Bachelor's thesis

Machine Learning-based User Movement Prediction in Layer 2 Networks

Vorhersage von Benutzerbewegungen in Layer 2 Netzwerken basierend auf Maschinellem Lernen

by
Lina Wilske

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

Hasso Plattner Institute at University of Potsdam

September 1, 2023

Abstract

Mobile device roaming on Wireless Fidelity (Wi-Fi) networks currently does not consider human movement, leading to performance issues. This thesis presents a machine learning approach that uses an LSTM model to predict the nearest Access Point (AP) based on Received Signal Strength Indication (RSSI) values from the surrounding AP. The model was trained using real-world data from a Microsoft Research competition and achieved a top 3 prediction accuracy of 76%. However, the performance was only marginally better than a heuristic method. Improvements could be achieved with fewer classes, a different sized sliding window, additional sensor data. Future research should focus on generated data with more information such as the location of the AP, so that human movement can be utilized to precisely predict the next AP.

Zusammenfassung

Beim Roaming mobiler Geräte in Wi-Fi-Netzwerken werden menschliche Bewegungen derzeit nicht berücksichtigt, was zu Leistungsproblemen führt. Diese Arbeit stellt einen maschinellen Lernansatz vor, der ein Long Short-Term Memory (LSTM)-Modell verwendet, um den nächsten AP basierend auf RSSI-Werten aus dem umgebenden AP vorherzusagen. Das Modell wurde anhand realer Daten aus einem Microsoft Research-Wettbewerb trainiert und erreichte eine Top-3-Vorhersagegenauigkeit von 76%. Allerdings war die Leistung nur unwesentlich besser als bei einer heuristischen Methode. Verbesserungen könnten durch weniger Klassen, ein unterschiedlich großes Schiebefenster und zusätzliche Sensordaten erreicht werden. Zukünftige Forschungen sollten sich auf generierten Daten mit mehr Informationen wie der Position des AP konzentrieren, damit menschliche Bewegungen zur präziseren Vorhersage des nächsten AP genutzt werden können.

Contents

1 Introduction

In large-scale Wi-Fi environments such as office buildings, shopping malls, and airports, where multiple APs are required, people often move around indoors with their mobile devices. To maintain a stable connection to the Service Set Identifier (SSID), the station must remain in the range of the AP or may roam to another AP with the same SSID. However, the current roaming process in 802.11k/r[9][8] does not consider human movement. For example, if a user's station is moving away from AP1 towards AP2 and further towards AP3. AP3 does not initiate the roaming process, but instead the station will connect to AP2 first and then to AP3, which increases the number of handovers. It would be ideal if the movement from AP1 to AP3 was detected and a roam from AP1 to AP3 was initiated. An AP may instruct a client device to roam based on signal strength, but currently without considering the device's trajectory or the user's likely destination. Real time applications such as video conferencing are particularly sensitive to these hand-offs, which may result in dropped connections and unsatisfied users.

Therefore, this thesis will explore if a time series Machine Learning (ML) model can predict the nearest AP a station may connect to next. Because of the many APs in large-scale Wi-Fi environments, this prediction is a multi-class classification problem with many classes. This leads to a hard prediction task, as the model needs to predict the next AP out of many APs. To make the prediction task easier, the model will predict the top 3 APs the station may connect to next. So the nearest station needs to be in the top 3 of the predictions to be considered a correct prediction.

A time series ML model requires time series data as input. There are two possible data sources: generate new or utilize existing data. Data generation needs a comprehensive plan for accounting data setup and collection. As this process is time-consuming and needs a lot of planning and evaluation beforehand, this thesis will not generate data. Thus, this thesis will use a pre-existing dataset with sensor data such as acceleration, waypoint data and Wi-Fi data from large-scale environments. The only dataset which I found with these requirements is from a 2021 competition by Microsoft Research [10] on kaggle [14]. The data will be analyzed in Chapter 3 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 4. Because of findings in Chapter 3, this thesis needs to preprocess the prepared data further and will implement the LSTM model for one site and floor of the competition in Chapter 5. Finally, in Chapter 6, I will evaluate the model's performance and, in Chapter 7, conclude if this prediction could be useful in the future.

2 Background

2.1 Machine Learning

2.1.1 Classification

Classification ML models predict specific categories or classes for input data. By training on input features and labels, these models categorize unseen data. They find use in many domains, producing outputs such as spam or not spam, positive or negative sentiment, and malignant or benign tumors. A specific type of classification is multi-class classification which categorizes more than two classes. [1, pp.179-182] In this thesis, a chosen classification model will predict the next AP a user will be nearest to based on the RSSI of the APs and the trajectory of the user.

