International Journal of
Geo-Information

Article

Recognition of Intersection Traffic Regulations from Crowdsourced Data

Stefania Zourlidou [1,*], Monika Sester [1] and Shaohan Hu [2]

1 Institut für Kartographie und Geoinformatik, Leibniz Universität, Appelstraße 9a, 30167 Hannover, Germany
2 Future Lab for Applied Research and Engineering (FLARE), JPMorgan Chase Bank, New York, NY 10017, USA
* Correspondence: stefania.zourlidou@ikg.uni-hannover.de; Tel.: +49-5117-624-852

Abstract: In this paper, a new method is proposed to detect traffic regulations at intersections using GPS traces. The knowledge of traffic rules for regulated locations can help various location-based applications in the context of Smart Cities, such as the accurate estimation of travel time and fuel consumption from a starting point to a destination. Traffic regulations as map features, however, are surprisingly still largely absent from maps, although they do affect traffic flow which, in turn, affects vehicle idling time at intersections, fuel consumption, CO_2 emissions, and arrival time. In addition, mapping them using surveying equipment is costly and any update process has severe time constraints. This fact is precisely the motivation for this study. Therefore, its objective is to propose an automatic, fast, scalable, and inexpensive way to identify the type of intersection control (e.g., traffic lights, stop signs). A new method based on summarizing the collective behavior of vehicle crossing intersections is proposed. A modification of a well-known clustering algorithm is used to detect stopping and deceleration episodes. These episodes are then used to categorize vehicle crossing of intersections into four possible traffic categories (p1: free flow, p2: deceleration without stopping events, p3: only one stopping event, p4: more than one stopping event). The percentages of crossings of each class per intersection arm, together with other speed/stop/deceleration features, extracted from trajectories, are then used as features to classify the intersection arms according to their traffic control type (*dynamic* model). The classification results of the dynamic model are compared with those of the *static* model, where the classification features are extracted from OpenStreetMap. Finally, a *hybrid* model is also tested, where a combination of dynamic and static features is used, which outperforms the other two models. For each of the three models, two variants of the feature vector are tested: one where only features associated with a single intersection arm are used (*one-arm model*) and another where features also from neighboring intersection arms of the same intersection are used to classify an arm (*all-arm model*). The methodology was tested on three datasets and the results show that all-arm models perform better than single-arm models with an accuracy of 95% to 97%.

Keywords: traffic-rules; traffic-regulations; crowdsourcing; GPS-trace; trajectories; classification; movement patterns; clustering; collective-behavior; smart city

check for updates

Citation: Zourlidou, S.; Sester, M.; Hu, S. Recognition of Intersection Traffic Regulations from Crowdsourced Data. *ISPRS Int. J. Geo-Inf.* **2023**, *12*, 4. https://doi.org/10.3390/ijgi12010004

Academic Editors: Wolfgang Kainz and Wei Huang

Received: 30 June 2022
Revised: 18 December 2022
Accepted: 21 December 2022
Published: 23 December 2022

1. Introduction

The idea of creating and sharing geographic information through individuals is not new. Individuals acting as sensors of their environment have been described as *citizens as sensors* [1], who can collect various kinds of data or share information from the environment they are in, such as photos, news, noise, speed measurements, air pollution data, etc. Since these data are combined with the geographical location from which they are taken, interesting information about a particular phenomenon in these locations for a given time or a given period of time can be estimated, e.g., the noise level of a place, the speed limit of a road, etc.

The widespread use of modern mobile devices has opened up new possibilities for *spatial crowdsourcing (SC)*, a term that describes "the potential of the crowd to perform real-world tasks with strong spatial nature that are not supported by conventional crowdsourcing (CC) techniques" [2]. CC techniques lack the spatial element and focus on transactions conducted entirely over the Internet. In contrast, SC requires a physical on-site presence and such information is collected either opportunistically or with active participation, an approach which has increasing potential [3].

Some examples of leveraging data collected from individuals include automatic detection of road network changes and map updates using GPS trajectory data [4,5], pothole detection using a crowd-sourced vehicle sensor data [6], estimation of road roughness from crowd-sourced bicycle acceleration measurements [7], and inferring the traffic state of roads by analysing the aggregated acoustic signal collected from the microphone sensor of the user's smartphone [8]. Predictions of phenomena, such as earthquakes (earthquake early warning), which until recently required special equipment, can now be implemented using common consumer devices, such as smartphones with low-cost sensors [9]. Another crowdsourcing-based service for citizens of large cities is information about the existence of vacant parking spaces near a destination [10]. Finally, crowdsourcing social media can also increase our understanding of human dynamics and spatio-temporal characteristics of cities and convey information about cities [11].

Other location-based applications that promise to make our everyday life much easier are the accurate estimation of the travel time from a starting point to a destination [12], the elimination of false warnings in the advanced driver assistance systems offered by modern vehicles [13] and the ambient sensing of autonomous vehicles, where traffic-related risks can be anticipated and driving actions can be planned accordingly [14]. Furthermore, traffic regulators, such as traffic signals, significantly affect the traffic flow at intersections, which, in turn, influence the fuel consumption and air pollution. Intersections are one of the dominant locations where excessive fuel is consumed [15] and certain traffic regulations, i.e., traffic signals, contribute more to air pollution compared to others (e.g., stop signs), due to the excessive vehicle emissions that are observed at those regulated locations [16]. Therefore, for environmentally friendly and sustainable solutions related to daily commuting and traffic, such information is critical. Nevertheless, the type of localised traffic regulations, represented as map features, is surprisingly still largely absent from National Agencies maps and from open maps, such as OpenStreetMap (OSM) [17]. This study is motivated by this fact and the main research question addressed is how to automatically and cost-effectively identify traffic regulations using crowdsourced data. According to [18], roads change up to 15% annually, and mapping them using surveying equipment has serious cost and time constraints. Therefore, the answer to the above question will be explored in the context of crowdsourcing. In the next paragraph we review existing methodologies for traffic regulation recognition from GPS data.

1.1. Existing Work

Studies related to the recognition of traffic regulation (TRR), mainly use either GPS tracks or images [19]. Traffic-sign recognition from in-vehicle cameras is a popular topic in the computer vision community, providing accurate detection of traffic signs [20,21]. However, although modern cars do have cameras, manufacturers do not share their data. However, even if such data were available, adopting it in the crowdsourcing scenario defined earlier has the disadvantage of generating a large amount of data (images) and, therefore, consuming resources, such as bandwidth and storage space. Additionally, the cameras have to be placed in vehicles, adding further constraints for broad user participation. According to [22], the key issue of the launching stage of a crowd-sourcing project is to convince people to start volunteering, and to persuade them, it is important to clearly demonstrate how easy it is to participate. In a hypothetical scenario of crowd-sourcing images for traffic regulation detection with cell phone cameras, people would need to place their cell phones on a car cell phone holder each time they drive, make sure the camera lenses and car window are clean, check that the phone's battery is charged or plugged into a phone charger (battery consumption is very high when taking photos), and address other possible issues, such as whether there is enough data storage or internet data available on the phone device, whether there is an internet connection, etc. All of these factors obviously do not make the crowd-sourcing scenario seem easy or appealing to participate in. In addition, there are privacy issues when identifying people or number plates in the images shared, as well as issues of occlusion, clutter, and illumination that one often has to deal with when processing images for object recognition [23].

Another image-based approach could use images from street-level photos offered by platforms, such as Google Street and Mapillary. As Hu et al. [24] point out, there are still many cities and places that are not covered by these services and, therefore, there are no images available to be crawled for traffic regulation detection. For example, in Google Street View, although countries such as the US and most European countries are fairly well covered, other countries are either for legal or privacy reasons partially covered (e.g., in Germany only some major cities have agreed to allow Google to take street photos) or not at all (e.g., eastern countries). In contrast, GPS traces (i.e., time-ordered sequences of recorded locations) are compact representations of the successive locations that a moving object passes through over time and can be recorded without special equipment and without the need to install a device on the front window of the car. For example, according to [25], an iPhone 6 has an overall average positional accuracy in an urban environment of 7–13 m (depending on the season, time of day, and Wi-Fi usage period) and could, therefore, be used for this purpose. Therefore, for the purposes of this study, that are listed at the end of this section, we chose to use GPS traces to achieve our goal.

Focusing on the existing TRR methodologies that use non visual data, we propose a new classification of the related studies according to the features used to classify traffic regulations. This taxonomy distinguishes three categories, namely *static-*, *dynamic-*, and *hybrid-approaches*. We call *static* features those that do not change over time, or if they do, they do so not very often. Such information can be extracted from maps. In contrast, we call *dynamic* features those that are extracted from dynamic entities that change over time, such as trajectories from moving objects (e.g., vehicles). From the trajectory of a vehicle, the speed of the vehicle, the duration of its stops, etc., can be extracted. *Hybrid-based* approaches include methods that use a mixture of static and dynamic features. The taxonomy is depicted graphically in Figure 1.

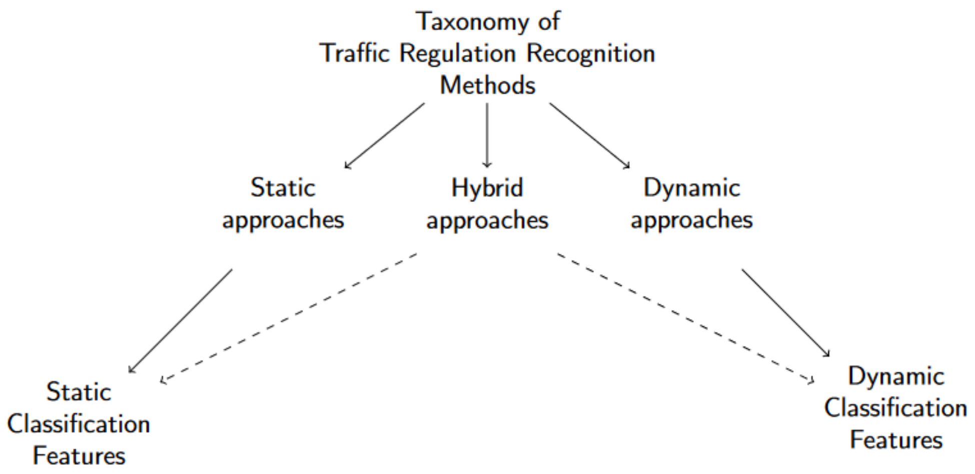

Taxonomy of
Traffic Regulation Recognition
Methods
Static approaches
Hybrid approaches
Dynamic approaches
Static Classification Features
Dynamic Classification Features

Figure 1. Taxonomy of methods for traffic regulation detection from crowdsourced GPS data.

Here, we review only some distinct studies of the field. A detailed description of the methods of the reviewed articles (those published before 2019), their limitations and a comprehensive critical overview of the research field can be found in a related systematic literature review [19], the first, and so far only, attempt to illustrate the progress and challenges of the research field. A common methodological element of all these studies is the extraction of classification features from the available data and then using them to classify the intersection arms. The term intersection arm refers to the road that connects one intersection with another one, as illustrated in Figure 2. A three-arm (three-way) intersection has three arms and a four-way intersection has four arms, as shown in Figure 2a,b.

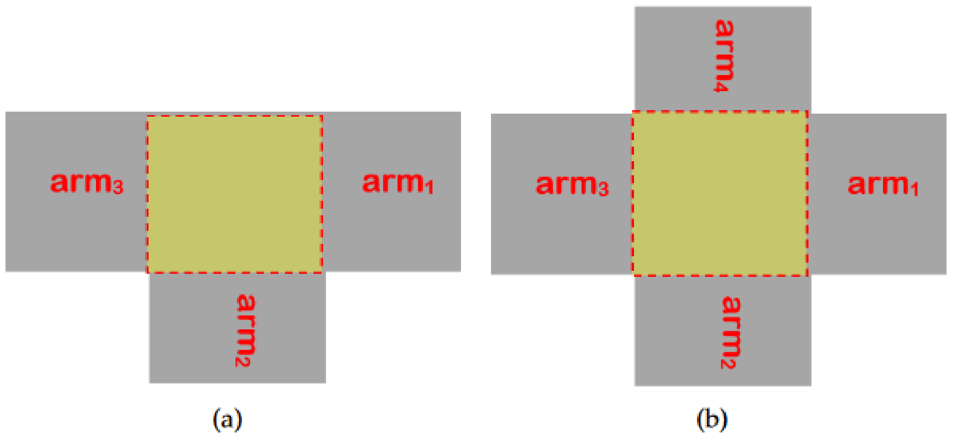


Figure 2. These figures depict the intersection arms of a three-way (**a**) and a four-way (**b**) intersections.

In the *static* category there is only one study that uses static classification features, extracted from open street maps, such as OSM. Saremi and Abdelzaher [26] extract OSM features related to speed as well as distance. In particular, they extract the speed rating of road segments, the distance of the nearest connected intersections, the end-to-end distance of the road to which an intersection belongs, the semi-distances of an intersection from both ends of the road to which it belongs, and the category of the road segment that characterizes its importance in the road network (e.g., primary, secondary, highway, etc.). An important observation in this methodology is that the classification features for an intersection arm contain information also from neighboring arms of the same junction, which from now one we will call *context* arms. A Random Forest classifier with 500 trees is trained to categorize three types of regulators: traffic lights, stop signs, and uncontrolled intersections. The classification accuracy, with a confidence level of 80% in the prediction, is reported as 95% (worst case in the different cities tested). No tests were conducted on the effect that the number of trajectories from which classification features are extracted may have on classification performance.

The *dynamic* TRR category includes approaches that use various features mainly related to stopping and/or deceleration episodes, as well as to speed sequences (speed-profiles) of the vehicles while approaching the intersection. These features are usually extracted from a large set of trajectories. Stopping/deceleration episodes are detected on each trajectory crossing an intersection arm and then statistical measures are calculated from these attributes for all trajectories crossing an intersection arm, such as the average number of stopping episodes, the minimum number of deceleration episodes observed on tracks crossing an intersection, etc. A distinctive work of this category is that of Hu et al. [24] which defines two categories of classification features, *physical* and *statistical*. The physical features include the duration of the last stop episode before crossing the junction, the minimum crossing speed, the number of vehicle deceleration episodes, the number of stop episodes, and the distance of the last stop episode from the intersection. Statistical features are defined as the minimum, maximum, average, and variance of the physical features. Only straight trajectories for feature extractions are used to eliminate the possible bias that may turning trajectories have. By excluding turning trajectories, certain arms in T-shape intersections where trajectories always have to turn, will be always excluded entirely from classification. For this reason, the following domain knowledge rule is applied, labeling those arms from T-intersections: (denote A, B, and C for left, right, and bottom ends of the intersection): If A/B are uncontrolled (right-of-way rule is applied), then C is stop controlled, otherwise C has the same type as A and B. In other words, if A/B have traffic lights, so does C, otherwise C has a stop sign. The Random Forest classifier, as well as Spectral Clustering, were tested for a three-category classification problem (traffic lights, stop signs, and uncontrolled intersections), achieving accuracy above 90% for various feature settings and classification experiments.

Saremi and Abdelzaher [26] propose a classification method (they call it the *crowd-based method*) that uses the mean, minimum, maximum, and standard deviation of the values of three attributes extracted from the trajectories crossing an intersection: the crossing speed, the number of stopping episodes, and the duration of the latter. Crossing speed is considered to be the lowest instantaneous speed of the vehicle when crossing an intersection on its approach along the given intersection arm. The number of stop episodes is considered to be the number of time intervals during which the vehicle has stopped and is idling when crossing the last section of the road along a given intersection arm. The stopping duration is taken as the length of the last time interval, during which the vehicle stopped and idled. It is not clear whether the number of intersections used to test the map-based model (see static category) is the same for this model, as no detailed classification report is provided in the article. Additionally, no quantitative description of the datasets used (e.g., the number of regulators per regulator category) is given. However, similar to their proposed map-based model, a Random Forest classifier is trained to categorize three types of regulators: traffic lights, stop signs, and uncontrolled intersections. The classification accuracy, with a confidence level of 80% in prediction, is reported as 91%. Because a detailed classification report describing the results per regulator class is not provided, nor is a quantitative description of the datasets given (e.g., the number of regulators per regulator class), we cannot compare our results with this study, although we acknowledge this work as methodologically closest to our work.

