Indoor Navigation and Localization

Machine Learning Engineer Nanodegree
Capstone Project

Jay Patel
August 05, 2018

Table of Contents

Glossary

iBeacon (beacon, iBeacon device): Low energy wireless portable devices
data variables (features)
beacon variables (b3001, ..., b3013)
samples, entries, data points, n
Variational Auto Encoder (VAE)
Deep Neural Network (DNN)
Reinforcement Learning (RL)
Deep Reinforcement Learning using Q- Learning (DQRL)
Bluetooth Low Energy (BLE)
Received Signal Strength Indication (RSSI, iBeacon values)
A 2D square matrix, representing a floor map, made up of little squares (Grid)
Rows of the grid (rows)
Column of the grid (columns)
A tuple or array of row and column (position on the grid)
Internet of Things (IoT)

Abstract

The combination of Deep Neural Network (DNN), Reinforced Learning (RL), and semi-supervised learning is used to utilize unlabeled data collected by the IoT devices to help users navigate inside the buildings. A reinforcement learning agent is trained, using both labeled and unlabeled data, to help the users of smart cities navigate indoors. The main aim of the project is to guide the user as close to the target as possible. The unsupervised learning model performs better in guiding an RL agent close to the target compared to the supervised learning model.
Definition

Definition

Project Overview

The problem caused by the unlabeled data in smart cities is growing rapidly. The smart equipment used in the smart cities generates the tremendous amount of data. However, most of the data generated are not labeled; therefore, all the unlabeled data, which accounts for the eighty percent of the total data, get thrown out and not used in the machine learning to predict the outcome or to find a pattern in the data.

Most of the current methods that use the data from the smart cities utilize the method of sampling to make the predictions or to find a general pattern in the data. Even though sampling method produces good estimates in some cases, the sampling method is not the most effective way to utilize the data since the preference of the people living in different parts of the cities matters more. In order to use sampling and predict the data well, the sampling of the data needs to be large and, it is not possible to use unlabeled data for the sampling.

The detection of criminal activities through the social media sites like Facebook and Twitter is an example where sampling method does not work. In order to detect a criminal with great accuracy, the data needs to sufficiently large. Therefore, the sampling methods are not useful in such scenarios. The unlabeled data could be used to increase the datasets.

The combination of Deep Neural Network, Reinforced Learning, and semi-supervised learning is a potential solution to the problem created by a large amount of unlabeled data.

DNN is a learning based on the neural nets and it is a part of supervised learning. Reinforced learning is part of unsupervised learning which is a good match for the smart city data since many of the smart city data does not need an output but need an action to do a certain task. RL is a reward-based system where each action taken by the system has a reward and the RL is trained in such a way that it maximizes the reward.

Similarly, semi-supervised learning uses the combination of labeled and unlabeled data in order to make decisions. Since there are many data points without the label, the semi-supervised learning is a great solution to the problem.

Problem Statement

The indoor navigation of a user using the information collected by the smart IoT devices is the goal of the project. The dataset contains the information about a discrete floor map and

the readings from IoT devices with which disabled or blind users can be navigated. The project aims to navigate the user from one location to another location within the floor map with minimum distance traveling. The initial and final position of the user will be set in advance. The path to travel from the initial to target location is the challenge that needs to be solved. The problem can have different initial and final positions that are within the floor map.

The project has two datasets, labeled and unlabeled. The unlabeled data needs to be used to travel from initial position to the target position efficiently. The solution model should be able to utilize the unlabeled data to increase the performance of the model compare to the model that uses only labeled data.

Summary of the problem tasks:

- Understand the labeled data and extract relevant features
- Create a model to estimate the labels of unlabeled data based on the labeled dataset
- Use both labeled data and unlabeled data with estimated labels to train an RL agent.
- Compare the performance of the RL agent using labeled data and the combination of labeled and unlabeled data.

The combination of Deep Neural Network, Reinforced Learning, and semi-supervised learning is used as the solution to the problem created by a significant amount of unlabeled data.

Summary of the solution tasks:

- The techniques like scaling, encoding, clustering, supervised learning, feature correlation, data visualization, etc. are used to investigate the data and extract critical features.
- The models like variational autoencoder, deep neural network, mathematical models, supervised learning, etc. are used to predict the labels of the unlabeled data from the iBeacon RSSI values and vice-versa.
- The deep Q-learning with experience replay and epsilon-greedy policy is used to train an RL agent for a given start and end position.

Metrics

The solution model will be evaluated based on the distance traveled toward the target per episode. As the agent trains, the agent should be able to get to the target position faster or with fewer time steps. The supervised learning model has only labeled dataset to train from and decide the path to the destination while semi-supervised learning model has both labeled and unlabeled data. Therefore, the semi-supervised learning model should be able to learn better than the supervised learning model and get to the target position in more optimized manner.

The rewards and the distance traveled by the agent will be calculated and stored at each time step for both supervised and semi-supervised learning models. If the agent goes more distance or takes fewer steps to get to the destination in semi-supervised model compare to the supervised model, the semi-supervised model has performed better than the supervised learning model. Therefore, better performance of the semi-supervised model proves that the unlabeled data help increase the efficiency of the model.

Analysis

Data Exploration

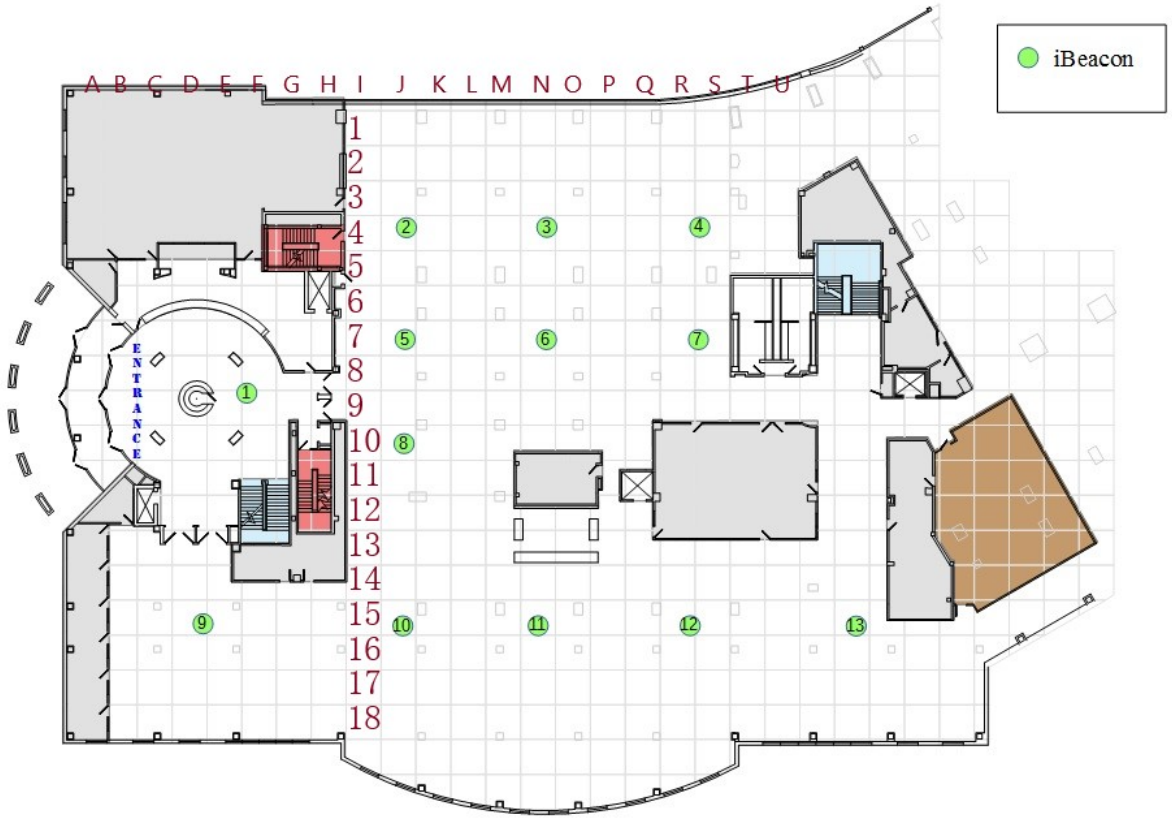

Figure 1 Floor map of library locating beacons and positions (Mohammadi, Al-Fuqaha, Guizani, & Sorour, 2018)

Figure 1 shows the first floor of Waldo Library in Western Michigan University with the markings iBeacons in green color and a grid consisting little squares. IBeacons are low power Bluetooth devices that send a signal to a user. The closer a user to a beacon, the stronger the signal from the beacon.

As shown in figure 1, there are thirteen iBeacons set up on the first floor of the library. Each beacon transmits the signal which is picked up by the nearby user Bluetooth device. The strength of the signal from the beacon to the user is measured by the user device in RSSI. RSSI is the short form of Received Signal Strength Indicator. RSSI is a value indicating the strength of the signal from an iBeacon to the user device. The RSSI values measured are negative with minimum being -200. The RSSI value -200 suggests that the iBeacon is not in the range of the user device. A bigger value of RSSI from an iBeacon like -75 suggests that the iBeacon is closer to the user device. Since there are thirteen iBeacon devices, there are thirteen data variables, one data variable for each iBeacon. The data variables of the iBeacon are called as b3001, b3002, …, b3013. The iBeacon data variables contain only negative integers indicating RSSI values.

Figure 1 is divided into small rectangles of which columns are named by a letter and rows are named by a number. The little squares are the position of the library mapped by the letters and the numbers shown in figure 1. Each position, square, in the figure, is named as the ColumnRow format. For example, the iBeacon 3 is the square named N03 where N is the column and 03 is the row of the square location. In the dataset, each location has been assigned thirteen iBeacon RSSI values and a date.

The dataset also contains the time and date upon which location name and thirteen iBeacon data variables are captured. The format of the date variable is MM-DD-YYYY HH:MM:SS.

The dataset has fourteen input variables and one output variable. The fourteen-input variable includes all the thirteen iBeacon RSSI values data variables and date variable. The output variable is the symbolic location (square).

The dataset has two sets. One set contains the symbolic location values and the other set do not contain the location values. However, both dataset contains all the values of the fourteen input variables with no missing data.

	location	date	b3001	b3002	b3003	b3004	b3005	b3006	b3007	b3008	b3009	b3010	b3011	b3012	b3013
0	O02	10-18-2016 11:15:21	-200	-200	-200	-200	-200	-78	-200	-200	-200	-200	-200	-200	-200
1	P01	10-18-2016 11:15:19	-200	-200	-200	-200	-200	-78	-200	-200	-200	-200	-200	-200	-200
2	P01	10-18-2016 11:15:17	-200	-200	-200	-200	-200	-77	-200	-200	-200	-200	-200	-200	-200
3	P01	10-18-2016 11:15:15	-200	-200	-200	-200	-200	-77	-200	-200	-200	-200	-200	-200	-200
4	P01	10-18-2016 11:15:13	-200	-200	-200	-200	-200	-77	-200	-200	-200	-200	-200	-200	-200
5	P01	10-18-2016 11:15:11	-200	-200	-82	-200	-200	-200	-200	-200	-200	-200	-200	-200	-200

Figure 2 First six data rows in labeled dataset

As shown in figure 2, the data set contains fifteen variables. The output variable location could have the total of 21*18 = 378 potential output categories. However, many of the categories are not accessible due to furniture and other objects on the floor. The thirteen RSSI values of iBeacons vary from -200 to a higher negative number. Most of the values of RSSI are -200. The date variable contains date and time of the data point collection.

