DEEPSPOT: INTELLIGENT SENSOR FUSION FOR INDOOR LOCALIZATION VIA DEEP SPATIOTEMPORAL NEURAL NETWORKS

By

RAHUL SENGUPTA

A THESIS PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

UNIVERSITY OF FLORIDA

2017

To my family, friends, teachers and the Illuminated Ones

ACKNOWLEDGMENTS

I am grateful to Prof. Andy Li, Prof. Jose Principe, Prof. Daisy Wang, Prof. Wei Zhang for their support and guidance, John Fodero and Shiv Rajora for the robotics and hardware aspects of the thesis, and members of LI lab for their general help with various technical issues. I'm also grateful to my family, especially my parents and my sister, my relatives, friends and teachers, who have given me their constant love and support. I would also like to thank the fine people associated with the University of Florida and BITS-Pilani University who have helped me advance in my academic pursuits. Finally, I'd like to thank all those who have helped build our human civilization, especially the people of India and the United States of America.

TABLE OF CONTENTS

page

LIST OF TABLES

Table page

LIST OF FIGURES

LIST OF ABBREVIATIONS

GPS	Global Positioning System
GLONASS	Globalnaya Navigazionnaya Sputnikovaya Sistema
LORAN	Long Range Navigation
WiFi	Wireless Fidelity

Abstract of Thesis Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Master of Science

DEEPSPOT: INTELLIGENT SENSOR FUSION FOR INDOOR LOCALIZATION VIA DEEP SPATIOTEMPORAL NEURAL NETWORKS

By

Rahul Sengupta

December 2017

Chair: Xiaolin Li
Major: Computer Science

Given the pervasiveness of WiFi networks within indoor spaces, there arises the possibility of using WiFi signals for indoor localization. In this work, we start by performing a literature survey of the existing approaches and start work on a practical and scalable system for commercial and industrial use. We use an off-the-shelf robot with commodity hardware and sensors to create the dataset. We operate it in a semi-autonomous manner in the corridors of a university office building and map unrestricted hallways on two floors to create a large dataset of over 6000 individual readings of over 200 locations, spaced 3 feet (0.9144m) apart. The robot collects WiFi Received Signal Strength (RSS) readings of available Access Points, magnetometer heading and scene images (captured using two fisheye cameras). We then explore the performance of various deep neural network architectures and present their results. We see that deep learning models can effectively perform indoor localization in corridors, which can be used for location estimation by people and autonomous robots operating inside of large indoor complexes

CHAPTER 1
INTRODUCTION

In a world that is constantly on the move, knowing one's location is of paramount importance. In the olden days, navigators used various tools like sextants and positions of celestial objects to cross vast oceans. Lighthouses were a literal beacon of hope for wary seafarers. But in the past century, we have witnessed a range of advances in the field of navigational technologies, mainly due to advances in radio technology, electronics and computers.

Figure 1-1. Prof. Hermann Thiersch. Pharos of Alexandria. 31 December 1908. Source: Public Domain image reprinted from Wikimedia Commons, https://commons.wikimedia.org/wiki/File:Lighthouse_-_Thiersch.png, (accessed 25 October 2017).

One of the oldest types of navigation relies on a method called "dead reckoning". In this method, previously determined positions are used, along with other current and past quantities (such as speed and direction), are used to get an estimate of the present location. In the animal kingdom, foragers like ants, geese and rodents, that have a fixed home base are known to use this method. It is hypothesized that special neural structures in the nervous system aid this path integration [1, 2]. Beyond the natural world, this method was widely used by navigators of both early ships, airplanes and

even ballistic missile systems [3]. However, its major drawback is the accumulation of errors. Since every successive estimate includes previous errors, the errors can compound and greatly affect future accuracy [4, 5].

Thus, to ensure accuracy over a sustained period of time, non-inertial navigation systems are used. These rely on an external phenomenon from which the navigating agent can determine its position [6]. Lighthouses of the ancient times served such a purpose. In the early 20th century, the lighthouse's modern reincarnation the radio beacon, transmits radio waves from a fixed known position, allows ships and airplanes equipped with a radio set to tune in and work out a position estimate. A number of coordinated radio beacons along with the use of microcontrollers in the mid-20th century led to the development of larger radio navigation networks such as LORAN (for use by the United States military) [7].

However, with the advent of the space age, satellite-based radio navigation technologies started gaining momentum [8, 9]. The first operational satellite navigation system was TRANSIT system, operated primarily by the United States Navy, was first launched in 1959 [10].

Successive improvements to the system led to the birth of the NAVSTAR GPS or better known as "Global Positioning System" in 1978 [11, 12, 13]. While this system was originally reserved for use by the United States military, it was made available for public use in 1983 when a passenger airline was shot down after accidentally straying into prohibited airspace. Other such systems [14] were developed such as GLONASS by the former Soviet Union in 1982, as well as regional systems like BeiDou by China and

NAVIC by India. While GLONASS is large-scale reach, others are mainly concentrated over their country's landmass.

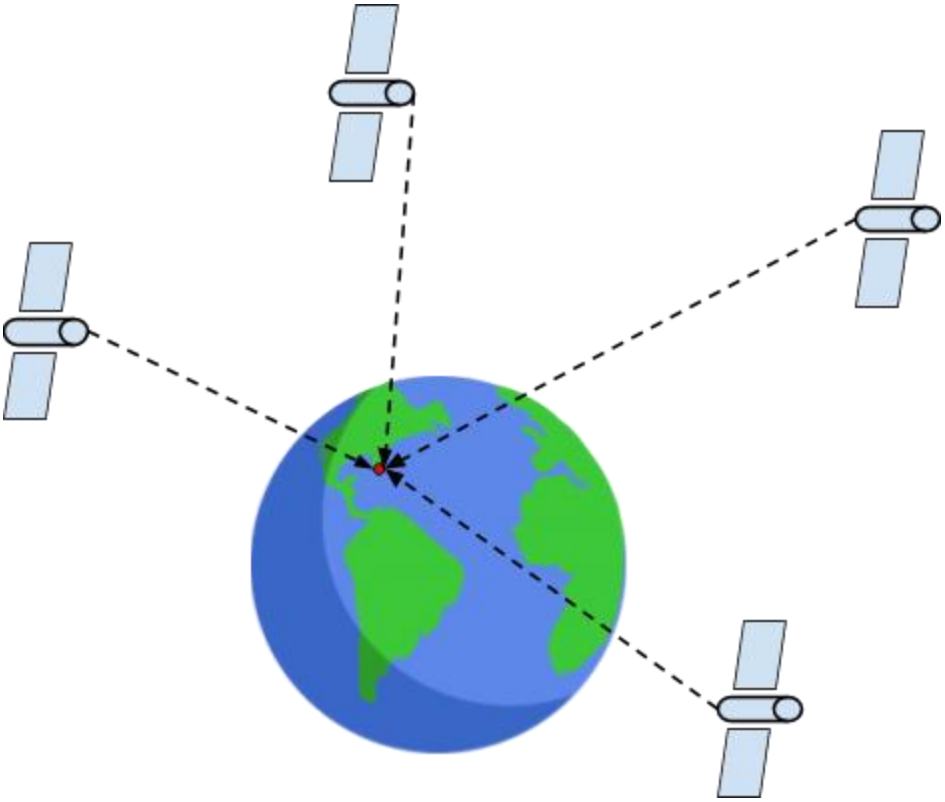

Figure 1-2. Global Positioning System

Today satellite-based positioning systems, especially the GPS, dominate the field of localization, with accuracies of 7.8 meter (95% Confidence Interval) for the civilian version of GPS. However, GPS has significant issues when used in multi-level indoor environments. The GPS signal is attenuated and scattered by walls and roofs of structures, leading to very imprecise and unstable location fixes [15].

However, with industrialization and urbanization on the rise world over, there is a growing need for indoor localization technologies. Given the proliferation of wireless mobile devices such as smartphones among the general public, as well as the ever-growing use of autonomous mobile robots (such as in warehouses and industrial complexes), indoor localization is more important than ever. Such technologies open up

vast potential for indoor localization applications [16, 17], especially in the fields of logistics [16], disaster management, industry, surveillance etc.

Based on the mode of localization, indoor localization [18] can be classified into:

- Device-based localization (DBL), in which the user device uses some Reference Nodes (RN) to obtain its location with respect to a predefined coordinate system. DeepSpot can be considered a DBL-based solution, as it uses the RSS of various WiFi access points.

- Monitor-based localization (MBL) in which a set of sensing nodes passively obtain the position of the user or entity emitting some characteristic for which the sensing nodes are monitoring. An example of this is the Guoguo system [19] which uses a network of acoustic sensors to achieve an average localization accuracy of about 6~25cm in typical office and classroom environments.

- Proximity-based Localization in which only the distance between a user and a Point of Interest (PoI) is detected. Such services are often used to provide advertising in shopping malls etc.

Several technologies are used to perform indoor localization. Some popularly-used ones are:

- WiFi, more formally the IEEE 802.11 standard [20], commonly known as WiFi, is primarily used for enabling wireless networking capabilities to mobile devices. It is widely used in private and public indoor spaces. While initially, WiFi had a reception range of about 100 meters, the newer versions have a range of up to

1000 meters. Using properties of WiFi signals, especially the Received Signal Strength, it is possible to estimate the approximate location of a user.

- Bluetooth, more formally the IEEE 802.15.1 standard [21], consists of specifications for connecting different fixed or moving wireless devices (such as mobile phones and laptops) within a certain personal space. Bluetooth Low Energy, which is the latest version of Bluetooth [22], has been used for context-aware proximity-based services, with prominent examples being iBeacons by Apple and Eddystone by Google [18].

- Radio Frequency Identification Device (RFID), which is primarily for storing and transmitting relatively small amounts of data using a Radio Frequency (RF) compatible circuit [23]. One of the common uses is in supermarkets for preventing theft [24]. An RFID system consists of two components, an RFID and RFID tags. The reader can communicate with RFID tags while the RFID tags emit data that the RFID reader can read.

- Ultra-Wideband (UWB), which uses extremely short radio bursts (<1 ns) but over a large bandwidth (>500 MHz), in the frequency range of 3.1 to 10.6 GHz. UWB techniques have been shown to be immune to noise and fading effects in indoor settings [25, 26].

- Ultrasound, which mainly relies on Time of Flight measurements of ultrasonic signals and sound velocity to calculate the distance between a transmitter and receiver node(s) [18]. While this approach has shown to achieve fine-grained

centimeter-level accuracy in indoor environments, it requires significant acquisition of specialized hardware, especially in the form of acoustic sensors.

For DeepSpot, we rely on WiFi RSS as our primary means of localization, given the ease of data collection and its widespread use. We design DeepSpot with practicality and scalability in mind. We use an off-the-shelf Kobuki robot to perform data collection in a semi-autonomous mode. We thus make a dataset of over 6000 individual readings of over 200 location points separated by 3 feet (.9144 meters) each, in the hallways of 4th and 5th floors of New Engineering Building, University of Florida, United States of America. The hallways provide a representative dataset which can be used for indoor localization in college campuses, hospitals, office spaces, malls, apartment complexes etc. We also analyze our dataset and use deep neural networks equipped with a neural memory to perform indoor localization. Our end goal is to build an indoor localization system that can be widely adopted by industrial and commercial entities in an affordable and uncomplicated manner.

In the next chapter, we shall briefly study various approaches to indoor localization using WiFi.

CHAPTER 2
INDOOR LOCALIZATION USING WIFI

WiFi is a wireless technology used for local area networking. It is based on the IEEE 802.11 standards [20], which are a set of standards created and maintained by the Institute of Electrical and Electronics Engineers (IEEE). These standards specify how Medium Access Control (OSI Model Layer 2) and Physical Layer (OSI Model Layer 1) should be implemented for wireless local area network (WLAN) [27, 28]. WiFi is widely used indoors in residential, commercial and public indoor spaces worldwide.

WiFi's range is usually 20 meters indoors and several WiFi access points are usually required to provide coverage for a large indoor space. WiFi commonly uses frequency bands of 2.4 GHz and 5 GHz. Most modern mobile devices such as mobile phones and laptops, usually support both these bands, if not just 2.4 GHz.

While WiFi is primarily used as a means of sending data (usually over the Internet), its use to perform indoor localization has been extensively studied [29]. Different aspects of WiFi signals are used for indoor localization, such as Received Signal Strength(RSS), Channel State Information, Angle of Arrival, Time of Flight, Time Difference of Arrival, Return Time of Flight, Phase of Arrival etc. [30, 31]. Given the simplicity and cost-effectiveness of using RSS based approach, we will use RSS data for fingerprinting.

2.1 Received Signal Strength

Received Signal Strength [29] is the strength of the power of a WiFi signal as received at the receiver. Usually, WiFi RSS is measured in decibel-milliwatts (dBm). On the other hand, Received Signal Strength Indicator (RSSI) is a relative measure of the

RSS which can vary across WiFi chip vendors. RSSI can be used to find an approximate distance between the WiFi transmitter and WiFi receiver with the formula

$$RSSI = -10n\log_{10}(d) + A$$

where d is the distance between the transmitter and receiver, n is the path loss exponent (2 for outdoors to 4 for indoor environments), and A is RSSI value at reference point.

Using RSS of several different WiFi transmitters with known locations, it is possible to estimate distances from an unknown receiving point using geometry (trilateration or more generally n-point lateration) [32]. However, this value can be significantly inaccurate given the presence of objects (such as walls, roofs, doors) that can reflect or attenuate WiFi signals, differently in different areas and directions. Furthermore, the user performing localization needs to know the locations of the transmitting WiFi access points.

2.2 Fingerprinting-based localization

Fingerprinting-based localization techniques using WiFi can mitigate the need to know positions of the transmitters and layout of the indoor space [29]. In order to perform a fingerprinting-based approach, a survey of the locale must be done in order to map certain features such as the RSS of different WiFi access points at various locations in the indoor space. This is done during the offline phase and is usually only done from time to time. After the system is deployed, in the online phase, measurements are collected in real-time and are compared in some manner to previously obtained offline measurements. There are several methods that can be used to perform the matching between online and offline measurements.

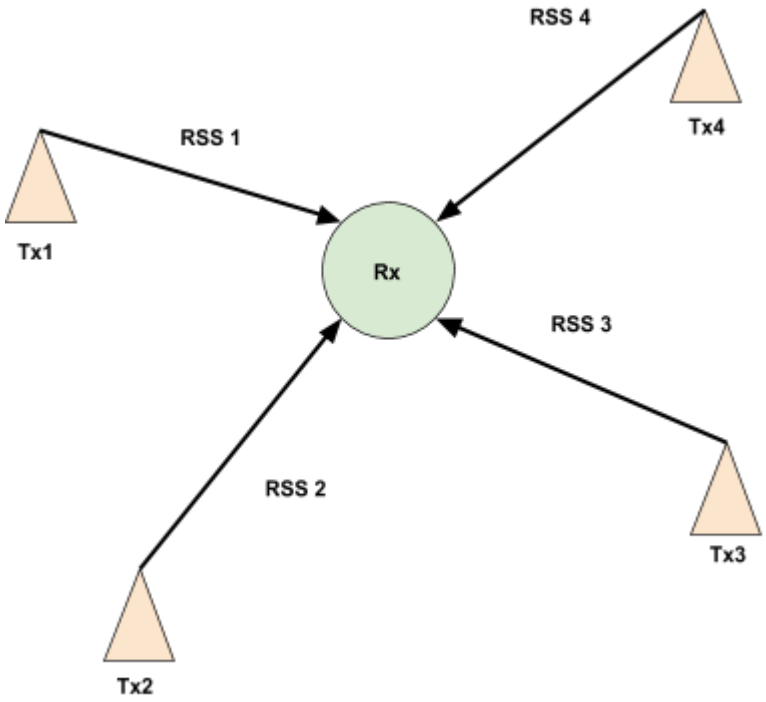

Figure 2-1. Mapping of WiFi RSS

2.2.1 k-Nearest Neighbors (kNN)

k-Nearest Neighbor algorithm relies on comparing the root mean square error(RMSE) of the query RSS reading vector (ordered set of RSS readings from different WiFi access points) against all previously collected offline RSS readings [33]. The k vectors that give the lowest RMSE are considered as the best matches and their average location coordinates is the estimated location of the query RSS reading vector.

