Smartphone-based Location Tracking using Recurrent Neural Networks

Xijia Wei

Master of Science

Artificial Intelligence

School of Informatics

University of Edinburgh

2018

Abstract

Various solutions of indoor positioning system have been proposed to compensate for the impractical use of the GPS signal for the indoor environment. Current solutions are designed to estimate the location of people inside buildings by taking advantages of the built-in smartphone sensors such as magnetometer, accelerometer, gyroscope and WiFi received signal strength [1]. However, the accuracy and reliability of these system highly rely on the well-defined equations which identify the relationship between sensor data and location information [2].

By relying on machine learning to automatically learn features in a large datasets of sensor data is a potentially better approach to improve the robustness of indoor positioning systems rather than hand-tuning the signal processing equations [3]. When tracking the movement data of walking inside a building, those data are processed as sequential data with the location label. Hence, a recurrent neural network of long short term memory based location tracking system is proposed as a new solution for indoor positioning problems.

Hence, a recurrent neural network based location tracking system is proposed as a new solution for indoor positioning problems. This work presents a range of RNN models for tracking mobile phone location inside a building. In particular, this involved collecting a large dataset of movement sensor data and signal strength data with location labels for the same indoor routine as the training data. Training is done by feeding the data into RNN models to learn the pattern features by setting a time window. At run time, the RNN model predicts the next location based on the previous input features and the time distributed prediction location outputs.

From a system perspective, our solution made a balance between energy consumption and inference accuracy, which is achieved by compressing input data dimensionality (down sampling and PCA). The result shows that with the estimation accuracy of 80%, the system error is 5.57 metres in average.

Acknowledgements

I am writing this acknowledge to express my most sincere appreciation for all the people who gave me great support during my postgraduate study in artificial intelligence.

Firstly, I would like to offer my most profound gratitude to my supervisor Valentin Radu. Since the last six months, I have learned a lot from him. The critical thinking; the academic attitude; the brave attempt and the patient practice. He provides me with a flexible research environment that allows me to propose my idea and meanwhile offers me great help whenever there is a new idea coming up. Also, I would like to thanks for his trust in me and my work and provides me with an opportunity to present the work in MobiUK in Cambridge. I feel lucky and do enjoy the time working with him and learning from him. Few words can not express my sincere gratitude. All in all, I appreciate my supervisor Valentin Radu who inspires me in both of my academic and daily life.

Secondly, thanks to all of my friends who gave me help and supports in my study and life.

Finally, I would give my biggest thanks to my parents. Without the inspiration, drive, and support from my parents, I might not be the person I am today.

Last but not the least, thanks, my University of Edinburgh.

(This thesis is hereby dedicated to all the people that I mentioned above)

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Xijia Wei*)

Table of Contents

Chapter 1

Introduction

Predicting object's indoor location is a challenging task due to the limitation of Global Positioning System (GPS). The GPS signal is affected by the indoor building materials. Instead, an indoor location tracking system is supposed make use of the built-in smartphone sensors such as accelerometer, gyroscope, magnetometer as well as the received signal strength (RSS) for positioning [3]. In this project, we aim to develop a machine learning based indoor location tracking system using only the sequential signal of built-in smartphone sensors.

1.1 Objective

GPS is a system level navigation system relying on satellite signals. However, concerning the indoor environment, the GPS signal is sheltered out by building. Therefore, an indoor positioning system (IPS) is given which relies on alternative signals such as WiFi, Bluetooth, Infrared RSS and movement tracking sensors for locationing tracking system [4].

Integrating indoor location tracking into smartphone has become a favourite research topic in recent years as smartphone provides all the necessary built-in sensors for location tracking such as collecting acceleration, orientation, magnetic field and RSS data. Based on the RSS data and other sensor data received from the access points (AP) or the smartphone, the system could return a location estimation by calculating user's location based on set of well-defined equations. Currently, most indoor localisation systems contain two main techniques, WiFi Fingerprinting and Pedestrian Dead Reckoning (PDR) [5]. These systems work on a firm set of equations defining the possible mobility frame, such as step counting, direction estimation and WiFi fin-

gerprint matching. However, the accuracy is usually affected by the variations of the signal distribution so that those equations need to be up to date. This process adds to the complexity of the indoor positioning system which impedes its deployment.

Recently, using machine learning algorithms for indoor location tracking becomes popular as it avoids artificial intervention which automatically learns the dependencies between the sensor features and the location labels. It is extremely hard to extra relevant hand-crafted features from sensor signals, a task that is relatively easy for machine learning. Meanwhile, the machine learning based system shows its advantages of higher robustness and lower application costs [6]. Not many previous works use machine learning to extract these features through machine learning for indoor positioning system. One specific class of ML techniques is to use the artificial neural network such as backpropagation algorithm for location tracking.

In this work, we aim to build a recurrent neural network (RNN) regression model for the indoor location tracking system. RNN could predict the location by making use of the sequential data of acceleration, orientation and magnetic field data collected from built-in smartphone sensors. **We believe the RNN based model could avoid any intervention and hand-tuned equations by relying on data alone and machine learning end-to-end to learn the features for locationing automatically.**

Data is of vitally importance to the model training process. The project starts by collecting the raw sensor data during the walking in an indoor environment. Specifically, we collected training sensor data in several runs over the same path. The whole procedures include collecting the training data by matching the raw sensor data with the ground truth locations. Selecting a proper time window to feed the training data into the RNN model for training and tune the model structure. Meanwhile, implementing data processing such as overlapping to increase the data amount but at the same time using PCA and downsampling to compress the data size of the input to improve training speed. We use one of the RNN called Long Short-Term Memory (LSTM) for extract locationing features. LSTM shows its advantage in dealing with sequential data [7] [8]. It is typically used for speech and translation tasks. Similarly, by feeding the sequential training data into the LSTM network, we find that even with a reduced memory functionality, the well-trained model could provide the user's real-time location estimation of *(x,y)*. Table 1.1 lists the main procedures of the experiment.

1	Select the smartphone sensors which are suitable for positioning
2	Build the training dataset (Collect sensor data, Label matching)
3	Data processing (Overlapping, Down Sampling, PCA)
4	Develop an RNN model for indoor location tracking
5	Optimise the RNN model (Adjust model setting, change data structure)
6	Evaluation and Test

Table 1.1: objectives

1.2 Motivation

We believe that using machine learning to solve indoor positioning problems instead of relies on a firm set of equations like traditional fingerprinting or PDR techniques could provide automated solutions for indoor locationing [9]. By taking the advantages of LSTM that learn feature from sequential sensor data, we could build a self-learn model for location tracking. Without artificial intervention, the system can update its parameters by feeding further training data if the indoor environment is changed. The model can easily be applied in many indoor environments such as parking navigation, patients tracking in hospitals and location-based advertisement in shopping malls.

1.3 Results

By collecting and processing the sensor data in time sequential order, we provide a new solution that uses RNN for indoor location tracking. After comparing different models configured with different settings as well as the input data structure, we select the stateless LSTM model with downsampling 90% overlapping data for location tracking. The testing result shows that it achieves an average estimation accuracy of 5.57 metres.

1.4 Thesis Structure

The rest of this thesis is organised as follows:

Chapter 2 Background: Introduce the location based service and the indoor location tracking system. Specifically, it includes indoor positioning algorithms of both traditional and machine learning based methods.

Chapter 3 Experiment Design: Explain the methodology used in this project

such as the data processes, recurrent neural networks with different structures and the evaluation methods.

Chapter 4 Data Processes: Describe the data collection and processing procedures.

Chapter 5 Model Selection and Evaluation: Explain the model building, training and comparison sections with the evaluation and testing the performance of the LSTM based locationing system.

Chapter 6 Conclusion: Summaries the experiment and introduce the later works with future planning.

Chapter 2

Background

2.1 Location-Based Service

Location-Based Service (LBS) is an application-level information service based on user's location [10]. Its applications include navigation system, location tracking system and promotion notification system that integrates with the map on smartphones based on user's location.

Federal Communications Commission (FCC) of the United States is the first organisation that deployed LBS which known as Enhanced 911 (E-911) Act in June 1996. The reason to integrate LBS to the 911 emergency service is that the number of calls of 911 made by mobile phones increases 30% by each year. However, with the mobile phone incoming call, it is hard to track caller's location. E-911 Act required that the system has to able to provide caller's location information with the precision of 125 metres by October 2001 while the confidence probability of achieving this requirement should be higher than 67% [11]. Similarly, the European Union proposed a similar regulation known as E-122 Act in 2002.

With the development of LBS, it has been widely used in not only in national defence but also commercial behaviours. The applications of LBS mainly includes the following four aspects.

Notification System: Notification such as shopping promotions, travel guidance and public safety will be pushed on the user's smartphone when the user enters the area that covered by those services.

Location Search System: Searching information based on location such as nearby friends finding or public bus searching.

Location Tracking System: Track object's location such as navigation system,

tracking patients in hospital, car parking system and bus service tracking.

Games: Combining a player's location with virtual scenes such as Pokemon Go [12]. (An augmented reality game to catch monsters based on instant user's location provided by Nintendo)

2.2 Indoor Location Tracking System

There is an increasing demand for location-based service in the indoor environment such as indoor navigation and location search service. However, due to the limitation of the GPS that the satellite signal is distorted or attenuated affected by building materials, GPS cannot usually work in an indoor environment. Therefore, the indoor positioning system is proposed to replace GPS for indoor location tracking service.

The indoor location tracking system or indoor positioning system is a system level service that tracks user's indoor location information based on smartphone built-in sensor signals such as accelerator, gyroscope and radio wave signals such as earth magnetic field signal, acoustic signals, WiFi signals, Bluetooth signals.

WiFi: Through the access points of the wireless local area network (WLAN), location estimations could be made by the position information of the network node's signal. Based on the signal propagation equations, the system could return the location of the mobile devices. Generally, the highest accuracy is from 1 to 20 metres. However, WiFi-based locationing errors become larger when the location measurement is only based on the real-time connected access point with no reference to nearby signal strength composition maps. Meanwhile, WiFi signal covers the radius of around 90 meters and sometimes it is affected by irrelevant signal interference. On the other side, power consumption is relatively high.[13]

Bluetooth: Bluetooth communication is a short-range wireless transmission method with a lower power consumption compared to WiFi. Bluetooth-based locationing requires the installation of several Bluetooth APs under the multiple user connection modes, estimating user location is based on the signal strength of the Bluetooth. Bluetooth locationing is usually suitable for a small environment such as the warehouse. However, for indoor environments with more complex structure, Bluetooth locationing system is slightly unstable because of noise from other electronic devices.[14]

Infrared: Infrared locationing estimates the user's location based on The infrared signal which achieves higher positioning accuracy compared to WiFi and Bluetooth system. However, due to infrared ray physical feature, it is impossible to go through

the obstacles. Meanwhile, the transmission distance is short as it requires the direct transmission. Furthermore, infrared receivers are necessary for installing which increases the system's overall cost.[15]

ZigBee: ZigBee is a short-range indoor locationing technique. The sensors work based on the signal transmitted from a signal generator to the signal receiver via the radio wave with low power consumption. Meanwhile, it is low cost.[16]

2.3 Traditional Indoor Positioning Algorithms

Traditional indoor positioning algorithms work on a set of well-defined numerical equations. Here, we mainly introduce the pedestrian dead reckoning that works on inertial sensor and the fingerprint (range-free) positioning as well as the range-based algorithms using both inertial sensors and radio signals.

2.3.1 Pedestrian Dead Reckoning

Dead reckoning is a navigation method that estimates an object's current position based on the previous location. It is different from the GPS or Indoor positioning system (GPS) that relies on the external radio signals. It uses the inertial built-in smartphone sensors such as accelerator, gyroscope and magnetometer.

Smartphone provides all the necessary sensors for inertial based navigation. Specifically, the accelerometer could be used as a pedometer for calculating the step and step distance; Magnetometer can be used as a compass for heading detection. PDR provides an alternative navigation solution when other external radio signals are not available for localisation.

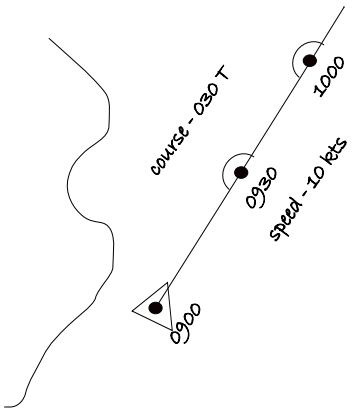


Figure 2.1: Pedestrian Dead Reckoning Workflow

Figure 2.1 shows the workflow of the PDR. By using the inertial sensor, the algorithm calculates the step distance and the angle or direction of the front of the user using the compass. Based on the previous position or initial starting point, the system can predict the next footstep with the step distance and direction.

It is a simple implementation that only using the inertial sensors. The accuracy of the next estimation depends on whether the current location is accurate or not. In general, estimation accuracy is affected by the sensor sensitivity. PDR requires a set of well-defined equations that accurately utilise the accelerator and compass to estimate user's next location. This limits its applications. Meanwhile, other activities such as climbing stairs between different floors is a challenge of using only PDR algorithm [17].

2.3.2 Range-free Positioning

Range-free Positioning mainly includes fingerprint algorithm which relies on the received signal strength (RSS). Fingerprint positioning algorithms contain two phases: offline data collection stage and online positioning stage.

Offline data collection phase: Collect RSS signal in the indoor environment and match location labels with the signal to create fingerprint library. For example, WiFi-based fingerprint requires to build the data library that includes all the RSS information received from each place with the location labels. Specifically, the format of signal data is (RSS_1, RSS_2, ..., RSS_j, ..., RSS_n) at point (X_i, Y_i) while n is the total number of access points (AP) measured by the mobile.

Online positioning phase: During the online real-time positioning phase, the real-time sensor data would be received from a user's mobile phone and compared with the RSS data stored in the fingerprint library. Based on the similarity between the real-time data and fingerprint data, the possible location label would be returned to users as the estimation location.

To be specific, the fingerprint positioning system includes deterministic based and probabilistic localisation algorithms.

2.3.2.1 Deterministic Positioning Algorithm

One of the deterministic positioning systems is called RADAR is given by Microsoft [18]. RADAR consists of two stages of the offline data collection stage and the online positioning stage as explained in the fingerprinting algorithm.

During the data collection phase, the room is divided into clusters with different precision requirements. The system builds the fingerprinting library formatted as $F_i = (x_i, y_i, V_1, V_2, \ldots, V_m)$ where V_m is the RSS measurement. During the online locationing phase, RADAR works on the algorithm of KNN. By matching the real-time signal data with the data stored in the fingerprint library, the location of the mobile could be calculated by the Euclidean Distance is shown in equation 2.3.2.1.

$$Distance = \sqrt{(v_1 - v_1')^2 + (v_2 - v_2')^2 + \ldots, +(v_m - v_m')^2} \tag{2.1}$$

After deriving the Euclidean distance between the mobile node with every location stored in the library dataset, selecting the first K shortest Euclidean distance from the fingerprint library. The final position estimation is calculated by getting the average value by KNN or the weighted locations.

2.3.2.2 Probabilistic Positioning Algorithm

Horus positioning system is based on probability theory. It belongs to the fingerprinting idea that includes offline data collection stage and online locationing phase. However, the difference compared to deterministic positioning algorithm such as RADAR is during the online positioning phase. Under the assumption that RSS signal contributes to a Gaussian Distribution, Horus estimates user's location based on the statistical distribution of the RSS [19].

During the offline data collection stage, building the fingerprinting library which contains not only the RSS value but also the mean and variance variables which represents the likelihood of how much of the sampling data belongs to the ground truth location. Regarding the online positioning phase, it predicts user's location based on the Naive Bayesian and the posterior probability.

