Bachelor Thesis

RSSI-based Indoor Localization Using Machine Learning and Deep Learning

by

Máté Nagy

(2697010)

First supervisor: Kees Verstoep

Daily supervisor: Kees Verstoep

Second reader: dr. Lin Wang

July 10, 2023

Submitted in partial fulfilment of the requirements for the degree of Bachelor of Science in Computer Science

Abstract

This thesis presents a comparison of 4 different machine learning models: Random Forest, Decision Tree, Support Vector Machine, K-Nearest Neighbours and a Deep Learning model to evaluate their performance in predicting the indoor positioning of a device based on Wi-Fi RSSI data. The study evaluates 3 different approaches: making predictions based on purely the RSSI data, predictions using the orientation of the device and predictions with the use of the location data of the fixed access points installed in the building. The RSSI data was collected and applied to all models to understand the impact of machine learning and deep learning in accurately predicting Wi-Fi-based indoor localization. The best resulting model was Random Forest in all 3 approaches with an accuracy of 97.34%.

1 Introduction

In recent years navigation has gone through some great advancements due to the rise of technology. Global Positioning Systems (GPS) have advanced greatly and are a crucial part of our daily lives helping us navigate around our world. However, when it comes to navigation in indoor environments GPS-based navigation systems are not as reliable as in outdoor environments due to the buildings' blocking and weakening effect that is caused by their construction materials. This provides a great disadvantage to individuals attempting to get around in shopping malls, hospitals, airports, large office buildings or university buildings.

To overcome these limitations, researchers and companies have been exploring what are the possibilities to provide accurate localization in indoor environments. Although there is not one single standard yet, many possible solutions are available. Some of these include the use of technologies such as Bluetooth Low Energy (BLE), Ultra-Wideband (UWB), Radio Frequency Identification (RFID) and Wi-Fi [11] that are widely available in most of the larger buildings. Some of the techniques that can utilize these technologies are Received Signal Strength Indicator (RSSI), Round Trip Time (RTT) and Trilateration. RSSI measures the power of the radio signal emitted by the beacon or wireless device, while Round Trip Time measures the time it takes for a signal to travel from the transmitter device to the receiver device by recording timestamps [7]. Trilateration on the other hand works by measuring the distance to three anchor nodes whose positions are already known, by measuring RSSI and calculating the intersection point of them [8]. Even though these methods can give a location estimate, they are not reliable and accurate enough for indoor environments. Therefore, applying technologies such as machine learning and deep learning could yield better accuracy in predicting a device's

position. Machine learning and deep learning have been gaining a substantial amount of attention and development over the past decade, also in this domain.

This thesis aims to investigate the impact of applying machine learning and deep learning to accurately determine the indoor location of a mobile device. The thesis starts by building a basic indoor RSSI-based localization model using Wi-Fi – based on a previous study [3] that also used RSSI – and compares its precision with the literature. Then we will investigate the impact of the following modifications to our basic model:

- the use of machine learning versus deep learning models;
- the impact of the orientation of the mobile device;
- the use of absolute AP location information versus a fingerprinting approach;
- and using trajectory estimates based on a sequence of measurements in contrast with independent localization estimates.

By addressing these questions this study aims to provide an insight into the accuracy of using machine learning and deep learning to solve the challenges of indoor localization.

The thesis starts by reviewing related literature in section 2, then in section 3 describing the methods used to collect the data, how the models were created and eventually how the output location was created and then compared to other models. After that, the results will be presented along with an experiment in section 4 followed by a discussion in section 5 and a conclusion in section 6.

2 Related Work

In this section, an overview will be provided of related studies and methodologies that have contributed to the domain of indoor localization using the Wi-Fi RSSI method. By examining the relevant literature, we have the intention to highlight the key findings of other studies.

The study [1] explored the potential of using a fingerprinting method on Wi-Fi RSSI values from various access points where deep learning was utilized. Their model demonstrated an accuracy of 95.95%. The findings of the study seem promising and suggest that using deep learning for predicting indoor positions can be a good approach.

This project [2] investigated the impact of using Wi-Fi-based indoor positioning with the main focus being on the device's orientation. The study uses a fingerprinting approach on RSSI gathered from Wi-Fi Access Points to determine the device's location and orientation within 1.8 meters in an elderly centre. The results show that the orientation and the location of the device can be distinguished with their approach.

Another study [3] explored the impact of using a trilateration method and the use of a deep learning model where both approaches used a dataset containing RSSI values amongst other features. The research concludes that the application of deep learning had a significant impact on the accuracy of predicting the device's position in the university building over the naive trilateration approach. It was highlighted, however, that the deep learning model is less generalizable and computationally extensive. Furthermore, the study of the author was also carried out in the same building as this study, however, the data collection took place on one floor while this study did data collection on multiple floors.

A study [5] proposed the use of a Deep Neural Network (DNN) for a multi-building environment. It is using a multilayer perceptron (MLP) algorithm and it showed a 99.15% accuracy with a classification-based approach and a 93.1% accuracy for smaller than 0.9 meters estimation errors for the regression-based approach. The neural network had 4 hidden layers and used a rectified linear activation function for all hidden layers.

A paper by Kyeong Soo Kim, Sanghyuk Lee & Kaizhu Huang [6] also investigated the performance of a deep neural network in a multi-building and floor environment with WiFi RSSI fingerprinting. Their results demonstrated near state-of-the-art results that can be implemented with lower complexity on mobile devices.

A paper by Aigerim Mussina and Sanzhar Aubakirov [15] explored how applicable machine learning is for indoor localization problems with a classification-based approach. The paper used Naïve Bayes and Support Vector Machine algorithms and achieved 95.8% accuracy.

