THESIS APPROVAL FORM

Student Name:  Farhan Al Ghareeb       I.D. #: 200603406

Thesis Title:  Open Space Indoor Navigation with RSS Sequence Transduction Neural Networks

Program:  MSE in Computer Engineering

Department:  Computer Engineering

School:  Engineering

The undersigned certify that they have examined the final electronic copy of this thesis and approved it in Partial Fulfillment of the requirements for the degree of:

MSE  in the major of  Computer Engineering

Thesis Advisor’s Name:  Dr. Samer Saab

Signature:  
Date:  18 / 06 / 2020

Committee Member’s Name:  Dr. Zahi Nakad

Signature:  
Date:  18 / 06 / 2020

Committee Member’s Name:  Dr. Wissam Fawaz

Signature:  
Date:  18 / 06 / 2020
By signing and submitting this license, you (the author(s) or copyright owner) grants the Lebanese American University (LAU) the non-exclusive right to reproduce, translate (as defined below), and/or distribute your submission (including the abstract) worldwide in print and electronic formats and in any medium, including but not limited to audio or video. You agree that LAU may, without changing the content, translate the submission to any medium or format for the purpose of preservation. You also agree that LAU may keep more than one copy of this submission for purposes of security, backup and preservation. You represent that the submission is your original work, and that you have the right to grant the rights contained in this license. You also represent that your submission does not, to the best of your knowledge, infringe upon anyone’s copyright. If the submission contains material for which you do not hold copyright, you represent that you have obtained the unrestricted permission of the copyright owner to grant LAU the rights required by this license, and that such third-party owned material is clearly identified and acknowledged within the text or content of the submission. IF THE SUBMISSION IS BASED UPON WORK THAT HAS BEEN SPONSORED OR SUPPORTED BY AN AGENCY OR ORGANIZATION OTHER THAN LAU, YOU REPRESENT THAT YOU HAVE FULFILLED ANY RIGHT OF REVIEW OR OTHER OBLIGATIONS REQUIRED BY SUCH CONTRACT OR AGREEMENT. LAU will clearly identify your name(s) as the author(s) or owner(s) of the submission, and will not make any alteration, other than as allowed by this license, to your submission.

Name: Farhan Al Ghareeb

Signature: ___________________________ Date: 18 / 06 / 2020

Day Month Year
PLAGIARISM POLICY COMPLIANCE STATEMENT

I certify that:

1. I have read and understood LAU’s Plagiarism Policy.
2. I understand that failure to comply with this Policy can lead to academic and disciplinary actions against me.
3. This work is substantially my own, and to the extent that any part of this work is not my own I have indicated that by acknowledging its sources.

Name: Farhan Al Ghareeb

Signature: __________________________ Date: 18 / 06 / 2020
ACKNOWLEDGMENT

I would like to take this opportunity to thank everyone who supported me in this work whom without their support this wouldn’t have been accomplished.

Many thanks to all ECE faculty members for their valuable contribution and support throughout this work. Special thanks to my advisor Dr. Samer Saab, for his motivation and follow up on my progress. Thanks for giving ideas and valuable advices that helped in developing the project and shaping it to get better results.

With deep gratitude I thank my lovely parents and family for all the precious support, for the long nights, for the prayers, and for the constant moral support and encouragement.

Many thanks to those whose presence and motivation made a big difference and added a big value to my work. First, to a brother and a best friend that I am forever grateful, Sari Malaeb many thanks for partaking part of the enormous stress I passed through and giving valuable technical assistance. Also, thanks to all best friends and brothers for their much-appreciated support especially William, Hussein, Wael, Bahaa, Ayoub, Ehab, Alaa, and many others.

A big thank you to a best friend, sister, and motivator, Lillian, I am forever indebted to your support, motivation, and prayers.

Also, thanks to the LAU family who provided technical and logistic assistance in using LAU resources, especially Ms. Rana Awada and Mr. Saad El Zein.

Last but certainly not least, thanks God for giving me the courage and patience to continue, for granting me the capability to proceed and finish this research, and for blessing me with such a lovely family and friends whom I am forever indebted.
Open Space Indoor Navigation with RSS Sequence Transduction Neural Networks

Farhan Baheej Al Ghareeb

ABSTRACT

In conjunction to the extensive research conducted in outdoor positioning and navigation systems, indoor localization and navigation techniques have been gaining interest and attention with various technologies proposed in an attempt to achieve high positioning accuracy. Out of many proposed techniques, the received signal strength (RSS) fingerprinting-based approach remains widely adopted for indoor localization. One of the key challenges facing this approach is navigating an open space indoor location where the signal strength may vary indifferently from one position to the subsequent one. In this work, we propose a transduction neural network approach that takes as an input a sequence of past RSS fingerprints and accordingly predicts the current location, navigates the movement, and even predicts the future positions. The proposed models are based on Gated-Recurrent Unit (GRU) and Long-Short Term Memory (LSTM) recurrent neural networks (RNN). Moreover, we compare the accuracy of the proposed transduction neural networks to a non-recurrent neural network, then we evaluate the performance of all suggested models in a realistic environment with only WLAN RSS fingerprints, Cellular RSS Fingerprints, and both WLAN/Cellular RSS fingerprints. Our experimental results show that the LSTM-based architecture achieves accuracy of about one meter in an open area of 130m² while only using Wi-Fi fingerprints.

Keywords: Neural Networks, Recurrent Neural Networks, Transduction Neural Networks, Indoor Navigation, Localization, Prediction, Received
Signal Strength (RSS), RSS Fingerprinting, Long-Short Term Memory (LSTM), Gated Recurrent Unit (GRU)
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I- Literature Review</td>
<td>1</td>
</tr>
<tr>
<td>II- Introduction</td>
<td>7</td>
</tr>
<tr>
<td>III- Background</td>
<td>10</td>
</tr>
<tr>
<td>3.1 Uniqueness of The RSS Signature</td>
<td>10</td>
</tr>
<tr>
<td>3.2 Neural Networks Fundamentals</td>
<td>12</td>
</tr>
<tr>
<td>IV- Proposed Scheme and Methodology</td>
<td>16</td>
</tr>
<tr>
<td>4.1 Phone Application for Data Collection</td>
<td>17</td>
</tr>
<tr>
<td>4.2 Site Inspection and Data Collection</td>
<td>23</td>
</tr>
<tr>
<td>4.3 Neural Network Approach for Indoor Navigation</td>
<td>27</td>
</tr>
<tr>
<td>4.3.1 Basic Neural Network</td>
<td>30</td>
</tr>
<tr>
<td>4.3.2 Long-Short Term Memory (LSTM) Neural Network</td>
<td>34</td>
</tr>
<tr>
<td>4.3.3 Gated Recurrent Unit (GRU) Neural Network</td>
<td>41</td>
</tr>
<tr>
<td>4.3.4 Experimental Results and Analysis</td>
<td>47</td>
</tr>
<tr>
<td>V- Conclusion</td>
<td>64</td>
</tr>
<tr>
<td>References</td>
<td>65</td>
</tr>
</tbody>
</table>
Table of Figures

Figure 1: LSTM vs GRU Gate Level (Phi) ............................................................... 13
Figure 2: LSTM Forget Gate ("Understanding Lstm Networks") ......................... 13
Figure 3: LSTM Input Gate ("Understanding Lstm Networks") ......................... 14
Figure 4: LSTM Cell State ("Understanding Lstm Networks") .......................... 14
Figure 5: LSTM Output Gate ("Understanding Lstm Networks") ...................... 14
Figure 6: GRU Gate Level (Muccino) ............................................................... 15
Figure 7: Android application manifest controlling the permissions granted to the application and the runtime of the main activity ........................................ 18
Figure 8: Phone Application Interface developed in this work .......................... 18
Figure 9: Route tracking button to create directories for saving RSS values ....... 19
Figure 10: Code behind the RSS button ............................................................ 19
Figure 11: Snapshot of code for detecting cellular stations with their cell IDs ...... 19
Figure 12: Snapshot of a code detecting the type of cellular neighbor cells ....... 20
Figure 13: Snapshot of a code detecting Wi-Fi APs with their RSS values ......... 21
Figure 14: Phone App sample output ............................................................... 21
Figure 15: Sample CSV output saved at each step ......................................... 22
Figure 16: Floorplan of the Surveying Site in Irwin Hall Level 7 ...................... 23
Figure 17: First hallway (44m2) for data collection ........................................ 24
Figure 18: Second hallway (77m2) for data collection ..................................... 24
Figure 19: All RSS Values(dBm) with their corresponding (x, y) Coordinates ..... 26
Figure 20: Basic Neural Network Representation ........................................... 27
Figure 21: Flowchart of the proposed scheme/methodology ........................... 28
Figure 22: Basic neural network structure simulating the proposed NN but with half the number of neurons in the hidden layers for representation purposes ....... 30
Figure 23: Sample of the basic neural network training/validation loss .......... 31
Figure 24: Sample prediction - Error calculation ............................................ 31
Figure 25: Basic NN regression model response Expected vs Prediction values ... 32
Figure 26: CDF of Mean Error for Basic Neural Network .............................. 33
Figure 27: LSTM/GRU input blocks ............................................................... 34
Figure 28: LSTM Sample Training vs Validation loss (x-abscissa) ................... 35
Figure 29: LSTM Sample Training vs Validation loss (y-ordinate) ................. 36
Figure 30: LSTM Sample X prediction True vs Predicted future ...................... 36
Figure 31: LSTM Sample Y prediction True vs Predicted future ...................... 37
Figure 32: LSTM sample response Expected vs Prediction 5-steps X values ....... 37
Figure 33: LSTM sample response Expected vs Prediction 5-steps Y values ....... 38
Figure 34: LSTM sample response Expected vs Prediction 1-step X values ....... 38
Figure 35: LSTM sample response Expected vs Prediction 1-step Y values ....... 39
Figure 36: LSTM sample CDF showing the 5 forecasted steps ....................... 39
Figure 37: CDF of Mean Error for LSTM Recurrent Neural Network .............. 40
Figure 38: GRU Sample Training vs Validation loss (x-abscissa) ................. 41
Figure 39: GRU Sample Training vs Validation loss (y-ordinate) ................... 41
Figure 40: GRU Sample X prediction True vs Predicted future ...................... 42
Figure 41: GRU Sample Y prediction True vs Predicted future ...................... 42
Figure 42: GRU sample response Expected vs Prediction 5-steps X values ....... 43
Figure 43: GRU sample response Expected vs Prediction 5-steps Y values ....... 43
Figure 44: GRU sample response Expected vs Prediction 1-step x-values ....... 44
Figure 45: GRU sample response Expected vs Prediction 1-step Y values ....... 44
Figure 46: GRU sample CDF showing the 5 forecasted steps ........................................ 44
Figure 47: CDF of Mean Error for GRU Recurrent Neural Network .............................. 46
Figure 48: Cellular-based Models Comparison ............................................................. 48
Figure 49: Wi-Fi based Models Comparison .................................................................. 49
Figure 50: Wi-Fi/Cellular-based Models Comparison .................................................. 49
Figure 51: Testing Heterogeneous LSTM X prediction ............................................... 51
Figure 52: Testing Heterogeneous LSTM Y prediction ............................................... 51
Figure 53: Route prediction with heterogeneous LSTM .............................................. 52
Figure 54: Testing Wi-Fi LSTM X prediction ............................................................... 52
Figure 55: Testing Wi-Fi LSTM Y prediction ............................................................... 53
Figure 56: Route prediction with Wi-Fi based LSTM .................................................. 54
Figure 57: Projecting model prediction to real-time environment ................................ 54
Figure 58: CDF of Mean Error for LSTM RNN current location prediction ............... 56
Figure 59: CDF of Mean Error for GRU RNN current location prediction ................. 56
Figure 60: CDF Comparison LSTM vs GRU for current location prediction ............. 56
Figure 61: CDF Comparison LSTM WiFi vs GRU WiFi for 1-step prediction ............. 57
Figure 62: CDF Comparison LSTM 4G vs GRU 4G for current location prediction .... 57
Figure 63: CDF Comparison 1-step vs 5-steps LSTM ................................................. 59
Figure 64: CDF Comparison 1-step vs 5-steps LSTM WiFi ....................................... 59
Figure 65: CDF Comparison 1-step vs 5-steps LSTM Cellular .................................. 59
Figure 66: CDF Comparison 1-step vs 5-steps GRU .................................................... 60
Figure 67: CDF Comparison 1-step vs 5-steps GRU WiFi .......................................... 60
Figure 68: CDF Comparison 1-step vs 5-steps GRU Cellular ..................................... 60
Figure 69: 1-step Route prediction with Wi-Fi based LSTM ....................................... 62
Figure 70: Projecting model prediction to real-time environment 1-step ................. 62
Table of Tables:

Table 1: Results of Basic Neural Network (Cellular/Wi-Fi) ........................................... 32
Table 2: 5 step forecasting mean error of LSTM sample ............................................... 40
Table 3: Results of LSTM Neural Network (Cellular/Wi-Fi)........................................ 40
Table 4: 5-step forecasting mean error of GRU sample ................................................. 45
Table 5: Results of GRU Neural Network (Cellular/Wi-Fi)........................................ 45
Table 6: Overall results for all neural network models.................................................. 47
Table 7: Testing results of 2 LSTM models taking only the first step............................. 50
Table 8: LSTM Model Testing with 5-steps Forecasting .............................................. 50
Table 9: Comparing all proposed models with the current location prediction.............. 55
Table 10: Comparison of 5-steps vs 1-step prediction.................................................. 58
Table 11: 1-step LSTM Testing (WiFi model vs Heterogeneous model)....................... 61
Chapter One

Literature Review

Outdoor navigation and localization have long been an objective of the research community, where GPS is still the most common technology followed for outdoor navigation. As a matter of fact, a recent study (Orabi et al.) proposes a machine learning approach to improve the GPS code phase estimate in multipath environments. With an equally interesting and challenging domain, many indoor localization techniques and solutions were proposed to get better accuracy in terms of indoors localization, positioning, and navigation. For the past two decades many techniques tackled the issue of indoor localization with various solutions involving Bluetooth, RFID, ultrasound, infrared technologies (Otsason et al.), triangulation, angulation, Time of Arrival (TOA), Time Difference of Arrival (TDOA), Angle of Arrival (AOA), Phase of Arrival (POA), RSS-based fingerprinting methods; cellular based GSM/WCDMA RSS, WLAN based (Liu et al.) and others techniques.

In what follows we provide brief explanations of some of the latter techniques while focusing on RSS-based and neural network approaches in a way that relates to the presented work.

Outdoors GPS based solutions are broadly adopted for outdoor navigation with line-of-sight (LOS) transmission paths. However, for indoor environments GPS and GSM systems become unreliable where the satellite and cellular signals become broken indoors resulting in deep shadowing (Yassin et al.), multipath effect, fading, and delay distortion (Wang et al.). In the absence of LOS, multipath effect is due to the building geometry, reflection, diffraction, human body absorption, obstruction/blockage and interference of neighboring devices, and dynamic nature environments (Ahmad et al.) (Nerguizian, Despins and Affes). With the limitations of GPS based solutions, the WiFi techniques became more favorable since many infrastructures already support WLAN in its indoor spaces. Some techniques that exploits frequency diversity via frequency hopping (Chen et al.) for example achieved a centimeter localization accuracy.

Localization techniques can be roughly classified into two categories: Triangulation-Trilateration and fingerprinting. Triangulation techniques are
used to estimate the position of the user given the angle between the transmitter and the main axis of the receiver's antenna (Khalife, Kassas and Saab) while Trilateration methods are used to estimate the position of the receiver given the distances from the receiver to each of the transmitters. In order to compute distances and angles, methods such as TOA, TDOA, POA, and AOA are used (Khalife, Kassas and Saab). Fingerprinting is a cheaper and simpler technique than triangulation and is derived from received signal strength (RSS) measurements (Khalife, Kassas and Saab), but it does require much more work mainly in the process of building the fingerprints map.

Within the Time of Arrival (TOA) approach, the time needed by the signal to travel from the user to the anchor node (AN) is calculated. The user is then localized to a circle centered on the AN and with a radius $d$ that is estimated through the TOA. In order to detect the exact location of the user, at least three ANs are required. In this case, the estimated position of the user is simply within the region of intersection of the three circles (Yassin et al.).

Time Difference of Arrival (TDOA) examines the time difference at which the signal arrives at many measuring units where measurements are taken between multiple pairs of reference points with known locations. The transmitter must lie on a hyperboloid for each measurement with a constant range difference between the two measuring units. The location to be estimated is the intersection of many hyperbolic curves (Yassin et al.). Angle of Arrival (AOA) technique calculates the angle at which the signal arrives from the user to the anchor nodes, then, the region where the user could exist can be drawn. This region is represented by a line that has a certain angle with the ANs (Yassin et al.).

Radio Frequency Identification (RFID) was proposed as well (Saab and Nakad; Saab and Msheik) to estimate the position and orientation of an object indoors. Based on power map matching algorithm, (Saab and Msheik) proposes a scheme based on power map matching algorithm that is capable of estimating the location and orientation of objects. Furthermore, (Saab and Nakad) presented a standalone indoor positioning system (IPS) with RFID reading RSS from passive tags. The RSSI has three independent variables: distance $d$, elevation angle $\theta$, and the azimuth angle $\phi$ between the tag and the reader antennas. The measured signal level model adopted in the latter work is entirely based on the average received signal, decreasing logarithmically with distance and log-normal.
shadowing (Saab and Nakad). Using a different approach, (S. S. Saab and K. K. Saab) proposes a novel light-emitting diode (LED)-based system that can estimate the 3-D position of an object from the measurements of received signal strength (RSS). Being a relatively cheap approach for remotely estimating the location of an object, Visible Light Communication (VLC) has proven itself to be more accurate, even down to the orders of a couple centimeters and tens of centimeters(S. S. Saab and K. K. Saab). The RSS approaches roughly include two main methods: the path loss lognormal shadowing model to deduce a trilateration, and the RSS fingerprinting (Yassin et al.).

GSM Indoor localization systems with deep shadowing effects proposed in (Otsason et al.) achieved a median accuracy of 5 meters in large multi-floor buildings using wide signal-strength fingerprints. In (Ahmadi, Viani and Bouallegue), an innovative target tracking algorithm which combines learning regression tree approach and filtering methods using RSSI metric is proposed. Using Wireless Sensor Networks, Regression Tree algorithm is investigated in order to estimate the position using the RSSI (Ahmadi, Viani and Bouallegue). Another approach (Ahmadi and Bouallegue) proposes a study on well-known localization algorithms which introduce machine learning and RSSI fingerprint; namely a learning regression tree (RT) based method for indoor localization using WSN. In (Xue et al.), a gaussian process regression (GPR) localization model that is trained to learn the interrelated connection between the fingerprints and the locations is proposed. MEGPR WiFi localization system is composed of three components. First, the selection of an informative access point (AP) set which is a priority in the proposed indoor localization system. Second, to obtain an accurate location estimation model, the localization residual is fed to the model to enhance the localization system. Third, large errors are mitigated by modifying the predicted locations (Xue et al.). Relying on indoors RSS WiFi fingerprinting, and in a way to identify and minimize the localization error, (Wu et al.) revisits the received signal strength (RSS) fingerprint-based localization scheme and reveals crucial observations that act as the root causes of localization errors. Similarly, in (Khalife, Kassas and Saab), a power map consisting of a set of RSS values associated with a corresponding set of (x, y) coordinates in the space is built in an offline stage. In the online stage, the position of the object is estimated by choosing the (x, y) coordinates that minimize the error.
between the measured RSS and the pre-mapped RSS values. For such
distance dependent algorithms, several techniques have been proposed,
such as probabilistic methods, K-nearest neighbor (KNN), neural networks,
support vector machine (SVM), and smallest M-vertex polygon (SMP).
(Khalife, Kassas and Saab) presents an indoor positioning technique that
combines trilateration and fingerprinting coupled with an extended Kalman
filter (EKF). As introduced above, an approach proposed in (Yassin et al.) is
used to estimate the distance between the serving base station (BS) and the
user device (UD) based on a path loss lognormal shadowing model. Then,
trilateration is used to estimate the location of the UD using at least 3 serving
BSs. RSS-based fingerprinting firstly collects RSS fingerprints then estimates
the location of the UD by matching online measurements with the closest
possible location that corresponds to measurements in a database (Yassin et
al.).

The idea of using WLAN RSS fingerprints for indoors localization with neural
networks was proposed long time ago with “Modular Multi-Layer Perceptron”
in 2006 (Ahmad et al.). Deploying an artificial neural network (ANN), another
method for mobile station localization uses narrowband channel
measurement results applied to an ANN (Nerguizian, Despins and Affes). A
deep learning based indoor fingerprinting system using channel state
information (CSI) was proposed in (Wang et al.) where deep learning is
utilized to train all the weights of a deep network as fingerprints. Then, a
greedy learning algorithm is used to train the weights layer by layer in an
offline phase. In the online localization phase, a probabilistic method based
on the radial basis function is used to obtain the estimated location (Wang et
al.). A machine learning approach for localization in cellular environments
was proposed in (Abdallah, Saab and Kassas) where the proposed
localization scheme integrates a weighted K-nearest neighbor (WKNN) and a
multi-layer neural network. The integration takes advantage of the robust
clustering ability of WKNN and implements a neural network that could
estimate the position within each cluster (Abdallah, Saab and Kassas).

Another recent study proposes a neural network approach for indoor
fingerprinting-based localization (Jaafar and Saab) where the solution
handles indoor localization with cellular and Wi-Fi RSS combined together
with a basic neural network. The proposed solution in this work approaches a
similar environment with hybrid RSS (Cellular and WiFi) for indoor navigation
with larger open space and using transduction neural networks instead of basic neural networks. The first study on neural networks for indoor localization was reported by Battiti et al. (Battiti, Villani and Le Nhat) and Brunato and Battiti (Brunato and Battiti) back in 2002 and 2005. In 2012, instead of employing a basic feed-forward neural networks, (Graves) uses Recurrent Neural Networks (RNN) that have a powerful sequence learning architecture. The combination of a high-dimensional multivariate internal state and nonlinear state-to-state dynamics offers more expressive power than conventional sequential algorithms (Graves). RNN can be used for hand-writing recognition, text generation, language translation and modeling and time series forecasting. Back in 2012 (Graves) proposed a Long-Short Term Memory (LSTM) RNN for speech/phoneme recognition in a way that demonstrates its ability to integrate acoustic and linguistic information during a speech recognition task. (Jang, Shin and Choi) proposes an indoor localization technique that uses magnetometer sensor readings as input to the artificial neural network models where the RNN can characterize a particular location based on the current input as well as the past sequence of inputs. (Jang, Shin and Choi) uses geomagnetic field signal, which is quite stable in time domain as opposed to RF signal. The position is determined based not only on the current geomagnetic field value, but also on the past geomagnetic field values. The scheme proposes a RNN model that is trained to memorize the geomagnetic field map of indoor environment, and uses the model trained in an offline stage to predict the current location of the pedestrian in an online phase at a later time (Jang, Shin and Choi).

RNN is often used in situations where input and output data have a continuous relationship with time, and projecting the latter to indoor navigation, a recent study (Hoang et al.) proposes a RNN approach for Wi-Fi indoor localization with RSS fingerprinting. Going quickly over the techniques used with RSS fingerprints, the K-nearest neighbors (KNNs) determines the user’s location by calculating the fingerprint distance measured at the unknown point and the reference locations in the database. Support vector machine (SVM) provides a direct mapping from RSSI values collected at the mobile devices to the estimated locations through nonlinear regression by supervised classification technique. Stochastic filter estimates the most likely current location based on prior measurements, assuming a Gaussian noise
of the RSSI and linear motion of the detecting object (Khalife, Kassas and Saab) (K. K. Saab and S. S. Saab).