Specifically, selecting the maximum expectation as the estimated location from all the expectation calculated based on the posterior probability shown in equation 2.3.2.2.

$$\begin{aligned} p(l|v) &= \frac{p(v|l)p(l)}{p(v)} \\ E[l|v] &= \sum_{i=1}^{n} l_i p(l_i|v) \end{aligned} \tag{2.2}$$

Under the assumption that each RSS value is independent which follows the Gaussian distribution, $p(v|l)$ could be derived as equation 2.3.2.2.

$$p(v|l) \propto \prod_{j=1}^{m} \frac{1}{\sigma_j} exp(-\frac{(v_j - u_j)^2}{2\sigma_j^2}) \tag{2.3}$$

Probability-based positioning system works on a set of well-defined equations with the assumption of RSS follows the normal distribution. However, in the real situation such as a complex indoor environment, the signal distribution frequently changes by many reasons such as refurbishment and moving electronic devices.

2.3.3 Range-based Positioning

The range-based positioning system includes trilateral and triangular methods which estimate mobile's location based on the high precision signal such as signal receiving angle and the transmission distance between the mobile nodes and the signal generator.

2.3.3.1 Trilateration

By using the Time of Arrival (TOA) algorithm or signal propagation algorithm, the distance R between the mobile A and the beacon nodes B could be calculated [20]. Here, R represents the radius of a circle which the circle centre is beacon B while mobile A is on the circumference. Figure 2.2 shows the tree circle which represents three signal generators. Assume that location of the mobile node is (x_0, y_0) while three signal generators' coordinate are (x_1, y_1), (x_2, y_2), (x_3, y_3). The cross point of three circles is the estimation of the mobile location.

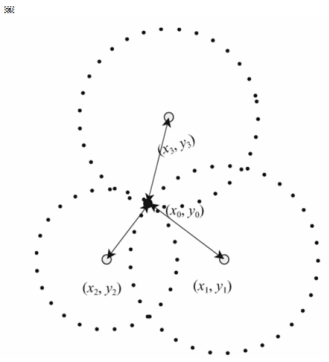


Figure 2.2

This cross point could be calculated by the equation 2.3.3.1:

$$\begin{aligned}(x_1 - x_0)^2 + (y_1 - y_0)^2 &= d_1^2 \\ (x_2 - x_0)^2 + (y_2 - y_0)^2 &= d_2^2 \\ (x_3 - x_0)^2 + (y_3 - y_0)^2 &= d_3^2\end{aligned} \tag{2.4}$$

After elimination from the above equations, we could get the following equations:

$$\begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \begin{bmatrix} 2(x_1 - x_3) & 2(y_1 - y_3) \\ 2(x_2 - x_3) & 2(y_2 - y_3) \end{bmatrix} = \begin{bmatrix} x_1^2 - x_3^2 + y_1^2 - y_3^2 + r_3^2 - r_1^2 \\ x_2^2 - x_3^2 + y_2^2 - y_3^2 + r_3^2 - r_2^2 \end{bmatrix} \tag{2.5}$$

2.3.3.2 Triangulation

Triangulation locationing algorithm using the Arrival of Angle (AOA) to estimate mobile's location [21]. Specifically, mobile's position is estimated based on the signal received angle from two signal generators. The antenna installed in the mobile phone could measure the received signal angle from the beacon. This is achieved by calculating the azimuth direction of the mobile to the beacon. Figure 2.3 illustrates the AOA algorithm. Here, N represents the mobile while A1 and A2 represent for the signal generator.

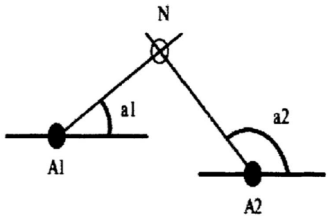


Figure 2.3

If the mobile location is $N(x_0, y_0)$, α_1 and α_2 represent the angle of signal generators of $A_1(x_1, y_1)$ and $A_2(x_2, y_2)$ respectively. The angle could be calculated by the following equation.

$$\tan(\alpha_i) = \frac{x_0 - x_i}{y_0 - y_1}, i = 1, 2 \tag{2.6}$$

2.4 Machine Learning Indoor Positioning Algorithms

Instead of the range-free and range-based positioning algorithm that require the fine-tuning equations to estimate the locations, machine learning provides automatic tuning solutions for the locationing systems. Currently, there are insufficient applications that are integrating machine learning algorithms into an indoor positioning system.

2.4.1 Neural Network Based Positioning

Battiti provided a machine learning positioning solution based on an artificial neural network (ANN) using WiFi received signal strength [22]. It is developed based on

a feed-forward neural network which uses the RSS data as the training data and the matched locations as the labels. Their experiment was deployed in a room with 56 WiFi routers, and each measurement was generated by getting the average value of 100 scans. Battiti's research proves that ANN is possible to apply to the indoor locationing system which reduces the requirement of the external locationing devices such as a large number of signal generators (beacon). However, his experiment requires at least three access points.

Borenovic provided an optimised solution based on ANN locationing system which includes a cascade structure of two parallel neural networks [23]. During the data collection phase, the WiFi RSS training data is collected from only eight access points. Therefore, the training data is formatted as $(X_i, Y_i, \mathrm{RSS}_1, \mathrm{RSS}_2, \ldots, \mathrm{RSS}_8)$. The indoor experiment environment is divided into 12 blocks while each one of the blocks is further divided into small grids. Figure 2.4 shows the structure of the cascade artificial neural network. The system firstly feeds the data into type 2 ANN which returns a block estimation. Secondly, reuse the data and feed into type 1 ANN which returns the grid estimation of the block predicted from the type 1 ANN. This double parallel ANN

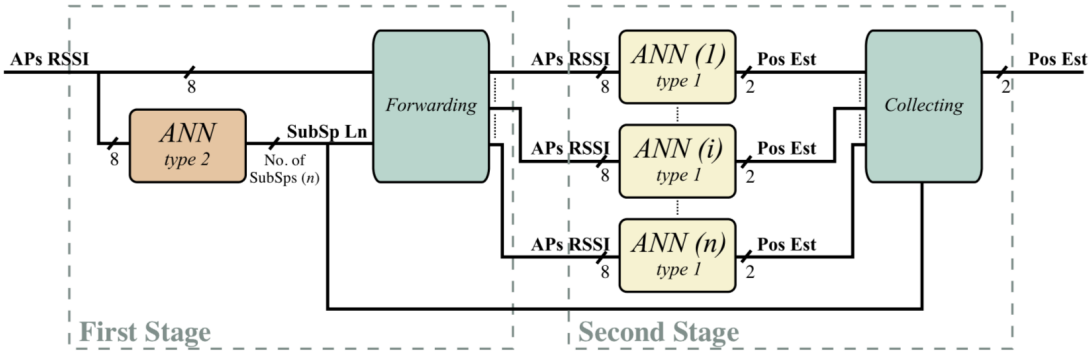


Figure 2.4: Cascade structure of ANN locationing system

structure reduces the complexity of the system by firstly estimate a large area then scale into a smaller location prediction. Meanwhile, it has a relatively loose requirement of the number of the signal generators which could be less than three. The overall accuracy is 8.33 metres, compared to the fingerprinting algorithm of Fuzzy logic [24] with the error of 16.1 metres.

2.4.2 Decision Tree Based Positioning

CaDet is another machine learning based indoor positioning system that using decision tree algorithm for location estimations. It works on the fundamental theory of fingerprint positioning as well that first learn the feature from the fingerprint library and then compare the real-time data with the library to provide the position estimations [25].

CaDet used 25 WiFi routers in a room. During the data collection stage, the room is divided into clusters. Five access points' signal covers each cluster. By comparing the information entropy of the RSS, the system could select several access points (less than the total number of the APs) for locationing shown in equation 2.7.

$$InfoGain(AP_i) = H(G) - H(G|AP_i) \tag{2.7}$$

Here, the information gain is the subtraction that using the entropy of the cluster H(G) to minus the conditional entropy H(G|AP_i). After deciding several APs based on the information gain value, the system has filtered out the necessary APs for locationing. Figure 2.5 shows the structure of the decision tree. It is growth based on the RSS value from different AP. Here, growing starting from AP2, the tree develops into three sub end with three RSS range. Later, it continues growing based on the RSS received from AP1, AP4 and AP1. As shown in figure 2.5, each cluster of the room contains a decision tree that ends with a particular location.

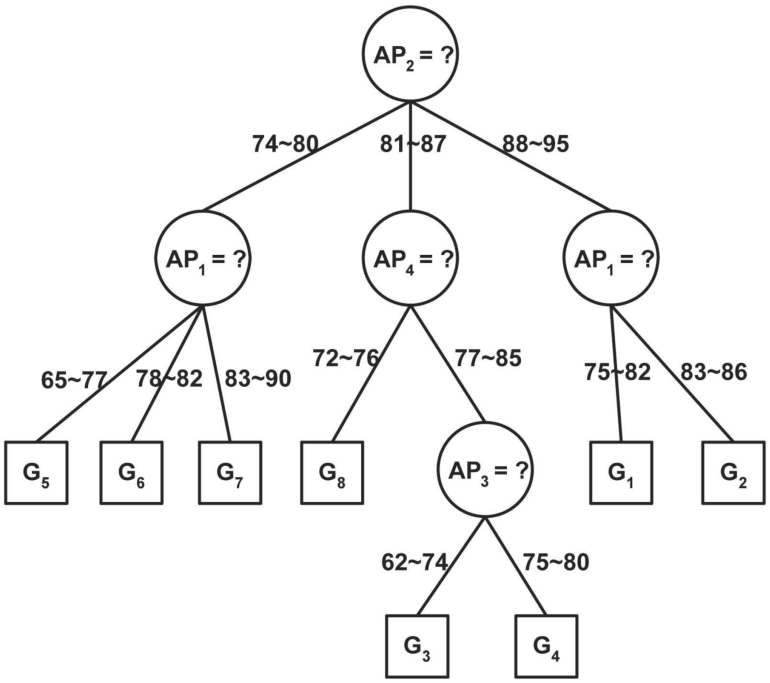


Figure 2.5: Decision tree of a cluster

During the real-time locationing phase, CaDet estimates the location by firstly choosing the Kth sub-grids from the clusters. It compares the difference between the real-time data with fingerprinting library data and then following the flow of the decision tree generated during the training process until it reaches the end of the tree with an estimated location value.

By clustering a room into six groups, CaDet achieves a locationing accuracy of 82.3%. It shows the decision tree based locationing system is reliable for the indoor positioning system. It could deal with a large space indoor environment that clustering into some subgroups and then create decision trees for each of the clusters.

Chapter 3

Experiment Design

3.1 Methodology

The indoor location tracking system includes two phases. The first phase is to build the data library. Specifically, walking through the route that selected for experiment and collects the smartphone built-in sensor data with the location information that indicates the ground truth locations. The second phase is to build the machine learning model for real-time user location estimations. Figure 3.1 shows the workflow of the location tracking system in this experiment. We start by collecting the suitable data for localisation. After that, data processing methods are implemented such as normalisation and dimension reduction for improving accuracy and efficiency. Those processed data are later used for the model as the training dataset. Here, we aim to use a recurrent neural network called Long Short Term Memory (LSTM) as the locationing algorithm. The cumulative distribution function (CDF) is used for evaluation. After we select the proper model, we would use it for real-time data based location estimations.

3.1.1 Data Collection

By calling the Android API of SensorManager, We could choose the refreshing sampling rate defined by Android SDK, written in code 3.1.1. Therefore, with recording the location of latitude and longitude during the data acquisition, we could build the data library with formats as shown in table 3.1.

```
private static final int ACCELEROMETER_SAMPLING = SensorManager.SENSOR_DELAY_GAME;
private static final int MAGNETIC_SAMPLING = SensorManager.SENSOR_DELAY_GAME;
private static final int GYROSCOPE_SAMPLING = SensorManager.SENSOR_DELAY_GAME;
```

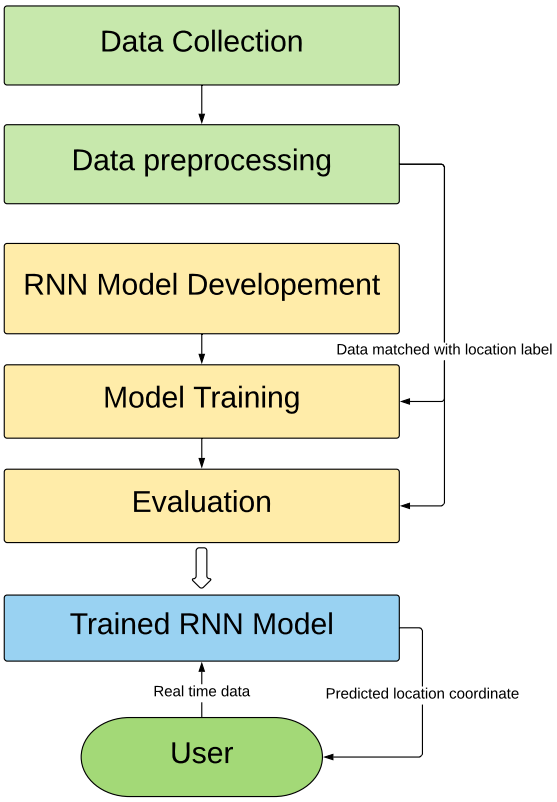


Figure 3.1: Location tracking system work flow

Place	Acceleration	Gyroscope	Magnetic Field	WiFi	Coordinate
1	$X1_1, Y1_1, Z1_1$	$X1_1, Y1_1, Z1_1$	$X1_1, Y1_1, Z1_1$	$RSS1_1, ...$	x_1, y_1
...	...	...	...	...	...
i	Xi_1, Yi_1, Zi_1	Xi_1, Yi_1, Zi_1	Xi_1, Yi_1, Zi_1	$RSSi_1, ...$	x_i, y_i

Table 3.1: Data library

Hence, we collect the data of accelerator, gyroscope and magnetometer with the WiFi scans signal used for location tracking system.