The study by Vladimir Bellavista-Parent, Joaquín Torres-Sospedra, and Antoni Pérez-Navarro [19] provided an analysis of over 100 papers written about the application of machine learning for indoor positioning. The analysis found that the most commonly used parameter is RSSI which is showing promising results. Furthermore, it found that there is a tendency for using Neural Networks for indoor positioning while Support Vector Machine is also widely used. It also found that Mean Squared and Accuracy are the most commonly used metrics for evaluating the performance of such models. Additionally, most of these papers prioritized improved results over working in real environments.

3 Method

This section describes how the data was collected and pre-processed. Furthermore, it will describe the fine-tuning and training of the machine learning models created as well as the

hyperparameter tuning and training of the deep learning model. Lastly it will describe the process of carrying out an experiment.

3.1 Data Collection Application

For data collection, an Android application was created [16] using Android Studio. To test the application an Android emulator with Android 10 (API level 29) was used and a Google Pixel 2 that also has Android 10 as its operating system. Even though there are newer versions of Android available, this device only supports Android up to version 10. Furthermore, it was important to use Android 10 for this application since this version introduced a new developer option in the phone's settings to turn off throttling for local testing of the Wi-Fi scanning capabilities provided by the WifiManager API. The application was able to scan for Wi-Fi signals from surrounding access points and gather information about them, most importantly the RSSI of each access point. This data was written to the test phone's storage in CSV format for later use.

3.2 Data Collection

The data collection process involved the use of the Android application made for the data collection. The data collection process took place at the Vrije Universiteit Amsterdam on the 5th and 6th floors of the NU building. These two floors were selected due to their versatility of room sizes and shapes, isles, and open study areas. There are 29 access points on the 5th floor and 34 access points on the 6th floor of the building. On the 5th floor 30 measurement points and on the 6th floor 31 measurement points were selected. These points were distributed through the floors to have data on most of the rooms and spaces. At each measurement point, 60 Wi-Fi RSSI scans were made resulting in a total of 3660 instances. Each instance that represents a scan involved information on the RSSI of a given access point. Access points that were too far for the Wi-Fi scanner to pick up their signal were represented with a 1 in the dataset and the ones that were close enough to the scanner had their RSSI in the dataset. Moreover, each instance held the location information of the measurement point represented by the X, Y and Z coordinate values of the building. The X and Y coordinate values represent 1-meter distance in the building and the Z value represents the floor where the measurement was taken.

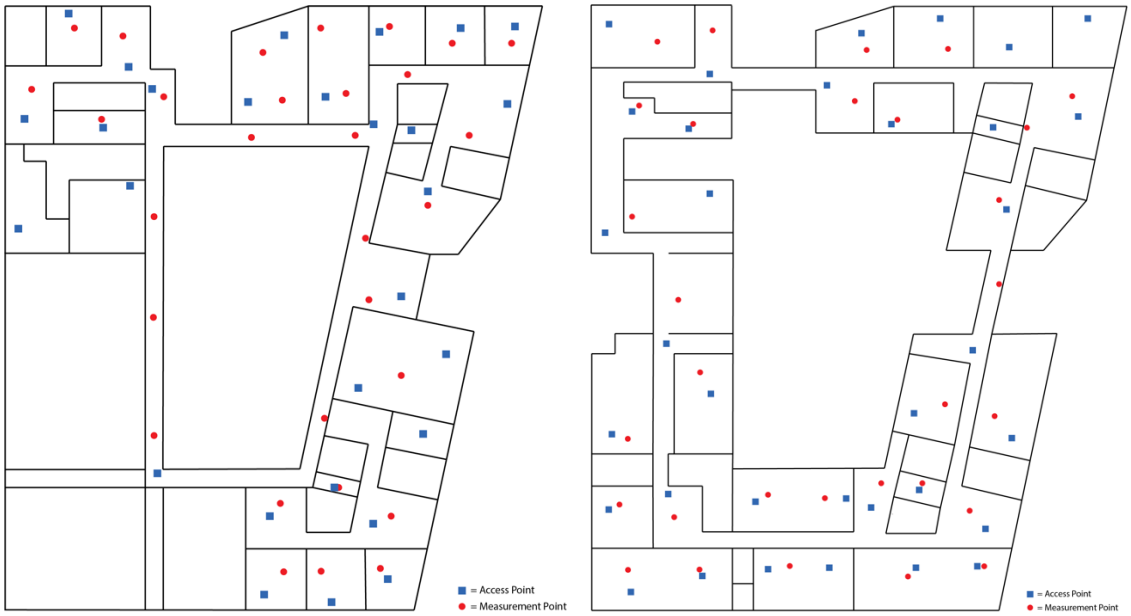

Figure 1: Location of Access Points and Measurement Points on the 5th and 6th floors of the building

Additionally, to record the directionality of the device after every 15th scan the device was oriented towards the next cardinal direction starting from north. The intention of collecting this data was to investigate the possible influence of orientation, and potentially improve the results.

3.3 Data preparation and pre-processing

When the data was collected, it was transformed [17] using the programming language Python and with the help of several Python libraries such as NumPy and Pandas. Since the data was in many separate CSV files, they had to be organized into one data frame containing 3660 instances and 327 columns. Each row contained information about each AP's RSSI in the whole building and the location of the measurement. Then the two floor's coordinate values could be added to the data frame to enrich the dataset with additional features. Furthermore, the directionality of the device had to be represented by numerical values, therefore when the device was facing north the directionality was expressed as a 0, when facing east with the value 90, when facing south with the value 180 and lastly when facing west with 270.