2.2.2 Support Vector Machine (SVM)

Support Vector Machines are supervised learning algorithms that perform classification and regression by constructing a set of hyperplanes in high or infinite dimensional space in order to separate different classes of training data [34, 35]. Very often the “kernel trick” [35] is used to compute this mapping in a computationally-viable manner. The RSS vectors collected during the offline phase are used to train an SVM

(often with a Radial Basis Function kernel) and this trained SVM is used to predict query RSS vectors coordinates.

2.2.3 Probabilistic methods

Probabilistic methods predict the likelihood of a user being at a certain location based on the RSS values obtained [36]. This likelihood can be calculated by using histograms and kernel methods. Often a gaussian prior is used for calculating the likelihood of the location. Given the vector of RSS values r obtained from n beacons (WiFi access points), the Maximum Likelihood algorithm computes the apriori probability of receiving such an r for each potential position coordinates. The coordinates that maximizes the probability can be considered as the estimated position.

2.2.4 Neural Networks

Neural Networks [37, 38] (or Artificial Neural Networks) is a learning algorithm said to be inspired by the working of biological neurons in the brain. They consist of a large number of interconnected processing units that change their output state based on the incoming inputs. These outputs are then fed into further units. The entire network is usually trained using gradient descent. When a large number of neurons are arranged in hierarchical layers, this paradigm is referred to as Deep Learning [39]. Both deep and shallow networks can be used for classification and regression.

One of the WiFi signal characteristics, the CSI (channel state information) [40], can be used to improve accuracy. CSI describes how a signal propagates from the transmitter to the receiver. CSI data provides the attenuations and phase shift of a wireless connection [41].

2.3 Channel State Information

Channel State Information is defined as:

$$h = |h| \, exp(j \, sin(\theta)) ,$$

where h is the CSI value of a subcarrier, θ is its phase

Some of the best localization accuracies obtained using indoor WiFi fingerprinting use CSI. Yet this is limited as it requires a specific network interface cards (such as Intel 5300 NIC) to obtain CSI information. Such cards may not be available on low-end WiFi-enabled mobile devices like commodity smartphones and laptops. However, the RSS information is easily available when using any general WiFi-enabled device. We thus use RSS characteristic of WiFi to perform indoor localization. Our methods are general enough to be adapted for the use of Channel State Information as well as other modalities that may come up in the future.

We shall discuss neural networks and deep learning in detail in the next chapter.

CHAPTER 3
DEEP NEURAL NETWORKS AND NEURAL MEMORY

In the field of Machine Learning, Neural Networks (Artificial Neural Networks) [38] are a class of models that can be said to be inspired by biological neurons inside the human nervous system. Neural Networks consist of systems of interconnected units (called “neurons”) which take inputs from other similar units and give a single output. The connections have weights that can be tuned based on the data, making neural nets capable of learning. The tuning is done through a technique called backpropagation based on gradient descent [42, 43].

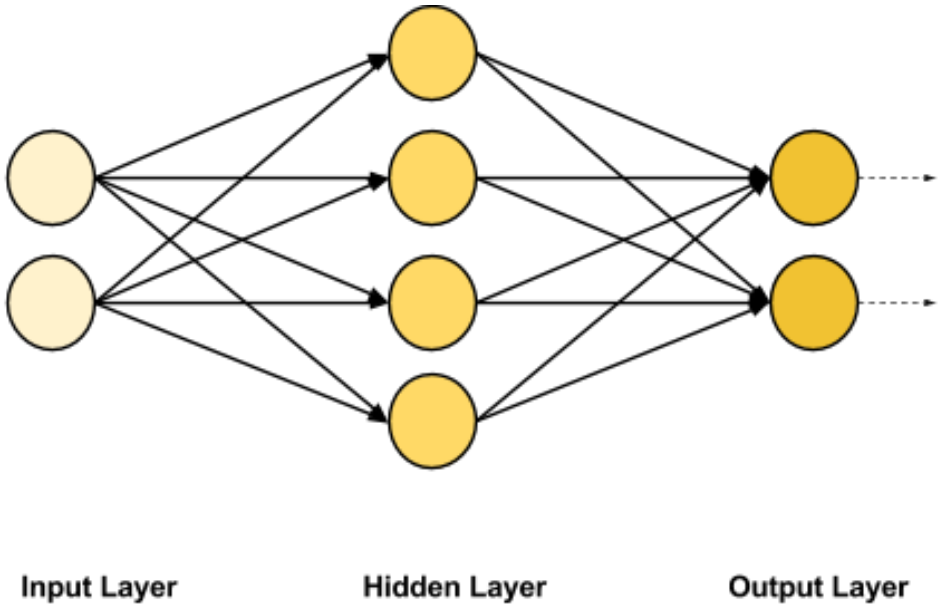

Figure 3-1. An Artificial Neural Network.

$$a^l = \sigma(z^l), \text{ where } z \equiv w^l a^{l-1} + b^l$$

$$\text{Cost } C = C(a^L) \text{ at the output layer } L$$

$$\delta^L = \nabla_a C \odot \sigma'(z^L) \text{ at the output layer } L$$

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$$

Deep Learning [39] is a way of connecting neural networks into large layered and structured networks. Deep Learning is a representation-learning method that consists of structures that learn a hierarchy of representations, starting from simple features to higher and more abstract ones [44, 45]. Popular Deep Learning architectures are:

- Convolutional Neural Networks
- Recurrent Neural Networks
- Deep Autoencoders

3.1 Convolutional Neural Network

CNN resembles a Multi-Stage Hubel-Wiesel Architectures based on Hubel and Wiesel's classic 1962 work on visual cortex of cats and monkeys [46]. In a CNN, kernels perform convolution and convolution of kernels with input results into feature maps. Once feature maps are created, a pooling layer follows [47]. A typical pipeline of this architecture is illustrated.

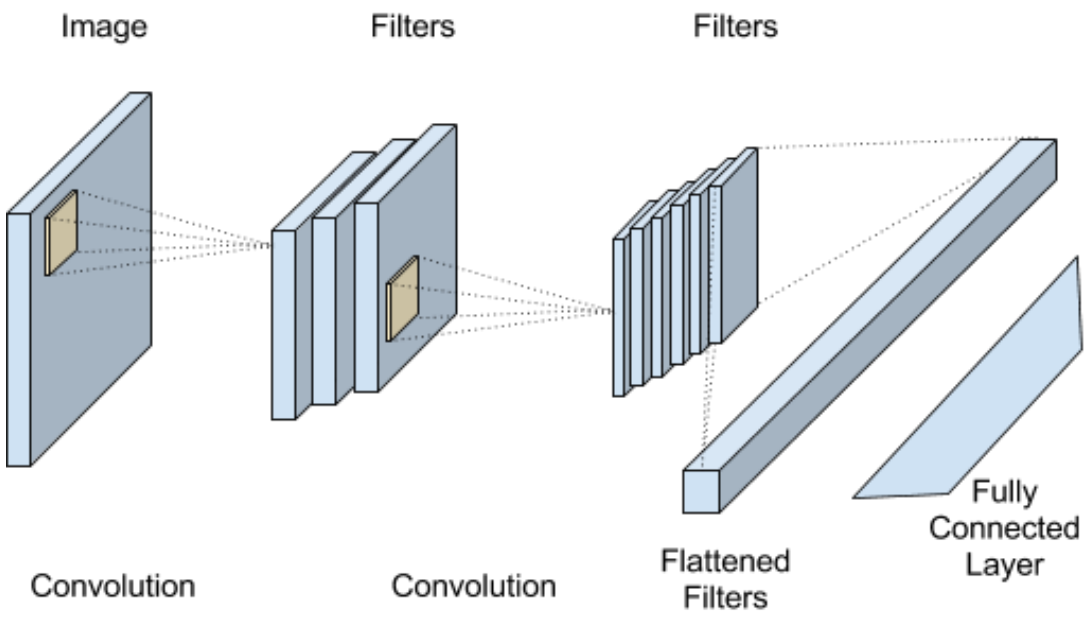

Figure 3-2. A Convolutional Neural Network.

For training CNNs, various optimization methods, such as Stochastic gradient descent (SGD), AdaGrad, AdaDelta, RMSProp, Adam [48] are used. To produce nonlinear output there are various activation functions proposed. The sigmoid function was the most popular activation function whose output ranges from 0 to 1. Most recently, Rectified Linear Unit (ReLU) [49] became much popular because of its sparsity and lesser gradient vanishing problem.

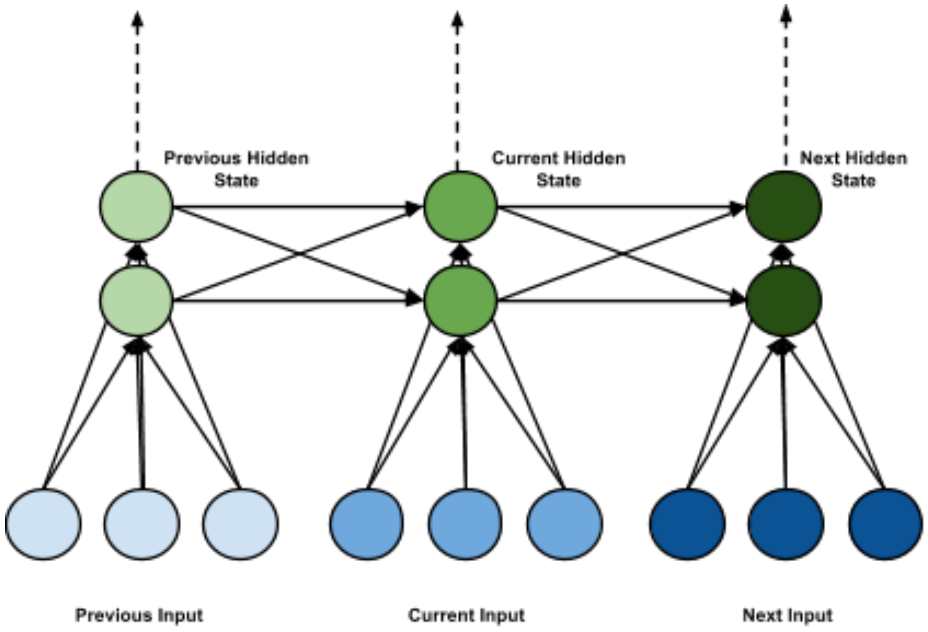

Figure 3-3. A Recurrent Neural Network.

3.2 Recurrent Neural Network

A recurrent neural network (RNN) is a neural network in which connections between units form a directed cycle. This allows the network to exhibit dynamic temporal behavior.

One of the issues with using regular RNN's is that they are unable to learn long-term dependencies. The reason for this is the vanishing of gradient [51].

$$h_t = relu(Ux_t + Wh_{t-1})$$

$$o_t = \varphi(Vh_t)$$

where h is the hidden state, x is the present input,

o is the output, φ is an output embedding network

To mitigate it, gated cells such as Long Short-Term Memory (LSTM) [52] and GRU [53] were created. This allows the overall architecture to remember when the input is significant as well as forget information.

For example, an LSTM network contains special gated LSTM blocks in addition to regular artificial neurons.

An LSTM unit has the following gates:

- Forget Gate: Determines how much cell state information to keep
- Input Gate: Decides what new information should be added to the cell state.
- Output Gate: Decides how much of the cell state should be sent ahead.

The current cell state is a sum of the previous cell state regulated by the forget gate and the intended value to be added regulated by the input gate. The output gate regulates the final output based on the cell state.

The GRU is a computationally efficient advancement of the LSTM and has only two gates (reset and update gates).

3.3 Autoencoder

An Autoencoder [54, 55] is an unsupervised learning algorithm that tries to replicate the input after passing it through an information bottleneck. It is made of several layers that

first compress the signal by putting the signal through an information bottleneck. This bottleneck can be imposed by having fewer neurons in the succeeding layers in the first half of the autoencoder (i.e. the encoder). It can also be imposed by having a sparsity criterion, by which only a certain fraction of neurons is allowed to be active. The model then tries to recreate the signal as close as possible to the original signal. This forces the model to learn features that can be used to effectively represent the data, despite the information bottleneck. The autoencoder is trained using the backpropagation algorithm.

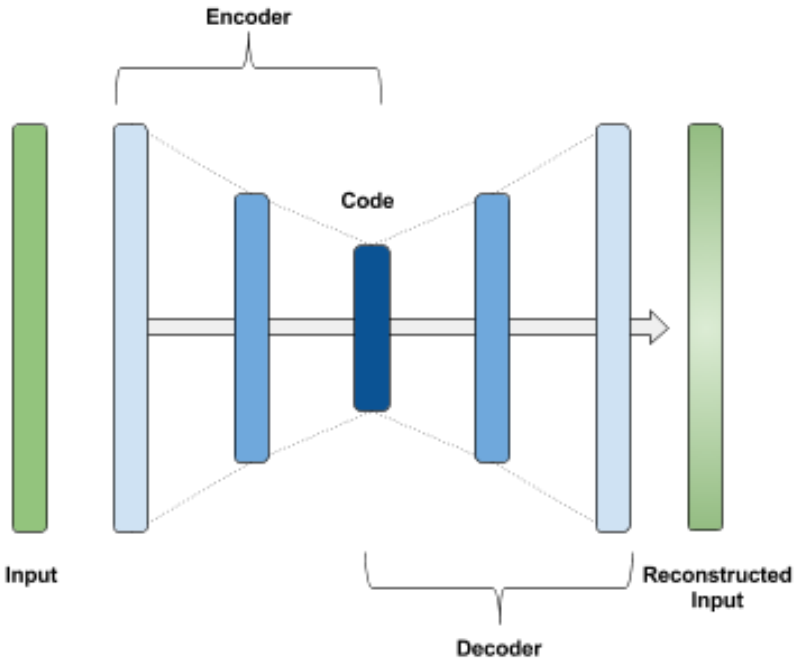

Figure 3-4. An Autoencoder.

Beyond these basic architectures, there are also advanced architectures such as the attention mechanism and the neural memory.

3.4 Attention Mechanism

An attention mechanism [56] can be thought of as comparing a query to a set of input vectors, resulting in a set of output vectors that are the inputs multiplied by a

weight, where the weight assigned to each input vector is computed by a compatibility function of the query and the input vectors.

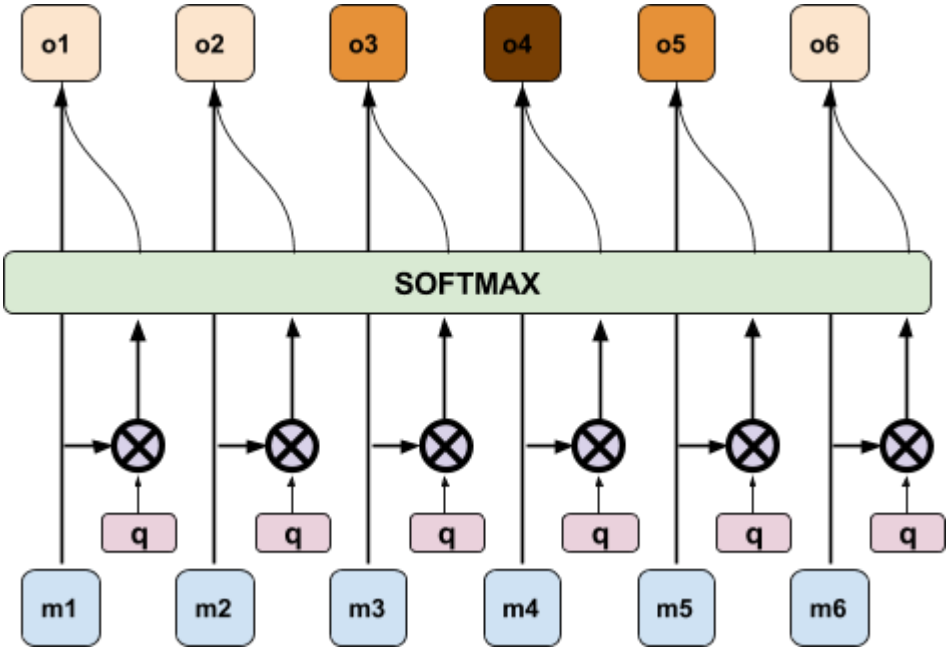

Figure 3-5. Attention Mechanism.