2.1.2 Time Series Prediction

Time series prediction is a type of supervised learning problem where a model is trained on a sequence of observations and learns to predict the next value in the sequence. Those sequences consist of data points arranged chronologically, prevalent in numerous domains like stock prices. Due to its inherent temporal dependencies, where subsequent data points influence previous ones, specific machine learning techniques are applied. These include Multilayer Perceptron (MLP), Hidden Markov Model (HMM), and Recurrent Neural Network (RNN) models such as LSTM. Each model is designed to capture and leverage temporal patterns within the data, predicting future trends based on historical observations. [16]

2.1.3 Univariate and Multivariate Time Series

A time series is univariate, if one observation recorded sequentially over time, e.g., temperature or stock prices. The focus of using ML is to understand and forecast a single variable's behavior. In the dataset observed in this thesis, has more than just one value which is recorded sequentially over time such as RSSI, waypoint and sensor data. Therefore, the dataset has multiple observations were recorded over the same time intervals, ML allows for the analysis of interrelationships and interdependencies between these variables, e.g., temperature, humidity, uv-index and wind speed. The focus here is on delving into understanding dynamic interactions and co-movements between multiple variables

2.1.4 Multilayer Perceptron

MLP, also known as a feedforward artificial neural network, is a class of deep learning models primarily used for supervised learning tasks. 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 [5].

2.1.5 Hidden Markov Model

HMM is a statistical model that assumes the system being modeled is a Markov process with unobserved (hidden) states[21]. 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.

2.1.6 Recurrent Neural Networks

RNN is an artificial neural network well-suited to sequential data because of its intrinsic design. Unlike traditional feedforward neural networks, an RNN possesses loops in its topology, allowing information to persist over time. This unique characteristic enables the model to use its internal state (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[4].

2.1.6.1 Long Short-Term Memory

LSTM is a special kind of RNN, capable of learning long-term dependencies, which Hochreiter and Schmidhuber introduced in 1997[7]. 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 accompanying gates (input, forget, and output gate), which regulate the flow of information in the network. LSTMs use `tanh` as activation function.

2.1.7 Hyperparameter tuning

In machine learning, hyperparameters play a vital role in model development. These are parameters such as the learning rate, neural network layers, and the number of windows 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[18] to tune the hyperparameters of the LSTM model.

2.2 Related Work

Montavont et al. [17] propose a handover decision algorithm based on the Global Positioning System (GPS) location of the mobile device. Khan et al. [15] address the problem of handover prediction and AP selection in dense Wi-Fi networks with Software Defined Networking (SDN). They use ML models such as Support Vector Regression (SVR) and MLP for estimating the throughput of the network to. The predictions outperform the current approaches of strongest received signal first by 9.2% and least loaded signal first by 8%.

Both approaches do not consider the trajectory of the mobile device in the handover or AP selection. Furthermore, in large-scale and dense Wi-Fi environments, GPS may not be available or not accurate enough for indoor trajectories. While Khan et al. focuses 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. Lastly, these approaches do not use user generated data, but synthesized data.

This thesis however will use user generated data, instead of synthesized, and will use ML to predict the next AP a user will be nearest to based on the RSSI of the APs and the trajectory of the user. The prediction will be difficult, because the dataset does not provide the location of the APs, and factors such as walls and other obstacles may influence the RSSI. Additionally, there is no information about the throughput or load an access point has, so the predictions cannot be use this information.

These predictions can be used to improve the handover process in Wi-Fi networks initiated by a mobile device, because the predictions use real-world user generated data. This thesis has a different approach, so the results of this thesis cannot be compared to the related work.

3 Dataset analysis and preparation

As mentioned in Chapter 1, the dataset used in this thesis is the Indoor Location & Navigation from kaggle [14], which was part of a competition of Microsoft Research in 2021 [10]. The company XYZ[10] recorded the data in shopping malls and was provided by Microsoft Research for this competition. The goal for the competition was to give a file with a path of a shopping mall and predict the floor and waypoint locations at a timestamp given in the submission files for that shopping mall. The following will analyze and prepare the dataset and data for the ML model

3.1 Components of the dataset

As noted in the kaggle notebook "Indoor Navigation: Complete Data Understanding" [12] 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 site where the data was recorded. In each site folder are a minimum of one and a maximum of twelve subfolders, which represent the floors of the site; the median is five floors. Overall, there are 26,925 files, each containing the movement of one 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)) which was hashed as "5d27075f03f801723c2e360f" 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 our test set out of the train data because our goal is not to predict the floor and site name for a specific timestamp but to predict the Basic Service Set Identifier (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.