After all the data was prepared, a scaling procedure was implemented to normalize the values within the range of 0 and 1. Scaling is the process of transforming the values of the features of a dataset to a standardized range such that they are easily comparable. It allows for the prevention of skewed results and better performance of the machine and deep learning

models. Finally, the dataset got split into a training set and a test set with sizes of 80% and 20% respectively.

3.4 Machine Learning Hyperparameter Tuning and Training

After the data has been prepared, scaled, and split, 4 different machine learning models were created. These models were created with the use of scikit-learn, an open-source Python library used for data analysis and machine learning solutions. Since the basic models' goal is to predict numerical coordinates for an indoor environment a regression-based approach was used on all 4 methods. The 4 methods were the Decision Tree algorithm, Random Forest algorithm, Support Vector Machine algorithm and K-Nearest Neighbours algorithm. These models were selected due to literature [9] and [10] discussing the performance of these machine learning algorithms. The Decision Tree algorithm is a supervised machine learning method that uses a tree-like structure. Each node in the tree represents a decision and the leaves of the node describe the decisions [12]. The Random Forest algorithm combines the output of multiple decision trees to make its predictions by randomly extracting data from the dataset [13]. The Support Vector Machine algorithm (SVM) works by attempting to find a hyperplane in an N-dimensional space where the data is linearly separable [12]. The K-Nearest Neighbours algorithm (KNN) operates on the principle of assigning a value to a class based on its k nearest neighbours in the training set that are the most similar [14]. In addition to the mentioned models, a multi-output regressor was used to allow the models to have 3 outputs since not all models support multidimensional output.

After the models were created the hyperparameter tuning was applied. For this purpose, grid search was used to optimize the performance of our models. When the best hyperparameters got determined the models got adjusted with them to achieve a better performance and accuracy on the data (see Appendix for the hyperparameters used and their best settings).

3.5 Deep learning Hyperparameter Tuning and Training

After fine-tuning and training the machine learning models a deep learning model was created with the use of Keras, an open-source library to build artificial neural networks that acts as an interface for the TensorFlow library. For determining the best hyperparameters Keras Tuner was used, a hyperparameter tuning framework. This process involved defining the hyperparameters which were the number of layers, the number of neurons, if the network should have batch normalization or not, the dropout rate, the learning rate, and the number of epochs as well as the batch size. The activation function of the search space was set to ReLU

(Rectified Linear Unit) which sets all of the negative values it gets to zero and leaves the positive ones unchanged. For the output layer linear regression was used as its activation function. When the search space was defined with the range of values of the hyperparameters a tuner had to be chosen. For this purpose, Bayesian optimization was used which operates on a statistical principle and utilizes an exploration-exploitation method to find the most optimal hyperparameters. The main objective of the optimizer was to minimize the Mean Squared Error (MSE) that measures how close the regression line gets to the data points of the dataset. Next, the tuner was configured to have a maximum of 5 trials to test a set of hyperparameters with 3 executions per trial.

After the search for the hyperparameters, the model was trained on various epochs and batch sizes with the option of shuffling the data at each epoch. Once this training was done the model was retrained with the best hyperparameters that were achieved from the previous training. While the above-mentioned study [5] had 4 hidden layers, this thesis' model had 3 hidden layers with 208, 280, and 352 neurons respectively. 10% of the neurons were excluded from the network specified by the 0.1 dropout rate. The dropout rate helps to generalize the model and avoid overfitting. Furthermore, the learning rate of the model was 0.01 which manages at what pace the algorithm should update its values. The optimizer used for the training of the model was the Adam optimizer, a stochastic gradient-based optimizer. The model also had a batch size of 80 and 275 epochs for an efficient training process.

Lastly, the model got evaluated on its performance and using the test set, predictions were made for further evaluation to calculate its accuracy with the R-squared statistical measure.

In the end, the predictions with the actual results got visualized in 3-dimensional space and the training and validation loss were plotted against each other.

3.6 Trajectory-based approach

As an experiment, instead of only making predictions on data that was collected by taking measurements in fixed locations multiple times, a different approach was also applied. On a selected route on the 6th floor of the NU building, 31 walks were completed between two points while repeatedly making Wi-Fi scans. To make repeated Wi-Fi scans the Android application was adjusted such that after one scan returned all the information about the access points another scan was automatically initiated. This resulted in 100 scans that varied in where the scan took place.

Since the scans were not made in static locations and the locations of the scans were unknown, the start time and the end time of the walk were recorded as well as when the scans

returned the results. Based on this information the time it took to complete the walk and the walking speed could be calculated. Furthermore, due to knowing the start and end locations, it was possible to calculate the scan locations as well and finally round them to whole numbers.

Then the resulting dataset's values were scaled and split into features and target data frames. To measure the performance of this data on the basic models the predictions were made on all 4 fine-tuned basic machine learning models and also the fine-tuned deep learning model. Additionally, some post-processing could be applied where the Y and Z coordinates got adjusted to the walking route to improve the accuracy of the experiment.

Lastly, the predictions and the actual locations of the scans were plotted on a map of the 6th floor to visualize the results.

4 Results

In this section, the performance evaluation of all approaches will be presented. The goal is to assess the accuracy and effectiveness of the models for accurately predicting the indoor position of the device from the RSSI values retrieved from the access points installed in the building. The results will be evaluated on the R-Squared (R2) value of the model which is a statistical measure describing how well a regression-based model is capable of predicting the outcomes. The other metric that will be used is the Mean Squared Error (MSE) which is a measure used to determine how close the regression line is to the data points.

