Gathering Bicycle Ride Data and Using it for the Good

vorgelegt von

M.Sc.

Ahmet-Serdar Karakaya

ORCID: 0000-0002-2040-2782

an der Fakultät IV – Elektrotechnik und Informatik

der Technischen Universität Berlin

zur Erlangung des akademischen Grades

Fakultät IV

Technische Universität Berlin

Promotionsausschuss:

Vorsitzender: Prof. Dr. Stefan Tai

Gutachter: Prof. Dr. David Bermbach

Gutachter: Prof. Dr. Heather Kaths

Gutachter: Prof. Dr. Odej Kao

Tag der Wissenschaftlicher Aussprache: 21.03.2024

März 2024

Sworn Affidavit

I hereby declare that the thesis submitted is my own, unaided work, completed without any unpermitted external help. Only the sources and resources listed were used.

The independent and unaided completion of the thesis is affirmed by affidavit:

Berlin, January 1st, 1970

Jana Eggers

II

Abstract

An increased modal share of bicycle traffic is a key mechanism to reduce emissions and solve traffic-related problems. However, a lack of (perceived) safety keeps people from using their bikes more frequently. To improve safety in bicycle traffic, city planners need an overview of accidents, near miss incidents, and bike routes. Such information, however, is currently not available. In this paper, we describe SimRa, a platform for collecting data on bicycle routes and near miss incidents using smartphone-based crowdsourcing. We also describe how we identify dangerous near miss hotspots based on the collected data and propose a scoring model.

Kurzdarstellung

Eine Erhöhung des Fahrradverkehrs ist ein wichtiges Instrument zur Verringerung der Emissionen und zur Lösung verkehrsbedingter Probleme. Allerdings hält ein Mangel an (gefühlter) Sicherheit die Menschen davon ab, ihr Fahrrad häufiger zu benutzen. Um die Sicherheit im Radverkehr zu verbessern, benötigen Stadtplaner einen Überblick über Unfälle, Gefahrensituationen und Radwege. Solche Informationen sind jedoch derzeit nicht verfügbar. In diesem Beitrag beschreiben wir SimRa, eine Plattform zur Sammlung von Daten über Fahrradrouten und Gefahrensituationen mithilfe von Smartphone-basiertem Crowdsourcing. Außerdem beschreiben wir, wie wir auf der Grundlage der gesammelten Daten gefährliche Hotspots für Gefahrensituationen identifizieren und ein Scoring-Modell vorschlagen.

Table of Contents

List of Figures

List of Tables

List of Listings

Part I

Foundations

Chapter 1

Introduction

The global climate crisis is and will be one of mankind's main challenges in the coming decades. More and more societies worldwide are aware of this problem and search for solutions to adapt to the new changes that comes with the climate change and try to decrease the impact of the global climate crisis. For the latter, the main approach is to reduce greenhouse gas emissions such as CO_2 and NO_x. This is why cities worldwide try to increase the modal share of bicycle traffic and by doing so, decrease the usage of individual motorized transport, which is one of the main greenhouse gas emitters in urban areas (CITATION NEEDED). This, however, puts cyclists, city planners and politicians into a difficult spot. Since cities were developed with the individual motorized transport as the main transportation mode in mind, the traffic infrastructure highly favors the usage of cars. Because of that, cyclists often have a hard time commuting, since they have to cycle on roads that were made for cars, which often puts them into dangerous and stressful situations. Although crash statistics do not convey an increased danger for cyclists (CITATION NEEDED), studies have shown, that the perceived safety of commuting with a bicycle is very low (CITATION NEEDED) and one of the main reasons why people prefer other transportation modes to cycling (CITATION NEEDED). It is up to city planners to improve the traffic infrastructure for cyclists, but it is not easy to pinpoint the most problematic areas, that need to be prioritized for them to be fixed, which

is needed because of the limited resources available to the city planning department. Besides, city planners are bound by city planning codes and rules, which oftentimes also reflect the car-centric approach. This passes the responsibility to the politicians, who need to change the city planning codes and rules and also increase the funds for bicycle infrastructure. For this, however, they need more than anecdotal evidence, that the perceived safety in bicycle traffic is low, so that they can change the legislation and increase funding in favor of bicycle infrastructure.

1.1 Problem Statement

To enable city planners and politicians to do evidence-based decision-making, they need a) to be aware of the problems in bicycle traffic and b) to know where and how to improve the bicycle infrastructure. There are surveys that show that the perceived safety in bicycle traffic is low (MULTIPLE CITATIONS) DO I NEED TO GIVE EXAMPLES HERE FOR SURVEYS?

1.2 Contributions of this Thesis

1.3 Structure of this Thesis

Chapter 2

Background (rename?)

2.1 Crowdsensing

2.2 Citizen Science

2.3 Deep Learning

2.4 SUMO

SUMO is an open source traffic simulation tool that offers macroscopic as well as microscopic simulation of vehicle mobility [**lopez2018microscopic**]. SUMO includes models for different types of "vehicles", including, among others, cars, bicycles, and even pedestrians. Due to its large feature set, it has become the de-facto standard for traffic simulation and is used even beyond the transport community, e.g., [**beilharz2021towards**].

Traffic scenarios are, among other things, defined by road networks and vehicle traffic. The road network includes roads and their (sub-)lanes as well as exclusive lanes for cyclists and pedestrians, or road-side infrastructure such as traffic lights. Furthermore, connections between these lanes and traffic lights can be configured.

When modeling vehicle traffic, users specify demand for a specific road segment per vehicle type and can adjust vehicle-specific parameters of SUMO's simulation model to control their respective behavior. In general, vehicle parameters are usually specified in the vehicle type declaration (*vType*), applying the changes to all instances of the respective *vType*, e.g., to all cars. An alternative, however, is to obtain multiple *vType* realizations which typically differ in at least one parameter by using so-called *vTypeDistributions*. This way, when spawning a new vehicle, SUMO randomly picks a specific *vType* from the *vTypeDistribution* and instantiates the vehicle's parameters accordingly, e.g., cars can thus have individual maximum velocities.

In SUMO, vehicle behavior is, among other things, defined by *Car Following (CF) models* for the longitudinal kinematic behavior, *Lane Change (LC) models* for the lateral kinematic behaviour, and *junction models* for the behavior at junctions and intersections.

Despite including several of these models for cars and trucks, SUMO does not provide a dedicated movement model for cyclists. Instead, cyclists are simulated by modeling them either as slow cars or fast pedestrians. Both of these approaches use movement models of the corresponding vehicle type and adapt their respective shape and kinematic characteristics (e.g., velocity and acceleration profiles) to match cyclists. While this is obviously a rough approximation, it is unlikely to reflect the behavior of real-world cyclists [**grigoropoulos2019modelling**].

2.5 Surface Quality

Traffic departments frequently analyze the road surface quality to find out road segments that need to be repaired to increase the traffic safety. There are mainly three types of methods

used for, which we briefly introduce here. *Profilographs* are one of the oldest devices used to measure road surface quality. They consist of a profiling wheel in the center and multiple support wheels held together by a rigid frame. They measure the road quality by tracking the vertical movement of the profiling wheel. The higher the vertical movement of that wheel, the worse the surface quality of the road. This method of measuring surface quality is very slow, since the profilograph has to be pulled very slowly by another vehicle. *Scanner-based systems* are more sophisticated and rely either on accelerometers measuring the vibrations caused by driving on a specific surface, lasers scanning the surface in front or beneath the vehicle, cameras making photos later to be analyzed with the help of computer vision and photogrammetry, or a combination of the aforementioned methods. This system provides the highest resolution and accuracy, however it is also the most costly and complex one. More recently, *smartphone-based systems* are gaining traction due to their cost-efficiency. A smartphone is being attached inside the car and the vibrations caused by the road surface are recorded. This is the least accurate system, since the measured vibrations are very indirect, due to car tires and suspension.

However, these systems are impractical for measuring bicycle road surface quality, because cars are not suited to drive on bicycle roads and these systems are too costly. Hence, we propose a novel system using crowdsourced smartphone bicycle ride data.

Chapter 3

Contributions

3.1 CycleSense

In recent years, more and more cities worldwide aim to reduce the modal share of car traffic in favor of bicycle traffic to reduce NO_x, CO_2, and particulate matter emissions, to reduce traffic jams, and to free up space that is urgently needed for other purposes ranging from vegetation that provides natural cooling in a heating world to new flats for a growing population.

A key mechanism to support this is to make bicycle traffic more attractive. In practice, however, what keeps people from using their bikes more frequently is a lack of safety or perceived safety in cities with a car-centric traffic infrastructure [2]. Hence, city planners urgently require an overview of safety and perceived safety in their city.

Aside from actual accidents, an often overlooked aspect of cycling safety are near miss incidents[1] such as close passes or near dooring [2, 25]. Information on these have previously not been available or have not been easily accessible as they are distributed over private video recordings, social media posts, public CCTV footage, and other sources. In 2019, we

[1] In the remainder of this paper, we will also refer to them as "incidents".

therefore launched the SimRa[2] project in which cyclists record their rides and annotate them with incident information via a smartphone app [25].

While our previous approach [25] already used a rudimentary heuristic for automatically detecting incidents based on acceleration sensors, it is inherently limited resulting in significant manual annotation efforts which makes it unattractive for some groups of cyclists – in particular, elderly cyclists who have problems using smartphone apps and cyclists with significant daily mileage for whom the labeling approach is too much effort. To increase participation, it is hence crucial to decrease manual efforts through an improved pre-detection of incidents.

As a first step towards this goal, we supervised a master's thesis which developed a neural network-based method for incident detection using the public SimRa data set[3] [31] which, however, does not show the desired detection quality despite being an improvement over our original heuristic. Both detection methods are currently used in the live version of the app, hence, we will later use them as baseline in our evaluation. Besides that, there are alternative approaches for quantifying the perceived safety of bicycle traffic, e.g., [8, 9, 41], but to the best of our knowledge no other methods are based on mobile motion sensory data.

Since its start in 2019, the SimRa data set has grown to more than 65,000 rides with almost 30,000 reported incidents. These amounts of data now enable the application of much more sophisticated methods, namely Machine Learning (ML) and/or DL, for automatic incident detection. Therefore, we here propose CycleSense – a DL model trained on smartphone sensory data from the SimRa data set to detect incidents in the SimRa app. We hope, that it will lead to more reported incidents due to an easier reporting, and make the following contributions:

- We propose an approach that combines signal processing and ML techniques to detect incidents based on motion sensor data of cyclists with an Area under the curve (AUC) Receiver Operating Characteristic (ROC) of 0.906 (Section **??**).

[2]SimRa is a German acronym for safety in bicycle traffic.
[3]https://github.com/simra-project/dataset

- We evaluate our approach using the SimRa data set and compare it to two baselines as well as common Deep Learning models used in the context of Time Series Classification (Section **??**).
- We discuss to which degree our approach can automate incident detection and which additional sensors are needed for full automation (Section **??**).

3.1.1 Detecting Near Miss Incidents

This section describes the process of automatically detecting incidents. Please note that the SimRa data are not optimized for automated processing and ML but rather for aggregated statistics and review by humans. Therefore, several preprocessing steps are needed (Section **??**). We describe our ML model in Section **??** and the training process in Section **??**.

Preprocessing

To overcome some limitations of the SimRa data set, we use data cleaning and preprocessing steps, in a sequential multi-stage manner, some of which are specific to some model types that will be used afterwards to classify the incidents within rides.

Before the preprocessing phase, a typical ride can be expressed by a $n \times d$ sparse matrix $X^{(i)}$, where d describes the number of sensor features and n represents the number of timestamps in a given ride $i \in \{1, ..., R\}$.