The output variable: Location

The input variables: Date, b3001-b3013

Since the output variable location is categorized in possible 378 categories, the problem is a classification problem.

	location	date	b3001	b3002	b3003	b3004	b3005	b3006	b3007	b3008	b3009	b3010	b3011	b3012	b3013
0	?	11-7-2016 12:29:01	-200	-200	-80	-200	-77	-56	-81	-200	-200	-200	-200	-200	-200
1	?	11-7-2016 12:29:00	-200	-200	-80	-200	-78	-56	-200	-200	-200	-200	-200	-200	-200
2	?	11-7-2016 12:28:59	-200	-200	-81	-200	-74	-63	-200	-200	-200	-200	-200	-200	-200
3	?	11-7-2016 12:28:58	-200	-200	-200	-200	-75	-56	-200	-200	-200	-200	-200	-200	-200
4	?	11-7-2016 12:28:57	-200	-200	-200	-200	-82	-56	-200	-200	-200	-200	-200	-200	-200

Figure 3 First five rows of unlabeled dataset

In the unlabeled dataset, the format of the data variables is same as the labeled dataset except for the location variable. The location variable values are not known in the unlabeled dataset; therefore, the values are represented by the question mark as shown in figure 3. The unlabeled dataset cannot be used to train different classifiers since the output variable, location is unknown. Hence, the unlabeled dataset is only used to visualize the data.

```
RangeIndex: 1420 entries, 0 to 1419
Data columns (total 15 columns):
location    1420 non-null object
date        1420 non-null object
b3001       1420 non-null int64
b3002       1420 non-null int64
b3003       1420 non-null int64
b3004       1420 non-null int64
b3005       1420 non-null int64
b3006       1420 non-null int64
b3007       1420 non-null int64
b3008       1420 non-null int64
b3009       1420 non-null int64
b3010       1420 non-null int64
b3011       1420 non-null int64
b3012       1420 non-null int64
b3013       1420 non-null int64
dtypes: int64(13), object(2)
memory usage: 166.5+ KB
```

Figure 4 Data types of variable in labeled dataset

As shown in figure 4, the data type of the output variable is of the object since it is formatted as ColumnRow. The date variable is also of object type since it is formatted as MM-DD-YYYY HH:MM:SS. The iBeacon reading of RSSI values is of 64-bit integer type. All the data variables have no null entries.

	b3001	b3002	b3003	b3004	b3005	b3006	b3007	b3008	b3009	b3010	b3011
count	1420.000000	1420.000000	1420.000000	1420.000000	1420.000000	1420.000000	1420.000000	1420.000000	1420.000000	1420.000000	1420.000000
mean	-197.825352	-156.623944	-175.533099	-164.534507	-178.378169	-175.063380	-195.637324	-191.970423	-197.145070	-197.442254	-197.748592
std	16.259105	60.217747	49.452958	56.523261	47.175799	49.596627	22.880980	30.733742	19.160207	17.741632	16.852535
min	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000
25%	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000
50%	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000
75%	-200.000000	-78.000000	-200.000000	-80.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000
max	-67.000000	-59.000000	-56.000000	-56.000000	-60.000000	-62.000000	-58.000000	-56.000000	-55.000000	-61.000000	-59.000000

Figure 5 statistics of the iBeacon RSSI readings for labeled dataset

As per figure 5, the dataset has the total of 1420 data points. The statistics like mean, standard deviation, first quartile, second quartile and median does not make sense for the dataset since most of the values in the dataset are -200. The data values are not sparse but concentrated at the values like -200, and in the range [-55,-67] as shown in figure 7 below. The minimum value in the dataset is -200 and the maximum values ranges from -55 to -67 among the data variables.

	b3001	b3002	b3003	b3004	b3005	b3006	b3007	b3008	b3009	b3010	b3011
count	5191.000000	5191.000000	5191.000000	5191.000000	5191.000000	5191.000000	5191.000000	5191.000000	5191.000000	5191.000000	5191.00000
mean	-179.238682	-166.233674	-181.122327	-183.614910	-158.049701	-157.634175	-185.345983	-169.053747	-184.328838	-185.461183	-187.83953
std	45.321908	54.078027	42.953228	40.498886	57.019260	57.497039	39.557530	51.313281	40.940442	38.361122	35.57793
min	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.00000
25%	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.00000
50%	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.00000
75%	-200.000000	-90.500000	-200.000000	-200.000000	-84.000000	-83.000000	-200.000000	-103.000000	-200.000000	-200.000000	-200.00000
max	-63.000000	-62.000000	-60.000000	-61.000000	-58.000000	-56.000000	-61.000000	-60.000000	-59.000000	-60.000000	-59.00000

Figure 6 Statistics of the iBeacon RSSI readings for unlabeled dataset

As shown in figure 6, the unlabeled dataset contains 5191 data points. The statistics of the labeled and unlabeled dataset looks similar.

Exploratory Visualization

As shown in figure 7 below, the histogram for the RSSI values for each of the beacons versus the number of the data points at each RSSI value for labeled dataset. In the histogram, all of the thirteen beacons have 1000 or more data points at the value -200. The second highest bar is around -75 for most of the beacons in the histogram. For beacons 7-13, the number of points other than -200 is very low. The value -200 shows that the beacon did not detect any user and the value around -75 represents the detection of the user near the beacon.

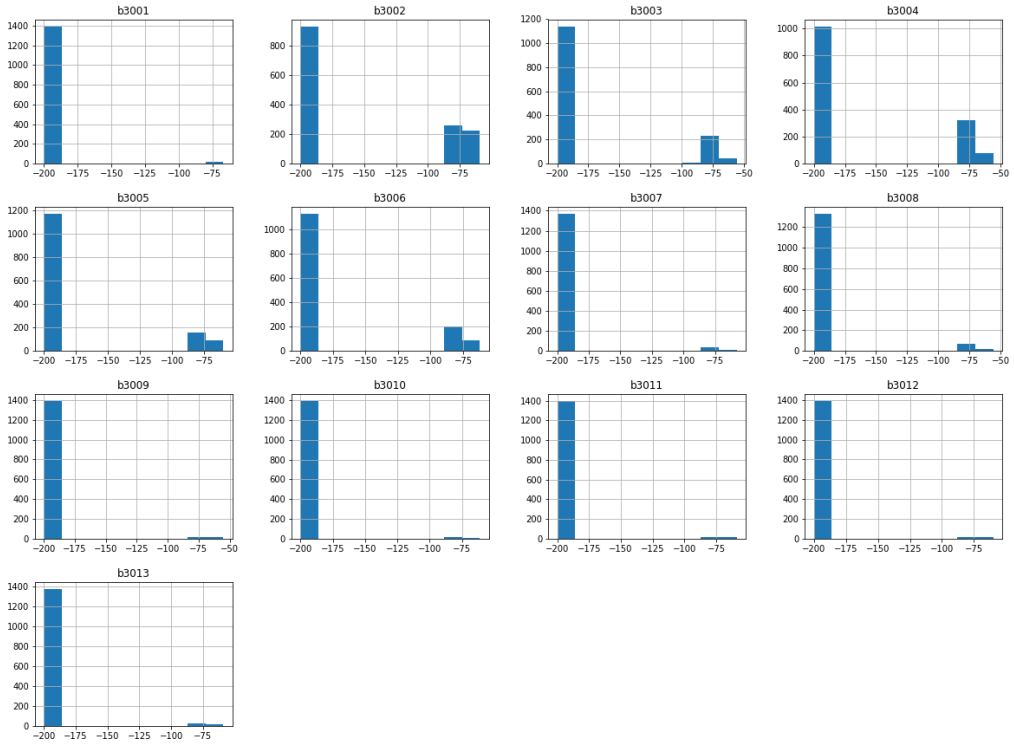

Figure 7 iBeacon RSSI vs Number of data points Histogram for 13 beacons labeled data

	b3001	b3002	b3003	b3004	b3005	b3006	b3007	b3008	b3009	b3010	b3011	b3012	b3013
b3001	1.000000	-0.073123	-0.066219	-0.083980	0.103180	-0.043067	-0.025520	0.330190	-0.019943	-0.019296	-0.017881	-0.019969	-0.023878
b3002	-0.073123	1.000000	0.075160	-0.409673	0.054061	-0.210235	-0.115099	0.007335	-0.107406	-0.094341	-0.096299	-0.107542	-0.128599
b3003	-0.066219	0.075160	1.000000	-0.144218	-0.162481	0.135232	-0.047100	-0.123059	-0.073771	-0.071377	-0.066143	-0.073865	-0.088328
b3004	-0.083980	-0.409673	-0.144218	1.000000	-0.264527	-0.185911	0.070760	-0.158229	-0.093558	-0.090521	-0.083883	-0.093677	-0.112018
b3005	0.103180	0.054061	-0.162481	-0.264527	1.000000	0.010307	-0.068146	0.285083	-0.068340	-0.066122	-0.061273	-0.068427	-0.081825
b3006	-0.043067	-0.210235	0.135232	-0.185911	0.010307	1.000000	-0.074927	-0.068524	-0.074970	-0.049186	-0.067217	-0.075065	-0.089763
b3007	-0.025520	-0.115099	-0.047100	0.070760	-0.068146	-0.074927	1.000000	-0.049850	-0.028430	-0.027507	-0.025490	-0.028466	-0.034040
b3008	0.330190	0.007335	-0.123059	-0.158229	0.285083	-0.068524	-0.049850	1.000000	-0.038956	0.103724	-0.034928	0.001415	-0.046643
b3009	-0.019943	-0.107406	-0.073771	-0.093558	-0.068340	-0.074970	-0.028430	-0.038956	1.000000	0.073096	-0.019920	-0.022246	-0.026601
b3010	-0.019296	-0.094341	-0.071377	-0.090521	-0.066122	-0.049186	-0.027507	0.103724	0.073096	1.000000	0.254659	0.040681	0.026559
b3011	-0.017881	-0.096299	-0.066143	-0.083883	-0.061273	-0.067217	-0.025490	-0.034928	-0.019920	0.254659	1.000000	0.247945	0.003452
b3012	-0.019969	-0.107542	-0.073865	-0.093677	-0.068427	-0.075065	-0.028466	0.001415	-0.022246	0.040681	0.247945	1.000000	0.127178
b3013	-0.023878	-0.128599	-0.088328	-0.112018	-0.081825	-0.089763	-0.034040	-0.046643	-0.026601	0.026559	0.003452	0.127178	1.000000

Figure 8 Correlation Matrix of all thirteen beacons

Figure 8 shows the correlation matrix of all the beacon variables. Most of the correlation matrix values are close to zero. Therefore, the beacon variables have no correlation with each other.

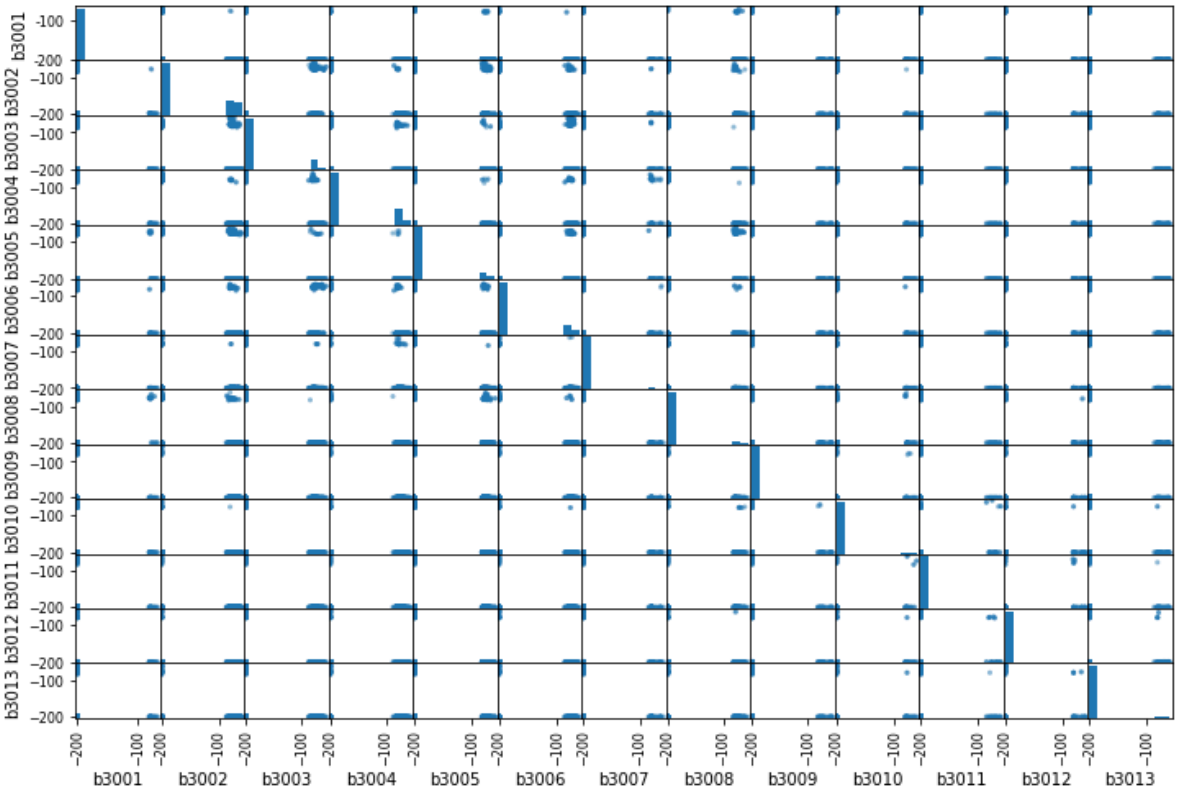

Figure 9 Scatter plot matrix of beacon variables

Figure 9 shows the scatter plot matrix of the beacon variables. Scatter plot matrix conforms that most of the data in all thirteen beacon variables are concentrated at the RSSI value -200, and only a few points are at other RSSI values.

As shown in figure 10 below, the histogram for the RSSI values for each of the beacons versus the number of the data points at each RSSI value for unlabeled dataset. Similar to the labeled dataset, in the histogram, all of the thirteen beacons have 1000 or more data points at the value -200. The second highest bar is around -75 for most of the beacons in the histogram. For beacons 7-13, the number of points other than -200 is very low. The value -200 shows that the beacon did not detect any user and the value around -75 represents the detection of the user near the beacon.