Golze et al. [27] proposed a Random Forest classifier with oversampling and a Bagging Booster to predict intersection regulators with 90.4% accuracy. Along with other physical features, such as the number of standstill events, the duration of standstill events, the mean distance from junction of all standstill events, the duration of the last standstill event, the distance from junction of the last standstill event, the mean speed and maximum speed while approaching the junction, they also calculate the percentage of trajectories with at least one standstill event. By also conducting a feature importance analysis, they show that the last feature is of great importance compared to other classification features. In addition, they tested the case of using only straight trajectories, as well as both straight and turning trajectories, finding that by eliminating turning trajectories, the classification performance was better.

Two distinct dynamic approaches that use speed-based features instead of episode-based features, such as stop and deceleration episodes, are those of [28,29]. Méneroux et al. [28] detect traffic signals (binary classification problem) using speed profiles. By testing three different ways for feature extraction—functional analysis of speed recordings, raw speed measurements, and image recognition techniques—they find that the functional description of speed profiles using wavelet transforms outperforms the other approaches. Random Forest classification achieved the best accuracy (95%) compared to the other tested classification techniques. However, the authors point out that the lack of data is a severe limitation of the experiments, as their dataset contained only 44 instances of traffic lights. Moreover, Cheng et al. [29] propose a sequence-to-sequence framework for dealing with a three-class classification problem (traffic lights, priority signs, and uncontrolled junctions) by feeding speed-profiles to a deep-learning classifier, showing that a Conditional Variational Autoencoder (CVAE) can predict regulators with 90% accuracy, outperforming the baseline model (a Random Forest classifier with 88% accuracy) that uses summarized statistics of movement as features.

In the *hybrid-based* category, there are two studies that use a mixture of static and dynamic features. Saremi and Abdelzaher [26] is the first to initiate such a model, where, in addition to the map-based information extracted from OSM, they use the availability of dynamic crowd-sourced information (GPS traces) to incorporate features in the classification model which are extracted from the trajectory, such as the traverse speed, the number of stops and the stopping duration of the last time interval that the vehicle has stopped and is idling. Furthermore, after the classification, the methodology includes a step where a consistency check is completed among the predicted labels at the intersection level. The following domain knowledge rule is employed: either all or none of the approaches contributing to the same intersection have a traffic light. This implies that when the classifier labels some of the approaches of an intersection, but not all of them, as traffic-light, the predicted label should be revised. The revision makes either all or none of the intersection approaches have a traffic light, this being decided upon utilizing computed probabilities based on the fraction of decision trees voting for the approaches' alternative labels. A Random Forest classifier with 500 trees is trained to categorize three types of regulatory types: traffic lights, stop signs, and uncontrolled intersections. The classification accuracy, with a confidence level of 80% in prediction, is reported as 97%, outperforming both map-based and crowd-based (dynamic) models (see static and dynamic categories).

Furthermore, Liao et al. [30] described a traffic-light detection (binary classification problem) and impact assessing framework that can detect the presence of traffic signals and estimate the influence range of traffic lights (in space and time) using speed time series extracted from GPS trajectories and intersection-related features, such as intersection type (connects arterial roads, connects secondary roads, connects arterial and secondary roads), road type (according to two speed limits) and traffic flow information. A distributed long short-term memory (DLSTM) neural network is used in the proposed framework, which treats discrete and sequential features separately and achieves an AUC value under the ROC curve of 0.95.

A first observation about the existing methodologies is that, only in the work of Saremi and Abdelzaher [26] and for the map-based (static) model, the classification features for an intersection arm contain information from context arms. To the best of our knowledge, no other methodology proposed to date, e.g., dynamic models, has considered using features that include information from neighboring arms. We consider this to be an interesting aspect of the problem to investigate, as information from neighboring arms may be informative. Additionally, all methods use very similar classification features (stop and deceleration episodes, speed values, map extracted features). It would be interesting to apply a feature importance analysis and examine which features are indeed important for the classification task and whether a feature selection would improve the classification importance. Moreover, the effect of turning trajectories on classification performance has been examined only in one study [27], while the work of [24] reports that it eliminates turning trajectories from the feature computation. All other papers make no mention of this aspect of the problem. Furthermore, motivated by the single fundamental rule that [26] used to correct possible misclassifications in traffic-light-controlled intersections and the labeling rule that [24] uses to label arms of T-shaped intersections, we believe that checks of predicted labels at a step after classification could trigger a mechanism for correcting predicted labels based on fundamental knowledge rules that preserve label consistency at the intersection level. In this way, possible misclassifications of intersection arms could be corrected.

Additionally, more conclusions could be drawn if classification performances were assessed across all studies with the same metrics and if detailed classification reports were provided, including the number of training and test samples per regulatory class. It would be even better if all TRR methodologies were tested on the same datasets, as trajectory datasets can vary significantly in terms of sampling rate, trajectory density, number of intersections, and traffic regulatory classes. Reference datasets (benchmarks) would facilitate direct comparisons between different TRR methods. Moreover, it seems that hybrid methods, such as that of [26], that combine static and dynamic features perform better than those using only static or dynamic features, and given that this idea is only addressed in two articles [26,30], it may be an interesting methodological direction to explore further.

1.2. Research Gap and Article Contributions

The idea of using GPS data to identify intersection traffic regulation is based on two assumptions: (1) traffic controls affect driver behavior in an indicative and uniform manner for all drivers crossing the same intersection (or for all crossings of a single driver that crosses an intersection several times), and (2) similar movement patterns are observed at different intersections that are regulated by the same traffic control. In other words, the assumptions are that a specific movement pattern can be identified at a regulated location from the crossing trajectories of that certain location and similar movement patterns can be observed at different locations (intersections) that are regulated by the same traffic control. Of course there may be drivers who may violate a traffic signal or stop sign under certain circumstances, and, consequently, their movement behavior differs from the mainstream pattern, but TRR methodologies assume that the vast majority of drivers do not systematically violate traffic rules, and if there are such cases expressed in the dataset, they can be considered outliers. Therefore, the expectation is that the vast majority of drivers respect road traffic rules, and the two assumptions mentioned earlier can be valid only under this prerequisite.

Considering the limitations of existing methodologies, as described in the previous section, new research directions open up, that this article addresses. More specifically, the research contributions of this study can be summarized as follows:

1. It proposes and tests a new traffic regulator recognition methodology and evaluates it in different dataset settings, i.e., different cities, regulators, trajectory densities and dataset sizes (Section 2.2.2).
2. Since the literature review identified that hybrid methods seem to perform better, a proposed hybrid TRR method tests this hypothesis (Hybrid Model: TRR from Crowdsourced Data (Dynamic and Static Features)).
3. It examines the effect of GPS sampling rate in the classification performance (The Effect of Sampling Rate). This aspect of the problem was motivated from the different sampling rates of the datasets used for testing the methodology, as well as from the observation that the quality of the GPS trajectories may affect the classification performance.
4. The proposed methodology is tested under different feature settings, i.e., including information from context arms and using exclusively information from one arm (Section 2.2.3). An analysis of the importance of the features is also carried out to determine the key features for the classification task.
5. The proposed methodology explores the effect (if any) of turning trajectories in the classification performance (The Effect of Turning/No-Turning Trajectories).
6. The proposed methodology examines whether there is a certain number of trajectories per intersection arm that leads to optimal classification performance (The Effect of Number of Trajectories).
7. The proposed methodology proposes an additional consistency check of the predicted labels at an intersection level, correcting misclassified regulators when possible (Section 2.3).
8. It makes available a new trajectory dataset along with the groundtruth intersection regulators that the trajectories cross, which can be used as benchmark from TRR methodologies.

2. Materials and Methods

2.1. Datasets

In this section, we describe the data requirements for addressing the problem of traffic regulation recognition (TRR) from crowdsourced data (GPS trajectories and OSM), as well as the limitations under those requirements.

2.1.1. Dataset Requirements and Limitations

As the proposed method is based on supervised classification, both GPS traces (for feature computation) and regulators (as labels) of intersection arms are needed. The process of labelling is time-consuming and poses general limitations for exploring the problem of detecting regulations, as such a groundtruth map is always needed for training and/or validation purposes (more discussion on this limitation can be found at [19]). Although there are many open trajectory datasets that can be freely downloaded to use in the context of the research question we address here, the additional labelled data that are also required (the regulations of junctions) are not available.

Furthermore, another limitation regarding the trajectory dataset one can use is the sampling rate of GPS traces. Most open trajectory datasets have a low sampling rate (e.g., 1 sample every 15 s or per minute) and cannot be used to extract features, such as stopping or deceleration events, because between two GPS samples taken e.g., every 15 s, one or more stopping/deceleration events could occur and would not be detected. Therefore, having to deal with these challenges of the datasets, we were able to access three suitable datasets in total: one recorded by the first author of this article and now available as open dataset [31,32], one shared by the first author of the journal article [24], and one open trajectory dataset [33] for which the groundtruth map had to be manually conducted by the first author of this article and is now available as an open dataset too [34]. The three datasets are described in the next section.

2.1.2. Datasets for Testing the Proposed Method

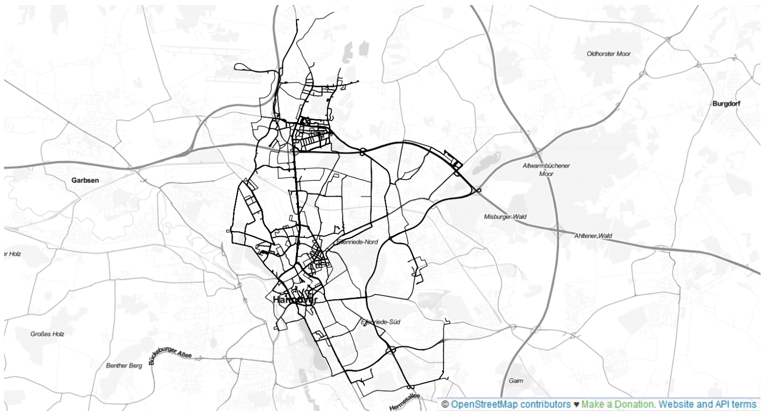
In Table 1, we give a description of the datasets we used to test the proposed method and to carry out experiments on the number of trajectories per intersection required for optimal classification accuracy (The Effect of Turning/No-Turning Trajectories). The datasets contain various combinations of rules, with the Champaign [24] and Chicago [33] datasets containing the same rule classes (uncontrolled (UN), stop sign (SS) and traffic signals (TS)) and the Hanover dataset containing a subset of rules from the Champaign and Chicago datasets, plus another regulator (UN, priority sign (PS) and TS). We consider one rule per intersection arm, which means that a three-way or a T-intersection has three rules and a four-way junction four rules. Hence, depending on the types of intersections (e.g., three-way, four-way, etc.), the total number of rules per dataset varies accordingly. The richest dataset in terms of rules is the Hanover dataset and the richest in terms of GPS traces (trajectories) is the Champaign dataset. Only the Chicago trajectory dataset is publicly available [33]. The rest are self-collected. All data are naturalistic (naturalistic data can be defined as data that make up records of human activities that are neither elicited by nor affected by the actions of researchers [35].) in the sense that drivers were not given external instructions on how to drive. Figure 3 illustrates the three datasets.

Table 1. Dataset used for testing the proposed methods.

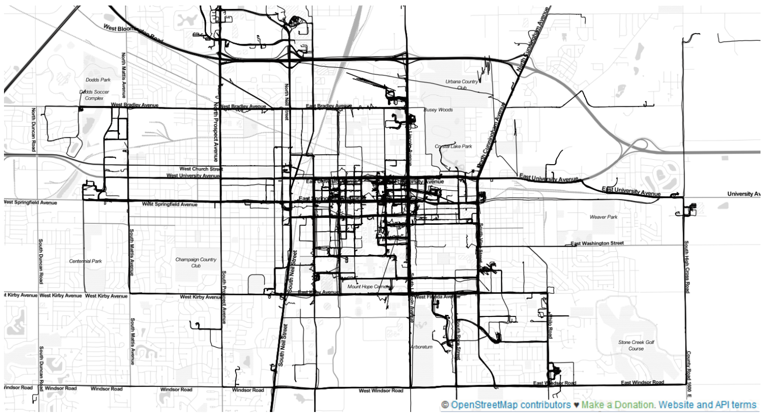
City $^{\diamond}$	Junc.	Rules	Traj.	Rules *
Champaign (Illinois, US)	713	2501	2202	TS, SS, UN
Chicago (Illinois, US)	156	568	889	TS, SS, UN
Hanover (DE)	1063	3538	1204	TS, PS, UN

$^{\diamond}$ Country names: DE (Germany), US (United States); * TS: Traffic Signals, SS: Stop Sign, UN: Uncontrolled, PS: Priority Sign.

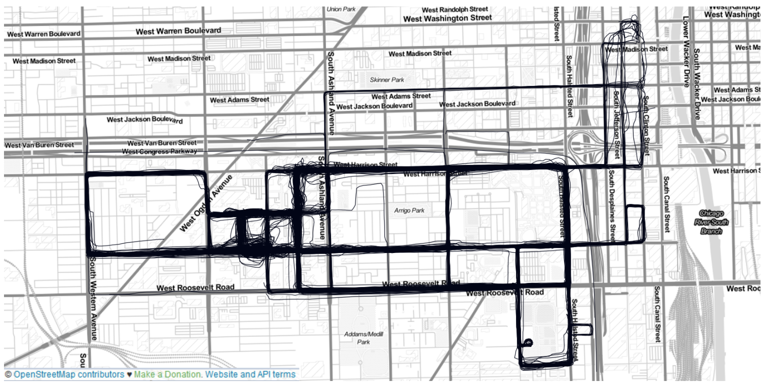
Furthermore, all but the Chicago groundtruth map of regulators were created manually by field observation (visiting all intersections and recording regulations). To obtain the regulations for the Chicago dataset, we used the Mapillary street imagery data [36], verified by other sources, and manually extracted the intersection rules. The Chicago groundtruth map is available here [34]. The Hanover dataset (trajectories and groundtruth traffic regulations) is available here [31,32]. Moreover, the Champaign dataset has an average sampling rate of 1 Hz, Hanover of 0.59 Hz (1 sample every 1.7 s), and Chicago has, on average, 0.28 Hz. Finally, although the Hanover dataset contain yield-controlled arms (YS), most of them are sparsely sampled (few tracks cross them). For this reason, we excluded YS from our analysis.



(**a**) Hanover dataset.



(**b**) Champaign dataset.



(**c**) Chicago dataset.

Figure 3. The three datasets that have been used in this study.

2.2. Methodology

An important element of movement patterns are stop and deceleration events, which are detected based on a clustering approach. In this section, we describe a modification of a known clustering algorithm for detecting short-term significant events (Section 2.2.1), as well as a new TRR methodology (Section 2.2.2) whose efficacy is examined under various settings (The Effect of Turning/No-Turning Trajectories and The Effect of Number of Trajectories).

2.2.1. Clustering-Based Stop and Deceleration Event Detection in Trajectories (The CB-SDot Algorithm)

The CB-SDoT algorithm (see Algorithm A1 in Appendix A) is a modification of the CB-SMoT algorithm [37] for detecting stop/deceleration events. CB-SMoT was originally proposed for discovering interesting places in trajectories. A place has the potential to be *interesting* [38] or *significant* [39], if someone spends a certain amount of time (i.e., over a time limit) in it. This means that by analyzing the trajectory of a moving object (e.g., a pedestrian, vehicle, animal, etc.), we can detect locations that are interesting to the observed object, given that it stayed there for a relatively long time. The problem of partitioning trajectories into sequences of *stops and moves* is a well-studied topic [40] and there are many different algorithms that provide solutions (e.g., [41–43]).

