



THE HONG KONG
POLYTECHNIC UNIVERSITY
香港理工大學

Department of Electronic and Information Engineering

Final Year Project Final Report

(2019/20)

RSSI-Based Sigfox Localization by using Machine Learning

Student Name: HST

Student ID: 150xxxxxD

Programme Code: 42470

Supervisor(s):

Submission Date: May, 2020

Abstract

In recent years, the demand for services providing accurate positioning of objects, such as people and cars, increases rapidly. Localization (positioning) is a common technique applied in different applications. The main purpose of this project is to investigate the performance of the RSSI-based Sigfox localization with machine learning applied, and to find out the best solution to minimize the distance of localization error. Sigfox is one of the wireless technologies in a low-power wide-area network (LPWAN). It is mainly developed for the data communication of battery saving Internet of Things (IoT). Sigfox characterizes low-power consumption, long-range, low cost and resilient to interference [1]. Those characteristics enable Sigfox to become the most competitive alternative for localization rather than just a link to transceive data.

In this project, localization of an object performs by received signal strength indicator (RSSI). However, RSSI is easy to be affected by the environment factors like blocking and multi-path. Fortunately, not all the base stations (receivers) are highly affected by noise. Some of them are still in good receiving condition. Based on this finding, a RSSI fingerprint classifier by machine learning is employed to boost the performance of RSSI-based localization. The fingerprint is the RSSI values of the base stations with their coordinates. This project believes that there are certain RF features capable of identifying a location uniquely and stably. Based on this approach, some algorithms are proposed to perform localization:

1. RSSI-based Linear Least Square (LLS) Localization
2. RSSI-based Linear Least Square Localization with Support Vector Machine (LLS-SVM)
3. RSSI-based Linear Least Square Localization with Deep Neural Network (LLS-DNN)
4. RSSI-based Linear Least Square Localization with 1 Dimensional Convolutional Neural Network (LLS-1DCNN)

The first algorithm proposes to perform localization by considering RSSI of all received base stations. The last three algorithms are to identify a group of base stations which is not highly affected by noise by the classifier and use them to perform localization.

The performance of the proposed algorithms is investigated by on-field experiments at Avenue of Stars in Tsim Sha Tsui, Hong Kong. An online system is developed to collect, store and check the RSSI data in real-time. Finally, the comparison of the performance of different proposed localization algorithms is shown. The results show that our proposed algorithms are effective.

The Hong Kong Polytechnic University

Department of Electronic and Information Engineering

EIE4430 / EIE4433 Honours Project

1. Student Name: _____HST_____ (Student No.: 150xxxxxD)
2. Programme Code: 42470 / 42477
3. Project Title: RSSI-Based Sigfox Localization by using Machine Learning
4. Supervisor Name: Dr.
5. Project summary (State clearly the project objectives and results achieved by yourself.)

[Please list in point form where appropriate]

- Construction of a platform for data collection, data storage and instant data verification,
- Data collection from real world (in Hong Kong),
- Model training of machine learnings (SVM) and neural networks (DNN and CNN),
- Classification accuracy comparison between different machine learnings,
- Performance investigation between pure RSSI-based localization, and localization with SVM, DNN and CNN.

DECLARATION OF ORIGINALITY

Except where reference is made in the text of this report, I declare that this report contains no material published elsewhere or extracted in whole or in part from any works or assignments presented by me or any other parties for another subject. In addition, it has not been submitted for the award of any other degree or diploma in any other tertiary institution.

No other person's work has been used without due acknowledgement in the main text of the Report.

I fully understand that any discrepancy from the above statements will constitute a case of plagiarism and be subject to the associated.

Signature

Table of Contents

Table of Contents	5
Table of Figures.....	7
Table of Tables.....	10
1 Introduction	11
1.1 Motivation and Objective	11
1.2 Definition of Localization	12
1.3 Location Estimation Approaches	12
1.4 Sigfox Technology	12
1.5 Previous Works	13
1.6 Hardware Platforms.....	14
1.7 Software Platform Used	16
2 Background	17
2.1 Location Estimation Mathematical Model – Linear Least-Squares	17
2.2 Support Vector Machine Algorithm.....	18
2.3 Deep Neural Network.....	21
2.4 1D Convolutional Neural Network	23
3 Methodology	24
3.1 Proposed Localization Algorithm with Machine Learnings	24
3.2 Proposed Coordination System on Hong Kong Map	25
3.3 Data Structure.....	26
3.4 Sigfox RSSI Data Collection, Storage and Checking System.....	26
System Overview	26
Overall Operation	27
Overall Design.....	28
Target.....	28
Target – PCB Design.....	28
Target – Firmware	29

Software Platform Design	30
Real-time Localization Estimation Server Design	31
3.5 Structure of the Proposed Algorithm.....	33
3.6 RSSI Data Pre-processing	34
Empty Cell Handling.....	34
Data Augmentation on Training Data	35
Normalization.....	36
4 Discussion	36
5 Experiments and Results	37
5.1 Locations to Collect Sigfox RSSI Dataset	37
5.2 The Dataset.....	38
5.3 Characteristics of Sigfox RSSI.....	39
5.4 Assessment Methods to Classifiers	40
5.5 Performance of SVM Classifiers.....	41
SVM Classification Accuracy	41
SVM Localization Accuracy	42
5.6 Performance of DNN Classifiers.....	42
DNN Classification Accuracy	42
DNN Localization Accuracy	44
5.7 Performance of 1D-CNN Classifiers.....	44
1D-CNN Classification Accuracy	44
1D-CNN Localization Accuracy	45
5.8 Discussion and Comparison	46
Accuracy Comparison of Classifiers	46
Localization Error Comparison	47
6 Conclusion and Future Developments	49
6.1 Conclusion.....	49

6.2	Future Developments	50
	Noise Reduction by Kalman Filter	50
	Increase the Number of Reference Regions	50
	Increase the Resolution of Reference Regions	50
	Localization as Regression by Deep Neural Networks	51
7	Reference.....	52

Table of Figures

Figure 1.1	A Sigfox base station with antenna on a building's rooftop in Kwai Hing, HK. ...	13
Figure 1.2	Raspberry Pi 4B (left) and Jetson Nano (right) by NVIDIA.	14
Figure 1.3	Arduino Mega 2560, Sigfox shield (Xkit) and Sigfox wireless module.	15
Figure 1.4	The front and back side of the SIM5320E module.	16
Figure 1.5	The user interface to access the Sigfox backend.	16
Figure 2.1	Linear Support Vector Machine.....	19
Figure 2.2	Illustration of the mapping function ϕ	20
Figure 2.3	One-vs-rest method in SVM.	21
Figure 2.4	A neuron. Source: [11].....	22
Figure 2.5	The structure of a DNN. Source: [12].....	22
Figure 2.6	The structure of 1D-CNN. Source: [13].	24
Figure 3.1	The overview of the localization algorithm.	24
Figure 3.2	A proposed plane coordinate system.	25
Figure 3.3	Block diagram of the data collection & storage and checking system.	27
Figure 3.4	(Left) Arduino Mega 2560 with 3G&GPS module and Sigfox shield (Xkit).....	28
Figure 3.5	(Right) Block diagram of the target for RSSI & GPS data collection.	28
Figure 3.6	Schematic (circuit diagram) of the target.....	29
Figure 3.7	(Left) PCB layout drawing of the shield for the target.	29
Figure 3.8	(Right) Physical PCB of the shield for the target.	29

Figure 3.9 The logic flow of the firmware in Arduino Mega 2560.....	30
Figure 3.10 The button on the Sigfox Xkit shield to trigger GPS data transmission.	30
Figure 3.11 The logic flow of the web application running in Google Cloud Functions.	31
Figure 3.12 The logic flow of the server runs in Jetson Nano/Raspberry Pi.	32
Figure 3.13 The Jetson Nano/Raspberry Pi server is ready to receive HTTP requests.....	32
Figure 3.14 The result of the real-time localization estimation from the server.	32
Figure 3.15 Detail structure of the localization algorithm.	33
Figure 3.16 Steps in the data pre-processing.....	34
Figure 3.17 Illustration of the way to deal with empty cells.	35
Figure 3.18 Original RSSI set vs. Augmented RSSI set of a measurement.....	36
Figure 5.1 The location to collect data is in Avenue of Stars in Tsim Sha Tsui, Hong Kong.	37
Figure 5.2 Distribution of the number of measurements.....	38
Figure 5.3 Distribution of the training and testing sets.	39
Figure 5.4 RSSI values over time.....	39
Figure 5.5 The distribution and histogram of RSSI values of two base stations.	40
Figure 5.6 The relationship between RSSI values and distances.	40
Figure 5.7 Comparison of the results of SVM with different kernels.....	42
Figure 5.8 Confusion matrix of the results from SVM with different kernels.	42
Figure 5.9 Accuracy of the DNN classifier changes with the number of hidden nodes.	43
Figure 5.10 Performance comparison between linear SVM, DNN and 1D-CNN. The output plots show the results of each classifier in 2-dim space whose dimensions are reduced from 71 by principal component analysis (PCA). The accuracies shown here are close to their individual best average accuracy.....	47
Figure 5.11 Result comparison of the pure LLS and LLS-SVM or LLS-DNN or LLS-1DCNN. The red marker with a dialog box is the real location. The grey dot is the estimated location by pure LLS while the blue dot is the estimated location by LLS-SVM, LLS-DNN or LLS-1DCNN.....	49

Figure 6.1 The structure of DNN to perform regression rather than classification tasks. Source: [11].	52
--	----

Table of Tables

Table 1.1 Detail technical specification of Sigfox.	13
Table 1.2 Specification of the Sigfox wireless communication module.	15
Table 3.1 Data structure for data storage.	26
Table 4.1 Experimental results of the classifiers with and without data augmentation.	37
Table 5.1 The accuracy of SVM classifiers with different kernel functions.	41
Table 5.2 The statistics of the localization error of LLS-SVM.	42
Table 5.3 Structure of the Deep Neural network.	43
Table 5.4 Performance comparison of DNN with different number of hidden layers.	43
Table 5.5 The statistics of the localization error (LE) of LLS-DNN.	44
Table 5.6 Structure of the 1D Convolution Neural Network.	45
Table 5.7 Hyperparameter tuning for 1D-CNN classifier.	45
Table 5.8 The statistics of the localization error (LE) of LLS-1DCNN.	46
Table 5.9 Comparison of the best average accuracy of different classifiers.	47
Table 5.10 Comparison of the mean localization error LE_{mean} of pure LLS localization, and LLS-SVM, LLS-DNN and LLS-1DCNN.	47
Table 5.11 Estimated locations and the min. LE by LLS-SVM, LLS-DNN or LLS-1DCNN over 4 regions.	48
Table 5.12 The statistics of the localization error (LE) of pure LLS localization.	48
Table 5.13 Localization error comparison of the pure LLS localization and LLS with machine learning.	48

1 Introduction

1.1 Motivation and Objective

Localization (positioning) plays a crucial role in many real-life applications. The most commonly used positioning system is the Global Navigation Satellite System (GNSS), such as GPS from the United States, GLONASS from Russian and BDS from China, etc. It is common to see that GPS-based localization is applied in different applications such as car navigation and asset tracking. But this kind of satellite-based localization technology is optimal for outdoor use but not indoor, due to signal blocking inside buildings. Also, systems using GNSS for continuous positioning are quite power-consuming and are not very suitable for low-powered applications like Internet of Things, which expect to operate over a long period of time with a very small in size. Sigfox is another wireless technology that characterizes low power-consuming and long-range. On top of this technology, the Received Signal Strength Indicator (RSSI) is adopted to develop different localization algorithms in this project. However, the performance of the localization algorithm with pure measured RSSI is not as good as expected, because RSSI is easy to be interfered by environment factors. The value of RSSI swings in a range of values, which increases the localization error. To deal with this issue, this project proposes different algorithms.

This project is to develop RSSI-based localization algorithms by applying machine learnings, and to prove that with the help of machine learnings they can increase localization accuracy and can minimize the localization error. Also, to obtain machine learning models, data collection and model training must be conducted first. So, the objectives aim to be achieved in this project:

1. Construction of a platform for data collection, data storage and instant data verification,
2. Data collection from real world (in Hong Kong),
3. Model training of machine learnings (SVM) and neural networks (DNN and CNN),
4. Classification accuracy comparison between different machine learnings,
5. Performance investigation between pure RSSI-based localization, and localization with SVM, DNN and CNN.

In future work, we can collect more RSSI datasets to increase the accuracy of the machine learning classifiers. Also, the spaces to collect RSSI data can be enlarged so that the localization service can be expanded.

1.2 Definition of Localization

Localization (positioning) is a process to give information about the location (position) of an object. The information may include the coordinates or environment parameters. A localization system is a system providing location information of an object [1]. Localization system is widely adopted in daily applications, such as logistics tracking and car navigation. It is an important field to study. To measure the performance of a localization system, the coverage area and accuracy (localization error) are employed [2].

1.3 Location Estimation Approaches

Various approaches are presented for location estimation. Several popular methods are the Phase of Arrival (PoA), Angle of Arrival (AoA), Time Difference of Arrival (TDOA), Received Signal Strength Indicator (RSSI) and vision positioning. The TDOA estimates the location of a target (also is a transmitter) by measuring the time differences of the arrival signal between more than 2 base stations (receivers). But it highly depends on the accurate timer synchronization between target and receivers [1]. Vision positioning usually uses cameras to recognize objects and to calculate its location by computer vision model and camera model. It is accurate, but the coverage area is small [2].

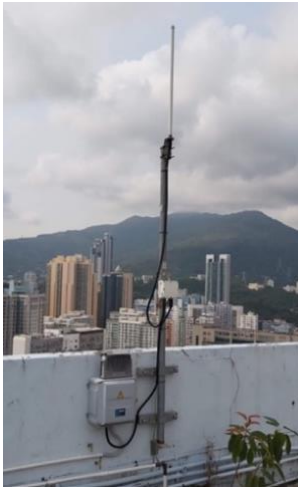
RSSI is the “total signal power received in milliwatts”, according to [3]. It relates to the distance from the signal source. The further the distance to the target is, the lower the RSSI value the receiver gets. The basic idea of the RSSI-based localization is to utilize the RSSI received by receivers to estimate the distances between target and those receivers, which then locates the target by finding the unique intersection point of at least three circles [1]. It is time efficient, easy to adopt and low cost.

1.4 Sigfox Technology

Sigfox is a wireless technology used in a low-power wide-area network (LPWAN). It is mainly developed for data communication of the Internet of Things (IoT) along a long distance. Also, Sigfox is a global company that operates the low-power wireless (Sigfox) network and provides Sigfox communication services. The Sigfox networks have been built to cover 60 countries by the end of 2018 [4]. Sigfox characterizes low-power consumption (battery lifetime up to 10+ years) [5], long-range (10 km for urban and 40 km for rural), low cost (low cost in hardware and chipsets, and network subscription) and resilient to interference (capacity to resist jamming) [6].

According to [6], Sigfox uses Ultra Narrow Band (UNB) technology to transmit message through region-depending frequency band. In Hong Kong, the licensed frequency band for Sigfox is 920 to 925 MHz [7]. A Sigfox message is 100 Hz wide. It allows Sigfox devices to send messages under a very limited bandwidth environment. In Hong Kong, Sigfox adopts frequency hopping technique to transmit the radio signals, so the broadcast device transmits an uplink message to Sigfox network 3 times on 3 different frequencies. Its center frequency for uplink and downlink is in 920.8 to 922.3 MHz. Each message can be sent at a 600 bit/s uplink data rate. Thanks to the 163.3 dB high budget link, Sigfox possesses a great ability of penetration and long-range communications. Table 1.1 shows the detail specification of Sigfox.

In Hong Kong, Thinxtra Network Limited deploys the Sigfox network and provides a variety of device-to-cloud IoT services for enterprises [8]. Figure 1.1 shows one of the base stations with antenna deployed by Thinxtra Network Limited in Hong Kong. In this project, a set of raw RSSI data was collected at Avenue of Stars in Tsim Sha Tsui, Hong Kong. The system to collect, store and check the RSSI data is introduced in “Methodology”.



Region	Hong Kong
Uplink center frequency	920.800 MHz
Downlink center frequency	922.300 MHz
Signal transmission technology	Frequency Hooping
Bandwidth of a message	100 Hz
Max data payload per message	Uplink: 12-bytes, Downlink: 8-bytes
Uplink/Downlink data rate	600 bit/s
Link budget	163.3 dB

Figure 1.1 A Sigfox base station with antenna on a building's rooftop in Kwai Hing, HK.

Table 1.1 Detail technical specification of Sigfox.

1.5 Previous Works

A study published in 2019 proved that a mathematical model for RSSI-based localization, Linear Least-Squares (LLS) position estimation model, is a low computational cost solution to locate a target (transmitter) [1]. By utilizing received RSSI values of all receivers, the position of the target can be estimated. Also, the suggested LLS method can handle Gaussian noise and provides the most optimal estimation. However, the study indicates that the accuracy of this LLS method degrades significantly if the non-Gaussian noise like multi-path and blocking

presents because the value of the received RSSI by some of the receivers is much smaller or higher than the expected value. It exceeds the noise handling ability of LLS method. So, this study suggested using clustering techniques, K-mean and DBSCAN, to eliminate the high non-Gaussian noise affected “bad” base stations, and proceed LLS localization by the remaining “good” base stations.