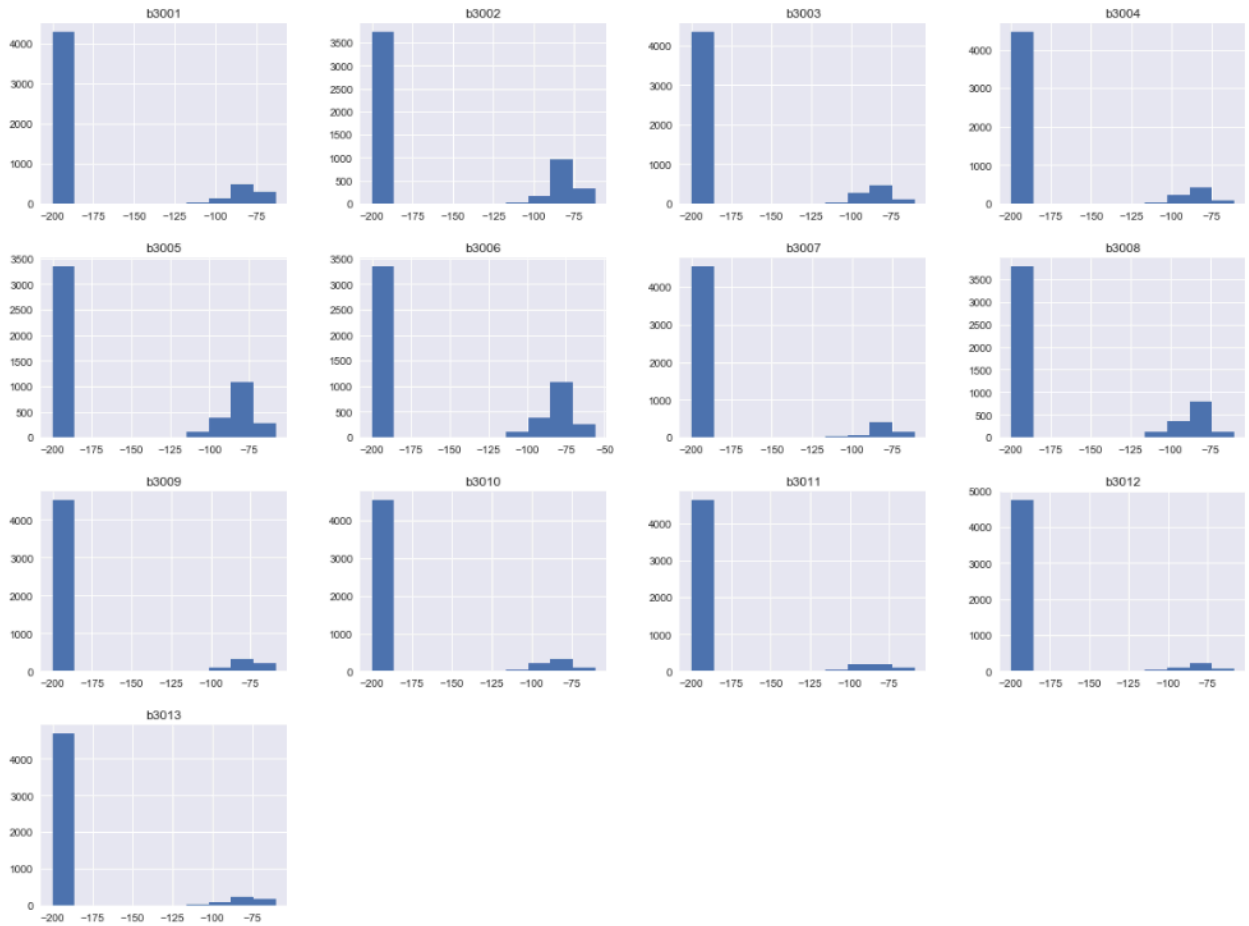

Figure 10 iBeacon RSSI vs Number of data points Histogram for 13 beacons unlabeled data

Table 1 Summary of Date feature

No.	Date	Minutes	Number of Data Points
1	18-Oct	40	601
2	4-Oct	9	37
3	3-Oct	5	50
4	27-Sep	2	21
5	21-Sep	8	73
6	19-Sep	8	50
6	4-Aug	4	26
7	21-Jul	15	62
8	25-May	21	160
9	20-Apr	50	303
10	19-Apr	8	29
	Total:	170	1420

The date is another input variable, the feature that is an object or customized format. The summary of the date variable for labeled dataset is provided in table 1. The data in the labeled dataset was taken at ten different dates over the span of six months. The dates are from the year 2016. The second column of table 1 shows the number of minutes spend on each day to collect the data, and the third column shows the number of data points collected on a given day.

Figure 11 Number of minutes spent per day for data collection

Figure 11 shows the histogram of the number of minutes spend on a particular day. The number of minutes on each day varies from 2 to 50, and no two days have the same number of minutes. Therefore, the minutes were spent randomly on each day to collect the data.

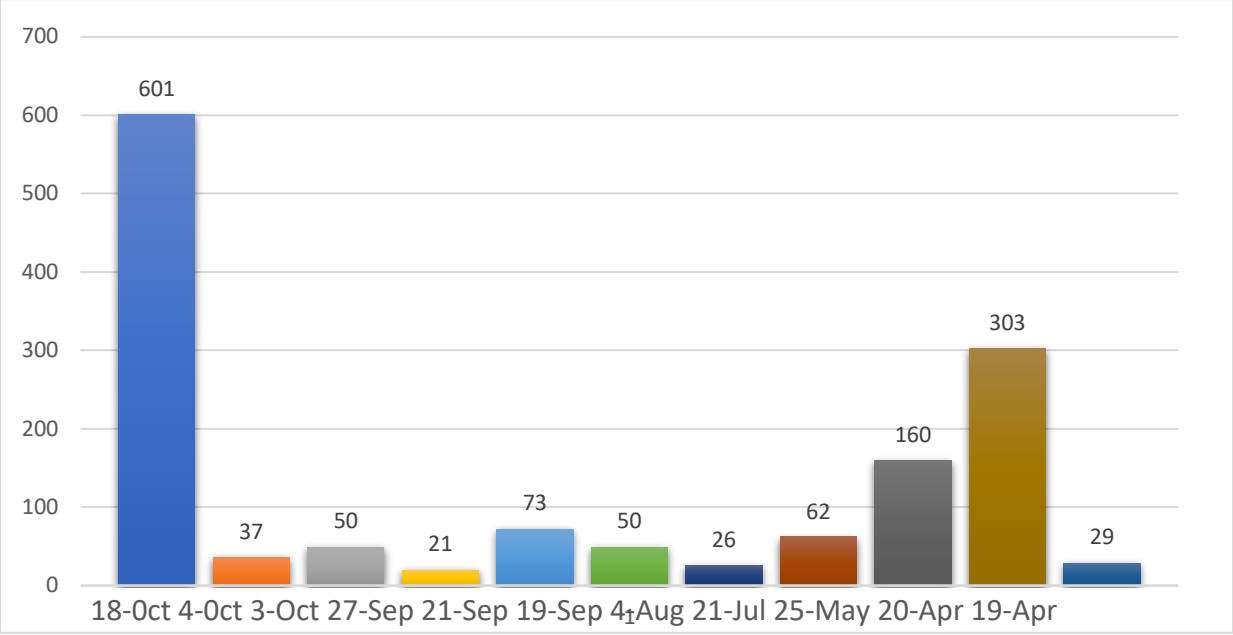

Figure 12 Collected data points per day

Figure 12 shows the number of data points collected on each day. The comparison between the figures 11 and 12 shows that the number of collected data points is higher when the spent minutes are higher in most of the cases.

The analysis of the time variable was done in an excel file and the values shown in table 1 was observed manually. Even though, the date variable is not used in the analysis of building models for user navigation, the data collected in table 1 and figures 11 and 12 are critical in understanding the data.

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

Equation 1: RSSI based on the distance (d)

where n is the signal propagation constant, d is the distance in meters, and A is the offset RSSI reading at 1 m from the transmitter (Mohammadi, Al-Fuqaha, Guizani, & Sorour, 2018). Equation 1 makes it clear that the RSSI values from each iBeacon are directly proportional to the logarithm of the distance from the iBeacon. However, the RSSI value involves two random variable n and A. Therefore, the RSSI values for one state or one category cannot be the same. Also, the RSSI values also depend on the environment around the user. For example, the number of people moving around the people, the object placement around the user, etc. Changing the situation like the structure of the building from inside or changing the locations of the objects like furniture will affect the RSSI readings. If something or someone blocks the user from the iBeacon, the user may not detect the RSSI reading from the iBeacon for a given position even if the iBeacon is in the range.

Algorithms and Techniques

Since the distance and the RSSI values are closely related to each other, the feature related to the distance can be very important in the analysis.

The input variables and feature b3001-b3013 contain only negative integer numbers, and from the histogram in figure 7, it is clear that the features do not have any outliers. All the numbers in the beacon variables are concentrated around the numbers -200 and -75. Therefore, the features b3001-b3013 do not need cleaning. However, the analysis can be made much simpler if the RSSI values are scaled from 0 to 1 rather than from -200 to -75. Therefore, the data scaling is needed.

The date variable is an object type that represents the date and time of the measured data point. However, the prediction of the location is purely based on the combination of beacon RSSI values since the user can walk at the location at any time. In addition, each data point has a different timestamp; therefore, the date variable could throw the classifier prediction off since there is no pattern to make the decision on. Hence, the date feature is not used to train any classifiers.

The location variable is of categories, and the name of the categories consists of columns letter and row numbers. In order to train a classifier, the category needs to be in the

form of a unique decimal or binary number. The decimal number like 1,2, etc. could produce some unwanted correlation between the location categories. Therefore, the one hot encoding is used to distinguish one hundred five categories from each other. The one hot encoding was implemented using SKLearn's OneHotEncoding library. Also, the row and column location can be separated from each other and a model for predicting each row and column can drastically reduce the prediction categories. Therefore, the model to predict the rows and columns are to be trained separately.

Removing the data points from the data set would not be a good idea for labeled dataset because the number of data points in the dataset is low compared to the number of categories to be predicted. In order to predict categories with high accuracy, the classifiers need to be trained on large datasets. Cleaning the data set may result in the reduction of the data point, and reduction in data points could reduce the performance of the classifier. Therefore, it is best not to remove any data entries to clean the data for the labeled dataset.

The predictions of the location variable will be done based on the thirteen beacon variables and the date variable will be discarded. The beacon variables have the same scaling of the data; therefore, no further feature scaling is needed to increase the performance of the classifier.

Since the beacon variables are not correlated with each other as shown in the correlation matrix in figure 8, there is no need for custom transformations to improve data or add features.

The main aim of the project is to navigate a user indoors using the information from the iBeacons and the floor map available. Assuming the floor map does not change, the data from the iBeacons is the input for the navigation process. The output of the navigation process is to predict the user position based on the information and guide the user to the destination accordingly. Figure 13 summarizes the algorithmic view of the project.

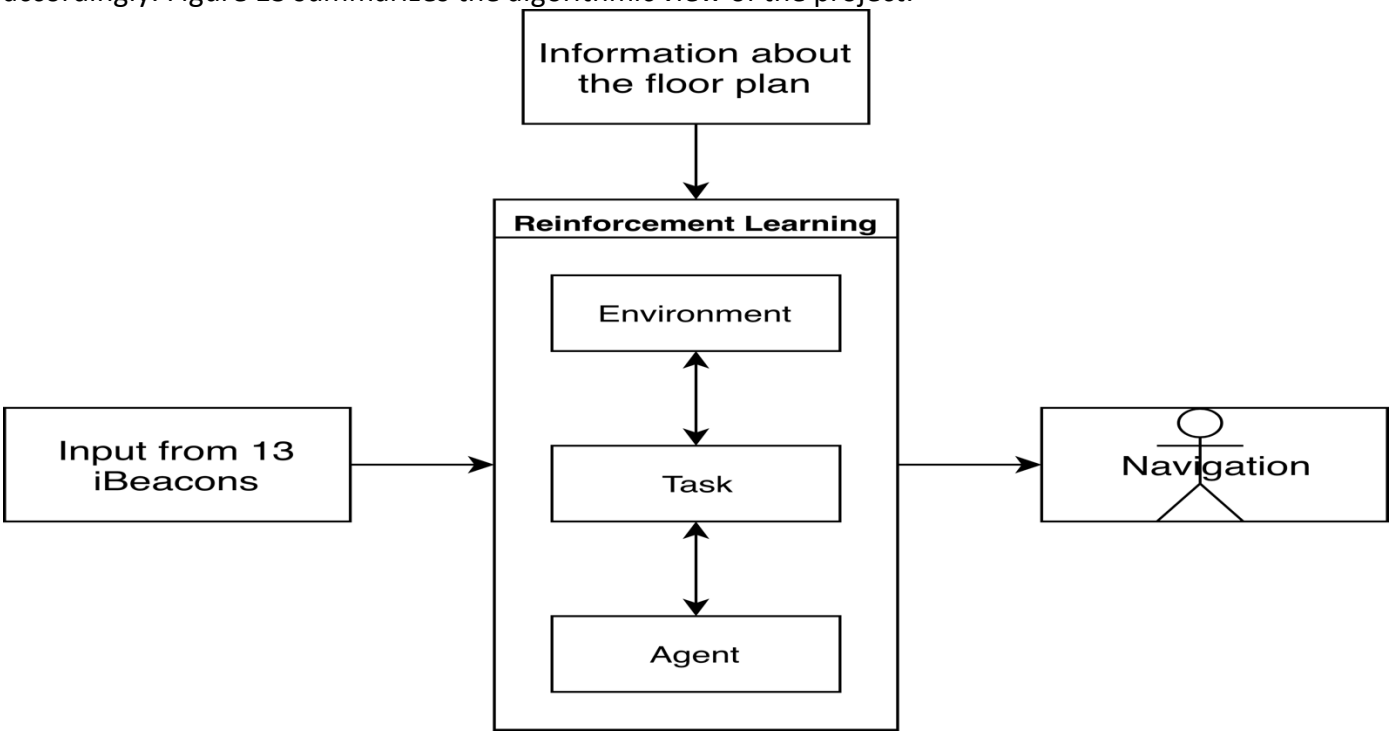

Figure 13 High level algorithmic view

The reinforcement learning algorithm is similar to the deep Q-learning algorithm. The deep neural network used in the algorithm is the variational encoder network. The agent interacts with the environment using the ε-greedy policy. The agent learns through experiences using experience replay method.