Let $(m_1 , m_2, ..., m_n)$ *be a set of inputs to which attention has to be applied*

Let q *be the query which is to be compared with the inputs*

Let $(o_1 , o_2 , ..., o_n)$ *be the outputs after softmax attention* m_i *has been applied*

Then $x_i = q \, . \, m_i$ *and* $o_i = m_i \, . \, softmax(x_1 , x_2 , ..., x_n)$

3.5 Neural Memory

Neural memories [57, 58, 59, 60, 61, 62] are data structures that are coupled with deep neural architectures. They can be written to and read from and can store information across time

The neural memory that we wish to use is based on the paper “Learning to Remember Rare Events” [58] by Kaiser et. al. and it consists of several components.

We have a matrix of memory keys, K, which consists of “keys”, which are abstract representation vectors, of the data that has been seen before. These keys are generated by an embedding network, which can be a CNN or an RNN or (as in our case) a feedforward network.

We define the size of the memory (mem-size) and the size of the keys (key-size) as hyperparameters. The size of the memory is the number of memory slots where entries can be written into. The key-size is the length of any one individual slot where abstract representation of different data points is stored. These keys typically occupy the bulk of the memory space.

We have a vector of memory values V, which consists of “values”, usually class labels, for each key entry in the memory. Every memory slot has a corresponding label value. Usually they are unsigned integers, though they can be a collection of real numbers. Usually, -1 is used as the default label when a certain memory slot key value has not been written. In classification task, it is these values that are output as the result of the entire network.

Then, vector A, that keeps track of the age of items stored in memory. Every memory slot will have an age corresponding to it, indicating when it was last updated.

We now describe the overall idea of using this type of neural memory, stepwise and describe the equations in detail.

An input data exemplar is presented to the network. The exemplar is first sent through an embedding network. This embedding network could be an architecture like a feedforward network, convolutional neural network or a recurrent neural network. The

job of this embedding network is to parse the input and transform it into a query vector to the memory.

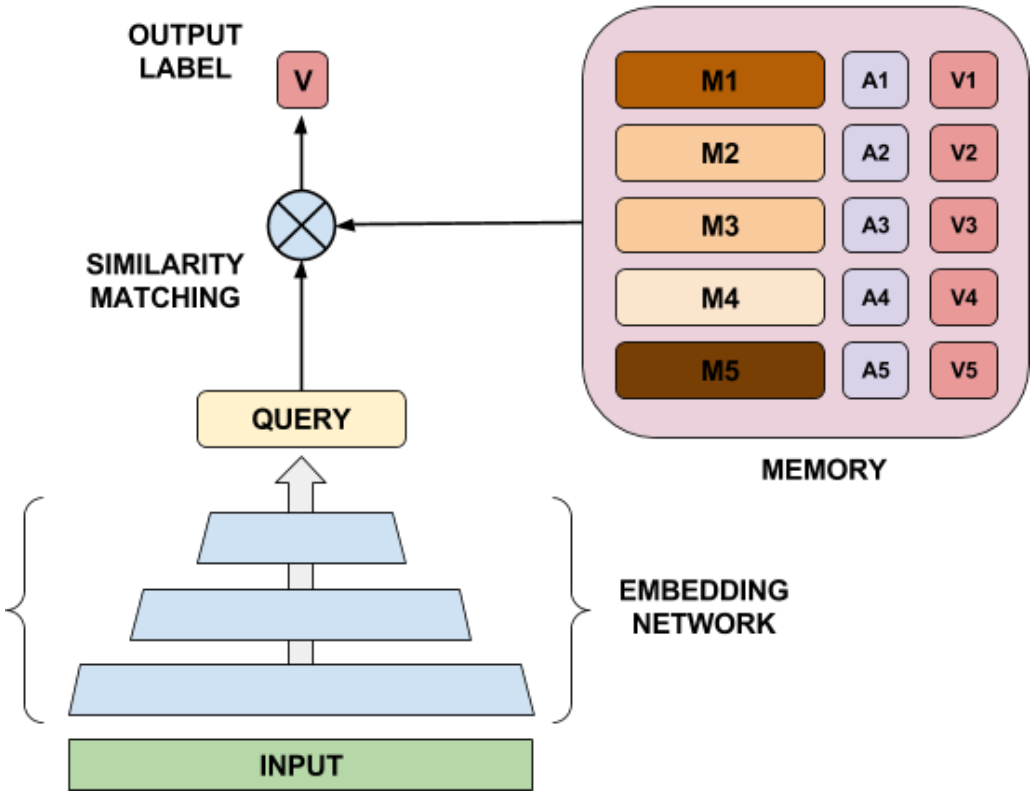

Figure 3-6. Neural Memory.

Whether this embedding network is a convolutional neural network, a recurrent neural network or a feed-forward neural network, will depend on the type of data exemplar input. The end result of this embedding network is a 1D vector query vector of fixed size. This size is identical to the key vectors in the memory. This must be ensured to be able to find the cosine similarity between the query vector and each and every key vector in the memory slots.

Given an input exemplar x, *query vector* q *is given by*

$$q = \varphi(x)$$

where φ *is the embedding network*

We set up the memory module M with all key vectors K as zero vectors, all label values V as -1 and all ages A as zero

$$M = (K_{mem-size \times key-size}, V_{mem-size}, A_{mem-size})$$

where M is the memory

We ensure that all query vectors (and thus the key vectors) are normalized i.e.

$$|q| = 1$$

Given a query vector, we define the nearest neighbor of it in the memory as any of the keys that maximize the dot product with the query vector i.e. we find the cosine similarity between the query vector and all the keys in the memory. We find top-g such nearest neighbors

$$(n_1, n_2, n_3 \ldots\ldots, n_g) = NN_g(q, M)$$

The main result is returned as the label value at the memory location which topped the nearest neighbor list

$$Output = V[n_1]$$

This concludes the forward pass operation using the neural memory. Now we consider the calculation of the loss. The loss function is calculated based on the memory output. Given the correct ground truth label v, we can have either the case that it matches the label value returned by the memory or it doesn't. We will calculate the loss and update differently in both cases.

Let us define p to be the smallest index such that label value at that index in the memory matches the ground truth label. Let us also define b as the smallest index such that the label value at that index in the memory does not match the ground truth label. We state the loss function and update rule as follows:

Case 1

$$If\ V\left[n_1\right] = v,\ then\ Loss = \left[q.k_b - q.k_1 + \alpha\right]_+$$

$$Update\ K\left[n_1\right] \leftarrow \frac{q + k_1}{|q + k_1|}\ ,\ A\left[n_1\right] \leftarrow 0$$

Case 2

$$If\ V\left[n_1\right] \neq v,\ then\ Loss = \left[q.k_1 - q.k_p + \alpha\right]_+$$

$$Update\ K\left[n'\right] \leftarrow q\ ,\ V\left[n'\right] \leftarrow v\ ,\ A\left[n'\right] \leftarrow 0$$

In both cases, all non-updated memory locations will have their ages incremented by one.

The products of query vectors and the key vectors in the loss terms correspond to cosine similarities between the query and the positive key, and between the query and the negative key. Since cosine similarity is maximum for equal terms, we maximize the similarity to the positive key and minimize the similarity to the negative one. But if they are far apart by a margin of α, we do not propagate any loss.

While updating, if the returned label value matches the ground truth label, we try and make its key vector closer to the query vector and reset the age to 0. But if the returned label value doesn't match the ground truth label, we find a new memory location and write the query vector into it, its correct label values and reset the age to 0

for that location. We find this memory location based on age. Among those memory locations with the maximum age, we randomly choose one to write.

In the next chapter, we describe the DeepSpot system in detail.

CHAPTER 4
DEEPSPOT DEEP LEARNING INDOOR LOCALIZATION SYSTEM

We propose DeepSpot as a practical deep learning-enabled indoor localization system. We use off-the-shelf components for building the mobile collection bot, pre-existing WiFi infrastructure and deep learning architectures to perform indoor localization.

4.1 Background and Motivation

Indoor robotics and automation are increasing in demand as general availability of robots improves. Rapid growth in electronics, computer science and manufacturing in the 1940s and 50s ensured that mobile robots [66] that were once a staple of science fiction, were slowly starting to become a reality. During this era, several prestigious research universities like MIT, Stanford, Carnegie Mellon etc. started their robotics research projects. Mobile robots caught the imagination of the public, with serious science fiction works of Issac Asimov [67] such as “I, Robot” as well as light-hearted cartoons like “The Jetsons” (which is coincidentally the name of the NVIDIA computing device we are using). This era of rapid development reached its apex in 1970, when Lunokhod 1 drove on the surface of the moon [68].

A major field deployment of an indoor robot was in the aftermath of the Chernobyl Nuclear Power Plant disaster in April 1986 in the former Soviet Union (today Ukraine) [69].

This custom robot was designed based on the Lunokhod lunar rover design. It was designed to investigate areas of the highly radioactive plant where it was deemed too dangerous for humans to venture.

Over the next two decades, improvements in robotics as well as the general fall in prices of hobby electronics equipment, would lead to more mundane use of mobile indoor robots.

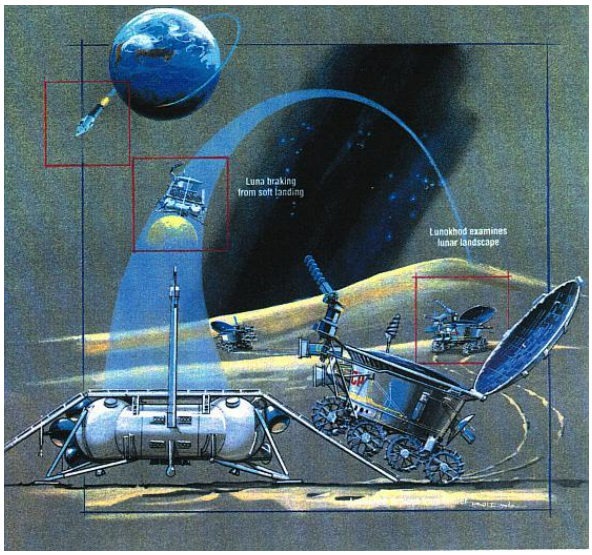

Figure 4-1. Lunokhod Moon Rover. Unknown Soviet artist. *Lunokhod Mission*. 1966-1970. Source: Public Domain image reprinted from Wikimedia Commons, https://commons.wikimedia.org/wiki/File:Lunokhod-mission.jpg, (accessed 25 October 2017).

One of the most popular and enduring indoor robots is the Roomba [70] automatic vacuum cleaning robot, first introduced commercially in 2002. The Roomba features a set of easily available sensors that enable it to roam about the floors and vacuum dirt, while avoiding obstacles. Its two independently operating wheels allow it to complete circular turns in place. The Kobuki robot [71] we use can be said to be a spiritual successor to the Roomba.

In the industrial setting, indoor robots are widely used by logistics firms in their massive warehouses. These robots are usually flat platform mobile robots, capable of moving under a stack of goods, lift them and place them at a desired location. Important players in this industry are Amazon Robotics [72], Grey Orange Robotics [73] etc.

In the coming decades, it is believed that indoor robots will perform more complex tasks such as navigating indoor spaces shared with humans in order to deliver goods. They may even help the differently-abled to achieve seamless mobility. All of this will require reasonably accurate relative positioning within an indoor space.

Additionally, augmented reality technologies are also becoming widespread. Many augmented reality technologies rely on localization to accurately overlay media onto the real world [74, 75].

4.2 DeepSpot

This thesis centers around DeepSpot, a deep learning-enabled scalable and practical system for indoor localization. During the course of this thesis, we study and investigate indoor localization using ubiquitous WiFi infrastructure with scalable fingerprinting using a mobile robot and inference of the location coordinates using memory-enabled deep learning architectures.

Overall, we use a Kobuki mobile robot as a data collection platform and use it to map the hallways 4th and 5th floors of the New Engineering Building, University of Florida, Gainesville, Florida, United States of America. We collect RSS readings, the absolute heading with respect to the Earth's magnetic field and front and rear images using an ultra-wide angle 180 degree view fisheye camera lens. We then use deep autoencoders to compress the collected data modes into feature-rich vectors and then process them through an embedding network equipped with a neural memory to output the estimated location coordinates.

4.2.1 Hardware

1. NVIDIA Jetson TX2 [76]: NVIDIA Jetson TX2 is an embedded computing device, designed for Artificial Intelligence applications. It is powered by dual-core

NVIDIA Denver 2 and quad-core ARM Cortex-A57 and an 8GB LPDDR4, 128-bit interface RAM. It contains an NVIDIA 256-core Pascal Graphics Processing Unit with 8 GB of memory and a memory bandwidth of 59.7 GB/s. It also has a 802.11a/b/g/n/ac 2×2 867Mbps chip for WiFi capabilities. We install Ubuntu 14.04 operating system and control the Jetson remotely using SSH from a personal IBM Thinkpad laptop.

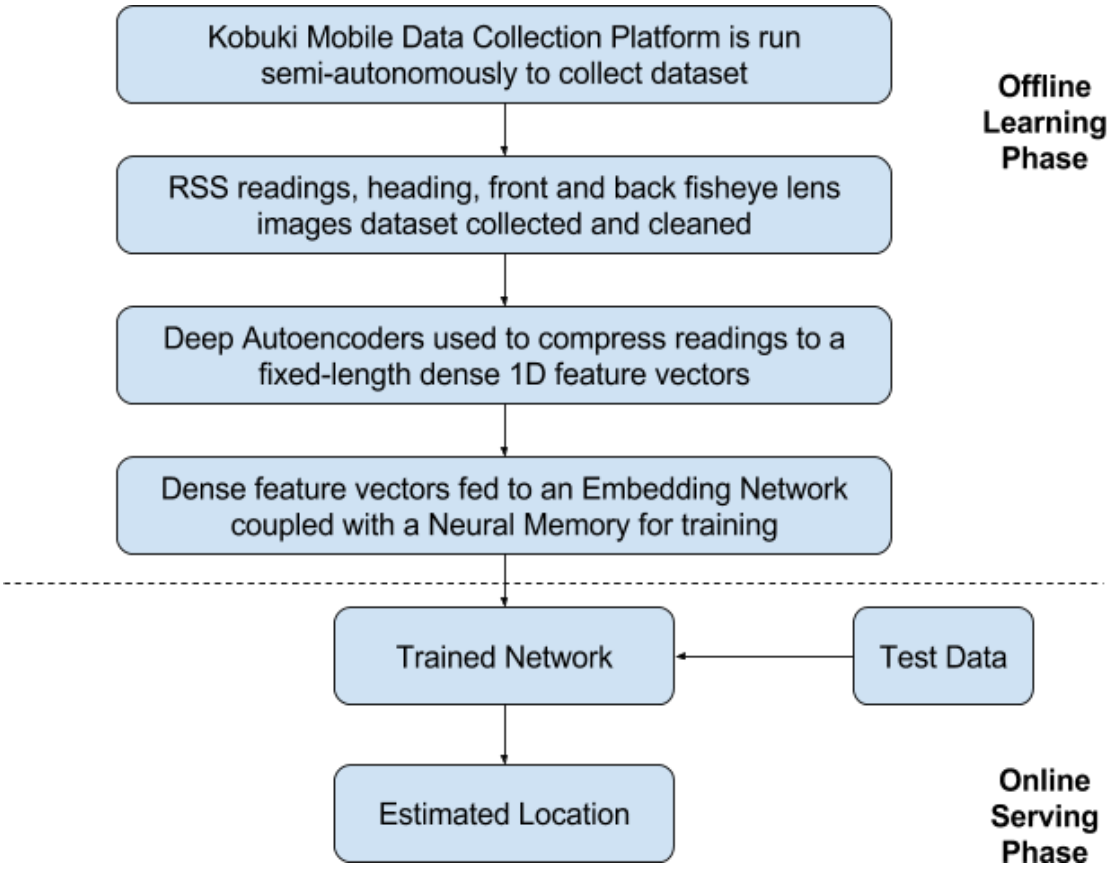

Figure 4-2. DeepSpot process flow.