3.1.2 Data Preprocessing

Principal component analysis: In machine learning area, in order to avoid the curse of dimensionality, dimensionality reduction is a method to reduce the size of training

data by calculating principal variables [26]. Here, we use Principal component analysis to compress the input data size. PCA is a dimension reduction technique which is usually used for machine learning area to compress the data size of model inputs. It is a statistical method that transforms a number of possibly correlated observations to a smaller size of linearly uncorrelated variables which is called principal components. It offers a linear mapping transformation from a higher-dimensional size of data into a lower-dimensional space. During the transformation, new variables are generated which replaces the original variables but contain similar data features. The covariance matrix is limited by the eigenvectors which relate to the largest eigenvalues (principal components). It is used to reconstruct a bigger fraction of the variance from the original dataset. Hence, by calculating the eigenvectors and the eigenvalues, a higher-dimensional data can be compressed into a lower-dimensional dataset. The package we use for dimension reduction is imported from sklearn listed in code 3.1.2

Listing 3.1: sklearn.decomposition.PCA Code

```
import numpy as np
from sklearn.decomposition import PCA
pca = PCA()
Compressd_Data = pca.fit(Original_Data)
```

Normalisation: Data is normalised before feeding into neural network. Normalisation is a statistic method that adjusting data measured from different scales (here, it refers that receiving data from different smartphone, different situations.etc) to a notionally common scale [27]. The mathematics is shown in equation 3.1

$$X_{new} = \frac{x - \mu}{\sigma} \tag{3.1}$$

We use the function of MinMaxScaler from sklearn to normalise sensor data. The code is listed in code 3.1.2

Listing 3.2: sklearn.decomposition.PCA Code

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
Normalised_Data = scaler.fit(data)
```

3.1.3 Recurrent Neural Network

Recurrent neural network is one of the artificial neural network that includes connections between nodes which flow along a sequence. This structure offers the RNN model has the ability to exhibit temporal information through a time serial data.

The recurrent neural network has proven its advantages in dealing with sequential data such as speech recognition, image captions and machine translation tasks. RNN is similar to feedforward neural network. The difference is that the recurrent connections link a neuron from the current layer to the next neuron of the next layer. This feature makes the RNN model 'remember' the features from previous loop [28]. RNN transfers the state within each loop, therefore, it could deal with sequential data such as the sensor data we used in the experiment.

The RNN (Fully Recurrent) structure is shown in Figure 3.2. It is an unfolded

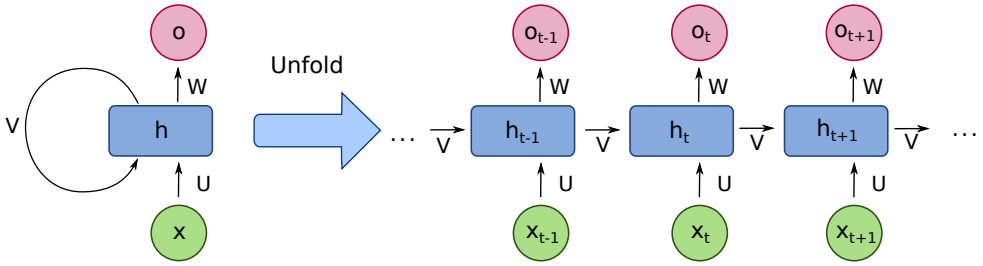


Figure 3.2: RNN

basic RNN structure which contains a number of neuron-like nodes. Those connected nodes which are either input, output or hidden nodes organise into successive layers which follows a one-way direction connected to the next layer. Each neuron includes an activation that varied based on the time sequence. Errors are calculated from each sequence while the total error is the total value of the deviations of the target label values calculated from each sequence.

However, SimpleRNN has the problem of vanishing gradient when feeding a long sequential data. It cannot catch the feature of the dependencies between samples in a relatively longer sequential data. Hence, Long short-term memory (LSTM) is given to solve this problem.

LSTM is an optimised RNN model that solves the basic RNN problem of vanishing gradient. This is achieved by adding a forget gate. It prevents vanishing or exploding caused by back propagated errors. Figure 3.3 shows the internal structure of the LSTM unit. In each unit, there are not only input, output gates but also a forget gate that

controls the 'memory' to either propagate into the next layer or being forgotten in the current layer [29].

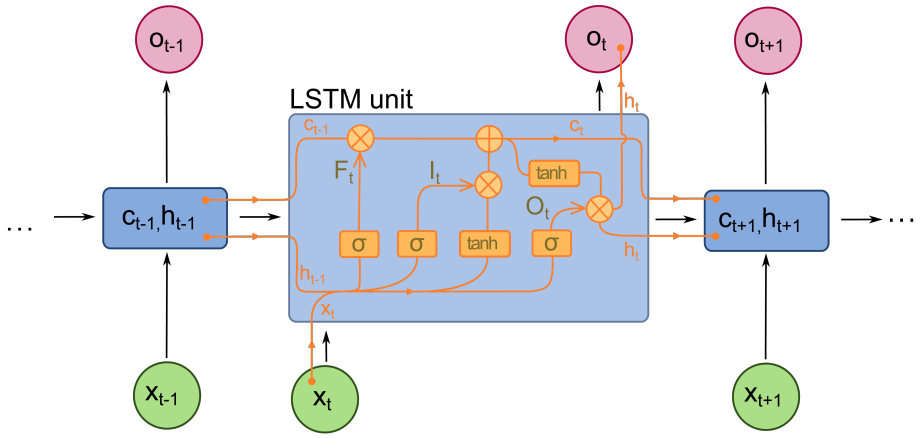


Figure 3.3: Long short-term memory (LSTM)

The value in the current state is controlled by the forget gate f gate signal. Specifically, save the value when the signal is set as *1* while forget the value if gate is set as *0*. The activation of receiving a new input or propagating are determined by its input gate and output gate respectively [7]. The equations 3.2 to 3.7 show the numerical definition. The softmax output of m_t determined the final probability distribution while $\odot$ is the product of the cell value of the gate value.

$$i_t = \sigma(W_{ix}x_t + W_{im}m_{t-1}) \tag{3.2}$$

$$f_t = \sigma(W_{fx}x_t + W_{fm}m_{t-1}) \tag{3.3}$$

$$o_t = \sigma(W_{ox}x_t + W_{om}m_{t-1}) \tag{3.4}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot h(W_{cx}x_t + W_{cm}m_{t-1}) \tag{3.5}$$

$$m_t = o_t \odot c_t \tag{3.6}$$

$$p_{t+1} = \text{Softmax}(m_t) \tag{3.7}$$

As the sensor data is presented in time sequence, the LSTM model is ideal for location estimations. As mentioned in the data acquisition section, the data will be fed via samples to the LSTM model. Here, the size of one sample is time step * feature. The feature of each data point is the three total values of acceleration and gyroscope and magnetic field data. The number of the data point in each sample depends on the time window. Each sample is matched with a coordinate of (X_i, Y_i). The output of the LSTM model is the estimated location coordinate of (X_{est}, Y_{est}) when feeding a set of

new real-time sensor data. The numerical definition is shown in equation 3.8 to 3.10.

$$x_{-1} = \text{Sensor Data}(I) \tag{3.8}$$

$$x_t = W_e S_t, \quad t \in \{0...N-1\} \tag{3.9}$$

$$p_{t+1} = \text{LSTM}(x_t), \quad t \in \{0...N-1\} \tag{3.10}$$

LSTM updates its weight and state by each time reading a new incoming sensor data and returns a location estimation result. In this experiment, we would build a regression LSTM model as the prediction is the numerical coordinates of latitude and longitude.

3.1.4 Stateful/ Stateless LSTM

In the machine learning framework such as Tensorflow, PyTorch and Keras, there are mainly two types of LSTM: stateful and stateless LSTM. Theoretically, Two LSTM models are the same regarding the structure and mathematical definitions as explained before. However, the difference between stateful and stateless LSTM is about the 'state' [30]. As the data is fed in batch through the training process, the model learns the feature by batch and batch until batch covers all the samples.

Stateless LSTM: LSTM updates its parameters on the first batch and then initiates its weight and state on the second batch. The hidden states and the cell states which contain memory history would be reset as zero for the next batch. Therefore, parameters are isolated between different batch or 'state' which means stateless.

Stateful LSTM: LSTM continues to use the parameters settings for the next coming batch after it has updated its parameters on the first batch. This means that the hidden and cell states are shared between different batch and emphasis the dependencies between batch and batch known as stateful.

For example, in the natural language processing tasks, regarding understanding an article and a set of poems, we could consider those two tasks are in different situations. An article includes certain sentences and those sentences are related between each other as the context effect. Poems are isolated between different poems. There are no connections external to the poem. Therefore, when data between batches are related to each other (e.g. a lengthy sequential input of a whole article), stateful LSTM supposes to perform better while stateless LSTM could handle the shorter length of the sequential data of a set of poems. Comparably, the sensor movement data is collected in time series which means it has dependencies from the previous data sampling to the next sampling. In another word, it is impossible for a person to walk from one place to

another place instantly.

In this experiment, depends on the size (or length) of the training data, we would deploy both stateful and stateless LSTM models for location tracking.

3.1.5 Evaluation Methods

We would use a statistical evaluation of the cumulative distribution function (CDF, also cumulative density function) for evaluating the model's performance [31]. CDF includes the estimation error from zero to the maximum estimation error. The accuracy in each error precision is calculated using equation 3.11.

$$\text{Accuracy} = \frac{\text{Estimation inside the precision range}}{\text{Total number of the estimation}} \tag{3.11}$$

Meanwhile, besides using CDF to compare models' performances, we would also visualise the prediction pathway upon the geographical map as each pixel of the map has the location information of latitude and longitude.

3.2 Experiment Preparation

3.2.1 Experiment Environment

Informatics Forum of the University of Edinburgh is selected as the experiment environment as a result of its complex indoor structure with sufficient signal coverage. The magnetic field distribution is identifiable and could be used for indoor location tracking. The indoor position is easy to locate (label to the sensor data) with the symbol of the lifts, stairs, opening area and offices with room numbers.

In this experiment, we selected the route on the second floor of the Informatics Forum. The whole route is in the middle of the path shown in figure 3.4. Starting from the right side of the lift following the anti-clockwise direction and finally back to the starting point.

3.2.2 Experiment App and Devices

App: MapLoc Android App is used for data collection. It could record the built-in smartphone sensors raw data into a log file in XML format. This function is achieved by calling the Android API which monitors the sensors movement information with different monitoring frequency. The type of sensors used for positioning includes the

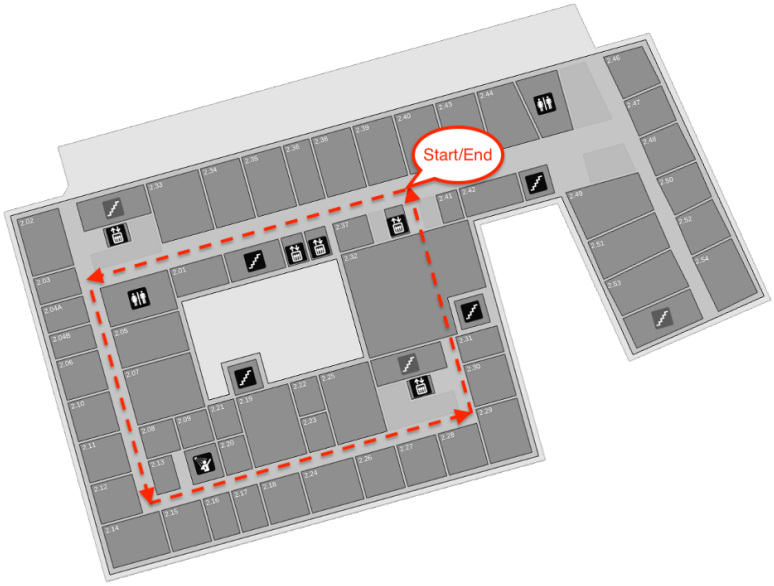
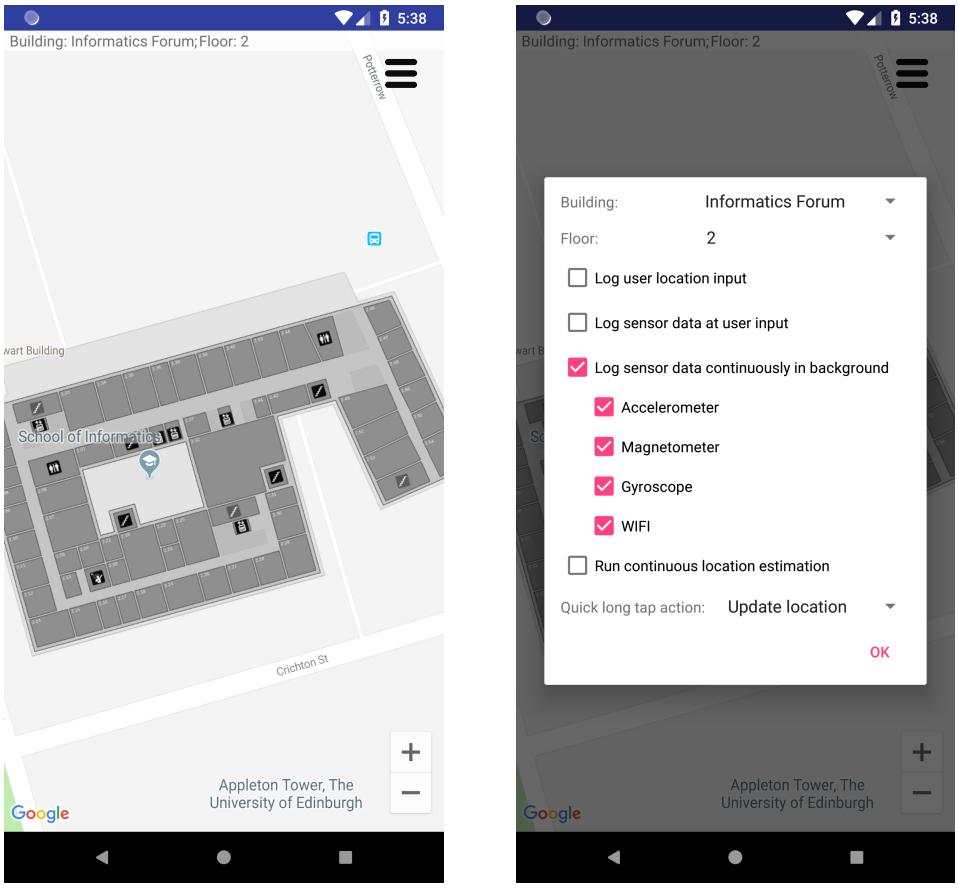


Figure 3.4: Route in the second floor of Informatics Forum

accelerator, gyroscope, magnetometer and WiFi module. The app records the X, Y, Z values of sensors with both system timestamp and built-in sensor timestamp. WiFi data includes the current timestamp which monitors the WiFi received signal strength from certain WiFi routers, represented as certain MAC addresses with following the signal strength in dBm and the channel from each MAC address. Each data point is logged starting with the name of the sensor, the timestamp of when the data is logged and the sensor data. Here, it is noticed that the monitoring frequency is not in constant time gaps, which means the time difference between the two sampling of the sensor varies. It is limited by the Android API that only if the system monitors the changes of the sensors position, a new data point would be saved into the log file. Besides, by accessing the Google Map API, the location information which represented by latitude and longitude could be saved simultaneously with the sensor data when clicking the current location on the map showing on the mobile screen.



(a) Homepage Screenshot (b) Options Screenshot

Figure 3.5: MapLoc Screenshot

Devices: OnePlus 6 and Smartisan U2 are used for data collection. OnePlus is a flagship phone with high sensitivity sensors while Smartisan is an entry-level phone with less sensitive sensors. MapLoc is installed in both devices with the same settings of the monitoring frequency.

3.2.3 Develop Environment

The LSTM based indoor locationing tracking system is developed in Python 3.6. The packages used for data processing and machine learning are TensorFlow, Scikit-learn and Keras.

TensorFlow™ is an open source library for machine learning jobs provided by Google. Initially developed by Google Brain team within Googles AI organisation. It contains support for data mining and deep learning algorithms with flexible numerical computation in many scientific domains.

Scikit-learn is a simple and efficient data mining and data analysis tools running in Python. It is an open source commercially usable tools built on NumPy, SciPy, and matplotlib.

Keras is a high-level neural networks application programming interface (API) running based on top of TensorFlow, CNTK and Theano. Here, we use Keras with the backend of Tensorflow while the models are built on the Keras sequential model.

(a) TensorFlow (b) Scikit-learn (c) Keras

Figure 3.6: Open Source Library

Chapter 4

Data Processes

4.1 Data Collection

4.1.1 Data Gathering

Standing at the starting point of the route, click the start button in the MacLoc App and put the phone in the right pants pocket. In the same time, use a timer to record when the App starts to generate a log file. Keep the phone stable in the pocket to avoid it shaking when the person collecting the data are walking to avoid random noise. After clicking the start button, wait for three seconds to let the sensors finish initialisation. Start walking and record the time. Waking at a constant speed with the same step distance to collect the data while using a secondary phone to capture participant input labels (location points). Tap the screen to log the location information when passing by special places such as lifts, stairs, kitchen and corners. Record the time when back to the starting point and stop walking. Finally quit the App to stop writing the log file. Time records are used to clip the raw file which only includes walking data with no unnecessary data logged in the file such as operating and preparation behaviour. Repeat those steps to collect multiple rounds of the data.

Listing 4.1: log file