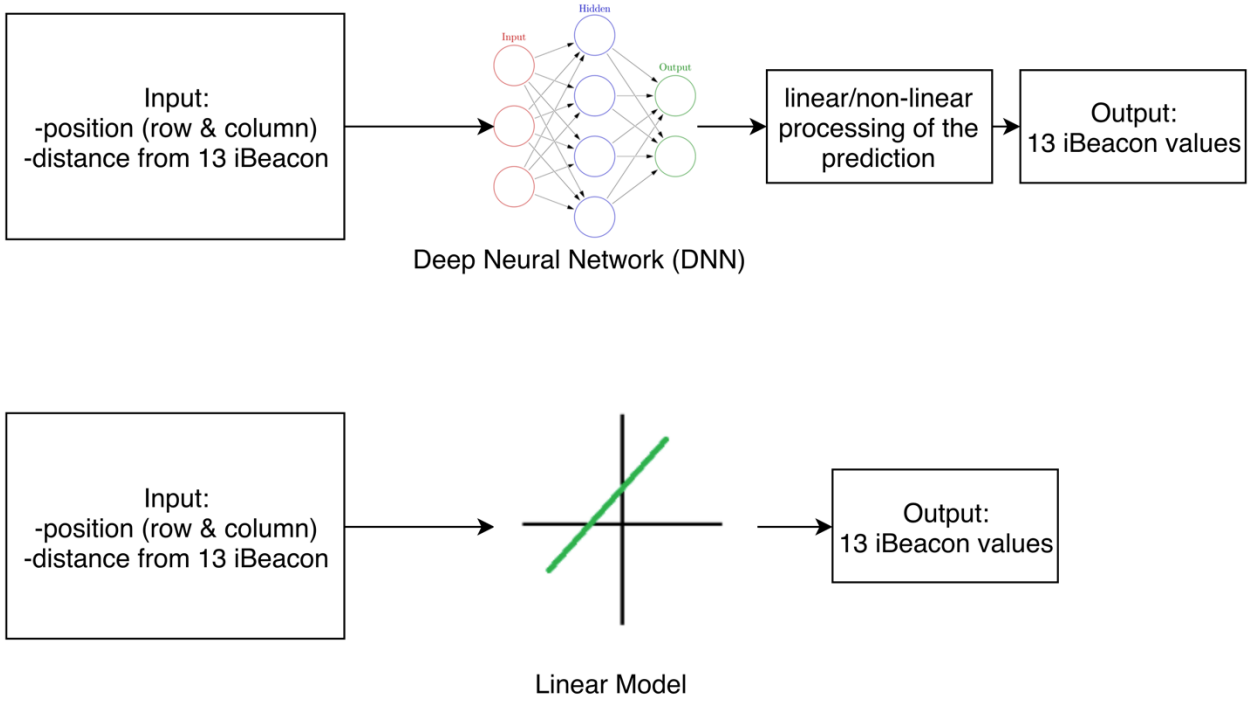

Figure 14 Environment class function models

The environment class has to predict the 13 iBeacon values for a particular position. Besides, the environment class is also responsible for deciding the location of the user based on a given action. The prediction of the 13 iBeacon values can be made through a simple linear mathematical modeling and neural network based modeling as shown in figure 14.

The rewards will be calculated based on the distance the agent travels towards the target location. If the agent goes close to the target, the rewards will be positive in proportion to the distance and vice versa. Also, if the user reaches within a certain distance of the target, the rewards will be positively amplified to encourage the agent to stay close to the target.

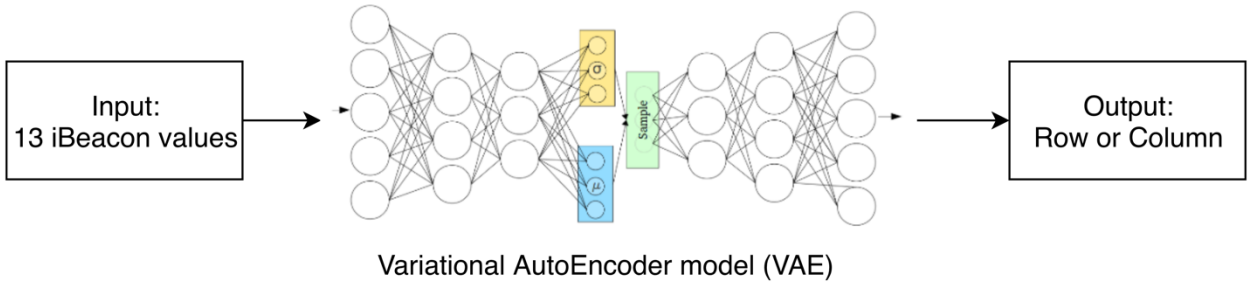

Figure 15 Prediction of the location for unlabeled data using VAE model

The labels for the unlabeled data will be predicted using a variational autoencoder model. The input to the model will be 13 iBeacon values present in the dataset, and the output will be row or column of the position as shown in figure 15. There will be two models, one to predict the row location and other to predict the column location.

Benchmark

The reinforcement learning model described in Algorithms and Techniques section will be used for two datasets. The one dataset consists of only labeled data provided initially. The second dataset will include of the labeled dataset and unlabeled dataset with predicted labels using VAE model. As per the benchmark, the model with both the labeled and unlabeled dataset should be able to perform better than the labeled dataset. The performance of the models is evaluated by the distance traveled by the agent toward the target. The more the distance traveled by the user to the destination, the better the performance.

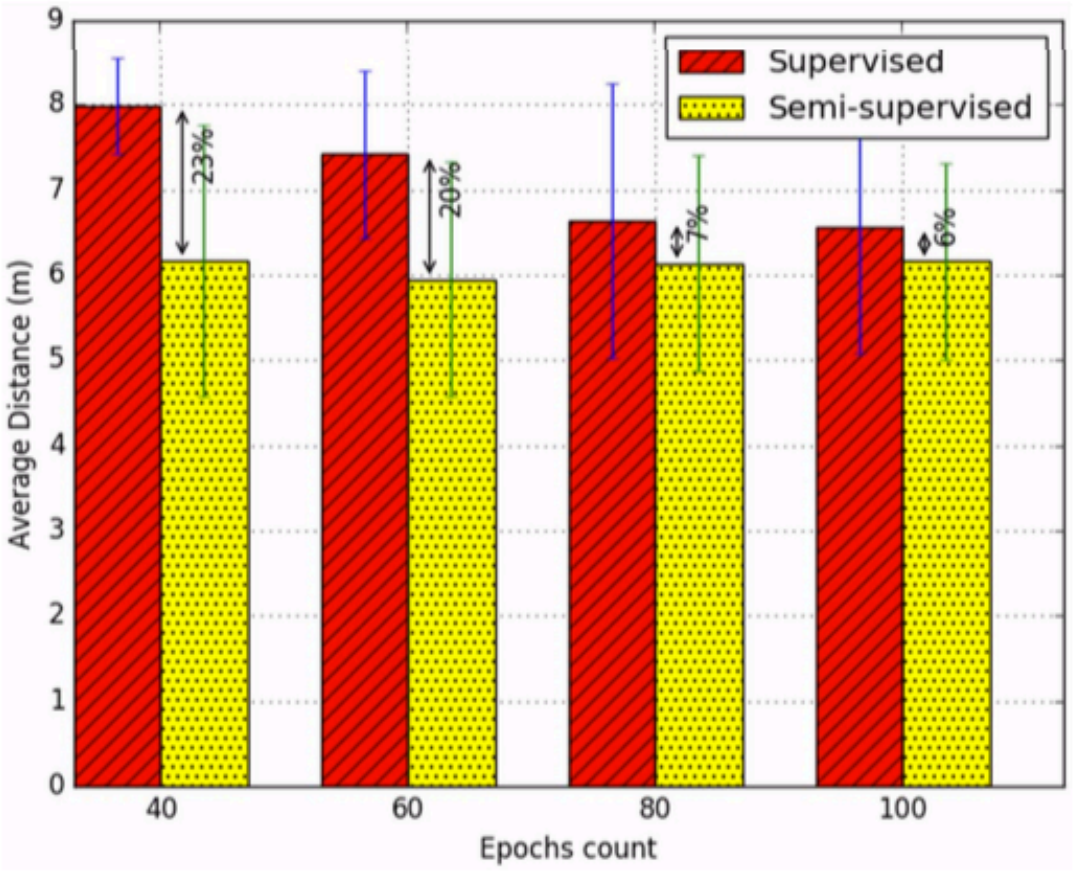

Figure 16 Hypothetical benchmark (Mohammadi, Al-Fuqaha, Guizani, & Sorour, 2018)

The article poses a hypothetical benchmark as shown in figure 16. The benchmark consists of supervised and semi-supervised learning which refers to the labeled dataset RL and a combination of labeled and unlabeled dataset RL respectively. As per the hypothetical

benchmark, the distance traveled by the agent trained by the combination of labeled and unlabeled dataset performs better. Similarly, the performance regarding the distance traveled towards the target should be better for the agent trained with both labeled and unlabeled dataset.

Methodology

Data Preprocessing

Data Scaling and Encoding

The BLE RSSI values from 13 iBeacons are in a range from -200 to -75. The scaling of the RSSI values from (-200) – (-75) to 0 – 1 will make the data processing simple. Therefore, the RSSI data of both labeled and unlabeled dataset was scaled to 0-1.

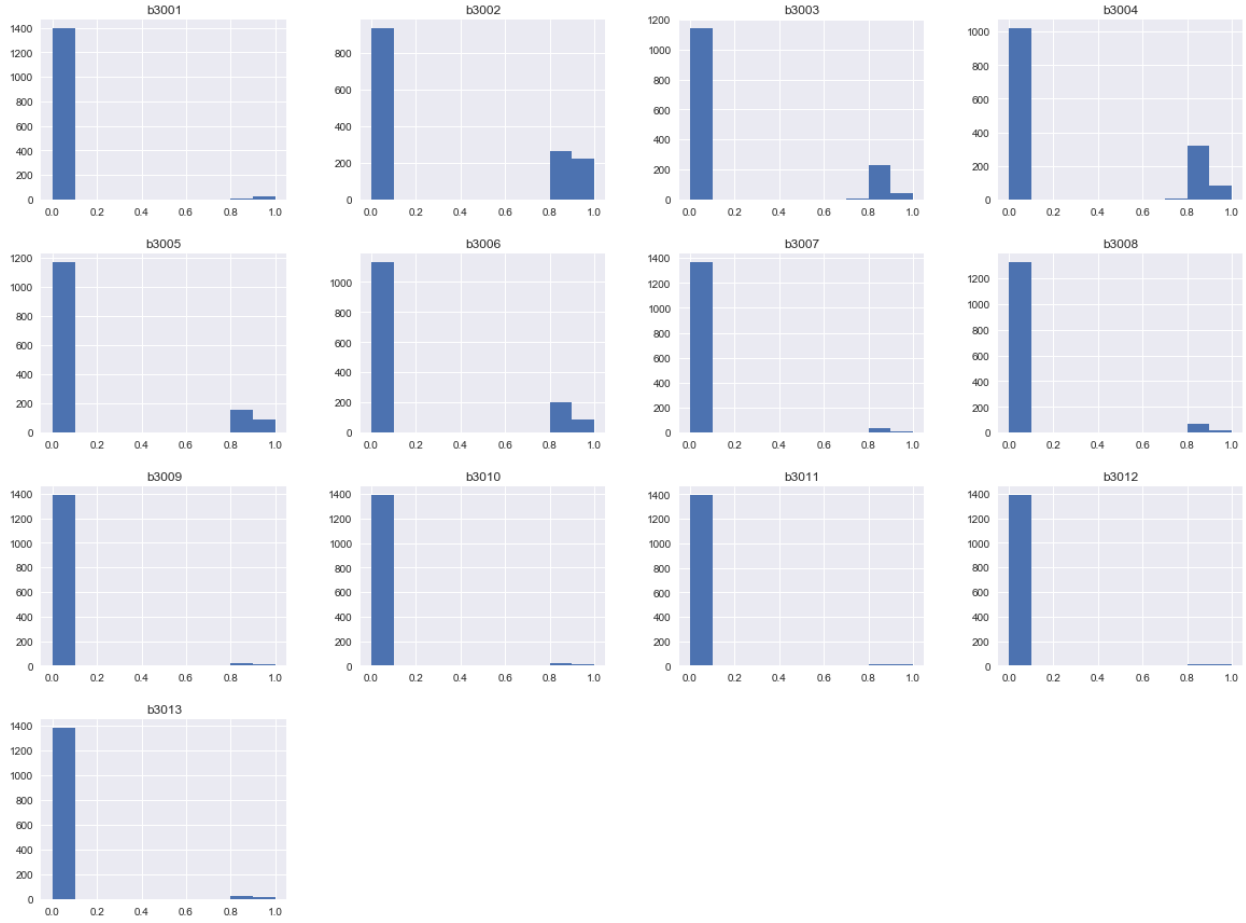

Figure 17 Scaled labeled dataset histogram

Figure 17 shows the histogram of the scaled labeled dataset. The x-axis is the RSSI values, and the y-axis is the number of data points. As we can see in figure 17, all thirteen iBeacon devices have very few points with values other than zero, and the RSSI values from 0 to 0.7 have no data points in all the iBeacons. The unlabeled dataset is also scaled similarly.