3.2 File structure

Each file in each floor folder is a **.txt** file. The first two lines and the last are denoted with "#". The first contains the start time of the recording, the second site information SiteID as hash, SiteName, FloorId as hash, and FloorName. 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 [11] shows that the data type in the second column followed by its data can be one of the following:

(1) TYPE_ACCELEROMETER with x, y and z acceleration and an accuracy value

(2) TYPE_MAGNETIC_FIELD with x, y and z magnetic field and an accuracy value

(3) TYPE_GYROSCOPE with x, y and z gyroscope and an accuracy value

(4) TYPE_ROTATION_VECTOR with x, y and z rotation vector and an accuracy value

(5) TYPE_MAGNETIC_FIELD_UNCALIBRATED with x, y and z magnetic field and an accuracy value

(6) TYPE_GYROSCOPE_UNCALIBRATED with x, y and z gyroscope and an accuracy value

(7) TYPE_ACCELEROMETER_UNCALIBRATED with x, y and z acceleration and an accuracy value

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

(9) 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.

(10) TYPE_WAYPOINT with x and y coordinates, which are 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 (10). Lines with types from (1) to (7) occur every 20 ms and are measured at the same time. (8) occurs about every 1800-2200 ms. (10) data is not evenly distributed. An assumption for this is that the recording of the waypoint data is triggered by an exterior event, e.g., a button press. As seen in Listing 3.1, the data are measured separately from each other, so there are no combinations of the data types.

A prediction of the next BSSID will only work per site due to the different APs and, therefore, BSSIDs per site. 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. To better predict, I will focus on a single floor of a site and use the data from the fourth floor of the shopping mall in Yintai City (Chengxi Branch). To further know how much data there is for the model, Table 3.1 shows a more detailed analysis.