Back to RNN, the most popular RNN cell designs are the Long-Short Term Memory (LSTM) and Gated Recurrent Unit (GRU) RNNs. LSTM was introduced in 1997 (Lipton, Berkowitz and Elkan) to overcome the problem of vanishing gradients. The LSTM model introduces an intermediate type of storage via the memory cell. Each ordinary node in the hidden layers is replaced by a “memory cell”, where each memory cell contains a node with a self-connected recurrent edge of fixed weight one, ensuring that the gradient can pass across many time steps without vanishing or exploding (Lipton, Berkowitz and Elkan). LSTM creates an internal memory state which adds a forget gate to control the time dependency and effects of the previous inputs.

GRU proposed in 2014 by Cho et al (Cho et al.) is similar to LSTM, but only consists of the update and reset gate instead of the forget, update and output gate in LSTM. Both LSTM and GRU are able to decide whether to keep the existing memory from the past data by their gates or disregard it, so if LSTM and GRU detect an important feature from an input sequence at an early stage, they can easily carry this information over a long distance and capture potential long-distance dependencies. GRU is a simpler version of LSTM, which means that some of the feature of LSTM are reduced in GRU, e.g., the exposure of the memory content control. It is well established in the field that the LSTM unit works well on sequence-based tasks with long-term dependencies (Chung et al.), and LSTM is expected to have a slightly better performance than GRU (Hoang et al.).
Chapter Two

Introduction

Outdoor localization and navigation remain active research fields where the key challenge lies in the ability to minimize localization error. GPS is still the most common technology followed for outdoor navigation, but when it comes to indoor localization and navigation, GPS-based techniques are not efficient due to shadowing, multipath fading, inefficient signal penetration, and delay distortion (Yassin et al.). Over the past two decades, extensive research has been conducted to invest in technologies that tackle the indoor navigation problem with many proposed solutions as discussed in the literature review. Focusing on the received signal strength (RSS) fingerprinting, we built up our research by relying on cellular RSS from cellular base transceiver stations (BTSs) and WiFi RSS fingerprints from wireless access points (APs). The main issues with the RSS approach (further discussed in Section I) are building geometry for instance where wide and open space, thickness of the walls, and the number of doors and windows might affect the signal propagation. Also, the signal strength may be affected by signal reflection, diffraction, human body absorption, obstruction and interference of neighboring devices, in addition to weather conditions. To handle and overcome these challenges we propose a transduction neural network approach with heterogeneous RSS fingerprints, namely WiFi and cellular, where the focus is no longer on the uniqueness of the RSS but rather on the correlation to previous RSS vectors and the corresponding mapping to the current location. In other terms, the proposed approach aims to discover existing and useful dependencies between current and previous RSS vectors for more accurate localization. Building up a radio map of RSS vectors mapped to each and every \( (x, y) \) location, the recurrent neural network (RNN) is trained by the data from consecutive locations in a route to exploit the time correlation between these locations. The objective of this training is to minimize the loss function defined as the Euclidean distance between the actual and predicted \( (x, y) \) coordinates. RNN can be used for time series
A recent study similar to our work has been proposed (Hoang et al.) where RNN techniques are employed for indoor navigation with WiFi-only RSS. The latter uses for data collection an autonomous driving robot equipped with multiple sensors including; odometer, IMU, LIDAR, sonar sensors, and depth RGB-D cameras. Moreover, (Hoang et al.) uses 365 RPs, 365,000 random trajectories, and 6 wireless APs compared to 121 RPs, 50 random trajectories, and 2 APs with 3 BTSs in this study. Furthermore, in this study we propose heterogeneous fingerprinting (Wi-Fi + cellular) for indoor navigation with transduction neural networks. The proposed model can track the movement and navigation of the user and predict 4 future steps that might be followed by the user.

While referring to the paper “A Neural Network Approach for Indoor Fingerprinting-Based Localization” (Jaafar and Saab), the objective is to extend the work relying on a RNN approach for indoor navigation by taking “time” as a major factor into consideration. In addition, we consider the challenge of navigation in an open space which is characterized by more shadowing and less signal scattering as further discussed in (Section I). Using the heterogeneous RSS vectors, one goal is to check the localization error achieved using a basic NN approach and compare it to the one obtained with RNN where it is hypothesized that the latter will yield better results and higher localization accuracy. In addition, within the recurrent neural networks context, another goal is to compare the performance of LSTM RNN to that of GRU RNN where it is expected that LSTM may outperform GRU (Hoang et al.). Finally, we repeat and compare all NN models in real-time environments of cellular-only RSS, WiFi-only RSS, and heterogeneous RSS (WiFi + Cellular) to come up with the minimal mean localization error.

The rest of the thesis is organized as follows: In section III, we motivate the problem using the pathloss model and highlighting the NN fundamentals. In section IV, we describe and detail the methodology we followed along with the proposed solution for the considered neural network models, then we represent and analyze the experimental results. Finally, Section V presents concluding remarks and proposes future possible extensions of this work.
Chapter Three

Background

3.1 Uniqueness of The RSS Signature

Many indoor localization techniques have been proposed in order to achieve better localization accuracy and minimum localization error. In this work, we focus on the received signal strength (RSS) fingerprints using both WLAN and cellular wireless networks. The aim is to map the \((x, y)\) cartesian coordinates of a User Device (UD) to the corresponding RSS fingerprint, so every coordinate on the \((x, y)\) grid has its corresponding RSS vector. In navigation, time is an important factor where we have to keep track of the consequent steps. For this purpose, we consider tracking multiple routes while taking consequent steps and simultaneously saving the RSS vector at every step. Our aim is to build a navigation model, such that in the testing phase, the model takes a single or a set of consequent RSS vector(s) of a random route as an input and consequently predicts the current location and/or the current location with 4 future steps to be navigated through.

The RSS vector saved at a single \((x, y)\) location can be represented as:

\[
S_{l,k} = (s_{i,j,1} \ s_{i,j,2} \ s_{i,j,3} \ ... \ s_{i,j,N})
\]  

(1)

where \(l \in \{1, 2, ... L\}\) is the number of routes

\(k \in \{1, 2, ... K\}\) is the number of steps per route \(l\)

\(i\) is the reference of the \(x - axis\) abscissa of the grid

\(j\) is the reference of the \(y - axis\) ordinate of the grid

\(s_{i,j}\) is the received signal strength at coordinates \((i, j)\)

\(n \in \{1, 2, ... N\}\) where \(N\) is the number of anchor nodes \(N \geq 2\)

Accordingly, \(S_{l,k}\) is the RSS vector at the \(k^{th}\) step of the \(l^{th}\) route mapped to coordinate \((i, j)\) and \(s_{i,j,n}\) is the RSS read from one AN at coordinates \((i, j)\).

The RSS measurement is modeled as a nonlinear function of the distance between the UD and the AN. We consider the log-normal shadowing for the signal strength where the received signal decreases logarithmically with distance with log-normal shadowing. Hence, at a specific distance \(d_{AN}\)
between the UD and the AN, the measured RSS, $P_r(d_{AN})$, has a Gaussian distribution about the path loss distance-dependent mean.

The measured signal level in dBm at reference point $k$ is represented as:

$$P_{r,k} = p^k_{AN} - PL(d_k, \phi) - \sum_{1}^{M} L_m$$

(2)

$$PL(d_k, \phi) = PL(d_0) + 10q \log\left(\frac{d_{AN}^k}{d_0}\right) + X_\sigma^k + L_D(\phi)$$

(3)

where at the $k^{th}$ RP, $p^k_{AN}$ is the transmitted power of the AN, $PL(d_k, \phi)$ is the large-scale path loss with distance $d_{AN}^k$ between the UD and the AN, and $L_m$ is the loss imposed by $M$ obstacles (walls, doors, glass, furniture, etc.) between the UD and the AN. $d_0$ is the close-in reference distance, $q$ is the path loss exponent, $X_\sigma^k$ represents the random shadowing errors, which is a zero-mean Gaussian random variable in dB with standard deviation $\sigma$, also in dB, and $L_D(\phi)$ is the antenna angle-dependent loss.

In order to have a unique RSS signature with the RSS varying from one location to another, we need to have at least one of the powers coming from the ANs to be different. The distance between the cellular BTSs and the possible indoor locations is more or less the same, and the angle $\phi$ that the UD makes with the main radiation beam of the BTS is going to be very small and can be neglected, hence, the path loss is expected to be the same.

However, the fluctuations caused by the effects of multipath and obstacles produce the variations in the RSS i.e. the sum of losses $\sum_1^M L_m$ should overcome the random shadowing $X_\sigma^k$. If there is no difference in the RSS between two consecutive locations, then the system won’t be able to differentiate between these locations. Therefore, we hope whenever we use more ANs to get variations at least from one AN to have a unique RSS signature.
3.2 Neural Networks Fundamentals

Considering a multi-layered perceptron (MLP) feed-forward neural network with one input layer holding the RSS vector from $N$ anchor nodes, $H$ neurons in the hidden layer(s), and an output layer holding the $(x, y)$ coordinates, the estimated location at the $k^{th}$ reference point can be represented as:

$$Y_k = f_o \left\{ \sum_{h=1}^{H} w_h f_h \left\{ \sum_{n=1}^{N} w_{n,h} P_{k,n} \right\} \right\}$$

(4)

where $f_o$ and $f_h$ are the activation functions of the output layer and hidden layers respectively, $w_h$ is the weight at each node between the hidden layer and the output layer where $1 < h < H$ defines the index of the neurons in the hidden layer composed of $H$ neurons. $W_{n,h}$ represents the weight between the hidden layer and the input layer, and $P_{k,n}$ represents the RSS of the input neuron at the $k^{th}$ reference point (RP).

In the training phase of neural networks, loss functions are used to calculate the gradients that are used to update the weights of the NN. In basic neural networks a backpropagation algorithm is used to calculate the derivatives of the error with respect to the network weights, then update the weights to minimize the error.

Considering recurrent neural networks, where a sequence of timesteps is inputted to the network, backpropagation through time (BPTT) algorithm is used to unroll the network, calculate the errors across each timestep, roll up the network, and update the weights (Lipton, Berkowitz and Elkan). With long sequences, a vanishing gradient problem arises where the gradient shrinks as it back propagates to the previous layers. As the gradient value becomes extremely small it doesn’t contribute too much to the learning process. In other words, RNN layers that get a small gradient update doesn’t learn properly, consequently RNNs can forget what was seen in longer sequences and thus have short-term memory. LSTMs and GRUs were created as a solution to the short-term memory where they have internal mechanisms called gates (Figure 1) (Phi) that can regulate the flow of information (As further discussed in Section I). LSTMs and GRUs are special kind of RNN capable of learning long-term dependences where the gates can learn which data in a sequence is important to keep and which data to throw away by saving relevant information to make better predictions.
In LSTM, the gates learn what information is relevant to keep or forget during the training phase by using the sigmoid activation function that squishes and outputs values between 0 and 1. So, if there is any value that is intended to disappear then it will vanish when it get multiplied by 0, similarly any value that is intended to be stored or stay the same to be held to the next time step then it will be multiplied by 1, and in-between values will be stored partially.

As illustrated in (Figure 1) (Phi) and (Figure 2) ("Understanding Lstm Networks"), the forget gate decides what information should be thrown or kept away. Information from the previous hidden state $h_{t-1}$ and information from the current input $x_t$ is passed to the sigmoid function where values come out between 0 and 1.
To update the cell state (Figure 3) ("Understanding Lstm Networks"), first we have the input gate (Figure 1) where we pass the previous hidden state and the current input to a sigmoid function to get $i_t$, and the same values are to be passed to a $tanh$ function to get $\tilde{C}_t$, then we multiply the $tanh$ output $\tilde{C}_t$ with the sigmoid output $i_t$.