Also, a previous final year project report employed a machine learning method, Support Vector Machine (SVM), to learn from the RSSI values with known reference locations in simulation and real measurements. Then, eliminates the “bad” base stations which receive the out-of-range RSSI values (the expected RSSI) based on the classification results of SVM and perform LLS localization with remaining “good” base stations afterward. In the simulation, the classification accuracy of SVM is around 70%, but the accuracy decreases with the increases in the magnitude of non-Gaussian noise (caused by blocking and multi-path). To deal with the complex real-world environment, the following paragraphs, not only SVM but also neural networks, are introduced to help identify the “good” base stations and estimate the position of a target.

1.6 Hardware Platforms

Raspberry Pi and Jetson Nano Development Kit

Raspberry Pi and Jetson Nano are the single board computers with USB, HDMI, Ethernet ports, and even I/O pins supported for UART/SPI/I2C. Raspberry Pi is designed by Raspberry Pi Foundation, while the Jetson Nano is designed by NVIDIA. They are small in size and lower power consumption than PCs. Raspberry Pi 4B is in 88 x 58 x 19.5mm, and Jetson Nano is in 100 x 80 x 29 mm. Both of them are suitable for this project to build a personal server. The significant difference between the Jetson Nano and Raspberry Pi is that there is a GPU (128-core NVIDIA Maxwell) on the Jetson Nano. This GPU provides Jetson Nano with more powerful capabilities for machine learning than Raspberry Pi does. It makes Jetson Nano more suitable for this project in the future.



Figure 1.2 Raspberry Pi 4B (left) and Jetson Nano (right) by NVIDIA.

Arduino and Sigfox Development Kit

This project adopts Arduino development board, Arduino Mega 2560, to be the controller of the target (transmitter). Arduino is famous for easy to use, open-source hardware and software, enrich documents, and large amount of samples and users. Arduino Mega 2560 is one of the products in Arduino. A 8-bit microcontroller unit, Atmel ATmega2560, with 16 MHz maximum clock speed is used on this board. It has 54 digital I/O pins, 16 analog inputs and 4 hardware serial ports (UARTs).

Another electronic board is a Sigfox development shield. A Sigfox wireless module, the light and temperature/pressure sensors, accelerometer and reed (magnetic) switch are embedded on this shield. WSSFM10R4AT, a model of the Sigfox modules, is used in Hong Kong region. The details of the Sigfox wireless module are shown in Table 1.1.

Table 1.2 Specification of the Sigfox wireless communication module.

Module Name	WSSFM10R4AT	Tx RF Frequency	920.8 MHz (typ.)
Input Voltage	2.7 to 3.6 V	Rx RF Frequency	922.3 MHz (typ.)
Tx Current	200 mA (typ.)	Max Tx Output power	22.5 dBm (typ.)
Rx Current	32 mA (typ.)	Data rate	600 bps
Sleep Current	2.5 A (typ.)		

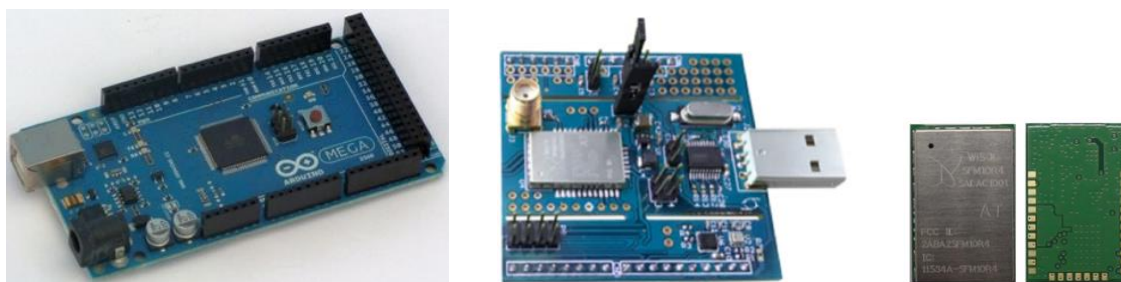


Figure 1.3 Arduino Mega 2560, Sigfox shield (Xkit) and Sigfox wireless module.

3G & GPS Module

The 3G&GPS module, SIM5320E, is used in this project to record the GPS location of the reference points. This module is designed by SIMCom. It supports up to HSDPA (3.5G) mobile protocol with maximum 3.6 Mbps downlink data rate. Rich interfaces including USB2.0, UART, I2C, and SPI, etc. are provided. Also, it supports large amount of application protocols, like TCP, UDP, HTTP, HTTPS and SMTP, etc. For GPS/location specifications, it supports not only stand-alone GPS, but also A-GPS (mobile-station-based and mobile-station-assisted), which can entirely fulfil the needs of this project.

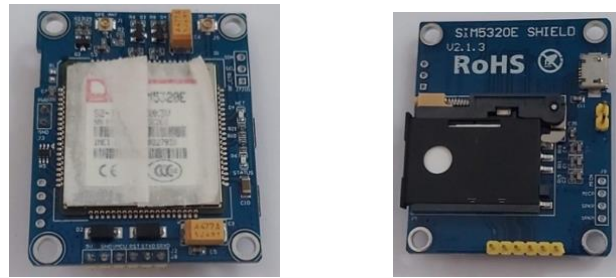


Figure 1.4 The front and back side of the SIM5320E module.

1.7 (X) Software Platform Used

Sigfox Backend Server

The Sigfox backend server called “Sigfox portal” is an internet server that stores and forwards the received RSSI from the base stations. It is operating by Sigfox.

Device 3E81CB - Messages

From date: To date:

RESET FILTER

page 1

Time	Delay (s)	Seq Num	Data / Decoding	Base station reception attributes				Callbacks	Location
				Station	RSSI (dBm)	SNR (dB)	Freq (MHz)		
2020-05-12 13:57:11	1.2	1870	9e093f5a6901f5ffa6fffa00 ASCII: 32L.....	790C	-106.00	19.01	920.8715		
				80C7	-104.00	16.82	920.8714		
				7C6B	-124.00	9.15	920.8343		
2020-05-07 16:28:10	2	1869	ba0c648242011900f8000e00 ASCII: .d.B.....	80C7	-107.00	22.99	920.8533		
				790C	-114.00	10.62	920.8124		
				7C6B	-129.00	10.06	920.8312		

Copyright © Sigfox - 9.3.2-50503cf-20200430.084513 - 296 - Terms and conditions / Cookie policy.

Figure 1.5 The user interface to access the Sigfox backend.

(X) TensorFlow

2 Background

2.1 Location Estimation Mathematical Model – Linear Least-Squares

Linear Least-Squares (LLS) is a popular mathematical algorithm for RSSI-based localization. RSSI is to describe the total signal power received [9]. RSSI is developed based on the free-space path loss model.

The free-space path loss model is defined as follows,

$$(2.1) \quad \text{free space path loss } L = 32.45 + 20\log_{10}f_c + 20\log_{10}d(p_t, p_n),$$

where

$f_c = \text{carrier frequency}$

$d(p_t, p_n) = \text{distance between a target and a base station (receiver)}$

Measured RSSI between the target (transmitter) and the receiver is the difference between the path loss of the target at a known distance d_0 and an unknown point $p_t = (x_t, y_t)$ from the receiver. So, the distance $d(p_t, p_n)$ between the target and the receiver can be estimated by its measured RSSI value in dBm, $\tilde{Z}(p_t, p_n)$, i.e.,

$$(2.2) \quad \tilde{Z}(p_t, p_n) = L_0 - L = 10\log_{10}(d_0) - 20\log_{10}d(p_t, p_n) = \tilde{Z}_0 - 10\tilde{\alpha}\log_{10}d(p_t, p_n) + \omega,$$

where

$$(2.3) \quad d(p_t, p_n) = \sqrt{(x_t - x_n)^2 + (y_t - y_n)^2} = \text{distance between a target and receiver},$$

$p_t = (x_t, y_t) = \text{coordinate of the target},$

$p_n = (x_n, y_n) = \text{coordinate of the } n\text{th receiver}, n = 1, 2, \dots, N$

$\tilde{Z}_0 = \text{reference RSSI value of the target measured at a known distance } d_0$

$\tilde{\alpha} = \text{measured path loss exponent, normally 2 to 5}$

$\omega = \text{noise, represent by a zero mean Gaussian random variable}$

The above formulas show the relationship between RSSI values and distances. The below mathematical steps show how to derive an LLS localization model, which uses the measured RSSI values from a set of receivers to estimate the location of the target. Although RSSI-against-distance is not linear due to the \log_{10} function, LLS model (linear) can still be an approximate solution to estimate the location of the target based on the measured RSSI values (close to linear) in low computational cost.

By combining equation (2.2) with (2.3) and ignore the noise ω , we have

$$\begin{aligned}
 (2.2) \quad & \tilde{Z}(p_t, p_n) = \tilde{Z}_0 - 10\tilde{\alpha} \log_{10} d(p_t, p_n) + \omega, \\
 (2.2) + (2.3) \quad & 10^{\frac{2}{10\tilde{\alpha}}[\tilde{Z}_0 - \tilde{Z}(p_t, p_n)]} = (x_t - x_n)^2 + (y_t - y_n)^2, \\
 & 10^{\frac{2}{10\tilde{\alpha}}[\tilde{Z}_0 - \tilde{Z}(p_t, p_n)]} = (x_t^2 - 2x_t x_n + x_n^2) + (y_t^2 - 2y_t y_n + y_n^2), \\
 (2.4) \quad & -2x_t x_n - 2y_t y_n + (x_t^2 + y_n^2) = 10^{\frac{2}{10\tilde{\alpha}}[\tilde{Z}_0 - \tilde{Z}(p_t, p_n)]} - (x_n^2 + y_n^2),
 \end{aligned}$$

where $n = 1, 2, \dots, N$ is the ID of the receivers. Change it in vector form, we have

$$(2.5) \quad \mathbf{A}\theta = \mathbf{b},$$

where

$$\mathbf{A} = \begin{bmatrix} -2x_1 & -2y_1 & 1 \\ -2x_2 & -2y_2 & 1 \\ \vdots & \vdots & \vdots \\ -2x_N & -2y_N & 1 \end{bmatrix}, \quad \theta = \begin{bmatrix} x_t \\ y_t \\ x_t^2 + y_t^2 \end{bmatrix} \text{ and } \mathbf{b} = \begin{bmatrix} 10^{\frac{2}{10\tilde{\alpha}}[\tilde{Z}_0 - \tilde{Z}(p_t, p_n)]} - (x_1^2 + y_1^2) \\ 10^{\frac{2}{10\tilde{\alpha}}[\tilde{Z}_0 - \tilde{Z}(p_t, p_n)]} - (x_2^2 + y_2^2) \\ \vdots \\ 10^{\frac{2}{10\tilde{\alpha}}[\tilde{Z}_0 - \tilde{Z}(p_t, p_n)]} - (x_N^2 + y_N^2) \end{bmatrix}$$

Applying Linear Least-Square (LLS) [1] on the vector form equation (2.5), we have

$$(2.6) \quad \hat{\theta} = \mathbf{A}^\dagger \mathbf{b},$$

where

$$\hat{\theta} = \begin{bmatrix} \hat{x}_t \\ \hat{y}_t \\ (\hat{x}_t)^2 + (\hat{y}_t)^2 \end{bmatrix} \text{ and } \mathbf{A}^\dagger = (\mathbf{A}^T \mathbf{A})^{-1} - \mathbf{A}^T = \text{conjugate transpose matrix of } \mathbf{A}.$$

Note that $p_n = (x_n, y_n)$ is the location of the Sigfox base station (receiver) where $n = 1, 2, \dots, N$ is the ID of the base stations and $\hat{p}_t = (\hat{x}_t, \hat{y}_t)$ where \hat{p}_t is the LLS estimated location of the target. To get the vector $\hat{\theta}$, put the measured RSSI values into \tilde{Z}_0 and $\tilde{Z}(p_t, p_n)$, and put coordinates of base stations into x_n and y_n to matrix \mathbf{A}^\dagger and vector \mathbf{b} . Compute the $\mathbf{A}^\dagger \mathbf{b}$. Extracts the \hat{x}_t and \hat{y}_t in $\hat{\theta}$, they are the LLS estimated (x, y) coordinate of the target.

2.2 Support Vector Machine Algorithm

Vapnik and Chervonenkis introduced the Support Vector Machine (SVM) in 1963 [10]. SVM is a supervised learning model and is a binary classifier. SVM needs to learn from training data with labels and can solve linear and non-linear classification problems. The conceptual idea of SVM is as follows: map the input data to a very high-dimensional feature space if data is non-linearly separable. Construct a linear hyperplane (decision boundary/plane) in that feature space

to separate the support vectors of all two classification sets or all datasets, which separate them to be furthest away from the hyperplane (i.e., maximizing the margin of two classes).

For Linearly Separable Training Data

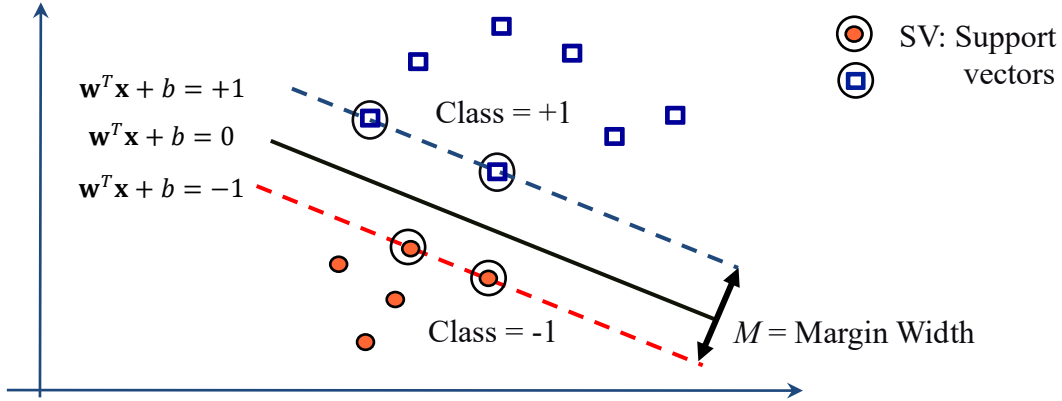


Figure 2.1 Linear Support Vector Machine

The classifying function (function of the hyperplane) is defined as follows,

$$(2.7) \quad g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$$

where \mathbf{x} = a data point (vector), \mathbf{w} = weight, b = bias.

The optimal hyperplane can maximize the margin between two classes. The maximal margin M equals $\frac{2}{\|\mathbf{w}\|}$. To derive the maximal M , \mathbf{w} should be minimized. The following optimization problem provides a solution for \mathbf{w} and b :

$$(2.8) \quad \min \frac{1}{2} \|\mathbf{w}\|^2, \quad \text{subject to } y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1$$

where $i = 1, 2, \dots, N$ = index of data, labels $y_i \in \{-1, +1\}$. Lagrange function can be employed to solve this quadratic optimization problem, i.e., to get \mathbf{w} and the bias b . After applied the Lagrange function, an objective function $Q(\alpha)$ with the Lagrange multiplier α is derived:

$$(2.9) \quad Q(\alpha) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

subject to the constraints $\sum_{i=1}^N \alpha_i y_i = 0$ and $\alpha_i \geq 0$ for $i = 1, 2, \dots, N$

After deriving the optimal Lagrange multipliers, $\{\alpha_{o,i}\}_{i=1}^N$ by $\frac{\partial Q(\alpha)}{\partial \alpha_i} = 0$, where $i = 1, 2, \dots, N$, the optimal weight vector \mathbf{w}_o and optimal bias b_o can be found.

The linear classifying function becomes:

$$(2.10) \quad g_o(\mathbf{x}) = \mathbf{w}_o^T \mathbf{x} + b = \sum_{i \in SV} y_i \alpha_{o,i} \mathbf{x}_i^T \mathbf{x} + b_o$$

where \mathbf{x}_i is the i -th support vector in the training dataset and \mathbf{x} is the testing dataset.

For Non-Linearly Separable Training Data

A linear hyperplane cannot be constructed on the input space due to non-linearly separable data. A non-linear mapping function $\phi(\mathbf{x})$ is used to map the input data to a new high-dimensional feature space, where the data is linearly separable on that space. Figure 2.2 illustrates how $\phi(\mathbf{x})$ works. The non-linear classifying function becomes:

$$(2.11) \quad g(\mathbf{x}) = \sum_{i \in SV} y_i \alpha_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x}) + b.$$

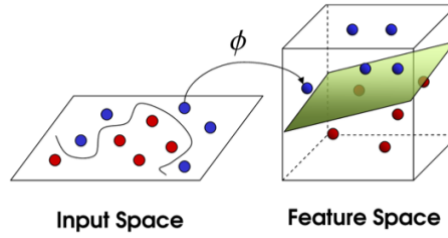


Figure 2.2 Illustration of the mapping function ϕ .