Here, we adopted the CB-SMoT solution [37], which is a clustering technique that works similarly to the well-known density-based clustering algorithm DB-SCAN [44], but in addition to the distance between points, it takes into account the temporal distances between them to determine the clustering criteria. CB-SDoT identifies clusters of points (Figure 4) that within a certain distance Eps remain at least $minTime$ and (unlike CB-SMoT) no more than $maxTime$. The extra $maxTime$ restriction is required so that longer stops, not related to traffic events, such as shop visits, are not considered as interesting events. The values for these parameters were defined experimentally (stops: Eps = 10 m, $minTime$ = 4 s, $maxTime$ = 600 s, decelerations: Eps = 10 m, $minTime$ = 2.4 s, $maxTime$ = 3.9 s). Each detected cluster is a time-ordered sequence of points that represents a stopping or a deceleration event. For each cluster, we define a point as the *representative* of the cluster. We define such a point as the last point in the time series of the points of the detected cluster that has the lowest speed. The notions of *core point*, *linear neighborhood*, and *neighboring points* refer to the same ones originally defined in [37,44] and for the sake of space, we omit to give their definitions here.

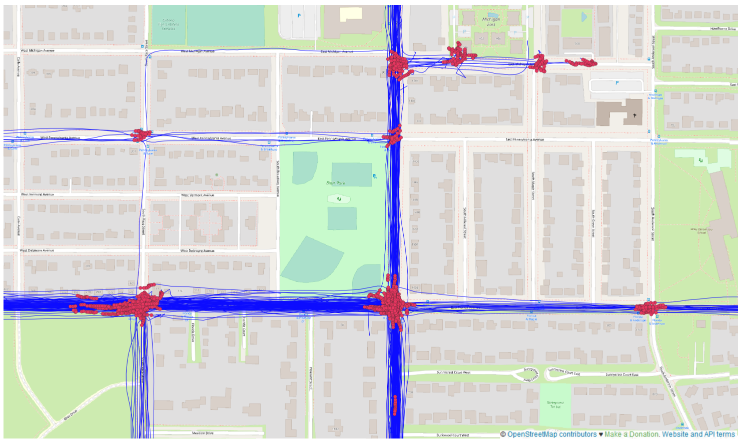

© OpenStreetMap contributors ♥ Make a Donation. Website and API terms

Figure 4. Stop events (red points) detected from the CB-SDoT algorithm in vehicle trajectories (blue lines).

2.2.2. Learning Traffic Regulators from Crowdsourced Data

c-Dynamic Model: TRR via Summarization of Collective Movement Behavior

The proposed regulation recognition method is based on the hypothesis that each regulator class *enforces* vehicles to move on certain moving patterns, and by detecting those patterns we can then *recover* the regulators. We describe the observed movement patterns by using, as core elements (pattern blocks), the stop and deceleration events, as well as their non-observation, that is, no stop and no deceleration event (four pattern blocks). For example, a movement pattern can be a free crossing of a junction where no stop or deceleration event is observed. Another pattern can be stopping only one time before crossing the junction. Numerous patterns can be defined by combining these patterns blocks. Then, each regulated junction arm can be described from the movement patterns observed at its location, by simply summarizing the patterns (each described by stop/deceleration events) of all the trajectories that cross that junction arm.

For example, suppose N trajectories cross a junction arm i_arm. From the N trajectories, M trajectories cross the i_arm having a constant speed (p_1: free flow, i.e., no stop, no deceleration events) and $N - M$ trajectories stop one time at the junction and wait for a few seconds (p_2: one stop before crossing the junction). We can then describe the i_arm using the ratios of the trajectories following the two motion patterns, p_1 and p_2. Defining p_1 as the motion pattern of free flow and p_2 as the motion pattern with stops, then i_arm can be quantitatively described as a location where a *mixed* motion behavior is collectively observed and which can be summarized as follows:

$$[p_1, p_2]_{i\_arm} = [\frac{M}{N}, \frac{N-M}{N}], with \sum_{n=1}^{2} p_n = 1$$

Applying this idea in the context of our problem, we define four different movement patterns, depicted in Figure 5, instead of the two we used in the previous example ($\sum_{n=1}^{4} p_n = 1$):

- p_1: Free-flowing (unobstructed) movement while crossing the intersection. Consequently, no deceleration or stopping events are observed.
- p_2: The vehicle slows down without stopping.
- p_3: The vehicle stops only once before crossing the intersection. However, it may slow down more than once.
- p_4: The vehicle stops more than once before crossing the intersection.

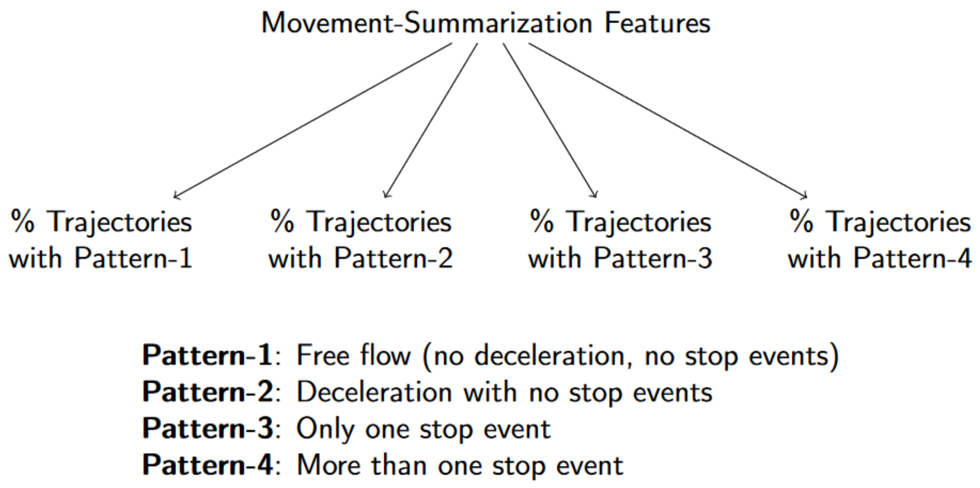

Movement-Summarization Features
% Trajectories with Pattern-1
% Trajectories with Pattern-2
% Trajectories with Pattern-3
% Trajectories with Pattern-4
Pattern-1: Free flow (no deceleration, no stop events)
Pattern-2: Deceleration with no stop events
Pattern-3: Only one stop event
Pattern-4: More than one stop event

Figure 5. The four movement patterns that describe a vehicle's crossing of a junction arm.

Schematically, this idea is illustrated in Figure 6. Such a mixture of motion patterns has been used in [45], but in a different context. There, the goal was to dynamically determine the range of an intersection for obtaining the traffic flow speed and intersection delay under different traffic patterns. Here, we define movement patterns for summarizing the collective behavior of vehicles at an intersection. The selection of the four patterns was motivated after generating plots of vehicle speed profiles at various intersections and made the following observations: at a traffic light a mixture of patterns was observed where proportionally patterns 1 and 4 were distinct compared to the corresponding values at a priority controlled intersection or at a priority sign.

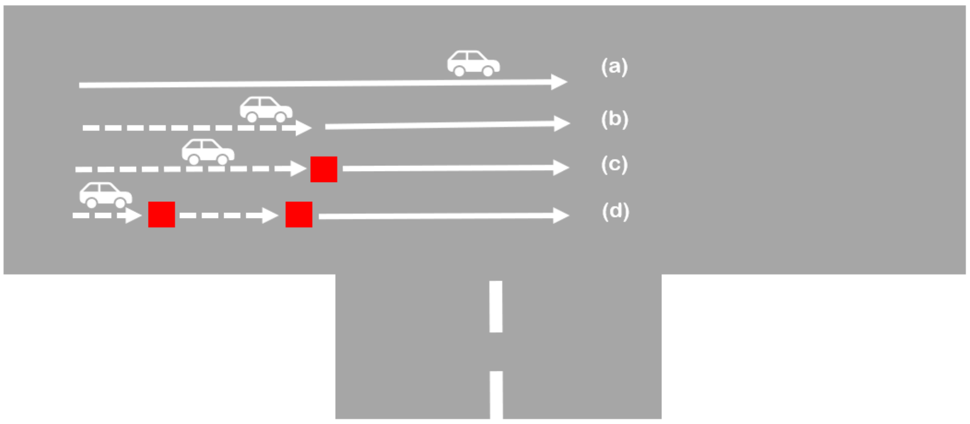


Figure 6. The four movement patterns that describe a vehicle's crossing of a junction: (**a**) unhindered crossing, (**b**) deceleration (dotted line) without stopping, (**c**) stop once (red dot), and (**d**) stop more than once (here two stop events are depicted with the two red dots).

Along with these four patterns computed per junction arm, that are used as classification features, we add in the feature vector additionally six percentiles of average speed (10th, 20th, 40th, 60th, 80th, and 95th) of the trajectories that cross each junction arm. These features, again, were motivated from the speed profiles we plotted at different intersections, and observed different speed distributions between intersections controlled by different regulators. Therefore, a 10-dimensional feature vector (four pattern values plus six percentile values) is fed to the classifier for TRR. We refer to this method as c-dynamic model (c- stands for compact, we explain later the difference of the c-dynamic from the dynamic model). Figure 7 depicts the workflow of the proposed approach.

Regarding the implementation of this idea, all steps, from feature extraction to intersection classification, are expressed in Algorithm 1. Since the problem is formulated as a classification problem, we first extract the features that the classifier needs to learn how to map to the label space. As we explained earlier, each intersection is represented by 10 features. We first compute all stop and deceleration events for each trajectory by using the CB-SDoT algorithm (Section 2.2.1).

Next, for each intersection arm of the dataset, we find all the trajectories that cross it, and for each trajectory we find the number of stopping and slowing events (if any) that occur within the half of the distance between the current intersection and the previous intersection visited, as well as the average speed. As Figure 8 depicts, for an intersection that has four arms, *i_arm*, *m_arm*, *j_arm*, and *k_arm*, for each arm and for each trajectory that crosses it, we compute the stop and deceleration events within the half distance that connects the intersection with the previously visited one (white arrows). By using such a non-fixed distance, we avoid having features calculated along a road segment that run along more than one intersection arm. For example, assuming we use a fixed distance of 100 m, in residential areas there may be intersections that are less than 100 m away from their neighboring intersections. In such cases, the calculated features would be calculated along intersection arms crossing more than one intersections. According to the stop/deceleration events found in the trajectory that crosses an intersection arm, we categorize the movement behavior of each trajectory into one of four movement patterns. We then calculate the percentages of the trajectories for each pattern and for each intersection arm in the dataset.

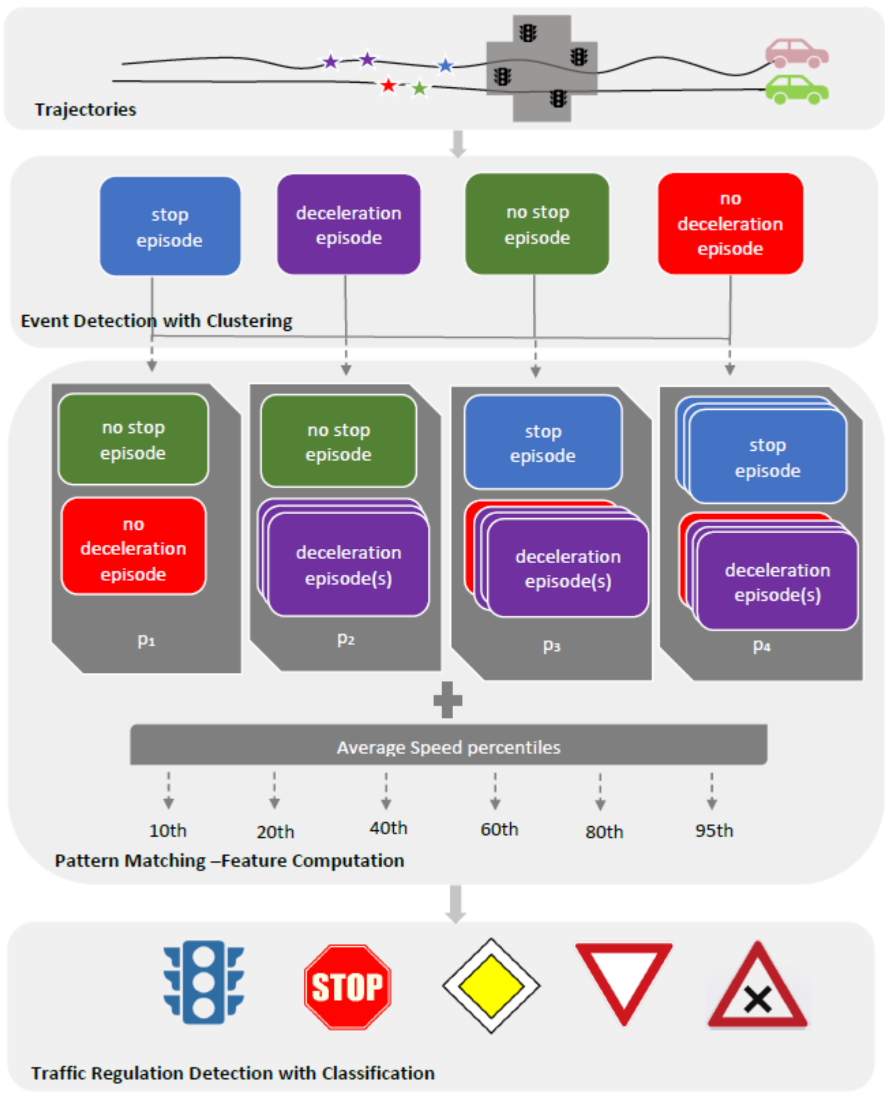


Figure 7. The steps of the proposed methodology for traffic regulation detection from GPS tracks.

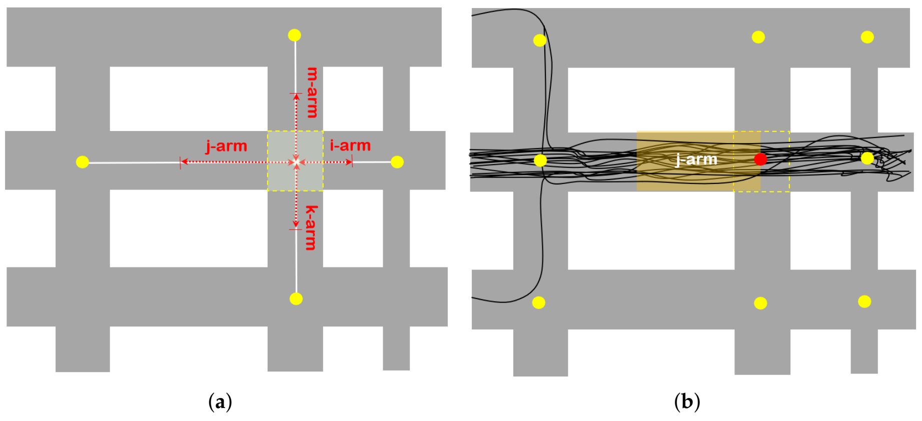


Figure 8. Each intersection (yellow dots) is composed from intersection arms, that connect it to nearby intersections (white lines). Classification features are computed per arm, within half the distance of the road segment that connects the current arm with the one previously visited by the trajectory (red dotted arrows in (**a**)). For each trajectory (black lines) in (**b**) that crosses the intersection arm *j-arm* from west to east, stop and deceleration events are computed within the orange indicated area along *j-arm*.

