617/MLRD/handbook/dataset-splits.pdf`. Last accessed: Sep. 17, 2019.

[49] Ministry of Defence (Navy). *Admiralty Manual of Navigation, Volume 1*. The Stationery Office, 1987.

[50] Douwe Osinga. *Deep Learning Cookbook-Practical Recipes to Get Started Quickly*. O'Reilly Media, Inc., 1st edition, 2018.

[51] Nico Podevijn, David Plets, Jens Trogh, Luc Martens, Pieter Suanet, Kim Hendrikse, and Wout Joseph. Tdoa-based outdoor positioning with tracking algorithm in a public lora network. *Wireless Communications and Mobile Computing*, 2018:1–9, 05 2018.

[52] Theodore Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition, 2001.

[53] Sakshama Ghoslya. All about lora and lorawan. `https://www.sghoslya.com/`. Last accessed: Sep. 08, 2019.

[54] A. H. Salamah, M. Tamazin, M. A. Sharkas, and M. Khedr. An enhanced wifi indoor localization system based on machine learning. In *2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–8, Oct 2016.

[55] Semtech. Low energy consumption design.

[56] Semtech. Lora modem design guide. `https://www.semtech.com/uploads/documents/LoraDesignGuide_STD.pdf`, 2013. Last accessed: Sep. 08, 2019.

[57] Rashmi Sharan Sinha, Yiqiao Wei, and Seung-Hoon Hwang. A survey on lpwa technology: Lora and nb-iot. *ICT Express*, 3(1):14 – 21, 2017.

[58] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[59] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, January 2014.

[60] TensorFlow. Landing page. `https://www.tensorflow.org/`. Last accessed: Sep. 15, 2019.

[61] The Things Network. Landing page. `https://www.thethingsnetwork.org/`. Last accessed: Sep. 15, 2019.

[62] The Things Network. Lorawan adaptive data rate. `https://www.thethingsnetwork.org/docs/lorawan/adr.html`. Last accessed: Sep. 09, 2019.

[63] The Things Network. Lorawan overview. `https://www.thethingsnetwork.org/docs/lorawan/`. Last accessed: Sep. 11, 2019.

[64] The Things Network. Ttn mapper. `https://www.thethingsnetwork.org/docs/applications/ttnmapper/`. Last accessed: Sep. 15, 2019.

[65] The Things Network. Lorawan distance world record broken, twice. 766 km (476miles) using 25mw transmission power. `https://www.thethingsnetwork.org/article/lorawan-distance-world-record`, 2019. Last accessed: Sep. 11, 2019.

[66] Tony Pallone. Powering iot's next generation. `https://electronics360.globalspec.com/article/10806/powering-iot-s-next-generation`, 2018. Last accessed: Sep. 11, 2019.

[67] Joaqun Torres-Sospedra, Ral Montoliu, Sergio Trilles, scar Belmonte, and Joaqun Huerta. Comprehensive analysis of distance and similarity measures for wi-fi fingerprinting indoor positioning systems. *Expert Systems with Applications*, 42(23):9263 – 9278, 2015.

[68] Zeynep Turgut, Serpil Üstebay, Gülsüm Zeynep Gürkaş Aydın, and Ahmet Sertbaş. Deep learning in indoor localization using wifi. In Ali Boyaci, Ali Riza Ekti, Muhammed Ali Aydin, and Serhan Yarkan, editors, *International Telecommunications Conference*, pages 101–110, Singapore, 2019. Springer Singapore.

[69] Feng Yu, Ming Jiang, Jing Liang, Xiao Qin, Ming Hu, Tao Peng, and Xin Hu. An indoor localization of wifi based on support vector machines. *Advanced Materials Research*, 926-930:2438–2441, 05 2014.

[70] Feng Yu, Minghua Jiang, Jing Liang, Xiao Qin, Ming Hu, Tao Peng, and Xinrong Hu. 5 g wifi signal-based indoor localization system using cluster -nearest neighbor algorithm. *International Journal of Distributed Sensor Networks*, 2014, 12 2014.

[71] F. Zafari, A. Gkelias, and K. K. Leung. A survey of indoor localization systems and technologies. *IEEE Communications Surveys Tutorials*, 21(3):2568–2599, thirdquarter 2019.