Note that we typically only use the accelerometer, gyroscope, and GPS sensor features. Using the linear accelerometer features in addition did not lead to a significant improvement. Furthermore, the linear accelerometer and the rotation vector features are only available in the newer Android rides. The non-sensory features such as phone location and bike type have a non-logical strong correlation with incidents caused by issues in the data recording phase and are therefore not used.

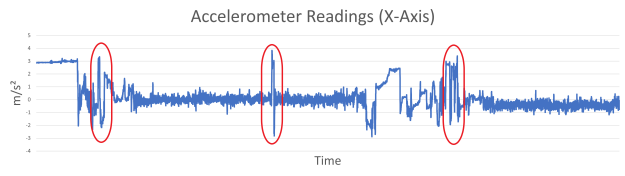


Figure 3.1: The accelerometer readings of an example ride. The encircled spots could indicate an incident but also driving over a curb.

The preprocessing pipeline starts off with a manual label cleaning procedure that aims to remove some of the wrongly labeled incidents. Identifying them solely based on time series data is practically impossible for a human. When visualizing the accelerometer sensor readings, incidents usually stand out as sudden spikes (see Figure 3.1), but they also could be caused by driving over a curb or suddenly stopping due to a red light. Therefore we focus on the incidents that feature an additional description that was provided by the users. As it is very time-consuming, this is the only manual preprocessing step, and we apply it only to the newer Android data set. Note that this procedure did not result in a fully cleaned data set. Next, the timestamps within rides are sorted. Afterwards, we sort out invalid rides, i.e., rides that contain adjacent timestamps that have been recorded with a gap of more than 6 seconds. Furthermore, we remove outliers based on the statistical definition of outliers by Tukey et al. [37] that characterizes a data point as an outlier if it fulfills one of the equations $Outlier < q_{25} - k \cdot IQR$ or $Outlier > q_{75} + k \cdot IQR$ where the Interquartile Range (IQR) is equal to the difference between the upper and lower quartiles [38, 44]. The k-values we are utilizing are 1.5 for GPS outliers regarding the accuracy feature and 3.0 for velocity outliers, as we have seen reasonable results for these values.

In a further step, speed is calculated from the distance between two GPS coordinates and their respective timestamp. Moreover, the accelerometer and gyroscope sensor data are interpolated to create equidistance over the whole time series. This is advisable, as unevenly spaced time series data tend to pose a problem to typical ML solutions [39]. Therefore, we up-sample to a frequency of 10 Hz via linear interpolation on uniformly generated timestamps with a 100 ms interval. That means the up-sampling factor is usually above 2. Although some

argue that interpolation is a bad solution for unevenly spaced time series data in the context of TSC [15, 19], initial experiments have shown that this improves model performance. This preprocessing stage results in dense matrices $X^{(i)}$

For better convergence of the stochastic gradient descent optimizer used in the neural network, we normalize each feature individually by its maximum absolute value. This is nearly always an advantageous preprocessing step as it improves model stability [7].

Training the model on individually labeled timestamps did not appear to be a promising approach since incidents have a certain duration, which is typically longer than 100 ms, and it is highly unlikely that the user correctly specifies the label at the exact timestamp when the incident occurred. For that reason, we split our ride data into 10-second buckets, following a non-overlapping sliding window approach [28]. These buckets are then labeled in the following manner: we define a bucket as an incident bucket if any timestamp inside that bucket was labeled as an incident. Otherwise, we define it as a non-incident bucket.

Additionally, we apply a one-dimensional f-point Discrete Fourier Transform (DFT) on each dimension of the accelerometer and the gyroscope sensor data contained in a bucket individually. This results in a more advanced temporal feature extraction approach that exploits the spectral power changes as time evolves by converting the time series from the time domain to the frequency domain [12].

To cope with the heavy label imbalance (e.g., $\approx 1 : 170$ on rides that have recently been recorded on Android devices) that is present in the data, we use a Generative Adversarial Network (GAN) with a Convolutional Neural Network (CNN) architecture to generate augmented data and thereby lower the imbalance gap by 10% as this has shown to produce good results in our experiments. The aforementioned f-point DFT is applied on these synthetic incident buckets as well.

Model Architecture

As our problem setting is similar to the Human Activity Recognition (HAR) task (see Section **??**), we build a customized Artificial Neural Network (ANN) inspired by the DeepSense architecture proposed by Yao et al. [42].

In a first step, the network input is split based on the sensor that has produced it into accelerometer, gyroscope and GPS (i.e., velocity). Simultaneously, the previously Fourier transformed accelerometer and gyroscope data are separated into their real and imaginary parts.

Then, Sensor-based Fusion (SF) is applied, a method that considers each sensor individually in order to extract sensor-specific information [16]. Furthermore, it also enables the application of different individual subnets that are varying in complexity for each sensor input. Each subnet has three convolutional layers that use 64 kernels, kernel sizes between $(3, 3, 1)$ and $(3, 3, 3)$, and a stride size of 1. While in the first convolutional layer no padding is used, the second and third convolutional layers apply zero-padding which differs from the original DeepSense framework proposed by Yao et al. [42]. Another difference is that we use 3D-convolution instead of the 2D- and 1D-convolution that were applied in the original model. 3D-CNNs are more suitable for detecting spatiotemporal features compared to 2D-CNNs [36]. The described convolutional layers are complemented by batch normalization layers to reduce internal covariate shifts [22], by Rectified Linear Unit (RELU) activation, and by Dropout layers for regularization.

Our addition of residual blocks is also a slight modification of the original framework. The reasoning behind that change is that, in some cases, deeper models might have difficulties in approximating identity mappings by multiple nonlinear layers [20]. Residual blocks have been applied with great success to overcome this issue [20].

Next, the outputs of the different subnets are merged in a convolutional fusion network. Its architecture is similar to the individual subnets containing six convolutional layers, residual

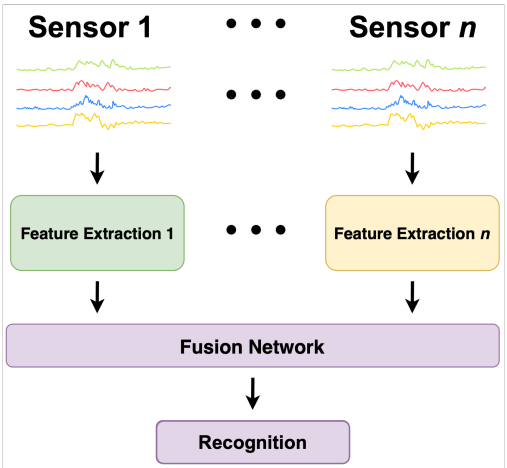


Figure 3.2: The model architecture in a nutshell: Using subnetworks for feature extraction of the different sensors (accelerometer, gyroscope, and GPS) and afterwards a fusion network to combine the features for final incident recognition [12].

blocks, batch normalization, RELU activation, and Dropout. The full process is shown in Figure 3.2.

The last big component of CycleSense is a RNN. RNN architectures such as Long Short-Term Memory (LSTM) [21] or Gated Recurrent Units (GRUs) [13] are capable of holding information the network has seen before and using it to make predictions in the current state. In doing so, it is possible to identify patterns or relationships inside the timestamps of a bucket or between buckets. Similar to Yao et al. [42], we also chose stacked GRU cells as they efficiently improve the model capacity [18].

To determine the optimal set of parameters for training CycleSense, we have conducted a grid search on a variety of hyperparameters, some of which are shown in Figure 3.3.

In the following, we use GPS, accelerometer, and gyroscope data as model input if not indicated otherwise. Linear accelerometer data was only used in a few experiments as it is not available in our iOS and older Android data sets. Our implementation of CycleSense is available on GitHub[4].

[4]https://github.com/simra-project/CycleSense

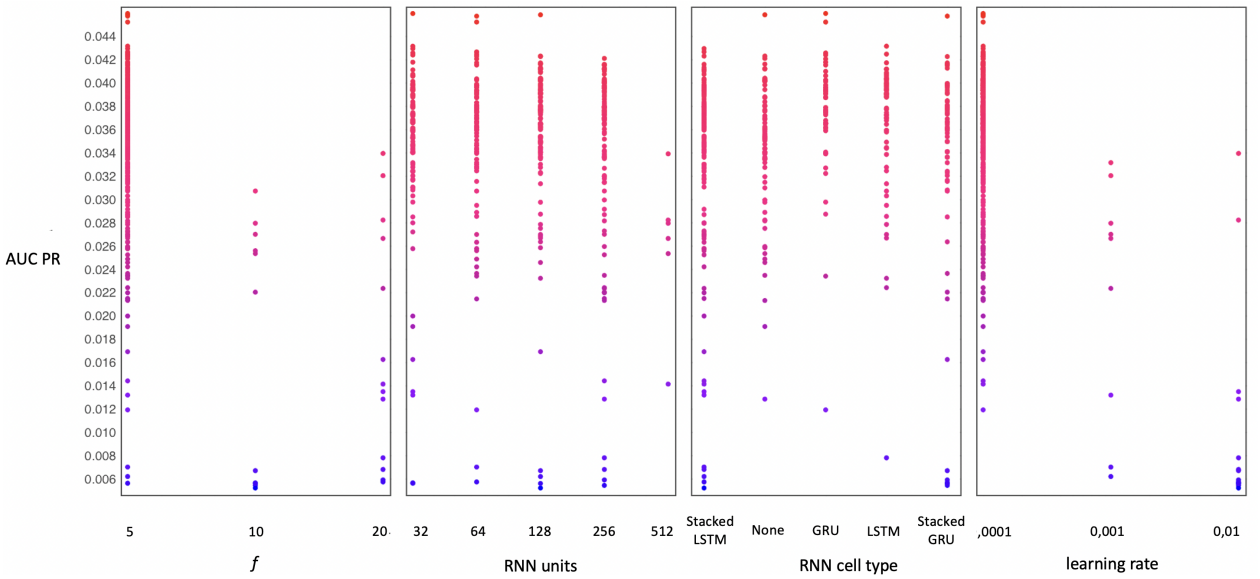


Figure 3.3: Results of the hyperparameter optimization for the four variables f (the window size), the number of RNN units used, the RNN cell type utilized, and the learning rate from left to right.

Model Training

For model training, the data set was split randomly into a training set (60%), a validation set (20%), and a test set (20%). Furthermore, the exact same splits are used for each model to improve comparability. We trained the final model for 60 epochs on an NVIDIA K80 GPU. We utilized a Binary Cross-Entropy (BCE) loss function that was updated with Adam optimization, a Stochastic Gradient Descent (SGD) method, a learning rate of 0.0001, and early stopping with a patience value of 10 epochs on the AUC ROC of the validation set. It is important to note that we did not use the complete SimRa data set. Instead, we used only a smaller subset of more recent rides recorded on Android devices since the heterogeneity of the data set across different versions and operating systems (see Section **??**) did not allow us to train a model properly on the full data set. In addition, due to limited access to hardware, we focused predominantly on data originating from the Berlin region if not stated otherwise.

As previously described, one notable challenge we were facing was the extreme label imbalance present in the data. This is due to the fact that incident buckets are far rarer than non-

incident buckets. To cope with that, we trained our model by using a weighted loss function with the class weights of the train data set as weights. For example, the class weights were 1 and 170 for the rides that have recently been recorded on Android devices.

While the DeepSense model was trained in a standard fashion, we use stacking during the training of CycleSense. Stacking (or stacked generalization) is an ensemble learning method that combines the predictions of several different models in order to contribute equally to a collective prediction.

However, we are also not interested in equal contributions of the network since that could overvalue models with a poor performance. We therefore changed the CycleSense model to an integrated stacking model by adopting the idea of stacked generalization [40], where the fusion network acts as the meta-learner. Also, we deviate from a pure stacking model. This is the case, as the meta-learner does not get any classification output of the subnetworks as input aside from the latent features in the last layer of the subnetworks. Thereby, the weights of the submodel layers that have been pretrained individually are loaded and frozen, so they are not updated during the training of the whole CycleSense model. This learning procedure further improved our results as shown in Section **??**.

