

INSTITUT POLYTECHNIQUE DE PARIS





IP PARIS



NNT : 2020IPPAS012

Thèse de doctorat

Machine Learning based Localization in 5G

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à Télécom SudParis

École doctorale n°626 Institut Polytechnique de Paris (ED IP Paris)
Spécialité de doctorat : Information et Communication

Thèse présentée et soutenue à Evry, le le 6 Novembre 2020, par

ABDALLAH SOBEHY

Composition du Jury :

Nadjib AIT SAADI Professeur, Université de Versailles Saint-Quentin-en-Yvelines	Président
Sidi-Mohammed SENOUCI Professeur, Université de Bourgogne, France	Rapporteur
Hamida SEBA Maître de conférences, Université Claude Bernard Lyon 1, France	Rapporteur
Nadjib AIT SAADI Professeur, Université de Versailles Saint-Quentin-en-Yvelines, France	Examinateur
Stephane MAAG Professeur, Télécom SudParis, France	Examinateur
Oyunchimeg SHAGDAR Research Team Leader, VEDECOM, France	Examinateur
Éric Renault Professeur, ESIEE Paris (LIGM), France	Directeur de thèse
Paul MUHLETHALER Research Team Leader, Inria (EVA), France	Co-directeur de thèse

Titre : Localisation basée sur l'apprentissage artificielle en 5G

Mots clés : Localisation, Machine Learning, 5G

Résumé : La localisation est le processus d'estimation de la position d'une entité dans un système de coordonnées. Les applications de localisation sont largement réparties dans des contextes différents. Dans les événements, le suivi des participants peut sauver des vies pendant des crises. Dans les entrepôts, les robots transférant des produits d'un endroit à un autre nécessitent une connaissance précise de ses positions, la position des produits ainsi que des autres robots. Dans un contexte industriel, la localisation est essentielle pour réaliser des processus automatisés qui sont assez flexibles pour être reconfiguré à diverses missions. La localisation est considérée comme un sujet de grand intérêt tant dans l'industrie que dans l'académie, en particulier avec l'avènement de la 5G avec son "Enhanced Mobile Broadband (eMBB)" qui devrait atteindre 10 Gbits/s, "Ultra-Reliable Low-Latency Communication (URLLC)" qui est moins d'une milliseconde et "massive Machine-Type Communication (mMTC)" permettant de connecter $\approx$ 1 million d'appareils par kilomètre. Dans ce travail, nous nous concentrons sur deux principaux types de localisation; la localisation basée sur la distance entre des appareils et la localisation basée sur les empreintes digitales. Dans la localisation basée sur la distance, un réseau d'appareils avec une distance de communication maximale estime les valeurs de distances par rapport à leurs voisins. Ces distances ainsi que la connaissance des positions de quelques nœuds sont utilisées pour localiser d'autres nœuds du réseau à l'aide d'une solution basée sur la triangulation. La méthode proposée est capable de localiser $\approx$ 90% des nœuds d'un réseau avec un degré moyen de 10. Dans la localisation basée sur les empreintes digitales, les réponses des chaînes sans fil sont utilisées pour estimer la position d'un émetteur communiquant avec une antenne MIMO. Dans ce travail, nous appliquons des techniques d'apprentissage classiques (K-nearest neighbors) et des techniques d'apprentissage en profondeur (Multi-Layer Perceptron Neural Network et Convolutional Neural Networks) pour localiser un émetteur dans des contextes intérieurs et extérieurs. Notre travail a obtenu le premier prix au concours de positionnement préparé par IEEE Communication Theory Workshop parmi 8 équipes d'universités de grande réputation du monde entier en obtenant une erreur carrée moyenne de 2,3 cm.

Title : Machine Learning based localization in 5G

Keywords : Localization, Machine Learning, 5G

Abstract : Localization is the process of determining the position of an entity in a local or global coordinate system. The applications of localization are widely spread across different contexts. For instance, in events, tracking the participants can save lives during crises. In warehouses, robots transferring products from one place to another require accurate knowledge of products' positions as well as other robots. In industrial context of the factory of the future, localization is invaluable to achieve automated processes that are flexible enough to be reconfigured for various purposes. Localization is considered a topic of high interest both in the academia and industry especially with the advent of 5G. The requirements of 5G pave the way for revolutionizing localization capabilities; Enhanced Mobile Broadband (eMBB) that is expected to reach 10 Gbits/s, Ultra-Reliable Low-Latency Communication (URLLC) which is less than 1 ms and massive Machine-Type Communication (mMTC) allowing to connect around 1 million devices per km. In this work, we focus on two main types of localization; range-based localization and fingerprinting based localization. In range-based localization, a network of devices with a maximum communication range estimate inter-distance values to their first-hop neighbors. These distances along with knowledge of positions of few anchor nodes are used to localize other nodes in the network using a triangulation based solution. The proposed method is capable of localizing $\approx$ 90% of nodes in a network with an average degree of $\approx$ 10. In the second contribution, wireless channel responses, aka. Channel State Information (CSI) is used to estimate the position of a transmitter communicating with a MIMO antenna. In this work, we apply classical learning techniques (K-nearest neighbors) and deep learning techniques (Multi-Layer Perceptron Neural Network and Convolutional Neural Networks) to localize a transmitter in indoor and outdoor contexts. Our work achieved the first place in the indoor positioning competition prepared by IEEE's Communication Theory Workshop among 8 teams from highly reputable universities worldwide by achieving a Mean Square Error of 2.3 cm.

Institut Polytechnique de Paris
91120 Palaiseau, France

Disclaimer

I hereby declare that this thesis is my own original work and that no part of it has been submitted for a degree qualification at any other university or institute.

I further declare that I have appropriately acknowledged all sources used and have cited them in the references section.

Name: Abdallah SOBEHY Date: 01/12/2020

Signature: Abdallah Sobehy

Dedication

All thanks to Allah for allowing me to reach this point, I have been blessed with his generosity since day one. I was born within the hands of two loving and caring parents whom I owe the most for the person I have become. They never hesitated to invest in educating and raising me with all their best. They also gave me two beautiful sisters with whom I played a lot and fought a lot but they will always be my little girls who I wish to see in the happiest of places. No matter how much life distanced us, their places in my heart are unchanged.

I have been blessed with great friends from school where life separations did not change their friendship and support. We still play "Among us" at midnight and laugh from the bottom of our hearts. In the University, I was blessed to know many great friends whom I truly cherish. Mohamed Badr and Mohamed Atoua have been by my side for the longest and we lived together the good and the bad and I ask Allah to preserve our friendship for good and even in our children afterwards. Zahraa Badr proof read all my papers and thesis and was never late to my request. Two people became much closer when I travelled to Europe, Oumar Yehia stood by my side and supported me in the most difficult moments and Mostafa Fateen has been by my side during my MSc and I am lucky to have known him and his family.

In 2014, I started my trip in Europe and I was again blessed with friends and family. The hardest thing that one faces when travelling in search of education or career is being distanced from family. I have been honored to meet Ahmed Haitham, Amine Essnabi, Adrien Costa who always considered me as family, real family and I owe them a lot of moments of happiness and support. Europe would not have felt as home as it feels without their support. Even if we do not meet a lot, knowing that they are there makes everything much easier.

The start of the PhD marked a life-changing event as I was blessed to meet my soul mate and her lovely family who became my family. She lived and saw like no one else the good and the difficult moments of the PhD. And while the difficult moments lasted much longer than the good ones, she never doubted me and comforted me all along the way like no one could ever do. And on the 27th of July 2019, I was reborn by marrying her and finding true home and warmth by her side. And even during

the current mixed emotional moment with the end of my PhD and hardships with the economic, health crisis, I never saw her belief in me changes. I knew love by knowing her so thank you for everything my love Enayat.

This PhD, or in other words my little girl that has been a very rich experience where I grew personally much more than technically. To the hundreds of lines of notes and codes who fell in the battle and did not make it to the thesis content, I want you to know that if it was not for you, the final result would not have been possible.

Acknowledgements

I was lucky to be supervised by Robert Eidenberger during my MSc thesis in Siemens and during the PhD I was lucky to enroll in the uncertainty propagation course with Pascal Pernot as he is a great, very generous teacher. Paul Gibson was my referent and I appreciated his follow up and guidance. So, many thanks to all of you.

Arriving at the PhD, I was lucky again to be supervised by Eric Renault and Paul Muhlethaler. Paul has always welcomed us in his office for all our meetings in Inria and supported my activities especially that of the indoor competition which was a turning point in my PhD. Many thanks and gratitude to you Paul.

Eric gave me the first class in my masters and right away when we met, I realised there is chemistry between us as he is easy-going, highly skilled and very understanding person. I have been his Msc student, teacher assistant, PhD student and yet he is much more than this. I remember, In the most frustrating moments of my PhD, not once did he undermine my work or lost faith in me. And While other PhD students wished to meet their supervisors less, I have always wanted to meet and discuss with him more. Thank you Eric for everything.

Table of Contents

Disclaimer iii

Dedication iv

Acknowledgements vi

Table of Contents vii

List of Tables x

List of Figures xi

List of Publications xiv

Abstract xv

1 INTRODUCTION xvi

1.1 Localization 2

1.2 Applications 2

1.3 Localization Challenges 4

1.4 Thesis outline 6

2 LITERATURE REVIEW 7

2.1 Range-based Localization 7

2.2 CSI Localization 9

2.3 Deep Learning Enhancements 12

2.3.1 Convolutional Neural Networks 13

2.3.2 Graph Neural Networks 15

3 POSITION CERTAINTY PROPAGATION IN MANETS 17

3.1 Problem Formulation 17

3.2 Position Certainty Propagation 18

3.2.1 Three nodes with known position in their vicinity 18

3.2.2 Two nodes with known positions in vicinity 22

3.3 Algorithmic Implementation 23
3.3.1 Semi-Flooding based Position Sharing 24
3.3.2 Self Localization 26
3.4 Performance Evaluation 28
3.4.1 Comparison with GPS-free method 28
3.4.1.1 Experimental Setup 28
3.4.2 Experimental Results 29
3.5 Behavior in simulated network 29
3.5.1 Distance Noise Effect 30
3.5.2 Error propagation towards the network's edge 33
3.6 Conclusion 34

4 CSI-BASED INDOOR LOCALIZATION 36
4.1 Channel State Information 37
4.2 Experimental Setup 39
4.3 Methodology 40
4.3.1 CSI Components Analysis 40
4.3.2 NDR: Noise and Dimensionality Reduction 41
4.3.3 Learning Models 43
4.3.3.1 Multi-Layer Perceptron Neural Networks 44
4.3.3.2 Data Augmentation and Ensemble NNs 47
4.3.3.3 K-Nearest Neighbors 50
4.4 Conclusion 54

5 GENERALIZATION OF DEEP LEARNING LOCALIZATION 57
5.1 Motivation 57
5.2 Generalization: MLP NN vs KNN 58
5.2.1 Square Test set selection 58
5.2.2 Sequential Test set selection 61
5.2.3 Conclusion 62
5.3 Outdoor Localization Problem 64
5.3.1 Experimental Setup 65
5.3.2 Data Preprocessing 66

5.3.2.1 Fourier based Noise and Dimensionality reduction . 66
5.3.2.2 Missing Readings replacement 69
5.3.3 Learning from Labelled dataset 70
5.3.4 Generalizing to unlabelled dataset 72
5.3.4.1 Distance estimation between CSI pairs 72
5.3.4.2 CNNs: Generalization Evaluation 76
5.3.4.3 From Distance estimation to position estimation . . 77
5.4 Conclusion 78

6 CONCLUSION AND FUTURE WORK 79
6.1 Conclusion 79
6.2 Future Work 81

References 83

Appendix A UNCERTAINTY PROPAGATION 95
A.1 Uncertainty Definition 95
A.2 Sources of Errors 97
A.3 Uncertainty Propagation 99
A.3.1 Combination of Variances 102
A.3.2 Monte Carlo 102
A.3.3 Bayesian Inference 104

Appendix B MACHINE LEARNING 105
B.1 Regression and Classification 105
B.1.1 Linear Regression 106
B.1.2 Logistic Regression 108
B.1.3 Remarks 111
B.2 Multi Layer Perceptron Neural Networks 112
B.2.1 Motivation 112
B.2.2 Structure and Feed Forward 114
B.2.3 Learning and Backward Propagation 115
B.2.4 Remarks 118

List of Tables

2.1 Localization approaches. 9

4.1 Hyperparameters selection. 44

4.2 Hyperparameters selection. 49

5.1 Indoor vs Outdoor characteristics 65

5.2 Performance of different preprocessing steps 72

5.3 CNN Hyperparameters 74

List of Figures

1.1 Autonomous Driving example [5] 3
1.2 Routing example [6] 4
1.3 Amazon's warehouse order picking robots [6] 5

2.1 MNIST dataset written digit example [59] 13
2.2 MLP NN unit input [61] 14
2.3 CNN unit input [61] 15

3.1 15-node network example 18
3.2 Counting number of known position neighbours 19
3.3 Computing position from the intersection of three circles 20
3.4 Computing position based on two nodes with known positions . . . 23
3.5 The final result of the algorithm 24
3.6 Varying Max Communication Range 30
3.7 Varying Node Density 30
3.8 Grid network 31
3.9 Stability of localization error over time 32
3.10 Average Localization error with different standard deviation values . 32
3.11 16-node Network Example 33
3.12 Error Propagation 34
3.13 Trace of number of broadcasts per time step 35

4.1 Real, Imaginary, Magnitude and Phase components example over 924 subcarriers. 38
4.2 Experimental setup 39
4.3 Real, Imaginary, Magnitude, and phase components estimated from 4 transmissions at the same position. 40
4.4 Average Correlation for Real, Imaginary, Magnitude, and Phase components. 41
4.5 Average fitting error for various polynomial degrees. 42
4.6 Underfitting example and dividing subcarrier space. 42

4.7 Discontinuity Removal 43
4.8 Test set Error distribution. 45
4.9 Log of errors plots. 45
4.10 Log of MSE showing the 95th error bars. 46
4.11 Error Cumulative distribution. 47
4.12 Data augmentation sample from an original training sample. 48
4.13 Effect of data augmentation on localization accuracy. 49
4.14 Mixing the predictions of a neural network ensemble. 51
4.15 Mean Square Error using different closeness criteria. 53
4.16 Mean Square Error using different k values. 53
4.17 Relating the Euclidean distance between the test and closest training sample to the prediction error. 54
4.18 Comparison between proposed solution and state-of-the-art method. 56

5.1 Square Test set selection 59
5.2 KNN predicted positions vs actual positions for square test set . . . 59
5.3 KNN error distribution for square test 60
5.4 MLP NN predicted positions vs actual positions for square test set . 60
5.5 MLP NN error distribution for Square test set 61
5.6 Sequential Test set selection 62
5.7 KNN predicted positions vs actual positions for sequential test set . 62
5.8 KNN error distribution for sequential test 63
5.9 MLP NN predicted positions vs actual positions for sequential test set 63
5.10 MLP NN error distribution for sequential test 64
5.11 Traversed roads in a residential area in Stuttgart [2] 66
5.12 CSI components stability per position 67
5.13 Magnitude CSI readings for 5 transmissions per position 67
5.14 Fourier Series approximation of 5 transmissions with $N = 5$ and $N = 15$. 68
5.15 SNR color map 69
5.16 Log scaled SNR differences between position pairs within 3m range 70
5.17 SNR differences between position pairs within 3m range for each replacement method. 71

5.18 mean per sample replacement method: color map (left figure) and SNR differences between positions within 3m (right figure) 71
5.19 CNN error distribution for distance estimation 75
5.20 CNN distance estimation errors per maximum predicted distance thresholds 75
5.21 Map train and test sets which are separated for generality evaluation. 76
5.22 CNN distance estimation errors per maximum predicted distance thresholds for the generality evaluation 77

List of Publications

1. A. Sobehy, É. Renault and P. Mühlethaler. “Position certainty propagation: a location service for MANETs”. In Proc. 4th International Conference on Mobile, Secure and Programmable Networking (MSPN), 2018, Springer LNCS 11005, pp.131-142.

2. A. Sobehy, É. Renault and P. Mühlethaler. "Position certainty propagation: a localization service for Ad-Hoc Networks". MDPI Computers Journal, vol 8, no.1, 2019.

3. A. Sobehy, É. Renault and P. Mühlethaler. “NDR: Noise and Dimensionality Reduction of CSI for indoor positioning using deep learning”. In Proc. IEEE International Conference on Global Communications (IEEE Globecom), 2019.

4. A. Sobehy, É. Renault and P. Mühlethaler. “CSI based indoor localization using Ensemble Neural Networks”. In Proc. 2nd IFIP International Conference on Machine Learning for Networking (IFIP MLN), 2019, Springer LNCS 12081, pp.367-378.

5. A. Sobehy, É. Renault and P. Mühlethaler. “CSI-MIMO: K-nearest neighbor applied to indoor localization”. In Proc. IEEE International Conference on Communications (IEEE ICC), 2020.

6. A. Sobehy, É. Renault and P. Mühlethaler. "Generalization aspect of accurate Machine Learning models for CSI based localization". Annals of Telecommunications Journal, 2020 (submitted).

Abstract

Localization is the process of determining the position of an entity in a local or global coordinate system. The applications of localization are widespread across different contexts. For instance, in events, tracking the participants can save lives during crises. In health-care, elderly people can be tracked to respond to their needs in critical situations. In warehouses, robots transferring products from one place to another require accurate knowledge of products' positions as well as humans' and other robots' to achieve their tasks. In the industrial context of the factory of the future , localization is invaluable to achieve automated processes that are flexible enough to be reconfigured for various purposes. Localization is considered a topic of high interest both in academia and industry especially with the advent of 5G. Localization accuracy can be enhanced by exploiting the unprecedented 5G's requirements: Enhanced Mobile Broadband (eMBB) that is expected to reach 10 Gbits/s, Ultra-Reliable Low-Latency Communication (URLLC) which is less than 1 ms, and massive Machine-Type Communication (mMTC) allowing to connect around 1 million devices per km. We exploit machine learning frameworks to model the relation between communication-related measurements such as Channel State Information (CSI) and Received Signal Strength Indicator (RSSI) to achieve accurate localization. In this work, we focus on two main types of localization: range-based localization and fingerprinting based localization. In range-based localization, we aim to localize nodes in MANETs with known maximum communication range and distances between first-hop neighbors. These distances along with the knowledge of the positions of few anchor nodes are used to localize other nodes in the network using a triangulation based solution. The proposed method is capable of localizing $\approx 90\%$ of nodes in a network with an average degree of ≈ 10. In the second contribution, CSI is used to estimate the position of a transmitter communicating with a MIMO antenna. This is achieved by applying classical learning techniques (K-nearest neighbors) and deep learning techniques (Multi-Layer Perceptron Neural Network and Convolutional Neural Networks) to localize a transmitter in indoor and outdoor contexts. Our work achieved the first place in the indoor positioning competition [1] prepared by IEEE's Communication Theory Workshop among 8 teams from highly reputable universities worldwide by achieving a Mean Square Error of 2.3 cm. Finally, the generalization capability is evaluated for machine learning models: MLP NNs and KNNs. We propose enhancements to the proposed machine learning models to achieve better localization. These enhancements are applied to an outdoor localization problem [2] achieving a 10 cm distance error estimation between locations up to 3 m apart.

Chapter 1

Introduction

The recent resurrection of AI and machine learning ignited innovations in multiple areas including computer vision, communications, mobility, industrial processes, and others. This technological boom was made possible thanks to advances in computers and algorithms. Meanwhile, the fifth generation (5G) of cellular networks technology was on the rise. 5G promises unprecedented capabilities in cellular communications especially on the data rate and reliability fronts. The combined capabilities of AI and 5G pave the way for breakthroughs in plenty of applications. 5G and AI have a strong synergy between them as they play integral roles in improving each other. For instance, heavy duty AI tasks are not suitable to be processed on mobile devices due to computational and energy constraints. That is when 5G comes in to provide a reliable and fast connection to allow AI tasks to be processed on the cloud or the edge cloud in real time. On the other hand, AI improves the network tasks such as beam management and routing decisions by predicting the location and velocity of users. Machine learning achieves this by fitting a highly complex model to collected data and extracting patterns that allow it to forecast future events. Predicting user demands in different areas helps in deciding the locations of new cells in the environment and thus enhancing the user experience.

In this work, we attempt to exploit the combined strengths of 5G and AI to solve the localization problem. In Mobile Adhoc Networks (MANETs), for instance, devices communicate directly to achieve a certain task. These networks are highly dynamic in terms of nodes' mobility and nodes entering or exiting the network. Location awareness helps in enhancing the performance for multiple applications such as routing. Communication signals offer accessible and reliable measurements to achieve localization such as Channel State Information (CSI) and Received Signal Strength Indicator (RSSI). Machine learning techniques can be used to extract patterns in order to model the relation between such measurements and the location of the device of interest. High localization accuracy often leads to high performance in multiple contexts as illustrated in this work.

1.1 Localization

Localization refers to the determination of the position of an entity with respect to a reference coordinate system. The entity to be localized can be a communication device, a robot, a human, or any object of interest. The localization accuracy can vary from pin pointing the position with an error of few millimeters to rough position estimation with high error of hundreds of meters, or estimating a region where the entity exists such as a room or a floor in a building. The localization accuracy depends heavily on the available measurements and the application that uses the location information. Some measuring technologies enable direct location estimation such as the Global Positioning System (GPS). Other measurements are used to estimate the location of the entity indirectly, e.g., cameras, Received Signal strength Indicator, LiDAR sensors, etc. Location services play an important role in many applications in industrial, civilian, and military contexts.

1.2 Applications

One of the earliest and most widespread uses of location based services is the Global Positioning System (GPS) which was devised by the U.S. Department of Defense in the 1970s for military purposes [3]. The GPS relies on a satellite infrastructure with the aim of localizing devices. In the 1980s, it was made available for civilian and industrial consumption. This decision opened doors for innovations that shapes the world we live in. For instance, one of the most successful commercial exploitations of GPS that revolutionized how people commute is achieved by Uber, the ride sharing application [4]. By knowing the location of users who are interested in commuting from one place to another and the location of captains who drive cars, Uber matches the commute requests between users and nearby captains. In their 2019 annual report, Uber reported their presence in more than 10k cities rising 65B $ gross bookings from 7 billion trips. Autonomous driving is one of the hot technologies that relies heavily on location information of the vehicle and the surrounding objects. Figure 1.1 illustrates the need for location information to achieve autonomous driving. The vehicle has to know its own location, the location of other cars, and the location of other objects including pedestrians. This knowledge is essential in order acquire enough environmental awareness to be able to navigate.

Autonomous driving attracted a lot of research efforts from academia and industry. While GPS information can provide valuable information, it is not enough to acquire location awareness of surrounding objects such as pedestrians or sidewalks. Hence, other forms of measurements are needed such as vision based sensors (e.g., cameras and LiDAR sensors). Integrating information from different sensors in order to build an accurate understanding of the environment elevates the complexity of the problem.

Location information plays a vital role in saving lives in emergency situations.

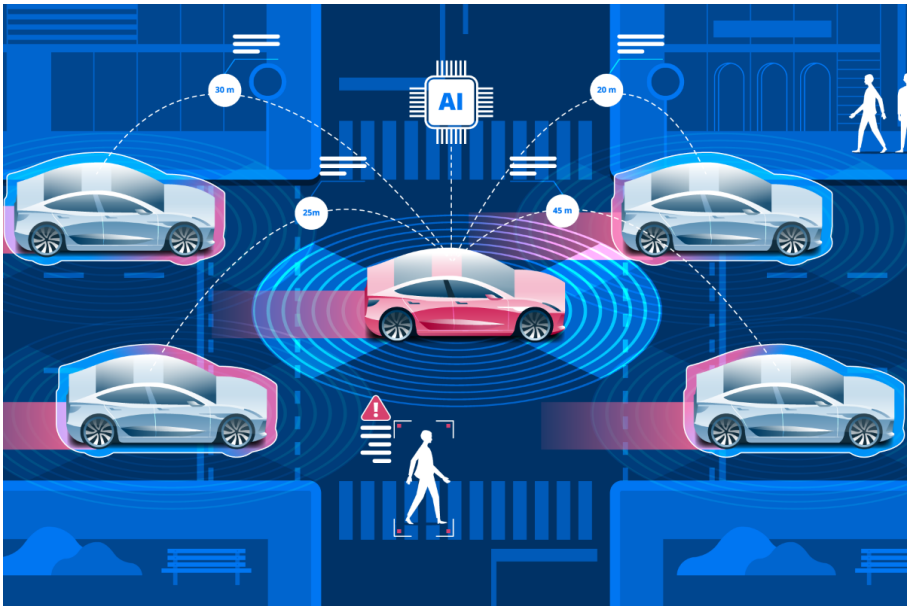


Figure 1.1: Autonomous Driving example [5]

In the USA alone, around 200k emergency calls are made daily, one-third of which are via mobile devices where most of the people requesting help are not fully aware of their locations. Having the location information readily available when the emergency call is initiated accelerates the arrival of help. In Europe, industrial resources estimate that 5000 lives could be saved every year with the knowledge of position information. This valuable effect of location information led authorities in Europe and the USA to oblige commercial wireless carriers to provide location information automatically with emergency calls [3].

In networking, packets transmission between devices depend on different routing techniques. Location information has been intensively used to enhance routing algorithms to decrease network congestion. Figure 1.2 displays a simple example where location information can be used to prioritize one path over the other using the position information. In this simple example, node x needs to send a packet to node D; knowing the location of intermediate node helps in refining the chosen path [6].

In this work, we investigate the localization problem where measurements come from communication between devices, especially the technologies related to 5G. The synergy between communication reliability and location services is very strong as better location awareness enhances communication and vice versa. More revolutionizing applications will arise with 3GPP's [7] three requirements for 5G:

1. Enhanced Mobile Broadband (eMBB): High data rates within the range of 10 to 20 Gbits/s.
2. Ultra-Reliable Low-Latency Communication (URLLC): Communication latency expected to be less than 1 ms.
3. massive Machine-Type Communication (mMTC): Highly dense machine to

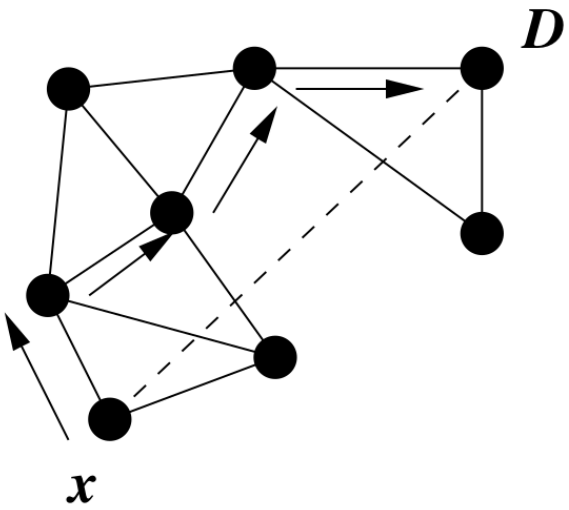


Figure 1.2: Routing example [6]

machine communication with up to 1 million connected devices per km.

These new capabilities of 5G paves the way for a lot of enhancements on the industrial level. One highly anticipated application is known as the factory of the future which refers to re-configurable factories that make use of 5G and AI advances. In such factory, plenty of sensors, actuators, and robots need to be connected to monitor different processes and adapt the factory to real time situations and objectives. Localization is a fundamental aspect of these factories as robots would need real-time instructions to avoid accidents and achieve tasks seamlessly. A peak on the need for localization in such context can be seen in Amazon's warehouses where robots transfer products in a highly dynamic environment shown in figure 1.3. The robots need to identify products and precisely know their position as well as their own to place themselves accurately under the shelf to be able to lift it. Furthermore, they need to be aware of the location of other robots to avoid accidents; this information needs to be communicated reliably and with low latency to implement the process successfully.

1.3 Localization Challenges

The localization problem has been intensively studied for decades; however, it is not considered as a solved problem [8]. This can be explained by the multitude of different contexts where localization can be applied making it difficult to come up with one accepted solution. Moreover, the advancements in hardware and software technologies open the door to pushing existing solution to unprecedented limits. The contexts depend on multiple factors including: the environment, whether indoors or outdoors, used measurements such as GPS, RSSI, or CSI, and the mobility of nodes, etc.



Figure 1.3: Amazon's warehouse order picking robots [6]

An adequate localization solution has to consider different contexts. One of the principal variables is the environment. Indoor environment has been a challenging problem for decades and is yet to settle on a widely accepted solution that meets both cost and accuracy requirements [9]. One of the challenges in indoor positioning is that it lacks access to GPS service. On the other hand, outdoor positioning has an upper hand due to its access to GPS readings from line-of-sight (LOS) communication. Other aspects of the environment include the stability of objects around the devices. Highly dynamic environment induces noise in the localization process. Also, the mobility of the nodes makes the problem more challenging.

The solution is also dependent on the available sensors or measurements used to localize. Environment detecting sensors such as cameras, LiDARs, and sonar sensors fall under the category of visual localization. In this case, the environment is frequently scanned, and any change in the scanned scene is translated to a change of the position of the device. Inter-node communication is another source of measurements to deduce distances between nodes through Received Signal Strength Indicator (RSSI), Time Of Arrival (TOA), or Time Difference Of Arrival (TDOA). The estimated distances are then used to locate nodes in what is known as range-based localization. Recently, with the increasing demand for high throughput data transmission, massive Multiple Input Multiple Output (MIMO) systems are spreading and becoming a viable option to power 5G wireless communication systems [10]. Information is sent on multiple subcarriers, and the Channel State Information (CSI) can be estimated at each subcarrier. The CSI, or channel's frequency response, describes the change that occurs to the transmitted signal due to the channel nature, frequency, and antenna's quality. This kind of richer information has been shown in [11] to be stable with respect to time, and robust against environmental changes. Thus, CSI is a reliable measurement to use for position fingerprinting.

In almost all estimation contexts, there are some state variables of interest to measure (e.g., location of nodes) based on some measurements/inputs. The input values vary largely in terms of their accuracy of presenting the truth values. Thus, their uncertainty has to be considered as they are used to compute the output and

have direct impact on its uncertainty. The output is often a function of the input. Sometimes, the functions parameters are unknown or need to be tweaked to match available data. In Appendix A, we present some of the fundamental frameworks for uncertainty propagation from input to output, and model parametrization.

In other cases, the function relating the input to output is unknown, and a general complex function is optimized to approximate the relation between input and output data. Appendix B introduces basic machine learning building blocks starting from linear regression up to Multi Layer Perceptron Neural Networks (MLP NN) used in deep learning.

1.4 Thesis outline

The rest of the thesis is organized as follows: Chapter 2 presents state-of-the-art localization solutions. Chapter 3 and 4 introduce our contributions in range-based localization and machine learning based localization, respectively. In Chapter 5, we discuss the issue of generalization of deep learning models when applied to the localization problem. Chapter 6 encompasses the conclusion and our suggestions for future work.

Chapter 2

Literature Review

2.1 Range-based Localization

Range-based localization is based on knowing the distance between nodes in a network. The network is commonly presented as a graph structure $G = (V, E)$ where vertices represent nodes, and edges represent the knowledge of the distance between connected node pairs which are in the communication vicinity of one another. Localization of network nodes is crucial for a plethora of applications, such as improvement of routing techniques [12–14] A straightforward method to address the localization problem is to equip all nodes with a location sensor (e.g., a GPS) and share the location information. There are two main paradigms to share position information: Rendez-vous-based and flooding-based. In rendez-vous-based methods, some nodes are elected to store node position information and respond to inquiries when position information is requested. The disadvantage of such a method is that it centralizes the information in only a few nodes, thus there is a risk of information loss if such nodes break down [15]. Flooding-based methods overcome this by broadcasting node position information to the network. The cost involved in such an approach is excessive message exchange, contributing to network overhead. In Semi-Flooding based Location Service (SFLS) [15], the location information is forwarded with higher frequency to close neighbors (in terms of the number of hops) and lower frequency to further nodes. This decreases bandwidth consumption while maintaining adequate knowledge of neighbors' positions.

In the case where not all nodes are equipped with a GPS, localizing non-anchor nodes require additional information to relate nodes to each other. According to the classification specified in [16], node relations can be connectivity-based, which simply indicates if a node is in connection with anchor nodes. This is used in [17] where a node's position is assumed to be the average of the known positions of the other nodes. Another type of inter-node relation is the distance between one-hop neighbors which is widely used to estimate the positions of non-anchor nodes [18, 19] . Relative distance information helps to improve the accuracy of the location. In addition, some solutions use the Angle-of-Arrival to relate nodes, or a combination of the aforementioned measurement types [20].

Some works consider the knowledge of positions of a subset of nodes known as

anchor nodes along with the distances and formulate an optimization problem to solve for the positions of non-anchor nodes. Different numerical techniques can then be applied to estimate positions, such as semidefinite programming [21] or linear programming [22]. Using Received Signal Strength Indicator (RSSI) to deduce the distance between nodes yields inaccurate and often unstable indications. Thus, some works defer from using the estimated distance values directly and utilize interval-analysis [23–25]. In interval-based analysis, instead of computing direct distances from RSSI, a set of inequalities is formulated to indicate that a node is on a ring between 2 radii based on its relative position to other nodes. Then, the defined areas for nodes can be further restrained using the Waltz algorithm [26].

Some approaches divide the problem into subproblems as in [27] where 1) base stations classify ordinary nodes into clusters based on their proximity to anchor nodes and 2) within each cluster, a node seen by three anchor nodes is located using a simple geometrical computation. In [28] , the region where the network is deployed is divided into rectangular grids as a first step, then, within each small grid, the location is refined. However, in the previously mentioned approaches, the number of GPS nodes needs to be high in order to satisfy the necessary constraints.

Generally speaking, if the distance of a non-anchor node to three anchor nodes is known, it can be located with simple geometric computations, except in the rare cases where the nodes are collinear or when some nodes overlap. This might imply a conclusion that at least three GPS nodes in the network are needed for location. However, in [16] , the proposed solution attempts to locate the network with a single anchor node. The position of each node is initially estimated as a uniform probability distribution over the deployment region. As the roaming anchor node passes by the ordinary nodes, it tweaks the distribution to locate nodes. In this context, the authors experiment their solution where the anchor nodes use a random model to traverse the network; they have a reasonable claim that traversing the network can be optimized to improve the location process. In [16, 29, 30] the probability distribution is integrated in the location process, rather than attributing a single position for each node. Such a method allows the uncertainty of measurements to be taken into account—be it the GPS position or the relative distance between nodes [16]. In [16, 29] negative information about the absence of a node in the proximity of the anchor nodes is used in the location process. This kind of information is used as a basis for our algorithm.

An approach which is seemingly far from the mentioned methods, yet can be used to address the node location problem, is Graph Layout Algorithms, such as FDP [31] and neato [32]. Even though the objective of these algorithms is generally to create an easy-on-the-eye graph, some variations make it possible to fix the positions of some nodes (the anchor nodes) and to set the suitable edge length (i.e., the distance between nodes). We have tested these algorithms using graphviz [33] and the estimated node positions are satisfactory.

Methods that use a set of distance equations and optimization techniques usually require high computation power, which is impractical for these IoT networks. In this case, the computation is done remotely in a centralized fashion. The graph

structure might not yield a unique solution. Even if the graph has a unique solution, finding this solution is proved to be NP-hard [34].

Table 2.1 summarizes different approaches to solve the localization problem.

Table 2.1: Localization approaches.

Approach	**Sensors Cost**	**Comments**
Flooding based	Expensive	Information well distributed, but bandwidth-consuming.
Rendez-vous based	Expensive	Centralized, but bandwidth-friendly.
Semi-Flooding	Expensive	Sweet spot between flooding and rendez-vous.
Connectivity based	Not Expensive	Low complexity, but low accuracy.
Optimization	Not Expensive	Adequate accuracy, but dependent on environment noise.
Interval analysis	Moderately Expensive	Adequate accuracy, and mitigates environment noise.
GPS-less	Not Expensive	Adequate accuracy, and position information is in a local coordinate system.
PCP (proposed method)	Not Expensive	Adequate accuracy, and high localization percentage.

2.2 CSI Localization

While range-based localization has been the focus of many research works for decades [35] , the demand for higher accuracy localization requires measurements more accurate than RSSI. As discussed, RSSI shows instability and is sensitive to environmental changes and various sources of noise, such as fading, distortion, and multi-path effect [11]. One attempt to overcome the noisy nature of RSSI is to incorporate some assumptions like dead reckoning [36] where the estimation of the current position is partially dependant on the previous position assuming some motion model (e.g., constant velocity). Another way is to enrich the measurement set by using different sensor data and fusing them using Bayesian methods, such as Kalman filters [37].