Source: S., Kernel yöntemi ile veriyi daha fazla boyutlu uzaya tasima islemi.png. 2016.

However, the dimension of $\phi(\mathbf{x})$ could be very high or can even be infinite, which indicates lots of computational power is needed, or this method is even not practicable. A kernel function $K(\mathbf{x}_i, \mathbf{x})$ can be used to replace the inner product, i.e.,

$$(2.12) \quad K(\mathbf{x}_i, \mathbf{x}) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}),$$

where computing the $K(\mathbf{x}_i, \mathbf{x})$ is much feasible than $\phi(\mathbf{x}_i)^T \phi(\mathbf{x})$. This “kernel trick” method avoids the need of mapping the data to a higher dimension space. Instead, it allows SVM to work in the original data dimension. But the decision boundary is not necessarily linear anymore, it can be non-linear. The below are some of the common kernel functions used in SVM,

5. Linear kernel: $K(\mathbf{x}_i, \mathbf{x}) = \mathbf{x}_i \cdot \mathbf{x} = \mathbf{x}_i^T \mathbf{x}$
6. Radial basis function (RBF) kernel: $K(\mathbf{x}_i, \mathbf{x}) = \exp \left\{ -\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{2\sigma^2} \right\}$
7. Polynomial kernel: $K(\mathbf{x}_i, \mathbf{x}) = (1 + \mathbf{x}_i \cdot \mathbf{x})^p, p > 0$

Multi-Class Classification In SVM

Although SVM can only handle two-group classification task, but the strategy called “one-vs-the-rest” can be employed to perform multi-class classification. Figure 2.3 shows the architecture of the one-vs-the-rest SVM model. Each SVM is trained to classify one label from the rest labels and to give a score about that label for a testing data. The one-vs-the-rest model picks the label from the SVM model with maximum score as the predicted label of that testing data.

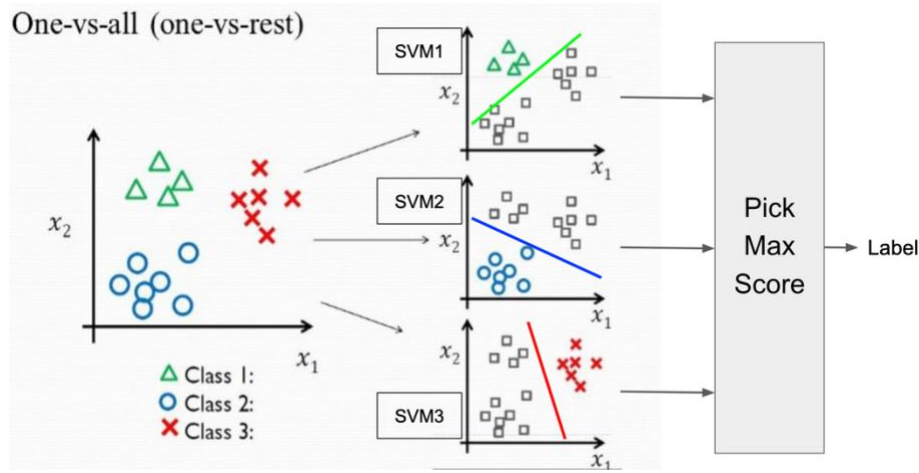


Figure 2.3 One-vs-rest method in SVM.

Source: M. Terry-Jack, one-vs-rest: combining multiple binary classifiers for multi-class classification. 2018.

2.3 Deep Neural Network

Artificial neural networks (ANNs) are the human-made machine learning models inspired by the biological brain. They are used to model and solve some complex problems like pattern recognition from a large amount of input data. The major advantage of ANNs is the ability to extract not only simple information (features) but also more abstract/hidden information from the known data. Deep Neural Network (DNN) is one of the types of ANNs.

Neurons are the smallest units in DNN. They equip with a non-linear activation function and are connected to each other to form a neural network so that they can receive and send signals. Figure 2.5 shows a neuron with inputs x_i , with the weights w_i and the bias term β where $i = 1, 2, \dots, n$. In this project, the rectified linear unit (ReLU) is employed in each neuron and the activation function σ of ReLU is

$$(2.13) \quad z = \sigma(\sum_{i=0}^n w_i x_i + \beta)$$

where z is the output of the ReLU.

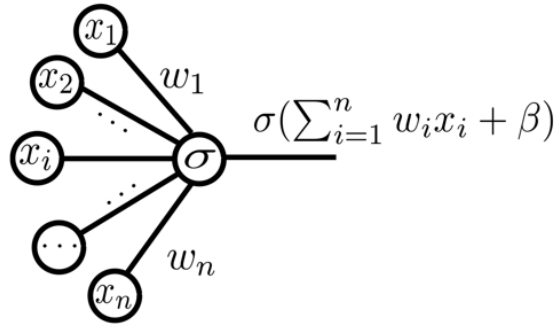


Figure 2.4 A neuron. Source: [11].

Each neuron with ReLU forms a decision boundary on the data feature space. The neurons can be connected to form a layer. With more and more layers, the DNN can model and solve much more complex problems because more decision boundaries present. Also, layers with different number of neurons and neurons in each layer with different activation functions can perform different tasks. In this project, our DNN consists of an input layer, hidden layers with ReLU neurons and a soft-max output layer. Figure 2.5 shows a common structure of the DNN.

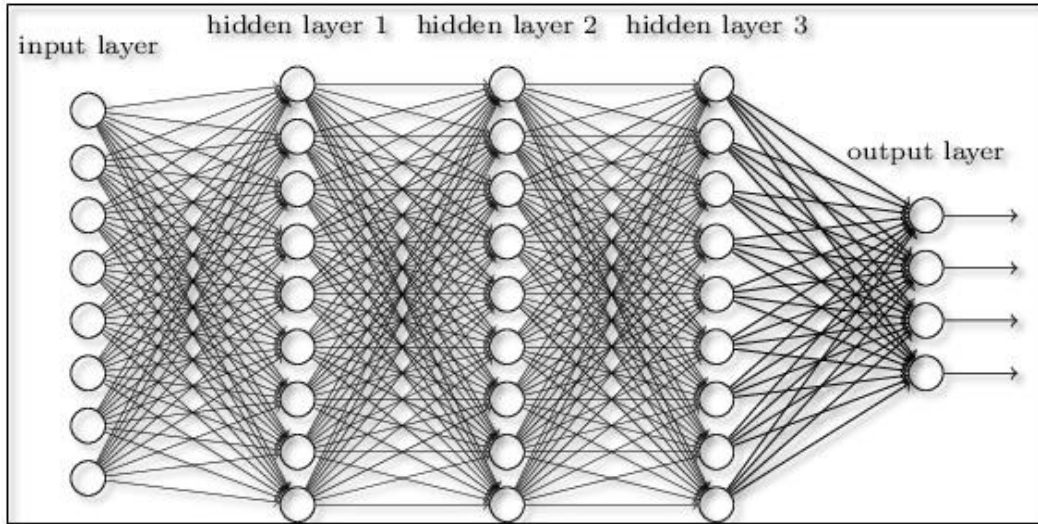


Figure 2.5 The structure of a DNN. Source: [12].

The number of neurons (nodes) in the input layer is at least equal to the number of base stations in this localization application because this layer treats the RSSI from the base stations as input data. The hidden layers are responsible to extract features from the input data. The last output layer employs a soft-max activation function to ensure the output nodes produce posterior probabilities to the given class labels. So, the predicted class label can be determined by the

largest posterior probability of that class. There are K nodes in the soft-max output layer if it is a K -class classification problem. The soft-max function is defined as follows,

$$(2.14) \quad y_k = \frac{\exp\{z_k^{(L)}\}}{\sum_{r=1}^K \exp\{z_r^{(L)}\}} \quad k = 1, 2, \dots, K \text{ in the } K\text{-class classification,}$$

where

$(L - 1)$ = the number of hidden layers in DNN,

$y_k = P(\text{Class} = k | \mathbf{x})$ = the posterior probability of the k th class label by given input data \mathbf{x} .

To train the DNN, the training data \mathbf{x}_n with target output $t_{n,k}$ should be provided, where $t_{n,k} \in \{0,1\}$ are the target outputs following one-hot encoding scheme and $n = 1, 2, \dots, N$ is the number of training data. The training of the DNN can be considered as the update of the weights between the neurons so that the output $y_{n,k}$ of the DNN can be as close as the target output $t_{n,k}$. Also, the training of DNN is to minimize the cross-entropy error between the target outputs and actual outputs. The cross-entropy error is defined as follows,

$$(2.15) \quad E = -\frac{1}{2} \sum_{n=0}^N \sum_{k=0}^{K-1} t_{n,k} \log(y_{n,k})$$

2.4 1D Convolutional Neural Network

CNN consists of several convolution layers which is usually followed by a pooling layer and a dropout layer, a flatten layer, and fully connected layers. The convolution layers are responsible to feature extraction. The dropout layer is used to help CNN avoid overfitting. The flatten layer transforms the input matrix into a vector and feeds this vector to the fully connected layers while the fully connected layers are used for classification. 2D-CNN is a popular deep learning model to perform image related application. However, in this project, 1D-CNN is employed to perform classification on the RSSI data. The reasons of utilizing 1D-CNN instead of 2D-CNN are: (1) RSSI is a 1D-signal (2) prevention of converting 1D-signal to 2D-image (3) the complexity is lower, which help reduce the computational burden of the system. In this project, a convolution layer, a pooling layer and a dropout layer are combined to form a convolution block. After several convolution blocks, they follow a flatten layer and the fully connected layers. Also, Table 5.6 shows the proposed structure of the 1D-CNN in this project.

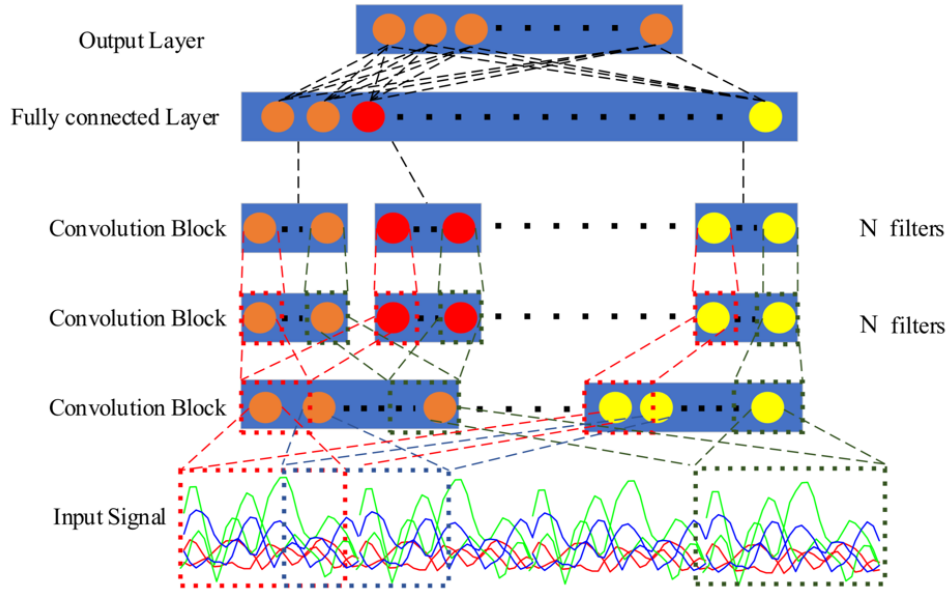


Figure 2.6 The structure of 1D-CNN. Source: [13].

3 Methodology

3.1 Proposed Localization Algorithm with Machine Learnings

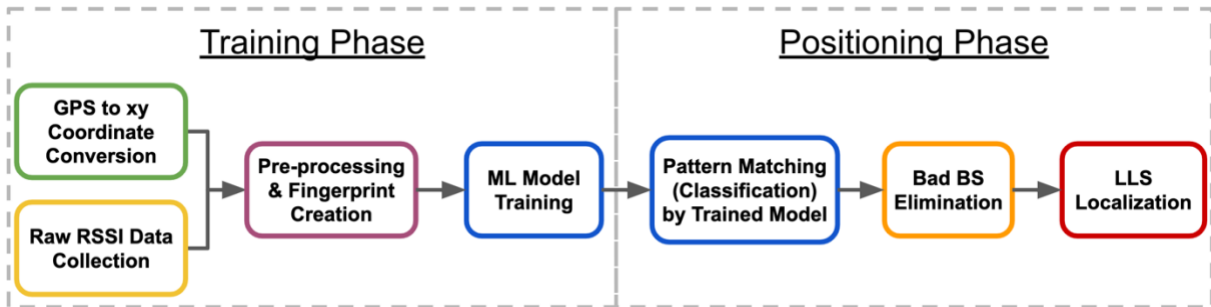


Figure 3.1 The overview of the localization algorithm.

The localization algorithm proposed in this project utilizes raw RSSI data, machine learning and LLS localization to estimate the position of the target. Figure 3.1 shows the overview of the localization algorithm. The details cover at “Structure of the Proposed Algorithm”. The algorithm can be roughly divided into two phases, training phase and positioning phase. In the training phase, the GPS of all the objects, like the base stations (BS, receivers) and the reference regions, in HK map, converts to (x, y) coordinates first. Also, a certain amount of raw RSSI datasets need to be collected at reference regions (class labels) and then need to be pre-processed before the training of the machine learning models. Each RSSI dataset, with the coordinates of the corresponding base stations, defines as the unique “radio fingerprint”. This project assumes that there are certain unique RF features in the fingerprints at different locations. Actually, we are building a radio map full of radio fingerprints at different locations. Under the

positioning phase, the algorithm performs “pattern matching” by the trained model on the new fingerprint to estimate the region of the target (transmitter). This new fingerprint collects at one of the reference regions without predetermined. Then, the algorithm removes the bad base station(s) (BS, the receiver) which RSSI values are out of range, and then performs LLS localization by the remaining good BS to calculate the location of the target. This proposed algorithm is a data-driven approach. Lots of RSSI data need to be collected and the data should be updated in the future if a large radio map is wanted to be built.

3.2 Proposed Coordination System on Hong Kong Map

A plane coordinate system is proposed when performing RSSI-based LLS localization. Latitude and longitude are the well-known representation of an object’s location. But it is not optimal for a small region to calculate the distance differences. Earth is an ellipsoid, and the surface is not flat. Because the latitude and longitude take the curved surface of the Earth into account, the calculated distance difference based on them is not as accurate as of reality. So, it is better to convert latitudes and longitudes (GPS) to plane coordinates on the map.

Figure 3.2 shows the proposed plane coordinate system. The point (GPS: 22.167603, 113.908512) at Tai a Chau Landing No. 2 is defined as the origin. This location is the most southwest island, the Soko Islands group, in Hong Kong. So, any other point in Hong Kong, for example, a point at Tsim Sha Tsui, results in a positive value x-coordinate and y-coordinate. It is convenient for calculation. Also, there are 71 base stations in Hong Kong according to the information from Sigfox.

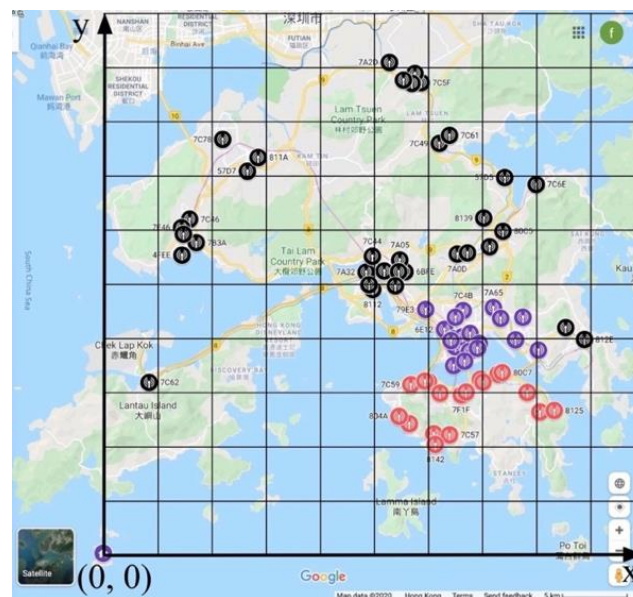


Figure 3.2 A proposed plane coordinate system.

3.3 Data Structure

Table 3.1 shows the data structure stored in the database. A space at one location is partitioned into L regions. Each region denotes a quantized location and represents a class C . In each region, there are N measurements. In each measurement, there is a set of received RSSI by the base stations (BS). Each set of received RSSI with coordinates of the corresponding base stations is a unique fingerprint of a known region. The sets of *RSSI fingerprint-region* pairs ($RSSI, C$) are the training datasets of the machine learning model. With the received RSSI and the GPS of each base station (fingerprint), the location of the target can be estimated by LLS after classification. There are P reference points in each region. The reference point is the location of the target to transmit Sigfox signals. The GPS of the reference points is used to investigate the performance of different localization algorithms.

Table 3.1 Data structure for data storage.