3.1.2 Evaluation of CycleSense

To evaluate CycleSense' training results, we have to put them into context. For this purpose, we compare them to the two detection methods currently used in the app as discussed in Section **??**. We give an overview of the changes we made to the baseline methods with the goal of a fair comparison in Section **??**. We also describe the metrics that we use to compare our model to the baseline methods (Section **??**) before presenting the results of our evaluation (Section **??**).

Baselines

The first baseline is our original heuristic [25] which is based on the underlying assumption that incidents will often result in sudden acceleration spikes, e.g., when braking or swerving to avoid obstacles. We made some small changes to this heuristic to enable its compatibility with the AUC ROC metric, thus, increasing the comparability with our approach.

As a second baseline, we retrained the Fully Connected Network (FCN) model from the alternative approach [31]. We used the original preprocessing pipeline (which differs significantly from the here presented one) but used the full data set as introduced in Section **??**. We skipped the under-sampling step, disregarded the phone location and the bike type feature for the reasons mentioned in Section **??**, and used a non-overlapping sliding window approach with 10 second windows for better comparability.

The third baseline is DeepSense [42], which we implemented and trained as the authors describe in their work. For the differences between DeepSense and CycleSense, see sections **??** and **??**.

Based on these changes for improved comparability, we retrained the original model. We use both baselines for comparison as they are, to our knowledge, the only approaches for (semi-)automatically detecting incidents based on sensory time series data. Furthermore, they have been developed on the SimRa data set, which enables a fair comparison.

Metrics

Due to the massive label imbalance already mentioned earlier, common metrics such as accuracy, F1-score, and precision are difficult to interpret. Moreover, in our scenario it is more important to find the true near miss incidents than to classify non-incidents correctly, as False Positives can be more easily corrected by the user of the SimRa app. For both reasons, a high number of False Positives is more acceptable than a low number of True Positives, which

further limits the usefulness of such metrics like precision, F1-score, or Matthews correlation coefficient (MCC). Therefore, we focus on the AUC of the ROC metric, which is insensitive to changes in class distribution [17] while also reporting the respective confusion matrices.

Evaluation Results

In a first step, we compare CycleSense to the two baselines and common model architectures used in the context of DL for TSC [23]. All of these were trained on the Android data set consisting of more recent rides which provides the best results for all approaches. Figure 3.4 and Table 3.1 show the differences in performance.

The FCN and CycleSense clearly outperform the modified heuristic (0.621 AUC ROC). However, there is still a big performance gap between our model and the FCN model. While the FCN model scores 0.847 AUC ROC, CycleSense achieves an AUC ROC score of 0.906, i.e., there is a chance of $\approx 90.6\%$ that the model can distinguish correctly between a randomly chosen incident and non-incident bucket. Furthermore, our model performs better than other model architectures that are common for DL in TSC [23]: Auto Encoder, Gramian Angular Field (GAF), Echo State Network (ESN), and the CNN-LSTM model. With regard to the increasing model complexity, we clearly see diminishing returns. We can see this in the example of the rather simple CNN-LSTM model which exhibits a relatively close performance to the much more complex CycleSense model with stacking. The CNN-LSTM model has $\approx$ 90,000 parameters, while the CycleSense model has $\approx$ 1,100,000 parameters. As a consequence, the time to evaluate the test set of the newer Android data consisting of 795 rides took 4 seconds with the CNN-LSTM model and 56 seconds using CycleSense on the NVIDIA GPU.

In another experiment, we include the linear accelerometer sensor values in addition to the accelerometer, gyroscope and GPS data we used so far. The result for CycleSense is again an AUC ROC of 0.906, although the model requires more memory, training and processing time. Therefore, we leave out the linear accelerometer feature.

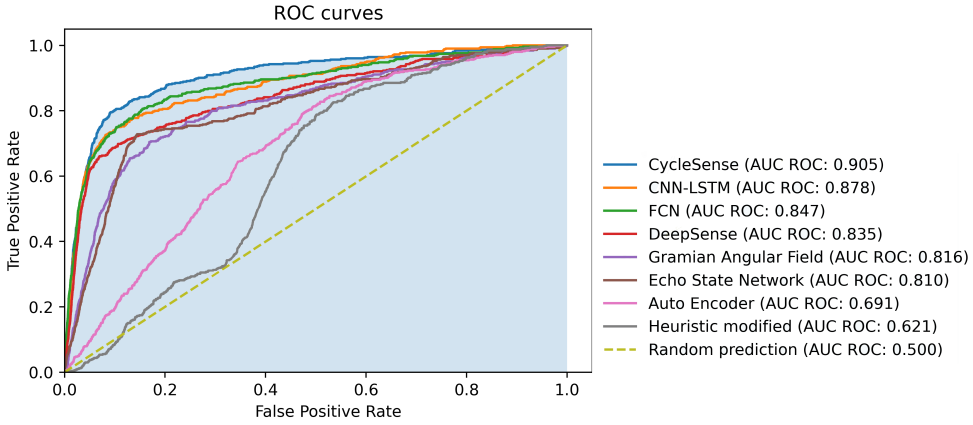


Figure 3.4: Comparison of the baselines and common model architectures used in the context of DL for TSC [23] with the CycleSense model. Note that all of these models have been trained and evaluated on rides contained in the SimRa data set that have been recorded with more recent versions of the SimRa Android app.

So far, we have predominantly focused on newer rides recorded on the Android version of the SimRa app. As shown in Figure 3.5a, the model performs far worse on the other splits of the data set. Therefore, it was necessary to train an individual model for each part of the data set. This yields far better results (Figure 3.5b).

As the Berlin region has by far recorded the most rides, we have so far used only those for tuning, training, and evaluating our model. The SimRa app, however, is deployed in many more regions, so our model is clearly required to perform there, too. For this reason, we have evaluated the Berlin CycleSense model on the newer Android rides coming from Hanover and Nuremberg.

The outcome of this experiment is visualized in Figure 3.6a. It clearly shows, that CycleSense does not perform as good as in Berlin. Since most regions lack training data, it would not be a feasible solution to train models individually per region in the current stage of the SimRa project. Instead, we retrain CycleSense on a data set that includes all the rides recorded on the newer versions of the Android app within the Berlin, Nuremberg, and Hanover regions. The results from Figure 3.6b show that this clearly improves the performance in these additional regions. At the same time, the performance on the Berlin data set has

	TN	FP	FN	TP	AUC ROC	Precision	Recall	F1-Score	MCC
CycleSense	107934	10893	104	400	0.906	0.035	0.794	0.068	0.156
CNN-LSTM	106427	12400	127	377	0.878	0.030	0.748	0.057	0.135
FCN	100932	18500	98	402	0.847	0.021	0.804	0.041	0.115
DeepSense	106192	12635	153	351	0.835	0.027	0.696	0.052	0.123
GAF	98782	20045	150	354	0.816	0.017	0.702	0.034	0.092
ESN	101670	17157	138	366	0.810	0.021	0.726	0.041	0.107
Auto Encoder	58416	60411	88	416	0.691	0.007	0.825	0.014	0.041

Table 3.1: Comparison of the baselines and common model architectures used in the context of DL for TSC [23] with the CycleSense model. See the discussion in Section **??** about the usefulness of various metrics. Note also that all of these models have been trained and evaluated on rides contained in the SimRa data set that have been recorded with more recent versions of the SimRa Android app. For all models the threshold which optimizes Youden's index[43] were chosen.

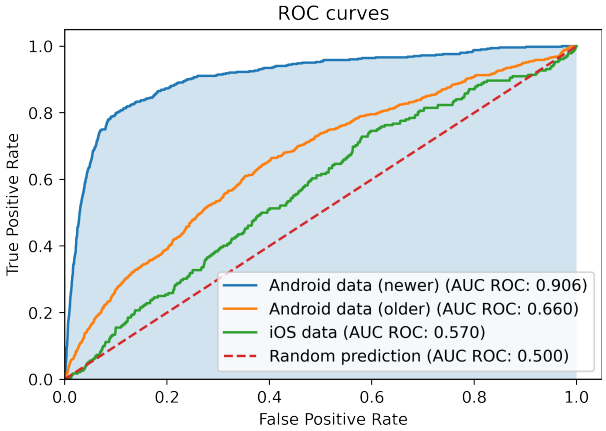
declined only slightly (0.029 AUC ROC) by comparison. Nevertheless, the AUC ROC for Berlin is still the highest and clearly above Nuremberg, which is well ahead of Hanover.

3.1.3 Discussion of CycleSense

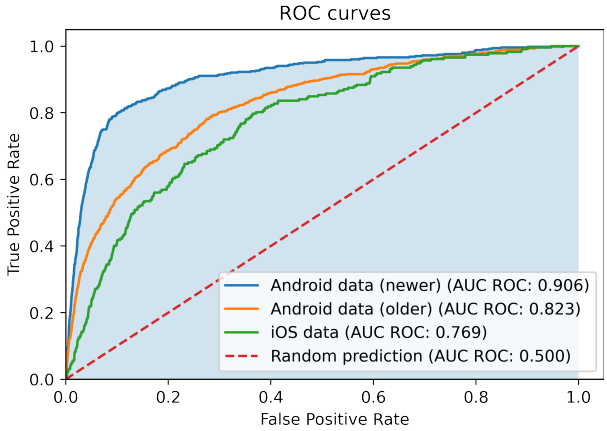
Our results demonstrate that the proposed CycleSense model outperforms every other model that we have compared to. While improving the model's architecture and other components, we identified a number of challenges inherent to the problem setting that we discuss in the following.

Impact of Preprocessing & Training Steps

In a first step, we want to highlight the importance of our preprocessing and training methods for the success of our model. Therefore, Figure 3.7 illustrates the performance of CycleSense when one of the preprocessing or training methods is skipped, resulting in a significant drop in performance in each of the four examples.



(a) CycleSense trained only on the newer Android rides and evaluated on all parts of the data set.

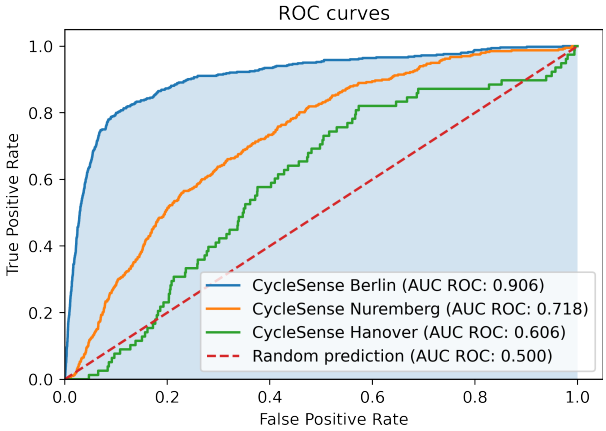


(b) CycleSense trained and evaluated individually on all parts of the data set.

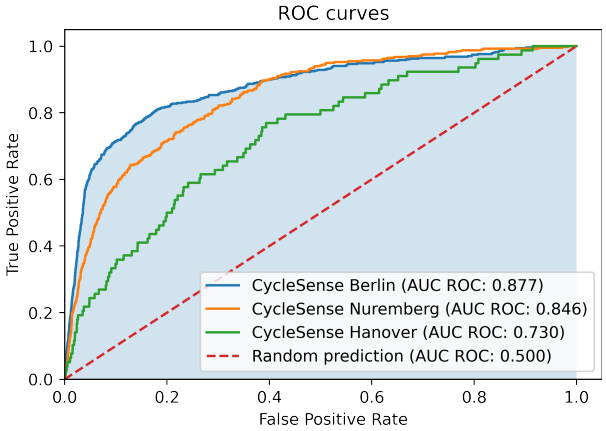
Figure 3.5: Comparison of CycleSense trained only on the newer Android rides or individual CycleSense models trained for each part of the data set. Note that only the newer Android data set was manually cleaned.