The locations are given as a letter for the column and a number for the row. The rows and columns need to be separated and encoded to train the neural network for the prediction of the rows and columns. Therefore, the rows and column letters or numbers were separated and encoded numerically.

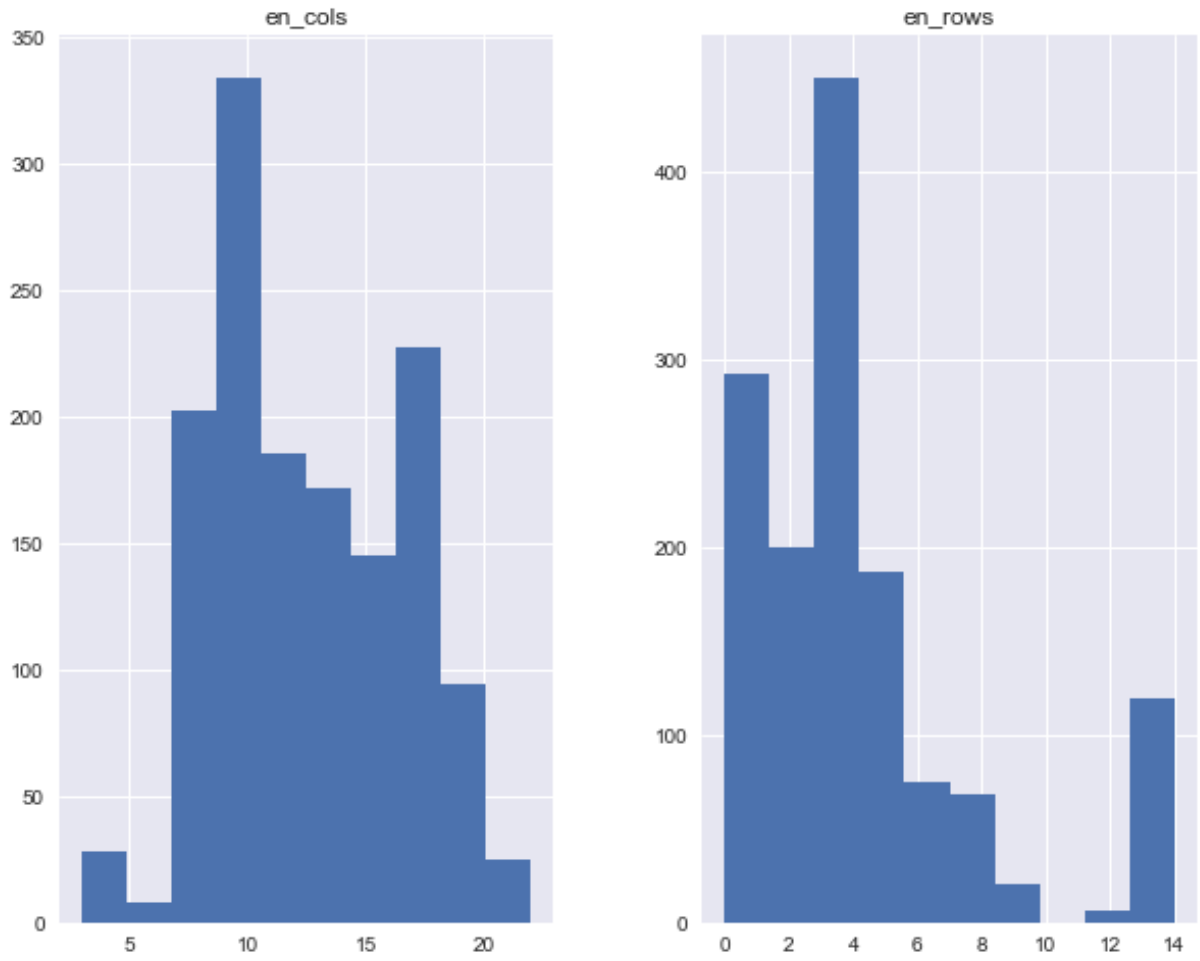

Figure 18 Distribution of labeled data points among encoded rows and columns

Figure 18 shows the distribution of data points among encoded rows and columns. The x-axis is the number of the encrypted columns or rows, and the y-axis is the number of data points. From figure 18, the data point distribution is low in first seven columns and last five columns. Similarly, the data point distribution is very low for the rows from 8 to 13. Therefore, the data to predict the rows and columns with low or none data points will be challenging.

Floor Map Analysis

As discussed before, the distance is a crucial feature since the RSSI values are related to the distance as shown in equation 1. Therefore, with the help of the floor map shown in figure 1, the locations of the iBeacon devices can be derived. Based on the location of the iBeacon

devices, the distance of each location in the labeled dataset to the iBeacon devices can also be calculated. The dataset of distance from the locations in the labeled dataset and iBeacon devices will be used to predict the RSSI values from a location using a deep neural network.

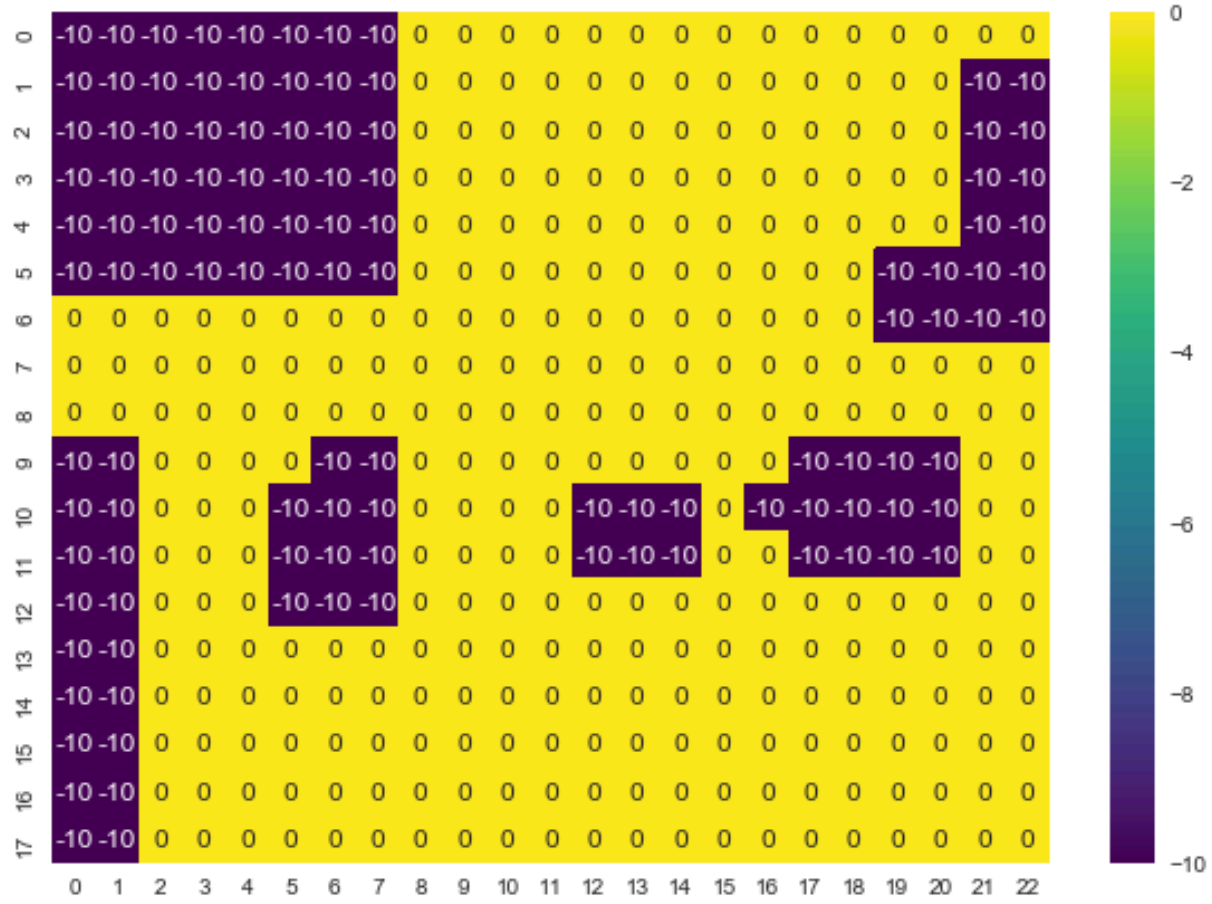

Figure 19 Virtual floor map aka the grid using 2D array of rows and columns

Figure 19 shows the virtual floor map aka the grid based on the number of columns and rows shown in figure 1. The value 0 in the grid represent the space available for navigation while the value -10 denotes an object or blocked location. The virtual grid is wholly based on my observation form figure 1.

Figure 20 shows the distribution of the data points in the labeled dataset on the grid. The numbers greater than zero represents the amount of data points available in the labeled dataset for a particular cell or location on the grid. The visualization in figure 20 is critical because it gives an idea of how the data is spread along the grid. As we can see in figure 20, most of the data points are concentrated around the top middle portion of the grid while the bottom of the grid only has few scatted data points. Due to low data points in the bottom of the grid, the navigation could be challenging at the bottom of the grid.

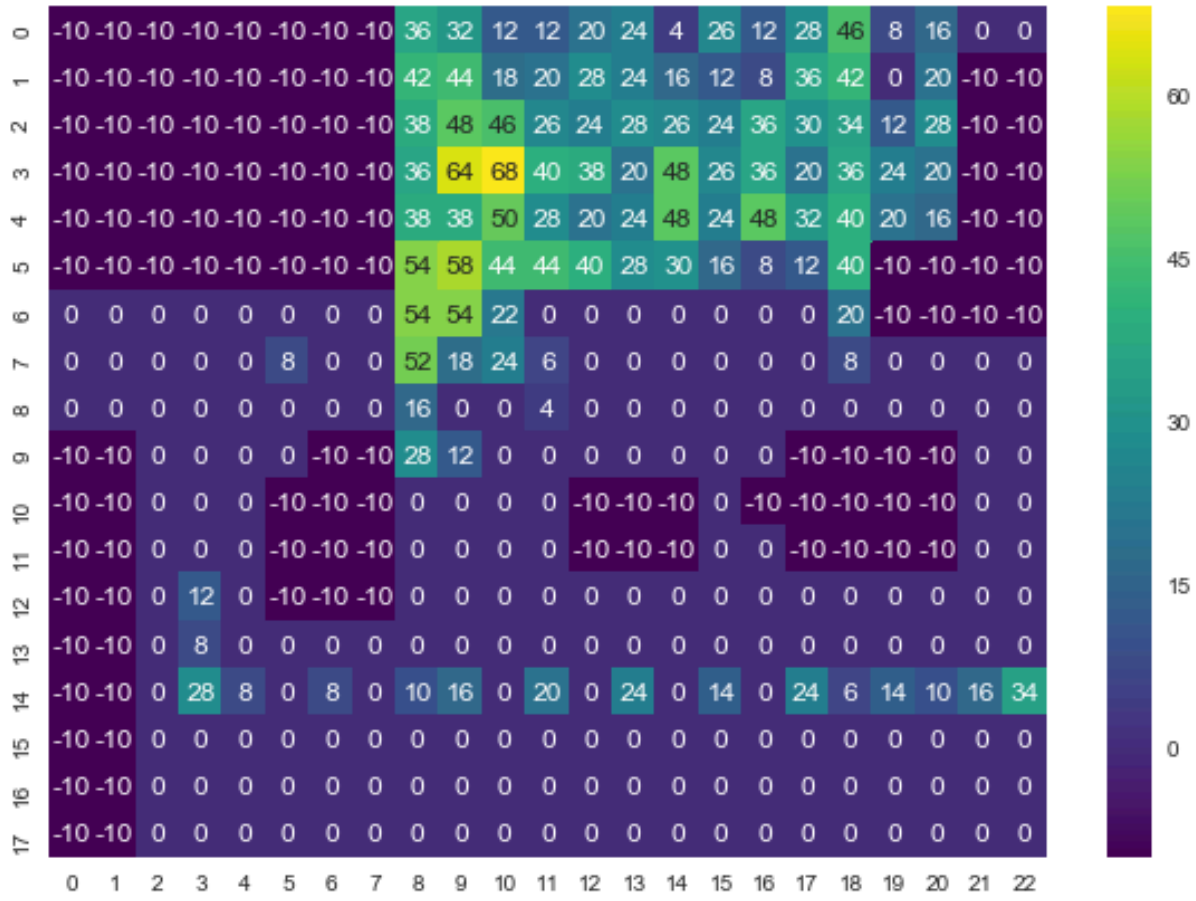

Figure 20 Distribution of labeled data points on the grid

Implementation

Pre-RL implementations

In order to implement the reinforcement learning, the following task needs to complete.

- Design and implement VAE model
- Train DNN model to predict 13 iBeacons values similar to figure 14
- Train two VAE model to predict row and column locations similar to figure 15

Figure 21 shows the design of the decoder and encoder models on the left and right respectively. The encoder model takes the input as the state array used in the reinforcement learning. There are three dense layers of 16, 32 and 64 units respectively. The output of the last dense layer goes into the mean and variance layers. The output from the mean and variance layer is then used to do a data sampling using zero mean unit variance Gaussian random distribution. The sampled data is the input to the decoder model. The decoder model decodes the input data of the encoder model using three dense layers of 64, 32 and 16 units and the output of the decoder layer is the same size of the input of the encoder model. Figure 22 shows

the structure of the VAE model using the encoder and decoder model.