```
<data phone="6169b50559d0c09" building="undefined" t="2018-06-21 13:45:55.479">
<m t="1529585155482" x="-12.648" y="-1.9653" z="-161.131" st="1710227306959" />
<a t="1529585155491" x="0.96646" y="2.56660" z="9.1324" st="1710242840406" />
<g t="1529585155572" x="-0.089385" y="-0.082061" z="-0.0577" st="1710322888014" />
</data>
```

4.1.2 Variations

There are some factors which brings variations during gathering the sensor data.

Magnetic Field Distribution: Different standards of the sensors affect the accuracy of the data in terms of the sampling density and precision. Meanwhile, during the collection process, magnetic field data is mainly influenced by the environment. Despite the fact that the rotation of the earth changes the density of the earth magnetic field in daily time and seasons, indoor moving items affect the distribution and the density of the indoor magnetic field data more obviously. Sensed magnetic field data increases when two electronics items come closer to each other. Meanwhile, the moving lift changes the distribution of the magnetic field. During the weekday and weekends, the indoor magnetic field distribution changed mainly by peoples behaviour. To eliminate this influence, collecting the data on both weekdays and weekends in a different time to make the training data include most of the situations.

Walking speed and step distance: Consider increasing the robustness of the model. It is better to contain the sensor data collected in different situations. Here, we obtain the data in different walking speed with different step distance. In general, we collect the data in three cases: fast walking with large step distance, slow walking with short step distance, dynamic speed and step distance walking. Those three situations can cover many different people's walking behaviours. In this project, we collect 12 rounds of sensor data in total from 2.5 minutes; 3 minutes, 3.5 minutes to 4 minutes with three rounds of each time span.

4.2 Data Pretreatment

4.2.1 Data Transformation

In this section, we explain how the data is transformed from the log history into sensor training data (features). The raw file includes sensors movement information in different timestamps. Due to the delay of the system timestamp compared to the sensor built-in instant timestamp, we use sensor timestamp as the time standard. Meanwhile, sensors timestamp is not starting from 0. We use the relative time (millisecond) to replace the timestamp. Specifically, get the difference between each sensor timestamp with the initially logged sensor timestamp. Each data point is represented in milliseconds with the first log history starting from time zero.

Clip the raw sensor data based on the timer record with the start and end walking

time. Therefore, the raw sensor data only contains the walking information.

In order to eliminate the difference of the smartphone position in each rounds when collecting the data, we calculate the total value of the accelerator, gyroscope, magnetometer.

$$Sensor_{total} = \sqrt{Sensor_x^2 + Sensor_y^2 + Sensor_z^2} \tag{4.1}$$

Hence, the sensor data include discontinues time in millisecond with three total values shown in Table 4.1.

ST	AccTotal	GyrTotal	MagTotal
0ms	9.602782476110509	0.34956840840168957	147.6066924925367
1ms	9.600623329987638	0.34944948195427106	147.5879780562939
4ms	9.594145891619025	0.3490927026120156	147.5318347475655

Table 4.1: Discontinues data

To generate whole sensor data in every millisecond, we use interpolation to fill the empty data in between each sampling. For example, the first two sampling of the accelerator appears at 0, 1 ms and 4 ms with sensor data. Insert two empty datapoint that represent 2ms and 3 ms. Use linear interpolation to fill those three datapoint as giving the both side of the value (1 ms and 4 ms). So far, we have the sensor data in every millisecond exported from the raw log file shown in Table 4.2.

ST	AccTotal	GyrTotal	MagTotal
0ms	9.602782476110509	0.34956840840168957	147.6066924925367
1ms	9.600623329987638	0.34944948195427106	147.5879780562939
2ms	**9.598464183864767**	**0.34933055550685255**	**147.5692636200511**
3ms	**9.596305037741896**	**0.34921162905943411**	**147.5505491838083**
4ms	9.594145891619025	0.3490927026120156	147.5318347475655

Table 4.2: Data with interpolation

To create the training file, we need to add labels to each datapoint which match the locations with sensor data based on the time of tapping on the map during the data collection process.

4.2.2 Label Matching

During the data collection procedure, certain special points have been marked down with the time. Meanwhile, we collect the data at a relatively constant speed. Therefore, using linear interpolation could fill out the missing labels to the sensor data as the label are represented by the numerical values of latitude and longitude. In particular, firstly match the marked location information (label) to the sensor data at the same time. Secondly, interpolate value in between two marked locations. Here, the total number of labels depends on the time window we used in data processing. For example, if the time window is set to cover ten data points, there will be ten data points as 1 sample with a location label of latitude and longitude.

4.3 Data Processing

The recurrent neural network is used to build the location tracking model. Therefore, the training data needs to be reshaped to a three-dimensional array of (Samples, Time Step, Features). To be noticed, time window equals to time step without data compression. Here, the data point is sensor scanning with the location label information in each millisecond. Samples represent the total training samples in the training set where each sample includes the number of time step datapoint. Time step is related to the time window that covers certain datapoint. Features include the accelerator, gyroscope, magnetometer x, y, z and total value. In this experiment, the time window is set as 10ms, 100ms, 1000ms and 2000ms.

4.3.1 Data without overlapping

Depends on the time window setting, the data is reshaped as (Samples, Time Step, Features). Figure 4.1 shows the detail of the label matching without overlapping values. Specifically, if there are 10k datapoint in the original sensor data file while we set the time window as 1000ms and consider using the three total values of accelerator, gyroscope, magnetometer as the feature input. The data would be reshaped as (10, 1000, 3). Meanwhile, each sample (here we have ten samples in total) is followed by the location label which is matched based on the time record.

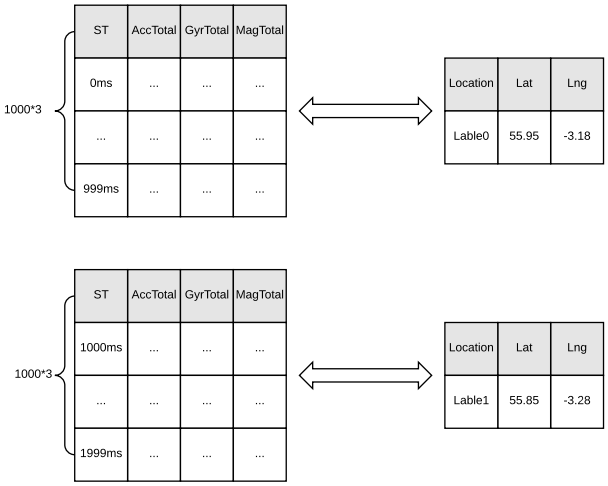


Figure 4.1: Data without overlapping

4.3.2 Data with overlapping

Taking the advantages of RNN that it could remember the features from the previous input, the overlapping parameter allows each sample to include previous information that emphasises the dependencies between samples. In other words, the next sample contains part of the data point in the previous sample. For example, if there are 10k data points and three feature inputs, set the time window as 1000ms and the overlapping as 900ms. The first sample includes the datapoint from 1 to 1000. The second sample includes the datapoint from 100 to 1100 shown in Figure 4.2.

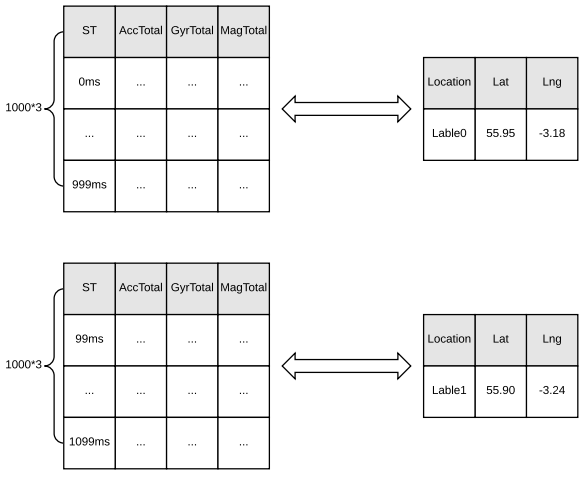


Figure 4.2: Data with overlapping

The reshaped training data will be as (90, 1000, 3). As the data has been overlapped

of 90% (repeated by nine times), therefore, the sample increases nine times from 10 to 90. Meanwhile, 1000 is the number of data points in one sample which represents the 1000 milliseconds. 3 is the input feature of three total sensor values.

4.3.3 Down-sampling Data

To improve feed-forward speed with a smaller input size without losing too much information, a down-sampling method is used to compress the data. For example, set a pickup value as 100ms which means we pick up data points from the original sensor data every 100ms. If there are 10k datapoint, the down-sampled training set only includes ten data points but with the same time cover range illustrated in Figure 4.3.

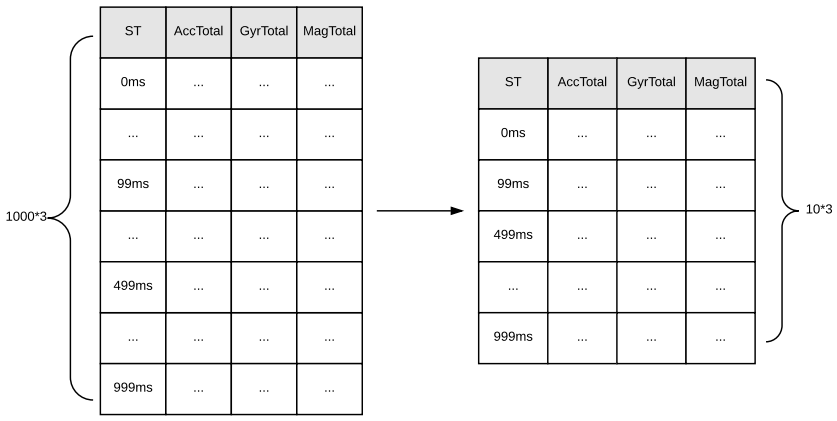
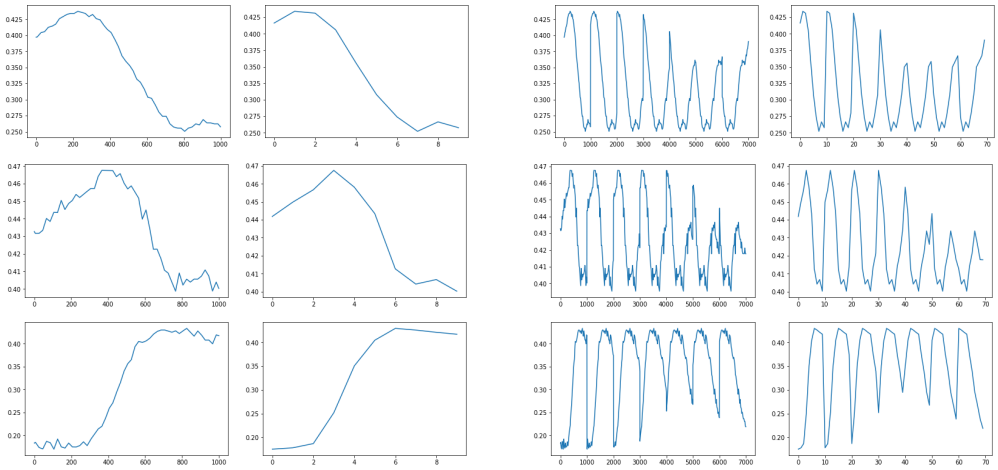


Figure 4.3: Data down sampling

Figure 4.4(a) shows three total sensor values of the accelerator, gyroscope, magnetometer of 1000 data point (1000ms) on the left side and the down-sampled total value of 10 data points on the right side in both 1000ms. Figure 4.4(b) compares the original data and the down-sampled in 7000ms.



(a) Sensor values in the range of 1000ms (b) Sensor values in the range of 7000ms

Figure 4.4: Compare between the original data and the down-sampled data. It shows that loss of information is minimal across the two time windows, being able to follow the trend in signal for the walking activity.

The comparison from the figure 4.4 illustrates that when minimising the data size, the down sampled data keeps the data features without significant information loss. Therefore, with the down-sampled data, there would be only 10 datapoint of the training data which covers the original 10K ms datapoint with the pick up value of every 100ms while the training data would be reshaped as (10, 10, 3).

4.3.4 PCA Data

Considering an automatic method to compress the training data instead of artificially selecting the pickup value such as downsampling processing, we implement PCA dimension reduction by giving the output dimension size. The original data can be compressed into a lower-dimensional matrix. To keep the output size the same as the downsampling for later comparison, we set the PCA output as 10. Due to the requirement of PCA from sklearn, we need to firstly reshape the original data horizontally, which means that each sample includes 3*1000 features. Three represents that there are three types of the sensors while 1000 represents that the time window for one sample covers 1000ms of the measurement. Figure 4.5 shows the workflow of PCA. On the left-hand side is the reshaped 2D original data with overlapping of 90%. By implementing PCA, the 3*1000 features is compressed to 3*10 features of each sample.

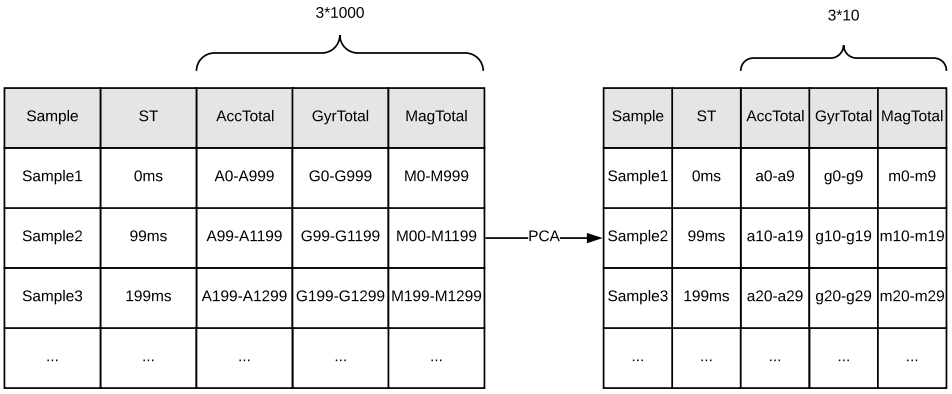


Figure 4.5: Data down sampling

Chapter 5

Model Selection and Evaluation