4.1 Basis Models

First, the basic models' performance will be evaluated, namely the four machine learning models namely Random Forest, Decision Tree, Support Vector Machine, K-nearest Neighbours, and the Deep Learning model. To compare the models, the mean and standard deviation of the test MSE and R2 scores are reported over five runs of the models in Table 1.

Model	MSE	R2
Random Forest	0.001 ± 0.00001	97.34% ± 0.03
Decision Tree	0.003 ± 0.0001	92.28% ± 0.27
Support Vector Machine	0.004 ± 0	61.83% ± 0
K-Nearest Neighbours	0.002 ± 0	94.86% ± 0
Deep Learning	0.003 ± 0.0003	92.5% ± 1.23

Table 1: Mean and standard deviation of MSE and R2 score of the basic models.

As can be seen from Table 1, Random Forest outperforms the other models with the highest R2 score of 97.34% and the lowest MSE being 0.001 shown in Figure 1. This demonstrates that Random Forest is the most suitable model for a purely RSSI-based solution that does not include additional features. The second-best model is K-Nearest Neighbours with its R2 score of 94.86% and its MSE of 0.002. This suggests that K-Nearest Neighbours is also capable of accurately providing indoor location estimates. The deep learning model seems to be able to provide a somewhat accurate estimate, however, its standard deviation on the R2 score is quite high compared to the other models. With a 92.28% R2 score and a 0.003 MSE the Decision Tree has quite similar results compared to the Deep Learning model except for their standard deviations which are significantly different.

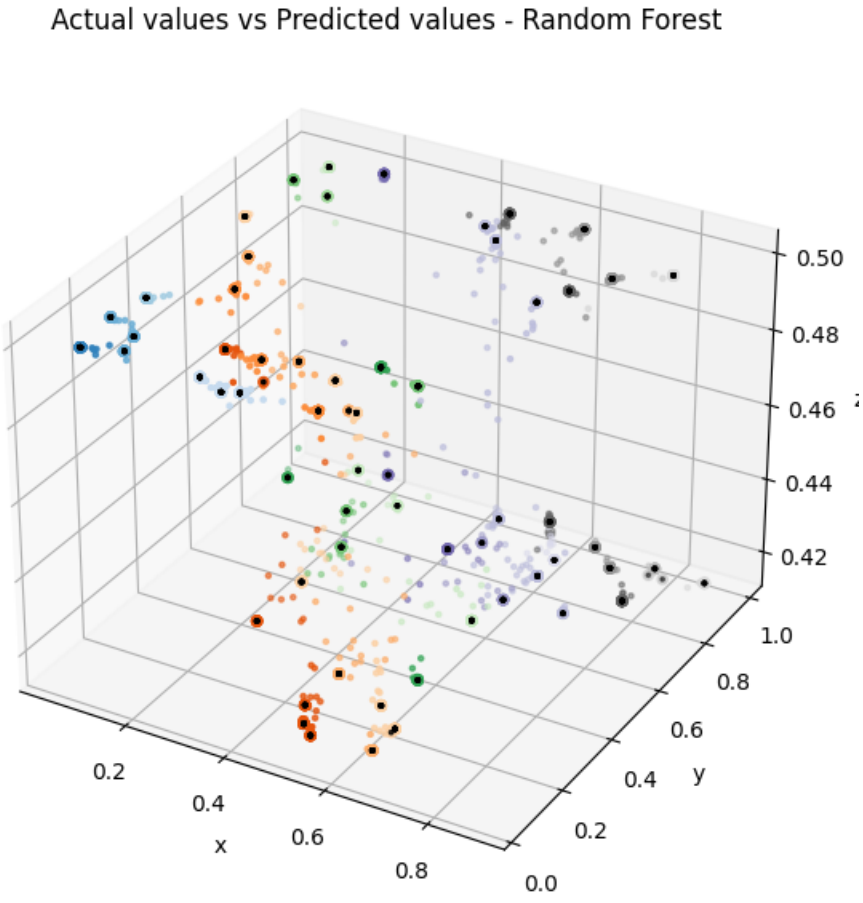

Figure 1: Actual values versus the predicted values of Random Forest plotted in 3D space.

The Support Vector Machine has the poorest performance with a R2 score of 61.83% and a MSE of 0.004 which is quite low compared to the other models. This result is mostly due to

the fact that the SVM was not able to distinguish between different floors shown in Figure 2 taken from the first run. The plot depicts the actual values and the predicted values in 3-dimensional space. The black markers are the actual values, and the coloured ones represent the predictions. Each colour represents an actual value's prediction.

As can be observed from the presented evidence, all the values that SVM predicted are on the same Z value which is in between the two floors. This is an immense disadvantage of using SVM for indoor localization. A possible reason for this happening is that the Z coordinate has a more complex relationship with the RSSI data and therefore when mapping the data to a higher dimension, the model is not able to find a linearly separable class for the Z coordinate values. One possible way around this problem is to create 2 SVM models instead of one. One of these models needs to be a multi-output model predicting the X and Y coordinate values and the other model needs to be trained and fine-tuned such that it predicts the Z coordinates values. Then the predicted X, Y and Z values can be concatenated so the full coordinate values can be presented.