Region (Class Label)	Measurement	BS1	BS2	...	Reference Point
C_1 <i>region center = (x, y)</i>	1	$RSSI_{1,1}$	$RSSI_{1,2}$...	$v_1 = (x, y)$
	2	$RSSI_{2,1}$	$RSSI_{2,2}$...	$v_1 = (x, y)$
	3	$RSSI_{3,1}$	$RSSI_{3,2}$...	$v_2 = (x, y)$

	N	$RSSI_{N,1}$	$RSSI_{N,2}$...	$v_P = (x, y)$
...
C_L <i>region center = (x, y)</i>	1	$RSSI_{1,1}$	$RSSI_{1,2}$...	$v_1 = (x, y)$

	N	$RSSI_{N,1}$	$RSSI_{N,2}$...	$v_P = (x, y)$

3.4 Sigfox RSSI Data Collection, Storage and Checking System

Performing machine learning, data is the key. Certain amount of data needs to be first collected before training the machine learning models. Collecting certain amount of RSSI data, traditional methods, such as recording data by hand, are time-consuming because the data needs to be put back on the computer and to be tidy up before performing localization or training machine learning models. So, an automatic system for data is needed. In this project, a data collection, storage, and checking system are built.

System Overview

The below system is developed for data collection, storage and checking. The target (transmitter), the cloud-based data collection and storage platform, and the real-time localization estimation server are developed. The whole system aims to provide an automatic

solution to help collect and store a certain amount of RSSI data. It eliminates the need for manual data input and helps facilitate the efficiency of data collection.

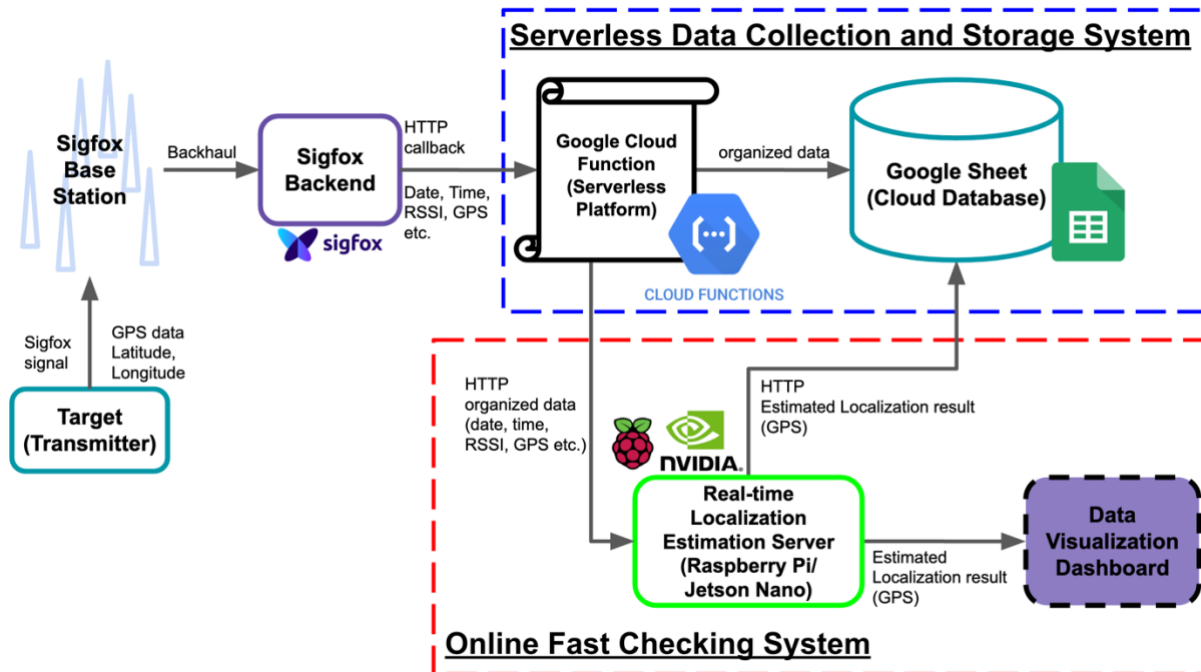


Figure 3.3 Block diagram of the data collection & storage and checking system.

Overall Operation

To collect the RSSI data between the base stations and the target (transmitter), the target should send a message to the Sigfox network. Also, GPS data is needed for the reference location. A hardware integrated with Arduino Mega 2560, the Sigfox transceiver (Xkit) and the 3G&GPS receiver (SIM5320E) is developed to transmit signals to the Sigfox network, with GPS values as message payload. Once the base stations in Sigfox network receive the message, a RSSI measurement is performed.

The RSSI values are then stored in the Sigfox backend server. The Sigfox backend server generates an HTTP callback [14], containing different information in JSON format, and sends the callback to a serverless platform, called “Cloud Functions”, that runs by Google. This cloud platform extracts needed data in the HTTP JSON message and stores the data in an organized manner in the cloud database (Google Sheet).

Another function of this cloud platform is to send RSSI related information to a self-construction “real-time localization estimation server”. This server constructs by Jetson Nano/Raspberry Pi. It sends back the estimated location of the target to the cloud database once the RSSI-based localization is finished.

Overall Design

Target

The target is a transmitter to help collect RSSI and GPS data by sending messages to Sigfox network. It is developed for data collection. Figure 3.4 and Figure 3.5 show the self-design PCB with all the modules assembled and the block diagram of the target, respectively. The microcontroller (Atmega2560) in the Arduino board, first requests GPS data from the 3G&GPS receiver (SIM5320E). It then sends the received GPS value to Sigfox network through the Sigfox transceiver shield (Thinextra “Xkit”).

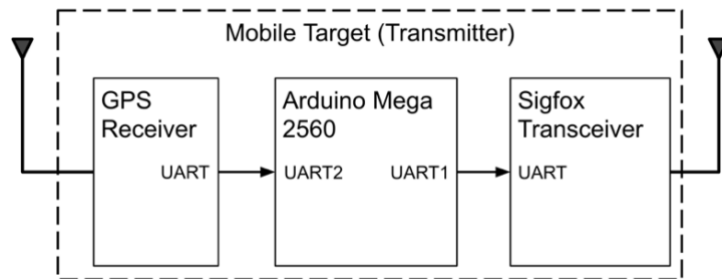


Figure 3.4 (Left) Arduino Mega 2560 with 3G&GPS module and Sigfox shield (Xkit).

Figure 3.5 (Right) Block diagram of the target for RSSI & GPS data collection.

Target – PCB Design

To design a printed circuit board (PCB), a schematic and a layout are the necessary drawings. The target shown above is designed in an online free website called EasyEDA, which is a circuit design tool and can run in a web browser. A schematic gives the electrical connectivity between the electronic components. It focuses on the logic and the function of a circuit without considering any physical factors. A layout is a drawing that shows the physical connections between electronic components and the physical placements of the components. In this PCB, the size and shape of the components and the width of the wire (e.g., signal traces, power traces, and ground traces) should consider carefully, so that the components can be well placed on the PCB and can be well connected to others.

Figure 3.6 shows the schematic of the shield for the target. There are different logical components, like Arduino Mega 2560, 3G&GPS receiver, LEDs, resistors, capacitors, and jumpers. Figure 3.7 and Figure 3.8 show the soft copy layout and physical layout of that shield,

respectively. This PCB is designed to connect the 3G&GPS receiver to the Arduino Mega 2560 development board to facilitate the data collection process.

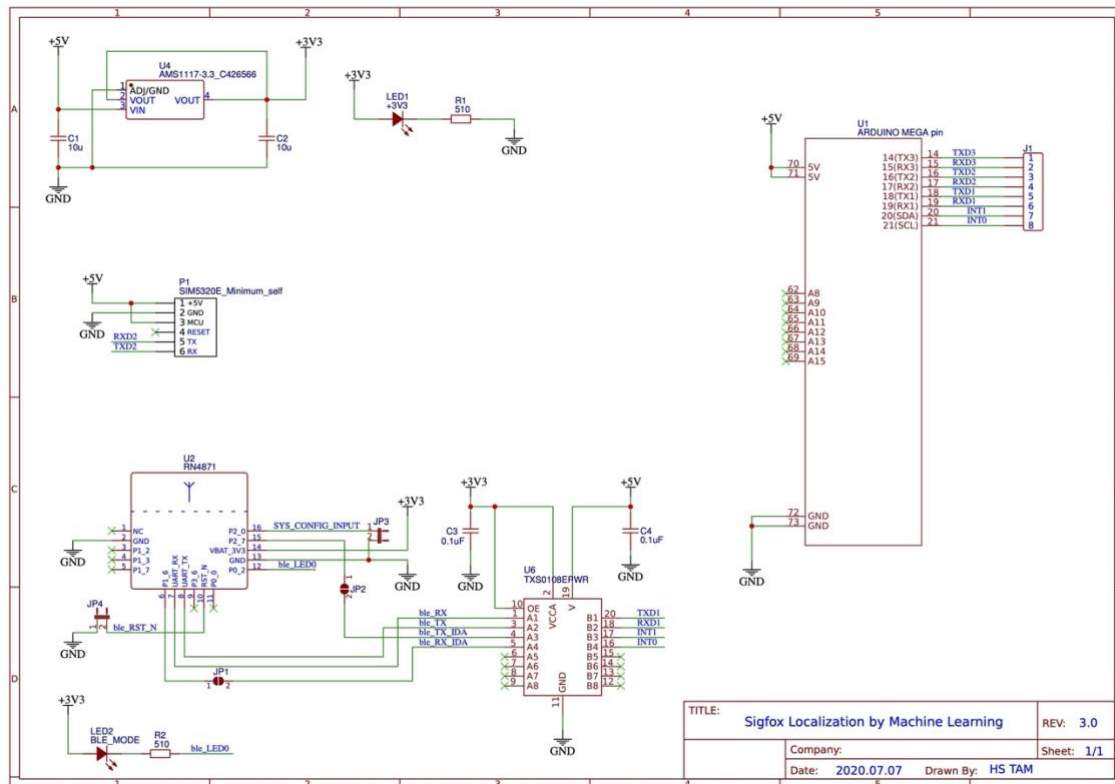


Figure 3.6 Schematic (circuit diagram) of the target.

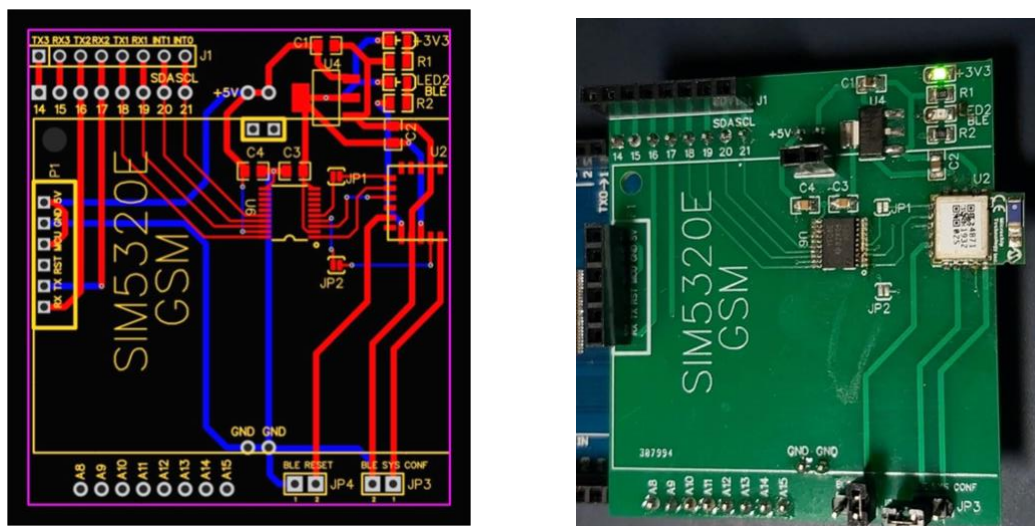


Figure 3.7 (Left) PCB layout drawing of the shield for the target.

Figure 3.8 (Right) Physical PCB of the shield for the target.

Target – Firmware

Arduino IDE is used to program the microcontroller in the target. Figure 3.9 and Figure 3.10 show the logic flow of the firmware, and the hardware trigger button, respectively. The

firmware enables the microcontroller to send GPS data to Sigfox network if the onboard button is pressed. The microcontroller checks the status of the onboard button every 0.3s. Once the button is pressed (longer than 0.3 seconds), the microcontroller requests GPS data from the 3G&GPS receiver through UART serial port and forwards it to Sigfox transceiver to send GPS data to the Sigfox network. If the button is not pressed over 6 seconds, the microcontroller requests the signal strength from the 3G&GPS receiver and prints it on the serial terminal if the microcontroller is connected to a PC. It is for debugging purposes.

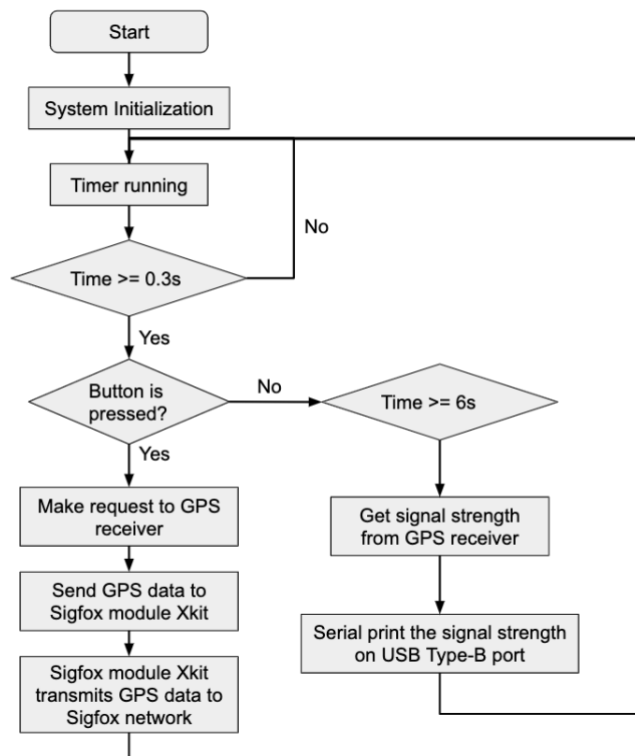


Figure 3.9 The logic flow of the firmware in Arduino Mega 2560.

Figure 3.10 The button on the Sigfox Xkit shield to trigger GPS data transmission.

Software Platform Design

The system shown in Figure 3.3 (circled in dotted blue) is the data collection and storage system. There are two components, a serverless platform (Cloud Functions) supported by Google Cloud Platform and a cloud database (Google Sheet). They enable a seamless data collection to the storage process without having to provision servers. Cloud Functions, is a serverless platform, allows event-driven web applications; for example, an HTTP request can trigger a specific script to run. The advantage of using this type of platform is serverless. Fewer maintaining efforts are needed, it runs without looking for a fixed IP address, and around-the-clock data storage service can be provided. A script is a small program that primarily designs for executing a specific task. It is like a “function” in a big program. It can only be triggered by one external

event. If a script is triggered by an event and is finished after execution, it never restarts. The only way to retrigger a script is to perform another external event. In this system, Cloud Functions is chosen as the cloud-based platform to hold the web application and to be executed once an HTTP request is sent from Sigfox backend. This serverless web application is written in Python programming language.

Figure 3.11 shows the logic flow of the web application. A callback is generated once the Sigfox backend server receives RSSI values from base stations. The callback is then forwarded to the web application by HTTP POST request. This HTTP POST request triggers the HTTP function on the web application and enables it to perform data extraction (from the HTTP request), data organization and data storage. The data, such as RSSI, GPS and information related to base stations, is stored at Google Sheet in an organized manner.

The web application also sends a copy of the organized data to the self-construction “real-time localization estimation server”. The design and the function of this server is covered at “Real-time Localization Estimation Server Design”.

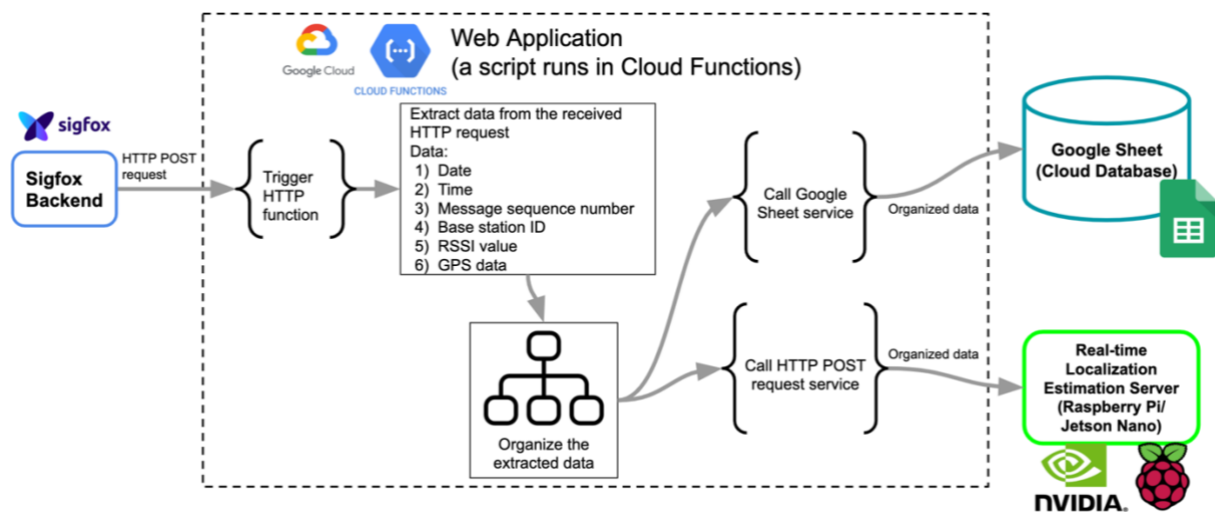


Figure 3.11 The logic flow of the web application running in Google Cloud Functions.

