Bachelor's thesis

Machine Learning-based User Movement Prediction in Layer 2 Networks

by
Lina Wilske

Supervisors
Prof. Dr. Holger Karl
Leonard Paeleke
Internet Technology and Softwarization Group

Hasso Plattner Institute at University of Potsdam

September 11, 2023

Abstract

Mobile device roaming on Wireless Fidelity (Wi-Fi) networks currently does not consider human movement, leading to performance issues, e.g., in video conferencing [15]. This thesis presents a Machine Learning (ML) approach that uses a Long Short-Term Memory (LSTM) model to predict the following Access Point (AP) based on Received Signal Strength Indication (RSSI) values from the surrounding APs.[1] The model's inputs include time sequences of waypoint, acceleration, and Wi-Fi data such as Basic Service Set Identifiers (BSSIDs) with their RSSI values. The output of the model will be used to get three BSSIDs, named `3AP`, where one of them must be the following AP the station may connect to be a correct prediction. The model was trained using modified real-world data from a Microsoft Research competition and achieved a prediction accuracy of 76 %. With these predictions, the roaming process in Wi-Fi networks initiated by a mobile device could be improved. However, the performance was only marginally better than a heuristic method, which selects the BSSIDs with the highest RSSI from the latest measured RSSI values. Further improvements could be achieved with fewer classes, a different-sized sliding window, and additional sensor data. Future research should focus on real-world data with these improvements, so that human movement can precisely predict the following AP.

Zusammenfassung

Das Roaming mobiler Geräte in Wi-Fi-Netzwerken berücksichtigt derzeit nicht die Bewegungen des Menschen, was zu Leistungsproblemen führt, z. B. bei Videokonferenzen [15]. In dieser Arbeit wird ein ML-Ansatz vorgestellt, der ein LSTM-Modell zur Vorhersage des folgenden AP auf der Grundlage von RSSI-Werten aus dem umgebenden APs. Die Eingaben des Modells umfassen Zeitsequenzen von Wegpunkt-, Beschleunigungs- und Wi-Fi-Daten wie BSSIDs mit ihren RSSI-Werten. Die Ausgabe des Modells wird verwendet, um drei BSSIDs, genannt `3AP`, zu erhalten, von denen einer der folgende AP sein muss, mit dem sich die Station verbinden kann, um eine korrekte Vorhersage zu erhalten. Das Modell wurde mit modifizierten realen Daten aus einem Microsoft Research-Wettbewerb trainiert und erreichte eine Vorhersagegenauigkeit von 76 %. Mit diesen Vorhersagen konnte der Roaming-Prozess in Wi-Fi-Netzen, die von einem mobilen Gerät initiiert werden, verbessert werden. Die Leistung war jedoch nur geringfügig besser als eine heuristische Methode, die aus den zuletzt gemessenen RSSI-Werten die BSSIDs mit dem höchsten RSSI auswählt. Weitere Verbesserungen könnten mit weniger Klassen, einer anderen Größe des Schiebefensters und zusätzlichen Sensordaten erzielt werden. Zukünftige Forschungen sollten sich auf reale Daten mit diesen Verbesserungen konzentrieren, damit die menschliche Bewegung die folgenden AP genau vorhersagen kann.

[1] Implementation of this thesis: `https://github.com/linaScience/ba-implementation`

Contents

1 Introduction

In large-scale Wireless Fidelity (Wi-Fi) environments such as office buildings, shopping malls, and airports, where multiple Access Points (APs) are required, people often move around indoors with their mobile devices. To maintain a stable connection to the Wi-Fi, the station must remain in the range of an AP or may roam to another with the same Service Set Identifier (SSID). However, the current roaming process, as defined in the 802.11k/r[17][16] Wi-Fi standard, does not consider human movement factors such as acceleration and trajectory. For example, consider that a user's station moves away from AP1 towards AP2 and further towards AP3, as seen in Figure 1.1. Then, there is no initiation of a roam from AP1 to AP3, but instead the station will first roam from AP1 to AP2 and then to AP3, which increases the number of roamings leading to interruptions in, e.g, video confereces [15].

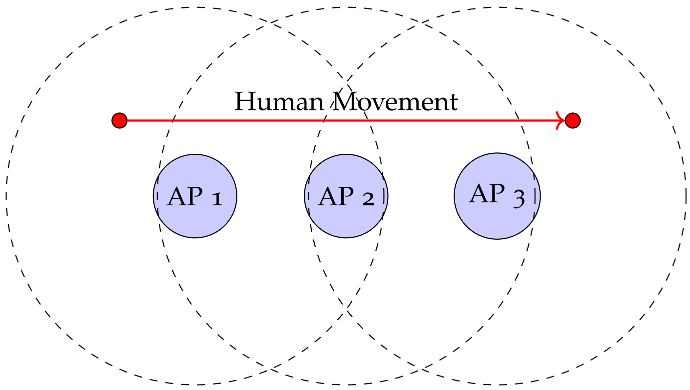

Figure 1.1: Roaming process of a user's station from AP1 to AP3 bypassing also AP2.

Real-time applications such as video conferencing are susceptible to these frequent roamings, which may result in dropped connections and unsatisfied users. Reducing the number of roams will reduce the probability of connection losses. Instead, it would be ideal if the movement from AP1 to AP3 were detected by the station beforehand and a roam from AP1 to AP3 was initiated.

Therefore, this thesis explores if a time series Machine Learning (ML) model can predict the following AP a station may connect to next. Because of the many APs in large-scale Wi-Fi environments, I interpret this prediction as a multi-class classification problem where each Basic Service Set Identifier (BSSID) will be a class. Typical applications for multi-class classification are image recognition of, e.g., animals or handwriting recognition with a limited number of classes [12]. This thesis has nearly 4800 BSSIDs, making the prediction task much more challenging. Hence, the model predicts a set of three APs, defined as `3AP` in the following, that the station may connect to next, which must include the most likely AP to be claimed as a correct prediction. This prediction will be compared to a heuristic approach, which will choose the following AP based on the latest measured Received Signal Strength Indication (RSSI) value. If the RSSI is the highest, it is considered the next one to connect to.

A time-series ML model requires time series as input. There are two possible data sources: generate new or utilize existing data. Creating a large-scale environment with many APs and users is not feasible for this thesis, as this process is time-consuming and needs a lot of planning and evaluation beforehand. Thus, this thesis will use an existing dataset with sensor data such as acceleration, waypoint, and Wi-Fi data from large-scale environments. The only dataset I found meeting these requirements is from a 2021 competition by Microsoft Research [18], located on kaggle[2]

The prediction will be difficult due to the large number of classes and because multi-class classification, such as image recognition, deals with a limited number of classes. The more possible classes to be predicted, the more likely a prediction may be wrong. The relaxation to a `3AP` will make the prediction task easier. Still, it will be challenging to outperform the heuristic approach, as it is simple because it relies on the last measured RSSI value, which may not change that much in a short time. I expect the ML model accuracy to be better than the heuristic approach because we can utilize the user's trajectory, which the heuristic approach cannot.

The rest of this thesis is structured as follows: In Chapter 2, I will give an overview of the terms of ML concepts and models used in this thesis. Chapter 3 will discuss related work and compares it to this thesis. The data will be analyzed in Chapter 4 to determine what parts of the data I will use for the ML model. After that, I will discuss the suitability of some pre-selected time series ML models for the task in Chapter 5. Because of findings in Chapter 4, this thesis needs to preprocess the prepared data further and will implement the Long Short-Term Memory (LSTM) model for one site and floor of the competition in Chapter 6. In Chapter 7, I will evaluate the model's performance and conclude if this prediction could be useful in the future in Chapter 8.