To calculate the cell state $C_t$ (Figure 4) ("Understanding Lstm Networks"), first the previous cell state $C_{t-1}$ is multiplied by the output of the sigmoid function of the forget gate $f_t$. The output is then passed to a pointwise addition with the output of the input gate, and this updates the cell state to the new value $C_t$.

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t$$ (8)

The output of the LSTM (Figure 5) ("Understanding Lstm Networks") is calculated as:

$$o_t = \sigma(W_o, [h_{t-1}, x_t] + b_o)$$ (9)
\[ h_t = o_t \cdot \tanh(C_t) \]  

(10)

The output gate (Figure 5) ("Understanding Lstm Networks") decides what the next hidden state should be. The hidden state holds information of the previous input and it is used for prediction as well. A previous hidden state \( h_{t-1} \) along with the current input \( x_t \) is passed to the sigmoid function, and the state \( C_t \) is passed to the \( \tanh \) function and its output is multiplied with the output of the sigmoid \( O_t \) to decide what information the hidden state \( h_t \) and the cell state \( C_t \) should carry to the next time step.

\[ z_t = \sigma(W_z \cdot [h_{t-1}, x_t]) \]  

(11)

\[ r_t = \sigma(W_r \cdot [h_{t-1}, x_t]) \]  

(12)

\[ \tilde{h}_t = \tanh(W \cdot [r_t \cdot h_{t-1}, x_t]) \]  

(13)

\[ h_t = (1 - z_t) \cdot h_{t-1} + z_t \cdot \tilde{h}_t \]  

(14)

The GRU - a newer generation of recurrent neural networks, has a similar architecture to LSTM, but internally instead of a cell state it uses a hidden state to transfer the information (Figure 6) (Muccino). The update gate in GRU (Figure 1) decides what information to forget and what information to add - similar to the forget and input gates of LSTM, and the reset gate is used to decide how much past information to forget. GRUs has less tensor operations and therefore they are expected to be a bit faster in training compared to LSTM.
Chapter Four

Proposed Scheme and Methodology

In this section we highlight the schemes and methods used while developing the project starting with the data collection setup to building up the neural network models. The first challenge was to come up with an efficient way for recording the RSS values from different anchor nodes (ANs) and associating these values to reference points (RPs) distributed over a cartesian grid of \((x, y)\) locations. Stage two was building up neural networks dissecting and analyzing the collected data to come up with an efficient model predicting the \((x, y)\) locations from a series of past RSS values.

Our methodology is divided into the following subsections; writing a phone application for data collection (4.1), inspecting the site for data collection (4.2), and building/designing the desired neural networks (4.3).
4.1 Phone Application for Data Collection

One of the major challenges in this research was data collection! We had to find an efficient manner to collect data, refine it, and prepare it to be inputted to the neural network at a later stage. The first goal was detecting and saving the RSS values. So, first we have to look at the GSM signals from different mobile operators (Alfa and Touch in our case) and save them. Also, we have to search for RSS values from different WLAN access points (APs) and save them in conjunction with the cellular RSS.

To start with the RSS data collection, we need an application capable of reading and recording the signal strength received from the BTSs and their neighbor cells, and another application to record the Wi-Fi signal detected from the APs located in the field. To do so, technically we need two phones with two SIM cards with every SIM connected to a different operator, and a third device to record the Wi-Fi RSS. Another challenge is to capture and synchronize the sequence of both RSS values in a timely manner.

This task has to be done at every single step – where every step is mapped to a certain \((x, y)\) coordinate on the grid. In a way to avoid moving in the site collecting data with 3 devices, we developed an android phone application that searches for the nearby Wi-Fi APs and cellular BTSs with their neighbor cells, and finally save this data as a .CSV file on the phone storage. The phone that is used for data collection is a “Samsung Note 8” that supports dual SIM with every SIM connected to a different operator. This helped us in having one device capable of recording the RSS from the both Wi-Fi APs and the two local mobile operators.
The android application we created has control over the Wi-Fi state of the phone to turn it on/off when needed, and it has a read/write privilege to write on the phone storage as shown in the below manifest (Figure 7):

![AndroidManifest.xml](image)

*Figure 7: Android application manifest controlling the permissions granted to the application and the runtime of the main activity*

Below is the application interface (Figure 8) where we have 6 buttons for the interaction between the phone and the surrounding environment.

![Application Interface](image)

*Figure 8: Phone Application Interface developed in this work*
First, the “Route” button tracks the routes we will move through, and when clicked it increments +1 to start recording a new set of steps (Figure 9).

```
://Route Button Listener
public void createRoute(){
    route = ++route;
    if(route>10){
        dir = new File( pathname = "/storage/emulated/0/3G_RSS/+File.separator=0+route/+" );
        dir.mkdirs();
    } else {
        dir = new File( pathname = "/storage/emulated/0/3G_RSS/+File.separator=+route/+" );
        dir.mkdirs();
    }
}
```

Figure 9: Route tracking button to create directories for saving RSS values

After initiating the route, the “RSS” button is used to capture the RSS values of all detected wireless signals (Figure 10).

```
://Show 3G and Wi-Fi info Button
bCell.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        try {
            String[] par = null;
            BaseStation main_BS = bWIFInfoList.get(0);
            //textView.setText(telephonyManager.getNetworkOperatorName() + "|cellIndex: " + cellNumber + "|n = " + main_BS.tostring() + "|n = date|n [e];
            for (CellInfo cellInfo : cellInfos){
                if (cellInfo.getCellId().equals(main_BS.getCellId())) {
                    Log.i(TAG, "cellNumber = " + cellNumber + "cellIds= " + main_BS.getCellId());
                    String[] par = null;
                    for (CellInfo cellInfo : cellInfos){
                        if (cellInfo.getCellId().equals(main_BS.getCellId())) {
                            Log.i(TAG, "cellNumber = " + cellNumber + "cellIds= " + main_BS.getCellId());
                            String[] par = null;
                        }
                    }
                }
            }
        } catch (Exception e) {
            Log.e(TAG, "Error while getting network operator name: ", e);
        }
    }
});
```

Figure 10: Code behind the RSS button

The “RSS” button runs the main functionalities, where starting with step 0 by pressing this button the step counter increments +1 and nested subfunctions are triggered to detect the cellular base stations with their neighbor cells (Figures 11-12) and the Wi-Fi APs (Figure 13).

```
for (CellInfo cellInfo : cellInfos){
    if (cellInfo.getCellId().equals(main_BS.getCellId())) {
        Log.i(TAG, "cellNumber = " + cellNumber + "cellIds= " + main_BS.getCellId());
        String[] par = null;
    }
}
```

Figure 11: Snapshot of code for detecting cellular stations with their cell IDs
One of the limitations we faced was having the base stations with many neighbor cells that are shown with constant RSS of “-51 dBm”. This RSS doesn’t enclose any valuable information since it doesn’t reflect the real RSS of the neighbor cells, so we excluded these cells and focused on the major ones with specific cell IDs in the site which the phone was connected to; namely “Alfa” operator for GSM calls, “Touch” operator for GSM calls as well, and “Touch LTE” cell for the 4G data bundle (Figure 11).

As shown in (Figure 12) the application detects the type of the main cell and neighbor cells as well whether it is of type “WCDMA”, “LTE”, or “GSM”. We saved the strings as operator name, followed by the cell ID, and the RSS in dBm.

The “RSS” button triggers another subroutine responsible for searching for Wi-Fi access points (wifiManager.getScanResults()) and saving the AP name followed by the RSS in dBm as shown in (Figure 13).
Before pressing the “RSS” button we had to input the current \((x,y)\) location at which we are detecting the received signals in a way to map this recording with the \((x, y)\) coordinates. The values saved after clicking the “RSS” button are displayed screen (Figure 14):

![Phone App sample output](image)

**Figure 14: Phone App sample output**

The sequence of the steps is a major factor to be taken into consideration, accordingly, a steps counter that keeps track of the steps is added to the input vector along with the time when the RSS was captured. A sample of the output saved on the phone as a CSV file at each step is displayed in (Figure 15):
The remaining buttons were added due to many hurdles we faced while collecting the data, and we manipulated the phone app accordingly. This to be further highlighted in the next section.
4.2 Site Inspection and Data Collection

After setting up the phone application, the goal was to build up a radio map of RSS vectors mapped to each and every \((x, y)\) before feeding this input to the neural network models. The target area for our study has to be a location where we have open space and wide hallways. Pursuing such environment, we targeted a building located amid an urban area; the “Irwin Hall” building in LAU Beirut campus -7th floor the “LAU Fouad Makhzoumi Innovation Center”. The floor is around 547m² and the surveying site is around 374m² -the grid in (Figure 16). We were able to work in an open area of 130 m² due to the fact of having staff offices and classes closed due to the lockdown. Our RPs were distributed on a \((x, y)\) grid basis as shown in (Figure 14):

![Floorplan of the Surveying Site in Irwin Hall Level 7](image)

We started with a grid of 1m x 1m as shown in (Figure 16), but to get better navigation accuracy we followed a grid of 0.6m x 0.6m and we took advantage of the tiles each of area 60cmx60cm. The floor is surrounded with a layer of thick stone walls and glass windows with double glazing, so the signal attenuation was expected to be high and any factor like opening a window or a door can affect the signal strength.
The origin of the grid is located to the left of the elevator while entering the floor facing a hallway of an area of around 44m² as shown in (Figure 17):

![Figure 17: First hallway (44m²) for data collection](image1)

Then the hallway extends to another hallway of an area around 77m² as shown in (Figure 18):

![Figure 18: Second hallway (77m²) for data collection](image2)

As shown in the above images we had a wide space and very long hallways (4-meters wide) for data collection, and as mentioned above this is one of the challenges for RSS fingerprinting tackled in this study.
We have 2 Cisco Air-CAP2702I access points (“Cisco Aironet Series 1700/2700/3700 Access Points Deployment Guide”) covering the whole floor, and two APs between the 7th and 6th floor; one next to the elevator and another one next to the second hallway. While recording the RSS values, it is essential to have readings from the same source of info (BTSs and APs) at any location in the floor. If the source BTS or AP disappears then it won’t be valuable anymore. As a matter of fact, and due to the long hallways, we have readings from only two APs covering the whole area located in the hallways (Figure 16).

Starting with the data collection process, our goal was to record many random routes to provide the neural networks with enough data for training. While moving at every reference point, we enter the \((x, y)\) coordinate on the phone manually and press on the “RSS” button to save the RSS values. Many hurdles faced us in the phase, and we had to manipulate and add code to the phone app to accommodate the challenges we faced. The first obstacle was enforced by “Android 9.0” OS where we were able to run only 4 scans for the Wi-Fi every two minutes while using our application (“Wi-Fi Scanning Overview : Android Developers”). To overcome this issue, we had a workaround of opening the Wi-Fi settings on the phone in parallel to working on our application (Figure 14), so this way the app uses the default Wi-Fi scanning of the phone in parallel to recording the Wi-Fi RSS in our application. Also, while moving from one point to another sometimes we were detecting exactly the same RSS values within two consecutive steps, and this imposed a limitation to our data collection where we had to wait few seconds for the phone to refresh before saving the RSS vectors. For this we added another “Undo” button to undo the step by decrementing the step counter, and undo the last recording by deleting it from the phone storage. Furthermore, while recording the RSS vectors, one of the APs suddenly disappears for few seconds then appears again, and the phone switches temporarily from one cell to another for few seconds then switches back to the main cell that was connected to it. The latter issue resulted in saving RSS values less than 5, so we added a constraint of having a temporary counter to save the values only when we have readings from 5 ANs. We made sure to have a timestamp and a step counter added to every recording after each save to guarantee the sequential manner of the steps.
Over 121 reference points we were able to record 50 random routes holding 1029 steps in which every route had between 17 to 38 steps. The raw data we got were saved repeatedly as CSV files with the same format (Alfa, Touch, LTE, AP1, and AP2) for all steps. In order to have all data unified with one table header, we merged the CSV files and refined the raw data using macro code in MS Excel. The code takes the merged CSV files, trims the unneeded cells, reorders all input nodes to make sure that all their corresponding values are saved sequentially next to each other.