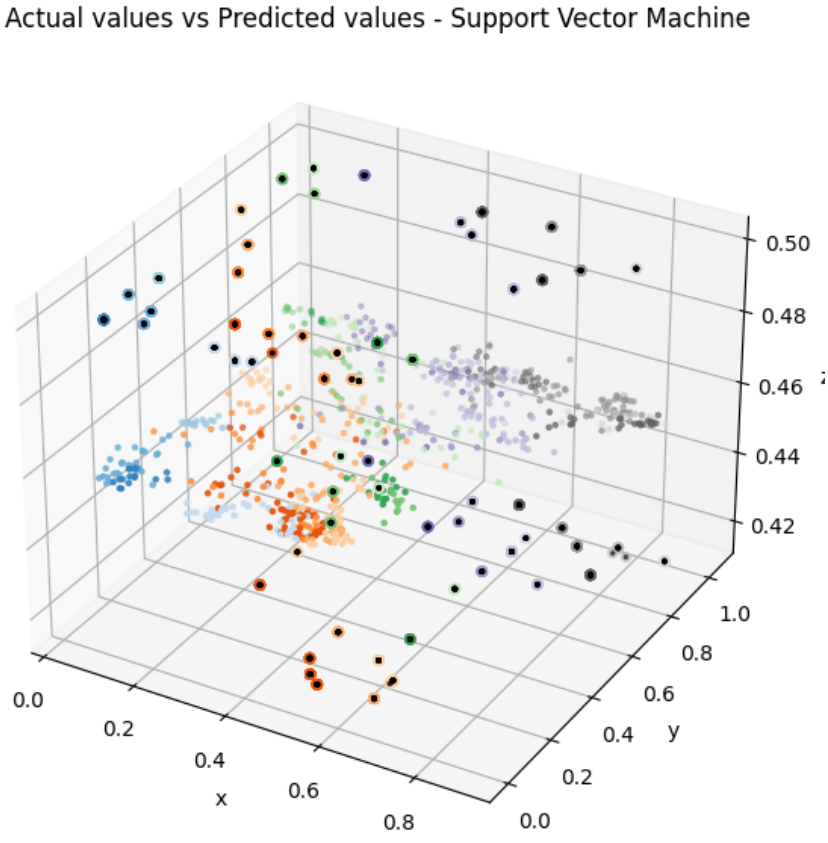

Figure 2: Actual values versus the predicted values of SVM plotted in 3D space.

4.2 Models with Orientation

To evaluate the performance of the models when the orientation of the device was also included as a feature in the dataset, the results of the models are reported in Table 2.

Model	MSE	R2
Random Forest	0.001 ± 0	97.16% ± 0
Decision Tree	0.006 ± 0.002	84.95% ± 2.86
Support Vector Machine	0.002 ± 0	95.29% ± 0
K-Nearest Neighbours	0.002 ± 0	94.64% ± 0
Deep Learning	0.002 ± 0.0001	93.82% ± 0.26

Table 2: Mean and standard deviation of MSE and R2 score of the models including the orientation of the device.

As indicated by Table 2, Random Forest has the best performance with the highest R2 score of 97.16% and the lowest MSE of 0.001. The second-best performing model for this approach is the Support Vector Machine with a R2 score of 95.29% and a MSE of 0.002. K-Nearest Neighbours still performs quite well, but not as well as the previously mentioned models. It has an R2 score of 94.64% and an MSE of 0.002. The deep learning model has slightly improved compared to the basic model, however, the machine learning models seem to outperform it in this approach as well. The worst-performing model in this case was the Decision Tree with a R2 score of 84.95% and a high standard deviation of 2.86 and a MSE of 0.006.

In contrast to the basic models, the approach including device orientation showed promising performance results across its models with Random Forest achieving an R2 score of 97.16% closely followed by SVM with a 95.29% R2 score. The SVM in the basic models had a 61.83% R2 score which is significantly lower than what was achieved by introducing device orientation. The Decision Tree of the orientation-based approach in contrast with the basic Decision Tree model had an almost 8% decrease in performance on the R2 score. Lastly, the Deep Learning model of the basic model had a 92.5% R2 score which in contrast to the device orientation approach was more than 1% less accurate.

Overall, this approach had great improvements on the SVM compared to the basic models while the Decision Tree had a great loss in performance.

4.3 Models with Access Points

In this section, the models with the coordinate values of the access points on the 5th and 6th floor of the building included are evaluated. This approach is still using a fingerprinting approach, however, the location values of the access points are present in each row of the dataset. This could allow the model to learn how far the device could be from each access point and provide more accurate estimates. To report the results the mean and standard deviation of the test MSE and R2 scores are compared over 5 runs in Table 3.

Model	MSE	R2
Random Forest	0.001 ± 0	97.26% ± 0.06
Decision Tree	0.0034 ± 0.001	91.57% ± 0.63
Support Vector Machine	0.002 ± 0	94.23% ± 0
K-Nearest Neighbours	0.002 ± 0	94.86% ± 0
Deep Learning	0.002 ± 0.0001	93.92% ± 0.16

Table 3: Mean and standard deviation of MSE and R2 score of the models including the access points of 2 floors.

Looking at Table 3, it is noticeable that Random Forest has the best performance once more with a R2 score of 97.26% and a MSE of 0.001. The second-best model of the approach including access points is the K-Nearest Neighbours with a 94.86% R2 score and a 0.002 MSE. The Support Vector Machine compared to KNN has almost the same performance differing only by a small amount on both metrics. The deep learning model ranks on the fourth place in this scenario as well with a 93.92% R2 score and a 0.002 MSE.