2. Kobuki Robot [77]: Kobuki is a low-cost mobile research robot aimed at education and research. The Kobuki consists of a Roomba-like base on which several decks of equipment can be stacked. The Kobuki was designed for continuous operation over flat surfaces. Among the many use cases listed, the Kobuki was used as waiter bot in a cafe'. Kobuki can provide power supply to an external computing platform, in our

case the NVIDIA Jetson TX2. The Kobuki can carry a payload of 4 Kgs, which is more than enough for our present equipment.

4.2.2 Assembly

To assemble the platform, we mount the NVIDIA Jetson TX2 onto the lower deck of the Kobuki mobile robotic platform. On the upper deck, we place two, 180° fisheye cameras, held upright using a thermocol plank to achieve a net height of around 4 feet. The magnetometer is also mounted on this deck to avoid interference emanating from the robot assembly. It is also interfaced with the Jetson. We also attach an ice cream stick and use it as a visual aid to calibrate the heading and position on the floor tiles.

4.2.3 Data Collection Process

Since the data we were collecting was in the field and not inside a controlled environment like a laboratory, we had to contend with several real-world challenges and improvise. This is expected as any commercial entity wishing to use such an indoor localization system to map their premises, will also have to face considerations.

Starting with the equipment, the robot was over 2 years old and was not designed for precise locomotion. It had significant drift and errors. The electromechanical hardware didn’t react precisely.

For example, the translational and the rotational speeds often fluctuated. This led to the bot not traversing a set distance properly or being unable to complete full turns properly. For example, we programmed the robot to first take a full circular turn, then move straight 3 feet repeat. This was repeated 4 times in succession. There was a net approximate drift of -12 inches (0.3048 m) along the straight path, as well as a sideways drift of 8 inches towards the left. Thus, we couldn’t rely on dead-reckoning to build an ego-centric coordinate system.

We relied on the floor pattern of the hallways which was neatly arranged in a grid of 1 feet by 1 feet tiles. There was a great deal of slippage on carpeted portions (around 25% of the hallway floor space), so we stayed only on the tiled areas where traction was good. We conducted the mapping late at night to prevent people working in the building from being inconvenienced. This issue would be of practical concern to the commercial entity operating a similar system. It often may not be viable to cordon off large portions for extended periods of time.

The Kobuki's batteries were old and discharged well before we could finish mapping a floor. Thus, the mapping for both the floors took 6 nights to complete. This is of practical concern to commercial entities as they too may not be able to complete the mapping in one sitting. This leads to a noisy dataset as some access points may be available on one day but perhaps not on another day.

Such mapping cannot be a one-time exercise. It will have to be redone periodically, say every 2 months, as changes to the WiFi infrastructure, minor and major changes to the building layout, positioning and removal of equipment etc. can cause the older data collected, to become outdated.

While most other similar datasets were collected by humans, using economical off-the-shelf robots to fingerprint the premises not only saves time but also reduces the cost of hiring people to take a large number of readings, in the long term.

We started with 4th Floor NEB. We restricted ourselves to the non-carpeted areas while performing the fingerprinting. 5th Floor NEB layout is quite similar as well.

To prepare for the data collection, we paste tiny removable stickers every 3 feet to mark waypoints along the path. This way we could visually inspect the robot and

manually correct for any deviations from the path. It also enabled us to periodically check if our readings matched the location coordinates as a single mismatch could possibly mislabel all further readings while saving the data. The stickers are imperceptible in images captured.

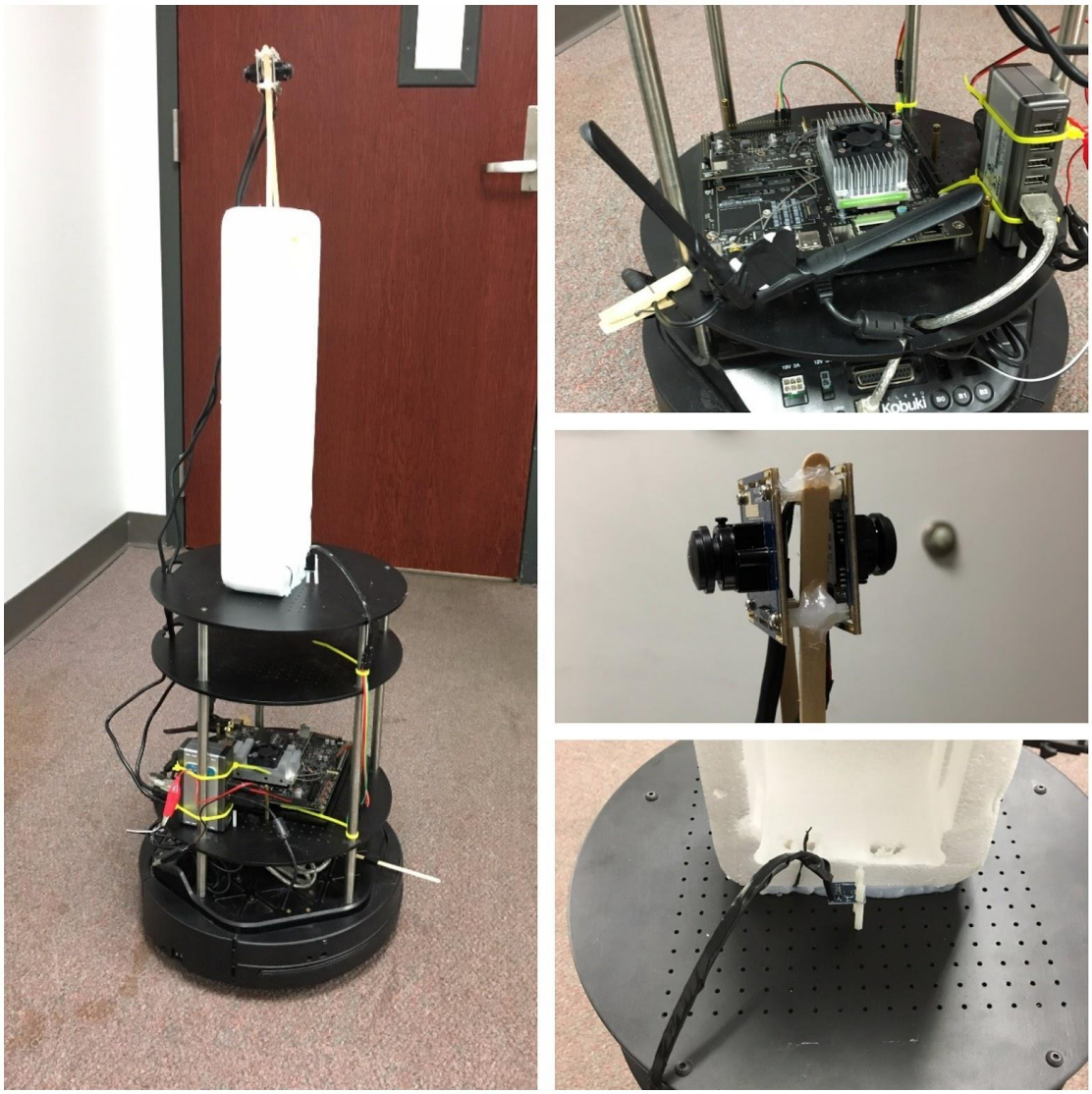

Figure 4-3. Left, Kobuki robot platform with NVIDIA Jetson TX2. Right (from top), Jetson TX2 with fixed antennas, USB cameras, magnetometer mounting location. Photo courtesy of author.

We then start the data collection. The Kobuki platform would first rotate two full circles on its axis and collect readings with different headings. It would collect 30

readings in about 50 seconds. While it was possible to speed up the rotational speed, it led to jerks in the rotation and affected the end heading (which should be 0º after taking a full circle). Too slow and it would delay our collection process. Covering a full circle in 25 seconds seemed appropriate.

Figure 4-4. NEB 4th Floor.

Figure 4-5. Marking tiles with stickers. Photo courtesy of author.

Using this we can calculate our rotational speed. We also calculate the data collection frequency.

$$Rotational\ speed\ =\ 2\pi\ radians\ /\ 25s\ =\ 0.251\ radians/s$$

$$Data\ collection\ frequency\ =30\ /\ 50s\ =\ 0.6\ Hz$$

We experimented with turning the robot clockwise in the first rotation and anti-clockwise in the second rotation, when taking readings at a particular location. The idea behind this was to cancel errors when the robot wanted to achieve a certain heading. However, this caused strong jerks when the robot tried to reverse its direction of rotation. The whole robot would shake and displace its wheels. This led to enormous drift over time.

So, we decided that the robot would turn only clockwise at a particular location and only anticlockwise at the successive one, thus alternating between the two.

After completing two rotations at a particular location, the Kobuki moves 3 feet at a translational speed of 50 cm per minute. While the Kobuki is capable of moving much faster (with an advertised maximum translational speed of 70 cm per second), the additional vertical loading destabilizes the robot and leads to undesirable displacements of the wheels. It also drains the battery much faster.

For practical reasons we only let the robot cover 12 feet in an automated manner, due to the drift incurred. Every 12 feet we manually calibrated the position and heading of the robot, using the stickers and ice cream stick. We also paused and re-did those sections where people (such as students, faculty, janitorial staff etc.) had to traverse. Ideally, it would have been possible to pre-program the entire path, but practical considerations didn’t allow it.

While the dataset collection was done using the Kobuki robot, we would also like to mention that we also considered using a Tacon Remote controlled car. We removed the original body and mounted it with a 3D printed bed to hold a digital camera. We also remotely drove it around on the floors of the New Engineering Building to capture a

dataset. We trained a Convolutional Neural Network, end-to-end, along the lines of the NVDIA Self-driving car paper [79].

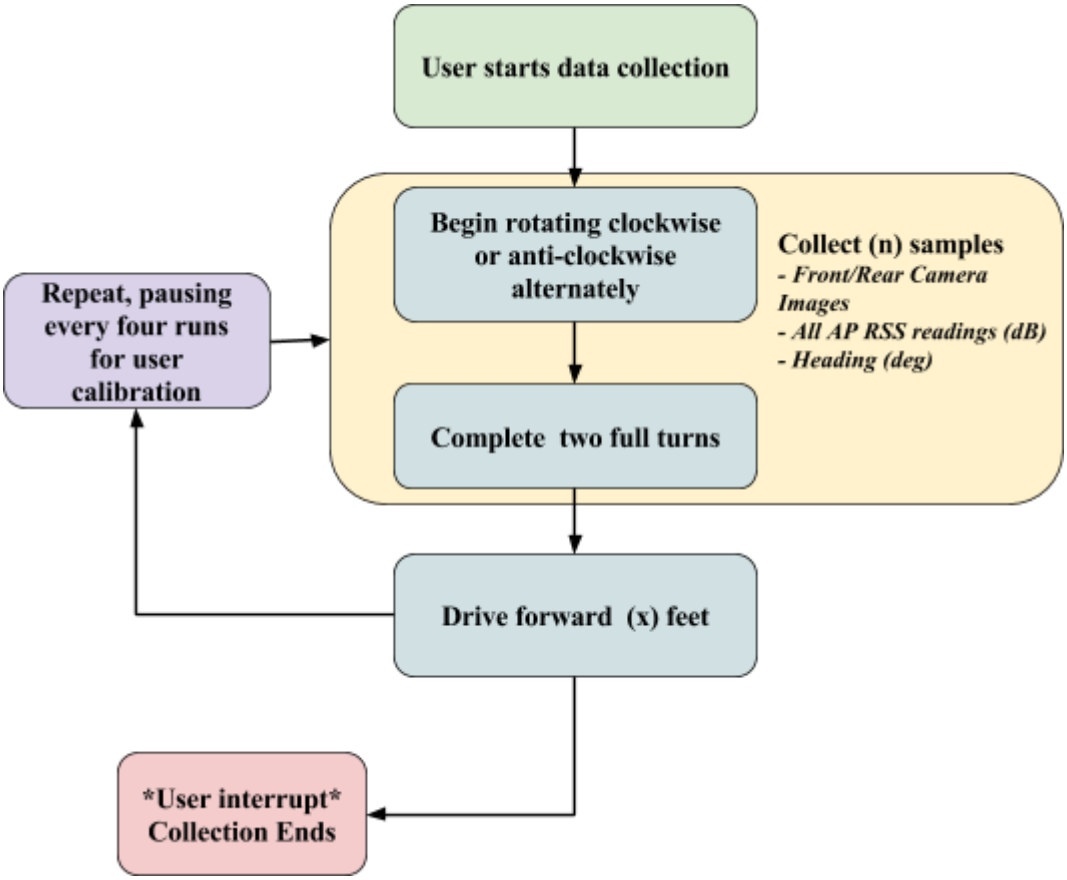

Figure 4-6. Data collection procedure with, t = 50 seconds, n = 30 samples, and x = 3 feet.

In this manner, we collect the raw data i.e. RSS readings, heading and scene images.

Figure 4-7. Example of a scene image collected. Photo courtesy of author.

While we were successful in our attempt to enable the car to drive itself along predefined “lanes”, we found that this testbed had many shortcomings that made us opt for the Kobuki instead.

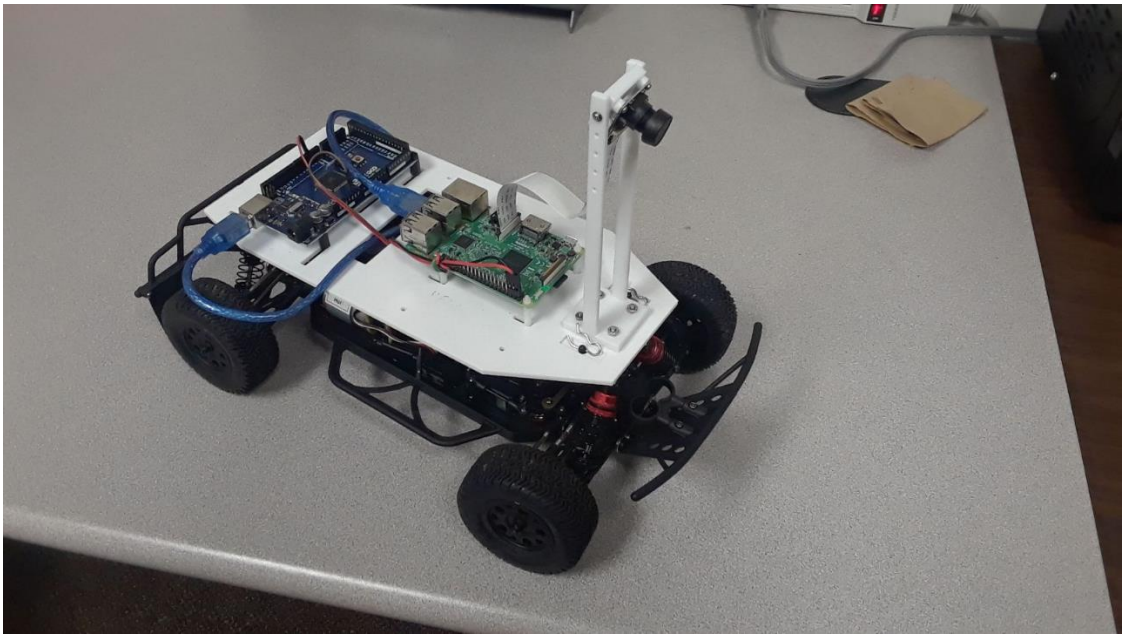

Figure 4-8. Modified Tacon RC car testbed. Photo courtesy of author.