Listing 3.1: 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    9
     f967858afcbb907af6e5adef766c7e7b936ef07    -63 2462    1571462190744
1571462194883   TYPE_WIFI   8204870beb9d02995dab3f08aad97af5eab723cc    0413
     b35df78fc865af15b4721d5aeb33ff57da45    -64 2447    1571462188686
...
1571462194020   TYPE_BEACON 07efd69e3167537492f0ead89fb2779633b04949
     b6589fc6ab0dc82cf12099d1c2d40ab994e8410c    76e907e391ad1856762f70538b0fd13111ba68cd    -57 -71
      5.002991815535578   1b7e1594febd760b00f1a7984e470867616cee4e    1571462194020
...
1571462195943   TYPE_WAYPOINT   59.72475    69.02152
#   endTime:1571462195976
```

Table 3.1: Summary 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 (1) to (7) 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

3.3 Preprocessing data for an ML model

As seen in previous sections, a location for the time of `TYPE_WIFI` data points is not provided. As seen in Table 3.1, this floor has 2,027 waypoints and 1,862,044 lines of Wi-Fi data, there are multiple lines per timestamp, because the devices gathers data from all nearby APs for each timestamp. The visualization of the waypoints can be seen in Figure 3.1.

Further human movement between `TYPE_WAYPOINT` and `TYPE_WIFI` data points may have occurred. Therefore, a combination of the data points directly to get a location for the

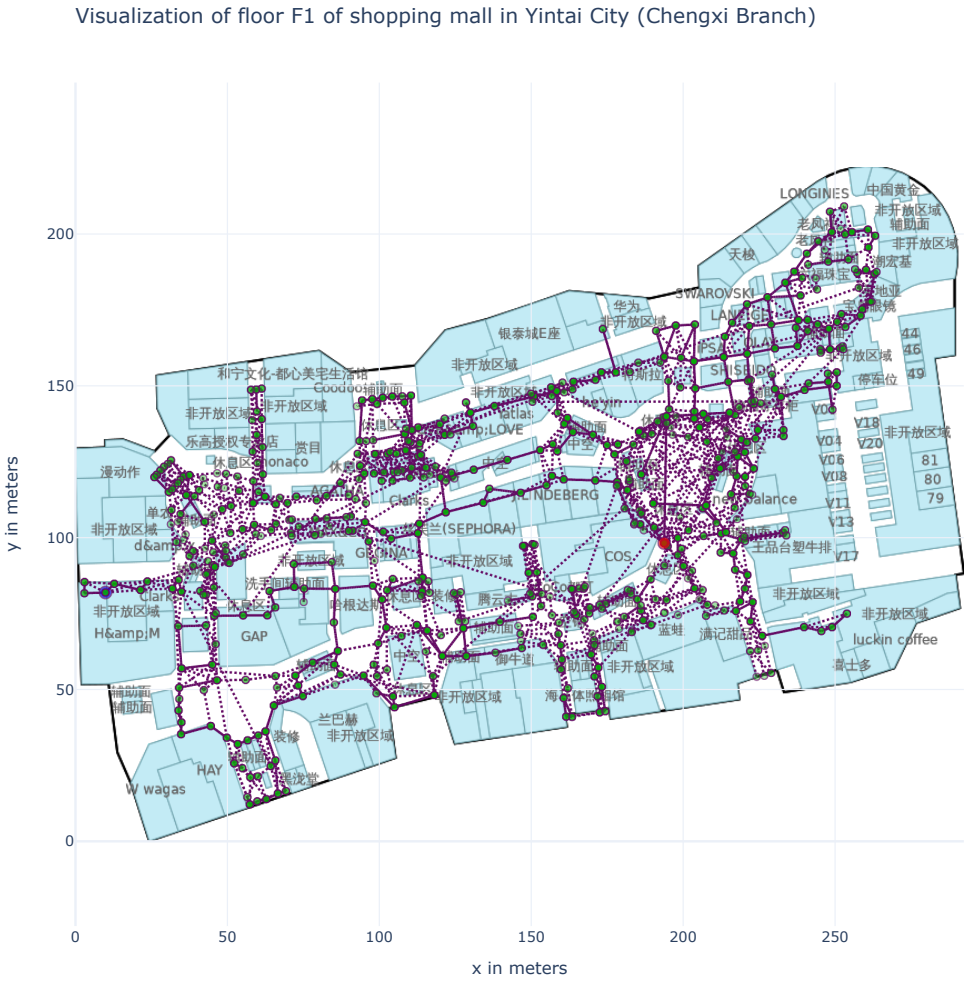

Figure 3.1: Visualization of the 2,027 waypoints of the site Yintai City (Chengxi Branch) on floor F1