Limitations of Crowdsourced Data

Crowdsourced data are generally noisy which could contribute to a reduced model performance. They do also contain several biases, e.g., the Selection Bias, Device Positioning Bias, Extreme Aversion Bias, or Confirmation Bias [4, 11, 24]. In this context, the heterogeneity across devices and platforms is likely another factor of influence. As already described, the application that is used by contributors to record their rides is available on two platforms. Those platforms are supported by a wide range of different devices and models with different hardware inside. Phone manufacturers use different GPS, gyroscope, and accelerometer sensors that can vary immensely in sensitivity and overall behavior. Stisen et al. [34] have shown that there is major heterogeneity when it comes to the accuracy of sensor readings. While the main focus of that study was on accelerometer data, Kuhlmann et al. [26] have shown that orientation sensor data is also affected by this variability. This is in contrast to the HAR tasks, we compare our task to. The HAR data set [3] was generated under laboratory conditions that always used the same smartphone type body mounted to the exact same position on selected study participants. This significantly simplifies the classification



(a) Performance of CycleSense trained on the new Android data set of rides recorded in Berlin.



(b) Performance of CycleSense trained on the new Android data set of rides recorded in Berlin, Nuremberg, and Hanover.

Figure 3.6: Comparison of the performance of CycleSense trained on Berlin and trained on the other regions combined.

task. Another factor that contributes to the issue of noisy crowdsourcing data in the SimRa data set is the fact that some users misinterpret the purpose of the SimRa app. Instead of labeling incidents, they report, e.g., dangerous areas or annoying traffic lights. These can of course not be captured by the sensors used. Many such "incidents" can be identified through the comment column of the data set. It should also be noted that the SimRa data set was not recorded and labeled with the goal of developing ML models – the goal was to record data which will be analyzed and processed by humans.

Technical Limitations of the SimRa Data Set

Recording rates of sensor readings deviate among devices and operating systems and impair model predictions [34], this may also affect the performance of CycleSense: As illustrated in Figure 3.8, the recording rates differ significantly in older and newer Android rides as well as in iOS rides. While the iOS rides' median ($\approx$ 300 ms) is similar to the one of newer Android rides, the IQR is much greater and spans approximately 150 ms. This circumstance could indicate that the relatively weak model performance on this data set can partly be explained

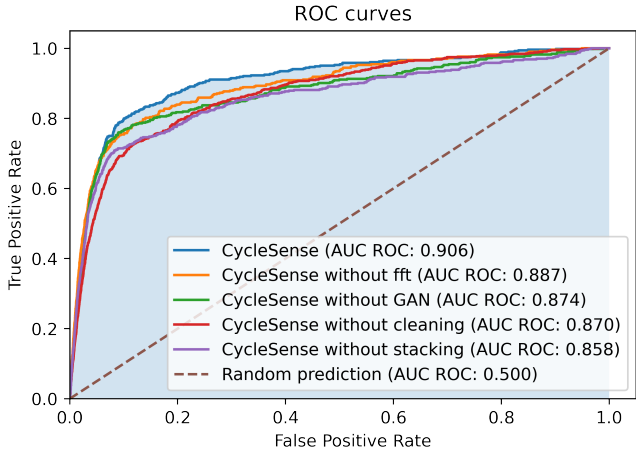


Figure 3.7: Impact of different preprocessing and training steps on the performance of CycleSense.

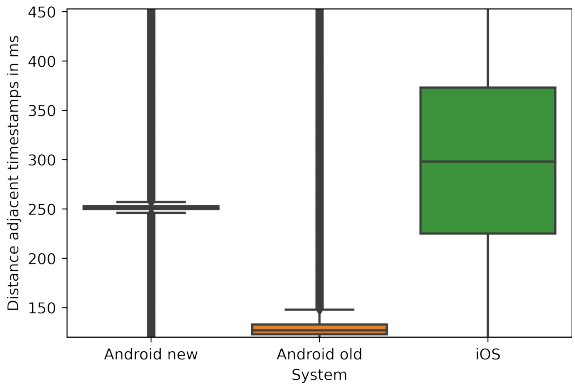


Figure 3.8: Box plot showing the empirical measurement times observed in the SimRa data set for rides recorded with different versions of the SimRa app.

by that factor. Furthermore, the gyroscope data is recorded with a higher frequency in newer Android rides than in older Android or iOS rides. This factor could also contribute to the different model evaluation results. The achieved AUC ROC for uncleaned newer Android data was 0.870, while it was 0.823 for the older Android data.

Aside from that, the moving average that is used in the SimRa app to condense the data and comply to users' upload volume constraints [25] reduces the amplitude and shifts the exact point in time of incidents and other events. This has the effect that incidents and non-incidents are hard to distinguish as illustrated in Figure 3.9. Both these factors could hurt the model's ability to classify incidents correctly based on the data. It is important to

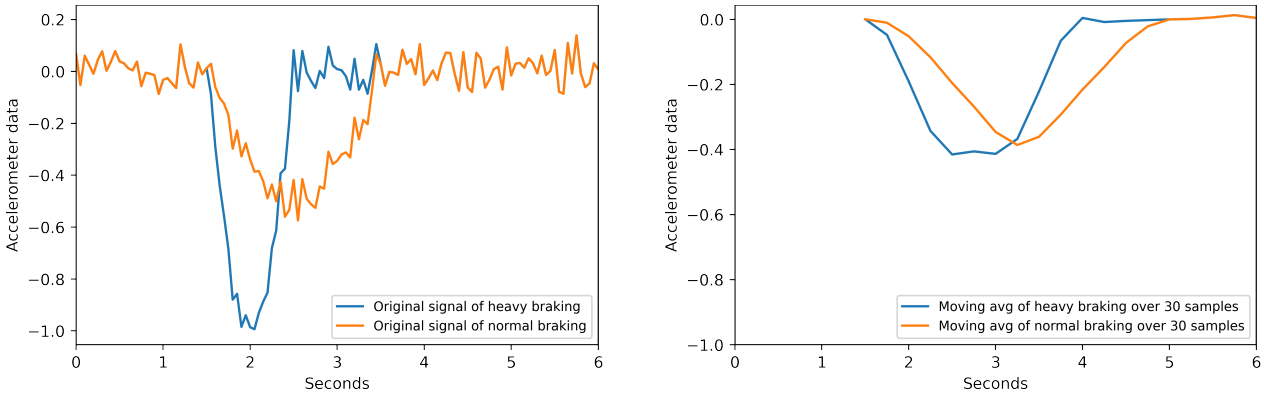


Figure 3.9: Visualization of the sensor data of a simulated heavy braking incident vs. a moderate braking event before and after the moving average has been applied.

acknowledge that this issue becomes less severe when the sampling frequency is higher, as it is the case for the older Android data.

Limitation of the Classification Task

There is an inherent label imbalance present in our data set. Only relatively few rides contain incidents. Moreover, the ratio between timestamps that represent an incident and those that do not is even smaller. As a consequence, there is an enormous imbalance between the different label categories.

Furthermore, some incident types such as tailgating or close passes might not be detectable at all in accelerometer and gyroscope sensors [2, 25] as cyclists might not change their motion profile despite (or even due to) a dangerous situation. For detecting these kinds of incidents it may be necessary to include different sensors. In the SimRa project, this is already happening through an integration of OpenBikeSensor (OBS)[5] data which measures the passing distance of cars. The current data set, however, only includes very few OBS-supported rides.

In contrast to our classification task, the inherent classification problem of the HAR data set [3], is to classify reoccurring ongoing patterns such as walking or jogging, while an incident

[5]https://www.openbikesensor.org/

might need to be detected by one short non-reoccurring event such as sudden braking. This further complicates the matter.

Data Set Shift

A major challenge in the context of crowdsourced data are data set shifts. Depending on the device (see also **??**), but also on other factors, such as the city of origin of the recorded data (see also **??**), there could be non-stationarities in the data that violate the assumptions underlying almost all ML models and thus could impact the generalization performance of the trained models. Several types of data set shifts can be distinguished, including *covariate shifts*, meaning shifts in the input data [35], *label shifts*, meaning shifts in the distribution of targets [27], or mixtures thereof. Detecting these shifts [1, 5, 10, 29, 30] and predicting [32] or reducing their impact on the generalization performance is an active field of research [6, 33]. Some approaches aim at model specific improvements to alleviate data set shift [35]. These approaches have a decisive disadvantage, most of these approaches only work for one model class and require access to the inner workings of the ML pipeline, often after the feature extraction step. More promising and easier to build and maintain are model agnostic solutions that focus on the data, rather than the models, to detect and counteract data set shifts [6]. Extending the training data sets to account for all variation and shifts in the data that the ML model should be invariant to, often called *augmentation*, is a popular and effective way of counteracting data set shift, see for instance [14]. In our work we follow this line of thought of a data centric AI approach.

3.1.4 Summary of CycleSense

An increased modal share of bicycles is necessary for solving emission and traffic related urban problems. A key challenge for this, is the lack of (perceived) safety for cyclists. Improving the situation requires detailed insights into safety levels of street segments – the SimRa

platform [25] has been proposed as a data gathering mechanism for incidents and cycling tracks. While this is an important step towards data collection, the platform relies on manual annotation of tracks which limits the number of potential users.

In this paper, we have proposed CycleSense – a model for automatic detection of such incidents. Using the SimRa data set, we have shown that CycleSense is capable of detecting incidents on the basis of accelerometer and gyroscope time series data in a real-world scenario. It can correctly distinguish between an incident and a non-incident with a probability of up to 90.5%. We have also compared it to the heuristic currently used in the SimRa platform and an existing FCN that was specifically developed for SimRa. Additionally, we have implemented several DL models that are frequently used in the context of TSC. We were able to show that our model outperforms all of these approaches.

While this is an important step towards fully automatic incident detection, we believe that – in the context of the SimRa platform – CycleSense should be complemented with human annotation in a semi-automated way for the foreseeable future. Although this does not quite reach our long-term goal of full automation, it should significantly decrease the annotation effort and should also lead to improved labeling quality. In the future, this could, in turn, be used to further improve CycleSense. Since April 2022, the SimRa app has been using CycleSense for incident detection.

3.2 CycleQuality

Cities all over the world aim to increase the modal share of bicycle traffic, e.g., to address emission problems, frequent traffic jams, but also to improve the citizens' health through more daily activity. Key factors for this are traffic safety, topography but also the comfort level of cyclists [**Nyberg1996Road**, **gadsby2022understanding**, 25]. All these factors are affected by the availability and quality of cycling infrastructure which needs to be monitored and maintained continuously. Even monitoring of surface quality alone, however, can be

challenging due to the sheer dimensions of existing infrastructure. Germany's capital Berlin, for example, has around 2,300 km of bicycle tracks as of March 2023, i.e., manually inspecting cycling infrastructure to monitor surface quality is infeasible.

To address this problem, we propose to use a crowdsensing approach in which cyclists record their daily rides using a smartphone app. Using the built-in motion sensors, the surface quality (or rather the lack of) can be measured as vibrations or bumps. In a second step, we can then combine data from multiple rides to derive an estimate for the surface quality. This way, monitoring of surface quality can be automated to a high degree at little cost and the resulting data can be used for maintenance planning or surface quality-aware routing. Especially for highly frequented cycling tracks, surface quality problems can be detected quickly.

While there are other projects studying the surface quality of cycling infrastructure through crowdsourcing (e.g., Luedemann et al. [**luedemann2022bikevibes**]), our work is unique and novel because alternative approaches (i) either focus on categorizing the surface *type* (e.g., cobblestones) where we focus on the surface *quality* (i.e., the level of roughness) or (ii) have strict assumptions on phone positioning and other properties where we rely on a "wisdom of the crowds" strategy to filter out noise. The latter is also an advantage because, unlike related approaches, which use a dedicated app for measuring the road surface quality, thus, potentially forcing cyclists to run multiple apps, our approach can easily be retrofitted to existing apps and can even be used to analyze already stored datasets.