In comparison with the basic models, the Random Forest is the highest performing model in both approaches with the access point locations included being 97.26%, which is a slight decrease compared to the basic model, however, it is insignificant. The SVM of Table 3 shows a 94.23% R2 score that is significantly greater than the results of the SVM in Table 1 being 61.83%. The results of the Deep Learning model presented in Table 3 show a 93.92% R2 score while the deep learning model of Table 1 is slightly worse with a 92.5% R2 score. In conclusion, including the AP coordinates seem to show an overall improvement compared to the basic models.

4.4 Trajectory-based experiment

This section describes the results obtained from the experiment using the approach of collecting data while the device is in motion. The results that were achieved on the experiment dataset by using the basic models are shown in Table 4 where no post-processing was applied.

Model	MSE	R2
Random Forest	0.004 ± 0.0002	-4.63 ± 0.56
Decision Tree	0.006 ± 0.0001	-2.18 ± 0.06
Support Vector Machine	0.005 ± 0	-1.89 ± 0.01
K-Nearest Neighbours	0.008 ± 0	-6.78 ± 8.88
Deep Learning	0.004 ± 0.0003	-5.95± 4.05

Table 4: Mean and standard deviation of MSE and R2 score of the experiment.

Based on the R2 scores of this approach it is clear that the results are not as good as if we used measurements while staying stationary. A negative R2 score means that the results are worse than the mean of the target values. Due to this, it is better to evaluate the performance of the models by the MSE. The best model with this approach was the Random Forest model with an MSE of 0.004.

Figure 3: The test set and the predictions of the experiment using Random Forest on a map of the 6th floor.

The accuracy of the predictions can also be seen in Figure 3 where the predictions (in red) and the actual values (in grey) are shown. The predictions are relatively close to the actual values with this model, however, sometimes the predictions show that the device is in a different room and not the aisle.

The second-best model was the Deep Learning model with the same MSE of 0.004 but a slightly greater standard deviation closely followed by the Support Vector Machine with 0.005.The Decision Tree with an MSE of 0.006 and the K-Nearest Neighbours with an MSE of 0.008 performed quite badly with this approach. The results of these two models are visualized in Figure 4. The predictions have barely got close to the actual location values of the measurements. The KNN in particular had such a low MSE due to many predictions that are in the middle of the building.

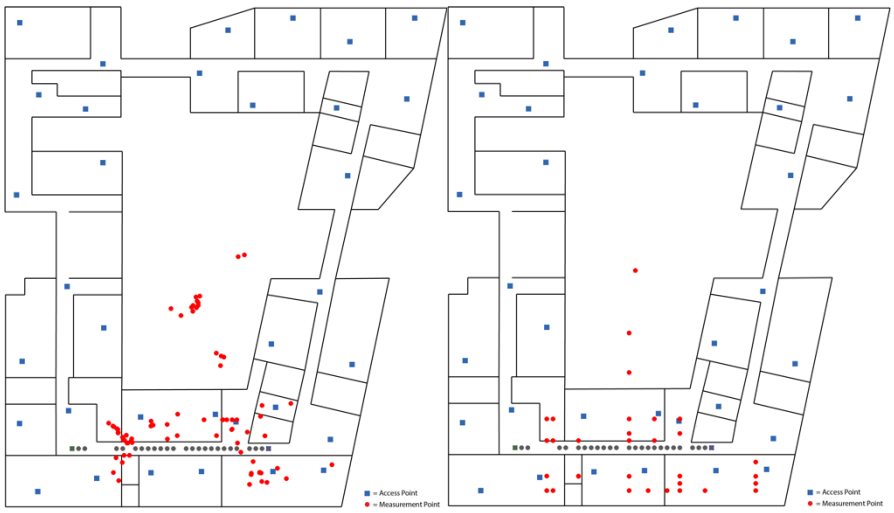

Figure 4: The test set and the predictions of the experiment using Decision Tree (left) and K-Nearest Neighbours (right) on a map of the 6th floor.

When the location estimates were projected to the walking route much different results were achieved. Using this approach is suitable since the Y and Z coordinates ended up having the same values as the original values of the dataset. The results are shown in Table 5. The best model was K-Nearest Neighbours with 88.55% R2 and a 0.002 MSE shown on Figure 5, followed by the Support Vector Machine having an R2 score of 84.2% and an MSE of 0.003. Random Forest performed slightly worse compared to the Support Vector Machine with an 82.97% R2 and a 0.004 MSE. The Decision Tree had a 76.87% R2 and a 0.005 MSE. All models improved greatly except for the Deep Learning model which even with post-processing

Model	MSE	R2
Random Forest	0.004 ± 0	82.97% ± 0.36
Decision Tree	0.005 ± 0.0002	76.87% ± 0.9
Support Vector Machine	0.003 ± 0	84.2% ± 0
K-Nearest Neighbours	0.002 ± 0	88.55% ± 0
Deep Learning	0.003 ± 0.0002	-2.26 ± 0

Table 5: Mean and standard deviation of MSE and R2 score of the experiment with post-processing.

had worse results than the mean of the target values, however, its MSE is 0.003 which is just slightly worse than of the K-Nearest Neighbours'.

All in all, applying post-processing to the results with this approach improved the performance significantly.

Figure 5: The test set and the predictions of the experiment using K-Nearest Neighbours on a map of the 6th floor.

5 Discussion

In this study, 4 machine learning models and a deep learning model with 3 different approaches got evaluated and compared to accurately determine the indoor localization of the device as well as an experiment with a trajectory-based approach. This section is dedicated to summarizing the key highlights of the study and interpretations as well as describing the main limitations and possible future improvements or experiments.