Algorithm 1: Traffic regulation recognition from GPS tracks via summarization of movement patterns.

```
Data: GPS tracks, ground truth map (coordinates of junctions, traffic rules of
      junction arms)
Result: Label junction arms with the regulator type they are controlled with;
while not all trajectories have been processed do
    Find all stop events within the trajectory;
    Find all deceleration events in the trajectory;
    Add stop events in DB table $StopTB$;
    Add deceleration events in DB table $DecTB$;
end
for $i \leftarrow 1, numJunctionarms$ do
    Find the trajectory $TrjIds[]$ that cross the $i$ junction arm ;
    $numTrj \leftarrow$ number of $TrjIds[]$;
    for $j \leftarrow 1, numTrj$ do
        $Trj = TrjIds[j]$;
        Find the stop events of trajectory $Trj$ that are along the $1/2$ length of road
          segment between junction arm $i$ and the previous visited junction arm ;
        Find the deceleration events of trajectory $Trj$ that are along the $1/2$ length
          of road segment between junction arm $i$ and the previous visited junction
          arm ;
        Match the crossing behavior (num. of stop/deceleration events) of the
          trajectory $Trj$ to one of the four patterns;
        Estimate the average crossing speed
    end
    $p1, p2, p3, p4 \leftarrow$ Compute the % of the four patterns for junction arm $i$ ;
    $s1, s2, s3, s4, s5, s6 \leftarrow$ Compute the 10th, 20th, 40th, 60th, 80th and 95th average
      speed percentiles for junction arm $i$ ;
    Add feature vector $p1, p2, p3, p4, s1, s2, s3, s4, s5, s6$ to DB table $FeaturesTB$ ;
end
Classification training and testing with data from $FeaturesTB$ ;
Print classification report ;
```

Dynamic Model: TRR from GPS Trajectories

The dynamic model can be considered as an extension of the c-dynamic model, using, in addition to the ten features used by the latter model, some statistical features (average, variance, minimum, and maximum values) derived from the stopping and deceleration events and from the estimated vehicle speed. Compared to existing models, e.g., [24,26,27,46], this model has a richer feature vector (86 features in total) that includes more deceleration- and speed-related features. All features are listed in Table 2. The computation of the features and the steps of the TRR are completed in a similar way as for the c-dynamic model.

Table 2. Overview of the features derived from the *dynamic* model.

	#	Physical Feature *	Statistical Features **			
Stop events	32	Number of stops	avg	var	min	max
		Duration of last stop	avg	var	min	max
		Duration of all stops	avg	var	min	max
		Mean Duration of all stops	avg	var	min	max
		Median Duration of all stops	avg	var	min	max
		Distance of last stop	avg	var	min	max
		Mean Distance of all stops	avg	var	min	max
		Median Distance of all stops	avg	var	min	max
Decel. events	32	Number of decel. events	avg	var	min	max
		Duration of last decel. event	avg	var	min	max
		Duration of all decel. events	avg	var	min	max
		Mean Duration of all decel. events	avg	var	min	max
		Median Duration of all decel. events	avg	var	min	max
		Distance of last decel. event	avg	var	min	max
		Mean Distance of all decel. events	avg	var	min	max
		Median Distance of all decel. events	avg	var	min	max
Speed	18	Minimum speed	avg	var	min	max
		Maximum speed	avg	var	min	max
		Average speed	avg	var	min	max
			Percentile avg speed (0.1)			
			Percentile avg speed (0.2)			
			Percentile avg speed (0.4)			
			Percentile avg speed (0.6)			
			Percentile avg speed (0.8)			
			Percentile avg speed (0.95)			
Mov. Pattern	4		Traj. % with no stops/decels			
			Traj. % with decels			
			Traj. % with one stop			
			Traj. % with more than one stop			
Sum	86					

* Derived per trajectory, ** Derived from the physical features per intersection arm, i.e., from all trajectories that cross the intersection approach. avg: average, var: variance, min: minimum, max: maximum.

Static Model: TRR from OSM Extracted Features

The static model uses features extracted from OSM, originally proposed from Saremi and Abdelzaher [26]. Each intersection approach (arm) is described by five features. Three of them regard street lengths and are illustrated in Figure 9. More specifically, the features of the static model are the following:

1. The **end-to-end distance** of the street that the intersection arm belongs to (light blue arrow in Figure 9). The length of a street is indicative of its importance in the street network. The same rationale applies also to the other distance-based features (2, 3).
2. The **semi-distance** of an intersection arm is the distance from the center of the junction to the center of the most distant intersection that the intersection arm is connected to (yellow arrow in Figure 9).
3. The **closest distance** of an intersection arm is the distance from the center of the junction that the arm belongs to, to the center of the nearest junction that the arm is connected to (green arrow in Figure 9).
4. The **maximum speed** of an intersection approach is the maximum allowed speed along it. Intersections controlled by a traffic signals in general have higher speed limit (e.g., 50 Kmh) compared to stop-sign controlled intersections (e.g., 30 kmh).

5. The **street category** refers to the street type category of the intersection arm (e.g., primary, secondary, tertiary, residential).

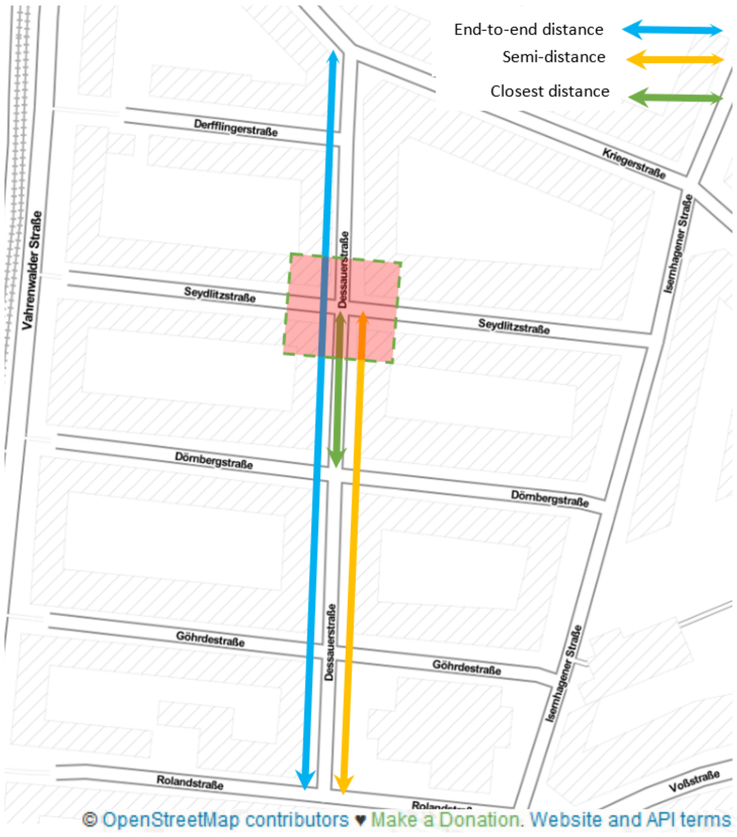


Figure 9. Illustration of the distance-related features of the static model along the north–south intersection approach of a four-way intersection (shown in red).

Hybrid Model: TRR from Crowdsourced Data (Dynamic and Static Features)

The hybrid model uses the features from both the dynamic and static model, i.e., the 86 features of the dynamic model and the 5 features of the static, in total 91 features.

2.2.3. One-Arm vs. All-Arm Models

So far, in the four classification models (c-dynamic, dynamic, static, hybrid), each intersection arm is represented by a set of features extracted exclusively from that arm (*one-arm models*). Motivated by the fact that for the classification of an intersection arm, information from adjacent intersection arms may also be relevant, each model is enriched with further features leading to the corresponding *all-arm model*, where each intersection arm is represented in the feature vector by a combination of features extracted from context arms. For example, the *i_arm* of the intersection depicted in Figure 8a according to the all-arm c-dynamic model has 4 (arms) $\times$ 10 (features) = 40 features: 10 features for each arm of the intersection, starting from *i_arm* and adding features from context arms in a clockwise order:
$[p_1, p_2, p_3, p_4, s_1, s_2, s_3, s_4, s_5, s_6]_{i\_arm}$, $[p_1, p_2, p_3, p_4, s_1, s_2, s_3, s_4, s_5, s_6]_{k\_arm}$,
$[p_1, p_2, p_3, p_4, s_1, s_2, s_3, s_4, s_5, s_6]_{j\_arm}$, $[p_1, p_2, p_3, p_4, s_1, s_2, s_3, s_4, s_5, s_6]_{m\_arm}$
The *j-arm* is similarly represented by the following feature vector, starting from *j_arm* and adding features from the other context arms in a clockwise order:
$[p_1, p_2, p_3, p_4, s_1, s_2, s_3, s_4, s_5, s_6]_{j\_arm}$, $[p_1, p_2, p_3, p_4, s_1, s_2, s_3, s_4, s_5, s_6]_{m\_arm}$,
$[p_1, p_2, p_3, p_4, s_1, s_2, s_3, s_4, s_5, s_6]_{i\_arm}$, $[p_1, p_2, p_3, p_4, s_1, s_2, s_3, s_4, s_5, s_6]_{k\_arm}$

For the hybrid model, three all-arm variants are investigated. An arm *i_arm* of an intersection *X* is defined under the three hybrid variant models as following:

1. Under the *hybrid-all static* model, all static features from all intersection arms of *X* are included in the feature vector, as well as the dynamic features of *i_arm*.
2. Under the *hybrid-all dynamic* model, all dynamic features from all arms of *X* are considered along with the static features of *i_arm*.
3. Under the *hybrid* model, all static and dynamic features from all arms of intersection *X* are included in the feature vector.

Of the existing dynamic and hybrid methodologies, to the authors' knowledge, none has considered such feature settings.

2.2.4. Testing Classification Performance under Various Trajectory Settings

The Effect of Sampling Rate

One observation looking at the three trajectory datasets is that they have different sampling rates: Champaign—1 Hz, Chicago—0.28 Hz (1 sample every 3.6 s), and Hanover—0.59 Hz (1 sample every 1 s). The sampling rate can affect both the calculated vehicle speed and the detection of stop and deceleration events, as described in Section 2.2.1. Obviously, the higher the sampling rate, the more accurate the speed calculation is and the more movement episodes are detected. To test this hypothesis, we conducted experiments on the two datasets with higher sampling rate, Champaign and Hanover, by undersampling the original datasets and comparing the performance on the undersampled datasets with that on the original datasets. The Champaign dataset was subsampled at ≈2 s, ≈3 s, and ≈4 s. The Hanover dataset was subsampled at ≈4 s.

The Effect of Turning/No-Turning Trajectories

Depending on the shape of the junction they are crossing, vehicles can go straight, turn left or right. This means that in the dataset we have straight or curved trajectories. The effect of turning at an intersection generally affects the driving behavior before and after the turn compared to a crossing by driving in a straight line, because the vehicle has to slow down before the turn and accelerate again after the turn. For this reason, other relevant studies have excluded curved trajectories from the dataset [24,27]. By excluding such trajectories, however, we reduce the data available to train and test the classifier. Therefore, we investigate the effect of using either all available trajectories (all combinations of right, left and straight trajectories) or exclusively straight trajectories on classification performance (Section 3.3).

The Effect of Number of Trajectories

We examine the minimal number of trajectories per junction arm that are needed to apply the proposed method (Section 2.2.2). In addition, we investigate whether there is an optimal number of trajectories per intersection arm with which the classifier performs best. That is, how many trajectories do we need to have in order for the extracted patterns to be sufficiently *descriptive* for classification purposes? On the one hand, by setting a minimum number of trajectories as a condition in order for a junction arm to be selected for training/testing, we shrink the dataset: the higher this number, the fewer junction arms satisfy the condition, as most junction arms have only few trajectories. On the other hand, summarizing the collective movement behavior using only a few trajectories can lead to an incorrect representation of the *actual* behavior of the movement. We address this aspect of the problem by conducting experiments on the minimum number n trajectories that a junction arm must have to be included in the training-test: (a) using all trajectories that cross the junction arm and (b) using exactly n trajectories to compute the patterns of a junction arm.

Thus, in (a), suppose we set the minimum number of trajectories to $min = 10$, we exclude from training and testing all junction arms crossed by less than 10 trajectories, and compute the patterns for the remained arms using *all* the trajectories that cross each of them. If an intersection arm has 35 trajectories, we compute the patterns based on all 35 traces. In (b), conversely, having excluded crossing arms with less than 10 traces, we compute the patterns using *exactly* 10 trajectories (we selected the most recent ones).

2.3. Domain Knowledge Rules

Table 3 shows the combinations of traffic regulators contained in each dataset. The Champaign dataset has 350 three-way junctions, 293 of which are controlled with UN/SS (UN-UN-SS), 33 are all-way UN, 15 are all-way SS, and 9 are all-way TS. Similar combinations are found in the Chicago dataset. In Hannover there are combinations of UN, PS, TS, SS, and YS. The classification, as explained previously, is implemented at a junction-arm level, so after classification each junction has a predicted label for each arm sampled from trajectories. As Table 3 shows, the regulators are not randomly combined with each other, but there are underlying *domain knowledge rules* [26] that can be used in a post-classification step to *correct* misclassified arms by comparing the predicted arm labels at a junction level. Hu et al. [24] use another domain knowledge rule regarding T-junctions (denote A, B, and C for left, right, and bottom ends of the junction); if A and B are UN, then C is SS controlled, otherwise C has the same type as A and B. Such simple knowledge rules have been used by [24,26] (see Section 1), but without investigating how much they contribute to overall accuracy, probably because both used only a single rule.

In defining the rules for domain knowledge, we also include the probability of the predictions, so that only predictions with high probability can be considered. We go one step further and compare the predictions of intersection arms belonging to the same intersection, both for correcting misclassified labels and for predicting labels for arms for which we have no data to make predictions. In the latter case, we make predictions for arms with missing data based on the predictions of the intersection arms of the same intersection for which we have data, e.g., for a three-way intersection, if one arm is predicted to be TS with high probability (>0.80), we can infer that the other two (unlabeled) arms are also TS.

Another rule states that if in a three-way intersection there are two predictions for two arms of an intersection, one TS with probability 0.95 and the other SS with probability 0.79, we can conclude that the SS prediction is wrong, and, therefore, we correct SS to TS and also label the third unlabeled arm of the intersection as TS, so that the intersection conforms to the domain knowledge rule that *if one arm is TS, then all other arms of the intersection are TS*. We use 0.15 as the correction threshold, i.e., the difference between the two predictions, e.g., here $0.95 - 0.79 = 0.16$, so that the correction of the lowest predicted label is triggered.

Similar consistency checks are performed for the other regulator combinations. Due to space constraints and the intuitive nature of the domain knowledge rules, we refrain from listing the other consistency checks.

Table 3. Combinations of traffic regulators at intersections. In Champaign and Chicago there are UN/SS controlled junctions, all-way UN controlled junctions, all-way SS, and all-way TS controlled junctions. In Hanover, there are UN/PS, YS/PS, all-way UN, and all-way TS controlled junctions.

Dataset	**Three-Way Junctions**								**Four-Way Junctions**							
	UN SS	**UN PS**	**YS PS**	**PS SS**	**UN (all)**	**SS (all)**	**TS (all)**	**Total**	**UN SS**	**UN PS**	**YS PS**	**PS SS**	**UN (all)**	**SS (all)**	**TS (all)**	**Total**
Champaign	293	0	0	0	33	15	9	350	220	0	0	0	9	52	80	361
Chicago	36	0	0	0	4	8	8	56	10	0	0	0	0	17	71	98
Hanover	0	230	386	5	0	0	88	709	0	0	82	9	94	0	153	338

2.4. Classification Settings

Two tree-based classifiers are used for the classification of the intersection arms: the Random Forest and the Gradient Boosting classifier. For the implementation we used the XGBoost library [47], which has recently dominated many Kaggle competitions. All programming tasks have been implemented in Python 3.