For this, we make the following contributions:

- We present a data processing pipeline for deriving surface quality which combines signal processing with geographical clustering techniques in the form of edge-based preprocessing on the phone and cloud analytics (**??**).
- We describe how we integrate this data processing pipeline in the existing crowdsensing project SimRa [25] which up to now has only focused on traffic safety(**??**).

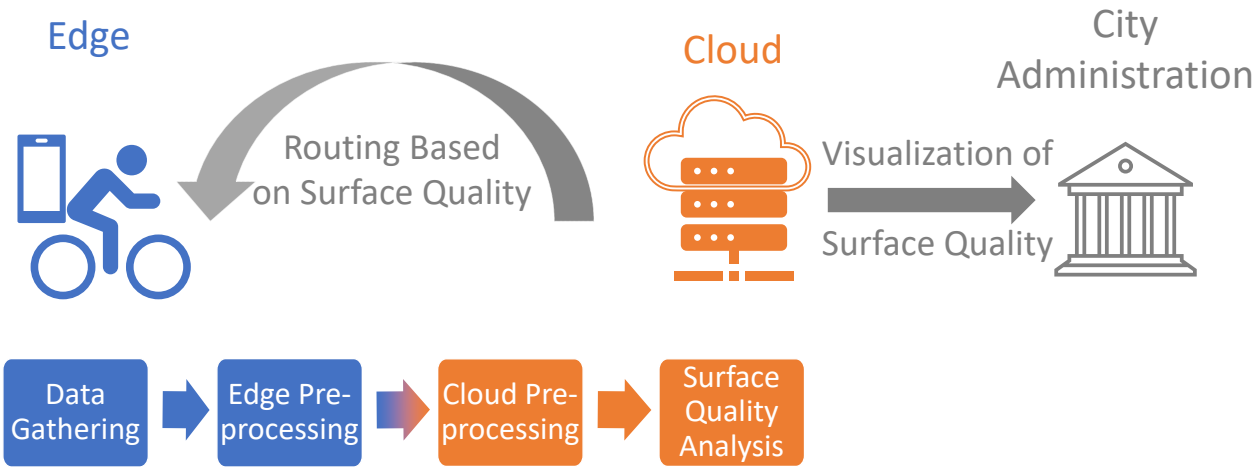


Figure 3.10: Overview of our surface quality analysis pipeline.

- We describe how the resulting data can be used to increase the comfort of cyclists through surface quality-aware routing and how the data can be exposed to city administrations(**??**).
- We evaluate our approach by analyzing the SimRa dataset [**dataset˙simra˙set1**, **dataset˙simra˙se dataset˙simra˙set3**] and comparing it on-site conditions in eight streets with different surface quality (**??**).
- We discuss to which degree our approach can automate surface quality monitoring and how additional sensors could possibly help to improve data quality (**??**).

3.2.1 Data Processing Pipeline

In this section, we start by giving a high-level overview of our data processing pipeline for deriving surface quality **??** before describing the individual steps in the process (**??** to **??**).

Overview of Pipeline

The tasks of our data processing pipelines are distributed over the edge (i.e., on the smartphone) and the cloud, see also Figure 3.10. After collecting data in the SimRa app using the

built-in motion sensors, data is preprocessed locally before upload to our backend servers. There, additional preprocessing steps are executed before the actual surface quality analysis.

The parts of the preprocessing which reduce the amount of data need to be run on the edge to preserve user privacy and to reduce bandwidth consumption. The data cleaning parts are too compute-intensive and are executed in the cloud to reduce power consumption on the phone.

Preprocessing in the Cloud

The preprocessing in the cloud has two main goals: cleaning noisy data and calculating a value that represents the smoothness of the ride. For the first goal, we cut off ten seconds each at the start and end of the ride to remove noise resulting from users (un)mounting their bikes or putting the phone in a pocket. Ideally, this would of course already be done on the phone but could not be done here since the main use case of SimRa requires this data, i.e., we would have had to upload the data twice. We also remove stops, i.e., only the parts of the ride which have a minimum speed of 5 km/h and a minimum duration of 1 minute are considered. Such low speeds usually occur, when the bicycle is being pushed by the cyclist rather than being ridden or when stopping at traffic lights. Even if the cyclist should manage to ride so slowly, the ride will be prone to unsteady motions, leading to noisy data. This also removes parts where users forgot to stop the recording after their ride.

For the second goal, we normalize the accelerometer data by calculating the mean of the three moving variances (window size 10) of the axes X, Y and Z. This preserves information about amplitude but disregards the direction which is highly affected by the position of the phone. Thus, we obtain a time series of single values representing the bumpiness of the surface in each in-motion ride segment.

Surface Quality Analysis

Each cyclist creates a different data track on a given road, and without calibration, it is not possible to accurately analyze each track individually. Ideally, we would at least use cyclist-specific profiles (i.e., per cyclist aggregates) which, however, are not available due to privacy reasons (rides are pseudonymized individually [25]). The idea behind our approach is to take advantage of the size of the data set and use the law of large numbers to obtain robust results without having to calibrate the data and without affecting the results too much by noise.

For this, we consider in the first step each ride individually: We take the preprocessed ride (which is a single time series of aggregated motion sensor readings without stops plus the GPS trace) and calculate the percentiles (0.2, 0.4, 0.6, 0.8, and 1) of the time series. For normalization, we then replace all motion sensor readings with values 1 to 5 depending on which interval they fall into, i.e., a motion sensor reading from the interval $(0.2; 0.4]$ would be replaced with the value 2. The intuition behind this is that longer rides are likely to encounter very different surface quality, hence we calculate the relative bumpiness of an area in comparison to the rest of the ride's bumpiness.

In the second step, we use the data from all such normalized rides and, using the GPS trace, map them to a grid of 10m^2 cells. For each cell, we hence have a distribution of values 1 to 5. As a metric for the bumpiness of that cell, we use the average of all values – e.g., when color-coding a map – but make other statistical metrics available as well (see, e.g., the distribution function charts in our evaluation section).

3.2.2 Using Road Surface Quality Information

In this section, we describe how such surface quality results can be used. In **??**, we describe how we implement a navigation feature into the SimRa app that uses the road surface quality

as an additional parameter in routing. We then show in **??** how we can expose the output data of our pipeline to city administrators.

Routing with Surface Quality

With the surface quality scores calculated, it is possible to provide a route planning feature, where not only distance and time are considered, but also the surface quality. The main question here is how much the surface quality should be weighed when calculating the best route from A to B or rather what detour lengths are acceptable. We decided to give the user the opportunity to influence this factor with the usage of a slider, that can be set between 0 for not considering surface quality in the routing and 10 for the highest importance of the surface quality (fig. 3.11). We host a modified GraphHopper[6] for routing. GraphHopper uses edges for streets, that are connected via nodes. Each has a weight for routing purposes and it is possible to change the weight according to custom data. This is where we use the surface quality by increasing the weight depending on surface quality and user-specified influence factor.

Output Data and Visualization

Our bicycle road surface quality pipeline creates a GeoJSON file as an output. It contains the cells with a surface area of $10m^2$ in a grid as `Features` of the `geometry` type `Polygon` with the surface quality score information such as mean, median, standard deviation, number of rides going through the cell and coloring information in the `properties` key. With such an output file, it is very easy to create a simple visualization[7], e.g., with Leaflet[8], as depicted in fig. 3.12. Using a visualization like this, also non-tech-savvy users can easily monitor the bicycle road surface quality of a large area and take action where needed.

[6]https://github.com/graphhopper/
[7]https://simra-project.github.io/surfaceQuality/Berlin.html
[8]https://leafletjs.com/

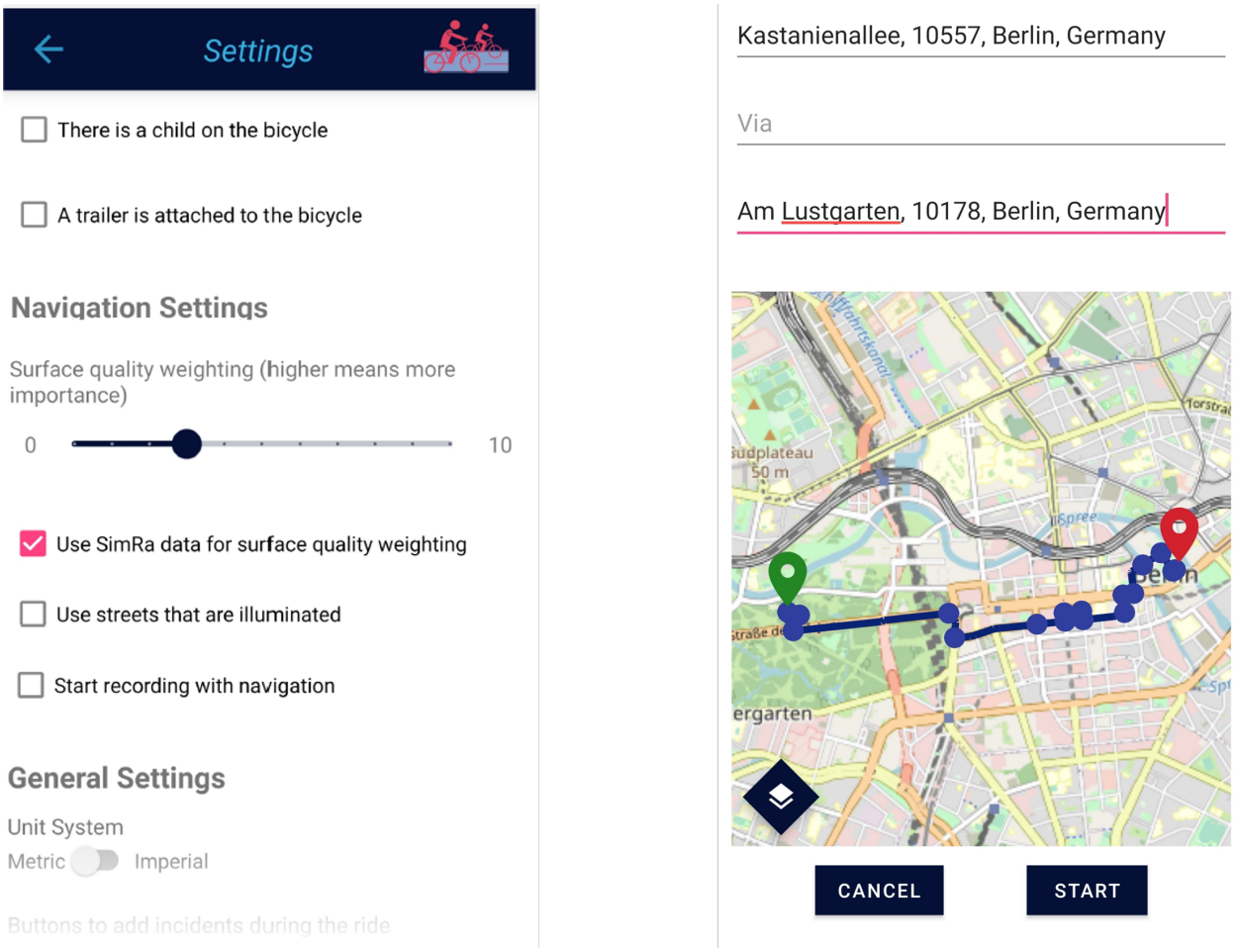


Figure 3.11: With a slider in the settings menu, the weight of the surface quality in the routing can be set.