Channel State Information (CSI) is a more granular and stable measurement that allows for fingerprinting based localization. Fingerprinting is a method to map from a large subset of input data to an output of smaller dimension. CSI represents the effect of the channel on the transmitted signal and is represented by a complex

number as follows:

$$R_{i,j} = T_{i,j} \cdot CSI_{i,j} + N \tag{2.1}$$

The transmitted signal $T_{i,j}$ from antenna i at subcarrier frequency j is multiplied by $CSI_{i,j}$, and white noise N is added to form the received signal $R_{i,j}$. CSI is mostly used in the context of MIMO antennas where the diversity of multiple antennas and the large number of subcarriers are exploited to provide a rich granular information per position. For instance, for a 2 × 8 MIMO antenna with 1024 subcarriers, one transmission from a particular position yields $2 \times 8 \times 1024$ CSI values. MIMO antennas are gaining more popularity as they pave the way for attaining 5G's unprecedented data rate requirement of 10 Gbps [38]. Multiple simultaneous transmissions over adjacent subcarriers on the same antenna is made possible through orthogonal frequency-division multiplexing (OFDM). The fingerprinting trend has been steadily moving towards CSI and away from RSSI to achieve higher localization accuracy [39]. This is due to the richer information content provided by CSI since it is calculated per subcarrier, while the RSSI is calculated per packet. Moreover, CSI shows higher temporal stability as opposed to the high variability of RSSI.

FIFS [40] and FILA [41] are examples of the early attempts to use CSI-based indoor localization. The former utilizes MIMO antennas from several access points to build an offline radio map of CSI fingerprints to user position. This is followed by an online prediction phase where the input CSI readings are compared to the map using a probabilistic method [42]originally designed for RSSI fingerprinting. The FILA solution [41]is one of the very first initiatives to use CSI for localization in complex indoor environments. The CSI of 30 adjacent subcarriers are reduced to $CSI_{effective}$ which is then used in a parametric equation to compute the distance to target node. The parameters of the equation are deduced using a supervised learning method. Finally, using a simple trilateration method [43], the position of the target node is estimated from the computed distances to three anchor nodes. FILA's closest experimental setup to ours is a 3×4 empty room where it is safe to assume that the received signals were LOS. They attained a mean error less than 0.5 m and they reasonably argued that with the availability of more anchor nodes and with the use of a more accurate trilateration method, the error can be further reduced. In [44],a k-nearest neighbor method is used on a fingerprinting database based on the magnitude of CSI. Their estimation results outperforms FILA [41] and FIFS [40].

K-nearest neighbours algorithm estimates the position by computing a weighted average of k-nearest positions. However, this introduces a complexity due to the need to store the training samples used in the off-line learning phase which can be a critical memory and processing limitation in some applications. In [45], the correlation between CSI values is captured by creating a visibility graph. Statistical features of the graph (e.g., degree deviation, degree assortativity coefficient) are then used as input features to different learning techniques (e.g., SVM, random forest).

In [46] , the authors propose a channel sounder and utilize both real and imaginary

components of CSI with a Convolutional Neural Network (CNN) to achieve position fingerprinting. CNNs are able to extract more complex and descriptive higher-level features by processing a window of input features all together [47]. In this work the authors mainly focused on creating a flexible channel sounder architecture that allows for CSI estimation at various frequency bands and environments. More importantly, the dataset collected from their experiments is publicly available to the scientific community. This allows for making fair comparison between different methods using the same testbed. We use their public dataset to verify the accuracy of the proposed method.

Their proposed CNN with input features being the real and imaginary components yields an estimation error of 32 cm in a Line-Of-Sight (LOS) scenario. With the 2×8 MIMO antenna and 924 subcarriers per antenna, their input dimension is $2 \times 8 \times 924 \times 2$ (the last dimension represents both the real and imaginary components). With such high dimensional input along with the use of CNN, the learning and inference processes become more computationally demanding. By selecting the magnitude of the CSI as the input feature and using polynomial regression, we are able to reduce the input dimension by a factor of 38 and use a lighter weight MLP neural network and still achieve ≈ 8 times better accuracy.

Other works have used different components to achiever better accuracy. One of the very first attempts to use the phase component is PhaseFi [48]. The authors used linear transformation to calibrate the phase component estimated at thirty subcarriers and three antennas of Intel's WiFi Link 5300 NIC. The calibrated phase is then used as input to a three-layer Neural Network to achieve position fingerprinting. The authors showed that localization using CSI with commodity hardware is more accurate than using RSSI. The mean error in the Line-Of-Sight experiment is $\approx$ 1 m. Direct comparison with our results is not feasible due to the differences in the experimental area, the number of subcarriers, and the antennas. However, an important point of comparison with our method is the choice of the phase component. Their reasoning in choosing the phase over the magnitude is that it is less sensitive to obstacles and that it is more stable in general. Nevertheless, we show that the magnitude is more stable through a statistical analysis of the dataset.

In DeepFi [11], Intel's WiFi link 5300 NIC with three antennas and 90 subcarriers per antenna is used for localization. They reached a similar conclusion to use the magnitude of CSI as the input features. They use trained weights between layers in a four-layer MLP as fingerprints to the position of the transmitter. This is achieved through a greedy learning method that trains the weights of one layer at a time based on a stack of Restricted Bolzmann Machines (RBMs) to reduce the complexity [49] . Our method of complexity reduction is based on a simpler process using polynomial regression. It is difficult to compare the estimation error between our method and theirs because of the difference in the number/quality of antennas, environmental noise level, etc. However, the mean error obtained using our proposed method with two antennas and 924 subcarriers is ≈ 6 times less than their mean error which is relatively large, giving us some confidence that our method outperforms theirs. Similar accuracy is achieved using both the magnitude and the phase of CSI fed into the K-nearest neighbours algorithm [50].

2.3 Deep Learning Enhancements

Traditional MLP NN has been proved in [51] to be a universal approximator. Concretely, for any continuous hypercube function with d dimensional input within the range $[0,1]^d$, real number output and every positive ϵ, there exists a 1-hidden layer neural network with a sigmoid activation that can approximate the function with at most ϵ error in functional space. This proof shows the strength of NN approximation capabilities, especially since ϵ can be chosen to be infinitesimal (very close to zero) which would mean a highly accurate approximation of the function. However, it does not tell us anything about the needed number of units in the 1 hidden layer which could be exponentially large and infeasible to train. Also, fitting a set of data points representing a function does not necessarily mean that the approximation would fit well the data points to be predicted.

The fundamental end goal of artificial intelligence is to mimic human intelligence in adapting and reacting to new experiences based on previous ones that might not be strongly related. Thus making "infinite use of finite means" is a principal aspect of human intelligence [52]; this reflects the use of few resources to learn new references and relations, for example, having knowledge about the meaning of few English words (finite means) and being able to generate lots of sentences with different meaning (infinite use). This general aim remains out of reach for the current AI despite the unprecedented advancements in the last decade [53]. Prior to the recent boom of AI, models relied on engineered features that are chosen by experts in the field of application. This method achieved decent success since experts are able to choose adequate features capable of expressing the transfer from input to output. However, the choice of features is limited by the experts' knowledge of the problem which is seldom complete. Deep NN on the other hand delegate the feature selection to the optimization process adding flexibility that allows for discovering interesting features beyond experts' knowledge.

With cheap data and computational resources, many AI approaches switched from the manual feature engineering part of the spectrum to the end-to-end learning without restrictions or bias on the learnt features [54–56]. We believe that both paradigms have their pros and cons and the best of interest of AI is to combine their strengths to take one step towards human intelligence. The midway between manually choosing features by experts and complete flexibility of MLP NN to find features can be achieved by inducing some bias in the deep learning model to learn features about the relation between input features. In this paradigm, the experts contribute by suggesting meaningful connections between input features, and the deep learning model architecture is altered accordingly. The architecture does not enforce selecting some specific features but rather shifts the focus to interesting relations and the model uses its flexibility to find interesting features in the focused area. This paradigm led to the emergence of several variants of MLP NN which achieved major breakthroughs in different applications, such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Graph Neural Networks (GNNs).

2.3.1 Convolutional Neural Networks

In appendix B, we give a basic introduction to the traditional multi-layer perceptron Neural Networks. MLP NN is able to achieve highly accurate results in some tasks like recognizing hand written digits [57]. The input features in this case are individual pixels in a low resolution image where the value of each pixel represents the grayscale value. However, the general form fails to achieve accurate results for more complex tasks [53]. Thus, several attempts have been made to enhance the general NN form to adapt it to the nature of input features. Let us take the MNIST dataset as an example for hand written digit detection [58]. This dataset is composed of 60,000 train set samples and 10,000 test set samples hand written by hundreds. Each sample is a 28 × 28 grayscale image. Figure 2.1 shows several image examples. A reduced numerical visualization of the digit "One" is shown in a 14 × 14 matrix where the grayscale is normalized between 0 and 1.

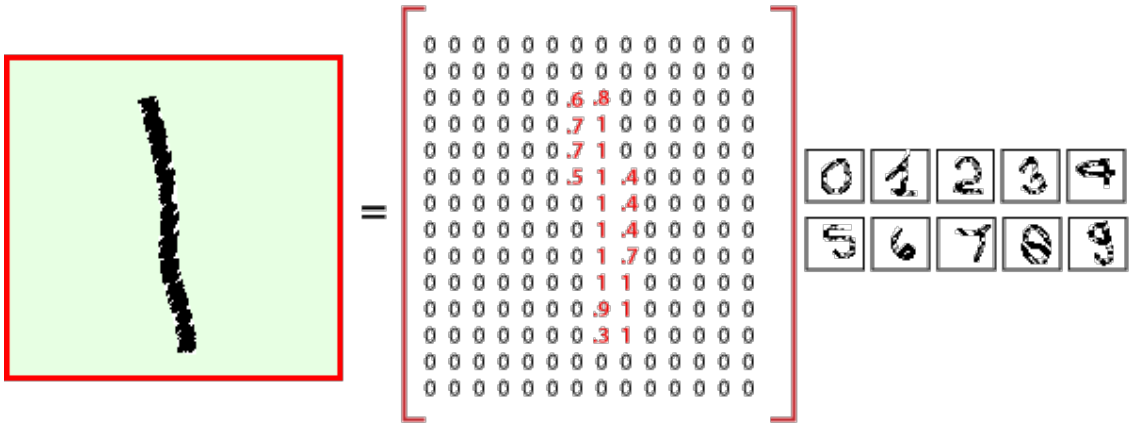


Figure 2.1: MNIST dataset written digit example [59]

While our brains are able to analyze the images quickly, the problem is far from being easy. To appreciate the underlying complexity of such operation, one has to think of the actual representation of the image. The computer sees the set of numbers represented in the matrix at each pixel. Our brain receives similar information and translates them quickly to the colors that form our visualized world. However, if we are just to see a set of numbers, especially if the pixel values are RGB and the background is not zeros but has some color, the problem's complexity would probably be unsolvable by our brains.

Keeping in mind that each pixel is an individual input feature, concluding the digit depends on a set of neighboring lit features. The architecture MLP NN does not enforce any bias towards neighboring pixels. Rather, it attempts to learn weights and biases relating all pixels equally important whether close or not. This is due to the fully connected architecture. This makes the learning process much less efficient. Take the digit "one" example in figure 2.1, the MLP NN can learn that this specific combination of pixels outputs "one" as it sees the pixel values as 1D vector. If the

digit is shifted right or left, the model would treat shifted "ones" as different and would have to learn separate weights and biases to estimate "one". It will be very difficult to capture the similarity between the original digit and the shifted one in the MLP NN 1D input format. Thus, to generalize the learning, each image should be shifted right and left for the model to learn all possible shifts. However, this is inefficient because the number of learning examples could explode making the learning process infeasible in terms of time.

Several deep learning architectures have emerged to introduce a bias depending on the nature of the problem. One heavily used NN architecture especially in image processing context is Convolutional Neural Networks (CNN) [60]. CNNs' architecture is especially resilient to the shift problem. CNN makes use of two main characteristics of images: locality and translation invariance. Locality means that nearby pixels tend to be correlated, while location invariance corresponds to the invariance of output when objects are shifted in the image [61]. This is achieved by moving from the fully connected architecture of MLP NN to a locally connected one in CNNs. The local connection is achieved by sliding a 2D kernel over the image and convolving the kernel weights with the pixel values using dot product. The side of the kernel is a design choice but it is commonly of a small size (e.g., 3×3 or 5×5). Multiple kernels are convolved with the image to produce multiple feature maps. If the kernel values highly match the pixel window of the image, this means that the feature of the kernel exists in this particular region of the image. Figures 2.2 and 2.3 from Deepmind's lecture on CNN at UCL illustrate the difference between input for one unit in MLP NN and CNN.

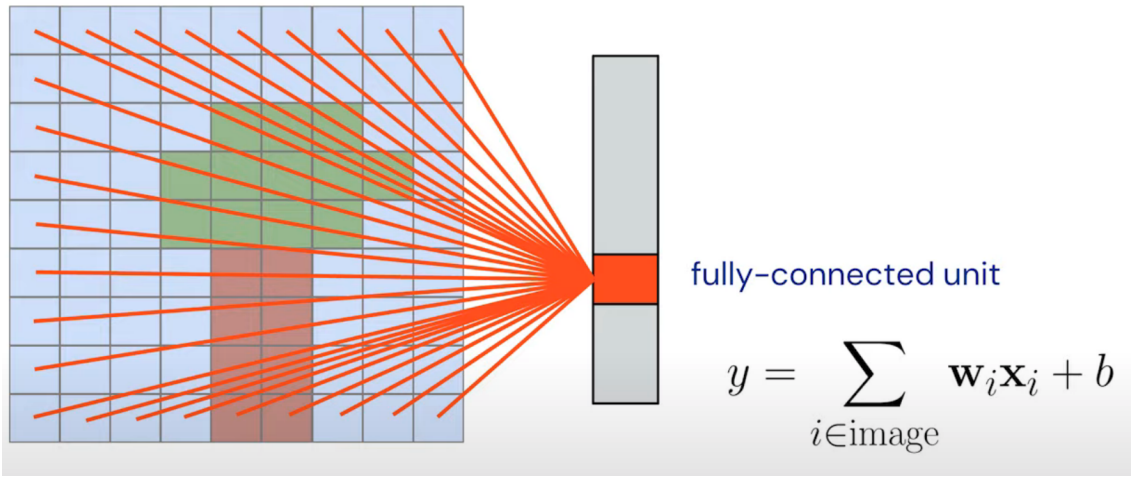


Figure 2.2: MLP NN unit input [61]

The reduced number of input to each unit in the CNN paradigm has two main advantages. First, the number of weights is reduced which leads to faster computations. Second, it introduces a kind of bias or focus to extract features from nearby pixels and excludes relating far pixels which is a sensible bias in the image recognition context. The kernel weights are learnt in a similar fashion to that of the MLP NN explained in chapter 1. This change of architecture which introduces a relational bias between nearby pixels has revolutionized the image recognition field.

Probably the highest impact of deep learning on image recognition capability

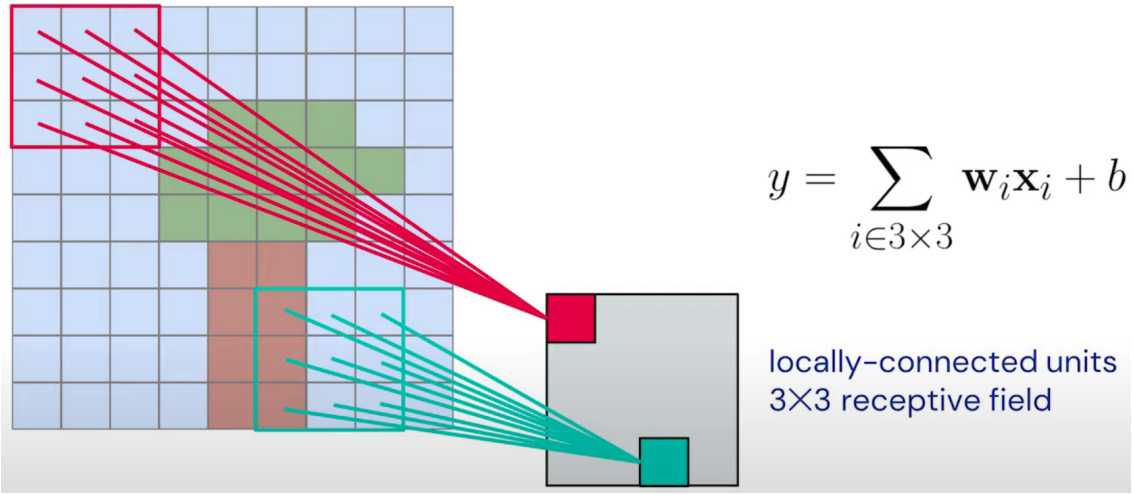


Figure 2.3: CNN unit input [61]

can be seen by analyzing the methods used in the ImageNet challenge [62]. This competition provides for competitors a set of $\approx$ 1.4 M images that belongs to 1000 classes. The challengers are expected to build a model to predict classes from images by learning from the provided dataset. This large pool of images attracted numerous researchers from different institutes in the industry and the academia. Back when the competition was first launched in 2010, the dominant image classification techniques were feature engineering based. In those techniques, features were manually designed or chosen by competitors and used to differentiate between classes. These features are selected based on intuition or experience of the participant; thus, there is almost no learning involved by the model [61]. These methods were able to achieve an error rate around 27 %. In 2012, one of the first CNNs used at this scale, AlexNet [63], achieved a remarkable jump in error rate reducing it to 16 %. The unprecedented success of AlexNet lead to a drastic increase in the use of CNNs in the years to follow over traditional computer vision techniques [61]. VGGNet [64] and GoogLeNet [65] are prominent examples of the evolution of CNNs using different architectures and increasing the network depth, achieving $\approx$ 7 % error rate. Residual connections technique was a particular innovation in CNNs that lead ResNet [66] to decrease the error to 4 %. Improving error rate was marginal after ResNet [66] in subsequent years. The takeaway from the evolution of ImageNet challenge [62] over the years is that hand crafted features are far less efficient that automatically learnt features by deep learning models. More importantly, altering the deep learning model architecture to bias it towards learning features based on the nature of the problem can lead to breakthroughs in the learning process.

2.3.2 Graph Neural Networks

As previously seen, the availability of data in the ImageNet challenge drove a huge amount of research and innovation over the years. Besides data availability, powerful computation resources opened new horizons for model complexity and accuracy. Computational capabilities drove one interesting trend in deep learning research

which is building a deeper and more complex networks to improve performance. To put things into perspective, accurate models in the ImageNet challenge [62] encompassed a large number of weights, in the order of hundreds of millions of parameters [64]. Paradigm changes lead to breakthrough improvements (e.g., using deep learning instead of feature engineering or innovating in CNNs architecture with residual connections). Increasing the complexity of the model especially by going deeper can lead to accuracy improvements [67]. However, the model could easily suffer from incapability to generalize.

Other biased forms of deep learning have emerged for other types of data. While CNNs handle space invariance, Recurrent neural networks (RNNs) are structured in a way that conserves time invariance and generally deals with sequences [68]. Natural Language Processing [69–71] is one application where RNNs are used to understand spoken languages. In this paradigm, instead of enforcing relational learning between nearby pixels, the current state is related to previous states. Object tracking is another form of sequence where consecutive positions of a moving object are strongly correlated.

In spite of the unprecedented success achieved using deep learning models, more complex contexts in language and scene understanding remain challenging to solve [72–75]. Adding to the problem's complexity is the fact that not all problems are structured in a matrix format (2D image) or in some ordered sequence. This raises a demand for a flexible framework that is able to introduce the bias in finding relations between inputs depending on the nature of the problem. This direction hits the sweet-spot between full manual feature engineering, and complete end-to-end fully connected MLP NN where no bias is included in the architecture. Graph structure is a flexible framework that can be used to introduce this relational bias for the NN to learn more meaningful features that are able to generalize beyond the training data. The family of neural networks with the graph architecture are often referred to as Graph Neural Networks [53].

Deepmind's *graph_nets* open source framework is one of the recent powerful implementations of GNNs [76]. In their demo, a GNN is trained to predict the shortest path between two given nodes in an 8 to 16 node graph. The generalization capability is vividly highlighted where the GNN is able to accurately predict the shortest path in graphs that are twice as large as the graphs used to train the model. The main aspect that GNNs aim to satisfy is combinatorial generalization which is considered a key true generalization [53]. The relational bias included in the GNN architecture which relates some entities forces the choice of a solution over another without depending on the specific training value [77]. Abstracting away from the training set values is a crucial step towards generalization which is depicted in the example of training to estimate the shortest path in small graphs and being able to generalize to larger ones.

Chapter 3

Position Certainty Propagation in MANETs

MANETs are characterized by the highly dynamic nature of their nodes in terms of mobility as well as entering and exiting the network. Thus, realizing a service in such context requires a distributed solution. Moreover, in some cases such as IOT, the solution needs to be light to avoid high energy consumption. In this chapter, we introduce a location service (PCP) for MANETs that takes into consideration the distributive and lightweight requirements. Furthermore, the localization problem is solved on two levels:

1. Self-localization where nodes estimate their own positions when enough first hop neighbors have their positions estimated.

2. Global localization where the estimated positions of nodes are shared among nodes using an efficient algorithm SFLS [15].

3.1 Problem Formulation

To address the localization problem, the network is modeled as an undirected graph $G = (V, E)$ where $V = \{0, 1, ..., n\}$ are the nodes of the network and $E = \{(0,1), (0,2)...\}$ are edges connecting two nodes located within the communication range of each other [78]. The edges connecting nodes have weights representing the distance between communicating nodes. Nodes have two types: anchor nodes whose positions are known in advance and non-anchor nodes whose positions are unknown. The network is assumed to be composed of m anchor nodes and n non-anchor nodes. The proposed solution requires two constraints in the network:

(a) The network has a minimum of three anchor nodes.

(b) Three anchor nodes are in the communication range of at least one non-anchor node.

In order to illustrate the proposed solution, consider a small network example shown in figure 3.1. The network is composed of 15 nodes. Each node is labelled with a unique *id* where blue colored nodes are anchor nodes and the red ones are non-anchor nodes. The nodes are distributed uniformly over a 20m $\times$ 20m region. All nodes have a communication range of 8 m. An edge between two nodes indicates that the distance between them is less than the communication range. The objective is to localize as many non-anchor nodes as possible exploiting the prior knowledge which is composed of the positions of the anchor nodes and the distance between nodes within the communication range of each other.

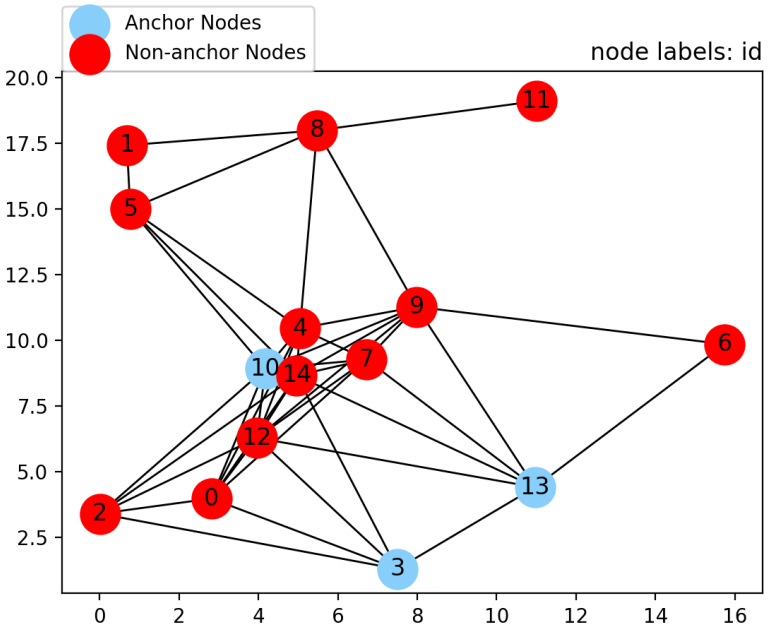


Figure 3.1: 15-node network example

3.2 Position Certainty Propagation

The distance between nodes are conventionally estimated in range-based localization techniques through the measurements of RSSI, TOA, AoA etc. In this section, we do not address this part of distance estimation and assume that the distance information is readily available. The algorithm of two main cases, the first is when a non-anchor node is in vicinity of three or more anchor nodes or nodes whose positions have been computed. The second case is when a non-anchor node is in the vicinity of exactly two nodes with known positions.

3.2.1 Three nodes with known position in their vicinity

Consider the "three nodes in vicinity" case which at the start of the algorithm has to be satisfied based on the two mentioned constraints in section 3.1.

Each node communicates with nodes within the communication range and counts how many of them have their positions determined. Lets take the network example in figure 3.1, initially, the only nodes with known positions are the three anchor nodes 10, 3, 13. The number of first hop neighbors with known positions is shown for all non-anchor nodes in figure 3.2. Two nodes are within communication range with three nodes with known positions which are the three anchor nodes. Those two nodes attempt to compute their own positions via triangulation using the position of their neighbors and the distance between them.

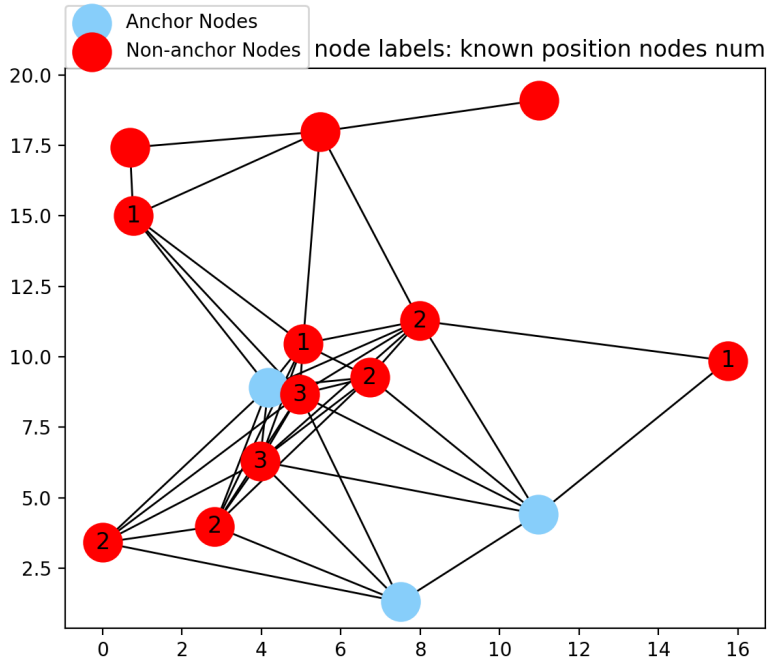


Figure 3.2: Counting number of known position neighbours

In order to visualize the self localization process, let's take node 12 as an example. Knowing the position of the anchor nodes, three circle are constructed centered at each of the anchor nodes with radii equal to the distance from each anchor node to node 12. Figure 3.3 shows the intersection of the three circles that is used to compute the position of node 12. The intersection of the three circles is calculated via triangulation and is chosen as the node position.

Since the distance measurements are not certain, their errors are modeled with a Gaussian distribution with a zero mean and a standard deviation that represents the degree of noise in the environment. The Gaussian assumption is is a widely accepted model that is also experimentally validated in [79]. Due to measurements errors, the three circles do not necessarily intersect at one exact point. To overcome this, we use a variant of the residual weight algorithm [80]. This algorithm was originally implemented to mitigate the effect of Non-Line-Of-Sight (NLOS) measurement errors. When the distance and position information of more than three nodes is collected, an estimated position is computed for each possible combination of three or more nodes using least square error minimization. To illustrate this, for position/distance information of four nodes, $\binom{4}{3}$ positions are computed for all three

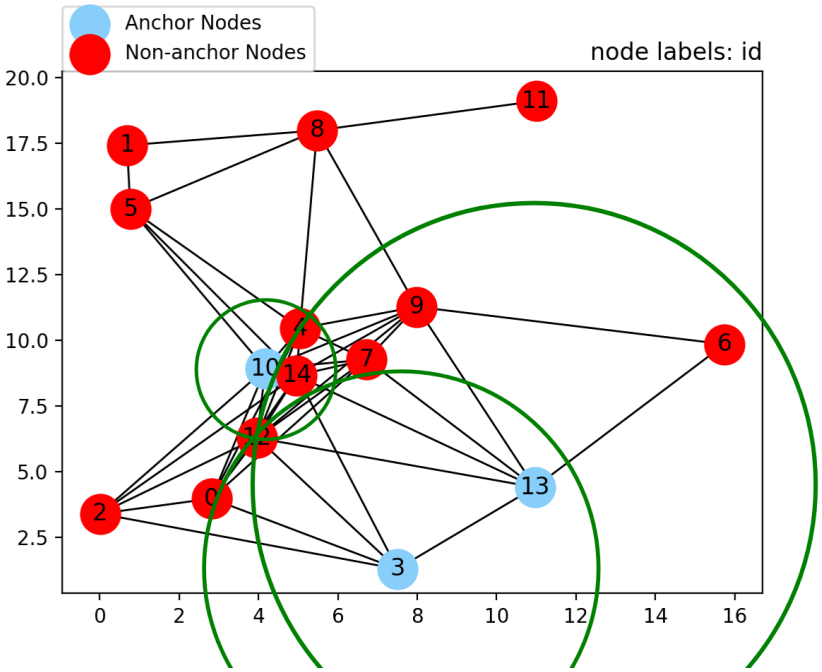


Figure 3.3: Computing position from the intersection of three circles

node combinations and one position for the four nodes together. Each estimated position is given a weight that is inverse to the normalized residual error. Finally, the chosen position is the weighted average of all positions. Another possibility is to choose the position with the minimum residual error. Since minimizing the error does not always guarantee an estimation that is the closest to the actual position [18], we choose not to do the least-square error minimization in order to minimize computation effort. Rather, for position/distance information for three nodes, we compute a position for each combination of node pairs (in this case 3 combinations) by computing two intersections of the two circles and eliminating one of the positions using the third circle. Then the weighted average of the three positions is chosen to be the estimation. The weight is the inverse of the error between the distances to the computed position and the measured distance.

The localization process is formally described in our work [81]. "Let us consider for a node pair $[A, B]$ whose known positions are $[a.x, a.y]$ and $[b.x, b.y]$ respectively. The distances to node C whose position is to be computed are ac and bc respectively. First the distance between A and B is computed:

$$ab = \sqrt{(a.x - b.x)^2 + (a.y - b.y)^2}. \tag{3.1}$$

The intersection between two circles generally yield two positions. The computation of the two possible positions is easier if the two circles lie on the horizontal axis and one of them is at the origin. Thus, $\vec{a}$ is to be subtracted from a and b. Next, the rotation angle shown in Equation 3.2 is applied so that a is transferred to the origin and b resides on the x-axis.

$$\theta_{rot} = arctan\left(\frac{b.y - a.y}{b.x - a.x}\right) \tag{3.2}$$

After applying the translation and rotation, the two intersection points will have the same x-coordinate. The y-coordinate for one intersection point is above the x-axis and the other is below the x-axis. The rotated coordinates of the intersection points are shown in Equations 3.3 and 3.4.

$$c.x_{rot} = \frac{ab^2 + ac^2 - bc^2}{2ab} \tag{3.3}$$

$$c.y_{rot} = \pm\frac{\sqrt{(ab+ac+bc)\times(ab+ac-bc)\times(ab-ac+bc)\times(-ab+ac+bc)}}{2ab} \tag{3.4}$$

Finally, to restore the intersection points in the original coordinate system, the opposite rotation and translation are applied to the rotated coordinates as follows

$$c = \begin{bmatrix}\cos(\theta_{rot}) & -\sin(\theta_{rot}) \\ \sin(\theta_{rot}) & \cos(\theta_{rot})\end{bmatrix}\begin{bmatrix}c.x_{rot} & \pm c.y_{rot}\end{bmatrix} + \vec{a} \tag{3.5}$$

The next step is to eliminate one of the two positions using the third position/distance pair $[d, dc]$ of node D, whose position is also known."

The elimination in this case is done by choosing the position whose distance to the third node D is approximately equal to dc. This process is valid in general. However, in some rare case this process is not applicable when the three known position nodes are collinear or when two of the nodes have the same position. When such cases occur, one of the known position nodes obstructing the localization is neglected. Then other known position nodes in vicinity are utilized instead. The error for the estimated known position is the absolute difference between dc and the distance between the chosen position and d. The weight given for the chosen position is $1/error$. Finally, the estimated position is the weighted average of the chosen positions. If there are more than three one-hop neighbors with known positions, the process is done for all 3 one-hop neighbors combinations and the estimate chosen is the one that yields the least error.

In case the three known position nodes are collinear, two of the three nodes are chosen so that the largest angle of the triangle of the two nodes (with known positions) and the node to be localized, is as close as possible to 90°. This criterion is chosen so that the two possible positions of the intersection of two circles are further from each other, which helps in the elimination step that follows. The localization process using the positions and distances to only two one-hop neighbors is explained

in the following section.

3.2.2 Two nodes with known positions in vicinity

The case where there are only two known position nodes in vicinity is considered a bottle neck which inhibits other algorithms [18, 19] from localizing nodes at less-connected regions in the network and thus the position discovery does not propagate. The first step is to compute the two possible positions as previously shown in Equations (3.1) to (3.5). The information required to eliminate one of the two positions is:

1. The maximum range of the communication.

2. The list of received positions of other nodes in the network and the IDs of one-hop neighbors.

The position to be estimated is expected to be within the maximum range of one-hop neighbors, and outside the maximum range for n-hop neighbors where $n > 1$. This requires that the node stores the IDs of all one-hop neighbors and updates them frequently. Also, the received positions of neighbors in the network are stored in a table.

In order to eliminate a position, the distances to all stored positions of other nodes in the network are computed. If a computed distance to a one-hop neighbor is larger than the maximum range of communication, the position is eliminated because it implies that this 1-hop neighbor is outside the maximum range of communication. On the other hand, if the computed distance to an n-hop neighbor is smaller than the communication range, the position is also eliminated because it implies that this n-hop neighbor is one hop away. If one of the two positions is eliminated, the other position is chosen to be the estimated position. If neither of the two positions is eliminated, the estimation is discarded until more information is available.

To illustrate, we continue the localization process of nodes in the 15-node network previously shown in Figure 3.3. The continuous localization of nodes propagates until there are no more nodes that can be localized with three or more nodes in the vicinity. This state is shown in Figure 3.4 where node 6 has only two nodes with known positions in the vicinity, 9 and 13. The two possible positions from the two circle intersections are shown as blue octagons. Since node 6 is aware that it has only two 1-hop neighbors, computing the distance between the lower left blue octagon to any node with known position (e.g. node 0), yields a distance smaller than the maximum communication range. And since node 0 does not belong to the list of 1-hop neighbors, this position is eliminated and the other position is chosen to be the estimate as it does not introduce any conflicts. Similarly, node 1 can be localized using nodes 5 and 8.

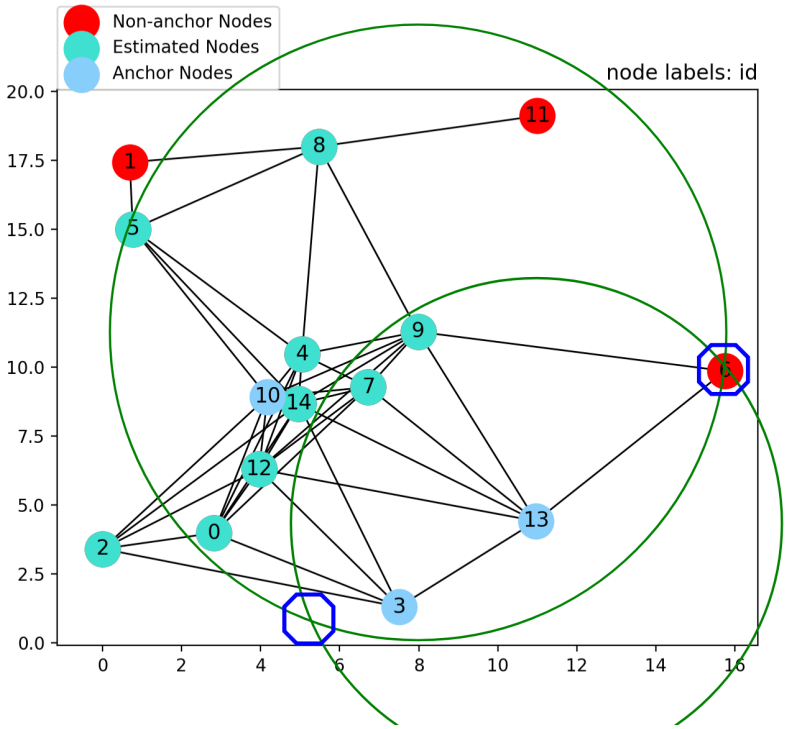


Figure 3.4: Computing position based on two nodes with known positions

The algorithm continues to compute the position of nodes with the two previously mentioned cases until no more positions can be estimated. This is the case when a node sees two nodes with known positions within its vicinity but none of the two possible positions introduces conflicts and thus cannot be eliminated. Also, when a node is seen only by one neighbor with known position, it may be localized anywhere on the circle centered at the node with known position and whose radius is the distance between the two nodes. Therefore, its position cannot be effectively established. This can be seen when attempting to compute the position of node 11, where its only neighbor with known position is node 8. The final outcome of the algorithm in Figure 3.5 shows that only node 11's position cannot be computed.