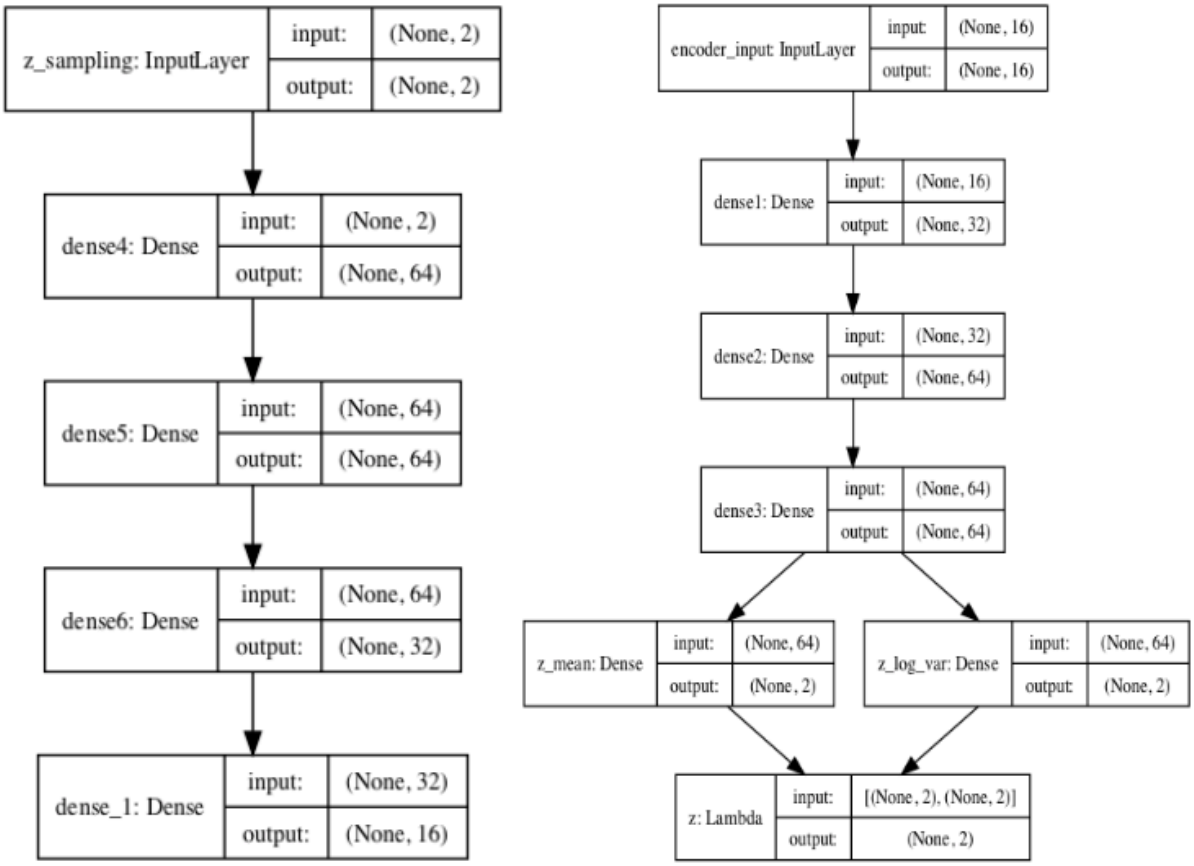

Figure 21 Decoder and Encoder DNN models

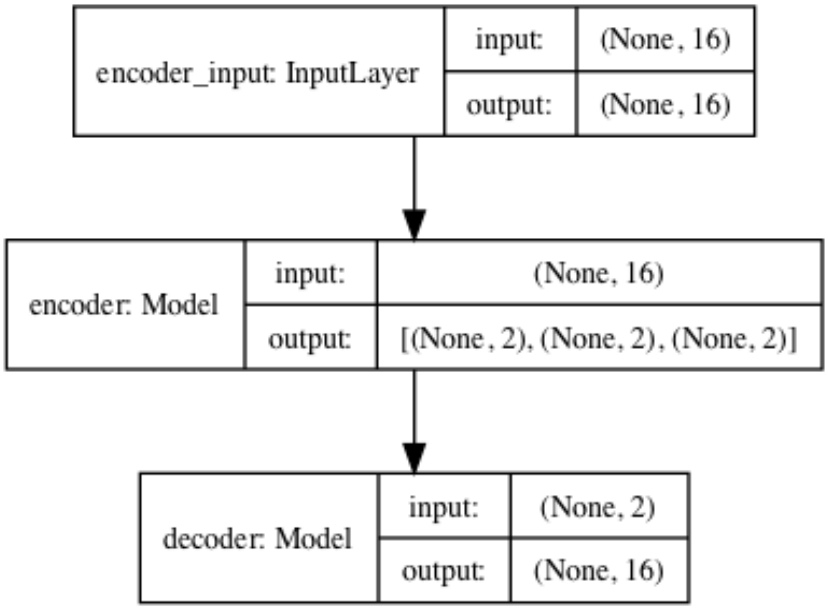

Figure 22 VAE model using Encoder and Decoder models

The VAE model needs to be compiled using Adam optimizer and custom loss function using KL divergence. The loss function using KL divergence is also included in the VAE class.

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 30)	480
dense_2 (Dense)	(None, 128)	3968
dense_3 (Dense)	(None, 256)	33024
dense_4 (Dense)	(None, 512)	131584
dense_5 (Dense)	(None, 13)	6669

Total params: 175,725
Trainable params: 175,725
Non-trainable params: 0

Figure 23 Environment DNN model to predict the 13 iBeacon values

The environment model to train an agent using reinforcement learning required a model that could predict the BLE RSSI values based on the location. In order to predict the 13 iBeacon values, a DNN shown in figure 23 is used. The input to the model is the location and the distance of the location from the thirteen iBeacon devices. The model is compiled using Adam optimized and binary cross entropy loss function. The input to the model is from the labeled dataset.

Layer (type)	Output Shape	Param #
encoder_input (InputLayer)	(None, 13)	0
encoder (Model)	[(None, 2), (None, 2), (N	43268
decoder (Model)	(None, 13)	42765
dense5 (Dense)	(None, 128)	1792
row (Dense)	(None, 18)	2322

Total params: 90,147
Trainable params: 90,147
Non-trainable params: 0

Figure 24 DNN model build on top of VAE to predict row and column

In order to predict the row and column location on the grid for the unlabeled data, a DNN model using VAE and additional dance layer and the output layer is used. Figure 24 shows the structure of the DNN model used to predict the rows and columns. Two separate models with the same structure are used to predict the row and column separately. The input data is the 13 RSSI values from the labeled data. The DNN model is compiled using the Adam optimizer with default settings and the categorical cross entropy loss function.

RL Implementation

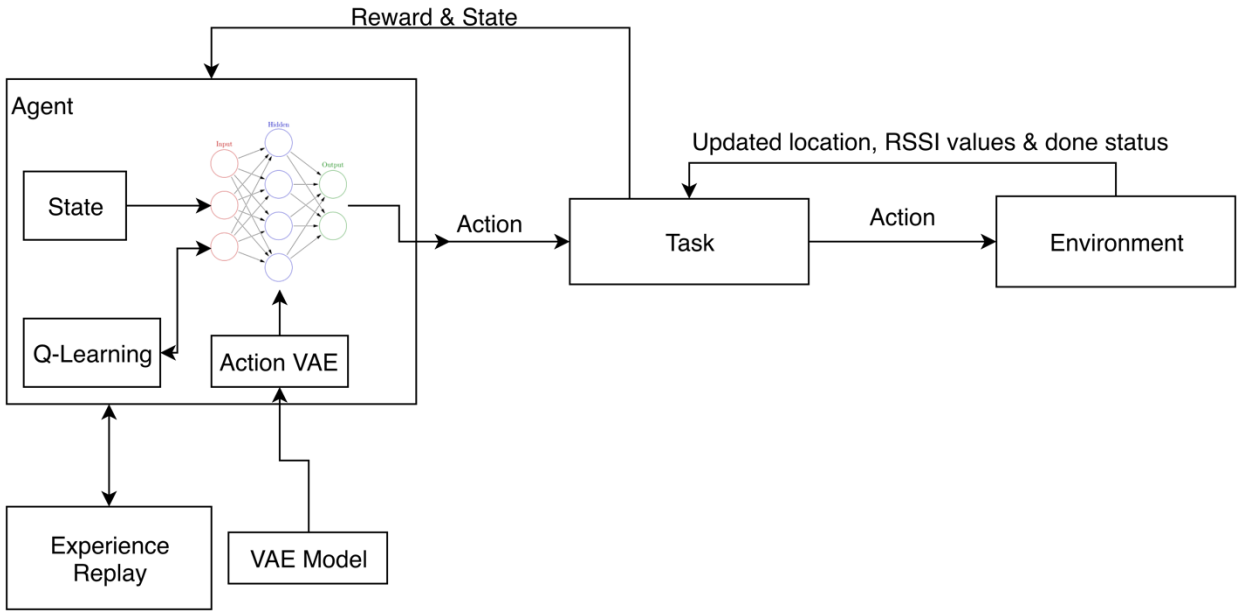

Figure 25 DQRL block diagram

The high-level structure of the reinforcement learning environment is shown in figure 25 above. The environment class takes action as input and updates the status of the agent's position like done, position and RSSI values. The task class uses the updated status of the agent from the environment class and calculates rewards, current state, next state and done. The agent uses the calculated rewards and state values by the task class. The agent saves the batches of state, action, reward, next state and action for experience replay. Also, the agent learns from the experience replay using Q-learning. The deep learning model is derived from the VAE model described above. On the top of the VAE model, there are three dense layers as shown in figure 26. The output of the Action VAE model is the number of action categories which is set to four.

Table 2 Action table

Number	0	1	2	3
Action	East	South	West	North

Tables 2 shows the meaning of the action values used in the environment class. Action 0, 1, 2, and 3 means moving to east, south, west, and north respectively.

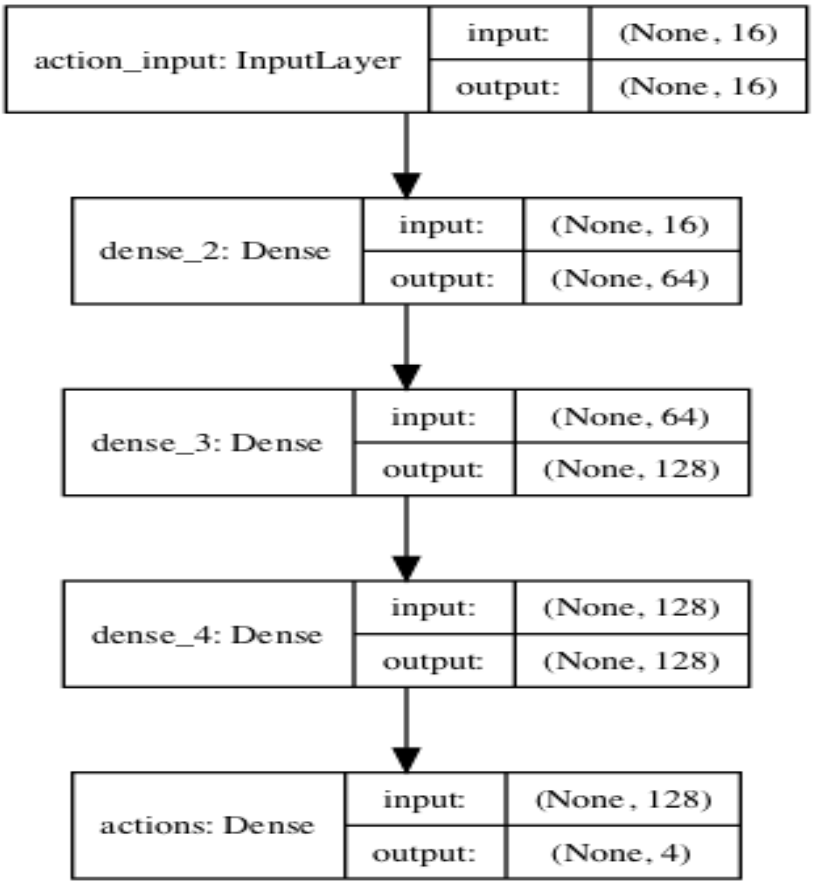

Figure 26 Action VAE model design

Table 3 State value table

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
State Value	RSSI 1	RSSI 2	RSSI 3	RSS 4	RSSI 5	RSSI 6	RSSI 7	RSSI 8	RSSI 9	RSSI 10	RSSI 11	RSSI 12	RSSI 13	Row	Column	Distance

States: The state of the agent is represented as a tuple of these observations.

1) A vector of 13 RSSI values.
2) Current location (identified by row and column numbers).
3) Distance to the target.

As shown in table 3, the state is composed of thirteen RSSI values from each iBeacon in addition with the position related to the RSSI values and the distance of the position from the target location.