Wi-Fi data points will not work, because the waypoint may have changed in that time. Nevertheless, they can be combined using linear interpolation.

Therefore, an interpolation of `TYPE_WAYPOINT` data for `TYPE_WIFI` timestamps will be done 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. Hence, 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 3.2.

As the `TYPE_WAYPOINT` data may have changed, also the `TYPE_ACCELEROMETER` data may have changed. 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.

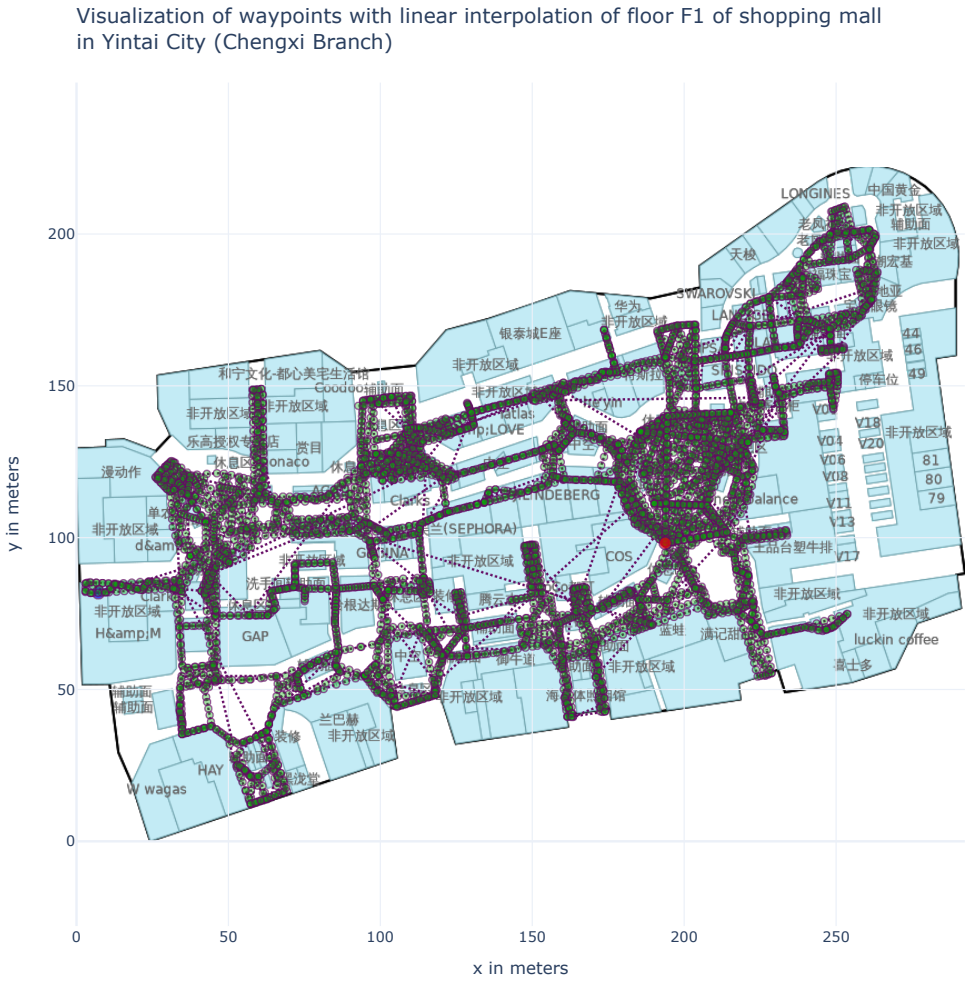

Figure 3.2: Visualization of the interpolated waypoints for site Yintai City (Chengxi Branch) on floor F1

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 genreated. This time series will be used for the machine learning model. Further interpolation of data, such as `TYPE_GYROSCOPE`, is possible, but this thesis only utilizes the abovementioned data. Therefore, it will not be done.

3.3.1 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 3.1 shows, some waypoints seem to be very distant from the next one, which can be detected by the dotted lines that go all across the floor. Listing 3.2 shows the top 10 pairs of waypoints with the most significant metric differences, where 174.77 meters is the most significant difference.

Listing 3.2: Top 10 pairs with the most significant metric differences of data from floor F1 of site Yintai City (Chengxi Branch)