The first observation is that out of the 5 basic models Random Forest had the best performance with an R2 score of 97.34% and an MSE of 0.001 while SVM had the worst performance. It seems that SVM was not able to predict which floor the prediction should belong to and therefore placed all predictions in between the two floors on the same Z coordinate. The metrics of the basic models show that overall, the machine learning models performed much better than the deep learning model regardless of the poor performance of the SVM.

The approach including the orientation of the device seemed to have very different results compared to the basic models. In this approach, the Random Forest performed the best as well with an R2 score of 97.16% although this is a slightly worse performance than the basic models' Random Forest. Furthermore, the SVM had a substantial improvement compared to the basic models resulting in a 95.29% R2 score and a 0.0017 MSE. In this case, the SVM was able to predict values around the 5th and 6th floors. Additionally, the deep learning model had a more than 1% improvement on the R2 score and a significant improvement on the MSE as well.

The approach including the location of the access points on the 5th and 6th floor seemed to have the best overall result with all models performing above 91% on the R2 score. Similar to the other 2 approaches, Random Forest performed the best in this case as well with a 97.26% R2 score and a 0.001 MSE. The SVM had a greatly improved performance compared to the basis models in this approach too resulting in a 94.23% R2 score and a 0.002 MSE. Furthermore, the deep learning model had a slight improvement in this approach as well compared to the basic model's performance.

Overall, in all 3 approaches, Random Forest had the best results with the highest accuracy in predicting indoor location coordinates. SVM seemed unreliable without the additional features that were missing from the basic models. The deep learning model performed generally well in all approaches, however, using a Random Forest regressor seems to outperform the accuracy of it. KNN had almost the same results in all 3 approaches meaning

that the additional features did not significantly improve its performance. Lastly, the Decision Tree had somewhat acceptable results, however, other models outperformed it meaning that it is not the best one to use for this problem.

Comparing the results to the existing literature some differences can be found due to various factors. Firstly, the results achieved from the deep learning model used in this thesis seem to perform slightly worse than the literature. Other studies presented results with accuracies varying from 93% to 99%. This could be due to the literature having more fine-tuned models by performing an experimental approach for determining the hyperparameters or using autoencoders. The results obtained from the SVM seem to have a similar performance in the approach that includes the orientation of the devices in the dataset. It is to note, that results achieved in the literature are from a classification-based approach, therefore the training of the model was different to the one in this thesis. Other reasons for the differences between both the machine learning models and the deep learning models can be due to using different datasets. The dataset in some of the existing papers includes a significantly larger dataset than the one in this thesis. The method of training the models could also make a difference in the results since it is also very common to use trial and error way of training models.

All in all, adding the orientation of the device or the access points to the dataset, only had improvements on the performance of the deep learning model and the SVM, while the other models' performance got slightly worse or did not change at all.

5.1 Limitations

Even though this study showed various insights into the performance of the machine learning models and the deep learning model, there are some limitations that should be considered.

The results of this study are based on the dataset that was gathered and the models might have different results on a different dataset. This can also be influenced by the fact that smartphones have different antennas and Wi-Fi standards used therefore the dataset could have very different results after data collection.

Furthermore, training these models requires a significant amount of computational power that not all computers have, therefore using a more powerful computer that can train on more hyperparameters with a greater range of values could provide very different results. Having more time to train the models might have resulted in even better accuracy in some cases.

The dataset used in this study was collected at times of the day when there was little to noone in the university building. Therefore, when applied in a real-world environment, these models could have very different results in terms of accuracy.

5.2 Experiment

In this section, the findings and implications of the trajectory-based approach will be discussed. The experiment aimed to explore what results could be obtained – by using applying the data collected throughout this experiment – compared to the basic models used earlier.

The results that were acquired did not present the best outcomes. The predictions were worse than the mean of the target values, however, the MSE of the Random Forest and the Deep Learning model showed relatively good results compared to the other models. The fact that the predictions are not as accurate as the other approaches is explainable since the data was gathered in a fundamentally different way. Furthermore, since the device was moving while scanning, scan results could have been obtained from one or more meters away from where the location of the scan was recorded, causing a shift in location predictions. Additionally, the scan results in some scans had less RSSI than in the other static scanning approaches which made it harder for the models to predict the indoor positions.

After applying post-processing to the predictions and projecting the results to the walking path the results seemed to have improved significantly. Google Maps uses the same approach by accepting coordinate values and looking for the nearest road segment [18]. Applying such post-processing, however, is harder to make generalizable for indoor localization. Comparing this to the results obtained without post-processing it is interesting how different the performance of the models changed. Random Forest always seemed to perform much better than the other models, however, after post-processing K-Nearest Neighbours presented the best results. This might indicate that K-Nearest Neighbours is a more generalizable model for indoor positioning.

All in all, carrying out this experiment showed how different the results are when the device is in motion while scanning.

5.3 Future Work

The study's findings provided valuable information on how well machine learning and deep learning models can perform in predicting indoor positions, however, there are several ways to build upon this research.

One possible experiment could investigate how well such models can predict indoor location using RSSI data from multiple devices [20] and how generalizable the model could become.

Another possible improvement could be including information about the environment such as categorical values about the type of room or area the measurements were taken from. A similar experiment has been carried out [4] on BLE RSSI data that presented some promising results by having information about the environment.

Since the results of the experiment are not accurate enough to be used without post-processing, adjustments need to be made to improve the results. One of these could be that in the case of this project, the Y coordinates are adjusted to the straight line that was being followed.

Another way to improve the experiment's results is to make a distinction between reasonable and bad estimates because of missing or conflicting RSSI measurements. This way we could filter out the estimates that are not good enough.