[2]Kaggle, a website containing competitions and datasets for machine learning: `https://www.kaggle.com`

2 Background

The basic information in this chapter helps to comprehend the key ideas covered in this thesis. The field of ML is extensive, with many models created for diverse tasks, each with advantages and uses. This chapter will provide a brief overview of the models discussed and used in this thesis, as well as the concepts of classification, multivariate time series, time series prediction, and hyperparameter tuning.

2.1 Classification

Classification models in ML are designed to categorize input data into specific classes using input features and labels. In supervised learning, models are trained with input vectors and their associated target vectors, which represent the desired output or correct category for each input. Classification problems, such as digit recognition, assign input vectors to discrete categories. Practical applications include determining whether an email is spam or a fraudulent transaction [21]. Multi-class classification differentiates among more than two classes [2]. In this thesis, a multi-class classification model will be used to predict which AP a user will be closest to, based on the RSSI from the APs and the user's trajectory.

2.2 Univariate and Multivariate Time Series

A time series is univariate if one observation is recorded sequentially over time, e.g., temperature or stock prices. If another observation was recorded over time together with, e.g., the temperature, such as humidity, then the time series is multivariate [5].

2.3 Time Series Prediction

Time series prediction involves training models on sequences of observations to predict the next value in the sequence [5]. These sequences, consisting of chronologically arranged data points, are prevalent in numerous domains. Due to the inherent temporal dependencies in time series data, where subsequent data points influence previous ones, specific ML techniques are applied. These techniques aim to capture and leverage temporal patterns within the data, predicting future trends based on historical observations [24].

2.4 Machine Learning Models for Time Series Prediction

Various ML models have been developed to handle sequential data. Some of these models derive from established statistical techniques, while others emerge from recent developments. This section overviews a few pre-chosen models relevant to this thesis.

2.4.1 Hidden Markov Model

Hidden Markov Model (HMM) is a statistical model that assumes the system being modeled is a Markov process with unobserved states [30]. HMMs are mainly known for their application in temporal pattern recognition, such as speech and handwriting. They describe the probability of a sequence of observable data, which is assumed to result from a sequence of hidden states, each producing an observable output according to a particular probability distribution.

2.4.2 Multilayer Perceptron

Multilayer Perceptron (MLP), also known as a feedforward artificial neural network, is a class of deep learning models primarily used for supervised learning tasks [31]. An MLP consists of multiple layers of nodes in a directed graph, each fully connected to the next one. Each node in one layer is connected with certain weights to every node in the following layer. MLPs apply a series of transformations, typically nonlinear, to the input data using activation functions, such as the sigmoid or Rectified Linear Unit (ReLU), facilitating the model's ability to model complex patterns and dependencies in the data [10].

2.4.3 Recurrent Neural Networks

Recurrent Neural Network (RNN) is a neural network well-suited to sequential data because of its design [14]. Unlike to traditional feedforward neural networks, a RNN possesses loops in its topology, allowing information to persist over time. This unique characteristic enables the model to use its "memory" to process sequences of inputs, making it ideally suited for tasks involving sequential data such as speech recognition, language modeling, and time series prediction [8].

2.4.3.1 Long Short-Term Memory

LSTM is a special kind of RNN, capable of learning long-term dependencies [13]. LSTMs were designed to combat the "vanishing gradient" problem in traditional RNNs. This problem made it difficult for other neural networks to learn from data where relevant events occurred with significant gaps between them. The key to the ability of the LSTMs is its cell state and the input, forget, and output gates, which regulate the flow of information in the network.

2.5 Hyperparameter tuning

In ML, hyperparameters play an important role in model development as they may improve the model's performance [34]. These are parameters such as the learning rate, neural network layers, and the sliding window or batch sizes. Proper selection of hyperparameters, known as hyperparameter tuning or optimization, is crucial to optimize model performance. This iterative procedure involves exploring various hyperparameter combinations for the configuration that yields the most accurate predictions. Hyperparameters can be tuned by, e.g., random search, which can be done manually or using libraries. This thesis will use keras-tuner[3] to tune the hyperparameters of the LSTM model. `RandomSearch` is a hyperparameter optimization algorithm that randomly searches the hyperparameter space for the best configuration. The user predefines the hyperparameter space, and the algorithm tries out different hyperparameters in this space.

[3] Keras-tuner, the hyperparameter optimization framework for keras: `https://github.com/keras-team/keras-tuner`

3 Related Work

Numerous studies have focused on various topics in the fast-developing field of wireless communications and networking, from handover prediction to user mobility and network traffic prediction. This chapter gives a summary of the important research and their relation to this thesis.

3.1 Handover Prediction

Montavont et al. [26] propose a handover decision algorithm based on the Global Positioning System (GPS) location of the mobile device. Unfortunately, in large-scale and dense Wi-Fi environments, GPS may not be available or not accurate enough for indoor trajectories. Khan et al. [23] address the problem of handover prediction and AP selection in dense Wi-Fi networks with Software Defined Networking (SDN). Their AP selection predictions outperform the current approaches of strongest received signal first by 9.2 % and least-loaded AP first by 8 %. Khan et al. focus on using ML for throughput estimation of the network and accordingly choose the best AP to roam to; they do not consider the trajectory of the mobile device in the AP selection or use ML for the AP selection process directly.

3.2 User Movement Prediction

Bakirtzis et al. [1] treat their indoor-outdoor detection problem as a multivariate time-series classification. They use ML containing LSTM for their prediction and demonstrate that a multivariate time-series classification approach can be used to monitor a user's environment. To predict user movement, Bourjandi et al. [4] use a mix of multiple ML models, where LSTM is used to learn long-term dependencies. Prasad et al. [29] propose a HMM to predict the next possible location. They use real-world data, which contain times, direction, and movement speed.

3.3 Network Prediction

There are also approaches to predict network traffic. To forecast network traffic, Ferreira et al. [9] compare different ML models such as RNN and LSTM and perform experiments with real-world data. Mirza et al. [25] predict Transmission Control Protocol (TCP) throughput for arbitrary network paths in the Internet with the ML model Support Vector Machines.

3.4 Summary

Various research contributions have been made in wireless communications and networking, focusing on challenges ranging from handover prediction to user mobility and network traffic prediction. While the abovementioned studies have focused more on sparse Wi-Fi environments, this thesis focuses on dense Wi-Fi environments. To the best of my knowledge, a combination of user movement and handover predictions for large Wi-Fi environments has not been explored yet, which is addressed in this thesis.

4 Dataset analysis and preparation

As mentioned in Chapter 1, the dataset used in this thesis is the Indoor Location & Navigation from kaggle, which was part of a competition of Microsoft Research in 2021 [18]. The company "XYZ^{10}"[4] recorded the data in shopping malls and was provided by Microsoft Research for this competition. The competition's goal was to predict the indoor position of users' smartphones based on real-time sensor data and user trace data. The prediction for this competition contains the floor and waypoint at a particular timestamp. However, this thesis will not predict the floor and waypoint but the following BSSID a device may connect to based on the RSSI of the APs and the trajectory of the user, as this prediction is more beneficial for the roaming process. Therefore, the dataset will be analyzed to determine what parts of the data I use for the ML model.

4.1 Components of the dataset

As noted in the kaggle notebook "Indoor Navigation: Complete Data Understanding" [19] the data consists of 3 parts:

- a `train` folder with train path files, organized by site and floor.
- a `test` folder with test path files, organized by site and floor but without waypoint data.
- a `metadata` folder with floor metadata, organized by site and floor, which includes floor images, further information, and a geojson map.