```
1. Point 1: (247.96523998265695, 168.7631635050295), Point 2: (117.92375106521739,
     51.997759545341616), Metric Difference: 174.77113148839442
2. Point 1: (98.66346, 127.5971), Point 2: (258.75049789436116, 181.23350740899357), Metric
     Difference: 168.83342057049657
3. Point 1: (189.58672, 71.454666), Point 2: (89.73448203762376, 102.255128190099), Metric
     Difference: 104.49467879858156
4. Point 1: (223.49295, 145.0939), Point 2: (174.26284532732006, 78.86335505811792), Metric
     Difference: 82.52325908119289
5. Point 1: (34.864815, 35.45561), Point 2: (33.284438514193546, 110.76117936967742), Metric
     Difference: 75.32215057954856
6. Point 1: (50.31085719185683, 92.03105531572366), Point 2: (114.97229034709193,
     123.04521228267667), Metric Difference: 71.71456525741291
7. Point 1: (150.91972285390713, 145.15169976783693), Point 2: (222.23440919809525,
     146.11043967333333), Metric Difference: 71.32113060360388
8. Point 1: (64.64140228107132, 25.345204272946134), Point 2: (34.47475562023909, 84.44615420072283)
     , Metric Difference: 66.35471990088624
9. Point 1: (56.83799274253731, 74.52090035349569), Point 2: (94.43750948519768, 125.97292215289488)
     , Metric Difference: 63.72624425248555
10. Point 1: (172.3439286324042, 56.200716101045295), Point 2: (212.4478039192399,
     99.33967479470648), Metric Difference: 58.90068395354572
```

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 has its maximum at about 2.53 meters per second, according to [2]. So in 2.2 seconds, a human can walk 5.57 meters, which is much less than each of the values in the top 10 in Listing 3.2. Therefore, waypoints, which are more than about 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 generate separated files, 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 do not contain any information about the BSSID and corresponding RSSI values, which are needed for the prediction. This will be solved in Section 3.3.2.

3.3.2 Wi-Fi data for each timestamp

It is evident that for a waypoint, not all RSSI values for each AP is 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, a 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 save it 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 into the field. This value is chosen because it is very unlikely that an AP has this RSSI value. Then, I iterate over each file with interpolated

data, and for each timestamp, the BSSID and the corresponding RSSI value from the Wi-Fi file is added to the interpolated file. At each `TYPE_WIFI` timestamp with waypoint and acceleration data , the RSSI value for each BSSID is now saved. The code for this can be found in the GitHub repository [23] in the file `preparation.ipynb`. These files will train the ML model.

4 Suitable Machine Learning Model

As mentioned in Chapter 3, the floor analyzed there has 4795 BSSIDs. Resulting in 4795 classes for the classification problem. The selection of a suitable model for this task is even more critical. 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 I want to predict the next BSSID, I need a model that can handle temporal dependencies. MLPs have no loop [20], making them less suitable for time series data where temporal dependencies are crucial. While HMMs can handle temporal dependencies to some extent, they often struggle with longer sequences and multivariate data due to their Markovian assumption [21], which limits their memory to the most recent state. Standard RNNs were designed to handle temporal dependencies, but they suffer the vanishing gradient problem, making them less effective in capturing long-term dependencies compared to LSTMs [19]. LSTMs, by design, are equipped to handle long-term temporal dependencies. Their unique cell state and gating mechanisms allow them to store, modify, and access information over extended periods, making them adept at capturing patterns from long sequences [7].

Capacity and Complexity MLPs can also scale their capacity by adding more hidden layers and units. They are capable of modeling complex relationships within data through their nonlinear activations. HMMs have limitations in handling the complexity of multi-class and multivariate problems due to their inherent Markovian assumptions and discrete state representations. Traditional RNNs suffer from the vanishing gradient problem, especially in longer sequences, which limits their ability to capture long-term dependencies effectively [19]. With 4,795 classes, the model needs a considerable capacity to differentiate between the subtle differences in patterns that might exist among them. 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 While MLPs can also handle multivariate data, they treat each feature and time step independently, often missing out on 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 effectively process each time step with multiple features.