The machine learning model for location tracking develops on a recurrent neural network called Long-Short-Term-Memory (LSTM). We start to customise the model in two stages. The first stage is to determine the baseline model which includes finding the model hyperparameters setting and the structure such as time step. After determining the structure of the LSTM model with keeping the hyperparameters the same, we move to find out a balance between learning efficiency and location estimation accuracy by adjusting the data input such as adding overlapping or downsampling as well as PCA in the second stage.

5.1 Baseline Model

5.1.1 Time Window

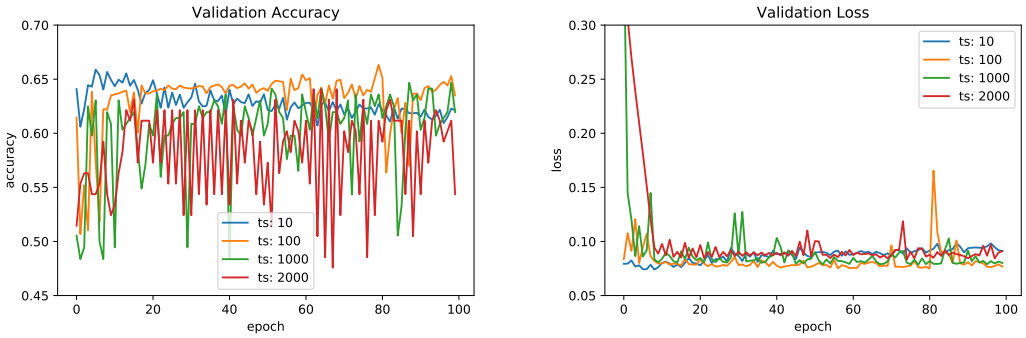
As LSTM read the data by time window, a proper time window setting could make the model catch the detail information within each time window while has an overview of the features among the whole dataset. In other words, a narrow time window setting could catch the detail information but lose the general features of the training set. In contrast, a wide time window could cover the overall feature but miss the detail information be computationally expensive and slow to react on changes. A proper time window setting which captures sufficient variations allows model identify different activities such as walking at different speeds or climbing stairs.

Here, we trained the model with the time window of 10ms, 100ms, 1000ms and 2000ms with the model setting listed in table 5.1.

Parameter	Settings
Epoch	100
Batch Size	100
LSTM Hidden Units	128
LSTM Layer	1 Layer
Learning Rate	0.005
Learning Rules	RMSprop
Training Data	One Round

Table 5.1: Baseline Parameters

According to figure 5.1, it shows that based on one round of the training data, 10ms model performs best with the validation accuracy of aapproximately 64%. 100ms model has a slightly lower accuracy at approximately 60%. 1000ms and 2000ms models perform relatively poorer than the narrow time window models. Specially, both of them have the validation accuracy around 56%. In general, the differences between various time window setting does not affect the accuracy significantly. From figure 5.1(b), we could find that all four models have a similar validation loss at about 0.08.

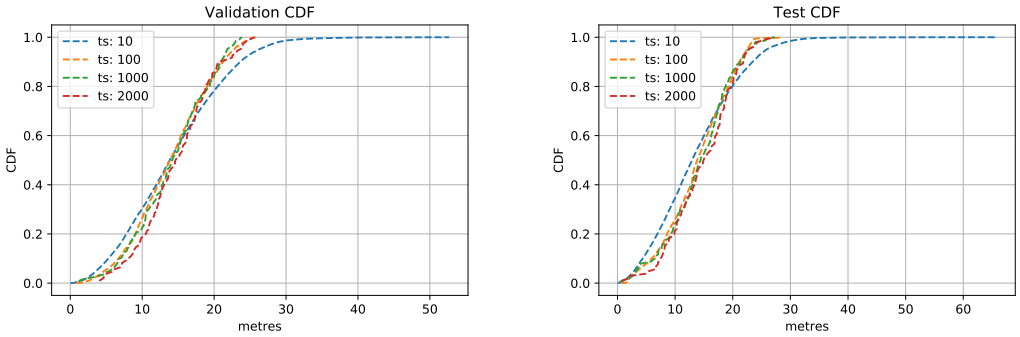


(a) Validation Accuracy (b) Validation Loss

Figure 5.1: Model performance with different time windows

We use three rounds data as the validation set and the one round data as the test set to plot the CDF. According to figure5.2, all models perform similarly with the estimation accuracy of 80% at the precision of 20 metres. In terms of the testing set CDF, they perform similarly as well. However, according to the x axis of the test CDF, 10ms based model has the maximum prediction error of over 60 metres (showing on the right end of the x axis). In general, 1000ms based model shows an

acceptable estimation accuracy. On the other hand, selecting a large time window of 1000ms allows the model capture variations and different activities for further function expansion. Therefore, in the later research, we aim to improve model's efficiency and locationing accuracy based on the model with 1000ms time window setting.



(a) Validation CDF

(b) Test CDF

Figure 5.2: CDF with different time windows

5.1.2 Input Features

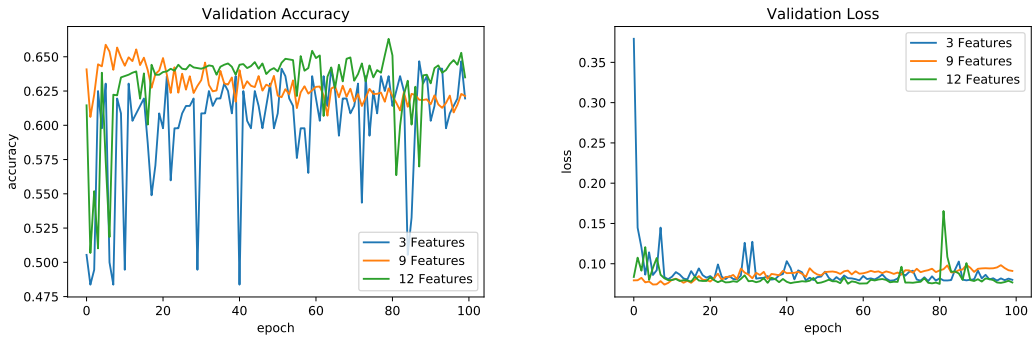
After deciding the time window, we move to select the input features of the model. As mentioned in the data collection section, MapLoc App provides x, y, z values of accelerator, gyroscope and magnetometers. Meanwhile, we could calculate the total values from each sensor's three-dimensional measurement by the square root explained in equation 4.1.

We compared the models with 3, 9 and 12 features input listed in table 5.2.

3 Features	(Acc_{total}, Gyro_{total}, Mag_{total})
9 Features	(Acc_x, Acc_y, Acc_y), (Gyr_x, Gyr_y, Gyr_y), (Mag_x, Mag_y, Mag_y)
12 Features	3 * (Sensor_x, Sensor_y, Sensor_y) + (Acc_{total}, Gyro_{total}, Mag_{total})

Table 5.2: Input Feature

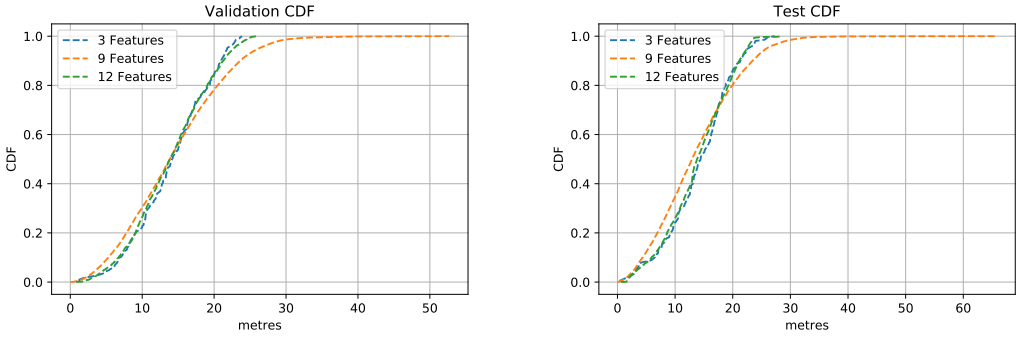
From figure 5.3, all models with 3, 9 and 12 features input have similar validation accuracy at approximately 62.5%. In terms of the validation loss, 3 features performs relatively stable which its loss remains around 0.005.



(a) Validation Accuracy (b) Validation Loss

Figure 5.3: Model performance with different numbers of input features

According to the figure 5.4, both models with 3 and 9 features input reach the validation accuracy about 80% with the error of 18 metres. Model with 12 features input reach the same accuracy with a slightly larger error at 20 metres.



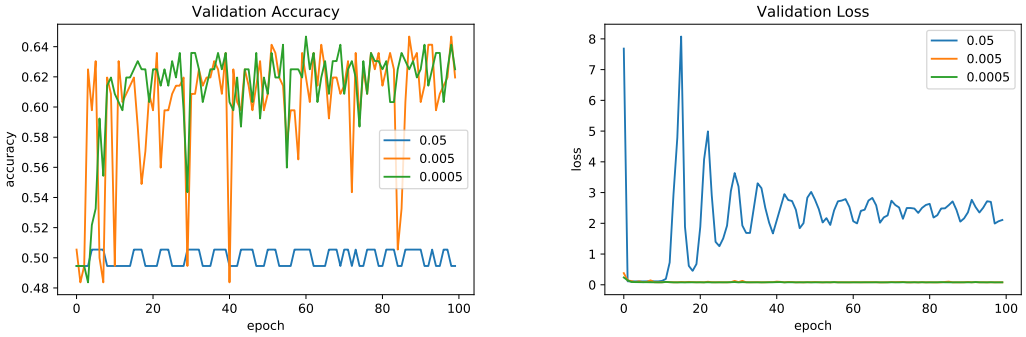
(a) Validation CDF (b) Test CDF

Figure 5.4: CDF with different numbers of input features

From the comparison, model with 3 total value input shows the best performance. Meanwhile, when collecting the data for either building the training data library or the real-time user's data, smartphone's position affects the three individual measurement. However, the total value would not be affected by the position. Therefore, using the three total values as the input features could eliminate the influence of the smartphone position.

5.1.3 Learning Rate

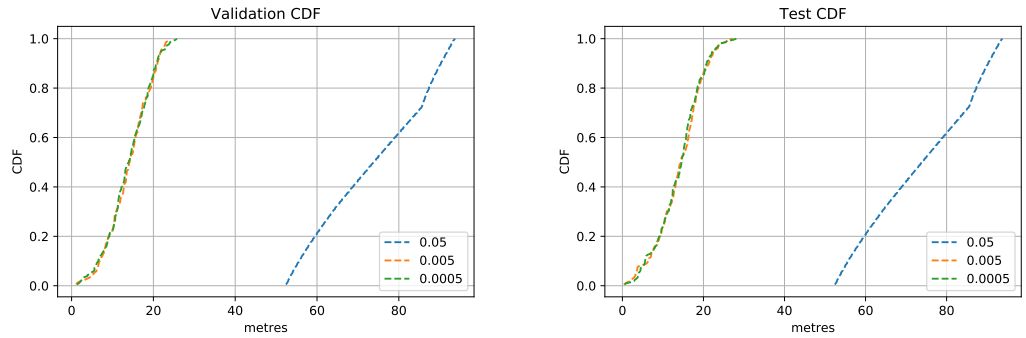
We explored model's performance based on three learning rate settings of 0.05, 0,005 and 0.0005 while kept the other settings same. Settings are listed in table 5.1.



(a) Validation Accuracy (b) Validation Loss

Figure 5.5: Model performance with different learning rates

According to figure 5.5(a), both models with 0.005 and 0.0005 learning rate have an acceptable validation accuracy around 62% after 100 epoch while the 0.05 learning rate based model remains approximately only 50% of the accuracy. Meanwhile, figure 5.5(b) illustrates that the 0.005 and 0.0005 learning rate models have similar performance with both of the validation loss reaches 0.2. However, 0.05 based model has an unstable performance which its loss fluctuates between 0.3 and 8.



(a) Validation CDF (b) Test CDF

Figure 5.6: CDF with different learning rates

From both validation and test set CDF according to figure 5.6(a) and 5.6(b), 0.005 and 0.0005 models have similar prediction accuracy. Specifically, with the precision

of 20 metres, both of them reach the prediction accuracy around 85% while the 0.05 learning rate model minimum prediction error starts from around 46 metres. Therefore, consider of the time spend of training the model, we determine to use learning rate as 0.005 for the following model.

5.1.4 Learning Rules

In terms of optimiser, we explore model's performance based on Stochastic gradient descent (SGD) and RMSProp with the learning rate of 0.005 and train the model based on one round of the training data.

Gradient Descent: The size of the training dataset affects the learning efficiency of the basic gradient descent as it requires prediction results by calculating each of the training datapoints. SGD optimised its efficiency by sarcastically updating the model weight from each training instance [32]. SGD has a faster learning efficiency when dealing with a large dataset. The numerical explain is shown in equation 5.1.4.

$$\Delta w_i(t) = -\eta g_i(t) \tag{5.1}$$

RMSProp: RMSProp (Root Mean Square Propagation) is an optimised learning rule based on SGD [33]. It brings an adapted learning rate during the model training. The equation is given in 5.1.4.

$$S_i(t) = \beta S_i(t-1) + (1-\beta) g_i(t)^2, \tag{5.2}$$

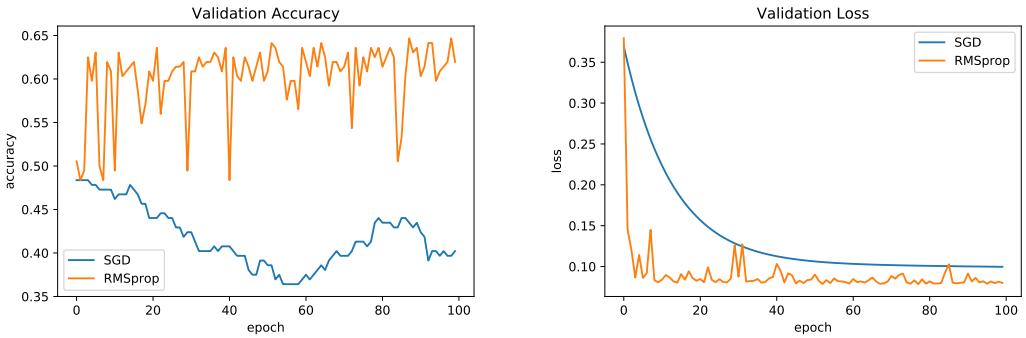
$$\Delta w_i(t) = \frac{-\eta}{S_i(t)+\varepsilon} g_i(t), \tag{5.3}$$

$$w_i(t) = w_i(t-1) + \Delta w_i(t), \tag{5.4}$$

where S is the adjusted sum of the previous parameters while β (usually set as 0.9) is attenuated by the decay rate . The ε is a relatively small constant value to prevent the divider comes to 0.

Figure 5.7 shows the performance with different learning rules of SGD and RMSProp. It is obvious that RMSprop based model performs better than the SGD based model. In particular, according to the figure 5.7(a), RMSProp based model has a validation accuracy around 60% despite there are some fluctuations through the training process while the validation accuracy of the SGD based model starts decreasing from 47% to lowest 37%. From figure 5.7(b), we could find that RMSprop based model

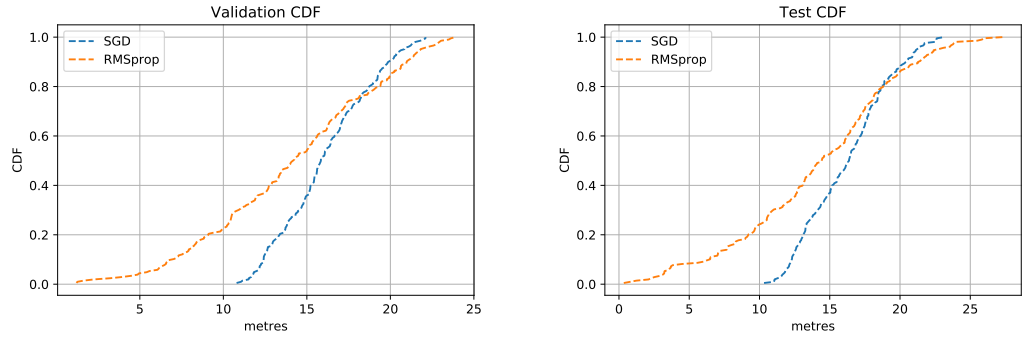
converge faster as it reaches 0.1 loss after three epoch while the SGD based model has not converged to the validation loss of 0.1 after 100 epoch.



(a) Validation Accuracy (b) Validation Loss

Figure 5.7: Model performance with different learning rules

Figure 5.8 shows the model prediction CDF of both validation set and test test. With the precision of 15 metres, according to the figure 5.8(a), RMSprop based model has the accuracy of 47% while the SGD based model has an accuracy of 28%. The test cdf shown in figure 5.8(b) has the same trend.



(a) Validation CDF (b) Test CDF

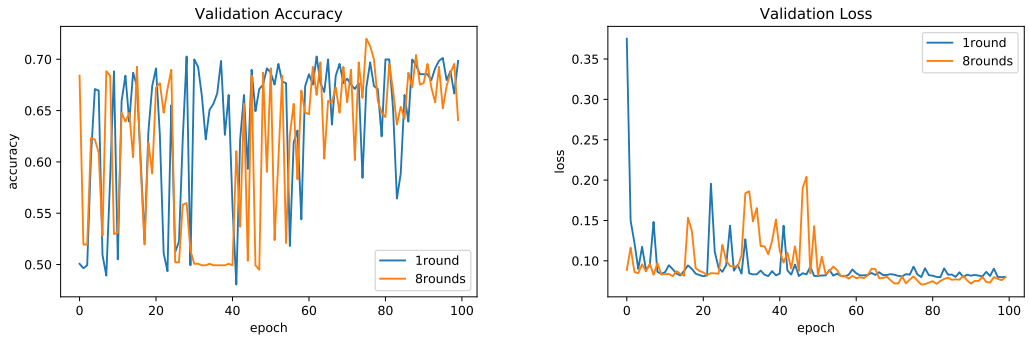
Figure 5.8: CDF with different learning rules

It is interesting that the cross point that two models reach the same prediction accuracy when the precision is set as around 18 metres. SGD based model has relatively higher accuracy than the RMSprop based model. However, to reach a higher accuracy performance with a higher precision requirement, RMSprop based model shows its advantage of faster convergence and higher prediction accuracy. Therefore, we select RMSprop as the optimizer of the model for later model training.

5.1.5 Training Data Size

Recall the model performance in different time window settings. 10ms based model has the best performance compared to the model with 100ms, 1000ms and 2000ms time window. One of the reason is that when reshaping the training dataset into the RNN required shape (Samples, Time Step, Features), if set the time step (same as the time window of 10ms here) as 10, which means the total number of the samples is the total datapoint divided by the time window (10ms). Compare to the time window of 1000ms, which divides the total sample by 1000, the samples that are offering for 10ms based model is 100 times larger than the samples of 1000ms based model if given the same size of the training data. In another word, one of the reasons that 10ms based model performs better than the 1000ms based model is because the 10ms based model has sufficient training data to learn the positioning features. Therefore, in order to find out the influence of the training size, we compare the model by giving only one round of the data, and eight rounds of the data with other parameters keep the same as shown in table 5.1.