The train folder contains 204 subfolders representing each shopping mall (site) where data was recorded. In each folder of the 204 subfolders, there are at least one and at most twelve subfolders representing the site's floors; the median is five floors. Overall, there are 26,925 files, each containing the movement of a person for a specific site and floor. Per floor, there are between one and 284 files with a median of 14. The floor F1 of the site 银泰城(城西店) (Yintai City (Chengxi Branch)) in the train folder of the competition, has the most files.

The submission files and the test folder will not be used for this thesis. Instead, I will generate the test set out of the train data because the goal is not to predict the floor and site name for a specific timestamp but to predict the BSSID to which a device may connect next, which is an entirely different task. Therefore, I will not analyze the content of the test and metadata folders in detail but will further focus on the content of the train folder.

[4]Website of XYZ^{10}: `https://dangwu.io`

4.2 File structure

Each file in each floor folder is a **.txt** file. The first contains the start time of the recording, the second site information `SiteID` as hash, `SiteName`, `FloorId` as hash, and `FloorName`.

Listing 4.1: A snippet from the dataset of the file 5daa9e38df065a00069beb79.txt of the floor F4

```
#   startTime:1571462193934
#   SiteID:5d27099303f801723c32364d SiteName:银泰百货(庆春
    店) FloorId:5d27099303f801723c323650 FloorName:4F
1571462193944   TYPE_WAYPOINT   57.885998   69.501526
1571462194071   TYPE_ACCELEROMETER  -0.95254517 0.7944031   8.928757    2
1571462194071   TYPE_MAGNETIC_FIELD -25.65918   -4.4784546  -28.201294  3
1571462194071   TYPE_GYROSCOPE  -0.22373962 -0.07733154 -0.16847229 3
1571462194071   TYPE_ROTATION_VECTOR    0.04186145  -0.02101801 -0.72491926 3
1571462194071   TYPE_MAGNETIC_FIELD_UNCALIBRATED    -4.8568726  10.406494   -387.44965
    20.802307   14.884949   -359.24835  3
1571462194071   TYPE_GYROSCOPE_UNCALIBRATED -0.22218323 -0.068359375    -0.1628418
    0.0026245117    9.765625E-4 -7.6293945E-4   3
1571462194071   TYPE_ACCELEROMETER_UNCALIBRATED -0.95254517 0.7944031   8.928757    0.0 0.0 0.0
    3
...
1571462194883   TYPE_WIFI   b06c4e327882fab58dfa93ea85ca373a54e887b5
    9f967858afcbb907af6e5adef766c7e7b936ef07    -63 2462    1571462190744
1571462194883   TYPE_WIFI   8204870beb9d02995dab3f08aad97af5eab723cc
    0413b35df78fc865af15b4721d5aeb33ff57da45    -64 2447    1571462188686
...
1571462194020   TYPE_BEACON 07efd69e3167537492f0ead89fb2779633b04949
    b6589fc6ab0dc82cf12099d1c2d40ab994e8410c    76e907e391ad1856762f70538b0fd13111ba68cd
    -57 -71 5.002991815535578   1b7e1594febd760b00f1a7984e470867616cee4e    1571462194020
...
1571462195943   TYPE_WAYPOINT   59.72475    69.02152
#   endTime:1571462195976
```

The last line contains the end time of the recording. The central part of the data consists of the collected data. Each line contains a UNIX timestamp in milliseconds, followed by a data type and the data itself, all separated by a tabulator. The GitHub repository of the competition[5] shows that the data type in the second column followed by its data can be one of the following:

T1 `TYPE_ACCELEROMETER` with x, y and z acceleration and an accuracy value.

T2 `TYPE_MAGNETIC_FIELD` with x, y and z magnetic field and an accuracy value.

T3 `TYPE_GYROSCOPE` with x, y and z gyroscope and an accuracy value.

T4 `TYPE_ROTATION_VECTOR` with x, y and z rotation vector and an accuracy value.

T5 `TYPE_MAGNETIC_FIELD_UNCALIBRATED` with x, y and z magnetic field and an accuracy value.

T6 `TYPE_GYROSCOPE_UNCALIBRATED` with x, y and z gyroscope and an accuracy value.

T7 `TYPE_ACCELEROMETER_UNCALIBRATED` with x, y and z acceleration and an accuracy value.

[5]The repository for the Indoor Location Competition 2.0: `https://github.com/location-competition/indoor-location-competition-20`

T8 TYPE_WIFI with SSID, BSSID, RSSI, frequency, and last seen timestamp of the access point. The SSID and BSSID are hashed.

T9 TYPE_BEACON with Universally Unique Identifier (UUID), Major Identifier (MajorID), Minor Identifier (MinorID), Transmission Power (TxPower), RSSI, distance to the device measured by the beacon, Media Access Control (MAC) address and a timestamp as padding data. The MajorID and MinorID are hashed.

T10 TYPE_WAYPOINT with x and y coordinates, the ground truth locations labeled by the surveyor.

Each file contains a different amount of waypoints and sensor data. Each file's first and last data type is a TYPE_WAYPOINT. Lines with types from T1 to T7 occur every 20 ms and are measured at the same time. TYPE_WIFI occurs about every 1800-2200 ms. TYPE_WAYPOINT data is not evenly distributed. Presumably, the waypoint data recording is triggered by an exterior event, e.g., a button press. As seen in Listing 4.1, the data are measured separately from each other, so there are no combinations of the data types. Most importantly, there is no combination of TYPE_WAYPOINT and TYPE_WIFI data, which would be needed for the prediction. The location provided by TYPE_WAYPOINT may be necessary because not only the strength of the RSSIs is important but also the trajectory of the user, which can be identified by the TYPE_WAYPOINT data. Furthermore, the RSSI values may be influenced by the environment, e.g., walls, which may result in a lower RSSI value.

A prediction of the following BSSID will only work per site due to the different APs. Still, the prediction could be difficult for a whole site because the APs are different on each floor, which may result in many APs for the prediction. Because the mall's first floor in Yintai City (Chengxi Branch) has the most trajectory data, this thesis will concentrate on that site for better prediction. To further know how much data there is for the model's input, Table 4.1 shows a more detailed analysis.

Table 4.1: Overview of data for F1 of site Yintai City (Chengxi Branch)

Information	Value
Total data points	7,157,081
Average data points per file	25,201
Number of waypoints	2,027
Lines of each T1 to T7 data	746,689
Lines of Wi-Fi data	1,862,044
Lines of beacon data	66,187
Number of BSSIDs	4,795
Number of APs	4,795
Number of SSIDs	1,421
RSSI range	-93 to -13 dBm

4.3 Prepare data for an ML model

As seen in previous sections, a location for the time of `TYPE_WIFI` data points is not provided, but in order to predict the following BSSID more precisely than only using the RSSI from the `TYPE_WIFI`, a location is needed. Furthermore, there is also unnecessary data for the prediction, such as `TYPE_BEACON` data. Using the dataset for a ML training requires further data preparation. Details are given in the following. As seen in Table 4.1, this floor has 2,027 waypoints and 1,862,044 lines of Wi-Fi data. Multiple lines per timestamp exist because the devices gather data from all nearby APs for each timestamp. Figure 4.1 shows the waypoints.