Flexibility and Integration MLPs are very flexible and can be used generally to learn a mapping from inputs to outputs. [3] As I want to learn, which BSSID is the next one, this may be a good fit. HMMs are primarily designed for capturing state transitions in sequential data and may not be suitable for tasks requiring the integration of spatial and temporal information. Their rigid assumptions about state transitions limit their flexibility in capturing complex patterns [21]. Traditional RNNs can capture short-term dependencies and are relatively simpler to integrate with other architectures due to their sequential nature. LSTMs can be easily integrated with other deep learning architectures, such as Convolutional Neural Networks (CNNs), to capture both temporal and spatial features. This flexibility is advantageous when dealing with complex and varied data sources.

Regularization and Overfitting Dropouts may be used to prevent overfitting for each model [22]. But traditional RNNs are prone to vanishing gradient issues, which can increase overfitting in general [19]. Therefore, MLPs, HMMs or LSTMs are better suited in this regard.

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

Table 4.1: Suitability of models for various requirements.

In conclusion, 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. This problem demands a model that can efficiently capture intricate temporal patterns, scale in capacity, and handle multivariate data. LSTMs, with their unique architecture and properties, can deal with this challenge, making them the preferred choice for this task and the selected model for our implementation.

5 Implementation

All the code for this implementation can be found in the GitHub repository [23]. It is structured in preprocessing and LSTM implementation, which will be described in the following sections.

5.1 Preprocessing

As described in Chapter 3, 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 data values x and y in meters, the fourth to sixth columns contain the acceleration data values x, y, and z in meters per second squared, and 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. [13], 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 our dataset has Wi-Fi timestamps for every about 2 seconds. If a file has less 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 our 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 our 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` which ranges from -1 to 1, as LSTMs use `tanh` as activation function. Furthermore, our model needs to scale the RSSI values so that the RSSI values are considered in the learning process. By ensuring all RSSI features have similar scales, the learning process can be more stable and faster.

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

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

sequences per file, which results in

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

sequences in total.

I shuffle the data to eliminate chronological biases. Instead of a basic train-test split, I use 5-fold cross-validation. This trains the model on 4 partitions and tests on the remaining one, ensuring a comprehensive evaluation. A k-value of 5 is standard in machine learning due to its balance between computational efficiency and robust evaluation across varied data subsets.

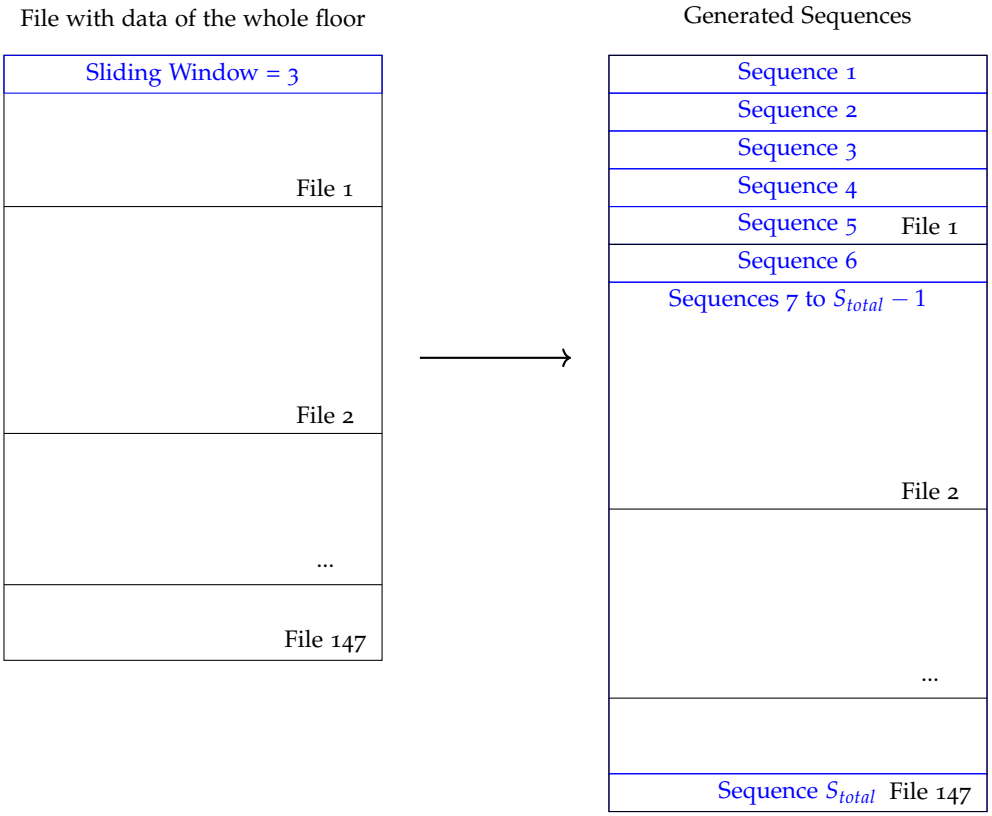

Figure 5.1: Size of sequence generation for all files

5.2 LSTM Tuning, Training and Testing

I test different models with keras-tuner [18] `RandomSearch` with different hyperparameters. As a LSTM model needs at least one LSTM layer, the number of layers must be set, which is one hyperparameter of the model. 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. This conditional structure provides flexibility in model depth. The activation function for the LSTM layers will be `tanh` as it is a traditional choice. A Dropout layer can be optionally added, to randomly select neurons to be ignored during training, helping prevent overfitting, which was men-

tioned in Chapter 4. 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 [6]. As optimizers the `RandomSearch` will try out `Adam`, `SDG`, and `RMSprop` with adapted learning rates, which are used to minimize the loss function. If a learning rate should not be set, the default learning rate for each of the optimizers is 0.001.

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` 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