According to figure 5.9, the same model with different input data size performs similar despite the data increases eight times. From the figure 5.9(a), both of the validation accuracies fluctuate between 50% and 70% while the validation loss from figure 5.9(b) also proves that both versions have similar performance with the validation loss stays around 0.05 despite that the model with 1 round input data has a large initial loss around 0.35.

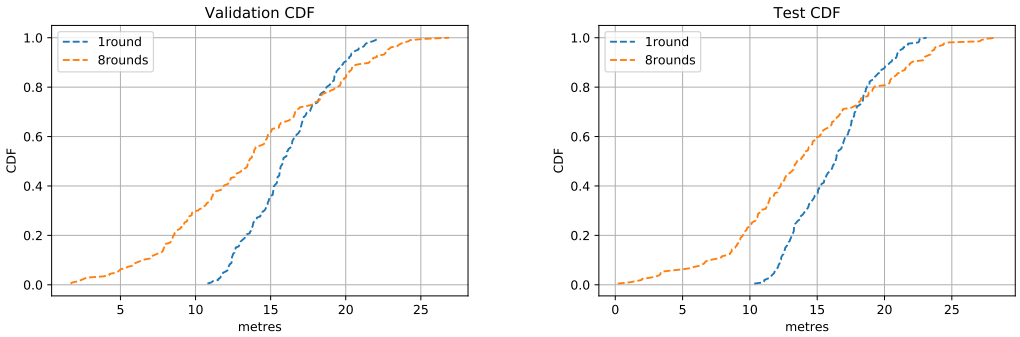


(a) Validation Accuracy (b) Validation Loss

Figure 5.9: Model performance with different training data size

From figure 5.10, it is interesting that 1 round input model performs better than the 8 round version. This possibly happens because of the validation set and the test set

extract similar features from the data that feed into the 1 round model. In general, both versions have a roughly 70% accuracy when the precision is required as 16 metres.



(a) Validation CDF (b) Test CDF

Figure 5.10: CDF with different training data size

This comparison between different size of the training data input has not shown an obvious result of the influence of the input data size. However, in order to avoid the possibility that there is no sufficient dataset for training model, we increase the training dataset to include 8 rounds of the sensor data with different walking speeds and step sizes to allow models to learn more variations. Therefore, the baseline model is set as shown in table 5.3.

Parameter	Settings
Epoch	100
Batch Size	100
LSTM Hidden Units	128
LSTM Layer	1 Layer
Time Window	1000ms
Learning Rate	0.005
Learning Rules	RMSprop
Training Data	Eight Rounds

Table 5.3: Baseline Model Setting

5.2 Overlapping Model

Recall the overlapping diagram 4.2, by giving an overlapping rate (here, set as 30%, 50% and 90%) to the original sensor data. We could generate a new dataset in which each sample contains part of the sensor data from the previous sample. Figure 5.11 shows the procedure of generating the training data by an overlapping rate of 50%. Each sample contains half sensor data of the previous sample.

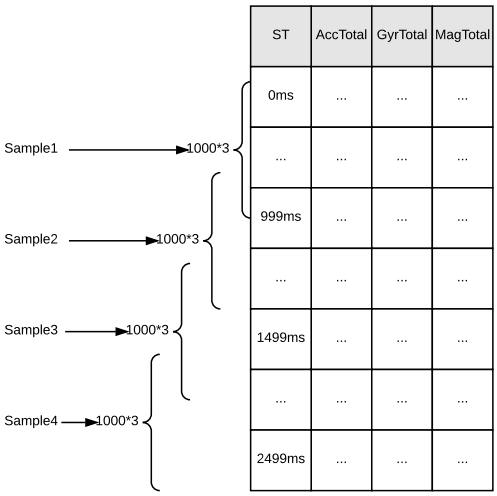
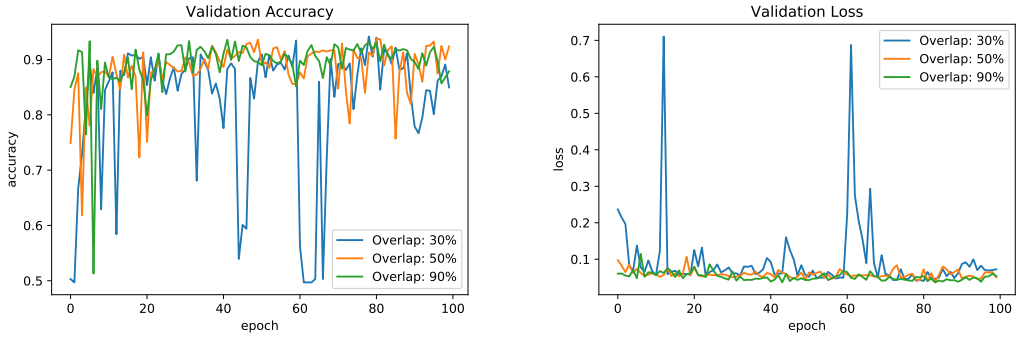


Figure 5.11: Overlap: 50 Percentage

There are mainly two reasons for implementing overlapping to the original data. The first reason is to emphasise the dependencies between neighbouring samples by including repeated information. This allows the recurrent neural network to remember more information from the previous sample to the current sample input. The second reason is to increase the training dataset as the model needs more samples to learn the required features when the time window is set larger than 1000ms. Therefore, we can increase the dataset not only by adding more rounds of the data but also by overlapping the sensor data.

Based on the baseline model settings listed in table 5.3, feed the training data of overlapping 30%, 50% and 90%. Figure 5.12 shows the three models' performance based on the validation set. From figure 5.12(a), we could find that 90% overlapping based model has the best performance reaching 90% of the validation accuracy after approximately six epoch despite there being a noticeable decrease at ten epoch, but it increases back to around 93% accuracy with the following iterations. Meanwhile, overlapping 50% based model shows an acceptable accuracy of 86% at 10th epoch and stays around 89% accuracy. Overlapping 30% based model has an unstable validation

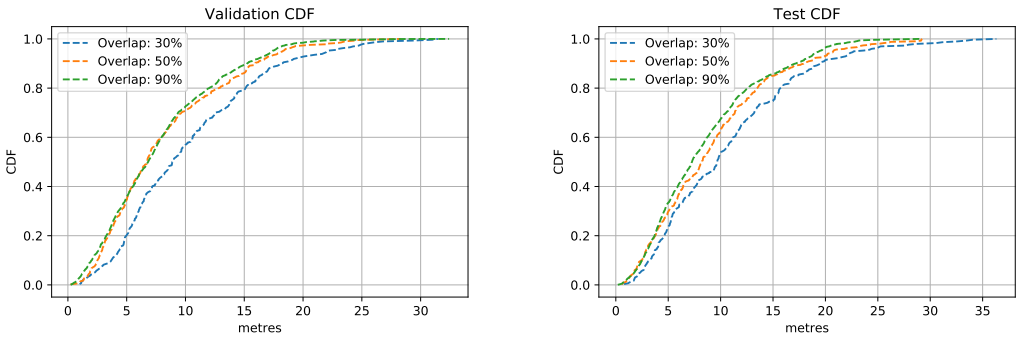
accuracy that fluctuates from 50% to 86%. This result illustrates that with the increasing the overlapping rate, model can learn features better. This is firstly because the training data size is increased and secondly each neighbouring sample contains more repeated data points which allows model to extract localisation features better. On the other hand, according to figure 5.12(b), both models with the overlapping rate of 50% and 90% have a relatively small validation loss about 0.008 while overlapping 30% based model's validation loss shifts between 0.2 to 0.7.



(a) Validation Accuracy (b) Validation Loss

Figure 5.12: Model performance with different overlapping rate

From the prediction CDF shown in figure 5.13, it is noticed that overlapping 90% based model performs slightly better than the 50% based model while 30% based model has a lower prediction accuracy. Based on the test CDF shown in figure 5.13(b), 90% based model reaches 82% of the prediction accuracy when the precision is 15 metres.



(a) Validation CDF (b) Test CDF

Figure 5.13: CDF with different training data overlapping rate

From the comparison results, by implementing overlapping strategy, model performance improves compared to the model with same hyper-parameters settings but feed with original data. Specifically, overlapping 90% of the training data has the best improvement to the model's prediction accuracy. Therefore, we would use 90% overlapped sensor data for the following model training. However, as the training data increased, the model training time increases as well. Hence, we need to balance between the time spent and the prediction accuracy as well as avoiding losing training data information.

5.3 Compressed-Data Models

Regarding down-sampling processing, recall from figure 4.3. By setting a pickup value, we could downsample the data without losing significant data information which is confirmed in figure 4.4. For the PCA data processing, by defining output size, new variables in a lower dimension could be calculated based on eigenvalues and eigenvectors and therefore replace the original variables in a higher dimension matrix. In order to compare the performance between downsampling and PCA with keeping the compressed data the same size. Therefore, we restrict the output of the PCA as 10*3 of each sample either.

Figure 5.15 on the left side represents the procedure of downsampling (pick up datapoint every 100ms) based on the training data with overlapping of 90%. Therefore, one sample has been downsampled from the size of (1000,3) to (10,3). On the right side, it shows the procedure of implementing PCA dimensionality reduction to compress training data from (1000,3) to (10,3) of each sample. It should be noted that both downsample and PCA are implemented based on the training data with 90% overlapping rate.

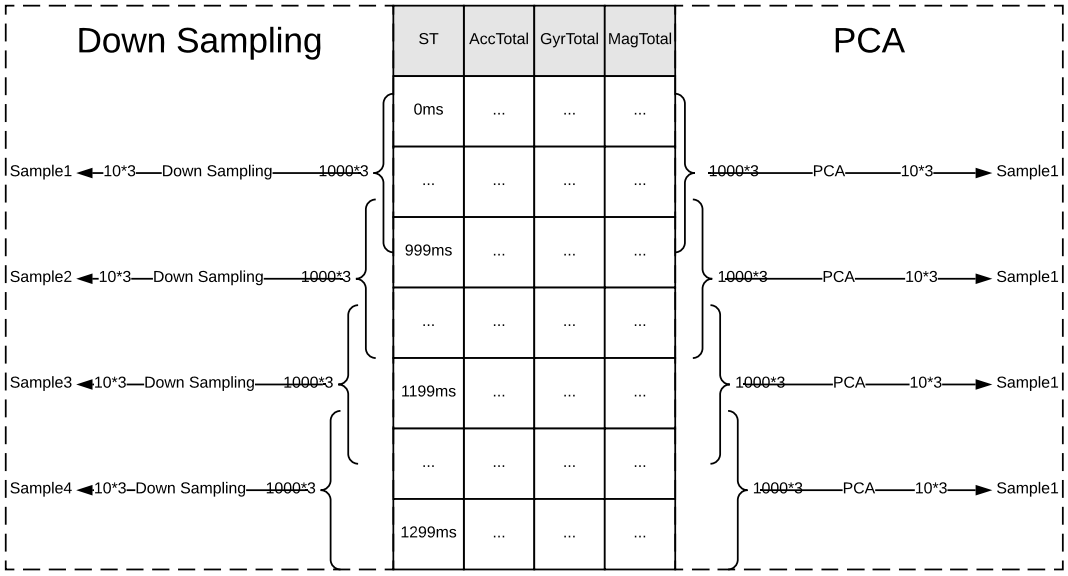
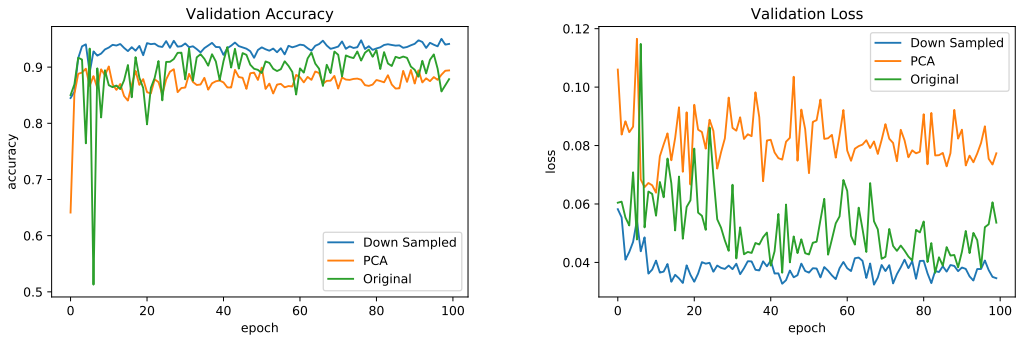


Figure 5.14: Down sampling based on the overlapping of 90 Percentage

By keeping the time window unchanged as 1000ms, a sample in a time window is compressed from 1000*3 to 10*3. In another word, setting the time step as ten could cover the data features of 1000ms from the original data. This avoids having LSTMs a heavy time step setting of 1000 which increases the model's complexity and learning time spent.

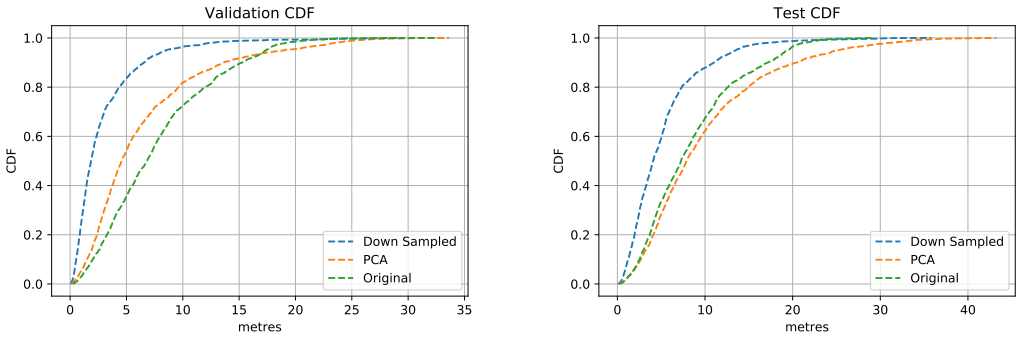
Figure 5.15 shows the comparison between the down-sampled based model and PCA based model with the overlapping 90% model. According to figure 5.15(a), the down-sampled based model has a faster convergence rate that reaching 95% of the validation accuracy at eighth epoch and then stays at high accuracy. PCA based model has a lower accuracy rate at around 87%. Original dataset based model is relatively unstable which has an obvious fluctuation between 78% to 92%. From figure 5.15(b), it is shown that the down-sampled based model converges rapidly from the validation loss of 0.06 to 0.005 within 20 epochs while the original data based model has an unstable validation loss that shifts from 0.2 to 0.04. PCA based model has a larger validation loss of 0.08.



(a) Validation Accuracy (b) Validation Loss

Figure 5.15: Model performance of input data down-sampled, PCA on input data and original data

Figure 5.16 shows the prediction CDF based on validation set and test set. Both figures illustrates that down-sampled based model performs better than the original data based model that it reaches 80% of the prediction accuracy with the precision requirement of 8 metres of the testing set. PCA based model performs similarly to the original based model in both validation and testing sets.



(a) Validation CDF (b) Test CDF

Figure 5.16: CDF of down-sampled, PCA and original data

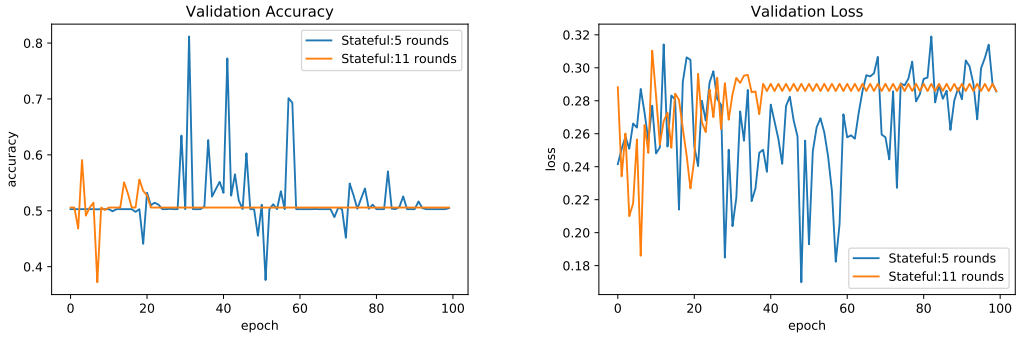
In general, the downsampled based model has the highest accuracy and reliability compared to PCA based and the original model. By implementing downsampling to the overlapping data, the model reduces its complexity by setting a smaller time step to save calculation time spend. It could handle the data with a larger time window without losing significant data features. Specifically, the model of the time step of 10 could cover the data with the time window of 1000ms. In other words, it allows the model to

further increase the time window allowance with little model complexity increased. It could make the model process more activities. Meanwhile, it allows the model to deal with larger training dataset with large overlapping rate and more rounds of data in a shorter time span compared with feeding the original data. This is of vital importance when integrating the model into mobile devices to improve the prediction efficiency which reduces application response time and also saves the power consumption.

5.4 Stateful LSTM Model

Besides the stateless LSTM models as mentioned above, we also build a stateful LSTM model in the experiment. As explained in the methodology section, stateful LSTM model could learn the dependencies between the data points in the current batch and those in the next neighbouring batch when feeding the training data in sequential order. According to the features of the sensor data, it is collected in time sequence while each of the measurement has connections between the previous and next data point. Therefore, we deploy the stateful LSTM model for location tracking.

Here, we start by training the model with only five rounds input data and then increase the dataset to 11 rounds of the data. Results are shown in figure 5.17. Both two versions perform unreliably that the validation accuracy is approximately 55% with the validation error is around 0.29.

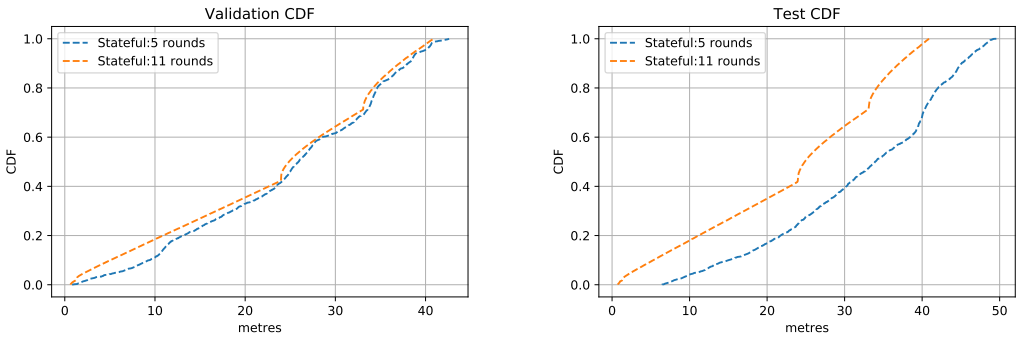


(a) Validation Accuracy (b) Validation Loss

Figure 5.17: Model performance of stateful LSTM with different rounds

Concerning the estimation performance which is shown in figure 5.18, both versions of the models perform worse than the downsampling based model (stateless). However, it is noticeable that there is a trend of the prediction accuracy increased when adding more rounds of training data.