The figure below (Figure 19) displays all RSS values in conjunction with their corresponding 1029 \((x, y)\) coordinates. Every column represents a footstep that holds info of the 5 RSS values with its matching \((x, y)\) coordinate.

It is noted on site at LAU Beirut campus that the average signal strength of “Touch” mobile operator is better than “Alfa” – where “Touch” has a BTS installed on campus. This was reflected in the mean calculation of the RSS values for the 3 cellular base stations with Alfa: -89.143 dBm, Touch: -86.648 dBm, and Touch LTE: -83.837 dBm.
4.3 Neural Network Approach for Indoor Navigation

After collecting the data and saving it in an offline database, the goal was to feed this data to a neural network and check the outcome of predicting \((x, y)\) locations after feeding the RSS values as an input. (Figure 20) represents a basic neural network diagram in which the data at the input layer propagates through the network via a sequence of affine transformations followed by nonlinear activation functions in a layer-wise fashion to reach the output layer. The process of training a NN begins by randomly initializing all the parameters (weights and biases), and iteratively updating them with every batch of training data. Using ground-truth labels, the error can be calculated using a given loss function, which is then used to calculate the gradient of error in order to update the parameters. The calculation of the gradient proceeds backwards through the network layers updating the weights in a process to minimize the error. In the training process overfitting might occur when the network starts memorizing the training set where the training loss continues to go down and the validation loss either saturates or increases. To avoid overfitting, we added an “early stopping” technique that compares the validation loss to the training loss after every epoch and stops the training process if the validation loss starts to increase after a certain number of runs.
As detailed in (section 4.2) all RSS values are mapped to each and every \((x, y)\) location on the grid, and as illustrated in the above flowchart (Figure 21) the next thing we did was splitting the data into 80\% for training (824 samples out of 1029) and 20\% for validation (205 samples). We pass the training data for three neural network models: one basic neural network (Section 4.3.1) and two transduction neural networks; namely Long-Short Term Memory (LSTM) recurrent neural network (Section 4.3.2), and Gated Recurrent Unit (GRU) recurrent neural network (Section 4.3.3). Transduction neural networks take as an input a sequence of data and output another sequence. In this work, the RNN models take a sequence of 40 RSS vectors as an input, and output a sequence of 5 consequent locations including the current location of the user and the next 4 future steps (locations) to be followed. Finally, the performance of the RNN models predicting 5
consequent locations is compared to outputting the current location only without the 4 future footsteps (Section 4.3.4).

After training the three network models (Figure 21), we validate the three models with the validation data, evaluate the performance of every model, and select the one with the smallest mean localization error.

The next subsections will detail and explain how every neural network model was built, and analyze the experimental results, with conclusions drawn at the end of the report.
4.3.1 Basic Neural Network

Starting with a basic neural network, as illustrated in Figure 22, the model takes one RSS vector with 5 RSS values (Alfa, Touch, LTE, AP1, AP2) as input at its input nodes, and the \((x, y)\) location tied to this unique RSS vector at the output nodes. The number of neurons and hidden layers was chosen after several trial and errors. The best number that was used among all neural network models was as follows: 2 hidden layers, with the first hidden layer consisting of 32 neurons and 16 neurons in the second hidden layer.

While setting up the basic neural network and to avoid overfitting we used “early stopping” technique where the model at the end of every epoch checks the validation loss and compares it with the next validation loss. If this loss starts to increase then the model starts to overfit, but it’s not always the case when the validation loss increases after one epoch then there is overfitting, since sometimes it increases for few epochs then keeps on decreasing in the next epochs. In the “early stopping” technique we added a “patience” parameter of 10 steps to check if the validation loss will keep on increasing, and if so then the model stops training and restores the best weights when it had the least training/validation loss.
To evaluate the basic NN model and compare it to other models we added several metrics that are commonly used to measure the accuracy for continuous time series data and error prediction namely the "root mean square error - RMSE", standard deviation, and the “mean absolute error - MAE” for the X coordinate and Y coordinate prediction in addition to the cumulative distribution function “CDF” that shows the percentages of all predicted values within the 2m, 1.5m, and 1m mean localization error.

From a sample taken from the basic NN testing, we input a sample RSS of a known location of coordinates (1,15) as shown in (Figure 24) and the model outputs the predicted coordinates (2.65, 9.5). To calculate the cost function, we get the Euclidian distance between the expected coordinate and the predicted one as follows:

\[ E_k = \sqrt{(x_{\text{nominal},k} - \hat{x}_k)^2 + (y_{\text{nominal},k} - \hat{y}_k)^2} \]

where \((x_{\text{nominal}}, y_{\text{nominal}})\) is the actual location, and \((\hat{x}, \hat{y})\) is the predicted coordinate at the \(k\)th RP. In this example the localization error is 3.44m.

![Figure 23: Sample of the basic neural network training/validation loss](image-url)

![Figure 24: Sample prediction - Error calculation](image-url)
Figure 25: Basic NN regression model response Expected vs Prediction values

(Figure 25) shows the performance of one of the basic NN samples (Sample #10) with “cellular and WiFi” RSS values in terms of expected coordinates vs the predicted ones. The predicted values greatly differ from the expected values with a mean of 4.5m (Code and results attached with the report). For validation, we have 205 samples (20% of 1029) for the x-abscissa and 205 samples for the y-ordinate, accordingly the x-axis in (Figure 25) represents the 410 samples of x and y flattened as \((x_0, y_0, x_1, y_1, \ldots, x_{204}, y_{204})\), and the y-axis shows the value of the coordinates ranging from 0 to 18 as per the grid (section 4.2). The blue plot represents the nominal \((x, y)\) coordinates and the orange one represents the output of the validation data.

In the basic neural network, we trained the first model with RSS data of cellular signals and Wi-Fi signals combined together as a fingerprint, then we added two models to see the effect of having WiFi-only RSS, and cellular-only RSS. The results of the 3 models are summarized in (Table 1):

<table>
<thead>
<tr>
<th></th>
<th>Mean (m)</th>
<th>Std. Dev</th>
<th>RMSE X</th>
<th>RMSE Y</th>
<th>MAE X</th>
<th>MAE Y</th>
<th>CDF &lt;2m</th>
<th>CDF &lt;1.5m</th>
<th>CDF &lt;1m</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Basic NN Wi-Fi/4G</strong></td>
<td>3.907</td>
<td>1.82</td>
<td>2.962</td>
<td>3.091</td>
<td>2.456</td>
<td>2.606</td>
<td>17.85%</td>
<td>8.24%</td>
<td>1.9%</td>
</tr>
<tr>
<td><strong>Basic NN 4G</strong></td>
<td>3.329</td>
<td>1.221</td>
<td>1.963</td>
<td>2.953</td>
<td>1.876</td>
<td>2.52</td>
<td>16.05%</td>
<td>5.32%</td>
<td>0.44%</td>
</tr>
<tr>
<td><strong>Basic NN Wi-Fi</strong></td>
<td>4.002</td>
<td>1.997</td>
<td>3.178</td>
<td>3.115</td>
<td>2.537</td>
<td>2.612</td>
<td>18.73%</td>
<td>9.32%</td>
<td>2.63%</td>
</tr>
</tbody>
</table>

Table 1: Results of Basic Neural Network (Cellular/Wi-Fi)
We ran the 3 models 10 times and recorded all metrics then averaged them to get the values in the above table. The green-highlighted values correspond to the best results when comparing the three models together. The model with cellular-only RSS fingerprints achieved the best results among the other 2 models in terms of mean localization error, standard deviation, x-RMSE, y-RMSE, x-MAE, and y-MAE. These measures show how the cellular-RSS model outperforms the other two models, but this doesn’t reflect how good the overall performance of the 3 models is – where the percentage of values less than 2-meters accuracy are almost the same (all less than 19%). This low percentage reflects the poor performance of the basic neural network in such environments with open space and wide hallways.

(Figure 26) shows the cumulative distribution function (CDF) of the localization error for the proposed models. The results show that the cellular-only RSS model has 69% of the values less than 4 meters, 55% for the WiFi-only RSS model, and 57% for the heterogeneous model (WiFi/cellular). It is to be noted that if the any RSS value change abruptly then the prediction model will result in coordinates that might be far from the desired ones. So, relying on one RSS vector is not enough. This opened the door for us to dig more and check if we can have models that can memorize how the RSS values are fluctuating and build a correlation among these changes to give better predictions. Here comes the idea of using transduction neural networks.
4.3.2 Long-Short Term Memory (LSTM) Neural Network

Transduction neural networks perform in a different way than regular neural networks, where instead of having one RSS vector as an input, the network takes a sequence of RSS vectors as an input and predicts the current position and a sequence of four consequent coordinates. We need a neural network that has some kind of memory where it can look back at the past history of the RSS values, build a correlation among these values, learn from this correlation, be resilient to changes in RSS values, and give accurate predictions of the current location based on the previous values of the RSS vectors. For this purpose, we used two state-of-the-art recurrent neural networks that have internal memory; namely LSTM and GRU. As detailed in the background (Section III) both LSTM and GRU have an internal memory state at the neuron level, where LSTM has “forget”, “update”, and “output” gates to control the dependencies of previous inputs, and GRU has “update” and “reset” gates.

Our proposed LSTM model takes $x \in \mathbb{R}^{40\times5}$ sequence as one input block (Figure 27) and outputs a sequence of 5 consequent coordinates. The number of input blocks “$b$” depends on the length of the training set:

$$b = L - h - f$$

$L$ is the length of the training, $h$ is the past history of RSS vectors (40 in our case), and $f$ is the future steps to be predicted (5 in our case).

![Figure 27: LSTM/GRU input blocks](image-url)
The “Mean Square Error” was used as the loss function, and “RMSprop” as an optimization function to minimize the error while backpropagating the values in the training phase. Also, with the “early stopping” technique we used a “patience” of 3 steps to detect overfitting. It is important to mention that in the LSTM and GRU models we trained the networks to predict the x-abscissa alone and another identical model for the y-ordinate prediction. (Figure 28) shows a sample “Training loss vs Validation loss” for the x-coordinate, and (Figure 29) for the y-coordinate.

(Figure 28) shows that the validation loss starts to increase at the 7th iteration – where this might be a sign of overfitting, but after 2 steps the loss starts to decrease again. Since we have a patience of 3, the model didn’t stop the training at the 7th iteration hoping that the error will decrease again, and that what happened. After the 10th iteration the validation loss starts to increase again, so after 3 iterations (since patience is 3) “early stopping” stops the training and restores the best weights where the validation/training loss was minimal.
Figure 29: LSTM Sample Training vs Validation loss (y-ordinate)

(Figure 29) shows how the validation loss is fluctuating to reach a steady state at the last epoch. In this sample “early stopping” didn’t stop the training since the model didn’t cross the patience of 3 steps every time the validation loss starts to increase. After training our LSTM model, we can input any (40 RSS) vectors and check the predicted \((x, y)\) coordinates.

Figure 30: LSTM Sample X prediction True vs Predicted future

(Figure 30) shows a sample prediction of our model where for displaying purposes we had prior info about the 5 expected x-coordinates. The y-axis shows the x-abscissa while moving along the x-axis in real-time environment, and the x-axis of the graph shows the past 40 steps and the 5 predicted steps. The blue dots represent the true future of how we should be moving, and the red dots represent the 5 forecasted steps. The difference is between 2 and 3.5 steps equivalent to 1.2m to 2.1m. It is to be noted that the
prediction accuracy of the first location is expected to be better than the accuracy of the following steps where the accuracy starts to decrease gradually.