Reward: The reward of the agent's movements on the grid is calculated as shown below.

$$Rewards = \begin{cases} \pm\, Distance\ (\|P_C - P_T\|) \\ 10 \quad if\ 0 < Distance < 12 \\ 20 \quad if\ 0 < Distance < 04 \end{cases}$$

where P_C is the current position and P_T is the target position of the agent. The Distance is the distance between the agent's current and target positions. The distance is calculated in meters.

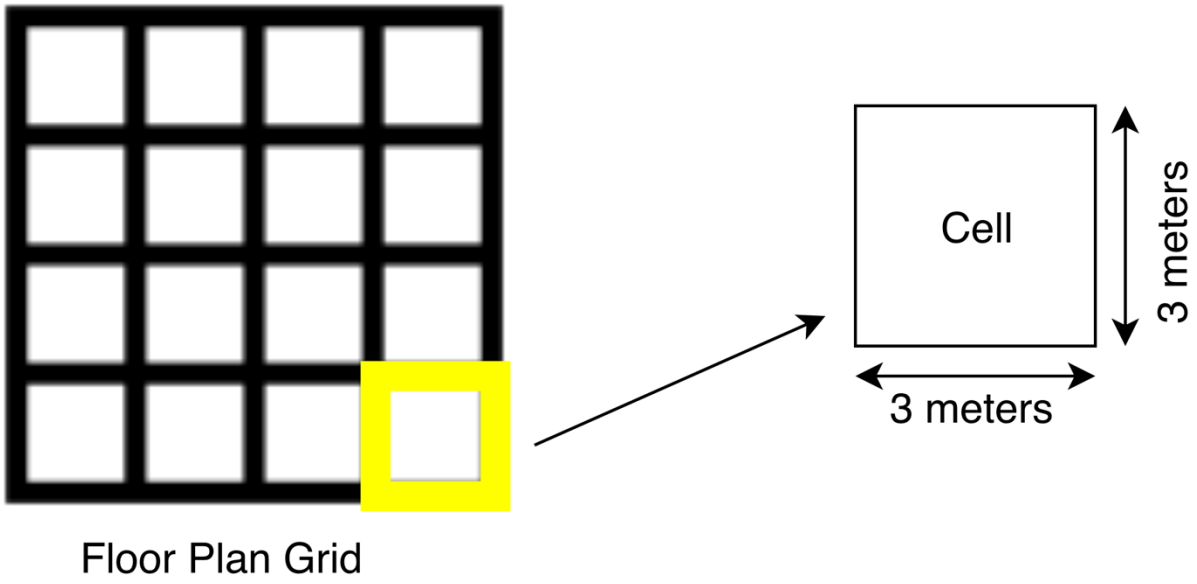

Figure 27 Cell dimension of a grid

As shown in figure 27, each cell of the grid is of 3×3 meters in dimension. Therefore, the distance measured between two cells is multiplied by three.

Figure 28 Non-linear model to predict the RSSI values based on distance

$$RSSI = \frac{1.1}{e^{0.1 * Adjusted\ Distance}} - 0.03 * 5$$

Equation 2 Non-linear modeling of RSSI based on adjusted distance

where Adjusted Distance was calculated as described next. First, an array of the distance of the current position to the thirteen iBeacons is calculated. The minimum distance of the array is then subtracted from all the elements. The resulting distance vector represents the adjusted distance of the current position to the iBeacon device positions. The adjusted is then passed through the equation 2 to calculate the RSSI values. The vector of RSSI values then passes through a simple filter to remove unnecessary iBeacon values. Figure 28 shows the plot of the RSSI values shown in equation 2.

The following algorithm is used to implement Deep Q-Learning to train the agent to get to destination using labeled and unlabeled dataset. The algorithm structure is derived from the one shown in the article (Mohammadi, Al-Fuqaha, Guizani, & Sorour, 2018); however, the implementation is completely different.

Algorithm: Semi supervised DQRL Algorithm

1: Input: A dataset of labeled and unlabeled data {(X_l, Y_l), X_u}
2: Initialize the model parameters learning rate, DNN parameters, task, environment, state space, and replay memory D
3: **for** *episode*←1 to *M* **do**
4: $s_0 \leftarrow$ make observation of sample x
5: **for** $t \leftarrow 0$ to T **do**
6: Take an action a_t using ε-greedy strategy
7: Perform action a_t to change the current state s_t to the next state s_{t+1}
8: **if** sample is unlabeled **then**
9: Infer the label using a model $q_\phi(y|x)$ and get approximate reward $r_t = closeness(s_{t+1}, y)$
10: **else**
11: Observe reward r_t that corresponds to label y
12: Store transition (s_t, a_t, r_t, s_{t+1}) in D
13: Take a random minibatch of transitions (s_k, a_k, r_k, s_{k+1}) from D; $0 < k \leq length(minibatch)$
14: **if** s_{k+1} is a terminal state **then**
15: $\xi_k = r_k$
16: **else**
17: $\xi_k = r_k + \gamma \max_{a'} Q(s_{k+1}, a'; \theta)$
18: Apply gradient descent on $(\xi_k - Q(s_k, a; \theta))^2$ based on (13)
19: **end for**
20: **end for**

Table 4 Hyperparameters of DQRL model

Variable	Value
Starting Position P_S ([row, column])	[02, 14]
Target Position P_T ([row, column])	[17, 10]
Distance from P_S to P_T (meters)	46.5733
Learning Rate LR	0.001
ε , ε decay rate	1, 0.995
Gamma (Discount factor Q-Learning)	0.95
Memory batch, buffer size	80, 10000

Refinement

There were countless number of refinement processes throughout the project. The project contains the implementation of many DNN models. Implementation of each network took multiple trials. For example, the implementation of the VAE model to predict rows and columns took multiple trials to adjust dense layer with minimum units and maximum accuracy.

The modeling of the DNN to predict the RSSI values did not work well. Hence, I had to model the RSSI values for a particular location mathematically. I started with simple linear models; however, the relationship shown in equation 1 led me to model the prediction of RSSI values as a non-linear complex exponential function. The idea of the distance multiplied by three did not occur initially. After some tries, I realized that the distance calculated does not seem practical. After researching the floor plan, I had to adjust the distance by multiplying by three.

The action has four categories that include moving north, south, east and west. However, initially, I implemented action in eight categories. The eight categories were north, south, east, west, northeast, south-west, north-west and south-east. The agent with eight action categories did not learn well. Even after tuning the hyperparameters like learning rate, rewards, policy, there was no improvement in the learning of the agent. After implementing four categories, the agent showed improvement in learning by traveling more toward the target.

The reward function also took some tuning. In my initial trials, the reward function had a design such that most of the time the rewards were negative. Hence, I had to change the rewards function multiple times to train the agent.

In addition to refinement in the models and the hyperparameters of the RL, I had to test and modify many small functions used in the pre-training process.

Results

Model Evaluation and Validation

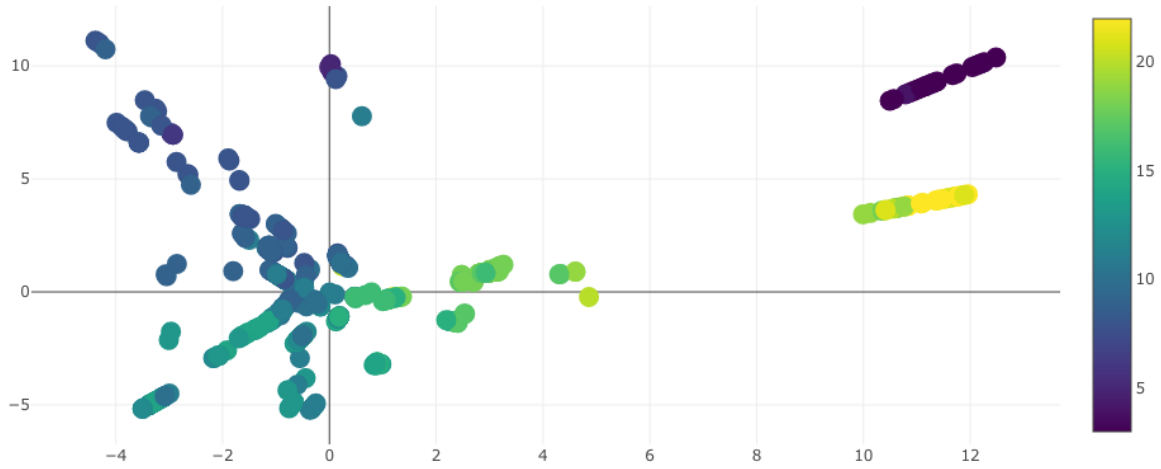

Figure 29 Latent distribution of column prediction VAE model

Figure 29 shows the distribution of the latent variables using the lambda function in VAE model. The latent variables for column predicting VAE model are distributed using zero mean unit variance Gaussian random distribution. The color bar on the side of figure 29 shows the column number.

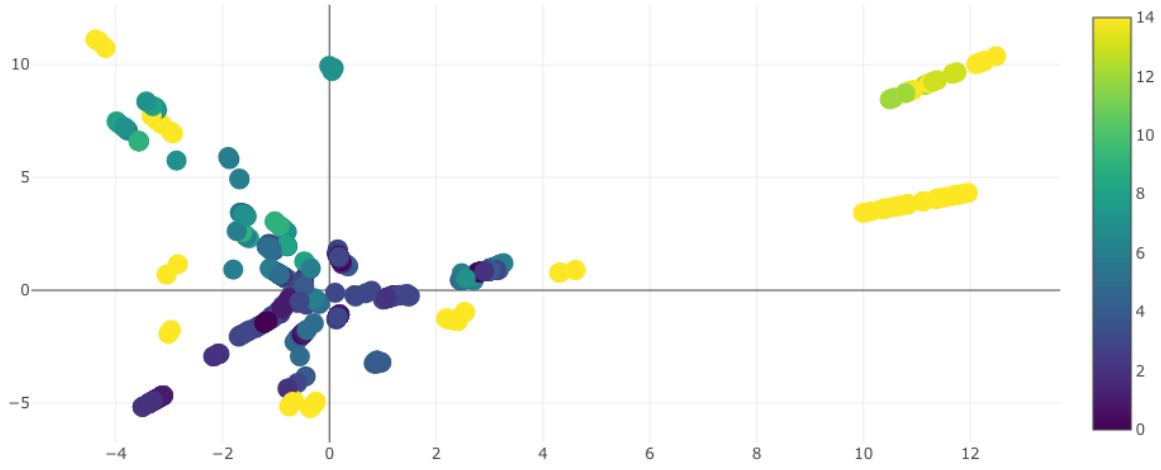

Figure 30 Latent distribution of rows predicting VAE model

Figure 30 shows the distribution of the latent variables using the lambda function in VAE model. The latent variables for row predicting VAE model are distributed using zero mean unit

variance Gaussian random distribution. The color bar on the side of figure 30 shows the row number.

The main purpose for using the VAE model is to get the distribution of the latent variables as shown in figures 29 and 30. The distribution of the latent variable provides the power of knowledge to the DNN to predict the positions for which there is no data available. The prediction of unseen data through VAE model provides a stable and converging learning experience for the RL agent to train on.

Figure 31 shows the performance of the Deep Q Reinforcement Learning (DQRL) for the RSSI data of both labeled and unlabeled dataset. The top plot is the rewards plot for every, and the bottom plot is for the distance traveled towards the target per episode. The primary performance metric for the project is the distance toward the target. From figure 31, the average distance the agent traveled toward the target is about 25 meters. Here, the distance of zero means the agent has reached the target. Hence, the lower the distance the better the performance of the agent. The distance graph is shaky at the beginning where most of the steps were taken randomly as per greedy policy. As the agent learned through Q-Learning and took less random steps, the agent consistently traveled closer to the target.

Figure 31 Performance of RSSI data of labeled and unlabeled dataset in terms of rewards and distance

The data provides in the labeled, and unlabeled dataset seemed to have much noise. One location had multiple uncorrelated RSSI data vectors. Also, the RSSI data vectors did not follow the relationship shown in equation 1. Therefore, I trained the DQRL model based on the distance of the position to each iBeacon instead of RSSI values to cross check the model performance. The performance result of the consequent model is shown in figure 32.

Figure 32 Performance of distance from iBeacons data of labeled and unlabeled dataset in terms of rewards and distance

As shown in figure 32, the learning from distance dataset results are very similar to the figure 31. Therefore, the model performance is validated. The noise in the data dis not affect the performance of the agent much.

Figure 33 Performance of distance from iBeacons data of only labeled dataset in terms of rewards and distance

Figure 33 shows the performance of the DQRL model with the only labeled dataset. Both the rewards and the distance plots are shaky through all the episodes. The average distance traveled by the agent using the labeled dataset is about 38 meters. As discussed before, the lower the distance, the better the performance. Since 38 is larger than 25, the performance of the agent degraded using only the labeled dataset. Also, the performance of the agent was not consistent over the period of one thousand episodes. Therefore, the DQRL agent struggled to navigate to the target using the labeled dataset only.