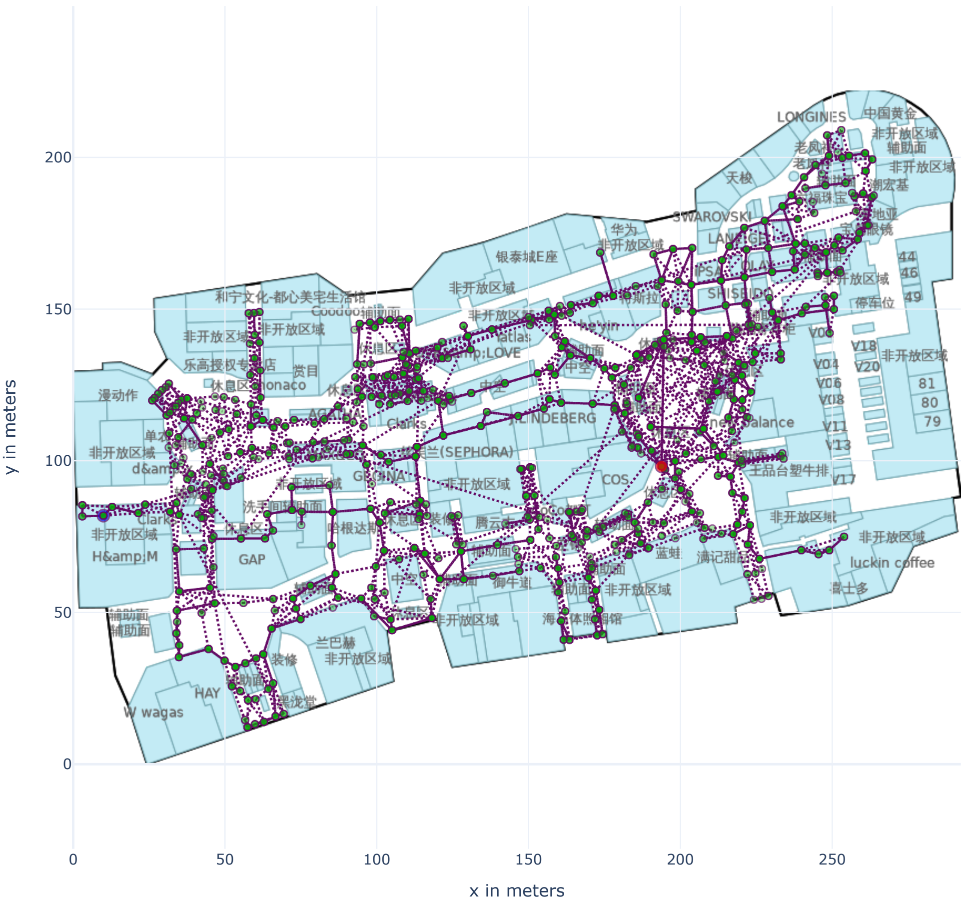

Figure 4.1: Visualization of the 2,027 waypoints of the site Yintai City (Chengxi Branch) on floor F1

Further human movement between `TYPE_WAYPOINT` and `TYPE_WIFI` data points may have occurred. A concatenation of the data points to directly get the user's location for the Wi-Fi data points will not work because the waypoint may have changed in that time or the RSSI value may have changed for the location. Therefore, I choose to interpolate the data points of the `TYPE_WAYPOINT` data for each `TYPE_WIFI` timestamp.

Therefore, I perform an interpolation of TYPE_WAYPOINT data for TYPE_WIFI timestamps in order to get a location for the Wi-Fi. With this interpolation, a combination of TYPE_WAYPOINT and TYPE_WIFI data can be done, and more data could be used for the prediction. The interpolation results in 6549 waypoints, three times more than the original waypoints, as seen in Figure 4.2.

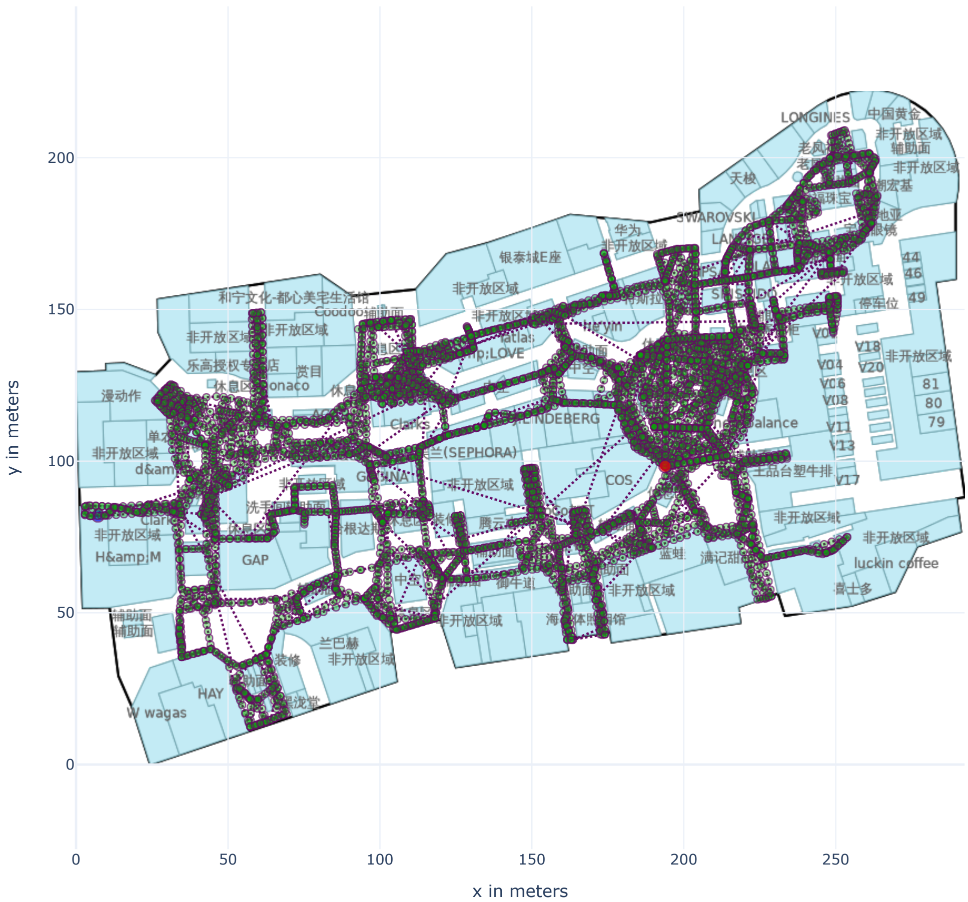

Figure 4.2: Visualization of the interpolated waypoints for site Yintai City (Chengxi Branch) on floor F1

As the TYPE_WAYPOINT data may have changed, the TYPE_ACCELEROMETER data may have changed as well. Although the TYPE_ACCELEROMETER data is measured every 20 ms, the TYPE_WIFI data may not completely match the TYPE_ACCELEROMETER timestamps. Therefore, an interpolation of TYPE_ACCELEROMETER data for TYPE_WIFI timestamps is done. The acceleration values may not change significantly, but it may result in a more accurate prediction than without interpolation. A multivariate time series with TYPE_WAYPOINT and TYPE_ACCELEROMETER data for each TYPE_WIFI timestamp is generated. This time series will be used for the ML model. Further interpolation of the data, such as TYPE_GYROSCOPE, is possible, but this thesis only utilizes the abovementioned data.

4.3.1 Peculiarities of the data

The dataset analysis revealed some peculiarities, which are described in the following.

Different devices collect the data at different timestamps and days. A problem for the time series is that the waypoint data were measured irregularly. As Figure 4.1 shows, some waypoints seem to be very distant from the next one, which can be detected by the dotted lines across the floor. Table 4.2 shows the top 10 pairs of waypoints with the most significant metric differences, where 174.77 meters is the most significant difference.