Figure 31: LSTM Sample Y prediction True vs Predicted future

Taking the same sample into consideration (Figure 31) shows the true coordinates versus the forecasted steps over the y-ordinate.

Figure 32: LSTM sample response Expected vs Prediction 5-steps X values
Compared to the basic neural network response (Figure 25), LSTM has a much better accuracy and it’s clear in (Figures 32-33) how the model is responding to the changes in RSS values and outputting results with high accuracy compared to the basic NN. The y-axis shows the steps coordinate, and the x-axis shows the 163 validation steps after flattening the array (163 x 5) to see the response of the model for all 5 locations (current steps + 4 forecasted future steps). As mentioned before, the accuracy of predicting the next 4 steps is less than the prediction of the first step, and that’s why a gap is still showing between the expected and predicted values in (Figures 32-33).
(Figures 34) displays the current x coordinate prediction/validation, and (Figure 35) displays the current y coordinate prediction/validation with the y-axis showing the coordinates and the x-axis showing the 163 validation steps. In the above figures (34-35) the difference between the expected and predicted values improved compared to the basic NN, and the average mean error of this run (10th sample) is 1.5m.

As mentioned before, the accuracy of predicting the current location is expected to be better than that of the consequent step. This is projected in (Figure 36) where the current location (LSTM_0) has better CDF compared the 4 steps. For example, the 4th step (LSTM_4) has 55% of the values within a mean error of 2 meters, however (LSTM_0) has 77% of the values within 2m.

The mean error of predicting the current location for this sample, along with the following 4 steps are shown in (Table 2):
Table 2: 5 step forecasting mean error of LSTM sample

<table>
<thead>
<tr>
<th>Current location</th>
<th>1st step</th>
<th>2nd step</th>
<th>3rd step</th>
<th>4th step</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1.547</td>
<td>1.679</td>
<td>1.851</td>
<td>2.036</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2.318</td>
</tr>
</tbody>
</table>

Following the same metrics used with the basic NN, we evaluated the LSTM model after running it for 10 times and averaging the values to get better figures. We also executed 10 runs for the WiFi-only RSS model and cellular-only RSS model to evaluate their performance and compare it to the basic NN models. The results are summarized in the below table:

Table 3: Results of LSTM Neural Network (Cellular/Wi-Fi)

<table>
<thead>
<tr>
<th>LSTM Wi-Fi/4G</th>
<th>Mean (m)</th>
<th>Std. Dev</th>
<th>RMSE X</th>
<th>RMSE Y</th>
<th>MAE X</th>
<th>MAE Y</th>
<th>CDF &lt;2m</th>
<th>CDF &lt;1.5m</th>
<th>CDF &lt;1m</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.498</td>
<td>1.112</td>
<td>1.111</td>
<td>1.453</td>
<td>0.878</td>
<td>0.966</td>
<td>79.87%</td>
<td>64.44%</td>
<td>37.12%</td>
<td></td>
</tr>
<tr>
<td>LSTM 4G</td>
<td>2.997</td>
<td>1.611</td>
<td>2.096</td>
<td>2.592</td>
<td>1.428</td>
<td>2.267</td>
<td>29.62%</td>
<td>16.94%</td>
<td>8.31%</td>
</tr>
<tr>
<td>LSTM Wi-Fi</td>
<td>1.338</td>
<td>1.013</td>
<td>1.090</td>
<td>1.17</td>
<td>0.884</td>
<td>0.789</td>
<td>87.87%</td>
<td>75.31%</td>
<td>42.56%</td>
</tr>
</tbody>
</table>

The green-highlighted values in (table 3) are the best results among the 3 LSTM models. By examining these values, LSTM model with Wi-Fi-only RSS vectors outperforms the other 2 models with almost all metrics with a mean error of 1.338 meters, and a very high accuracy where 88% of the values are within 2-meters.

Figure 37: CDF of Mean Error for LSTM Recurrent Neural Network

As shown in (Figure 37), the distribution of the mean error with WiFi-only RSS is close to that of the Wi-Fi and cellular RSS combined together, and both models are much better than the cellular-only RSS model.
4.3.3 Gated Recurrent Unit (GRU) Neural Network

The setup of building the GRU model is the same to that of LSTM with the only difference at the neuron level with the gates of GRU compared to LSTM as detailed in (section III). Below figures are taken from one of the samples (1st sample) while running the GRU model.

![Figure 38: GRU Sample Training vs Validation loss (x-abscissa)](image)

![Figure 39: GRU Sample Training vs Validation loss (y-ordinate)](image)

After training the heterogeneous RSS GRU model, Figures (38-39) represent the validation versus training error for the \((x, y)\) coordinates. With the above 2 samples the “early stopping” technique didn’t stop the model while training since the validation loss didn’t increase for 3 consecutive steps and thus the model finished the training with minimal losses.
Taking a past history of 40 RSS values, GRU had good results while comparing the true location of the future steps to the predicted ones (Figures 40-41). At this specific run, GRU achieved better results than LSTM (Figures 30-31) where the difference between the predicted and expected locations is better than LSTM. Also, similar results were projected in the below graphs (Figure 42-45) where we see how the GRU model resulted in a very good performance compared to the basic neural network, and a similar performance compared to LSTM.
Figure 42: GRU sample response Expected vs Prediction 5-steps X values

Figure 43: GRU sample response Expected vs Prediction 5-steps Y values
Figure 44: GRU sample response Expected vs Prediction 1-step x-values

Figure 45: GRU sample response Expected vs Prediction 1-step Y values

Figure 46: GRU sample CDF showing the 5 forecasted steps

Taking the same sample (results attached to the report) we notice that while predicting the first step we have a better accuracy compared to forecasting
the 4 next steps, and in this specific run (Sample #10) the mean of the 5-steps prediction is shown in the following (table 4):

Table 4: 5-step forecasting mean error of GRU sample

<table>
<thead>
<tr>
<th>Current location</th>
<th>1st step</th>
<th>2nd step</th>
<th>3rd step</th>
<th>4th step</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1.565</td>
<td>1.821</td>
<td>2.018</td>
<td>2.013</td>
</tr>
</tbody>
</table>

The above results can be observed in (Figure 46), where 80% of the errors are less than 2m for the current location (GRU_0) prediction, and the future steps are next to it but with less accuracy.

Again we repeated the GRU runs for 10 times to get a better figure about the performance of the model, then we repeated the runs for the GRU model with “Wi-Fi only” RSS vectors, and “cellular-only” RSS. The results are summarized in the following (table 5):

Table 5: Results of GRU Neural Network (Cellular/Wi-Fi)

<table>
<thead>
<tr>
<th>Neural Network</th>
<th>Mean (m)</th>
<th>Std. Dev</th>
<th>RMSE X</th>
<th>RMSE Y</th>
<th>MAE X</th>
<th>MAE Y</th>
<th>CDF &lt;2m</th>
<th>CDF &lt;1.5m</th>
<th>CDF &lt;1m</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRU Wi-Fi/4G</td>
<td>1.635</td>
<td>1.343</td>
<td>1.151</td>
<td>1.750</td>
<td>0.884</td>
<td>1.097</td>
<td>77.12%</td>
<td>61.62%</td>
<td>36.44%</td>
</tr>
<tr>
<td>GRU 4G</td>
<td>2.731</td>
<td>1.752</td>
<td>2.212</td>
<td>2.456</td>
<td>1.622</td>
<td>1.7</td>
<td>42.37%</td>
<td>29.19%</td>
<td>15.37%</td>
</tr>
<tr>
<td>GRU Wi-Fi</td>
<td>1.523</td>
<td>1.114</td>
<td>1.195</td>
<td>1.325</td>
<td>0.988</td>
<td>0.887</td>
<td>80%</td>
<td>62.87%</td>
<td>36.06%</td>
</tr>
</tbody>
</table>

As shown in (table 5), the green-highlighted values correspond to the better results achieved while running the 3 GRU models. The GRU model with “WiFi-only” RSS proved again to outperform the “cellular-only” RSS model with all metrics, and have a similar performance to the heterogeneous RSS model but with slight improvement in terms of mean, standard deviation, y-RMSE, y-MAE. The “Wi-Fi only” GRU model also proved to be better than the other 2 models in terms of the values less than 2-meter (80%) compared to 77% and 42% for the heterogeneous RSS, and cellular-only RSS respectively.
Figure 47: CDF of Mean Error for GRU Recurrent Neural Network

The results are better illustrated in (Figure 47) where the Wi-Fi GRU clearly outperformed the GRU with cellular RSS vectors, but very close in performance to the of (cellular and Wi-Fi) -almost the same. It is worth mentioning that having a model with the 5 anchor nodes (cellular and Wi-Fi) gave very good results, but again with a slight difference less than the Wi-Fi only GRU model.
4.3.4 Experimental Results and Analysis

In this section we represent a summary of all the above-mentioned results, and we share the testing results of two of the best models in real-time environment. Then we validate and test the same models with only one step (current location) prediction - without forecasting the 4 additional locations. The latter approach achieved the better results compared to the 5-steps prediction. Finally, we compare our results to a study (Hoang et al.) conducted recently and draw up a conclusion at the next section.

The performance metrics are the average mean localization error, standard deviation error, root mean square error for X and for Y, mean absolute error for X and Y, percentage of values within 1m, 1.5m, and 2m.

The results of predicting the first step (current location) for all 9 models are summarized in (Table 6) below:

<table>
<thead>
<tr>
<th></th>
<th>Mean (m)</th>
<th>Std. Dev</th>
<th>RMSE X</th>
<th>RMSE Y</th>
<th>MAE X</th>
<th>MAE Y</th>
<th>CDF &lt;2m</th>
<th>CDF &lt;1.5m</th>
<th>CDF &lt;1m</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic NN Wi-Fi/4G</td>
<td>3.907</td>
<td>1.82</td>
<td>2.962</td>
<td>3.091</td>
<td>2.456</td>
<td>2.606</td>
<td>17.85%</td>
<td>8.24%</td>
<td>1.9%</td>
</tr>
<tr>
<td>Basic NN 4G</td>
<td>3.329</td>
<td>1.221</td>
<td>1.963</td>
<td>2.953</td>
<td>1.876</td>
<td>2.52</td>
<td>16.05%</td>
<td>5.32%</td>
<td>0.44%</td>
</tr>
<tr>
<td>Basic NN Wi-Fi</td>
<td>4.002</td>
<td>1.997</td>
<td>3.178</td>
<td>3.115</td>
<td>2.537</td>
<td>2.612</td>
<td>18.73%</td>
<td>9.32%</td>
<td>2.63%</td>
</tr>
<tr>
<td>LSTM Wi-Fi/4G</td>
<td>1.498</td>
<td>1.112</td>
<td>1.111</td>
<td>1.453</td>
<td>0.878</td>
<td>0.966</td>
<td>79.87%</td>
<td>64.44%</td>
<td>37.12%</td>
</tr>
<tr>
<td>LSTM 4G</td>
<td>2.997</td>
<td>1.611</td>
<td>2.096</td>
<td>2.592</td>
<td>1.428</td>
<td>2.267</td>
<td>29.62%</td>
<td>16.94%</td>
<td>8.31%</td>
</tr>
<tr>
<td>LSTM Wi-Fi</td>
<td>1.338</td>
<td>1.013</td>
<td>1.090</td>
<td>1.17</td>
<td>0.884</td>
<td>0.789</td>
<td>87.87%</td>
<td>75.31%</td>
<td>42.56%</td>
</tr>
<tr>
<td>GRU Wi-Fi/4G</td>
<td>1.635</td>
<td>1.343</td>
<td>1.151</td>
<td>1.750</td>
<td>0.884</td>
<td>1.097</td>
<td>77.12%</td>
<td>61.62%</td>
<td>36.44%</td>
</tr>
<tr>
<td>GRU 4G</td>
<td>2.731</td>
<td>1.752</td>
<td>2.212</td>
<td>2.456</td>
<td>1.622</td>
<td>1.7</td>
<td>42.37%</td>
<td>29.19%</td>
<td>15.37%</td>
</tr>
<tr>
<td>GRU Wi-Fi</td>
<td>1.523</td>
<td>1.114</td>
<td>1.195</td>
<td>1.325</td>
<td>0.988</td>
<td>0.887</td>
<td>80%</td>
<td>62.87%</td>
<td>36.06%</td>
</tr>
</tbody>
</table>

The first three rows correspond to the basic NN models, and the yellow highlighted values represent the best results among the basic NN models. Similarly, the middle three rows correspond to the 3 LSTM models where the green-highlights represent the best values among all 9 models, and the last three rows correspond to the 3 GRU models with the yellow-highlighted values highlighting the best results among the GRU models.