As default model feature settings, we regard the features extracted from straight trajectories (we exclude the trajectories that turn at junctions). Moreover, junction arms that are crossed with less that five trajectories, are excluded from training and testing.

3. Results

In this section, we list all the classification results of the experiments discussed in the previous Section 2.2. We first present the accuracy of the one-arm and all-arm models (Section 3.1). We then tune the best model and use it for all the other experiments regarding the effect of different trajectory settings on classification performance (Sections 3.3 and 3.4).

3.1. One-Arm vs. All-Arm Models

Table 4 shows the classification accuracy of all models. We can see that the Gradient Boosting (GB) classifier performs as good or better than Random Forest (RF) for almost all experiments. Only in the c-Dynamic model for the Chicago and Hannover dataset, RF performs slightly better than GB (+0.1 accuracy).

Table 4. Classification accuracy of the TRR models. The best performance per dataset is shown in bold.

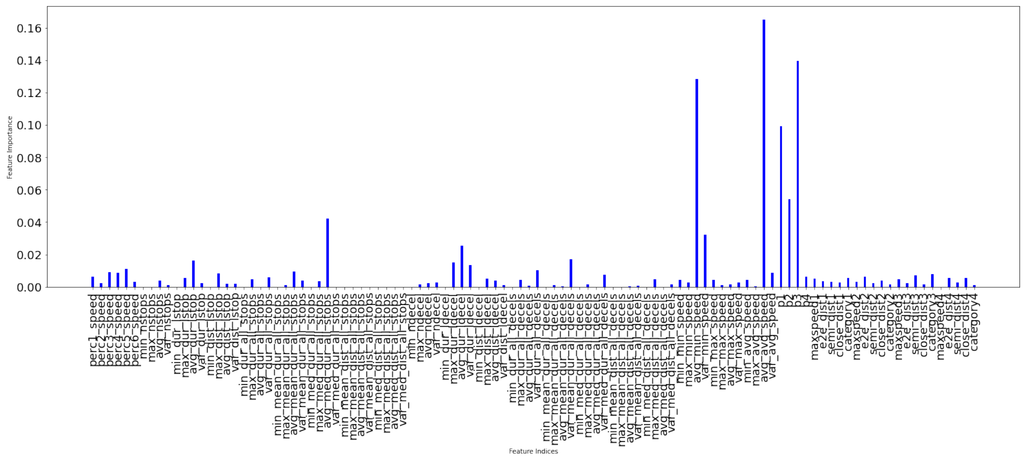
	Method	**Champaign**		**Chicago**		**Hanover**	
		RF	**GB**	**RF**	**GB**	**RF**	**GB**
One-arm	c-Dynamic	0.93	0.93	0.78	0.77	0.86	0.85
	Dynamic	0.94	0.94	0.81	0.81	0.86	0.87
	Static	0.67	0.69	0.72	0.72	0.61	0.62
	Hybrid	0.94	0.95	0.82	0.82	0.87	0.88
All-arm	c-Dynamic	0.94	0.94	0.78	0.78	0.89	0.90
	Dynamic	0.94	0.95	0.83	0.84	0.90	0.91
	Static	0.86	0.86	0.89	0.89	0.86	0.87
	Hybrid-all static *	0.94	**0.95**	0.88	**0.91**	0.91	**0.95**
	Hybrid-all dynamic **	**0.95**	**0.95**	0.82	0.86	0.91	0.93
	Hybrid ***	**0.95**	**0.95**	0.88	0.90	0.92	**0.95**

RF: Random Forest, GB: Gradient Boosting. * Only the dynamic features from one arm are included, along with the static features from all arms of the junction. ** Only the static features from one arm are included, together with the dynamic features from all arms of the junction. *** The model use the dynamic features from the adjacent arms and the static features of all the arms of the junction.

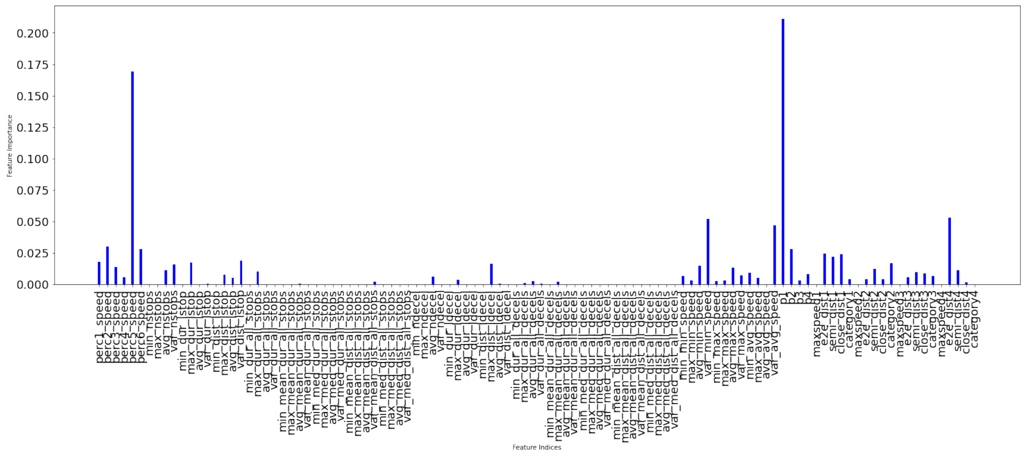
Comparing the one-arm models with each other, the static model has much lower accuracy than the other models for all datasets. The hybrid model has the best accuracy (0.95 in Champaign, 0.88 in Hanover, and 0.82 in Chicago) and the dynamic model performs better than the c-dynamic but worse than the hybrid.

With respect to the all-arm models, we observe that the static model performs much better than the one-arm model, but only for the Chicago dataset does it manage to outperform the c-dynamic. In all other experiments the other models have better accuracy. The c-dynamic model has lower accuracy than the dynamic model. For all datasets using the GB classifier, the hybrid-all-static model performs the same or better than the hybrid and hybrid-all-dynamic models and better than the c-dynamic, dynamic and static models. The same observation holds for the RF classifier, except for the Hanover dataset, where the hybrid model has an accuracy of 0.92 compared to the hybrid-all-static model with an accuracy of 0.91.

Therefore, the all-arm hybrid-all-static model with the GB classifier performs better for all datasets (0.95 in Champaign and Hanover and 0.91 in Chicago) and for this reason we select this model to use it for the experiments in the following sections. In addition, we performed feature selection and tuning of the classifier. In Figure 10 we provide plots showing the importance of the features. Interestingly, the most important features differ from dataset to dataset, even between the Champaign and Chicago datasets that share the same traffic regulator categories (UN, SS, TS). For example, in Champaign there are more important features related to deceleration compared to Chicago, while in Chicago the important features are more related to speed percentiles and map features. Common significant features for all datasets are the pattern features ($p1$, $p2$, $p3$, and $p4$). The classification results and confusion matrices for the three datasets after feature selection and tuning are presented in Table 5 and Figure 11.

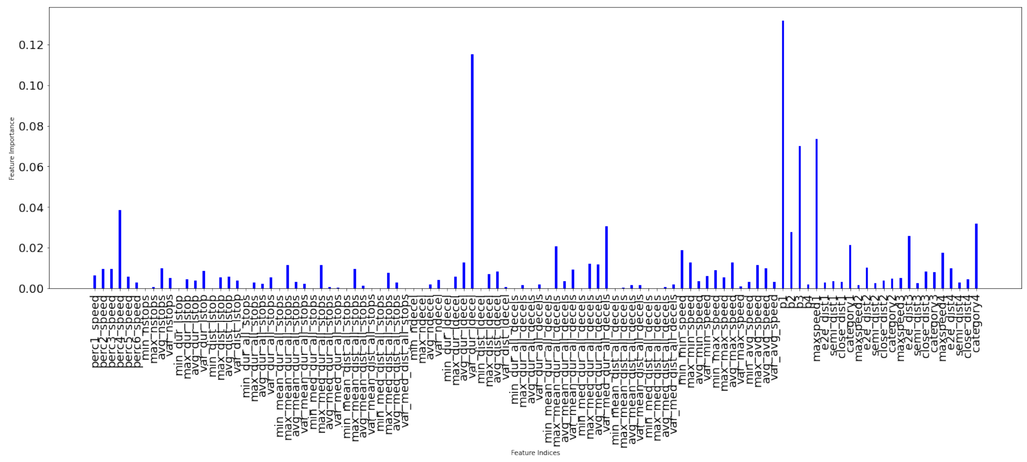


(**a**) Champaign



(**b**) Chicago

Figure 10. *Cont.*



(**c**) Hanover

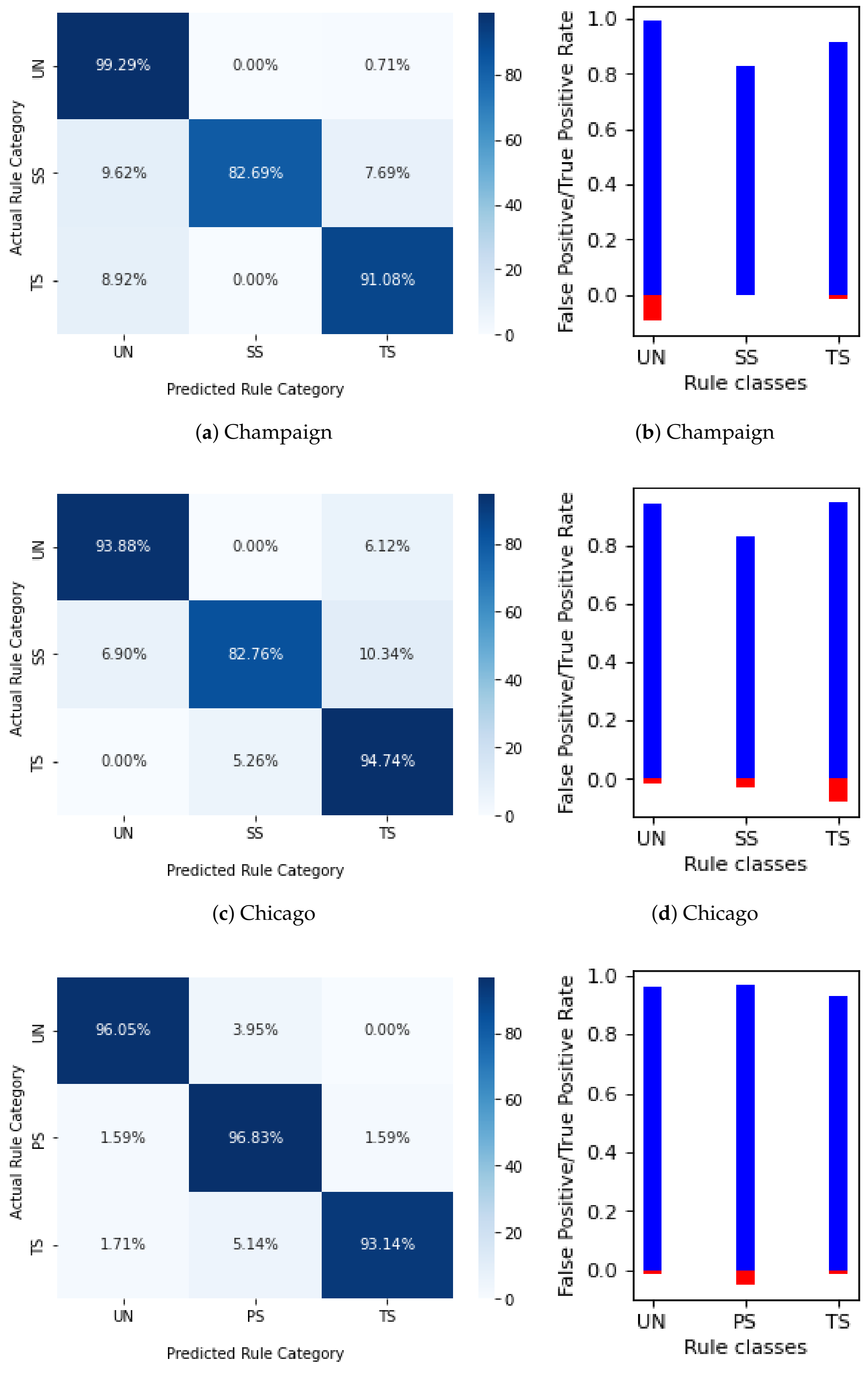
Figure 10. Feature importance for the the three datasets.

Table 5. Classification results of the *tuned hybrid all-static* model.

Dataset	Classifier	Label	Recall	Precision	F-Measure	Accuracy	Support
Champaign	GB	UN	0.99	0.96	0.97		424
		SS	0.83	1.0	0.90		52
		TS	0.91	0.95	0.93		157
		W.Avg.	0.96	0.96	0.96		633
						0.96	
Chicago	GB	UN	0.94	0.97	0.95		49
		SS	0.83	0.86	0.84		29
		TS	0.95	0.93	0.94		76
		W.Avg.	0.92	0.93	0.92		154
						0.92	
Hanover	GB	UN	0.96	0.91	0.93		76
		PS	0.97	0.96	0.97		315
		TS	0.93	0.97	0.95		175
		W.Avg.	0.96	0.96	0.96		566
						0.96	

As we can see in Table 5, feature selection and classifier tuning increased the accuracy by 1%, from 0.95 to 0.96 for the Champaign and Hanover datasets, and from 0.91 to 0.92 for the Chicago dataset. In Champaign and Chicago, the stop sign (SS) category is predicted slightly worse than the other two categories (F-Measure in Champaign: 0.90 (SS), 0.97 (UN), and 0.93 (TS), and in Chicago: 0.84 (SS), 0.95 (UN), and 0.94 (TS)). In Hanover, the per-class F-Measures are similar for the three classes. This observation is highlighted in the confusion matrices in Figure 11, which visually depicts the actual versus predicted classes. In the same Figure, there are also graphs of the false positive rate (FPR) and true positive rate (TPR). We can see in Figure 11b,d,f that the highest FPRs in the three datasets are observed in different classes: UN in Champaign (0.09), TS in Chicago (0.077), and PS in Hanover (0.048). Additionally, the highest TPRs are observed in the same classes as the highest FPRs: Champaign: 0.99 (UN), Chicago: 0.93 (TS), and Hanover: 0.97 (PS).

The lower performance in Chicago (accuracy of 0.92) compared to Champaign and Hanover (0.96 and 0.96) can be explained by the fact that the Chicago dataset is significantly smaller than the other datasets (154 regulators versus 633 (Champaign) and 566 (Hanover)), which limits the training possibilities. In addition, as already mentioned in Section 2.1.2, the sampling rate in Chicago is lower than the other two datasets (average 0.28 Hz vs. 1 Hz and 0.59 Hz), which may affect the computation of the feature calculation (short-term detected events).



(**a**) Champaign (**b**) Champaign

(**c**) Chicago (**d**) Chicago

(**e**) Hanover (**f**) Hanover

Figure 11. Confusion matrices and false/true positive rates for the three datasets.

3.2. Testing the Effect of Sampling Rate

Table 6 shows the classification performance of TRR methods at different sampling rates. On the Champaign dataset, we see that the performance between 1 s and 2 s either decreases by about 1–2% or stays the same. Between 2 s and 3 s, the accuracy remains the same in the majority of TRR methods and there are two cases where the performance varies by ±1% (all-arm dynamic models and hybrid all-static models). The drop in accuracy is largest between 3 s and 4 s, where the difference varies between 1% and 3%. Regarding the detected stop and deceleration episodes, it seems that they are affected by the sampling rate: the higher the sampling rate, the more events are detected. In the Hanover dataset, between 2 s and 4 s the accuracy either drops by about 1% or remains the same. When we compare the performance between Champaign and Hanover, between 2 s and 4 s, we see that in Champaign there is a decrease of between 2% and 3%, while in Hanover there is a decrease of 0–1%. When we compare the performance between Champaign and Chicago at 4 s, we see a difference in accuracy between 2% and 12%, with the smallest difference seen in the hybrid all-static model.