We noticed that the direction in which bot was facing had an impact on the received access point RSS values. It would have been very cumbersome to make the car take multiple readings in different directions for every location coordinate. The Kobuki though not as fast, had the ability to rotate along its axis and make complete circular turns while standing at one position. Another issue with the car was that its camera was mounted too low and that increasing the height of the camera made the testbed very unstable. The Kobuki provided a wide and heavy enough platform for us to mount the cameras at a net height of approx. 4 feet from the floor. This ensured the images captured were at a height not very dissimilar from that of a human holding a smartphone camera at chest height.

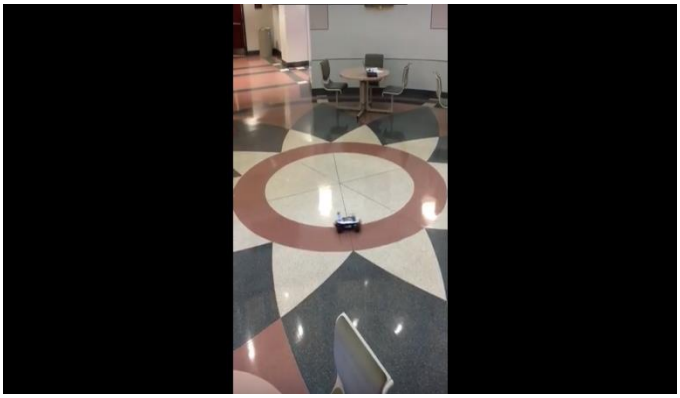

Figure 4-9. Tacon RC car trained with deep learning following the circular lane. Photo courtesy of author.

CHAPTER 5
DATASET ANALYSIS

RSS, compass headings, and front and back scene images are collected by mobile Kobuki platform equipped with NVIDIA Jetson TX2 [76]. The RSS of all available unique access points are stored and are measured in decibels (dB) by default. The compass heading is measured in degrees. The compass is calibrated to magnetic north by compensating the magnetic declination of the location (Gainesville, Florida, United States of America). Two images are captured per sample location. Forward and rearward facing cameras with 180-degree fisheye lenses are mounted back-to-back to capture a broad view of the environment. The images captured by the cameras are stored by default as 641 by 481 pixel JPEG images. The location (in a relative, predetermined x, y coordinate system), heading, and all Wi-Fi access points are stored in a JSON formatted file.

A reading stored in the JSON file consists of the following information:

- A dictionary containing WiFi access points IDs as keys, paired with their RSS values in decibels (dBm)
- Heading in degrees with respect to the north magnetic pole
- Relative folder and file path of the image from the front camera with timestamp
- Relative folder and file path of the image from the rear camera with timestamp
- Relative X coordinate
- Relative Y coordinate

Two such JSON files were captured, one for each floor i.e. 4th floor and 5th floor. Both of these were combined to form the complete dataset. A new Z coordinate was added as 4 or 5, indicating the respective floor.

```
"access_points": {
    "13640875216176": -52,
    "13640875216177": -52,
    "13640875216179": -52,
    "13640875216188": -52,
    "13640875216189": -51,
    "13640875216190": -51,
    "13640875216191": -52,
    "13640875216224": -67,
    "13640875216225": -66,
    "13640875216226": -67,
    "13640875216227": -67,
    "13640875216233": -79,
    "13640875216236": -79,
    "13640875216237": -79,
    "13640875216238": -79,
    "13640875216239": -80,
    "13640878125025": -73,
    "13640878125026": -73,
    "13640878125027": -73,
    "13640878125033": -67,
    "13640878125036": -66,
    "13640878125037": -66,
    "13640878125038": -67,
    "13640878125039": -67,
    "157605898618": -59,
    "260453955565155": -75,
    "260453955565158": -75,
    "260453955570576": -70,
    "260453955570577": -70,
    "260453955570578": -70,
    "922274856689": -83,
    "92676818070753": -58,
    "92676818070755": -58,
    "92676818070762": -60,
    "92676818070764": -60,
    "92676818070765": -59,
    "92676818070766": -59,
    "92676818070767": -60
},
"heading": 113.35327715672835,
"img1": "../datasets/images/2017-07-31_19_03_04.914978_cam1.jpg",
"img2": "../datasets/images/2017-07-31_19_03_04.914978_cam2.jpg",
"x": 169,
"y": 3
```


Figure 5-1. A sample data point in JSON format.

For generality, the make, model, and location of the access points in the building are unknown and will not be considered in our analysis. This also reduces the burden for the proprietor of the indoor space.

After collecting the data, we clean the dataset. We remove readings that were not taken properly and thus resulted in an NA. Overall there were less than 10 such readings that were malformed and thus removed. This led to a complete dataset of

6593 individual readings. Together they encompass 205 unique locations and track 382 unique WiFi access points. If a WiFi router was not detected because its RSS was too low, we set it to -110 dBm.

For 205 locations, 15 of them have more than 30 readings (due to path overlap) and 3 locations have between 15 and 29 readings and only 1 has 8 readings. The remaining 186 locations have exactly 30 readings each.

We now visualize and see the following characteristics of access point RSS values.

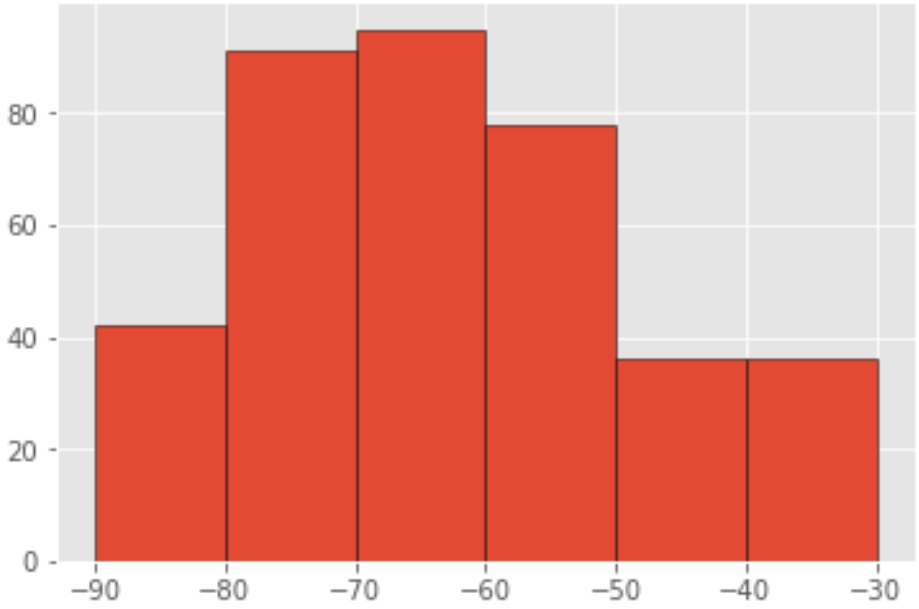

Figure 5-2. Distribution of Maximum RSS values.

We see that most WiFi access points in the dataset have maximum RSS between -70 dBm and -60 dBm.

We see that most WiFi access points in the dataset have mean RSS value between -90 dBm and -80 dBm, assuming they were detected (i.e. had an RSS value of greater than -110 dBm).

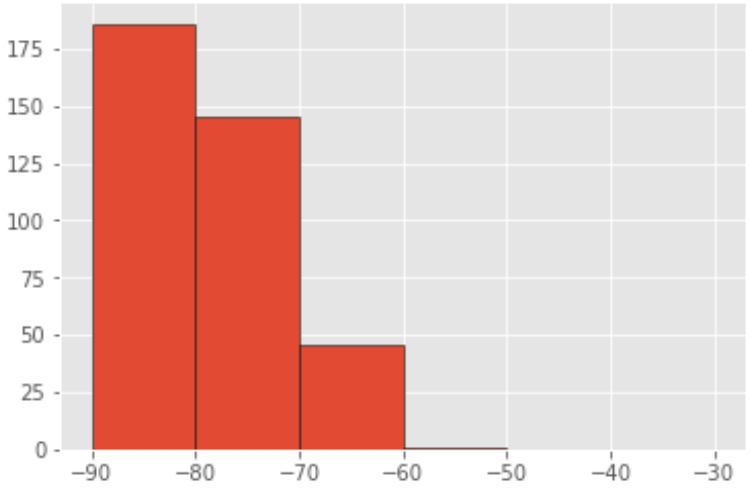

Figure 5-3. Distribution of Mean RSS values.

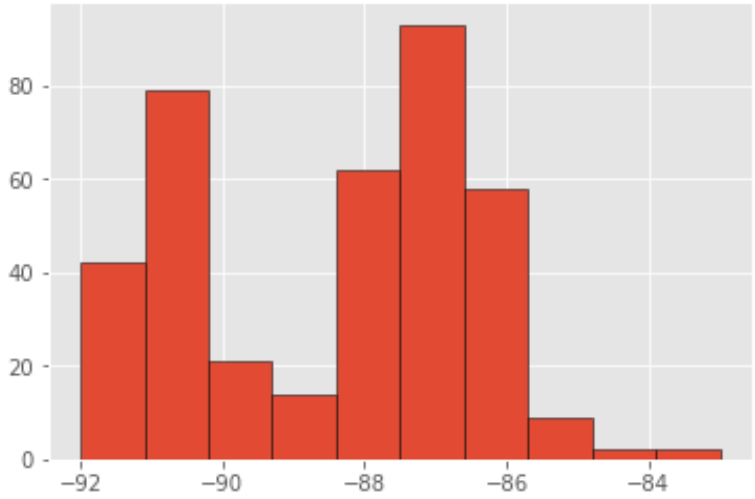

Figure 5-4. Distribution of Minimum RSS values.

We see that the lowest RSS reading for all WiFi access points is -92 dBm, assuming they were detected (i.e. had an RSS value of greater than -110 dBm). This is likely due to the hardware limitations for detecting very low RSS signals.

While we don't know the locations of the access points, we can expect that the access points will radiate WiFi signals outwards, with RSS decreasing with distance, as well as the opacity of the path.

To illustrate this, we visualize the heatmap of the RSS readings for a particular access point, with ID “92676818070767”.

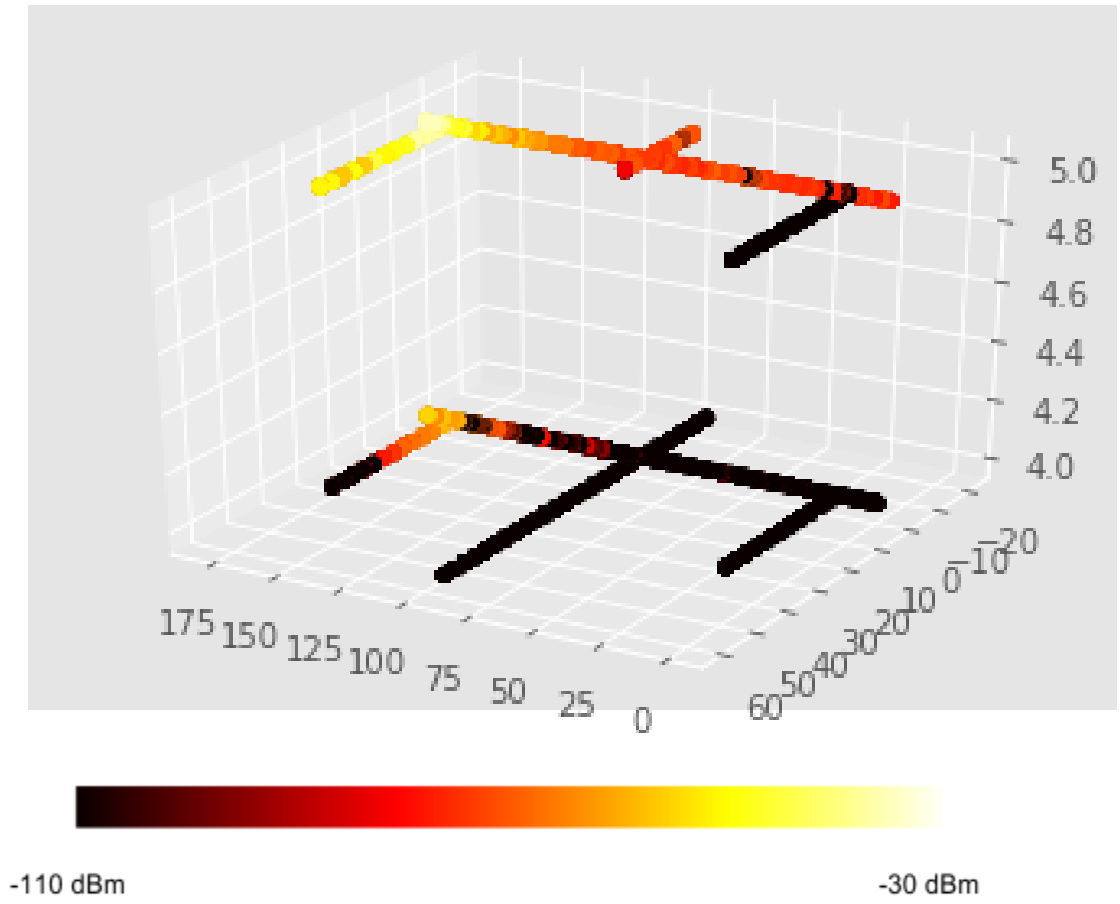

Figure 5-5. RSS heatmap for WiFI access point ID 92676818070767.

This 4D plot shows the X, Y, Z(floor) coordinates of all readings of RSS from this particular access point. While we don’t know its location for sure, we can give a reasonable guess that it is on the 5th floor, towards the upper left corner, near (175,0).

We can see how the RSS gradually decreases with distance down the straight immediate hallways on the 5th floor. But on the perpendicular hallway on the upper right, on the 5th floor itself, we see that access point “92676818070767” is not detectable. This is likely due to the severe attenuation faced when traversing the

laboratories and other rooms across the building. We also notice that directly below, at the same coordinates (175,0) but on the 4th floor, we see significant signal strength. However, for most of the 4th floor access point “92676818070767” is not detectable.

Another relationship we found was the variation of RSS with heading. Over the course of a complete rotation at a single location coordinate, access point signal strength can vary significantly.

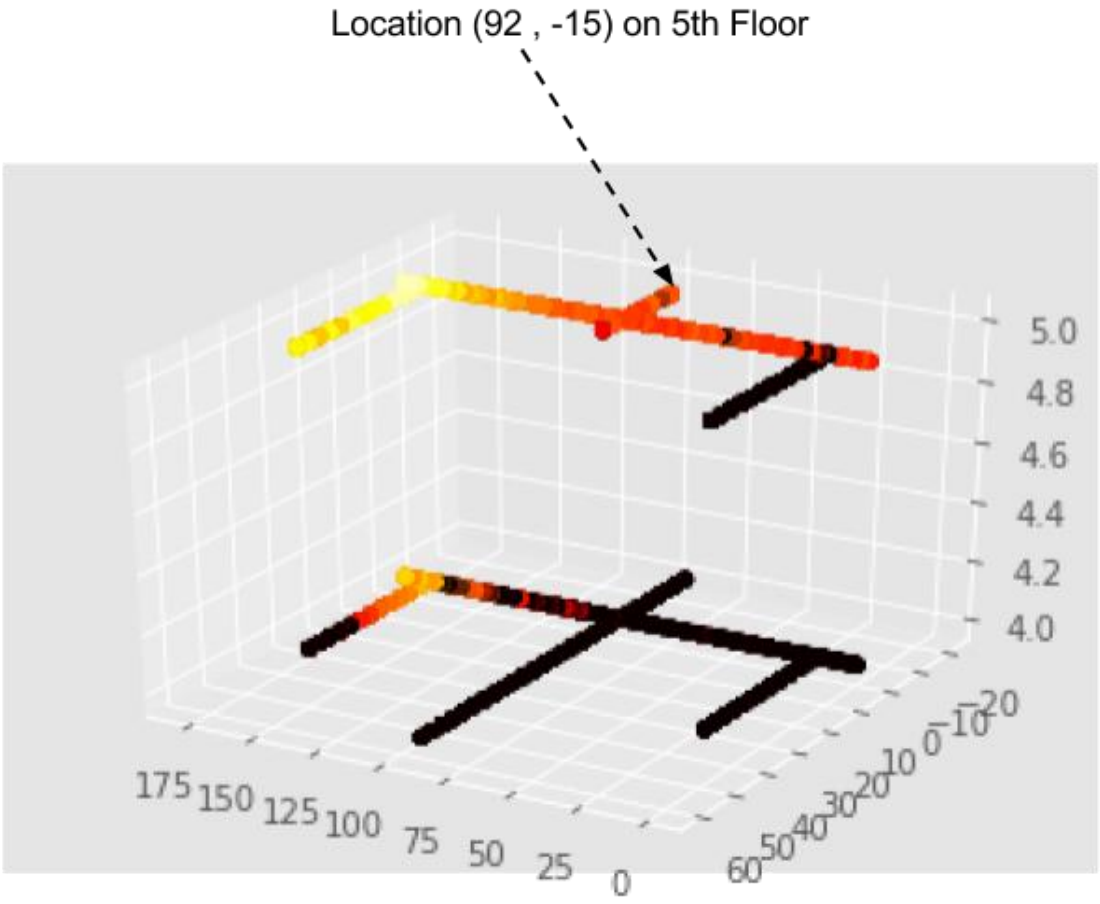

Figure 5-6. RSS heatmap for WiFI access point ID 92676818070767.