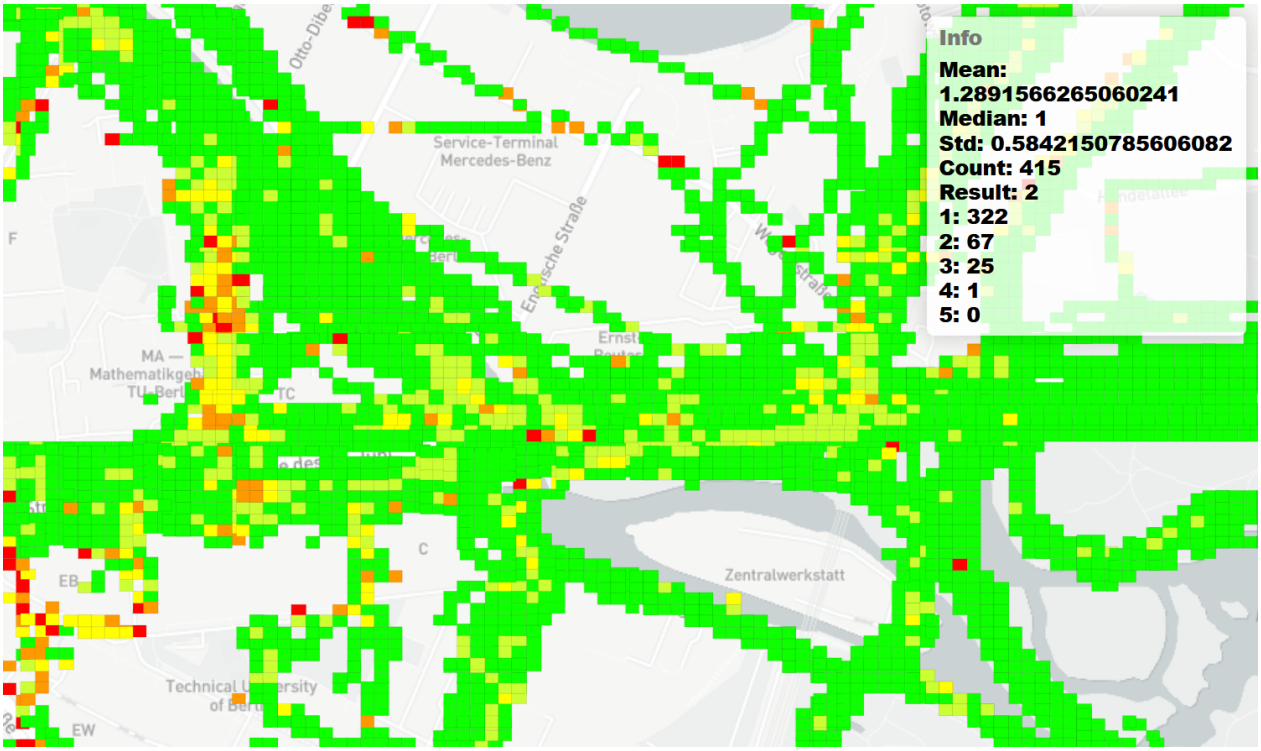


Figure 3.12: A visualization of the output file showing the surface quality of the boxes when hovering over them with the mouse cursor. Color coding is based on the average value.

3.2.3 Evaluation of CycleQuality

In this section, we evaluate the surface quality analysis by comparing the calculated surface quality of selected street segments across Berlin with their surface type in the real world, which we get from OpenStreetMap (OSM). As data input, we use the existing SimRa datasets [**dataset˙simra˙set1**, **dataset˙simra˙set2**, **dataset˙simra˙set3**] (almost 90,000 rides with more than 650,000km in total). Based on the intuition that different surface types will also be partially correlated with surface quality, we randomly picked four spots with different surface types. To also show the limitations of our approach, we then explored the dataset and manually picked four additional spots where our approach appears to have returned the wrong results. Each evaluated section has a surface area of 10 m^2 and to compare them to each other we analyze their mean, median, and standard deviation values.

We first describe the clear results from the first group (**??**). Afterwards, we categorize the additional four "problem spots" as mixed results (**??**) or seemingly incorrect results (**??**).

Sections with Clear Results

We evaluate at least one example for each of the following surface types, which are sorted in descending order with regard to their expected surface quality [**titov2019monitoring**]: asphalt, flat paving stones, fine gravel, cobblestones. We chose the sections in a way that (i) asserted that we have sufficient data for them and (ii) to cover all different surface types. After filtering based on these criteria, we randomly picked four sections.

table 3.2 and fig. 3.13 show the results of the surface quality analysis of the selected sections with very clear and intuitive results.

It can be observed, that the aforementioned list of surface types, which was sorted in descending surface quality order, was ordered correctly. A newly maintained asphalt section in *Straße des 17. Juni* has a nearly perfect score, which means that its surface quality was in

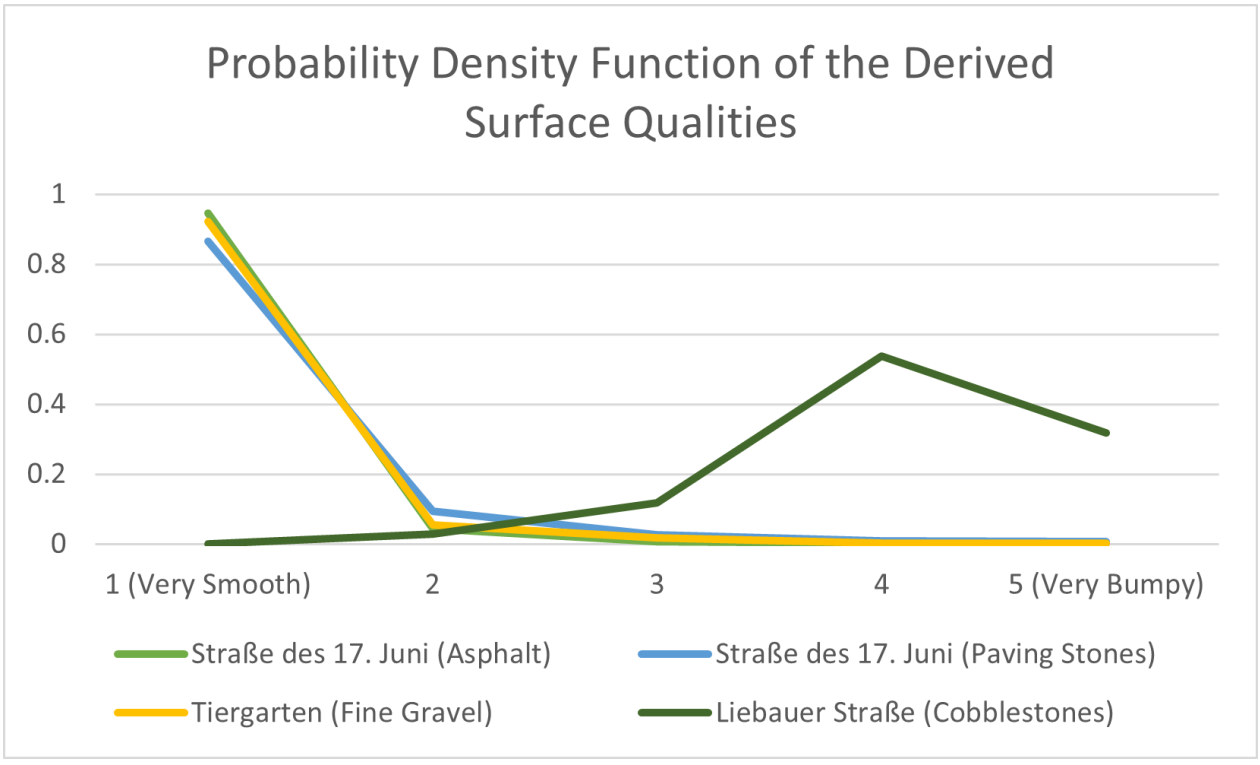


Figure 3.13: The Probability Density Function of the Derived Surface Qualities shows that these segments have very undisputed road surface quality values, since they are either very good or very bad.

Table 3.2: Surface Quality Analysis Evaluation Results Showing Mean, Median and Standard Deviation of Sections With Clear Results

Street Name	Surface	GPS Location	Mean	Median	Std. Dev.
Straße des 17. Juni	Asphalt	52.515369,13.630855	1.03	1	0.17
Straße des 17. Juni	Paving Stones	52.513501,13.335127	1.25	1	0.5
Tiergarten	Fine Gravel	52.514745,13.34622	1.32	1	1.06
Liebauer Straße	Cobblestones	52.509132,13.453806	4.15	4	1.07

the top 20% in almost all rides crossing this section. Followed by that are two sections with flat paving stones and fine gravel as their surface type, which have very similar results. This means, that both flat paving stones and fine gravel have comparable surface quality in terms of bumpiness from the perspective of a cyclist. However, it should be noted, that fine gravel can be less favorable in areas with a lot of precipitation (more on that in **??**). Not very surprisingly, the *Liebauer Straße* has very bad surface quality scores, since it is paved with cobblestones and has very busy sidewalks with restaurants and cafes, which prevent cyclists from (illegaly) cycling there instead of on the street.

Sections with Mixed Results

While most results are intuitive, e.g., new bicycle roads paved with asphalt and separated from the motorized vehicle traffic having very good surface conditions, some other sections (see fig. 3.14) seem confusing at the first glance.

According to the Probability Density Functions (PDFs) of street sections in the *Invalidenstraße* and *Kaiserin-Augusta-Allee*, there seem to be two distinct surface qualities in each section. A closer look into the specific sections reveal the causes:

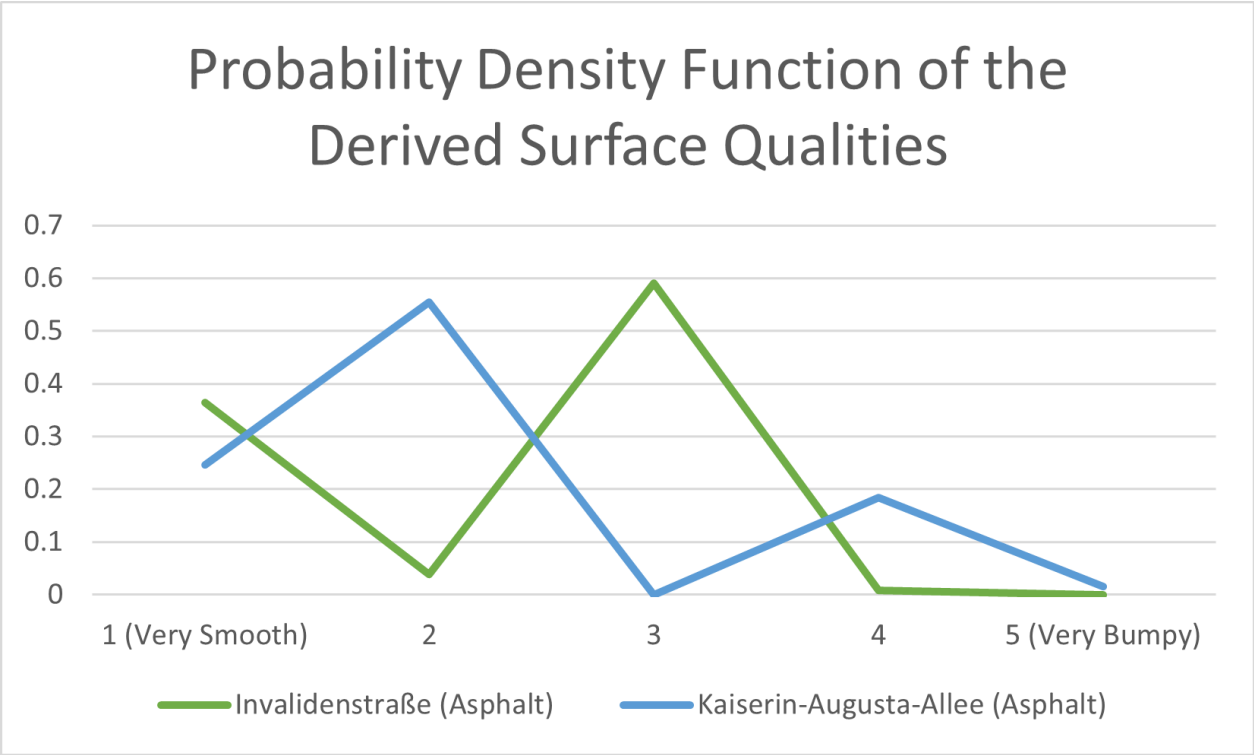


Figure 3.14: The Probability Density Function of the Derived Surface Qualities shows that these segments have confusing road surface quality values: Depending on the ride, they appear to have either very good or very bad surface quality (multiple peaks).