Table 4.2: Top 10 pairs with the most significant metric differences of data from floor F1 of site Yintai City (Chengxi Branch)

Rank	Point 1 (X)	Point 1 (Y)	Point 2 (X)	Point 2 (Y)	Metric Difference
1	247.965	168.763	117.924	51.998	174.771
2	98.663	127.597	258.750	181.234	168.833
3	189.587	71.455	89.734	102.255	104.495
4	223.493	145.094	174.263	78.863	82.523
5	34.865	35.456	33.284	110.761	75.322
6	50.311	92.031	114.972	123.045	71.715
7	150.920	145.152	222.234	146.110	71.321
8	64.641	25.345	34.475	84.446	66.355
9	56.838	74.521	94.438	125.973	63.726
10	172.344	56.201	212.448	99.340	58.901

This difference is too high for a human to walk in 1.8 to 2.2 seconds, the time between two waypoints. A human's gait speed is maximum at about 2.53 meters per second [3]. So, in 2.2 seconds, a human can walk 5.57 meters, much less than each of the values in the top 10 in Table 4.2. Therefore, waypoints that are more than 5.57 meters apart are defined as "too apart from each other" to walk in this time, and therefore, a split in the path will be done, and separated files will be generated, indicating a new path. This results in 147 files with data interpolation, which will be used for the ML model. However, the data in those interpolated files does not contain any information about the BSSID and corresponding RSSI values, which are needed for the prediction. This will be solved in the following Section 4.3.2.

4.3.2 Wi-Fi data for each timestamp

It is evident that for a waypoint, not all RSSI values for all APs are present because the AP may be out of range. In order to use the RSSI in the prediction and treat each BSSID as a class for the ML model, an RSSI value for each AP for each timestamp is needed.

For this, all BSSIDs of the site will be gathered by iterating over all Wi-Fi data and saved in one file. Every line contains the timestamp, and each column header is the BSSID of the AP, and each value of the line is the RSSI value of the BSSID at the timestamp. If an AP is absent, a very low value -999 is inserted because the typical RSSI value ranges from -55 to -90 [7]. This ensures that it is highly improbable for an AP to be selected for the prediction with this value. Then, I iterate over each file with interpolated data. For each timestamp, the BSSID and the corresponding RSSI value from the Wi-Fi file are added to the interpolated file. At each `TYPE_WIFI` timestamp with waypoint and acceleration data, the RSSI value for each BSSID is now saved.

4.4 Summary

The previous sections analyze the datasets and file structure. The folder with the most user movement will be used for the prediction task. The data was prepared into multivariate time series data for a ML model based on Wi-Fi, location, and acceleration data. An interpolation of the acceleration and waypoint data for each `TYPE_WIFI` timestamp was necessary to combine this information for the prediction task. The implementation of the preparation can be found in the GitHub repository [33] in the file "preparation.ipynb". This prepared data will be further preprocessed in Chapter 6 and utilized in the chosen ML model, which will be discussed in the next Chapter.

5 Suitable Machine Learning Model

The floor I analyzed in Chapter 4 has 4795 BSSIDs. Building upon the previous insights, the floor one Yintai City (Chengxi Branch) data has 4795 BSSIDs. I will interpret each BSSID as a class, translating to a high-dimensional classification problem with 4795 classes. This classification task necessitates a ML model that can capture the underlying patterns in the data and generalize well to unseen instances. I will discuss the classification models of Chapter 2 by the following topics: Temporal Dependency Handling, Capacity and Complexity, Multivariate Data, Flexibility and Integration, and Regularization and Overfitting.

Temporal Dependency Handling As mentioned in Section 4.3.1, the irregularities in waypoint data further emphasize the need for a model that can handle such temporal structures. While MLPs lack this capability [28], HMMs do offer some temporal structure but often fall short for longer sequences and multivariate data [30]. Traditional RNNs are better equipped but have limitations, such as the vanishing gradient problem [27]. LSTMs, in contrast, is specifically designed to handle long-term temporal dependencies, making them well-suited for our dataset with interpolated waypoints and Wi-Fi data.

Capacity and Complexity Given the 4795 classes, the model needs to have a considerable capacity. MLPs can scale their capacity by adding more hidden layers and units. They are capable of modeling complex relationships within data through their nonlinear activations to differentiate between the subtle differences in patterns that might exist among them. HMMs have limitations in handling the complexity of multi-class and multivariate problems due to their inherent Markovian assumptions and discrete state representations. They may struggle with such a high-dimensional problem. Traditional RNNs suffer from the vanishing gradient problem, especially in longer sequences, which limits their ability to capture long-term dependencies effectively [27]. LSTMs, being deep learning models, can scale effectively in terms of capacity by adding more layers or units while still maintaining their ability to handle temporal data.

Multivariate Data The data at hand is multivariate, with features such as waypoint, accelerometer, and Wi-Fi data. While MLPs can handle multivariate data, they treat each feature and time step independently, often missing the interdependencies. HMMs are primarily designed for univariate data. Extending them to multivariate scenarios requires additional complexities and assumptions. RNNs and LSTMs can seamlessly handle multivariate time series data. Their recurrent nature allows them to process each time step with multiple features effectively.

Flexibility and Integration Considering the interpolated data and the various preprocessing steps undertaken, as mentioned in Section 4.3, a flexible model that can be integrated with other architectures or preprocessing steps is desirable. MLPs are very flexible and can generally be used to learn a mapping from inputs to outputs [6]. This may be a good fit as I want the ML to learn which BSSID is the next one. HMMs are primarily designed for capturing state transitions in sequential data and may not be suitable for tasks requiring integrating spatial and temporal information. Their rigid assumptions about state transitions limit their flexibility in capturing complex patterns [30]. Traditional RNNs can capture short-term dependencies and are relatively more uncomplicated to integrate with other architectures due to their sequential nature. Also, LSTMs can easily be integrated to capture temporal and spatial features. This flexibility of RNNs and LSTMs is advantageous when dealing with complex and varied data sources.

Regularization and Overfitting The potential for overfitting is high given the large number of data points, as detailed in Table 4.1, and the intricate relationships between them. Therefore, dropouts may be used to prevent overfitting for each model [32]. While RNNs might be more prone to overfitting [27], MLPs, HMMs, and LSTMs offer better regularization capabilities.

Table 5.1: Suitability of models for various requirements.

	MLP	HMM	RNN	LSTM
Temporal Dependencies		✓	✓	✓
Capacity	✓			✓
Multivariate Data			✓	✓
Flexibility	✓		✓	✓
Overfitting	✓	✓		✓

To summarize, while MLPs, HMMs, and traditional RNNs have their strengths and have been successful in many applications, they have problems with multivariate time series classification with many classes. The classification problem for this thesis demands a model that can efficiently capture temporal patterns, scale in capacity, and handle multivariate data. As Table 5.1 shows, LSTMs can deal with this challenge due to their unique architecture and properties, making them the preferred choice for this task and the selected model for our implementation.

6 Implementation

All the code for this implementation can be found in the GitHub repository [33]. The implementation uses a standard LSTM architecture with prepared real-world data from Chapter 4. The code is structured into preprocessing and LSTM implementation, containing tuning, training, and testing, and will be described in the following sections.

6.1 Preprocessing

As described in Chapter 4, the data was prepared and has the following structure: The first column contains the Wi-Fi timestamps in milliseconds, the second and third columns contain the waypoint x and y coordinates in meters, the fourth to sixth columns contain the acceleration values x, y, and z in meters per second squared, as each acceleration value may influence the RSSI. The rest of the columns contain the RSSI data for each BSSID in the dataset. This data is preprocessed for the model as follows.