The main advantage of this method is its lightness as it comes down to a series of triangulation steps. The next section discusses the implementation of the proposed solution in a distributed manner.

3.3 Algorithmic Implementation

In this section, we present the algorithm implemented on each node in the network. The first part of the algorithm is sharing positions (if found) through the network in a bandwidth efficient manner. The second part is the previously described localization method.

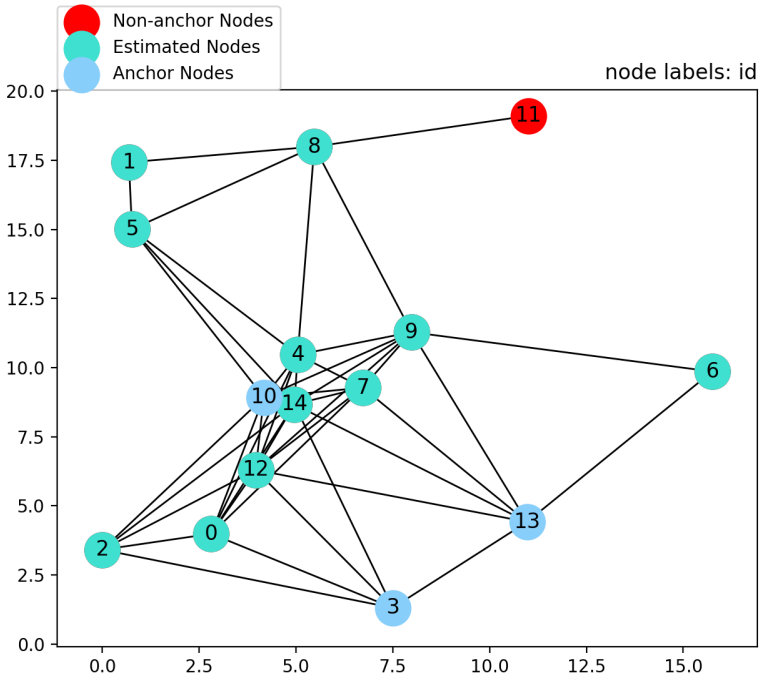


Figure 3.5: The final result of the algorithm

3.3.1 Semi-Flooding based Position Sharing

We use a semi-flooding method (SFLS) [15] to share known positions because it takes advantages of flooding and rendez-vous based methods while mitigating their drawbacks. In this method each node broadcasts its own position (if found) with an identifier like the time-stamp or a sequence number. A node that receives another node's position with a higher sequence number or time-stamp re-broadcasts once every two receivables. In other words, when a node receives a position for the first time, it rebroadcasts, the next time it receives the new position of the same node, it does not rebroadcast and keeps alternating the rebroadcast decision for the following received positions. This method is bandwidth friendly and compatible with our localization algorithm because nodes need more frequent information about their closer neighbors. Assuming the broadcast interval is t seconds, one-hop neighbors receive the position info every t seconds, two-hop neighbors receive position information every 2 t seconds, three-hop neighbors receive position info every 4 t, n-hop neighbors receive info every 2^{n-1} t. In [15], it has been shown mathematically that the mean number of broadcasts $\overline{m_i}$ performed to update the position information of node i grows approximately linearly with the number of nodes in the network where nodes are uniformly distributed. This ensures scalability with respect to bandwidth consumption. The broadcasting of positions, receiving them and initiating the self localization algorithm is illustrated in Algorithm 1.

Three procedures are described in Algorithm 1:

(a) broadcast own position: startup method is called every t seconds for anchor

Algorithm 1 SFLS Position sharing

1: $ownSeqNum = 0$ (Sequence number starts at 0 for all nodes)
2: $retransmitDecision$ (stores retransmission decision for all nodes, initialized to true for all nodes)
3: N (list of received node positions) and N_1 (1-hop neighbors with known positions)
4: Id_1 : list of IDs of all one-hop neighbors and $distanceTo$ stores distances to 1-hop neighbors
5: **procedure** BROADCAST_OWN_POSITION
6: **if** $nodeType == anchor$ **then**
7: $ownSeqNum++$
8: $pos = anchor_pos$
9: **else if** position is estimated **then**
10: $pos = estimatedPos$
11: **end if**
12: packet = [$ownID$, pos, $ownSeq$, $ownID$]
13: broadcast packet
14: **end procedure**
15: **procedure** RECEIVE($sendID, pos, seqNum, ID_{1-hop}$)
16: $prevSeqNum = seqNum[sendID]$
17: **if** $seqNum > prevSeqNum$ **then**
18: $seqNum[sendID] = seqNum$
19: $N[sendID] = pos$
20: update $distanceTo[ID_{1-hop}]$
21: **if** $sendID == ID_{1-hop}$ **then**
22: $N_1[sendID] = pos$
23: **end if**
24: **if** $nodeType != anchor$ and $seqNum > ownSeqNum$ and $N_1.size >= 2$ **then**
25: $selfLocalize()$
26: **end if**
27: **end if**
28: $rebroadcast(sendID, pos, seqNum, ownID)$
29: **end procedure**
30: **procedure** REBROADCAST($sendID, pos, seqNum, ownID$)
31: **if** $retransmitDecision[sendID] == true$ **then**
32: packet = $[sendID, pos, seqNum, ownID]$
33: broadcast packet
34: $retransmitDescision[sendID] = false$
35: **else**
36: $retransmitDescision[sendID] = true$
37: **end if**
38: **end procedure**

nodes at the beginning of the simulation. For non-anchor nodes, it is called once their positions are estimated to start broadcasting their positions with the interval t . Note that the sequence number that indicates the freshness of position information is incremented only by anchor nodes. This is because the position information originates from the anchor nodes while the localization of all the other nodes is more or less inherited from them. Thus, when a non-anchor node is localized, it updates its sequence number to be equal to that of the nodes used to localize it.

(b) receive: called every time a packet is received. The inputs to the procedure are the ID of the original creator of the packet $sendID$, the position of sending node pos, the associated sequence number $seqNum$, and the ID of the one-hop neighbor that relayed the packet ID_{1-hop}. When new information arrives, this method checks if the gathered information is sufficient to attempt to estimate a node's position. If so, it calls $selfLocalize$ method which is described in Algorithm 2.

(c) rebroadcast: called by receive method after the received position is processed to decide whether to forward the received packet according to the SFLS rule.

3.3.2 Self Localization

The second part of the algorithm is the localization part that was explained in the previous section. Thus, each time updated position information is received an attempt is made to localize node by calling $selfLocalize()$ in $receive$ method. The localization procedure is further detailed in algorithm 2.

Let N_1 be the list of detected neighbors with known positions within communication range of the node. When N_1 includes two or three nodes, the position of the node can be estimated using the *intersection* method. If N_1 includes two nodes, the *intersection* method returns the two possible positions, one of which is to be eliminated if possible. Let N be the list containing position information of all the received positions of network nodes. The first line gets the 1-hop neighbors whose positions are known and have the highest sequence number available. This ensures that the positions used to localize the node are from the same time step and thus the information is coherent. Then, if the number of 1-hop node positions is three or more, the *intersection* method returns one position that is stored in *estimatedPos* and the sequence number is updated to be equal to sequence number of nodes used in localization. If there were two 1-hop nodes with known positions, the *intersection* method returns two possible positions which are then tested in an attempt to eliminate one of them using the procedure explained above. Once the *estimatedPos* is updated it is then used together with the sequence number in the next *broadcast_own_position* call.

Algorithm 2 Position Certainty Propagation: $selfLocalize()$

1: $mostRecentOneHop$ = list of 1-hop neighbors with the highest sequence number
2: $maxSeqNumFound$ = seqNum of any of the nodes in the $mostRecentOneHop$
3: **if** $mostRecentOneHop$ has 3 or more nodes **then**
4: $estimatedPos = intersection(mostRecentOneHop)$
5: $ownSeqNum = maxSeqNumFound$
6: **else if** $mostRecentOneHop$ has 2 nodes **then**
7: [pos1, pos2] $= intersection(mostRecentOneHop)$
8: $P1Neighs$ = n **for** each $n \in N$ where $distance(n,pos1) <= CommRange$
9: $\overline{P1Neighs} = N - P1Neighs$
10: $P2Neighs$ = n **for** each $n \in N$ where $distance(n,pos2) <= CommRange$
11: $\overline{P2Neighs} = N - P2Neighs$
12: $pos1IsCompatible \iff$ **for** each $n \in P1Neighs$, $n \in Id_1$ and **for** each $m \in \overline{P1Neighs}$, $m \notin Id_1$
13: $pos2IsCompatible \iff$ **for** each $n \in P2Neighs$, $n \in Id_1$ and **for** each $m \in \overline{P2Neighs}$, $m \notin Id_1$
14: **if** $pos1IsCompatible$ and not $pos2IsCompatible$ **then**
15: $estimatedPos = pos1$
16: $ownSeqNum = maxSeqNumFound$
17: **else if** $pos2IsCompatible$ and not $pos1IsCompatible$ **then**
18: $estimatedPos = pos2$
19: $ownSeqNum = maxSeqNumFound$
20: **end if**
21: **end if**

3.4 Performance Evaluation

3.4.1 Comparison with GPS-free method

In order to assess our algorithm we compare it with an existing solution [19], which we refer to as GPS-free. We implemented the first part of the GPS-free algorithm where node positions are computed in a local coordinate system of one of the nodes in the network. The paper further explains how to choose a stable coordinate system for the network, which does not concern us since we evaluate the solution for a static instance of the network. The algorithm compromises the following steps:

1. Each node creates a local map composed of itself and the maximum possible number of 1-hop neighbors via triangulation, making itself the origin.
2. Node k can transfer its coordinate system to node i if both nodes exist in the local map of one another in addition to a third node common to both local maps.
3. All nodes in the network attempt to transfer their coordinate system to node i so that all node positions are computed in one common coordinate system

We refer the reader to the GPS-free article [19] for further details of how these steps are executed. Observing the behavior of GPS-free in some graphs we figured out an improvement that can increase the percentage of localized nodes. The improvement concerns the condition that only nodes who cannot build a local map are transferred to another coordinate system via triangulation. We quote from the GPS-free paper: "The nodes that are not able to build their local coordinate system but communicate with three nodes that already computed their positions in the referent coordinate system can obtain their position in the Network Coordinate System by triangulisation" [19]. We extended this behaviour to nodes that built their local map but still cannot transfer their local coordinate system to the referent coordinate system; in this case the node only computes its position in the reference coordinate system. This extended version of GPS-free will be referred to as GPS-free ext.

3.4.1.1 Experimental Setup

Our objective is to compute the positions in the global coordinate system using GPS information of three nodes. The distances between nodes are assumed to be certain in this experiment. In the GPS-free method, even though the positions are computed in a local coordinate system, it is possible to transfer the node positions to the global coordinate system if the 3 GPS nodes have their positions computed. Similar to our approach, GPS-free is based on triangulations. However, it does

not consider the case when a node is within the range of only two known position nodes. The results of this study highlight the added benefit of using such a case. Each simulation run is a connected geometric graph (aka a disc graph) where nodes are randomly positioned in a 100×100 m grid. Three anchor nodes are randomly chosen so that at least one non-anchor node is within their communication range. Our algorithm is run to compute the positions of the non-anchor nodes as previously described. GPS-free is run when the node chosen to have all nodes transferred to its coordinate system is the one that maximizes the number of estimated nodes. This ensures that the best possible result from the GPS-free method is obtained without taking into account whether the GPS nodes are among the nodes with the computed positions. The success percentage is the percentage of nodes whose their position is computed. Two experiments are conducted, one varying the maximum communication range $[14, 15, 16.., 23]$ m while keeping the average number of nodes constant at 100 nodes. In the second experiment, the communication range is kept constant at 14 m while the average number of nodes varies $[100, 115, 130, 145, 160]$. The number of nodes in each experiment follows a poisson distribution with the given average. In each graph configuration the experiment is repeated 1000 times and the confidence interval is shown as a vertical bar around the point.

3.4.2 Experimental Results

PCP is compared to GPS-free and the extended version GPS-free ext by comparing the percentage of localized nodes in different graph configurations. We start by varying the maximum communication range in figure 3.6 from 14 m up to 23 m with 1 m increment from one experiment to the other. The x-axis shows the average node degree at each maximum communication range. Put differently, for the first configuration each node has a maximum communication range of 14 m and the average node degree over the 1000 simulations is approximately 5.5; the next configuration is with a range of 15 m which gives an average node degree of 6.3, etc. It can be clearly seen that the maximum communication range has a direct impact on the number of localized nodes. As the maximum communication range increases and consequently so does the average node degree, it is possible to estimate the positions of more nodes. When the average degree is ≈ 10, our algorithm is able to compute the positions of $\approx 90\%$ of nodes. Also, our algorithm shows a higher success percentage for all configurations than GPS-free and the GPS-free extended version. In another attempt to study the effect of varying the number of nodes while keeping the maximum communication range constant at 14 m, increasing the number of nodes has a similar effect to increasing the maximum communication range. Here, the node density (nodes/m^2) is shown against the success rate in figure 3.7.

3.5 Behavior in simulated network

In this section we study the behavior of our algorithm in a simulated network created using NS-3 simulator [82].

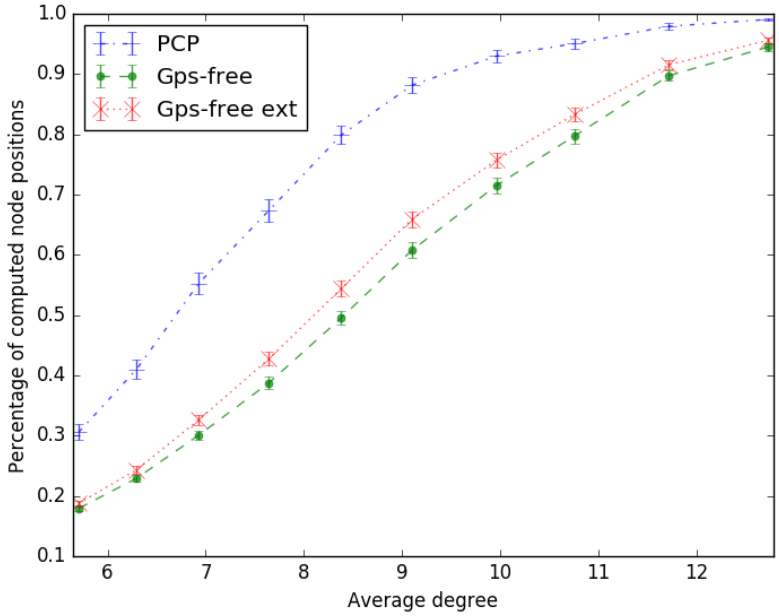


Figure 3.6: Varying Max Communication Range

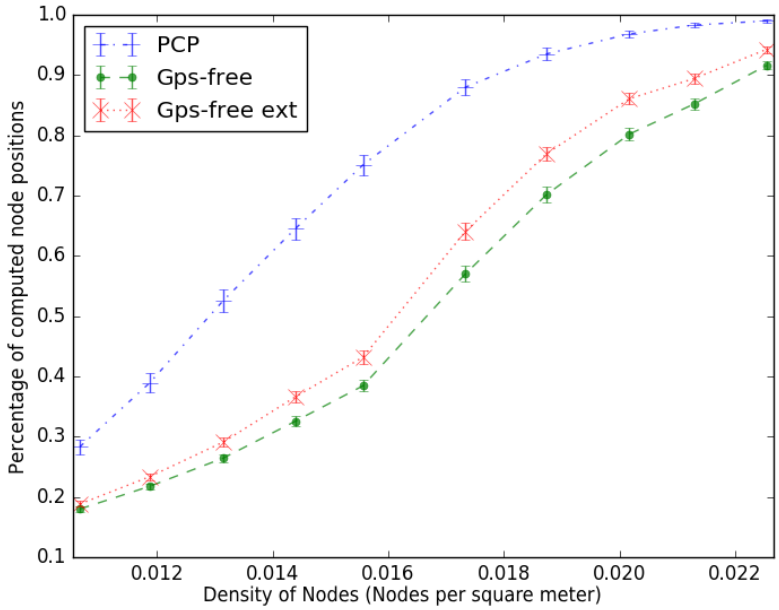


Figure 3.7: Varying Node Density

3.5.1 Distance Noise Effect

The network is composed of 49 nodes spread out on a 7×7 grid. The horizontal and vertical distances between the nodes are 10 m. The anchor nodes are chosen to be in the middle of the networks as depicted in Figure 3.8. The maximum range of communication is set to 15 m using the *RangePropagationLossModel* class. Communication between nodes is through wifi 802 11.b standard with a data rate

equal to 1 Mbps. It is to be pointed out that WiFi is not the only standard to be used for inter-node communication, it is provided here as an example. Generally speaking any communication technology that allows broadcasting can be used. It is to be noted that performing a broadcast is far easier than point-to-point communication since the latter might need to store and update routing tables. Thus, this simplicity of just needing broadcasts is an advantage of our solution.

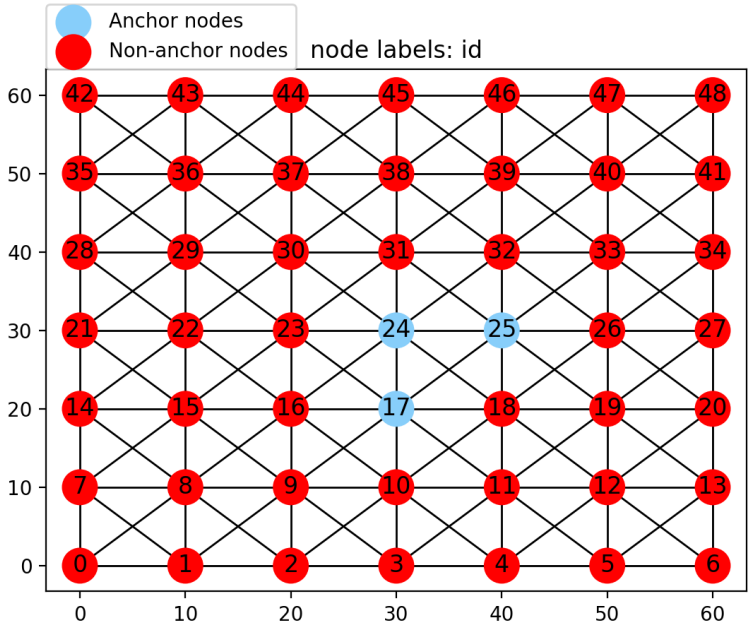


Figure 3.8: Grid network

The time interval between broadcasts is 1 second. The anchor nodes start broadcasting their own position at slightly different instants to avoid packet loss due to multiple transmissions in the same instant. Once a non-anchor node estimates its position it starts its own broadcast cycles with the 1 second interval. This means that after the three anchor nodes start broadcasting their position, the first node to be localized is node 18 which starts broadcasting its own position so that further nodes are localized. If there are no measurement errors, all the nodes are correctly localized within the first second by the successive localization and broadcasting mechanism. We introduce distance measurement errors between nodes by adding a Gaussian noise with zero mean and different standard deviation values for each experiment. The average position error for all nodes in the network is computed over a period of time that includes 5 broadcasts for all network nodes. The position error for a node is the difference in the distance between the estimated and the actual position. The results are shown in the left-hand graph of Figure 3.9. Since not all the nodes know their position at the same instant, the average localization error is shown against the sequence number. The ith sequence number on the x-axis indicates the ith anchor nodes' own position broadcast which is inherited by non-anchor nodes when localized. Even though the nodes are localized at different time instants, during the first second all the nodes update their position and set their sequence number to one. The experiment is repeated 1000 times at each standard deviation value. The distance measurements error and the confidence interval are plotted. For

1 *cm* standard deviation, the average error fluctuates around 3.5 *cm*. For 3 *cm* and 5 *cm*, the average error is around 11 *cm* and 18 *cm* respectively. Figure 3.9 shows the stability of position estimation over a period of time.

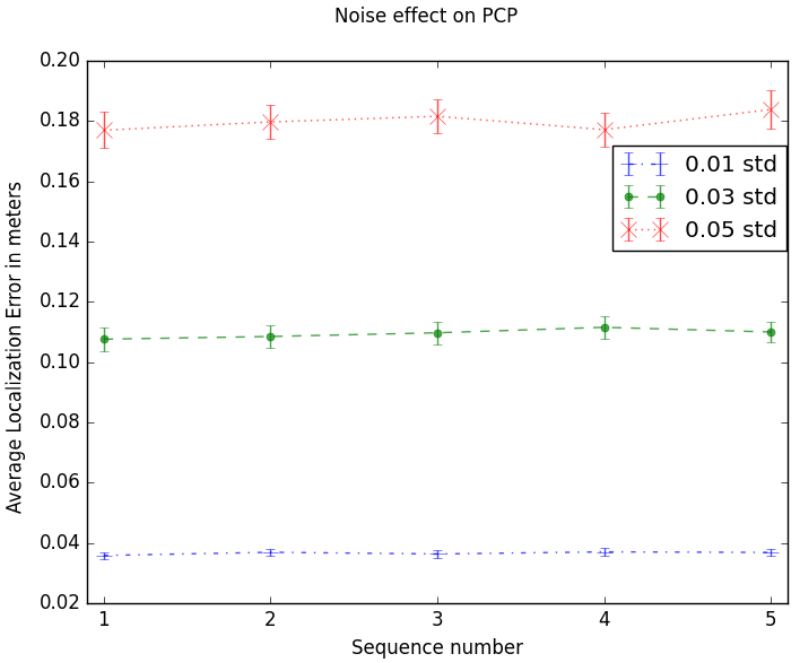


Figure 3.9: Stability of localization error over time

Figure 3.10 shows the average localization error during one time step for a wider range of noise values by varying the standard deviation from 1 *cm* to 15 *cm*. The graph shows a linear relation between noise and localization error.

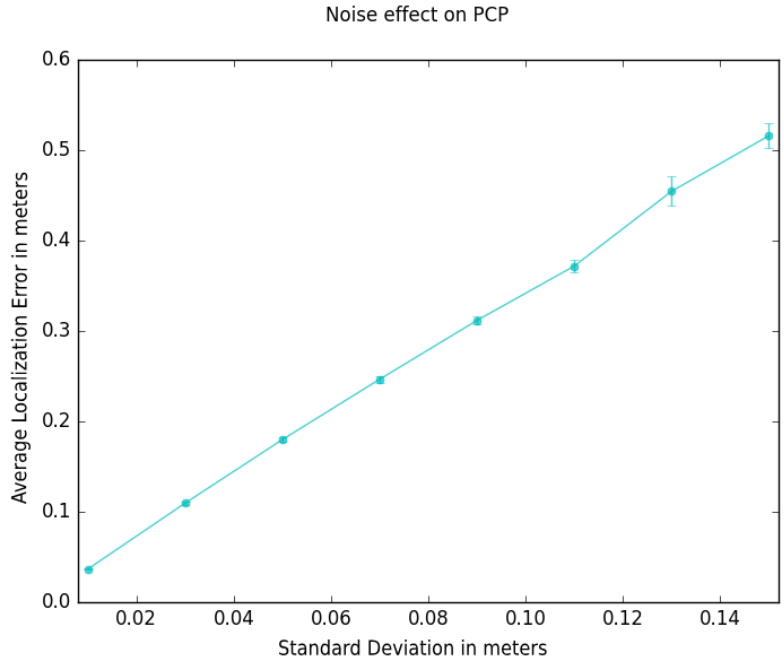


Figure 3.10: Average Localization error with different standard deviation values

3.5.2 Error propagation towards the network's edge

To study the effect of error propagation as nodes towards the edge of the network are localized, we repeat the experiment for different numbers of nodes in the network. The 3 anchor nodes are always chosen to be in the center of the network and the standard deviation for the Gaussian noise is set to 5 *cm*. The results are shown in Figure 3.12. The first setup is a network with four nodes, three of which are anchor nodes. This means that the fourth node is localized directly through anchor nodes, yielding a very small average localization error. The second setup includes 16 nodes, which means that the anchor nodes are surrounded by one layer of non-anchor nodes colored in green as shown in the left part of Figure 3.11.

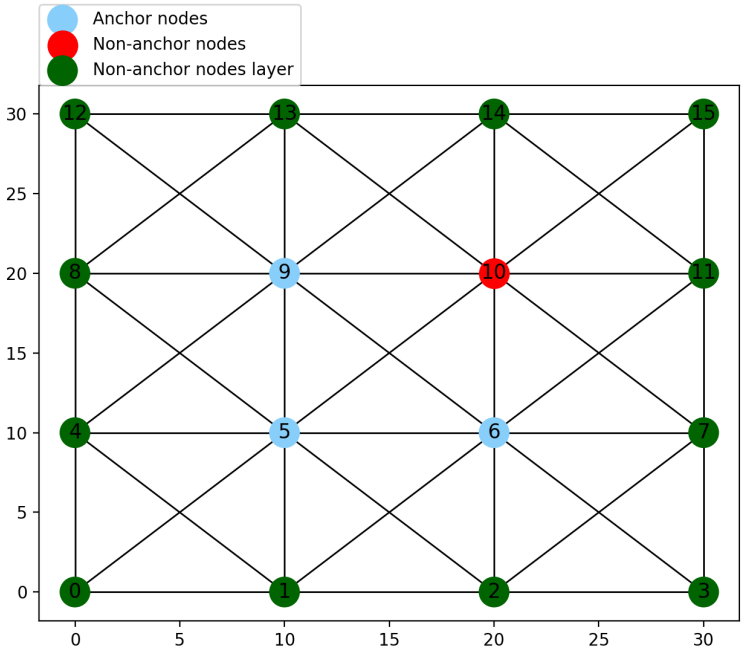


Figure 3.11: 16-node Network Example

Each subsequent setup adds one extra layer of nodes; a 36-node network has two layers of nodes surrounding the anchor nodes etc. As the a further layer from the center is added, the added nodes are expected to have larger errors due to error propagation. This phenomenon is reflected in figure 3.12 with the increase of average localization error with the addition of new layers. A mathematical introduction to uncertainty propagation is presented in Appendix A.

Another aspect to look into is the network traffic consumed to maintain the position knowledge in the network. As previously mentioned, we use SFLS to share position information. The first broadcast is a traditional flooding because the first request to rebroadcast is set to true for all nodes. However, the second broadcast is limited to the 1-hop neighbors as they do not rebroadcast received positions. Figure 3.13 shows the number of broadcasts at each time step also indicated by the sequence number of the transmitted positions. The number of broadcasts is high for the first broadcast as all nodes broadcast the positions of all nodes thus yielding a

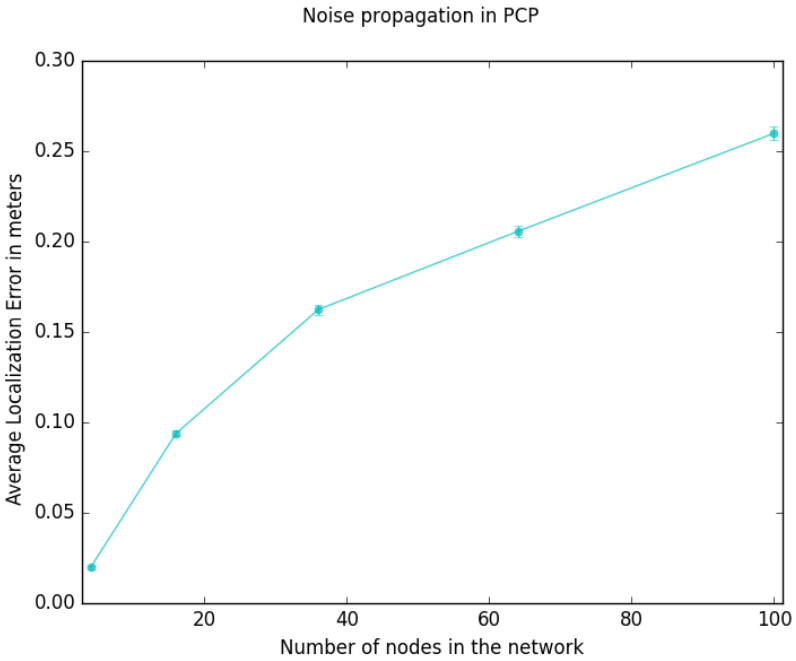


Figure 3.12: Error Propagation

number of broadcasts equal to n^2 where n is the number of nodes in the network. The next broadcast wave includes a broadcast to one-hop neighbors only, then the rebroadcasting is cut off yielding a small number of broadcasts. Nodes alternate allowing and blocking rebroadcasts such that closer nodes receive higher frequency updates while further nodes are updated less frequently. Using traditional flooding for each position update leads to a number of broad casts $\approx n^2$ for all time steps as the number of broad casts at sequence number 1 in figure 3.13. This increase in bandwidth consumption does not yield any improvement to the localization accuracy. Since SFLS [15] does not flood the whole network with position packets for every position update of node i and the position update rate is relatively small compared to channel bandwidth, the collision rate is negligible. Even in cases of losses, since the update frequency is highest for nearby nodes, the position update re-broad-cast will reach nearby nodes sooner than further nodes. For further nodes, in case of nodes' mobility the relative position change is less significant compared to nearby nodes. Consequently, the effect of data loss on further nodes is expected to be less significant. If for any reason, the data loss rate is significantly affecting the solution's accuracy, SFLS can be adjusted to increase the ratio of forwarding received positions. In other words, instead of forwarding once every two received positions (or n times out of p in general), the ratio n/p can be increased to account for losses.

3.6 Conclusion

PCP is a lightweight, bandwidth friendly, and cost and energy efficient location service that is suitable for MANETs. The cost and energy consumption is conserved by requiring a minimum of three GPS-equipped nodes. The computations used to

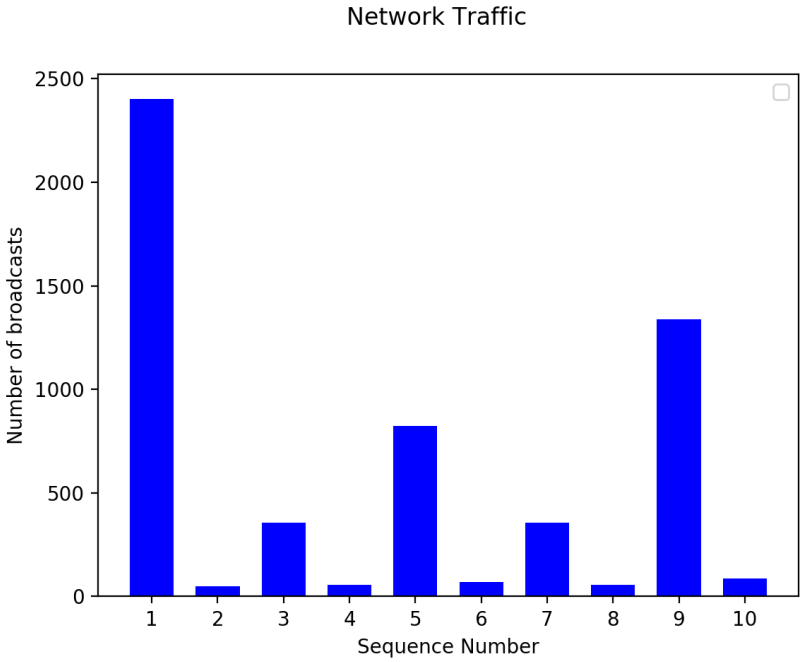


Figure 3.13: Trace of number of broadcasts per time step

localize nodes are not complicated and do not require many cycles as in the case of optimization. Sharing positions with other nodes in the network is done using a semi-flooding method to conserve network bandwidth. The solution requires that the anchor nodes are in the vicinity of at least one non-anchor node in order to be able to launch the algorithm. This starting condition constraint might appear to hinder the generality of the algorithm but this can be mitigated by electing some virtual anchor nodes that satisfy the condition. These virtual anchor nodes are given assumed positions so that they form a triangle from the known distance between them. For instance, let us assume non-collinear nodes i, j, k see each other and at least a fourth node l. Node i is positioned at the origin, node j is on the horizontal axis at a distance equal to $dist(i, j)$ and node k is added to have a positive y-value using triangulation. The algorithm then treats them as anchor nodes and computes the position of the rest of the nodes as previously explained. When at least three real anchor node positions have been computed, all the nodes can then be transferred to the global coordinate system using the computed and actual positions of anchor nodes [19].

The solution is compared to GPS-free [19] and it was shown that our solution is able to localize a higher percentage of nodes. In a network with an average degree ≈ 10, around 90% of the nodes are localized. Furthermore, we tested our algorithm using NS-3 simulator while introducing distance measurement errors and the average error is shown to be stable for a 50-node network. Also, the average localization error increases linearly with the standard deviation of the Gaussian measurement noise. The used position sharing technique (SFLS [15]), conserves bandwidth by reducing position update frequency to further nodes. For future work, the error propagation effect towards the edge of the network needs to be mitigated. Additionally, the impact of nodes' mobility on the accuracy of localization is also to be considered.

Chapter 4

CSI-based Indoor Localization

The advent of 5G is leading to higher demand in transmission throughput. One of the principal contributors to achieving the demanded rate is Multiple-Input-Multiple-Output (MIMO) antennas. With multiple antennas installed on 5G cells, the transmission rate is vastly augmented [46]. In addition, the transmission at each antenna is further accelerating by transmitting over multiple sub-carriers. This is possible through OFDM over orthogonal sub-carriers in a given bandwidth [83].

This methodology provides rich information to be used in in localization which is the measure of Channel State Information (CSI). In this chapter, we briefly introduce CSI and how it can be used in localization. Then, our work in CSI-based localization is presented along with the position estimation accuracy results when applied to an indoors experiment. The experiment's dataset is generated and published by the organizers of the Communication Theory Workshop's Indoor positioning competition in which our solution secured the first place [1].

4.1 Channel State Information

Channel State Information (CSI) is a measure of the effect of the channel on the transmitted signal. The alteration that occurs to the signal could be due to channel's noise, fading, multi-path, antenna's quality [84], etc. In comparison with RSSI, which has been the dominant measure for localization, CSI is more fine-grained and stable [35]. The fine-grain characteristic is due to the possibility to compute CSI per subcarrier per antenna whereas RSSI is a single measure per transmission. It has been shown through experimentation that CSI exhibits more temporal stability when compared to RSSI.

To formalize the CSI, assume a MIMO antenna with i sub-antennas which is communicating with a device over j subcarriers. Equation 4.1 shows the relation between the transmitted signal $T_{i,j}$ and the received signal $R_{i,j}$ over a wireless channel. The transmitted signal is affected by the channel depicted by the complex number $CSI_{i,j}$ then white noise N is added.

$$R_{i,j} = T_{i,j} \cdot CSI_{i,j} + N \tag{4.1}$$

Since the CSI is a complex number it can be expressed in polar or cartesian forms, as presented in equations (4.2) and (4.3), respectively [84]. This is interesting to note because, at a later, a decision has to be made to choose one or more of the components in the localization process.

$$CSI_{i,j} = |Mag| \angle \phi \tag{4.2}$$

$$CSI_{i,j} = Re + iIm \tag{4.3}$$

$$\begin{aligned} Mag &= \sqrt{Re^2 + Im^2} \\ \phi &= \arctan(Re, Im) \end{aligned} \tag{4.4}$$

In order to concretize the concept, an example of the four components is presented in figure 4.1. The CSI components are computed for 924 subcarriers for one antenna in a 2×8 MIMO antenna. It is to be noted that the phase is scaled to be visible in the figure.