The values extracted from (Table 6) show how the transduction RNN outperforms the basic NN in terms of all metrics - to be further discussed.
The “cellular-based” RSS model of the basic NN achieved a mean of 3.33m with a standard deviation of 1.22 meters, which is considered a bit high when compared to the RNN models. Also, while comparing the other metrics of the basic NN, it shows how RNN models surpass the basic NN in terms of standard deviation, RMSE, MAE, and the percentage of values less than 2m. The results of the basic neural network were not promising due to the fact of the open space environment and the limitation of having limited number of ANs. Switching to transduction neural networks was the breakpoint where we got better scores to be discussed and analyzed further in this section.

Inspecting the best results of all 9 models, the WiFi-based LSTM model achieves the best mean error of 1.338m which is 60% better than the cellular-based basic NN with mean of 3.329m, and 12% better than the WiFi-based GRU of 1.523m mean (Wi-Fi GRU - the best model among the 3 GRU models).

Comparing the models with cellular-based RSS vectors, the CDF (Figure 48) shows the GRU RNN model outperforming the LSTM and basic NN models. As presented in (Table 6) and displayed in (Figure 48), the percentage of mean error less than 2m of the GRU cellular-based model surpasses the LSTM cellular-based model and basic NN model as illustrated in with 42.37% for GRU, 29.62% for LSTM, and 17.85% for basic NN.
Comparing the models with WiFi-based RSS, the CDF (Figure 49) displays how the LSTM model overtakes the GRU and basic NN models. In terms of mean error less than 2m, LSTM shows 12% improvement of GRU WiFi model, and 66% over WiFi-based RSS basic NN.

By examining the CDF of the 5 anchor nodes combining cellular and Wi-Fi RSS fingerprints (Figure 50), we observe how the RNN models overtakes the basic NN model with 80% and 77% of the error values of LSTM and GRU models respectively are less than 2 meters, compared to 8% with the basic NN model. Moreover, in terms of average mean error, LSTM model with 1.498m is 8% better than GRU (1.635m) and 62% better than the basic NN (3.907m).

The improvement of the transduction NN over the basic NN is due to the sequential nature of the time series steps and the advantage of having a memory with the RNN models. So, with the correlation of the past RSS values, the prediction of the current and future locations is improved.
compared to the basic NN models – regardless of the signal strength fluctuation at specific coordinates. It is expected that we can have much better results with submeter accuracy if we have more wireless access points resulting in a better signature, and sure collecting more and more steps and RSS vectors certainly will improve the accuracy of the models. This was obvious when we did testing of the best model that we got with Wi-Fi only RSS, and “Wi-Fi and cellular” combined together where it was clear when we decreased the number of steps in the training/testing phase, the accuracy was a bit affected as explained below.

The best model was the LSTM Wi-Fi based model, so we evaluated its performance by imposing new data that is not included in the training/validation phase. We then compared it to LSTM with heterogeneous RSS (cellular and Wi-Fi) fingerprints. In the testing phase we feed the model with 40 sequential RSS vectors and observe the predicted \((x, y)\) locations. Having a total of 1029 steps, we split the dataset into 74.34% training, 18.46% validation, and 7.2% for testing. Training the model with 74% of the data instead of 80% affected a bit the accuracy with a mean error of 1.368m and 1.398m for two runs of the LSTM Wi-Fi based model. Moreover, testing the LSTM model with heterogeneous RSS, we got a mean error of 1.461m and 1.402m after two runs. The results for the first step (current location) prediction are summarized in the below (table 7) where we averaged the metrics values of the two runs held per model.

**Table 7: Testing results of 2 LSTM models taking only the first step**

<table>
<thead>
<tr>
<th>Model</th>
<th>Mean (m)</th>
<th>Std. Dev</th>
<th>RMSE X</th>
<th>RMSE Y</th>
<th>MAE X</th>
<th>MAE Y</th>
<th>CDF &lt;2m</th>
<th>CDF &lt;1.5m</th>
<th>CDF &lt;1m</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM RNN Wi-Fi/4G</td>
<td>1.432</td>
<td>0.747</td>
<td>1.113</td>
<td>1.744</td>
<td>0.981</td>
<td>0.798</td>
<td>87.93%</td>
<td>62.07%</td>
<td>20.69%</td>
</tr>
<tr>
<td>LSTM RNN Wi-Fi</td>
<td>1.383</td>
<td>0.626</td>
<td>1.119</td>
<td>1.004</td>
<td>0.968</td>
<td>0.814</td>
<td>75.86%</td>
<td>63.79%</td>
<td>25.86%</td>
</tr>
</tbody>
</table>

**Table 8: LSTM Model Testing with 5-steps Forecasting**

<table>
<thead>
<tr>
<th>LSTM RNN</th>
<th>Current</th>
<th>Step #1</th>
<th>Step #2</th>
<th>Step #3</th>
<th>Step #4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wi-Fi/4G</td>
<td>1.432m</td>
<td>1.534m</td>
<td>1.94m</td>
<td>2.386m</td>
<td>2.804m</td>
</tr>
<tr>
<td>Wi-Fi</td>
<td>1.383m</td>
<td>1.384m</td>
<td>1.677m</td>
<td>2.266m</td>
<td>2.735m</td>
</tr>
</tbody>
</table>

(Table 8) shows the average mean error for the forecasted steps, where it shows how the accuracy of predicting the first step is better than the 4 forecasted steps. In the testing phase as shown in (Table 7), the LSTM Wi-Fi model achieved better results compared to the LSTM of heterogeneous RSS
vectors – as it was observed in the training phase. The Wi-Fi LSTM model in the testing phase resulted in an accuracy of 1.38m although we had less data for training compared to the models detailed in the previous sections. So, we expect to have better accuracy if we have more data for training.

Figures (51-52) displays the expected \((x, y)\) coordinates versus the predicted ones while testing the heterogeneous RSS LSTM model. The predicted coordinates are almost within the 2 meters shift of the expected path.
Figure 53: Route prediction with heterogeneous LSTM

(Figure 53) shows a simulation of the expected path compared to the predicted path while testing the heterogeneous RSS LSTM model. The predicted path shows some fluctuation at the beginning of the path and midway at the first corner.

Figure 54: Testing Wi-Fi LSTM X prediction
Both LSTM models have an average mean error of less than 1.5m, and most of the predicted coordinates are within the 2 meters offset from the predicted coordinates as shown in figures 51, 52, 54, & 55. Moreover, with the WiFi LSTM model had better accuracy in the testing phase, and this was evident when we compare the predicted path to the expected one as simulated in (Figure 56) and projected to the real environment in the Irwin 7th floor (Figure 57).

*Figure 55: Testing Wi-Fi LSTM Y prediction*
Figure 56: Route prediction with Wi-Fi based LSTM

Figure 57: Projecting model prediction to real-time environment
Using the same setup of both LSTM and GRU models, we reproduced the same models but with predicting only the current location – excluding the 4 future steps. The models now take 40 RSS vectors as an input and outputs only 1 node for the current \((x, y)\) prediction instead of 5. The runs of the RNN models are averaged after running each model for 5 times, and since this approach doesn’t affect the basic NN approach the results are kept the same for the basic NN with 10 runs. The results are summarized in (Table 9).

**Table 9: Comparing all proposed models with the current location prediction**

<table>
<thead>
<tr>
<th>Neural Network</th>
<th>Mean (m)</th>
<th>Std. Dev</th>
<th>RMSE X</th>
<th>RMSE Y</th>
<th>MAE X</th>
<th>MAE Y</th>
<th>CDF &lt;2m</th>
<th>CDF &lt;1.5m</th>
<th>CDF &lt;1m</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Basic NN</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wi-Fi/4G</td>
<td>3.907</td>
<td>1.82</td>
<td>2.962</td>
<td>3.091</td>
<td>2.456</td>
<td>2.606</td>
<td>17.85%</td>
<td>8.24%</td>
<td>1.9%</td>
</tr>
<tr>
<td>4G</td>
<td>3.329</td>
<td>1.221</td>
<td>1.963</td>
<td>2.953</td>
<td>1.876</td>
<td>2.52</td>
<td>16.05%</td>
<td>5.32%</td>
<td>0.44%</td>
</tr>
<tr>
<td><strong>Basic NN</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wi-Fi</td>
<td>4.002</td>
<td>1.997</td>
<td>3.178</td>
<td>3.115</td>
<td>2.537</td>
<td>2.612</td>
<td>18.73%</td>
<td>9.32%</td>
<td>2.63%</td>
</tr>
<tr>
<td><strong>LSTM</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wi-Fi/4G</td>
<td>1.201</td>
<td>0.836</td>
<td>0.880</td>
<td>1.056</td>
<td>0.726</td>
<td>0.762</td>
<td>90.732</td>
<td>76.829</td>
<td>45.610</td>
</tr>
<tr>
<td>4G</td>
<td>2.641</td>
<td>1.562</td>
<td>2.002</td>
<td>2.174</td>
<td>1.504</td>
<td>1.716</td>
<td>43.780</td>
<td>27.317</td>
<td>12.805</td>
</tr>
<tr>
<td><strong>LSTM</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wi-Fi</td>
<td>1.139</td>
<td>0.847</td>
<td>0.916</td>
<td>1.025</td>
<td>0.765</td>
<td>0.634</td>
<td>95.000</td>
<td>82.805</td>
<td>49.390</td>
</tr>
<tr>
<td><strong>GRU</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wi-Fi/4G</td>
<td>1.342</td>
<td>1.145</td>
<td>1.013</td>
<td>1.458</td>
<td>0.776</td>
<td>0.869</td>
<td>86.220</td>
<td>71.707</td>
<td>45.976</td>
</tr>
<tr>
<td>4G</td>
<td>2.559</td>
<td>1.576</td>
<td>1.999</td>
<td>2.184</td>
<td>1.482</td>
<td>1.690</td>
<td>46.951</td>
<td>30.122</td>
<td>14.634</td>
</tr>
<tr>
<td><strong>GRU</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wi-Fi</td>
<td>1.220</td>
<td>0.940</td>
<td>0.984</td>
<td>1.105</td>
<td>0.768</td>
<td>0.733</td>
<td>92.073</td>
<td>79.634</td>
<td>43.171</td>
</tr>
</tbody>
</table>

Following the same scheme of (Table 6), the first three rows correspond to the basic NN models, and the yellow highlighted values represent the best results among the basic NN models. Similarly, the middle three rows correspond to the 3 LSTM models where the green-highlights represent the best values among all 9 models, and the last three rows correspond to the 3 GRU models with the yellow-highlighted values highlighting the best results among the GRU models.

The LSTM model with WiFi-only RSS proved to be the best model among the other models with a mean localization error of 1.139m which is 7% better than the GRU WiFi model and 66% better than basic NN model with cellular RSS. Also, with the LSTM WiFi-based model 95% of the values are within the 2m accuracy compared to 92% for heterogeneous RSS LSTM, and 18.7% for cellular-based RSS LSTM as displayed in (Figure 58).
Inspecting the GRU models as shown in (Table 9) and (Figure 59), the values that are within the 2-meters accuracy are distributed as 92% for GRU WiFi model, 86% for the heterogeneous GRU, and 47% for the cellular-based RSS GRU models.