Another option would be to build such a model that can predict location estimates when certain RSSI measurements are missing. This, however, could also involve collecting additional data to better adjust the model to different environmental conditions and make it more generalizable.

6 Conclusion

In the course of this study, we collected a vast amount of RSSI data and other additional information, evaluated and compared the performance of each machine learning and deep learning model in various approaches. The results showed us that using machine learning or deep learning has great potential and can be extremely advantageous to accurately predict the position of devices in indoor environments based on Wi-Fi RSSI data.

References

[1] Gülsüm Zeynep Gürkaş Aydın Ahmet Sertbaş Zeynep Turgut, Serpil Üstebay. 2018. Deep Learning in Indoor Localization Using WiFi. (2018).

[2] Mohd Nizam Husen and Sukhan Lee. 2014. Indoor human localization with orientation using WiFi fingerprinting. (2014). https://doi.org/10.1145/2557977.2557980

[3] Rik Timmer. 2021. RSSI-Based Indoor WiFi Localization Using Deep Learning. (2021).

[4] Yoga Yustiawan Hani Ramadhan and Joonho Kwon. 2020. Applying Movement Constraints to BLE RSSI-Based Indoor Positioning for Extracting Valid Semantic Trajectories. (2020).

[5] Getaneh Berie Tarekegn Abebe Belay Adege, Hsin-Piao Lin and Shiann-Shiun Jeng. 2018. Applying Deep Neural Network (DNN) for Robust Indoor Localization in Multi-Building Environment. (2018)

[6] Lee S. Huang K. Kim, K. 2018. A scalable deep neural network architecture for multi-building and multi-floor indoor localization based on Wi-Fi fingerprinting. (2018)

[7] Christian Gentner, Markus Ulmschneider, Isabel Kuehner, and Armin Dammann. 2020. WiFi-RTT Indoor Positioning. (2020), 1029–1035. https://doi.org/10.1109/PLANS46316.2020.9110232

[8] Li, J., Yue, X., Chen, J., & Deng, F. (2017). A Novel Robust Trilateration Method Applied to Ultra-Wide Bandwidth Location Systems. Sensors (Basel, Switzerland), 17(4), 795. https://doi.org/10.3390/s17040795

[9] Navneet Singh, Sangho Choe, and Rajiv Punmiya. 2021. Machine Learning Based Indoor Localization Using Wi-Fi RSSI Fingerprints: An Overview. IEEE Access 9 (2021), 127150–127174. https://doi.org/10.1109/ACCESS.2021.3111083

[10] Ahmed H. Salamah, Mohamed Tamazin, Maha A. Sharkas, and Mohamed Khedr. 2016. An enhanced WiFi indoor localization system based on machine learning. In 2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN). 1–8. https://doi.org/10.1109/IPIN.2016.7743586

[11] Ahasanun Nessa, Bhagawat Adhikari, Fatima Hussain, and Xavier N. Fernando. 2020. A Survey of Machine Learning for Indoor Positioning. IEEE Access 8 (2020), 214945–214965. https://doi.org/10.1109/ACCESS.2020.3039271

[12] Susmita Ray. 2019. A Quick Review of Machine Learning Algorithms. (2019).

[13] Jinah Kim Nammee Moon Sunmin Lee. 2019. Random forest and WiFi fingerprint-based indoor location recognition system using smart watch. (2019).

[14] Ann Transl Med. 2016. Introduction to machine learning: k-nearest neighbors. (2016)

[15] Aigerim Mussina and Sanzhar Aubakirov. 2019. Improving Indoor Positioning via Machine Learning. (2019)

[16] Máté Nagy (2023) wifi-indoor-navigation-rssi [Source code] https://github.com/n-mate11/wifi-indoor-navigation-rssi

[17] Máté Nagy (2023) wifi-model-rssi [Source code] https://github.com/n-mate11/wifi-rssi-model

[18] Google (2023) Google https://developers.google.com/maps/documentation/roads/overview

[19] Bellavista-Parent, V., Torres-Sospedra, J., & Pérez-Navarro, A. (2022). Comprehensive Analysis of Applied Machine Learning in Indoor Positioning Based on Wi-Fi: An Extended Systematic Review. Sensors (Basel, Switzerland), 22(12), 4622. https://doi.org/10.3390/s22124622

[20] Saideep Tiku, Prathmesh Kale, and Sudeep Pasricha. 2021. QuickLoc: Adaptive Deep-Learning for Fast Indoor Localization with Mobile Devices. ACM Trans. Cyber-Phys. Syst. 5, 4, Article 44 (sep 2021), 30 pages. https://doi.org/10.1145/3461342

Appendix

Hyperparameters and best settings of the basic machine learning models

Model	Hyperparameter	Best Setting
Random Forest	n_estimators	175
	max_depth	60
	min_samples_split	4
	bootstrap	false
	max_features	sqrt
	min_samples_leaf	1
	min_weight_fraction_leaf	0
	max_leaf_nodes	None
	min_impurity_decrease	0
	oob_score	False
	ccp_alpha	0
	max_samples	None
Decision Tree	criterion	poisson
	splitter	best
	max_depth	None
	min_samples_split	5
	min_samples_leaf	1
	min_weight_fraction_leaf	0
	max_features	None
	max_leaf_nodes	230
	min_impurity_decrease	0
	ccp_alpha	0
SVM	degree	1
	coef0	1e-06
	gamma	1
	tol	1e-05
	C	1
	kernel	rbf
	epsilon	0.1

	shrinking	true
KNN	algorithm	kd_tree
	leaf_size	10
	n_neighbors	7
	weights	distance
	metric	manhattan
	p	1
	metric_params	None