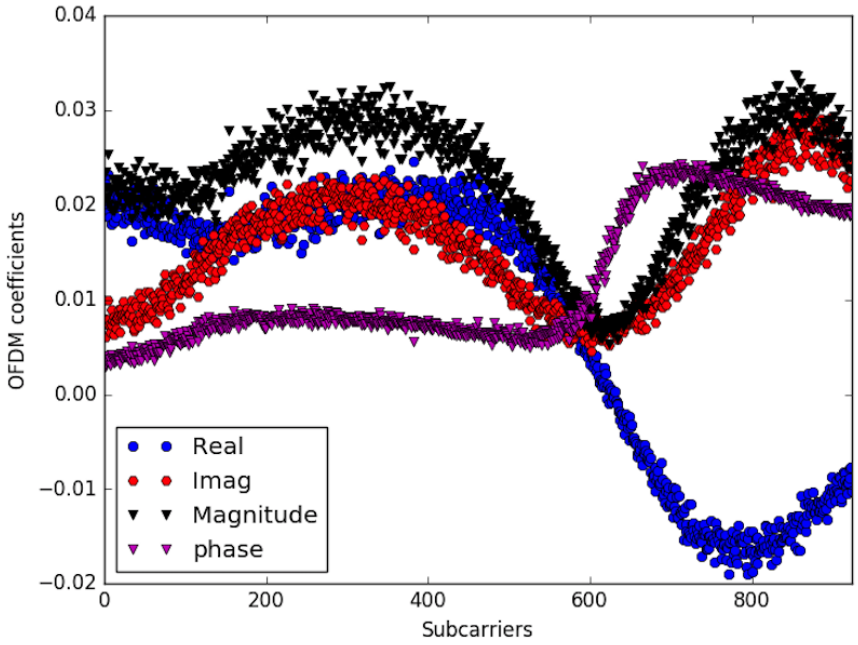
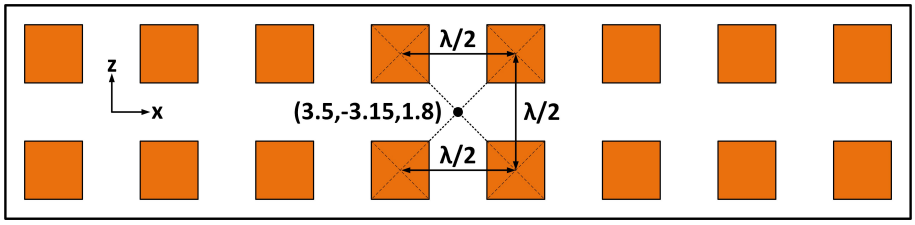


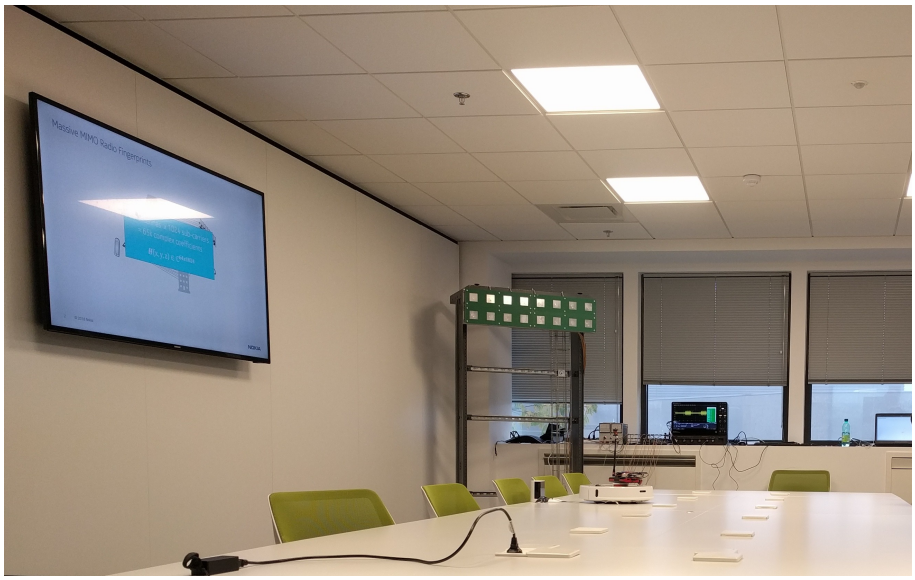
Figure 4.1: Real, Imaginary, Magnitude and Phase components example over 924 subcarriers.

4.2 Experimental Setup

Before presenting our localization solution, the experiment [1] on which our approach was tested is introduced in this section in order to put the reader in context. The experiment is conducted in a conference room where a transmitter traverses a 4 × 2 meter table. The transmitter uses a MIMO channel sounder [46]. Signals are received at a 2 × 8 MIMO antenna array after which the CSI is computed per antenna per subcarrier. Figure 4.2 illustrates the environmental setup where the sub-figure 4.2a shows a sketch of the MIMO antenna seen in the middle of the sub-figure 4.2b. Figure 4.2a demonstrates the geometric coordinates of the MIMO antenna as well as the distances between adjacent antennas which are set to $\lambda/2$ where the carrier frequency is 1.25 GHz.



(a) Antenna sketch [84]



(b) Indoor Environment [46]

Figure 4.2: Experimental setup

The transmission occurs over 1024 sub-carrier from which 10% are guard bands. Thus, 924 subcarriers are effectively used over a 20 MHz bandwidth at each of the 16 sub-antennas. The ground truths positions are computed using a tachymeter with a 1 cm measurement error. For each transmission, the recorded position is matched with the computed CSI constituting a dataset of 17k CSI to position samples. In the subsequent sections, the steps applied to the dataset are explained and the estimation results are plotted.

4.3 Methodology

4.3.1 CSI Components Analysis

The first step in the proposed solution is to analyze the stability of each of the four CSI components with respect to the position of the transmitter. A preliminary step to test the stability of the four components is to plot each of the components for multiple transmissions from the same position. Figure 4.3 shows four plots corresponding to the real, imaginary, magnitude, and phase components. Each plot contained the computed component values from different transmissions that occurred from the same position.

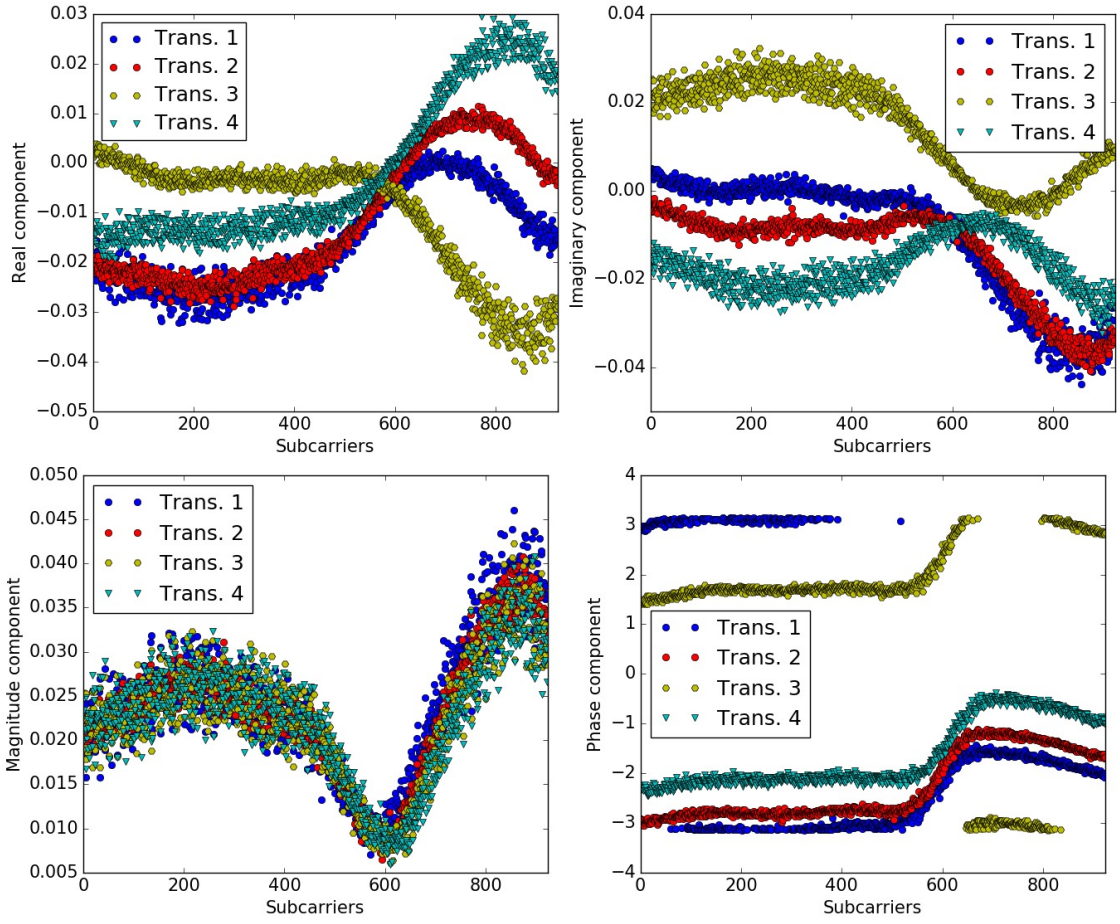


Figure 4.3: Real, Imaginary, Magnitude, and phase components estimated from 4 transmissions at the same position.

The magnitude plot has the four readings almost overlapping which is a strong sign of its stability when compared to the other three components. A more robust test of stability is achieved by computing the correlation coefficient between CSI readings of each component at one position and the CSI readings from the closest position in the dataset. This process is repeated for 1000 sample pairs where the stability of each component corresponds to the average correlation coefficient. The results are demonstrated in figure 4.4.

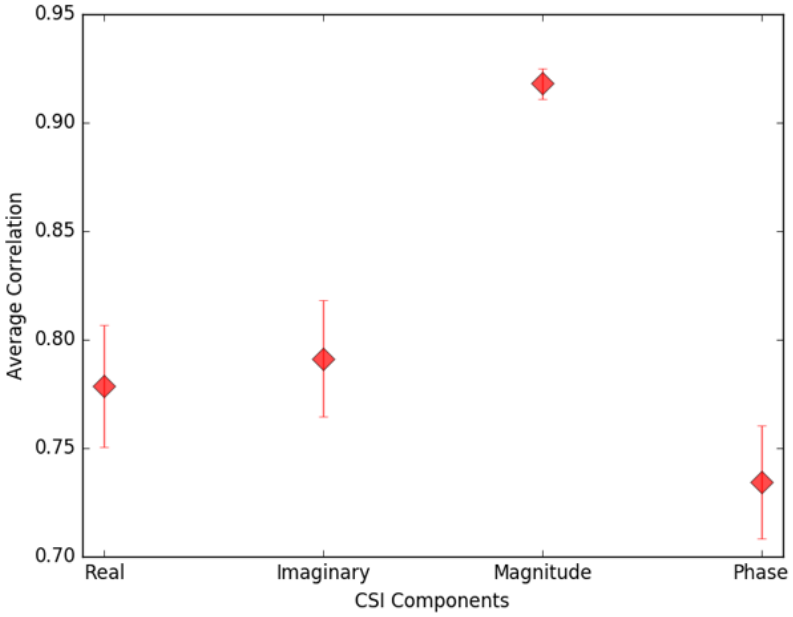


Figure 4.4: Average Correlation for Real, Imaginary, Magnitude, and Phase components.

The statistical analysis shows that the magnitude has the highest average correlation coefficient which supports the result of the preliminary analysis. In conclusion, the magnitude proves to be the most stable when compared to the other components. Consequently, we use the magnitude as the input feature to different learning models. Using any CSI component other than the magnitude or a combination of components deteriorates the estimation results which further supports the use of the magnitude component as the only feature.

4.3.2 NDR: Noise and Dimensionality Reduction

Selecting the magnitude as the input feature leads to a 16×924 input values for each position to be estimated. This corresponds to a reduction of half the number of features when compared to solutions that use a pair of components such as real and imaginary [46]. To put this into perspective, dividing the data set into 90% training set and 10% test set yields $15k \times 16 \times 924 = 222M$ input features for the 15k training samples [84]. Building a complex learning model such as Neural Networks for this large number of features is computationally demanding. We introduce a technique called Noise and Dimensionality Reduction (NDR) to simultaneously reduce the number of features and noise. This method improves the training time as well as the position estimation accuracy.

The reduction of input feature is achieved through polynomial fitting over the magnitude values at each antenna [85]. The degree of the polynomial should not be too low to be able to represent the variations in magnitude values. It also should not be too high to avoid computational complexity. In order to select a convenient polynomial degree, polynomial fitting with different degrees is attempted

with different values: from 3 to 8. The mean fitting error and standard deviation are calculated for all samples. The average fitting error and standard deviation for each degree value are plotted in figure 4.5.

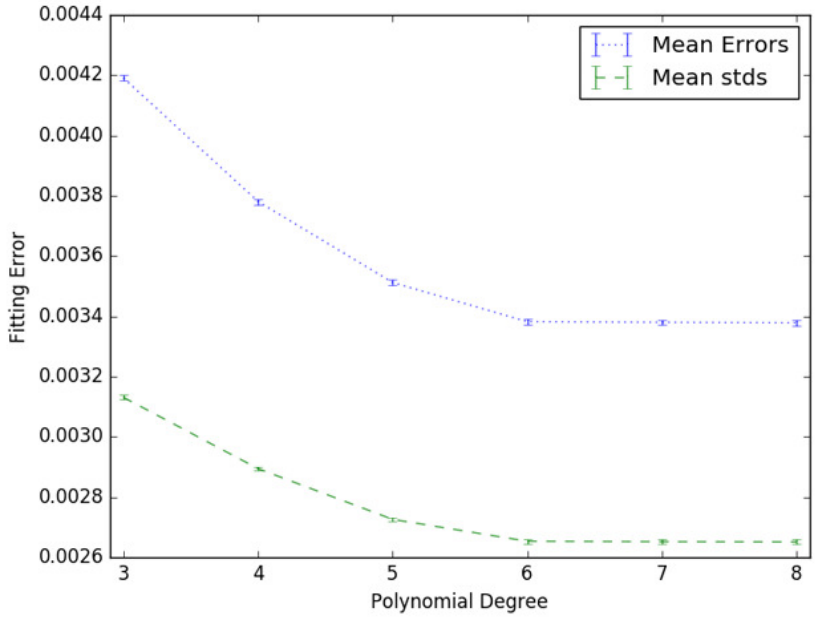
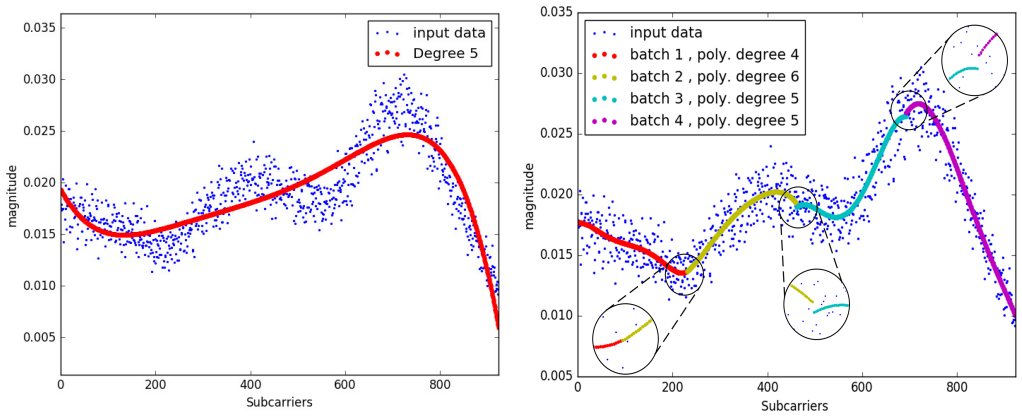


Figure 4.5: Average fitting error for various polynomial degrees.

The fitting error decreases as the polynomial degree increases until it stabilizes at the degree of 6. Hence, a degree of 6 appears to be a reasonable choice. However, in some cases, the fitting process is not able to represent the variation in magnitude values as depicted in figure 4.6a. In order to overcome this limitation, the magnitude values are divided into four adjacent equal chunks. Each chunk is processed solely using various polynomial degrees and the line with the least error is chosen for each division. This improves the fitting process but introduces discontinuities between adjacent divisions as presented in figure 4.6b.

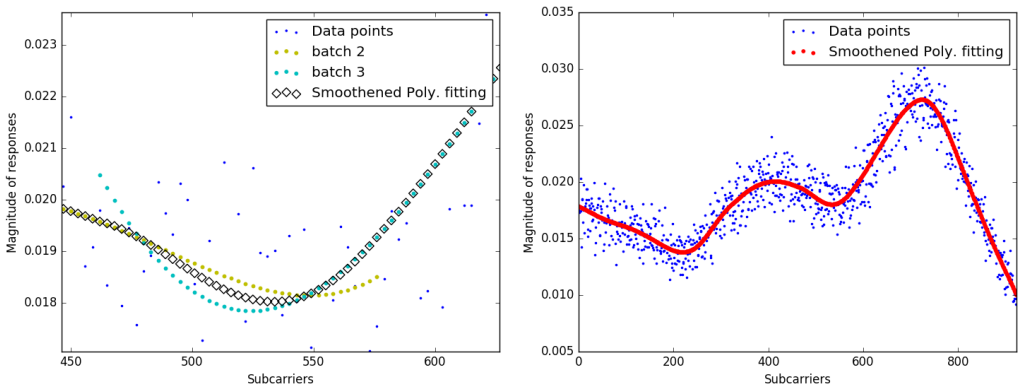


(a) Polynomial regression limitations. (b) Discontinuity between adjacent batches.

Figure 4.6: Underfitting example and dividing subcarrier space.

The introduced discontinuities are then suppressed by extending each of the

batches so that they intersect in a region. Take for example; batches 2 and 3 in figure 4.6b, when each batch is extended, both batches will have points for the same subcarriers in the intersection region. The final magnitude values for the subcarriers in the intersection region are computed using a weighted linear combination of the values from both batches. At the left most subcarrier, a weight of one is given to the value from batch 2 (left batch) and a weight of zero is given to the value from batch 3 (right batch). The following values moving to the right of the intersection region are calculated by linearly decreasing the weight given to batch 2 until it reaches zero at the right most point. The exact opposite occurs with the weight of batch 3 as it increases linearly until it reaches one at the right most point. At any point the sum of both weights is equal to one. This process mitigates the discontinuities and yields an accurate representation of the raw CSI data as shown in figure 4.7b.



(a) Intersection between 2 adjacent batches. (b) Result after discontinuity removal.

Figure 4.7: Discontinuity Removal

The fitted line is used to reduce the number of features by choosing a subset of the magnitude values that are equidistant instead of using all 924 magnitude values. The number of chosen magnitude values should not be too low to maintain the stability and accuracy of results. Nonetheless, it should not be too high to take advantage in terms of the complexity of the learning process. The values are tweaked and empirically chosen based on the performance of different learning models. Moreover, this fitting mitigates the noise affecting the magnitude values leading to a considerable gain in estimation accuracy.

4.3.3 Learning Models

In this section, we propose different learning models based on deep learning [84, 86] and K-nearest neighbors methods [87] to estimate the transmitter position from the CSI dataset that was presented in section 4.2. Furthermore, we introduce some techniques to improve the learning process. Finally, the proposed methods' results are compared with each other and with the results of the Convolutional Neural Networks solution [46]. All experiments are conducted on a machine using a Linux-based operating system equipped with a 3.8-GHz, 32-GB RAM Intel(R)

Xeon(R) quad core CPU E3-1270 v6. The GPU is a 2-GB RAM NVIDIA Quadro K420.

4.3.3.1 Multi-Layer Perceptron Neural Networks

An MLP is used to train on 90% of the dataset. The hyper-parameters are varied in order to reach a combination of hyperparameters that yields a high position estimation accuracy on the 10% test set. The input to the MLP is a 16×66 magnitude values where the first dimension refers to the number of subantennas and the second dimension refers to the number of equidistant magnitude points along the fitted line. The output of the learning model is a 3×1 position vector representing the 3D position of the transmitter. Table 4.1 shows the tweaked hyperparameters and the final chosen values based on the estimation accuracy criteria.

Table 4.1: Hyperparameters selection.

Hyperparameter	**Tested values**	**Best found**
Number of Layers	[4, 5, 6, 7, 8]	7
Units per Layer	[128, 256, 512, 1024, 1200]	1024
Epochs	[50, 100, 150, 200]	150
Activation Functions	[relu, selu, tanh, softmax]	relu
Learning Rate	$[25 \times 10^{-5}, 5 \times 10^{-4}, 1 \times 10^{-3}]$	5×10^{-4}
Optimizers	[Adam, SGD, AdaDelta]	Adam
L2 Regularization	[without, 1×10^{-4}, 1×10^{-5}, 1×10^{-6}]	without L2
Dropout Percentage	[1%, 2%, ..., 10%]	3%
batch sizes	[128, 256, 512, 1024, 2096], [32, 64, 128, 512, 1024], [512, 1024, 2048, 4094, 8188]	[128, 256, 512, 1024, 2096]

A 10-fold cross validation is used to evaluate the performance of the MLP constructed with the best found hyperparameters depicted in table 4.1. The time consumed for training, polynomial fitting per antenna, and inference are 1:10 hrs., 22 ms, and 0.1 ms, respectively. The polynomial fitting step can be accelerated to 8 ms by fixing a polynomial degree at each subset of magnitude values instead of attempting several degrees. The evaluation of estimation accuracy is achieved by computing the mean square error (MSE) for the 1.7k test samples. The standard deviation is also computed. The mean error over the 10-cross validation runs and the mean standard deviation are 0.0445 m and 0.137 m, respectively. The large standard deviation can be explained by the outliers where the model fails to relate the test sample to the training sample and estimates a position very far from the

ground truths. The error distribution of the test set positions for one of the runs is shown in figure 4.8.

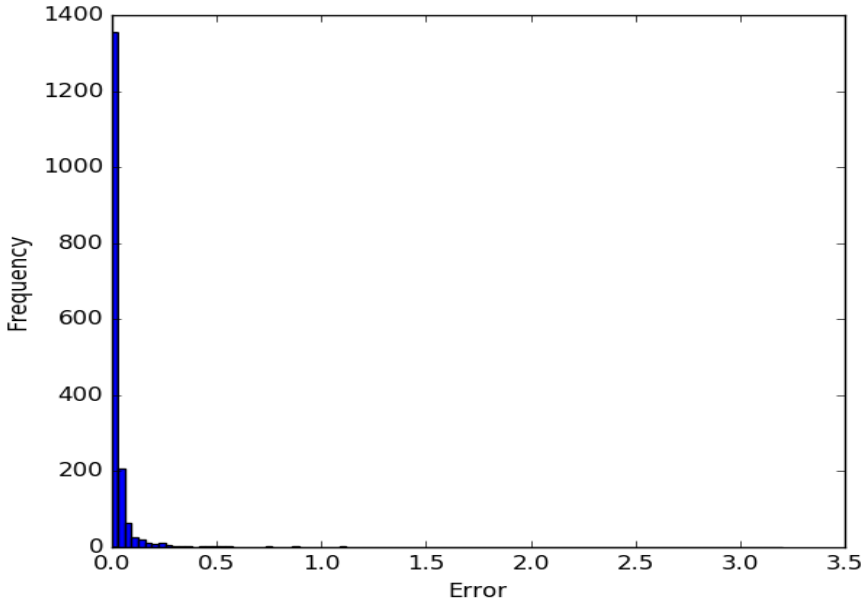
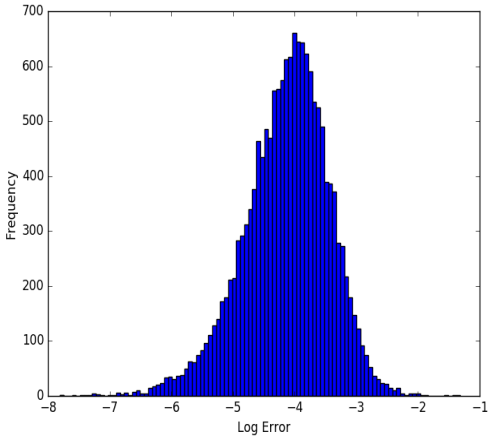
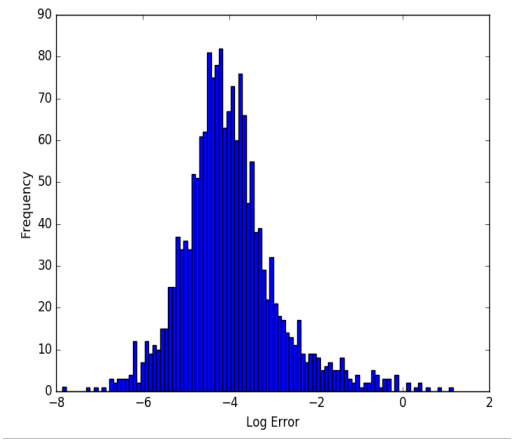


Figure 4.8: Test set Error distribution.

It is clearly noticeable that most estimation errors are close to 0 whereas very few estimations are large. These errors induce a right skew to the distribution. Thus, the arithmetic mean and standard deviation misrepresent the error distribution. In order to better study the distribution of errors, the log of errors is plotted in figure 4.9 when the learnt model is applied to both training and test sets, respectively. The first remark is that both plots follow a log-normal distribution. Consequently, the median of errors or the mean of the logged errors are better representatives of the estimation accuracy. The second remark is that the distribution of the training set error is smoother than that of the test set, which hints that the model is adapted to the samples used for learning from as expected.



(a) Train set Log Error distribution.



(b) Test set Log Error distribution.

Figure 4.9: Log of errors plots.

Since the training time is relatively long, it is possible to reduce the training time for a relatively smaller loss in accuracy. The training time can be reduced by using a simpler MLP. We repeat the 10-fold cross validation with 100 epochs, 5 layers, and 512 units per layer. The resulting training and inference times are 0.0659 m and 0.14 m, respectively.

Another method for evaluating the performance is to vary the number of antennas used in the localization process. The mean of log of errors is plotted against the number of antennas in figure 4.10 along with the 95th percentile error bar to better represent the estimation accuracy.

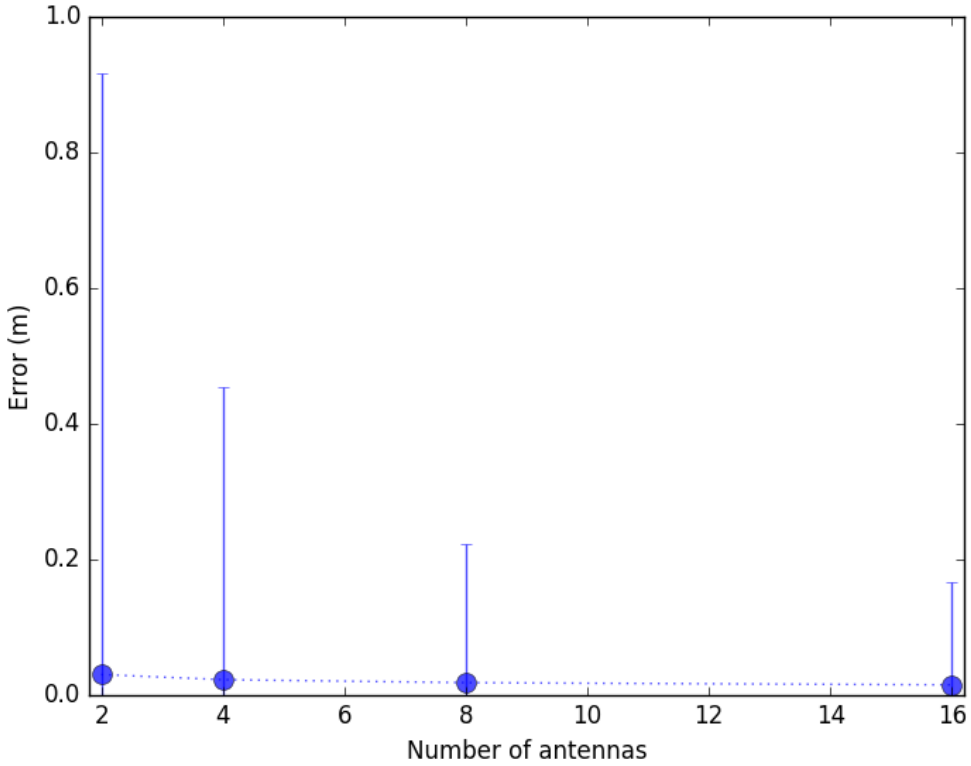


Figure 4.10: Log of MSE showing the 95th error bars.

The right skewed nature of the distribution is reflected in the largely skewed error bars. Due to the large error bars, the mean values of the MSE are not clearly distinguishable. To avoid this ambiguity, the values of MSE when using 2, 4, 8, and 16 antennas are 0.03m, 0.023 m, 0.019 m, and 0.015 m, respectively. As expected, when using only two antennas the error is highest because less data are available for training. When more antennas are used, the position estimation accuracy is improved. The cumulative distribution functions when using 2, 4, 8, and 16 antennas are shown in figure 4.11 where the x-axis representing the error values is logarithmically scaled.

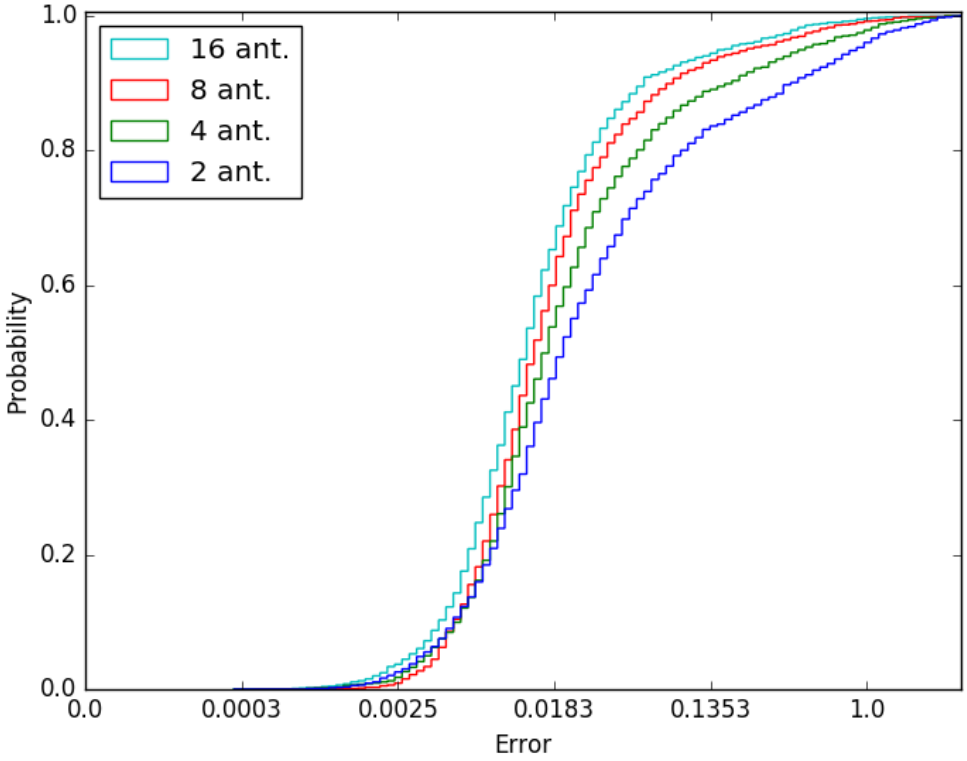


Figure 4.11: Error Cumulative distribution.

4.3.3.2 Data Augmentation and Ensemble NNs

In this section, we examine several techniques to improve the deep learning approach. A data augmentation technique is presented to increase the number of training examples. In addition, the ensemble neural networks method is used to combine the estimations from different MLPs which leads to a significant improvement in position estimation. Furthermore, we conducted an experiment to examine the factor affecting the position estimation accuracy in the CSI. The attempt was to use the difference between adjacent CSI values instead of the absolute values as the input for the learning model. Using the MLP with the best found hyperparameters in table 4.1, a 10-fold cross validation resulted in a mean square error of 4.2 cm which is around 3 mm lower than the estimation error when the magnitude values are used. This is arguably a slight improvement to the performance, which might be considered insignificant. However, it leads to an interesting conclusion that the change of magnitude values from one subcarrier to the next is sufficient to estimate the position rather than the absolute values.

The first technique to improve performance is data augmentation. It is based on the fact that Neural Networks in general are sensitive to the number of training examples. Their performance tends to improve when more training examples are utilized for the training process. Thus, data augmentation technique is applied in several applications to create new training examples from available ones. This

method is especially used in image recognition applications where available images are transformed through rotation, blurring, or zooming to create new examples. Instead of $n \times m$ pixel values for an image, in our case there are 924×16 magnitude vales for each position. The augmented data is generated by altering both the input magnitude values and the output position. The output position is given in the sample through a tachymeter with a 1 cm error as stated in [1]. We model this error by a Gaussian distribution with a zero mean and a 1/3 cm standard deviation. The position of the augmented sample is computed by adding the Gaussian noise to the position of the original sample. The CSI per antenna is augmented by first fitting a line as described in section 4.3.2. The standard deviation α of the error between the fitted line and the actual magnitude points is used to create the augmented sample input. Each of the magnitude values of the augmented sample input is computed by adding a random number from a gaussian distribution with a zero mean and a 2α standard deviation to the point on the fitted line. It can be considered equivalent to blurring an image in the image recognition context. Figure 4.12 shows an example of a data sample and the corresponding augmented data.

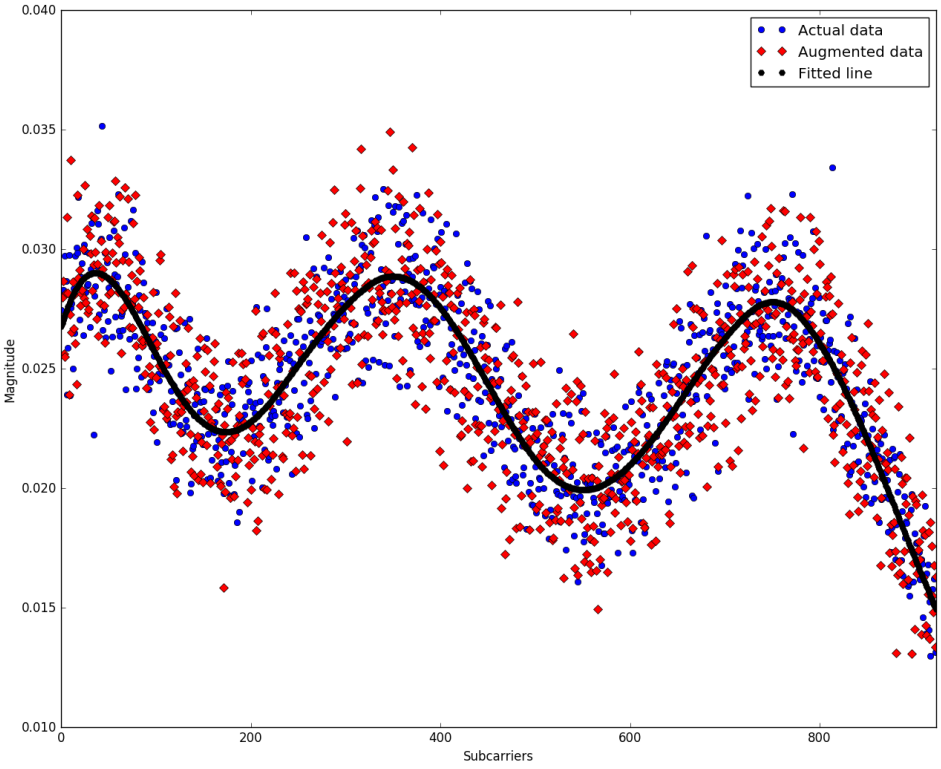


Figure 4.12: Data augmentation sample from an original training sample.

In order to test the effect of the number of augmented data on the performance of the MLP, an MLP with the hyperparameters detailed in table 4.2 is evaluated with percentages of augmented data added to the training set. The results are shown in figure 4.13. With the original training samples only, the mean square error of position estimation is approximately 8 cm. Increasing the percentage of the training samples used to create augmented data leads to a decrease of the estimation error

until it reaches 6.7 cm when 100% of the training samples are used.

Table 4.2: Hyperparameters selection.