When configured with the parameters mentioned above, the model is shown in Figure 5.2.

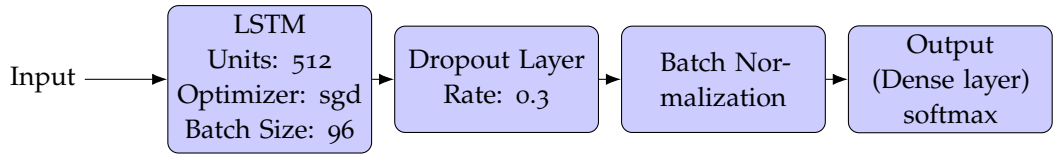

Figure 5.2: The final model architecture.

6 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, a conversion of 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 5. 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 highest probabilities 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 top 3 prediction accuracy of the LSTM model results in 76%. The predictions ranging from top 1 to top 5 are also evaluated, which can be seen in Figure 6.1.

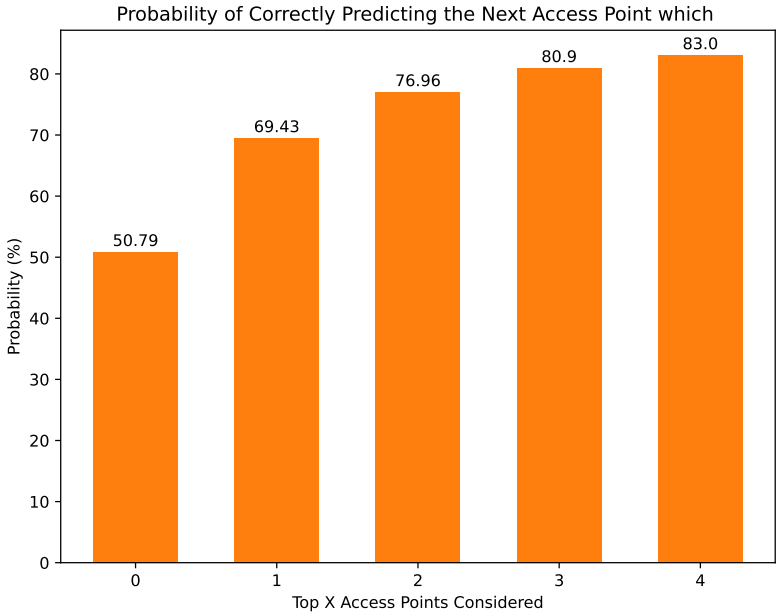

Figure 6.1: Accuracy of the model that the predicted AP is among the top X.

Instead of predicting the top 3 APs, a heuristic approach could be used and will be compared with the model in the following. The heuristic chooses the AP with the highest signal strength as next AP, if this is not the next strongest, it will proceed with the second strongest and so on. Figure 6.2 shows the accuracy of this heuristic approach for choosing the strongest to the fifth strongest AP.

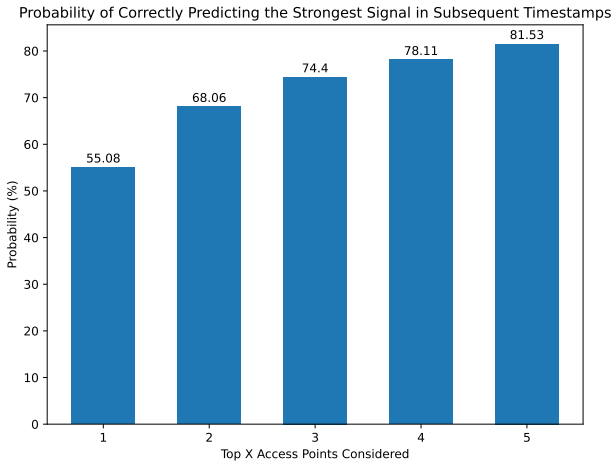

Figure 6.2: Accuracy of the model that the heuristic AP is among the top X.

Also, for predicting the top 1 to top 5 AP the ML models predictions are from Top 2 on better than the heuristic approach, as seen in Figure 6.3. As seen, the ML model is slightly better than the heuristic approach for the top 3 APs.

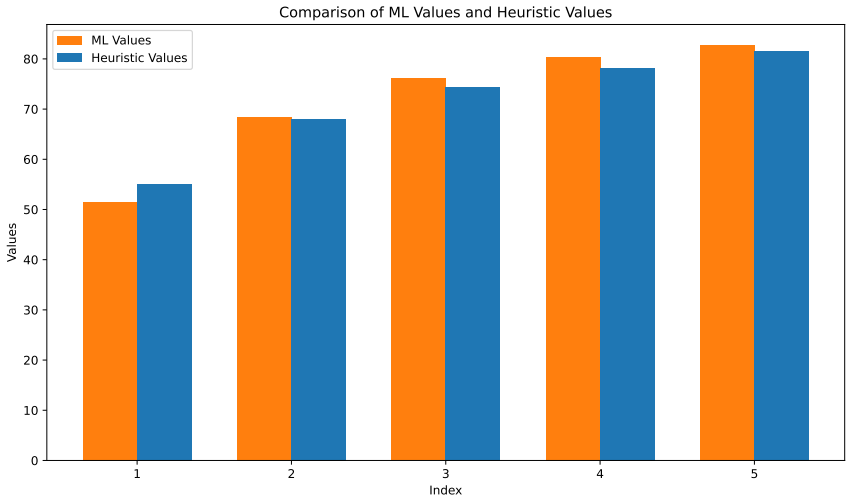

Figure 6.3: Comparison of the Probabilities of correctly predicting the Top 1 to Top 5 APs in the ML Model and the Heuristic Approach.

7 Conclusion

AP prediction with user movement is a hard task with many classes. Although LSTM is the best choice among the discussed models for this task, the prediction accuracy for selecting on of the top 3 APs is XX%. In the future, the model could also be trained for other floors and sites, which would likely result in similar accuracies.

With fewer classes, the LSTM model could predict better, which could be a reason why ML model prediction is only slightly better than the simpler heuristic approach. Furthermore, the model could be enhanced with other sensor data such as gyroscope and magnetic field data. Maybe the sliding window size is too small, and a larger window size could improve the prediction. Also, one could generate data from mobile devices or APs, so that more information such as location of the AP and setup is known and can be used in prediction.

Bibliography

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. 2006. (Visited on July 21, 2023).

[2] 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.

[3] 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).

[4] 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. URL: http://doi.wiley.com/10.1207/s15516709cog1402_1 (visited on Aug. 6, 2023).

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

[6] 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).

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

[8] "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.

[9] "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.

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

[11] *indoor-location-navigation-20*. URL: https://github.com/location-competition/indoor-location-competition-20 (visited on July 31, 2023).

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

[13] 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).

[14] *Kaggle: Your Home for Data Science*. URL: https://www.kaggle.com/ (visited on July 23, 2023).

[15] 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).

[16] 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).

[17] 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.

[18] Tom O'Malley, Elie Bursztein, James Long, François Chollet, Haifeng Jin, Luca Invernizzi, et al. *KerasTuner*. https://github.com/keras-team/keras-tuner. 2019. (Visited on Aug. 21, 2023).

[19] 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).

[20] Marius-Constantin Popescu, Valentina Balas, Liliana Perescu-Popescu, and Nikos Mastorakis. "Multilayer perceptron and neural networks". In: *WSEAS Transactions on Circuits and Systems* 8 (July 2009).

[21] 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).

[22] 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.

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

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
SVR	Support Vector Regression
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" ("Vorhersage von Benutzerbewegungen in Layer 2 Netzwerken basierend auf Maschinellem Lernen") 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 1. September 2023,

(Lina Wilske)