Table 3.3: Surface Quality Analysis Evaluation Results Showing Mean, Median and Standard Deviation of Sections With Mixed Results

Street Name	Surface	GPS Location	Mean	Median	Std. Dev.
Invalidenstraße	Asphalt	52.526236,13.369196	2.24	3	0.96
Kaiserin-Augusta-Allee	Asphalt	52.524429,13.327253	2.17	2	1.05

Figure 3.15: A bus lane, a bus station and a bicycle lane on the sidewalk can create different results in a small area. (Source: Apple Maps)

In *Invalidenstraße*, there are a bus lane, a bus stop, and an on-curb bike lane (see fig. 3.15). Most bus lanes in Berlin can be legally used by cyclists, however, some cyclist may still prefer the relatively bumpy bike lane. Additionally, when a bus stops at the bus station, cyclists that used the bus lane might overtake the bus on the very smooth car lane to the left. These two factors most certainly are the reason for the distinct two-peak distribution of recordings in that spot.

A closer look at the near-miss incident data in *Kaiserin-Augusta-Allee* indicates that people use the bike lane in one direction and the street in the other direction [**dataset˙simra˙set1**, **dataset˙simra˙set2**, **dataset˙simra˙set3**] as one side seems to be in an unusable condition while the other is acceptable. Since the bicycle lane is paved with paving stones and the street with asphalt, this leads to different surface quality of the road, depending on which direction the ride was. Both directions, however, regularly end up in the same 10x10m box as the street is not overly wide, especially considering GPS accuracy.

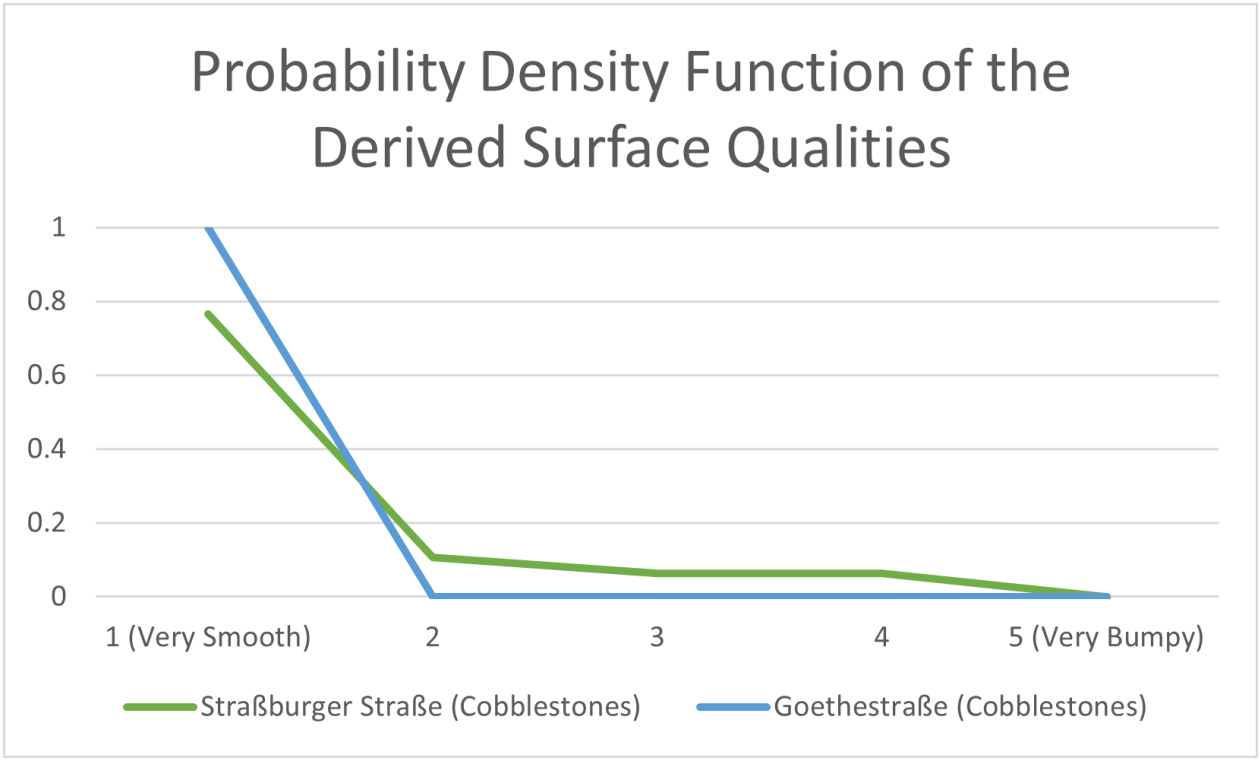


Figure 3.16: The Probability Density Function of the Derived Surface Qualities shows that these segments have very smooth road surfaces, although they are paved with cobblestones.

Sections with (Seemingly) Confusing Results

There are also sections where the surface quality result seem to be completely wrong, when compared to the OSM surface type labels. Table 3.4 shows two sections, that are – according to OSM – paved with cobblestones and without separate bike lanes, but with very good results, which is confirmed by the PDFs depicted in fig. 3.16.

The possibility, that this section has a wrong label and is in fact, not paved with cobblestones may come to mind. However, as fig. 3.17 reveals, both streets are indeed paved with cobblestones.

Figure 3.17 also shows very likely reasons why these sections have very smooth bicycle rides. The infrastructural conditions incentivize cyclists to prefer the sidewalks over the actual streets. First, the cobblestones on the streets make it very unpleasant for cyclists to cycle

Table 3.4: Surface Quality Analysis Evaluation Results Showing Mean, Median and Standard Deviation of Sections With (Seemingly) Confusing Results

Street Name	Surface	GPS Location	Mean	Median	Std. Dev.
Straßburger Straße	Cobblestone	52.532273,13.416521	1.43	1	0.87
Goethestraße	Cobblestone	52.508889,13.308333	1	1	0

on them. Second, the sidewalks are wide, quiet, and paved with large flat-surfaced paving stones. Third, the streets are contested by parking cars, or by cars searching for a parking spot, service vehicles, construction vehicles, etc. which will often block cyclists and also create safety hazards. Considering these circumstances, it is not surprising to see good surface quality values in these two and other similar streets as cyclists are likely to (illegaly) use the sidewalks instead. In fact, photos on Apple Maps (not included in this paper) actually show cyclists using the sidewalk instead.

3.2.4 Related Work of CycleQuality

Surface quality is usually quantified based on the International Roughness Index (IRI), e.g., [**sayers1986The**]. Traditionally, this was done using so-called profilographs (approximately resembling a large ladder with wheels) which are towed by a car or pushed manually. A study from the 1980s [**cumbaa1986road**] found that they often break down and are difficult to maneuver in narrow streets which renders them infeasible for measuring the quality of bike lanes. Furthermore, such measurement are very personnel-intensive which makes it unrealistic to apply them on a broad scale.

Taniguchi et al. [**taniguchi2015evaluation**] detect road hazards such as debris, potholes, or bumps. For this, they attach an ultrasonic distance sensor to a bike handlebar, scanning the ground in front of the bicycle. While the approach can warn cyclists about incoming

bumps on the road ahead in real-time, this approach does not scale for city-wide analysis due to the sheer number of sensors needed.

Peng et al. [**peng2019road**] follow the same goal, using motion sensors instead of distance measurements. Data is analyzed offline using a classifier which can identify asphalt, pebbles, and very bumpy underground. In contrast to our work, their emphasis is on identifying specific surface *types* rather than the surface *quality*. Also, their approach again requires dedicated hardware.

Zhou et al. [**zhou2022smartphone**] try to detect manhole covers. For this, they analyze a video stream from a bike handlebar-mounted smartphone camera using a convolutional neural network. While their approach is similar to ours regarding hardware, constantly recording video means high power consumption. Furthermore, the phone needs to be mounted in an awkward angle which makes it impractical for every day use.

Luedemann et al. [**luedemann2022bikevibes**] use a smartphone app to record accelerometer data as an indicator for surface quality. Their approach, however, relies on single rides and has no concept of aggregating data from multiple rides.

Similar to us, Yamaguchi et al. [**yamaguchi2015simple**] want to measure the surface quality. To achieve highly precise results, they combine the smartphone motion sensors with a cyclometer. This approach will always give more precise results than our approach but again requires dedicated hardware which renders a city-wide usage infeasible.

Beyond these, there are several car-centric approaches which either use dedicated hardware, e.g., [**paper˙chen˙crsm**], have very specific phone placement requirements, e.g., [**paper˙daraghmi˙surface˙c**
or focus on detecting the transients, i.e., individual pot holes, e.g., [**paper˙alam˙surface˙transients**, **paper˙li˙surface˙transients**, **paper˙lima˙surface˙transients**].

3.2.5 Discussion of CycleQuality

Overall, the results presented in this paper show, that our approach can derive surface quality information using data recorded in the SimRa app. Nevertheless, it still has a number of limitations which we discuss in this section.

Methodological Challenges

Since the SimRa dataset consists of crowdsourced data generated by smartphones, the recorded data is very heterogeneous. This comes from the fact that different smartphone models have different GPS modules and motions sensors, which leads to the problem that a road can be very smooth according to one smartphone and very rough according to another. This problem is further aggravated by the wide range of different bicycle types, e.g., racing bicycles or mountain bikes, which in turn also heavily influence the vibrations the smartphone can sense. To solve these problems, we rely on the *law of large numbers* and compare how a section's surface quality was relative to the ride.

Preprocessing in SimRa

As input for our surface quality analysis pipeline, we used the SimRa dataset, since it is to our knowledge the only public dataset containing a very high number of anonymized individual rides in a non-aggregated form. However, it is important to note, that SimRa was developed for identifying near miss incident hotspots in bicycle traffic. For that, a relatively low accelerometer sampling rate of 50 Hz is used before further reducing the level of detail by calculating the moving average with a window size of 30 and then taking every fifth value. While with this, it is possible to automatically detect near-miss incidents [**karakaya2022cyclesense**], a higher resolution of data points would allow us to develop a more sophisticated measurement approach. This would, however, further increase

the disk space, memory, battery and bandwidth usage of the SimRa app and as we have shown in **??**, the surface quality we derive are sufficiently precise for our intended use cases.

Behavior of Cyclists

Due to the GPS inaccuracy, it is impossible to identify whether a cyclist used the actual road or the sidewalk. For instance, as discussed in the previous section, fig. 3.17 showing *Straßburger Straße* in Berlin, would be expected to have poor results but in fact has surprisingly good values. We believe that this is due to cyclists illegaly using the smooth sidewalk instead of the bumpy road.

Furthermore, cyclists will in practice also avoid the worst potholes and similar bumps if possible, i.e., these will usually be missing from our analysis.

Sensor Inaccuracy

The inaccuracy of the GPS sensors [**merry2019smartphone**] combined with noise produced by the motion sensors of the smartphones form another limitation of the dataset. We tried to partially address these limitations with our preprocessing steps but they can, of course, not be fully mitigated. The only alternative would be using dedicated hardware which, however, will result in significantly less recorded rides due to the adoption barrier.