Justification

Table 5 Supervised & semi-supervised Learning comparison

	Average Distance (m)
Semi-Supervised Learning (Labeled & unlabeled dataset)	46.57 – 24.60 = 21.97
Supervised learning (Labeled dataset only)	46.57 – 36.20 = 10.37

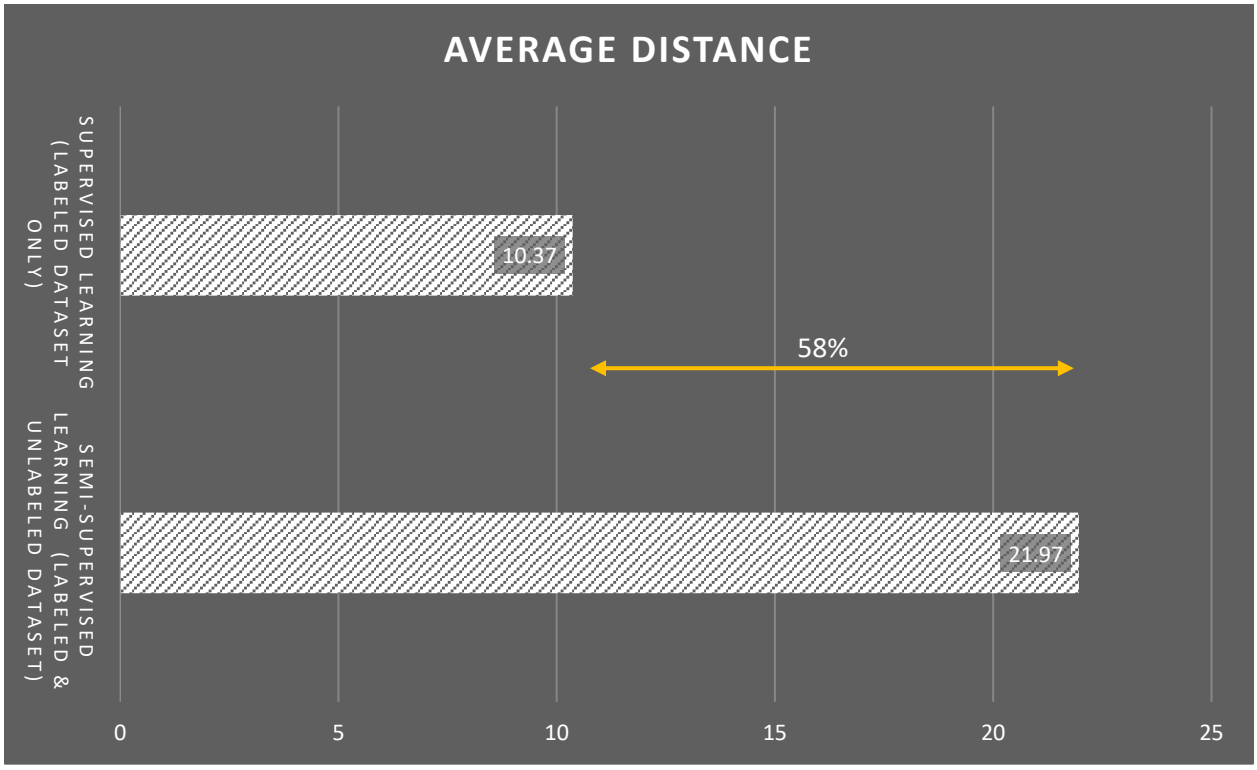

Figure 34 Supervised and semi-supervised learning performance comparison

Table 5 shows the calculation of distance traveled toward the target by the agent trained on the labeled dataset only and both labeled and unlabeled dataset. The supervised learning refers to the agent trained only on the labeled dataset while the semi-supervised learning refers to the agent trained on the combination of labeled and unlabeled dataset. As shown in figure 34, the semi-supervised trained agent was able to travel about 21.97 meters of distance toward the target while the supervised, trained agent was able to travel only 10.37 meters toward the target. Hence, the semi-supervised learning model increased the performance of the agent by 58%.

Comparing figure 34 to the benchmark shown in figure 16, the DQRL semi-supervised learning agent in figure 34 performs far better and travels 22 - 8 = 14 meters more than the benchmark agent shown in figure 16.

The primary goal of implementing semi-supervised learning is to improve the performance of the algorithms using unlabeled data. The performance shown in figure 34 shows that the semi-supervised learning approach succeeds in increasing the accuracy of the algorithm in achieving the goal.

Conclusion

Free-Form Visualization

As shown in figure 35, the semi-supervised learning approach provides the information about almost all the grid points. Number one on the grid shows that the grid cell has a data point while zero means no data point for that cell. The value negative ten represents an object on the grid cell. Therefore, the agent learns better and faster. As a result, the performance of the algorithm increases. Even though the data about most of the grid cell is available in the semi-supervised learning, some of the data could be redundant and inaccurate due to out of bound latent distribution. In out of bound latent distribution, the distribution of the input data for one grid cell overlaps with another grid cell. Resulting in the same output for two different inputs. Even after facing such challenges, the semi-supervised learning model provides better performance than the supervised learning model.

Figure 36 shows that the bottom half of the grid has very few data points in the labeled dataset. Therefore, the supervised learning model struggles for the navigation of the agent where the data points for the grid cells are low. The performance numbers shown in figure 34 reflects the challenge of the low grid points for supervised learning. The supervised learning's low performance is mainly due to the small and concentrated dataset.

Both figure 35 and 36 summarizes the performance of the DQRL agent just by the visualization.

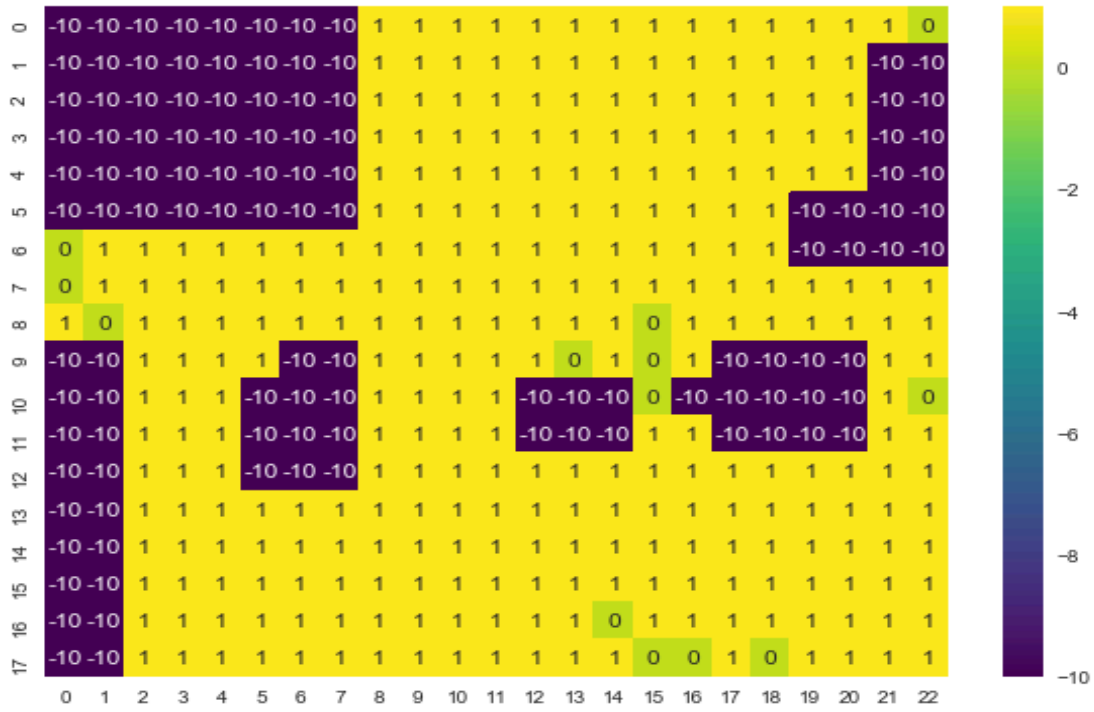

Figure 35 Semi-supervised learning grid data availability

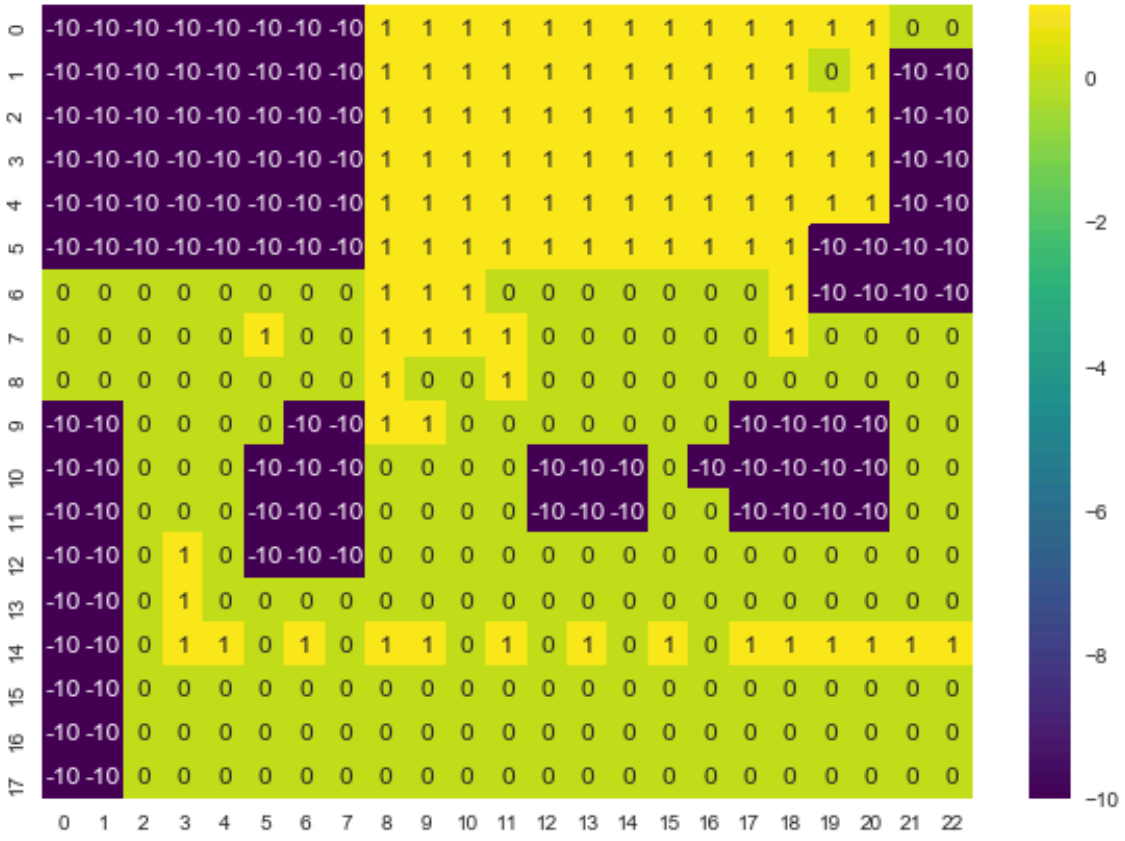

Figure 36 Supervised learning grid data availability

Reflection

The process of solving the problem of indoor navigation is shown below.

1. Reading the article and understanding the problem
2. Analyzing the article for additional data extraction from images like floor plans
3. Data exploration and correlation analysis, implementation and experimentation
4. Data saving and loading technique
5. Design various models like mathematical, DNN and others to small problems within the project
6. Design of RL process
7. RL analysis, statistics
8. Improvement based on observed and expected results

The process of finishing the project included many trial and error steps. The process shown above were not derived right in the beginning. The steps like 4, 5 and 8 were realized during the project. The modeling of the functions like predicting 13 iBeacon values included extensive research on DNN model implementation using Keras, the choice of the optimizer and the loss function, etc. The design and operation of DQRL environment took a lot of effort and time.

Even though the project was full of challenges and experimentation, I enjoyed every part of it. The excitement of learning on my own and solving the problem to its full potential kept my moral high. The documentation also made me relies on some processes like saving the graphs even if the graphs do not make sense. Visualization of the progression of improvement is easy to show than explain. Overall, I enjoyed the project, and I am sure the learning experience of the project will be helpful to me in the future.

Improvement

Even though I tried many experimentation and innovation in the project, there are still a few points that could be improved if implemented correctly.

- Trying the different combinations of input features for the DNN and other models to improve performance. For example, the VAE DNN model to predict row and column can be further improved by adding additional features.
- The non-linear modeling of the function can be enhanced using supervised learning model shown in the Data_testing.ipynb file of the project. I found that the relationship between RSSI values and distance was different from the one shown in equation 1. A robust supervised learning model or unsupervised learning techniques can be used to extract some new information that could be specific to a place and use the information to improve the accuracy of the algorithm.
- In DQRL, the state size is the only variable I did not change throughout the process. I strongly feel that reducing the number of elements in the state vector could improve the performance. However, the reduction in state vector needs better modeling of the environment class.

References

[1] M. Mohammadi, A. Al-Fuqaha, S. Sorour and M. Guizani, "Deep Learning for IoT Big Data and Streaming Analytics: A Survey," in *IEEE Communications Surveys & Tutorials*. doi: 10.1109/COMST.2018.2844341
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8373692&isnumber=5451756

[2] M. Mohammadi, A. Al-Fuqaha, M. Guizani and J. S. Oh, "Semisupervised Deep Reinforcement Learning in Support of IoT and Smart City Services," in *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 624-635, April 2018.
doi: 10.1109/JIOT.2017.2712560
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7945258&isnumber=8334665

[3] Link to the dataset:
URL:https://archive.ics.uci.edu/ml/datasets/BLE+RSSI+Dataset+for+Indoor+localization+and+Navigation

Table of Figures

Table of Tables