Hyperparameter	**Value**
Number of Layers	5
Units per Layer	512
Epochs	100
Activation Function	relu
Learning Rate	0.0005
Optimizer	Adam
L2 Regularization	without L2
Dropout Percentage	0%
Batch Sizes	[128, 256, 512, 1024, 2096]

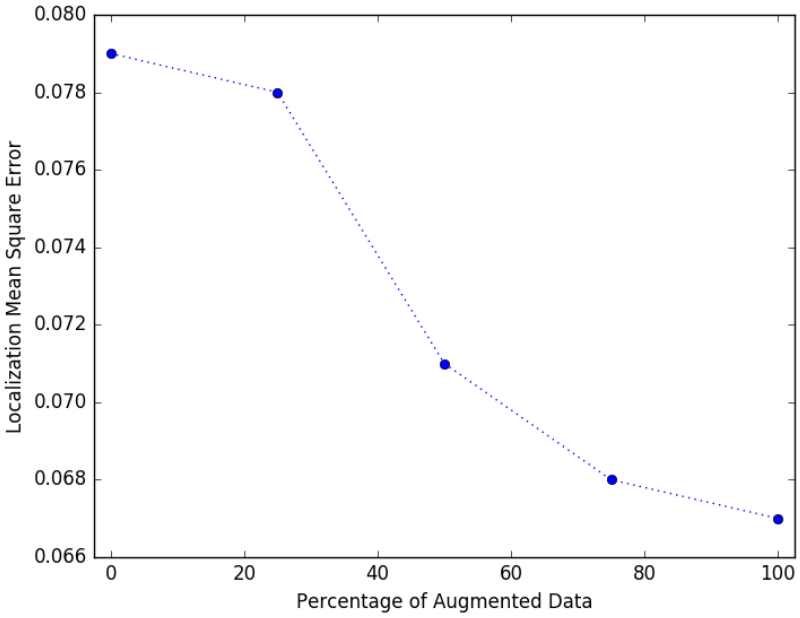


Figure 4.13: Effect of data augmentation on localization accuracy.

Another method to improve the performance of neural networks is to construct several neural networks with different characteristics and deduce the estimation from the individual estimations. This method is known as ensemble neural networks. In this work, different neural networks are constructed using different combinations of hyperparameters, e.g. the neural networks constructed with parameters from tables 4.1 and 4.2. Also, using different training sets results in different NNs like the NNs used to plot each point in figure 4.13 or the NNs used in each of the 10-fold cross

validation experiments. The outcome of each of the neural networks contributes to the final estimation of the ensemble. We examined different methods to process the multiple MLP estimations [86].

1. Mean: The simplest way to mix the results is to compute the arithmetic mean position of all the predictions.
2. Weighted mean: Each of the MLPs is given a weight that is proportional to its individual localization accuracy. Thus, the higher the accuracy, the higher the weight. Then the final prediction is a weighted average of the individual predictions.
3. Weighted power mean: The effect of weights is further magnified by raising them to a certain power before computing the weighted average.
4. Median: The idea is to pick one of the predictions that is closest to all the other predictions. This makes sense when the ensemble has three or more MLPs. This mitigates the effect of the large errors of some predictions.
5. Random: The final prediction is a randomly selected individual prediction.
6. Best pick: This is used as an indication of the best possible result one can attain with the given ensemble. The final prediction is the closest individual prediction to the actual position. This is not feasible since in normal cases the actual position is not given.

The performance of the presented methods is evaluated in figure 4.14 as the number of MLPs in the ensemble increases. The x-axis depicts the number of MLPs used in the ensemble; the number between brackets shows the MSE in cm of the added MLP if used alone. This means that the first MLP has an MSE of 3.9 cm which is the highest accuracy achieved when augmented data is used. The MLPs are added in ascending order of their MSE where the first MLP has the lowest MSE and the last has the highest MSE. In addition, two methods can be combined by averaging the outcome of both methods as in "median + wght" where the outcome of the median and weight methods are averaged. With an exception of the random pick method, adding an MLP to the ensemble generally improves the estimation of the ensemble. Even though the added MLP has an individual MSE higher than the previously added MLP, its contribution still improves the ensemble estimation. The best result is achieved when 11 MLPs are used with the combination of the median and weighted mean methods. It is to be noted that the estimation accuracy begins to stabilize when four or more MLPs are used in the ensemble. The best pick method indicates a potential room for improvement if it is possible to determine which of the individual MLPs estimation is the closest to the real estimation.

4.3.3.3 K-Nearest Neighbors

In this section, the classical learning method "K-nearest neighbors" is used for the position estimation process. In this method, the whole training set is traversed

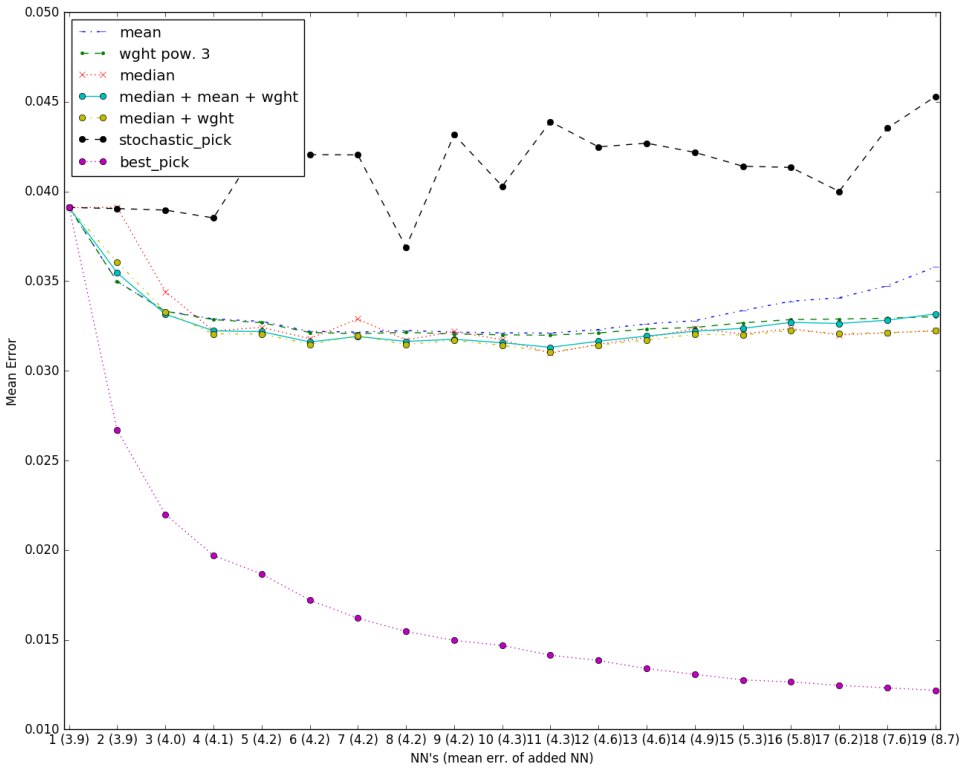


Figure 4.14: Mixing the predictions of a neural network ensemble.

for each CSI sample in the test set. Thus, a simplification of the computational complexity is attempted. For instance, instead of dividing the CSI into four divisions and attempting the line fitting as presented in section 4.3.2, the fitting is achieved using the least square optimization [88] with a fixed degree of 6 which was found to achieve a good balance between accuracy and complexity as shown in figure 4.5. This leads to a loss in accuracy when compared to the previously explained method. However, the lost fitting accuracy does not affect the performance of the K-nearest neighbors method. Also, 33 equidistant points are used instead of 66 as it was found that the stability of the estimation can be maintained with this reduced number of magnitude points.

In order to build the learning model, two criteria have to be decided: the neighboring criterion and the k value. The neighboring criterion must be computationally light and also be able to capture meaningfuls difference between CSI values. To this end, three neighboring criteria are examined assuming that M^1 and M^2 are two sets of 33×16 magnitude values for all 16 antennas.

1. The *Absolute Difference* between corresponding magnitude values, which is averaged over all magnitude values for all antennas.

$$\left|diff_{M^1,M^2}\right| = \frac{1}{16}\sum_{a=1}^{16}\frac{1}{33}\sum_{n=1}^{33}\left|M^1_{a,n} - M^2_{a,n}\right| \tag{4.5}$$

2. The *Euclidean Distance* between two sets of 33 magnitude values averaged for all antennas.

$$dist_{M^1,M^2} = \frac{1}{16}\sum_{a=1}^{16}\sqrt{\sum_{n=1}^{33}\left(M^1_{a,n} - M^2_{a,n}\right)^2} \tag{4.6}$$

3. The *Correlation Coefficient* between two sets of 33 magnitude values is averaged over all antennas.

$$Corr_{M^1,M^2} = \frac{1}{16}\sum_{a=1,n=1}^{16,33}\frac{Cov\left(M^1_{a,n}, M^2_{a,n}\right)}{\alpha_{M^1_{a,n}} \times \alpha_{M^2_{a,n}}} \tag{4.7}$$

The neighboring criterion is evaluated based on the achieved MSE as well as the computational time needed to traverse the training set. For simplicity, the k value is set to one. This means that for each test sample, the training set sample that has the smallest difference to the test sample is selected and the corresponding position is chosen as the estimated position. The test is repeated using different numbers of antennas. Figure 4.15 shows the MSE using each of the three presented criteria. Both the Euclidean distance and absolute difference have lower MSE and are quicker to compute when compared to the correlation coefficient criterion. The MSE is very close between Euclidean distance and absolute difference criteria, especially when using the 16 antennas. The Euclidean distance achieves a 2.4 cm and is fast to compute. The absolute difference achieves a 2.5 cm. Consequently, the Euclidean distance is selected as the neighboring criterion.

After choosing the neighboring criterion, the k value which represents the number of neighbors used to estimate the position has to be set. When k value is larger than one, the positions of the k neighbors are averaged to compute the estimated position. Different k values are tested and the MSE is plotted on figure 4.16. The MSE appears to increase along with the increase of the value of k. This can be explained by the nature of CSI that exhibits abrupt changes from one position to another that are not very far [11]. Thus, as more neighbors are added, the euclidean distance between CSI values is larger and does not reflect the position closeness leading to the deterioration of position estimation. Based on this result, the k value is set to one for the k-nearest neighbor learning model.

With the Euclidean distance chosen to be the neighboring criteria and the k value equals one, the 1.7k test samples are used to evaluate the learning model. The data preprocessing step of line fitting using least square optimization takes approximately 2.6 ms per antenna per position. The traversal of the 17k training samples to deduce one position takes around 1.1 s. The K-nearest neighbor learning model achieves the lowest MSE (2.4 cm) when using all the 16 antennas. Figure 4.17 relates the MSE of the test sample to the Euclidean distance between the CSI of the

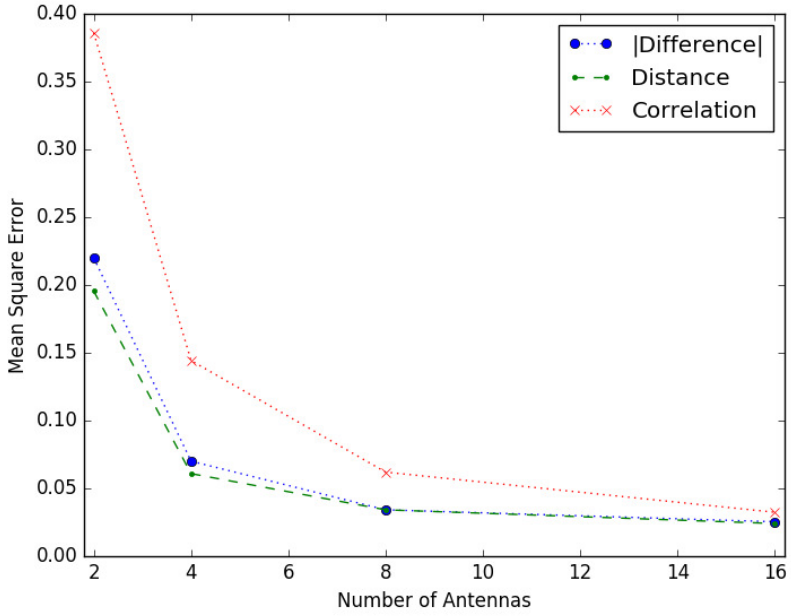


Figure 4.15: Mean Square Error using different closeness criteria.

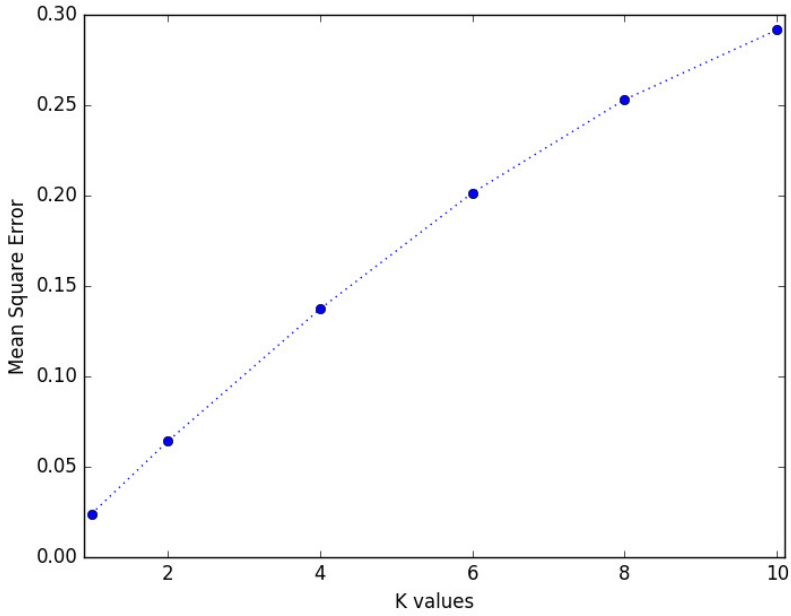


Figure 4.16: Mean Square Error using different k values.

test and training samples. This is an interesting relation because it can be used to predict outliers based on the value of the Euclidean distance. The x-axis shows the Euclidean distance difference between the training and the test samples where the y-axis shows the corresponding MSE. The right sub-figure shows the distribution of errors of the test samples where most estimation errors are very close to zero and few outliers have large errors. It is to be noted that the frequency is log-scaled. The upper figure shows the distribution of the Euclidean distance values between CSI of

the test sample and the closest training sample. A weak relation is depicted in the plot that indicates the increase in estimation error when the Euclidean distance is relatively high. However, this relation is difficult to define deterministically because there are some cases where high Euclidean distance difference still yields an accurate position estimation. This phenomenon can be further studied with a dataset where more outlier cases are available to be able to deduce a trend or a common feature causing this result.

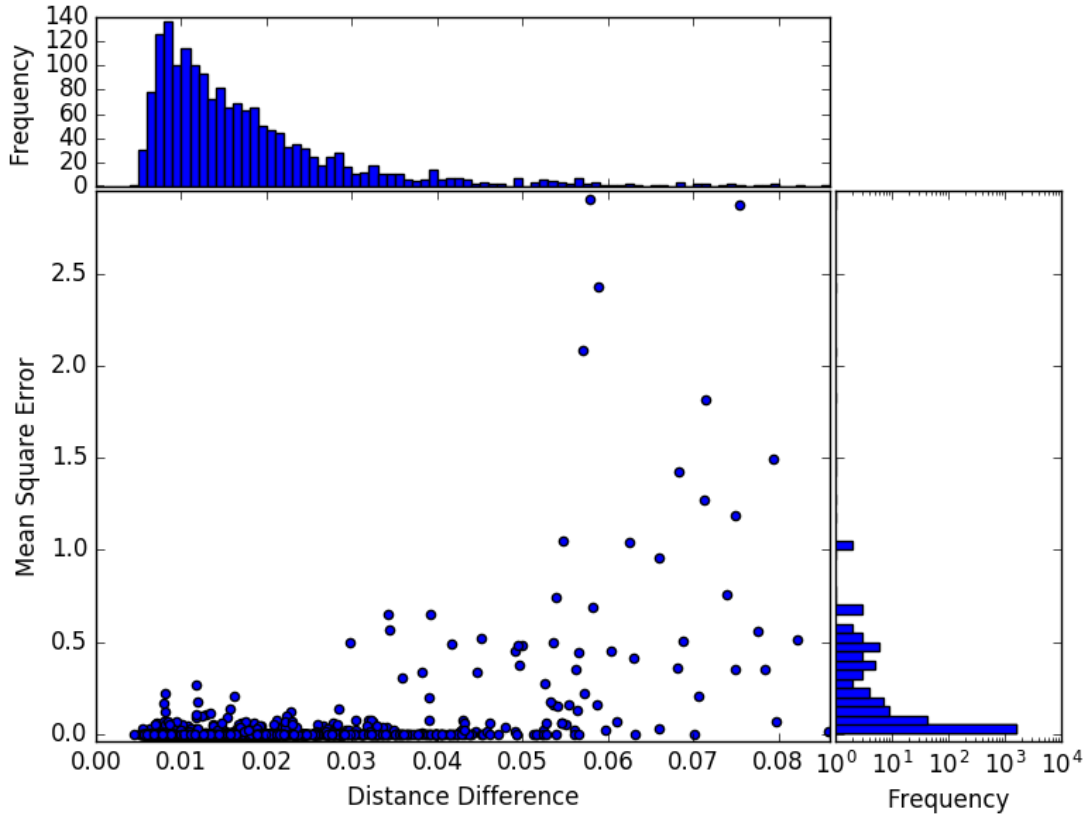


Figure 4.17: Relating the Euclidean distance between the test and closest training sample to the prediction error.

4.4 Conclusion

In this chapter, CSI-based localization in indoor environment has been studied based on the dataset provided in the Communication Theory Workshop's Indoors competition [1] held in Selfoss, Iceland. The objective was to localize a transmitter using CSI computed at a Massive MIMO antenna which is one of the main drivers of the 5G. The evaluation criterion is the Mean Square Error (MSE) of the predicted positions. With an MSE of 2.3 cm, the proposed solution clinched the first place among 8 teams from top universities around the world such as: University of Toronto (Canada), Ruhr University Bochum (Germany), Heriot-Watt University (England), University of Padova (Italy), IMdea networks institute (Spain), Aalborg University (Denmark), and Yuan Ze University (Taiwan).

Several solutions were implemented based on the choice of the magnitude component of CSI as the input feature for several learning models. The magnitude values are preprocessed to simultaneously reduce noise and dimensionality allow for building more complex learning models that are trained in an adequate time. The detailed process of noise and dimentsionality reduction is presented in the paper "NDR: Noise and Dimensionality Reduction of CSI for Indoor Positioning using Deep Learning" [84] as well as the construction of an MLP model to estimate the transmitter position. Furthermore, enhancements to the learning process are achieved through the introduction of data augmentation technique to increase the training and improve the MLP estimation. Instead of using one MLP, several MLPs are created by varying hyperparameters and the training set sizes and combining their individual estimations to further improve estimation accuracy. This work has been presented in the paper "CSI based Indoor localization using Ensemble Neural Networks" [86]. Finally, the K-nearest neighbor technique has been used with some alteration to the preprocessing step to improve efficiency to adapt to the fact that the whole training set has to be traversed to estimate one position. This work has been published in the paper "CSI-MIMO: K-nearest Neighbor applied to Indoor Localization" [87].

In figure 4.18, we show the estimation accuracy of each of the proposed methods as well as a state-of-the-art method. The compared solutions can be summarized as follows:

1. CNN [46]: Convolutional Neural Network is used with the real and imaginary components as input features.
2. NDR [84]: The magnitude component is reduced using a polynomial regression and used as an input to one MLP.
3. Ensemble [86]: multiple MLPs are constructed with different hyperparameters and training set along with data augmentation.
4. K-nearest [87]: K-nearest neighbor method is based on Euclidean distance as the neighboring criterion.

The CNN solution yields higher error due to the use of real and imaginary components as input features which we have shown to be less stable than the magnitude component. NDR is a deep learning based solution where the preprocessed CSI is used to train an MLP achieving a 4.5 cm MSE. Enhancing NDR with data augmentation and ensemble neural network method improves the estimation to 3.1 cm when all the 16 antennas are used. Finally, the K-nearest neighbor method starts with a higher error when only two antennas are used showing higher sensitivity to the number of the train samples. However, when the training samples from all antennas are used, it achieves the highest accuracy with a 2.4 cm MSE.

This work can be extended by studying the outlier estimations which are very far from the average estimation error. This maybe due to some Non-Line-Of-Sight transmissions that led to erroneous CSI. The outlier phenomenon can be studied

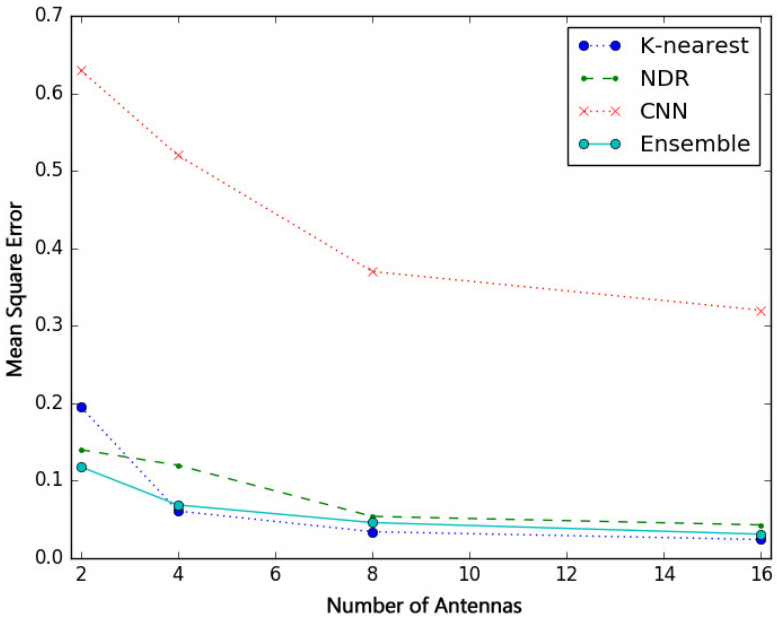


Figure 4.18: Comparison between proposed solution and state-of-the-art method.

when more cases are available which might lead to further improvement of the estimation accuracy.

Chapter 5

Generalization of Deep Learning Localization

5.1 Motivation

In this chapter, we reflect on the results of the indoor positioning experiment discussed in chapter 4. In particular, the different accuracy achieved by the classical learning method (K-nearest neighbor) and the deep learning approach (MLP NN). This analysis is useful in unraveling some mysteries concerning the behaviour of the Deep Neural Networks which are often difficult to explain.

The fact that KNN outperformed Deep MLP NN is rather unexpected and requires further debugging. What makes this result unexpected is that MLP NN is a much more complex model that aims to learn a complex function between input and output. KNN, in comparison, is a simple model. In essence, KNN is a form of lazy learning which attempts to find the closest k training samples in the train set and combines them to predict the test sample output, instead of inferring a general mapping between input and output. Specifically, it can be seen as a table; when given some input, KNN tries to pick the most matching samples in the training table. At first glance, one would expect MLP NN to outperform KNN which is not the case in our problem.

Let us take a step back and consider the chosen criterion to decide which approach outperforms the other in the localization problem. The chosen criterion is the error between estimation and actual positions which is probably the most intuitive choice. We argue that the question that should be asked is how capable the model in predicting the output for input samples beyond its experience with the training data. In other words, how generalizable is the model is?

This question is different from the bias-variance trade off where a highly biased model underfits the training data and yields a high error on both training and test set. A model with high variance overfits the data yielding very low training error with very high test error. This problem does not show in either KNN or MLP NN models as they both achieve low errors on training and test sets. The difference between the bias-variance problem and the generalization problem manifests when the test

set comes from a different distribution than the training set. In this case, will the learning model be able to accurately predict the output or will it fail tremendously?

In this chapter, we expose the generalization capability of both KNN and MLP NN models for the indoor localization problem presented in chapter 4. The analysis would lead to a more solid conclusion about the better model. We then introduce a more challenging outdoor localization problem [2] where generalization is integral to achieve adequate results. We then propose several attempts to achieve this generalization.

5.2 Generalization: MLP NN vs KNN

The generalization capabilities of both MLP NN and KNN are tested by altering the splitting technique of the train and test sets from the available dataset. Commonly, the splitting is achieved by a uniform random selection of samples for the train and test sets. This was the technique used when training the learning models in chapter 4. To test the generality of each model, we maintain the 90 %, 10 % sizes of train and test sets, respectively. However, the test set samples are selected such that no test set position intersects with any of the training set positions. This means that a model that achieves generalization must be able to somehow interpolate or extrapolate from the input CSI and the corresponding positions in the training set to predict sensible estimations for the test set samples. We propose two methods for test set selection, square and sequential selections, to test the extrapolation and interpolation capabilities, respectively.

5.2.1 Square Test set selection

The test set samples in this selection technique are selected from a square in the middle of the table which the transmitter traverses. The square size is adjusted such that the size of the test set is $\approx 10\%$ of the whole data set. The training set and the test set are shown in figure 5.1.

The test sample region makes the prediction task very challenging. The blue square area resembles a blind spot for the learning model which has to be predicted without having any experience (data) in this region. This test set distribution examines the extrapolation capability of the algorithm as most of the test set samples are confined in a region where no training samples exist.

Figure 5.2 shows the predicted sample positions linked with edges to the corresponding ground truths. Figure 5.3 demonstrates the error distribution of the predicted positions.

The KNN's closeness criterion is the Euclidean distance and the value of k is equal to one, as presented in the previous chapter. It can be seen from figure 5.2

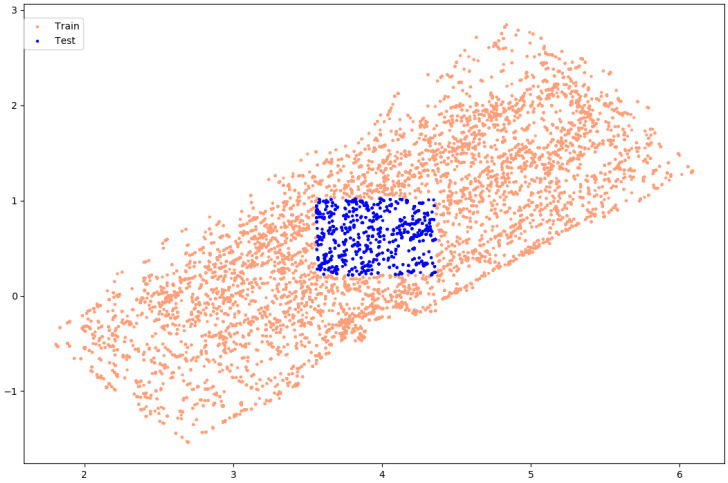


Figure 5.1: Square Test set selection

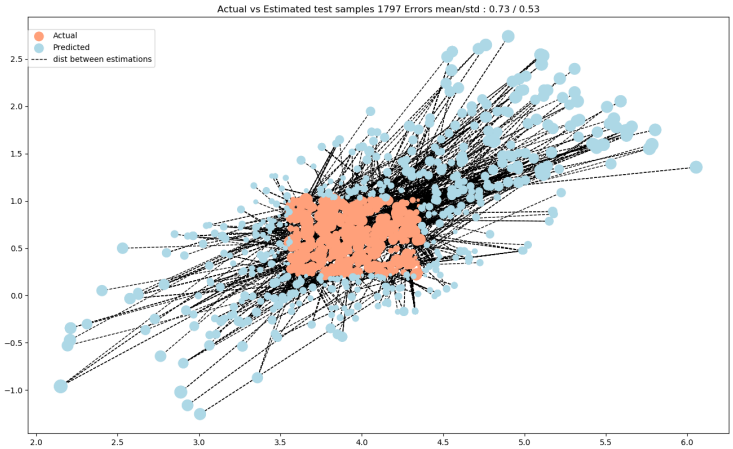


Figure 5.2: KNN predicted positions vs actual positions for square test set

that all predicted positions are outside the test square region. This is expected because the value of k is one; thus, the KNN finds the closest training sample which turns out to be like a random selection of training samples. The error distribution illustrated in Figure 5.3 does not show a particular known distribution. The mean error is 0.73 m and the standard deviation is 0.53 m. This is a complete failure since the error when the test set samples were drawn randomly is 0.023 m.

The MLP NN is built with the hyperparameters from table 4.2. This MLP NN achieved a mean error of 0.065 m with random dataset splitting and trains in a relatively short time. Since the aim of this experiment is to test the generalization ability, there is no need to train a very complex NN. The difference between the error in the random test set and the square test set is enough to conclude the generalization ability. Figure 5.4 displays a map relating the predicted positions to

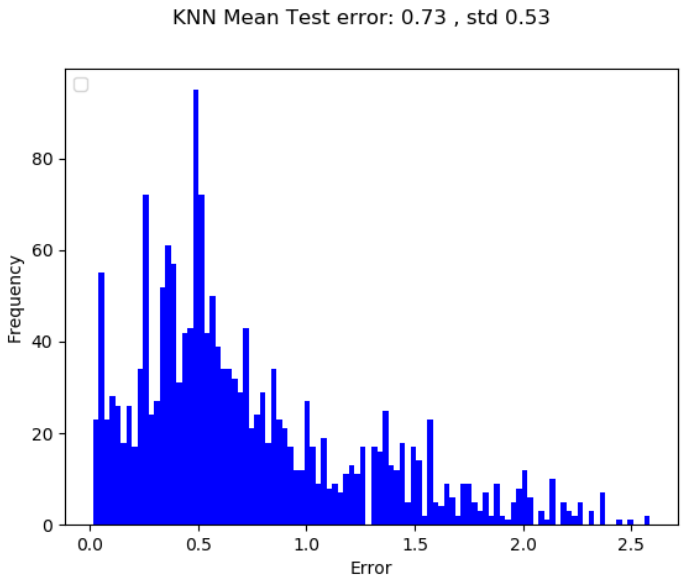


Figure 5.3: KNN error distribution for square test

the ground truths. A clear difference between the estimations of the MLP NN and that of the KNN is that some of these estimations are inside the test set region. This can be explained by the NN's non-linear function that has a continuous output where some input samples results in estimations inside the test set region. The estimated positions still appear to be random. Figure 5.5 shows the error distribution of the MLP NN estimations. While the mean error and standard deviation are less than that of KNN, 0.54 m and 0.33 m respectively, the error is still much larger than that of the random test sample selection. Even though the error distribution seems more structured than that of the KNN, it is difficult to draw a conclusion about a decent extrapolation ability of the model with such large error.

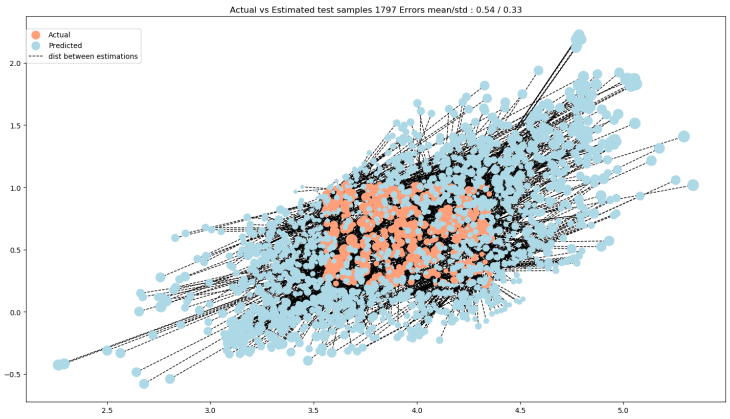


Figure 5.4: MLP NN predicted positions vs actual positions for square test set

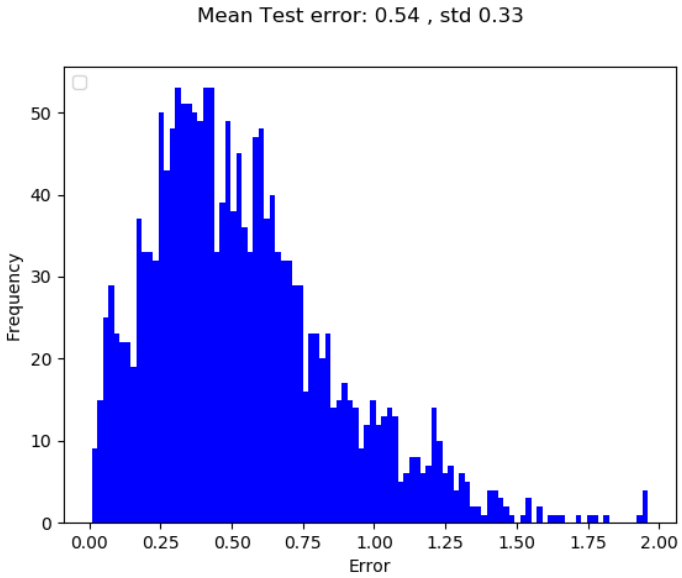


Figure 5.5: MLP NN error distribution for Square test set

5.2.2 Sequential Test set selection

The square test set selection is very challenging for the learning model because the test set region is a relatively large blind spot. The model has to achieve a very challenging extrapolation outside the train set region to make adequate estimations within the square region. In the sequential test set selection, we make use of the order in which the data set is given. As previously mentioned, the data set was created by moving the transmitter along the table using a small vacuum cleaner robot. The data set is provided in the order in which the transmissions were sent while moving along the table. This means that the path of the transmitter can be tracked by traversing the positions in order. Therefore, we set the test set to be the first 10 % of samples read sequentially in order from the provided dataset. This mitigates any intersection between the positions of the train and test sets since moving randomly along the table makes it highly improbable to visit the exact same position twice. Figure 5.6 demonstrates the train set and test set selection using the sequential method.

The test set samples in this selection method are relatively spread along the table. Test set positions are close to but not superposing the train set positions. The prediction task appears easier than the square selection since the model does not need to extrapolate in a blind region. Rather, the model needs to relate test set samples to their nearby training samples then interpolates to estimate the test sample position. We start with the KNN estimations; Figure 5.7 shows the KNN's predictions and the corresponding ground truths of the test samples. KNN predictions do not suffer from being far from the test set region since the training sets are spread along the table. However, it still suffers from very large errors with a mean square error of 0.72 m. The distribution of root mean square errors is shown in figure 5.8.

The mean square error of the KNN model is as large as that of the square test

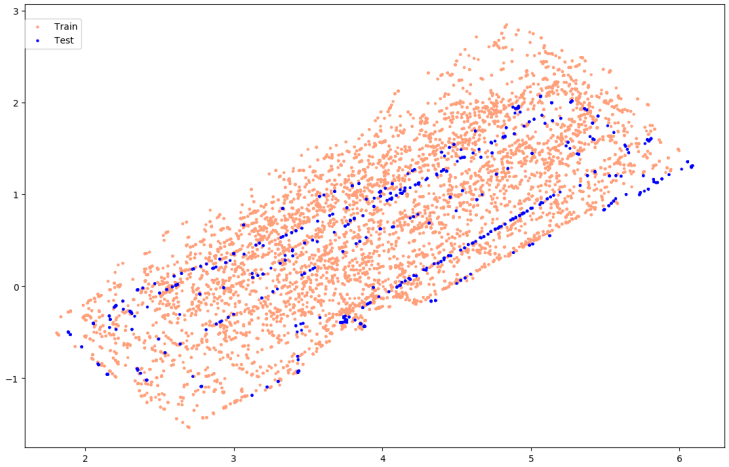


Figure 5.6: Sequential Test set selection

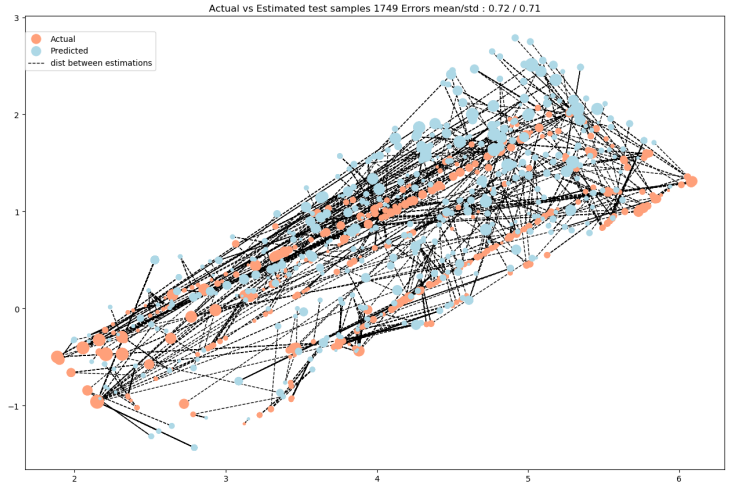


Figure 5.7: KNN predicted positions vs actual positions for sequential test set

selection. Using k values larger than 1 does not improve the performance. The experiment is repeated using MLP NN, figure 5.9 shows the MLP NN predictions and the corresponding ground truths positions. The dispersed predictions show that even the MLP NN does not perform well for the sequential test. This conclusion is backed up by the high mean square error of 0.55 m and the error distribution shown in figure 5.10.