Temporal Influences

The rides in the SimRa dataset date back up to 2019 and it is possible that the surface quality changed throughout the time. One reason for that could be that the surface type is changed for example from cobblestones to asphalt or potholes and cracks are repaired. This would presumably lead to better results after the change, but show up as two-peak distribution in

our dataset. When using this approach in practice, we hence propose to only consider the most recent rides.

Another temporal influence might be the weather. On rainy, windy or snowy days, the surface quality of the road, especially with wet gravel, can suffer significantly, which is not considered by our surface quality analysis approach. However, adding this factor to the analysis, would in our dataset reduce the validity of the results, since it would reduce the number of considered rides.

3.2.6 Summary of CycleQuality

Cities all over the world aim to increase the modal share of bicycle traffic, e.g., to address emission problems, frequent traffic jams, but also to improve the citizens' health through more daily activity. Aside from safety, a key influence factor for this is comfort, particularly in the form of the surface quality of cycling infrastructure. Monitoring the surface quality manually, however, is infeasible due to the dimensions of such infrastructure.

In this paper, we proposed a crowdsensing approach in which cyclists record their daily rides using a smartphone app and the phone's built-in motion sensors. We proposed a data processing pipeline that starts on the edge (i.e., the phone) and ends in a cloud backend. Furthermore, we showed that our crowdsensing approach can indeed derive surface quality and implemented two use cases for using such data.

3.3 Cyclist Model for SUMO

3.3.1 Cycling Behavior in SimRa and SUMO

Preprocessing

Categorizing by Velocity

Acceleration

Deceleration

Velocity

Left-turn Behavior at Intersections

3.3.2 Improving SUMO's Bicycle Simulation

Longitudinal Kinematic Behavior

Left-turn Behavior at Intersections

Different Cyclist Models

3.3.3 Evaluation of the Cyclist Model for SUMO

Simulation Setup

Acceleration

Deceleration

Velocity

Left-turn Behavior at Intersections

(a) Straßburger Straße

(b) Goethestraße

Figure 3.17: Straßburger Straße and Goethestraße in Berlin are paved with large cobblestones and the roads are contested by cars that are parking or searching for a parking spot. In contrast, the sidewalks are paved with flat-surfaced paving stones and are quiet due to the absence of shops, restaurants or cafes. (Source: Apple Maps)

Chapter 4

Conclusions

4.1 Discussion

4.2 Summary and Outlook

Appendix A

Technologies and Concepts of a Larger Context

This appendix chapter contains definitions for technologies and concepts that are mentioned in the thesis, but are not of higher importance for it. Section A.1 introduces a consensus algorithm for distributed systems.

A.1 Finding Consensus in a Distributed System

The Paxos algorithm has first been described by Leslie Lamport in their work about the parliament on the island of Paxos.

Appendix B

Acronyms

AI Artificial Intelligence

ANN Artificial Neural Network

AUC Area under the curve

MCC Matthews correlation coefficient

BCE Binary Cross-Entropy

CNN Convolutional Neural Network

DBN Deep Belief Network

DFT Discrete Fourier Transform

DL Deep Learning

DNN Deep Neural Network

ECG Electrocardiogram

ESN Echo State Network

FCN Fully Connected Network

FPR False Positive Rate

GAF Gramian Angular Field

GAN Generative Adversarial Network

GDPR General Data Protection Regulation

GPS Global Positioning System

GRU Gated Recurrent Unit

HAR Human Activity Recognition

IQR Interquartile Range

LSTM Long Short-Term Memory

ML Machine Learning

MLP Multi-Layer Perceptron

NLP Natural Language Processing

OBS OpenBikeSensor

ROC Receiver Operating Characteristic

RNN Recurrent Neural Network

SAS Sensitivity at Specificity

SDAE Stacked Denoising Auto Encoder

SF Sensor-based Fusion

SGD Stochastic Gradient Descent

SVM Support Vector Machine

TPR True Positive Rate

TSC Time Series Classification

XAI Explainable Artificial Intelligence

Appendix C

Lexicon

Byzantine failure A malfunction of a component that leads to the distribution of wrong/-conflicting information to other parts of the system is called Byzantine failure. The name is based on the Byzantines Generals Problem, in which three Byzantine generals need to agree on a battle plan while one or more of them might be a traitor trying to confuse the others.

Appendix D

Listings

This is the appendix for code, that does not need to be provided directly inside the thesis.

D.1 Configuration for Node A

Listing D.1: Configuration for Node A

```
{
  "nodeID" : "nodeA",
  "publicKey" : "<public key>",
  "encryptionAlgorithm" : "RSA",
  "machines" : [ "192.168.0.132", "192.168.0.165" ],
  "publisherPort" : 8000,
  "messagePort" : 8010,
  "restPort" : 8080,
  "location" : "52.515249, 13.326476",
  "description" : "Raspberry Pi Cluster in EN 004"
}
```

Bibliography

[1] Moloud Abdar et al. "A Review of Uncertainty Quantification in Deep Learning: Techniques, Applications and Challenges". In: *Information Fusion* 76 (Dec. 2021), pp. 243–297.

[2] Rachel Aldred and Anna Goodman. "Predictors of the frequency and subjective experience of cycling near misses: Findings from the first two years of the UK Near Miss Project". In: *Accident Analysis & Prevention* 110 (2018).

[3] Davide Anguita et al. "A public domain dataset for human activity recognition using smartphones". In: *Proceedings of the 21th international European symposium on artificial neural networks, computational intelligence and machine learning*. 2013.

[4] Anahid Basiri et al. *Crowdsourced geospatial data quality: Challenges and future directions*. 2019.

[5] Stephen Bates et al. "Testing for Outliers with Conformal p-values". In: *arXiv:2104.08279 [math, stat]* (Apr. 2021).

[6] Felix Biessmann et al. "Automated data validation in machine learning systems". In: *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering* (2021).

[7] Christopher M Bishop et al. *Neural networks for pattern recognition*. Oxford university press, 1995.

[8] Bryan Blanc and Miguel Figliozzi. "Modeling the impacts of facility type, trip characteristics, and trip stressors on cyclists' comfort levels utilizing crowdsourced data". In: *Transportation Research Record* 2587.1 (2016).

[9] Bryan Blanc and Miguel Figliozzi. *Safety perceptions, roadway characteristics, and cyclists' demographics: A study of crowdsourced smartphone bicycle safety data*. Tech. rep. 2017.

[10] Eric Breck et al. "Data Validation for Machine Learning". In: *SysML*. 2019, pp. 1–14.

[11] Abhijnan Chakraborty et al. "Who makes trends? understanding demographic biases in crowdsourced recommendations". In: *AAAI ICWSM 2017*. Vol. 11. 1. 2017.

[12] Kaixuan Chen et al. "Deep learning for sensor-based human activity recognition: Overview, challenges, and opportunities". In: *ACM Computing Surveys* 54.4 (2021).

[13] Junyoung Chung et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling". In: *arXiv preprint arXiv:1412.3555* (2014).

[14] Ekin D. Cubuk et al. "Autoaugment: Learning augmentation strategies from data". In: *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.* Vol. 2019-June. May 2019, pp. 113–123.

[15] Andreas Eckner. "A framework for the analysis of unevenly spaced time series data". In: *Preprint. Available at: http://www. eckner. com/papers/unevenly_spaced_time_series_analysis* (2012).

[16] Wilfried Elmenreich. "Sensor fusion in time-triggered systems". PhD thesis. Citeseer, 2002.

[17] Tom Fawcett. "An introduction to ROC analysis". In: *Pattern recognition letters* 27.8 (2006).

[18] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[19] Takaki Hayashi and Nakahiro Yoshida. "On covariance estimation of non-synchronously observed diffusion processes". In: *Bernoulli* 11.2 (2005).

[20] Kaiming He et al. "Deep residual learning for image recognition". In: *IEEE CVPR 2016*. 2016.

[21] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997).

[22] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International conference on machine learning*. PMLR. 2015.

[23] Hassan Ismail Fawaz et al. "Deep learning for time series classification: a review". In: *Data mining and knowledge discovery* 33.4 (2019).

[24] Daniel Kahneman, Jack L Knetsch, and Richard H Thaler. "Anomalies: The endowment effect, loss aversion, and status quo bias". In: *Journal of Economic perspectives* 5.1 (1991).

[25] Ahmet-Serdar Karakaya, Jonathan Hasenburg, and David Bermbach. "SimRa: Using crowdsourcing to identify near miss hotspots in bicycle traffic". In: *PMC* 67 (2020).

[26] Tim Kuhlmann, Pablo Garaizar, and Ulf-Dietrich Reips. "Smartphone sensor accuracy varies from device to device in mobile research: The case of spatial orientation". In: *Behavior research methods* 53.1 (2021).

[27] Zachary C. Lipton, Yu Xiang Wang, and Alexander J. Smola. "Detecting and correcting for label shift with black box predictors". In: *35th Int. Conf. Mach. Learn. ICML 2018* 7 (Feb. 2018), pp. 4887–4897.

[28] Javier Ortiz Laguna, Angel Garcia Olaya, and Daniel Borrajo. "A dynamic sliding window approach for activity recognition". In: *UMAP 2011*. Springer. 2011.

[29] Neoklis Polyzotis et al. "Data lifecycle challenges in production machine learning: A survey". In: *SIGMOD Rec.* Vol. 47. 2018, pp. 17–28.

[30] Stephan Rabanser, Stephan Günnemann, and Zachary C. Lipton. *Failing loudly: An empirical study of methods for detecting dataset shift*. 2018.

[31] Albert Sánchez Fuster. "Detecting Near Miss Incidents in Bicycle Traffic Using Acceleration Sensor Data". MA thesis. Universitat Politècnica de Catalunya, 2020.

[32] Sebastian Schelter, Tammo Rukat, and Felix Biessmann. "Learning to Validate the Predictions of Black Box Classifiers on Unseen Data". In: *Proc. 2020 ACM SIGMOD Int. Conf. Manag. Data*. New York, NY, USA: ACM, June 2020, pp. 1289–1299.

[33] Sebastian Schelter et al. "On Challenges in Machine Learning Model Management". In: *Bull. IEEE Comput. Soc. Tech. Comm. Data Eng.* (2018), pp. 5–13.

[34] Allan Stisen et al. "Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition". In: *ACM SenSys 2015*. 2015.

[35] M Sugiyama and Motoaki Kawanabe Motoaki. *Machine Learning in Non-Stationary Environments Introduction to Covariate Shift Adaptation*. MIT Press, 2012.

[36] Du Tran et al. "Learning spatiotemporal features with 3d convolutional networks". In: *Proceedings of the IEEE international conference on computer vision*. 2015.

[37] John W Tukey et al. *Exploratory data analysis*. Vol. 2. Reading, Mass., 1977.

[38] Graham Upton and Ian Cook. *Understanding statistics*. Oxford University Press, 1996.

[39] Philip B Weerakody et al. "A review of irregular time series data handling with gated recurrent neural networks". In: *Neurocomputing* 441 (2021).

[40] David H Wolpert. "Stacked generalization". In: *Neural networks* 5.2 (1992).

[41] Jiahui Wu, Lingzi Hong, and Vanessa Frias-Martinez. "Predicting perceived cycling safety levels using open and crowdsourced data". In: *IEEE Big Data 2018*. IEEE. 2018.

[42] Shuochao Yao et al. "Deepsense: A unified deep learning framework for time-series mobile sensing data processing". In: *TheWebConf 2017*. 2017.

[43] William J Youden. "Index for rating diagnostic tests". In: *Cancer* 3.1 (1950).

[44] Daniel Zwillinger and Stephen Kokoska. *CRC standard probability and statistics tables and formulae*. Crc Press, 1999.