Real-time Localization Estimation Server Design

RSSI is to describe the total signal power received [9]. It is abstract. It is difficult to verify whether a set of RSSI values is reasonable or not when a person is collecting data in a location. So, a self-own online server is built for providing a data checking service. It allows a person to quickly verify the set of RSSI values by the real-time location estimation server.

This server runs in a small single-board computer, Jetson Nano or Raspberry Pi. It is a fully functional computer equipped with Wi-Fi, Bluetooth and Ethernet port, etc., but it is smaller in

size and consumes less power compared to notebooks and desktop computers. A lightweight web application framework coded in Python, called Flask, is adopted to build this location estimation server. Figure 3.12 shows the logic diagram of this server. Once the organized RSSI related data is prepared and is sent by the web application (in Google Cloud Functions) in HTTP request, the Flask's RESTful API is triggered. The server then performs RSSI-based LLS localization. Once the localization algorithm is finished, the estimated location result is sent to the Google Sheet for storage through an HTTP POST request. People who are collecting data in outdoor can check the estimated location in their Google Sheet mobile apps to quickly verify the set of RSSI values. Figure 3.13 and Figure 3.14 show the initialization and the result of the real-time localization estimation server.

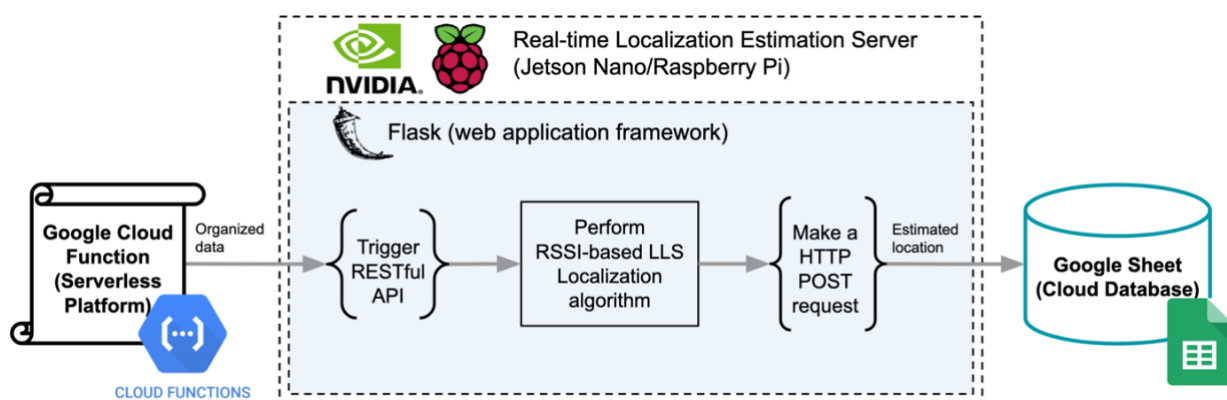


Figure 3.12 The logic flow of the server runs in Jetson Nano/Raspberry Pi.

```

(base) pi@raspberrypi:~$ python3 /home/pi/Desktop/fyp_rpi_server/flask_lls.py
* Serving Flask app "flask_lls" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://0.0.0.0:52836/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 304-824-219

```

Figure 3.13 The Jetson Nano/Raspberry Pi server is ready to receive HTTP requests.

```

33 2020/01/21 11:55:00 1200 79000 -121 7.27 ... Kowloon xxx
34 2020/01/21 11:55:00 1200 8142 -121 14.27 ... Hong Kong Island Ap Lei Chau
35 2020/01/21 11:55:00 1200 7C43 -87 27.09 ... Kowloon xxx
36 2020/01/21 11:55:00 1200 8043 -107 23.2 ... Kowloon xxx
37 2020/01/21 11:55:00 1200 7A4E -89 24.65 ... Hong Kong Island Central

[37 rows x 22 columns]>

=== LLS <BEGIN> ===
LLS xy: 26533.124367483277,17350.351005340282
LLS GPS: 22.32374848290821,114.16633948628865
=== LLS <END> ===
data:2020/01/21:time:11:54:10;cellNumber:1200;device:3E81CB;PathLossExponentAlpha;5.803;ReferenceRSSIZ0;-45;DeviceLL
viceLLSX;26533.124367483277;DeviceLLSY;17350.351005340282;DeviceGPSLng;0;DeviceGPSLat;0;LocalizationError;31647.6798
<Response [200]>
107.178.193.205 - - [21/Jan/2020 11:55:09] "POST /lls HTTP/1.1" 200 -

```

Figure 3.14 The result of the real-time localization estimation from the server.

3.5 Structure of the Proposed Algorithm

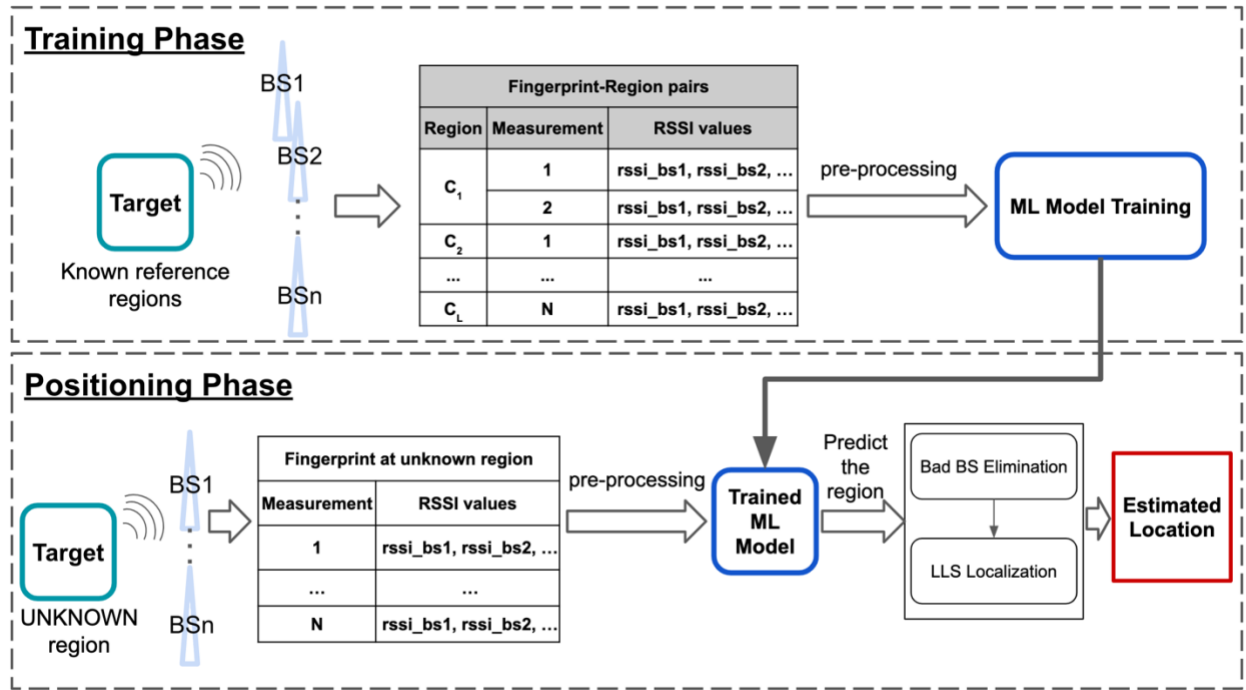


Figure 3.15 Detail structure of the localization algorithm.

Figure 3.15 illustrates the structure of the proposed algorithm in detail. The proposed algorithm is a data-driven approach. So, a certain amount of data needs to be collected. In the training phase, the raw RSSI datasets (fingerprints) should be obtained at predetermined known reference regions first. The target (transmitter) sends several messages to the Sigfox network at a reference region to collect several RSSI datasets. The RSSI datasets with the coordinates of the corresponding base stations define as the “radio fingerprints”. The fingerprints first undergo pre-processing. The pre-processing methods mention in the next chapter “RSSI Data Pre-processing”. By partitioning a space into 2D regions (e.g., Figure 5.1), we formulate the task as a classification problem. With the reference regions, it creates the fingerprint-region (**RSSI, C**) pairs. The pairs are the training datasets of machine learning. The training of machine learning can be understood as “a model is learning how to map each radio fingerprint to the regions with known GPS”. It is a supervised learning method and a classification task, so that a trained machine learning model can infer the region of the target from a new fingerprint.

In the positioning phase, a new fingerprint is collected at one of the reference regions without predetermination. After pre-processing of this fingerprint, the trained ML model predicts the possible region (class label) of the target by that fingerprint. Then, makes use of the predicted region of the target, the base stations with the out-of-range RSSI values, which define as bad base stations, are eliminated, because the out-of-range RSSI implies that the environmental factors (like blocking and multi-path) highly interfered with those base stations. The interfered

RSSI values would significantly degrade the accuracy of the LLS localization. After the elimination of the bad base stations, we perform LLS localization by the RSSI with the remaining good base stations and estimate the location of the target.

The reason why does not directly use the predicted region as the estimated location of the target is to reduce the great localization error induced by false classification if the radio map covers a large area in the future. Localization error refers to the distance between the estimated location and the real location of a target.

3.6 RSSI Data Pre-processing

The purpose of data pre-processing is to prepare the datasets for training and testing the machine learning models. Figure 3.16 shows the steps. The steps are the following: train/test data splitting, empty cell handling, data augmentation, and normalization. The details are covered below. However, this project does not adopt data augmentation, and the reason mentions in “Discussion”. In this project, three sets of raw RSSI data are collected on three different dates. Two-third (~70%) sets are split as raw training data, and the remaining one-third (~30%) is the raw testing data. Two split sets then undergo further processing individually.

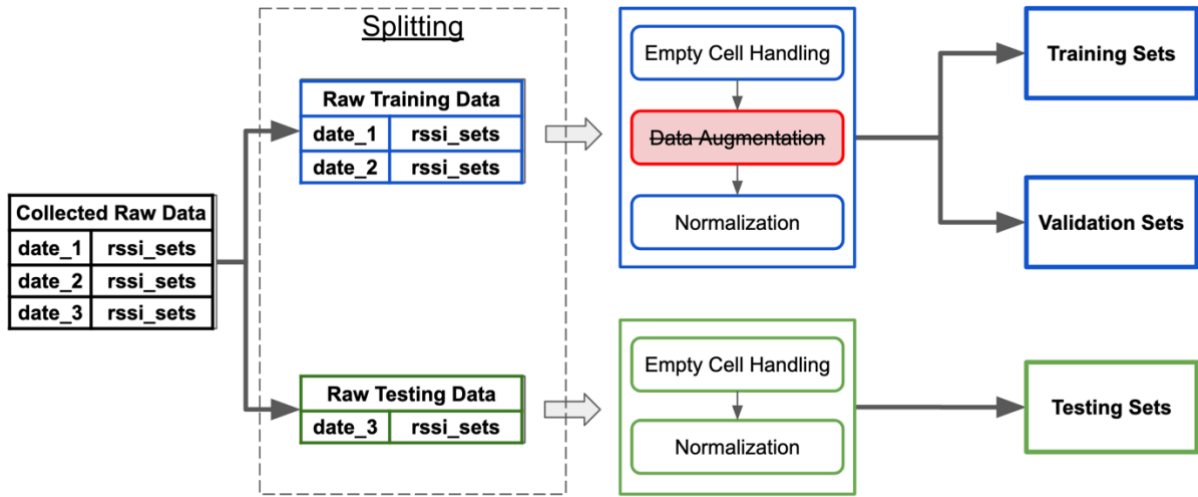



Figure 3.16 Steps in the data pre-processing.

Empty Cell Handling

The number of measurements from different base stations differs at each location. Some of the base stations cannot receive the signal from the target (transmitter); not all base stations can receive the signal from the target because that base stations are far away from it, and the signal strength degrades along with distance. So, the RSSI values of that base stations become “null”, and it is not accepted by the machine learning model. To deal with the empty values, the

“minimum received RSSI value insertion” approach is adopted. This approach performs the best within other approaches, “out-of-range value insertion” and “thresholding”, according to [15]. When no measurement is available from a base station, a minimum RSSI value is used. Figure 3.17 shows the before and the after.

Location	BS1	BS2	BS3	...
1	-97	null	-123	...
2	-117	-103	null	...
3	null	-121	-125	...
...



Location	BS1	BS2	BS3	...
1	-97	min	-123	...
2	-117	-103	min	...
3	min	-121	-125	...
...

Figure 3.17 Illustration of the way to deal with empty cells.

Data Augmentation on Training Data

According to [16], data augmentation is a common method for training deep learning models to reduce the chance of overfitting. It is usually adopted when the number of existing data is limited. Also, the experimental results of [16] find that their neural classifier achieves higher test accuracy with their proposed data augmentation scheme, compared to the classifier without applying data augmentation. The idea of data augmentation is to expand the existing dataset by using only the existing data so that the machine learning model can extract task-essential features more effectively. This project did try to adopt the “augmentation with the Mean and Uniform Random Numbers” scheme suggested by [16]. This scheme increases the number of data in a fashion such that the overall dataset remains in the range of minimum value and keeps the same mean with the original dataset. The following shows the steps of the scheme,

For each reference region,

1. Calculate the mean RSSI value for each base station (receiver)
2. Identify the range for the uniform random numbers, i.e.,

$$range \in [current\ RSSI, mean\ RSSI]$$

3. For each measurement,
 - a. Compare each RSSI value with the mean RSSI of each base station
 - b. Randomly selects a number from the range
 - c. Repeats N times so that one measurement generates N augmentation sets.

Figure 3.18 shows an example of data augmentation. We take the top-left corner RSSI value (= -70) in that measurement as an example. The mean of this base station is -60, and the current RSSI is -70 so that the range for the uniform random numbers of that base station is in [-70, -

60]. The number -66 is generated randomly from the range. So, the RSSI value of that base station in the new augmented dataset is -66. All other RSSI values of the base stations in that measurement are all augmented by the above steps. Repeats the above steps N times so that N augmentation sets can be generated by a measurement.

Figure 3.18 Original RSSI set vs. Augmented RSSI set of a measurement.

Source: [16]

However, in this project, data augmentation is not adopted because the machine learning model trained by augmented data performs worse than the model without data augmentation. Details are mentioned in “Discussion”.

Normalization

Normalization of the data for machine learning can improve training efficiency [17]. It is common to see that normalization appears in machine learning of image applications. Normalization applies in image applications to remove the variations not essential to the task and the variations caused by different contrast. In this project, the values of the RSSI are the negative numbers and can be very large in values, like -121 and -94. The large values may lead to gradient vanishing problem for non-linear activation functions like sigmoid function [11]. The effective x & y values of activation functions usually are in the range of 0 to 1. This project applies the min-max normalization in the training set and testing set individually, after empty cell handling, to scale the values in the range between 0 and 1. So, the normalized RSSI value, \hat{Z} , is

$$(3.1) \quad \hat{Z} = \frac{Z - Z_{min}}{Z_{max} - Z_{min}}$$

where Z is the original RSSI value, Z_{min} and Z_{max} are the minimum and maximum RSSI value respectively.

4 Discussion

Although data augmentation can increase the number of datasets and may reduce the chance of overfitting for machine learning according to [16], this project tests that the accuracy of

classifiers with augmented data is worse than the classifiers without augmented data. In this project, 1,336 measurements are collected in 4 different regions. 1,246 measurements and 90 measurements are split to the training set and testing set respectively. After data augmentation on the training set, each measurement is augmented to 60 datasets. So, the total number of the augmented training set can increase to 76,006 from 1,246. This augmented training set is then used to train the SVM, DNN and 1D-CNN classifiers. Table 4.1 shows the accuracy of the classifiers trained with and without augmented data. The accuracies of the classifiers with augmented data are lower than those without data augmentation about 2~10%. It implies that, in this project, the data augmentation approach does not help the machine learning classifiers learn about the essential features from the radio fingerprints. Instead, the data augmentation method produces unmeaningful/strange features that violate the real environment or produces more noises to the training data.

Table 4.1 Experimental results of the classifiers with and without data augmentation.