Table 6. Classification accuracy of TRR methods under different sampling rates (undersampling). The original datasets are highlighted in grey.

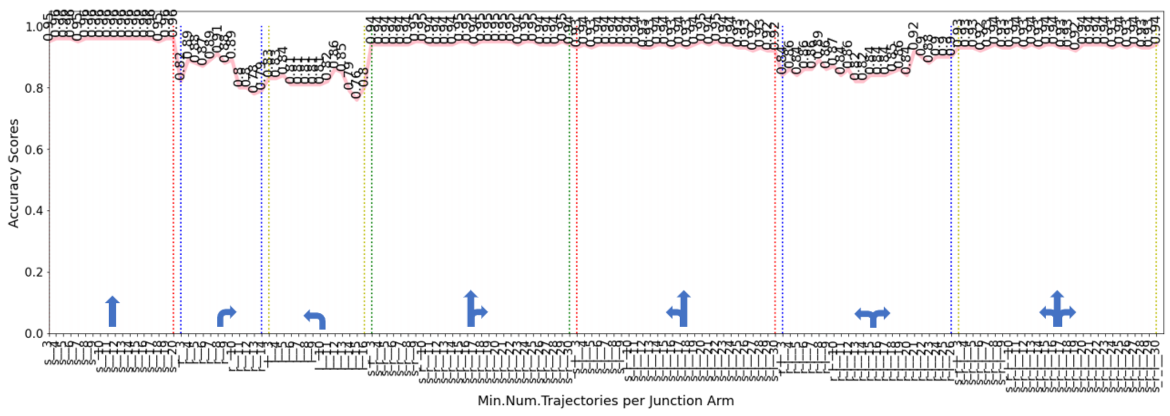
		Champaign				**Chicago**	**Hanover**	
		≈1 s	**≈2 s**	**≈3 s**	**≈4 s**	**≈4 s**	**≈2 s**	**≈4 s**
	Stop episodes	112,401	64,767	40,491	28,541	11,015	161,436	90,315
	Decel. episodes	100,853	42,454	21,897	18,471	5589	116,349	55,487
One-arm	c-Dynamic	0.93	0.91	0.91	0.88	0.77	0.85	0.84
	Dynamic	0.94	0.93	0.93	0.91	0.81	0.87	0.87
	Hybrid	0.95	0.94	0.94	0.92	0.82	0.88	0.88
All-arm	c-Dynamic	0.94	0.93	0.93	0.90	0.78	0.90	0.89
	Dynamic	0.95	0.94	0.95	0.92	0.84	0.91	0.91
	Hyb. (all-static)	0.95	0.95	0.94	0.93	0.91	0.95	0.94

A general conclusion from these experiments is that sampling rate can affect classification performance: in the Champaign dataset, when 1 s and 4 s were compared, no method remained unaffected by subsampling. However, the decrease in performance is not large enough to explain why the accuracy in Chicago differs so much from that in Champaign (4 s) (2–12%). If the sampling rate was the only reason for the lower performance in the Chicago dataset, then Champaign in 4 s would have similar performance to Chicago, which is not the case. As mentioned previously, perhaps the lower performance in Chicago is due to the size of the dataset (number of regulators), which affects the training of the classifier.

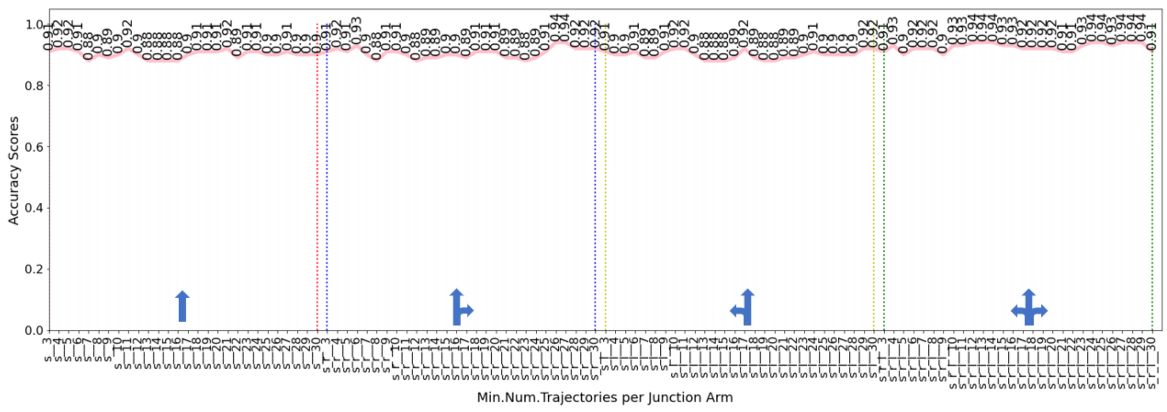
3.3. *Testing the Effect of Turning Trajectories*

In Figure 12, we report the classification performance for the three datasets under different traversal settings: crossing direction and number of trajectories per intersection arm. In the first case, we examine whether considering samples moving only straight ahead positively affects classification, assuming that turning behavior affects speed, so excluding curved trajectories can eliminate their bias. In the second case, we seek whether there is an optimal number of trajectories that an intersection arm should have during training and, thus, exclude from the training dataset intersections with fewer trajectories than this number.

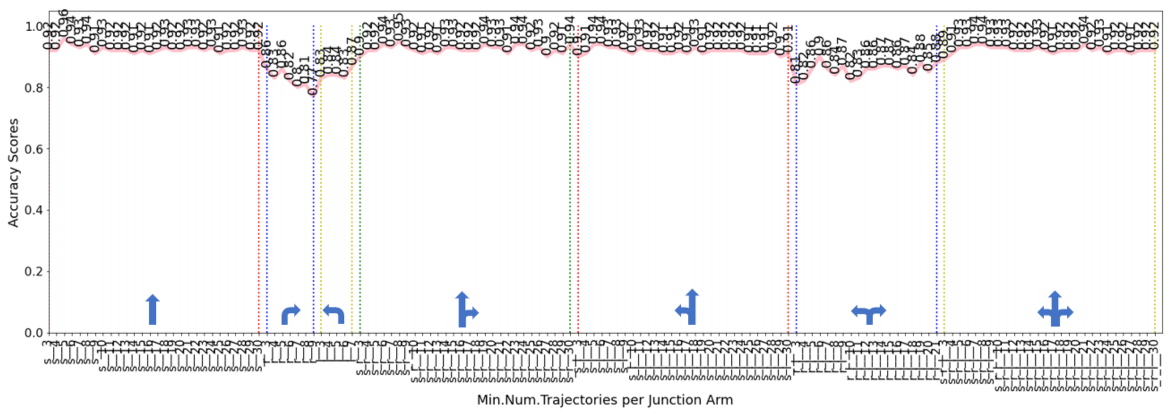
We looked at all possible turning settings and their combinations. Regarding the labels on the horizontal axis of Figure 12, *s_* refers only to straight trajectories, *r_* to trajectories that turn right, *l_* to trajectories that turn left, *s_r_* refers to straight/right turn, *s_l_* to straight/left turn, *r_l_* to right/left turn, and *s_r_l_* to straight/right/left trajectories. The number after these prefixes refers to the number n of trajectories used to select the intersection arms (minimum number of trajectories per intersection arm) and to calculate the motion patterns (i.e., motion patterns observed on an intersection arm are calculated by summarizing the behavior of at least n trajectories that cross it). Not all turning settings are tested with the same number of trajectories, because for each turning/crossing setting, we require the test dataset to contain at least 7 junction arms per class. For example, in the Champaign dataset (Figure 12a), we tested the straight trajectories for various numbers 3, 4,. . . , 20, because for minimum number of trajectories equal to 21, the number of junction arms in the test set (10-fold cross-validation) could not contain more than 7 stop controlled (SS) junction arms.



(**a**) Champaign.



(**b**) Chicago.



(**c**) Hanover.

Figure 12. Experiments with different turning settings (s_: straight trajectories, r_: right turning trajectories, l_: left turning, s_r_: straight and right turning, s_l_: straight and left turning, r_l_: right and left turning, s_r_l: straight, right, and left turning trajectories).

Regarding the effect of turning trajectories on classification performance, we see that using right, left or right/left traces has lower performance than using straight traces and combinations of straight and turning traces (Figure 12a,c). When using a combination of straight and turning traces, we cannot see a strong negative effect, but this can perhaps be explained by the fact that there are significantly more straight crossings than left and right in the dataset (in Champaign 20,514 straight, 2619 right, and 2768 left;

in Hanover 19,092 straight, 3394 right, and 3073 left; in Chicago 12,638 straight, 2820 right, and 1301 left).

A possible explanation for the poor classification performance when using exclusively turning trajectories (e.g., in Champaign 0.88 accuracy for right turning trajectories when junction arms have at least 5 trajectories—see r_5 at Figure 12a), is the smaller dataset used for training compared to the other settings. Table 7 shows the number of train/test junction arms per control type in Champaign dataset, when each arm has at least five trajectories. We see that the training sets when only right, left, or right/left trajectories are much smaller than the set with straight trajectories. Additionally, the fact that features are computed from a small number of trajectories (as the number of right and left trajectories per junction arm is small), may also explain the bad performance. Therefore, one reason for the poor performance may be related to the dataset itself (which affects the feature computation and the size of the training dataset) and not solely to the condition we consider here (drive straight through an intersection or turn). Perhaps the same analysis on a dataset with numerically more junction arms sampled from a higher density would not show a negative effect of turning trajectories.

Table 7. Dataset size for different trajectory direction settings with minimum number of trajectories per junction arm equal to 5 (Champaign dataset).

Trajectory Direction		Rule	Junction Arms	Classification Accuracy
Straight		UN	424	
		SS	52	0.96
		TS	157	
	Sum		633	
Right		UN	26	
		SS	29	0.88
		TS	59	
	Sum		114	
Left		UN	36	
		SS	18	0.81
		TS	48	
	Sum		102	
Right/Left		UN	71	
		SS	59	0.84
		TS	108	
	Sum		238	
All		UN	457	
		SS	100	0.93
		TS	192	
	Sum		749	

Another observation is that on the Champaign dataset the performance when only straight trajectories are used is better than when straight and turning trajectories are used. For example, with at least 5 straight trajectories the accuracy is 96% and with at least 5 straight/turning trajectories the accuracy is 93% (Figure 12a). On the Hanover dataset (Figure 12c), on average the performance when using only straight trajectories is better than when using straight and turning trajectories, but the effect is less strong than in then Champaign dataset. For both datasets, the difference in accuracy between using straight and all trajectories (straight and curved) is between 1% and 3%. Therefore, in both datasets, excluding curved trajectories has a positive effect on classification performance and the optimal number of straight trajectories is 5.

However, in the Chicago dataset the same observation does not hold, i.e., with at least 15 straight trajectories the accuracy is 88% and with at least 15 straight/turning trajectories the accuracy is 94% (Figure 12b). Although there are not a sufficient number of crossing arms crossed by turning trajectories for training and testing as in the other two datasets, a possible explanation for the slightly increased performance when all trajectories are used is that the training dataset becomes larger when straight/turning trajectories are used than when only straight trajectories are used, so the classifier learns better. For example, when at least 5 straight trajectories are used, the dataset contains 49 UN, 29 SS, and 76 TS, while when at least 5 tracks (turning and straight tracks) are used the dataset contains 50 UN, 40 SS, and 115 TS.

Therefore, judging from the two larger datasets that have consistent results, we can conclude that curved tracks at intersections affect the classification by about 1–3% in accuracy and, therefore, for the next experiments we will only use straight tracks (minimum number of tracks per junction arm equal to 5), excluding all curved tracks at intersections, similar to what Hu et al. [24,27] do in their studies.

3.4. Testing the Effect of Number of Trajectories on Classification Performance

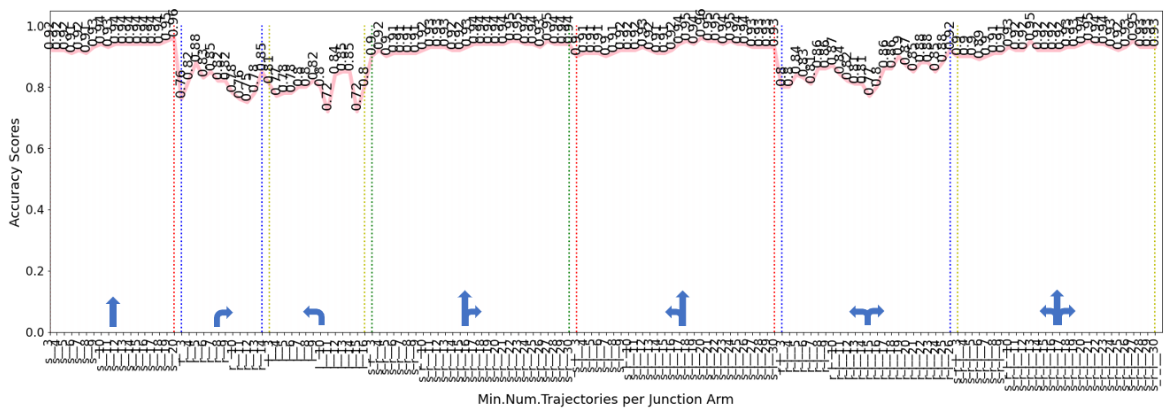
The number of trajectories used as a minimum requirement for the calculation of classification features, as well as for the selection of junction arms for training and testing, seems to affect performance. Comparing the case of using a *certain* number of trajectories, i.e., a subset of all trajectories crossing a junction arm, with the case of using *all* available trajectories per junction arm (as explained in The Effect of Number of Trajectories), the latter case achieves better performance on average. This result holds for all datasets (compare Figure 12 with Figure 13) and seems reasonable, as the more trajectories are used to compute the classification features (which are statistical values of physical features, as explained in Section 2.2.2), the better the latter reflect the actual movement behavior.

On the Champaign dataset, we see from Figure 13a that with just 3 straight trajectories per junction arm, we can achieve 92% accuracy. Increasing the number of trajectories also increases the classification performance. The best result, 96%, is achieved with 20 trajectories. However, when we compare the results with those in Figure 12a, we see that limiting the number of trajectories per arm yields lower classification performance, e.g., with *at least* 4 straight trajectories per junction arm the accuracy is 96%.

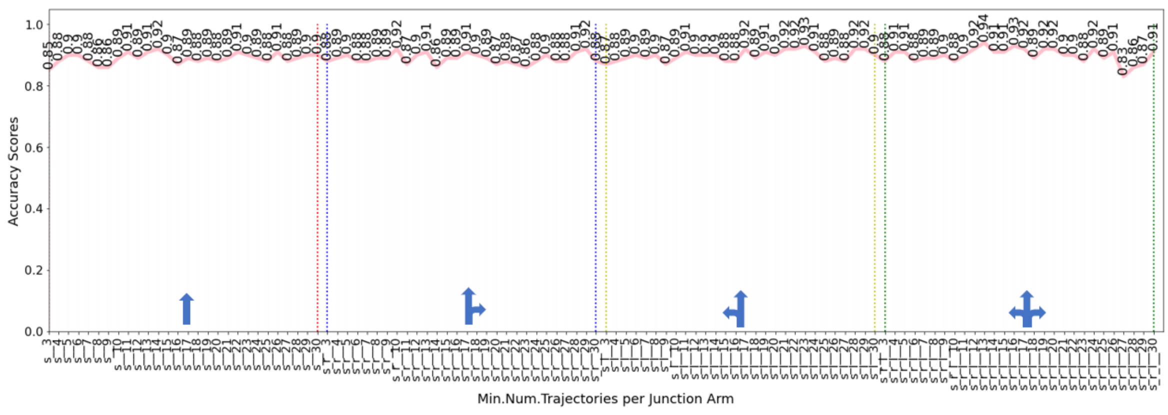
In the Hanover dataset, we can also see from Figure 12c that when all available traces are used the performance is better than using a certain number (Figure 13c). For accuracy above 91%, at least 7 traces per junction arm are required (Figure 13c). The best accuracy (93%) is achieved with 9 trajectories. In contrast, by allowing the use of all available trajectories per arm, with at least 3 traces per arm, the accuracy is always equal to or greater than 91%, and the best accuracy of 96% is achieved with at least 5 trajectories (Figure 12c).