First, I set a `window_size` for the sliding window I want to use for the model, as LSTM needs sequence data. According to Jaén-Vargas et al. [20], the sliding window size for acceleration-based activity recognition should be $25 * 0.25 = 6.25$ seconds, so I choose 3 as the sliding window size, as the dataset has Wi-Fi timestamps for about every 2 seconds. If a file has less than or equal to 3 lines, it will not be used because I cannot apply a sliding window for this data. Furthermore, the length of each file will be saved to know where I need to split the sliding window in the preprocessing later. Then, I create the target variable, which is a variable where the BSSID with the highest RSSI is saved for each timestamp, which results in a list with 4795 entries, as there are 4795 BSSIDs in the dataset. An encoding of the target variable is necessary for the model, so I encode the target variable with a one-hot encoding. I initialize a `MinMaxScaler` that ranges from -1 to 1, as LSTMs use `tanh` as default activation function [22]. Furthermore, the model needs to scale the RSSI values so that they are considered in the learning process. Ensuring that all RSSI features have similar scales makes the learning process more stable and faster.

After this, I use the `window_size` to create sequences with the files. If the length of the file mentioned above is reached, a stop in creating the sequences for this file is done, and the following sequences will be created out of the following file. There are

$$S = \texttt{Length of file} - \texttt{window\_size} + 1$$

sequences per file, which results in

$$S_{\text{total}} = \sum_{i=0}^{146} (\texttt{Length of } i^{\text{th}} \texttt{ file} - \texttt{window\_size} + 1)$$

sequences in total.

I shuffle the order of the sequences to ensure that the model does not learn any unintended pattern from the order of the data. Instead of a basic train-test split, I use k-fold cross-validation. This trains the model on four partitions and tests on the remaining one, ensuring a proper evaluation. A k-value of five is standard in machine learning due to its balance between computational efficiency and robust evaluation across varied data subsets.

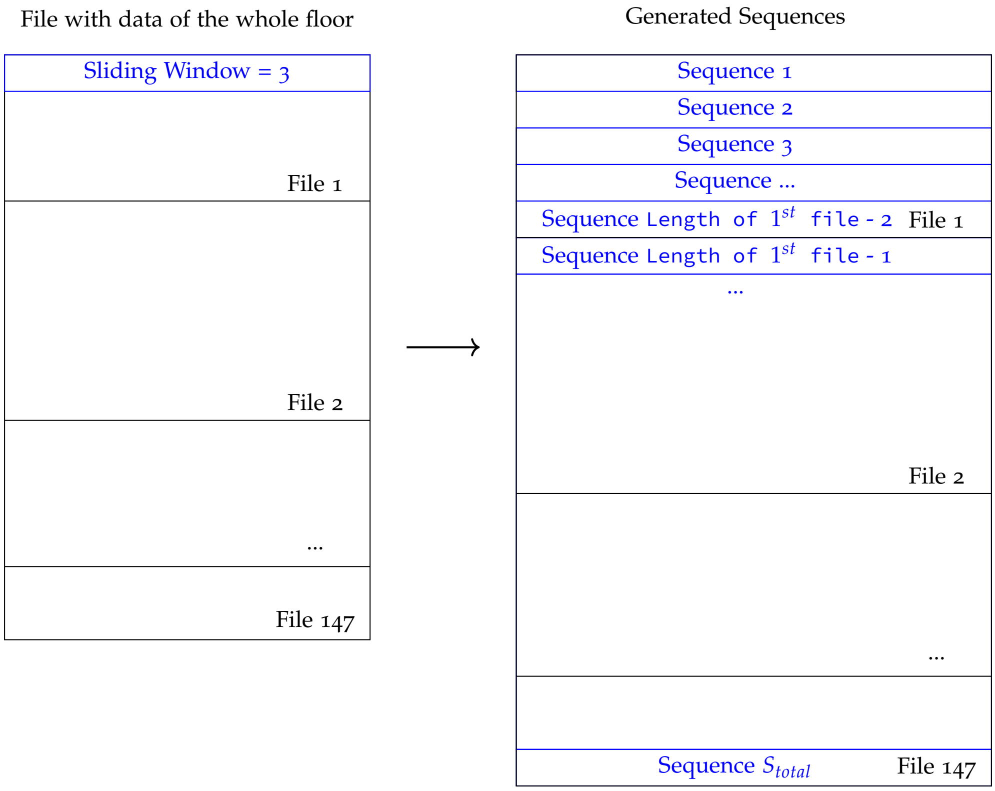

Figure 6.1: Size of sequence generation for all files

6.2 LSTM Tuning, Training and Testing

I test different models with keras-tuner `RandomSearch` with different hyperparameters. As described in Section 2.5, we are tuning hyperparameters such as the number of units for the LSTM layer, the dropout rate, and the batch size. As an LSTM model needs at least one LSTM layer, the number of layers must be set, which is one hyperparameter of the model. The number of units that the tuner tries out is between 64 and 1024, with a step size of 64.

The first LSTM layer's architecture is contingent upon the potential presence of a subsequent LSTM layer, and it gets the number of samples, timesteps, and features. If there will be a second LSTM layer, the first LSTM must return sequences to feed the subsequent layer; it has the same step sizes if chosen to be tried out by the tuner. This conditional structure provides flexibility in model depth. A Dropout layer can be optionally added to randomly select neurons to be ignored during training, helping to prevent overfitting, as mentioned in Chapter 5. The value of the dropout rate is between 0 and 0.5, with a step size of 0.05.

A Batch Normalization layer can also be optionally added. Batch normalization standardizes the activations of a given input volume before passing it to the next layer, helping improve the model's convergence speed and overall accuracy.

The final layer is a Dense layer with a `softmax` activation function, which is needed for multi-class classification problems to get the probabilities for each class [11]. As optimizers, the `RandomSearch` tries out `Adam`, `SDG`, and `RMSprop` with adapted learning rates, which are used to minimize the loss function.

A learning rate may be selected with a value between $1e^{-6}$ and $1e^{-4}$ with a step size of $1e^{-6}$. If a learning rate is not set, each optimizer's default learning rate is 0.001. Also, a batch size is tried out between 16 and 128 with a step size of 16.

Finally, the model is compiled and tested with the chosen optimizer and the loss function `categorical_crossentropy`, which converts the probabilities to target values.

This `RandomSearch`, which will be executed 25 times to try out many combinations, leads to the following hyperparameters:

- lstm_units: 512
- second_lstm_layer: False
- dropout: True with rate 0.3
- batch_norm: True
- learning_rate: False
- optimizer: sgd
- batch_size: 96

The resulting model is shown in Figure 6.2.

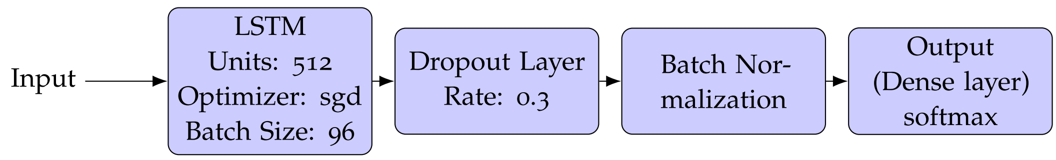

Figure 6.2: The final model architecture.

7 Evaluation

For the evaluation of the model, I do the following: First, the predictions of the test set are in `X_test`. Since the predictions are probabilities for each class, conversing the highest probability as true labels in `y_test` is done because this value will be the predicted AP. Those are one-hot encoded, as described in Chapter 6. Then, I decode the one-hot encoded classes to the original classes, which can be done by `inverse_transform` with the encoder.

Finally, I select the target value of the predictions and compare it with the corresponding true label for the timestamp. If they are equal, the prediction is correct, otherwise it is false. The `3AP` prediction accuracy of the LSTM model, so that one of the three predicted BSSID is the one with the highest RSSI value, is 76 %. The predictions ranging from a set size of one to five are also evaluated, as shown in Figure 7.1.