We take a point (92, -15) on the 5th floor and plot the RSS readings from access point “92676818070767” (discussed above) with respect to their headings (after scaling to have 0 mean and standard deviation of 1). When we plot a histogram, we see that

most of the readings are clustered between -85 dBm and -80 dBm, except for 2 readings which are -110 dBm (not detected).

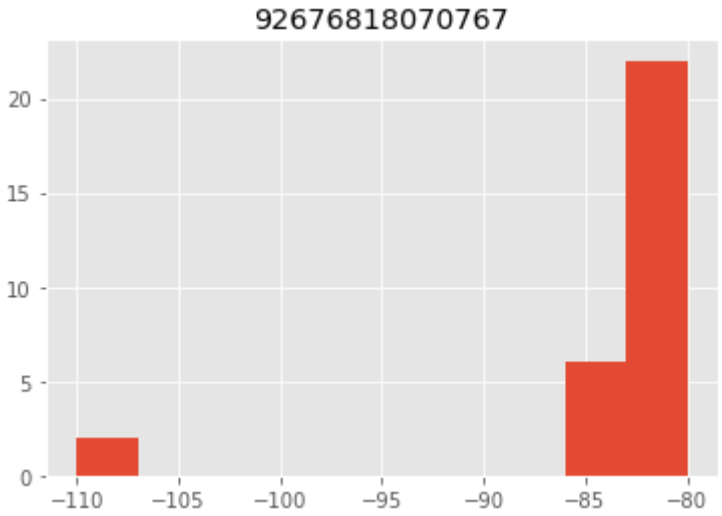

Figure 5-7. Histogram of RSS readings at different headings for WiFI access point ID 92676818070767.

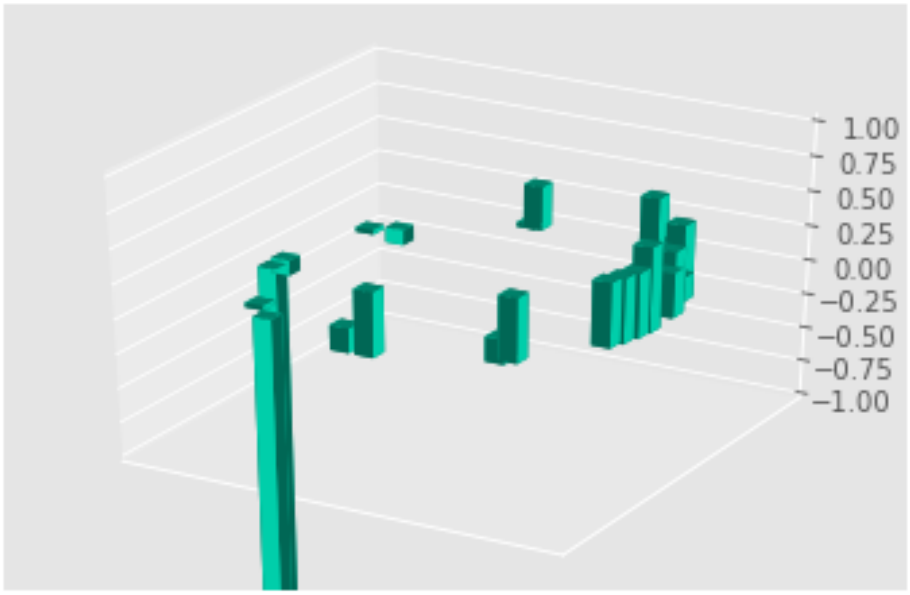

Figure 5-8. Normalized RSS readings at different headings for WiFI access point ID 92676818070767.

We investigate the scene images at that location and at the heading which received the lowest RSS reading.

Figure 5-9. Front camera image at (92, -15) on the 5th floor at weakest signal strength. Photo courtesy of author.

We see that the lowest readings (-110 dBm i.e. undetected) occur when facing towards the doorway, whereas the highest reading (-80 dBm) occurs when roughly facing opposite to the doorway.

Figure 5-10. Front camera image at (92, -15) on the 5th floor at strongest signal strength. Photo courtesy of author.

In the next chapter, we present the experimental results obtained by deep architectures.

CHAPTER 6
EXPERIMENTAL RESULTS AND DISCUSSION

We use the data collected and convert it to three broad datasets:

- Dataset consisting of all locations only on the 4th floor: DS_4th
- Dataset consisting of all locations only on the 5th floor: DS_5th
- Dataset consisting of all locations on the 4th and 5th Floors: DS_Full

We partition the dataset into training, validation and test sets in the ratio of 72 : 8 : 20. This can be thought of as first dividing the entire dataset into 80 : 20 ratio for train and test sets, and then subdividing the train set into a smaller train set and validation set in the ratio of 90 : 10. We also ensure that all points have 30 readings. This is to maintain balanced classes. We use stratified sampling to ensure that readings from every location ID are present in all the three sets.

Table 6-1. Dataset breakup

Dataset	Training Data	Validation Data	Test Data
DS_4th	2311	257	642
DS_5th	1706	190	474
DS_Full	4017	447	1116

The dataset consists of three modes of sensors i.e. WiFi, magnetometer heading and two scenes images from forward-facing and rear-facing cameras. We preprocess them in the following manner.

For the RSS readings of various WiFi access points, we scale them to the range of 0 and 1. This gives us a vector of RSS readings. For the heading angle, we take the sine and cosine of the heading. This converts a real-valued heading angle numerically between 0 and 360, into a heading direction on a unit circle in a 2D plane. We do not

further scale them. For the images, we first convert them to grayscale images. Since the images were captured by a fisheye lens, they are highly distorted and not of high-resolution. Text in the scenes is not visible, along with other finer objects. So we apply Canny edge detection to map the edges in the images. Floors, walls, doorways etc. get highlighted. For every reading, we then append the rear-facing camera's image to the right of the front-facing camera's image. This gives a rough view of the entire scene at the time and place of that reading. We rescale the two concatenated images to a final size of 128 by 128 pixels. Finally, we scale the pixel values by dividing them by 255. This ensures that individual pixel values are between 0 and 1. We would like to note that we were unable to undistort the image reasonably well. Several artifacts were introduced when undistorting rendering the net result unusable.

The output which we wish to predict consists of location coordinates (x, y) along with the floor (4 or 5). While working on datasets that pertain to only one floor (DS_4th and DS_5th), we do not predict the floor.

We broadly divide the experiments into three sections:

- Regression: We treat the location coordinates as real values and try and predict them as close as possible. Each coordinate (x, y or floor) will be predicted by one neuron each.

- Classification: We treat the location coordinates as categories and try and classify them correctly. The final output layer will have those many neurons as unique location points in that particular dataset.

- Label prediction using neural memory: We use a neural memory to output the correct coordinate label, using cosine similarity. The output label will be a location ID, an integer which maps to a particular location. There is no output layer in this case.

We run experiment with different random seeds and report average estimates of the mean error in predicting the coordinates i.e. the Localization Error. The results obtained are noisy and vary with a deviation ~0.2 meters.

$$Mean\ Error = \frac{1}{n} \sum_{i=1}^{n} \sqrt{(\hat{x}_i - x_i)^2 + (\hat{y}_i - y_i)^2}$$

where $\hat{x}$, $\hat{y}$ *are the predictions and* x_i, y_i *are the ground truth coordinates*

6.1 Regression

We begin by using a 3-layer simple feedforward network to process the RSS input vector. We also implement a 3 CONV-layer VGG-based Convolutional Neural Network [80, 81] in parallel to learn from the images. We then concatenate the feature vectors coming from both these streams and the heading vector. After the concatenation, the core network of 2 layers further processes the unit. We vary this core portion with 64-64 and 128-128 neurons in the two layers. Then the network branches into three output networks with separate output heads for each coordinate for a location (i.e. separate branches for predicting each of x, y and floor coordinates). The final output neurons have linear activation. The output is then rescaled to get the predicted result in terms of actual coordinates (which are in feet). This result is compared to the

ground truth x and y coordinates (and floor coordinates, if required). The mean absolute errors are then calculated for all test points.

We use ReLU nonlinearity as well as dropout [45] with a probability of 30% in the layers. We initialize the layers using He Normal Initializer [45]. We run each model for 100 epochs and use the validation test set loss to select the best model.

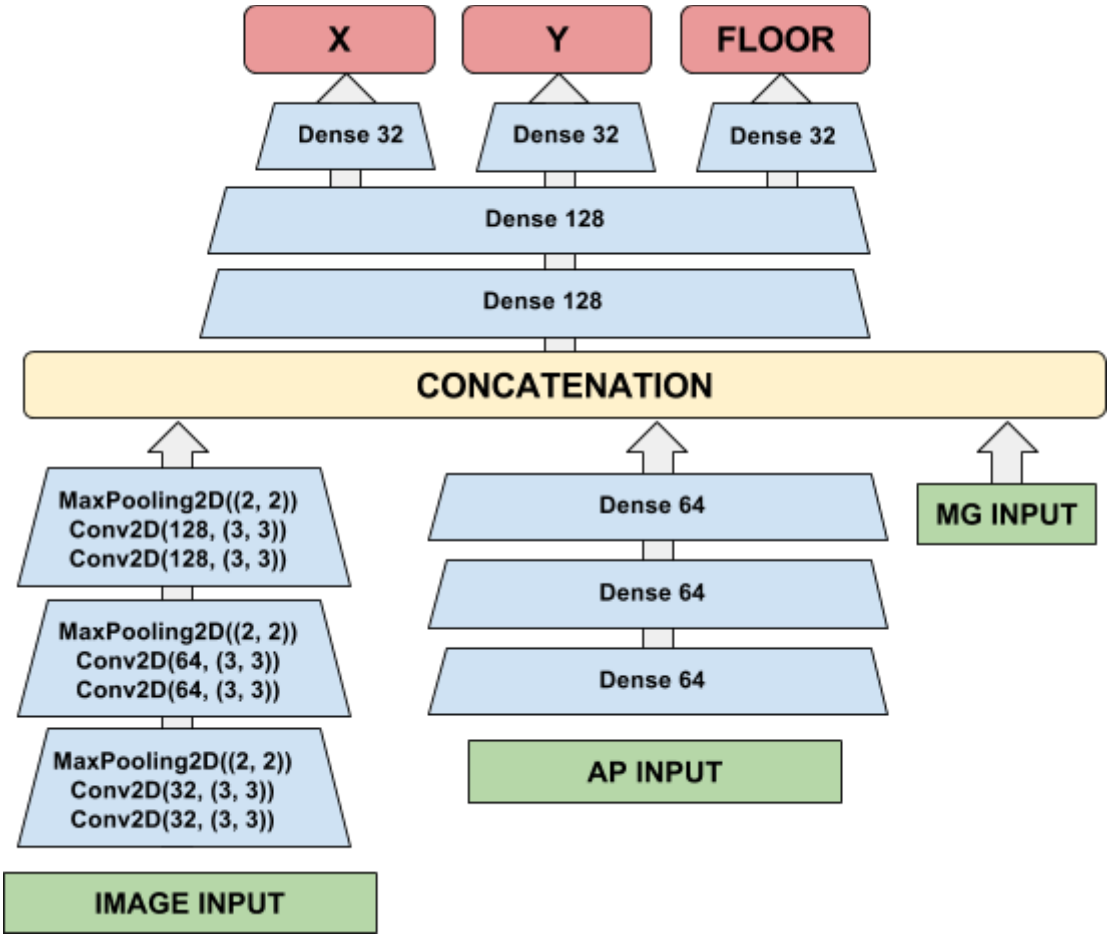

Figure 6-1. Deep Architecture for regression.

The loss function used for the regression experiments is Mean Squared Error [45], which is a popular choice when performing regression. We see that the network with the larger core layers, having greater representational power, gives lower error. We also try regression using auto-encoded features. We use stacked autoencoders [83] for

100 epochs to self-encode and compress the RSS input and the image inputs separately.

Table 6-2. Estimated results for Regression without SAE

Network	Dataset	Localization Error (meters)
Reg_Core_64-64	DS_4th	1.83
Reg_Core_64-64	DS_5th	1.53
Reg_Core_64-64	DS_Full	1.71
Reg_Core_128-128	DS_4th	1.61
Reg_Core_128-128	DS_5th	1.47
Reg_Core_128-128	DS_Full	1.55

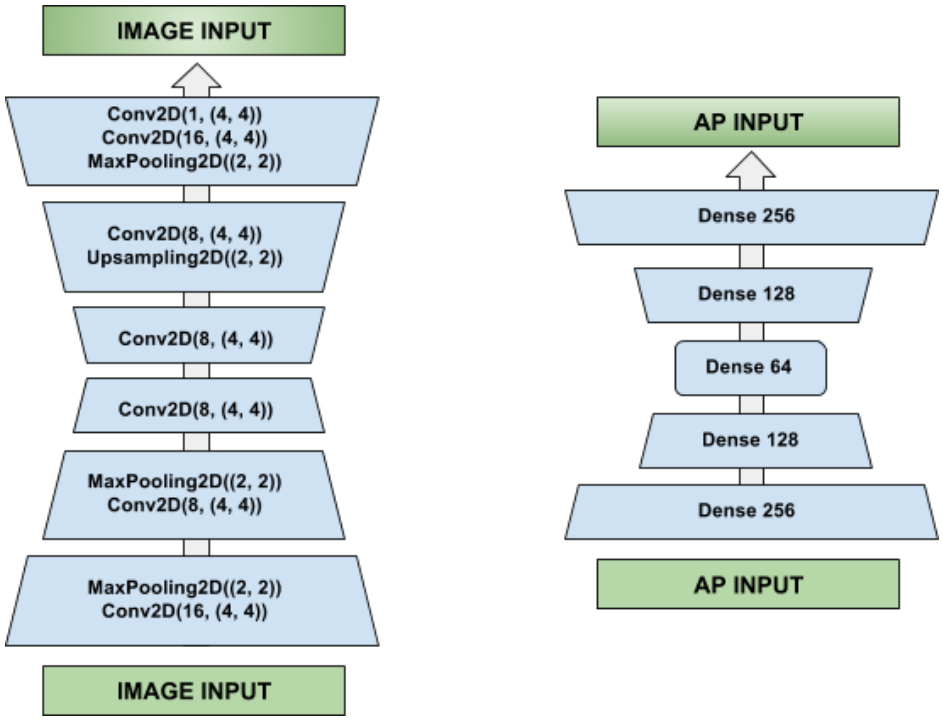

Figure 6-2. Convolutional Autoencoder for feature concentration.