5.2.3 Conclusion

Prior to these experiments, we did not expect the KNN to be able to generalize. However, we expected the MLP NN to show better generalization performance.

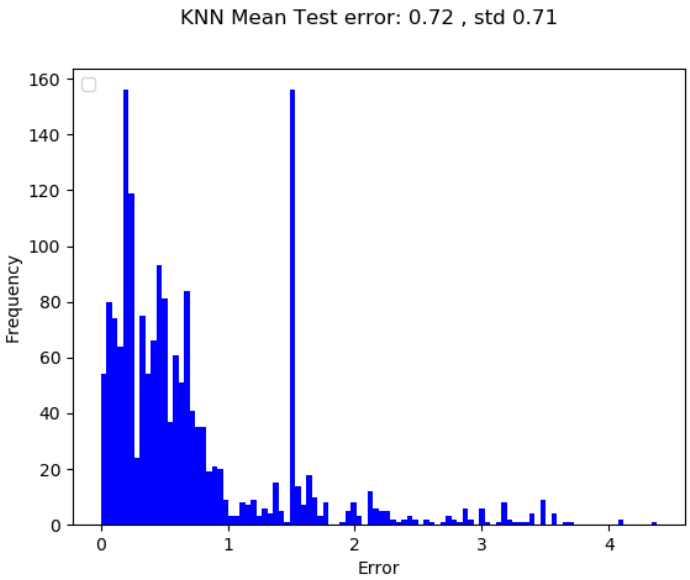


Figure 5.8: KNN error distribution for sequential test

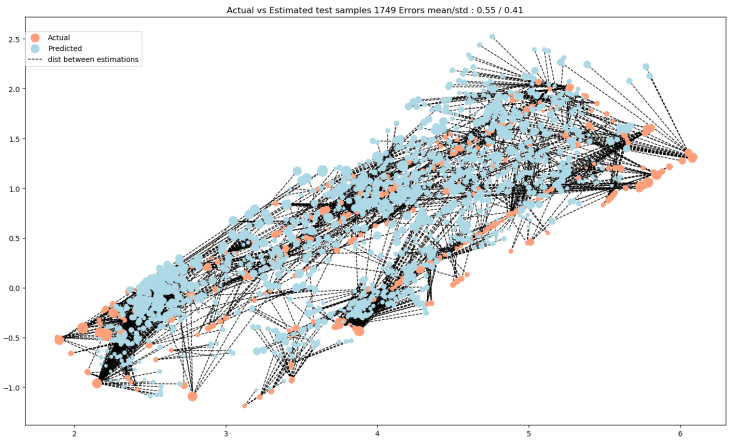


Figure 5.9: MLP NN predicted positions vs actual positions for sequential test set

The experimental results show large errors for both models in both test scenarios. The obvious conclusion is that both models fail to generalize. However, there is a considerable difference between the mean errors of KNN and MLP NN. MLP NN's error is approximately 25 % lower than that of KNN in both experiments. It is not clear whether this difference gives any edge for MLP NN over KNN since their errors are still very large. With the results in hand, we conclude that both KNN and MLP NN fail to generalize beyond the training data even though both achieve very high accuracy when test set is randomly selected.

These experiments explain why KNN outperformed MLP NN when the test set was randomly selected. The sequential test selection experiment shows that both KNN and MLP NN cannot relate the test sample to nearby train samples. However, CSI samples measured from the same position are relatable using both models. This

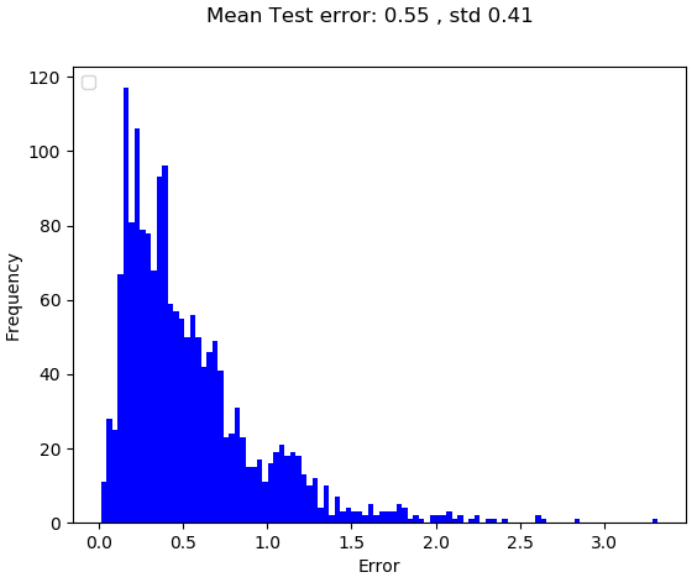


Figure 5.10: MLP NN error distribution for sequential test

is expected because we showed the stability of the magnitude component at the same position in chapter 4, specifically in Figures 4.4 and 4.3. Since the number of dataset samples is large with respect to the traversed area of the table, multiple transmissions occur at the same positions. When random test sample selection occurs, almost always at least one of the repetitions will be at the train set. Thus, at inference time, the learning model is able to relate one of the repetitions found in the learning phase to the test sample. However, the model fails with the proposed test set selection methods since the repetitions at any position are all in either the train or test set. The way KNN method works gives it an edge over the MLP NN since it focuses on relating the test set sample to the closest one in the train set which often turns out to be one of the repeated measurements at the same position. MLP NN, on the other hand, forms a complex highly non-linear function where similar CSI input results in similar output positions but not superposing. This results in a lower mean error when KNN is used.

In the subsequent sections, we discuss a more challenging localization problem where the generalization capability is a must to achieve decent error. We show some attempts to improve the performance of the KNN and MLP NN models used so far.

5.3 Outdoor Localization Problem

In the previous indoor localization problem, we showed that both classical and deep learning approaches lack generalization. While this is an interesting insight, both models performed very well in the typical case with random selection of training and test samples which is considered satisfactory. In this part, we introduce another localization problem where the context makes it very difficult to achieve decent error without the generalization aspect. A more challenging localization problem

introduced by IEEE's Communication Society [89] on a larger scale [2] in their "Communication Theory Workshop 2020" event. The event was postponed due to the COVID-19 pandemic but the experimental dataset was shared with the scientific community.

The common aspect between the indoor positioning competition [1] introduced in chapter 4 and the outdoor positioning competition is that both are CSI-based localization. Table 5.1 summarizes the characteristics of both competitions:

Table 5.1: Indoor vs Outdoor characteristics

Aspect	**Indoor [1]**	**Outdoor [2]**
Measurements	CSI + SNR	CSI + SNR
Antennas	2×8	8×8
Subcarriers	924	924
Environment	Conference room	Residential area
Noise	Low	High
Area	Table: 8 m^2	Streets: several kms
Learning Algorithm	Supervised	Semi-supervised

5.3.1 Experimental Setup

The main component of the experiment is the massive MIMO antenna [46] which is composed of 8×8 sub antennas. The available subcarriers are the same as that of the indoor competition. The high noise aspect is caused by NLoS transmissions because of the surrounding buildings and obstacles while traversing the road. Figure 5.11 shows the map of the traversed region in the left subfigure and the Cartesian positions at which transmissions occurred in the right subfigure where the antenna is positioned at the origin. The learning problem in this case is semi-supervised since only a small percentage of the CSI readings are labelled with positions and the rest are not. There is a total of $\approx$ 5k labelled samples and 36k unlabelled samples. One more complexity aspect is that the organizers deliberately selected the labelled samples randomly from the traversed streets except for two streets. This means that a model should be general enough to deduce positions from two streets which were not seen by the model in the labelled dataset.

The readings in this experiment were recorded by sending 5 transmissions to the MIMO antenna at each position. Eight sub antennas were faulty and thus the readings were recorded for 56 antennas. When the noise is very high, the SNR value is manually set to -100 and CSI values are set to zero for all subcarriers. The ground

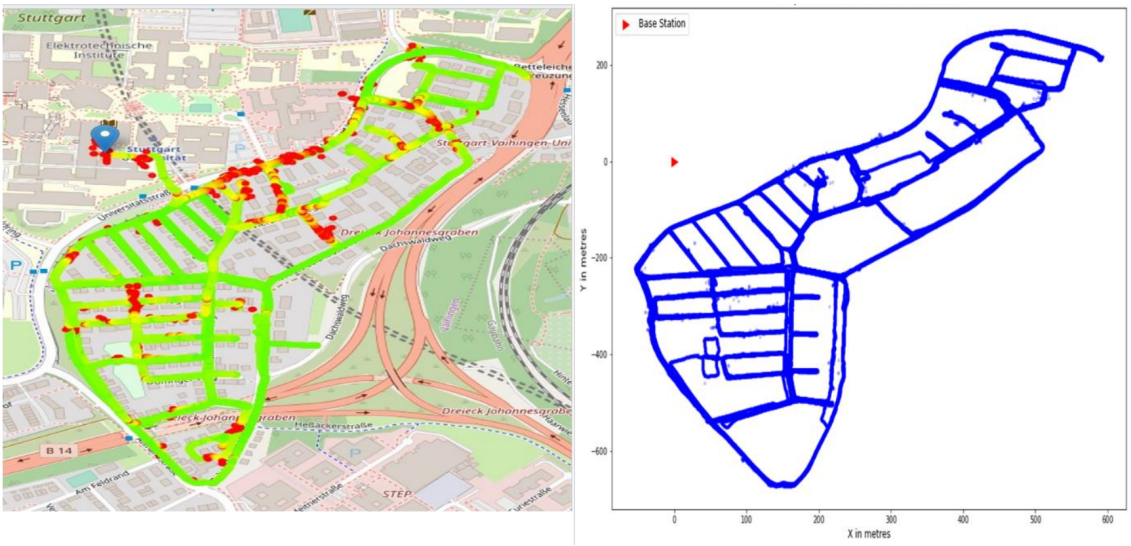


Figure 5.11: Traversed roads in a residential area in Stuttgart [2]

truth positions were recorded using a differential GPS with a mean error of 10 cm and a worst case error of 1 m. For each position there are $5 \times 56 \times 924 \times 2$ CSI which correspond to the 5 measurements, 56 antennas, 924 subcarriers, and Re, Im CSI complex components.

5.3.2 Data Preprocessing

In the indoor positioning problem, we showed the stability of the magnitude component and decided accordingly to use it as the input for the learning model. Only a small subset of readings are labelled with positions, and most of the readings are sparse because they are distributed along the large map. This sparsity makes it difficult to study the component stability in time. However, by computing the correlation coefficient between the 5 transmissions from the same position, similar results were obtained revealing the stability of magnitude and instability of phase, real, and imaginary components. Figure 5.12 presents the mean correlation coefficient between the 5 transmissions at each labelled position. The magnitude component still shows the highest stability with an average correlation coefficient very close to one. Thus, we choose it as the input to the learning model.

5.3.2.1 Fourier based Noise and Dimensionality reduction

One of the major differences between the magnitude readings in outdoor and indoor contexts is the variation of magnitude along the subcarriers. In the LoS scenario indoors, the variation is less fluctuating than in the noisy outdoor case. In addition, there are 40 more antennas than the previous case. Hence, we need a method other than the introduced polynomial regression that is able to represent the variation accurately in acceptable time. Figure 5.13 gives an idea of the high

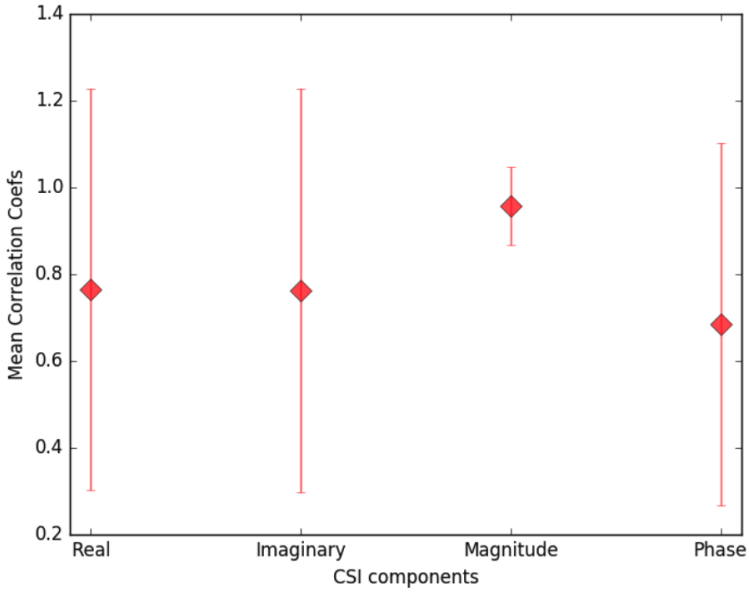


Figure 5.12: CSI components stability per position

information content for each measurement by showing the magnitude values for the 5 transmissions from a random position in the map at the 56 antennas. The cases where all values are zeros represent the high noise transmission that is set to zero manually. The magnitude fluctuations are much vigorous and thus difficult to approximate. It is possible to approximate the readings by a line using many small subdivisions along the subcarrier spectrum. However, it is difficult to select a suitable number of divisions because this fluctuation is not the same along all positions. When the subdivisions are very small, the line approximation will not be smooth. Moreover, the process will be much slower because of the polynomial regression optimization repeated on more divisions.

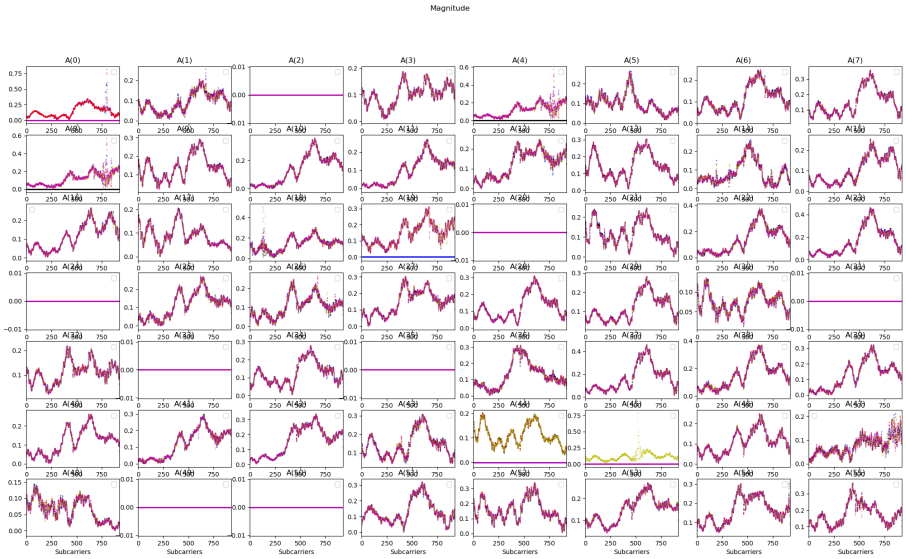


Figure 5.13: Magnitude CSI readings for 5 transmissions per position

A better approximation in terms of consistency along different fluctuation cases

and processing time is Fourier series [90]. The Fourier series approximates any function in terms of infinite sums of sines and cosines of increasing frequencies. Thus, we can control the fluctuation flexibility by limiting the number of used sines and cosines. Furthermore, its computation is much faster especially with the possible use of the Fast Fourier transform [91]. Equation 5.1 depicts the Fourier series approximation of a function $f(x)$ that is R periodic. Equations 5.2 and 5.3 show the computation of the coefficients of sine and cosine waves (A_n and B_n), respectively.

$$f(x) = \frac{A_0}{2} + \sum_{n=1}^{N} \left(A_n sin\left(\frac{2\pi}{R} nx\right) + B_n cos\left(\frac{2\pi}{R} nx\right)\right) \tag{5.1}$$

$$A_k = \frac{2}{R} \int_R f(x) cos\left(\frac{2\pi nx}{R}\right) dx \tag{5.2}$$

$$B_n = \frac{2}{R} \int_R f(x) sin\left(\frac{2\pi nx}{R}\right) dx \tag{5.3}$$

The choice of N specifies the maximum frequency of sine waves that are used to represent $f(x)$. Therefore, a higher value of N allows more fluctuations in the Fourier approximation. The Fourier series can also be applied when there is a data point representation of the $f(x)$ rather than a closed form. We refer the reader to [90] for more details. Choosing a low value of N can misrepresent the fluctuation while a high value for N can unnecessarily overfit. Figure 5.14 shows the Fourier series approximation for 5 transmissions at one antenna. The magnitude values are averaged over the 5 transmissions and the Fourier series components are computed with $N = 5$ and $N = 15$. In the left subfigure, it could be seen that the approximation could not follow the variation as accurately as when $N = 15$ in the right sub figure. Consequently, we use a value of $N = 15$. We use 66 equidistant magnitude values along the fitted line based on the previous success in the indoor experiment.

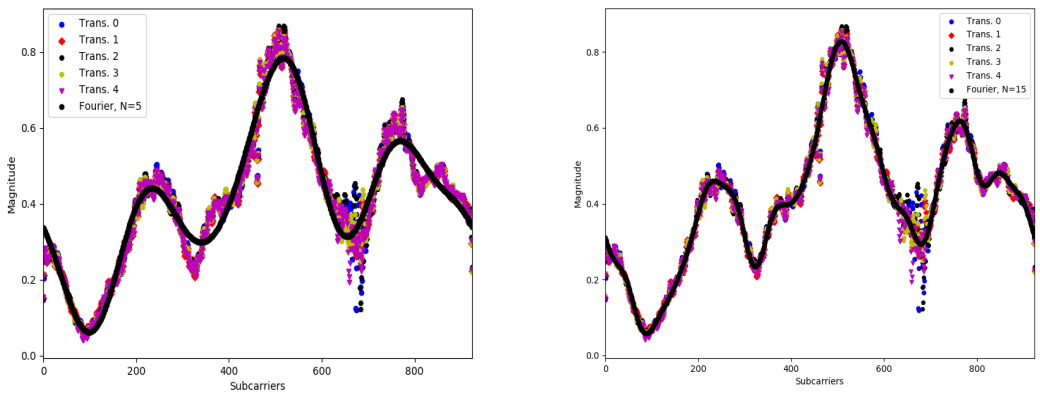


Figure 5.14: Fourier Series approximation of 5 transmissions with $N = 5$ and $N = 15$.

5.3.2.2 Missing Readings replacement

One main component of the data preprocessing pipeline in machine learning is missing values replacement. Indeed, using the magnitude input with zero values or SNR = -100 for noisy readings would disrupt the learning process. Hence, we studied the effect of the manual setting of values and attempted to smooth the manually-set values. Figure 5.15 shows the SNR color map where SNR values of -100 db have the darkest red color and the highest SNR value (20 db) transmissions have the lightest green color. The SNR values in between are linearly scaled. The MIMO antenna is placed at (0,0) coordinates. It can be seen that the SNR value at the upper right part of the map contains the lowest SNR values represented in red. The highest SNR readings are those closest to the antenna which is expected.

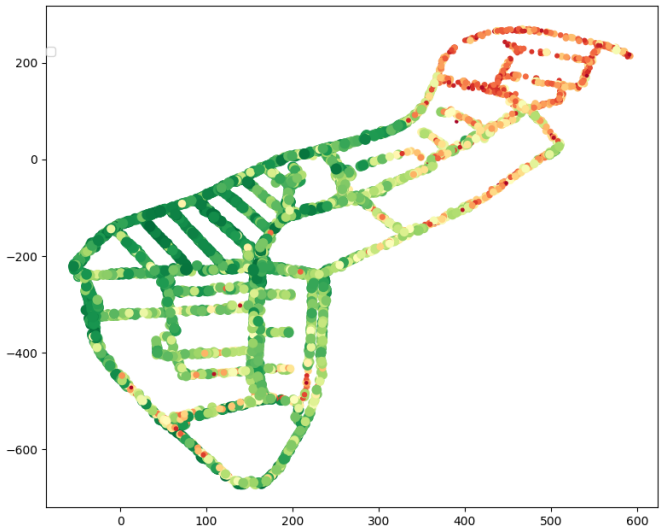


Figure 5.15: SNR color map

To better understand the SNR readings of nearby positions, the SNR difference between positions within 3m range is calculated and their frequencies are recorded. Figure 5.16 displays the log scaled SNR differences between all position pairs within 3m range in the labelled dataset. The mean SNR difference is 20 db between positions within 3m distance from each other. The fact that the standard deviation is ≈ 18 db reveals the inconsistent difference in SNR values which could be a result of the manually-set SNR values when high noise transmissions occur.

We attempted three methods of replacing the -100 db SNR values:

1. Global minimum replacement: replace the -100 SNR values in any transmission by the minimum global SNR value less than -100 db in the dataset. This value was found to be ≈ -40 db.
2. Minimum per sample replacement: replace the -100 SNR values at any antenna in a given transmission by the minimum non -100 db SNR value measured

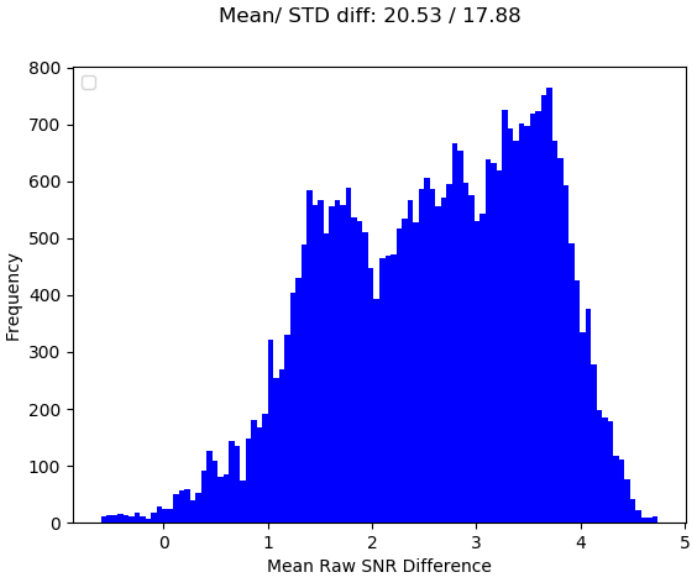


Figure 5.16: Log scaled SNR differences between position pairs within 3m range

at other antennas of the same sample. If one antenna has -100 db SNR, this antenna is assigned the same SNR value of the antenna with the minimum SNR value above -100 db from the same transmission.

3. Mean per sample replacement: similar to sample minimum replacement but instead of the minimum SNR value, the mean SNR of other antennas with SNR above -100 db is the replacement value.

The SNR difference between nearby positions is a good measure of the smoothness of the change of SNR along the map. Figure 5.17 demonstrates the mean and standard deviation of the raw SNR values and the three replacement methods. These replacement methods result in a smoother SNR variation between nearby positions. This facilitates the learning process as the spike changes that disrupt the learning process are mitigated. The mean replacement method results in the smallest average SNR difference between nearby positions; thus, we believe it is a decent choice. Figure 5.18 depicts the color map and SNR difference between positions within 3m range when the mean per sample replacement method is applied.

The same method is applied to CSI values which are set to zero. At any given antenna in any sample, if all 5 repeated transmissions from the same position yield zero values, each subcarrier value is replaced with the mean of non-zero CSI values from all other antennas at the same subcarrier in the same sample.

5.3.3 Learning from Labelled dataset

After preprocessing the data, the following step is building a learning model. We start by trying to learn as much as possible from the labelled portion of the dataset

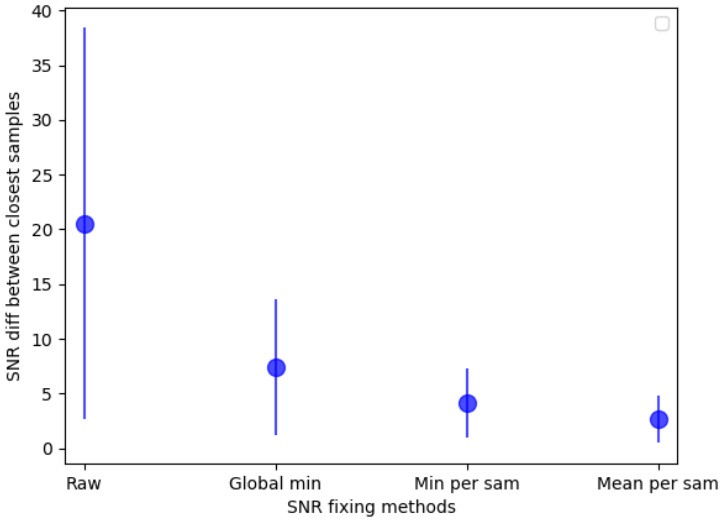


Figure 5.17: SNR differences between position pairs within 3m range for each replacement method.

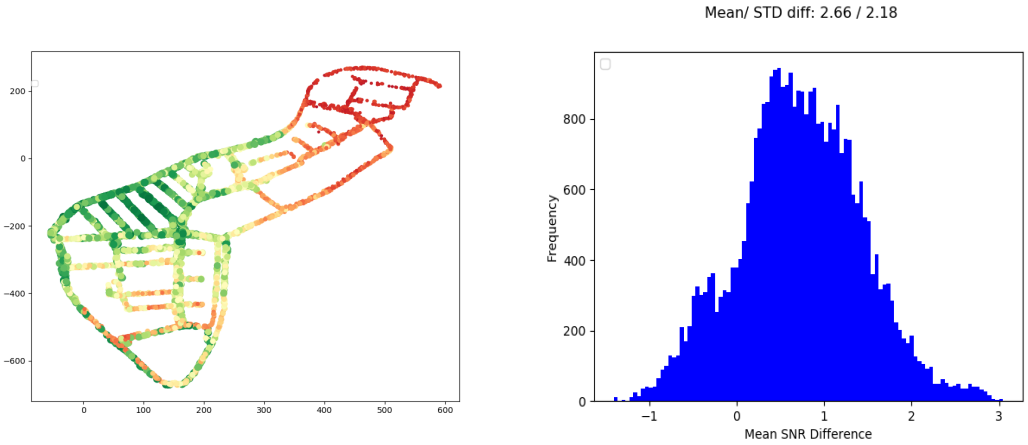


Figure 5.18: mean per sample replacement method: color map (left figure) and SNR differences between positions within 3m (right figure)

(5k samples). It is clear that a satisfactory low error estimation cannot be achieved with the labelled dataset only due to the sparsity of readings. However, making the best use of the labelled dataset gives interesting insights and paves the way for exploiting the unlabelled dataset.

Starting from the best solution in terms of error in the indoor positioning competition, we apply the K-nearest neighbor method to the labelled dataset. The data is split into 90 % training set and 10 % test set. The value of k is set to one and the neighboring criterion is chosen to be the Euclidean distance between CSI magnitude component values. KNN is experimented with polynomial regression and Fourier fitting yielding a mean error of 118 m and 130 m, respectively. The reason of high error can be explained by the fact that the data is too sparse for the model to learn or that KNN is too naïve to capture the effective features. The answer to this question is a combination of both; this is evident by the performance

of MLP NN on the labelled dataset. The MLP NN hyperparameters are the same as those of the highest performing MLP NN in the indoor problem summarized in table 4.1. The MLP NN achieves a mean error of 44 m with polynomial regression approximation of the CSI magnitude values and 37 m with the use of Fourier. Using Fourier approximation without the replacement of missing values yields an error of 47 m. The large gap between the error of KNN and MLP NN shows that MLP NN was able to capture some interesting features that KNN's simplistic approach failed to detect. However, the fact that the error is still much higher than the accuracy of the differential GPS used to record the coordinates ($<$ 1m) shows that the sparsity of the datapoints hinders the learning process significantly. Table 5.2 summarizes the experiments with KNN and MLP NN with some variation of the preprocessing steps. One phenomenon that we are not yet able to explain is why our experiment shows a little deterioration in accuracy when SNR readings are aggregated with the CSI readings. Hence, we focus our work on exploiting CSI only to predict positions as we previously did in the indoor problem.

Table 5.2: Performance of different preprocessing steps

Learning Model	**Preprocessing**	**RMSE**
KNN	Polynomial Regression, mean replacement	118m
KNN	Fourier fitting, mean replacement	130 m
MLP NN	Polynomial regression, mean replacement	44 m
MLP NN	Fourier fitting, mean replacement	37 m
MLP NN	Fourier fitting, no replacement	47 m

5.3.4 Generalizing to unlabelled dataset

5.3.4.1 Distance estimation between CSI pairs

Generalizing from the small portion of the labelled dataset to the unlabelled dataset requires learning more relations and common features between CSI readings from different samples. CNNs are deep learning models that have proven their great potential in capturing meaningful relations from related input features. As previously mentioned in chapter 2, CNNs have accomplished unprecedented success in the image recognition field. While CNNs have a lot in common with MLP NNs, their principal innovation lies in the architecture alteration that forces the model to learn useful features from neighboring pixels in an image. Our attempt lies in exploiting CNNs' ability to capture effective relational features between unlabelled dataset samples to labelled ones to fill the sparsity gap in the labelled dataset.

A vivid correlation to exploit is that of the magnitude value of consecutive subcarriers. The values are generally very close to each other and a group of values form unique shapes such as maxima, minima, lines with different slopes, etc. However, treating the CSI readings of each antenna (figure 5.13) as a separate image to process would lead to a very high computational demand. The fact that each subfigure is sparse and is made up of only one line allows us to exploit this sparsity and represent the variation in a more compact way. We propose representing the CSI magnitude values in an $A \times N$ 2D matrix, where A is the number of antennas (56 in our case) and N is the number of selected values along the fitted line. Looking at the formed matrix as an image, each element in the matrix could be thought of as the pixel's intensity. High pixel intensity represents a high magnitude value, e.g., a maxima would look like a group of adjacent pixels with the highest intensity in the middle.

Our proposal to learn from the unlabelled dataset is by learning relational features that allow distance prediction between CSI sample pairs. Then when enough distances are predicted between CSI pairs, whether labelled or not, the samples from the unlabelled dataset with unknown positions can be estimated using a variant of the range based localization presented in chapter 3. In such context, the anchor nodes would be represented by the known positions of the labelled samples and the non-anchor nodes would be the unlabelled samples. This approach is based on three assumptions. First, the distance information can be deduced from two CSI pairs up to a given distance. Second, the distance deduction is general enough to be estimated between sample pairs from any region in the map. This is important because it is explicity said that samples collected from two streets in the map were purposely omitted from the labelled portion of the dataset. Third, the dataset is dense enough to be able to predict enough distances and use a range-based localization solution.

The arrangement of CSI pairs is selected to fit in the CNN model so that the CNN can learn relations between consecutive CSI values in one sample and corresponding CSI values between sample pairs. This is achieved by using a similar method as the 2D matrix arrangement of one CSI sample. The dimension of the 2D matrix is still $A \times N$ where A is the number of antennas and N is the number of magnitude values along the fitted line. The difference is in the number of channels; like RGB in the image context, we add another channel for the second pair. The reason behind this arrangement is to enforce local neighborhood between possibly related values. Horizontally in each row, consecutive CSI values are correlated. In the channel depth direction, corresponding CSI values at the same antennas could be related to deduce the distance.

It is worth mentioning that there is one degree of freedom in choosing which of the two samples to be in the front or back channels. CNNs are sensitive to ordering; thus, choosing one order and leaving out the other could possibly hurt the generalization of CNNs. Therefore, each sample pair is used in both orders for the same distance output. In order to generate a dataset with CSI pair input and distance output, we must choose which pairs to include in the learning process. Using all possible CSI pairs in the labelled dataset is not feasible because the number

of possible pairs is $5k!$, which is obviously a very large number. It is highly probable that CSI values of pairs which are tens of meters apart are not relatable while pairs which are close could be relatable.

Choosing all CSI pairs within 3m distance from each other as the new dataset yields $\approx$ 17.5k samples. This number is doubled because we use both orders for the channel arrangement of CSI pairs. Using an 80 %, 20 % split for training and test sets respectively, a CNN is trained to predict distance from CSI sample pairs. The CNN is made up from 3 convolutional layers, each followed by a pooling layer. The convolutional part is then followed by 3 dense layers. Table 5.3 summarizes the hyperparameters of the used CNN.

Table 5.3: CNN Hyperparameters

Hyperparameter	**Value**
Conv. Layers	3
Kernel size	1×2
Filters per Layer	10
Pooling method	Max pooling
Pool size	1×2
Stride size	1×1
Dense layers	3
Units per dense layer	256
Activation Function	relu
Learning Rate	0.001
Optimizer	Adam
Epochs	100
Dropout Percentage	0%
Batch Sizes	[32, 64, 128, 256, 512, 1024]

The CNN was able to predict with high accuracy the distance between CSI input pairs up to 3m. The mean error is $\approx$ 10 cm; Figure 5.19 shows the distribution of the log of errors on test set predictions. The error distribution is almost following a log-normal distribution with a small bump towards the right part of the distribution. This could be explained by our initial hypothesis that there is possibly no relatable features between readings which are few meters apart. Thus, we suspect that the large errors are due to distant CSI pairs that the CNN was not able to find rules to relate them. To evaluate this hypothesis, the estimation errors are calculated for pairs where predicted distances are less than defined thresholds by up to 3m. We

expect that the error would increase as the distance between pairs increases reflecting lack of relatable features. Figure 5.20 demonstrates the mean error and standard deviation for each maximum distance threshold. It could be clearly seen that the error and standard deviation increase with the increase of threshold, especially after 1.5 m. The large standard deviation is due to the skewed nature of log normal distributions. We chose to use predicted distance thresholds rather than ground truths distances for evaluating this hypothesis. The reason is that when ground truth is not available as in the unlabelled dataset, there would be no access to ground truth values but the predicted distances can be an adequate criterion to accept or reject the prediction.

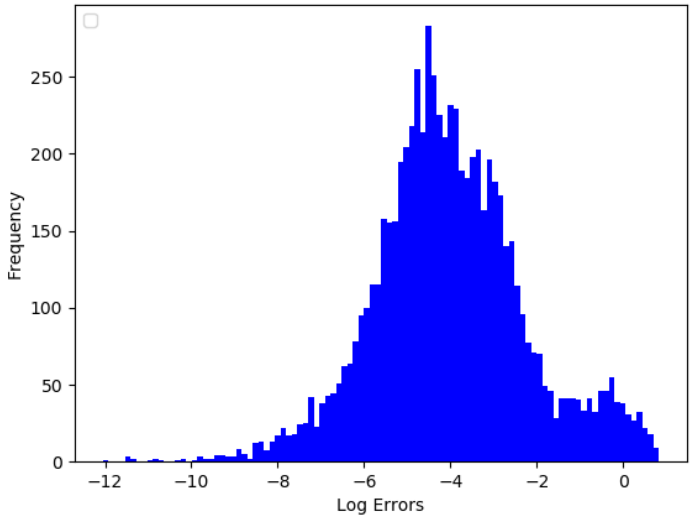


Figure 5.19: CNN error distribution for distance estimation

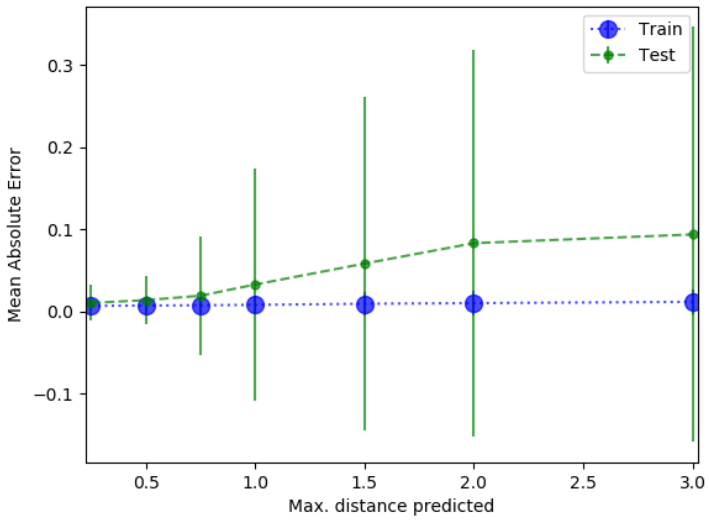


Figure 5.20: CNN distance estimation errors per maximum predicted distance thresholds

5.3.4.2 CNNs: Generalization Evaluation

Another important aspect that has to be considered is the generalization of the distance estimation capability beyond the regions where training examples are found. This could let us know if, during the learning process, the CNN memorized the distance between each particular pair of CSI samples or if it learned general features that are independent from the absolute CSI values and dependent on some general difference between the CSI pair. To do so, we repeated the same concept of separating the training set from the test set as previously experimented in the beginning of this chapter on the indoor positioning problem. This is particularly important in this competition because the authors purposely omitted readings from two streets in the map. Thus, to be able to predict positions in such region, the CNN must be able to generalize beyond the training set region. Figure 5.21 depicts the separation between train and test sets. The training set makes up 71 % of the dataset and the test set makes up 29 %. This separation leads to a small deterioration of performance especially when the distance separating CSI pairs is large. However, for predicted distances of 1 m or below, the performance is almost the same as shown in figure 5.22.