On the Chicago dataset, the same result is also observed with the other two datasets. Using a *certain* number of trajectories per arm gives a lower classification accuracy than when all available trajectories are used (Figure 13b vs. Figure 12b). However, with only 3 straight trajectories per arm the accuracy is always equal to or greater than 85% and with only 4 straight trajectories equal to or greater than 86% (Figure 13b).

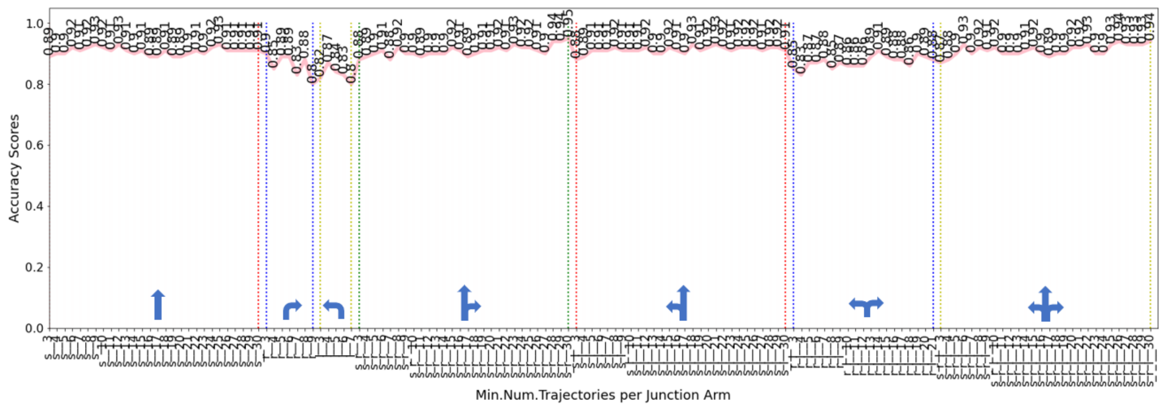
Therefore, for feature computation, excluding the number of trajectories to a certain number negatively affects the classification performance. However, with only 3 straight trajectories per junction arm, the classification accuracy is equal to or greater than 85% across all datasets (85% in Chicago, 89% in Hanover, and 92% in Champaign). Additionally, with only 5 straight trajectories the accuracy is equal to or greater than 90% (90% in Chicago and 92% in Champaign and Hanover).



(**a**) Champaign.



(**b**) Chicago.



(**c**) Hanover.

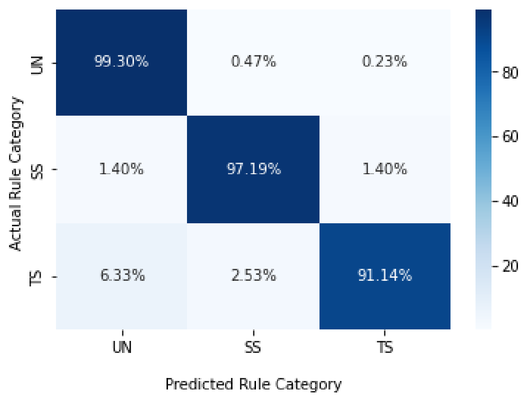
Figure 13. Experiments with different number of trajectories where classification features are computed using a *certain* number of trajectories, i.e., 3, 4, . . . , and not all available crossing trajectories.

3.5. Application of Domain Knowledge Rules

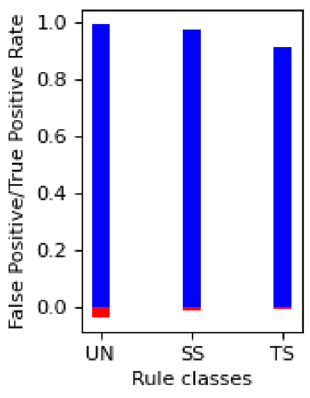
Table 8 shows the classification report after applying the domain knowledge rules consistency check, as explained in Section 2.3. Figure 14 also shows the confusion matrices along with the FPR/TPR graphs for the three datasets. The first observation from Table 8 is that accuracy increases by 1% in Champaign and Hanover (from 96% to 97%) and by 3% in Chicago (from 92% to 94%). Additionally, by comparing the number of predictions (compare Support column in Tables 5 and 8), we can see that in Champaign we have 315 predictions from arms with missing data, which means that not only were predictions made that are equal to 50% of the original dataset with no data (original dataset: 633 arms), but these predictions are correct (accuracy increased to 1%). In Chicago, 47 predictions were made on arms with missing data, corresponding to 30.5% of the original dataset. Similarly, in Hanover, 152 regulators were predicted from arms with missing data, corresponding to 27% of the original dataset.

Table 8. Classification results of the *tuned hybrid all-static* model after applying a consistency check of domain knowledge rules.

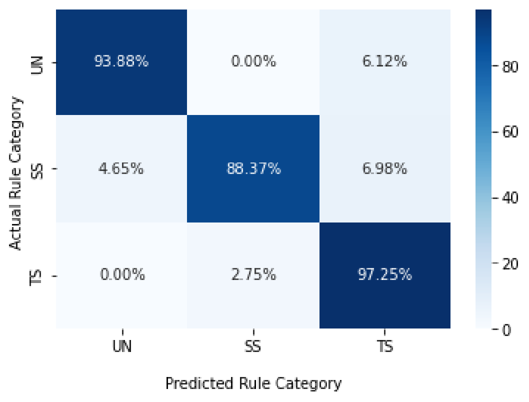
Dataset	Classifier	Label	Recall	Precision	F-Measure	Accuracy	Support
Champaign	GB	UN	0.96	0.99	0.97		426
		SS	0.97	0.97	0.97		285
		TS	0.98	0.91	0.94		237
		W.Avg.	0.97	0.97	0.97		948
						0.97	
Chicago	GB	UN	0.96	0.94	0.95		49
		SS	0.93	0.88	0.90		43
		TS	0.95	0.97	0.96		109
		W.Avg.	0.94	0.94	0.94		201
						0.95	
Hanover	GB	UN	0.94	0.98	0.96		121
		PS	0.97	0.98	0.98		315
		TS	0.99	0.97	0.98		282
		W.Avg.	0.98	0.97	0.98		718
						0.97	



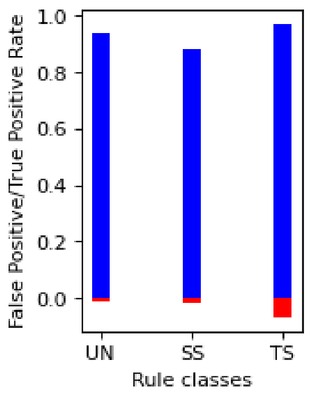
(**a**) Champaign



(**b**) Champaign

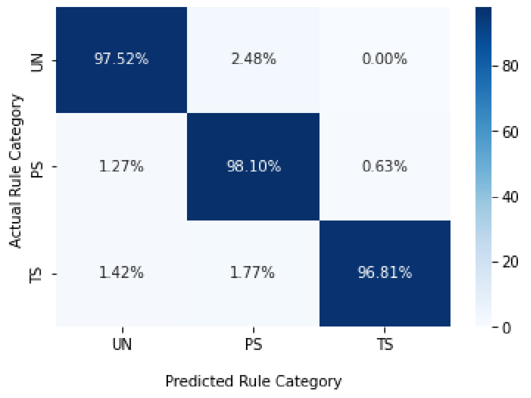


(**c**) Chicago

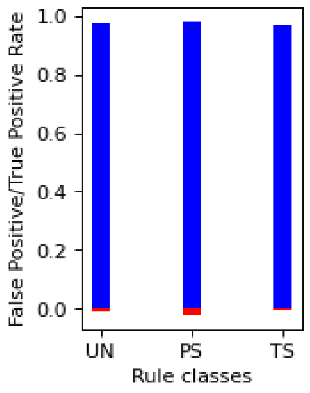


(**d**) Chicago

Figure 14. *Cont.*



(**e**) Hanover



(**f**) Hanover

Figure 14. Confusion matrices and false/true positive rates for the three datasets after applying consistency check using domain knowledge rules.

In terms of FPRs, in Champaign the FPRs for UN, SS, and TS are: 3.6%, 1.2%, 0.7%, respectively, and in Chicago 1.3%, 1.9%, and 6.5%. In Hanover the FPRs for UN, SS, and TS are 1.3%, 2%, and 0.5%. Compared to the FPRs of Figure 11, in Champaign the FPR for UN decreased from 9% to 3.6% (60% decrease), in Chicago the FPR for TS decreased from 7.7% to 6.5% (15.6%), and for SS decreased from 3.2% to 1.8% (44% decrease), and in Hanover the FPR for PS decreased from 4.8% to 2% (58.3% decrease). Moreover, the average FPR (TPR) across regulator classes decreases (increases) in Champaign from 3.5% (90.5%) to 1.8% (96%), in Chicago from 4.3% (90.5%) to 3.2% (93.2%), and in Hanover from 2.6% (95.3%) to 1.3% (97.5%).

Therefore, by applying domain knowledge rules, there is a gain in accuracy between 1% and 3%, but more importantly, accurate predictions can be made for arms with incomplete data corresponding to 27–50% of the original data. Furthermore, the FPR of the class with the highest FPR decreases between 15.6% and 60% (15.6% in Chicago, 60% in Champaign, and 58.3% in Hanover), validating our proposal to use domain knowledge rules for both recovering misclassified arms and predicting regulators for arms without data.

4. Discussion

The main findings of this research paper are the following:

1. The traffic rule recognition method proposed in this paper, which combines data from trajectories and OSM, can provide accurate results for rule sets consisting of controlled intersections of UN, SS, PS, and UN: 97% accuracy in Champaign and Hanover, and 95% in the smaller Chicago dataset.
2. Including information in the feature vector from context intersection arms proved beneficial for classification: all-arm models outperformed single-arm models.
3. The sampling rate can affect the performance of the classification. A low value affects both the calculated speed values from GPS traces and the detected stopping and deceleration episodes. The accuracy of the hybrid-all-static model decreased between 1% and 2% when the sampling interval was doubled from 2 s to 4 s.
4. The negative effect on accuracy, as validated by the two larger datasets, when both straight and curved trajectories are used, was found to be between 1% and 3%. Therefore, the exclusion of curved trajectories has a positive effect on classification performance.
5. The optimal number of straight tracks is five.
6. Excluding the number of tracks to a certain number negatively affects the classification performance. However, with only three straight trajectories per junction arm, classification accuracy is equal to or greater than 85% across all datasets (85% in Chicago,

89% in Hanover, and 92% in Champaign). With only five straight trajectories, accuracy is equal to or greater than 90% (90% in Chicago and 92% in Champaign and Hanover).

7. By applying domain knowledge rules, there is a gain in accuracy between 1% and 3%, but more importantly, accurate predictions can be made for arms with no data, corresponding to 27–50% of the original data. Moreover, the FPR of the class with the highest FPR decreases between 15.6% and 60%, so our proposal to use domain knowledge rules for both recovering misclassified arms and for predicting regulators from arms without data, is validated.

We find it more interesting to extend this study to how classification can predict rules with high accuracy on limited labeled data. As discussed in Section 2.1, labeling intersections is a time-consuming task, and although there are many trajectory datasets that one can use for TRR, the need for labeled data for training and testing makes the latter unsuitable for this purpose. One idea is to explore unsupervised methods where no labelled data are required. A second idea is to explore semi-supervised methods, such as label propagation and self-learning, where only limited labeled data are used to train a classifier. A third idea is to explore the possibility of transferring learning from one city to another, i.e., training a classifier with labeled data from a city X and testing in a city Y, assuming no labeled data from the latter city.

In addition, an important aspect of the problem that needs to be considered is whether the proposed approach would perform equally well in smaller cities, where driving behavior is influenced by factors other than traffic regulations, e.g., pedestrians who, knowing that vehicles are moving at low speed, cross intersections more freely.

Moreover, it would be interesting to test the performance of the proposed hybrid model with missing OSM data. In the three datasets we tested, very often speed-related data were missing from OSM, but the performance remained high. It would be interesting to investigate under which conditions of missing data performance would be affected to the extent that, for example, the dynamic model would be preferable to the hybrid model.

Another interesting topic to investigate is whether the number of required trajectories differs between locations controlled by the same regulation. For example, what is the variation in the required number of trajectories for intersections regulated by a stop signal? Such an analysis can be performed by finding the minimum required number of trajectories that predict the regulation with high probability at the intersection arm level (separately for each arm) and then finding the variation in the number of trajectories within intersection arms of the same regulation. In this paper, we found the minimum number of trajectories (five) by applying the same analysis to *all* arms, without finding the optimal number of trajectories per intersection arm. With this recommended analysis, we could identify intersection features that make certain locations (and perhaps classes of regulations) easier to classify, regarding their traffic control.

Finally, the way in which the variability of trajectory densities affects the classification would be another parameter of the TRR problem that deserves further investigation. Since not all intersections attract the same traffic, the datasets are irregular, e.g., one section of a city is sampled from dozens of trajectories and others from only a few tracks. This aspect is not taken currently into account when splitting the datasets into training and test sets. Perhaps splitting the dataset taking this aspect into account would provide even better results.

In conclusion, this study has shown the importance of using low-cost crowd-sourced information for the task of identifying traffic regulators, the cost of which in terms of time and money is much higher if instead standard technology is used for surveying. Trajectories can reveal the movement patterns of drivers, and our hypothesis that traffic rules can be retrieved in a reverse-engineering fashion by mining the movement patterns imposed by traffic rules has been verified. The predictive ability of the classifier becomes more accurate when static information from the road network (OSM) is merged with dynamic features extracted from the trajectories. This finding motivates the idea of crowdsourcing more traffic rules that can be added to maps and exploited by location-aware applications, which generally try to optimize transportation or offer an optimal way to reach a location B starting from a location A based on personalized criteria, e.g., avoiding tolls, highways, etc. For example, [48] go one step further than existing personalized navigation and route finding services and propose the idea of providing routing suggestions to drivers avoiding *complicated crossings* in urban areas. A complicated crossing can be an intersection where the road network is intersected by bike lanes and tram tracks. Similarly, a complex crossing may be a left turn at an intersection that is not controlled by a traffic light. Conversely, an easy intersection may be a controlled intersection with stop sign everywhere. Such recommendations may be useful to inexperienced drivers or drivers who for whatever reason are not comfortable driving in an urban environment. Traffic controls could also be explored in the context of intersection complexity and incorporated into the intersection complexity assessment for personalized route recommendations. Therefore, the results of this study can be encouraging for further research aimed at benefiting the citizens of smart cities.