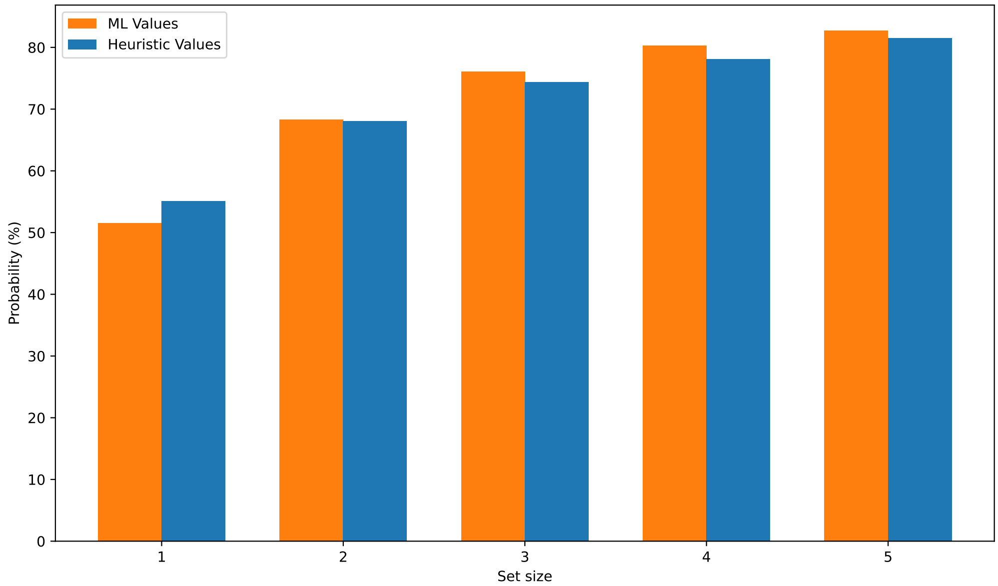

Figure 7.1: Comparison of the Probabilities of correctly predicting the set size of one to five APs in the ML Model and the Heuristic Approach.

Instead of predicting `3AP`, a heuristic approach could be used and will be compared with the model in the following. The heuristic chooses the latest AP with the highest signal strength as following AP. If it is not the following highest RSSI, it will proceed with the latest second highest until the fifth highest to compare it to our approach. Figure 7.1

shows the accuracy of this heuristic approach for choosing the latest highest to the fifth highest AP.

For predicting the set size of one, so the predicted AP is correct, the heuristic's accuracy outperforms the ML model by 3.5 %. For the set size of two, the ML model outperforms the heuristic by 0.3 %. For 3AP the prediction accuracy improves by 1.7 % compared to the heuristic approach. Four and five set sizes outperform the heuristic by 2.2 %, and 1.2 %, respectively. Training for malls with similar-sized floors would likely result in similar accuracies.

Regarding the execution time, the ML model needs nearly 12 hours to tune on a M1 MacBook Pro. The training of the proposed LSTM model takes about 25 minutes for a five-fold cross-validation. The heuristic approach must only choose the three highest RSSI values out of 4795. Choosing the three maximums of a set of 4795 values takes 75 milliseconds for one line of the prepared dataset, which is much faster than training the LSTM model. Hence, the heuristic is much easier to implement and execute than the ML model.

The ML model needs to be trained for each mall floor separately, as the AP are different on each floor. The heuristic approach can be used for all floors, as it only needs the latest RSSI values of the APs.

One of the significant strengths of the machine learning model is that it utilizes user trajectories. However, the heuristic is a simpler and faster approach for predicting the following AP. This approach can capture complex patterns and relationships that other models might overlook. However, there are also inherent weaknesses. The model has to deal with 4795 classes, and given that it only relies on six features, this may compromise its predictive accuracy. This limitation might result in the model predicting worse than initially expected.

8 Conclusion

To reduce the number of roamings in large-scale Wi-Fi environments, this thesis explored a first step towards selecting the following AP a station may connect. The proposed model can be used to predict which AP has the highest probability of being the following AP with the highest RSSI value. However, other information is also needed, such as the load of the AP and the number of connected stations, to know whether the roam will be beneficial. For the selection process and use in future Wi-Fi setups, the prediction needs to be implemented and tested in a real-world environment, which is essential for evaluating the prediction.

AP prediction with user movement is challenging with 4795 classes. LSTM is the best choice among the discussed models for this task, and the prediction accuracy for `3AP` is 76%. The model could also be trained for other floors and sites, which would likely result in similar accuracies.

With 4795 classes, the LSTM model faces the challenge of distinguishing between many classes. Such a high number of classes can introduce greater complexity, making it harder for the model to generalize well across all classes. This could be one reason why the ML model prediction accuracy is only slightly better than the simpler and faster heuristic approach. Moreover, the data not being explicitly generated for this type of prediction suggests that it might not be the most relevant or informative for the task, leading to potential shortcomings in learning the underlying patterns effectively.

Regarding features, incorporating other sensor data, such as gyroscope and magnetic field data, can offer a more comprehensive view of the environment, potentially capturing patterns not evident with the existing features. Additionally, the sliding window size of 3 might not provide enough historical context. An increased window size could offer more temporal information, improving the model's prediction capabilities.

Lastly, generating data from mobile devices or APs to gain knowledge about the location and setup can provide valuable context. Understanding specific locations or setups can aid the model in capturing patterns unique to certain environments, enhancing its predictive power.

Bibliography

[1] Stefanos Bakirtzis, Kehai Qiu, Ian Wassell, Marco Fiore, and Jie Zhang. "Deep-Learning-Based Multivariate Time-Series Classification for Indoor/Outdoor Detection". In: *IEEE Internet of Things Journal* 9 (Dec. 2022), pages 24529–24540. ISSN: 2327-4662. DOI: 10.1109/JIOT.2022.3190555.

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. 2006. ISBN: 978-0387-31073-2. (Visited on July 21, 2023).

[3] R. W. Bohannon. "Comfortable and Maximum Walking Speed of Adults Aged 20-79 Years: Reference Values and Determinants". In: *Age and Ageing* 26.1 (Jan. 1997), pages 15–19. ISSN: 0002-0729. DOI: 10.1093/ageing/26.1.15.

[4] Masoumeh Bourjandi, Meisam Yadollahzadeh-Tabari, and Mehdi GolsorkhtabariAmiri. "Predicting User's Movement Path in Indoor Environments Using the Stacked Deep Learning Method and the Fuzzy Soft-Max Classifier". In: *IET Signal Processing* 16.5 (2022), pages 546–561. ISSN: 1751-9683. DOI: 10.1049/sil2.12125. (Visited on Aug. 27, 2023).

[5] Jason Brownlee. *Deep Learning for Time Series Forecasting*. URL: https://machinelearningmastery.com/deep-learning-for-time-series-forecasting/ (visited on Sept. 3, 2023).

[6] Jason Brownlee. *When to Use MLP, CNN, and RNN Neural Networks*. URL: https://machinelearningmastery.com/when-to-use-mlp-cnn-and-rnn-neural-networks/ (visited on Aug. 23, 2023).

[7] Android Developers. URL: https://developer.android.com/reference/android/net/wifi/WifiManager#calculateSignalLevel(int).

[8] Jeffrey L. Elman. "Finding Structure in Time". en. In: *Cognitive Science* 14.2 (Mar. 1990), pages 179–211. ISSN: 03640213. DOI: 10.1207/s15516709cog1402_1. (Visited on Aug. 6, 2023).