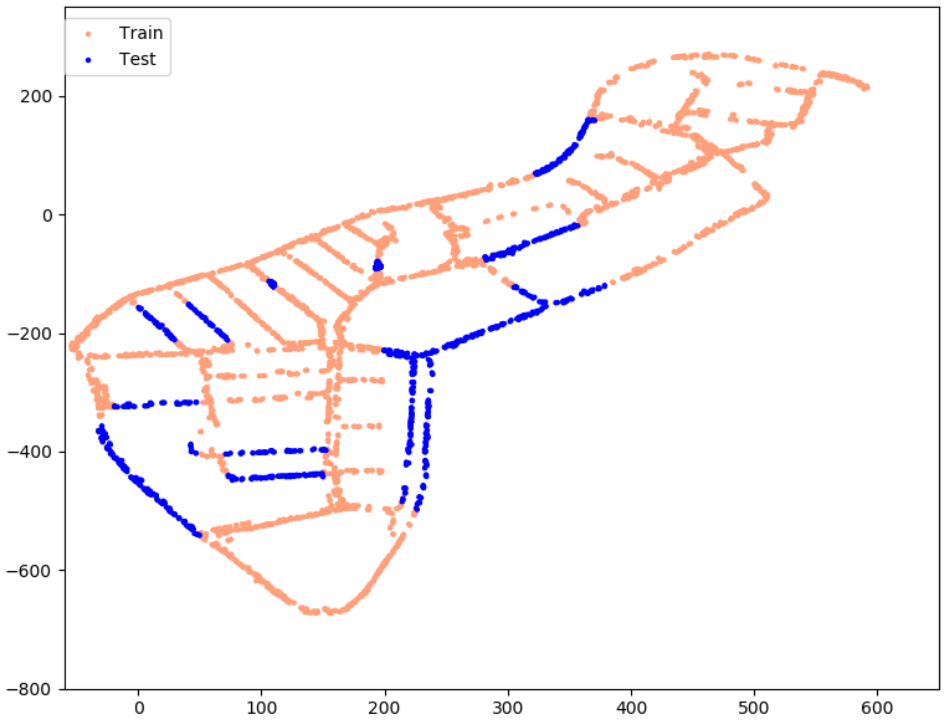


Figure 5.21: Map train and test sets which are separated for generality evaluation.

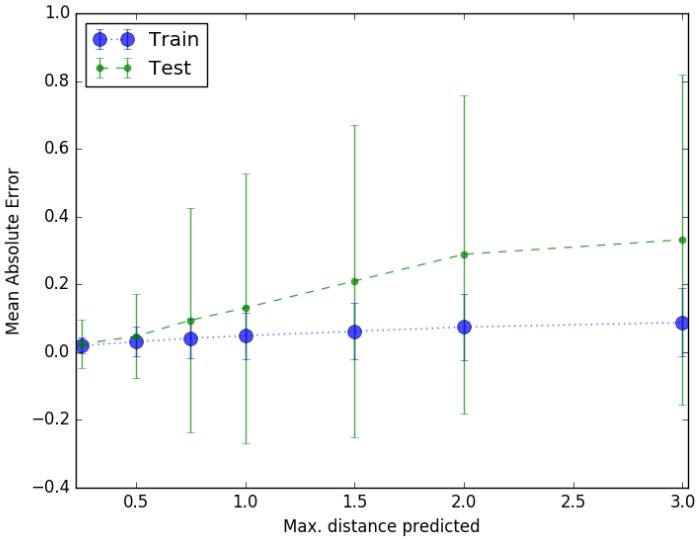


Figure 5.22: CNN distance estimation errors per maximum predicted distance thresholds for the generality evaluation

5.3.4.3 From Distance estimation to position estimation

While the CNN was able to achieve high accuracy in distance estimation between CSI pairs and showed decent generalization potential, the original aim is to predict positions. The idea is to build a graph where all CSI samples are connected to other sample pairs where edges would indicate distances predicted by CNN. Using distance knowledge between CSI samples and position knowledge from the labelled dataset, positions of unlabelled samples could be estimated using a triangulation based method such as the one presented in chapter 3. Other solutions are possible; for example Graph Neural Networks (GNNs) would fit nicely since the problem is formulated as a graph. Also, an optimization framework such as factor graphs [92]could be used to reach the positioning goal.

The current obstacle hindering the progress towards this step is the inability of CNNs to reject very far pairs and wrongly assign small distance separations. The fact that CNNs are trained to estimate distances for pairs which are 3 meters away restrain its output to the 3m range. Thus, in some case where the CSI pairs are several hundred meters apart, the CNN could still estimate a small distance between them. These wrong estimations are catastrophic for the position estimation step that is to follow. Given time constraints, this part is set to be solved in the future work. We see great potential in this approach which is supported by the presented results and generalization capability of the learning model.

5.4 Conclusion

In this chapter we touched on an essential aspect of machine learning which is the generalization capability of learning models. In the first part of the chapter, we showed that despite the highly accurate position prediction results, both the KNN and MLP NN learning methods fail to generalize in the indoor positioning problem presented in chapter 4. This motivated the exploration of more challenging problems where the generalization ability is essential to achieve accurate prediction results.

We then presented the CTW 2020 outdoor positioning competition [2] which is more challenging due to noise and missing position labels for most of the dataset samples. Thus, in order to achieve good estimation accuracy, the learning model has to make the best use of the small labelled dataset. Also, it has to be able to generalize and relate the labelled dataset to the unlabelled one. We present two data preprocessing steps to accommodate for the new challenges. First, we use Fourier based fitting instead of polynomial regression for a faster and more consistent representation of the magnitude variation. Also, we represent a method to replace missing readings due to high noise that leads to better learning process and more accurate estimation.

We propose the use of CNN to relate CSI sample pairs with the aim of estimating the distances between pairs. The objective is to build a graph where nearby CSI sample pairs are related by distance, then using position knowledge from the labelled dataset, the position of unlabelled samples could be deduced. CNNs showed highly accurate distance predictions up to 3 m. Also, when the train and test datasets where separated, the distance prediction ability was only slightly affected which shows a good generalization potential. The current bottle neck lies in the miss-prediction of some very far CSI sample pairs as being close in distance. These miss-predictions, even if they are not very frequent, inhibit decent positioning prediction. The enhancement to this issue is left for future work as the presented results of accurate distance predictions, in our opinion show a great potential of this approach.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This work entailed two principal aspects; the first focused on solving the localization problem using classical and machine learning methods under different constraints. The second part focused primarily on enhancing the generalization aspect of machine learning models.

The work presented in Chapter 3 targets the localization problem in a MANETs context. The solutions assume the knowledge of distance between 1-hop neighbors with Gaussian noise. The distances could be provided using various measurements such as TOA or RSSI. In addition to distance knowledge, the positions of few nodes (minimum three) are assumed to be known in order for the proposed solution to work. This problem is known in the community as range-based localization. The proposed solution is built with two main criteria. The first criterion is the simplicity of the solution with respect to: required devices, computational cost, and network bandwidth. Consequently, the proposed method suits scenarios where cost, energy, and bandwidth are scarce. The second criterion is the distributive nature of the solution to avoid single point of failure. The presented solution (PCP) is based on triangulation with some variations to accommodate the nature of the problem. The energy consumption is reduced by requiring only 3 anchor nodes which could be achieved using GPS. Also, the triangulation computations are simple when compared to other optimization based solutions. Finally, bandwidth is conserved by using a semi-flooding location service [15] to share location information with other nodes in the network. This solution was compared to GPS-free [19] and showed better ability to localize a larger set of nodes in a fairly dense network. PCP localizes ≈ 90 % of nodes within a network with an average degree of ≈ 10. In future work, this solution can be enhanced by incorporating the uncertainty of estimated positions and mitigate the increased uncertainty towards the edge of the network. Also, the solution could be improved to adapt to the mobility of nodes.

In Chapter 4, we tackled the localization problem from a machine learning perspective where a public dataset is used to make a mapping from CSI to transmitter position. The CSI is measured for transmissions between a massive 2×8 MIMO antenna [46] and a transmitter moving along 2×4 meter table. The public dataset

was made available to the community by the organizers of IEEE's Communication Theory Workshop for an indoor positioning competition during the event [1]. We proposed a classical learning solution, KNN, and a deep learning solution, MLP NN, to model the mapping between CSI and the location of the transmitter. Before applying the models, an essential data preprocessing step is proposed which enhances the learning process. First, the CSI complex components are studied leading to a conclusion that the magnitude component is the most stable among others. This conclusion is supported by statistical experiments and state-of-the-art works which supported our choice to use the magnitude component as the input to the learning model. This is followed by a noise and dimensionality reduction step where polynomial regression is used to fit a line through the magnitude readings. A reduced number of points is selected along the fitted line to be the input to the learning model. This preprocessing step paved the way for highly accurate position estimation using KNN and MLP NN. Our solution achieved an error 2.3 cm RMSE securing the first place in IEEE's CTW indoor positioning competition [1] among 8 teams from top universities around the world such as: University of Toronto (Canada), Ruhr University Bochum (Germany), Heriot-Watt University (England), University of Padova (Italy), IMdea networks institute (Spain), Aalborg University (Denmark), and Yuan Ze University (Taiwan). During the announcement ceremony, the organizer informed Abdallah Sobehy that the estimation accuracy is better than that of the experiment authors which is an honorable testimony.

Chapter 5 includes the second main aspect of our work which is the generalization aspect of machine learning models. We started this study by evaluating the generalization capability of our solutions using KNN and MLP NN which, to the best of our knowledge, yield the highest accuracy on the introduced dataset. The evaluation entailed the separation between the training set and test set in a way such that the learning model would have to estimate test samples outside the training samples region. We proposed two ways of separation to evaluate the interpolation and extrapolation capabilities of the learning models. The estimation accuracy deteriorated immensely with the proposed arrangement leading us to conclude that both KNN and MLP NN fail to generalize. These experiments led to interesting insights on the dataset and the reason that gave KNN an edge over MLP NN in the positioning accuracy.

The last piece of this work entails the study of a more challenging outdoor localization problem [2]. This problem entailed a high noise environment and only a small portion of the dataset is labelled with the corresponding position which introduces a new set of challenges. These challenges made the generalization aspect of the learning models inevitable to achieve a decent positioning accuracy. The larger antenna which is 8×8 provides more measurements that needed a quicker noise and dimensionality reduction method. Also, the high noise aspect of the experiment required the method to be much more flexible. We proposed the usage of Fourier based line fitting to satisfy these requirements. Also, there were missing readings that hindered the learning process which we mitigated by replacing the missing value by an averaging technique. Since only a small subset of the dataset is labelled with positions, we propose the usage of CNNs to capture relational features between CSI pairs to estimate the distance. This distance could then be used to estimate

the positions of the unlabelled dataset with the help of the labelled positions. Our solution achieves accurate distance estimation with 10 cm error between CSI pairs up to 3m apart.

6.2 Future Work

The outdoor localization problem is yet to be solved as we are able to accurately estimate distance between nearby samples but face some challenges in estimating distance to further positions. We believe that there is a great potential in this approach as it shows good generalization capability outside the training set. A more flexible deep learning method that could be used to take one more step towards the solution is Graph Neural Networks (GNN) [53]. GNN is a suitable framework since the problem can be formulated as a graph where labelled positions are nodes and estimated distances are edges. It is also possible to use the range based localization solution (PCP) by considering the labelled dataset positions as anchor nodes and using distances and triangulation to estimate unlabelled dataset positions. However, this approach is sensitive to large distance errors.

References

[1] *IEEE Communication Theory Workshop positioning algorithm competition*, https://ctw2019.ieee-ctw.org/authors/, Accessed: 2020-23-02.

[2] *IEEE Communication Theory Workshop 2020 outdoor positioning algorithm competition*, https://ctw2020.ieee-ctw.org/data-competition/, Accessed: 2020-07-01.

[3] J. Schiller and A. Voisard, *Location-based services*. Elsevier, 2004.

[4] *Uber*, https://www.uber.com/fr/en/about/, Accessed: 2020-15-09.

[5] *Autonomous driving example*, https://www.smartcitiesworld.net/, Accessed: 2020-15-09.

[6] B. Karp and H.-T. Kung, "Gpsr: Greedy perimeter stateless routing for wireless networks", in *Proceedings of the 6th annual international conference on Mobile computing and networking*, 2000, pp. 243–254.

[7] *The 3rd generation partnership project (3gpp)*, https://www.3gpp.org/about-3gpp, Accessed: 2020-15-09.

[8] D. Lymberopoulos and J. Liu, "The microsoft indoor localization competition: Experiences and lessons learned", *IEEE Signal Processing Magazine*, vol. 34, no. 5, pp. 125–140, 2017.

[9] A. Nordrum, "The internet of fewer things [news]", *IEEE Spectrum*, vol. 53, no. 10, pp. 12–13, 2016.

[10] F. Boccardi, R. W. Heath, A. Lozano, T. L. Marzetta, and P. Popovski, "Five disruptive technology directions for 5g", *IEEE communications magazine*, vol. 52, no. 2, pp. 74–80, 2014.

[11] X. Wang, L. Gao, S. Mao, and S. Pandey, "Deepfi: Deep learning for indoor fingerprinting using channel state information", in *2015 IEEE wireless communications and networking conference (WCNC)*, IEEE, 2015, pp. 1666–1671.

[12] W.-S. Jung, J. Yim, and Y.-B. Ko, "Qgeo: Q-learning-based geographic ad hoc routing protocol for unmanned robotic networks", *IEEE Communications Letters*, vol. 21, no. 10, pp. 2258–2261, 2017.

[13] Y. Terao, P. Phoummavong, K. Utsu, and H. Ishii, "A proposal on void zone aware greedy forwarding method over manet", in *2016 IEEE Region 10 Conference (TENCON)*, IEEE, 2016, pp. 1329–1333.

[14] Y.-B. Ko and N. H. Vaidya, "Location-aided routing (lar) in mobile ad hoc networks", *Wireless networks*, vol. 6, no. 4, pp. 307–321, 2000.

[15] É. Renault, E. Amar, H. Costantini, and S. Boumerdassi, "Semi-flooding location service", in *2010 IEEE 72nd Vehicular Technology Conference-Fall*, IEEE, 2010, pp. 1–5.

[16] R. Huang and G. V. Záruba, "Monte carlo localization of wireless sensor networks with a single mobile beacon", *Wireless Networks*, vol. 15, no. 8, p. 978, 2009.

[17] N. Bulusu, J. Heidemann, and D. Estrin, "Gps-less low-cost outdoor localization for very small devices", *IEEE personal communications*, vol. 7, no. 5, pp. 28–34, 2000.

[18] D. Moore, J. Leonard, D. Rus, and S. Teller, "Robust distributed network localization with noisy range measurements", in *Proceedings of the 2nd international conference on Embedded networked sensor systems*, 2004, pp. 50–61.

[19] S. Čapkun, M. Hamdi, and J.-P. Hubaux, "Gps-free positioning in mobile ad hoc networks", *Cluster Computing*, vol. 5, no. 2, pp. 157–167, 2002.

[20] R. M. Buehrer, H. Wymeersch, and R. M. Vaghefi, "Collaborative sensor network localization: Algorithms and practical issues", *Proceedings of the IEEE*, vol. 106, no. 6, pp. 1089–1114, 2018.

[21] P. Biswas and Y. Ye, "Semidefinite programming for ad hoc wireless sensor network localization", in *Proceedings of the 3rd international symposium on Information processing in sensor networks*, 2004, pp. 46–54.

[22] E. G. Larsson, "Cramer-rao bound analysis of distributed positioning in sensor networks", *IEEE Signal processing letters*, vol. 11, no. 3, pp. 334–337, 2004.

[23] F. Mourad, H. Snoussi, F. Abdallah, and C. Richard, "Guaranteed boxed localization in manets by interval analysis and constraints propagation techniques", in *IEEE GLOBECOM 2008-2008 IEEE Global Telecommunications Conference*, IEEE, 2008, pp. 1–5.

[24] F. Mourad, H. Snoussi, F. Abdallah, and C. Richard, "Model-free interval-based localization in manets", in *2009 IEEE 13th Digital Signal Processing Workshop and 5th IEEE Signal Processing Education Workshop*, IEEE, 2009, pp. 474–479.

[25] F. Mourad, H. Snoussi, and C. Richard, "Interval-based localization using rssi comparison in manets", *IEEE Transactions on Aerospace and Electronic Systems*, vol. 47, no. 4, pp. 2897–2910, 2011.

[26] D. L. Waltz, "Generating semantic descriptions from drawings of scenes with shadows", 1972.

[27] H. Sallouha, A. Chiumento, and S. Pollin, "Localization in long-range ultra narrow band iot networks using rssi", in *2017 IEEE International Conference on Communications (ICC)*, IEEE, 2017, pp. 1–6.

[28] Z. Chen, F. Xia, T. Huang, F. Bu, and H. Wang, "A localization method for the internet of things", *The Journal of Supercomputing*, vol. 63, no. 3, pp. 657–674, 2013.

[29] R. Peng and M. L. Sichitiu, "Robust, probabilistic, constraint-based localization for wireless sensor networks", in *2005 Second Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, 2005. IEEE SECON 2005.*, Citeseer, 2005, pp. 541–550.

[30] M. L. Sichitiu and V. Ramadurai, "Localization of wireless sensor networks with a mobile beacon", in *2004 IEEE international conference on mobile Ad-hoc and sensor systems (IEEE Cat. No. 04EX975)*, IEEE, 2004, pp. 174–183.

[31] T. M. Fruchterman and E. M. Reingold, "Graph drawing by force-directed placement", *Software: Practice and experience*, vol. 21, no. 11, pp. 1129–1164, 1991.

[32] T. Kamada, S. Kawai, *et al.*, "An algorithm for drawing general undirected graphs", *Information processing letters*, vol. 31, no. 1, pp. 7–15, 1989.

[33] J. Ellson, E. Gansner, L. Koutsofios, S. C. North, and G. Woodhull, "Graphvizâfixme"open source graph drawing tools", in *International Symposium on Graph Drawing*, Springer, 2001, pp. 483–484.

[34] T. Eren, O. Goldenberg, W. Whiteley, Y. R. Yang, A. S. Morse, B. D. Anderson, and P. N. Belhumeur, "Rigidity, computation, and randomization in network localization", in *IEEE INFOCOM 2004*, IEEE, vol. 4, 2004, pp. 2673–2684.

[35] Z. Yang, Z. Zhou, and Y. Liu, "From rssi to csi: Indoor localization via channel response", *ACM Computing Surveys (CSUR)*, vol. 46, no. 2, pp. 1–32, 2013.

[36] A. Agarwal and S. R. Das, "Dead reckoning in mobile ad hoc networks", in *2003 IEEE Wireless Communications and Networking, 2003. WCNC 2003.*, IEEE, vol. 3, 2003, pp. 1838–1843.

[37] W. Kim, M. Park, and S.-J. Lee, "Effects of shadow fading in indoor rssi ranging", in *2012 14th International Conference on Advanced Communication Technology (ICACT)*, IEEE, 2012, pp. 1262–1265.

[38] V. Jungnickel, K. Manolakis, W. Zirwas, B. Panzner, V. Braun, M. Lossow, M. Sternad, R. Apelfröjd, and T. Svensson, "The role of small cells, coordinated multipoint, and massive mimo in 5g", *IEEE communications magazine*, vol. 52, no. 5, pp. 44–51, 2014.

[39] J. Xiao, Z. Zhou, Y. Yi, and L. M. Ni, "A survey on wireless indoor localization from the device perspective", *ACM Computing Surveys (CSUR)*, vol. 49, no. 2, pp. 1–31, 2016.

[40] J. Xiao, K. Wu, Y. Yi, and L. M. Ni, "Fifs: Fine-grained indoor fingerprinting system", in *2012 21st international conference on computer communications and networks (ICCCN)*, IEEE, 2012, pp. 1–7.

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

[42] S.-H. Fang, T.-N. Lin, and K.-C. Lee, "A novel algorithm for multipath fingerprinting in indoor wlan environments", *IEEE transactions on wireless communications*, vol. 7, no. 9, pp. 3579–3588, 2008.

[43] Y. Liu, Z. Yang, X. Wang, and L. Jian, "Location, localization, and localizability", *Journal of Computer Science and Technology*, vol. 25, no. 2, pp. 274–297, 2010.

[44] Q. Song, S. Guo, X. Liu, and Y. Yang, "Csi amplitude fingerprinting-based nb-iot indoor localization", *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1494–1504, 2017.

[45] Z. Wu, L. Jiang, Z. Jiang, B. Chen, K. Liu, Q. Xuan, and Y. Xiang, "Accurate indoor localization based on csi and visibility graph", *Sensors*, vol. 18, no. 8, p. 2549, 2018.

[46] M. Arnold, J. Hoydis, and S. ten Brink, "Novel massive mimo channel sounding data applied to deep learning-based indoor positioning", in *SCC 2019; 12th International ITG Conference on Systems, Communications and Coding*, VDE, 2019, pp. 1–6.

[47] S. Hijazi, R. Kumar, and C. Rowen, "Using convolutional neural networks for image recognition", *Cadence Design Systems Inc.: San Jose, CA, USA*, pp. 1–12, 2015.

[48] X. Wang, L. Gao, and S. Mao, "Phasefi: Phase fingerprinting for indoor localization with a deep learning approach", in *2015 IEEE Global Communications Conference (GLOBECOM)*, IEEE, 2015, pp. 1–6.

[49] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks", in *Advances in neural information processing systems*, 2007, pp. 153–160.

[50] Y. Chapre, A. Ignjatovic, A. Seneviratne, and S. Jha, "Csi-mimo: Indoor wi-fi fingerprinting system", in *39th annual IEEE conference on local computer networks*, IEEE, 2014, pp. 202–209.

[51] G. Cybenko, "Approximation by superpositions of a sigmoidal function", *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[52] N. Chomsky, *Aspects of the Theory of Syntax*. MIT press, 2014, vol. 11.

[53] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, *et al.*, "Relational inductive biases, deep learning, and graph networks", *arXiv preprint arXiv:1806.01261*, 2018.

[54] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning", *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[55] J. Schmidhuber, "Deep learning in neural networks: An overview", *Neural networks*, vol. 61, pp. 85–117, 2015.

[56] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[57] S.-B. Cho, "Neural-network classifiers for recognizing totally unconstrained handwritten numerals", *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 43–53, 1997.

[58] Y. LeCun, "The mnist database of handwritten digits", *http://yann. lecun. com/exdb/mnist/*, 1998.

[59] *Mnist dataset in cnn*, `https://www.javatpoint.com/tensorflow-mnist-dataset-in-cnn`, Accessed: 2020-24-08.

[60] K. Fukushima, "Neocognitron", *Scholarpedia*, vol. 2, no. 1, p. 1717, 2007.

[61] S. Dieleman, *Deepmind x ucl | deep learning lectures | 3/12 | convolutional neural networks for image recognition*, `https://www.youtube.com/watch?v=shVKhOmT0HE&t=2863s`, Accessed: 2020-29-08.

[62] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge", *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[63] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks", in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[64] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition", *arXiv preprint arXiv:1409.1556*, 2014.

[65] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[66] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[67] W. Czarnecki, *Deepmind x ucl | deep learning lectures | 2/12 | neural networks foundations*, `https://www.youtube.com/watch?v=FBggC-XVF4M&list=PLqYmG7hTraZCDxZ44o4p3N5Anz3lLRVZF&index=2`, Accessed: 2020-29-08.

[68] M. Garnelo, *Deepmind x ucl | deep learning lectures | 6/12 | sequences and recurrent networks*, `https://www.youtube.com/watch?v=87kLfzmYBy8&list=PLqYmG7hTraZCDxZ44o4p3N5Anz3lLRVZF&index=6`, Accessed: 2020-29-08.

[69] W. Yin, K. Kann, M. Yu, and H. Schütze, "Comparative study of cnn and rnn for natural language processing", *arXiv preprint arXiv:1702.01923*, 2017.

[70] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks", in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[71] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate", *arXiv preprint arXiv:1409.0473*, 2014.

[72] S. Shalev-Shwartz, O. Shamir, and S. Shammah, "Failures of gradient-based deep learning", *arXiv preprint arXiv:1703.07950*, 2017.

[73] B. Lake and M. Baroni, "Still not systematic after all these years: On the compositional skills of sequence-to-sequence recurrent networks", 2018.

[74] G. Marcus, "Deep learning: A critical appraisal", *arXiv preprint arXiv:1801.00631*, 2018.

[75] A. L. Yuille and C. Liu, "Deep nets: What have they ever done for vision?", *arXiv preprint arXiv:1805.04025*, 2018.

[76] DeepMind, *Graph_nets library*, `https://github.com/deepmind/graph_nets`, Accessed: 2020-29-08.

[77] T. M. Mitchell, *The need for biases in learning generalizations*. Department of Computer Science, Laboratory for Computer Science Research, 1980.

[78] A. Sobehy, E. Renault, and P. Muhlethaler, "Position certainty propagation: A location service for manets", in *International Conference on Mobile, Secure, and Programmable Networking*, Springer, 2018, pp. 131–142.

[79] J. Liu, Y. Zhang, and F. Zhao, "Robust distributed node localization with error management", in *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*, 2006, pp. 250–261.

[80] P.-C. Chen, "A non-line-of-sight error mitigation algorithm in location estimation", in *WCNC. 1999 IEEE Wireless Communications and Networking Conference (Cat. No. 99TH8466)*, IEEE, vol. 1, 1999, pp. 316–320.

[81] A. Sobehy, E. Renault, and P. Muhlethaler, "Position certainty propagation: A localization service for ad-hoc networks", *Computers*, vol. 8, no. 1, p. 6, 2019.

[82] K. Wehrle, M. Günes, and J. Gross, *Modeling and tools for network simulation*. Springer Science & Business Media, 2010.

[83] A. Shahmansoori, *Localization with OFDM signals in 5G systems*. 2017.

[84] A. Sobehy, E. Renault, and P. Muhlethaler, "Ndr: Noise and dimensionality reduction of csi for indoor positioning using deep learning", 2019.

[85] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python", *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[86] A. Sobehy, E. Renault, and P. Muhlethaler, "Csi based indoor localization using ensemble neural networks", 2019.

[87] A. Sobehy, E. Renault, and P. Muhlethaler, "Csi-mimo: K-nearest neighbor applied to indoor localization", Jun, 2011.

[88] E. Jones, T. Oliphant, and P. Peterson, "Scipy: Open source scientific tools for python", 2001.

[89] *Ieee communication society*, `https://www.comsoc.org/`, Accessed: 2020-09-01.

[90] S. L. Brunton and J. N. Kutz, *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2019.

[91] H. J. Nussbaumer, "The fast fourier transform", in *Fast Fourier Transform and Convolution Algorithms*, Springer, 1981, pp. 80–111.

[92] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm", *IEEE Transactions on information theory*, vol. 47, no. 2, pp. 498–519, 2001.

[93] P. Pernot, "Initiation à la propagation des incertitudes", *Laboratoire de Chimie Physique, Orsay, France, présentation en date du*, vol. 2, 2013.

[94] P.-S. Laplace, *Pierre-Simon Laplace philosophical essay on probabilities: translated from the fifth french edition of 1825 with notes by the translator*. Springer Science & Business Media, 2012, vol. 13.

[95] M. HIMBERT, *Bulletin du Bureau national de métrologie*, no. 93, pp. 3–15, 1993.

[96] *International vocabulary of metrology - basic and general concepts and associated terms (vim)*, 2012.

[97] *Guide to the expression of uncertainty in measurement*, Geneva, 1993.

[98] D. Theodorou, L. Meligotsidou, S. Karavoltsos, A. Burnetas, M. Dassenakis, and M. Scoullos, "Comparison of iso-gum and monte carlo methods for the evaluation of measurement uncertainty: Application to direct cadmium measurement in water by gfaas", *Talanta*, vol. 83, no. 5, pp. 1568–1574, 2011.

[99] W. Bich, M. G. Cox, and P. M. Harris, "Evolution of the guide to the expression of uncertainty in measurement", *Metrologia*, vol. 43, no. 4, S161, 2006.

[100] C. Wang and H. Iyer, *Metrologia 42:406*, 2005.

[101] M. M. et al., *Measurement 44:1526*, 2011.

[102] C. Robert and G. Casella, *Monte Carlo statistical methods*. Springer Science & Business Media, 2013.

[103] L. Zhu and G. Gigerenzer, "Children can solve bayesian problems: The role of representation in mental computation", *Cognition*, vol. 98, no. 3, pp. 287–308, 2006.

[104] *Spam detection using bayesian filtering*, `http://web.archive.org/web/20080207010024/http://www.808multimedia.com/winnt/kernel.htm`, Accessed: 26-07-2020.

[105] A. Sobehy, "Graph-based localization with velodyne vlp-16", Master's thesis, Siemens, 2016, p. 54.

[106] A. Ng, *Machine learning course*, `https://www.coursera.org/learn/machine-learning`, Accessed: 2020-15-09.

[107] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning", *arXiv preprint arXiv:1811.03378*, 2018.

Résumé français

Ce travail comportait deux aspects principaux; le premier s'est concentré sur la résolution du problème de localisation à l'aide de méthodes de triangulation et d'apprentissage deep et classique sous différentes contraintes. La deuxième partie était principalement axée sur l'amélioration de l'aspect généralisation des modèles d'apprentissage artificielle.

Le travail présenté au chapitre 3 cible le problème de localisation dans un contexte de MANET. Les solutions supposent la connaissance de la distance entre les voisins à 1 saut avec un bruit Gaussien. Les distances peuvent être fournies à l'aide de diverses mesures telles que TOA ou RSSI. En plus de la connaissance à distance, les positions de quelques nœuds (au minimum trois) sont supposés être connus pour que la solution proposée fonctionne. Ce problème est connu dans la communauté scientifique sous le nom de “range based localization”. La solution proposée est construite avec deux critères principaux. Le premier critère est la simplicité de la solution en ce qui concerne: les périphériques requis, le coût de calcul et la bande passante du réseau. Par conséquent, la méthode proposée convient aux scénarios où le coût, l'énergie et la bande passante sont rares. Le deuxième critère est la nature distributive de la solution pour éviter le point de défaillance. La solution présentée (PCP) est basée sur la triangulation avec quelques variantes pour tenir compte de la nature du problème. La consommation d'énergie est réduite en ne nécessitant que 3 nœuds “anchor” qui pourraient être réalisés à l'aide du GPS. En outre, les calculs de triangulation sont simples par rapport à d'autres solutions basées sur l’optimisation par exemple. Enfin, la bande passante est conservée en utilisant un service de localisation “Semi-flooding based location service (SLFS)”[15] pour partager les informations de localisation avec d'autres nœuds du réseau. Cette solution a été comparée au solution “GPS-free” [19] et a montré une meilleure capacité à localiser un plus grand nombre de nœuds dans un réseau assez dense. PCP est capable de localiser environ 90% des nœuds dans un réseau avec un degré moyen de 10. Dans les travaux futurs, cette solution peut être améliorée en incorporant l'incertitude des positions estimées et en atténuant l'incertitude accrue vers le bord du réseau. Aussi, la solution pourrait être améliorée pour s'adapter à la mobilité des nœuds.

Dans le chapitre 4, nous avons abordé le problème de localisation dans une perspective d'apprentissage artificielle où un ensemble de données public est utilisé pour estimer la relation entre le CSI et la position de l'émetteur. Le CSI est mesuré pour les transmissions entre une antenne massive 2 × 8 MIMO [46] et un émetteur qui traverse une table de 2 × 4 mètres. L'ensemble de données public a été mis à la disposition de la communauté scientifique par les organisateurs de l'événement "Communication Theory Workshop" de l'IEEE pour un concours de positionnement a l'intérieur [1]. Nous avons proposé une solution d'apprentissage classique, KNN, et une solution d'apprentissage en profondeur, MLP NN, pour modéliser la relation entre le CSI et la position de l'émetteur. Avant d'appliquer les modèles, une étape essentielle de prétraitement des données est proposée, ce qui améliore le processus d'apprentissage. Tout d'abord, les composantes complexes du CSI sont étudiées pour trouver la composante la plus stable. Après l'étude statistique, on arrive à la conclusion que la composante de magnitude est la plus stable parmi d'autres. Par conséquence, la magnitude est choisie comme entrée du modèle d'apprentissage. Ceci est suivi d'une étape de réduction du bruit et de la dimensionnalité où la régression polynomiale est utilisée pour ajuster une ligne à travers les mesures de magnitude. Un nombre réduit de points est sélectionné sur la ligne d'ajustement pour être l'entrée du modèle d'apprentissage. Cette étape de prétraitement a ouvert la voie à une estimation de position très précise à l'aide de KNN et MLP NN. Notre solution a obtenu une erreur de 2,3 cm assurant la première place au concours de positionnement [1] parmi 8 équipes des meilleures universités du monde entier telles que: University of Toronto (Canada), Ruhr University Bochum (Allemagne), Heriot-Watt Université (Angleterre), Université de Padoue (Italie), Institut des réseaux IMdea (Espagne), Université d'Aalborg (Danemark) et Université Yuan Ze (Taiwan). Lors de la cérémonie d'annonce, l'organisateur a informé Abdallah Sobehy que la précision de l'estimation est meilleure que celle des auteurs de l'expérience, ce qui est un témoignage honorable.

Le chapitre 5 contient le deuxième aspect principal de notre travail qui est l'aspect de généralisation des modèles d'apprentissage artificielle. Nous avons commencé cette étude en évaluant la capacité de généralisation de nos solutions en utilisant KNN et MLP NN qui, à notre connaissance, donnent la plus grande précision sur l'ensemble de données introduit. L'évaluation impliquait la séparation entre l'ensemble d'apprentissage et l'ensemble de test

d'une manière telle que le modèle d'apprentissage devrait estimer des échantillons de test en dehors de la région d'échantillons d'apprentissage. Nous avons proposé deux modes de séparation pour évaluer les capacités d'interpolation et d'extrapolation des modèles d'apprentissage. La précision de l'estimation s'est considérablement détériorée avec l'arrangement proposé, ce qui nous a amenés à conclure que KNN et MLP NN ne parviennent pas à se généraliser. Ces expériences nous ont aidées a expliquer des raisons qui ont donné à KNN un avantage sur MLP NN dans la précision de positionnement.

La dernière partie de ce travail implique l'étude d'un problème de localisation en extérieur plus difficile [2]. Ce problème impliquait un environnement très bruyant et seule une partie de l'ensemble de données est étiquetée avec la position correspondante, ce qui introduit un nouvel ensemble de défis. Ces défis ont rendu l'aspect généralisation des modèles d'apprentissage inévitable pour obtenir une précision de positionnement décente. L'antenne qui est plus grande (8 ×)8, fournit plus de mesures qui nécessitaient une méthode plus rapide de réduction du bruit et de la dimensionnalité. En outre, l'aspect bruit élevé de l'expérience exigeait que la méthode soit beaucoup plus flexible. Nous avons proposé l'utilisation d'un raccord de ligne basé sur “Fourier Transofrm” pour répondre à ces exigences. En outre, il y avait des lectures manquantes qui ont entravé le processus d'apprentissage que nous avons atténué en remplaçant la valeur manquante par une technique de moyenne. Étant donné que seul un petit sous-ensemble de l'ensemble de données est étiqueté avec des positions, nous proposons l'utilisation de CNN pour capturer les caractéristiques relationnelles entre les paires CSI afin d'estimer la distance. Cette distance pourrait ensuite être utilisée pour estimer les positions de l'ensemble de données non étiqueté à l'aide des positions étiquetées. Notre solution permet une estimation précise de la distance avec une erreur de 10 cm entre des paires CSI distantes de 3 m.

Le problème de localisation à l'extérieur reste à résoudre car on est capable d’estimer avec précision la distance entre des échantillons à proximité. Mais on confronte des défis pour estimer la distance entre positions qui sont loins. Nous pensons qu'il y a un grand potentiel dans cette approche car elle montre une bonne capacité de généralisation en dehors de l'ensemble d'entraînement. Une méthode d'apprentissage en profondeur plus flexible qui pourrait être utilisée pour faire un pas de plus vers la solution est “Graph Neural Networks” (GNN) [53]. Il est également possible d'utiliser la solution proposée (PCP) en considérant les positions des ensembles de données étiquetées comme des nœuds d'ancrage et en utilisant les distances et la triangulation pour estimer les positions des ensembles de

données non étiquetées. Cependant, cette approche est sensible aux erreurs de grande distance.