One possible explanation for this situation is stateful LSTM updates its parameters of hidden states and cell stated by each batch in the time sequential order. The similarity between data points in the current batch and the next batch affects how much of the parameters have been updated after each batch. In another word, it requires a large amount of training data to tune its model setting fully. However, due to the time limit of the experiment, it is hard to collect sufficient training data for the stateful LSTM.



(a) Validation CDF (b) Test CDF

Figure 5.18: CDF of stateful LSTM with different rounds

Therefore, we could make an assumption based on the improvement of the stateful LSTM model that sufficient amount of the training data allows the stateful model to learn the features of the internal dependencies between sensor sequential measurements. The stateful model can fully tune its weight and state if there are sufficient of rounds of data.

5.5 Evaluations

Starting with selecting a suitable time window from 10ms, 100ms, 1000ms and 2000ms, we choose 1000ms as the time window as it allows more variations of the data. Tuning the hyperparameters of the model with the learning rate of 0.05, 0.005, 0.0005, learning rules of SGD and RMSprop, we determine to use the stateless LSTM model with 0.005 learning rate and RMSprop learning rule. After that, we explore the influence of the training size between one round of input data with eight rounds of the training data. The results are not improved significantly. Therefore, we move to improve estimation accuracy by further increasing the training size with overlapping. With the 90% of the overlapping, the model shows a significant improvement of the locationing accu-

racy. To reduce the complexity and the training time of the models, we compress the training data size by downsampling and PCA dimension reduction. Finally,by comparing all the different model and data input settings in terms of different learning rate, learning rules, size of the training data, with overlapping, downsampling and PCA, we selected the stateless LSTM model with down sampling based on 90% overlapped training data as the locationing model which balances between the estimation accuracy and efficiency.

Figure 5.19 contains all models that have been explored in the experiment, CDF is generated based on the same one round of the test set. It is noticed that to satisfy each model's input data structure requirement. The test file has been reshaped to the required data structure with different time windows, overlapping rate and downsampling as well as PCA dimension reduction. In general, all models' performance concentrates into three subgroups.

With the prediction accuracy of 80%, it is obvious that the downsampled overlapping based model has the best performance regarding the convergence rate and prediction accuracy that it has the maximum prediction error of 6 metres. PCA based model and overlapped based models have a larger estimation error of around 14 metres. Raw training data based models with different learning rules and learning rates settings about 18 metres. Compare between the downsampled model shown in blue line and the initial model of 1000 time window (time step) labelled as RMSprop in grey colour. The estimation error reduces significantly from 18 metres to 6 metres.

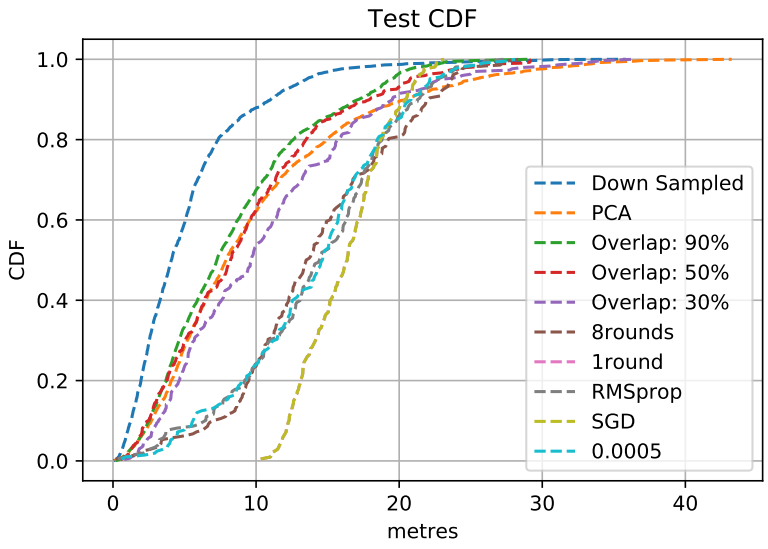


Figure 5.19: Overlap comparison CDF based on test set

Regarding evaluating our down sampled based stateless LSTM model, we collected two rounds of sensor data from the entry-level phone called Smartisan and the flagship phone called OnePlus. To be noticed, the sensor data received from the flagship phone has never used for model training. The model was only trained based on the data collected from the entry-level smartphone which has a lower sensor scanning rate.

Here, we feed two testing data into both down sampled based model and PCA based model. Figure 5.20 shows the CDF results. In general, down sampled based model performs better than PCA model for both Smartisan and OnePlus data. Meanwhile, it is reasonable that estimations from Smartisan data have higher accuracy than those from OnePlus data as all models were trained based on the data collected from Smartisan Phone. It approves model's robustness its ability to process unseen data from multiple devices.

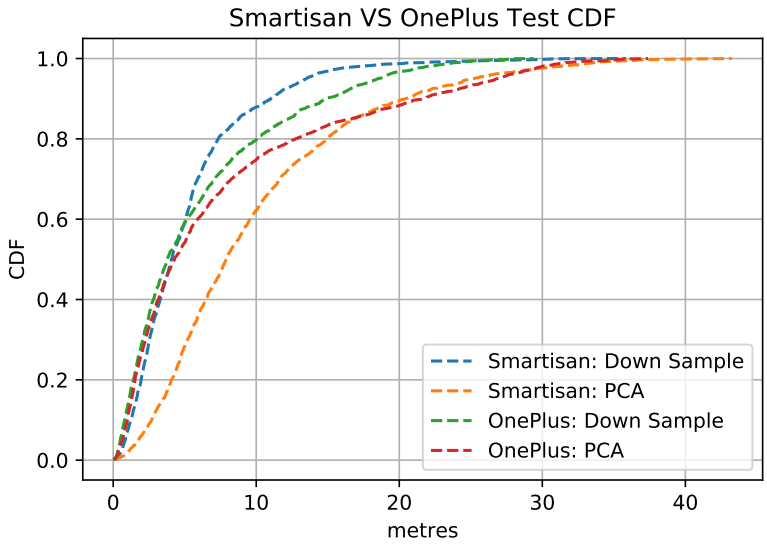


Figure 5.20: Test set prediction map

Table 5.5 listed part of the samples' estimation from the test data collected from Smartisan phone. (Prediction generated by the down sampled based model with 90% overlapping). The estimation errors shift from minimum 0.05 to maximum 38 metres while the average estimation error is 5.57 metres.

	Ground Truth		Estimation		
Sample	X	Y	X'	Y'	Error
1	29.92286	38.69607	22.85031	37.00532	7.27183
2	24.73692	37.00729	30.52618	39.40655	6.266732
3	19.55098	35.31851	12.22434	32.76583	7.7586
4	14.36505	33.62973	9.862818	33.39841	4.508168
5	9.179109	31.94095	20.13337	36.04194	11.69675
6	3.993172	30.25217	2.94912	28.24183	2.265285
7	0.35678	27.32769	0.897456	26.5135	0.977354
8	1.908	20.26627	0.800159	21.99939	2.056949
9	3.45922	13.20485	3.225421	14.8013	1.613477
10	5.010439	6.143433	4.655814	6.730763	0.686087
11	7.060323	0.227901	7.745199	3.065923	2.919491
12	12.44742	1.980984	9.279351	1.11019	3.285565
13	17.83452	3.734067	3.569337	-1.4285	15.17061
14	23.22161	5.48715	18.92965	3.849344	4.593837
15	28.60871	7.240233	23.72109	5.402205	5.221798
16	33.99581	8.993316	30.19011	8.259878	3.875723
17	39.3829	10.7464	9.624118	13.176	29.8578
18	38.26973	17.42241	39.0442	12.40384	5.077978
19	36.66729	24.46897	37.72407	26.48625	2.277321
20	35.06485	31.51553	37.2802	18.89889	12.80966
...	...	...	...	...	...
Minimum error: 0.050345603 metres					
Maximum error: 37.94992097 metres					
Average error: 5.573350694 metres					

Table 5.4: Numerical comparison between ground truth and estimations

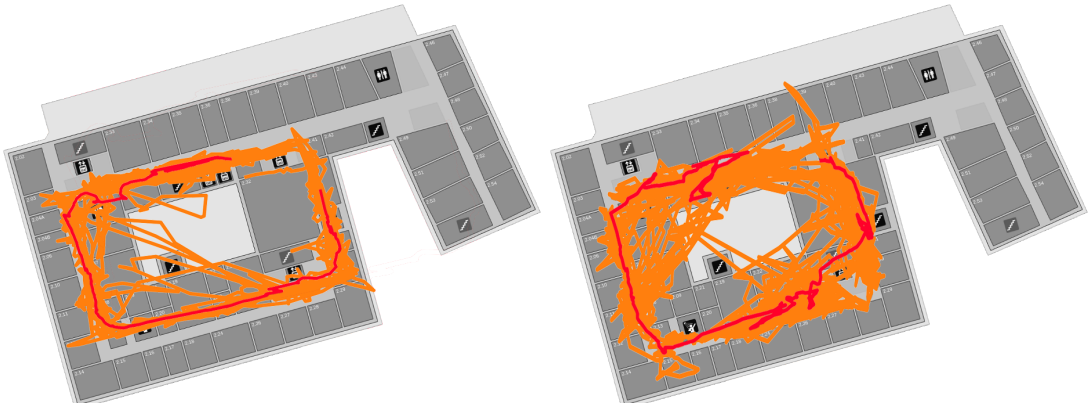
5.6 Visualisation

Besides of CDF figures, we visualise the results by plotting the estimation over the graphical map. As the map has integrated with location information in each pixel, prediction results could be matched with the pixel on the map.

Figure 5.21(a) shows the estimation results of the testing data collected from the entry-level phone. The orange colour route is the raw prediction. It is generated by lining all prediction results of latitude and longitude together. As the testing data is processed with overlapping 90% and then downsampled, there are a large number of samples included in one round of data. This is the reason why the orange route is almost cover the corridor densely. We plot the filtered routine in red colour by getting the average value every 100 samples' predictions.

Figure 5.21(b) shows the locationing results of the testing data collected from the flagship phone. The orange colour route is the raw prediction while the red path is the averaged results from every 100 samples.

According to both results, downsampled based model can provide a reliable estimation as it generates the route which follows the ground truth routine. Smartisan based prediction accurately converge to the corridor while OnePlus base estimations roughly return a stable prediction routine.



(a) Location Prediction from Smartisan (b) Location Prediction from OnePlus

Figure 5.21: Location Prediction Map

Chapter 6

Conclusion

6.1 Summary

Due to the limitations of the GPS in the indoor environments, researchers are looking for alternative indoor localisation solutions. Indoor locationing systems based on smartphone built-in sensors is a popular research topic due to the complexity of predicting user's location from noisy and unreliable smartphone sensors. Traditional indoor localization systems work on a set of well-defined equations such as fingerprinting or pedestrian dead reckoning as well as on probabilistic positioning methods. In this work, we provide a new solution that uses a stateless LSTM regression model to estimate user's location based on smartphone sensors (accelerator, gyroscope and magnetometer). It returns the location estimation from the sensor data input on time windows.

By selecting a proper time window which allows the model to cover more variations, choosing the input features, tuning the model hyper parameter settings such as learning rate, optimiser, we determined the model structure of using time window of one second, three sensors' magnitude across the three orthogonal axis value inputs, learning rate of 0.005 and optimiser of RMSprop and eight rounds of training data size as the baseline model. After that, we implement overlapping of 30%, 50% and 90% to further increase the training data by repeating measurements in each samples. Meanwhile, in order to improve efficiency, we used the down sampling and PCA to compress the input data size.

Finally, with the down sampling of 90% overlapping input data, the stateless LSTM regression model achieves the prediction accuracy of 80% with the error of 8 metres. The median estimation error is 5.57 metres.

6.2 Future Work

As mentioned in time window section. A larger time window allows the model to deal with variations and other activates such as walking and climbing stairs. However, due to the time constrain for this project, there is no sufficient time to build new training data which contains climbing stairs data. Therefore, to extend model's functions, we continue our research by collecting climbing data and train the model with a dataset extension using the current model settings.

Meanwhile. in terms of the stateful LSTM model which read the training data in time sequential order mentioned in the thesis. By increasing the amount of training data, the model has a trend to perform more accurate and stable. Therefore, by using cloud computing in the future [34], users' movement data is authorised [35] to upload to the server which allows the model to learn the pattern between sensor measurements and location information continuously. Stateful LSTM model could therefore fully tune its weight and state.

For further updates, please check project Github page:

`https://github.com/jsd1994wxj/LocationTracking`

Appendix A

MobiUK 2018 Research Symposium

MobiUK 2018 research symposium is the first symposium that provides an opportunity for discussion and presentation for the research within in the UK mobile, wearable and ubiquitous systems community.