[9] Gabriel O. Ferreira, Chiara Ravazzi, Fabrizio Dabbene, Giuseppe C. Calafiore, and Marco Fiore. "Forecasting Network Traffic: A Survey and Tutorial With Open-Source Comparative Evaluation". In: *IEEE Access* 11 (2023), pages 6018–6044. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2023.3236261.

[10] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. eng. OCLC: 987005922. Cambridge, Massachusetts: The MIT Press, 2016. ISBN: 9780262337434.

[11] Raúl Gómez. *Understanding Categorical Cross-Entropy Loss, Binary Cross-Entropy Loss, Softmax Loss, Logistic Loss, Focal Loss and all those confusing names*. URL: https://gombru.github.io/2018/05/23/cross_entropy_loss/ (visited on Aug. 31, 2023).

[12] H2O.ai. *What is Multiclass classification?* URL: https://h2o.ai/wiki/multiclass-classification/ (visited on Sept. 8, 2023).

[13] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: (1997). URL: https://papers.baulab.info/Hochreiter-1997.pdf.

[14] J J Hopfield. "Neural Networks and Physical Systems with Emergent Collective Computational Abilities." In: *Proceedings of the National Academy of Sciences* 79.8 (1982), pages 2554–2558. DOI: 10.1073/pnas.79.8.2554. (Visited on Sept. 2, 2023).

[15] Xiaokun Hu, Lei Zheng, Jianxiang Chen, Lingwu Sun, and Lihua Zhu. URL: https://mentor.ieee.org/802.11/dcn/22/11-22-1874-02-0wng-roaming-handoff-time-reduction-to-improve-user-experience.pptx.

[16] "IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 2: Fast Basic Service Set (BSS) Transition". In: *IEEE Std 802.11r-2008 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008)* (July 2008). Conference Name: IEEE Std 802.11r-2008 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008), pages 1–126. DOI: 10.1109/IEEESTD.2008.4573292.

[17] "IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 11: Wireless LAN Medium Access Control (MAC)and Physical Layer (PHY) Specifications Amendment 1: Radio Resource Measurement of Wireless LANs". In: *IEEE Std 802.11k-2008 (Amendment to IEEE Std 802.11-2007)* (June 2008). Conference Name: IEEE Std 802.11k-2008 (Amendment to IEEE Std 802.11-2007), pages 1–244. DOI: 10.1109/IEEESTD.2008.4544755.

[18] *Indoor Location & Navigation | Kaggle*. URL: https://www.kaggle.com/competitions/indoor-location-navigation (visited on July 11, 2023).

[19] *Indoor Navigation: Complete Data Understanding*. URL: https://kaggle.com/code/andradaolteanu/indoor-navigation-complete-data-understanding (visited on Apr. 25, 2023).

[20] Milagros Jaén-Vargas, Karla Miriam Reyes Leiva, Francisco Fernandes, Sérgio Barroso Gonçalves, Miguel Tavares Silva, Daniel Simões Lopes, and José Javier Serrano Olmedo. "Effects of Sliding Window Variation in the Performance of Acceleration-Based Human Activity Recognition Using Deep Learning Models". In: *PeerJ Computer Science* 8 (Aug. 2022), e1052. ISSN: 2376-5992. DOI: 10.7717/peerj-cs.1052. (Visited on Aug. 27, 2023).

[21] Faith Karabiber. *Binary Classification*. URL: https://www.learndatasci.com/glossary/binary-classification/ (visited on Sept. 3, 2023).

[22] keras. *LSTM layer*. URL: https://keras.io/api/layers/recurrent_layers/lstm/ (visited on Sept. 3, 2023).

[23] Muhammad Asif Khan, Ridha Hamila, Adel Gastli, Serkan Kiranyaz, and Nasser Ahmed Al-Emadi. "ML-Based Handover Prediction and AP Selection in Cognitive Wi-Fi Networks". In: *Journal of Network and Systems Management* 30.4 (Aug. 2022), page 72. ISSN: 1573-7705. DOI: 10.1007/s10922-022-09684-2. (Visited on Apr. 26, 2023).

[24] Joos Korstanje. *How to Select a Model For Your Time Series Prediction Task.* URL: https://neptune.ai/blog/select-model-for-time-series-prediction-task (visited on Aug. 23, 2023).

[25] Mariyam Mirza, Joel Sommers, Paul Barford, and Xiaojin Zhu. "A Machine Learning Approach to TCP Throughput Prediction". In: *ACM SIGMETRICS Performance Evaluation Review* 35.1 (June 2007), pages 97–108. ISSN: 0163-5999. DOI: 10.1145/1269899.1254894. (Visited on Apr. 27, 2023).

[26] J. Montavont and T. Noel. "IEEE 802.11 Handovers Assisted by GPS Information". In: *2006 IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*. June 2006, pages 166–172. DOI: 10.1109/WIMOB.2006.1696358.

[27] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. *On the Difficulty of Training Recurrent Neural Networks*. Feb. 2013. arXiv: 1211.5063 [cs]. (Visited on Aug. 17, 2023).

[28] Marius-Constantin Popescu, Valentina E Balas, Liliana Perescu-Popescu, and Nikos Mastorakis. "Multilayer perceptron and neural networks". In: *WSEAS Transactions on Circuits and Systems* 8.7 (2009), pages 579–588. URL: https://www.academia.edu/download/69679997/29-485.pdf.

[29] Pratap S. Prasad and Prathima Agrawal. "Movement Prediction in Wireless Networks Using Mobility Traces". In: *2010 7th IEEE Consumer Communications and Networking Conference*. Jan. 2010, pages 1–5. DOI: 10.1109/CCNC.2010.5421613.

[30] L.R. Rabiner. "A tutorial on hidden Markov models and selected applications in speech recognition". In: *Proceedings of the IEEE* 77.2 (Feb. 1989), pages 257–286. ISSN: 00189219. DOI: 10.1109/5.18626. URL: http://ieeexplore.ieee.org/document/18626/ (visited on Aug. 6, 2023).

[31] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323 (1986), pages 533–536. URL: https://api.semanticscholar.org/CorpusID:205001834.

[32] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56 (2014), pages 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[33] Lina Wilske. *GitHub Repository for bachelor thesis*. URL: https://github.com/linaScience/ba-implementation.

[34] Tong Yu and Hong Zhu. *Hyper-Parameter Optimization: A Review of Algorithms and Applications*. Mar. 2020. DOI: 10.48550/arXiv.2003.05689. eprint: 2003.05689 (cs, stat). (Visited on Sept. 3, 2023).

Acronyms

AP	Access Point
BSSID	Basic Service Set Identifier
GPS	Global Positioning System
HMM	Hidden Markov Model
LSTM	Long Short-Term Memory
MAC	Media Access Control
MajorID	Major Identifier
MinorID	Minor Identifier
ML	Machine Learning
MLP	Multilayer Perceptron
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
RSSI	Received Signal Strength Indication
SDN	Software Defined Networking
SSID	Service Set Identifier
TCP	Transmission Control Protocol
TxPower	Transmission Power
UUID	Universally Unique Identifier
Wi-Fi	Wireless Fidelity

List of Tables

List of Figures

List of Listings

Eidesstattliche Erklärung

Hiermit versichere ich, dass meine Bachelor's thesis "Machine Learning-based User Movement Prediction in Layer 2 Networks" selbstständig verfasst wurde und dass keine anderen Quellen und Hilfsmittel als die angegebenen benutzt wurden. Diese Aussage trifft auch für alle Implementierungen und Dokumentationen im Rahmen dieses Projektes zu.

Potsdam, den 11. September 2023,

(Lina Wilske)