A regular feedforward network-based autoencoder is used to compress the RSS vectors, whereas a convolutional autoencoder [84] is used to compress the images. We remove the latter half i.e. the decoder and keep on the encoder portions of the two autoencoders. Both yield compressed vectors of length 64 for their respective inputs. We then concatenate the compressed RSS vector (length 64), the heading vector (length 2) and compressed image vector (length 64), to yield a final vector of length 130 (i.e. 64 + 2 + 64). Thus, for each reading, we have compressed the RSS values, the heading and the front and rear images into a 130 length feature vector. This vector now becomes the input vector to the final regressor, which is a feedforward network with 2 hidden layers core layers and like earlier, branches for predicting the coordinates. As before, we use ReLU nonlinearity and dropout with a probability of 30%.

Table 6-3. Estimated results for Regression with SAE

Network	Dataset	Localization Error (meters)
SAE_Reg_Core_64-64	DS_4th	1.44
SAE_Reg_Core_64-64	DS_5th	1.28
SAE_Reg_Core_64-64	DS_Full	1.38
SAE_Reg_Core_128-128	DS_4th	1.30
SAE_Reg_Core_128-128	DS_5th	1.16
SAE_Reg_Core_128-128	DS_Full	1.25

We notice an improvement in the result when using the stacked autoencoders. This is likely due to the concentration of useful features by the encoder. As with

regression, we use the same network to get the feature concatenation layer. However, instead of branching into separate x, y and floor heads, we put two hidden layers and a final layer for classification. The final layer consists of output neurons with softmax activation, to output a probability distribution over all label classes. We output that label class as the predicted result. The network is trained using multi-class cross entropy loss [45].

6.2 Classification

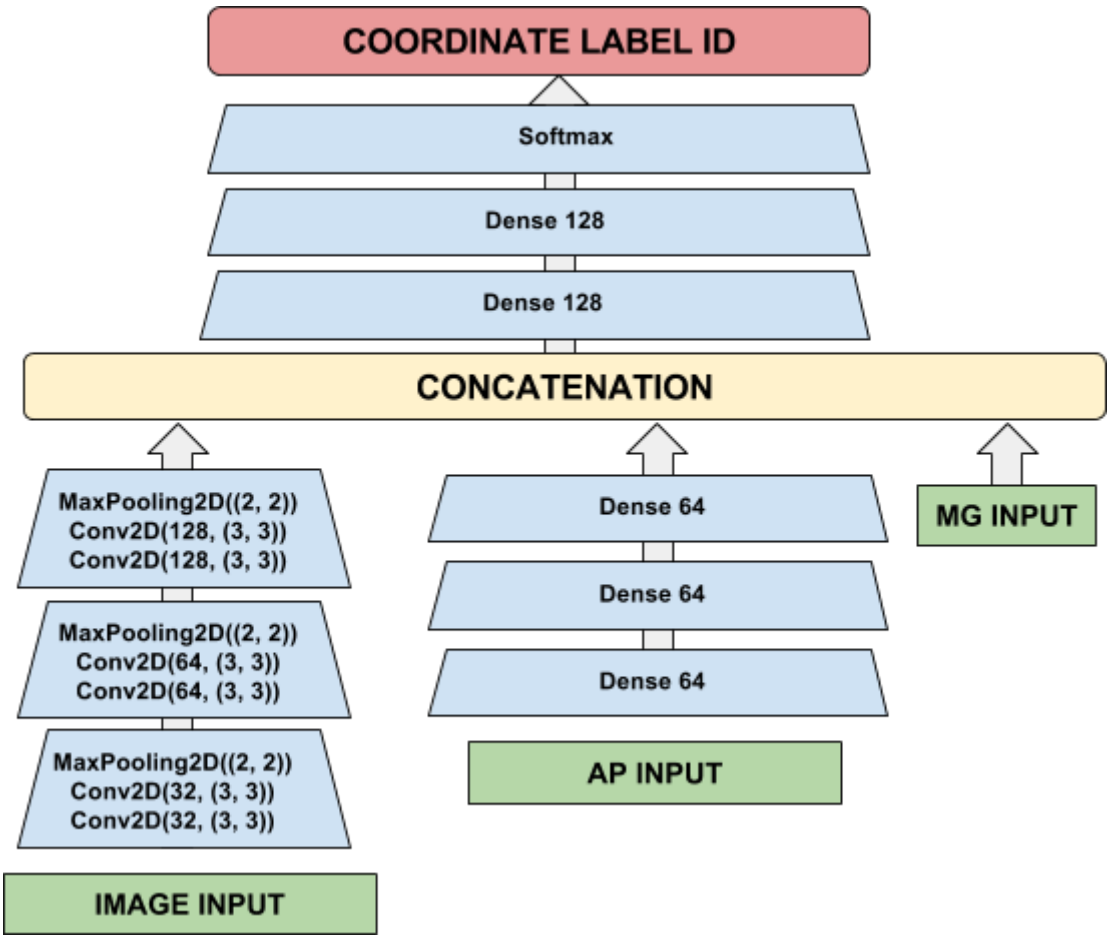

Figure 6-3. Deep Architecture for classification

However, this predicted result is simply a location ID. We use a dictionary lookup to find the individual x and y coordinates (and floor coordinates, if required). Now we

have the coordinates of the location. These coordinates were obtained during the data collection process but are not the ground truth coordinates. They are the coordinates corresponding to the predicted ID.

These are then compared to the ground truth x and y coordinates (and floor coordinates, if required). The mean errors are then calculated for all test points.

Table 6-4. Estimated results for Classification without SAE

Network	Dataset	Localization Error (meters)
Cls_Core_64-64	DS_4th	1.51
Cls_Core_64-64	DS_5th	1.35
Cls_Core_64-64	DS_Full	1.37
Cls_Core_128-128	DS_4th	1.42
Cls_Core_128-128	DS_5th	1.29
Cls_Core_128-128	DS_Full	1.41

We see that classification gives an overall better result. This may be due to the network, which could be predicting approximate values, would be forced to map it to the closest correct answer. Thus, the network need not guess the exact location coordinates, just close enough to map it to the closest location ID.

We also use stacked autoencoders to self-encode and compress the RSS input and the image inputs separately and append the heading vector, just as we did for regression. As before, the final classifier is a feedforward network with 2 hidden layers

(64-64 and 128-128) and a final softmax layer. As before, we use ReLU nonlinearity (except in the final output layer) and dropout with a probability of 30%.

We notice an improvement in the result. As before, this is likely due to the concentration of useful features by the encoder. Thus, we can see that using the deep autencoded representation of the inputs helps get better results.

Table 6-5. Estimated results for Classification with SAE

Network	Dataset	Localization Error (meters)
SAE_Cls_Core_64-64	DS_4th	1.36
SAE_Cls_Core_64-64	DS_5th	1.21
SAE_Cls_Core_64-64	DS_Full	1.27
SAE_Cls_Core_128-128	DS_4th	1.23
SAE_Cls_Core_128-128	DS_5th	1.14
SAE_Cls_Core_128-128	DS_Full	1.23

6.3 Classification with the Neural Memory

As with regression and classification, we start by using stacked autoencoders to self-encode and compress the RSS input and the image inputs separately and append the heading vector. We get a final vector of length 130 (i.e. 64 + 2 + 64) for each reading. We then use an embedding feedforward network with 2 hidden layers. The output of the embedding network has a length of 128 and is then sent to the neural memory. The neural memory is initialized with a query size (and thus key size) of 128, in order to receive the query from the embedding network. We initialize the memory to

have 4096 slots. The nearest neighbors list size was kept at 2048 i.e. during every similarity match, a list of 2048 nearest keys would be called up. We would like to mention that these neural memory hyperparameters were obtained after extensive grid search.

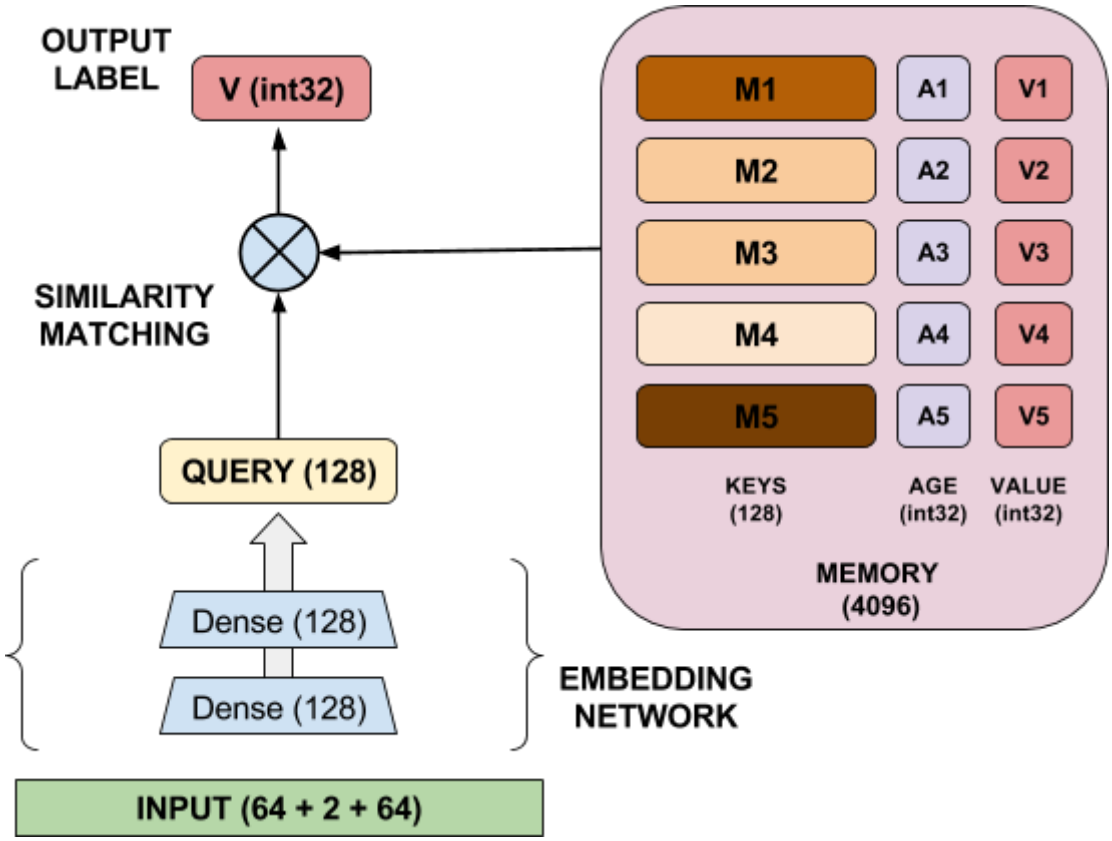

Figure 6-4. Embedding network with neural memory.

We compare the performance of the above embedding network with and without the memory. We notice that the error results are similar to the earlier experiments. On the whole, the performance is similar that observed by state-of-the-art indoor localization systems using WiFi characteristics (such as RSS i.e. Horus, and CSI i.e. FIFS and PhaseFi) [84]. It should be noted that those results are for a dataset which was collected in a living room, with one WiFi Access Point. 50 locations in a grid pattern in a 4 meter by 7 meter space were used to create their dataset.

Our dataset and approach are primarily meant to be used outside in hallways and cover a much larger space and also a large number of WiFi access points.

Table 6-6. Estimated results for Classification with Neural Memory

Network	Dataset	Localization Error (meters)
SAE_Cls_128-128	DS_4th	1.33
SAE_Cls_128-128	DS_5th	1.17
SAE_Cls_128-128	DS_Full	1.22
SAE_Cls_128-128_Mem	DS_4th	1.36
SAE_Cls_128-128_Mem	DS_5th	1.19
SAE_Cls_128-128_Mem	DS_Full	1.27

Weighted k-Nearest Neighbors (with RSS and 4 headings) by Jekabsons et. al. [86] uses a dataset is similar to ours with 82 training points and 68 test points, across two floors, manually collected in a university building. One of the key contributions of neural memories is to be able to remember rare and infrequent exemplars. We thus perform an experiment by unbalancing the data on purpose. We select those location IDs which have 30 readings. Of those location IDs, we choose 70 percent of those IDs and randomly remove 24 of 30 readings. Thus, we now have a dataset in which 70% of the location IDs have only 6 readings associated and the remaining 30% have all the 30 readings. Now we compare the same embedding network with and without the neural memory. We notice that the accuracy is higher when the memory is used. We also change the size of the embedding network by reducing the number of layers from 2 to 1,

and test again. Here too we notice that when using the memory, we do not suffer much degradation as performance.

Table 6-7. Comparison with other similar methods

Algorithm	Localization Error (meters)
PhaseFi	1.08
FIFS	1.24
Horus	1.54
Maximum Likelihood	2.16
Weighted k-Nearest Neighbors	1.82 to 2.44
DeepSpot with auto-encoded features, 128-128 Embedder and Neural Memory	1.19 to 1.36

Table 6-8. Estimated results for Classification with Neural Memory with unbalanced dataset

Network	Dataset	Localization Error (meters)
SAE_Cls_128-128	DS_4th	2.10
SAE_Cls_128-128	DS_5th	1.78
SAE_Cls_128-128	DS_Full	1.99
SAE_Cls_128-128_Mem	DS_4th	1.86
SAE_Cls_128-128_Mem	DS_5th	1.52
SAE_Cls_128-128_Mem	DS_Full	1.71

Thus, it would appear that for a given embedding network, the addition of a neural memory helps in the cases when the data is unbalanced. This is a likely scenario

in when deployed in the field, as different sections of the indoor premises may be mapped differently and at different times.

Table 6-9. Estimated results for Classification with Neural Memory with unbalanced dataset with smaller embedder

Network	Dataset	Localization Error (meters)
SAE_Cls_128	DS_4th	2.76
SAE_Cls_128	DS_5th	2.31
SAE_Cls_128	DS_Full	2.58
SAE_Cls_128_Mem	DS_4th	1.90
SAE_Cls_128_Mem	DS_5th	1.61
SAE_Cls_128_Mem	DS_Full	1.77

This can lead to a severely unbalanced dataset which can lead to degradation in performance of the overall system.

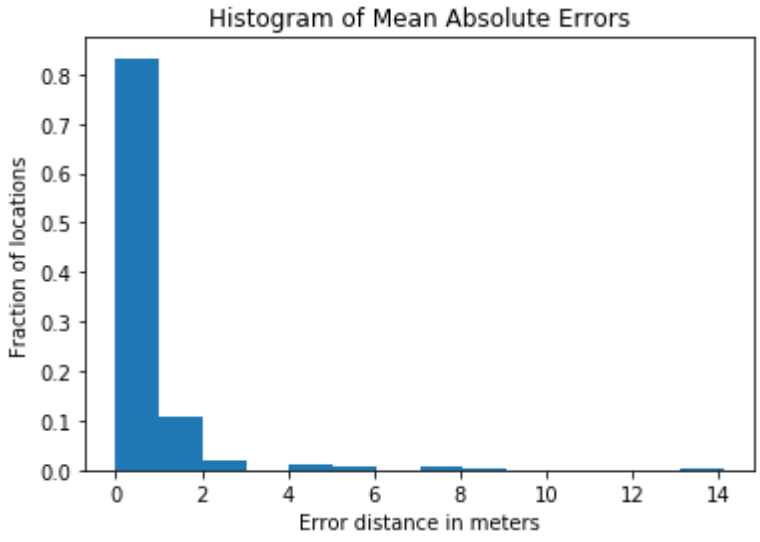

Figure 6-5. Histogram of errors with neural memory.