Machine Learning Classifier	Average Accuracy without Data Augmentation	Average Accuracy with Data Augmentation
SVM with linear kernel	75.556 %	73.333 %
DNN	77.722 %	74.833 %
1D-CNN	88.148 %	77.777 %

5 Experiments and Results

5.1 Locations to Collect Sigfox RSSI Dataset

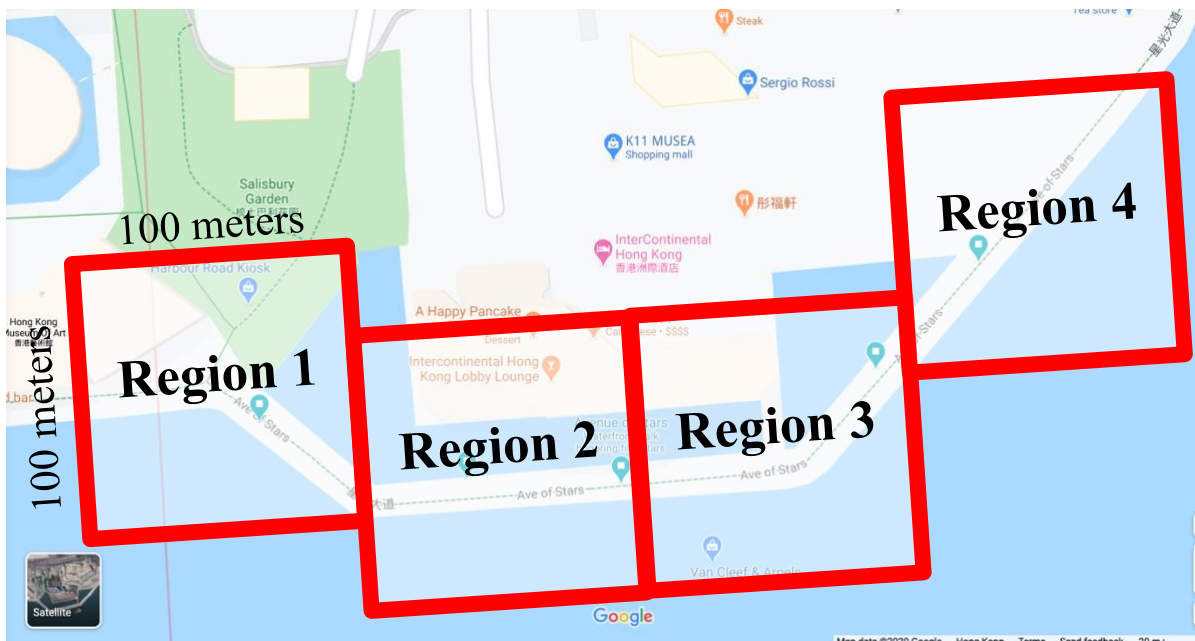


Figure 5.1 The location to collect data is in Avenue of Stars in Tsim Sha Tsui, Hong Kong.

This project collects 1,336 raw RSSI datasets manually in a space, Avenue of Stars in Tsim Sha Tsui, Hong Kong. This space is divided into 4 regions (i.e. class labels for machine learning) in size of 100m x 100m and each region represents a class. Figure 5.1 shows the locations of this space and the regions. The raw RSSI datasets are collected on 3 different dates (2rd Jan, 17th Feb & 16th Mar 2020) on 4 regions. The number of 90, 46, and 1200 datasets collects on the first, second, and last day respectively. The number of measurements over regions is shown in Figure 5.2. Each measurement collects at a random point within a region.

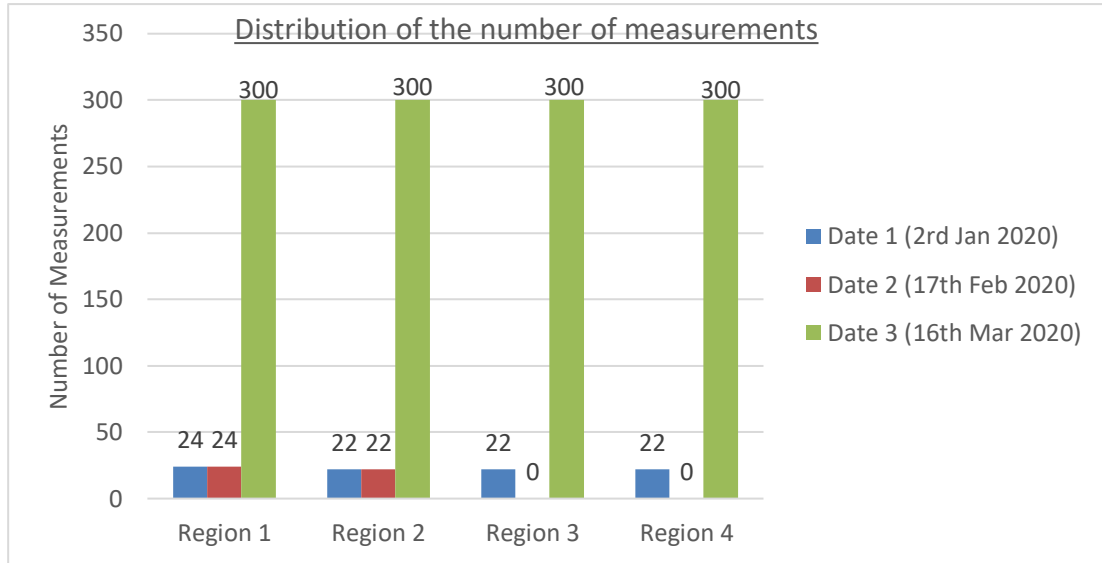


Figure 5.2 Distribution of the number of measurements.

5.2 The Dataset

The raw RSSI datasets are split into the training and testing sets according to dates. The reason is to totally isolate the training data from the testing data, which can ensure the machine learning model only learns from the training data to extract RF features. It also simulates the future situation. When a future user requests a localization service, the new RSSI fingerprint is collected at an unknown location, and this fingerprint has never appeared in the training data yet. Splitting data by dates can truly test the performance of the machine learning classifiers. Figure 5.3 shows the distribution of the number of the training and testing sets. There are 1,246 training sets (Date 2 & Date 3) and 90 testing sets (Date 1). The dimension of each training/testing set is 71-dimension which is same as the number of base stations in Hong Kong.

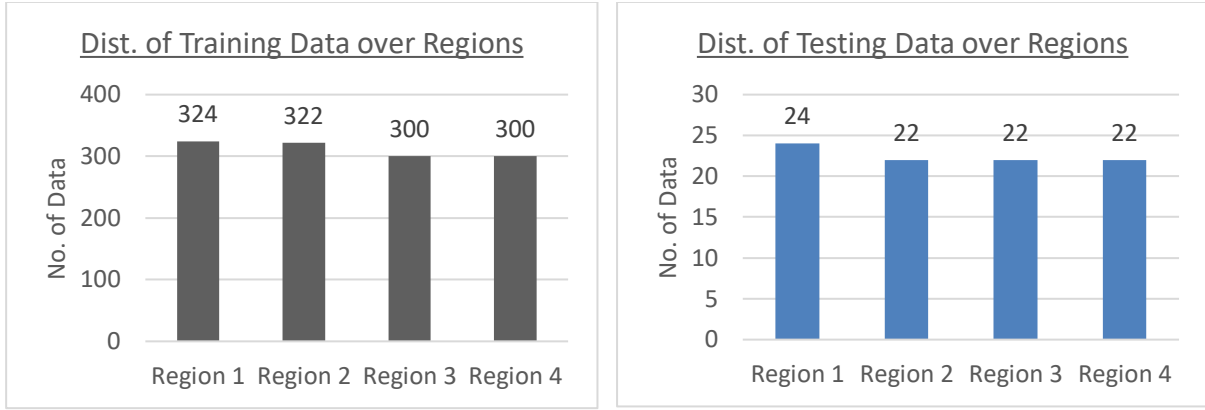


Figure 5.3 Distribution of the training and testing sets.

5.3 Characteristics of Sigfox RSSI

The RSSI value in outdoor environments is difficult to predict precisely because of factors like blocking and multipath. Also, RSSI value at a point varies with time due to environmental noise. Figure 5.4 shows the variation of the RSSI values collected at a point on Region 2 from two base stations over 30 minutes. There are 100 measurements each on two base stations. The time between each measurement takes around 18 seconds. For base station “8043”, although the mean RSSI is -109.53 dBm and the standard deviation is 2.59547 dBm, the minimum and maximum RSSI are -119 and -106 dBm respectively. They difference 13 dBm.

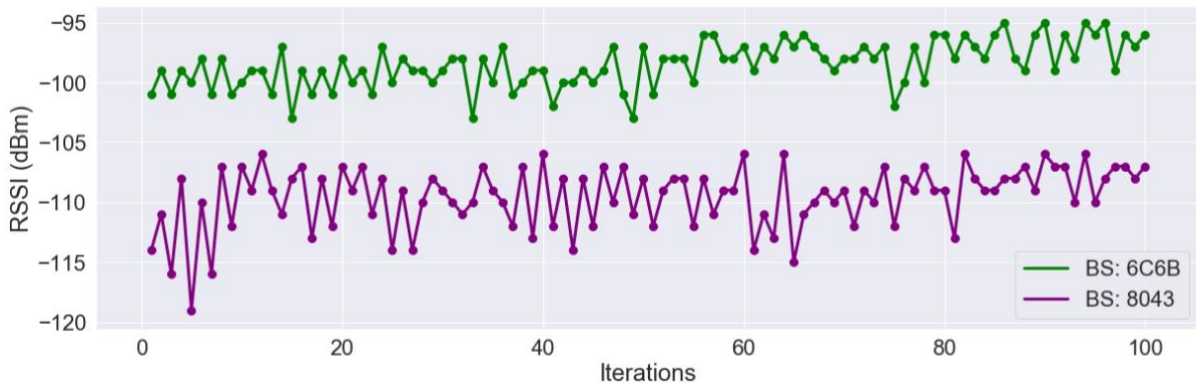


Figure 5.4 RSSI values over time.

However, according to [18], the distributions of the RSSI are believed to follow a log-normal distribution about the mean. Figure 5.5 shows the distributions of RSSI values collected from two base stations in this project. These two distributions are the common shapes of the RSSI distributions of other sixty-nine base stations. There are 71 base stations in Hong Kong, according to the information from Sigfox. Although the shape of the RSSI distribution in this project does not exactly follow log-normal distribution, it is close.

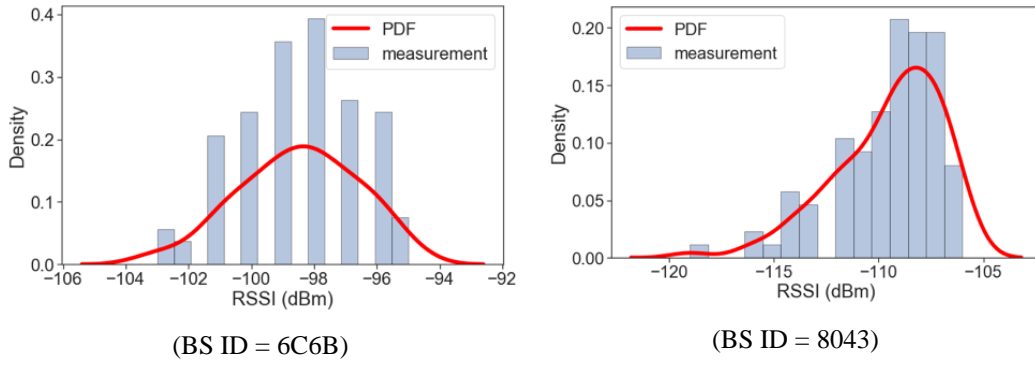


Figure 5.5 The distribution and histogram of RSSI values of two base stations.

Also, Figure 5.6 shows the RSSI decreases nearly linearly with distance, which indicates that RSSI represents the distance to a certain degree, and LLS localization is effective in estimating the target's location from the received RSSI values. This dataset collected at Hung Hom Promenade on 18th March 2019 by the previous year FYP students.

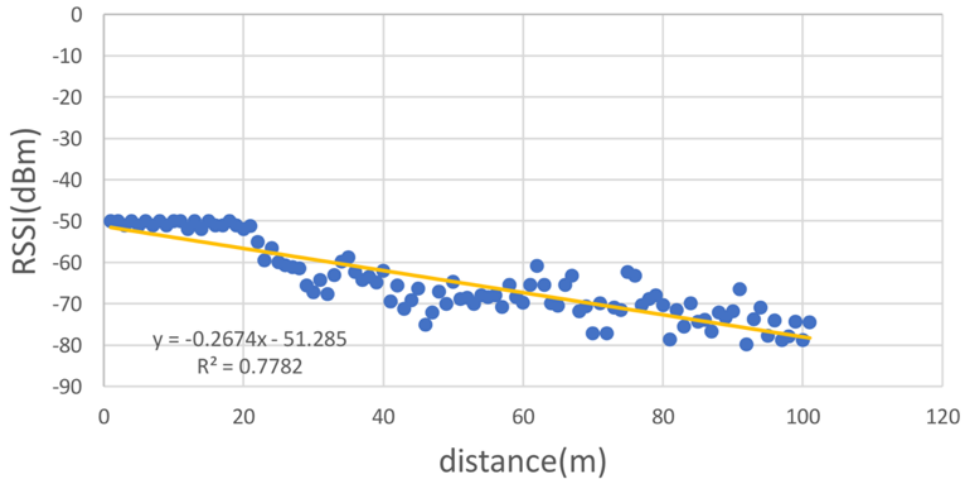


Figure 5.6 The relationship between RSSI values and distances.

Sources: Ki Fung Leung, Localization by using SigFox Technology. 2019.

5.4 Assessment Methods to Classifiers

Classification accuracy and localization accuracy are employed to assess the performance of the classifiers with LLS localization. Classification accuracy shows how well the machine learning classifiers can correctly infer the region from the received RSSI values.

Localization error (LE) are the criteria of localization accuracy. After classification by classifiers, the predicted region (class label) is outputted. The bad base stations can be identified and eliminated by calculating the expected RSSI between the base stations and the predicted region. If the value of the received RSSI of a base station is much smaller (or higher in some chances) than the expected RSSI calculated by the distance between the predicted region and

that base station, this base station is identified as a bad base station and would not involve in the LLS localization. After LLS localization by the remaining good base stations, the target's location is estimated. Localization error (LE) is the distance between the estimated target's location and the real target's location. The smaller the value of the localization error is, the better the algorithm is. Mean localization error (LE_{mean}) over all measurements would be calculated to further assess the classifiers with LLS localization.

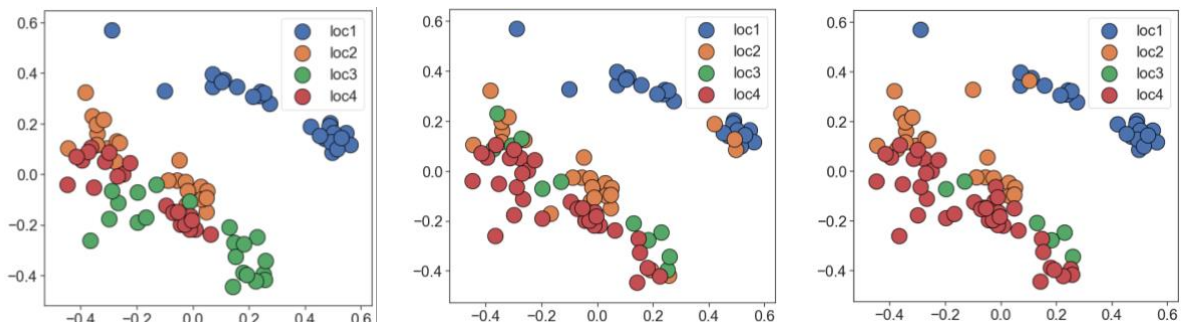
5.5 Performance of SVM Classifiers

SVM Classification Accuracy

The project uses three kernel functions, linear, RBF, and polynomial kernel functions, to train the SVM classifiers. Table 5.1 shows the performance of the SVM classifiers with different kernels. The SVM classifiers with both linear and RBF kernels perform the best and achieves 75.556 % accuracy on testing data. Figure 5.7 shows the comparison of the original testing data and the outputs of linear-SVM and RBF-SVM in 2D space by the Principal Component Analysis (PCA) dimension reduction. Although linear-SVM and RBF-SVM achieve the same accuracy, there are slight differences in the confusion matrices shown in Figure 5.8, which shows that the classification abilities of two SVM classifiers are slightly different. However, the polynomial kernel fails to perform classification on the RSSI testing sets (Figure 5.8 (c)) because it predicts all RSSI testing sets as Region 1. After training, the SVM classifier predicts the region (class label) from the RSSI of the testing data to eliminate the bad base stations. We found that linear and RBF kernels perform the best in SVM.