Appendix A

Uncertainty Propagation

Most of the materials presented in this section are inspired by the "Uncertainty Propagation" Doctoral course by Pascal Pernot, Université Paris-Sud [93].

A.1 Uncertainty Definition

Typically, any experiment in a given scientific field includes measurements of some kind. In chemistry, temperature and pH of chemicals are measured. In physics, weights, masses, velocities, and other phenomena are measured. Similarly, a plethora of phenomena is estimated in biology, engineering, geology, etc. Even though the measured phenomena may differ vastly in nature and quantification tools, they all have one fundamental feature in common: *uncertainty*. Loosely we can say that uncertainty measures the fluctuation of repeated measurements of the same phenomenon. For example, consider the case of measuring the pH of a chemical substance (measurand) using a digital pH meter (measuring device). If the pH level is measured several times without changing the conditions of the experiment, we expect to get different measured values fluctuating around a certain value. This assumption is supported by the fact that on almost all measuring devices, there is an indication of the expected measurement error in the form $\pm\epsilon$. One might think that this is due to some fault in the device and thus using a more accurate device would not result in these differences. While a more accurate device would probably decrease the fluctuation effect, it can never completely eliminate it. The role of uncertainty quantification is to describe this fluctuation effect.

Uncertainty is an inherent characteristic of any measurement which quantifies the lack of our knowledge about the behaviour of the phenomenon [94]. In other words, it describes our belief of how close the measured value is to the truth value. Interestingly enough, the truth value is not calculable as we need an infinite number of repeated measurements to find it, which is of course infeasible. We can only assume a reference value based on our knowledge of the phenomenon, repetition of measurements, etc., and hope it is sufficiently close to the truth. This leads to another question, how can one compute the closeness of the measured value to the truth if the truth value itself -by definition- cannot be found? Well, there is no mystery; we can only do an estimation of uncertainty using a best-effort approach where we use available information to compare measurements to some reference value.

Let's take a step back and think, why do we get different measured values for the same phenomenon in the first place? or in other words why does uncertainty exist in all measurements? The answer is that no matter how stable the measured phenomenon is or how accurate the measurement tool is, there are always some hidden factors contributing to some kind of fluctuation of the calculated value for repeated measurements. To our eyes, the repeated measurement values may form a distribution of values around a mean (e.g. normal distribution) which, with respect to our knowledge, is considered as random error or noise. The only way to eliminate uncertainty is by having complete knowledge of all the factors affecting the measurement process and their truth values (e.g. temperature, gravity, pressure, humidity, etc.) for measuring the desired phenomenon. This is simply not achievable even for the simplest phenomena. Laplace brilliantly summarized this inevitable lack of knowledge in his philosophical essay on probability [94]:

> We may regard the present state of the universe as the effect of its past and the cause of its future. An intellect which at a certain moment would know all forces that set nature in motion, and all positions of all items of which nature is composed, if this intellect were also vast enough to submit these data to analysis, it would embrace in a single formula the movements of the greatest bodies of the universe and those of the tiniest

atom; for such an intellect nothing would be uncertain and the future just like the past would be present before its eyes.

One interesting result of this formulation of the uncertainty principle is that it is subjective in nature since it depends on our knowledge of the system. This knowledge can vary from one person to another. Take the example of the pseudo-random generator in any programming language; if the generator is set to a normal distribution with mean μ and standard deviation α, repeating the experiment gets us some apparently random values with the given mean and standard deviation. However, since a true pseudo random generator does not really exist, if we get access to the underlying function generating the -apparently- random values, we would be able to know exactly the next draw from the function and thus eliminate uncertainty completely. Probability theory is used to abstractly measure the randomness of any event including measurements. Opinions still differ on whether probability is an objective characteristic of the system or is subjective as previously explained. Independently from the philosophical view of probability, the math describing uncertainty is the same.

A.2 Sources of Errors

To summarize, a measurement is described via numerical value and a description of its uncertainty along with its unit. The uncertainty, which is our point of focus, depends on the quality of repeated experiments, the measuring device, and our knowledge of the reference value[95]. The following list entails some important definitions related to uncertainty [96]:

- **Accuracy** Difference between the measured value and the truth value which is not practically measurable by a numerical value. However, a measurement is said to be more accurate if its error is smaller.
- **Trueness** Difference between the average of infinite measurements and a reference value which is also not quantifiable as a numerical value.

- **Precision** The level of agreement between repeated measurements of the same phenomenon; commonly presented by the standard deviation.
- **Measurement Error** Difference between the measured value and the reference value. The reference value can be a standard value with negligible uncertainty.

There are two main sources of errors: systematic errors and random errors. The former is a type of error that is static or varies in a well predictable way. This can be a calibration error where a weight device is always adding a certain value to the weighted object. The latter is an error that varies randomly in an unpredictable way. The total measurement error is a summation of both error types [96]. Since the probability theory used to quantify uncertainty is related to random events, commonly uncertainty is estimated after mitigating the systematic error.

It is required to put as much effort as possible to mitigate any kind of systematic errors or any avoidable sources of uncertainty. For instance, when the measurement is repeated, the environmental conditions that are known to affect the measurement should be kept constant. Also, the experiment specification must be as precise as possible to avoid errors due to misunderstanding or ill-definition of the experiment (i.e. definitional uncertainty).

It is important to be aware of possible sources of errors whether avoidable or not. This gives a good idea about which sources are avoidable, which are negligible, and which can be partially mitigated. The following list presents some of the common error sources:

1. **Method** The measurement method includes the device's resolution, precision. Also, if a computer algorithm is involved in the measurement process, its numerical precision capability would affect the measurement value.
2. **Matter** This is related to the nature of measurand itself, its stability over time or how much is it affected by slight changes in the environment.
3. **Medium** The environment in which the experiment is made. This includes temperature, pressure, pollution, noise, etc.

4. **Human Factor** An experienced person is more likely to introduce errors in the measurements using a correct unbiased process to measure the phenomenon.

Clarifying the different sources of errors allows for proper modeling of the problem with respect to the error sources. It is worth mentioning that the phenomenon of interest can be measured directly or indirectly. In direct measurements, the target phenomenon is measured directly using a measurement tool. However, commonly, the desired phenomenon's value is not directly measurable with a device. In such case, directly measured phenomena are combined through some equation to calculate the target value. A simple example is measuring the force F acting on an object with mass m moving with acceleration a using Newton's second law: $F = ma$. Imagine that we have an accelerometer and a weight to measure a and m respectively. Since the directly measured phenomena (m and a) used to estimate the target value (F) are uncertain themselves, their uncertainties are propagated via the equation relating them to the target value.

A.3 Uncertainty Propagation

Uncertainty quantification is a principal pillar in describing almost any experiment and its results. This makes it invaluable for scientific advancement. Consequently, tremendous effort has been done by the scientific community to unify the expression of uncertainty including the 'Guide to the Expression of Uncertainty in Measurement' (GUM) and complementary works [97–99]. Assuming all possible ambiguities are mitigated, let us assume that the degree of confidence in a possible value x for phenomenon X is given by the probability $P(X = x)$. The probabilities for all values of X from the minimum possible value x_{min} to the maximum possible value x_{max} form a probability density function (PDF). In indirect measurement, the phenomenon Y can be estimated using one or more measurands X where $Y = f(X)$. The model function $f(X)$ can be in the form of:

- **Theoretical Model** used when a physical model exists that relates the input

measurands X to the target measurand Y as the given example of Newton's second law; $F = ma$.

- **Statistical Model** when there is no known accepted physical model that accurately relates the input to the output, a general model with adjustable parameters is attempted then the parameters are optimized to fit the measured data. One example is polynomial equation with degree n where $y = \theta_0 + \theta_1 + ... + \theta_n$. Another popular form of this modeling is deep learning models e.g. Multi Layer Perceptron Neural Networks and Convolutional Neural Networks.

- **Hybrid Model** A combination of the theoretical and hybrid models where the theoretical model is used with the addition of some parameters to be statistically estimated to correct the theoretical model. Consider $f(x)$ as the theoretical model, the target value y is estimated as $y = af(x) + b$ where a and b are parameters to be estimated to statistically based on available data.

The choice of model depends on many criteria that are problem dependant such as: the availability of an accurate theoretical model, computation power, required precision, time constraints etc. For any of the model types and whether the measurand is direct or indirect, the GUM [97] recommends to provide a mean value $E(X)$ and standard deviation $u(X)$ to represent the measurand and its uncertainty respectively as follows:

$$E(X) = \int_{x_{min}}^{x_{max}} xp(x)dx \tag{A.1}$$

$$u^2(X) = \int_{x_{min}}^{x_{max}} (x - E(X))^2 p(x)dx \tag{A.2}$$

In the case where there are multiple variables, the probability distribution function (PDF) is known as joint probability function or multivariate distribution. Here, the standard deviation alone is not enough to quantify the uncertainty as we need to incorporate the covariances between variables (variance-covariance matrix)

which represents the correlation between variables. For two random variables X_1 and X_2, their covariance $u^2_{X_1,X_2}$ and variance-covariance matrix are defined in equations A.3 and A.4 respectively. The variance covariance matrix for n random variables $X_1, X_2...X_n$ is a symmetric square $n \times n$ matrix where the diagonal elements are the variances $u^2_{X_1}, u^2_{X_2}...u^2_{X_n}$ and off-diagonal elements are the covariance between all pairs of variables.

$$u^2_{X_1,X_2} = E\left[(X_1 - E(X_1))\,(X_2 - E(X_2))\right] \tag{A.3}$$

$$\Sigma = \begin{pmatrix} u^2_{X_1} & u^2_{X_1,X_2} \\ u^2_{X_1,X_2} & u^2_{X_2} \end{pmatrix} \tag{A.4}$$

Consider a random variable Y that is not directly measurable but is rather estimated through a known model $Y = f(X_1, X_2...X_n)$ where X_i is a random variable with a joint probability distribution defined by $p(X)$. The expected value of Y, $E(Y)$ and its vairance $Var(Y)$ can be computed in terms of the integrals of $f(X)$ and $p(X)$ as follows:

$$E(Y) = \int f(x)p(x)dx \tag{A.5}$$

$$u^2(Y) = \int (f(x) - E(Y))^2 p(x)dx \tag{A.6}$$

The take away from equations A.5 and A.6 is that it is possible to calculate statistics of Y without having to formally describe the PDF $p(Y)$. With these definitions in hand, we can discuss the propagation of uncertainty from inputs to outputs. In the following sections, we present three methods to estimate the statistical components of Y (mean and standard deviation). The first one is known as the combination of variances where uncertainty propagation is computed for linear models. The second method is known as Monte Carlo which can be used when the

linearity constraint does not hold or when the derivations are very complex. Finally, if some model parameters are not known and we need to find the optimal given a prior and some data, the Bayesian inference is used.

A.3.1 Combination of Variances

The combination of variances method can be used in the case where the variable of interest Y is linear in the input random variables or approximately linear in the range of variation of X. Using the linearity assumption, the function $f(X)$ can be approximated to its first order taylor series around a chosen central input value $\bar{x}$ (mean of X) as shown in equation A.7 where J^T is the transposed jacobian vector composed of partial derivatives of Y with respect to each random variable in X_i; $J_i = \frac{\partial Y}{\partial X_i}$. By substituting the $f(X)$ with its first order taylor term in equation A.5 and A.6, we can derive a simplified equation for $E(Y)$ and $u^2(Y)$ in equations A.8 and A.9 respectively.

$$Y = f(X) = f(\bar{x}) + J^T(X - \bar{x}) \tag{A.7}$$

$$E(Y) = E(f(X)) = f(E(X)) \tag{A.8}$$

$$u^2(Y) = J^T \Sigma J = \sum_i \left(\frac{\partial Y}{\partial X_i}\right)_{\bar{x}}^2 u_{X_i}^2 + \sum_{i \neq j} \left(\frac{\partial Y}{\partial X_i}\right)_{\bar{x}} \left(\frac{\partial Y}{\partial X_j}\right)_{\bar{x}} u^2(X_i, X_j) \tag{A.9}$$

The second term in the variance equation A.9 is zeroed out when the input variables are statistically independent. The partial derivative terms are evaluated at the mean value of input variables $\bar{x}$.

A.3.2 Monte Carlo

When the non-linearity constraint is not a good approximation of the model, the combination of variances method would fail to describe the uncertainty propagation adequatly. This is because the first order taylor expansion (A.7) used to derive

the simplified forms of mean (A.8) and variance (A.9) does not hold anymore. Some works have been conducted to approximate the model to higher order Taylor expansions and derive the output variables' statistical information [100, 101]. These derivations become more complex as the order of Taylor's expansion becomes higher and are not applicable to complex models. Moreover, they are still subject to a similar bottle neck as that of the combination of variances when the taylor approximation is not good enough.

One method to mitigate the complex computations of the integrals of the mean (A.5) and variance (A.6) is Monte Carlo [102]. The idea is based on using the joint probability distribution of input variables and use it to generate a data set $D = (X_i, Y_i)$ by applying the model $f(X)$ on the generated input values. This results in a sample of Y that is a representation of the actual distribution. The larger the data set, the more accurate the representative set matches the actual distribution. With the representative sample in hand, the mean is simply the arithmetic mean of the representative sample and the variance can also be computed as follows:

$$E(Y) = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{A.10}$$

$$u^2(Y) = \frac{1}{n-1}\sum_{i=1}^{n} (y_i - E(Y)) \tag{A.11}$$

The idea behind the Monte Carlo method is simple but comes with some challenges. One main question is how many samples do we need in the dataset to reach a good approximation of the PDF of the output? The general objective is the convergence of mean of mean / variance when more samples are generated. This is done for batches of data samples and using some convergence threshold the generation stops when the values stabilize.

A.3.3 Bayesian Inference

The bayesian framework [103] is at the heart of probability and statistical inference. It is concerned with updating the belief (aka. posterior) that an event (aka. hypothesis) would happen given some evidence and previous belief (aka. prior). This framework is used in a tremendous number of applications including spam detection [104] and robotics localization [105]. For a hypothesis H and evidence E, the posterior probability representing the updated belief of H given E: $P(H|E)$ is computed in terms of the prior belief of H: $P(H)$ and the likelihood of E given H: $P(E|H)$ as follows:

$$\begin{aligned} P(H|E) &= \frac{P(E|H)P(H)}{P(E|H)P(H) + P(E|\neg H)P(\neg H)} \\ &= \frac{P(E|H)P(H)}{P(E)} \end{aligned} \tag{A.12}$$

In many contexts, the aim of using the Bayesian framework is to estimate some model parameters θ such that the model would best fit the evidence (data) given some prior knowledge of the model parameters. For example, the model can be a second degree polynomial $m = f(x;\theta) = \theta_0 + \theta_1 x + \theta_2 x^2$ and we are given some data composed of a number of input, output pairs $D = (x_i, y_i)$. Assume we have some prior knowledge from experience or previously used data of rough estimates of the distribution of θ for the model m: $p(\theta|m)$. The aim is to find a set of values for θ parameters given the new data D: $p(\theta|D,m)$. This is achieved by maximizing the likelihood term: $p(D|\theta,m)$ in the Bayesian equation shown in equation A.13. The process of maximizing likelihood term in the Bayesian context is known as Maximum a Posteriori estimation (MAP).

$$\begin{aligned} p(\theta|D,m) &= \frac{p(D|\theta,m)p(\theta|m)}{P(D|m)} \\ &\propto p(D|\theta,m)p(\theta|m) \end{aligned} \tag{A.13}$$

Appendix B

Machine Learning

Most of the presented materials in this appendix are collected from Andrew Ng's Machine learning course on Coursera platform [106].

B.1 Regression and Classification

The most classical form of machine learning is Supervised learning. Its essence is fitting a model that maps input data –which will refer to as features– to the desired output. The output is a dependant variable which we are interested to predict in the future depending on the application in hand. For instance, in a business context, input features could be customers' data (e.g. income, age, sex, location etc.) and the output could be whether or not he/she will buy a certain product.

To be able to build a model, the first thing we need is data. This component is usually collected from previous experiences. Again using the business use case, that would be previous marketing campaigns where customers' data and their decision to buy a product are stored in some form of a data base. For simplicity, let's assume that input data is a vector x where each row holds data of a customer while output is a vector y where each row is the corresponding decision of the customer to buy or decline the product. The next component of the model would be a hypothesis, that is the function that takes input data or features and outputs the desired output.

B.1.1 Linear Regression

The hypothesis design choice is largely dependant on the application in hand, and whether the output is continuous or discrete. When the output is continuous, the simplest form of a hypothesis function can be linear: $h(x) = \theta_0 + \theta_1 x$ where x is the input, $h(x)$ is the output and $[\theta_0, \theta_1]$ are the model parameters which the machine learning model select to optimize the matching from input to output. The process of learning in this case is known as linear regression.

To quantify how good the hypothesis function is able to map input to output variables, we introduce the cost function. Again, the cost function can vary depending on the use case but in its simplest form it is the sum of the square differences between the model estimated output $h(x)$ for each customer's input x and the actual output y. The learning model aims to minimize the cost function, hoping to get $h(x)$ to be equal to y (difference is zero) which would mean that the model is able to perfectly match input to output. Formally, the cost function can be written as:

$$c(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i)^2 \tag{B.1}$$

where m is the total number of input output pairs. Reaching a zero cost function is often not feasible and in some cases, getting too close to a zero cost function might hurt the ability of the model to generalize to future data. For simplicity, we stick with this formulation of the problem and handle extreme cases later. It is important to note that the cost function is a function of θ_0 and θ_1 not x. The learning model aims to minimize the cost function by varying θ_0 and θ_1.

Gradient descent is a heavily used method to minimize arbitrary functions which in our case is the cost function. For the gradient descent to work properly, the cost function should be differentiable. The first step is to choose initial values for θ_0 and θ_1. Starting from the initial values, gradient descent attempts to make a relatively small change to θ_0 and θ_1 so that the cost function value decreases. The step size is referred to as the learning rate, which is a parameter of the gradient descent that can be tuned. The decision to choose the direction of the step for θ_0 and θ_1 is computed

using the partial derivative of the cost function with respect to each θ. The partial derivative represents the tangent to the cost function at the current values for θ_0 and θ_1. Formally, the learning steps updates of thetas are computed by repeatedly applying the following equation:

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} c(\theta_0, \theta_1) \tag{B.2}$$

where $j = [0, 1]$ representing both thetas. The learning step equation should be applied simultaneously to θ_0 and θ_1. The value of the learning rate *alpha* is a design decision that is important to tune for a reasonable performance of gradient descent. If *alpha* is too small, gradient descent will be slow to converge. If *alpha* is too large, gradient descent might not converge and theta values overshoot away from the minimum. One of the general behavior of gradient descent is that steps tend to automatically get smaller when the cost function value is closer to the minimum as the gradient value which is multiplied by α becomes closer and closer to 0.

One of the issues of gradient descent is that it is susceptible to converge to a local minimum. However, in some cases, the cost function is a convex function (with only one optimum), the gradient descent is a reasonable choice. Also gradient descent performs well with cost function has multiple local minimums but they are all almost as small as the global minimum. Using all training data for computing the cost function at each learning step is known as batch gradient descent. One of the advantages of gradient descent is that it is computationally scalable with respect to training data as compared to other methods such as normal equations method.

It is also to be noted that the presented linear regression hypothesis function is for a one feature per sample. If there were n features, then there would be $n+1$ thetas to describe the linear regression model as follows:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n \tag{B.3}$$

For convenience, we assume that there is a feature $x_0 = 1$ that is multiplied by

θ_0 in equation B.3 to be able to write the hypothesis function in a vector form as follows:

$$h_\theta(x) = \theta^T x = \theta x_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n \tag{B.4}$$

The learning step used to solve the partial derivative term for each feature variable x_i can be written in the vector form as follows:

$$\theta_j = \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i) x_j^i \tag{B.5}$$

In the multivariate linear regression j ranges from 0 to n where n is the number of input features for each sample in the dataset. The vectorized form of the learning step is repeated iteratively until convergence. The convergence condition is another design decision which can simply be a threshold depicting the minimum change in the cost function after the last update beyond which the learning halts. It can also be a minimum value for the cost function, once it is reached the learning stops.

B.1.2 Logistic Regression

In some problems, the model aims at estimating a discrete set of values e.g. whether an image is a cat or a dog. In such cases, linear regression is not suitable. These kind of problems are known as classification problems where one of the methods to solve them is logistic regression. In binary classification, the output takes one of two values $[0, 1]$ where 0 or 1 can refer to any kind of output e.g. the image is of a cat or a dog.

The first difference to consider between hypothesis functions of linear regression and logistic regression is that logistic regression needs to output values between 0 and 1, since those are the only two valid output. A simple way to ensure this property is to use the $h_\theta(x)$ presented in equation B.4 and as an input to another

function $g(\theta^T x)$ that squishes its input value to be between 0 and 1. The sigmoid function aka. logistic function is one popular example of such squishing functions:

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}} \tag{B.6}$$

The sigmoid function asymptotically reaches zero as the value of its input approaches $-\infty$, and asymptotically reaches one as its input approaches ∞. When the input value increases from negative to positive values around zero, the sigmoid function shows a smooth transition from ≈ 0 to ≈ 1 with a value of 0.5 when its input is equal to 0. Since the sigmoid output is between 0 and 1, its output is interpreted as the probability that it belongs to one of the two classes. For example, $h_\theta(x) = 0.6$ means that the probability that the output belongs to class 1 is 0.6. More formally, $P(y = 1|x, \theta) = 0.6$. Generally, the model estimates one of the classes if its probability is greater than 0.5. The advantage of such representation is that we get a sense of the confidence the model has in its estimation.

As in linear regression, we need to determine the cost function that the model sims to minimize by varying the parameters θ. While the square difference cost function B.1 is a possible choice, it is not convex. This means that the gradient descent is not guaranteed to converge. This non-convex property is induced due to the non-linearity of the sigmoid function B.6. A cost function that conserves the convex property is known as logistic, cross entropy or negative log likelihood. It can be constructed by first establishing the cost for one sample estimation $h_\theta(x)$ and actual class y. The function is has two definitions one for $y = 0$ and another for $y = 1$ as follows:

$$cost(h_\theta(x), y) = \begin{cases} -log(h_\theta(x)) & \text{if } y = 1 \\ -log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases} \tag{B.7}$$

The idea behind this definition, is to give a cost of zero if $h_\theta(x) = y$ and a cost approaching ∞ as $h_\theta(x)$ is further from y. Note that $h_\theta(x)$ varies from 0 to 1 while

y is either 0 or 1. Equation B.7 can be written in a more compact form as follows:

$$cost(h_\theta(x), y) = -ylog(h_\theta(x)) - (1-y)log(1-h_\theta(x)) \tag{B.8}$$

This form leverages on the fact that when $y = 0$ the left term is zeroed leaving only the right term and vice versa when the $y = 1$. One important aspect of this function is that it is additive, in other words to calculate the cost over all training samples, we simply make a summation. This allows for efficient learning. The total cost over all training samples can be written as follows:

$$\begin{aligned} c(\theta) &= \frac{1}{m}\sum_{i=1}^{m} cost(h_\theta(x), y) \\ &= -\frac{1}{m}\left[\sum_{i=1}^{m} y^i log(h_\theta(x^i)) + (1-y^i)log(1-h_\theta(x^i))\right] \end{aligned} \tag{B.9}$$

The advantage of the presented cost function B.9 is its convex property. To minimize the cost, we need to do partial differentiation with respect to each θ_j in the vector θ. Having the derivative, we can then repeat the learning steps depicted in equation B.2. The learning step after the partial derivative the is written as follows:

$$\theta_j = \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^i) - y^i)x_j^i \tag{B.10}$$

Equation B.10 looks identical to B.5 of the linear regression, however, the $h_\theta(x)$ is different. Similar to linear regression, the learning step is repeated until some measure of convergence.

B.1.3 Remarks

Linear Regression and Logistic Regression are presented in their most basic form to build the foundation to introduce Neural Networks in the next section. In this section, we would like to point out to some techniques used in these methods. For instance, we only introduced Gradient Descent as the method to minimize the cost function by varying the parameters θ. There are other more complex methods that achieve the same objective when the cost function and its partial derivatives are provided (e.g. Conjugate gradient, BFGS, L-BFGS). One of the advantages of such methods is that they do not require the manual setting of the learning parameter *alpha* which, as previously mentioned, may cause the convergence to be too slow if it is smaller than it should, or shoot out of convergence if it is too large. In terms of performance, they are faster than Gradient Descent. However, they are more complex than Gradient Descent.

Another important issue usually faced in the learning context is over-fitting. This is the case when the model matches the training data very accurately but fails to do so on test data which were not seen during train. This phenomenon tends to appear when the number of parameters θ for instance when high polynomial degree cost functions are used. More parameters means more degrees of freedom to shape the hypothesis function $h_\theta(x)$ to follow $y(x)$ which leads in extreme cases to the over-fitting phenomenon. Regularization is a well-known technique to mitigate the effect of over-fitting by adding a term to the cost function with the role of inhibiting the optimization method from extremely minimizing the cost function. The idea of regularization is to counter the effect of having large number of parameters by decreasing the effect of each parameter. This can be done by adding terms to the cost function in terms of θ parameters so that having a lower values for θ yields a lower cost. Let's extend the cost functions for linear regression (B.1) and logistic regression (B.9) respectively by adding the regularization term to each of them as follows:

$$c(\theta) = \frac{1}{2m}\left(\sum_{i=1}^{m}(h_\theta(x^i) - y^i)^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right) \tag{B.11}$$

$$c(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^i log(h_\theta(x^i)) + (1-y^i)log(1-h_\theta(x^i))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2 \tag{B.12}$$

The regularization term: $\lambda\sum_{j=1}^{n}\theta_j^2$ incites the optimizer to decrease the values of θ which neutralizes the overfitting effect coming from higher values of θ when only the original cost function is minimized. Here we also introduce a new parameter λ which is another design choice to quantify the effect of regularization. Caution is important when selecting this parameter because while we pitched the use of regularization to avoid over-fitting, it is important to point out that too much regularization (setting a very high value for λ) would lead to under fitting. This means that the the regularization term becomes too dominant over the original cost function term and thus the optimizer sets very small values to θ and thus diminishes all degrees of freedom the hypothesis function originally had.

Adding the regularization term to the cost function in both the linear regression and logistic regression requires recalculation of the partial derivative for the optimizer to minimize the new cost function. We refer the reader to [106] for the derivation of the derivative. When the derivative is computed the learning steps are exactly as previously explained.

B.2 Multi Layer Perceptron Neural Networks

B.2.1 Motivation

The linear hypothesis model can only represent simple relations between input and output variables. In other words, when the output is directly or inversely

proportional to the input data which is not always the case. In most use cases, the relations are more complex than being linear which motivates the introduction of non-linear models. A simple way to introduce non-linearity in the hypothesis function is by using a hypothesis function with a polynomial degree higher than one. Assuming only two input variables x_1 and x_2 and a polynomial degree of two for simplicity the hypothesis can be written as follows:

$$h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 + \theta_5 x_2^2 \tag{B.13}$$

A notable difference between linear hypothesis and higher degree polynomial hypotheses is the number of parameters (thetas) with respect to the number of features n. In a first degree polynomial the number of thetas $\approx n$. However, in higher degree polynomials the number of theta parameters grows much faster. With just the second or third degree polynomials, the number of thetas grows in the order of n^2 and n^3 respectively. This introduces two issues especially that in most cases $n >> 2$; first the large number of tune-able theta parameters would probably lead to over-fitting of data which inhibits the model's ability to generalize well to future data. Second, the computation demand becomes very high for this large number of thetas.

Neural Networks come in to find a good balance by introducing non-linearity with a controlled increase of tune-able parameters. They were originally introduced in the 80s in an attempt to mimic the behaviour or the thought of flow in the brain. Back in the days, they did not receive much attention because of the lack of computational resources to handle their complexities. However, with moore's law surviving the test of time, machines have come a long way since then and in the past decade they were able to train deep neural networks in a reasonable amount of time for multitude of applications. In addition, there has been a massive research effort to make NNs more effecient and memory friendly. Thus, NNs became a principal building block for performant machine learning solutions.

B.2.2 Structure and Feed Forward

The way NNs mimic brain's neurons is by being composed of units corresponding to brain neurons. The neurons are organized in a form of vertical sequential layers. The first layer receives its input from the input features x while subsequent layers receive their input from previous layers. The output of the last layer is the model output which we referred to previously as $h(x)$. The input features and the last output are referred to as input and output layers respectively while rest of the layers are called hidden layers. In a fully connected NN, each neuron (aka. unit) receives an input from each neuron in the previous layer. Each neuron is assigned a tune-able parameter vector θ which is multiplied by the input vector to produce the neuron's output. The tune-able parameters in NN context are usually referred to as weights. In addition to weights there is one bias parameter that is added to the vector multiplication output. For convenience of notation in the previous section, an additional input feature x_0 is added to the input so that θ_0 would represent the bias. Since there are multiple nodes per layer, the theta parameter space will be represented by a matrix Θ with dimensions $(u \times m)$ where u is the number of units in the layer in question and m is the number of inputs from the previous layer. The result of the vector multiplication at each neuron is passed through an activation function which kind of normalizes the computed output and introduces the non-linearity property. Examples of these functions are sigmoid, relu, selu [107]. For the time being, we will denote the activation function by $act(v)$ where v is the vector multiplication result at a given unit. For units from 1 to u in the first hidden layer with m nodes in the previous input layer (input features) their output is computed as follows:

$$\begin{aligned} a_1 &= act(\Theta_{10}x_0 + \Theta_{11}x_1 + \Theta_{12}x_2 + ... + \Theta_{1m}x_m) \\ a_2 &= act(\Theta_{20}x_0 + \Theta_{21}x_1 + \Theta_{22}x_2 + ... + \Theta_{2m}x_m) \\ &\vdots \\ a_u &= act(\Theta_{u0}x_0 + \Theta_{u1}x_1 + \Theta_{u2}x_2 + ... + \Theta_{um}x_m) \end{aligned} \tag{B.14}$$

For subsequent layers, x is replaced by the activation outputs $[a_1, a_2, ..., a_u]$. For simplicity, the output of all units of a layer l can be expressed in a more compact form using the Θ matrix multiplied by the output vector of the previous layer a^{l-1} as follows:

$$a^l = act(\Theta^l a^{l-1}) \tag{B.15}$$

One should keep in mind that as we added an extra feature $x_0 = 1$ to the input vector to represent the bias, the same is done for all hidden layers by adding a neuron with $a_0 = 1$. The process of computing the output from one layer and sending its output to the subsequent layer until reaching the output layer is known as forward propagation. Since the processes of the forward propagation are composed of series of matrix multiplications and additions, the computations are efficiently calculated on CPUs and GPUs which gives NNs an edge over other methods. Also, this efficient way introduces non-linearity which suits a multitude of problems. It is also worthy to note that the choice of the number of layers as well as the number of units at each layer is a design choice which is referred to as the network architecture.

B.2.3 Learning and Backward Propagation

Now that we have the basis for the neural network, we assume that an architecture of the neural network is chosen with L layers and the number of units at each layers is s_l. The cost function for the neural network is a generalization of the logistic regression cost function shown in equation B.11. In fact, a neural network architecture with only one input layer connected to the output layer with one unit operates exactly like logistic regression. The complexity and power of NNs appear when it has more hidden layers which is known as Deep Neural Networks (DNNs). To accommodate for the number of layers L, the number of units at each layer s_l and the number of classes K (the number of units in the output layer), the cost function is written as follows:

$$c(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^i log(h_\Theta(x^i))_k + (1-y_k^i)log(1-h_\Theta(x^i))_k\right] + \frac{\lambda}{2m}\sum_{l=2}^{L}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l-1}}(\Theta_{i,j}^l)^2 \tag{B.16}$$

Here, the subscript k indicates the kth output at the output layer. In the binary classification context where there is only one output unit at the output layer, summing over k would not have been necessary but for the sake of generality it is indicated. Note that the summation for the regularization parameters in the rightmost term is from layer 2 because there are no weight parameters for layer 1 (the input layer) because these are the raw input data which are not operated upon. We need to provide the cost function along with partial derivatives with respect to each parameter $\theta_{i,j}^l$ to minimize the cost. The hypothesis function output is reached through the feed-forward process. At each layer there is a set of parameters (weights and biases) stored in a form of a matrix $\Theta_{i,j}^l$ where l is the layer, i is the unit in the layer and j finally defines the weight associated with the output of the jth unit in the previous layer. The partial derivatives for all these parameters can be computed sequentially starting from the output layer and moving backwards one layer after the other until the first hidden layer. Note that the first layer (input layer) does not have a parameter matrix associated with it since its output is the raw input data. This process is known as backward propagation

The output of unit j in layer l is denoted by a_j^l. The starting point of the backward propagation is the output layer. To simplify the process we first present the derivative derivation using only one training sample. We start by first computing the error δ^L at the output layer L between the prediction a^L and the actual value y. Note that a^L is also the hypothesis function output $h_\Theta(x)$ because it is the output at the last layer. Note also that the that a^l is a vector of length equal to the number of units at layer l. This error simply the difference between the layer output and the actual values as depicted in the following equation:

$$\delta^L = a^L - y \tag{B.17}$$

The error for prior layers $[L-1, L-2..., 1]$ is computed using the output of the error of subsequent layer. For instance the second layer to be processed by back propagation is the last hidden layer $L-1$ (1 layer before the output layer). The error calculation uses the error of the last layer, the Θ matrix of the last layer and the output of the last hidden layer itself a^{L-1}. The derivative of the activation function $act'(x)$ is also required. The error calculation for the hidden layers is computed sequentially in the following set of equations where the $.*$ operator denotes element wise multiplication.

$$\begin{aligned}
\delta^{L-1} &= (\Theta^{L})^{T}\delta^{L} .* act'(\Theta^{L}a^{L-1}) \\
\delta^{L-2} &= (\Theta^{L-1})^{T}\delta^{L-1} .* act'(\Theta^{L-1}a^{L-2}) \\
&\vdots \\
\delta^{2} &= (\Theta^{3})^{T}\delta^{3} .* act'(\Theta^{3}a^{2})
\end{aligned} \tag{B.18}$$

The partial derivatives are easily computed from the errors (ignoring the regularization terms for simplicity) as follows:

$$\frac{\partial}{\partial \Theta^{l}_{i,j}} c(\Theta) = a^{l}_{j}\delta^{l+1}_{i} \tag{B.19}$$

With partial derivatives and the cost function, NN weights and biases can be updated until convergence as previously explained. Using training examples one by one to update parameters is known as stochastic gradient descent. It is computationally faster than batch gradient descent where the whole dataset is collectively collectively for one learning step towards the cost function minimum. However, accuracy is sacrificed for the gain in computational speed. A mid-way between those two extremes is known as mini batch gradient descent is when a subset of the training set is collectively used to update the model parameters.

B.2.4 Remarks

We presented a basic overview of NNs to provide a decent understanding of this powerful tool for the reader. However, many aspects are left out since the focus of this work is not to improve the NNs' fundamental operations but rather to make the best use of them for the localization problem. However, we would like to point out briefly some aspects to be considered when working with NNs.

There are many activation functions in the literature and their choice affects the performance of NNs. For instance, the presented sigmoid function is susceptible to the issue known as vanishing gradient. This issue appears when computing partial derivatives where the absolute input to the activation function $act(x)$ is very large (approaching ∞ or $-\infty$). In these extreme values of input the the sigmoid curve is almost horizontally flat, meaning that the gradient ≈ 0 which in return inhibits the ability of gradient descent to give any direction for the parameters to update. Also, it is worth mentioning that the sigmoid function works with binary classification. When there are more classes a generalization of the sigmoid is the softmax function. The softmax ensures that the summation of the outputs for all classes is equal to 1 and thus maintains the probabilistic sense that was introduced with the sigmoid. The softmax can be expressed as follows:

$$sig(x)_j = \frac{e^{x_j}}{\sum_{j=1}^{k} e^{x_j}} \tag{B.20}$$

One of the most interesting theorems that manifest the power of NNs is the universal approximation theorem. It states that any continuous function with any input dimension d with domains $[0, 1]$ and a real value output, there exists a sigmoid based NN with 1 hidden layer that approximates this function with error ϵ. Moreover, *epsilon* can be any positive real number no matter how small. Keep in mind that this is possible with only one hidden layer. Since more hidden layers adds more degrees of freedom to the NN and thus increases its ability to approximate function, the power of deep NNs are expected to be immense.