End-to-End Machine Learning for Smartphone-based Indoor Localisation and Tracking using Recurrent Neural Networks

Xijia Wei and Valentin Radu

School of Informatics, University of Edinburgh

Indoor localization with smartphones is a favourite research topic for Mobile Systems researchers. This is due to the complexity of estimating accurate locations from noisy sensor data, and the potential benefit this can have on many location-based services. Various solutions to estimate indoor locations have been proposed to compensate for the obvious limitation of GPS inside buildings. Predominantly, these solutions rely on built-in sensors, such as the accelerometer, magnetometer, gyroscope and WiFi received signal strength, although integrating their signal streams to achieve a single accurate location estimation is not easy.

Currently, most indoor localization systems build on top of two main techniques, WiFi Fingerprinting and Pedestrian Dead Reckoning (PDR) [Radu and Marina, 2013]. Each of these two techniques functions on a firm set of equations defining the possible mobility frame, such as step counting, direction estimation and WiFi fingerprint matching. Their accuracy inherently depends on the well-defined mathematical formulation of these models, which identify the relationship between sensor data and location information. These are generally hard to determine to start with and are commonly fitted based on a small set of observations. We believe this is limiting the potential of indoor localization systems. Instead, we propose to avoid any intervention and hand-tuned formulation by relying on data alone and machine learning end-to-end to extract the relevant patterns that are not trivial for us to observe.

To replicate the construction of PDR, which starts from a known location and estimates consecutive locations based on sensor observations, we employ a range of sequential machine learning algorithms, commonly referred to as Recurrent Neural Networks (RNN), which estimates location from time windows of sensor data as input and an intermediary representation of previous estimations. Multimodal neural network constructions [Radu et al., 2018] permit the integration of a diverse set of sensors in different time window settings of the sensor signal.

Data is crucial to the proposed solutions, so we collected training sensor data in several runs over a long path on corridors, together with their ground truth location. We find that even with a reduced memory structure within the Long Short-Term Memory (LSTM) networks, estimations approximate the shape of the path well, with more than 80% of location estimations having error below 6 meters, while taking as input only inertial sensors (accelerometer, gyroscope and magnetometer) data. From a system perspective, our solution is energy-efficient without compromising on inference accuracy, which is achieved by reducing input data dimensionality (down sampling and PCA), making this ideal for mobile phones. We are expanding the system to capture more user mobility patterns including moving between floors.

References

[Radu and Marina, 2013] Radu, V. and Marina, M. K. (2013). Himloc: Indoor smartphone localization via activity aware pedestrian dead reckoning with selective crowdsourced wifi fingerprinting. In *Proc. IEEE IPIN 2013*.

[Radu et al., 2018] Radu, V., Tong, C., Bhattacharya, S., Lane, N., Mascolo, C., Marina, M., and Kawsar, F. (2018). Multimodal deep learning for activity and context recognition. *IMWUT*, 1(4).

Appendix B

Training Time Spent

Figure C.1 shows the time spend per epoch of different models.

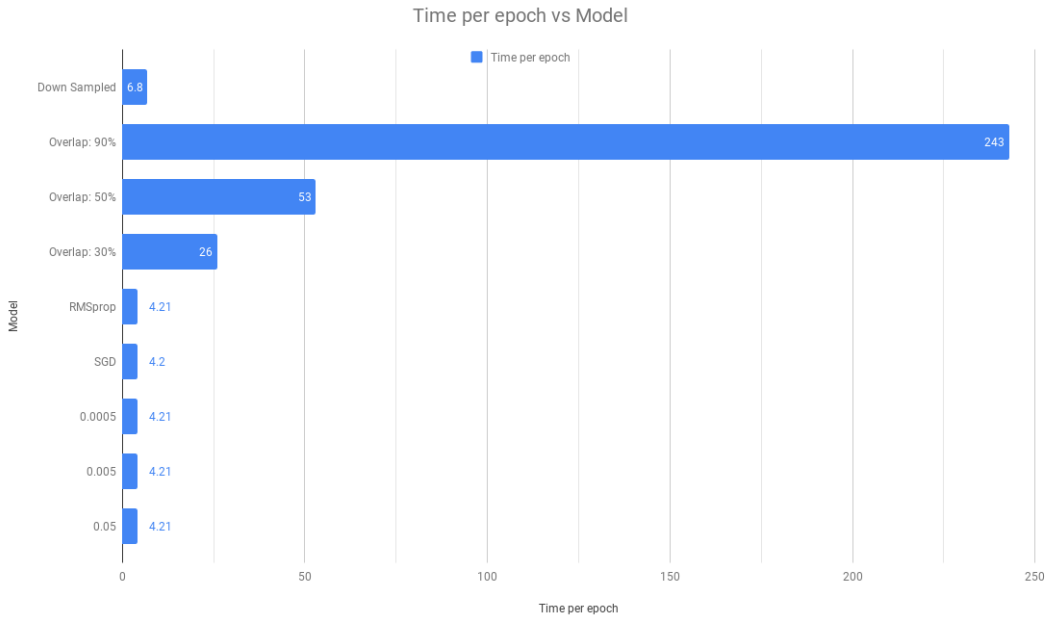


Figure B.1: Time spend per epoch VS Models

Appendix C

Data Visualisation Tool

Due to some unknown errors, when gathering the sensor data, we found that part of the log histories are missing in the raw log file. It happens occasionally and those data can not be used for training data. Therefore, we write the code to visualise the sensor data before transforming it to the training file Figure C.1 shows one example that contains missing measurements.

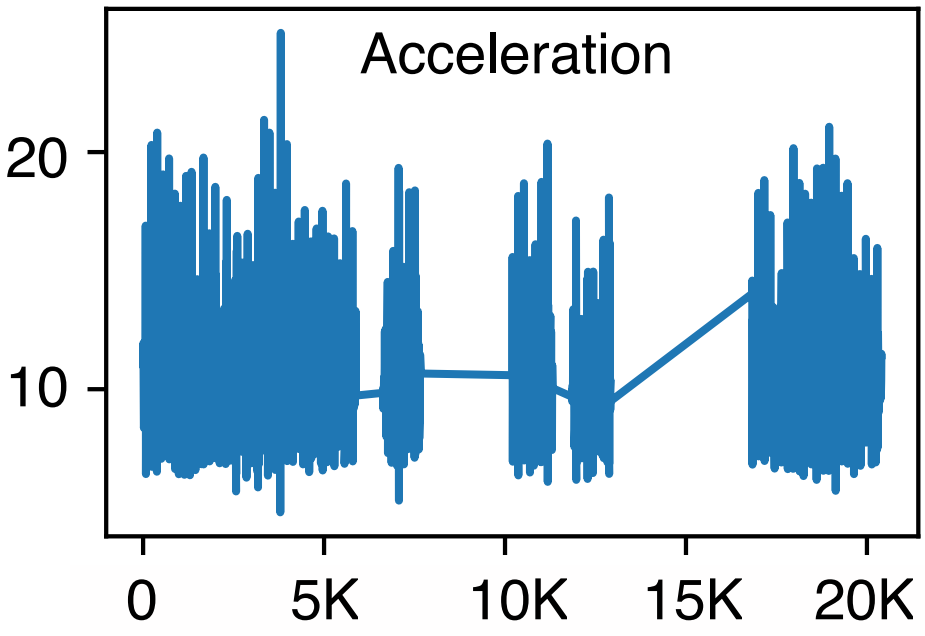


Figure C.1: Data Visualisation

Bibliography

[1] Jouni Rantakokko, Joakim Rydell, Peter Strömbäck, Peter Händel, Jonas Callmer, David Törnqvist, Fredrik Gustafsson, Magnus Jobs, and Mathias Gruden. Accurate and reliable soldier and first responder indoor positioning: multisensor systems and cooperative localization. *IEEE Wireless Communications*, 18(2):10–18, 2011.

[2] V Radu, C Tong, S Bhattacharya, ND Lane, C Mascolo, MK Marina, and F Kawsar. Multimodal deep learning for activity and context recognition. *IMWUT*, 1(4), 2018.

[3] Hui Liu, Houshang Darabi, Pat Banerjee, and Jing Liu. Survey of wireless indoor positioning techniques and systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(6):1067–1080, 2007.

[4] Pratap Misra and Per Enge. Global positioning system: Signals, measurements and performance second edition. *Massachusetts: Ganga-Jamuna Press*, 2006.

[5] Valentin Radu and Mahesh K. Marina. Himloc: Indoor smartphone localization via activity aware pedestrian dead reckoning with selective crowdsourced wifi fingerprinting. In *Proc. IEEE IPIN 2013*, 2013.

[6] Tsung-Nan Lin and Po-Chiang Lin. Performance comparison of indoor positioning techniques based on location fingerprinting in wireless networks. In *Wireless Networks, Communications and Mobile Computing, 2005 International Conference on*, volume 2, pages 1569–1574. IEEE, 2005.

[7] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 1999.

[8] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase repre-

sentations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[9] Stephane Beauregard and Harald Haas. Pedestrian dead reckoning: A basis for personal positioning. In *Proceedings of the 3rd Workshop on Positioning, Navigation and Communication*, pages 27–35, 2006.

[10] Jack Steenstra, Alexander Gantman, Kirk Taylor, and Liren Chen. Location based service (lbs) system and method for targeted advertising, March 23 2006. US Patent App. 10/931,309.

[11] Jeffrey H Reed, Kevin J Krizman, Brian D Woerner, and Theodore S Rappaport. An overview of the challenges and progress in meeting the e-911 requirement for location service. *IEEE Communications Magazine*, 36(4):30–37, 1998.

[12] Margaret McCartney. Margaret mccartney: game on for pokémon go. *BMJ*, 354:i4306, 2016.

[13] Beom-Ju Shin, Kwang-Won Lee, Sun-Ho Choi, Joo-Yeon Kim, Woo Jin Lee, and Hyung Seok Kim. Indoor wifi positioning system for android-based smartphone. In *Information and Communication Technology Convergence (ICTC), 2010 International Conference on*, pages 319–320. IEEE, 2010.

[14] Anja Bekkelien, Michel Deriaz, and Stéphane Marchand-Maillet. Bluetooth indoor positioning. *Master's thesis, University of Geneva*, 2012.

[15] Chunhan Lee, Yushin Chang, Gunhong Park, Jaeheon Ryu, Seung-Gweon Jeong, Seokhyun Park, Jae Whe Park, Hee Chang Lee, Keum-shik Hong, and Man Hyung Lee. Indoor positioning system based on incident angles of infrared emitters. In *Industrial Electronics Society, 2004. IECON 2004. 30th Annual Conference of IEEE*, volume 3, pages 2218–2222. IEEE, 2004.

[16] Yao Zhao, Liang Dong, Jiang Wang, Bo Hu, and Yuzhuo Fu. Implementing indoor positioning system via zigbee devices. In *Signals, Systems and Computers, 2008 42nd Asilomar Conference on*, pages 1867–1871. IEEE, 2008.

[17] Antonio R Jimenez, Fernando Seco, Carlos Prieto, and Jorge Guevara. A comparison of pedestrian dead-reckoning algorithms using a low-cost mems imu. In *Intelligent Signal Processing, 2009. WISP 2009. IEEE International Symposium on*, pages 37–42. IEEE, 2009.

[18] Paramvir Bahl and Venkata N Padmanabhan. Radar: An in-building rf-based user location and tracking system. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 775–784. Ieee, 2000.

[19] Krishna Chintalapudi, Anand Padmanabha Iyer, and Venkata N Padmanabhan. Indoor localization without the pain. In *Proceedings of the sixteenth annual international conference on Mobile computing and networking*, pages 173–184. ACM, 2010.

[20] Marc Ciurana, Francisco Barceló, and Sebastiano Cugno. Indoor tracking in wlan location with toa measurements. In *Proceedings of the 4th ACM international workshop on Mobility management and wireless access*, pages 121–125. ACM, 2006.

[21] Nirupama Bulusu, John Heidemann, and Deborah Estrin. Gps-less low-cost outdoor localization for very small devices. *IEEE personal communications*, 7(5):28–34, 2000.

[22] Roberto Battiti, Nhat Thang Le, and Alessandro Villani. Location-aware computing: a neural network model for determining location in wireless lans. Technical report, University of Trento, 2002.

[23] MILOŠ BORENOVIĆ, ALEKSANDAR NEŠKOVIĆ, and Djuradj Budimir. Space partitioning strategies for indoor wlan positioning with cascade-connected ann structures. *International journal of neural systems*, 21(01):1–15, 2011.

[24] Andreas Teuber, Bernd Eissfeller, and Thomas Pany. A two-stage fuzzy logic approach for wireless lan indoor positioning. In *Proc. IEEE/ION Position Location Navigat. Symp*, volume 4, pages 730–738, 2006.

[25] Yiqiang Chen, Qiang Yang, Jie Yin, and Xiaoyong Chai. Power-efficient access-point selection for indoor location estimation. *IEEE Transactions on Knowledge and Data Engineering*, 18(7):877–888, 2006.

[26] Andrzej Mackiewicz and Waldemar Ratajczak. Principal components analysis (pca). *Computers and Geosciences*, 19:303–342, 1993.

[27] Carl R May, Frances Mair, Tracy Finch, Anne MacFarlane, Christopher Dowrick, Shaun Treweek, Tim Rapley, Luciana Ballini, Bie Nio Ong, Anne Rogers, et al.

Development of a theory of implementation and integration: Normalization process theory. *Implementation Science*, 4(1):29, 2009.

[28] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[29] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[30] Antonio Gulli and Sujit Pal. *Deep Learning with Keras*. Packt Publishing Ltd, 2017.

[31] Bernard W Silverman. *Density estimation for statistics and data analysis*. Routledge, 2018.

[32] Léon Bottou. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pages 421–436. Springer, 2012.

[33] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[34] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, et al. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.

[35] Peter Carey. *Data protection: a practical guide to UK and EU law*. Oxford University Press, Inc., 2018.