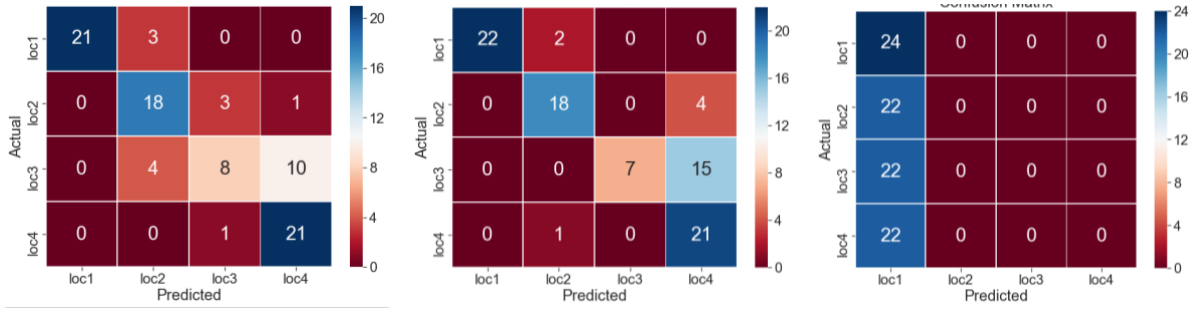
Table 5.1 The accuracy of SVM classifiers with different kernel functions.

Kernel Function	Training Accuracy	Testing Accuracy
Linear	99.117 %	75.556 %
RBF	100.000 %	75.556 %
Polynomial	26.003 %	26.667 %



(a) Original test data by PCA dim. reduction (b) Output of SVM linear kernel by PCA dim. reduction (c) Output of SVM RBF kernel by PCA dim. reduction

Figure 5.7 Comparison of the results of SVM with different kernels.



(a) Output of SVM linear kernel (75.556% accuracy) (b) Output of SVM RBF kernel (75.556% accuracy) (c) Output of SVM poly. kernel (26.667% accuracy)

Figure 5.8 Confusion matrix of the results from SVM with different kernels.

SVM Localization Accuracy

After the elimination of bad base stations, the number of base stations in each testing set decreases to 3~7 from 71. Also, the number of testing sets reduces to 48 from 90. The reason is that all base stations on the deleted testing sets are identified as bad base stations, and all are eliminated. Table 5.2 shows the statistical analysis of LE. On the 48 testing sets, the LE_{mean} of LLS-SVM is 543.152 meters; the minimum LE is 72.233 meters on Region 3 while the maximum LE is 2755.620 meters on Region 4. Table 5.11 shows the best estimated location with minimum LE over four regions by LLS-SVM. Also, Figure 5.11 shows the performance comparison of the LLS-SVM and pure LLS localization on Google Maps.

Table 5.2 The statistics of the localization error of LLS-SVM.

Measurement count	Mean LE (m)	Standard deviation LE (m)	Min LE (m)	Q1 LE (m)	Q2 LE (m)	Q3 LE (m)	Max LE (m)
48	543.152	625.950	72.233	231.625	332.474	537.280	2755.620

5.6 Performance of DNN Classifiers

DNN Classification Accuracy

The structure of the DNN is shown in Table 5.3. The DNN consists of an input layer, hidden layers, and a soft-max output layer. The input dimension is as large as the total number of base stations in HK, which is 71. There are L hidden layers and N nodes in each hidden layer. Each hidden node equips with the rectified linear unit (ReLU) activation function. The soft-max layer outputs the posterior probabilities of four individual class labels (regions). So, the predicted class label (region) can be determined by the largest value of posterior probability. The DNN

is trained by using Adam algorithm with learning rate 0.001, momentum parameter 0.9, and mini-batch size 50.

Figure 5.9 shows the relationship between the accuracy and the number of hidden nodes. So, the optimized number of nodes N (in each hidden layer) can be found. The average accuracy is calculated from ten DNN classifiers with the same number of hidden nodes N in a hidden layer. It shows that the 100-node DNN classifier achieves the best performance (76.111%) while the 175-node DNN classifier achieves the second-best performance (75.611%). The accuracy keeps around seventy-something percent until 1000 nodes and starts degrading after 1000 nodes.

Table 5.4 shows how the accuracy changes with the number of hidden layers with 100 nodes each so that the optimized number of hidden layers can be found. The performance is the best at 4-hidden-layer DNN and is the poorest at 2-hidden-layer DNN. The accuracy then degrades if the number of the hidden layers is more than 4. So, we found that the DNN classifier performs the best with 4 hidden layers ($L_{best} = 4$) and each layer with 100 nodes ($N_{best} = 100$).

Table 5.3 Structure of the Deep Neural network.

Raw RSSI input layer with 71 nodes
Hidden layer (N ReLU nodes)
...
Hidden layer (N ReLU nodes)
Soft-max output layer (4 dimensional)

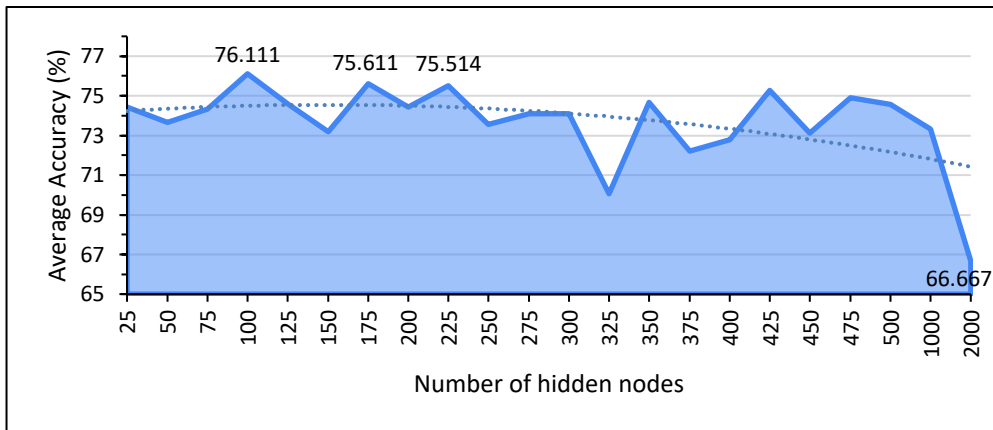


Figure 5.9 Accuracy of the DNN classifier changes with the number of hidden nodes.

Table 5.4 Performance comparison of DNN with different number of hidden layers.

No. of hidden layers (100 nodes each)	Average Accuracy
1, [71 100 4]	76.111 %
2, [71 100 100 4]	73.500 %
3, [71 100 100 100 4]	77.000 %

4, [71 100 100 100 100 4]	77.722 %
5, [71 100 100 100 100 100 4]	75.778 %
9	73.778 %

DNN Localization Accuracy

The number of testing sets reduces to 50 after the elimination of bad base stations. Table 5.5 shows the statistical analysis of LE. The LE_{mean} of LLS-DNN is 667.383 meters; the minimum LE is 72.233 meters on Region 3 while the maximum LE is 6666.681 meters on Region 2. Table 5.11 shows the best estimated location with minimum LE over four regions. Also, Figure 5.11 shows the performance comparison of the LLS-DNN and pure LLS localization on Google Maps.

Table 5.5 The statistics of the localization error (LE) of LLS-DNN.

Measurement count	Mean LE (m)	Standard deviation LE (m)	Min LE (m)	Q1 LE (m)	Q2 LE (m)	Q3 LE (m)	Max LE (m)
50	667.383	1063.854	72.233	209.160	352.001	559.353	6666.681

5.7 Performance of 1D-CNN Classifiers

1D-CNN Classification Accuracy

Table 5.6 presents the architecture of the 1D-CNN. The 1D-CNN consists of an input layer, 1D convolutional layers, 1D max-pooling layers, dropout layers, a flatten layer, fully connected layers, and a soft-max output layer. The input dimension is as large as the total number of base stations in HK, which is 71. A convolution block is composed of a 1D-convolution layer, a 1D max-pooling layer and a dropout layer. The dropout layer is employed to prevent overfitting. There are L convolution blocks, and each convolution layer contains N filters in size $= S$. The fully connected layer acts as the classifier. The number of 128 nodes with ReLU present in this layer. The soft-max layer outputs the posterior probabilities of four individual class labels (regions). Table 5.7 shows the list of the filter size S , the number of filters N and the number of convolution blocks L , and their candidate values. By tuning the hyperparameters S , N , and L , the 1D-CNN classifier can achieve the optimal performance. The average accuracy of each value in each hyperparameter is calculated from the accuracy of the five 1D-CNN classifiers. When tuning the value of S , the value of N is 32 and the value of L is 1. When tuning the value of N , the values of S and L are 10 and 1 respectively. When tuning L , the values of S and N are 4 and 128 respectively.

We found that the 1D-CNN classifier performs the best when it consists of 1 convolution block ($L = 1$) and each convolution layer contains 128 filters ($N = 128$) in size $S = 10$.

Table 5.6 Structure of the 1D Convolution Neural Network.

Raw RSSI input layer	71 nodes
1D-Conv layer	N filters, filter size = S
1D max-pooling layers	Pool size = 2
Dropout layer	25% drop rate
...	
1D-Conv layer	N filters, filter size = S
1D max-pooling layers	Pool size = 2
Dropout layer	25% drop rate
Flatten layer	
Fully connected layer	128 nodes
Dropout layer	25% drop rate
Soft-max output layer	4 nodes

Table 5.7 Hyperparameter tuning for 1D-CNN classifier

Hyperparameters	Values	Average Accuracy
Filter Size, S	4	83.333 %
	7	82.222 %
	10	86.222 %
	12	84.000 %
	15	84.222 %
	20	80.667 %
Number of Filters, N	32	86.222 %
	64	86.222 %
	128	88.148 %
	256	85.556 %
Number of convolution blocks, L	1	80.000 %
	2	79.111 %
	3	68.889 %
	4	70.222 %

1D-CNN Localization Accuracy

The number of testing sets reduces to 49 after the elimination of bad base stations. Table 5.8 shows the statistical analysis of LE. The LE_{mean} of LLS-1DCNN is 676.368 meters; the minimum LE is 72.233 meters on Region 3 while the maximum LE is 6666.681 meters on

Region 2. Table 5.11 shows the best estimated location with minimum LE over four regions. Also, Figure 5.11 shows the performance comparison of the LLS-1DCNN and pure LLS localization on Google Maps.

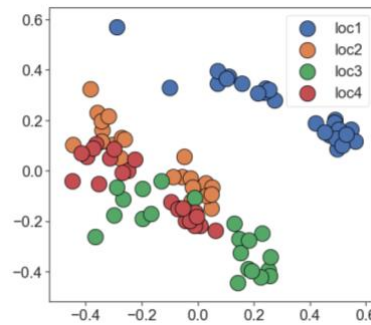
Table 5.8 The statistics of the localization error (LE) of LLS-1DCNN.

Measurement count	Mean LE (m)	Standard deviation LE (m)	Min LE (m)	Q1 LE (m)	Q2 LE (m)	Q3 LE (m)	Max LE (m)
50	676.368	1072.828	72.233	214.721	338.289	561.195	6666.681

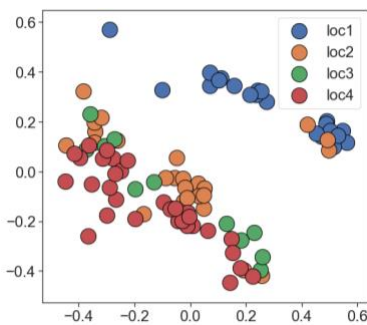
5.8 Discussion and Comparison

Accuracy Comparison of Classifiers

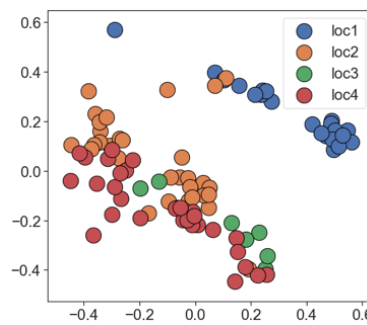
Table 5.1 from SVM shows that linear kernel performs the best; Table 5.4 from DNN suggests that the structure of 4 hidden layers with 100 nodes each achieves the best performance; Table 5.7 from 1D-CNN shows that the structure of 1 convolution block and 128 filters with filter size = 10 in each convolution layer performs the best. By comparing three classifiers with their optimal setting, Figure 5.10 and Table 5.9 suggest that 1D-CNN possesses higher ability to extract the RF features and can infer the corresponding region (class label) from the testing sets more accurately than linear SVM and DNN. The accuracy of 1D-CNN is approximately 10% higher than linear SVM and DNN.



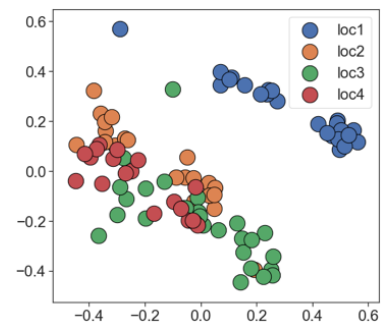
(a) Original test data



(b) Output of linear SVM (75.56% acc.) by PCA dim. reduction



(c) Output of DNN (77.78% acc.) by PCA dim. reduction



(d) Output of 1D-CNN (88.89% acc.) by PCA dim. reduction

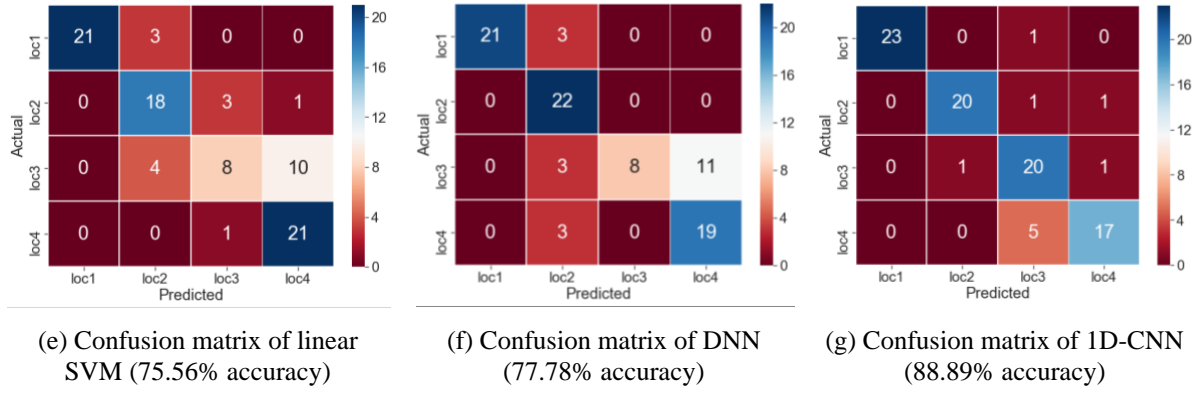


Figure 5.10 Performance comparison between linear SVM, DNN and 1D-CNN. The output plots show the results of each classifier in 2-dim space whose dimensions are reduced from 71 by principal component analysis (PCA). The accuracies shown here are close to their individual best average accuracy.

Table 5.9 Comparison of the best average accuracy of different classifiers.

	SVM with linear kernel	DNN	1D-CNN
Best Average Accuracy	75.556 %	77.722 %	88.148 %

Localization Error Comparison

Although the three classifiers (SVM, DNN and 1D-CNN) are performing the same classification task, their distributions of the localization error (LE) are different. Table 5.2, Table 5.5, and Table 5.8 show the differences in LE distribution between three classifiers. The reason of the LE differences is that the three classifiers possess different classification accuracies so that their predicted regions (class labels) on the same testing set are different, and their eliminated bad base stations are different, which finally causes different estimated locations of the target by LLS localization. Although the LE_{mean} of LLS-SVM is the lowest in this project according to Table 5.10, its classification accuracy is the lowest. Indeed, the classification accuracy of the classifiers is much more critical in future use. Especially in the future, when the coverage space is much larger than that in this project, a classifier with lower classification accuracy would induce more misclassifications and the misclassified region (class label) might be much far away from the real target's location, which causes that the base stations would be eliminated incorrectly and leads to larger LE. Table 5.11 shows the best estimated locations (with min. LE) by LLS-SVM, LLS-DNN and LLS-1DCNN.

Table 5.10 Mean localization error LE_{mean} comparison of pure LLS localization, LLS-SVM, LLS-DNN and LLS-1DCNN.

	Pure LLS	LLS-SVM	LLS-DNN	LLS-1DCNN
Mean Localization Error (meters)	2385.939	543.152	667.383	676.368

Table 5.11 Estimated locations and the min. LE by LLS-SVM, LLS-DNN or LLS-1DCNN over 4 regions.

Region	Real target's GPS (lat, lng)	Estimated target's GPS (lat, lng)	Min. Localization Error
1st	22.293210, 114.172877	22.293125, 114.173786	94 meters
2nd	22.293025, 114.173606	22.292127, 114.173780	101 meters
3rd	22.293381, 114.175046	22.292768, 114.174813	72 meters
4th	22.29373, 114.175408	22.292821, 114.175811	109 meters

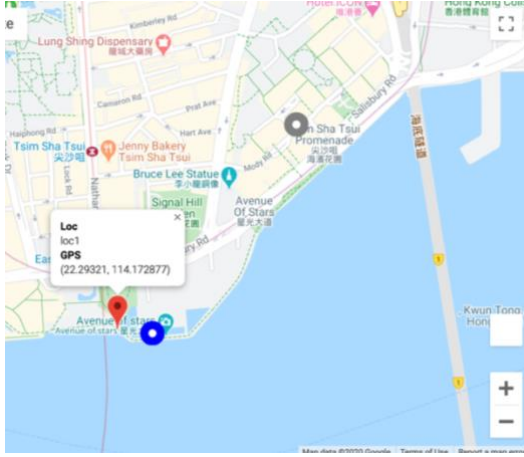
Table 5.12 shows the statistics of the LE of pure LLS localization. Table 5.13 and Figure 5.11 compare the performances of the pure LLS localization and the LLS localization with machine learning. It is easy to see that the minimum LE of LLS with machine learning over four regions is about 100 meters, while the minimum LE of pure LLS over four regions is at least 741 meters. Also, by comparing the statistics of LE from LLS-SVM & LLS-DNN & LLS-1DCNN (Table 5.2, Table 5.5 and Table 5.8) to that from pure LLS localization (Table 5.12), the LE of LLS with machine learning is statistically much lower than that of pure LLS localization. They all indicate that the performance of the LLS with machine learning surpasses the pure LLS localization.