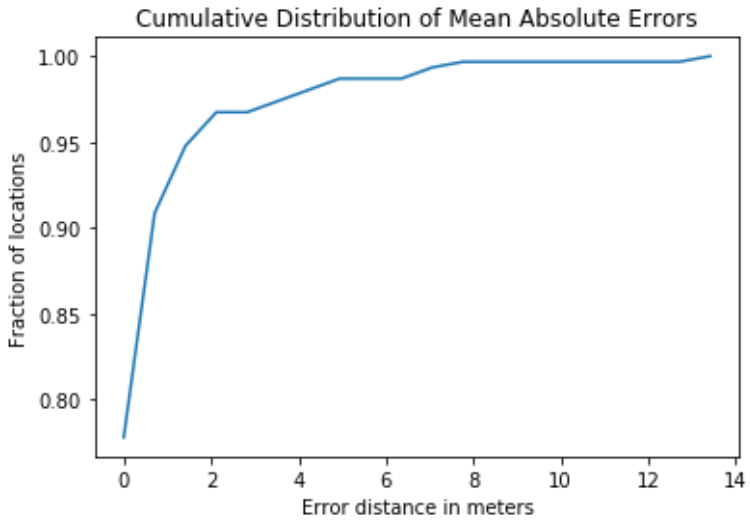

Figure 6-6. Cumulative distribution of errors with neural memory.

We also take a closer look at the distribution of the distance errors when using the neural memory. We see that over 95% of the errors made are within 2 meters. This is quite a is reasonable error when operating in a large indoor space. Given that the indoor space could be potentially hundreds of meters in length, such an error would be sufficient to locate the position of the entity in terms of the nearest room with reasonable accuracy.

In the next chapter, we conclude this thesis and discuss how this work could be extended in the future.

CHAPTER 7
CONCLUSION AND FUTURE RESEARCH DIRECTIONS

In this paper we presented DeepSpot, a scalable and practical indoor-localization system employing deep learning. We collected data in a real-world university office building using a semi-automated mobile robot. We used deep autoencoders to compress the data into feature-rich representations. We used these representations and experimented with regression, classification and neural memory-equipped classification and presented the results.

Given the flexibility of the proposed system, we believe that this methodology can be easily extended well beyond WiFi. Based on the nature of the premises that need to be mapped, other modalities of sensors, such as pervasive magnetic field strength detectors, Bluetooth receivers, ultrasound microphones, depth-sensing radar etc. can be incorporated seamlessly into this paradigm. The data from each modality can be individually compressed using deep stacked autoencoders to create feature vectors of desired lengths. They can be concatenated and used with an embedding network and a neural memory to output the predicted location coordinates. The neural memory is a relatively recent innovation in deep learning and could significantly improve localization accuracies.

Also, we believe that our system is scalable as it doesn’t rely on humans to perform the large-scale data collection. Since most commercial and industrial indoor spaces, such as office buildings, malls, industrial complexes etc. have standardized layouts, it would be possible to pre-program the entire path in advance, with periodic waypoints to correct for path deviations. The mobile robot platform can even be an

aerial drone, allowing it to map areas where the floor may not be suitable for wheeled or tracked robots.

We hope that this thesis will contribute to the development of more accurate, scalable and economical indoor-localization techniques for widespread adoption by commercial and industrial entities.

LIST OF REFERENCES

[1] B. L. McNaughton, L. L. Chen, and E. J. Markus, "'Dead Reckoning,' Landmark Learning, and the Sense of Direction: A Neurophysiological and Computational Hypothesis," *J. Cogn. Neurosci.*, vol. 3, no. 2, pp. 190–202, Apr. 1991.

[2] S. Beauregard and H. Haas, "Pedestrian dead reckoning: A basis for personal positioning."

[3] M. Kayton, *Navigation: Land, Sea, Air & Space*. .

[4] C. Fischer and H. Gellersen, "Location and navigation support for emergency responders: A survey," *IEEE Pervasive Comput.*, vol. 9, no. 1, pp. 38–47, 2010.

[5] H. Choset, "Coverage for robotics–A survey of recent results," *Ann. Math. Artif. Intell.*, vol. 31, no. 1, pp. 113–126, 2001.

[6] D. Titterton and J. L. Weston, *Strapdown inertial navigation technology*, vol. 17. IET, 2004.

[7] S. Duke, *United States military forces and installations in Europe*. Oxford University Press on Demand, 1989.

[8] A. Noureldin, T. B. Karamat, and J. Georgy, *Fundamentals of inertial navigation, satellite-based positioning and their integration*. Springer Science & Business Media, 2012.

[9] I. A. Getting, "Perspective/navigation-the global positioning system," *IEEE Spectr.*, vol. 30, no. 12, pp. 36–38, 1993.

[10] T. A. Stansell, "Transit, the navy navigation satellite system," *Navigation*, vol. 18, no. 1, pp. 93–109, 1971.

[11] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins, *Global positioning system: theory and practice*. Springer Science & Business Media, 2012.

[12] A. El-Rabbany, *Introduction to GPS: the global positioning system*. Artech house, 2002.

[13] T. H. Dixon, "An introduction to the Global Positioning System and some geological applications," *Rev. Geophys.*, vol. 29, no. 2, pp. 249–276, 1991.

[14] B. Hofmann-Wellenhof, H. Lichtenegger, and E. Wasle, *GNSS–global navigation satellite systems: GPS, GLONASS, Galileo, and more*. Springer Science & Business Media, 2007.

[15] S. Shen, N. Michael, and V. Kumar, “Autonomous multi-floor indoor navigation with a computationally constrained MAV,” in *Robotics and automation (ICRA), 2011 IEEE international conference on*, 2011, pp. 20–25.

[16] A. Wessels, X. Wang, R. Laur, and W. Lang, “Dynamic indoor localization using multilateration with RSSI in wireless sensor networks for transport logistics,” *Procedia Eng.*, vol. 5, pp. 220–223, 2010.

[17] S. M. George *et al.*, “DistressNet: a wireless ad hoc and sensor network architecture for situation management in disaster response,” *IEEE Commun. Mag.*, vol. 48, no. 3, 2010.

[18] Zafari, F., Gkelias, A., & Leung, K. (2017). A Survey of Indoor Localization Systems and Technologies. *arXiv preprint arXiv:1709.01015*.

[19] Liu, K., Liu, X., & Li, X. (2013, June). Guoguo: Enabling fine-grained indoor localization via smartphone. In *Proceeding of the 11th annual international conference on Mobile systems, applications, and services* (pp. 235-248). ACM.

[20] B. P. Crow, I. Widjaja, J. G. Kim, and P. T. Sakai, “IEEE 802.11 wireless local area networks,” *IEEE Commun. Mag.*, vol. 35, no. 9, pp. 116–126, 1997.

[21] C. Bisdikian, “An overview of the Bluetooth wireless technology,” *IEEE Commun. Mag.*, vol. 39, no. 12, pp. 86–94, 2001.

[22] C. Gomez, J. Oller, and J. Paradells, “Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology,” *Sensors*, vol. 12, no. 9, pp. 11734–11753, 2012.

[23] K. Finkenzeller, *RFID handbook: fundamentals and applications in contactless smart cards, radio frequency identification and near-field communication*. John Wiley & Sons, 2010.

[24] S. A. Weis, S. E. Sarma, R. L. Rivest, and D. W. Engels, “Security and privacy aspects of low-cost radio frequency identification systems,” in *Security in pervasive computing*, Springer, 2004, pp. 201–212.

[25] S. Gezici *et al.*, “Localization via ultra-wideband radios: a look at positioning aspects for future sensor networks,” *IEEE Signal Process. Mag.*, vol. 22, no. 4, pp. 70–84, 2005.

[26] L. Yang and G. B. Giannakis, “Ultra-wideband communications: an idea whose time has come,” *IEEE Signal Process. Mag.*, vol. 21, no. 6, pp. 26–54, 2004.

[27] H. Zimmermann, “OSI reference model--The ISO model of architecture for open systems interconnection,” *IEEE Trans. Commun.*, vol. 28, no. 4, pp. 425–432, 1980.

[28] J. D. Day and H. Zimmermann, “The OSI reference model,” *Proc. IEEE*, vol. 71, no. 12, pp. 1334–1340, 1983.

[29] H. Liu, H. Darabi, P. Banerjee, and J. Liu, “Survey of wireless indoor positioning techniques and systems,” *IEEE Trans. Syst. Man, Cybern. Part C (Applications Rev.*, vol. 37, no. 6, pp. 1067–1080, 2007.

[30] C. Koweerawong, K. Wipusitwarakun, and K. Kaemarungsi, “Indoor localization improvement via adaptive RSS fingerprinting database,” in *Information Networking (ICOIN), 2013 International Conference on*, 2013, pp. 412–416.

[31] M. Stella, M. Russo, and D. Begusic, “Location determination in indoor environment based on RSS fingerprinting and artificial neural network,” in *Telecommunications, 2007. ConTel 2007. 9th International Conference on*, 2007, pp. 301–306.

[32] D. E. Manolakis, “Efficient solution and performance analysis of 3-D position estimation by trilateration,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 32, no. 4, pp. 1239–1248, 1996.

[33] A. Kushki, K. N. Plataniotis, and A. N. Venetsanopoulos, “Kernel-based positioning in wireless local area networks,” *IEEE Trans. Mob. Comput.*, vol. 6, no. 6, 2007.

[34] A. J. Smola and B. Schölkopf, “A tutorial on support vector regression,” *Stat. Comput.*, vol. 14, no. 3, pp. 199–222, 2004.

[35] B. Schölkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[36] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum likelihood from incomplete data via the EM algorithm,” *J. R. Stat. Soc. Ser. B*, pp. 1–38, 1977.

[37] X. Yao, “Evolving artificial neural networks,” *Proc. IEEE*, vol. 87, no. 9, pp. 1423–1447, 1999.

[38] M. H. Hassoun, *Fundamentals of artificial neural networks*. MIT press, 1995.

[39] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[40] O. Muñoz-Medina, J. Vidal, and A. Agustin, “Linear transceiver design in nonregenerative relays with channel state information,” *IEEE Trans. Signal Process.*, vol. 55, no. 6, pp. 2593–2604, 2007.

[41] Z. Yang, Z. Zhou, and Y. Liu, “From RSSI to CSI: Indoor localization via channel response,” *ACM Comput. Surv.*, vol. 46, no. 2, p. 25, 2013.

[42] B. Yegnanarayana, *Artificial neural networks*. PHI Learning Pvt. Ltd., 2009.

[43] J. M. Zurada, *Introduction to artificial neural systems*, vol. 8. West St. Paul, 1992.

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

[45] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[46] D. H. Hubel and T. N. Wiesel, “Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex,” *J. Physiol.*, vol. 160, no. 1, pp. 106–154, 1962.

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

[48] S. Ruder, “An overview of gradient descent optimization algorithms,” *arXiv Prepr. arXiv1609.04747*, 2016.

[49] X. Glorot, A. Bordes, and Y. Bengio, “Deep sparse rectifier neural networks,” in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011, pp. 315–323.

[50] H. B. Demuth, M. H. Beale, O. De Jess, and M. T. Hagan, *Neural network design*. Martin Hagan, 2014.

[51] R. Pascanu, T. Mikolov, and Y. Bengio, “On the difficulty of training recurrent neural networks,” in *International Conference on Machine Learning*, 2013, pp. 1310–1318.

[52] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[53] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” *arXiv Prepr. arXiv1412.3555*, 2014.

[54] P. Baldi, “Autoencoders, unsupervised learning, and deep architectures,” in *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, 2012, pp. 37–49.

[55] Q. V Le, “Building high-level features using large scale unsupervised learning,” in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, 2013, pp. 8595–8598.

[56] M.-T. Luong, H. Pham, and C. D. Manning, “Effective approaches to attention-based neural machine translation,” *arXiv Prepr. arXiv1508.04025*, 2015.

[57] C. Stuart, Y. Takahashi, and H. Umezawa, “Mixed-system brain dynamics: neural memory as a macroscopic ordered state,” *Found. Phys.*, vol. 9, no. 3–4, pp. 301–327, 1979.

[58] Ł. Kaiser, O. Nachum, A. Roy, and S. Bengio, “Learning to remember rare events,” *arXiv Prepr. arXiv1703.03129*, 2017.

[59] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” *arXiv Prepr. arXiv1409.0473*, 2014.

[60] Y. Wu *et al.*, “Google’s neural machine translation system: Bridging the gap between human and machine translation,” *arXiv Prepr. arXiv1609.08144*, 2016.

[61] J. Weston, S. Chopra, and A. Bordes, “Memory networks,” *arXiv Prepr. arXiv1410.3916*, 2014.

[62] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, “One-shot learning with memory-augmented neural networks,” *arXiv Prepr. arXiv1605.06065*, 2016.

[63] V. Cherkassky and F. M. Mulier, *Learning from data: concepts, theory, and methods*. John Wiley & Sons, 2007.

[64] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin, *Learning from data*, vol. 4. AMLBook New York, NY, USA:, 2012.

[65] S. J. Pan and Q. Yang, “A survey on transfer learning,” *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, 2010.

[66] M. T. Mason, “Creation myths: The beginnings of robotics research,” *IEEE Robot. Autom. Mag.*, vol. 19, no. 2, pp. 72–77, 2012.

[67] I. Asimov, *I, robot*, vol. 1. Spectra, 2004.

[68] S. Kassel, “Lunokhod-1 Soviet lunar surface vehicle,” RAND CORP SANTA MONICA CA, 1971.

[69] Z. Medvedev, *The legacy of Chernobyl*. WW Norton & Company, 1992.

[70] J. Forlizzi and C. DiSalvo, “Service robots in the domestic environment: a study of the roomba vacuum in the home,” in *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, 2006, pp. 258–265.

[71] A. P. Gritti *et al.*, “Kinect-based people detection and tracking from small-footprint ground robots,” in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, 2014, pp. 4096–4103.

[72] “Amazon Robotics.” [Online]. Available: https://www.amazonrobotics.com/#/.

[73] “Grey Orange Robotics.” [Online]. Available: http://www.greyorange.com/.

[74] R. T. Azuma, “A survey of augmented reality,” *Presence Teleoperators virtual Environ.*, vol. 6, no. 4, pp. 355–385, 1997.

[75] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre, “Recent advances in augmented reality,” *IEEE Comput. Graph. Appl.*, vol. 21, no. 6, pp. 34–47, 2001.

[76] “Nvidia Jetson TX2.” [Online]. Available: https://developer.nvidia.com/embedded/buy/jetson-tx2.

[77] “Kobuki Robot.” [Online]. Available: http://kobuki.yujinrobot.com/.

[78] “Tacon Remote Controlled Car.” [Online]. Available: http://www.nitrorcx.com/tacon.html.

[79] M. Bojarski *et al.*, “End to end learning for self-driving cars,” *arXiv Prepr. arXiv1604.07316*, 2016.

[80] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[81] L. Wang, S. Guo, W. Huang, and Y. Qiao, “Places205-vggnet models for scene recognition,” *arXiv Prepr. arXiv1508.01667*, 2015.

[82] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv Prepr. arXiv1412.6980*, 2014.

[83] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, “Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion,” *J. Mach. Learn. Res.*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[84] B. Leng, S. Guo, X. Zhang, and Z. Xiong, “3D object retrieval with stacked local convolutional autoencoder,” *Signal Processing*, vol. 112, pp. 119–128, 2015.

[85] Wang, X., Gao, L., Mao, S., & Pandey, S. (2017). CSI-based fingerprinting for indoor localization: A deep learning approach. *IEEE Transactions on Vehicular Technology*, *66*(1), 763-776.

[86] Jekabsons, G., Kairish, V., & Zuravlyov, V. (2011). An analysis of Wi-Fi based indoor positioning accuracy. Scientific Journal of Riga Technical University. Computer Sciences, 44(1), 131-137.

BIOGRAPHICAL SKETCH

Rahul Sengupta was born in Mumbai, India and did his schooling in several cities across India. He earned an Integrated Master of Science degree in chemistry and a Bachelor of Science degree in electrical and electronics engineering at BITS-Pilani K.K. Birla Goa Campus, India. He then worked at Ericsson, India as a System Software Engineer and as a Consultant at the Data Mining Lab, General Electric Research, India. He then pursued a Master of Science degree in computer science at the University of Florida, United States of America. His academic interests include Machine Learning, Deep Learning, Agent-based Simulation and Artificial Intelligence.