The CDF graphs of the 1-step prediction (Figures 58-59) prove again how the WiFi-based models are close in performance to the heterogeneous RSS models, but achieving better performance compared to the cellular-based RSS models.
Figures 60, 61, and 62 show how LSTM outperforms GRU with a small difference in terms of the mean localization error. Considering the mean error, LSTM WiFi model shows an improvement of 6.6% over GRU WiFi model, and a 66% improvement over the cellular-based RSS of the basic NN.
Comparing the two approaches (5-steps vs 1-step prediction), the 1-step prediction models achieve better results than the 5-steps models with both LSTM and GRU models.

Table 10: Comparison of 5-steps vs 1-step prediction

<table>
<thead>
<tr>
<th>Neural Network</th>
<th>Mean (m)</th>
<th>Std. Dev</th>
<th>RMSE X</th>
<th>RMSE Y</th>
<th>MAE X</th>
<th>MAE Y</th>
<th>CDF &lt;2m</th>
<th>CDF &lt;1.5m</th>
<th>CDF &lt;1m</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic NN Wi-Fi/4G</td>
<td>3.907</td>
<td>1.82</td>
<td>2.962</td>
<td>3.091</td>
<td>2.456</td>
<td>2.606</td>
<td>17.85%</td>
<td>8.24%</td>
<td>1.9%</td>
</tr>
<tr>
<td>Basic NN 4G</td>
<td>3.329</td>
<td>1.221</td>
<td>1.963</td>
<td>2.953</td>
<td>1.876</td>
<td>2.52</td>
<td>16.05%</td>
<td>5.32%</td>
<td>0.44%</td>
</tr>
<tr>
<td>Basic NN Wi-Fi</td>
<td>4.002</td>
<td>1.997</td>
<td>3.178</td>
<td>3.115</td>
<td>2.537</td>
<td>2.612</td>
<td>18.73%</td>
<td>9.32%</td>
<td>2.63%</td>
</tr>
<tr>
<td>LSTM Wi-Fi/4G 1-step</td>
<td>1.201</td>
<td>0.836</td>
<td>0.980</td>
<td>1.056</td>
<td>0.728</td>
<td>0.762</td>
<td>90.73%</td>
<td>76.83%</td>
<td>45.61%</td>
</tr>
<tr>
<td>5-steps</td>
<td>1.498</td>
<td>1.112</td>
<td>1.111</td>
<td>1.453</td>
<td>0.878</td>
<td>0.966</td>
<td>79.87%</td>
<td>64.44%</td>
<td>37.12%</td>
</tr>
<tr>
<td>LSTM 4G 1-step</td>
<td>2.641</td>
<td>1.562</td>
<td>2.002</td>
<td>2.174</td>
<td>1.504</td>
<td>1.716</td>
<td>43.78%</td>
<td>27.3%</td>
<td>12.8%</td>
</tr>
<tr>
<td>5-steps</td>
<td>2.997</td>
<td>1.611</td>
<td>2.096</td>
<td>2.592</td>
<td>1.428</td>
<td>2.267</td>
<td>29.62%</td>
<td>16.94%</td>
<td>8.31%</td>
</tr>
<tr>
<td>LSTM Wi-Fi 1-step</td>
<td>1.139</td>
<td>0.847</td>
<td>0.916</td>
<td>1.025</td>
<td>0.765</td>
<td>0.634</td>
<td>95%</td>
<td>82.8%</td>
<td>49.39%</td>
</tr>
<tr>
<td>5-steps</td>
<td>1.338</td>
<td>1.013</td>
<td>1.090</td>
<td>1.17</td>
<td>0.884</td>
<td>0.789</td>
<td>87.87%</td>
<td>75.31%</td>
<td>42.56%</td>
</tr>
<tr>
<td>GRU Wi-Fi/4G 1-step</td>
<td>1.342</td>
<td>1.145</td>
<td>1.013</td>
<td>1.458</td>
<td>0.776</td>
<td>0.869</td>
<td>86.22%</td>
<td>71.71%</td>
<td>45.98%</td>
</tr>
<tr>
<td>5-steps</td>
<td>1.635</td>
<td>1.343</td>
<td>1.151</td>
<td>1.750</td>
<td>0.884</td>
<td>1.097</td>
<td>77.12%</td>
<td>61.62%</td>
<td>36.44%</td>
</tr>
<tr>
<td>GRU 4G 1-step</td>
<td>2.559</td>
<td>1.576</td>
<td>1.999</td>
<td>2.184</td>
<td>1.482</td>
<td>1.690</td>
<td>46.95%</td>
<td>30.12%</td>
<td>14.63%</td>
</tr>
<tr>
<td>5-steps</td>
<td>2.731</td>
<td>1.752</td>
<td>2.212</td>
<td>2.456</td>
<td>1.622</td>
<td>1.7</td>
<td>42.37%</td>
<td>29.19%</td>
<td>15.37%</td>
</tr>
<tr>
<td>GRU Wi-Fi 1-step</td>
<td>1.220</td>
<td>0.940</td>
<td>0.984</td>
<td>1.105</td>
<td>0.768</td>
<td>0.733</td>
<td>92.07%</td>
<td>79.63%</td>
<td>43.17%</td>
</tr>
<tr>
<td>5-steps</td>
<td>1.523</td>
<td>1.114</td>
<td>1.195</td>
<td>1.325</td>
<td>0.988</td>
<td>0.887</td>
<td>80%</td>
<td>62.87%</td>
<td>36.06%</td>
</tr>
</tbody>
</table>

(Table 10) summarizes all the results conducted in this study, and the green-highlighted values are the best values compared to all models. LSTM WiFi-based RSS with 1-step (1.139m) is 15% better than the 5-steps model (1.338m) in terms of the mean localization error, similarly the GRU WiFi-based RSS with 1-step (1.220m) is 20% better than the 5-steps GRU model (1.523m).

Also comparing LSTM to GRU, the 1-step LSTM WiFi model is 7% better than the 1-step GRU WiFi model. Similarly, the 1-step LSTM WiFi/Cellular model is 10% better than the 1-step GRU WiFi/Cellular model.
The accuracy improvement is displayed in the CDF (Figures 63 till 65) when comparing the LSTM 1-step prediction models to the LSTM 5-steps models of the heterogeneous based RSS (Figure 63), WiFi-based RSS (Figure 64), and cellular-based RSS (Figure 65).
Figures (66-68) show the CDF comparison of the 1-step GRU versus the 5-steps GRU models with the heterogeneous based RSS (Figure 66), WiFi-based RSS (Figure 67), and cellular-based RSS (Figure 68). Hence, the 1-step forecasting outperformed the 5-steps prediction as discussed with both RNN models.
Testing the 1-step models, we split the dataset again into 74.34% training, 18.46% validation, and 7.2%. The results are summarized in (table 1) while testing the LSTM WiFi-only model and LSTM WiFi/cellular model.

<table>
<thead>
<tr>
<th>Neural Network</th>
<th>Mean (m)</th>
<th>Std. Dev</th>
<th>RMSE X</th>
<th>RMSE Y</th>
<th>MAE X</th>
<th>MAE Y</th>
<th>CDF &lt;2m</th>
<th>CDF &lt;1.5m</th>
<th>CDF &lt;1m</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM RNN Wi-Fi/4G</td>
<td>1.427</td>
<td>0.945</td>
<td>1.262</td>
<td>1.157</td>
<td>1.06</td>
<td>0.76</td>
<td>86.2%</td>
<td>58.62%</td>
<td>34.48%</td>
</tr>
<tr>
<td>LSTM RNN Wi-Fi</td>
<td>1.322</td>
<td>0.383</td>
<td>1.137</td>
<td>0.777</td>
<td>1.019</td>
<td>0.649</td>
<td>96.55%</td>
<td>65.52%</td>
<td>13.79%</td>
</tr>
</tbody>
</table>

Since we have less data for training in the testing phase, the mean error was expected to be a bit higher compared to the training/evaluation phase. As per the testing results of (Table 1), LSTM WiFi model outperforms the heterogeneous LSTM model.

The 1-step prediction with WiFi LSTM achieves a mean error of 1.322m compared to 1.383m of the same model with the 5-steps prediction (4% improvement). The standard deviation shows 39% improvement compared to the 5-steps model.

After testing the 1-step prediction with the WiFi-only RSS based LSTM model, the predicted path is compared to expected path (Figure 69), and the paths are then projected to the real-time environment (Figure 70).
Figure 69: 1-step Route prediction with Wi-Fi based LSTM

Figure 70: Projecting model prediction to real-time environment 1-step

The predicted path shows better response when going over the corners, and better accuracy with 96.5% of the values within the 2-meters accuracy compared to 75.8% with the 5-steps prediction model.
Comparing our work to (Hoang et al.), in terms of mean localization error the results in (Hoang et al.) achieved a better mean error of 0.75m compared to 1.14m in this study. The improvement of the results goes back to many reasons compared to the limitations that we have. In the WiFi-only RSS we relied on 2 wireless access points covering the whole floor of an area of 374m², the time (Hoang et al.) deploys 6 APs covering almost the same area. In practical scenarios, when more APs are used, then more RSSI features can be extracted and better performance can be achieved (Hoang et al.) however increasing the number of APs also creates more computational cost and extends the training time. Moreover, in the latter study, the RSSI data for both training and testing was collected using an autonomous driving robot that has multiple sensors including. The sensors included a wheel odometer, an inertial measurement unit (IMU), a LIDAR, sonar sensors, and a color and depth (RGB-D) camera where it can navigate to a target location within an accuracy of 0.07±0.02m (Hoang et al.). Also, the reference points in (Hoang et al.) are 365 compared to 121 RPs in our study, and the robot in (Hoang et al.) recorded 100 scans of RSSI measurements per RP. Moreover, the total number of random trajectories was 365,000 compared to 50 random trajectories in our study with data collected manually. One downside in the study conducted in (Hoang et al.) is that the first (x, y) location of each trajectory has to be inputted to the model in addition to the RSS vectors, whereas in our case we only take any continuous 40 RSS vectors and output the current location or the current location with 4 future steps to be predicted.
Chapter Five

Conclusion

In Conclusion, this work has tackled open space indoor navigation following a heterogeneous RSS fingerprinting-based approach. Transduction neural networks were adopted in this study due to their advantage in building up long-term dependencies within the input sequence and correlating them with the output sequence. Taking advantage of these correlations, LSTM and GRU RNN models are deployed in this study as an input a sequence of RSS vectors, and as an output sequence of consequent \((x, y)\) coordinates. Comparing the RNN models to a non-recurrent model, the transduction neural networks outperformed the latter model achieving higher localization accuracy. Furthermore, dissecting the heterogeneous RSS fingerprints into cellular-only RSS and WiFi-only RSS, the proposed models were examined and evaluated to test for the best model with the highest localization accuracy – i.e. getting the lowest mean localization error. The results demonstrate that the combination of WiFi and cellular RSS fingerprints achieving promising results with a mean localization error of 1.2m with LSTM where 91% of the values are within the 2-meters accuracy compared to 1.1m and 95% with LSTM WiFi-only model. With no requirement of line-of-sight between the anchor node and the user device, and with no prior knowledge of the \((x, y)\) coordinates in the online phase, the combination of cellular and WiFi RSS fingerprints achieved promising results in light of the limitations that we had in this study. As transduction neural networks prove to serve well the goals of the time series forecasting in indoor navigation, it might be interesting to see the impact of adding “Attention” (Vaswani et al.) into such kind of problems in a future work, where some RSS vectors might gain more attention and play a major role in giving better predictions.
References