Table 5.12 The statistics of the localization error (LE) of pure LLS localization.

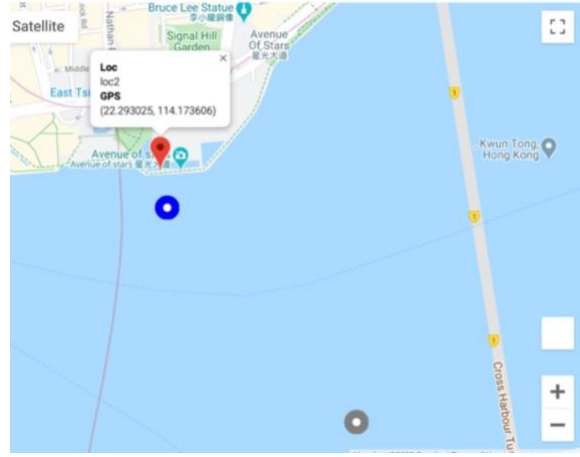
Measurement count	Mean LE (m)	Standard deviation LE (m)	Min LE (m)	Q1 LE (m)	Q2 LE (m)	Q3 LE (m)	Max LE (m)
50	2385.939	1099.127	740.721	1841.225	2318.825	2676.980	7959.107

Table 5.13 Localization error comparison of the pure LLS localization and LLS with machine learning.

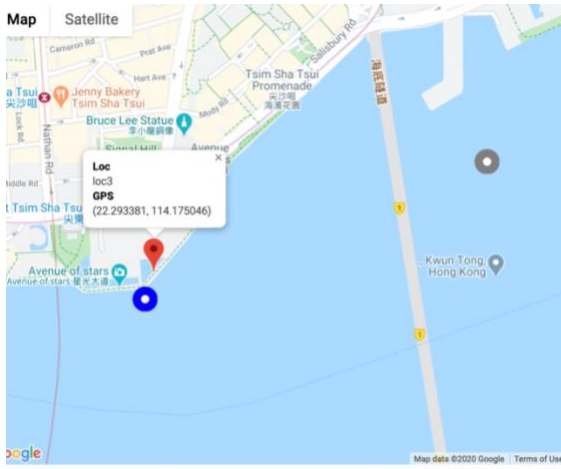
Region	Real target's GPS	LE of pure LLS		LE of the LLS with ML	
		Min. (meters)	Max. (meters)	Min. (meters)	Max. (meters)
1st	22.293210, 114.172877	741	5915	94	2581
2nd	22.293025, 114.173606	867	7959	101	6666
3rd	22.293381, 114.175046	934	3151	72	1584
4th	22.29373, 114.175408	1657	3400	109	2755



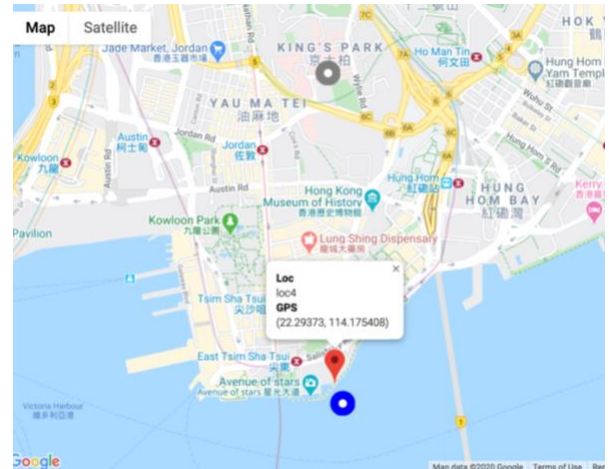
(a) On Region 1



(b) On Region 2



(c) On Region 3



(d) On Region 4

Figure 5.11 Result comparison of the pure LLS and LLS-SVM or LLS-DNN or LLS-1DCNN. The red marker with a dialog box is the real location. The grey dot is the estimated location by pure LLS while the blue dot is the estimated location by LLS-SVM, LLS-DNN or LLS-1DCNN.

6 Conclusion and Future Developments

6.1 Conclusion

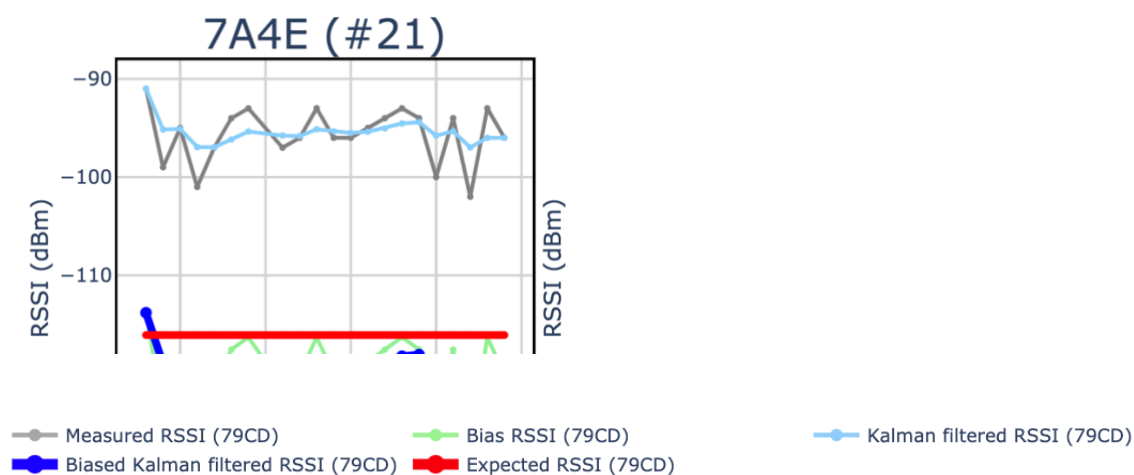
All in all, from the above system and experiments, this project shows that machine learning is a potential tool to help RSSI-based localization become more accurate. The tables and figures from the results of the experiment show that the localization error is reduced effectively by LLS with machine learning. Also, the performance of 1D-CNN in RSSI classification is the best while the LLS-SVM is the best in the performance of localization.

6.2 (X) Future Developments

(X) Noise Reduction by Kalman Filter

Kalman filter can smoothen the RSSI in time series.

- See picture below, it shows the RSSI variation of a BS over 22 measurements.
 - just focus on grey line, 淺藍色 line and red line (ignore green line and deep blue line)
 - must look at the legends at the bottom of this picture
- Can see that grey line (= raw collected RSSI) is smoothened to 淺藍色 line (= noise reduction)



(X) Increase the Resolution of Reference Regions

= collect more data in the same space, i.e. just in the above 4 regions (Avenue of Stars in Tsim Sha Tsui, Hong Kong), but partitions each region into sub-regions in smaller size (smaller than 100x100 meter)

= only increases the number of regions but keeps the coverage area as same as the area used in this project

(X) Increase the Number of Reference Regions

= increases the coverage area, i.e. collects RSSI data in not only Avenue of Stars but also other nearby places

= increases both coverage area and the number of regions (each region is still in the same size, i.e. 100x100 meter)

(X) Add another feature, like SNR (or CSI if it is possible), to train the ML model

1. Signal-to-noise ratio (SNR), Sigfox backend already provided RSSI with SNR.
2. According to [13], Channel State Information (CSI) fingerprints perform better than RSSI fingerprints. But Sigfox backend has not provided any CSI values/info to the users yet.

DNN-RSSI \leq 81.25% accuracy

VS CNN-RSSI \leq 82.93% accuracy

DNN-CSI = avg. 99% accuracy

VS CNN-CSI = avg. 99% accuracy

C.-H. Hsieh et al.: Deep Learning-Based Indoor Localization Using RSS and CSI

TABLE 1. Comparison of MLP architectures using RSS signal.

MLP Architectures	Total Number of Parameters	Prediction Accuracy(%)
360 x 4	412577	67.94
360 x 5	542176	78.26
720 x 5	2121136	81.25
960 x 5	3749776	80.53
720,620,520,420,320	1161736	77.37
960,860,560,460,360	1782576	81.07
960,860,560,460,360,360	1912897	6.92
960,860,560,460,360,360,360	2042857	9.07

TABLE 2. Comparison of MLP architectures using CSI signal.

MLP Architectures	Total Number of Parameters	Prediction Accuracy(%)
360 x 4	460816	99.43
360 x 5	590776	99.54
720 x 5	2218336	99.80
960 x 5	3879376	99.61
720,620,520,420,320	1258936	99.83
960,860,560,460,360	1912176	99.93
960,860,560,460,360,360	2042136	99.68
960,860,560,460,360,360,360	2172096	99.61

C.-H. Hsieh et al.: Deep Learning-Based Indoor Localization Using RSS and CSI

TABLE 3. Comparison of CNN architectures using RSS signal.

Kernel sizes	Number of filters	Total Number of Parameters	Prediction Accuracy(%)
26	16-32	38465	70.68
10	16-32-32	38976	80.65
16	16-32-32	48288	82.92
26	16-32-32	63808	82.32
36	16-32-32	79328	79.94
26	16-32-32-32	92033	63.46
26	16-32-32-32-32	118817	67.10
26	16-32-32-32-32-32	145601	66.38

TABLE 4. Comparison of CNN architectures using CSI signal.

Kernel sizes	Number of filters	Total Number of Parameters	Prediction Accuracy(%)
26	16-32	106144	99.04
10	16-32-32	108096	99.79
16	16-32-32	117408	99.93
26	16-32-32	132928	99.97
36	16-32-32	148448	99.97
26	16-32-32-32	159712	99.96
26	16-32-32-32-32	196496	99.90
26	16-32-32-32-32-32	213280	99.82

(X) Localization as Regression by Deep Learning Models

The above system and experiments show the potentials of applying machine learning into RSSI-based localization. However, the usages of machine learning are far beyond classification tasks. Instead, the deep learning models, DNN and 1D-CNN, can be designed to perform regression. For example, to design a DNN to perform regression tasks rather than classification tasks, the last output layer should be changed to 2-node layer instead of a soft-max node. This 2-node layer can estimate the location of the target by the RSSI fingerprints in the format of (x, y) coordinates or (latitude, longitude) GPS values [11]. Figure 6.1 shows one of the potential designs for the regression DNN.

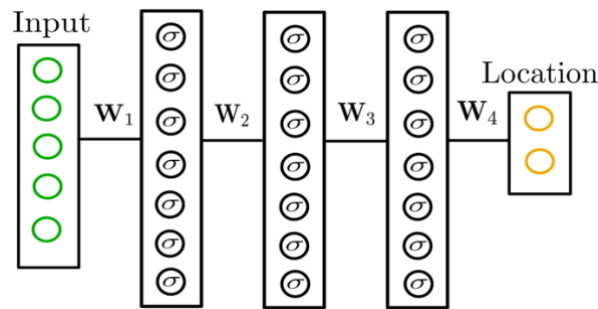


Figure 6.1 The structure of DNN to perform regression rather than classification tasks. Source: [11].

(X) Improvement of the cloud-based database

Don't use Google Sheet, use

- (1) Google Firebases in Google Cloud Platform (GCP) to collect RSSI callbacks from Sigfox backend server
 - a. Advantages: faster to handle large number of callbacks at the same time
 - b. Disadvantages: (1) need to setup GCP first which may cost money; (2) the concepts/configurations in GCP may be confusing if it is the first time for a user to use
- (2) setup a local SQL etc. database in RPi to collect callbacks from Sigfox backend server
 - a. Advantages: (1) more traditional, the concepts are easier to understand
 - b. Disadvantages: (1) need a fixed IP; (2) need electricity to power on the RPi; (3) maybe slower than GCP so that RPi cannot handle too much callbacks at the same time)

(X) Construction of the real-time ML localization system (maybe run in api server, in mobile app/web app)

Embed the trained ML models into the script/program of LLS localization running in the server (in RPi or cloud-server). So, an instant/real-time machine-learning LLS localization service can be provided.

7 Reference

- [1] "K.-H. Lam, C.-C. Cheung, and W.-C. Lee, "LoRa-based localization systems for noisy outdoor environment," 2017 IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), 2017."
- [2] "Abdullah AI-Ahmadi, Tharek Abd. Rahman, One Stage Indoor Location Detennination Systems, New Approach of Indoor and Outdoor Localization Systems,p 163-190".
- [3] M. Sauter, "3.7.1 Mobility Management in the Cell-DCH State," in *GSM to LTE: An Introduction to Mobile Networks and Mobile Broadband*, John Wiley & Sons, 2010, p. 160.
- [4] "S. Bush, "Sigfox IoT network reaches 50 countries," Electronics Weekly, 02-Oct-2018. [Online]. Available: <https://www.electronicsworld.com/news/products/rf-microwave-optoelectronics/sigfox-iot-network-reaches-50-countries-2018-10/>. [Accessed: 17-Oct-2019".
- [5] "Gomez, Carles, et al. "A Sigfox Energy Consumption Model." Sensors, vol. 19, no. 3, 2019, p. 681., doi:10.3390/s19030681."
- [6] ""Sigfox." The Global Communications Service Provider for the Internet of Things (IoT), 22 Apr. 2020, www.sigfox.com/en."
- [7] "Communications Authority - List of Licensees, www.coms-auth.hk/mobile/en/licensing/telecommunications/wiot/list_of_licensees/index.html," [Online].
- [8] ""The IoT Telco." Thinxtra, www.thinxtra.com/," [Online].
- [9] "M. Sauter, "3.7.1 Mobility Management in the Cell-DCH State," in *From gsm to lte-advanced pro and 5g: an introduction to mobile networks and mobile broadband*, Hoboken, NJ: Wiley, 2017, p. 160."
- [10] "C. Cortes and V. Vapnik, "Support-vector networks," Machine learning, vol. 20, no. 3, pp. 273–297, 1995."
- [11] "Xiao, Linchen & Behboodi, Arash & Mathar, Rudolf. (2018). Learning the Localization Function: Machine Learning Approach to Fingerprinting Localization."

- [12] "A. Kyrykovich, "Deep Neural Networks," KDnuggets. [Online]. Available: <https://www.kdnuggets.com/2020/02/deep-neural-networks.html>. [Accessed: 12-May-2020].," [Online].
- [13] "C.-H. Hsieh, J.-Y. Chen, and B.-H. Nien, "Deep Learning-Based Indoor Localization Using Received Signal Strength and Channel State Information," IEEE Access, vol. 7, pp. 33256–33267, 2019."
- [14] ""Custom Callback Creation," Custom Callback Creation | Sigfox Resources. [Online]. Available: <https://support.sigfox.com/docs/custom-callback-creation>. [Accessed: 17-Oct-2019].".
- [15] "G. G. Anagnostopoulos and A. Kalousis, "A Reproducible Analysis of RSSI Fingerprinting for Outdoor Localization Using Sigfox: Preprocessing and Hyperparameter Tuning," 2019 International Conference on Indoor Positioning and Indoor Navigation (IPIN), 2019."
- [16] "R. S. Sinha, S.-M. Lee, M. Rim, and S.-H. Hwang, "Data Augmentation Schemes for Deep Learning in an Indoor Positioning Application," Electronics, vol. 8, no. 5, p. 554, 2019."
- [17] "G. Zhang, P. Wang, H. Chen, and L. Zhang, "Wireless Indoor Localization Using Convolutional Neural Network and Gaussian Process Regression," Sensors, vol. 19, no. 11, p. 2508, 2019."
- [18] "B. Sklar, "Rayleigh fading channels in mobile digital communication systems .I. Characterization," IEEE Communications Magazine, vol. 35, no. 7, pp. 90–100, 1997."
- [19] "K. Mekki, E. Bajic, F. Chaxel, and F. Meyer, "A comparative study of LPWAN technologies for large-scale IoT deployment," ICT Express, vol. 5, no. 1, pp. 1–7, 2019."
- [20] "J. Xu, W. Liu, F. Lang, Y. Zhang, and C. Wang, "Distance Measurement Model Based on RSSI in WSN," Wireless Sensor Network, vol. 02, no. 08, pp. 606–611, 2010."
- [21] "H. Xu, M. Wu, P. Li, F. Zhu, and R. Wang, "An RFID Indoor Positioning Algorithm Based on Support Vector Regression," Sensors, vol. 18, no. 5, p. 1504, Oct. 2018."