5. Conclusions

In this paper, a new method for identifying traffic regulations from GPS trajectories is proposed. A modification of a well-known clustering algorithm for detecting stopping and deceleration events was presented. By detecting such driving events, we categorize intersections into four traffic classes, which, together with other statistical values of the detected events and the average vehicle traverse speed at these locations, describe the driving behavior at the regulated locations (*dynamic* classification model). Mixing into the dynamic model static features extracted from the OSM (*static* model), leads to a *hybrid* model that was shown to have better classification performance than the other two models. For each of the three models, two variants of the feature vector were tested, one where only features associated with a single junction arm are used (*one-arm* model) and another where features from neighboring junction arms of the same junction are used to classify one arm (*all-arm* model). The hybrid all-arm model provided the best classification accuracy on the three datasets used to test the methodology, 94% on the smallest dataset and 97% on the other two datasets. The minimum optimal number of trajectories crossing the intersections was found to be five (straight trajectories). The exclusion of curved trajectories from the feature calculation was found to have a positive effect on classification performance. Finally, by applying a set of domain knowledge rules to the predicted labels, we were able to both recover misclassified intersection arms and predict labels from arms with no data, corresponding to 27–50% of the original dataset, while further increasing classification accuracy by 1–3%. New research directions were proposed based on the limitations of the study, discussing ideas that can better clarify these issues.

Author Contributions: Conceptualization, Stefania Zourlidou, Monika Sester, and Shaohan Hu; methodology, Stefania Zourlidou; software, Stefania Zourlidou; validation, Stefania Zourlidou and Monika Sester; formal analysis, Stefania Zourlidou; investigation, Stefania Zourlidou; resources, Stefania Zourlidou and Shaohan Hu; data curation, Stefania Zourlidou and Shaohan Hu; writing—original draft preparation, Stefania Zourlidou; writing—review and editing, Stefania Zourlidou and Monika Sester; visualization, Stefania Zourlidou; supervision, Monika Sester; project administration, Stefania Zourlidou and Monika Sester; funding acquisition, Monika Sester. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the German Research Foundation (Deutsche Forschungsgemeinschaft (DFG)) with grant number 227198829/GRK1931. The publication of this article was funded by the Open Access Fund of the Leibniz Universität Hannover.

Data Availability Statement: The groundtruth map from the Chicago dataset, is accessible in a public repository [34] and has an open data licence. The Chicago trajectory dataset, as already mentioned, is publicly available in [33]. The Hannover dataset (trajectories and groundtruth regulations) is also available in [31,32].

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data, in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

TRR	Traffic Regulation Recognition
OSM	OpenStreetMap
GPS	Global Positioning System
SC	Spatial Crowdsourcing
CC	Crowdsourcing
DLSTM	Distributed Long Short Term Memory
ROC	Receiver Operating Characteristic
AUC	Area Under the ROC Curve
CVAE	Conditional Variational Autoencoder
TRR	Traffic Regulator Recognition
TS	Traffic Signals
SS	Stop Sign
PS	Priority Sign
YS	Yield Sign
RF	Random Forest
GB	Gradient Boosting
FPR	False Positive Rate
TPR	True Positive Rate

Appendix A

Algorithm A1: The CB-SDoT algorithm: Clustering-based Stop and Deceleration Event Detection in Trajectories.

```
Data:
$T$: set of GPS trajectories
$Eps$: interpoint distance
$minTime$: minimum time
$maxTime$: maximum time
Result: CB-SDoT identifies clusters of points that within a certain distance $Eps$
        remain at least $minTime$ and no more than $maxTime$.
Returns: for each cluster with cluster_id, the sequence of points of the cluster
  SeqPoints, the point representative RepCluster of the cluster and the duration
  Dur of the detected event

initialize clusters to an empty list
initialize all points of $T$ as unprocessed
```

Algorithm A1: *Cont.*

```
for each trajectory $t$ in $T$ do
    for each unprocessed point $p$ in $t$ do
        // find the neighbors of $p$
        $neighbor\_list$ = linear_neighborhood($p$, $Eps$)
        if $p$ is a core point wrt $Eps$, $minTime$, $maxTime$ then
            for each neighbor $n$ in $neighbor\_list$ do
                $N\_neighbor\_list$ = linear_neighborhood($n$, $Eps$)
                $neighbor\_list = neighbor\_list \cup N\_neighbor\_list$
            end
            add $neighbor\_list$ as cluster with $cluster\_id$ in $clusters$
            find the $RepCluster$ of the cluster compute the $Dur$ of the cluster
            set all points in $neighbor\_list$ as processed
        end
    end
end
```

References

1. Goodchild, M. Citizens as sensors: The world of volunteered geography. *GeoJournal* **2007**, *69*, 211–221. [CrossRef]
2. Gummidi, S.R.B.; Xie, X.; Pedersen, T.B. A Survey of Spatial Crowdsourcing. *ACM Trans. Database Syst.* **2019**, *44*, 8. [CrossRef]
3. Heipke, C. Crowdsourcing geospatial data. *ISPRS J. Photogramm. Remote Sens.* **2010**, *65*, 550–557. [CrossRef]
4. Tang, J.; Deng, M.; Huang, J.; Liu, H.; Chen, X. An Automatic Method for Detection and Update of Additive Changes in Road Network with GPS Trajectory Data. *ISPRS Int. J. Geo-Inf.* **2019**, *8*, 411. [CrossRef]
5. Shan, Z.; Wu, H.; Sun, W.; Zheng, B. COBWEB: A Robust Map Update System Using GPS Trajectories. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*; UbiComp '15; Association for Computing Machinery: New York, NY, USA, 2015; pp. 927–937. [CrossRef]
6. Fox, A.; Kumar, B.V.; Chen, J.; Bai, F. Multi-lane pothole detection from crowdsourced undersampled vehicle sensor data. *IEEE Trans. Mob. Comput.* **2017**, *16*, 3417–3430. [CrossRef]
7. Wage, O.; Sester, M. Joint Estimation of Road Roughness from Crowd-Sourced Bicycle Acceleration Measurements. *ISPRS Ann. Photogramm. Remote. Sens. Spat. Inf. Sci.* **2021**, *V-4-2021*, 89–96. [CrossRef]
8. Vij, D.; Aggarwal, N. Smartphone based traffic state detection using acoustic analysis and crowdsourcing. *Appl. Acoust.* **2018**, *138*, 80–91. [CrossRef]
9. Minson, S.E.; Brooks, B.A.; Glennie, C.L.; Murray, J.R.; Langbein, J.O.; Owen, S.E.; Heaton, T.H.; Iannucci, R.A.; Hauser, D.L. Crowdsourced earthquake early warning. *Sci. Adv.* **2015**, *1*, 36. [CrossRef]
10. Salpietro, R.; Bedogni, L.; Di Felice, M.; Bononi, L. Park Here! A smart parking system based on smartphones' embedded sensors and short range Communication Technologies. In Proceedings of the 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT), Milan, Italy, 14–16 December 2015; pp. 18–23.
11. Zhou, X.; Zhang, L. Crowdsourcing functions of the living city from Twitter and Foursquare data. *Cartogr. Geogr. Inf. Sci.* **2016**, *43*, 393–404. [CrossRef]
12. Gao, R.; Sun, F.; Xing, W.; Tao, D.; Fang, J.; Chai, H. CTTE: Customized Travel Time Estimation via Mobile Crowdsensing. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 19335–19347. [CrossRef]
13. Lefevre, S.; Laugier, C.; Ibanez-Guzman, J.; Bessiere, P. Modelling Dynamic Scenes at Unsignalised Road Intersections. *Inria Res. Rep.* **2011**, *RR-7604*.
14. Lefèvre, S.; Laugier, C.; Ibañez-Guzmán, J. Risk assessment at road intersections: Comparing intention and expectation. In Proceedings of the Intelligent Vehicles Symposium (IV), 2012 IEEE, Madrid, Spain, 3–7 June 2012; pp. 165–171. [CrossRef]
15. Alshayeb, S.; Stevanovic, A.; Effinger, J.R. Investigating impacts of various operational conditions on fuel consumption and stop penalty at signalized intersections. *Int. J. Transp. Sci. Technol.* **2021**, *11*, 690–710. [CrossRef]
16. Gastaldi, M.; Meneguzzer, C.; Rossi, R.; Lucia, L.D.; Gecchele, G. Evaluation of Air Pollution Impacts of a Signal Control to Roundabout Conversion Using Microsimulation. *Transp. Res. Procedia* **2014**, *3*, 1031–1040. [CrossRef]
17. OpenStreetMap Contributors. 2020. Available online: https://www.openstreetmap.org (accessed on 17 August 2020).
18. Mapscape. Incremental Updating. Available online: http://www.mapscape.eu/telematics/incremental-updating.html (accessed on 14 August 2019).
19. Zourlidou, S.; Sester, M. Traffic Regulator Detection and Identification from Crowdsourced Data—A Systematic Literature Review. *ISPRS Int. J. Geo-Inf.* **2019**, *8*, 491. [CrossRef]

20. Huang, S.; Lin, H.; Chang, C. An in-car camera system for traffic sign detection and recognition. In Proceedings of the 2017 Joint 17th World Congress of International Fuzzy Systems Association and 9th International Conference on Soft Computing and Intelligent Systems (IFSA-SCIS), Otsu, Japan, 27–30 June 2017; pp. 1–6.
21. Ardianto, S.; Chen, C.; Hang, H. Real-time traffic sign recognition using color segmentation and SVM. In Proceedings of the 2017 International Conference on Systems, Signals and Image Processing (IWSSIP), Poznan, Poland, 22–24 May 2017, pp. 1–5.
22. Kosonen, M.; Henttonen, K. Cheer the crowd? Facilitating user participation in idea crowdsourcing. *Int. J. Technol. Mark.* **2015**, *10*, 95–110. [CrossRef]
23. Balali, V.; Golparvar-Fard, M. Evaluation of Multiclass Traffic Sign Detection and Classification Methods for U.S. Roadway Asset Inventory Management. *J. Comput. Civ. Eng.* **2016**, *30*, 04015022. [CrossRef]
24. Hu, S.; Su, L.; Liu, H.; Wang, H.; Abdelzaher, T.F. SmartRoad: Smartphone-Based Crowd Sensing for Traffic Regulator Detection and Identification. *ACM Trans. Sen. Netw.* **2015**, *11*, 55:1–55:27. [CrossRef]
25. Merry, K.; Bettinger, P. Smartphone GPS accuracy study in an urban environment. *PLoS ONE* **2019**, *14*, e219890. [CrossRef]
26. Saremi, F.; Abdelzaher, T.F. Combining Map-Based Inference and Crowd-Sensing for Detecting Traffic Regulators. In Proceedings of the 2015 IEEE 12th International Conference on Mobile Ad Hoc and Sensor Systems, Dallas, TX, USA, 19–22 October 2015; pp. 145–153.
27. Golze, J.; Zourlidou, S.; Sester, M. Traffic Regulator Detection Using GPS Trajectories. *J. Cartogr. Geogr. Inf.* **2020**, *70*, 95–105. [CrossRef]
28. Méneroux, Y.; Guilcher, A.; Saint Pierre, G.; Hamed, M.; Mustiere, S.; Orfila, O. Traffic signal detection from in-vehicle GPS speed profiles using functional data analysis and machine learning. *Int. J. Data Sci. Anal.* **2020**, *10*, 101–119. [CrossRef]
29. Cheng, H.; Zourlidou, S.; Sester, M. Traffic Control Recognition with Speed-Profiles: A Deep Learning Approach. *ISPRS Int. J. Geo-Inf.* **2020**, *9*, 652. [CrossRef]
30. Liao, Z.; Xiao, H.; Liu, S.; Liu, Y.; Yi, A. Impact Assessing of Traffic Lights via GPS Vehicle Trajectories. *ISPRS Int. J. Geo-Inf.* **2021**, *10*, 769. [CrossRef]
31. Zourlidou, S.; Golze, J.; Sester, M. *[Dataset] GPS Trajectory Dataset of the Region of Hannover, Germany*; Institut für Kartographie und Geoinformatik: Hannover, Germany, 2022. [CrossRef]
32. Zourlidou, S.; Golze, J.; Sester, M. *[Dataset] Traffic Regulator Ground-truth Information of the City of Hannover, Germany*; Institut für Kartographie und Geoinformatik: Hannover, Germany, 2022. [CrossRef]
33. Ahmed, M.; Karagiorgou, S.; Pfoser, D.; Wenk, C. A comparison and evaluation of map construction algorithms using vehicle tracking data. *GeoInformatica* **2015**, *19*, 601–632. [CrossRef]
34. Zourlidou, S.; Golze, J.; Sester, M. *[Dataset] Traffic Regulator Ground-Truth Information for the Chicago Trajectory Dataset*; Institut für Kartographie und Geoinformatik: Hannover, Germany, 2022. [CrossRef]
35. Given, L.M. *Naturalistic Data*; SAGE Publications: Thousand Oaks, CA, USA, 2008; p. 547. [CrossRef]
36. Mapillary. Mapillary: A Street-Level Imagery Platform. 2022. Available online: https://www.mapillary.com/ (accessed on 20 April 2022).
37. Palma, A.T.; Bogorny, V.; Kuijpers, B.; Alvares, L.O. A Clustering-based Approach for Discovering Interesting Places in Trajectories. In *Proceedings of the 2008 ACM Symposium on Applied Computing*; SAC '08; ACM: New York, NY, USA, 2008; pp. 863–868. [CrossRef]
38. Comito, C.; Falcone, D.; Talia, D. Mining human mobility patterns from social geo-tagged data. *Pervasive Mob. Comput.* **2016**, *33*, 91–107. [CrossRef]
39. Niu, X.; Wang, S.; Wu, C.Q.; Li, Y.; Wu, P.; Zhu, J. On a clustering-based mining approach with labeled semantics for significant place discovery. *Inf. Sci.* **2021**, *578*, 37–63. [CrossRef]
40. Spaccapietra, S.; Parent, C.; Damiani, M.L.; de Macedo, J.A.; Porto, F.; Vangenot, C. A Conceptual View on Trajectories. *Data Knowl. Eng.* **2008**, *65*, 126–146. [CrossRef]
41. Kang, J.H.; Welbourne, W.; Stewart, B.; Borriello, G. Extracting Places from Traces of Locations. In *Proceedings of the 2Nd ACM International Workshop on Wireless Mobile Applications and Services on WLAN Hotspots*; WMASH '04; ACM: New York, NY, USA, 2004; pp. 110–118. [CrossRef]
42. Feuerhake, U.; Kuntzsch, C.; Sester, M. Finding interesting places and characteristic patterns in spatio-temporal trajectories. In *Proceedings of the 8th International Symposium on Location-Based Services*; Forschungsgruppe Kartographie: Wien, Austria, 2011.
43. Wu, T.; Shen, H.; Qin, J.; Xiang, L. Extracting Stops from Spatio-Temporal Trajectories within Dynamic Contextual Features. *Sustainability* **2021**, *13*, 690. [CrossRef]
44. Ester, M.; Kriegel, H.P.; Sander, J.; Xu, X. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD ' 96), Portland, Oregon, 2–4 August 1996; pp. 226–231.
45. Tang, L.; Kan, Z.; Zhang, X.; Yang, X.; Huang, F.; Li, Q. Travel time estimation at intersections based on low-frequency spatial-temporal GPS trajectory big data. *Cartogr. Geogr. Inf. Sci.* **2016**, *43*, 417–426. [CrossRef]
46. Carisi, R.; Giordano, E.; Pau, G.; Gerla, M. Enhancing in vehicle digital maps via GPS crowdsourcing. In Proceedings of the 2011 Eighth International Conference on Wireless On-Demand Network Systems and Services, Bardonecchia, Italy, 26–28 January 2011; pp. 27–34. [CrossRef]

47. XGBoost Python. XGBoost Python Library. 2022. Available online: https://xgboost.readthedocs.io/en/stable/python/index.html (accessed on 15 Feburary 2022).
48. Krisp, J.M.; Keler, A. Car navigation–computing routes that avoid complicated crossings. *Int. J. Geogr. Inf. Sci.* **2015**, *29*, 1988–2000. [CrossRef]